RCA Information Systems

301 Processor Student Guide

Computer Systems Division
Field Engineering Technical Education

301 Processor

Student Guide

September 1970
July 1970
October 1969
July 1969
PSM 213

RCA Computer Systems Division
Field Engineering Technical Education
Camden, New Jersey 08101

CONTENTS

301 PROCESSOR

SECTION I

General System Information

SECTION II

SECTION III

SECTION IV

SECTION V

SECTION V

(continued)

SECTION VI

SECTION VII

SECTION VII

(continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

SECTION I

GENERAL SYSTEM INFORMATION

A. INTRODUCTION

An exact definition of a digital computer would be quite detailed, lengthy, and difficult to fully formulate. This is due in part to the large and rapidly changing field of computer development. However, in a few words we might say simply that a digital computer is a device that counts. A general purpose digital computer is capable of performing arithmetic operations on information and is under the control of what is known as a program. A program is a sequence of logical instructions that a computer performs in order to obtain a desired result. The Automatic Sequence Control Calculator (an early computer) was an electro-mechanical computing device under the control of an external program. The external program consisted of a plug-board which was wired to sequence the computer through certain arithmetic and logical steps. After the plug-board was wired, it was inserted into the computer, and all operations were sequenced through this plug-board. However, using an externally programmed computer provided relatively little flexibility in the operations that the computer could perform. In the case of electronic devices, internal programmed steps can be inserted in such a manner as to perform virtually an infinite number of steps.

In addition to counting, a digital computer can perform many "logical" functions and decisions. Considering the computer as a data processing device, then improvising symbolism for the representation of data and instructions, the accomplishment of data processing in the electronic system is reduced to a process of manipulations, of direction and control of numbers and other characters as represented by electronic equivalents. Essentially, manipulation is accomplished by equipment design, utilizing the characteristics and capacities of various devices to move the data and instructions through the system.

The heart of the electronic data processing system is the digital computer, sometimes referred to as the central processing unit (CPU). The computer receives data and instructions, stores them in cells (memory) and calls them out of its memory as needed by the computer during a processing function. It can perform the arithmetic operations of addition, subtraction, multipli-

cation, and division and has further ability, found only to a very limited degree in mechanical systems, to make comparisons between numbers or other characters and take the action called for by the results. It also directs the processing operations within itself and controls the flow of input and output information. All these operations are performed at electronic speed. The electronic pulses and other electrical manifestations, acting as signals in the functional operation of the unit are symbolic representations of numbers and other characters.

B. DIGITAL COMPUTER COMPONENTS

The majority of digital computers on the market today are functionally similar and each consists of the basic components in Figure 1.

The heart of every computer is the Program Control Unit which directs and governs the operation of all other components. The Program Control Unit is usually constructed in such a manner as to obey coded instructions. In the early days of digital computers, a plug-board type of programming was used which was cumbersome and not very versatile. If it was desired to change the pattern of events in the computer, a technician had to change a number of wires on the plug-board. Therefore, human intervention was necessary for program variation and a great deal of processing time was lost.

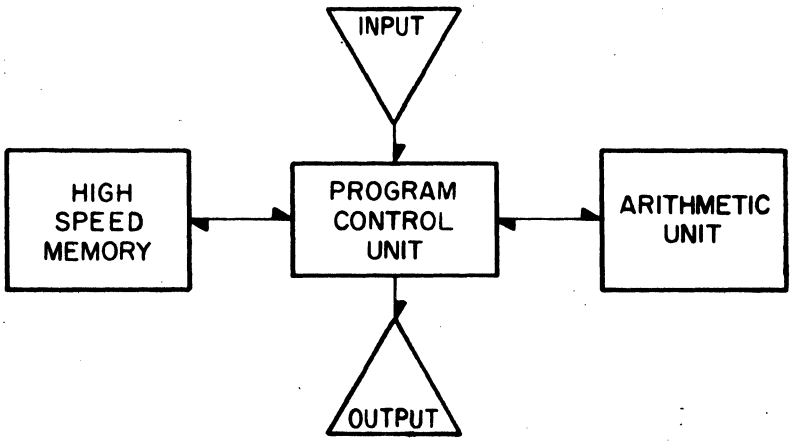


Figure 1 Basic Components of a Digital Computer

Internal wiring of the computer to follow a set number of coded operations increased the processing efficiency and versatility of the machine. The problem then became one of informing the computer which instruction or instructions to execute. A need for communication arose between man and machine; hence, the birth of programming.

1. Component Functions

Ultimately, any computer works with information in binary form, as this is really all that digital circuits are capable of. However, writing programs or entering data directly in binary form is extremely awkward. Therefore, the majority of modern computers use an internal code (based on octal, hexadecimal, or binary-coded-decimal) for representing all numeric and alphabetic characters and the more common symbols (such as $, #, @). The use of such a code requires that logic be provided within the computer to recognize the various characters of the code and correlate them with the binary configurations that the machine actually uses. The cost of this logic is more than compensated for by the flexibility and ease of operation that is gained by not having to use direct input binary. The RCA 301 uses a six-bit code that is based on binary-coded-decimal.

The High Speed Memory (HSM) like the Program Control Unit (PCU) has evolved to a high degree of efficiency. Early memory units used mercury delay tanks, cathrode ray tubes, or magnetic drums, all of which were serial accessible and quite slow. The modern day computers use magnetic core matrices which are extremely fast and random accessible. Many High Speed Memories are separate units which require only a command from the Program Control Unit before beginning a memory cycle, while other High Speed Memories rely on almost all commands from the program control and only perform a portion of a memory cycle at a time. (i.e., only read out or only regenerate.)

The logic involved with arithmetic operations is usually considered as a separate unit in most computers. Depending upon the size and speed of the machine, the Arithmetic Unit may operate on various multiples of characters, half words or words or even entire operands at one time. The means used in

carrying out these arithmetic functions may be by an adder or by table look-up.

2. Input/Output

Input-output control logic mates peripheral devices to the Program Control Unit and HSM. Data may be placed directly into memory from an external device, or buffered to permit more than one operation at the same time (simultaneity). Likewise, data may be sent from memory to a device directly or by way of a buffer. The timing for this transfer of information hinges on the peripheral device and is developed by the control logic.

Input information applied into the "reading" device must be expressed in, or converted into, the code of the computer and conversely, output information emitted from the computer through the "writing" device in the same code, must be converted to plain language and transcribed upon the output document. To do these things necessitates the use of input preparation equipment and output printing equipment.

3. Computer Operation

By means of equipment design, devices are provided which will bring the data and instructions to the computer, manipulate them there, and take out the results. Without troubling at the moment to understand how, it is sufficient to know that the devices within the computer can add, subtract, multiply, and divide, and make comparisons between numbers, alphabetic or other characters, and that they have the power of memory. They can retain data and instructions and introduce them into the manipulation routine as needed. They can even amend basic instructions as required in the course of processing. Further, the computer does all these things at electronic speed. How this is done, in terms of the methods used in directing the system to make the manipulations is important as a basic concept. These methods may be illustrated by an elementary analogy.

Assume that a newly hired office clerk is given the assignment of adding a column of figures using a desk calculator. Also, assume that this is the clerk's first encounter with a desk calculator. In order to perform this

addition, a list of instructions will be given to the clerk which details the procedures to be followed.

Step 1. Clear the keyboard and the accumulator.
Step 2. Insert first number into keyboard.
Step 3. Depress ADD button. (This will insert the first number into the accumulator.)
Step 4. Insert next number into keyboard.
Step 5. Depress ADD button. (This will cause the accumulation of the first two numbers.)
Step 6. Repeat Steps 4 and 5 until all the numbers to be added have been accumulated.

In this example, the desk calculator functioned as the arithmetic unit and storage device for results, and the clerk as the control unit and instruction and data storage device. As stated earlier, the control unit interprets the program of the problem and directs the processing operations. Instructions in pre-planned sequence are routed into the control unit. The internal logic design of the computer interprets and transmits the instructions to the system as directions for the processing of data. In order to perform this data processing, there must be a means for storing both the data and the instructions in this operation.

Memory devices provide a place to which data and instructions may be directed in the first instance and there held for introduction into a processing routine as required. In the processing which takes place within the computer, everything (all data and instructions) must be assigned to specific locations (addresses) within the system at all times; otherwise, the system would be in chaos.

Speed of processing (computer access time) is affected by the time required to find data and instructions as needed in the sequence of processing operations. Since all, and not merely part, of the data and instructions flow in and out of storage in the course of processing, it is obvious that storage volume and speed of access to storage are very important factors in deter-

mining the capabilities of the system.

The principal types of storage devices are transistor flip-flops, cathode ray tubes, magnetic cores, acoustic delay lines, magnetic drums, magnetic tapes, paper and magnetic cards.

The RCA 301, 501, 601 Systems use magnetic core storage. Magnetic cores are doughnut-shaped ferro-magnetic rings, usually about 1/16 inch in diameter. Bits of information are written into the cores by sending current through the centers of the cores. Each core stores only one bit of information at a time, hence, storage volume depends upon the number of cores used. Bits are read from the cores by sending current through the wires passing through the centers of the cores and transferring the resulting pulses to sensing lines linked to the main circuitry of the equipment.

Magnetic tapes are used extensively as secondary storage, as well as input and output media. When used as secondary storage, "bits" in primary storage are read out and written upon magnetic tape by the same reading-writing devices used in the input and output functions. Access to "bits" upon magnetic tape occurs as the tape moves mechanically past the reading head and, therefore, the rate of access is relatively slow as compared to the primary or "built-in" time of the computer.

Where the medium is a punched card, the method of representing data is by means of one or more holes in various positions in the vertical columns of the card. The same principle is used in respect to paper tape. The coding of a character takes the form of holes or inked dots across the width of the and in channels running the length of the tape. Paper tape is approximately one inch in width. Various codes are used, having from five to eight channels or positions. The recording of characters on magnetic tape takes the form of signals placed laterally on the tape by means of pulses from small electromagnets. Magnetic tapes vary from 1/4 inch to 3 inches in width. They are either metallic or plastic and contain a magnetizable material. Characters may be compactly stored upon magnetic tape -- up to 800 or more, to each inch of length.

Following these operations by the input preparation equipment, the input medium passes to the input reading devices, the first link in the chain of true electronic processing. These devices function under the direction of the control unit of the system, in accordance with the program of instructions. Reading occurs as the medium physically moves through the reading device, which translates the data and instructions expressed upon the medium in binary code into their electronic equivalents in the computer. The reading rate depends upon the type of medium used.

Reading devices consist of punched-card readers, paper-tape readers, and magnetic tape readers. Punched cards may be read at speeds up to 1500 a minute, or a maximum of about 2000 characters a second. Paper tapes may be read at speeds of 10 to 1,000 characters a second and magnetic tapes may be read at speeds of 360 to 120,000 characters a second.

The input medium may be introduced directly into the reading device. However, in some situations the original medium is converted into another, either to provide greater reading speed or to bring the medium into compatability with the system. Conversion is effected by special devices.

In addition to the devices previously mentioned as employed in the input cycle, a certain device is necessary in all electronic systems to provide access to the system. It is required to give the system its first instruction to start the processing routine and also, to permit intervention in unusual circumstances by direct insertion of correntional data or instructions. This device manually operated is a keyboard known as the console.

Output data are emitted from the control processing unit in binary coded form. The output function is essentially one of conversion, which may be direct or indirect. Direct conversion occurs when the output data are transferred to a medium which carries or incorporates the data in end-use form. This medium will be magnetic or paper tape, or punched cards, if the end use is storage of the output data for use as input in subsequent processing. If the end use is the production of report data, the medium will be the final report document.

Indirect conversion occurs when it is desirable, in connection with the preparation of the final report document, to hold all or part of the output data in intermediate storage (magnetic or paper tape or punched cards) in order to avoid reducing the overall speed of processing.

The computer emits information in the form of pulses and these must be translated into their electronic or other equivalents in the end-use medium. Even though the code of the end-use medium may still be in binary form, it is necessary to effect translation into the code scheme of the medium, that is, from pulses to magnetic tape and from pulses to holes or inked dots in the case of cards or paper tape. If the code of the end-use medium is not in binary form, which is the case in the final report document, the pulses must be translated into final report language.

Various conversion devices are interposed to make these translations. Where the translation of the pulses is made upon magnetic tape, a tape reading-writing device is used -- one identical with that employed for reading in connection with the input function. Where the translation is made upon paper tapes or cards, the device used is a punching or writing unit which is actuated by the pulses and internal circuitry to produce punched holes. Where the translation is made upon the final report document, the device is a printing unit, again actuated by pulses and internal circuitry, to print graphic characters in report language. To effect translations from intermediate storage to report documents, the device is the same printing unit, actuated in this case, however, by circuitry and by pulses created by the spots, dots or holes upon the tapes or cards to print characters in the language of the report.

C. INTRODUCTION TO THE 301 COMPUTER

The 301 Computer (Figure 2) has three basic racks of logic. The Program Control Unit occupies three quarters of Rack 2 and is governed by 41 different wired-in instructions. Being synchronous machine, the 301 contains a timing generator which is controlled by a one-megacycle oscillator and produces seven sequential one-microsecond time pulses per machine cycle. The

time pulse generator generates the basic timing for almost the entire system.

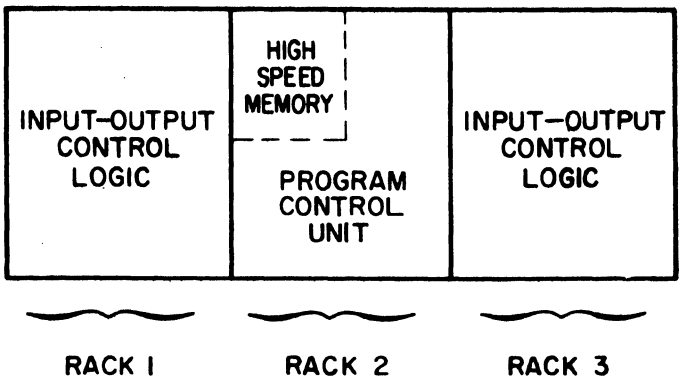


Figure 2 301 Rack Layout

1. 301 Component Specifications

The HSM and its associated logic occupies the remaining quarter of Rack 2. Depending upon which model is desired, the capacity of the memory is 10,000, 20,000, or 40,000 characters (Models 303A, 304A, and 305, respectively). However, the 301 10K or 20K memory cannot be expanded to 40K in the field. Two characters (a diad) are read out and regenerated during one memory cycle. This takes approximately seven microseconds. Each character contains six binary information bits and one parity bit for error checking purposes. Decimal addressing is used in the 301, and each address is constructed of four characters. For the 10K memory, addresses range from 0000 to 9999, but for the 20K and 40K memories, a special character is necessary in the most significant digit position. The second ten thousand from -000 to R999, and the fourth ten thousand from "000 to Z999. (See page I-6 of the 301 Programmers' Reference Manual.)

The 301 does not have a specific arithmetic unit since no adder exists; the table look-up technique is used instead. However, what logic there is, which is peculiar to the arithmetic operations, exists in the PCU. Only decimal addition and decimal subtraction can be executed in the 301. Three other

operations performed by logic are also available, but multiplication and division must be performed by programming.

The control logic which communicates with the peripheral gear can be found in racks one and three. Two racks are used only if the system is sufficiently large. Each piece of input-output equipment has its own panel of control logic of from three to eleven rows on one side of a rack. A common bus connects the control modules to the Processor but otherwise the modules are independent of one another. Should a customer wish to extend his system, he simply buys the necessary control logic and device, bolts in the modules, and connects the cables. No common input-output buffer exists. Each piece of control logic has effectively its own buffer, hence, several different operations may be proceeding at the same time.

2. Simultaneous and Record File Modes

Available at optional cost along with the peripheral equipment are two modes of logic called Simultaneous and Record File. Each mode is almost another complete Program Control Unit in itself except that they both use the Processor timing pulses. Specific instructions are designed to work only in these modes with input-output equipment. The Record File Mode is restricted to Record File Units and Communications only, however. The Simultaneous and Record File modes are located in racks one and three with the peripheral control logic. If a system contains three modes - Normal, Simultaneous and Record File - then three different instructions can be executed simultaneously while time sharing memory. In addition, there are three independent operations; namely, rewinding magnetic tape, paper advancing on the Line Printer, and band selecting on the Record File. These operations need only be initiated and they will proceed independently of any computer modes. Thus it is possible to perform six different operations at the same time on one 301 System.

The Power Supply for the 301 Computer System is found in one or two additional racks, depending on the size of the system.

D. THE RCA 301 SYSTEM

The RCA 301 Electronic Data Processing System is composed of a Processor Model 303/304 (Program Control Unit and High Speed Memory) and the following "on-line" peripheral equipment:

a. Paper Tape Reader
b. Paper Tape Punch
c. On-Line Printer
d. Card Punch
e. Card Reader
f. 581 Tape Station
g. Hi-Data Tape Group
h. Record File
i. Check Sorter-Reader
j. R.A.C.E. Unit
k. Interrogating Typewriter
l. Monitor Printer
m. Various Communications Equipment

The Burrough's Check Sorter-Reader is not considered part of the 301 System, but optional control logic is available to make it compatible with RCA equipment. The above peripheral equipment is termed "on-line" because it is tied-in directly to the Processor and receives its control levels from the Processor. Another term used is "input-output equipment". Certain pieces of peripheral equipment are used as input devices while others are used for output. The magnetic tape units (581 and Hi-Data Tape Group) as well as the RACE Unit can be used as either input or output devices with the Processor. (See Figure 3).

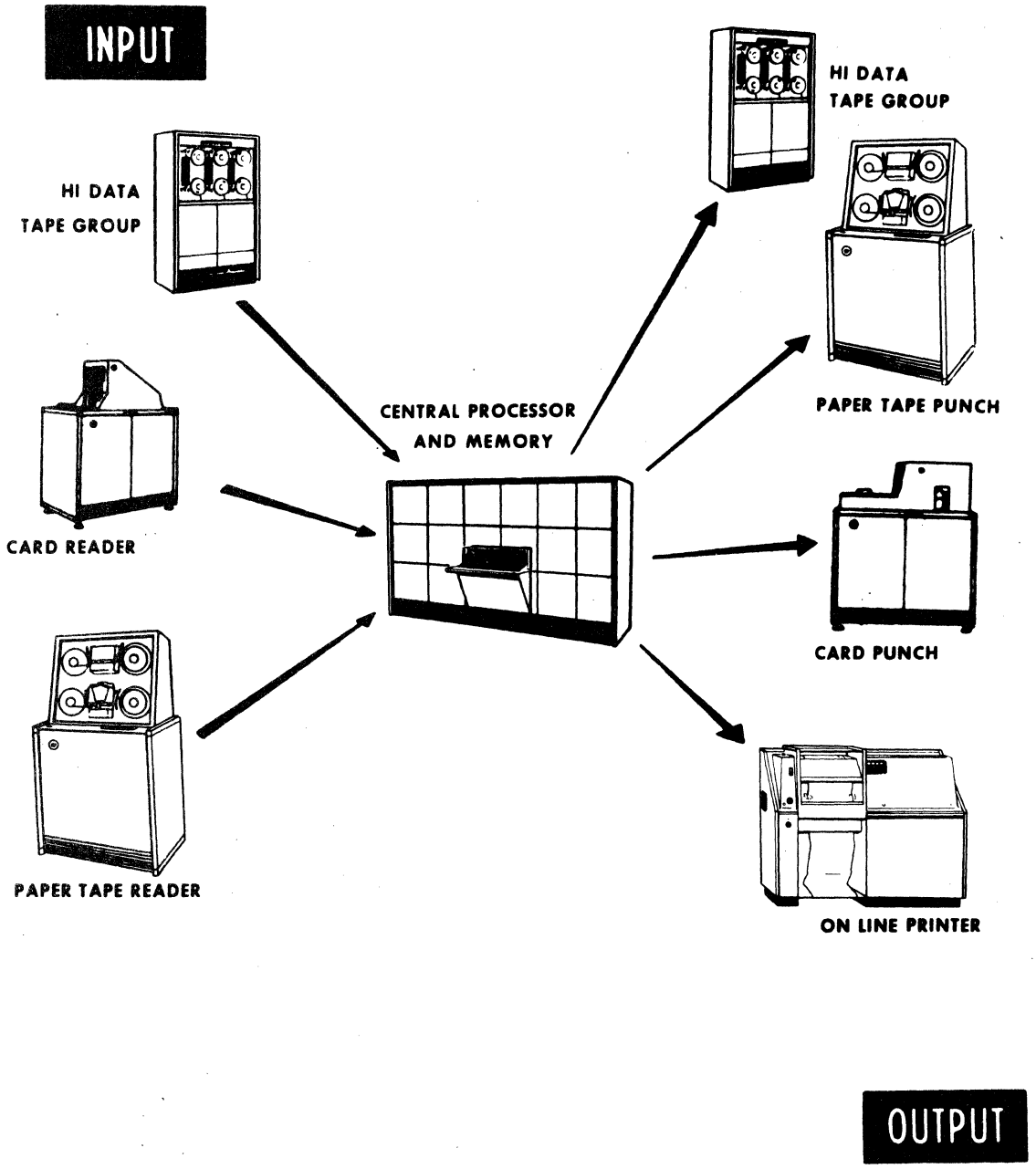


Figure 3 RCA 301 Input-Output Equipment

The Program Control Unit and High Speed Memory occupy one rack of logic while the input-output control panels occupy two additional racks of logic -- one on either side of the Processor rack. (See Figure 4). Included with the Processor rack is the console for the Computer.

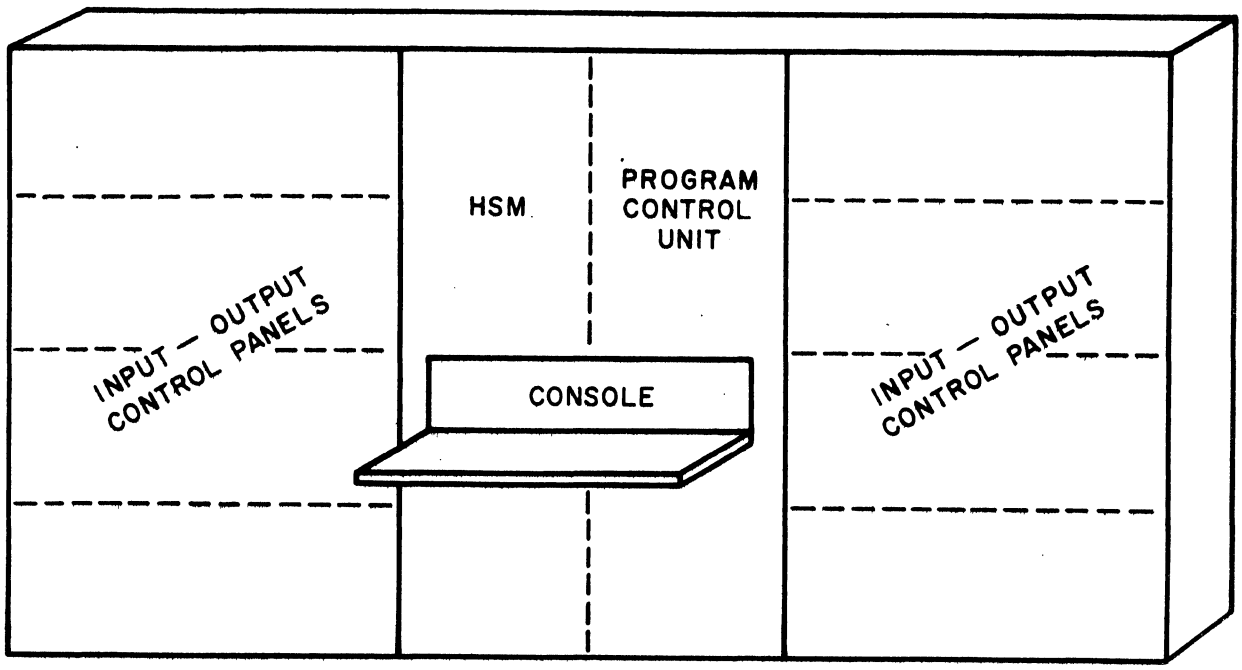


Figure 4 The Three Basic Racks of the 301 Computer

1. System Elements Description

Processor, Model 303/304 - The Processor is a general purpose, digital, stored program, transistorized machine consisting of the following integrated units: High Speed Memory, Program Control, Console Panel and Power Supply. The High Speed Memory is a random access, magnetic core device which provides storage and work area for programs and data.

The memory capacity is either 10,000 (Model 303) or 20,000 (Model 304) alpha-numeric characters. The Program Control executes the instructions of the program stored in the High Speed Memory and performs the required accuracy checks. The Console Panel provides for complete monitoring of the Computer

operation. Adequate indicators and controls are provided on the panel to initiate normal computer operation and to facilitate program checkout and maintenance.

2. Paper Tape Reader and Punch, Model 321 - Paper Tape input and output is provided by the Paper Tape Reader and the Paper Tape Punch. The Paper Tape Reader provides for the entry of 7-channel paper tape at a rate of 100 characters per second. Accuracy of data on paper tape is assured by parity checks. The Paper Tape Punch produces paper tape data from the Computer at 100 characters per second. The accuracy of the output is assured by an echo check.

3. On-Line Printer, Model 333 - The On-Line Printer provides high-speed printed output for the RCA 301 System. The printer, operating completely under program control, has a line capacity of 1000 lines per minute, and paper can be advanced in excess of 70 lines per second. Variations in format, as well as complete editing, are under the control of the stored program. Paper advance is independent of the normal data processing activity.

4. Card Punch, Model 334 - The Card Punch automatically translates RCA 301 characters from memory to 80-column card code and punches the information into cards. The output rate is 100 cards per minute. The card punching unit includes an automatic card reading station for automatic accuracy checks. Information is edited and re-arranged under program control.

5. Card Reader, Model 323 - The Card Reader reads information into memory from 80-column card code. Cards are read at the rate of 600 cards per minute. Reading and editing is under complete control of the stored program. Two reading stations are provided for automatic accuracy checks.

6. Tape Station, Model 581 - To provide data compatability with other RCA Data Processing Systems, Tape Adapters can be added to the RCA 301 System. The Tape Adapter permits the RCA 301 Computer to read from or write to a Model 581 Tape Station. The 581 Tape Station reads or writes information on magnetic tape at a density of 333 characters to the inch, while moving

tape at 100 inches per second.

7. Hi-Data Tape Group, Model 381 - The Hi-Data Tape Group is composed of a cluster of six tape decks with a common set of control, power supply, and switching circuits. Each tape deck responds to programmed instructions by reading and writing information on magnetic tape. Reading is performed in either the forward or reverse direction. Data is recorded in seven channels, with a density of 333 characters per inch. Tape speed is 30 inches per second. Tape rewind at 90 inches per second is independent of the normal data processing ability.

8. Record File, Model 361 - Record files, with a capacity of over 4.6 million characters each, are available with each RCA 301 System. The Record File contains 128 magnetic discs, and each side of one of these discs is divided into two bands with each band containing ten cells. Each cell has a storage capacity of 900 characters.

Data may be transferred between the core memory and the Record File. The contents of as many as ten cells can be transferred with one instruction. The transfer rate between the Record File and the High Speed Memory is 2500 characters per second.

9. Check Sorter-Reader, Model 101 - The Check Sorter-Reader processes documents at the maximum rate of 1560 per minute. Documents are sorted to one of 13 pockets. The Sorter-Reader can be operated in the sort (local) mode or in the external mode controlled by the 301 Processor.

SUMMARY OF PERIPHERAL EQUIPMENT PERFORMANCE

Record File.....	Storage of over 4.6 million alpha-numeric characters, transfer rate of 2500 characters per second.
Hi-Data Tape.....	333 characters per inch, tape speed of 30 inches per second.

581 Tape Station (with Tape Adapter).....333 characters per inch, tape speed of 100 inches per second.

Card Reader.....600 cards per minute.

Card Punch.....100 cards per minute.

On-Line Printer.....1000 lines per minute, 120 characters per line.

Paper Tape Reader.....100 characters per second.

Paper Tape Punch.....100 characters per second.

Check Sorter Reader.....1560 documents per minute.

E. TYPICAL SYSTEM APPLICATION

The flexibility of the RCA 301 System makes it ideal for a widely diversified range of applications. To illustrate the efficiency and versatility of the RCA 301, an example has been chosen of its application to common distribution reports and the weekly payroll. Figure 5 illustrates the system configuration for this application.

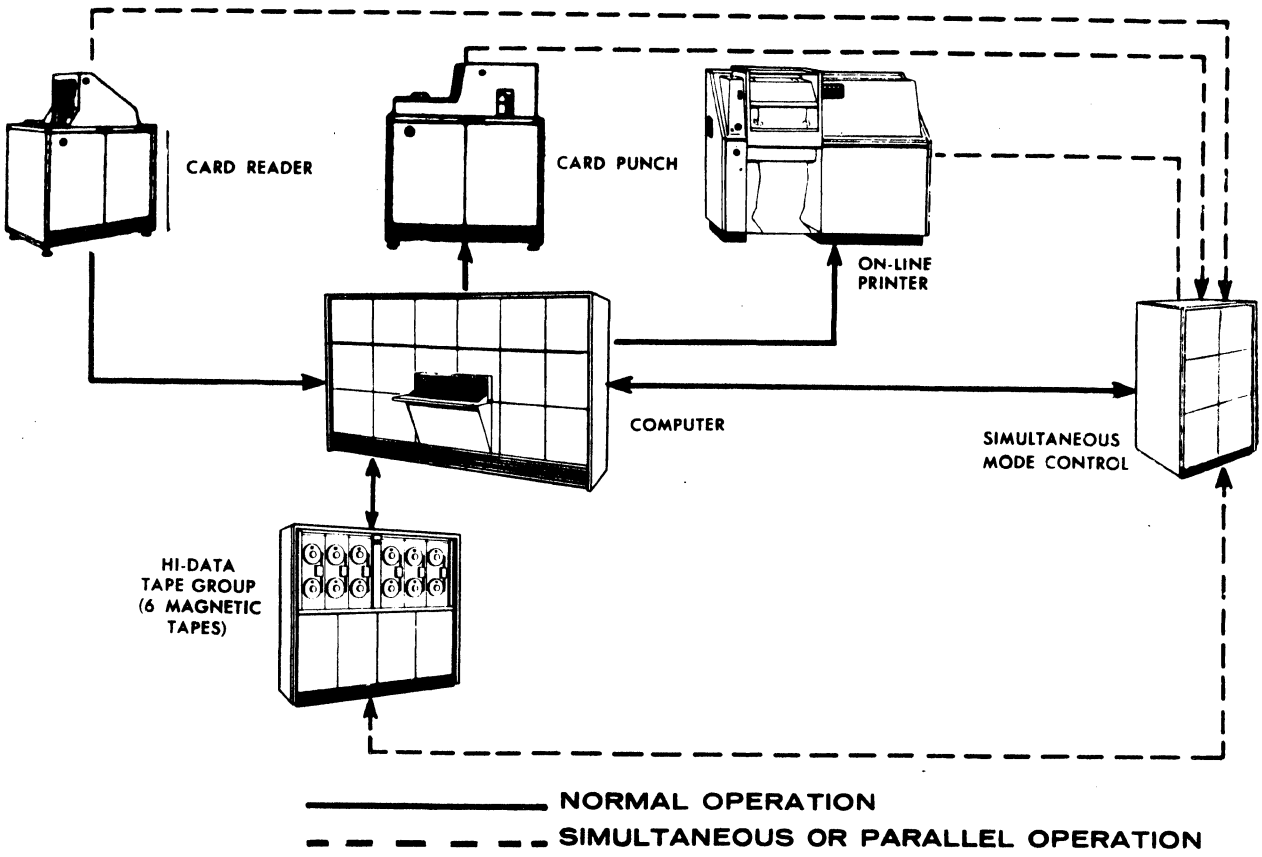


Figure 5 RCA 301 System for Labor Distribution-Payroll Application

Inputs to the daily run are time-clock cards and labor tickets for straight time or piece work. A report of labor distribution by department is prepared for cost accounting as output.

As a by-product of the daily run, the input is prepared for the payroll and stored within the 301 System in an optimum fashion. The weekly payroll run produces the paycheck for each employee. It also creates a payroll register and deduction register.

The daily input of clock cards, straight-time cards, and piece work cards is

read in batches and sorted within the high speed memory. (See Figure 6).

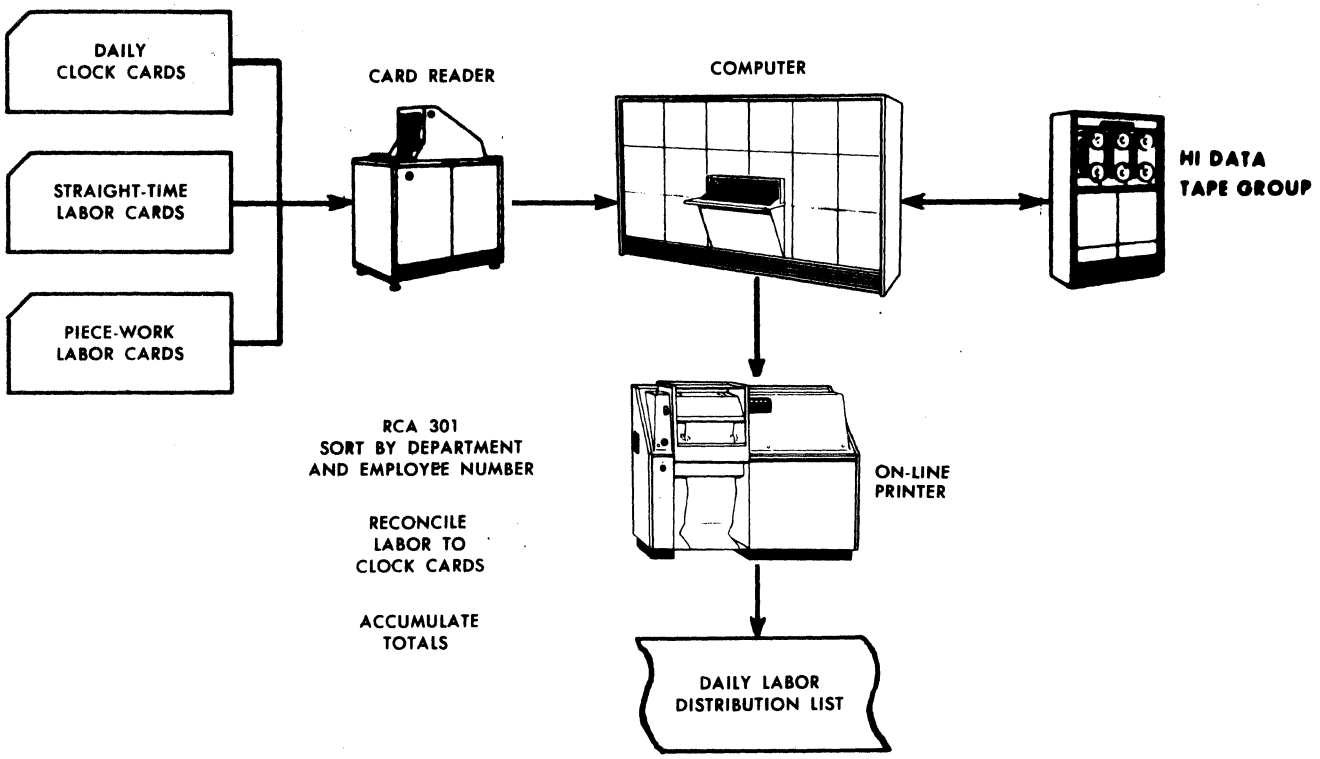


Figure 6 Daily Labor Distribution to Departments, Using RCA 301 System

Off-line sorting equipment is not required. The data, sorted by department and employee number, is sent to the Tape Station. This procedure is repeated until the input data is exhausted. The data stored on the Tape Station is then transferred in sections to the core memory and merged in the required sequence. A reconciliation of clock cards to labor cards is performed for each employee. Concurrently, employee data is grouped and sent to the Tape Station to be used as input for the weekly payroll. The department totals are accumulated and labor distribution information edited and printed out.

The Payroll, which is prepared weekly, utilizes the data provided by the daily labor distribution run (Figure 7). The RCA 301's ability to store the data in an expedient manner on the Tape Station requires only simple merging in the high speed core memory.

The master employee file is run against the employee information as each section is in high speed memory. At the same time, the payroll and deduction registers are created on magnetic tape and the payroll checks are prepared on the On-Line Printer. Eliminated in this phase is the necessity of running a transaction tape against the master file tape. This means that full advantage is taken of simultaneity. The payroll and deduction registers may be printed after the checks.

This application shows the capabilities and power of the RCA 301 System. All data manipulation, sorting, intermediate storage of data, processing, and printing is accomplished within the System. System efficiency is reflected in the time required to perform these operations. For a 1,000 employee firm, each labor distribution function requires only one-half hour, and the weekly payroll run takes less than 15 minutes.

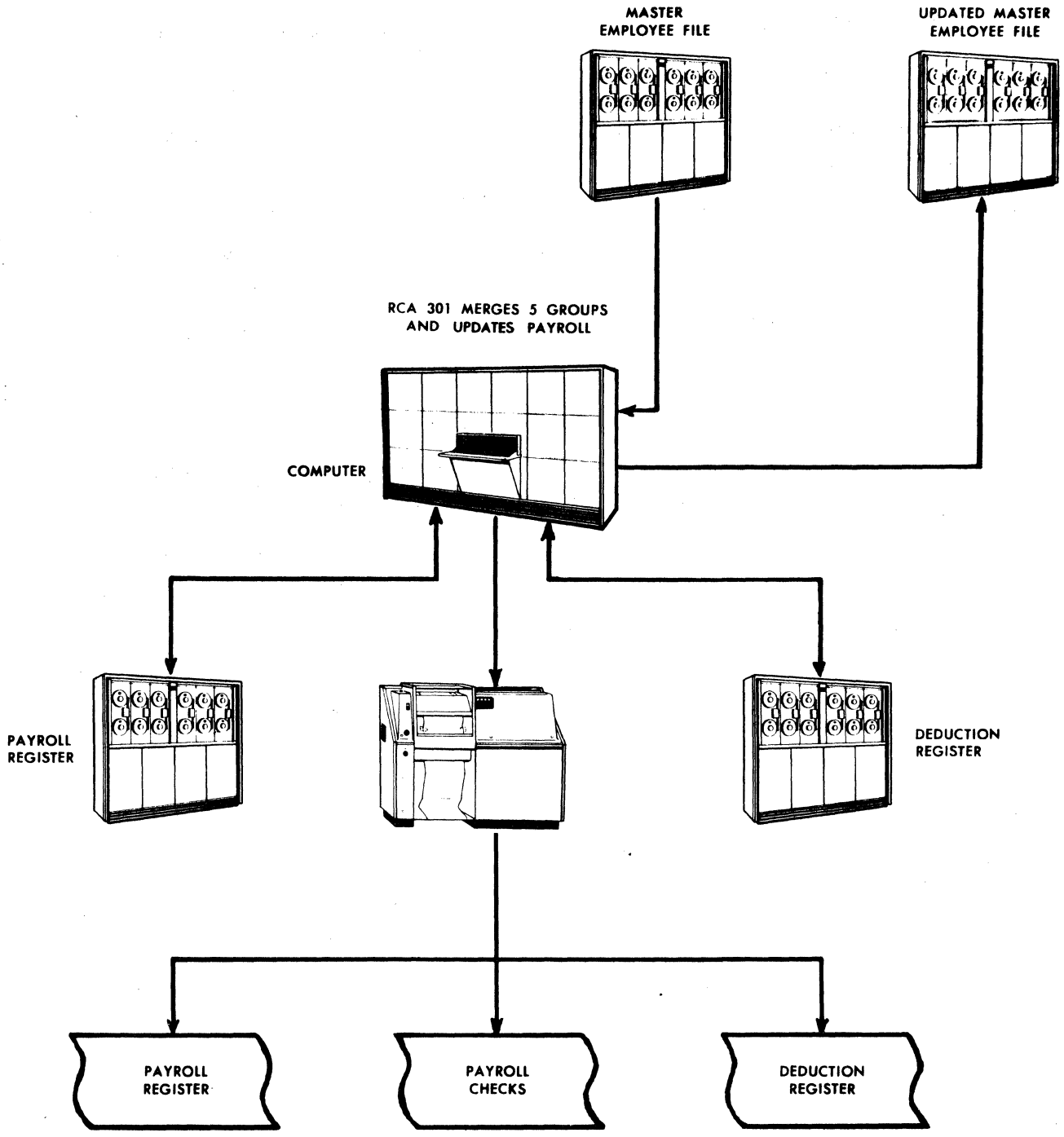


Figure 7 Weekly Payroll Flow, Using RCA 301 System

F. THE ROLE OF INSTRUCTIONS

Nothing of significance can be accomplished in the 301 Computer unless it is done by instruction. The computer can only obey the 41 wired-in instructions that exist in the Program Control Unit. Basically speaking, the Computer executes only one instruction at a time; hence, every program must be solved step-by-step.

A 301 instruction consists of ten characters in coded form which, when decoded, will dictate a specific course of action. If, for example, it was desired to add two given numbers, the computer must be told (1) the operation that is to be performed and (2) the number of digits contained in each operand. The first character (Operation Code) of the instruction will indicate the type of operation. In the case of an Add instruction, this would be a "plus" symbol (+). The second character (N character) of the instruction will determine the number of digits contained in each operand. Therefore, when adding two five-digit numbers, the Operation Code sould be + and the N Character would be 5. It should be noted that the function of the N Character is not the same for all instructions, e.g., in a Tape Write instruction it determines which tape station is to be utilized.

Almost all data, as well as the program, are stored in the High Speed Memory (HSM). Every character contained in memory (including the characters that make up the instructions) has its own memory address. An address is simply four decimal digits which represent a location or a series of magnetic cores in memory. These cores can hold information in binary bit form. If it is desired to retrieve or insert information into these cores, they must be identified by an address. In a 10K memory, addresses range from 0000 to 9999. In a 20K memory, addresses range from 0000 to I999, and in a 40K from 0000 to Z999. See page I-6 of the Programmers' Reference Manual.

There are four basic groups of instructions, namely:

a. Data Handling Instructions.
b. Decision and Control Instructions.
c. Arithmetic Instructions.
d. Input/Output Instructions.

In an Add instruction, for example, the address of each operand must be specified. These two addresses are called the A Address and the B Address, respectively. The A Address will designate the location of the least significant digit (LSD) of the augend and the sum. The B Address will designate the location of the LSD of the addend.

All 301 instructions follow a similar instruction format of:

O N AAAA BBBB

where:

O = Operation Code
N = N Character
AAAA = 4-digit A Address
BBBB = 4-digit B Address

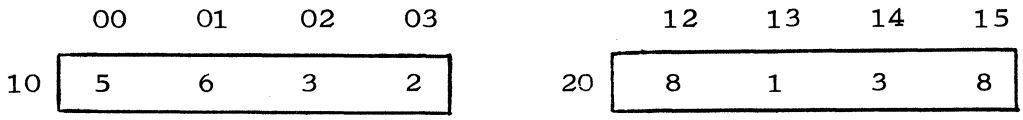
The N character can be used for many different functions, depending on the instruction, such as specifying length of operands, specifying certain options, etc.

A typical Add instruction, where it is desired to add two three-digit numbers would be as follows:

\+ 3 1002 2015

This instruction states that we will add two operands consisting of three digits each. The first addition will take place with characters from each operand found at locations 1002 and 2015. The second addition will be upon characters found at locations 1001 and 2014 and the third addition will be upon characters found at locations 1000 and 2013. The result of each addition will be placed back in memory at locations 1002, 1001, and 1000.

Memory contents are often illustrated as follows:



HSM Before Execution of Instruction

HSM After Execution of Instruction

The two digits on the left of each box denote the first two digits of the address. The last two digits of each address are shown above the box. Within the box are the actual contents of memory. Thus, after executing the Add instruction, memory would contain the characters shown above.

1. The Role of Status Levels

The instruction itself is broken down into subdivisions called status levels. (See Figure 8). Each status level exists for seven microseconds. There are twenty-four status levels, each of which accomplishes one basic function (such as loading a register, accessing HSM, etc.). Each instruction is made up of a certain combination of status levels, in a certain order, to produce a desired result. A status level can be described as a gating level which opens paths in the Computer Logic for information flow. Usually a status level is named for the register with which it primarily works. For example, a P1 status level works with the P register; while an A3 status level works with the A register. Some status levels such as Sense Input-Output (SIO) are named after their function.

2. The Role of Time Pulses

The status level, in turn, is broken down into time pulses. In almost every synchronous machine, the memory cycle dictates the timing for internal operations. Since the 301 memory cycle consumes approximately seven microseconds, seven one-microsecond pulses called time pulses (TP0, TP1, TP2, TP3, TP4, TP5, TP6) comprise the basic timing of the Computer. Almost every function carried out in the 301 is done at a specific time pulse of a given status level, during a particular instruction.

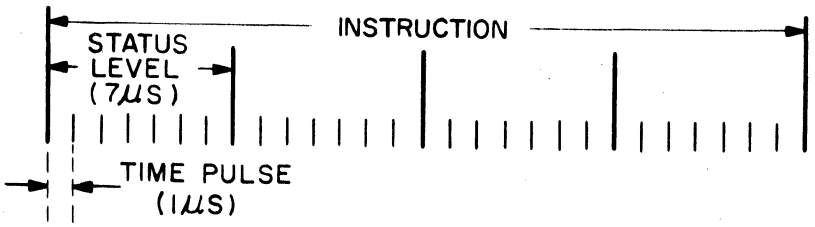


Figure 8 Timing Breakdown of a typical 301 Instruction

An instruction is identified by its operation code. The operation code is the major control level which in most cases, exists for the entire execution of the instruction. It is the operation code which determines what status levels are necessary for carrying out the instruction, and it is the time pulses which determine when these status levels will occur. All the logic responsible for generating operation codes, status levels, and time pulses exists in the Program Control Unit.

3. Processor Block Diagram

A block diagram of the Basic Processor is shown in Figure 9. Note that a common bus, capable of transmitting four 301 characters in parallel, connects to all registers. Bus lines 2 and 3 each have seven isolated wires to handle seven bits per character, but Bus lines 0 and 1 have six and five wires, respectively, for Models 303A and 304A. Model 305 has seven wires for Bus line 0. The bus lines which have less than seven wires are those used to handle addresses only. Most 301 addresses are four decimal digits and all decimal digits have no "one" bits in the 2^4 and 2^5 position (301 code). The 2^4 bit in Model 304 and the 2^4 and 2^5 bit in Model 305 are used in the most significant digits of addresses over ten thousand.

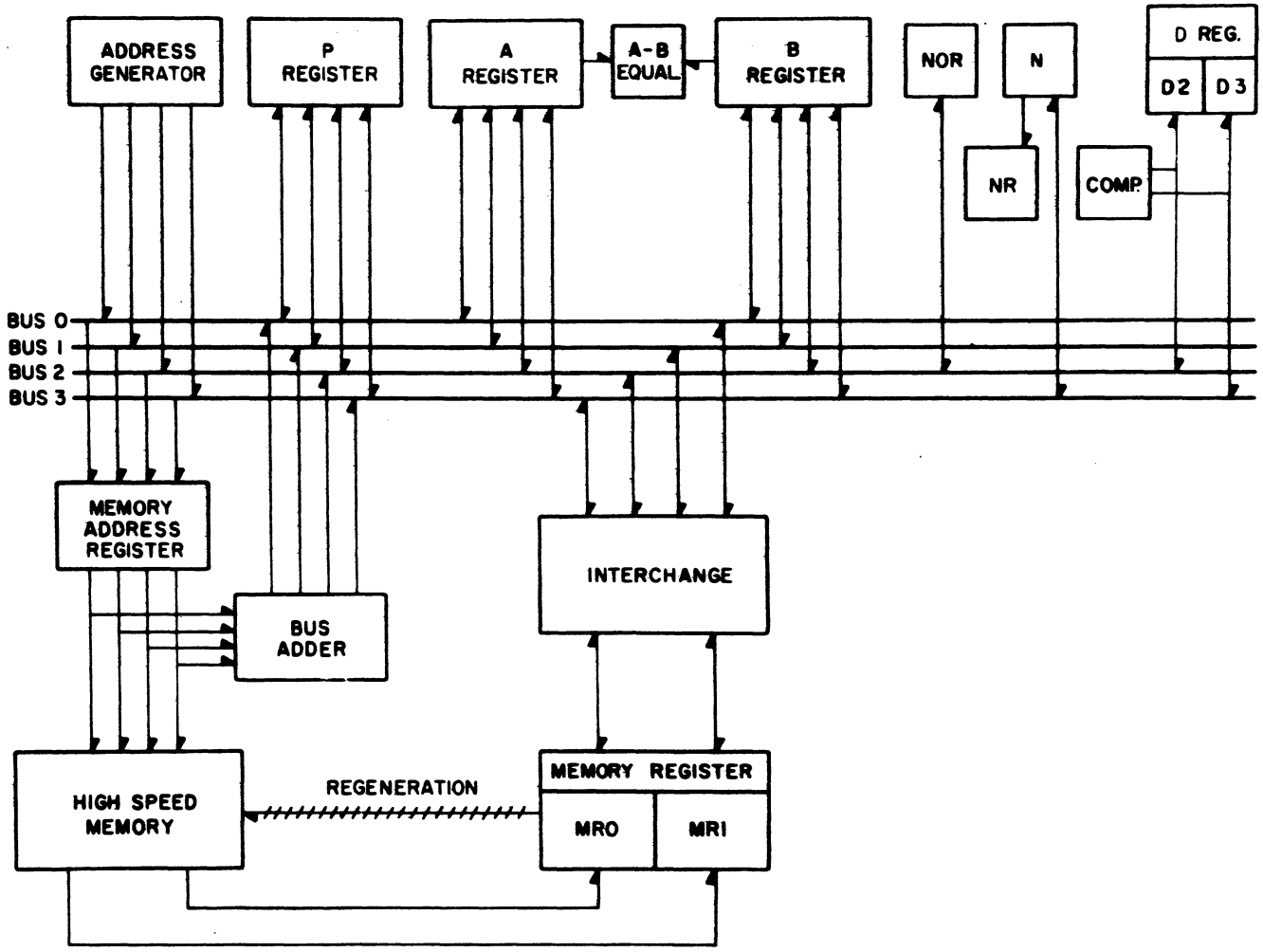


Figure 9 Processor Block Diagram

Therefore, Bus 0 which carries the MSD characters of addresses needs six or seven wires (depending on the model) while Bus 1 needs only five wires - no wire necessary for 2^4 or 2^5.

In reality, there are two four-character buses in the 301. One is called the positive bus and the other the negative bus. Inverters separate the two buses such that any signal on one is carried on the other, but in opposite polarity (See Figure 10).

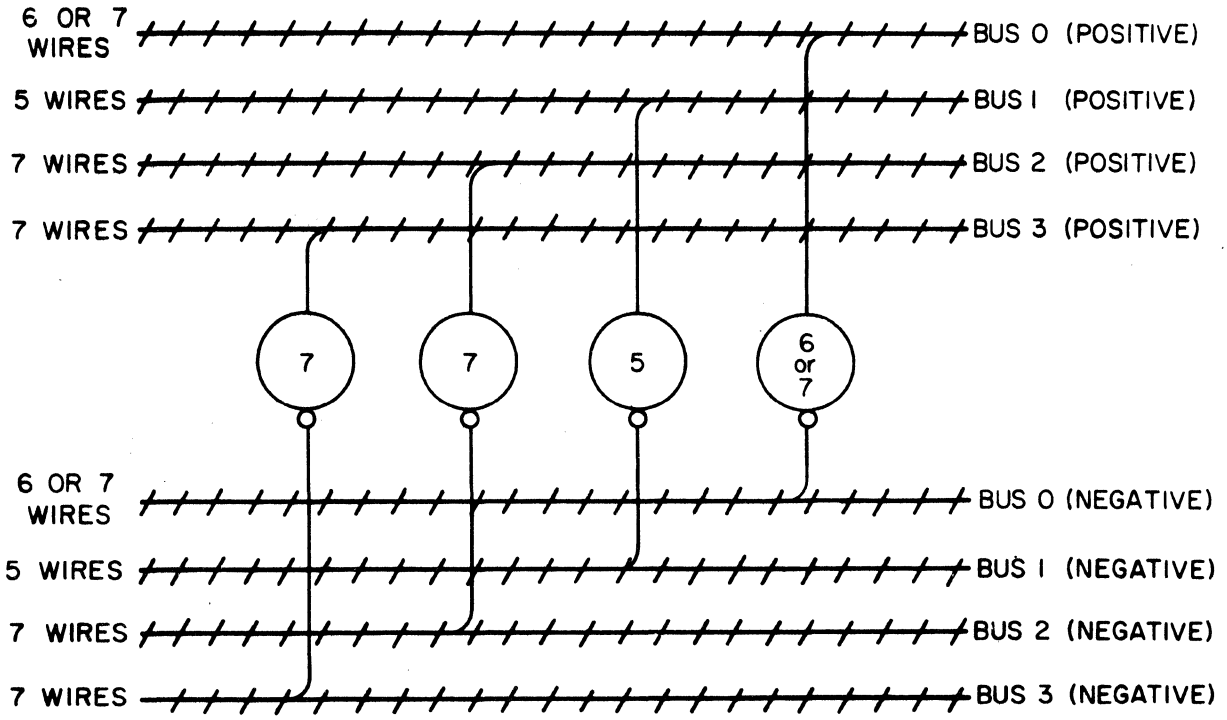


Figure 10 Positive and Negative Bus Lines

A +6.5 volt signal on the positive bus indicates a "one" bit and a zero volt signal a "zero" bit. Conversely, on the negative bus a zero volt signal is a "one" bit and a +6.5 volt signal is a "zero" bit. The reason for a double set of bus lines is that the basic 301 AND gate requires low inputs to produce a high output. If "one" bits on the positive bus were required to prime the AND gates, the signals would all have to be inverted. Therefore, to

economize on hardware, two sets of bus lines exist; the positive bus to receive outputs from gates, and the negative bus to provide inputs to gates.

4. Gating Information From Register To Register

To illustrate transmission of data from one register to another, an example involving the N register and the D register will be used. (See Figure 11).

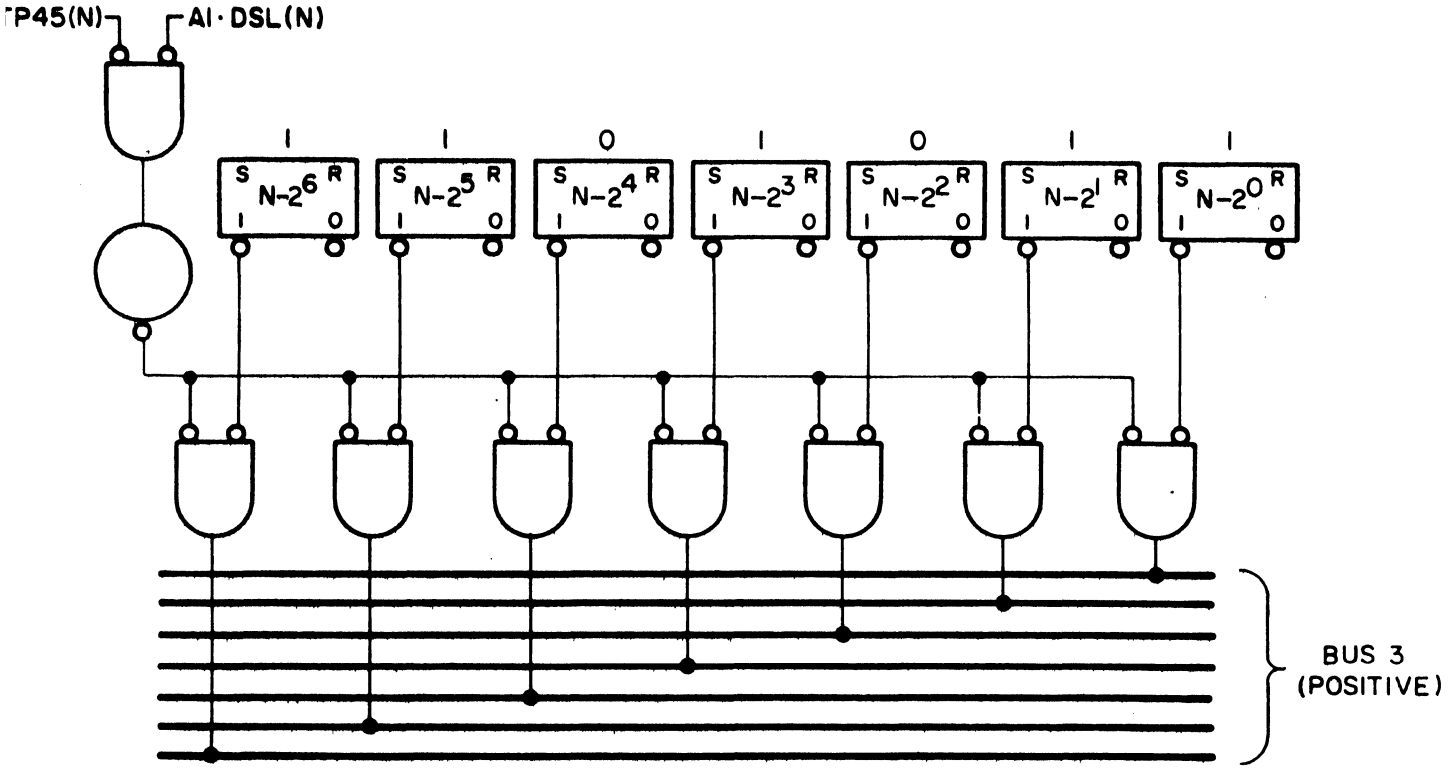


Figure 11 Gating Out of N Register

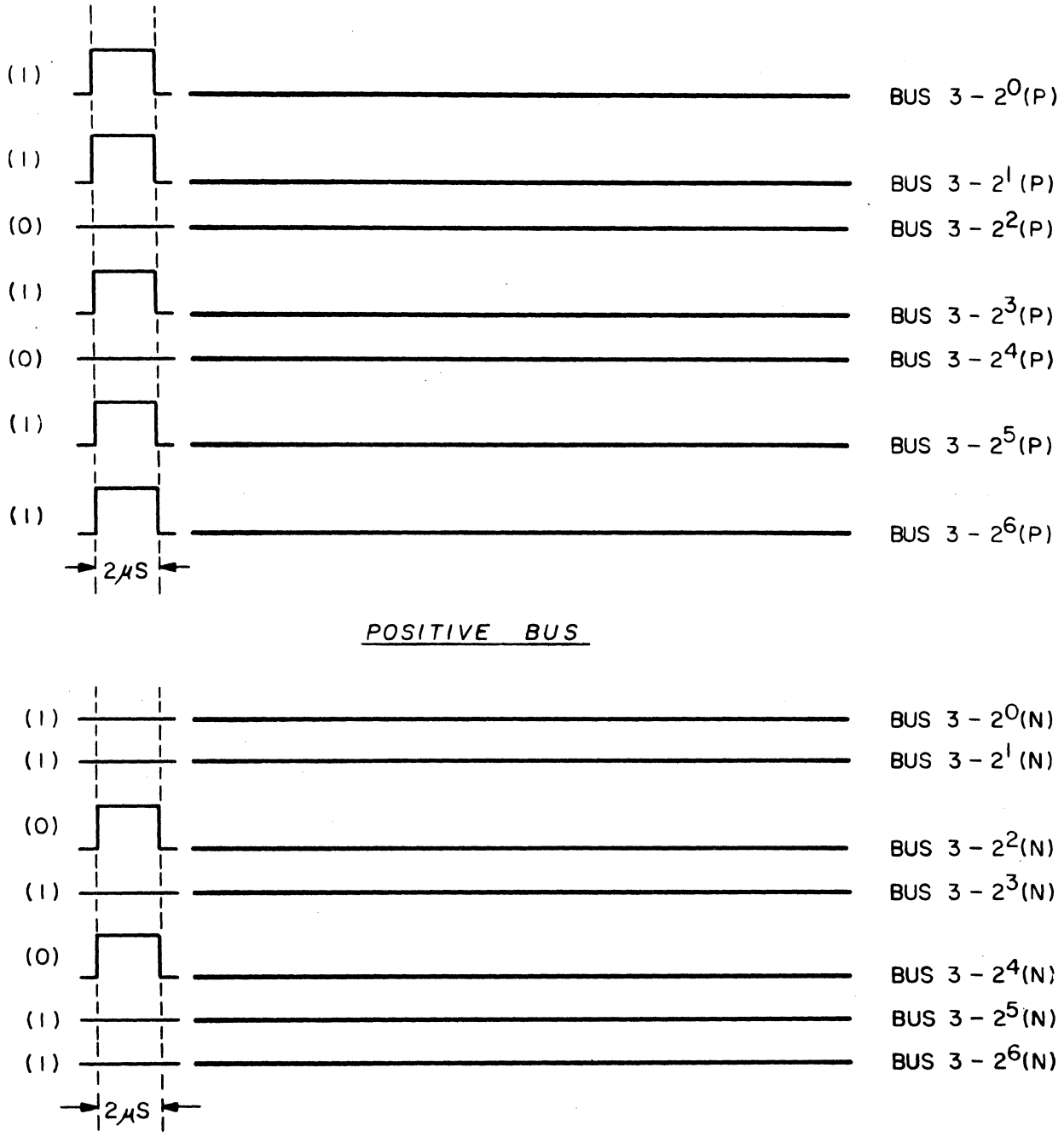


Figure 12 Pulse Train for Dollar Symbol ($)

Assume that a $ is in the N register. Since the bit configuration for a $ is 1 101 011 including parity, the N register 2^6, 2^5, 2^3, 2^1 and 2^0 flip-flops would all be set, while 2^4 and 2^2 would be reset. At TP4 time of an A1 status level, during a Transfer Data by Symbol Left instruction (DSL), the upper leftmost AND gate would be primed and produce a high output. This output would exist for two microseconds (duration of TP4 and TP5) and after inversion would prime all of the AND gates shown below the N flip-flops. Any flip-flops in the set state would also be placing a low level on the aforementioned AND gates. Thus, for each flip-flop that is holding a one bit, its corresponding AND gate would be primed and produce a high onto the respecti e bus line. This high would be a two microsecond signal. For the character $, the pulse train would be as shown in Figure 12.

While the character $ is on the bus during TP4 and TP5, the D3 register is prepared for receiving this character. At TP4 of an A1 status level during a DSL instruction, the D3 register is reset and at TP5 (See Figure 13) the character is gated into D3 from Bus 3 negative. It should be noted that whenever the 301 transfers data onto the bus, a two-microsecond gating pulse is the minimum used, and in some cases it is a 3 microsecond pulse. Gating from the bus into a register is usually done by a 1 microsecond pulse.

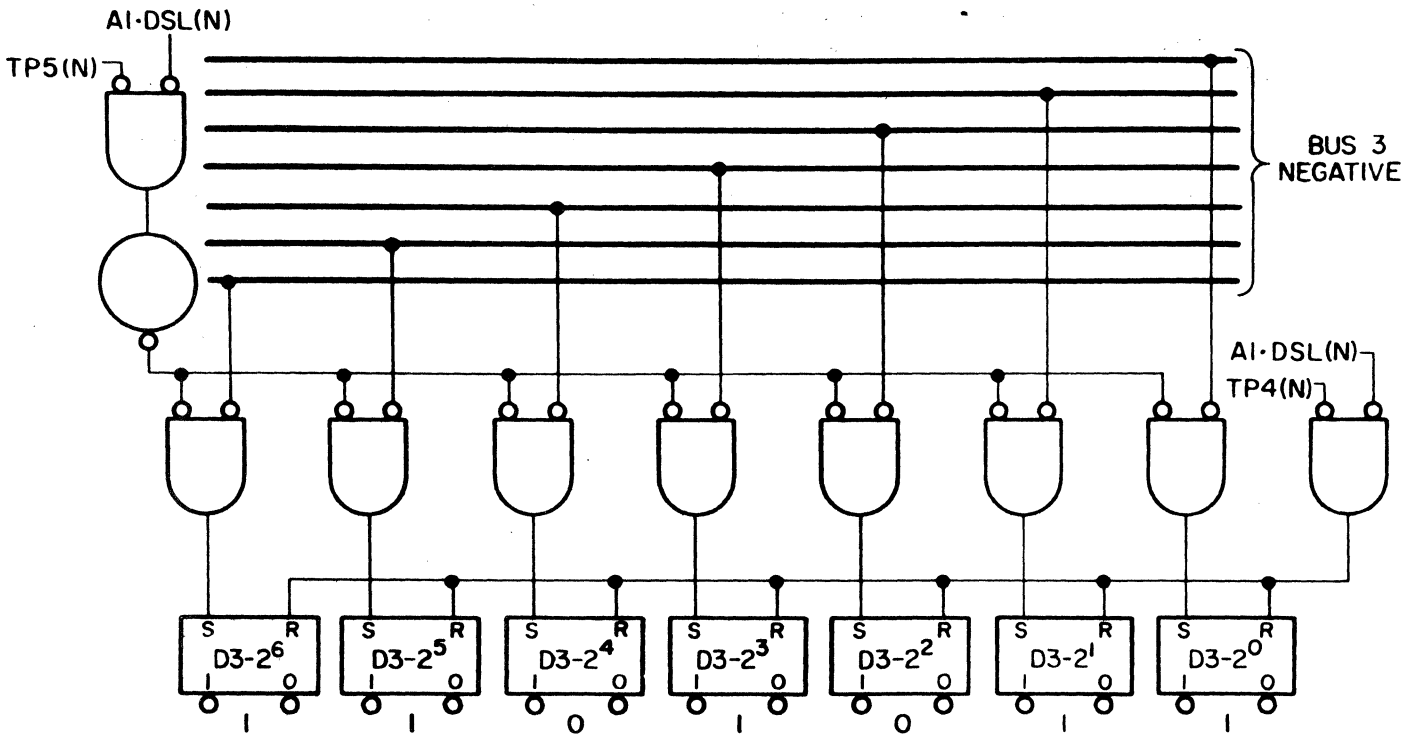


Figure 13 Gating Into the D3 Register

5. The 301 Memory Cycle

Being familiar with 301 instruction format, one can note that the instruction registers in the block-diagram are the Normal Operation Code Register (NOR), the N Register, and the A and B Registers. The other addressable register in the Basic Processor is the P register, which is used for program control. The D register is used primarily for temporary storage, while the NR register is only used during the repeat instruction. The address generator is responsible for creating a special address whenever needed, such as in the STA process.

The bus adder is used to increment or decrement addresses found in the Memory Address Register (MAR), and the interchange is a series of AND gates and OR gates which govern the gating between the bus lines and the Memory Register (MR). Any character coming from memory or going to memory must pass through the MR.

The MAR and HSM as well as some other logic are located in that quarter of rack 2 housing memory. Everything else on the block diagram can be found in the remaining three quarters of rack 2 designated as the Program Control Unit.

The function of the MAR is to hold the address while it is decoded and the memory cores are accessed. It is the basic structure of the 301 to access two locations, or a diad, for each address sent to the MAR. A diad can be remembered as the even address on the left and the odd address on the right. The locations shown below are divided into diads.

32	33	34	35	36	37	38	39	40	41

84

Effectively, the computer ignores the 2^0 bit of the Least Significant Digit (LSD) of an address when addressing memory. For example, in the address 8437, the LSD is a seven or binary 000 111 excluding parity. If the 2^0 bit is ignored, the character could also be 000 110, which represents 6 to the 301. Hence, the diad is composed of locations 8436 and 8437.

Usually the status level alone is sufficient to request a memory cycle. Occasionally the operation code is also required. Nevertheless, the computer will not generate a memory cycle unless the Program Control Unit commands one. A signal known as command level is sent to memory logic at the beginning of the status level which is to use memory, and when coupled with TP1, the command level begins the memory timing generator sequence. The memory timing generator produces a series of control signals which carry out the memory cycle. These signals are not the same as TP's, but are so regulated that the memory cycle will be completed by TP-6, when the status level is completed. The first phase of the cycle is read-out, and the last phase is write-in (Regenerate). Once a memory cycle is started, it is always carried out to completion. When a diad is read, the contents of 14 cores are automatically destroyed; the two characters found there are sent to the MR. However, these two characters must be gated into the MR or they will be lost completely. The reason for the gating process is to permit the insertion of

new information into the memory cores. Whatever exists in the MR during the write-in phase will be regenerated into the selected 14 cores. Hence, if both of the characters are permitted to reach the MR during read-out, then the same two characters will be regenerated and memory would be effectively unchanged. If, however, only one character is permitted to reach the MR during read-out, and a new character is inserted from the bus into the MR prior to write-in time, memory would contain one new character after regeneration. Two new characters can be written into a diad if the original contents of the diad are not permitted to reach the MR and the new characters are inserted into the MR before write-in time.

Obviously something must control the gating into the MR during read-out. Once again, the status level and the instruction make the primary decision as to whether or not a new character will be written into memory. Many times the computer simply wants to transfer information from memory to register and permits both characters during read-out. The other decision as to which character to permit of the two, if not both, is made by the address in the MAR. The address being odd or even specifies what character is gated from memory to the MR and also designates what position of the MR will receive the new character from the bus.

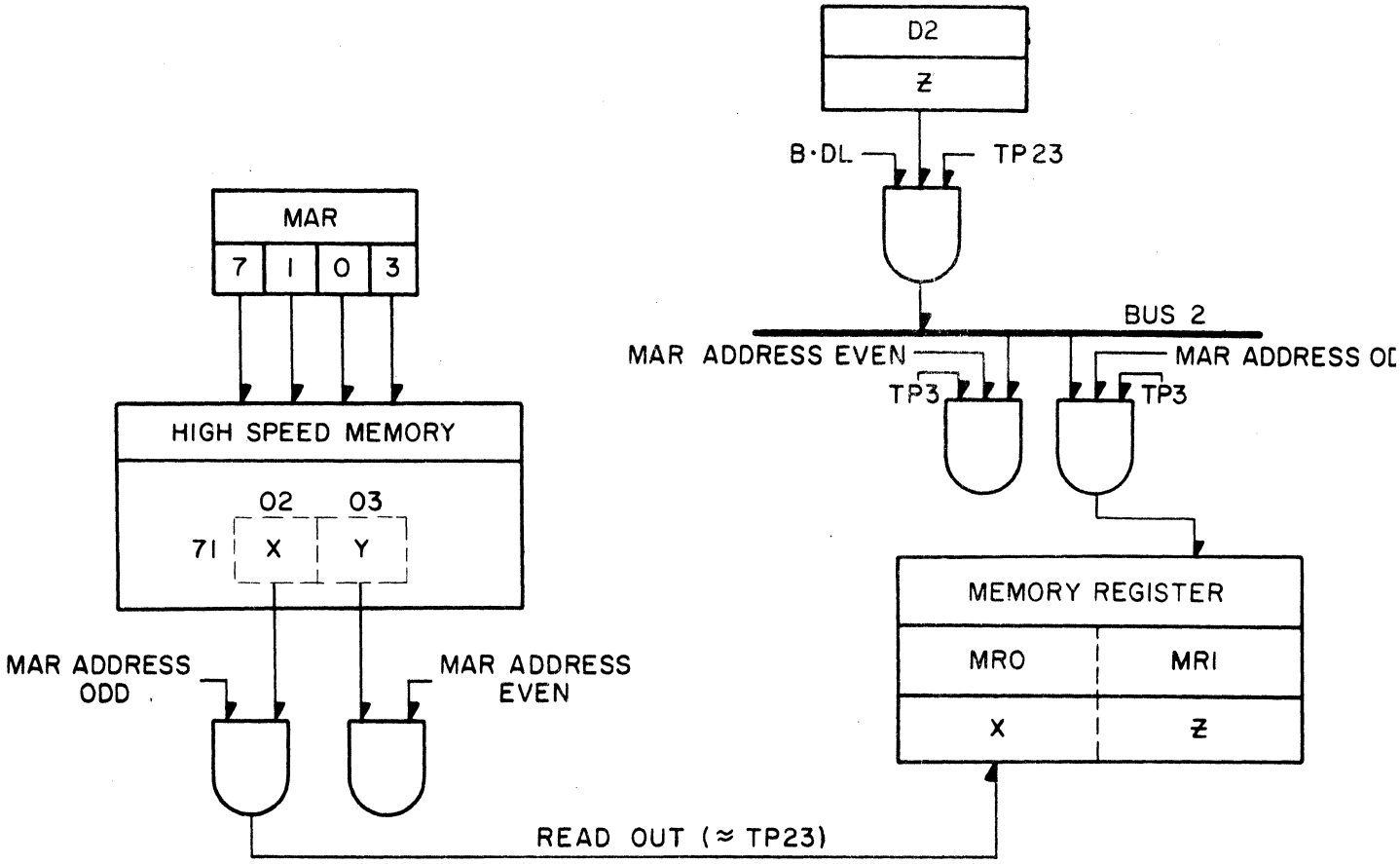
Because of the timing generator in memory, all memory cycles perform read-out and regeneration at approximately the same time. Read-out occurs around TP2 and TP3 of a status level and regeneration at about TP4 and TP5. The MAR is reset at every TPO and the MR at every TP1, in preparation for the new memory cycle.

Assume that a B status level of a Transfer Data Left instruction is about to be executed. During this status level a character will be transferred from the D2 register to memory as specified by the B address.

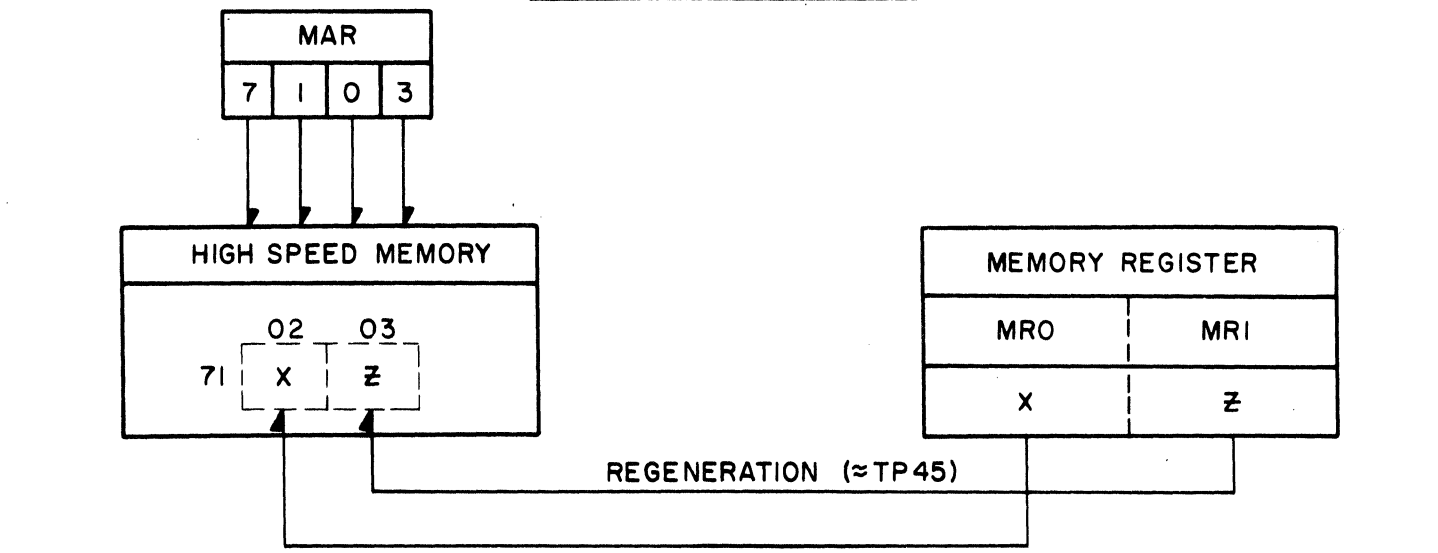
At TPO of the B status level the MAR is reset and at TP1 the B address is gated into the MAR. Also at TP1 the MR is reset. If the B address is 7103, the diad selected would be 7102 and 7103. Since the B status level is writing a new character to memory at the address in the MAR, only one character

would be read-out at TP23 into the MR. This character would come from 7102 and would be placed in MR0 because the MAR address is odd. The character from the D2 register would be on Bus 2 by TP2 time and would be gated into MR1 at TP3 time through the interchange. Regeneration would occur at TP45 time and the character that was held in D2 would now exist in memory at 7103, while 7102 would be unchanged.

The discussion above is illustrated in Figure 14 by using X and Y as original contents of memory, and Z as the character in the D2 register.



READ OUT CYCLE



WRITE IN CYCLE

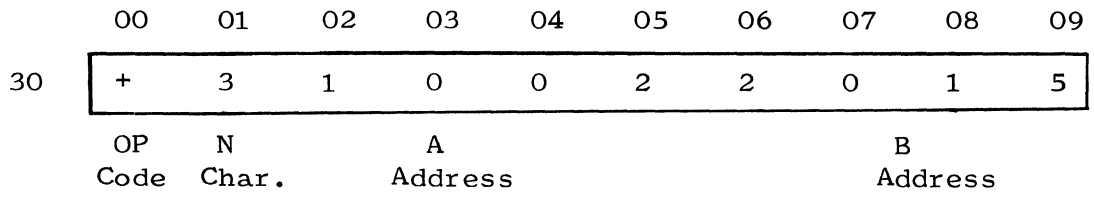
Figure 14 Typical Memory Cycle

G. STATICIZING

Before discussing individual instructions, it should be pointed out that, although the actual instruction is stored in memory as part of a program, the computer must place the entire instruction in registers for execution.

The process of bringing an instruction out of memory and distributing it to appropriate registers is known as staticizing. Each instruction in the program must be staticized before the computer can execute it. The Operation Code is placed in the Normal Operation Register (NOR), the N Character in the N Register and the A and B addresses in the A and B Registers, respectively.

Since an instruction consists of ten characters (an Operation Code, an N Character and two addresses of four characters each), ten locations in memory are needed to accommodate an instruction. Usually, the Operation Code is placed at an address ending in zero. If this is true of the first instruction in a program, all succeeding instructions will have their Operation Codes at an address ending in zero. For example, an instruction might be placed in memory beginning at address 3000. The individual characters of this instruction would then have the addresses as shown:



A second instruction would start at address 3010 with the Operation Code in that location. A third instruction in this program would have its Operation Code in location 3020 and so on.

A program can be placed anywhere in memory except between addresses 0000 and 0225 which are reserved for certain operations. See page E-1 of Programmers' Reference Manual.

In order to staticize an instruction, the computer must know the address of that instruction. In the 301, this function is controlled by the P Register.

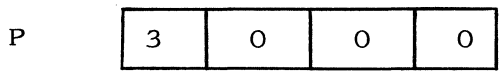
The P Register must be supplied with the address of the first instruction in a program (the address of the first Operation Code). After being given an address, the P Register will proceed to increment this address during the process of staticizing and will thus locate or keep track of each instruction in the program.

The 301 memory cycle reads two characters at a time from memory in seven microseconds. This pair of characters is called a diad. A diad consists of two consecutive memory locations with the even address on the left and the odd address on the right. (The computer actually ignores the 2^0 bit of the LSD of the address in the process of addressing memory.) Thus, to staticize an instruction, the computer must go through five memory cycles (two characters per cycle times five equals 10 characters). When staticizing an instruction, in order to keep track of the addresses of the individual diads, the P Register must be incremented by two, five times. For this reason, the OP code must be in an even address; that is, the OP code and N count must be in the same diad.

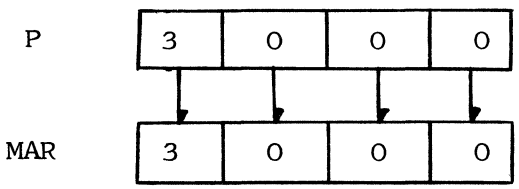
Thus, it can be stated that the purpose of the P Register is to hold the address of the next instruction to be executed.

To clarify the concept of staticizing, the following illustration is given.

After the program has been loaded into memory, from paper or magnetic tape (this is accomplished by an instruction manually set up on the console), the address of the first instruction in the program must be manually placed in the P Register.

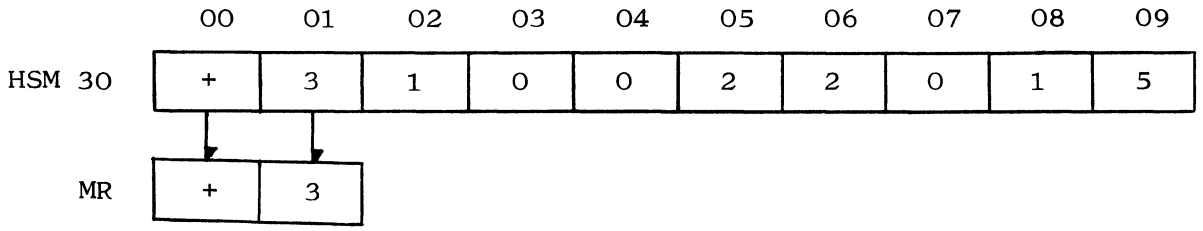


Once the start button is depressed, the computer will gate the contents of the P Register to the MAR (memory Address Register).

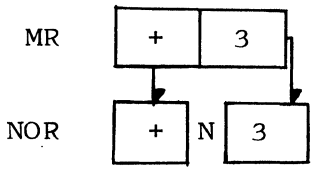


The MAR will hold the address while decoding logic selects the locations in memory. The output of the MAR also feeds the Bus Adder which modifies the address by +2.

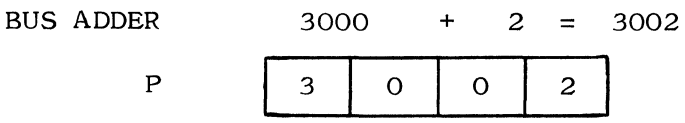
The diad which is addressed by the contents of the MAR is brought from memory to the MR (Memory Register). During this memory cycle, the diad will be regenerated into the original locations in memory.



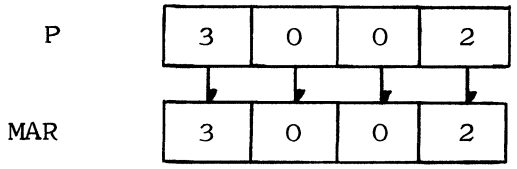
The contents of the MR are then sent to their respective registers, the NOR and N.

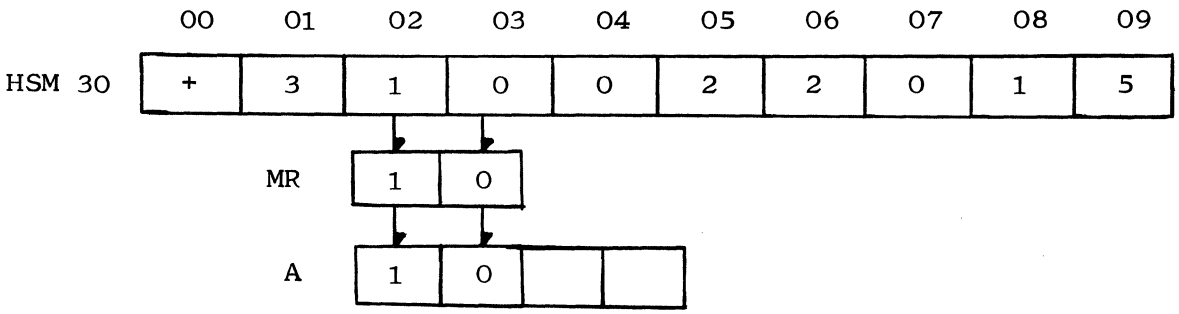


Meanwhile, the Bus Adder has finished modifying the P Address by +2. The P Register is reset and the new address is gated back from the Bus Adder to the P Register.

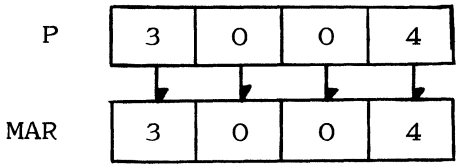


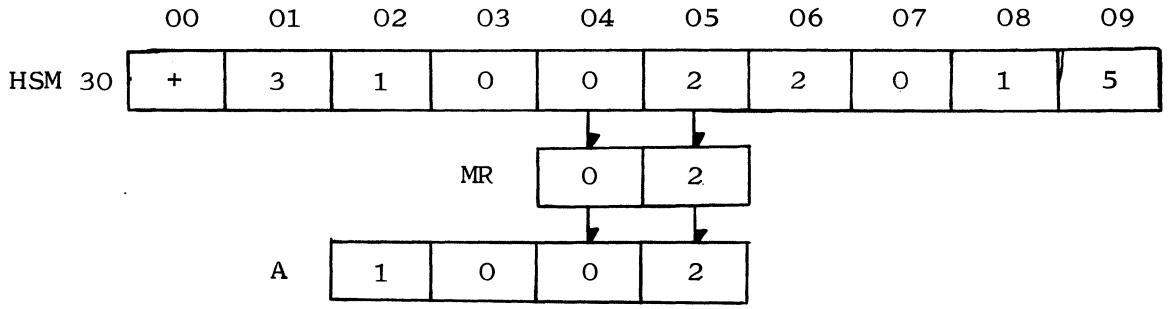
The P Register is now addressing the second diad of the instruction. A similar process is repeated four more times to staticize the entire instruction.



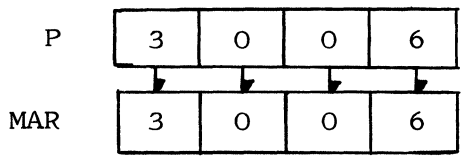


BUS ADDER 3002 + 2 = 3004

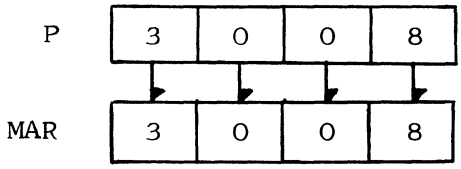


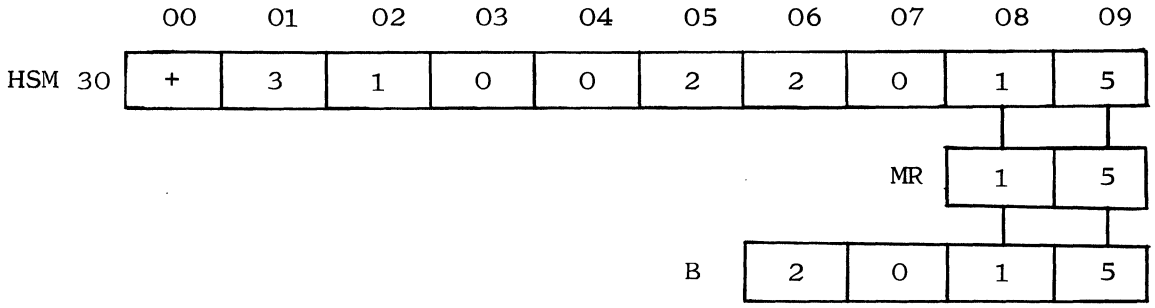


BUS ADDER 3004 + 2 = 3006

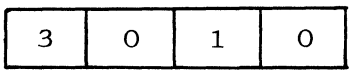


BUS ADDER 3006 + 2 = 3008

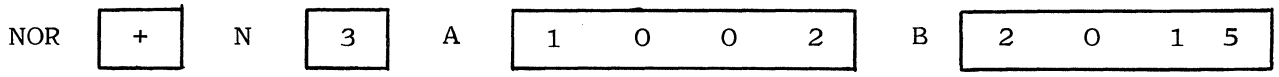




BUS ADDER 3008 + 2 = 3010



Note that P is addressing the operation code of the second instruction in the program when staticizing is completed. Final register contents would be as follows:



1. P1 Thru P5 Status Flow

Staticizing is accomplished by five status levels, P1 through P5. Each one staticizes one diad or two characters of the instruction. Five of them, then, will staticize ten characters or the complete instruction. They are designated "P" because they work with the P Register.

The following are charts that show what happens at each TP of the status levels, and the pertinent portions of the processor block diagram.

	P1
TP01	P Address is sent to Bus.
TP1	Memory Register is reset. Contents of Bus are sent to MAR and the Bus Adder where 2 is added to the address. A command level is sent to memory and the memory timing sequence begins. The MAR address is decoded and the diad selected.
TP2	P Register is reset. Read out from memory has begun.
TP23	Both characters read out are permitted to reach the MR. The modified address in the Bus Adder is gated back to Bus.
TP3	Read out is complete. The contents of the Bus are placed in the P register.
TP4	The NOR and N registers are reset.
TP456	The contents of MRO are gated thru the interchange onto Bus 2 and the contents of MR1 onto Bus 3.
TP456	Regeneration of the characters in the MR occurs.
TP5	The contents of Bus 2 are gated into the NOR and the contents of Bus 3 are gated into the N register.
TP6	A P2 status level is automatically selected. Reset MAR.

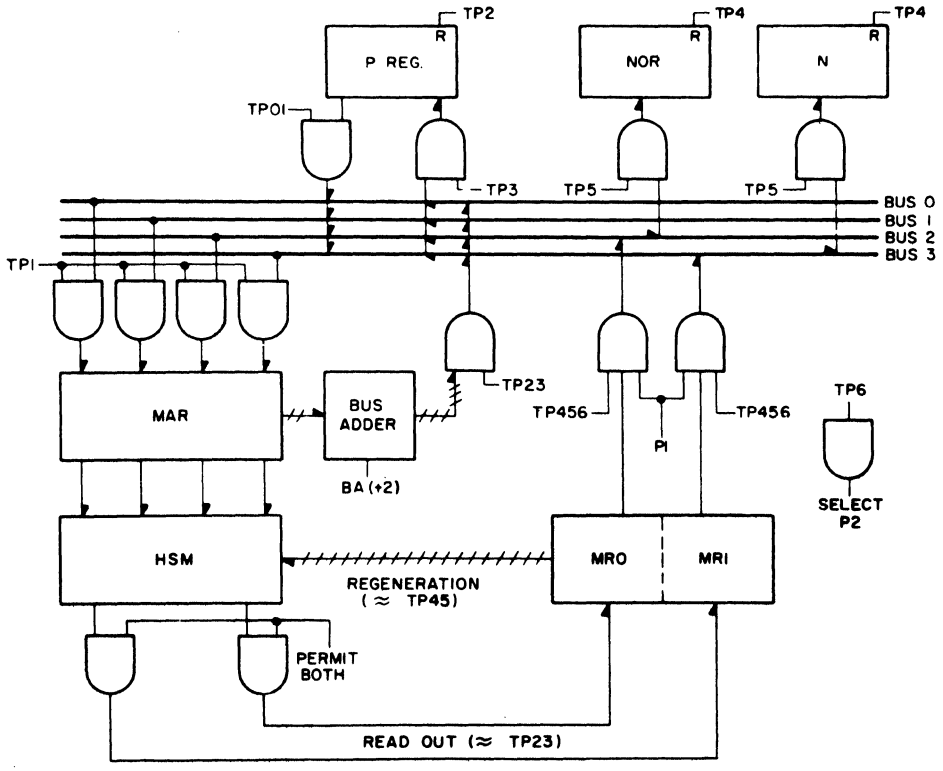


Figure 15 P1 Status Level

The main purpose of the P1 status level, therefore, is to bring out the operation code and the N character, and place them in the NOR and N registers, respectively. Note that the address in P register is now 2 greater and that a new diad is to be addressed.

The P2 status level automatically follows the P1 with the following events occurring.

	P2
TP0	MAR is reset.
TP01	P address is sent to Bus.
TP1	MR is reset. Contents of Bus are sent to MAR and Bus Adder where 2 is added. A Command Level is generated and sent to HSM. The MAR address is decoded and the diad is selected.
TP2	P register is reset. Readout begins.
TP23	Both characters readout are permitted to reach MR. The modified address in the Bus Adder is gated onto the Bus.
TP3	Readout is complete. Contents of Bus are gated into P register.
TP4	A register is reset.
TP456	The contents of MR0 are gated thru interchange onto Bus 0 and contents of MR1 onto Bus 1. Regeneration takes place.
TP5	The contents of Bus 0 and Bus 1 are gated into A register.
TP6	A P3 status level is automatically selected.

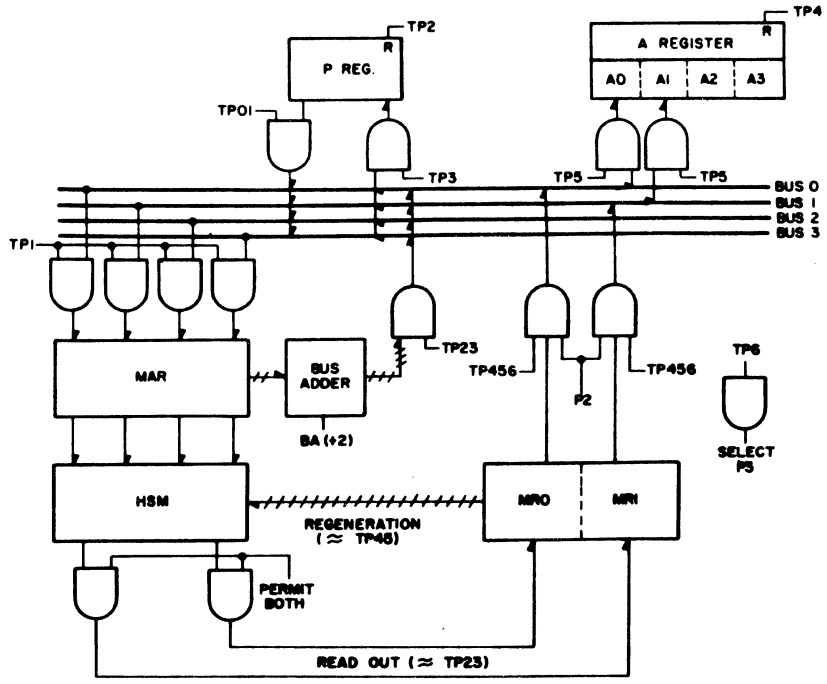


Figure 16 P2 Status Level

The function of the P2 status level then is to bring out the first two characters of the A address and place them in the A register.

The P3 status level follows immediately and brings out the last two characters of the A address. Note that the A register is not reset at TP4 of P3 as in P2, since the computer would destroy the first two characters brought out if this were permitted.

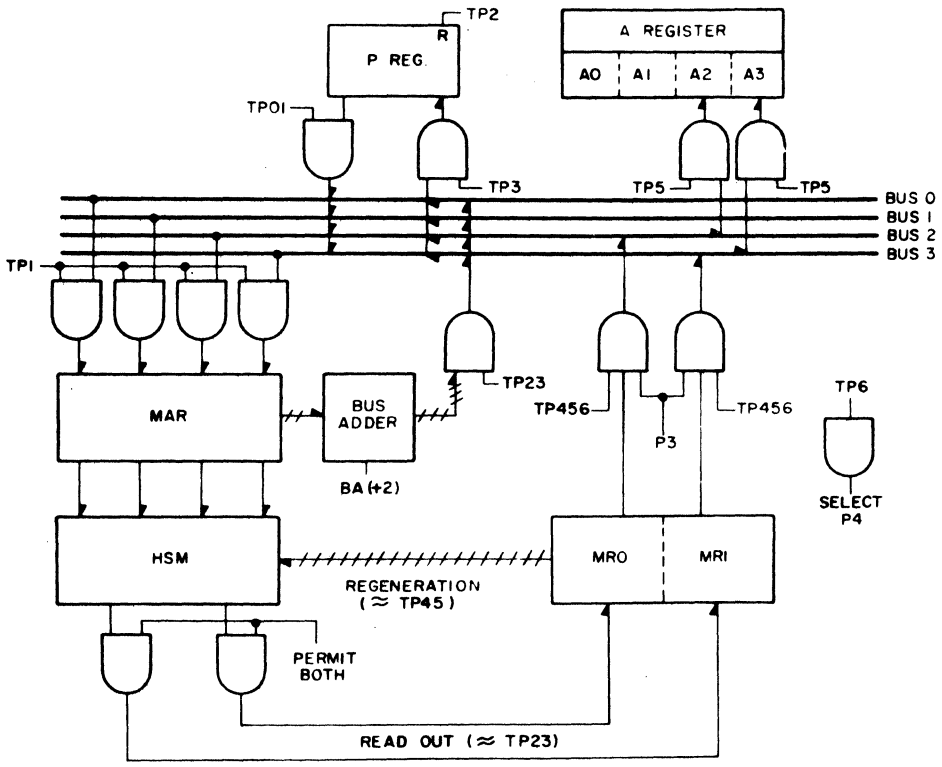


Figure 17 P3 Status Level

The P4 and P5 status levels bring out the B address and place it in the B register in the same general steps performed by P1, P2 and P3.

At the end of P5 the P register would hold an address of ten greater than the address it started with at the beginning of P1. This address would remain in the P register while the computer executed the current instruction just brought out. When the instruction was finished, the computer would automatically select a P1 status level and the process of staticizing would again take place bringing out the second instruction in the program.

It should be noted that after a P5 status level, the Computer would examine the operation code and select the first processing level. This is the first

status level involved with the actual processing of an instruction.

Upon the completion of the instruction, the P1 status level is automatically selected and staticizing of the next instruction begins. The P Register at this time contains the address of the first diad of this instruction.

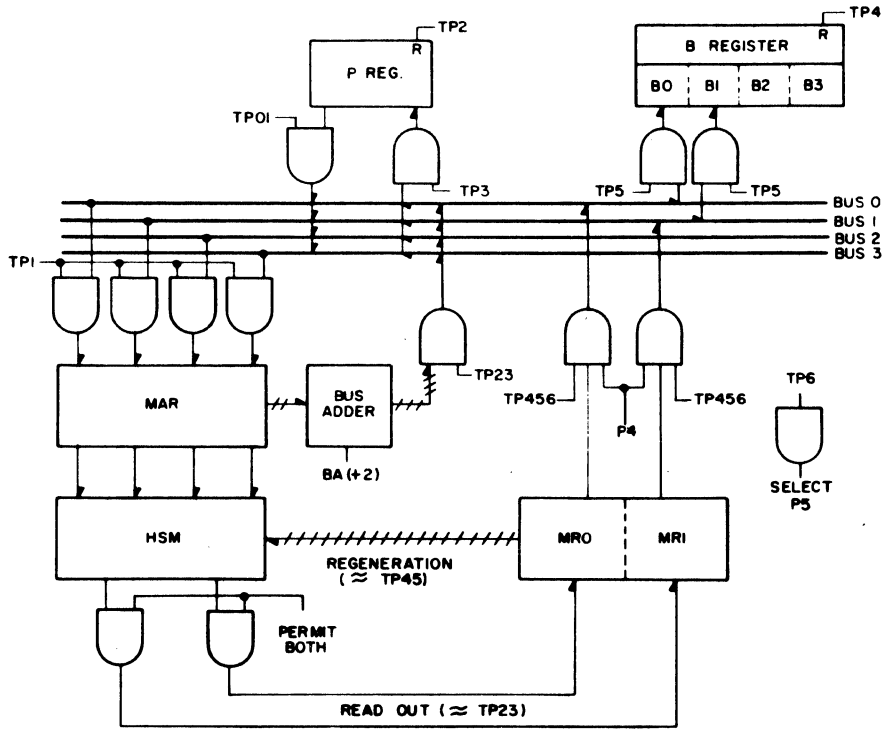


Figure 18 P4 Status Level

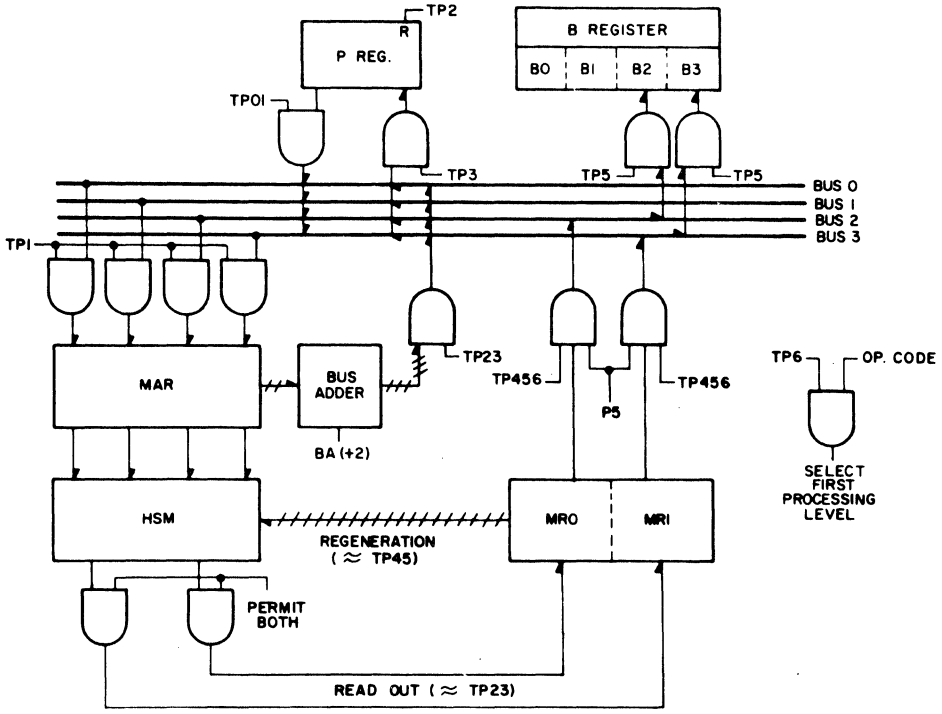


Figure 19 P5 Status Level

By means to be explained in the next section, it is possible for the Console Operator to manually insert the characters of an instruction into the NOR, N, A and B Registers. The net result of this resembles that of staticizing, but since it is accomplished by different means, it is not to be confused with staticizing.

H. 301 CONSOLE

The 301 Processor Console is located on the front of Rack 2 or the PCU Rack. It contains the pushbuttons and indicators necessary to operate the 301 processor. For the arrangement of the buttons, see the Programmers' Reference

Manual, Figure 6, on Page III-13. On Pages III-10 thru III-12, each push-button and its use are listed.

The console switches are two basic types: microswitches and strip switches. Some of the microswitches are momentary contact (activated only as long as the switch is held depressed); the others are alternate action switches, that is Press On, Press Off. Those in the latter category are:

1. OCSP	6. WTAB	11. BCT
2. FPLS	7. BAI	12. STLR
3. ICSP	8. INT	13. HSMI
4. RDM	9. ISIM	14. ALI
5. WRM	10. SMDI	

(See Programmers' Reference Manual for Switch Function.)

The upper center two rows of switches on the console are the register select switches. Each row of switches is called a "strip switch". The register select switches are series leaf-type contact switches that are mechanically interlocked, so that only one switch can be "set" at a time. The mechanism is such that depressing any given switch causes any other switch that is "set" in that row to be released. Viewing the console from the front, the rightmost buttons of the two upper center rows are reset buttons. They are associated with strip switches--depressing one of these reset switches releases any other switch that may be set in its row. Both strip switches must be reset before the Processor can be started.

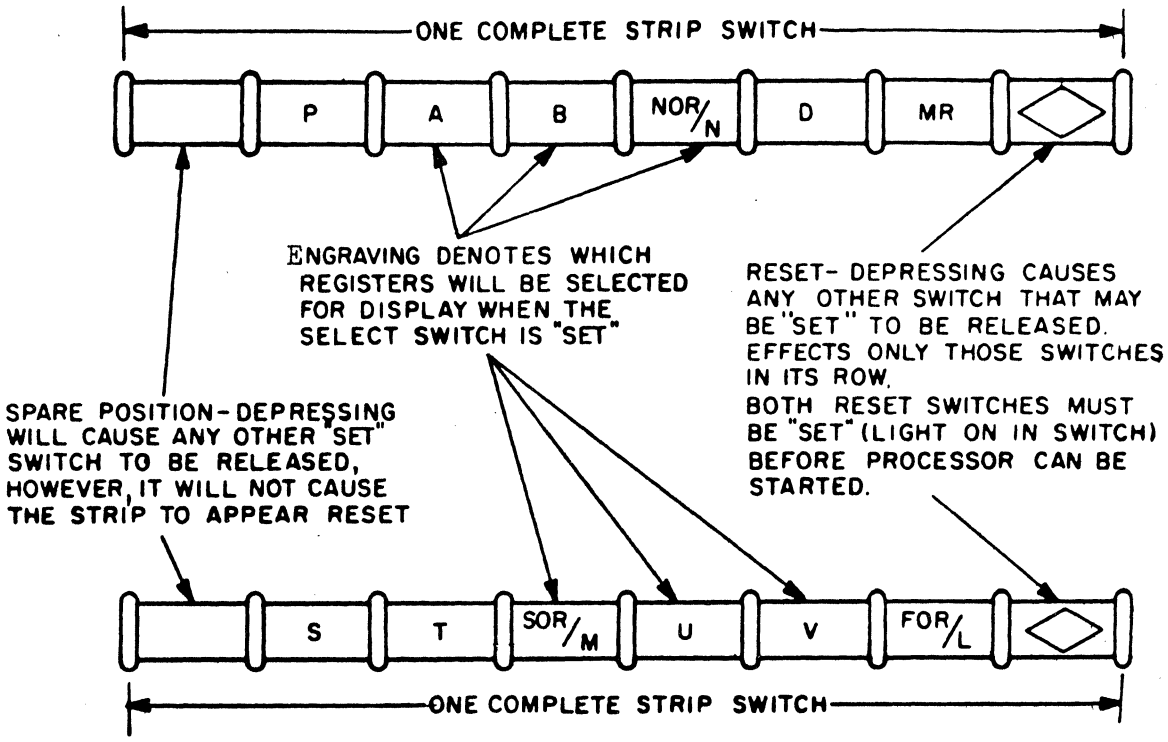


Figure 20 Strip Switch

On the right of the console (viewed from operator's side) are four rows of switches and lights. Each row within the block has eight positions. They are designated by the engraving on the rightmost button.

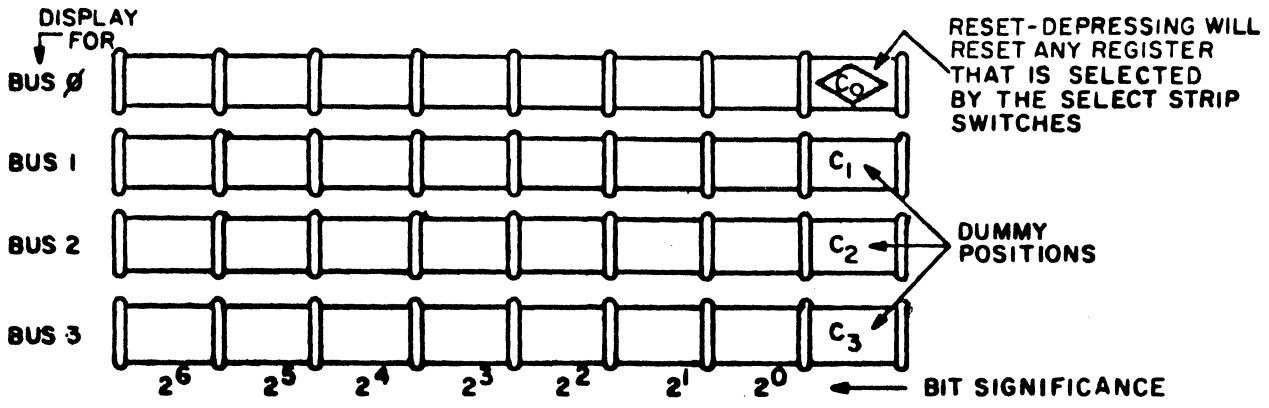


Figure 21 Bus Switches

The upper row of switches and indicator are connected to the Processor Bus Ø. The character as displayed on the console is called the CØ character. The second row from the top is connected to Bus 1 and called the C1 character, and so on through the C3 character. The console switches for the C3 character are connected to the Processor Bus 3 and through wiring unique to these switches to the N register.

DISPLAYED IN CONSOLE LOCATION	REGISTER SELECTED											
	P	A	B	NOR/N	D	MR	S	T	SOR/M	U	V	FOR/L
$C_Ø$	PØ	AØ	BØ	NONE	NONE	NONE	SØ	TØ	NONE	UØ	NONE	NONE
C_1	P1	A1	B1	NONE	NONE	NONE	S1	T1	NONE	U1	V1	NONE
C_2	P2	A2	B2	NOR	D2	MRØ	S2	T2	SOR	U2	V2	FOR
C_3	P3	A3	B3	N	D3	MR1	S3	T3	M	U3	V3	L

Figure 22 Bus Display Chart

When a register is selected by the register select switches, the contents will be displayed in the CØ through the C3 console indicators as shown in Figure 22.

The proper operating technique for the strip switches, is to reset them, each by its own reset button, between the selection of registers. For example, if the operator wished to place or view information in both the A and B registers, the selection sequence should be:

1 - Select A
2 - Reset strip switch
3 - Select B
4 - Reset strip switch

OR

1 - Select B
2 - Reset strip switch
3 - Select A
4 - Reset strip switch

This technique will preclude the possibility of inadvertently gating the contents of one register to another.

Many lights on the console are lit by lamp driver circuits and not directly by the switches. The bus display lights, for instance require lamp drivers. Switches are provided to place information on the buses, and lamp drivers formed to the buses connect back to the lights in the switches on the console.

In other cases, the console provides display (lights) only without switches. Examples of this are the error indicators to the left of the console. It is desirable to have some means of checking for open lamp filaments in circuits of this type with no switch to cause a lamp to light. This need is fulfilled by a lamp check circuit. The lamp check circuit checks only those lamps in the error indicators. The FAL (Record File mode alarm) indicator is above the lamp check switch.

1. Read From Memory Procedure

The following procedure will enable you to read successive memory locations out of memory at a diad rate.

Procedure:

1. Depress general reset.
2. Select the "A" register.
3. Set the required four character address, with correct parity, in CØ, C1, C2 and C3.
4. Reset the Strip Switch.
5. Set RDM switch.
6. Depress START.
7. Select MR to display the diad that was addressed in C2 and C3.
8. To read the next diad, repeat steps 4, 6 and 7.

The status-flow for Read From Memory is shown in Figure 23.

READ FROM MEMORY (RDM)

TP01	A → BUS
TP1	BUS → MAR BA(+2) Generate CL Permit Both
TP2	Reset A
TP23	BA → BUS
TP3	BUS → A
TP6	Inhibit Parity Checking on: NOR; N; STL

Figure 23 Read From Memory Status-Flow

2. Write to Memory Procedure

The following procedures will allow you to change information in memory or to insert new information in memory at a diad rate:

Procedure:

1. Depress general reset.
2. Select the "A" register.
3. Set the required four character address with correct parity in CØ, C1, C2, and C3.
4. Reset the strip switch.
5. Select the MR register.
6. Set the diad that is to be written in memory in C2 and C3 with correct parity.
7. Reset the strip switch.
8. Set WRM switch.
9. Depress START.
10. To write to the next successive diad locations repeat steps 5, 6, 7 and 9.

3. Write to Memory (WRM)

TP01	A → BUS
TP1	Inhibit Resetting MR BUS → MAR BA(+2) Generate CL Inhibit Both
TP2	Reset A
TP23	BA → BUS
TP3	BUS → A
TP6	Inhibit Parity Checking ON: NOR, N, STL

Figure 24 Write to Memory Status-Flow

The two characters inserted in the MR during the write memory cycle are now regenerated into the memory at the address specified by the "A" register. The status-flow for this operation is shown in Figure 24.

4. Manual Staticizing

To manually staticize an instruction, you must do what is normally done during the P1 thru P5 status levels.

Procedure:

1. Press general reset.
2. Select NOR/N (you will note that C3 is all lit. This is because the N register is down counter and must be "set" and then the unwanted bits "reset"out).
3. Enter the OP code in C2 and the N count in C3 by resetting unwanted bits. Be sure C2 and C3 contain good parity.
4. Reset the strip switch.
5. Select the "A" register.
6. Enter the required four digit address in CØ, C1, C2, and C3 with good parity.

7. Reset the strip switch.
8. Select the "B" register.
9. Enter the required four digit address in CØ, C1, C2, C3 with good parity.
10. Reset the strip switch.
11. Reset the status level register by pressing STL button. (Note General Reset put a P1 status level in the status level register.)
12. Enter the first processing level of the instruction to be done in the status level register with good parity. (This information can be found on your code card or your status flow manual.)
13. Press the ICSP button.
14. Press any other buttons necessary to do the instruction such as WTAB or ALI.
15. Press START.

If you make a mistake when setting information in a register, use the CØ Reset button to reset the selected register, only. If you press General Reset, all the registers but the selected register will be reset.

I. MEMORY DISPLAY PANEL (Figure 25)

The memory display panel (MDP) is located on the wiring side (side "A") of the Processor rack. The display panel consists of 42 lights for the purpose of displaying the condition of numerous flip-flops. The upper four rows display the information in the memory addressing registers. Starting at the top and working down they are CØ through C3 or the MSD through the LSD of an address. The bit positions are arranged with 2^0 on the right and 2^6 on the left. The lower two rows of lights show the conditions (set or reset) of certain logic flip-flops. In all cases, the light is lit when the flip-flop is "set".

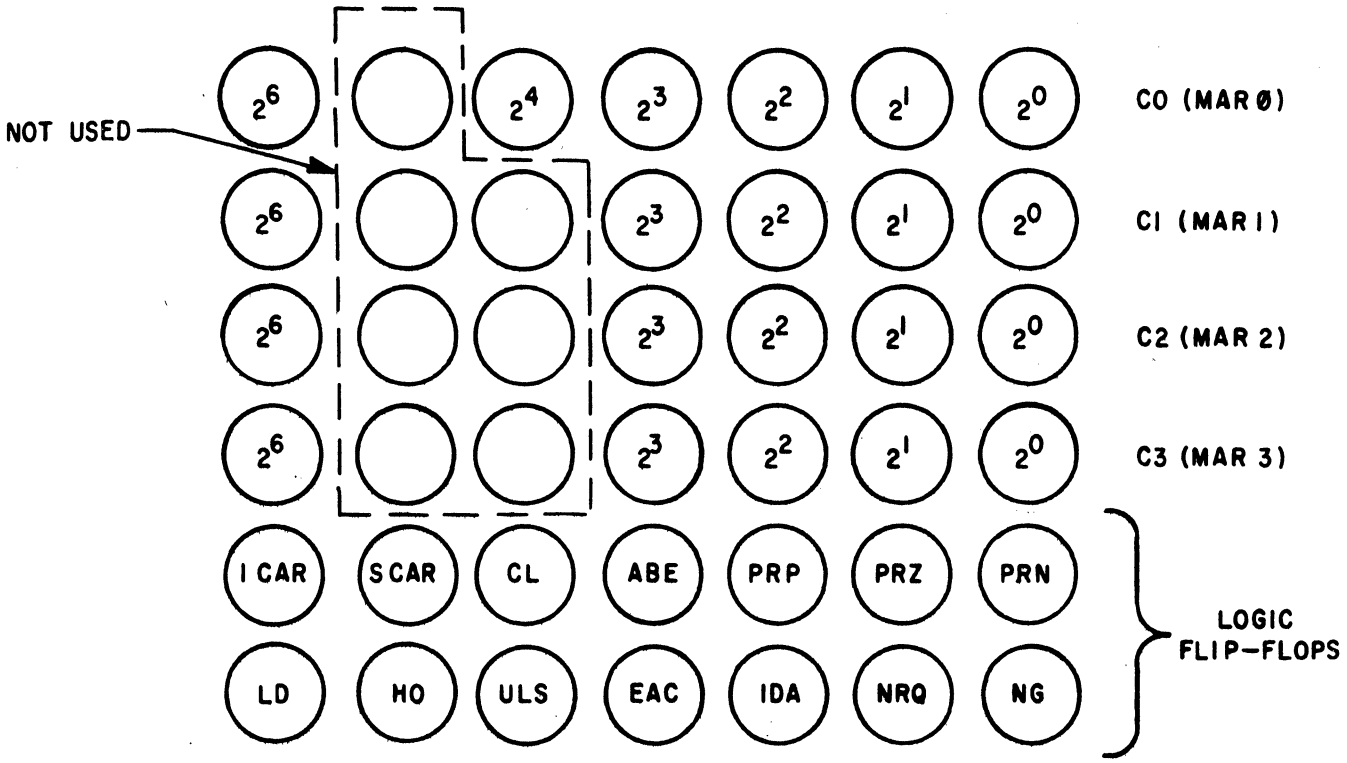


Figure 25 Memory Display Panel

SECTION II

DATA HANDLING INSTRUCTIONS

TO THE STUDENT

This lesson is designed to develop the student's understanding of the data handling instructions. Each instruction is first described in terms of its purpose and effect. Then the details of its execution are shown with the aid of a simplified block diagram of the computer and simplified status level timing charts. Finally, practice problems are provided to help the student gain skill in using the data handling instructions.

A. The DATA HANDLING INSTRUCTIONS, INTRODUCTION

The Data Handling instructions are eight in number and each performs the function that its name implies. These instructions will be involved with transferring data and locating specific characters in memory. The following is a list of the 301 instructions classified as Data Handling:

Op. Code	
J	- TRANSFER SYMBOL TO FILL
M	- TRANSFER DATA LEFT
N	- TRANSFER DATA RIGHT
#	- TRANSFER DATA BY SYMBOL LEFT
P	- TRANSFER DATA BY SYMBOL RIGHT
K	- LOCATE SYMBOL LEFT
L	- LOCATE SYMBOL RIGHT
A	- TRANSLATE BY TABLE

B. J - TRANSFER SYMBOL TO FILL (SF)

In order to make use of HSM for storage and as a work area, we must be able to insert into HSM the characters we wish to store and use. An instruction which may be used for inserting characters is the Transfer Symbol to Fill instruction. The Transfer Symbol to Fill instruction places a selected symbol in each memory location between and including two given addresses. During staticizing, the operation code (J) is placed in the NOR register. The selected symbol is placed in the N register. The two given addresses are placed in the A and B registers.

SF operates from left to right in HSM starting with the A address and terminating after the location addressed by B has been filled. The A address determines the location into which the N character is to be written during the current memory cycle. The A register is incremented by one during each memory cycle, which causes the instruction to operate from "left to right". When A-B equality is reached (i.e. when the A address = the B address) the instruction terminates. The SF instruction is useful for "filling" many consecutive locations in memory, or even all of memory, with a selected character. It may also be used to insert a selected character into one specific memory location.

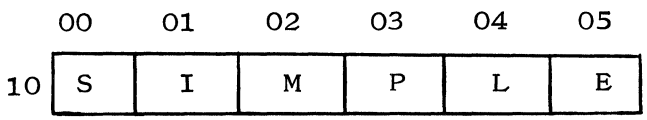
NOTE: For a brief summary of the operating characteristics, see page V-13 of the Programmer Reference Manual.

1. Transfer Symbol to Fill Instruction Format

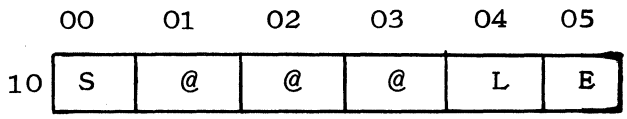
Op. Code	N	A	B
J	Selected Symbol (any 301 character)	Leftmost location to receive selected symbol	Rightmost location to receive selected symbol

2. Instruction Execution

Example: J @ 1001 1003



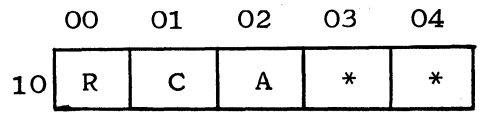
HSM before instruction execution



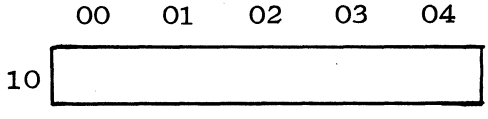
HSM after instruction execution

PRACTICE PROBLEMS

1. J * 1000 1002

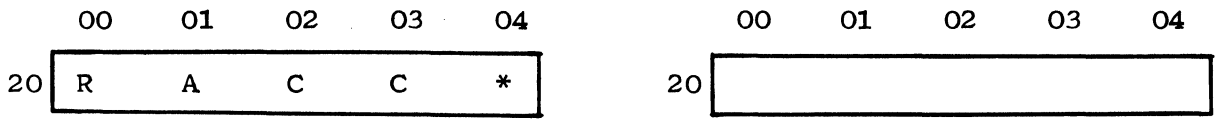


HSM Before



HSM After

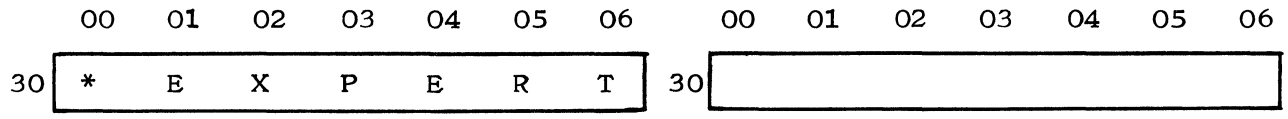
2. JE 2003 2003



HSM Before HSM After

3. Show how the following program will affect memory. (The numbers 1000, 1010, and 1020 are the addresses of the instructions in memory.)

1000	J5	3005	3006
1010	JD	3002	3002
1020	J*	3004	3006



HSM Before HSM Final

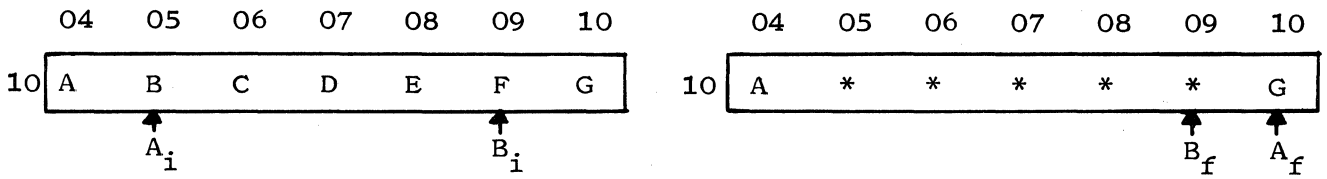
Because all 301 instructions are wired to operate in a certain manner, and because each instruction is executed while it is held in specific registers, there will exist an initial address and a final address for both the A and B Registers, at given times. Immediately after staticizing, the address in the A Register will be called A_i (A initial) and the address in the B Register will be called B_i (B initial). During the execution of the Symbol to Fill instruction, the B Address will remain the same while the A address is incremented by one each time, thereby designating where the selected symbol (N character) will be placed in memory. Once the A Address equals the B Address, a flip-flop called ABE will become set. However, the symbol must still be placed in the last location (B Address) and during the final memory cycle, the A Address is counted up once more. Therefore, A_f (A final) or the contents of the A Register, once the instruction has been completed, will be $B_i + 1$. This is one location to the right of the initial B Address. The control flip-flop ABE is the deciding factor in terminating the instruction.

Throughout this manual, reference will be made to A_i, A_f, B_i, and B_f because initial and final register contents play an important role in programming. It will become evident, as we discuss each instruction, that successive instructions in a program can be made dependent upon final register contents of preceding instructions.

Also, it should be noted that, where HSM or Register contents are displayed, the symbol Ø will be substituted for a zero, when it is considered that it might be mistaken for the letter "o".

Example:

J * 1005 1009

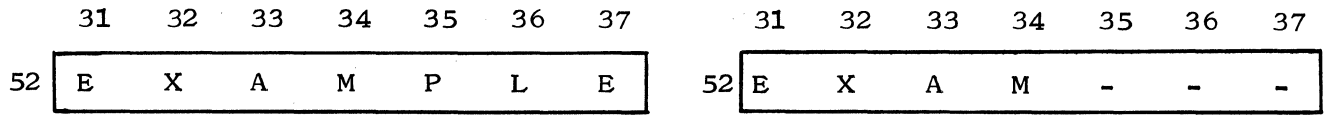


HSM Before HSM After

4. Write an instruction which would fill the last four locations of a 10K memory with zeros.

_____

5. Write an instruction to change memory as shown. Give A and B final



HSM Before HSM After

_____

A_f = _____ B_f = _____

3. Machine Operation

After staticizing, the Symbol to Fill instruction uses just one status level - an A2. The selected symbol is placed in memory at the A address, and the address is incremented by one. The process continues until the A address is equal to the B address at which time the instruction terminates. The sequence of events for an A2 is:

	A2
TP01	A address is gated onto Bus.
TP1	Contents of Bus gated into MAR and sent to Bus Adder to be modified by +1. Generate a command level. Decode MAR address and select diad. Set ABE flip-flop if A register contents equal B register contents.
TP23	Gate N character onto Bus 3. Permit one character opposite to that specified by MAR address, to be read out into MR.
TP3	Gate contents of Bus 3 into MR as specified by MAR address.
TP4	Reset A register.
TP45	Regenerate old and new character into selected diad. Gate modified Bus Adder address onto Bus.
TP5	Gate contents of Bus into A register.
TP6	If ABE flip-flop is set, select P1; if not set, select A2.

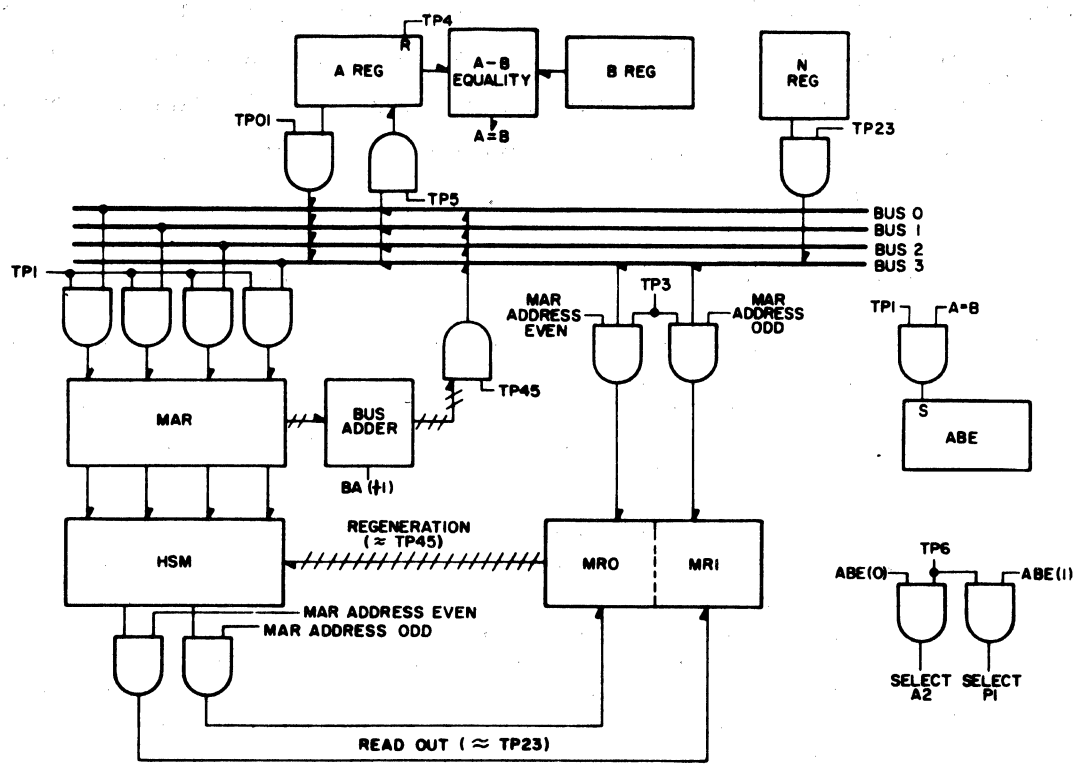


Figure 26 A2 of Symbol to Fill

The phrase "generates a command level" at TP1 means "initiate a memory cycle." If no command level is generated, the read-out-regenerate cycle involving HSM will not be performed.

Note that the equality of addresses is checked at the beginning of A2, and not after modification. Therefore, when A-B equality is reached and the ABE flip-flop becomes set, the A address will be incremented by one once more. Hence, the final contents of the A register is one location greater than that in the B register.

It should be pointed out that the ABE flip-flop is reset during P5 of a Symbol to Fill instruction.

Notice that if the SF instruction is to terminate (i.e., AB equality has been reached), P1 is selected at TP6 of the last A2 status level. In fact, when any instruction terminates, P1 will be selected at TP6 of its last processing status level. This selection of P1 is the factor which provides for sequential execution of the instructions in the order they are stored in memory.

4. Programming Errors

One programming mistake that can be made with the SF instruction is the reversing of the A and B addresses. The A address must identify the leftmost location to be filled and the B address the rightmost. If A is greater than B, A will count up for each location filled and move farther away from A-B equality. As soon as the upper limit of memory is reached, adding one more will revert the address back to all zeros and a WTT alarm will occur, provided WTAB is not set. If WTAB is set, the entire memory may be filled and the instruction will terminate on A-B equality.

Example:

```
J  0  1091  1090
```

Locations 1091 through 9999 (10K Memory) will be filled with zeros before a WTT alarm would occur (Assuming WTAB is not set).

Since SF terminates on A-B equality, if for some reason the A-B equality circuit fails to set the ABE flip-flop, the instruction will continue to fill HSM with the N character until a WTT error occurs. If WTAB is depressed, not setting ABE will cause the instruction to fill all of HSM repeatedly -- the computer will be "cycling" in a SF instruction.

A common way for this error to occur is to staticize a non-numeric address such as ;000 in the B register. Since the A register will be counting numerically during execution of the SF instruction, A-B equality will never be reached. However, notice that a simple parity error in the B register is not enough to cause this failure since the AB equality circuitry compares only the data bits, it does not examine the parity bits. Also notice that

not any non-numeric address will work. For example, a "K" in B2 would appear as a "2" since B2 has no 2^5 flip-flop. A $, however, would appear as a non-numeric character in any position of the B register since the four low-order bits are "1011" which is greater than "1001" or nine, the largest numeric.

PRACTICE PROBLEMS

6. The WTAB button is depressed. The following instruction is inserted manually into a 303A processor (10K).

   J 0 00001 0000

   What are: A_i=_____ B_i=_____ A_f=_____ B_f=_____

7. How many memory locations will be filled with 0's in Problem 6?

   a. 10,000
   b. 1
   c. 10,001
   d. none

8. How many A2 status levels will be performed in problem 6?

   a. 10,001
   b. 10,000
   c. 0
   d. 1

9. Write an instruction to fill every location in a 20K memory with *'s. (Assume WTAB is depressed.)

---

10. An operator attempts to manually insert 0's in the first 300 HSM locations with a "symbol fill" instruction (assume WTAB and ICSP is depressed). He enters J → NOR register, 0 → N register, 0000 → A register, but he forgets to enter a B address, so the B register remains completely blank. What happens?

a. total number of locations filled --

b. does computer stop --

If so A_f=_____ B_f=_____

c. Any errors --

If so which?

11. At TP4 time during the execution of the following instruction what will be the content of the MR? Complete "HSM After."

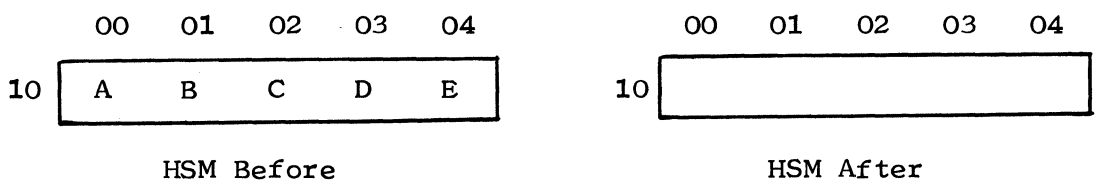
```
J    *    1002    1002
```

a. MR at TP4: MRO_____, MR1_____

b. At which TP time will the character "C" appear in the MR?

12. At what TP time of which A2 status level during the execution of the following instruction will the "D" appear in the MR? Will it appear in MRO or MR1 or neither?

```
J4    1001    1003
```



C. M - TRANSFER DATA LEFT (DL) REPEATABLE

N - TRANSFER DATA RIGHT (DR) REPEATABLE

The Transfer Data Left and Right instructions are used to transfer a number of characters from one group of consecutive locations in memory to another group of consecutive locations. The N character gives the number of characters to be transferred. The A address gives the starting location from which characters are to be taken. The B address gives the starting location into which the characters being moved are written. (Because of the regeneration portion of the memory cycle, the character "transferred" will also be rewritten into the location from which it was "taken.")

The DL instruction works from left to right (incrementing both the A and B register by 1, each memory cycle). The DR instruction works from right to left (decrementing both the A and B registers by 1, each memory cycle).

Each time a character is transferred the N count is reduced by one.

Both instructions terminate upon reducing the N count to zero. If N is zero when the instruction is staticized, no characters are transferred; the next instruction will be staticized and executed (i.e., P5 of DL or DR selects P1 to start staticizing the next instruction). The final addresses are:

DL: A_f= Address location one to the right of the last character transferred.

B_f= Address location one to the right of the last destination address.

DR: A_f= Address location one to the left of the last character transferred.

B_f= Address location one to the left of the last destination address.

These instructions are labeled "REPEATABLE" which means they may be used in conjunction with the repeat instruction which will be covered in the next lesson.

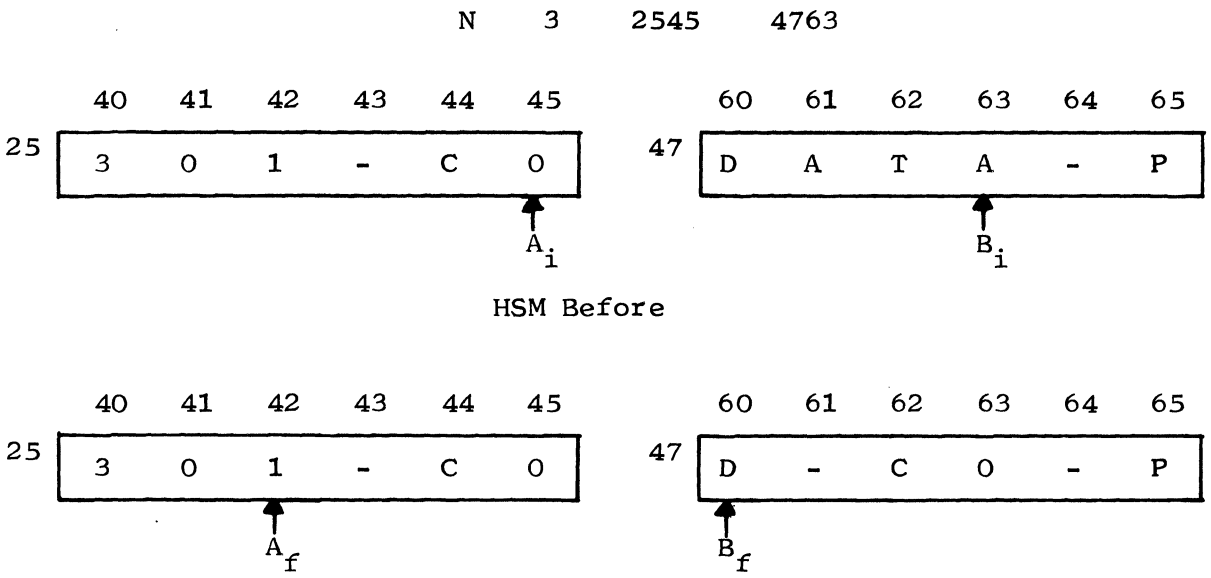
NOTE: The transfer data instructions are covered on pages V-9 and V-10 of the Programmer's Reference Manual.

1. Instruction Format

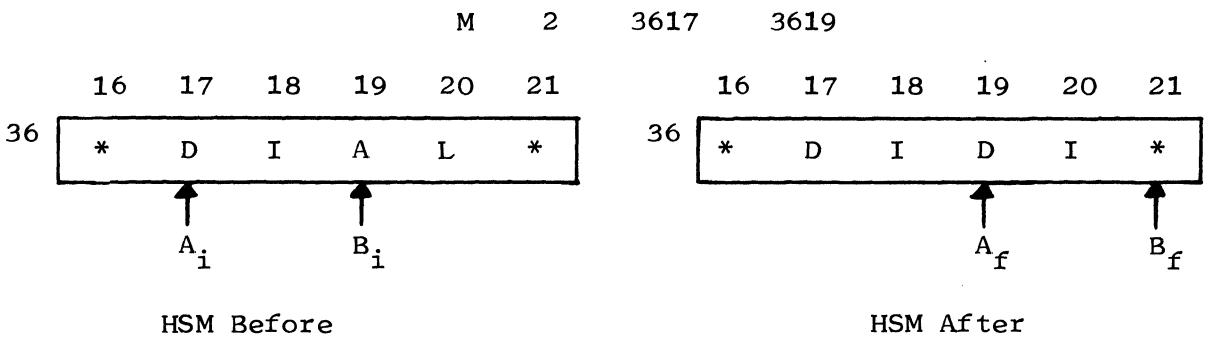
Op. Code	N	A	B
M or N	0-44 Characters Transferred	HSM Location of First Character to be transferred	Destination Location of First Character

2. Instruction Execution

Example 1:



Example 2:



The question might arise as to the need for transfer instructions working in either direction (in the general transfer of characters, there is really

not much need). However, if it is desired to shift a certain number of characters in memory for positioning purposes, in one direction or the other, the two instructions, M and N, are convenient.

For example, assume that ten characters in memory between 4000 and 4009 are to be shifted two positions to the right. The instruction would be:

N & 4009 4011

Note that an M instruction could not be used because of the "overlapping" of characters.

The instruction, M & 4000 4002, would shift ten characters to the right but would wipe out all characters from 4002 to 4009 with the two characters in 4000 and 4001.

Similarly, if it was desired to shift the original ten characters two positions to the left, an M instruction must be used

M & 4000 3998

The "overlapping" problem again inhibits using an N instruction in this case.

PRACTICE PROBLEMS

1. Execute the following instruction and show final HSM contents.

M 4 0212 0216

	10	11	12	13	14	15	16	17	18	19	20	21	22	23
02	A	5	6	3	2	1	7	8	4	9	3	2	0	6

HSM Before

	10	11	12	13	14	15	16	17	18	19	20	21	22	23
02														

HSM After

2. Execute the following instruction and show final HSM contents.

M 2 1562 1561

	58	59	60	61	62	63
15	A	B	C	D	E	F

HSM Before

	58	59	60	61	62	63
15						

HSM After

3. Write an instruction which will transfer 31 characters from address 5038 to address 7196 (working from right to left).

____

4. Execute the following two instructions and show final HSM contents.

N 4 3104 3105
J * 3101 3101

	00	01	02	03	04	05	06
31	*	C	A	S	E	*	*

HSM Before

	00	01	02	03	04	05	06
31							

HSM After

3. Machine Operation

The Transfer Data Left instruction has two different status levels, an A1 and a B. The A1 status level brings out the character found at the A address and stores it in the D2 register. In addition, the A address is incremented by one and sent back to the A register. The N register contents are decreased by one. The B status level transfers the character from D2 to memory as addressed by B, then increases the B address by one and sends it back to the B register. If N is equal to zero, the instruction terminates by selecting a P1. If N is not equal to zero, another A1 status level is selected. The process is repeated until N equals zero.

Note that A final will always be one location to the right of the last character transferred and B final one to the right of the last destination location. The DL instruction works from left to right with both addresses incrementing during execution. However, the DL instructions can be used to transfer characters from any HSM location to almost any other location if one bears the "overlap" problem in mind. For example: M2 1009 1006 or M2 1006 1009 are both legal operations, but with different end results.

	A1
TP01	A address is gated onto Bus.
TP1	Contents of Bus are sent to MAR and Bus Adder and one is added to the address. Command Level is generated. MAR address is decoded and diad is selected.
TP2	A register is reset.
TP3	N-1 Bus → A
TP23	Both characters are read out into MR. Contents of Bus Adder are gated onto Bus.
TP4	D register is reset.
TP45	Regeneration occurs and contents of MRO or MR1 as determined by address in MAR are sent to Bus 2.
TP5	Contents of Bus 2 are gated into D2 register.
TP6	B status level is automatically selected.

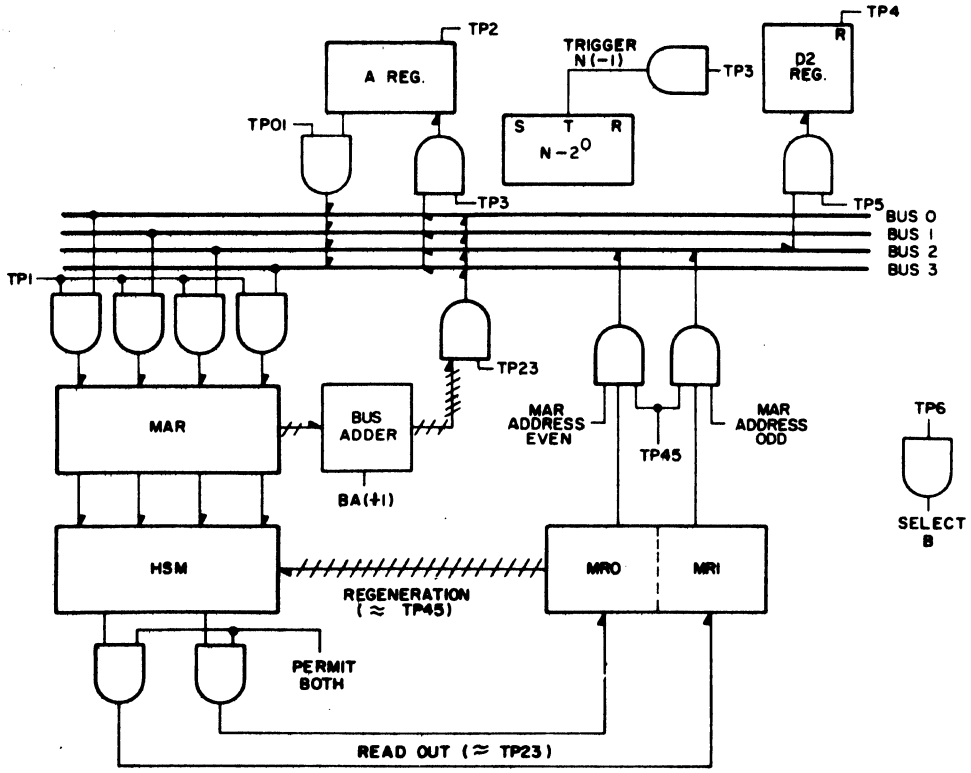


Figure 27 A1 of Transfer Data Left

	B
TP01	B address gated onto Bus.
TP1	Contents of Bus to MAR and Bus Adder where one is added to address. Generate Command Level. Decode address and select diad.
TP2	Contents of D2 are gated onto Bus 2. Permit one character from opposite location, as addressed by MAR to reach MR during read out.
TP3	Gate character from Bus 2 into MR as specified by address in MAR.
TP4	Reset B register.
TP45	Regenerate old and new character. Gate modified address from Bus Adder to Bus.
TP5	Contents of Bus are gated into B register.
TP6	If contents of N are equal to zero select P1, if not equal to zero select A1.

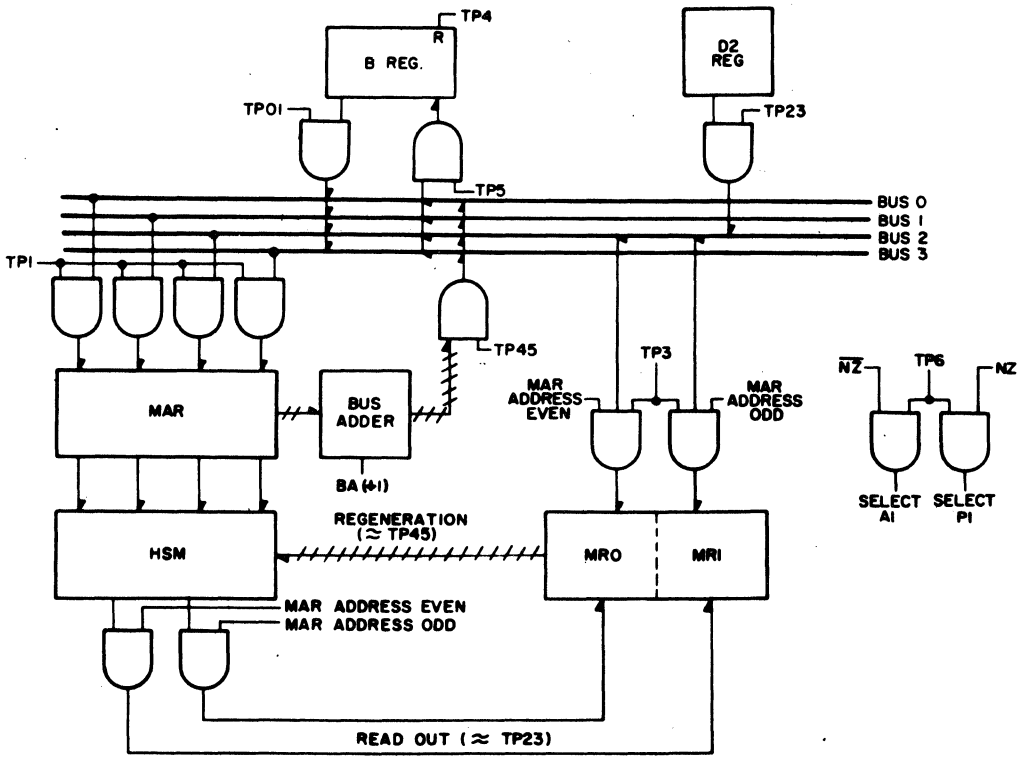


Figure 28 B of Transfer Data Left

The Transfer Data Right instruction has two different status levels, an A1 and a B. The A1 status level brings out the character found at the A address and stores it in the D2 register. The A address is decremented by one and sent back to the A register. The N register contents are also decreased by one. The B status level transfers the character from D2 to memory as addressed by B, then decreases the B address by one and sends it back to the B register. If N is equal to zero the instruction terminates by selecting a P1 and if N is not zero another A1 status level is selected. The process is repeated until N equals zero.

Note that A final will always be one location to the left of the last character transferred and B final one to the left of the last destination location.

The DR instruction works from right to left. That is, both addresses count down during execution. However, the Transfer Data Right can transfer characters from any HSM location to almost any other location without regard to relative position of the addresses to one another. For example: N 2 1006 1009 or N 2 1009 1006 are both entirely legal operations, but with different end results of course.

	A1
TP01	A address is gated onto Bus.
TP1	Contents of Bus are sent to MAR and Bus Adder and one is subtracted from the address. Command Level is generated. MAR address is decoded and diad is selected.
TP2	A register is reset.
TP23	Both characters are read out into MR. Contents of Bus Adder are gated onto Bus.
TP3	Contents of Bus are gated into A register. N is triggered down by one.
TP4	D register is reset.
TP45	Regeneration occurs and contents of MRO or MR1 as determined by address in MAR are sent to Bus 2.
TP5	Contents of Bus 2 are gated into D2 register.
TP6	B status level is automatically selected.

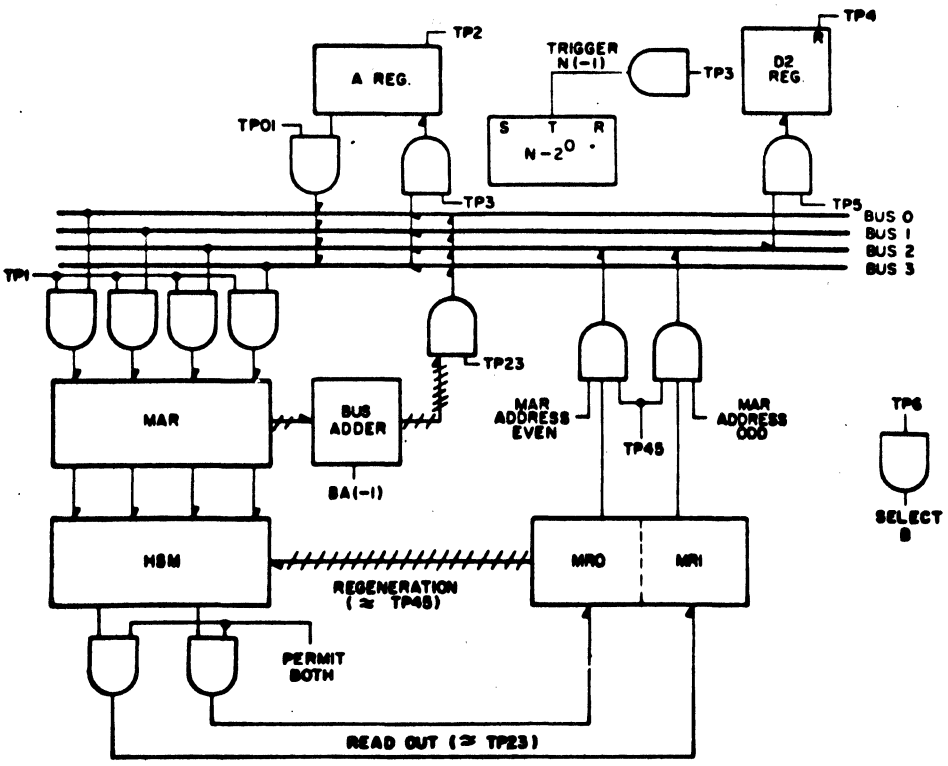


Figure 29 A1 of Transfer Data Right

	B
TP01	B Address gated onto Bus.
TP1	Contents of Bus to MAR and Bus Adder where one is subtracted from address. Generate Command Level. Decode address and select diad.
TP23	Contents of D2 are gated onto Bus 2. Permit one character from opposite location, as addressed by MAR to reach MR during read out.
TP3	Gate character from Bus 2 into MR as specified by address in MAR.
TP4	Reset B Register
TP45	Regenerate old and new character. Gate modified address from Bus Adder to Bus.
TP5	Contents of Bus are gated into B register.
TP6	If contents of N are equal to zero select P1, if not equal to zero select A1.

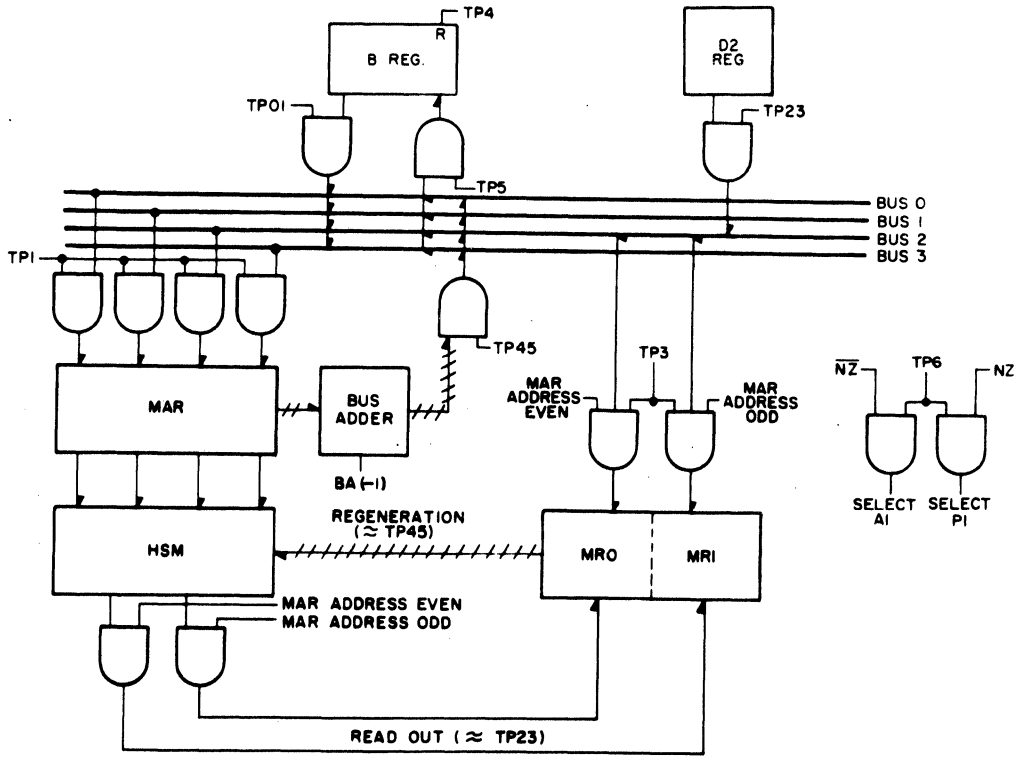
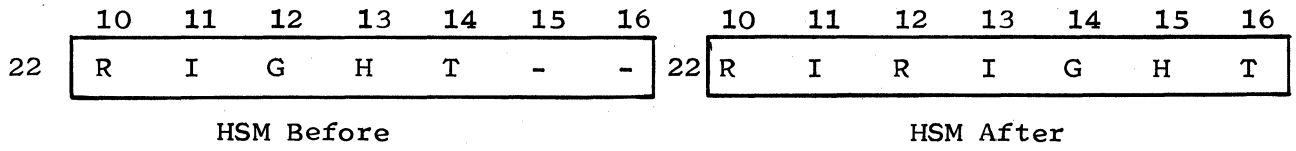


Figure 30 B of Transfer Data Right

4. Programming Errors

There are basically two ways that the M or N instructions may cause unsuspected trouble. One way is in transferring an overlapping field in one direction or the other. For example, assume one wishes to transfer five characters two positions to the right.



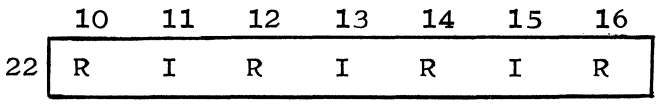
HSM Before HSM After

The only single instruction which will accomplish this is a Transfer Data Right:

```
N   5   2214   2216
```

If a Transfer Data Left is used, the following would occur.

```
M   5   2210   2212
```



HSM After

A similar problem occurs in shifting to the left. Therefore, the B address of an M or N instruction should not fall between the initial A address and A ± N(+ for Data Left, - for Data Right) where N is the N character.

The second method whereby the M or N instructions might not function properly is where an N character is used which does not exist in the N count. In this instance there is no alarm, but the number of characters that will be transferred will be a number which is an extension of the nearest lower N count. For example in the following instruction, the N character is an asterisk(*).

```
M   *   1000   3000
```

The nearest lower N count character is R which represents 29. An * would transfer 32 characters since it is 3 characters away from R in the code. An S (legal N count for 32) would transfer the same number of characters.

It should be noted that if one forgets to use an N count character and instead writes a two digit number for the N character, every character of the instruction as well as those in succeeding instructions will be shifted over one location to the right since a maximum of ten characters is permitted per instruction.

For example: N 12 3046 5028

would be executed as: N 1 2304 6502

The operation code of the next instruction would be an 8.

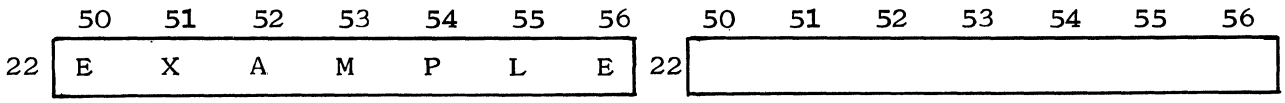
The correct instruction is: N B 3046 5028

PRACTICE PROBLEMS

5. Refer to the simplified status flow just presented and describe the differences between the Transfer Data Right and Transfer Data Left instructions.

6. Perform the following instruction and give the required information.

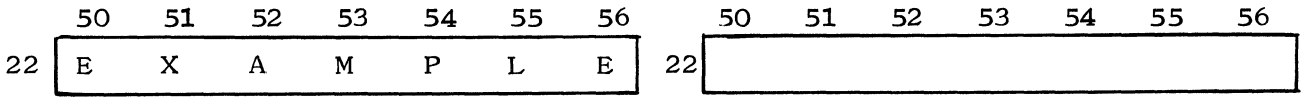
M 5 2250 2252



HSM Before HSM After

A_f= _____ B_f= _____

7. Execute: N 5 2254 2256



A_f= _____ B_f= _____

8. Which instruction should you use to successfully move a block of six characters two positions to the right.

9. Execute the following program and indicate the final HSM contents.

```
1000  J1  2004  2004
1010  M2  2000  2005
1020  N1  2009  1031
1030  N4  2008  2003
```

	00	01	02	03	04	05	06	07	08	09
20	A	N	T	I	*	*	O	D	A	2

HSM Before

	00	01	02	03	04	05	06	07	08	09
20										

HSM After

10. Use either a DL or DR instruction to duplicate the results obtained by "J * 1001 1005."

Instruction: _____

	00	01	02	03	04	05
10	*	A	B	C	D	E

HSM Before

	00	01	02	03	04	05
10						

HSM After

D. # - TRANSFER DATA BY SYMBOL LEFT (DSL) REPEATABLE

P - TRANSFER DATA BY SYMBOL RIGHT (DSR) REPEATABLE

The DSL and DSR instructions are, like the DL and DR instructions, used to transfer a number of characters from one group of consecutive locations in memory to another group of consecutive locations. Also, like the DL and DR instructions, DSL and DSR use the A register to address the transfer location and the B register to address the destination location. However, unlike the DL and DR instructions, DSL and DSR continue to transfer characters until a selected character has been transferred. The selected character is the N character. DSL works from left to right, and DSR works from right to left.

DSL and DSR store A_f in a standard location called STA (Store A). STA consists of memory locations 0212-0215 inclusive. Because we know A_i (A initial) which is just the contents of the A register after staticizing, by consulting STA after execution, the number of characters transferred during execution can be determined.

Again the word "repeatable" after DSL and DSR means that the instructions can be repeated using the Repeat instruction.

NOTE: Pages V-11 and V-12 of the Programmer's Reference Manual give operations summaries.

The final addresses are:

DSL: A_f = One location to right of selected symbol in original area.

B_f = One location to right of selected symbol in destination area.

DSR: A_f = One location to left of selected symbol in original area.

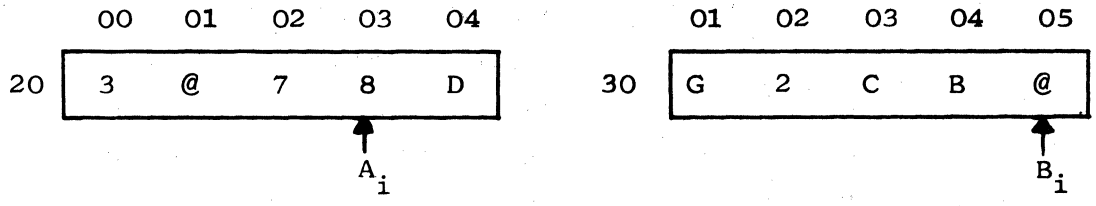
B_f = One location to left of selected symbol in destination area.

1. Instruction Format

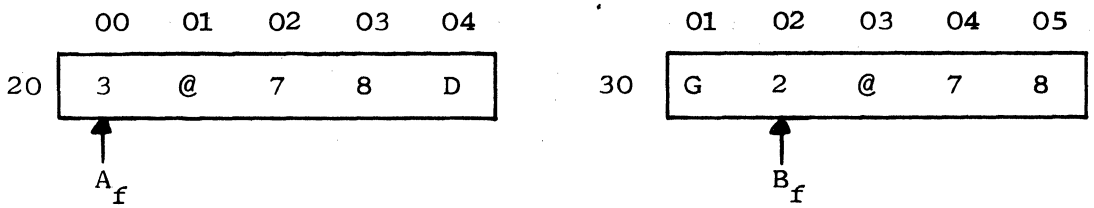
Op. Code	N	A	B
# or P	Selected Symbol on Which to Stop Transferring	HSM Location of First Character to be Transferred	Destination Location First Character

2. Instruction Execution

Example #1: P @ 2003 3005

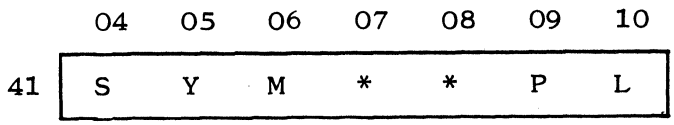


HSM Before

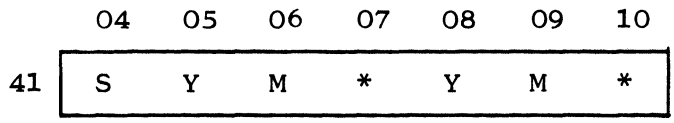


HSM After

Example #2: # * 4105 4108

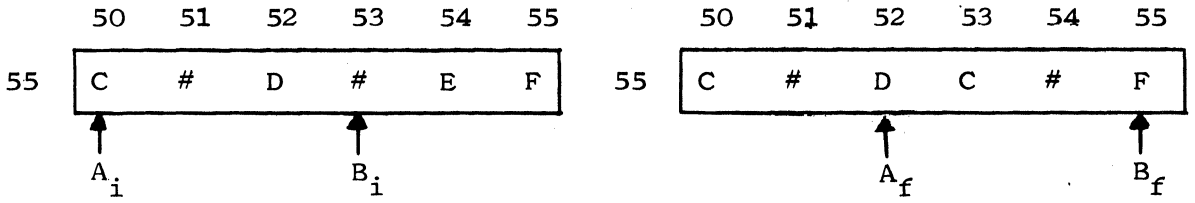


HSM Before



HSM After

Example #3: # # 5550 5553



HSM Before HSM After

Note that the selected symbol is itself the last character to be transferred.

PRACTICE PROBLEMS

1. Using a DSR instruction and the following HSM contents, transfer the characters, * P R O, to locations 4818, 4819, 4820, and 4821, respectively.

	10	11	12	13	14	15	16	17	18	19	20	21
48	*	P	R	O	G	R	A	M	*	1	2	3

---

2. Execute the following instruction and show final HSM contents.

E 8826 8825

	25	26	27	28	29	30	31
88	R	E	S	U	L	T	S

HSM Before

	25	26	27	28	29	30	31
88							

HSM After

	12	13	14	15
02				

HSM After

3. If the following instruction were attempted what would occur?

P / 3564 3563

	60	61	62	63	64	65	66
35	1	2	3	/	4	5	6

HSM Before

a. Use of DSL/DSR to Locate a Symbol in HSM

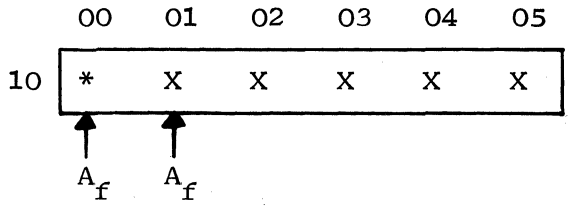
The DSL and DSR instructions can be used to locate a specified character and

not change the contents of memory in the process.

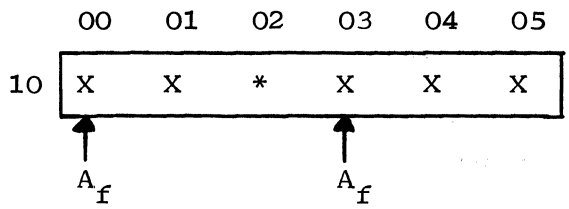
NOTE: These instructions are the true locate symbol instructions.

Assume a quantity of unknown length exists in memory between 1000 and 1005, and that the quantity will never exceed five characters. The quantity will always, however, be preceded by an asterisk (*). If it is desired to locate the MSD of the quantity and not disrupt memory, the following instruction will suffice.

Example #4: # * 1000 1000



Example #5: * 1000 1000



NOTE: In both of the above examples, $A_i = B_i$ and $A_f = B_f$.

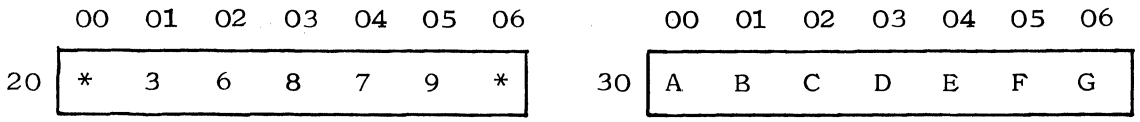
The only assumption needed is that the characters preceding the asterisk are not asterisks.

The LSD of a quantity can similarly be located by using a P (DSR) instruction.

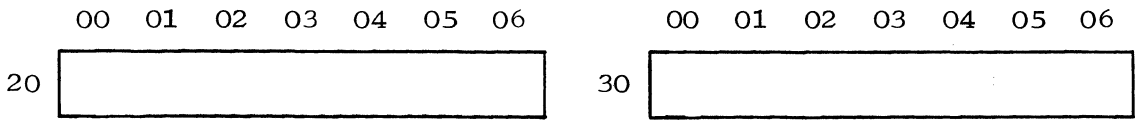
PRACTICE PROBLEMS

4. Execute the following program and show final HSM contents.

```
1000  J  Ø  3000  3006
1010  P  *  2005  3006
1020  #  *  3000  3000
```

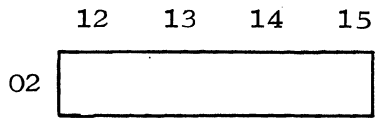


HSM Before



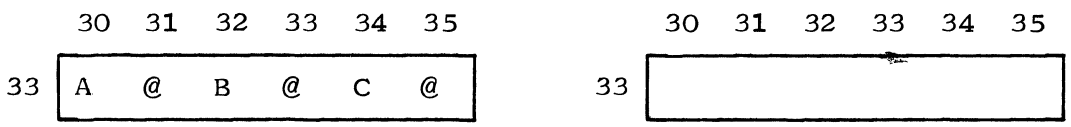
HSM After

What address would be in 0212 to 0215?



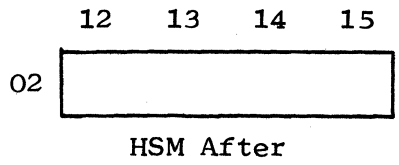
5. Execute the following program and show final HSM contents. [1]

```
1000  P  @  3334  3335
1010  N  4  0215  1025
1020  #  @  3330  3330
```



HSM Before

HSM After



[1] The Transfer Data Right instruction transfers the contents of STA to the locations which make up the A Address of the Transfer Data by Symbol Left instruction.

3. Machine Operation

Again only two basic status levels are used, an A1 and aB. The function of the A1 is to bring out the character as addressed by A and place it in the D2 register. The N character is gated into D3 and a comparison takes place between D2 and D3. Meanwhile the A address is incremented or decremented by one and sent back to the A register. The B status level transfers the character from D2 to memory at the B address and increments or decrements that address by one. The result is sent back to the B register. The comparator output is then examined. If the character transferred is equal to the N character (D2 = D3) the instruction terminates; if not the instruction continues by selecting another A1 status level.

	A1
TP01	Gate A address onto Bus.
TP1	Contents of Bus sent to MAR and Bus Adder where one is added or subtracted from address. Command Level is generated and MAR address is decoded.
TP2	A register is reset.
TP23	Both characters are read out into the MR. The contents of Bus Adder are gated onto the Bus.
TP3	The address on the Bus is sent back to the A register.
TP4	The D register is reset.
TP45	Regeneration occurs. Character as specified by address in MAR is gated onto Bus 2. Contents of N register gated onto Bus 3.
TP5	Contents of Bus 2 gated into D2 and contents of Bus 3 gated into D3.
TP6	Select B status level.

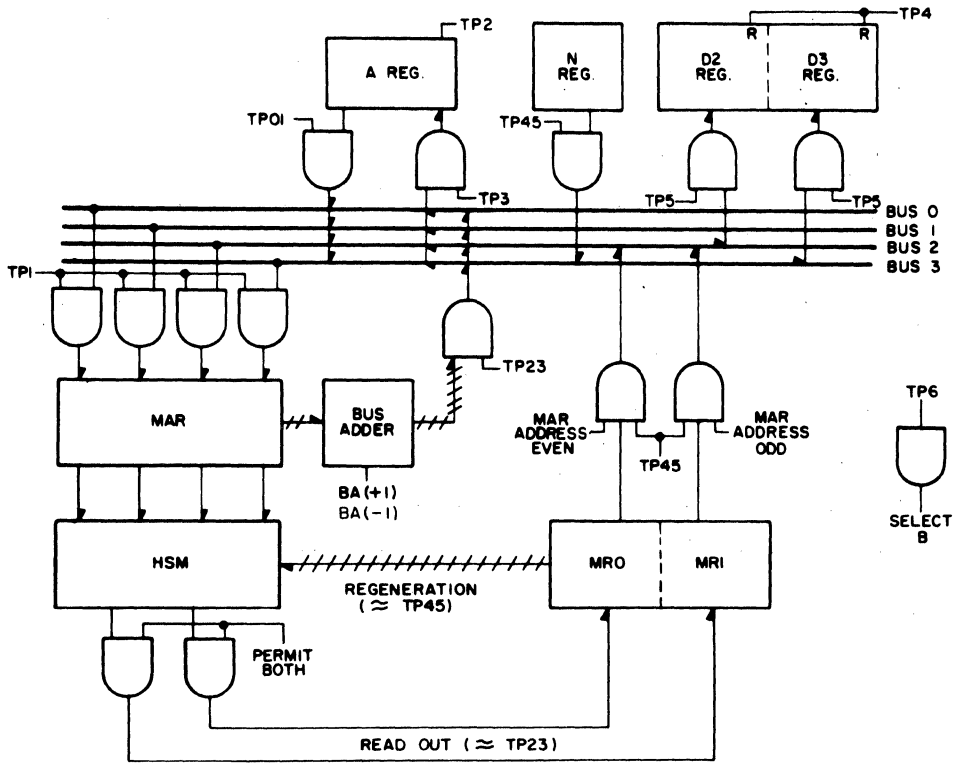


Figure 31 A1 of Transfer Data by Symbol Left or Right

	B
TP01	B address onto Bus.
TP1	Contents of Bus to MAR and Bus Adder where on is added to or subtracted from the address. Command Level is generated and MAR address is decoded.
TP23	The contents of D2 are gated onto Bus 2. One character is read out from the opposite location of the address in the MAR into the MR.
TP3	Depending upon the MAR address, the character on Bus is gated into either MRO or MR1.
TP4	B register is reset.
TP45	Regeneration occurs. Contents of Bus Adder gated onto Bus.
TP5	Address on Bus gated into B register.
TP6	If contents of D2 equal contents of D3 select STA 1, if not select A1.

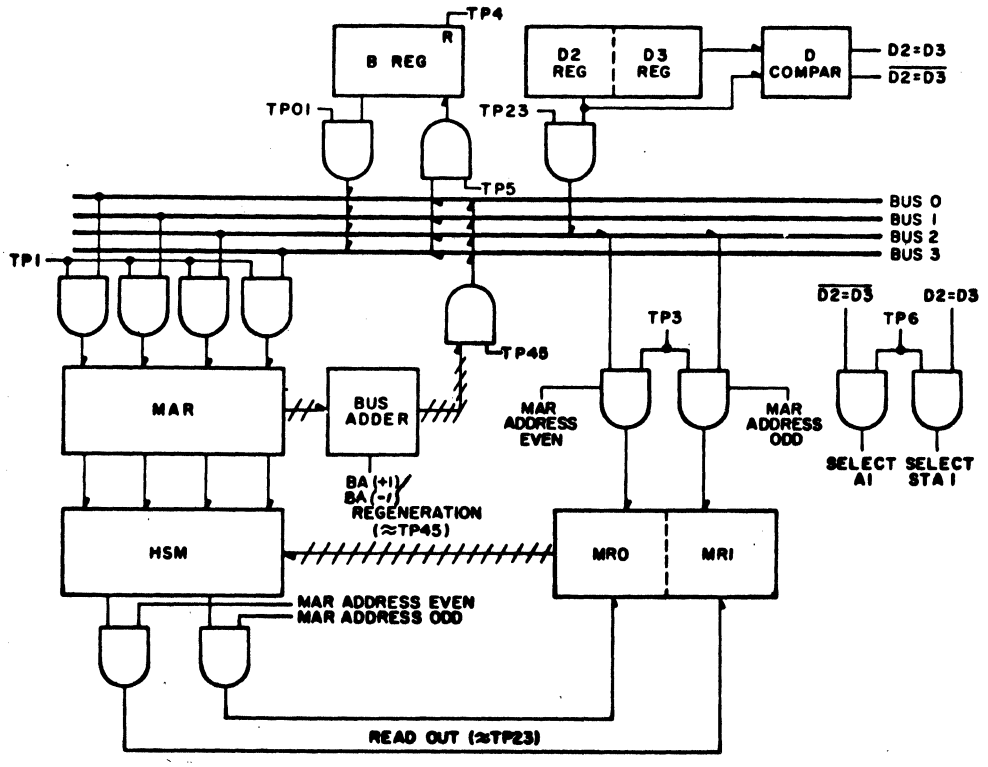


Figure 32 B of Transfer Data by Symbol Left or Right

a.) STA 1 and STA 2

Termination for the DSL and DSR is slightly different than the previous instructions discussed. Two additional status levels called STA 1 and STA 2 are executed when transferring is complete. These two status levels comprise the process known as Store A - storing the final A address in standard locations 0212 through 0215. The time pulse breakdown for the A1 of a DSL is as follows.

STA 1 and STA 2 were not used with DL and DR because with these instructions the N count is used to determine the number of characters moved, so given the initial A address and the N count, the programmer can easily determine A final. However, when using DSL or DSR, transfer continues until a selected symbol is encountered in the "A-field." The programmer may not know in advance where in HSM the termination symbol will be found, so the STA 1 and STA 2 are used to insert this information into a standard location (0212-0215) so that the programmer will know how many characters were transferred, etc.

	STA 1
TP01	Generate address 0212 from address generator onto Bus.
TP1	Contents of Bus to MAR. Generate command level and decode MAR address.
TP23	Gate A address onto Bus. Inhibit both characters being read out from reaching MR.
TP3	Gate contents of Bus 0 into MRO and Bus 1 into MR1.
TP45	Regenerate.
TP6	Select STA 2.

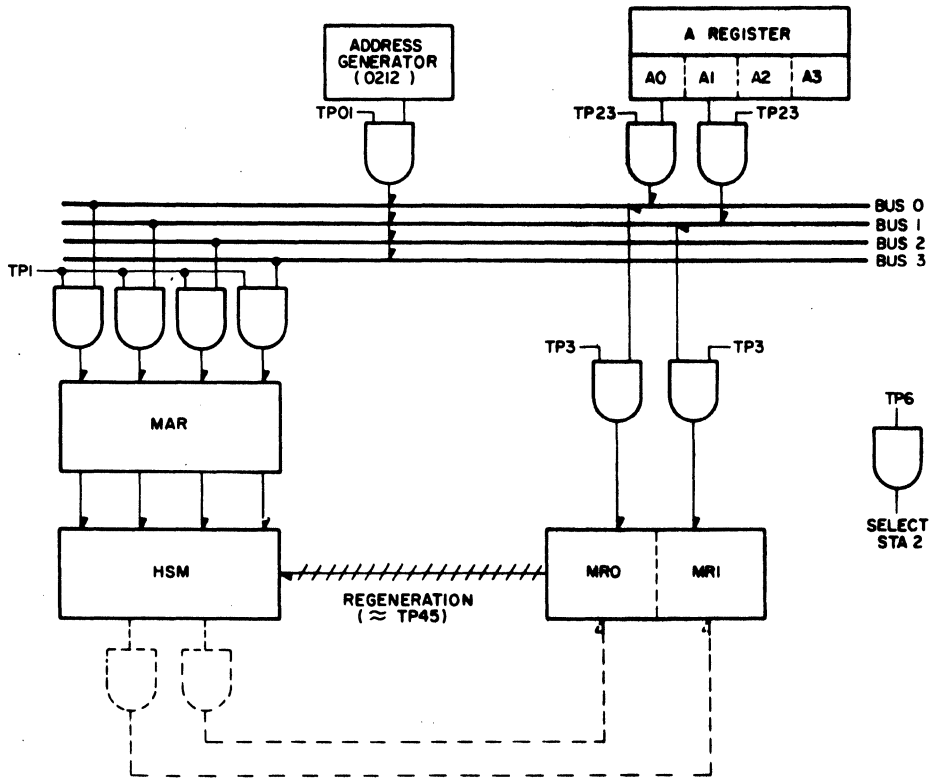


Figure 33 STA 1 Status Level

	STA 2
TP01	Generate 0214 onto Bus from Address Generator.
TP1	Gate contents of Bus into MAR. Generate command level and decode MAR address.
TP23	Gate A address onto Bus. Inhibit two characters being read from reaching MR.
TP3	Gate contents of Bus 2 into MRO and contents of Bus 3 into MR1.
TP45	Regenerate.
TP6	Select P1 status level.

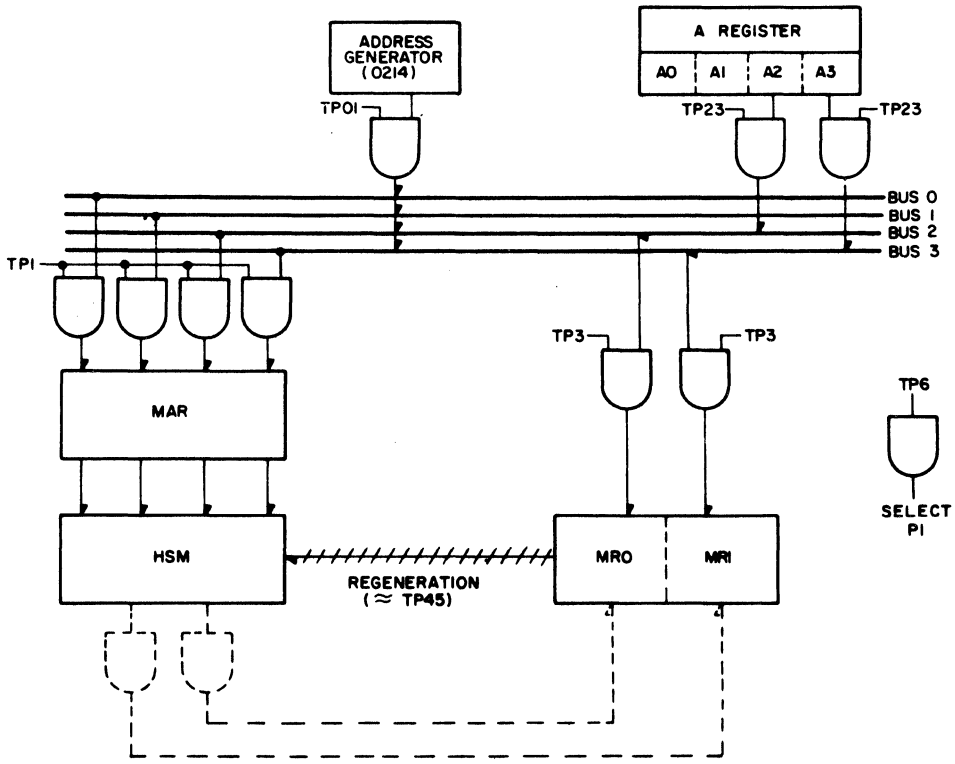
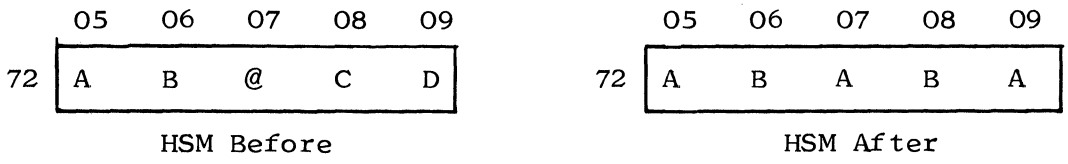


Figure 34 STA 2 Status Level

4. Programming Errors

The DSL and DSR instructions may go awry if care isn't taken with the address. Transferring an overlapping field was described as a problem in the DL and DR instructions and this problem also exists in the DSL and DSR instruction. However, in the DSL and DSR instructions the end result is more serious in that the selected symbol(s) will be eliminated and the instruction will continue to transfer until a WTT alarm occurs.

Example #6: # @ 7205 7207



The characters A B are inserted in memory up through the highest address thereby wiping out any symbols. A WTT would occur when location 0000 was addressed if WTAB was not set. If WTAB is set, the computer would cycle through memory without terminating the instructions.

Example #7: P $ 4325 4324

"Twos" would be transferred to every location down to 0199 and a WTT alarm would occur.

PRACTICE PROBLEMS

6. Write an instruction to move 30 characters from 1020-1049 to 5030-5059.

7. Memory is first filled completely with *'s. Then a word of indeterminate length is written into HSM starting at 1000.

   Use one instruction to move the entire word to 5000. (1000 is the location of the first letter of the word) How would you determine quickly how many letters the word contained?

8. With the WTAB button depressed, execute the following program:

```
1000  J  *  0000  0000
1010  P  *  0000  0000
```

How can the DSR instruction in this program be used to determine the size of memory?

9. Given initial HSM contents and the following program to be executed, what are final HSM contents, A final, and the final contents of 0212-0215?

```
1000  P  *  2009  2009
1010  M  4  0212  1032
1020  N  4  0215  1039
1030  J  Ø  2000  2009
```

	00	01	02	03	04	05	06	07	08	09
20	P	A	C	E	S	*	O	N	E	S

HSM Before

	00	01	02	03	04	05	06	07	08	09
20										

HSM After

At end of execution: A_f = _____

Final Contents:

	12	13	14	15
02				

10. With the WTAB button depressed, execute the following program and give the required information:

```
1000  #  *  2001  2000
1010  P  *  2009  2008
1020  J  *  0001  0000
```

	00	01	02	03	04	05	06	07	08	09
20	P	A	C	E	S	*	O	N	E	S

HSM Before

	00	01	02	03	04	05	06	07	08	09
20										

HSM After

At end of execution: A_f = _____

Final Contents.

	12	13	14	15
02				

E. K - LOCATE SYMBOL LEFT (LSL)

L - LOCATE SYMBOL RIGHT (LSR)

These instructions search a designated area for a selected symbol. Both instructions terminate upon finding a non-selected symbol or upon searching the entire designated area (A-B Equality). The LSL instruction operates from left to right and the LSR instruction operates from right to left. The final contents of the A register are stored in STA during both instructions. The PRI's (Previous Result Indicators) are set during execution of the LSL and LSR with their indications as follows:

PRN is set when the first character searched is not equal to contents of N.
PRZ is set when all characters are equal to contents of N.
PRP is set if a non-symbol is found in the specified HSM area after a character equal to the contents of N has been found.

The PRI's are flip-flops which may be sensed (using the CTC instruction) during the execution of a program. Control is transferred to different points in the program depending upon which PRI is set.

Final Register Contents:
If all characters searched are equal to N

$A_f = B_i$

$B_f = B_i$

If a non-selected symbol is found

LSL: A_f = One location to left of non-selected symbol. $B_f = B_i$

LSR: A_f = One location right of non-selected symbol. $B_f = B_i$

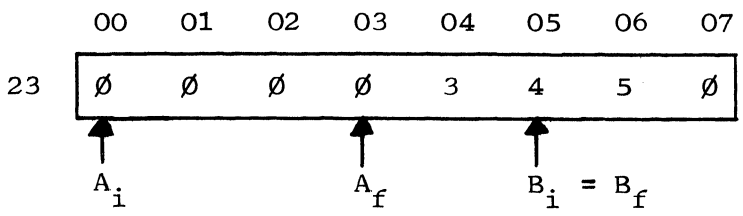
NOTE: See pages V-5 and V-7 of Programmers' Reference Manual for operations summary.

1. Instruction Format

Op. Code	N	A	B
K or L	Selected Symbol	Leftmost Location to be Searched in K Instruction-Rightmost In L Instruction	Rightmost Location to be Searched in K Instruction-Leftmost in L instruction

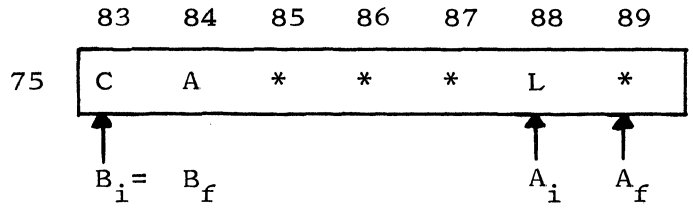
2. Instruction Execution

Example #1: K Ø 2300 2305



HSM Before and After -- PRP would be set.

Example #2: L * 7588 7583

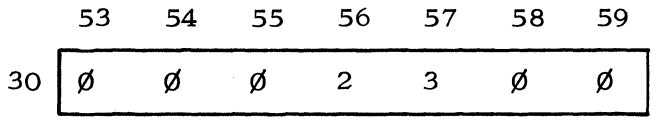


HSM Before and After -- PRN would be set.

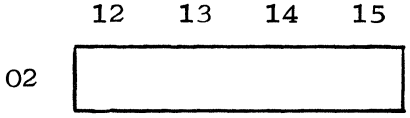
PRACTICE PROBLEMS

1. If the following instruction were executed, what would locations 0212-0215 hold, and what PRI would be set?

L Ø 3058 3055



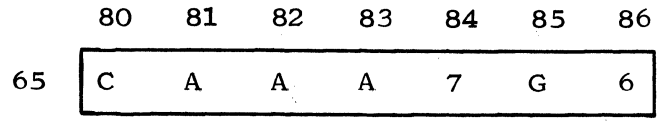
HSM Before and After



PRI?_____

2. What PRI would be set if the following instruction were executed?

```
K  A  6581  6583
```



HSM Before

PRI - _____

3. A quantity exists in memory between locations 4000 and 4007. The number of digits this quantity contains is unknown, but is is known that non-significant zeros precede the most significant digit (MSD).

Example: MSD

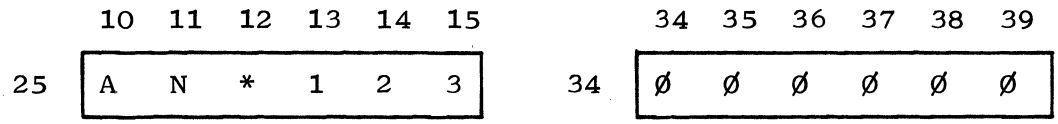
OOOXXXXX

Non-significant zeros — Quantity

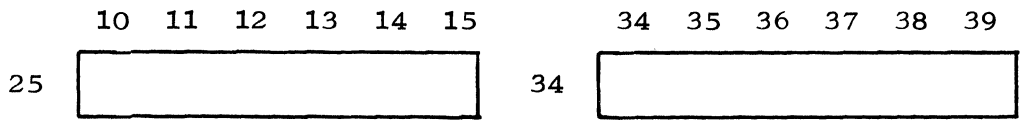
The total number of characters is always 8. Write an instruction which would locate the position one to the left of the MSD for any case.

4. Execute the following program and show final HSM contents.

```
5000  P  *  2515  3439
5010  K  Ø  3434  3439
5020  N  4  0215  5039
5030  J  *  3434  3434
```



HSM Before



HSM After

3. Machine Operation

In this section we will introduce a new format for describing the sequence of events during the execution of an instruction. This new format is less cluttered and will make it easier for the student to follow the "status flow" or sequence of status levels. For example, in the new format "A → BUS" will replace "The A address is gated onto the bus." "$\overline{(D2=D3)}$" means "The character in the D2 register is unequal to the character in the D3 register."

Termination of the LSL/LSR instruction is a bit more complex than that of any previous instruction. There are three things to be done: (1) the appropriate PRI must be set, (2) the contents of the A register must be adjusted properly, and (3) the final contents of the A register must be stored.

Notice that PRZ is set at TP2 of the P5 status level which completes staticizing of the LSL/LSR instruction. If the instruction is completed without finding a non-selected character (i.e. terminates on A-B equality) PRZ remains set.

Consulting the status flow chart (Figure 35) we find that as long as neither A-B equality nor a non-selected character have been found, the instruction loops through A1 and X1 status levels, performing one A1 and one X1 for each character examined. The A1 status level is used to bring the N character to D3 and the character addressed by the A register to D2 for comparison. The X1 status level is used to (a) set PRN if a non-compare is found in the first location checked, or to (b) set PRP is a non-compare is found in some succeeding location.

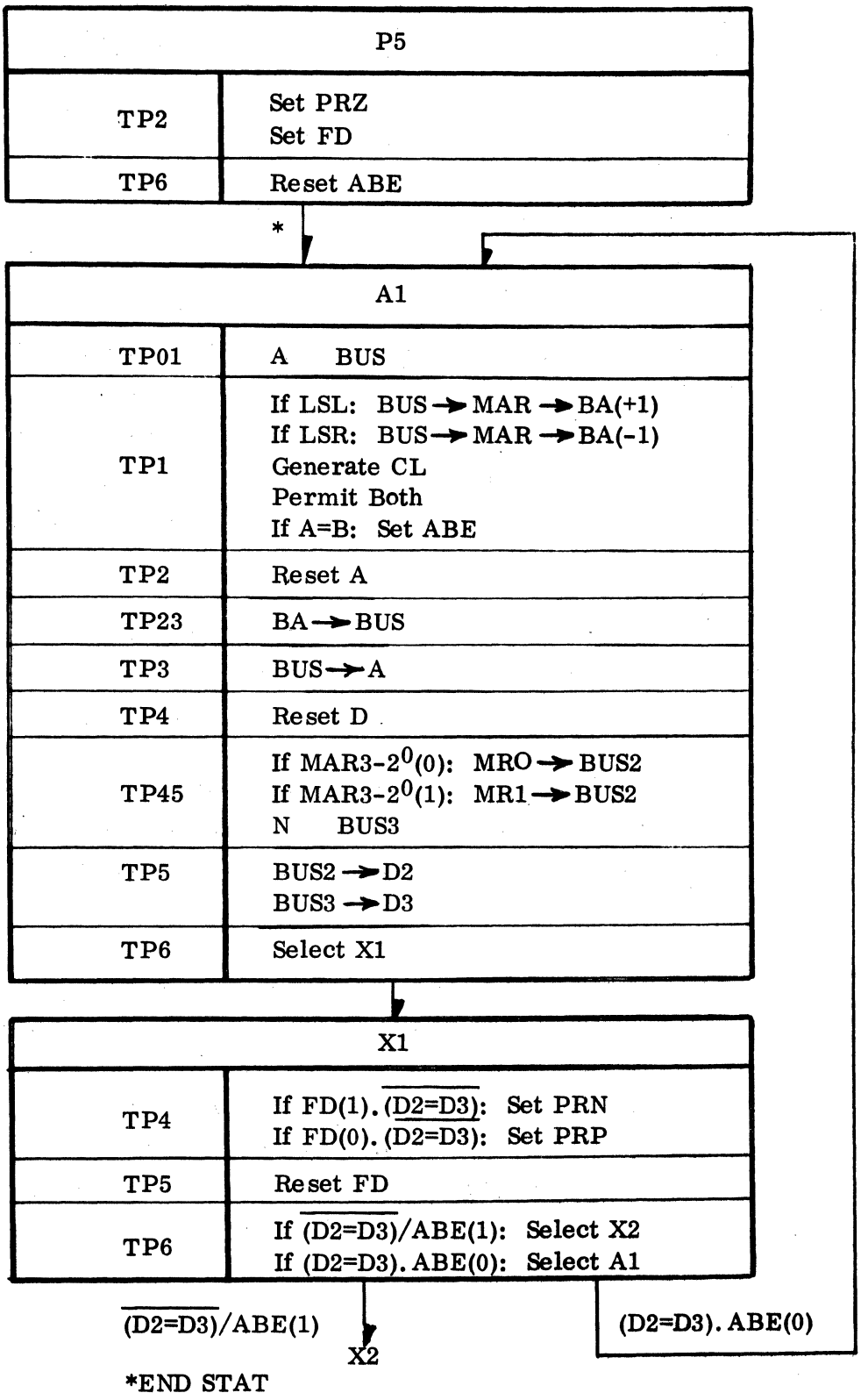


Figure 35 A1 and X1 of LSL/LSR

The method used to determine whether or not a character being examined is the first character examined by the instruction depends upon the setting of the FD (first digit) flip-flop at TP2 of P5 during staticizing of the instruction. FD remains set until TP5 of the first X1 status level when it is reset. Thus when checked at TP4 of an X1 when D2=D3 if FD is set it means "This is the first X1 status level since P5" indicating that the first character addressed is now being compared to the N character in the D register.

From examining TP4 of the X1 status level we see that PRN is set if the first character is a non-selected character, and PRP is set if any succeeding character is a non-selected character.

Finally, if a non-selected character is found or A-B equality is reached an X2 status level is selected at TP6 of the X1. The X2 status level is used to modify the the A final address. Examining TP1 of the X2 we see that if a non-selected character has been found, the A address is adjusted by two in a direction opposite to that of its movement during the execution of the A1 status level.

The address thus produced will be that of the last selected character to be examined, provided that at least the first character examined was a selected character. If the first character examined was a non-selected character, the final address will then be one to the left of this character for LSL or one to the right of the character for LSR. If no non-selected character is found, the adjustment produces an A final address which is that of the last character of the field, that is, the same as the B initial address.

The X2 status level selects a STA1 status level, which in turn automatically selects a STA2 status level. The STA1 and STA2 store the newly adjusted A final address.

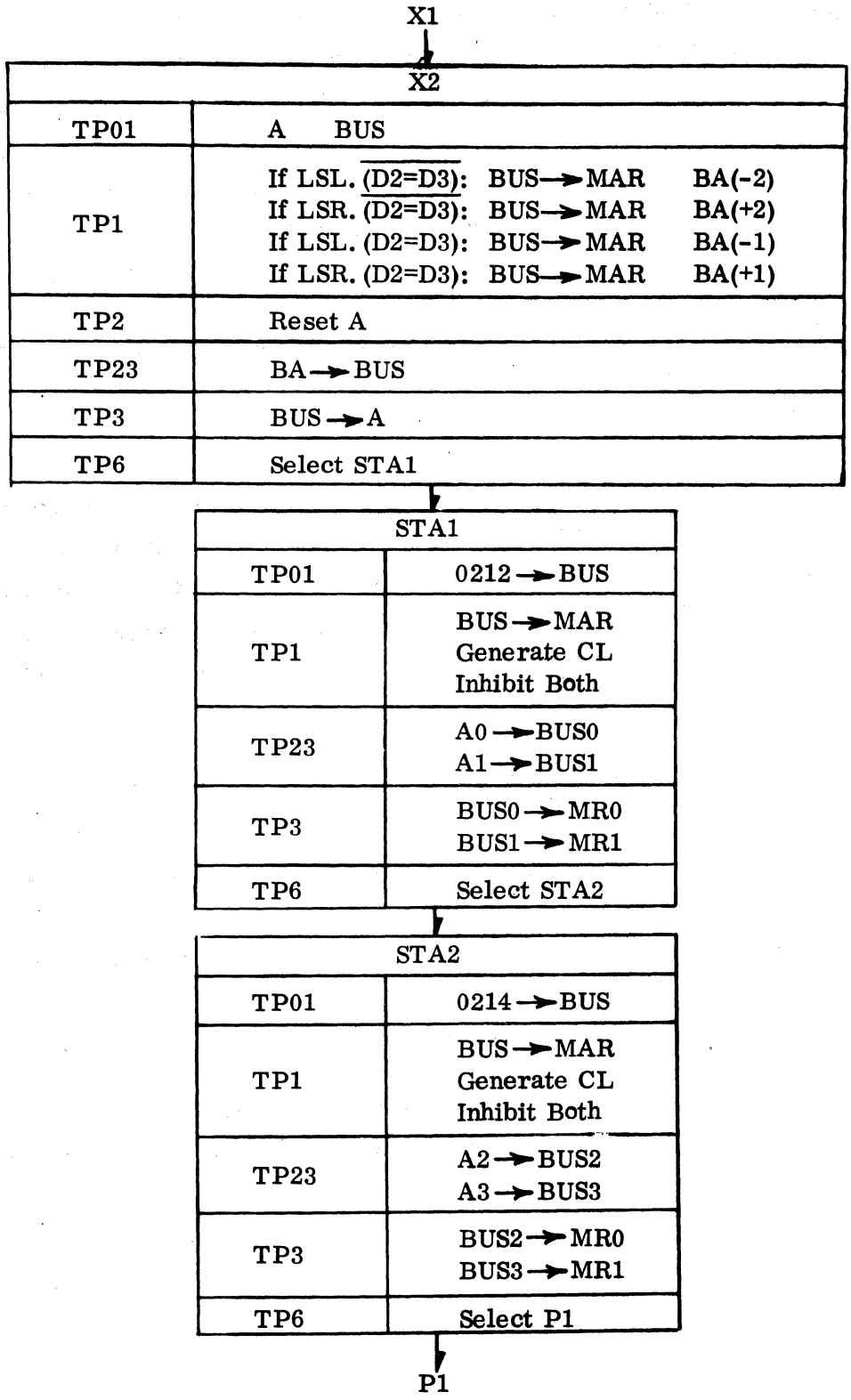


Figure 36 X2 of LSL/LSR; STA1 and STA2

4. Programming Errors

In the LSL and LSR instructions memory is not changed, only examined, hence very little can go wrong with any combination that is written. However, it is possible to obtain incorrect PRI settings and final STA contents by reversing the A and B addresses. For example suppose the field to be examined exists between 1003 and 1008 but the instruction was written as:

	03	04	05	06	07	08
10	-	-	-	3	6	5

K - 1008 1003

The final A address would be 1007 and PRN would be set while if the instruction were written correctly:

K - 1003 1008

The final A address would be 1005 and PRP would be set. Note that although the field limits are misaddressed, the instruction can still terminate since a non-selected symbol as well as A - B equality halts the operation.

PRACTICE PROBLEMS

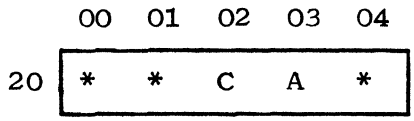
5. It has been found that the D2 register equals the D3 register and the processor has selected an X2 status level of an LSL instruction. What causes the selection of the X2 STL?

6. What is the purpose of the X1 of the LSL/LSR? The X2?

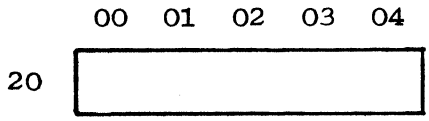
7. A 10K HSM is first filled with Ø's, and then a word of indeterminate length is inserted at an arbitrary location higher than 2000. You are asked to find: (a) the starting location of the word, and (b) the number of letters in the word. (The word will contain no numeric data.) How would you do it?

8. Execute the following program and show HSM final.

```
1000  K*  2000  2004
1010  M4  0212  1032
1020  M4  0212  1036
1030  JR  2000  3000
```



HSM Before

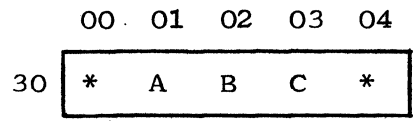


HSM After

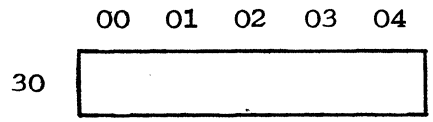
What PRI will be set? _____

9. The following instruction is manually inserted and executed. What are the HSM final contents? What is A_f? Which PRI will be set?

```
L*  3001  3000
```



HSM Before



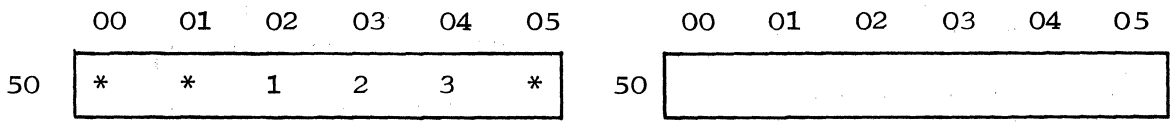
HSM After

A_f = _____ Which PRI_____

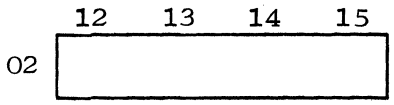
Is ABE set_____.

10. What is HSM content after execution?

```
1000  K*  5000  5005
1010  M4  0212  1022
1020  L1  5002  5000
1030  N4  0215  1045
1040  J*  5004  5004
```



What PRI is set _____



F. A-TRANSLATE (TRA) Repeatable

This instruction translates a specified number of characters in a designated area from one binary code to another by use of a translate table. The table can be stored anywhere within memory and must be previously inserted before the Translate instruction is executed. The characters which are to be translated are replaced by their equivalent from the table. The TRA instruction operates from left to right and does not go through STA.

NOTE: An operations summary of the Translate instruction may be found on Page V-3 of the Programmer's Manual.

A_f = One location to the right of last character translated.

$B_f = B_i$

1. Instruction Format

Op. Code	N	A	B
A	Number of Characters to be Translated (0-44)	Location of Leftmost Character to be Translated and its Result	Location of First Character of Translate Table. (B_2B_3 must be (00)

NOTE: If N = Ø, no characters are translated and the next instruction in sequence is executed (i.e., P5 of the TRA selects P1).

2. Instruction Execution

The character to be translated is read out of memory and split into two octal digits. The 2^0, 2^1, and 2^2 bits form one digit and this digit is inserted in the D3 register. The 2^3, 2^4, and 2^5 bits form the second octal digit which is placed in the D2 register in the 2^0, 2^1, and 2^2 positions. The 2^3, 2^4, and 2^5 bits in both D2 and D3 are left as zero bits thus forming two decimal numbers.

Example:

Character to be translated:

	2^5	2^4	2^3	2^2	2^1	2^0
C =	0	1	0	0	1	1

2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0

D2 initially

2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0

D3 initially

The character C is split into 0 1 0 and 0 1 1 and the bits are inserted into the 2^0, 2^1 and 2^2 positions of D2 and D3 as follows:

2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	1	0

$= 2_{10}$

D2 after

2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	1	1

$= 3_{10}$

D3 after

The octal numbers created from the splitting process are now the decimal numbers 2 and 3.

The first two digits of the B Address are then combined with the contents of D2 and D3 to form an address which is addressing within the translate table. If the table starts at address 5500, then the address 5523 would be constructed in this example. At location 5523 should exist a combination of binary bits which represents the translated character for the letter C. This translated character is read from the table and written over the original letter C as addressed by A. The A Address is then incremented by one and the next character is translated in a similar fashion. Translation continues until the N count is reduced to zero.

Example: A 3 1000 2700

	00	01	02	03
10	A	B	C	D

	21	22	23	24
27	-	J	K	L

A_i

HSM Before

	00	01	02	03
10	-	J	K	D

	21	22	23	24
27	-	J	K	L

A_f $B_f = 2700$

HSM After

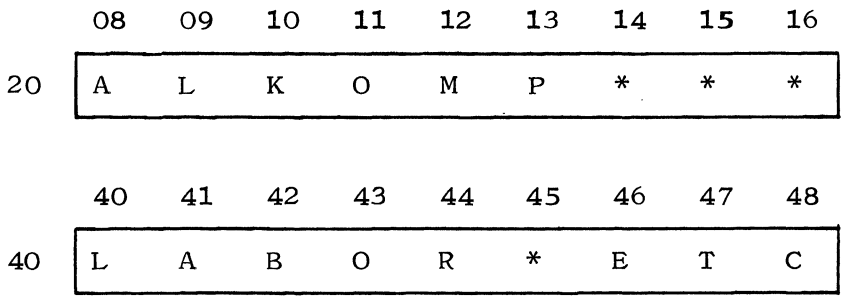
In the above example, the 301 characters A, B and C were translated into 501 code. Note that in locations 1000, 1001 and 1002, after the instruction has been executed, exists octal numbers 40, 41 and 42, respectively. Octal numbers 40, 41, and 42 are 501 characters A, B, and C.

Translation from 301 to 501 or vice versa is not the only form of translation that can be done with the TRA instruction. On the contrary, any code can be translated to any other code. All that is required is the proper translate table.

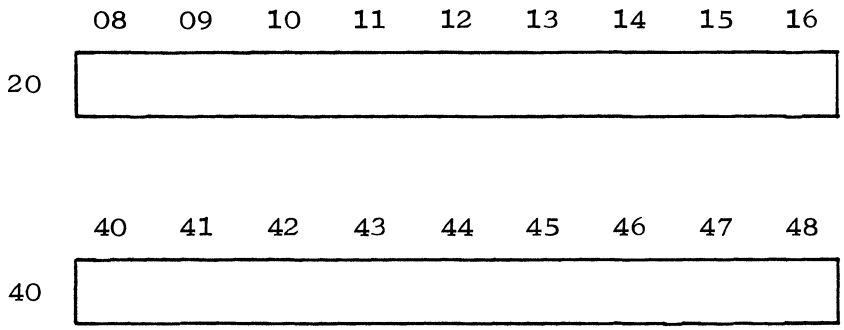
PRACTICE PROBLEMS

1. Execute the following instruction and show final HSM contents.

A 4 2010 4000



HSM Before



HSM After

2. Give the table addresses of the proper characters which are necessary to change memory by the instruction shown below.

A 7 1452 3600

	50	51	52	53	54	55	56	57	58
14	T	R	A	N	S	L	Z	T	E

HSM Before

	50	51	52	53	54	55	56	57	58
14	T	R	O	P	I	C	A	L	*

HSM After

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

Table location_____ should contain a_____.

3. Machine Operation

The translate instruction requires three status levels: an A1, or D, and an A2. The A1 status level brings a character out of memory and puts the

2^3, 2^4, and 2^5 bits into the 2^0, 2^1, and 2^2 positions of D2 and the 2^0, 2^1, and 2^2 positions of D3.

The D status level gates the contents of D2 and D3 register (plus parity bits if needed) onto BUS 2 and BUS 3, and simultaneously gates the B register contents onto BUS 0, BUS 1, BUS 2, and BUS 3 all at TP01.

Since B2 and D2 are both gated onto BUS 2 and B3 and D3 are both gated onto BUS 3, you might expect to find "garbage" on BUS 2 and BUS 3. But remember, B2 and B3 must both be equal to ZERO; and furthermore, the parity bits in B2 and B3 are both inhibited from reaching the buses. Therefore, no bits from B2 or B3 actually reach BUS 2 or BUS 3 (unless there is a programming error). So finally, B0, B1, D2, and D3 are used to address memory. The character thus addressed is gated via BUS 3 into D3. (See Figure 37, A1 and D of TRA)

The A2 status level again addresses the location where the character we are translating is located. (Notice that during the A1 status level BA(+0) was used so the A address is the same now.) Then the character from D3 is written into this location in HSM. There are ways to leave the A2 status level: either REP1, P1, or A1 may be selected. REP1 is used only if the translate instruction is in the field of a repeat instruction, which will be covered later. If the N count has not yet gone to ZERO, the next character must be translated, so an A1 (of the translate instruction) is selected. If the N count has been exhausted (and the translate instruction is not being repeated using the repeat instruction) a P1 is selected to start staticizing the next instruction.

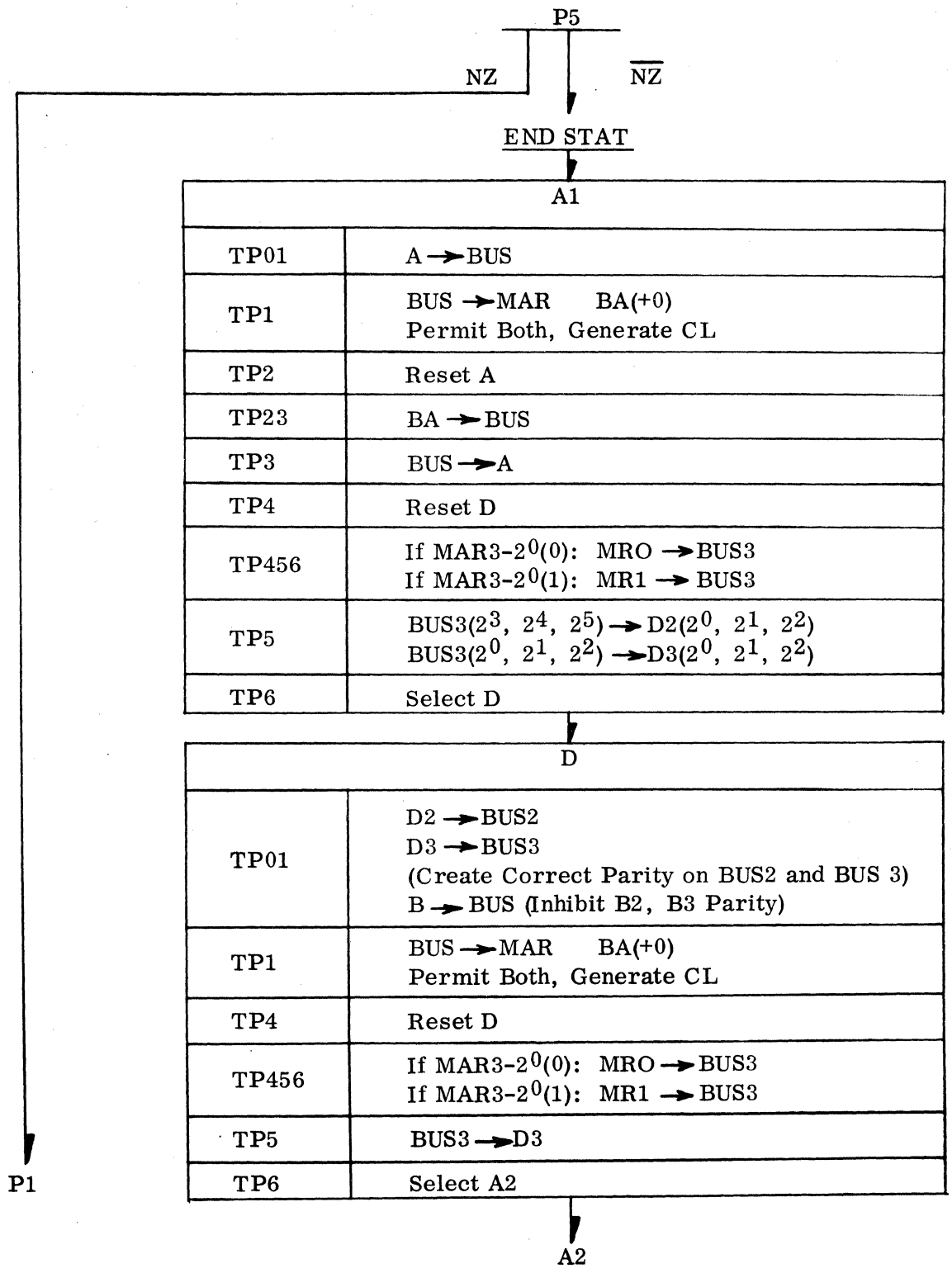


Figure 37 A1 and D of TRA

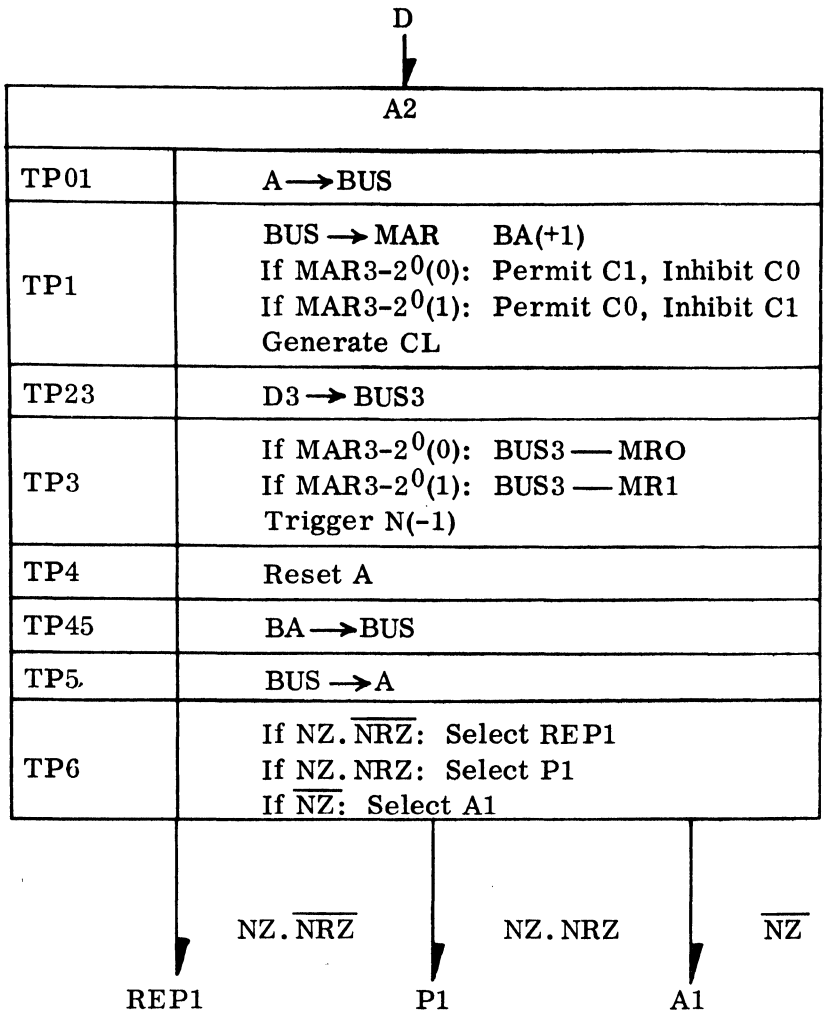


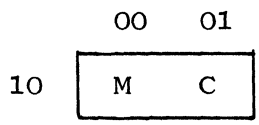
Figure 38 A2 of TRA

4. Programming Errors

Besides an incorrect table or an N character outside of the N count (described in DL and DR instructions), the Translate instruction can be misused by making the initial B2 and B3 characters something other than zero. During the process of creating the table address (D status level) D2 and D3 are gated onto Bus 2 and Bus 3 and the B address excluding B2 and B3 parity is gated onto all four bus lines. Thus, if some bit combination existed in the last two characters of the B address originally, the voltage levels would combine on BUS 2 and BUS 3 and probably create bad parity

which would produce a MAPE alarm.

Example: A 2 1000 5502



Character M = $(44)_8$ = 1 100 100

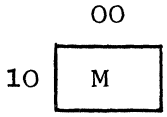
D2 and D3 would each create a decimal four. During the D status level, the B address 5502 is gated out onto the BUS and 44 is placed on BUS 2 and BUS 3. BUS 3 would receive the following:

From D3 = 0000100 = 4
From B3 = 0000010 = 2
Result
on Bus 3 = 0000110 = 6 with bad parity

This would cause a MAPE alarm.

It is possible that the end result would have good parity and no alarm would occur, however, the resulting table address may not be the correct one.

Example: A 1 1000 5503



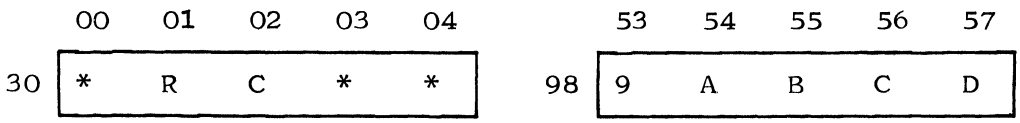
From D3 = 0000100 = 4
From B3 = 0000011 = 3
NOTE Parity is inhibited
Result on
Bus 3 = 0000111 = 7 with good parity

Constructed table address would be 5547 instead of 5544 as it should be.

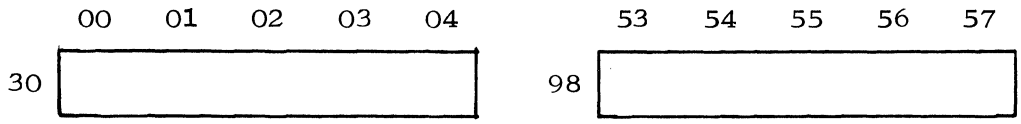
PRACTICE PROBLEMS

3. Describe the execution of the following instruction:

A 1 3003 9803



HSM Before



HSM After

Any alarm lights? _____

4. In problem 3. the translate table started at 9800. Where did it end? Could we start a translate table at 4550? Explain your answers.

5. Given the following instruction and the contents of HSM <u>after</u> its execution, determine the initial HSM contents.

A5 2055 5000

	55	56	57	58	59
20	7	Ø	/	1	5

	30	31	32	33	34
50	/	1	Ø	7	5

HSM After

	55	56	57	58	59
20					

	30	31	32	33	34
50					

HSM Before

6. There is a character located in HSM at location 8919. Write a two-instruction program to determine whether the character is Group I, II, III, or IV and if:

Group I - Place a "1" in location 4001

Group II - Place a "2" in location 4002

Group III - Place a "3" in location 4003

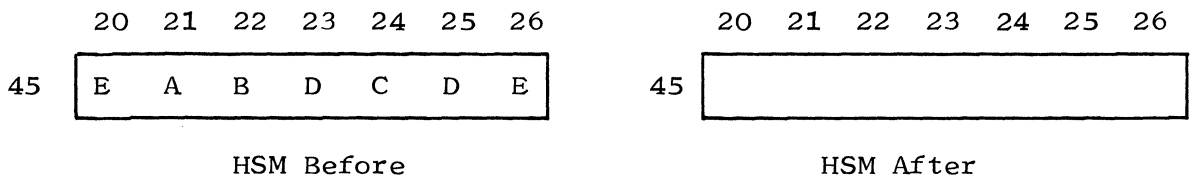
Group IV - Place a "4" in location 4004

HSM may be prepared in any way you wish before executing the two instructions, but be sure to mention these preparations in your answer.

7. During the D status level, which register receives the modified address from the Bus Adder?

8. Show HSM final contents.

A7 4520 4500



G. INDIRECT ADDRESSING

Many times in programming it is necessary and/or advantageous to use the final register contents of some preceding instruction as the A or B address of another instruction. In previous problems a Transfer Data instruction has been used to perform this operation. There is another method, however, which in most cases involves fewer instructions and less computer time. This second method is called indirect addressing.

An indirect address is not the ultimate address which will be used in execution of the instruction, but is instead the address of a storage location which will contain the ultimate or direct address. The computer will recognize an indirect address by the presence of the 2^4 bit in the LSD. For example, the address 0215 is a direct address since the 2^4 bit is a zero in the number 5 (000101). On the other hand, 021E is an indirect address since the LSD does contain a 2^4 bit (E=010101). Note that the information bits, 2^0 through 2^3, still represent a decimal 5. Therefore, an E is a 5 with a 2^4 bit.

In order that the process of indirect addressing be carried out correctly, the indirect address must specify the rightmost diad of the storage locations housing the direct address. Thus, to use the contents of STA (1006) as the A Address of an M instruction, the instruction would be coded as follows:

```
            M    2    021E    2000

                12   13   14   15
          02  | 1    0    0    6  |
```

In this example, the contents of memory locations 0214 and 0215 (rightmost diad) would be stored temporarily in the D Register. During this operation, the address 021E will be modified by -2. The A Register would then hold the address 021C, which specified the leftmost diad. The contents of this diad (1 and 0) are then read from memory and combined with the contents of the D register (0 and 6). Finally, all four characters are gated into the A

Register as the address 1006. The M instruction then would be executed as though it were written:

M 2 1006 2000

NOTE: - The same result could have been obtained using 021D, since the indirect address must initially specify the rightmost diad.

An indirect address is not restricted to standard locations. One can indirectly address any location in memory. Nor is the indirect address associated with only the A Address of an instruction. The A Address, the B Address, or both can be indirect addresses. Also, an indirect address may specify the location of another indirect address. The computer will continue to bring out four new characters for every indirect address, until it finds an address which does not contain a 2^4 bit in the LSD.

Example #1: P * 021H 021E

Instruction Initially

	12	13	14	15
02	3	0	7	8

	16	17	18	19
02	4	9	5	2

HSM Before and After Indirect Addressing

P * 4852 3078

Instruction Just Prior to Execution

Example #2:

Assume the following instruction in memory starting at 1000

```
1000  J  *  200H  200G
```

	04	05	06	07	08	09
20	9	7	6	4	5	3

HSM Before and After

J * 6453 9764

Instruction Just Prior to Execution

Example #3:

```
1000  M  2  101I  101E
1010  N  1  102I  102E
1020  J  *  103I  103E
1030  .  0  2034  2032
```

	30	31	32	33	34	35	36
20	A	B	C	D	E	F	G

HSM Before Execution of Program

Prior to execution of each instruction, the respective addresses shown below would be placed in the A and B Registers as:

```
M  2  2032  2034
N  1  2034  2032
J  *  2032  2034
.  0  2034  2032
```

	30	31	32	33	34	35	36
20	A	B	*	*	*	D	G

HSM After Execution of Program

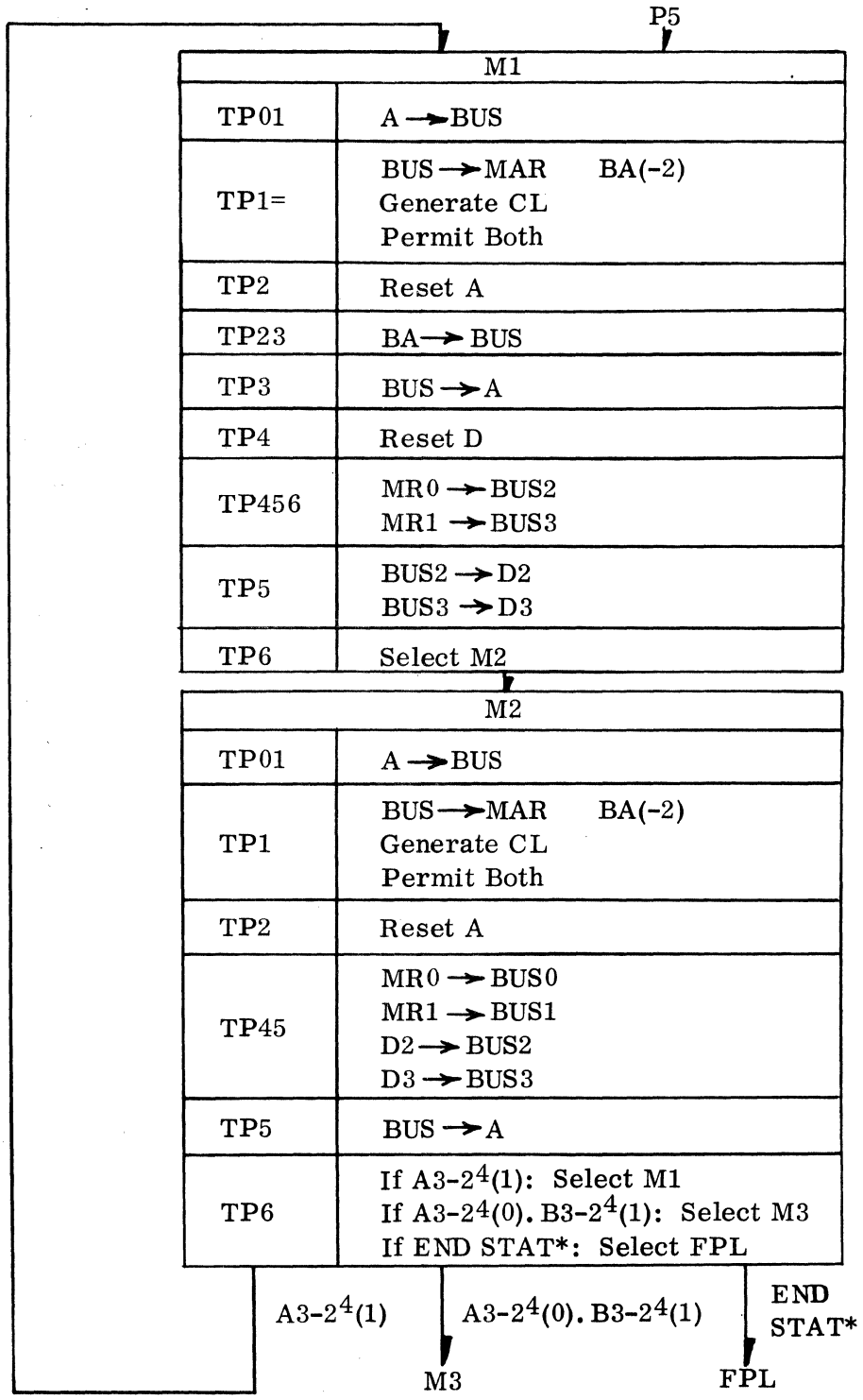
It should be noted that the original instruction is unchanged in memory, i.e., indirect addresses still exist in the memory. The instructions are brought out one at a time and modified in the registers during indirect addressing and then executed.

1. Machine Operation

The process of replacing an indirect address by the contents of the location addressed until there is no 2^4 bit in the LSD of the A or B addresses must all take place before execution of the instruction. Since this process, performed by the M1, M2, M3, and M4 status levels, is completed before the first processing level (FPL) is selected, the process is considered part of staticizing.

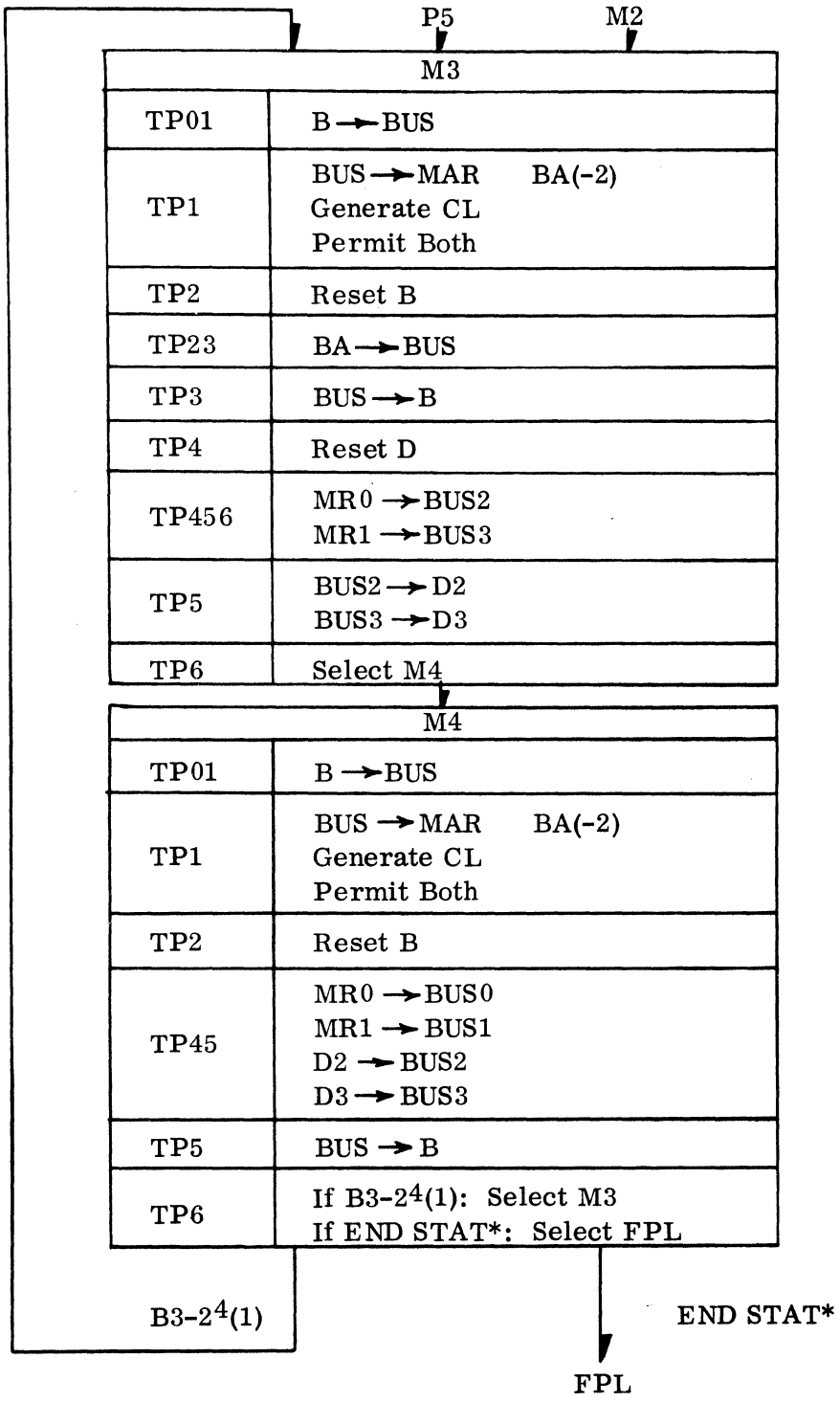
The M1 and M2 (Figure 39) status levels are used to replace the address in the A register with the contents of the location addressed. At TP6 of M2, if there is no longer a 2^4 bit in the LSD of the A address, either an M3 or the FPL is selected. An M3 is selected only if the LSD of the B address contains a 2^4 bit. In fact, if only the B address is indirect, the P5 status level will select the M3 directly. M3 and M4 (Figure 40) bring the contents of the location addressed by the B register into the B register. When there is no longer a 2^4 bit in the LSD of the B address at TP6 of M4, the FPL is selected.

Figure 41 shows the complete staticizing procedure. Notice that if an instruction which uses the N REGISTER as a counter or selector is staticized with N = 0, P1 will be selected at TP6 of P5 even if the instruction has an indirect address. This may be expressed symbolically: NZ.RINZ.P5 P1. Figure 41 defines "RINZ" and also defines END STAT.



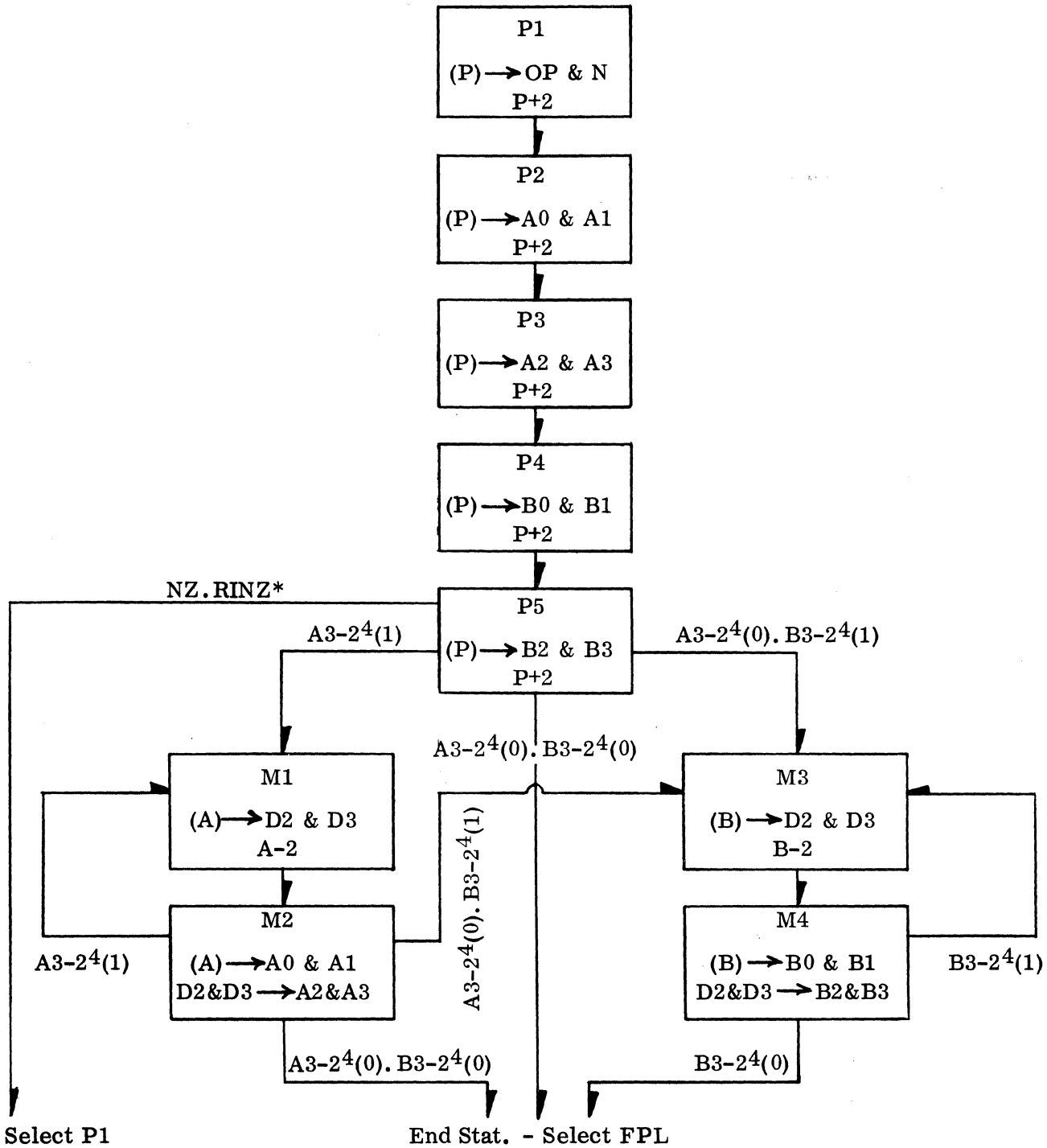
*END STAT = $A3\text{-}2^4(0).B3\text{-}2^4(0)$

Figure 39 M1 and M2 Status Levels



*END STAT = $B3\text{-}2^4(0)$

Figure 40 M3 and M4 Status Levels



*RINZ = Any instruction that uses the N Register as a counter or selector (i.e. COM/DL/DR/ADD/SUB/OR/AND/EXO/TRA/REG/CTC)

Figure 41 Staticizing Block Diagram

2. Programming Errors

The most common error in the use of indirect addressing results from an attempt to use as a final address four characters the leftmost of which is in an odd location. Notice that the M1 and M2 (or M3 and M4) status levels pull two diads out of HSM. Therefore, the final address must be the four characters of the two diads, the leftmost character of which is in an even location.

Example: K 1 400D 5500 is the instruction in HSM.

	00	01	02	03	04	05	06	07	08
40	5	4	4	Ø	Ø	Ø	2	3	7

The programmer may intend the address "4400" to replace the indirect A address, but actually the final A address just prior to execution will be "4000." An indirect address of 400D or 400E will bring the same two diads out of HSM, so, the address pulled out of HSM by an indirect address will always start at an even location.

PRACTICE PROBLEMS

1. Write a two-instruction program that is equivalent to the following four instruction program.

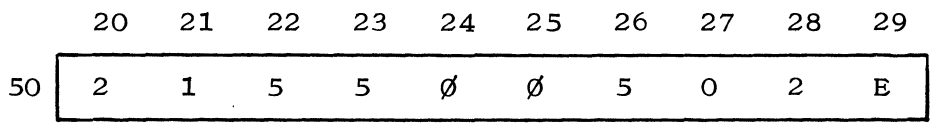
```
1000  P  *  2009  2009
1010  M  4  0212  1032
1020  N  4  0215  1039
1030  J  0  2000  2009
```

2. Write a five-instruction program equivalent to the following three-instruction program.

```
1000  K  *  5000  5005
1010  L  1  021D  5000
1020  J  *  021E  5004
```

3. How many status levels are used to staticize the following instruction?

```
M  4  502I  6000
```



HSM Before

What does the instruction in the registers look like just prior to execution?

4. How many status levels are used to staticize the following instruction?

```
J  *  200D  200H
```

	00	01	02	03	04	05	06	07	08	09
20	1	2	2	Ø	Ø	C	0	0	B	*

HSM Before

What does the instruction in the registers look like just prior to execution. Describe the execution.

5. Show the HSM contents just prior to execution, and after execution: 5020 → 5029 in problem three and 2000 → 2009 in problem four.

6. At some location greater than 1000 in a 10K memory which has been cleared to Ø's, an "X" has been placed. Write a program to fill HSM from 1000 to the "X" with *'s. Use indirect addressing.

7. Describe the staticizing of the following instruction.

```
J  2  400V  400N
```

	00	01	02	03	04	05	06
40	2	3	4	7	1	5	A

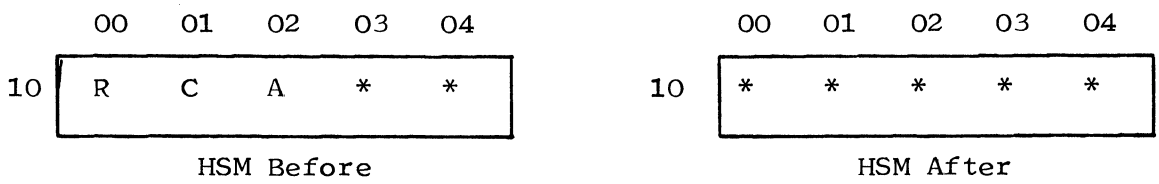
HSM

8. A block of information read into HSM from tape starts at location 2000 and consists of four sub-blocks separated from each other by *. There is also a * at the end of the fourth sub-block. Write a program to transfer sub-block #1 to 5000, sub-block #2 to 6000, sub-block #3 to 7000, and sub-block #4 to 8000.

ANSWERS TO PRACTICE PROBLEMS

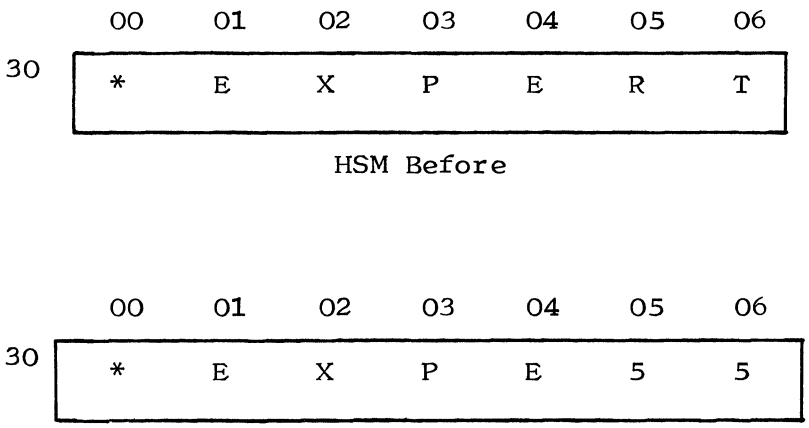
Pgs. II-56, 57, 58, 62

1. J * 1000 1002



2. JE 2003 2003

3. At end of first instruction: J5 3005 3006



At end of second instruction: JD 3002 3002

	00	01	02	03	04	05	06
30	*	E	X	P	E	5	5

HSM Before

	00	01	02	03	04	05	06
30	*	E	D	P	E	5	5

HSM After

At end of last instruction: J * 3004 3006

	00	01	02	03	04	05	06
30	*	E	D	P	E	5	5

	00	01	02	03	04	05	06
30	*	E	D	P	*	*	*

Answer

4. J Ø 9996 9999

5. J - 5235 5237

   A_f=5238, B_f=5237

7. a. 10,000 locations will be filled. First 0001 if filled, then 0002 is filled, and so on until 9999 is filled. The next and last location to be filled in 0000 since for a 10K memory machine 9999 + 1 = 0000 since 9999 is the largest number the A register can hold. When 0000 is filled, A-B equality occurs and when ABE is checked at TP6 of this last A2 status level, a P1 status level is selected and the instruction terminates. Note, however, that at TP4-TP5 time of this last A2 status level, the A register had already been modified by +1, so the A_f=0001.

8. 10,000. See explanation to No. 7.

9. There are three ways to approach this problem. The most straightforward method is to use the instruction:

J * 0000 I 999

Note on your 301 code card that I is equivalent to 19, so the instruction above will fill, starting at 0000, each location in memory through 19,999 or every location in the 20K memory.

A second method makes use of what you learned in problem number six and is a bit more elegant since it can be used to fill a memory of any size, even if this size is unknown to the operator. You could use the instruction:

J * 0001 0000

This will fill memory starting with 0001 to the "top" of memory (be it 10K, 20K, or 40K) and then "wrap-around" to fill 0000 as the last location. ABE will be set while 0000 is being filled and the A_f=0001.

The third method is to use in the B address a non-numerical character such as ";"

J * 0000 :000 for instance

Write an instruction to fill every location in a 20K memory with *'s. (Assume WTAB is depressed.)

Since the A register is augmented numerically, it will never hold a configuration such as ;000 so A-B equality will never be reached and the computer will continue wrapping around memory, filling every location many times, looking for A-B equality. Since ABE is never set, the instruction will never terminate, and must be stopped by use of the OCSP button.

Note that it makes no difference which A initial address we use since all memory will be filled many, many times before you can push OCSP. Also, note that any non-numeric character in any digit location of the B address will give the same result (provided that the B register supplies all the needed bit positions in the chosen digit position -- care

must be exercised expecially in B_1 and B_2).

10. a. Since the A-B equality circuitry doesn't look at the 2^6 bits, the 0000 in the A register and the blank B register will set ABE. Thus, one location (0000) will be filled with Ø.

    b. A_f=0001, B_f=blank. Computer does stop.

    c. Since WTAB was depressed, there are no errors. If WTAB had not been depressed, a WTT error would have been generated.

11.

	00	01	02	03	04
10	A	B	*	D	E

a. MR at TP4:

MRO	MR1
*	D

b. "C" will not appear in the MR. It is inhibited from reaching MRO so that the N character (*) can be gated in instead.

12. The D will appear in MR1 during the second A2 status level--actually during the time that a "4" is being inserted into location 1002. It will appear at about TP23 time of the second status level and remain in the MR until the MR is reset at TP1 of the third A2 status level. (See page 2-1 of STATUS FLOW manual. This shows the functions performed during every status level).

Pgs. II-67 & 75

1.

	10	11	12	13	14	15	16	17	18	19	20	21	22	23
02	A	5	6	3	2	1	6	3	2	1	3	2	0	6

HSM After

2.

	58	59	60	61	62	63
15	A	B	C	E	F	F

HSM After

3. N/5038 7196 (or M/5008 7166)

4.

	00	01	02	03	04	05	06
31	*	*	C	A	S	E	*

HSM After

5. DL increments the A and B addresses; DR decrements them.

6.

	50	51	52	53	54	55	56
22	E	X	E	X	E	X	E

HSM After

A_f=2255, B_f=2257

7.

	50	51	52	53	54	55	56
22	E	X	E	X	A	M	P

A_f=2249, B_f=2251

8. Transfer Data Right

9.

	00	01	02	03	04	05	06	07	08	09
20	A	N	D	A	1	A	N	D	A	2

HSM After

Note that the instruction at 1020 changes the instruction at 1030 to read N2 3008 2003 before 1030 is staticized.

10. M5 1000 1001

	00	01	02	03	04	05
10	*	*	*	*	*	*

HSM After

Pgs. II-79, 81, 89, 90, 91

1. P * 4813 4821

2.

	25	26	27	28	29	30	31
88	E	E	S	U	L	T	S

HSM After

	12	13	14	15
02	8	8	2	7

HSM After

3. The selected symbol (/) would be destroyed and HSM would be filled with 4's down to 0199-WTT alarm.

4.

	00	01	02	03	04	05	06
20	*	3	6	8	7	9	*

	00	01	02	03	04	05	06
30	Ø	*	3	6	8	7	9

	12	13	14	15
02	3	0	0	2

HSM After

5.

	30	31	32	33	34	35
33	B	@	B	@	@	C

	12	13	14	15
02	3	3	3	4

HSM After

6. M "1020 5030 or N" 1049 5059

7. # * 1000 5000. To quickly determine how many letters the word contained, examine the A final address by selecting the A register. A - 1000 - the number of letters in the word +1. The "two extra" are due to the facts that an *, not a letter of the word was the last to be transferred, and even after this transfer the A register will have been incremented by one more. So the number of letters in the word + A_f - 1001.

8. The DSR instruction will transfer the * from 0000 to 0000 and the A register will decrement by one going to the top of memory. Then by either examination of A final or of the contents of 0212-0215, memory size may be determined. 9999 → 303A, I999 → 304A, Z999 → 305. Note that the instruction "N1 0000 0000" will allow you to determine HSM size by examination of the A final. This is simpler for manual determination. But also notice that "N" doesn't store A final. So is HSM size is to be determined in a program, the "P" instruction will have to be used in order to store the top HSM address.

9.

	00	01	02	03	04	05	06	07	08	09
20	P	A	C	E	Ø	*	O	N	E	S

A — HSM After

	12	13	14	15
02	2	0	0	4

Final

A_f = 2005

The important points in this problem are: the fact that none of the instructions following the DSR affect the store A area; and the fact that the information in 0212 - 0215 is used to fill both the A and B addresses of the SF instruction.

10. The instruction at 1010 will fill memory with "S's" and continue looping around HSM looking unsuccessfully for a *. Thus A_f will depend upon when the operator hits OCSP to terminate the cycling, and both locations 2000-2009 and 0212-0215 will contain all "S's."

Pgs. II-93, 94, 99, 100, 101, 102

1.

	12	13	14	15
02	3	0	5	8

PRP Set

2. PRZ set

3. KØ 4000 4007

4.

	10	11	12	13	14	15
25	A	N	*	1	2	3

	34	35	36	37	38	39
34	*	*	*	1	2	3

HSM After

5. Set a PRI, adjust A_f, store A_f.

6. The X1 is used to set the PRI's; the X2 is used to adjust A final.

7. One method is to use a LSL instruction to locate the last Ø before the word. $A_f + 1$ = starting address of word. Then using $A_f + 1$ as your starting A address, do a DSL from $A_f + 1$ to $A_f + 1$. This will stop after finding the first Ø on the other side of the word. The final A address of the DSL minus one will be the address the last character of the word.

8.

	00	01	02	03	04
20	*	R	C	A	*

PRP will be set.

HSM After

9.

	00	01	02	03	04
30	*	A	B	C	*

A_f = 3002. PRP is set.

ABE is not set.

Since the first character examined was a non-selected character, we would expect to set PRN. However, we inserted this instruction into the registers manually, so there was no P5 status level to set the FD flip-flop. On a non-compare, PRN is set only if FD is set at TP4 of the X1 status level. Since in this case FD will not be set, PRP is set instead of PRN.

10.

	00	01	02	03	04	05
50	*	*	*	*	*	*

	12	13	14	15
02	5	0	0	2

PRN is set (by the LSR at 1020).

Pgs. II-105, 106, 111, 112, 113

1.

	08	09	10	11	12	13	14	15	16		40	41	42	43	44	45	46	47	48
20	A	L	B	E	R	T	*	*	*	40	L	A	B	O	R	*	E	T	C

HSM After

2.

Location	3621	contains	O
Location	3645	contains	P
Location	3662	contains	I
Location	3643	contains	C
Location	3671	contains	A
Location	3663	contains	L
Location	3625	contains	*
Location		contains	

3.

	00	01	02	03	04		53	54	55	56	57
30	*	R	C	D	*	98	9	A	B	C	D

The B address must be 9800 not 9803. Because of this programming error, the D status level will address 9857 instead of 9854. Since parity will be correct, there will be no error lights.

4. It ended at 9876. Actually the table could go to 9877, but = is the 301 character with the highest binary bit value (octal 76). 77 is the largest octal number possible with six data bits so the table must stop at 9877. In fact it must start at XX00 and extend to XX77 due to the method used for translation, so you would not find a translate table starting at 4450.

5.

	55	56	57	58	59		30	31	32	33	34
20	.	+	H	I	;	50	/	1	Ø	7	5

HSM Before

6. Prepare HSM as follows. Fill with: "0" from 4000 thru 4004
"1" from 5000 thru 5017
"2" from 5020 thru 5037
"3" from 5040 thru 5057
"4" from 5060 thru 5077

Insert the following instructions at HSM locations shown:

8900 A1 8919 5000
8910 M1 8919 400?

The question mark in location 8919 indicates that this location contains the unknown character and was not changed when inserting the two instructions. The translate instruction will replace this unknown character with a 1, 2, 3, or 4 depending upon the group to which it belongs. Then the 1, 2, 3 or 4 will be written by the DL instruction to 4001, 4002, 4003 or 4004, respectively.

7. The Bus Adder is not gated to the BUS during the D status level.

8.

	20	21	22	23	24	25	26
45	D	A	B	C	C	C	C

HSM After

Pgs. II-121 through 126

1. 1000 P * 2009 2009
   1010 J 0 021E 021E

2. 1000 K * 5000 5005
   1010 M 4 0212 1022
   1020 L 1 0000 5000
   1030 N 4 0215 1045
   1040 J * 0000 5004

3. 9. (P1-P5, M1, M2, M1, M2). M 4 5500 6000

4. Eleven status levels. When staticized, the instruction looks like this: J * 1220 002@, where the B2 and B3 characters have had parity. However, since the B register doesn't check parity and is not used to address memory during execution, no alarm occurs until a WTT at location 0000. If WTAB is depressed, the instruction will not terminate since "@" is not a BCD character and thus A-B equality will not be reached.

5. In neither problem 3. nor problem 4. will the contents of HSM be altered by the indirect addressing procedure. In problem 4. all memory will be filled with * by execution of the instruction.

6. 0900 K 0 1000 9999
   0919 J * 1000 021E

7. P1-P5 would be performed correctly. M1 selected. When the contents of the A register are used to address HSM during M1, a MAPE will occur since the 2^5 bit of the "V" was dropped creating bad parity in A^3.

8.
```
1000  #  *  2000  5000
1010  #  *  021E  6000
1020  #  *  021E  7000
1030  #  *  021E  8000
```

SECTION III

DECISION AND CONTROL INSTRUCTIONS

A. DECISION AND CONTROL INSTRUCTIONS, INTRODUCTION

The Decision and Control Instructions might be considered the heart of modern data processing machines. One instruction in particular, the Conditional Transfer of Control (CTC), is responsible for making several different types of decisions (one per instruction), and then directing the path the computer should take. It simulates human reasoning. A similar instruction within the group is the Input-Output Sense (IOS), which is involved with decisions about the peripheral equipment. The seven decision and control instructions are:

OP Code	Instruction
V	STORE REGISTER
W	CONDITIONAL TRANSFER OF CONTROL
Y	COMPARE LEFT
X	TALLY
.	HALT
R	REPEAT
S	INPUT-OUTPUT SENSE

B. V - STORE REGISTER (REG)

The REG instruction is used to transfer the contents of a selected register (P, A, B, S, or U) into four specified HSM locations. The P address transferred will be the address of the REG instruction plus 10. The A or B address stored is the A or B of the instruction immediately preceding the Store Register instruction.

The A address of the REG instruction specifies the address of the rightmost diad where the four characters of the selected register are to be stored. The only exception is when the A register (A of the previous instruction) is to be stored. In this case, STA1 and STA2 are used and A is stored in 0212-0215, the standard STA location. The last four characters of the REG instruction are not used unless the contents of the P register are being stored. In this case, the B address of the REG instruction will be gated into the P register. The program will then continue by sta-

ticizing and executing the instruction at the newly created P address.

The N character determines which register will be stored. A list of N characters and their corresponding register is given under "Instruction Format."

NOTE: Pages VII-3 and VII-4 of the Programmers' Reference Manual give an operations summary of the REG instruction.

1. Instruction Format

Op. Code	N	A	B
V	See Table Below	Address of Rightmost Diad to Receive Contents of P, B, S, or U Register. A Register Stored Automatically in STA, if N=2.	Ignored Unless Storing P Register. Address of next Instruction to be executed, if N=1

REGISTER TO BE STORED	N CHARACTER
P	1
A	2
B	4
S	8
U	&

$A_f = A_i - 2$ if P, B, S, or U are stored; if A is stored, A_f of previous instruction.

$B_f = B_i$ if P, S, or U are stored; if A or B, B_f of previous instruction.

2. Instruction Execution

Example 1:

Assume B Register contains the address 2963 prior to executing the following instruction:

V 4 3725 7186

The B Register contents (2963) will be stored at address 3725 and the B Address of the instruction (7186) will be ignored. Memory after execution of the B instruction would be as follows:

	22	23	24	25
37	2	9	6	3

NOTE: If N = 0, no register is stored and next instruction in sequence is executed.

Had the A Address of the instruction been 3724, the same result would occur since this is the other half of the diad. A diad always consists of an even address on the left and an odd address on the right. If the A Address were 3723 initially, then the 6 and 3 would have been placed in locations 3722 and 3723, respectively, and the 2 and 9 in locations 3720 and 3721, respectively.

Example 2: V 1 2003 1030

If the above instruction is located in memory, beginning at address 1000, the P Register would hold the address 1010 after staticizing. This store instruction specifies storing the P Register contents at address 2003 and transferring control to the B address 1030. Therefore, memory would hold the following after executing the instruction.

	00	01	02	03
20	1	0	1	0

The contents of the B Register, 1030, would then be placed in the P Register. The address in the P Register is always that of the next instruction to be executed. Thus, the operation code of the next instruction to be executed is located at address 1030.

Practice Problems:

1. List the four memory locations which would receive the contents of the selected register in the following instruction. Assume the A Register holds 6034, prior to staticizing of this instruction.

   V 2 5061 3025

   a. HSM Location _____ will contain _____.

   b. HSM Location _____ will contain _____.

   c. HSM Location _____ will contain _____.

   d. HSM Location _____ will contain _____.

2. If the Operation Code (V) of the following instruction is located at address 7750 in memory, show the contents of memory between addresses 1032 and 1035 after execution of the instruction.

   V 1 1035 8000

   32 33 34 35

   10 [ ]

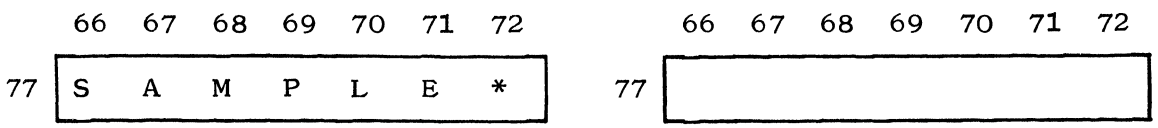
   What would the P Register hold after execution of the above instruction?

   P [ ]

3. Write an instruction to store the S Register contents in memory locations 5106, 5107, 5108, and 5109.

4. Execute the following program showing final HSM contents.

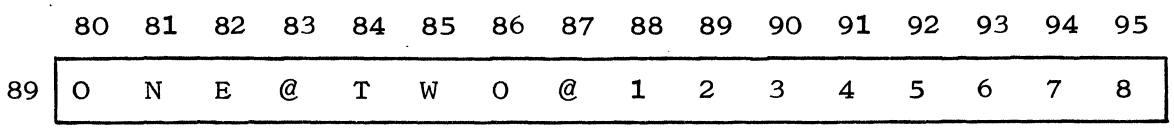
```
2000   N   1   7770   7772
2010   M   3   7766   7770
2020   V   2   0000   0000
2030   N   4   0215   2049
2040   J   *   7766   0000
```



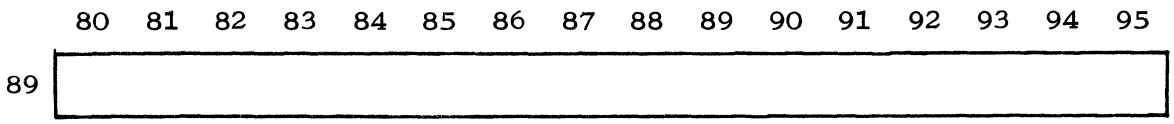
HSM BEFORE HSM AFTER

5. Execute the following program showing final HSM contents.

```
3000   #   @   8980   8988
3010   V   4   3039   0000
3020   M   4   0212   3032
3030   #   @   0000   0000
```



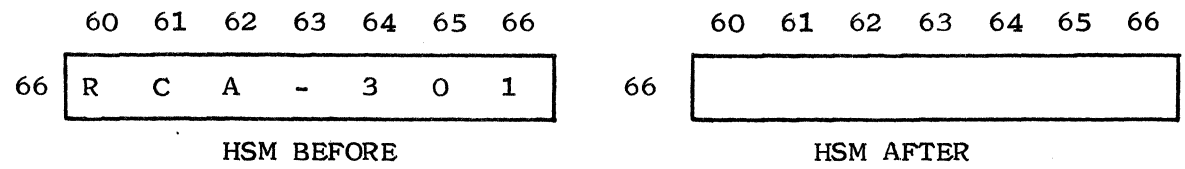
HSM BEFORE



HSM AFTER

6. Execute the following program showing final HSM contents.

```
1000  N  3  6662  6666          3000  J  *  6665  6665
1010  V  4  1039  0000          3010  V  1  0219  0000
1020  V  1  3019  3000
1030  N  2  6662  6666
```



HSM BEFORE HSM AFTER

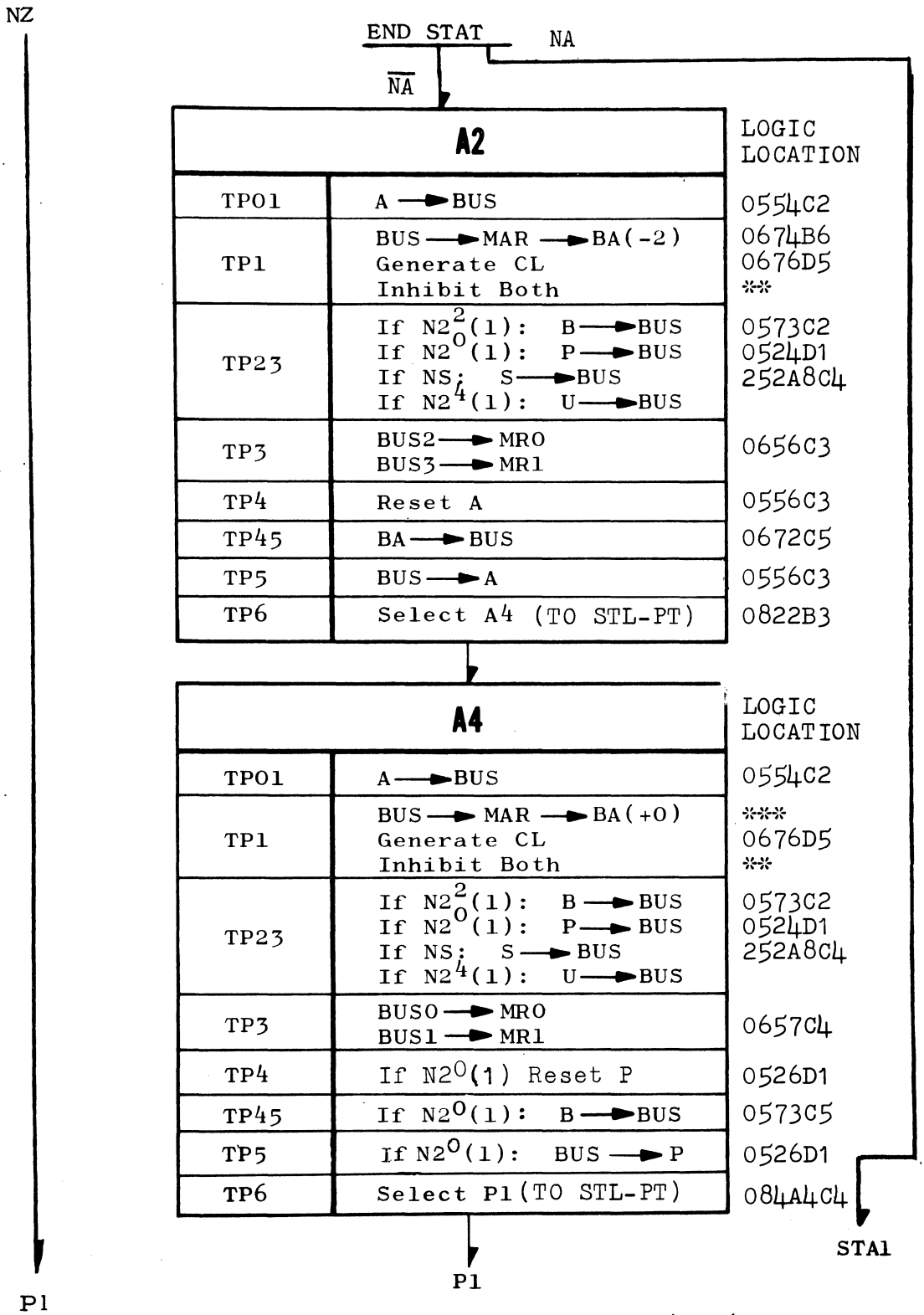
3. Machine Operation

The Store Register instruction uses two status levels for execution, either an A2 and A4 or a STA1 and STA2. During TP6 of P5, if the N character is a "2" (indicating that the A Register is to be stored) a STA1 is selected. STA1 and STA2 are then used to store the A Register.

If the N character is not a "2", an A2 will be selected. The A2 and A4 status levels are used to store the P, B, S, or U Register. The contents of the selected register will be stored at the location specified by the A address of the Store Register instruction.

The B address of the Store Register instruction is not used unless the P Register is being stored. In this case, the contents of the B Register are gated into the P Register at TP5 of the A4 status level. Thus, the B address of the Store P instruction indicates the address of the next instruction to be executed.

Remember, the purpose of the Store Register instruction is to store the final register contents of the preceding instruction. Staticizing, however, will destroy the A-final and B-final addresses of the previous instruction. Therefore, something must be done during staticizing to preserve the A or B address of a store A or store B instruction.



** Absence of Permit Both on 0677C6
*** Absence of BA +1, +2, -1, -2

Figure 42 A2 and A4 of a REG

Refer to Page 2-8, 2-9 amd 2-10 in the Status Flow Manual (Staticizing). At TP4 of P2, the A Register is reset if $\overline{\text{"REG} \cdot \text{NA"}}$. "REG" means that a Store Register instruction code is in the NOR register and "NA" means that a 2^1 bit is present in the N register indicating that the A Register is the one whose contents are to be stored. If a Store A instruction is being staticized, the A register is not reset at TP4 of P2 and the new A address being brought out of HSM is not gated into the A register at TP5 of P2 and P3.

The B Register will be reset at TP4 of P4 if $\overline{\text{REG} \cdot (\text{NA}/\text{N2}^2)}$. That is, the B register will <u>not</u> be reset if <u>either</u> a store A or store B is being staticized. (N2^2 indicates a 2^2 bit is present in the N register.) Also, if <u>either</u> a store A or store B is being staticized, the B address of the store instruction is not gated to the B register at TP5 of P4 and P5.

The question naturally arises: "Why is the B_f of the previous instruction saved during a store A?" Assume that you wanted to store both the A-final and the B-final of the previous instruction. If you do a store B instruction and then a store A instruction, the A address stored will be the A address (-2) used to store the B register contents. If, however, you do a store A instruction and then a store B instruction, you will successfully store A_f and B_f of the instruction immediately preceding the two store instructions. This is possible only because the store A instruction saves <u>both</u> the A_f and B_f of the previous instruction.

4. Programming Errors

The Store Register instruction can be used incorrectly thereby creating strange results by having an N character other than those which are specified or by storing certain constants in the B address of a V8 or V & instruction.

If the N character is 2, 3, 6, 7 or any character with a 2^1 bit, the instruction will automatically go through STA with no alarms, since a 2^1 bit in the N register automatically selects STA1. If, however, the combination is 5, 9, A, D, H or some other character which specifies more than one register to be stored, the contents of the designated registers will all be gated onto the Bus at the same time and more than likely bad parity will result

causing a MRPE alarm. The computer only examines the single bit of the N character set aside for each register.

2^0 = P Register
2^1 = A Register
2^2 = B Register
2^3 = S Register
2^4 = U Register

A one bit in the respective position is sufficient to gate the contents of that register onto the bus during an A4 status level. EXAMPLE: Register contents prior to staticizing V instruction:

P	A	B
1020	2360	5143

Instruction at 1020 is V 5 7004 0000.

Register contents after staticizing V instruction:

P	A	B
1030	7004	5143

Since the N character is a 5 (000101 excluding parity) both the P and B registers are specified to be stored. Therefore, during the A4 status level addresses 1030 and 5143 are gated onto the Bus.

Bus 0 receives a 1(0000001) and a 5 (1000101) resulting in a 5 (1000101) with good parity.

Bus 1 receives a 0 (1000000) and a 1 (0000001) resulting in a 1 (1000001) with bad parity.

Bus 2 receives a 3 (1000011) and a 4 (0000100) resulting in a 7 (1000111) with bad parity.

Bus 3 receives a 0 (1000000) and a 3 (1000011) resulting in a 3 (1000011) with good parity.

The resulting address on the Bus is 5173 with bad parity in the C1 and C2 characters thus causing a MRPE during the A4 status level.

Care must be exercised in specifying the rightmost diad of the four locations which are to receive the contents of the designated register. For example, assume one desires to store the contents of the B register in locations 1066, 1067, 1068, and 1069. The instruction must be:

```
V    4    1068    0000
          or
V    4    1069    0000
```

If the instruction is written using the A address 1066 or 1067; then locations 1064, 1065, 1066, and 1067 would receive the contents of B.

It should be noted that the B address is ignored in all cases of the Store Register instruction except when N = 1. However, the B address is staticized for N = 8 and N = &. Therefore, if constants are placed in memory in the locations which make up the B address of a V8 or V& instruction and a 2^4 bit exists in the LSD position, the computer will go through indirect addressing possibly causing a MAPE or MRPE alarm. EXAMPLE:

```
V    8    1032    -END
```

The above instruction would cause a MAPE alarm during indirect addressing (M1 status level) because bad parity is created by the loss of the 2^5 bits of the C0 and C2 characters and the 2^4 bit of C1 while placing -END into the B register.

A MRPE can also be generated by trying to store the contents of a non-existing register. For instance, if a "V8" is attempted on a computer with no Simultaneous Mode or a "V&" is attempted on a computer with no File Mode, nothing will be gated onto the BUS at TP23 time of the A2 status level, and a MRPE will occur since no bits will be present in the MR.

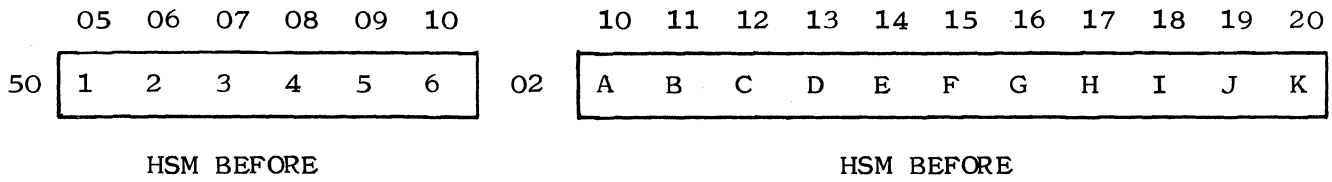
However, attempting a "V9" on a computer with no Simultaneous Mode, will cause a "V1" to be performed. In the case, since the S Register is non-

existent, it will not be gating any information onto the BUS for comparison with the P Register bits to create a MRPE.

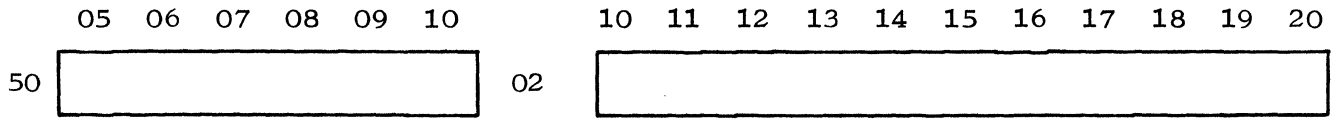
PRACTICE PROBLEMS

7. Execute the following program and show the final HSM contents indicated.

```
1000   V1   0218   1010
1010   V2   0219   2250
1020   V4   5009   1020
```



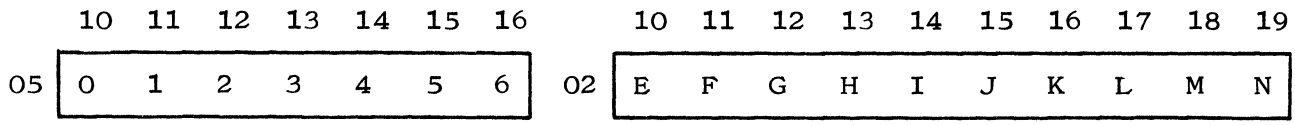
HSM BEFORE HSM BEFORE



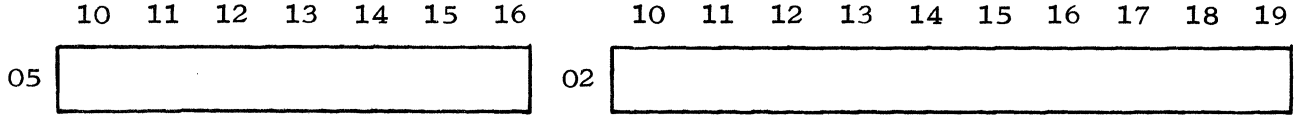
HSM AFTER HSM AFTER

8. Execute the following program and show the final HSM contents.

```
1000   J*   2000   3000
1010   V4   0515   1010
1020   V2   0218   1000
```



HSM BEFORE HSM BEFORE



HSM AFTER HSM AFTER

9. Describe execution of the following program. What will be stored and where will it be stored? What is the P address after execution? Any alarm lights?

```
1000   M4   2000   3000
1010   V3   5025   1030
```

10. Describe execution of the following program. What will be stored and where will it be stored? What is the P address after execution? Any alarm lights?

```
1000   N2   2001   3000
1010   V5   0218   0218
```

C. W - CONDITIONAL TRANSFER OF CONTROL (CTC)

The Conditional Transfer of Control instruction is the principal decision-maker used by the 301. During the execution of a program, there will be times when the next step will depend upon the outcome of a group of operations already performed. For instance, assume a certain number of orders are to be processed. The total number of orders is stored as a two-digit quantity in HSM. After processing an order, this two-digit quantity is decremented by one, indicating that there is one less order still to be processed. Now a decision based on the present state of the two digit quantity must be made. (1) If the quantity is a non-zero positive number, there are more orders still to be processed and control must be transferred to the "top" of the program in order to repeat these operations on the next order in line. (2) If the quantity is equal to zero, all orders have been processed and control must be transferred to a termination routine. (3) If the two-digit quantity is a negative number, a mistake has been made and control is transferred to an error routine. (It is impossible to process more orders than the number of orders available.) The CTC instruction could be used to make the three-way decision described above.

The CTC instruction makes decisions by: (1) sensing certain levels which indicate the present state of the processor and (2) transferring control to either the next instruction in sequence, the instruction addressed by A of the CTC, or the instruction addressed by B of the CTC. To transfer control to the instruction addressed by A of the CTC means to insert the A address of the CTC into the P register and select a P1. In this case, the next instruction executed would be the instruction whose P address is the same as the A address of the CTC instruction.

The N character of the CTC instruction determines which indicator will be sensed. The N codes and their corresponding indicators as well as the conditions for selecting the A or B address for control transfer are listed under "Instruction Format." If neither of the conditions necessary for selecting the A or B address is met, the next instruction in normal sequence will be staticized and executed. The instruction does not go through STA,

but does go through STP (STORE P) which is an abbreviation for storing the P register contents in 0216 through 0219. This is done on transfer only as a means of saving for future reference the location from which control was transferred.

NOTE: Pages VII-5 and VII-6 of the Programmer's Reference Manual give an operations summary of the CTC instruction.

1. Instruction Format

Op. Code	N	Sense	A	B
W	1	PRI's	Address of Next Instruction, if One Set of Conditions Described Below is True.	Address of Next Instruction, If One Set of Conditions Described Below is True.
	2	Overflow Indicators		
	4	Simultaneous Indicator		
	8	ED/EF Normal Indicator		
	&	Interrupt Indicator		
	- (minus)	ED/EF Simultaneous Indicator		

A Address

N = 1 and PRP is set.

N = 2 and First Overflow Indicator is set.

N = 4 and a Read instruction is in Simultaneous Mode.

N = 8 and the EF/ED Normal Indicator is set.

N = & and the Interrupt (INT) Indicator is set.

N = - and the EF/ED Simultaneous Indicator is set.

B Address

N = 1 and PRN is set.

N = 2 and neither Overflow Indicator is set.

N = 4 and Write instruction is in the Simultaneous Mode.

N = 8 and the EF/ED Normal Indicator is not set.

N = & and the Interrupt Indicator is not set.

N = - and the EF/ED Simultaneous Indicator is not set.

If N = 0, or N = 1 and PRZ is set, or N = 2 and Second Overflow Indicator is set, or N = 4 and Simultaneous Mode is not busy, no transfer of control takes place and the next instruction in sequence is executed.
$A_f = A_i$ and $B_f = B_i$.

2. Instruction Execution

Example:

Write an instruction that will transfer control to 3750 if PRP is set, or 3760 if PRN is set.

W 1 3750 3760

Practice Problems:

1. Write an instruction that will transfer control to address 5580 if the INT button is not set or to address 4320 if INT is set.

---

2. Assume the following instruction exists in memory beginning at address 2000. If the First Overflow Indicator is set when this instruction is executed, what will memory locations 0216 through 0219 contain to show :

W 2 3120 2050 ?

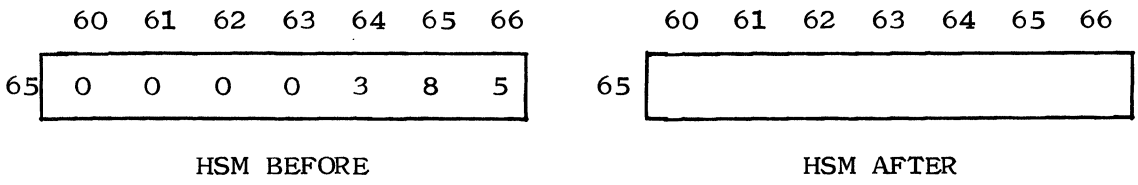
	16	17	18	19
02				

3. Write an instruction which will sense the ED/EF normal indicator. When indicator is set, transfer control 4650; when not set, transfer to 4630.

---

4. Execute the following program showing final HSM contents.

```
1500   K   0   6560   6566
1510   W   1   1530   1530
1520   J   *   6566   6566
1530   N   4   0215   1549
1540   J   *   6560   0000
```



a.) CTC Indicators

The N character of the CTC instruction determines which of six possible indicators the instruction will sense. If the N = 0 no indicator will be sensed and the next instruction in sequence is staticized and executed.

If N = 1, the PRI's will be sensed. There are eight 301 instructions which set the PRI's as part of their normal operation. The student has already studied two of these instructions; LSL and LSR. The other instructions which set the PRI's are: Compare Left, Logical "AND," And, Subtract, Tape Read Forward Normal and Tape Read Reverse Normal. When the CTC instruction is sensing the PRI's, a PRP will cause transfer of control to the A address, a PRN will cause transfer to the B address, and a PRZ will allow the next instruction in sequence to be staticized and executed.

If N = 2, the Overflow Indicators will be sensed. The "Overflow Indicators" are actually two flip-flops called SCAR 1 and SCAR 2 (SCAR = sum carry). These flip-flops indicate that there is a carry present and will be mentioned at greater length in the arithmetic instruction lesson. The 10K and 20K 301's have only one SCAR slip-flop; only the 40K 301 has SCAR 2. If SCAR (SCAR 1 in the 40K machine) is set, the CTC will transfer control to the A address. If no overflow indicator is set, control is transferred to the B address. If SCAR 2 is set (40K only), the next instruction in sequence will be staticized and executed.

If N = 4, the Simultaneous Indicator is sensed. If a Simultaneous Read is being executed, control will be transferred to the A address. If write instruction in the Simultaneous Mode is being executed, control will be transferred to the B address. If the Simultaneous Mode is unoccupied, the next instruction in sequence will be staticized and executed.

NOTE: The Simultaneous Mode is used only for I/O Instructions.

If N = 8, control will be transferred to the A address if the ED/EF Normal Indicator is set, or to the B address if the ED/EF Normal Indicator is not set. ED and EF are 301 characters usually used to mark the end of a block, or group of blocks, of data. ED stands for "End Date", EF for "End File."

If N = &, the Interrupt Indicator will be sensed. This indicator is controlled by the Interrupt Button (INT) on the console. If this button is lit, control will be transferred to the A address. If INT is not lit, control will be transferred to the B address.

If N = -, the ED/EF Simultaneous Indicator will be sensed. If ED or EF had been read during the execution of a Simultaneous Instruction currently being executed, the CTC will transfer control to the A address. Otherwise, control will be transferred to the B address.

3. Machine Operation

The Conditional Transfer of Control instruction uses two status levels, an X1 and X2. These status levels are used to store the P address in standard location 0216-0219. The X1 stores the two most significant digits in

0216, 0217. The X2 stores the two least significant digits in 0218, 0219. In addition, at TP4 of X2, the P register is reset and at TP5 the appropriate address (A or B) is gated into the P register. P1 is selected at TP6 of X2.

The abbreviations used in X2 are:

SRB - Simultaneous Read Busy

SWB - Simultaneous Write Busy

NEDF - An ED or EF sensed during a normal read instruction.

SEDF - An ED or EF sensed during a simultaneous read instruction.

a.) Non-Transfer of Control

There are four instances where a P1 status level will be selected immediately upon completion of staticizing the CTC instruction. These four are:

(1) When N = 0

(2) When N = 1 and PRZ is set

(3) When N = 2 and SCAR 2 is set (40K only)

(4) When N = 4 and neither a simultaneous read nor a simultaneous write is being performed.

4. Programming Errors

In the CTC instruction, it is possible to use an N character which is not one of the characters specified. Each of the bits 2^0, 2^1, 2^2, 2^3, 2^4 and 2^5 represent the testing of a given indicator or indicators. If a combination of these bits is used for the N character, more than one indicator will be sensed.

Any N character of a CTC instruction, containing a 2^0 bit when PRZ is set, a 2^1 bit when SCAR 2 is set, or a 2^2 bit when the Simultaneous Mode is unoccupied (or not existing) will cause a P1 status level to be selected immediately after staticizing, thus not permitting a transfer of control.

Any other combinations will select an X1 status level after staticizing and then, depending upon the N character and the selected indicators, may

or may not attempt to gate both the A and B addresses onto the Bus at the same time during the X2 status level. If both addresses are gated onto the Bus, bad parity might result which would be gated into the P register. During the next P1 status level, bad parity would produce a MAPE alarm. However, good parity may result and the computer could jump to a completely different address without any alarm occurring.

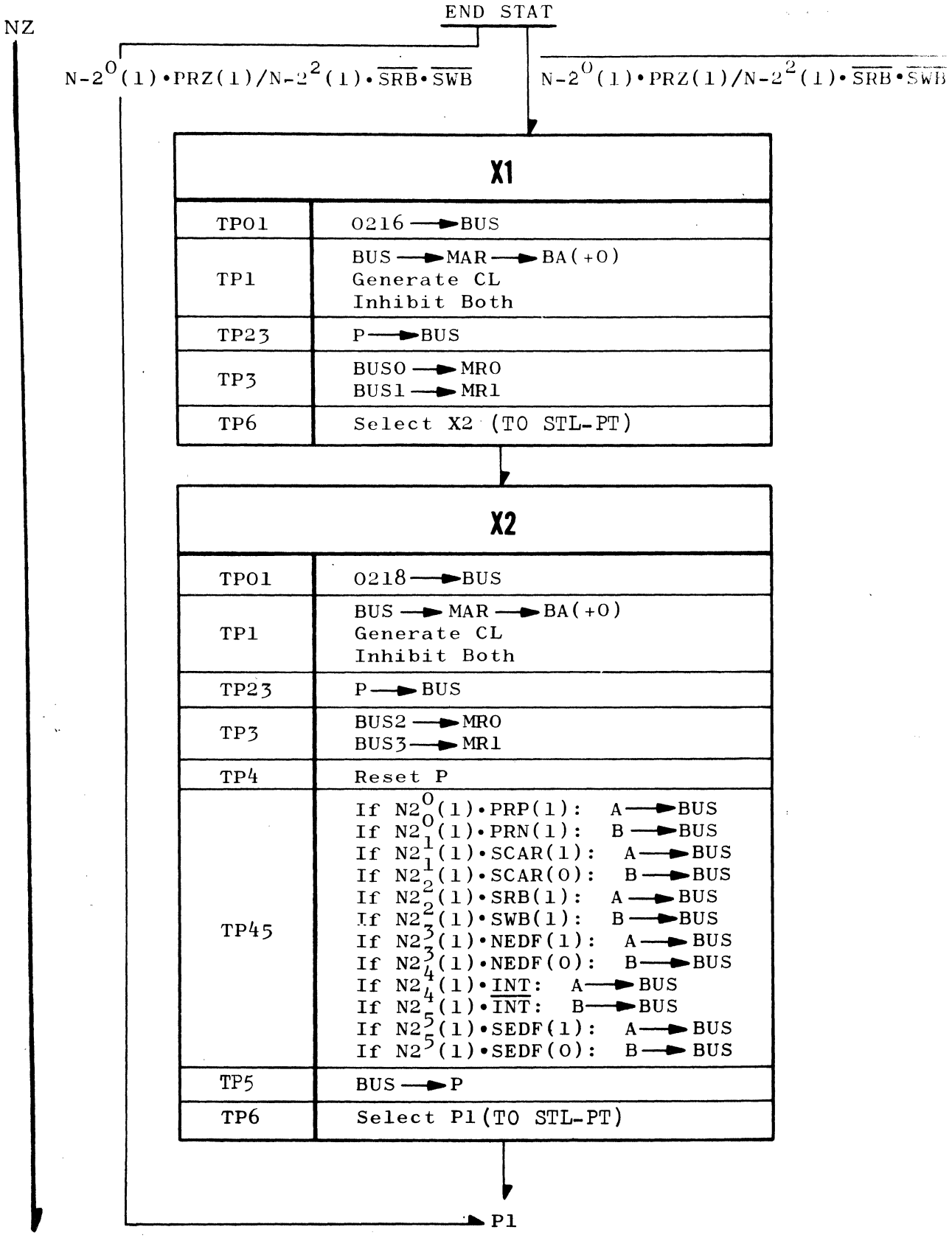


Figure 43 X1 and X2 of a CTC

Example 1:

Assume PRP is set and INT is not set on the console when the following instruction is staticized.

W A 4210 4280

Since an A contains a 2^0 bit and a 2^4 bit, both the PRI's and the Interrupt Button will be sensed. The X1 and X2 status levels are executed after staticizing and during X2, the PRI's and INT are examined. With PRP set and N-2^0 a one bit, the A address is gated onto the Bus. Also with N-2^4 set and INT not set, the B address is gated onto the bus. The Addresses 4210 and 4280 combine to create 4290 with bad parity in P2. Therefore, a MAPE alarm would occur during the next P1 status level.

Example 2:

Assume the simultaneous mode is unoccupied when the following instruction is attempted and SCAR (overflow) is set.

W 6 7350 7370

Since the N character is 6, the 2^2 and 2^1 flip-flops would be set in the N register. The 2^2 indicates testing the Simultaneous Mode and because the mode is unoccupied, a P1 status level is selected after staticizing and SCAR is not tested. No transfer of control takes place and the next instruction in sequence is executed. No alarm would occur.

Practice Problems:

5. A CTC instruction always uses X1 and X2 status levels - True or False? Explain your answer.

6. There is one ";" in HSM located somewhere between 2000 and 2999. Write a program to search 2000 → 2999 for the ";" and replace the ";" with a "." if INT on the console is set. If INT is not set, the program should replace the ";" with a "Ø". At the completion of this program, transfer control to location 4000.

7. Write a program to fill 2050 through 2100 with "Ø's" and check to make certain 2050 through 2100 all contain "Ø's"; if not, repeat operation until "0's" appear in these locations. When area is filled with "0's" transfer control 3000.

8. Why are A3 and B3 of a CTC instruction usually "Ø's"?

9. Write a three-instruction program starting at 1000 to search 5000 thru 5999 inclusive for any character other then "*". If the character at 5000 is something other than "*", transfer control to 1080. If there is a character other than "*" in 5001 thru 5999 inclusive, transfer control to 2020. If all Characters are "*'s" transfer to 3000.

10. An ED has just been sensed in the Simultaneous Mode. No I/O instructions have been performed in the normal mode. Describe the execution of: W Q 1230 5300.

D. Y - COMPARE LEFT (COM)

In some cases during the execution of a program, the choice "what course of action next" must be based on the contents of a particular location or locations in HSM. There is need for an instruction which will examine the contents of a certain portion of memory and make a decision based on those contents. "Compare Left" is such an instruction.

This instruction compares two given quantities or items and determines which is the larger. The PRI's are set to indicate the result. Comparison begins with the most significant digits and proceeds from left to right. The instruction terminates upon finding the first non-comparison or upon decreasing the N Count to zero, if all characters are equal. PRZ is set initially and, if still set after execution of the Compare, will indicate that the operands are equal. PRP being set upon termination of the instruction signifies that a positive result was obtained (first operand larger than second operand, i.e., the operand addressed by A is larger than the operand addressed by B). PRN being set indicates a negative result (first operand smaller than second operand). This instruction does not go through STA or STP.

The COM instruction does not transfer control, but it does set the PRI to indicate the results of its comparison. Thus it may be used in conjunction with the CTC instruction (which can sense the PRI) to cause transfer of control. This transfer will then depend upon the contents of that part of memory examined by the COM instruction.

NOTE: Pages VII-7 and VII-8 of the Programmers' Reference Manual give an operation summary of the COM instruction.

1. Instruction Format

Op. Code	N	A	B
Y	Numbers of Characters to be Compared (0-44)	HSM Address of Leftmost Character of First Operand	HSM Address of Leftmost Character of Second Operand

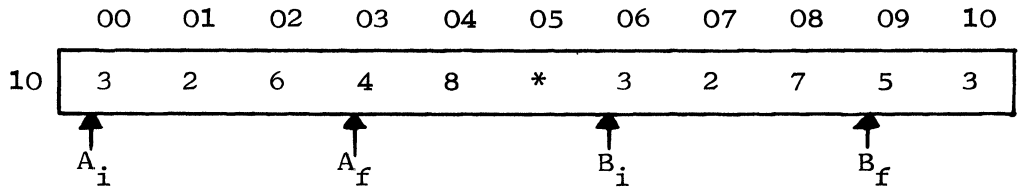
A_f = One location to the right of the last character compared in the first operand.

B_f = One location to the right of the last character compared in the second operand.

NOTE: If N = 0, no characters are compared and the next instruction in sequence is executed.

2. Instruction Execution

Example: Y 5 1000 1006



PRN would be set upon completion of the above instruction.

Practice Problems

1. Write an instruction to compare two operands consisting of six characters each. The LSD of one is at address 3150 and the LSD of the other is at address 7582.

2. If the following instruction were executed, how many characters would the computer actually compare?

Y B 8000 8105

	00	01	02	03	04	05	06	07
80	4	8	2	5	6	4	9	7

	05	06	07	08	09	10	11	12
81	4	8	2	5	3	4	9	7

What PRI would be set upon termination?

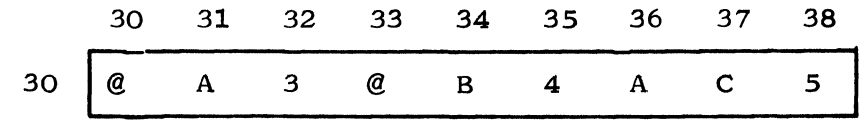
_____ PRI? _____

3. Write two instructions which will compare unknown items A and B, and determine where to transfer control. If A is greater than B, transfer to address 5500. If A is less than B, transfer to address 5550.

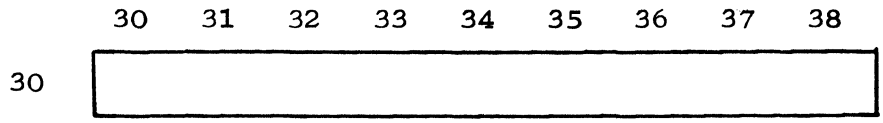
	30	31	32	33	34	35	36
76	A	A	A	*	B	B	B

4. Execute the following program showing final HSM contents.

6000	N	3	3032	3038
6010	#	@	3033	3030
6020	M	4	0212	6032
6030	Y	2	0000	3037
6040	W	1	6060	6070
6050	V	1	0219	6070
6060	J	0	3037	3038



HSM BEFORE



HSM AFTER

5. Two unknown characters, X and Y, exist in memory at locations 1000 and 1001. Write a program to compare these characters and determine which one is larger. Move the larger character to HSM location 1005. If both characters are equal, place an "E" in location 1005. Use 2000 as the address of the first instruction in the program. Terminate by transferring control to 3000.

6. What are the two situations which cause a Compare Left Instruction to terminate?

7. How many status levels are used to staticize and execute the COM Instruction of Problem #2?

3. Machine Operation

There are three basic status levels involved with the COM Instruction. An A1 status level brings out the character addressed by A and places it in D2. N is triggered down by one and the A address is incremented by one. A "B" status level brings out the character from the second operand as addressed by B and places it in D3. The comparison takes place while B is incremented by one and sent back to B. The third status level is an X1 which checks the result of the comparison and sets the proper PRI. (Notice that the X1 status level does not generate a command level). If the N Count is down to zero or the comparison was unequal, the instruction terminates by selecting a P1 status level. However, if N is not down to zero and the comparison just completed was equal, another A1 status level is selected. The sequence of A1, B, X1 continues until a non-compare occurs or until N is reduced to zero. It should be noted that PRZ is set during staticizing (P5) and will remain set until an X1 resets it while setting PRP or PRN. Therefore, if all comparisons prove equal, PRZ will remain set throughout the entire instruction, while N is reduced to zero.

NOTE: In Figure 44, " D2 > D3" means "D2 is greater than D3", and "D2 < D3" means "D2 is less than D3".

4. Programming Errors

About the only misuse of the Compare Instruction that can occur, is using an N character which is not part of the N count. (This was described in the DL and DR section of the previous lesson.) However, one must remember that the initial A and B addresses must specify the location of the MSD's of the operands to be compared and not the LSD locations.

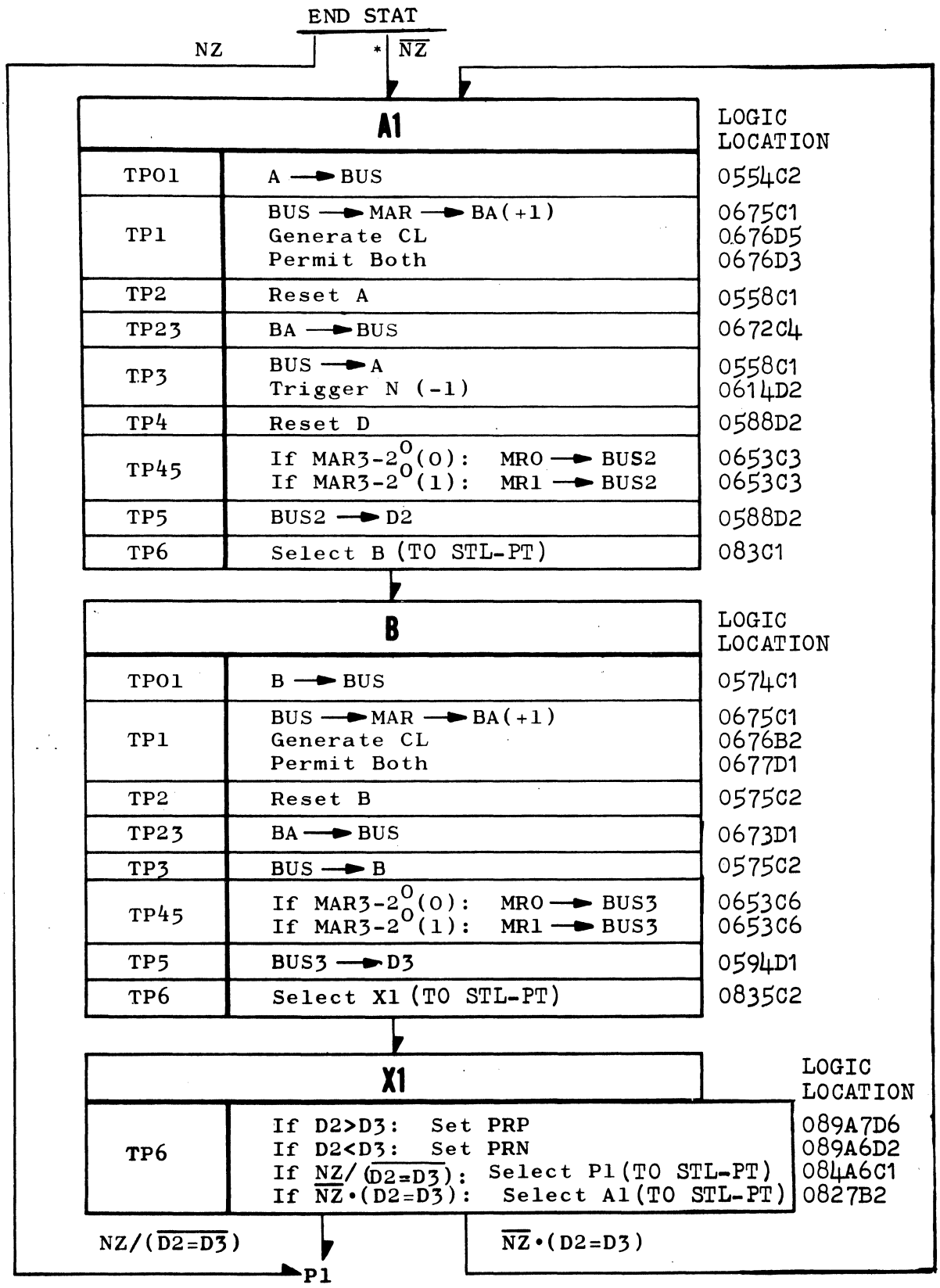


Figure 44 A1, B, and X1 of COM

Practice Problems

8. Write a program to compare the character in 1086 with the character in 5023. If the character in 1086 is the larger, transfer control to 2000. If the character in 5023 is the larger, transfer control to 3000. If rhw characters are equal, move the contents of 0532-0535 to 1066-1069.

9. There is a character, either Group 1 or Group 11, in location 5239. If it is Group 1, write a "1" in location 5501. If it is Group 11 write a "2" in location 5611. Terminate by transferring control to 3000. (Start program at 1000). Use location 8000 for storage of the character to be compared against the character in location 5239.

10. Locations 9000-9099 inclusive contain a mixture of numeric and alphabetic information. Write a program to replace each letter of the alphabet between 9000-9099 with a "0". The numeric data should not be changed. Program starts at 1000. After all alphabetic characters have been replaced with "0", transfer control 2000.

B. X - TALLY (TA)

The Tally Instruction enables a certain portion of a program to be repeated a specified number of times by transferring control to a specified location a given number of times. The quantity which determines how many times this transfer of control is to be made is called the "Tally Quantity". This quantity can be as large as 99. Each time the transfer of control is made, the Tally quantity is decremented by one. Exhausting the Tally quantity terminates the effectiveness of the instruction and the instruction following the Tally is staticized and executed. The Tally instruction goes through STP when transfer of control is effected.

> NOTE: Operation of the Tally Instruction is summarized on Page VII-9 of the Programmers' Reference Manual.

1. Instruction Format

Op. Code	N	A	B
X	Ignored	HSM address of diad containing Tally quantity.	HSM Address of next instruction to be executed if Tally Quantity is not exhausted.

The A address given the location of the Tally quantity in memory. Remember that this consists of an even and odd location, the even location holding the MSD of the diad. The B address is the address to which control is transferred, provided the contents of the diad addressed by A are not equal to 00. The N character is ignored.

2. Instruction Execution

Example 1:

X 0 1005 3000

	02	03	04	05	06
10	3	2	0	2	8

The above instruction will transfer control to address 3000 two times.

NOTE: If Tally Quantity is initially 00, no transfer of control takes control.

Example 2:

1000	J	*	1550	1600
1010	N	K	2300	1600
1020	X	0	1034	1000
1030	.	0	0002	0000

The J and N instructions would be executed a total of three times in the above program. Each instruction would be executed once initially and then repeated twice, since the Tally quantity is 02.

If it is desired to perform a segment of a program the exact number of times of the Tally quantity, two more instructions are needed. The following is an example of this.

1000	X	0	1054	1020
1010	V	1	0219	1050
1020	J	*	1550	1600
1030	N	K	2300	1600
1040	V	1	0219	1000
1050	.	0	0002	0000

The third time the Tally instruction is staticized in this example, the tally quantity will have been reduced to zero and the Store instruction (1010) will be staticized. When the Store instruction is executed, control will be transferred to 1050, which is a Halt instruction. Thus, the J and N instructions will be executed a total of two times.

Practice Problems

1. How many times will the computer transfer control to address 2000, if the following instruction is executed?

X 9 1608 2000

	05	06	07	08	09
16	8	5	6	3	4

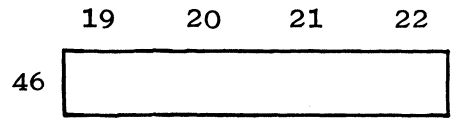
2. Assume the following instruction is located in memory, beginning at address 4240. What will locations 0216 through 0219 contain after this instruction is executed?

X 5 3622 4250

	20	21	22	23
36	0	1	0	2

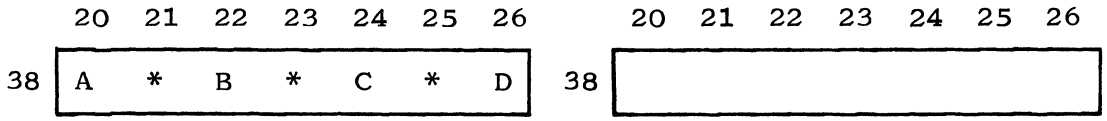
	16	17	18	19
02				

3. Write an instruction which will loop back to address 2300 nine times. Designate tally quantity in memory at 4620.



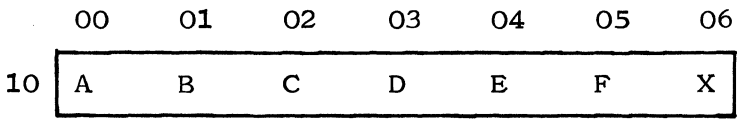
4. Execute the following program and show final HSM contents.

3300	M	1	3820	3821
3310	V	4	3309	0002
3320	X	0	3318	3300
3330	.	0	0000	0000



HSM Before HSM After

5. Explain what the following program is doing (Note: X = unknown character).



2000	N	4	2019	2045
2010	Y	1	1006	1000
2020	V	4	2019	0000
2030	W	1	2060	2060
2040	M	1	0000	1010
2050	V	1	0219	2070
2060	X	-	2072	2000
2070	.	0	0500	0000

3. Machine Operation

The Tally instruction uses five status levels during execution: A1, X3, A2, X1, and X2. The A1 status level is used to gate the Tally Quantity (i.e., the contents of the diad addressed by the A register) into the D register. Notice that BA (+0) is used since we may want to address this same diad again and again.

The X3 status level is used to decrement the Tally Quantity by one. The bus adder circuitry is used to perform this function. At TP01 of the X3, the standard address generator puts 0's on BUS 0 and BUS 1 and D2 and D3 are gated onto BUS 2 and BUS 3, respectively. This pseudo-address is then gated into the MAR where the bus adder will decrement it by one. Since the X3 status level does not generate a memory cycle, the "address" in the MAR will not be used to address memory. At TP5 of X3, the decremented Tally Quantity is gated into the D register. At TP6, the D register is checked to see if it contains a count of 99. A count of 99 at this time would indicate that there had been a count of 00 present at the beginning of the X3 status level. If D = 99, a P1 is selected and the instruction following the tally will be executed. $\overline{\text{If D = 99}}$, an A2 status level is selected in order to transfer control.

The A2 status level is used to write the decremented Tally Quantity into the diad in HSM addressed by the A register. The X1 and X2 status levels are used to store the P register contents in 0216-0219 and to replace the contents of P with the contents of B. Then a P1 status level is selected.

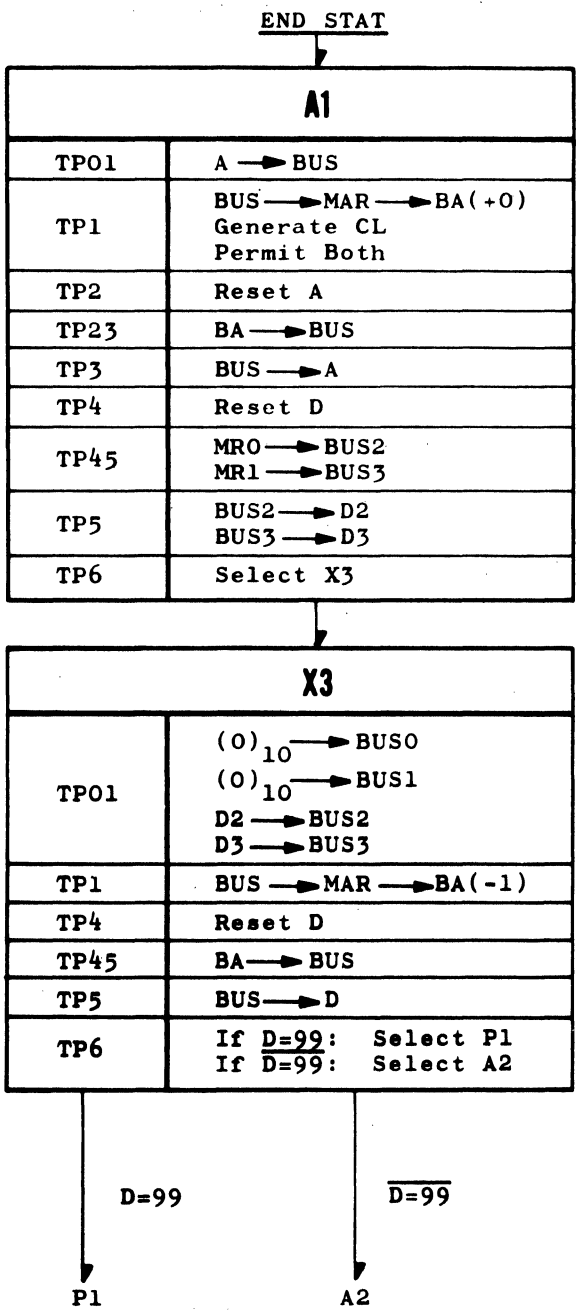


Figure 45 A1 and X3 of a Tally

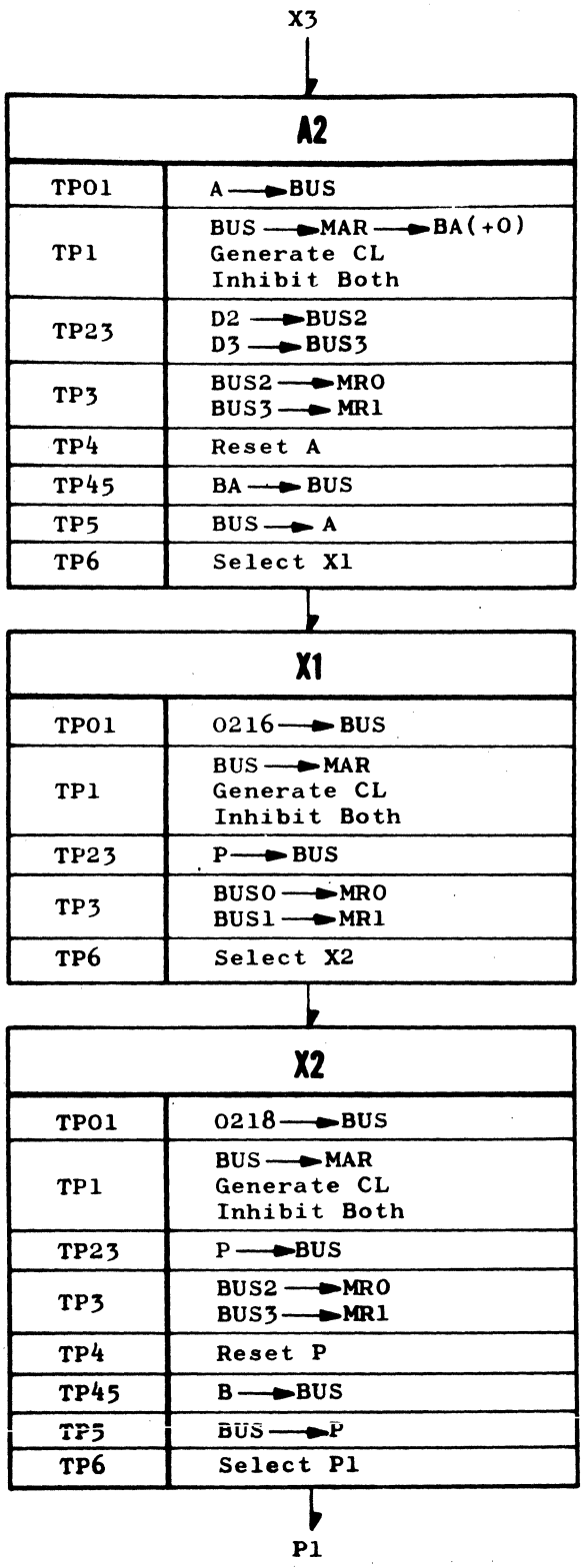


Figure 46 A2, X1, and X2 of a Tally

4. Programming Errors

In the Tally instruction, some unusual things can occur if an incorrect Tally Quantity is used. A Tally Quantity between 00 and 99 causes no problems but should certain zone bits occur in the tally diad, the computer will malfunction due to the Bus Adder.

Any 2^5 bit in either character of the Tally Quantity will cause a DPE (D register parity error).

The reason is that the Bus Adder does not provide for a 2^5 bit in either character when subtracting one from the Tally Quantity (X3 status level). If this bit exists, bad parity will be generated and a D register parity error will result.

A 2^4 bit in the C2 character of a Tally Quantity will also generate a DPE, since the Bus Adder does not provide for a 2^4 bit in C2.

However, because of the provision for indirect addressing, a 2^4 bit in C3 will be handled by the Bus Adder. This will cause erratic tallying and intermittent DPE's.

Care must be taken when addressing the Tally Quantity. Since the Tally Quantity is always a diad, one might easily obtain a larger number than desired for the Tally Quantity.

Example:

	X	0	1006		5000	
	04	05	06	07	08	09
10	5	0	2	0	0	2

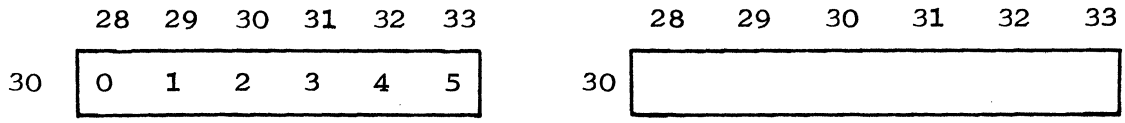
The above instruction would transfer to address 5000 twenty times and not two times. If it were desired to tally twice, the tally quantity should be arranged as shown in diad 1008 and 1009.

Practice Problems

6. Write a program to move 600 characters from location 2000-2599 to location 3000-3599. Use the Tally Instruction.

7. Describe the execution of the following program.

```
1000    J    1    2000    4000
1010    X    0    3031    1000
```



HSM Before HSM After

8. Describe the execution of the following program.

```
1000    J    0    2000    3000
1010    J    1    3001    3099
1020    X    0    1002    1020
1030    X    0    3000    1000
```

9. What is the difference between a Tally instruction and a "Store P"?

10. Write a program that will translate 6000 characters from 3000-8999. Program to start at location 1000 and table to start at location 2000

F. HALT (HLT)

In the programs you have been asked to write so far, and in the sample programs given, the problem of termination has been largely avoided. You have been told to "transfer to 3000 to terminate" or the problem has been ignored altogether. A brief example will illustrate the need for a means of terminating.

```
1000    J    @    5000    5099
1010    M    &    5000    1020
```

During execution of the preceeding program, the SF instruction will fill the indicated locations with "@" and the DL instruction will move ten "@" to 1020-1029. The program will now read in memory:

```
1000    J    @    5000    5099
1010    M    &    5000    1020
1020    @    @    @@@@    @@@@
```

At TP6 of the P5 used to staticize the last two characters of the "instruction" in location 1020, no status level will be selected since it is not a legal operation code. Thus, the computer will stop on an STLE with an empty Status Level Register. If location 1020 had contained a legal operation code, a status level would have been selected and execution attempted leading to any of a number of errors.

The Halt instruction is usually used to solve the problem of termination. You have already seen this instruction used several times in the section on the Tally Instruction. The Halt instruction is unique among RCA 301 instructions in that it has no first processing level.

NOTE: Page VII-10 of the Programmers' Reference Manual summarizes operation of the Halt Instruction.

1. Instruction Format

Op. Code	N	A	B
(period)	Ignored	Ignored	Ignored

Because everything except the operation code is ignored in the Halt instruction, there are nine convenient locations which can be used for storing constants. This procedure has been illustrated in previous problems of this lesson.

2. Instruction Operation

One useful technique is the use of numbers in the N character to indicate which halt terminated processing. For instance, a ".1" (Halt 1) might indicate normal termination; a ".2" (Halt 2) might be chosen in the event of a machine or programming error.

Another useful feature of the Halt instruction is its ability to stop processing while needed information is entered. (This feature is used extensively by the Test and Maintenance routines). For instance, if Halt 4 is followed by a Store A and Store B, information inserted manually into the A and B registers while the computer is stopped at ".4" will be stored in memory at the indicated locations.

Example:

1000	.	4	0000	0000
1010	V	2	0000	0000
1020	V	4	1008	0000
1030	.	5	0000	0000

If the P register is manually set to 1000 and "Start" is depressed, the ".4" in location 1000 will be staticized and then the computer will stop. Now the operator may insert characters into the A and B registers manually, say "1234" and "5678" into the B register. "Start" is depressed a second time. The "1234" will be written into memory at location 0212-0215. The "5678" will be written into memory at 1006-1009. The computer will then stop on ".5".

NOTE: This method is somewhat limited by the fact that not all characters can be inserted into certain digit positions of the A and B register.

Practice Problems

1. Write a program to search 2000-3000 for 0 starting at 2000. If the first character searched is not zero, transfer control to "Halt 1". If the first character is zero but some character following the first is not zero, transfer control to "Halt 2". If every character in 2000-3000 inclusive is zero, transfer control to "Halt 3".

2. Write a program which can be used to store manually inserted numericaly information in location 1054 - 1056 and 2132 - 2135.

3. Which Status Level is the First Processing Level used by the Halt Instruction?

3. Machine Operation

Now we must consider how the Halt instruction stops the computer. Remember that the computer is started by pushing the START button. This action starts the Timing Pulse Generator. It would seem logical that to stop the computer, one must stop the TP Generator. This is exactly what the Halt Instruction does.

If the Simultaneous Mode and Record File Mode are unoccupied when the Halt instruction is staticized, completion of staticizing produces TPO·ST(P). These signals stop the TP generator before P1 (which is the next status level selected) can be executed. If either the Simultaneous Mode or the Record File Mode is busy, the instruction in that mode is completed before the processor stops. Remember, all 10 characters of the Halt instructions will be staticized before the computer will stop.

4. Programming Errors

The Halt instruction appears to be very simple and, therefore, free of programming problems since everything except the operation code is ignored. However, if the A and B addresses are used for constants, characters containing 2^4 bits may occupy the A3 position or the B3 position and produce an indirect address. If the A address is indirect, nothing will occur regardless of the B address. The computer will halt selecting a P1 status level after a P5. (NOTE: An M1 and P1 are selected but "ORing" the two together produces a P1.) Should an indirect address exist in B and not in A, the computer will attempt to select both an M3 and a P1 status level and produce an STLE (37_8 with bad parity).

Example 1:

```
.    0    _301    TAPE
```

Since the B3 character is an E, the computer would attempt indirect addressing and would stop on an STLE alarm.

Example 2:

```
     · FIRST TAPE
```

In the above example, both A and B appear as indirect addresses. However, A is always handled first during indirect addressing, therefore, no alarms occur and the computer stops having selected a P1.

Practice Problems

4. Rewrite the program required in Practice Problem 5 of this lesson using the Halt instruction. Do not use the REG instruction. (It is not necessary to transfer control to 3000 to terminate).

5. P is set to 1000. What happens when Start is depressed? What happens when Start is depressed a second time?

```
1000   .   0   A   BAD   TIME
1010   .   1   F   OR2   MORE
```

G. R - REPEAT (RPT)

The Repeat instruction is used in conjunction with "Repeatable" instructions. There are 12 Repeatable instructions in all; you have been introduced to 5: TRA, DL, DR, DSL, and DSR. (The remaining seven are: EXO, AND, OR, ADD, and SUB which are all arithmetic instructions, and RFN and RRN which are input from tape instructions.) The Repeat instruction causes the next Repeatable instruction in sequence to be executed the number of times specified by the N character of the Repeat. All non-repeatable instructions which occur between the Repeat and the next Repeatable instruction in sequence will also be repeated. All these instructions which are repeated are referred to as the "field" of the specific Repeat instruction. The A and B addresses of the Repeat instruction are not used to address memory, but they have a very important function. This function makes the Repeat instruction quite different from the Tally instruction, which it so far seems to resemble closely. If the A address is even (i.e., no 2^0 bit in A3), no A address of any succeeding instruction is staticized except the A address of the first instruction following the Repeat instruction. The A address of the first instruction following the Repeat will be staticized the first time it is to be executed only. After this the A address will be used (in its incremented or decremented form) as the A address of every instruction in the field of the Repeat instruction until the repeat quantity is exhausted.

However, if the A address is odd (i.e., A3 has a 2^0 bit), each A address will be staticized in the usual way. This even/odd convention holds for the B address also. That is, if B of the Repeat is even, only the B address of the instruction immediately following the Repeat will be staticized, and it will be staticized the first time it is to be executed only. From then on, the B of one execution will be the B of the next execution. If B of the Repeat is odd, staticizing of the B address will proceed in a normal fashion.

The Repeat instruction holds its N count in the NR Register. It is the only instruction to use this register. The Repeat instruction uses a special N count with a maximum count of 14. The Group 1 characters are the only legal N characters for Repeat.

NOTE: Page F-1 of the Programmers' Workbook gives a table of legal N characters for Repeat.

The Repeat instruction uses a standard location for storing the contents of the P register: 0222-0225. Notice that this is a different location from the STP location (0216-0219). The P address of the instruction immediately following the Repeat must be stored in a standard location because this is the location to which control must be transferred after execution of the Repeatable instruction as long as NR $\neq$ 0. This location must be different from STP, because an instruction which uses STP may be in the field of the Repeat. From the above, it becomes evident that a Repeatable instruction is merely one which checks the contents of NR at the end of its execution and if NR $\neq$ 0 selects a status level which will start the transfer of control to the address contained in 0222-0225.

NOTE: Page VII-11 of the Programmers' Reference Manual summarizes the operation of the Repeat instruction.

1. Instruction Format

Op. Code	N*	A	B
R	Number of Times to Repeat the Repeatable Instruction (0-14)	Even-Do Not Staticize A Address of Instruction. Odd-Always Staticize.	Even-Do Not Staticize B Address of Instruction. Odd-Always Staticize.

Those instructions which are repeatable are:

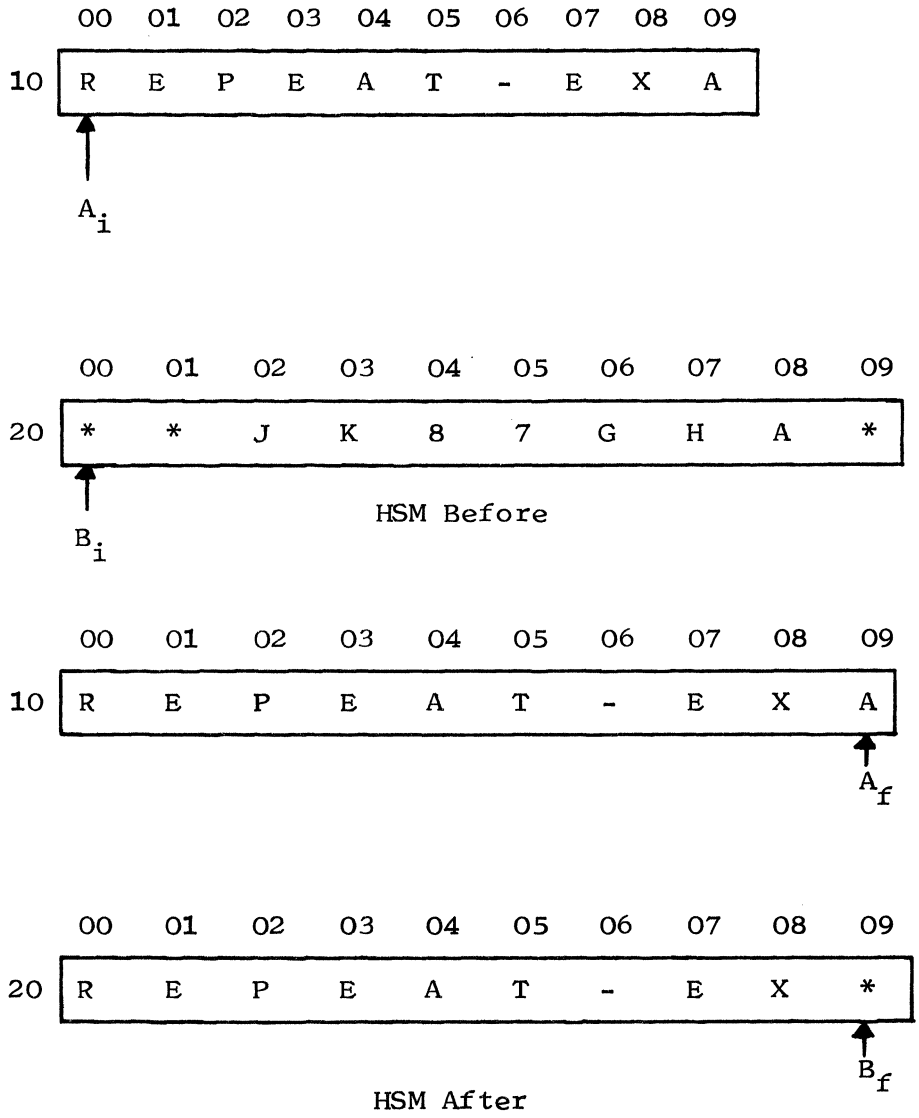
1. Data Left
2. Data Right
3. Transfer Data By Symbol Left
4. Transfer Data By Symbol Right
5. Logical AND/OR/EXO
6. Add
7. Subtract
8. Tape Read Forward Normal
9. Tape Read Reverse Normal
10. Translate

* NOTE: If N = 0, no instruction will be repeated.

2. Instruction Operation

Example:	R	2	0000	0000
	M	3	1000	2000

This combination will transfer a total of nine consecutive characters starting at address 1000 to the address beginning at 2000. Note that the total number of times the repeatable instruction is executed is one more than the N Character of the Repeat instruction. This is because the repeatable instruction is executed once initially before the N Count is reduced.

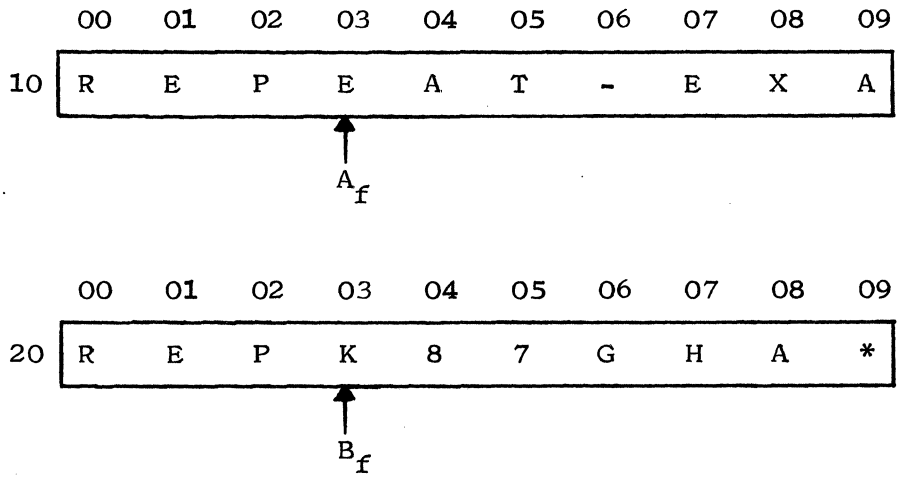


The characters "REP" would be transferred during the first execution of the M. A final would be 1003. This would become A initial for the second execution of the M instruction. B final would be 2003 which would also be B initial for the second execution. The M instruction would be repeated a total of two times but executed a total of three times. The sequence of its initial addresses after staticizing each time would be:

M	3	1000	2000	NR = 2
M	3	1003	2003	NR = 1
M	3	1006	2006	NR = 0

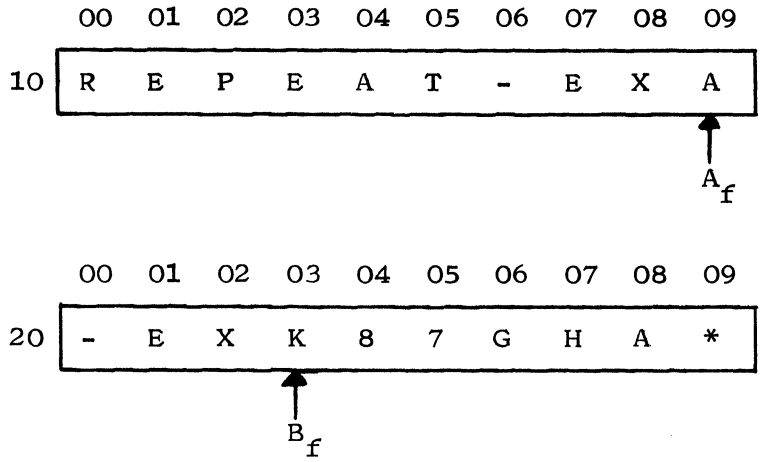
NOTE: - NR = N Count for Repeat Instruction.

Had the A and B addresses of the Repeat instruction been odd instead of even, final HSM contents would be as follows:

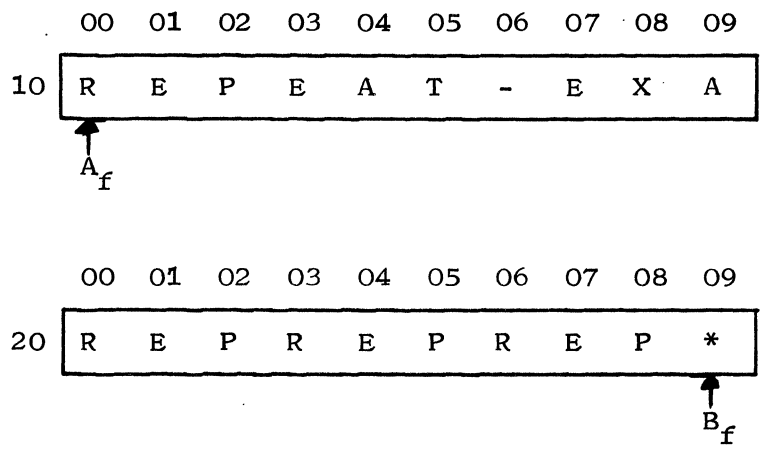


In this case, the same three characters were transferred to the same three addresses three times.

Had the A address been even and the B odd in the Repeat instruction, final HSM contents would appear as follows:



Here all nine characters were placed in the same three locations 2000, 2001 and 2002, since A kept counting up but B was restaticized each time. The final case of A initially odd and B even in the Repeat instruction gives a HSM result as follows:



Practice Problems

1. Execute the following instructions and show final HSM contents.

R	1	5660	4753
P	#	3006	4009

	00	01	02	03	04	05	06	07	08
30	6	4	#	9	3	#	P	G	H

	01	02	03	04	05	06	07	08	09
40	2	3	8	G	#	X	B	A	#

HSM Before

	00	01	02	03	04	05	06	07	08
30									

	0	02	03	04	05	06	07	08	09
40									

HSM After

2. Write two instructions which will transfer 132 characters from consecutive locations starting at 1000 to consecutive locations starting at 2000 (left to right).

3. What would A final be after the # instruction has been executed for the last time?

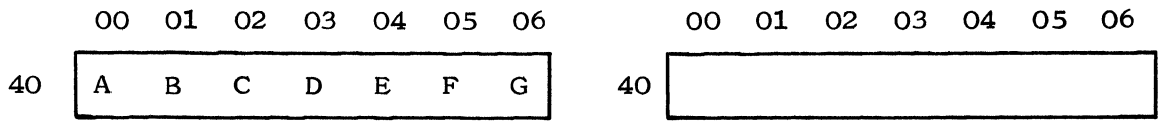
R	3	0000	0000
#	*	5530	5530

	30	31	32	33	34	35	36	37	38
55	A	*	B	*	C	*	D	*	E

A_f = _____

4. Execute the following program and show final HSM contents.

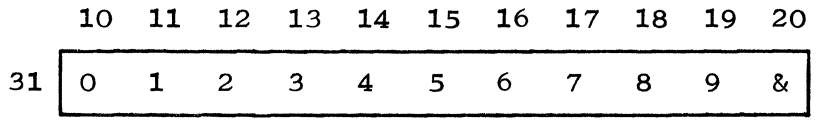
```
3000   R   2   1005   2030
3010   M   1   4000   4001
3020   V   4   3015   0000
3030   X   0   3047   3000
3040   .   0   0000   0102
```



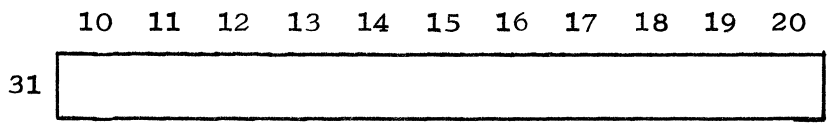
HSM Before HSM After

5. Describe the execution of the following program and show final HSM contents.

```
1000   R   1   3113   3115
1010   J   *   3112   3114
1020   M   2   3110   3116
1030   .   0   0000   0000
```



HSM Before



HSM After

3. Machine Operation

The Repeat instruction utilizes four status levels: X1, X2, REP1, and REP2. The X1 and X2 status levels are used primarily to store the P address of the instruction immediately following the Repeat in sequence. The X1 status level has several other important functions. The standard address generator is used to produce the conventional addresses 0222 for X1 and 0224 for X2. At TP3 of X1, the 2^0, 2^1, 2^2, 2^3, and 2^6 bits of the N character are gated into the NR register (NR uses only Group 1 characters). At TP4 of the X1, the A and B addresses of the RPT are checked. If A is even, the INHA flip-flop (Inhibit A) is set to prevent staticizing of the A address. If B is even, the INHB flip-flop (Inhibit B) is set to prevent staticizing of the B address. FREP (First Repeat) is also set at TP4 of X1.

Now refer to pages 2-8, 2-9, and 2-10 of the Status Flow Manual (Staticizing). Notice that at TP4 and TP5 of P2 and P3, normal staticizing will occur if INHA is not set or FREP is set. Also, at TP4 and TP5 of P4 and P5, normal staticizing will occur if INHB is not set or FREP is set. FREP will be reset at TP6 of the next P5 executed, or after staticizing the instruction immediately following the Repeat in sequence. This means that the instruction immediately following the Repeat when the A and/or B address of the Repeat are/is even will be staticized normally the first time. After that INHA and/or INHB will prevent staticizing of the associated register(s) INHA and INHB will be reset at TP6 of the first P5 after the NR count is exhausted. This can lead to trouble if a group of instructions are being repeated with an odd A or B address in the Repeat.

Every instruction after the one immediately following the Repeat instruction will be staticized when the NR count = ZERO. The problem will be mentioned again under "Programming Errors".

The REP 1 Status Level is selected after execution of the first repeatable instruction if NR $\neq$ 0. REP1 and REP2 are used: (1) to read the stored P address of the instruction immediately following the Repeat out of memory and (2) to insert it into the P register. This transfers control for another execution of the instruction(s) to be repeated. At TP2 of REP1, a parity

adjustment for the NR character "to be" is made. At TP3 of REP1 the NR count is triggered down one and assumes its "good parity" condition. At TP6 of REP1, NRPE (NR register parity error) is set and stops the computer if the NR register contains bad parity. At TP6 of REP2 (which was automatically selected by REP1) P1 is selected to start staticizing the instruction immediately following the Repeat in sequence.

4. Programming Problems

One of the major sources of trouble with the Repeat instruction is following the RPT with a non-repeatable instruction when the A or B address of the Repeat is even.

The R instruction itself sets up initial conditions for repeating and is normally executed just once. One of the functions of the Repeat is to store the contents of the P register (address of next instruction in sequence) in standard locations 0222-0225. If the next instruction is a Repeatable instruction, it selects a REP1 and REP2 status level upon completion. These two status levels count the NR register down one and bring out the address stored in 0222-0225 to be placed in the P register.

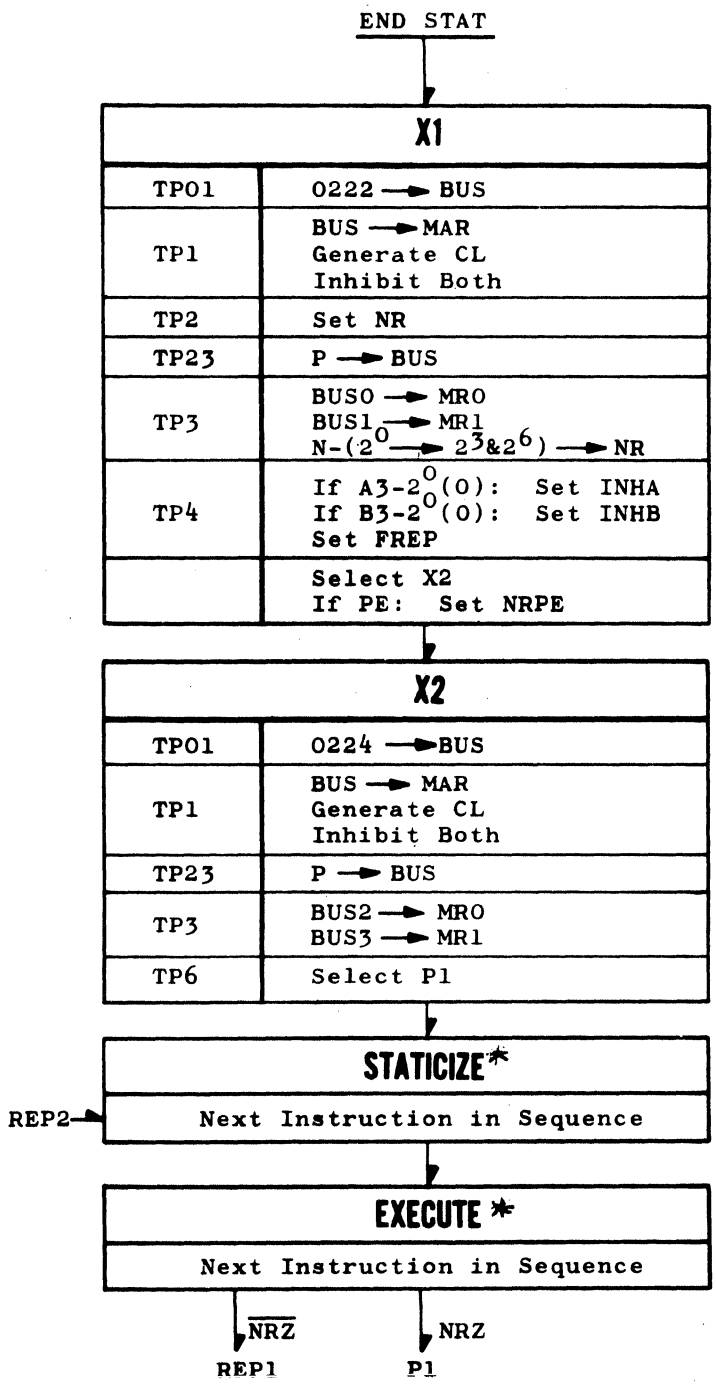


Figure 47 X1 and X2 of a RPT

*Note: It is assumed here that the "Next Instruction in Sequence" is a repeatable instruction.

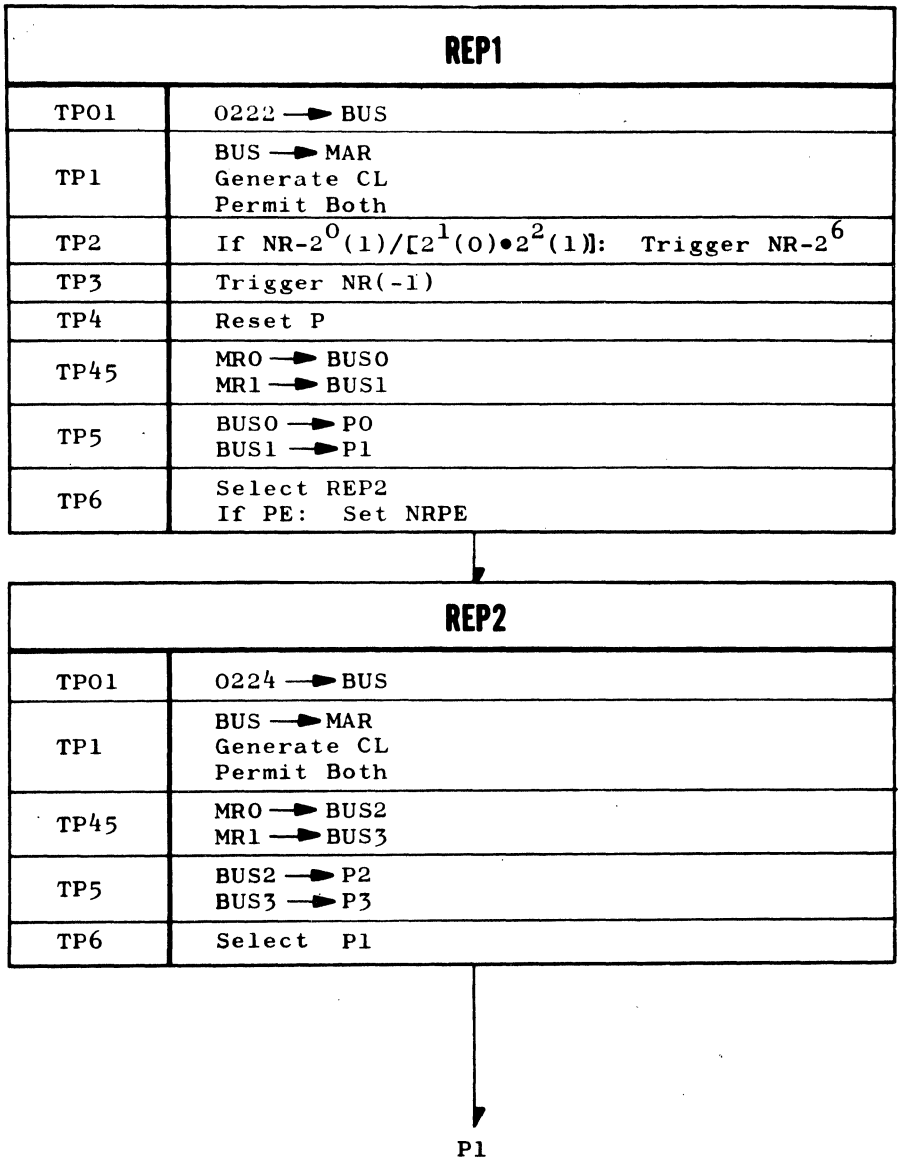


Figure 48 REP1 and REP2

Therefore, the next instruction to be executed should be the same instruction just completed which is the one immediately following the R instruction. The process of repeating continues until NR is decreased to zero.

If the instruction immediately following the R instruction is non-repeatable, however, the computer will continue to execute instructions until a repeatable instruction is found. Then when REP1 and REP2 bring out the pre-stored address from 0222-0225 the computer goes back to the instruction immediately following the R instruction. Therefore, a group of instructions can be repeated.

Example:

```
5000    R    2    0001    0001
5010    J    0    3000    3050
5020    M    1    3000    3050
```

Symbol to Fill (J) is a non-repeatable instruction but DL (M) is Repeatable. Both instructions will be executed a total of three times using the same addresses each time. (The A and B addresses of the RPT are odd.) If the A and/or B address(es) of the Repeat instruction were even, each instruction repeated would use the final address(es) of the preceeding instruction.

Example:

```
5000    R    2    0000    0001
5010    J    0    3000    3050
5020    M    1    3000    3050
```

When the J instruction is executed initially, the addresses used are those read out of memory during staticizing. A_f will be 3051. Since the A address of the RPT is even, the A address of the DL instruction is not staticized (since INHA is set) and A_i will be 3051. A_f of the DL is 3052. Upon termination of the DL, REP1 and REP2 would be selected and control would be transferred to 5010 to do the SF a second time. Again, since INHA is set, the A address of the J instruction is not staticized and 3052 is used as A of the SF. Memory would then be filled to the top, A-B equality

could not be reached and a WTT would occur.

Example 2:

```
5000    R    2    0001    0000
5010    J    0    3000    3000
5020    M    1    2000    0000
```

Again, the A and B addresses of the SF instruction are both staticized the first time through, since FREP is set. However, FREP will be reset at TP6 of the P5 used to staticize SF. A_f of SF = 3001, B_f of SF = 3000. Since the B address of the Repeat instruction is even: A_i of the DL = 2000 and B_i of the DL = 3000. Control will now be transferred (by REP1 and REP2) to 5010 where A_i = 3000 (A will be staticized), B_i = 3000 (B will not be staticized). The NR count was decremented by 1 by REP1. The same process is repeated again. However, the third time the SF instruction is staticized, INHB will be reset at TP6 of P5 since NR = 0 at this time. Thus, the third time the DL instruction is staticized, the B address will be staticized also. The computer will be stopped by a WTT or the DL tries to move a character to 0000.

The conclusions are: (1) The Repeat instruction functions as it should if a repeatable instruction is coded immediately following the R, (2) The Repeat instruction can be used to repeat a group of instructions if the A and B addresses of the R instruction are odd, (3) Sharing of final addresses occurs if the A or B of the Repeat is even and at least one non-repeatable instruction exists between the Repeat and the Repeatable instruction. Situation (3) can rarely be used to advantage and should be avoided if at all possible since it can cause errors.

Another common source of error is use of an improper N character. It is possible to cause a NRPE if an incorrect N character is used in the Repeat instruction. The N character should be one of those characters existing in group 1 of the 301 code (00 for zone bits 2^5 and 2^4). If one of the characters in groups II or III is used as the N character of an R instruction, the zone bit (2^5 or 2^4) will be lost upon transfer into the NR register and

an NRPE will occur (X1 status level). The loss of the zone bits occurs because there is no 2^5 or 2^4 flip-flop in the NR register. If, however, a character from code group IV (11 for zone bits 2^5 and 2^4) is used as the N character, both zone bits are dropped and good parity is maintained. Nevertheless the computer can only repeat the decimal equivalent of the information bits 2^0, 2^1, 2^2, and 2^3, thus no advantage is gained by using group IV characters.

Another possible source of error is misuse of indirect addressing. If indirect addresses are used in the Repeat instruction, the addresses which finally determine whether or not INHA and/or INHB will be set are the final addresses after all indirect addresses have been replaced. Indirect addresses in the field of the Repeat instruction can be responsible for mistakes if either the A or B address of the RPT is even.

Practice Problems

6. List the Status Flow for the following program.

1000	R	1	0000	0001
1010	N	2	3000	4000
1020	.	0	0000	0000

(i.e., P1, P2, P3, P4, X1, X2)

What are A_f and B_f before the halt?

7. List the Status Flow for the following programs

(a)

1000	R	1	0001	0000
1010	J	*	2999	3000
1020	N	1	3000	4000

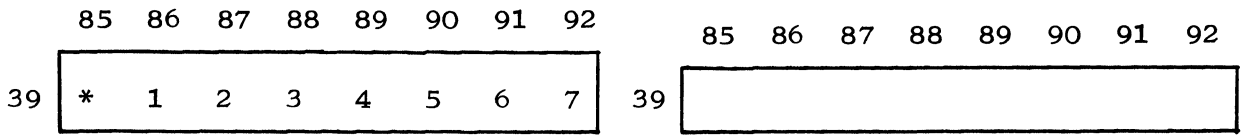
(b)

2000	R	0	0000	0000
2010	J	*	3000	3000
2020	N	1	3000	2999

What are A_f and B_f in both cases?

8. Show HSM final and A_f and B_f just before the "Halt" is staticized.

```
1000    R    1    0001    0000
1010    J    0    3989    3990
1020    J    *    3988    3988
1030    P    *    3990    3990
1040    .    0    0000    0000
```



A_f =

B_f =

9. Write a three-instruction program including the Halt to perform the same function as the following program:

```
1000    N    4    1055    0215
1010    M    "    021E    3000
1020    V    2    0000    0000
1030    V    4    1018    0000
1040    X    0    1058    1010
1050    .    0    3001    0012
```

10. List the Status Flow of the following program.

```
1000    R    1    0000    0000
1010    V    1    0225    1030
1020    .    0    0000    0000
1030    M    1    3000    3000
```

Write an equivalent two-instruction (including HALT) program.

H. S - INPUT-OUTPUT SENSE (IOS)

The Input-Output Sense instruction, like the CTC, is a decision making instruction. There are a number of differences between CTC and IOS, however. The CTC senses indicators of internal states of operation; the IOS senses indicators of the states of operation of peripheral devices. Since the choice of indicators for the CTC may be covered entirely by the information given by the N character, CTC is capable of transferring control to either the A or B address. The IOS has to choose from a much larger number of indicators, so more information is needed to determine which indicator is to be sensed. The N Character determines first which device will be addressed. A0 is used to determine which indicator of the addressed device will be sensed (A1, A2, and A3 are Ignored). Thus, control may be transferred only to the B address.

The CTC can provide a three-way decision: Transfer control to the A address, transfer control to the B address, or staticize and execute the next instruction in sequence. The IOS (since the A address is needed to determine which indicator is to be sensed) can provide only a two-way decision: transfer control to the B address, or staticize and execute the next instruction in sequence. The IOS goes through STP on transfer of control only.

NOTE: The IOS instruction is covered in the Input-Output instruction section of the Programmers' Reference Manual. Operations summaries are given separately for each N character, i.e., a different description for each device. See Sections VIII - XIV.

1. Instruction Format

Op. Code	N	A	B
S	Device to be tested (See Figure 49)	A0- Indicator to be checked. A1, A2, and A3 ignored. (See Figure 50)	HSM Address of next instruction to be executed if the condition or conditions being tested are present.

Device	N	
	Unit #1	Unit #2
Hi Data Tape Group	1, 2, 3, 4, 5, 6	A, B, C, D, E, F
33 KC Adapter	J	N
66 KC Adapter	L	P
Dual Tape Channel (2 x 6)	1, 2, 3, 4, 5, 6	
Dual Tape Channel (2 x 12)	123456 ABCDEF	
Paper Tape Reader	8	
Paper Tape Punch	9	
Card Reader or Read Unit of R-P	(	: :
Card Punch or Punch Unit of R-P	)	
On-Line Printer	7	G
Interrogating Typewriter	U	
Record File	R	Z
Record File Mode	# $. ,	
Data Disc File	R	Z

Fig. 49 IOS N Characters

DEVICE	"1" BIT IN	NUMERIC EQUIV.	TESTS
Magnetic Tape	2^0	1	Is the Tape Station Inoperable?
	2^1	2	Is the Tape in Motion?
	2^2	4	Has ETW been Sensed?
	2^3	8	Is Tape Positioned at BTC?
	2^4	&	Is Tape Moving in Reverse?
Paper Tape Reader or Punch	2^0	1	Is the Selected Device Inoperable?
	2^1	2	Is the Selected Device Operating?
Card Reader or Punch	2^0	1	Is the Selected Device Inoperable?
	2^1	2	Is the Selected Device Operating?
On Line Printer	2^0	1	Is the Printer Inoperable?
	2^1	2	Is a Line Being Printed?
	2^4	&	Is the Paper Advancing?
Interrogating Typewriter	2^0	1	Is the Typewrinter Inoperable?
	2^1	2	Has a Read Parity Error Occurred?
	2^2	4	Has "Program Interrogate" been received?
	2^3	8	Has a Write Parity Error occured?
	2^4	&	Has "Message Erase" been received?
Record File	2^0	1	Is the Record File Inoperable?
	2^1	2	Is the Device Reading or Writing?
	2^2	4	Is a Record on the Turntable?
Data Disc File	2^0	1	Is the Disc File Inoperable?
	2^1	2	Is the Device Busy?
	2^2	4	Is the Track Select Complete?
	2^3	8	Has Incorrect Parity been Read?

Figure 50 AØ Character of IOS

2. Instruction Operation

Example: S 7 1000 3650

The above instruction is sensing the Printer for non-operability. If the Printer is non-operable, the Computer will transfer control 3650. If the Printer is operable, the Computer will execute the next instruction in sequence.

Practice Problems

1. Write an instruction to sense for ETW on tape station B. If the Tape Station is at ETW, transfer control to 2550.

_____

2. Write an instruction to sense if the Card Reader (1st unit) is reading any cards. If yes, transfer control to 7580.

_____

3. Tape Station 5 is rewinding to BTC, when the following instruction is executed. What is the address of the next instruction to be executed? (Assume the S instruction is in memory beginning at 1000).

S 5 A000 1030

Address of next instruction _____

(See notice on next page.)

4. If the following instructions are attempted, explain what would occur. (Assume tape station 2 is rewinding).

1000 S 2 8000 1020
1010 V 1 0219 1000

1020 Remaining Portion of Program

Notice in Practice Problem #3 that two conditions are being sensed simultaneously since "A" contains both a 2^4 bit and a 2^0 bit. We shall see that if any condition sensed is present, transfer of control to the B address will take place.

3. Machine Operation

The IOS instruction uses three status levels: SIO, X1, and X2. The SIO (Sense Input-Output) status level is the foundation of the IOS instruction. (Refer to Figure 52) Two flip-flops are used by the SIO to determine which status level will be selected next: HO (Hold-Off) and JMP (Jump). Both flip-flops are reset at TP01 of the SIO. If HO is in the set state at TP6 of the SIO, another SIO is automatically selected. If HO is not set, X1 will be selected if JMP is set and P1 will be selected if JMP is not set. TP2 and the first line of TP5 and TP6 refer to the Hi-Data Tape Station and will be covered later. The JMP flip-flop will be set if the tested condition is present at TP5 of the SIO. Figure 51 gives an interpretation of levels used in setting JMP.

DEVICE	MOTION DENOTES	ETW DENOTES	BTC DENOTES	REVERSE DENOTES
Hi Data Tape	Any tape movement except Rwd.	ETW	BTC	Tape movement in reverse
33KC Tape	Any tape movement	ETW	BTC	Tape movement in reverse
66KC Tape	Any tape movement	ETW	BTC	Tape movement in reverse
Paper Tape Reader	Reading	NA*	NA*	NA*
Paper Tape Punch	Punching	NA*	NA*	NA*
Printer	Printing	NA*	NA*	Paper advancing
Record File	Reading or Writing	Disc on Turntable	NA*	NA*
Card Reader	Reading	NA*	NA*	NA*
Card Punch	Punching	NA*	NA*	NA*

Figure 51 Levels Used in Setting JMP During SIO

*NA = Not Applicable

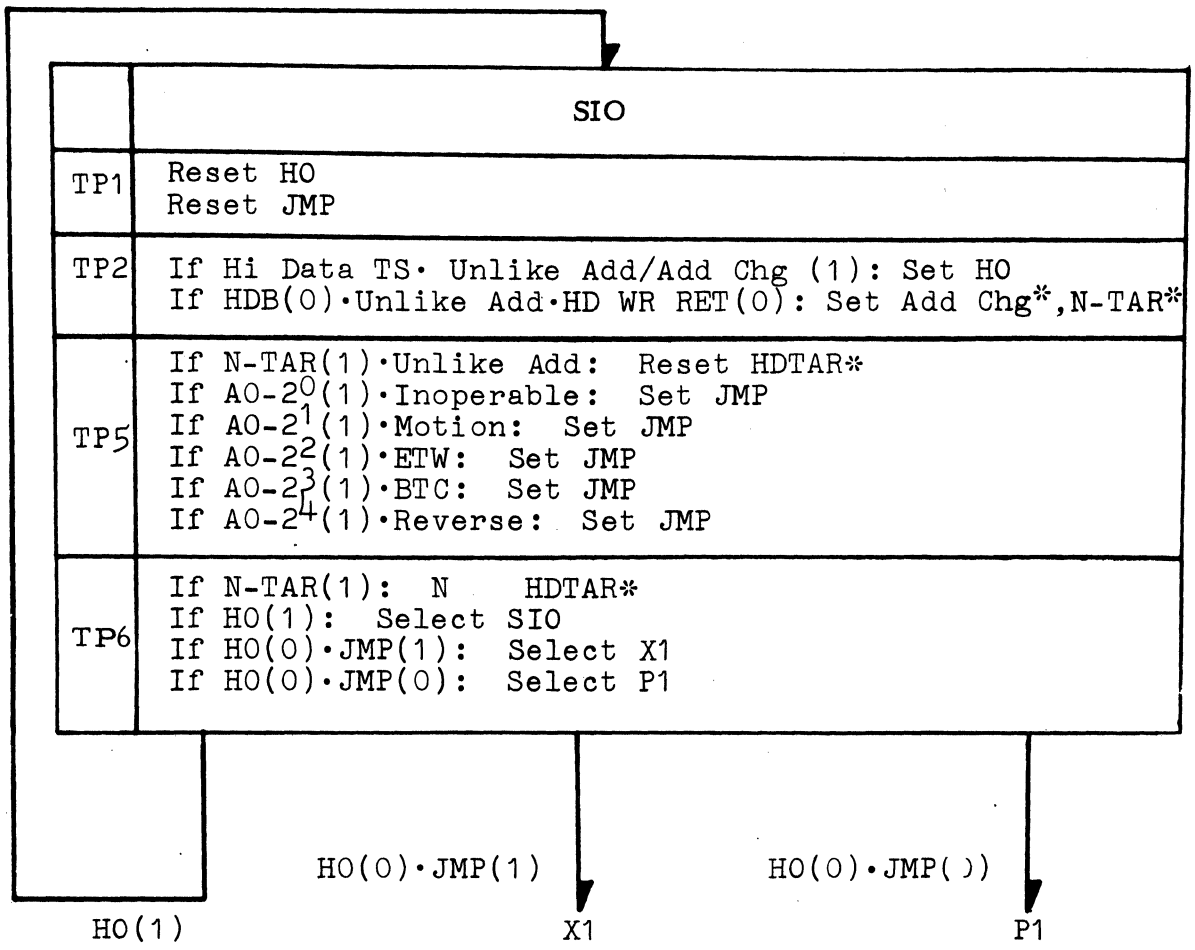


Figure 52 SIO of an IOS

Notice that if any one of the tested conditions is present, JMP will be set and if HO is not set, an X1 status level will be selected at TP6.

The X1 and X2 status levels are used to store the contents of the P register and to gate the contents of the B register into P. X1 and X2 are used only when the tested condition was found to be present and control is to be transferred to the B address.

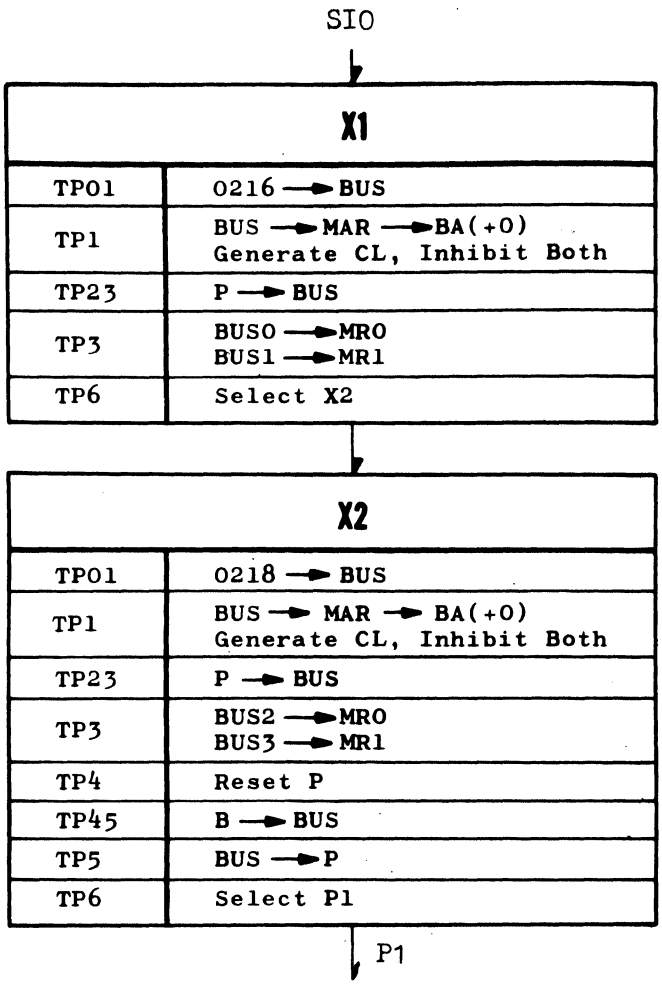


Figure 53 X1 and X2 of an IOS

There is one situation where the desired "sense" cannot be made immediately. This involves the Hi-Data Tape Station. The Hi-Data has multiple tape transports (2-6) sharing a common set of logic. In order to address transport #3 after performing an instruction on transport #2, for example, the logic must be "switched" from transport #2 to transport #3. Since relays are used to switch the logic channels, this operation takes several milliseconds, which is several thousand microseconds.

Now, if the programmer attempts to sense transport #4 to see if it is at BTC just after performing a read from transport #1, the computer will have

to wait until the Logic "switch" is made before #4 can be sensed. The HO flip-flop is used to keep the computer "waiting" until the switch has been made. At TP2 of the SIO status level of the IOS instruction, HO is set if a Hi-Data Transport is being addressed and if it is necessary to make an address change or "switch." Notice that with HO set, TP6 of the SIO will select another SIO status level. HO will continue to be set at TP2 of the SIO until the address change has been completed. Thus the computer will "cycle in SIO" until the logic switch is made.

The process of making this address change is also shown in the SIO chart of Figure 52. The "HDB(O)" found at TP2 of the SIO means that the Hi-Data Busy Flip-flop is not set. If HDB were set, it would indicate that one of the Hi-Data transports is executing a simultaneous mode instruction. For instance, "Unlike Add" means that the current N character which determines the transport to be addressed is different from the character in the Tape Address Register (TAR) which is the number of transport currently being used. "HD WR RET (O)" means that no Hi-Data Write Return is present; i.e., a Hi-Data Write instruction is not engaged in the execution of an instruction, and an "address change" is called for to set the address change flip-flop and the N-TAR (N character to the Tape Address Register) flip-flop.

At TP5 of the SIO if N-TAR is set and unlike addresses are present, the Hi-Data Tape Address Register will be reset. At TP6 if N-TAR is set, the N character is gated into the Hi-Data Tape Address Register. This action will initiate the pulling of the relays necessary to "switch" the Hi-Data Logic channels. HO will continue to be set at TP2 until the Address Change flip-flop is reset (actually 11ms, the amount of time the relays are allowed to switch the Logic). Finally, when the address change is completed, HO is not set, and depending upon the results of sensing the desired indicator, either an X1 or a P1 is selected.

4. Programming Errors

The Input-Output Sense instruction can be coded incorrectly and thus produce difficulty.

Since the input-output tests are specified by the A0 character bits, it is possible to test for more than one condition with a single instruction. However, only one "yes" reply to any given test will cause the computer to jump to the B address.

Example: S 2 6000 2000

The A0 character 6, is testing for tape in motion (2^1) and ETW having been sensed (2^2). If the computer jumps to address 2000 because of a yes response, the programmer would not know whether the response was from both tests or only one; nor would he know which of the two produced the yes reply.

Example: S 3 H000 4000

Since the A0 character is H, the tests to be performed are for tape moving in reverse (2^4) and BTC (2^3). If it is desired to rewind a given tape but it is not known whether or not the tape is at BTC, in the process of rewinding, or positioned beyond BTC, the above instruction could ask two questions at once. A "yes" reply would indicate that it is unnecessary to rewind trunk 3 while a "no" response would indicate trunk 3 can be rewound. (i.e., the tape on trunk 3 is positioned beyond BTC and is not moving in reverse.)

The A1, A2 and A3 characters are ignored for most combinations of S. However, constants being stored in the A3 position will cause indirect addressing to occur if the 2^4 bit is a "one" bit. It is very possible that a MAPE alarm or incorrect results may occur as a result of this indirect addressing.

Example: S 2 19DC 3000

The 301 would attempt indirect addressing in the above example and since the 2^4 bit of the A2 character is dropped in the A register, bad parity would result and a MAPE alarm would occur during the M1 status level.

Example: S 2 194C 3000

The 301 would again attempt indirect addressing. This time indirect addressing would be successful, but the test performed would then depend upon the new contents of A0 which would be the MSD of the new address.

Practice Problems

5. What is the difference between these two programs?

(a)
```
1000    S    2    4000    2000
1010    S    2    1000    2000
```

(b)
```
1010    S    2    5000    2000
```

6. What would be the state of the Hi-Data transport #3 if the instruction at 1010 is executed immediately after the IOS instruction?

```
1000    S    3    .000    2000
1010    .    0    0000    0000
```

7. Describe the purpose and execution of the following instruction?

```
S    )    4000    2000
```

8. Write a program to check for On-Line Printer Paper Advance. As long as paper is advancing, keep checking. When advance is finished, transfer control to 3000. Start the program at 1000.

9. Write an instruction at location 1000 to transfer to 3050 if the Paper Tape Reader is inoperable. Otherwise, execute the instruction at 1010.

10. Write a program to determine whether or not Hi-Data transport #6 is in rewind or at BTC. If it is, transfer to 3000. If not, transfer to 2000. Start program at 1000.

ANSWERS TO PRACTICE PROBLEMS

Pgs. III-144, 145, 146, 151, and 152

1. 0212 will contain 6
   0213 will contain 0
   0214 will contain 3
   0215 will contain 4

2.

	32	33	34	35
10	7	7	6	0

P

8	0	0	0

HSM AFTER

3. V 8 5108 0000 or V 8 5109 0000

4.

	66	67	68	69	70	71	72
77	*	*	*	*	S	A	M

HSM AFTER

5.

	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
89	O	N	E	@	T	W	O	@	O	N	E	@	T	W	O	@

HSM AFTER

6.

	60	61	62	63	64	65	66
66	R	C	C	A	R	*	A

HSM AFTER

7.

	05	06	07	08	09	10
50	1	1	0	1	0	6

HSM AFTER

	10	11	12	13	14	15	16	17	18	19	20
02	A	B	0	2	1	6	1	0	1	0	K

HSM AFTER

8.

	10	11	12	13	14	15	16		10	11	12	13	14	15	16	17	18	19
05	0	1	3	0	0	0	6	02	E	F	0	5	1	3	K	L	M	N

HSM AFTER

9. Notice that there are two bits in the N character of the Store Register instructions. There is a 2^1 bit, however, STA1 will be selected and "2004" will be stored in standard store A location 0212-0215. P is 1020 after execution.

10. Again there are two bits present in the N character of the Store Register instructions. Since no 2^1 bit is present, an A2 will be selected. However, the A2 will try to store both P and B. At TP3 a "20" from the P register and a "18" from the B register will both be gated into the MR, giving bad parity in both MR0 and MR1. The computer will stop with a MRPE.

Pgs. III-156, 157, 162, 163, and 164

1. W & 4320 5580

2. 2010

3. W 8 4650 4630

4.

	60	61	62	63	64	65	66
65	*	*	*	*	3	8	5

5. False. There are four conditions when a P1 is selected by P5 of the CTC directly. These are listed in Section III, C, 3a - Non-transfer of Control.

6.

1000	#	;	2000	2000		1000	P	;	2999	2999
1010	K	;	021E	0000		1010	L	;	021E	0000
1020	W	&	1030	1050		1020	W	&	1030	1050
1030	J	.	021E	021E	or	1030	J	.	021E	021E
1040	V	1	0219	4000		1040	V	1	0219	4000
1050	J	0	021E	021E		1050	J	0	021E	021E
1060	V	1	0219	4000		1060	V	1	0219	4000

7.

1000	J	0	2050	2100
1010	K	0	2050	2100
1020	W	1	1000	1000
1030	V	1	0218	3000

8. A3 and B3 of a CTC instruction usually contains "0" because they replace P addresses, and instructions usually start at a location with 0 and an LSD.

9.

1000	K	*	5000	5999
1010	W	1	2020	1080
1020	V	1	0218	3000

10. "Q" is an illegal N character for the CTC instruction since it contains <u>two</u> bits plus parity. Both the 2^5 bit and the 2^3 bit are present, so both the Normal ED/EF Indicator and the Simultaneous ED/EF Indicator will be sensed. In this case, sensing the Normal Indicator will transfer control to the B address and sensing the Simultaneous Indicator will transfer control to the A address. So at TP45 of the X2 status level, both the A and the B address will be gated into the P register. The addresses "1230" and "5300" when OR'ed together produce the address "5330". Control will be transferred to 5330 and since the newly created address had good parity, no MAPE will occur during staticizing.

Pgs. III-165, 166, 167, 170

1. Y 6 3145 7577

2. 5 , PRP

3. Y 3 7630 7634

   W 1 5500 5550

4.

	30	31	32	33	34	35	36	37	38
30	@	A	3	@	B	4	@	0	0

5.

2000	Y	1	1000	1001
2010	W	1	2040	2060
2020	J	E	1005	1005
2030	V	1	0219	3000
2040	N	1	1000	1005
2050	V	1	0219	3000
2060	N	1	1001	1005
2070	V	1	0219	3000

6. Either (1) A-B equality or (2) a non-compare

7. Twenty

8.

1000	Y	1	1086	5023					
1010	W	1	2000	3000					
1020	M	4	0532	1066	or	N	4	0535	1069

9. **Put a ")" in location 8000 before execution.**

1000	Y	1	8000	5239
1010	W	1	1020	1040
1020	J	1	5501	5501
1030	V	1	0219	3000
1040	J	2	5611	5611
1050	V	1	0219	3000

10.

1000	M	4	1022	0212
1010	Y	1	021E	1026
1020	V	2	9000	9099
1030	W	1	1080	1040
1040	M	4	0212	1022
1050	Y	4	1022	1026
1060	W	1	2000	1010
1070	V	1	0219	1010
1080	J	0	102E	102E
1090	V	1	0219	1040

Pgs. III-174, 175, 180

1. 34 times

2.

	16	17	18	19
02	4	2	5	0

3. X 0 4620 2300

	19	20	21	22
46	X	0	9	X

4.

	20	21	22	23	24	25	26
38	A	A	A	A	C	*	D

HSM AFTER

5. The program compares unknown character X with each of the six letters A, B, C, D, E, and F. The letter which is found equal to X (if any) is transferred to location 1010 and the computer stops. If X is not equal to any of the six characters, the computer will stop upon exhaustion of the Tally Quantity located in 2072.

6.

1000	M	4	1022	0212
1010	M	"	021E	3000
1020	V	2	2000	0000
1030	V	4	1019	0000
1040	X	0	1058	1010
1050	.	0	0000	0019

7.

	28	29	30	31	32	33
30	1	1	1	*	1	1

* Could be one or zero, depending on when looked at.

The SF instruction will continually replace the Tally Quantity with "11". Thus, the program will never leave the loop created by the Tally instruction.

8. The two SF instructions at 1000 and 1010 will perform correctly. The Tally instruction at 1020 is designed to continue transferring control to itself until the Tally Quantity is exhausted. However, the Tally Quantity is, in this case the first two digits of the A address of the SF instruction at 1000. So when the Tally in 1020 has reduced its Tally Quantity to "00", the Tally in 1030 will transfer control to 1000 (Note: The Tally Quantity is in this case "01".) The instruction in 1000 now reads: J 0 0000 3000. So, a WTT alarm will occur.

9. Tally will transfer control a certain number of times. "Store P" always transfers control.

10.

1000	M	4	1022	0212
1010	A	E_B	021E	2000
1020	V	2	3000	9950
1030	X	0	1026	1010
1040	X	0	1028	1010

Pgs. III-183 and 184

1.

1000	K	0	2000	3000
1010	W	1	1030	1040
1020	.	3	0000	0000
1030	.	2	0000	0000
1040	.	1	0000	0000

2.

1000	.	0	0000	0000
1010	V	2	0000	0000
1020	V	4	2135	0000
1030	M	4	0212	1056
1040	.	1	0000	0000

Manually insert numerical info in A and B register.

3. The Halt instruction uses no First Processing Level.

4.

2000	Y	1	1000	1001
2010	W	1	2040	2060
2020	J	E	1005	1005
2030	.	0	0000	0000
2040	N	1	1000	1005
2050	.	0	0000	0000
2060	N	1	1001	1005
2070	.	0	0000	0000

5. The first time Start is depressed, the Halt instruction in 1000 is staticized. Since both A3 and B3 contain 2^4 bits, indirect addressing will be attempted. In this case both P1 and M1 will be selected and "ORed" together giving a result of P1 with good parity. So the Halt at 1000 will operate properly.

   The second time Start is depressed, the Halt instruction in 1010 is staticized. Since only B3 contains a 2^4 bit this time, both a P1 and an M3 will be selected. The result is a $(37)_8$ with bad parity. The computer will stop with an STLE.

Pgs. III-189, 190, 191, 198, 199

1.

	00	01	02	03	04	05	06	07	08
30	6	4	#	9	3	#	P	G	H

HSM After

	01	02	03	04	05	06	07	08	09
40	2	3	8	G	#	X	#	9	3

HSM After

2. One combination:

R	2	0000	0000
M	=	1000	2000

3. 5538

4.

	00	01	02	03	04	05	06
40	A	E	E	E	E	F	G

5. Since both the A and B addresses are odd, the SF and DL instructions will each be executed twice using the addresses shown.

	10	11	12	13	14	15	16	17	18	19	20
31	0	1	*	*	*	5	0	1	8	9	&

6. P1-P5, X1, X2, P1-P5, A1, B, A1, B, REP1, REP2, P1-P5, A1, B1, A1, B1, P1-P5, HALT. A_f = 2996, B_f = 3998.

7. (a) **P1-P5, X1, X2, P1-P5, A2, A2, P1-P5, A1, B, REP1, REP2,**
P1-P5, A2, P1-P5, A1, B

A_f = 2999, B_f = 3999

(b) P1-P5, X1, X2, P1-P5, A2, P1-P5, A1, B,
A_f = 2999, B_f = 2998

Notice in (b) that although INHA and INHB are set during X1, they will be reset during P5 of the SF since the NR count = 0.

8.

	85	86	87	88	89	90	91	92
39	*	1	2	*	*	*	6	7

A_f 3989, B 3988.

9.

1000	R	B	0000	0000
1010	M	"	3001	3000
1020	.	0	0000	0000

10. P1-P5, X1, X2, P1-P5, A2, A4, P1-P5, A1, B, REP1, REP2, P1-P5, HALT*

1000	M	1	0223	1030
1010	.	0	0000	0000

Pgs. III-204, 210, 211

1. S B 4000 2550

2. S (2000 7580

3. 1030

4. Tape Station #2 is being sensed for BTC. If the Tape Station is still rewinding, the answer is no and the next instruction in sequence is executed. This instruction transfers control back to the Sense instruction. This little loop is continued until transport #2 reaches BTC. Then the computer jumps to 1020 to resume execution of the program.

5. They both will have the same effect.

6. The . in A0 of the IOS has a 2^4, 2^3, 2^1, and 2^0 bit. If any of the conditions specified by these bits is sensed, control will be transferred to 2000. So transport #3 must be not: in reverse, at BTC, in motion, or inoperable. Thus, #3 is operable, not at BTC, and tape is not moving.

7. This instruction is addressing card punch #1. The A0 character contains only a 2^2 bit. By consulting Table 8.2, we see that this will not perform a test, so it is impossible to set the JMP flip-flop. So, the next instruction in sequence will be executed.

8.
```
1000   S   7   &000   1000
1010   V   1   0218   3000
```

9.
```
1000   S   8   1000   3050
```

10.
```
1000   S   6   2000   2000   Is tape in motion?
1010   S   6   H000   3000   Is tape in reverse or at BTC?
1020   V   1   0219   2000
```

If transport #6 is in RWD, it will not be motion, but tape will be moving in reverse.

SECTION IV

ARITHMETIC INSTRUCTIONS

A. ARITHMETIC INSTRUCTIONS, INTRODUCTION

The Arithmetic Instructions, as their name implies, are used to perform arithmetic operations. This group is composed of two decimal and three binary bit instructions. The five Arithmetic instructions are:

	Op. Code		Instruction
Decimal Operations	+	-	ADD (ADD)
	-	-	SUBTRACT (SUB)
Binary Bit Operations	Q	-	LOGICAL OR (OR)
	T	-	LOGICAL AND (AND)
	U	-	EXCLUSIVE OR (EXO)

B. +/- ADD OR SUBTRACT (ADD OR SUB) Repeatable

The Add and Subtract instructions are the foundation of any mathematical function performed by the 301. These instructions perform algebraic addition or subtraction upon two equal length operands of up to 44 characters each. Only one addition or subtraction is performed at a time. The result is stored in memory in place of the augend or minuend (i.e. at the A address). The instructions operate from right to left and terminate upon decreasing the N count to zero. The PRI's are used to indicate the sign of the result. PRP is set if the sum or difference is positive; PRZ indicates that the sum or difference is zero; and PRN indicates a negative sum or difference. Neither the standard STA nor STP locations are used by the Add or Subtract instruction.

NOTE: Pages VI-3 through VI-7 of the Programmers' Reference Manual give operational outlines and cover exceptional cases of the Add and Subtract instructions.

1. Instruction Format

Operation Code: +/-

N Character: Number of characters per operand (0-44 using N count)

A Address: LSD location of 1st operand and result

B Address: LSD location of 2nd operand

NOTE: If N = 0, no characters are added or subtracted. The next instruction in sequence is executed.

The sign of each operand is incorporated in the 2^5 bit position of the LSD. If the 2^5 bit of the LSD is a one, the entire operand is negative; if the 2^5 bit of the LSD is a zero, the entire operand is positive.

Example 1:

+ 2 4205 4208

	04	05	06	07	08
42	4	K	3	9	1

HSM Before

	04	05	06	07	08
42	4	9	3	9	1

HSM After

PRP is set.

The augend in the above example is 4K or a minus 42. The addend is a positive 91. Algebraic addition produces a positive result of 49 and PRP is set. Note that the result (49) replaced the augend (4K). If a carry is generated on the last addition, the carry is incorporated as a one bit in the 2^4 position of the MSD and the overflow indicator SCAR is set, however, in this example, no carry is generated.

Example 2:

- 3 5102 5105

	00	01	02	03	04	05
51	7	2	P	5	8	4

	00	01	02	03	04	05
51	C	1	J	5	8	4

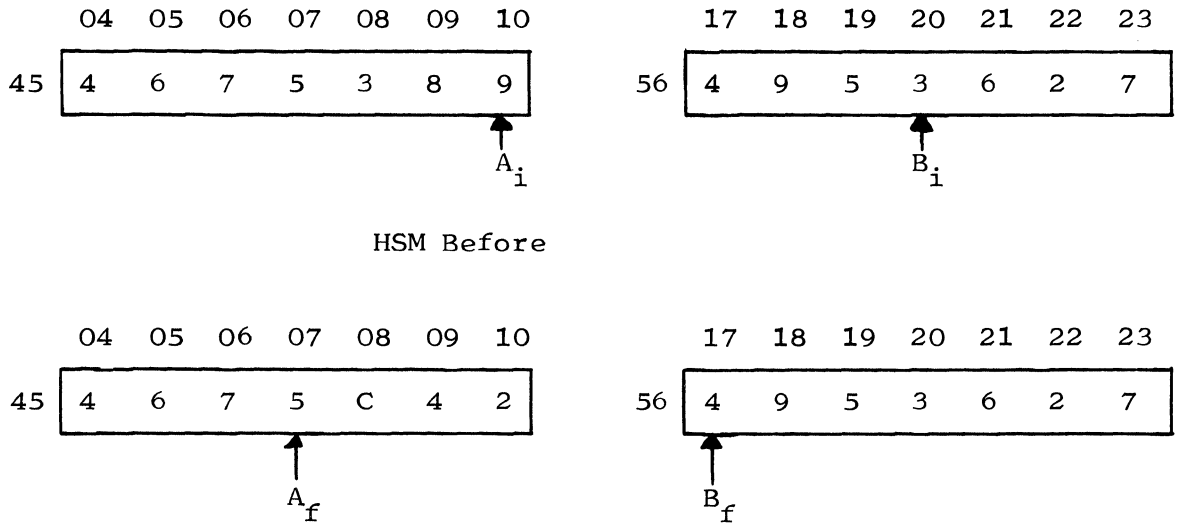
PRN is set.

In the above example, the minuend is 72P or a negative 727. The subtrahend is a positive 584 but due to the algebraic subtraction, the subtra-

hend appears to be negative and the result is a negative 1311. This is recorded as C1J.

2. Instruction Execution

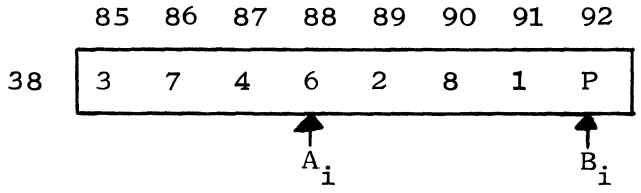
Example: + 3 4510 5620



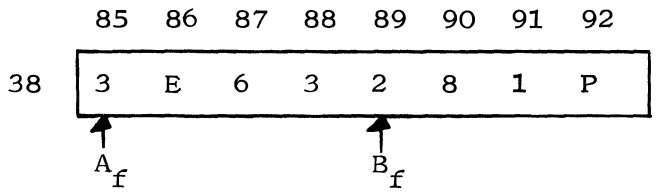
In this example, three additions were performed and the result is "C42". Note that a carry was produced from the addition of the most significant digits. Since only three characters were specified for each operand, the result must be three characters. Therefore, a carry, if it occurs, must be incorporated in the MSD of the result. This is done by inserting a 1 in the 2^4 bit position. The actual result should be 1342 in this example, but becomes C42 since a C is a decimal three with a 1 in the 2^4 bit position.

It should also be noted that in this case both operands were positive, thereby producing a positive result. Had the result been negative, the 2^5 bit in the LSD would have been a 1. A negative 1342 would be C4K, where K is a decimal two with a 1 in the 2^5 bit position.

Example: - 3 3888 3892



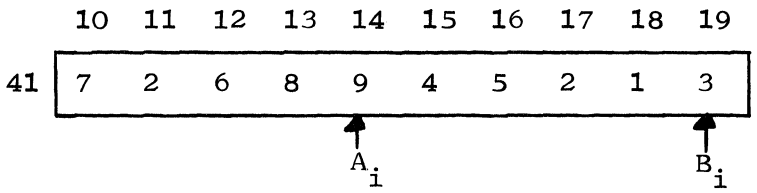
HSM Before



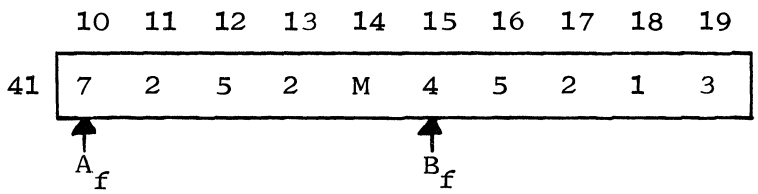
HSM After

In this example, a positive number, 746, is the minuend and a negative number, 817, is the subtrahend. Since the instruction is a Subtract, the computer effectively changes the sign of the subtrahend and adds. Thus, a positive 746 and a positive 817 give a result of 1563 or E63. This is a positive result, which would set PRP.

- 4 4114 4119



HSM Before



HSM After

In this example, the subtrahend (+5213) is larger than the minuend (+2689). Consequently, a negative result of 2524 is obtained, and PRN is set. The negative 2524 is placed in memory as 252M.

PRACTICE PROBLEMS

Problem No. 1

Execute the following instruction and show final HSM contents. Which PRI is set?

```
- 2 5652 5656
```

	50	51	52	53	54	55	56	57
56	6	7	P	8	9	5	2	3

HSM Before

	50	51	52	53	54	55	56	57
56								

HSM AFTER

PRI? _____

Problem No. 2

Execute the following instruction and show final HSM contents.

```
- 3 2622 2628
```

	20	21	22	23	24	25	26	27	28
26	8	6	5	1	3	2	5	6	2

HSM Before

	20	21	22	23	24	25	26	27	28
26									

HSM After

PRI? _____

Problem No. 3

Execute the following instruction and show final HSM contents.

+ 4 2003 4008

HSM Before

	00	01	02	03	04
20	8	6	5	3	2

	05	06	07	08	09
40	1	4	8	0	9

HSM After

	00	01	02	03	04
20					

	05	06	07	08	09
40					

PRI? ____________________

Problem No. 4

Execute the following instruction and show final HSM contents. Which PRI becomes set?

+ 3 4862 2113

HSM Before

	60	61	62	63	64	65
48	3	8	N	6	K	4

	09	10	11	12	13	14
21	B	4	6	4	4	A

HSM After

	60	61	62	63	64	65
48						

	09	10	11	12	13	14
21						

PRI? ____________________

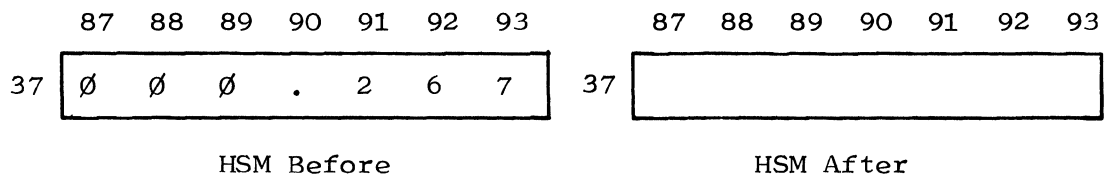
Problem No. 5

Write an instruction which will add ten characters. The MSD of the Augend is located at address 1003 and the MSD of the addend is at address 4632.

Problem No. 6

Execute the following instructions and show final HSM contents.

```
2500  K  Ø  3787  3793
2510  +  4  0215  2539
2520  J  Ø  3787  021E
2530  .  0  0000  0001
```



HSM Before HSM After

Problem No. 7

Two unknown operands exist in memory between 1000 and 1008, and 1010 and 1018. Each consists of an unknown number of digits from one to eight preceded by an @ symbol. Any characters to the left of the @ symbol will be insignificant zeros. Write a program which will add these two variable length operands and place the result in locations 1000 through 1008. Place an @ symbol one location to the left of the MSD of the result. Use address 2000 as the address of the initial instruction in your program. Show addresses of any constants used.

Problem No. 8

```
1000  #  Ø  1200  1200
1010  -  1  0215  1035
1020  J  @  021E  021E
1030  .  0  0001  0000
```

What is the purpose of the program above? Will it work? Why?

3. Machine Operation

The Add and Subtract instructions use just four status levels for the addition or subtraction of two characters. A "B" status level brings out one character and places it in D3; an A1 status level brings out the other character and places it in D2; a D status level obtains the result by table look up, and an A2 stores the result back in memory.

There are other status levels in the Add and Subtract instructions but they are not part of the basic operation. The X1 and X2 status levels perform preparatory steps such as setting and resetting certain flip-flops, and storing the A address at standard locations 0206-0209. The storing of the A address is done in case the computer must complement the answer. Complementation requires three additional status levels (A3, D, and A4) for each digit of the result.

The actual process of addition or subtraction is done by "table look-up". Prestored in memory, there is a sum table between 0000-0099 and a difference table between 0100-0199.

After each LSD has been brought out and placed in D3 and D2 (B and A1 status levels, respectively) the signs are examined along with the operation codes. If "like signs" are found, the address generator creates 00 on Bus 0 and Bus 1, D2 is placed on Bus 2 and D3 during the D status level. For example if 3 and 4 are the operands, the address 0034 is generated. At this address is a 7 which is brought out (also during the D status level) and stored in D3. (See Figure 54) The answer is then written back into memory at the A address during an A2 status level.

If "unlike signs" are found, the digits 01 are generated and the computer goes to the difference table for the result. Subtracting 5 from 8 would generate the address 0185. At this location would be a 3. Note that the computer can use the sum table or difference table for either an Add or Subtract instruction.

Notice that the sign of the instruction (+ or -) as well as the signs of the operands are considered in deciding whether these are "like" or "unlike".

In fact the only time a "+" instruction will differ from a "-" is in deciding whether or not to set the "unlike" signs flip-flop. This flip-flop determines whether the sum or the difference table will be accessed. (See ULS FF Fig. 5.)

Examples:

Add Instruction

Augend = 743
Addend = 752

Sum table is used with result positive D95.

Add Instruction

Augend = 81L
Addend = 49Ø

Difference table is used with result negative 32L.

Subtract Instruction

Minuend = 95N
Subtrahend = 322

Sum table is used with result negative B7P.

Subtract Instruction

Minuend = 7Ø5
Subtrahend = 699

Difference table is used with result positive ØØ6.

SUM TABLE HSM 0000-0099

Augend (D2) or Minuend

Addend (D3) or Subtrahend

	0	1	2	3	4	5	6	7	8	9
000	0	1	2	3	4	5	6	7	8	9
001	1	2	3	4	5	6	7	8	9	&
002	2	3	4	5	6	7	8	9	&	A
003	3	4	5	6	7	8	9	&	A	B
004	4	5	6	7	8	9	&	A	B	C
005	5	6	7	8	9	&	A	B	C	D
006	6	7	8	9	&	A	B	C	D	E
007	7	8	9	&	A	B	C	D	E	F
008	8	9	&	A	B	C	D	E	F	G
009	9	&	A	B	C	D	E	F	G	H

DIFFERENCE TABLE HSM 0100-0199

Augend or Minuend (D2)

Addend or Subtrahend (D3)

	0	1	2	3	4	5	6	7	8	9
010	0	I	H	G	F	E	D	C	B	A
011	1	0	I	H	G	F	E	D	C	B
012	2	1	0	I	H	G	F	E	D	C
013	3	2	1	0	I	H	G	F	E	D
014	4	3	2	1	0	I	H	G	F	E
015	5	4	3	2	1	0	I	H	G	F
016	6	5	4	3	2	1	0	I	H	G
017	7	6	5	4	3	2	1	0	I	H
018	8	7	6	5	4	3	2	1	0	I
019	9	8	7	6	5	4	3	2	1	0

Figure 54 Sum and Difference Tables

The 2^4 zone bit is used to recognize a carry or borrow during an arithmetic operation. For example, assume that the characters 8 and R are being subtracted. The subtrahend being negative would appear positive for algebraic subtraction, and "like signs" would be recognized. The address 0089 would be formed and at this address would be a 7 with a carry (2^4 bit or a G). During the D status level, when the answer is brought out, the 2^4 bit is examined and if it is a one bit, Sum Carry (SCAR) becomes set. SCAR sets Initial Carry (ICAR) and before the next addition takes place D3 is triggered up one, thus incorporating a carry.

The reader should note that in most cases the 2^4 zone bit is eliminated prior to writing the answer back into memory.

The 2^4 bit is also used to execute a borrow by using the same steps. Assume the minuend is 3K and the subtrahend is 1M during a subtract operation. The K and M are brought out in D2 and D3, respectively, and unlike signs are found. The address 0124 is generated during the D status level and at that location is an 8 with a borrow (4 from 2 = 8 with borrow) or an H. SCAR is set, and the first character of the result (8) is written back into memory during the A2 status level. (Actually a Q is written to show the negative sign of the result.) The next series of status levels, B and A1, bring out the 3 and 1. Since SCAR was set, ICAR becomes set and D3 is triggered up to 2. The address 0132 is generated during a D status level, and a 1 is brought out of memory. Hence a borrow occurred in the same manner as a carry - the difference being the table that was addressed.

The Rule is:

If a 2^4 bit comes from the Sum table it is recognized as a carry.

If a 2^4 bit comes from the Difference table, it is a borrow.

Since the computer cannot recognize the MSD's of the operands until it has first processed the other characters, the machine must predict the sign of the result during the first A1 status level. It can be stated that the computer will assume the sign of the character in D2 (the A operand) as the sign of the result. Only in two instances will the computer be wrong:

(1) When the subtrahend is larger than the minuend and the Difference Table is being addressed; and (2) When a zero result is obtained and a negative result is predicted. For both of these cases, the computer must go through complementation.

Figure 55 is a block diagram of an add or subtract not requiring complementation. Figure 56 is a simplified chart of the B and A1 status levels; and Figure 57, a simplified view of D and A2. Notice how a carry (or borrow) is performed at TP4 of the A1. (Incrementing D3 will add one to a sum or subtract one from a difference.) At TP6 of the A1 status level, if an add instruction is being executed and the signs of the addend and augend are unlike, or a sub instruction is being executed and the signs of the subtrahend and minuhend are like, the ULS FF will be set. If ULS is set, the difference table will be addressed during the D status level. ULS and other important flip-flops are shown schematically in Figure 58.

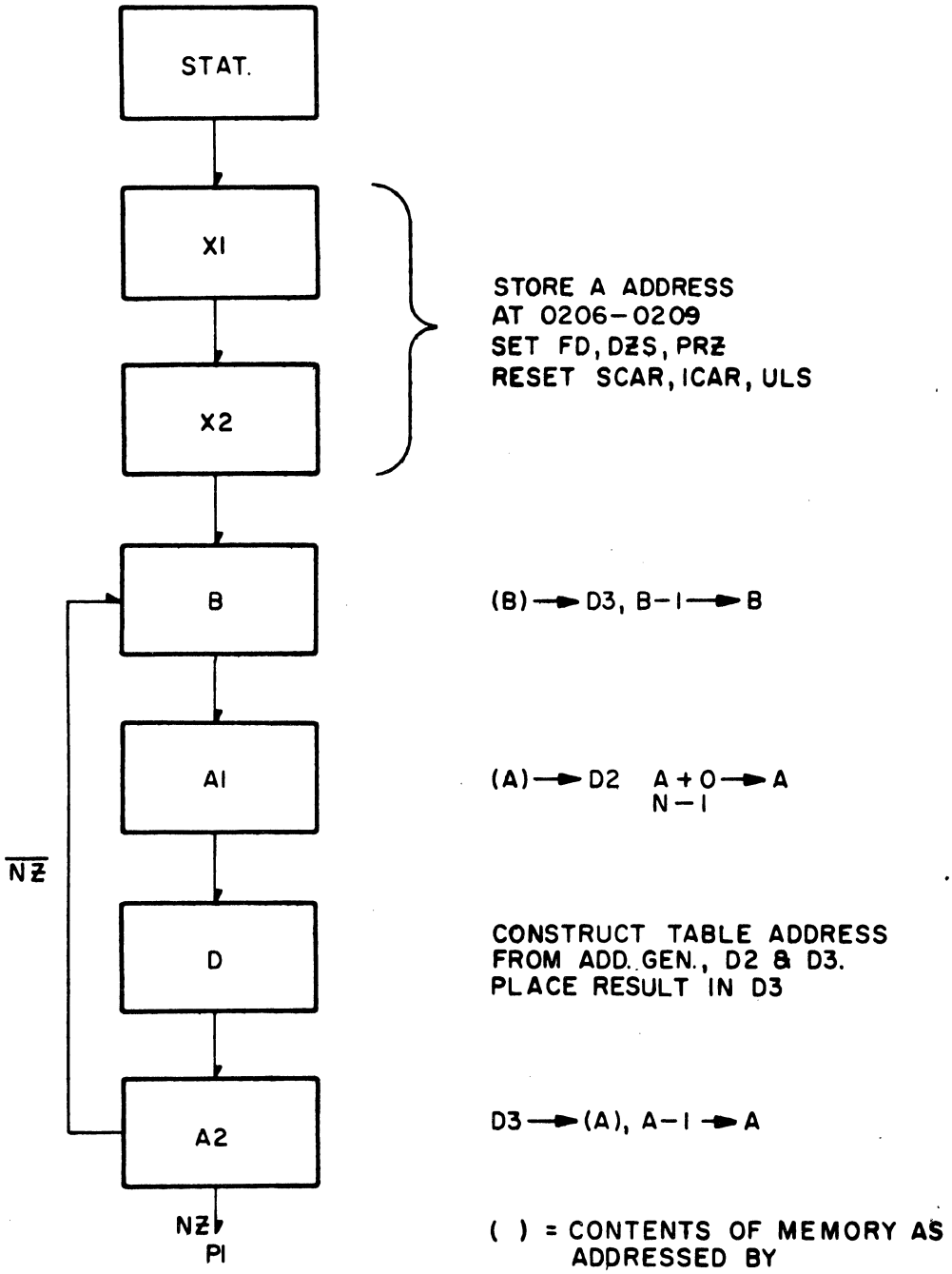


Figure 55 Basic Block Diagram of Add or Subtract

B	
TP01	B address onto Bus.
TP1	Contents of Bus to MAR. Generate Command Level and decode address in MAR. Modify address in Bus Adder by minus one.
TP2	Reset B register.
TP23	Permit both characters being read to reach MR. Gate modified address from Bus Adder to Bus.
TP3	Gate Bus contents into B.
TP4	Reset D register.
TP45	Regenerate diad. Gate MRO or MR1 onto Bus 3 as directed by MAR address.
TP5	Gate contents of Bus 3 into D3.
TP6	Select A1 status level.

A1	
TP01	Gate A address onto Bus.
TP1	Contents of Bus to MAR and Bus Adder where modified by 0. Generate a command level and decode MAR address. If SCAR is set, set ICAR.
TP2	Reset A register. Reset SCAR.
TP23	Gate contents of Bus Adder onto Bus. Permit both characters being read out to reach MR.
TP3	Gate contents of Bus to A register. Trigger N down one.
TP4	If ICAR is set, trigger D3 up one.
TP45	Regenerate diad. Gate MRO or MR1 onto Bus 2 as directed by MAR address.
TP5	Gate contents of Bus 2 to D2. Reset ICAR.
TP6	If FD is set and D2 - 2^5 is set, set PRN. If FD is set and Add instruction has opposite signs in D2 and D3, set Unlike Signs (ULS). If FD is set and subtract instruction has same signs in D2 and D3, set ULS. Select D status level.

FIGURE 56 B AND A1 OF ADD OR SUB

D	
TP01	Generate zero onto Bus 0. If ULS is reset, generate zero onto Bus 1. If ULS is set, generate 1 onto Bus 1. Gate D2 onto Bus 2 and D3 onto Bus 3.
TP1	Gate address from Bus to MAR. Generate command level and decode MAR address.
TP23	Permit both characters to be read out into MR.
TP4	Reset D register.
TP45	If MAR address is even, gate MR0 onto Bus 3; if odd, gate MR1 onto Bus 3. Regenerate diad.
TP5	Gate contents of Bus 3 into D3.
TP6	If D3 - 2^4 is set, set SCAR; if not D3Z or if D3 - 2^4 is set, reset DZS. Select A2 status level.

A2	
TP01	Gate A address to Bus.
TP1	Contents of Bus to MAR and Bus Adder where one is subtracted. Generate command level and decode MAR address. If $\overline{NZ}$, reset D3 - 2^4. If FD is set and PRN is set, set D3 - 2^5.
TP2	If DZS is reset and PRN is reset, set PRP. Reset FD.
TP23	Inhibit one character from reaching MR during read out as addressed by MAR. Gate D3 onto Bus 3.
TP3	Gate contents of Bus 3 into MR0 or MR1 as directed by MAR address.
TP4	Reset A register.
TP45	Regenerate diad. Gate address from Bus Adder onto Bus.
TP5	Gate contents of Bus into A register.
TP6	If $\overline{NZ}$ select B status level. If NZ select P1 status level.

FIGURE 57 D and A2 of ADD or SUB

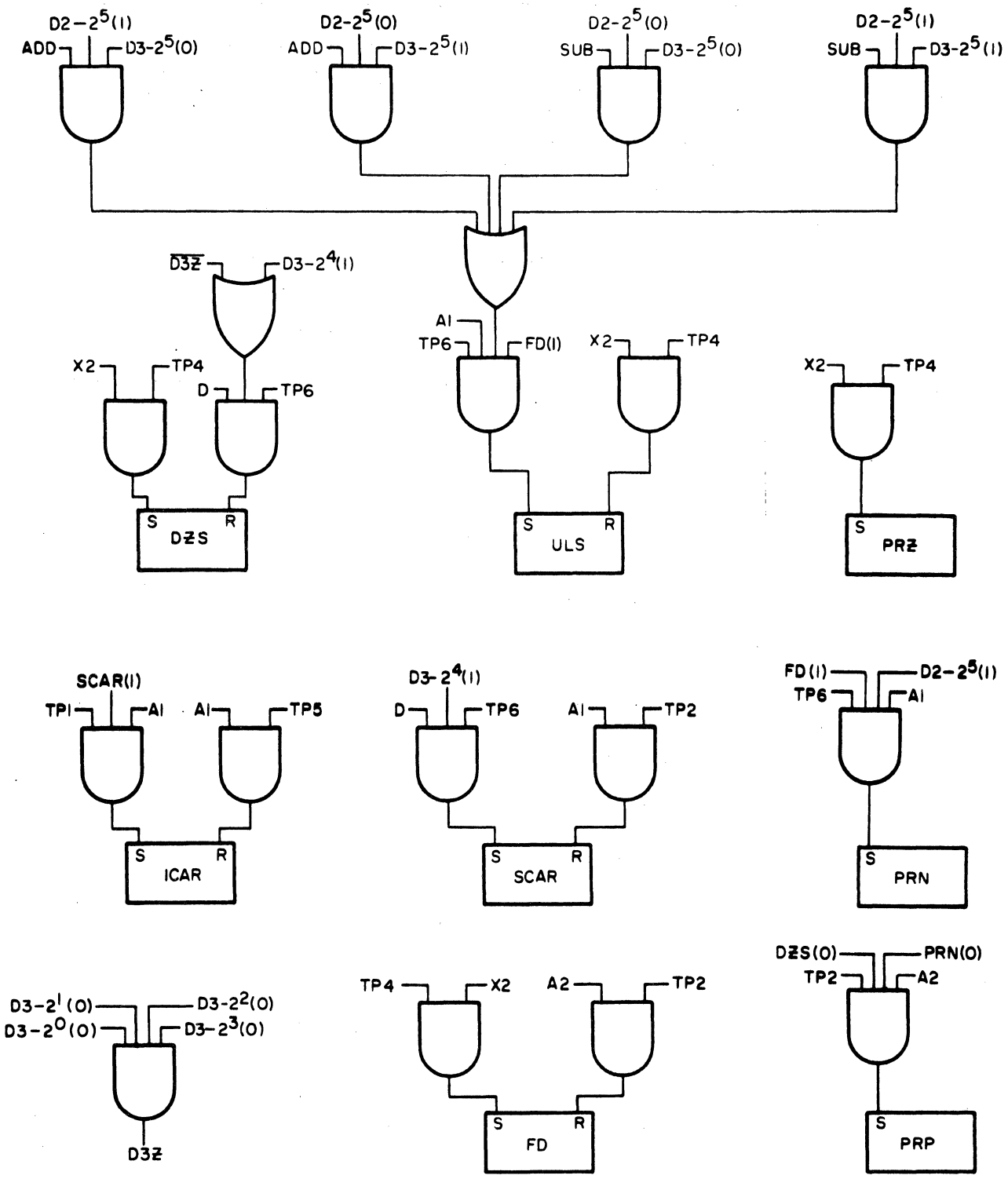


FIGURE 58 FLIP-FLOPS

The following is an example of the status flow of an add instruction not requiring complementation of the answer.

+ 2 1506 1509

	04	05	06	07	08	09
15	6	4	M	3	2	9

HSM Before

P1 -	P5
X1	
X2	
B	9 → D3, 1508 → B
A1	M → D2, Set ULS, Set PRN, N-1 = 1
D	Generate address 0149, E → D3, Set SCAR, Reset DZS
A2	Reset D3 - 2^4, Set D3 - 2^5, N → (A) 1505 → A, Reset FD
B	2 → D3, 1507 → B
A1	Set ICAR, Reset SCAR, 4 → D2, Trigger D3 to 3, N-1 = 0
D	Generate Address 0143, 1 → D3
A2	1 → (A), 1504 → A
P1	

	04	05	06	07	08	09
15	6	1	N	3	2	9

HSM After

Figure 59 shows the complete block diagram of the status flow of add or subtract. The right-hand branch is used to complement the answer when a mistake is made in predicting the result.

Example: - 2 1001 2001

	00	01
10	2	3

HSM

	00	01
20	5	1

HSM

Since the A operand (the minuend) of the instruction above is a positive number, the processor assumes that the result will be positive. ULS is set so the difference table is addressed. If you were to perform the subtraction:

```
  23
- 51
----
```

100 must be subtracted from the result previously obtained which contains a borrow. This is performed by the X3, X4, A3, D, and A4 status levels. The EAC flip-flop will be set at the end of the last A2, if the result obtained by that time must be complemented.

At TP6 of the A2 status level (Figure 63) EAC will be set if "NZ·AA". "NZ" just means that all digits of the operands have been added or subtracted. "AA" is defined beneath the chart of the status level. There are 3 ways to get "AA": (1) When a borrow was made while handling the MSD's during a "subtract" (i.e., ULS set) and the MSD of the minuend was equal to 9 or less or the MSD of the subtrahend was greater than 9. (2) When a negative result was predicted but a zero result was obtained, and the MSD of the minuend was equal to 9 or less or the MSD of the subtrahend is greater than 9. (3) When the difference table is being used and the MSD of the minuend is 9 or less and the MSD of the subtrahend is greater than 9.

If any of these conditions are present at TP6 of the last A2, EAC will be set and X3 selected. The purpose of the X3 and X4 status levels is to replace the present contents of the A register with the address of the LSD of the minuend (which is also the result). This is the A initial of the instruction which was stored in standard location 0206-0209 by X1 and X2 immediately after staticizing. Now the process of complementing can begin.

In the example of 51 subtracted from 23, the answer was larger than it should have been by the amount of the borrow (100). Subtracting 100 from 72 will give the correct answer. This is the same as subtracting 72 from 100 and making the result negative which is just what the computer does. During the A3 status level, "2", the LSD of the false answer, is sent to D3 and a Ø, representing the LSD of "100", is sent to D2. Table look-up is

performed by the D status level. Notice that the difference table is still being used since we are subtracting 72 from 100. During A4, a 2^5 bit is added to the character from the table to make it negative and the final result ("Q") is written into memory as the new LSD. During the next A3, the "G" is brought out of memory and put into D3 and a 9 is put into D2. Since a borrow had to be made to subtract "2" from "Ø" during the first A3, the "2" was actually subtracted from 10; and now the "70" must be subtracted from "90". (i.e., $\begin{array}{r} 100 \\ -\ 72 \\ \hline \end{array} \rightarrow \begin{array}{r} {}^{9}\not{1}{}^{1}\not{\text{Ø}}0 \\ -\ 72 \\ \hline \end{array}$)

The D status level performs table look-up, and the A4 stores the result ("2") in memory. "LD" is now set because the MSD of the result contained a 2^4 bit. This will always be the case when complementing is to be performed since this 2^4 bit indicates a "borrow". (Note: No digits of the result except the MSD or LSD may contain a 2^4 bit.) The final "difference" in our example is then "2Q" which represents "-28".

Reviewing the example briefly:

A. "23-51" gives a result of "G2" after the last A2.

B. Since a borrow was generated when subtracting the MSD's of the operands, complementation must be performed.

C. "72-100" = -28 = "2Q" is the final result after complementation is complete.

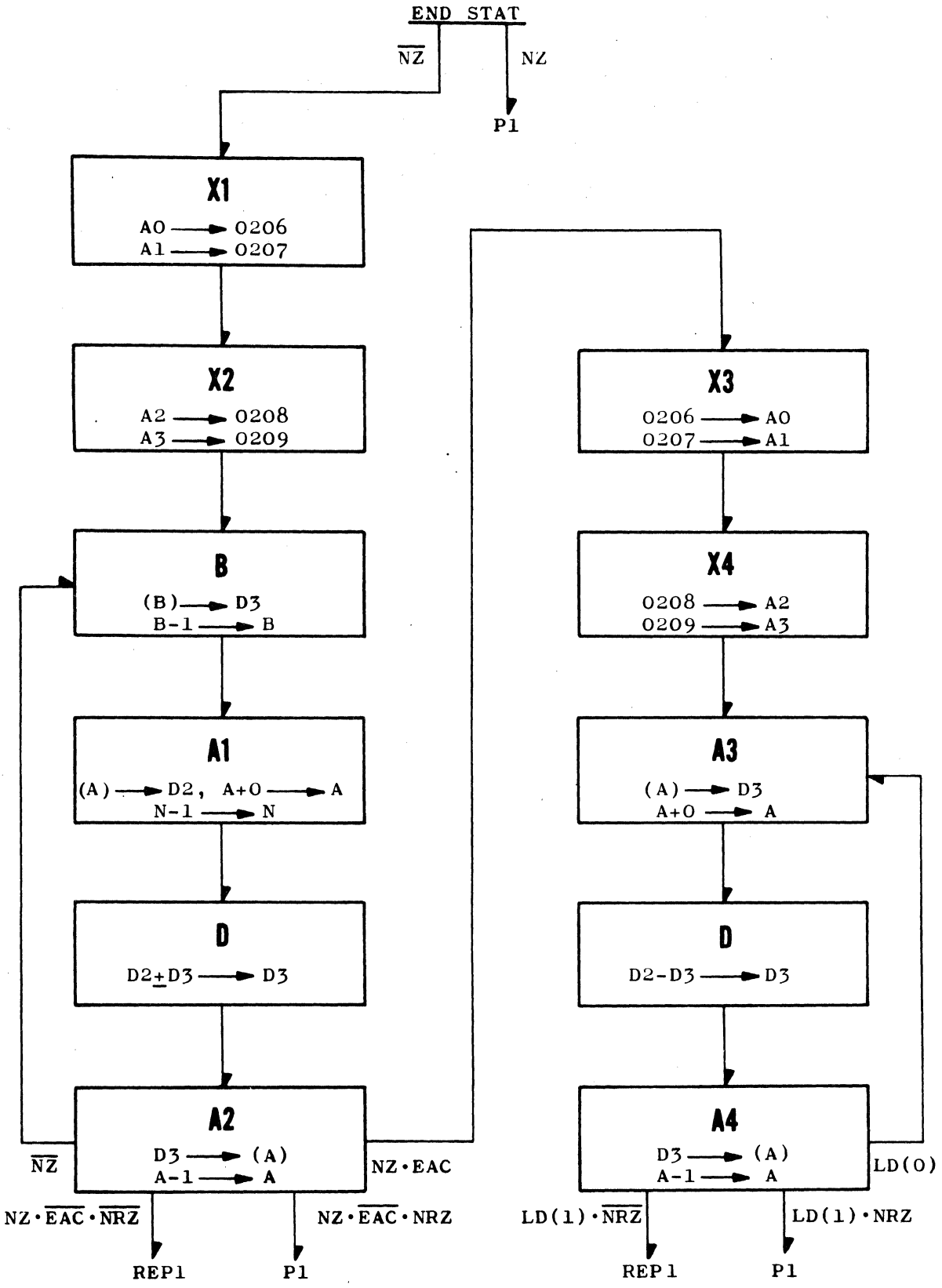


FIGURE 59 COMPLETE BLOCK DIAGRAM OF ADD OR SUBTRACT

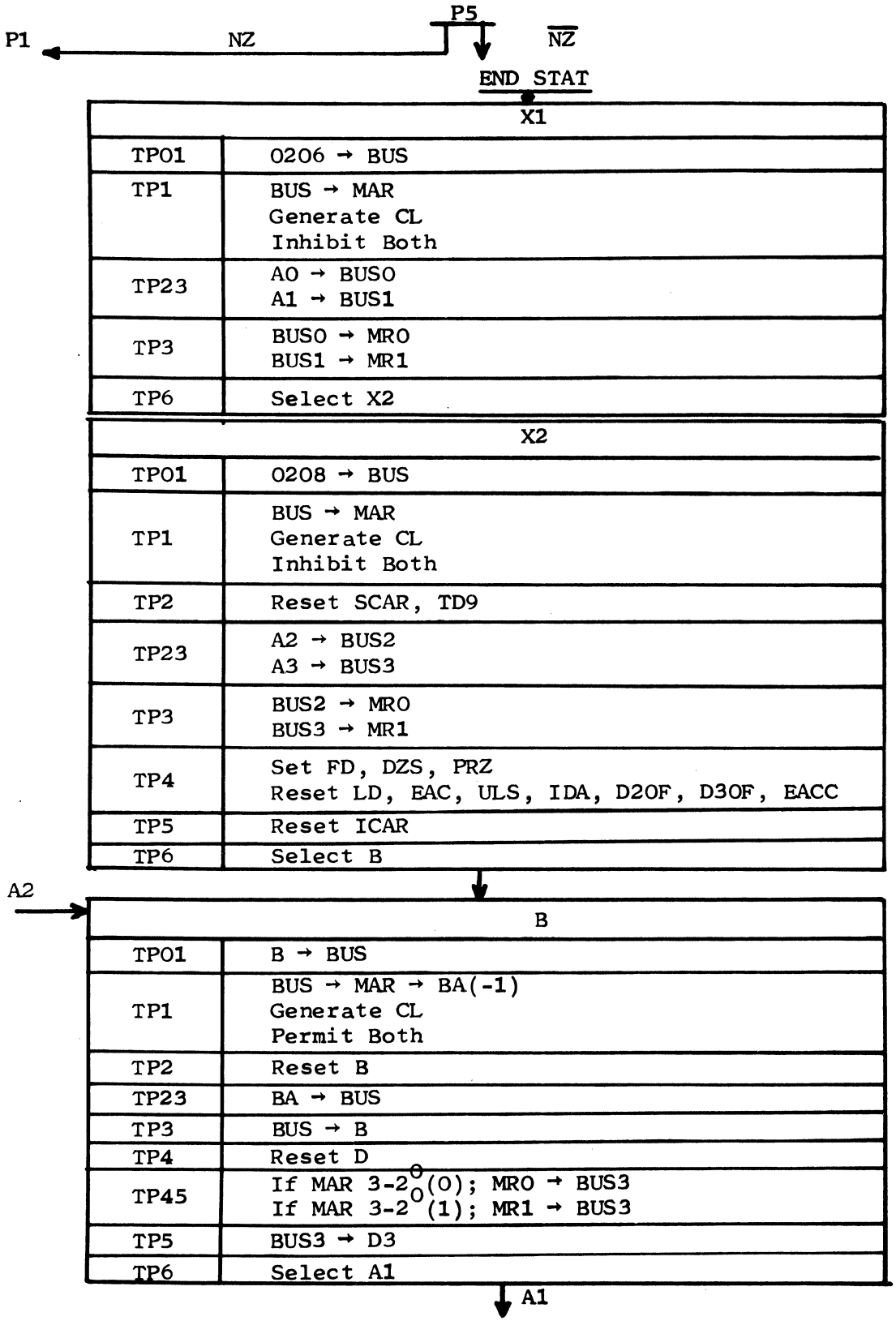
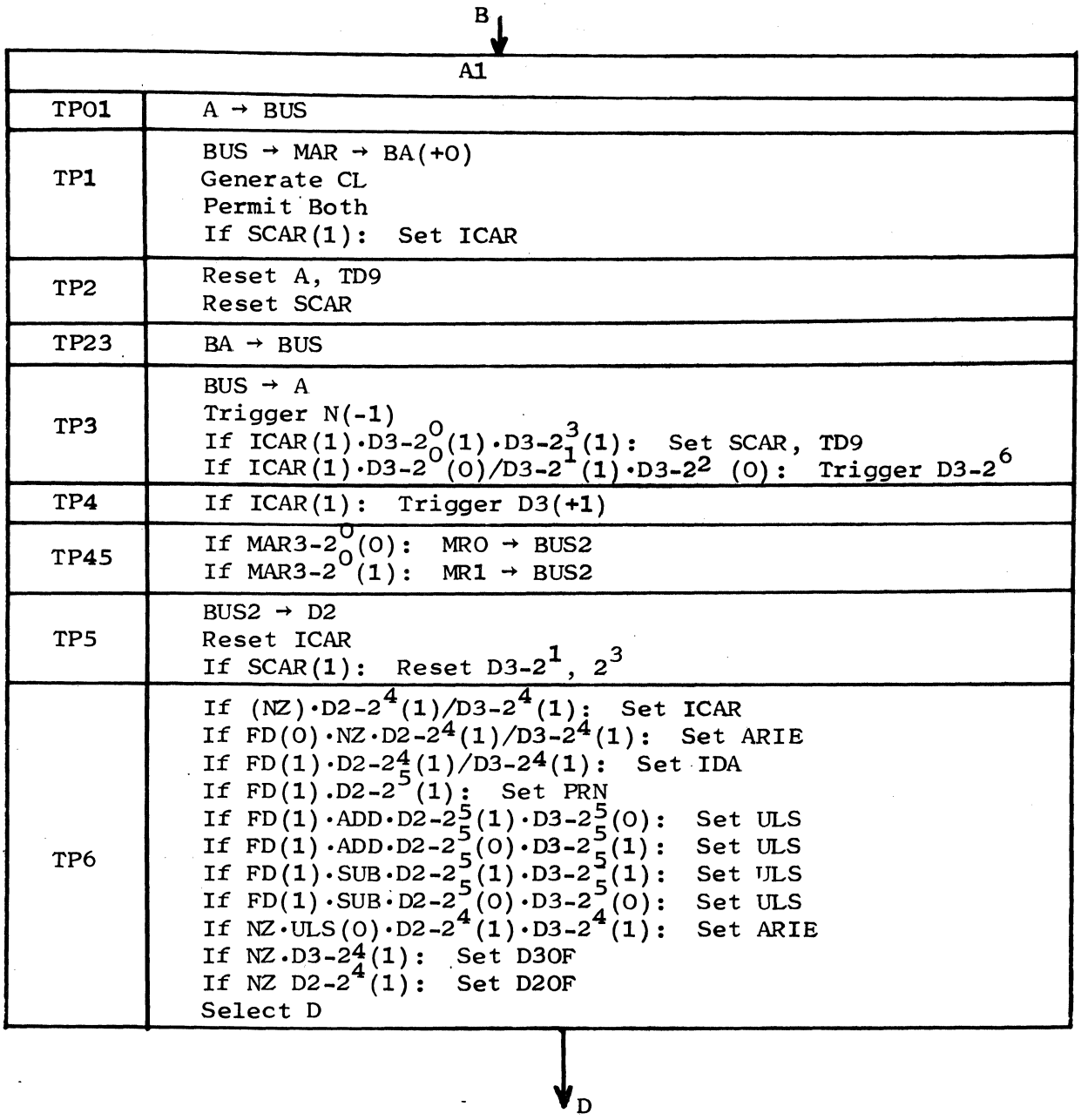
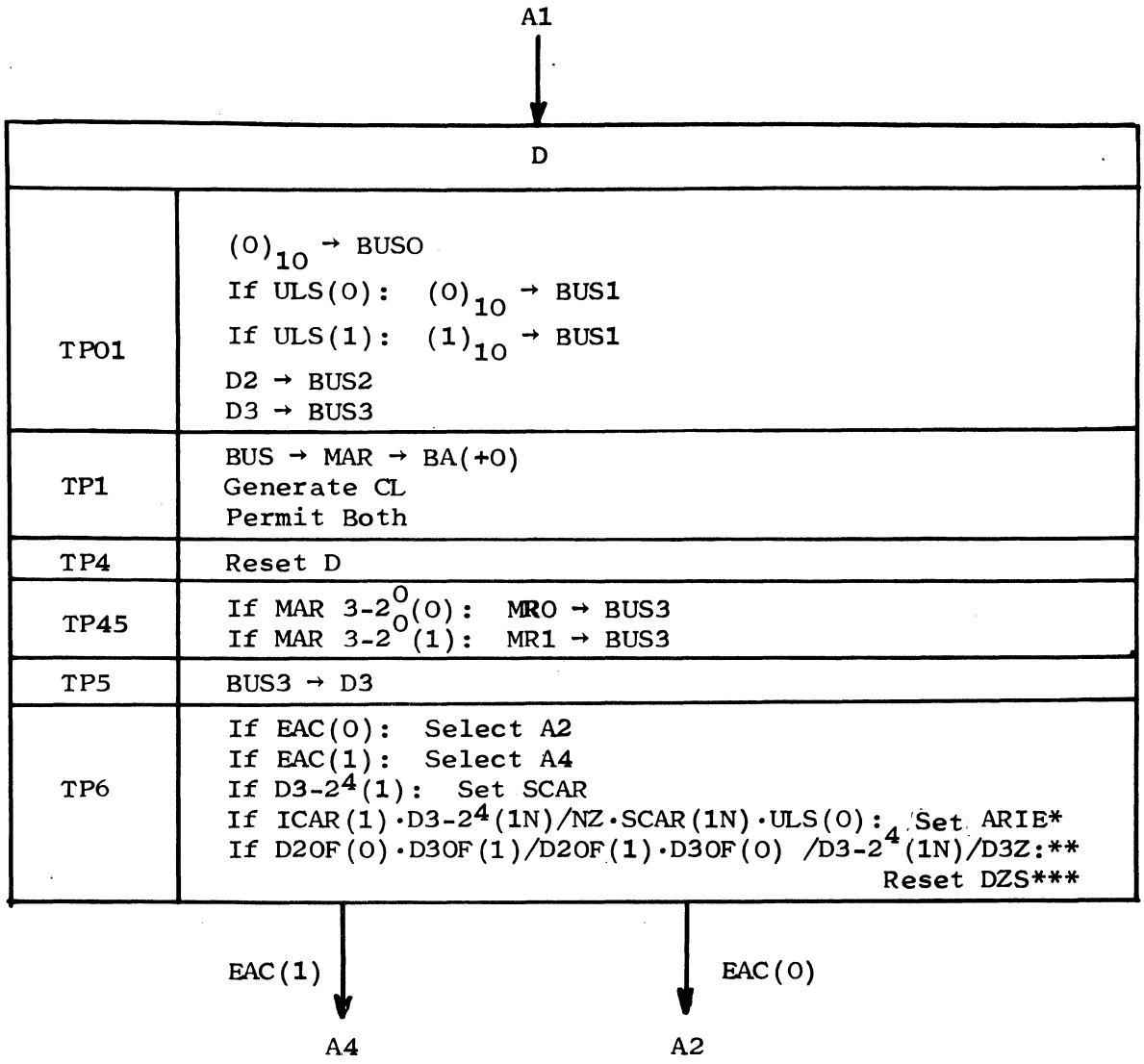


Figure 60 Detailed Status Flow (X1, X2, B)



*D2OF/D3OF = See Page IV-249

Figure 61 Detailed Status Flow (A1)

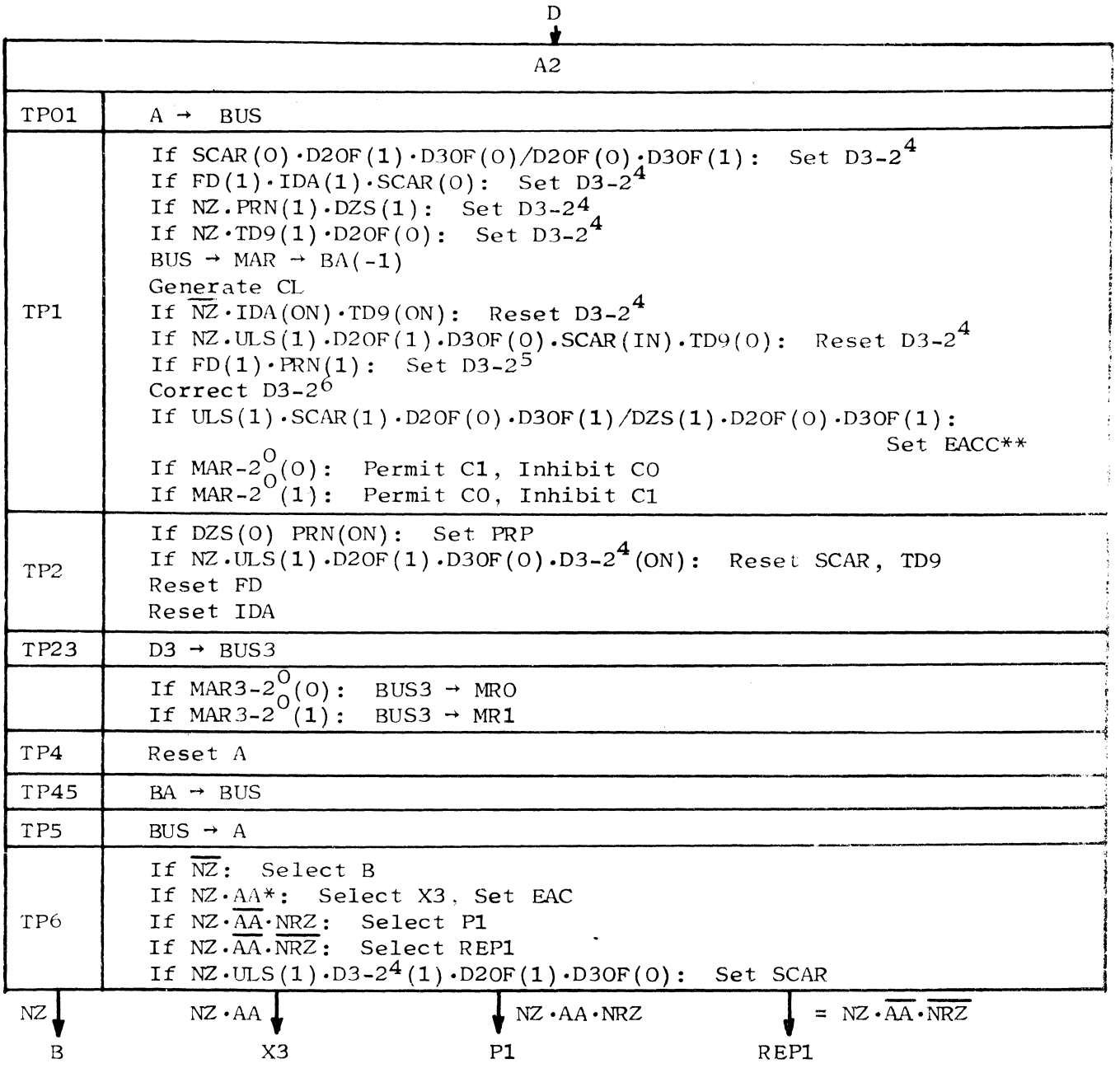


*ARIE - See Page IV-253

**D3Z = indicates the decimal portion of the D3 register (2^0 thru 2^3) contains zero (0000).

***DZS = See Page IV-249

Figure 62 Detailed Status Flow (D)



*AA = ULS(1)·SCAR(1)·$\overline{\text{D2OF(1)·D3OF(0)}}$ / DZS(1)·PRN(1)·$\overline{\text{D2OF(1)·D3OF(0)}}$ / ULS(1)·D2OF(0)·D3OF(1)

**EACC = For explanation see Page 30.

(A2)

Figure 63 Detailed Status Flow

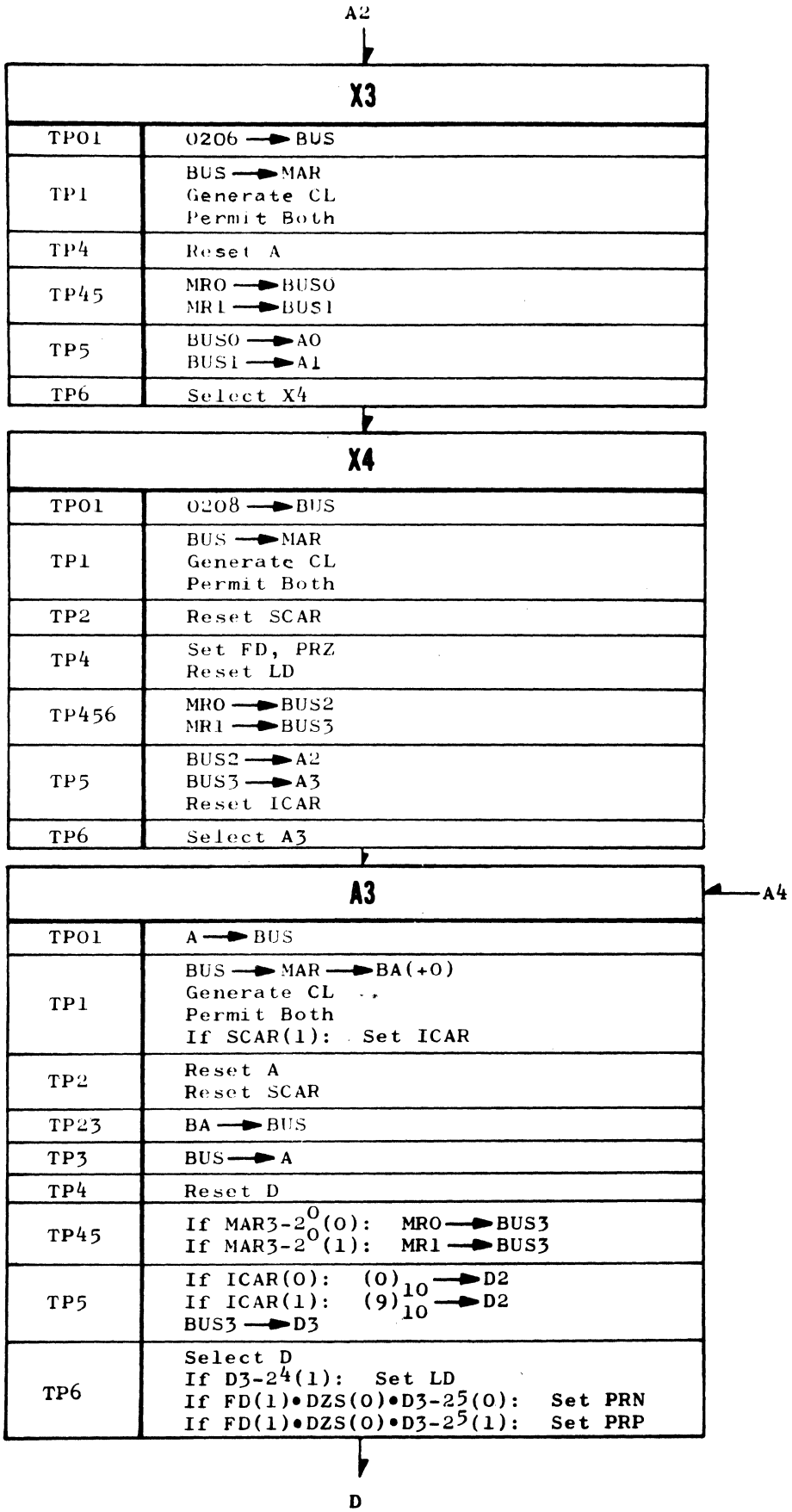
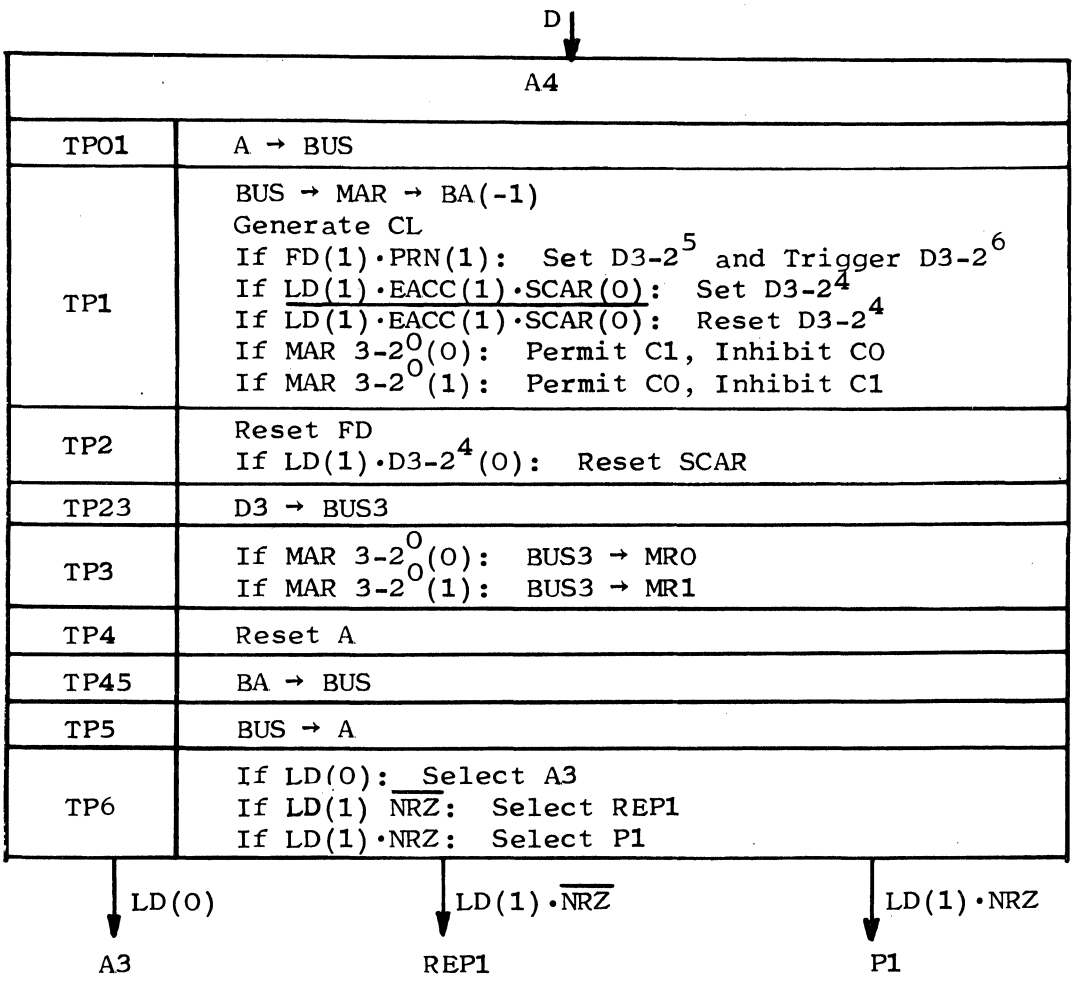


Figure 64 Detailed Status Flow (X3, X4, A3)



NOTES:
- Sign is in 2^5 bit of LSD: $2^5(0)$--positive, $2^5(1)$--negative.
- Final carry is stored in 2^4 bit of MSD and SCAR is set.
- Operands must not contain a 2^4 bit in any position other than the LSD or MSD.
- Indirect address is indicated by $2^4(1)$ in LSD.
- Addresses should always be positive.
- If the sum of any MSD exceeds 19 (I), an Alarm Stop will occur (excluding 305 Processor).

Figure 65 Detailed Status Flow (A4)

4. Special Flip-Flops

1. Overflow Indicators (D2OF and D3OF)

The purpose of these flip-flops is to indicate if either or both of the operands contain a carry bit (2^4) in the most significant character position. The condition of these flip-flops will be checked at various times and their state will help determine the steps the processor will take in executing an add/sub operation.

2. Decimal Zero Sum FF (DZS)

DZS will be in the set state at the completion of an add or subtract instruction, if the end result is zero.

DZS is set initially during the X2 status level. An attempt is made to Reset DZS during every D status level. If the character taken from the table area of HSM is something other than zero, or if either (not both) of the overflow indicators (D2OF/D3OF) is set, TP6 of the D status level during which this occurred will cause DZS to be reset.

EACC: The "End Around Condition Carry" flip-flop is used in certain special cases during an add or subtract to insert a 2^4 bit (carry or borrow) into the most significant character of the result.

This can only occur when an answer must be complemented, and then only in certain special cases. The following example will illustrate one of these cases.

1. Augend = 05-
   Addend = +I60

Decimal Equivalent →

-0050
(+)+1960
+1910
Correct Result

In this example, a -50 is being added to a +1960. Since the sign of the augend is negative, PRN will be set. The signs are unlike. Therefore, ULS will be set. This tells the processor to utilize the difference table during execution of the add instruction.

Starting with the least significant characters, the first address generated during the first "D" status level will be 0100. The zero (0) in this location will become the LSD of the result.

2.
```
  05-
 +I60
------
    0  LSD of Result
```

However, since the processor predicted a negative result, this must be indicated in the LSD. During the first "A2" status level, the signal "FD·PRN" will set the 2^5 bit in the LSD of the result. This changes the zero to a minus.

3.
```
  05-
 +I60
------
    -  Set 2^5 Bit
```

During the second "D" status level the address 0156 will be generated. The I in this location indicates a 9 with a borrow (2^4 bit).

4.
```
  05-
 +I60
------
  I-      2^4 indicates borrow
```

The 2^4 bit will cause the SCAR FF to become set. SCAR now indicates the borrow.

During the second A2 status level the signal "$\overline{NZ}\cdot\overline{IDA}\cdot\overline{TD9}$" will cause the 2^4 bit to be reset, changing the I to a 9.

5.
```
  05-
 +I60
------
  9-      2^4 Bit Reset
```

We are now ready to process the most significant characters of the augend and addend. During the third "B" status level, the I (MSD of addend) will be placed in D3.

The third "A1" status level will place the MSD of the augend (0) in D2. Since SCAR is set, ICAR will now also be set. (SCAR is then RESET.)

At TP3 of the "A1" STL, since ICAR is set and D3 contains an I, (9 with a 2^4 bit) SCAR and TD9 will be set. We must now incorporate the borrow from the last subtraction by triggering D3 + 1. Before triggering, D3 contained an I $[011001]_{(2)}$; after triggering, D3 will contain a + $[011010]_{(2)}$.

At TP5 of the "A1" STL, since SCAR is set, D3 - 2^1, 2^2 and 2^3 will be reset, changing the character in D3 from a "+" to an "&". At TP6 of the A1 STL, the signal "NZ·D3-2^4" will set the D3OF FF. We will now go into the third "D" status level generating the address 0100. The "0" in this location will become the MSD of the result.

```
6.      05-
       +I60
       ----
MSD of Result →  09-
```

At TP1 of the third A2 STL, the signal "ULS·SCAR·D2OF·D3OF" will set the EACC FF. The condition of EACC will be utilized at the end of the complementation process.

During the third "A2" status level, the processor will realize that it has made a mistake. It had assumed that the augend was the larger quantity and had taken its sign as the sign of the result. However, at this point since D2OF is reset and D3OF is set, the processor realizes that the addend was in fact the larger quantity.

To correct for this error in judgment, the result must be 10's complemented and the sign must be changed. This is indicated by the signal "ULS·D2OF·D3OF" setting the EAC (End Around Condition) FF. The processor must now complement each character of the result (using the difference table) and restore the

complemented answer in HSM.

NOTE: During the third "A2" status level, the signal "NZ·TD9·D2OF" caused the 2^4 bit in the MSD of the result to be set.

7. 05-
+I60
Set 2^4 Bit → &9-

During complementation, the processor has no way of knowing the number of characters in the result since N is now zero. It will check each character it complements looking for a 2^4 bit. When one is found, the processor knows that it is complementing the MSD of the result.

The first status levels used during complementation are X3 and X4. These are used to restore in the A address the location of the LSD of the result.

An "A3" status level is now executed. This will place the LSD of the result (-) in D3, and since ICAR is reset, a zero (0) in D2. The signal "FD·$\overline{DZS}$·D3-2^5" will set PRP correcting the sign of the result. A "D" status level is now executed generating the address 0100. The zero in this location becomes the LSD of the complemented result.

8. & 9 - = Result
0 = Complement

The first "A4" status level will place the zero into HSM.

The second "A3" status level will place the second character of the result (9) into D3 and a zero into D2.

The second "D" status level will generate the address 0109. The A in this location (one with a borrow) will cause SCAR to set.

During the second "A4" status level, since we are not on the last digit ($\overline{LD}$), the 2^4 bit will be reset and a (1) will be placed in HSM.

```
9.      &   9   -  = Result
            1   0  = Complement
```

During the third "A3" status level SCAR will set ICAR and the third character of the result (&) will be placed in D3. Since D3-2^4 = 1, LD will be set (last digit) ICAR being set will place a (9) in D2.

The third "D" status level will generate the address 0190. The (9) in this location will become the MSD of the complement result.

```
10.     &   9   -  =  Result
        9   1   0  =  Complement
```

However, we must first set the 2^4 bit. This will be accomplished during the last "A4" status level by the signal "LD·EACC·$\overline{\text{SCAR}}$".

```
11.     &   9   -  =  Result
        I   1   0  =  Complement
    1   9   1   0  =  Decimal Equivalent
```

The MSD (I) is now stored in HSM, and LD being set, tells the processor that the complementation process is now complete.

5. Arithmetic Error (ARIE)

Listed below are the conditions that will generate an ARIE during the execution of an ADD or Subtract instruction.

1. $(D2\text{-}2^4/D3\text{-}2^4)\cdot(\overline{FD}\cdot\overline{NZ})$ = ARIE

The only locations in an operand where a 2^4 bit is allowed will be the LSD (indirect address) and the MSD (carry). The presence of a 2^4 bit in any character of either operand will generate an ARIE.

```
Example:    1 2 E 0  ARIE
          + 3 4 5 1
          ---------
```

2. $\underline{ULS \cdot A1 \cdot NZ \cdot D2\text{-}2^4 \cdot D2\text{-}2^4 = ARIE}$

Any time a summation operation is in progress ($\overline{ULS}$), and the MSD of the operands are being added (NZ), and both MSD's contain carry bits ($D2\text{-}2^4 \cdot D3\text{-}2^4$) an overflow will occur generating an ARIE.

Example:

ARIE	A	4 2 0
	+ C	6 2 0

3. $\underline{D \cdot \overline{ULS} \cdot ICAR \cdot D3\text{-}2^4 = ARIE}$

ICAR will be set in this condition when "A1·NZ ($D2\text{-}2^4/D3\text{-}2^4$)" indicating that we are processing the MSD's of the operands and either or both contain a carry bit. If on the following "D" status level, while accessing the sum table, a result with a carry bit ($D3\text{-}2^4$) is extracted, an overflow will occur generating an ARIE.

	ICAR	I 2 0 0
ARIE		+4 5 0 0
	$D3\text{-}2^4$	C 7 0 0

4. $\underline{D \cdot \overline{ULS} \cdot NZ \cdot SCAR \cdot ICAR = ARIE}$

This condition could occur while doing a summing operation ($\overline{ULS}$) if the addition of the characters preceding the most significant characters of the two operands resulted in a carry. This carry would have to be incorporated into the addition of the MSD's by triggering D3 + 1. If however, D3 contained a 9 before triggering, it would be triggered to zero and SCAR would be set indicating a carry.

If at this time, the MSD of either operand contains a carry bit (indicated by ICAR being set), an ARIE will be generated (two carries out of the MSD addition).

Example:

	Set ICAR	A 5 0 0
ARIE		0
	Set SCAR TRIG + 1	9 5 0 0
		0 0 0

↑ with a carry

6. Programming Errors

Besides using an N character outside of the N count as described in the DL/DR section, the Add and Subtract instructions have a few combinations that might cause difficulties. All of the problems in Add and Subtract result from the actual characters involved in the arithmetic operation and not from the instruction itself.

As stated in the 301 Programmer's Reference Manual, any character containing a 2^4 bit in any position of the operand other than in the LSD or MSD positions causes an ARIE alarm. Also generating an ARIE is the addition of two MSD's which contain 2^4 bits. Effectively the 2^4 bit in an MSD indicates an overflow or carry. For example, the operand G13 represents 1713 because a G is a seven with a 2^4 bit. Therefore, if two such characters are added (Sum Table only) the result would be 20 or greater which effectively would require a single character with a double carry. The largest number that can be represented by one character is 19 or the letter I. Hence the addition of two MSD's containing 2^4 bits causes an alarm.

If one MSD contains a 2^4 bit and an incoming carry along with the other MSD produces a result greater than 19, an ARIE will occur.
Example:

Add Instruction

Augend = H257 → 18257
Addend = 1923 → +1923
= 2Ø18Ø

An ARIE would be generated during the fourth D status level because 20 cannot be represented by one character.

To eliminate either of the aforementioned problems, one can insert insignificant zeros into the operands before performing the addition.
Example:

18257 Add 5 characters
Ø1923
2Ø18Ø

The 2^5 bit in the least significant digit is recognized as a minus sign. However, the 2^5 bit is ignored in any other character position and no alarms occur as a result of it. The 2^4 bit is recognized and permitted in the LSD position. The computer will assume an indirect address is being used as an operand if a 2^4 bit is found in the LSD, hence a 2^4 bit is always inserted into the LSD of the result if this occurs, even though the operands and result are greater or less than 4 characters.

Example 1:

Add Instruction

58Ø32
4176F

9979H Result (No Alarm)

Example 2:

Add Instruction

2A
34

5E Result (No Alarm)

If, however, the computer must perform complementation when a 2^4 bit exists in the LSD, an incorrect result will be obtained. This is true even in the case of indirect addresses.

Subtract Instruction

328D - Minuend
329Ø - Subtrahend

I99W - Result (Incorrect but no alarm)

Result should be ØØØW.

Note that any resultant address, even if indirect, should not be negative. Therefore, the number of occurrences of an incorrect result such as described should be very few if any.

Since the Add and Subtract instructions use the characters of each operand as part of an address during the D status levels, it is possible to generate a MRPE alarm if illegal characters are used. All characters used in the

operands of an Add or Subtract must have their 2^0 through 2^3 bit configuration between binary 0000 and 1001. In other words, no characters other than the decimal digits, letters of the alphabet, symbols &, minus sign, quotation mark, and virgule (/) are permitted within either operand of a decimal operation.

Example:

Add Instruction

Augend = 64*M

Addend = E32K

The character (*) will generate a MRPE and DPE because the address formed would be 00*2 during the second D status level and no X drive line can be selected. Therefore, nothing is read out of memory and bad parity is detected in the Memory Register as well as the D register.

C. Logical Instructions (Q=OR) (T=AND) (U=EXO) Repeatable

The logical instructions used in the 301, perform Boolean Algebra (AND, OR, EXO) on each information bit ($2^0 \rightarrow 2^5$) of each character in both operands. The proper parity is also generated as a result of the operation.

Examples:

1. AND -

$7_{(10)}$ = 0 000111

$9_{(10)}$ = 1 001001

Result → 0 000001 = $1_{(10)}$

2. OR -

$6_{(10)}$ = 1 000110

$1_{(10)}$ = 0 000001

Result → 0 000111 = $7_{(10)}$

3. EXCLUSIVE OR -

A = 1 010001

I = 0 011001

Result → 0 001000 = $8_{(10)}$

NOTE: Pages VI-8 through VI-13 of the Programmers' Reference Manual give operational outlines and some examples of the "AND", "OR" and "EXO" instructions.

Since the logical instructions do not involve carries, their functions might be questioned. Because the 301 code uses zone bits to specify certain characteristics in characters, the logical instructions serve a very useful purpose in extracting or adding these zone bits without disturbing the information bits. This process is known as "making". For example, to make a quantity negative, all that need be done is to insert a 1 in the 2^5 bit position of the LSD of that quantity. A logical OR instruction could be used in this case. If a decimal 5 were OR'ed with a minus symbol, the result would be the letter N, which is a negative 5.

```
    5 = 000101
    - = 100000
    ----------
Result = 100101 = N = -5
```

To remove a given bit or bits, the Exclusive OR and Logical AND instructions are used. For example, to remove the 2^5 bit from the letter N, thereby producing the number 5, an Exclusive OR could be used.

```
    N = 100101
    - = 100000
    ----------
Result = 000101 = 5
```

1. Instruction Format

Operation Code: Q/T/U
N Character: Number of characters per operand (N count = 0-44)
A Address: LSD location of first operand and result
B Address: LSD location of second operand.

NOTE: If N=0, nothing is done and the next instruction in sequence is executed.

DIRECTION OF OPERATION: Right to Left.

PRI's: The PRI's are set only in the Logical AND instruction, where PRN indicates all resultant bits are 0's and PRP indicates at least one resultant 1 bit.

Example 1:

Q 1 2016 2018

	2015	2016	2017	2018
HSM Before	1010001	1010010	0010011	1010100
		(B)		(D)
HSM After	1010001	0010110	0010011	1010100
		(F)		

In this example, the letter B was logically OR'ed with the letter D. The result is an F.

Example 2:

U 4 1003 1008

	00	01	02	03	04	05	06	07	08
10	8	6	J	K	B	*	U	V	3
				A_i					B_i

HSM Before

	00	01	02	03	04	05	06	07	08
10	M	S	D	J	B	*	U	V	3
					B_f				

HSM After

A_f = 0999

PRACTICE PROBLEMS

Problem No. 1

Execute the following instruction and show final HSM contents. What PRI is set?

T 3 2105 2102

	00	01	02	03	04	05
21	8	6	A	=	G	B

HSM Before

	00	01	02	03	04	05
21						

HSM After

PRI? _____

Problem No. 2

Execute the following instruction and show final HSM contents.

Q 4 2012 2013

	08	09	10	11	12	13	14
20	H	P	7	6	*	@	K

HSM Before

	08	09	10	11	12	13	14
20							

HSM After

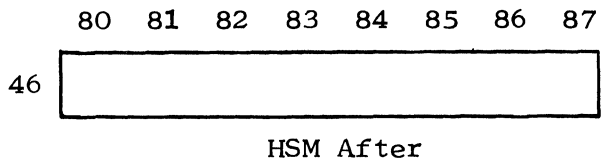
Problem No. 3

Execute the following instructions and show final HSM contents.

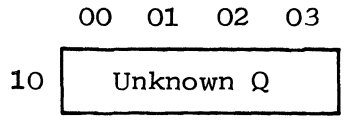
U 2 4685 4682

	80	81	82	83	84	85	86	87
46	1	2	3	4	5	S	T	U

HSM Before



Problem No. 4



There is an unknown quantity (positive or negative) in HSM locations 1000 → 1004. Write a program that will check to see if it is a negative quantity. If it is, remove the negative indicator (2^5 bit) and halt. If it is a positive quantity, just Halt.

Problem No. 5

1000 M = 1000 2000

How could you modify the DL instruction to move 45 characters?

2. Machine Operation

The logical instructions all use the same status levels and operate in much the same manner, the main difference being the logical operation performed on the two operands (AND, OR, EXO).

If an "AND" instruction is being staticized, PRN will initially be set during P5.

The A1 status level is used to access memory bringing out one character and placing it in D2. The N count is also decremented by one.

The B status also accesses memory bringing out the other character and placing it in D3. The logic operation specified is now being performed on the two characters in D2 and D3.

The A2 status level will take the logical result (LR) and store it back into memory at the original "A" address. If an "AND" instruction is being executed, and there is at least one "1" bit in the result, PRP will be set during the A2 status level.

If the N count is not zero at this time, another A1 status level is selected and the entire cycle is repeated on the next two characters.

If however, NZ and NRZ (used in conjunction with the Repeat instruction) are present, a P1 status level will be selected and the next instruction in sequence is executed.

In order to trace through the status flow, utilize the following example:

NOR	N	A	B		O1	O2
T	1	1001	1002	10	9	8

NOTE: See Status Flow on pages 273 and 274.

The instruction will initially be staticized and, being an "AND" instruction, P5•TP2 will set "PRN".

Since we have $\overline{NZ}$, an A1 status level is selected. The A address is gated to the MAR (TP1) and used to address memory. You will note that the A address will not be modified at this time, since the result must be placed back into this location.

A read-type memory cycle is initiated (GEN CL•Permit both) and the original address is gated back to the A register. The N count is then decremented by a (TP3). The D register is reset (TP4) and the character in MR1 (address odd) is gated to Bus 2 (TP45). Bus 2 is then gated to D2 (TP5). TP6•A1 will select a B status level. At this time, the registers contain the following:

NOR	N	A	B	D2	D3
T	0	1001	1002	9	

During the B status level, the B address is gated to the MAR (TP1) and used to address memory. The B address is modified (BA-1). A read-type memory cycle is initiated (GEN·CL Permit Both) and the modified address (1001) is gated back to the B register. The character in MRO (even location) is gated to Bus 3 (TP45). TP5 gates Bus 3 into D3. TP6·B will select an A2 status level. At this time, the registers contain the following:

NOR	N	A	B	D2	D3
T	0	1001	1001	9	8

Since the 8 and 9 in D2 and D3 are being AND'ed, the LR (Logical Result) would be = 8.

The A2 status level will gate the original A address into the MAR (TP1). It will now be modified (BA-1). A write-type memory cycle is now initiated (address add - Permit CO, inhibit C1).

The logical result (8) is gated to bus 2 (TP23) and then bus 2 is gated to MR1 (address add). The A register is reset, and since at least one "1" bit is present in the result (8), PRP is now set (TP4·A2). During the regeneration portion of the memory cycle (TP45), the LR (8) will be written into location 1001. The modified A address is now gated back to the A register (TP5). At TP6, since we have NZ·NRZ, indicating completion of the AND instruction, a P1 status level is selected. The final register contents are as follows:

NOR	N	A	B
T	0	1000	1001

LOGICAL INSTRUCTIONS (Q = OR) (T = AND) (U = EXO)

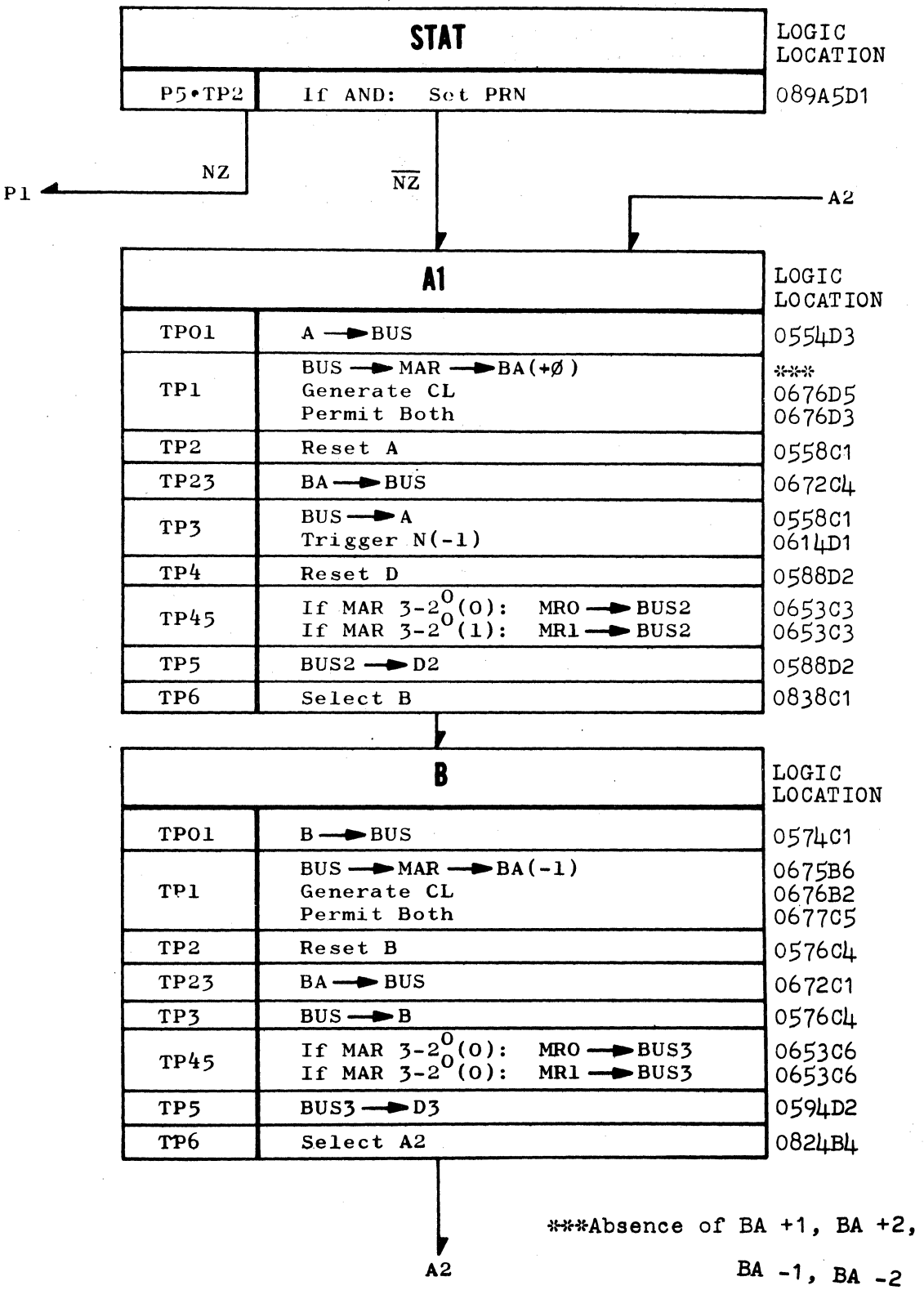


Figure 66 Logical Instruction Status Flow

B

A2		LOGIC LOCATION
TP01	A → BUS	0554D3
TP1	BUS → MAR → BA(-1) Generate CL If MAR $3\text{-}2^{0}_{0}(0)$: Permit C1, Inhibit C0 If MAR $3\text{-}2^{0}_{0}(1)$: Permit C0, Inhibit C1	0675B5 0676D5 0677C2 0677C2
TP23	If OR: D2/D3 → BUS2 If AND: D2·D3 → BUS2 If EXO: D2/D3·(D2·D3) → BUS2	0692D1 0693D2 0693D1
TP3	If MAR $3\text{-}2^{0}_{0}(0)$: BUS2 → MR0 If MAR $3\text{-}2^{0}_{0}(1)$: BUS2 → MR1	0655C1 0655C1
TP4	Reset A If AND·$\overline{\text{COZ}}$: Set PRP	0556C3 089A6D1
TP5	BUS → A	0556C3
TP6	If NZ: Select A1 If NZ NRZ: Select P1 If NZ $\overline{\text{NRA}}$: Select REP1	0827C1 084A7C2 084A7C2

$\overline{\text{NZ}}$ → A1

NZ·NRZ → P1

NZ·$\overline{\text{NRZ}}$ → REP1

ANSWERS TO PRACTICE PROBLEMS

Pgs. IV-227, 228, 229

Problem #1

	50	51	52	53	54	55	56	57
56	6	B	R	8	9	5	2	3

HSM After

PRI PRN

Problem #2

	20	21	22	23	24	25	26	27	28
26	3	0	3	1	3	2	5	6	2

HSM After

PRI PRP

Problem #3

	00	01	02	03	04
20	&	1	3	3	2

	05	06	07	08	09
40	1	4	8	0	9

HSM After

Problem #4

	60	61	62	63	64	65
48	2	5	9	6	K	4

	09	10	11	12	13	14
21	B	4	6	4	4	A

HSM After

PRI PRP

Problem #5 - + & 1012 4641

Problem #6

	87	88	89	90	91	92	93
37	Ø	Ø	Ø	Ø	2	6	7

Problem #7 One Possible Solution

2000	K	Ø	1000	1008
2010	+	4	0215	2099
2020	J	Ø	1000	021E
2030	K	Ø	1010	1018
2040	+	4	0215	2099
2050	J	Ø	1010	021E
2060	+	8	1008	1018
2070	K	Ø	1000	1008
2080	J	@	021E	021E
2090	.	0	0000	0001

Problem #8

The purpose of the program in Problem #8 is to search memory, (beginning at location 1200) for a location containing a zero. If a zero is found, STA will contain the HSM address of the location to the right of the zero. The program is then subtracting a constant of one (1) from the contents of STA to establish the exact location of the zero. An "at the rate of" symbol is then placed in this location. The program works, provided:

1. That at least one zero will always be located somewhere in memory. This is because no provision has been made in the program to take care of the case where there are no zeros in memory.
2. That the zero is never stored in a memory location ending in nine.

Example: Let us assume that HSM location 1299 contains a zero.

The contents of STA after completion of the "Transfer Data by Symbol" instruction would be:

	12	13	14	15
02	1	3	0	0

STA

The "Sub" instruction will subtract the constant "1" in location 1035 from the "0" in location 0215.

```
    0
  - 1
  ---
  - 1 = J
```

Result

The contents of STA after completion of the "Sub" instruction will be:

	12	13	14	15
02	1	3	0	J

STA

The "Symbol to Fill" instruction is then staticized; and during indirect addressing on A and B, the processor will attempt to place the contents of 0215 (J) in A3 and B3 neither of which contain a 2^5 bit. Therefore, after staticizing, (the 2^5 bit having been dropped) A and B will contain:

A•B

1	3	0	1

Bad Parity (1000001)

During the first A2 status level of the "SF" instruction, when the "A" address is gated into the MAR, the processor will stop with a "MAPE".

ANSWERS TO PRACTICE PROBLEMS

Problem No. 1

Pgs. IV-260, 261

	00	01	02	03	04	05
21	8	6	A	Y	F	&

HSM After

PRI PRP

Problem No. 2

	08	09	10	11	12	13	14
20	H	$(57)_8$	$(57)_8$	E_f	*	@	K

NOTE: $(57)_8$ does not correspond to any 301 character but will still be generated as such.

Problem No. 3

	80	81	82	83	84	85	86	87
47	1	2	3	4	7	/	T	U

Problem No. 4

Possible Solution

0300	T	1	0332	1003
0310	W	1	0320	0330
0320	U	1	1003	0336
0330	.	0	-000	-000
			↑ Minus	↑ Minus

Problem No. 5

By ORing the N char (=) at 1001 with a "1", generating an octal 77 (77_8). This would increase the N count by 1, from 44 to 45.

SECTION V

INPUT/OUTPUT INSTRUCTIONS

A. INPUT-OUTPUT INSTRUCTIONS, INTRODUCTION

The largest group of 301 instructions is the Input-Output group. Twenty-one instructions perform all operations involved with the peripheral equipment. All input-output equipment is tied in directly to the computer and must therefore receive commands and information from the computer. The following list comprises the Input-Output group.

Op. Code		Instruction
0	-	CARD READ NORMAL
1	-	CARD READ SIMULTANEOUS
2	-	CARD PUNCH NORMAL
3	-	CARD PUNCH SIMULTANEOUS
4	-	TAPE READ FORWARD NORMAL
5	-	TAPE READ FORWARD SIMULTANEOUS
6	-	TAPE READ REVERSE NORMAL
7	-	TAPE READ REVERSE SIMULTANEOUS
8	-	TAPE WRITE NORMAL
9	-	TAPE WRITE SIMULTANEOUS
;	-	REWIND TO BTC
B	-	PRINT AND PAPER ADVANCE NORMAL
C	-	PRINT AND PAPER ADVANCE SIMULTANEOUS
D	-	BAND SELECT NORMAL
D	-	TRACK SELECT
E	-	BAND SELECT RECORD FILE MODE
F	-	BLOCK READ FROM RECORD NORMAL
F	-	SECTOR READ DISC NORMAL
G	-	BLOCK READ FROM RECORD SIMULTANEOUS
G	-	SECTOR READ DISC SIMULTANEOUS
H	-	BLOCK WRITE TO RECORD NORMAL
H	-	SECTOR WRITE DISC NORMAL
I	-	BLOCK WRITE TO RECORD SIMULTANEOUS
I	-	SECTOR WRITE DISC SIMULTANEOUS
*	-	RECORD FILE MODE READ
%	-	RECORD FILE MODE WRITE

B. CARD READ NORMAL (Ø) (CRN/BCRN)

This instruction reads the contents of one punched EAM card in the 323 Card Reader and places the information into HSM. The word "Normal" indicates the mode in which the instruction is executed and will be described later in greater detail.

There are two types of code which can be punched in a 301 EAM (Electronic Accounting Machine) card. The more common of the two is the 301 Card Code which uses one, two or three punches in a column to represent one 301 character.

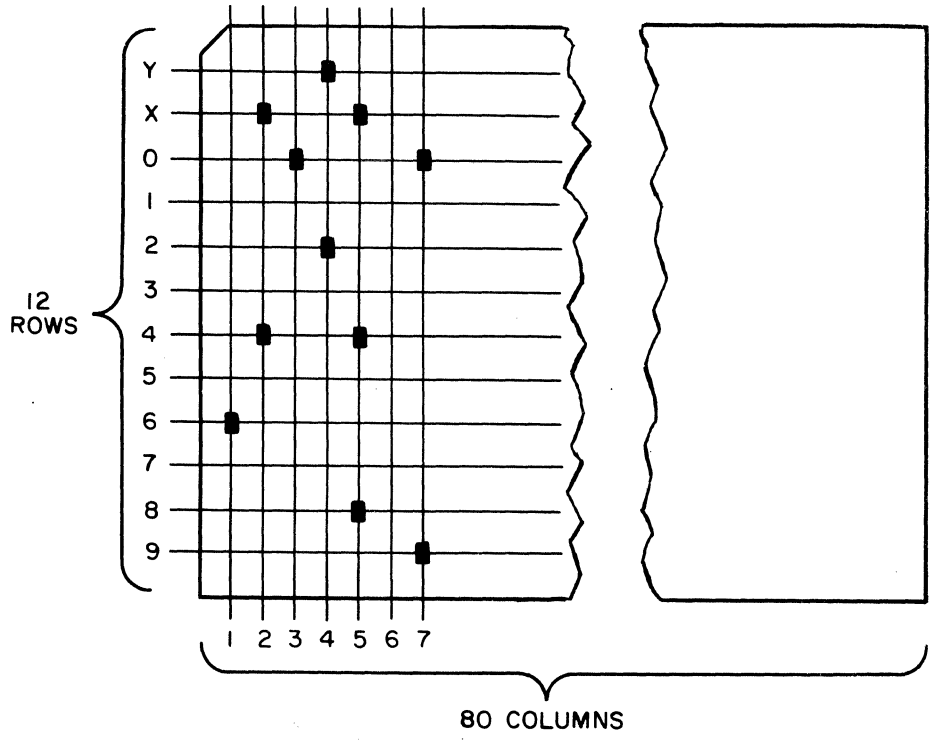


Figure 67 EAM Card Format (301 Card Code)

1. 301 Card Code

Figure 67 illustrates a portion of a card which has been punched in 301 Card Code. Note that there are 12 rows and 80 columns. Each column contains one 301 character, hence a maximum of 80 characters can be represented on

one card in 301 Card Code. The rows Y, X and 0 are special zone punches. No zone punch corresponds to Group I in 301 Machine Code (2^5 bit = 0, 2^4 bit = 0). A Y zone punch corresponds to Group II in 301 Machine Code (2^5 bit = 0, 2^4 bit = 1), an X zone punch corresponds to Group III in Machine Code (2^5 bit = 1, 2^4 bit = 0) and a 0 zone punch is equivalent to a character in Group IV (2^5 bit = 1, 2^4 bit = 1). All the decimal digits are represented by a single punch in the proper row 0 through 9. All the letters of the alphabet have a zone punch (Y, X or 0) in conjunction with a numeric punch. The special control symbols for the most part are represented by two or three punches. They are a combination of row 8 and some other numeric punch whose total will be the decimal equivalent of the binary bits 2^0 - 2^3, and a zone punch if necessary. For example, the 301 Machine Code for a period is 011011 (excluding parity). The zone bit 2^4 being a one, indicates the need for a Y zone punch. The remaining bits (2^0 - 2^3) add up to a decimal 11 (8 + 3). Thus, a period would be represented by three punches in a column - a punch in each of the rows Y, 3 and 8.

In Figure 67, columns 1 through 7 contain 6, M, Ø, B, *, SP and Z, respectively.

Note that there are a few violations to the rule of zone punches, namely that a punch in row 0, alone, is the character zero which has no zone bits and the character quotation marks is a punch in rows X and 0. Also, the character +0 is a punch in row Y and a punch in row 0. A fourth exception to the rules presented is the space or underline character. This character is represented by no punches at all in a card column, although in Machine Code the character is a binary coded decimal 10 (001010). See column 6 of Figure 67. Any more than three punches in a column or any combination of punches which do not agree with the 301 Card Column causes a multipunch alarm.

2. Straight Binary Machine Code

The second type of code used on 301 EAM cards is the straight binary Machine Code. Since there are 12 rows, two six-bit characters can be represented in each of the 80 columns. Parity is not recorded and rows 9 through 4 are equivalent to the 2^5 through 2^0 bit, respectively, as are rows 3 through Y.

In Figure 68, the characters punched in the "A" half of the card columns 1 through 4 are N, SP, B and $, respectively. The characters punched in the "B" half of the card columns 1 through 4 are 1, G, Ø and &, respectively. There are no exceptions in this type of recording. Wherever a hole is punched, a one bit is represented and no hole represents a zero. The binary Machine Code of six bits per character is represented exactly on the card. This type of code provides twice as many characters on a card (160) as the other type (Card Code).

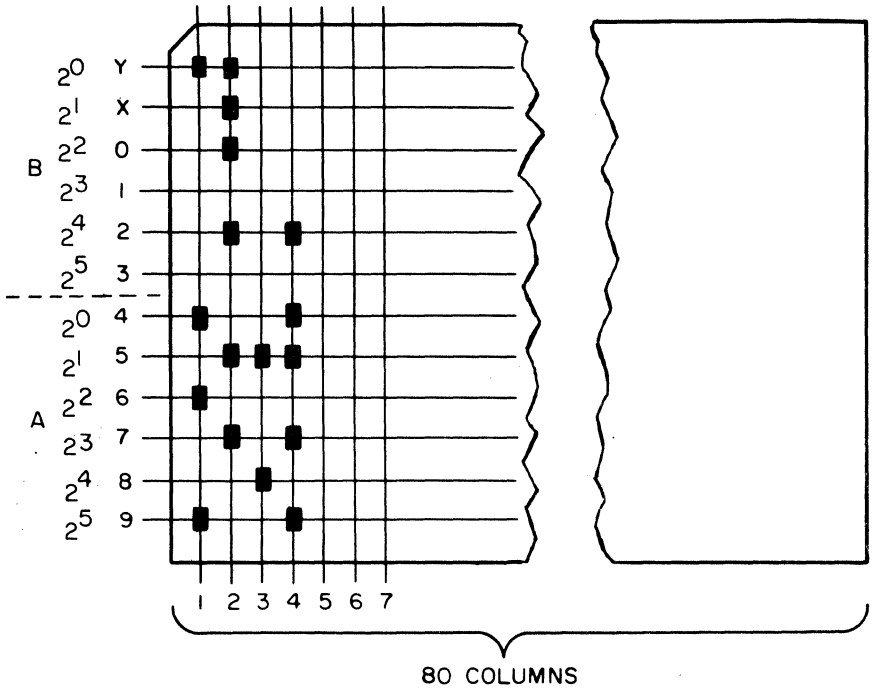


Figure 68 EAM Card Format, Straight Binary Machine Code

3. The 323 Card Reader and Codes

With the 323 Card Reader, the type of code being used on the cards read is indicated by the BCT (Bypass Card Translate) console button. If 301 Card Code is used, translation must be performed and the BCT button must not be set. If binary Machine Code is used, the BCT button must be set in order

to bypass card translation and permit each row to represent a bit position.

Due to mechanical restrictions of the 323 Card Reader, the N character of the Card Read instruction is a code designating how many cards will be fed and read. Only one card is read for each instruction. However, if the N character is 4, it denotes continuous reading and cards will be fed at a specific rate. At the maximum reading rate of 600 cards per minute, there are three cards always moving on the transport. A Card Read instruction with a N character of 4 must be executed for every Card Read until the last two cards are about to be read. Card Read instructions with N = 2 and N = 1 are then executed to read the last two cards without feeding any additional cards. Obviously, this means that in order to use the 323 Card Reader at the maximum reading rate, either the exact number of cards to be read must be known or a special character must exist on the third from last card to indicate when the termination sequence should begin. If it is desired to read cards at a slower rate (200 cards per minute), a CRN instruction could be executed with an N character of 1 as many times as desired. This technique reads the same amount of information but at one third the rate. However, the 323 Reader was not designed to be used as a demand feed type reader in this manner and thus should be programmed at the maximum rate or by using alternate feed.

Alternate feed is a means of picking every other card. This uses a special N character of M. Instead of three cards moving on the transport only two cards are moving. The reading rate of course is halved to 300 cards per minute, but the advantage is that processing time is gained between cards. At 600 cards a minute, there are 100 milliseconds (MSEC) between cards. Eighty msecs are consumed by the reading process leaving 20 msecs for processing. A new card read instruction must come up before the 20 msecs have elapsed or information will be lost resulting in an alarm. The use of alternate feed provides an additional 100 msecs before a new instruction must be executed. The termination sequence then requires execution of one instruction with an N character of 8 to read the last card without feeding any additional cards.

The Card Read Normal instruction does not go through STA and is not repeatable.

Op. Code	N	A	B
Ø (ZERO)	Code for number of cards to be read (see below	Leftmost location to receive first character being read.	Ignored

N = 4 continuous reading at 600 cards per minute

N = 2 used in terminating 600 card a minute sequence

N = 1 used in terminating 600 card a minute sequence or single card feed (200 cards per minute).

N = M alternate card reading at 300 cards per minute.

N = 8 used in terminating 300 card a minute sequence.

Once a terminating sequence has begun, it must be completed before a new sequence is started.

$A_f = A_i + 80$ If BCT was not set or

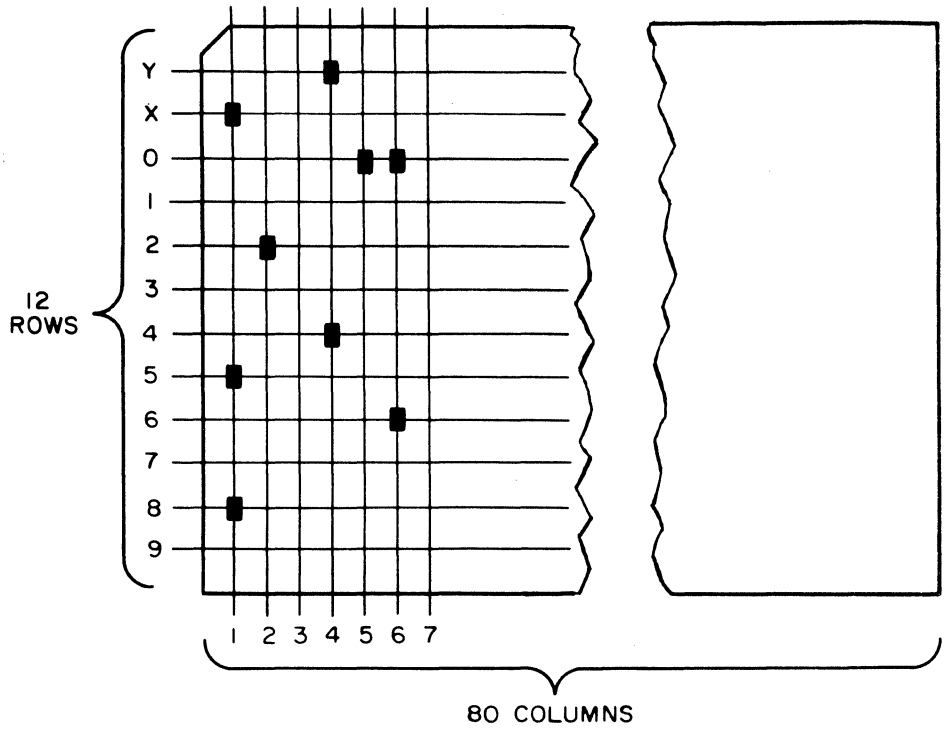
$A_i + 160$ If BCT was set.

$B_f = A_i$ If BCT was not set or

$A_i + 80$ If BCT was set.

Since a blank column on a card produces a space or underline in memory, there will always be 80 memory locations consumed per card if BCT is not set and 160 memory locations if BCT is set. When cards are read with BCT, set, information in rows 9 through 4 is placed in the first 80 memory locations while information in rows 3 through Y will be placed in memory beginning at $A_i + 80$. See Figure 70.

Example: Ø 1 3000 0000



Columns 7 through 80 are blank

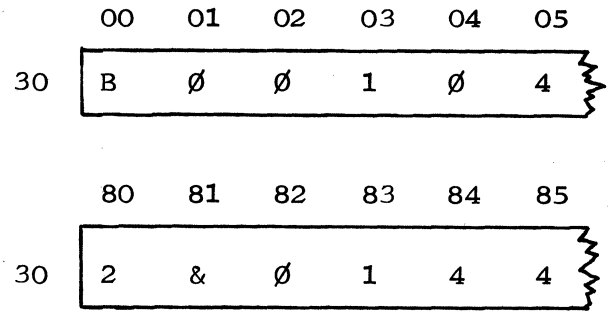
Figure 70 EAM Card, 323 Card Reader Code

If BCT were not set when the above instruction was executed, the following would exist in memory.

	00	01	02	03	04	05
30	E_D	2	SP	D	Ø	W

Locations 3006 through 3079 would contain spaces (SP).

If BCT were set when the above operation was carried out, HSM would contain:



Locations 3006 through 3079 would contain zeros.

Locations 3086 through 3159 would contain zeros.

Example:

1000	Ø	4	2000	0080	
1010	+	4	1005	1009	
1020	X	0	1039	1000	
1030	Ø	2	100E	0022	Assume BCT not set.
1040	V	2	0000	0000	
1050	Ø	1	021E	0000	
1060	.	0	0000	0000	

The above program would read in 25 cards at the maximum rate of 600 cards a minute. The contents of the entire 25 cards would be strung end to end in HSM beginning at 2000 and ending at 3999.

Example:

1500	Ø	M	3000	0160	
1510	+	4	1505	1509	
1520	X	0	1539	1500	Assume BCT is set.
1530	Ø	8	150E	0008	
1540	.	0	0000	0000	

The above program would read in a total of 10 cards using alternate feed or the 300 card a minute rate. Information on each card would consume 160 locations since BCT was set and the data would be strung end to end in HSM between 3000 and 4599.

C. CARD READ SIMULTANEOUS (1) (CRS/BCRS)

Three modes exist in the 301 System-Normal, Simultaneous and Record File. Each mode has specific instructions which can be executed only in that mode and the latter two modes are optional equipment. A complete set of registers exists for each mode:

NOR	N	A	B	Normal
SOR	M	S	T	Simultaneous
FOR	L	U	V	Record File

All instructions are staticized in the Normal mode and from there are transferred accordingly to the mode for which they were written. The Normal mode is all inclusive. That is, the Normal mode can carry out any internal operation such as Add or Subtract, Data Handling or Decision and Control, and it can control any type of peripheral gear. The Simultaneous mode, on the other hand, can only carry out Input-Output instructions. It cannot perform internal operations. The Record File Mode is further restricted to operating four Record Files which are under its exclusive control. The File mode can not operate any other device nor can it perform any internal operations.

Once an instruction has been transferred to the Simultaneous or Record File mode, the Normal mode is free to staticize the next instruction in the program. Thus, if programmed correctly, three separate instructions could be executed at the same time, time-sharing memory. In using this programming tool, one does not need to worry about the type of operations being carried out. The only restriction is that the same device cannot be controlled by two or three different instructions at the same time. If the device is busy or the mode is busy when the instruction is staticized, the computer will automatically hold off until they are free. Another point concerning simultaneity is that the programmer must take care not to attempt storing a final register setting before the completion of instruction execution in the Simultaneous or Record File mode.

The Card Read Simultaneous instruction has the exact same format as the CRN except that the Operation Code is a 1 and the instruction is carried out in the Simultaneous Mode registers.

$S_f = A_i + 80$ if BCT is not set

$A_i + 160$ is BCT is set

$T_f = A_i$ if BCT is not set

$A_i + 80$ is BCT is set

Page VIII-8 of the Programmers' Reference Manual covers the CRS instruction for the 323 reader.

D. CARD READ NORMAL (Ø) (CRN)

This instruction will read the contents of one EAM card on the 330 Card Reader-Punch into High Speed Memory. The same 301 system cannot contain a 323 Card Reader and a 330 Card Reader-Punch, therefore, no conflict arises from the use of the same operation code for both Card Read instructions. The 330 Card Read unit is a demand feed type mechanism which can read at a maximum rate of 800 cards a minute. Because of its technique in reading, it is not necessary to program the card rate by the N character as on the 323 operation. There is another instruction (IOC), however, which is used in conjunction with the CRN in order to provide more processing time between cards. This will be explained shortly.

The 330 CRN instruction also permits the N character to select the binary mode or translate mode (301 Card Code) for reading. (K and 1, respectively). The BCT console button has no effect on this instruction.

Op. Code	N	A	B
Ø	K - Binary Mode 1 - Translate Mode	Left most location to receive the first character read.	Ignored

$A_f = A_i + 80$ if translate mode is used.

$A_i + 160$ if binary mode is used.

$B_f = A_i$ if translate mode is used.

$A_i + 80$ if binary mode is used.

Example: Ø K 4207 0000

Assume punches exist in rows 5 and Ø of column 1; row 9 of column 2; rows 8 and 3 of column 3; and rows 7, 6, 2 and Y of column 4. The remaining columns are blank.

Memory would contain the following after reading the card:

	07	08	09	10
42	2	-	&	@

	87	88	89	90
42	4	Ø	-	A

Locations 4211 through 4286 and 4291 through 4366 would contain zeros.

A_f = 4367 B_f = 4287

When the 330 reader unit is operating at maximum speed, only 10 milliseconds are available between cards for processing. During this time, another Card Read instruction (or an IOC instruction) must be executed if it is desired to maintain the 800 card a minute rate. If a Card Read instruction does not come up within the 10 millisecond period, an additional 25 msecs must elapse before the reader can accept the clutch command. (See timing chart) The clutch command must precede the process feed signal which occurs every 25 msecs. Once a clutch command has been accepted, no more card feeds can occur until the read has been completed.

The Read instruction itself, is not really needed until the card has moved under the brushes (about 21 ms. after clutch command is accepted). Therefore, if a clutch command alone could be sent at the proper time, an additional 21 msecs could be gained before another Read instruction would be needed. This is one of the functions of the Input-Output Control (IOC) instruction which is known as Read Release.

The Input-Output Control instruction is covered on page VIII-19 of the Programmers' Reference manual and has three primary functions. One of the

functions is Read Release which releases the computer of the obligation for a second read instruction to be staticized within 10 msecs after the first has finished. The IOC instruction can provide the clutch command necessary to feed a new card and thus give an additional 21 msecs or a total of 31 msecs processing time between read instructions. (See timing chart).

A second function of the IOC instruction is punch release which is the same concept as read release except that it is involved with the punch unit of the 330. And, finally, the third function is stacker select which enables the programmer to select one of two output hoppers for either the read unit or the punch unit.

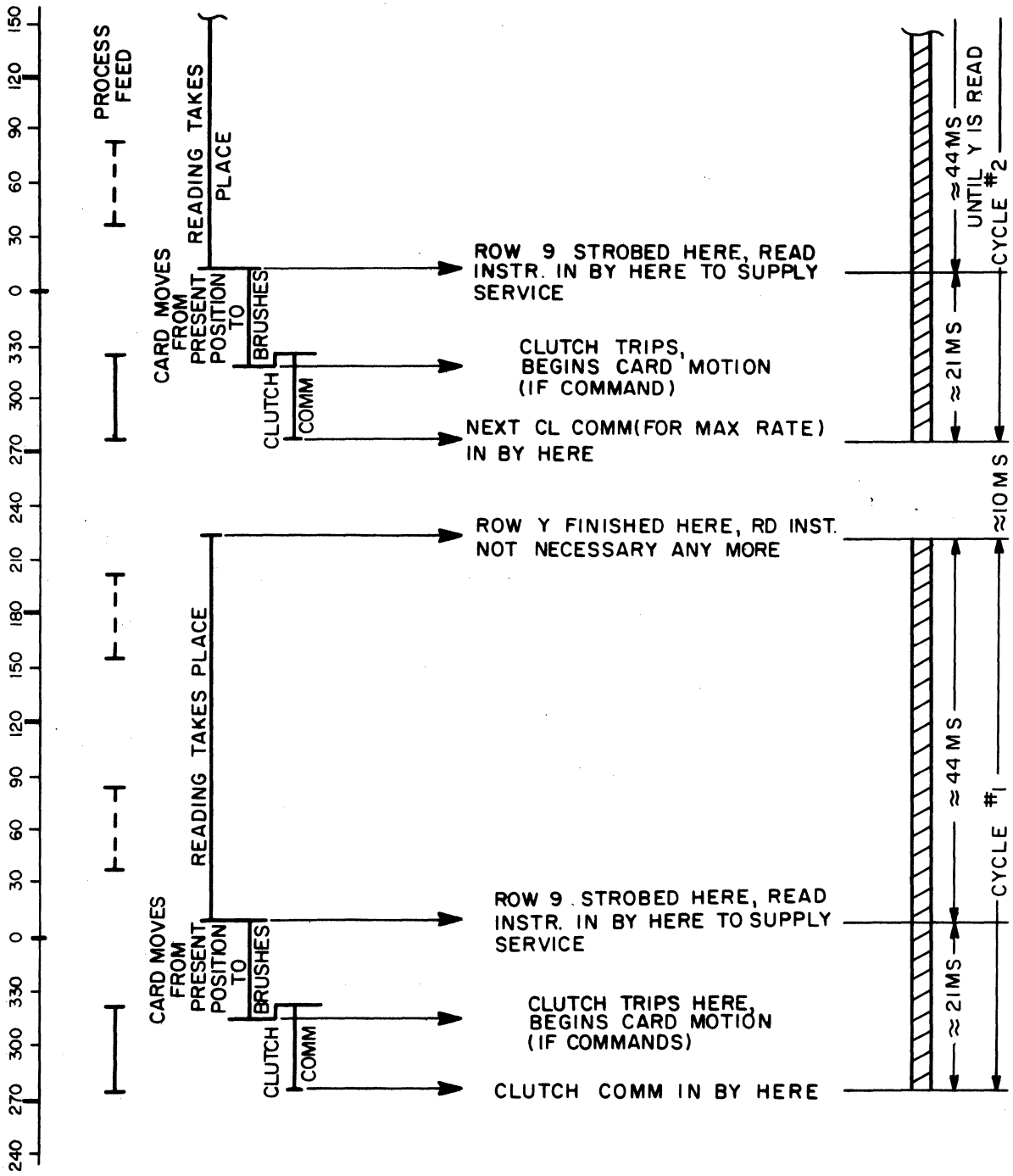


Figure 71 Timing Chart (330 Reader Unit)

There are five output hoppers on the 330 Card Reader-Punch as shown below.

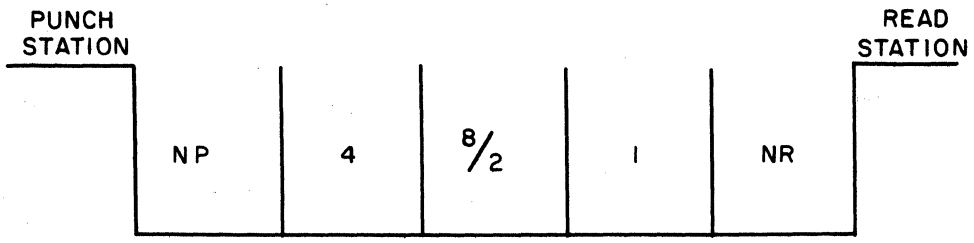


Figure 72 Front View of Output Hoppers

The NR (Normal Read) and 1 hoppers are used only with the reader and the NP (Normal Punch) and 4 hoppers are used only with the punch unit. The center hopper (8/2) can be used by either the reader or the punch. If an IOC instruction selecting an output hopper does not follow the card read operation, then the card will automatically go to the NR hopper. Unfortunately, if a reject card occurs during reading, it will also end up in the NR hopper. Thus, if the programmer desires to keep the reject cards separate from the rest, he must follow every card read with an IOC instruction and select hopper 1 or 2.

A similar thing happens when the punch is used. If an IOC instruction does not follow the card punch operation, the card will end up in the NP hopper. Reject cards, likewise, are sent to NP during punching. Therefore, a programmer will normally follow a card read or card punch instruction with an IOC instruction to select the proper output hopper and also to exercise the option of read or punch release.

The IOC instruction format is:

Op. Code	N	A	B
;	(	$A_0A_1A_2$ Ignored A_3 designates function (See Below)	Ignored

A_3 Character	Function
1	Select Reader Stacker No. 1
2	Select Reader Stacker No. 2
3	Select Punch Stacker No. 4
4	Select Punch Stacker No. 8
5	Read Release
6	Punch Release
7	Read Release and Select Stacker No. 1
8	Read Release and Select Stacker No. 2
9	Punch Release and Select Stacker No. 4
SP	Punch Release and Select Stacker No. 8

The IOC instruction only needs to be initiated and the Processor becomes free to bring out a new instruction from the program.

$$A_f = A_i$$

$$A_f = B_i$$

Example:

```
Ø   1   2000   0000
;   (   0007   0000
```

The prior combination would read in one card from the 330 Card Reader-Punch unit in the translate mode and locations 2000 -- 2079 would receive the data which existed on the card. As soon as the read finished, the IOC instruction would request a new card (read release) and permit the Processor to continue on with the program, allowing about 31 msecs before a second read instruction was necessary.

NOTE: - If another read does not come up in time, a CIG alarm will occur.

The IOC instruction would also select output hopper number 1, where the card which was read would be deposited.

E. CARD READ SIMULTANEOUS (1) (CRS)

This instruction has the exact same format as the CRN operation for the 330 Card Reader-Punch except for the operation code and the fact that it is executed in the Simultaneous Mode. Page VIII-10 of the Programmers' Reference Manual covers the CRS instruction.

$S_f = A_i + 80$ (translate mode) or $A_i + 160$ (binary mode)

$T_f = A_i$ (translate mode) or $A_i + 80$ (binary mode)

F. CARD PUNCH NORMAL (2) (CPN)

This instruction enables the 334 Card Punch to punch 80 column cards from information contained in memory between the A and B addresses. The information will be punched exclusively in 301 Card Code and will handle as many cards as are necessary at 80 characters per card. The CPN instruction operates from left to right, is not repeatable and does not go through STA. See page VIII-12 of the Programmers' Reference Manual.

Op. Code	N	A	B
2	Must be zero	Leftmost Character to be punched	Rightmost Character to be punched

A_f = One location to the right of the last character punched or $B_i + 1$.

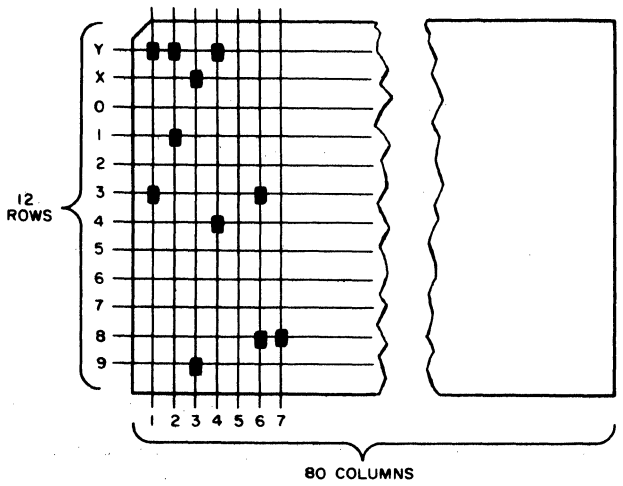
$B_f = B_i$

Example 1:

```
    2   Ø   1000   1006
```

	00	01	02	03	04	05	06	07
10	C	A	R	D	SP	#	8	3

One card would be punched as follows:



Columns 8 through 80 would be blank

A_f = 1007

B_f = 1006

Example 2:

```
    2   Ø   6320   6492
```

The above instruction would punch out two full cards and the first 13 columns of a third card with information which existed between locations 6320 and 6492. A_f = 6493. B_f = 6492.

Due to the mechanics of the 334 Card Punch, the last card punched is not read checked nor placed in the output hopper until two additional "dummy" card punch instructions are executed. These two CPN instructions should both address a single "space" character with the A and B addresses equal.

Example 3:

	78	79	80	81	82	83
14	4	8	2	1	5	SP

2	Ø	1478	1482
2	Ø	1483	1483
2	Ø	1483	1483

The above instructions would punch one card, read check it and move it to the output hopper.

G. CARD PUNCH SIMULTANEOUS (3) (CPS)

This instruction has the same format as CPN except the operation code is a 3 and it is carried out in the Simultaneous Mode. The N character must be zero to properly address the 334 Card Punch. Page VII-16 of the Programmers' Reference Manual covers the CPS instruction for the 334 Card Punch.

H. CARD PUNCH NORMAL (2) (CPN)

This instruction permits the Punch Unit of the 330 Card Reader-Punch to punch 80 column cards according to the information in memory between the A and B addresses. The N Character will designate whether punching will take place in the Binary mode or the Translate mode. The operation is not repeatable and does not go through STA. Page VIII-15 of the Programmers' Reference Manual covers the Card Punch Normal instruction for the 330 Card Reader-Punch. The same 301 system cannot contain a 330 Card Reader-Punch and a 334 Card Punch.

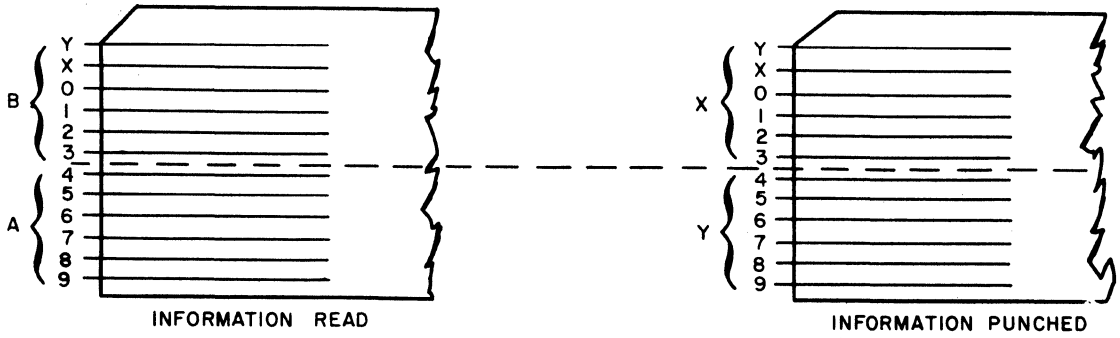
Op. Code	N	A	B
2	& - Binary Mode Ø - Translate Mode	Leftmost character to be punched	Rightmost character to be punched

A_f = One location to right of last character punched or $B_i = 1$

$B_f = B_i$

When using the Binary mode of punching, the characters from A_i to $A_i + 79$

are punched in rows Y through 3 (Y = 2^0, 3 = 2^5) while the characters in memory between A_i + 80 and A_i + 159 are punched in rows 4 through 9 (4 = 2^0, 9 = 2^5). Note that this is directly opposite to the Binary Read operation. Hence, if a card were to be duplicated using the Binary mode, the information would be reversed.

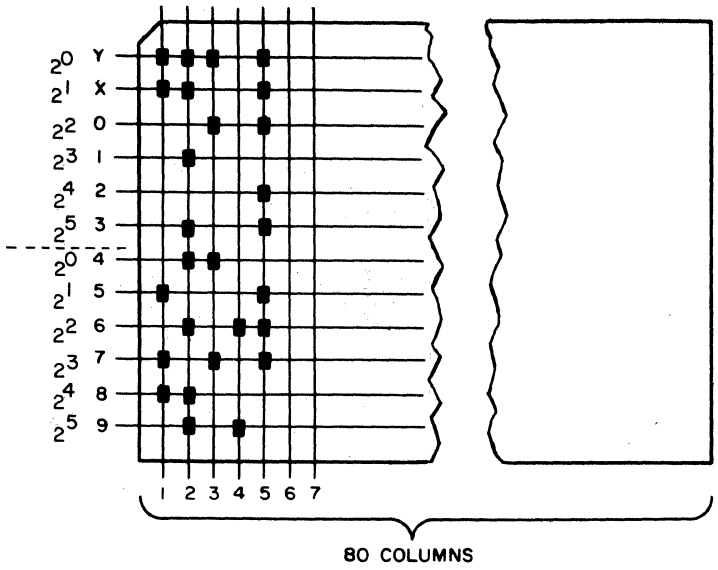


Data read from the A half of the card on the left would end up punched on the X half of the card on the right. Data read from the B half of the card on the left would end up on the Y half of the card on the right. The programmer must also use a multiple of 160 characters when punching in the Binary mode.

Example 1: 2 & 0700 0859

HSM Before and After

Locations 0705 to 0779 and 0785 to 0859 contain zeros. One card would be punched on the 330 Punch Unit as follows:



Columns 6 through 80 would be blank

A_f = 0860

B_f = 0859

Example 2:

Eight full cards and column 1 of a ninth card would be punched in the translate mode according to information in memory between 2100 and 2740. Each Card Punch instruction which uses the 330 Card Reader-Punch should be followed by an IOC instruction (see page VIII-19 of Programmers' Reference Manual) to select the desired output hopper as well as give the optional punch release. If the output hopper is not selected, the card will be placed in the NP hopper along with any reject cards.

To move the last card to the output hopper (and also read check the card) one additional "dummy" punch instruction must be written. This instruction should have the A and B addresses equal with both addressing a non-punchable character such as a space (using the translate mode).

Example 3:

```
2   Ø   3400   3479
;   (   0003   0000
2   Ø   3416   3416        (Assume location 3416
                            contains a space.)
```

The above group of instructions will punch one card in the Translate Mode, read-check it, and place the card in Output Hopper Number 4.

When the 330 punch is operating at maximum speed, the punching rate is 250 cards per minute. This gives approximately 22.5 msecs of processing time between cards. If the punch release option is used, an additional 37 milliseconds of processing time could be gained.

NOTE: If more than one card is punched by the same CPN instruction, the punch release feature and stacker select option only affects the last card of the series.

I. CARD PUNCH SIMULTANEOUS (3) (CPS)

This instruction has the same format as the CPN operation which uses the 330 Card Reader Punch except that the operation code is a 3 and the instruction is carried out in the Simultaneous Mode. Page VIII-18 of the Programmers' Reference Manual covers the CPS instruction.

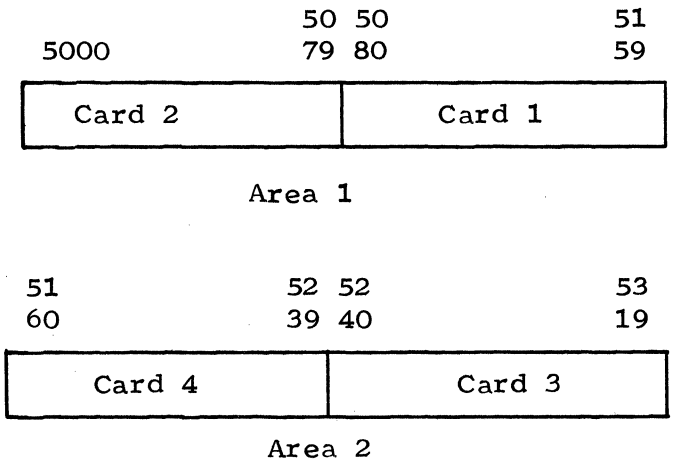
Example Program

```
                                          Remarks
1000   Ø   1   5080   0000     Read-in area 1
1010   ;   (   0001   0000     Select read hopper 1
1020   Ø   1   5000   0000     Read-in area 1
1030   ;   (   0001   0000     Select read hopper 1
1040   W   4   1040   1040     Sense Simultaneous Mode
1050   ;   (   0003   0000     Select punch hopper 4
1060   3   &   5000   5159     Punch out area 1
1070   Ø   1   5240   0000     Read-in area 2
1080   ;   (   0001   0000     Select read hopper 1
```

					Remarks
1090	Ø	1	5160	0000	Read-in area 2
1100	;	(	0001	0000	Select read hopper 1
1110	W	4	1110	1110	Sense Simultaneous Mode
1120	;	(	0003	0000	Select punch hopper 4
1130	3	&	5160	5319	Punch out area 2
1140	X	0	1175	1000	Tally back to read-in 1
1150	2	Ø	1171	1171	Move last card out
1160	;	(	0003	0000	Select punch hopper 4
1170	•	SP	0004	0000	Halt

The above program places the contents of 20 cards (which have been punched in 301 Card Code) on 10 cards in Binary code. To increase the total number of cards processed, one only need change the tally quantity at 1165. (Note for each single increment of this tally quantity, 4 additional cards will be read and processed).

The program also exemplifies the use of the Simultaneous Mode by reading two cards in the Normal Mode and punching one in the Simultaneous. The reader will operate at about 500 cards a minute and the punch at 250 cards a minute. Note that two separate read-in areas are used. While the punch is punching the contents of two previous cards which have been read, the reader reads two additional cards into a different area. (See timing chart on page 7-23). The read-in areas are:



Because the Binary Mode card format is reversed for read and punch operations, the first card read is placed in the last 80 memory locations of a given read-in area while the second card read goes into the first 80 memory locations. When the 160 locations are punched, the first 80 locations (Card No. 2) will end up in rows Y through 3 while the remaining 80 locations (Card No. 1) are punched in rows 4 through 9. The end result is that if the binary punched cards are read at a later date, the information comes into memory in the same sequence as that of successive card reads from the original 20 cards.

It should be pointed out that the cards being read are directed to Hopper Number 1 and the cards being punched are directed to Hopper Number 4. The CTC sense of the Simultaneous Mode is to determine when the CPS instruction has finished and therefore when to select the punch output hopper. The initial execution of the CTC and IOC instructions at 1040 and 1050 have no function - they are only useful for subsequent times around.

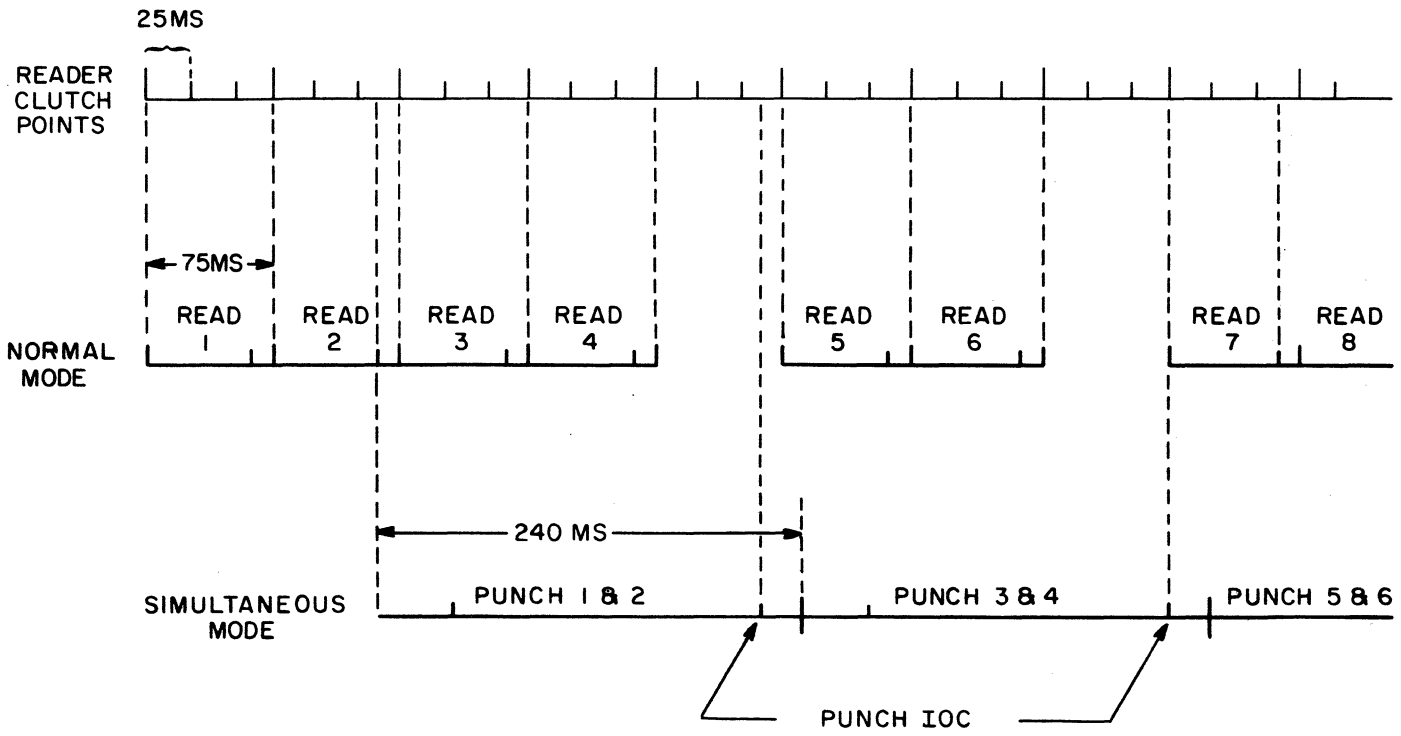


Figure 73 Timing Chart for Example Program

J. REWIND TO BTC (;) (RWD)

This instruction causes a specified magnetic tape to start rewinding. Once the instruction is executed, the rewinding becomes independent of the computer, and another instruction can be executed. See page IX-13 of Programmers' Reference Manual.

Op. Code	N	A	B
;	Identification of Tape Station A-F, 1-6, J or N, L or P	Ignored	Ignored

K. TAPE READ FORWARD NORMAL (4) (RFN) Repeatable

This instruction brings a series of characters, one at a time, from magnetic tape or punched paper tape into the HSM. Transfer from tape begins with the first character following a gap and ends when the next gap is sensed or the specified HSM area is filled. The instruction operates from left to right and goes through STA. The PRI's are set according to the following:

PRP: The A and B Registers are equal before a gap has been found on tape.

PRN: A gap has been found on tape and the A and B Registers are not equal.

PRZ: At the time a gap has been found on tape, the A and B registers are equal.

NOTE: If an EF or ED alone is read from magnetic tape, the ED/EF Normal Indicator is set.

A_f = One location to the right of the last character read into HSM.

$B_f = B_i$

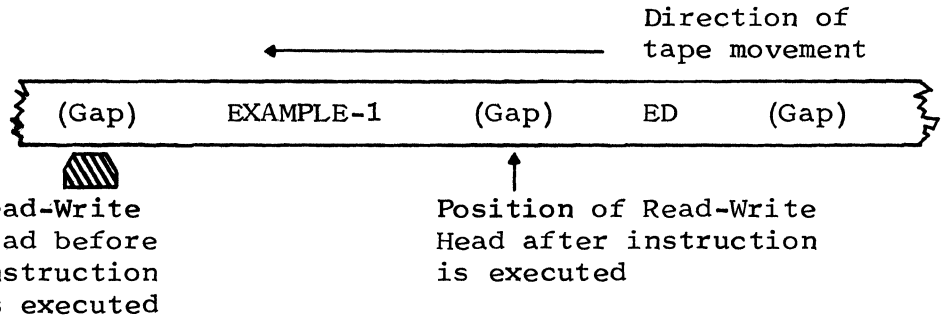
Op. Code	N	A	B
4	Identification Character of Device (see below)	HSM Address which will Receive First Character from Tape	HSM Address which will Receive Last Character from Tape

N	DEVICE
1, 2, 3, 4, 5, 6	Magnetic Tape
A, B, C, D, E, F	Magnetic Tape
J	33KC Adapter (1st unit)
N	33KC Adapter (2nd unit)
8	Paper Tape Reader
L	66KC Adapter (1st Unit)
P	66KC Adapter (2nd Unit)

See page IX-3 of Programmers' Reference Manual.

Example: 4 2 3016 3023

Tape On Tape Station 2



	14	15	16	17	18	19	20	21	22	23	24
30	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

HSM Before

	14	15	16	17	18	19	20	21	22	23	24
30	Ø	Ø	E	X	A	M	P	L	E	-	Ø

HSM After

PRP would be set and A_f would be 3024.

L. TAPE READ FORWARD SIMULTANEOUS (5) (RFS)

This instruction follows the same format as the RFN, except that the Operation Code is a 5 and the instruction is executed in the Simultaneous Mode. The RFS instruction does not go through STA and does not set the PRI's. If an ED or EF is read-in along from magnetic tape, the ED/EF simultaneous indicator is set.

See page IX-6 of Programmers' Reference Manual.

M. TAPE READ REVERSE NORMAL (6) (RRN) Repeatable

This instruction transfers a series of consecutive characters from magnetic tape or paper tape into the HSM. Transfer from tape begins with the first character following a gap, and ends when the next gap is sensed or the specified HSM area is filled. Though the tape moves in reverse, the characters will be placed in their proper relative positions within HSM. The instruction operates from right to left and goes through STA. The PRI's are also set as follows:

PRP: A Register equals B Register before gap is sensed on tape.

PRN: Gap is found on tape before A and B Registers are equal.

PRZ: A Register equalled B Register at time gap was sensed on tape.

An ED or EF alone read in from magnetic tape sets the ED/EF Normal Indicator.

A_f = Address of location one to the left of the last character read in.

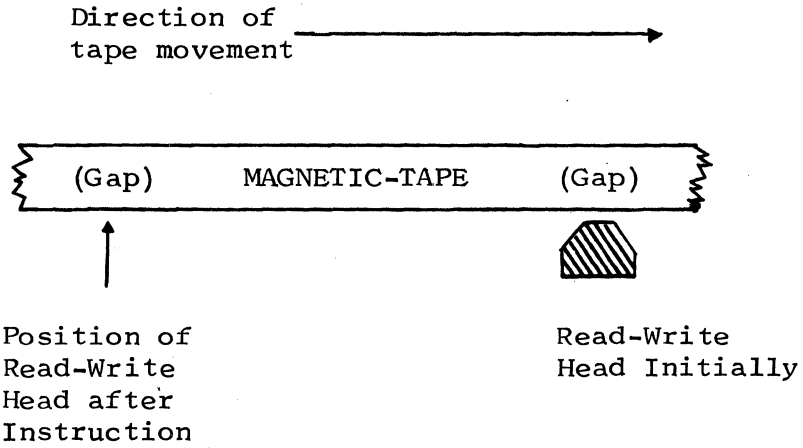
$B_f = B_i$

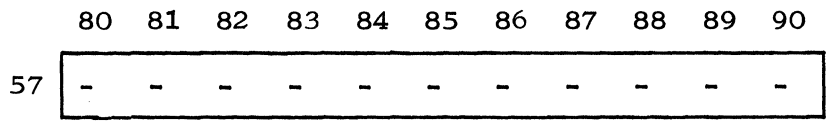
See page IX-7 of Programmers' Reference Manual.

Op. Code	N	A	B
6	Identification Character of Tape Station or Paper Tape Reader	Address which will Receive First Character from Tape	Address which will Receive Last Character from Tape

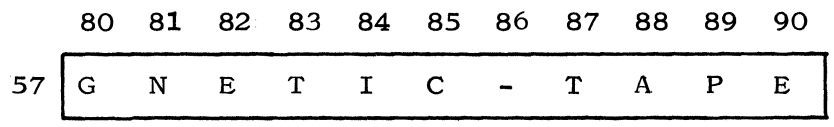
Example: 6 C 5790 5780

Tape on Tape Station C





HSM Before

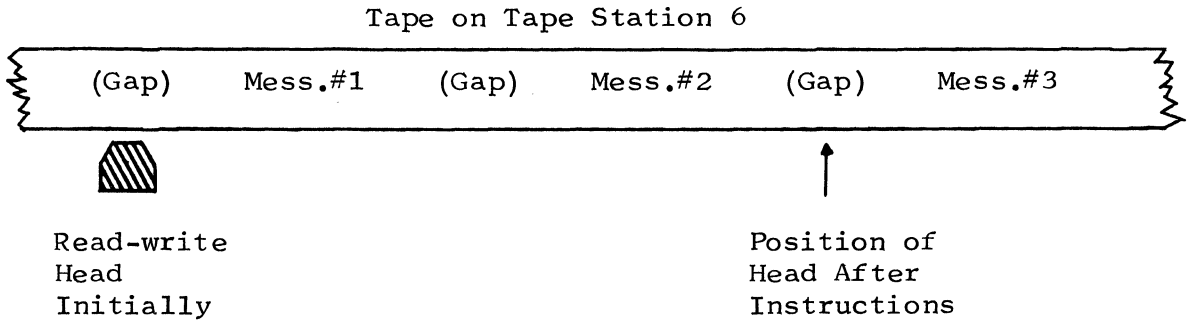


HSM After

PRP would be set and A_f would be 5779.

Although the tape instructions terminate on A-B equality, even if a gap has not been sensed, the Tape Station itself continues to run until it finds a gap. The characters on tape which were not read into HSM still exist on tape. Because of this double method of terminating the read instructions, positioning tape without destroying memory is quite easy.

For example, assume that the read-write head was positioned as shown below and it was desired to read Message No. 3 only from Tape Station 6.



R	1	0001	0001
4	6	1000	1000

The above two instructions would read one character from message No. 1 and one character from message No. 2 into the same HSM location (1000). The read-write head would be in position to read Message No. 3, with only one location in memory having been disrupted. This technique is very desirable for variable length messages.

N. TAPE READ REVERSE SIMULTANEOUS (7) (RRS)

This instruction has the same format as RRN except that the operation code is a 7 and the instruction is executed in the Simultaneous Mode. The RRS instruction does not go through STA and does not set the PRI's. The ED/EF Simultaneous Indicator will become set if an ED or EF is read in alone.

See page IX-9 of Programmers' Reference Manual.

0. TAPE WRITE NORMAL (8) (TWN)

This instruction writes a specified number of characters from HSM to a designated Tape Station, Paper Tape Punch or Monitor Printer. The instruction operates from left to right and the tape moves forward. The TWN instruction does not go through STA and does not set the PRI's. Instruction terminates on A-B equality. See page IX-10 of Programmers' Reference Manual.

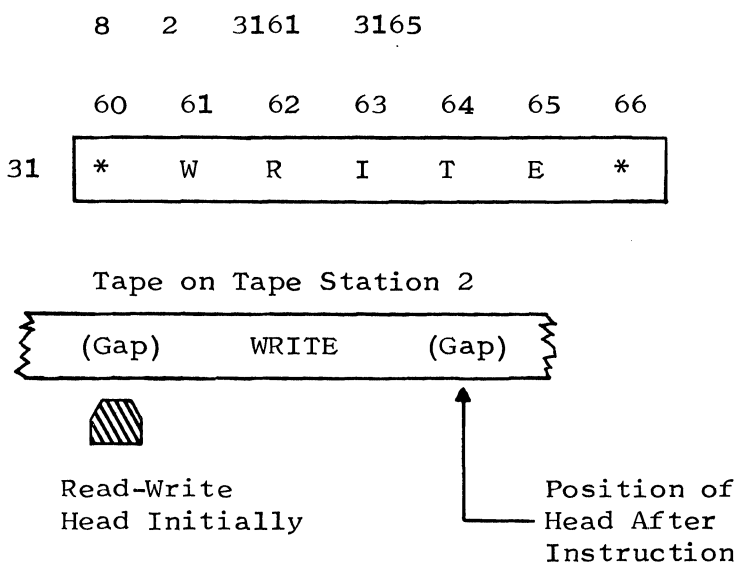
Op. Code	N	A	B
8	Identification Character of Tape Station Paper Tape Punch or Monitor Printer	Address of First Character to be Written	Address of Last Character to be Written

N	DEVICE
1, 2, 3, 4, 5, 6	Magnetic Tape
A, B, C, D, E, F	Magnetic Tape
J	33KC Adapter (1st Unit)
N	33KC Adapter (2nd Unit)
7	Monitor Printer
9	Paper Tape Punch
L	66KC Adapter (1st Unit)
P	66KC Adapter (2nd Unit)

A_f = Address of location one to the right of the last character written or punched.

$B_f = B_i$

Example:



The letters "WRITE" are placed on tape when the above instruction is executed.

P. TAPE WRITE SIMULTANEOUS (9)(TWS)

This instruction follows the same format as the TWN instruction, except that the operation code is a 9 and the instruction is executed in the Simultaneous Mode. Page IX-12 of the Programmers' Reference Manual covers the TWS instruction.

Q. PRINT AND PAPER ADVANCE NORMAL (B) (PAN)

This instruction can cause the Line Printer to print 120* consecutive characters (one line) from the contents of HSM, and/or advance paper for the next line of printing. The paper advance can be controlled by the instruction itself or by a paper tape loop on the printer. The instruction operates from left to right and does not go through STA. See page X-3 of Programmers' Reference Manual.

A_f = Address of location one to the right of the last character printed if printing is done, otherwise, it is A_i.

$B_f = B_i$ with B_3 set to zero.

*335 Printer normally prints 160 characters per line.

Op. Code	N	A	B
B	Number of Lines (0-14) to Advance Paper if B_3 equals 1 2^5(0) - Asychronous 2^5(1) - Synchronous 2^4(0) - 1st Unit 2^4(1) - 2nd Unit	Ignored	Address of Data to be Printed from HSM Excluding B_3 B_0 - MSD of address B_1 - If digit is even printing will occur. If odd - no printing. B_2 - Always zero B_3 - Indicates type of paper advance

NOTE: N Count for PAN & PAS is shown on page X-3 of Programmers' Reference Manual.

Type of Paper Advance		B_3
No Paper Advance		0
Line Shift	using N as count	1
Vertical Tab	(using paper tape loop)	2
Page Change	(using paper tape loop)	3

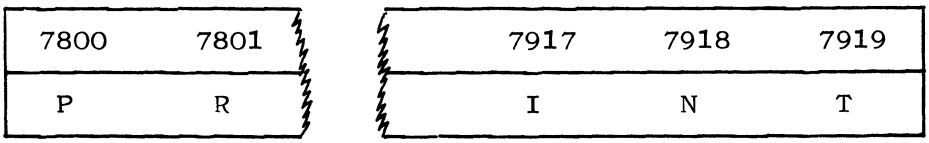
Two modes exist in the printer - synchronous and asychrounous. The synchronous mode gives the fastest printing rate at 1000 lines per minute or 660 lines per minute, but has only 47 printable characters. The non-synchronous mode has 64 printable characters but prints at a slower rate or 790 or 590 lines per minute. See Page II-3 of Programmers' Reference Manual for a list of printable characters in each mode.

Example 1: B 3 0000 3501

In this example, no printing will occur since the B_i character (5) is odd. Only Paper Advance will take place. Three lines of paper will be advanced on the 1st unit printer.

A_f = 0000 B_f = 3500

Example 2: B 2 0000 7803



In this example, characters between locations 7800 through 7919 would be printed and paper would be advanced, using the paper tape loop, one page. Printing would occur in the asychronous mode on the 1st unit printer.

A_f = 7920 B_f = 7800

R. PRINT AND PAPER ADVANCE SIMULTANEOUS (C) (PAS)

This instruction follows the same format as the PAN instruction, except that the operation code is a C and the instruction is executed in the Simultaneous Mode.

S. BAND SELECT NORMAL (D) (BSN)

This instruction searches one Record File for a specific disc, places the disc with the correct side facing up on the turntable and electronically positions the read-write head over one of two bands.

Before progressing further with the Band Select Normal instruction or any other Record File instruction, a general description of the device involved is necessary.

The Record File has a capacity of 128 discs which are coated with a substance similar to magnetic tape on both sides. Each surface of the disc is divided into two concentric spiral bands. Each band is divided into ten cells and each cell can hold a maximum of 900 characters recorded in serial fashion.

A cell is defined as two revolutions of the disc (speed of disc is 300 rpm) and the revolutions are counted by a black dot passing before a photo-sensing diode.

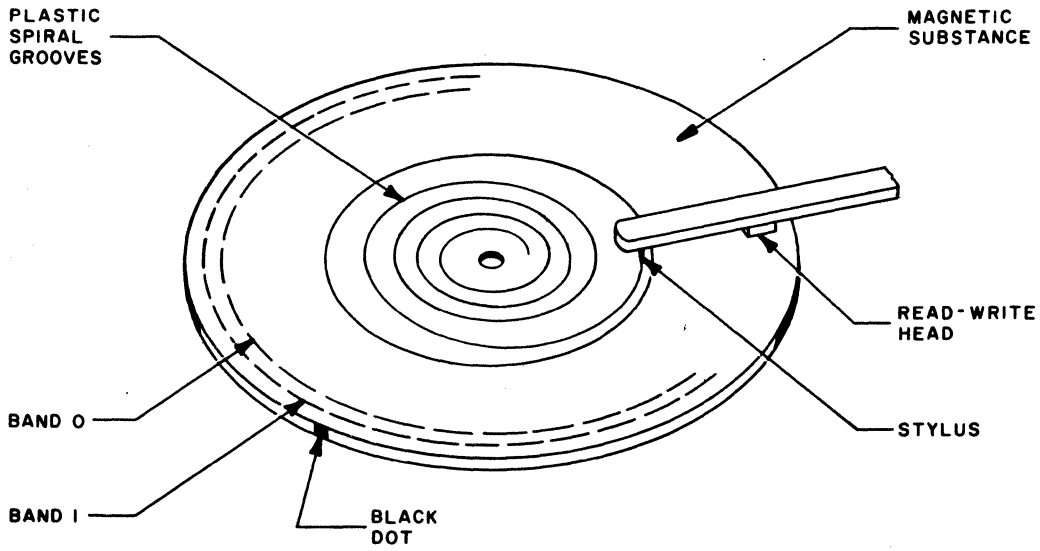


Figure 74 Simplified Illustration of Disc on Record File

The read-write head is connected to a stylus which tracks a plastic spiral groove in the center of the disc. As the disc revolves, the stylus moves toward the center pulling the read-write head with it. Thus the read-write head tracks the magnetic surface in relationship to the plastic spiral

grooves. Two read-write heads permit the tracking of two interwoven "bands." The Band Select instruction must determine which read-write head will be energized electronically.

Since the capacity of one cell is 900 characters, one band (10 cells) can hold 9000 characters, and one disc (4 bands at 2 per side) can store 36000 characters. Therefore, one Record File with 128 discs can hold approximately 4.6 million characters. Two Record Files can be incorporated under control of the Normal and Simultaneous Modes and four additional Record Files can be included under control of a separate mode - the Record File Mode. Hence, six Record Files with a capacity of about 27.6 million characters could be controlled by one 301 Processor.

Since it is necessary to locate a specific band on one of 128 records, some arrangement of identification is needed. With four bands per disc and 128 discs in a file, there are 512 bands which must be identified. Addresses 000 through 511 are used to identify these 512 bands. The breakdown of bands and their addresses is as follows:

Disc No. 0		Disc No. 1	
Band No. 000	Top Side (zero side)	Band No. 004	Top Side (zero side)
Band No. 001		Band No. 005	
Band No. 002	Reverse Side (one side)	Band No. 006	Reverse Side (one side)
Band No. 003		Band No. 007	

The lowest band address on a record can be obtained by using the formula, 4N, where N is the number of the record (0 through 127). For example, to determine the band addresses for disc number 15, the formula can be applied (4 x 15 = 60) to show that 060 and 061 are the band addresses for the zero side of the disc and 062 and 063 are the band addresses for the one side of the disc. Note that each side of a record contains an even and an odd numbered band and that the innermost band is the even numbered band.

To determine the disc number, when the band address is known, the band address can be divided by four, e.g., 063 $\div$ 4 = 15. Note that any remainder would indicate the side of disc and odd or even band.

From the above information, it can be seen that the Band Select instruction must include an address between 000 and 511. From this address, the Computer will locate the correct disc, determine which surface is required, place the disc on the turntable in proper position, and electronically select the even or odd band on that side.

Once a Band Select instruction has been initiated, a large portion of the operation performed is independent of the Computer. A Band Select is necessary before every Read or Write instruction, unless it is known that the desired band has already been selected. Once a Read or Write has been accomplished, the read-write head returns to the beginning of the band that was selected prior to the read or write operation.

The BSN instruction does not go through STA and is covered on page XI-3 of the Programmers' Reference Manual.

$A_f = A_i$

$B_f = B_i$

Op. Code	N	A	B
D	See Below	Ignored	B_0 Ignored B_1, B_2, B_3 - Band Address (000-511)

N Character Bits	Function
2^5, 2^3, 2^2, 2^1	Ignored
$2^4(0)$	Selects Unit 1
$2^4(1)$	Selects Unit 2
$2^0(0)$	Returns any disc on turntable to basket
$2^0(1)$	Uses disc already on turntable

The N character 2^0 bit has the function of returning the disc presently on the turntable to the basket and searching for the correct disc which corresponds to the B Address or leaving the present disc on the turntable and choosing the even or odd band. The object of this is to save time if the disc presently on the turntable is known to be the desired one. If N 2^0 is a zero bit, the disc currently on the turntable if any, will be returned to the cage and the search for the correct one will begin. If N 2^0 is a one bit, the current disc is left on the turntable and the odd or even band will be selected according to the address.

NOTE: If the disc on the turntable is the desired one but the desired band is on the opposite side of the disc, the N character 2^0 bit must be a zero in order that the disc will be flipped over.

Example: D 0 0000 0012

The even band on the zero side of disc number 3 on the first unit is selected from the address 012. Because N 2^0 is a zero bit, any disc on the turntable prior to this instruction would be returned to the basket.

T. BAND SELECT RECORD FILE MODE (E) (BSM)

This instruction has the same format as the Band Select Normal instruction except that the N character will select one of four Record Files under control of the Record File Mode. The BSM instruction does not go through STA but is partially executed independent of the Computer once it is initiated. Page XI-5 of the Programmers' Reference Manual covers the Band Select Record File Mode instruction. $A_f = A_i$ $B_f = B_i$

Op. Code	N	A	B
E	See Below	Ignored	B_0 Ignored B_1, B_2, B_3 - Band Address (000-511)

N Character Bits	Function
2^3, 2^2, 2^1	Ignored
$2^5(0)$, $2^4(0)$	Selects Unit 1
$2^5(0)$, $2^4(1)$	Selects Unit 2
$2^5(1)$, $2^4(0)$	Selects Unit 3
$2^5(1)$, $2^4(1)$	Selects Unit 4
$2^0(0)$	Returns any disc on turntable to basket
$2^0(1)$	Uses disc already on turntable

Example: E " 0000 0104

This instruction would select band 104 (first band on disc number 26) of Unit 4. Any disc presently on the turntable would be returned to the basket first.

U. BLOCK READ FROM RECORD NORMAL (F) (BRN)

This instruction reads from a selected band on a Record File (beginning with a specified cell) into high speed memory. From one to ten "blocks" of information can be read with one BRN instruction. A "block" is defined as the contents of one cell. This can be from one to 900 characters, but if less than 900, the block of characters must be terminated by an E_B symbol (End of Block). The Read instruction will terminate a block of information by finding either an E_B symbol or a 900 count as programmed. When the specified

number of blocks have been read (N = 0), the instruction terminates. Block Read from Record Normal goes through STA and is covered on page XI-7 of the Programmers' Reference Manual.

A_f = One location to the right of the last character read.

$B_f = B_i$

Op. Code	N	A	B
F	Number of Blocks to be Read (From one to Ten with Zero Representing Ten) $2^4(0)$ = Unit 1 $2^4(1)$ = Unit 2	Location to Receive First Character of First Block	See Below

B_0 - Ignored.

B_1 - Determines if disc remains on turntable after read.

B_1 = 1 - Disc is returned.

B_1 = 0 - Disc remains on turntable and read-write head is positioned back at the beginning of previously selected band.

B_2 - Specifies type of termination for each block.

B_2 = 1 - Block is terminated by 900 character count only.

B_2 = 0 - Block is terminated by either E_B or 900 count.

B_3 - Addresses cell (0-9) from which to begin the read.

NOTE: A Band Select instruction must have previously been executed.

Example:

```
D   0   0000   0002
F   4   1000   0105
```

The above combination of instructions would read the contents of cells 5, 6, 7 and 8 of band 002 on Unit 1 into memory beginning at 100. The number of characters read in could vary since termination of a block could be by an E_B symbol or by 900 count. The disc would be returned to the cage when reading was finished.

V. BLOCK READ FROM RECORD SIMULTANEOUS (G) (BRS)

This instruction has the same format as the Block Read from Record Normal except that the operation code is a G and the instruction is executed in the Simultaneous Mode. BRS does not go through STA and is covered on page XI-9 of the Programmers' Reference Manual.

W. BLOCK WRITE TO RECORD NORMAL (H) (BWN)

This instruction has a similar format to the Block Read from Record instructions except that writing is done rather than reading. From one to ten blocks of information can be written to a previously selected band. The blocks of information can be defined by E_B symbols or 900 character count. If the number of blocks to be written exceeds the number of available cells remaining in the selected band, the excessive blocks of information will be lost. The BWN instruction does not go through STA and is covered on page XI-11 of the Programmers' Reference Manual.

A_f = One location to the right of the last character written.

$B_f = B_i$

Op. Code	N	A	B
H	Number of Blocks to be Written (From One to Ten with Zero Representing Ten) $2^4(0)$ = Unit 1 $2^4(1)$ = Unit 2	Location of First Character to be Written	See Below

B_0 - Ignored.

B_1 - Determines if disc remains on turntable after read.

B_1 = 1 - Disc returned to cage.

B_1 = 0 - Disc remains on turntable and read-write head is positioned at beginning of previously selected band.

B_2 - Specifies type of termination of each block.

B_2 = 1 - 900 count only.

B_2 = 0 - E_B or 900 count.

B_3 - Addresses first cell (0-9) to receive first block.

Example:

D	0	0000	0116
H	3	3100	0002

	00	01	02	03	04	05	06	07	08	09
31	F	I	L	E	E_B	0	F	E_B	E_B	I

HSM Before and After

In the above combination of instructions, band 116 of disc number 29 will be selected on Unit 1. The Write instruction will write out, "FILE E_B" to cell 2 of band 116, "OF E_B" to cell 3, and "E_B" to cell 4 since three blocks must be written and each block can be terminated by an E_B symbol or 900 character count. The disc would remain on the turntable when writing is complete and the read-write head would be positioned at the beginning of band 116. A final would be 3109.

X. BLOCK WRITE TO RECORD SIMULTANEOUS (I) (BWS)

This instruction is identical to the BWN except the operation code is an I and the instruction is executed in the Simultaneous Mode. Page XI-13 of the

Programmers' Reference Manual covers the BWS instruction.

Y. RECORD FILE MODE READ (*) (RMR)

This instruction has the same format as the Block Read from Record Normal except the instruction is executed in the Record File Mode and the N character must specify one of four Record Files under control of the Record File Mode. A Band Select Record File Mode (E) instruction must be used in conjunction with this instruction rather than Band Select Normal (D). The RMR instruction does not go through STA and is covered on page XI-15 of the Programmers' Reference Manual.

U_f (Same as A_F in Normal Mode) = One location to right of last character read.

V_f (Same as B_f in Normal Mode) = B_i

Op. Code	N	A	B
*	Number of Blocks to be Read from One to Ten with Ten Specified by Zero. The Zone Bits Select the Unit as Shown Below.	Location to Receive First Character of First Cell	See Below

N Bits	Select
$2^5(0)$ $2^4(0)$	Unit 1
$2^5(0)$ $2^4(1)$	Unit 2
$2^5(1)$ $2^4(0)$	Unit 3
$2^5(1)$ $2^4(1)$	Unit 4

B_0 - Ignored.

B_1 - Determines if disc remains on turntable.

B_1 = 1 - Disc returned to cage.

B_1 = 0 - Disc remains on turntable.

B_2 - Specifies type of block termination.

B_2 = 1 - 900 count only.

B_2 = 0 - E_B or 900 count.

B_3 - Addresses cell (0-9) to be read.

Example:

```
E   &   0000   0425
*   E   1000   0103
```

The preceding pair of instructions would read the contents of cells 3, 4, 5, 6 and 7 of band 425 on Unit 2 Record File (under control of the Record File Mode). The disc would be returned to the cage after the operation was complete.

Z. RECORD FILE MODE WRITE (%) (RMW)

This instruction functions similarly to the Block Write to Record Normal except the instruction is executed in the Record File Mode and the N character must select one of four units to be used. The RMW does not go through STA and is covered in the Programmers' Reference Manual on page XI-17.

U_f = One location to right of last character written.

$V_f = B_i$

Op. Code	N	A	B
%	Number of Blocks to be Written from One to Ten with Ten Specified by Zero	Location of First Character to be Written	See Below

N Bits	Select
$2^5(0)$ $2^4(0)$	Unit 1
$2^5(0)$ $2^4(1)$	Unit 2
$2^5(1)$ $2^4(0)$	Unit 3
$2^5(1)$ $2^4(1)$	Unit 4

B_0 - Ignored.

B_1 - Determines if disc remains on turntable.

$B_1 = 1$ - Return disc to cage.

$B_1 = 0$ - Leave disc on turntable.

B_2 - Specifies type of block termination.

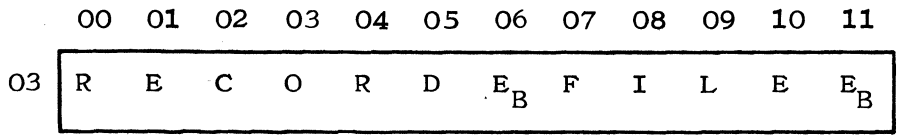
$B_2 = 1$ - 900 count only.

$B_2 = 0$ - E_B or 900 count.

B_3 - Addresses cell (0-9).

Example:

```
        E   "   0000   0105
        %   S   0302   0102
```



HSM Before and After

Executing the above instructions would write out "CORD E_B" to cell number 2 and "FILE E_B" to cell number 3 on band 105 unit number 4 under the Record File Mode control. The disc would be returned to the cage upon termination of the write instruction. U final would be 0312.

SECTION VI

Flow Charting and Coding

A. FLOW CHARTING AND CODING, INTRODUCTION

The RCA 301 computer is essentially a machine designed to automatically process information for business, commercial and industrial organizations. Since the majority of business operations are straightforward and basic in nature, the interpretation and conversion of business routines and transactions into machine language will be of primary concern to the programmer.

A thorough understanding of flowcharting is not required to code short programs, but when a program reaches any significant length, flowcharting becomes a very effective tool, i.e., it allows the programmer to solve the general problem and isolate the details. The purpose of this section is to help the reader develop a technique for flowcharting and coding to produce a program.

Most programmers maintain relatively standard flowcharting symbols. These symbols and their meaning are shown in Figure 75.

Figure 76 shows the 301 Computer Program Record. A completed record will be illustrated later.

Figure 77 shows the 301 Computer HSM Record. This is completed by the programmer and shows how the data pertaining to his program will be placed in memory. A completed 301 HSM Record will be shown later.

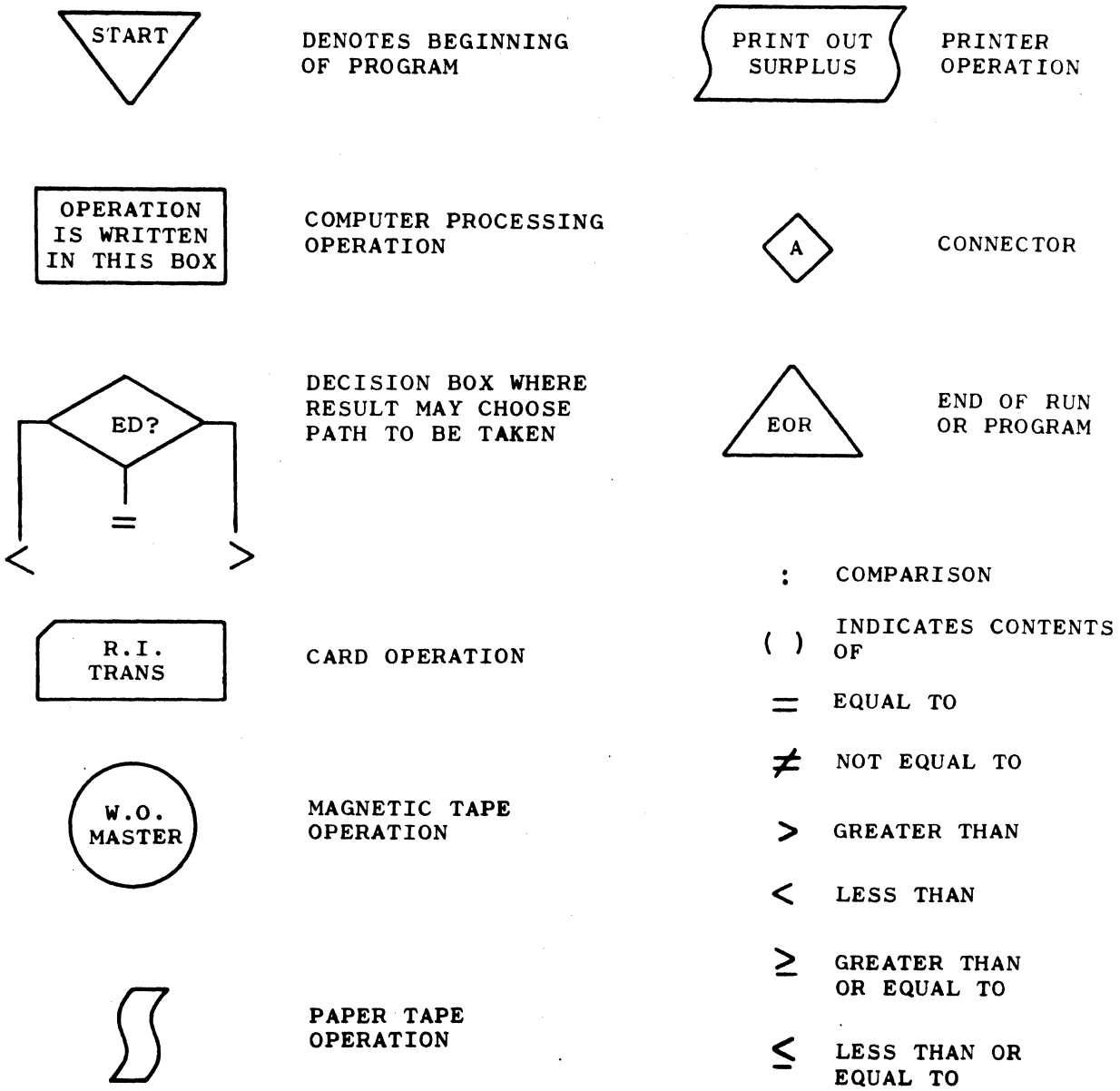


Figure 75 Flowchart Symbols

TITLE **301 COMPUTER PROGRAM RECORD** DATE
CODER INDEX NO.
REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	0												
	1												
	2												
	3												
	4												
	5												
	6												
	7												
	8												
	9												
	0												
	1												
	2												
	3												
	4												
	5												
	6												
	7												
	8												
	9												

Figure 76 301 Computer Program Record

301 COMPUTER HSM RECORD

	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						
	00 01 02 03 04 05 06 07 08 09	10 11 12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39	40 41 42 43 44 45 46 47 48 49						
	50 51 52 53 54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74 75 76 77 78 79	80 81 82 83 84 85 86 87 88 89	90 91 92 93 94 95 96 97 98 99						

TITLE: ______________ BLOCK NO.: ________ INDEX NO.: ________ PROGRAMMER: __________ DATE ________ PAGE___ OF__

Figure 77 301 Computer HSM Record

In flowcharting, the steps the computer must take to solve this problem would begin with reading in a transaction and the first message on the master file. The next step would be to compare the serial numbers to determine whether this particular item is being shipped or not. If the item is being shipped, the quantity on hand must be reduced and if the item is not being shipped, the quantity on hand should be unaltered. Thus, the beginning of the flowchart would be as follows:

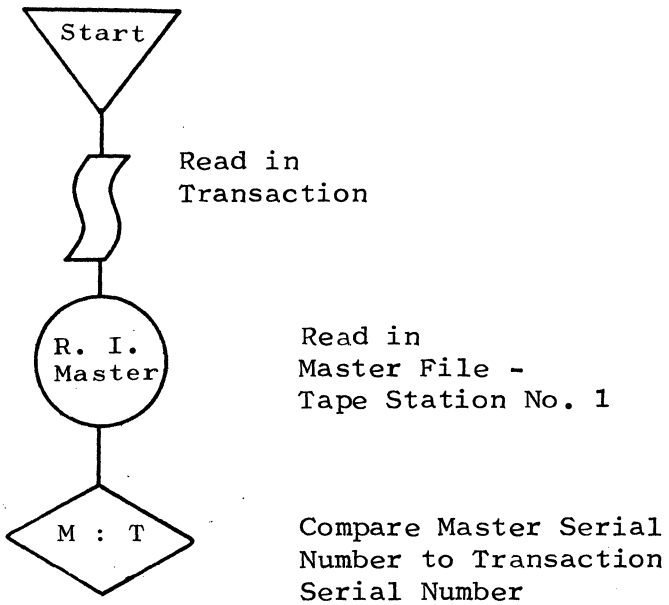


Figure 78 Start of Flowchart

From the decision block, which compares the serial numbers, three paths must be provided. The three paths would correspond to: M=T (serial numbers equal): M < T (transaction serial number is greater than master serial number); and M> T (master serial number is greater than transaction serial number). Since all master items are in sequence according to their serial numbers and all transactions are in sequence according to their serial numbers, at no time should the master serial number be greater than the transaction serial number. If this happens, an error has occurred in the original sorting of these items. Therefore, the program should provide for the recognition of this error.

If the serial numbers are equal, the item must have been shipped and the quantity on hand must be updated. After updating the quantity on hand, the master item must be recorded on the new master tape. Having done this, the computer must bring in the next message from the original master (on Tape Station 1). The process of comparing serial numbers would again resume with the next transaction read in from paper tape.

The third and final path provided (M < T) would signify that there is no transaction against this particular master. In other words no shipment of this item has been made, and the original master item must be recorded on the new master tape without any alterations.

The transaction, which is still in memory, has not been matched with its master item. Thus, a new master item must be read in, and these serial numbers must be compared. The final flowchart with just the bare essentials would be as follows.

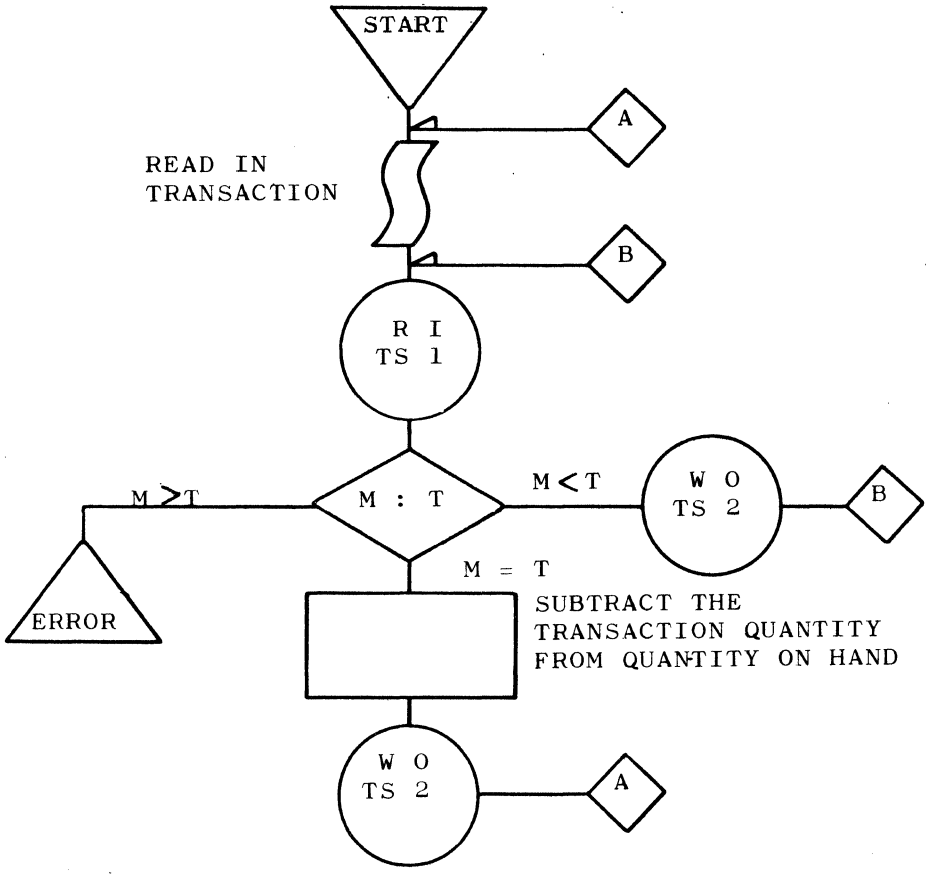


Figure 79 Simplified Flowchart

Before coding the program, read-in areas for the master and transaction files must be determined. Since the master file item is a total of nine characters, nine memory locations would be required for the master read-in area. Assume this area to be locations 1000 through 1008. The transaction consists of nine characters, thus, locations 1010 through 1018 would suffice as the transaction read-in area. If the program exists in memory beginning at address 2000, the coding for the flowchart would be as follows:

HSM LOCATION	INSTRUCTION				
2000	4	8	1010	1018	Read-in Transaction
2010	4	1	1000	1008	Read-in Master
2020	Y	4	1000	1010	M : T
2030	W	1	2070	2080	Sense PRI's PRP = M T PRN = M T
2040	-	5	1008	1018	Subtract shipment quantity from quantity on hand
2050	8	2	1000	1008	Write out new master
2060	V	1	0219	2000	Transfer control to 2000
2070	.	1	1111	0000	Error Stop
2080	8	2	1000	1008	Write out new master
2090	V	1	0219	2010	Transfer control to 2010

From the coding of the flowchart, it can be seen that approximately one instruction exists for every block on the flowchart. It should also be noted that no provision was made to stop the computer by instruction (except for Error Stop). This means that the computer would continue to process information contained on the master file tape and paper tape until one or the other was depleted. This would cause an Alarm Stop, due to physical end of tape. Therefore, the program should contain instructions which will recognize when the operation has been completed and stop the computer accordingly without alarms.

B. ED, EF AND ETW ROUTINES

Within the 301 System, certain control symbols exist. On magnetic tape two control symbols are used extensively. The EF symbol means end of file and will be placed on tape at the end of every distinct file of information, e.g., the master inventory file. This control symbol must be preceded and followed by a gap, and will constitute a message. All information on tape in the form of a file must be terminated by an EF symbol. The ED symbol means end of data and is placed on tape to specify the end of information or data on that particular reel. Thus, if a file extends beyond one reel of tape, there will be an ED at the end of the first reel and an EF followed by an ED on the last reel. Also, the ED symbol is preceded and followed by a gap and constitutes a message in itself.

The 301 computer incorporates a single indicator for each mode. There is an ED/EF Normal indicator for the Normal Mode, and there is an ED/EF Simultaneous indicator for the Simultaneous Mode. These indicators will become set when an ED or EF is read in from magnetic tape. Writing out an ED or EF does not set the indicator.

The fact, that an ED/EF control symbol will exist on tape, was not considered in the preceding inventory problem. Reading an ED or EF from paper tape does not set the ED/EF Indicator. Therefore, the check for these symbols must be by a compare against a constant if paper tape is used. Therefore, the flowchart must be ammended by incorporating ED/EF decision blocks as shown on the following page.

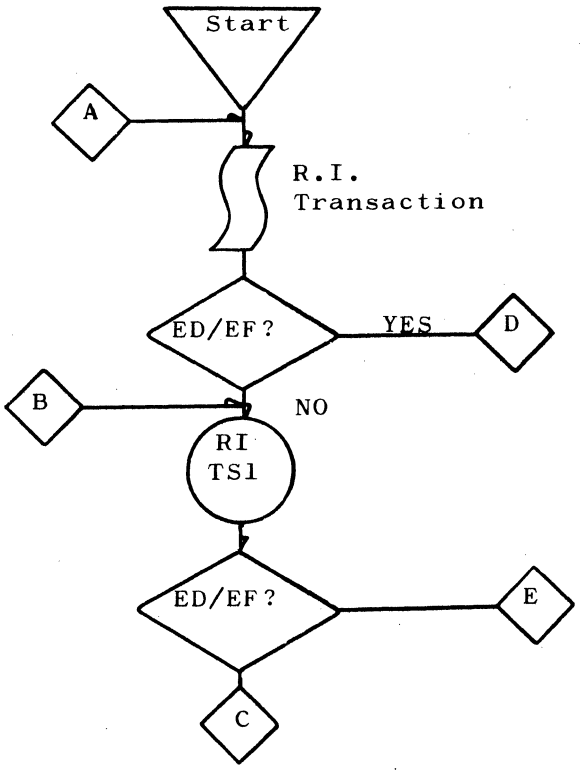


Figure 80 Flowchart Incorporating ED/EF Check

An ED/EF subroutine must be flowcharted from connectors D and E. A typical flowchart for this routine would be drawn as follows:

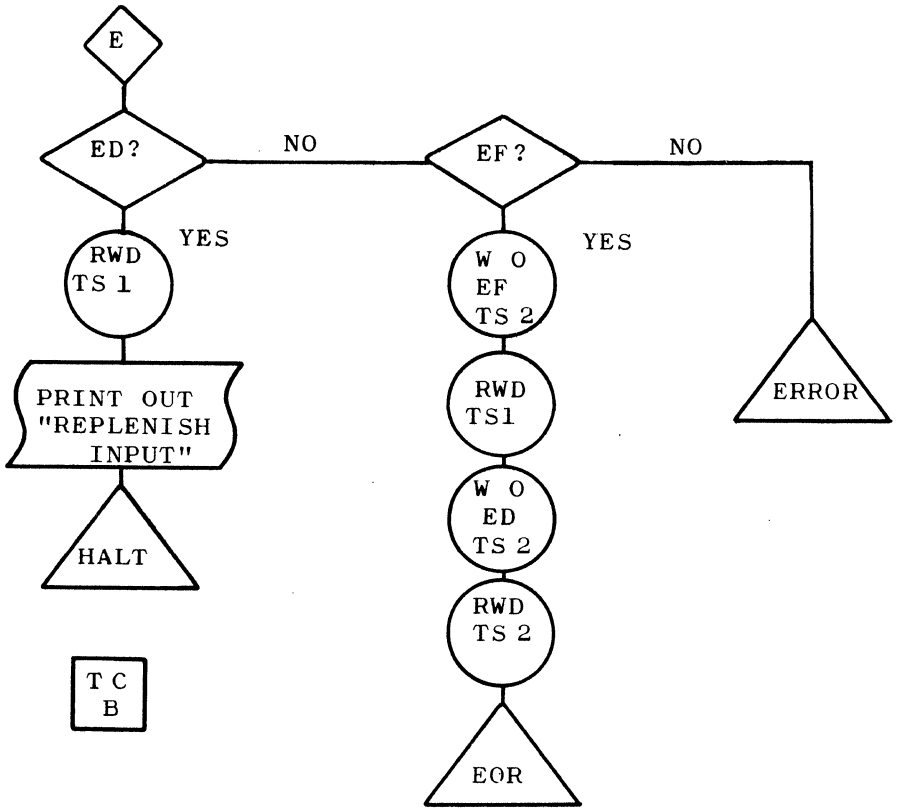


Figure 81 Flowchart Showing ED/EF Subroutine

This subroutine will determine whether the ED/EF indicator was set by (1) an ED symbol, (2) an EF symbol or (3) some character other than ED/EF.

Upon sensing an ED control symbol, the tape on Tape Station No. 1 will be rewound, the operator will be instructed by the print-out to place a new master file tape on Tape Station No. 1, and the computer will halt. The T.C.B (transfer control to connector B) block allows the computer to resume its processing when the operator depresses the Start button.

The presence of an EF on the master tape will indicate that the updating has been completed. An EF followed by an ED will be placed on the new master file tape and both tapes will be rewound.

Setting the ED/EF indicator, when neither an ED or EF control symbol exists, will cause an error stop.

The operator must know why the computer stopped. If the machine stops on an ED, "Replenish Input" will be printed, but, if the ED/EF indicator is set by some condition other than an ED, the operator would have to examine the contents of the P Register and then refer to the program in order to determine whether the updating has been completed or an error halt occurred. To simplify the above procedure, the three halts involved can be coded.

One method of coding the halts is by filling the N, A and B Registers with zeros or ones to indicate normal or error halts, respectively. The instructions would be coded as:

```
.   0   0000   0000  -  Normal Halt
.   1   1111   1111  -  Error Halt
```

Using this method, if there is no print-out, the operator can select the N, A or B Register, on the console and determine the type of halt without referring to the program.

There is a second method of coding halts, which is more inclusive, due to the fact that a program usually incorporates more than one normal halt and more than one error halt. Using the normal N Register count, 45 individual halts can be recognized without reference to the program. The ED/EF subroutine could be coded as:

```
.   0   0000   0000  -  Halt on ED
.   1   0000   0000  -  End of Run
.   2   0000   0000  -  Error Stop
```

The operator can recognize a specific halt with this method and, since the A and B Registers are not coded, the respective HSM locations can be used as storage or work areas. This method of coding the halts also has an advantage in the case where the printer is inoperable and "Replenish Input" is not printed. The print instruction can actually be eliminated, but, normally, most programmers include it in their program as an "insurance

policy." Another feature that should be included in the inventory problem is a means for checking the output tape to insure that information will not be lost because the new master file is too large to be recorded on one reel of tape. Sensing for ETW before every write-out will provide a solution to this problem. ETW is an abbreviation for End of Tape Warning and when this condition is sensed an indicator is set, which effectively tells the computer that approximately 24 feet of usable tape is left on the reel. A subroutine should be flowcharted to sense for ETW as follows:

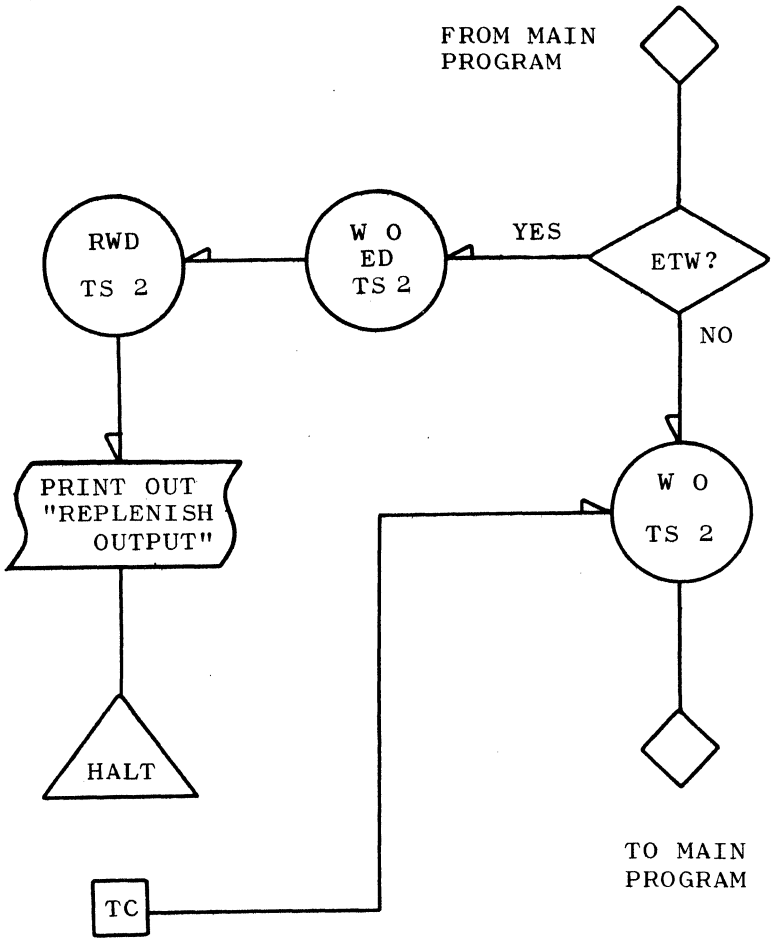


Figure 82 Flowchart Showing ETW Check

When ETW is sensed, an ED will be written to the New Master on Tape Station No. 2, the tape will be rewound and a command for the operator (replenish output) will be printed.

C. SWITCHES

When one subroutine is to be used for two or more portions of a program, it becomes economical to incorporate a switch or variable connector. A switch is normally represented in a flowchart as:

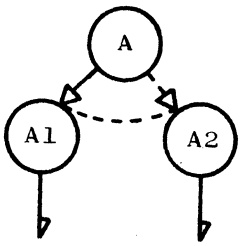


Figure 83 Symbol for Software Switch

From point A, the computer will take one of two paths - A1 or A2. The switch is set previous to the time the computer will reach point A. Physically a switch is simply one instruction - A transfer of control instruction or, in the case of the 301, a Store Register instruction with the N Character equal to 1. This designates storing the P Register contents and transferring control the B Address. To set the switch, one merely needs to transfer a predetermined address into the locations which make up the B Address of the Store Register instruction. Once a program has been coded, the addresses which have been temporarily omitted can be included as constants of the program.

For example, if the same subroutine were needed for two different sections of the main program, a switch could be used to determine where to return after executing the subroutine. The following coded example should help clarify this switch concept.

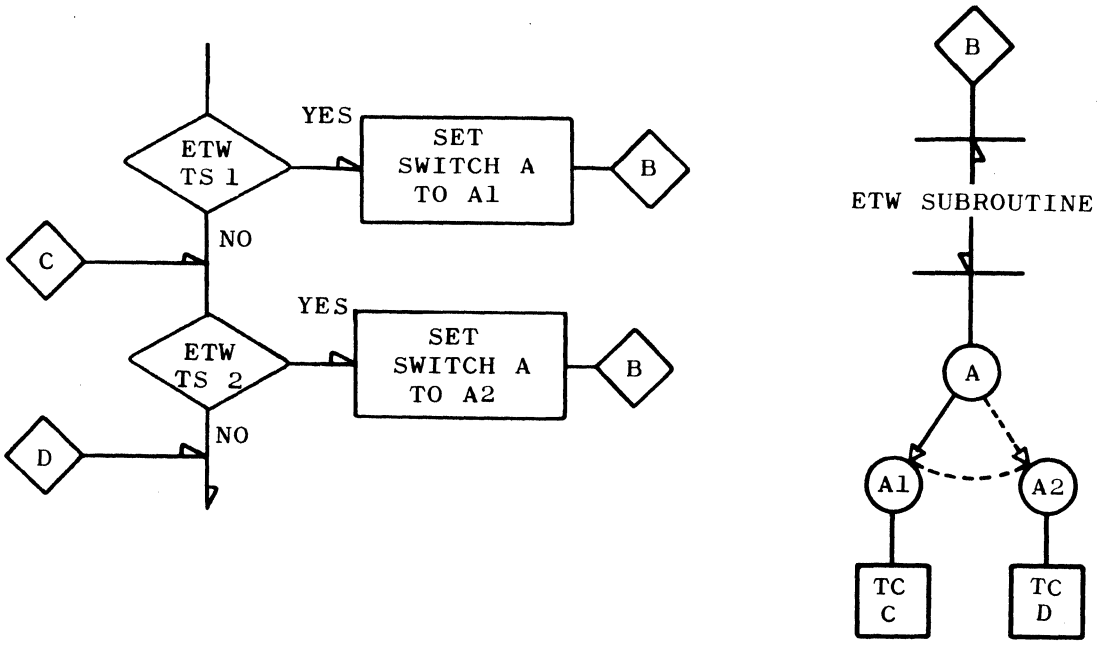


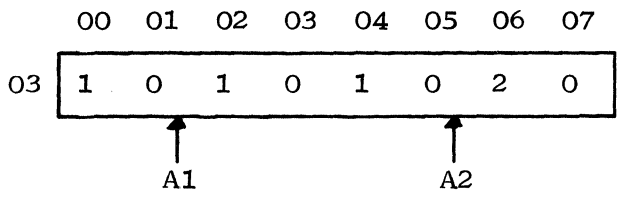
Figure 84 Example of Switch in Flowcharting

```
1000    S  1  4000  3000          3000    N  4  0303  4009
1010    S  2  4000  3020          3010    V  1  0219  3030
1020    Continue Main Program     3020    N  4  0307  4009

                                        ETW Subroutine

                                  4000  V  1  0219  Switch A
```

D. CONSTANTS



Switch A is the B Address of the Store Register instruction at 4000. The instruction at 3000 sets Switch A to A1 and the instruction at 3020 sets Switch A to A2.

Switches do not have to be limited to two paths. On the contrary, a switch may incorporate any number of alternatives. Of course, an address must be stored for each alternative.

E. HOUSEKEEPING

Included in almost every flowchart, and consequently in every program, is a series of preparatory operations, such as clearing work areas, rewinding tapes, restoring tally quantities and setting switches to their initial positions. Normally these operations are included under one large block called "housekeeping." This is usually the first block after the start symbol, and it represents all miscellaneous operations which ensure proper execution of the program.

The following pages illustrate a typical program. A banking problem is chosen which involves the need for producing an updated master account file for a given day's transactions. The transactions will be in the form of checks or deposits. The master file is on magnetic tape and blocks are of variable format. See the flowchart of Figure 85 (4 shts).

This flowchart illustrates the problem. Figure 86 presents the coded portion of the program and Figure 87, the 301 Computer HSM Record showing the information as it will appear in memory. All of the locations necessary for the program starting at 1000 are not shown. The program occupies locations 1000 through 2349.

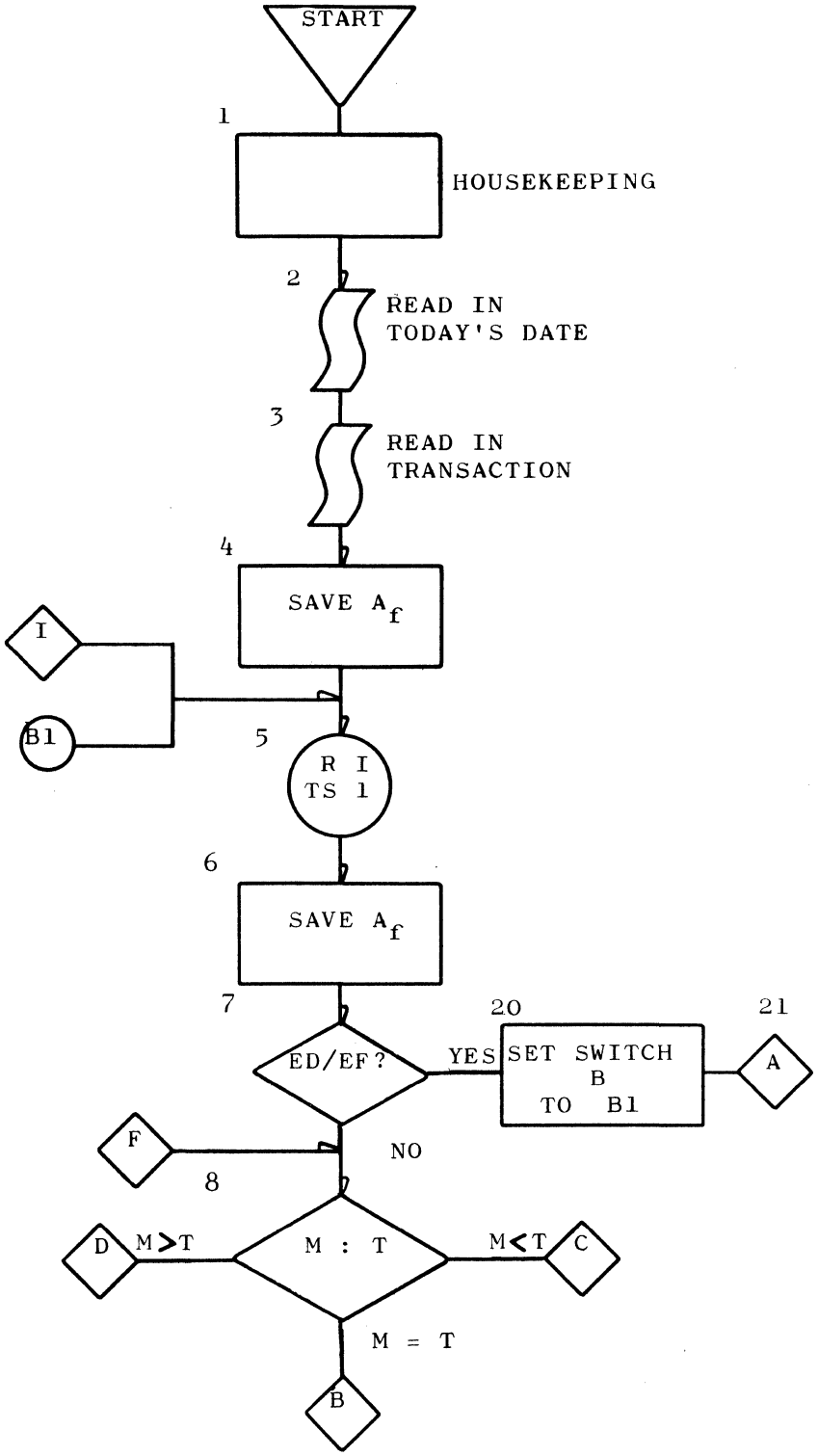


Figure 85 (Sheet 1) Flowchart for Updating A File

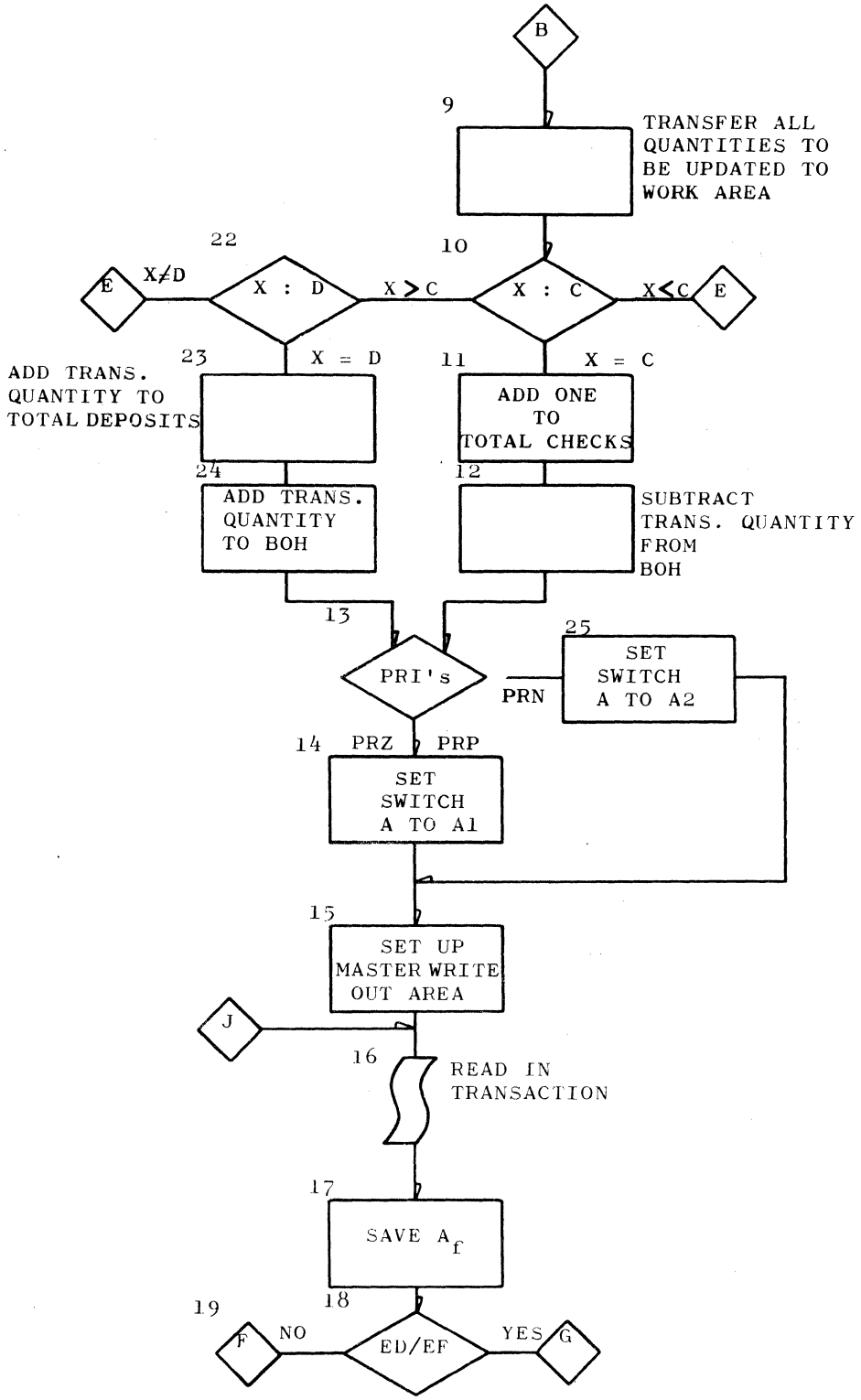


Figure 85 (Sheet 2) Flowchart for Updating a Blank File

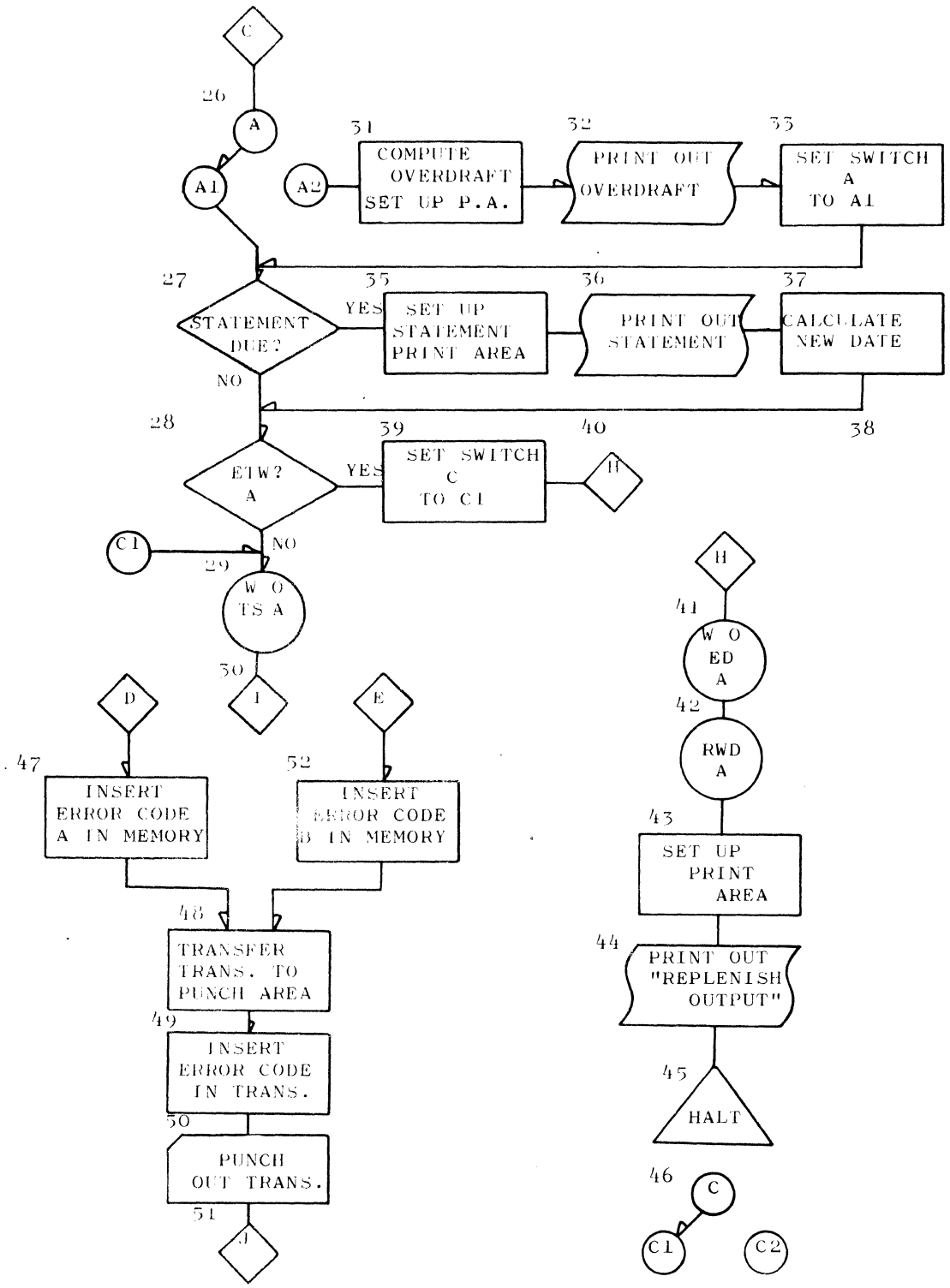


Figure 85 (Sheet 3) Flowchart for Updating a Bank File

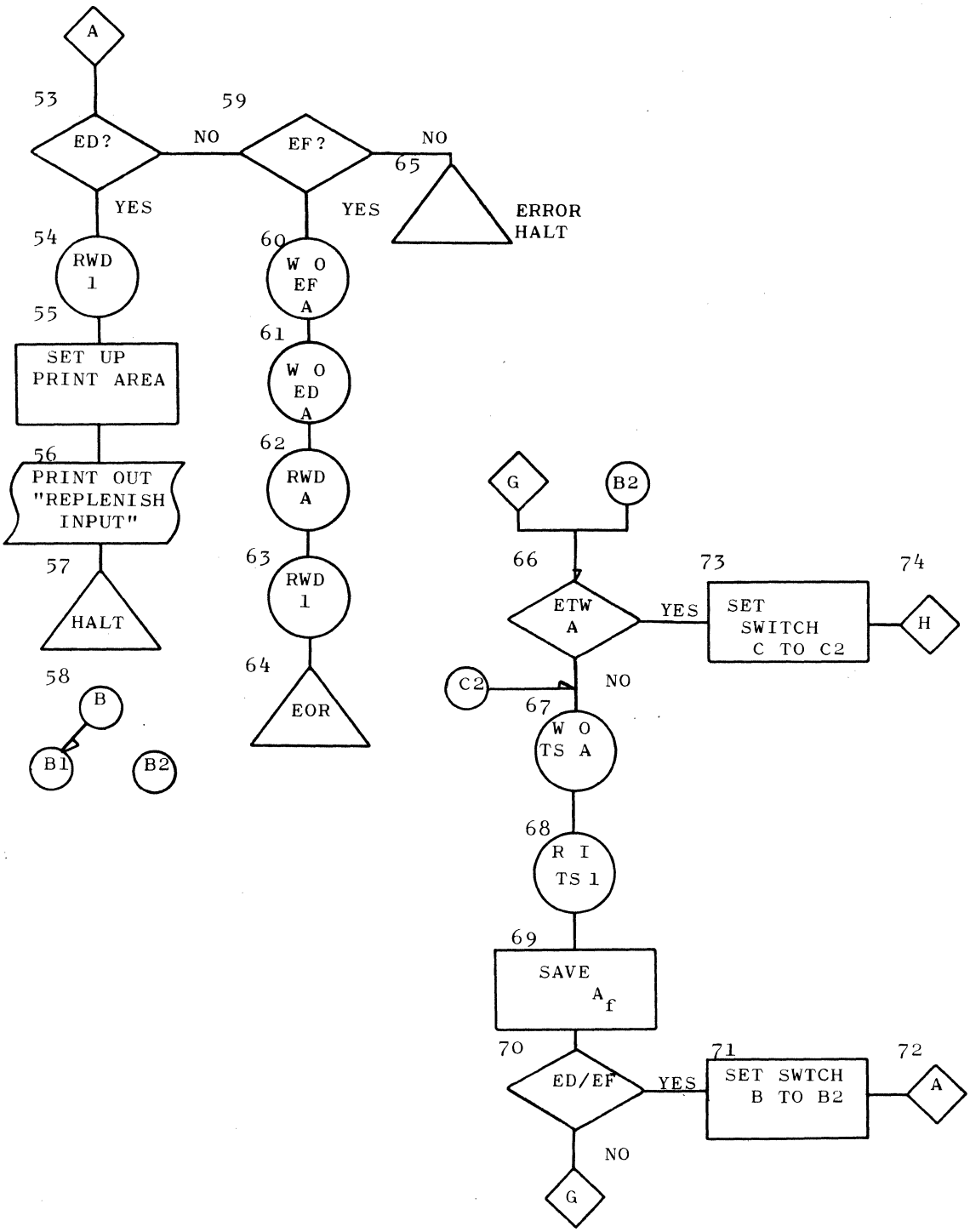


Figure 85 (Sheet 4) Flowchart for Updating a Bank File

TITLE UPDATING A MASTER BANK ACCOUNT FILE DATE

CODER INDEX NO.

REMARKS LOAD CONSTANTS IN AT 0635. LOAD PROGRAM AT 1000 BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	10 0	;	1	0	0	0	0	0	0	0	0	Rewind T.S. 1	1
	1	;	A	0	0	0	0	0	0	0	0	Rewind T.S. 2	1
	2	N	4	0	6	7	7	1	6	7	9	Set switch A to A1	1
	3	J	Ø	0	3	0	0	0	5	4	9	Fill work area	1
	4	J	@	0	5	3	0	0	5	3	0	Place @ symbol in BOHWA	1
	5	J	@	0	5	3	7	0	5	3	7	" " " " TCWA	1
	6	J	@	0	5	4	8	0	5	4	8	" " " " TDWA	1
	7	4	8	0	3	9	4	0	3	9	9	Read in today's date	2
	8	4	8	0	3	7	0	0	3	8	4	" " transaction	3
	9	N	4	0	2	1	5	0	3	6	5	Save A_f	4
1790	11 0	J	Ø	0	3	0	0	0	3	6	0	Clear master R.I. area	5
	1	4	1	0	3	0	0	0	3	6	0	Read in master	5
	2	N	4	0	2	1	5	0	3	6	9	Save A_f	6
	3	-	4	0	3	6	9	0	6	6	9	Adjust $A_f 2(-1)$	6
1560	4	W	8	1	5	7	0	1	1	5	0	ED/EF? yes-20 no-8	7
	5	Y	4	0	3	0	1	0	3	7	1	M:T (account numbers)	8
	6	W	1	2	0	7	0	1	6	7	0	PRI'S PRP→47 PRN→26	8
	7	P	@	0	3	6	I	0	3	9	3	Transfer statement date	9
	8	P	@	0	2	1	E	0	5	4	7	" total deposits→TDWA	9
	9	P	@	0	2	1	E	0	5	3	6	" " checks→TCWA	9

Figure 86 (Sheet 1) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	12 0	P	@	0	2	1	E	0	5	2	9	Transfer balance on hand→BOHWA	9
	1	K	Ø	0	5	3	8	0	5	4	7	Locate @ of TDWA	9
	2	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	9
	3	J	Ø	0	2	1	E	0	2	1	E	Mask out @	9
	4	K	Ø	0	5	3	1	0	5	3	6	Locate @ of TCWA	9
	5	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	9
	6	J	Ø	0	2	1	E	0	2	1	E	Mask out @	9
	7	K	Ø	0	5	2	0	0	5	2	9	Locate @ of BOHWA	9
	8	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	9
	9	J	Ø	0	2	1	E	0	2	1	E	Mask out @	9
	13 0	-	4	0	3	6	5	0	6	6	9	Adjust A_f1 of transaction (-1)	10
	1	Y	1	0	3	6	E	0	6	9	8	X:C	10
	2	W	1	1	5	9	0	2	1	6	0	PRI'S PRP→22 PRN→52	10
	3	+	5	0	5	3	6	0	7	1	1	Add 00001 to total checks	11
	4	P	@	0	3	6	E	0	3	6	E	Locate LSD of trans. quant.	12
	5	-	8	0	5	2	9	0	2	1	E	Subtract trans. quant. from BOH	12
1640	6	W	1	1	3	7	0	1	6	5	0	PRI'S PRP→14 PRN→25	13
	7	N	4	0	6	7	7	1	6	7	9	Set switch A→A1	14
1660	8	#	@	0	3	0	6	0	3	0	6	Locate MSD of BOH in original R.I. area	15
	9	J	Ø	0	2	1	E	0	3	6	I	Clear original read-in area	15

Figure 86 (Sheet 2) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	14 0	N	4	0	2	1	5	1	4	3	9	Transfer address of MSD of BOH → (B) of #	15
	1	K	Ø	0	5	2	0	0	5	2	9	Locate MSD of BOHWA	15
	2	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	15
	3	#	@	0	2	1	E	0	0	0	0	Transfer new BOH → W.O. area	15
	4	V	4	1	4	7	9	0	0	0	0	Store B_f → (B) of #	15
	5	K	Ø	0	5	3	1	0	5	3	6	Locate MSD of TCWA	15
	6	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	15
	7	#	@	0	2	1	E	0	0	0	0	Transfer new TC → W.O. area	15
	8	V	4	1	5	1	9	0	0	0	0	Store B_f → (B) of #	15
	9	K	Ø	0	5	3	8	0	5	4	7	Locate MSD of TDWA	15
	15 0	+	4	0	2	1	5	0	6	6	9	Adjust A_f(+1)	15
	1	#	@	0	2	1	E	0	0	0	0	Transfer new TD → W.O. area	15
	2	V	4	1	7	1	9	0	0	0	0	Store B_f → (B) of 28	15
2150	3	J	Ø	0	3	7	0	0	3	8	4	Clear transaction R.I. area	16
	4	4	8	0	3	7	0	0	3	8	4	R.I. trans.	16
	5	N	4	0	2	1	5	0	3	6	5	Save A_f1	17
	6	W	8	2	3	4	0	1	1	5	0	ED/EF? yes → 66 no → 8	18
1140	7	N	4	0	6	8	5	2	2	5	9	Set switch B → B1	20
	8	V	1	0	2	1	9	2	1	8	0	T.C. to 53	21
1320	9	Y	1	0	3	6	E	0	6	9	9	X:D	22

Figure 86 (Sheet 3) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	16 0	W	1	2	1	6	0	2	1	6	0	PRI'S PRP→52 PRN	22
	1	P	@	0	3	6	E	0	3	6	E	Locate trans. quantity LSD	23
	2	+	9	0	5	4	7	0	2	1	E	Add quantity to TDWA	23
	3	+	9	0	5	2	9	0	2	1	E	" " to BOHWA	24
	4	V	1	0	2	1	9	1	3	6	0	T.C.→13	24
1360	5	N	4	0	6	8	1	1	6	7	9	Set switch A to A2	25
	6	V	1	0	2	1	9	1	3	8	0	T.C.→15	25
1160	7	V	1	0	2	1	9	0	0	0	0	Switch A	26
1880	8	Y	6	0	3	8	8	0	3	9	4	Statement date: today's date	27
	9	W	1	1	7	1	0	1	8	9	0	PRI'S PRP→28 PRN→35	27
	17 0	V	1	0	2	1	9	1	8	9	0	T.C.→35	27
1960 1980	1	M	6	0	3	8	8	0	0	0	0	Transfer new statement date→W.O. area	28
	2	V	4	0	3	6	9	0	0	0	0	Store B_f over A_f2	28
	3	-	4	0	3	6	9	0	6	6	9	Modify $A_f2(-1)$	28
	4	S	A	4	0	0	0	1	9	9	0	Sense ETW yes→39	28
	5	8	A	0	3	0	0	0	3	6	I	Write out new master	29
	6	J	Ø	0	5	2	0	0	5	2	9	Fill BOHWA with zeros	29
	7	J	Ø	0	5	3	1	0	5	3	6	" TCWA " "	29
	8	J	Ø	0	5	3	8	0	5	4	7	" TDWA " "	29
	9	V	1	0	2	1	9	1	1	0	0	T.C.→5	30

Figure 86 (Sheet 4) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	18 0	J	@	0	4	0	0	0	5	1	9	Fill print area	31
	1	R	2	0	0	0	0	0	0	0	0	Transfer acct. no., name, address and BOH	31
	2	#	@	0	3	0	1	0	4	4	0		31
	3	V	4	1	8	5	5	0	0	0	0		31
	4	-	4	1	8	5	5	0	6	7	3	Subtract 0002 from B_f	31
	5	U	1	0	0	0	0	1	8	6	5	Mask out 2^5 bit	31
	6	C	9	0	0	0	$\overline{mi}$	0	4	0	1	Print out overdraft	32
	7	N	4	0	6	7	7	1	6	7	9	Set switch A to A1	33
	8	V	1	0	2	1	9	1	6	8	0	T.C.→27	34
1690 1700	9	J	@	0	4	0	0	0	5	1	9	Fill print area	35
	19 0	R	3	0	0	0	0	0	0	0	0	Transfer acct. no., name, address, BOH and	35
	1	#	@	0	3	0	1	0	4	4	0	Total deposits	35
	2	C	9	0	0	0	0	0	4	0	1	Print out statement	36
	3	Y	2	0	3	9	0	0	7	0	1	YR:12	37
	4	W	1	1	9	7	0	1	9	7	0	PRI'S	37
	5	+	4	0	3	9	1	0	7	0	5	Add 0089 to YRMO	37
	6	V	1	0	2	1	9	1	7	1	0	T.C.→28	38
	7	+	2	0	3	9	1	0	7	1	1	Add 01 to MO	37
	8	V	1	0	2	1	9	1	7	1	0	T.C.→28	38
1740	9	N	4	0	6	9	3	2	0	6	9	Set switch C→C1	39

Figure 86 (Sheet 5) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
2440	20 0	8	A	2	0	5	8	2	0	5	8	W.O. ED → T.S.A.	41
	1	;	A	0	0	0	0	0	0	0	0	Rewind A	42
	2	J	@	0	4	0	0	0	5	1	9	Fill print area	43
	3	M	F	0	6	5	0	0	4	5	0	Transfer "replenish output" → P.A.	43
	4	B	9	0	0	0	0	0	4	0	1	Print out	44
	5	.	0	1	1	1	1	0	0	E_D	E_F	Halt	45
	6	V	1	0	2	1	9	0	0	0	0	Switch C	46
1160	7	J	A	0	6	3	4	0	6	3	4	Insert error code A in constant location	47
2170	8	J	sp	0	5	5	0	0	6	2	9	Fill punch area	48
	9	R	2	0	0	0	0	0	0	0	0		48
	21 0	P	@	0	3	6	E	0	5	6	5	Transfer transaction to punch area	48
	1	N	4	0	3	6	5	0	6	3	3	Store B_f	48
	2	+	4	0	6	3	3	0	6	6	9	Adjust B_f (+1)	48
	3	N	1	0	6	3	4	0	6	3	C	Insert error code in transaction	49
	4	2	1	0	5	5	0	0	0	0	0	Punch out transaction	50
	5	V	1	0	2	1	9	1	5	3	0	T.C. → 16	51
1320 1600	6	J	B	0	6	3	4	0	6	3	4	Insert error code B in constant location	52
	7	V	1	0	2	1	9	2	0	8	0	T.C. → 48	52
2420 1580	8	Y	1	0	3	0	0	2	0	5	8	Char: ED	53
	9	W	1	2	2	6	0	2	3	3	0	PRI'S PRP → 59 PRN → error halt 65	53

Figure 86 (Sheet 6) 301 Computer Program Record

TITLE DATE

CODER INDEX NO.

REMARKS BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	22 0	;	1	0	0	0	0	0	0	0	0	Rewind 1	54
	1	J	@	0	4	0	0	0	5	1	9	Clear print area	55
	2	M	E	0	6	3	5	0	4	5	0	Transfer "replenish input"→P.A.	55
	3	B	9	0	0	0	0	0	4	0	1	Print out	56
	4	.	0	2	2	2	2	0	0	0	0	Halt	57
	5	V	1	0	2	1	9	0	0	0	0	Switch B	58
2190	6	Y	1	0	3	0	0	2	0	5	9	Char: EF	59
	7	W	1	2	3	3	0	2	3	3	0	PRI'S PRP-PRN→65 error halt	59
	8	8	A	2	0	5	9	2	0	5	9	W.O. EF→A	60
	9	8	A	2	0	5	8	2	0	5	8	W.O. ED→A	61
	23 0	;	A	0	0	0	0	0	0	0	0	Rewind A	62
	1	;	1	0	0	0	0	0	0	0	0	" 1	63
	2	∘	0	0	0	0	0	0	0	0	0	E.O.R.	64
2270 2190	3	.	1	0	0	0	0	0	0	0	0	Error halt	65
2400 1560	4	S	A	4	0	0	0	2	4	3	0	Sense ETW on A yes→73	66
	5	8	A	0	3	0	0	0	3	6	I	Write out new master	67
	6	J	Ø	0	3	0	0	0	3	6	0	Clear R.I. area	68
	7	4	1	0	3	0	0	0	3	6	0	R.I. master	68
	8	N	4	0	2	1	5	0	3	6	9	Save A_f	69
	9	-	4	0	3	6	9	0	6	6	9	Adjust $A_f(-1)$	69

Figure 86 (Sheet 7) 301 Computer Program Record

TITLE

CODER

REMARKS

DATE

INDEX NO.

BLOCK NO.

FROM INSTRUCTION LOCATION	HSM LOCATION	OP	N	A				B				REMARKS	CHART NO.
		0	1	2	3	4	5	6	7	8	9		
	24 0	W	8	2	4	1	0	2	3	4	0	ED/EF? yes→71 no→66	70
	1	N	4	0	6	8	9	2	2	5	9	Set switch B→B2	71
	2	V	1	0	2	1	9	2	1	8	0	T.C.→53	72
2340	3	N	4	0	6	9	7	2	0	6	9	Set switch C→C2	73
	4	V	1	0	2	1	9	2	0	0	0	T.C.→41	74
	5												
	6												
	7												
	8												
	9												
	0												
	1												
	2												
	3												
	4												
	5												
	6												
	7												
	8												
	9												

PAGE 8 OF 8

Figure 86 (Sheet 8) 301 Computer Program Record

301 COMPUTER HSM RECORD

	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
03	←			MASTER	READ	IN	AREA →				
03	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
			→	Af1	Af2	← TRANSACTION READ	IN AREA →		STATEMENT DATE	TODAY'S DATE	
04	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
	←										
04	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
					← PRINT OUT	AREA →					
05	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
				→	← BOHWA →		@ TCWA →	@ ←	TDWA →	@	
05	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
	←			CARD PUNCH AREA							
06	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
						→	Bf1 Ec	REPLE	NISH_	INPUT	
06	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
	REPLE	NISH_	OUTPU	T0001	00021	68018	00110	02340	17502	35OCD	
07	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
	12008	90000	01								
07	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
08	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
08	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
09	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
09	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
10	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
	←										
10	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
11	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
					PROGRAM						
11	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	
12	00 01 02 03 04	05 06 07 08 09	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28 29	30 31 32 33 34	35 36 37 38 39	40 41 42 43 44	45 46 47 48 49	
12	50 51 52 53 54	55 56 57 58 59	60 61 62 63 64	65 66 67 68 69	70 71 72 73 74	75 76 77 78 79	80 81 82 83 84	85 86 87 88 89	90 91 92 93 94	95 96 97 98 99	

TITLE: UPDATING A MASTER BANK ACCOUNT FILE BLOCK NO.: _____ INDEX NO.: _____ PROGRAMMER: _____ DATE _____ PAGE 1 OF 1

Figure 87 301 Computer HSM Record

SECTION VII

Processor Logic Descriptions

A. PROCESSOR LOGIC DESCRIPTIONS, INTRODUCTION

Various portions of logic in the Basic Processor will now be discussed in detail with reference to the Engineering Logic Diagrams. The logic blocks to be covered in sequence are:

A. Time Pulse Generator
B. Status Level Generation and Selection
C. NOR and Operation Decode Matrix
D. N Register
E. Addressable Registers
F. Bus Adder
G. MR and Interchange
H. NR Register
I. D Register
J. D Comparator
K. Standard Address Generator

B. TIME PULSE GENERATOR (Engineering Diagram 3506929, Training No. 079).

The 301 Time Pulse Generator under control of a one-megacycle oscillator produces seven sequential one-microsecond pulses, TP0, TP1, TP2, TP3, TP4, TP5, and TP6. One cycle of time pulses is equivalent to the time of one 301 memory cycle, or one status level. The Time Pulse generator also produces combination pulses by means of OR gates such as TP01, TP123, etc. Normally these combination pulses exist for the duration of the individual time pulses which produce them.

1. Detailed Logic

Pushing the START button on the console sets Flip-flop 0797C1, which in turn produces a low output to pin 10 of AND Gate 0797C3. If the Start Inhibit ST INH(P) level is not present, a high output will result from that AND Gate. ST INH exists if any select switch is set on the console. The output from 0797C3 will set the Delayed Start (DELST) Flip-flop and also produce, through a pulse chopping network, a one-microsecond pulse labeled Start Reset, ST RES(P) . ST RES (P) then sets the TP01 Flip-flop and initiates TP01.

ST RES(P) also sets the DPO Flip-flop and resets the DP6 Flip-flop. These DP or Delayed Pulse Flip-flops act as intermediate stages between time pulses, and ensure that the time pulses will occur in sequence, rather than simultaneously with every low oscillator pulse.

Once the DELST Flip-flop is set, a low is present on pin 16 of AND Gate 0797B2. A second low on pin 12 of that AND Gate will occur when the START button is released and Flip-flop 0797B3 becomes reset. The output of AND Gate 0797B2 will not be high, however, until a third low is present on pin 15. This low will be obtained from the oscillator which produces a one-megacycle output, as long as power is supplied to the system (See Figure 88). All three lows being present on 0797B2 will reset the Stop Flip-flop. The reset output of the Stop Flip-flop then provides a prime to AND Gate 0796D1 (pin 18). The other low to that AND Gate (pin 17) will not occur until AND Gate 0797B4 receives a high input pulse from the oscillator. Thus, when AND Gate 0796D1 receives its two lows, it will produce a high output which will reset the DELST Flip-flop and also prime pin 9 of AND Gate 0796D4. Once the DELST Flip-flop becomes reset, AND Gate 0797B4 will be continually inhibited and its output always low. This low output in turn will provide a constant prime to 0796D1 which was previously dependent upon the oscillator pulse. Note that STOP becoming reset inhibits generation of another series of time pulses by applying a high to pin 19 of Flip-flop 0797C1 (set output). AND Gate 0796D4 is the key to starting the series of time pulses. In its present state, 0796D4 simply needs a low pulse from the oscillator, to produce a high output. If termination conditions exist, ST(P) (Stop) will be present (originates on Training Drawing 078, A-7). This level will inhibit the production of TP1 and all succeeding time pulses. If ST(P) goes high after TP1 is initiated, it can do nothing until TP1 is about to occur again. Therefore, the computer will always finish generating time pulses through TP0 before stopping.

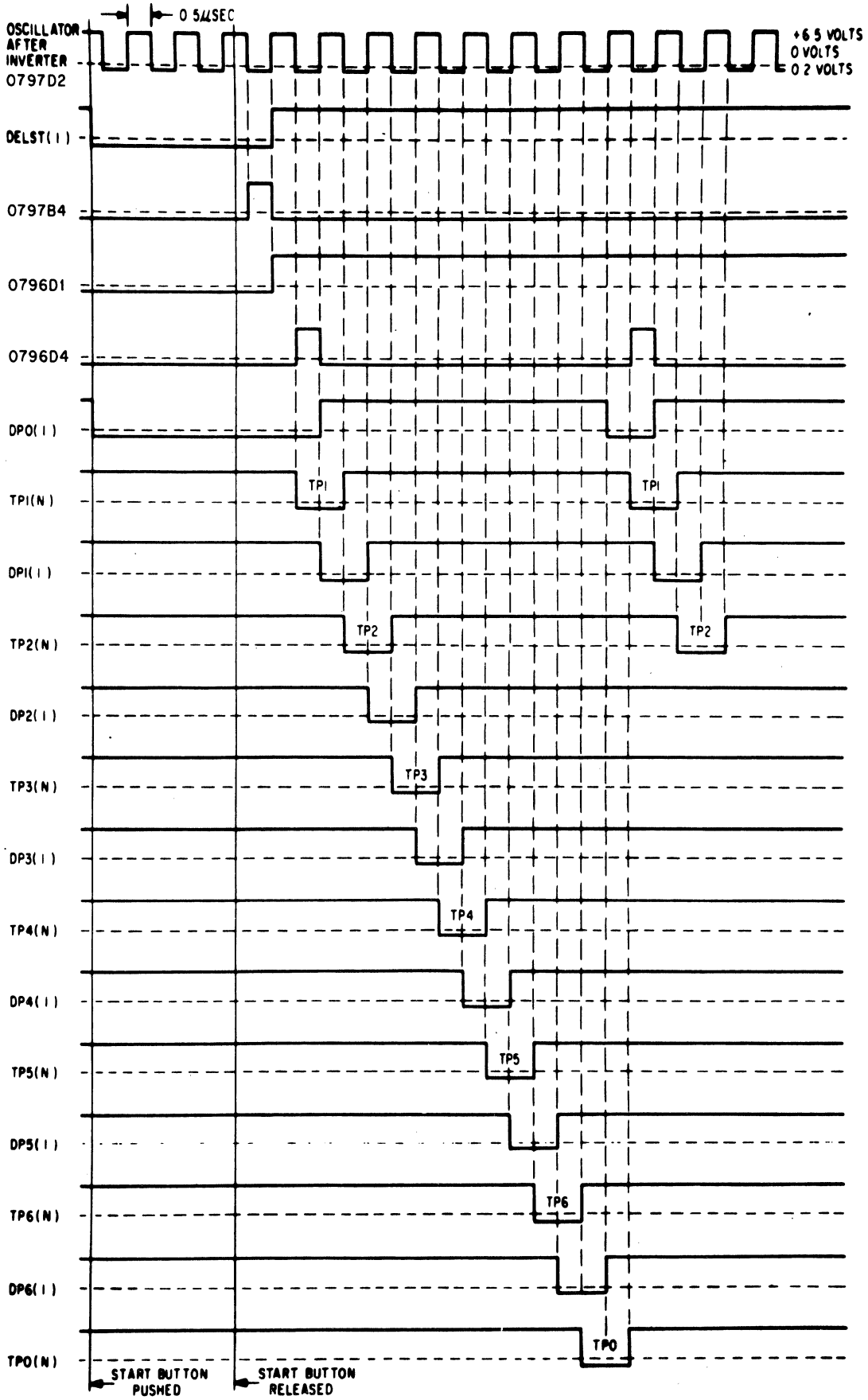


Figure 88 301 Time Pulse Generation Timing Chart

As soon as 0796D4 has a high output, the TP1 Flip-flop becomes set and TP1 (N) is produced. One-half microsecond later, when the oscillator pulse goes high, two lows will be present on AND Gate 0796C4, and the DPO Flip-flop will become reset and DP1 will become set. The DP1 Flip-flop being set, provides a prime to AND Gate 0795D1. On the next low oscillator pulse, TP1 will become reset and TP2 will become set. Thus, Time Pulse 1 existed for one microsecond and upon its termination, Time Pulse 2 began.

All the remaining time pulses are produced similarly. First the TP Flip-flop is set, then the DP Flip-flop is set. The DP Flip-flop then provides a prime for resetting the first TP Flip-flop, and at the same time sets the succeeding TP Flip-flop.

Once TP6 becomes set, DP6 will receive a setting pulse on the next high oscillator pulse (DP5 being reset at the same time). The low output now present from the set side of DP6 will, when coupled with a low oscillator pulse, reset TP6 and set TPO which is the last time pulse in the series. Once again when the oscillator pulse goes high, DPO will become set and if ST(P) is present, nothing else will occur since (TPO · ST) (P) would have set the Stop Flip-flop, and TP1 will not be produced because (ST(P) inhibits AND Gate 0796D5. AND Gate 0796D1 will also be inhibited since STOP (ON) will now be high.

Combination time pulses exist for as long as the title designates. For example, TP23 will commence at TP2 and terminate at the end of TP3 (a total of two microseconds). The problem arising from the possibility of spikes or divisions between time pulses is overcome by using the DP Flip-flops as additional primes to the OR Gates producing the combination time pulses (note OR Gate 079 4B3 and Figure 88). Thus, if a slight division occurs between TP2 and TP3, the combination TP23 will be unaffected.

One additional small point concerning the Time Pulse Generator is that the START light on the Console will be lit when the STOP Flip-flop is set (area 7B). In other words, the light under the start button will be out when the computer is running.

C. STATUS LEVEL GENERATION AND SELECTION

In the 301 System, a status level is a level which exists for seven microseconds and is one of three major factors in computer operation, the other two major factors being operation codes and time pulses. Each status level performs a specific function or functions.

Generation of status levels is somewhat dependent upon the time pulses, in that TP0 begins a status level and TP6 ends a status level, under normal computer operation.

Detailed Logic

Logically speaking, there are two ways of producing a status level. The first is by means of console buttons, which can manually set up a status level in the Status Level Flip-flops. The second way (which is the standard method) is by producing a SELECT level which sets Pretransition Flip-flops whose outputs, in turn, set the Status Level Flip-flops. In either case, the ultimate objective is to set a specific bit configuration into a series of six Flip-flops known as the Status Level (STL) Flip-flops on drawing 3506931, area B (Trng. No. 081). The outputs of these Flip-flops are then decoded into what is known as a Status Level.

Drawing No. 8617058 (Trng. No. 001) is the Console Wiring Diagram. In area B3 through B5 exists the Console STL switches 2^0, 2^1, 2^2, 2^3, 2^4 and parity 2^6. By depressing any of these switches, the corresponding STL 2^x (P) signals will be generated. These signals directly set the STL Flip-flops in the Status Level Generator. (Dwg. No. 3506931, Trng. No. 081). Note also that the Reset Status Level button (RES STL) on the Console produces a positive signal which when present resets the same STL Flip-flops.

As soon as a bit configuration exists in the STL Flip-flops, a Status Level will be generated, provided the Normal Gate (NG) Flip-flop is set. If in the set condition, the NG Flip-flop basically states that an STL is desired for the Processor's Normal Mode. Dwg. No. 3506925 (Trng. No. 076) .

On drawing 3506931 (081) in areas A3 and A5 exist two octal decoders. The first octal decoder (area A3) produces one of eight levels by examining the outputs of the 2^0, 2^1 and 2^2 Flip-flops. The binary configuration in these three Flip-flops is converted into one octal level called STL - 8^0 - 0(N) through STL - 8^0 - 7(N). The other octal decoder in area A5 produces one of four levels, by examining the contents of STL 2^3 and STL 2^4 Flip-flops. The outputs from the second decoder are labeled STL - 8^1 - 0(N) through STL - 8^1 - 3(N). Keep in mind that only one STL - 8^0 level and only one STL - 8^1 level will exist for any given bit configuration in the STL Flip-flops. (Also note that NG is a necessary prime on the 8^1 decoder.)

The STL - 8^0 and STL - 8^1 levels are then sent to Print No. 8617030 (Trng. No. 080) where they are coupled on AND Gates to produce the actual status levels. For example, assume that an operator at the Console first depressed RES STL and then depressed STL 2^3, 2^2, and 2^0. If NG were set, STL - 8^1 - 1(N) and STL - 8^0 - 5(N) would be generated (octal 15). Gate B-0803C3 on Print (Trng. No. B-080) would be primed and an SIO status level would exist.

If the General Reset (GEN RES) button is depressed on the Console, a P1 status level (octal 31) is automatically set up in the STL Flip-flops. (See Print No. 8617058, Console Wiring Diagram, area B4 and B5.)

The octal configurations for the 24 Processor Status Levels are:

STL	Octal Code	STL	Octal Code	STL	Octal Code
A1	03	X4	14	REP1	24
A2	04	SIO	15	REP2	25
A3	05	B	16	M4	27
A4	06	M3	17	P1	31
M1	10	M2	20	P2	32
X1	11	D	21	P3	33
X2	12	STA1	22	P4	34
X3	13	STA2	23	P5	35

The second means of status level generation is slightly more complicated and involves selection of a status level prior to generation. The Computer's very first status level must be supplied manually from the Console. Normally if a program exists in memory and is to be executed, the first status level is a P1.

Once the first status level is executed, succeeding status levels are automatically selected and generated.

A P1 status level generates SEL P2 on Dwg. No. 8617034 (Trng. No. A-084) area A3. This level, in turn, is taken to Dwg. No. 3506931 (Trng. No. 081) and primes OR Gates 0814D6, 0815D3, and 0816C4. At TP5 time, AND Gate 318D1 will produce a resetting pulse, if the Normal Gate is set and Console Button Status Level Repeat (STLR) is not depressed. Note that this resetting pulse occurs at TP5 of the P1 status level, and resets only the Pretransition (STL-PT) Flip-flop and not the STL Flip-flops. Therefore, at TP6 when the P2 status level (octal 32 is inserted in the Pretransition Flip-flops, the P1 status level is still being carried out. At TP0 a signal known as Gate Normal Pretransition (Gate N-PT) is generated, Dwg. No. 3506925 (Trng. No. 076) area A5. The contents of the STL-PT Flip-flops are gated into the STL Flip-flops on Dwg. 081, area B8. Immediately, the STL Flip-flop contents are decoded, and a P2 status level begins.

Subsequent status levels P3, P4 and P5 are similarly produced by first generating an SEL level (to set the Pretransition Flip-flops) and then gating the new status level into the STL Flip-flops at TP0 time, with the Gate N-PT signal.

After staticizing, a level known as END STAT is generated, and this together with the instruction operation code generates a Select (SEL) level for the First Processing Level. (The first status level involved with the actual processing of the instruction.) For example, END STAT and SF (Symbol to Fill) on Gate 0824B2 Dwg. No. 3506932 (Trng. No. 082) produces SEL A2. Thus the FPL for a Symbol to Fill is an A2 (octal 04) status level.

At the end of the First Processing Level, the Computer will automatically select the next status level, depending upon the following factors:

(1) OPERATION CODE
(2) CURRENT STATUS LEVEL
(3) MISCELLANEOUS CONTROL SIGNAL

In the case of the Symbol to Fill instruction, the next status level after an A2 could be another A2 or a P1, depending upon the ABE Flip-flop. Dwg. No. 3506932 (Trng. No. 082), Gate 0823B2 shows the selection of another A2 if ABE is reset, while Gate 346B3, on Dwg. No. 3506934 (Trng. No. A-084), selects a P1 if ABE is set.

A Print breakdown for the SEL levels:

(082) Produces SEL....A1, A2, A3, A4, SIO
(083) Produces SEL....B, D, X1, X2, X3, X4
(084) Produces SEL....STA1, STA2, REP1, REP2, P1, P2, P3, P4, P5, M1, M2, M3, M4

Figure 89 illustrates the selection and generation of status levels.

The development of Gate N-PT and NG is very involved, especially for input-output gear and simultaneity. However, for normal processing one can state that Pre-Normal Request (P-NRQ) is always set except for an input-output instruction Dwg. No. 3506925 (Trng. No. 076) ; therefore, Normal Request (NRQ) will become set at every TP6. With NRQ set and again under normal conditions, GATE N-PT will be developed at TP0; NG will receive a 2-micro-second setting pulse at that time as well.

Odd parity is maintained in the STL Flip-flops to ensure proper status level generation. If more than one SEL level is generated or none at all, a Status Level Error (STLE) will probably occur. In area D2 on Dwg. No. 3506931 (Trng. No. 081) exists the STL parity checker. If bad parity is found in the STL Flip-flops, STLE (N) is generated which is sent to the Alarm Stop logic on Dwg. No. 3506927 (Trng. No. 077) area 4C.

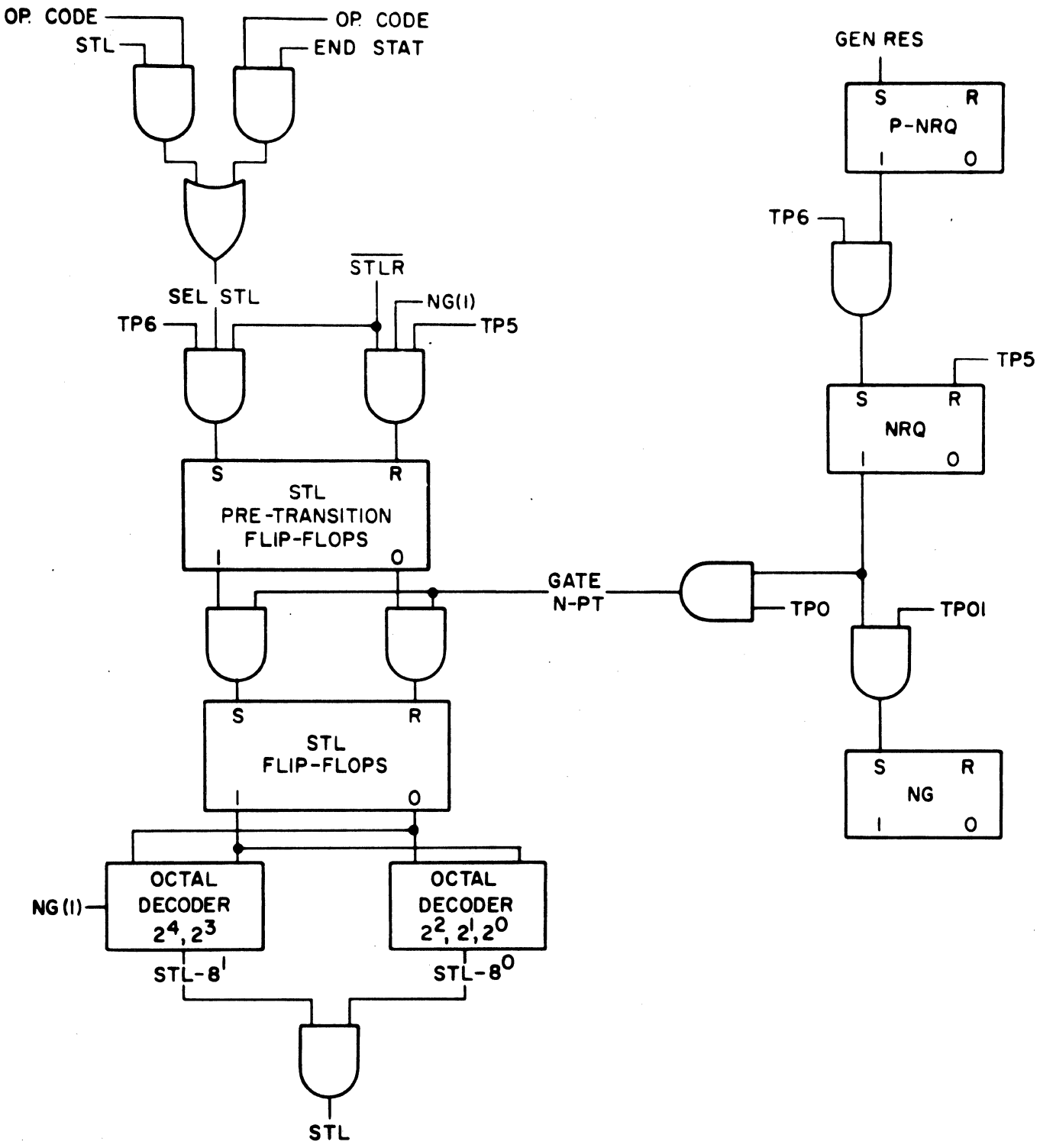


Figure 89 Status Level Selection and Generation

One final point on status level generation is that if the Console Button STLR is depressed, the machine will cycle in the status level which is currently being executed. Setting STLR inhibits changing the STL-PT Flip-flop contents and the Computer cannot recognize any SEL levels. Therefore, the Computer keeps repeating the status level in the STL - PT Flip-flops.

Besides generating status levels, the 301 decodes the contents of the STL-PT Flip-flops to generate a "PT" level. These PT levels are generated across the top of Dwg. No. 3506930 (Trng. No. A-080) and have two advantages over normal status levels. First they exist one time pulse sooner (at TP6) and secondly some PT levels are developed to cover a number of status levels, such as P-PT. The level P-PT will exist for any P status level, P1, P2, P3, P4, or P5. This combination level is advantageous for gating into and out of the P register, since all five status levels use P.

D. NOR AND OPERATION DECODE MATRIX

The Normal Operation Code Register (NOR) exists on Dwg. No. 3506912 (Trng. No. 062) and is composed of seven flip-flops. All flip-flops connect only to Bus 2 positive and negative. The only way to gate the contents from the NOR to the Bus is by selection from the Console (NOR/N SEL). The instruction operation code is gated into the NOR at TP5 of a P1 status level and remains there until the next P1 of a new instruction.

The Operation code used throughout the computer logic comes from a decoding matrix which connects to the NOR. Drawing No. 3506938 (Trng. No. 088) contains a series of AND Gates across the top of the print, which examine the outputs of the NOR Flip-flops. The 2^5, 2^4, 2^3 outputs produce one of eight octal levels labeled C0 through C7, while the 2^2, 2^1 and 2^0 outputs produce one of eight octal levels labeled D0 through D7. The C and D levels are then combined on an AND Gate on dwgs. 3506938 (088) or 3506937 (087) to produce the operation code.

For example, assume a J (Symbol to Fill) is gated into the NOR. The 301 bit configuration for a J is 100 001, excluding parity. Since the 2^5, 2^4, and 2^3

bits are 100, respectively, AND Gate 088 7D3 Dwg. No. 3506938 (Trng. No. 088) is primed and generates the level C4. Gate 0882D2 on the same drawing is also primed by the combination of bits in the 2^2, 2^1, 2^0 Flip-flops (001), and produces the level D1. The two levels C4 and D1 are then combined on Gate 0876B1 Dwg. No. 3506937 (Trng. No. 087) and the output is SF(N) for Symbol to Fill.

On Dwg. No. 3506937 (Trng. No. 087), the reader should note that input-output instructions always develop the normal operation code. For example, a 5 instruction which is a Tape Read Forward Simultaneous, produces a C0 and D5 combination on AND Gate 0873D1. But the output of this AND Gate generates the operation code Read Forward Normal (RFN) just as the 4 instruction does. The reason for this is that if the Simultaneous Mode Inhibit (SMDI) Console Button is set and a simultaneous instruction is staticized, the instruction will be carried out in the normal mode. If SMDI is not set, the instruction is transferred to the simultaneous mode where the simultaneous operation code is developed. Therefore, the normal mode generates a normal operation code for every simultaneous instruction just in case the SMDI button is set.

E. N REGISTER

The main function of the N register on Dwg. No. 3506911 (Trng. No. 061) is to hold the N character of each instruction. However, depending upon the operation code, the N register can be used as a counter, storage device or selector. Because of its counting function, the N register is composed of seven triggerable flip-flops, set up to count down. In other words, if a flip-flop changes from the reset state to the set state, the following stage will be triggered. Therefore, the normal means of inserting a character into a register by resetting all stages first and setting those which should contain one bits, cannot be used in the case of the N register. The reason is, whichever stages went from reset to set would trigger the succeeding stages and the original character would become altered. Thus, when a new character is to be placed in the N register from Bus 3, all stages are first set, then those stages that should hold zero bits are reset.

Gate 0613C1 at TP4 of a P1 is responsible for setting all stages of the N register while inverter 064C2 receives the signal which gates into N at TP 5, of P1 status level from Bus 3. (Note that zeros on Bus 3 (N) are positive signals).

The N register is used as a counter in two different ways. One way is as a straight binary down-counter, while the other way is in effect as a binary coded decimal type down-counter. The Print and Paper Advance instruction used the N register as a straight binary down counter during the X1 status level (Gate 0614C6). However, all of the instructions which use the N count are listed as inputs to OR Gate 0614D2, and the N count is a form of binary coded decimal. The output of 0614D2 goes to two places - one being the same path as the Print and Paper Advance to trigger the 2^0 stage and the other path leads to AND Gate 0614A3, which is involved with adjusting the N count.

When an N count of 30, 20 or 10 is reached in the N register, the computer must produce an N count of 29, 19 or 9, respectively, with one trigger pulse. However, the N count for 30, for example, is binary 110000 (301 Character Quotation Marks) and a trigger pulse will create 101111, which is not the character representing the N count of 29. Hence, an adjustment is necessary.

AND Gate 0614B3 is fully primed when $N\text{-}2^0$, $N\text{-}2^1$, $N\text{-}2^2$, and $N\text{-}2^3$ are all reset. The output of 0614B3, in turn, primes Gate 0614A3 and at TP2 is an instruction using the N count is being executed, the N Count Adjust Flip-flop (NCA) will become set. At TP3, the N register is triggered down, and with NCA set, Gate 0615D1 at TP4 will generate an output which resets the 2^2 and 2^1 stages. Thus if N was triggered from 110000 to 101111 during an instruction which was using the N count, NCA would have become set and the end result would be 101001, or the proper combination for an N count of 29. Note that dropping two bits does not change parity.

The ultimate goal in using N as a down-counter is to reach a zero count. AND Gates 0614B3 and 0615A1 detect when all stages of N are reset; they in turn generate the level N Equals Zero (NZ).

Parity correction takes place when N is used as a counter. Gates 0618C1, 0618C2, 0618C3 and 0618C4 handle the correction of parity if an output exists from Gate 0618C4. Note that correction occurs one time pulse prior to the triggering down. In area D2 exists a parity checker which checks parity at all times in the N register. However, bad parity must exist at TP6 time, to stop the Computer on an alarm. (Dwg. No. 3506927 (Trng. No. 077) area 6D.)

When N is used for temporary storage, no triggering occurs. In area 2C of Dwg. No. 3506971 (Trng. No. 061) are the three gates which permit gating the N character onto Bus 3, and listed on two of these gates are the instructions which use N for temporary storage: Symbol to Fill on Gate 0612C2; Locate Symbol Left/Right on Gate 0613C3; and Transfer Data by Symbol Left/Right on Gate 0613C3.

The N register is used as a selector in several instructions and for the majority of those, the individual flip-flop outputs are examined. In two instances, there are special levels generated. One of these is the level NA which is produced if the 2^1 Flip-flop is set (area A4). The primary function of NA is during a Store Register instruction when N = 2, denoting the storage of the A register contents in STA. During a Store Register instruction, NA selects the STA1 status level after END STAT.

The other special selection level is NS in area B4. NS is generated when N = 8 or when the 2^3 Flip-flop is set. Again the NS level is used during the Store Register instruction, for storing the contents of the S register. The S register, however, exists in the simultaneous mode. Therefore, if the Console Button SMDI is depressed, all simultaneous instructions will be executed in the normal mode, and the S register will not be used. Since the S register corresponds to the A register in the normal mode, the level NA will be generated when N = 8 and SMDI is set.

F. ADDRESSABLE REGISTERS

In the 301 Processor itself, there are three addressable registers; namely the P register, the A register, and the B register. Each register has the function of holding an address, therefore, it is capable of storing four characters. All three registers connect to the common Bus and each one can gate its contents onto the Bus, as well as receive characters from the Bus.

1. P Register

The function of the P register is for program control, or, to hold the address of the next instruction to be executed. The logic of the P register exists on Dwg. No. 3506901 (Trng. No. 051) and is composed of 21 Flip-flops. For the 10K and 20K Processor there should never be a 2^5 bit in any character of any normal address. Nor should the 2^4 bit exist in the P1, P2, or P3 characters. The 2^4 bit is necessary in PO, however, for addresses over 10 thousand. Therefore, the P register contains six flip-flops for the PO character, and five for each of the characters P1, P2, and P3. To gate into or out of the P register, or to reset the P register, a control level must be generated on Dwg. No. 3506902 (Trng. No. 052). Bus to P is generated in area A6, Reset P (RES P) is generated in area A5 and P to Bus is generated in area, during a specific status level and time pulse.

2. A Register

The function of the A register is to hold the A address of each instruction during the execution of that instruction. Dwg. No. 3506904 (Trng. No. 054) contains the 22 flip-flops which make up the A register. The one additional flip-flop that is found in the A register and not in the P register is the A3 character 2^4 bit. This flip-flop is necessary to handle indirect addresses.

Just as with the P register, the A register is controlled by three basic levels which are Bus to A, Reset A, and A to Bus. Dwg. No. 3506905 (Trng. No. 055) contains the A register controls. (See areas A6 and A4)

3. B Register

The function of the B register is to hold the B address of each instruction. Like the A register, B is composed of 22 flip-flops with a 2^4 stage in B3, to handle indirect addresses. Also similar to the A and B registers, the logic which controls the B register exists on Dwg. No. 3506906 (Trng. No. 057). One difference is that the resetting level is split up for the Print and Paper Advance instruction where it is desired to reset B3 alone. (See areas A7, A5, A4 and A2 for the control levels on Dwg. No. 057.)

Note that no parity checkers exist for the addressable registers. It is therefore possible to gate into a register, drop one or more bits and never detect the loss. This is only true, however, if the address never reaches the MAR to address memory since a parity checker does exist in the MAR. The majority of instructions do address memory with the A and B addresses at one time or another.

G. BUS ADDER

The purpose of the Bus Adder is to modify four given characters by plus one, minus one, plus two, or minus two. In some instances no modification at all takes place. The four characters to be modified must come from the MAR and in all but one case, represent an address. The only exception is in a TALLY instruction in which case the Bus Adder is used to reduce the tally quantity.

The Bus Adder is divided into four stages - C0, C1, C2 and C3 - which correspond to the characters in the MAR. The outputs from the MAR are taken directly from the set and reset sides of the flip-flops, and are fed to the Bus Adder by bits. The Bus Adder provides for information bits (2^0, 2^1, 2^2, and 2^3) in all four stages but only makes provision for a 2^4 bit in the C3 and C0 stages. No provision is made for a 2^5 bit in any stage.

To perform any modification, the Bus Adder needs control levels which are shown on Dwg. No. 3506917 (Trng. No. 067). These control levels consist of modifying levels BA(-1), BA(+1), BA(-2) and BA(+2) as well as the gate-out levels BA01-Bus and BA23-Bus. The Modifying levels are used for internal

operation and the gate-out levels are used to place the modified result on the bus at a specific time. The original modifying levels (BA(-1), etc., are used only in the C3 stage of the Bus Adder (Dwg. No. 3506923) (Trng. No. 073). These levels will then give rise to carry and borrow levels, which will become modifying levels for subsequent stages. The Bus Adder and connecting logic is shown in Figure 90.

The most complicated stage in the Bus Adder is the C3 state since the majority of modifications will affect only the least significant digit of an address. However, all stages of the Bus Adder have a great deal in common. Each stage generates carry and borrow levels abbreviated CAR and BOR. These two levels signify a decimal carry or decimal borrow to the succeeding stage. They also are involved in converting the binary output of the adder into decimal form for each stage when the result is gated onto the Bus lines.

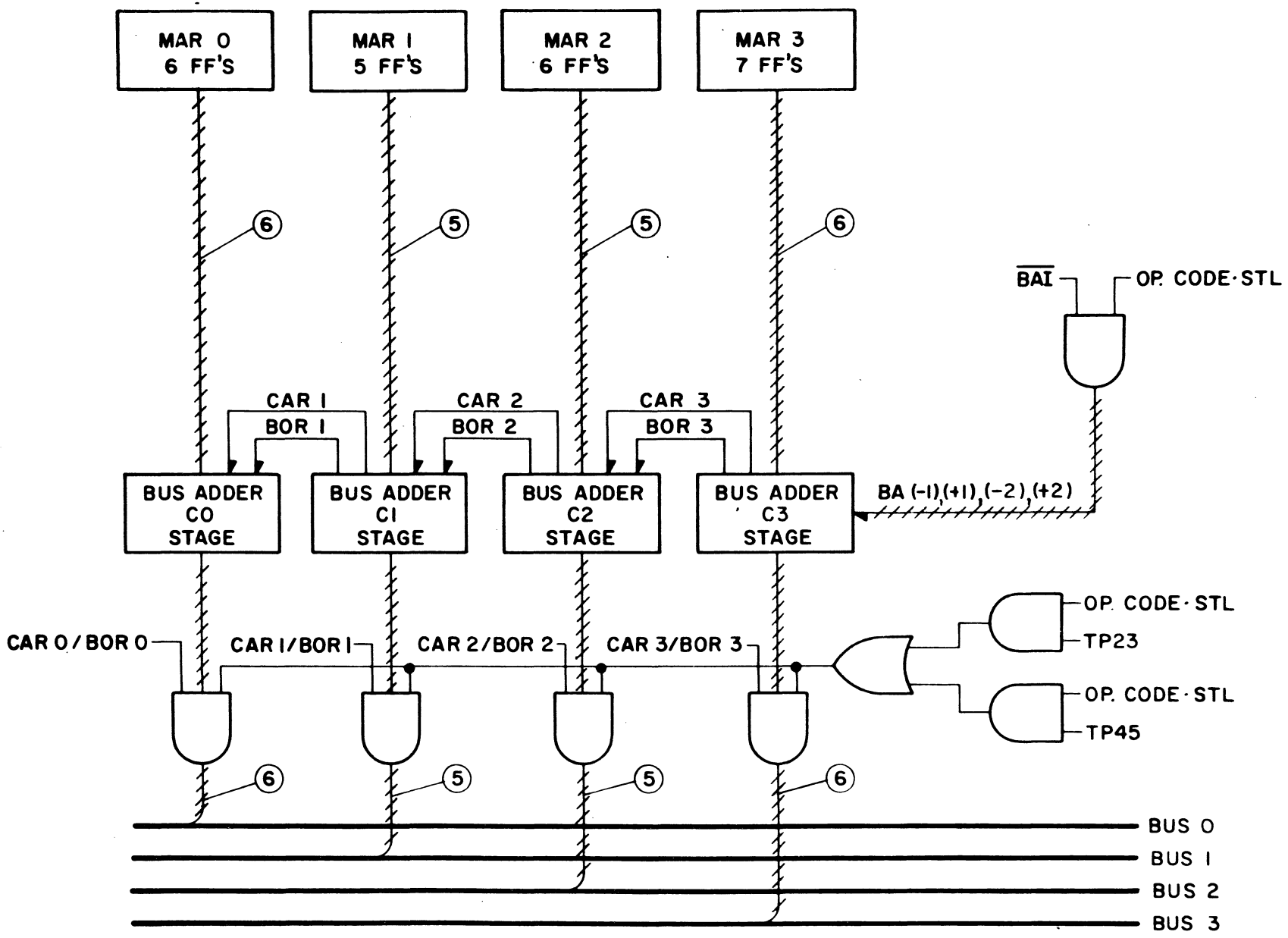


Figure 90 301 Bus Adder

For the C3 stage on Dwg. No. 3506923 (Trng. No. 073) in area D2 and D3 exist the gates which give rise to the CAR 3 and BOR 3 levels. Basically speaking, Gate 0732C1 will produce a carry to the C2 stage when adding a one to a decimal nine and Gate 0732C3 will generate a carry when adding two to eight or nine. AND Gate 0732C5 causes a borrow from the C2 stage when subtracting one from zero, while Gate 0731C1 produces BOR 3 when two is subtracted from zero or one. Producing CAR 3 or BOR 3 also generates the combination level CAR 3/BOR 3.

Because the C3 stage is somewhat more involved, a portion of the C2 stage will be extracted in order to explain how binary addition and subtraction is performed. The C2 stage is on Dwg. No. 3506922 (Trng. No. 072).

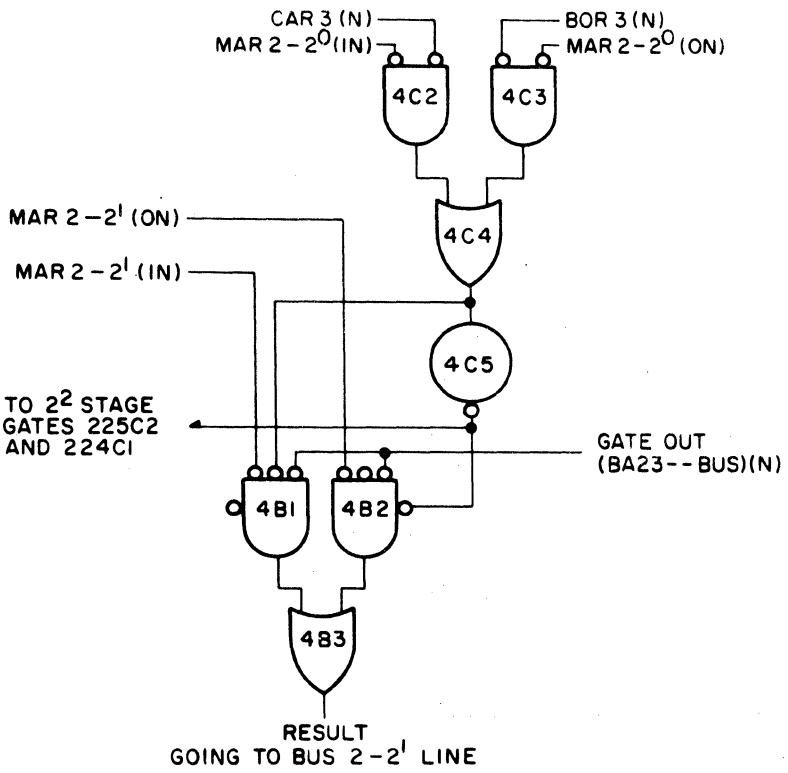


Figure 91 Logic Extract From Bus Adder C2 Stage

AND Gate 4C2 will produce a high output if the 2^0 bit of the C2 character is a one and a decimal carry was generated from the C3 stage. AND Gate 4C3 will produce a high output if the 2^0 bit of C2 is a zero and a decimal borrow was generated from the C3 stage. Note that regardless of the incoming bit for the 2^1 stage of C2, if a carry or borrow is generated from the 2^0 stage of C2 (high output from 4C4), the result for the 2^1 stage will be the complement of the bit coming in.

Incoming 2^1 bit	0	1	0	1
Modification (From 2^0 bit)	+1	+1	-1	-1
Resulting 2^1 bit	1	0	1	0

Effectively then, the output of 4C4, if high, states that an interbit carry or borrow exists from the 2^0 stage to the 2^1 stage of the C2 character. Therefore, depending upon the state of the 2^1 Flip-flop the complement will be generated onto Bus 2-2^1 line at gate-out time. (Bear in mind that the decimal conversion is ignored for the present.)

For example, if the 2^1 bit is a one and the output of Gate 4C4 is high, Gates 4B1 and 4B2 would both be inhibited, and no positive pulse is gated onto Bus 2-2^1 line when gate-out occurs. (This represents a zero bit result). If the 2^1 bit is a zero, initially, Gate 4B2 will be primed to generate a one bit onto Bus 2-2^1 when the result is gated out.

On the other hand, if the output of 4C4 is low, this indicates no carry or borrow from the 2^0 stage, hence no change in the 2^1 output. That is, if the 2^1 is a one, Gate 4B1 will produce a high output at gate-out time to generate a one on Bus 2 and if the 2^1 bit is a zero neither 4B2 nor 4B1 will be primed, thus no one bit is generated on Bus 2.

The output from 4C4 also is used to determine whether or not a carry or borrow is generated from the 2^1 stage to the 2^2 stage (Gates 0725C2 and 0724C1 on Dwg. No. 072).

Examination of the logic of the Bus Adder C3 stage on Dwg. No. 073 reveals a similar composite of gates, such as those used in the C2 stage. A slight

difference exists in the fact that the original modifying levels (BA+1 etc.) are used and also that the 2^1 logic must handle the addition and subtraction of 2.

The 2^0 bit is complemented if the output from Gate 0733B1 is a low, denoting modification by +1 or -1. If the output from 0733B1 is high, the 2^0 bit is gated unchanged onto Bus 3-2^0 line.

Adding or subtracting two is done by priming Gate 234B1 with the levels BA(+2) and BA(-2) to act as a carry and a borrow, respectively, for the 2^1 bit.

The second function of the CAR and BOR levels other than acting as decimal carries and borrows between the four stages C0, C1, C2, and C3; is that they convert the output of the Bus Adder into decimal notation.

Note that adding one to a decimal nine as the C3 character, would generate CAR 3. Binary addition would be as follows for the 2^0 through 2^3 bits.

C3 Character	=	1001	=	9
Modified by	=	+0001	=	+1
Result		1010		(Binary Coded Decimal 10)

The result should be zero for the C3 stage therefore CAR 3 must inhibit the output gates for the 2^3 and 2^1 bits to produce 0000 on the Bus. (See Gates 6B2, 6B3, 3B3 and 3B4, on Trng. No. 073).

Adding two to a nine or eight causes the generation of CAR 3. Binary bit addition is:

C3 Char.	1001	= 9		1000	= 8
Modified by	+0010	= 2		+0010	= 2
Result	1011			1010	

Once again, the CAR 3 level inhibits the output from the 2^3 and 2^1 stages since a one (0001) is desired when adding 2 to 9 and a zero (0000) is desired when adding 2 to 8.

BOR 3 is generated when one is subtracted from zero or two is subtracted from zero or one. Binary subtraction would be:

C3 Char.	0000	= 0	C3 Char.	0000	= 0	C3 Char.	0001	= 1
Modified by	-0001	= -1	Modified by	0010	= -2	Modified by	0010	= -2
Result	1111		Result	1110		Result	1111	

Since the results should be nine, eight and nine respectively, BOR 3 inhibits the 2^1 and 2^2 stage outputs. (See Gates 5B2, 5B3, 4B3 and 4B4, on T ng. Dwg. 073).

The 2^5 bit is not provided for in any stage of the Bus Adder and the 2^4 bit is simply gated out unchanged for the C3 stage (Gates 0737B1 and 0737B2). The presence of a 2^4 bit in the output of the C3 stage would signify an indirect address under normal conditions. However, the 2^4 bit for stages C2 and C1 is not permitted. The 2^4 bit is permitted for the C0 stage, to represent addresses over ten thousand.

Each stage generates its own parity as needed by examining the original character and the modification level for that stage.

Depending upon the model, the C0 stage of the Bus Adder (Dwg. No. 3506920) will generate addresses over ten thousand or inhibit the generation of such addresses by the presence of three +6.5 volt levels on Gates 0706B4, 0706B3, 0706C2 and 0707C3. In all cases, the Bus Adder will always keep within the limits of the memory. For example, adding one to 9999 in a 10K system produces 0000, while in a 20K system &000 will be produced. Likewise, adding one to I999 in a 20K system will produce 0000. Subtracting one from 0000 gives a result of 9999 for the 10K system and I999 for the 20K system.

H. MEMORY REGISTER AND INTERCHANGE

The 14 flip-flops which make up the Memory Register are shown on Dwg. Nos. 3506913 (Trng. No. 063) and 3506914 (Trng. No. 064).

The MR is divided into two sections, MR0 and MR1, to correspond to the two characters in a diad which are labeled C0 and C1. The MR0 portion of the

Memory Register connects to Bus lines 0, 2 and 3 while the MR1 portion connects to Bus lines 1, 2 and 3. The control levels which set up the paths for information flow between the Bus and the MR come from a series of gates known as the interchange on Dwg. No. 3506915 (Trng. No. 065). Also shown on Dwg. 065 are the reset levels for the MR (area A2) and the two parity checkers (area D5).

At the bottom of Dwg. Nos. 063 and 064 are shown the light drivers for the common display lights on the Console. Any voltage levels on the four Bus lines will light the corresponding lights on the Console. Thus if it is desired to view the contents of the particular register, the proper select switch is set and the contents of the selected register are gated onto the Bus where they light their respective lights.

The P, A and B registers use all four Bus lines, but the MR and D register use only Bus 2 and Bus 3, to display their contents. The NOR contents are shown by way of Bus 2 and the N register contents are shown by way of Bus 3.

The select switch, when set, not only gates out of the selected register onto the Bus but also permits gating in from the Bus. Hence to insert new information into a register, bits are inserted on the Bus while the select switch is set. The common display lights on the Console are also momentary contact switches. On Dwg. Nos. 063 and 064, the signals generated by these switches are shown as inputs to the AND Gates below the MR Flip-flop; (CO - 2^0 SET(N) on Gate 064782 etc.). Depressing the proper Console switches produces momentary voltages on the corresponding bus lines. These voltage (bits) are then gated into the flip-flops of the selected register. When the Console switches are released, the lights will remain lit for those flip-flops which were set, since the select switch is still gating out of the selected register. If a flip-flop does not become set, the light on the Console will be lit only for the duration of the time that the momentary contact switch is making contact.

I. NR REGISTER

The N for Repeat register, abbreviated NR, exists on Dwg. No. 3506916 (Trng.

No. 066). The NR register is only used during the Repeat instruction to hold the count of the number of times to repeat. Five triggerable flip-flops, including parity, comprise a down counter. Since the register is set up to count down, all stages must be set first (TP2 of an X1 status level of a Repeat instruction Gate 0663C1) and those stages which are to hold zeros will be reset according to the contents of the N register (TP3 of X1 RPT - Gate 0665D1). Note that only the 2^0, 2^1, 2^2 and 2^3 bits are gated into the NR from the N register and that if the 2^4 or 2^5 bits exist in N, they will be dropped upon transfer into the NR. Thus, if one bit is dropped, bad parity will result and an NRPE will be generated from the parity checker, in area B3. If both the 2^5 and 2^4 bits are dropped, the parity remains good and no alarm occurs.

The NR is triggered down at TP3 of an REP1 status level and parity is corrected each time. The end result is to trigger down to zero, and Gate 0664B1 will produce NRZ (NR equals zero) when this occurs.

The three flip-flops on the right-hand side of the print (INHA, INHB and FREP) are used during the repeating process.

J. D REGISTER

The D register can hold one or two characters for temproary storage. One half of D, called D2, connects to Bus 2 and the other half of D, called D3, connects to Bus 3. The D2 portion of the register is shown Dwg. No. 3506908 (Trng. No. 058) while the D3 portion exists on Dwg. No. 3506909 (Trng. No. 059).

The D2 register is composed of seven normal flip-flops whose outputs can be gated only onto Bus 2 (P). Information is normally inserted into D2 from Bus 2 (N), however, during the Translate instruction, bits from Bus 3 (N) 2^5, 2^4 and 2^3 lines can be gated into D2 2^2 , 2^1 and 2^0, respectively. (Gates 0583B1, 0583B2, and 0582B2) The parity checker for D2 exists in area D3. Parity is checked on D2 only during certain status levels of certain instructions. Gates 0584D1 and 0583D2 are mainly responsible for determining when parity should be checked. If bad parity is found, a low will

exist from the output of the D2 parity checker and Gate 0583D6 will set the DPE Flip-flop in area A8.

The D3 register is composed of five triggerable flip-flops (2^0, 2^1, 2^2, 2^3, and 2^6) and two normal flip-flops (2^5 and 2^4). The triggerable flip-flops constitute an up-counter which is used during the Add and Subtract instructions. Otherwise D3 is used as a storage register for one character. The D3 register can only be accessed from Bus 3 whether it is during gating in or gating out. Parity is also checked on D3 during certain instructions; its parity checker exists on Dwg. No. 3506906, area D2.

On Dwg. No. 3506910 (Trng. No. 060) a number of controls exist which affect the D3 register. The majority of these control levels are for the Add and Subtract instructions.

K. D COMPARATOR

The purpose of the D Register Comparator is to compare the contents of the D2 register to those of the D3 register, and to produce an appropriate level (D2 $>$ D3, D2 $<$ D3, or D2 = D3). Since no time pulses or instructions gate the comparator logic, one of the three output levels should exist at all times. If no output level exists, a Comparator Error (COME) will be generated.

On Dwg. No. 3506918 (Trng. No. 068), the outputs of both the D2 Register and the D3 register flip-flops are compared directly to one another, bit by bit. In the series of AND Gates in area D (3, 4, 5, 6 and 7) exact equality is sought.

Both bits must be present or absent for a given position in D2 and D3 to produce a desired output. All outputs culminate on AND Gate 0682B1. If six lows are present, the level D2 = D3 is generated (area A2). If six lows are not present, then obviously one register's contents are greater than the other's. Allowance had been made to make use of what equalities were found, if not all six. The series of AND Gates in area B (5, 6 and 7) check all possibilities of D2 being greater than D3. First, the most sig-

nificant bits are compared on AND Gate 0687B1. If D2 2^5 is present and D3 2^5 is not, a high output is produced, which will generate the level D2 > D3, area A7. If the 2^5 bits from D2 and D3 are the same, AND Gate 0687B2 compares the 2^4 bits and so on. Any high output from this series of AND Gates will produce the level D2 > D3.

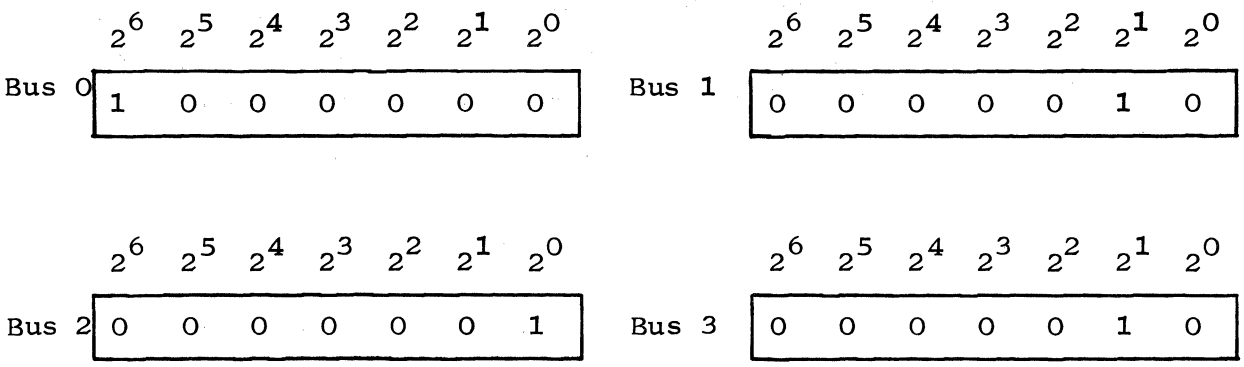
In area B (2, 3 and 4), bit-by-bit comparison is made to determine if D2 < D3. Any equalities found through the first series of AND Gates in area D (3, 4, 5, 6 and 7) are used as inputs to these gates.

If no final result is obtained or if more than one result is obtained, the error level COME(N) is produced from Gate 0686A3. AND Gate 0686A1 checks for the level D2 > D3 existing and the other two levels not existing. If this is ture, COME(N) will be high, and therefore does not indicate an error. AND Gate 0686A2 checks for D2 < D3 existing with the other levels absent. AND Gate 0684A1 checks for D2 = D3 existing, but not the other two levels. The reader should note that each of these three gates, 0686A1, 0686A2 and 0684A1, must be inhibited if COME(N) is low. Only in case no output levels are produced from the comparator, or more than one level is produced, will this condition be brought about.

L. STANDARD ADDRESS GENERATOR

The Standard Address Generator is found on Dwgs. 074 and A074. All 3 models of the 301 Processor utilize this generator. The function of this generator is to generate bits on the bus which will constitute an address or a portion of an address. This will depend upon the status level and the instruction.

For example, during the STA1 status level when the A0 and A1 characters of the A address are to be stored in memory, the address 0212 is generated (Dwg. No. 3506029, area C2) and the level STA1 - PT(P) primes Gates 293A1, 295A2, 297D1, 297A1, and 298A1. The outputs of these gates at TP01 time produce positive voltage signals on Bus 3 - 2^1, Bus 2 - 2^0, and Bus 0 - 2^6. On all other lines of the Bus there are effectively zero bits. Therefore the bit configuration for each Bus line, at TP01 of STA 1, would be:



This represents the numbers 0212 respectively.

The only difference between the Model 303A Generator and the Model 304A Generator is in the X1 or Y1 status level of a Print and Paper Advance Instruction. The PAN/PAS instructions use a table look-up technique to obtain the 301 Character Code of the character appearing on the drum about to be printed. In a 10K System this table ranges from 9900 to 9977, but in a 20K system the table exists between I900 and I977. Therefore, during the X1/Y1 status level of a PAN/PAS instruction, the 303A Model generates 99 onto Bus 0 and Bus 1, respectively. (Gates B0746A1, 6A2, 6A3, 5A1, 5A3, 4A2). The 304A Model generates I9 on Bus 0 and Bus 1, respectively, during the same status levels (Gates B0746A1, 6A2, 6A3, 5A1, 5A2, 5A3).

1. Signals Generating Complete Addresses	Numbers Generated
S - (0278) -- Bus (P)	0202
S - (0280) -- Bus (P)	0204
CPN/BCPN (N) • X1-PT/X3-PT (P)	0202
CPN/BCPN (N) • X2-PT/X4-PT (P)	0204
Add/Sub (N) • X1-PT/X3-PT (P)	0206
Add/Sub (N) • X2-PT/X4-PT (P)	0208
STA 1 - PT	0212
STA 2 - PT	0214
X1 - PT (N) • CTC/TA/IOS (P)	0216
X2 - PT (N) • CTC/TA/IOS (P)	0218

Signals Generating Complete Addresses (cont'd)	Numbers Generated
(X1-PT) (N) • RPT (N) / REP 1-PT (P)	0222
(X2-PT) (N) • RPT (N) / REP 2-PT (P)	0224

2. Signals Generating Part of Addresses - Bus 0 and Bus 1

ULS (1N) • Add/Sub (N) • D-PT(N)	-1--
ULS (ON) • Add/Sub (N) • D-PT(N)	-0--
X3 - PT • TA (N)	00--
D - PT • (Add/Sub) (N)	0---
X1 - PT • PAN (P) (303A)	99--
X1 - PT • PAS (P) (303A)	99--
X1 - PT • PAN (P) (304A)	I9--
X1 - PT • PAS (P) (304A)	I9--

M. STOP ALARM LOGIC

The Stop Alarm Logic exists on two prints within the Processor; Dwg. Nos 3506927 (Trng. No. 077) and 3506928 (078).

The function of the Stop Alarm Logic is to stop the Computer upon: (1) a Halt instruction, (2) a Console button manual stop or (3) an alarm. To accomplish this, the level ST(P) must be generated to inhibit the Time Pulse Generator from functioning, and to set the Stop Flip-flop (Trng. Dwg. 079, area B7). ST(P) is generated from Gate 0786B1 on Dwg. No. 078 in one of two ways: if the Manual Stop (MSP) Flip-flop is set which covers functions (1) and (2) above, or if Error Stop (ERSP) is generated which covers function (3).

A Halt instruction can only stop the Computer after the instruction has been completely staticized, i.e., at the end of the P5 status level. In area C-7, HLT(P), the operation code, energizes OR Gate 0787D1 and only if P1-PT is present can an output be obtained from AND Gate 0787C1. The presence of P1-PT denotes that staticizing has been completed. This output from 0787C1 will in turn generate the level MSTP which means machine stop. The level MSTP

alone cannot stop the Computer. A check must first be made on the other modes - Simultaneous and Record File. Only if these modes are not busy or an alarm exists in them, will the level MSTP be permitted to halt the Computer (AND Gate 0786C1). Once this check has been made, MSTP will set the MSP Flip-flop at TP6, thereby producing ST(P).

Two other manual stops which produce the level MSTP are Console buttons FPLS and ICSP.

FPLS stands for First Processing Level Stop and will stop the Computer after staticizing and indirect addressing are complete. ICSP means Instruction Complete Stop and will stop the Computer once P1 - PT is generated, i.e., the present instruction has been completed and the next instruction is ready to be staticized.

Three other Console buttons which set MSP but do not generate the level MSTP are OCSP, RDM and WRM. All three buttons stop the Computer after one series of time pulses without checking other modes. Thus, these buttons could be called immediate stops. In fact, One Cycle Stop (OCSP) is used as an emergency stop button since it stops the computer at the end of the next status level. Read From Memory (RDM) and Write to Memory (WRM) are not used to stop the Computer normally, but do permit only one series of time pulses to occur.

All modes of stopping the Computer by setting MSP also permit the Computer to be started again by simply depressing the start button (ST RES, Dwg. No. 079). Error stops, however, will constantly produce ST(P) as long as the alarm is present. Thus, if the Computer stops on an alarm, it cannot be started again until the alarm is reset.

There are two general types of alarms; namely, immediately stop alarms and delayed stop alarms. The immediate stop alarms are Memory Address Register Parity Error (MAPE) and Memory Register Parity Error (MRPE). These alarms will stop the Computer regardless of what is occurring in other modes. However, a certain restriction does exist. These alarms must occur when Memory is in use, since the Command Level Flip-flop provides a necessary

prime (area B-2) in stopping the Computer. Either one of these alarms will then stop the Computer upon the next TPO.

The delayed stop alarms are all of the remaining alarms in the Computer. Each mode (Normal, Simultaneous and Record File) has characteristic alarms, but no one alarm will stop the machine until the other two modes are free, unless an alarm also exists in those modes. Immediate stop alarms, MAPE and MRPE, are involved with a portion of the Computer which is common to all three modes. If the Memory Address Register or Memory Register contains bad parity, the Computer must stop immediately, since all three modes are effectively tied up. However, if only a register parity error or user equipment error occurs in one mode, the other two modes are still free to carry on and will until they finish their operation or until they, too, develop an alarm.

AND Gate 0785C1 will produce an output if a Normal Mode error exists, AND Gate 0784C2 will produce an output if a Simultaneous Mode error exists and AND Gate 0783C4 will produce an output if a Record File Mode error exists.

In addition, if a Simultaneous alarm occurs, the SAL light becomes lit on the Console. This is to indicate that a peripheral device alarm (OR Gate 0785D4) or the Status Level Error (STLE) occurred in the Simultaneous Mode and not in the Normal Mode or Record File Mode. On the other hand, the FAL light becomes lit for a Record File alarm to distinguish the fact that the Record File Mode caused the peripheral equipment alarm, and not the Normal or Simultaneous Mode. And if neither SAL nor FAL are lit, the error occurred in the Normal Mode. With the exception of MAPE and MRPE, the alarm flip-flops and associated lights are shown on (Trng. Dwg. No. 077).

Any mode alarm, except MAPE and MRPE, must check the other modes first before stopping the Computer by way of OR Gate 0783C5 on Dwg. No. 078. It is possible to inhibit alarm stops by the Console Button Alarm Inhibit (ALI) on Gate 0785B1. However, in the case of most alarms, the ALI button does not inhibit the Alarm light from becoming lit but only inhibits stopping the Computer.

RCA Information Systems

Computer Systems Division

Field Engineering Technical Education