[illegible]

INTERIM PROGRAMMERS' MANUAL FOR MANIAC III

THE UNIVERSITY OF CHICAGO

THE INSTITUTE FOR COMPUTER RESEARCH

INTERIM PROGRAMMERS' MANUAL
FOR
MANIAC III

prepared by Mary Lu Lind
with the assistance of Mr. Robert Ashenhurst,
Mr. David Jacobsohn, and Mr. Herbert Kanner.

This manuscript is to function as a programmers' manual until a more complete document on Maniac III is available.

May 26, 1961

INTRODUCTION

The Introduction contains information on the register structure of Maniac III, on the ground rules of arithmetic, and on the trapping procedures. Standard nomenclature and symbolism used in Maniac III literature are presented.

EXTERNAL AND INTERNAL STATES

Maniac III is at all times in one of two states: external state or internal state. When in external state, the computer is under control of either the tape reader—in which case we say that it is in external state - tape—or the console typewriter—in which case we say that it is in external state - keyboard. When the computer is in external state, symbols in the range 0 through f are read in the same manner as is called for by Instruction 70 (RET). Ten other symbols exert control functions as shown in the table in Appendix II. All other symbols are ignored.

When in internal state, the computer is either idling or is running under the control of a stored program. We thus say it is in internal state and idling or internal state and running.

Switching between external and internal states is accomplished by the execution of appropriate instructions or by the use of equivalent console buttons. Instruction 74(SWE) leaves the computer in external state-tape or in external state-keyboard. The console "tape" and "keyboard" buttons put the computer in external state-tape and external state-keyboard respectively. Instruction 00 (STP) or the console "stop" button always leaves the computer in internal state and idling. Instruction 01 (RUN) or the console "run" button leaves the computer in internal state and running.

There is one important difference between the console buttons and the equivalent instructions: an interlock disables all buttons but the "stop" button except when the computer is in internal state and idling.

REGISTER STRUCTURE, 48-BIT REGISTERS

Maniac III has three 48-bit arithmetic registers: U, R, and S. ("U" stands for "universal" and "R" for "residual".) The results of arithmetic and logical operations usually appear in U or in U and R. S is a service register used to hold operands fetched from memory, and for other purposes not of direct interest to the programmer.

Maniac III has an 8,192 word core memory and has eight transistor memory registers, $T_7, T_6, \ldots, T_0$. The transistor memory registers have a shorter access time than the core memory registers. The computer has three core memory registers known as trap locations. (See trapping below.)

Maniac III has one register, W, that is permanently wired to contain 48 bits 1 and has one register, Z, that is permanently wired to contain 48 bits 0. Although the contents of W or Z may be fetched, an attempt to store into either of these registers is ignored by the computer.

All of the registers mentioned thus far are addressable. The address assignments in sexadecimal are given in Appendix III.

When we wish to discuss sections of a 48-bit register, we number the bit positions in the register from 0 through 47 from right to left and subscript the register name. Single bit positions of U are:

$$U_{47}, U_{46}, \ldots, U_0.$$

Sections of U are identified by writing:

$$U_{47\ldots42},$$

$$U_{45\ldots0}, \text{ etc.}$$

Similarly, we may speak of $T_{0,47}$ or of $T_{0,47\ldots13}$ etc.

Numbers are represented in Maniac III with an 8-bit exponent part and a 40-bit fractional part. For convenience in discussing arithmetic procedures, an additional notation is available. U^E is the leftmost 8-bit section of U. U^F is the rightmost 40-bit section of

U. Bit positions in U^E are (from left to right):

$$U_7^E, \ldots, U_0^E$$

and in U^F are (from left to right):

$$U_0^F, \ldots, U_{39}^F.$$

Similarly, we may speak of T_0^E or of $T_{0, 39}^F$, etc.*

Each of the arithmetic registers U, R, and S possesses an additional bit to those mentioned above, U_X^F, R_X^F, and S_X^F. U_X^F, R_X^F, and S_X^F are immediately to the left of U_0^F, R_0^F and S_0^F, respectively. These bits are used to extend the numerical range of the registers in case of overflow and for the purpose of executing multi-precision arithmetic. They cannot be stored in memory.

REGISTER STRUCTURE: 14-BIT REGISTERS

Maniac III has eight 14-bit index registers, B_7, $B_6, \ldots, B_0$. Two of these registers are used for special purposes: B_7, called the sequence register, normally contains the address of the next instruction to be performed; B_3, called the reference register, normally contains the address which was in B_7 before the last effective jump instruction was performed.

Maniac III has two banks of 14 lights or "indicators" on the console, D^S and D^T, called respectively the sense indicators and the trap indicators. Any of the 28 indicators may be turned on (set to 1) or turned off (set to 0), or may be interrogated by the program through the use of appropriate instructions. D^S and D^T may each be stored. (See instructions

* Lower case letters are used to represent the contents of a register.
u is the contents of U.
$u_{33...20}$ is a group of binary values held in $U_{33...20}$.
u_{10} is a binary digit held in U_{10}.
Similarly we speak of u_0^F and u_7^E, etc.

6b, 6c, 6d, 6e.)

The sense indicators are entirely free to be used by the program. For example, each indicator may be used to represent a Boolean variable. The rightmost four sense indicators, called character indicators, are also used to store the results of test instructions. (See instructions, 4c, 4d, 4e, 4f). A table showing the manner of representing character test results is given in Appendix IV.

The trap indicators, D^T, are turned on by various conditions in the computer (see below on trapping). Thus it is recommended programming procedure not to use these indicators for purposes other than those associated with trapping control.

The Maniac III console has one bank of 14 sense switches, each of which has three positions: 0, 1, and "ignore". Their purpose is to permit manual intervention while the computer is in internal state and running. (See instructions 6b and 6c.)

The bit positions in 14-bit registers are numbered from right to left, 0 through 13. We speak of:

$$B_{K,13}, \ldots, B_{K,0} \text{ where } 0 \leq K \leq 7;$$
$$D^S_{13}, \ldots, D^S_0;$$
$$D^T_{13}, \ldots, D^T_0.$$

The character indicators are:

$$D^S_3, D^S_2, D^S_1, \text{ and } D^S_0.$$

ARITHMETIC: NUMBER REPRESENTATION

All arithmetic is performed on 48-bit operands with 8-bit exponents and 40-bit fractions. (The terms "fraction" and "coefficient" are used synonymously). Exponents are represented excess 128. The acceptable range of exponents is:

$$-128 \leq E \leq +127,$$

which in machine representation is $0 \leq E \leq +255$. Exponent -128, called e_Z, is used for a special purpose, to represent absolute zero. (Note that a zero fraction might have an exponent anywhere in the range $-127 \leq E \leq +127$. Such a number is called a "relative zero".) Adding to a number with exponent e_Z, subtracting, multiplying, and dividing, have the same result as adding a number to 0, etc. Thus if N is any number, and Z is a number with exponent e_Z, and Z_0 is that number whose exponent is e_Z and fraction is zero:

$$\begin{aligned} N \pm Z &= N \\ Z \pm N &= \pm N \\ Z \times N &= Z/N = Z_0 \\ Z + Z_0 &= Z \end{aligned}$$

(Division by Z, as well as by a relative zero, causes trapping.) In no other ways than those mentioned here can Z be created by an arithmetic operation (except by exponent overflow or underflow which causes trapping).

The leftmost fraction bit functions as a sign bit and is considered to be followed by a point. Negative fractions are represented as the 2-complements of their negations. Thus -0.0110...0 is represented as the 2-complement of +0.0110...0 or as 1.1010...0. Thus all positive fractions have a leftmost (sign) bit of 0 and all negative fractions have a leftmost (sign) bit of 1 and the acceptable range of fractions is:

$$-1 \leq F < 1.$$

Due to the 2-complement representation of negative numbers, a leading 1 in a negative number has the same function as a leading 0 in a positive number.* When a fraction is shifted to the right (to preserve significance

* Exception: in a fraction of -2^{-K} the rightmost leading 1 is a significant bit.

in floating point arithmetic, or to preserve the designated exponent in specified point arithmetic, or in instructions 20 and 21 which perform arithmetic shifts) leading 0's are introduced in a positive number, and leading 1's are introduced in a negative number.

ARITHMETIC: MODES

Maniac III can perform four modes of arithmetic: floating point, specified point, normalized, and basic.

Floating Point arithmetic is effected by Class 1 instructions. In the execution of a floating point arithmetic operation, the number of significant bits in the result is calculated as a function of the numbers of significant bits in the operands. The result of the operation is represented with the correct number of significant bits in the fractional part preceded by leading zeros—if the result is positive—or by leading ones—if the result is negative. In the case of addition and subtraction, simply omitting the normalization step that ordinarily follows these floating point operations leads to a representation of the result with the correct number of significant bits. (The only adjustment of the coefficient that ever occurs following a floating point addition or subtraction consists in a shift of one to the right to prevent coefficient overflow. In the case of multiplication and division, the result is represented with the same number of leading zeros (if positive, or ones, if negative) as that of the two operands possessing the greater number.

Specified point arithmetic is effected by Class 3 instructions. The specified point mode is a generalization of the more conventional fixed point mode. Before a specified point operation is carried out, one operand is in U and one operand is fetched from memory. The result of the operation is represented with the same exponent as the initial operand in U. Control of the position of the point, with respect to the fraction bits, is effected by control of the exponent.

Normalized arithmetic, of the same nature as the conventional floating point arithmetic, is effected in Maniac III by the use of one instruction, Instruction 23 (NOA). This instruction performs addition and subtraction, giving a normalized result. Special instructions for multiplication and division are not needed, since those in Class 1 always give a normalized result from normalized operands (except when normalization is impossible due to exponent underflow).

Basic, or multiprecision, arithmetic is effected by Instructions 26, 27, 28. These instructions enable one to do arithmetic with operands which have fractional parts longer than 40 bits.

ARITHMETIC: ROUNDING

Rounding of the results of addition, subtraction and multiplication is performed as follows:

if $u^F_{39} = 0$ and if any 1's were shifted from U into R during the operation, then u^F_{39} and r^F_0 are both set to 1.

The rounding procedure for division is described in the section of this manual on Class 1, floating point, instructions.

An unround, which is performed at the beginning of certain instructions (e.g., 20, 21), is performed as follows:

if $r^F_0 = 1$, r^F_0 is set to 0 and 1 is subtracted from the rightmost bit position of U.

* TRAPPING

Maniac III has three registers known as trap locations, and a bank of 14 indicators, D^T, called trap indicators. Corresponding to each trap indicator is a trap toggle. When the computer "traps"—interrupts the normal sequence of computation because an anomolous condition has occurred—a trap toggle is set to 1, a trap indicator is set to 1, and usually an instruction in a trap location is performed. The conditions which cause the computer to trap are called <u>trap conditions</u>.

Each trap is categorized as an:

<u>arithmetic trap</u>,
<u>instruction trap</u>, or
<u>real-time trap</u>

depending upon the nature of the condition causing the trap. Each of the three trap locations corresponds to one of these categories of traps.

An arithmetic trap is further categorized as a:

<u>coefficient trap</u>,
<u>exponent trap</u>,
<u>parameter trap</u>,
<u>operation trap</u>, or
<u>segment trap</u>.

There is no further categorization of instruction traps, and that of real-time traps has not yet been completed. A trap indicator (and the corresponding trap toggle) is assigned to each kind of arithmetic trap condition, to the instruction trap condition, and to each kind of real-time trap condition. A diagram of the assignments of trap indicators to trap conditions is given in Appendix V.

<u>Arithmetic Trap Procedure</u>: More than one arithmetic trap condition may occur during the performance of an instruction. When a trap condition arises, the appropriate toggle is set to 1. Whether or not a given condition causes an immediate interrupt is discussed below under the section on arithmetic trap conditions. When the interrupt occurs, all trap indicators corresponding to trap toggles containing 1 are set to 1. The next instruction to be performed is examined, and the remaining procedure depends upon its nature.

If it is not a Disable instruction, it is discarded,[*] the instruction in the arithmetic trap location is executed,[**] all arithmetic trap toggles are set to 0, and the arithmetic trap location is cleared to 0 (a Stop instruction).

If the next instruction to be performed is a Disable instruction, the appropriate trap toggle is set to 0. (The "appropriate" trap toggle is determined by the Disable instruction.) If any arithmetic trap toggles are still on, the above procedure is repeated—the next instruction to be performed is examined to determine whether or not it is a Disable instruction, etc.

It should be noted that if a Disable instruction is encountered when no arithmetic trap toggle is on, the instruction is ignored. Further note that a Disable instruction resets a trap toggle but not the associated trap indicator. It is thus possible to continue the program in sequence retaining the information that a trap condition has occurred.

<u>Instruction Trap Procedure:</u> The instruction trap toggle is set by the fetching of a tagged instruction (before it is performed). The instruction trap indicator is set to 1 also. The instruction in the instruction trap location is executed, the toggle is reset to 0, and the instruction trap location is cleared to 0 (a Stop instruction).

---

[*] It is assumed that the program will return to normal sequence after a correction routine initiated by the instruction in the trap location has been performed. At such a time the "discarded" instruction is refetched and computing continues normally.

[**] A distinction is made between executing an instruction and jumping to the instruction. When the instruction is executed, the sequence register is not changed. See the discussion of Class 6 instructions, especially 6b and 6c.

Under certain circumstances the tag on an instruction is ignored and trapping does not occur (neither the toggle nor the indicator is set to 1). They are as follows:

the tagged instruction is a Disable instruction;

the instruction is about to be discarded in favor of execution of the contents of the arithmetic trap location;

the instruction is in a trap location;

the instruction is about to be performed as a direct consequence of the performance of an Execute instruction (instruction 6c).

interrupts

Real-time Trap Procedure: Real-time traps are associated with the operation of independently sequenced units, such as one or more sub-computers for input-output control, a real-time clock, and a console interrupt button. These have not been definitely assigned at this time.

When a real-time trap condition occurs, the appropriate trap toggle is set to 1. The interruption of computing does not occur until a fetch cycle on which the following conditions are satisfied:

no arithmetic trap toggle is on;

the fetched instruction is not tagged;

the fetched instruction is not a Disable instruction;

the fetched instruction is not to be performed as the result of the performance of an Execute instruction (6c);

the fetched instruction is not to be performed as the result of a trap;

the fetched instruction is not a Class 6 (index) instruction.

When these conditions are satisfied, the fetched instruction is discarded. All trap indicators corresponding to real-time trap toggles which are set to 1, are set to 1, and the toggles are reset to 0. The instruction in the real-time trap location is performed and the trap location is set to 0 (a Stop instruction).

<u>Correction of conditions causing trapping</u>: Instruction 6f is designed primarily for use in the trap locations. It facilitates a jump to an appropriate subroutine to correct the trapping condition, and at the same time records the location thus far reached in the main program. The correction subroutine may be designed to return to the location thus recorded.

<u>Arithmetic Trap Conditions</u>: The conditions causing instruction and real-time traps have been mentioned immediately above. The conditions causing arithmetic traps are discussed here.

A coefficient overflow in u causes the coefficient overflow toggle to be set. (A coefficient overflows when a fraction bit is shifted into the sign bit position, i. e. bits have been lost at the left.) Before the trap occurs the performance of the current instruction is completed, and the sign bit is reset to match u_X.

The exponent trap toggle is set when the exponent in U tries to become greater than +127 or less than -127. (The exponent e_Z cannot legitimately be created by an arithmetic operation.) Before the trap occurs the performance of the current instruction is completed.

Due to the excess 128 representation of exponents, an exponent overflow or underflow leaves an unexpected (in sign and magnitude) result in the exponent portion of the register. When an exponent tries to become greater than +127, 1 adds to the rightmost bit of the exponent (mod 256) and the exponent becomes all zeros representing e_Z. When an exponent tries to become less than -127, it first becomes e_Z and then becomes +127, +126, etc.

On all shift instructions the shift parameter, n, may be either positive or negative, $-128 \leq n \leq +127$. (Unless otherwise specified for a particular instruction, if n is positive, the shift is a right shift, and if n is negative, the shift is a left shift.)

14 bit designators are used for shift parameters. When treated arithmetically, designators are considered (mod 2^{14}) and the range of values a designator may assume is thus $-2^{13} \leq \mathcal{d} \leq 2^{13}-1$. If a designator is to function as a shift parameter in the performance of an instruction and is not within the specified numerical range, the parameter trap toggle is set, no fetches or shifts are performed, and a parameter trap occurs immediately.

It should be noted that within the range of a shift parameter, the leftmost seven bits of a designator are identical and indicate the sign of the parameter, for example:

```
0000000 0000000  =   0
0000000 1111111  =  +127
1111111 1111111  =  -1
1111111 0000000  =  -128
```

INSTRUCTION FORMAT

This section presents the instruction format and the uses of the various fields in the format. Further standard notations to be used with reference to Maniac III are presented.

The operation trap toggle is set, when an operation cannot be properly carried out due to some anomolous condition (e. g. , a divisor of value 0, a negative square root operand, etc.). The point at which the trap occurs depends upon the instruction and is indicated in the descriptions of instructions which may cause the operation trap toggle to be set.

The segment trap toggle is set when an instruction which is not a basic arithmetic instruction (26, 27, 28) calls for the use of u as an operand when $u_X \neq u_0$. Before trapping occurs, u_0 is set to match u_X, and the instruction is carried out.

The operation trap toggle is set also if $r_X \neq r_0$ at the time of the store in the basic instruction, BAD(28). r_0 is changed to match r_X, the store is effected, and then trapping occurs.

INSTRUCTION FORMAT

A Maniac III instruction consists of 8 fields, structured as follows:

$$\underset{\varepsilon}{\boxed{1}}\quad \underset{\omega}{\boxed{\ 7\ }}\quad \underset{\gamma^L}{\boxed{1}}\quad \underbrace{\underset{\beta^L}{\boxed{\ 5\ }}\quad \underset{\alpha^L}{\boxed{\quad 14\quad }}}_{\delta^L}\quad \underset{\gamma^R}{\boxed{1}}\quad \underbrace{\underset{\beta^R}{\boxed{\ 5\ }}\quad \underset{\alpha^R}{\boxed{\quad 14\quad }}}_{\delta^R}$$

ε	Tag (1 bit)
ω	Operation Code (7 bits)
γ^L	Left Inflector (1 bit)
β^L	Left Modifier (5 bits)
α^L	Left Designator (14 bits)
γ^R	Right Inflector (1 bit)
β^R	Right Modifier (5 bits)
α^R	Right Designator (14 bits)

TAG

The tag bit may be either 0 or 1. If 0 it has no function. If 1 an instruction trap is initiated when the instruction is fetched.

OPERATION CODE

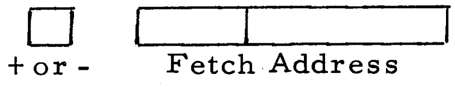
The left 3 bits of an operation code give the class of the instruction and the right 4 bits give the number of the instruction within the class. In the next section, each operation code is presented with a description of the function of an instruction in which the code is used.

INFLECTORS

An inflection bit can have one of seven interpretations, as follows:

+ or -	Plus or Minus
H or C	Hold or Clear
R or L	Right or Left
O or A	Operand or Address
J or P	Jump or Proceed
K or T	Keyboard or Tape
I or E	Ignore or Examine

The meaning of an inflection bit in its various contexts are as follows:



Depending upon the instruction in which a "±" inflection bit occurs, it is given one of three interpretations. It may be interpreted arithmetically: a number or its negation is fetched from memory. It may be interpreted formally in one of two ways: a word or its 48 bit 2-complement is fetched from memory; the operand or its 48 bit reflection is fetched from memory.*

*Henceforth, "±" is written in front of a symbol, e.g., ±m, indicating the choice of using m, or the negation of m. When "±" is written in front of a symbol and encircled, e.g. $\textcircled{\pm}$ m, it indicates the choice of using m, or its 48 bit 2's complement. When "±" is written over, instead of in front of a symbol, e.g., $\overset{\pm}{m}$, it indicates the choice of using the designated quantity or the reflection of the designated quantity.

H or C	Store Address	The contents of register U are stored in memory, after which U may (C) or may not (H) be cleared.
H or C	Fetch Address	An operand is fetched, and register U or, in some cases, register R may (C) or may not (H) be cleared before the operation proceeds.
R or L	Substitute Address	A 14-bit number (14-bit numbers are known as "designators") is substituted into either the right or left designator field of some memory location.
O or A	Designator	The modified designator functions as a 14-bit operand, or else gives the address of the location where, in the α^R field, the 14-bit operand resides.
J or P	(as left inflector in some instructions in class 6)	If γ^L is J: the instruction effects a jump if a specified condition is met, and the program proceeds in normal sequence if the condition is not met. If γ^L is P: the program proceeds in normal sequence if the condition is met, and the instruction effects a jump if the condition is not met.
K or T	(as right inflector in some instructions in class 7)	The instruction invokes either the console typewriter or the paper tape equipment.
I or E	(as left inflector)	At some point in the instruction the character of a specific set of bits may (E) or may not (I) be examined and the indicators D_3^S, D_2^S, D_1^S, D_0^S set accordingly.

DESIGNATOR MODIFICATION

A β field consists of five bits β_4, β_3, β_2, β_1, β_0, which serve to specify a subset of the set of four index registers B_0, B_1, B_2, B_3 (if $\beta_4 = 0$) or a subset of the set B_4, B_5, B_6, B_7 (if $\beta_4 = 1$). This is done in the obvious way. $B_{k+4\beta_4}$ is included in the set if $\beta_k = 1$ $(0 \leq k \leq 3)$, and excluded otherwise.

A β, α pair determines a modified designator, δ. The modified designator is computed by adding (modulo 2^{14}) the contents of the index registers specified in β to α (the instruction designator). The function of the modified designator, δ, depends upon the instruction in which β and α appear, and upon whether β and α are β^L, α^L or β^R, α^R in that instruction.

The β configuration 00000 is interpreted consistently with the above, i. e., to indicate that no index registers are specified. The configuration 10000 is used for a special purpose, to signify indirect addressing. If β is 10000, a new pair β', α' is obtained from the β^R, α^R field of the location whose address is α, and this new pair is used to determine a modified designator. If β' is 10000 a third pair β'', α'' is obtained, and the process can be repeated indefinitely.

The modified designator, δ, so defined is still subject to the O or A option in the inflection field in instructions which involve 14 bit operands. That is, in such instructions the β, α pair determines a designator which then is treated as an operand or an address depending upon the state of γ.

NOTATION

As was mentioned in the Introduction, capital letters are used as names of specific registers, and the corresponding lower

INSTRUCTIONS

This section contains a detailed description of the function of those Maniac III instructions planned by this time. Most of the instructions described are available for use now; those which are not available are listed in Appendix VI.

The instructions are discussed in the order of their Class and number within their Class, the classes being numbered and designated as follows:

0 Miscellaneous
1 Floating Point Arithmetic
2 Numerical Manipulation
3 Specified Point Arithmetic
4 Formal Manipulation
5 Data Transmission
6 Index Computation
7 Auxiliary System Operation

case letters designate the contents of the registers. For example, U is a register and u is the contents of that register. We further speak of M_ℓ and M_r as the memory locations whose addresses are given respectively by δ^L and δ^R in a particular instruction under discussion, and m_ℓ and m_r as the contents of M_ℓ and M_r respectively. The additional convention is used: if x is an address, then $\langle x \rangle$ is the contents of the location with that address. Thus, if δ^L is the address of M_ℓ, $\langle \delta^L \rangle = m_\ell$.

Designators — 14-bit numbers — are represented by "j" (jump addresses), "n" (shift parameters), "p" (patterns), "h" (increments), and "i" (operands).

The notation "i or $\mathcal{A}$ (i)", "n or $\mathcal{A}$ (n)", etc. meaning i or the address of i, etc., is used in discussions of instructions which have an O or A option in the inflection field.

The notation "∂ (B)" meaning selection of index registers, is used in the descriptions of Class 6 (index) instructions.

"χ ( )" is used to indicate that the character of the quantity designated by the letter in parenthesis is to be put in the character indicators.

CLASS O: MISCELLANEOUS

The performance of an instruction in this Class does not disturb u or r.

The operation code is the only relevant part of the format in all instructions in this Class; all other fields are ignored by the machine at run time.

No.	Mnemonic	Name	Description
0	STP	Stop	Discontinue externally or internally controlled action. Switch to internal control state.
1	RUN	Run	Switch to internal control state. Start running.
2	DCT	Disable Coefficient Trap	Reset coefficient trap toggle to 0.
3	DXT	Disable Exponent Trap	Reset exponent trap toggle to 0.
4	DPT	Disable Parameter Trap	Reset parameter trap toggle to 0.
5	DOT	Disable Operation Trap	Reset operation trap toggle to 0.
6	DST	Disable Segment Trap	Reset segment trap toggle to 0.
8	ICS	Indicate Coefficient Spill	$0 \rightarrow D_4^S$. If $u_X \neq u_0$, $1 \rightarrow D_4^S$.
f	IGN	Ignore	Do nothing.

CLASS 1: FLOATING POINT ARITHMETIC

There are four operations which can be effected by instructions in this class: add, multiply, divide, and store. The arithmetic operations are done in floating point (significant digit) arithmetic. Each instruction effects two operations, each of which is one of the given four. The sixteen instructions in class 1 are:

No.	Mnemonic	Name
0	FAA	Floating Add-Add
1	FAM	Floating Add-Multiply
2	FAD	Floating Add-Divide
3	FAS	Floating Add-Store
4	FMA	Floating Multiply-Add
5	FMM	Floating Multiply-Multiply
6	FMD	Floating Multiply-Divide
7	FMS	Floating Multiply-Store
8	FDA	Floating Divide-Add
9	FDM	Floating Divide-Multiply
a	FDD	Floating Divide-Divide
b	FDS	Floating Divide-Store
c	FSA	Floating Store-Add
d	FSM	Floating Store-Multiply
e	FSD	Floating Store-Divide
f	FSS	Floating Store-Store

In each class 1 instruction γ^L and δ^L are associated with the first operation to be performed, and γ^R and δ^R are associated with the second. If γ^L and δ^L (similarly for γ^R and δ^R) are associated with an arithmetic operation, γ^L specifies + or - and the modified designator, δ^L, gives the address of M_ℓ, where $\pm m_\ell$ is to be fetched as an operand. If γ^L and δ^L (similarly for γ^R and δ^R) are associated with a store operation, γ^L specifies H or C and the modified designator, δ^L, gives the address of M_ℓ, where $u \rightarrow M_\ell$ and U is held or cleared. For example, FAS has the format:

γ^L : + or -

δ^L : M_ℓ

γ^R : H or C

δ^R : M_r

Before an arithmetic operation (add, multiply, divide) is performed, one operand is in U and another is fetched from memory. R is then cleared, the operation is performed, and the result (in U, R) is rounded.

After an add or multiply operation is performed, including the round, r^E is set to be u^E -39 unless $u^E -39 < -127$ in which case r^F is shifted right n places and r^E is set to be -127 (where $u^E -39 + n = -127$).

Thus, to continue the above example, FAS effects:

(clear R)

$u + (\pm m_\ell) \rightarrow u$

$(u^E -39 \rightarrow r^E)$

$u \rightarrow M_r$

hold or clear U

The division process is performed in such a way that the division algorithm, $N = QD + R'$ where $|R'| < D$, is satisfied. However, the remainder may be negative. The sign of the remainder depends upon the signs of N and D and upon the rounding procedure.

When a divide instruction is given, the dividend, N, is in U and the divisor, D, is fetched from memory (thus the divisor is $\pm m$, where γ determines the sign, and δ is the address of M). $|N|$ is divided by $|D|$. The resultant quotient is in U and the remainder in R, and rounding proceeds as follows: if $r \neq 0$ and $U^F_{39} = 0$ then $1 \rightarrow U^F_{39}$ and $r - |D| \rightarrow R$. If N and D were of different sign, $-u \rightarrow U$. If N was negative, $-r \rightarrow R$. u is then the quotient, Q, and r is the remainder, R'.*

* Detailed specifications of the nature of the exponent of the remainder will be available in the near future.

An exponent trap occurs after any instruction which causes exponent overflow. The trap occurs after the instruction is performed, including the round. Thus, on the instruction FMA, if u^E overflows during multiplication, the multiplication is completed and the add effected before the trap occurs.

An operation trap occurs if division by zero is attempted. In this case the trap occurs before u is disturbed and after R is cleared.

A segment trap occurs if $u_X \neq u_0$ and a command in class 1 is given.

CLASS 2: NUMERICAL MANIPULATION

No.	Mnemonic	Name	Format
0	ADJ	Adjust	γ^L : + or -
1	SCL	Scale	δ^L : M_ℓ
			$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{n or } \ell \text{ (n)}\end{array}\right\}$ n

Clear R. If $m_\ell^E = e_Z$, clear U and continue to next instruction. otherwise, $\pm m_\ell \rightarrow$ U. $u^E -39 \rightarrow R^E$.

Instruction 0: Shift u^F and r^F as a 78 bit fraction (sign bits are not shifted) n places incrementing or decrementing u^E and r^E accordingly. When shift is complete, $u^E = m_\ell^E + n$ and $r^E = m_\ell^E + n - 39$. Round.

Shifting stops one step before coefficient overflow, exponent overflow, or exponent underflow in U; if n shifts have not been completed, u is rounded and an operation trap occurs. If necessary, r^F is shifted right to prevent exponent underflow in r^E.

The special case may arise in which $m_\ell = -1$ and γ^L is -. If so, when m_ℓ is fetched to U, coefficient overflow is prevented by a right shift of 1, accompanied by an increment of 1 in the exponent. If n = 0, u is rounded and the instruction has been carried out. If $n < 0$, u is rounded and an operation trap occurs indicating that n left shifts have not been effected. If $n > 0$, the shift to prevent coefficient overflow counts as one of the right shifts to be performed.

Instruction 1: Shift u^F and r^F as a 78 bit fraction (sign bits are not shifted) n places. Round.

If u^F overflows on a left shift, a coefficient overflow trap occurs after n shifts and the round have been effected.

The special case may arise in which $m_\ell = -1$ and γ^L is +. If so, unless $n > 0$ (in which case overflow is prevented by a right shift of u^F, r^F), a coefficient overflow trap occurs after n shifts and the round have been completed.

No.	Mnemonic	Name	Format
2	SGN	Sign	γ^L : + or -
			δ^L : M_ℓ
			γ^R : H or C
			δ^R : M_r

If $u \geq 0$, $\pm m_\ell \rightarrow U$. If $u < 0$, $\pm m_\ell \rightarrow U$. $u \rightarrow M_r$. Hold or clear U.

A coefficient trap occurs if a coefficient of -1 is negated. In this case, u_0 is set to match u_X before trapping occurs, thus leaving 0 in U.

r is not disturbed by the execution of this instruction.

3	NOA	Normalized Add	γ^L : + or -
			δ^L : M_ℓ
			γ^R : H or C
			δ^R : M_r

Clear R. $u \pm m_\ell \rightarrow U, R$. Round. $u \rightarrow M_r$. Hold or clear U.

The addition of u and m_ℓ is performed in such a manner that the final normalized fraction is correct to 78 bits. r^E is set to $u^E -39$ after u and m_ℓ have been added, the sum has been normalized, and u has been rounded. If $u -39 < -127$ then r^F is shifted right n places (where $u^E -39 + n = -127$) and r^E is set to -127.

NOA functions like the floating point arithmetic instructions in that coefficient traps cannot occur—a right shift accompanied by an increase in u^E corrects a coefficient overflow. In this case u^E may overflow and an exponent trap occur.

A segment trap occurs if $u_X \neq u_0$ when the instruction is read.

Left shifting of $u \pm m_\ell$ ceases if $u^E = -127$ before normalizing shifts are completed. In this case an operation trap occurs.

No.	Mnemonic	Name	Format
4	SQT	Square Root	γ^L : + or - δ^L : M_ℓ γ^R : H or C δ^R : M_r

Clear R. $\pm\sqrt{m_\ell} \rightarrow$ U, remainder $\rightarrow$ R. u $\rightarrow M_r$. Hold or clear U. $\sqrt{m_\ell}$ has the same number of significant bits as m_ℓ and is not rounded. The remainder in R is positive and is such that $u^2 + r = m$ (this equation gives the arithmetic properties of r, but does not suggest that m_ℓ can be retrived in the machine by squaring u and adding r).*

If m_ℓ is negative $\sqrt{-m_\ell}$ is computed and then an operation trap occurs.

No.	Mnemonic	Name	Format
5	RNV	Round to Nearest Value	γ^L : I or E δ^L : M_ℓ γ^R : H or C δ^R : M_r

Rounded u, r $\rightarrow$ U. If γ^L is E, $\chi(r^F)$. r $\rightarrow M_\ell$. u $\rightarrow M_r$. Hold or clear U.

* Detailed specifications of the nature of the exponent of the remainder are not available at this time.

In rounding u, r: if $r^F \geq 1/2$, 1 is added to the rightmost bit of u^F and r_0^F is set to 1;

if $-1/2 \leq r^F < 1/2$, no change is made;

if $r^F < -1/2$, then 1 is subtracted from the rightmost of bit of u^F and r_0^F is set to 0.

A coefficient overflow trap occurs if $r^F \geq 1/2$ and $u^F = 0.1\ldots1$, or $r^F < -1/2$ and $u^F = 1.0\ldots0$. In these cases r appears as it would have had there not been a coefficient overflow in U, and u_0^F is set to match u_X before trapping occurs. (In the former case then, $u^F = 0$ when trapping occurs, and in the latter case, $u^F = -2^{-39}$.)

No.	Mnemonic	Name	Format
6	BAA	Basic Add	γ^L : H or C
			δ^L : M_ℓ
			γ^R : + or -
			δ^R : M_r

It is assumed that one operand is in U and one operand is in S. U^E is ignored. Hold or clear U. $m_\ell \rightarrow$ R. $s \pm r$ is performed to give a double precision result. u^F is added in to the low order part of the result (unless the result has exponent e_Z). The result now in U, R is not rounded in any manner. $r \rightarrow M_r$. The exponent of the result (in both U^E and R^E) is the larger of the two exponents of the numbers originally in S and R.

No trapping can occur. However, both the original and final contents of U^F may be such that $u_X^F \neq u_0^F$ (i.e., u^F may be a number whose magnitude exceeds 1). This condition will cause a segment trap on a subsequent arithmetic instruction which is not a Basic instruction.

No.	Mnemonic	Name	Format
7	BAM	Basic Multiply	γ^L : H or C δ^L : M_ℓ γ^R : + or - δ : M_r

It is assumed that one operand is in U and one operand is in S. U^E is ignored. Hold or clear U. $m_\ell \rightarrow R$. $\pm(s^F \times r^F)$ is performed to give a double precision result and u^F is added in to the low order part of the result (unless the result has exponent e_Z. The result now in U, R is not rounded. $r \rightarrow M_\ell$.) The exponent of the result (in both U^E and R^E) is the sum of the exponents originally in S and R. If this exponent is not in the range $-127 \leq E \leq +127$, an exponent trap occurs (unless one operand had exponent e_Z).

Both the original and final contents of U^F may be such that $u_X^F \neq u_0^F$ (i. e., u^F may be a number whose magnitude exceeds 1). This condition will cause a segment trap on a subsequent arithmetic instruction which is not a Basic instruction.

8	BAD	Basic Divide	γ^L : + or - δ^L : M_ℓ γ^R : + or - δ^R : M_r

It is assumed that one operand is in U and one operand is in S. $m_\ell \rightarrow R$. R^E is ignored. $\pm$(u, r/(Normalized s)) $\rightarrow$ U, R (quotient $\rightarrow$ R and remainder $\rightarrow$ U). γ^R determines the sign of the remainder. $r \rightarrow M_r$. The exponent of the result (both in U^E and R^E) is the exponent of the number originally in U, minus the exponent of the number residing in S after normalization. An exponent trap occurs if this exponent is not in the range $-127 \leq E \leq +127$ (except when the dividend had exponent e_Z).

A segment trap occurs on the final store if $r_X \neq r_0$. In multiprecision work this can occur on the first division of a series only.

An operation trap occurs before the division is executed if the original $s^F = 0$.

No.	Mnemonic	Name	Format
b	SEX	Set Exponent	γ^L : + or -
c	AUX	Augment Exponent	δ^L : M_ℓ
			γ^R : H or C
			δ^R : M_r

Instruction b: $\pm m_\ell^E \rightarrow U^E$.

Instruction c: $u^E \pm m_\ell^E \rightarrow U^E$.

u^F remains unchanged. $u \rightarrow M_r$. Hold or clear U.

If γ^L is - and m_ℓ^E is -128, on instruction b, $-128 = e_Z \rightarrow U$. ($-e_Z = e_Z$.) If $u^E \pm m_\ell^E$ is not in the range $-127 \leq u^E \pm m^E \leq +127$, on instruction c, an exponent trap occurs.

r is not disturbed by the execution of these instructions.

d	CSX	Convert Signed Exponent	γ^L : + or -
			δ^L : M_ℓ
			γ^R : H or C
			δ^R : M_r

$\pm m^E$ as integer (exponent 39) $\rightarrow$ U. $u \rightarrow M_r$. Hold or clear U.

The excess 128 machine representation of exponents is compensated for in the conversion of the exponent to an integer. Thus, the converted exponent is in the range $-127 \leq u^F \leq +127$.

An operation trap occurs on an attempted conversion of e_Z.

r is not disturbed by the execution of this instruction.

No.	Mnemonic	Name	Format
e	CPD	Convert Positive Designator	γ^L : H or C
f	CSD	Convert Signed Designator	δ^L : M_ℓ $\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{i or } \mathcal{A}\text{(i)}\end{array}\right\\}$ i

Instruction e: i, considered positive, as integer (exponent 39) $\rightarrow$ U.
Instruction f: i, considered a signed integer, as integer (exponent 39) $\rightarrow$ U.

$u \rightarrow M_\ell$. Hold or clear U.

A designator, i, being a 14 bit entity, when considered positive is in the range $0 \leq i \leq 2^{14}-1$. When treated as signed integers, designators are considered mod 2^{14} with the leftmost bit functioning as a sign bit, e. g. :

$$\begin{array}{rcl} 0\ldots0 & \equiv & 0 \\ 01\ldots1 & \equiv & 2^{13}-1 \\ 10\ldots0 & \equiv & -2^{13} \\ 1\ldots1 & \equiv & -1 \end{array}$$

The range of a signed designator is thus $-2^{13} \leq i \leq 2^{13}-1$.

r is not disturbed by the execution of these instructions.

CLASS 3: SPECIFIED POINT ARITHMETIC

Any arithmetic instruction in this Class (add, multiply, divide), in contrast to the floating point arithmetic instructions, leaves a result in U which has the same exponent as the operand originally in U.

If listed, the instructions in this class, numbered 0 through f, would appear the same as those in Class 1, except that the mnemonics would begin with "S" instead of "F" and the names would be "Specified . . ." instead of "Floating . . .". The format of the instructions is the same as those in Class 1.

The sequence of operations effected by an arithmetic instruction at run time is the same as described for class 1 instructions: fetch operand; clear R; perform add, multiply, or divide; round u; adjust r^E. The division process is the same as described for Class 1 instructions. On all class 3 instructions, including "divide", the round is performed after u has been adjusted to the point that u^E is equal to the original exponent in U.

Exponent traps cannot occur on class 3 instructions. However, coefficient overflow may occur in which case a coefficient trap occurs after the total instruction has been executed.

An operation trap occurs if division by zero is attempted. The trap occurs before u is disturbed and after R is cleared.

A segment trap occurs if $u_X \neq u_0$ and a Class 3 command is given.

CLASS 4: FORMAL MANIPULATION

No.	Mnemonic	Name	Format
0	CON	Combine through Normal Mask	γ^L : H or C
1	COR	Combine through Reflected Mask	δ^L : M_ℓ
			γ^R : + or -
			δ^R : M_r

Instruction 0: $m_\ell \rightarrow R$.

Instruction 1: $\overline{m}_\ell \rightarrow R$.

$m_2 \rightarrow S$. Hold or clear U. $\overset{\pm}{u} \rightarrow U$. Insert a bit of s into U wherever the corresponding bit of r is 1.

No.	Mnemonic	Name	Format
2	SHN	Shift with Normal Connection	γ^L : H or C
3	SHC	Shift with Circular Connection	δ^L : M_ℓ
4	SHR	Shift with Reverse Connection	$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{n or } \lambda\text{ (n)}\end{array}\right\}$ n

These shifts are logical shifts, and thus the separation into exponent and fractional part is disregarded.

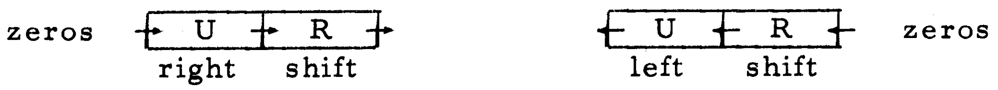
$m_\ell \rightarrow R$. Hold or clear U.

Instruction 2: Shift the contents of U and R as a 96 bit entity, u, r (r as the low order part), which shifts as a single unit. Bits which are shifted off the right of R are lost as are bits which are shifted off the left of U. e.g.:

on right shift of 1, $u_0 \rightarrow R_{47}$, r_0 is lost, $0 \rightarrow U_{47}$;

on left shift of 1, $r_{47} \rightarrow U_0$, u_{47} is lost, $0 \rightarrow R_{47}$;

where bit positions are numbered 0 through 47 from right to left in each of the U and R registers.

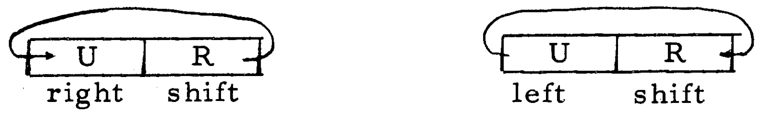


Instruction 3: Shift the contents of U and R as a 96 bit entity, u, r (r as the low order part), which shifts as a single unit. Bits shifted off either end of the double length register U, R are re-introduced at the other end. e.g. :

on right shift of 1, $u_0 \rightarrow R_{47}$, $r_0 \rightarrow U_{47}$;

on left shift of 1, $r_{47} \rightarrow U_0$, $u_{47} \rightarrow R_0$;

where bit positions are numbered 0 through 47 from right to left in each of the U and R registers.

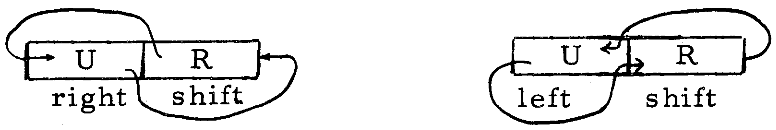


Instruction 4: Shift the contents of U and of R each treated as a 48 bit entity. The contents of both registers are shifted, but in opposite directions, u shifting in the direction indicated by the sign of the shift parameter. Bits shifted off the end of one register are introduced at the far end of the other register. e.g. :

on right shift of 1, $u_0 \rightarrow R_0$, $r_{47} \rightarrow U_{47}$;

on left shift of 1, $r_0 \rightarrow U_0$, $u_{47} \rightarrow R_{47}$;

where bit positions are numbered 0 through 47 from right to left in each of the U and R registers.



No.	Mnemonic	Name	Format
8	PSE	Position and Set	γ^L : + or -
9	PAU	Position and Augment	δ^L : M_ℓ
			$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{n or } \mathcal{A}(\text{n})\end{array}\right\}$ n

$m_\ell \rightarrow S$. Shift s with circular connection (i. e. bits shifted off one end of S are reintroduced at the other end) treating s as a 48 bit entity (the separation into exponent and fractional part is disregarded).

Instruction 8: $\pm s \rightarrow U$.

Instruction 9: $u \pm s \rightarrow U$.

r is not disturbed by the performance of these instructions.

No.	Mnemonic	Name	Format
c	TEW	Test Word	γ^L : R or L
d	TEC	Test Coefficient	δ^L : M_ℓ
e	TEX	Test Exponent	γ^R : H or C
f	TES	Test Segment	δ^R : M_r

Instruction c: $\chi(u)$

Instruction d: $\chi(u^F)$

Instruction e: $\chi(u^E)$

Instruction f: $\chi(u_X, u^F)$

Substitute the sense indicators, d^S, into M_ℓ, right or left designator field depending on γ^L. $u \rightarrow M_r$. Hold or clear U.

r is not disturbed by the performance of these instructions.

CLASS 5: DATA TRANSMISSION

The performance of an instruction in this class does not disturb u or r.

No.	Mnemonic	Name	Format
0	TRW	Transmit Word	γ^L : I or E
1	TRC	Transmit Coefficient	δ^L : M_ℓ
2	TRX	Transmit Exponent	γ^R : H or C
			δ^R : M_r

Hold or clear M_r.

Instruction 0: $m_\ell \rightarrow M_r$. (Obviously, the value of γ^R does not affect the performance of this instruction.)

Instruction 1: $m_\ell^F \rightarrow M_r^F$

Instruction 2: $m_r^E \rightarrow M_r^E$

No.	Mnemonic	Name	Format
4	TRS	Transmit Sign	γ^L : + or -
5	TRT	Transmit Tag	δ^L : M_ℓ
			γ^R : H or C
			δ^R : M_r

Hold or clear M_r

Instruction 4: $\overset{\pm F}{m}_{\ell,0} \rightarrow M^F_{r,0}$. (If γ^L is +, sign of $m_\ell \rightarrow$ sign position of M_r. If γ^L is -, reflected sign of $m_\ell \rightarrow$ sign position of M_r.)

Instruction 5: $\overset{\pm E}{m}_{\ell,7} \rightarrow M^E_{r,7}$. (If γ^L is +, tag of $m_\ell \rightarrow$ tag position of M_r. If γ^L is -, reflected tag of $m_\ell \rightarrow$ tag position of M_r.)

No.	Mnemonic	Name	Format
8	SUD	Substitute Designator	γ^R : R or L δ^L : M_ℓ $\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{i or } \mathcal{A}(\text{i})\end{array}\right\}$ i

i $\rightarrow M_\ell$, right or left designator field as indicated by γ^L .

CLASS 6: INDEX COMPUTATION

The performance of an instruction in this class does not affect u or r.

During the performance of all effective jump and execute instructions a step $j \rightarrow A$ occurs and a toggle, JT, is set. If at any point in the performance of an instruction b_7 is changed, a toggle, PA, is set (e. g. , in SEI if B_7 is specified in β^L then $\pm i \rightarrow B_7$ and $b_7 + h \rightarrow B_7$). The final step in the performance of an effective jump instruction is: $a \rightarrow B_7$. (Note that, as a result of this step, a change made in b_7 during the performance of an effective jump instruction is of no consequence.)

It may be that neither JT nor PA is on after an instruction has been performed (e. g. , any instruction not in class 6; a SEI which does not specify B_7 in β^L; a jump instruction which is not effective, i. e. , which proceeds). In that event, $b_7 + 1 \rightarrow B_7$ and the address of the next instruction to be performed is b_7 . If JT is on after an instruction has been performed, the address of the next instruction to be performed is a. If PA is on, the address of the next instruction to be performed is b_7 . JT overrides PA.

In the following descriptions, "B_k" represents any index register specified in β^L.

No.	Mnemonic	Name	Format
0	SEI	Set Index	γ^L : + or -
1	AUI	Augment Index	β^L : $\mathcal{S}$ (B)
2	TEI	Test Index	α^L : h
			$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{i or } \mathcal{A}\text{(i)}\end{array}\right\}$ i

Instruction 0: $\pm i \rightarrow B_k$; $b_k + h \rightarrow B_k$.

Instruction 1: $b_k \pm i \rightarrow B_k$; $b_k + h \rightarrow B_k$.

Instruction 2: $\chi(i \pm \Sigma b_k)$; $b_k + h \rightarrow B_k$.

No.	Mnemonic	Name	Format
4	JIE	Jump on Index Equal	γ^L : J or P β^L : $\mathcal{S}$(B) α^L : j $\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{i or } \mathcal{A}(\text{i})\end{array}\right\}$ i

$\Sigma b_k : i$.

Condition is met if $\Sigma b_k = i$.

If equality condition is not met, $b_k + 1 \rightarrow B_k$.

If jump is effective, $b_7 \rightarrow B_3$.

No.	Mnemonic	Name	Format
5	JIZ	Jump on Index Zero	γ^L : J or P
6	JIP	Jump on Index Positive	β^L : $\mathcal{S}$(B)
7	JIN	Jump on Index Negative	α^L : h $\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{j or } \mathcal{A}(\text{j})\end{array}\right\}$ j

$b_k + h \rightarrow B_k$.

$\Sigma b_k : 0$.

Instruction 5: condition is met if $\Sigma b_k = 0$.

Instruction 6: condition is met if $\Sigma b_k > 0$.

Instruction 7: condition is met if $\Sigma b_k < 0$.

If jump is effective, $b_7 \rightarrow B_3$.

No.	Mnemonic	Name	Format
8	JFC	Jump on Formal Comparison	γ^L : J or P
9	JAC	Jump on Arithmetic Comparison	δ^L : $M\ell$
a	JLC	Jump on Lexicographic Comparison	$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{j or } \mathcal{A}(\text{j})\end{array}\right\}$ j

u:m .

Instruction 8: condition is met if u and m_ℓ are identical when u and m_ℓ are considered as 48 bit entities.

Instruction 9: condition is met if $u \geq m_\ell$ when u and m_ℓ are considered as arithmetic entities with 8 bit exponents and 41-bit fractions.*

If $u^E = e_Z = m_\ell^E$, condition is met.

If $u^E = e_Z \neq m_\ell^E$, and $m_\ell^F < 0$, condition is met.

If $u^E \neq e_Z = m_\ell^E$, and $u^F \geq 0$, condition is met.

Instruction a: condition is met if $u \geq m_\ell$ when u and m_ℓ are considered as 48 bit integers.

If jump is effective, $b_7 \rightarrow B_3$.

No.	Mnemonic	Name	Format
b	JPC	Jump on Pattern Comparison	γ^L : J or P
c	EPC	Execute on Pattern Comparison	β^L : additional inflection
			α^L : p
			$\left.\begin{array}{l}\gamma^R : \text{O or A} \\ \delta^R : \text{j or } \mathcal{A}(\text{j})\end{array}\right\}$ j

In each bit position where there is a 1 in p, a sense or trap (determined by β_0^L) indicator is specified.

---

* If $u_X = u_0$, u^F is essentially 40 bits. If $u_X \neq u_0$, u^F is 41 bits and u_X functions as the sign bit. In this case m^F may be considered as having a 41st bit, $m_{\ell, X}^F$, which is identical to $m_{\ell, 0}^F$.

$\beta_4^L = 0$: condition is met if all specified indicators are set to 1.*

$\beta_4^L = 1$: condition is met if any specified indicator is set to 1.**

$\beta_3^L = 0$: set B_3 from $B_7 (b_7 \rightarrow B_3)$ if jump (or execute) is effective.

$\beta_3^L = 1$: do not set B_3 from B_7 if jump (or execute) is effective.

$\beta_2^L = 0$, $\beta_1^L = 0$: do not alter the state of specified indicators.

$\beta_2^L = 0$, $\beta_1^L = 1$: set specified indicators from sense switches.

$\beta_2^L = 1$, $\beta_1^L = 0$: set specified indicators to 0.

$\beta_2^L = 1$, $\beta_1^L = 1$: set specified indicators to 1.

$\beta_0^L = 0$: D^S specified

$\beta_0^L = 1$: D^T specified

If jump, or execute, is effective and $\beta_3^L = 0$, $b_7 \rightarrow B_3$.

Instructions b and c function alike except that $a \rightarrow B_7$ is omitted at the conclusion of the performance of an effective execute instruction. Thus one instruction out of sequence is executed and, unless that instruction is a jump instruction, the program returns to the original sequence. Furthermore, this (out of sequence) instruction which is executed will not cause an instruction trap to occur even if $\epsilon = 1$.

* If $\beta_4^L = 0$ and $\alpha^L = 0$, condition is met.

** If $\beta_4^L = 1$ and $\alpha^L = 0$, condition is not met.

No.	Mnemonic	Name	Format
d	SES	Set Sense Indicators	γ^L : R or L
e	SET	Set Trap Indicators	δ^L : M_ℓ
			$\left.\begin{matrix}\gamma^R : \text{O or A} \\ \delta^R : \text{i or } \mathcal{A}(\text{i})\end{matrix}\right\}$ i

Instruction d: $d^S \rightarrow M_\ell$, right or left designator field as indicated by γ^L. $i \rightarrow D^S$.

Instruction e: $d^T \rightarrow M_\ell$, right or left designator field as indicated by γ^L. $i \rightarrow D^T$.

f	JUN	Jump Unconditionally	γ^L : R or L
			δ^L : M_ℓ
			$\left.\begin{matrix}\gamma^R : \text{O or A} \\ \delta^R : \text{j or } \mathcal{A}(\text{j})\end{matrix}\right\}$ j

$b_7 \rightarrow M_\ell$, right or left designator field as indicated by γ^L .

CLASS 7: AUXILIARY SYSTEM OPERATION

The performance of an instruction in this class does not disturb u or r.

Fields γ^L, β^L, and α^L are <u>not</u> relevant in instructions in this class; these fields are ignored by the machine at run time.

No.	Mnemonic	Name	Format
0	RET	Read Tetrad Word	γ^R : K or T δ^R : M_r

Clear S. Read a tetrad (the rightmost four bits of a character) from the typewriter if $\gamma^R = K$, or from paper tape if $\gamma^R = T$, and place in the rightmost 4 bit positions of S. Read another tetrad, shift s left 4 places, and place tetrad in rightmost 4 bit positions of S. Continue reading in this manner until the symbol "/" is read. $s \rightarrow M_2$.

When this instruction is performed, any symbol read which is not between 0 and f, or is not "/", is ignored. Such a symbol is called a non-tetradic symbol.

1	REO	Read Octad	γ^R : K or T δ^R : M_r

Clear S. Read a single octad character from typewriter if $\gamma^R = K$, or from paper tape if $\gamma^R = T$, and place in the rightmost 7 positions of S. $s \rightarrow M_r$. (In most cases, the leftmost bit of an octad is a parity bit. Parity bits are not read into S.)

2	WRT	Write Tetrad Word	γ^R : K or T δ^R : M_r

$m_r \rightarrow S$. Write s on typewriter if $\gamma^R = K$, or on paper tape if $\gamma^R = T$, interpreting s as a string of 12 tetrads.

No.	Mnemonic	Name	Format
3	WRO	Write Octad	γ^R : K or T δ^R : M_r

$m_r \rightarrow S$. Shift s left 40 places. Write the character corresponding to the leftmost 8 bits of s on typewriter if $\gamma^R = K$, or on paper tape if $\gamma^R = T$. (The leftmost bit of S is ignored and a parity bit is synthesized before printing.)

No.	Mnemonic	Name	Format
4	SWE	Switch to External Control State	γ^R : K or T δ^R : i

$i \rightarrow$ effective address register. Switch to external control state, typewriter or tape.

APPENDIX I

CHARACTER SET

<u>lower case</u>

STOP	0	0	0	0	1	1	0	0
CR	1	0	0	0	1	1	0	1
TB	1	0	0	0	1	1	1	0
SP	0	0	0	0	1	1	1	1
0	1	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	0	1	0	0	1	0
3	1	0	0	1	0	0	1	1
4	0	0	0	1	0	1	0	0
5	1	0	0	1	0	1	0	1
6	1	0	0	1	0	1	1	0
7	0	0	0	1	0	1	1	1
8	0	0	0	1	1	0	0	0
9	1	0	0	1	1	0	0	1
a	1	0	0	1	1	0	1	0
b	0	0	0	1	1	0	1	1
c	1	0	0	1	1	1	0	0
d	0	0	0	1	1	1	0	1
e	0	0	0	1	1	1	1	0
f	1	0	0	1	1	1	1	1
g	1	0	1	0	0	0	0	0
h	0	0	1	0	0	0	0	1
i	0	0	1	0	0	0	1	0
j	1	0	1	0	0	0	1	1
k	0	0	1	0	0	1	0	0
l	1	0	1	0	0	1	0	1
m	1	0	1	0	0	1	1	0
n	0	0	1	0	0	1	1	1
o	0	0	1	0	1	0	0	0
p	1	0	1	0	1	0	0	1
q	1	0	1	0	1	0	1	0
r	0	0	1	0	1	0	1	1
s	1	0	1	0	1	1	0	0
t	0	0	1	0	1	1	0	1
u	0	0	1	0	1	1	1	0
v	1	0	1	0	1	1	1	1
w	0	0	1	1	0	0	0	0
x	1	0	1	1	0	0	0	1
y	1	0	1	1	0	0	1	0
z	0	0	1	1	0	0	1	1

<u>upper case</u>

STOP	1	1	0	0	1	1	0	0
CR	0	1	0	0	1	1	0	1
TB	0	1	0	0	1	1	1	0
SP	1	1	0	0	1	1	1	1
)	0	1	0	1	0	0	0	0
§	1	1	0	1	0	0	0	1
−	1	1	0	1	0	0	1	0
+	0	1	0	1	0	0	1	1
/	1	1	0	1	0	1	0	0
∨	0	1	0	1	0	1	0	1
∧	0	1	0	1	0	1	1	0
X	1	1	0	1	0	1	1	1
¬	1	1	0	1	1	0	0	0
(	0	1	0	1	1	0	0	1
A	0	1	0	1	1	0	1	0
B	1	1	0	1	1	0	1	1
C	0	1	0	1	1	1	0	0
D	1	1	0	1	1	1	0	1
E	1	1	0	1	1	1	1	0
F	0	1	0	1	1	1	1	1
G	0	1	1	0	0	0	0	0
H	1	1	1	0	0	0	0	1
I	1	1	1	0	0	0	1	0
J	0	1	1	0	0	0	1	1
K	1	1	1	0	0	1	0	0
L	0	1	1	0	0	1	0	1
M	0	1	1	0	0	1	1	0
N	1	1	1	0	0	1	1	1
O	1	1	1	0	1	0	0	0
P	0	1	1	0	1	0	0	1
Q	0	1	1	0	1	0	1	0
R	1	1	1	0	1	0	1	1
S	0	1	1	0	1	1	0	0
T	1	1	1	0	1	1	0	1
U	1	1	1	0	1	1	1	0
V	0	1	1	0	1	1	1	1
W	1	1	1	1	0	0	0	0
X	0	1	1	1	0	0	0	1
Y	0	1	1	1	0	0	1	0
Z	1	1	1	1	0	0	1	1

Continued on next page

<u>lower case</u>		<u>upper case</u>	
←	1 0 1 1 0 1 0 0	→	0 1 1 1 0 1 0 0
↑	0 0 1 1 0 1 0 1	=	1 1 1 1 0 1 0 1
<	0 0 1 1 0 1 1 0	>	1 1 1 1 0 1 1 0
≠	1 0 1 1 0 1 1 1	≡	0 1 1 1 0 1 1 1
}	1 0 1 1 1 0 0 0	{	0 1 1 1 1 0 0 0
,	0 0 1 1 1 0 0 1	:	1 1 1 1 1 0 0 1
.	0 0 1 1 1 0 1 0	▾	1 1 1 1 1 0 1 0
]	1 0 1 1 1 0 1 1	[	0 1 1 1 1 0 1 1

IGNORE:

- _ 0 0 0 0 0 0 0 through _ 0 0 0 1 0 1 1
- _ 0 1 1 1 1 0 0 through _ 0 1 1 1 1 1 1
- _ 1 0 0 0 0 0 0 through _ 1 0 0 1 0 1 1
- _ 1 1 1 1 1 0 0 through _ 1 1 1 1 1 1 1

The leftmost bit of each character representation is a parity bit. The parity bit does not read into the computer, nor is it necessary to synthesize a parity bit before punching or typing octads.

APPENDIX II

CONTROL CHARACTERS

control character		function
)	0 1 0 1 0 0 0 0	Read following characters.
§	1 1 0 1 0 0 0 1	$\langle a\rangle \to S$; $a+1 \to A$.
-	1 1 0 1 0 0 1 0	$a-1 \to A$.
+	0 1 0 1 0 0 1 1	$a+1 \to A$.
/	1 1 0 1 0 1 0 0	$s \to \langle a\rangle$; $a+1 \to A$; $0 \to S$.
∨	0 1 0 1 0 1 0 1	$s \to I'$; perform I'.
∧	0 1 0 1 0 1 1 0	$s \to A$; $0 \to S$.
×	1 1 0 1 0 1 1 1	$0 \to S$.
¬	1 1 0 1 1 0 0 0	$\langle a\rangle \to S$; $a+1 \to A$; type s as 12 tetrads.
(	0 1 0 1 1 0 0 1	Ignore all following characters except ")".

A is the effective address register and a is its contents. $\langle a\rangle$ is the contents of the register whose address is a. I' is a register, and "perform I'" means execute the instruction currently held in I'.

APPENDIX III

ADDRESS ASSIGNMENTS
(in sexadecimal)

register	address
Real-time Trap	3fed
Instruction Trap	3fee
Arithmetic Trap	3fef
T_0	3ff0
T_1	3ff1
T_2	3ff2
T_3	3ff3
T_4	3ff4
T_5	3ff5
T_6	3ff6
T_7	3ff7
U	3ff9
R	3ffa
S	3ffb
Z	3ffe
W	3fff
Core Memory	0000 through 1fff

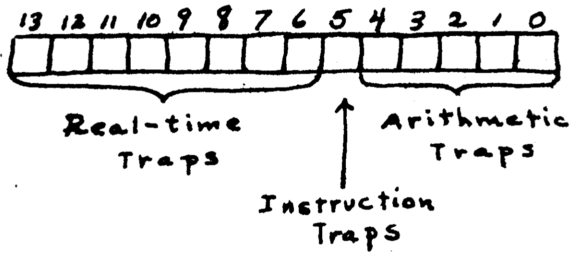
APPENDIX IV

CHARACTER INDICATORS

d_3^S	d_2^S	d_1^S	d_0^S	character of bit group
0	0	0	1	Leftmost bit is 0, and all other bits are 0.
0	0	1	0	Leftmost bit is 0, and at least one other bit is 1.
0	1	0	0	Leftmost bit is 1, and all other bits are 0.
1	0	0	0	Leftmost bit is 1, and at least one other bit is 1.

APPENDIX V

TRAP INDICATORS



Assignment of arithmetic traps:

D_0^T	-	Coefficient Traps
D_1^T	-	Exponent Traps
D_2^T	-	Parameter Traps
D_3^T	-	Operation Traps
D_4^T	-	Segment Traps

APPENDIX VI

At present, indirect addressing ($\beta = 10000$) is not available on Class 6 (index) instructions*, or on instruction 8 of Class 5. Nor are the following instructions available:

DCT	ADJ	BAM	CPD
DXT	SCL	BAD	CSD
DPT	SGN	SEX	SES
DOT	SQT	AUX	SET
DST	RNV	CSX	JUN
ICS	BAA		

It is foreseen that SHN, SHC, and SHR will be replaced by six instructions, SRN, SRC, SRR, SLN, SLC, and SLR (numbered 2 through 7), named "Shift Right with Normal Connection," ..., "Shift Left with Normal Connection," ... . The first three are merely mnemonic and name changes for SHN, SHC, and SHR. The last three function in the same manner as the first three with the shift parameter convention reversed, i.e. a positive parameter will cause a left shift and a negative parameter will cause a right shift.

---

* Indirect addressing <u>is</u> available on three Class 6 instructions: instructions 68, 69, and 6a. (footnote ~~added~~ added in proof)