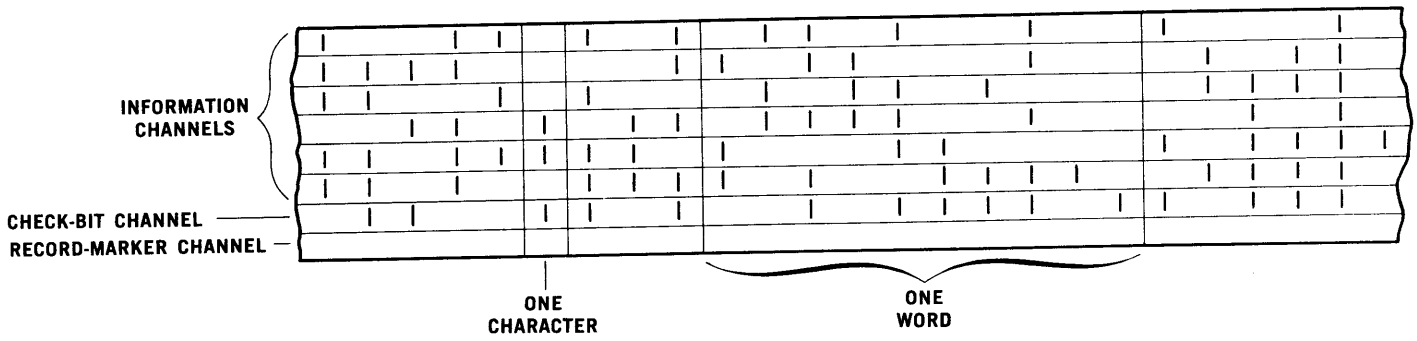
# MAGNETIC FILE OPERATIONS

The National 304 System, in common with all large-scale Data Processing systems, uses Magnetic Tape as its File medium. Information is recorded on Magnetic Tape in serial form; that is, successive characters are recorded one after another, with each character being represented by a row of magnetized spots (called "bits") across the width of the tape. Each of these spots may hold either positive or negative magnetization, and the pattern of bits in each row defines the character recorded in that row.

The tape is divided along its length into 8 Channels, or tracks. Six Channels contain the six bits which define each character; the 7th Channel contains the check-bit, or parity-bit, which is recorded with each character, and which functions as one of the self-checking features of the Magnetic Tape System; the 8th Channel contains control information (Record-Markers).

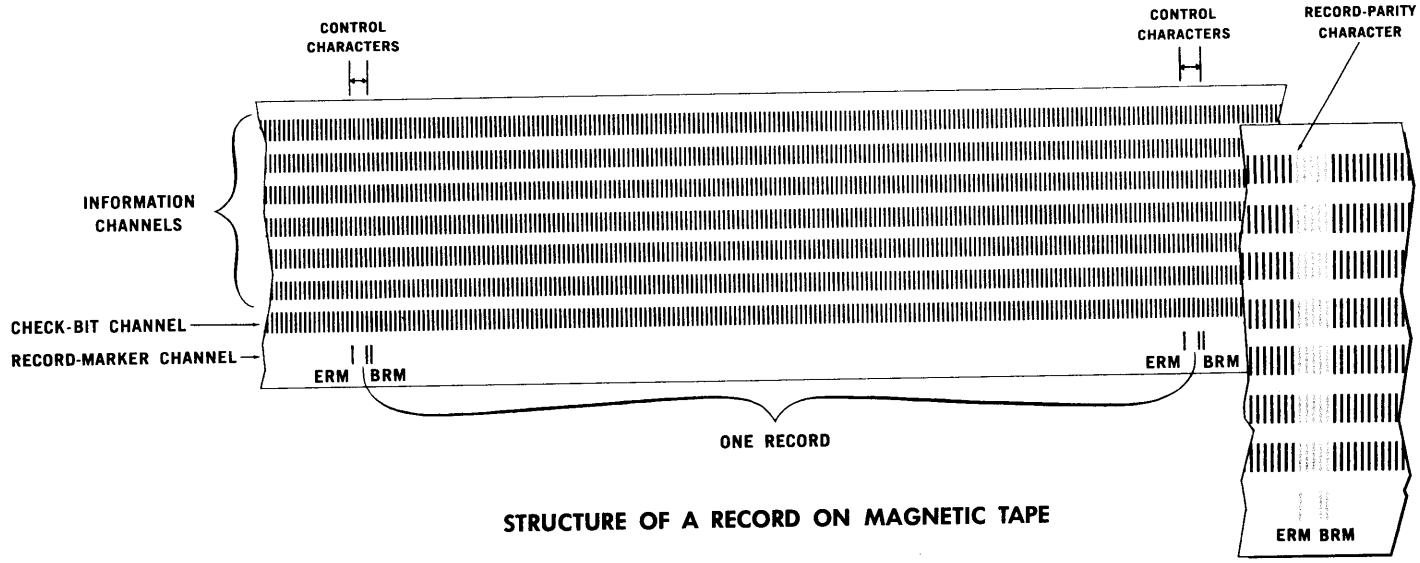
Throughout the NCR 304 System, information is regarded as organized into "Words". A Word is simply a grouping of 10 alphanumeric characters, and each successive 10 characters on the tape, therefore, constitute one Word of information.




## REPRESENTATION OF INFORMATION ON MAGNETIC TAPE

However, the major organization of information on Magnetic Tape is into Records. A Record may contain any number of Words from 10 to 100, and is bounded on the tape by a Beginning-of-Record Mark (BRM) and an End-of-Record Mark (ERM).

It is important to recognize that a Record is defined by its *physical* characteristics and not necessarily by the information it contains: It is a group of 10 to 100 Words recorded on Magnetic Tape, bounded by a BRM and an ERM. While one record will most often contain one File-Item (one Account, for example), convenience may suggest that several Items be included in a single Record, or that the information pertaining to one Item be recorded as several Records.




STRUCTURE OF A RECORD ON MAGNETIC TAPE

The Record illustrated on the previous page, consisting of 170 alphanumeric characters, when drawn to actual scale, would look like this:

Since, by the very nature of the medium, the speed of reading and recording Magnetic Tape is lower, by a substantial factor, than the operating speed of a Data Processor, it is of the utmost importance that information transfer rates to and from Magnetic Tape be as high as possible. The NCR 304 System has achieved extremely high Magnetic Tape speeds, but what is even more significant, it has increased the efficiency of the recording technique, so that for the first time in any Data Processing System, the *effective* speed of information transfer is almost identical with the *nominal* speed.

This efficiency has been accomplished by:

- Reducing the delays inherent in any Magnetic Tape System (Saving Time).
- Eliminating blank areas in which no information is recorded on the tape (Saving Space). Any blank area on the tape must be passed over in order to reach the information beyond, and therefore, blank spaces would also represent lost time within the System.
- Full Alphanumeric Recording. Since 6 bits (magnetic spots) are used to record each character, the stated information transfer rate in the NCR 304 System applies to alphanumeric characters, and not merely to numeric digits. However, digits may be recorded in condensed form, in which case the effective information transfer rate is increased by 50%. (See Item 4 under "Saving Space".)

### *SAVING TIME:*

Since Magnetic Tape can be read or recorded reliably only when moving at full speed, there must be an acceleration delay every time the tape is started.

In the NCR 304 System, not only has this acceleration time been reduced to a very small figure, but the necessity for frequent stops and starts has been largely eliminated. Instead of reading or recording only one Record at a time, a single operation will read or record up to 99 Records, of either fixed or varying lengths, as a single continuous flow of information. All the information read or recorded with a single Magnetic Tape operation is called a "Gulp" of information.

The use of Index Registers (described in Chapter III) makes it particularly convenient to handle the successive Records of a Gulp within the Processor, without the time-consuming necessity of moving each File Record in turn to some standard location in Memory.

### *SAVING SPACE:*

1) *Variable-length Records* permit each record to be only as long as its information content requires it to be. All blank words within the Record have been eliminated.

2) *"Father-son" expanding file technique* completely transcribes the File as a by-product of every updating "run". The principal advantages of this technique are:

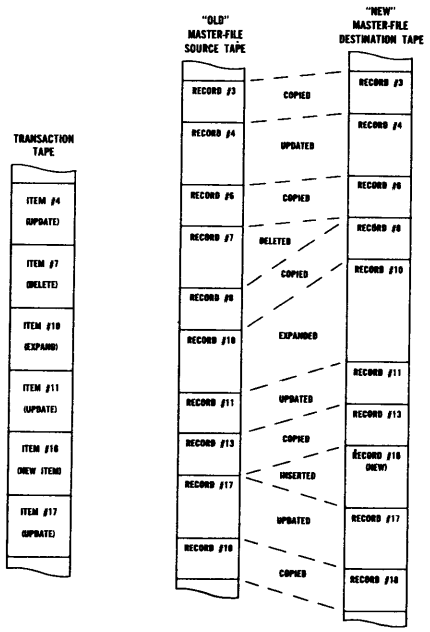
a) Permits new Records to be inserted, or obsolete Records to be deleted, wherever required, and still keeps the file "closed up tight" at all times.

b) Permits the length of any individual Record to be expanded or contracted during the transcription, as the varying information content of the Record may require.

c) Since "yesterday's file" is retained unchanged after the updating run which creates "today's file", it is always available as a rescue point from which "today's file" can be re-created in case of damage to the Magnetic Tape.

d) The COPY operation gives simultaneous READ-WRITE-COMPUTE over inactive Records in the file, and is particularly effective in file posting when, typically, a minority of the Records are active in any one day. COPY is simple to program, and affords convenient, effective time-sharing while passing over the inactive Records.

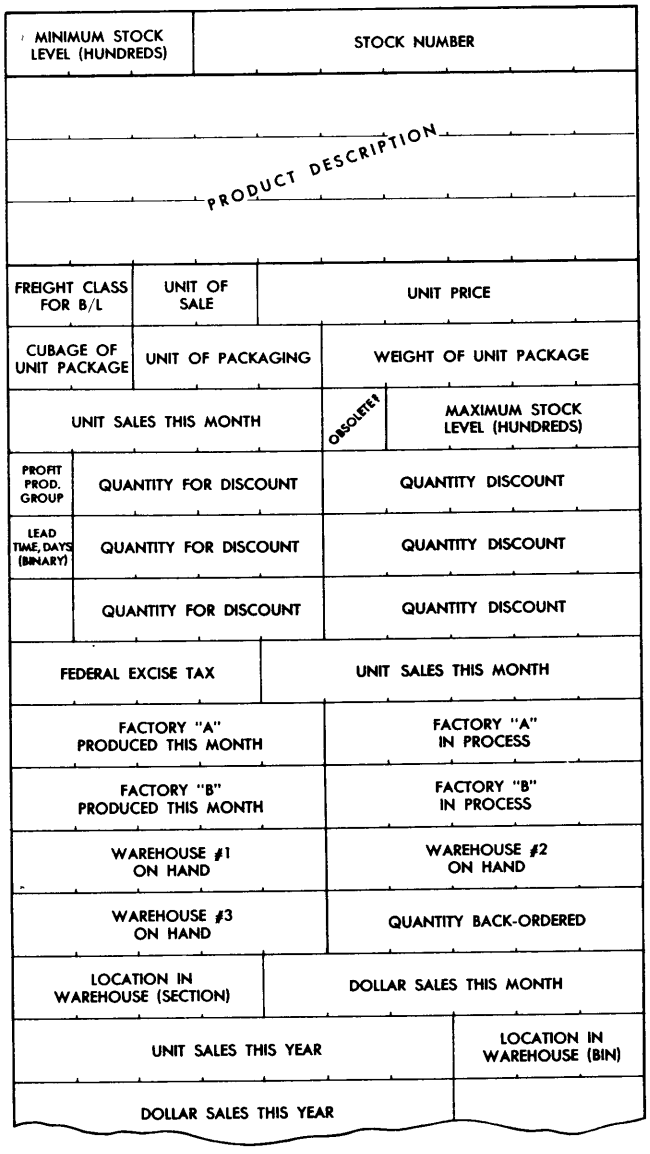
COPY operates as a continuous flow of information from "father" tape to "son"; both tapes move from one active Record to the next with no intermediate stops, and with no supervision by the Processor.




**UPDATING AN EXPANDING AND CONTRACTING FILE**

**(SAVING SPACE:)**

3) *Partial-Word Operation.* In order to perform computation within a Data Processor, it is conventionally necessary to have each information field (Account Number, Transaction Code, Unit Price, On-Hand Balance, etc.) isolated in a separate Word since Processors, in general, can operate only upon complete Words. However, most information fields require less than 10 characters, and it would be an intolerable waste of space on Magnetic Tape to record, for example, a 7-digit Account Number, a 2-digit Code, a 5-digit Quantity, each as a 10-character Word in the File. Therefore it is common practice, in using Magnetic Tape systems, to combine several information fields into a single Word for recording on Magnetic Tape. Such a File Record, when ready for recording, might look like this:




**EXAMPLE OF A RECORD IN A PRODUCT MASTER FILE**

When a File Record is read into the Processor, these combined fields must be split apart into separate Words in order to operate on them; then the fields must be recombined so that the updated Record may be recorded back on Magnetic Tape.

In the NCR 304 System, this process of combining several information fields into a single Word is fully automatic, with no restriction on the number of fields within a Word, nor on their placement within a Word. Since any Partial-Word field may be directly referred to by any operation, the programmer may use this combining technique to its fullest extent, yet pay no penalty in either programming effort, or Processor operating time for:

- Extracting each information field in order to isolate it;
- Shifting one or both fields in order to align them with each other, so the desired operations can be performed;
- Shifting the result in order to align it with the desired putaway field;
- Testing the result to be sure it is not too large to fit into the putaway field (overflow test);
- Inserting the result into the putaway field;
- Tagging the putaway field with the proper algebraic sign.

The Partial-Word operation of the Processor is described in detail in Chapter III.

4) *Condensed Recording of Numeric Information.* Although 6 bits are required to record one alphanumeric character, 4 bits would be sufficient to record one numeric digit. Since a large portion of the information in business files is purely numeric, the use of 6 bits for every character would mean that, for this numeric information, one-third of the recorded bits (and therefore one-third of the Magnetic Tape) would contain no useful information, and would therefore, in effect, be blank.

In the NCR 304 System, the programmer is able to record numeric information in a condensed, or PACKED, form in which only 4 bits are used for eacn digit. Three condensed digits, therefore, occupy only as much Magnetic Tape as two alphanumeric characters, and can be read or recorded in the same amount of time as two alphanumeric characters.

A single PACK operation will condense any desired amount of numeric information (with an automatic alarm if alphabetic information is inadvertently included), and a single UNPACK operation will later expand the condensed information to its original form, so that arithmetic operations can be performed on it.

However, when updating a File Record, the programmer need UNPACK only as much of the Record as is subject to arithmetic operations. All comparing

***(SAVING SPACE:)***

and testing operations can be performed on the numeric data while it remains in condensed form.

5) *Binary Operations.* While the NCR 304 Processor operates in the decimal number system, it also offers the programmer a set of extremely flexible binary operations, which permit access to the individual bits within an information field. The programmer can, therefore, achieve a high degree of compactness in recording certain types of coded information in a Magnetic Tape File. (See EXTRACT, INSERT, ADD BINARY, COMPLEMENT BINARY, TEST BIT, in Chapter IV.)

Thus, for example, a "State" code (including the 50 States, Puerto Rico, and perhaps the Provinces of Canada — in fact, up to 64 different "States") can be stored as a single character in each customer's File Record; or a calendar date (month, day, year) can be stored in three characters in the File Record. In a payroll file, the presence or absence of each of 30 different optional pay-deductions can be stored in 30 bit-positions (5 characters) in each employee's File Record.

6) *No Inter-Record Gaps.* The ability to read and record Magnetic Tape continuously, in Gulps of Records, would be of relatively little value if there were still a substantial gap (blank space) after every Record in the file, to allow for the possibility that *any* Record might be the last one of a Gulp. Conventional recording techniques require a large enough blank space after each Record, so that the tape may be stopped there, and still have enough space to come up to full speed again before reaching the next Record.

In the NCR 304 System, there are no gaps between records on Magnetic Tape. Therefore, whenever a Gulp ends, the tape coasts to a stop with the Magnetic Head within the next Record. Special repositioning circuitry in the Tape Controller immediately assumes control of the Tape Handler, backs up the tape until the BRM-ERM is reached, and then stops the tape again, with the Magnetic Head now within the last Record of the Gulp, ready for a "flying start" at the next Record.

The Processor is released from the Magnetic Tape System as soon as the Gulp itself is finished, and immediately proceeds to process the information, while the Tape Controller performs the repositioning as a "time-shared" operation. Extensive self-checking circuitry guarantees the integrity of the repositioning, and the operation is completely inter-locked so that the Processor cannot perform another operation involving that Controller until repositioning has been completed. The repositioning takes so little time that, in practice, the Processor rarely has occasion to wait for it.

## MAGNETIC TAPE INSTRUCTIONS

***WRITE TAPE:***

The recording of information from Processor memory onto magnetic tape is performed by the WRITE TAPE instruction, which writes one or more records as a single operation. As each record is written on the tape, it is automatically bounded by a BRM (Beginning-of-Record-Mark) and an ERM (End-of-Record-Mark); thus even though a number of records are written by a single instruction, they remain separate entities, and may later be searched for or read independently and selectively.

All the records written by a WRITE TAPE instruction may be of the same length ("fixed" length), or they may be of different lengths ("variable" length).

Thus the abilities of WRITE TAPE are:

1) Write 1 to 99 fixed-length records at a Gulp, all records of the same length, 10 to 100 words.
2) Write 1 to 99 variable-length records at a Gulp, each record may be of a different length and may be 10 to 100 words long.

***READ TAPE:***

The reading of information from magnetic tape into Processor memory is performed by the READ TAPE instruction, which reads 1 to 99 records at a Gulp, either fixed-length or variable-length records.

A variation of the READ TAPE instruction will read only a selected portion of each record, while the remaining information will not enter the Processor memory.

Other variations (Index Forward, Index Backward) will move a magnetic tape forward or backward 1 to 99 records without *any* of the information entering the Processor memory.

***WRITE-COPY:***

When updating a magnetic tape file, the transactions are always sorted into the same account-number sequence as the records in the file. Therefore, when an active record has been read into the Processor memory from the "father" tape, and all transactions affecting that account have been posted to it, and the updated record has been written on the "son" tape, the System must then start reading all the subsequent records on the "father" tape. These records must be examined one by one, and each of them must be compared with the next transaction to see if it is affected by that transaction (i.e., if that account is active today). As each record is found to be inactive, it must be written unchanged on the "son" tape, and the process is repeated over and over until the next active account is reached.

Tape-to-tape COPY transcribes the inactive records in a file from "father" tape to "son" as a single con-

tinuous operation, *without reading those records into the Processor memory.* During this transcription, each record is examined, so that the operation terminates when it reaches the next active account.

The functions of the WRITE-COPY instruction are:

1) It WRITES one record from memory. This record will previously have been read into memory and updated there; in its changed form it must now be recorded on the "son" tape. This step is omitted if the record is to be deleted from the file.
2) Immediately after that record has been written, the Processor tells the Magnetic Tape Controller which record is desired next, and thus *initiates* the off-line COPY from "father" to "son". The Processor is now free to proceed with any other work which does not use that Controller, while the Controller is looking for the next active record and transcribing all intervening inactive records from "father" to "son".

The Controller conducts the COPY independently of the Processor, moving both tapes continuously, and transcribing character by character from "father" to "son" while the two tapes are in motion. As the information flows through it, the Controller examines each record to determine if it is the desired one. When that record is reached, the Controller stops both tapes and repositions them so that the "father" is ready to *read* that record into the Processor memory, and the "son" is ready to *write* that record from the Processor memory after it has been updated. While the COPY is going on, the Controller reports "busy" so that the Processor continues with its alternate program until the COPY is finished.

Any of the first 8 words of each record may be a "key word" containing Account Number or other identification, and the Controller will examine any desired selection of characters within the key word, ignoring all other characters. Further, the key word may be divided, with the Controller testing two different keys simultaneously. A common use of this feature considers Account Number (up to 10 digits condensed into 7 characters) as one key, and "next-action" Date (day, month, year, century, coded into 3 characters) as the other. The Account Number is compared with the next transaction, while the Date is compared with today's date, so that COPY finds all accounts for which transactions were received today, *and also* all accounts requiring some action (billing, follow-up, expiration, etc.) today.

Two types of test may be used with COPY:

*Equality* COPY will terminate when it reaches the desired record.

*Range* COPY will terminate when it reaches either the desired record, or else a record whose key is *greater* than the one desired. The latter case occurs whenever the desired record is not in the file, which usually means that a new account has been opened, and a new record must be inserted in the file.

When the key word is divided, either side of the key may be tested for Range or for Equality, independently of the kind of test being performed on the other side.

### SEARCH:

A variation of COPY uses only the "father" tape, and therefore accomplishes off-line SEARCH of magnetic tape.

### WRITE-COPY-READ: (on-line Copy)

This operation is essentially the same as WRITE-COPY, *together with* the subsequent READ, executed as a single continuous operation. It requires supervision by the Processor itself, and is used whenever the particular problem does not offer any useful work for the Processor to do during the COPY. In that case, terminating the COPY with the desired record in the memory elminates all delay between the COPY and the READ.

### SEARCH-READ:

A variation of COPY-READ uses only the "father" tape, and therefore accomplishes on-line SEARCH-READ of magnetic tape.

### REWIND:

The rewinding of any reel of magnetic tape is *initiated* by the REWIND instruction. The actual rewinding is performed entirely by the Magnetic Tape Handler, so that both the Processor and the Controller are immediately free to do other work. Any number of Handlers may be rewinding at the same time without interfering with the rest of the System.

A variation of the REWIND instruction will place a USE LOCKOUT on the Handler so that the Processor cannot use it again until its reel of tape has been changed.

## MAGNETIC TAPE HOUSEKEEPING

The National 304 Magnetic Tape System, together with the subroutine called **STEP** (Standard Tape Executive Program), relieves the programmer of all tape "housekeeping" problems.

The Magnetic Tape System automatically *detects,* and automatically *identifies,* ALL housekeeping conditions (such as Busy, Use Lockout, Error, End of Tape, End of File, etc.) and automatically establishes appropriate branches in the program. For all routine matters these branches lead to **STEP**, which takes care of the situation and then returns to the main program without attention from the programmer. Thus, for example, when a Handler reaches the end of a reel of tape, **STEP**:

- Rewinds that Handler;
- Types-out Controller and Handler number on the Console Typewriter;
- If Source Tape, types-out to indicate which reel of tape must next be mounted on that Handler;
- If Destination Tape, types-out to indicate what identification should be placed on the outside of the reel;
- Changes the entire main program to refer to an alternate Handler for the next reel in that file;
- If Source Tape, checks the first record on the next reel (containing tape-identification) to be sure the *correct* reel has been mounted;
- If Destination Tape, checks the first record on the next reel to be sure the tape is obsolete and is *safe* to write on. Then **STEP** writes a new first record on the tape to identify the information about to be recorded on it;
- Returns to the main program, which proceeds as though nothing had happened.

**STEP** corrects errors in recording or reading tape, and it identifies defective spots on the tape and skips over them, all without attention from the programmer or the operator.

**STEP** also brings in Overlay Programs on demand, and automatically steps the Processor from program to program.

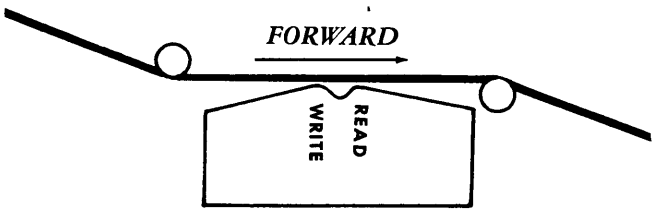
Out of all possible housekeeping circumstances, the programmer must plan for Busy, if he wishes to utilize the system's ability to perform other operations simultaneously; and he must plan for End of File, because that means the program is finished and the Processor must proceed to the next job. Aside from these, he writes his programs as though all magnetic tapes were endless, and as though all other housekeeping problems could never arise, because the Magnetic Tape System and **STEP** detect, identify, and correct all those conditions.

## GUARANTEES OF TAPE ACCURACY

In any Electronic Data Processing System the *only* way to be assured that the recording on magnetic tape is as it was intended to be, is to READ the tape and test its correctness. Further, this reading should be done at the *one* time when an error, if it should occur, can be corrected most easily. That time is immediately after the information has been recorded, since only then is the correct information still available in the Processor memory.

The National 304 System automatically reads, and checks, the magnetic tape *during the recording process.* If a recording error should occur, it will be discovered immediately, and may be corrected automatically without guesswork, and without operator intervention. This procedure guarantees, at the time of recording, that all tapes are recorded without error.

The read-checking of magnetic tape recording is performed by a reading head located on each Handler, immediately behind the writing head:




As the newly recorded information passes under the reading head, it is read and immediately checked by the Controller. The self-checking system is extremely comprehensive, and consists of five basic checks, so organized that any possible source of error is covered by at least two independent checking methods:

### *1. CHARACTER PARITY:*

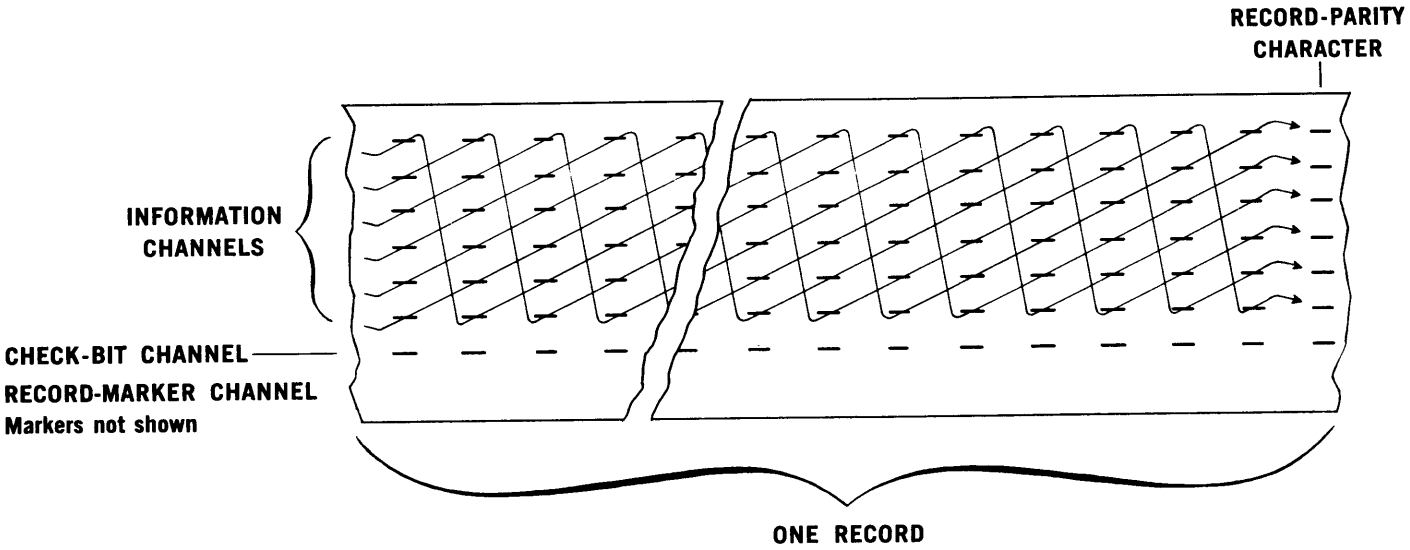
Each character is recorded with a parity bit. This seventh bit is assigned automatically, and is chosen so that the seven bits of each recorded character always contain an odd number of 1-bits.

This check protects against the garbling of individual bits within the characters of the record.

### *2. RECORD PARITY:*

One extra character is automatically recorded with each record. The Controller sets up the individual bits of this character so that there is always an odd number of 1-bits along each of the diagonal arrows shown in the diagram.

This furnishes independent protection against garbling of bits, and also protects against the omission of complete characters or the accidental recording of additional, spurious characters.




SELF-CHECKING OF INFORMATION ON MAGNETIC TAPE

### 3. *CHARACTER COUNT:*

Each record must contain an exact multiple of 10 characters, plus the record-parity character.

This is a consequence of always recording complete words from memory, and furnishes important independent protection against dropping or adding characters on the tape.

### 4. *ERM-BRM ALTERNATION:*

Whenever a tape is in motion, the record-markers must always show perfect alternation between ERM and BRM, to guarantee the boundaries of each record.

Even if exactly the three bits of a record-marker pair (and no other bits in that channel) should be accidentally changed, checks 2 and 3 each furnish independent detection of this condition.

### 5. *TIMING CHECK:*

Whenever a tape is started into motion, the Controller sets a rigid time-limit within which the Handler must detect the first record-marker. This check is made at the writing head when writing, and at the reading head when reading.

When the record-marker is detected within the time-limit, it is certain that the very next record on the tape is the one which was written or read.

### 5a. *FIRST-INFORMATION CHECK:*

Whenever a Handler is about to read or write the first record on a reel of tape, the Timing Check is not appropriate, since it is desirable to leave several feet of blank tape at the beginning of the reel, and this would cause the time-limit to be greatly exceeded. Therefore, when the Handler senses that the tape is positioned on the leader, the Timing Check is automatically replaced by the First-Information Check.

When writing at the beginning of a tape, the writing head erases the first three feet of the tape, and then writes the first record. The reading head must then find that a record-marker is the *first information* on the tape. Nothing, not even recording so faint that it is detected only as undecipherable "noise", may appear on the tape before the first record-marker.

These five checks are made by the reading head while the writing process is going on, and they guarantee that all tapes are correctly recorded.

The same five checks are also made during the reading process, and guarantee that all tapes are correctly read. During a Copy, these checks are made on *both* the "father" and "son" tapes.

The Timing Check, and the ERM-BRM Alternation Check, are also made on all repositioning operations.

## INTERLOCKS

### 1. *WRITE LOCKOUT:*

Every Magnetic Tape Handler is normally in a Write Lockout state, in which the recording circuits are completely disconnected. In order to engage the writing circuits, and permit information to be recorded on a reel of magnetic tape, it is necessary to affix a "write-permissive" ring to that reel.

In the absence of this ring, it is impossible, whether through operator error, programmer error, or equipment malfunction, to write on and thereby destroy a tape whose information is still useful.

If a ring is accidentally left on a reel after it is removed from the Handler, there will be no ring available to place on the next reel; further, the reel will not fit into its plastic storage-box if the ring is still attached to it.

### 2. *USE LOCKOUT:*

When the end of a tape has been reached, and the rewind has been started, the program may impose a Use Lockout on that Handler. This prevents any use of that Handler until the tape has been completely rewound, and an operator has mounted a new reel.

### 3. *HANDLER DOOR LOCKED:*

The door to the Handler cabinet, furnishing access to the reel of tape, is always locked by an electrically-operated bolt, except when the Handler is in Use Lockout *and* the tape is not moving. Therefore the operator is denied access to the tape except when the program has finished with it. Even during a rewind with Use Lockout, the door remains locked until the rewind is completed.

### 4. *MOUNTING OF TAPES:*

Unless a reel of tape is correctly mounted, and the cabinet door properly closed, it is impossible for the operator to remove the Use Lockout from the Handler.

### 5. *NO TAPE, BROKEN TAPE:*

If the Processor attempts to use a Handler which does not have a tape mounted on it, or if a tape should break, an immediate error-halt occurs.

### 6. *LONG LOOP, SHORT LOOP:*

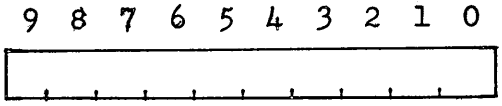
In the event that either of the tape loops in the vacuum chambers should grow, or shrink, beyond the capacity of the servo system to correct the condition, there would be a danger of breaking or of crumpling the tape. Special independent circuitry detects this condition before damage occurs; the tape drive is immediately released, the brakes are set on the reels, and the Handler error-halts.

# GENERAL CHARACTERISTICS OF THE PROCESSOR

A fundamental characteristic of any Electronic Data Processor is its internal information-storage, or "Memory", in which it is able to store both that part of the data being operated upon at the moment, and the program for processing that data. The program is a sequential list of Instructions, each of which (such as ADD, COMPARE, READ MAGNETIC TAPE) represents a single operation to be executed by the Processor.

The NCR 304 Data Processor is available with a Memory of either 2000 or 4000 permanently-numbered storage locations, or "Cells", which contain stored information. The number assigned to each Cell is the "Address" of that Cell. In addition to the basic 2000 or 4000 Cells of Memory, there are, respectively, 400 or 800 "Special" Cells, outside the Main Memory package, which are all individually addressible by any Instruction. These "Special" Cells are described, with the limitations and conventions governing their use, in Appendix "C" to this Chapter.

The contents of each Cell is a "Word" of 10 characters, and the character-positions within a Word are conventionally numbered from right to left:




Information is stored in Memory by means of Magnetic Cores, which are tiny rings of ferrite material, strung on a lattice of wires. Each core may be selectively magnetized in either of two states which, for convenience, are designated "0" and "1". The symbols "0" and "1" are not numbers. They are merely convenient marks used to distinguish the two

states of a single Magnetic Core. Any other pair of conventional symbols would serve as well, for example:

"N" and "S"  "." and "-"  "↑" and "↓"  "o" and "/"  "+" and "-" .

However, "0" and "1" are in common use, and we shall use them in this book, subject to the caution that they must not be thought of as numbers. The marks "0" and "1", corresponding to the two possible magnetized states of one core, are called "bits", and therefore a single core may store either a "0-bit" or a "1-bit".

Six cores are assigned to each character-position of each Cell in the Memory, and characters are spoken of as being represented by a 6-bit code. Since each bit has only two "values", the bit-configuration of a character is called the "binary" (2-way) representation of that character. There are 64 possible combinations of the "0" and "1" states of each group of 6 cores, corresponding to the 64 different characters in the "language" of the Processor. A character may be a numeric digit, a letter of the alphabet, or one of 28 symbols.

It is convenient, when speaking of the 6-bits which make up a character, to classify them into "zone bits" and "numeric bits", and to write them in the following pattern:

| zone | numeric |
|---|---|
| 00 | 0000 |

The 64 characters of the Processor language, and their respective 6-bit binary configurations are:

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

Every one of the 64 possible configurations of 6-bits corresponds to a legitimate character in the Processor language. However, 56 of these (called the "Data Characters") are sufficient for all business data requirements. The remaining 8 characters, lower-case m n ø p u v w x, are simply convenient labels for their respective bit-configurations, and are referred to as "Non-Data Characters".

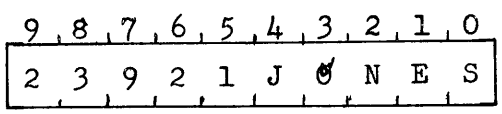
## REPRESENTATION OF INFORMATION

Information stored in Memory may constitute either Instructions or Data. There is no absolute distinction between the two types of information, since one of the fundamental properties of an Electronic Data-Processor is its ability to operate upon its own Instructions, and modify them as though they were Data.

INSTRUCTIONS: The program for processing the Data is stored in Memory as a list of Instructions, which the Processor executes in sequence. An Instruction may specify that a certain operation be performed upon Data, or that a test of some sort be performed. Depending upon the result of the test, the Processor may continue to execute Instructions in the normal sequence, or break the sequence and start a new sequence elsewhere in the program. Breaking the normal sequence of Instructions is called "Branching" or "Jumping".

A substantial number of the Instructions in the 304 System serve both purposes; they may perform an extended series of operations upon Data, and then test the result as well.

DATA: The basic unit of Data within the Processor is the Field, which occupies all, or any continuous portion, of a Word. A Field is specified by naming the address of the Word containing it, and the left-most and right-most character-positions occupied by the Field within the Word. Thus, suppose that the contents of some Cell in Memory is:




Then: the number 2392 occupies the 96-field of that Cell,
the number 3921 occupies the 85-field of that Cell,
the name JONES occupies the 40-field of that Cell,
the number 1 occupies the 55-field of that Cell.

The Data stored in a Field may take one of three forms:

1) ALPHABETIC INFORMATION:

This is sometimes referred to as "alphanumeric" information, and may consist of any combination of the 64 digits, letters, and symbols, which make up the Processor language.

2) NUMERIC INFORMATION with algebraic sign:

A Numeric Field consists of digits, representing a number. The algebraic sign of that number is combined with the left-most digit of the Field, and is specified by the right-hand zone bit of that digit. If this "sign-bit" is a 0-bit, the algebraic sign of the entire Field is positive; if the "sign-bit" is a 1-bit, the algebraic sign of the Field is negative.

Consequently, the left-most character of a Numeric Field always contains two pieces of information: the sign of the Field (specified by the sign-bit), and the first digit of the Field (specified by the numeric bits). In all arithmetic operations, the Processor automatically handles the signs in the proper manner, and properly stores the sign of the result; the programmer need never concern himself with the problem of storing algebraic signs, or even of allocating storage-space for the signs.

Consider the number 942 stored in a 5-character field; it will appear as 0 0 9 4 2 where the first "0" has the double significance "positive" (sign-bit) and "zero" (numeric bits). If the same number were stored in a 3-character field, it would appear as 9 4 2 where the "9" has the double significance "positive" and "9". (Refer to the Language Code Table for the binary configurations of "0" and "9").

Consider the number -174 stored in a 5-character field; it will

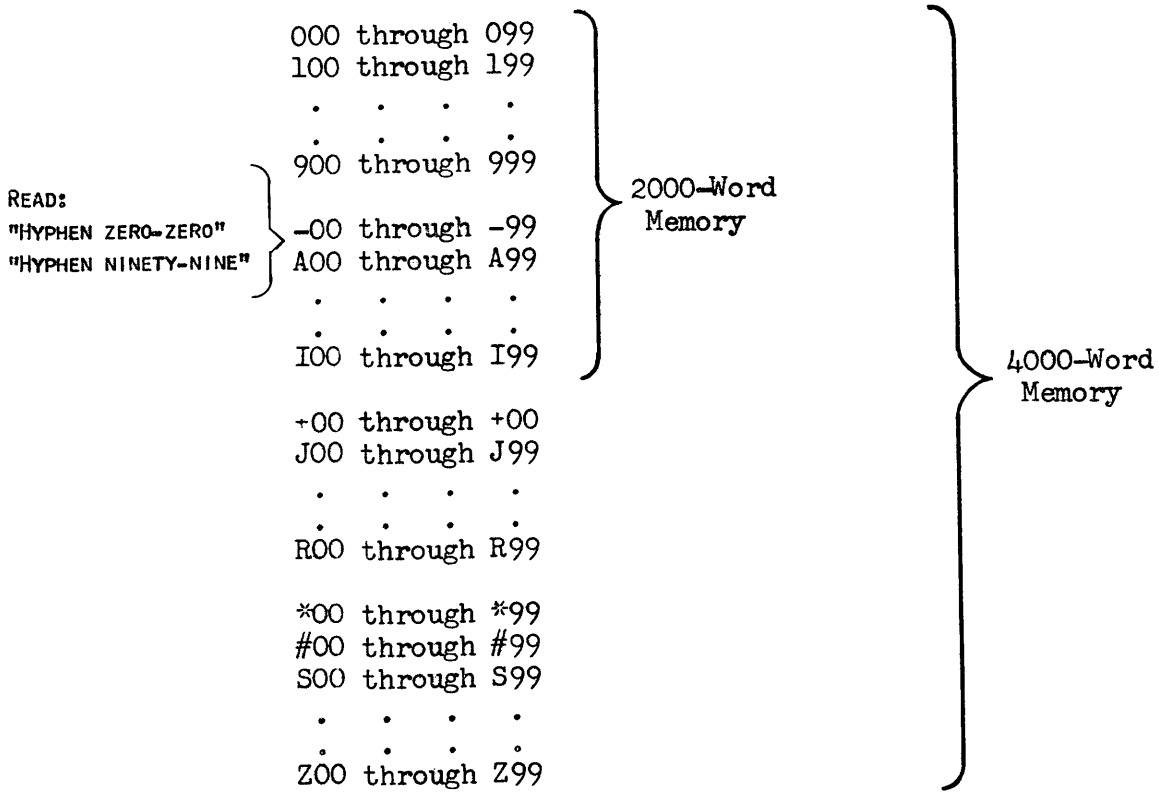
appear as - 0 1 7 4 where the hyphen "-" has the double significance "negative" (sign-bit) and "zero" (numeric bits). If the same number were stored in a 3-character field, it would appear as A 7 4 where the "A" has the double significance "negative" and "1". (Refer to the Language Code Table for the binary configurations of "-" and "A".)

Since the combined representation of "positive" and "zero" is the character "0" (zero), and the combined representation of "negative" and "zero" is the character "-" (hyphen), all input and output is conveniently performed in conventional format; that is, positive numbers preceded by no sign (but note that if "+" is used, it is also interpreted arithmetically as meaning "positive" and "zero"), and negative numbers preceded by a hyphen.

Appendix "A" to this Chapter discusses the disposition of redundant zone bits during arithmetic operations, and also shows the results of performing arithmetic operations upon characters whose numeric bits do not correspond to any of the 10 decimal digits.

ADDRESS-TYPE INFORMATION:

The Cells of Memory are numbered:




In order to perform arithmetic operations on Addresses, a special kind of addition and subtraction, called MODIFY ADD and MODIFY SUBTRACT, have been included among the Processor operations; the symbols ⊕ and ⊖ are used to distinguish them from ordinary addition and subtraction.

Subtracting one Memory Address from another gives the number of Cells separating them, expressed as an "Address-Type Number". In order for the programmer to duplicate the Processor's result of a MODIFY ADD or MODIFY SUBTRACT, or for him to interpret the Address-Type answer, he may regard any Address-Type Number as a condensed form of a 4-digit number from 0000 to 3999, in which the left-most two digits have been combined into the zone bits and the numeric bits of a single alphanumeric character. The zone bits of this character specify the first digit, according to the column "Numeric

Value of Zone Bits" in the Language table, and the numeric bits of this character specify the second digit. Thus, for example, the Address-Type number G55 may be interpreted as 1755. The zone bits of "G" have the numeric value "1", and the numeric bits of "G" are the numeric bits of "7". Several other examples follow:

| Address-Type Number | Interpreted As |
|---|---|
| 001 | 0001 |
| L07 | 2307 |
| 699 | 2699 |
| X21 | 3721 |
| Z98 | 3998 |

The following rules, then, will enable the programmer to duplicate the answer which the Processor obtains as the result of a MODIFY ADD or MODIFY SUBTRACT operation:

1) Interpret the two Address-Type Numbers which are to be added or subtracted, as 4-digit numbers.

2) Perform the addition or subtraction.

3) If the result is greater than 3999, subtract 4000 from it. If the result is negative, add 4000 to it.

4) Translate the result back into an Address-Type Number.

EXAMPLES: 998 ⊕ 004 = -02

THIS RESULT IS NOT "MINUS 02"; IT IS "HYPHEN ZERO TWO".

998 ⟶ 0998
⊕ 004 ⟶ + 0004
1002
-02 ⟵

H50 ⊕ 305 = J55

H50 ⟶ 1850
⊕ 305 ⟶ + 0305
2155
J55 ⟵

F21 ⊕ J44 = X65

F21 ⟶ 1621
⊕ J44 ⟶ + 2144
3765
X65 ⟵

EXAMPLES: (Cont'd)

P39 ⊕ *41 = G80

P39 ⟶ 2739
⊕ *41 ⟶ + 3041
5780 (greater than 3999)
- 4000
1780
G80 ⟵

P39 ⊖ *41 = W98

P39 ⟶ 2739 ⟶ 6739
⊖ *41 ⟶ - 3041 ⟶ - 3041
negative 3698
W98 ⟵

B83 ⊕ Z99 = B82

B83 ⟶ 1283
⊕ Z99 ⟶ + 3999
5282 (greater than 3999)
- 4000
1282
B82 ⟵

It will be seen that the operations of MODIFY ADD and MODIFY SUBTRACT are "cyclic modulo 4000"; that is, the address following Z99 is 000, and in performing arithmetic, the addresses may be regarded as being arranged in a circle. Any counting process which crosses one "end" of the Memory picks up without a break at the other "end" as shown in the previous examples. Note also that there are no negative numbers in the system of Address-Type Numbers; no algebraic sign is associated with them, and they are always positive numbers. However, the last example shows the use of "complementary addition", in which Z99 is equivalent to "-1".

Whenever an Instruction specifies the number of Words, or the number of Items, to be operated upon -- or whenever the Processor itself generates a tally -- such a number is always an Address-Type Number.

Appendix "C" to this Chapter discusses what happens if an Instruction refers to some non-existent address, such as L31 in a 2000-Word Memory, or &76 in any Memory.

One of the fundamental principles of programming must be made clear at this point. The nature of information stored in the Processor Memory is known only to the programmer; it is in no way inherent in the information itself. For example, a field might contain U 8 Q R 6 9 . If this field is named as containing an operand for an arithmetic Instruction, its contents will be interpreted by the Processor as the negative number -488969 (see Appendix "A"). If this field is named as containing an operand for an Instruction appropriate to Address-Type Numbers, its contents will be interpreted as the pair of Address-Type Numbers U88 R69 (see Appendix "B"). If the same field is named as containing an operand for one of the general data-handling operations, its contents will be interpreted as the set of alphanumeric characters U8QR69.

On the other hand, the Processor might be instructed to find its next Instruction in the Cell in which that field is stored. In this case, the field loses its identity, and its characters become part of the Instruction, to be interpreted according to the specific format of that Instruction.

## INSTRUCTIONS

Instructions in the NCR 304 System are in 3-address form. An Instruction (in general) names an operation to be performed, the addresses of two operands, and the address in which the result is to be stored.

This principle, however, has been very considerably expanded, so that execution of each Instruction includes most of the attendant "housekeeping" chores which ordinarily complicate the programmer's task. As a result, each Instruction is an entire unit operation, corresponding very closely to an operation as it is naturally conceived by the Systems Analyst without reference to any Data Processor. The comprehensiveness of each Instruction, the elimination of almost all housekeeping Instructions, and the consequent relief from the necessity of storing large numbers of Constants in a program, combine to make the System extremely easy to program, and to eliminate many sources of programming errors.

An Instruction specifies the following factors governing an operation:

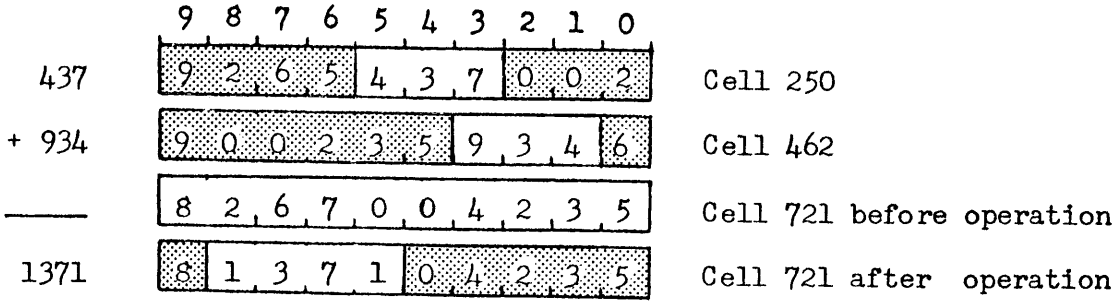
*** In addition to the actual Operation Code, each Instruction contains a Variation Designator, since most operations have several possible variations. For example, a MULTIPLY Instruction may yield either a full-length product, or a product which has been rounded off to any desired number of digits, depending on which variation of MULTIPLY is used.

The Variation Designator also governs the Self-Linking facility, described later in this chapter.

*** An Instruction names not only the addresses of the Cells containing the operands, but also the respective partial-word fields in thse Cells. Thus, an Instruction might specify:

Add: Contents of 53-field of Cell 250
Plus: Contents of 31-field of Cell 462
Store: Result in 85-field of Cell 721

The operation will be performed as follows:




The Processor automatically isolates each of the operands, aligns them at the right (Right-Justifies them), performs the addition, and inserts the result into the designated putaway-field without disturbing the contents of the rest of the putaway Cell.

*** Any of the addresses named in an Instruction may be modified, during execution of the Instruction, by the contents of an Index Register (Relative Address Register).

The ten Cells, 000 through 009 have the special property that they may be used as Index Registers. They are in every respect ordinary Cells in the Processor Memory, and may be referred to by any Instruction without restriction, just as any other Cells; but they have the additional ability to serve as Index Registers when required, and are conventionally reserved for this purpose.

Each Instruction designates one of these ten Cells as an Index Register for that Instruction, and also specifies which of the addresses named in the Instruction are to be modified. At the programmer's option:

- none of the addresses,
- any one of the addresses,
- any two of the addresses,
- all three of the addresses,

may be modified by the contents of the corresponding fields of the designated Index Register.

When an Instruction is about to be executed, the Processor copies that Instruction into its "private" working-registers (which are not accessible to the programmer). The Processor changes this copy of the Instruction, by MODIFY ADDING the contents of the corresponding field of the Index Register to each of the designated addresses in the Instruction. Then the changed copy of the Instruction is executed by the Processor.

An address named in an Instruction, therefore, is not necessarily the actual address of an operand or of a putaway; it is the base of the address. If that address is relative to an Index Register, then the base

is augmented by the contents of the corresponding field of the Index Register to obtain the actual address.

Note that the Instruction itself, stored in Memory, is never changed by the use of an Index Register. Every time an Instruction is to be executed, a new copy of the Instruction is set up, modified by the Index Register, and it is this modified copy of the Instruction which the Processor executes.

By making a series of Instructions all relative to the same Index Register, the programmer avoids the necessity of pre-setting and modifying each Instruction in the series. Modifying the contents of the Index Register has the effect of temporarily modifying (at the time of execution) every Instruction which is relative to it.

Some of the situations in which the Index Registers are used to advantage are:

- magnetic tape file management,
- working with multiple file records,
- table lookups,
- distribution and analysis.
- programming repetitive loops.

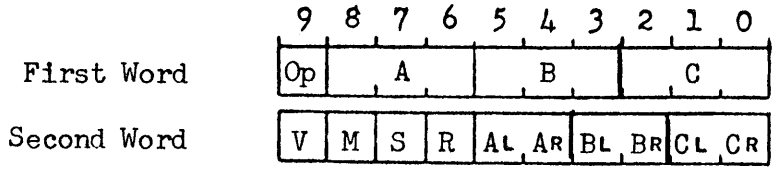
*** The NCR 304 System has very extensive automatic facilities (discussed in Chapter V) for selectively monitoring execution of the Instructions in a program, as an aid in program-checking. Some of these facilities make use of a Monitor-level Digit, which is part of each Instruction.

*** Many Instructions perform long series of operations, equivalent to complete subroutines, and require that additional factors be specified to define the total operation.

*** The contents of certain character-positions within some Instructions are irrelevant to the operation. These positions may be used to store a substantial portion of whatever constants are required by the program.

## INSTRUCTION FORMAT

An arithmetic Instruction, such as ADD, may be taken as typical of the format of the 304 Instructions. This basic format is varied and expanded to meet the requirements of all the Instructions.




Op: Operation code.

M: Automonitor level: 0, 1, 2, 3.

S: Selects addresses for modification by Index Register OOR. In many Instructions, the fields of the first word of the Instruction contain something other than addresses; nevertheless, the contents of these fields are modified just like addresses.
The precise significance of S is explained on the next page.

R: Designates Cell OOR as Index Register.

A: Base of address of first operand.

AL, AR: Locations of left-most and right-most character-positions, respectively, of the first operand.

B: Base of address of second operand.

BL, BR: Locations of left-most and right-most character-positions, respectively, of the second operand.

C: Base of address in which the result is to be put away.

CL, CR: Locations of left-most and right-most character-positions, respectively, of field allocated to the putaway of the result.

V: Variation designator.

## SELECTIVE MODIFICATION OF FIRST WORD OF INSTRUCTION BY AN INDEX REGISTER

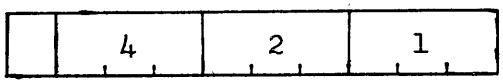
The three fields of the first word of an Instruction, which normally contain addresses, are called the syllables of that word. Each of these syllables is regarded as having an assigned "S-value" in connection with the digit S in the Instruction format:

The 86-field is the A syllable, with S-value 4
The 53-field is the B syllable, with S-value 2
The 20-field is the C syllable, with S-value 1

When the programmer has determined which syllables of an Instruction are to be relative to the designated Index Register, he adds the S-values of those syllables, to obtain the value of S.

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

S-values of syllables



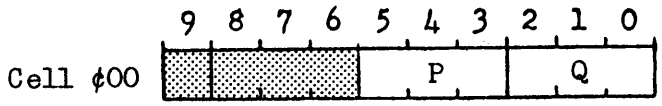

If S = 0, no Index Register is used, and R is irrelevant.

S = 0: No syllables modified
S = 1: C syllable modified
S = 2: B syllable modified
S = 3: B and C syllables modified
S = 4: A syllable modified
S = 5: A and C syllables modified
S = 6: A and B syllables modified
S = 7: A, B, C syllables modified

## OPERATION OF THE

## SEQUENCE-CONTROL REGISTER

The Processor uses the 53-field and the 20-field of Cell ¢00 as the Sequence-Control Register. Cell ¢00 is one of the "Special Cells" mentioned on page III-1, and described in Appendix "C".




The 20-field contains Q, the address of the first word of the next Instruction to be executed in the Program. In order to execute it, the Processor copies the Instruction into its working-registers, and at the same time augments Q by the number of words in the Instruction. For the moment, then, the new value of Q is the address of the next Instruction in the normal sequence.

The copy of the Instruction, in the working-registers, is then appropriately modified by the contents of the designated Index Register, and the modified copy of the Instruction is executed. If the Instruction causes a branch in the program, the new value of Q is automatically replaced by the branch address named in the Instruction.

Thus, whether or not the Instruction causes a branch, the Processor will always find the address of the next Instruction in the 20-field of Cell ¢00.

## SELF-LINKING

If the Variation Designator of an Instruction is negative, (if the sign-bit of V is a 1-bit), then under certain circumstances P and Q will be interchanged in ¢00, as part of the execution of the Instruction:

Decision Instruction which does not branch:

No interchange takes place, regardless of the sign of V.

Decision Instruction which branches:

The Processor makes the decision to branch; if V is negative, P and Q are interchanged; then the branch address named in the Instruction is stored in the 20-field of ¢00 in the usual manner.

In this way Q, the address of what would have been the next Instruction if there had been no branch, is retained in the 53-field of ¢00, and can serve as a link back to the branch-point at a later time.

Sequential Instruction:

When execution of the Instruction has been completed, and the result has been stored, then if V is negative, P and Q are interchanged before selecting the next Instruction for execution, with the result that the next Instruction is selected from the address initially stored as P.

Thus any sequential Instruction can impose a jump, and store a return link.

Since Cell ¢00 is fully addressible, and is accessible without restriction, the programmer has a great deal of freedom in branching and linking. Some of the techniques are:

1) Branch without linking:

Use a decision Instruction, V positive. If a link had previously been stored as P, it will be undisturbed.

2) Branch, and link back to the branch-point:

Use a decision Instruction, V negative. The link address will be preserved as P.

3) Branch, and link to some other point in the program:

Use any appropriate Instruction to store the address of the link-point as P. Then, for the branch, use a decision Instruction, V positive, which will not disturb P.

4) Return to the link-point:

Make V negative in the last Instruction (which will be a sequential Instruction) of the branch sequence.

5) Conditionally return to the link-point:

In the branch sequence, use an appropriate Instruction to pick up P and store it as the branch address named in the decision Instruction.

6) Jump unconditionally without linking:

Use an appropriate Instruction, V positive, to store the jump address as Q. Any link previously stored as P will be undisturbed.

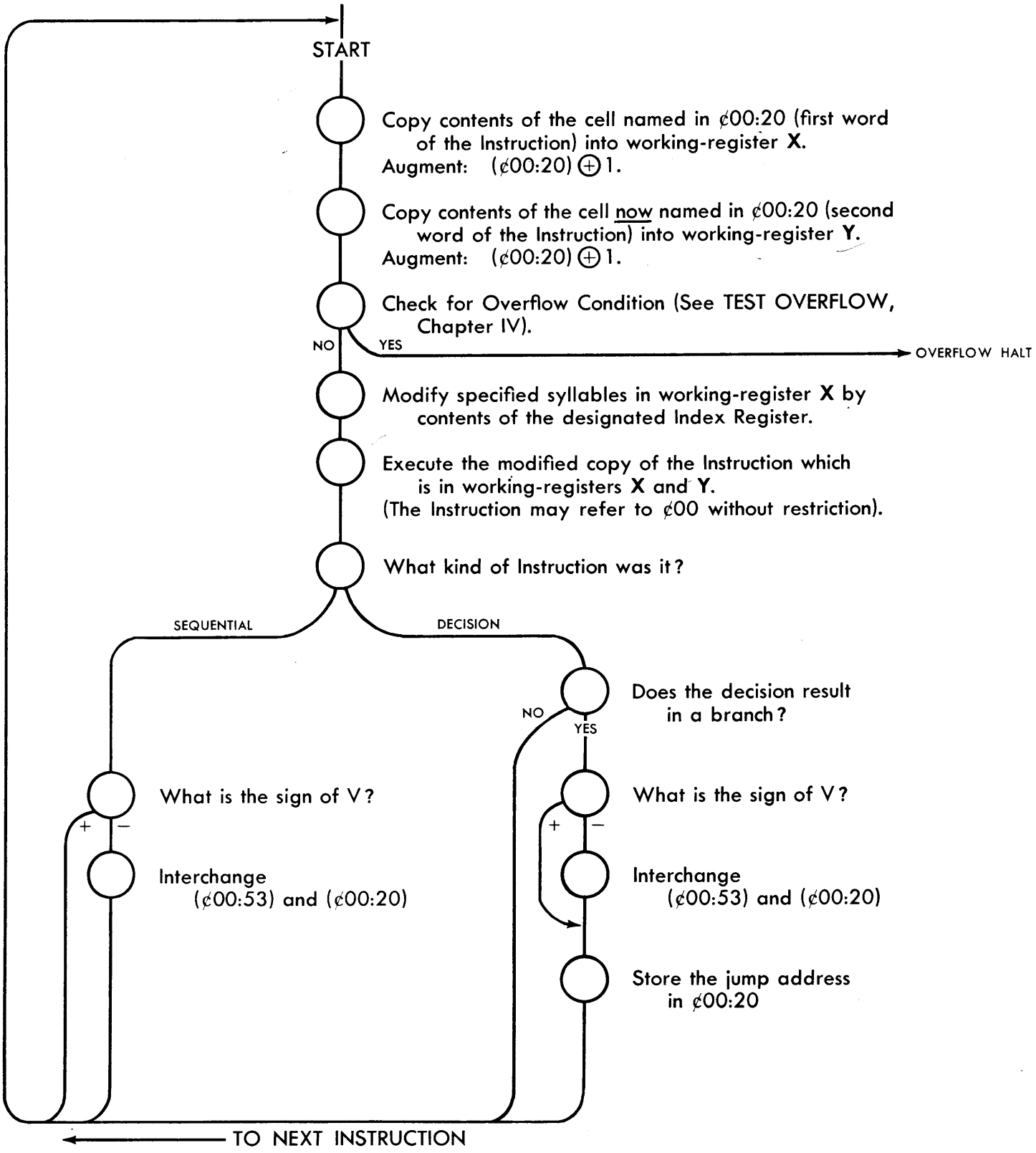
7) Jump unconditionally and link:

Use an appropriate Instruction, V negative, to store the jump address as P. Since V is negative, P and Q will be interchanged after the putaway, establishing the jump and the link.

8) Jump unconditionally, and link to some other point:

Use an appropriate Instruction, V positive, to store the link address and the jump address as P and Q respectively.

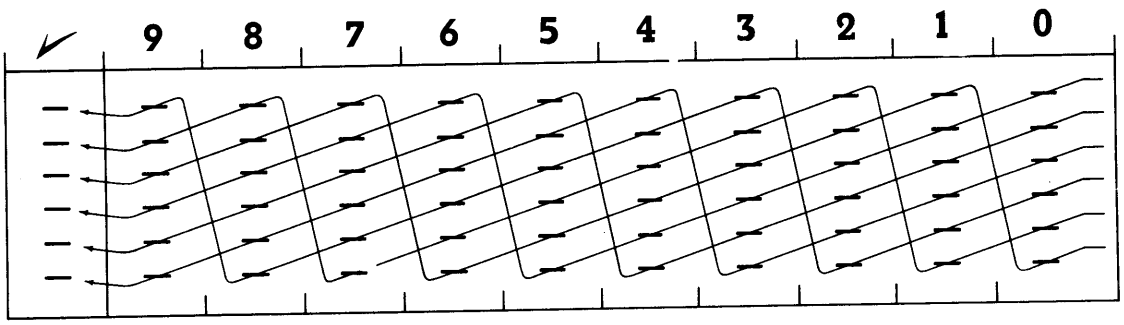
**The complete execution cycle of a standard Instruction (except for automonitoring) is shown in the following flow-chart: The notation (¢00:20) means "The contents of the 20-field of Cell ¢00"**



## SELF-CHECKING OF INFORMATION STORAGE

All storage of information in magnetic cores, and all transmission of information between units of the System, are automatically checked.

Every word in the Processor Memory is actually stored as 11 characters -- the 10 characters of the word itself, and one additional check-character assigned by the Processor. Each bit in the check-character establishes parity along one diagonal of the bits in the word, as shown in the diagram:




When information is to be read from Memory, an entire word is read into a working register, and the check-character verified. Then the appropriate field is selected from the word, and at the same time each character in the field is assigned its own seventh parity-bit. Information is stored in all operating registers in the form of 7-bit characters, and parity is checked during every readout.

When a field is to be stored in Memory, the entire word which already occupies the putaway cell is read into a working register, its check-character verified and a 7th parity-bit assigned to each character. Then the information to be putaway is inserted in the word, a new check-character is computed, the parity-bits of the individual characters are dropped, and the revised word is recorded in the putaway cell.

In all transmissions to and from Input and Output Units, every character is transmitted with its parity-bit, and parity is checked at both ends of the transmission. In addition, all Input and Output Units have their own self-checking facilities, described in Chapter VI.

In recording on Magnetic Tape, every character is recorded with its parity-bit, and in addition every record is assigned a diagonal-parity check-character. Other automatic checks on Magnetic Tape recording are discussed in Chapter II.

Any failure of word-parity or character-parity within the System causes an immediate error-halt, except for Magnetic Tape operations, where automatic program branches are furnished to permit repetition of the operation.

# APPENDIX A

## ARITHMETIC OPERATIONS ON ALPHABETIC CHARACTERS

1) Treatment of Zone Bits:

In performing any arithmetic operation, the Processor treats the contents of each field as though all its zone bits (except the sign-bit) were 0-bits. Regardless of the presence of irrelevant zone bits in the data, the result of an arithmetic operation will have all its zone bits (except the sign-bit) 0-bits. This is illustrated for ADD in Chapter IV.

In performing the operations of MODIFY ADD and MODIFY SUBTRACT, the Processor treats the first and second characters from the right, in each field, as though their zone bits were 0-bits; it treats the zone bits of the third character as representing the fourth digit of the Address-Type Number; and so on for each triad across the field from right to left. Regardless of the presence of irrelevant zone bits in the data, the result will have all its zone bits in the first and second, the fourth and fifth, the seventh and eighth, and the tenth character positions, 0-bits. This is illustrated for MODIFY ADD in Chapter IV.

2) Treatment of non-decimal configurations of numeric bits:

The results of adding any character to any other character, or of subtracting any character from any other character, are given in the tables on the following two pages. In order to use these tables, note that they refer only to configurations of numeric bits; the Processor always treats zone bits as described above. The results of these tables

# ADDITION

## NO CARRY FROM PREVIOUS DIGIT-POSITION

*Digits from the B-field*

| *Digits from the A-field* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | . | , |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | c2 | c3 | c4 | c5 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | ¢ | c2 | c3 | c4 | c5 | c6 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 |
| 5 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 |
| 6 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 |
| 7 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 |
| 8 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c2 | c3 | c4 | c5 | c6 | c7 |
| 9 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 |
| @ | @ | ¢ | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 |
| ¢ | ¢ | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ |
| SPACE | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ |
| & | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE |
| . | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE | c& |
| , | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE | c& | c. |

## CARRY FROM PREVIOUS DIGIT-POSITION

*Digits from the B-field*

| *Digits from the A-field* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | . | , |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | ¢ | c2 | c3 | c4 | c5 | c6 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 |
| 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c2 | c3 | c4 |
| 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 |
| 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c2 | c3 | c4 | c5 | c6 |
| 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 |
| 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 |
| 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 |
| @ | ¢ | c2 | c3 | c4 | c5 | c6 | c7 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ |
| ¢ | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ |
| SPACE | c3 | c4 | c5 | c6 | c7 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE |
| & | c4 | c5 | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE | c& |
| . | c5 | c6 | c7 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE | c& | c. |
| , | c6 | c7 | c8 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c@ | c¢ | c SPACE | c& | c. | c, |

# SUBTRACTION

## NO BORROW BY PREVIOUS DIGIT-POSITION

*Digits from the Minuend field*

*Digits from the field which is subtracted (Subtrahend)*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | . | , |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 4 | 5 | 6 | 7 | 3 | 9 |
| 1 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 | 6 | 7 |
| 3 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 | 6 |
| 4 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 |
| 5 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 |
| 6 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 |
| 7 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 8 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 9 | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | *b*¢ | 2 | 3 | 4 | 5 | 6 |
| @ | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 |
| ¢ | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 |
| SPACE | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 |
| & | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| . | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| , | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

## BORROW BY PREVIOUS DIGIT-POSITION

*Digits from the Minuend field*

*Digits from the field which is subtracted (Subtrahend)*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | . | , |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 | 5 | 6 |
| 3 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 2 | 3 | 4 | 5 |
| 4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 |
| 5 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 2 | 3 |
| 6 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 7 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | *b*¢ | 2 | 3 | 4 | 5 | 6 |
| 9 | *b*0 | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | *b*@ | *b*¢ | 2 | 3 | 4 | 5 |
| @ | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 3 | 4 |
| ¢ | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 2 | 3 |
| SPACE | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| & | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| . | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | 0 | *b*¢ | 2 | 3 | 4 | 5 | 6 |
| , | *b*0 | *b*1 | *b*2 | *b*3 | *b*4 | *b*5 | *b*6 | *b*7 | *b*8 | *b*9 | *b*@ | *b*¢ | 2 | 3 | 4 | 5 |

also apply to the numeric bits in the third, sixth and ninth positions during MODIFY ADD and MODIFY SUBTRACT; any "carry" or "borrow" out of these positions will affect the subsequent addition of the zone bits when they are treated as the leftmost digits of the Address-Type Numbers.

In the tables, "c" means a carry into the next digit position to the left; "b" means a borrow from the next digit position to the left. Note that different tables must be used for any digit position, depending on whether or not that position is affected by a carry from, or borrow by, the previous position to the right.

EXAMPLES:

ADD: 24& + 3¢¢

"A" = 0 0 0 0 0 0 0 2 4 &
"B" = 0 0 0 0 0 0 0 3 ¢ ¢
"C" = 0 0 0 0 0 0 0 6 6 8

SUB: 24& - 3¢¢

"A" = 0 0 0 0 0 0 0 2 4 & (Minuend)
"B" = 0 0 0 0 0 0 0 3 ¢ ¢ (Subtrahend)
"C" = b9 9 9 9 9 9 9 8 9 8 initially

Since there is a borrow out of the left end of "C", the result must be complemented, and then stored as negative. Another subtraction is automatically performed:

"A" = 1 0 0 0 0 0 0 0 0 0 0 (Minuend)
"B" = 9 9 9 9 9 9 9 8 9 8 (Subtrahend)
"C" = 0 0 0 0 0 0 0 1 0 2 finally

The result, therefore, is "minus 102"; this is putaway into a field of the specified length, and the leftmost character of that field automatically receives a 1-bit in its sign-bit position, to indicate that the result is negative.

In using the tables note that the operation ADD, performed on two numbers with different algebraic signs, will actually perform a Subtraction; and the operation SUBTRACT, performed on two numbers with different algebraic signs, will actually perform an Addition. Therefore it is necessary to choose a table on the basis of the operation actually performed, rather than on the basis of the Operation Code used.

Thus, in Addition of two numbers with like signs, or Subtraction of two numbers with unlike signs, use the "Addition" table. In Addition of two numbers with unlike signs, use the "Subtraction" table, noting that the negative number is always the Subtrahend. In Subtraction of two positive numbers, the B-field is the Subtrahend; in Subtraction of two negative numbers, the A-field is the Subtrahend, in the "Subtraction" table.

When the first word of an Instruction is modified by the contents of an Index Register, the Instruction is treated as the A-field, and the Index Register is treated as the B-field.

In performing the Modify Add which is part of COUNT, the contents of the Index Register being augmented is treated as the B-field, and the augmenter is treated as the A-field.

# APPENDIX B

## "SPECIAL" CELLS - RESTRICTIONS AND CONVENTIONS

In addition to the 2000 or 4000 words of Main Memory, another 400 or 800 Cells, respectively, are present in the Processor. These are called "Special" Cells because they are outside Main Memory, because they have unconventional addresses, and because there are certain restrictions and conventions governing their use. Even though the addresses of these Cells are unconventional, they are perfectly good addresses, and may be used by any Instruction, subject only to the limitations described in this Appendix. The "Special" Cells are:

| In 2000-word Memory | | | Additional in 4000-word Memory | | |
|---|---|---|---|---|---|
| @00 | through | @99 | %00 | through | %99 |
| ¢00 | through | ¢99 | £00 | through | £99 |
| □00 | through | □99 | d00 | through | d99 |
| △00 | through | △99 | s00 | through | s99 |

The conventions and restrictions imposed upon the use of 21 of these Cells are shown on the next page. The remaining 379 of the first 400 "Special" Cells are used by STEP (Standard Tape Executive Program) which is described briefly in Chapter IV in the section "Magnetic Tapes - General" and in detail in the *Neat* Manual. They are, therefore, not available to the programmer; but instead, the full Main Memory is at his disposal without having to sacrifice any space for "housekeeping" and "executive" programs.

The additional 400 "Special" Cells in a 4000-word Processor are available to the programmer without restriction, except that they are sequential only within each group of 200 addresses. £00 is the next Cell after %99, and s00 is the next Cell after d99, but there is not (in any

useful sense) any "next" Cell after £99 or s99. The "Special" Cells exist in blocks of 200, and no sequential operation may be permitted to proceed past the "barriers" £99 and s99. Note also that these Cells are sequential only in the forward direction; they cannot be counted backward either by MODIFY SUBTRACT or by MODIFY ADD of a complement.

| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| @00 | Used by the Processor to store supplemental information generated by Input and Magnetic Tape operations, and by Multiply, Divide, Summarize | | | | | | | | | |
| ¢00 | | May be used by the Programmer to store a programmed link | | | Used by the Processor for self-linking | | | Used by the Processor as Sequence-Control Register | | |
| ¢01 | | | | | | | | | | |
| ¢02 | Used by the Processor to store Automonitor information (See Chapter V) | | | | | | | | | |
| ¢03 | | | | | | | | | | |
| ¢04 | | | | | | | | | | |
| ¢05 | To be used by the Programmer as temporary storage for intermediate results of computation or for information which is to be held from program to program during a CONTINUOUS run | | | | | | | | | |
| ¢06 | | | | | | | | | | |
| ¢07 | | | | | | | | | | |
| ¢08 | | | | | | | | | | |
| ¢09 | This Cell will always be named as the irrelevant address whenever only one half of a DISTRIBUTE or a COMBINE is used | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| □00 | Today's Unit Period Number | | | | | TODAY'S DATE Month | | Day | | Year |
| □01 | Current Major Period Number | | | Memory Size as a 4-digit Number | | | | Memory Size as an Address-Type Number | | |
| □02 | ← Today's Dated Information → | | | | | | | | | |
| □03 | Additional Fixed Information for monitor as required by each installation | | | | | | | | | |
| □04 | | - | 0 | * | + | 0 | * | + | - | 0 |
| □05 | x | x | x | x | x | x | x | x | x | x |
| □06 | C | R | M | space | space | v | space | 0 | 1 | 0 |
| □07 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| □08 | space | space | space | space | space | space | space | space | space | space |
| □09 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

(□05), (□07), (□08) permit the programmer to pick up variable-length fields of "x's", "zeros", or "spaces" with a minimum of setups.

(□06) is used as the first word of a CRM-record. If the rest of the record is irrelevant, it may be written directly from this address, as either a Fixed-length or a Variable-length record.

(□09) furnishes a convenient standard location from which to pick up any 1-digit constant.

(□04:30) is used in converting a 4-digit Memory address into an Address-Type Number. Assume the address is in Cell XXX:85.

DIST: (XXX:88) ⟶ the 55-field and the 44-field of the second word of the next Instruction. (Set up the $A_L$ and $A_R$ positions.)

SERT: (□04:--) by (□04:33) ⟶ XXX:77

(XXX:75) now contains the Address-Type Number

THERE SHOULD NEVER BE ANY REASON FOR THE PROGRAMMER TO MAKE A PUTAWAY INTO ONE OF THE □-CELLS

## OTHER PROGRAMMING CONVENTIONS

Index Register #0 (Cell 000) is reserved for use by STEP (Standard Tape Executive Program) and by other self-contained subroutines. While the programmer is permitted to use Index Register #0 in his own coding, he must remember to preserve its contents elsewhere in Memory before entering any subroutine, and before using any Magnetic Tape operation.

Console Option Switches #0 and #1 are reserved for use in connection with the Label-checking functions of STEP, which are described in detail in the *Neat* Manual.

Handler #0 on Controller #0 is always reserved for the Program-Library Tape.

Index Forward: The Variation Designator must be "K" instead of "2", so that STEP may discriminate more easily between READ PARTIAL RECORDS and INDEX FORWARD.

# APPENDIX C

## MEMORY LOOKUP OF "IMPROPER" ADDRESS-TYPE NUMBERS

In order to interpret a 3-character field as a Memory address, the Processor goes through the following steps:

1) The two right-hand characters of the field are interpreted as though their zone bits were 0-bits.

2) In a Processor with a 2000-word Memory, the left-hand character of the field is interpreted as though its leftmost zone bit were a 0-bit.

3) If the numeric bits of either of the two right-hand characters do not correspond to a decimal digit, the Processor goes into Error-Halt.

4) If the numeric bits of the left-hand character do not correspond to either a decimal digit, or to "@" or "¢", the Processor goes into Error-Halt.

5) If the address passes tests (3) and (4), the lookup is then performed.

EXAMPLE:

ADD: (YES:30) + (ANY:42) ⟶ £+Q:95

This Instruction is interpreted as:

ADD: (Y52:30) + (A58:42) ⟶ £08:95 in 2000-word Memory

ADD: (H52:30) + (A58:42) ⟶ ¢08:95 in 4000-word Memory

Needless to say, the intentional use of this information in writing programs is most strongly discouraged, since such "tricks" serve no useful purpose, and they seriously impair the intelligibility of a program.

This information is furnished solely as an aid to identifying programming errors during code-checking.

# CHAPTER IV

# OPERATIONAL CHARACTERISTICS OF THE PROCESSOR

The following list illustrates the conventions which have been adopted in this chapter:

| | |
|---|---|
| CELL | A memory location. |
| ADDRESS | The number assigned to a cell. |
| WORD | The contents of a cell. |
| FIELD | Any set of adjacent characters within a word. |
| 72 FIELD | The field which extends from character-position 7 through character-position 2 within a word. |
| CELL 241 | The cell whose address is 241. |
| CELL A | The cell whose address is designated by A. |
| 241:72 | The 72 field of cell 241. |
| 241:33 | The 33 field (character-position 3 only) of cell 241. |
| A:20 | The 20 field of cell A. |
| (241) | The contents of cell 241. |
| (A) | The contents of cell A. |
| (241:72) | The contents of 241:72. |
| (A:20) | The contents of A:20. |
| A-FIELD | An Instruction names A as the base of the address of an operand. This base may be modified by the contents of an index-register to form the actual address of the operand.<br>The Instruction further specifies that the operand occupies a certain field within its cell.<br>The specified field in the actual address of the A-operand is referred to as the A-Field. |
| [A] | The contents of the A-field. |
| SYLLABLE | It is convenient to refer to the three fields of the first word of an Instruction, which normally contain addresses, as the syllables of that word.<br>The A Syllable is the 86 field.<br>The B Syllable is the 53 field.<br>The C Syllable is the 20 field. |

Shaded areas in an Instruction Format indicate character-positions which are irrelevant to the execution of the Instruction. The programmer may use these character-positions for storage of any information he desires.

OVERFLOW ALARM — A signal, set within the Processor, to indicate that the programmer has written certain Instructions improperly, and an incorrect or meaningless result has been obtained. When the Overflow Alarm has been set, the Processor will halt before executing the next Instruction, unless that next Instruction is the TEST OVERFLOW variation of TEST.

REGISTER — An auxiliary cell, not part of the memory proper, used by the Processor in carrying out its operations.

RIGHT-JUSTIFIED
LEFT -JUSTIFIED — These terms, borrowed from the printing trade, describe the placement of information within a field or a register so that the right-most or left-most character of the information occupies the right-most or left-most character-position of the field or register. This operation DOES NOT involve dropping non-significant zeros. As the terms "right-justified" and "left-justified" are defined here, non-significant zeros are considered to be part of the information.

Whenever, in this Manual, it is stated that the contents of a field are transferred right-justified or left-justified to a register, it must be understood that the register is first "cleared" by placing zeros in each character-position.

Whenever it is stated that the contents of a register are transferred right-justified or left-justified to a field, it must be understood that only as many characters are transferred as the field can contain. The remaining characters in the register are not transferred.

**It must be clearly understood that all registers mentioned in this Manual have been "invented" by the authors as aids to clarity and conciseness in describing operations.**

**These registers are entirely fictitious, and bear only coincidental resemblance (if any) to the registers actually present in the Model 304 Processor.**

## EXECUTION TIMES OF INTERNAL OPERATIONS

The basic unit of time within the Processor is a Cycle of 60 micro-seconds (60 μs = 0.000060 seconds). For convenience, this unit is referred to as 1 micro-minute (1 μmin = 0.000001 minute).

Execution times in this table are expressed in micro-minutes. To convert total micro-minutes into minutes, move the decimal point six places to the left.

| Operation | Fixed | I.R. Used | -V | Jump +V | Jump -V | Additional |
|---|---|---|---|---|---|---|
| 1 Add | 10 | 1 | 2 | | | + 1 if signs different } { AND result |
| 2 Subtract | 10 | 1 | 2 | | | + 1 if signs alike } { negative |
| 3 Multiply | 17 | 1 | 2 | | | + Sum of the digits of [B] + number of significant digits of [B] |
| 4 Divide RJ | 14 | 1 | 2 | | | + Sum of the Quotient digits + 2 per Quotient digit + 2½ average to round Quotient |
| 5 Divide LJ | 15 | 1 | 2 | | | |
| 6 Modify Add | 10 | 1 | 2 | | | |
| 7 Modify Subtract | 10 | 1 | 2 | | | |
| 8 Extract | 10 | 1 | 2 | | | |
| 9 Insert | 14 | 1 | 2 | | | |
| □ Add Binary | 10 | 1 | 2 | | | |
| △ Complement Binary | 10 | 1 | 2 | | | |
| A Test Bit | 9 | 1 | | | | |
| B Compare Numeric | 9 | 1 | | | | - 1 if signs are different AND neither number is zero |
| C Compare Alpha | 9 | 1 | | | | |
| D Compare Equality | 9 | 1 | | 1 | 1 | + 1 if [B] greater than [A] |
| E Count | 11 | 1 | | | | + 1 if [I] greater than T, or if no jump |
| F Test | 5 | 1 | | | | |
| H Combine | 12 | 1 | 2 | | | |
| J Distribute | 11 | 1 | 2 | | | + 1 for sign splitoff variation |
| K Suppress | 15 | 1 | 2 | | | + 1 for sign splitoff variation |
| L Edit | 13 | 1 | 2 | | | |
| M Merge | | | | | | See Instruction Description |
| N Move | 6 | 1 | 1 | | | + 2N + (1 per full 100 words) |
| Ø Pack | 6 | 1 | | 1 | | +13N + (1 per full 100 triples) |
| P Unpack | 6 | 1 | 1 | | | +13N + (1 per full 100 pairs) |
| Q Sift, single key | 15 | 1 | 2 | | | + 4 per item compared with sieve + 2 if AF termination |
| double key | 16 | 1 | 2 | | | + 5 per item compared with sieve + 2 if AF termination |
| R Summarize | 14 | 1 | | 1 | 1 | + 4N + (number of intermediate negative results) |
| BRANCHES WITHOUT EXECUTION: | | | | | | |
| All Tape Instructions | 11 | | | | | |
| PRINT | 7 | | | | | |

AUTOMONITOR: Add 12 micro-minutes between termination of the Instruction to be monitored, and execution of the first Instruction in the Monitoring Program. This time is used for storage of information in Cells ¢01 through ¢04.

## EXECUTION TIMES OF EXTERNAL OPERATIONS

These operations are usually timed separately, and the milli-second (0.001 sec) is the most convenient unit. Times in this table are in milli-seconds, and include execution time of each Instruction, and all acceleration and repositioning times.

To convert total milli-seconds into minutes, divide by 6 and move the decimal point four places to the left.

| Operation | Processor Time | | | | Processor freed-Controller Busy |
|---|---|---|---|---|---|
| | | per word | | | |
| | Fixed | Written | Copied or Searched | Read | |
| WRITE MAGNETIC TAPE | $6\frac{1}{3}$ | $\frac{1}{3}$ | | | 12 |
| READ MAGNETIC TAPE | 5 | | | $\frac{1}{3}$ ① | $10\frac{1}{3}$ ① |
| INDEX FORWARD | 5 | | | $\frac{1}{3}$ | $10\frac{1}{3}$ |
| INDEX BACKWARD | 11 ② | | | $\frac{1}{3}$ | 50 |
| WRITE-COPY | 8 | $\frac{1}{3}$ | | | $15\frac{1}{3}$ + $\frac{1}{3}$ per word copied |
| COPY or SEARCH | 1 | | | | $24\frac{2}{3}$ + $\frac{1}{3}$ per word copied or searched |
| WRITE-COPY-READ | | | | | |
| Unequal stop | 8 | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $15\frac{1}{3}$ |
| Equal stop | $5\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $9\frac{1}{3}$ + $\frac{1}{3}$ per word in the terminating record |
| COPY-READ | | | | | |
| Unequal stop | $10\frac{1}{3}$ | | $\frac{1}{3}$ | $\frac{1}{3}$ | $15\frac{1}{3}$ |
| Equal stop | $7\frac{1}{3}$ | | $\frac{1}{3}$ | $\frac{1}{3}$ | $9\frac{1}{3}$ + $\frac{1}{3}$ per word in the terminating record |
| SEARCH-READ | | | | | |
| Unequal stop | $10\frac{1}{3}$ | | $\frac{1}{3}$ | $\frac{1}{3}$ | $15\frac{1}{3}$ |
| Equal stop | $7\frac{1}{3}$ | | $\frac{1}{3}$ | $\frac{1}{3}$ | $12\frac{2}{3}$ |
| REWIND | $\frac{1}{3}$ | | | | —— Only Handler remains Busy |

| | | | |
|---|---|---|---|
| PRINT | 3.0 | | PRINTER BUSY |
| PUNCH PAPER TAPE | 0.5 | 17 per char. | Set to 600 lines/minute: 100 ms + 14 ms per blank line. |
| TYPE ON CONSOLE | 0.5 | 150 per char. | Set to 850 lines/minute: 70 ms + 14 ms per blank line. |
| PUNCH ON CONSOLE | 0.5 | 100 per char. ③ | With proper program timing, numeric information may be printed at 750 or 1200 lines/minute, respectively. |
| READ PAPER TAPE | 6.5 | 0.56 per char. | |
| READ CARDS | ~~0.5~~ 13.5 | ~~40~~ 30 per card | |

NOTES: ① If READ terminates because memory-allocation has been exceeded, increase the time in both columns by $\frac{1}{3}$ ms per word for average record-length in the file.

② If previous use of the Handler was completed less than 50 ms ago, there is 50 ms automatic delay before initiating Index Backward.

③ Control characters, such as "putaway" and "compute" must be counted.

SPECIFICATIONS on which execution-times of external operations are based:

| | | | |
|---|---|---|---|
| MAGNETIC TAPES: | Acceleration | 4 | ms |
| | Deceleration | 4 | ms |
| | Data Transfer | 30,000 | characters per second, effective |
| | Data Recorded | 200 | characters per inch |
| | Tape Movement | 150 | inches per second |
| | Rewind Speed | 225 | inches per second |
| | Tape Thickness | 1 | mil (0.001") mylar, laminated |
| | Tape Length | 3,600 | feet per reel |
| | Tape Holds more than | 8½ | million alphanumeric characters |
| PRINTER: | Printing Speed | 600 | lines per minute, best quality ( 750 for numeric information) |
| | | 850 | lines per minute, good quality (1200 for numeric information) |
| | Blank Lines (Vertical spacing) | 4,200 | lines per minute regardless of how many, or how few, lines are to be left blank |
| PAPER TAPE PUNCH: | Punching Speed | 60 | characters per second |
| CONSOLE TYPEWRITER: | Typing Speed about | 6½ | characters per second |
| | Punching Speed | 10 | characters per second |
| PAPER TAPE READER: | Reading Speed | 1,800 | characters per second |
| PUNCHED CARD READER: | Reading Speed | 1,500 | cards per minute |
| Card Punch (only off line) | Punch Speed | 100 | card / Min |

## OFF-LINE CONVERSION SPEEDS

PAPER TAPE to MAGNETIC TAPE:

Effective rate (if no "putaways" or "compute codes") . . . 1300 char/second

If there are "putaways" or "compute codes":
24.7 ms per record on Magnetic Tape
0.56 ms per character on Paper Tape

PUNCHED CARDS to MAGNETIC TAPE:

Effective rate . . . 1050 cards/minute

MAGNETIC TAPE to PAPER TAPE:

Effective rate . . . 59.5 characters/second

MAGNETIC TAPE to PUNCHED CARDS:

Effective rate . . . 100 cards/minute

MAGNETIC TAPE to PRINTER (with 320 Multiple-Purpose Converter):

Printer set for 600 lines per minute --

Alphanumeric: 100 ms per printed line
14 ms per blank line
Effective rate . . . 600 lines/minute

Numeric: 80 ms per printed line, and up to 2 blank lines
Effective rate . . . 750 lines/minute ①

Printer set for 850 lines per minute --

Alphanumeric: 70 ms per printed line
14 ms per blank line
Effective rate . . . 850 lines/minute

Numeric: 50 ms per printed line, and no blank lines
Effective rate . . .1200 lines/minute ①

---

NOTES: ① If any printed line is followed by more than the stated number of blank lines, figure the additional blank lines at 14 ms, and figure the following printed lines at alphanumeric speed.

Numeric timing is resumed with the first printed line which is not preceded by a blank line.

MAGNETIC TAPE to PRINTER (with 322 Printer Converter):

Printer set for 600 lines per minute --

Alphanumeric: 115 ms per printed line
14 ms per blank line
Effective rate . . . 525 lines/minute

Numeric: 80 ms per printed line, and up to 1 blank line (1)
Effective rate . . . 750 lines/minute

Printer set for 850 lines per minute --

Alphanumeric: 85 ms per printed line
14 ms per blank line
Effective rate . . . 705 lines/minute

Numeric: 50 ms per printed line, and no blank lines (1)
Effective rate . . .1200 lines/minute

---

NOTES (1) If any printed line is followed by more than the stated number of blank lines, figure the additional blank lines at 14 ms, and figure the following printed lines at alphanumeric speed.

Numeric timing is resumed with the first printed line which is not preceded by a blank line.

The following list gives the Operation Codes, Operation Names, and Standard Abbreviations for all Operation in the National 304 Data Processor

| | | |
|---|---|---|
| 1 | Add | ADD: |
| 2 | Subtract | SUB: |
| 3 | Multiply | MULT: |
| | Round | MULT:R |
| 4 | Divide Right-Justified | DRJ: |
| | Round | DRJ:R |
| 5 | Divide Left-Justified | DLJ: |
| 6 | Modify Add | MADD: |
| 7 | Modify Subtract | MSUB: |
| 8 | Extract | EXT: |
| 9 | Insert | SERT: |
| □ | Add Binary | BINA: |
| | Modulo 64 | BINA:M |
| Δ | Complement Binary | BINC: |
| A | Test Bit | TBIT: |
| B | Compare Numeric | CN: |
| C | Compare Alphanumeric | CA: |
| D | Compare Equality | CE: |
| E | Count | CNT: |
| F | Test | |
| | Overflow | TEST:O |
| | Option Switch | TEST:S |
| | Reader Code | TEST:R |
| | Punch Code | TEST:P |
| H | Combine | COMB: |
| J | Distribute | DIST: |
| | Sign split-off | DIST:S |
| K | Suppress | SUPP: |
| | Sign split-off | SUPP:S |
| L | Edit | EDIT: |
| | Check-protection | EDIT:P |
| M | Merge | |
| | Cutoff | MRGE:C |
| | Runout | MRGE:R |
| N | Move | MOVE: |
| O | Pack | PACK: |
| P | Unpack | UNPK: |
| Q | Sift | SIFT: |
| R | Summarize | SUMM: |
| S | Rewind Magnetic Tape | |
| | Source Tape | WIND:S |
| | Destination Tape | WIND:D |
| | Use Lockout, Source | LOCK:S |
| | Use Lockout, Destination | LOCK:D |
| T | Read Magnetic Tape | |
| | Complete Records | READ: |
| | Partial Records | READ:P |
| | Test Branch Conditions | READ:T |
| | Index Forward | INDX:F |
| | Index Backward | INDX:B |
| U | Write-Copy | WC: |
| V | Write-Copy-Read | WC:R |
| U | Copy | COPY: |
| V | Copy-Read | COPY:R |
| W | Write Magnetic Tape | |
| | Fixed-Length Records | WT:F |
| | Variable-Length Records | WT:V |
| | Test Branch Conditions | WT:T |
| X | Print | PRNT: |
| Y | Type-Punch | |
| | Console Typewriter (variations 0 thru 5) | TYPE: |
| | High-Speed Punch | |
| | Fixed Format | PPT:F |
| | Programmed Format | PPT:P |
| Z | Read Paper Tape | RPT: |
| Z | Halt | HALT: |
| * | Read Cards | RCD: |

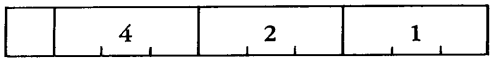
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

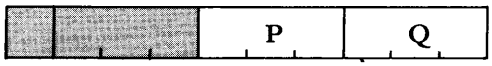
S-values of syllables

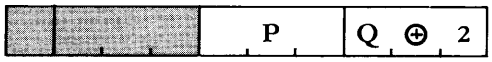
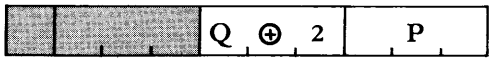



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation



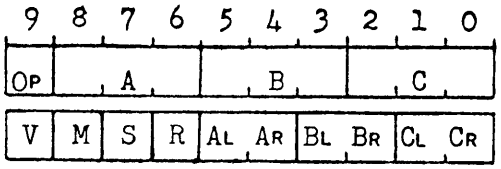

| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

## A D D

The respective contents of the A-field and the B-field are right-justified, and added. The result is stored, right-justified, in the C-field.

OVERFLOW ALARM: Will be set if the sum exceeds the capacity of the C-field.

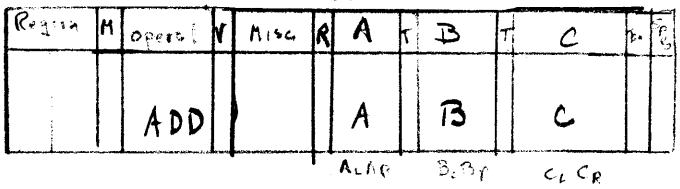
INSTRUCTION FORMAT:




**Operation: ADD (ADD)**

**Operation Code: 1**

Neat




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of addend.

AL, AR: locations of left-most and right-most digits, respectively, of the addend.

B: base of address of augend.

BL, BR: locations of left-most and right-most digits, respectively, of the augend.

C: base of address in which sum is to be stored.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the sum.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: ADD

This is a right-justified operation.

[A] + [B] ⟶ [C]

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb).

In the course of both these transfers, the zone-bits of all the characters transferred are replaced by 0-bits.

With due regard to the algebraic signs of [A] and [B], as stored in $A_L$ and $B_L$, the contents of registers Ra and Rb are added, and the sum is stored in a third working register (Rc), 11 characters-long.

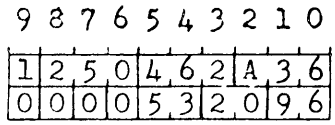
The contents of Rc then replace [C], right-justified. The correct algebraic sign of the sum is then inserted into the sign-bit position of character $C_L$.

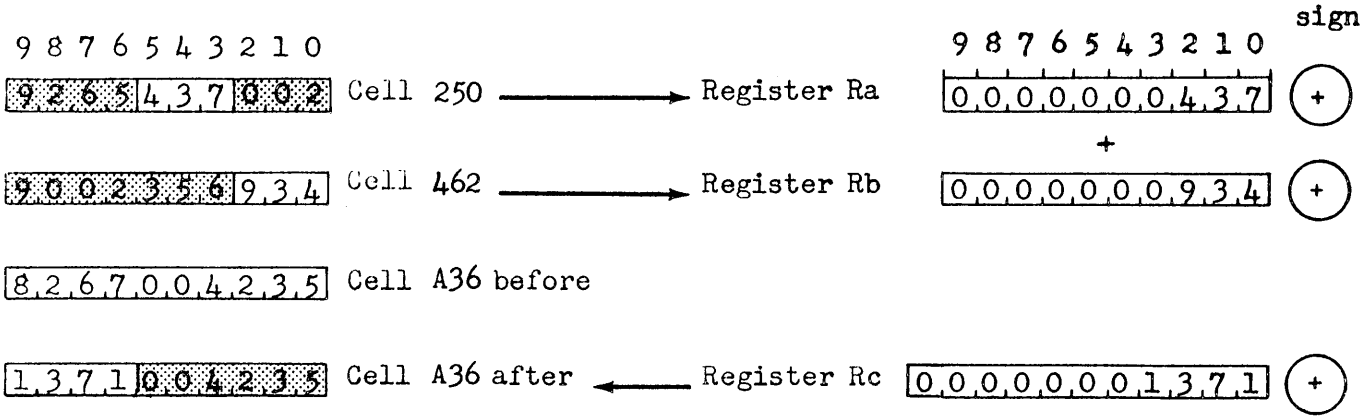
OVERFLOW ALARM: Will be set if some character (other than "zero") is not transferred from Rc to [C].

NOTE: (Rc) will equal Negative Zero only if [A] equals Negative Zero and [B] equals Negative Zero.

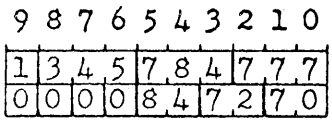
## Examples - ADD

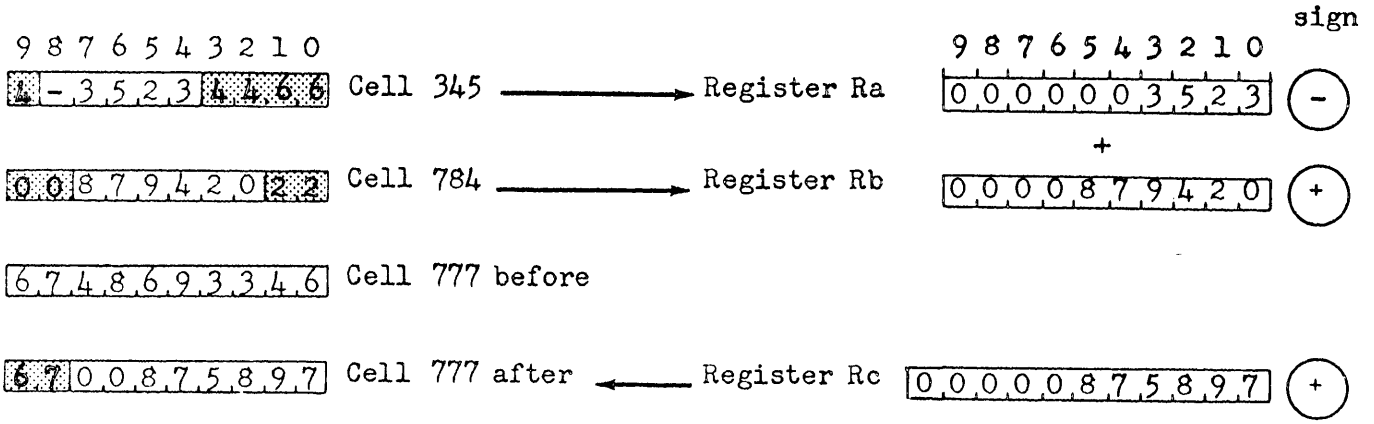
Example 1 - Add a positive number to a positive number.



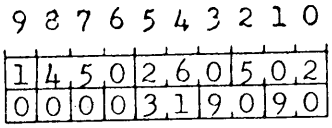



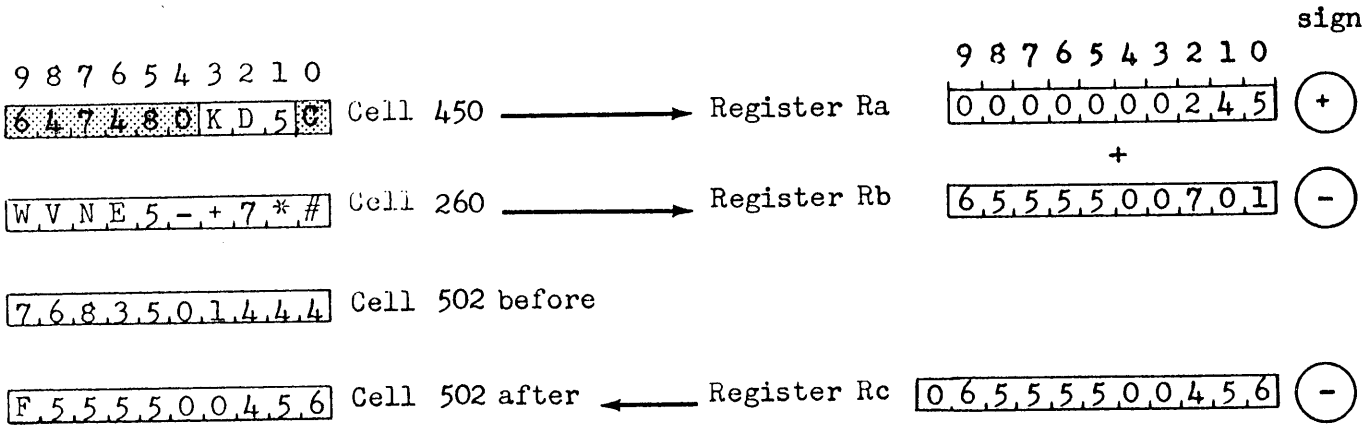

Example 2 - Add a positive number to a negative number. Result is a positive number.



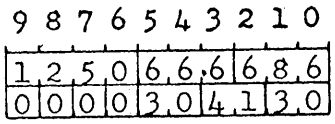



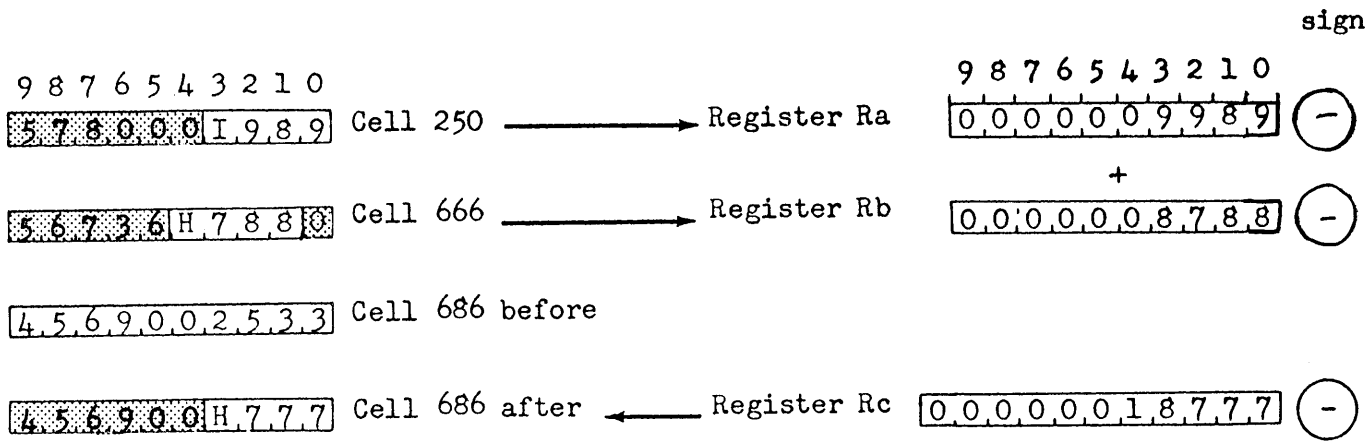
Example 3 - Add a negative number to a positive number. Sign of negative result combined with left-most character.




Example 4 - Add two negative numbers. Set overflow alarm.




Overflow alarm will be set

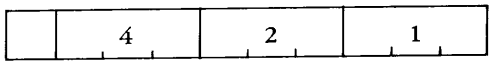
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

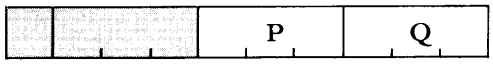
S-values of syllables




If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation



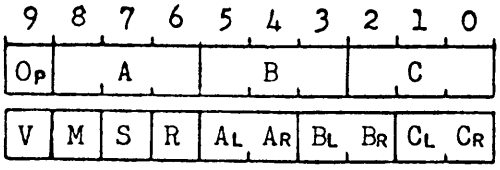

| V | (¢00) after operation |
|---|---|
| positive | P, Q ⊕ 2 |
| negative | Q ⊕ 2, P |

## S U B T R A C T

The respective contents of the A-field and the B-field are right-justified, and the difference, [A] - [B] is generated. The difference is stored, right-justified, in the C-putaway field.

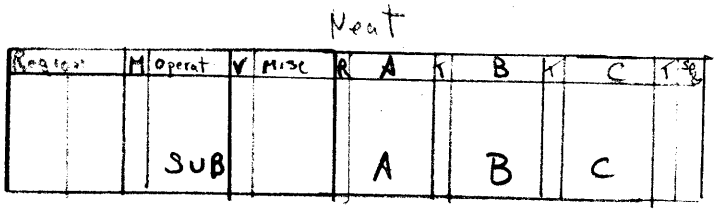
OVERFLOW ALARM: Will be set if the sum exceeds the capacity of the C-field.

INSTRUCTION FORMAT:




**Operation: SUBTRACT** (SUB)

**Operation Code: 2**




DEFINITIONS:

Op: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of minuend.

AL, AR: locations of left-most and right-most digits, respectively, of the minuend.

B: base of address of subtrahend.

BL, BR: locations of left-most and right-most digits, respectively, of the subtrahend.

C: base of address in which difference is to be stored.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the difference.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: SUBTRACT

This is a right-justified operation.

[A] - [B] ——→ [C]

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb).

In the course of both these transfers, the zone-bits of all the characters transferred are replaced by 0-bits.

The algebraic sign of [B] is then considered to be the opposite of the sign stored in $B_L$.

With due regard to the sign of [A], as stored in $A_L$, and the changed sign of [B], the contents of registers Ra and Rb are <u>added</u>, and the sum is stored in a third working register (Rc), <u>11</u>-characters long.

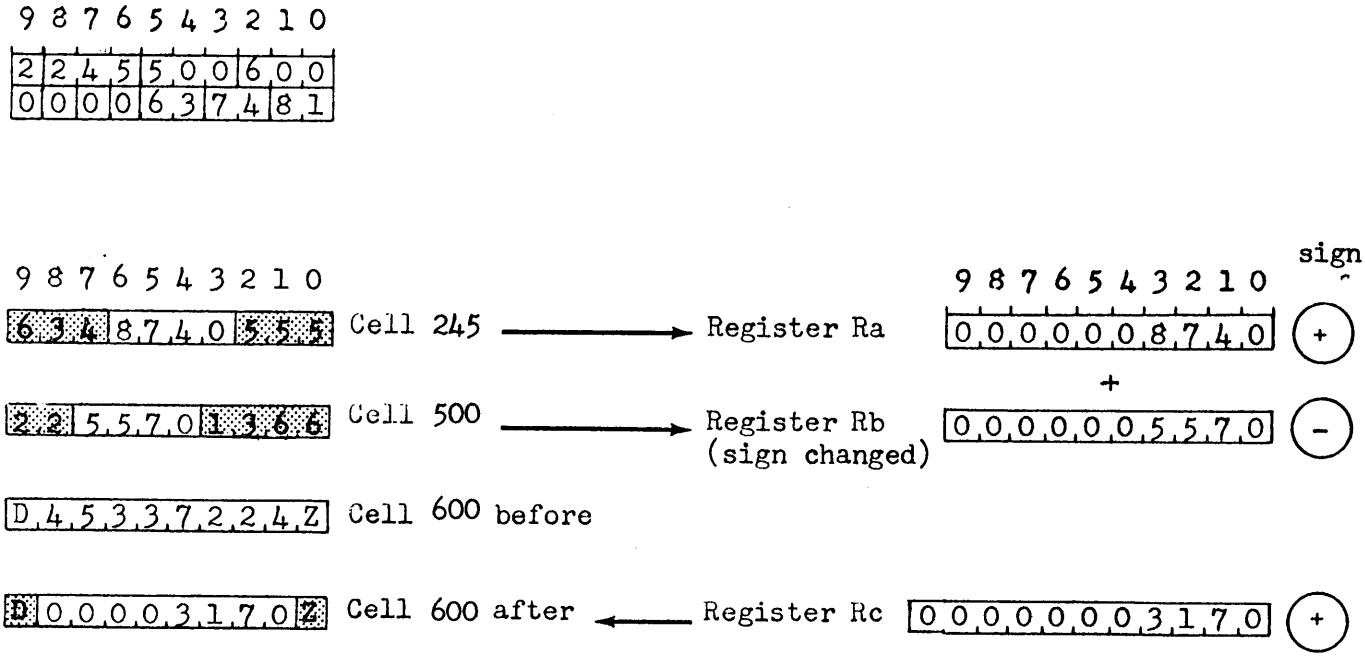
The contents of Rc then replace [C], right-justified. The correct algebraic sign of the result is then inserted into the sign-bit position of character $C_L$..

<u>OVERFLOW ALARM</u>: Will be set if some character (other than "zero") is not transferred from Rc to [C].

NOTE: (Rc) will equal Negative Zero only if [A] equals Negative Zero <u>and</u> [B] equals Positive Zero.

Examples - SUBTRACT

Example 1 - Subtract a positive number from a positive number.




Example 2 - Subtract a negative number from a positive number.

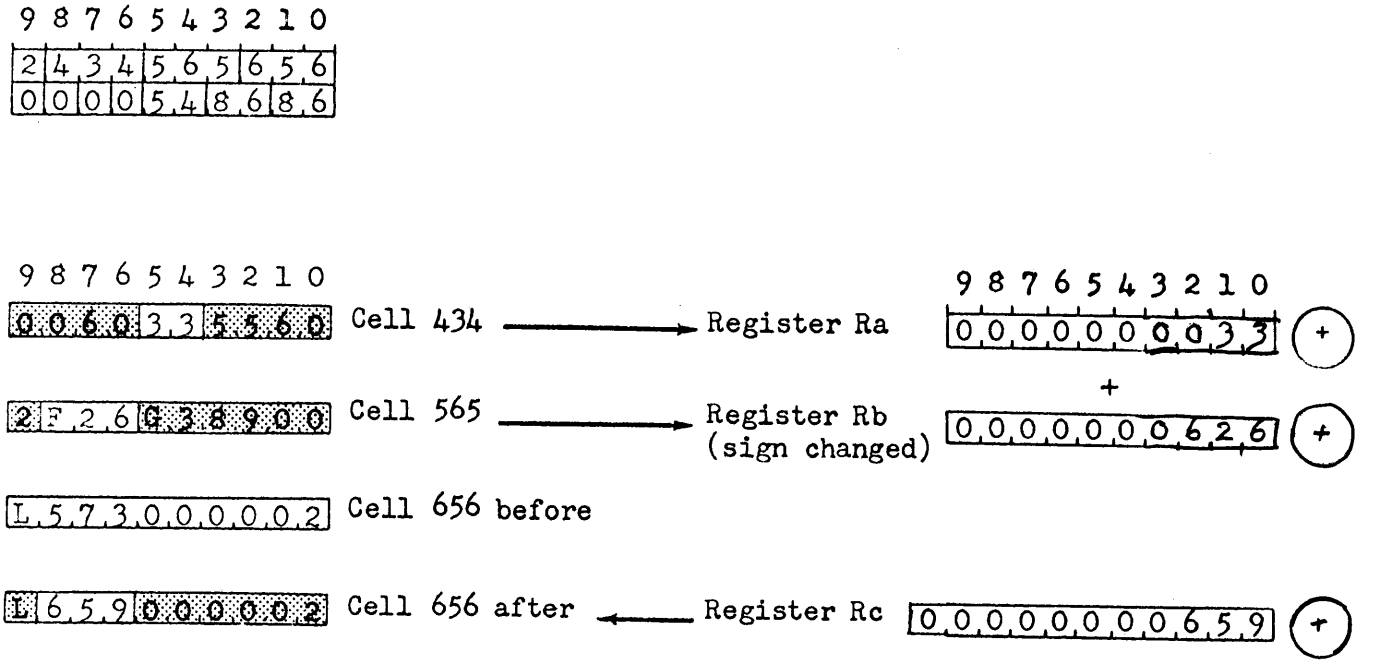


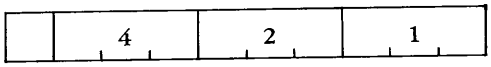
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

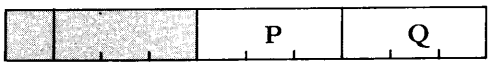
S-values of syllables

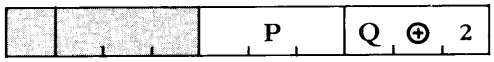
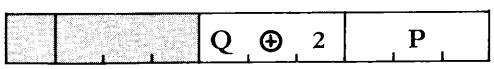



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |   |
| negative |   |

# M U L T I P L Y

The respective contents of the A-field and the B-field are left-justified and multiplied. The C-putaway is left-justified, and consists of as many digits as the C-field will accommodate. The next 10 digits of the product (from left to right) are then put away in cell @00, with the albegraic sign of the product.

There are two variations of this Instruction:

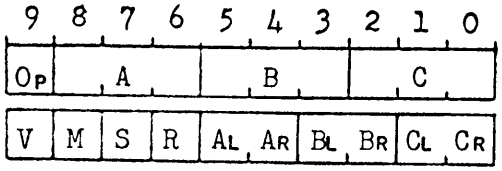
"Normal variation" in which the Instruction operates as just described.

"Rounded variation" in which the C-putaway is rounded at digit position $C_R$. The next 10 digits are still stored in @00.

The Instruction has the restriction that the A-field may not contain more than 9 digits.

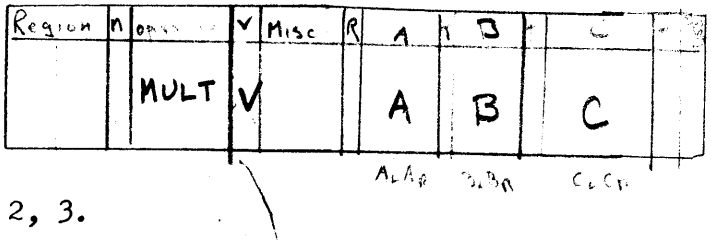
OVERFLOW ALARM: Will be set if the A-field contains more than 9 digits. Note also that rounding may cause overflow, and set the Overflow Alarm.

INSTRUCTION FORMAT:




Operation: MULTIPLY (MULT)

Operation Code: 3




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of multiplicand.

AL, AR: locations of left-most and right-most digits, respectively, of multiplicand.

B: base of address of multiplier.

BL, BR: locations of left-most and right-most digits, respectively, of multiplier.

C: base of address in which product is to be stored.

Next 10 digits → @00:90 with sign of product.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the product.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Unrounded product | MULT |
| 1 | Product rounded at CR. | MULT:R |

DESCRIPTION OF: MULTIPLY

This is a left-justified operation.

[A] x [B] ⟶ [C] and into @00:90. Both putaways with sign.

[A] is transferred, left-justified, to a 10-character-long working register (Ra); [B] is transferred, left-justified, to another 10-character-long working register (Rb). In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

The contents of Ra are then multiplied by the contents of Rb, and the 20-character-long product is generated in a third working register, 20-characters long (Rc). Note that the product of two 10-digit numbers is always 20 digits long.

V = 0: The contents of Rc are transferred, left-justified, to [C], for as many digits as exhaust the capacity of [C]. The next 10 digits, counting from left to right, are then transferred to @00:90. Any additional non-zero digits of the product are lost.

V = 1: The putaway in this variation is the same as that in V = 0, except that the portion of the product transferred to [C] is rounded in digit-position $C_R$: that is, if the digit in position 9 of @00 is a "5" or greater, then [C] is augmented by "1".

If the algebraic signs of the two operands are alike (as indicated by the sign-bits of $A_L$ and $B_L$) then the sign of the product is positive; a 0-bit is inserted into the sign-bit position of digit $C_L$, and into the sign-bit position of the digit in position 9 of @00.

If the algebraic signs of the two operands are unlike, then the sign of the product is negative; a 1-bit is inserted as the sign-bit of $C_L$, and as the sign-bit of digit-position 9 of @00.

Decimal Point: The number of digits to the left of the decimal point in the product is equal to the sum of the number of digits to the the left of the decimal point in each of the factors. Thus:

```
      2.000000000  (Ra)
    x 3.000000000  (Rb)
-----------------------
06.000000000000000000  (Rc)
```

RESTRICTION: [A] may not be more than 9 digits long. Otherwise, the product will be invalid, and the Overflow Alarm will be set.

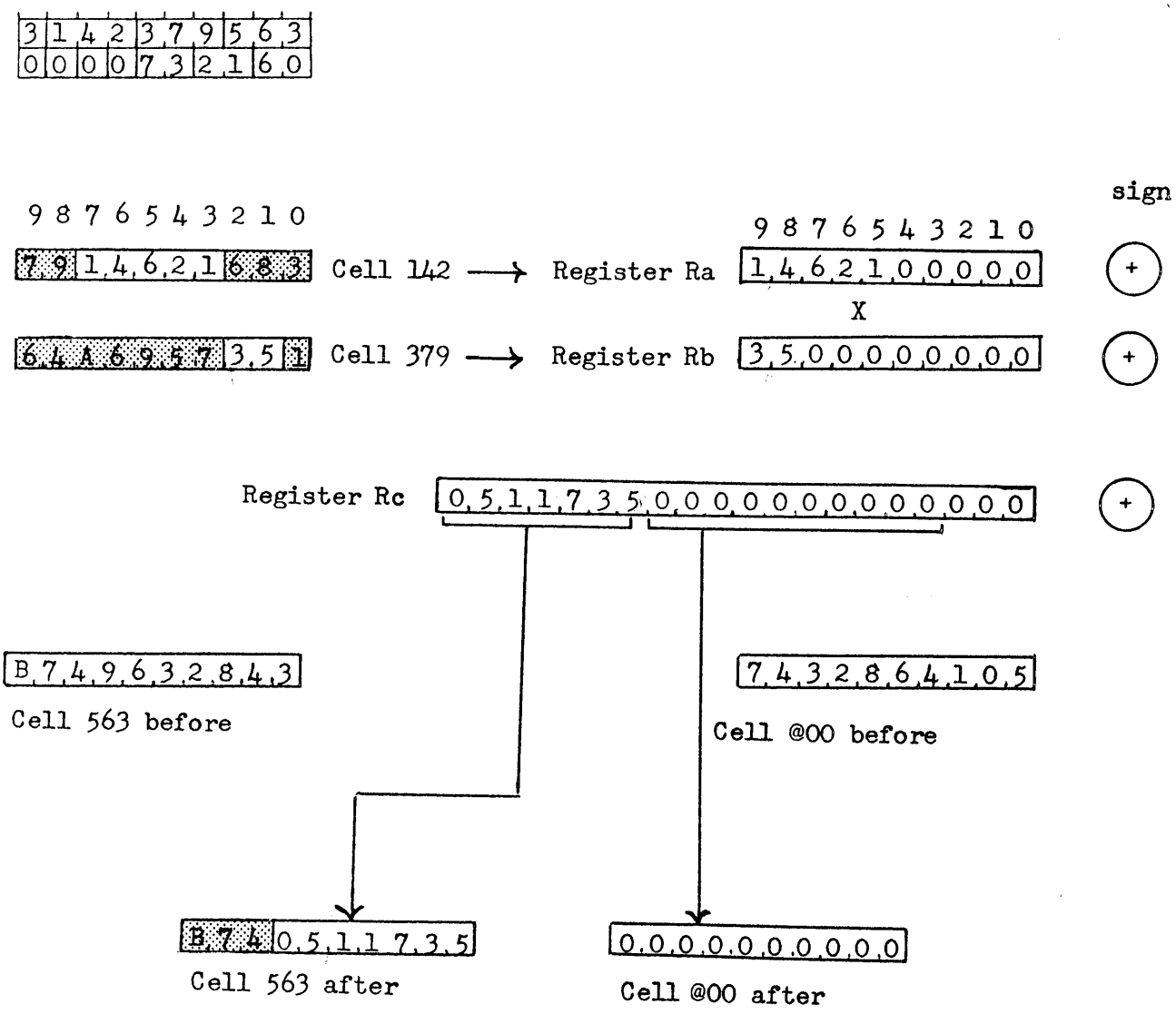
OVERFLOW ALARM: Will not be set if some non-zero character is not transferred from Rc to @00.

Will be set if the above restriction is violated.

Rounding may cause a carry to be generated out of $C_L$; if so, the Overflow Alarm will be set. The carry will be lost.

Example 1: V=0

[A] and [B] both are positive numbers



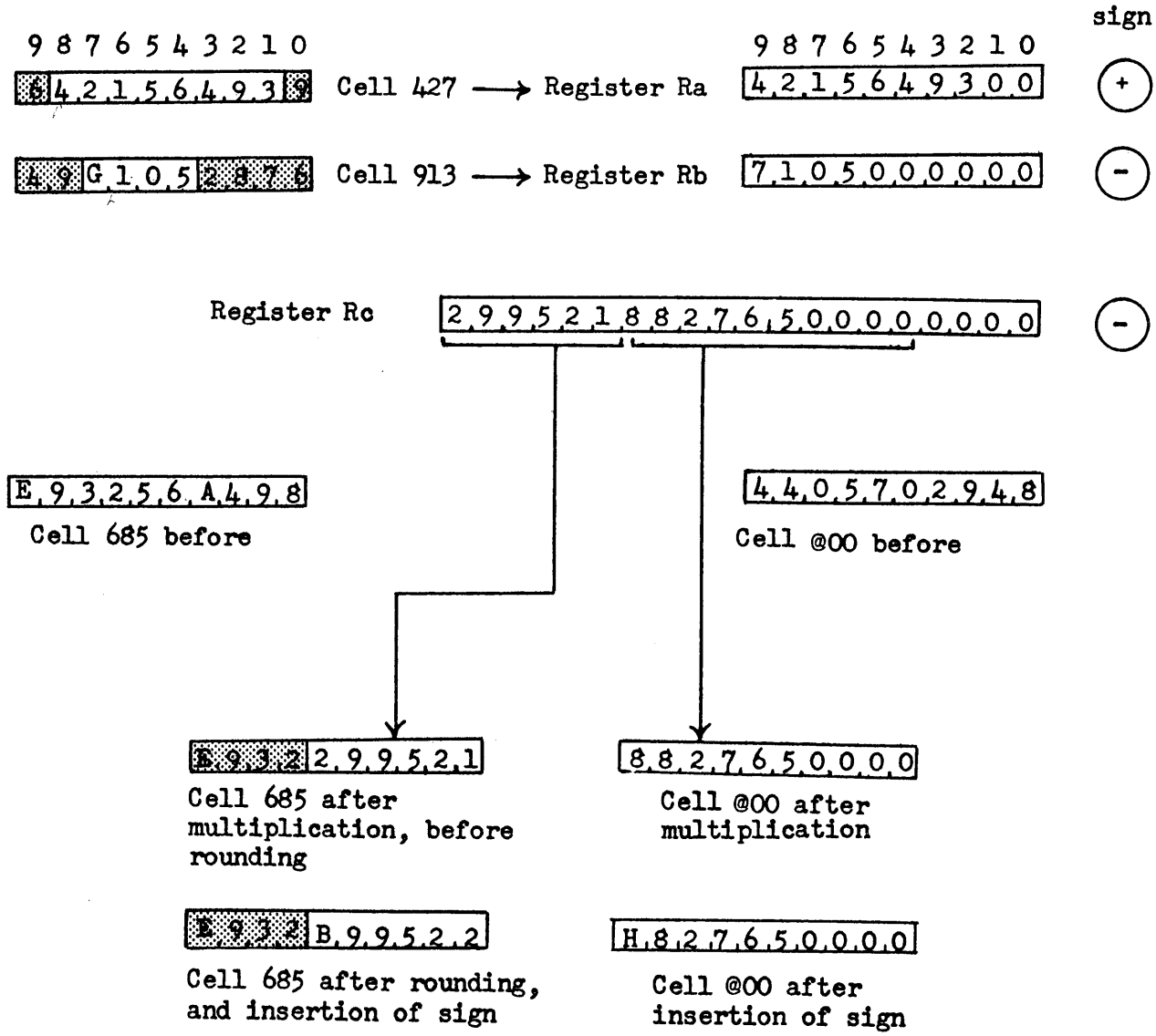
Example 2: V=1

Sign of [A] is positive

Sign of [B] is negative

Putaway in [C] is rounded at $C_R$

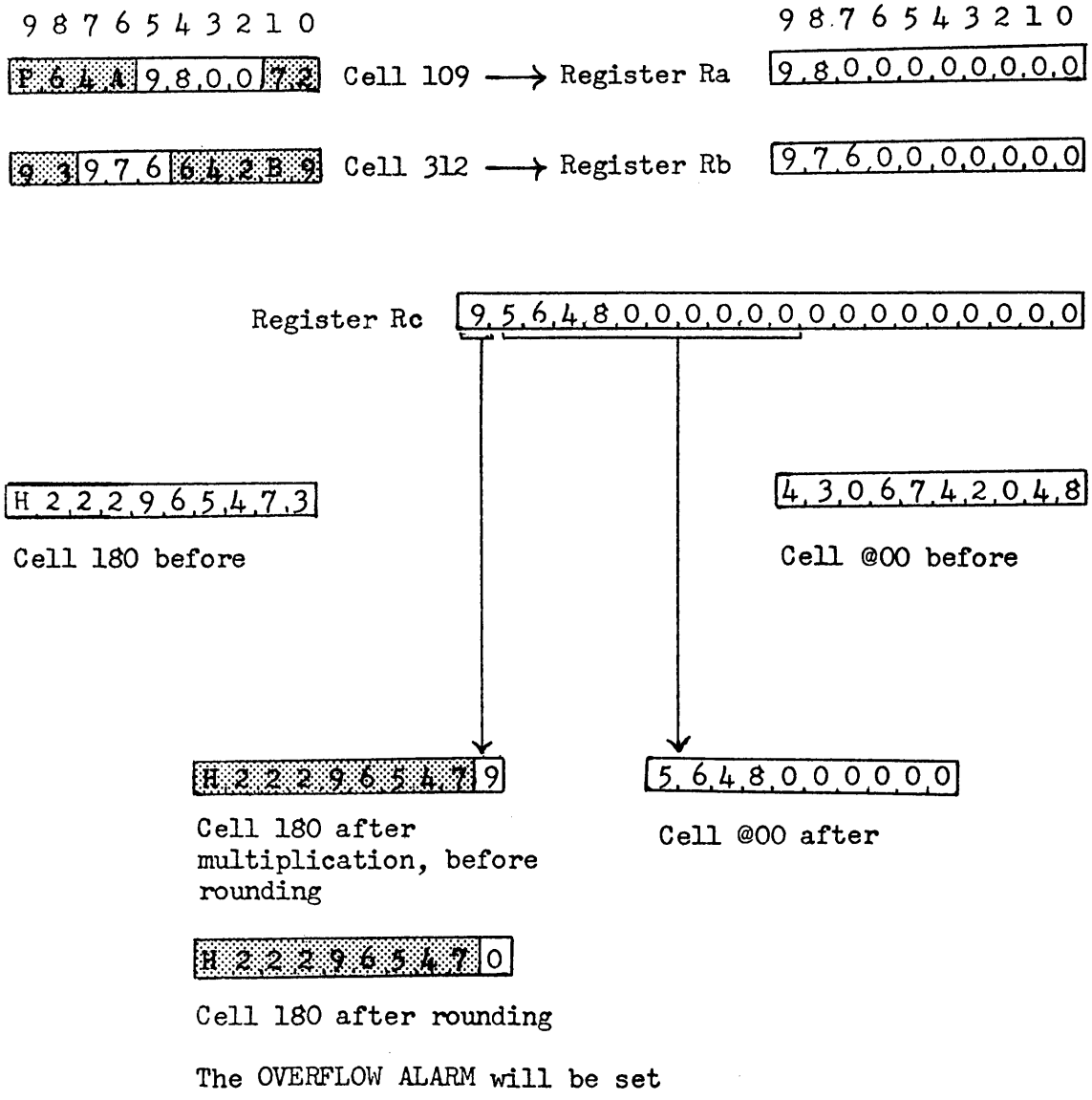
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 2 | 7 | 9 | 1 | 3 | 6 | 8 | 5 |
| 1 | 0 | 0 | 0 | 8 | 1 | 7 | 4 | 5 | 0 |



Example 3: V=1

Illustration of how rounding can set Overflow Alarm

| 3 | 1 | 0 | 9 | 3 | 1 | 2 | 1 | 8 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 5 | 2 | 7 | 5 | 0 | 0 |




The OVERFLOW ALARM will be set

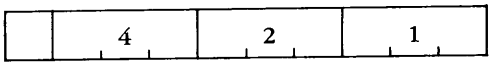
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

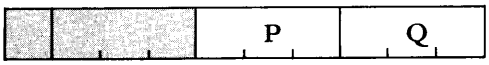
S-values of syllables

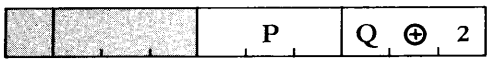
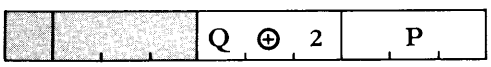



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

## D I V I D E  R I G H T - J U S T I F I E D

The respective contents of the A-field and the B-field are right-justified, the quotient $\frac{[B]}{[A]}$ is generated as though [A] and [B] were integers, and the quotient is stored, right-justified, in the C-Field.

In one variation, the remainder is stored, right-justified, in cell @00, with the algebraic sign of [B].

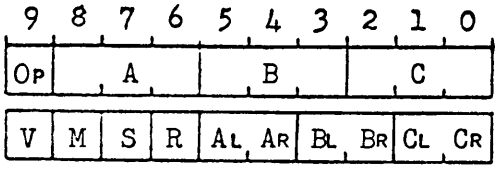
In the other variation, the quotient is rounded at the right-most digit position of the C-field, and no putaway is made to @00.

The Instruction has the restriction that the A-field may not contain more than 9 digits.

OVERFLOW ALARM: Will be set if the A-field contains more than 9 digits, or if an attempt is made to divide by zero. Note also that rounding can cause overflow, and set the Overflow Alarm.

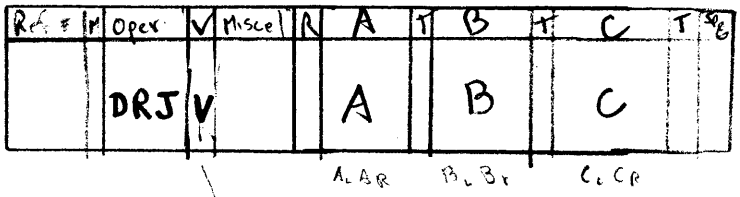
Will be set if the result exceeds the capacity of the C-field.

INSTRUCTION FORMAT:




**Operation: DIVIDE, RIGHT-JUSTIFIED (DRJ)**

**Operation Code: 4**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of divisor.

AL, AR: locations of left-most and right-most digits, respectively, of the divisor.

B: base of address of dividend.

BL, BR: locations of left-most and right-most digits, respectively, of the dividend.

C: base of address in which quotient is to be stored.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the quotient.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Quotient unrounded | DRJ |
| 1 | Quotient rounded at CR | DRJ:R |

NOTE: If quotient is unrounded, remainder is stored in @00:90 with the algebraic sign of [B]; putaway is right-justified.

Note Assumed decimal places in $R_c$ = [No. Decimal place of $R_b$] − [No. Decimal Places of $R_a$]
+ there are no decimal places and

IF the above result is negative, then the equivalent number of least significant zeros should be planted in the putaway to the right of the Rc putaway to round out all the digits to the left of the decimal point

No examples show num > denom.
what if denom > num?

DESCRIPTION OF: DIVIDE RIGHT-JUSTIFIED

This is a right-justified operation.

$$\frac{[B]}{[A]} \longrightarrow [C], \text{ unrounded,} \qquad \text{remainder} \longrightarrow \underline{@00:90}$$

or $$\frac{[B]}{[A]} \longrightarrow [C], \text{ rounded.}$$

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb). In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

The division is performed, generating the same number of digits of the quotient as the number of digits specified by the size of [B]. The quotient appears, right-justified, in a third 10-character-long working register (Rc), with the remainder being held in Rb.

The contents of Rc then replace [C], right-justified.

V = 0: The contents of Rb (the remainder) replace the contents of @00:90, and the sign-bit of the character in position $B_L$ (i.e.-the sign of the dividend) is copied into the sign-bit position of character-position 9 of @00, so that the remainder has the same algebraic sign as [B].

V = 1: The division is resumed, to generate one more digit of the quotient. If that digit is a "5" or greater, then [C] is augmented by "1". No putaway is made to @00, in this variation.

If the algebraic signs of the two operands are alike, (as indicated by the sign-bits of the digits in positions $A_L$ and $B_L$) then the sign of the quotient is positive; a 0-bit is inserted into the sign-bit position of $C_L$. If the algebraic signs of the two operands are unlike, then the sign of the quotient is negative; a 1-bit is inserted as the sign-bit of $C_L$.

RESTRICTIONS: 1) An attempt to divide by zero will set the Overflow Alarm, and no division will take place.

2) [A] may not be more than 9 digits long, Otherwise, the Overflow Alarm will be set and the quotient will be invalid.

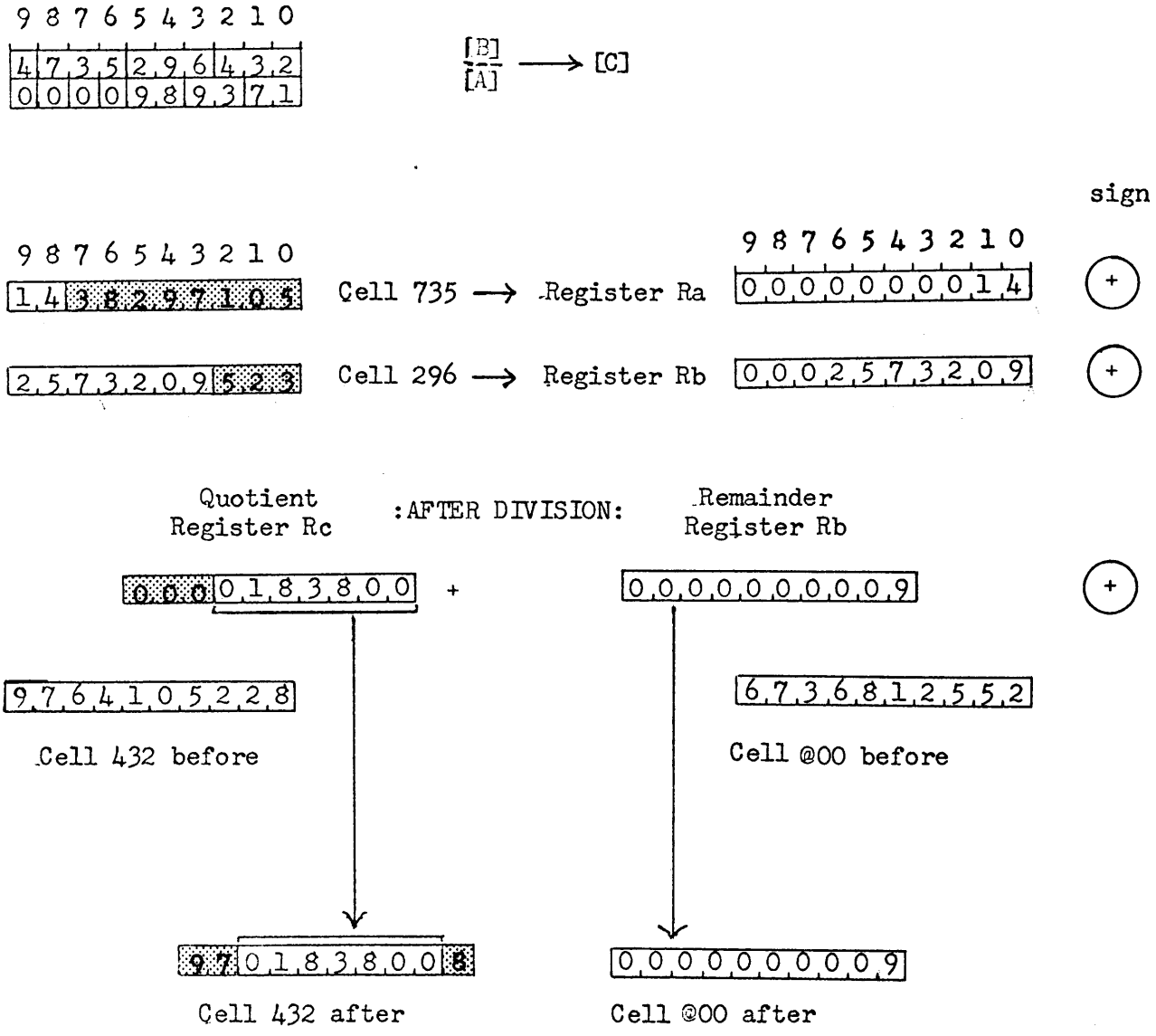
OVERFLOW ALARM: Will be set if restrictions 1 or 2 are violated.

Will be set if some character (other than "zero") is not transferred from Rc to [C].

Rounding may cause a carry to be generated out of $C_L$; if so, the Overflow Alarm will be set. The carry will be lost.

Example - DIVIDE RIGHT-JUSTIFIED

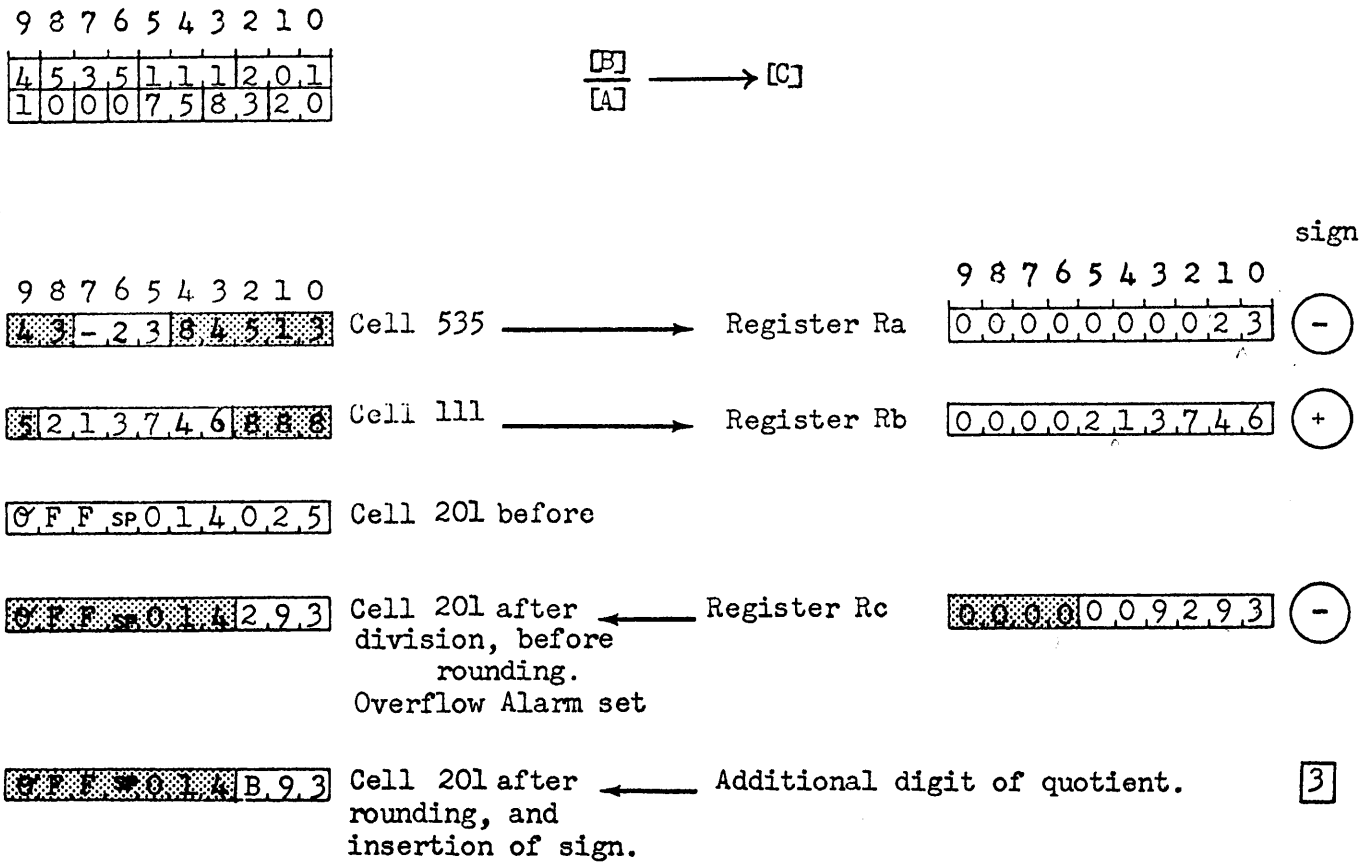
Example 1 - Divide a positive number by a positive number and save the remainder. V=0



## Example - DIVIDE RIGHT-JUSTIFIED

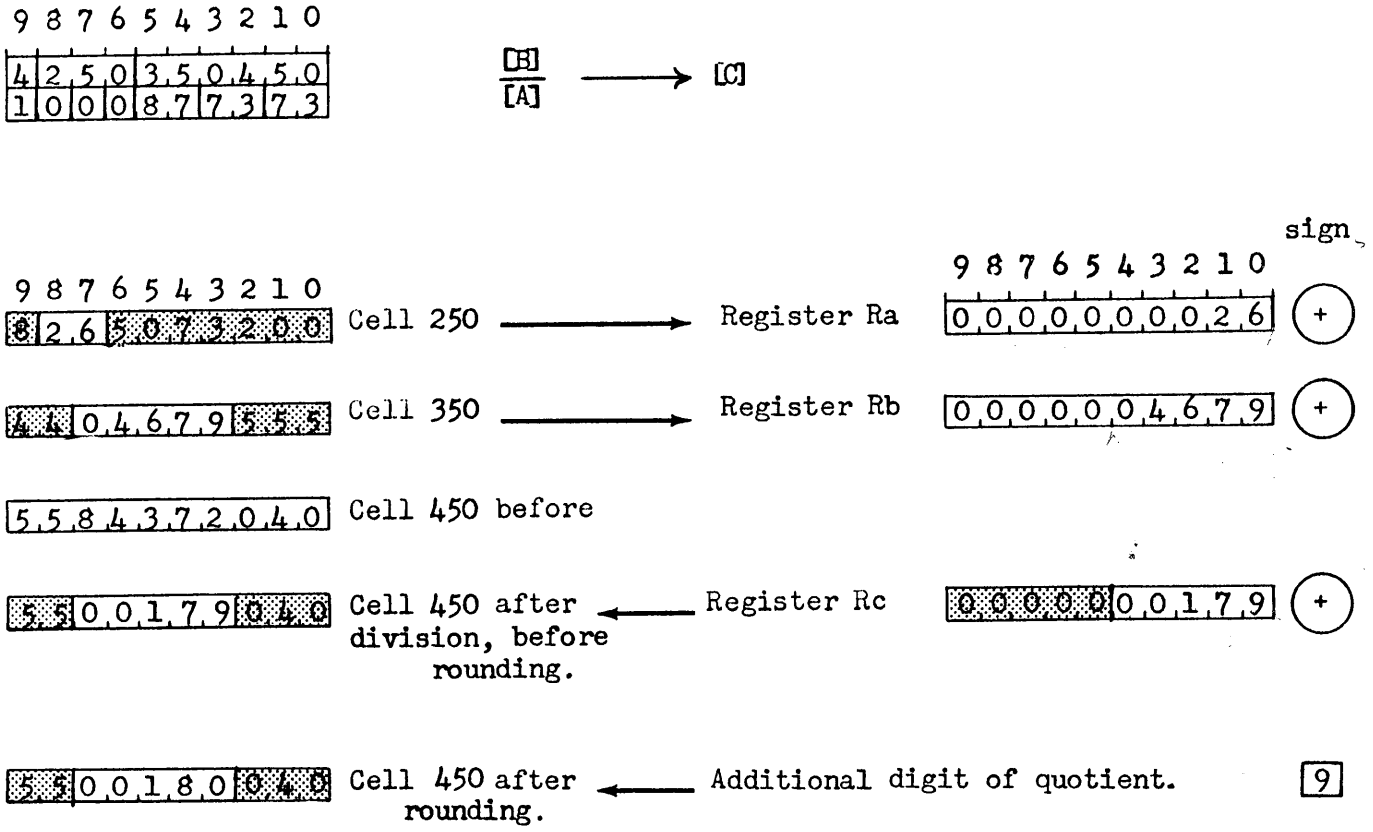
Example 2 - Divide a positive number by a negative number and round. V = 1

Putaway to cause overflow.



Example - DIVIDE RIGHT-JUSTIFIED

Example 3 - Divide a positive number by a positive number and round. V = 1



Example 4 - Divide a negative number by a positive number and round. V = 1

Rounding to cause overflow.

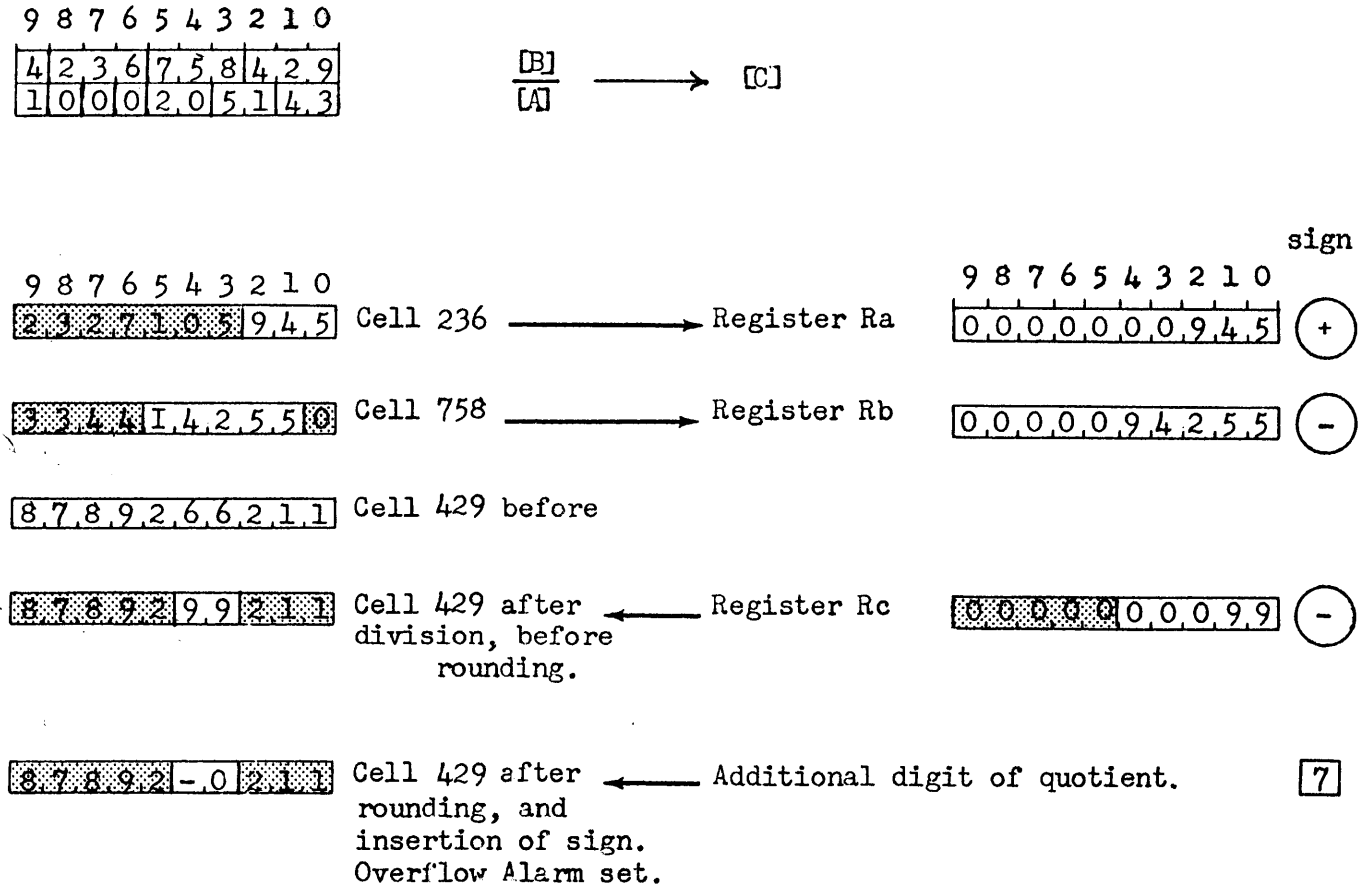


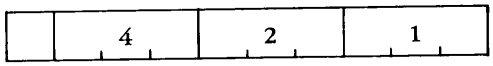
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

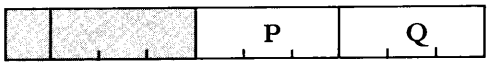
S-values of syllables




If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive | P, Q ⊕ 2 |
| negative | Q ⊕ 2, P |

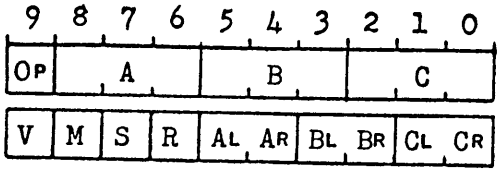
## D I V I D E  L E F T - J U S T I F I E D

The respective contents of the A-field and the B-field are left-justified, and the quotient $\frac{[B]}{[A]}$ is generated and stored, left-justified, in the C-field, and rounded.

The Instruction has the restriction that the left-justified contents of the A-field, regarded as a 10-digit number, must be greater in magnitude than the contents of the B-field, similarly regarded.

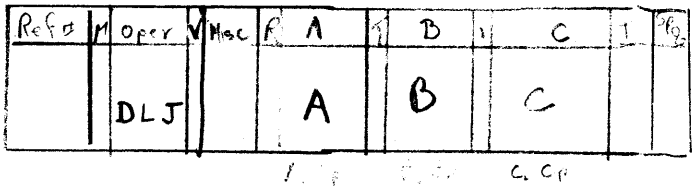
OVERFLOW ALARM: Will be set if the above restriction is violated, or if an attempt is made to divide by zero. Note also that rounding can cause overflow, and set the Overflow Alarm.

INSTRUCTION FORMAT:




**Operation: DIVIDE, LEFT-JUSTIFIED** (DLJ)

**Operation Code:** 5




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of divisor.

AL, AR: locations of left-most and right-most digits, respectively, of divisor.

B: base of address of dividend.

BL, BR: locations of left-most and right-most digits, respectively, of dividend.

C: base of address in which quotient is to be stored.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the quotient.

V: variation designator;
only the sign of V is relevant.

Note: Compute number of places to left of [illegible] point by

IF result is negative, the [illegible] number of zeros must be placed into the putaway area to provide for zeros to right of [illegible] point

DESCRIPTION OF: DIVIDE LEFT-JUSTIFIED

This is a left-justified operation. $\frac{[B]}{[A]} \longrightarrow [C]$, rounded.

[A] is transferred, left-justified, to a 10-character-long working register (Ra). [B] is transferred, left-justified, to another 10-character-long working register (Rb). In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

At this point, the 10-digit number in Ra must be greater than the 10-digit number in Rb. The quotient, therefore, will be generated as though it were a decimal fraction.

The contents of Ra are then shifted one character-position to the right, introducing a zero digit at the left, and dropping the right-most digit. If [A] is 10 digits long, therefore, division is performed as though the right-most digit were zero. The result of this shift is to position the divisor in Ra so that, when the quotient is generated in a third 10-character-long working register (Rc), the first digit to the right of the imaginary decimal point will appear in position 9 of Rc.

The number of digits of the quotient generated in Rc is the same as the number of digits allocated for the putaway in [C].

The contents of Rc then replace [C], left-justified.

Next, the division is resumed, to generate one more digit of the quotient. If that digit is a "5" or greater, then [C] is augmented by "1".

If the algebraic signs of the two operands are alike (as indicated by the sign-bits of the digits in positions $A_L$ and $B_L$) then the sign of the quotient is positive; a 0-bit is inserted into the sign-bit position of $C_L$. If the algebraic signs of the two operands are unlike, then the sign of the quotient is negative; a 1-bit is inserted as the sign-bit of $C_L$.

RESTRICTIONS: 1) [A], regarded as a 10-digit left-justified number, must be greater than [B], similarly regarded. Otherwise, the Overflow Alarm will be set, and no division will take place.

2) An attempt to divide by zero will set Overflow Alarm, and no division will take place.

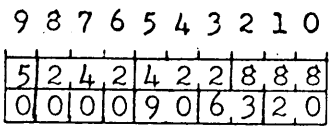
3) If [A] is more than 9 digits long, division will be performed as though the right-most digit were zero.

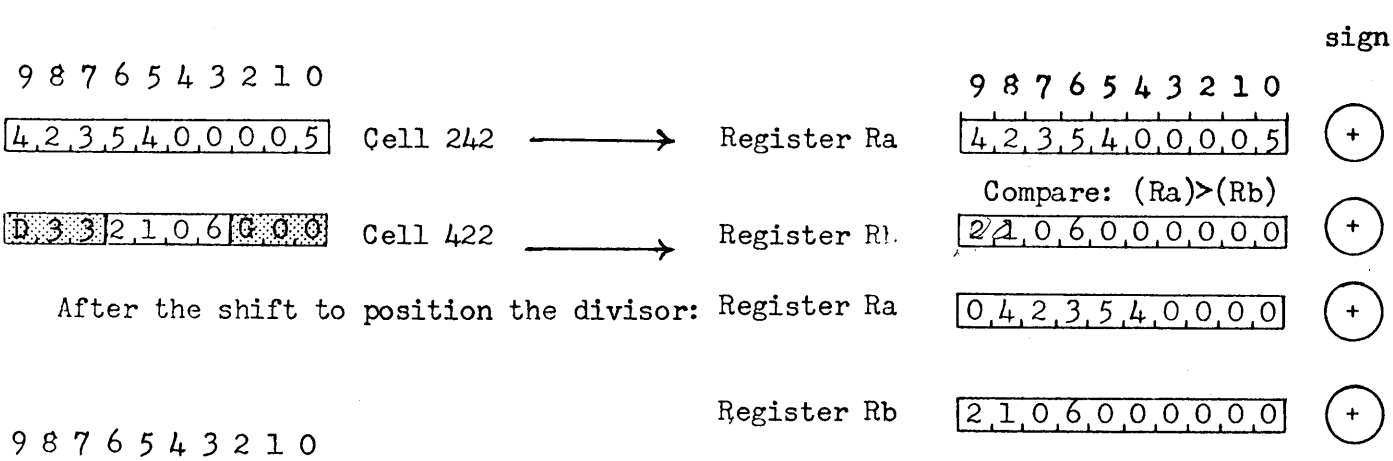
OVERFLOW ALARM: Will be set if restrictions 1 or 2 are violated.

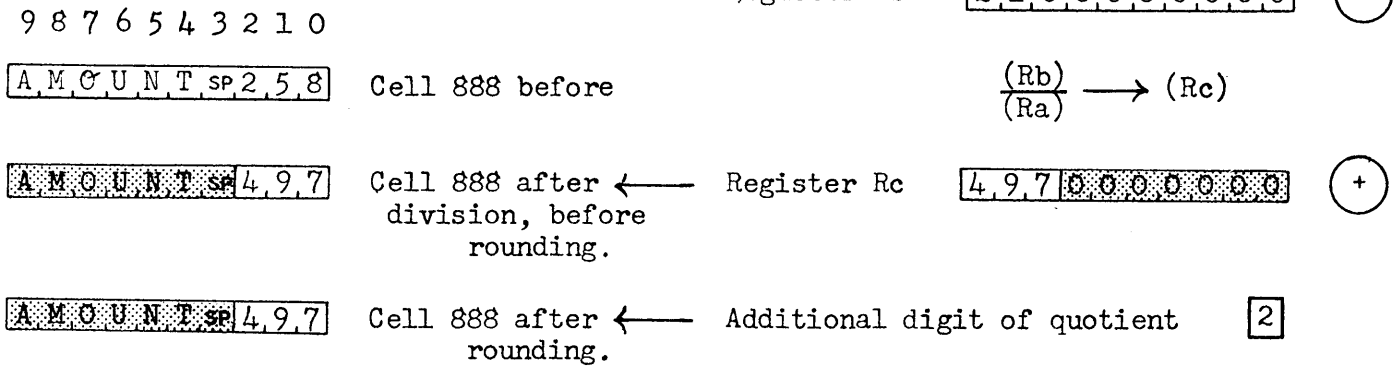
Rounding may cause a carry to be generated out of $C_L$; if so, the Overflow Alarm will be set. The carry will be lost.

## Example - DIVIDE LEFT-JUSTIFIED

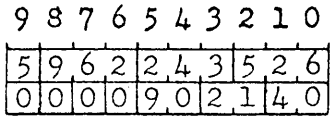
Example 1 - Divide a positive number by a positive number.

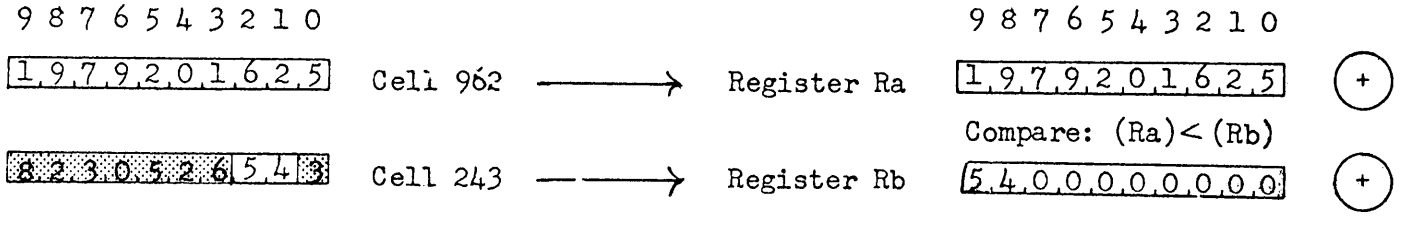



Example 2 - Restriction violated, overflow alarm set.




Compare: (Ra) < (Rb) the overflow alarm will be set, and no division takes place.

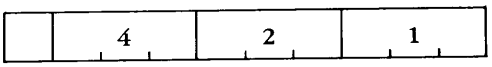
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

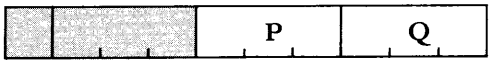
S-values of syllables

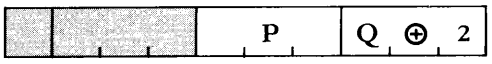
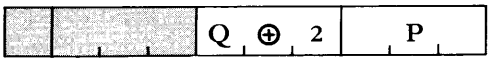



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

## MODIFY ADD

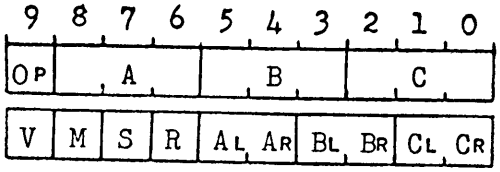
This instruction performs the special kind of addition required when the quantities to be added are addresses of memory locations, or address-type tallies. This kind of addition is described and illustrated in Chapter III.

The respective contents of the A-field and the B-field are right-justified and added, 3 characters at a time, using this special kind of addition.

The result is stored, right-justified, in the C-putaway field.

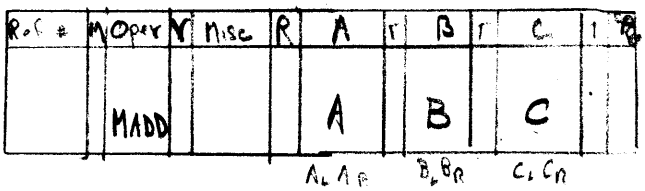
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: MODIFY ADD (MADD)

Operation Code: 6




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of addend.

AL, AR: locations of left-most and right-most digits, respectively, of the addend.

B: base of address of augend.

BL, BR: locations of left-most and right-most digits, respectively, of the augend.

C: base of address in which sum is to be stored.

CL, CR: locations of left-most and right-most digits, respectively, of field allocated to the sum.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: MODIFY ADD

This is a right-justified operation.

[A] ⊕ [B] ⟶ [C]

[A] is transferred, right-justified, to a 12-character-long working register (Ra). [B] is transferred, right-justified, to another 12-character-long working register (Rb).

In the course of both these transfers, the zone-bits of the right-most two characters in each group of three, are replaced by 0-bits.

The contents of registers Ra and Rb yield four sums, each of which is generated in the indicated field of a third 12-character-long working register (Rc).

| 4 | 3 | 2 | 1 |
|---|---|---|---|
| (Ra:11,9) | (Ra:86) | (Ra:53) | (Ra:20) |
| ⊕ | ⊕ | ⊕ | ⊕ |
| (Rb:11,9) | (Rb:86) | (Rb:53) | (Rb:20) |
| (Rc:11,9) | (Rc:86) | (Rc:53) | (Rc:20) |

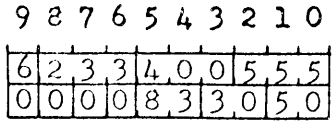
For detailed description of the operation designated by the symbol ⊕, see discussion in Chapter III. Note that any "carry" which may result from any of the four sums is ignored, and does not affect the next sum.

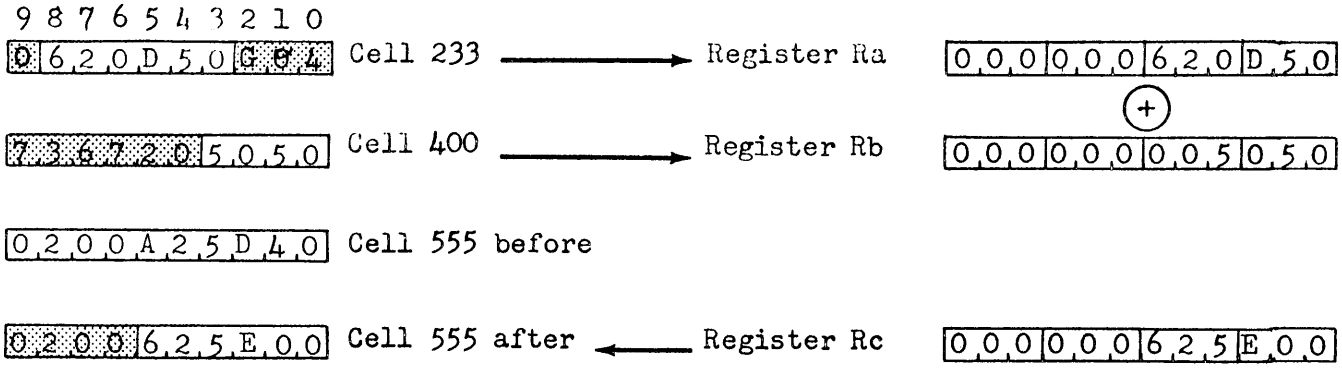
The contents of Rc then replace [C], right-justified.

OVERFLOW ALARM: Will not be set by this Instruction.

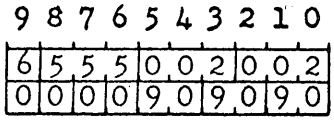
Examples - MODIFY ADD

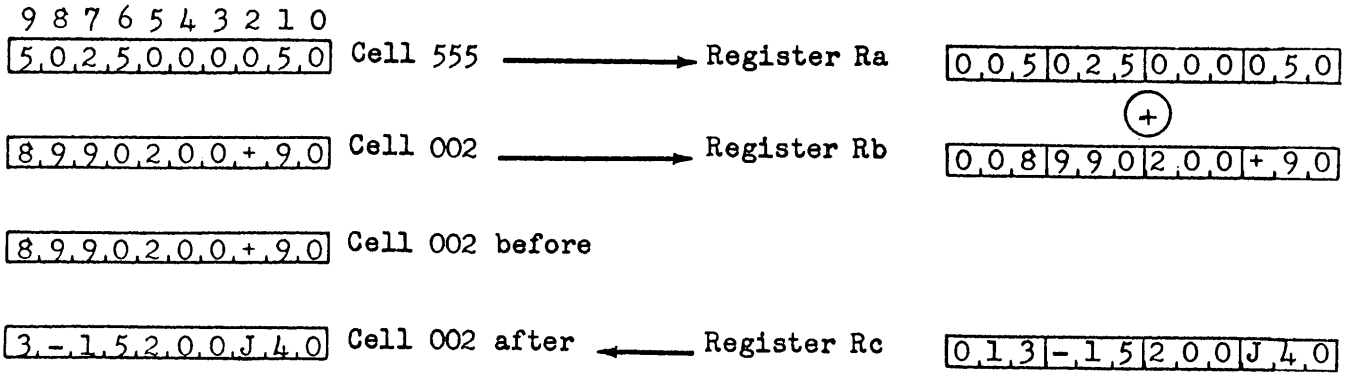
Example 1 -




Example 2 -




Overflow Alarm will not be set.

Example 3 -

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 4 | 5 | 6 | 2 | 5 | 0 | 3 | 3 | 0 |
| 0 | 0 | 0 | 0 | 9 | 0 | 9 | 0 | 9 | 0 |

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Z | Z | 8 | 4 | * | + | - | M | N | P | Cell 456 | ⟶ | Register Ra | 0 0 9 | Z 8 4 | * 0 0 | M 5 7 |
| | | | | | | | | | | | | | | ⊕ | | |
| 2 | 0 | 8 | 8 | 2 | 2 | 7 | A | B | C | Cell 250 | ⟶ | Register Rb | 0 0 2 | 0 8 8 | 2 2 7 | A 2 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Cell 330 before | | | | | | |
| 1 | 0 | 7 | 2 | S | 2 | 7 | V | 8 | 0 | Cell 330 after | ⟵ | Register Rc | 0 1 1 | 0 7 2 | S 2 7 | V 8 0 |

Overflow Alarm will not be set.

Example 4 -

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 4 | 5 | 0 | 0 | 0 | 5 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 5 | 1 | 5 | 1 | 5 | 1 |

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | Cell 450 | ⟶ | Register Ra | 0 0 0 | 0 0 0 | 0 1 2 | 0 1 2 |
| | | | | | | | | | | | | | | ⊕ | | |
| 0 | 0 | 0 | 0 | A | 3 | 0 | B | 3 | 0 | Cell 005 | ⟶ | Register Rb | 0 0 0 | 0 0 0 | 0 1 3 | 0 2 3 |
| 0 | 0 | 0 | 0 | A | 3 | 0 | B | 3 | 0 | Cell 005 before | | | | | | |
| 0 | 0 | 0 | 0 | 2 | 5 | 0 | 3 | 5 | 0 | Cell 005 after | ⟵ | Register Rc | 0 0 0 | 0 0 0 | 0 2 5 | 0 3 5 |

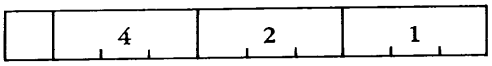
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

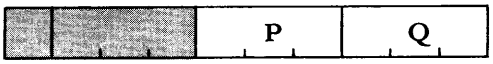
S-values of syllables

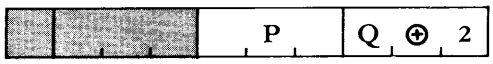
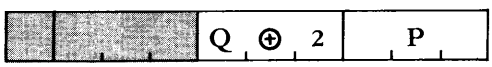



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# MODIFY SUBTRACT

This Instruction performs an operation which is the reverse of the special kind of addition called MODIFY ADD. This operation is described and illustrated in Chapter III.

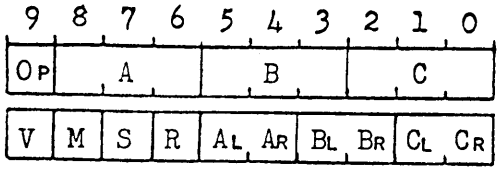
The respective contents of the A-field and the B-field are right-justified, and the difference [A] θ [B] is generated, 3 characters at a time, using this special kind of subtraction.

The result is stored, right-justified, in the C-putaway field.

If any 3-character field of [A] should represent a smaller address-type number than the corresponding 3-character field of [B], then the result of [A] θ [B] will be the "4000-complement" of [B] θ [A].

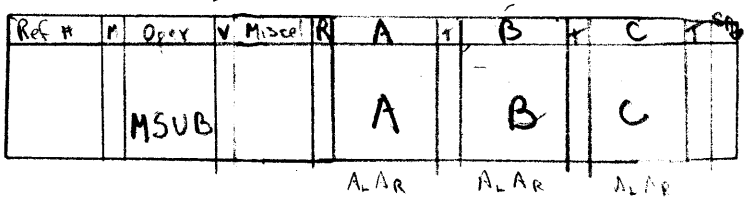
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: MODIFY SUBTRACT (MSUB)

Operation Code: 7




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of minuend.

AL, AR: locations of left-most and right-most characters, respectively, of the minuend.

B: base of address of subtrahend.

BL, BR: locations of left-most and right-most characters, respectively, of the subtrahend.

C: base of address in which difference is to be stored.

CL, CR: locations of left-most and right-most characters, respectively, of field allocated to the difference.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: MODIFY SUBTRACT

This is a right-justified operation.

[A] $\ominus$ [B] $\longrightarrow$ [C].

[A] is transferred, right-justified, to a 12-character-long working register (Ra). [B] is transferred, right-justified, to another 12-character-long working register (Rb).

In the course of both these transfers, the zone-bits of the right-most two characters in each group of three, are replaced by 0-bits.

The contents of registers Ra and Rb yield four subtractions, each of which is generated in the indicated field of a third 12-character-long working register (Rc).

| 4 | 3 | 2 | 1 |
|---|---|---|---|
| (Ra:11,9) | (Ra:86) | (Ra:53) | (Ra:20) |
| $\ominus$ | $\ominus$ | $\ominus$ | $\ominus$ |
| (Rb:11,9) | (Rb:86) | (Rb:53) | (Rb:20) |
| (Rc:11,9) | (Rc:86) | (Rc:53) | (Rc:20) |

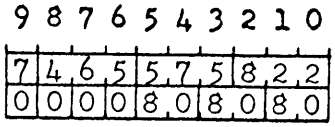
For detailed description of the operation designated by the symbol $\ominus$, see discussion in Chapter III. Note that any "borrow" which may result from any of the four subtractions is ignored, and does not affect the next subtraction.

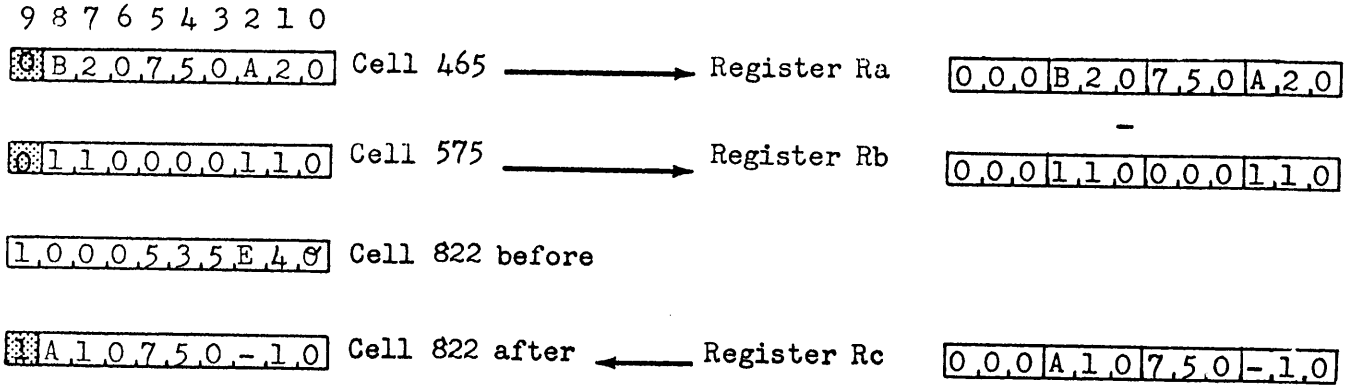
The contents of Rc then replace [C], right-justified.

OVERFLOW ALARM: Will not be set by this Instruction.

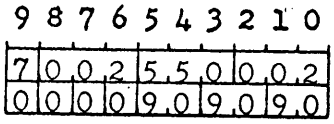
Examples - MODIFY SUBTRACT

Example 1 -




Example 2 -



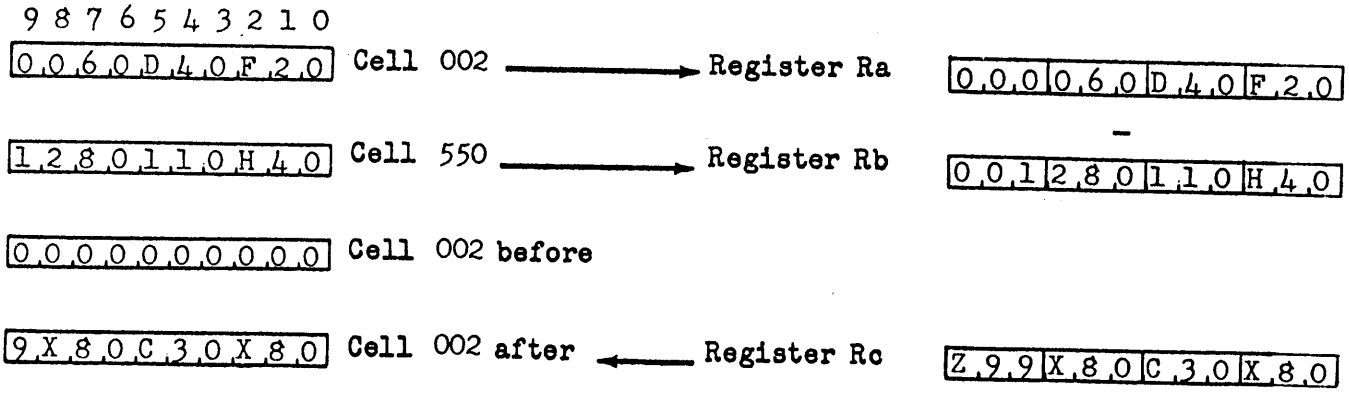



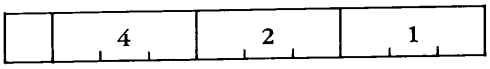
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

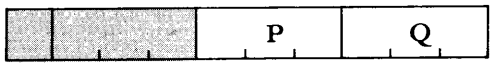
S-values of syllables

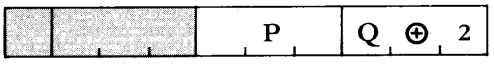
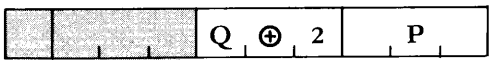



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# E X T R A C T

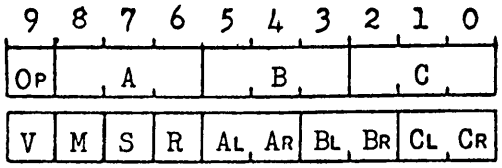
This Instruction permits the contents of any designated collection of bit-positions within a cell to be "extracted" from that cell, so that they may be examined without regard to the balance of the contents of the cell.

An important use of this Instruction arises when a number of characteristics of an Item are coded in binary form, so that the Item may be easily classified according to any set of categories.

This Instruction will operate equally well upon numeric information which has been "packed" by the PACK Instruction.

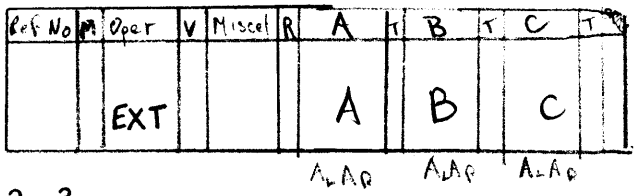
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: EXTRACT (EXT)

Operation Code: 8




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first operand ("source").

AL, AR: locations of left-most and right-most characters, respectively, of first operand.

B: base of address of second operand ("extractor").

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

C: base of address in which result is to be stored.

CL, CR: locations of left-most and right-most characters, respectively, of the result.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: EXTRACT

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra).

[B] is transferred, right-justified, to a 10-character-long working register (Rb).

A third 10-character-long working register, (Rc), is filled with zeros.

Each of the sixty bits in Rb is examined: wherever there is a 1-bit in Rb the corresponding 0-bit in Rc is replaced by the corresponding bit, 0 or 1, in Ra; wherever there is a 0-bit in Rb, the corresponding 0-bit in Rc is unaltered.

The contents of Rc replace [C], right-justified.

It should be observed that the result generated in Rc is the bit-by-bit "logical product" of the contents of registers Ra and Rb. This operation is symmetrical, in that the result will be unchanged if [A] and [B] are interchanged.

OVERFLOW ALARM: Will not be set by this Instruction.

Example - EXTRACT

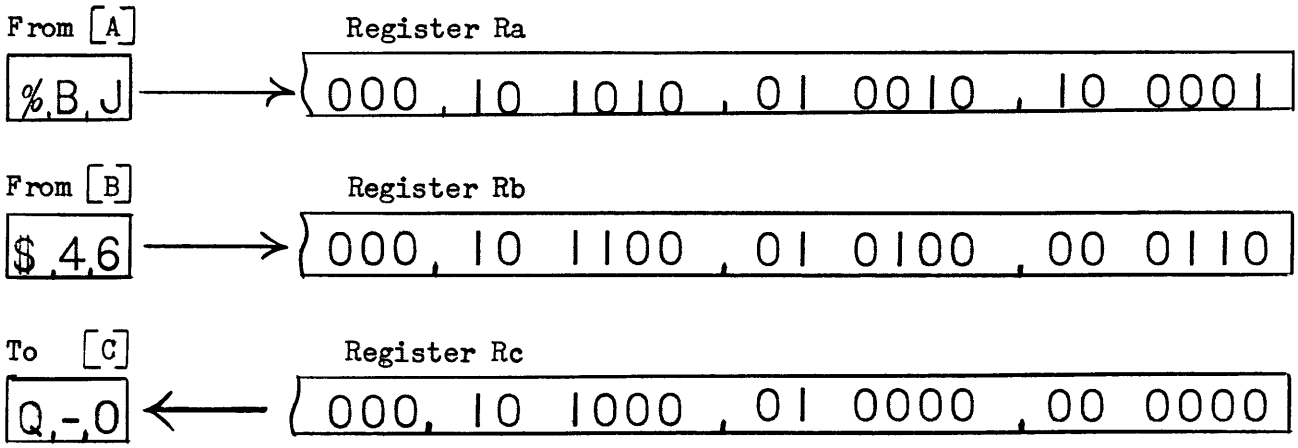


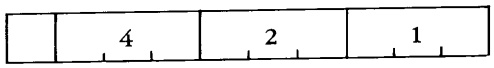
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

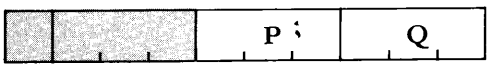
S-values of syllables

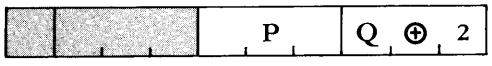
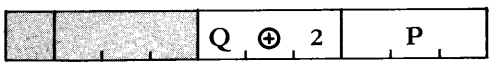



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# I N S E R T

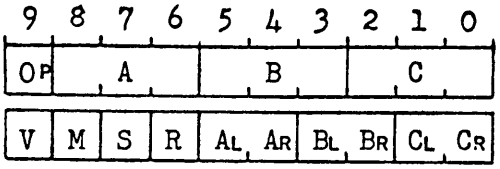
This Instruction permits a set of new values of any designated collection of bit-positions within a cell to be "inserted" into that cell, without disturbing the balance of the contents of the cell.

An important use of this Instruction arises when a number of characteristics of an Item are coded in binary form, and one of the characteristics is changed, requiring a change in the binary coding.

This Instruction will operate equally well upon numeric information which has been "packed" by the PACK Instruction.

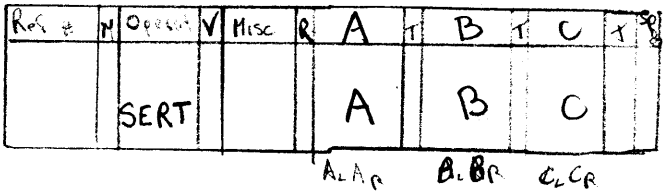
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




**Operation: INSERT (SERT)**

**Operation Code: 9**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first operand ("source").

AL, AR: locations of left-most and right-most characters, respectively, of first operand.

B: base of address of second operand ("inserter").

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

C: base of address of third operand, and result.

CL, CR: locations of left-most and right-most characters, respectively, of third operand and result.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: INSERT

[A], [B], [C] are each transferred, right-justified, to a 10-character-long working register (Ra, Rb, Rc, respectively).

Each of the sixty bits in Rb is examined: wherever there is a 1-bit in Rb, the corresponding bit in Rc is replaced by the corresponding bit, 0 or 1, in Ra; wherever there is a 0-bit in Rb, the corresponding bit in Rc is unaltered.

The contents of Rc then replace [C], right-justified.

If [C] happens to be zero, then the operation of INSERT is the same as that of EXTRACT.

OVERFLOW ALARM: Will not be set by this Instruction.

Example - INSERT

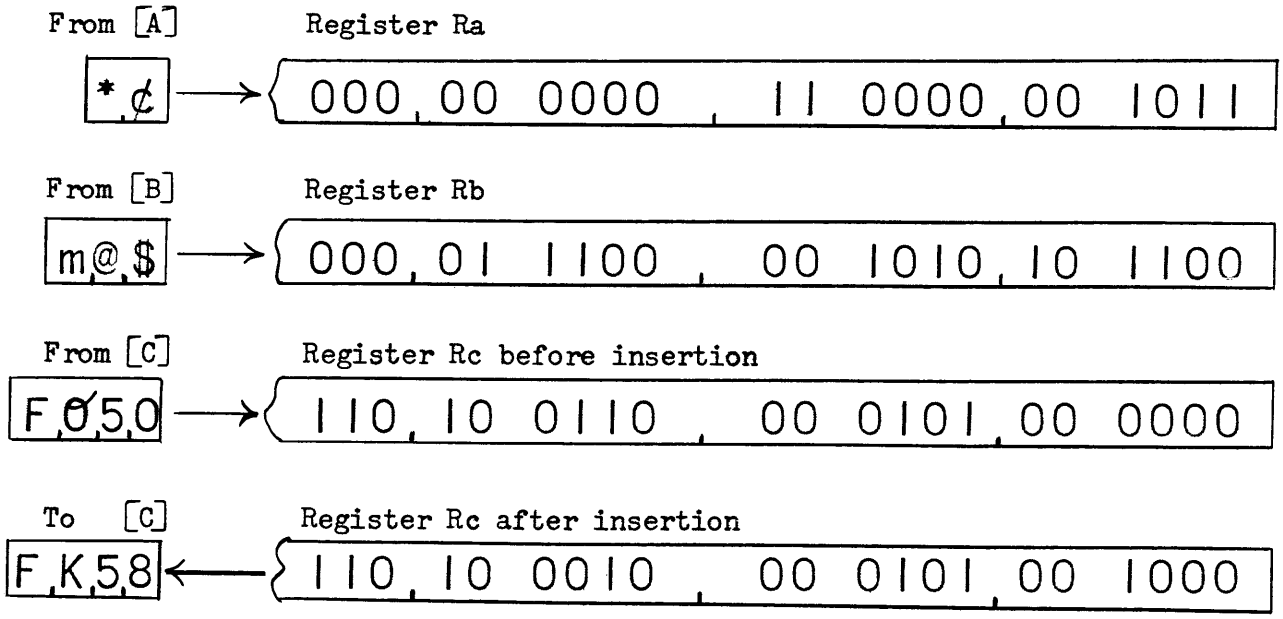


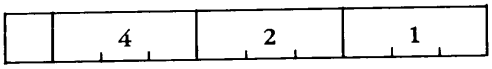
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

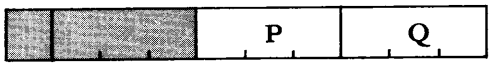
S-values of syllables

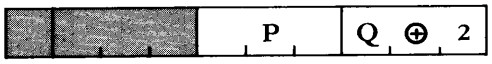
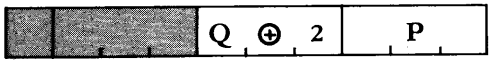



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# ADD BINARY

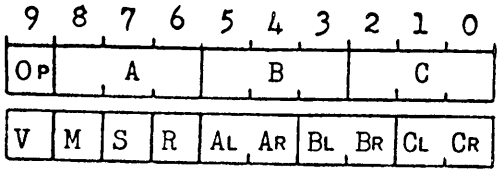
The respective contents of the A-field and the B-field, right-justified, are treated as two 60-bit binary numbers. Their binary sum is generated, and stored from right to left in the C-putaway field.

"Normal" variation: The Instruction operates as just described. One of the principal uses for this operation will be to add a binary-coded field to itself, and thus achieve a binary left shift of one bit-position. In this way, characteristics which have been recorded in binary-coding, may be successively tested.

"Mod-64" variation: In this variation, binary addition takes place character by character, with no carry between characters. In this manner, the individual characters of a sorting-key may be transformed into pseudo-characters, which will sort into a sequence different from the normal sequence of the 304 Language Code. Normal sorting routines may then be used to achieve any desired sorting sequence.

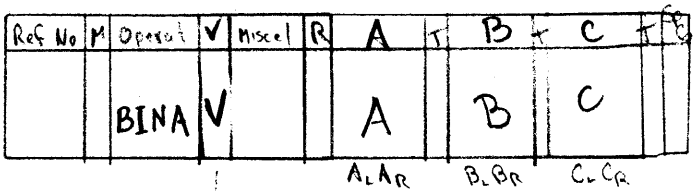
OVERFLOW ALARM: Will be set if the sum exceeds the capacity of the C-field.

INSTRUCTION FORMAT:




Operation: **ADD BINARY** (BINA)

Operation Code: □




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of addend.

$A_L$, $A_R$: positions of left-most and right-most characters, respectively, of addend.

B: base of address of augend.

$B_L$, $B_R$: positions of left-most and right-most characters, respectively, of augend.

C: base of address in which the binary sum is to be stored.

$C_L$, $C_R$: positions of left-most and right-most characters, respectively, of the field allocated for the sum.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Normal | BINA |
| 1 | Mod-64 | BINA:M |

DESCRIPTION OF: ADD BINARY

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb).

The contents of registers Ra and Rb are considered as positive binary numbers, each 60-bits in length.

V = 0: The contents of registers Ra and Rb are added binarily, the 60-bit binary sum being generated in a third working register (Rc), 11 characters long.

V = 1: The binary addition is performed as in V = 0, except that no carry is generated between character-positions in register Rc. Thus the transformation of each character is independent of the result of the transformation of any other character.

The contents of Rc then replace [C], right-justified.

OVERFLOW ALARM: Will be set if some character (other than "zero") is not transferred from Rc to [C].

ADDITION TABLE for Add Binary (Bit-by-bit addition)

| ⊥ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | c0 |

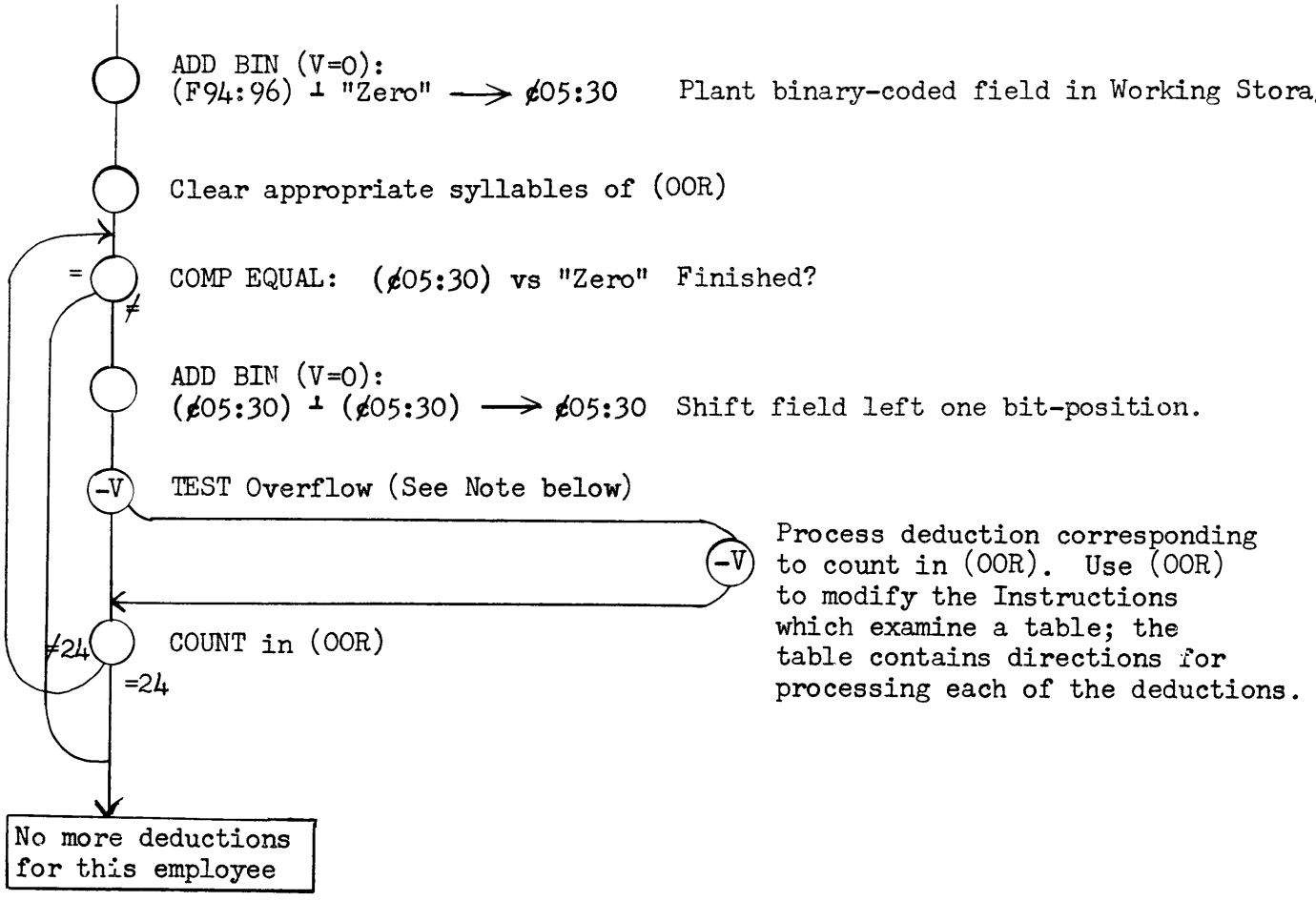
"⊥" is the special symbol used to designate binary addition.

"c" indicates carry into the next bit-position.

Example - ADD BINARY, V = 0. ("Normal")

In processing an industrial payroll, there are 24 possible deductions from an employee's Gross Pay. In each employee's payroll file, a 4-character field has been allocated for recording the presence or absence of each of these deductions; each of the 24 bit-positions in this field corresponds to one deduction. A 1-bit in any position indicates that the corresponding deduction is to be made, while a 0-bit indicates that the deduction is not to be made. The 24 bit-positions are assigned to the deductions in their relative order of frequency, with the most frequently-used deduction assigned the left-most bit-position.

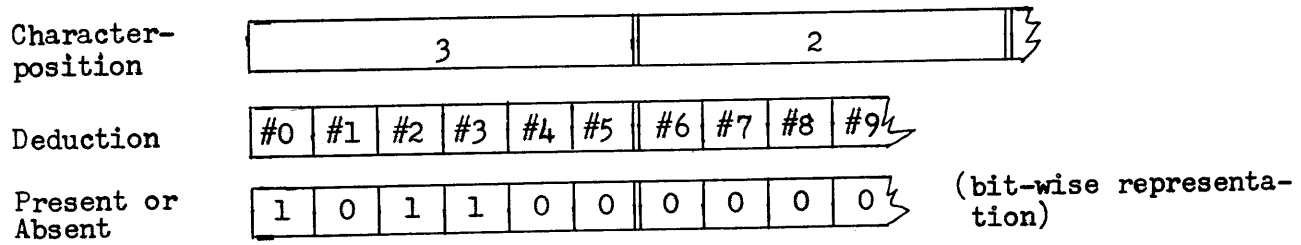
An employee's payroll file has been read from Magnetic Tape into Memory; F94:96 contains the deduction code. The following flow-chart indicates the program to test for the presence of each deduction, and terminate when the last deduction in an individual file has been detected.



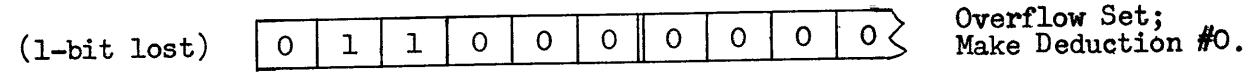

NOTE: If the left-most bit of (¢05:30) is a 1-bit, then the operation of ADD BINARY (shift the field left one bit-position) will result in Overflow.

Suppose that deductions #0, #2, #3 are to be made from the Gross Pay of a particular employee.

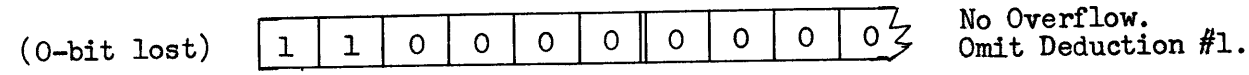
Then, at the start of the testing loop, (¢05:30) will appear as:



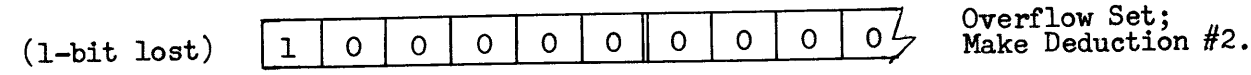

The effect of adding a binary field to itself is to shift the field 1 bit-position to the left. The left-most bit is lost, and a 0-bit appears at the right end. After the first ADD BINARY, (¢05:30) will contain:



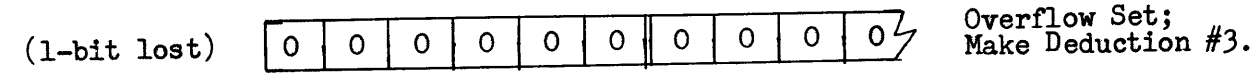

After the second ADD BINARY, (¢05:30) will contain:




After the third ADD BINARY, (¢05:30) will contain:




After the fourth ADD BINARY, (¢05:30) will contain:




But now, as the operation comes around to the beginning of the loop again, and executes the COMPARE EQUALITY, (¢05:30) = 0, and no further testing is done.

Example - ADD BINARY, V = 1. ("mod-64")

Required: To sort data according to some key (part number, account number, &c), using standard Sorting Routines, despite the fact that the following sorting sequence (which is different from the Processor sorting sequence) is required:

"space", "-", letters, digits.

This requires one Instruction as part of regimenting the input for sorting:
ADD BINARY (V=1): Sorting key ⊥ "UUU...U" ⟶ Sorting key.

Each character of the sorting key is independently transformed into a pseudo-character in such a way that, when the pseudo-key is sorted, the items will be arranged in the desired order:

| Character | | Pseudo-Character |
|---|---|---|
| 0 | ⟶ | U |
| 1 | ⟶ | V |
| : | | : |
| 9 | ⟶ | v |
| : | | : |
| space | ⟶ | 0 |
| : | | : |
| : | | : |
| - | ⟶ | 4 |
| A | ⟶ | 5 |
| : | | : |
| N | ⟶ | I |
| : | | : |
| Z | ⟶ | ( |

In the Master File, the items will be stored with their keys in pseudo-characters; all Searching, Copying and posting will be performed with pseudo-keys, and it is only necessary to reverse the transformation, from pseudo-key back to original key, for output printing.

The transformation is reversed by including one additional Instruction in the Editing Routine: A field of all "spaces" is added to the pseudo-key, using this variation of ADD BINARY.

More elaborate transformations into pseudo-keys may be performed by using TEST BIT and COMPLEMENT BINARY, in addition to the "mod-64" variation of ADD BINARY. These are illustrated under COMPLEMENT BINARY.

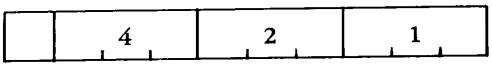
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

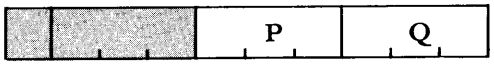
S-values of syllables

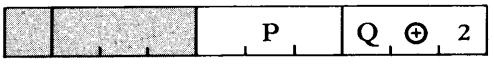
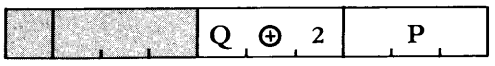



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

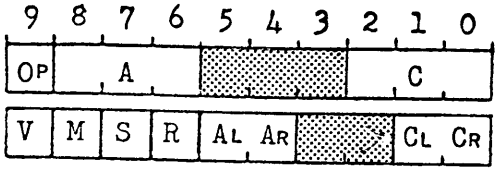
## C O M P L E M E N T  B I N A R Y

The contents of the A-field, right-justified, are treated as a 60-bit binary field. This field is complemented by changing all its 1-bits into 0-bits, and all its 0-bits into 1-bits, and is stored from right to left in the C-putaway field.

If the C-field is longer than the A-field, it will be filled out to the left with "x's" (code 11 1111).

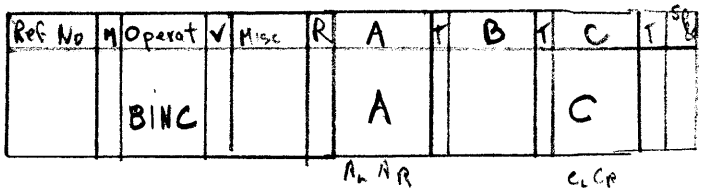
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




**Operation: COMPLEMENT BINARY** (BINC)

**Operation Code: △** (V = 1)




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of field which is to be complemented.

AL, AR: positions of left-most and right-most characters, respectively, of A-field.

C: base of address in which the complemented A-field is to be stored.

CL, CR: positions of left-most and right-most characters, respectively, of field allocated for the complement.

V: variation designator;
V = 1 specifies COMPLEMENT BINARY.

DESCRIPTION OF: COMPLEMENT BINARY

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra).

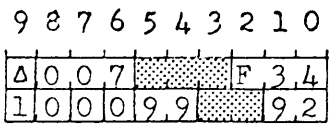
The contents of Register Ra is then transferred to another 10-character-long working register (Rc). In the course of this transfer, all 1-bits are replaced by 0-bits, and all 0-bits are replaced by 1-bits.

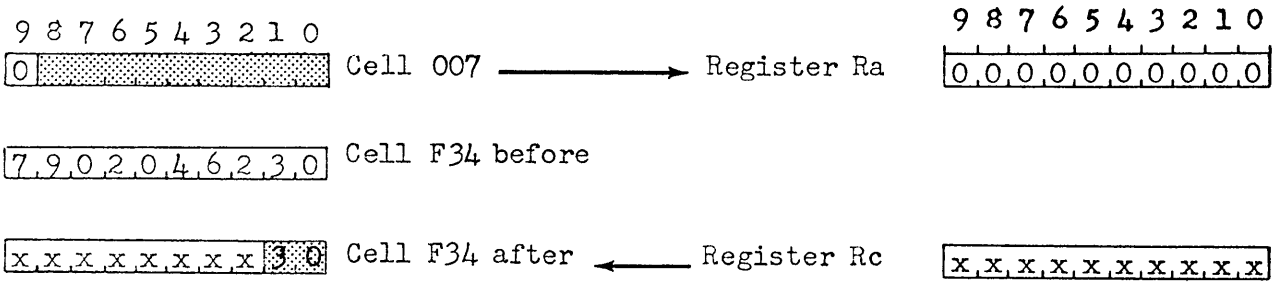
The contents of Register Rc then replace [C], right-justified.

OVERFLOW ALARM: Will not be set by this Instruction.

## Examples - COMPLEMENT BINARY

Example 1 - Generate a field of all 1-bits ("xxxxxxxx") in F34:92, given that (007:99) = 0.




---

Example 2 - The sales records of 5000 salesmen are to be sorted, so that the salesman with the greatest dollar-amount of sales shall head the list, and the salesman with the lowest dollar-amount of sales shall be at the bottom of the list.

In order to conserve Magnetic Tape storage space for the file, and also to permit the Sort to proceed most rapidly, all numeric information in each salesman's record has been PACKED.

Required, therefore, to transform the sorting key (the field containing PACKED dollar-amount of sales) so that normal Sorting Routines will sort the list into reverse order.

To do this, it is necessary merely to COMPLEMENT BINARY the packed dollar-amount in each record, then sort. When the sort has been completed, the dollar-amounts are re-COMPLEMENTED, and UNPACKED for printing.

Example 3. While the following problem, as such, will not be encountered often, it probably presents as complex a sorting transformation as the programmer will ever be required to perform, and it is therefore given as an illustrative example.

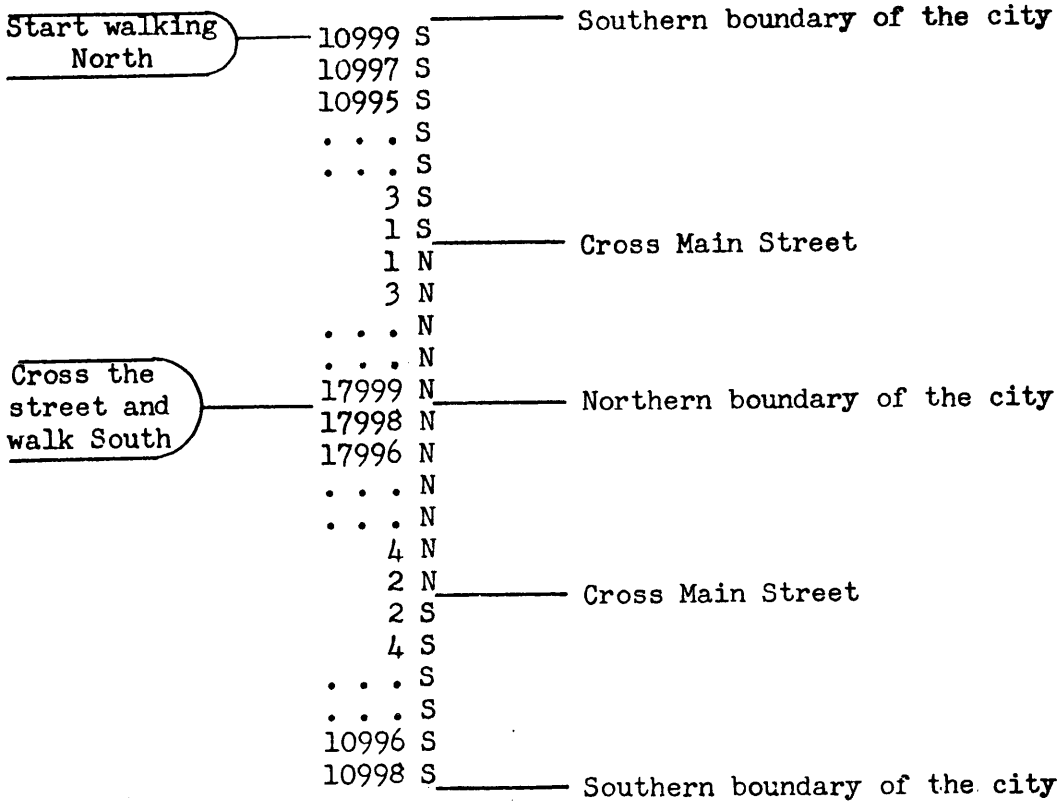
Required:

Using standard Sorting Routines, sort a collection of street-addresses alphabetically by name of street. The house-numbers on each street are to be sorted so that by following the sorted list, a person will start at the South end of each North-South street, and at the West end of each East-West street, walk along the odd-numbered side of the street the full length (or width) of the city to its Northern (or Eastern) boundary; then cross the street and walk along the even-numbered side of the street, all the way back to its South (or West) end.

This requires that the house-numbers on any street shall be sorted in the following sequence:

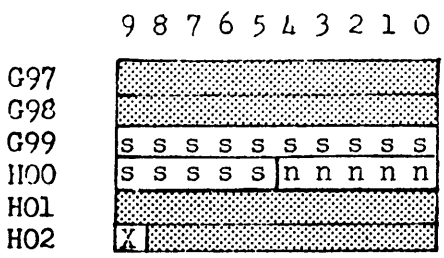
| | | |
|---|---|---|
| "lowest" numbers: | | S or W odd, in reverse sequence |
| next | : | N or E odd, in forward sequence |
| next | : | N or E even, in reverse sequence |
| "highest" numbers: | | S or W even, in forward sequence |

Therefore, the sorted list for North-South streets, for example, will look like this:



Assume:

Each item has been already regimented for sorting:




s.....s represents the street name, filled out to the right with spaces.

nnnnn represents the 5-digit house-number, filled out to the left with zeros.

X represents one of the letters N, E, S, or W which has been stored in an otherwise unused position.

Note that, for the MERGE, (G99:90) is the major key, and (H00:90) is the minor key. N, E, S, or W is not part of the key.

The codes for the letters representing the four directions are:

N: 10 0101
E: 01 0101
S: 11 0010
W: 11 0110

TEST BIT Instructions, coded as follows, will cause a Jump under the conditions shown:

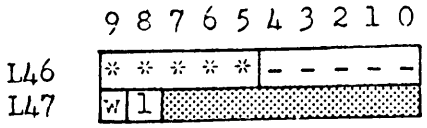
| A-field | B-field | Jump if | Step if |
|---|---|---|---|
| (H02:99) | vs "1" | N or E | S or W |
| "w" | vs (H00:00) | Even | Odd |
| (H00:00) | vs "1" | Odd | Even |

There are many ways of performing the desired sorting transformation by transforming the house-numbers. The most straight-forward set of transformations is:

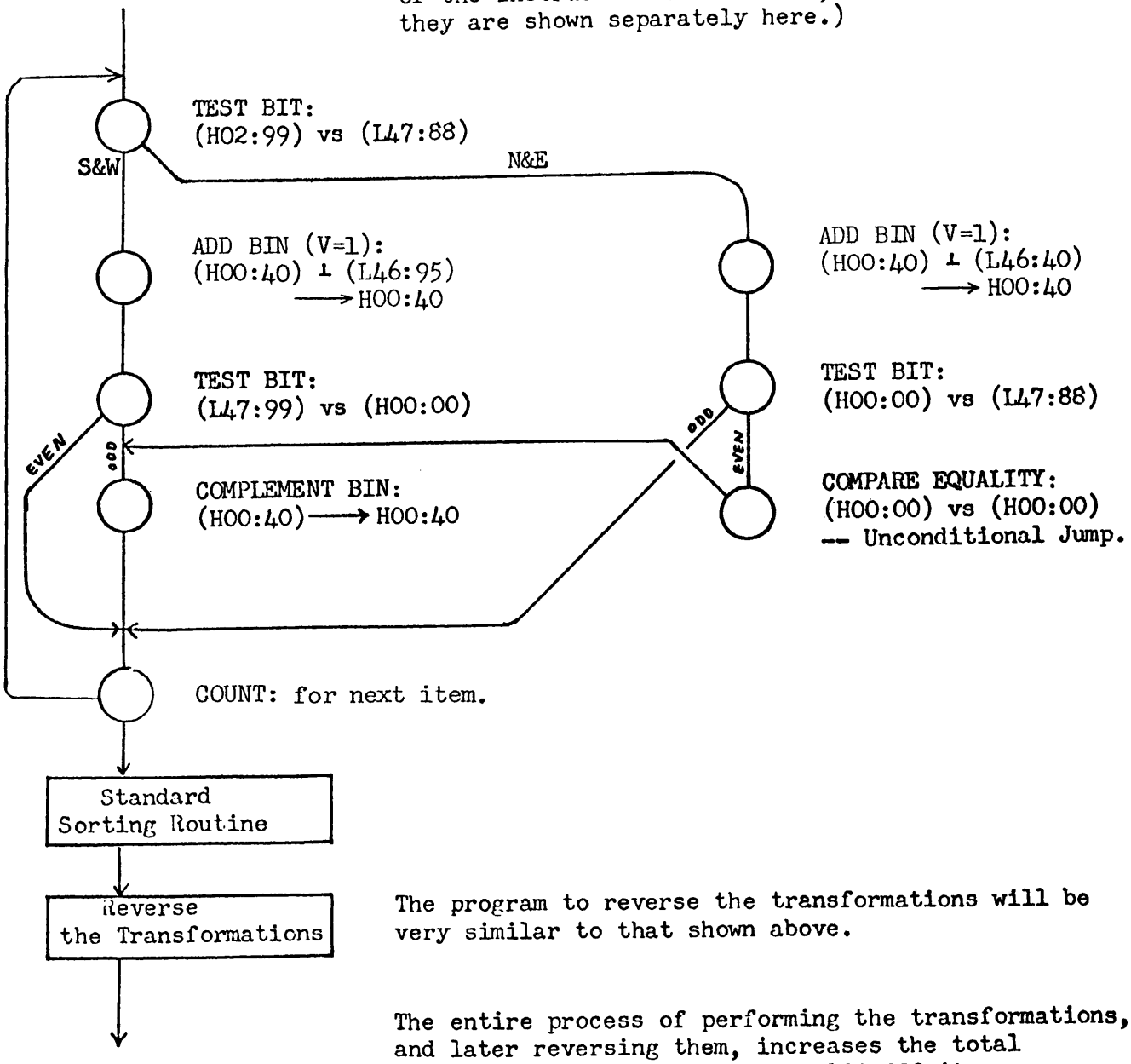
| | | |
|---|---|---|
| S & W odd : | Zone 0, backward: | Add "******", Complement. |
| N & E odd : | Zone 1, forward : | Add "------". |
| N & E even: | Zone 2, backward: | Add "------", Complement |
| S & W even: | Zone 3, forward : | Add "******". |

It must be understood that wherever, in this example, the operation "Add" is mentioned, it is to be understood as ADD BINARY (V = 1).

The following Constants are required:




(In actually coding this problem, the "w" and "1" would be stored in irrelevant character-positions of the Instructions. However, for simplicity, they are shown separately here.)




The program to reverse the transformations will be very similar to that shown above.

The entire process of performing the transformations, and later reversing them, increases the total sorting time by 10 minutes per 100,000 items.

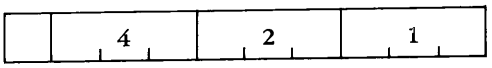
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

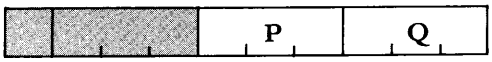
S-values of syllables

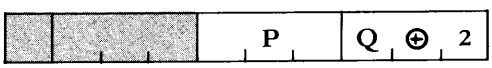
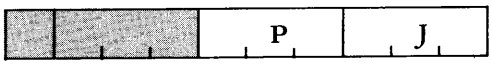
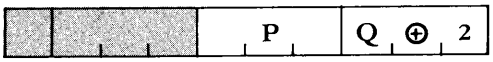
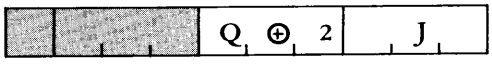



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation



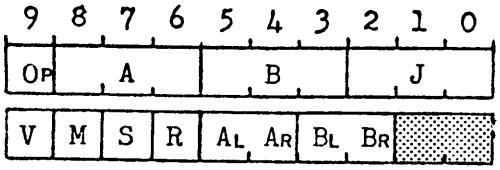

| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# T E S T  B I T

This Instruction is used to test for the presence or absence of any conditions which have been expressed and recorded in binary coding.

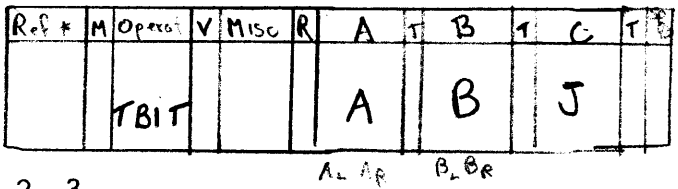
Thus, categories such as payment status of Accounts Receivable; availability of machines for loading; ability of machines to perform desired operations; geographic location, credit status, classification, &c of customers -- may all be selected singly or in combination with the TEST BIT Instruction.

INSTRUCTION FORMAT:




Operation: TEST BIT (TBIT)

Operation Code: A




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of first operand.

AL, AR: locations of left-most and right-most characters, respectively, of first operand.

B: base of address of second operand.

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

J: base of address of next instruction if tested condition is met.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: TEST BIT

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra); [B] is transferred, right-justified to another 10-character-long working register (Rb).

The respective contents of registers Ra and Rb are compared, bit-position by bit-position. The comparison may be regarded in either of two ways, which are logically equivalent:

I- The instruction whose address is designated by J will be executed next, if the result of the comparison is one of the following:

a) Wherever there is a 1-bit in Rb, there is a 1-bit in the corresponding bit-position in Ra.

b) Rb contains all 0-bits.

The next instruction in sequence will be executed next, if for <u>any</u> 1-bit in Rb there is a 0-bit in the corresponding bit-position of Ra.

II- The instruction whose address is designated by J will be executed next, if the result of the comparison is one of the following:

a) Wherever there is a 0-bit in Ra, there is a 0-bit in the corresponding bit-psotion in Rb.

b) Ra contains all 1-bits.

The next instruction in sequence will be executed next, if for <u>any</u> 0-bit in Ra there is a 1-bit in the corresponding bit-position of Rb.

<u>That is</u>: JUMP IF:

I - [A] contains all the 1-bits of [B].

II - [B] contains all the 0-bits of [A].

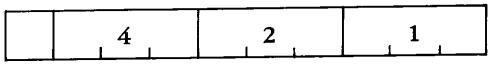
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

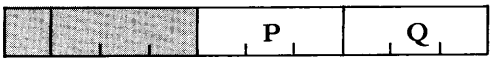
S-values of syllables

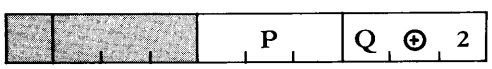
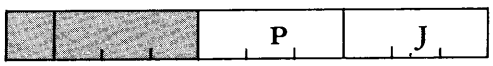
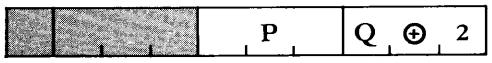
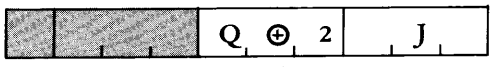



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

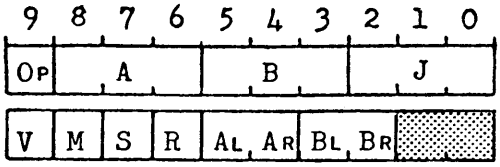
## COMPARE NUMERIC

This Instruction regards the contents of the A- and B-fields as numbers, each with algebraic sign.

If the right-justified A-field contains a greater number than the right-justified B-field, then the next Instruction to be executed will be selected from a "branch" address specified in the Instruction; otherwise, the next Instruction in normal sequence will be executed.

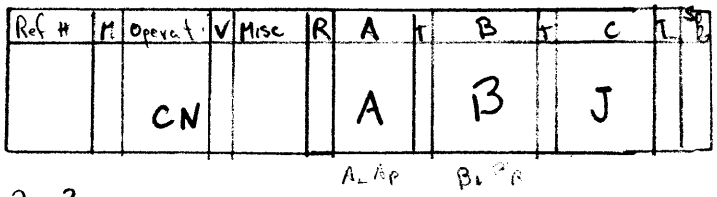
It will be recalled, of course, that any positive number is greater than any negative number; and that, of two negative numbers, the one with the greater magnitude is the smaller number.

INSTRUCTION FORMAT:




Operation: COMPARE NUMERIC (CN)

Operation Code: B




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first operand.

AL, AR: locations of left-most and right-most characters, respectively of first operand.

B: base of address of second operand.

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

J: base of address of next instruction if [A] > [B], numerically.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: COMPARE NUMERIC

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified to another 10-character-long working register (Rb). In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

The algebraic signs of [A] and [B] (as indicated by the respective sign-bits of the characters in positions $A_L$ and $B_L$) are examined, and four cases are recognized:

1- If both signs are positive, the contents of registers Ra and Rb are compared. If (Ra) is a greater number than (Rb), then the next Instruction to be executed will be selected from the address specified by J. If (Ra) is less than or equal to (Rb), the next Instruction in sequence will be executed.

2- If both signs are negative, the contents of registers Ra and Rb are compared. If (Ra) is a smaller number than (Rb), then the next Instruction to be executed will be selected from the address specified by J. If (Ra) is greater than or equal to (Rb), the next Instruction in sequence will be executed.

3- If the sign of [A] is positive, and the sign of [B] is negative, the next Instruction to be executed will be selected from the address specified by J.

4- If the sign of [A] is negative, and the sign of [B] is positive, the next Instruction in sequence will be executed.

NOTE: Positive zero will always be a greater number, for the purpose of this Instruction, than negative zero.

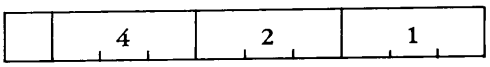
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

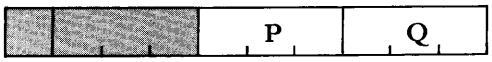
S-values of syllables

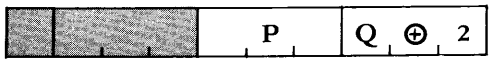
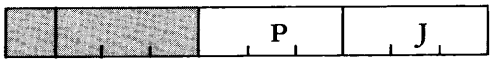
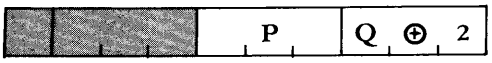
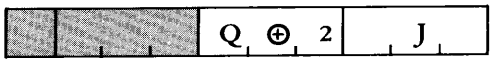



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

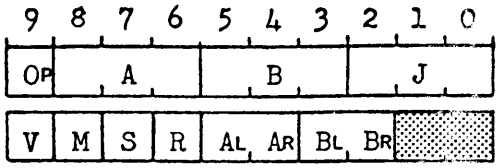
## C O M P A R E  A L P H A - N U M E R I C

This Instruction regards the contents of the A- and B-fields as alpha-numeric information.

If the right-justified contents of the A-field follows the right-justified contents of the B-field, according to the "sorting sequence" shown in Table IV-1, then the next Instruction to be executed will be selected from a "branch" address specified in the Instruction; otherwise, the next Instruction in normal sequence will be executed.

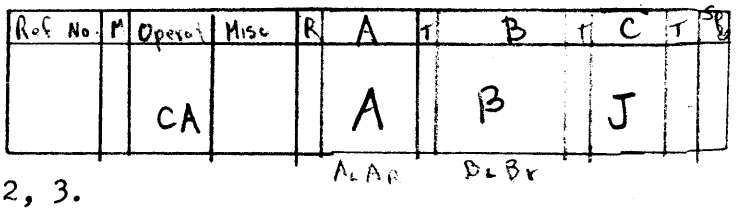
If positive numbers have been PACKED, the result of COMPARE ALPHA-NUMERIC will be the same as if the information were in unpacked form.

INSTRUCTION FORMAT:




Operation: COMPARE ALPHA-NUMERIC (CA)

Operation Code: C




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first operand.

AL, AR: locations of left-most and right-most characters, respectively, of first operand.

B: base of address of second operand.

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

J: base of address of next Instruction if [A] follows [B], in the Processor's sorting sequence. i.e.- if [A]>[B], when both are regarded as positive binary numbers.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: COMPARE ALPHA-NUMERIC

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb).

The contents of registers Ra and Rb are regarded as positive, 60-bit-long binary numbers, and are compared.

If (Ra) is a greater binary number than (Rb), then the next Instruction to be executed will be selected from the address specified by J; otherwise, the next Instruction in sequence will be executed next.

Since the "sorting sequence" of the 64 characters which are significant to the Processor (as shown in Table IV-1) is the same as the numeric sequence of their binary configurations, the comparison has the effect of determining whether or not (Ra) follows (Rb) according to the alpha-numeric sorting sequence of the Processor.

If the COMPARE ALPHA-NUMERIC Instruction is applied to signed numeric quantities, a negative number will always follow a positive number in the sorting sequence.

An important consequence of the correspondence between the binary sequence and the alpha-numeric sorting sequence, arises in connection with the PACK Instruction (Code 07), applied to positive numeric information. After this information has been PACKed, the relative significance of the binary bits comprising that information is unchanged, and COMPARE ALPHA-NUMERIC will always give precisely the same result as if the information had not been packed.

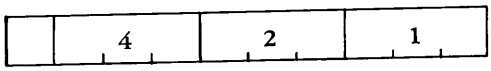
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

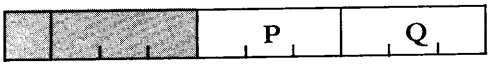
S-values of syllables

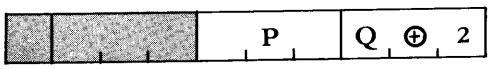
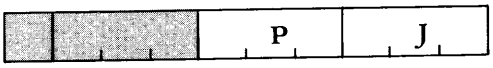
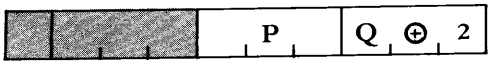
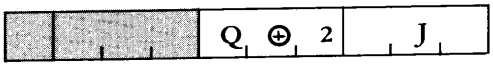



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation



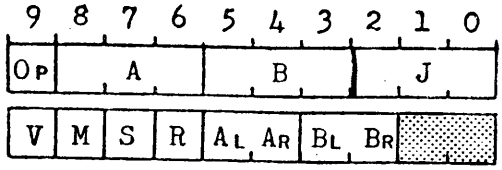

| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

## COMPARE EQUALITY

This Instruction right-justifies, and compares, the contents of the A- and B-fields.

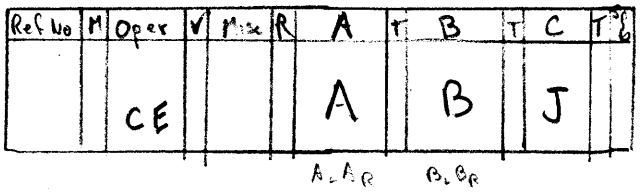
If the contents of both fields are the same, then the next Instruction to be executed will be selected from a "branch" address specified in the Instruction; otherwise, the next Instruction in the normal sequence will be executed.

INSTRUCTION FORMAT:




Operation: COMPARE EQUALITY (CE)

Operation Code: D




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

A: base of address of first operand.

AL, AR: locations of left-most and right-most characters, respectively, of first operand.

B: base of address of second operand.

BL, BR: locations of left-most and right-most characters, respectively, of second operand.

J: base of address of next Instruction if [A] = [B], in every bit-position.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: COMPARE EQUALITY

This is a right-justified operation.

[A] is transferred, right-justified, to a 10-character-long working register (Ra). [B] is transferred, right-justified, to another 10-character-long working register (Rb).

The respective contents of registers Ra and Rb are compared, bit-position by bit-position.

If the contents of the two registers are identical in each of the 60-bit positions, then the next Instruction to be executed will be selected from the address specified by J; otherwise, the next Instruction in sequence will be executed next.

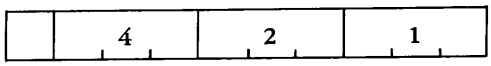
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | T syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

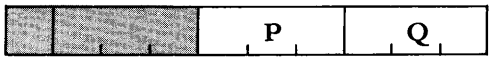
S-values of syllables

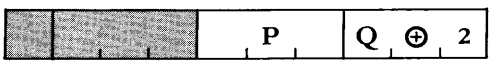
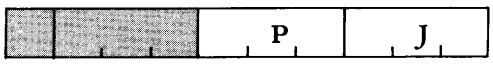
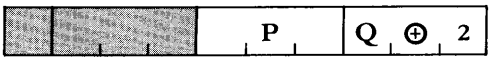
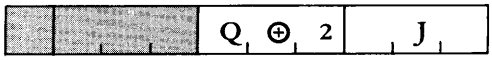



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# C O U N T

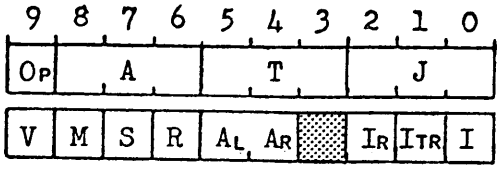
This Instruction combines the functions of several standard Processor operations within a single operation, and operates on the contents of any one of the 10 memory locations 000 through 009.

The Instruction uses MODIFY ADD to augment a selected field of one of these 10 cells, using as an augmenter the right-justified contents of the A-field named in the Instruction. Then a selected 3-character field of the augmented cell is compared with a 3-character Tester, which is part of the Instruction itself. The tested field within the augmented cell need have no relationship to the augmented field within that cell.

If the two 3-character fields being compared DO NOT contain the same address-type number, then the next Instruction to be executed will be selected from a "branch" address specified in the Instruction; if the two 3-character fields DO contain the same address-type number, then the next Instruction in the normal sequence will be executed.

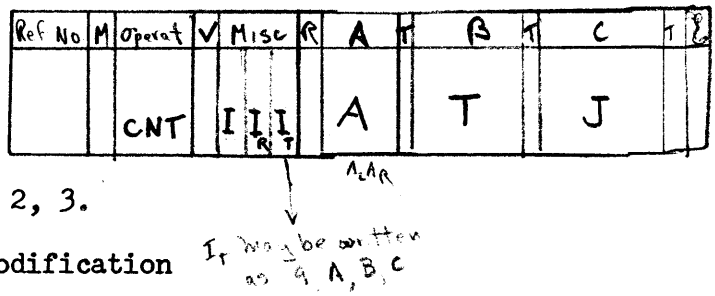
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




**Operation: COUNT (CNT)**

**Operation Code: E**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of quantity by which the designated field of cell OOI is to be augmented.

AL, AR: locations of left-most and right-most characters, respectively, of the augmenter.

I: designates OOI as the cell whose contents are to be augmented.

IR: location of right-most character of the field of OOI which is to be augmented.

T: as modified by (OOR), this <u>is</u> the Tester.

ITR: location of right-most character of 3-character tested field of cell OOI.

J: base of address of next Instruction if T, as modified by (OOR), and the tested field of OOI <u>DO</u> <u>NOT</u> contain the same address-type number.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: COUNT

This is a right-justified operation, consisting of two separate steps:

1) MODIFY ADD: (OOI:$I_L I_R$) $\oplus$ [A] $\longrightarrow$ OOI:$I_L I_R$. $I_L$ is not specified by the programmer, since the Processor always adds to the 9-$I_R$ field of (OOI).

2) COMPARE: (OOI:$I_{TL} I_{TR}$) vs T. $I_{TL}$ is not specified by the programmer, since the Processor always makes $I_{TL} = I_{TR} + 2$, and compares a 3-character field of (OOI).

[A] is transferred, right-justified, to a 12-character-long working register (Ra).

(OOI), from position 9 through position $I_R$, i.e. - [OOI], is transferred, right-justified, to another 12-character-long working register (Rb).

In the course of both these transfers, the zone-bits of the right-most two characters in each group of three are replaced by 0-bits.

The contents of registers Ra and Rb are then added, using the special kind of addition described under MODIFY ADD, and the result is stored in a third 12-character-long working register (Rc).

The contents of Rc then replace [OOI], right-justified. Starting at $I_R$ with information in OOI to right of $I_R$ remaining unchanged.

Next, (OOI), from position 9 through position $I_{TR}$, is transferred, right-justified, to a 3-character-long working register (Rd). The contents of the 3-character field in the first word of the Instruction (the field called T) are modified (if appropriate) by (OOR), and transferred, right-justified, to another 3-character-long working register (Re). In the course of both these transfers, the zone-bits of the right-most two characters are replaced by 0-bits.

If, at this point, the contents of Rd and Re are identical, the next Instruction in the normal sequence will be executed next. If the contents of Rd and Re are not identical, the Instruction whose address is designated by J will be executed next.

OVERFLOW ALARM: Will not be set by this Instruction.

Examples- COUNT

Example 1 - Double the quantity stored in field 86 of the 2nd word of each of ten 3-word items. Store the results in field 63 of each of 10 words.

THREE-WORD ITEMS

| | 9 8 7 6 5 4 3 2 1 0 |
|---|---|
| 700 | |
| 701 | 1 2 4 |
| 702 | |
| 703 | |
| 704 | 0 1 0 |
| 705 | |
| 706 | |
| 707 | 7 1 6 |
| 708 | |
| 709 | |
| 710 | 5 5 8 |
| 711 | |
| 712 | |
| 713 | 0 5 2 |
| 714 | |
| 715 | |
| 716 | 0 0 0 |
| 717 | |
| 718 | |
| 719 | 4 0 4 |
| 720 | |
| 721 | |
| 722 | 4 6 1 |
| 723 | |
| 724 | |
| 725 | 4 5 0 |
| 726 | |
| 727 | |
| 728 | 1 0 7 |
| 729 | |

INSTRUCTIONS

| | | |
|---|---|---|
| 273 | 2 0 0 2 0 0 2 0 0 2 | Clear location 002. |
| 274 | 0 0 0 0 9 0 9 0 9 0 | |
| 275 | 1 7 0 1 7 0 1 2 0 1 | Perform the addition. |
| 276 | 0 0 7 2 8 6 8 6 6 3 | Select the operands relative to 002. |
| 277 | E 2 5 0 0 1 0 2 7 5 | Augment (002) and test for end. |
| 278 | 0 0 0 0 6 0 2 0 0 2 | |
| 250 | 9 7 7 3 0 0 3 0 0 1 | Augmenter |

RESULTS OF ADDITIONS

| | 9 8 7 6 5 4 3 2 1 0 |
|---|---|
| 201 | 0 2 4 8 |
| 202 | 0 0 2 0 |
| 203 | 1 4 3 2 |
| 204 | 1 1 1 6 |
| 205 | 0 1 0 4 |
| 206 | 0 0 0 0 |
| 207 | 0 8 0 8 |
| 208 | 0 9 2 2 |
| 209 | 0 9 0 0 |
| 210 | 0 2 1 4 |

Example 2 - Same as Example 1, except:

Perform the additions, starting at the bottom of the list of items. Store the sums, starting at the top of the list of results. Use a different method of clearing (002).

INSTRUCTIONS

| | | |
|---|---|---|
| 273 | 1 2 7 4 2 7 4 0 0 2 | Clear location 002. |
| 274 | 0 0 0 0 8 8 8 8 9 0 | |
| 275 | 1 7 2 8 7 2 8 2 0 1 | Perform the addition. |
| 276 | 0 0 7 2 8 6 8 6 6 3 | Select the operands relative to 002. |
| 277 | E 2 5 0 Z 7 0 2 7 5 | Augment (002) and test for end. |
| 278 | 0 0 0 0 8 0 9 0 6 2 | |
| 250 | 0 Z 9 7 Z 9 7 0 0 1 | Augmenter |

RESULTS OF ADDITIONS

| | 9 8 7 6 5 4 3 2 1 0 |
|---|---|
| 201 | 0 2 1 4 |
| 202 | 0 9 0 0 |
| 203 | 0 9 2 2 |
| 204 | 0 8 0 8 |
| 205 | 0 0 0 0 |
| 206 | 0 1 0 4 |
| 207 | 1 1 1 6 |
| 208 | 1 4 3 2 |
| 209 | 0 0 2 0 |
| 210 | 0 2 4 8 |

It should be observed that (002) is being COUNTED "backward" in the A and B syllables, and "forward" in the C syllable.

TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

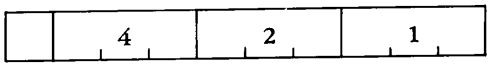
S-value

4 — N syllable modified

1 — J syllable modified

Sum — Determines combination of syllables modified.

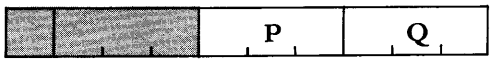
S-values of syllables

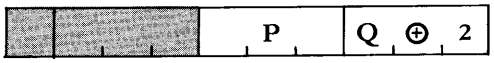
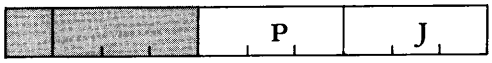
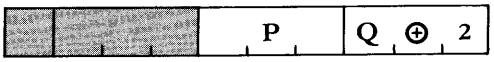
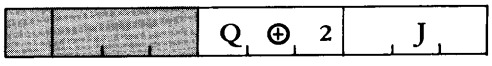



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# T E S T

This Instruction really embodies four different Instructions.

1) TEST Overflow. If any Instruction sets the Overflow Alarm, the Processor will immediately halt unless the next Instruction is TEST Overflow; in that case, the Overflow Alarm will be turned off, and the next Instruction to be executed will be selected from a "branch" address.

If the programmer anticipates that Overflow might occur, this feature permits him to provide for this exceptional case automatically, without requiring operator intervention during processing.

2) TEST Option Switch. This feature provides convenient facilities for permitting operator intervention during processing. If the designated one of ten Console Option Switches is ON, the next Instruction to be executed will be selected from a "branch" address. This feature is particularly useful for interrogating the files, or in permitting routine processing to be interrupted for high-priority work.

3) TEST Reader Code. The High-Speed Paper-Tape Reader can read any three codes, with the particular code used being selected by a manual switch on the Console. This facility permits the Reader to be compatible with a wide variety of tape-punching devices.

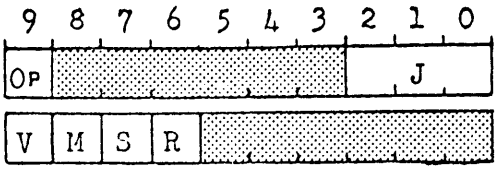
This variation of TEST assures that the operator has set the STET Code-Selection Switch properly, and also provides important audit and protection facilities.

Accounting Machines, recording monetary transactions, can punch a code incompatible with, for example, the code punched, by typewriters recording non-monetary transactions. Therefore, any attempt to simulate a monetary transaction with an unauthorized device can be detected immediately.

4) TEST Punch Code. The High-Speed Paper-Tape Punch can punch any two codes, with the particular code used being selected by a manual switch on the Console. This facility permits the Punch to produce paper-tape which is compatible with tape-to-card converters, typewriters, telegraphic transmitters, &c.

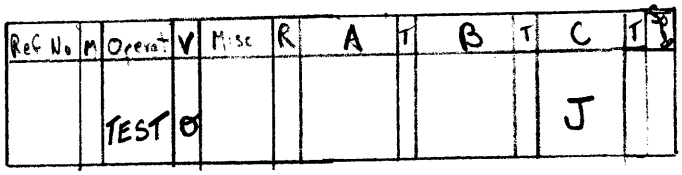
This variation of TEST assures that the operator has set the Punch Code-Selection Switch properly, for the operation to be performed.

INSTRUCTION FORMAT:




## Operation: TEST OVERFLOW (TEST:0)

Operation Code: F (V = 0)




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

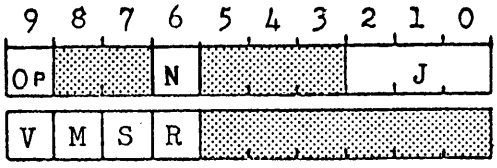
J: base of address of next instruction if Overflow Alarm had been set by the preceding instruction.

If the Overflow Alarm had been set, TEST Overflow turns it off.

V: Variation designator;
V = 0 specifies TEST Overflow.

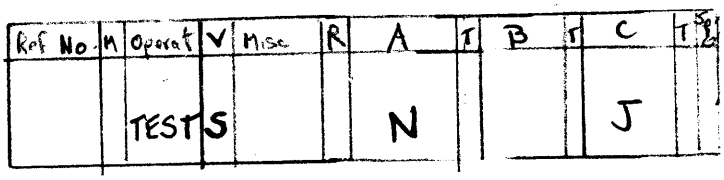
---

INSTRUCTION FORMAT:




## Operation: TEST OPTION SWITCH (TEST·S)

Operation Code: F (V = 2)




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

N: as modified by (OOR), specifies the number of the Console Option Switch to be tested. Switches are numbered from 0 through 9.

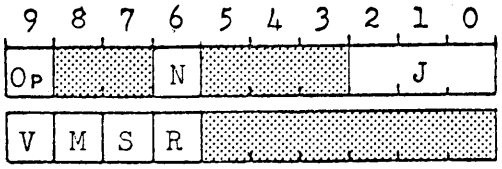
Each Option Switch shows:
Yellow light when ON.
Blue light when OFF.

J: base of address of next instruction if the designated Option Switch is "ON".

The comparison is made against all four numeric bits of N.

V: variation designator;
V = 2 specifies TEST Switch.

INSTRUCTION FORMAT:




Operation: **TEST READER CODE** (TEST:R)

Operation Code: F (V = 4)

| Ref No | M | Operat | V | Misc | R | A | B | C | S |
|---|---|---|---|---|---|---|---|---|---|
| | | TEST | R | | | N | | J | |

DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index register.

N: as modified by (OOR), specifies the setting (0, 1, 2) of the Code-Selection Switch for the High-Speed Paper-Tape Reader.

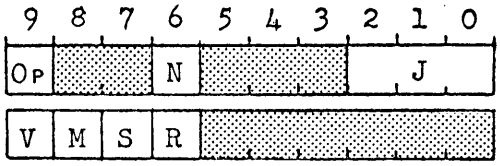
J: base of address of next instruction, if the switch-setting is the same as specified by N.

The comparison is made against all four numeric bits of N.

V: variation designator;
V = 4 specifies TEST Reader Code.

---

INSTRUCTION FORMAT:




Operation: **TEST PUNCH CODE** (TEST:P)

Operation Code: F (V = 6)

| Ref No | M | Operat | V | Misc | R | A | B | C | S |
|---|---|---|---|---|---|---|---|---|---|
| | | TEST | P | | | N | | J | |

DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

N: as modified by (OOR), specifies the setting (0, 1) of the Code-Selection Switch for the High-Speed Paper-Tape Punch.

J: base of address of next instruction, if the switch-setting is the same as specified by N.

The comparison is made against all four numeric bits of N.

V: variation designator;
V = 6 specifies TEST Punch Code.

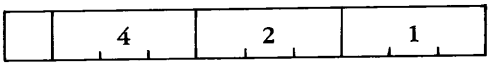
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

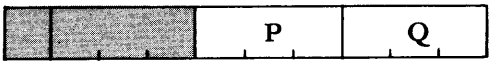
S-values of syllables

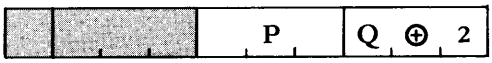
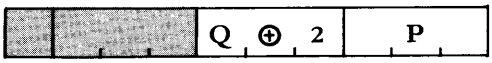



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |   |
| negative |   |

# C O M B I N E

The contents of the A- and B-fields are combined into a single field for the C-putaway.

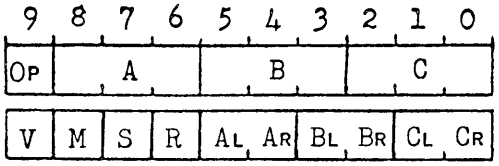
The B-field is put away first, from right to left; then the A-field is put away, from right to left, immediately to the left of the B-field putaway.

If the C-field is shorter than the combined A- and B-fields, characters will be lost from the left end of the A-field. If the C-field is shorter than the B-field alone, then the entire A-field, and characters from the left end of the B-field, will be lost.

If the C-field is longer than the combined A- and B-fields, zeros will be added at the left to fill out the putaway field.

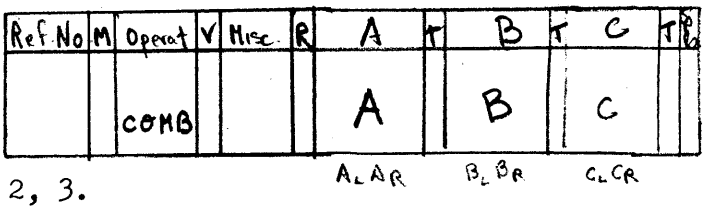
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: **COMBINE** (COMB)

Operation Code: **H**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of left-hand field.

AL, AR: locations of left-most and right-most characters of left-hand field.

B: base of address of right-hand field.

BL, BR: locations of left-most and right-most characters of right-hand field.

C: base of address into which the A- and B-fields will be combined.

CL, CR: locations of left-most and right-most characters of the combined field.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: COMBINE

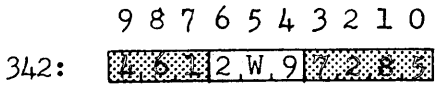
This is a right-justified operation.

[B] is transferred, right-justified, to a 20-character-long working register (Rc). Then [A] is transferred to Rc, right-justified against the left-most character from [B]. In the course of these transfers, all zone-bits are preserved, and the information is unchanged.

The contents of Rc then replace [C], right-justified.

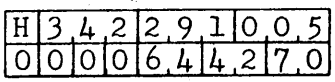
OVERFLOW ALARM: Will not be set.

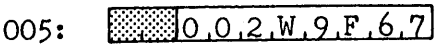
## Examples - COMBINE




---

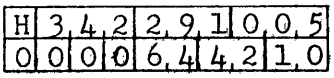
Example 1




---

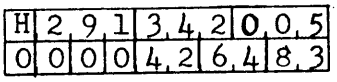
Example 2





---

Example 3



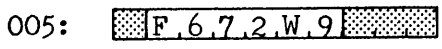



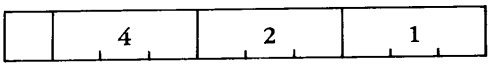
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

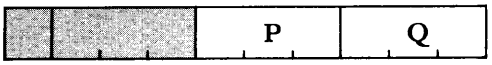
S-values of syllables

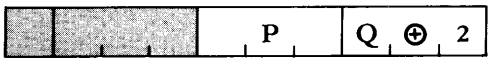
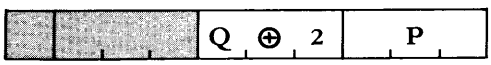



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

THREE INSTRUCTIONS, DISTRIBUTE, SUPPRESS, EDIT, ARE GENERALLY SIMILAR. BASICALLY, EACH OF THEM COPIES THE CONTENTS OF THE A-FIELD INTO THE B- AND C-FIELDS. THE C-PUTAWAY IS MADE FIRST, FROM RIGHT TO LEFT; THEN THE B-PUTAWAY IS MADE, FROM LEFT TO RIGHT. EACH INSTRUCTION HAS TWO VARIATIONS, AND EACH VARIATION CHANGES THE CONTENTS OF THE A-FIELD IN A DIFFERENT MANNER BEFORE MAKING THE PUTAWAYS.

## D I S T R I B U T E

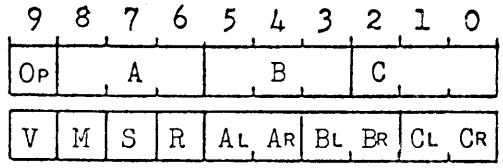
"Normal" variation: The contents of the A-field are unchanged in the putaways. If either putaway field is shorter than the A-field, characters will be lost in that putaway; if either putaway field is longer than the A-field, zeros will be added to fill out that putaway field.

"Sign Split-Off" variation: The sign of the A-field (designated by the 5th bit in the left-most character) is split off, and appears in the putaways as a separate character; a positive sign appears as a "space" character, while a negative sign appears as a "-" character, immediately to the left of the character from which it was split off. The zone-bits of all the original characters from the A-field are replaced by 0-bits in the putaways, leaving only the numeric bits in the B- and C-fields.

The putaways, therefore, are each one character longer than the A-field. If the C-field is longer than this, "space" characters will be added to the left of the sign character; if it is the same length as the A-field, the sign character is lost (once split off, the sign is never recombined with the left-most character); and if the C-field is shorter than the A-field, other characters will be lost from the left end of the field. If the B-field is longer than the putaway, zeros will be added at the right end of the field; if it is too short for the full putaway, characters will be lost from the right end of the field.

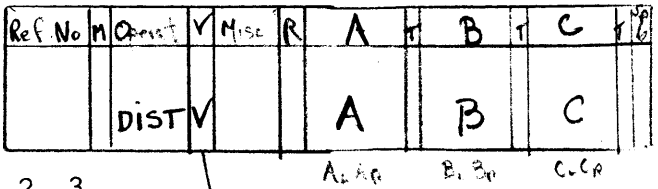
OVERFLOW ALARM: Will not be set by either variation.

INSTRUCTION FORMAT:




Operation: **DISTRIBUTE** (DIST)

Operation Code: **J**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register 00R.

R: designates 00R as index-register.

A: base of address of word containing the field whose contents are to be distributed.

AL, AR: locations of left-most and right-most characters, respectively, of the A-field.

B: base of address of location into which the distributed field is to be placed, left-justified.

BL, BR: locations of left-most and right-most characters, respectively, of the B-putaway field.

C: base of address of location into which the distributed field is to be placed, right-justified.

CL, CR: locations of left-most and right-most characters, respectively, of the C-putaway field.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Normal | DIST |
| 1 | Sign Split-Off | DIST:S |

DESCRIPTION OF: DISTRIBUTE

V = 0: [A] is transferred to a 10-character-long working register (Rc), right-justified, and also to another 10-character-long working register (Rb), left-justified. In the course of these transfers, zone-bits are preserved, and the information is unchanged.

The contents of Rc replace [C], right-justified; then the contents of Rb replace [B], left-justified.

V = 1: This is the "Sign Split-Off" variation.

An 11-character-long working register (Rc), is filled with "space" characters, and then [A] is transferred to Rc, right-justified. [A] is also transferred to another previously cleared, 11-character-long working register (Rb), left-justified at character-position 9.

In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

If the sign-bit of the character in position A4 is a 1-bit (minus sign), then a "minus" character (code 01 0000) is inserted into Rb at character-position 10, and into Rc at the character position immediately to the left of the character transferred from position A4.

If the sign-bit of the character in position A4 is a 0-bit (plus sign), then a "space" character (code 00 0100) is inserted into Rb at character-position 10. (Note that the corresponding character-position in Rc already contains a "space" character.)

The contents of Rc replace [C], right-justified; then the contents of Rb replace [B], left-justified.

OVERFLOW ALARM: Will not be set by either variation.

## DISTINCTIONS

### DISTRIBUTE, SUPPRESS, EDIT

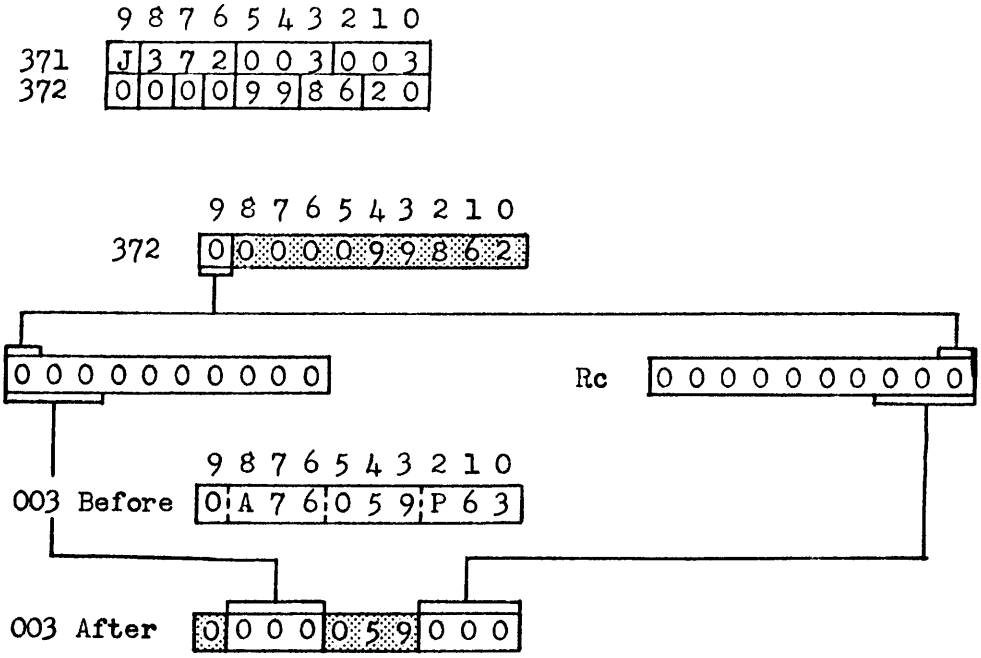
| [A] | Instruction | V | B-Putaway → | ← C-Putaway |
|---|---|---|---|---|
| 00465 | DISTRIBUTE | 0 | 0,0,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | DISTRIBUTE | 1 | SP,0,0,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,0,0,4,6,5 |
| | SUPPRESS | 0 | SP,SP,4,6,5,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | SUPPRESS | 1 | SP,SP,SP,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | EDIT | 0 | SP,SP,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | EDIT | 1 | *,*,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| -00009 | DISTRIBUTE | 0 | -,0,0,0,0,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | DISTRIBUTE | 1 | -,0,0,0,0,0,9,0,0,0 | SP,SP,SP,-,0,0,0,0,0,9 |
| | SUPPRESS | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | SP,SP,SP,SP,-,SP,SP,SP,SP,9 |
| | SUPPRESS | 1 | -,SP,SP,SP,SP,SP,9,0,0,0 | SP,SP,SP,-,SP,SP,SP,SP,SP,9 |
| | EDIT | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | EDIT | 1 | *,*,*,*,*,9,0,0,0,0 | 0,0,0,0,0,0,0,0,0,9 |

EDIT → Sets Overflow Alarm

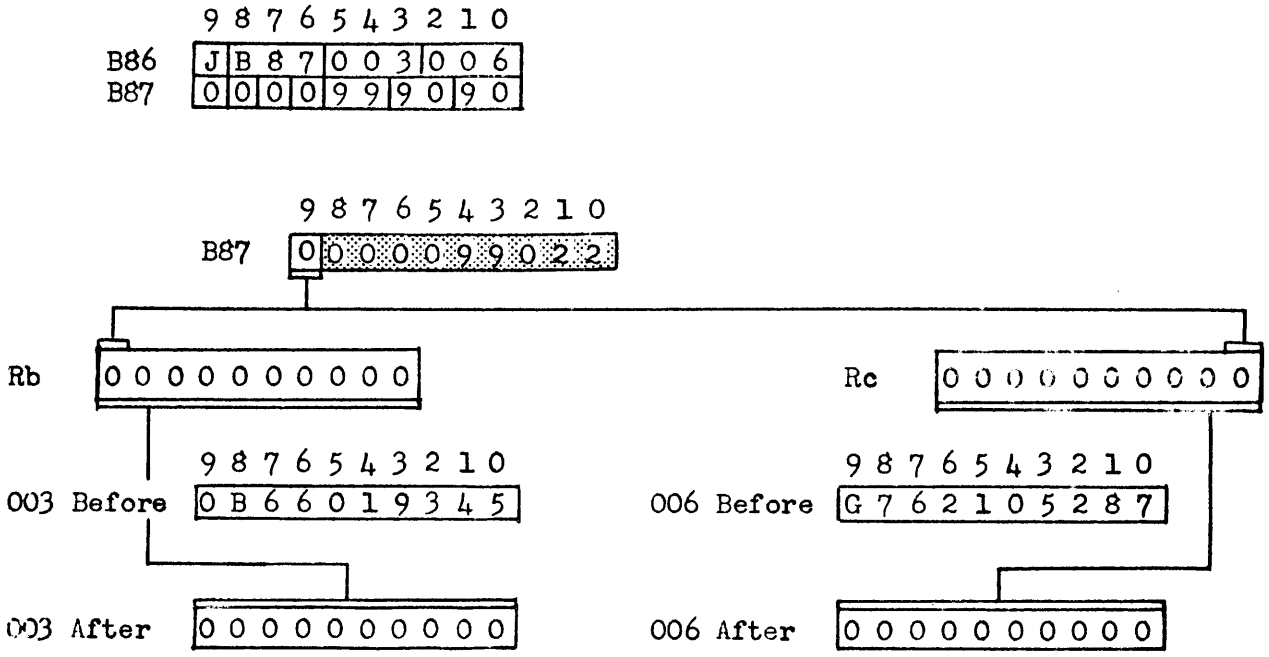
| A | Instruction | V | B-Putaway → | ← C-Putaway |
|---|---|---|---|---|
| G26A9 | DISTRIBUTE | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | DISTRIBUTE | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | SUPPRESS | 0 | G,2,6,A,9,0,0,0,0,0 | SP,SP,SP,SP,SP,G,2,6,A,9 |
| | SUPPRESS | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | EDIT | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | EDIT → Sets Overflow Alarm | 1 | 7,2,6,1,9,0,0,0,0,0 | 0,0,0,0,0,7,2,6,1,9 |
| 0 | DISTRIBUTE | 0 | 0,0,0,0,0,0,0,0,0,0 | 0,0,0,0,0,0,0,0,0,0 |
| | DISTRIBUTE | 1 | SP,0,0,0,0,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,0 |
| | SUPPRESS | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | SUPPRESS | 1 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | EDIT | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | 0,0,0,0,0,0,0,0,0,0 |
| | EDIT | 1 | *,*,*,*,*,*,*,*,*,* | 0,0,0,0,0,0,0,0,0,0 |

"Normal" DISTRIBUTE is an extremely versatile Instruction for performing logical operations. The following examples illustrate some of the techniques to which it lends itself.

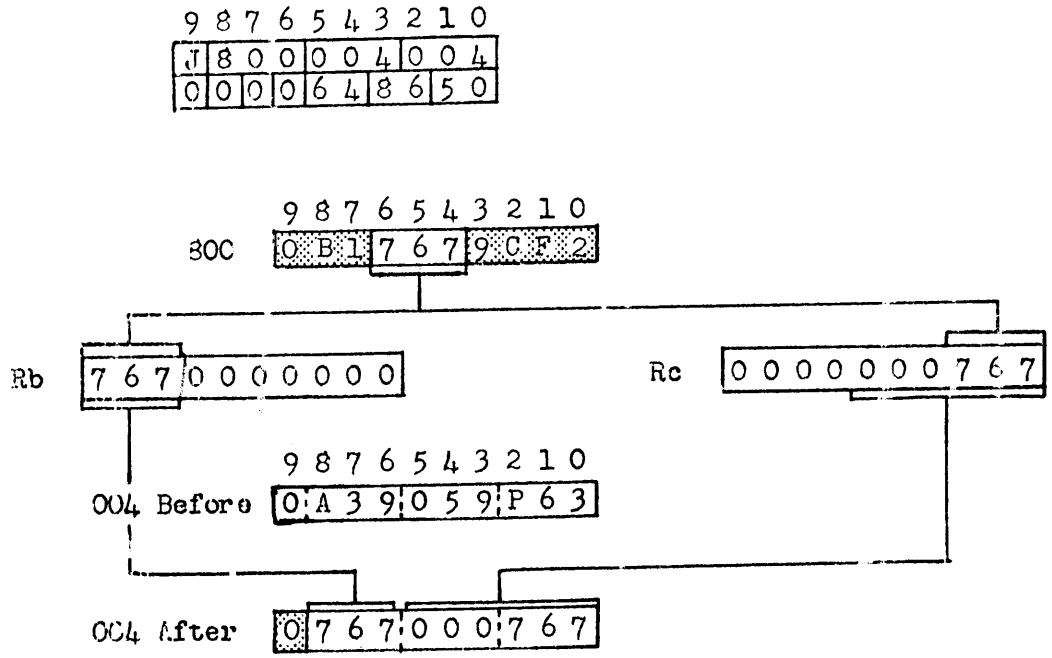
Example 1. Clear two fields of an Index Register.



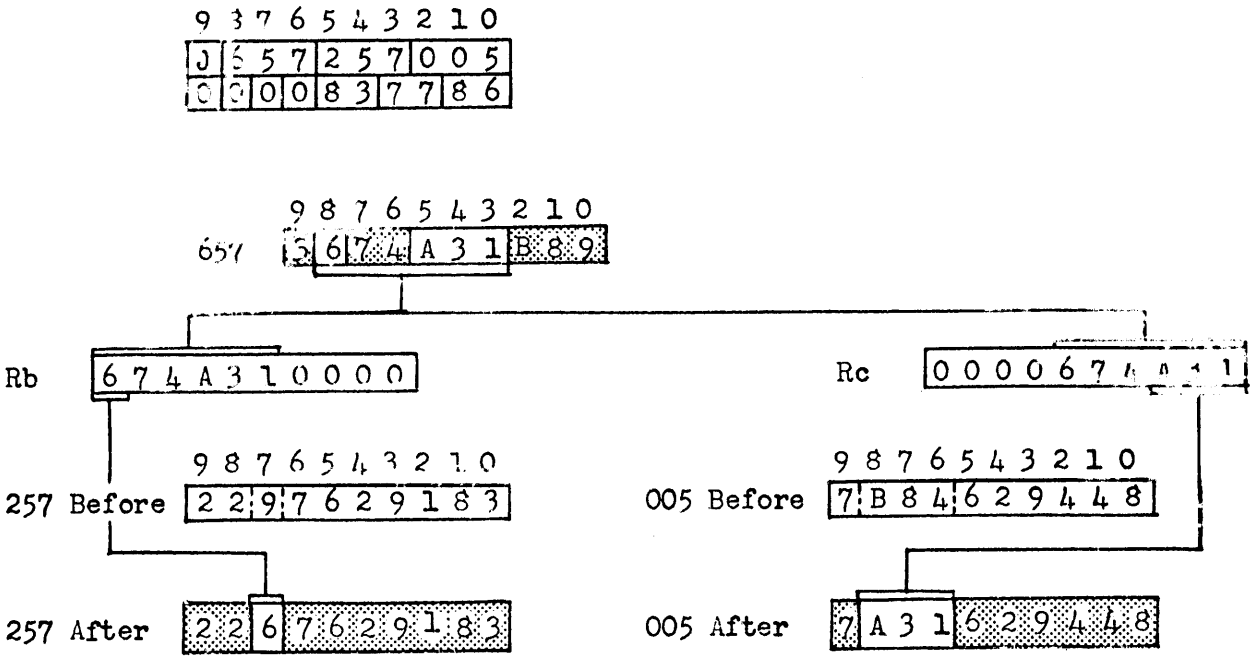

Example 2. Clear two Index Registers.



Example 3. Plant a Constant in the A and C fields of 004. Clear the B field of 004.

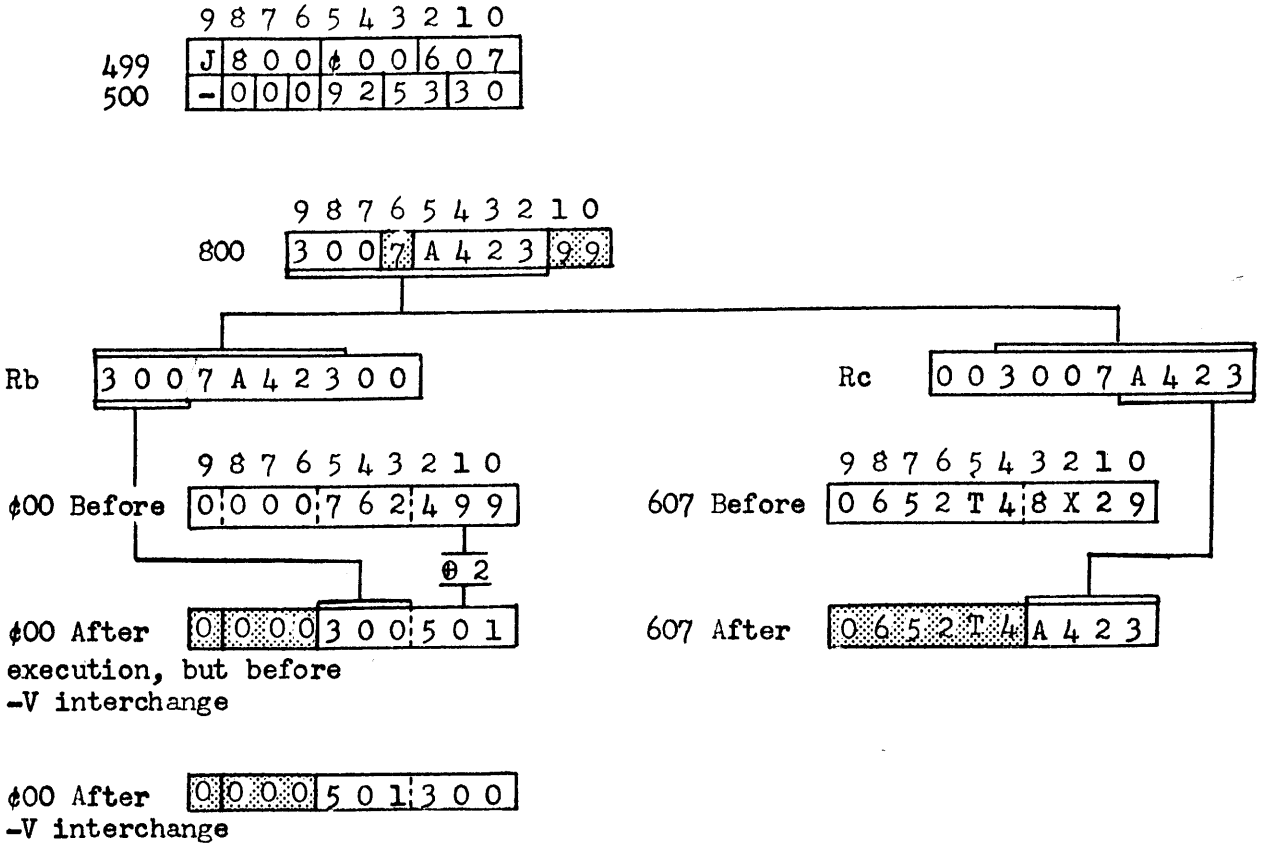



Example 4. Plant one Constant from 657:88 in 257:77 and another Constant from 657:53 in 005:86.



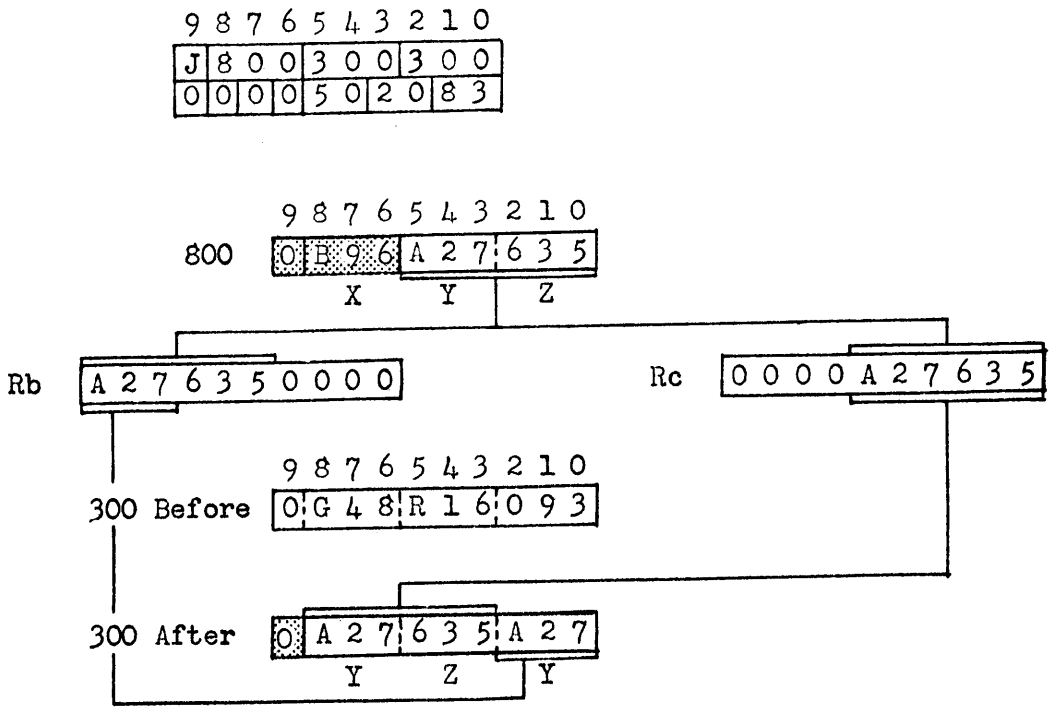
## "Normal" DISTRIBUTE

Example 5. Plant a Constant from 800:52 in 607:30.
Jump to Instruction stored in Cell 300, and set return link.

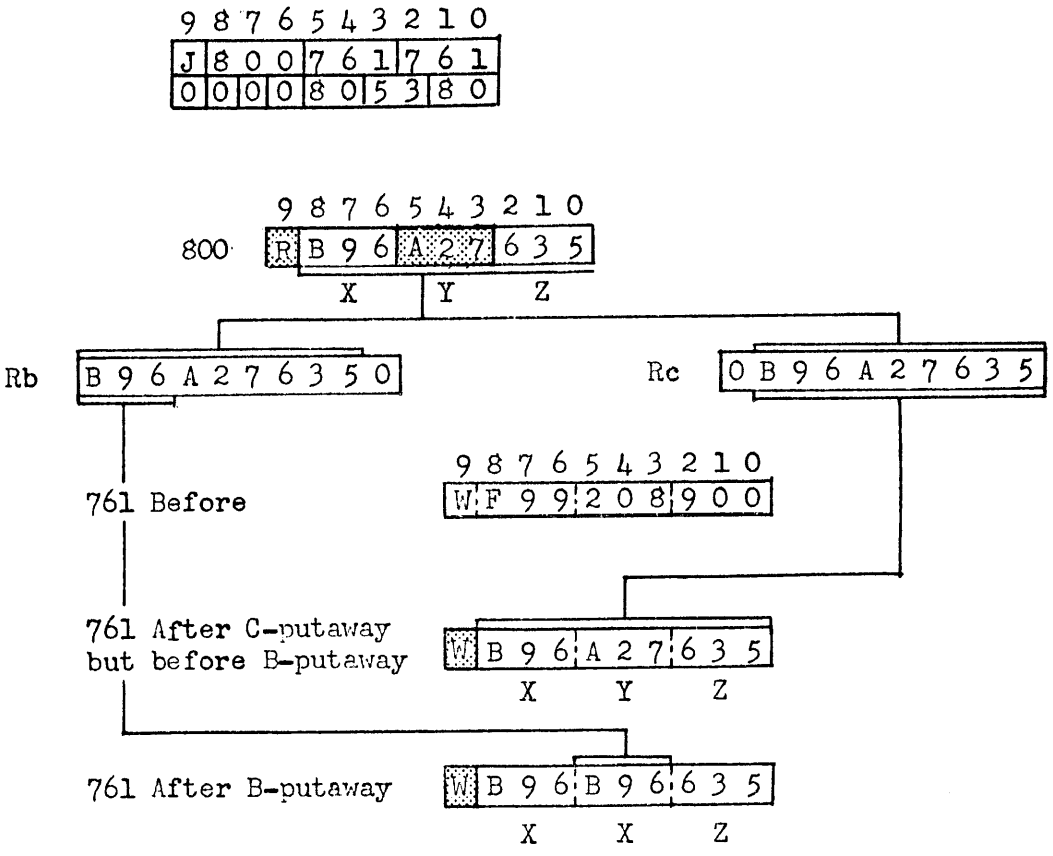


## "Normal" DISTRIBUTE

Example 6a. Three Constants (X, Y, Z) are stored in Cell 800.
Plant Y in the A and C fields, and Z in the B field, of Cell 300.

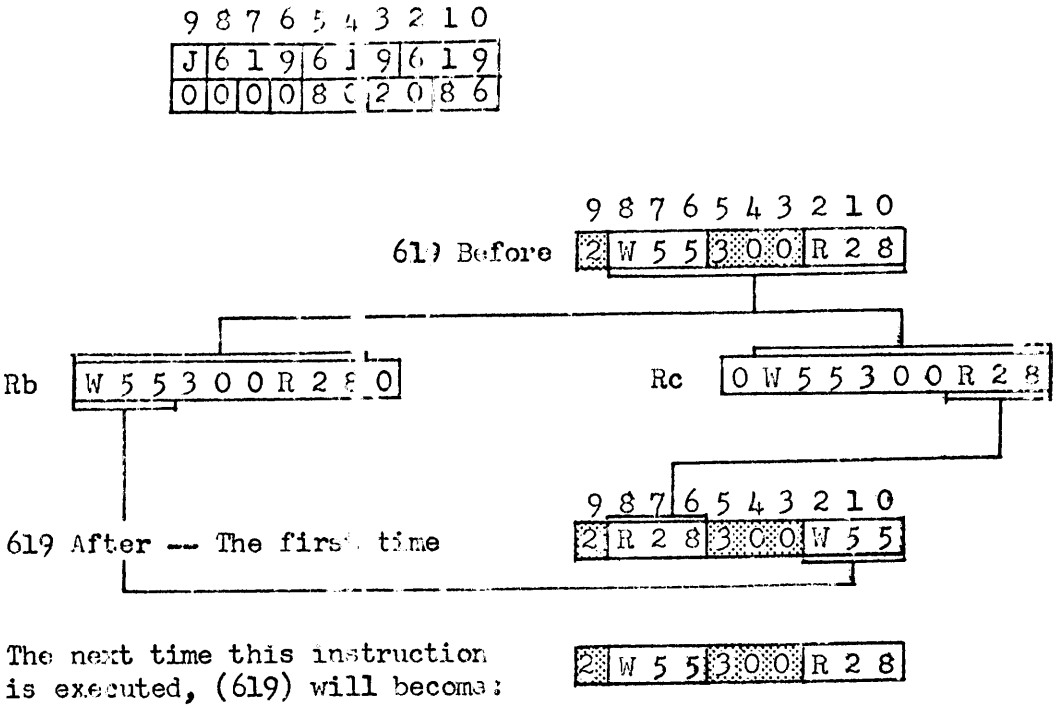



Example 6b. The same three Constants (X, Y, Z) are stored in Cell 800.
Plant X in the A and B fields, and Z in the C field, of Cell 761.



## "Normal" DISTRIBUTE

Example 7. Two-way alternator.
Interchange the A and C fields of Cell 619.



## "Normal" DISTRIBUTE

Example 8a. Three-way alternator. Cycle the A,←B,←C fields of Cell 574.

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| J | 5 | 7 | 4 | 5 | 7 | 4 | 5 | 7 | 4 |
| 0 | 0 | 0 | 0 | 8 | 0 | 2 | 0 | 8 | 3 |

574 Before

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 0 | 1 | 2 | 1 | 3 | 2 | 2 | 1 | 4 |

Rb

| 0 | 1 | 2 | 1 | 3 | 2 | 2 | 1 | 4 | 0 |
|---|---|---|---|---|---|---|---|---|---|

Rc

| 0 | 0 | 1 | 2 | 1 | 3 | 2 | 2 | 1 | 4 |
|---|---|---|---|---|---|---|---|---|---|

574 After — The first time

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 1 | 3 | 2 | 2 | 1 | 4 | 0 | 1 | 2 |

The second time this instruction is executed, (574) will become:

| 8 | 2 | 1 | 4 | 0 | 1 | 2 | 1 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|

The third time this instruction is executed, (574) will become:

| 8 | 0 | 1 | 2 | 1 | 3 | 2 | 2 | 1 | 4 |
|---|---|---|---|---|---|---|---|---|---|

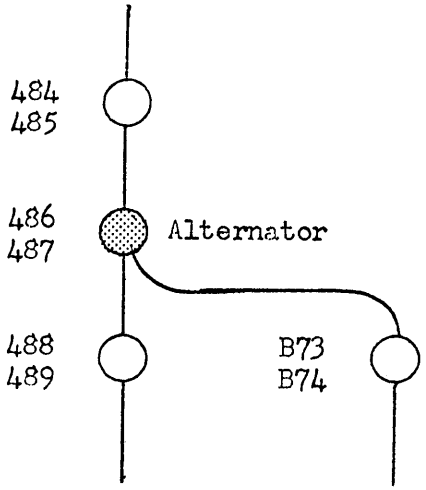
-- and so on --

Example 8b. The following Instruction will cause the three fields to cycle in the opposite direction:

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| J | 5 | 7 | 4 | 5 | 7 | 4 | 5 | 7 | 4 |
| 0 | 0 | 0 | 0 | 8 | 0 | 5 | 0 | 8 | 6 |

Example 9. Two-way branching alternator.

Write an Instruction in Cells 486-487 so that: Every other time it is executed, the program will select its next Instruction alternately from Cell 488 and Cell B73.




| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 486 | J | 9 | 6 | 2 | 4 | 8 | 7 | ¢ | 0 | 0 |
| 487 | 0 | 0 | 0 | 0 | 9 | 0 | 5 | 4 | 2 | 0 |

| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 962 | 7 | 3 | 9 | 0 | B | 7 | 3 | 4 | 8 | 8 |

Initially, the Instruction picks up (962:90).

The first time it is executed, it plants 488 in ¢00:20, and changes itself so that the next time it is executed, it will pick up (962:73).

The second time it is executed, it plants B73 in ¢00:20, and changes itself so that the next time it is executed, it will pick up (962:90) again.

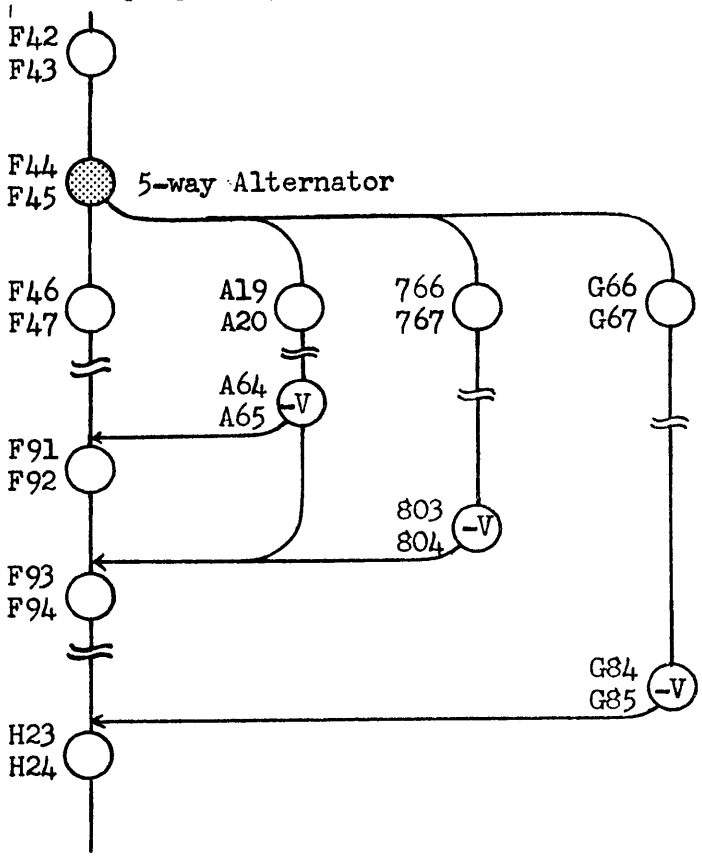
Example 10. Five-way branching alternator. Set return link.

Write an Instruction in Cells F44-F45 to set up a 5-way cycle. After executing this Instruction, the program will select its next Instruction as follows:

1st time: from Cell F46
2nd time: from Cell A19 and link to Cell F91
3rd time: from Cell A19 and link to Cell F93
4th time: from Cell 766 and link to Cell F93
5th time: from Cell G66 and link to Cell H23

6th time: from Cell F46 again
\- - - and so on - - -

The last Instruction in each branch will return to the proper point in the main program by using "minus V".




| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| F44 | J | H | 9 | 3 | F | 4 | 4 | ¢ | 0 | 0 |
| F45 | 0 | 0 | 0 | 0 | 8 | 0 | 8 | 6 | 5 | 0 |

| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| H93 | | H | 9 | 4 | | | | F | 4 | 6 |
| H94 | | H | 9 | 5 | F | 9 | 1 | A | 1 | 9 |
| H95 | | H | 9 | 6 | F | 9 | 3 | A | 1 | 9 |
| H96 | | H | 9 | 7 | F | 9 | 3 | 7 | 6 | 6 |
| H97 | | H | 9 | 3 | H | 2 | 3 | G | 6 | 6 |

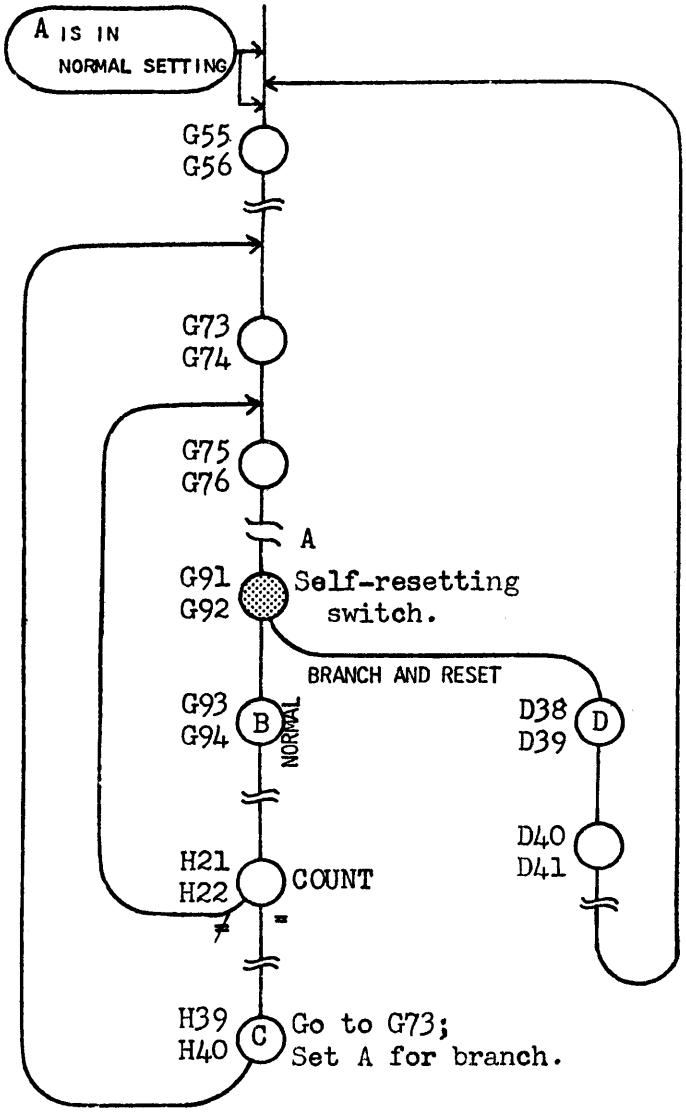
Each time the Instruction is executed, it plants a pair of addresses in ¢00:53 and ¢00:20; and changes itself so that the next time it is executed, it will pick up the next line of the table.

## "Normal" DISTRIBUTE

Example 11. Self-resetting Program Switch.

Instruction A will normally cause the program to pass to Instruction B. However, after Instruction C has been executed, Instruction A will:

1) Cause the program to branch to Instruction D;
2) At the same time reset itself to the normal condition.




| | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (A) | G91 | J | 2 | 8 | 7 | ¢ | 0 | 0 | 2 | 8 | 7 |
| | G92 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 5 | 3 |
| (C) | H39 | J | 2 | 8 | 6 | 2 | 8 | 7 | ¢ | 0 | 0 |
| | H40 | 0 | 0 | 0 | 0 | 5 | 0 | 5 | 3 | 2 | 0 |

| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 286 | | | | | | D | 3 | 8 | G | 7 | 3 |
| 287 | | | | | | G | 9 | 3 | G | 9 | 3 |

After Instruction C has been executed, 287:53 contains D38. The next time Instruction A is executed, it plants D38 in ¢00:20, and resets (287:53) back to G93.

Alternate Method

| | | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (A) | G91 | J | 5 | 7 | 0 | ¢ | 0 | 0 | G | 9 | 2 |
| | G92 | 0 | 0 | 0 | 0 | 9 | 0 | 2 | 0 | 5 | 5 |
| (C) | H39 | J | 5 | 7 | 1 | G | 9 | 2 | ¢ | 0 | 0 |
| | H40 | 0 | 0 | 0 | 0 | 8 | 5 | 5 | 5 | 2 | 0 |

| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 570 | G | 9 | 3 | D | 3 | 8 | | | | 9 |
| 571 | | 6 | G | 7 | 3 | | | | | |

After Instruction C has been executed, Instruction A has been set to pick up (570:60). The next time Instruction A is executed, it will plant D38 in ¢00:20 and will reset itself to pick up (570:90).

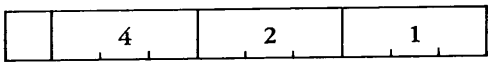
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

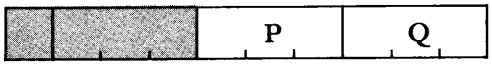
S-values of syllables

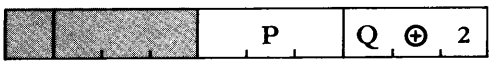
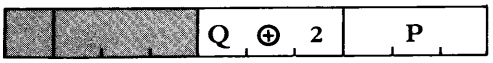



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

THREE INSTRUCTIONS, DISTRIBUTE, SUPPRESS, EDIT, ARE GENERALLY SIMILAR. BASICALLY, EACH OF THEM COPIES THE CONTENTS OF THE A-FIELD INTO THE B- AND C-FIELDS. THE C-PUTAWAY IS MADE FIRST, FROM RIGHT TO LEFT; THEN THE B-PUTAWAY IS MADE, FROM LEFT TO RIGHT. EACH INSTRUCTION HAS TWO VARIATIONS, AND EACH VARIATION CHANGES THE CONTENTS OF THE A-FIELD IN A DIFFERENT MANNER BEFORE MAKING THE PUTAWAYS.

## SUPPRESS

"Normal" variation: The contents of the A-field are zero-suppressed in the B- and C-putaways; the character "-" is treated in a special manner, as it is not suppressed (replaced by "space"), yet zero-suppression resumes to the right of a "-" character. If either putaway field is shorter than the A-field, characters will be lost in that put-away. If the C-field is longer than the A-field, "space" characters will appear at the left; if the B-field is longer than the A-field, zeros will be added at the right (unless the A-field contains all "zeros", in which case the B-field will be filled out to the right with "space" characters).

"Sign Split-Off" variation: The sign of the A-field (designated by the 5th bit in the left-most character) is split off, and appears in the putaways as a separate character; a positive sign appears as a "space" character, while a negative sign appears as a "-" character, immediately to the left of the character from which it was split off. The zone-bits of all the original characters from the A-field are replaced by 0-bits, leaving only the numeric bits, and the putaways are zero-suppressed.

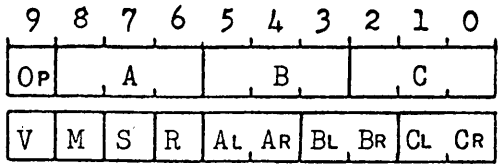
The putaways, therefore, are each one character longer than the A-field. If the C-field is longer than this, "space" characters will be added to the left of the sign character; if it is the same length as the A-field, the sign character is lost (once split off, the sign is never re-combined with the left-most character); and if the C-field is shorter than the A-field, other characters will be lost from the left end of the field. If the B-field is longer than the putaway, zeros will be added at the right (unless the A-field contains all "zeros", in which case the B-field will be filled out to the right with "space" characters); if it is too short for the full putaway, characters will be lost from the right end of the field.

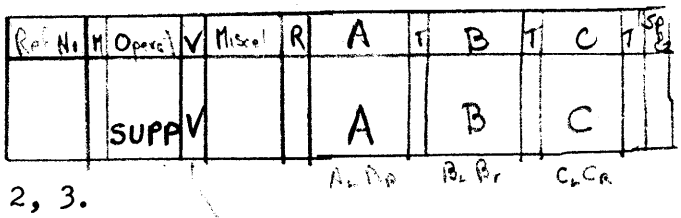
OVERFLOW ALARM: Will not be set by either variation.

INSTRUCTION FORMAT:

Operation: **SUPPRESS** (SUPP)

Operation Code: **K**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of word containing the field whose contents are to be distributed.

AL, AR: locations of left-most and right-most characters, respectively, of the A-field.

B: base of address of location into which the distributed field is to be placed, left-justified.

BL, BR: locations of left-most and right-most characters, respectively, of the B-putaway field.

C: base of address of location into which the distributed field is to be placed, right-justified.

CL, CR: locations of left-most and right-most characters, respectively, of the C-putaway field.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Normal | SUPP |
| 1 | Sign Split-Off | SUPP:S |

DESCRIPTION OF: SUPPRESS

V = 0: [A] is transferred to a 10-character-long working register (Rc), right-justified, and also to another 10-character-long working register (Rb), left-justified. In the course of these transfers, zone-bits are preserved, and the information is unchanged.

The contents of registers Rc and Rb are then zero-suppressed; that is, starting at the left end of each register, all "zero" characters are replaced by "space" characters (code 00 1100), until a non-zero character is encountered, or until all the characters in the register have been replaced by spaces. For the purpose of this variation, the character "-" is considered neither a zero, nor a non-zero character; it is not replaced by "space", nor does it terminate zero-suppression.

The contents of Rc replace [C], right-justified; then the contents of Rb replace [B], left-justified.

V = 1: This is the "Sign Split-Off" variation.

[A] is transferred to an 11-character-long working register (Rc), right-justified, and also to another 11-character-long working register (Rb), left-justified at character-position 9. In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

The contents of registers Rc and Rb are then zero-suppressed; that is, starting at the left end of each register, all "zero" digits are replaced by "space" characters (code 00 1100), until a non-zero digit is encountered, or until all the characters in the register have been replaced by spaces.

If the sign-bit of the character in position A1 is a 1-bit (minus sign), then a "minus" character (code 01 0000) is inserted into Rb at character-position 10, and into Rc at the character-position immediately to the left of the character transferred from position A1.

If the sign-bit of the character in position A1 is a 0-bit (plus sign), note that the proper character-positions of registers Rc and Rb already contain "space" characters.

The contents of Rc replace [C], right-justified; then the contents of Rb replace [B], left-justified.

OVERFLOW ALARM: Will not be set by either variation.

# DISTINCTIONS

## DISTRIBUTE, SUPPRESS, EDIT

| [A] | Instruction | V | B-Putaway → | ← C-Putaway |
|---|---|---|---|---|
| 00465 | DISTRIBUTE | 0 | 0,0,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | DISTRIBUTE | 1 | SP,0,0,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,0,0,4,6,5 |
| | SUPPRESS | 0 | SP,SP,4,6,5,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | SUPPRESS | 1 | SP,SP,SP,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | EDIT | 0 | SP,SP,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | EDIT | 1 | *,*,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| -00009 | DISTRIBUTE | 0 | -,0,0,0,0,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | DISTRIBUTE | 1 | -,0,0,0,0,0,9,0,0,0 | SP,SP,SP,-,0,0,0,0,0,9 |
| | SUPPRESS | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | SP,SP,SP,SP,-,SP,SP,SP,SP,9 |
| | SUPPRESS | 1 | -,SP,SP,SP,SP,SP,9,0,0,0 | SP,SP,SP,-,SP,SP,SP,SP,SP,9 |
| | EDIT | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | EDIT | 1 | *,*,*,*,*,9,0,0,0,0 | 0,0,0,0,0,0,0,0,0,9 |

EDIT → Sets Overflow Alarm

| A | Instruction | V | B-Putaway → | C-Putaway ← |
|---|---|---|---|---|
| G26A9 | DISTRIBUTE | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | DISTRIBUTE | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | SUPPRESS | 0 | G,2,6,A,9,0,0,0,0,0 | SP,SP,SP,SP,SP,G,2,6,A,9 |
| | SUPPRESS | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | EDIT | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | EDIT | 1 | 7,2,6,1,9,0,0,0,0,0 | 0,0,0,0,0,7,2,6,1,9 |
| | | | → Sets Overflow Alarm | |
| 0 | DISTRIBUTE | 0 | 0,0,0,0,0,0,0,0,0,0 | 0,0,0,0,0,0,0,0,0,0 |
| | DISTRIBUTE | 1 | SP,0,0,0,0,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,0 |
| | SUPPRESS | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | SUPPRESS | 1 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | EDIT | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | 0,0,0,0,0,0,0,0,0,0 |
| | EDIT | 1 | *,*,*,*,*,*,*,*,*,* | 0,0,0,0,0,0,0,0,0,0 |

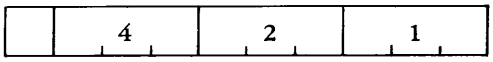
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | , |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

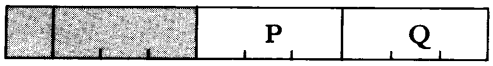
S-values of syllables

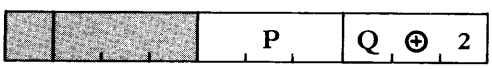
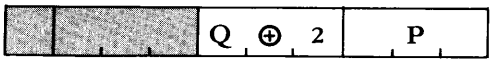



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

THREE INSTRUCTIONS, DISTRIBUTE, SUPPRESS, EDIT, ARE GENERALLY SIMILAR. BASICALLY, EACH OF THEM COPIES THE CONTENTS OF THE A-FIELD INTO THE B- AND C-FIELDS. THE C-PUTAWAY IS MADE FIRST, FROM RIGHT TO LEFT; THEN THE B-PUTAWAY IS MADE, FROM LEFT TO RIGHT. EACH INSTRUCTION HAS TWO VARIATIONS, AND EACH VARIATION CHANGES THE CONTENTS OF THE A-FIELD IN A DIFFERENT MANNER BEFORE MAKING THE PUTAWAYS.

## E D I T

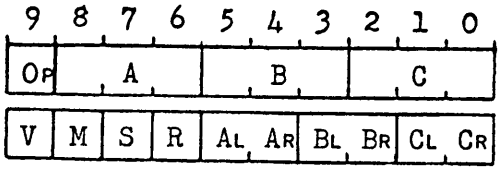
"Normal" variation: The contents of the A-field are unchanged in the C-putaway, and are zero-suppressed in the B-putaway. In zero-suppression, the character "-" is treated in a special manner, as it is not suppressed (replaced by "space"), yet zero-suppression resumes to the right of a "-" character. If either putaway field is shorter than the A-field, characters will be lost in that putaway; if either putaway field is longer than the A-field, zeros will be added to fill out that putaway field (unless the A-field contains all "zeros", in which case the B-field will be filled out to the right with "space" characters).

"Dollar-Protection" variation: The zone-bits of all the characters from the A-field are replaced by 0-bits, leaving only the numeric bits. The C-putaway is made in that form, while the B-putaway is "dollar-protected"; that is, the B-field is zero-suppressed in the usual manner, except that "*" characters (rather than "space" characters) replace suppressed zeros.

OVERFLOW ALARM: Will not be set by the "Normal" variation.

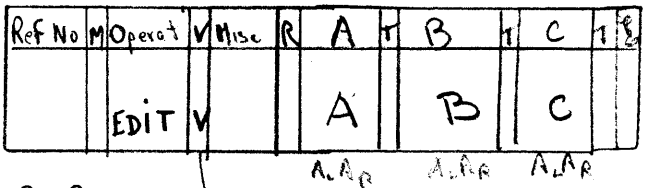
Will be set by the "Dollar-Protection" variation if any character in the A-field contains a 1-bit in any zone-bit position, including the sign-bit position of the field.

INSTRUCTION FORMAT:




**Operation: EDIT** (EDIT)

**Operation Code: L**




DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of word containing the field whose contents are to be distributed.

AL, AR: locations of left-most and right-most characters, respectively, of the A-field.

B: base of address of location into which the distributed field is to be placed, left-justified.

BL, BR: locations of left-most and right-most characters, respectively, of the B-putaway field.

C: base of address of location into which the distributed field is to be placed, right-justified.

CL, CR: locations of left-most and right-most characters, respectively, of the C-putaway field.

V: variation designator:

| V | Specifies | Abbreviation |
|---|---|---|
| 0 | Normal | EDIT |
| 1 | Dollar-Protection | EDIT:P |

DESCRIPTION OF: EDIT

V = 0: [A] is transferred to a 10-character-long working register (Rc), right-justified, and also to another 10-character-long working register (Rb), left-justified. In the course of these transfers, zone-bits are preserved, and the information is unchanged.

The contents of Rb are then zero-suppressed; that is, starting at the left end of the register, all "zero" characters are replaced by "space" characters (code 00 1100), until a non-zero character is encountered, or until all the characters in the register have been replaced by "spaces". For the purpose of this variation, the character "-" is considered neither a zero, nor a non-zero character; it is not replaced by "space", nor does it terminate zero-suppression. The contents of Rc are not changed.

(Rc) replaces [C], right-justified; then (Rb) replaces [B], left-justified.

V = 1: This is the "Dollar-Protection" variation.

[A] is transferred to a 10-character-long working register (Rc), right-justified, and also to another 10-character-long working register (Rb), left-justified.

In the course of both these transfers, the zone-bits of all characters transferred are replaced by 0-bits.

The contents of Rb are then zero-suppressed and "dollar-protected"; that is, starting at the left end of Rb, all "zero" digits are replaced by "*" characters (code 11 0000), until a non-zero digit is encountered, or until all the characters in the register have been replaced by asterisks. The contents of Rc are not changed.

(Rc) replaces [C], right-justified; then (Rb) replaces [B], left-justified.

OVERFLOW ALARM: Will not be set by the "Normal" variation (V = 0).

Will be set by the "Dollar-Protection" variation (V = 1) if any character in [A] contains a 1-bit in any zone-bit position.

# DISTINCTIONS

## DISTRIBUTE, SUPPRESS, EDIT

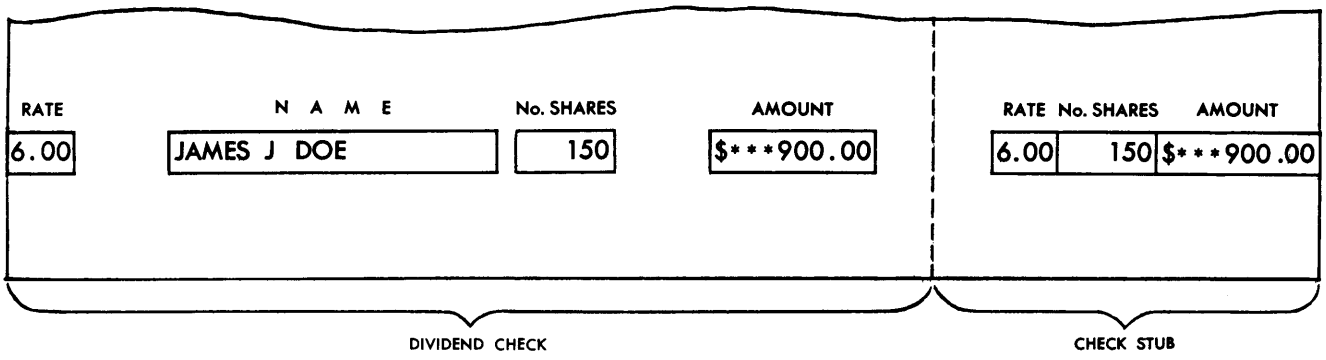
| [A] | Instruction | V | B-Putaway → | ← C-Putaway |
|---|---|---|---|---|
| 00465 | DISTRIBUTE | 0 | 0,0,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | DISTRIBUTE | 1 | SP,0,0,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,0,0,4,6,5 |
| | SUPPRESS | 0 | SP,SP,4,6,5,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | SUPPRESS | 1 | SP,SP,SP,4,6,5,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,4,6,5 |
| | EDIT | 0 | SP,SP,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| | EDIT | 1 | *,*,4,6,5,0,0,0,0,0 | 0,0,0,0,0,0,0,4,6,5 |
| -00009 | DISTRIBUTE | 0 | -,0,0,0,0,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | DISTRIBUTE | 1 | -,0,0,0,0,0,9,0,0,0 | SP,SP,SP,-,0,0,0,0,0,9 |
| | SUPPRESS | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | SP,SP,SP,SP,-,SP,SP,SP,SP,9 |
| | SUPPRESS | 1 | -,SP,SP,SP,SP,SP,9,0,0,0 | SP,SP,SP,-,SP,SP,SP,SP,SP,9 |
| | EDIT | 0 | -,SP,SP,SP,SP,9,0,0,0,0 | 0,0,0,0,-,0,0,0,0,9 |
| | EDIT | 1 | *,*,*,*,*,9,0,0,0,0 | 0,0,0,0,0,0,0,0,0,9 |

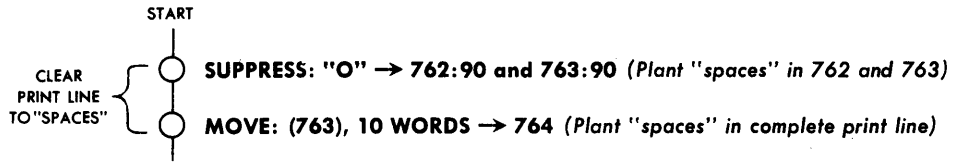
EDIT → Sets Overflow Alarm

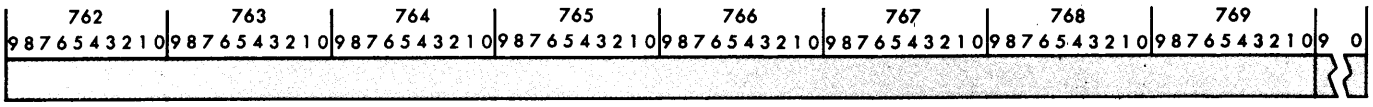
| A | Instruction | V | B-Putaway → | ← C-Putaway |
|---|---|---|---|---|
| G26A9 | DISTRIBUTE | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | DISTRIBUTE | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | SUPPRESS | 0 | G,2,6,A,9,0,0,0,0,0 | SP,SP,SP,SP,SP,G,2,6,A,9 |
| | SUPPRESS | 1 | -,7,2,6,1,9,0,0,0,0 | SP,SP,SP,SP,-,7,2,6,1,9 |
| | EDIT | 0 | G,2,6,A,9,0,0,0,0,0 | 0,0,0,0,0,G,2,6,A,9 |
| | EDIT | 1 | 7,2,6,1,9,0,0,0,0,0 | 0,0,0,0,0,7,2,6,1,9 |
| | | | → Sets Overflow Alarm | |
| 0 | DISTRIBUTE | 0 | 0,0,0,0,0,0,0,0,0,0 | 0,0,0,0,0,0,0,0,0,0 |
| | DISTRIBUTE | 1 | SP,0,0,0,0,0,0,0,0,0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,0 |
| | SUPPRESS | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | SUPPRESS | 1 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP |
| | EDIT | 0 | SP,SP,SP,SP,SP,SP,SP,SP,SP,SP | 0,0,0,0,0,0,0,0,0,0 |
| | EDIT | 1 | *,*,*,*,*,*,*,*,*,* | 0,0,0,0,0,0,0,0,0,0 |

# AN EXAMPLE OF EDITING

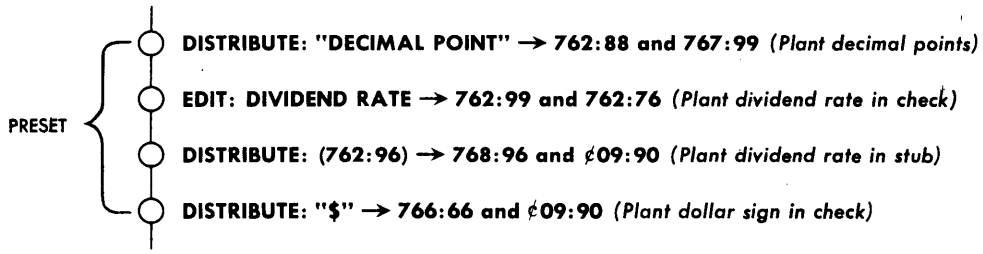
**Required: To edit a one-line Dividend Check and Stub**
**Using COMBINE; DISTRIBUTE (V=O); SUPPRESS (V=O); EDIT (V=O, V=I); MOVE.**

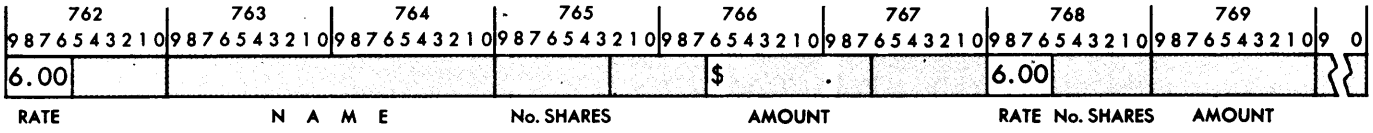


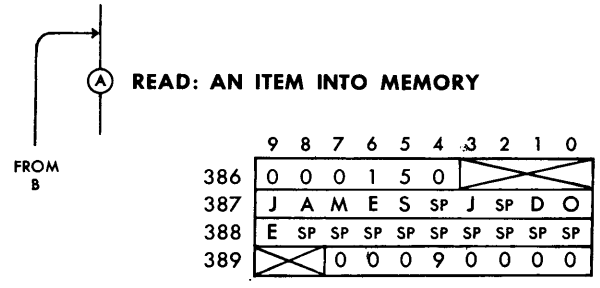



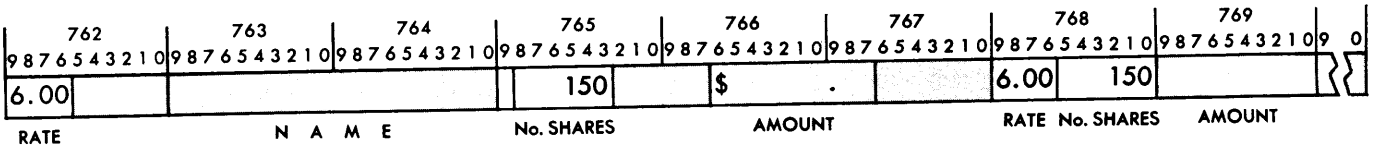

120-CHARACTER PRINT-LINE IMAGE
SHADED AREAS REPRESENT "SPACE" CHARACTERS PLANTED BY THE PREVIOUS OPERATIONS



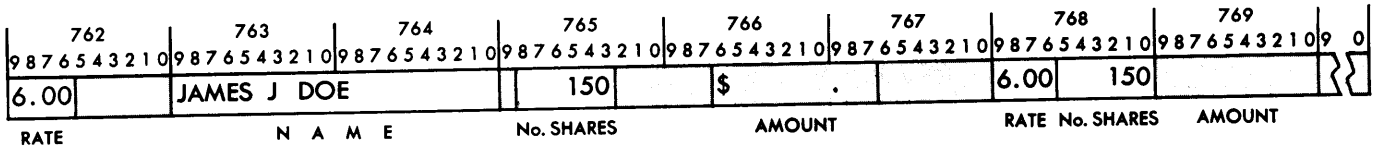



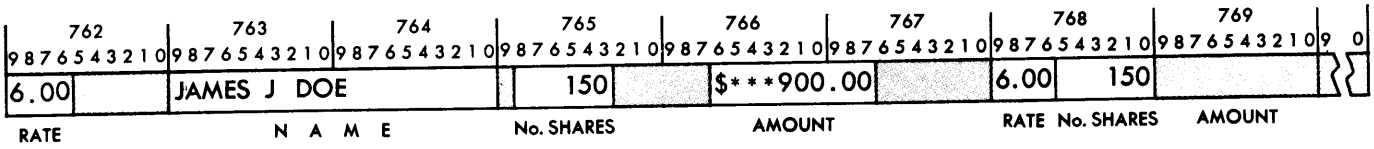
**SUPPRESS: (386:94) → 765:83 and 768:50** *(Number of shares in check and stub)*



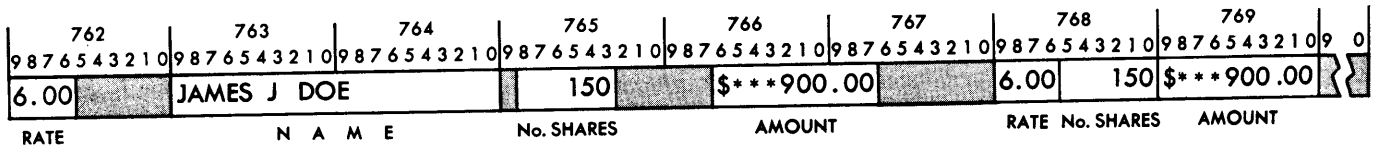

**MOVE: (387), 2 WORDS → 763** *(Name)*




**EDIT (Dollar Protection): (389:70) → 766:50 and 767:87** *(Amount on check)*




**COMBINE: (766:60) and (767:97) → 769:90** *(Amount on stub)*




**OUTPUT THE EDITED LINE**

**GO TO A** (B)

SHADED AREAS REPRESENT "SPACE" CHARACTERS PLANTED BY THE INITIAL CLEARING OPERATIONS. BLANK AREAS REPRESENT "SPACE" CHARACTERS PLANTED BY THE EDITING OPERATIONS.

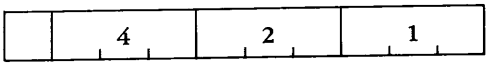
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A and AF syllables modified |
| 2 | B and BF syllables modified |
| 1 | C and CF syllables modified |
| Sum | Determines combination of syllables modified. |

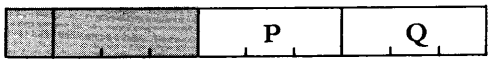
S-values of syllables

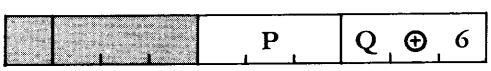
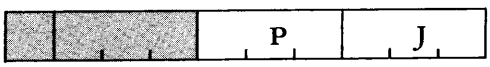
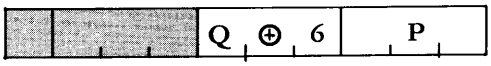
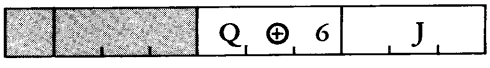



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | Runout MERGE | Cutoff MERGE |
| positive |   |   |
| negative |   |   |

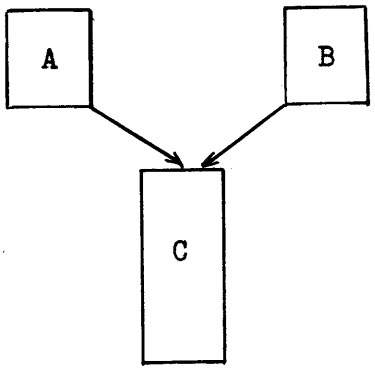
J: May be JA, JB, JC

# M E R G E

MERGE assumes the designation of three memory areas (called "pockets"), referred to, respectively, as the A, B, C pockets. Except as stated below, there is no necessary relationship among the sizes of the three pockets.

The A and B pockets each contain a "string" of items: that is, a collection of items which has previously been sorted according to some key.

MERGE combines the A and B strings into a single new string, which appears in the C pocket.



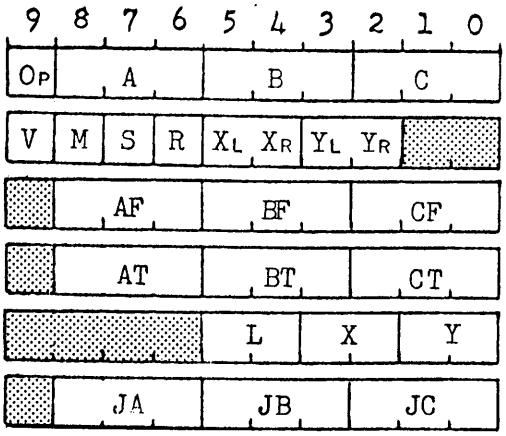

Runout MERGE proceeds until either the A pocket or the B pocket has been exhausted by copying the last item into the C pocket. Then all items remaining in the other pocket are "run out" by being copied into the C pocket. This type of MERGE assumes, therefore, that the C pocket is equal in length to the combined lengths of the A and B pockets.

Cutoff MERGE proceeds until:

A pocket exhausted
or B pocket exhausted
or C pocket filled

and then terminates. The next instruction to be executed is selected from the appropriate one of three locations, according to the reason for termination, and the MERGE instruction, as recorded in the Memory, indicates the point at which it terminated.

INSTRUCTION FORMAT:




Operation: "Cutoff" MERGE (MRGE:C)

Operation Code: M (V = 0, 1)

Execution Time: (in micro-minutes)

22 FIXED
+ 1 IF INDEX REGISTER USED
+ 10 PER COMPARISON OF AN A-ITEM WITH A B-ITEM (12 IF 2-WORD KEY)
+ 2 PER WORD MOVED INTO C-POCKET
+ 1 IF CF TERMINATION

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register. Note. 1st ... words are modified by index register OOR

A, B, C: bases of addresses of first word, first item, A, B, C pockets.

AF, BF, CF: bases of addresses of first word, last item, A, B, C pockets.

AT, BT, CT: word-counts, by which A, B, C are augmented before operation begins. During "Cutoff" MERGE, AT, BT, CT are automatically augmented, and indicate the point at which the MERGE terminated.

L: item length (number of words). $0 < L \leq 99$.

X, Y: relative addresses, with respect to the first word of each item, of words containing the major and minor control keys, respectively.

$X_L$, $X_R$: locations of left-most and right-most characters, respectively, of
$Y_L$, $Y_R$: major and minor control keys.

NOTE: if a single control key is used, it is specified by Y, $Y_L$, $Y_R$. X, $X_L$, $X_R$ are then irrelevant.

JA, JB, JC: address of next instruction if A, B, C termination condition is encountered in "Cutoff" MERGE.

V: variation designator:

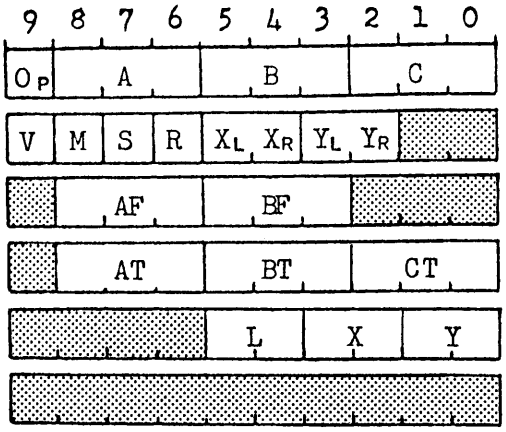
Neat

| No. | M | Operation | V | Misc. | R | A | t | B | t | C | t | ½ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MRGE | C | L | | A | | B | | C | | |
| | | 1 | | K | | $N_A$ | | $N_B$ | | $N_C$ | | |
| | | 2 | | | | | | $X_1X_LX_R$ | | $Y_1Y_LY_R$ | | |
| | | 3 | | | | $J_A$ | | $J_B$ | | $J_C$ | | |

| Cutoff MERGE | |
|---|---|
| 0 | one control key |
| 1 | two control keys |

A, B, C : Designate First Word, First Item, each string
$N_A$, $N_B$, $N_C$ Number of item in each string
L : Length of each item
K : Number of Keys (1 or 2) for merge
$X_1X_LX_R$ : Relative Position of Major Key within item (Minor Key)
$Y_1Y_LY_R$
$J_A$ $J_B$ $J_C$ : Address of next instruction

INSTRUCTION FORMAT:




Operation: "Runout" MERGE (MRGE:R)

Operation Code: M (V = 2, 3)

Execution Time: (in micro-minutes)

19 FIXED
+ 1 IF INDEX REGISTER USED
+ 1 IF -V
+ 10 PER COMPARISON OF AN A-ITEM WITH A B-ITEM (12 IF 2-WORD KEY)
+ 5 PER ITEM RUNOUT INTO C-POCKET
+ 2 PER WORD MOVED INTO C-POCKET (INCLUDING ITEMS RUNOUT)

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Note: 1st + last words are modified by index register OOR

A, B, C: bases of addresses of first word, first item, A, B, C pockets.

AF, BF : bases of addresses of first word, last item, A, B pockets.

AT, BT, CT: word-counts, by which A, B, C are augmented before operation begins. During "Runout" MERGE, AT, BT, CT remain unchanged.

L: item length (number of words). $0 < L \leq 99$.

X, Y: relative addresses, with respect to the first word of each item, of words containing the major and minor control keys, respectively.

$X_L$, $X_R$: locations of left-most and right-most characters, respectively, of
$Y_L$, $Y_R$: major and minor control keys.

NOTE: if a single control key is used, it is specified by Y, $Y_L$, $Y_R$. X, $X_L$, $X_R$ are then irrelevant.

In "Runout" MERGE, the next instruction in sequence is always executed.

V: variation designator:

| | Runout MERGE |
|---|---|
| one control key | 2 |
| two control keys | 3 |

Next

| Ref No | M | Operat | V | Misc | R | A | I | B | I | C | I | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MRGE | R | L | | A | | B | | C | | |
| | | 1 | | K | | $N_A$ | | $N_B$ | | | | |
| | | 2 | | | | | | $X \cdot X_L X_R$ | | $Y \cdot Y_L Y_R$ | | |

## DESCRIPTION OF: MERGE

NOTES:

1) The comparison made by MERGE is the same as that made by COMPARE ALPHANUMERIC. Therefore, if the keys are numeric, all negative numbers (in sequence by increasing magnitude) will sort behind all positive keys. MERGE may be made to yield any desired sorting sequence, however, by using ADD BINARY and COMPLEMENT BINARY to transform the sorting keys.

2) If the keys are positive numbers, they may be PACKED, and the operation of MERGE will be unchanged. Alphabetic characters may be combined in a single key with PACKED digits, by using INSERT. Therefore, MERGE will operate over a 20-character alphabetic field, over a 30-digit PACKED numeric field, or over a field containing any proportional combination of alphabetic characters and PACKED digits.

3) "Assembly of Key" mentioned in the Description, takes place as follows:

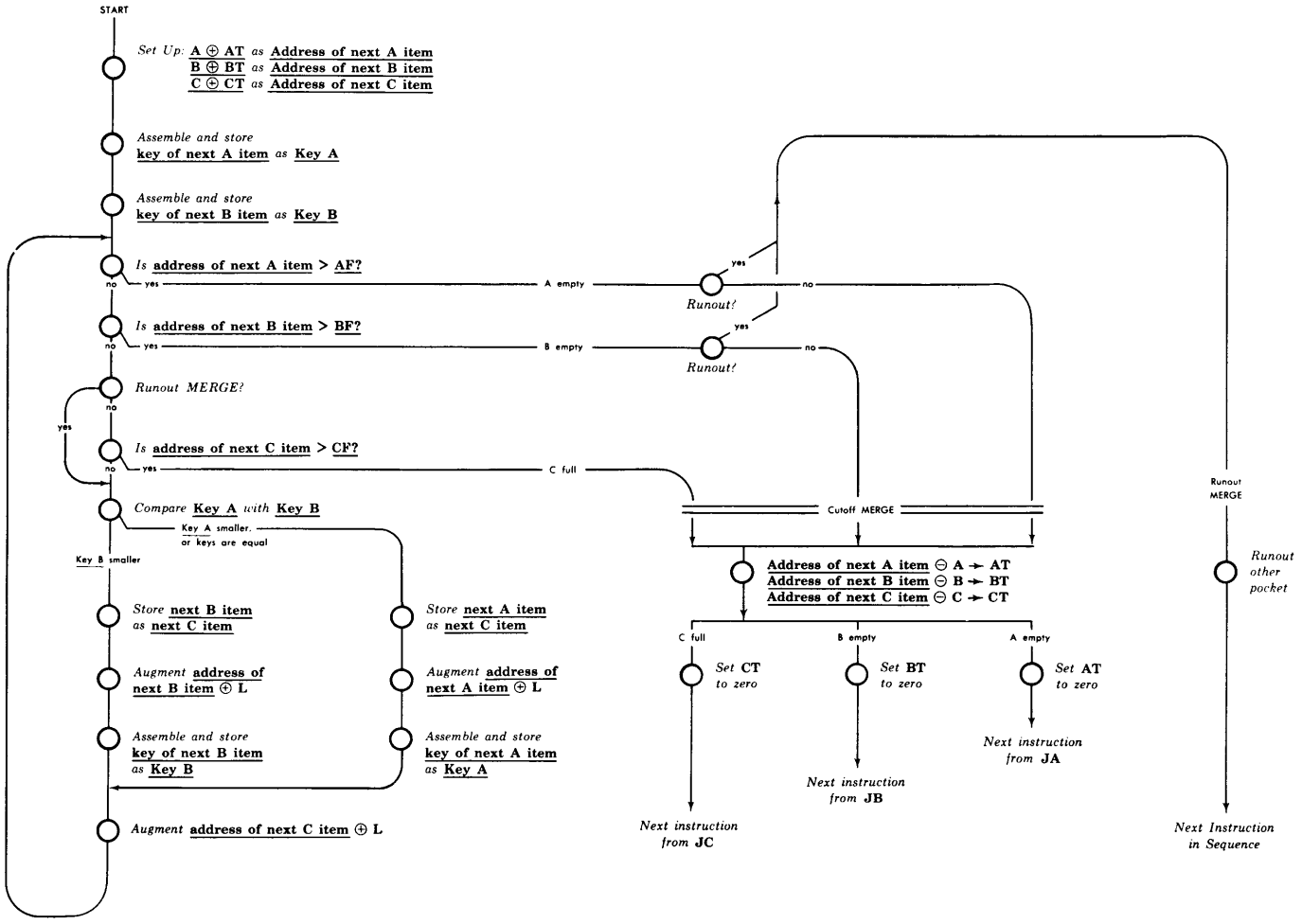
The minor control key of the next A item is transferred to a 20-character-long working register (Ra), right-justified.

If two control keys have been specified, the major control key of that item is next transferred to Ra, right-justified against the left-most character of the minor control key.

The contents of Ra is then the stored Key of the next A item.

The key of the next B item is assembled and stored in the same fashion in Rb.

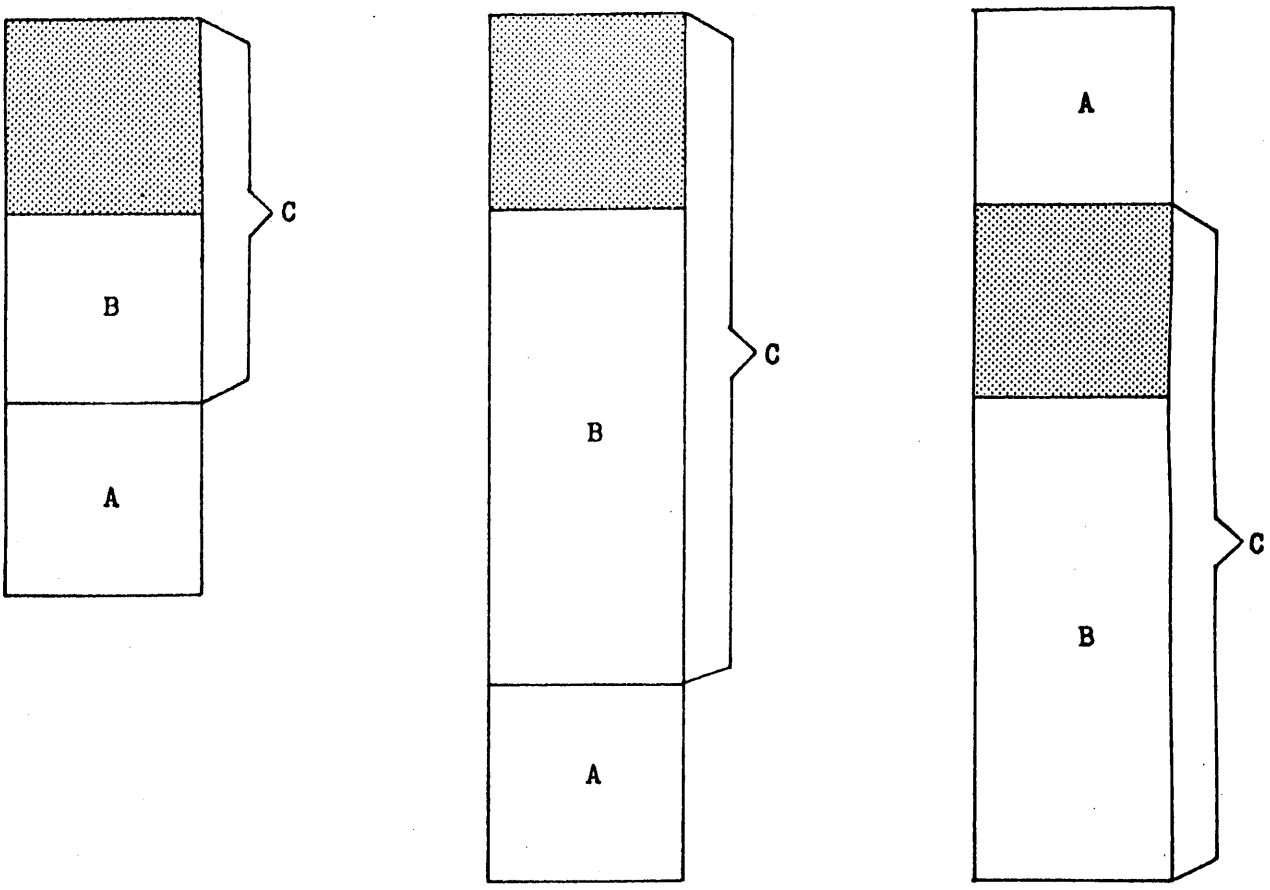
4) Note that both the first and third words of the MERGE Instruction are modified by the Index Register.




OVERFLOW ALARM: Will not be set by this Instruction.

## Runout MERGE

### Suggested patterns of memory space allocation for minimum space·requirements




If the number of items, or the item-length, is large, these Runout MERGES can be accomplished with a considerable saving in Processor time by using Cutoff MERGE:

- If the A-Exit is selected, the MERGE is complete, since runout would only copy the remaining items of B on top of themselves.
- If the B-exit is selected, reset the Instruction to Runout MERGE, set BT equal to BF ⊕ L ⊖ B (i.e. - B ⊕ BT will refer to the non-existent B-item which immediately follows BF), and repeat the MERGE Instruction to accomplish the runout of A.

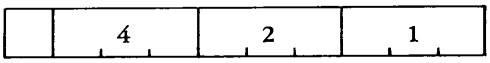
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

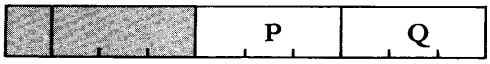
S-values of syllables

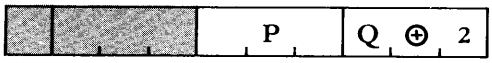
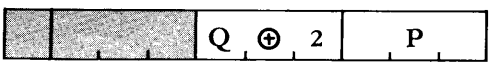



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation



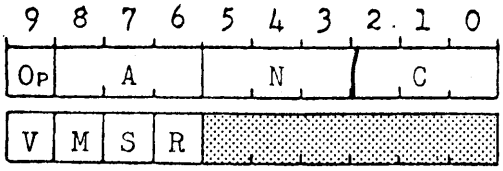

| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# M O V E

This Instruction moves any designated number of consecutive full words from one memory area to another.

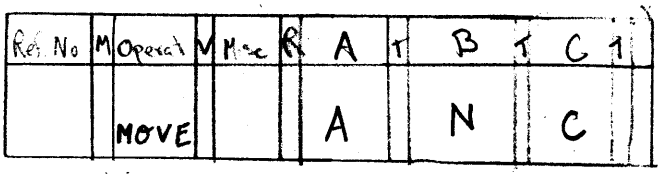
OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: MOVE (MOVE)

Operation Code: N




DEFINITIONS:

Op: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first word which is to be moved.

N: as modified by (OOR), is an address-type number which specifies how many consecutive words are to be moved.

$0 \leq N \leq Z99$

C: base of address in which the first word moved is to be stored.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: MOVE

This is not a partial-word operation.

The full word [A] is transferred to [C] . Then the full word [A ⊕ 1] is transferred to [C ⊕ 1]. The operation proceeds until the number of words specified by N has been moved.

Note that the first word moved is stored in its new location before the second word is looked up, &c. Therefore, if C designates the cell following the cell designated by A, then [A] will be copied into each of the N following cells.

If N specifies that zero words shall be moved, the Instruction moves zero words -- that is, it leaves the contents of the memory unchanged.

OVERFLOW ALARM: Will not be set by this Instruction.

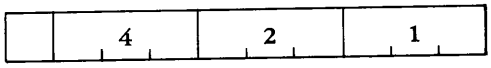
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

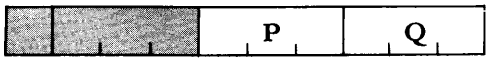
S-values of syllables

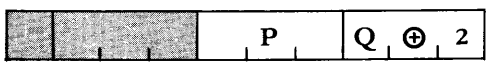
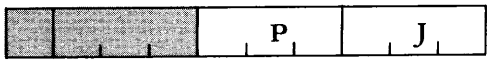
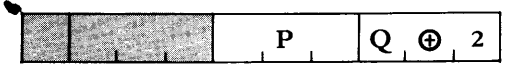
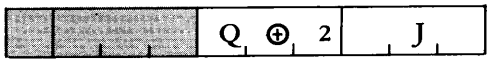



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# P A C K

This Instruction permits the programmer to take advantage of the fact that only four binary bits are required to specify any numeric digit. With numeric data, therefore, the storage space ordinarily occupied by the zero zone-bits of each digit is, in a sense, "wasted", particularly in magnetic tape files.

PACK takes the thirty 6-bit characters in three consecutive memory-cells, transforms them into 4-bit digits by discarding their zone-bits, and stores them, fifteen to a word, in two consecutive cells. The order in which the thirty characters appeared is not changed in this transformation. The Instruction performs this operation on as many consecutive 3-word groups as may be specified.

If any character with a non-zero zone-bit is encountered, the next Instruction to be executed will be selected from a "branch" address.

Since large portions of most files consist of numeric data, the numeric portion of the file may be packed for storage, saving one-third of the magnetic tape required to store that part of the file, with a proportionate saving in tape travel time during file processing.

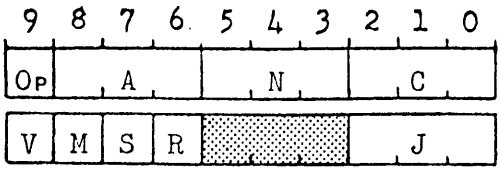
While arithmetic operations cannot be performed on the data in its packed 4-bit form, all binary and logical operations can be performed:

| | |
|---|---|
| Extract | Combine |
| Insert | Distribute ("Normal" variation) |
| Add Binary ("Normal" variation) | Merge |
| Complement Binary | Move |
| Test Bit | Sift |
| Compare Alpha-numeric | Magnetic Tape Copy and Search |
| Compare Equality | |

In particular, note that MERGE and SIFT can be performed over a 30-digit numeric key.

OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: PACK (PACK)

Operation Code: Ø

| Ref No | M | Operat | V | Misc | R | A | T | B | T | C | T | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PACK | | | | A | | N | | C | | |
| | | 1 | | | | | | | | J | | |

DEFINITIONS:

Op: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first word of first 3-word group to be packed.

N: as modified by (OOR), is an address-type number which specifies how many consecutive 3-word groups are to be packed. $0 \leq N \leq Z99$

C: base of address in which first word of first 2-word packed group is to be stored.

J: address of next Instruction, if any character in the data to be packed contains a non-zero zone-bit.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: PACK

This is not a partial-word operation.

[A] is transferred to a 30-character-long working register (Ra), left-justified. [A ⊕ 1] is transferred to Ra, left-justified against the right-most character from [A] . [A ⊕ 2] is transferred to Ra, left-justified against the right-most character from [A ⊕ 1], completing the 30-character capacity of Ra.

At this point, we regard the contents of Ra as a "stream" of 180 bits, which flows from the left end of Ra, through a "valve", toward the left end of a 20-character (120-bit) long working register (Rc).

As the first 2 bits flow from Ra, the valve is closed, and those bits are lost. Then the valve is opened to permit the passage of the next 4 bitis into Rc. Then the valve is closed again to discard the next 2 bits, opened for the next 4 bits, and so on, until Ra has been emptied.

The left-most 60-bits of Rc then replace [C], and the right-most 60-bits of Rc replace [C ⊕ 1]. Then [A ⊕ 3], [A ⊕ 4], [A ⊕ 5] are packed, and put away into [C ⊕ 2], [C ⊕ 3]. The process is repeated until N triplets have been packed. If N specifies zero groups to be packed, the Instruction performs no operation, and leaves the contents of the memory unchanged.

It will be seen that each pair of bits discarded in the packing process comprises the zone-bits of one of the original characters in Ra, and that the numeric bits of those characters are "packed tight" into Rc. The 20 characters into which each 3-word group is packed are "pseudo-characters" which have no meaning in themselves, but which arise out of the characteristics of the Processor, which in all other operations regards information as being made up of 6-bit characters.

If any of the discarded zone-bits was a 1-bit, the next Instruction to be executed will be selected from J. However, the full N triplets are always packed.

OVERFLOW ALARM: Will not be set by this Instruction.

## ILLUSTRATION OF PACK

In the following diagram, the 30 characters of the 3-word group to be packed are numbered from 1 to 30, in order to illustrate the arrangement of information before and after packing.

BEFORE PACKING

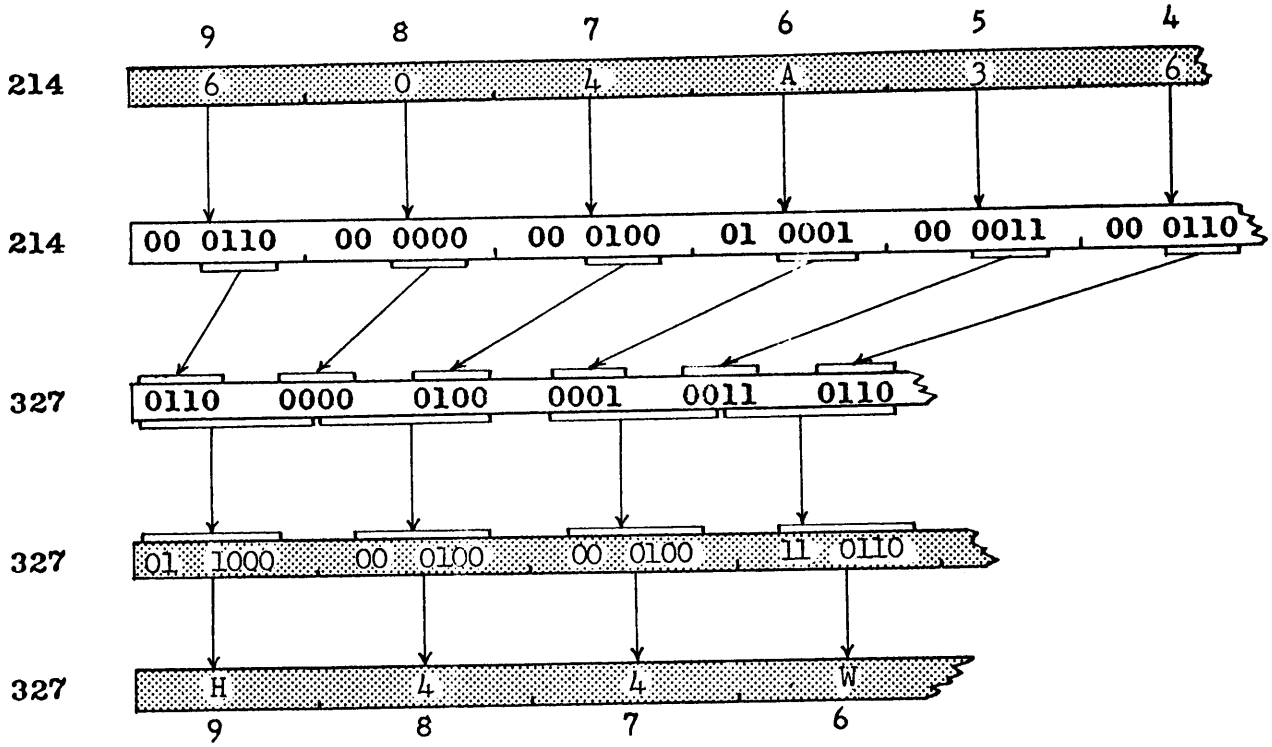
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 |
| #11 | #12 | #13 | #14 | #15 | #16 | #17 | #18 | #19 | #20 |
| #21 | #22 | #23 | #24 | #25 | #26 | #27 | #28 | #29 | #30 |

AFTER PACKING

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | #11 | #12 | #13 | #14 | #15 |
| #16 | #17 | #18 | #19 | #20 | #21 | #22 | #23 | #24 | #25 | #26 | #27 | #28 | #29 | #30 |

Example - PACK Instruction

Pack three words - starting at location 214, into two words, starting with location 327.




The next Instruction will be taken from the "Branch" address J, since a non-zero zone bit appears in character-position 6 of cell 214.

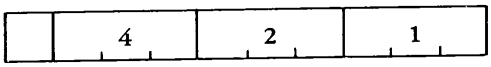
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N' syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

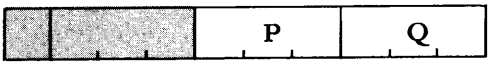
S-values of syllables

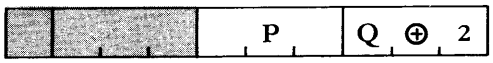
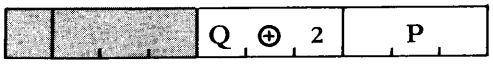



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# U N P A C K

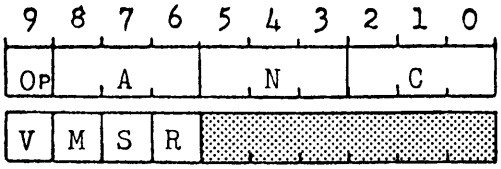
This Instruction operates upon information which has previously been PACKED.

It regards the contents of two consecutive memory cells as a collection of thirty 4-bit digits, adds a pair of zero zone-bits to each of them, and stores the resulting thirty 6-bit characters in three consecutive cells. The Instruction performs this operation on as many 2-word groups as may be specified.

UNPACK is the converse of PACK, and restores information to precisely the same form as it had before it was packed (except, of course, that any 1-bits in zone-bit positions in the original information will be replaced by 0-bits).

OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: UNPACK (UNPK)

Operation Code: P

| Ref No | M | Operat | V | Misc. | R | A | T | B | T | C | T | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UNPK | | | | A | | N | | C | | |

DEFINITIONS:

$O_P$: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first word of first 2-word group to be unpacked.

N: as modified by (OOR), is an address-type number which specifies how many consecutive 2-word groups are to be unpacked.

$0 \leq N \leq Z99$

C: base of address in which first word of first 3-word unpacked group is to be stored.

V: variation designator;
only the sign of V is relevant.

DESCRIPTION OF: UNPACK

This is not a partial-word operation.

This Instruction regards [A] and [A ⊕ 1] as consisting of thirty 4-bit digits, packed by a PACK Instruction. It adds a pair of 0-bits as zone-bits to each 4-bit digit, reconstituting it as a standard 6-bit character, and stores the resulting 30 characters in [C], [C ⊕ 1], and [C ⊕ 2].

Then [A ⊕ 2], [A ⊕ 3] are unpacked, and put away into [C ⊕ 3], [C ⊕ 4], [C ⊕ 5]. The process is repeated until N pairs have been unpacked. If N specifies zero groups to be unpacked, the Instruction performs no operation, and leaves the contents of the memory unchanged.

OVERFLOW ALARM: Will not be set by this Instruction.

Example - UNPACK Instruction

Unpack two words - starting at location 327, into three words, starting at location 214.

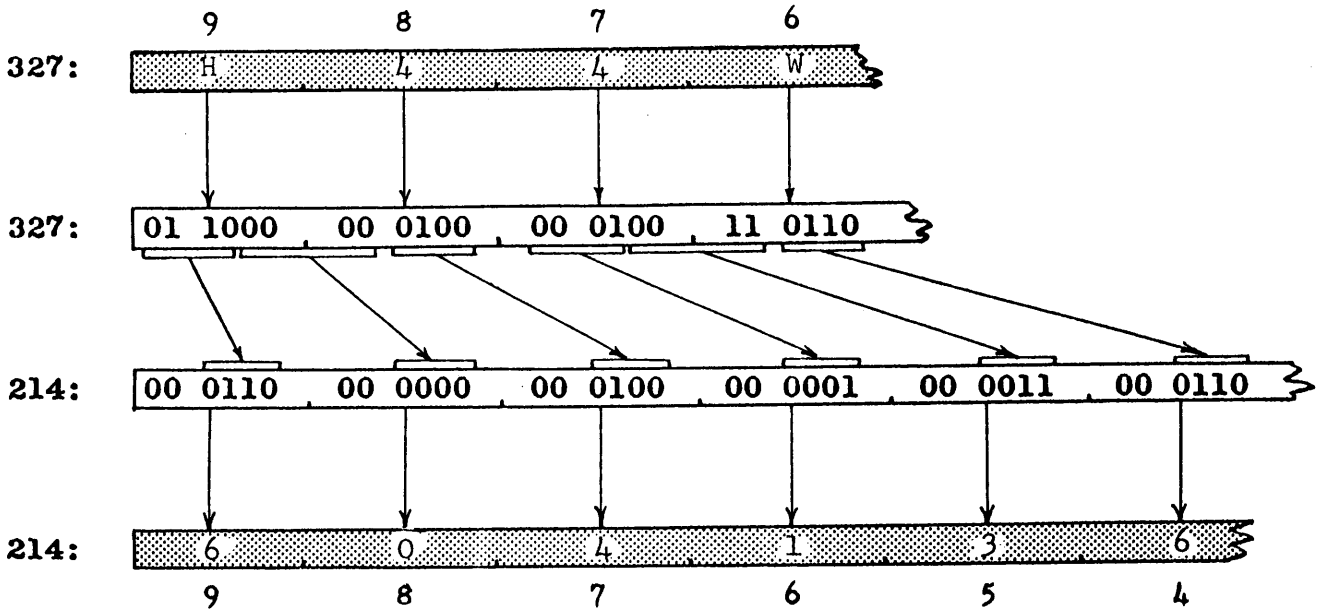


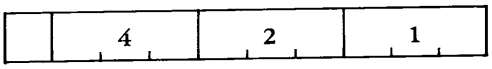
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

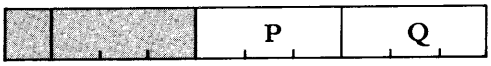
S-values of syllables

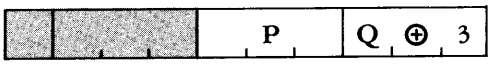
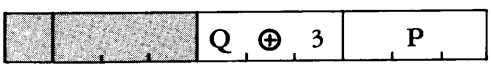



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |  |
| negative |  |

# S I F T

The SIFT Instruction calls for a standard item (the sieve) and a list of items, sorted according to some key, which are to be compared with the standard (that is, sifted through the sieve).

Those items in the list whose keys are less than, or equal to, the key of the sieve will "pass through the sieve". The first item whose key is greater than the key of the sieve will fail to pass through, and will terminate the operation.

The Instruction counts the number of items which have passed through the sieve, and also counts the total number of words in those items.

The principal uses of SIFT are:

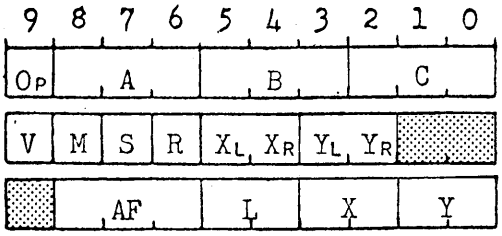
1) For table lookup. The sieve is a dummy item, set up as a structural image of the items in the list, and it contains the key whose location within the list is to be determined.

2) As part of a SIFT-SUMMARIZE operation, to summarize similar items in a sorted list. The sieve is the first item in the list itself, and SIFT counts the number of items (including the first) which have the same key as the first item. SUMMARIZE then refers to this item-count, to summarize the proper number of items.

The word-count is then used to augment an Index Register, and the SIFT-SUMMARIZE operation is repeated, causing the operation to resume at just the point where it previously stopped. The operation of SIFT-SUMMARIZE-COUNT is repeated until the entire list has been summarized.

OVERFLOW ALARM: Will not be set under any circumstances.

INSTRUCTION FORMAT:




Operation: SIFT (SIFT)

Operation Code: Q

| Ref No | n | Operat | V | Misc | R | A | T | | T | | T | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SIFT | | L | | A | | B | | C | | |
| | | 1 | | K | | N | | X: $X_L X_R$ | | Y: $Y_L Y_R$ | | |

A: Designates 1st word, 1st item in list.
B: Designates 1st word of the sieve-item.
C: Designates location in which the counts are to be stored.
L: Length of each item.
N: Number of items in list.
K: Number of keys (1 or 2) for the SIFT.
X: $X_L X_R$: Relative position of Major Key (if any), within item.
Y $Y_L Y_R$: Relative position of Minor Key within item.

DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first word, first item in the list to be sifted.

B: base of address of first word of the sieve.

C: base of address in which the counts are to be stored.

AF: address of first word, last item in the list to be sifted.

L: item length (number of words per item) $0 < L \leq 99$

X , Y : relative addresses, with respect to the first word of each item, of words containing the major and minor control keys, respectively.

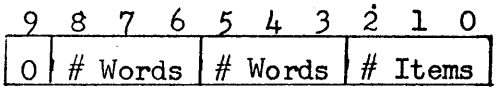
$X_L$, $X_R$: locations of left-most and right-most characters,
$Y_L$, $Y_R$: respectively, of major and minor control keys.

NOTE: X, $X_L$, $X_R$ are not relevant if only one control is used.

V: variation designator:

| V | Specifies |
|---|---|
| 0 | one control key |
| 1 | two control keys |

The sieve must be a structural image of the items in the list.



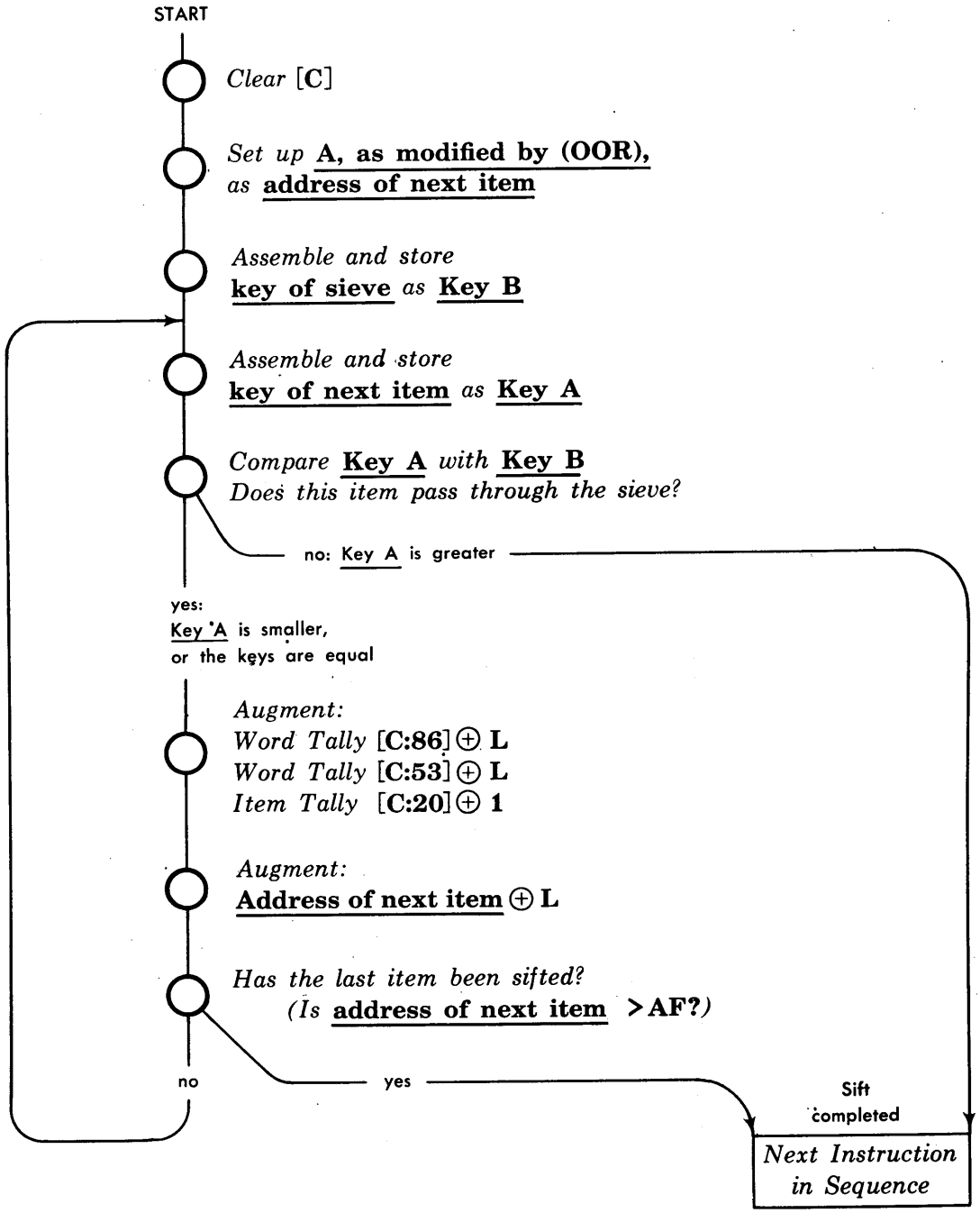
## DESCRIPTION OF: SIFT

NOTES

1) The comparison made by this Operation is the same as that made by COMPARE ALPHANUMERIC. The Instruction is, therefore, designed to operate upon data which has been sorted by use of MERGE.

2) Keys may be alphanumeric, PACKED numeric, or a combination, as in MERGE.

3) "Assembly of Key" is performed as in MERGE.

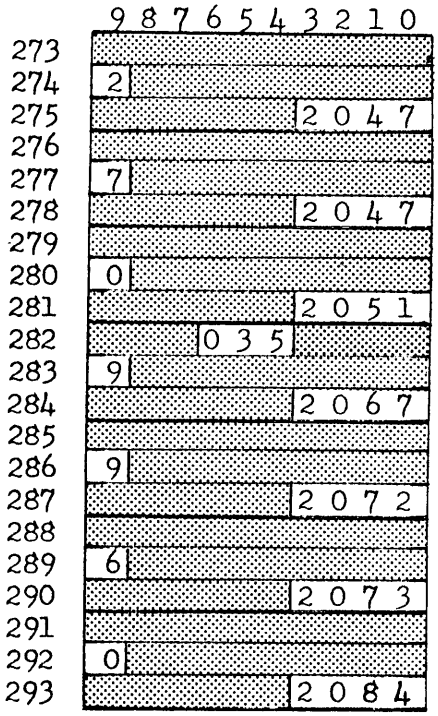


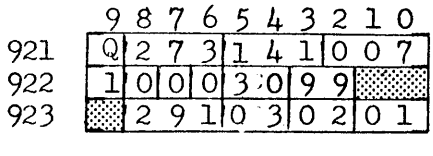

OVERFLOW ALARM: Will not be set by this Instruction.

## Example - SIFT used for Table Lookup

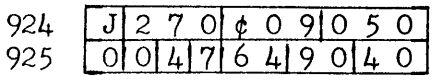
SIFT terminates when the first item is found whose control characters are greater than those of the sieve. It will not terminate on an "equal to" condition. This must be considered when using the tallies.

Assume the storage of the list of items shown below. Find the item whose control number is 2067-9. Then store the contents of the 64-field of the 1st word of that item, in 050:40 right-justified.

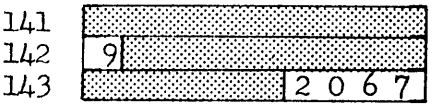



SIFT

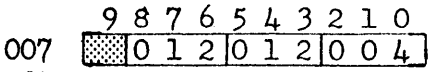



Plant the 64-field of the desired item into 050:40

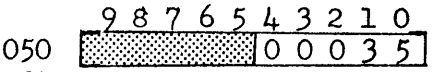



"sieve"

NOTE: ¢09:90 is used as a "throw-away" cell for the unused half of the DISTRIBUTE.




after




after

Location 007 is used as an Index Register by the DISTRIBUTE Instruction. The tallies stored in 007 include the item found. Therefore, it is necessary for the DISTRIBUTE Instruction to use as a base address a location one item-length before the beginning of the list of the items.

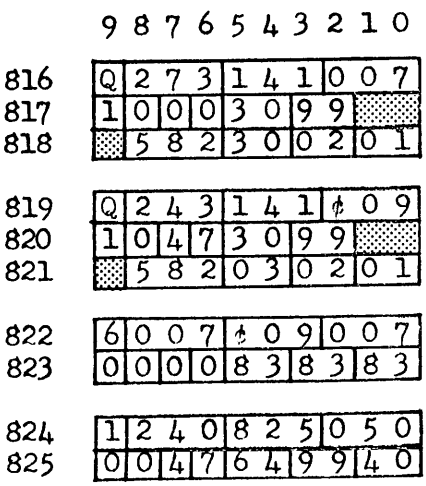
An example of SIFT used with SUMMARIZE appears on page IV-R-5.

## SIFTING LONG TABLES

In dealing with long tables, considerable time can be saved by sifting the table twice: the first time to learn what part of the table holds the desired item, and the second time to locate the exact item.

This procedure is similar to that used in locating a specific folder in a filing cabinet when each drawer bears a label showing the lowest account-number filed in that drawer. The clerk first examines these labels to learn which drawer to open, and then selects the desired folder from within that drawer.

Suppose that the example on the preceeding page had required a table of 104 items. We could use a coarse SIFT (in which the item-length is stated as 30) to examine every 10th item in the table, and then a fine SIFT which would have to examine no more than 10 items within the proper portion of the table:




Coarse SIFT. L = 30 causes every 10th item to be examined. In effect, we are sifting a table of 11 items.

Fine SIFT. Since the large group containing the desired item has already been tallied, this SIFT is referenced 30 words before the actual table.

MODIFY ADD the result of the fine SIFT (¢09) to the result of the coarse SIFT (007).

Plant the proper field of the desired item. Since this item has been tallied again, this Instruction is referenced 33 words before the actual table.

The fact that the last large group contains only 4 items, rather than 10, need give no concern, since the AF specified in both SIFTS will terminate them properly in all cases.

In using a double SIFT, it is important that the sum of:

the number of groups

plus the number of items per group

should be as small as possible. Thus, a table of 120 items might be divided into:

| | |
|---|---|
| 3 groups of 40 | 3 + 40 = 43 |
| or 5 groups of 24 | 5 + 24 = 29 |
| or 6 groups of 20 | 6 + 20 = 26 |
| or 8 groups of 15 | 8 + 15 = 23 |
| or 12 groups of 10 | 12 + 10 = 22 |
| or 10 groups of 12 | 10 + 12 = 22 |
| or 15 groups of 8 | 15 + 8 = 23 |
| and so on. | |

Since 12 and 10 comprise the pair of factors with the smallest sum, the 120 items should be divided into 12 groups of 10 items, or into 10 groups of 12 items.

This same procedure may be extended to a triple SIFT for extremely long tables. The items should be divided so that the three factors have the smallest possible sum.

On the assumption that lookups are spread evenly throughout the table, the minimum table-lengths on which it is worth-while to use multiple SIFTS are:

| | Double SIFT | Triple SIFT |
|---|---|---|
| 1 control key | 30 items | 900 items |
| 2 control keys | 25 items | 400 items |

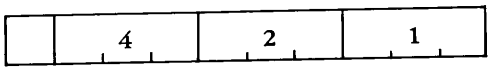
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | B syllable modified |
| 1 | C syllable modified |
| Sum | Determines combination of syllables modified. |

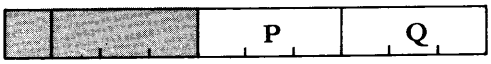
S-values of syllables

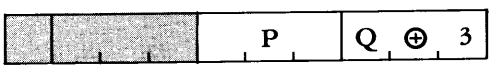
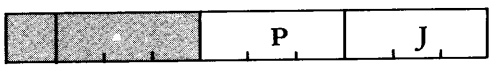
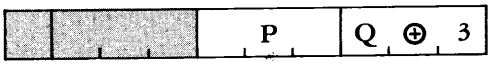
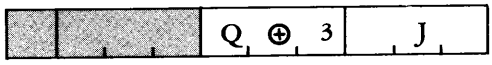



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

J: May be JC, J@

# S U M M A R I Z E

This Instruction examines, in turn, a series of Items stored in the Processor Memory, and forms the arithmetic sum, for all the Items, of the quantity appearing in a specified location within each Item.

The summary total is stored in the C-putaway field, and also in cell @00.

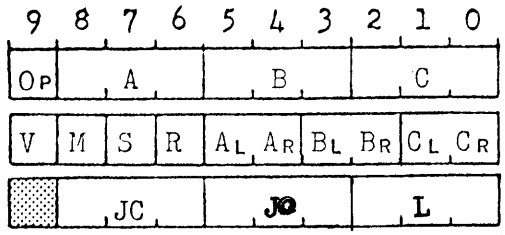
If the summary total exceeds the capacity of the C-field, the next instruction to be executed is selected from a "Branch Address" specified in the Instruction; if the summary total also exceeds the 10-digit capacity of cell @00, the next Instruction to be executed is selected from a different "Branch Address" specified in the Instruction.

OVERFLOW ALARM: Will not be set by this Instruction, since overflow automatically causes a branch.

INSTRUCTION FORMAT:

Operation: SUMMARIZE (SUMM)

Operation Code: R




| Ref No | M | Operat | V | Mixed | R | A | T | B | T | C | T | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SUMM | | L | | A: AL-AR | | B: BL-BR | | C: CL-CR | | |
| | | 1 | | | | Jc | | J@ | | | | |

DEFINITIONS:

OP: operation code.

M: auto-monitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

A: base of address of first quantity to be included in the Summary.

AL, AR: locations of left-most and right-most character-positions, respectively, of quantity to be summarized.

B: base of address containing the number of quantities to be summarized. This number is an address-type number.

$0 \leq [B] \leq Z99$

BL, BR: locations of left-most and right-most character-positions, respectively, of the field which designates the number of quantities.

C: base of address in which summary total is to be stored.

NOTE: Summary total is also stored, with sign, in @00:90

CL, CR: locations of left-most and right-most character-positions, respectively, of the C-field.

L: amount by which [A] is to be augmented to find successive quantities to be included in the summary. (L specifies the length of the items.)

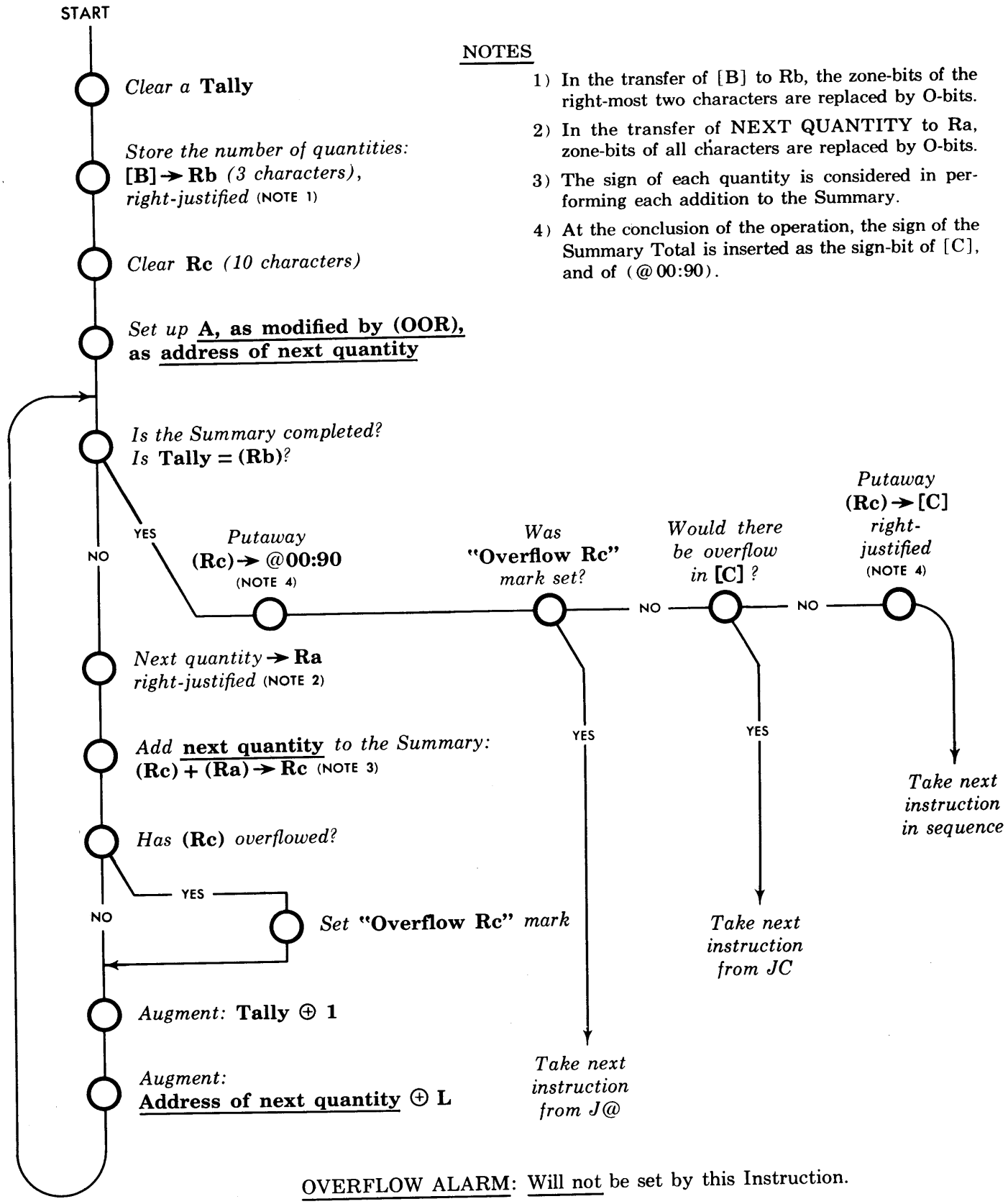
JC: address of next instruction, if summary total exceeds the capacity of [C].

J@: address of next instruction, if summary total exceeds the capacity of @00:90.

Jumps only after summary is complete.

V: variation designator;
only the sign of V is relevant.

## DESCRIPTION OF: SUMMARIZE



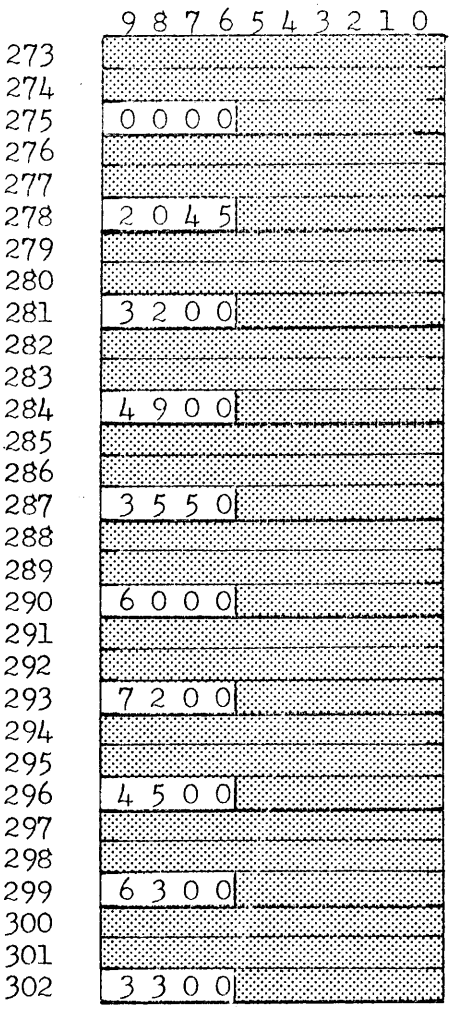

NOTES

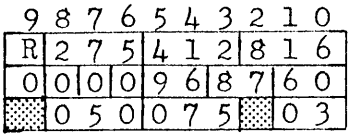
1) In the transfer of [B] to Rb, the zone-bits of the right-most two characters are replaced by O-bits.
2) In the transfer of NEXT QUANTITY to Ra, zone-bits of all characters are replaced by O-bits.
3) The sign of each quantity is considered in performing each addition to the Summary.
4) At the conclusion of the operation, the sign of the Summary Total is inserted as the sign-bit of [C], and of (@00:90).

OVERFLOW ALARM: Will not be set by this Instruction.

## Example - SUMMARIZE Instruction

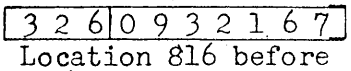
Summarize the amounts stored in partial word field 96 of the third word of each of the following 3- word items:



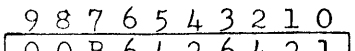





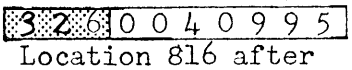
Location 412



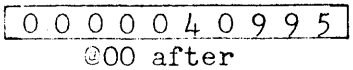

Location 816 before




@00 before




Location 816 after




@00 after

EXAMPLE - SIFT and SUMMARIZE Instructions

There is in memory a list of 35 one-word items describing sales. They contain the class of merchandise, the salesman's number, and the amount of the sale.

9 8 7 6 5 4 3 2 1 0

Cell 100 | C C C | S S | $ $ $ $ $ |

Summarize the sales by salesman; list the number of sales made by each salesman. Then accumulate total sales and number of salesmen active this day.

Cell 200 | S S | # # | $ $ $ $ $ $ |

Cell 300 | $ $ $ $ $ $ $ $ | # # |

LIST

9 8 7 6 5 4 3 2 1 0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 100 | 0 0 2 | 0 2 | 0 5 5 9 1 |
| 101 | 0 0 2 | 0 2 | 2 6 1 0 4 |
| 102 | 0 0 2 | 0 2 | 0 0 4 0 0 |
| 103 | 0 4 5 | 0 2 | 3 2 8 1 1 |
| 104 | 0 4 5 | 0 2 | 1 0 7 7 8 |
| 105 | 0 4 5 | 0 2 | 5 1 0 6 5 |
| 106 | 0 4 5 | 0 2 | 3 7 1 3 5 |
| 107 | 2 1 8 | 1 4 | 1 7 3 2 2 |
| 108 | 2 1 8 | 1 4 | 4 4 7 2 0 |
| 109 | 2 1 8 | 1 4 | 0 2 4 7 3 |
| 110 | 7 7 0 | 2 5 | 9 5 7 9 7 |
| 111 | 0 0 2 | 3 8 | 1 2 8 5 0 |
| 112 | 0 0 2 | 3 8 | 2 1 2 2 5 |
| 113 | 0 4 5 | 3 8 | 4 1 0 4 0 |
| 114 | 2 1 8 | 3 8 | 5 0 0 6 5 |
| 115 | 3 5 9 | 3 8 | 7 1 2 3 5 |
| 116 | 3 5 9 | 3 8 | 0 0 8 1 0 |
| 117 | 3 5 9 | 3 8 | 5 1 4 7 5 |
| 118 | 3 5 9 | 3 8 | 3 7 5 1 4 |
| 119 | 4 1 2 | 3 8 | 1 2 4 3 0 |
| 120 | 4 1 2 | 3 8 | 5 6 5 1 0 |
| 121 | 4 1 2 | 3 8 | 1 1 8 0 0 |
| 122 | 5 7 2 | 3 8 | 7 5 1 1 5 |
| 123 | 5 9 0 | 3 8 | 0 5 0 6 5 |
| 124 | 7 7 9 | 3 8 | 2 7 4 0 5 |
| 125 | 0 4 5 | 6 1 | 0 1 4 1 9 |
| 126 | 0 4 5 | 6 1 | 3 6 8 1 0 |
| 127 | 0 4 5 | 6 1 | 3 3 0 2 5 |
| 128 | 2 1 8 | 6 1 | 1 9 6 5 5 |
| 129 | 3 5 9 | 6 1 | 4 2 5 0 8 |
| 130 | 4 1 2 | 6 1 | 4 9 8 3 5 |
| 131 | 5 7 2 | 6 1 | 2 1 8 3 0 |
| 132 | 5 9 0 | 6 1 | 1 2 2 3 5 |
| 133 | 5 9 0 | 6 1 | 5 7 8 7 6 |
| 134 | 0 0 2 | 6 6 | 0 0 3 9 9 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 200 | 0 2 | 0 7 | 1 6 3 8 8 4 |
| 201 | 1 4 | 0 3 | 0 6 4 5 1 5 |
| 202 | 2 5 | 0 1 | 0 9 5 7 9 7 |
| 203 | 3 8 | 1 4 | 4 7 4 5 3 9 |
| 204 | 6 1 | 0 9 | 2 7 5 1 9 3 |
| 205 | 6 6 | 0 1 | 0 0 0 3 9 9 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 300 | 0 1 0 7 4 3 2 7 | 0 6 |

PROGRAM

9 8 7 6 5 4 3 2 1 0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 387 | 1 | 3 8 8 | 3 8 8 | 0 0 2 |
| 388 | 0 | 0 0 0 | 0 9 9 | 9 9 0 |

ADD: Clear (002).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 389 | Q | 1 0 0 | 1 0 0 | ¢ 0 8 |
| 390 | 0 | 0 6 | 2 3 4 | 6 5 | 2 9 |
| 391 | 0 | 1 3 4 | 0 1 | A B | 0 0 |

SIFT: Tally the like items in ¢08.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 392 | R | 1 0 0 | ¢ 0 8 | 2 0 0 |
| 393 | 0 | 0 5 | 2 4 0 | 2 0 | 5 0 |
| 394 | 7 | A 3 6 | F 5 0 | R | 0 1 |

SUMMARIZE: The like items.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 395 | H | 1 0 0 | ¢ 0 8 | 2 0 0 |
| 396 | 0 | 0 5 | 2 6 5 | 1 0 | 9 6 |

COMBINE: Salesman's #, and # of sales.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 397 | 6 | 0 0 2 | 3 9 8 | 0 0 2 |
| 398 | 1 | 0 0 0 | 2 0 9 | 9 2 0 |

MOD ADD: Add 1 to (002:20)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 399 | E | ¢ 0 8 | 0 3 5 | 3 8 9 |
| 400 | 0 | 0 0 0 | 0 8 3 | F | 3 6 2 |

COUNT: Augment (002:93) and test (002:86) for end.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 401 | R | 2 0 0 | 0 0 2 | 3 0 0 |
| 402 | 0 | 0 0 0 | 5 0 | 2 0 | 9 2 |
| 403 | 3 | 7 0 4 | 7 0 4 | 0 | 0 1 |

SUMMARIZE: Total sales.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 404 | 2 | 0 0 2 | 4 0 5 | 3 0 0 |
| 405 | 0 | 0 0 0 | 2 0 | 9 9 | 1 0 |

Store number of salesmen active this day.

# MAGNETIC TAPES

## GENERAL

The following terms are used in this section:

SOURCE TAPE: Any magnetic tape from which information is being read.

DESTINATION TAPE: Any magnetic tape upon which information is being written.

ITEM: A body of information which it is convenient (in a given context) to regard as a unit. Thus in one context, a list of all the transactions affecting a single account may be considered a single item; while in another context, each individual transaction in the list may be considered an item.

FILE-ITEM: An item in the file; an account.

RECORD: A body of information, recorded on magnetic tape, and bounded by a Beginning-of-Record Mark (BRM) and an End-of-Record Mark (ERM). A record will often contain a single item; however it may, at the programmer's convenience, contain several items; or a single item may be divided into several records.

GULP: All the information read from, or written on, magnetic tape with a single Instruction. A gulp may consist of one, or many, records.

CONTROL RECORD MARK (CRM): The character "v" (code 11 1101), recorded in Character-Position 4 of the first word of a record.

CONTROL RECORD, CRM RECORD: A record containing a CRM. Such a record is most often used as an End-of-File sentinel.

REJECT MARK: The character "w" (code 11 1110) recorded in Character-position 3 of the first word of a record.

LEADER; TRAILER: At each end of a magnetic tape, there is a portion of the tape upon which information cannot be recorded, which serves to attach the tape to the reel, and which is commonly referred to as the "leader". It is often important, however, to refer unambiguously to the leader at one end of the tape or the other; in order to avoid expressions such as "leading leader" and "trailing leader", the two leaders are herein referred to as LEADER and TRAILER, respectively.

The functions performed by the NCR 304 Magnetic Tape System are:

READ MAGNETIC TAPE
- One or many records
- Fixed or variable length records
- All or part of each record
- Index forward or backward without reading

WRITE MAGNETIC TAPE
- One or many records
- Fixed or variable length records

WRITE-COPY
- Using Source and Destination Tapes simultaneously
- Off-line Copy

WRITE-COPY-READ
- Using Source and Destination Tapes simultaneously
- On-line Copy

REWIND
- Any Handler in the System
- Off-line rewind
- Any number of Handlers simultaneously

The System has been designed to furnish the utmost in reliability and convenience in providing for all non-routine circumstances which may arise during Magnetic Tape operations. Many of these circumstances, such as end of tape, reading error, or writing error, can be programmed for; all such circumstances are automatically detected and identified by the system. Circumstances which cannot be programmed for, such as broken tape, or reference to a non-existent Controller or Handler, are also detected automatically, and cause immediate termination of all Processor operations, with identifying information displayed in the Control Console lights.

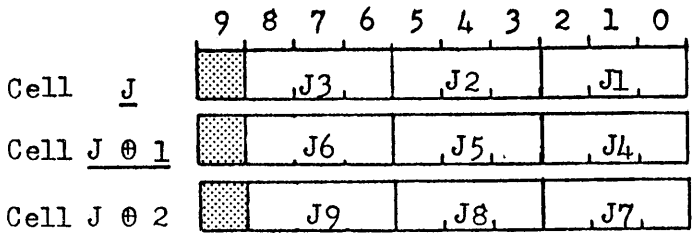
An important feature of the 304 Magnetic Tape System is its ability to terminate any operation under a wide variety of automatically detected circumstances. Termination may occur whether or not the Instruction, as specified by the programmer, has been completely executed. If circumstances are such that it is not practicable or advisable to execute a Magnetic Tape

Instruction at all, that Instruction is aborted; it is not executed, and an alternate Instruction is automatically selected for execution instead.

In order to provide this facility, every Magnetic Tape Instruction designates, in the 20-field of the first word, an address J, which is the location of the first word of a 3-word Jump Table. This table contains nine addresses, or "exits" from the Tape Instruction; each exit corresponds to some "non-normal" circumstance which may arise before, or during, execution of the Instruction. Each of these circumstances will be detected automatically whenever it arises, whereupon execution of the Instruction will be either aborted or terminated, and the Processor will take its next Instruction from the appropriate address in the Jump Table. If no "non-normal" circumstance arises, execution of the Instruction terminates "normally", and the Processor executes the next Instruction in the normal sequence.

In most operations, a single Jump Table may be used for all Magnetic Tape Instructions.

The addresses in the Jump Table are numbered J1 through J9, and are arranged in the following format:




The detailed significance of each exit is shown in the Table which appears at the end of this section. Briefly, the exits correspond to the following respective circumstances, many of which can cause either "branch

without execution" or "branch after termination", as appropriate:

J1 - Busy, or Use Lockout
J2 - Error during Read
J3 - Control Record Mark
J4 - End of Destination Tape
J5 - End of Source Tape
J6 - Error during Write
J7 - Error during Non-Write portion of Write-Copy or Write-Copy-Read.
J8 - Non-Equal Termination of Write-Copy or Write-Copy-Read.
J9 - Error during Write portion of Write-Copy or Write-Copy-Read.

The branch conditions are divided into two groups:

Branch without execution of the tape operation.
Branch after termination of the tape operation.

The conditions which cause branch without execution are the same for all Magnetic Tape Instructions, and will not be repeated under the individual Instruction descriptions. They are:

J1 {
- Handler busy rewinding
- Controller busy copying or repositioning
- Handler in Use Lockout state

- One of the following conditions had been encountered during a previous off-line COPY on the same Controller:
  - J7 - Error in copying
  - J8 - "Inequality" termination
  - J5 - End of Source Tape
  - J4 - End of Destination Tape

If any of these conditions exist, the current Magnetic Tape Instruction will not be executed; instead, the Processor will select its next Instruction from the appropriate address stored in the Jump Table named by the aborted Instruction.

Once execution of an Instruction has begun, a number of circumstances may cause it to terminate, and some of those circumstances will cause a branch after termination.

The description of each Instruction, in later pages, includes a list of all conditions which cause termination of that Instruction. Some of the conditions are defined as "normal", and cause the Processor to select the next Instruction in the normal sequence. The other conditions are defined as "non-normal" and cause the Processor to select its next Instruction from the appropriate address stored in the Jump Table named by the terminated Instruction.

It will seem strange at first that many circumstances which terminate execution of an Instruction while it is still incomplete are defined as "normal" and do not cause a branch. However, in the design of the system, the list of terminations which were to be defined as "non-normal" was carefully chosen for maximum convenience to the programmer.

READ TAPE in particular, may often terminate "normally" even though execution of the Instruction is incomplete. Since the design philosophy of the system must be understood in order to program it most effectively, some of the unusual terminations of READ TAPE, which are defined as "normal", will be discussed in detail.

It must be understood, first, that although the READ TAPE Instruction says "Read N records", it may always terminate before the full N records have been read, and this termination is defined as "normal" if at least one record has been read without incident. Therefore, the READ TAPE Instruction must be interpreted as "Read not more than N records", with the recognition that other circumstances will determine exactly how many records are actually to be read.

In reading Magnetic Tape, the programmer must allocate an area of memory to receive the information being read. In order to operate most

efficiently, and minimize tape starts and stops, he should arrange to fill this area as completely as possible with each READ TAPE Instruction.

If the file being processed consists of fixed-length records, there is no difficulty in deciding how many records shall be specified in the Instruction.

But if the file consists of variable-length records, there is no way of anticipating, in advance of each READ TAPE Instruction, how many records will fit into the allocated memory space.

Therefore the READ TAPE Instruction specifies:

> "Read as many records (not exceeding N) as will fit into the allocated memory space. Then terminate, and note how many records were read."

The tally of the number of records read is automatically stored in Cell @00, as described later, and in this case, the termination is defined as "normal", if at least one record has been read. With a single Instruction, following the READ TAPE, the programmer then plants this tally in a COUNT Instruction, so that the correct number of records will be processed, and also in the succeeding WRITE TAPE Instruction, so that the correct number of records will be recorded on the updated file tape.

Even though fixed-length records are being processed, a READ TAPE Instruction may be terminated by a reading error, or by finding a Control Record, before the full N records have been read. Unless the error record, or the Control Record, is the very first record read, this information is of no value at this point, since the preceding records read must be processed and disposed of before dealing with the error record or Control Record. Therefore, if at least one record has been read, before the error record or

the Control Record, this termination is also defined as "normal", and the tally of records read, not including the error record or the Control Record, is automatically stored in Cell @00. On the next repetition of the READ TAPE Instruction, the Control Record or (if the error persists) the error record will be encountered again, but this time it will be the first, and only, record read. The Processor will branch after termination of the READ TAPE, to execute that portion of the program which deals with READ errors, or with Control Records.

Similarly, if READ TAPE is terminated by reaching the end of the tape, less than N records will have been read, but as long as at least one record has been read, the information that end-of-tape has been reached is of no value until after those records which have been read are processed and disposed of. Therefore, this termination, too, is defined as "normal" if at least one record has been read, and the tally of records read is automatically stored in Cell @00. The next time the READ TAPE is executed, the end of the tape will again be encountered, and the Instruction will terminate. But this time, the Processor will not have read "at least one record"; no records will have been read, and the Processor will branch after termination of the READ TAPE, to execute that portion of the program which alternates Handlers so that the next reel of the file may be processed.

Thus, by defining termination as "normal" whenever at least one record has been read without incident, information about an unusual situation is made available at just the moment when it can most efficiently and conveniently be used.

It is not necessary to discuss in detail other unusual terminations of Magnetic Tape Instructions which are defined as "normal", since increasing

familiarity with the system will bring out the reasoning behind the definitions.

REJECT MARK:

A Reject Mark is the character "w" (code 11 1110) recorded in character-position 3 of the first word of a record. Such a record will be "rejected" by COPY and SEARCH, in the sense that the record will never cause the operation to terminate, regardless of any correspondence between the Search Key in the record, and the Search Control stored in Memory. A Reject Mark has no significance during READ.

The Reject Mark is used with files in which items may be longer than the maximum record-length (100 words); such items will be recorded as two or more records -- a "first" record, and as many "trailers" as necessary.

In some files, each item consists principally of account number, descriptive information, several balances, and a sequential list of transactions. The main posting operation consists of locating each active account, adjusting the balances, and adding the current transactions to the list. In such a case, the balances will be carried in the last record, the account number repeated in each record, and every record except the last one for each file-item will contain the Reject Mark. Then the COPY will stop on the last record for each file-item, with only this one record being brought into the Processor for updating. There will be a periodic (perhaps monthly) operation in which the descriptive information and transaction lists will be printed out, and each file-item reduced to a single minimum-length record; this operation will require Reading and Writing every record, and the Reject Mark will have no significance at that time.

Some other files containing multi-record items, require that a good deal of identifying information be stored in each "first" record, and various operations on the file may call for different Search Keys. In order that a "trailer" record (containing information irrelevant to the Search) should not, by coincidence, satisfy the Search Condition, the programmer need only arrange to record the Reject Mark in every record except the first one for each file-item, and the SEARCH or COPY will stop only on "first" records. The presence of the Reject Mark in a "trailer" record nullifies, in effect, the preceding ERM-BRM during the SEARCH or COPY, and therefore each file-item is treated by the SEARCH or COPY as though it were a single record of indefinite length.

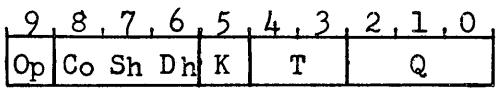
CONTROL RECORD MARK (CRM):

A CRM is the character "v" (code 11 1101) recorded in character-position 4 of the first word of a record. A Control Record will cause termination of any READ, COPY or SEARCH, with appropriate branches as specified in the Jump Table. A common use of the CRM is to mark the end of a file, or of a section of a file.

If a record contains both a CRM and a Reject Mark, it will be treated as a Reject Record by SEARCH and COPY, but as a CRM Record by READ.

## C E L L  @ 0 0

As its last act during execution, before permitting the Processor to proceed to the next Instruction, every Magnetic Tape Instruction automatically stores certain information about itself in Cell @00, where the program may later refer to that information:




Op: Operation code of the Magnetic Tape Instruction which set up (@00).

Co, Sh, Dh: Controller, Source-Handler, Destination-Handler numbers <u>actually used</u> by the Instruction.
i.e.-Those <u>named</u> in the Instruction, <u>modified</u> by the contents of an Index Register.

K: Differentiates between: Busy or Use Lockout
Branch without execution
or after termination

T: After READ, Tally of number of complete records correctly read.
After any other Magnetic Tape Instruction, <u>T = 0</u>.

Q: Address of the Instruction which set up (@00).

<u>Significance of K</u>:

After exit to J1 -- Leftmost bit-position of K contains:

1-bit if Handler is in Use Lockout state.
0-bit if Controller or Handler is busy.

After any other exit:

K = 0 if Branch after termination.
K $\neq$ 0 if Branch without execution.

## INDEX REGISTERS and FILE MANAGEMENT

As with all other Instructions, the first word of every Magnetic Tape Instruction can be modified by the contents of an Index-Register, as specified by the S and R digits in the Instruction. For this function, the A-syllable of the Instruction is treated just as though it were an address.

In examining the Instruction formats, it will be seen that Co (Controller number), Sh (Source-Handler number), and Dh (Destination-Handler number) are specified in the same character-positions in every Magnetic Tape Instruction, although Dh is irrelevant to some Instructions, and Sh is irrelevant to others.

As a result, it becomes quite convenient to designate all Magnetic Tape Instructions referring to the Source and Destination Tapes of a single file, as relative to the same Index-Register. Usually, in fact, every Magnetic Tape Instruction will be written as though it referred to Controller #0, Source-Handler #0, and Destination-Handler #0, with the actual numbers being specified entirely in the Index-Register. Therefore, for example, to charge Handlers in order to process successive reels of a long file, it is sufficient to alternate a single digit in the Index-Register for the Source-Handler or for the Destination-Handler, and then resume the program; each of the many Magnetic Tape Instructions used in connection with that file will thereby be modified to refer to the alternate Handler.

| BUSY OR USE-LOCKOUT J 1 | ERROR DURING READ J2 | CONTROL RECORD MARK CRM J 3 | END-OF-TAPE WARNING SIGNAL ON DESTINATION TAPE J 4 | END OF SOURCE TAPE J 5 | | |
|---|---|---|---|---|---|---|
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | READ |
| | ERROR IN THE FIRST RECORD READ.<br>TALLY IN @00:43 IS ZERO.<br>(@00:55)=0 | CRM IN THE FIRST RECORD READ.<br>ONLY THAT RECORD IS READ.<br>THE TALLY IN @00:43 IS 1. (@00:55)=0 | | END OF TAPE WAS ENCOUNTERED BEFORE ANY RECORDS WERE READ.<br>TALLY IN @00:43 IS ZERO.<br>(@00:55) = 0 | EXIT<br>AFTER TERMINATION | |
| | TO RE-READ THE ERROR RECORD. | TO READ THE RECORD FOLLOWING THE CRM RECORD. | | AT THE TRAILER. | TAPE POSITION AFTER TERMINATION | |
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | INDEX FORWARD |
| | ERROR IN THE FIRST RECORD READ.<br>TALLY IN @00:43 IS ZERO.<br>(@00:55)=0 | CRM IN THE FIRST RECORD INDEXED OVER.<br>ONLY THAT RECORD IS INDEXED OVER.<br>THE TALLY IN @00:43 IS 1.<br>(@00:55)=0 | | END OF TAPE WAS ENCOUNTERED BEFORE ANY RECORDS WERE INDEXED OVER.<br>TALLY IN @00:43 IS ZERO.<br>(@00:55) = 0 | EXIT<br>AFTER TERMINATION | |
| | READ RECORD FOLLOWING ERROR RECORD. | TO READ THE RECORD FOLLOWING THE CRM RECORD. | | AT THE TRAILER. | TAPE POSITION AFTER TERMINATION | |
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | INDEX BACKWARD |
| | | | | LEADER WAS ENCOUNTERED BEFORE ANY RECORDS HAVE BEEN INDEXED OVER.<br>TALLY IN @00:43 IS ZERO.<br>(@00:55) = 0 | EXIT<br>AFTER TERMINATION | |
| | | | | AT THE LEADER | TAPE POSITION AFTER TERMINATION | |
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | WRITE |
| | | | WARNING SIGNAL DURING WRITE, ONLY IF V = 0,2<br>THE FULL N RECORDS ARE WRITTEN.<br>TAPE FROM LAST ERM TO TRAILER IS ERASED.<br>(@00:55) = 0 | | EXIT<br>AFTER TERMINATION | |
| | | | AT THE TRAILER. | | TAPE POSITION AFTER TERMINATION | |
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | WRITE-COPY |
| | | | | | EXIT<br>AFTER TERMINATION | |
| | | | | | TAPE POSITION AFTER TERMINATION | |
| BUSY COPYING<br>REWINDING<br>REPOSITIONING<br>(@00:55)< "+"<br>USE-LOCKOUT ON HANDLER<br>(@00:55)> "+" | | | WARNING ON DESTINATION TAPE DURING PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | END OF SOURCE TAPE DURING COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER.<br>(@00:55) ≠ 0 | EXIT<br>WITHOUT EXECUTION | WRITE-COPY-READ |
| | | CRM HAS TERMINATED THE COPY-READ WITH CRM RECORD IN THE MEMORY.<br>(@00:55)=0 | WARNING ON DESTINATION TAPE DURING WRITE OR COPY PORTION.<br>FINISH WRITING OR COPYING THE RECORD DURING WHICH THE WARNING WAS SENSED.<br>ERASE DESTINATION TAPE TO TRAILER.<br>(@00:55) = 0 | END OF SOURCE TAPE HAS TERMINATED THE COPY-READ.<br>(@00:55) = 0 | EXIT<br>AFTER TERMINATION | |
| | | SOURCE TAPE TO READ THE RECORD FOLLOWING THE CRM RECORD.<br>DESTINATION TAPE TO WRITE THE CRM RECORD. | SOURCE TAPE TO READ THE NEXT RECORD.<br>DESTINATION TAPE AT THE TRAILER. | SOURCE TAPE AT THE TRAILER.<br>DESTINATION TAPE TO WRITE THE NEXT RECORD. | TAPE POSITION AFTER TERMINATION | |

## STANDARD TAPE EXECUTIVE PROGRAM

In writing the programs for any file-processing operation, the programmer soon learns that processing the data is only part of his task. He must also devote a good deal of attention and effort to a wide range of problems, which may arise at any point during the processing operation, and which are concerned with management of the file (as distinct from processing the information in the file). File Management includes the sequential processing of successive reels of Magnetic Tape within the file, and proper handling of those input and output tapes which are relevant to the file being processed.

Conditions such as Busy, End of Tape, Error, End of Input, End of File, &c, must all be detected whenever they occur, and each condition requires that appropriate "housekeeping" action be taken within the program. These "housekeeping" chores are numerous, and may be classified as follows:

1) Testing, in order to detect that some situation has arisen which requires "housekeeping" action.

2) Identifying the condition, and branching to the proper subroutine to take care of precisely that single condition.

3) Further testing and identifying, since two or more such conditions might arise simultaneously.

4) Writing standard programs to perform those chores which are substantially the same for all file operations.

5) Writing a few additional programs for handling those conditions whose significance is unique to the immediate operation being performed.

It has already been pointed out that the electronics of the Magnetic Tape System automatically perform the first three functions.

Not only is the programmer thereby relieved of a good deal of time-consuming (and therefore expensive) effort, but it becomes impossible for him to overlook, and thus fail to provide for, any circumstance or combination of circumstances whose possibility does not happen to occur to him.

In order that the programmer may also be relieved of the necessity of providing for function 4, a Standard Tape Executive Program (STEP) has been written to perform all routine chores for him. The programmer may, in a sense, look upon STEP as having been "built in" to the Processor since, from his point of view, it occupies no space in the Processor Memory.

The entire STEP program is stored in the "special cells" (@00-@99, ¢00-¢99, □00-□99, Δ00- Δ99), leaving the full capacity of the Main Memory available for the operating program and for data. No memory space is sacrificed to these chores, and no program "overlays" are ever necessary in order to perform them.

Even though the chores which STEP performs for the programmer are substantially the same for all file operations, there are many options available to the programmer within STEP . He may wish to intervene with some special programming of his own in the middle of an otherwise routine chore, or he may wish to omit certain features of STEP altogether; to do any of these, he need merely specify to STEP which of the available options he wishes to exercise.

The detailed characteristics of STEP have been published in a separate manual, but they may be summarized briefly here:

LABEL-CHECK:

The first record on each reel of Magnetic Tape will be recorded as a descriptive label, containing File Name; Reel Number within the File; Date Recorded; and (if desired) Date on which the information becomes obsolete, when the tape may safely be used for other purposes.

Before permitting each new Source-Tape reel to be processed, **STEP** verifies that the correct reel, of the correct file, recorded on the correct date, is about to be processed.

Before permitting each new Destination-Tape reel to be processed, **STEP** verifies (if desired) that the information previously recorded on this tape is obsolete; **STEP** then records a new label-record on this tape, to identify it when it will later be used as a Source-Tape. The programmer will specify, separately for each file, whether or not this check is desired, and if so, he will then specify the length of time for which that file must be protected.

END OF TAPE:

Rewind the Handler; set Use Lockout if desired; if this is End of Source-Tape inform the operator (through the Console Typewriter) which reel is now to be mounted on the rewound Handler; if this is End of Destination-Tape inform the operator what label is to be placed on the reel which is to be removed from the rewound Handler.

If a multi-reel file is being processed, modify the appropriate Index-Register so that the entire program now refers to the alternate Handler; if End of Tape is encountered within what should be a single-reel file (no alternate Handler specified), **STEP** will type an appropriate alarm on the Console Typewriter, and halt the Processor.

The programmer has the option of inserting a program of his own, before the Handler is rewound.

ERROR:

Repeat a specified number of times; resume if satisfactory; otherwise halt.

DEFECTIVE SPOT ON THE MAGNETIC TAPE SURFACE:

Automatically skip over all defective spots without attention from the programmer or the operator; provide a record of the number of such spots on each reel.

BUSY:

Wait; that is, continue to repeat the same Instruction until the Controller is no longer busy, and can accept the operation.

The programmer has the option, if there is alternative work to be time-shared, of taking this exit out of **STEP**, and programming it himself.

USE LOCKOUT:

Halt, ready to resume. The programmer has the option of programming this condition himself.

RESCUE POINTS:

If desired, **STEP** will establish a rescue point each time a reel of Magnetic Tape is rewound, and perform a "memory dump" on a specified Magnetic Tape.

BOOTSTRAP FOR RESCUE PROGRAM:

The Rescue Program reads the last "memory dump" back into Memory, positions every reel of Magnetic Tape at the place corresponding to the rescue point, and resumes execution of the program from there.

Note that, since a rescue point is established just before beginning each new reel of Magnetic Tape, the latest rescue point always corresponds to the precise reels of tape which are mounted on the Handlers at any moment. Therefore, the rescue procedure requires no manual reel-changing, and operates entirely under the automatic control of the Rescue Program.

EXECUTIVE PROGRAM:

Furnishes a simple, convenient tool for proceeding from one program to another, either automatically under program control, or manually under operator control. All tape-exit conditions which may arise while reading or recording program-tapes are covered.

ALWAYS READ "N" RECORDS:

There are occasional circumstances in which the programmer, having called for the reading of "N" records from a Magnetic Tape, is not content with obtaining fewer than "N" records. He then follows the READ with COMPARE NUMERIC; if "N" is greater than (@00:43) he names as a branch address a specified cell within **STEP** , whereupon a special subroutine will perform all necessary "bookkeeping" and cause the remaining records to be read into the proper area of Memory.

STANDARD HALT LOCATION:

It often happens that some exits cannot (theoretically) occur while running certain programs. For example, a program which does not use COPY is not expected to encounter conditions which

can only arise during a COPY. Such exits might occur as the result of malfunction or programming error; and in any case, the 304 Magnetic Tape System requires that every branch be specified, whether or not the programmer thinks it can occur. For such branches, he will specify the Standard Halt Location within **STEP** . From this point, a standard (and simple) prescribed procedure will permit the operator to resume the processing, invoke the Rescue Program, or call for a complete re-run; each of these options may be chosen with or without a printout of the contents of Memory.

**AS A CONSEQUENCE OF THESE FEATURES OF STEP, THE ONLY TAPE CONDITIONS WITH WHICH THE PROGRAMMER NEED CONCERN HIMSELF ARE:**

** CONTROL RECORD MARK:
** INEQUALITY TERMINATION OF COPY:

Since the programmer may, in designing his program, attach any significance he chooses to these two circumstances, **STEP** does not provide for these exits. The programmer write his own programs for these two circumstances.

** BUSY:

If the programmer wishes to time-share a COPY with processing, he will, of course, program this exit himself. Otherwise he will not concern himself with it.

A complete SUMMARY OF TAPE-EXIT TABLE appears on the following pages, in which all possible Tape-Exit conditions are listed. This table is, of course, quite formidable; yet it should be appreciated that (aside from the comprehensiveness due to a highly sophisticated Magnetic Tape processing system), the entries in the table are typical of the conditions which must always be detected, identified, and programmed for, whenever an Electronic Data-Processor uses Magnetic Tape Files.

This table, however, is of little more than academic interest to the programmer, except as an indication of the amount of work which the 304 Magnetic Tape System, and **STEP**, have removed from his shoulders. He may thus devote his entire time and effort to the processing job itself, and write his programs as though all Magnetic Tape equipment remained eternally perfect, and as though all Magnetic Tapes were in fact endless.

# SUMMARY OF

# TAPE-EXIT TABLE

**SHOULD TWO OR MORE CONDITIONS ARISE SIMULTANEOUSLY THEY ARE TESTED IN THE FOLLOWING SEQUENCE:**

**DETECTED BEFORE EXECUTION**

1. CONTROLLER BUSY.
2. PREVIOUS ERROR.
3. PREVIOUS INEQUALITY TERMINATION.
4. PREVIOUS WARNING ON DESTINATION TAPE.
5. PREVIOUS END OF SOURCE TAPE.
6. BUSY HANDLER OR USE-LOCKOUT.

**DETECTED DURING EXECUTION**

1. EXCEED MEMORY ALLOCATION.
2. ERROR.
3. CRM.
4. INEQUALITY TERMINATION OF WRITE-COPY-READ.
5. WARNING ON DESTINATION TAPE.
6. END OF SOURCE TAPE.

| NORMAL TERMINATION— TAKE NEXT INSTRUCTION IN SEQUENCE | TAPE POSITION AFTER TERMINATION. | TALLY IN @00:43 |
|---|---|---|
| READ AT LEAST ONE RECORD;<br>TERMINATE BECAUSE:<br>N RECORDS READ.<br>CRM AFTER FIRST RECORD.<br>ERROR AFTER FIRST RECORD.<br>EXCEED MEMORY AFTER FIRST RECORD.<br>END OF TAPE AFTER FIRST RECORD. | <br>TO READ THE NEXT RECORD.<br>TO READ THE CRM RECORD.<br>TO READ THE ERROR RECORD.<br>TO READ THE INTERRUPTED RECORD.<br>AT THE TRAILER. | NUMBER OF COMPLETE RECORDS READ CORRECTLY. |
| INDEX OVER AT LEAST ONE RECORD;<br>TERMINATE BECAUSE:<br>N RECORDS INDEXED OVER.<br>CRM AFTER FIRST RECORD.<br>ERROR AFTER FIRST RECORD.<br>END OF TAPE AFTER FIRST RECORD. | <br>TO READ THE NEXT RECORD.<br>TO READ THE CRM RECORD.<br>TO READ THE ERROR RECORD.<br>AT THE TRAILER. | NUMBER OF RECORDS INDEXED OVER. |
| INDEX OVER AT LEAST ONE RECORD;<br>TERMINATE BECAUSE:<br>N RECORDS INDEXED OVER.<br>BEGINNING OF TAPE AFTER FIRST RECORD. | <br>TO READ THE Nth RECORD.<br>AT THE LEADER. | NUMBER OF RECORDS INDEXED OVER. |
| WRITE N RECORDS CORRECTLY.<br>IF V = ~~0~~ 1/2 NO END-OF-TAPE WARNING SIGNAL ENCOUNTERED. | TO WRITE THE NEXT RECORD. | |
| WRITE THE RECORD WITHOUT ERROR.<br>ANY OTHER CIRCUMSTANCE DURING EXECUTION IS TREATED AS OFF-LINE.<br>FOR TERMINATION OF OFF-LINE PORTION, SEE OPPOSITE PAGE. | | |
| EQUALITY TERMINATION OF COPY-READ.<br>THE TERMINATING RECORD IS IN THE MEMORY. | SOURCE TAPE<br>TO READ THE NEXT RECORD.<br>DESTINATION TAPE<br>TO WRITE THE TERMINATING RECORD. | |

| | | ERROR DURING WRITE J 6 | ERROR DURING THE NON-WRITE PORTION OF WRITE-COPY OR WRITE-COPY-READ J 7 | NON-EQUAL TERMINATION J 8 | ERROR DURING WRITE PORTION OF WRITE-COPY OR WRITE-COPY-READ J 9 |
|---|---|---|---|---|---|
| READ | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | | | | |
| | TAPE POSITION AFTER TERMINATION | | | | |
| INDEX FORWARD | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | | | | |
| | TAPE POSITION AFTER TERMINATION | | | | |
| INDEX BACKWARD | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | | | | |
| | TAPE POSITION AFTER TERMINATION | | | | |
| WRITE | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | ERROR IN ANY RECORD DURING SINGLE OR MULTIPLE WRITE. THE FULL N RECORDS ARE ALWAYS WRITTEN. (@00:55)=0 | | | |
| | TAPE POSITION AFTER TERMINATION | TO WRITE THE NEXT RECORD. | | | |
| WRITE-COPY | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | | | | ERROR DURING WRITE PORTION. (@00:55)=0 |
| | TAPE POSITION AFTER TERMINATION | | | | SAME AS BEFORE EXECUTION. |
| WRITE-COPY-READ | EXIT WITHOUT EXECUTION | | ERROR DURING THE COPY PORTION OF A PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | INEQUALITY TERMINATION OF COPY PORTION OF PREVIOUS WRITE-COPY ON SAME CONTROLLER. (@00:55) ≠ 0 | |
| | EXIT AFTER TERMINATION | | ERROR DURING THE COPY OR READ PORTION OF WRITE-COPY-READ. (@00:55) = 0 | INEQUALITY TERMINATION OF COPY-READ. THE TERMINATING RECORD IS IN THE MEMORY. (@00:55) = 0 | ERROR DURING WRITE PORTION. THE FIRST 8 WORDS OF THE RECORD ARE OBLITERATED IN THE MEMORY. (@00:55)=0 |
| | TAPE POSITION AFTER TERMINATION | | SOURCE TAPE TO RE-READ THE ERROR RECORD. DESTINATION TAPE TO WRITE THE ERROR RECORD. | SOURCE TAPE TO READ THE TERMINATING RECORD. DESTINATION TAPE TO WRITE THE TERMINATING RECORD. | SAME AS BEFORE EXECUTION. |

| | |
|---|---|
| READ | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |
| INDEX FORWARD | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |
| INDEX BACKWARD | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |
| WRITE | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |
| WRITE-COPY | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |
| WRITE-COPY-READ | EXIT WITHOUT EXECUTION |
| | EXIT AFTER TERMINATION |
| | TAPE POSITION AFTER TERMINATION |

| TERMINATION OF OFF-LINE PORTION OF WRITE-COPY | MARKER SET | TAPE POSITION AFTER TERMINATION | |
|---|---|---|---|
| | | SOURCE | DESTINATION |
| INEQUALITY TERMINATION. | J 8 | TO READ THE TERMINATING RECORD. | TO WRITE THE TERMINATING RECORD. |
| EQUALITY TERMINATION. | NONE | | |
| CRM. | NONE | TO READ THE CRM RECORD. | TO WRITE THE CRM RECORD. |
| ERROR. | J 7 | TO RE-READ THE ERROR RECORD. | TO WRITE THE ERROR RECORD. |
| WARNING SIGNAL ON DESTINATION TAPE DURING WRITE OR COPY. | J 4 | TO READ THE NEXT RECORD. | AT THE TRAILER. |
| END OF SOURCE TAPE. | J 5 | AT THE TRAILER. | TO WRITE THE NEXT RECORD. |

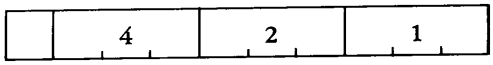
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m̄ | n̄ | ō | p̄ |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | ū | v̄ | w̄ | x̄ |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | CSD syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

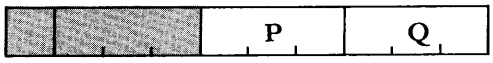
S-values of syllables

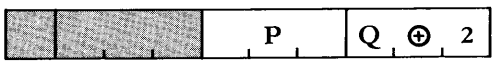
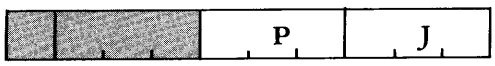
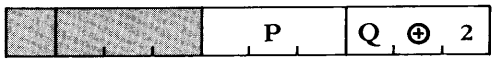
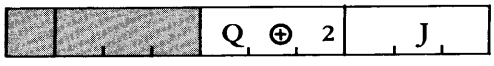



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

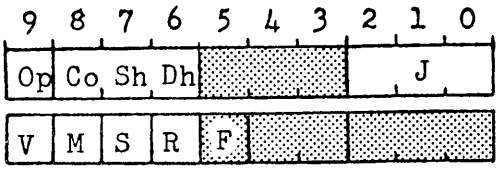
# REWIND

## (AND SET USE LOCKOUT)

This Instruction initiates the Rewind of the tape mounted on any Handler on any Controller. As soon as the Rewind has been started, control of the actual rewinding is transferred to the Handler, and both the Processor and the Controller are free to perform any other operations not involving that Handler.

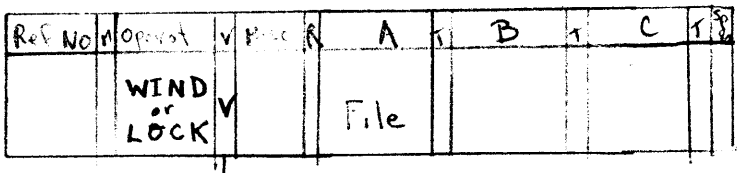
At the programmer's option, this Instruction will also set a Use Lockout on the rewound Handler. This Lockout will prevent access to that Handler by any subsequent Instruction, until the Lockout has been released by the operator.

INSTRUCTION FORMAT:




**Operation: REWIND (& SET USE LOCKOUT)**
(WIND)
(LØCK)

**Operation Code: S**




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co: as modified by (OOR), specifies controller number.

Sh: as modified by (OOR), specifies number of the source-handler used in current processing.

Dh: as modified by (OOR), specifies number of the destination-handler used in current processing.

F: file-number, used by **STEP.** See page IV-Tapes-12.

J: base of address of first word of Jump Table.

V: ~~variation designator~~:

| Abbreviation | REWIND ONLY | V specifies | REWIND & LOCKOUT | Abbreviation |
|---|---|---|---|---|
| WIND:S | 0 | Rewind Sh. Dh irrelevant | 2 | LØCK:S |
| WIND:D | 1 | Rewind Dh. Sh irrelevant | 3 | LØCK:D |

DESCRIPTION OF: REWIND (AND SET USE LOCKOUT)

In order to achieve maximum convenience of tape-management by using Index Registers, this Instruction can cause the rewind operation to be executed on any Handler, whether specified in the Source-Tape position, or in the Destination-Tape position, of the Instruction.

As soon as the REWIND signal has been transmitted to the specified Handler, both the Processor and the Controller are free from the operation, and complete control is transferred to the Handler.

The Instruction may specify that a Use Lockout shall also be placed on the Handler. This lockout will cause a "branch without execution" of any subsequent Instruction which refers to that Handler, and the Processor will at that time select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified, in the Instruction not executed, by J.

The only circumstance which can be programmed for, which will terminate REWIND, is reaching the Leader at the beginning of the tape.

Any circumstance which cannot be programmed for, if detected before the REWIND begins, will cause an error-halt, with appropriate indication in the Console lights. If such a circumstance arises during the REWIND, a "halt marker" will be set in the Controller, as in WRITE-COPY.

Complete details of terminating conditions are shown on page IV-S-4.

## TERMINATING CONDITIONS for REWIND

| CAUSE OF TERMINATION | RESULT | |
|---|---|---|
| Non-existent Controller, or two Controllers have the same number | Processor "Hang-Up" | BEFORE REWIND |
| Non-existent Handler, or two Handlers have the same number | Controller Error-Halt | |
| No Tape, or Broken Tape | Controller Error-Halt | |
| Broken Tape during Rewind | Handler Error-Halt | DURING REWIND |
| Beginning of Tape | Stop Rewind, await next Instruction. If specified, set Use Lockout. | |

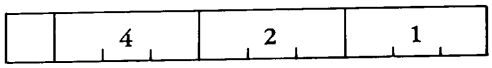
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | CSD syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

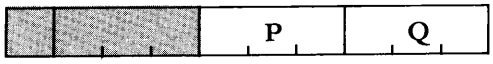
S-values of syllables

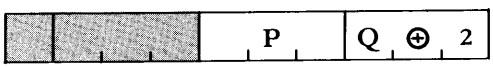
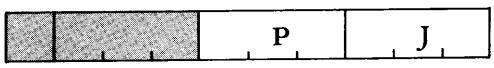
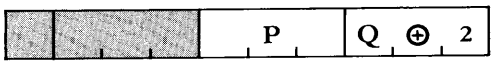
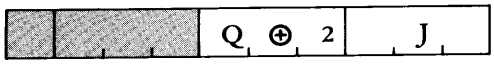



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# R E A D  M A G N E T I C  T A P E

It is most convenient to describe the READ TAPE Instruction as though each of its four major functions were a separate Instruction:

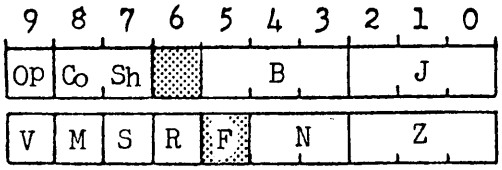


Each of the first three functions will be terminated by a Control Record. Such a record was identified when it was recorded, by storing a Control Record Mark ("v" -- code 11 1101) in character-position 4 of the first word of the record.

All functions will automatically detect the physical end of the tape, and set up appropriate conditions to permit this circumstance to be conveniently handled by the program.

All functions will operate on records of either uniform or varying length.

INSTRUCTION FORMAT:




**Operation: READ COMPLETE RECORDS (REA**

**Operation Code: T (V = 0)**

| Ref No | P | Operation | V | Misc | R | A | I | B | I | C | I | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | READ | | N | | File | | B | | Z | | |
| | | READ | T | – | | File | | – | | – | | |

DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co: as modified by (OOR), specifies controller number.

Sh: as modified by (OOR), specifies source-handler number (handler whose tape is to be read).

B: base of address into which first word of first tape-record is to be read.

F: file-number, used by **STEP.** See page IV-Tapes-12.

N: number of records to be read.
$00 \leq N \leq 99$

Z: number of words of processor memory allocated for information to be read. Z is an address-type number.

J: base of address of first word of Jump Table.

V: variation designator:
V = 0 specifies READ Complete Records.

If N = 0 the Instruction only tests for branch before execution, and does not move tape. (READ:T)

DESCRIPTION OF: READ MAGNETIC TAPE (Read Complete Records)

The Instruction specifies that the next N records (whether of uniform or varying length), on the tape mounted on the specified Handler, shall be read into memory. The first word of the first record is to be stored in Cell B, and successive words read from the tape are to be stored in successive memory cells, B@1, B@2, etc. The operation terminates with the tape repositioned to READ again.

It is important to appreciate the fact that Record-Marks (BRM-ERM) exist only on Magnetic Tape, and are not recorded in Memory. Thus, once information has been read into Memory, the Magnetic Tape records lose their individual identities, as such, and become simply a string of items.

The Processor is free to proceed with the next Instruction as soon as the last character of the last record has been read. Stopping and repositioning the tape after the READ are supervised entirely by the Controller.

A number of circumstances, all of which can be programmed for, will terminate execution of the READ, with fewer than N records actually having been read. Some of these circumstances are defined as "normal" and will cause the Processor to select the next Instruction in the normal sequence; the others will cause the Processor to select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J.

Whether or not the full N records have been read, the Processor will always automatically store in @00:43 a tally of the actual number of complete records correctly read.

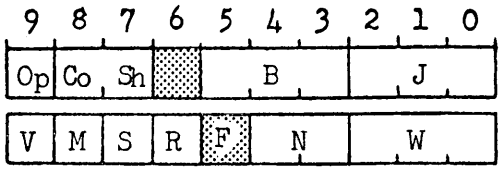
Any circumstance which cannot be programmed for, will cause an error-halt, with appropriate indication in the Console lights.

Complete details of terminating conditions are shown on page IV-T-8.

Operation: **READ PARTIAL RECORDS** (READ:P)
or
**INDEX FORWARD** (INDX:F)
Without Reading

Operation Code: **T (V = 2)**

INSTRUCTION FORMAT:




| Ref No | M | Operation | V | Miscel | F | A | T | B | T | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | READ | P | N | | File | | B | | W | |
| | | READ | T | | | File | | | | | |
| | | INDX | F | N | | File | | — | | — | |

DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co: as modified by (OOR), specifies controller number.

Sh: as modified by (OOR), specifies source-handler number (handler whose tape is to be read).

B: base of address into which first word of first tape-record is to be read.

F: file-number, used by **STEP.** See page IV-Tapes-12.

N: number of records to be read.
00 ≤ N ≤ 99

If N = 0 the Instruction only tests for branch before execution, and does not move tape.

W: number of words to be read from each record (the first W words)
000 ≤ W ≤ 099

If W = 0 the Instruction becomes INDEX FORWARD N Records without reading.
B is irrelevant.

Minimum record is W ⊖ 1 words

J: base of address of first word of Jump Table.

V: variation designator:
V = 2; specifies READ Partial Records or INDEX FORWARD without reading.

DESCRIPTION OF: READ MAGNETIC TAPE (Read Partial Records or Index Forward)

The Instruction specifies that the first W words from each of the next N records (whether of uniform or varying length), on the tape mounted on a specified Handler, shall be read into memory. The first word of the first record is stored in Cell B, and successive words read from the tape are stored in successive memory cells, B@1, B@2, etc. After the first W words of a record have been read, the balance of that record is ignored (although all accuracy-checks are made on the complete record), and the first word of the following record is stored in the next successive memory cell. The operation terminates with the tape repositioned to READ again.

It is important to appreciate the fact that Record-Marks (BRM-ERM) exist only on Magnetic Tape, and are not recorded in Memory. Thus, once information has been read into Memory, the Magnetic Tape records lose their individual identities, as such, and become simply a string of items.

The time required to read the first W words of each record is exactly the same as that required to read each record completely, since the Processor is occupied during the entire length of each record. However, if all the desired information is in the early portion of each record, then reading only a portion of each record will permit more records to be read in a single "gulp", and result in a saving of time because fewer tape accelerations will be required. Furthermore, even though the complete records may be of varying length, the portion of each record containing the desired information may be of uniform length, with a resulting increase in programming convenience.

The Processor is free to proceed with the next Instruction as soon as the last character of the last record has cleared the Read Head. Stopping and repositioning the tape after the READ are supervised entirely by the Controller.

If the Instruction specifies that the first zero words of each record shall be read, the Instruction becomes Index Forward.

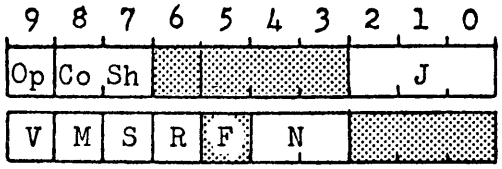
A number of circumstances, all of which can be programmed for, will terminate execution of the READ, with fewer than N records actually having been read, or indexed over. Some of these circumstances are defined as "normal" and will cause the Processor to select the next Instruction in the normal sequence; the others will cause the Processor to select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J.

Whether or not the full N records have been read, or indexed over, the Processor will always automatically store in @00:43 a tally of the actual number of records read, or indexed over.

Any circumstance which cannot be programmed for, will cause an error-halt, with appropriate indication in the Console lights.

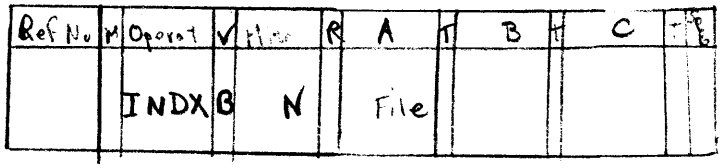
Complete details of terminating conditions are shown on page IV-T-8.

INSTRUCTION FORMAT:




Operator: **INDEX BACKWARD** (INDX:B) **Without Reading**

**Operation Code: T (V = 4)**




DEFINITION

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co: as modified by (OOR), specifies controller number.

Sh: as modified by (OOR), specifies source-handler number (handler whose tape is to be indexed).

F : file-number, used by **STEP.** See page IV-Tapes-12.

N: number of records to be indexed over. $\underline{00 \leq N \leq 99}$

If $\underline{N = 0}$ the Instruction only tests for branch before execution, and does not move tape.

J: base of address of first word of Jump Table.

V: variation designator:
$\underline{V = 4}$; specifies INDEX BACKWARD without reading.

DESCRIPTION OF: READ MAGNETIC TAPE (Index Backward)

The Instruction specifies that the tape mounted on a specified Handler shall be indexed backward N records (whether of uniform or varying length), without reading. The operation terminates with the tape repositioned to READ.

If the Handler encounters the Leader at the beginning of the tape, ~~before the full N records have been indexed over~~ at the start of the Instruction, the Processor will select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J.

Whether or not the full N records have been indexed over, the Processor will always automatically store in @00:43 a tally of the actual number of records indexed over.

Any circumstance which cannot be programmed for, will cause an error-halt with appropriate indication in the Console lights.

Complete details of terminating conditions are shown on page IV-T-8.

# TERMINATING CONDITIONS for READ MAGNETIC TAPE

| APPLIES TO | | | | CAUSE OF TERMINATION | | NEXT INSTRUCTION | AFTER EXECUTION TAPE REPOSITIONED TO READ | TALLY IN (@00:43) | REMARKS |
|---|---|---|---|---|---|---|---|---|---|
| READ COMPLETE | READ PARTIAL | INDEX FORWARD | INDEX BACKWARD | | | | | | |
| x | x | x | x | N complete correct records | | Sequence | Next Record ① | N | |
| x | x | x | | CRM | in 1st record | From J3 | Next Record | 1 | CRM Record *is* read or Indexed over |
| x | x | x | | | after 1st record | Sequence | CRM Record | # Records | CRM Record *not* read or Indexed over |
| x | | | | Exceed Memory Allocation | during 1st record | Processor Error-Halt | Incomplete Record | O | |
| x | | | | | after 1st record | Sequence | Incomplete Record | # Records Complete | |
| x | x | x | x | End of Tape or Beginning of Tape | before 1st record | From J5 | Trailer or Leader | O | |
| x | x | x | x | | after 1st record | Sequence | Trailer or Leader | # Records | |
| x | x | x | | Error | in 1st record | From J2 | Error Record ② | O | Error Record in Memory |
| x | x | x | | | after 1st record | Sequence | Error Record | # Records correct | |
| | x | | | Record Less than W⊕1 Words | | Processor Error-Halt | | | |
| x | x | x | x | Failure of Timing-Check | | Controller Error-Halt | | | |
| x | x | x | x | Failure BRM-ERM Alternation | | Controller Error-Halt | | | |
| x | x | x | x | Non-existent Controller, or two Controllers have the same number | | Processor "Hang-Up" | | | |
| x | x | x | x | Non-existent Handler, or two Handlers have the same number | | Controller Error-Halt | | | |
| x | x | x | x | No Tape, or Broken Tape | | Controller Error-Halt | | | |

① After Indexing Backward, Tape repositioned to read the Nth record.

② After Indexing Forward, Tape repositioned to read the record *following* the Error Record.

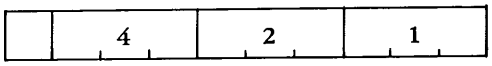
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | CSD syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

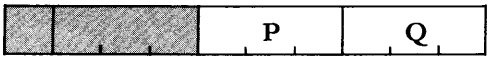
S-values of syllables

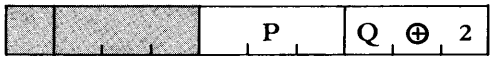
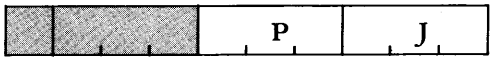
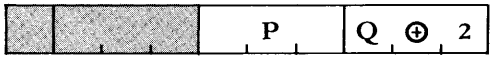
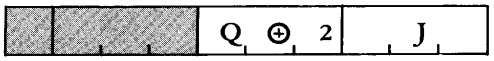



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# W R I T E - C O P Y

# C O P Y

# S E A R C H

The WRITE-COPY Instruction requires the assignment of two Handlers (Source and Destination) on a single Controller. The Instruction performs three functions:

FIRST: 1) Write one record, of a specified length, on the Destination Tape. This function may be omitted at the programmer's option, in which case the Instruction becomes COPY.

THEN: 2) Copy the records from the Source Tape onto the Destination Tape. This function may also be omitted at the programmer's option, in which case the Instruction becomes SEARCH.

AND: 3) Examine each record on the Source Tape to determine whether it satisfies a set of conditions specified in the Instruction. When such a record has been found, execution of the Instruction terminates with the tapes properly repositioned to read that record from the Source Tape, and to write it on the Destination Tape.

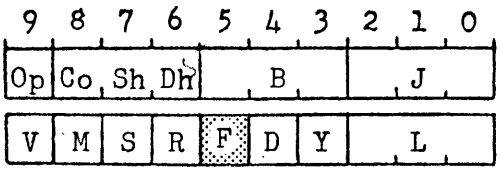
Immediately upon the conclusion of function 1 (the writing of one record), control of the balance of the operation is transferred from the Processor to the Controller as soon as the last character of the record has been verified by the Read-Check Head, and the Processor is free to proceed with the program while the Controller independently performs functions 2 and 3. During this shared time, the Processor may perform, without restriction, any other operation whatever which does not require the use of that Controller; this includes all operations on other Controllers which may be part of the system.

Function 2 operates as a continuous flow of information from the Source Tape to the Destination Tape. Both tapes move at full speed throughout the operation, and since there are no inter-record gaps on either tape, no time is lost between records.

Function 3 may be conducted either as an Equality Search or as a Range Search. Equality Search terminates when a record is found whose key is identical in every character-position with the Search Control Word stored in Memory. Range Search terminates on any record whose key is either greater than, or equal to, the Search Control Word, where "greater than" is defined as following according to the alpha-numeric sorting sequence of the Processor code as shown in Table IV-1. The Search Control Word may be split into two portions, with independent searches of either type specified for each portion. It may also contain "ignore" characters "u" (Code 11 1100), which are "equal to anything".

A record may contain a Control Record Mark ("v" - Code 11 1101) in character-position 4 of the first word, and will then always terminate a SEARCH or COPY. A record may contain a Reject Mark ("w" - Code 11 1110) in character-position 3 of the first word, and will then never terminate a SEARCH or COPY. If both marks are present in the same record, the Reject Mark takes precedence.

INSTRUCTION FORMAT:




Operation: **WRITE-COPY** (WC)
**COPY** (COPY)

Operation Code: **U**

| Rec No | Operat | V | Miscel | R | A | T | B | T | C | TT |
|---|---|---|---|---|---|---|---|---|---|---|
| | WC | | Kind | | File | | B | | L | |
| | 1 | | DIY | | — | | — | | — | |
| | COPY | | Kind | | File | | B | | - | |
| | 1 | | DIY | | — | | — | | — | |
| For Search, see Page C-15 in New Manual | | | | | | | | | | |

DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co,Sh,Dh: as modified by (OOR), specify controller number, source-handler number and destination-handler number.

If Dh = 8 or 9, the Instruction becomes SEARCH. L is irrelevant. [B θ 1] remains as search control words.

B: base of address from which first word of tape-record is to be written.

B θ 1: base of address of search control word. May contain "ignore" character (u).

L: length of record to be written.
~~015~~ 010 ≤ L ≤ 100

Minimum length of written record is ~~15~~ 10 words.
If L = 0, no record is written, and the Instruction becomes COPY.

F: file-number, used by **STEP.** See page IV-Tapes-12.

D: divides search control word into right-hand (RH) and left-hand (LH) portions. D is number of characters in RH portion.

0 ≤ D ≤ 7
No portion of the Search Key may lie wholly within the 30-field of the first word of a record.

Y: relative address, with respect to first word of each record, of Search Key.

0 ≤ Y ≤ 7

J: base of address of first word of Jump Table.

V: variation designator:

| Stop on EITHER Kind | LH Search | RH Search | Stop on BOTH Kind | Marker Set for "Inequality" Termination ? |
|---|---|---|---|---|
| 0 EOE | Equality | Equality | 1 EAE | No |
| 2 EOR | Equality | Range | 3 EAR | If Greater found on RH |
| 4 ROE | Range | Equality | 5 RAE | If Greater found on LH |
| 6 ROR | Range | Range | 7 RAR | If Greater found on LH |

If D=0: V = 0,1,2,3 causes Equality Search
V = 4,5,6,7 causes Range Search

DESCRIPTION OF: WRITE-COPY

Write: A single record, of the specified length, is recorded on the tape mounted on the Destination Handler. The record which is to be recorded is stored in Cell B and in successive memory cells B@1, B@2, etc. As soon as the WRITE is completed and checked, the Processor is freed from the operation, and control is transferred to the Controller. If a zero-length record is specified, the WRITE is omitted.

Copy: Without stopping the Destination Tape, the Source Tape is set in motion, and the information from the Source Tape flows through the Controller to the Destination Tape, and is written there. If the non-existent Handler Number 8 or 9 is specified as the Destination Handler, the WRITE and COPY are omitted.

Search: While the information from the Source Tape is passing through the Controller, a specified Search Key in each record is examined by the Controller and compared with the Search Control stored in Cell B@1. When the Search condition is satisfied, the operation terminates with the Source Tape repositioned to READ the terminating record, and the Destination Tape repositioned to WRITE the terminating record, and the Controller awaits another Instruction from the Processor.

Two circumstances, both of which can be programmed for, may arise during the WRITE: Error during writing
End-of-tape warning signal

In both cases, the operation terminates without initiating the COPY and SEARCH. In case of error during the WRITE, the Processor selects the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J in the terminated WRITE-COPY Instruction. In case of end-of-tape warning during the WRITE, a "branch-marker" is set in the Controller, which will cause the next Instruction addressing that Controller (presumably a READ) to branch without execution, as described below.

Any circumstance arising during the WRITE, which cannot be programmed for, will cause the entire system to halt, with appropriate indication on the console lights.

The COPY and SEARCH will normally terminate on "equality" (as defined on the opposite page). However, a number of other circumstances, all of which can be programmed for, may also terminate the operation; in some of these cases, a "branch-marker" will be set up in the Controller. The next time any Instruction refers to that Controller, this marker will be detected, and that Instruction will branch "without execution"; instead of executing that Instruction, the Processor will at that time select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified, in the aborted Instruction, by J.

Any circumstance arising during the COPY and SEARCH, which cannot be programmed for, will terminate the operation, and set a "halt marker" in the Controller. The next time any Instruction refers to that Controller, this marker will be detected, and cause an error-halt, with appropriate indication in the Console lights.

Complete details of terminating conditions are shown on page IV-U-4.

# TERMINATING CONDITIONS for WRITE-COPY

| CAUSE OF TERMINATION | RESULT | SOURCE TAPE REPOSITIONED TO READ | DESTINATION TAPE REPOSITIONED TO WRITE | |
|---|---|---|---|---|
| Error during WRITE | Next Instruction from J9 | Same as start of Instruction | | DURING ON-LINE WRITE |
| Attempt to record a record of less than ~~15~~ 10, or more than 100, words | Processor Error-Halt | | | |
| All other error conditions listed for WRITE (page IV-W-4) | Same as for WRITE | | | |
| Search Control all "ignores" | Copy is aborted. No branch marker set | First record on the Source-Tape | Record following the written record | DURING OFF-LINE COPY |
| "Equality" termination of COPY ① | No branch marker set | Terminating Record | Terminating Record | |
| "Inequality" termination of COPY ① | Set marker for branch to J8 | Terminating Record | Terminating Record | |
| CRM | No branch marker set | CRM Record | CRM Record | |
| End-of-tape warning on Destination Tape during WRITE | Set marker for branch to J4 | Same as at start | Trailer. The record is written | |
| End-of-tape warning on Destination Tape during COPY | Set marker for branch to J4 | Next Record | Trailer | |
| End of Source Tape during COPY | Set marker for branch to J5 | Trailer | Next Record | |
| Error during COPY | Set marker for branch to J7 | Error Record | Error Record | |
| Failure of Timing-Check | Controller Error-Halt | | | |
| Failure alternation BRM-ERM during COPY | Controller Error-Halt | | | |
| Broken tape during COPY | Controller Error-Halt | | | |
| Attempt to COPY onto Trailer | Controller Error-Halt | | | |

① Defined on page IV-U-2

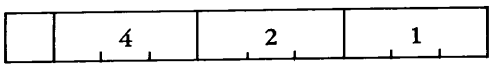
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | CSD syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

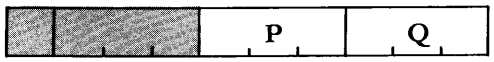
S-values of syllables

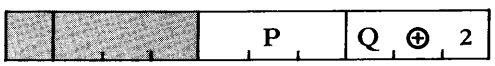
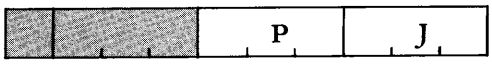
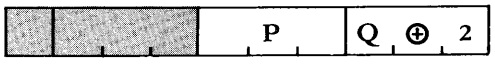
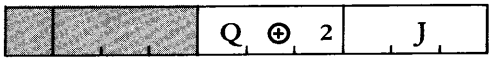



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# W R I T E - C O P Y - R E A D

# C O P Y - R E A D

# S E A R C H - R E A D

The WRITE-COPY-READ Instruction requires the assignment of two Handlers (Source and Destination) on a single Controller. The Instruction performs four functions:

FIRST: 1. Write one record, of a specified length, on the Destination Tape. This function may be omitted at the programmer's option, in which case the Instruction becomes COPY-READ.

THEN: 2. Copy the records from the Source Tape onto the Destination Tape. This function may also be omitted at the programmer's option, in which case the Instruction becomes SEARCH-READ.

AND: 3. Read each record on the Source Tape into the Processor memory, storing the first word of each record in the same cell, so that each record is stored "on top" of the previous records.

AND: 4. Examine each record on the Source Tape to determine whether it satisfies a set of conditions specified in the Instruction. When such a record has been found, execution of the Instruction terminates, with that record stored in the Processor memory, and with the tapes properly repositioned to read the following record from the Source Tape, and to write the terminating record on the Destination Tape.

During this entire operation, the Processor itself is controlling execution of the Instruction.

Function 2 operates as a continuous flow of information from the Source Tape to the Destination Tape. Both tapes move at full speed throughout the operation, and since there are no inter-record gaps on either tape, no time is lost between records.

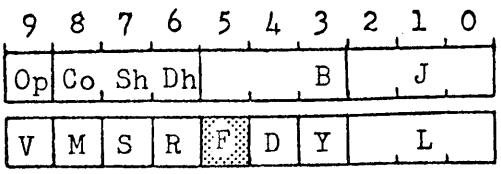
Function 4 may be conducted either as an Equality Search or as a Range Search. Equality Search terminates when a record is found whose key is identical in every character-position with the Search Control Word stored in Memory. Range Search terminates on any record whose key is either greater than, or equal to, the Search Control Word, where "greater than" is defined as following according to the alpha-numeric sorting sequence of the Processor Code, as shown in Table IV-1. The Search Control Word may be split into two portions, with independent searches of either type specified for each portion. It may also contain "ignore" characters, "u" (code 11 1100) which are "equal to anything".

A record may contain a Control Record Mark ("v" - code 11 1101) in character-position 4 of the first word, and will then always terminate a SEARCH or COPY. A record may contain a Reject Mark ("w" - code 11 1110) in character-position 3 of the first word, and will then never terminate a SEARCH or COPY. If both marks are present in the same record, the Reject Mark takes precedence.

INSTRUCTION FORMAT:

Operation: **WRITE-COPY-READ** (WC:R)
**COPY-READ** (COPY:R)

Operation Code: **V**




| Ref No | M | Operat | V | Misce | R | A | T | B | T | C | T |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | WC 1 | R | Kind D\|Y | | File — | | B — | | L — | |
| | | COPY | R | Kind D\|Y | | File — | | B — | | — — | |
| | | For SEARCH-READ, see | | Page C-15 | | | | Next Manual | | | |

DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register 00R.

R: designates 00R as index-register.

Co,Sh,Dh: as modified by (00R), specify controller number, source-handler number and destination-handler number.

If Dh = 8 or 9, the Instruction becomes SEARCH-READ. L is irrelevant. [B θ 1] remains as search control word.

B: base of address:
-- from which first word of tape-record is to be written;
-- into which first 8 words of each record copied are to be read;
-- into which complete terminating record is to be read.

B θ 1: base of address of search control word. May contain "ignore" characters (u).

L: length of record to be written.
~~015~~ 010 ≤ L ≤ 100

Minimum length of written record is ~~15~~ 10 words.
If L = 0, no record is written, and the Instruction becomes COPY-READ.

F: file-number, used by **STEP.** See page IV-Tapes-12.

D: divides search control word into right-hand (RH) and left-hand (LH) portions. D is number of characters in RH portion.

0 ≤ D ≤ 7
No portion of the Search Key may lie wholly within the 30-field of the first word of a record.

Y: relative address, with respect to first word of each record, of Search Key.

0 ≤ Y ≤ 7

J: base of address of first word of Jump Table.

V: variation designator:

| Stop on EITHER Kind | LH Search | RH Search | Stop on BOTH Kind | Marker Set for "Inequality" Termination ? |
|---|---|---|---|---|
| 0 EOR | Equality | Equality | 1 EAE | No |
| 2 EOR | Equality | Range | 3 EAR | If Greater found on RH |
| 4 ROE | Range | Equality | 5 RAE | If Greater found on LH |
| 6 ROR | Range | Range | 7 RAR | If Greater found on LH |

If D=0: V = 0,1,2,3 causes Equality Search
V = 4,5,6,7 causes Range Search

DESCRIPTION OF: WRITE-COPY-READ

Write: A single record, of the specified length, is recorded on the tape mounted on the Destination Handler. The record which is to be recorded is stored in Cell B and in successive cells B@1, B@2, etc. If a zero-length record is specified, the WRITE is omitted.

Copy: Without stopping the Destination Tape, the Source Tape is set in motion, and the information from the Source Tape flows through the Controller to the Destination Tape, and is written there. If the non-existent Handler Number 8 or 9 is specified as the Destination Handler, the WRITE and COPY are omitted.

Read: At the same time, each record from the Source Tape is transmitted to the Processor and stored in memory. The first word of each record is stored in Cell B, with successive words occupying successive memory cells. Therefore, the first record to be read replaces in memory the record which was written on the Destination Tape, and each succeeding record replaces the one before it. When the operation terminates, the last record transmitted will remain in memory.

Search: During the COPY-READ, a specified Search Key in each record is examined by the Controller, and compared with the Search Control stored in Cell B@1. When the Search condition is satisfied, the operation terminates with the Source Tape repositioned to read the record following the terminating record, and the Destination Tape repositioned to write the terminating record, and the Processor goes to the next Instruction.

The COPY-READ-SEARCH will normally terminate on "equality" (as defined on the opposite page). However, a number of other circumstances, all of which can be programmed for, may also terminate the operation; in some of these cases, the Processor will select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J.

Any circumstance which cannot be programmed for, will cause an error-halt, with appropriate indication in the Console lights.

Complete details of terminating conditions are shown on page IV-V-4.

NOTE: If the Search Control, stored in Memory, is the word "uuuuuuuuuu" (all "ignore" characters) then the Instruction becomes WRITE-READ. The record in Memory will be written on the Destination Tape, the COPY will be aborted, and the next record from the Source Tape will be read into Memory, with no repositioning delay between the WRITE and the READ. It is sometimes convenient to process long records one at a time, and in such a case, WRITE-READ will defer all repositioning until after the READ is completed, when it may conveniently be shared with processing.

# TERMINATING CONDITIONS for WRITE-COPY-READ

| CAUSE OF TERMINATION | NEXT INSTRUCTION | SOURCE TAPE REPOSITIONED TO READ | DESTINATION TAPE REPOSITIONED TO WRITE | CONTENTS OF MEMORY |
|---|---|---|---|---|
| "Equality" Termination ① | Sequence | Next Record | Terminating Record | Terminating Record |
| "Inequality" Termination ① | From J8 | Terminating Record | Terminating Record | |
| CRM | From J3 | Next Record | CRM Record | CRM Record |
| Error during WRITE portion | From J9 | Same as at start of Instruction | | First 8 words of record written have been replaced in memory by beginning of first record copied. |
| Error during COPY portion | From J7 | Error Record | Error Record | Error Record |
| Error during READ of Terminating Record | From J7 | Terminating Record | Terminating Record | Terminating Record |
| End of Destination Tape during WRITE | From J4 | No Movement | Trailer | Unchanged |
| End of Destination Tape during COPY | From J4 | Next Record | Trailer | First 8 words of last record copied |
| End of Source Tape during COPY | From J5 | Trailer | Next Record | First 8 words of last record copied |
| Search Control all "ignores". COPY is aborted. | Sequence | Record following the one read into memory | Record following the Written Record | First record on the Source Tape |
| Attempt to record a record of less than 15, or more than 100, words | Processor Error-Halt | | | |
| Write Lockout on Destination Handler | Controller Error-Halt | | | |
| Failure of Timing-Check | Controller Error-Halt | | | |
| Failure BRM-ERM alternation on either Handler | Controller Error-Halt | | | |
| Non-existent Controller, or two Controllers have the same number | Processor "Hang-Up" | | | |
| Non-existent Handler, or two Handlers have the same number | Controller Error-Halt | | | |
| No Tape, or Broken Tape | Controller Error-Halt | | | |
| Try to Write or Copy on Trailer | Controller Error-Halt | | | |

① Defined on Page IV-V-2

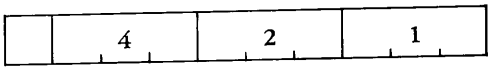
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | CSD syllable modified |
| 2 | B syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

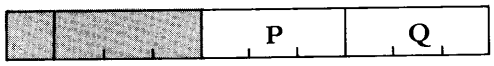
S-values of syllables

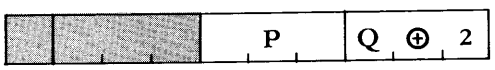
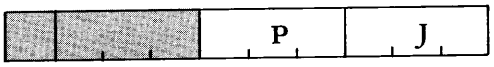
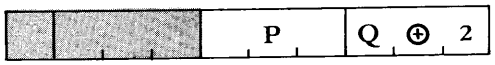
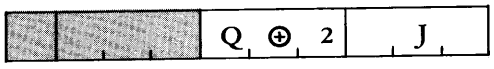



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# WRITE MAGNETIC TAPE

The WRITE TAPE Instruction will cause the recording on Magnetic Tape of either uniform or varying length records.

If uniform-length records are desired, the record-length is specified in the Instruction.

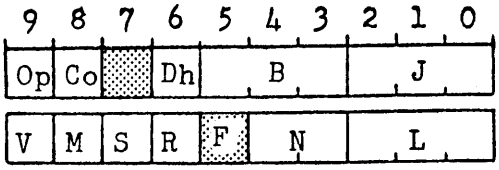
If varying length records are desired, the length of each record must be stored in the 20-field of the first word of each record. The Processor will examine this field, and automatically record the correct number of words in each record.

Any record may be designated as a Control Record by storing a Control Record Mark ("v" -- code 11 1101) in character-position 4 of the first word of the record.

As a safeguard against running off the end of the tape during recording, a reflective spot is placed on the back of the tape some distance from the end, and is detected automatically as an end-of-tape warning during the recording operation. This warning spot is placed far enough from the actual end of the tape to permit recording the entire contents of the memory between the spot and the end of the tape. Therefore, if the warning is detected during execution of any WRITE Instruction, there will always remain sufficient usable tape to permit that Instruction to be completed. Upon detection of this warning spot, the WRITE Instruction is completed, the remaining portion of the tape is erased, and the Processor selects its next Instruction from the appropriate one of the addresses listed in the Jump Table whose location is specified by J. However, the Processor does not wait for the erasing to be performed; it proceeds to the next Instruction as soon as the last record has cleared the Read-Check Head.

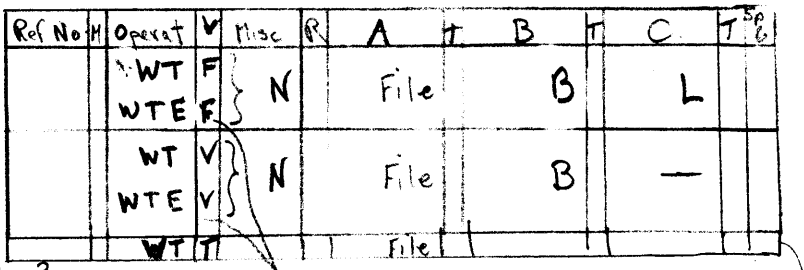
The reflective spot may be attached by any Service Engineer in the field, and places no restriction upon using shorter-length tapes whenever desired.

INSTRUCTION FORMAT:




## Operation: WRITE MAGNETIC TAPE (WT)

Operation Code: W




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: designates syllables for modification by index-register OOR.

R: designates OOR as index-register.

Co: as modified by (OOR), specifies controller number.

Dh: as modified by (OOR), specifies destination-handler number (handler whose tape is to be written).

B: base of address from which first word of first tape-record is to be written.

F: file-number, used by **STEP.** See page IV-Tapes-12.

N: number of records to be written.
$00 \leq N \leq 99$

If N = 0 the Instruction only tests for branch before execution, and does not move tape. (WT:T)

L: FIXED-LENGTH RECORDS:
Length of each record.
$010 \leq L \leq 100$

VARIABLE-LENGTH RECORDS:
Length of each record must be stored in the 20-field of the first word of each record.
L is irrelevant.

J: base of address of first word of Jump Table.

V: variation designator:

| Abbreviation | Fixed Length Records | After Execution | Variable Length Records | Abbreviation |
|---|---|---|---|---|
| WT:F | 0 | Reposition to write | 2 | WT:V |
| WTE:F | 1 | Erase tape to trailer<br>Ignore end-of-tape warning (do not branch) | 3 | WTE:V |

DESCRIPTION OF: WRITE MAGNETIC TAPE

The Instruction specifies that N records shall be recorded on the tape which is mounted on the specified Handler. The first word of the first record is recorded from Cell B, and successive words are recorded from successive memory cells, B01, B02, etc. The records may be of either uniform or varying length. The operation terminates with the tape repositioned to WRITE again.

It is important to appreciate the fact that Record-Marks (BRM-ERM) exist only on Magnetic Tape, and are not recorded in Memory. These marks are automatically generated within the Processor, and are recorded in the proper places on the tape with no attention from the programmer.

The Processor is free to proceed to the next Instruction as soon as the last character of the last record has been verified by the Read-Check Head. Stopping, repositioning and (if appropriate) erasing the tape are supervised entirely by the Controller.

Two circumstances, both of which can be programmed for, may arise during the WRITE:

End-of-tape warning signal
(ignored if V = 1 or 3)
Error during writing

In both cases, the full N records will always be recorded, and the Processor will select the next Instruction from one of the alternate addresses listed in the Jump Table whose location is specified by J.

Any circumstance which cannot be programmed for, including an attempt to write a record of less than 10 words or more than 100 words, will cause an error-halt, with appropriate indication in the Console lights.

Complete details of terminating conditions are shown on page IV-W-4.

# TERMINATING CONDITIONS for WRITE MAGNETIC TAPE

| CAUSE OF TERMINATION | | NEXT INSTRUCTION | AFTER EXECUTION TAPE REPOSITIONED TO WRITE | REMARKS |
|---|---|---|---|---|
| N records written correctly | | Sequence | Next Record ① | |
| End-of-tape Warning | V = ~~0, 1~~ 0, 2 | From J4 | Trailer | The full N records are written; then tape erased to Trailer |
| | V = ~~2, 3~~ 1, 3 | Sequence | | |
| Error | | From J6 | Next Record ① | The full N records are written |
| Attempt to record a record of less than 10, or more than 100, words | | Processor Error-Halt | | |
| Write Lockout on Destination Handler | | Controller Error-Halt | | |
| Failure of Timing-Check | | Controller Error-Halt | | |
| Failure BRM-ERM Alternation | | Controller Error-Halt | | |
| Non-existent Controller, or two Controllers have the same number | | Processor "Hang-Up" | | |
| Non-existent Handler, or two Handlers have the same number | | Controller Error-Halt | | |
| No Tape, or Broken Tape | | Controller Error-Halt | | |
| Attempt to Write on Trailer | | Controller Error-Halt | | |

① Repositioned at Trailer if V = 1 or V = 3

TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

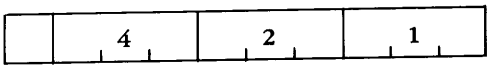
S-value

4 — A syllable modified

1 — J syllable modified

Sum — Determines combination of syllables modified.

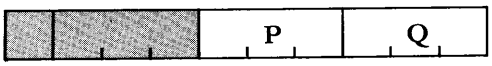
S-values of syllables

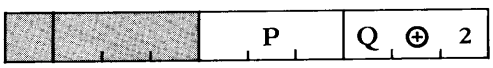
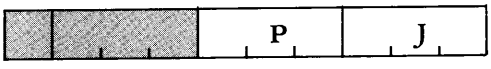
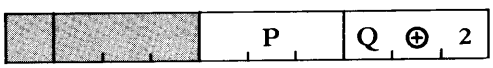
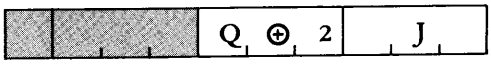



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

# P R I N T

The High-Speed Printer may be operated "on-line" (directly connected) to the Processor with the PRINT Instruction. The contents of twelve consecutive Memory Cells are transmitted to a 12-word magnetic core register in the Printer. These twelve words comprise a previously-edited image of the complete 120-character line which is to be printed. At the same time, one additional character (the "slew" control) is also transmitted to the Printer.

"Slewing" is the term used to refer to the ultra-high-speed movement of paper between successive lines of printing. Slewing speed is independent of the number of lines slewed, and is the same whether one line, or an entire page, is left blank. Therefore, printing time is completely determined by the total number of printed lines, and the total number of blank lines, regardless of their respective distribution on the page.

There are several types of slew control, specified jointly by the slew-control character transmitted with the print-line, and by a loop of punched paper-tape (the slew-control loop) mounted on the Printer:

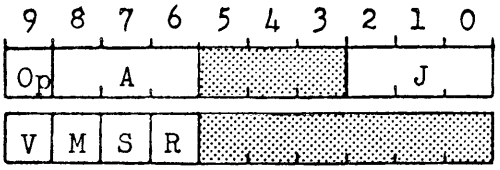
- Slew so as to leave a specified number of blank lines (0 to 15) between successive lines of printing.

- Slew to a specified line on the page, identified by one of 15 possible recognition-codes punched into the slew-control tape.

- Two of the 15 punched codes are assigned special functions, as "skip" code and "stop" code. They may be used as end-of-page and beginning-of-page sentinels.

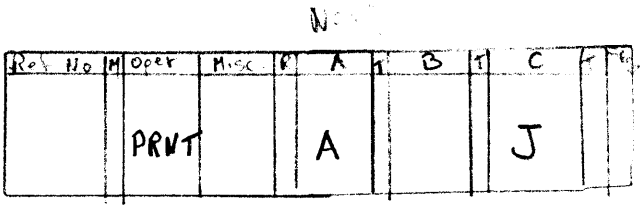
Whenever the Printer encounters a "skip" code in the slew-control loop, all other slew control is cancelled, and the Printer slews until a "stop" code is reached. At the same time, a signal is sent to the Processor, with the result that the next PRINT Instruction will branch without execution, and an alternative Instruction will be executed instead.

INSTRUCTION FORMAT:




Operation: **PRINT** (PRNT)

Operation Code: **X**




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: selects syllables for modification by Index-Register OOR.

R: designates OOR as Index-Register.

A: base of address of first word of 13-word Print Block, set up in Memory:

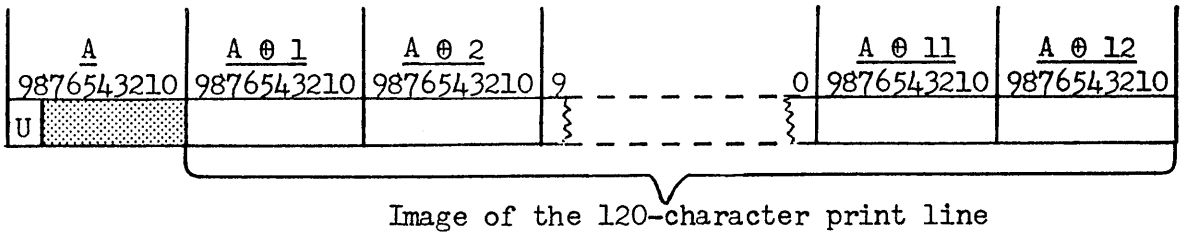



Image of the 120-character print line

U: slew-control. According to Numeric Bits of U:
U positive: Leave 0 to 15 blank lines.
U negative: Slew till one of 15 recognition-codes is found on the slew-control loop.

J: base of address of Instruction to be executed instead of this PRINT Instruction if, after the previous line of print, a "skip" code was encountered on the slew-control loop.

V: variation designator.
Only the sign of V is relevant.

**BINARY VALUE OF NUMERIC BITS**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

The slew-control character [A:99] and the twelve words [A ⊕ 1] through [A ⊕ 12] are transmitted to the High-Speed Printer and stored, respectively, in a 1-character-long Slew-control Register, and in a 120-character-long Print-line Register, in the Printer.

As soon as this transmission has been completed and checked, the Processor is released from the Printer which performs all printing, checking, and slewing functions under its own independent control. The Processor immediately proceeds to execute the next Instruction in the Program.

When the line has been printed and checked, the Printer slews to the next printing position on the page. When slewing has been completed, the Printer is ready to accept another line of print. If a PRINT Instruction is issued while the Printer is still printing or slewing, the Processor will wait until slewing has been completed, and then execute the Instruction.

Slew Control:

If U is positive (its sign-bit position contains a 0-bit) the Printer will leave a number of blank lines (0 to 15) equal to the binary value of the numeric bits of U.

If U is negative (its sign-bit position contains a 1-bit) the Printer will slew until it reaches a punched code in the slew-control loop, exactly corresponding to the numeric bits of U. Punches in the tape-loop correspond to 1-bits, and non-punches correspond to 0-bits.

Punched code 1111 (binary 15) is the "skip" code, and may be used as end-of-page sentinel. Whenever this code is encountered in the slew-control loop, all other slew control is cancelled, and "-14" (slew to the next "stop" code) is automatically placed in the 1-character-long Slew-control Register in the Printer. At the same time, a "branch" marker is set, and the next PRINT Instruction will branch without execution.

Punched code 1110 (binary 14) is the "stop" code, and may be used as beginning-of-page sentinel. Whenever this code is encountered in the slew-control loop, slewing immediately stops, regardless of the type of control which initiated the slewing.

A "stop" code may be encountered under several circumstances:

a) A "skip" code had previously been encountered; this changed the contents of the Printer's Slew-control Register to "-14" (slew to the next "stop" code), and set a "branch"marker. Slewing terminates at the "stop" code. The next PRINT Instruction will detect the "branch" marker, and will not be executed; instead, the Processor will select its next Instruction at that time from the address specified by J in the aborted Instruction.

b) A "stop" code is not preceded by a "skip" code, and the slew-control character is "-14" ("ø" or "w"). Slewing will terminate at the "stop" code.

c) A "stop" code is encountered during any slewing operation other than those described in (a) and (b). Slewing will terminate at the "stop" code, and the Printer will error-halt.

Punched code 0000 (binary zero) corresponds to unpunched, or blank tape, and is not a recognized code on the slew-control loop. If the slew-control character is "-0" ("-" or "*") the Printer will slew, looking for the punched code corresponding to binary zero, which does not exist. Slewing will continue until a "skip" code or a "stop" code is found. This furnishes a convenient way to slew through end-of-page to beginning-of-page.

Overflow Alarm:

Of the 64 characters in the Processor language code, 56 are data-characters which are pertinent to normal printed output. The eight characters m n ø p u v w x are non-data characters; they are merely convenient names for their respective bit-configurations, and are not present on the High-Speed Printer.

If one of these non-data characters should be transmitted for printing, the result depends on the setting of a Switch on the Printer, whose two positions are marked PRINT □ and HALT.

In normal operation a non-data character can occur in the output only as the result of a programming error. The switch will be set to HALT, and the transmittal of a non-data character will cause the Printer to error-halt without printing the line.

However, the eight non-data characters will often be used in programs. For Memory printouts, therefore, the Switch will be set to PRINT □. Any non-data character will be automatically replaced in the Printer by the character "□", and the OVERFLOW ALARM will be set in the Processor before it is released from the Printer.

All of the standard Memory-printout programs test for this condition. If any non-data characters occurred in a line just printed, the program locates them, transforms them into the corresponding capital letters, and prints them on an otherwise blank line with each transformed character directly below the corresponding "□".

---

In order to print "off-line" (from Magnetic Tape, through either the Universal Converter or the Printer Converter, to the Printer), a similar Print Block is recorded on Magnetic Tape for each line of Print. The contents of the 9-character field [A:80] may then contain identifying information to be used in connection with the Searching and Selective Printing abilities of the Converters.

---

Detains of Printer operation are given in Chapter VI - PERIPHERAL EQUIPMENT.

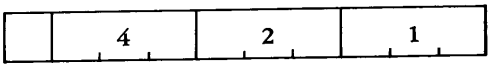
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

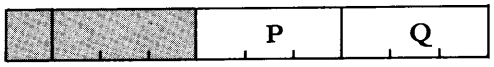
S-values of syllables

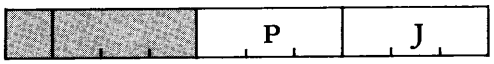
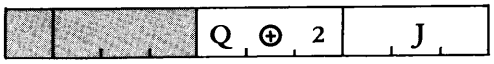



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |   |
| negative |   |

# T Y P E / P U N C H

This Instruction provides facilities for output of information in typewritten, or punched paper-tape form, through the Console Typewriter itself, the Console Typewriter's paper-tape punch, or the High-Speed Paper-Tape Punch.

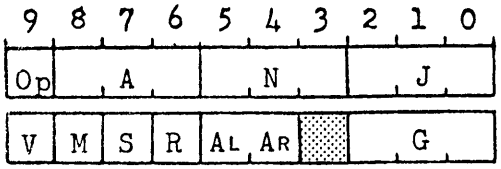
The Console Typewriter may be used to furnish instructions to the operator, and a Log of Operations.

Replies to interrogations may be furnished through the Console Typewriter's paper-tape punch, producing paper tape which can then be printed through another typewriter, if it is preferred that this information should not appear on the Log of Operations.

The High-Speed Paper-Tape Punch may be used for exception-reporting, producing a paper-tape which is to be printed through a typewriter; it may punch information in telegraphic code, for wire transmission; or it may, by punching in the appropriate code, provide paper-tape for conversion to punched cards.

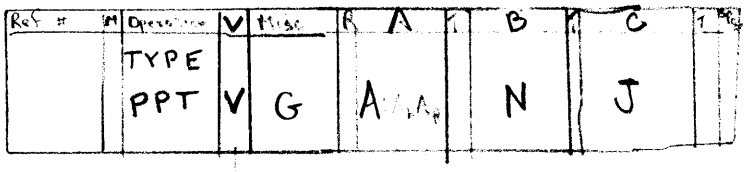
Magnetic Tape may also be transcribed to paper tape "off-line" through the Universal Converter. For details of Punch operation, Converter operation, and punched-tape codes, see Chapter VI - PERIPHERAL EQUIPMENT.

INSTRUCTION FORMAT:




Operation: **TYPE/PUNCH** (TYPE) (PPT)

Operation Code: **Y**




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: selects syllables for modification by Index-Register OOR.

R: designates OOR as Index-Register.

A: base of address of first cell whose contents are to be output.

AL, AR: left-most and right-most character-positions of the partial-word field to be output from each successive cell.

N: as modified by (OOR), specifies the number of cells whose contents are to be output.

$\underline{000 \leq N \leq Z99}$
$\underline{N = 0}$ means "do nothing"

G: amount by which the addresses of successive output cells are to be augmented. (i.e. output the contents of every $G^{\underline{th}}$ cell.)

$\underline{000 \leq G \leq Z99}$
$\underline{G = 0}$ means output [A], N times.

J: base of address of next Instruction, unconditionally.

V: variation designator:

| Abbreviation | With Fixed Format | | With Programmed Format | Abbreviation |
|---|---|---|---|---|
| TYPE: 0 | 0 | Type and Punch on Console Typewriter | 1 | TYPE: 1 |
| TYPE: 2 | 2 | Punch only on Console Typewriter | 3 | TYPE: 3 |
| TYPE: 4 | 4 | Type only on Console Typewriter | 5 | TYPE: 5 |
| PPT: F | 6 | Punch on High-Speed Punch | 7 | PPT: P |

"Tab" after contents of each cell

"w" becomes "Tab"
"x" becomes "Carr Ret"

Starting with the character in position A1, [A] is transmitted, character by character, to the output device designated by the chosen variation. Then [A ⊕ G] is similarly transmitted; then [A ⊕ 2G] is transmitted; and so on until the contents of N cells have been transmitted.

If the code being punched by the output device is a "shifting" code, such as Telegraph Code or the NCR Typewriter Code, the Processor will automatically insert the required "upshift" and "downshift" characters wherever they are required, and transmit them to the output device.

Fixed Format: After the contents of each cell have been transmitted, the Processor automatically transmits the character "Tab" to the output device.

Programmed Format: Whenever the characters "w" (code 11 1110) and "x" (code 11 1111) are encountered in the information to be transmitted to the output device, the Processor substitutes for them the characters "Tab" and "Carriage Return", respectively.

The characters "Tab" and "Carriage Return" do not appear in the Language Code of the Processor, but they are significant characters in many punched paper-tape codes, and the ability to punch these characters from the Processor furnishes valuable format-control facility. Some paper-tape codes use format-control characters other than these, and when punching those codes, the Processor transmits format-control characters appropriate to the specific code. (See Chapter VI - PERIPHERAL EQUIPMENT)

An error-halt will occur if the Processor is called upon to transmit a character which does not exist in the code being punched, or if the self-checking circuits in the output device detect an error.

TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | ø | p |
| 10 | 2 | + | J | K | L | M | N | Ø | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

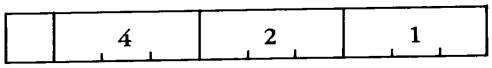
S-value

4 — A syllable modified

1 — J syllable modified

Sum — Determines combination of syllables modified.

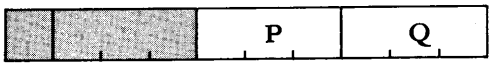
S-values of syllables

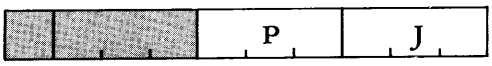
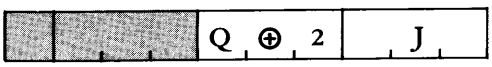



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation |
|---|---|
| positive |   |
| negative |   |

# H A L T

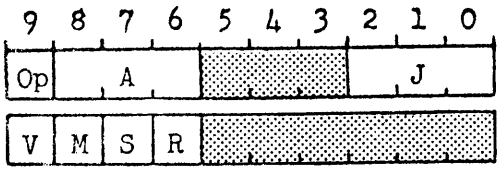
This Instruction stops the Processor, and places it in the REST state.

In the REST state, the Processor will accept information from the Console Typewriter. The Instruction provides for automatic disposition of any information which the operator may enter; when reaching a point in the program at which an operator entry is called for, he need not concern himself with the address in which the information is to be stored.

When the operator presses the START button on the Console, automatic execution of the program is resumed.

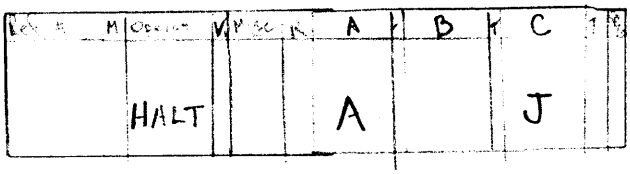
The operator may override the automatic provisions for putaway of this information, and for resumption of the program, by typing other control information, if desired.

INSTRUCTION FORMAT:




**Operation: HALT** (HALT)

**Operation Code: Z (V = O)**




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: selects syllables for modification by Index-Register OOR.

R: designates OOR as Index-Register.

A: base of address of first putaway if information is entered in Memory through the Console Typewriter, after the HALT.

J: base of address of next Instruction, unconditionally.

V: variation designator:
V = 0 specifies HALT.

---

NOTE: After the HALT, the operator may, by entering appropriate control information through the Console Typewriter, over-ride the addresses specified by A and J (as modified by OOR), and specify alternative addresses.

TALLIES IN @00:

When Processor operation is resumed:

(@00:86) contains number of putaways made.
(@00:33) contains number of characters in last putaway.
(@00:33) = 0 means 10 characters.
If no information has been entered, then
both (@00:86) and (@00:33) will be zero.

DESCRIPTION OF: HALT

The Processor is placed in the REST state, and the REST light on the Console is turned on.

The following conditions are set up within the Processor: (*)

Rp: The address A, as modified by (OOR), is placed in the 3-character-long Putaway-address Register (Rp).

¢00: The address J, as modified by (OOR), is placed in ¢00:20.

Ri: The length of the Input Register (Ri) is set at 10 characters, and Ri is cleared.

Any information entered through the Console Typewriter will, therefore, be stored as a sequence of 10-character putaways to Cells [A], [A ⊕ 1], [A ⊕ 2], &c, with tallies in @00:86 and @00:33.

When the START Button on the Console is pressed, or the "Compute Code" key on the Typewriter keyboard is struck, there will be an automatic putaway from Ri (unless Ri is empty), and the Processor will resume execution of the program, selecting its next Instruction from the address stored in ¢00:20.

At any time while the Processor is in the REST state, the operator may change (Rp) or (¢00:20) by means of input-control codes "a" and "c". (*)

NOTE: Punched paper-tape may be read through the Console Typewriter, while the Processor is in the REST state. The function of the paper-tape in such an operation is to store in "frozen" form the typing which caused it to be punched; reading the paper-tape then exactly simulates the typing operator.

Details of Console Typewriter operation are given in Chapter VI - PERIPHERAL EQUIPMENT.

---

(*) See discussion under READ PAPER TAPE. ("The Input Process", page IV-Z6-4)

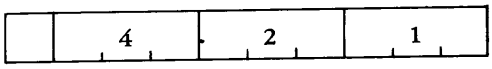
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | – | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

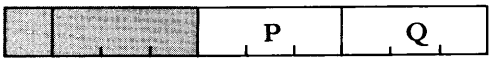
S-values of syllables

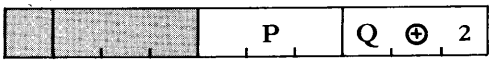
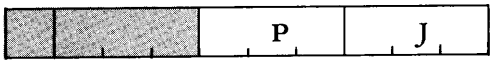
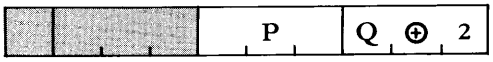
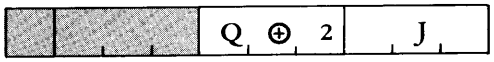



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

J: May be J1, J2, J3

# READ PAPER TAPE

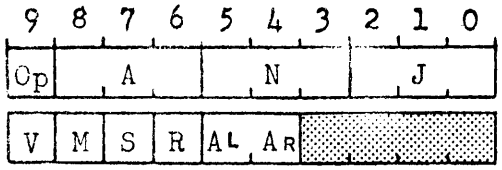
This Instruction permits the reading of Punched Paper-Tape directly into the Processor Memory at extremely high speed.

Paper tape punched at a wide variety of data-origination points may be read into a single Processing System without the need for code-converters, since the High-Speed Paper-Tape Reader will accept tape punched in any three codes.

The ease and accuracy with which punched paper-tape can be prepared (often as a by-product of normal data-recording operations) make it an ideal input medium for many applications. The ability to use Input-control codes with the data, makes Paper Tape extremely convenient in handling unregimented transactions, in which each transaction may not contain all possible types of information-fields, and in which the length of the transactions may vary widely.

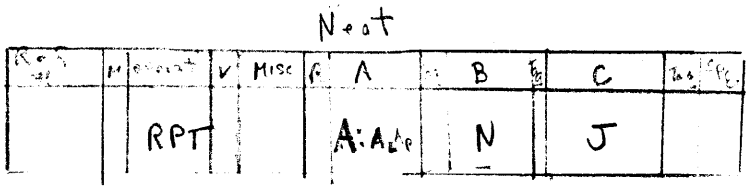
If the volume of input warrants the additional equipment, Paper Tape may also be transcribed "off-line" to Magnetic Tape through the Universal Converter, as described in Chapter VI - PERIPHERAL EQUIPMENT.

INSTRUCTION FORMAT:




## Operation: READ PAPER TAPE (RPT)

Operation Code: Z (V = 6)




DEFINITIONS:

Op: operation code.

M: automonitor level: 0, 1, 2, 3.

S: selects syllables for modification by Index-Register 00R.

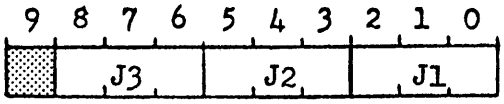
R: designates 00R as Index-Register.

A: base of address of first putaway.

$A_L$, $A_R$: left-most and right-most character-positions of each partial-word putaway.

N: as modified by (00R), specifies the maximum number of putaways to be made. $000 \leq N \leq Z99$
N=0 means 4000, regardless of Memory-Size.

J: base of address of 1-word Jump Table for "branch" termination.




V: variation designator:
V = 6 specifies READ PAPER TAPE.

---

Normal Termination: Processor executes next Instruction in the normal sequence, if operation terminates because of "Compute" code in the paper tape.

Branch Terminations:

Next Instruction from J1: If operation terminates with N putaways.

Next Instruction from J2: If end of paper tape, before N putaways.

Next Instruction from J3: If parity error detected in paper tape. The character x (code 11 1111) is stored instead of the error-character.

NOTE: If "Compute" code, end of tape, or error causes the Nth putaway, the J1 branch is not taken. but rather the appropriate branch is taken.

TALLIES IN @00:

After any termination: (@00:86) contains number of putaways made.

(@00:33) contains number of characters in last putaway. (@00:33) = 0 means 10 characters.

If no information has been read, then both (@00:86) and (@00:33) will be zero.

DESCRIPTION OF: READ PAPER TAPE

This Instruction starts the High-Speed Paper-Tape Reader, and controls the input process while the tape is being read.

The following conditions are set up within the Processor: (*)

Rp: The address A, as modified by (OOR), is placed in the 3-character-long Putaway-address Register (Rp).

¢00: The address of the next Instruction in the normal sequence, remains in ¢00:20. Terminating conditions of the operation may cause this stored address to be replaced by one of the addresses J1, J2, J3.

Ri: The length of the Input Register (Ri) is set to the length of the ALAR field, and Ri is cleared.

Information received from the Paper-Tape Reader will, therefore, be stored in the ALAR field of each of the Cells [A], [A ⊕ 1], [A ⊕ 2], &c, with tallies in @00:86 and @00:33.

The control-codes "a" and "c" punched into the paper-tape will cause the contents of Rp and of ¢00:20 to be changed. However, (¢00:20) will still be replaced by the originally-specified J1, J2 or J3 if the terminating conditions cause a branch.

If the Reader's self-checking circuits detect a reading error, a transmission error, or an illegal configuration punched into the tape, it will error-halt.

---

Paper Tape may be transcribed to Magnetic Tape "off-line" through the Universal Converter. For details of Paper-Tape Reader operation, Converter operation, and paper-tape codes, see Chapter VI - PERIPHERAL EQUIPMENT.

---

(*) See discussion of THE INPUT PROCESS, on succeeding pages.

# THE INPUT PROCESS

The following discussion of Paper-Tape input applies equally to input of information through the Console Typewriter while the Processor is in the REST state, and should be read in connection with the description of HALT.

Whenever the Processor is receiving data from the Paper-Tape Reader, or from the Console Typewriter, the operation is controlled by two working registers:

The Putaway-address Register (Rp) is 3 characters long, and holds the address of the next Memory Cell scheduled to receive information. Immediately after each "putaway" is made, the contents of Rp are automatically augmented by 1 (using MODIFY ADD), so that successive putaways are made to consecutively-numbered Cells.

The Input Register (Ri) varies from 1 to 10 characters in length. Before input begins, the length of Ri is automatically set as specified by the Instruction, and Ri is then "cleared" by storing a zero in every character-position.

As each character is received from the input device, it is stored in the right-most character-position of Ri, and the previous contents of Ri are shifted one character-position to the left, dropping one of the originally-stored zeros. When the Register has been filled, a "putaway" is made automatically; that is, the contents of Ri are transferred to the proper partial-word field of the next Cell designated to receive information. That cell is the one whose address is currently stored in Rp. After the putaway has been made, Ri is cleared, ready to receive more data, (Rp) is augmented by 1, and input continues.

Two tallies are automatically maintained during input, and appear in Cell @00 when input terminates.

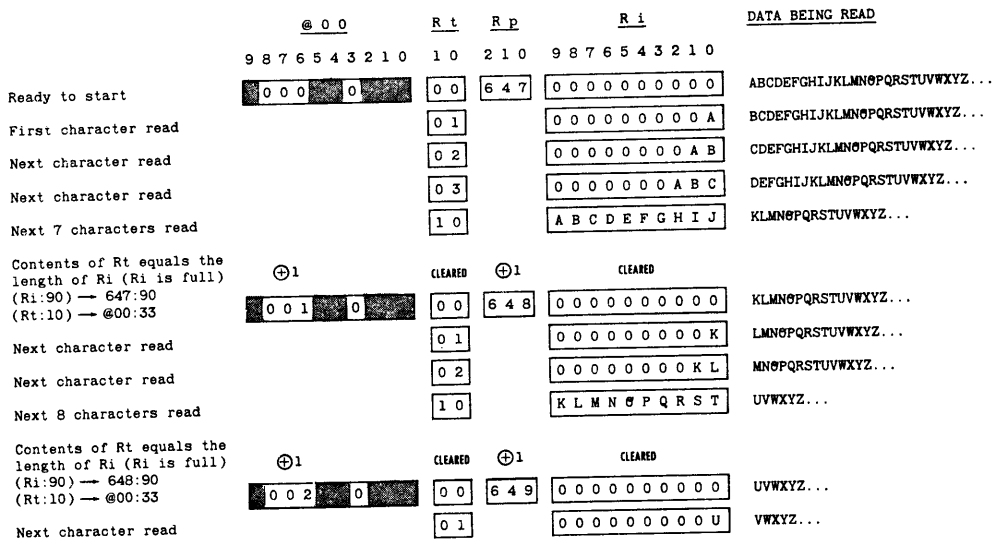
Putaway-count in @00:86. This field is cleared before input starts, and is augmented by 1 when each putaway is made. When input terminates, this field holds the number of Cells into which information was stored during input, as an address-type number.

Character-count in @00:33. This tally is maintained in a 2-character-long working register (Rt), and is transferred, right-justified, to @00:33 after each putaway, replacing the tally generated by the previous putaway.

> Rt is cleared before input starts, and is augmented by 1 as each succeeding character is received into Ri. When (Rt) equals the current length of Ri, then Ri is full. A putaway is made, (Rt) → @00:33, right-justified, and Rt is cleared in readiness for continued input.

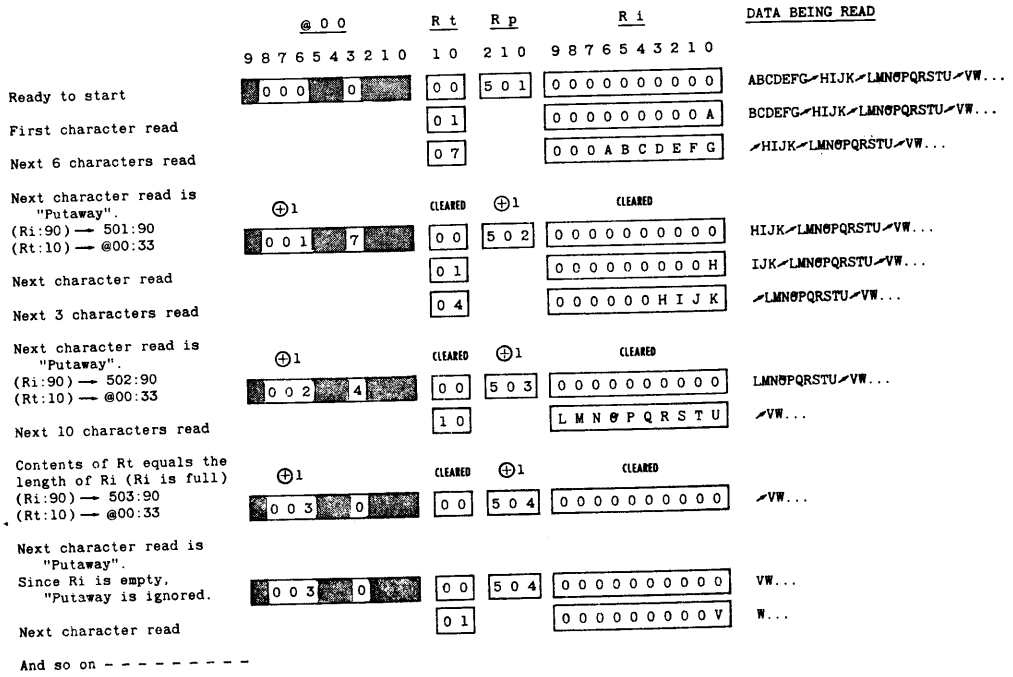
When input terminates, @00:33 holds the number of input characters contained in the last putaway.

The input process may be illustrated by the following diagram, which shows Ri set to its full 10-character length, and successive putaways being made to the 90-field of Cells 647, 648, &c.




Several control codes are available, and affect the input process:

"Putaway" code ("/") will cause the putaway function to be performed, even though Ri may not be completely full. If Ri is empty, the "Putaway" Code will always be ignored. The operation of the "Putaway" Code is shown in the following diagram:



"Compute" Code (">") causes the putaway function to be performed (unless Ri is empty), and terminates input. The Processor then resumes execution of the program, selecting its next Instruction from the address stored in ¢00:20.

Upshift and Downshift: Some code-systems conventionally used with punched paper-tape are "shifting" codes. That is, two different characters in the code may be located on the same key of the tape-punching keyboard, and be represented by the same configuration on the paper-tape. When a tape punched in such a code-system is read by the Processor, many of the configurations must be interpreted as one of two possible characters, according to the shift-position of the tape-punching keyboard at the time that configuration was punched.

Therefore, all devices which punch such a "shifting" code will punch distinctive configurations on the paper-tape whenever the "upshift" and "downshift" keys are struck. The Processor recognizes these configurations as input-control codes, and switches back and forth from one set of decoding circuits to another as these codes are detected.

The following input-control codes are used principally when typing on the Console Typewriter, or when recording programs on punched paper-tape:

"a" ("new putaway address follows"): This code causes the Processor to store the next three input characters in Register Rp. On the typewriter, this code is obtained by striking the "A" key in upper shift (see Typewriter Code, in Chapter VI).

"c" ("new starting address follows"): This code causes the Processor to store the next three input characters in ¢00:20. On the typewriter, this code is obtained by striking the "C" key in upper shift (see Typewriter Code, in Chapter VI).

In detail, "a" and "c" perform the following functions:

- Clear Ri and Rt. Any previous contents of these registers is discarded.
- (Rp) is not disturbed, at this point.
- Set the length of Ri to 3 characters.
- When (Rt) = 3 (i.e. Ri is full), the contents of Ri are transferred to Rp or to ¢00:20, as appropriate.
- The length of Ri is reset to that specified by the HALT or READ PAPER TAPE Instruction. Ri and Rt are cleared.
- If a "putaway" code is received during this operation, the Processor will error-halt.
- If a "compute" code is received during this operation, Ri and Rt are cleared (their contents being discarded), and the Processor resumes execution of the program.
- Note that the tallies in @00:86 and @00:33 are not disturbed by an "a" or "c" operation.

"⊗" ("clear"): This code causes Ri and Rt to be cleared, and their contents discarded; it furnishes a convenient method of correcting typing errors. If "clear" occurs during an "a" or "c" operation, the length of Ri is also reset.

In addition to these control codes, which can be punched into the paper-tape or entered on the Console Typewriter, there are two circumstances, which can arise during the READ PAPER TAPE operation, which the Processor interprets as though it had received control codes.

Parity error: Many of the codes which are used with punched paper-tape are "self-checking" or "parity" codes. That is, every character in such a code must have an even number of holes if "even-parity" is used, or an odd number of holes if "odd-parity" is used.

The Paper-tape Reader has the ability to check such parity codes. If it should detect an error in parity, it will store the character "x" (code 11 1111) in Ri instead of the garbled character, and there will then be an immediate putaway. The input operation terminates, and the Processor resumes execution of the program, selecting its next Instruction from address J3.

That portion of the program which the programmer has stored at J3 has available to it the two tallies in @00:86 and @00:33; it can thus provide for reading the rest of the transaction and, if desired, typing it out on the Console Typewriter, discarding it from the input, and resume the reading of the paper-tape. The operator may then correct the transaction by comparing it with the original data, and arrange for its re-entry later.

End of Tape: The Paper-Tape Reader has the ability to distinguish between broken tape (which causes the Reader to error-halt), and end-of-tape.

Detection of the physical end of the tape causes a putaway (unless Ri is empty), termination of input, and resumption of the program. The Processor selects its next Instruction from address J2.

This ability to identify the end of the tape, and to cause a unique branch in the program, eliminates the necessity for punching a "sign-off" code at the end of the input, and further eliminates the necessity of having the program test every transaction for this sign-off code.

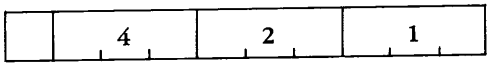
TABLE IV-1: Language Code

| ZONE BITS | NUMERIC VALUE OF ZONE BITS | NUMERIC BITS | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 00 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | @ | ¢ | SPACE | & | • | ' |
| 01 | 1 | — | A | B | C | D | E | F | G | H | I | □ | △ | m | n | o | p |
| 10 | 2 | + | J | K | L | M | N | O | P | Q | R | % | £ | $ | ( | ) | / |
| 11 | 3 | * | # | S | T | U | V | W | X | Y | Z | d | s | u | v | w | x |

TABLE IV-2: Modification of first word of Instruction by Index Register

| S-value | |
|---|---|
| 4 | A syllable modified |
| 2 | N syllable modified |
| 1 | J syllable modified |
| Sum | Determines combination of syllables modified. |

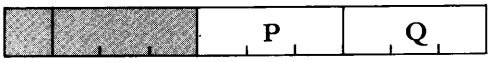
S-values of syllables

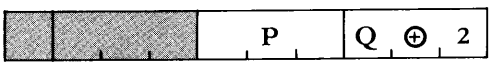
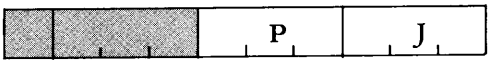
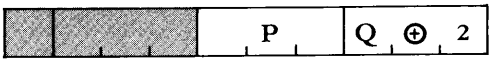
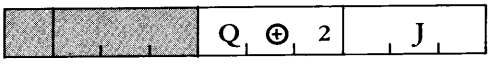



If S = O, no Index Register is used, and R is irrelevant.

TABLE IV-3: Interchange of syllables in ¢00

(¢00) before operation




| V | (¢00) after operation | |
|---|---|---|
| | no branch | branch |
| positive |   |   |
| negative |   |   |

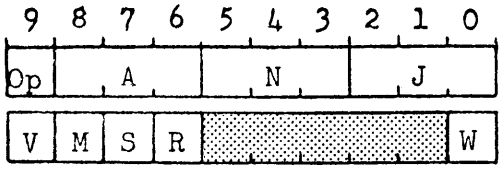
# READ PUNCHED CARDS

This Instruction permits the reading of 80-column cards directly into the Processor Memory at extremely high speed.

As many cards as desired may be read with a single Instruction, and either all, or a portion of, each card may be read.

If the volume of input warrants the additional equipment, cards may also be transcribed "off-line" to Magnetic Tape through the Universal Converter, as described in Chapter VI - PERIPHERAL EQUIPMENT.

INSTRUCTION FORMAT:




Operation: **READ PUNCHED CARDS** (R

Operation Code: *

Neat

| Ref # | Operat | V | Misc | F | A | b | B | C | Tag |
|---|---|---|---|---|---|---|---|---|---|
| | RCD | W | | | A | | N | J | |

DEFINITIONS:

Op: operation code.

M; automonitor level: 0, 1, 2, 3.

S: selects syllables for modification by Index-Register OOR.

R: designates OOR as Index-Register.

A: base of address in which first 10 columns of first card are to be stored.

N: as modified by (OOR), specifies the number of cards to be read.

$\underline{000 \leq N \leq Z99}$
$\underline{N = 0}$ means "do nothing"

W: number of Memory cells to be filled from each card. (i.e. - read the first 10W columns from each card.)

$\underline{0 \leq W \leq 8}$
$\underline{W = 0}$ means pass $\underline{N}$ cards without reading.

J: base of address of next Instruction, if fewer than $\underline{N}$ cards have been read and no more cards remain to be read. (See note)

V: variation designator:
only the sign of V is relevant.

---

NOTE: There is a LAST BATCH Switch on the Card Reader.

Switch in NO position: If hopper empty and fewer than $\underline{N}$ cards read, the Processor will wait, and resume execution of the Instruction when more cards have been put into the hopper, and the "Operate" button on the Card Reader has been pressed.

Switch in YES position: If hopper empty and fewer than $\underline{N}$ cards read, the Instruction will terminate, and the next Instruction will be selected from the address specified by $\underline{J}$.

TALLY IN @00:

After termination, (@00:86) contains number of putaways made.

DESCRIPTION OF: READ PUNCHED CARDS

The Processor starts the High-Speed Punched-Card Reader, which then continues to run until:

a) N cards have been read, after which the Processor executes the next Instruction in the program.

b) The Input Hopper is empty (and/or the Output Hopper is full). Depending on the setting of the LAST BATCH Switch, the Processor either waits for the Reader to be reloaded; or terminates the operation and selects its next Instruction from the address specified by J.

The first 10 columns of the first card are read into Cell [A], with the character punched in column 1 appearing in character-position 9, etc. The next 10 columns into Cell [A ⊕ 1], and so on until W Cells have been filled from the first card. The remainder of the card is passed through the Reader without storing any information in Memory. The first 10 columns of the next card are read into Cell [A ⊕ W], and so on until the operation terminates.

Blank columns on any card are read as "spaces".

The conventional punched-card code has been extended, so that all 64 characters of the Processor's Language Code can be represented in cards.

If the Reader's self-checking circuits detect an error, or an illegal configuration punched into a card, it will error-halt.

---

Punched Cards may be transcribed to Magnetic Tape "off-line" through the Universal Converter. For details of Card-Reader operation, Converter operation, and the extended Punched-Card Code, see Chapter VI - PERIPHERAL EQUIPMENT.

---

NOTE: It is possible that columns 81 and 82 of an "80-column" card might have been punched. These columns are not read.

# AUTOMONITOR

As an aid to Code-Checking, or "Debugging", two techniques are available for selectively monitoring programs:

## MONITORING BY PROGRAM LEVEL

Every Instruction contains a digit (designated as M) which specifies the level of monitoring to which that Instruction will be subject. Four levels (0, 1, 2, 3) are available, and after a program has been written, the programmer will assign a monitoring level to each Instruction in the program. Major check-points will be assigned level 3, lesser check-points levels 2 and 1, down to the individual Instructions between check-points, which will be assigned level 0.

A Program Level Monitor Switch is provided on the Console, with positions OFF, 0, 1, 2, 3. When this switch is in the OFF position, no Program-Level Monitoring will take place. When the switch is set to one of the numbered positions, every Instruction whose Monitor Digit has a numeric value (modulo 4) greater than or equal to the switch-setting, will be monitored.

## MONITORING BY ADDRESS LOOKUP

A set of four Address Selection Switches is provided on the Console, and these switches may be set to any address in the Processor Memory. For the operator's convenience, these switches portray an address in its 4-digit numeric form, rather than as a condensed Address-Type Number. There is also an Address Monitor Switch with OFF and ON positions. When this switch is OFF, it will show a blue light, and no Address-Lookup Monitoring will take place. When the switch is ON, it will show a yellow

light, and the Processor will monitor every Instruction which requires a lookup to the Memory location specified in the Address Selection Switches. An Address-Lookup occurs whenever the Processor selects:

* Any word of the Instruction itself,
* An Index-Register to modify the Instruction,
* An operand,
* A location for storage of a result.

Program-Level Monitoring and Address-Lookup Monitoring may be carried on independently or simultaneously, as the operator may choose. If the M-digit of any Instruction contains a negative character (a character with a 1-bit in its sign-bit position), that Instruction will never be monitored by either method.

It is important to provide complete flexibility, not only in the amount of information furnished during monitoring, and in the format in which the information will be presented, but also in the medium (console typewriter, punched paper tape, line printer, or magnetic tape) in which the information will be recorded. In order to achieve this flexibility, the actual compilation and editing of the information is performed by means of a Monitoring Program, which the operator stores in the Processor Memory before beginning to monitor. The Monitoring Programs may range from a simple tracing procedure which records only the address of each Instruction monitored, to a complete printout of each Instruction in its coded form and also as modified by an Index-Register, the contents of the Index-Register, the operands, and the results.

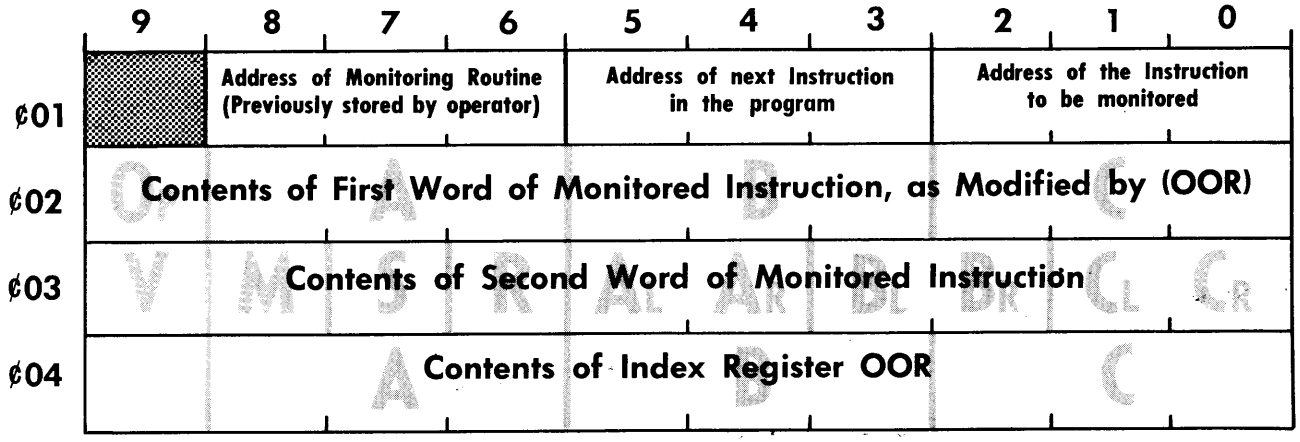
The operator stores the Monitoring Program in the Memory, and plants the address of the first Instruction of the Monitoring Program in

the A-syllable of Cell ¢01 (that is, in ¢1:86). He sets:

* The Program Level Monitor Switch, or the Address Monitor Switch, or both;
* The address Selection Switches, if they are to be used;
* A Monitor Run-Halt Switch, whose function will be described shortly.

The operator then runs, in the usual manner, the main program, which is to be monitored.

After executing each Instruction in the main program, and just before selecting the next Instruction, the Processor examines the two Monitoring Switches, to see if either of them is ON; if so, the Instruction just completed is reviewed to see if it qualifies for monitoring. If the Instruction does not qualify for monitoring, the main program is resumed and the next Instruction is selected for execution in the usual manner. If the Instruction does qualify for Monitoring, the Processor automatically stores certain information about the Instruction, in the special Memory locations ¢01 through ¢04, and then inspects the setting of the RUN-HALT Switch. The stored information is in the following form:



It must be observed that the information in Cells ¢01 through ¢04 is compiled after execution of the monitored Instruction has been completed. If the Instruction changes itself (e.g. in Example 9 of "Normal" DISTRIBUTE, page IV-J-12), it will be displayed as it appears after execution.

After the information has been stored, the next step depends on the setting of the Monitor Run-Halt Switch.

If the Monitor Run-Halt Switch is set to RUN, it shows a yellow light; the Processor plants the address of the Monitoring Program (¢01:86) into ¢00:20 (the Sequence-Control Register), and then proceeds to execute the Monitoring Program. The last Instruction of the Monitoring Program must be written so that it plants the address of the next Instruction in the main program (¢01:53) into ¢00:20, so that the Processor immediately resumes the main program.

If the Monitor Run-Halt Switch is set to HALT, it shows a blue light; the Processor halts immediately after storing the information in Cells ¢01 through ¢04, and turns on the Console MONITOR HALT light. The address of the Instruction being monitored is displayed as four arabic digits in the Address Indication Lights on the Console. The operator may then change the settings of the Program Level Monitor Switch, the Address Selection Switches and/or the Monitor Run-Halt Switch, if he desires, and then press the START button, whereupon the Processor will enter the Monitoring Program, and then continue with the main program, exactly as though the Monitor Run-Halt Switch had been set to RUN.

Or the operator may press the RESET button on the Console, which puts the Processor into the "Rest" state, and causes the REST light on the Console to be turned on in place of the MONITOR HALT light. The operator now may manually determine the next step, as described in Chapter VI, under CONSOLE OPERATION.

Some of the options available to the operator at this point are:

Start. If the START button on the Console is pressed, the Processor will still enter the Monitoring Program, just as though the RESET button had not been pressed.

Printout. If the PRINTOUT button on the Console is pressed, the Processor will type on the Console Typewriter the contents of whatever Memory location is specified by the Address Selection Switches. Before Printout, of course, these switches may be changed to designate any desired Memory location.

Manual Input. By typing on the Console Typewriter, the operator may change the contents of any Memory locations he chooses.

New "Start" Address. The operator may designate any Memory location as the address of the next Instruction, before he presses the START button to resume processing. He may wish to execute a special Monitoring Program for this part of the main program; to monitor a different part of the main program; or to change to some other operation altogether.

# OPERATOR'S CONSOLE CONTROL PANEL

On all Control Panels within the NCR 304 System, the display lights have been restricted to three colors with, generally, the following significance:

| YELLOW | BLUE | RED |
|---|---|---|
| On | Off | Alarm |
| Ready | Not Ready | |
| Operating | Stop | |
| Normal | Non-Normal | |

Whenever it appears that the distinction between the two states of a switch, as indicated by Yellow and Blue lights, is in any way ambiguous, the switch markings on the panel are underlined in the appropriate colors.

## OPTION SWITCHES

0 - 9: These are examined by the TEST OPTION SWITCH Instruction.

Each switch contains two colored lights:
ON: Yellow
OFF: Blue

## OPERATING SWITCHES

RESET: Pressing this button removes the Processor from the MONITOR HALT, OVERFLOW HALT, or ERROR HALT state, and places it in the REST state.

If any peripheral unit goes into Error Halt, the Processor will "hang up". The operator should press the RESET button and bring the Processor to REST before resetting the peripheral unit.

If the RESET button is pressed during Processor operation, it will place the Processor in the REST state, but it will not be possible to resume the program from that point.

The RESET button is inoperative when the Processor is in the HALT or the TEST PANEL state.

START: When the Processor is in the REST state, pressing this button causes it to start executing the program.

When the Processor is in the HALT state, pressing this button causes it to execute the next Instruction in the program, and then halt again (single-step operation). At each halt, the ADDRESS INDICATION lights display the address of the next Instruction in the program.

When the Processor is in the MONITOR HALT state, pressing this button causes it to start executing the Monitoring Program.

The START button is inoperative in all other circumstances.

The START button shows a yellow light while the Processor is executing a program.

PRINTOUT: This button is operative only when the Processor is in the REST or the HALT state. At that time, pressing this button will cause the contents of one Memory Cell to be typed automatically on the Console Typewriter; that Cell is the one whose address is specified in the ADDRESS SELECTION switches.

OPERATOR HALT: Pressing this button will cause the Processor to halt the program as soon as the current Instruction is completed, and enter the HALT state, at which time this button will show a red light.

While the Processor is in the HALT state, the only controls which can be operated are:

Power ON-OFF buttons, which are not operative at any other time.

The START button, which will (in this circumstance) cause single-step operation.

Pressing the HALT button the second time will turn off the red light, and place the Processor in the REST state.

After an OPERATOR HALT, the ADDRESS INDICATION lights show the address of the next Instruction in the program.

# STATUS LIGHTS

REST: Blue light. The Processor enters this state:

- As the result of a HALT Instruction in the program. The ADDRESS INDICATION lights display the address specified by A of the Instruction.
- As the result of pressing the RESET button.
- As the result of pressing the HALT button a second time.

MONITOR HALT: Blue light. While the Automonitor is being used, with the MONITOR RUN-HALT switch set to HALT, an Instruction has been selected for monitoring.

The ADDRESS INDICATION lights display the address of the Instruction to be monitored.

If the START button is pressed, the Processor will enter the Monitoring Program. If RESET, and then START, are pressed, the result will be the same.

OVERFLOW HALT: Red light. An Instruction has set the Overflow Alarm, and the next Instruction is not TEST OVERFLOW.

The ADDRESS INDICATION lights give no useful information.

The operator may PRINTOUT the contents of Cell ¢00, and subtract 2 from the contents of its C-syllable to obtain the address of the Instruction which detected the overflow.

ERROR HALT: Red light. One of several things has happened:

Either the Processor is attempting to execute a non-existent Operation Code, in which case the ADDRESS INDICATION lights display the address of the impossible Instruction;

Or the programmer has made a serious error, such as specifying an impossible Memory address, or a non-existent Controller;

Or the Processor's self-checking circuits have indicated that some component of the system has failed.

In the last two cases, the ADDRESS INDICATION lights will display the address of the last Memory lookup. Detailed information as to the source of the error is shown on the Engineer's Test Panel.

TEST PANEL: Blue light. The Processor is under the control of the Engineer's Test Panel, and is not available to the operator.

## CLASSIFICATION LIGHTS

These are two rows of yellow lights, which flicker as each Instruction is executed, indicating the classification to which the current Instruction belongs; after any halt, the classification of the last Instruction executed is shown. These classifications correspond to the color-coding of the index tabs in Chapter IV:

| | |
|---|---|
| Yellow tabs | Arithmetic |
| Blue tabs | Decision |
| White tabs | Input (HALT, READ PAPER TAPE, READ CARDS) |
| | Output (PRINT, TYPE/PUNCH) |
| Dark Green tabs | File On-Line (READ TAPE, WRITE TAPE, WRITE-COPY-READ) |
| | File Off-Line (WRITE-COPY, REWIND) |
| Light Green tabs | Editing |
| Pink tabs | Data-Handling |

## ADDRESS INDICATION LIGHTS

These lights display, as 4 arabic numerals, every address-lookup performed by the Processor. Addresses of the "Special" Cells are shown as 3 characters, using the first, third and fourth lights.

The significance of the address displayed in these lights after any halt is stated under the description of each of the STATUS LIGHTS.

## ADDRESS SELECTION SWITCHES

The operator may set into these switches the address of the Memory Cell whose contents he wishes typed out on the Console Typewriter, when he presses the PRINTOUT button.

These switches are also used to designate a specific Memory Cell for the ADDRESS MONITOR.

An address is set into these switches as a 4-digit number; addresses of the "Special" Cells are set as 3 characters, using the first, third and fourth switches.

## MONITOR SWITCHES

(See Chapter V)

ADDRESS MONITOR: This switch contains two colored lights:
ON: Yellow
OFF: Blue

PROGRAM LEVEL MONITOR: The setting of this switch determines the level of Monitoring, in connection with the M-digit of each Instruction in the program.

MONITOR RUN-HALT: This switch contains two colored lights:
RUN: Yellow
HALT: Blue

The words RUN and HALT on the panel are underlined in yellow and blue, respectively, to remind the operator of the significance of the colored lights.

If no monitoring is being performed, this button shows no light.

## POWER SWITCHES

All Power ON-OFF switches are interlocked with the HALT button, and are operative only when the Processor is in the HALT state.

Power in peripheral units may be turned on only after Processor power is on. When Processor power is turned off, all peripheral units are turned off also.

PROCESSOR POWER: As soon as the ON button is pressed, the Processor enters an automatic warmup-test cycle and the lights in the OPTION SWITCHES are turned on.

When the Processor has completed its self-testing operation, the red light in the HALT button goes on, indicating that the Processor is in the HALT state.

Pressing the HALT button the second time (the first time occurred just before the power was turned off) places the Processor in the REST state, and processing may begin.

PERIPHERAL UNIT POWER:

Once the Processor power has been turned on, power may then be turned on in the peripheral units. Each unit is represented on the Console by:

an ON button;
an OFF button;
a BUSY light.

When power is first turned on at each unit, its ON button will show a blue light (not ready), while that unit goes through its own warmup-test cycle.

When the unit has completed its self-testing operation, its ON button will show a yellow light (ready).

Whenever the unit is actually in use, its yellow BUSY light will be turned on.

If the unit should, for any reason, go into an Error-Halt, its ON button will show a red light.

INPUT-OUTPUT CODE INDICATORS:

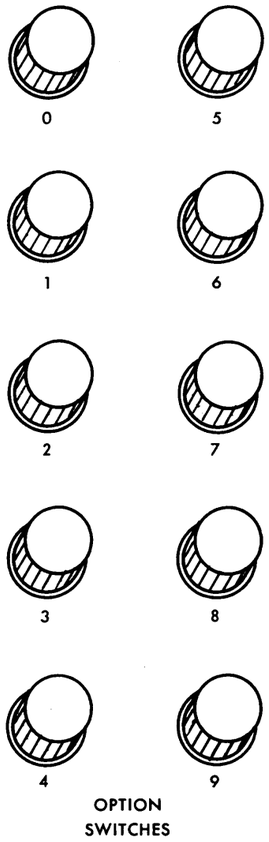
Selection among three codes on the Paper Tape Reader, and between two codes on the Paper Tape Punch, is made manually at each unit.

In order that the Console operator may conveniently monitor these selections, they are displayed as yellow lights next to the power buttons for the respective units.

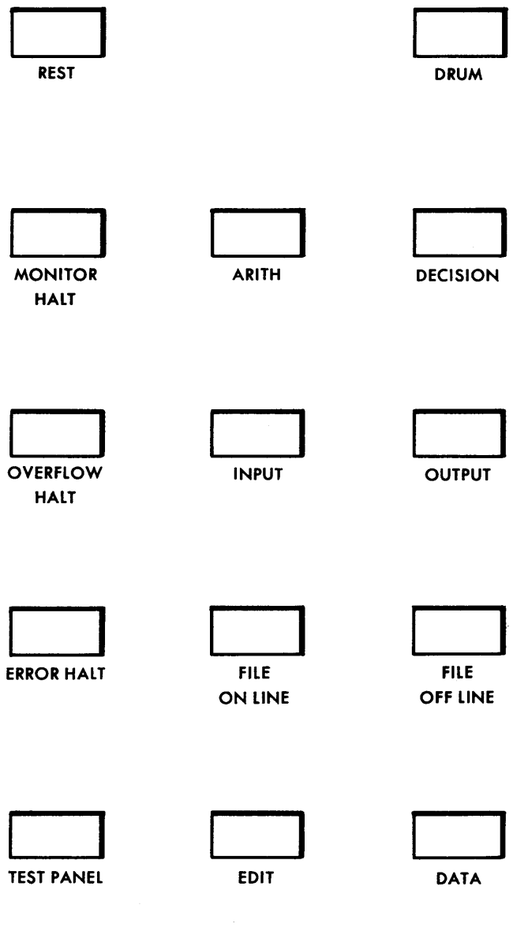
# OPERATOR'S CONSOLE
# CONTROL PANEL

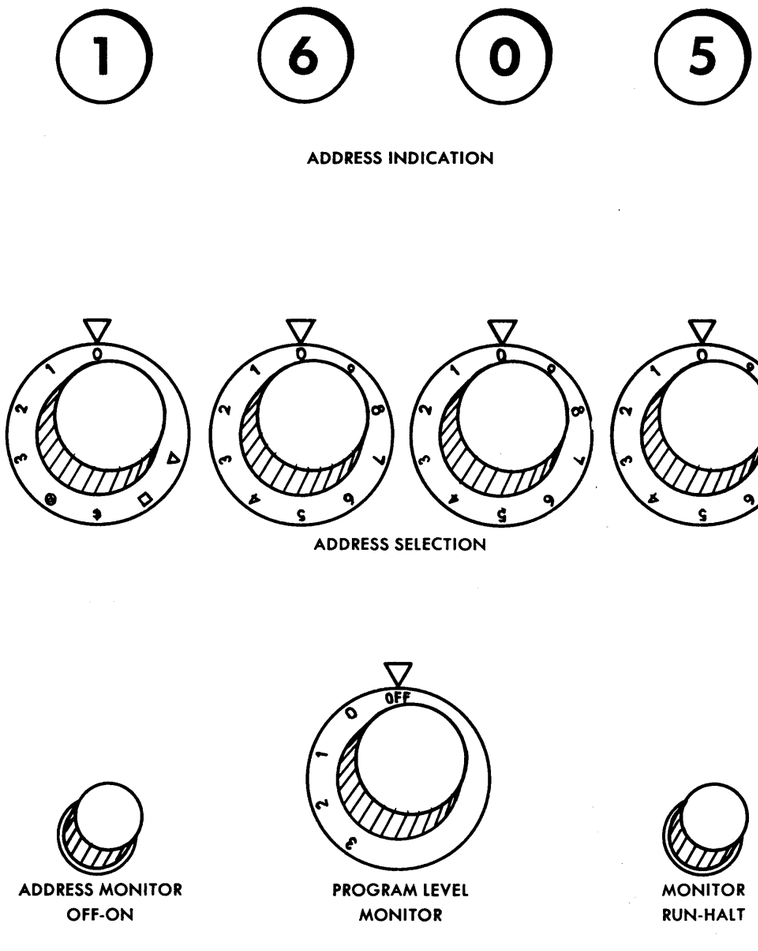
National

ELECTRONIC DATA
PROCESSOR (304)



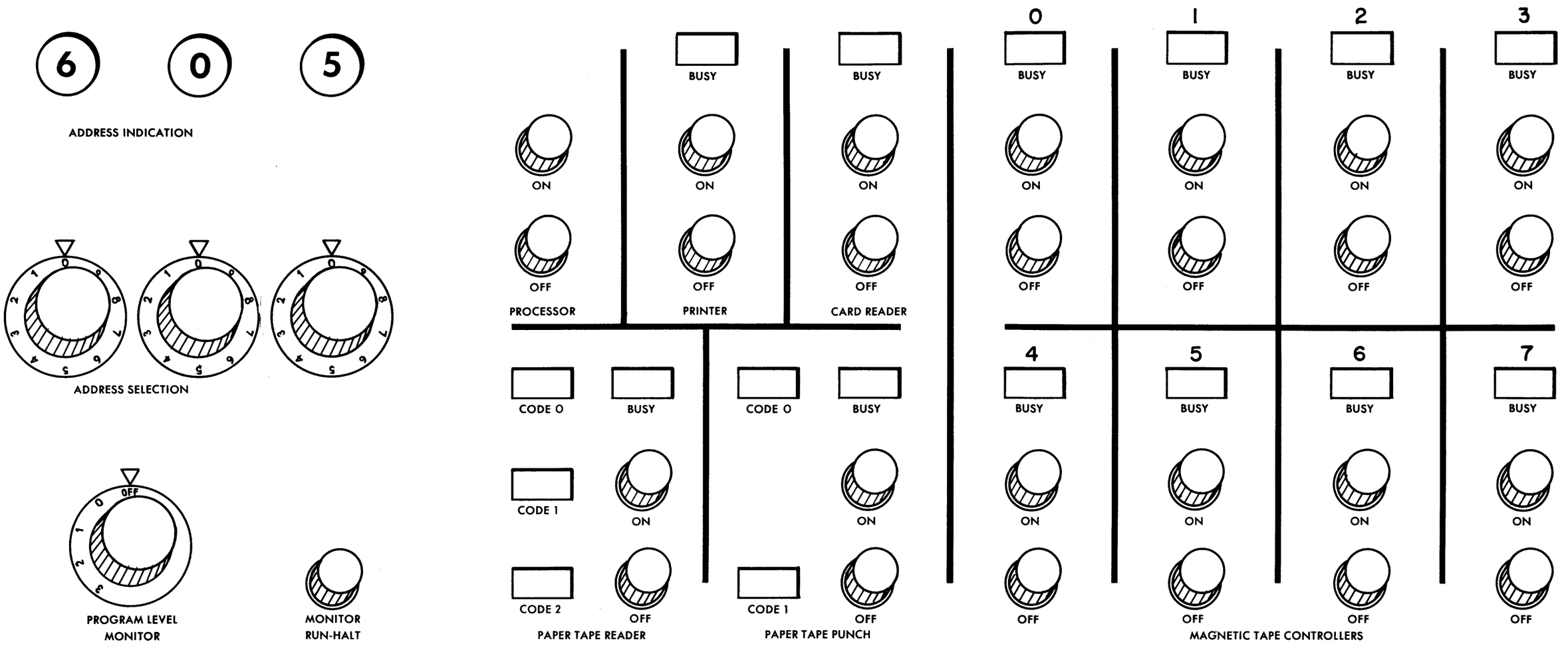

OPTION
SWITCHES


# OPERATOR'S CONSO

## CONTROL PANEL




# OPERATOR'S CONSOLE

## CONTROL PANEL

# PAPER TAPE INPUT-OUTPUT

## NCR TYPEWRITER CODE

(Odd-Parity)

| TYPEWRITER KEY | | CHANNELS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LOWER SHIFT | UPPER SHIFT | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| 0 | ) | | ● | | | ● | • | | ● | |
| 1 | 1 | | | | | | • | | | ● |
| 2 | £ | | | | | | • | | ● | |
| 3 | # | | ● | | | | • | | ● | ● |
| 4 | $ | | | | | | • | ● | | |
| 5 | % | | ● | | | | • | ● | | ● |
| 6 | / | | ● | | | | • | ● | ● | |
| 7 | & | | | | | | • | ● | ● | ● |
| 8 | * | | | | | ● | • | | | |
| 9 | ( | | ● | | | ● | • | | | ● |
| A | a ① | | ● | | ● | | • | | | ● |
| B | B | | ● | | ● | | • | | ● | |
| C | c ① | | | | ● | | • | | ● | ● |
| D | d | | ● | | ● | | • | ● | | |
| E | E | | | | ● | | • | ● | | ● |
| F | F | | | | ● | | • | ● | ● | |
| G | G | | ● | | ● | | • | ● | ● | ● |
| H | H | | ● | | ● | ● | • | | | |
| I | I | | | | ● | ● | • | | | ● |
| J | J | | ● | ● | | | • | | | ● |
| K | K | | ● | ● | | | • | | ● | |
| L | L | | | ● | | | • | | ● | ● |
| M | m | | ● | ● | | | • | ● | | |
| N | n | | | ● | | | • | ● | | ● |
| O | o | | | ● | | | • | ● | ● | |
| P | p | | ● | ● | | | • | ● | ● | ● |

| TYPEWRITER KEY | | CHANNELS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LOWER SHIFT | UPPER SHIFT | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| Q | Q | | ● | ● | | ● | • | | | |
| R | R | | | ● | | ● | • | | | ● |
| S | s | | | ● | ● | | • | | ● | |
| T | T | | ● | ● | ● | | • | | ● | ● |
| U | u | | | ● | ● | | • | ● | | |
| V | v | | ● | ● | ● | | • | ● | | ● |
| W | w | | ● | ● | ● | | • | ● | ● | |
| X | x | | | ● | ● | | • | ● | ● | ● |
| Y | Y | | | ● | ● | ● | • | | | |
| Z | Z | | ● | ● | ● | ● | • | | | ● |
| @ | + | | | ● | | | • | | | |
| ¢ | □ | | ● | ● | ● | | • | | | |
| - | △ | | | | ● | | • | | | |
| . | , | | | | | ● | • | ● | ● | |
| SPACE | SPACE | | ● | | | ● | • | ● | | |
| PUTAWAY ↗ ① | PUTAWAY ↙ ① | | | | | ● | • | ● | | ● |
| COMPUTE > ① | STOP ⊐ ② | | ● | ● | | ● | • | ● | | ● |
| CLEAR ⊠ ① | CLEAR ⊠ ① | | ● | | | ● | • | ● | ● | ● |
| UP SHIFT ① | UP SHIFT ① | | ● | | ● | ● | • | | ● | ● |
| DOWN SHIFT ① | DOWN SHIFT ① | | | | ● | ● | • | | ● | |
| CODE DELETE ② | CODE DELETE ② | | ● | ● | ● | ● | • | ● | ● | ● |
| TAPE FEED ② | TAPE FEED ② | | ● | | | | • | | | |
| ② CARR RET ③ | ② CARR RET ③ | | | | ● | ● | • | ● | ● | ● |
| ② TAB ③ | ② TAB ③ | | ● | | ● | ● | • | ● | ● | |
| BACKSPACE ② | | DOES NOT PUNCH | | | | | | | | |

**NOTES**

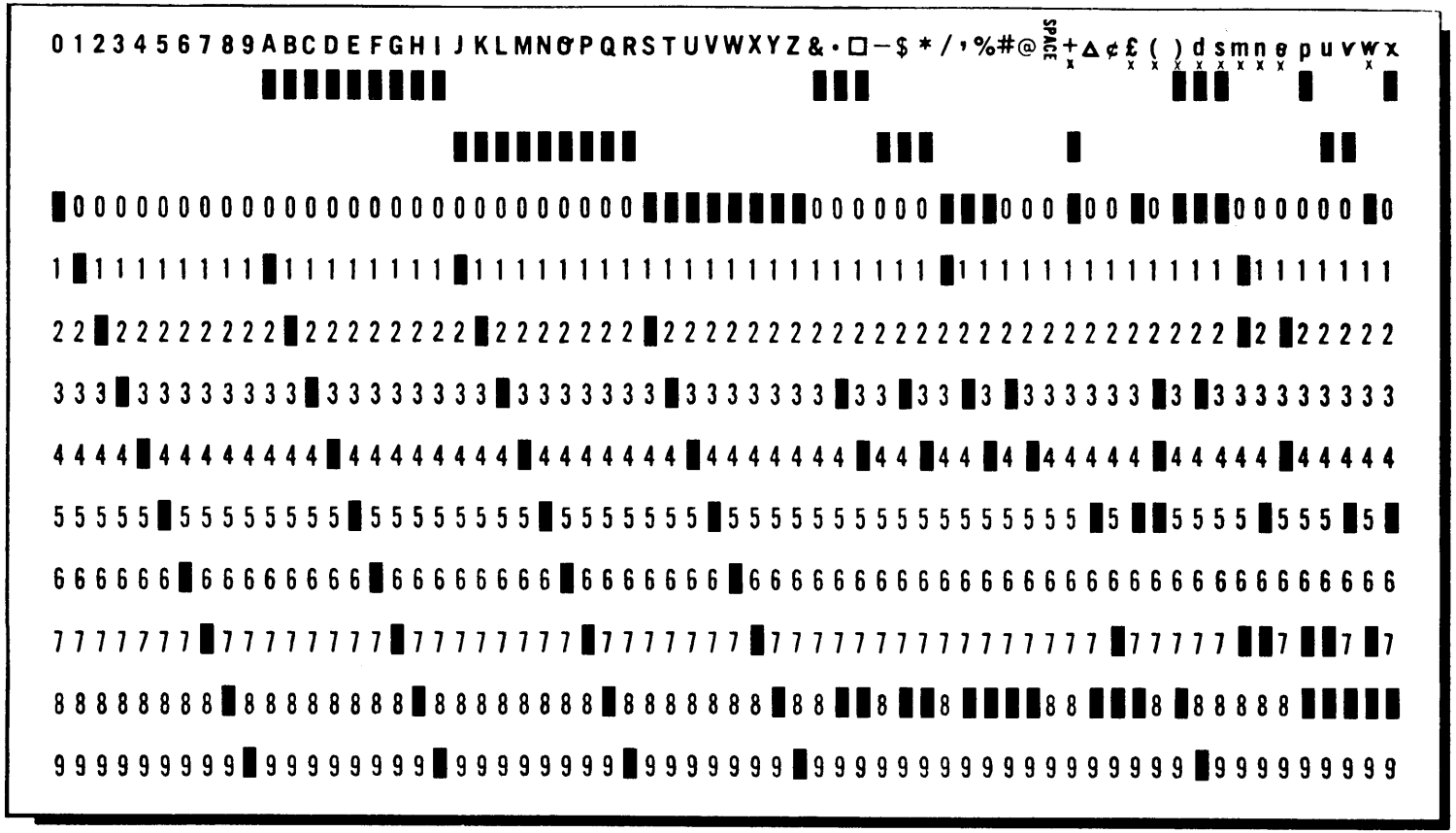
① Input Controls. These exercise control functions during input, and store no information in memory.

② Typewriter controls. These have no function during input, and are ignored.

③ When punching with Programmed Format from Processor or Converter
- w (code 11 1110) becomes TAB
- x (code 11 1111) becomes CARR RET

# PUNCHED CARD CODE

## INPUT-OUTPUT




X WILL NOT PRINT ON I.B.M. 026 PRINTING KEY PUNCH

NOTE: The characters shown above, following "space", can be punched through a modification of the I.B.M. 026 Printing Key Punch, whereby these characters have been assigned to the number-shift position on the following keys:

| KEY | R E Q T A F D S G N H Z C V W X |
|---|---|
| CHARACTER | + Δ ¢ £ ( ) d s m n e p u v w x |

The I.B.M. 523 Summary Punch can also be modified to permit punching these characters from the Universal Converter.

# PAPER TAPE INPUT-OUTPUT

## I. B. M. 046-047 8-CHANNEL CODE

(Odd Parity)

NCR 7 channel

| CHARACTERS | | CHANNELS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 046-047 | 304 | 8 EL | 7 X | 6 0 | 5 √ | 4 8 | • • | 3 4 | 2 2 | 1 1 | |
| 0 | 0 | | | ● | | | • | | | | @ |
| 1 | 1 | | | | | | • | | | ● | 1 |
| 2 | 2 | | | | | | • | | ● | | 2 |
| 3 | 3 | | | | ● | | • | | ● | ● | C |
| 4 | 4 | | | | | | • | ● | | | 4 |
| 5 | 5 | | | | ● | | • | ● | | ● | E |
| 6 | 6 | | | | ● | | • | ● | ● | | F |
| 7 | 7 | | | | | | • | ● | ● | ● | 7 |
| 8 | 8 | | | | | ● | • | | | | 8 |
| 9 | 9 | | | | ● | ● | • | | | ● | I |
| A | A | | ● | ● | | | • | | | ● | J |
| B | B | | ● | ● | | | • | | ● | | K |
| C | C | | ● | ● | ● | | • | | ● | ● | T |
| D | D | | ● | ● | | | • | ● | | | M |
| E | E | | ● | ● | ● | | • | ● | | ● | V |
| F | F | | ● | ● | ● | | • | ● | ● | | W |
| G | G | | ● | ● | | | • | ● | ● | ● | P |
| H | H | | ● | ● | | ● | • | | | | Q |
| I | I | | ● | ● | ● | ● | • | | | ● | Z |
| J | J | | ● | | ● | | • | | | ● | A |
| K | K | | ● | | ● | | • | | ● | | B |
| L | L | | ● | | | | • | | ● | ● | 3 |
| M | M | | ● | | ● | | • | ● | | | D |
| N | N | | ● | | | | • | ● | | ● | 5 |
| O | O | | ● | | | | • | ● | ● | | 6 |
| P | P | | ● | | ● | | • | ● | ● | ● | G |
| Q | Q | | ● | | ● | ● | • | | | | H |
| R | R | | ● | | | ● | • | | | ● | 9 |
| S | S | | | ● | ● | | • | | ● | | S |
| T | T | | | ● | | | • | | ● | ● | L |
| U | U | | | ● | ● | | • | ● | | | U |
| V | V | | | ● | | | • | ● | | ● | N |
| W | W | | | ● | | | • | ● | ● | | O |

| CHARACTERS | | |
|---|---|---|
| 046-047 | 304 | |
| X | X | |
| Y | Y | |
| Z | Z | |
| . | . | |
| , | , | |
| SPACE | SPACE | |
| - | - | |
| & | & | |
| @ | @ | |
| P. I. 1 | ¢ | |
| □ | □ | |
| P. I. 2 | △ | |
| P. I. 3 | + | |
| % | % | |
| P. I. 4 | £ | |
| $ | $ | |
| E.C. 1 | ( | |
| E.C. 2 | ) | |
| / | / | |
| * | * | |
| # | # | |
| P. I. 5 | d | |
| P. I. 6 | s | |
| S.P. 1 | PUTAWAY ① | ↗ |
| S.P. 2 | COMPUTE ① | > |
| P. I. 7 | x | |
| END OF LINE | ② | ← |
| ERROR | ② | |
| SKIP | W | |
| TAPE FEED CODE DELETE | ② | |
| CORRECTION | ② | |
| CR | ② | |

| CHANNELS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 8 EL | 7 X | 6 0 | 5 √ | 4 8 | • • | 3 4 | 2 2 | 1 1 | |
| | | ● | ● | | • | ● | ● | ● | X |
| | | ● | ● | ● | • | | | | Y |
| | | ● | | ● | • | | | ● | R |
| | ● | ● | | ● | • | | ● | ● | 9 L |
| | | ● | ● | ● | • | | ● | ● | C Y |
| | | | ● | | • | | | | — |
| | ● | | | | • | | | | Tape Feed |
| | ● | ● | ● | | • | | | | ¢ |
| | | | ● | ● | • | ● | | | 48 — |
| | | | ● | ● | • | | ● | | 28 — |
| | ● | ● | ● | ● | • | ● | | | QU |
| | ● | | | ● | • | | ● | | 0 |
| | | ● | | ● | • | | ● | | 28@ |
| | | ● | | ● | • | ● | | | 48@ |
| | | ● | ● | ● | • | ● | | ● | E Y |
| | ● | | ● | ● | • | | ● | ● | Up Shift |
| | | | | ● | • | ● | ● | | • |
| | | ● | | ● | • | ● | ● | ● | 7 R |
| | | ● | ● | | • | | | ● | 1 - @ |
| | ● | | | ● | • | ● | | | space |
| | | | | ● | • | | ● | ● | 128 |
| | ● | ● | | ● | • | ● | | ● | > |
| | ● | | ● | ● | • | ● | | ● | 9 E |
| | ● | ● | ● | ● | • | | ● | | Q S |
| | ● | ● | | ● | • | ● | ● | | 6 Q |
| | | | | ● | • | ● | | ● | ↗ |
| ● | | | | | • | | | | |
| | ● | | | ● | • | ● | ● | ● | ⊠ |
| | | ● | ● | ● | • | ● | ● | | O Y |
| | ● | ● | ● | ● | • | ● | | ● | Code Delete |
| | | | ● | ● | • | ● | ● | ● | Carr Ret. |
| | ● | | ● | ● | • | ● | ● | | Tab |

**NOTES**

① Input Controls. These exercise control functions during input, and store no information in memory.

② These have no function during input, and are ignored.

# PAPER TAPE INPUT-OUTPUT

## TELEGRAPHIC CODE

(Non-Parity)

| CHARACTERS | |
|---|---|
| TELEGRAPH LETTER SHIFT | 304 |
| A | A |
| B | B |
| C | C |
| D | D |
| E | E |
| F | F |
| G | G |
| H | H |
| I | I |
| J | J |
| K | K |
| L | L |
| M | M |
| N | N |
| O | O |
| P | P |

| CHARACTERS | |
|---|---|
| TELEGRAPH FIGURE SHIFT | 304 |
| - | - |
| ? | COMPUTE ① |
| : | + ④ |
| $ | $ |
| 3 | 3 |
| ! | PUTAWAY ① |
| & | & |
| £ | # ④ |
| 8 | 8 |
| ' | * ④ |
| ( | ( |
| ) | ) |
| . | . |
| , | , |
| 9 | 9 |
| 0 | 0 |

| CHANNELS | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 4 | • | 3 | 2 | 1 | NCR |
| 1 | 2 | • | 3 | 4 | 5 | TEL. |
| ● | ● | • | | | | |
| ● | | • | | ● | ● | |
| | ● | • | ● | ● | | |
| ● | | • | | ● | | |
| ● | | • | | | | |
| ● | | • | ● | ● | | |
| | ● | • | | ● | ● | |
| | | • | ● | | ● | |
| | ● | • | ● | | | |
| ● | ● | • | | ● | | |
| ● | ● | • | ● | ● | | |
| | | • | ● | ● | ● | |
| | | • | ● | ● | | |
| | | • | | ● | ● | |
| | ● | • | ● | | ● | |

| CHARACTERS | |
|---|---|
| TELEGRAPH LETTER SHIFT | 304 |
| Q | Q |
| R | R |
| S | S |
| T | T |
| U | U |
| V | V |
| W | W |
| X | X |
| Y | Y |
| Z | Z |
| SPACE | SPACE |
| FIGURE SHIFT | FIGURE SHIFT ① |
| LETTER SHIFT | LETTER SHIFT ① |
| TAPE FEED | ② |
| LINE FEED | ② ③ |
| CARR RET | ② ③ |

| CHARACTERS | |
|---|---|
| TELEGRAPH FIGURE SHIFT | 304 |
| 1 | 1 |
| 4 | 4 |
| BELL | @ ④ |
| 5 | 5 |
| 7 | 7 |
| ; | □ ④ |
| 2 | 2 |
| / | / |
| 6 | 6 |
| " | △ ④ |
| SPACE | SPACE |
| FIGURE SHIFT | FIGURE SHIFT ① |
| LETTER SHIFT | LETTER SHIFT ① |
| TAPE FEED | ② |
| LINE FEED | ② ③ |
| CARR RET | ② ③ |

| CHANNELS | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 4 | • | 3 | 2 | 1 | NCR |
| 1 | 2 | • | 3 | 4 | 5 | TEL. |
| ● | ● | • | ● | | ● | |
| | ● | • | | ● | | |
| ● | | • | ● | | | |
| | | • | | | ● | |
| ● | ● | • | ● | | | |
| | ● | • | ● | ● | ● | |
| ● | ● | • | | | ● | |
| ● | | • | ● | ● | ● | |
| ● | | • | ● | | ● | |
| ● | | • | | | ● | |
| | | • | ● | | | |
| ● | ● | • | | ● | ● | |
| ● | ● | • | ● | ● | ● | |
| | | • | | | | |
| | ● | • | | | | |
| | | • | | ● | | |

### NOTES

① Input Controls. These exercise control functions during input, and store no information in memory.
The character "?" is interpreted as COMPUTE CODE.
The character "!" is interpreted as PUTAWAY.

② These have no function during input, and are ignored.

③ When punching with Programmed Format from Processor or Converter
w (code 11 1110) becomes LINE FEED
x (code 11 1111) becomes CARR RET

④ Every character in telegraphic code has been assigned a character in 304 code. As a result, the 304 can communicate with any non-standard keyboard.

# PAPER TAPE INPUT

**KIMBALL PUNCHED-TAG CODE (Non-Parity)**

**DENNISON PUNCHED-TAG CODE (Even-Parity)**

| CHARACTERS | KIMBALL CHANNELS 7 6 5 4 • 3 2 1 | DENNISON CHANNELS 7 6 5 4 • 3 2 1 |
|---|---|---|
| + ③ | 7 • 4 | √ 7 • 4 |
| 1 | • 1 | • 1 |
| 2 | • 2 | • 2 |
| 3 | • 3 | • 3 |
| 4 | • 4 | • 4 |
| 5 | • 4 1 | • 4 1 |
| 6 | • 4 2 | • 4 2 |
| 7 | 7 • | 7 • 1 |
| 8 | 7 • 1 | 7 • 2 |
| 9 | 7 • 2 | 7 • 4 |
| PUTAWAY ① | 5 • 2 | 5 7 • 4 1 |
| COMPUTE ① | 5 • 4 | 5 7 • 4 2 |
| TAPE FEED ② | • | • |
| V | 5 • 3 | 7 • 3 2 |
| — | 5 • 3 1 | 5 • 4 2 1 |
| A | 7 • 4 2 | 5 • 3 2 1 ④ |
| B | • 4 2 1 | 6 5 • 3 |
| C | 7 • 3 1 | 6 • 3 1 |
| D | 7 • 4 2 1 | 6 • 4 2 |
| E | 7 • 4 1 | 6 • 3 2 |
| F | 5 7 • ④ | 6 • 4 ④ |

When using NCR 461-3 Recorder (single program, fixed diode-board) the following codes are read out from positions:

| |
|---|
| S – 2 |
| S – 3 |
| S – 9 |
| S – 0 |
| S – 1 |

## NOTES

① Input Controls. These exercise control functions during input, and store no information in memory.

② These have no function during input, and are ignored.

③ Whenever the parent machine reads a "zero", the character "+" will be punched into the paper tape. This will enable the programmer to verify, in every case, that a field of the proper length has been read into the Processor. The first Arithmetic operation performed on each field within the Processor will, of course, replace every "+" with a "zero".

④ All of Kimball Code, and Dennison Code down through the letter "A", may be used for 5-channel telegraphic transmission from outlying data-origination points to the central Data-Processing Center.

# PAPER TAPE INPUT

## CASH-CONTROL CODES

**Non-Parity**
**May be punched with**
**Kimball Tags**

**Even-Parity**
**May be punched with**
**Dennison Tags**

| CHARACTERS | CHANNELS (Non-Parity) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| + ③ | | | | | ● | • | ● | | |
| 1 | ● | | | | | • | | | |
| 2 | | | | | | • | | ● | |
| 3 | ● | | | | | • | | ● | |
| 4 | | | | | | • | ● | | |
| 5 | ● | | | | | • | ● | | |
| 6 | | | | | | • | ● | ● | |
| 7 | | | | | ● | • | | | |
| 8 | ● | | | | ● | • | | | |
| 9 | | | | | ● | • | | ● | |
| PUTAWAY ① | | | | ● | | • | | ● | |
| COMPUTE ① | | | | ● | | • | ● | | |
| TAPE FEED ② | | | | | | • | | | |
| V | | | | ● | | • | | | |
| — | ● | | | ● | | • | | | |
| A | | | | | ● | • | ● | ● | |
| B | ● | | | | | • | ● | ● | |
| C | ● | | | | ● | • | | ● | |
| D | ● | | | | ● | • | ● | ● | |
| E | ● | | | | ● | • | ● | | |
| F | | | | ● | ● | • | | | |

| CHARACTERS | CHANNELS (Even-Parity) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| + ③ | | | | | ● | • | ● | | |
| 1 | ● | | | ● | | • | | | |
| 2 | | | | ● | | • | | ● | |
| 3 | ● | | | | | • | | ● | |
| 4 | | | | ● | | • | ● | | |
| 5 | ● | | | | | • | ● | | |
| 6 | | | | | | • | ● | ● | |
| 7 | | | | ● | ● | • | | | |
| 8 | ● | | | | ● | • | | | |
| 9 | | | | | ● | • | | ● | |
| PUTAWAY ① | ● | | | ● | ● | • | ● | | |
| COMPUTE ① | | | | ● | ● | • | ● | ● | |
| TAPE FEED ② | | | | | | • | | | |
| V | ● | | | | ● | • | ● | ● | |
| — | ● | | | ● | ● | • | | ● | |
| A | ● | | | ● | | • | ● | ● | |
| B | | | ● | | | • | | | |
| C | ● | | ● | | | • | | | |
| D | | | ● | | | • | | ● | |
| E | | | ● | | | • | ● | | |
| F | | | ● | | ● | • | | | |

When using NCR 461-3 Recorder (single program, fixed diode-board) the following codes are read out from positions:

| |
|---|
| S – 2 |
| S – 3 |
| S – 9 |
| S – 0 |
| S – 1 |

### NOTES

① Input Controls. These exercise control functions during input, and store no information in memory.

② These have no function during input, and are ignored.

③ Whenever the parent machine reads a "zero", the character "+" will be punched into the paper tape. This will enable the programmer to verify, in every case, that a field of the proper length has been read into the Processor. The first Arithmetic operation performed on each field within the Processor will, of course, replace every "+" with a "zero".

Cash-Control Codes are formed by substituting Channel 8 for Channel 1 in conventional codes. The Cash-Control Codes will normally be used only for input to the Data-Processing System, and should be specified for Cash Registers and for all other recording devices which provide mechanical control over the integrity of cash entries.

By using the TEST READER CODE Instruction, the Processor program can guarantee that cash transactions are input *only* in Cash-Control Code. Since the Tape-Punching Typewriter is incapable of punching in Channel 8, it cannot be manipulated to punch Cash-Control Code, and it is therefore impossible to simulate a cash-transaction paper tape with the Typewriter.

# PAPER TAPE INPUT

## NCR CASH-CONTROL CODE (EXTENDED)

**(Odd-Parity)**

| TYPEWRITER KEY | | CHANNELS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LOWER SHIFT | UPPER SHIFT | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| 0 | ) | | ● | | | ● | • | | ● | |
| 1 | 1 | ● | | | | | • | | | |
| 2 | £ | | | | | | • | | ● | |
| 3 | # | ● | ● | | | | • | | ● | |
| 4 | $ | | | | | | • | ● | | |
| 5 | % | ● | ● | | | | • | ● | | |
| 6 | / | | ● | | | | • | ● | ● | |
| 7 | & | ● | | | | | • | ● | ● | |
| 8 | * | | | | | ● | • | | | |
| 9 | ( | ● | ● | | | ● | • | | | |
| A | a ① | ● | ● | | ● | | • | | | |
| B | B | | ● | | ● | | • | | ● | |
| C | c ① | ● | | | ● | | • | | ● | |
| D | d | | ● | | ● | | • | ● | | |
| E | E | ● | | | ● | | • | ● | | |
| F | F | | | | ● | | • | ● | ● | |
| G | G | ● | ● | | ● | | • | ● | ● | |
| H | H | | ● | | ● | ● | • | | | |
| I | I | ● | | | ● | ● | • | | | |
| J | J | ● | ● | ● | | | • | | | |
| K | K | | ● | ● | | | • | | ● | |
| L | L | ● | | ● | | | • | | ● | |
| M | m | | ● | ● | | | • | ● | | |
| N | n | ● | | ● | | | • | ● | | |
| ⟳ | ⟲ | | | ● | | | • | ● | ● | |
| P | p | ● | ● | ● | | | • | ● | ● | |

| TYPEWRITER KEY | | CHANNELS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LOWER SHIFT | UPPER SHIFT | 8 | 7 | 6 | 5 | 4 | • | 3 | 2 | 1 |
| Q | Q | | ● | ● | | ● | • | | | |
| R | R | ● | | ● | | ● | • | | | |
| S | S | | | ● | ● | | • | | ● | |
| T | T | ● | ● | ● | ● | | • | | ● | |
| U | u | | | ● | ● | | • | ● | | |
| V | v | ● | ● | ● | ● | | • | ● | | |
| W | w | | ● | ● | ● | | • | ● | ● | |
| X | x | ● | | ● | ● | | • | ● | ● | |
| Y | Y | | | ● | ● | ● | • | | | |
| Z | Z | ● | ● | ● | ● | ● | • | | | |
| @ | + | | | ● | | | • | | | |
| ¢ | □ | | ● | ● | ● | | • | | | |
| - | △ | | | | ● | | • | | | |
| . | , | | | | | ● | • | ● | ● | |
| SPACE | SPACE | | ● | | | ● | • | ● | | |
| PUTAWAY ↗ ① | PUTAWAY ↙ ① | ● | | | | ● | • | ● | | |
| COMPUTE > ① | STOP ⌋ ② | ● | ● | ● | | ● | • | ● | | |
| CLEAR ⊠ ① | CLEAR ⊠ ① | ● | ● | | | ● | • | ● | ● | |
| UP SHIFT ① | UP SHIFT ① | ● | ● | | ● | ● | • | | ● | |
| DOWN SHIFT ① | DOWN SHIFT ① | | | | ● | ● | • | | ● | |
| CODE DELETE ② | CODE DELETE ② | ● | ● | ● | ● | ● | • | ● | ● | |
| TAPE FEED ② | TAPE FEED ② | | ● | | | | • | | | |
| CARR RET ② | CARR RET ② | ● | | | ● | ● | • | ● | ● | |
| TAB ② | TAB ② | | ● | | ● | ● | • | ● | ● | |
| BACKSPACE ② | | DOES NOT PUNCH | | | | | | | | |

The complete Extended Code (equivalent to NCR Typewriter Code) is mechanized in the 360 Paper-Tape Reader, offering a wide choice of identification characters when this Code is used with a keyboard machine and Paper-Tape Recorder.

### NOTES

① Input Controls. These exercise control functions during input, and store no information in memory.

② Typewriter controls. These have no function during input, and are ignored.

# MULTI-PURPOSE CONVERTER CONTROL PANEL

## POWER SWITCHES

At the right side of the panel, in the upper row, are five power switches for the Converter and its peripheral units. From left to right, the switches control:

Paper Tape Reader
Punched Card Reader
Printer
Paper Tape Punch
Converter and Magnetic Tape Handler
(Controls for the Card Punch are mounted on its own cabinet)

Each of the five units is represented on the Control Panel by:

An ON button;
An OFF button;
A BUSY light.

When the power is first turned on at any unit, its ON button will show a blue light (not ready), while that unit goes through its own warmup-test cycle.

When the unit has completed its self-testing operation, its ON button will show a yellow light (ready).

Whenever the unit is actually in use, its yellow BUSY light will be turned on.

If the unit should, for any reason, go into an Error-Halt, its ON button will show a red light.

## OPERATION SELECTOR

This dial switch selects any one of ten operations to be performed by the Converter:

| | | |
|---|---|---|
| Paper Tape Reader | to | Magnetic Tape |
| Card Reader | to | Magnetic Tape |
| Card Punch | from | Magnetic Tape |
| Paper Tape Punch | from | Magnetic Tape |
| Printer | from | Magnetic Tape |

Advance magnetic tape one record
Backup magnetic tape one record
Write a CRM-record on magnetic tape
Erase magnetic tape to the end of the reel
Search magnetic tape for a record corresponding to the Record Selection Switches, then Halt

Paper Tape to Magnetic Tape:

The information on the paper tape is transcribed to magnetic tape, character by character, as ~~12~~ 13-word records.

The "putaway" code on paper tape has the same function as when the Processor itself is reading directly into Memory, except that putaways on magnetic tape are left-justified. After a "putaway" code, the previous field will appear at the left end of the word on magnetic tape, with the rest of the word filled out with ~~zeros~~ spaces to the right.

The "compute" code on paper tape causes a putaway (unless the Converter's Input Register is empty), and then fills out the balance of the current record on magnetic tape with ~~zeros~~ spaces.

Punched Cards to Magnetic Tape:

Each card is transcribed to magnetic tape as a minimum-length 10-word record.

Column 1 on the card goes into the 99-field of the first word in the record, column 2 goes into the 88-field, etc. Columns 11-20 occupy the second word of the record, and so on for eight words. Then the Controller records two more words of "spaces" to complete the 10-word record.

Magnetic Tape to Punched Cards:

Eight words of each magnetic tape record are delivered to the Card Punch (see description of Magnetic Tape Mode Switch).

The plugboard of the Card Punch may be used for its conventional functions.

Magnetic Tape to Paper Tape:

Twelve words of each magnetic tape record are delivered, character by character, to the Paper Tape Punch, starting with the 99-field of the initial word (see description of Magnetic Tape Mode Switch).

If the record is shorter than 12 (or 13) words, the Converter fills out the balance of the 12-word Punch Block with "spaces".

Magnetic Tape to Printer:

Twelve words of each magnetic tape record are delivered to the Printer (see description of Magnetic Tape Mode Switch).

If the record is shorter than 12 (or 13) words, the Converter fills out the balance of the 12-word Print Block with "spaces".

## MAGNETIC TAPE MODE

This dial switch is operative only during output operations, and it selects "edited" or "unedited" magnetic tape operation.

For this purpose, an "edited" magnetic tape is defined as a tape on which every record has had its first word edited to contain a slew control in the 99-field (if the tape is to be printed) and a record identification in the 80-field.

Operating in the "edited" mode, the Converter delivers the second through the thirteenth word of every record to the Paper Tape Punch or the Printer, and the second through the ninth word of every record to the Card Punch.

When printing, all the slew controls described on page IV-X-3 are effective, except that there is no "branch" marker set by a "skip" code on the Printer's slew-control tape loop. Therefore, when editing for off-line printing, it is necessary to have the editing program count the lines on each page.

For this purpose, an "unedited" magnetic tape is defined as a tape on which the records have not had their first words edited in this fashion.

Operating in the "unedited" mode, the Converter delivers the first through the twelfth word of every record to the Paper Tape Punch or the Printer, and the first through the eighth word of every record to the Card Punch.

The three positions 0, 1, 2 of this switch are differentiated only when printing. Since there is no slew control in each magnetic tape record, these positions govern the vertical spacing of the printer; they cause it to skip 0, 1, or 2 lines after each printed line (that is, print single, double, or triple-space). On the Printer's slew-control tape loop, the configurations "skip" and "stop" retain their functions, and may be used in their conventional capacities, except for automatic end-of-page branch (see page IV-X-3).

When operating the Paper Tape Punch or the Card Punch, the three positions 0, 1, 2 of this switch have identical functions; any one of them specifies "unedited" tape.

# OPERATING CONTROLS

At the left side of the panel, in the second row, are five Operating Control Switches:

OPERATE CONVERTER:

Pressing this switch causes the Converter to begin operating as defined by the Operation Selector Switch, the Magnetic Tape Mode Switch, and by the other Operating Control Switches.

RESET CONVERTER:

Pressing this switch returns the Converter to the operating state after an Error-Halt. The red light in the Converter's Power On Switch will then be replaced by a yellow light.

CONTINUOUS/SINGLE:

Yellow light: Continuous operation. The Converter will perform the operation specified by the Operation Selector Switch, continuously.

Blue light: Single-record operation. The Converter will perform the operation for one magnetic tape record (i.e. one printed line, one punched card, etc.) and then halt.

The five operations specified at the right side of the Operation Selector Switch are always single; they are independent of the state of this Control Switch.

SELECTIVE OUTPUT:

Yellow light: No. The Converter delivers every record on the magnetic tape to the Printer, Paper Tape Punch, or Card Punch.

Blue light: Yes. The Converter delivers to the output device only those records in which the 80-field of the first word exactly corresponds to the settings of the Record Selection Switches.

This Control Switch functions only when performing an output operation (Print, Punch Cards, Punch Paper Tape) from "edited" magnetic tape.

CRM: Yellow light: Halt, if the Converter encounters any magnetic tape record containing a CRM ("v" in the 44-field of the first word). Do not output the CRM-record.

Blue light: Ignore the presence of a CRM in any record, treating it just like any other record.

This Control Switch has no function when performing an input operation.

## STATUS LIGHTS

At the left side of the panel, in the top row, are five Status Lights:

HANDLER OR CARD PUNCH, OPERATOR ATTENTION: Blue light.

The Handler is at the end of the tape (positioned on the trailer), or else it has finished rewinding (positioned on the leader).

In the Card Punch, either the input stacker is empty, or the output stacker is full.

PAPER TAPE READER, MEDIA ERROR: Blue light.

The Paper Tape Reader has encountered a punched configuration which does not meet the (odd or even) parity requirement of the code being read.

The character "x" (code 11 1111) is substituted for the illegible character on the magnetic tape, and reading continues until the record is complete. Then the operation halts, and this blue light is turned on.

The recommended procedure is for the operator to record a CRM-record on the tape (following the record containing the illegible character), and then to resume the operation. When this magnetic tape is later read into the Processor, the program must take account of this possibility.

This procedure is based on the fact that, if the Paper Tape Reader finds the same configuration punched into the tape, at both of its duplicate reading stations, and if that configuration does not meet parity, this can only be the result of an incorrect punching of the tape, and there is no point in attempting to re-read.

HANDLER ERROR HALT: Red light. Reset at the Handler.

HANDLER SEARCH HALT: Blue light.

When performing the Search-Halt Operation, the Handler has found the desired record, and has therefore halted.

HANDLER CRM HALT: Blue light.

The Handler has encountered a CRM-record, and has therefore halted.

## RECORD SELECTION SWITCHES

These nine switches are operative during the Search-Halt operation, and during Selective Output of "edited" magnetic tape. The switches are numbered to correspond to the character-positions of the first word of each record on the tape: the field which has been edited to contain a record identification.

Each of these switches may be set to any one of 41 characters, or to "ignore".

During Search-Halt operation, the Handler searches the magnetic tape until it finds a record corresponding to the setting of the Record Selection Switches. Then the Handler halts, and the blue Handler Search Halt light on the Converter panel is turned on.

During Selective Output, the Handler performs the same search, and then the Converter delivers the selected record to the Printer, Paper Tape Punch, or Card Punch.

If the Converter is set to Single-record operation, the operation terminates, and the Converter halts.

If the Converter is set to Continuous operation, the Handler immediately begins another search, and so on.

During the search, each of the nine Record Selection Switches (other than those set to "ignore") is compared with the corresponding character-position in the 80-field of the first word in each record on the magnetic tape. The search terminates when exact correspondence is found, in every character-position, between the record identification on the tape, and the setting of the Record Selection Switches.

## NON-DATA CHARACTERS AT THE PRINTER

When editing programs, or Memory Dumps, for off-line printing, it is not possible to rely on the Overflow Alarm to identify those lines which contain non-data characters. The editing program must test each line, before recording it on magnetic tape, to determine whether there are any non-data characters in it; if so, then they must be located, and the program must edit an additional line as explained on page IV-X-4. All of the standard Memory-printout programs include this feature when editing for off-line printing.

## CORRECTIONS TO THIS SECTION

Please make the following corrections in Section VI-320:

Page 1, add the following just before the heading "Operation Selector":

> All power switches are inoperative whenever the Converter is in operation.

Page 2, Paper Tape to Magnetic Tape:

> The information on paper tape is transcribed as 13-word records.
>
> After a "putaway" code, the rest of the word will be filled out with spaces to the right.
>
> The "compute" code causes a putaway, then fills out the balance of the record with spaces.

Page 2, Magnetic Tape to Paper Tape:

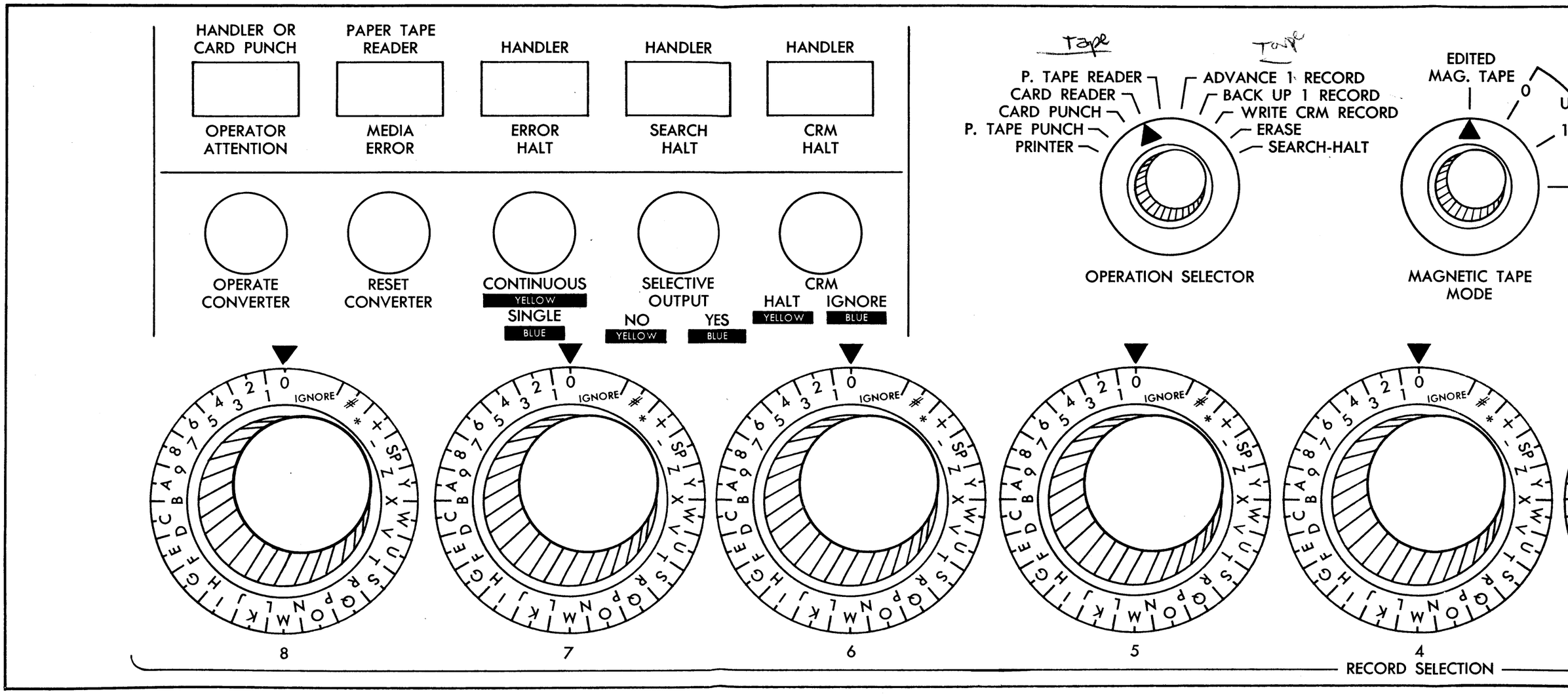
> If the record is shorter than 12 (or 13) words, the Converter does not fill out the balance of the 12-word Punch Block. Only the 9, 10 or 11 words from the Magnetic Tape will be punched.

Page 4, Selective Output:

> This Control Switch functions at all times when performing an output operation (Print, Punch Cards, Punch Paper Tape) from either "edited" or "unedited" magnetic tape.

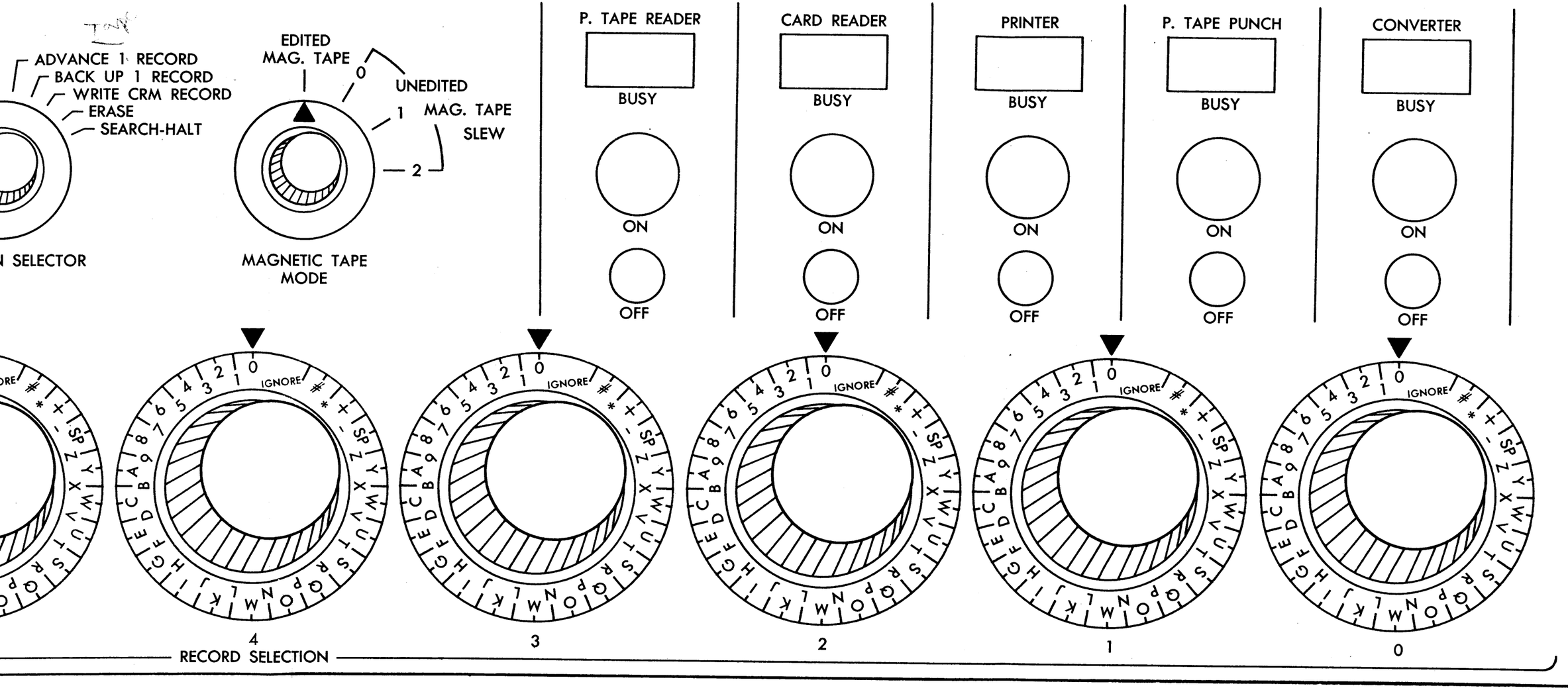
NOTE: All fill-out of the current record is normally with "spaces". However, there is a switch inside the Converter cabinet, available to maintenance personnel, which will cause all fill-out to be with "zeros" instead.

# 320 MULTI-PURPOSE CONVERTER
# CONTROL PANEL




# 320 MULTI-PURPOSE CON

## CONTROL PANEL




# 320 MULTI-PURPOSE CONVERTER

## CONTROL PANEL

# PRINTER CONVERTER CONTROL PANEL

## POWER SWITCHES

At the right side of the panel, in the upper row, are power switches for the Converter and Magnetic Tape Handler, and for the Printer.

Each of these units is represented on the Control panel by:

An ON button;
An OFF button;
A BUSY light.

When the power is turned on at either unit, its ON button will show a blue light (not ready), while that unit goes through its own warmup-test cycle.

When the unit has completed its self-testing operation, its ON button will show a yellow light (ready).

Whenever the unit is actually in use, its yellow BUSY light will be turned on.

If the unit should, for any reason, go into an Error-Halt, its ON button will show a red light.

## OPERATION SELECTOR

This dial switch selects the operation which is to be performed by the Converter:

Print "edited" magnetic tape
Print "unedited" magnetic tape
Selective Print "edited" magnetic tape

Search magnetic tape for a record corresponding to the
Record Selection Switches, then Halt
Advance magnetic tape one record
Backup magnetic tape one record

For this purpose, an "edited" magnetic tape is defined as a tape on which every record has had its first word edited to contain a slew control in the 99-field, and a record identification in the 80-field.

Operating in the "edited" mode, the Converter delivers to the Printer the second through the thirteenth word of every record. If the record contains less than 13 words, the Converter fills out the balance of the Print Line with "spaces".

All the slew controls described on page IV-X-3 are effective, except that there is no "branch" marker set by a "skip" code on the Printer's slew-control tape loop. Therefore, when editing for off-line printing, it is necessary to have the editing program count the lines on each page.

For this purpose, an "unedited" magnetic tape is defined as a tape on which the records have not had their first words edited in this fashion.

Operating in the "unedited" mode, the Converter delivers the first through the twelfth word of every record to the Printer. If the record contains less than 12 words, the Converter fills out the balance of the Print Line with "spaces".

Since there is no slew control in the magnetic tape record, the positions 0, 1, 2 of this switch govern the vertical spacing of the Printer; they cause it to skip 0, 1, or 2 lines after each printed line (that is, print single, double, or triple-space). On the Printer's slew-control tape loop, the configurations "skip" and "stop" retain their functions, and may be used in their conventional capacities, except for automatic end-of-page branch (see page IV-X-3).

## OPERATING CONTROLS

At the left side of the panel, in the second row, are four Operating Control Switches:

OPERATE CONVERTER:

Pressing this switch causes the Converter to begin operating as defined by the Operation Selector Switch, and by the other Operating Control Switches.

RESET CONVERTER:

Pressing this switch returns the Converter to the operating state after an Error-Halt. The red light in the Converter's Power On Switch will then be replaced by a yellow light.

CONTINUOUS/SINGLE:

Yellow light: Continuous operation. The Converter will perform the printing operation specified by the Operation Selector Switch, continuously.

Blue light: Single-record operation. The Converter will print the contents of one magnetic tape record (i.e. one line of print) and then halt.

CRM:

Yellow light: Halt, if the Converter encounters any magnetic tape record containing a CRM ("v" in the 44-field of the first word). Do not print the CRM-record.

Blue light: Ignore the presence of a CRM in any record, treating it just like any other record.

## STATUS LIGHTS

At the left side of the panel, in the top row, are four Status Lights:

HANDLER, OPERATOR ATTENTION: Blue light.

The Handler is at the end of the tape (positioned on the trailer), or else it has finished rewinding (positioned on the leader).

HANDLER ERROR HALT: Red light. Reset at the Handler.

HANDLER SEARCH HALT: Blue light.

When performing the Search-Halt Operation, the Handler has found the desired record, and has therefore halted.

HANDLER CRM HALT: Blue light.

The Handler has encountered a CRM-record, and has therefore halted.

## RECORD SELECTION SWITCHES

These nine switches are operative during the Search-Halt operation, and during Selective Printing of "edited" magnetic tape. The switches are numbered to correspond to the character-positions of the first word of each record on the tape: the field which has been edited to contain a record identification.

Each of these switches may be set to any one of 41 characters, or to "ignore".

During Search-Halt operation, the Handler searches the magnetic tape until it finds a record corresponding to the setting of the Record Selection Switches. Then the Handler halts, and the blue Handler Search Halt light on the Converter panel is turned on.

During Selective Printing, the Handler performs the same search, and then the Converter delivers the selected record to the Printer.

If the Converter is set to Single-record operation, the operation terminates, and the Converter halts.

If the Converter is set to Continuous operation, the Handler immediately begins another search, and so on.

During the search, each of the nine Record Selection Switches (other than those set to "ignore") is compared with the corresponding character-position in the 80-field of the first word in each record on the magnetic tape. The search terminates when exact correspondence is found, in every character-position, between the record identification on the tape, and the setting of the Record Selection Switches.

## NON-DATA CHARACTERS AT THE PRINTER

When editing programs, or Memory Dumps, for off-line printing, it is not possible to rely on the Overflow Alarm to identify those lines which contain non-data characters. The editing program must test each line, before recording it on magnetic tape, to determine whether there are any non-data characters in it; if so, then they must be located, and the program must edit an additional line as explained on page IV-X-4. All of the standard Memory-printout programs include this feature when editing for off-line printing.

# 322 PRINTER CONVERTER
# CONTROL PANEL

HANDLER
OPERATOR ATTENTION

HANDLER
ERROR HALT

HANDLER
SEARCH HALT

HANDLER
CRM HALT

PRI
EDITED M

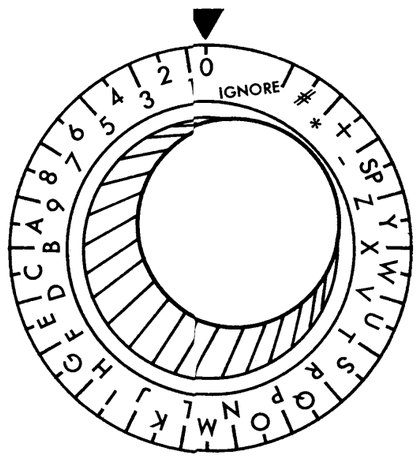
UNEDITED
SLEW

OPERATE CONVERTER

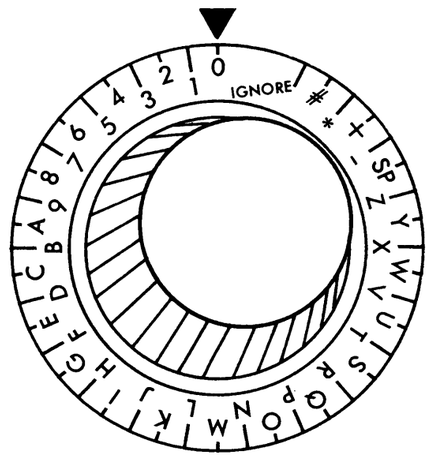
RESET CONVERTER

CONTINUOUS
YELLOW
SINGLE
BLUE

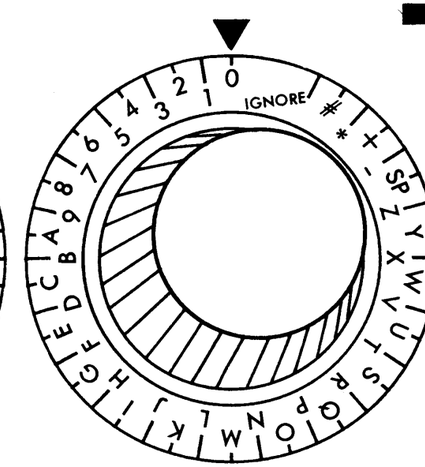
CRM
HALT YELLOW
IGNORE BLUE



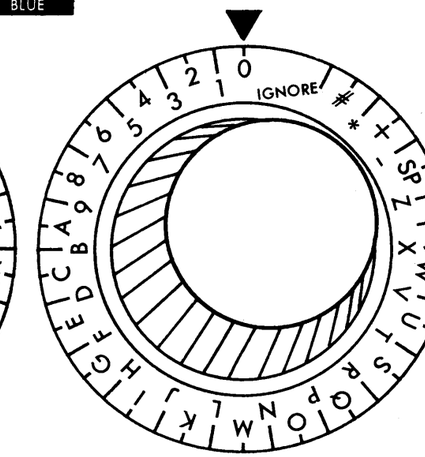
8



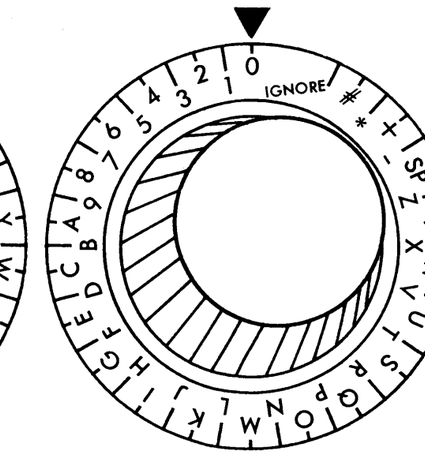
7



6



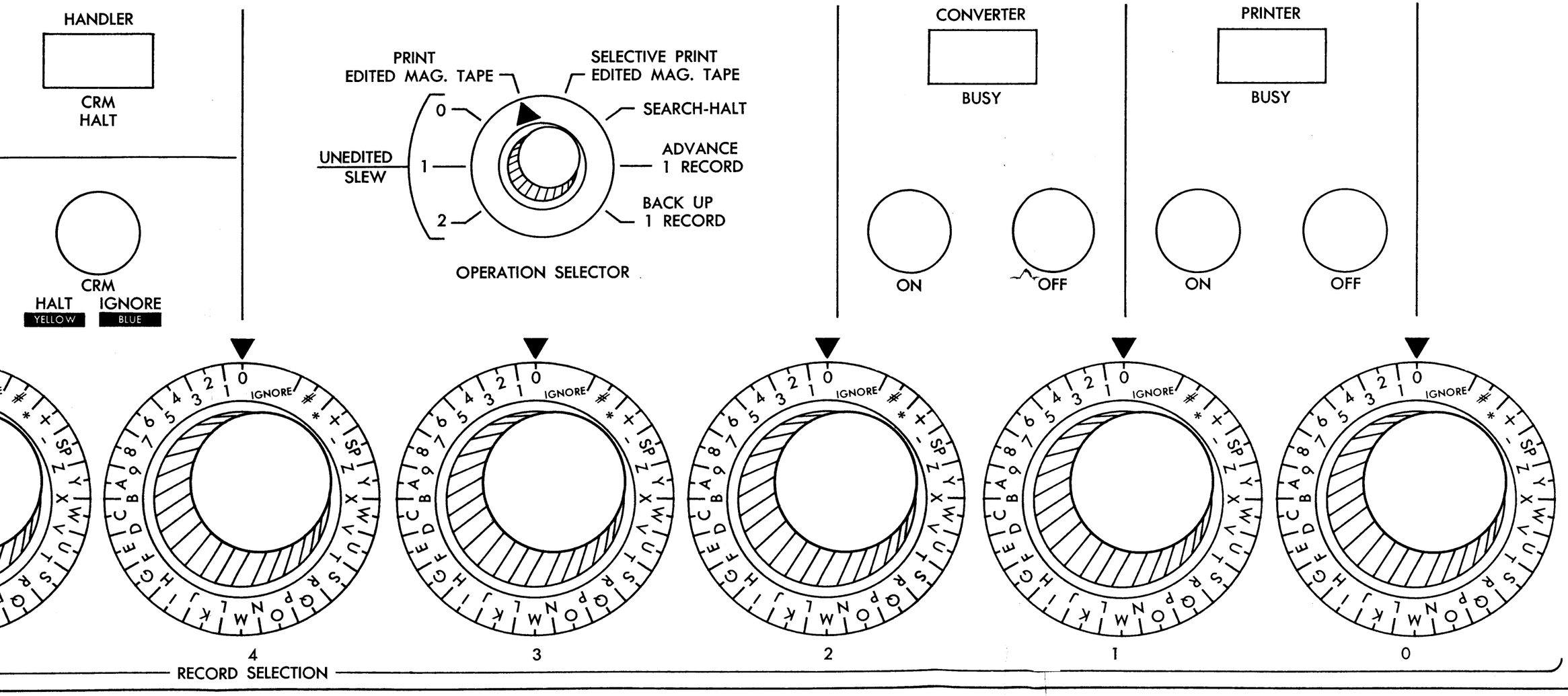
5



4

RECORD SELECTION

# 322 PRINTER CONVERT
## CONTROL PANEL




# 322 PRINTER CONVERTER
## CONTROL PANEL

# MAGNETIC TAPE CONTROLLER - CONTROL PANEL

## OPERATING SWITCHES

SELECTION:

This is a rotary switch which permits any number to be assigned to any Controller.

OPERATE:

After any error-halt this button resets the Controller, and places it back in the operating state.

Blue light: When power is turned on in the Controller (this is done at the Operator's Console) this button shows a blue light while the Controller and its Handlers are in their automatic warmup-test cycles.

Yellow light: When the Controller and all its Handlers have completed their cycles, this button shows a Yellow light, indicating "ready".

EMERGENCY OFF:

Power is normally turned on and off in all peripheral units from the main Operator's Console. However, in the event of fire or other emergency, each peripheral unit has an EMERGENCY OFF button which completely cuts off all circuits in the unit.

## A NOTE ON THE COPY-BUFFER

The Controller contains a small circulating buffer-storage, to accommodate the minor differences in speed which must occur between the two Handlers during a Copy. In a long Copy, the speed differential may build up beyond the capacity of this buffer; if so, the Controller automatically stops both tapes, repositions them at the beginning of the record in which the buffer overran, and then immediately resumes the Copy with no attention from the program, or from the operator. In a long Copy, this condition will occur after an average of 25,000 words have been copied; in a short Copy it will rarely occur.

# INDICATOR LIGHTS

DUPLICATE HANDLERS:

The Controller will immediately error-halt if two or more of its Handlers have been assigned the same number.

HALT ON BUFFER OVERRUN:

There may be occasions when Management will choose to take the risk of deferring routine maintenance in favor of emergency, or high-priority, processing. It is possible under these circumstances that two Handlers might eventually get far enough apart in speed so that the buffer would overrun during the first record copied, or during first record after the automatic resumption of a Copy, making it impossible to proceed.

In that case the Controller error-halts, and turns on the red light for BUFFER OVERRUN.

SEARCH CONTROL ERROR:

The Search Control Word, used by Copy, is stored in the Controller during the Copy, in a recirculating register. The contents of this register is continuously checked; if any parity-failure occurs, the Controller error-halts and turns on this red light.

CONTROLLER BUSY:

Yellow light whenever the Controller is busy. This duplicates the corresponding light on the main Operator's Console.

HANDLER ERROR BRANCH:

Blue light whenever one of the Handlers detects an error of the type which causes a program branch.

This light flashes momentarily if the error occurs in a Read, Write, or Write-Copy-Read; it stays on until the next time the Controller is used if the error occurs in a Write-Copy.

HANDLER WRITE LOCKOUT:

Red light to indicate error-halt whenever the program attempts to Write on any Handler which is in the Write Lockout state (ie - the tape reel does not have a Write-Permissive Ring attached to it).

HANDLER WRITE ON TRAILER:

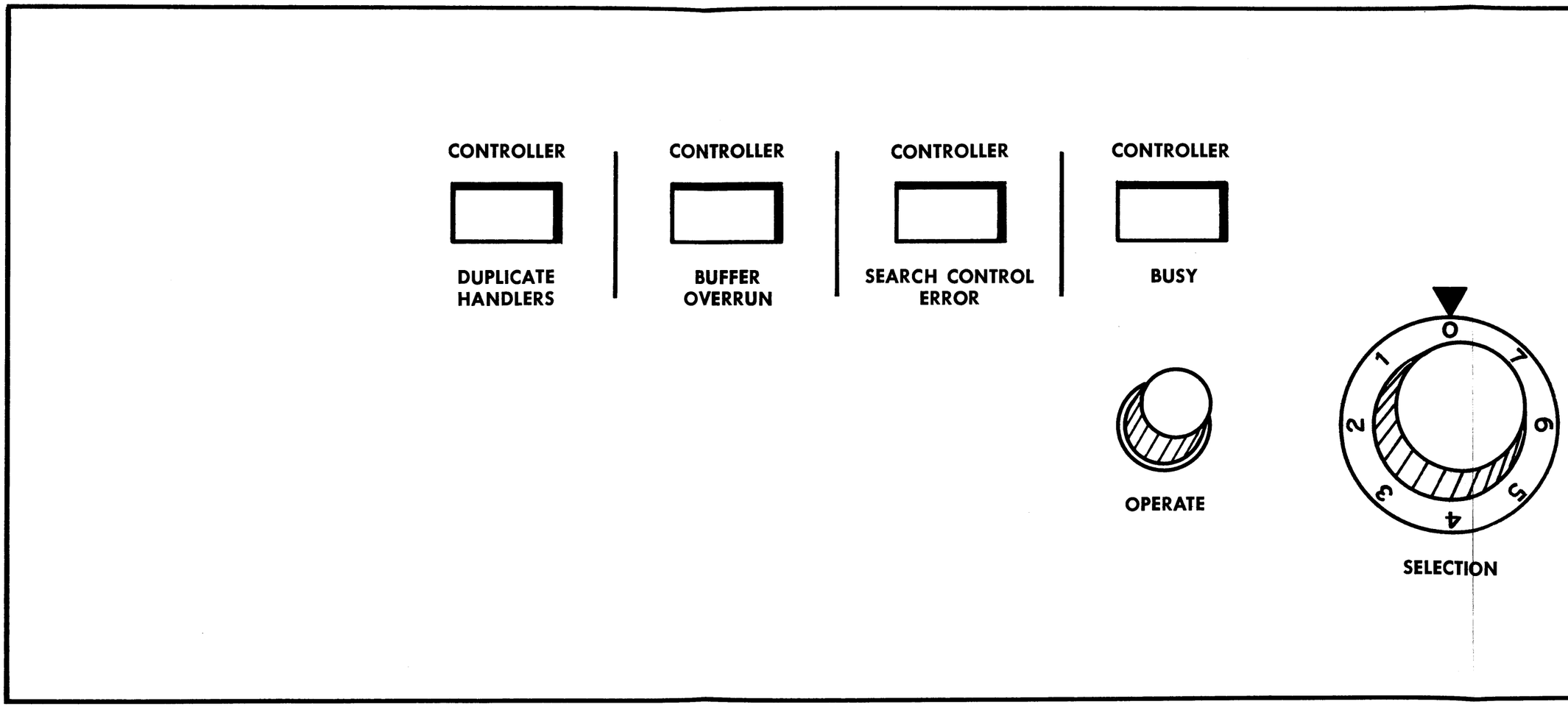
Red light to indicate error-halt if the program, in spite of the Destination-Tape Warning Marker (DWM), attempts to record on the trailer which fastens the tape to the reel.

HANDLER BRM/ERM ERROR:

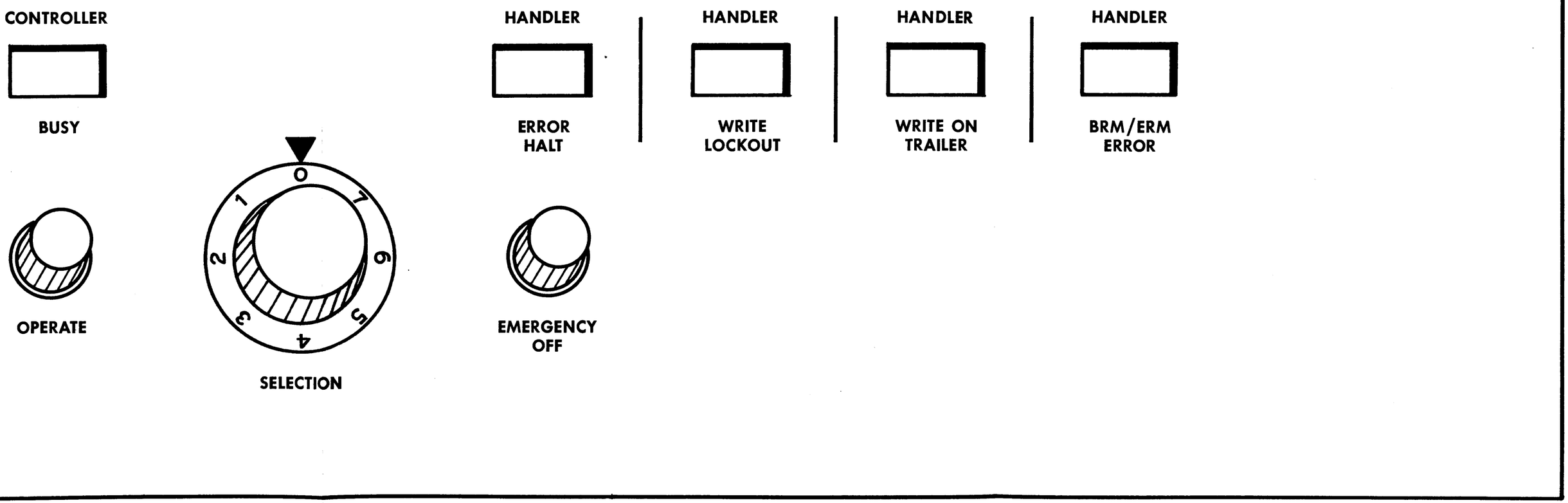
Red light to indicate error-halt if a Handler encounters a failure in the BRM/ERM alternation check, or does not find the record-marker within the time permitted by the Timing Check.

This light is also used to indicate any other condition (such as broken tape, loss of vacuum, etc.) which would cause a Handler to error-halt.

# 330 MAGNETIC TAPE CONTROLLER
## CONTROL PANEL




**330 MAGNETIC TAPE CONTROLLER**
**CONTROL PANEL**




**330 MAGNETIC TAPE CONTROLLER**
**CONTROL PANEL**

# MAGNETIC TAPE HANDLER CONTROL PANEL

## OPERATING SWITCHES

SELECTION:

This is a rotary switch which permits any number to be assigned to any Handler. If two Handlers on the same Controller are assigned the same number, the Controller immediately error-halts.

POWER ON-OFF:

The Handler is the only peripheral unit with its own power control, so that one or more Handlers may be turned off for several hours if current processing does not require them.

This switch is normally left ON, in which case the Handler power goes on and off simultaneously with the Controller power.

Blue light: When power is turned on in the Handler, this button shows a blue light while the Handler is in its automatic warmup-test cycle.

Yellow light: When the Handler has completed this cycle, this button shows a yellow light, indicating "ready".

USE LOCKOUT:

SET shows a blue light, or CLEAR shows a yellow light, at all times, to indicate the state of the Handler.

The buttons may be used to SET or CLEAR Use Lockout manually.

REWINDING:

Shows a blue light when the Handler is executing a Rewind.

The button may be used to manually initiate a Rewind, but this button is inoperative while the Handler is in use.

# INDICATOR LIGHTS

WRITE LOCKOUT:

SET shows a blue light, or FREE shows a yellow light, at all times, to indicate the state of the Handler.

TAPE SELECTED:

A yellow light is turned on for SOURCE or DESTINATION when the Handler is used. This light then remains on, and shows the function last executed by the Handler. A Handler will usually be used only as source or as destination for extended periods of time, and therefore these lights indicate the function of each Handler during the current program.

TAPE MALFUNCTION:

Red light to indicate Handler error-halt due to failure of any of the numerous automatic checks which are continuously being performed.

## WRITE LOCKOUT

Write Lockout consists of completely disconnecting the recording circuits in a Handler. It is established on any Handler by the absence of a Write-Permissive Ring attached to the reel of tape. Since NCR provides only one Ring per Handler (and several of those should be carefully locked away) it is impossible for the operators to attach Rings to reels promiscuously, and if a Ring is accidently left on after a reel has been removed from the Handler, there will be no Ring available for the next reel. As a further safeguard, the reel will not fit into its plastic storage-box if it still has a Ring attached to it.

If desired, Write Lockout may also be established on any Handler which is never to be used as Destination, by means of a switch available to the Maintenance Engineer.

# OPERATION OF MAGNETIC TAPE HANDLER

The Tape Drive mechanism uses a Supply Reel and a Take-up Reel, appropriate capstans to lead the tape by the proper route from one Reel to the other, pinch-rollers to drive the tape in either direction, and brakes to stop the tape. (For simplicity, the pinch-rollers and brakes are not shown on the schematic diagram which appears at the end of this section.) At each side of the Handler is a glass-enclosed vacuum chamber containing a free loop of tape; air is pumped out through ports at the upper and lower ends of each chamber, and the tape loops are continuously held in approximately the positions shown on the diagram.

While the Handler is operating, the tape is moved at a continuous rate past the Read-Write Heads. When the tape is moving forward, this causes the left-hand loop to shrink, and the right-hand loop to grow, in their respective chambers. These "out of balance" conditions are detected by devices which are highly sensitive to air-pressure changes in the chambers, and which cause the Reels to turn, adjusting the slack in the loops.

Whenever the left-hand loop shrinks, the Supply Reel begins to unwind; when the loop has resumed its proper size, the Reel stops turning. Whenever the right-hand loop grows, the Take-up Reel begins to wind; when the loop has resumed its proper size, the Reel stops turning. When the tape is moving in the reverse direction, the entire operation is reversed. Thus each Reel turns intermittently (although to the eye the motion usually appears continuous) in order to accommodate itself to the movement of the tape.

While the Handler is operating, three Tape Sensing Stations are activated by the presence or absence of an electrically conductive coating on the back of the tape:

Tape Sensing Station A:

Detects beginning of trailer in the forward direction.
End-of-tape for Source Tape.
Error-halt for Destination Tape.

Detects beginning of leader in the reverse direction.
Termination of Rewind.

Tape Sensing Station B: Active only for Destination Tape.

Detects presence of leader, indicating that Handler is at the beginning of tape. When tape is started, the Write Head erases the tape until this Station detects end of leader, at which time the Write Head begins to record the information on the tape, about 3 feet from the beginning.

Detects Destination-tape Warning Marker (DWM) which signals that end of tape is approaching.

Tape Sensing Station C: Active only for Source Tape.

Detects presence of leader, indicating that Handler is at the beginning of tape. When tape is started, no reading is attempted until this Station detects end of leader.

The Timing Check (see Chapter II) is then replaced by the requirement that the Read Head must find no information on the tape before the first ERM.

Length of Leader: 24 feet.

Length of Trailer: 12 feet.

DWM to Trailer: 40 feet.

Conductive coating on the back of the tape:
Leader and Trailer, full width of the tape.
Destination Warning Marker (DWM), half width of the tape.

## CHANGING TAPES

The cabinet door, which provides access to the tape, is automatically locked at all times by an electrically-operated bolt, except when the Handler is in a Use Lockout state. Therefore the operator cannot touch the tape unless it has been rewound, with Use Lockout, by the program. When a tape is to be rewound, but remain on the Handler for later use, the program will rewind without Use Lockout and deny the operator access to the tape. In unusual circumstances it may be necessary to provide manual access to the tape at a normally unauthorized time; the operator may then set Use Lockout manually on the Control Panel and permit the door to be opened, but this cannot happen inadvertently.

When the tape is being rewound, the operation continues until the beginning of the leader is detected at Tape Sensing Station "A", whereupon the tape comes to rest with the tape-to-leader splice lying between "A" and the Supply Reel. If the tape was rewound with Use Lockout, the operator may now open the door, and this automatically sets a brake on each Reel.

The operator must next swing the Tape Clamp into position to hold the tape against "A". He then:

Opens the splice

Removes the Supply Reel

Transfers the Write-Permissive Ring to the new Supply Reel if this is a Destination Tape

Mounts the new Supply Reel

Splices the tape to the leader

Releases the Tape Clamp

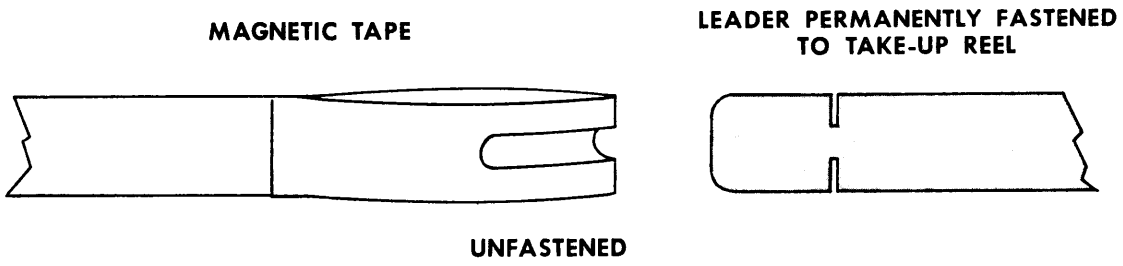
Closes the door (which releases the Reel Brakes)

Presses CLEAR USE LOCKOUT on the Control Panel.

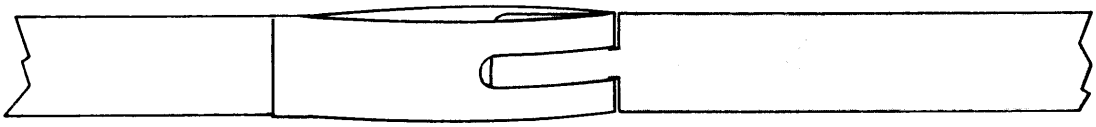
and the Handler is once more ready for use, with the door locked.

If the operator forgets to release the Tape Clamp, or fails to close the door properly, the CLEAR USE LOCKOUT button is inoperative, and the Handler will remain in the Use Lockout state until both conditions have been corrected. If a tape should be mounted on a Handler while the power is off (the door is unlocked at that time also) and then the power turned on, the Handler will never complete its warmup-test cycle (blue light in the Power button) unless the Tape Clamp is released and the door closed, even if the Handler is not in Use Lockout.

The tape-to-leader splice is accomplished in the following manner:




UNFASTENED

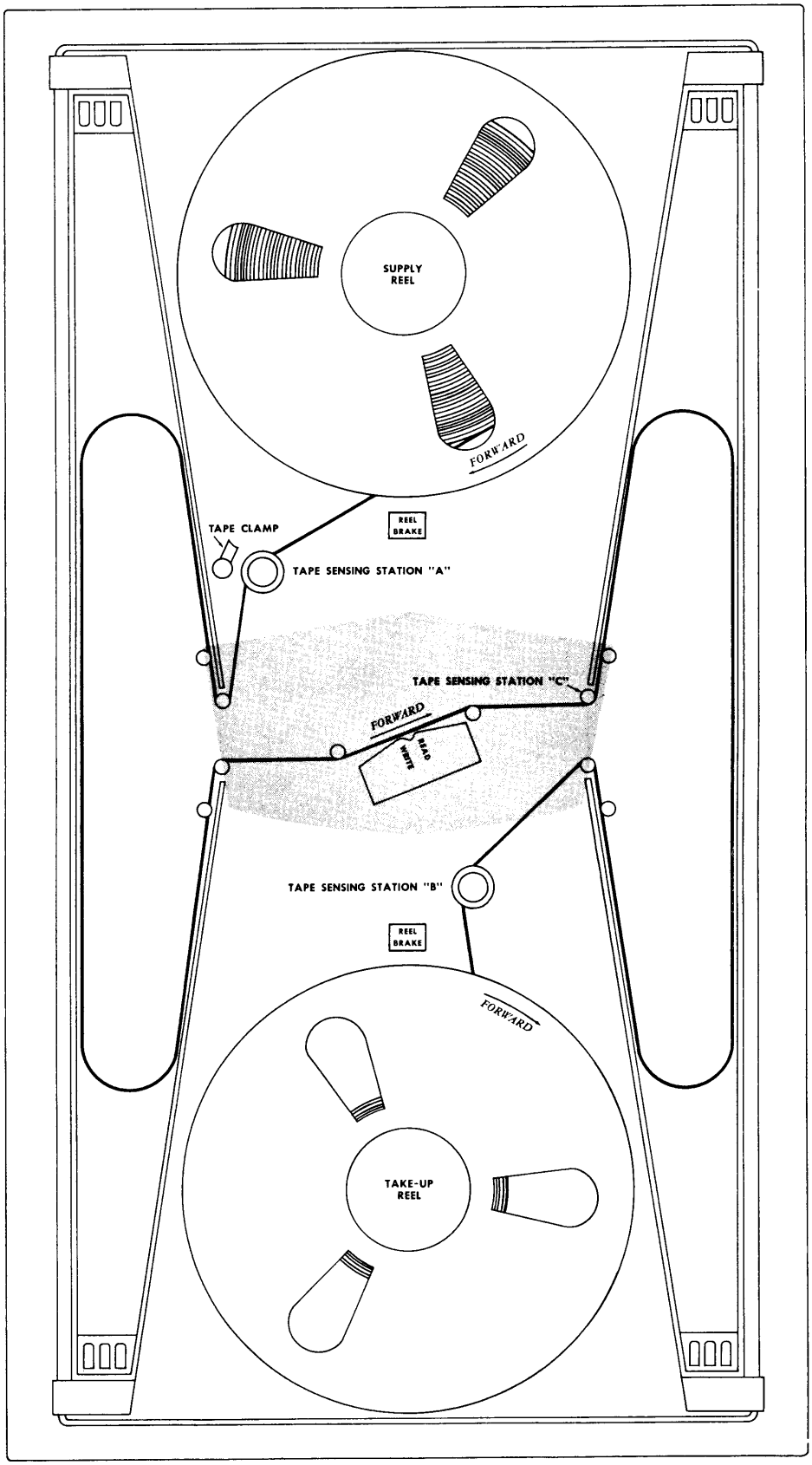


FASTENED

It will be noted that the leader (which is permanently fastened to the Take-up Reel) remains completely threaded at all times while changing tapes. In fact, the major portion of the tape drive is hidden by a cover, represented by the shaded area in the schematic diagram. This cover

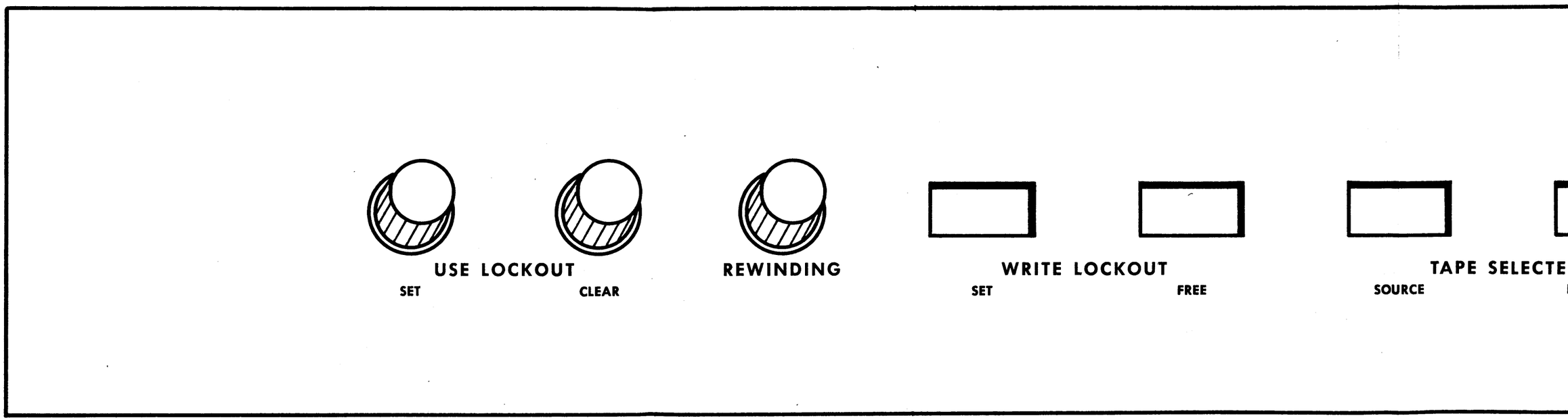
serves as a dust-guard for the Read-Write Heads, and also locks the glass doors of the vacuum chambers; the operator should never have any reason to open this cover.

Near each Reel is a button marked REEL BRAKE. While this button is pressed, the brake on the corresponding Reel is released. The operator will momentarily release the brake on the Supply Reel so that he may rotate it to the most convenient position for closing the tape-to-leader splice. Only the Maintenance Engineer should ever have occasion to touch the release button for the Take-up Reel.

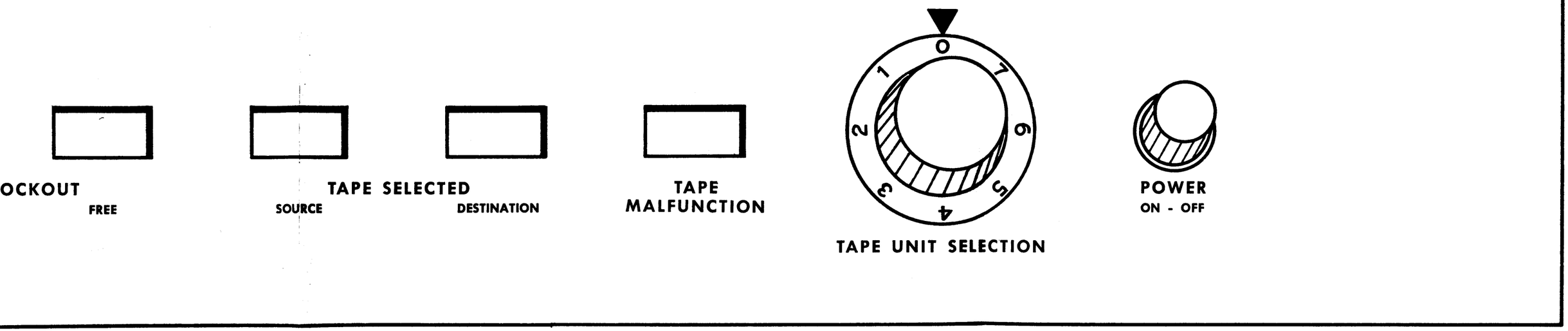



**MAGNETIC TAPE HANDLER**

# 332 MAGNETIC TAPE HANDLER
## CONTROL PANEL




**332 MAGNETIC TAPE HANDLER**
**CONTROL PANEL**




# 332 MAGNETIC TAPE HANDLER
CONTROL PANEL