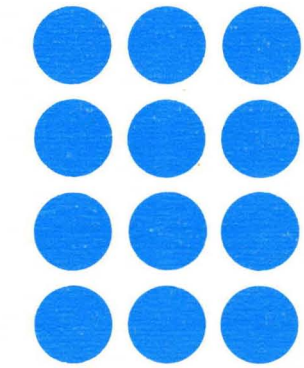
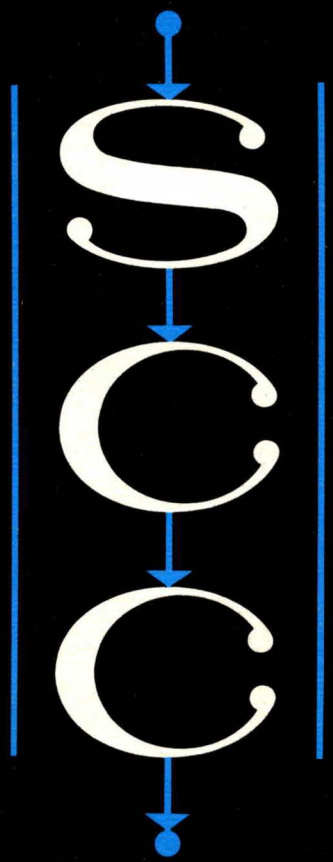
Scientific Control Corporation





SCC 6700 TIME SHARING COMPUTER

REFERENCE MANUAL

SCIENTIFIC CONTROL CORPORATION
14008 Distribution Way
Dallas, Texas 75234

Telephone: A/C 214 CH 1-2111

TABLE OF CONTENTS

FOREWORD

Computers brought a new era to every industry. Time sharing of single computing systems by many different users is a revolution in the art of data handling. The recent but now old yardstick of buying the minimum computer one can afford to suit a need has been outmoded by making the most powerful computers available to everyone. Scientific Control Corporation is a leader in this movement to provide any user the ease and power of the latest developments in software and hardware in the SCC 6700 Time Sharing Computer.

The computer products of Scientific Control Corporation are known for their fully parallel operation, modular design, large and flexible command repertoire, memory protect and direct access, and parallel processing. Data acquisition systems with a variety of input and output equipment have been designed around SCC computers. The SCC 6700 incorporates the past Scientific Control Corporation hardware and software techniques with the latest developments in equipment and programming.

Scientific Control Corporation is indebted to the University of California, Berkeley, for information on the Berkeley-Time Sharing System funded by Contract SD-185 of the Advanced Research Projects Agency, (ARPA), Office of the Secretary of Defense, Washington, D. C. The SCC 6700 employs improved versions of the Berkeley programs and advanced hardware structural characteristics suggested by experience with the Berkeley System and by advancements in equipment design by Scientific Control Corporation.

The SCC 6700 contains the hardware and software design techniques to provide file security and real-time access to a user's data through a powerful but economical system. The SCC 6700 is organized in a highly parallel fashion and may consist of single or multiple CPU's. The central processing unit has an operand fetch subsystem and an instruction fetch subsystem which are independent of each other and provide instruction overlap. Memory usage conflicts are held to a minimum through a unique system of communication buses, interleaving of memory modules and a technique of changing memory access priorities.

Variable field manipulating instructions, floating point instructions, and a paging structure which permit memory to be allocated by the

system in 2,048 word or 256 word blocks have been implemented to reduce system overhead. The 6700 also provides hardware detection of any access to or modification of a page of memory. Mutual protection of users from one another and protection of the system monitor is facilitated through complete and versatile hardware protection of memory and I-O access.

SYSTEMS DESCRIPTION

MACHINE ORGANIZATION

The logical organization of the SCC 6700 computer is shown in Figure 1.

The processor portion is made up of an index register (X) of 24 bits, an accumulator (A) of 24 bits, an auxiliary accumulator (B) of 24 bits, an instruction register (I) of 24 bits, and a location counter (LC) of 14 bits. Although the physical organization is actually different from this simplified picture, it is accurate as far as programming considerations are concerned.

In addition to these registers there are two individual flip-flops of importance to the programmer, the overflow (OV) and the carry (CARRY).

The location counter contains the address of the instruction being executed. During execution of each instruction, it is incremented by one (normal instruction progression), by two (certain skip instructions), or set to an altogether new value (branching instructions).

The index register is used for address modification, for loop control, or for a general auxiliary register.

The A register is the principal arithmetic register. It contains one of the operands in integer arithmetic operations, the most significant part of a floating point operand, and may be used as a general register which can be shifted and manipulated with versatility.

The B register is an auxiliary arithmetic register. It is used for the least significant part of double length fixed point operands or floating point operands. It can also be shifted and minipulated.

Effective Address Computation

In the SCC 6700, indexing and indirect addressing may be extended to any level, and is computed for all instructions in a uniform way as follows:

1. I is the instruction word (24 bits).

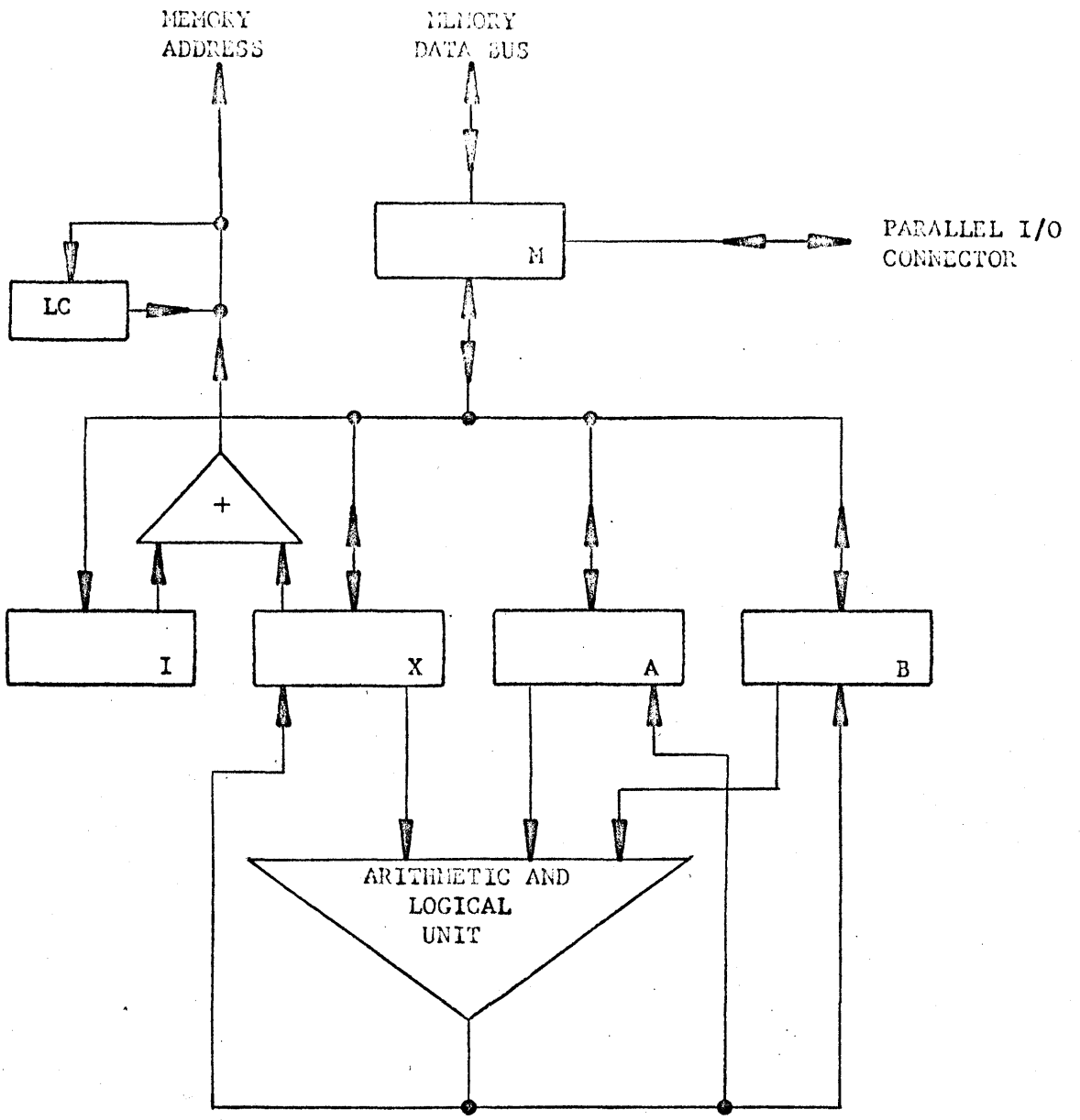


FIGURE 1

LOGICAL ORGANIZATION

2. $I_{10-23} \rightarrow Q$ — The address field of the instruction word is placed in the effective address register

3. If $I_1 = 1$, $Q + X \rightarrow Q$ — (if the index bit is set, add the index register to Q).

4. If $I_9 - 1$, $(Q)_{1,\ 9-23} \rightarrow I_{1,9-23}$ — Go to Step 2. (If the indirect bit is set, replace the index bit, indirect bit and address field of the instruction word with the corresponding bit of the specified memory word).

5. Q is now the effective address.

In instructions which interpret the address bits, the effective address is the actual source of the bits.

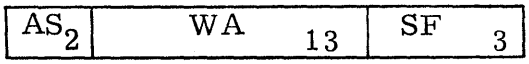
Memory Overlap

In order to provide the capability in the 6700 for minimum memory usage conflicts and thus, maximum operation speed, the system is designed around a foundation of high bandwidth communication buses and memory modules. Each memory module contains four address registers and four data registers and may, therefore, be processing requests from several different sources simultaneously. Each memory module consists of 8,192 twenty-five bit computer words. Each memory module is completely independent and may be operating simultaneously with any other memory module. The communication buses provide paths between the memory modules and the various devices served by the memories.

In a single CPU configuration, each memory module is connected to memory buses through a memory switching subsystem. Two of these buses communicate with the CPU. One communicates with the general I/O controller, and the fourth with the drum. In a multiple CPU system, two additional buses are added for use by the second CPU. If several buses simultaneously request memory service from the

same module, the requests will be honored in priority order. The drum controller has the provision for variable priority of the register in each module associated with the drum. If memory service is not required immediately, then the drum controller will request service with low priority. If the drum controller must have service, it switches the priority level to high and takes the next memory cycle. This technique of changing priority alleviates many memory usage conflicts by allowing the drum to steal cycles when they are not required.

Addressing of memory modules has been interleaved to place consecutive addresses in different modules. Memory modules are interleaved in groups of eight. Therefore, each of any eight consecutive addresses refers to a different memory module. This is accomplished by formatting the eighteen-bit actual address as follows:



18 bit actual address

WA - Word address within a memory module. The word address is a thirteen-bit field providing addresses in the range $0\text{-}17777_8$.

SF - Scan Field. The scan field causes the system to address eight different memory modules at the same word address before the word address is incremented. The scan field, then, selects a memory module within a memory array.

AS - Array, Select. This two-bit field selects one of four arrays, each array being eight memory modules. Thus, after scanning all words of memory from eight modules, the machine steps to the next array of eight and scans the new array.

By placing consecutive addresses in different modules, the possibility for conflict between processor, disc, and drum is reduced and the bandwidth of the memory is proportionately increased.

Processor Overlap

In order to achieve maximum computation speed, the central processing unit is separated, as shown in Figure 2, into three operating entities:

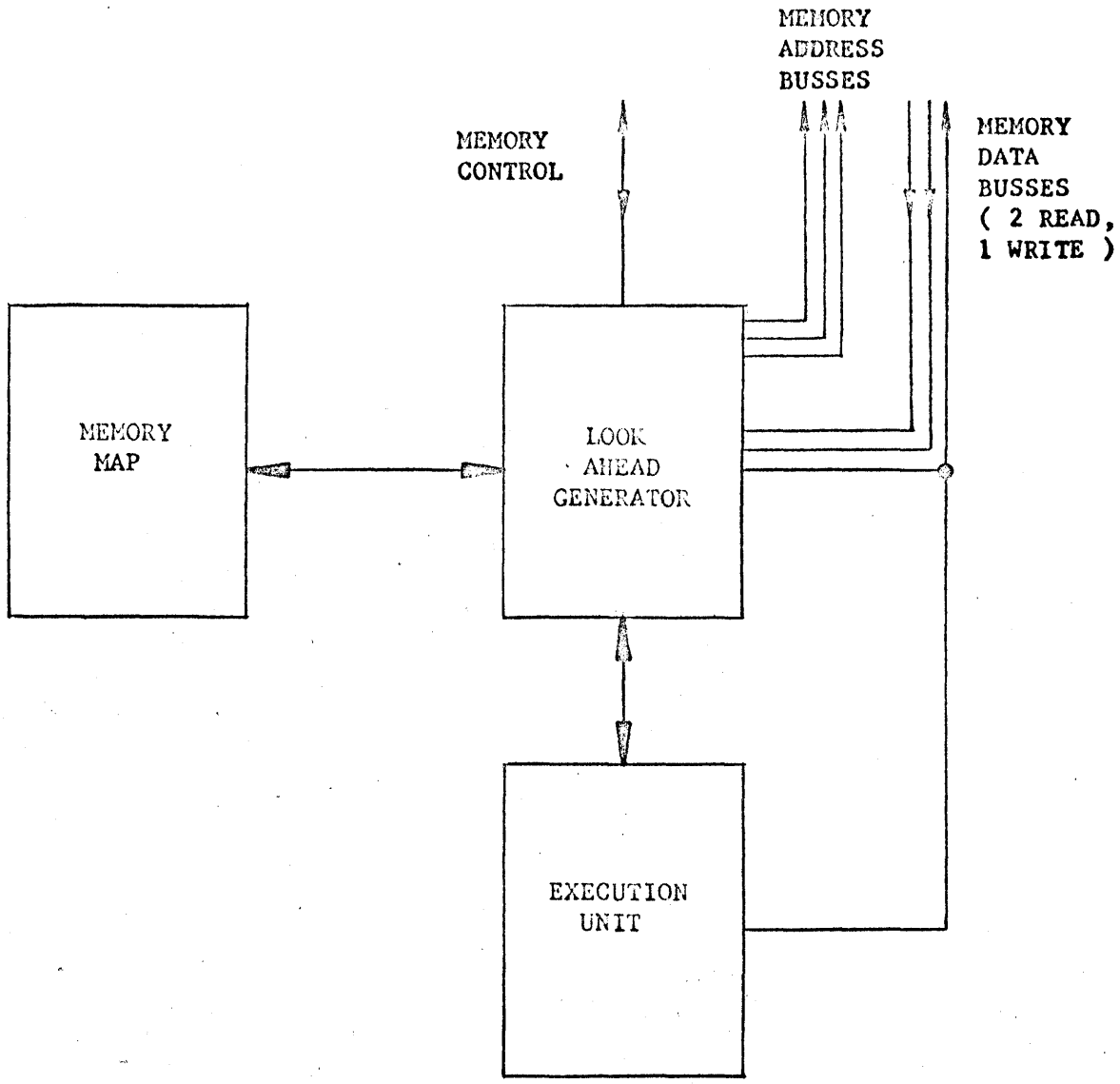


FIGURE 2

PRINCIPAL PHYSICAL COMPONENTS
OF CPU

Look-ahead Generator - consists of two subsystems:

Instruction Look-ahead - This subsystem communicates through a separate bus to all memory modules and will access instructions while previous instructions are being processed or while their indirect addresses or operand are being processed by the Operand Fetch Subsystem. The Instruction-Fetch Subsystem is inhibited when the current instruction may change the location counter or when the Operand Fetch Subsystem cannot receive the instruction due to other waiting instructions.

Operand Look-ahead - This subsystem provides for operand and indirect address fetches. It proceeds while previous instructions are being processed by the arithmetic processor. Instructions which require multi-level indirect addressing stay in the Operand Look-ahead subsystem until the final operand is obtained and placed in the holding register for entry into the execution unit. It then receives the next waiting instruction from the instruction look-ahead subsystem and proceeds to obtain the operand and/or effective address.

Since some instructions (such as the branch instructions) do not require services of the processor, they are carried to completion by the operand subsystem. It is possible, therefore, for complete instruction overlap to occur.

Execution Unit

This unit contains the registers visible to the programmer. It is a highly parallel arithmetic unit which operates completely autonomously with respect to the rest of the computer. It does not begin operation until the instruction has been completely prepared and the necessary operands are available.

The times quoted in the description of instructions are those required by the execution unit after all operands are available and are given in minor cycles (100 nanoseconds).

Memory Map

This unit contains the memory relabeling registers for both the

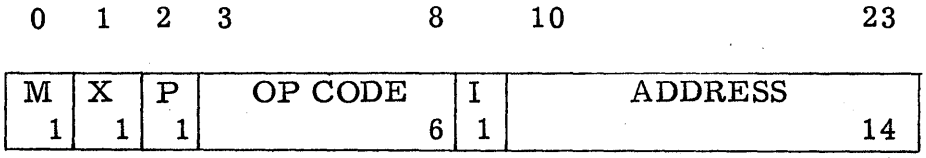
monitor and the user. It is used to transform apparent address into actual memory addresses. In the SCC 6700 it is not directly in the path to memory, but rather is used by the Look-ahead Generator to map address in advance of their actual use in memory accessing. This also improves memory performance by removing the mapping from the memory access paths.

WORD STRUCTURE

Each word in the computer is composed of 24 bits numbered from 0 to 23, from left to right. Information within the computer is stored in one of four formats: Instruction format, Fixed Point format, Floating Point format, or Field Descriptor format.

Instruction Format

Words which contain machine instructions are in the following format:



The interpretation of these bits is summarized below:

Bit	Meaning
0	SYSPOP BIT - This bit causes bits 2-8 to be interpreted as a system subroutine call.
1	INDEX BIT - If this bit is a one, the low order 14 bits of the index register are added to bits 10-23 of the instruction to form the effective address. If the bit is a zero, indexing is not used.
2	PROGRAMMED OPERATOR BIT - This bit causes the operation code to be interpreted as a subroutine call (if bit 0 is zero).
3-8	OPERATION CODE - These bits specify the machine operation to be performed.
9	INDIRECT ADDRESS BIT - This bit specifies indirection in preparing the effective address.

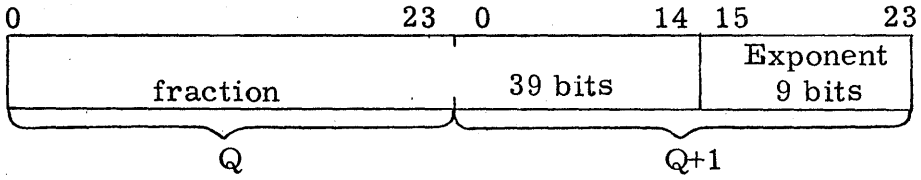
Bit	Meaning
10-23	ADDRESS - These bits specify the location of the operand as modified by indexing and indirect addressing considerations mentioned above.

Fixed Point Format

Single precision numeric data is represented internally in the computer as two's complement 24 bit numbers. Bit 0 of the word is the sign bit, and bits 1-23 are magnitude bits. If bit 0 is a one, the number is negative and is in two's complement form. For multiplication and division, the word is considered a fraction. That is, the assumed point is immediately to the right of the sign bit.

Floating Point Format

Floating point numbers in the SCC 6700 require two machine words with the following format:



The fraction part of a floating point number is a 39 bit two's complement fraction. Bit 0 of word 1 is the sign of the fraction. The exponent is a 9 bit two's complement integer. The binary point is assumed to be to the right of the sign bit, i.e., between bits 0 and 1. A floating point number is said to be normalized if the sign and most significant bit differ. The floating point arithmetic operations assume normalized operands, and with normalized operands will always produce normalized results except, of course, for unnormalized floating add and subtract. When the result fraction is zero, the exponent is set to zero so that zeros remain "clean".

Field Descriptor Format

The instructions which handle characters and variable length data refer to a word called a Field Descriptor. The format of

the Field Descriptor is discussed fully in the section on Field Handling Instructions.

MAP USAGE

In order to guarantee uninterrupted operation of the computer and to prevent users and the system from doing accidental damage to each other, a number of special features have been built into the computer. Foremost among these is full memory access control for both the system and the user. Access keys are separately specifiable for each page of memory in both the user and monitor maps to control whether words in the page may be

1. Read
2. Written
3. Executed as Instructions
4. Privileged Instructions

Any attempt to reference memory for a purpose not allowed will cause the monitor map to be invoked and a trap to occur to allow the monitor to regain control.

This approach has been adopted since there is no need for most of the system to have unconstrained access to the machine, especially the input-output instructions which are said to be "privileged":

Activate	ACT
Parallel Output	POT
Parallel Input	PIN

MAP TRANSITION

There are two types of conditions which can cause transitions between maps. The first, under the control of the program, is programmed transitions and can be performed as follows:

Monitor-to-user map transitions - The transfer from monitor to user map is made by executing any indirect jump instruction through a word in which bit zero of the indirect address contains a one in bit zero.

User-to-Monitor map transition - The user can cause an intentional transfer from user to monitor map by the execution of a SYSPOP. A detailed discussion on SYSPOPS is given in the paragraph labeled Systems Programmed Operators.

These are not the only ways in which a map transition can occur. There are two other causes for map transitions. First, the occurrence of an interrupt or trap when in the user map will cause the system to change to the monitor map. Second, following the execution of a single instruction interrupt routine, a transition to the user's map will occur if the machine was in the user's map at the time the interrupt occurred. In order that the system subroutines will be able to serve both the user and the system itself, indication of the mode before entry is preserved in the subroutine link. A one in bit zero implies a transfer from the user's mode and bit zero equals zero implies that the subroutine entry was from the system. Bit zero is used for this purpose in order to make data access independent of mode and to restore the proper mode upon return.

While in the monitor map, the user map can be invoked by indirect addressing through a word with bit zero set to one. Monitor programs can thus conveniently access information in the user's area. Specifically, if bit zero of the word fetched during an indirect address fetch is detected , all further references to memory made during this instruction will be relabeled using the user's memory map.

MEMORY RELABELING

The SCC 6700 provides a memory relabeling technique which permits dynamic hardware relocation of programs. Memory relabeling prevents a user from interfering with or being interfered with by other users. The 6700 memory system consists of up to 262, 144 twenty-four bit words partitioned into one hundred twenty-eight 2,048 word pages. One of the monitor pages is further subdivided into 256 word page segments. The address field of the instruction word consists of the rightmost 14 bits providing the capability of directly addressing 16,384 words. To the user, memory appears to be 16,384 words of contiguous storage. The monitor, however, may locate memory for the user in non-contiguous 2,048 word pages in the actual memory. This is accomplished through the use of the mapping registers. The monitor also has 2,048 word pages with one page further divided into 256 word page segments.

The user's memory map consists of an actual page register for each of the user pages. The three most significant bits of the address field of the instruction are used to address a relocation register. The contents of the actual page registers specify which page of actual memory each user page is to occupy. Since, to the user, memory appears as eight consecutive 2,048 word pages, eight relabeling registers are incorporated for user programs. Each relabeling register is twelve bits in length. The user's relabeling registers are laid out in four composite registers designated RLO-RL3 as follows:

24 Bits

RLO	UMO	UM1
RL1	UM2	UM3
RL2	UM4	UM5
RL3	UM6	UM7

User's Memory Map

The 14-bit address field of the instruction word is divided into two subfields. The three high-order bits designate a user page number. The 11 low-order address bits specify a word address within the page, hence the page size of 2,048 words. The relabeling hardware views the user's page number, i, as the address of a relabeling register, UM_i. The seven low-order bits of UM_i specify the actual page address in memory. That is, the seven low-order bits of UM_i are appended to the 11-bit word address to form an 18-bit actual memory address. Each relabeling register contains the following information:

12

AC	MD	AK	PN
1	1	3	7

scale

PN - Seven bits denoting the actual page in memory

AK - Three bits of access key as follows:

1	2	3	
0	0	0	No access
0	0	1	Read, Write
0	1	0	Execute
0	1	1	Execute privileged
1	1	0	Read, Execute
1	1	1	Read, Execute privileged
1	0	0	Read, Write, Execute
1	0	1	Read, Write, Execute privileged

MD - One bit, set automatically if any store occurs in this page

AC - One bit, set automatically when any word is accessed (including a store) in this page

Relabeling memory using these techniques allows for dynamic program relocation with complete memory protection and the ability to assign non-contiguous blocks of memory to a user. There is no degradation in performance as a consequence of memory mapping. The implementation of the relabeling registers is shown in Figure 3.

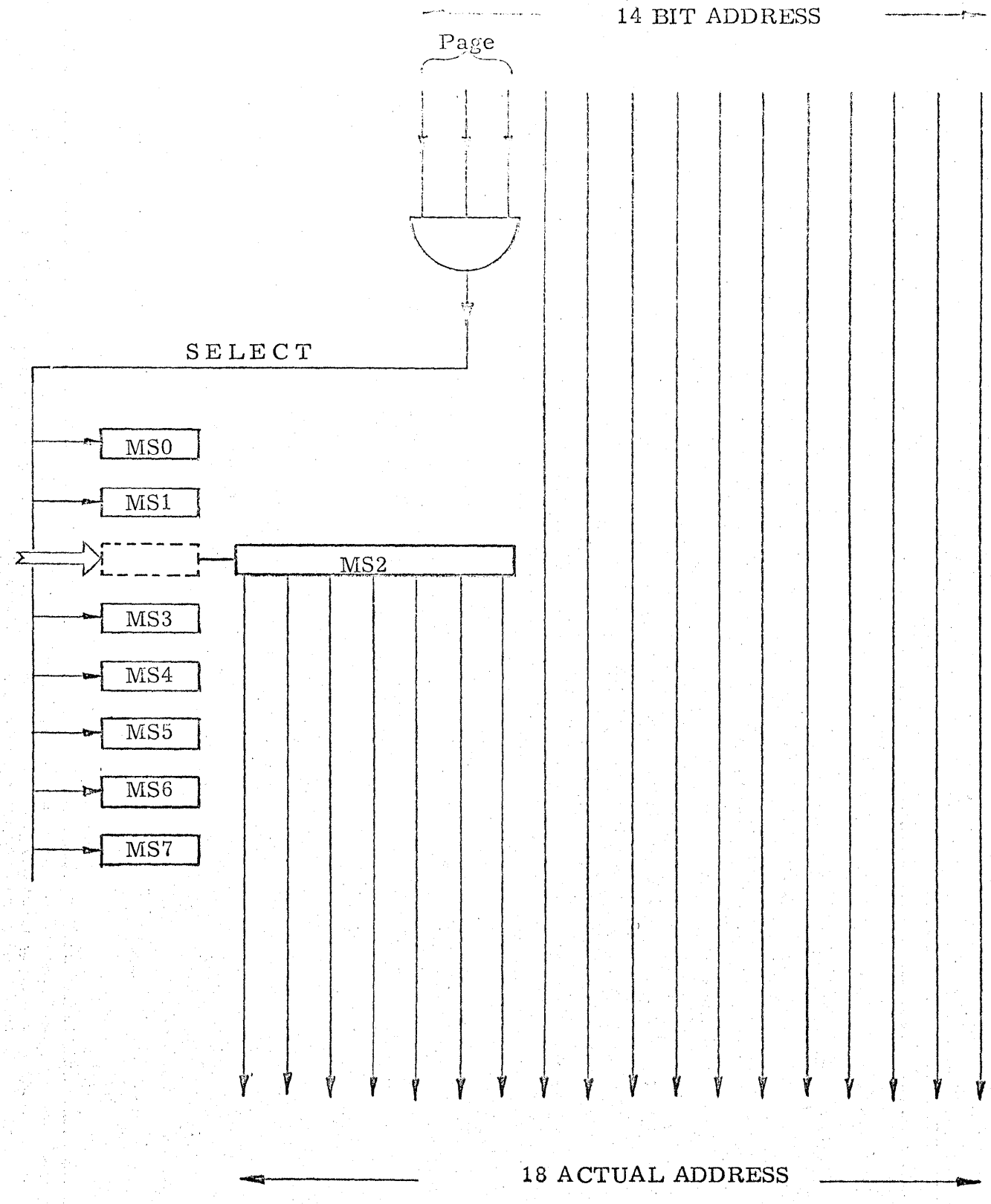
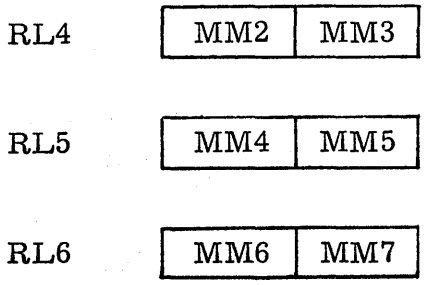

14 BIT ADDRESS
Page
SELECT
MS0
MS1
MS2
MS3
MS4
MS5
MS6
MS7
18 ACTUAL ADDRESS

FIGURE 3

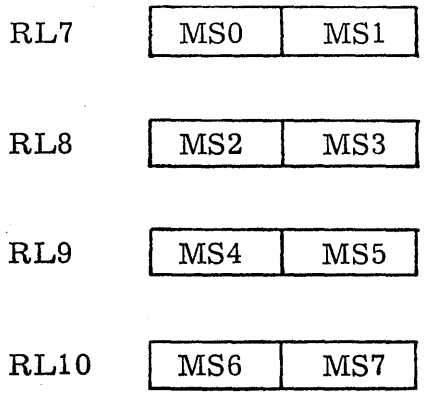
While in the user mode, relabeling is always performed on each address using the user's memory map. While in the monitor mode, it is possible to invoke relabeling using the user's map for individual instructions. In accessing memory to obtain the effective address of an instruction, any word encountered with bit zero set causes user relabeling to apply immediately and for the duration of that instruction. For example, this would occur if a one in bit zero is detected during a chain of indirect addressing. As soon as bit zero was detected as a one, relabeling using the user's memory map would apply immediately and continue to be applied on all further levels of indirect addressing. Thus, all subsequent memory references would come from user relabeled memory.

The monitor's memory map consists of six page relabeling registers and eight page segment relabeling registers. When the machine is operating in the monitor mode, the monitor's memory map is used in lieu of the user's memory map. Only the uppermost six page numbers are relabeled and the page one is constructed of page segments. Hence, addresses with page number 0 are taken as actual machine addresses while all other addresses are relabeled. Page 1 is further mapped using the eight page segment registers. It should be noted that the paging structure is invisible to the user. The fact that the monitor assigns non-contiguous blocks of memory to sequential page numbers is of no consequence to the user or the machine since memory appears as 16,384 words located at sequential addresses.

The monitor's relabeling registers are laid out as follows:



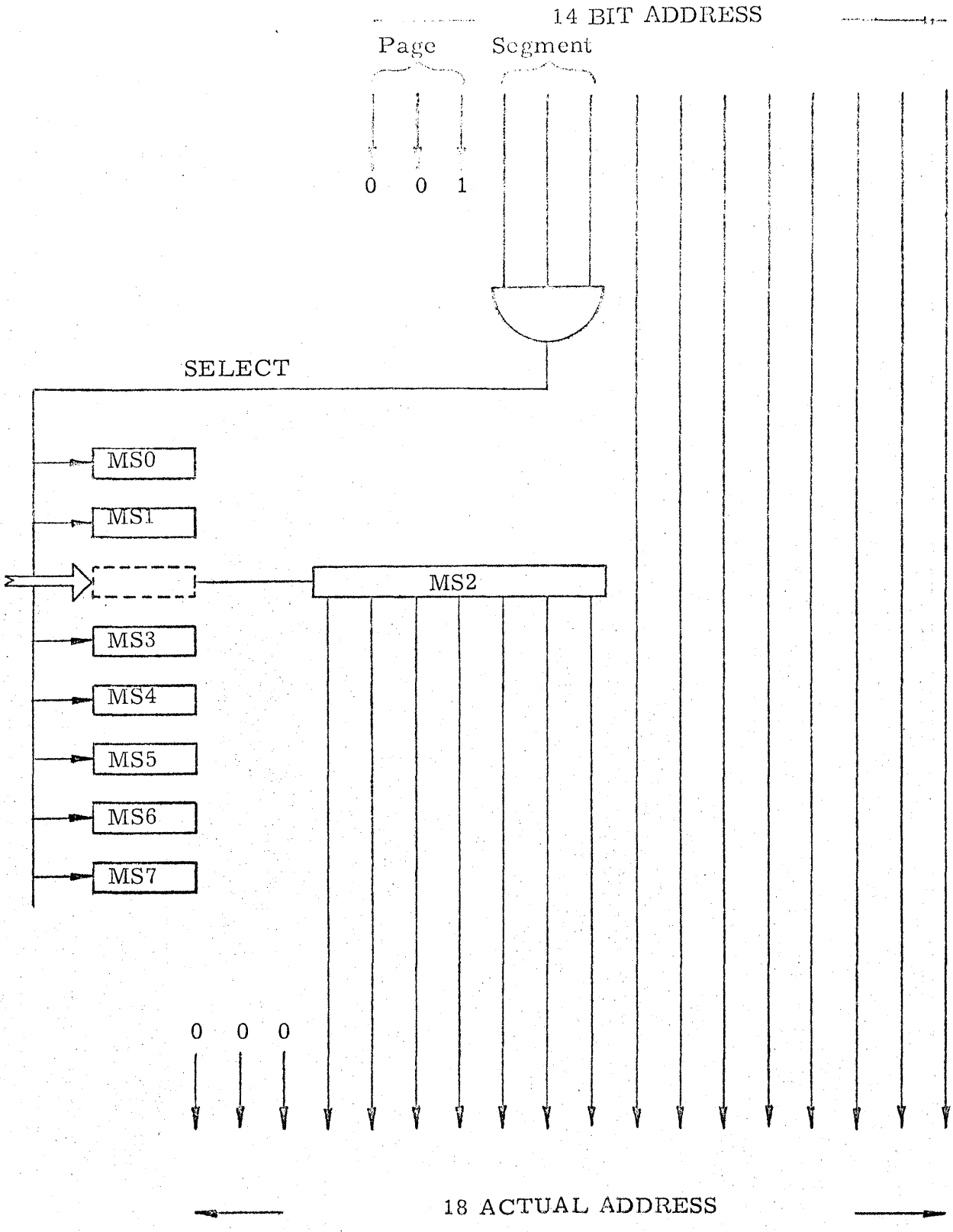
Page relabeling for Monitor Pages 2, 3, 4, 5, 6, and 7.



Page segment (256 words) relabeling for Monitor Page 1.

Monitors's Memory Map

Actual mapping of monitor Page 1 is shown in Figure 4. Note particularly that the three high order actual address bits are zero. Actual addresses corresponding to monitor address in Page 1 must, therefore, fall within the first 64K of memory.



SEGMENT MAPPING OF MONITOR PAGE 1

FIGURE 4

PROGRAMMED OPERATORS

One of the facilities provided in the 6700 for subroutine entry is the programmed operator. Through the use of this facility, a subroutine may be called by the execution of a single instruction of the same format as built-in machine hardware instructions. The user programmed operator instruction is specified by a zero in bit zero of the instruction word and a one bit in bit two of the instruction word. Upon detection of the user programmed operator tag, the operation code is not decoded in the normal manner. Instead it is used as a relative address of an execute instruction. User programmed operators cause the machine to execute locations $100\text{-}177_8$ in the user's memory. Thus an instruction which was tagged as a user programmed operator with an operation code of 164_8 would be interpreted as an execute of the instruction in core position 164_8. A subroutine linkage instruction is located in location 164_8. This provides entry into a subroutine with space in the address field of the instruction to pass a parameter. Return to the calling program is accomplished with an indirect branch instruction. It should be noted that all memory references throughout the execution of the programmed operator use the user's memory map. Hence, all subroutine linkages would be stored in the user's data block and the subroutines which are being called must be stored within the user's memory allocation.

SYSTEMS PROGRAMMED OPERATOR (SYSPOP)

A SYSPOP is similar to the user's programmed operator. It is distinguished by a 1 in the sign bit and either a 0 or a 1 in bit 2. There can be, therefore, up to 128 SYSPOP's. Before the instruction to be executed is accessed, the monitor map is invoked. Thus, instead of executing an instruction taken from the user's memory, the instruction comes from the monitor memory from absolute locations 1000_8-1177_8. All subsequent instructions are in the monitor mode. The storage of the return linkages is in the monitor memory and provision does not have to be made in the user's memory for their storage. The routine is also stored in the monitor memory, thus a user has the capability of calling a subroutine outside of his memory through the use of a SYSPOP. Thus, subroutines which are commonly used by many users can be called with a single instruction and storage does not have to be allocated for the linkage nor the subroutine. During the storage of the return linkage and the effective address, the zero bits are set to one if the subroutine was entered from the user mode. Therefore, any references into the calling program to acquire an argument will detect that the calling program was in the user mode and the user's memory map will apply during such argument fetches.

INSTRUCTIONS

This section describes the instruction repertoire of the SCC 6700 computer. Instruction description includes mnemonic, computer operation code, instruction name, number of machine cycles required to execute the instruction and machine function performed by the execution. Examples are given, when needed, to clarify the description. In discussing instruction functions, the following general conventions apply:

1. The letter "Q" refers to the effective address, i.e., Q refers to the actual address used in the execution of the instruction after all indexing and indirect addressing has been accomplished. In some instructions, Q is not the address of an operand but is itself the operand. When used in this manner, Q is said to be an immediate operand.

2. All numbers, locations, etc., are in octal unless otherwise noted.

3. Subscription is used to denote bit positions within a register. For example, A_{9-23} refers to bits 9 thru 23 of the A register.

4. A register name enclosed in parentheses denotes use of that register to address a memory location. For example, (Q) refers to the contents of memory at the effective address.

5. Timing given is in minor cycles (100 ns) for execution after operands are obtained from memory (or effective address in case of stores).

6. The carry and overflow flip-flops are affected only by instructions which put a result in A, or which affect these indicators explicitly.

7. All instructions are indexable and indirectly addressable.

The symbols and abbreviations used in instruction definitions are as follows:

Symbol	Definitions
A	Main arithmetic register or accumulator
B	Auxiliary accumulator
X	Index Register
Q	Effective operand address of instruction
LC	The location counter. Contains the address of the next instruction to be executed.
MAP	Current Map Bit (User or Monitor)
OV	Overflow indicator
CARRY	Carry flip-flop
$\longrightarrow$	Replacement designator. The value on the left is placed into the value on the right.
+	Add
-	Subtract (or negate)
/	Divide
*	Multiply
$\cap$	Logical AND
$\cup$	Logical OR
—	Complement
$<$	Less than
$>$	Greater than
=	Equal
$\neq$	Not equal

LOADS AND STORES

LDA LOAD A 2

(Q) → A

The contents of memory at the effective address replaces the contents of the A register.

STA STORE A 2

A → (Q)

The contents of the A register replaces the contents of memory at the effective address. The contents of the A register remains unchanged.

XMA EXCHANGE MEMORY AND A 3

A ↔ (Q)

The contents of the A register are exchanged with the contents of memory at the effective address.

LDB LOAD B 2

(Q) → B

The contents of memory at the effective address replaces the contents of the B register.

STB STORE B 2

B → (Q)

The contents of the B register replaces the contents of the effective address. The contents of the B register remains unchanged.

LDX LOAD INDEX 2

(Q) → X

The contents of memory at the effective address replaces the contents of the index register.

STX STORE INDEX 2

$X \longrightarrow (Q)$

The contents of the index register replaces the contents of memory at the effective address. The contents of the index register remains unchanged.

XMX EXCHANGE X AND MEMORY 3

$X \longleftrightarrow (Q)$

The contents of the X register are exchanged with the contents of memory at the effective address.

STM STORE MASKED 3

$$\{\bar{B} \cap (Q)\} \quad \cup \quad \{B \cap A\} \longrightarrow (Q)$$

A one bit in any position of the B register causes the corresponding bit in the A register to be stored in the corresponding bit of memory at the effective address. Bit positions in memory corresponding to zero bits in the B register remain unchanged. The contents of both the A and B registers remain unchanged by this instruction.

LDD LOAD DOUBLE 5

$(Q) \longrightarrow A; \quad (Q + 1) \longrightarrow B$

The contents of memory at the effective address are placed into the A register. The contents of the memory at the effective address plus one are placed into the B register.

STD STORE DOUBLE 5

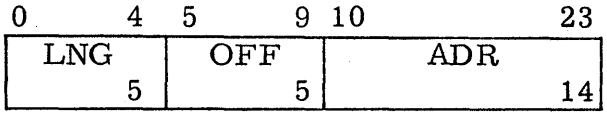
$A \longrightarrow (Q); \quad B \longrightarrow (Q + 1)$

The contents of the A register are stored into memory at the effective address. The contents of the B register are stored into memory at the effective address plus one.

FIELD LOADS AND STORES

The following six instructions are used to load and store parts of words in memory. They allow convenient handling of fields from zero to twenty-four bits in length arbitrarily positioned in either a single word in memory or two adjacent words. These instructions all make use of a common "field descriptor" or pointer word to control the field to be loaded or stored.

The field instructions expect the word at the effective address to be a word of the following format:



A word in this format is called a "field descriptor" or "FD" and defines a contiguous field in memory from zero to twenty-four bits in length. The meaning of the parts of an FD are as follows:

LNG - A five bit integer which defines the length of the field. LNG must be equal to or less than 24 in the standard case. Specification of a length greater than this will cause an interrupt whenever the FD is referenced. An LNG value of 31 (37 octal) is used as a special case to be described later.

OFF - A five bit integer which defines the offset of the field from the left side of the word addressed. The value of OFF must lie between 0 and 23 inclusive or an interrupt will occur. Bit 0 is the left (high order) bit of the word and bit 23 is the right (low order) bit of the word. An OFF value of 31 (37 octal) is used as a special case to be described later.

ADR - A fourteen bit integer which is the memory address of the word containing the left most bit of the defined field.

Although these instructions will most frequently be used to handle eight bit (or six bit) characters packed three (or four) to a word,

these instructions are explicitly intended to handle arbitrary fields which may overlap word boundaries in any way.

In order to avoid repetition in the following instruction descriptions, the acquisition and set-up of the field description will be given here and is common to all six field handling instructions.

1. $(Q)_{10-23} \rightarrow$ ADR, $(Q)_{5-9} \rightarrow$ OFF, $(Q)_{0-4} \rightarrow$ LNG
2. If OFF = 37_8, $(X)_{5-9} \rightarrow$ OFF
3. If LNG = 37_8, $(X)_{0-4} \rightarrow$ LNG
4. If OFF $>$ 23, Interrupt
5. If LNG $>$ 24, Interrupt
6. OFF $\rightarrow$ I, OFF + LNG -1 $\rightarrow$ J, 23 - LNG + 1 $\rightarrow$ K

That is, the contents of the effective address are separated into their component pieces. If either the offset or length is 31 (37_8), the value is taken from the corresponding part of the index register. The offset and length are then checked to be sure they are within limits.

Finally, the left and right bit numbers of the field in memory and the left bit number of the field in the A register are computed.

LDF LOAD FIELD 10

$0 \rightarrow A$, $(ADR)_{I-J} \longrightarrow A_{K-23}$

The field described is right aligned in the A register. The remainder of the A register is cleared.

STF STORE FIELD 10

$A_{K-23} \longrightarrow (ADR)_{I-J}$

The field described by the FD at the effective address is replaced by the right most LNG bits in the A register. A is not affected by this instruction.

LDFX LOAD FIELD INDEXED 12

$ADR + X_{10-23} \longrightarrow ADR$

$0 \longrightarrow A\ ,\ (ADR)_{I-J} \longrightarrow A_{K-23}$

This instruction is similar to LDF except that the contents of the address field of the index register are added to the FD address before the field is loaded.

STFX STORE FIELD INDEXED 12

$ADR + X_{10-23} \longrightarrow ADR$

$A_{K-23} \longrightarrow (ADR)_{I-J}$

This instruction is similar to STF except that the contents of the address field of the index register are added to the FD address before the field is stored.

LDFI LOAD FIELD AND INCREMENT 14

If $OFF = (Q+1)_{5-9}$ and $ADR = (Q+1)_{10-23}$, no action otherwise:

$(Q)_{5-9} + LNG \longrightarrow (Q)_{5-9}$ (if ≤ 23)

else $(Q)_{5-9} + LNG - 24 \longrightarrow (Q)_{5-9}\ ,\ (Q)_{10-23} + 1 \longrightarrow (Q)_{10-23}$

$LC + 2 \longrightarrow LC$

$0 \longrightarrow A\ ,\ (ADR)_{I-J} \longrightarrow A_{K-23}$

The contents of the effective address and contents of the next location are both considered to be FD's. The second of these words is a limit. If the left bit of the field to be loaded is at the limit, no action occurs. Otherwise, the FD at the effective address is adjusted by the length of the field defined, the field is loaded into A and a skip occurs to signify that the field was loaded.

STFI STORE FIELD AND INCREMENT 14

If OFF = $(Q+1)_{5-9}$ and ADR = $(Q+1)_{10-23}$, no action otherwise:

$(Q)_{5-9}$ + LNG → $(Q)_{5-9}$ (if $\leq$ 23)

else $(Q)_{5-9}$ + LNG-24 → $(Q)_{5-9}$, $(Q)_{10-23}$ + 1 → $(Q)_{10-23}$

LC + 2 → LC

A_{K-23} → $(ADR)_{I-J}$

The contents of the effective address and the contents of the next location are both considered to be FD's. The second of these words is a limit. If the left bit of the field to be loaded is at the limit, no action occurs. Otherwise, the FD at the effective address is adjusted by the length of the field defined, the right most LNG bits in A are stored in the designated field, and a skip occurs to signify that the field was stored.

FIXED POINT ARITHMETIC

ADD ADD 3

A + (Q) → A

The contents of the effective address are added to the contents of the A register. The sum replaces the contents of the accumulator. A carry from bit position 0 of the 24-bit adder is or'ed with the carry flip-flop. If both operands are of the same sign but the sign of the result is different, the overflow flip-flop is set.

SUB SUBTRACT 3

A - (Q) → A

The two's complement of the contents of memory at the effective address is added to the contents of the A register. The sum replaces the contents of the accumulator. A carry from the adder is or'ed with the carry flip-flop. If the original operands have opposite signs, their difference may exceed the capacity of the accumulator. In this case, the overflow is set and the result in the accumulator is incorrect.

MPY MULTIPLY 20

A * (Q) → AB

The contents of the A register are multiplied by the contents of the effective address to form a 47 bit product in A and B. The sign and most significant part of the product are in the A register. The least significant part is in the B register. Bit 23 of B will be zero.

If both numbers have the value 40000000_8, overflow occurs and the computer sets the overflow indicator. This multiply considers both operands to be binary fractions, and produces a 47 bit fraction result. If two integer are to be multiplied, the correct integer result can be obtained by shifting A and B right one bit. B will now contain the least significant part of an integer result.

DIV DIVIDE 20

AB / (Q) → A, Remainder → B

The contents of the accumulator and the B register are treated as a double-precision dividend (47-bit fraction) and the contents of the effective address as a 24-bit fractional divisor. The quotient appears in the A register and the remainder in the B register. The sign of the remainder in the B register is the same as the sign of the original divident.

Division takes place normally if:

$$-1 \leq \frac{AB}{(Q)} < 1$$

If the quotient exceeds these boundaries, overflow occurs and the overflow indicator is turned on. If overflow occurs, the contents of A and B are unchanged. This division is a fractional division. To divide a 48-bit integer, the DIV instruction should be preceded by shift A and B left one bit. A 24-bit integer can be converted to a 48-bit integer by extending its sign by 24-bit positions.

ADM ADD TO MEMORY 4

A + (Q) → (Q)

The contents of memory at the effective address is added to the contents of the accumulator. The sum is placed into memory at the effective address. The contents of the accumulator remains unchanged. The overflow and/or carry flip-flops are unaffected.

MIN MEMORY INCREMENT 3

(Q) + 1 → (Q)

One is added to the contents of the effective address; the sum replaces the contents of the effective address. The carry and overflow flip-flops are unaffected.

MDS MEMORY DECREMENT, SKIP IF NEGATIVE 4

(Q) - 1 → (Q)

If (Q) < 0, LC + 2 → LC

Else LC + 1 → LC

One is subtracted from the contents of memory at the effective address, the difference replaces the contents of memory at the effective address. If the contents of memory at the effective address is negative after the subtraction, the computer skips the next instruction. If the contents of memory is positive or zero, the computer executes the next sequential instruction. The carry and overflow are unaffected.

ADX ADD TO INDEX 3

X + (Q) → X

The contents of the effective address are added to the contents of the index register. The sum is then placed in the index register. The carry and overflow are unaffected by this instruction.

FLOATING POINT ARITHMETIC

FAD FLOATING ADD 18

$AB_F + (Q, Q + 1)_F \longrightarrow AB_F$

The floating point number at the effective address is added to the floating point number in A and B. The result will be normalized regardless of whether the operands were normalized. If exponent overflow occurs, the floating point trap will be taken. If exponent underflow occurs, A and B will be set to 0 and no interrupt will occur. Floating point addition and subtraction operations are performed on 48 bit fractions formed by separating out the exponent and replacing it with nine low-order zeroes. The appropriate fraction is shifted to align exponents and the addition or subtraction performed. The result is then normalized and the exponent corrected. The high order bit of the fraction discarded is saved in the CARRY flip-flop for use in rounding. The exponent is truncated to 9 bits and replaced in the exponent field. If exponent overflow has occurred, the floating point trap will be taken. The exponent will be in error by 1000_8.

FSB FLOATING SUBTRACT 18

$AB_F - (Q, Q + 1)_F \longrightarrow AB_F$

The floating point number at the effective address is subtracted from the floating point number in A and B. The result and exception conditions are as given under floating aid.

FMP FLOATING MULTIPLY 27

$AB_F * (Q, Q + 1)_F \longrightarrow AB_F$

The floating point number at the effective address is multiplied by the floating point number in A, B. The result on overflow and underflow is as described under floating add. The most significant bit of the discarded portion of the product is saved in the carry flip-flop. The operands are expected to be normalized; therefore, at most one bit of post normalization will occur.

FDV FLOATING DIVIDE 40

$AB_F / (Q, Q + 1)_F \longrightarrow AB_F$

The floating point number at the effective address is divided into the floating point number in A, B. If the divisor is zero or unnormalized,

the floating point trap will be taken with A and B unchanged. The overflow flip-flop will be set. Otherwise, overflow and underflow are handled as under floating add.

UFA UNNORMALIZED FLOATING ADD 14

$$AB_F + (Q,\ Q+1)_F \longrightarrow AB_F \text{ (without normalization)}$$

This instruction is performed exactly as floating add except that the normalization step is omitted. Exponent underflow cannot occur.

UFS UNNORMALIZED FLOATING SUBTRACT 14

$$AB_F - (Q,\ Q+1)_F \longrightarrow AB_F \text{ (without normalization)}$$

This instruction is performed exactly as floating subtract except that the normalization step is omitted. Exponent underflow cannot occur.

LOGICAL

AND LOGICAL AND 3

$A \cap (Q) \longrightarrow A$

This instruction performs a logical AND of the contents of memory at the effective address and the contents of the accumulator and places the result in the accumulator.

ORA LOGICAL OR 3

$A \cup (Q) \longrightarrow A$

This instruction performs the logical OR of the contents of memory at the effective address and the contents of the accumulator and places the result in the accumulator.

EOR EXCLUSIVE OR 3

$\{A \cap (\overline{Q})\} \cup \{\overline{A} \cap (Q)\} \longrightarrow A$

This instruction performs the exclusive OR of the contents of memory at the effective address and the contents of the accumulator and places the result in the accumulator.

SKL SKIP IF A IS LESS THAN OR EQUAL TO MEMORY

IF $A \leq (Q)$, LC + 2 → LC

The A register is compared to memory at the effective address. If A is algebraically less than or equal to the contents of the effective address, the next instruction is skipped. Otherwise, the next sequential instruction is taken.

SKB SKIP IF LOGICAL AND OF B AND MEMORY IS ZERO 4

IF $B \cap (Q) = 0$, LC + 2 → LC

If a logical AND performed on the contents of memory at the effective address and the B register produces a zero result, the computer skips the next instruction. If a logical AND produces a one bit in any position, the computer takes the next sequential instruction. The contents of B and memory are not affected by this instruction.

EXAMPLES:

Q	EXPLANATION
00000001	Skip if B is even
77777777	Skip if B = 0
40000000	Skip if B positive
40000001	Skip if B positive and even

SKA SKIP IF LOGICAL AND OF A AND MEMORY IS ZERO 4

IF $A \cap (Q) = 0$, LC + 2 → LC

If a logical AND performed on the contents of memory at the effective address and the A register produces a zero result, the computer skips the next instruction. Otherwise, the computer takes the next sequential instruction.

SKC SKIP ON FLAG AND CLEAR 4

IF $(Q) \geq 0$, LC + 2 → LC

$-1 \rightarrow (Q)$

If the word at the effective address is positive a skip occurs. Otherwise the next sequential instruction is taken. In either case, the word at the effective address is set to -1 (77777777_8).

SKN SKIP IF MEMORY NEGATIVE 4

IF (Q) $<$ 0, LC + 2 $\longrightarrow$ LC

If the word at the effective address is negative, the next instruction is skipped. Otherwise, the next sequential instruction is taken.

SKP SKIP IF MEMORY POSITIVE 4

IF (Q) $\geq$ 0, LC + 2 $\longrightarrow$ LC

If the word at the effective address is positive, the next instruction is skipped. Otherwise, the next sequential instruction is taken.

BRANCHING

BRU BRANCH UNCONDITIONAL

$Q \longrightarrow LC$

This instruction causes an unconditional transfer to the location specified by the effective address.

BSL BRANCH AND SAVE LINK

$LC + 1 \longrightarrow ((Q))_{10-23}$

$MAP \longrightarrow ((Q))_{0}$

$OV \longrightarrow ((Q))_{3}$

$CARRY \longrightarrow ((Q))_{4}$

$Q + 1 \longrightarrow LC$

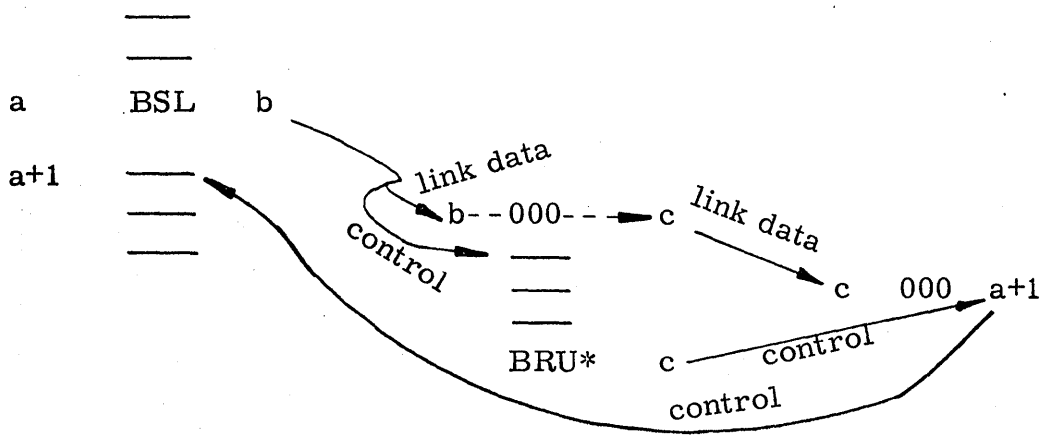
$0 \longrightarrow CARRY$

$0 \longrightarrow OV$

This instruction provides the entry mechanism for reentrant subroutines. The location of the next instruction is stored into bits 10-23 of the link word specified by the address of the memory word at the effective address. Bit 0 of the link word is set to zero if the machine is operating in the monitor mode and is set to one if the machine is in the user mode. Bits 3 and 4 of the link word are set to the contents of the overflow indicator and the carry flip-flop respectively. The overflow indicator and the carry flip-flop are then cleared and the computer branches to the effective address plus one.

Thus, the instruction BSL Y at location P first looks in Y to find a link address Z. The P + 1 is then stored in Z which is outside the body of the subroutine. Control is then transferred to Y + 1. Return to the calling program is accomplished with indirect branch through Z.

Example:



Note that the address saved is the address of the next instruction to be executed. This is a general policy which applies whether the BSL is executed

1. Directly
2. Indirectly
3. By an interrupt
4. By a programmed operator

When the BSL is executed via a programmed operator, one additional step occurs. This is the computation and saving of the effective address of the programmed operator in the location following the link word. This will expedite finding operands in the programmed operator routine.

BIX BRANCH AND INCREMENT INDEX 4

$X + 1 \rightarrow X$; IF $X < 0$, $Q \rightarrow LC$

This instruction adds one to the contents of the index register. If the index register is negative, branch to the effective address. If the index register is positive, the computer takes the next sequential instruction.

BRI BRANCH AND RESET INTERRUPT 3

$(Q)_{10-23} \rightarrow LC$

$(Q)_3 \rightarrow OV$

$(Q)_4 \rightarrow CARRY$

Clear Current Priority Interrupt

The address portion of memory at the effective address is placed into the location counter. The overflow indicator and the carry flip-flop bits are replaced by the contents of bits 3 and 4, respectively, of the location specified by the effective address. This instruction also terminates the current priority interrupt. This instruction may be used to return from a subroutine when it is desired to restore the overflow and carry.

BDX BRANCH AND DECREMENT INDEX 4

$X - 1 \rightarrow X$, IF $X \geq 0$, $Q \rightarrow LC$

This instruction subtracts one from the index register. If the result is positive, branch to the effective address. If X is negative, proceed to the next sequential instruction.

INPUT-OUTPUT AND CONTROL (PRIVILEGED)

ACT ACTIVATE (P) 10

The 14 bits of effective address are used for setting various internal computer conditions and for controlling the peripheral devices. The basic function is selected by bits 10 and 11 as follows:

10	11	
0	0	Unassigned
0	1	Unassigned
1	0	Set internal condition
1	1	Set external condition

Interpretation of bits 12-23 will be discussed under the sections on "Setting Internal Conditions" and "Setting External Conditions."

POT PARALLEL OUTPUT (P) 3

(Q) → Parallel Output Lines

The contents of the effective address is brought to the storage register and held, awaiting transfer to an external device. This instruction allows up to 24 bits to be transmitted in parallel to an external device.

PIN PARALLEL INPUT (P) 2

(Parallel Input Lines) → Q

Twenty-four parallel bits are input into the contents of the memory location specified by the effective address.

MISCELLANEOUS

EAX EFFECTIVE ADDRESS INTO INDEX 2

$Q \rightarrow X_{10-23}$

The effective address is placed in the address field of the index register. Bits 0-9 of the index register are unaffected.

XEC EXECUTE 1+

(Q) → Instruction Register

The instruction at the effective address is executed. This instruction does not alter the location counter unless the instruction it executes changes the location counter. If a skip instruction is executed, the skip occurs relative to the XEC instruction.

XCI EXECUTE INDIRECT 5+

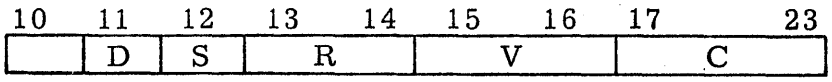
((Q)) → Instruction Register
(Q)+1 → (Q)

The instruction which is addressed by the contents of the effective address is executed and the contents of the effective address are incremented by one. If a skip instruction is executed, the skip occurs relative to the XCI instruction. If the instruction executed is an unconditional branch or a conditional branch for which the branch conditions are satisfied, the location counter is incremented by three and the branch is suppressed.

The purpose of this instruction is to simplify tracing routines and debugging routines which can execute a sequence of instructions without fear of losing control. If the monitor map is in effect, a 1 in the sign bit of the word at the effective address causes the instruction to be fetched using the user map.

SHF SHIFT 7

All shifts in the computer are performed by one instruction. The type of shift is determined by the address field of the instruction. The address field has the following format:



Where

D Specifies the shift direction
0 - Left
1 - Right

S Specifies the type logical or arithmetic
0 - Logical. The overflow indicator is unaffected by this instruction.
1 - Arithmetic. On right shifts the sign bit is not shifted but is copied into vacated bit positions. Bits shifted out of the right bit of each active register are lost. Overflow is set if the sign bit of the A register changes during the shift.

R Specifies the active registers
0 - A and B are taken as a single 48 bit register
1 - A only is specified
2 - B only is specified
3 - A and B are both shifted but are treated as two independent 24 bit registers.

V Specifies the action to be taken on vacated bit positions
0 - Shift in 0's
1 - Shift in 1's
2 - Shift in bits shifted out from other end of register (cycle). (Sign bit on arithmetic right).
3 - Shift in complement of bits shifted out from other end of register. (Complement of sign bit on arithmetic right).

C Shift Count. The shift count is a seven bit two's complement count, $-63 \leq C \leq 63$. If C is negative, the direction of the shift indicated by the D field is reversed. If indexing is specified, the sign of C is extended to form a 24 bit two's complement number which is added to the contents of the index register to yield a 24 bit shift count

Shift instructions may be indirectly addressed. If indirect addressing is specified, the last word in the chain contains the shift specification fields.

OPR OPERATE

The operate instruction is used to perform many functions. Since no memory reference is made, the effective address is used to specify the operations performed.

This instruction contains the following instructions as determined by bits 10-12 of the effective address:

10	11	12		
0	0	0	SWP	Swap Registers
0	0	1	LRO	Logical Register Operate
0	1	0	ARO	Arithmetic Register Operate
0	1	1	RIN	Register Increment
1	0	0	BTO	Bit Test and Operate
1	1	1	PFM	Perform

Bits 13-23 of the address are treated for each subinstruction as discussed in the following section.

SWP SWAP REGISTER 5

This instruction permits the general exchange of the A, B, and X registers. In addition, any or all may be cleared or set to -1 (77777777_8).

The values to be placed in the A, B, and X registers are independently specifiable. The three bit fields which control the final contents of the registers are:

Bits	
15 - 17	A Register
18 - 20	B Register
21 - 23	X Register

The interpretation of each three bit field is as follows:

Bit 1	0	- Transfer true
	1	- Transfer complement
Bit 2 - 3	00	- Source register is zero
	01	- Source register is A
	10	- Source register is B
	11	- Source register is X

When this instruction is executed, any register which is to remain unchanged should specify a transfer to itself.

LRO LOGICAL REGISTER OPERATE 4

This instruction allows the formation of any of the sixteen possible bit wise logical functions of two registers and place the result in any register. In this instruction two source registers, S1 and S2, and a destination register, D, are specifiable independently by the three 2 bit fields:

Bits	
14 - 15	S2
16 - 17	S1
18 - 19	D

In each field the register designated by the bits is the same as in SWP:

0 0	Zero
0 1	A
1 0	B
1 1	X

Bits 20-23, called B1, B2, B3, and B4 respectively control the formation of the result as follows:

$$(B1 \cap S1 \cap S2) \cup (B2 \cap S1 \cap \overline{S2}) \cup (B3 \cap \overline{S1} \cap S2) \cup (B4 \cap \overline{S1} \cap \overline{S2}) \longrightarrow D$$

If the zero (empty) register is specified as a destination register, the result is discarded and no action occurs.

ARO ARITHMETIC REGISTER OPERATE 4

This allows for forming the sum or difference of any two registers placing the result in any register and performing a versatile skip test on the result.

The source (S1 and S2) and destination (D) registers are specified by the same fields in the same way as under LRO.

Bit 13 controls addition or subtraction:

Bit 13 = 0 Add (S1 + S2 → D)
1 Subtract (S1 - S2 → D)

Bits 20-23 control the testing of the result in the following way:

Bit	20	21	22	23
Skip if result is	< -1	= -1	= 0	> 0

Some of the conditions which may be tested by appropriate bit settings are:

	20	21	22	23
No test or skip	0	0	0	0
Skip on zero	0	0	1	0
Skip on positive	0	0	1	1
Skip on less than or equal to zero	1	1	1	0
Skip on mixed ones and zeros	1	0	0	1

Since the result can be discarded (D = 00), the sum or difference of registers, or a single register (S1 or S2 = 00), may be testing without altering them.

RIN REGISTER INCREMENT 4

This instruction is similar to the previous instruction ARO except that instead of using the S2 field to select a register the contents of the S2 field are used as an immediate operand. In this way a register may be incremented, or decremented by 0, 1, 2, or 3, and tested by a single instruction. Other than the change above, all fields are interpreted as in ARO.

BTO BIT TEST AND OPERATE 2

This instruction allows the selection and testing of the CARRY and OVERFLOW flip-flops and setting them to desired values.

The bits used to select the flip-flops is Bit 19 as follows:

Bit 19 = 0 Overflow
= 1 Carry

FIX CONVERT TO FIXED POINT 8

$AB_F \longrightarrow AB_I$

The normalized floating point number in the accumulator and extended accumulator is converted to a 48 bit integer in A and B. If the exponent of the floating point number is less than or equal to 0, the accumulator and extended accumulator are cleared. If the exponent is greater than 47, the overflow indicator is turned on and exit is made with the accumulator and extended accumulator unchanged. The most significant fraction bit is stored in the carry flip-flops.

Example:

	A	B	CARRY
Before Execution	24500000	00000005	0
After Execution	00000000	00000024	1

FRND FLOATING ROUND 8

$(A, B)_F + (CO) \longrightarrow (A, B)_F$ Adjust Exponent

The contents of the carry flip-flop is added to the low order bit of the fractional part of the floating point number in A, B. If the addition of the carry caused the fraction to overflow, the fraction is shifted right one place and one is added to the exponent. The resulting normalized floating point number is placed in A, B and the carry flip-flops is turned off.

FNEG FLOATING NEGATE 8

$-(A, B)_F \longrightarrow (A, B)_F$

The negative of the floating point number in A, B replaces the contents of A, B. Since it may be necessary to shift the fraction by one bit, either overflow or underflow can occur and will be treated as under floating add.

ANRM A NORM FD 14

$A_{10-23} * 24 + A_{5-9} \longrightarrow A$

This instruction converts a field descriptor (Words + Offset) into a bit count in A. It can be used to convert a string length pointer into the length of the string in bits. The length of the string in characters may then be obtained by shifting A right 3 bits (Divide by 8).

BNRM B NORM FD 14

$A/24 \longrightarrow A_{10-23}$, Remainder $\longrightarrow A_{5-9}$

This instruction converts a bit count in A into a FD with zero length field in A. (i.e. $0 \leq A_{5-9} \leq 23$)

NRM NORMALIZE FD 4

1. If $A_0 = 1$, $A + 01337777_8 \longrightarrow A$

   If $A_0 \geq 014_8$, $A - 01337777_8 \longrightarrow A$

2. If $0 \leq A_{0-9} \leq 23$, $0 \longrightarrow OV$

   Otherwise $1 \longrightarrow OV$

This instruction is used to restore a FD in A to normalized form after two FD's with zero LNG fields are added or subtracted. One normalization step is taken. If the result is normalized the overflow indicator is turned off. If the result is not normalized, the overflow indicator is turned on.

LLO LOCATE LEADING ONE 2

The bit position of the first (left most) one bit in the A and B registers is placed in the X register. The sign bit of the A register is bit 0 and the least significant bit of the B register is bit 47. If there are no one bits in A or B, X is set to -1.

LLZ LOCATE LEADING ZERO 2

The bit position of the first (left most) zero bit in the A and B registers is placed in the X register. If no bits are zero, X is set to -1.

LLT LOCATE LEADING TRANSITION 2

The bit position of the first zero bit followed by a one, or the first one bit followed by a zero, is placed in the X register. If all bits of both A and B are either 0 or 1, X is set to -1. Since bit 23 of B is always assumed to be followed by an identical bit, the largest number which can be set into X is 46.

CNT COUNT BITS 4

The number of bits in A and B which are 1 is placed in the X register. The result is X will therefore lie between 0 and 48 inclusive.

PRIORITY INTERRUPTS

The SCC 6700 has 16 priority interrupt channels as standard equipment, which may be selectively armed or disarmed and selectively enabled and disabled when disarmed interrupt conditions are ignored. When disabled, however, interrupt conditions are recorded for future response. Additional priority interrupt channels may be added in blocks of 16 channels as optional equipment. Each interrupt channel is numbered. The number assigned to a channel is determined by the location from which the computer obtains an instruction to be executed when interrupted by the given channel.

There are three flip-flops associated with each interrupt channel. Two of these flip-flops indicate the status of the corresponding channel and have the following meaning:

RFF	PFF	STATUS
0	0	Interrupt inactive
1	0	Interrupt requested but not being processed (Waiting)
1	1	Interrupt requested and being processed (Active)
0	1	(Not Used)

The third flip-flop is set if the active interrupt is to remain active until a BRI instruction has been executed and reset if the interrupt is to be cleared after a single instruction. A BSL instruction in the interrupt cell will set this flip-flop for a multi-instruction interrupt.

If the instruction at the interrupt location is any instruction other than a BSL, the interrupt is cleared immediately after executing the instruction. If the instruction in the interrupt location does not alter the program counter, the effect of an interrupt is that the instruction is effectively inserted into the program at the point of interrupt.

When an interrupt causes the computer to execute the location specified by the channel number, the location counter is not altered.

A BSL instruction may then be used to save the address of the interrupted program while branching to the interrupt routine.

The currently active interrupt is cleared and the interrupt channel flip-flops are reset by a BRI instruction.

When an interrupt for a given channel is being processed, a higher priority channel may interrupt the processing subroutine. If this situation occurs, the interrupted channel remains in the active status. After the higher priority interrupt has been processed, return is made to the subroutine processing the original interrupt. If an interrupt on a given channel is being processed, any new interrupt requests to that channel will not be honored, i.e., no interrupt will occur.

The lower number channels have higher priority, hence, given two channels, M and N, M $>$ N implies channel N has priority over M.

Interrupts may be placed in the Waiting Status by an appropriate ACT command. This feature may be used for both hardware and software checking, as well as simplifying certain I/O routines.

SYSTEM TRAPS

Illegal user actions are detected in the 6700 by the system trap mechanism. The system trap is similar to the interrupt in that a trap causes the instruction in a fixed location in memory to be executed. In fact, a trap may be viewed as an interrupt with a specifiable priority.

The trap conditions and the locations executed at their occurrence are as follows:

Location	Condition
40 -	Privilege Error
41 -	Undefined Op-Code
42 -	Write Error
43 -	Read Error
44 -	Execute Error
45 -	Floating Point Overflow
46 -	Interval Time Trap
47 -	Non-existent Memory
50 -	Return to User Map
51 -	Parity Error
52 -	FD Error

At the occurrence of a trap the BSL saves the location of the offending instruction, including the case of an attempted jump to an out of bounds location.

TIME SHARING INPUT/OUTPUT CONTROLLER

Each time sharing input/output controller connects 16 remote devices to the 6700 via the parallel input/output system. One I/O controller is included as standard equipment. Additional controllers may be added when a large number of remote devices is required.

The teletype interface permits the transfer of 11-unit, 10-character-per-second teletype information between the SCC 6700 computer and 32 Model 35 Teletype printer keyboards. Each of the 32 lines is full duplex and all of the lines can be active simultaneously. The interface operates through the parallel I/O connector of the computer and uses two of its interrupt locations.

Each of the 32 teletypes has a transmit and a receive character buffer which perform all necessary serial-to-parallel and parallel-to-serial operations and provide the necessary control timing. Flags associated with each transmit and receive buffer indicate when a character has been transmitted or received. The flags are continously scanned by a scanning unit within the interface. Upon encountering a transmit (receive) flag, the scanner is stopped and an interrupt unique to the transmit (receive) operation is issued. At this time the scanner register contains the address of the raised flag. The scanner is subsequently restarted when the computer reads the character into (out of) the corresponding buffer.

A program option is provided to allow for the suppression of a particular transmit flag, thus prohibiting an interrupt at the completion of the transmit operation.

Instructions

The following is a list of instructions which pertain to the teletype interface:

SELECT TELETYPE INTERFACE - STI

This instruction selects the teletype interface; i.e., the execution of this instruction causes the interface to be electrically connected to the I/O connector. The interface will remain selected until another ACT instruction is received.

OUTPUT CHARACTER AND SET INTERRUPT CONTROL - POT

This instruction transfers a word from the specified memory location to the teletype interface. The transferred word contains a TTY address, an interrupt specification and, in appropriate cases, a character to be transmitted. The format of this word is as follows:

BITS

0-7	The character to be transmitted
8	= 1 if the character (Bits 0-7) is not to be transferred to the transmit buffer
	= 0 if the character is to be transferred
9	= 1 if an interrupt is to occur at the completion of the transmission
	= 0 if the interrupt is not to occur
10	= 1 if the input and output interrupts are to be enabled (see the Operation Section for details).
	= 0 otherwise
11-14	Not interpreted
15-23	TTY address (256 may be addressed).

It should be noted that if Bit 8 of the transmitted word is one, this instruction can be executed at any time without affecting the specified transmit buffer.

READ TELETYPE ADDRESS AND DATA - PIN

This instruction transfers a word from the teletype interface into the memory location specified. When this instruction is executed in response to a teletype input or output interrupt, the word transferred into memory will contain the address of

the teletype causing the interrupt and, in the case of an input interrupt, it will also contain the inputted character. The format of this word is as follows:

BITS

0-7	The inputted character when responding to an input interrupt and zero when responding to an output interrupt
	zero
19-23	The address of the teletype causing the interrupt

Operation

As a result of pressing the start button on the CPU, the teletype interface is initially in a state such that it will not send interrupt requests to the CPU. The normal operating state is effected by outputting a word with Bit 10 equal to one. This word may also contain further information such as a character to be transmitted. When in the normal operating state, the interface will issue a receive interrupt request each time the scanner encounters a receive flag (indicating a character is in the receive buffer) and will issue a transmit interrupt request each time a transmit flag, together with its interrupt control flag, is encountered (indicating that the transmit buffer is clear and that the interrupt specification requested an interrupt at the completion of transmission).

When responding to a receive interrupt, it is necessary only to supplement the normal interrupt servicing routine with an ACT-PIN combination to effect the input operation.

When responding to a transmit interrupt, it is first necessary to input the teletype address with an ACT-PIN combination and then to output the interrupt specification and the character to be transmitted with an ACT-POT combination.

When transmitting a character at an arbitrary time (not in response to an interrupt), the ACT-POT combination should be preceded by a skip instruction to assure that the transmit

buffer is not busy. If only the interrupt specification is to be set (Bit 8 of the output word is one), then the ACT-POT combination can be executed at any time since the interrupt specification is independent of the transmit buffer.

Interrupt Procedure

The scanner is halted every time that a receive flag or a transmit flag, together with its interrupt control flag, is encountered and will remain halted until the flag is cleared by the execution of an RDP instruction in the case of a receive flag, or by a WTP instruction in the case of a transmit flag. During the time that the scanner is stopped, an interrupt request corresponding to the type of flag encountered (transmit or receive) will be sent to the CPU.

As a result of this scanning procedure and of the priority interrupt system of the CPU, the following facts hold:

1. The receive interrupt routine will not be interrupted by either a TTY transmit or receive interrupt.

2. The transmit interrupt routine will not be interrupted by a transmit interrupt and can be interrupted by a receive interrupt only after the transmit flag causing the interrupt has been cleared.

APPENDIX A
6700 INSTRUCTION LIST

Loads/Stores

LDA - Load A Register
STA - Store A Register
XMA - Exchange Memory and A
LDB - Load B Register
STB - Store B Register
LDX - Load Index Register
STX - Store Index Register
XMX - Exchange Memory and X
STM - Store under Mask
LDD - Load Double
STD - Store Double

Field Loads/Stores

LDF - Load Field
STF - Store Field
LDFX - Load Field Indexed
STFX - Store Field Indexed
LDFI - Load Field and Increment
STFI - Store Field and Increment

Logical Operations

AND - Logical AND to A
ORA - Logical OR to A
EOR - Logical Exclusive OR to A

Branching

BRU - Branch Unconditional
BSL - Branch and Save Location
BIX - Branch-Increment X
BDX - Branch-Decrement X
BRI - Branch and Restore Interrupts

Fixed Arithmetic

ADD - Add to A
SUB - Subtract from A
MPY - Multiply
DIV - Divide
ADM - Add A to Memory
MIN - Memory Increment
MDS - Memory Decrement and Skip
ADX - Add to Index Register

Floating Point Arithmetic

FAD - Floating Add
UFA - Unnormalized Floating Add
FSB - Floating Subtract
UFS - Unnormalized Floating Subtract
FMP - Floating Multiply
FDV - Floating Divide

Skip Tests

SKE - Skip on A Equal to Memory
SKU - Skip Unequal
SKG - Skip on A Greater
SKL - Skip on A Less or Equal
SKEM - Skip on Masked Equality
SKUM - Skip on Masked Unequal
SKN - Skip if Memory Negative
SKP - Skip if Memory Positive
SKA - Skip on A and Memory Zero
SKB - Skip on B and Memory Zero
SKC - Skip and Clear Flag

Miscellaneous

EAX - Effective Address to X
SHF - Shift
XEC - Execute
XCI - Execute Indirect
OPR - Operate Microinstruction
ANORM- FD into Bit Length

Miscellaneous continued

BNORM - Bit Length Into FD
NORM - Normalize FD
FIX - Float to Fix Conversion
FLT - Fix to Float Conversion
FNG - Floating Negate
FRD - Floating Round
SWP - General Register Swap
AOP - Arithmetic Ops on Registers
LOP - Logical Ops on Registers
RIN - Register Increments and Test
and many others

I/O (Privileged)

ACT - Activate
POT - Parallel Output
PIN - Parallel Input

6700 INSTRUCTION MAP

	0	1	2	3	4	5	6	7
0	TRP *	POT *	ACT *	PIN *	XMA		XMX	STM
1	LDA	LDB	LDX	LDD	STA	STB	STX	STD
2	OPR	LDF	LDFI	LDFX	SHF	STF	STFI	STFX
3	EAX	AND	ORA	EOR	FAD	FMP	FSB	FDV
4	SKE	SKU	SKG	SKL	ADD	MPY	SUB	DIV
5	SKEM	SKUM	SKN	SKP	ADM	MIN	MDS	ADX
6	SKA	SKB	SKC		UFA	XEC	UFS	XCI
7	BRU	BSL	BIX	BDX	BRI			

APPENDIX B
SCC 6700
TIME-SHARING SOFTWARE

INTRODUCTION

The SCC 6700 Time-Sharing Software is designed to take full advantage of the advanced hardware concepts presented by the SCC 6700. The software system consists of three major parts: the monitor, the executive and the subsystems. In general, the time-sharing system provides the following facilities.

1. Mutual protection of the users and time system against one another.

2. Facilities to allow users to communicate with one another via the computer.

3. Approximately equal allocation of the computing facilities to the current users.

4. Software packages necessary to permit one program to control others -- with overall control by the user from his terminal.

5. Software which permits communication between computer and peripherals without regard to the latter's special physical peculiarities.

6. A file management system for user's data sets and programs.

7. Response to a variety of requests that arise naturally in the course of a user's connection with the system.

SYSTEM MONITOR

The system monitor performs the functions of scheduling, input/output, interrupt processing, memory allocation and swapping, and the control of active programs. The monitor permits each program to be run a fixed period of time or until an input/output request is made. Following the occurrence of either of these events, the program is dismissed and a new program activated. If an active program requires memory presently occupied by a dismissed program, the user status register is checked to see if the page in question has been altered. If the page has been altered, it is put on the drum; if it has not been altered, swapping to the drum is recognized as being unnecessary.

All user input/out requests are handled by the monitor. The user is provided with a comprehensive file handling capability. The file handling programs make it unnecessary for the user to have detailed programming knowledge of the input/out device he is using. Special file handling capabilities are provided for the teletype since this is the most common user console. In particular, the file handler provides full duplex teletype input/output, in which a character typed on the keyboard is sent to the computer but is not printed. After receiving an input character, the file handler may output the received character to the teletype to be printed, thus creating an echo, or it may transmit some other character(s) to be printed. The full duplex capability permits the user to substitute a character or a string of characters for some character which is typed. Most important, it relieves the user from having to wait for a type-out to finish before being able to type information.

SYSTEM EXECUTIVE

The System Executive provides the command language by which the user controls the system from his teletype, the identification of users and specification of the limits of their access to the system, the control of the directory of symbolic file names and back-up storage for these files and several other miscellaneous matters. The executive provides user file security by denying unauthorized file access by other users.

The user controls the system from his teletype with the Executive Command language. The executive commands fall into seven broad classes:

1. Commands governing entry to and exit from the system.
2. Commands controlling the allocation of memory.
3. Commands relating to the interaction of teletypes.
4. Commands to control the handling of the files.
5. Miscellaneous commands.
6. Commands to call subsystems.
7. "Systems" commands.

Considerable flexibility is offered in the area of interaction between teletypes. Provision is made for linking several teletypes enabling a user to transmit a message to many teletypes. The user is given the capability to converse with someone on another teletype via the CONSULT WITH command. The executive also provides commands which enable a user to find out on which teletype a particular user is currently entered or who is using a particular teletype.

SYMBOLIC MACRO-ASSEMBLER

The Symbolic Macro-Assembler will transform a symbolic file consisting of a symbolic machine-language program into a binary file which can be loaded directly into core using the loader in the DDT sybsystem. The assembler also provides DDT with the symbolic tables necessary for symbolic level debugging. The assembler includes a powerful mechanism for expanding user-defined macros as well as a large variety of options which include the listing, skipping and repeating of parts of the assembly.

TIME-SHARING DEBUGGING SYSTEM (DDT)

DDT is the loading and debugging subsystem for SCC 6700 programs. Binary files prepared by the assembler can be loaded into core by DDT and executed under its continuous supervision. Other facilities include interrogating and changing memory locations, scanning memory for the specified digit patterns, inserting patches and breakpoints and performing traces.

HELP

This is a question-answering service intended to obviate the necessity of referring to a manual to resolve any small difficulties which arise while using the time-sharing system. Questions about the system may be put, via the teletype, in fairly free format, conversational English. HELP may be entered from any subsystem and is then set up to recognize and answer questions about that subsystem.

CONVERSATIONAL ALGEBRAIC LANGUAGE (CAL)

The "Conversational Algebraic Language" allows numerical computations to be performed interactively. That is to say, mathematical calculations can proceed under the continuous supervision of the user. Facilities are available for compiling and running complete programs delivered in a stylized semi-conversational form, as well as for carrying forward computations in short steps with printouts of intermediate results.

TIME SHARING EDITOR (Q. E. D.)

Q. E. D. is a rather powerful program for editing symbolic text which runs under the 6700 time sharing system. Its input and output are symbolic files which can also be handled by the executive COPY command. It has extensive facilities for inserting, deleting and changing lines of text, a line edit feature, a powerful symbolic search feature, automatic tabs which may be set by the user, and ten string buffers. Text can be read from any file and written onto any file. A replace command permits all occurrences of a specified string of characters to be replaced with another string.

TIME SHARING STRING PROCESSING SYSTEM (SPS)

The String Processing System is a collection of subroutines which perform operations on strings of characters. SPS provides string comparison operations, string output commands, and string manipulation via a hash table which facilitates table look-ups when the operand is a string.

SNOBOL

SNOBOL is a language whereby strings of alphanumeric characters can be manipulated. Specified strings can be input/output, scanned for the existence of possibly broken sequences of characters with specified properties, compared to be "greater than" or "less than" one another; substitutions, reversal and transplantation of groups of characters can also be done. Strings of decimal digits may be interpreted as numbers and some simple arithmetic may be done with them.

LISP

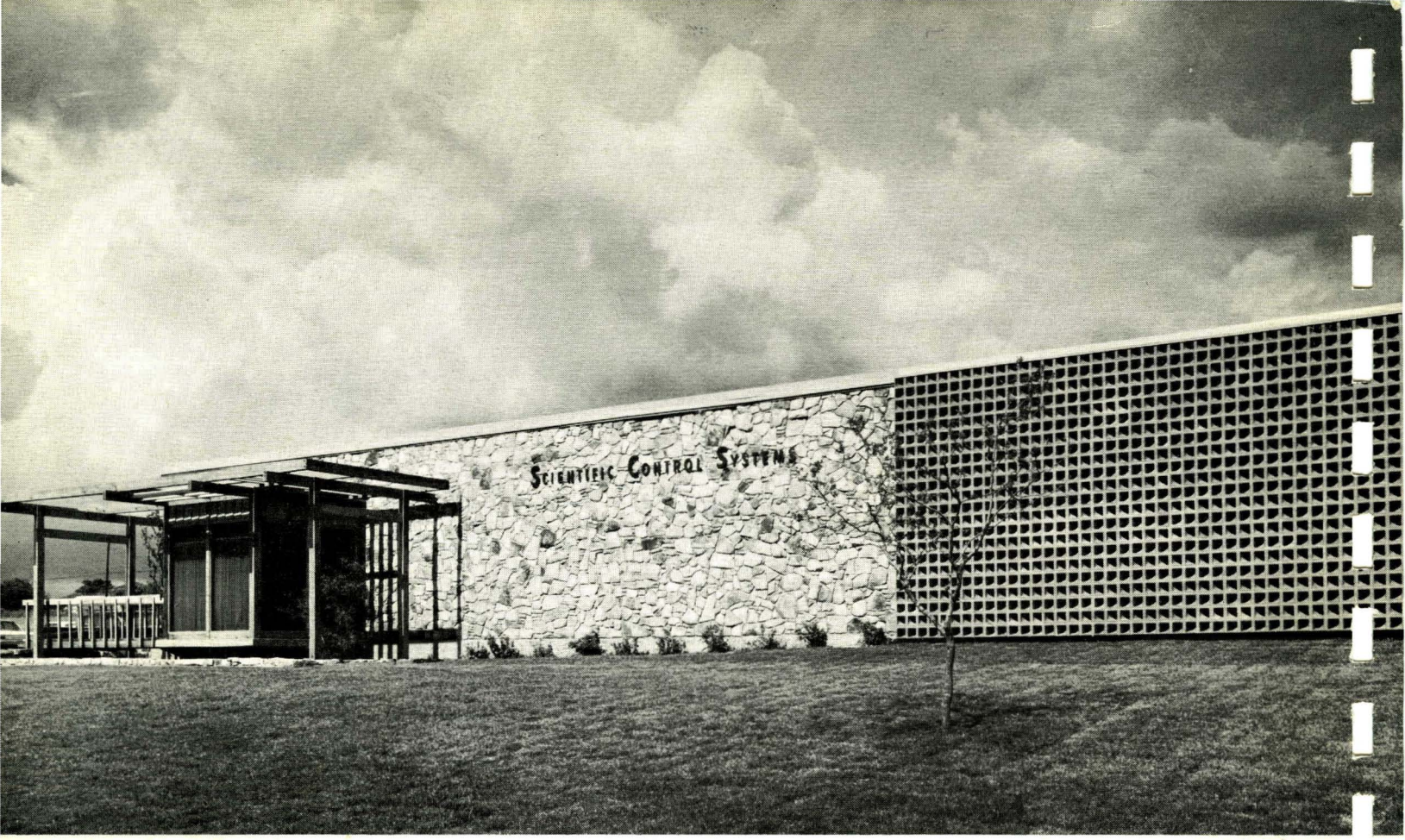
LISP is a general-purpose List Processing System. It can, with greater or lesser efficiency, perform user-defined operations on any set of entities capable of being represented as lists either of other lists or, ultimately, of a finite number of distinguishable elements. Problems which are recursive by nature, i.e., in which the definition of a computable entity involves the entity itself, are particularly susceptible to attack using LISP.

FORTRAN

FORTRAN, actually two processors, the 6700 Fortran System consists of a high efficiency compiler plus an interpretive (conversational) version with outstanding diagnostic facilities. Accepting exactly the same language, full ASA Fortran, the translators of the 6700 Fortran system greatly simplify the construction of efficient, correct Fortran programs.

TMG

TMG, a syntax-directed translator for the do-it-yourself fan or the user with an unusual problem, TMG provides easy programming of both batch and incremental translators. The Symbolic output of TMG may be used as input to any of the processors (including of course, TMG).



SCC maintains complete support activities for its users. Installation and maintenance services are available through SCC offices strategically located throughout the United States. For pre-procurement demonstration of hardware and programs in Dallas, contact local sales office or the Marketing Department in Dallas.

Arlington, Massachusetts
30 Park Avenue
617 — 648-2922
(Boston)

Skokie, Illinois
125 Old Orchard Arcade
312 — 675-6700
(Chicago)

Houston, Texas
7800 Westglen Drive
713 — 782-9851

Seattle, Washington
1806 South Bush Place
206 — 324-7911

Midland Park, New Jersey
36 Central Avenue
201 — 652-6750
(New York)

Crofton, Maryland
Village Green
301 — 647-6431
(Baltimore)

Orlando, Florida
2319 E. South Street
305 — 841-3556

Pasadena, California
180 East California Blvd.
213 — 681-2651
(Los Angeles)

Other SCC products include: telemetry systems and airborne signal conditioning equipment such as amplifiers, demodulators and converters.

Scientific Control Corporation

14008 Distribution Way • Dallas, Texas 75234 • 214 — 241-2111

Printed in U.S.A.