Standard

INNER COMPUTER–MODEL 9

PRINCIPLES OF OPERATION

C. M. (CLANCY) DAVIS
STANDARD COMPUTER CORP.
777 NO. 1ST ST. SUITE 600
SAN JOSE, CA. 95112
(408) 294-7150

Standard
Standard Computer Corporation

Inner Computer - Model 9
Principles of Operations

This manual provides information on the content and operation of the STANDARD Computer Corporation Inner Computer - Model 9 (IC-M9) Data Processing System. It contains detailed descriptions of computer instructions and input/output channel control. IC-M9 MINIFLOW programming techniques and emulation examples are also included. In addition, IC-M9 system architecture and philosophy are provided.

The first section is a general description of the emulation technique used by STANDARD Computer Corporation. The second section is a working description of the system. It includes illustrations to aid the reader in learning the emulation technique and successfully write basic MINIFLOW programs. The third section is a comprehensive description of the system components from the programmer's standpoint. It describes the complete set of MINIFLOW instructions and control conditions and assumes general familiarity with the IC-M9.

This manual, together with the IC-6000 System Operating Guide for 7090/7094 Emulation, Form No. S-6001, and the IC-6000 System Operating Guide for 7040/7044 Emulation, No. S-6002 are to be used as reference material for the STANDARD Computer Corporation IC-M9 System.

CONTENTS

CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

DEFINITION OF SPECIAL TERMS AND CONCEPTS

This manual was written to be used by programmers, logicians, field and test engineers, system and design engineers, marketing and management personnel. The following is a brief definition of common words used in this manual which may be ambiguous because of the mixture of technical disciplines.

On: True, one, minus.

Off: False, zero, plus.

Set: To turn on (or off if specified).

Reset: To turn off (or set to an initial state if specified).

Channel B Control: Automatic control for channel oriented operations to allow the same MINIFLOW to service both Channels A and B keeping their status information and control separate.

Control and Transfer Vector (C&TV): A half word in the entry table used to specify the wired-in-sequence control and MINIFLOW starting address for a particular group of object instructions decoded by the translators.

Control Memory: The smaller, faster memory used to store the MINIFLOW emulation routines and to provide buffering for I/O operations.

Entry Table: The block of 64 half words (C&TV's) in control memory at mini-location 200. This table defines the MINIFLOW entry points for each group of object instuctions, and the type of wired-in-sequence used for the group.

Exit: A provision to leave a MINIFLOW program sequence to either return to scheduler control or to return from a subroutine to the calling routine.

Group: All the object instructions which, when translated, cause the wired-in-sequence to select the same control and transfer vector. There is a different group for each C&TV and they are numbered from 0 to 77 (octal).

Hard Register: A set of flip-flop registers available for accumulating, counting, addressing and indexing. They are generally made available to the object programmer. They are faster to work with than memory locations and may directly participate in arithmetic operations.

IC: The hard register used by the IC-M9 to address the object instruction. It is directly affected by wired-in-sequence and serves as the object machine Instruction Counter.

Inner Computer The part of the IC-M9 used to simulate the control logic of the object machine.

Special Instruction: A specially designed MINIFLOW emulation routine used to perform a function normally performed by several object instructions.

Main Engine: The part of the IC-M9 used to perform the primary arithmetic, logical and shifting operations.

Main Memory: The larger memory (32,768 thirty-six bit words) used to hold the object program.

Mini Engine: The part of the IC-M9 used for program sequence control, and shift counting.

MINIFLOW: The IC-M9 program language used to emulate object machine instructions.

Mini-Instruction: An 18 bit instruction executed by the IC-M9 and stored in a half word of control memory. It has a 6 bit primary operation code (POP code) and other fields to further define the operation.

Mini-Location: The octal half word address assigned to a mini-instruction. RB holds the mini-location of the next mini-instruction.

Mini-Pair: A set of two mini-instructions, the left and right half of a 36 bit word in either control core or the MOP register.

Multitag Mode Control: Automatic control to switch between selecting one out of seven index registers or selecting from one up to three index registers out of three index registers.

Object Machine: The machine to be emulated by the IC-M9. There will be a distinct MINIFLOW program for each object machine.

Object Instruction: An instruction to be emulated. It is held in main core and pointed to by the IC.

POP Code: Primary operation code with the field specifying a particular machine operation. It may be modified by the SOP and TOP codes. It is always defined by bits 0 to 5, the first six bits of the instruction.

Precondition Controls: A set of special control flip-flops initialized by the translators.

Program Levels: The four different levels of MINIFLOW program execution. They are activated by the scheduler to service the I/O, console and object program requirements.

Program Sequence Counter (PSC): The RB register in the mini-engine used by MINIFLOW as an instruction counter.

Scheduler: A scanning device used to activate program levels on a priority basis to allow servicing of the I/O, console, and object program requirements.

SOP Code: The Secondary operation code is the second parameter modifying a machine operation.

Subroutine Mode Control: The control mode entered by a subroutine transfer. An Exit turns off the mode and returns control to the main program. This allows quick resumption of a main program after a subroutine is completed.

Target Machine: The machine to be emulated by the IC-M9. It may be a previously defined computer or one defined by the emulation MINIFLOW. This term is synomonous with object machine.

TOP Code: The Tertiary operation code is the third parameter modifying a machine operation.

Transfer Trap Mode Control: Automatic control in the wired-in-sequence to prevent a transfer from being made and to enter MINIFLOW with the TSAT flip-flop indicating if a transfer would have been taken.

Translators: The logic used to decode the object instruction's "op-code", select an emulation group and set the pre-condition controls.

Wired-In-Sequence, WIS: A wired sequence of steps used to fetch object instructions, perform effective address computation, control the IC and fetch the memory operand. This sequence is program controlled.

Zonable: Capable of having the field limits specified by a zone code as opposed to having them implicitly defined in the primary operation (POP code).

Zone Code: A TOP code used expressly for defining the field limits of a machine operation.

Zoned: Loaded or moved with field limits specified by a zone code.

SECTION I

IC-M9 SYSTEM ARCHITECTURE AND PHILOSOPHY

THE INNER COMPUTER CONCEPT

The most important quality of a Computer's language is its ability to control communications between the various functional stations (memory, registers, arithmetic units and input/output devices). In most present day computer systems the relationships among the functional stations are frozen by the design and wiring of the system. Such fixed relationships mean that a particular computer can execute only one machine language effectively and cannot utilize the full potential of each functional station.

A NEW CONCEPT IN COMPUTER DESIGN

The IC-M9 is a computer within a computer. This structure eliminates the fixed link between the various functional stations heretofore experienced in conventional computers. In the IC-M9 all functional stations communicate with each other through the inner computer. This inner computer is programmed to control communication between the various functional stations. By programming the inner computer the IC-M9 is set up to emulate both new and previous computers. This multilingual capability, implemented by a unique process called the MINIFLOW emulation system, allows the IC-M9 to use existing program libraries without reprogramming.

The IC-M9 should be viewed as divided into two parts - the external functional stations, and the inner computer. The external functional stations consist of the main memory, arithmetic units and registers, input/output channels, input/output devices and the console. They perform the function of similar devices on the computers being emulated. The inner computer consists of a scheduler, a wired-in-sequence unit, a control memory, translators, mini-instruction registers and decoders, indicators and display registers, and a mini engine. The inner computer takes the place of much of the wiring and control logic in a conventional computer.

Programs written in the language of the object machine (i.e., the machine being emulated) are stored in the main memory. When the program is run, the inner computer fetches an instruction from the main memory, and performs the necessary indexing and indirect addressing operations by means of a MINIFLOW controlled wired-in-sequence (hardware).

The object code of the instruction is translated and a MINIFLOW emulation routine in the control memory is entered which directs the inner computer through all the steps necessary to execute hs particular instruction. The next instruction is then fetched from main memory and the entire process is repeated until the program is terminated. Thus, the inner computer acts as an interpreter directing the IC-M9 system to respond with the same results as the computer it is intended to model.

SYSTEM COMPONENTS AND ARCHITECTURE

In this subsection, one configuration of the IC-M9 system is described to acquaint the reader with the various functional stations. Descriptions of each functional unit follow in the next two subsections. The MINIFLOW emulation process is described, showing how the functional stations work together to execute a program written for some other computer.

A configuration of the IC-M9 as it is used to emulate a well known second generation data processing machine consists of:

1. The inner computer, with associated registers and control memory.

2. External functional stations, such as, main memory, high speed registers, an arithmetic and logical unit, operator's console and two input/output channels.

One channel controls a card reader, console typewriter and up to 12 magnetic tape units. The other channel handles up to 12 tape units.

Figure 1 shows an IC-M9 configuration as outlined above and its external functional stations.

EXTERNAL FUNCTIONAL STATIONS (Refer to Figure 1)

THE MAIN MEMORY functions as the core storage of the "target machine". It stores data and object instructions in the form of a program in the machine language of the "target machine". The main memory contains 32,768 words, each consisting of 36 bits plus one parity bit. The full cycle time is 2 micro-seconds.

THE REGISTERS are high speed storage elements available to the operator and programmer on the same basis as those in the target machine. The inner computer assigns certain functions to the registers as required to duplicate those available on the target machine. Typical assignments are: Accumulator (AC) 38 bits, Multiplier-Quotient (MQ) 36 bits, Sense-Indicators (SI) 36 bits, Index Registers (XR1 through XR7) 15 bits each, Instruction Counter (IC) 15 bits.

THE ARITHMETIC UNIT or main engine performs arithmetic and logical functions, such as fixed and floating point addition, multiplication, logical AND, OR and masking operations. It is also used by the inner computer for internal operations.

THE OPERATOR'S CONSOLE AND DISPLAY UNIT simulates all the console functions of the target machine. The console contains the keys, switches, and lamps necessary for manual and semi-automatic control and the visual checking of information in the system. Power to the system may be controlled from the console. All memory and register locations can be displayed. An execute entry function permits execution of console-keyed instructions. Address stop control provides several optional stop modes.

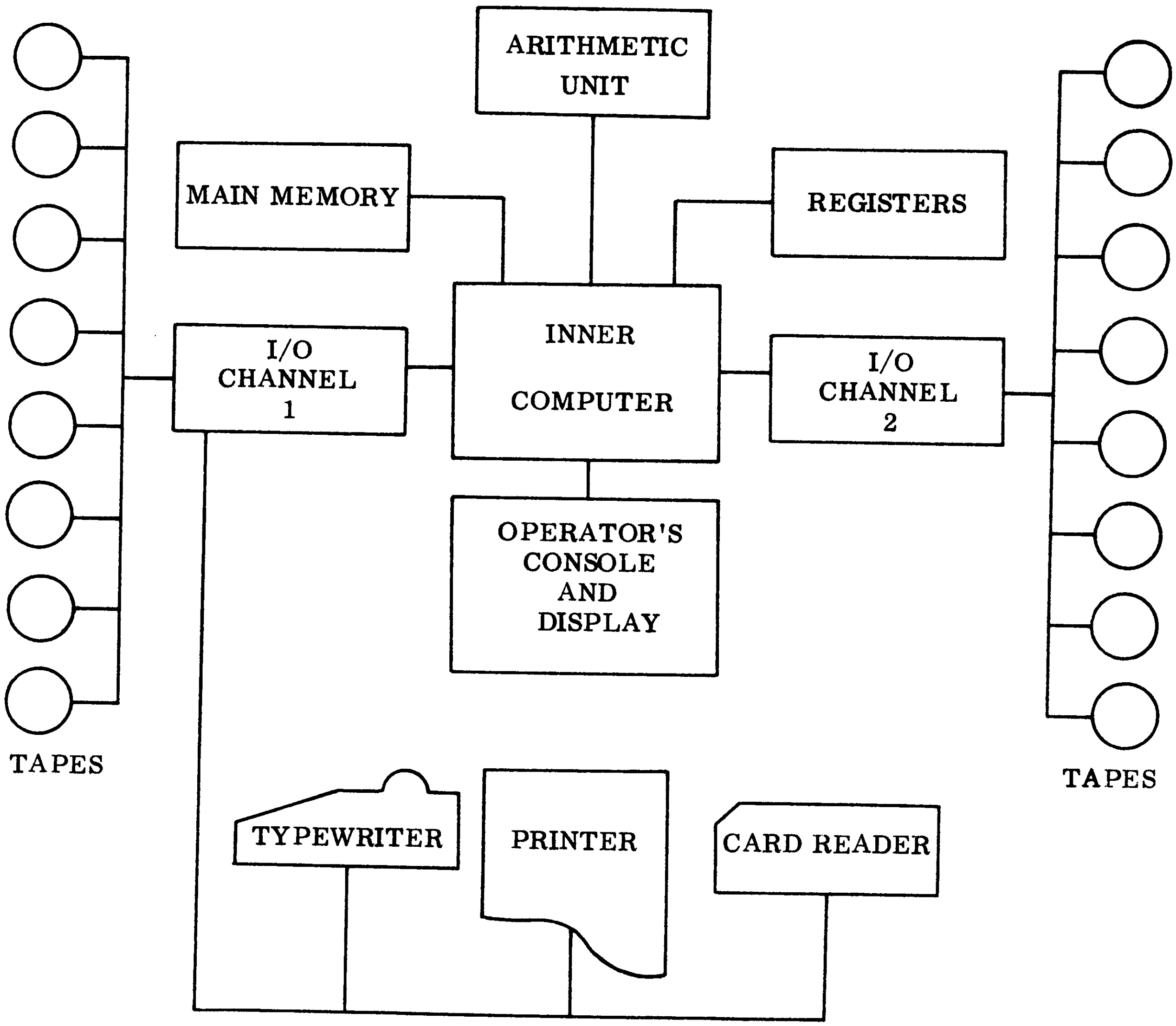


Figure 1. Typical IC-M9 System Configuration

THE INPUT/OUTPUT CHANNELS control the quantity and destination of all data transmitted between the inner computer and the peripheral units. The channels are small, specialized data processors which perform their functions independently of the inner computer and independently of each other.

THE PERIPHERAL UNITS consist of magnetic tape units, card readers, a printer and a typewriter. These are fully compatible with comparable devices on the target machine. The formats, where they differ from those of the target machine, may be converted within the inner computer to a compatible format by means of MINIFLOW.

INNER COMPUTER (Refer to Figure 2)

THE CONTROL MEMORY contains 1,024 words, each consisting of 36 data bits and one parity bit. The control memory is used to store MINIFLOW routines, data and constants used by the MINIFLOW system or the hardware, and as a buffer area for data transmitted between main memory and the input/output channels. Control memory and the main memory are independent and fully overlapped. The control memory full cycle time is 1 micro-second.

THE SCHEDULER responds to the service requests of the I/O devices, the operator's console, and the object program. It does so on a fixed priority basis to achieve the least interference between the simultaneous operations of these devices. The scheduler passes control to certain entry points in the wired-in-sequence depending on the type of the request honored.

THE WIRED-IN-SEQUENCE contains certain hard wired subroutines or complex sequences. For example, one of the sequences is used to fetch the instruction to be emulated, decode it, perform indexing and indirect address operations, update the instruction counter, fetch the operands required by the instructions, and fetch the first mini-pair instruction. Another sequence saves certain registers by storing them in predetermined control memory locations, then restores them at a later point of intervention. The scheduler will pass control to one of the sequences depending upon the type of request which is being honored. From there, the sequence is stepped from one state to another, although some steps may be skipped within a sequence.

THE TRANSLATORS decode the object instruction and point to a half word in the entry table called the CONTROL AND TRANSFER VECTOR which controls the wired-in sequence and gives the starting address of the routine necessary to complete the emulation. The translators also pre-condition certain control flip-flops in order to pass on specific information about instruction characteristics to the MINIFLOW routine.

THE MINI-OPERATION REGISTER receives the 18 bit mini-instructions two at a time as they are read from control memory. The bit configuration is sent to the instruction decoder which sends the appropriate control signals throughout the system to perform the operations.

THE MINI-ENGINE REGISTERS AND ADDERS perform the MINIFLOW program sequence control and shift counting. Certain instructions make the mini-engine registers available to the programmer.

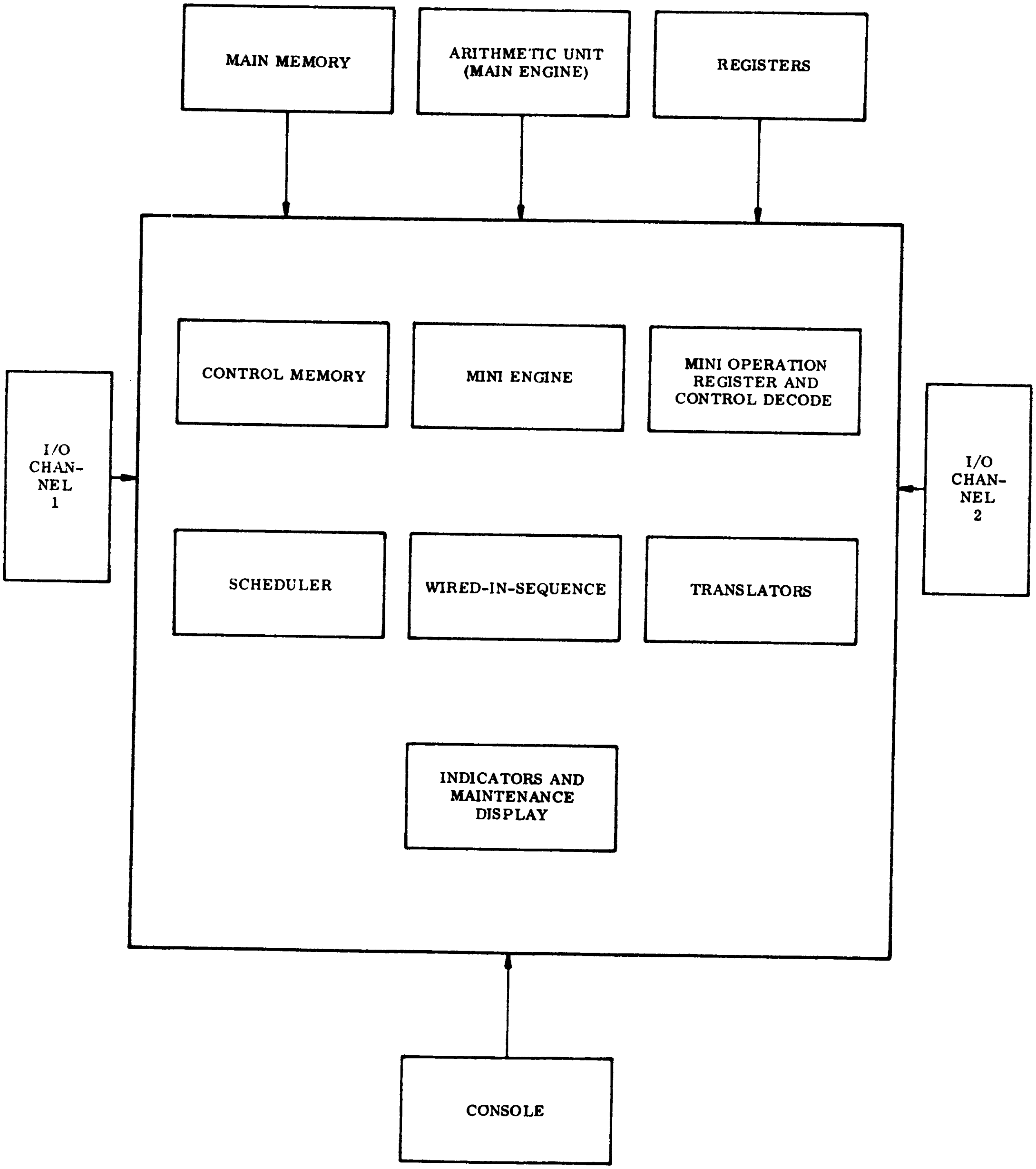


Figure 2. Components of the Inner Computer

THE INDICATORS AND DISPLAY REGISTERS are sets of flip-flops which hold hardware and MINIFLOW emulation program status. The status is determined by the occurrence of certain events within the system. The registers in this category include the display register, general indicators, and channel I/O indicators. Most of the bits in the indicators may be individually set and reset by mini-instructions. Many are connected to lamps on the operator's console; some are controlled by hardware; some directly control hardware functions.

THE MINIFLOW EMULATION PROCESS

For each instruction in the program of the target machine, the inner computer executes a particular routine made up of mini-instructions. For each input or output operation and each console function to be performed, the inner computer executes a similar set of routines. These routines are stored in the control memory of the inner computer. They are entered from the wired-in-sequence which is also controlled by a control and transfer vector in the MINIFLOW control memory. The entire collection of routines present in the Control Memory at any one time is called a MINIFLOW emulation system. It is the MINIFLOW emulation system which tells the inner computer how to interpret the machine language instructions of the emulated computer. It follows that in order to emulate a different computer, one has only to change the MINIFLOW system, i.e., a program resident in control memory.

THE GENERAL CONTROL SEQUENCE

Very briefly, when a "request" is set, the scheduler passes control the the wired-in-sequence which in turn passes control to MINIFLOW execution. When the MINIFLOW is completed, control is returned to the scheduler.

Specifically, when a program request is honored by the scheduler, control passes to the wired-in-sequence. At this time, an instruction from main memory is brought to the main engine. The operation code portion of the instruction is sent to the translators which generate an address pointing to a control and transfer vector (C&TV) in the control memory entry table. This C&TV provides a MINIFLOW starting address, and controls the following wired-in-sequence options:

1. Indexing and indirect addressing of the instruction address.

2. Fetching an operand from memory.

3. Treating the instruction as a transfer.

The starting address of the MINIFLOW routine is sent to the program sequence counter in the mini-engine. The address portion of the object instruction may be modified by the Index Registers and indirect addressing. An operand may be brought from main memory and the AC is put into the main engine. The MINIFLOW emulation program is then executed. When the MINIFLOW program is finished, it returns control to the scheduler. The process is repeated as the scheduler honors the program request again. The above explanation is described in more detail in sections II and III.

It should be noted that not all instructions require the same wired-in-sequence operations, and a few instructions require only the control and transfer vector for complete emulation. Moreoever, operations such as input-output and console functions which are not directly connected with the emulation of a specific instruction use a different set of MINIFLOW emulation routines, but the process is somewhat the same as that described above.

SECTION II

IC-M9 MINIFLOW PROGRAMMING EXAMPLES

GENERAL SYSTEM DESCRIPTION

The IC-M9 functions as a stored program computer when in the MINIFLOW execution mode. The data transfer paths are illustrated in Figure 3.

THE MAIN DATA BUS is used for general data transfer in a time shared manner. The memories, operator keys, I/O and other functions are accessed via the bus by special mini-instructions.

THE MAIN ENGINE performs the arithmetic and logical manipulations on the data. It contains two shift matrices capable of being coupled for long shifts. These are fed from the main bus or from the adder; they in turn feed three 36 bit holding registers. These registers also have sign and Q bit control logic associated with them. The adder is a full 36 bit binary ADDER/SUBTRACTOR with general logic capability. The adder is fed from the holding registers and from the register stack; it also feeds the main bus, the register stack and the holding registers (via the shift matrices).

THE MINI-ENGINE performs the MINI-INSTRUCTION fetching and decoding; it has six registers associated with it. The RB register (11 bits) acts as a MINIFLOW program sequence counter (PSC); instructions addressed by RB are fetched "two at a time" from the control memory and placed in the MOP-L/R register. From there they are passed one at a time to the RM and RBR drivers; the RM drivers feed the instruction decode logic and the RBR drivers gate immediate data back onto the bus. The RD register (11 bits) acts as a program link register to save the PSC for return from a subroutine. The RC register (8 bits) is used as a shift counter, or to hold a skip distance for decision type instructions.

NOTE

The programming examples shown in this manual are generated using the Inner Computer Assembly Language (ICAP) as described in Form 4010.

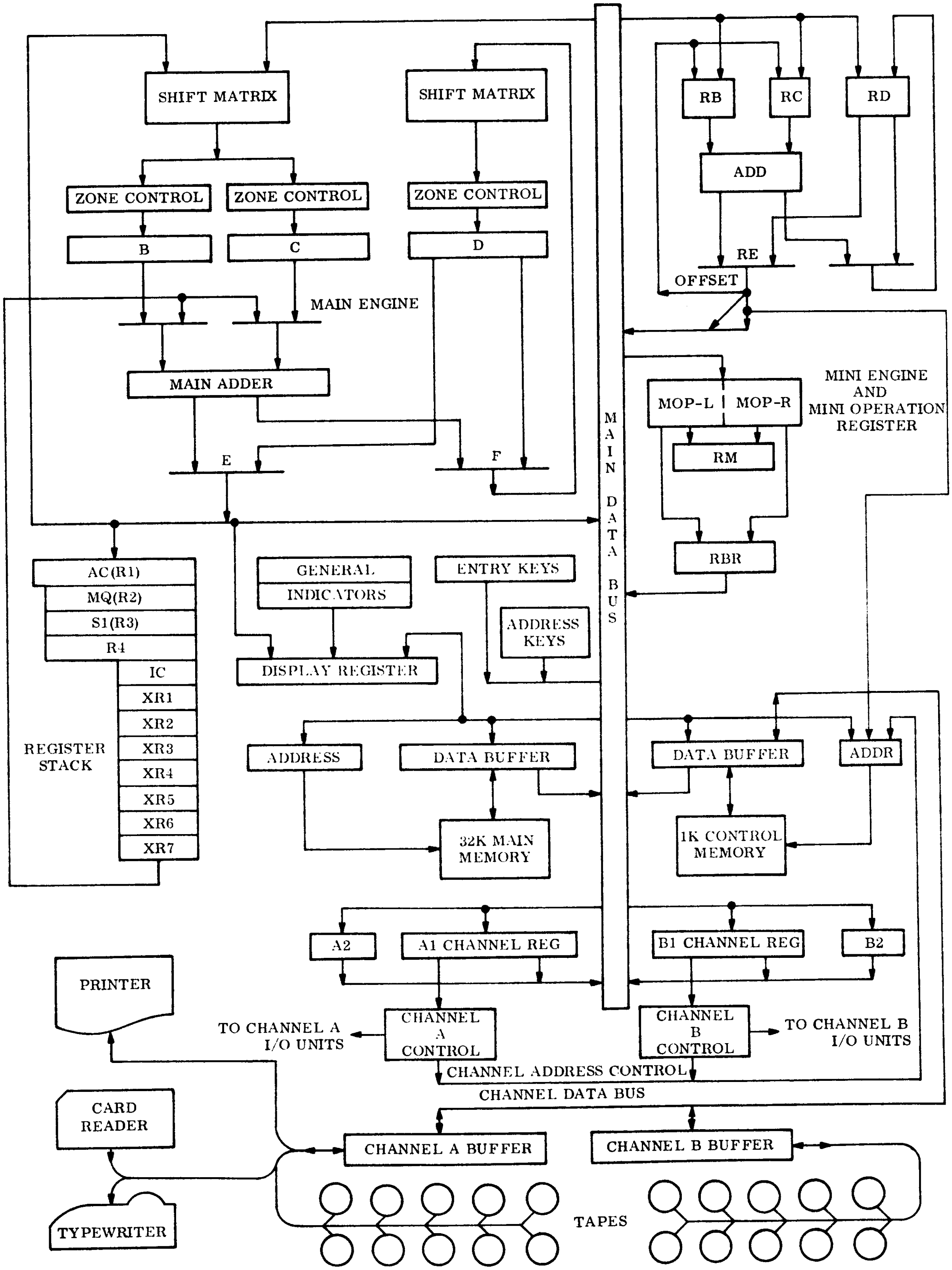


Figure 3. IC-M9 Data Flow Diagram

THE WIRED-IN-SEQUENCE

Normally in the first step of the wired-in-sequence, the 36 bit object instruction addressed by the IC is fetched from main memory and is loaded into both the C and D registers of the main engine. The first 12 bits of the D register are sent to the OP-code translators where special control flip-flops are set and an address is generated pointing to the entry table. In the next step this generated address is used to fetch a half word from the entry table in control core; this is the control and transfer vector which then controls how the wired-in-sequence proceeds. The format of the control and transfer vector is described in Figure 4.

INDEXING is accomplished by using the tag field (Bits 18-20 of the D register) to select which index register(s) to use; if these bits are all zero, zero is used as the index register value. Indexing is done by subtracting the Index register value from the C address field (bits 21-35) and putting the result in C or D or both. Binary arithmetic is used in this operation.

INDIRECT ADDRESSING is accomplished only if Bits 12 and 13 are both true in the D register. The address field (bits 21-35) of the D register (after indexing) is used to fetch a new word from main core. The right half of this word (bits 18-35) replace the previous right half of both the C and D registers. A second index cycle takes place if the tag field of the D register (after indirection) is non-zero with the result going to C and D (bits 21-35).

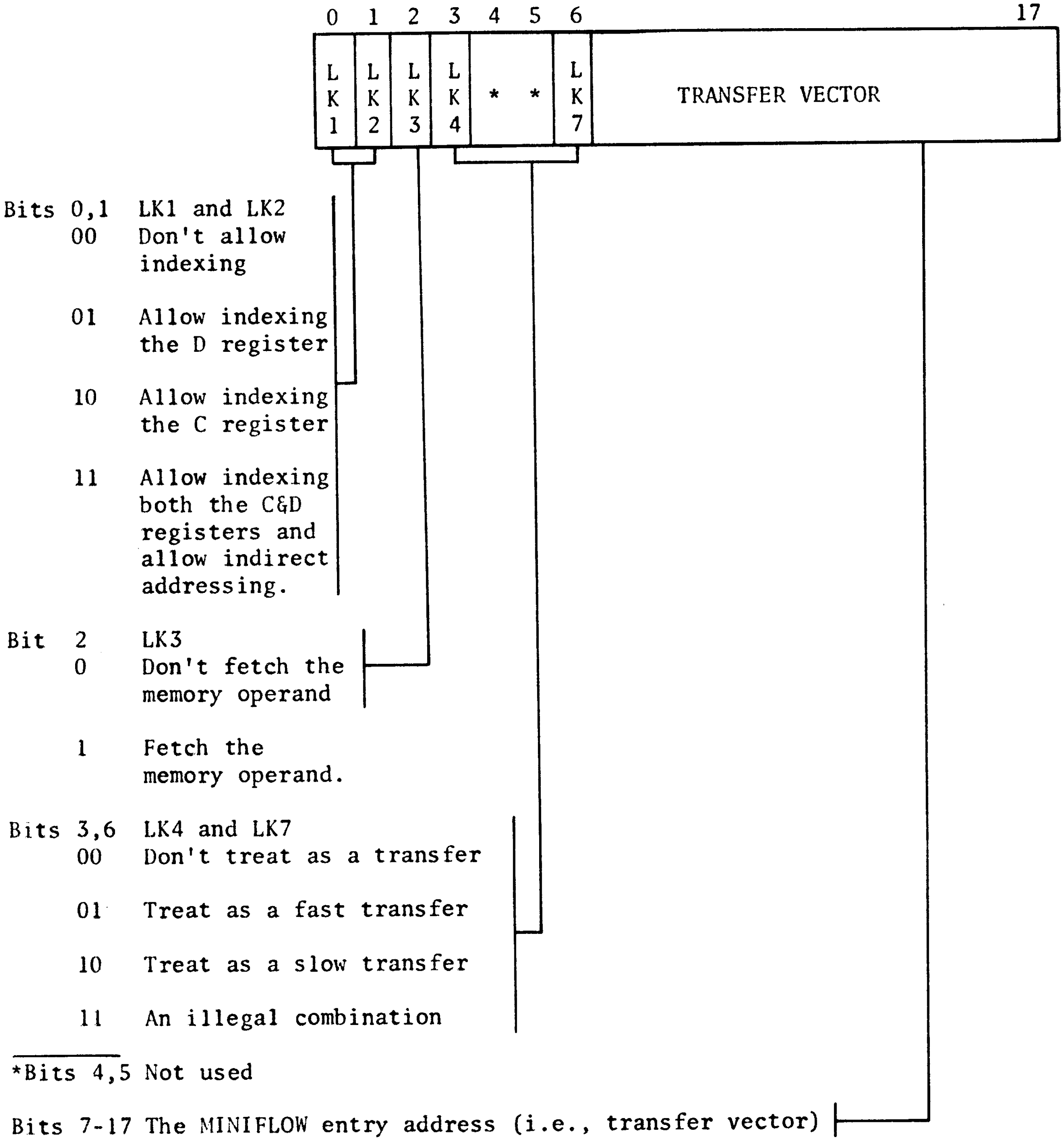
MEMORY OPERAND FETCH is accomplished by addressing main core from the address field of the D register and placing the 36 bit word into the C register.

AC OPERAND FETCH is always accomplished by bringing the contents of the AC (R1) into the B register of the main engine.

TRANSFER CONTROL allows bypassing MINIFLOW completely for a "fast" transfer; or allows MINIFLOW to set up the transfer and other conditions, and then re-enter wired-in-sequence to complete the "slow" transfer.

Details of the LK4 and LK7 relationship to the wired-in-sequence for automatic transfer controls are contained in Section III.

MINIFLOW TRANSFER ADDRESS is placed in the mini-engine sequence counter (RB) and points to the first MINIFLOW instruction of the emulation routine.



Bits 0,1 LK1 and LK2
00 Don't allow indexing

01 Allow indexing the D register

10 Allow indexing the C register

11 Allow indexing both the C&D registers and allow indirect addressing.

Bit 2 LK3
0 Don't fetch the memory operand

1 Fetch the memory operand.

Bits 3,6 LK4 and LK7
00 Don't treat as a transfer

01 Treat as a fast transfer

10 Treat as a slow transfer

11 An illegal combination

*Bits 4,5 Not used

Bits 7-17 The MINIFLOW entry address (i.e., transfer vector)

Figure 4. Control and Transfer Vector Half Word

SPECIAL MODE CONTROL FLIP FLOPS

MULTI-TAG MODE controls the manner in which indexing is done; it is set to the seven index mode (off) by the console reset key and is also turned on or off by a MINIFLOW instruction. Whenever an indexing operation takes place, either through wired-in-sequence or by a mini-instruction, the specific index(s) to be used in the operation are determined by the tag field (bits 18 to 20) of the main engine D register. When in multi-tag mode the logical OR of two or three registers may be used for the index value; or if changing index values, then two or three indexes may be altered simultaneously. The index used are specified by Table 1.

Table 1. Index Specification

D REGISTER TAG BITS	IN SEVEN INDEX MODE	IN MULTI-TAG INDEX MODE
000	NO INDEX	NO INDEX
001	INDEX 1	INDEX 1
010	INDEX 2	INDEX 2
011	INDEX 3	INDEX 1 "or" 2
100	INDEX 4	INDEX 4
101	INDEX 5	INDEX 1 "or" 4
110	INDEX 6	INDEX 2 "or" 4
111	INDEX 7	INDEX 1 "or" 2, "or" 4

EXIT AND SUBROUTINE MODES are used to quickly leave a MINIFLOW sequence and to easily link to subroutines. Most mini-instructions provide a means to EXIT at the end of the instruction. This usually may be accomplished by having bit 12 of the instruction on; there are also 3 test instructions which allow a conditional EXIT. When an EXIT is taken while not in the subroutine mode control is passed back to the scheduler and the wired-in-sequence.

The subroutine mode is entered by a special transfer mini-instruction. This turns the subroutine mode control on and sets up registers in the mini-engine to provide subroutine return linkage. Specifically, RB+1 goes to RD and RBR goes to RB to effect the transfer. If an EXIT is taken while in the subroutine mode, then the subroutine mode control is turned off and MINIFLOW transfers to continue where the subroutine transfer was taken, i.e., RD is placed in RB.

PRECONDITION CONTROLS are set by the translators during the wired-in-sequence. Their purpose is to allow the wired-in-sequence or MINIFLOW instructions to be modified in their actions so that the same routines may be used by similar object instructions. Two preconditions affect only wired-in-sequence during the operand fetch step. These are:

SUB (subtraction) causes the sign bit of the memory operand to be inverted when fetched.

MAG (magnitude) causes the sign bit of the C register to be set to zero (plus). If MAG and SUB are both set, then the sign of C is set to one (minus).

Two more preconditions may affect either wired-in-sequence or MINIFLOW:

ARI (arithmetic) causes bit zero of either memory or register stack operands to communicate with the main engine sign bits rather than the most significant bit of the main engine registers.

TSAT (test satisfied) is set by test conditions to cause a transfer via the wired-in-sequence or to cause a skip to be taken during MINIFLOW. Usually this flip-flop is set based on criteria other than just the combination of the op-code and the translator logic, i.e., generally an actual test is made of some machine condition.

The last five preconditions can only modify the MINIFLOW actions.

GEX (general exchange) allows the data sent on the main bus (or between the register stack and main engine) to be switched such that the most significant 18 bits and least significant 18 bits are interchanged. This swapping only occurs if GEX is on and the bit 6 flag is on in the instruction. But this swap may also be forced without GEX ON by the LOAD SOPS of the zoned Main Engine Operations explained in Section III.

GIN (general inhibit) prevents the data from the adder from going back into the main-engine registers or the index registers or IC register of the stack when GIN is on and the bit 6 flag is on in a main engine, XR or IC instruction.

GOP9 (general operation 9)

GOP10 (general operation 10)

GOP11 (general operation 11)

These flip-flops are set to differentiate object instructions entering MINIFLOW at the same location. The "DOL" main engine SOP and the "DOS" shift SOP are directly controlled by these GOPS.

TSAT, ARI, GEX, GIN and GOP 9, 10 and 11 may also be turned on or off by MINIFLOW as well as always being set by the translator. Section III contains a translator table, which contains the pre-condition flip-flop states when exiting from the wired-in-sequence.

SIGN AND Q BIT CONTROL

The main engine has a sign bit associated with each of its three registers (B, C&D) plus a single engine Q bit. The AC register also carries separate sign and Q bits, giving a total length of 38 bits for this register.

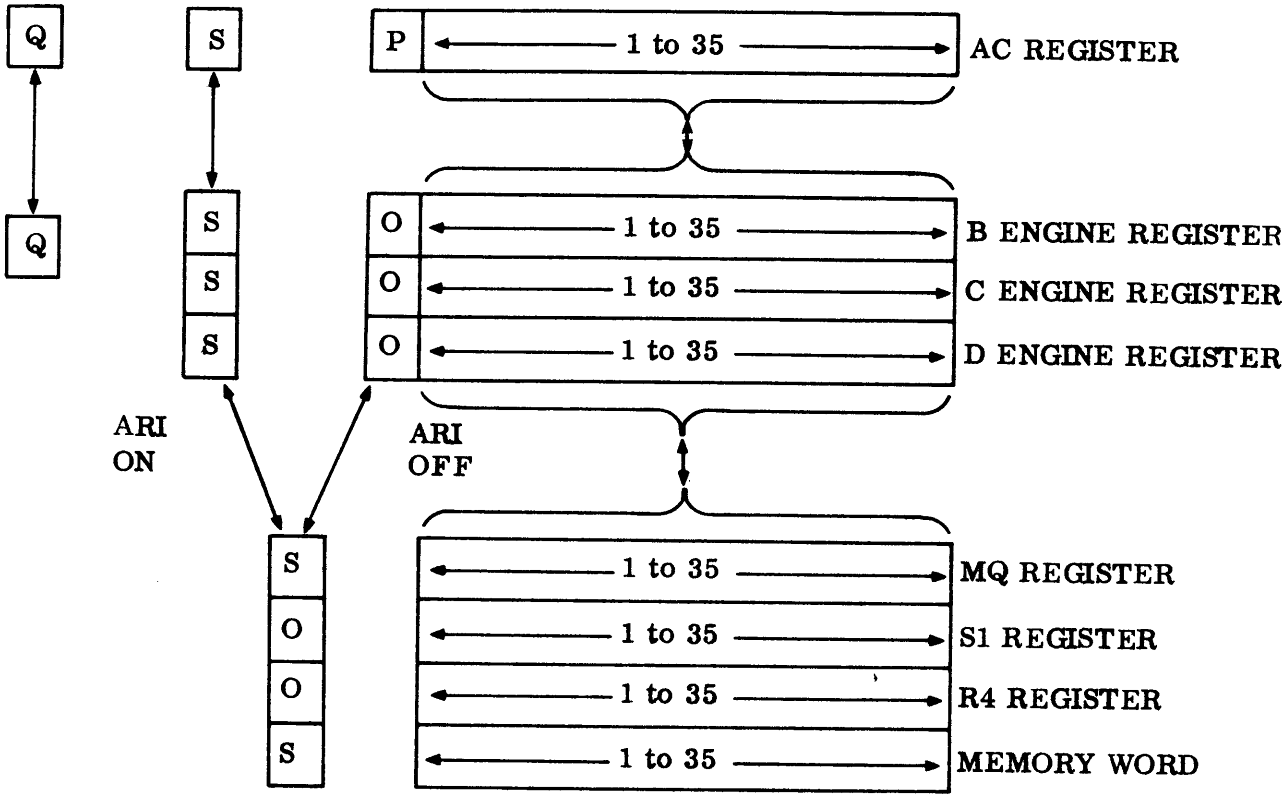


Figure 5. General Sign and Q Bit Control

When data is moved between the AC and one of the main engine registers the signs move with the other data bits. When data is moved between the MQ, SI, R4 or a memory word and one of the main engine registers, bit zero communicates with the engine register sign if ARI is on and communicates with the most significant bit of the engine register if ARI is off.

The engine signs are appropriately changed by signed arithmetic and shifting operations. The AC sign and Q bit may be turned on or off or toggled by the MISC POP with appropriate SOPS. The engine Q bit or AC Q bit may be changed under the following circumstances where X indicates a change which is described in Section III.

EQ	ACQ		
X		1.	Using a Q bit zone (TOP: 05 or 33) with the B or C register.
X		2.	An implicit shift (SOP:20)
X		3.	A multiply shift (SOP:11)
X		4.	AC POP with LDB or LDC TOP
X	X	5.	AC POP with load B or C and store
	X	6.	AC POP with s (store)
	1	7.	MISC SAQ
0	0	8.	MISC RAQ or RSM
X	X	9.	MISC TAQ

The particulars of these manipulations are in the description of the MINIFLOW instructions which may affect these flip-flops, and are specifically summarized in Section III. The state of the sign and Q flip-flops affects the operation of signed arithmetic and shifting operations as well as comparisons and sign tests.

CHANNEL B FLIP-FLOP CONTROL allows the programmer to take advantage of the symmetry found in channel control logic. This is done by using the same MINIFLOW programs for both Channels A and B and having their separate logic selected by the Channel B flip-flop. Specifically, MINIFLOW uses CHB flip-flop as follows:

1. The same mini-instructions will access even (for Channel A) or odd (for Channel B) words in control memory by addressing even words, since the Channel B flip-flop is ORed in with the low order address line in the CMI type of memory access.

2. The same test of an I/O channel indicator will test a group assigned to Channel A or a group assigned to Channel B according to how the Channel B flip-flop steers the test logic.

3. The channel commands will automatically be steered to the appropriate channel by the Channel B flip-flop.

4. Several other I/O oriented controls are automatically steered by this flip-flop.

OBJECT INSTRUCTION EXAMPLES

INSTRUCTIONS are 18 bits long. The first 6 bits (0-5) form the "primary operation" (POP) code; this POP code also determines the format of the rest of the instruction. Most instructions have four more fields, as shown below.

Bit 6	A FLAG bit controlling the GIN or GEX feature.
Bits 7-11	A "secondary operation" (SOP) code which further modifies the primary operation.
Bits 12	A FLAG bit (called EXIT) used to either return from a subroutine mode or to return back to the scheduler and wired-in-sequence.
Bits 13-17	A "tertiary operation" (TOP) code which may provide register zoning, a memory address, a skip distance, shift count, or special controls.

The specific function of bits 6 to 17 will be explained in the writeups for the individual "POP" groups described in the subsequent parts of this section.

There are instructions:

To perform shifting.
To access and operate on the main engine.
To access and operate on the Hard Registers.
To access and operate on the memories.
To access and operate on the mini-engine.
To access the I/O Channels and console.
To load immediate data into the main engine.
To perform tests and transfers.
To change control conditions.

In the symbolic code fields of the examples the POP will be listed first as an operation code. The SOP, TOP and FLAG fields will be listed in that order, as operands separated by commas; then remarks may follow. See the ICAP Assembly Language Programmer's Manual for symbolic coding standards.

The most generally used SOP codes are those controlling the main engine Adder logic and the SOPs which specify the loading of main engine registers. These two sets of SOP codes are defined in Tables 2 and 3.

Table 2. Load SOPs

SOP CODE	MNEMONIC	MAIN ENGINE REGISTER TO LOAD (where the POP specifies the source)
04	LDB	Load the B register
05	LDC	Load the C register
21	LDD	Load the D register

Table 3. Main Engine SOPs

SOP CODE	MNEMONIC	MAIN ENGINE ADDER OUTPUT
00	ZERO	All bits are zero
30	B	B register
31	C	C register
20	D	D register
34	B+1	B register plus one
35	C+1	C register plus one
14	B-1	B register minus one
15	C-1	C register minus one
36	B+C	B register plus C register
16	B-C	B register minus C register
17	C-B	C register minus B register
26	B+C+I	B register plus C register plus the previous carry
06	B-C-I	B register minus C register minus the previous carry
07	C-B-I	C register minus B register minus the previous carry
02	NB+1	Twos complement of B register
03	NC+1	Twos complement of C register
22	NB	Ones complement of B register
23	NC	Ones complement of C register
10	B.NC	B register AND ones complement of C register
11	NB.C	Ones complement of B register AND C register
24	B.C	B register AND C register
32	BUC	B register OR C register
12	BEC	EXCLUSIVE OR of B and C registers
01	DOL	DO the logical operation specified by the precondition* controls. See Table 4.

Table 4. The DOL Operations

GOPs			ADDER OUTPUT
9	10	11	
0	0	0	B
0	0	1	BEC
0	1	0	C
0	1	1	B.NC
1	0	0	B.C.
1	0	1	BUC
1	1	0	NB.C
1	1	1	B.NC

*Precondition controls are explained later in this section.

The most generally used TOP codes are those which control the main-engine field specification. There are 32 zone codes which select different contiguous fields of the main engine registers to participate in the operations. These zone fields are listed in Table 5.

Table 5. Zone TOPs

ZONE CODE	FROM TO BIT BIT	FIELD LENGTH	SPECIAL FIELD NAMES
00	00-35	36	Full 36 bit register
33	QQ-35	37	Full register and Q bit
05	QQ-08	10	Floating characteristic
20	09-35	27	Floating mantissa
35	00-08	9	1st quarter word
23	09-17	9	2nd quarter word
15	18-26	9	3rd quarter word
03	27-35	9	4th quarter word
37	00-17	18	Left half word
17	18-35	18	Right half word
27	03-17	15	Decrement field
07	21-35	15	Address field
13	24-35	12	Right third word
25	03-08	6	Translator Group Digits
34	00-05	6	Character 0
36	06-11	6	Character 1
21	12-17	6	Character 2
14	18-23	6	Character 3
16	24-29	6	Character 4
01	30-35	6	Character 5
30	00-02	3	Octal Character 0 (prefix field)
24	03-05	3	Octal Character 1
26	06-08	3	Octal Character 2
32	09-11	3	Octal Character 3
22	12-14	3	Octal Character 4
31	15-17	3	Octal Character 5
10	18-20	3	Octal Character 6 (tag field)
04	21-23	3	Octal Character 7
06	24-26	3	Octal Character 8
12	27-29	3	Octal Character 9
02	30-32	3	Octal Character 10
11	33-35	3	Octal Character 11

Zone control masks the operation such that only the specified bits of the zoned field will change in the receiving register, any initial carry will be introduced at the least significant bit of the zoned field, any carry out of the field will set the carry flip-flop.

TRANSFER EMULATION WITHOUT MINIFLOW

The TSAT flip-flop will be set by the translator during wired-in-sequence; the setting is off unless the object code translates to one of the transfer groups (as specified in section III) which may cause the TSAT flip-flop to be set on.

If the fast transfer control bit (LK7) is on in the group's control and transfer vector, the wired-in-sequence will test the TSAT flip-flop and directly emulate the transfer instruction without entering MINIFLOW.*

EXAMPLE 1. Fast Unconditional Transfer

Consider the Unconditional Transfer object instruction (TRA) with the following format:

0 11	12	14 17	18 20	21 35
+0020 INSTR. CODE	I	//////	TAG	BASIC TRANSFER ADDRESS

The instruction is to cause an unconditional program transfer to the memory location specified by the contents of the basic transfer address minus the contents of the specified index, if indexing is specified; OR if bits 12 and 13 are on (an indirect flag), then the above address is used to fetch a new tag and memory address field which together determine the final transfer address. New tag will be used for second level indexing.

During wired-in-sequence the OP code +0020 translates to group 2 and the TSAT flip-flop is set on, the control and transfer vector for Group 2 (at mini location 202) is fetched and analyzed. If LKs 1 and 2 are on, the indexing and indirect addressing called for by the object instruction will be performed in C and D. If LK7 is on the WIS will examine the TSAT flip-flop and cause the address field of D (21 to 35) to be placed in the IC so that the next object instruction fetch is from the transfer address.

The control and transfer vector needed is shown below and the address field points to the transfer trap routine if one exists.

```
        ORG       0/202           EXAMPLE  1    TRA
        VFD       07/141,11/TRAP TRA C+TV
```

EXAMPLE 2. Fast Conditional Transfers

Consider the conditional transfers defined below with the following format:

0 11	12	14 17	18 20	21 35
±01xx INSTR. CODE	I	//////	TAG	BASIC TRANSFER ADDRESS

Assume the instructions are to cause transfers under the conditions shown on the following page.

*MINIFLOW is entered however, if transfer trapping control mode is on; see Section III.

MNEMONIC	OP-CODE	TRANSFER CONDITIONS	TSAT FLIP-FLOP
TZE	+0100	AC equal to zero	on if AC is zero
TNZ	-0100	AC not equal to zero	on if AC is not zero
TPL	+0120	AC plus	on if AC is +
TMI	-0120	AC minus	on if AC is -
TOV	+0140	AC overlow on	on if OVFL on
TNO	-0140	AC overflow off	on if OVFL off
TQP	+0162	MQ plus	on if MQ is +

Indexing and indirect addressing are performed as specified by the instruction, and the overflow tests will reset the overflow indicator.

During wired-in-sequence all these OP CODES translate to group 10 and the TSAT flip-flop is turned on under the conditions specified in the table above. Indexing and indirect addressing as required is performed and then the TSAT flip-flop is examined to determine whether the IC is incremented by 1 (TSAT flip-flop off) or it is loaded with the contents of D (TSAT flip-flop on). The same control and transfer vector, at a different mini-location, would be used for these transfers.

```
ORG     0/210          EXAMPLE  2    TMI,TZE,ETC
VFD     07/141,11/TRAP COND. TRA C+TV
```

REGISTER POPs

The registers in the stack are accessed by a group of 6 mini-instructions. Zoning of the data is implicit in the particular register POP and they are described below.

POP CODE	POP MNEMONIC	AFFECTED REGISTER	IMPLICIT ENGINE ZONE CONTROL	TEST FLIP-FLOPS
33	AC	R1	QQ-35 37 bits and sign	Not affected
37	MQ	R2	OO-35 36 bits	Not affected
27	SI	R3	00-35 36 bits	Not affected
23	R4	R4	00-35 36 bits	Not affected
22	IC	IC	21-35 15 bits	Set appropriately
26	XR	XR*	21-35 15 bits	Set appropriately

*As specified by the tag field (Bits 18-20) of the D Register.

0 1 2 3 4 5	6	7 8 9 10 11	12	13	14	15	16 17
HARD REGISTER	GIN GEX	ENGINE OUTPUT	XT	RB	RC	SR	LDx

BITS

0-5 POP:

6 Flag: Inhibit storing back to IC or XR if GIN is on; Half exchange the register data if GEX is on.

7-11 SOP: Specifies a main engine operation from Figure 3.

12 Flag: EXIT

13 Replace "B" in SOP code with the POP register

14 Replace "C" in SOP code with the POP register

15 Store the SOP result back into the POP register

16-17 Store the SOP result back into a main engine register as specified below:

BITS		
16	17	
0	0	Do not store
0	1	Place in B Register
1	0	Place in C Register
1	1	Place in D Register

REGISTER MINI-INSTRUCTION FORMAT

EXAMPLE 3. Simple Register Load

Consider LDQ, an object instruction to load the MQ register with the following format:

0 11	12	14 17	18 20	21 35
+0560 INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

The instruction loads the MQ register with the word in memory specified by the address after any required address modification has taken place (indexing and indirect addressing).

This instruction translates to group 56 so the wired-in-sequence is controlled by the C&TV at mini-location 256. It should have LK's 1 and 2 on (allowing address modification) and also LK3 on which causes the memory operand to be fetched from main memory during WIS. The C&TV and the "program" required to complete the operation are below:

```
          ORG      0/256         EXAMPLE  3   LDQ
          VFD      3/7,15/LDQ LDQ C+TV
 * - - - - - - - - - - - - - - - *
          ORG      0/1001
 LDQ      MQ       C,S,EXIT      STORE (Y) IN MQ AND EXIT
 *
```

Example 4. Emulating Several Register Loads

Consider the object instructions used to load the AC: they are CAL, CLA, and CLS and they have the following format:

0 11	12 14	17	18 20	21 35
±050x INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRE-CONDITIONS SET ON
CAL	-0500	Clear and add logical	none
CLA	+0500	Clear and add	ARI
CLS	+0502	Clear and subtract	ARI & SUB

The ac register also may be loaded in any of these three modes with just one mini-instruction; however, when the memory operand is fetched if ARI is on the most significant bit goes to the sign bit of the C register and bit 0 of the C register is set to zero. In addition, if SUB is on this most significant bit is also inverted as it is loaded. Thus, one entry control and a one instruction MINIFLOW program are used to emulate three different object instructions. The translators not only point to the same control and transfer vector (Group 50), but also set the preconditions differently so the memory operand is loaded in three different ways.

The C&TV and MINIFLOW programs are shown below:

```
          ORG      0/250         EXAMPLE  4   CAL,CLA,CLS
          VFD      3/7,15/LDA CAL,CLA,CLS C+TV
 * - - - - - - - - - - - - - - - *
          ORG      0/2001
 LDA      AC       C,S,EXIT      STORE (Y) IN AC AND EXIT
```

MEMORY ACCESS POPs

MEM - Memory Instruction 67

Either the Main Memory or the control memory may be accessed by the MEMORY instruction. The address is generated by the main engine as specified by the SOP code; if a load SOP is used the address is taken from the console address keys. The Memory instruction has a mnemonic of MEM and an octal code of 67.

0 1 2 3 4 5	6	7 11	12	13	14	15	16 17
67 MEM	GEX	ADDRESS SPEC	XT	MS	CS	RB	REG

Bits	
0-5	POP:
6	Flag: Half exchange data if GEX is on.
7-11	SOP: If a "load" SOP (Table 2) then the address is taken from the address keys; if a main engine SOP (Table 3) then the address is appropriately generated by the adder.
12	Flag: EXIT
13	MEMORY SELECTION: 0 CONTROL MEMORY 1 MAIN MEMORY
14	CYCLE SELECTION: 0 READ DATA IN 1 WRITE DATA OUT
15	Replace R4 for B in the SOP code
16-17	DATA REGISTER: 00 Write Zeros (NOP if read) 01 Use the B register 10 Use the C register 11 Use the D register

CMI-Control Memory Immediate 77

The first 32 words of control memory may also be directly accessed with the address specified by the TOP field. The Data and Memory operations are specified by the SOP field. The Control Memory Immediate instruction has a mnemonic of CMI and an octal code of 77.

0 1 2 3 4 5	6	7 11	12	13 17
77 CMI	GEX	DATA SPEC	XT	MEMORY ADDRESS

BITS

0-5 POP:

6 Flag: Half exchange data if GEX is on.

7-11 SOP: If a "load" SOP (Table 2), the memory <u>reads</u> data into the specified register; If a Main Engine SOP (Table 3), data is appropriately generated in the adder and written into the control memory.

12 Flag: EXIT

13-17 TOP: The explicit address of one of the first 32 words of control memory

NOTE: The Channel B flip-flop is ORed in with bit 17 of the instruction so that only odd addresses are accessed if CHB is on.

EXAMPLE 5. Double Register Load

Consider DLD, an object instruction to load both the AC and MQ. It has the following format:

0 11	12	14 17	18 20	21 35
+0443 INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

The instruction loads the AC with the signed word in memory specified by the address (after modification) and loads the MQ with the word in memory located at the next higher address.

This instruction translates to group 45 with ARI set on. Two approaches are illustrated as follows:

1. This approach fetches the Y operand during wired-in-sequence.

```
       ORG      0/245        EXAMPLE  5    DLD
       VFD      3/7,15/DLD DLD C+IV
* - - - - - - - - - - - - - - *
```

```
        ORG     0/776
DLD     AC      C,S             STORE (Y) IN AC
        R4      D,S             MOVF ADDRESS TO R4
        MEM     R4+1,SRC        READ (Y+1) TO C
        MQ      C,S,EXIT        STORF (Y+1) IN MQ + EXIT
```

Note that the last line is the same as the "Program" in example 3; thus the two programs could be combined. This type of combination is directly illustrated in the second example of the Double Load (combined with example 4).

2. The second approach doesn't fetch the Y operand to C during the wired-in-sequence. It is fetched during the execution of MINIFLOW. It also only takes four mini-steps to complete the operation.

```
        ORG     0/245           EXAMPLF  6    DLD
        VFD     3/6,15/DLD DLC C+IV
        ORG     0/250
        VFD     3/7,15/LDA CAL,CLA,CLS C+IV
* - - - - - - - - - - - - - - - *
        ORG     0/1776
DLD     MEM     C+1,SRD         READ (Y+1) TO D
        MQ      D,S             STORE (Y+1) IN MQ
        MEM     C,SRC           READ (Y) TO C
LDA     AC      C,S,EXIT        STORE (Y) IN AC AND EXIT
```

There is very little difference between these two approaches. They both take the same amount of core space and the execution times are close. The 2nd is 2% faster when wired-in-sequence time is counted in with the MINIFLOW.

MAIN ENGINE POPs

These six mini-instructions modify the Main Engine Registers. The POP specifies what register receives the result of the operation. The SOP specifies what registers are used and how they are combined in the operation.

The TOP specifies the zone control field to be used: The adder operation,, the zero and carry test flip-flops and inserting the data into the receiving register(s).

POP CODE	POP MNEMONIC	PRIMARY OPERATION
50	PB	Test and zone the SOP results to B
51	PC	Test and zone the SOP results to C
53	PD	Test and zone the SOP results to D
52	PE	Only test the zoned SOP results
40	PBD	Test and zone D to B and zone the SOP results to D
41	PCD	Test and zone D to C and zone the SOP results to D

All these instructions will set the engine test conditions in the following manner:

PB, PC, PD and PE will test the zoned SOP results; PBD and PCD will test the zoned D register.

0-5	6	7-11	12	13-17
MAIN ENGINE REG	GIN	ENG OUTPUT	XT	ZONE CONTROL

BITS	
05	POP:
6	FLAG: Inhibit storing result in register if GIN is on.
7-11	SOP:* Specifies a main engine operation from Table 3.
12	Flag: EXIT
13-17	TOP: Specifies a main engine zone field from Table 5.

EXAMPLE 7. Add and Carry Logical

Consider ACL, an object instruction to add the unsigned value of a memory location (bits 0-35) to the magnitude of AC (bits P to 35) with end around carry. It has the following format:

0-11	12	14-17	18-20	21-35
+0361 INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

This instruction translates to group 36 with no preconditions turned on. The following code may be used:

```
        ORG     0/236         EXAMPLE  7   ACL
        VFD     3/7,15/ACL   ACL C+TV
*  - - - - - - - - - - - - - - - *
        ORG     0/500
ACL     PE      B+C,00-35     GENERATE P BIT CARRY
        PB      B+C+1,00-35   LOGICAL ADD WITH CARRY
        AC      B,S,EXIT      STORE SUM IN AC + EXIT
```

*The PB, PC, and PD POPs may also use the LDB and LDC SOPs (Table 2). These take data from the register specified by the POP and half exchange it and load it into the register specified by the load SOP under zone control.

Note: It might appear that the last two instructions could be replaced by:

```
          ORG       U/501        EXAMPLE  7   CONTINUED
  *
          AC        B+C+I,S,E  LOGICAL SUM TO AC + EXIT
```

However, this would allow the Q bit or the overflow indicator to be turned on. In this example the overflow cannot be turned on and the Q bit is unaltered because the engine Q bit is set to the AC Q bit during wired-in-sequence when the AC is unconditionally loaded into B. The Engine Q bit cannot change because it is not zoned into the engine operations; therefore it is the same when it is stored back into the AC Q bit.

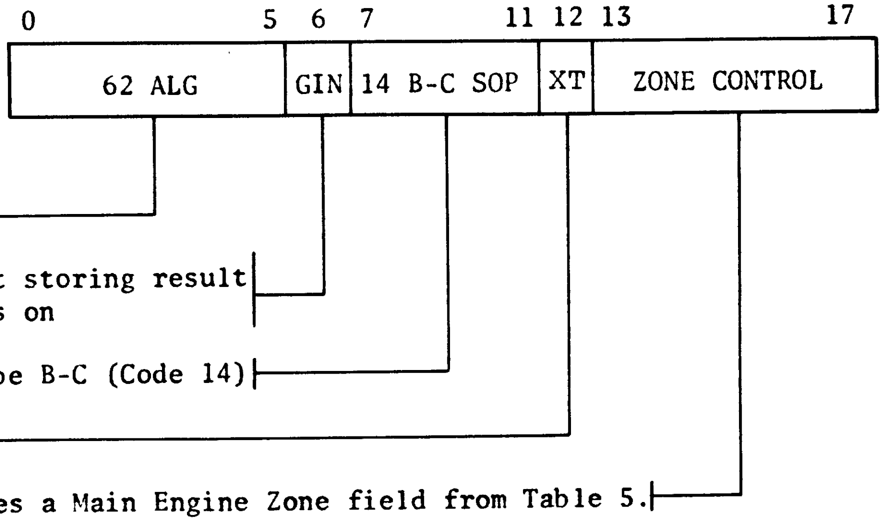
ALG-Algebraic Add 62

With the exception of the ALG POP the Main Engine SOPs ignore the signs assocciated with the B and C registers and treat all values as positive. An algebraic add may be performed taking the signs into account with the ALG POP and a B-C SOP (Octal code 16).

The ALG instruction is done in two cycles. First, ALG does a trial subtraction to determine which of the contents (B or C) is the greater. If the signs of B and C are the same the B-C SOP is interpreted as a B+C SOP for the second cycle where the result is zoned into B.

If the signs of B and C are different and C is less than B then the B-C SOP is performed again and the result zoned into B.

If the signs of B and C are different and C is greater than B then the B-C SOP is interpreted as a C-B SOP for the second cycle and the result in zoned into B. Also the sign of C is placed in B and a special indicator is turned on - the first carry flip-flop. This indicator is turned off if the signs are alike or if C is less than B. All other main engine tests are also set so that the arithmetic tests (TA, TAE, TAW) described later may test for zero, carry out, most significant and least significant bits, and like signs and first carry.



BITS	
0-5	POP:
6	FLAG: Inhibit storing result in B if GIN is on
7-11	SOP: Should be B-C (Code 14)
12	Flag: EXIT
13-17	TOP: Specifies a Main Engine Zone field from Table 5.

EXAMPLE 8. Emulating Several Arithmetic Instructions

Consider the fixed point arithmetic operations with the following format:

0 - 11	12 - 14	14 - 17	18 - 20	21 - 35
+040x INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITION SET ON
ADD	+0400	Add	ARI
ADM	+0401	Add Magnitude	MAG
SUB	+0402	Subtract	ARI and SUB
SBM	-0400	Subtract Magnitude	MAG and SUB

These instructions all translate to group 40. The memory operand is loaded into C if LK3 is on in the C&TV; in which case the sign of C will be appropriately set to allow emulation to be completed by the following MINIFLOW.

```
        ORG     0/240        EXAMPLE  8   ADD,SUB,ETC
        VFD     3/7,15/ARI   ADD,ADM,SUB,SBM C+TV
* - - - - - - - - - - - - - - - *
        ORG     0/1500
ARI     ALG     B-C,WQ-35    DO THE SIGNED ARITHMETIC
        AC      B,S,EXIT     STORE SUM IN AC AND EXIT
```

ARITHMETIC TEST POPs

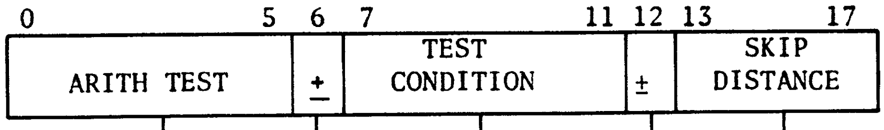
Arithmetic results and other machine conditions are tested by a group of three test instructions. They all work by testing the condition specified by the SOP field and setting the TSAT flip-flop if the test is met; the TOP field (bits 13-17) is always placed in the RC register of the mini-engine. The following points out the differences in the instructions:

POP CODE	POP MNEMONIC	PRIMARY OPERATION
61	TA	Test Arithmetic
65	TAE	Test Arithmetic or Exit
75	TAW	Test Arithmetic and enter Wired-in-sequence.

TA - Will skip forward or back if TSAT is set on and will take the next instruction if TSAT is off.

TAE- Will skip forward or back if TSAT is set on and will exit if TSAT is off.

TAW- Will always add RC to RB (or subtract RC from RB); then WIS is entered with TSAT setup and the operation is terminated like the end of a fast transfer with LK7 of the C&TV on.



BITS

0-5 POP:

6 TEST POLARITY:
0 - Test for False
1 - Test for True

7-11 SOP: Gives the condition to be tested for. See the TA SOP LIST which follows.

12 SKIP DIRECTION:
0 - Forward: Add RC to RB
1 - Backward: Subtract RC from RB

13-17 SKIP DISTANCE: The number of mini-instructions to move forward or back from this one.

NOTE: A skip of zero will execute the same test again.

Table 6. TA SOP Test Conditions

OCTAL CODE	SOP MNEMONIC	OCTAL CODE	SOP MNEMONIC	
00	NO	40	YES	UNCONDITIONAL TRUE
01	CF	41	CAR	CARRY/BORROW FROM ZONED FIELD
02	CX11F	42	CX11	CARRY OR ELSE GOP 11
03	G11F	43	G11	GOP 11
04	NZ	44	Z	ZERO IN ZONED FIELD
05	N910F	45	N910	GOP 10 and NOT GOP 9
06	G9F	46	G9	GOP 9
07	G10F	47	G10	GOP 10
10	RNZ	50	RZ	ZERO (IN BITS 3-10) OF RE REGISTER
11	FCF	51	FC	FIRST CARRY IN ALG POP
12	GINF	52	GIN	GIN
13	FOFF	53	FOF	FLTG OVERFLOW RMQ OR FAC (2 GEN. IND.)
15	LSF	55	LS	LIKE SIGNS IN B AND C
16	AQF	56	AQ	AC Q BIT
20	ZX11F	60	ZX11	ZERO OR ELSE GOP 11
21	MSBF	61	MSB	BIT 0 OF PE (MOST SIGNIFICANT BIT)
22	LSBF	62	LSB	BIT 35 OF PE (LEAST SIGNIFICANT BIT)
23	EQUF	63	EQU	PC EQUAL TO PB (Q & SIGN LOGIC)
24	LESSF	64	LESS	PC LESS THAN PB (Q & SIGN LOGIC)
		77	SAT	PREVIOUS TEST SATISFIED

EXAMPLE 9. INDEX CONTROL TRANSFERS

Consider the four index control transfers with the following format:

0 2	3 17	18 20	21 35
±2, 3	INDEX DECREMENT VALUE	INDEX	FINAL TRANSFER ADDRESS

MENMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
TIX	+2	Transfer on Index	GEX
TNX	-2	Transfer on no Index	GEX, GOP 11
TXH	+3	Transfer on Index high	GEX, GIN
TXL	-3	Transfer on Index low or equal	GEX, GIN, GOP 11

These instructions compare the value in the index registers specified by the INDEX field to the decrement value and do a conditional transfer based on the test result. The "2" types also subtract the decrement from the index if it is smaller.

These instructions all translate to group 1. The group 1 C&TV has LK's 1, 2, and 3 off since the address does not modify and a memory operand is not used; LK4 is set on to emulate a slow transfer. The code is below:

```
         ORG     0/201        EXAMPLE  9   TIX,TNX,ETC
         VFD     4/1,14/TX    TIX,TNX,TXH,TXL C+TV
*  - - - - - - - - - - - - - - *
         ORG     0/700
TX       XR      C-R,,GEX     TEST DECR. LESS THAN XR
         TA      CF,*+3       SKIP IF DECR. NOT LESS
         XR      R-C,S,G      SUBTRACT DECR, GEX + GIN
         TAW     G11F,TRAP    TRANSFER IF TXH OR TIX *
         TAW     G11,TRAP     TRANSFER IF TXL OR TNX *
*
TRAP     EQU     *+23
```

EXAMPLE 10. Index Linked Transfer

Consider a transfer instruction which preserves the location counter (in twos complement form) in a specified index register. It has this format:

0 11	12 17	18 20	21 35
+0074 INSTR. CODE	//////////	TAG	FINAL TRANSFER ADDRESS

MNEMONIC	OP-CODE	OPERATION
TSX	+0074	TRANSFER and SET INDEX

*The TAW Skips should point to the transfer trap routine if transfer trapping is being emulated; See Section III.

This instruction translates into group 7 with the following MINIFLOW coding:

```
           ORG       0/207          EXAMPLE 10    TSX
           VFD       4/1,14/TSX TSX C+TV
  * - - - - - - - - - - - - - - - *
           ORG       0/750
  TSX      IC        R,LDC          LOAD THE IC INTO C
           XR        NC+1,S         STORE 2'S COMP. TO INDEX
           TAW       YES,TRAP       UNCONDITIONAL TRANSFER
```

EXAMPLE 11. Storing Indexes

Consider the four instructions to store index registers into memory with the following format:

0 11	12 17	18 20	21 35
+063x INSTR. CODE	////////////////	TAG	FINAL OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
SXA	+0634	Store Index in Address	GIN
SXA	-0634	Store Index in Decrement	GEX,GIN
SCA	+0636	Store Complement in Address	none
SCD	-0636	Store Complement in Decrement	GEX

These instructions store the true value of 2's complement of the index specified by the INDEX field into the address field or the decrement field of the addressed memory word. The rest of the memory word is not changed.

These instructions all translate to group 63. The C&TV has LKs 1 and 2 off since there is no address modification. LK3 is also off since GEX control is not effective during operand fetch in WIS; and GEX control is useful while fetching the operand to control accessing the address or decrement field. The emulation code is below·

```
           ORG       0/263          EXAMPLE 11    SXA,SXD,ETC
           VFD       18/SX          SXA,SXD,SCA,SCD C+TV
  * - - - - - - - - - - - - - - - *
           ORG       0/2500
  SX       MEM       D,SRC,GEX      LOAD (Y), SWAP ON DECR.
           XR        R,LDC          ZONE IN THE INDEX VALUE
           PC        NC+1,21-35,GIN COMPLEMENT UNDER GIN
           MEM       C,SWC,GE       STORE UNDER GEX AND EXIT
```

*The TAW Skips should point to the transfer trap routine if transfer trapping is being emulated; see section III.

EXAMPLE 12. Sense Indicator Logical Instructions

Consider the instructions to test and manipulate the Sense Indicators with the following format:

0 11	12 17	18 35
±005x INSTR. CODE	///////////////	INDICATOR OPERATION MASK

MNEMONIC	OP-CODE	OPERATION	PRE-CONDITIONS SET ON
RZT	+0050	Right - Indicators all Zero Test	----
LZT	-0050	Left - Indicators all Zero Test	GEX
IIR	+0051	Invert Indicators - Right	GOP11
IIL	-0051	Invert Indicators - Left	GOP11,GEX
LIR	+0052	Load Immediate - Right	GOP10
LIL	-0052	Load Immediate - Left	GOP10,GEX
RFT	+0054	Right - Indi^ators Off Test	GOP9
LFT	-0054	Left - Indicators Off Test	GOP9, GEX
SIR	+0055	Set Indicators - Right	GOP9, 11
SIL	-0055	Set Indicators - Left	GOP9, 11, GEX
RNT	+0056	Right - Indicators on Test	GOP9, 10
LNT	-0056	Left - Indicators on Test	GOP9,10, GEX
RIR	+0057	Reset Indicators - Right	GOP9,10,11
RIL	-0057	Reset Indicators - Left	GOP9,10,11 GEX

These instructions will load immediate data i.e., the mask field into the left or right half of the indicators; or they will selectively turn on, turn off, toggle, test for zeros or test for ones either the left or right half of the sense indicators, using the right half of the object instruction as a mask.

They all translate into group 5. The symmetry is such that all 12 instructions are emulated by the 6 mini-instructions below:

```
          ORG       O/205          EXAMPLE 12    LFT,SIR,ETC
          VFD       16/ITC         SI TEST AND CONTROL C+TV
*  - - - - - - - - - - - - - - - - *
          ORG       O/1050
 ITC      SI        R,LDS,GEX      LOAD SI, SWAP IF LEFT
          PB        COL,18-35      DO ZONED LOGICAL OPER.
          TA        G11F,*+2       SKIP IF TEST INSTRUCTION
          SI        B,S,GEX        STORE UNDER GEX AND EXIT
          TA        N910,*-1       PUT IMMEDIATE DATA IN SI
          TAF       Z,*+1          TEST FALSE-EXIT
          IC        P+1,S,EXIT     TEST TRUE - BUMP IC,EXIT
```

POP MNEMONIC	POP CODE	OPERATION
LIB	14	Load Immediate to B
LIC	15	Load Immediate to C
LID	17	Load Immediate to D

0 5	6 11	12	13 17
LOAD IMMEDIATE	6 BIT DATA FIELD	CL	ZONE CONTROL

BITS

0-5 POP:

6-11 SOP: The 6 bit binary pattern broadcast onto the bus.

12 CLEAR BIT: If one, the unzoned part of the register will be reset.

13-17 TOP: A main engine zone code which gates the broadcast pattern into the register (Table 5)

Load Address

Registers B and C may be loaded with the entire mini-instruction in the right half (bits 18-35) and zeros in the left half (bits 0-17). These POPs are:

POP MNEMONIC	POP CODE	OPERATION
LAB	10	Load B with 00000010xxxx (octal)
LAC	11	Load C with 00000011xxxx (octal)

0 5	17
LOAD ADDRESS	DATA TO BE LOADED

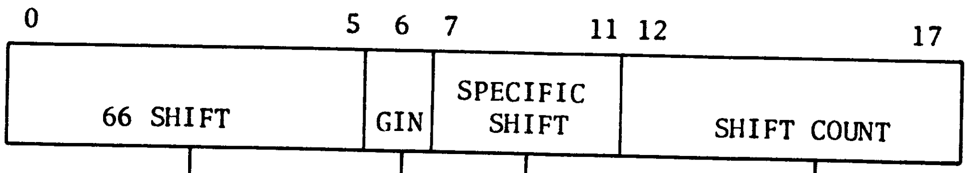
BITS

0-5 POP:

0-17 PATTERN: 18 bits to be placed into the low half of the register, high half is cleared.

SHIFT - SHIFT IN MAIN ENGINE 66

Skipping, Multiplication and Division are done with the SHIFT instruction. The SOP controls the type of shift and the participating registers. The TOP controls the extent of the shift except as noted on the following page.



BITS

0-5 POP:

6 FLAG: Inhibit register shifting if GIN is on.

7-11 SOP: Specifies the Shift operation.

12-17 TOP: A shift count specifying the number of bit positions to shift (0-76); however 77 specifies that the present contents of the RC is to be used for the shift count.

SPECIAL NOTE: With the BD-N SOP, the shift count is placed in RC and is incremented until a one bit shifts into bit position 9 of the B register to terminate the shift. With the DOS SOP, the shift count is taken from bits 28-35 of C in the main engine.

Table 7. The Shift SOPs

SOP CODE	MNEMONIC	NOTE	SHIFT DESCRIPTION
02	B-L		Open Shift B left
03	B-R		Open shift B right
22	C-L		Open shift C left
23	C-R		Open shift C right
04	D-L		Open shift D left
05	D-R		Open shift D right
12	D-ROT		Rotate D left, fill right end with left end bits
06	BD-L		Open shift B and D together left
07	BD-R		Open shift B and D together right
26	CD-L		Open shift C and D together left
27	CD-R		Open shift C and D together right
10	DIV	1	Divide B and D by C
11	MULT	2	Multiply C by D into B and D
16	BD-LF		Floating shift Bm and Dm together left
17	BD-RF		Floating shift Bm and Dm together right
14	BD-N	3	Normalize shift Bm and Dm together
32	BD-L9	4	Conditionally shift Bm and Dm together left
33	BD-R9	5	Floating shift Bm and Dm together right and insert one
30	FDIV	1	Floating divide Bm and Dm by Cm
31	FMUL	6	Floating multiply Cm by Dm into Bm and Dm
20	DOS		Do the shift specified by precondition controls. See Table 8.

Table 8. The DOS Operation

	GOPS			SEE	
9	10	11		NOTE	SHIFT PERFORMED
0	0	0			Perform no Shifting
0	0	1	D-ROT		Rotate D left (as SOP 12)
0	1	0	B-R		Open Shift B right
0	1	1	B-L		Open shift B left
1	0	0	BD-R	7	Open Shift B and D right
1	0	1	BD-L	8	Open Shift B and D left
1	1	0	BD-R		Open Shift B and D right
1	1	1	BD-L		Open Shift B and D left

The shift count is taken from register C in the main engine.

NOTES

1. Set sign of D to the algebraic sign of the quotient (EXCLUSIVE OR of B and C signs); also set PDCK general flip-flop* if C is less than B initially. The quotient is right justified in D with a length equal to the shift count.

2. Set signs of B and D to the algebraic sign of the product (EXCLUSIVE OR of C sign and initial D sign). The product is left justified in B and D with a length equal to 36 plus the shift count.

3. The normalize SOP loads the shift count to RC. Bm and Dm are shifted left until a one bit moves into bit position 9 of B; RC is incremented by one for each position shifted. If there is no one bit in B(9-35) or in D(9-35) the machine will hang.

4. Shift only if bit 9 of B is zero; use a shift count of one.

5. A one bit is inserted into bit 9 of B; use a shift count of one.

6. Set signs of B and D to the algebraic sign of the product (EXCLUSIVE OR of C sign and initial D sign). The product is left justified to bit 9 and is in B(9-35) and D(9-35); its length is equal to 27 plus the shift count.

7. The B sign replaces the D sign.

8. The D sign replaces the B sign.

*To MINIFLOW, the PDCK flip-flop set on signifies a divide check has occurred. Refer to the general flip-flop tests.

EXAMPLE 13. Emulation of Shifts

Consider these Shifts with the following format:

0 11	12 17	18 20	21 27	28 35
+07xx INSTR. CODE	////////////////	TAG	///////////	SHIFT COUNT

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
NOP	+0761	No operation	GIN
RQL	-0773	Rotate MQ left	GIN,GOP 11
ARS	+0771	Accumulator Right Shift	GIN,GOP 10
ALS	-0777	Accumulator Left Shift	GIN,GOP 10,11
LRS	+0765	Long Right Shift	ARI,GOP 9
LLS	+0763	Long Left Shift	ARI,GOP 9,11
LGR	-0765	Logical Right Shift	GIN,GOP 9, 10
LGL	-0763	Logical Left Shift	GIN,GOP 9,10,11

These shifts work in the following manner:

NOP - Do nothing

RQL - Shift the MQ left; bits shifted out of MQ(0) move into MQ(35)

ARS/ALS - Shift the AC(Q,P,1-35) to the right or left. Zero bits are shifted in; one bits shifted left into the P position turn on the overflow flag.

LRS/LSS - Shift the AC(Q,P,1-35) and the MQ(1-35) coupled together to the right or left. Zero bits are shifted in and one bits shifted left into the P position turn on the overflow flag. Put the AC sign in MQ(0) if LRS; put MQ(0) into the AC sign if LLS.

LGL/LGL - Shift the AC(Q,P,1-35) and the MQ(0-35) coupled together to the right or left. Zero bits are shifted in; one bits shifted left into the P position turn on the overflow flag. The signs are unchanged.

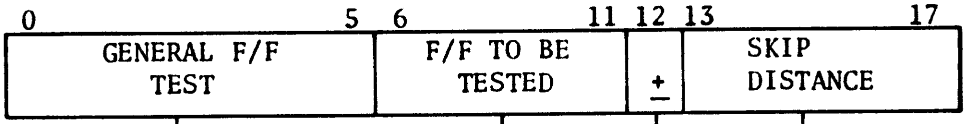
These instructions all translate into group 77. LK1 is set in the C&TV to allow only indexing the shift count in register C. The emulation code follows:

```
              ORG       0/277          EXAMPLE 13   ALL SHIFTS
              VFD       1/1,17/SHIFT SHIFT C+TV
   * - - - - - - - - - - - - - - - *
              ORG       0/2500
     SHIFT    MQ        R,LDD          LOAD THE MQ TO D
              SHIFT     D-L,1,GIN      CLOSE MQ SIGN GAP IF ARI
              SHIFT     DUS            PERFORM THE SHIFT INSTR.
              SHIFT     D-R,1,GIN      OPEN MQ SIGN GAP IF ARI
              MQ        D,S            RETURN THE MQ
              AC        B,S,EXIT       RETURN THE AC AND EXIT
```

TEST GENERAL FLIP-FLOP POPs

A set of general indicators may be tested and controlled by a group of five general flip-flop test instructions. They are similar to those described under Arithmetic Test POPs; that is, TSAT is set by the test condition and the TOP field is placed in RC. If TSAT is off RB+1 goes to RB, i.e., no skip; if TSAT is on then RB±RC goes to RB, i.e., skip.

POP CODE	POP MNEMONIC	PRIMARY OPERATION
60	TG	Test and skip if on, no skip if off
64	TGE	Test and skip if on, exit if off
71	TGF	Test and skip if off, no skip if on
70	TGS	Test and skip if on, no skip if off
74	TGR	Test and turn off and skip if on, no skip if off.



BITS	
0-5	POP:
6-11	SOP: Gives the general flip-flop to be tested. See the SOP list which follows.
12	SKIP DIRECTION: 0 - Forward: Add RC to RB 1 - Backward: Subtract RC from RB
13-17	TOP: The number of mini-instructions to move forward or back from this one.

There are 32 general purpose flip-flops which may be assigned any functions which have the following SOP codes: 04, 05, 07, 10, 11, 12, 13, 15, 17, 20, 21, 23, 25, 26, 27, 31, 33, 35, 37, 41, 43, 45, 47, 51, 53, 55, 57, 61, 63, 65, 71 and 75.

Seventeen general flip-flops which also drive console indicator lamps with specific labels are:

Table 9. TG SOP Test Flip-Flops

SOP CODE	MNEMONIC	NOTE	SPECIAL PURPOSE
52	SL1		Sense Light 1
54	SL2		Sense Light 2
56	SL3		Sense Light 3
50	SL4		Sense Light 4
30	DCK		Divide Check
36	IOC		Input/Output Check

Table 9. (Cont.)

SOP CODE	MNEMONIC	NOTE	SPECIAL PURPOSE
44	TCKA	1	Tape Check Error - Channel A
46	TCKB	1	Tape Check Error - Channel B
32	TCEA		Tape Check Enable - Channel A
22	TCEB		Tape Check Enable - Channel B
06	EOFA	2	End of File - Channel A
14	EOFB	2	End of File - Channel B
34	CTEA		Command Trap Enable - Channel A
24	CTEB		Command Trap Enable - Channel B
70	TCN		Trap Control
40	TRAP	3	Transfer Trapping Enabled
60	MTM	4	Multiple Tag Mode

Nine general flip-flops which are directly controlled by switches or may be set by means other than the TGS POP are:

SOP CODE	MNEMONIC	NOTE	SPECIAL PURPOSE
00	SSW	5	Sense Switch Tests
72	PDCK		Predivide Check - May be set by SHIFT Divide
16	DCTM		Divide Check Trap Mode - A console switch
74	CON	6	Console Request
42	DIS	7	Display
76	CIF		Current Instruction Display switch is OFF
62	FMQ	8	MQ factor exceeded
64	FPO	8	Floating Point Overflow
66	FAC	8	AC factor exceeded

Six SOP codes which when tested, will always be off are: 01, 02, 03, 67, 73 and 77.

NOTES

1. The Channel A Tape Check Error General Indicator (44) is turned off when the Channel A Tape Check Enable General Indicator (32) is off and a TRANSFER ON CHANNEL A REDUNDANCY CHECK (+0022) is processed as an object instruction with LK7 on (fast transfer control). Also TSAT is set on if Indicator 44 was on and 32 off.

   The same reset logic applies to Channel B; i.e., indicator 46 is turned off if indicator 22 is off when the object instruction -0022 is processed with LK7 on; also TSAT is set on if indicator 46 was on and 22 off.

2. The Channel A end of file General Indicator (06) is turned off when the Channel A Command Trap Enable General Indicator (34) is off and TRANSFER ON CHANNEL A END OF FILE (+0300) is processed as an object instruction with LK7 on. Also TSAT is set on if Indicator 06 was on and 34 off.

The same reset logic applies to Channel B, i.e., indicator 14 is turned off if indicator 24 is off when the object instruction -0030 is processed. Also, TSAT is set on if indicator 14 is on and 24 is off.

3. This flip-flop enables the Transfer Trapping Mode discussed in Section III under Trapping.

4. This flip-flop enables the Multiple Tag Mode explained under Special Mode Control Flip-Flops.

5. The specific Sense Switch tested, of the 6 on the console, is determined by bits 0-2 in the D register at the time of testing.

6. Any of the console request interrupt switches will turn on the CON flip-flop as the scheduler honors the request. These are: The console timer, the console reset, the clear core, the display switch, the enter keys switch, the load tape switch or the load card switch.

7. Display is set when any one of the following display conditions occur:

   a. A register is altered which has its corresponding console display switch down.

   b. An instruction is fetched for emulation and the Current Instruction Display Switch is down.

   c. A memory cell is altered which has that memory's display switch down and the memory address corresponds to the Address Keys on the console.

   In each of these cases when the display flip-flop is set it also loads the Display Register with the new data.

8. The MISC POP (as explained in Section III) may conditionally set these indicators on; however, they may only be turned off by the TGR POP.

   The conditions to turn on these General Indicators and the results of these conditions are shown in Table 10. NC indicates no change.

Table 10. General Indicator Conditions and Results

	CONDITIONS		GENERAL INDICATOR AND SPECIAL TEST RESULT		
MISC SOP	PE P BIT	ENG Q BIT	FMO	FPO	FAC
FOFA	0	X	NC	NC	NC
FOFA	1	0	NC	ON	ON
FOFA	1	1	NC	NC	ON
FOFQ	0	X	NC	NC	NC
FOFQ	1	0	ON	ON	NC
FOFQ	1	1	ON	NC	NC

The TA POP with FOF SOP will also test for logical OR of general indicators FMQ and FAC.

EXAMPLE 14. Fixed Divide

Consider DVP, a Divide object instruction which has the following format:

0 – 11	12 – 14	– 17	18 – 20	21 – 35
+0221 INSTR. CODE	I	//////	TAG	BASIC OPERAND ADDRESS

This instruction divides the AC and MQ registers by the signed memory operand. The signed 35 bit quotient is placed in the MQ and the Signed remainder is placed in the AC. If the magnitude of the AC is greater than the magnitude of the divisor, the division is not performed. Instead the divide check indicator is turned on and the computer proceeds to the next instruction.

```
        ORG     O/222        EXAMPLE 14    DVP
        VFD     3/7,15/DIV   DIVIDE C+TV
 * - - - - - - - - - - - - - - - *
        ORG     O/2222
 DIV    MQ      R,LDO        LOAD THE MQ TO O
        SHIFT   D-L,1        CLOSE MQ SIGN BIT GAP
        SHIFT   DIV,35       PERFORM THE DIVISION
        TGR     PDCK,DVCK    SKIP IF DIVIDE CHECK
        SHIFT   D-R,1        OPEN MQ SIGN BIT GAP
        MQ      D,S          STORE QUOTIENT IN MQ
        AC      B,S,EXIT     STORE REMAINDER AND EXIT
 * NOTE- THE CODE NEEDN'T BE CONTIGUOUS HERE
        ORG     *+6
 DVCK   TGS     DCK,*+1      SET DIVIDE CHECK FLAG ON
        EXIT                 EXIT FOR DIVIDE CHECK
 DCK    DGI     O/30
 PDCK   DGI     O/72
```

MAIN TO MINI POPs

The Mini-Engine Registers are aligned so that the least significant bits correspond to Bit 35 on the Bus. Virtual zeros exist in bit positions to the left of the Most Significant bit in a mini-engine register. Communication of Data between the Main Engine and the mini-engine is accomplished by 3 mini-instructions. Zone control is only effective with the Load SOPs when data is read into the Main Engine from the Mini-Engine. The Main Engine SOPs generate data and send it to the Mini-Engine with the following implicit zone control:

POP CODE	POP MNEMONIC	SOP TYPE	AFFECTED REGISTER	ZONE CONTROL
56	RB (note 1)	Load Engine	Specified by SOP NONE	Specified by TOP NONE
42	RC (note 2)	Load Engine	Specified by SOP RC	Specified by TOP 30-35
46	RD	Load Engine	Specified by SOP RD	Specified by TOP 25-35

0 – 5	6	7 – 11	12	13 – 17
MAIN-TO-MINI	GEX	ENGINE OUTPUT	XT	ZONE CONTROL

BITS	
0-5	POP:
6	FLAG: GEX allows communication with the left half of the Main Engine.
7-11	SOP: A load SOP (Table 2) will load the Main Engine from the Mini-Engine with zone control. A main engine SOP (Table 3) will load the adder output into RC (6 bits) or RD (11 bits). RB may not be loaded. See Note 1.
12	Flag: EXIT
13-17	TOP: Standard zone control (Table 5) applies for Load SOPS only; however, the exponent zone is treated specially for the RC POP. See Note 2.

NOTES

1. The RB POP only sends data to the Main Engine with a Load SOP. The Main Engine SOPS are not effective.

2. The RC POP has a special data path to move eight bits between RC and Bits 01 to 08 of the Main Engine. This path is enabled by the 00-08 zone control (TOP 35). The path is enabled for both load SOPS and Main Engine SOPS and is not affected by GEX control. In addition the RC POP sets the Zero, B and LSB test flip-flops.

EXAMPLE 15. Variable Length Divide

Consider VDP, a Variable Length Divide object instruction with the following format:

0 – 11	12 – 14	14 – 17	18 – 20	21 – 35
+0225 INSTR. CODE	I	COUNT	TAG	BASIC OPERAND ADDRESS

The VDP works the same as the DVP explained in the preceding example except that the length of the Quotient is determined by the count field (bits 11-17) rather than assumed to be 35. If the count is zero no operation takes place; otherwise, the C length quotient is right justified in the MQ and the least significant part of the remainder is in the left end of the MQ. The sign of the quotient is still in MQ(0).

This instruction also translates to group 22 with ARI, GEX, GOP 11 and GIN set on. In the following MINIFLOW both VDP and DVP are emulated, notice that the divisor is not loaded in WIS.

```
          ORG     0/222        EXAMPLE 15   DVP,VDP
          VFD     3/6,15/DIV DIVIDE C+IV
 * - - - - - - - - - - - - - - *
          ORG     0/2222
DIV       MQ      R,LDD        LOAD THE MQ TO D
          SHIFT   D-L,1        CLOSE MQ SIGN BIT GAP
          TA      GINF,FIXED SKIP IF FIXED LENGTH DIV
          PE      C,12-17      TEST DIVIDE COUNT FIELD
          TAE     NZ,*+2       EXIT ON ZERO COUNT FIELD
FIXED     LIC     43,12-17     LOAD A FIXED COUNT OF 35
          RC      C,12-17,SLX MOVE COUNT FIELD TO RC
          MFM     C,SRC        LOAD THE DIVISOR
          SHIFT   DIV,RC       DO THE DIVISION
          TGR     PDCK,DVCK    SKIP IF DIVIDE CHECK
          SHIFT   D-R,1        OPEN MQ SIGN BIT GAP
          MQ      D,S          STORE QUOTIENT IN MQ
          AC      B,S,EXIT     STORE REMAINDER AND EXIT
DVCK      TGS     DCK,*+1      SET DIVIDE CHECK FLAG ON
          EXIT                 EXIT FOR DIVIDE CHECK
```

The programmer should be aware of certain principles illustrated in this particular example. One is that the test for zero count could be set up by the RC without the use of the PE. But inserting a skip test after the RC would destroy the RC contents so it would have to be reloaded anyway. Another point is that the divisor was not loaded to C in wired-in-sequence because the count field needed to be loaded into RC from either C or D. If loaded from D it must precede loading the MQ into D, but the MQ must be preshifted one bit and the preshift would destroy the count being held in RC.

An alternative solution might involve loading the divisor in wired-in-sequence, separating the fixed and variable length division, saving the variable length count in RD while preshifting the MQ then moving it into RC with a MINI POP as described under Mini-Engine Operations.

EXAMPLE 16. Comparisons

Consider four comparison instructions with the following formats:

0 11	12 14	16	18 20	21 35
$\underline{+}$0340 INSTR. CODE	I	/////////	TAG	BASIC OPERAND ADDRESS

0 11	12 14	17	18 20	21 35
-01x4 INSTR. CODE	I	COUNT	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
CAS	+0340	Compare AC with Storage	ARI
LAS	-0340	Logical Compare AC with Storage	
VAS	-0154	Variagle Length Logical Compare AC with Storage	GEX
IAS	-0114	Variable Length Intersect AC with Storage	GEX,GIN

These instructions compare the contents of the AC to the contents of the memory location. CAS makes the comparison taking the signs into consideration (both AC sign and YO). LAS compares the magnitude of the AC to the unsigned memory operand (bits 0-35).

VAS logically compares the most significant portions of the comparands. The bit length of the comparison field is determined by the count field of the instruction.

IAS tests for any corresponding bits in the AC and the memory operand (i.e., the intersection). The fields tested start at bit 0 and have a length equal to the count field value in the instruction.

If the AC tests greater than the memory operand (or they have corresponding bits in the test field for IAS) the computer executes the next instruction and proceeds from there. If the AC tests equal to the memory operand (or there are no corresponding bits in the test field for IAS) the computer skips the next instruction and proceeds from there.

If the AC tests less than the memory operand the computer skips the next two instructions and proceeds from there.

CAS and LAS translate to Group 34. VAS and IAS translate to Group 15. The following code performs the comparisons:

```
            ORG       0/215         EXAMPLE 16    CAS,VAS,ETC
            VFD       3/7,15/VLC    VARI-LENGTH COMPARE C+TV
            ORG       0/234
            VFD       3/7,15/COM    COMPARE C+TV
   * - - - - - - - - - - - - - - - *
            ORG       0/3303
   VLC      LIB       77            LOAD B WITH ALL 1'S MASK
            RC        D,,GEX        LOAD RC WITH COUNT FIELD
            SHIFT     B-R,RC        GENERATE MASK OF ZEROS
            PC        NB.C          MASK OFF MEMORY OPERAND
            AC        R.NB,LDB      MASK AC INTO B
            TA        GINT,COM      SKIP IF VARI-LENGTH COMP
            PF        B.C           TEST FOR INTERSECTION
            TAE       Z,*+5         EXIT ON INTERSECTING 1'S
   COM      PL        B-C,00-35     COMPARE AC TO MEMORY
            TA        EQU,*+3       SKIP IF EQUAL
            TAE       LESS          EXIT IF AC GREATER
            IC        R+1,S         BUMP 2 IF AC LESS
            IC        R+1,S,EXIT    BUMP 1 IF AC EQUAL, EXIT
```

EXAMPLE 17. Range Comparison

Consider an instruction to compare two numbers and treat them equal if the N most significant bits of their difference is zero; where N is a range specified in the instruction count field. The format is as follows:

0 11	12 14	17	18 20	21 35
+0312 INSTR. CODE	1	COUNT	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
RAS	+0312	Ranged Compare AC with Storage	ARI,SUB

The N (value of count field) most significant bits of the difference of the AC (bits P to 35) and the signed memory operand are examined as a comparison result.

If the comparison result is greater than plus zero, the computer takes the next sequential instruction.

If the comparison result is zero the computer skips the next instruction and proceeds from there.

If the comparison result is less than minus zero, the computer skips the next two instructions and proceeds from there.

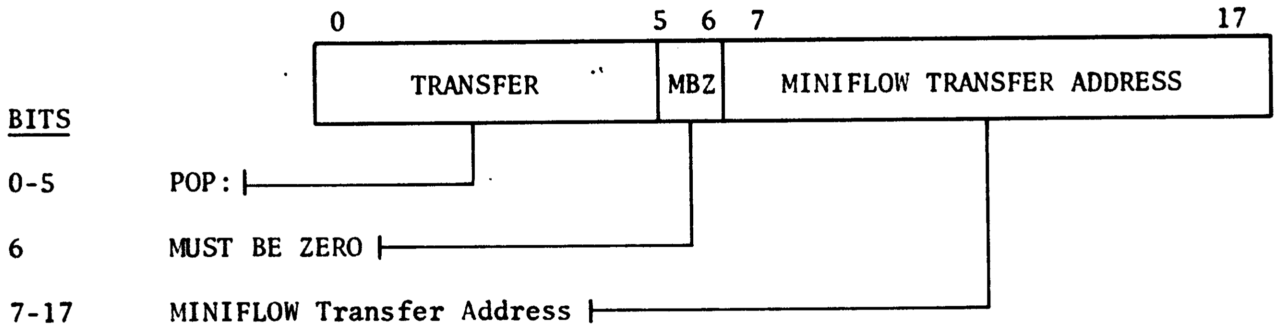
RAS translates to group 31. The MINIFLOW coding follows:

```
         ORG     0/231          EXAMPLE 17    RAS
         VFC     3/7,15/RAS     RANGE COMPARE C+TV
*  - - - - - - - - - - - - - - *
         ORG     0/3343
RAS      ALG     B-C,00-35      ALGEBRAIC DIFFERENCE
         LIC     77             LOAD MASK OF ALL ONES
         SHIFT   D-R,16         MOVE COUNT FIELD TO ADDR
         RC      D              MOVE COUNT FIELD TO RC
         SHIFT   C-R,RC         ZERO N SIGNIF. MASK BITS
         PE      B.NC           TEST SIGNIFICANT BITS
         TA      Z,*+3          SKIP ON SIGNIFICANT ZERO
         TAF     LESS           EXIT ON AC GREATER (+B)
         IC      R+1,S          BUMP 2 ON AC LESS  (-B)
         IC      R+1,S,EXIT     BUMP 1 IF EQUAL AND EXIT
```

TRANSFER POPs

There are two types of unconditional program sequence transfers for MINIFLOW. One is used to enter the Subroutine Mode Control and Transfer.

POP CODE	POP MNEMONIC	OPERATION
16	TRU	LOAD RB with bits 7-17 of mini-instruction
76	SMCT	Load RD with RB+1, load RB with bits 7-17 of mini-instruction, and set the subroutine mode flip-flop on.



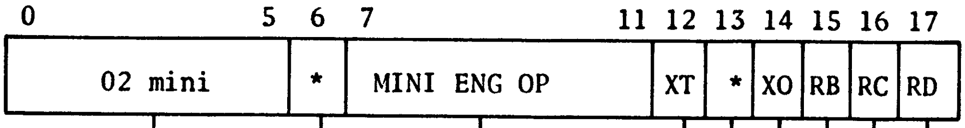
BITS	
0-5	POP:
6	MUST BE ZERO
7-17	MINIFLOW Transfer Address

NOTE

When the Subroutine Mode Control flip-flop is set on, then the exit condition, instead of returning to the scheduler, will cause a return to the main MINIFLOW program sequence from the subroutine. This return is accomplished by loading RD into RB and turning off the Subroutine Mode Control flip-flop.

MINI - MINI ENGINE OPERATION 02

The Mini-Engine may be controlled directly by MINIFLOW to perform arithmetic and data movement in the RB, RC and RD registers. The SOP defines the mini-engine operation and the TOP specifies the data movement.



BITS

0-5 POP:

6 NOT USED

7-11 SOP: Specifies the mini-engine operation performed

12 Flag: EXIT

13 NOT USED

14 CROSSOVER: Steers the SOP results over to the RD side of the mini-engine and the RD over to the Adder side. See Note 3.

15 ENABLE RB: Places the SOP results (or RD if crossover) into RB

16 ENABLE RC: Places the SOP results (or RD if crossover) into RC

17 ENABLE RD: Places the SOP results into RD if crossover (bit 14) is on.

Table 11. Mini-Engine SOPs

SOP CODE	MNEMONIC	NOTE	MINI ENGINE ADDER OUTPUT
00	ZERO		All bits are zero
30	RB	1	Contents of RB
31	RC		Contents of RC
34	RB+1	1	Contents of RB plus one
35	RC+1		Contents of RC plus one
14	RB-1	1,2	Contents of RB minus one
15	RC-1		Contents of RC minus one
16	RB-RC	1	Contents of RB minus contents of RC
36	RB+RC	1	Contents of RB plus contents of RC

NOTES

1. The value in RB is the mini-location of the current mini-instruction plus 1, i.e., the next mini-location.

2. If RB-1 is placed in RB the same mini-instruction will be executed again. This gives a dynamic halt in MINIFLOW unless the EXIT bit is on, then the exit is taken. If RB-1 is sent to RD with the exit bit on MINIFLOW will exit to the scheduler. This occurs whether or not MINIFLOW has entered the subroutine mode.

3. RD is loaded into RB or RC by setting crossover on, bit 14, and having RB bit 15 or RC bit 16 on. The Adder output is loaded into RD by setting the crossover on and having RD bit 17 on.

   The RZERO test flip-flop is set on a mini-instruction. The Adder output is tested for zero if crossover is off and the contents of RD is tested for zero if crossover is on.

EXAMPLE 18. Unpack

Consider a pair of UNPACK instructions. Both will take 9 bit bytes stored four to a word out of a string of consecutive words in mamory and unpack them from left to right storing them right justified one byte to a word into another string of consecutive words in memory. The difference between the two instructions is one zeros out the leftmost 27 bits of the put away words and the other leaves the leftmost 27 bits undisturbed. These instructions have the following format:

0 11	12	14	16	18 20	21 35
-073x INSTR. CODE	I	//	BYTE	TAG	BASIC BYTE MOVE COUNT

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
UPKZ	-0730	Unpack into zeros	GEX,GIN
UPKI	-0732	Unpack and insert	GEX

These instructions will unpack N BYTES into N consecutive locations where N is the value in the instruction address field after address modification (indexing and indirect addressing are both allowed). The address of the AC holds the address of the string of words to receive the unpacked bytes. The decrement of the AC holds the address of the string of words to be unpacked; and bits 16 and 17 of the instruction indicate which byte of the first word is to be the beginning of the byte string. Thus 00 in bits 16-17 starts with the left most byte and unpacks all bytes of the first word; while 11 in 16-17 would start with the right most byte and only unpack that byte before going on to unpack the next word.

Both UPKZ and UPKI translate to Group 73. The following code performs the unpacking:

```
          ORG      0/273        EXAMPLE 18    UNPACK
          VFD      3/6,15/UPK  UNPACK C+IV
*  - - - - - - - - - - - - - - *
          ORG      0/3400
UPK       PE       C,21-35      TEST FOR ZERO MOVE COUNT
          TAF      NZ           EXIT ON MOVE COUNT ZERO
          PB       B+C,21-35    ADD COUNT TO PUT ADDRESS
          R4       B,S          PICK AND PUT ADDR. TO R4
```

```
        PB      LDB           PICK ADDR. TO B ADDRESS
        MEM     B,SRD         FIRST PACKED WORD INTO D
        PB      C             COPY INSTRUCTION INTO B
        PC      B+C,15-17     DOUBLE 1ST BYTE ADDRESS
        LIB     06,12-17      PLACE BYTE ADD MASK IN B
        RC      B.NC,,GEX     LOAD SKIP DISTANCE TO RC
        PB      ZERO          ZERO B FOR UNPK TO ZERO
        MINI    RB+RC,RB      SKIP TO BYTE ENTRY POINT
ENTRY3  SMCT    INSERT+1      SPECIAL BYTE 3 ENTRY
        TRU     NEXT          FETCH NEXT PACKED WORD
ENTRY2  SHIFT   D-ROT,18      PRESHIFT BYTE 2
        TRU     BYTE2         STORE BYTE 2
ENTRY1  SHIFT   D-ROT,9       PRESHIFT BYTE 1
        TRU     BYTE1         STORE BYTE 1
BYTE0   SMCT    INSERT        STORE BYTE 0
BYTE1   SMCT    INSERT        STORE BYTE 1
BYTE2   SMCT    INSERT        STORE BYTE 2
BYTE3   SMCT    INSERT        STORE BYTE 3
NEXT    R4      R+1,LDD,GEX     LOAD AND INCR. ADDRESS
        R4      D,S,GEX         RESTORE ADDRESS TO R4
        MEM     D,SRD         LOAD D WITH UNPACK WORD
        TRU     BYTE0         STORE NEXT 0
* THE FOLLOWING IS THE INSERTION SUBROUTINE
INSERT  SHIFT   D-ROT,9       POSITION BYTE TO INSERT
        TA      GIN,*+2       SKIP IF UNPACK INTO ZERO
        MEM     R4-C,SRB      LOAD PUTAWAY WORD TO B
        PB      D,27-35       INSERT BYTE
        MEM     R4-C,SWB      STORE PUTAWAY WORD
        PC      C-1,21-35     DECREMENT THE MOVE COUNT
        TAE     Z             SUBROUTINE EXIT IF MORE
        EXIT                  FINAL EXIT TO SCHEDULER
```

Since there are several new techniques introduced in this example it would be well to point out some of these directly.

1. The original count value is added to the putaway address before storing in R4; thus, when the putaway address is used to address memory in the subroutine, the count is subtracted from it, giving the first putaway address initially. The same instruction that decrements the count (towards zero and a final exit) also effectively increments the putaway address.

2. The first byte specification field is transformed into an index value stored into RC and used in a mini-instruction to skip and preprocess the proper byte.

3. The first preprocessing for byte 3 enters the subroutine at "INSERT+1" to avoid shifting since the byte is already positioned for storing.

4. Rather than keeping a loop count for four bytes, four sequential SMCT instructions to the insert subroutine serve as an economical and easy way to keep track of a low count iterative process.

5. The Pickup Address for the words to be unpacked are in the decrement field but must be used out of the address field and must be incremented for each new word. The R4 instruction at "next" uses GEX control to bring the decrement field to the address position and increment it. The LDD TOP causes the incremented decrement to be stored in the address of D.

6. This object instruction emulation is terminated when the count goes to zero. Since the count is decremented and tested in the INSERT subroutine, this subroutine must either exit back to the calling sequence if the count is not yet zero, or exit to the scheduler if the count goes to zero. This is accomplished by a conditional exit from the subroutine followed by an unconditional exit to the scheduler (see note 2 in the preceding writeup on the MINI instruction).

EXAMPLE 19. Additive Indexed Store Index Instructions

Consider a set of instructions to insert the index into either the address field or the decrement field in either true value or complement value. The following format and op-codes will be used:

0 11	12		15 17	18 20	21 35
±053x INSTR. CODE	I	/	INDEX	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
IXA	+0531	Insert Index in address	none
IXD	-0531	Insert index in decrement	GEX
IXCA	+0532	Insert complement in address	GIN
IXCD	-0532	Insert complement in decrement	GIN,GEX

Also assume that, for the purposes of illustration, indexing is ADDITIVE in this object machine. In this type, the index value is added to the address field; since the IC-M9 does subtractive indexing in its WIS, the indexes will be held in their registers in complement form.

These instructions will then wish to complement the index value on storing for the "Store True" and store directly for the "Store Complement". These instructions are indexable and use the tag field for indexing and bits 15 to 17 to specify which index to store.

These instructions all translate to group 53; the following MINIFLOW provides the emulation.

```
          ORG      0/253         EXAMPLE 19    IXA,IXD,ETC
          VFD      3/6,15/IX     INSERT INDEX C+TV
   * - - - - - - - - - - - - - - - *
          ORG      0/1600
   IX     MEM      C,SRB,GEX     LOAD WORD TO B WITH GEX
          SHIFT    0-R,3         MOVE INDEX SPEC. TO TAG
          XR       R,LDB         INSERT INDEX INTO B
          PB       NB+1,21-35,GIN COMPLEMENT IF 'TRUE'
          MEM      C,SRB,GE      STORE WITH GEX AND EXIT
```

EXAMPLE 20. Cumulative Index Concept

Considering an idea similar to the Multi-tag mode except instead of ORing the multiple indexes together it ADDS them together before modifying the address value. This could be accomplished by maintaining the cumulative sums in index 3, 5, 6, and 7. Whenever an index was to be changed it would also change the "cumulative" indexes associated with it; for instance if XR2 changes XR3, XR6 and XR7 would also be appropriately changed.

This mode of indexing is called the Cumulative Index Mode. A special Instruction would be used to turn it on or off by setting a general indicator called CXM. This same instruction would save indexes 3, 5, 6 and 7 in control core and turn on the CXM flip-flop and generate the cumulative values into 3, 5, 6 and 7 when entering the mode. It would restore the original single index values from control core and turn off the CSM flip-flop when leaving the mode.

The means to accomplish this is left as a student exercise; however, the next example may be of some help.

EXAMPLE 21. Additive Indexed Load Index - Cumulative

Consider a set of instructions to load the indexes from an immediate value or from a word in memory. These instructions are themselves indexable so that on the Load Index Immediate the address and the tagged index are combined and loaded into the specified index. If these indexes are the same, then incrementing or decrementing will take place since additive indexing is used. If indirect address is used with LIX, Post indexing is effective on the indirect address value. We will use the following format and op codes:

0 11	12	15 17	18	21 35
+063x INSTR. CODE	I /	INDEX	TAG	BASIC OPERAND ADDRESS

MNEMONIC	OP-CODE	OPERATION	PRECONDITIONS SET ON
LIX	+0630	Load Immediate to Index	None
LAX	+0632	Load Address to Index	None
LDX	-0632	Load Address complement to Index	GIN
LDCX	-0634	Load Decrement complement to Index	GIN,GEX

The LIX +0630 translates to group 62 and the rest ±0632 and ±0634 translate to group 63.

The MINIFLOW to accomplish this emulation is below:

```
         ORG      0/262         EXAMPLE 21    LIX,LAX,ETC
         VFD      3/6,15/LIX LOAD XR IMMEDIATE C+TV
         VFD      3/6,15/LX  LOAD XR FROM MEMORY C+TV
 * - - - - - - - - - - - - - - *
         ORG      0/1500
LX       MEM      C,SRC,GEX  LOAD MEMORY WORD BY GEX
         PC       NC+1,21-35,GIN COMPLEMENT IF TRUE
LIX      TG       CXM,*+3    TEST FOR CUMULATIVE MODE
         SHIFT    D-R,3      MOVE INDEX SPEC. TO TAG
         XR       C,S,EXIT   LOAD SINGLE INDEX, EXIT
         LIBC     20,18-20   ZERO LOW B, TAG B WITH 2
         PD       LDB,33-35  INDEX SPEC TO B ADDRESS
         SHIFT    D-R,3      MOVE INDEX SPEC TO D TAG
         RC       B          INDEX SPEC TO RC
         MINI     RB+RC,RB   USE SPEC AS XFER VECTOR
TAGO     EXIT                EXIT ON NO INDEX SPEC.
   1     TRU      UPDATE     LOAD 1- UPDATE, AND EXIT
   2     TRU      UPDATE     LOAD 2- UPDATE, AND EXIT
   3     TRU      SET2       START WITH 2, THEN DO 1
   4     TRU      UPDATE     LOAD 4- UPDATE, AND EXIT
   5     PB       B-1,18-20  SET B TAG TO 1 TO START
   6     TRU      SET12      USE B TAG AS IS TO START
   7     SMCT     SET4       START WITH 4, THEN 2 + 1
SET2     LID      20,18-20   SET TO LOAD 2
         SMCT     UPDATE     LOAD 2- UPDATE, CONTINUE
         LID      10,18-20   SET TO LOAD 1
         TRU      UPDATE     LOAD 1- UPDATE AND EXIT
SET12    PD       B,18-20    SET TO LOAD 1 OR 2
         SMCT     UPDATE     UPDATE AND FINISH WITH 4
SET4     LID      40,18-20   SET TO LOAD 4 AND UPDATE
UPDATE   XR       C-R,LDB    LOAD UPDATE VALUE INTO B
         PC       D,18-20    LOAD OR MASK INTO C
         TRU      *+2
         PD       BUC,18-20  PUT NEXT TAG VALUE IN D
         PB       D,18-20    PLACE TAG BACK IN B
         XR       R+B,S      UPDATE INDEX VALUE
         PB       B+1,18-20  INCREMENT TAG VALUE
         TAF      NZ,*-4     EXIT AFTER TAG OF 7 DONE
CXM      DGI      4 .
```

The preceding example illustrated that the same subroutine may be entered at different points; it also shows that a SMCT may be used for a self-returning entry or a TRU for a final entry. The use of a transfer table is also illustrated again.

EXAMPLE 22. Additive Indexed Transfer With Index Link

An indexed transfer instruction which saves a return link value in specially designated index fits in with the foregoing example on Additive Indexed Load Index-Cumulative. In this transfer instruction the indexing is also additive and/ the same routine may be used to save the return value in the index. The op-code and instruction format is:

0 – 11	12 – 15	15 – 17	18 – 20	21 – 35
-0070 INSTR. CODE	I /	INDEX	TAG	BASIC TRANSFER ADDRESS

MNEMONIC	OP-CODE	OPERATION
TLX	-0070	Transfer and Link with Index

This op code translates to group 7 and the following MINIFLOW puts the transfer address in the IC and prepares the old IC value to be placed into the index. It then transfers to the same routine as was used in the previous example to load the index:

```
        ORG     0/207         EXAMPLE 22   TLX
        VFD     3/6,15/TLX INDEX LINK TRANSFER C+TV
*  - - - - - - - - - - - - - - - - *
        ORG     0/0774
TLX     IC      R-1,LCP       BACK DOWN IC, LOAD TO P
        IC      C,S           PUT TRANSFER  ADDR IN IC
        PC      NB+1,21-35 PUT IC COMPLEMENT IN C
        TRU     LIX           GO PUT C INTO PROPER XR
```

EXAMPLE 23. MINIFLOW Address Modification

It should not be supposed that the general manner in which address modification and instructions are decoded is limited by the wired-in-sequence. Rather, the case is, that the wired-in-sequence is used as a speedup technique to enhance a particular style of address modification (i.e., that used in the IBM 709X and 704X). This is a special feature not found in other micro-program machines.

Address modification and instruction interpretation is still extremely flexible when done in MINIFLOW. Consider a basic instruction format like the following:

0 – 8	9 – 11	12 – 13	14	15 – 17	18 – 20	21 – 35
BASIC OP CODE A	MOD B	AC C	± D	AXC E	TAG F	BASIC ADDRESS FIELD G

FIELD	BITS	PURPOSE AND INTERPRETATION
a.0-8	9	Basic operation
b.9-11	3	Op-code modifier
c.12-13	2	Accumulator designator (0,1,2,3)
d.14	1	Add (0) or subtract (1) index value

FIELD	BITS	PURPOSE AND INTERPRETATION
E.15-17	3	Address mode and index control 0-indexed immediate operand 1-indexed direct address-leave XR unchanged 2-indexed direct address-dec.XR by 1 test for ZERO 3-indirect post indexed address (allows double indexing) 4-indexed indirect address-leave XR unchanged 5-indexed indirect address-inc.XR by X field, test for carry 6-indexed indirect address-inc.XR by 1, test for X field 7-indexed indirect address-dec.XR by 1, test for X field
f.18-20		Index specification field
g.21-35		Basic address field

Assume an indirect address word format like this:

0 2	3 17	18 20	21 35
AXC E	INDEX CONTROL FIELD X	TAG F	BASIC ADDRESS FIELD G

With this type of address modification, multi levels of indirect are possible, double indexing is possible, immediate operands are generally available, automatic index control and testing is possible, plus the complexities of their combinations. For the purposes of this example we will consider only the MINIFLOW for the general address modification subroutine, INDEX. We will assume however, that:

1. The Reset machine MINIFLOW will turn on 3 general indicators - ALTER, USE 1, and DEC (these general indicators will always be on initially).

2. There will be object machine instructions to transfer on FLAG on or off so the index tests may be used.

3. The C&TV of indexable object instructions will call for indexingD i.e., VFD 3/2,15/XXX) but appropriate operand fetches will be done in MINIFLOW.

4. Indexable object instructions will be indexed by a SMCT INDEX call, following which the final address will be in D21-35 with the general indicator IMMED set on if D21-35 is an immediate operand.

5. The left half of the original object instruction will still be in C0-17 unaltered.

The MINIFLOW subroutine to accomplish this general address modification follows:

```
        ORG     0/400        EXAMPLE 23   MINI-INDEX
* THIS  INDEX SUBROUTINE IS ENTERED BY- SMCT INDEX
INDEX   LIBC    10,12-14     CLEAR B AND SET MASK BIT
        PE      B.C          TEST INDEX POLARITY
        TA      NZ,*+2       LEAVE AS SUBTRACT IF A 1
        TGS     ADD          SET TO ADD IF A ZERO
        PC      LDB,33-35    MOVE AXC INTO B ADDRESS
IXCONT  RC      B            MOVE AXC INTO RC
        MINI    RB+RC,RB     JUMP TO TRANSFER TABLE
DATA    TGS     IMMED        SET IMMEDIATE DATA FLAG
DIR     TGF     ADD,ADDXR    EXIT IF SUBTRACTIVE XR
AXC2    TRU     DECDIR       INDEXED DIRECT + DEC XR
AXC3    TRU     IXPOST       INDIRECT POSTINDEXED
AXC4    TGR     ALTER,AXC7   INDEXED INDIRECT- LEAVE
AXC5    TGR     USE1,AXC7    INDEXED INDIRECT- XR+XF
AXC6    TGR     DEC          INDEXED INDIRECT- XR+1
AXC7    TGF     ADD,*+2      INDEXED INDIRECT- XR-1
        XR      R+C,LDD      DO ADDATIVE INDEXING
        MEM     D,SRD        READ INDIRECT WORD TO D
        TGS     ALTER,XDEL   SKIP TO ALTER XR, SET ON
        PC      D,21-35      PUT NEW ADDRESS IN C
        TRU     INDIR+1      INDEX + ANALYSE INDIRECT
DECDIR  TGS     FLAG
        PC      C+1,21-35    OFSET XR DEC FOR XR ADD
        XR      R-1,S        DECREMENT THE INDEX
        TA      Z,DIR        FLAG IF ZERO AND SKIP
        TGR     FLAG,DIR     NO FLAG IF NOT ZERO-SKIP
ADDXR   TGR     ADD          RESET INDEX POLARITY
        XR      R+C,LDD,EXIT ADDATIVE INDEX + EXIT
XDEL    TGS     FLAG         PRESET FLAG ON
        PCD     C,18-20      SWAP TAG FIELDS-OLD TO D
        PD      LDC,21-35    LOAD X-FIELD TO C ADDR.
        TGS     USE1,BY1     SKIP TO INC/DEC, SET ON
        XR      R+C,S        ADD X-FIELD TO OLD INDEX
        TA      CAR,INDIR    FLAG IF CARRY AND SKIP
        TGR     FLAG,INDIR   NO FLAG IF NO CARRY-SKIP
BY1     TGS     DEC,DOWN1    SKIP IF XR-1, SET ON
        XR      R+1,S        INCREMENT XR
        TRU     *+2
DOWN1   XR      R-1,S        DECREMENT XR
        TA      Z,INDIR      FLAG IF EQUAL AND SKIP
        TGR     FLAG,INDIR   NO FLAG IF UNEQUAL- SKIP
IXPOST  MEM     C,SRD        READ UNINDEXED IND. TO D
        PCD     C,18-35      SWAP TAGS AND ADDRESSES
        TG      ADD,*+3      TEST FOR INDEX POLARITY
        XR      C-R,LDD      POST-INDEX INTO D, MINUS
        TRU     *+2
        XR      R+C,LDD      POST-INDEX INTO D, PLUS
INDIR   PCD     C,18-35      SWAP BACK TAGS,ADDRESSES
```

```
XR        C-R,LCD      PRE-INDEX INTO D, MINUS
PB        ZERO         CLEAR OUT B
SHIFT     BD-L,3       MOVE AXC FIELD TO B ADD
SHIFT     D-R          MOVE D BACK IN POSITION
TRU       IXCONT       CONTINUE THE INDIRECT
```

COMPARISON OF EMULATION AND OTHER INSTRUCTIONS

A BINARY to BCD CONVERSION routine might take a binary number in the AC (less than 20 bits right justified) and convert it to six 6 bit decimal characters in the MQ: It would also turn on a flag (divide check) if the number were too large to convert. Such a routine written in object code is given below:

```
        TXS CONVERT,4                          CALL CONVERSION SUBROUTINE
------------------------------------------------------------------------------
CONVERT LDQ=0
        VDP=Ø6065000,,4                        100,000B31
        VDP=Ø47040000,,6                        10,000B25
        VDP=Ø372000000,,6                        1,000B19
        VDP=03100000000,,6                         100B13
        VDP=024000000000,,6                         10B7
        VDP=0200000000000,,6                         1B1
        TRA,4                                  RETURN TO CALLING SEQUENCE
```

The time to run this routine on various machines:

```
7094 - 46 usec.               7090 - 76.3 usec.
7044 - 80 usec.               7040 - 176 usec.
IC-M9 - 115 usec. in EMULATION MODE
```

If the IC-M9 is mini-programmed to include a BINARY TO BCD CONVERT instruction, it could leave the AC unchanged, and the MQ unchanged if divide check is set. Assume CTQ (convert to MQ) has the following instruction format:

0 11	12 35
+0110 CODE	//

An op-code of +0110 translates to Group 11 and the following MINIFLOW would perform the conversion:

```
        ORG       0/211        EXAMPLE 24    CTQ MAXI
        VFD       18/CTQ       CTQ C+TV
* - - - - - - - - - - - - - - - *
        ORG       0/677
CTQ     PD        ZERO         SO 2 HIGH BITS WILL BE 0
        CMI       LDC,100M     LOAD=0/6065000
        SHIFT     DIV,4        DIVIDE BY 100,000 B31
        TGR       PDCK,2BIG    SKIP IF OVER 6 DIGITS
        CMI       LDC,10M      LOAD=0/47040000
        SHIFT     DIV,6        DIVIDE BY 10,000 B25
        CMI       LDC,M        LOAD=0/372000000
        SHIFT     DIV,6        DIVIDE BY 1,000 B19
        CMI       LDC,C        LOAD=0/3100000000
```

```
         SHIFT    DIV,6        DIVIDE BY 100 B 13
         CMI      LDC,X        LOAD=0/240000000000
         SHIFT    DIV,6        DIVIDE BY 10 B7
         CMI      LDC,I        LOAD=0/200000000000
         SHIFT    DIV,6        DIVIDE BY 1 B1
         MQ       D,S,EXIT     STORE RESULT IN MQ- EXIT
2BIG     TGS      DCK          SET DIVIDE CHECK
         EXIT                  EXIT FOR DIVIDE CHECK
*THE FOLLOWING ARE THE CONSTANTS USED FOR CONVERSION
         ORG      0/62
100M     OCT      6,65000          100,000 B31
10M      OCT      47,40000          10,000 B25
M        OCT      372,0              1,000 B19
C        OCT      3100,0               100 B13
X        OCT      24000,0               10 B7
I        OCT      200000,0               1 B1
```

The time to execute this CTO instruction on the IC-M9 is 32.2 usec. This is only 71.5% of the time required by the 7094 to execute the CONVERT subroutine.

The second instruction consists of floating a fixed point number. It is assumed that a signed whole number is in the AC right justified and may have up to 36 bits of significance. To float this number in object machine code the following FLOAT routine is commonly used;

```
FLOAT       LRS  8
            ØRA  =012430000000
            STØ  TEMP
            CLA  =01233000000
            LLS  8
            FAD  TEMP
```

If the number has typically 12 bits of significance, the time to execute this on a 7094 (with best case overlap) is 26 usec.

This process may be emulated on the IC-M9 with the following fix to float instruction. Assume FLT (float the AC) has the following format:

0 11	12 35
+0670 CODE	//

The op-code +0670 translates to group 67 and the following MINIFLOW would do the float operation:

```
         ORG      0/267        EXAMPLE 25    FLOAT MAXI
         VFD      18/FLOAT     FLOAT C+TV
*  - - - - - - - - - - - - - - - *
         ORG      0/2700
FLOAT    PE       B,00-35      TEST FOR AC ZERO
         TAE      N/           EXIT IF ZERO-IE ALL DONE
         PD       ZERO         CLEAR D
         SHIFT    BD-R,,       MAKE ROOM FOR EXPONENT
SHIFT    D-R,9        CHANGE TO FLOATING FORM
LIB      20,00-02     LOAD EXPONENT BASE VALUE
SHIFT    BD-N         NORMALIZE THE MANTISSA
LIB      44,03-08     SCALE THE EXPONENT
RC       LDC,00-08    LOAD THE NORMALIZE COUNT
P3       B-C,00-08    ADJUST THE EXPONENT
AC       B,S,EXIT     RESULT TO AC AND EXIT
```

Again assuming a 12 bit number is being floated, the time to execute this FLT instruction by the IC-M9 is 15.575 usec. or 60% of the time required by the 7094 to execute the FLOAT routine.

SECTION III

IC-M9 SYSTEM DESCRIPTION AND INSTRUCTION REPERTOIRE

SCHEDULER AND PROGRAM LEVELS

The scheduler is the basic sequence controller of the IC-M9. It is a priority decision network which responds to I/O data buffer servicing and I/o termination requirements. It also responds to some of the operator switches on the console and to the real time clock servicing requirements. In addition it is partially controlled by MINIFLOW to service trap situations, self interrupts, i.e., Hang or Delay, and normal program operation.

The scheduler buffers requests for attention in a set of interrelated flip-flops which are grouped to reflect various levels of urgency. There are four such levels:

LEVEL 1 Channel A data buffer servicing

LEVEL 2 Channel B data buffer servicing

LEVEL 3 Channels A and B termination servicing Real time clock servicing and console request servicing

LEVEL 4 Trap servicing
Object instruction execution

The scheduler determines which request for service to honor next, selects the level and turns the Channel B flip-flop on or off for the level. Only one level may be selected at any one time. It also selects the entry point to the wired-in-sequence.

Each level has flip-flops to buffer the request for servicing and to control the manner in which these requests are honored. There are four types of flip-flops:

REQUEST Buffers a request for new service.

RETURN Buffers a request to continue a self interrupted job.

JIP (Job-in-Progress) records that a request was honored and servicing for the job has not been completed.

HANG Records that a self interrupted job is not yet ready to return for completion.

Self interrupt allows a MINIFLOW routine to interrupt itself when it has determined that it cannot properly continue until another level has had a chance for servicing. Self interrupt may be done with either a DELAY or a MISC HANG instruction. DELAY inhibits the scheduler from honoring the delayed level for 75 microseconds. HANG requires that another active level unhang the hung level for it to again be honored by the scheduler. When a DELAY or HANG is executed in MINIFLOW, the D, RB, and RD registers and subroutine mode are preserved for the particular level. They are restored when the scheduler again honors that level and the job is resumed; restoring RB causes the job to continue at the MINIFLOW instruction following the DELAY or the HANG.

Delay logic is not provided for level 4; if DELAY is executed in level 4 it will lock out level 4 until the level is reset.

A DELAY or a HANG issued in either level 1 or level 2 will lock out both levels until the delayed or hung level resumes.

LEVEL 1 is requested in the scheduler when the Channel A buffer needs servicing. LEVEL 2 is requested in the scheduler when the Channel B buffer needs servicing. When the scheduler honors one of these requests it will reset the request when LEVEL 2 is selected and the Channel B flip-flop is set. The MINIFLOW routine that services a buffer request must acknowledge the request in the channel. If this is not done logic will cause another request in the scheduler. MINIFLOW acknowledges the channel by executing the instruction MISC ACK.

LEVEL 3 is requested in the scheduler when any of the following occur:

1. A channel terminates activity. The state of the Channel B flip-flop indicates which channel terminated.

2. An interval timer interrupt occurs. This will happen 60 times each second unless the timer is turned off.

3. A console interrupt occurs when the operator presses one of the following console keys:

   a. general reset
   b. clear storage
   c. load card
   d. load tape
   e. execute entry
   f. execute display

The scheduler level 3 will be reset when it is honored by MINIFLOW. MINIFLOW must reset the specific condition that caused the level 3 request. The scheduler will set a general indicator (CON,SOP74) if the honored request was a console function (timer,case 2, or key,case 3). This indicator may be tested and reset and the interrupt register may be examined by MINIFLOW to determine the cause of the level 3 interrupt. The originating requests are reset by the following MINIFLOW instructions:

1. for channel terminate - MISC RTER

2. for timer interrupt - MISC RTI

3. for console key interrupt - MISC RCN

LEVEL 4 is requested in the scheduler when a trap request is set by a MISC SRT or when a program request is set by either a MISC SR4 or by the START KEY on the console. When the scheduler honors a trap request it will reset the trap request. When the scheduler honors a program request, the program request will be reset only if the MANUAL SWITCH is down on the console. The scheduler gives a program request the lowest priority. This allows the program request to be continually honored until a higher priority request is set; thus providing normal object program execution flow with the interrupts to service the I/O and console requirements.

Figure 6 illustrates the scheduler action, the request priority, and the relationship to the wired-in-sequence and MINIFLOW.

The term W-I-S is used as an abreviation for Wired-in-seauence. The suffixs -T or -F is appended to scheduler control flip-flops to indicate their state (true or false).

WIRED-IN-SEQUENCE

There are two wired-in-sequences. The YA W-I-S is used to initiate the start of a MINIFLOW routine for execution of an object instruction or an interrupt function; the YS/R W-I-S is used to save and restore MINIFLOW program status during a delay or a hang/unhang operation.

THE YA WIRED-IN-SEQUENCE

There are seven steps to the YA W-I-S, some of which may be skipped. Each step involves accessing either the control memory or main memory. Other operations may also be performed in some steps.

The W-I-S steps are defined below:

W-I-S Step	Memory Accessed	Primary Function
YA0	Control	Fetch Alternate Instr. Address
YA1	Main	Fetch Object Instruction
YA2	Control	Fetch Control & Transfer Vector
YA3	Main	Fetch Indirect Address
YA4	Main	Fetch Memory Operand
YA5	Control	Fetch First Mini-Pair
YA6	Either	Execute MINIFLOW Program

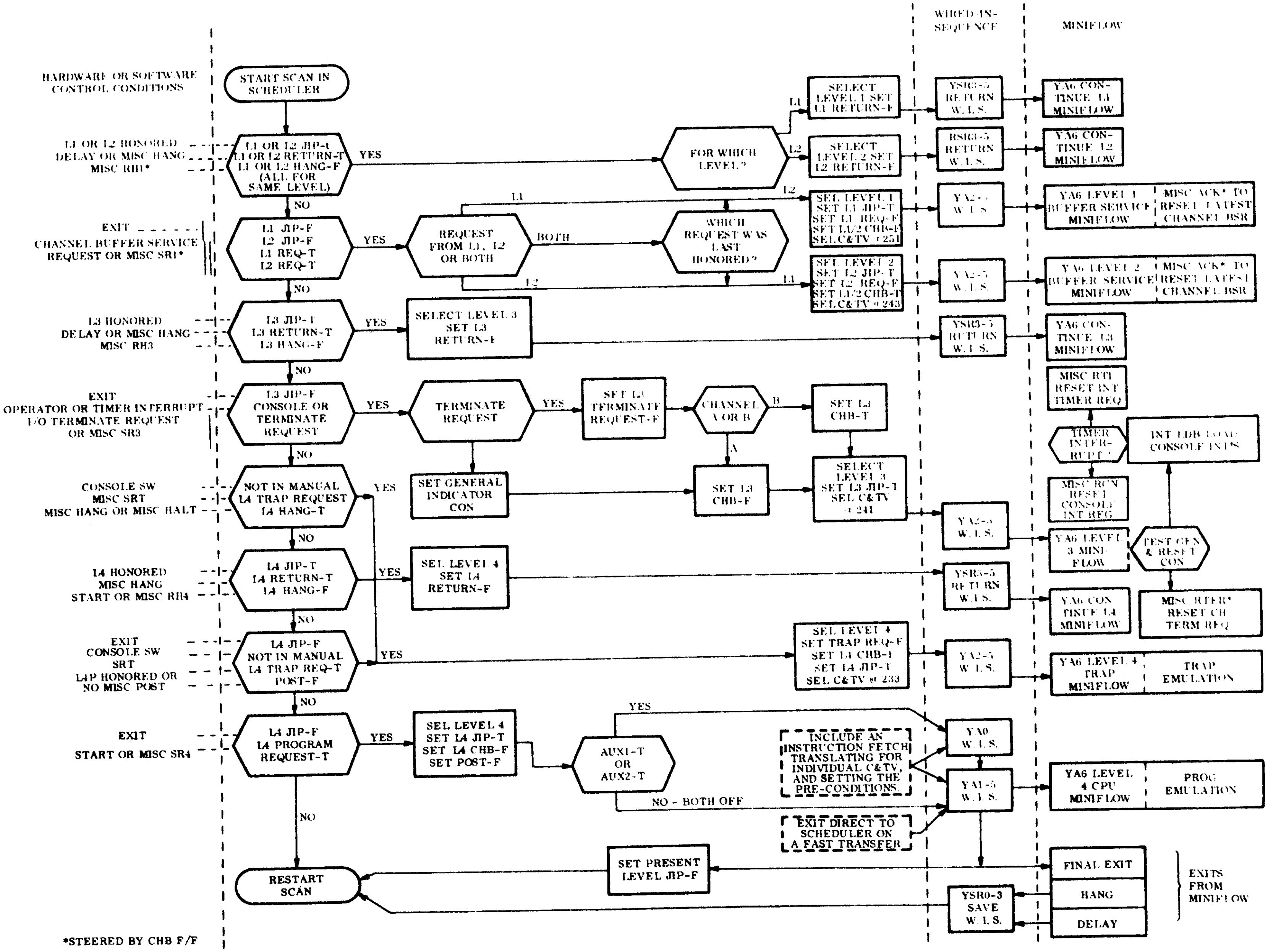


Figure 6. Flow Diagram of IC-M9 Scheduler

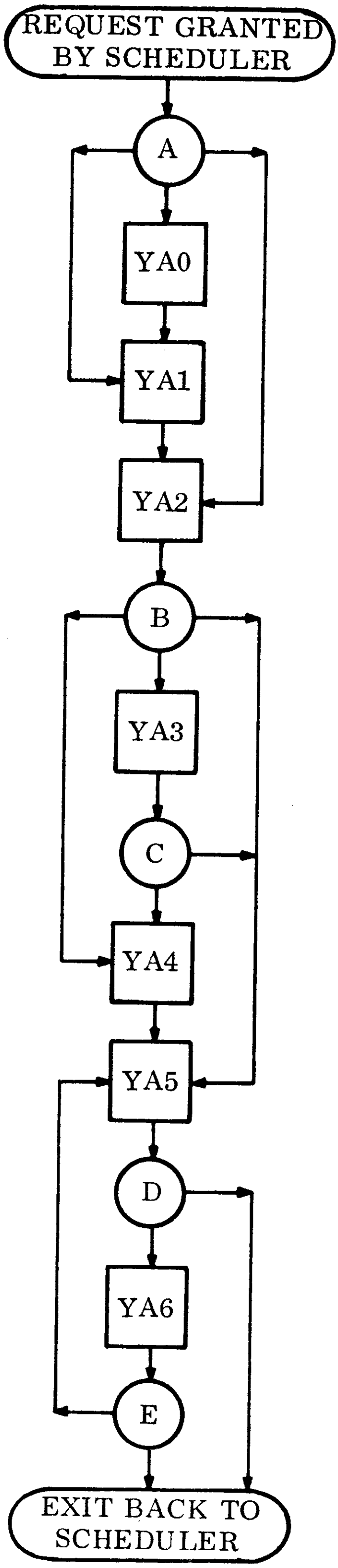


Figure 7 illustrates the possible entries and paths of the W-I-S. Each path is labeled with a letter which keys it to the ajoining test. In the text, the INDEXED ADDRESS is defined to be the value of the address field (C,21-35) minus the value in the index register as specified by the TOP field (D,18-20); if C or D are altered in an operation the initial values in these fields are used.

SEQUENCE OF STEPS IN WIRED-IN-SEQUENCE

A The Scheduler enters the W-I-S at:

1. YA0 - program request with AUX1 or AUX2 on.

2. YA1 - program request with neither AUX on.

3. YAW - L1, L2, L3 or L4 trap request.

YA0 Fetches the auxiliary instruction counter from control memory (at 04 if AUX 1 or 10 if AUX 2) and loads it into B.

YA1 Fetches the object instruction from main memory and loads it into C and D. The IC is used for the address unless AUX is on, then B is used.

YA2 Fetches the control and transfer vector from control memory and loads the 11 right most bits into RB and the other 7 bits are loaded into LK1 through LK7. The mini-location of the C&TV is as follows: Level 1 at 251, level 2 at 243, level 3 at 241, level 4 trap at 233, level 4 program as determined by the translators (Table 12) examining the op-code (D,0-11). The preconditions are also set and the AC is unconditionally loaded into B.

B After YA2, the next state of W-I-S is defined below:

1. YA3 - LK1, LK2, D,11-12 all are on, i.e., indirect addressing

2. YA4 - LK3 is on, i.e., memory operand fetch

3. YA5 - LK3 not on, no memory operand fetch

YA3 Uses the INDEXED ADDRESS to fetch the indirect address word from main core and to place the right half of the word into bits 21-35 of both C and D.

Figure 7. Possible Paths of the YA W-I-S

C After YA3, the next state of W-I-S is defined below:

1. YA4 - LK3 is on, i.e., memory operand fetch

2. YA5 - LK3 is off.

YA4 Fetches the memory operand from main memory as addressed by:

1. C,21-35, if LK1 and LK2 are both off.

2. The INDEXED ADDRESS if either LK1 or LK2 is on.

   The INDEXED ADDRESS is placed in D,21-35, if LK2 is on. The memory operand is loaded into C. LK1 and LK2 are reset to prevent indexing in YA5.

YA5 Performs the following functions:

1. Place the INDEXED ADDRESS in D if LK2 is on.

2. Place the INDEXED ADDRESS in C if LK1 is on.

3. Either execute a fast transfer or fetch the first mini-pair from control core addressed by RB.

Details of YA5 are shown in Figure 8.

D After YA5, the next state of W-I-S is defined below:

1. YA6-LK7 is off or transfer trap mode is on and D,0-11, is not ±0021 (trap transfer).

2. Scheduler-LK7 is on and transfer trap mode is off or D,0-11, is ±0021.

YA6 Execute MINIFLOW starting with the instruction fetched in YA5 and continue with new instructions from control core using RB as the program sequence counter.

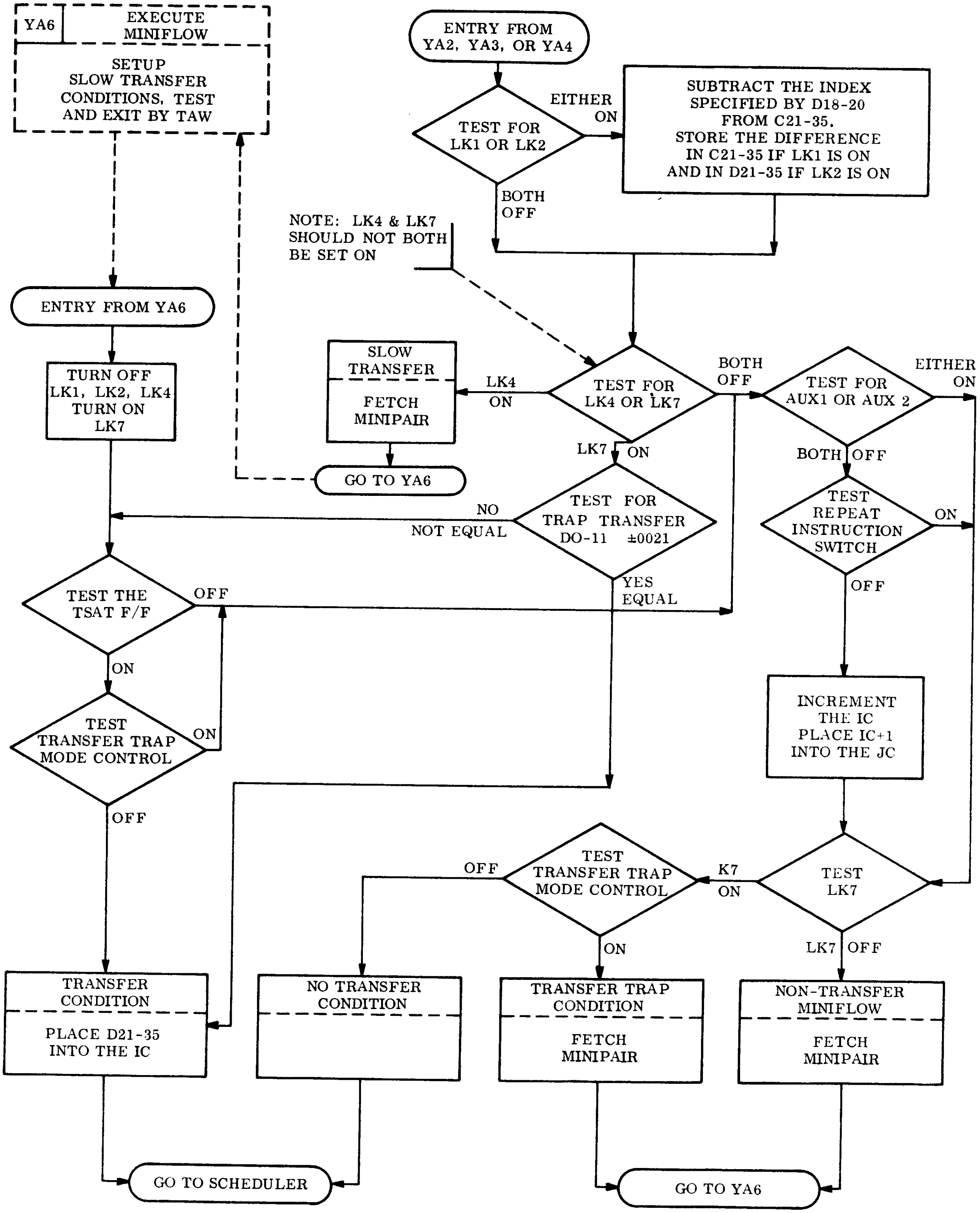
E The mini-instruction used to leave YA6 determines what state is next entered.

1. YA5 - The TAW instruction was executed.

2. Scheduler -EXIT when not in subroutine Mode.

THE YSR WIRED-IN-SEQUENCE

The YSR W-I-S is divided into two sequences. YSR 0-2 is used to SAVE the contents of the D, RB, RD registers and the status of the subroutine mode control flip-flops, YSR 3-5 is used to RESTORE the contents of these registers and the status of the subroutine.



YA6
EXECUTE MINIFLOW
SETUP SLOW TRANSFER CONDITIONS, TEST AND EXIT BY TAW
ENTRY FROM YA2, YA3, OR YA4
TEST FOR LK1 OR LK2
EITHER ON
SUBTRACT THE INDEX SPECIFIED BY D18-20 FROM C21-35. STORE THE DIFFERENCE IN C21-35 IF LK1 IS ON AND IN D21-35 IF LK2 IS ON
BOTH OFF
NOTE: LK4 & LK7 SHOULD NOT BOTH BE SET ON
ENTRY FROM YA6
TURN OFF LK1, LK2, LK4 TURN ON LK7
SLOW TRANSFER
FETCH MINIPAIR
LK4 ON
TEST FOR LK4 OR LK7
BOTH OFF
TEST FOR AUX1 OR AUX 2
EITHER ON
LK7 ON
GO TO YA6
NO NOT EQUAL
TEST FOR TRAP TRANSFER DO-11 ±0021
BOTH OFF
TEST REPEAT INSTRUCTION SWITCH
ON
TEST THE TSAT F/F
OFF
YES EQUAL
OFF
ON
TEST TRANSFER TRAP MODE CONTROL
ON
INCREMENT THE IC PLACE IC+1 INTO THE JC
OFF
OFF
TEST TRANSFER TRAP MODE CONTROL
K7 ON
TEST LK7
ON
LK7 OFF
TRANSFER CONDITION
PLACE D21-35 INTO THE IC
NO TRANSFER CONDITION
TRANSFER TRAP CONDITION
FETCH MINIPAIR
NON-TRANSFER MINIFLOW
FETCH MINIPAIR
GO TO SCHEDULER
GO TO YA6

Figure 8. Flow Chart of YA5 Operations

The SAVE sequence is entered by executing a DELAY instruction (540000) or a MISC HANG (065400) or a MISC HALT (066000). HALT should be executed only in Level 4. DELAY should not be executed in Level 4.

When DELAY is executed a RETURN request is set in the scheduler after 75 microseconds, after this the scheduler is free to start a RESTORE sequence. When a MISC HANG or a MISC HALT is executed the HANG flip-flop is set for that level; when the HANG flip-flop is reset by MINIFLOW in another level then the scheduler is free to start a RESTORE sequence. The RESTORE sequence will cause MINIFLOW to be re-entered at the mini-location just after the DELAY or HANG instruction with the contents of the D and RD registers and the state of the subroutine flip-flop restored to the status that existed when the DELAY or HANG was executed. There is a separate save area for each level; the following table gives the area and information stored in the save sequence:

Saved	Bit	Word Address of Control Memory			
Data	Positions	Level 1	Level 2	Level 3	Level 4
RB	23-35	40	44	50	54
SMCT F/F	22	41	45	51	55
RD	23-35	41	45	51	55
D	00-35	42	46	52	56

NOTE: Words 43, 47, 53 and 57 are not altered.

TRANSLATORS

During wired-in-sequence step YA2, the translators set up the precondition control flip-flops and select the Control and Transfer vector to be used to control the other states of the wired-in-sequence. For levels 1, 2, 3 and level 4 trap request, the C&TV selected is defined by the request being honored by the scheduler. For a level 4 program request, the object instruction is loaded into D during YA1 of the W-I-S; the translator examines bits 0-11 of D and generates a group code which specifies the C&TV to be used.

For most op-codes with bits 1 and 2 equal to zero the group code is determined by bits 3-8 of the object instruction. For example, op-code +0153 translates to group 15 as does op-code -0155. However, there are many exceptions to this rule as illustrated in the comprehensive translater Table 12.

The Comprehensive Translator Table gives the group number and the setting of the precondition control flip-flops for each object instruction, except those having a 1 in bit positions one or two. The first three octal digits of an instruction are given to the left of the table and the fourth octal digit is in the heading line. Each table item consists of a 2 digit octal number and a 3 digit octal number. The 2 digit number is the group number and when added to octal 200 it forms the mini-location of the associated C&TV. The 3 digit number describes the precondition controls and is interpreted as follows:

1st Digit	ARI	MAG	SUB
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

2nd Digit	GEX	GIN
0	OFF	OFF
1	OFF	ON
2	ON	OFF
3	ON	ON

3rd Digit	GOP9	GOP10	GOP11
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

Following the table are additional notes on special considerations.

Table 12

Comprehensive Translator - ±0000 to ±0177

OP CODE	xxx0		xxx1		xxx2		xxx3		xxx4		xxx5		xxx6		xxx7	
+000x	00	002	00	002	00	002	00	002	00	002	00	002	00	002	00	002
-000x	00	002	00	002	00	002	00	002	00	002	00	002	00	002	00	002
+001x	01	020	01	020	01	020	01	020	01	020	01	020	01	020	01	020
-001x	01	021	01	021	01	021	01	021	01	021	01	021	01	021	01	021
+002x	02	000	02	000	02	000	02	000	02	000	02	002	02	000	02	000
-002x	02	000	02	000	02	000	02	000	02	000	02	000	02	000	02	000
+003x	03	000	03	000	03	000	03	000	03	000	03	000	03	000	03	000
-003x	03	000	03	000	03	000	03	000	03	000	03	000	03	000	03	000
+004x	35	400	04	001	55	003	04	005	04	000	04	001	55	004	04	005
-004x	35	400	04	001	04	006	04	005	04	000	04	001	23	000	04	005
+005x	05	000	05	001	05	002	05	003	05	004	05	005	05	006	05	007
-005x	05	020	05	021	05	022	05	023	05	024	05	025	05	026	05	027
+006x	06	006	06	003	06	002	06	003	06	002	06	002	06	002	06	002
-006x	06	004	06	001	06	000	06	001	06	000	06	000	06	000	06	000
+007x	07	010	07	010	07	010	07	010	07	010	07	010	07	010	07	010
-007x	07	010	07	010	07	010	07	010	07	010	07	010	07	010	07	010
+010x	10	000	10	000	10	000	10	000	10	000	10	000	10	000	10	000
-010x	10	000	10	000	10	000	10	000	10	000	10	000	10	000	10	000
+011x	11	020	11	020	11	020	11	020	11	020	11	020	11	020	11	020
-011x	11	020	11	020	11	020	11	020	15	030	15	030	15	030	15	030
+012x	10	000	12	000	10	000	12	000	10	000	12	000	10	000	12	000
-012x	10	000	12	000	10	000	12	000	10	000	12	000	10	000	12	000
+013x	13	400	13	400	13	400	13	400	13	400	13	400	13	400	13	400
-013x	13	000	13	000	13	000	13	000	13	000	13	000	13	000	13	000
+014x	10	000	10	000	10	000	10	000	10	000	10	000	10	000	10	000
-014x	10	000	10	000	10	000	10	000	10	000	10	000	10	000	10	000
+015x	15	020	15	020	15	020	15	020	15	020	15	020	15	020	15	020
-015x	15	020	15	020	15	020	15	020	15	020	15	020	15	020	15	020
+016x	10	000	12	000	10	000	12	000	10	000	12	000	10	000	12	000
-016x	10	000	12	000	10	000	12	000	10	000	12	000	10	000	12	000
+017x	17	020	17	020	17	020	17	020	17	020	17	020	17	020	17	020
-017x	17	020	17	020	17	020	17	020	17	020	17	020	17	020	17	020

Table 12

Comprehensive Translator - ±0200 to ±0377

OP CODE	xxx0		xxx1		xxx2		xxx3		xxx4		xxx5		xxx6		xxx7	
+020x	20	420	20	420	20	420	20	420	20	430	20	430	20	430	20	430
-020x	20	421	20	421	20	421	20	421	20	431	20	431	20	431	20	431
+021x	21	030	21	030	21	030	21	032	21	030	21	031	21	030	21	034
-021x	21	030	21	030	21	030	21	032	21	030	21	031	21	030	21	034
+022x	22	420	22	421	22	420	22	421	22	430	22	431	22	430	22	431
-022x	22	420	22	421	22	420	22	421	22	430	22	431	22	430	22	431
+023x	23	000	23	000	23	000	23	000	23	000	23	000	23	000	23	000
-023x	23	000	23	000	23	000	23	000	23	000	23	000	23	000	23	000
+024x	24	410	24	400	24	410	24	400	24	410	24	400	24	410	24	400
-024x	25	410	25	400	24	410	24	400	24	410	24	400	24	410	24	400
+025x	25	410	25	400	25	410	25	400	25	410	25	400	25	410	25	400
-025x	25	410	25	400	25	410	25	400	25	410	25	400	25	410	25	400
+026x	26	400	27	400	26	400	26	400	26	400	26	400	26	400	26	400
-026x	26	404	27	404	26	404	26	404	26	404	26	404	26	404	26	404
+027x	27	400	27	400	27	400	27	400	27	400	27	400	27	400	27	400
-027x	27	404	27	404	27	404	27	404	27	404	27	404	27	404	27	404
+030x	30	400	31	400	30	500	31	500	30	600	31	600	30	700	31	700
-030x	30	401	31	401	30	501	31	501	30	601	31	601	30	701	31	701
+031x	31	400	31	400	31	500	31	500	31	600	31	600	31	700	31	700
-031x	31	401	31	401	31	501	31	501	31	601	31	601	31	701	31	701
+032x	61	004	32	015	32	001	32	011	32	004	32	015	32	001	32	011
-032x	32	004	32	015	32	001	32	011	32	004	32	015	32	001	32	011
+033x	33	000	33	000	33	000	33	000	33	000	33	000	33	000	33	000
-033x	33	000	33	000	33	000	33	000	33	000	33	000	33	000	33	000
+034x	34	400	34	400	34	400	34	400	34	400	34	400	34	400	34	400
-034x	34	000	34	000	34	000	34	000	34	000	34	000	34	000	34	000
+035x	35	400	35	400	35	400	35	400	35	400	35	400	35	400	35	400
-035x	35	400	35	400	35	400	35	400	35	400	35	400	35	400	35	400
+036x	36	000	36	000	36	000	36	000	36	000	36	000	36	000	36	000
-036x	36	000	36	000	36	000	36	000	36	000	36	000	36	000	36	000
+037x	37	010	37	010	37	010	37	010	37	010	37	010	37	010	37	010
-037x	37	000	37	000	37	000	37	000	37	000	37	000	37	000	37	000

Table 12

Comprehensive Translator - ±0400 to ±0577

OP CODE	xxx0		xxx1		xxx2		xxx3		xxx4		xxx5		xxx6		xxx7	
+040x	40	400	40	200	40	500	40	700	40	400	40	200	40	500	40	700
-040x	40	300	40	300	40	700	40	700	40	300	40	300	40	700	40	700
+041x	41	000	41	000	41	000	41	000	41	000	41	000	41	000	41	000
-041x	41	000	41	000	41	000	41	000	41	000	41	000	41	000	41	000
+042x	42	000	42	000	42	000	42	000	42	000	42	000	42	000	42	000
-042x	42	000	42	000	42	000	42	000	42	000	42	000	42	000	42	000
+043x	43	000	43	000	43	000	43	000	43	000	43	000	43	000	43	000
-043x	43	000	43	000	43	000	43	000	43	000	43	000	43	000	43	000
+044x	44	001	44	002	44	005	45	400	44	014	44	003	44	016	44	007
-044x	44	001	44	002	44	005	45	400	44	014	44	003	44	016	44	007
+045x	45	400	45	400	45	400	45	400	45	400	45	400	45	400	45	400
-045x	45	400	45	400	45	400	45	400	45	400	45	400	45	400	45	400
+046x	46	000	46	000	46	000	46	000	46	000	46	000	46	000	46	000
-046x	46	000	46	000	46	000	46	000	46	000	46	000	46	000	46	000
+047x	47	000	47	000	47	000	47	000	47	000	47	000	47	000	47	000
-047x	47	000	47	000	47	000	47	000	47	000	47	000	47	000	47	000
+050x	50	400	32	015	50	500	50	500	50	400	50	400	50	500	50	500
-050x	50	000	32	015	50	100	50	100	50	000	50	000	50	100	50	100
+051x	51	000	51	000	51	000	51	000	51	000	51	000	51	000	51	000
-051x	51	000	51	000	51	000	51	000	51	000	51	000	51	000	51	000
+052x	52	400	52	400	72	000	52	400	52	400	52	400	52	000	52	000
-052x	52	401	52	401	72	000	52	401	52	401	52	401	52	401	52	401
+053x	53	010	53	000	53	010	53	000	53	010	53	000	53	010	53	000
-053x	53	030	53	020	53	030	53	020	53	030	53	020	53	030	53	020
+054x	54	006	54	002	52	002	54	002	54	004	54	000	54	000	54	000
-054x	54	003	54	002	54	002	54	002	54	001	54	000	54	000	54	000
+055x	55	003	55	003	55	003	55	003	55	004	55	004	55	004	55	004
-055x	55	003	55	003	55	003	55	003	55	004	55	004	55	004	55	004
+056x	56	000	56	000	56	000	56	000	57	000	56	000	56	000	56	000
-056x	56	000	56	000	56	000	56	000	57	000	56	000	56	000	56	000
+057x	57	000	57	000	57	000	57	000	57	000	57	000	57	000	57	000
-057x	57	000	57	000	57	000	57	000	57	000	57	000	57	000	57	000

Table 12

Comprehensive Translator - +0600 to +0777

OP CODE	xxx0		xxx1		xxx2		xxx3		xxx4		xxx5		xxx6		xxx7	
+060x	71	000	60	410	60	010	70	400	60	000	60	400	60	000	60	400
-060x	71	010	60	410	61	005	70	400	60	000	60	400	60	000	60	400
+061x	61	004	61	004	61	004	61	004	61	004	61	004	61	004	61	004
-061x	61	005	61	005	61	005	61	005	61	005	61	005	61	005	61	005
+062x	65	000	74	000	47	000	66	010	66	010	67	000	66	010	66	010
-062x	65	000	74	000	47	000	66	000	66	000	46	000	66	000	66	000
+063x	62	000	63	010	63	000	63	000	63	010	63	010	63	000	63	000
-063x	62	000	63	030	63	020	63	020	63	030	63	030	63	020	63	020
+064x	64	004	64	000	64	000	64	000	64	004	64	000	64	000	64	000
-064x	64	001	64	000	65	000	64	000	64	000	64	000	64	000	64	000
+065x	65	000	65	000	65	000	65	000	65	000	65	000	65	000	65	000
-065x	65	000	65	000	65	000	65	000	65	000	65	000	65	000	65	000
+066x	66	010	66	010	66	010	66	010	66	010	66	010	66	010	66	010
-066x	66	000	66	000	67	000	66	000	66	000	66	000	66	000	66	000
+067x	67	000	67	000	67	000	67	000	67	000	67	000	67	000	67	000
-067x	67	000	67	000	67	000	67	000	67	000	67	000	67	000	67	000
+070x	70	400	70	400	70	400	70	400	70	400	70	400	70	400	70	400
-070x	70	400	70	400	70	400	70	400	70	400	70	400	70	400	70	400
+071x	71	000	71	000	71	000	71	000	71	000	71	000	71	000	71	000
-071x	71	010	71	010	71	010	71	010	71	010	71	010	71	010	71	010
+072x	72	000	72	000	72	000	72	000	72	000	72	000	72	000	72	000
-072x	72	000	72	000	72	000	72	000	72	000	72	000	72	000	72	000
+073x	73	010	73	010	73	000	73	000	73	010	73	010	73	000	73	000
-073x	73	030	73	030	73	020	73	020	73	030	73	030	73	020	73	020
+074x	74	000	74	000	74	000	74	000	74	000	74	000	74	000	74	000
-074x	74	000	74	000	74	000	74	000	74	000	74	000	74	000	74	000
+075x	75	010	75	010	75	000	75	000	75	010	75	010	75	000	75	000
-075x	75	030	75	030	75	020	75	020	75	030	75	030	75	020	75	020
+076x	66	010	77	010	76	001	77	405	76	000	77	404	76	005	77	003
-076x	66	000	77	012	76	001	77	017	76	000	77	016	76	005	77	013
+077x	76	000	77	012	76	000	77	403	37	010	77	406	76	004	77	003
-077x	76	000	77	010	76	000	77	011	37	000	77	014	76	004	77	013

NOTES

SPECIAL CONSIDERATIONS OF THE TRANSLATOR

1. Table 13 is a list of groups entered by op-codes which do not have the group numbers as the two center digits in its op-code. Unless otherwise specified, precondition controls are determined by the group number and bits 0 and bits 9-11 of the op-code.

Table 13. Translator Exceptions

GROUP	OP CODE	GROUP	OP CODE
01	±2xxx,±3xxx	41	LEVEL 3 REQUEST
10	±0120,±0122,±0124,	43	LEVEL 2 REQUEST
	±0126,±014x,±0160,	45	±0443
	±0162,±0164,±0166	46	-0625
12	±0161,±0163,±0165	47	±0622
	±0167	51	LEVEL 1 REQUEST
14	Can't Be Entered	55	+0042,+0046
15	-0114,-0115,-0116	57	±0564
	-0117	61	+0320,-0602
16	Can't Be Entered	62	±0630
17	+1xxx	65	±0620,-0642
21	-1xxx	66	±0623,±0624,±0626,
23	-0046		+0627,±0760
25	-0240,-0241	67	+0625,-0662
27	±0261	70	±0603
31	±0301,±0303,±0305	71	±0600
	±0307	72	±0522
32	±0501	74	±0621
33	LEVEL 4 TRAP REQUEST	76	±0770,±0772,±0776
35	±0040	77	±0771,±0773,±0775,
37	±0774		±0777

Note the following:

GROUPS 14 and 16 may not be reached by the TRANSLATOR. Control and transfer vectors for groups 33, 41, 43 and 51 are also used when the scheduler honors requests other than the program request.

2. In GROUP 01 GIN is set on if bit 2 of the op-code is a one (i.e., the threes); and GOP 11 is set on if bit 0 of the op-code is a one (i.e., the minus op-codes).

3. In GROUP 07, GIN is not set on if AUX 1 is on (i.e., the instruction address is taken from AUX 1 instead of from the IC).

4. In GROUP 21 GIN is not set on if AUX 1 is on (i.e., the instruction address is taken from AUX 1 instead of from the IC). Also in GROUP 21 GOP 11 is not set on if Bit 7 of the op-code is a one.

5. GROUPS 02, 03, and 10 will set the TSAT flip-flop under appropriate conditions. This allows fast transfers if LK7 is on in the control and transfer vectors and the transfer trap mode control flip-flop is off.

   When a trap transfer (op-code +0021) is translated then TSAT is set and the transfer trap mode control flip-flop doesn't affect the transfer condition. Table 14 shows the translator conditions that set the TSAT flip-flop.

Table 14. Translator TSAT Control

GROUP	NOTE	CODE	SPECIAL CONDITIONS
2		±0021	Unconditional (trap transfer)
2		±0020	Unconditional (normal transfer)
2	6	+0022	CHA Tape Check (Gen. Ind. SOP44) on and Tape Check Enable (Gen. Ind. SOP32) off.
2	6	-0022	CHB Tape Check (Gen. Ind. SOP46) on and Tape Check Enable (Gen. Ind. SOP22) off.
3	7	+003x	CHA EOF (Gen. Ind. SOP06) on and Command Trap Enable (Gen. Ind. SOP34) off.
3	7	-003x	CHB EOF (Gen. Ind. SOP14) on and Command Trap Enable (Gen. Ind. SOP24) off.
10		±01x2	MQ Bit 0 off
10		+010x	AC is zero (Q,0-35)
10		-010x	AC is not zero (Q,0-35)
10		+012x	AC sign bit off
10		-012x	AC sign bit on
10	8	+014x	AC overflow is on
10	8	-014x	AC overflow is off

6. If the Tape Check Enable Indicator for the channel is off, the Tape Check Indicator for the channel is turned off.

7. If the Command Trap Enable Indicator for the channel is off, the End of File Indicator for the channel is turned off.

8. The AC Overflow Indicator is turned off.

SIGN AND MOST SIGNIFICANT BIT CONTROL

At the start of MINIFLOW, the Engine signs and most significant bits are set as shown below.

Register	Contents	Register Sign	Engine Q Bit
B	Accumulator	AC Sign Bit	AC Q Bit
C	Object Instruction or Memory Operand		
D	Object Instruction		

If a memory operand was fetched into C during wired-in-sequence, the sign and most significant bit of register C are defined as shown below:

OPERAND	SIGN AND PRECONDITIONS:			C REGISTER	
BIT 0	ARI	MAG	SUB	SIGN	MSB
0	X	0	0	+	0
1	0	0	0	+	1
1	1	0	0	-	0
0	X	0	1	-	0
1	0	0	1	+	1
1	1	0	1	+	0
X	X	1	0	+	0
X	X	1	1	-	0

The "Engine Sign" may determine what is moved as a sign bit or as a most significant bit. The Engine Sign is defined as follows:

D Sign If the D SOP (20) is used.
C Sign If the C SOP (31) is used.
B Sign If any of the following SOPS are used:

DOL (01), LDB (04), B-C-1 (06), B-1 (14), B-C (16), LDD (21), NB (22), B·C (24), B+C+I (26), B (30), B+1 (34), B+C (36)

Zero if any of the following SOPS are used:

ZERO (00), NB+1 (02), NC+1 (03), LDC (05), C-B-I (07), B·NC (10), NB·C (11), BEC (12), C-1 (15), C-B (17), NC (23), BUC (32), C+1 (35).

If the Engine Sign is the B or the C sign, it may have a hard register sign combined with it. Specifically, the MSB of the Hard Register is logically ORed with the B sign (or the C sign) if an MQ, SI, or R4 POP is used and ARI is on and the B replace (or C replace) bit is on.

The "Engine Bit 0" is the result of the engine operation at bit position 0. The manner in which the B, C, and D register Signs and most significant bits may be altered by MINIFLOW are defined below. X signifies the particular Engine Register.

CONDITIONS:	RESULTS:	
OPERATION	X SIGN	X MSB
AC with LDX (as TOP)	Engine Sign	Engine Bit 0
MQ, SI, R4 with LDX and ARI off	Engine Sign	Engine Bit 0
MQ, SI, R4 with LDX and ARI on	Engine Sign	Engine Sign
AC, MQ, SI, R4 with no load TOP and with Replace type SOP B or C	Engine Sign	Unchanged
CMI with LDX (SOP)	Unchanged	Memory MSB
MEM or MKEY-Read with ARI off	Unchanged	Memory MSB
MEM or MKEY-Read with ARI on	Memory MSB	Zero

In addition, the B and D signs may be altered as follows:

OPERATION	RESULTS B SIGN	D SIGN
SHIFT MULT	B⊕C SIGN*	B⊕C SIGN*
SHIFT DIV	Unchanged	B⊕C SIGN*
SHIFT DOS WITH ARI ON and GOP11 ON	D Sign	Unchanged
SHIFT DOS WITH ARI ON and GOP11 OFF	Unchanged	B Sign
ALG with unlike signs in B & C and first carry true. (With AC Q Bit off or Q Bit not zoned in).	C Sign	Unchanged

*B⊕C SIGN is the EXCLUSIVE OR of the B sign bit and the C sign bit. This is the same as the algebraic sign resulting from a multiply or a divide operation.

The conditions and manner in which the AC and other Hard Register signs and memory signs may be altered by MINIFLOW are defined below:

AC SIGN AND P BIT

OPERATION	RESULTS AC SIGN	AC P BIT
MISC RSQ	Plus	Unchanged
MISC RAS	Plus	Unchanged
MISC TAS	Complemented	Unchanged
MISC SAS	Minus	Unchanged
AC with store	Engine Sign	Engine Bit 0

MQ, SI, R4 Most Significant Bit

	ARI ON	ARI OFF
MA, SI, R4 with store	Engine Sign	Engine Bit 0

MEMORY Most Significant Bit

MEM WITH RM BITS:

14	16	17		ARI ON	ARI OFF
1	0	0	(Store Zero)	Zero	Zero
1	0	1	(Store B)	B Sign	B Bit 0
1	1	0	(Store C)	C Sign	C Bit 0
1	1	1	(Store D)	D Sign	D Bit 0

Q BIT AND OVERFLOW CONTROL

The Engine Q bit equals the AC Q bit at the start of MINIFLOW YA6. The conditions and manner in which the AC Q bit or the Engine Q bit may be altered during MINIFLOW are defined below:

CONDITIONS OPERATION	AC Q BIT	ENGINE Q BIT
MISC RSQ	Zero	Zero
MISC RAQ	Zero	Zero
MISC TAQ	Complement AC Q bit	Complement AC Q bit
MISC SAQ	One	Unchanged
SHIFT MULT	Unchanged	Left extension of B reg.
SHIFT DOS	Unchanged	Left extension of B reg.
PB,PC PBD,PCD POPS with Q bit zone (33 or 05)	Unchanged	Generated Q bit

AC POP WITH RM BITS				
15	16	17		
x	0	0	----------	Unchanged
x	0	1	----------	Generated Q bit
x	1	0	----------	Generated Q bit
x	1	1	----------	Generated Q bit
0	x	x	Unchanged	--------------
1	x	x	Generated Q bit	--------------

The Generated Q bit equals the exclusive OR of the carry out of bit position 0 and the present ENGINE Q BIT**; except the Generated Q bit is zero for the following SOPS: ZERO, DOL, NB·C, B·C, C, C-1, C+1, D, and LDB, LDC, LDD.

The Overflow indicator is turned off by a console Reset or by the translators while translating an OP-CODE of ±014x or by MISC ROV (063000).

The Overflow indicator is turned on by MINIFLOW under any of the four following conditions:

1. An AC POP with Store bit on and a carry out of bit position 1.

2. An ALG POP with Q bit zoning, like signs, and a carry out of bit position 1.

3. A SHIFT MULT or SHIFT DOS with a one bit shifted left out of bit position 1.

4. A MISC SOV (063100)

**The engine Q bit is replaced by the AC Q bit on the AC POP if the replace B or Replace C bit is on.

INPUT/OUTPUT CONTROL

Channel Description

Peripheral devices are controlled by two independent channels, A and B, under MINIFLOW supervision. Channel A may have tapes, a card reader and a typewriter attached to it. Channel B may have tapes attached to it. The channels transmit data between the peripheral device and control core. An I/0 operation may be initiated in any level.

For read operations, Channel A will generate a level 1 request each time 8 words have been moved from the device to control memory. Channel B will generate a level 2 request each time 8 words have been moved to control memory.

For write operations, Channel A will generate a level 1 request each time 8 words have been moved from control memory to the device. Channel B will generate a level 2 request each time 8 words are moved from control core to the device.

Both Channels A and B will set a level 3 request on termination of activity. Activity is terminated in the following ways:

Read - Channel detects end of record on media.

Write- Channel is directed to generate an end of record on the media.

Non Data Operation
Backspace - Backspace is completed
Write EOF - Operation is completed
Rewind or Unload - Operation is started.

The channels use a fixed buffer region in control memory, octal word locations 200 to 237 for Channel A and 240 to 277 for Channel B. This gives 32 words of buffering for each channel. The channel can therefore stack 4 buffer requests (level 1 and 2). The buffer request is serviced by a MINIFLOW routine which may move the data from the buffer. This routine must acknowledge the request.

Channel operations are controlled by three control registers for each channel, as shown below:

a. Channel Register 1 receives the channel instruction when a channel activity is initiated by a CH1 POP, described under MINIFLOW instruction descriptions.

b Channel Register 2 controls the channel termination. It is accessed by the CH2 POP to either stop the channel data transfer or to locate the last word of a record or the first word in error.

c. Mode control is a set of 4 flip-flops used to control channel sequence and timing. The mode control are loaded into the main engine by a CH1 POP with a load SOP There are ten active states A to J, plus an idle state.

The peripherals and their interaction with the IC-M9 through the channels are described next.

THE CARD READER is a unit record device which is selected by placing the following bit pattern into Channel Register 1.

0	1 7	8 10	11 27	28 30	31	32 35
0	xxxxxxx	001	xxxxxxxxxxxxxxxxx	110	1	xxxxxxxxx

Notice that the object instruction +076200001321 which is used to read cards from the IBM 7094 conforms to this pattern.

The card is read endwise from column 1 to 80 in full binary fashion. Columns 1, 2, and 3 form the first word stored in the buffer at octal word location 200. Every succeeding three columns form a new word at the next higher address. The columns fill the word from left to right with the twelfth row most significant. Holes are read as ones. Twenty-seven full words are read into control core with column 81 read as zeros.

When the data is all moved, a level 1 buffer service request is set in the scheduler. When the trailing edge of the card is sensed, a channel terminate request is set in the channel and propogates into the scheduler. This terminate request must be reset in level 3 by MINIFLOW with a MISC RTER once the request is granted or it will cause a level 3 interrupt again.

A procedure to read a card is as follows:

a. Issue the CH1 to select the card reader and EXIT.

b. When the level 1 interrupt occurs, move and check the data as desired. This may include check sum tests, corner turning or binary to BCD code conversion, look ahead buffering, and moving to main storage under I/O command control.

c. When the level 3 (terminate) interrupt occurs, reset the terminate request with a MISC RTER and perhaps reissue a select if command chaining or other conditions require it.

Pressing the LOAD MACHINE key on the maintenance panel or placing a 10 (bits 0, 8-10) into the Channel Register 1 will also cause a card to be read identically except that the data will be read into octal word locations 100 to 132. This is the first part of the MINIFLOW entry table, which consists of mini-locations 200 to 265, and is the means provided to read and start execution of a bootstrap loader. The bootstrap loader is explained at the end of Section III.

THE CONSOLE TYPEWRITER is a unit record device also, and it will type when the following bit pattern is loaded into Channel A Register 1:

0	1 7	8 10	11 27	28 30	31	32 35
0	xxxxxxxx	011	xxxxxxxxxxxxxxxxx	111	1	xxxxxxxxxxxxx

Notice that the IBM 7094 instruction to select the printer conforms to the bit pattern used to select the typewriter. If bit zero is a one, i.e., -076600001361, then the typewriter will space vertically one line.

The procedure to TYPEWRITE A LINE is as follows:

a. Move the data to the Channel A buffer at octal word location 200 in control memory. This operation may include moving under I/O command control, corner turning and/or code conversion (refer to Table 15) and generating printer or time echos if simulating the 407 printer.

b. Issue the CH1 instruction to Channel A Register 1 as defined by the previously defined bit pattern. A level 1 buffer service request will not be set in the scheduler when a CH1 is directed to the typewriter.

c. Issue a CH2 instruction to Channel A Register 2 to give the channel the number of words to be typed. This is specified by the low order 5 bits of the data generated by the CH2 instruction. This count should be a value between 1 and octal 37 which gives a character count between 6 and 186. A speedup technique to use when simulating a printer on the typewriter is to examine words from right to left for blanks and to decrease the word count in the CH2 to eliminate typing out trailing blanks. The exit bit may be used in the CH2 instruction to exit to the scheduler.

d. A terminate request is set in the channel logic and the scheduler when the channel disconnects from the typewriter. This terminate request must be reset in the channel with a MISC RTER instruction.

Table 15. Octal Code vs. Typewriter Character

OCTAL CODE	X0	X1	X2	X3	X4	X5	X6	X7
0X	.	I	ǂ	[	F	D	B	?
1X	+	H	<	)	G	E	C	A
2X	$	R	Δ	]	O	M	K	!
3X	-	Q	;	*	P	N	L	J
4X	,	Z	⧻	m	W	U	S	ǂ
5X	b̸	Y	\	(	X	V	T	/
6X	=	9	√	:	6	4	2	Ø
7X	blank	8	>	'	7	5	3	1

THE MAGNETIC TAPE UNIT are physical record devices which are attached to either Channel A or Channel B. They are selected by placing the following pattern into Channel A (or B) Register 1 with a CH1 POP:

0	1 7	8 10	11 28	29	30	31	32 35
C	xxxxxxxxx	CCC	xxxxxxxxxxxxxxxxx	0	X	M	u u u u

C indicates the operation to be performed by the channel, as shown below:

C (Octal)	Operations
01	Read Tape
03	Write Tape
04	Write End-of-File Mark
13	Write Blank Tape
02	Backspace Record
12	Backspace File
05	Rewind Tape
15	Rewind Tape and Unload
07	Test Tape for Unit Present

Setting bit 29 to a zero causes tape to be selected. Bit 31 (M) indicates the tape mode as follows:

0	BCD Mode (even parity)
1	Binary Mode (odd parity)

Bits 32-35 (u) indicate which tape unit is to be selected. It is a binary coded number from one (0001) to ten (1010).

The specific channel selected by a CH1 or CH2 POP is determined by the CHB flip-flop. Channel A is selected if the CHB flip-flop is off and Channel B is selected if the CHB flip-flop is on. The specific buffer area in control memory is also specified by the CHB flip-flop. If CHB flip-flop is off, octal word locations 200-237 are used and if it is on, locations 240-277 are used.

If the tape operation does transmit data then the specific channel buffer is partitioned into four, 8 word, sub buffers. Data transmission always starting with the first buffer word. i.e., 200 for Channel A, 240 for Channel B. Whenever a new sub buffer is accessed, i.e., the channel moves 8 more words, a request counter in the channel is incremented. If this counter is not zero, a buffer service request is set in the scheduler. On granting this request from the scheduler, the buffer service MINIFLOW routine (executed in level 1 for Channel A and level 2 for Channel B) must acknowledge the channel request which caused the level 1 (or 2) interrupt. This is done with a MISC ACK instruction which will decrement the channel request counter. The MINIFLOW buffer service routine must move the data fast enough so that, on the average, it will stay ahead of the channel data movement. At the time the last buffer word is moved it must start again in order to move the first word of the buffer.

Similarly when the channel accesses the last word of the buffer (237 for Channel A, 277 for B) then channel addressing wraps around and starts back at the first buffer word. If the channel data movements starts to get more than 32 words ahead of MINIFLOW a transfer timing error will occur because data is lost while reading or repeated while writing. This condition is detected when the channel request counter attempts to count above 4, meaning that the channel is more than 4 sub buffers ahead of the servicing routine. However, when this occurs, all 32 words in the buffer will still be good. When a transfer timing error occurs the channel indicator ER2 will be turned on. The channel will be disconnected and a terminate request will be set in the channel and sent to the scheduler. Channel indicator ER1 is turned on if a parity or other data error is detected. A procedure to read or write tape is shown in Figure 9.

It should be emphasized that the flow chart in Figure 9 is only one example and is not intended to emulate the tape I/O control for any particular machine. The example assumes that commands are not chained and that a parity error will disconnect the channel and turn on a tape redundancy indicator. Many different options are available by proper MINIFLOW control, such as command chaining, special trapping, continuing to read on an error, synchronizing the CPU and channels and transfer timing error recovery procedures. The transfer timing recovery might include backspacing a record and reading without data transmission until the previously lost data is read again and then continuing with data transmission. If channel synchronizing is included, the data moving routines must be executed in levels 1 and 2 so that a delay may be given to allow the synchronizing instruction to be executed in level 4.

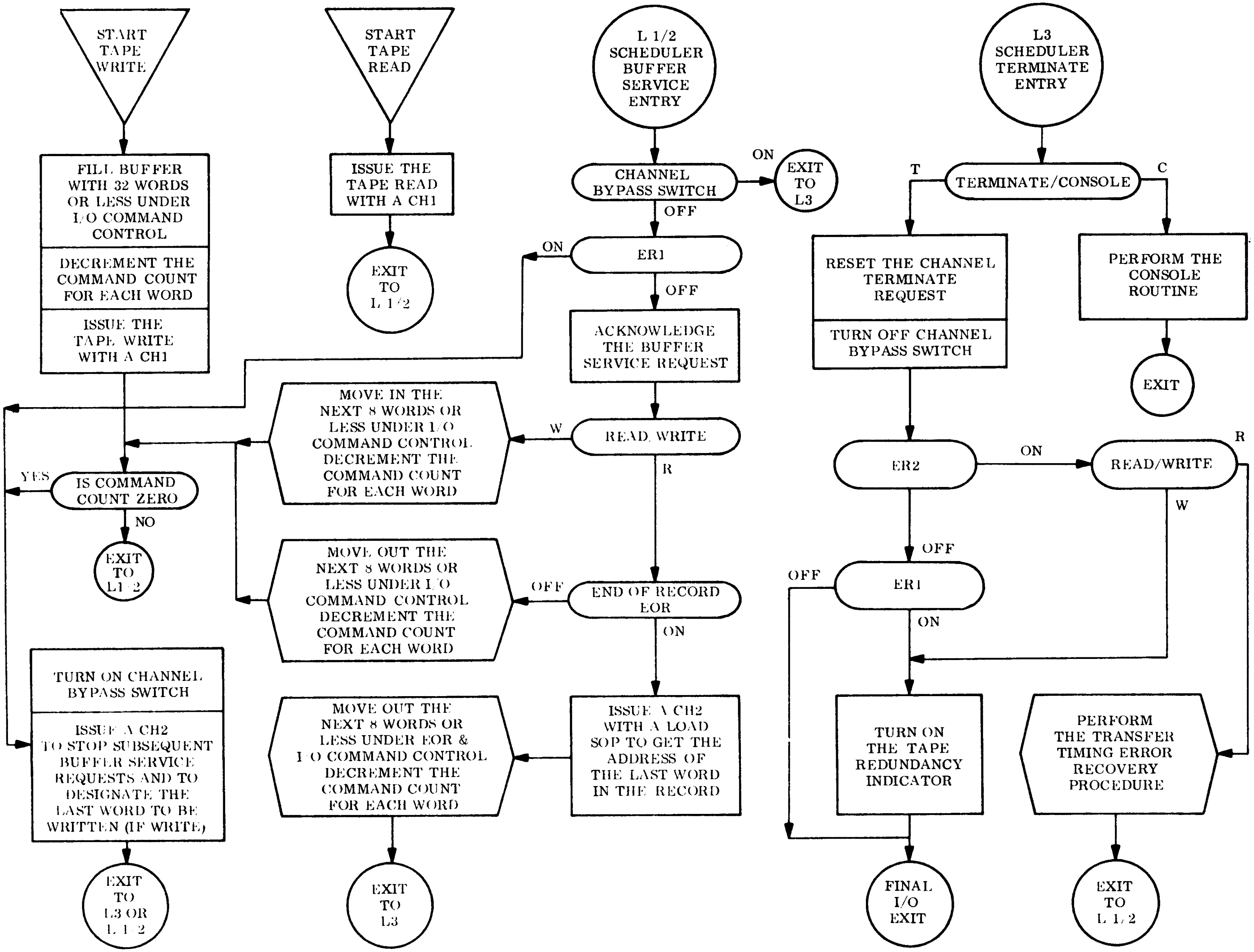

START TAPE WRITE
FILL BUFFER WITH 32 WORDS OR LESS UNDER I/O COMMAND CONTROL
DECREMENT THE COMMAND COUNT FOR EACH WORD
ISSUE THE TAPE WRITE WITH A CH1
START TAPE READ
ISSUE THE TAPE READ WITH A CH1
EXIT TO L 1/2
L 1/2 SCHEDULER BUFFER SERVICE ENTRY
CHANNEL BYPASS SWITCH
ON
EXIT TO L3
OFF
ER1
ON
OFF
ACKNOWLEDGE THE BUFFER SERVICE REQUEST
READ/WRITE
W
R
MOVE IN THE NEXT 8 WORDS OR LESS UNDER I/O COMMAND CONTROL DECREMENT THE COMMAND COUNT FOR EACH WORD
YES
IS COMMAND COUNT ZERO
NO
EXIT TO L1/2
MOVE OUT THE NEXT 8 WORDS OR LESS UNDER I/O COMMAND CONTROL DECREMENT THE COMMAND COUNT FOR EACH WORD
OFF
END OF RECORD EOR
ON
TURN ON CHANNEL BYPASS SWITCH
ISSUE A CH2 TO STOP SUBSEQUENT BUFFER SERVICE REQUESTS AND TO DESIGNATE THE LAST WORD TO BE WRITTEN (IF WRITE)
EXIT TO L3 OR L 1/2
MOVE OUT THE NEXT 8 WORDS OR LESS UNDER EOR & I/O COMMAND CONTROL DECREMENT THE COMMAND COUNT FOR EACH WORD
EXIT TO L3
ISSUE A CH2 WITH A LOAD SOP TO GET THE ADDRESS OF THE LAST WORD IN THE RECORD
L3 SCHEDULER TERMINATE ENTRY
T
TERMINATE/CONSOLE
C
RESET THE CHANNEL TERMINATE REQUEST
TURN OFF CHANNEL BYPASS SWITCH
PERFORM THE CONSOLE ROUTINE
EXIT
ER2
ON
READ/WRITE
R
W
OFF
OFF
ER1
ON
TURN ON THE TAPE REDUNDANCY INDICATOR
PERFORM THE TRANSFER TIMING ERROR RECOVERY PROCEDURE
FINAL I/O EXIT
EXIT TO L 1/2

Figure 9. Read or Write Tape Flow Chart

REMOTE EXECUTION AND TRAPPING

Remote execution is a means by which the machine may temporarily interrupt its normal program sequence to execute an extra instruction and then return to the normal program sequence (unless the extra instruction alters the IC, such as a skip or a transfer).

Trapping is a means by which the object program may be made aware of special machine conditions which need to be monitored without requiring frequent test instructions. The machine status is constantly monitored and when one of the special conditions occur, the machine will "TRAP". When the machine traps, the instruction counter (IC) is stored, along with trap identification data, into a location determined by the type of trap condition. The program then goes to another location (also determined by the type of trap) for its next instruction. This next instruction starts the trap analysis which determines the action to take. It may subsequently return back to the main program.

Several of the special situations which may be monitored with traps are:

a. Privileged Instruction - When in a non-privileged mode.

b. Transfers - When in transfer trap mode.

c. I/O Termination - Both data and non-data operations.

d. Timer - Service and overflow.

e. Exceptions - Floating point, divide check.

f. Self trapping - Such as a Store and Trap.

g. Attention - For operator or external control.

h. Illegal Instruction - Such as illegal op codes, illegal addresses, illegal data, and illegal instruction operands.

It should be kept in mind that many of these situations may not occur on some object machines or they may be handled in a different manner than by trapping. The four methods used in the IC-M9 to handle trapping or remote execution are:

1. The normal MINIFLOW

2. The transfer trap mode control

3. The channel trap request logic

4. The auxiliary instruction counters

NORMAL MINIFLOW handles trapping situations which occur in level 4 program (object instruction) execution which may be monitored by MINIFLOW and will cause a direct sequence change. Some examples would be floating point exceptions, self trapping, privileged instructions or illegal op codes.

TRANSFER TRAP MODE CONTROL allows object program transfer instructions to be trapped without monitoring them in MINIFLOW or slowing down their emulation when not trapping them. This form of monitoring is enabled when the General Indicator TRAP (SOP40) is on. Transfer trapping is performed during the W-I-S state YA5 just before entering MINFLOW. Refer to Figure 10 for details of transfer trapping. LK7 identifies a fast transfer which may be trapped. Special logic excludes the op code ±0021 from being trapped. If LK4 is on for a slow transfer it is assumed that the transfer will be trapped later when it is processed as a fast transfer. Figure 10 illustrates the logic involved in the W-I-S state YA5 and in MINIFLOW (W-I-S state YA6) to handle transfer traps.

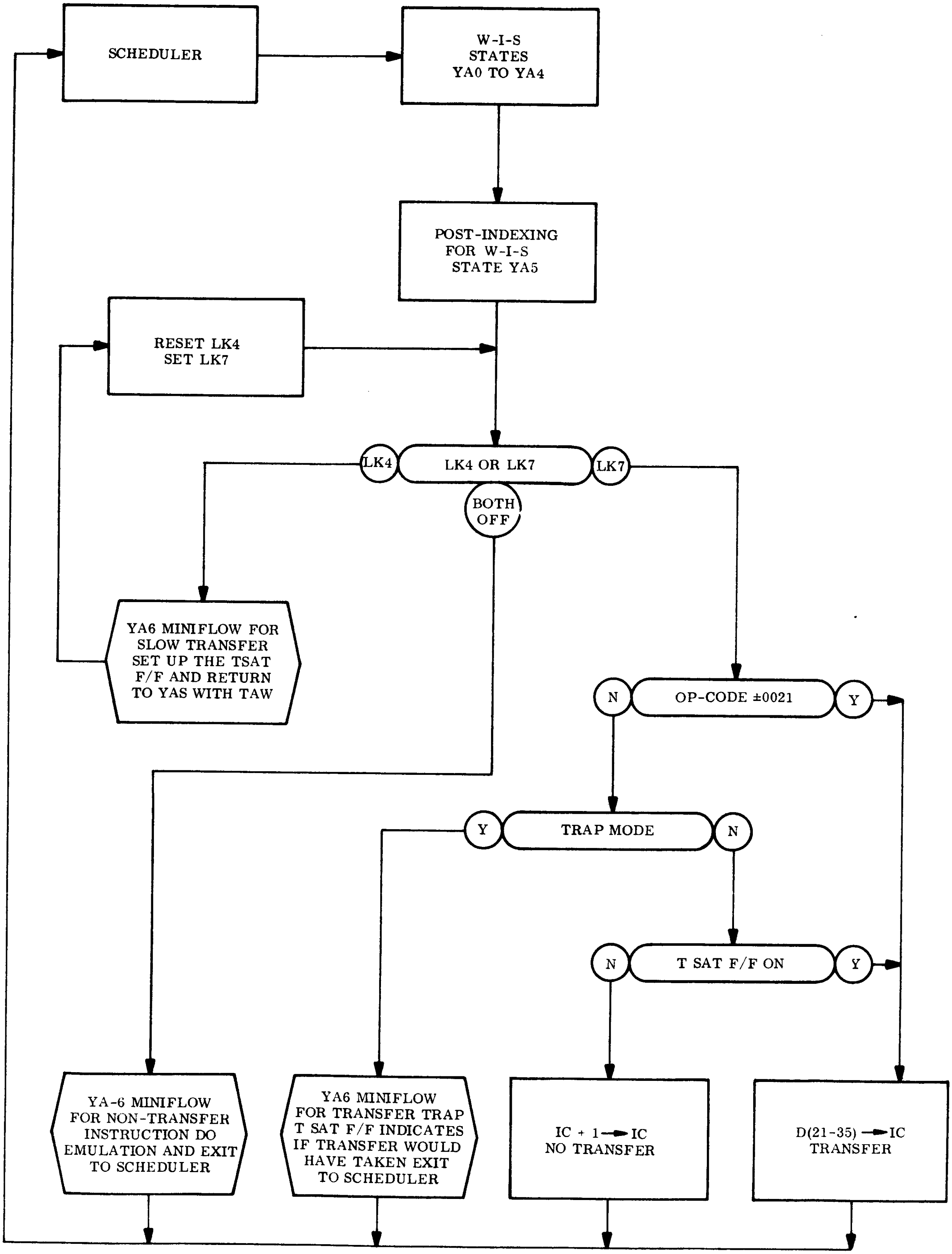

SCHEDULER
W-I-S STATES YA0 TO YA4
POST-INDEXING FOR W-I-S STATE YA5
RESET LK4 SET LK7
LK4
LK4 OR LK7
LK7
BOTH OFF
YA6 MINIFLOW FOR SLOW TRANSFER SET UP THE TSAT F/F AND RETURN TO YAS WITH TAW
N
OP-CODE ±0021
Y
Y
TRAP MODE
N
N
T SAT F/F ON
Y
YA-6 MINIFLOW FOR NON-TRANSFER INSTRUCTION DO EMULATION AND EXIT TO SCHEDULER
YA6 MINIFLOW FOR TRANSFER TRAP T SAT F/F INDICATES IF TRANSFER WOULD HAVE TAKEN EXIT TO SCHEDULER
IC + 1 → IC NO TRANSFER
D(21-35) → IC TRANSFER

Figure 10. Flow Chart of Transfer Trapping in YA5

NOTE

In both examples 1 and 2 in Section II, if the transfer trap mode control is on then the IC is always incremented by 1 and MINIFLOW is entered at the point specified by the control and transfer vector, bits 7-17.

CHANNEL TRAP REQUEST LOGIC is used to provide trapping for special conditions detected while not executing level 4 program MINIFLOW. Some examples are: a timer overflow trap, an operator interrupt trap, or termination of an I/O operation. These would set a level 4 trap request while executing MINIFLOW in level 3 and they would also set appropriate indicators to convey the trap conditions before EXITing to the scheduler.

The scheduler will normally honor the trap request ahead of the program request in level 4. The W-I-S will select the trap control and transfer vector at mini-location 233 which in turn points to the trap analysis routine. The trap request will be inhibited if the console AUTO/MANUAL switch is in manual. The trap request is also inhibited if the trap postpone flip-flop is on. This flip-flop may be set by the MISC POST instruction. It will be reset each time the scheduler honors a level 4 request. Therefore, this flip-flop will postpone trapping only long enough to execute one object instruction. If I/O trapping involves remote execution rather than directly altering the program sequence, then AUX 2 may be used as outlined in the next paragraph.

AUXILIARY INSTRUCTION COUNTERS are used to execute an instruction out of sequence, i.e., remote execution. This is done by storing the address of the instruction to be executed in either control memory octal location 4 (AUX 1) or octal location 10 (AUX 2) by means of the CMI POP. This causes the AUX 1 (or AUX 2) flip-flop to be set. When the scheduler next honors level 4 program, the W-I-S will start with YA0 where the AUX 1 (or AUX 2) is read from control memory and the low order 15 bits are used to address the object instruction. The IC is not used and not incremented. The AUX 1 (or AUX 2) flip-flop is reset so that the W-I-S control will return to the IC the next time. AUX 1 has priority over AUX 2 when both are set, but AUX 2 will be used before returning to execute instructions pointed to by the IC.

When AUX 1 is on and an instruction from group 7 (code ±007x) or group 21 (code -1xxx or code ±021x) is translated, then GIN is not set as it normally would be. This allows MINIFLOW to detect if the object instruction is being remotely executed.

CONSOLE CONTROLS

Console switches and indicators on the operator's console and on the maintenance panel are used to control and indicate machine status to the operator. Many, but not all, of these are accessible to MINIFLOW interruption and control. Some indicators are controlled jointly between MINIFLOW and non-program control conditions. This section will treat those controls of special significance to the MINIFLOW programmer.

Most controls on the maintenance panel must first be enabled by the MAINTENANCE ENABLE switch before they are effective. This switch is located on the maintenance panel. More information on console controls and indicators is found under the following computer instruction descriptions:

INT - Console Interrupt Status
AKEYS - Address Keys Operation
KEYS - Entry keys Operation
Test Instructions
General Conditional Test Instructions

Also refer to the IC-6000 System Operation Guides for 7090/7094 Emulation and 7040/7044 Emulation.

1. The GENERAL RESET switch on the operator's console will set a console interrupt (bit 9 for a reset request) and it will reset all of the following:

   a. All the scheduler and W-I-S flip-flops.
   b. The Halt, Postpone, AUX 1, AUX 2 and Program Run flip-flops.
   c. The timer interrupt request.
   d. The IC and all index registers
   e. The AC overflow flip-flop
   f. The AC and Engine Q bit
   g. All the General Indicator flip-flops
   h. All the Channel Indicators except SOPs 00, 01, 20, 21, 30.
   i. Channel Register 1 in both channels
   j. Channel terminates and buffer service requests
   k. Tape low threshold controls
   l. Card reader and load machine control flip-flops.

2. The MACHINE RESET switch on the maintenance console, when enabled by the MAINTENANCE ENABLE switch, will stop the clock, reset the console interrupts (including a console reset request) and resynchronize the mini-step control logic and both the memories.

3. The START switch on the operator's console will start the machine clock running and will reset a level 4 HAND if HALT is on and will set a level 4 program request in the scheduler.

4. If the PROGRAM START INHIBIT switch is on and the MAINTENANCE ENABLE switch is on then the START switch on the operator's console will only start the machine clock.

5. The HALT indicator on the operator's console will be turned on when a MISC HALT is executed. Level 4 will then be in a HANG condition and cannot be taken out of this condition by either a MISC RH4 of MISC RINT. However, level 4 will be taken out of the HANG condition by the START switch. The HALT indicator will be reset by GENERAL RESET or whenever the scheduler grants level 4. MISC RESL4 will also reset the hang condition.

6. If the MANUAL switch on the operator's console is on, the program request flip-flop will be reset each time it is, honored by the scheduler. This then requires that the START switch be pressed to execute each object instruction so that the machine operates in a single step mode. However, if either Channel A or B has a data select in process (Channel Indicator SOP 26) then the manual mode will not be effective until both channel data operations are completed. This allows the execution of any LCH's which command chaining may require. The MAINTENANCE ENABLE switch being on and the MANUAL OVERRIDE switch on will cause the MANUAL switch to disregard the channel data select and process indicators and go immediately into the manual mode.

   A trap request will be honored by the scheduler when not in the manual mode. The EXECUTE ENTRY switch is only enabled when the MANUAL switch is on.

7. The six CONSOLE REQUEST switches on the operator's console have the following W-I-S priority and cause the following bit positions to be set to one in the console interrupt register:

SWITCH FUNCTION	BIT POSITION	GENERAL NOTES
GENERAL RESET	9	See Para. 1
CLEAR STORAGE	10	
LOAD CARD	11	
LOAD TAPE	12	
EXECUTE ENTRY	13	Enabled by MANUAL
EXECUTE DISPLAY	14	

   If any of these switches are enabled, then a console request is set in the scheduler and the General Indicator CON will be turned on when the request is granted.

8. The STORAGE CLOCK OFF switch on the operator's console will prevent the real time clock from setting console requests when the switch is down. When this switch is up the interval timer interrupt flip-flop is turned on every sixtieth of a second. This sets a console request in the scheduler and turns on bit 8 of the console interrupt register.

9. There are thirteen REGISTER SELECTION switches on the operator's console. They are used to select a particular register or memory location to be loaded with information from the entry keys or displayed on the display indicators. They are listed below in the order of their wired in priority:

Index Register 1	XR1	15 bits displayed
Index Register 2	XR2	15 bits displayed
Index Register 3	XR3	15 bits displayed
Index Register 4	XR4	15 bits displayed
Index Register 5	XR5	15 bits displayed

Index Register 6	XR6	15 bits display
Index Register 7	XR7	15 bits display
Main Storage (word specified by the Address keys)		
Accumulator	AC	S & Q bit also display
Multiplier Quotient	MQ	36 bits display
Instruction Counter	IC	15 bits display
Current Instruction (may only be displayed dynamically)		
Sense Indicators	SI	36 bits display

The two REGISTER SELECTION switches on the maintenance panel do not have a wired in priority but they must be enabled by the MAINTENANCE ENABLE switch, they are:

Control Storage (word specified by address keys)		
Hard Register 4	HARD 4	36 bits display

The data is placed in the display register whenever a register is accessed and its switch is down and a switch of higher priority is not down. In the case of CONTROL STORAGE or HARD 4 switches, the maintenance panel must be enabled. The CURRENT INSTRUCTION switch does not have a register associated with it. Therefore, the display register is loaded at the time that the object instruction is fetched for execution, i.e., during W-I-S step YA1. The CURRENT INSTRUCTION switch may be directly interrogated as a General Indicator (CIF-SOP76).

The Display General Indicator is turned on (DIS-SOP42) whenever the display register is turned on (DIS-SOP42). This may be used by MINIFLOW to indicate when a register has been displayed and hence which register had its display switch down.

10. The LOAD MACHINE switch, located on the maintenance panel, will cause a card to be read from the card reader (if it is ready). This places the card image into the entry table at octal word locations 100 to 132. Also, a buffer service request and a terminate request will be set in the scheduler. The MAINTENANCE ENABLE switch need not be on.

11. The 36 ENTRY KEYS on the operator's console are used by the operator to enter data into the computer. They may be loaded into the main engine registers or stored directly into either the control or main memories. Refer to the following computer instruction descriptions:

    MEM - Memory Operation
    MKEY - Memory/Keys Operation
    KEYS - Entry Keys Operation
    AKEYS - Address Keys Operation

12. The 15 ADDRESS KEYS on the operator's console are used by the operator to enter address data into the computer. Also, they are used in conjunction with a comparator circuit to indicate when a memory display or an address stop condition has occurred. The ADDRESS KEYS are used either to load main engine registers or used to specify a memory address under MINIFLOW control. Refer to the computer instructions in paragraph 11.

13. The AC OVERFLOW indicator on the operator's console will be turned on by MINIFLOW by the following:

a. An AC POP and a carry out of bit 1.

b. An ALG POP and a carry out of bit 1 with like signs and Q bit zoning.

c. A SHIFT POP and DOS or MULT SOP with a 1 bit shifted left through bit 1.

This indicator will be turned off by the GENERAL RESET switch or by translating an op code of ±014x during W-I-S step YA2. During this translation, the TSAT flip-flop is set to the exclusive OR of the AC OVERFLOW and the sign of the ±014x instruction.

The following is a list of operator console controls which may not be interrogated by MINIFLOW coding:

a.	TAPE WORD INCOMPLETE -	(CHANNEL)	SWITCH
b.	NO WRITE RING -	(CHANNEL)	LAMP
c.	TAPE MULT-SELECTED -	(CHANNEL)	LAMP
d.	TAPE UNIT REQUESTED -	(CHANNEL)	4 LAMPS
e.	CARD READER NOT READY -	(CHANNEL A)	LAMP
f.	TYPEWRITER NOT READY -	(CHANNEL A)	LAMP
g.	AC OVERFLOW* -		LAMP
h.	POWER ON/OFF CONTROLS -		LAMP/SWITCH/BUTTON
i.	PROGRAM HALT* -		LAMP
j.	MANUAL or ADDRESS HALT		LAMP
k.	INTERNAL CHECK		LAMP
l.	MAINTENANCE ENABLE* -		LAMP
m.	ADDRESS TOP CONTROLS -		4 SWITCHES
n.	START -		SWITCH
o.	MANUAL -		SWITCH
p.	CARD END-OF-FILE -		SWITCH
q.	STORAGE CLOCK OFF*		

The following is a list of maintenance panel controls which may be interrogated by MINIFLOW coding:

a.	DIVIDE CHECK and TRAP MODE -	SWITCH (GENERAL INDICATOR)
b.	ABSOLUTE DISPLAY** -	SWITCH TO DISPLAY CONTROL STORAGE
c.	HARD 4** -	SWITCH TO DISPLAY HARD REG 4

* These controls are explained in the foregoing paragraphs.

**Must be enabled by the MAINTENANCE ENABLE switch. They are then interrogated by accessing them and testing the General Indicator DISPLAY (SOP42).

COMPUTER INSTRUCTIONS

This section defines all computer instructions and describes their execution and the indicators that may be affected.

Instruction Diagrams

A diagram representing the format of the instruction is given for each class of instructions, i.e., a class may include more than one instruction. In the diagram is the alphabetic and the numerical primary operation code (POP) given in the octal number system. This can easily be converted to the binary system for reference to the bit pattern interpreted by the computer. The numbers appearing above the diagram indicate the bit positions of the computer-word that are concerned with this particular instruction. A typical format is shown below:

0 5	6	7 11	12	13 17
PRIMARY OPERATION CODE (POP)	G	SECONDARY OPERATION CODE (SOP)	E	TERTIARY OPERATION CODE (TOP)

The terms POP, SOP, and TOP are defined in the definition of terms found in the front part of the manual.

The instructions subject to precondition control are identified by the symbol G appearing in bit 6. The precondition control symbol, identified by GIN or GEX in the description, represents a 1 bit in position 6 of the instruction.

Similarly, the symbol E appearing in bit 12 represents a 1 bit in this position. Its presence indicates that, on the completion of this instruction, control will EXIT back to the scheduler or to the call program if in subroutine mode.

Definitions

1. R(POP) denotes the contents of the register specified by POP. Unless otherwise stated, the entire field is implied.

2. C(C)Z denotes the contents of the field of register C as specified by the zone TOP.

3. RO(SOP)Z denotes the result of an engine SOP operation, as specified by zone TOP.

4. R(SOP) denotes the register specified by SOP field under zone control.

5. (A)HX denotes contents of register A are half exchanged.

6. CI(SOP) denotes channel indicator specified by SOP and channel B flip-flop.

7. (ME)Z denotes the outputs of the main engine under zone control.

8. C(Y) denotes the contents of location Y, where Y refers to some location in storage.

9. I(SOP) denotes the general indicator specified by the SOP field.

10. T(SOP) denotes the test condition specified by the SOP field.

11. When the word "load" is used, the transmission of a word or a part of a word from some location in the specified core storage to some specified register is always implied. Also, the transmission may be of one register to another.

In the following instruction descriptions, an instruction format is shown for each class of instructions. Beneath the format diagram a paragraph, or more, is included describing the overall characteristics of this group of instructions. Tables of SOPs and TOPs may also be referenced to in the process of describing the characteristics. Following this will be a functional (non-hardware) description, precondition controls and affected conditions.

Note again that all addresses and numbers, unless otherwise specified, are given in the octal number system.

SHIFT OPERATIONS

Shift instructions are used to move the contents of D and B (or C) either to the right or the left of their original positions. Coupled shifts between B and D, or between C and D may be done on the whole register or in just the mantissa zones (bits 9-35). Zeros are introduced in the vacated positions of a register, with the exception of conditional floating shift B and D (9-35) left. A shift larger than the bit capacity of the register will cause the contents of the register to be replaced by zeros.

When a shift instruction is interpreted, the amount of the shift is specified by RC, with the exception of BD-N(14). The C(SOP) specify the shift operation to be performed.

SHIFT - MAIN ENGINE SHIFTS

0 5	6	7 11	12 17
66	G	SOP	TOP

Description. The C(SOP), Table 16, specify the shift operation to be performed. If the DOS(20), Table 17 is used C(C)28-35 are loaded to RC. The C(TOP) are tested when other SOPs are used. If the TOP contains 77, RC is not altered; otherwise, the C(TOP) are loaded to RC. With the exception of SOP BD-N(14), RC specifies the number of bits to be shifted.

Table 16. The Shift SOPs

CODE	MNEMONIC	NOTE	SHIFT DESCRIPTION
02	B-L		Open shift B(0-35) left
03	B-R		Open shift B(0-35) right
04	D-L		Open shift D(0-35) left
05	D-R		Open shift D(0-35) right
06	BD-L		Coupled shift B&D(0-35) left
07	BD-R		Coupled shift B&D(0-35) right
10	DIV	(1)	Divide B&D(0-35) by C(0-35)
11	MULT	(2)	Multiply C(0-35) by D(0-35) into B&D(0-35)
12	D-ROT		Ring shift D(0-35) left
14	BD-N	(3)	Normalize shift B&D(9-35) left
16	BD-LF		Floating shift B&D(0-35) left
17	BD-RF		Floating shift B&D(9-35) right
20	DOS		Do the shift specified by the table below
22	C-L		Open shift C(0-35) left
23	(C-R		Open shift C(0-35) right
26	CD-L		Coupled shift C&D(0-35) left
27	CD-R		Coupled shift C&D(0-35) right
30	FDIV	(1)	Floating divide B&D(9-35) by C(9-35)
31	FMUL	(4)	Floating Mult. (9-35) by D(9-35) into B&D(9-35)
32	BD-L9	(5)	Conditional floating shift B&D(9-35) left
33	BD-R9	(6)	Floating shift B&D(9-35) right

Table 17. The DOS Operations

GOPS			MNEMONIC	NOTE	SHIFT PERFORMED
9	10	11			
0	0	0			Perform no Shifting
0	0	1	D-ROT		Ring Shift D(0-35) left
0	1	0	B-R		Open Shift B(0-35) right
0	1	1	B-L		Open Shift B(0-35) left
1	0	0	BD-R	(7)	Coupled Shift B&D(0-35) right
1	0	1	BD-L	(8)	Coupled Shift B&D(0-35) left
1	1	0	BD-R		Coupled Shift B&D(0-35) right
1	1	1	BD-L		Coupled Shift B&D(0-35) left

Note: The shift count is taken from register C in the main engine.

SPECIAL NOTES

1. Set sign of D to EXCLUSIVE OR of B and C signs: also set PDCK general flip-flop* if C is less than B initially. The quotient is right justified in D with a length equal to the shift count.

2. Set signs of B and D to EXCLUSIVE OR of C sign and initial D sign. The product is left justified in B and D with a length equal to 36 plus the shift count.

3. The normalize SOP loads the shift count to RC and increments RC until a one bit moves into bit position 9 of B; if there is not a one bit in B(9-35) or in D(9-35) the machine will shift independently.

4. Set signs of B and D to EXCLUSIVE OR of C sign and initial D sign. The product is left justified to bit 9 and is in B(9-35) and D(9-35); its length is equal to 27 plus the shift count.

5. Shift only if bit 9 of B is reset. Use a shift count of one unless otherwise specified.

6. A one bit is inserted into bit 9 of B. Use a shift count of one unless otherwise specified.

7. The B sign replaces the D sign and the B sign is not changed.

8. The D sign replaces the B sign and the D sign is not changed.

Execution. Shifting is accomplished by a two stage shift matrix which allows a shift from 0 to 4 in either direction for each cycle time. Normalize shift justifies to bit position 9 in the B register.

All shifts continue until the shift count is reduced to zero, except the normalize which increments the count and stops when a 1 bit is in bit 9 of B. At the completion of the SHIFT instruction, the C(RC) will be zero. For BD-N the RC will contain the C(TOP) plus the number of bits shifted.

Multiplication is accomplished by alternate adds and right shifts. Go to step one if the LSB of the D register is one, and go to step 2 if it is zero.

Step 1. Add the C register to the B register and shift B and D one bit position to the right.

Step 2. If the shift count is zero, condition the CARRY, ZERO, MSB and LSB indicators and test the multiplication result (product) in the B register. Then execute the next sequential instruction. If the LSB of D is on go to step 1, if it is off, continue shifting to the right. Repeat this until either a one bit is encountered or the shift count is reduced to zero.

Division is accomplished by alternate subtractions or additions followed by left shifts in a non-restoring algorithm until the shift count goes to zero. The remainder is then tested and, if in the adding mode, it is complemented.

*See general flip-flop tests.

Precondition Control. If GIN is set and bit 6 of the instruction is on, no shifting takes place but the zero, MSB, LSB and carry tests are performed on the unshifted data.

Affected Conditions. In addition to the registers being shifted the following will also happen:

1. Signs will be changed as indicated by notes 1, 2, 4, 7, and 8.
2. The ZERO, CARRY, MSB and LSB flip-flops will be conditioned for testing to reflect the last value of the shift operation.

COMBINATION OPERATIONS

Combination operations are used to move the adder outputs or the contents of main engine register (B, C, D) to any designated register.

When main engine SOPs are used, the POP field designates the receiving register, the SOP field designates the operands and the operation performed and the TOP field designates the zone field participating in the operation. The main engine SOPs are shown in Table 18, and the main engine zones are shown in Table 22. The ZERO, CARRY, MSB and LSB flip-flops reflect the result of the adder operation performed.

In all these instructions, the condition flip-flops are set before the half exchange (if any) takes place. In this case data from the sending register is tested rather than data from receiving register.

Table 18. The Main Engine SOPs

CODE	MNEMONIC	ENGINE OUTPUT
00	ZERO	All bits are zero
01	DOL	(See the Table 20)
02	NB+1	Twos complement of B
03	NC+1	Twos complement of C
06	B-C-I	B minus C minus previous carry
07	C-B-I	C minus B minus previous carry
10	B·NC	B AND ones complement of C
11	NB·C	C AND ones complement of B
12	BEC	B OR ELSE C (exclusive OR)
14	B-1	B minus one
15	C-1	C minus one
16	B-C	B minus C
17	C-B	C minus B
20	D	D register
22	NB	Ones complement of B
23	NC	Ones complement of C
24	B·C	B AND C

Table 18. The Main Engine SOPs (Cont)

CODE	MNEMONIC	ENGINE OUTPUT
26	B+C+I	B plus C plus previous carry
30	B	B register
31	C	C register
32	BUC	B OR C
34	B+1	B plus one
35	C+1	C plus one
36	B+C	B plus C

Table 19. The Load SOPS

SOP CODE	MNEMONIC	REGISTER LOADED
04	LDB	B
05	LDC	C
21	LDD	D

The DOL SOP, which is a main engine SOP, causes the logical operations to be performed, as shown in Table 20.

Table 20. The DOL Operations

GOPS			MNEMONIC	DOL OPERATIONS
9	10	11		ENGINE OUTPUT
0	0	0	B	B Register
0	0	1	BEC	B OR ELSE C
0	1	0	C	C Register
0	1	1	B·NC	B AND NOT C
1	0	0	B·C	B AND C
1	0	1	BUC	B OR C
1	1	0	NB·C	NOT B AND C
1	1	1	B·NC	B AND NOT C

Using the load SOP's, Table 19, in conjunction with the PB, PC and PD operation codes causes the register specified by the POP field to be tested and forces a half exchange of the word. Zone control is effective and the arithmetic indicators (ZERO, CARRY, MSB, LSB) are set by the zoned data before the half exchange takes place. However, the entire word is half exchanged independent of the zone code used. The word is then loaded into the receiving register under zone control.

Table 21. Main Engine Operations Associated With Load SOPs

ENGINE		LOAD SOP	
POP	LDB(04)	LDC(05)	LDD(21)
PB	Test B (B) HX to B	Test B (B) HX to C	
PC	Test C (C) HX to B	Test C (C) HX to C	
PD	Test D (D) HX to B	Test D (D) HX to C	Test D (D) HX to D

Table 22. Main Engine Zones

ZONE CODE	FROM-TO MNEMONIC	FIELD LENGTH	SPECIAL NAMES FOR THE FIELDS
00	00-35	36	Full 36 bit register
01	30-35	6	Character 5
02	30-32	3	Octal Char 10
03	27-35	9	4th Quarter word
04	21-23	3	Octal Char 7
05	QQ-08	10	Floating AC Characteristic
06	24-26	3	Octal Char 8
07	21-35	15	Address field
10	18-20	3	Tagfield and Octal Char 6
11	33-35	3	Octal Char 11
12	27-29	3	Octal Char 9
13	24-35	12	Right third word
14	18-23	6	Character 3
15	18-26	9	3rd Quarter word
16	24-29	6	Character 4
17	18-35	18	Right half word
20	09-35	27	Floating mantissa
21	12-17	6	Character 2
22	12-14	3	Octal Char 4
23	09-17	9	2nd Quarter word
24	03-05	3	Octal Char 1
25	03-08	6	
26	16-08	3	Octal Char 2
27	03-17	15	Decrement Field
30	00-02	3	Prefix and octal Char. 0
31	15-17	3	Octal Char 5
32	09-11	3	Octal Char 3
33	QQ-35	37	Full 37 bit (AC) register
34	00-05	6	Character 0
35	00-08	9	1st quarter word and floating Char.
36	06-11	6	Character 1
37	00-17	18	Left half word

PB-PLACE IN B

0 5	6	7 11	12	13 17
50	G	SOP	E	TOP

Description. With a main engine SOP, the operation specified by the SOP is performed. The results replace C(B)Z. With a load SOP, the C(B) are half exchanged and the resulting data replace C(SOP)Z.

Execution. See Figure 11.

Precondition Control. GIN set and bit 6 set to a one inhibits word from being changed in the registers but the zoned tests are still made. Bit 12 set causes the instruction to exit.

Affected Conditions. The ZERO, CARRY, MSB and LSB tests are conditioned.

PC-PLACE IN C

0 5	6	7 11	12	13 17
51	G	SOP	E	TOP

Description. With a main engine SOP, the operation specified by the SOP is performed. The results replace C(C)Z. With a load SOP, the C(C) are half exchanged and the resulting data replace C(SOP)Z.

Execution. See Figure 11.

Precondition Control. GIN set and bit 6 set to a one inhibits word from being changed in the registers but the zoned tests are still made. Bit 12 set causes the instruction to exit.

Affected Conditions. The ZERO, CARRY, MSB and LSB tests are conditioned.

PD-PLACE IN D

0 5	6	7 11	12	13 17
53	G	SOP	E	TOP

Description. With a main engine SOP, the operation specified by the SOP is performed. The results replace C(D)Z. With a load SOP, the C(D) are half exchanged and the resulting data replace C(SOP)Z.

Execution. See Figure 11.

Precondition Control. GIN set and bit 6 set to a one inhibits word from being changed in the registers but the zoned tests are still made. Bit 12 set causes the instruction to exit.

Affected Conditions. The ZERO, CARRY, MSB and LSB tests are conditioned.

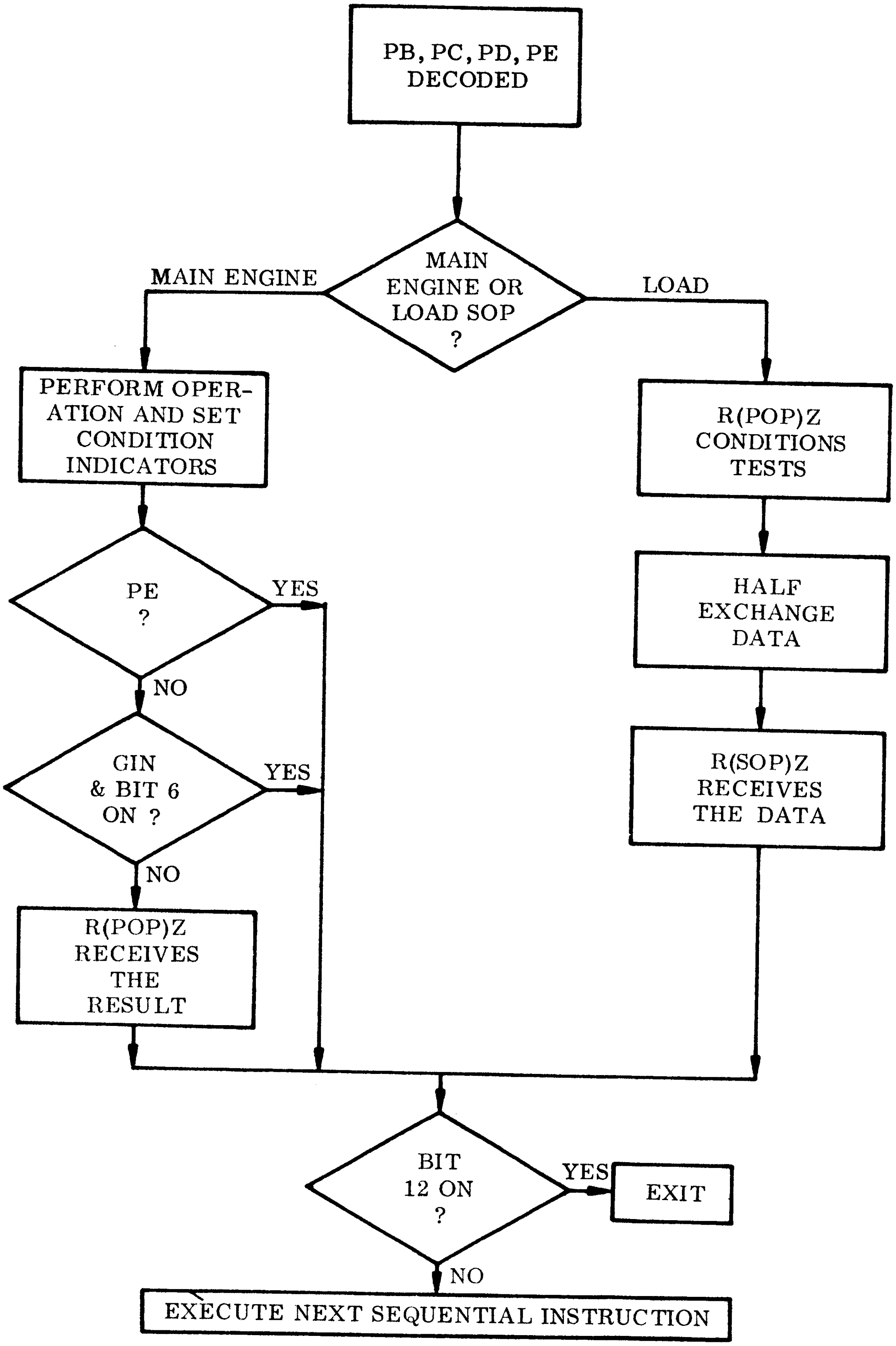

PB, PC, PD, PE DECODED
MAIN ENGINE OR LOAD SOP ?
MAIN ENGINE
LOAD
PERFORM OPERATION AND SET CONDITION INDICATORS
R(POP)Z CONDITIONS TESTS
PE ?
YES
NO
HALF EXCHANGE DATA
GIN & BIT 6 ON ?
YES
NO
R(SOP)Z RECEIVES THE DATA
R(POP)Z RECEIVES THE RESULT
BIT 12 ON ?
YES
EXIT
NO
EXECUTE NEXT SEQUENTIAL INSTRUCTION

Figure 11. PB, PC, PD, PE Flow Chart

PE-PERFORM OPERATION SPECIFIED BY MAIN ENGINE SOP

0 5	6	11	12	13 17
52	G	SOP	E	TOP

Description. With a main engine SOP the (ME)Z are tested. Only main engine SOPs are used.

Execution. See Figure 11.

Precondition Control. Bit 12 set causes the instruction to EXIT. GIN is meaningless when used with PE.

Affected Conditions. The ZERO, CARRY, MSB and LSB are conditioned.

PBD-PLACE D INTO B

0 5	6	7 11	12	13 17
40	G	SOP	E	TOP

Description. The C(D) will replace C(B)Z and the (ME)Z replace C(D). The ZERO, CARRY, MSB and LSB are conditioned by the original C(D)Z for testing. Only main engine SOPs are used.

Execution. See Figure 12.

Precondition Control. If GIN is set and bit 6 is set to one, the registers are not changed. Bit 12 set causes the instruction to EXIT.

Affected Conditions. The ZERO, CARRY, MSB and LSB are conditioned for testing.

NOTE

> The D SOP with a PBD will load B with C(D) and will replace C(D) with zeros.

PCD-PLACE D INTO C

0 5	6	7 11	12	13 17
41	G	SOP	E	TOP

Description. The C(D) will replace C(C)Z and then the (ME)Z replace C(C). The ZERO, CARRY, MSB and LSB are conditioned by the original C(D)Z for testing. Only main engine SOPs are used.

Execution. See Figure 12.

Precondition Control. If GIN is set and bit 6 is set to one, the word is not changed in the registers. Bit 12 set causes the instruction to EXIT.

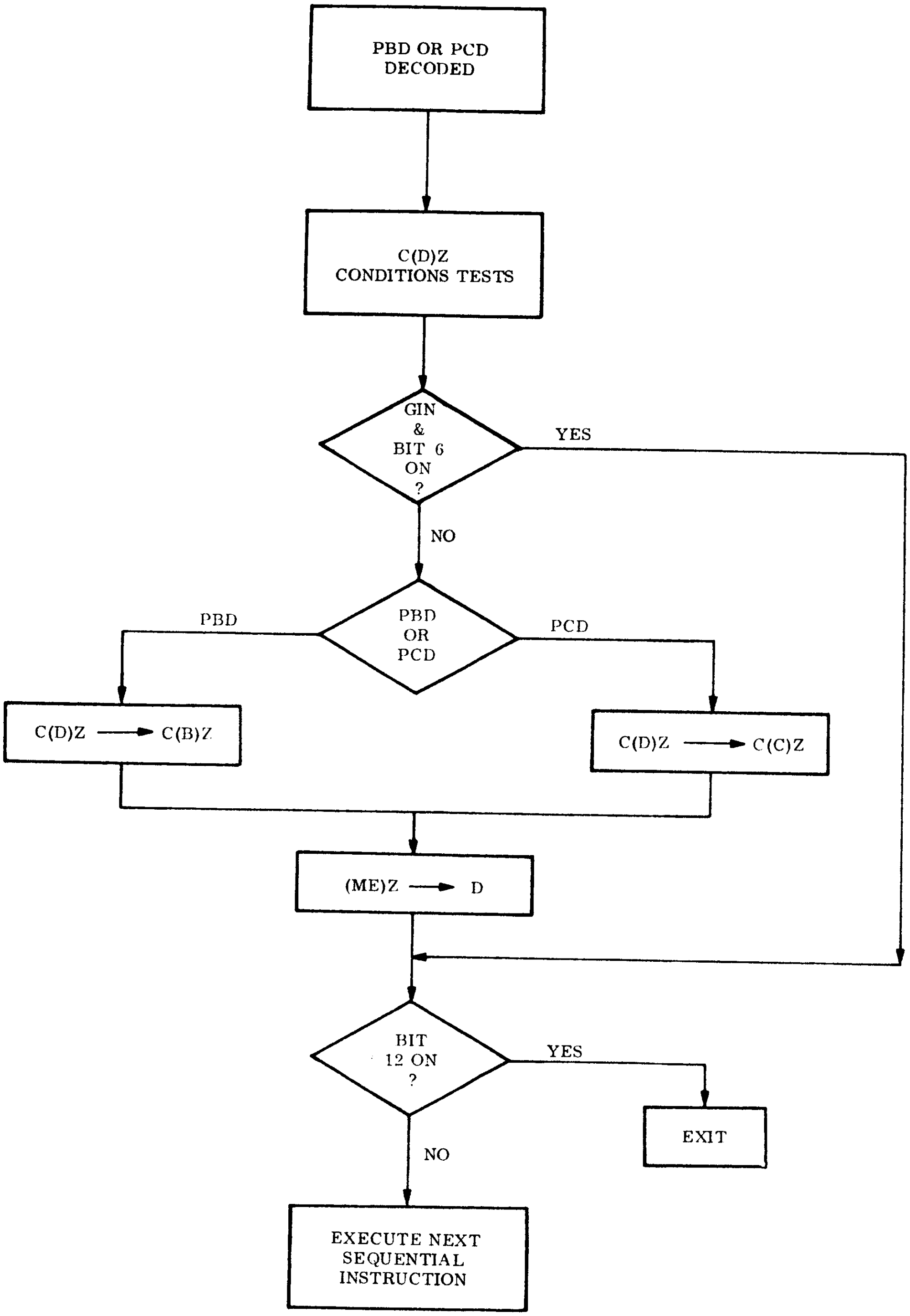


Figure 12. PBD and PCD Flow Chart

Affected Conditions. The ZERO, CARRY, MSB and LSB are conditioned for testing.

NOTE

> The D SOP with a PCD will load C with C(D) and will replace C(D) with zeros.

ALG-ZONED ALGEBRAIC MAIN ENGINE OPERATION

0 5	6	7 11	12	17
62	0	16	E	TOP

Description. The C(B)Z and the C(C)Z are added algebraically. The sum replaces C(B)Z and the sign of B is set to the algebraic sign of the sum.

Indicators. Zero, Carry, LSB, MSB, FIRST CARRY, AC OVERFLOW are set by a carry from bit position 1 if TOP code of 05 or 33 are used, i.e., the Q bit is included in the zone.

Execution. The instruction is executed in 2 cycles. In the first cycle the magnitudes are compared and in the second cycle the addition is performed. The way in which the additions are performed depends on the operand signs and the result of magnitude test performed in the first cycle. With like signs the operands are added and the sign of the result is the same as the sign of the operands. With unlike signs, the operand of smaller magnitude is subtracted from the operand of the larger magnitude. The sign of the result is the sign of the larger operand.

The only valid SOP used with the ALG POP is B-C with an octal code of 16. The following table summarizes the execution of the ALG POP with a B-C SOP.

ENGINE OPERATION

SOP	LIKE	UNLIKE SIGNS	
CODE	SIGNS	NO 1st CARRY	1st CARRY*
16	B+C	B-C	C-B

On the first cycle C(B) are not modified but if the signs are opposite then a carry out sets a special flip-flop called FIRST CARRY rather than setting the CARRY flip-flop. If this is not the case, the FIRST CARRY is not set. On the second cycle, the SOP is effectively changed as follows: If the signs of B and C are the same the operation performed is B+C. If the FIRST CARRY is set then the sign of B is toggled the operation performed is C-B. If the signs are not alike and the FIRST CARRY is false the operation performed is B-C. The C(B) now contain the algebraic sum of the B and C registers. The FIRST CARRY flip-flop is set only if the B and C signs are opposite and there is a carry out of the zoned field. If the AC Q bit is on and the zone includes the Q bit, the FIRST CARRY cannot be set.

*Toggle B sign to equal C sign.

Precondition Control. If GIN is set and bit 6 is set to a one no registers are altered. Bit 12 set to a one causes the instruction to exit.

Affected Conditions. The RB sign, AC OVERFLOW, FIRST CARRY, ZERO, CARRY, MSB, LSB.

REGISTER STACK OPERATIONS

The instructions in this group operate on the following hardware registers: AC, MQ, SI, HARD 4, IC, XR1, XR2, XR3, XR4, XR5, XR6 and XR7. Main engine SOPs are used in this group of instructions. A one in bits 13 or 14 cause the hard register to be used as one of the operands in place of B or C, respectively. Bits 16-17 specify the destination of the main engine result.

Arithmetic or logical operations may be performed using the main engine or the hardware stack registers as operands. The R(POP) designates the register participating in the execution of the instruction. The operation specified by the SOP field, bits 7-11, is performed. The result is placed into the register specified by the TOP, bits 15-17. The format is shown below:

0 5	6	7 11	12	13 17
POP	G	SOP	E	TOP

The B or C operand specified by the SOP will optionally be replaced by the hard register specified by the POP under control of bits 13 or 14, respectively. The programmer has a choice of placing the result into any one of the main engine registers and/or the hardware stack registers.

The TOP field controls the instruction as shown in Table 23.

Table 23. Function of Bits 13-17

BITS	VALUE	OPERATION
13	0	Use B operand
	1	Replace B operand with stack register
14	0	Use C operand
	1	Replace C operand with stack register
15	0	Stack register unchanged
	1	Result to stack register
16-17	00	Engine registers unchanged
	01	Result into B
	10	Result into C
	11	Result into D
15-17	111	The C(D) will go to the hard register and the engine output will go to D.

Because of the replace capability, it is possible to change the contents of either a main engine register or a stack register or both with a single instruction.

AC-AC REGISTER OPERATION

0 5	6	7 11	12	13 17
33	G	SOP	E	TOP

Description. The main engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. The results are placed into the AC and/or one of the main engine registers as specified by bits 15-17. A zone of QQ-35 applies to the AC.

Execution. See Figure 13.

Precondition Control. With GEX set and bit 6 set to a one, the AC will be half exchanged before being used as an operand in the main engine. The result will be half exchanged before being loaded into the AC.

Affected Conditions. If bit 15 of the instruction is a one (result to AC), a carry from position 1 to 0 of the adder will set the AC OVERFLOW. The engine Q bit, AC Q bit and AC sign bit may be changed.

MQ-MQ REGISTER OPERATION

0 5	6	7 11	12	13 17
37	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. The results are placed into the MQ and/or one of the main engine registers as specified by bits 15-17. A zone of 00-35 applies to the MQ.

Execution. See Figure 13.

Precondition Control. With GEX set and bit 6 set to a one, the MQ will be half exchanged before being used as an operand in the main engine. The result will be half exchanged before being loaded into the MQ.

Affected Conditions. The MQ bit zero communicates with the main engine sign if ARI is on.

SI-SI REGISTER OPERATION

0 5	6	7 11	12	13 17
27	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. The results are placed into the SI and/or one of the main engine registers as specified by bits 15-17. A zone of 00-35 applies to the SI.

Execution. See Figure 13.

Precondition Control. With GEX set and bit 6 set to a one, the SI will be half exchanged before being used as an operand in the main engine. The result will be half exchanged before being loaded into SI and/or register specified by bits 15-17.

Affected Conditions. SI bit zero communicates with the main engine sign if ARI is on.

R4-HARD REGISTER NO. 4 OPERATION

0 5	6	7 11	12	13 17
23	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. The results are placed into R4 and/or one of the main engine registers as specified by bits 15-17. A zone of 00-35 applies to R4.

Execution. See Figure 13.

Precondition Control. With GEX set and bit 6 set to a one, R4 will be half exchanged before being used as an operand in the main engine. The result will be half exchanged before being loaded into R4.

Affected Conditions. R4 bit zero communicates with the main engine sign if ARI is on.

IC-IC REGISTER OPERATION

0 5	6	7 11	12	13 17
22	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. The results are placed into IC and/or one of the main engine registers specified by bits 15-17. A zone of 21-35 or 03-17 applies to the IC.

Execution. See Figure 13.

Precondition Control. No registers will be altered with GIN set and bit 6 set to a one. If GEX is set and bit 6 set to a one then half exchange is enabled and zone 03-17 is used.

Affected Conditions. The ZERO, CARRY, MSB and LSB arithmetic tests are conditioned.

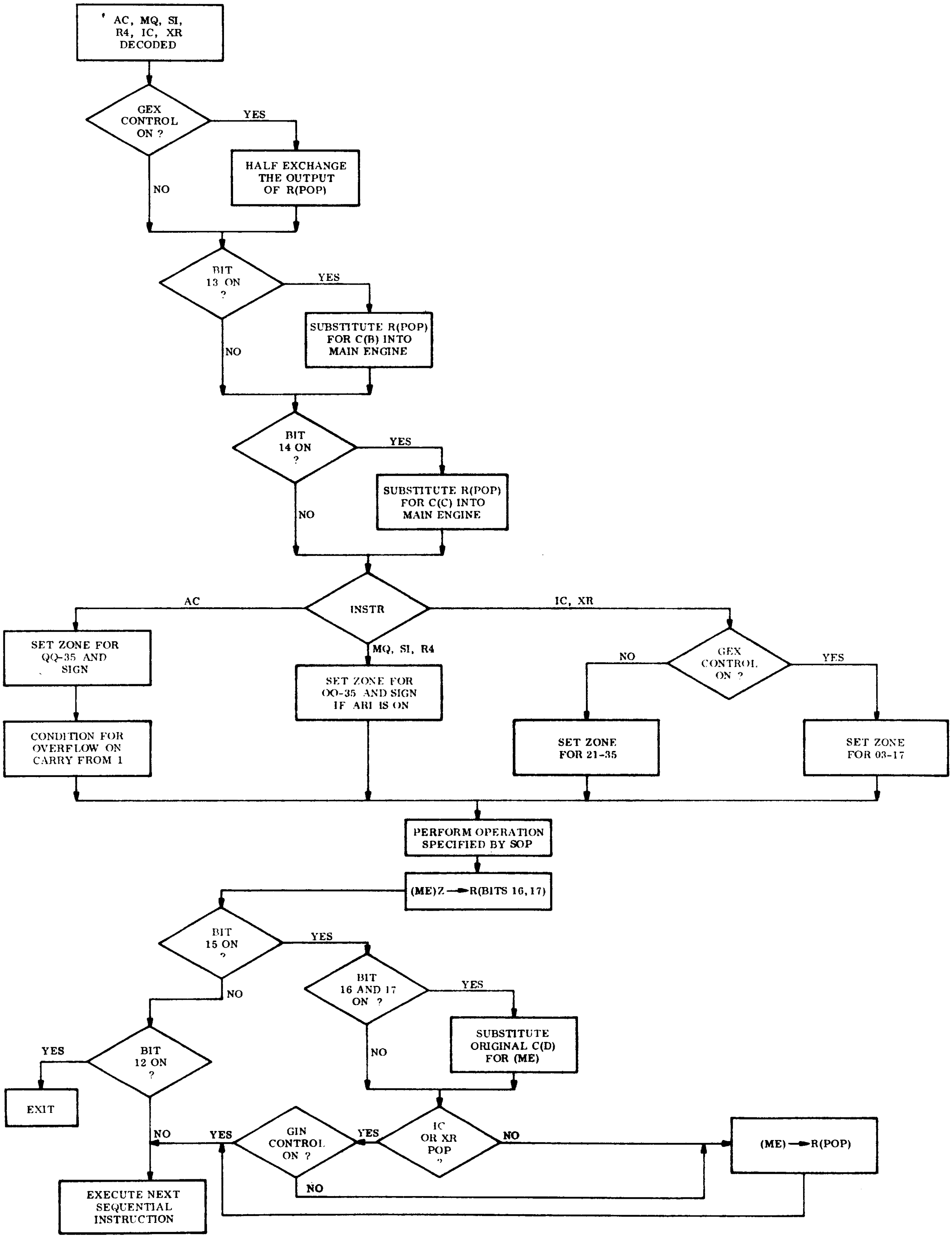

AC, MQ, SI, R4, IC, XR DECODED
GEX CONTROL ON ?
YES
NO
HALF EXCHANGE THE OUTPUT OF R(POP)
BIT 13 ON ?
YES
NO
SUBSTITUTE R(POP) FOR C(B) INTO MAIN ENGINE
BIT 14 ON ?
YES
NO
SUBSTITUTE R(POP) FOR C(C) INTO MAIN ENGINE
INSTR
AC
IC, XR
MQ, SI, R4
SET ZONE FOR QQ-35 AND SIGN
CONDITION FOR OVERFLOW ON CARRY FROM 1
SET ZONE FOR OO-35 AND SIGN IF ARI IS ON
GEX CONTROL ON ?
NO
YES
SET ZONE FOR 21-35
SET ZONE FOR 03-17
PERFORM OPERATION SPECIFIED BY SOP
(ME)Z → R(BITS 16,17)
BIT 15 ON ?
YES
NO
BIT 16 AND 17 ON ?
YES
NO
SUBSTITUTE ORIGINAL C(D) FOR (ME)
BIT 12 ON ?
YES
EXIT
NO
EXECUTE NEXT SEQUENTIAL INSTRUCTION
GIN CONTROL ON ?
YES
NO
IC OR XR POP ?
YES
NO
(ME) → R(POP)

Figure 13. AC, MQ, SI, R4, IC and XR Flow Chart

XR-INDEX REGISTER OPERATION

0 5	6	7 11	12	13 17
26	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The SOP code and bits 13 and 14 of the instruction determine which registers are to be used as operands. See Table 23. When the code specifies that XR is to be used as an operand or a result register, bits 18-20 of the D register specify the index register to be used. When bits 18-20 are zero, a zero value is assumed for the operand and no index register will receive the result. A zone of 21-35 or 03-17 applies to the XR POP.

Execution. See Figure 13.

Precondition Control. No registers will be altered with GIN set and bit 6 set to a one. If GEX is set and bit 6 is on, then half exchange is enabled and zone 03-17 is used.

Affected Conditions. The ZERO, CARRY, MSB and LSB arithmetic tests are conditioned.

NOTE

At the beginning of MINIFLOW executions, the D register contains the target instruction. However, the C(D) may be changed by the programmer at any time so that there is no guarantee that the tag field, bits 18-20 of D, will specify the same index register as originally specified by the target instruction.

MEMORY ACCESS OPERATIONS

In these instructions, the engine operation specified by the main engine SOP code is performed. The result is used as a memory address. If a load SOP is used, the address is taken from the console address keys. The TOP field specifies the memory (main or control) to be accessed and also indicates whether a read or write operation is to be executed. The functions of bits 13-17 are shown in Table 24, for MEM and MKEY only.

Table 24. Functions of the TOP Field

BITS	VALUE	OPERATION
13	0	Access the control memory
	1	Access the main memory
14	0	Read from the memory
	1	Write into the memory
		Write the ENTRY KEYS if POP=63 (MKEY)
15	0	Normal SOP generates address
	1	Replace B in SOP with R4
16-17	00	*Write zeros
	01	B is data register
	10	C is data register
	11	D is data register.

MEM-MEMORY OPERATION

0 5	6	7 11	12	13 17
67	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The result is used as a memory address. Main engine SOPs are used if bit 15 is a one. Refer to Table 24 for a description of functions specified by the bits of the TOP field. These bits specify the memory to be accessed. Also, they indicate whether a read or a write operation is to be performed.

The console ADDRESS KEYS will be loaded into the register specified by a load SOP. The load SOPs will cause the ADDRESS KEYS to be used to address the specified memory.

Execution. See Figure 14.

Precondition Control. With GEX set and bit 6 set to a one, the data word will be half exchanged between memory and the specified register. Bit 12 set to a one causes the instruction to exit.

Affected Conditions. The sign of the data register if ARI is on during a read operation. If ARI is set on a write operation, the data register sign bit, rather than bit 0, replaces bit 0 of the memory word.

*X0X00 is a NOP

MKEY-MEMORY/KEYS OPERATION

0 5	6	7 11	12	13 17
63	G	SOP	E	TOP

Description. The engine operation specified by the SOP is performed. The result is used as a memory address. Main engine SOPs are used if bit 15 is a one. Refer to Table 24 for functions specified by the bits of the TOP field. These bits specify the memory to be accessed and they also indicate whether a read or a write operation is to be performed.

When writing into memory, the data placed into the console ENTRY KEYS are used. The load SOPs will cause the console ADDRESS KEYS to be used to address the specified memory but they will not be loaded into any main engine register.

Execution. See Figure 14.

Precondition Control. With GEX set and bit 6 set to a one, the data word will be half exchanged between memory and the specified register. Bit 12 set to a one causes the instruction to EXIT.

Affected Conditions. The sign of the data register if ARI is on during a read operation.

CMI-DIRECTLY ADDRESSED CONTROL MEMORY OPERATION

0 5	6	11	12	13 17
77	G	SOP	E	TOP

Description.

For MAIN ENGINE SOPs: The operation specified by the C(SOP) is performed. The result replaces the C(control memory) addressed by the TOP.

For LOAD SOPs: The contents of memory addressed by the TOP replaces the register addressed by the SOP.

Execution. See Figure 14. It is possible to directly address the first 32 words of control memory. The data used to write into control memory is generated from the main engine as specified by a main engine SOP. A load SOP will specify which engine register to read into from control memory. The output of the Channel B flip-flop is ORed into the TOP field so that, if the channel B flip-flop is set, only the odd control memory addresses will be accessed.

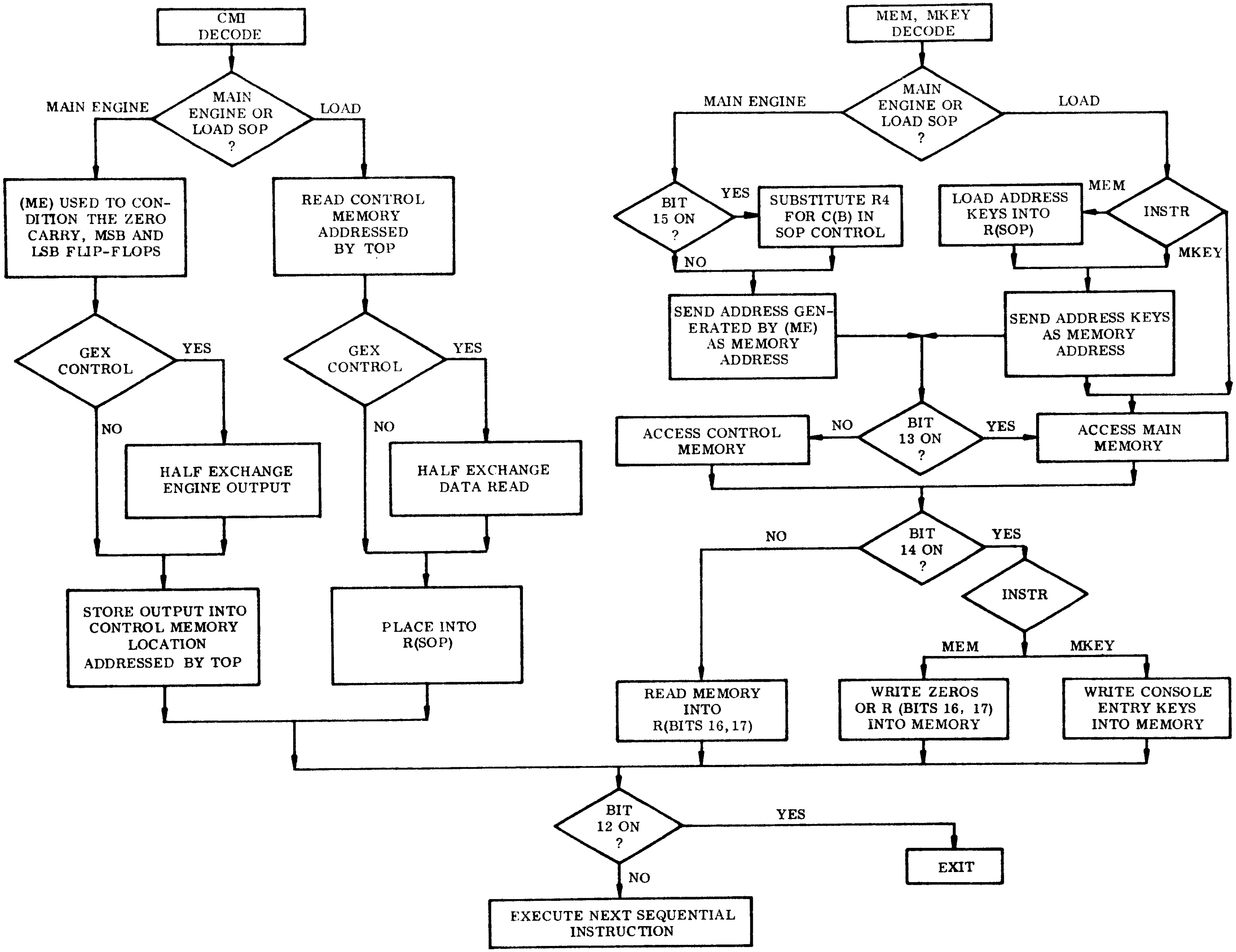

CMI DECODE
MAIN ENGINE OR LOAD SOP ?
MAIN ENGINE
LOAD
(ME) USED TO CONDITION THE ZERO CARRY, MSB AND LSB FLIP-FLOPS
READ CONTROL MEMORY ADDRESSED BY TOP
GEX CONTROL
YES
NO
HALF EXCHANGE ENGINE OUTPUT
GEX CONTROL
YES
NO
HALF EXCHANGE DATA READ
STORE OUTPUT INTO CONTROL MEMORY LOCATION ADDRESSED BY TOP
PLACE INTO R(SOP)
MEM, MKEY DECODE
MAIN ENGINE OR LOAD SOP ?
MAIN ENGINE
LOAD
BIT 15 ON ?
YES
NO
SUBSTITUTE R4 FOR C(B) IN SOP CONTROL
LOAD ADDRESS KEYS INTO R(SOP)
MEM
INSTR
MKEY
SEND ADDRESS GENERATED BY (ME) AS MEMORY ADDRESS
SEND ADDRESS KEYS AS MEMORY ADDRESS
ACCESS CONTROL MEMORY
NO
BIT 13 ON ?
YES
ACCESS MAIN MEMORY
NO
BIT 14 ON ?
YES
INSTR
MEM
MKEY
READ MEMORY INTO R(BITS 16, 17)
WRITE ZEROS OR R (BITS 16, 17) INTO MEMORY
WRITE CONSOLE ENTRY KEYS INTO MEMORY
BIT 12 ON ?
YES
EXIT
NO
EXECUTE NEXT SEQUENTIAL INSTRUCTION

Figure 14. MEM, MKEY and CMI Flow Chart

A write operation with a TOP specifying octal location either 04 or 10 will write into memory and also turn on either AUX 1 or AUX 2 flip-flop, respectively. Refer to the paragraphs describing REMOTE EXECUTION and TRAPPING in Section III for more information on the AUX 1 and AUX 2 flip-flops and the Wired-In-Sequence.

Precondition Control. GEX being set and the 6 bit set to a one causes the data to be half exchanged between control memory and the main engine. The 12 bit set to a one causes the instruction to exit.

Affected Conditions. Register signs are not affected by the CMI instruction. ZERO, CARRY, MSB and LSB tests are conditioned for write operations.

MINI ENGINE OPERATIONS

The instructions in this group are: MINI, RB, RC and RD. The MINI instruction uses the mini engine to perform arithmetic and logical operations as specified by the SOP field and the TOP field. The RB, RC and RD instructions are used for the transfer of data between the mini engine and the main engine.

Using main engine SOPs with either RC or RD causes a transfer of data from the main engine to the mini engine.

Using load SOPs with either RB, RC or RD causes a transfer of data from the mini engine to the main engine.

RB is an 11-bit register with bit positions 25-35.
RC is an 8-bit register with bit positions 28-35.
RD is an 11-bit register with bit positions 25-35.

MINI-MINI ENGINE COMBINATION OPERATION

0 5	6	7 11	12	13	14	15	16	17
02	///	SOP	E	//	DC	TO RB	TO RC	To RD

Description. Perform operation specified by the SOP field. The TOP field, i.e., the bit configuration of bits 14-17, indicate transfer path (straight or down crossed) through mini engine and destination register.

Execution. See Figure 15. Arithmetic and logical operations are performed in the mini engine using the MINI instruction. The mini engine SOPs, listed as follows, indicate the arithmetic operation to be performed. Therefore, the SOP also indicates the source register.

SOP	MNEMONIC	ENGINE OUTPUT
00	ZERO	ZERO
30	RB	RB register (MINI sequence counter)
31	RC	RC register (MINI shift counter)
14	RB-1	RB minus 1
34	RB+1	RB plus 1
35	RC+1	RC plus 1
15	RC-1	RC minus 1
36	RB+RC	RB plus RC
16	RB-RC	RB minus RC

Zone control is implied by the register length/ that is, 11 bits for RB and RD and 8 bits for RC. RC acts like an 11-bit register with bits 25-27 always zeros.

The TOP field (bits 14-17) controls the destination of the mini-engine output as shown below.

TOP	DESCRIPTION
00	R(SOP) affects mini-engine zero test (RZ) flip-flop.
02	R(SOP) affects RZ flip-flop. R(SOP replaces C(RC).
04	R(SOP) affects RZ flip-flop. R(SOP) replaces C(RB).
06	R(SOP) affects RZ flip-flop. R(SOP) replaces C(RB) and C(RC).
10	C(RD) affect RZ flip-flop.
11	C(RD) affect RZ flip-flop. C(SOP) replaces C(RD).
12	C(RD) affect RZ flip-flop. C(RD) replaces C(RC).
13	C(RD) affect RZ flip-flop. C(RD) replaces C(RD). C(SOP) replaces C(RD).
14	C(RD) affect RZ flip-flop. C(RD) replaces C(RB).
15	C(RD) affect RZ flip-flop. C(RD) replaces C(RB). C(SOP) replaces C(RD).
16	C(RD) affect RZ flip-flop. C(RD) replaces C(RB) and C(RC).
17	C(RD) affect RZ flip-flop. C(RD) replaces C(RB) and C(RC). C(SOP) replaces C(RD).

Precondition Control. Bits 6 and 13 are not used in the execution of this instruction. Bit 12 set to a one causes the instruction to exit.

Affected Conditions. The mini engine zero test (RZ) flip-flop.

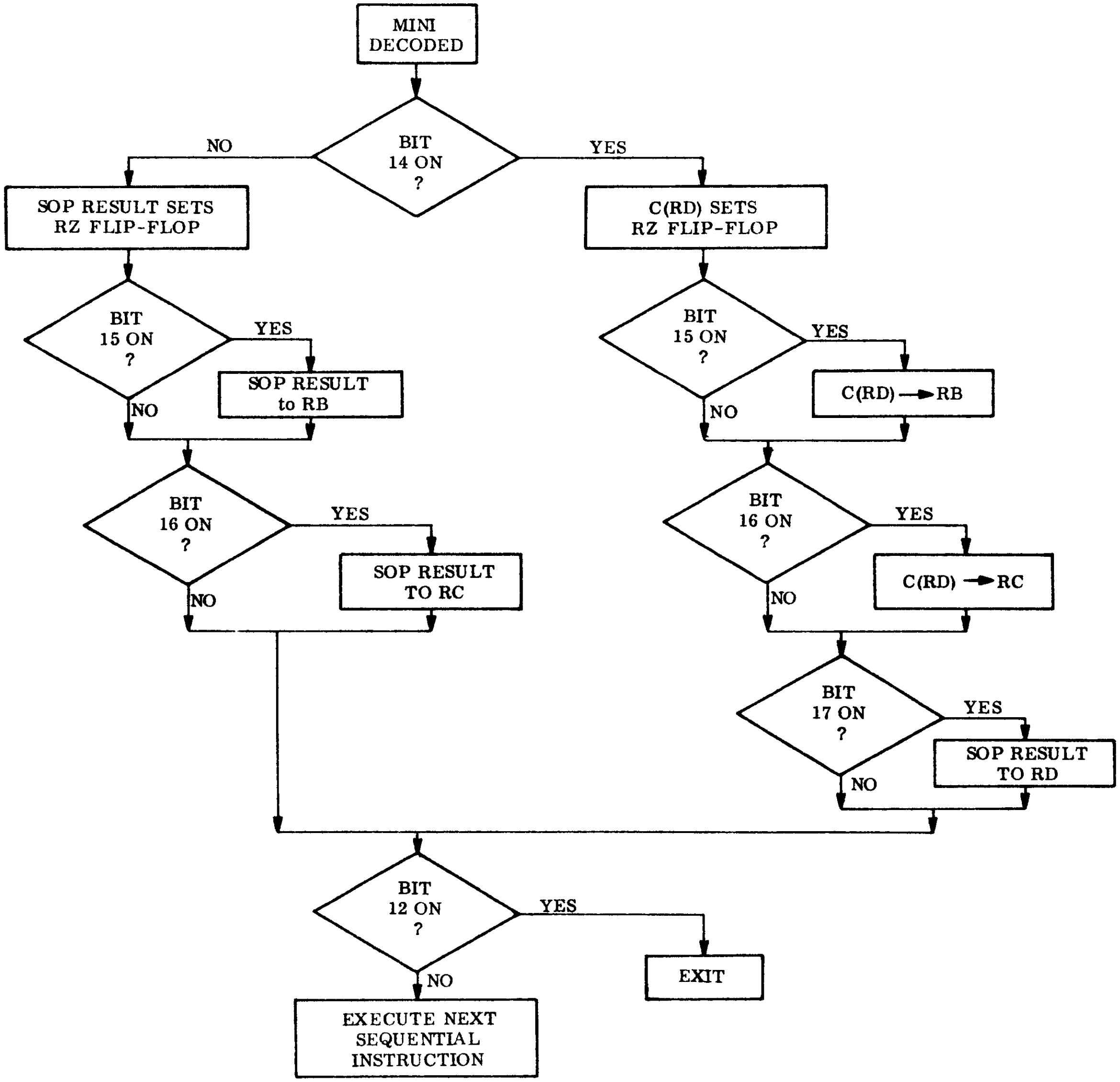


Figure 15. MINI Flow Chart

RC-MINI ENGINE C REGISTER OPERATION

0 5	6	7 11	12	13 17
42	G	SOP	E	TOP

Description.

When using:

a. Main engine SOPs, C(RC) 30-35 are replaced by the output of the adder specified by the SOP. Bits 28, 29 of RC are zeroed. The zone field is not interpreted except for zone 00-08, see the note below.

b. Load SOPs, the C(RC) 30-35 replace the register specified by the SOP under main engine zone control. Leading zeros are loaded if the zone field is for more than bits 30-35.

NOTE

> Using RC with the floating characteristics zone 00-08 (code of 35) will cause special communication between all 8 bits of the RC register and bits 1-8 of the main engine registers. This path of communication is in both directions, since load SOPs and main engine SOPs can be used.

Execution. See Figure 16.

Precondition Control. With GEX set and bit 6 set to a one, the word on the main bus will be half exchanged as it is transferred to or from the decrement fields of the main engine registers. GEX control is not effective when a zone of 00-08 is used. Bit 12 set to a one causes the instruction to exit.

Affected Conditions. The ZERO, CARRY, MSB and LSB general arithmetic tests and indicators are affected when using the main engine SOPs.

RD-MINI ENGINE REGISTER D OPERATION

0 5	6	7 11	12	13 17
46	G	SOP	E	TOP

Description.

When using:

a. Main engine SOPs, the C(RD) 25-35 are replaced by the output of the adder specified by the SOP.

b. Load SOPs, the C(RD) 25-35 replace the register specified by the SOP under main engine zone control. Leading zeros are loaded if the zone field is for more than bits 25-35.

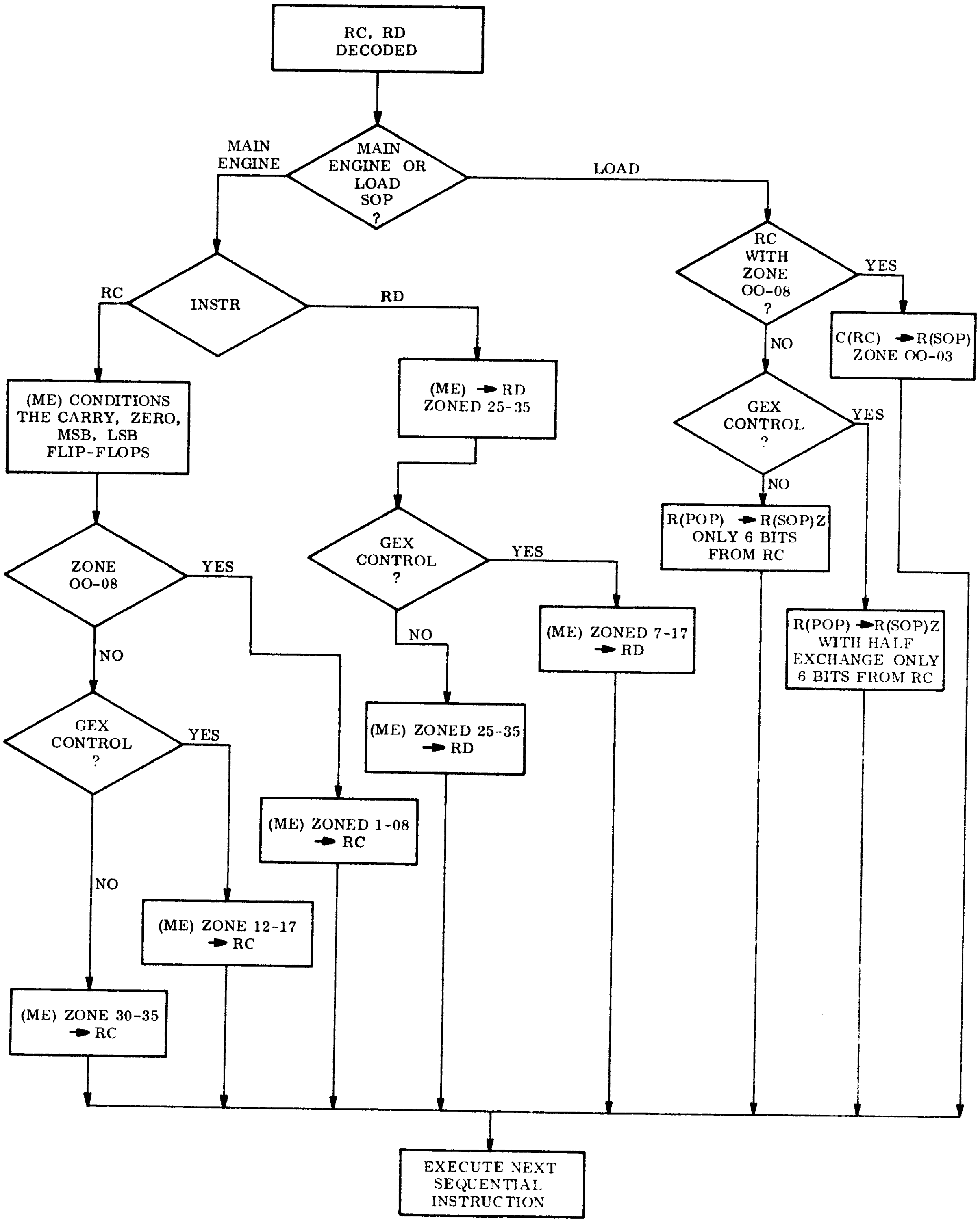


Figure 16. RC, RD Flow Chart

Execution. See Figure 16.

Precondition Control. With GEX set and bit 6 set to a one, the word on the main bus will be half exchanged as it is transferred to or from the decrement fields of the main engine registers. The instruction will exit if bit 12 is set to a one.

Affected Conditions. None.

SPECIAL DATA PATHS

The instructions in this group are: CH1, CH2, INT, AKEYS and KEYS. The CH1 and CH2 instructions initiate and terminate channel activity. Data is transferred to/from the I/O devices using the decode of channel register one to specify the type of operation.

The INT instruction allows console interrupt requests to be loaded into the R(SOP)Z. In this manner, the specific condition causing the interrupt can be determined.

The AKEYS and KEYS instruction allow the set of console keys specified to be loaded into the register specified by the SOP.

CH1-I/O CHANNEL OPERATION CONTROL

0 5	6	7 11	12	13 17
47	G	SOP	E	TOP

Description. The adder outputs are loaded to channel (A or B) register number one specified by the Channel B flip-flop. I/O operations are initiated depending on the decode of bits 0, 8-10.

Execution. Specifying a main engine SOP causes the data generated by the adder to be loaded into channel register number one and the I/O operation specified by the bit configuration of this word is initiated. Using a load SOP (LDB, LDC, LDD) reverses the load operation, i.e., the word loaded in the specified main engine register will be in the following format.

0, 9-11	Operation control bits (notice the one bit offset).
5-8	Channel mode counter flip-flops
28-30	Device type
31	BCD/BIN mode bit
32-35	Tape unit specified

When main engine SOPs are used, the following format applies.

CHANNEL REGISTER ONE I/O OPERATIONS

OPERATION CONTROL OCTAL BITS 0,8-10	TYPE DEVICE BINARY BITS 28-30	BCD/BIN MODE BIT 31	I/O OPERATION
			CARD READER OPERATIONS
01	110	1	Read from card Reader
10	XXX	X	Load machine from card reader
			TYPEWRITER OPERATIONS
03	111	1	Write on typewriter
13	111	1	Index typewriter (vertical space)
			TAPE OPERATIONS (UNIT IS SPECIFIED BY BITS 32 to 35)
01	XOX	0	Read tape BCD
01	XOX	1	Read tape BINARY
03	XOX	0	Write tape BCD
03	XOX	1	Write tape BINARY
04	XOX	X	Write end of file mark
13	XOX	X	Write blank tape (3 1/2 inches)
02	XOX	X	Backspace tape record
12	XOX	X	Backspace tape file
05	XOX	X	Rewind tape
15	XOX	X	Rewind and unload tape
07	XOX	X	Test tape for presence

Precondition Control. GEX being set and bit 6 set to a one causes the word to be half exchanged in its transmission between the main engine register and channel register one. Bit 12 being set to a one causes the instruction to exit.

Affected Conditions. A CH1 POP with a main engine SOP will always reset channel indicators ER1 and ER2, and will always cause a terminate interrupt when the I/O operation is completed. It will also reset EOR and CEF.

CH2-I/O TERMINATION CONTROL

0 5	6	7 11	12	13 17
45	G	SOP	E	TOP

Description. Channel write operations: Depending on the state of the channel B flip-flop, the adder outputs specified by the C(SOP)Z are transmitted to channel register number two. The end condition is thus specified for the channel. Any pending buffer service requests in the channel are reset and no more are set. However, if two or more requests are outstanding in the channel before the CH2 (or MISC. ACK) POP, then a buffer service request will remain in the scheduler and will be honored later.

The end condition is determined by the low order five bits of the data generated by the main engine and goes to the address stop register. The address stop register is loaded with the word count value (modulo 32) and will point to the first word which will not be written out of control memory.

When the channel address equals the stop address then the transmission of data words will terminate from control memory. An interrecord gap will be generated and the tape motion will stop. Note that the programmer determines the record length, the CH2 POP merely terminates a record when writing.

Channel read operations. Depending on the state of the channel B flip-flop, the adder outputs specified by the C(SOP)Z are transmitted to channel register number two. In the event a read select was decoded, i.e., 01 in CH1, to terminate the operation, the channel address counter stops counting. Data transmission ceases at this point; however, tape motion will not terminate until an interrecord gap is encountered.

For both read and write channel operations, a load SOP used in conjunction with a CH2 POP will cause the C(Channel reg. no. 2) to be zoned into the main engine register in the following format:

BITS	FORMAT
0-14	Zeros
15-17	Address stop register (low order 3 bits)
18-22	Zeros
23	One if bits 24-26 are all zero.
24-26	Buffer address counter (low order 3 bits)
27-29	Zeros
30	Channel B flip-flop
31-32	Address stop register (high order 2 bits)
33-35	Zeros

Precondition Controls. With GEX set and bit 6 set to a one, information is half exchanged as it is passed between the main engine and channel register number two. Bit 12 being set to a one causes the instruction to exit.

INT-CONSOLE INTERRUPT STATUS

0 5	6	7 11	12	13 17
57	G	SOP	E	TOP

Description. This instruction loads console interrupt requests into (R(SOP)Z. The console requests and their positions are shown below:

BIT	CONSOLE REQUEST FUNCTION
0	External Interrupt
1	Postpone Trap (Not a Console SW.)
8	Interval Timer Interrupt
9	Reset Request (Hardware Functions Also)
10	Clear Request
11	Load Card Request
12	Load Tape Request
13	Execute Entry Request
14	Execute Display Request
0,2-7,15-35	Zero

Execution. Console interrupt requests are zoned into a main engine register specified by a load SOP. An example is as follows:

A level 3 request enters state YA6 (MINIFLOW EXECUTION) at the address specified by the C&TV located at mini address 0241 (group 41). The routine entered should test General Indicator 74. The indicator is on if the request for service was the result of a console operation or a timer request. The indicator is off if the request for service was because of a channel termination. If the request was because of the console or timer, the instruction INT LDX can be used to determine the specific condition causing the interrupt. An example of a level 3 entry analysis routine follows:

```
00241                          ORG    0/241        EXAMPLE 26   L 3 ENTRY
00241   001500                 VFD    18/L3        LEVEL 3 C+TV
                       * - - - - - - - - - - - - - - - - - *
01500                          ORG    0/1500
01500   747402         L3      TGR    CON,*+2      CONSOLE INTERRUPT TEST
01501   161050                 TRU    IOTERM       GO TO I/O TERMINATION
01502   570400                 INT    LDB          LOAD CONSOLE INTERRUPTS
01503   523026                 PE     B,06-08      TEST FOR REAL TIME CLOCK
01504   614403                 TA     Z,*+3        SKIP IF NOT CLOCK
01505   061200                 MISC   RTI          RESET TIMER INTERRUPT
01506   161100                 TRU    ICLOCK       GO TO TIMER ROUTINE
01507   067200                 MISC   RCN          RESET CONSOLE INTERRUPTS
01510   601400                 SHIFT  BD-N         NORMALIZE PRIORITY TO RC
01511   023604                 MINI   RB+RC,RB     SKIP TO TRANSFER VECTOR
01512   161110                 TRU    RESET        GO TO RESET ROUTINE
01513   161116                 TRU    CLEAR        GO TO CLEAR CORE ROUTINE
01514   704301                 TGS    CARD         PRESET FLAG TO LOAD CARD
01515   161140                 TRU    LOAD         GO TO LOAD TAPE OR CARD
01516   704101                 TGS    ENTER        SET FLAG TO ENTER KEYS
01517   161120                 TRU    DISPLA       GO TO DISPLAY ROUTINE
00074                  CON     DGI    0/74
01050                  IOTERM  EQU    0/1050
01100                  ICLOCK  EQU    0/1100
01110                  RESET   EQU    0/1110
01116                  CLEAR   EQU    0/1116
00043                  CARD    DGI    0/43
01140                  LOAD    EQU    0/1140
00041                  ENTER   DGI    0/41
01120                  DISPLA  EQU    0/1120
```

NOTE

The normalize shift and the mini instruction provide a way to examine the console interrupt and transfer to the appropriate routine.

Precondition Control. With GEX set and bit six set to a one, the data are half exchanged os that it will be loaded into bit positions 18 and 26 through 32. Bit 12 set to a one causes the instruction to exit.

Affected Conditions. Issuing a main engine SOP with this instruction causes a NO-OPERATION.

AKEYS-ADDRESS KEYS OPERATION

0 5	6	7 11	12	13 17
54	G	SOP	E	TOP

Description. The console ADDRESS KEYS are loaded into the register specified by the load SOP under main engine zone control. AKEYS used with a main engine SOP will convert it to a DELAY instruction which causes the current level to hang for a specified time interval. The time delay is approximately 75 microseconds. Control will be returned to instruction following the delay after all other higher priority levels are serviced. With bit 13 set to a one, a half exchange will occur; thus, zoning the address keys into the decrement field.

Execution. See Figure 17.

Precondition Control. With GEX set and bit 6 set to a one, the word is half exchanged prior to loading into destination register. Bit 12 set to a one causes the instruction to EXIT.

KEYS-ENTRY KEYS OPERATION

0 5	6	7 11	12	13 17
55	G	SOP	E	TOP

Description. The console ENTRY KEYS are loaded into the register specified by the load SOP under main engine zone control. Using a main engine SOP with KEYS causes the contents of the main engine register specified by the SOP to be displayed.

Execution. See Figure 17.

Precondition Control. With GEX set and bit 6 set to a one, the word is half exchanged prior to loading into the destination register.

IMMEDIATE DATA OPERATIONS

The LIB, LIC and LID instructions in this group allow data to be zoned into any main engine register using bits 6-11 of the instructions for a 6-bit octal pattern.

The MOPB and MOPC instruction allow the entire mini instruction to be loaded into the specified register.

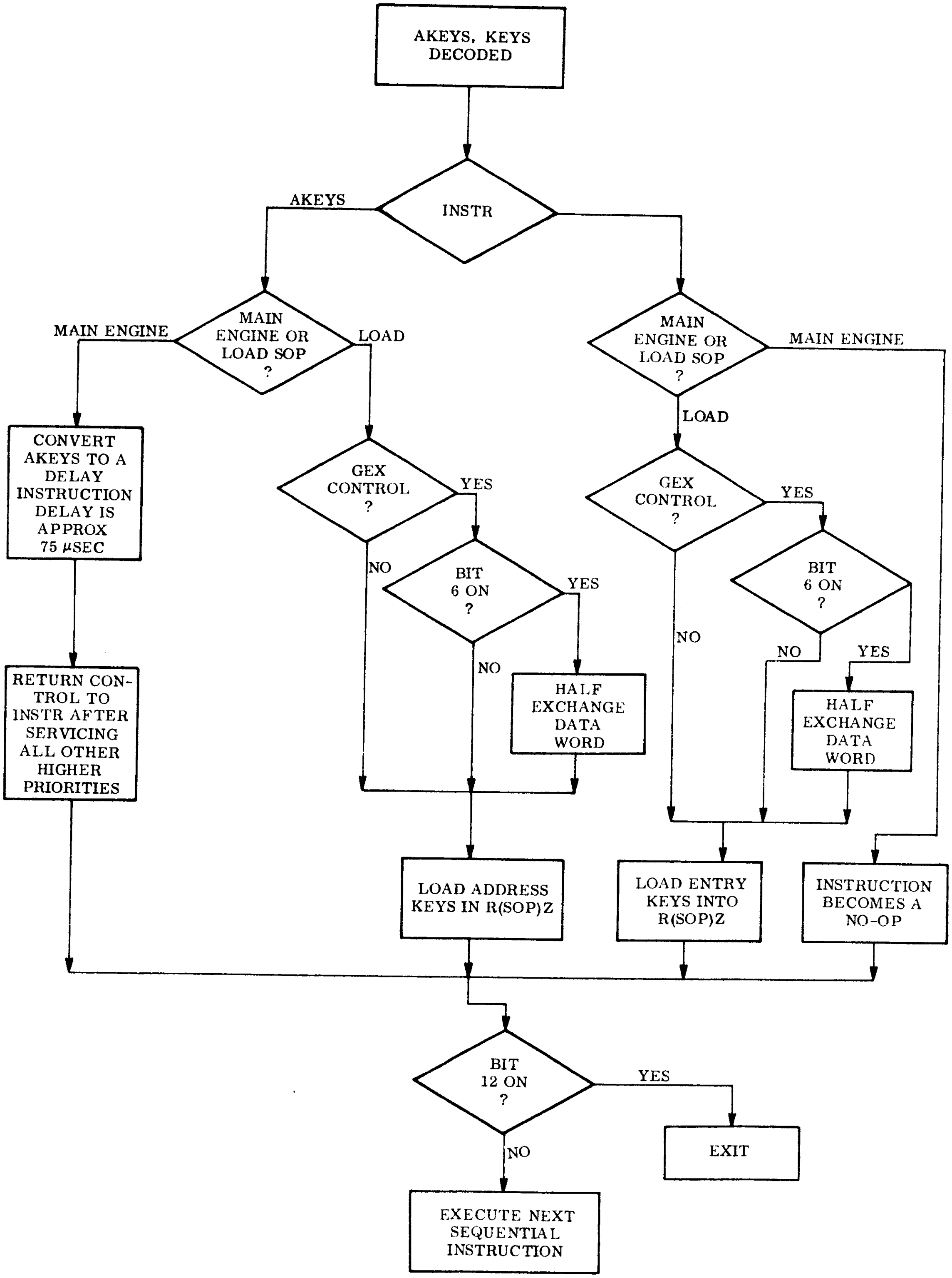

AKEYS, KEYS DECODED
INSTR
AKEYS
MAIN ENGINE
MAIN ENGINE OR LOAD SOP ?
LOAD
MAIN ENGINE OR LOAD SOP ?
MAIN ENGINE
LOAD
CONVERT AKEYS TO A DELAY INSTRUCTION DELAY IS APPROX 75 μSEC
GEX CONTROL ?
YES
GEX CONTROL ?
YES
NO
BIT 6 ON ?
YES
BIT 6 ON ?
NO
NO
NO
YES
RETURN CONTROL TO INSTR AFTER SERVICING ALL OTHER HIGHER PRIORITIES
HALF EXCHANGE DATA WORD
HALF EXCHANGE DATA WORD
LOAD ADDRESS KEYS IN R(SOP)Z
LOAD ENTRY KEYS INTO R(SOP)Z
INSTRUCTION BECOMES A NO-OP
BIT 12 ON ?
YES
EXIT
NO
EXECUTE NEXT SEQUENTIAL INSTRUCTION

Figure 17. AKEYS, KEYS Flow Chart

LIB-LOAD B

0 5	6 11	12	13 17
14	DATA	Z	TOP

Description. The 6-bit octal pattern is repeated six times across the main bus, then it replaces the C(B) under main engine zone control.

Execution. See Figure 18.

Precondition Control. Bit 6 is used as part of the 6-bit octal data pattern. Bit 12 set to a one causes the unzoned part of B to be set to zeros.

LIC-LOAD C

0 5	6 11	12	13 17
15	DATA	Z	TOP

Description. The 6-bit octal pattern in bits 6-11 is repeated six times across the main bus, then it replaces C(C) under main engine zone control.

Execution. See Figure 18.

Precondition Control. Bit 6 is used as part of the 6-bit octal data pattern. Bit 12 set to a one causes the unzoned part of C to be set to zeros.

LID-LOAD D

0 5	6 11	12	13 17
17	DATA	Z	TOP

Description. The 6-bit octal pattern in bits 6-11 is repeated six times across the main bus, then it replaces C(D) under main engine zone control.

Execution. See Figure 18.

Precondition Control. Bit 6 is used as part of the 6-bit octal data pattern. Bit 12 set to a one causes the unzoned part of D to be set to zeros.

MOPB-MOP REGISTER TO B

0 5	6 17
10	DATA

Description. Bits 18-23 of B are replaced by bits 0-5 of the instruction. The 12-bit octal data pattern in bits 6-17 of the instruction replace bits 24-35 of B. The most significant half, i.e., bits 0-17, of B are set to zeros.

Execution. Octal 00000010XXXX will replace C(B). The XXXX field is variable and defined by the 12-bit pattern in the instruction. Refer to Figure 18.

Precondition Control. Not effective.

MOPC-MOP REGISTER TO C

0 5	6 17
11	DATA

Description. Bits 18-23 of C are replaced by bits 0-5 of the instruction. The 12-bit octal data pattern in bits 6-17 of the instruction replace bits 24-35 of C. The most significant half of C, i.e., bits 0-17 are set to zeros.

Execution. Octal 00000011XXXX will replace the C(C). The XXXX field is variable and defined by the 12-bit pattern in the instruction. Refer to Figure 18.

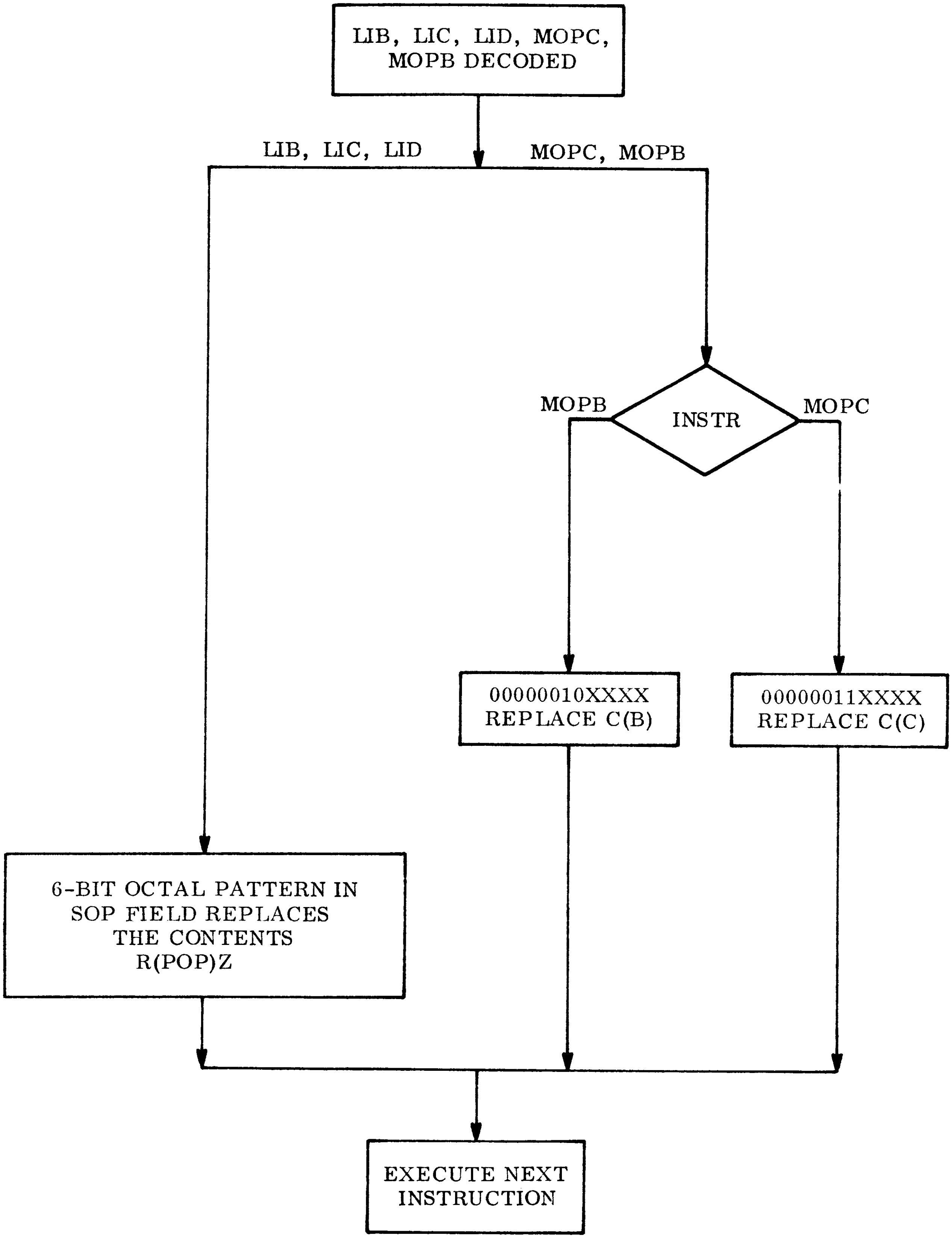

LIB, LIC, LID, MOPC, MOPB DECODED
LIB, LIC, LID
MOPC, MOPB
MOPB
INSTR
MOPC
00000010XXXX REPLACE C(B)
00000011XXXX REPLACE C(C)
6-BIT OCTAL PATTERN IN SOP FIELD REPLACES THE CONTENTS R(POP)Z
EXECUTE NEXT INSTRUCTION

Figure 18. LIB, LIC, LID, MOPC, MOPB Flow Chart

CONTROL INSTRUCTIONS

Instructions which govern the flow of a program, and in particular those which cause an alteration in the computer's normal process of taking its instructions from sequential locations, are called control instructions.

Uncondition transfer instructions specify the location "Y" from which the computer is to take the next instruction. Conditional transfer instructions also specify a location Y. However, whether the computer takes its next instruction from Y or the next sequential location depends upon the outcome of a test. This test is specified by the primary operation code (POP) of the instruction.

Test instructions are similar to conditional control instructions in that they cause some test to be performed. Unlike conditional instructions, however, test instructions do not specify a location Y to which control may be transferred. Instead the alternative location to which control may be transferred is fixed relative to the location of the test instruction. This is referred to as the "skip distance" and is added to or subtracted from the MINIFLOW program sequence counter.

NOP-NO OPERATION

0	5
01	//////////////////////////////////////

Description. This instruction causes the computer to take next instruction in sequence. Bits 6 through 17 arc not examined. NOP is used to fill in for deleted instructions and to provide instruction alignment.

HALT-STOP CLOCK

0	5
00	//////////////////////////////////////

Description. This instruction causes the computer to halt. The program sequence counter contains the location of the next sequential instruction. When the start key on the operator's console is depressed, the computer proceeds and executes the next sequential instruction The exception to this is if a HALT is executed at an odd mini-address, the machine will not start again because of an interrupted mini-fetch control memory cycle.

TRU-TRANSFER

0 5	6	7 17
16	///	Y

Description. Issuing a TRU causes the computer to take its next address from MINIFLOW address specified by Y and proceed from there. Bits 7-17 of the instruction replace the C(RB), resulting in a transfer of control.

Execution. MINIFLOW program sequence is unconditionally changed by a TRU.

Precondition Control. Bit 6 is not used.

SMCT-STORE MINI COUNT AND TRANSFER

0 5	6	7 17
76	0	Y

Description. This instruction causes C(RB)+1 to replace the C(RD). The subroutine mode is set, and C(Y) replace C(RB), transferring control to Y.

Execution. Refer to Figure 19. MINIFLOW program sequence is unconditionally changed by a SMCT. The execution of a SMCT causes the previous C(RB) to be incremented by one and replace C(RD), to be used at the next EXIT condition as a return address. The subroutine mode is entered so that the next EXIT will cause a return to the instruction following the SMCT. When not in the subroutine mode, an exit will return control to the scheduler. If already in the subroutine mode when a SMCT is issued, the machine stays in the subroutine mode and the previous C(RD) are replaced by C(RB)+1.

Precondition Control. Bit 6 must be zero.

Affected Conditions. The computer will be placed in the subroutine mode.

ARITHMETIC TEST INSTRUCTIONS

Several machine conditions are tested by the test instructions. The test criteria is specified by the SOP field, bits 7-11, T(SOP). Bit 6 will specify the polarity of the test. Zero sets the Test Satisfied (TSAT) flip-flop on if the test condition is false. One sets the TSAT flip-flop on if the test condition is true.

The skip distance is specified by the TOP field, bits 13-17. The state of bit 12 specifies the direction of the skip. Zero indicates a positive, i.e., forward skip distance. One indicates a negative, i.e., backward skip distance.

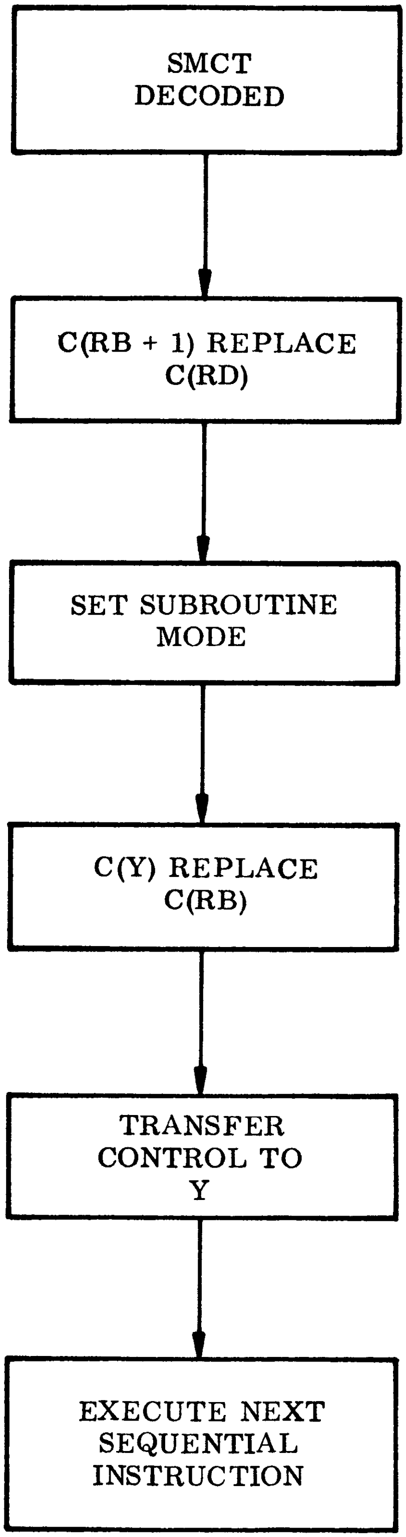


Figure 19. SMCT Flow Chart

The arithmetic test SOPs shown in Table 25 are used in conjunction with the arithmetic test instructions.

Table 25. Arithmetic Test SOPs

SKIP	ON FALSE	SKIP	ON TRUE		
CODE	SOP	CODE	SOP	NOTE	TEST CRITERIA
00	NO (No Skip)	40	YES (Skip)		Unconditional
01	CF	41	CAR	1	Carry/Borrow from zoned field
02	CX11F	42	CX11	1	Carry or Else GOP 11
03	G11F	43	G11		GOP 11 (Precondition)
04	NZ	44	Z	1	Zero in Zoned Field
05	N910F	45	N910		GOP 10 and not GOP 9 (Precond.)
06	G9F	46	G9		GOP 9 (Precondition)
07	G10F	47	G10		GOP 10 (Precondition)
10	RNZ	50	RZ	2	Zero in RE Register (Bits 3-10)
11	FCF	51	FC	3	First Borrow in ALG POP
12	GINF	52	GIN		GIN (Precondition)
13	FOFF	53	FOF	4	FMQ OR FAC (Floating overflow)
14	(skip)	54	(no skip)		
15	LSF	55	LS		Like signs in B and C
16	AQF	56	AQ		AC Q BIT
17	(skip)	57	(no skip)	7	
20	ZX11F	60	ZX11	1	Zero or else GOP 11
21	MSBF	61	MSB	1	Bit 0 of PE (most significant bit)
22	LSBF	62	LSB	1	Bit 35 of PE (least sign. bit)
23	EQUF	63	EQU	5	PC Equal to PB (Q&SIGN Logic)
24	LESSF	64	LESS	6	PC Less than PB (Q&SIGN Logic)
25	AOVF	65	AOV		AC Overflow On
26	ACP	66	ACM		AC Sign (Minus or Plus)
37	(Skip)	77	SAT	8	Previous test satisfied.

NOTES

1. CARRY (or BORROW) out of the zoned field of a main engine result, ZERO test of the zoned field of a main engine result, Bit position 0 and bit position 35 of an engine result are all stored for testing on any of the following POP's: PB, PC, PD, PE, PBD, PCD, ALG, IC, XR, RC, SHIFT and CMI (write data only).

2. Zero test of the mini engine result is stored on a MINI POP.

3. First carry test is stored on the trial subtraction of an ALG POP. It will be set on if the signs are not alike and there is a carry out of the zoned field, but it will be reset if the AC Q bit is on and the zone includes the Q bit.

4. This tests the logical OR of general indicator FMQ and FAC (floating overflow).

5. The EQUALITY test should be primed by an engine operation (SOP B-C, C-B, BEC, etc.) to set the zero flip-flop; it will test true if the AC Q bit is reset and the ZERO test is true and either the signs are alike or ARI is off.

6. The LESS test should be primed by an engine operation. (SOP B-C) to test if the AC (B) is less than C. The following table gives the conditions when the LESS test is true.

ARI FLIP-FLOP	CARRY TEST	AC Q BIT	B SIGN	C SIGN	ZERO TEST
OFF	TRUE	OFF	X	X	X
ON	FALSE	X	MINUS	X	FALSE
ON	X	X	MINUS	PLUS	X
X	TRUE	OFF	X	MINUS	X

However the LESS test is false if ARI is on, AC Q bit is one, and the B sign is minus.

7. Undefined tests will be false.

8. Does not alter previous state of TSAT, all other SOPs may alter the previous state of TSAT.

TA-TEST AND SKIP

0 5	6	7 11	12	13 17
61	P	SOP	D	TOP

Description. If T(SOP) is satisfied, set TSAT flip-flop and the computer skips the distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the computer will execute the next instruction in sequence.

Execution. See Figure 20.

Precondition Control. Bit six specifies the polarity of the test and bit 12 specifies the direction of the skip.

Affected Conditions. The TSAT flip-flop.

TAE-TEST AND SKIP OR EXIT

0 5	6	7 11	12	13 17
65	P	SOP	D	TOP

Description. If T(SOP) is satisfied, set TSAT flip-flop and the computer skips the distance specified by C(TOP) in direction indicated by bit 12. Otherwise an exit will occur.

Execution. See Figure 20.

Precondition Control. Bit 6 specifies polarity of test and bit 12 specifies the direction of skip.

Affected Conditions. The TSAT flip-flop.

TAW-TEST AND TRANSFER TO WIRED-IN-SEQUENCE

0 5	6	7 11	12	13 17
75	P	SOP	D	TOP

Description. If T(SOP) is satisfied, set TSAT flip-flop, otherwise TSAT flip-flop is reset. If bit 12 is set to a one, add C(RC) to C(RB). If bit 12 is set to zero, subtract C(RC) from C(RB). Program control will then transfer to wired-in-sequence state YA5.

Execution. See Figure 20.

Precondition Control. Bit 6 specifies polarity of test and bit 12 specifies direction of skip.

Affected Conditions. The TSAT flip-flop.

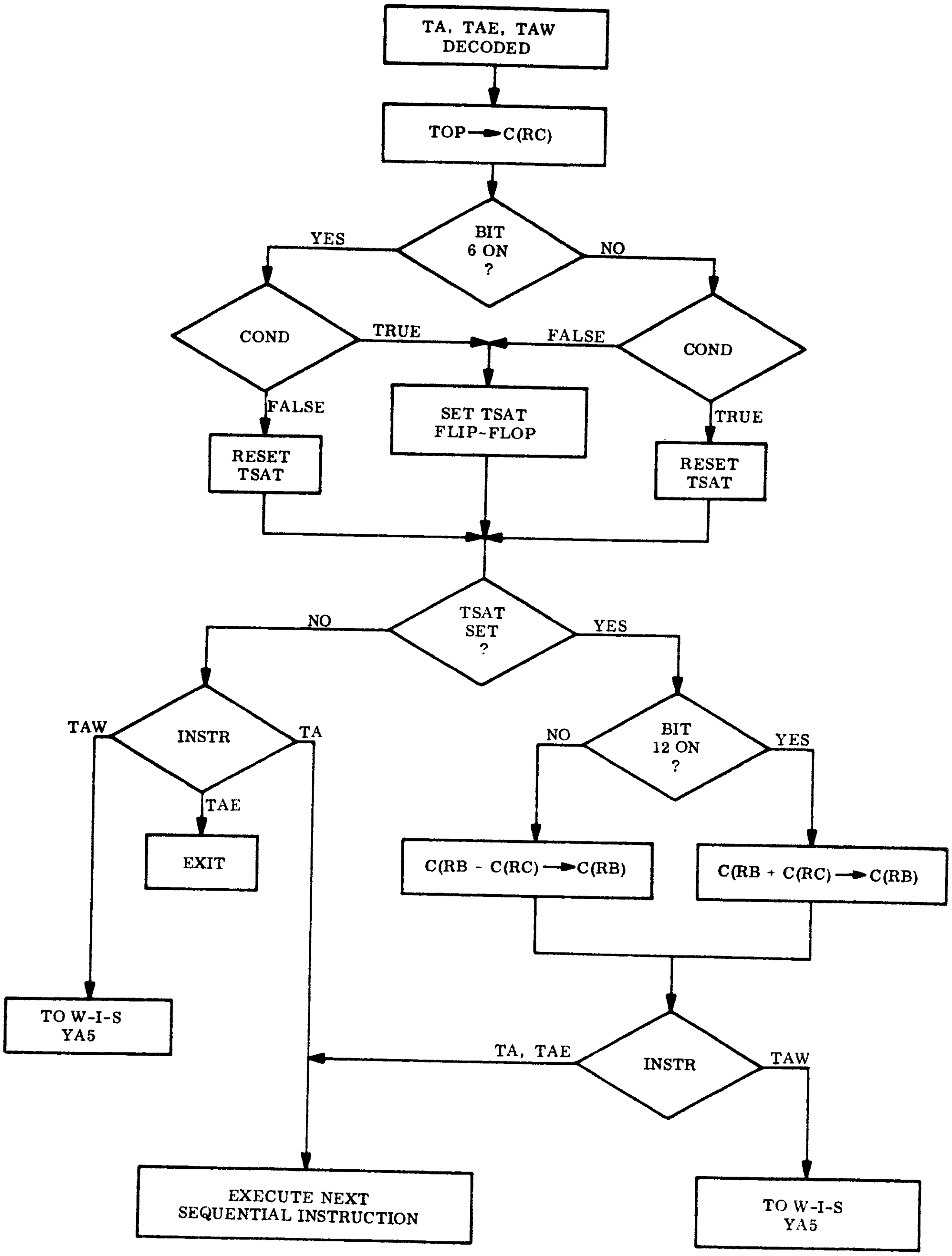


Figure 20. TA, TAE, TAW Flow Chart

GENERAL CONDITIONAL TEST INSTRUCTIONS

A set of fifty-eight general and special purpose indicator flip-flops may be tested and set or reset by the general conditional test instructions.

The test condition is specified by the SOP field, bits 6-11, which defines a flip-flop indicator, I(SOP). Some flip-flops have hardware defined functions, see Table 26, and the remaining flip-flops may be used for functions defined by the programmer.

The skip distance is specified by the TOP field, bits 13-17. The state of bit 12 (0,1) specifies the direction of the skip. Zero indicates a positive, i.e., forward skip distance. One indicates a negative, i.e., backward skip distance. Bits 13-17 always replace the C(RC). C(RC) are added to C(RB) or subtracted from C(RB), if the TSAT flip-flop is set on and the skip is taken. Otherwise, the TSAT flip-flop is turned off and the computer executes the next instruction in sequence.

The general indicator test SOPs are shown in Table 26.

Table 26. General Indicator Test SOPs

SOP CODE	SOP MNEMONIC	PURPOSE	CONSOLE LAMP	REMARKS
00	SSW	Sense Switch Tests		See Note 1
04		General Purpose		
05		General Purpose		
06	EOFA	Channel A END OF FILE	YES	See Note 2
07		General Purpose		
10		General Purpose		
11		General Purpose		
12		General Purpose		
13		General Purpose		
14	EOFB	Channel B END OF FILE	YES	See Note 2
15		General Purpose		
16	DCTM	Divide Check Trap Mode (a maintenance console switch)		
17		General Purpose		
20		General Purpose		
21		General Purpose		
22	TCEB	Channel B tape check enable	YES	See Note 3
23		General Purpose		
24	CTEB	Channel B Command trap enable	YES	See Note 2
25		General Purpose		
26		General Purpose		
27		General Purpose		

Table 26. General Indicator Test SOPs (Cont)

SOP CODE	SOP MNEMONIC	PURPOSE	CONSOLE LAMP	REMARKS
30	DCK	Divide Check	YES	
31		General Purpose		
32	TCEA	Channel A tape check enable	YES	See Note 3
33		General Purpose		
34	CTEA	Channel A Command Trap	YES	See Note 2
35		General Purpose		
36	IOC	I/O check	YES	
37		General Purpose		
40	TRAP	Transfer trap mode	YES	See Note 4
41		General Purpose		
42	DIS	DISPLAY		See Note 5
43		General Purpose		
44	TCKA	Channel A tape check error	YES	See Note 3
45		General Purpose		
46	TCKB	Channel B tape check error	YES	See Note 3
47		General Purpose		
50	SL4	Sense Light 4	YES	
51		General Purpose		
52	SL1	Sense Light 1	YES	
53		General Purpose		
54	SL2	Sense Light 2	YES	
55		General Purpose		
56	SL3	Sense Light 3	YES	
57		General Purpose		
60	MTM	MULTIPLE TAG MODE	YES	See Note 6
61		General Purpose		
62	FMQ	MQ Factor exceeded		See Note 7
63		General Purpose		
64	FPO	Floating Overflow		See Note 7
65		General Purpose		
66	FAC	AC Factor Exceeded		See Note 7
67	----	(Not Available)		See Note 10
70	TCN	Trap control	YES	
71		General Purpose		
72	PDCK	Predivide Check		See Note 8
73	----	(Not Available)		See Note 10
74	CON	Console Request		See Note 9
75		General Purpose		
76	CIF	Current Instruction OFF (a console display switch)		

NOTES

1. The Console Sense Switch to be tested is specified by bits 0-2 of the D register at the time of testing. Octal 0 and 7 will always test OFF (false); Octal 1 to 6 will test the position of the corresponding Console Sense Switch.

2. The End of File General Indicator is automatically turned off and the TSAT flip-flop set on during wired-in-sequence under these conditions: The End of File General Indicator is on and the Command Trap Enable (for the corresponding Channel) if OFF and an object instruction ±003X (TRANSFER ON END OF FILE) is translated (with Channel A corresponding to +003X and Channel B corresponding to -003X).

3. The Tape Check General Indicator is automatically turned off and the TSAT flip-flop set on during wired-in-sequence under these conditions: The Tape Check General Indicator is on and the Tape Check Enable (for the corresponding channel) if OFF and an object instruction ±0022 (TRANSFER ON REDUNDANCE CHECK) is translated (with Channel A corresponding to +0022 and Channel B corresponding to -0022. Also, provided that the group is defined as a fast transfer. On the 7044 emulator, op codes +0022 and -0022 are programmer controlled).

4. The Transfer Trap Mode is enabled when this flip-flop is on. Section III discusses the control exercised by this flip-flop.

5. The Display flip-flop is also set on automatically whenever the display register is loaded. The following events will cause the display to be loaded:

   a. A register (AC, MQ, SI, R4, IC or one of the XR's) is altered and the corresponding register's console display switch is down (ON).

   b. An object instruction is fetched for emulation and the current instruction switch is down (ON).

   c. A memory cell is altered and the corresponding memory's console display switch is down (ON) and the address used for the write cycle is the same as is held in the console address keys.

6. The multiple Tag Mode is enabled when this flip-flop is on. Section II discusses the control exercised by this flip-flop.

7. These Floating Point Exception Indicators also may conditionally be set on by two SOPS of the MISC POP (as explained earlier) the LOGICAL OR of FMQ and FAC will be tested by the FOF SOP of the TA (TAE, TAW) POP. The conditions to set these flip-flops by a MISC POP are illustrated in the following table.

GENERAL INDICATOR Turned On	MISC SOP	ENGINE CONDITION
FAC	FOFA	MSB Test true
FMQ	FOFQ	MSB Test true
FPO	FOFA or FOFQ	MSB Test true and
(if FPO set, then FAC or FMQ also set)		Engine Q bit false

8. The predivide check flip-flop is also turned on when a SHIFT POP with a Divide SOP (10 or 30) does not cause a borrow out on the first trial subtraction of the divide.

9. The Console Request flip-flop is also turned on whenever one of the console interrupt switches is pressed. These switches are: Reset, Clear, Enter Keys, Display, Load Cards and Load Tape.

10. SOPS corresponding to non-existant General Indicators will always test false.

TG-SKIP IF TEST SATISFIED; OTHERWISE, EXECUTE NEXT SEQUENTIAL INSTRUCTION

0 5	6 11	12	13 17
60	SOP	D	TOP

Description. If I(SOP) is on, set the TSAT flip-flop and skip the distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the computer will execute the next instruction in sequence.

Execution. See Figure 21.

Precondition Control. Bit 12 specifies the direction of skip.

Affected Conditions. The TSAT flip-flop.

TGE-SKIP IF TEST SATISFIED; OTHERWISE, EXIT

0 5	6 11	12	13 17
64	SOP	D	TOP

Description. If I(SOP) is on, set TSAT flip-flop and skip the distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the instruction will EXIT.

Execution. See Figure 21.

Precondition Control. Bit 12 specifies the direction of the skip.

Affected Combinations. The TSAT flip-flop.

TGF-SKIP IF TEST NOT SATISFIED

0 5	6 11	12	13 17
71	SOP	D	TOP

Description. If I(SOP) is off, set TSAT flip-flop and skip distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the computer will execute the next instruction in sequence.

Execution. See Figure 21.

Precondition Control. Bit 12 specifies direction of skip.

Conditions Affected. The TSAT flip-flop.

TGS-LIKE TG BUT ALWAYS SETS I(SOP) ON.

0 5	6 11	12	13 17
70	SOP	D	TOP

Description. If I(SOP) is on, set the TSAT flip-flop and skip the distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the computer executes the next instruction in sequence. I(SOP) is unconditionally set on after being tested.

Execution. See Figure 21.

Precondition Control. Bit 12 specified the direction of the skip.

Affected Conditions. I(SOP) is always set on and the TSAT flip-flop is either set or reset.

TGR-LIKE TG BUT ALWAYS SETS I(SOP) OFF.

0 5	6 11	12	13 17
74	SOP	D	TOP

Description. If I(SOP) is on, set TSAT flip-flop and skip the distance specified by C(TOP) in direction indicated by bit 12. Otherwise, the computer executes the next sequential instruction. I(SOP) is unconditionally set off after being tested.

Execution. See Figure 21.

Precondition Control. Bit 12 specifies the direction of the skip.

Affected Conditions. I(SOP) is always set off and the TSAT flip-flop is either set or reset.

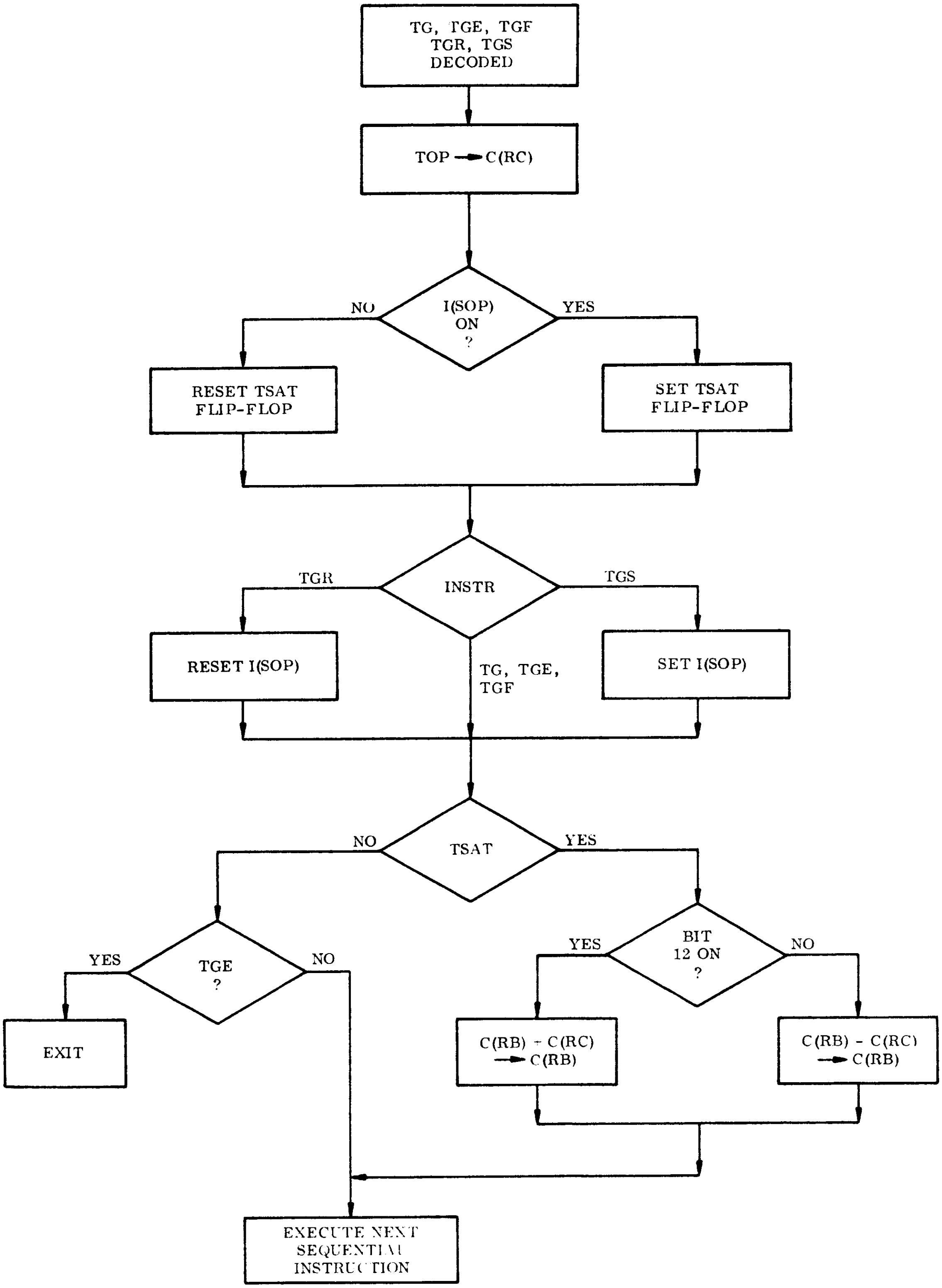


Figure 21. TG, TGE, TGF, TGR, TGS Flow Chart

MISC-SCHEDULER AND AC SIGN AND Q BIT CONTROL

0 5	6 11	12	13 17
06	SOP	E	/////////////

Description. This instruction causes the operation specified by the C(SOP), Table 27, to be performed.

Execution. Control over many functions of the scheduler operation and level control are performed by this instruction. Also, it is used to set, reset and toggle the AC sign and Q bit. The control action is specified by the Miscellaneous Control SOPs shown in Table 27.

Table 27. THE MISCELLANEOUS CONTROL SOPS

CODE	MNEMONIC	NOTE	CONTROL ACTION
10	RXI		Reset external interrupt
12	RTI		Reset Interval timer interrupt request
16	RTER	1	Reset CHANNEL terminate request
20	RAQ		Reset AC Q bit (also reset Engine Q bit)
21	SAQ		Set AC Q bit
22	TAQ		Toggle AC Q bit (also store in Engine Q bit)
23	FOFA	2	Conditionally set AC Floating Overflow
24	RAS		Reset AC Sign bit (plus)
25	SAS		Set AC Sign Bit (minus)
26	TAS		Toggle AC sign bit
27	FOFQ	2	Conditionally set MQ Floating Overflow
30	ROV		Reset AC Overflow
31	SOV		Set AC Overflow
33	RSQ		Reset AC sign and Q bit and Engine Q bit
37	ACK	1	Acknowledge last buffer request
42	SR1	1	Set scheduler Buffer Service Request Level 1 or 2
46	SR3		Set Scheduler Terminate request (level 3)
50	SRT	3	Set Channel Trap request (level 4)
52	SR4		Set Program Request (program level 4)
54	HANG	4	Hang the present level
56	POST	5	Postpone a trap request
60	HALT	4	Set program Halt and Hang (level 4)
62	RH1	1	Reset Buffer Service Hang (level 1 or 2)
66	RH3		Reset terminate Hang (level 3)
70	RH4		Reset Program Hang unless Halt is set (level 4)
72	RCN	6	Reset all console interrupt requests
74	RESL4	7	Reset trap and program (level 4) .
76	RINT	8	Reset interrupt (levels 1, 2, and 3)

NOTES

1. The Channel B F/F determines whether level 1 (Channel A) or level 2 (Channel B) is selected.

   RTER (code 16) resets the terminate request held in channel flip-flop.

ACK (code 37) counts down by one the number of outstanding buffer service requests; up to four may be safely stacked. But a fifth one causes a transfer timing error due to buffer spill (read) or buffer depletion (write.)

2. FOFA and FOFQ will set the General Indicators FAC, FMQ, and FPO based on the conditions described under General Conditional Test instructions.

3. A TRAP request will not be granted by the scheduler if the manual mode console switch is on (down).

4. A HANG will start a save Sequence and save the RB, RD, D registers and the Subroutine Mode Control flip-flop for later restoration. The Hung level will be locked out until the Hang is reset by MINIFLOW or the console reset key, if the hang is reset by MINIFLOW then the Return Sequence will be taken. A HALT should only be given in level 4 where it will hang level 4 and turn on the Console Halt Light.

5. Using the POST SOP will inhibit a trap request until one more main memory object instruction has been executed. However, if bit 12, the EXIT bit, is set to a one, the scheduler may select the trap request before the POST SOP has time to postpone one more instruction.

6. Reset Console Requests for: Reset, Clear, Load Card, Load Tape, Enter Keys, and Display.

7. Reset level 4 scheduler control flip-flop, i.e., Trap Request, Program Request, Job-in-Progress, Return and Hang.

8. Reset the Request, Job-in-Progress, Return and Hang flip-flops for levels 1, 2 and 3. Only a Terminate Request is reset in level 3, not a Console Request; also the Hang flip-flop for level 4 is reset if the Halt indicator is off.

An EXIT instruction is provided by using an undefined SOP with the MISC POP and setting bit 12 to a one. The undefined SOP causes the instruction to act as a No-Operation. An example of such an EXIT instruction would be 064440.

A Return Sequence is taken after a HANG by resetting the Hang flip-flop for the Hung level while in another level. An exception to this is if HALT is issued in level 4 the Hang 4 flip-flop cannot be reset except by one of the following three methods:

a. Pressing the start key resets the Hang flip-flop, activates a Return Sequence and resets the Halt indicator.

b. Executing a MISC SRT will set a trap request during the execution of a program. When the trap request is granted, the return sequence for this particular incident is aborted. A new program request is then set.

The only way to set a trap request is with an SRT. Trap request effective if JIP4 and postpone is off or if Hang 4 is on. Also only effective if not in manual or an I/O data operation is not being performed.

c. Reset Level 4 (SOP 74) followed by a Program Request (SOP 52) will start the program level with the next object instruction pointed to by the IC. If AUX1 or AUX2 is set, their contents will determine the next instruction to be executed, rather than the IC.

Precondition Control. Bit 6 is defined as being part of the SOP field in this instruction. The instruction will exit if bit 12 is set to a one.

Affected Conditions. The flip-flops specified by the SOP field.

DELAY-DELAY PRESENT LEVEL.

0 5	6	7 11	12	13 17
54	//		0	//////////////

Description. This instruction causes a delay to be executed by the computer when used in conjunction with any main engine SOP.

Execution. When DELAY is executed in level 1, 2 or 3, the SAVE sequence will be entered and control will return to the scheduler to scan for other re requests. The level issuing the delay is locked out for approximately 75 microseconds, after which a return request is set in the scheduler. When it is granted, a Return sequence will be entered and the previously interrupted program will continue as the C(RB), C(RD), C(D) and the subroutine mode control status are all restored from the predefined save area calls for the specified level.

When a DELAY or a Hang is issued in either level 1 or 2, both levels 1 and 2 will be locked out of the scheduler until the return is made and a normal EXIT is taken.

This instruction is primarily used to allow temporary execution in another level to simulate atonomous CPU and data channels.

A DELAY should not be issued in level 4 because a return request will not be set.

Precondition Control. Bit 6 is not used. Bit 12 must be zero

Affected Conditions. The predefined SAVE area in control memory for the level being delayed.

PRE-PRECONDITION CONTROLS

0 5	6 7	8 9	10 11	12 13 14 15 16 17
12	/////	Control	//////	TOP

Description. This instruction causes the precondition indicators specified by C(TOP) and bits 8 and 9 to be effective.

Execution. A one bit in the TOP field, bits 12-17 means the corresponding indicator is to be altered as indicated by bits 8 and 9. A zero bit in the TOP field means that the corresponding indicator is not to be altered. The Mnemonic and Purpose for bits 12-17 are shown below.

BIT	MNEMONIC	PURPOSE
12	GOP9	General Operation number 9
13	GOP10	General Operation number 10
14	GOP11	General Operation number 11
15	GEX	General Half Exchange on bus
16	GIN	General Inhibit result to Engine Register
17	ARI	Operate with signs

Bits 8 and 9 indicate the control to be performed on the precondition indicates as shown below.

BIT	MNEMONIC	PURPOSE
0	X	No change
1	0	Turn off specified flip-flops
1	1	Turn on specified flip-flops

Precondition Control. This instruction is a special case as previously described.

Affected Conditions. The specified precondition indicators.

I/O CHANNEL CONTROL INSTRUCTIONS

There is a set of 31 channel indicators for each channel (A or B). Each indicator is tested and set or reset by the I/O channel control instructions. Some of the channel indicators have hardware defined functions, shown in Table 28. The remaining indicators may be used for any function defined by the programmer.

Each instruction has the following format: The POP field, bits 0-5, specify the operation to be performed. Bit 6 must be zero. The SOP field, bits 7-11, and the channel B flip-flop specify the channel indicator to be tested and set or reset. The abbreviation for this is CI(SOP).

If the channel B flip-flop is off the SOP field specifies indicators associated with channel A. The channel B indicators are specified if the channel B flip-flop is on.

Bit 12 specifies the direction of skip A zero specifies a positive, i.e., forward skip. A one specifies a negative, i.e., backward skip. The TOP field, bits 13-17, always contain the number of instructions the computer skips. The C(TOP) always replaces the C(RC). If the test is satisfied, the C(RC) are added to or subtracted from C(RB), the program sequence counter, depending on the state of bit 12. If the test is not satisfied, plus one is added to C(RB) and the computer executes the next sequential instruction.

The channel indicator test SOPs and their purpose are shown in Table 28.

Table 28. Channel Indicator Test SOPs

SOP CODE	SOP MNEMONIC	PURPOSE	CONSOLE LAMP	NOTE
00	ER1	Tape Parity Error		Note 1
01	ER2	Transfer Timing Error		Note 2
02		General Purpose		
03		General Purpose		
04		General Purpose		
05		General Purpose		
06		General Purpose		
07		General Purpose		
10	BOT	Beginning of tape	YES	Note 3
11	EOT	End of Tape	YES	Note 3
12		General Purpose		
13		General Purpose		
14		General Purpose		
15		General Purpose		
16		General Purpose		
17		General Purpose		
20	CEF	Channel End of File		Note 4
21	EOR	End of Record		Note 4
22		General Purpose		
23		General Purpose		
24		General Purpose		
25		General Purpose		
26	DSP	Data Select in Process	YES	
27	IOP	I/O in Process	YES	
30	TNR	Tape not Ready		Note 5
31	----	(Not Available)		
32		General Purpose		
33		General Purpose		
34		General Purpose		
35		General Purpose		
36	GHB	Channel B Flip-Flop		
37	COP	Card or Printer (Channel A)		

NOTES

1. PARITY ERRORS are sensed by the channel during either a read or write operation. Any CH1 POP will turn the indicator associated with these errors off.

2. A TRANSFER TIMING ERROR sensed by the channel will cause indicator 01 to be set. The TCS instruction will not set this indicator and TCR will not reset it. The indicator will be reset by a CH1 instruction.

3. BEGINNING OR END OF TAPE sensed by the channel and either indicator 10 or 11 are set. The indicator will be reset by a CH1 instruction. If the tape is at load point and the BOT indicator is turned off, it will be turned back on if a backspace tape or rewind is issued. In this case, the tape will not move from the load point.

4. END OF FILE OR RECORD conditions set either indicator 20 or 21. These indicators are not set or reset by MINIFLOW.

5. TAPE NOT READY condition sets indicator 30. This indicator is not set or reset by MINIFLOW.

6. CHANNEL B Flip-Flop is tested by SOP 36. There are actually three Channel B Flip-Flops:

   a. One flip-flop is sensed in, and controls within levels 1 and 2. It is always turned off for level 1 (Channel A Buffer Service) and turned on for level 2 (Channel B Buffer Service).

   b. A second flip-flop is sensed in and controls within level 3. It is always turned off for a console interrupt or a timer interrupt or a Channel A terminate but is turned on for a Channel B terminate.

   c. The third flip-flop is only sensed in and controls within level 4. It is always turned off for a trap request or a program request; this level 4 Channel B flip-flop may be set on by MINIFLOW while in any level. It may not be reset by MINIFLOW.

   The status of the Channel B flip-flop is preserved on a level basis during Hangs or delays by this separation.

7. COP is an Indicator in Channel A only and will always test false if the Channel B flip-flop is on. It may be tested and turned on or off when the Channel B flip-flop is off.

TC-SKIP IF CI(SOP) IS ON

0 5	6	7 11	12	13 17
20	0	SOP	D	TOP

Description. If CI(SOP) is on, the test is satisfied, and the TSAT flip-flop is set. The computer skips the number of instructions specified by C(TOP) in the direction indicated by bit 12. If CI(SOP) is off, the computer will execute the next sequential instruction.

Execution. See Figure 22.

Precondition Control. Bit six must be zero. Bit 12 indicates direction of skip.

Affected Conditions. The TSAT flip-flop.

TCE-SKIP IF CI(SOP) IS ON; OTHERWISE, EXIT

0 5	6	11	12	13 17
24	0	SOP	D	TOP

Description. If CI(SOP) is on, the test is satisfied and the TSAT flip-flop is set. The computer skips the number of instructions specified by C(TOP) in direction indicated by bit 12. The instruction will EXIT if CI(SOP) is off.

Execution. See Figure 22.

Precondition Control. Bit 6 must be zero. Bit 12 specifies the direction of the skip.

Affected Conditions. The TSAT flip-flop.

TCF-SKIP IF CI(SOP) IS OFF; OTHERWISE EXECUTE NEXT INSTRUCTION.

0 5	6	7 11	12	13 14
31	0	SOP	D	TOP

Description. If CI(SOP) is off, the test is satisfied, and the TSAT flip-flop is set. The computer skips the number of instructions specified by C(TOP) in direction indicated by bit 12. If CI(SOP) is on, the computer will execute the next instruction in sequence.

Execution. See Figure 22.

Precondition Control. Bit 6 must be zero. Bit 12 specifies the direction of shift.

Affected Conditions. The TSAT flip-flop.

TCS-SKIP IF CI(SOP) IS ON. CI(SOP) IS TURNED ON AFTER TESTING.

0 5	6	7 11	12	13 17
30	0	SUP	D	TOP

Description. If CI(SOP) is on, the test is satisfied. The computer skips the number of instructions specified by the C(TOP) in direction indicated by bit 12. If CI(SOP) is off, the computer will execute the next sequential instruction. CI(SOP) is unconditionally turned on after testing.

Execution. See Figure 22.

Precondition Control. Bit 6 must be zero. Bit 12 specifies the direction of the skip.

Affected Conditions. The CI(SOP) is turned on. The TSAT flip-flop.

TCR-SKIP IF CI(SOP) IS ON. TURN OFF CI(SOP) AFTER TESTING.

0 5	6	7 11	12	13 17
34	0	SOP	D	TOP

Description. If CI(SOP) is on, the test is satisfied. The computer skips the number of instructions specified by C(TOP) in direction indicated by bit 12. The computer will execute the next sequential instruction if CI(SOP) is off. The CI(SOP) is unconditionally turned off after testing.

Execution. See Figure 22.

Precondition Control. Bit 6 must be zero. Bit 12 specifies the direction of skip.

Affected Conditions. The CI(SOP) is turned off. The TSAT flip-flop.

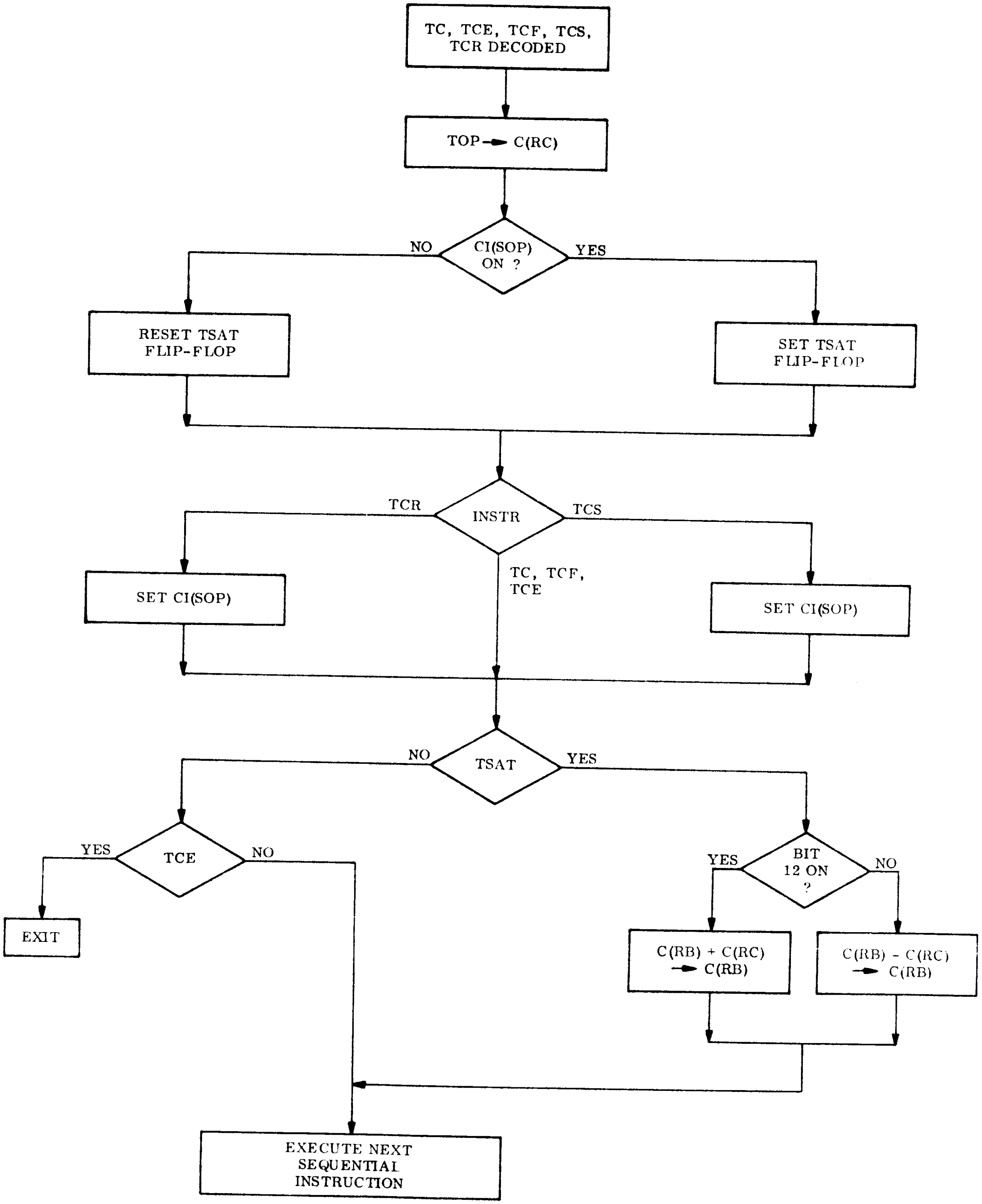
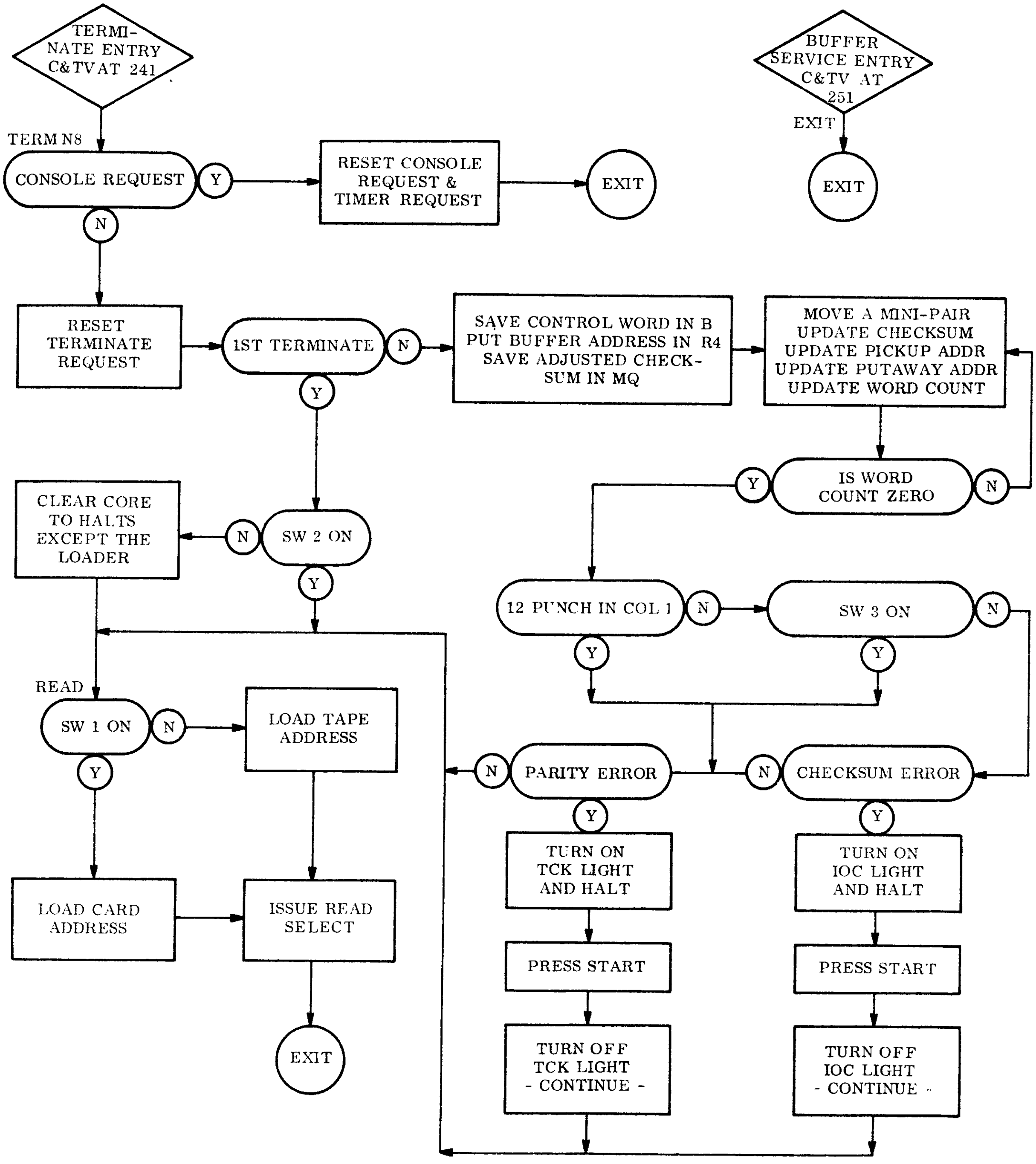


Figure 22. TC, TCE, TCF, TCS, TCP Flow Chart

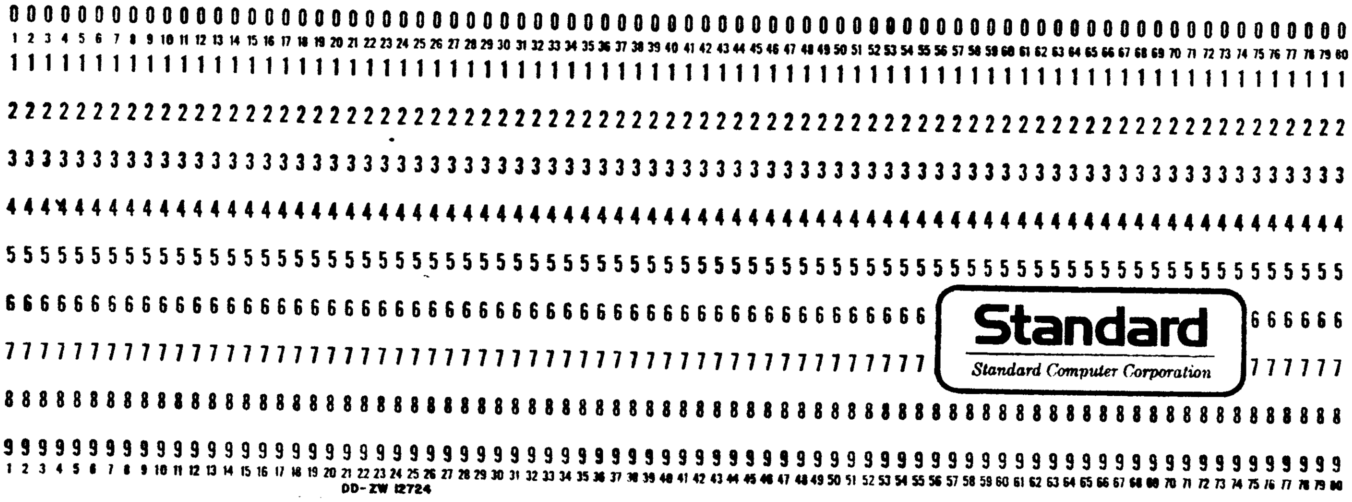
A BOOTSTRAP LOADER FOR BINARY CARDS

Figure 23 is a flowchart for a binary card bootstrap loader.



TERMI-NATE ENTRY C&TVAT 241
TERM N8
CONSOLE REQUEST
Y
N
RESET CONSOLE REQUEST & TIMER REQUEST
EXIT
BUFFER SERVICE ENTRY C&TV AT 251
EXIT
EXIT
RESET TERMINATE REQUEST
1ST TERMINATE
N
Y
SAVE CONTROL WORD IN B PUT BUFFER ADDRESS IN R4 SAVE ADJUSTED CHECK-SUM IN MQ
MOVE A MINI-PAIR UPDATE CHECKSUM UPDATE PICKUP ADDR UPDATE PUTAWAY ADDR UPDATE WORD COUNT
IS WORD COUNT ZERO
Y
N
CLEAR CORE TO HALTS EXCEPT THE LOADER
SW 2 ON
N
Y
12 PUNCH IN COL 1
N
Y
SW 3 ON
N
Y
READ
SW 1 ON
N
Y
LOAD TAPE ADDRESS
PARITY ERROR
N
Y
CHECKSUM ERROR
N
Y
TURN ON TCK LIGHT AND HALT
TURN ON IOC LIGHT AND HALT
LOAD CARD ADDRESS
ISSUE READ SELECT
PRESS START
PRESS START
EXIT
TURN OFF TCK LIGHT - CONTINUE -
TURN OFF IOC LIGHT - CONTINUE -

Figure 23. Flow Chart of a Bootstrap Loader



BOOT STRAP CARD IMAGE

```
        ORG       0/200         EXAMPLE 27   BOOTSTRAP
* BINARY WORD BOOTSTRAP LOADER FOR VERSION 9 CPL *
*
* BINARY CARD FORMAT-
* COLUMN 1,12 PUNCH- BYPASS CHECKSUM FOR THIS CARD
* COLUMN 2,12 TO 3- NUMBER OF MINI-PAIRS TO LOAD
* COLUMN 3, 0 TO 9- WORD ADDRESS OF 1ST MINI-PAIR
* COLUMNS 4 TO 6- 36 BIT 1'S COMPLEMENT CHECKSUM
* COLUMNS 7 TO 72- UP TO 22 MINI-PAIRS IN BINARY
* COLUMNS 73 TO 80- NOT EXAMINED, MAY BE SEQUENCE
*
* SWITCH 1-
*       OFF- LOAD FROM TAPE A1
*       ON- LOAD FROM CARD READER
* SWITCH 2-
*       OFF- PRE-CLEAR CORE WITH HALTS BEFORE LOAD
*       ON- DO NOT PRE-CLEAR CONTROL CORE
* SWITCH 3-
*       OFF- PERFORM CHECKSUM TESTS UNLESS COL.1 +
*       ON- OMIT CHECKSUM TESTS ON ALL CARDS
* SENSE LIGHT 1- TURNS ON WHEN LOADER STARTS
* HALT AND IOC LIGHT ON- CHECKSUM TEST FAILED
* HALT AND TCK LIGHT ON- PARITY ERROR IN READING
*
```

```
CLK      MISC    RTI,,EXIT     RESET TIMER REQ AND EXIT
TERMN8   TGR     CON,CLK       TEST + RESET CONSOLE REQ
         MISC    RTER          RESET TERMINATE REQUEST
         TGS     SL1,NOT1ST    TEST AND SET NEXT FLAG
         LID     20,00-02      LOAD SENCE SWITCH NUMBER
         TG      SW,READ       DONT CLEAR CORE IF SW ON
         LWAB    CLK           LOAD OFSET (END ADDRESS)
         MOPC    60,31         LOAD 1ST ADDRESS
* ONLY TEN BITS ARE USED AS A CONTROL MEM. ADDRESS
         MEM     B+C,ARD       LOAD A CLEARING PATTERN
         MEM     B+C,AWD       STORE A CLEARING PATTERN
         PC      C+1,24-35     INCREMENT THE ADDRESS
         TA      NZ,*-2        LOOP TILL CORE CLEARED
READ     MOPB    12,21         SET FOR TAPE A1
         LID     10,00-02      LOAD SENCE SWITCH NUMBER
         TGF     SW,*+2        SKIP IF TAPE LOAD
         MOPB    13,21         SET FOR CARDS
         LIB     02,09-11      LOAD READ CODE
         CHI     8,,EXIT       ISSUE SELECT AND EXIT
NOT1ST   LWAC    BUFFER+2      LOAD ADDRESS OF CHECKSUM
         MEM     C-1,ARB       READ CONTROL WORD TO B
         R4      C+1,S         SET R4 FOR FIRST INSTR.
         MEM     C,ARC         CHECKSUM TO C- 1'S COMP.
         PE      B+C           GENERATE END CARRY
         MQ      B+C+I,S       LOAD ADJ. CHECKSUM TO MQ
MOVE     MEM     R4,ARC        LOAD MINIPAIR TO C
         MEM     B,AWC         STORE MINIPAIR AWAY
         MQ      R+C           GENERATE END CARRY
         MQ      R+C+I,S       UPDATE CHECKSUM
         R4      R+1,S         UPDATE BUFFER ADDRESS
         PB      B+1,24-35     UPDATE PUTAWAY ADDRESS
         PB      B-1,12-17     DECREMENT MINIPAIR COUNT
         TA      NZ,MOVE       LOOP ON NON-ZERO COUNT
         TA      MSB,TAPERR    NO CKSUM TEST IF COL 1 +
         NOP     TERMN8        C+TV FOR TERMINATE ENTRY
         LID     30,00-02      LOAD SENCE SWITCH NUMBER
         TG      SW,TAPERR     NO CKSUM TEST IF SW 3 ON
         MQ      NR,LDD        COMPLEMENT CHECKSUM TO D
         PE      D             TEST CHECKSUM FOR ZERO
         TA      NZ,SETIOC     SET IOC ON INVALID CHECK
TAPERR   TCF     ER1,READ      SKIP ON NO PARITY ERROR
         TGS     TCK           TURN ON TCK LIGHT + HALT
         HALT    EXIT          HALT AND BUFF SERV C+TV
         TGR     TCK,READ      TURN OFF LIGHT AND READ
SETIOC   TGS     IOC           TURN ON IOC LIGHT + HALT
         HALT                  OPPERATOR DECISION STOP
         TGR     IOC           TURN OFF IOC LIGHT
         TRU     READ          READ NEXT CARD
```

```
EXIT    EXIT                     UNCONDITIONAL EXIT INSTR
        OCT       7700,17700 HALT AND NOP FILLER

        OCT       7700,17700 CORE CLEARING PATTERN

        OCT       7700,17700
```

APPENDIX A

NUMBER SYSTEMS AND CONVERSION

The common decimal notation of the commercial and scientific world is familiar to all of us. This notation is so familiar that you probably have never before questioned its use. Could it be possible that, for some purposes, another system is more convenient? The decision is entirely a matter of convenience. Decimal notation is used because it is most familiar and is understood by most people. However, had our primeval ancestors developed eight fingers instead of ten we would probably be more familiar with the octal system and would be questioning the decimal system.

The decimal system, with its ten digits, is learned by most people early in their training. This system serves very well for counting purposes. Why then, should computers which are designed to assist mathematicians, or engineers and businessmen, be designed to use the binary system of numbers?

Current digital computers use binary circuits and the mathematics of the computers is therefore binary in nature. The only convenient way to learn the operation of a computer is to learn the binary system. The octonary or octal system is a shorthand method of writing long binary numbers. Octal notation is used when discussing the computer but has no relation to the internal computer circuits.

Perhaps, as a first step, it would be well to see what is meant by the binary system of numbers. The binary, or base-two system, uses two symbols, 0 and 1, to represent all quantities. Counting is started in the binary system in the same manner as in the decimal system with 0 for zero and 1 for one. At two in the binary system it is found that there are no more symbols to be used. It is therefore necessary to take the same move at two in the binary system that is taken at ten in the decimal system. This move is to place a 1 in the next position to the left and start again with a 0 in the original position. A binary 10 is equivalent in this respect to a 2 in the decimal system. Counting is continued in an analogous manner with a carry to the next higher order every time a two is reached instead of every time a ten is reached. Counting in the binary system is as follows:

BINARY	DECIMAL	BINARY	DECIMAL
0	0	101	5
1	1	110	6
10	2	111	7
11	3	1000	8
100	4	1001	9

The binary system is used in computers because all present components are inherently binary. That is, a relay maintains its contacts either closed or open, magnetic materials are utilized by magnetizing them in one direction or the other, a vacuum tube is conveniently maintained either fully conducting or nonconducting, or the transmission of information along a wire may be accomplished by transmitting or not transmitting an electrical pulse at a certain time.

Although binary numbers in general have more terms than their decimal counterparts (about 3.3 times as many), computation in the binary system is quite simple.

For *addition*, it is only necessary to remember the following three rules:

1. Zero plus zero equals zero.
2. Zero plus one equals one.
3. One plus one equals zero with a carry of one to the next position on the left.

To see how the rules work, consider the addition of 15 plus 7 with these numbers expressed in binary notation:

	SIXTEENS	EIGHTS	FOURS	TWOS	ONES
(carries)	(1)	(1)	(1)	(1)	
	0	1	1	1	1 = 15
	+ 0	0	1	1	1 = 7
	1	0	1	1	0 = 22

In the ones column we have 1 plus 1 for a sum of 0 and a 1 carried to the two column. In the twos column we have 1 plus 1 for a sum of 0 but we must also add the carry from the ones column, making a final sum of 1 with a carry to the fours column. The same procedure occurs in the fours column. In the eights column we have a 1 plus a 0 giving a sum of 1, but adding in the carry from the fours column makes the final sum 0 with a carry to the sixteens column. In this column we have 0 plus 0 giving a sum of 0 and to this we add the carry from the eights column, making a final sum of 1.

The resultant sum of the addition contains 1's in the sixteens, fours, and twos columns, which is the binary representation of 22, the correct sum of 15 plus 7 (16 plus 4 plus 2 equals 22).

The rules for *subtraction* of binary digits are equally simple:

1. Zero minus zero equals zero.
2. One minus one equals zero.

3. One minus zero equals one.
4. Zero minus one equals one, with one borrowed from the left.

Using the same numbers as we did in the addition, the subtraction works as follows:

	SIXTEENS	EIGHTS	FOURS	TWOS	ONES	
(borrows)	0	0	0	0	0	
	0	1	1	1	1	= 15
	− 0	0	1	1	1	= 7
	0	1	0	0	0	= 8

In the ones column we have 1 minus 1 for a sum of 0 with no borrows. The same procedure occurs in the twos and fours columns. In the eights column we have 1 minus 0 for a sum of 1. In the sixteens column we have 0 minus 0 for a sum of 0. With the subtraction finished we have 1's in the eights column only, signifying the answer to be 8.

For *multiplication* only three rules need to be remembered:

1. Zero times zero equals zero.
2. Zero times one equals zero; no carries are considered.
3. One times one equals one.

The binary multiplication table is such that all that is necessary when multiplying one number (multiplicand) by another (multiplier) is to examine the multiplier digits one at a time and, each time a 1 is found, add the multiplicand into the result, and each time a 0 is found add nothing. Of course, the multiplicand must be shifted for each multiplier digit, but this is not different from the shifting that is done in the decimal system.

An example of binary multiplication is 26 multiplied by 19:

DECIMAL	BINARY
26 = 16 + 8 + 0 + 2 + 0 =	11010
× 19 = 16 + 0 + 0 + 2 + 1 =	10011
Using the above rules, the product	11010
will be arrived at by a series	11010
of adding the multiplicand	00000
and shifting whenever	00000
a 1 is found in the	11010
multiplier.	111101110

Interpreting the binary result of the multiplication by using the ones, twos, fours, . . . etc., system we find that we have,

256 + 128 + 64 + 32 + 0 + 8 + 4 + 2 + 0

which equals 494, thus proving the problem.

Binary division is accomplished by applying similar concepts. From the examples of addition, subtraction, and multiplication, it may be seen that whatever operation the computer is working on will be accomplished by repetitive addition.

The computer operates internally using the binary system. However, it is able to convert from one system to another by use of a stored program. Thus, input-output data may be expressed in decimal (or any other) form when the operator finds it more convenient to do so.

Octal Number System

It has already been pointed out that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. This is not much of a problem to the computer itself. However, in talking and writing, these binary numbers are bulky. A long string of ones and zeros cannot be effectively transmitted from one individual to another. Some shorthand method is necessary. The octal number system fills this need. Because of its simple relationship to binary, numbers can be converted from one system to another by inspection. The base or radix of the octal system is 8. This means there are eight symbols: 0, 1, 2, 3, 4, 5, 6, and 7. There are no 8's or 9's in this number system. The important relationship to remember is that three binary positions are equivalent to one octal position. The following table is used constantly when working on or about the computer.

BINARY	OCTAL
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

At this point a carry to the next higher position of the number is necessary, since all eight symbols have been used.

BINARY	OCTAL
001 000	10
001 001	11
001 010	12
001 011	13
001 100	14

and so on.

Remember that as far as the internal circuitry of the computer is concerned it only understands binary ones and zeros. The octal system is used to provide a shorthand method of reading and writing binary numbers.

Number Conversions

Before an attempt is made to convert numbers from one system to another, it is best to review what a number represents. In the demical system a number is represented or expressed by a sum of terms. Each individual term consists of a product of a power of ten and some integer from 0 to 9. For example, the number 123 means 100 plus 20 plus 3. This may also be expressed as:

$$(1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$$

Ten is said to be the base or radix of this system because of the role that the powers of 10 and the integers up to 10 play in the above expansion. If two is chosen as the base, numbers are said to be represented in the binary system. Consider the binary number 1 111 011. What do these zeros and ones represent? They represent the coefficients of the ascending powers of 2. Expressed in another way the number is:

$$(1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) +$$
$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

The various orders do not have the meaning of units, tens, hundreds, thousands, etc., as in the decimal system; instead they signify units, twos, fours, eights, sixteens, etc. In applying the above information it is found that the number 123 breaks down in both systems as follows:

```
BINARY                                DECIMAL
1 111 011                             1 2 3
| ||| ||└─1  units                    | | └─3   units
| ||| |└──2  twos                     | └───20  tens
| ||| └───0  fours                    └─────100 hundreds
| ||└──────8  eights                        ---
| |└──────16  sixteens                      123
| └───────32  thirty-twos
└─────────64  sixty-fours
          ---
          123
```

In the octal system, a number is represented in the same manner except that the base is 8. The digits ot the number represent the coefficients of the ascending powers of 8. Consider the octal number:

$$\begin{aligned} 173 &= (1 \times 8^2) + (7 \times 8^1) + (3 \times 8^0) \\ &= \quad 64 \quad + \quad 56 \quad + \quad 3 \\ &= 123 \text{ (decimal)} \end{aligned}$$

Similarly:

```
Octal 173
      ||└─3  units
      |└──56 eights
      └───64 sixty-fours
```

By remembering what a number represents in the binary or octal system, the number can be converted to its decimal equivalent by the method shown above. As the numbers get bigger, this method becomes quite impossible to use. The following section provides detailed methods for converting from one system to another.

Integers

DECIMAL TO OCTAL

Convert the decimal number 149 to its octal equivalent. RULE: Divide the decimal number by 8 and develop the octal number as per example.

```
8|_149 Remainder 5   ↑
8  |_18     "     2   |   = 225
8    |_2    "     2   |
       0             read
```

We first divided the original number to be converted by 8. The remainder of this first division becomes the low-order digit of the conversion (5). We then divide the quotient (received from the first division) by 8. Again the remainder becomes a part of the answer (next higher order, 2). This is continued until the quotient is smaller than the divisor. At this time the final quotient is considered the high order of the conversion (2).

OCTAL TO DECIMAL

Convert the octal number 225 to its decimal equivalent. RULE: Multiply by 8 and add, as per example.

```
  2 2 5
 × 8 | |
  16 | |
 + 2←┘ |
  18   |
 × 8   |
 144   |
 + 5←──┘
 149
```

The high-order digit is multiplied by 8 and the next lower-order digit is added to the result. The resultant answer is then multiplied by 8 and the next lower-order digit is added to the result. When the low-order digit has been added to the answer, the process ends. In the following examples, where multiplication or division is used, detailed explanations will not be used because the operations are similar.

Octal-Decimal Integer Conversion Table

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

APPENDIX B

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

Octal to Binary and Binary to Octal

Rule: Express the number in binary groups of three.

OCTAL TO BINARY

$$\overbrace{010}^{2}\ \overbrace{010}^{2}\ \overbrace{101}^{5} = 010\ \ 010\ \ 101$$

BINARY TO OCTAL

$$\underbrace{010}_{2}\ \underbrace{010}_{2}\ \underbrace{101}_{5} = 225$$

Decimal to Binary

Rule: Divide the decimal number by 2 and develop as per example; convert 149 to its binary equivalent.

$$\begin{array}{rlcc} 2 \lfloor 149 & \text{Remainder} & 1 & \uparrow \\ 2 \lfloor 74 & \text{"} & 0 & \\ 2 \lfloor 37 & \text{"} & 1 & \\ 2 \lfloor 18 & \text{"} & 0 & \\ 2 \lfloor 9 & \text{"} & 1 & = 010 \quad 010 \quad 101 \\ 2 \lfloor 4 & \text{"} & 0 & \\ 2 \lfloor 2 & \text{"} & 0 & \\ 2 \lfloor 1 & \text{"} & 1 & \\ 0 & \text{"} & \text{read} & \end{array}$$

Binary to Decimal

Rule: Multiply by 2 and add as per example; convert 010 010 101 to its decimal equivalent.

$$\begin{array}{r} 10 \quad 010 \quad 101 \\ \underline{\times\ 2} \\ 2 \\ \underline{+\ 0} \\ 2 \\ \underline{\times\ 2} \\ 4 \\ \underline{+\ 0} \\ 4 \\ \underline{\times\ 2} \\ 8 \\ \underline{+\ 1} \\ 9 \\ \underline{\times\ 2} \\ 18 \\ \underline{+\ 0} \\ 18 \\ \underline{\times\ 2} \\ 36 \\ \underline{+\ 1} \\ 37 \\ \underline{\times\ 2} \\ 74 \\ \underline{+\ 0} \\ 74 \\ \underline{\times\ 2} \\ 148 \\ \underline{+\ 1} \\ 149 \end{array}$$

OR 10 010 101

$$= 1\,(2^7) + 0\,(2^6) + 0\,(2^5) + 1\,(2^4) +$$

$$0\,(2^3) + 1\,(2^2) + 0\,(2^1) + 1\,(2^0)$$

$$= 128 + 16 + 4 + 1$$

$$= 149$$

Fractions

Decimal to Octal

Rule: Multiply by 8 and develop the octal number as per example:

$$\begin{array}{c|r} \text{Read} & .149 \\ \downarrow\ \bullet & \underline{\times\ 8} \\ 1 & .192 \\ & \underline{\times\ 8} \\ 1 & .536 \\ & \underline{\times\ 8} \\ 4 & .288 \\ & \underline{\times\ 8} \\ 2 & .304 \end{array}$$

$$= .1142 +$$

Octal to Decimal

Rule: Express as powers of 8, add and divide as per example:

$$\begin{aligned} .1142 &= 1\,(8^{-1}) + 1\,(8^{-2}) + 4\,(8^{-3}) + 2\,(8^{-4}) \\ &= 1/8 + 1/64 + 4/512 + 2/4096 \\ &= 610/4096 \\ &= .1489 \text{ plus} \\ &\text{or } .149 \end{aligned}$$

Octal to Binary and Binary to Octal

Rule: The same rule applies for fractions as for whole numbers.

Example:

$$\overbrace{.001}^{.1}\ \overbrace{001}^{1}\ \overbrace{100}^{4}\ \overbrace{010}^{2} \qquad \underbrace{.001}_{.1}\ \underbrace{001}_{1}\ \underbrace{100}_{4}\ \underbrace{010}_{2}$$

Binary to Decimal

The same rule applies as for whole numbers; for example:

.001 001 100 010

$$\begin{aligned} &= 1\,(2^{-3}) + 1\,(2^{-6}) + 1\,(2^{-7}) + 1\,(2^{-11}) \\ &= 1/8 + 1/64 + 1/128 + 1/2048 \\ &= 305/2048 \\ &= .1489 \text{ plus} \\ &\text{or } .149 \end{aligned}$$

Decimal to Binary

The same rule applies as for whole numbers. For example:

```
Read    .149
 |  •   × 2
 |  0  |.298
 |     |× 2
 |  0  |.596
 |     |× 2
 |  1  |.192
 |     |× 2
 |  0  |.384
 |     |× 2
 |  0  |.768
 |     |× 2
 |  1  |.536
 |     |× 2
 |  1  |.072
 |     |× 2
 |  0  |.144
 |     |× 2
 |  0  |.288
 |     |× 2
 |  0  |.576
 |     |× 2
 |  1  |.152
 |     |× 2
 |  0  |.304
 ↓  =   .001 001 100 010 +
```

Improper Fractions

Decimal to Binary

This requires conversion from decimal to octal and then to binary. For example, convert 149.149 to its binary equivalent.

```
8 |149.  remainder  5  ↑          |           .149
 8 |18.      "      2  |          |           × 8
   8 |2.     "      2  |          |      1  | .192
      0                read.      |         | × 8
                                  |      1  | .536
                                  |         | × 8
                                  |      4  | .288
                                  |         | × 8
                                  ↓ read. 2 | .304

  =  2     2     5     1     1     4     2
    010   010   101 • 001   001   100   010
```

$$149.149_{10} = 225.1142_8 = 010\ 010\ 101.001\ 001\ 100\ 010_2$$

Binary to Decimal

This requires conversion from binary to octal and then to decimal.

Convert to decimal:

```
    010  010  101 • 001  001  100  010
  =  2    2    5  •  1    1    4    2
   × 8    |    |     |    |     \    \
    16    |    |     ↓    ↓      ↓    ↓
   + 2 ←--┘    |    1/8 + 1/64 + 4/512 + 2/4096 =
    18         |
   × 8         |    610/4096 =
   144         |
   + 5 ←-------┘
   149.              .149
```

As with decimal-to-binary, conversion of the integer and fraction parts is performed independently.

Floating-Point Word

Decimal to Floating Poini

Convert decimal 149.149 to normal floating-point word.

Decimal to octal:

$149.149_{10} = 225.1142_8$

Octal to binary:

$225.1142_8 = 010\ 010\ \ 101.001\ 001\ 100\ 010_2$

Binary to floating point word:

$10\ 010\ 101.001\ 001\ 100\ 010 \times 2^0 =$

$.10\ 010\ 101\ 001\ 001\ 100\ 010 \times 2^8$

$8 + 128 = 136$ (Characteristic)

```
10 001 000.100 101 010 010 011 000 1 FP
└────┬───┘└───────────┬───────────┘
Characteristic     Fraction
 2  1  0 . 4  5  2  2  3  0  4₈
```

NOTE: Word is normal if the fraction is less than 1, but greater than or equal to one-half.

APPENDIX C

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

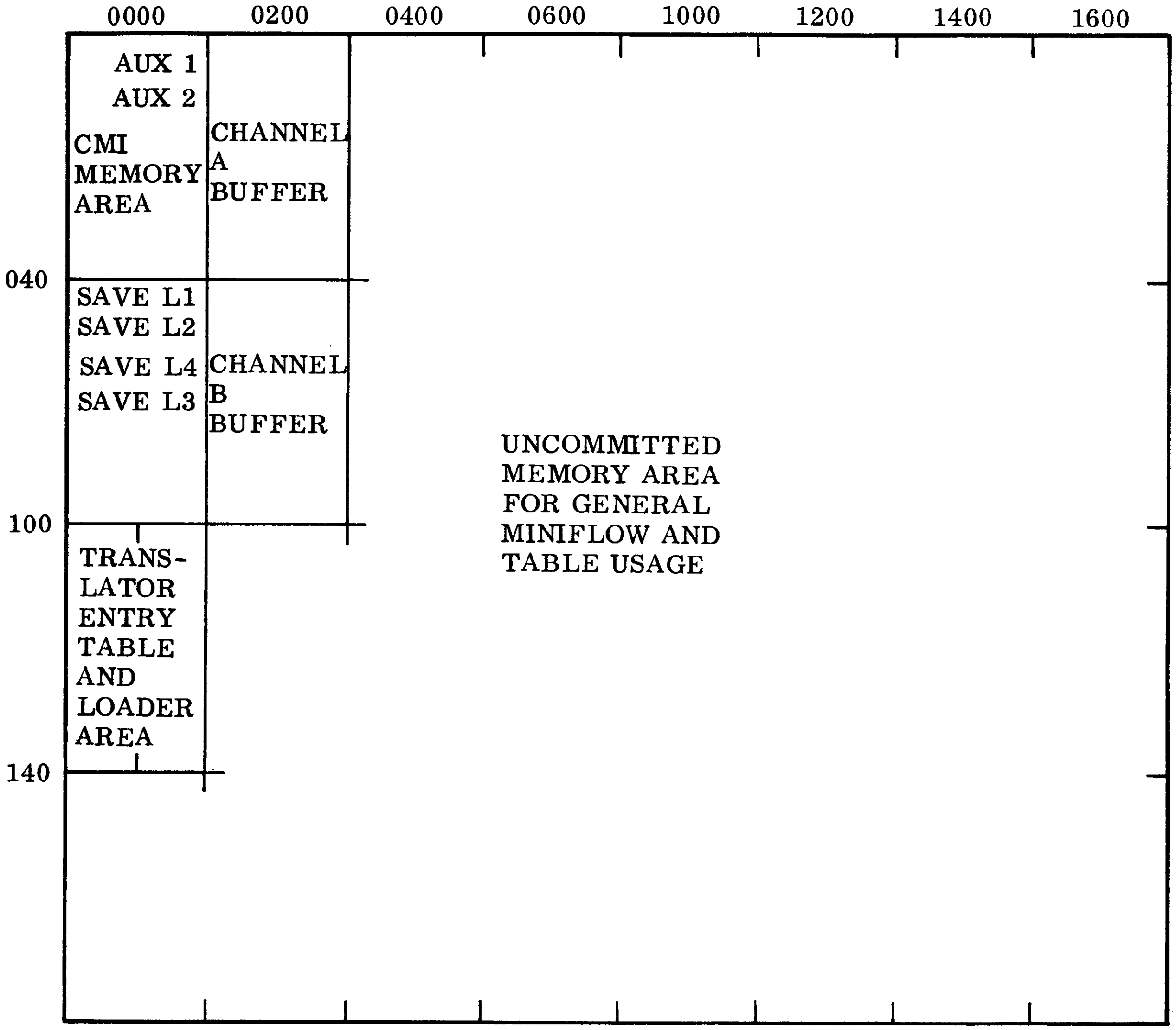
APPENDIX D

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

APPENDIX E

HARDWARE ORIENTED CONTROL MEMORY



CONTROL CORE MAP

FULL WORD ADDRESS	HARDWARE SPECIFIED AREAS
4	AUX 1 REGISTER
10	AUX 2 REGISTER
00-37	CMI MEMORY AREA
40-42	LEVEL 1 SAVE AREA
44-46	LEVEL 2 SAVE AREA
50-52	LEVEL 3 SAVE AREA
54-56	LEVEL 4 SAVE AREA
100-137*	TRANSLATOR ENTRY TABLE (control and transfer vectors)
200-237	CHANNEL A BUFFER AREA
240-277	CHANNEL B BUFFER AREA

SAVED AREA FORMAT

WORD	BITS	SAVED DATA
1	23-35	RB REGISTER
2	22	SMCT F/F
2	23-35	RD REGISTER
3	00-35	D REGISTER

*Correspond to MINIFLOW address 200-277.

APPENDIX F

WIRED-IN-SEQUENCE EXECUTION TIME

All times are in basic cycles and each cycle equals 175 nanoseconds. Memory interference time is considered and added to the number of basic cycles. No additional time is needed to index. All total times exclude the time in MINIFLOW (YA6), so execution times must be added to the times shown below.

No. of	W-I-S	
Basic Cycles	Steps Taken	Type of W-I-S
11	2,5,6	MINIFLOW entry from L1, L2, L3, and L4 trap.
18	1,2,5	Fast transfer (direct) - Total emulation time.
27	1-3,5	Fast transfer (indirect) - Total emulation time.
20	1,2,5,6	MINIFLOW program (direct).
28	1-3,5,6	MINIFLOW program (indirect).
28	1,2,4,5,6	MINIFLOW program (direct with op fetch).
42	1-6	MINIFLOW program (indirect with op fetch).
23	0-2,5	Execute fast transfer (direct) - Total emulation time.
32	0-3,5	Execute fast transfer (indirect) - Total emulation time.
25	0-2,5,6	Execute MINIFLOW program (direct).
33	0-3,5,6	Execute MINIFLOW program (indirect).
33	0-2,4-6	Execute MINIFLOW program (direct with op fetch).
47	0-6	Execute MINIFLOW program (indirect with op fetch).

Conversion Table

Cycles to Microseconds

1	.175	11	1.925	21	3.675	31	5.425
2	.350	12	2.100	22	3.850	32	5.600
3	.525	13	2.275	23	4.025	33	5.775
4	.700	14	2.450	24	4.200	34	5.950
5	.875	15	2.625	25	4.375	35	6.125
6	1.050	16	2.800	26	4.550	36	6.300
7	1.225	17	2.975	27	4.725	37	6.475
8	1.400	18	3.150	28	4.900	38	6.650
9	1.575	19	3.325	29	5.075	39	6.825
10	1.750	20	3.500	30	5.250	40	7.000

APPENDIX G

MINIFLOW INSTRUCTION PAIR EXECUTION TIME

All figures refer to times in YA6 (MINIFLOW) only and do not include W-I-S steps YA0-YA5. Each cycle equals 175 nanoseconds. Memory interference time is added and includes W-I-S interference in those cases where an exit is taken to the scheduler.

SECOND OR ONLY INSTRUCTION EXECUTED BEFORE AN EXIT OR A NEW PAIR IS FETCHED	ONLY ONE INSTRUCTION EXECUTED BEFORE AN EXIT OR A FETCH	FIRST INSTRUCTION EXECUTED OF A PAIR				
		2 Cycle Inst.	4 Cycle Inst.	Shift Inst. *	Control Memory Access	Main Memory Access
NEW PAIR FETCH						
With Overlap						
2 cycle inst.	6	7	9	9+2n	12	14
4 cycle inst.	7	9	11	11+2n	13	16
Without Overlap						
2 cycle inst.	7	9	11	11+2n	13	16
4 cycle inst.	9	11	13	13+2n	15	18
Shift*	9+2n	11+2n	13+2n	13+ 2 (n1+n2)	15+2n	18+2n
Control mem. access	12	13	15	15+2n	18	20
Main mem. access	14	16	18	18+2n	20	28
EXIT WITHOUT FETCH						
Exit to YA0						
2 cycle inst.	2	4	6	6+2n	8	11
4 cycle inst.	4	6	8	8+2n	10	13
Control mem. access	7	8	10	10+2n	13	15
Main mem. access	9	11	13	13+2n	15	23
Exit to YA1						
2 cycle inst.	2	4	6	6+2n	8	14
4 cycle inst.	4	6	8	8+2n	10	14
Control mem. access	6	7	9	9+2n	12	14
Main mem. access	14	16	18	18+2n	20	28

*n is determined as follows:

n_c is the shift count or,

n_c is the contents of RC if the TOP is 77 or,

n_c is the distance to the first bit on normalize or,

n_c is the contents of C_{27-35} on a DOS

then

For a DIV SOP $n = n_c$

For a shift other than a DIV or MULT SOP, where

n_c has been rounded up to a modulo 4 number, $n = n_{c/4}$, otherwise $n = n_c$.

For a MULT SOP, the bit content of the multiplier determines the value of n as follows:

Examining the multiplier from right to left (LSB first) each 1 bit requires 1 shift count; consecutive 0 bits of 3 or less, require no shift count; groups of four 0 bits require 1 shift count, except the MSB 4 bits which each require a shift count whether 1 or 0.

Overlap occurs when an eligible instruction (defined later) is executed in one of the three situations below:

1. At an odd location which does not exit back to the scheduler.
2. When an exit is taken from the subroutine mode.
3. A test instruction (not TAW) skips.

The eligible instructions for address cycle overlap during a mini fetch are listed below. Also shown are the basic cycle types.

Instruction	Cycles	Overlap
AC	2	yes
AKEYS	2	
ALG	4	
CH1	2	yes
CH2	2	yes
CMI	CA	
DELAY	2	
EXIT	2	
HALT	2	yes
IC	2	yes
INT	2	yes
KEYS	2	
LAB	2	yes
LAC	2	yes
LIB	2	yes
LIC	2	yes
LID	2	yes
MEM	MA	
MINI	2	
MISC	2	
MKEY	MA	
MQ	2	yes
NOP	2	yes
PB	2	yes
PBD	2	yes
PC	2	yes
PCD	2	yes
PD	2	yes
PE	2	yes
RC	2	
RD	2	
RESET	2	yes
R4	2	yes
SET	2	yes
SHIFT	4+2n	
SI	2	yes
SMCT	4	
TA	4	yes
TAE	4	yes
TAW	4	
TC	4	yes
TCE	4	yes
TCF	4	yes
TCR	4	yes
TCS	4	yes
TG	4	yes
TGF	4	yes
TGF	4	yes
TGR	4	yes
TGS	4	yes
TRU	2	
XR	2	yes

CA - Control memory access

MA - Main memory access

APPENDIX H

LISTING OF INSTRUCTIONS

Operation Alpha	Code Octal	Instruction	Page
AC	33	AC REGISTER OPERATION	104
AKEYS	54	ADDRESS KEYS OPERATION	120
ALG	62	ZONES ALGEBRAIC MAIN ENG. OPERATION	102
CH1	47	I/O CHANNEL OPERATION CONTROL	116
CH2	45	I/O TERMINATION CONTROL	117
CMI	77	DIRECTLY ADDRESSED CONTROL MEM. OPERATION	109
DELAY	54	DELAY PRESENᵢ LEVEL	140
HALT	00	STOP CLOCK	125
IC	22	IC REGISTER OPERATION	105
INT	57	CONSOLE INTERRUPT STATUS	118
KEYS	55	ENTRY KEYS OPERATION	120
LIB	14	LOAD B	122
LIC	15	LOAD C	122
LID	17	LOAD D	122
MEM	67	MEMORY OPERATION	108
MINI	02	MINI ENGINE COMBINATION OPERATION	111
MISC	06	SCHEDULER & AC SIGN & Q BIT CONTROL	138
MKEY	63	MEMORY/KEYS OPERATION	109
MOPB	10	MOP REGISTER TO B	122
MOPC	11	MOP REGISTER TO C	122
MQ	37	MQ REGISTER OPERATION	104
NOP	01	NO OPERATION	125
PB	50	PLACE IN B	98
PBD	40	PLACE D INTO B	100
PC	51	PLACE IN C	98
PCD	41	PLACE D INTO C	100
PD	53	PLACE IN D	98
PE	52	PERFORM OPER. SPECIFIED BY SOP	100
PRE	12	PRECONDITION CONTROLS	141
R4	23	HARD REGISTER NO. 4 OPERATION	105
RC	42	MINI ENGINE C OPERATION	114
RD	46	MINI ENGINE D OPERATION	114
SHIFT	66	MAIN ENGINE SHIFTS	92
SI	27	SI REGISTER OPERATION	104
SMCT	76	STORE MINI COUNT & TRANSFER	126
TA	61	TEST & SKIP	129
TAE	65	TEST & SKIP OR EXIT	130
TAW	75	TEST & TRANSFER TO W-I-S	130
TC	20	SKIP IF CI (SOP) IS ON	143
TCE	24	SKIP IF CI (SOP) IS ON; OTHERWISE, EXIT	144
TCF	31	SKIP IF CI (SOP) IS OFF	144
TCS	30	SKIP IF CI (SOP) IS ON THEN TURN CI (SOP) ON	144
TCR	34	SKIP IF CI (SOP) IS ON THEN TURN CI (SOP) OFF	144
TG	60	TEST & SKIP IF SATISFIED	135
TGE	65	SKIP IF SATISFIED; OTHERWISE, EXIT	130
TGF	71	SKIP IF TEST NOT SATISFIED	136

Operation Alpha	Code Octal	Instruction	Page
TGR	74	LIKE TG, BUT ALWAYS SETS I (SOP) OFF	136
TGS	70	LIKE TG, BUT ALWAYS SETS I (SOP) ON	136
TRU	16	TRANSFER	126
XR	26	INDEX REGISTER OPERATION	107

Operation Octal	Code Alpha	Instruction	Page
00	HALT	STOP CLOCK	125
01	NOP	NO OPERATION	125
02	MINI	MINI. ENGINE COMBINATION OPERATION	111
06	MISC	SCHEDULER & AC SIGN & Q BIT CONTROL	138
10	MOPB	MOP REGISTER TO B	122
11	MOPC	MOP REGISTER TO C	122
12	PRE	PRECONDITION CONTROLS	141
14	LIB	LOAD B	122
15	LIC	LOAD C	122
16	TRU	TRANSFER	126
17	LID	LOAD D	122
20	TC	SKIP IF CI (SOP) IS ON	143
22	IC	IC REGISTER OPERATION	105
23	R4	HARD REGISTER NO. 4 OPERATION	105
24	TCE	SKIP IF CI (SOP) IS ON; OTHERWISE, EXIT	144
26	XR	INDEX REGISTER OPERATION	107
27	SI	SI REGISTER OPERATION	104
30	TCS	SKIP IF CI (SOP) IS ON THEN TURN CI (SOP) ON	144
31	TCF	SKIP IF CI (SOP) IS OFF	144
33	AC	AC REGISTER OPERATION	104
34	TCR	SKIP IF CI (SOP) IS ON THEN TURN CI (SOP) OFF	144
37	MQ	MQ REGISTER OPERATION	104
40	PBD	PLACE D INTO B	100
41	PCD	PLACE D INTO C	100
42	RC	MINI ENGINE C REGISTER OPERATION	114
45	CH2	I/O TERMINATION CONTROL	117
46	RD	MINI ENGINE D REGISTER	114
47	CH1	I/O CHANNEL OPERATION CONTROL	109
50	PB	PLACE IN B	98
51	PC	PLACE IN C	98
52	PE	PERF. OPER. SPEC. BY SOP	100
53	PD	PLACE IN D	98
54	AKEYS	ADDRESS KEYS OPERATION	120
54	DELAY	DELAY PRESENT LEVEL	140
55	KEYS	ENTRY KEYS OPERATION	120
57	INT	CONSOLE INTERRUPT STATUS	118
60	TG	TEST & SKIP IF SATISFIED	135
61	TA	TEST & SKIP	129
62	ALG	ZONED ALGEBRAIC MAIN ENG. OPERATION	102
63	MKEY	MEMORY/KEYS OPERATION	109
64	TGE	SKIP IF TEST SATISFIED; OTHERWISE, EXIT	130
65	TAE	TEST & SKIP EXIT	130

Operation Code Octal	 Alpha	Instruction	Page
66	SHIFT	MAIN ENGINE SHIFTS	92
67	MEM	MEMORY OPERATION	108
70	TGS	LIKE TG, BUT ALWAYS SETS I (SOP) ON	136
71	TGF	SKIP IF TEST NOT SATISFIED	136
74	TGR	LIKE TG, BUT ALWAYS SETS I (SOP) OFF	136
75	TAW	TEST & TRANSFER TO W-I-S	130
76	SMCT	STORE MINI COUNT & TRANSFER	126
77	CMI	DIRECTLY ADDRESSED CONT. MEM. OPERATION	109

APPENDIX I

POP REFERENCE CHART

PAGE NO.	POP CODE	0 1 2 3 4 5		6	7 8 9 10 11	12	13	14	15	16	17
92	66	SHIFTS	SHIFTS, MULT & DIV.	GIN	SHIFT SOPS	SHIFT COUNT					
98	50	PB	MAIN ENGINE COMBINATIONS	GIN	MAIN ENGINE SOPS LDB AND LDC WITH HALF EXCHANGE	EXIT	MAIN ENGINE ZONES				
98	51	PC									
98	53	PD									
100	52	PE									
100	40	PBD	EXCHANGE COMBINATIONS	GIN	MAIN ENGINE SOPS	EXIT	MAIN ENGINE ZONES				
100	41	PCD									
102	62	ALG	ALGEBRAIC	GIN	(B-C) SOP, 16	EXIT	MAIN ENGINE ZONES				
104	33	AC	HARD REGISTER COMBINATIONS	GEX	MAIN ENGINE SOPS	EXIT	REPL B	REPL C	STORE IN REG	LOAD MAIN ENGINE REGISTER 00 NONE 01 PB 10 PC 11 PD	
104	37	MQ									
104	27	SI									
105	23	R4									
105	22	IC	INSTRUCTION COUNTER AND INDEXES	GIN & GEX							
107	26	XR									
108	67	MEM	MEMORY	GEX ON DATA	MAIN ENG SOP PROVIDE ADDR. USE ADDR. KEYS IF LOAD SOP	EXIT	0-CON 1-MN.	0-RD. 1-WR.	REPL. R4 FOR	00 ZEROS 01 PB 10 PC 11 PD	
109	63	MKEY									
109	77	CMI	IMMEDIATE MEMORY		MAIN ENG SOPS GIVE WR. DATA, LOAD SOPS GIVE READ DEST.	EXIT	CONTROL MEMORY ADDRESS				
111	02	MINI	ENGINE COMBINATIONS	▨	MINI ENGINE SOPS	EXIT	▨	CROSS	RB	RC	RD
114	42	RC	MAIN ENGINE TO MINI ENGINE	GEX	MAIN ENGINE AND LOAD SOPS	EXIT	MAIN ENGINE ZONES & FLOATING ZONE SPECIAL				
114	46	RD									
116	47	CH1	CHANNEL	GEX	MAIN ENGINE AND LOAD SOPS	EXIT	MAIN ENGINE ZONES EFFECTIVE ON LOADS ONLY				
117	45	CH2									
118	57	INT	INTERRUPT AND KEYS	GEX	LOAD SOPS ONLY	EXIT	MAIN ENGINE ZONES				
120	55	KEYS									
120	54	LAK									
122	14	LIB	LOAD IMMEDIATE DATA	6 BIT DATA PATTERN		CLEAR	MAIN ENGINE ZONES				
122	15	LIC									
122	17	LID									
122	10	MOPB	LOAD ADDRESS	18 BIT DATA PATTERN							
122	11	MOPC									
126	16	TRU	TRANSFER	▨	MINI FLOW TRANSFER ADDRESS						
126	76	SMCT									
129	61	TA	SPECIAL TEST AND SKIPS	TEST TRUE OR FALSE	TEST CONDITION	FWD OR REV	SKIP DISTANCE				
130	65	TAE									
130	75	TAW									
135	60	TG	GENERAL TEST AND SKIPS	GENERAL INDICATOR TEST FLIP FLOPS		FWD OR REV	SKIP DISTANCE				
135	64	TGE									
136	71	TGF									
136	70	TGS									
136	74	TGR									
143	20	TC	CHANNEL TEST AND SKIPS	▨	CHANNEL INDICATOR TEST FLIP FLOPS	FWD OR REV	SKIP DISTANCE				
144	24	TCE									
144	31	TCF									
144	30	TCS									
144	34	TCR									
138	06	MISC	SPECIAL CONTROL	MISCELLANEOUS ACTION		EXIT	▨				
140	540000	DELAY	TEMPORARY HANG	▨	▨	▨					
141	12	PRE	CONTROL PRESET	▨	0X NOP 14 ON 10 OFF	GP09	GP10	GP11	GEX	GIN	ARI
141											
138	064440	EXIT		▨							
125	01	NOP	DO NOTHING	▨							
125	00	HALT	STOP CLOCK	▨							

APPENDIX J

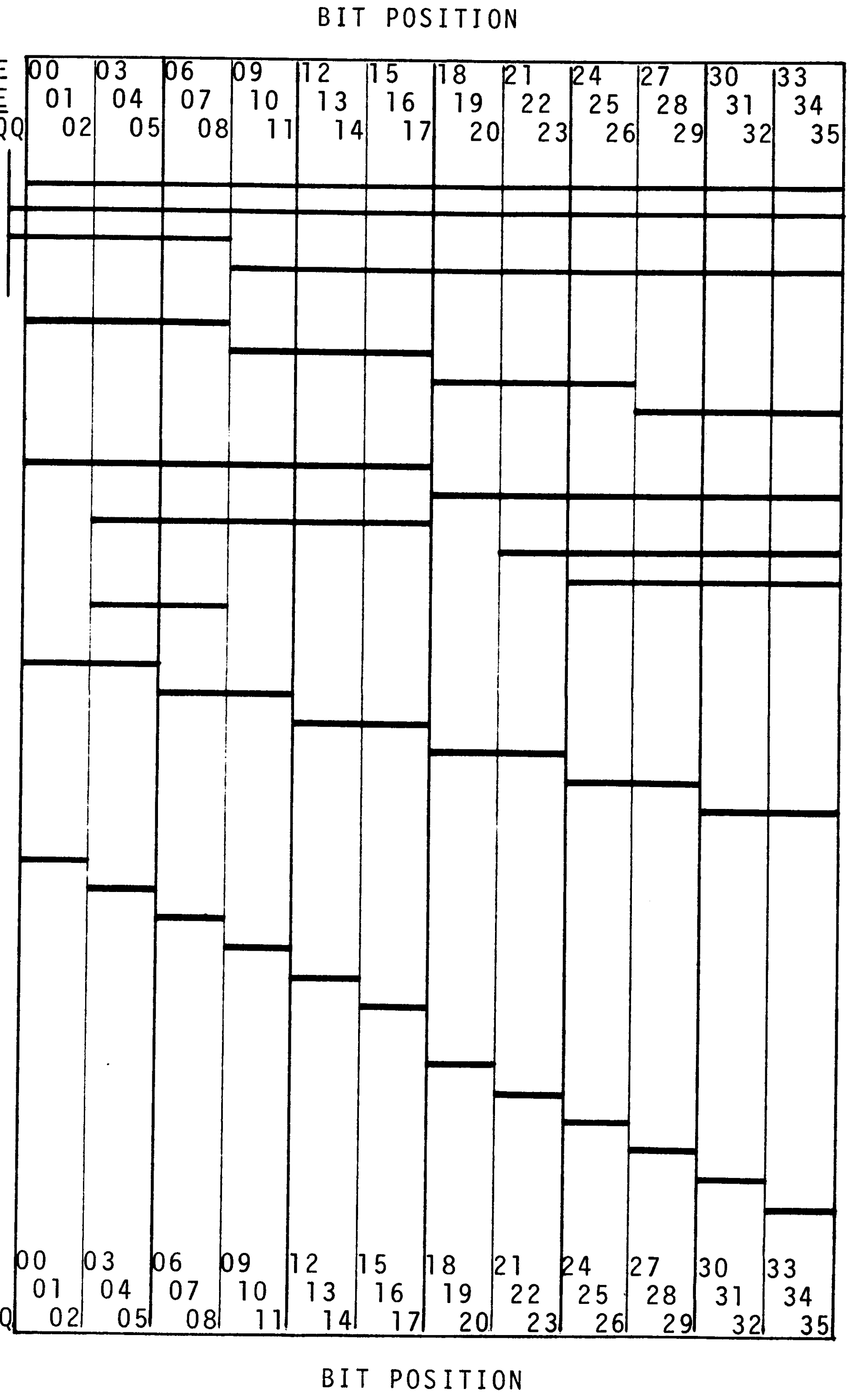
SOP REFERENCE SHEET

MAIN ENGINE SOPS AND LOADS	SHIFT SOPS	MISCEL-LANEOUS SOPS	DIRECT TEST SOPS	GENERAL/SPECIAL TEST SOPS		CHANNEL TEST SOPS	
				GEN-HDW	NOTE	CODE	NOTE
00 ZERO	02 B-L	54 HANG	40 SKIP	04 00 SSW	S	00 ER1	R
30 B *	03 B-R	62 RH1 *	41 CARRY	05 52 SL1	L	01 ER2	R
31 C *	22 C-L	66 RH3	42 CX11	07 54 SL2	L	02	
20 D *	23 C-R	70 RH4	43 G11	10 56 SL3	L	03	
34 B+1 *	04 D-L		44 ZERO	11 50 SL4	L	04	
35 C+1 *	05 D-R	60 HALT	45 9F.10	12 74 CON	N	05	
14 B-1	12 D-R	76 RINT	46 G9	13 42 DIS	N	06	
15 C-1 *		74 RESL4	47 G10	15 76 CIF	S	07	
16 B-C *	06 BD-L	42 SR1 *	50 RZERO	17 60 MTM	LC	10 BOT	LO
17 C-B	07 BD-R	46 SR3	51 FC	20 40 TRAP	LC	11 EOT	LO
36 B+C *	26 CD-L	50 SRT	57 GIN	21 66 FAC	NC	12	
06 B-C-I	27 CD-R	52 SR4	53 FOF	23 62 FMQ	NC	13	
07 C-B-I	10 DIV		54	25 64 FPO	NC	14	
26 B+C+I	11 MULT	56 POST	55 LS	26 72 PDCK	N	15	
02 NB+1			56 AQ	27 30 DCK	L	16	
03 NC+1		44 NOP	57 unused	31 16 DCTM	S	17	
		10 RXI					
22 NB	16 BD-LF	37 ACK *	60 ZX11	33 32 TCEA	LC	20 CEF	R
23 NC	17 BD-RF	16 RTER*	61 MSB	35 22 TCEB	LC	21 EOR	R
10 B.NC	14 BD-N	12 RTI	62 LSB	37 44 TCKA	LCF	22	
11 NB.C	32 BD-L9	72 RCN	63 EQU	41 46 TCKB	LCF	23	
24 B.C	33 BD-R9		64 LESS	43 54 CTEA	LC	24	
32 BUC	30 FDIV	23 FOFA	65 AOV	45 24 CTEB	LC	25	
12 BEC	31 FMULT	27 FOFQ	66 ACM	47 06 EOFA	LCF	26 DSP	L
01 DOL	20 DOS			51 14 EOFB	LCF	27 IOP	L
13		20 RAQ		53 70 TCN	L	30 TNR	R
25		21 SAQ		55 36 IOC	L	31 unused	
27		22 TAQ		57 01 unused		32	
33		24 RAS		61 02 unused		33	
37		25 SAS		63 03 unused		34	
04 LDB		26 TAS		65 67 unused		35	
05 LDC		33 RSQ		71 73 unused		36 CHB	RC*
21 LDD		30 SOV 31 ROV	77 SAT	75 77 unused		37 COP	A
*DENOTES MINI-ENGINE SOPS		*DENOTES CHB F/F STEERING IS USED		S-SWITCH CONTROLLED ONLY L-INDICATOR LAMP C-DIRECTLY CONTROLS LOGIC N-ALSO TURNED ON REMOTELY F-ALSO TURNED OFF REMOTELY R-ONLY CONTROLLED REMOTELY O-ONLY TURNED ON REMOTELY A-ONLY EXISTS ON A CHANNEL *ALSO SEE NOTE IN WRITEUP			

APPENDIX K

ZONE REFERENCE SHEET

NAME OF THE FIELD	ZONE CODE QQ
ENTIRE REGISTER	00
REGISTER & Q BIT	33
FLOATING EXPONENT	05
FLOATING FRACTION	20
1ST QUARTER WORD	35
2ND QUARTER WORD	23
3RD QUARTER WORD	15
4TH QUARTER WORD	03
LEFT HALF WORD	37
RIGHT HALF WORD	17
DECREMENT FIELD	27
ADDRESS FIELD	07
RIGHT THIRD WORD	13
TRANSLATOR DIGITS	25
CHARACTER 0	34
CHARACTER 1	36
CHARACTER 2	21
CHARACTER 3	14
CHARACTER 4	16
CHARACTER 5	01
OCTAL CHAR. 0	30
OCTAL CHAR. 1	24
OCTAL CHAR. 2	26
OCTAL CHAR. 3	32
OCTAL CHAR. 4	22
OCTAL CHAR. 5	31
OCTAL CHAR. 6	10
OCTAL CHAR. 7	04
OCTAL CHAR. 8	06
OCTAL CHAR. 9	12
OCTAL CHAR. 10	02
OCTAL CHAR. 11	11



READER'S COMMENT FORM

Standard Computer Corporation IC-M9

Principles of Operation

Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of Standard Computer Corp.

	Yes	No
Does this publication meet your needs?	□	□
Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

What is your occupation? ____________________

How do you use this publication?

As an introduction to the subject? □
For advanced knowledge of the subject? □
For information about operating procedures? □

As an instructor in a class? □
As a student in a class? □
As a reference manual? □

Other ____________________

Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 1762
SANTA ANA, CALIF.
92705

CUT ALONG THIS LINE

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

STANDARD COMPUTER CORPORATION
P.O. BOX 539
Santa Ana, Calif. 92705

ATTN: PUBLICATIONS UNIT

FOLD

STAPLE

STAPLE

Standard

Standard Computer Corporation
1411 W. Olympic Boulevard
Los Angeles, California 90015
(213) 387-5267

777 N. First Street, Suite 600
San Jose, California 95112
(408) 294-7150

55 S. 36th Street
Boulder, Colorado 80302
(303) 428-0529

7718 Tanglecrest Drive
Dallas, Texas 75240
(214) 239-9627

8 South Michigan Avenue
Suite 720
Chicago, Illinois 60605
(312) 726-0277

17500 Northland Park Court
Southfield, Michigan 48076
(313) 352-1710

Suite 1325 – Land Title Building
Broad & Chestnut Streets
Philadelphia, Pennsylvania 19102
(215) LO3-6350

7100 France Avenue South
Minneapolis, Minnesota 55435
(612) 926-0706

235 Bear Hill Road
Waltham, Massachusetts 02154
(617) 891-5083

2 West 45th Street
New York, New York 10036
(212) 661-1834