Property of
H. C. Kreide

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

LOGICAL DESIGN OF CG24
(A GENERAL-PURPOSE COMPUTER)

G. P. DINNEEN
J. A. DUMANIAN
I. L. LEBOW
I. S. REED
P. B. SEBRING

15 APRIL 1957

TECHNICAL REPORT NO. 139

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

LOGICAL DESIGN OF CG24
(A GENERAL-PURPOSE COMPUTER)

G. P. DINNEEN
J. A. DUMANIAN
I. L. LEBOW
P. B. SEBRING
Group 24

I. S. REED
Group 47

TECHNICAL REPORT NO. 139

15 APRIL 1957

ABSTRACT

A detailed design is presented for a high-speed general-purpose digital computer. The design considerations are governed by the assumption that implementation of the machine is to be accomplished using only solid state devices. Sections I through V describe the essential characteristics, structure and method of design of the computer. Sections VI through IX discuss its detailed logical structure.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY LEXINGTON, MASSACHUSETTS

GLOSSARY

( )	The parentheses denote the contents of a register. Thus (A) represents the contents of A.
[ ]	The square brackets denote a functional dependence upon a register. Thus I[A] signifies the instruction part of A.
< >	The angle brackets denote a selection of one register out of a set of registers depending upon an address register. Thus M<C> signifies the memory register depending upon or as addressed by C.
{ }	The braces are used for algebraic punctuation wherever parentheses, brackets or braces would normally be used.
$\vert\;\vert$	Vertical bars denote a set of configurations of the computer for which some function is true. Thus $\vert a'f_{21}P_2\vert$ signifies the set of configurations for which the function $a'f_{21}P_2$ is true.
+	The plus sign has two meanings: (a) When used between Boolean functions it means the "inclusive or" or "join" operation. Thus $\alpha + \beta$ is a function that is true when either α or β (or both) is true. (b) When used between pairs of parentheses it signifies the ordinary addition operation. Thus (A) + (R) represents the arithmetic sum of the contents of the A- and R-registers.
–	The bar denotes the subtraction operation. Thus (A) – (R) represents the arithmetic difference between the contents of the A- and R-registers.
$\oplus$	The plus sign enclosed by a circle denotes the "exclusive or" or binary sum operation. Thus $\alpha \oplus \beta$ is a function that is true if α or β (but not both) is true.
$\Rightarrow$	The double arrow denotes a transfer of information between registers. Thus (A) $\Rightarrow$ B signifies that the contents of A are transferred into B.
'	The prime represents the complement of a Boolean function. Thus α' is the complement of α.
,	A comma between register symbols denotes that the registers are to be considered as a single register ordered as written. Thus A, B is a single register with A to the left of B.
$(\bar{\ })$	The horizontal bar over a symbol in parentheses denotes the "one's complement" of the contents of a register. Thus $(\bar{A})$ signifies the "one's complement" of the contents of A.

; $\overline{}$	The semicolon followed by a horizontal bar over a symbol signifies the exclusion of the symbol under the bar from the symbol to the left of the semicolon. Thus $A; \overline{A_o}$ represents the A-register exclusive of bit A_o.
$\cdot$	The dot represents binary multiplication. Thus $\alpha \cdot \beta$ is a function that is true if both α and β are true. Usually the dot is omitted entirely and $\alpha \cdot \beta$ appears as $\alpha\beta$.
Σ	The summation sign represents the "inclusive or" or join operation between the functions included under the sum. Thus $\sum_{i=1}^{4} A_i \equiv A_1 + A_2 + A_3 + A_4$.
$\oplus \Sigma$	The summation sign preceded by the circle sum symbol represents the "exclusive or" operation between the functions included under the sum. Thus $\oplus \sum_{i=1}^{4} A_i \equiv A_1 \oplus A_2 \oplus A_3 \oplus A_4$.
Π	The product sign represents the product or "and" operation between the functions included under the product. Thus $\prod_{i=1}^{4} A_i \equiv A_1 \cdot A_2 \cdot A_3 \cdot A_4 = A_1A_2A_3A_4$.
$\hat{j}$	The subscript on a register symbol specifies a particular bit of a register. Thus A_3 is bit 3 of the A-register. In the computer 25-bit registers are labeled 0 to 24 from left to right.
$\underset{\vee}{k}$	The superscript on a register symbol specifies a particular register or subset of registers from a set of registers having the same symbol. Thus S^3 is the third index counter and M^o is the first core-memory bank.
a	The bit F_{12} used to designate instruction read-in cycles.
A	The accumulator.
A_{OF}	The overflow flip-flop.
Ad[N]	The address section (bits 10 to 24) of a register, here N.
B	The B-register, an extension of the accumulator.
BB	The busy bit controlling terminal equipment.
B[N] ✗	The index section (bits 0 to 4) of a register, here N.
C^k	The memory address registers. When used without superscript, there is implied a parallel transfer into all C-registers.

GLOSSARY (Continued)

Cm	The control memory.
D	The program counter.
E	The one-bit arithmetic register.
f_j	The j-th proposition from the F-register.
F	The control-memory output register.
G	The control-memory address register.
G[F]	The address section (bits 1 to 6) of F.
H	The input-output register.
HA	A front-panel toggle-switch register specifying a halt address.
H[B]	Bits 0 to 5 of the B-register.
I[N]	The instruction section (bits 4 to 9) of a register, here N.
J_k	The start-stop control flip-flops.
L^k	The core-memory input registers.
M, M^k	The memory; the k-th memory bank.
n	A number when stored in the address part of a word.
N^k	The core-memory output registers.
$O_k[F]$	The nine 3-bit sections of F, $O_1[F]$ to $O_9[F]$ comprising bits 13 to 39.
P_k	The basic timing intervals $P_1 - P_4$.
P_{Ad}	The address parity bit.
P_{BI}	The index-instruction parity bit.
Q	The start-stop flip-flop.
R	An arithmetic register holding operands.
S^k	The index counters.
s_k	The computer clock pulses $s_1 - s_4$.
Shl A	A register, here A, shifted left by one bit. A zero is inserted in the rightmost digit unless otherwise specified.
Shr A	A register, here A, shifted right by one digit. A zero is inserted in the leftmost digit unless otherwise specified.
SW_c	A proposition true if computer operates for one memory cycle and then halts.

GLOSSARY (Continued)

SW_H	A proposition true if computer is to stop at address in HA-register.
SW_I	A proposition true if identity alarm is suppressed.
SW_{OF}	A proposition true if overflow alarm is suppressed.
SW_1	A proposition ~~time~~ true if computer operates for one order and then halts.
T	The counter in control.
U^c	A register in display holding a number to be displayed.
U^s	A counter in display.
U^x	The horizontal deflection register in display.
U^y	The vertical deflection register in display.
V^k	The index criterion registers
W^k	The three flip-flop registers in M^2.
X	A memory location.
$\times$	The multiplication algorithm.
Y	A variation of $\times$.
Z	The divide algorithm.
Z_A, Z_B	Variations of Z.
α	The addition proposition.
Γ or $\Gamma[C^2]$	Bits 11 and 12 of C^2 used to address the memory banks and their associated registers.
λ	The control advancing function.
Λ_{BI}	A proposition true if a new index and instruction section of a word is to be written in memory.
Λ_{Ad}	A proposition true if a new address section of a word is to be written in memory.
μ	A proposition true if (G) = 26 or 27.
ν	A proposition true if (G) = 40, 41, 42, 43, 44 or 45.
σ	The subtraction proposition.
Σ_l, Σ_r	The shift-left and shift-right operations.
Φ_n	The add-subtract function.
χ_n	The carry functions n = 0, 1. . . 24.
ψ_l, ψ_r	The cycle-left and cycle-right operations.

LOGICAL DESIGN OF CG24 (A GENERAL-PURPOSE COMPUTER)

I. INTRODUCTION

A logical description of the Group 24 computer (CG24), a high-speed general-purpose digital computer, is presented in this report. The design of the computer has been governed by the following considerations:

(a) The high-speed storage is to be provided by two coincident-current ferrite-core memory banks each containing 4096 words, 27 bits in length.

(b) Transistor circuits are to be used throughout, including the memory driver circuits.

(c) A speed of about 40,000 add-type single-address operations per second is required.

Emphasis was placed on obtaining reliability of operation. With the exception of display, the computer is all solid state. Consequently, the physical size of the machine will be roughly the size of the console of machines of similar speed and capacity (see Fig. 1).

The control of this machine is a fixed-diode memory with a 6-bit memory address register which may be modified conditionally, either by the instruction section of a computer word or for iterative instructions by the address section of the control word. This static memory could be replaced by a dynamic memory with no necessary logical changes in the control. With the present control memory and its associated address and output registers, the creation of new computer instructions is conceptually equivalent to programming where a computer instruction is now replaced by a program of microinstructions.

For real-time data processing, three input buffer registers are provided. Two of these have direct access to the core memory. The necessary computer instructions for reading-in these input words, either one at a time or two at a time, have been provided.

Sections II, III and IV present a general description of the machine and its design techniques. Table I gives a general outline of the computer characteristics, and the remainder of the report discusses the detailed structure of the computer.

II. TERMINOLOGY AND CONVENTIONS

A register, physically, is a set of one or more bistable devices. In this report a register will be designated by an upper case, Roman letter. Parentheses are used to denote the contents of a register. Thus (A) signifies the contents of the A-register. Parts of registers are again registers that are functionally dependent upon the register of which they are a part, as well as upon the meaning to be attached to a particular part of a register. An abbreviation denotes the significance of part of a register and brackets describe the functional dependence upon the entire register. Thus Ad[R] signifies the address part of the R-register. This notation makes it possible to refer to the corresponding sections of several registers. For example Ad[R], Ad[A] and Ad[N] refer to the address parts of the R-, A- and N-registers, respectively. To refer to a register except for one bit, the symbol $[A; \bar{A}_{24}]$ is used, which means the A-register excluding A_{24}.

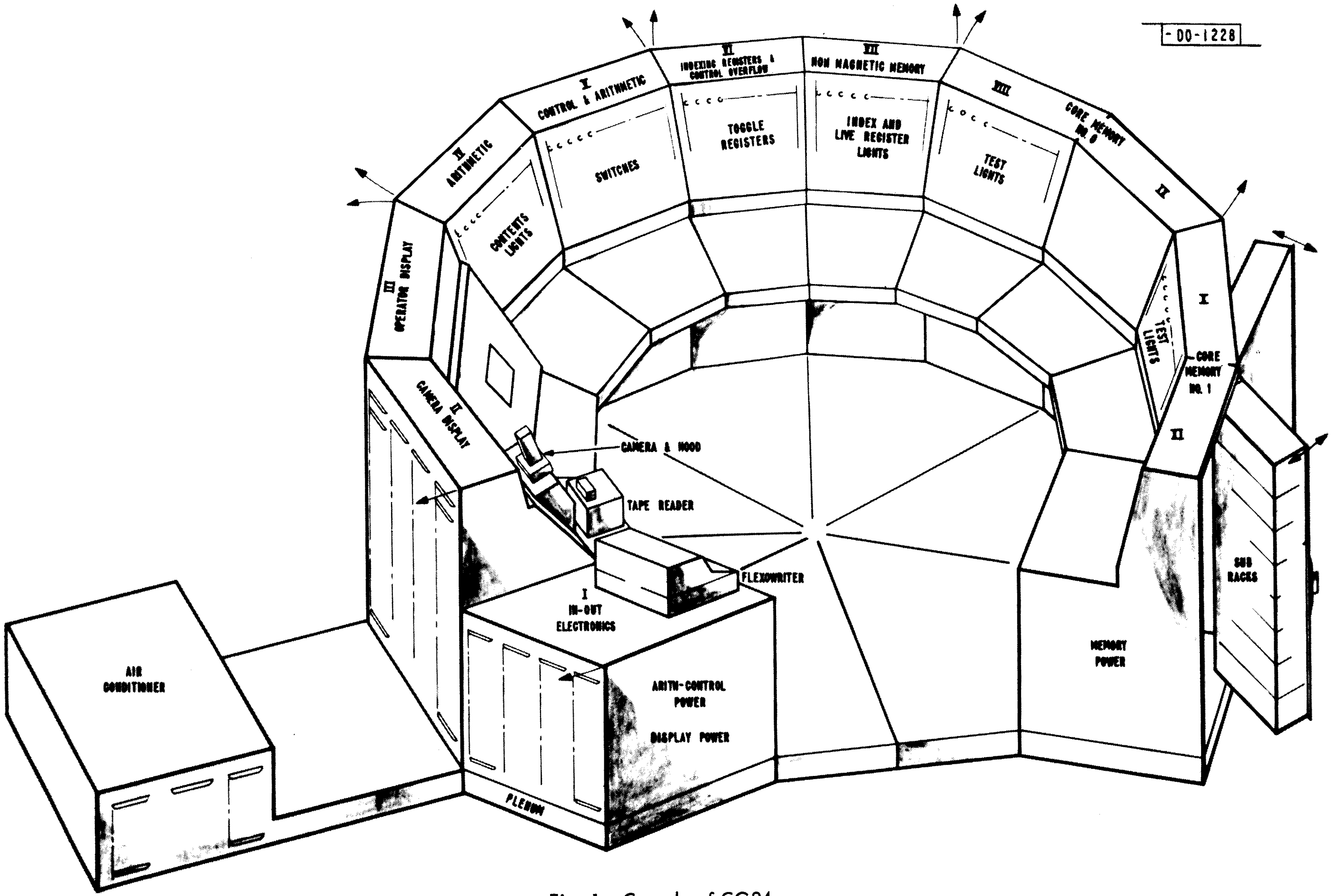

Fig. 1. Console of CG24.

TABLE I

COMPUTER CHARACTERISTICS

General System

Application – general purpose

Timing – synchronous

Operation – sequential concurrent

Numerical System

Internal number system – binary

Binary digits per word – 25

Binary digits per instruction – 25 (includes address and index bits)

Additional binary digits per word or instruction for parity check – 2

Instructions per word – 1

Instructions written now – 38

Arithmetic system – fixed point

Instruction type – one address

Number range – $1 \leqslant n \leqslant 1 - 2^{-24}$

Arithmetic Unit

Addition time	–	24 μsec (including memory access)
Multiplication time	–	84 μsec (")
Division time	–	84 μsec (")
Square root time	–	300 μsec (")

Construction – transistors, crystal diodes

Basic pulse repetition rate – 330 kcps

Storage

Media – magnetic cores

Words – 8192

Access time (to any word) – 12 μsec

Checking Features

Parity checking for words going into and out of the memory is made up of two checks, one for instruction and index bits, one for address section.

It is often necessary to use a single symbol to refer to a set of registers, one of which is to be selected on the basis of the contents of another register that addresses the set of registers. To illustrate, the symbol M<C> means the memory register determined by the contents of register C, the memory address register.

Often, two or more registers are treated as a single register. The symbol (A, B) signifies the contents of A- and B-registers in tandem from left to right; furthermore let $\big((A) + (R), B\big)$ represent a tandem register containing the sum of (A) and (R) together with B.

Another convenient symbol is Shr A which denotes a transformed A-register with all digits shifted right by one, with a zero inserted in the leftmost digit unless otherwise specified. Similarly Shl A denotes the A-register with all digits shifted left by one, with a zero inserted into the rightmost digit unless otherwise specified.

The individual bits of a register are designated by the letter denoting the register, together with a subscript referring to the bit in question. The bit numbers run from zero at the left to N-1 at the right where N is the register length (in this case 25). The complement or negation of an individual bit is denoted by the prime. Thus A_o is the leftmost bit of the A-register and A'_o is its complement.

The complement of a register is denoted as $\overline{A}$, and defined to be the transform of the register A obtained by complementing or negating each of the bits of the register. Thus $(\overline{A}_i) = (A'_i)$. This is often referred to as the "one's" complement. Hence the following identity holds:

$$(A) + (\overline{A}) = -2^{-(N-1)} \quad ,$$

where + means the arithmetic sum.

In referring to two-valued functions of the machine, we employ the notation of Boolean algebra summarized below:

α	β	$\alpha + \beta$	$\alpha\beta$	$\alpha \oplus \beta$	α'
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

where α and β represent two-valued functions. The function α or β indicated by $\alpha + \beta$ is true if and only if α or β (or both) is true. The function α and β indicated by $\alpha\beta$ is true if and only if both α and β are true. The function α "exclusive or" β indicated by $\alpha \oplus \beta$ is true if and only if either α or β (but not both) is true. α' is true if and only if α is false. The "exclusive or" function may be expressed as $\alpha \oplus \beta = \alpha\beta' + \alpha'\beta$ in terms of the "and", "or" and prime operations.

In referring to arithmetic operations between registers we use conventional notation. Thus (A) + (R) symbolizes the sum of the contents of the A- and R-registers and (A) – (R) represents the difference. There should be no confusion due to the use of the + to represent both the sum of the contents of two registers and the "inclusive or" operation between two-valued functions.

For the most part, the functions of the machine are described in terms of elementary operations between various registers. Basically, these operations are all transfers. We use the double arrow $\Rightarrow$ to represent the transfer operation. Thus $(A) \Rightarrow B$ means that the contents of the A-register are transferred into the B-register. Implied in this notation is $(A_n) \Rightarrow B_n$, or the contents of the n-th bit of A is transferred into the n-th bit of B. Similarly, $(A) \Rightarrow$ Shl B implies that $(A_{n+1}) \Rightarrow B_n$, $n = 0, 1 \ldots N-1$ and $0 \Rightarrow B_{24}$.

Conditional transfers are symbolized by defining Boolean functions specifying conditions. Suppose that λ is a two-valued function of one or more flip-flops of the computer, then $\lambda(F) + \lambda'(G) \Rightarrow G$ means that if λ is one (true) the contents of F are transferred to G, or if λ is zero (false) the contents of G are transferred to G or G remains unchanged.

In referring to the operations between registers, we shall always define a time interval between clock pulses during which the operations take place. One way of defining a time interval between two specific successive clock pulses is by a Boolean function σ which is "one" during this particular time interval and "zero" at all other times. Since the function σ is generated within the machine, it is a function of the contents of the various registers of the machine. This means that intervals of time for which $\sigma = 1$ are determined by sets of configurations of the machine.

Let the set of configurations of the machine for $\sigma = 1$ be denoted by $|\sigma|$. Then the symbolism $|\sigma| : (A) + (R) \Rightarrow A$ means that at the end of the time interval between successive clock pulses for which $\sigma = 1$, the sum of the contents of A and R at the beginning of the interval is transferred into A. For further remarks on the above nomenclature, see the references given in the footnotes below.

III. STRUCTURE OF THE COMPUTER

Figure 2 shows a block diagram of the computer. For the purposes of this description we have divided the computer into five large blocks: (1) the memory and its associated registers, (2) the program registers, (3) the arithmetic registers, (4) the control and (5) terminal equipment.

A. Memory

The memory of the computer consists of two coincident-current ferrite-core banks each containing 2^{12} registers, 27 bits in length, and a third memory bank containing 18 front-panel toggle-switch registers, 76 plug-board registers, and three flip-flop registers, each 25 bits long. The two extra bits in the core-memory registers are used for parity checking. We label the two core memories M^0 and M^1 and the noncore memory, M^2. Each of the core memories M^j $(j = 0,1)$ has associated with it an output register N^j, an input register L^j and an address register C^j containing 12 bits. The noncore memory M^2 has an address register C^2 containing 14 bits. Input and output buffer registers are unnecessary for the noncore memory.

When any register, say N^1, contains a number, the left-hand bit N_0^1 stores the sign bit and the rest of the register stores the number with the binary point always located between N_0^1 and N_1^1. Negative numbers are stored as the "two's" complement of the corresponding positive numbers. That is, to obtain the negative of a number we complement every bit of the number

*I.S. Reed, Technical Memorandum-23, Lincoln Laboratory, M.I.T. (19 January 1953).

† I.S. Reed, Group Report 312-2, Lincoln Laboratory, M.I.T. (January 1956).

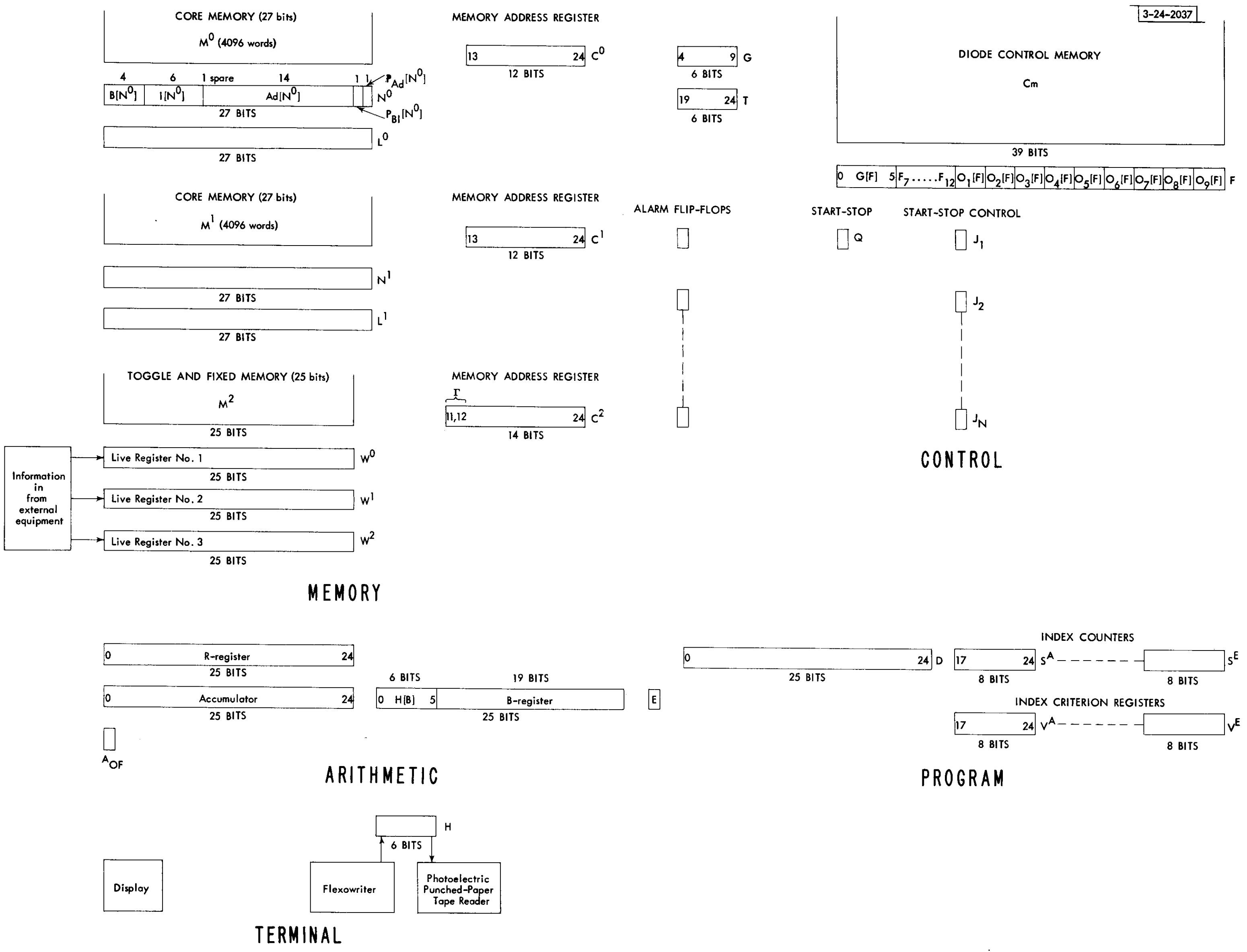

Fig. 2. Block diagram of CG24.

and add "1" to the least significant bit. Thus if a number is stored in N^1 such that $(N_k^1) = \alpha_k$, then

$$\alpha = (-1)\,\alpha_o + \sum_{k=1}^{24} \alpha_k 2^{-k} \quad ,$$

$\alpha_k = 0$ or 1, $k = 0,1\ldots 24$.

When any register, say N^2, contains an instruction, the index code is stored in the first four bits $N_0^2 - N_3^2$ labeled $B[N^2]$, the binary code for the instruction is stored in the next six bits $N_4^2 - N_9^2$ labeled $I[N^2]$, and the address is stored in the last fifteen bits $N_{10}^2 - N_{24}^2$ labeled $Ad[N^2]$. $BI[N^j]$ denotes the tandem register $B[N^j], I[N^j]$.

Because of physical considerations it is desirable for each core memory to be operative at all times, whether or not the rest of the computer is using the information. Thus when a 15-bit memory address is specified, the 12 least significant bits are transferred into each of the memory address registers $C^j (j = 0,1,2)$, while bits 11 and 12 go into C^2 and bit 10 is reserved as a spare. We label bits 11 and 12 of C^2, $\Gamma[C^2]$ or more simply Γ. It is the state of Γ which determines which of the three memory banks is being referred to. We use the symbol $N<\Gamma>$ to specify which memory output register will be used by the computer at a given time. By definition,

$$\begin{array}{ll} (\Gamma) = 0 & N<\Gamma> = N^0 \\ (\Gamma) = 1 & N<\Gamma> = N^1 \\ (\Gamma) = 2 & N<\Gamma> = M^2<C^2> \quad . \end{array}$$

Corresponding definitions apply to $L<\Gamma>$.

B. The Program Registers

The program register block of Fig. 2 contains the program counter D, a 25-bit register, the last 14 bits of which keep track of the address of the succeeding instruction and the index registers V^j and S^j, $j = A,B\ldots E$, which are used for indexing iterative programs. Each pair of registers V^j and S^j is addressed by the index portion of an instruction. Thus the symbol $V<B[N^1]>$ means the V-register depending upon the index part of register N^1. The V-registers are called index-criterion registers. They store the number of times a program is to be iterated. We call the S-registers, index counters. They are used to count the number of times the program has currently been iterated.

C. The Arithmetic Registers

The arithmetic section consists of three 25-bit registers, A, B and R, and a 1-bit register, E. The A-register is the accumulator which is used to store partial arithmetic results. The B-register may be regarded as an extension of the accumulator. The two registers together form a 50-bit shift register. For example, the 49-bit product, resulting from the multiplication of two 25-bit numbers (24 bits plus sign), appears in the accumulator plus bits 0 to 23 of the B-register. The R-register is used mostly to store operands, i.e., the addend, multiplicand, divisor, etc. Numbers entering the arithmetic section from memory always transfer into the R-register according to the command $(N<\Gamma>) \Rightarrow R$. All arithmetic operations are done between

TABLE II

LIST OF ORDERS

Orders	Operation Time (12-μsec units)	Orders	Operation Time (12-μsec units)
1 Add X	2	22 Store X	2
2 Subtract X	2	23 Replace address X	2
3 Clear and add X	2	24 Transfer X	1
4 Clear and subtract X	2	25 Transfer on negative X	2
5 Multiply nonroundoff X	7	26 Transfer on index X	2
6 Multiply roundoff X	7	27 Index n	2
7 Multiply shift X	7	30 Return from X	2
10 Divide X	7	31 Halt	2
11 Square root	25	32 Clear B	2
12 Subtract magnitudes X	2	33 Store one X	2
13 Extract X	2	34 Store both X	3
14 Identify X	3	35 Shift and add X	2
15 Shift left n	$[\frac{n+7}{4}]$	36 Transfer on busy bit X	2
		37 Transfer on overflow X	2
16 Shift right n	$[\frac{n+7}{4}]$	40 Display X	*
		41 Display word X	*
17 Cycle left n	$[\frac{n+7}{4}]$	42 Index camera	*
		43 Read-in	*
20 Cycle right n	$[\frac{n+7}{4}]$	44 Punch	*
		45 Print	*
21 Scale factor X	4 to 8	77 Exchange X	2

* Asynchronous orders.

R and A. Thus between R and A is an adder-subtractor network.

Communication between the computer and terminal equipment such as tape reader and Flexowriter is accomplished by a 6-bit register H feeding into the B-register. The 1-bit register E is used for the multiplication and division algorithms.

D. Control

Each order of the computer takes one or more memory cycles for execution. The subinstructions or microinstructions that occur during each memory cycle are coded and stored in a set of registers called the control memory Cm. The 6-bit instruction code from an order in the main memory is transferred to a 6-bit control register G that addresses the control memory. We thus refer to the control memory as Cm<G>. The F-register is the output buffer register of Cm<G>. The word length of Cm<G> and F is 39 bits.

The first six bits G[F] specify the address in Cm<G> of the control word for the succeeding memory cycle. The next six bits are used individually to specify the most widely used subinstructions (microinstructions) or groups of subinstructions. The other 27 bits are arranged in nine groups of three each $O_1[F] - O_9[F]$, each group specifying up to seven microinstructions.

Some orders, like multiplication, call for the repetition of a single memory cycle several times. To accomplish this, the register T is provided, which counts the number of times a control word is repeated. A function of T is used to "stick" the control on a single control-memory word.

Also in the control section are the start-stop flip-flop Q, the alarm flip-flops and several additional flip-flops associated with the start-stop control circuitry.

E. Terminal Equipment

Program input to the computer is from a photoelectric punched paper tape reader (PETR) via the in-out buffer register H.

Data may be supplied to the computer through the three flip-flop registers W^0, W^1 and W^2 that are part of M^2. Provision is also made to transfer the contents of W^1 and W^2 directly into the core memory.

Output from the computer is obtained from a Flexowriter, again via H, and from two display oscilloscopes fed directly from the arithmetic registers. One display scope has a camera for recording results; the other is for the use of the operator. The scopes may be used to display either points or octal representations of computer words.

IV. TABULATION OF MACHINE ORDERS

Thirty-eight orders have been designed for the computer. Most of these are the usual arithmetic and logical orders found in computers such as Whirlwind I. In addition, the orders "store one X" and "store both X" provide the mechanism for direct storage of real-time data into the high-speed memory.

The orders with their code numbers and execution times are presented in Table II.

V. DESCRIPTION OF MACHINE ORDERS

A. Basic Timing

The computer timing is based upon that of the core-memory cycle, i.e., all of the basic microoperations occur at 3-μsec intervals, corresponding roughly to the memory subintervals shown in Fig. 3. In reference to the memory cycle, zero time is defined by the read clock. The other clocks, i.e., write, and the post-write disturbs occur at 3.5, 6 and 9 μsec, respectively. A memory pulse occurring 2.5 μsec after the read clock serves as the basic timing pulse for the computer. We label this pulse s_1. Pulses s_2, s_3 and s_4 are generated 3, 6 and 9 μsec from s_1, thus defining the four pulse intervals P_1 through P_4 as shown in Fig. 3.

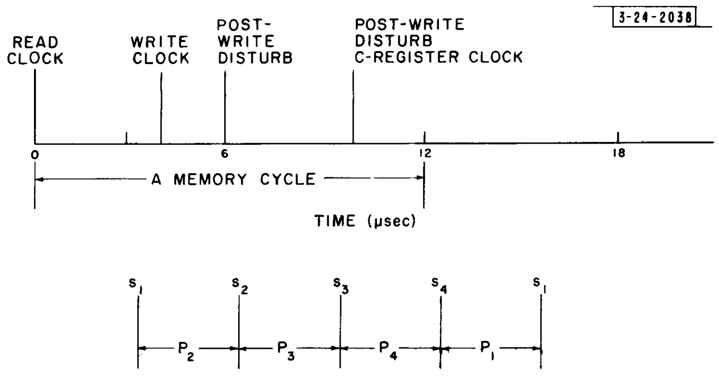

Fig. 3. Computer timing.

A timing function σ as defined in Sec. II is described by specifying a proposition f_j from the control output register, together with a timing proposition P_k. In addition, it is convenient to treat one bit of the F-register separately. This bit, designated "a", specifies whether or not the state represents a memory cycle during which an instruction is being read out of memory. If a = 1, an instruction is being read out; if a = 0, no instruction is being read out. An instruction is always read out of memory during the last memory cycle of the preceding order. This overlap feature shortens some orders by 12 μsec. Therefore, the timing function σ may be represented by $a'f_jP_k$ during all memory cycles, except the final one of each order which is labeled af_jP_k.

B. Description of Transfers Common to All Cycles

The following sequence of microinstructions occurs during every memory cycle. The states are labeled simply f_jP_k.

On All Cycles

$$|f_kP_1| : (M^j\langle C\rangle) \Rightarrow N^j, \quad j = 0, 1$$

$$P_2| : \Lambda_{BI}^{j'}(BI[N^j]) + \Lambda_{BI}^{j}(BI[L^j]) \Rightarrow BI[M^j\langle C\rangle]$$

$$\Lambda_{Ad}^{j'}(Ad[N^j]) + \Lambda_{Ad}^{j}(Ad[L^j]) \Rightarrow Ad[M^j\langle C\rangle]$$

$$a(I[N\langle\Gamma\rangle]) + a'\{\lambda(G[F]) + \lambda'(G)\} \Rightarrow G$$

$$\lambda = \{(T) = 0, 1, 2, 3\} + \{(G[F]) = 64, 65\} \left\{A'_o \sum_{i=1}^{4} A_i + A_o \sum_{i=1}^{4} A'_i\right\}$$

$P_3|$:

$P_4|$: $(Cm\langle G\rangle) \Rightarrow F$

$$\{(\Gamma) = 0, 1\} \left\{\oplus \sum_{i=0}^{9} N_i\langle\Gamma\rangle \oplus P_{BI}[N\langle\Gamma\rangle] \Rightarrow \overset{Q}{P_{BI}} \text{ alarm FF}; \oplus \sum_{i=10}^{24} Ni\langle\Gamma\rangle \oplus P_{Ad}[N\langle\Gamma\rangle] \Rightarrow \overset{Q}{P_{Ad}} \text{ alarm FF}\right\}$$

$$\nu(BB) \Rightarrow Q, \quad \nu BB'\{1 \Rightarrow BB\}, \quad \nu = \{(G) = 40, 41, 42, 43, 44, 45\}$$

The statement at P_1 and the first two at P_2 refer to the core memories alone. By the end of the P_1 interval each core-memory register as determined by its address register is read, the number from each appearing in its output register. The P_2 interval represents the write period of the memories. Either the old word in N or a new word in L or part of each is written back into each memory, depending upon the functions Λ_{BI} and Λ_{Ad}. The first statement at P_2 says, therefore, that if $\Lambda_{BI} = 0$, the instruction and index section of N are written into memory and if $\Lambda_{BI} = 1$, the corresponding sections of L are written into memory. The second statement refers to the address portion of the word.

The third statement at P_2 refers to control. If a = 1, the computer is in a memory cycle in which a new instruction is being accepted. Thus the instruction portion of N is transferred into the control address register G. If a = 0, either the contents of the address section of F, G[F], are transferred to G or G remains the same, depending upon the control-advancing function λ. We shall explain the makeup of λ later. As was explained in Sec. III, G[F] contains the address of the control memory word for the next memory cycle of the current order.

At P_4 the control word is transferred from memory into F in preparation for the next memory cycle. Also the parity of the word read out of memory is checked and if an error has been made, the start-stop flip-flop Q is set, turning off the machine, and the appropriate alarm is given. For checking parity, the word is broken into two sections, 10 and 15 bits in length, respectively. The parity bit for the first 10 bits is labeled $P_{BI}[N]$ and that for the last 15 bits, $P_{Ad}[N]$. P_{BI} of the word is by definition the sum modulo 2 of the first 10 bits of the word when it was inserted in memory. Thus if the sum modulo 2 of digits 0 to 9 plus P_{BI} is one when the word is read out, then a single error (or more exactly an odd number of errors) was made. The same holds true for the address section of the word.

The final statements at P_4 are concerned with orders involving asynchronous terminal equipment. These transfers will be discussed later in this section.

C. Instruction Read-In Cycles

The above transfers take place during every memory cycle. We next consider the additional transfers which occur during all cycles during which instructions are taken from memory.

Depending upon the particular instruction, the address part of the instruction contains either an address X, a number n or "nothing".

<u>On Instruction Read-In Cycles (a = 1)</u>

$$|af_jP_1| : (C^2) \Rightarrow D$$

$$P_2| : (D) + 1 \Rightarrow D$$

$$P_3| : \{(G) = 37\}'\ SW'_{OF}\ \{(A_{OF}) \Rightarrow \begin{matrix}\text{OF alarm FF}\\ Q\end{matrix}\}\ ,\ \{(G) = 37\}'\ \{0 \Rightarrow A_{OF}\}$$

$$P_4| : \mu(Ad[N<\Gamma>]) + \mu'\{(Ad[N<\Gamma>]) + (S<B[N<\Gamma>]>)\} \Rightarrow \begin{matrix}C\\T\end{matrix}$$

$$\mu'(R) + \mu(N<\Gamma>) \Rightarrow R \quad , \quad 0 \Rightarrow E$$

$$\mu = \{(G) = 26, 27\}$$

During P_1 the contents of C^2 are transferred to D and at P_2, D is counted up by one. At the beginning of the P_1 interval, the number in the memory address register C^2 is the address at which the current instruction is stored. This is transferred to the program counter D and the address number is increased by one in preparation for the next instruction. It will be recalled from Sec.III that each memory bank has associated with it its own address register; C^0 and C^1 going with the core-memory banks have 12 bits, while C^2 has the full 14 bits.

The transfers during P_3 refer to the overflow alarm. The proposition $\{(G) = 37\}$ refers to the order "transfer on overflow". SW_{OF} is a proposition stating that the overflow alarm is suppressed and A_{OF} is the overflow flip-flop. An overflow occurring in a previous order will have caused A_{OF} to be set to "one". The first transfer says, therefore, that if the current order is anything but "transfer on overflow" and if the alarm is not suppressed, the computer is halted and the alarm is given. The second transfer states that A_{OF} is cleared if the order is not "transfer on overflow".

The first statement at P_4 interprets the proper information to be transferred into the memory address register in preparation for the next memory cycle. The function μ is true if and only if the instruction in question is either "index" or "transfer on index". Thus if the instruction is one of the two above or if it is any other instruction that is not being indexed, the address part of the word from memory is transferred into C. If an indexed instruction is referred to, then the sum of the contents of the selected index counter and the address part of the word from memory is transferred into C. If the instruction is unindexed or $(B[N<\Gamma>]) = 0$, then $(S<B[N<\Gamma>]>)$ is defined to be zero so that the address of the instruction is left unmodified. The scheme just outlined makes it possible to employ the same word structure for indexed instructions as well as for the orders "index" and "transfer on index". It is evident that only for the case of indexed instructions do we want to modify the address that is transferred into memory address register C.

In the transfer described above we describe inputs to register C with no superscript. We shall use this same notation below. The notation implies a parallel transfer into all three address registers, C^0, C^1 and C^2. At this point it should be emphasized that the timing notation means that by the end of the interval in question, the stated transfers shall be completed. It does not specify precisely when during the interval the actual clocking takes place. Practically speaking, all the transfers into registers other than those associated with the core memories occur at the pulse times s_i. Thus, with reference to Fig. 3, the transfer $|a'f_jP_4| : (A) \Rightarrow B$ occurs with clock pulse s_4. But in the case of the transfer into the memory address registers, described above as occurring in the P_4 interval, it is only the transfer into C^2 which occurs on the clock s_4. The transfers into the core address registers C^0 and C^1 occur midway between s_3 and s_4. This is a constraint imposed by hardware considerations of the memory. But the statement of the transfer is still accurate within the definition of the timing proposition.

The same number transferred into C is also placed in T. This is in reference to the shift and cycle orders and will be described below.

The second transfer occurring at P_4 of instruction read-in cycles states that for the orders "index" and "transfer on index" the memory word just obtained is placed in R. Otherwise R remains unchanged. Also at P_4 the 1-bit E-register is cleared.

D. Specific Operations

We shall now consider the operations peculiar to each machine order. We emphasize that the operations that follow occur in addition to the ones already described in this section.

01 Add X

$$|a'f_1P_1| : 0 \Rightarrow T$$

$$P_2| : (N\langle\Gamma\rangle) \Rightarrow R$$

$$P_3| :$$

$$P_4| : (D) \Rightarrow C \; ; \; (A) + (R) \Rightarrow A \; ;$$

$$\chi_o A'_o R'_o + \chi'_o A_o R_o \Rightarrow A_{OF}$$

00

$$|af_2P_1| :$$

$$\downarrow$$

$$P_4| :$$

The addition order adds the contents of memory register X to those of the accumulator, the result appearing in A. It takes two memory cycles (24µsec). During the first memory cycle, labeled by the octal instruction code 01 (binary 000001) the actual addition takes place. During the second memory cycle labeled 00, nothing occurs other than obtaining the next instruction. The first cycle is therefore labeled with a' and the second with a.

During the P_1 interval of the first cycle, the T-register is cleared. This must be done to make the advancing function λ true, thus insuring that at P_2, (G[F]) is transferred to G. During this cycle G[F] must evidently be 00, since the next memory cycle is labeled 00. Also during P_1 the contents of the selected memory register are transferred into the memory output register and during P_2 this number is transferred into R. During P_4, the actual addition occurs. The final statement at P_4 is the overflow condition. The function χ_n is defined as the carry into the n-th bit of the sum. Thus if $A_n[\tau]$ is the state of A_n after the addition, then

$$A_n[\tau] = A_n \oplus R_n \oplus \chi_n$$

and

$$\chi_{n-1} = \chi_n(A_n + R_n) + A_n R_n$$

$$\chi_{24} = 0 \quad . \tag{1}$$

The overflow statement says, therefore, that if both (A) and (R) are initially positive, then a carry into the sign bit specifies an overflow, or if both (A) and (R) are negative, then the absence of a carry into the sign bit specifies an overflow.

It may be well to mention at this point that, in general,

$$A_n[\tau] = A_n \oplus \Phi_n \oplus \chi_n$$

and

$$\chi_{n-1} = \chi_n(A_n + \Phi_n) + A_n\Phi_n \quad , \tag{2}$$

where Φ_n is defined by

$$\Phi_n = \alpha R_n + \sigma R'_n$$

and

$$\chi_{24} = \sigma \quad . \tag{3}$$

The functions α and σ are propositions denoting addition and subtraction, respectively. Thus if $\alpha = 1$ (addition), $\Phi_n = R_n$. $\chi_{24} = 0$ and Eqs. (2) are the same as Eqs. (1).

During the P_4 interval the contents of D (the address of the next instruction) are transferred into the memory address registers in preparation for the next cycle, during which time the next instruction is obtained.

<u>02 Subtract X</u>

$|a'f_3P_1|$: $0 \Rightarrow T$

$P_2|$: $(N\langle\Gamma\rangle) \Rightarrow R$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$; $(A) - (R) \Rightarrow A$

$\chi_o A'_o R_o + \chi'_o A_o R'_o \Rightarrow A_{OF}$

<u>00</u>

$|af_2P_1|$:

$\downarrow$

$P_4|$:

In the subtraction order the number in memory register X is subtracted from the number in A, the result appearing in A. This order is essentially the same as addition, except the arithmetic operation subtraction is performed. In Eqs.(2) $\sigma = 1$, $\alpha = 0$. Thus in Eqs.(3), $\Phi_n = R'_n$ and $\chi_{24} = 1$. The overflow condition is the same as that in addition with R_o replaced by R'_o.

b

<u>03 Clear and Add X</u>

$|a'f_4P_1| : 0 \Rightarrow T\ ;\ 0 \Rightarrow A$

$P_2| : (N\langle\Gamma\rangle) \Rightarrow R$

$P_3| :$

$P_4| : (D) \Rightarrow C; (A) + (R) \Rightarrow A$

$$\chi_o A'_o R'_o + \chi'_o A_o R_o \Rightarrow A_{OF}$$

<u>00</u>

$$\begin{tikzcd}|af_2P_1| : \arrow[d] \\ P_4| :\end{tikzcd}$$

The clear and add order transfers the contents of memory register X into A. It is identical to addition with the A-register cleared during P_1.

<u>04 Clear and Subtract X</u>

$|a'f_5P_1| : 0 \Rightarrow T\ ;\ 0 \Rightarrow A$

$P_2| : (N\langle\Gamma\rangle) \Rightarrow R$

$P_3| :$

$P_4| : (D) \Rightarrow C\ ;\ (A) - (R) \Rightarrow A$

$$\chi_o A'_o R_o + \chi'_o A_o R'_o \Rightarrow A_{OF}$$

<u>00</u>

$$\begin{tikzcd}|af_2P_1| : \arrow[d] \\ P_4| :\end{tikzcd}$$

The clear and subtract order transfers the negative of the number in the memory register X into A. It is identical to subtraction with A cleared during P_1.

05 Multiply Nonroundoff X

$|a'f_6P_1|$: $0 \Rightarrow T$; $0 \Rightarrow A$; $(A) \Rightarrow B$

$P_2|$: $(N<\Gamma>) \Rightarrow R$

$P_3|$: $\times$

$P_4|$: $\times$; $(D) \Rightarrow C$; $24_{octal} \Rightarrow T$

50

$|a'f_7P_1|$: $\times$; $\{(T) \neq 0\}$ $\{(T) - 1 \Rightarrow T\}$

$P_2|$: " "

$P_3|$: " "

$P_4|$: " "

51

$|af_8P_1|$: $\times$

$P_2|$: $\times$

$P_3|$: Y, $(R_0) \Rightarrow E$

$P_4|$: $A_0B_{24}E\ \{1 \Rightarrow \overset{\text{M-D alarm FF}}{Q}\}$

$\times : \{B_{24} \oplus E\}'\{A, B, E) \Rightarrow \text{Shr } A, B, E\ ;\ (A_0) \Rightarrow A_0\}$

$+ B_{24}E'\Big\{\big((A) - (R), B, E\big) \Rightarrow \text{Shr } A, B, E\ ;\ (A_0) - (R_0) \Rightarrow A_0\Big\}$

$+ B_{24}'E\Big\{\big((A) + (R), B, E\big) \Rightarrow \text{Shr } A, B, E\ ;\ (A_0) + (R_0) \Rightarrow A_0\Big\}$

$Y : \{B_{24} \oplus E\}'\{(A, B, E) \Rightarrow A, B, E\}$

$+ B_{24}E'\Big\{\big((A) - (R), B, E\big) \Rightarrow A, B, E\Big\}$

$+ B_{24}'E\Big\{\big((A) + (R), B, E\big) \Rightarrow A, B, E\Big\}$

In this order the number in memory register X is multiplied by the number in the A-register, the 49-bit product (48 bits plus sign) appearing in A and bits 0 to 23 of B. The time for this order is 7 memory cycles or 84μsec. During the operation, the A-, B- and E-registers are treated as a single 51-bit shift register.

Initially the multiplier in A is transferred into B, and A is cleared. E was cleared during the previous "a" cycle. The multiplicand is held in R. The algorithm represented by × is performed 24 times and a variation of the algorithm Y is done once. The number 24 octal (or 20 decimal) is inserted in T. This is done to make the advancing function λ false during the next memory cycle. During this next cycle × is performed 4 times and each time T is counted down by one. Thus the cycle 50 is performed 5 times. At the end of P_1 of the fifth time T has been counted down to 3, λ becomes true, and memory cycle 51, the last cycle, is performed. This cycle is designated by "a", hence the next instruction is interpreted during this cycle.

The algorithm states that if B_{24} and E are alike, the 51-bit register A, B, E is shifted right by one bit, the sign of A being preserved. If $B_{24} = 1$ and $E = 0$, the number in R is subtracted from that in A; this difference, together with B and E, is shifted right by one bit and the sign bit of the difference is preserved in A_o. If $B_{24} = 0$ and $E = 1$, the same occurs with a sum between R and A. The last time, the same occurs with no shifting.

The final transfer states that if both the multiplier and multiplicand are negative and the product is negative, the computer is halted and the multiply-divide alarm is given. This condition will occur only if the multiplier and multiplicand are both –1 (1.000 . . . 0). The algorithm gives as a product for this case the number –1, since +1 is not representable in the number system used.

<u>06 Multiply and Roundoff X</u>

$$\begin{tikzcd}
\lvert a'\tilde{f}_6 P_1 \rvert : \arrow[d] & = \lvert a' f_6 P_1 \rvert \arrow[d] \\
P_4 \rvert : & P_4
\end{tikzcd}$$

<u>60</u>

$$\begin{tikzcd}
\lvert a'\tilde{f}_7 P_1 \rvert : \arrow[d] & = \lvert a' f_7 P_1 \rvert \arrow[d] \\
P_4 \rvert : & P_4
\end{tikzcd}$$

<u>61</u>

$\lvert a f_9 P_1 \rvert : \times$

$P_2 \rvert : \times$

$P_3 \rvert : Y\ ;\ 0 \Rightarrow [R\ ;\ \overline{R}_{24}],\ \ (B_o) \Rightarrow R_{24},\ \ (R_o) \Rightarrow E$

$P_4 \rvert : A_o'\{(A) + (R)\} + A_o(A) \Rightarrow A\ ;\ A_o B_{24} E\{1 \Rightarrow \substack{\text{M-D alarm FF} \\ Q}\}$

$0 \Rightarrow B$

This order is performed in the same way as nonroundoff multiply. The 24-bit (plus sign) roundedoff result is obtained during P_4 of the final cycle by adding 1 to A if the product is positive and if the digit in B_o is "one". A negative product is automatically rounded off. Finally the B-register is cleared.

<u>07 Multiply and Shift X</u>

$$\begin{tikzcd}
\lvert a'\overset{\approx}{f}_6 P_1 \rvert : \arrow[d] & = \lvert a' f_6 P_1 \rvert \arrow[d] \\
P_4 \rvert : & P_4
\end{tikzcd}$$

70

$$\begin{tikzcd} {|a'\overset{\approx}{f}_7P_1| :} \arrow[d] & {= |a'f_7P_1|} \arrow[d] \\ {P_4| :} & P_4 \end{tikzcd}$$

71

$|af_{37}P_1| : \times$

$P_2| : \times$

$P_3| : Y, \ (R_0) \Rightarrow E$

$P_4| : (B_n) \Rightarrow A_{n+1} \ ; \ (A_0) \Rightarrow A_0 \ , \ \{A_0B_{24}E\} \ \{1 \Rightarrow \overset{Q}{\text{M-D alarm FF}}\}$

$0 \Rightarrow B$

This order is to be used only when it is known ahead of time that all of the significant bits of the product lie in the B-register, i.e., in the 24 least significant bits. During the final pulse interval, these 24 bits are transferred into A while B is cleared.

10 Divide X

$|a'f_{10}P_1| : 0 \Rightarrow T$

$P_2| : (N\langle\Gamma\rangle) \Rightarrow R \ ; \ 0 \Rightarrow B \ ; \ (A_0) \Rightarrow E$

$P_3| : Z_A$

$P_4| : (D) \Rightarrow C \ ; \ 24_{\text{octal}} \Rightarrow T \ ; \ Z$

$$\left\{E \oplus \left(B_{24} + R_0 \prod_{i=0}^{24} A_i'\right)\right\}' \Rightarrow \overset{Q}{\text{M-D alarm FF}}$$

52

$|a'f_{11}P_1| : Z \ ; \ \{(T) \neq 0\} \ \{(T) - 1 \Rightarrow T\}$

$P_2| :$ " "

$P_3| :$ " "

$P_4| :$ " "

$|af_{12}P_1| : Z$

$P_2| : Z$

$P_3| : Z_B \ ; \ (B_n) \Rightarrow A_{n-1} \ ; \ 0 \Rightarrow [R; \overline{R}_{24}] \ ; \ (R_0) \Rightarrow R_{24}$

$P_4| : R_{24}'(\overline{A}) + R_{24}\{(A) + (R)\} \Rightarrow A \ ; \ 0 \Rightarrow B$

$$Z_A : \{A_0 \oplus R_0\} \big(B, (A) + (R)\big) + \{A_0 \oplus R_0\}' \big(B, (A) - (R)\big) \Rightarrow \text{Shl } B, A$$

$$Z : \{B_{24} \oplus R_0\} \big(B, (A) + (R)\big) + \{B_{24} \oplus R_0\}' \big(B, (A) - (R)\big) \Rightarrow \text{Shl } B, A$$

$$Z_B : \{B_{24} \oplus R_0\} \{(A) + (R)\}_0 + \{B_{24} \oplus R_0\}'\{(A) - (R)\}_0 \Rightarrow A_{24}$$

In the divide order, the number in A is divided by the number in register X. The divisor is stored in R and the quotient finally appears in A with B cleared.

During the operation A and B are used as a 50-bit shift register B, A. The algorithm is described by the statements Z_A, Z and Z_B, the first and last applied once and the second 23 times. Initially B is cleared, then Z_A is performed during P_3 of the first cycle. If the signs of the dividend and divisor are different (alike), the number in R is added to (subtracted from) that in A, the result being shifted left by one digit and B shifted left accordingly. At this point a check is made to determine if the dividend and divisor are such that the quotient is a representable number in the machine. The criterion for this is

signs alike	$\lvert R\rvert > \lvert A\rvert$
signs different	$\lvert R\rvert \geqslant \lvert A\rvert$.

If this criterion is not met, the machine is stopped and the multiply-divide alarm is activated.

The statement Z is then performed 23 times. During P_3 of the last cycle, the 24 digits already computed are transferred into A and the final digit is computed by Z_B and inserted in A_{24}. It will be noted that the calculation in Z_B is the same as that in Z, except that only the sign bit of the result is used. At this point R is cleared and the sign bit of R is inserted in R_{24}. During P_4 the quotient is obtained by complementing the number in A if the sign of the divisor is positive or by adding 1 into the least significant bit of A if the sign of the divisor is negative.

11 Square Root X

$\lvert a'f_{38}P_1\rvert : 0 \Rightarrow T,\ 0 \Rightarrow A,\ (A_0) \Rightarrow \overset{Q}{\text{square-root alarm FF}}$

$(A_n) \Rightarrow B_{n-1},\ 0 \Rightarrow B_{24}$

$P_2\rvert : (A, B) \Rightarrow \text{Shl } A, B,\ 0 \Rightarrow R$

$P_3\rvert : (A, B) \Rightarrow \text{Shl } A, B,\ \{A_{24} + B_0\}\ 1 \Rightarrow R_{24}$

$P_4\rvert : (D) \Rightarrow C,\ 0 \Rightarrow D; D_{24},\ (R_{24}) \Rightarrow D_{24}$

$\big((A) - (R), B\big) \Rightarrow \text{Shl } A, B,\ 32_{\text{octal}} \Rightarrow T$

55

$\lvert a'f_{39}P_1\rvert : (A, B) \Rightarrow \text{Shl } A, B,\ (D_n) \Rightarrow R_{n-2},\ 1 \Rightarrow R_{24},\ (T) - 1 \Rightarrow T$

$P_2\rvert : (A) - (R) \Rightarrow A$

$P_3\rvert : (D) \Rightarrow \text{Shl } D,\ (A_0') \Rightarrow D_{24},\ A_0\big((A) + (R), B\big) + A_0'(A, B) \Rightarrow \text{Shl } A, B$

$P_4\rvert :$

57

$|a'f_{41}P_1|$: $(D_n) \Rightarrow B_{n-1}$, $0 \Rightarrow A$

$P_2|$:

$P_3|$:

$P_4|$: $(B_n) \Rightarrow A_{n+1}$, $(A_o) \Rightarrow A_o$, $0 \Rightarrow B$

In this order the number in A is replaced by its positive square root. We employ the conventional algorithm where the number is examined two digits at a time. At each step the partial result is doubled then shifted left and, together with a trial number inserted in its least significant place, forms a trial divisor. This becomes particularly simple in binary arithmetic since the trial number can only be "1". We store the number at first in B and shift it left into A, two bits at a time. The partial result is stored in D and the trial divisor is held in R.

During the $a'f_{38}P_1$ interval, A_{1-24} is transferred to B_{0-23}, A is cleared and B_{24} is cleared. If the number is negative, an alarm is displayed and the computer is stopped. During P_2 and P_3, two shifts are made placing the two most significant bits in A_{23} and A_{24}. During P_2, R is cleared and in P_3 a "1" is placed in R_{24}, if either of the first two digits is a "1". In this case the first bit of the result is a "1" and this is inserted in D_{24} during P_4. Also during P_4, the rest of D is cleared, the first subtraction of R from A occurs and the result is shifted left by one bit. The next cycle is to be performed 23 times, hence 26 decimal (32 octal) is placed in T.

In the next memory cycle during P_1, another shift occurs leaving the second pair of digits in A_{23} and A_{24}. The number in D (the partial result) is transferred to R but shifted left by 2 bits and the trial number 1 is inserted in R_{24}. During $a'f_{39}P_2$ the trial divisor in R is subtracted from A. If the difference is positive, the next digit in the result is a "1" and this is inserted in D_{24}; the rest of D is shifted left by one. Also the A- and B-registers are shifted by one. If the result of the subtraction is negative, the next digit in the result is "0". Also the accumulator must be corrected by adding R to A and shifting. This cycle occurs 23 times. At the end of the 23rd time, a 24-digit result appears in $D_1 - D_{24}$.

During the final cycle, this result is transferred to A via B. The result is accurate to at least one part in 2^{22}.

12 Subtract Magnitudes X

$|a'f_{13}P_1|$: $0 \Rightarrow T$; $(A) \Rightarrow B$

$P_2|$: $(N\langle\Gamma\rangle) \Rightarrow R$

$P_3|$: $\{A_o \oplus R_o\}'\ \{(A) - (R) \Rightarrow A\}$

$+ \{A_o \oplus R_o\}\ \{(A) + (R) \Rightarrow A\}$

$P_4|$: $B_o(\bar{A}) + B_o'(A) \Rightarrow A$; $(D) \Rightarrow C$

<u>54</u>

$|af_{14}P_1|$:

$P_2|$:

$P_3|$: $0 \Rightarrow [R; \overline{R}_{24}]$; $(B_o) \Rightarrow R_{24}$

$P_4|$: $(A) + (R) \Rightarrow A$;

$\chi_o A_o' R_o' + \chi_o' A_o R_o \Rightarrow A_{OF}$

In this order, the magnitude of the number in memory register X is subtracted from the magnitude of the number in A, the result appears in A and the original contents of A are preserved in B. During the first cycle, if the two numbers have the same sign, a subtraction is performed. If the signs are different, an addition is performed, the result inserted in A. If the original number in A was positive, the desired result is obtained. However, if the original number in A was negative, then the negative of the desired result is in A, and hence the number in A is negated during the final memory cycle.

13 Extract X

$|a'f_{15}P_1|$: $0 \Rightarrow T$

$P_2|$: $(N\langle\Gamma\rangle) \Rightarrow R$

$P_3|$: $(A_k R_k) \Rightarrow A_k \quad k = 0, 1 \ldots 24$

$P_4|$: $(D) \Rightarrow C$

<u>00</u>

$|af_2P_1|$:

$\downarrow$

$P_4|$:

In this order, the logical product of the number in X and that in A is taken, and the result is placed in A.

14 Identify X

$|a'f_{16}P_1|$: $0 \Rightarrow T$

$P_2|$: $(N\langle\Gamma\rangle) \Rightarrow R$

$P_3|$: $(A) - (R) \Rightarrow A$

$P_4|$: $\left\{\prod_{0}^{24} A_i'\right\}(D) + \left\{\prod_{0}^{24} A_i'\right\}' \{(D) + 1\} \Rightarrow D$

$SW_I' \{\prod A_i'\}' \, 1 \Rightarrow \overset{Q}{\text{ID}}$ alarm FF

62

$|a'f_{35}P_1|$:

$P_2|$:

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

00

$|af_2P_1|$:

$\downarrow$

$P_4|$:

In this order the number in X is subtracted from that in A. If the result is zero, the program proceeds in sequence. Otherwise the program counter D is advanced by one. Thus if there is no identity and the identify alarm is suppressed, the program jumps one order. If the alarm is not suppressed the computer is halted and the alarm is activated.

15 Shift Left n

$|a'f_{17}P_1|$: $\{(T) \neq 0\}$ $\Big\{(T) - 1 \Rightarrow T\ ,\ \Sigma_1,\ \{A_0 \oplus A_1\}\ \{1 \Rightarrow A_{OF}\}\Big\}$

$P_2|$: "

$P_3|$: "

$P_4|$: " ; $(D) \Rightarrow C$

00

$|af_2P_1|$:

$\downarrow$

$P_4|$:

$\Sigma_1|$: $(A, B) \Rightarrow$ Shl A, B $(A_0) \Rightarrow A_0$

The shift-left order shifts the contents of the A- and B-registers in regular shift-register fashion, one shift per subinterval. The sign of A is preserved. The number of shifts n is transferred to T during the instruction read-in cycle and during cycle 15, T is counted down as each shift occurs. When the count reaches zero, no more shifting takes place. This cycle is used repetitively until T reaches zero. The control "advancing" function operates so as to continue shifting until total shifting requirements can be fulfilled in the current memory cycle, then the succeeding cycle is allowed to come up. If a significant digit is shifted out, the overflow flip-flop is set.

<u>16 Shift Right n</u>

$|a'f_{18}P_1|$: $\{(T) \neq 0\}$ $\{(T) - 1 \Rightarrow T\ ;\ \Sigma_r\}$

$P_2|$: "

$P_3|$: "

$P_4|$: " ; $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd}[row sep=small] |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

Σ_r : $(A, B) \Rightarrow$ Shr A, B ; $(A_o) \Rightarrow A_o$

Shift right operates in the right direction in the same way that shift left does in the left direction. The sign of A is preserved.

<u>17 Cycle Left n</u>

$|a'f_{19}P_1|$: $\{(T) \neq 0\}$ $\{(T) - 1 \Rightarrow T\ ;\ \psi_l\}$

$P_2|$: "

$P_3|$: "

$P_4|$: " ; $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd}[row sep=small] |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

ψ_l : $(A, B) \Rightarrow$ Shl A, B ; $(A_o) \Rightarrow B_{24}$

Cycle left is a closed-loop shift-register operation. As in the case of shift left, shift right and cycle right, the T-counter becomes preset to the desired number of shifts. In cycle left, the contents of A_1 are transferred into A_o and A_o is shifted into B_{24}.

<u>20 Cycle Right n</u>

$|a'f_{20}P_1|$: $\{(T) \neq 0\}$ $\{(T) - 1 \Rightarrow T\ ;\ \psi_r\}$

$P_2|$: "

$P_3|$: "

$P_4|$: " ; $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

ψ_r : (A, B) $\Rightarrow$ Shr A, B ; $(B_{24}) \Rightarrow A_o$

Cycle right is the closed loop shifting right of the contents of the A- and B-registers.

<u>21 Scale Factor X</u>

$|a'f_{21}P_1|$: (D) $\Rightarrow$ Ad[R] ; 0 $\Rightarrow$ D ; 0 $\Rightarrow$ T

$P_2|$: $\{A_o \oplus A_1\}'\{\Sigma_1,\ (D) + 1 \Rightarrow D\}$

$P_3|$: "

$P_4|$: " ; $60_{octal} \Rightarrow T$

<u>64</u>

$|a'f_{22}P_1|$: $\{A_o \oplus A_1\}'\{\Sigma_1,\ (D) + 1 \Rightarrow D\}$, $\{(T) = 0\}'\{(T) - 1 \Rightarrow T\}$

$P_2|$: " "

$P_3|$: " "

$P_4|$: " "

<u>65</u>

$|a'f_{23}P_1|$: (D) $\Rightarrow$ Ad[L^k], $\Lambda_{Ad}\langle\Gamma\rangle$, 0 $\Rightarrow$ T k = 0, 1

$$\oplus \sum_{10}^{24} D_i \Rightarrow P_{Ad}[L^k]$$

$P_2|$: $\Lambda_{Ad}\langle\Gamma\rangle$

$P_3|$:

$P_4|$: (Ad[R]) $\Rightarrow$ C

<u>00</u>

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

In the scale-factor order, the number in A, B is shifted left until the first "one" ("zero") for positive (negative) numbers appears in A_1. The number of shifts required is tabulated and then stored in the address part of memory register X. The D-register is used to count the shifts. The second term in the expression for λ (see operations for all memory cycles) is used here to repeat cycle 64, except for the special case where the A- and B-registers initially contain 0. Here the first term in the expression for λ causes the advance from 64 to 65 and the number 63 octal (51 decimal) is stored in register X. The original contents of D are stored temporarily in R during the shifting.

<u>22 Store X</u>

$|a'f_{24}P_1| : 0 \Rightarrow T; (A) \Rightarrow L^k ; \Lambda_{BI}\langle\Gamma\rangle, \Lambda_{Ad}\langle\Gamma\rangle$

$$\oplus \sum_0^9 A_i \Rightarrow P_{BI}[L^k] ; \oplus \sum_{i=10}^{24} A_i \Rightarrow P_{Ad}[L^k] \quad , \qquad k = 0, 1$$

$P_2| : \Lambda_{BI}\langle\Gamma\rangle, \Lambda_{Ad}\langle\Gamma\rangle$

$P_3| :$

$P_4| : (D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

The contents of the A-register are transferred to memory register X via the memory input register L, designated by Γ. The function Λ is made true, causing new information to be written into memory. The parity bits are calculated and are stored along with the contents of the A-register.

<u>23 Replace Address X</u>

$|a'f_{25}P_1| : 0 \Rightarrow T, (A) \Rightarrow L^k \qquad k = 0, 1$

$$\Lambda_{Ad}\langle\Gamma\rangle ; \oplus \sum_{10}^{24} A_i = [P_{Ad}L^k]$$

$P_2 : \Lambda_{Ad}\langle\Gamma\rangle$

$P_3 :$

$P_4 : (D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

Replace address stores the address section of the A-register in register X. The instruction section of X in this order remains unchanged. Only the address parity bit is recalculated.

24 Transfer X

$|af_{26}P_1|$: (D) $\Rightarrow$ Ad[R]

$P_2|$:

$P_3|$:

$P_4|$:

This order is an unconditional transfer of control to the register X. It may be used in conjunction with the return from order, and at $f_{26}P_1$ time, the address of the next order is stored in the address section of the R-register. The return from order then picks up this address and stores it at the returning point.

25 Transfer on Negative X

$|a'f_{27}P_1|$: $0 \Rightarrow T$

$P_2|$:

$P_3|$:

$P_4|$: $A'_o\{(D) \Rightarrow C\} + A_o\{(D) \Rightarrow Ad[R]\}$

$\underline{00}$

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

This order is a conditional transfer. If the number in the A-register is positive, the program continues in order. If the number is negative, the transfer to the point designated by transfer order is performed while the advance location D is stored in the address section of R for use with the return-from order.

26 Transfer on Index X

$|a'f_{28}P_1|$: $0 \Rightarrow T$; $(S<B[R]>) + 1 \Rightarrow S<B[R]>$

$P_2|$: $R_o(S<B[R]>) + R'_o\{(S<B[R]>) + 1\} \Rightarrow S<B[R]>$

$P_3|$:

$P_4|$: $\{(S<B[R]>) = (V<B[R]>)\}\ \{(D) \Rightarrow C\}$

$\underline{00}$

$$\begin{tikzcd} |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

This order is a conditional transfer which transfers to a point within an index loop whenever a particular referenced index counter S has not reached a count equal to that which is stored in an index criterion register V, which is the mate of the S under reference. Provision is made to index by either one or two. Bits 1 to 3 are used to determine the proper pair of index registers. If the sign bit (bit 0) contains a "1", then the selected S-register is counted up by one. If the sign bit contains a "0", then S is counted up by two. It will be recalled that during the instruction read-in cycle, the instruction is transferred into R in preparation for this operation. When the counts become equal, the transfer action is stopped and the computer goes on to the next order. A transfer-on-index order must always be preceded, although not immediately, in the program by an index order.

<u>27 Index n</u>

$|a'f_{30}P_1|$: $0 \Rightarrow T$; $(Ad[R]) \Rightarrow V\langle B[R]\rangle$

$0 \Rightarrow S\langle B[R]\rangle$

$P_2|$:

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

The index order is useful for the repetitive use of a simple subroutine, and for the repetitive use of a subroutine that has monotonically increasing register locations. When the index order is introduced in the program, a particular pair of index registers is prescribed, as well as the number of cyclings requested in the index loop. In the index-order operation, the address section of R, which is holding the number of cyclings requested, is transferred to the particular index criterion register selected by the index bit number that is located in the index section of the R-register. At the same time the index counter S, selected by the index bit number, is reset to zero.

<u>30 Return from X</u>

$|a'f_{29}P_1|$: $0 \Rightarrow T$: $(Ad[R]) \Rightarrow Ad[L^k]$, $k = 0, 1$

$\Lambda_{Ad}\langle\Gamma\rangle$; $\oplus \sum_{10}^{24} R_i \Rightarrow P_{Ad}[L^k]$

$P_2|$: $\Lambda_{Ad}\langle\Gamma\rangle$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

This is the introductory order of a subroutine. It must be immediately preceded by a transfer order in the main program which stores the location of the succeeding order of the main program in R. The address X is that of the final instruction of the subroutine (a transfer order). Thus the return-from order replaces the address of the final transfer order of the subroutine with the address of the instruction in the main program to which the subroutine returns.

<u>31 Halt</u>

$|a'f_{31}P_1| : 0 \Rightarrow T$

$P_2| :$

$P_3| :$

$P_4| : (D) \Rightarrow C\ ;\ 1 \Rightarrow Q$

<u>00</u>

$$\begin{tikzcd} {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

The halt order is used at the end of the program and in interior check points of the program. After $f_{31}P_4$ time the Q flip-flop is placed in the "1" state stopping the machine.

<u>32 Clear B</u>

$a'f_{36}P_1| : 0 \Rightarrow T$

$P_2| : 0 \Rightarrow B\ ;\ (A_0) \Rightarrow E$

$P_3| :$

$P_4| : (D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd} {af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

The clear-B order clears the B-register during the P_2 interval. The sign of A is transferred to the E-register only because this operation was used in conjunction with the clearing of the B-register in previous orders.

<u>33 Store One X</u>

$|a'f_{47}P_1|$: $0 \Rightarrow T$, $(W^1) \Rightarrow L^k$, $\oplus \sum_{i=0}^{9} W_i^1 \Rightarrow P_{BI}[L^k]$, $\oplus \sum_{i=10}^{24} W_i^1 \Rightarrow P_{Ad}[L^k]$, $k = 0, 1$

$\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$

$P_2|$: $\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd}|af_2P_1| : \arrow[d] \\ P_4| :\end{tikzcd}$$

This order transfers the contents of live register 1 (W^1) into memory location X. Its operation is the same as the "store" order.

<u>34 Store Both X</u>

$|a'f_{48}P_1|$: $0 \Rightarrow T$, $(W^1) \Rightarrow L^k$, $\oplus \sum_{i=0}^{9} W_i^1 \Rightarrow P_{BI}[L^k]$, $\oplus \sum_{i=10}^{24} W_i^1 \Rightarrow P_{Ad}[L^k]$, $k = 0, 1$

$\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$, $(C^2) \Rightarrow D$, $(D) \Rightarrow Ad[R]$

$P_2|$: $\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$, $(D) + 1 \Rightarrow D$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

<u>67</u>

$|a'f_{49}P_1|$: $(W^2) \Rightarrow L^k$, $\oplus \sum_{i=0}^{9} W_i^2 \Rightarrow P_{BI}[L^k]$, $\oplus \sum_{i=10}^{24} W_i^2 \Rightarrow P_{Ad}[L^k]$, $k = 0, 1$

$\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$

$P_2|$: $\Lambda_{BI}\langle\Gamma\rangle$, $\Lambda_{Ad}\langle\Gamma\rangle$

$P_3|$:

$P_4|$: $(Ad[R]) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd}[row sep=small] {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

In this order the contents of live registers 1 and 2 are stored in registers X and X + 1, respectively. The operation is similar to the previous order with two "store" type memory cycles. The program counter is used to advance the memory from X to X + 1 while the program address is stored in R.

<u>35 Shift and Add X</u>

$|a'f_{51}P_1|$: $0 \Rightarrow T$, $0 \Rightarrow A$, $(A) \Rightarrow B$

$P_2|$: $(N\langle T\rangle) \Rightarrow R$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$, $(A) + (R) \Rightarrow A$, $\chi_o A'_o R'_o + \chi'_o A_o R_o \Rightarrow A_{OF}$

<u>00</u>

$$\begin{tikzcd}[row sep=small] {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

In this order the number in A is transferred to B and the contents of memory register X are inserted in A.

<u>36 Transfer on Busy Bit X</u>

$|a'f_{50}P_1|$: $0 \Rightarrow T$

$P_2|$:

$P_3|$:

$P_4|$: $BB'\{(D) \Rightarrow C\} + BB\{(D) \Rightarrow Ad[R]\}$

<u>00</u>

$$\begin{tikzcd}[row sep=small] {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

This is a branching order depending upon the state of BB, the busy bit. Its operation is identical to "transfer on negative". The function of the busy bit will be described below in connection with the "display" order.

<u>37 Transfer on Overflow X</u>

$|a'f_{44}P_1| : 0 \Rightarrow T$

$P_2| :$

$P_3| :$

$P_4| : A'_{OF}\{(D) \Rightarrow C\} + A_{OF}\{(D) \Rightarrow Ad[R]\}$

<u>00</u>

$$\begin{tikzcd}[row sep=small] |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

This, again, is a branching order identical in operation to "transfer on negative" and "transfer on busy bit" with the decision based upon the state of A_{OF}, the overflow flip-flop.

<u>40 Display X</u>

$|a'f_{42}P_1| : 0 \Rightarrow T$

$P_2| : (N\langle I\rangle) \Rightarrow U^x, (A) \Rightarrow U^y$

$P_3| :$

$P_4| : (D) \Rightarrow C$

<u>00</u>

$$\begin{tikzcd}[row sep=small] |af_2P_1| : \arrow[d] \\ P_4| : \end{tikzcd}$$

The "display" order causes a point to be displayed on the oscilloscope with abscissa given by the number in register X and ordinate by the number in A. U^x and U^y are the horizontal and vertical deflection registers of the scope. Time of deflection is roughly 50μsec.

Now referring to the transfers common to all cycles at the beginning of this section, we observe a pair of transfers during P_4 conditional on a function ν that is true for this and the five succeeding orders. All of these orders are concerned with terminal equipments that have operation times independent of the timing of the computer proper. The first transfer says that if ν is true, the state of the busy bit is transferred to Q. Thus if the busy bit is in the "one" state, signifying that one of the terminal devices is in use, the computer is halted. If the busy bit is in the "zero" state, the computer is started if it had previously been halted. The second transfer says that if ν is true and the busy bit is in the "zero" state, the busy bit is set to a "one". The busy bit is set to "zero" by one of the terminal devices when its operation is concluded.

Thus if, say two, consecutive "display" orders appear in a program, the computer will be halted at the end of the cycle during which the second order is taken from memory, and will restart in cycle 40 after the first display has been completed.

41 Display Word X

$|a'f_{52}P_1|$: $0 \Rightarrow T$

$P_2|$: $(N\langle T\rangle) \Rightarrow U^x$, $(A) \Rightarrow U^y$, $(B) \Rightarrow U^c$, $11_{octal} \Rightarrow U^s$

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

$\underline{00}$

$|af_2P_1|$:

During this order the octal content of the word in B is displayed at (x, y) given, respectively, by the contents of registers X and A. U^c is a register in the display unit holding the word to be displayed. U^s is step counter in the display unit.

The order takes about 300 μsec for execution and is under control of the busy bit.

42 Index Camera

$|a'f_{53}P_1|$: $0 \Rightarrow T$, index camera

$P_2|$:

$P_3|$:

$P_4|$: $(D) \Rightarrow C$

$\underline{00}$

$|af_2P_1|$:

$P_4|$:

This order causes the film in the oscilloscope camera to be advanced one frame. The shutter is closed just before the film advance and opened after the film advance. It is under control of the busy bit.

43 Read-In from H

$|a'f_{32}P_1|$: $(H) \Rightarrow H[B]$

$P_2|$:

$P_3|$:

$P_4|$:

This order is still incomplete. A 6-bit word from the tape reader appears in H, whence it is transferred to bits 0 to 5 of B.

44 Punch

$|a'f_{33}P_1| :$

$P_2| : (H[B]) \Rightarrow H$

$P_3| :$ punch

$P_4| :$

This order is incomplete. A word to be punched out on Flexowriter tape is transferred from H[B] to H and from there to the Flexowriter.

45 Print

$|a'f_{34}P_1| :$

$P_2| : (H[B]) \Rightarrow H$

$P_3| :$ print

$P_4| :$

This order is like the preceding one with a print command given to the Flexowriter. It is incomplete.

77 Exchange X

$$|a'f_{54}P_1| : 0 \Rightarrow T,\ 0 \Rightarrow A,\ (A) \Rightarrow L^j, \quad j = 0, 1$$

$$\oplus \sum_0^9 A_i \Rightarrow P_{BI}[L^j],\ \oplus \sum_{10}^{24} A_i \Rightarrow P_{Ad}[L^j],\ \Lambda_{BI}<\Gamma>,\ \Lambda_{Ad}<\Gamma>$$

$P_2| : (N<\Gamma>) \Rightarrow R,\ \Lambda_{BI}<\Gamma>,\ \Lambda_{Ad}<\Gamma>$

$P_3| :$

$P_4| : (D) \Rightarrow C,\ (A) + (R) \Rightarrow A, \chi_o A'_o R'_o + \chi'_o A_o R_o \Rightarrow A_{OF}$

00

$$\begin{tikzcd} {|af_2P_1| :} \arrow[d] \\ {P_4| :} \end{tikzcd}$$

In this order the contents of the accumulator and register X are exchanged. The order is simply a combination of the orders "store X" and "clear and add X".

VI. THE CONTROL ORGANIZATION

The 39-bit code of the control-memory output register F defines singly the happenings that are part of the various f_j states. In addition to this control command, appropriate clock pulses during P_1, P_2, P_3 and P_4 intervals are needed to execute the command. To each network proper s_1,s_2,s_3 and s_4 pulses must be directed. It should be recalled that the various f_jP_i statements imply that the indicated transfers take place by the end of a particular 3-μsec period. Each f_j produces a unique state in the 39-bit control register F, which holds this for 12 μsec. After this time it may be repeated, depending upon the function λ. F is grouped in an 11-number code. The first six bits of the F-code make up a 6-bit order code for the next memory cycle, barring repetition of the present memory cycle. This number is designated as G[F]. The next six bits of F make up the first number; actually these are individual bits since there are only six configurations needed to be resolved. The next three bits of F make up octal number O_1[F]. The following three make up octal number O_2[F]. This continues up to O_9[F].

Table III is a list of the representation used in the F-register and Table IV presents the control-memory representation code.

VII. CONSOLE CONTROL

Several modes of starting and stopping the computer and of inserting information into the computer are provided. Information may be read in via punched paper tape or manually with the aid of two front-panel toggle-switch registers one of which holds the word to be stored and the other, the address in memory at which it is to be stored. A separate start button is provided for each of these modes. Both read-in programs are stored in a fixed memory that forms part of the third memory bank $\left((\Gamma) = 2\right)$.

Sixteen toggle-switch registers are provided on the front panel for general use. These also comprise a portion of the third memory bank (registers 20,000 to 20,017 octal). A start button is provided which takes the first instruction from the first toggle-switch register.

Provision is made to start the computer at a preselected address Y written into the program tape. The read-in program inserts the order "transfer to Y" into register 0 of the first core bank. The start button start program takes the first instruction from address 0.

A starting mode is available enabling one to start a program from some previous stopping point. This button is labeled "restart".

A program may be halted in one of three ways. First, the halt may be programmed. Second, depressing a halt button stops the program at its current point. Third, a toggle-switch panel register labeled "halt address" may be used to stop the program at the address in the register.

For test purposes, a program may be run one instruction at a time, one memory cycle at a time, or one pulse at a time.

A. Starting the Computer

We have indicated five different starting modes: (1) restart, (2) start program, (3) start at 20,000 (first toggle-switch register), (4) start tape read-in and (5) start manual read-in.

For the first starting mode, the assumption is made that the computer has been in operation and has been stopped for some reason. All that is necessary is to reset the start-stop flip-flop Q or $0 \Rightarrow Q$. For the other starting modes represented by start at X, the following sequence of events must occur:

$$X \Rightarrow C \; ; \; 0 \Rightarrow G$$

$$(Cm\langle G\rangle) \Rightarrow F$$

$$0 \Rightarrow Q \quad .$$

During the first interval, the proper starting address is inserted in the memory address register. Also G is cleared, selecting the control word represented by f_2 which denotes an instruction read-in only. After this control word has been transferred to F, Q is cleared and the program begins. The address X is for mode 2, register 0; for mode 3, register 20,000; for mode 4, register 20,040; for mode 5, register 20,021.

B. Manual Read-In

The manual read-in program is given below.

20,021	Index (A)	377
20,022	Transfer to	20,024
20,023	Halt	–
20,024	Clear and add	20,020
20,025	Store (A) in	–
20,026	Transfer on index to (1A)	20,023

Registers 20,021 to 20,024 and 20,026 are in the fixed memory. Register 20,020 is the front toggle-switch register in which the word to be read in is stored. Register 20,025 is a hybrid front-panel register. The address part is in toggle switches; the index and instruction parts are wired in. Ad[20,025] is the register that contains the address into which (20,020) is stored. Registers 20,020 and Ad[20,025] may be converted from toggle switches to a keyboard without affecting the rest of the computer.

C. Stopping

To stop the computer during operation, the halt button is depressed, which inserts a "1" into Q. To stop at a specified address, this address is set up on the front-panel, halt-address, toggle-switch register which we label HA. A switch is provided with this register which determines whether or not the computer is to stop at the indicated address. We define a proposition SW_H that is true if this switch is in the state requiring the computer to stop at the address in HA. Then we have

$$|af_jP_4| : SW_H\{(HA) = (C^2)\} \; 1 \Rightarrow J_2 + SW_H\{(HA) \neq (C^2)\} \; 0 \Rightarrow J_2 \; ; \; (J_2) \Rightarrow Q \quad .$$

During all instruction read-in cycles ($a = 1$) at P_4, if there is match between the contents of HA and those of C^2, a flip-flop J_2 is set. Then during the next cycle with $a = 1$, the computer is stopped and J_2 is reset.

To run the computer one instruction at a time, a switch is thrown making a function SW_1 true. Then $|af_jP_4| : SW_1 \Rightarrow Q$.

To run the computer one memory cycle at a time, another switch is thrown making a function SW_c true. Then $|f_jP_4| : SW_c \Rightarrow Q$.

ACKNOWLEDGMENT

The authors are grateful to the following members of Group 24 for their suggestions and criticisms: B. Jensen, J. Henry, F. Nagy, E.W. Bivans, J.S. Arthur, F.G. Popp, R.E. McMahon, F.L. McNamara and R.H. Baker.

TABLE III

F-REGISTER REPRESENTATION

			Individual Bits $F_7 - F_{12}$
1	F_{12}	P_1	$(C^2) \Rightarrow D$
		P_2	$(D) + 1 \Rightarrow D$
		P_3	$\{(G) = 37\}'\ SW'_{OF}\ \{(A_{OF}) \Rightarrow^{\text{OF alarm FF}}_{Q}\}, \{(G) = 37\}'\ \{0 \Rightarrow A_{OF}\}$
		P_4	$\mu(Ad[N\langle\Gamma\rangle]) + \mu'\left\{(Ad[N\langle\Gamma\rangle]) + \left(S\left\langle B[N\langle\Gamma\rangle]\right\rangle\right)\right\} \Rightarrow \substack{C \\ T}$
			$\mu'(R) + \mu(N\langle\Gamma\rangle) \Rightarrow R \quad , \quad 0 \Rightarrow E$
2	F_{11}	P_2	$(N\langle\Gamma\rangle) \Rightarrow R$
4	F_{10}	P_4	$(D) \Rightarrow C$
10	F_9	P_1	$0 \Rightarrow T$
20	F_8	P_1	$0 \Rightarrow A$
40	F_7	P_4	$24_{\text{octal}} \Rightarrow T$
			$O_1[F]$
1		P_1	$(Ad[R]) \Rightarrow Ad[L^j]\ ;\ \oplus \sum_{10}^{24} R_i \Rightarrow P_{Ad}[L^j]$
		$P_1 - P_4$	$\Lambda_{Ad}\langle\Gamma\rangle$
2		P_1	$(W^1) \Rightarrow L^j,\ \oplus \sum_{0}^{9} W_i^1 \Rightarrow P_{BI}[L^j],\ \oplus \sum_{10}^{24} W_i^1 \Rightarrow P_{Ad}[L^j]$
		$P_1 - P_4$	$\Lambda_{Ad}\langle\Gamma\rangle$
3		P_1	$(D) \Rightarrow Ad[L^j]\ ;\ \oplus \sum_{10}^{24} D_i \Rightarrow P_{Ad}[L^j]$
		$P_1 - P_4$	$\Lambda\langle\Gamma\rangle$
		P_4	$(Ad[R]) \Rightarrow C$
4		$P_1 - P_4$	$\{(T) = 0\}'\ \{(T) - 1 \Rightarrow T\}$
5		P_1	$(A) \Rightarrow L^j,\ \oplus \sum_{0}^{9} A_i \Rightarrow P_{BI}[L^j],\ \oplus \sum_{10}^{24} A_i \Rightarrow P_{Ad}[L^j]$
		$P_1 - P_4$	$\Lambda_{Ad}\langle\Gamma\rangle$
6		P_1	$(W^2) \Rightarrow L^j\ ;\ \oplus \sum_{0}^{9} W_i^2 \Rightarrow P_{BI}[L^j],\ \oplus \sum_{10}^{24} W_i^2 \Rightarrow P_{Ad}[L^j]$
		$P_1 - P_4$	$\Lambda_{Ad}\langle\Gamma\rangle$
		P_4	$(Ad[R]) \Rightarrow C$
7			

TABLE III (Continued)

		$\underline{O_2[F]}$
1	P_4	$0 \Rightarrow B$
2	P_2	$0 \Rightarrow B$, $(A_o) \Rightarrow E$
3	P_1	$1 \Rightarrow$ camera, SW'_c {no film $\Rightarrow Q$}
4	P_3	$\{A_o \oplus R_o\}' \{(A) - (R)\} + \{A_o \oplus R_o\} \{(A) + (R)\} \Rightarrow A$
	P_4	$B_o(\bar{A}) + B'_o(A) \Rightarrow A$
5	P_1	$(D) \Rightarrow Ad[R]$, $(C^2) \Rightarrow D$
	P_2	$(D) + 1 \Rightarrow D$
6		
7		
		$\underline{O_3[F]}$
1	P_1	$(A_n) \Rightarrow B_{n-1}$; $0 \Rightarrow B_{24}$; $(A_o) \Rightarrow \overset{Q}{\text{square-root alarm FF}}$
	P_2	$(A,B) \Rightarrow Shl\ A,B$, $0 \Rightarrow R$
	P_3	$(A,B) \Rightarrow Shl\ A,B$, $(A_{24} + B_o) \Rightarrow R_{24}$
	P_4	$(R_{24}) \Rightarrow D_{24}$; $0 \Rightarrow D;\bar{D}_{24}$; $((A) - (R),B) \Rightarrow Shl\ A,B$; $32_{octal} \Rightarrow T$
2	P_1	$(A,B) \Rightarrow Shl\ A,B$; $(D_n) \Rightarrow R_{n-2}$, $1 \Rightarrow R_{24}$
	P_2	$(A) - (R) \Rightarrow A$
	P_3	$(D) \Rightarrow Shl\ D$, $(A'_o) \Rightarrow D_{24}$, $A_o((A) + (R),B) + A'_o(A,B) \Rightarrow Shl\ A,B$
3	P_4	$A'_o \{(D) \Rightarrow C\} + A_o\{(D) \Rightarrow Ad[R]\}$
4	P_1	$(D) \Rightarrow Ad[R]$
5	P_1	$(Ad[R]) \Rightarrow V\langle B[R]\rangle$, $0 \Rightarrow S\langle B[R]\rangle$
6	P_3	$0 \Rightarrow R$; $\bar{R}_{24}$, $(B_o) \Rightarrow R_{24}$
7		

TABLE III (Continued)

		$\underline{O_4[F]}$
1	P_1	$(T) - 1 \Rightarrow T$
2	P_1	$(D_n) \Rightarrow B_{n-1}$
3	P_1	$(H) \Rightarrow H[B]$
4	P_4	$\left\{\prod_0^{24} A_i'\right\}' \{(D) + 1\} + \left\{\prod_0^{24} A_i'\right\} (D) \Rightarrow D$;
		$SW_i' \left\{\prod_0^{24} A_i'\right\}' \left\{1 \Rightarrow \begin{matrix} Q \\ \text{Id alarm FF}\end{matrix}\right\}$
5	P_2	$(H[B]) \Rightarrow H$
6	$P_1 - P_4$	$\wedge_{BI} \langle\Gamma\rangle$
7		
		$\underline{O_5[F]}$
1	P_4	$(A) - (R) \Rightarrow A,\ \chi_o A_o' R_o + \chi_o' A_o R_o' \Rightarrow A_{OF}$
2	P_4	$(A) + (R) \Rightarrow A,\ \chi_o A_o' R_o' + \chi_o' A_o R_o \Rightarrow A_{OF}$
3	P_4	$A_{OF}'\{(D) \Rightarrow C\} + A_{OF}\{(D) \Rightarrow Ad[R]\}$; $0 \Rightarrow A_{OF}$
4	P_4	$BB'\{(D) \Rightarrow C\} + BB\{(D) \Rightarrow Ad[R]\}$
5	P_1	$(S<B[R]>) + 1 \Rightarrow S<B[R]>$
	P_2	$R_o(S<B[R]>) + R_o'\{(S<B[R]>) + 1\} \Rightarrow S<B[R]>$
	P_4	$\{(S<B[R]>) = (V<B[R]>)\}\ \{(D) \Rightarrow C\}$
6		
7		

TABLE III (Continued)

		$\underline{O_6[F]}$
1	P_1	$0 \Rightarrow D$
	P_4	$60_{octal} \Rightarrow T$
2	P_4	$A_o'\{(A) + (R)\} + A_o(A) \Rightarrow A$
3	P_4	$(B_n) \Rightarrow A_{n+1}, (A_o) \Rightarrow A_o$
4	P_3	$(A) - (R) \Rightarrow A$
5	P_3	print
6	P_3	punch
7		
		$\underline{O_7[F]}$
1	$P_2 - P_4$	$\{A_o \oplus A_i\}' \{(D) + 1\} \Rightarrow D, \Sigma_I$
2	$P_1 - P_4$	"
3	P_3	$(A_k R_k) \Rightarrow A_k \quad k = 0, 1 \ldots 24$
4	P_1	$(A) \Rightarrow B$
5	P_4	$1 \Rightarrow Q$
6	P_2	$(B) \Rightarrow U^c, 11_{octal} \Rightarrow U^s$
7		

TABLE III (Continued)

		$\underline{O_8[F]}$
1	P_2	$(N\langle\Gamma\rangle) \Rightarrow U^x,\ (A) \Rightarrow U^y$
2	$P_1 - P_4$	$\{(T) = 0\}'\ \left\{\Sigma_l,\ \{A_o \oplus A_1\}\ \{1 \Rightarrow A_{OF}\}\right\}$
3	$P_1 - P_4$	$\{(T) = 0\}'\ \Sigma_r$
4	$P_1 - P_4$	$\{(T) = 0\}'\ \psi_l$
5	$P_1 - P_4$	$\{(T) = 0\}'\ \psi_r$
6		
7		
		$\underline{O_9[F]}$
1	$P_3 - P_4$	$\times$
2	$P_1 - P_4$	$\times$
3	$P_1 - P_2$	$\times$
	P_3	$Y,\ (R_o) \Rightarrow E$
	P_4	$\{A_o B_{24} E\}\ \{1 \Rightarrow \overset{Q}{\text{M-D alarm FF}}$
4	P_3	Z_A
	P_4	$Z\ ;\ \{E \oplus (B_{24} + R_o \Pi A_i')\}'\ 1 \Rightarrow \overset{Q}{\text{divide alarm FF}}$
5	$P_1 - P_4$	Z
6	$P_1 - P_2$	Z
	P_3	$Z_B,\ (B_n) \Rightarrow A_{n-1},\ 0 \Rightarrow R; \bar{R}_{24},\ (R_o) \Rightarrow R_{24}$
	P_4	$R_{24}'(\bar{A}) + R_{24}\{(A) + (R)\} \Rightarrow A$

TABLE IV
CONTROL-MEMORY CODE

G Code	Order	f No.	G[F]	$F_7 - F_{12}$	O_1[F]	O_2[F]	O_3[F]	O_4[F]	O_5[F]	O_6[F]	O_7[F]	O_8[F]	O_9[F]
01	Add X	1		16					2				
02	Subtract X	3		16					1				
03	Clear and add X	4		36					2				
04	Clear and subtract X	5		36					1				
05	Multiply nonroundoff X	6	50	76							4		1
06	Multiply roundoff X	$\tilde{6}$	60	76							4		1
07	Multiply and shift X	$\tilde{\tilde{6}}$	70	76							4		1
10	Divide X	10	52	56		2							4
11	Square root	38	55	34			1						
12	Subtract magnitudes X	13	54	16		4					4		
13	Extract X	15		16							3		
14	Identify X	16	62	12				4		4			
15	Shift left n	17		4	4							2	
16	Shift right n	18		4	4							3	
17	Cycle left n	19		4	4							4	
20	Cycle right n	20		4	4							5	
21	Scale factor X	21	64	10			4			1	1		
22	Store X	24		14	5			6					
23	Replace address X	25		14	5								
24	Transfer X	26		1			4						
25	Transfer negative X	27		10			3						
26	Transfer index X	28		10					5				
27	Index n	30		14			5						
30	Return from X	29		14	1								
31	Halt	31		14							5		
32	Clear B	36		14		2							
33	Store one X	47		14	2			6					
34	Store both X	48	67	14	2	5		6					
35	Shift and add X	51		36					2		4		
36	Transfer busy bit X	50		10					4				
37	Transfer overflow X	44		10					3				
40	Display X	42		14								1	
41	Display word X	52		14							6	1	
42	Index camera	53		14		3							
43	Read-in from H	32						3					
44	Punch	33						5		6			
45	Print	34						5		5			
46													
47													
50		7	51		4								2
51		8		1									3
52		11	53		4								5
53		12		1		1							6
54		14		1			6		2				
55		39	57				2	1					
56													
57		41		21		1		2		3			
60		$\tilde{7}$	61		4								2
61		9		1		1	6			2			3
62		35		4									
63													
64		22	65		4						2		
65		23		10	3								
66													
67		49			6			6					
70		$\tilde{\tilde{7}}$	71		4								2
71		37		1		1				3			3
72													
73													
74													
75													
76													
77	Exchange X	54		36	5			6	2				
00		2		1									