SUE

PROCESSOR

INSTRUCTION

SET

G3

GB13020009103

SUE PROCESSOR INSTRUCTION SET

GENERAL SYSTEM BULLETIN G3

Third Edition

This bulletin supercedes
SUE Processor Instruction Set
General System Bulletin G3, Rev. A
dated June 1972

Bulletin GB13020009103
May 1973

EFFECTIVE PAGES

New pages introduced in this third edition include Processor
Instruction Sets for SUE 1110A/B, 1111A/B, 1112A/B, and
Appendices D and E.

Changes in the second edition, which included Processor
Instruction Set SUE 1110, and Appendices A through C, are
indicated by a heavy line in the outer margin of the changed
page.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

LIST OF TABLES

PREFACE

This bulletin contains instructions to program seven types of SUE processors:

	Number of Instructions
SUE 1110 (basic)	108
SUE 1110A	109
SUE 1110B	111
SUE 1111A, Decimal Arithmetic	118
SUE 1111B, Decimal Arithmetic	120
SUE 1112A, Scientific Double Precision	144
SUE 1112B, Scientific Double Precision	146

SUE 1110 basic is the first instruction set described in this bulletin. SUE 1110A performs the basic instruction set and one additional instruction, Store Key (SKEY). Both SUE 1111A and 1112A processors have the speed and capabilities of SUE 1110A, and each has an extended instruction set. Descriptions of these extended instructions follow the description of the SUE 1110B. Instruction times for all instructions are summarized in Appendix A.

Processors SUE 1110B, 1111B, and 1112B perform the same instructions as the respective A-series processors, and two additional instructions Fetch and Clear Word (FCLW), and Fetch and Clear Byte (FCLB). These two instructions can be used in multiprocessor systems as a synchronizing mechanism.

Instructions in this bulletin are described in machine language for the system user possessing a background in digital computer terminology and operation. Additional information on the basic instruction set is contained in the LAP-2 Assembler manual. Operation and maintenance of SUE processors is contained in the respective reference and maintenance bulletins designated by the processor model number.

SUE 1110 INSTRUCTION SET

INTRODUCTION

SUE 1110 instruction set includes 108 basic instructions exclusive of 16 address-ing modes. Many of these instructions operate on either 16-bit data or 8-bit byte formats. Other instructions test one or more of the 16 status indicator bits. This bulletin presents a detailed description of word formats, addressing modes, and status indicators followed by a definition of each instruction operation.

The 108 instructions are divided into eleven classes according to type of instruction function. Seven of these classes are grouped as general register instructions. They contain arithmetic, logical, move, compare and test func-tions that involve the eight general registers of the processor. Two classes represent the branch instructions. They contain unconditional and conditional branch functions on the true or false condition of status indicators. The shift class contains full 15-bit shift capabilities with eight different operations and two address modes. The control class contains system control functions such as load/store of all general registers, load/store of status indicators and control of interrrupt operations.

The eleven instruction classes are:

Class Code	Description
1	Accumulator to Memory with Auto Decrement
2	Accumulator to Memory with Auto Increment
3	Accumulator to Memory
4	Data to Accumulator, Jump to Subroutine, Jump, and Register to Register
5	Memory to Accumulator with Auto Decrement
6	Memory to Accumulator with Auto Increment
7	Memory to Accumulator

8	Branch False and No Operation
9	Branch True and Unconditional
A	Shift
0	Control

Class codes are specified in the instruction word format by the four-bit C field. (Fields are defined later under instruction words in this bulletin). Five class codes are not defined for the basic instruction set. They have been reserved for specification of additional general purpose instructions in the SUE 1111A, B and 1112A, B Processors; or, for special purpose instructions in future SUE processors with expanded ROM control memories.

SUE 1110 Processor contains eight, 16-bit general registers including the program counter. Seven of these registers may be used as accumulators or index registers. The arithmetic-logic unit processes 16-bit operands but memory data may be 8-bit bytes or 16-bit words.

Memory addresses are 16-bit numbers that select up to 60k (k = 1024) bytes. Addresses 60k to 64k are used to directly address registers within system modules other than program memory modules.

WORD FORMATS

Bit positions within a word are numbered right to left starting with 0. Bit 0 is the least significant bit of the word and bit 15 is the most significant.

DATA WORDS

Two data word formats can be processed, an 8-bit byte and a 16-bit word. The most significant bit (15) represents the algebraic sign of numeric data. A ONE in bit position 15 represents a negative number, and a ZERO represents a positive number. Negative numbers are in twos complement form.

Byte Format

```
15                                                    0
+----------------------------+----------------------------+
|        Left Byte           |        Right Byte          |
|      (even address)        |       (odd address)        |
+----------------------------+----------------------------+
```

Word Format

```
15  14                                                0
+---+------------------------------------------------+
| S |                   Number                       |
+---+------------------------------------------------+
```
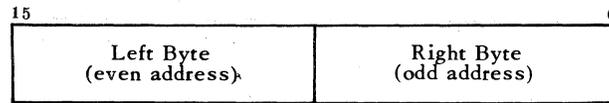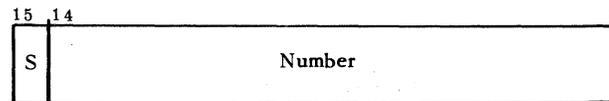
S	Sign
1	negative (-)
0	positive (+)

In byte operations, the entire selected 16-bit register is used in the operation with the byte operand. In register-to-memory instructions (byte mode), the right byte of the register operates on the designated byte in memory. In memory-to-register instructions (byte mode), the designated byte in memory operates on the full 16-bit register as though the memory operand has a left byte equal to ZERO attached to it. In either type of operation, arithmetic operations occur in a 16-bit register and carry and overflow are detected out of a 16-bit register.

ADDRESS WORDS

The 16-bit address represents a byte address. Bit zero selects the left or right byte of a 16-bit word. On word addresses, bit zero is used to specify more than one level of indirect addressing.

Byte Address

```
15                                          1   0
┌──────────────────────────────────────┬───┐
│                 Value                  │ B │
└──────────────────────────────────────┴───┘
```
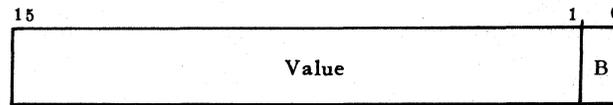
Bit 0 - <u>Byte</u>
   0   left
   1   right

Word Address

```
15                                          1   0
┌──────────────────────────────────────────┬───┐
│                                            │ I │
└──────────────────────────────────────────┴───┘
```
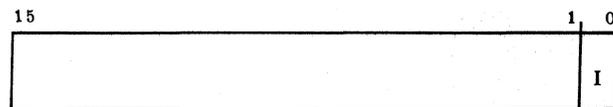
Bit 0 - <u>Addressing</u>
   0   address direct
   1   indirect

INSTRUCTION WORDS

Instruction words are constructed to facilitate encoding and decoding of the
machine language code. The words are defined so that the fields of the instruc-
tion do not overlap the four hexadecimal digits represented by H1, H2, H3, and
H4. Those fields that are subsets of a hexadecimal digit are right-justified,
with the high-order bit used to indicate the less common condition.

```
15        12│11      8│7      4│3       0
┌──────────┬──────────┬────────┬──────────┐
│    H1     │    H2     │   H3    │    H4     │
└──────────┴──────────┴────────┴──────────┘
```
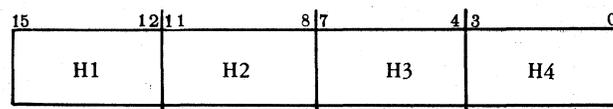
<u>Digit</u>

H1   Class Designation 0 through 15

H2   Operation Designator (usually)

H3   Accumulator Designator (usually)

H4   Index Designator (usually)

FIELDS. – A variety of word formats are interpreted by the processor. All of the fields used, and their positions, are defined below in a composite drawing. Functions of a given field may vary according to the instruction.

15 14 13 12	11 10 9 8	7	6 5 4	3 2 1 0
C	T	D		
	B	OP	I	F
				AR	K
					E	XR
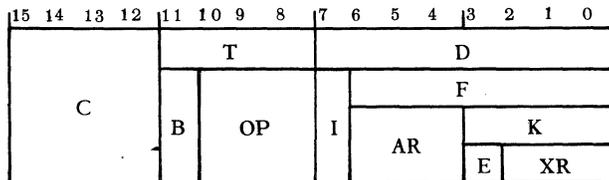
NOTE: Several fields have more than one function depending on the instruction that contains them. In the field definitions below, any function common to several instructions is defined. For descriptions of other functions, refer to corresponding instruction descriptions.

Field Definitions

Symbol	Description
C	Class Indicator (4 bits) – Specifies 1 of 16 classes or divisions of the instruction set. Classes indicate the type of function.
T	Test Operation (4 bits) – Defines operation codes for the Control and Branch classes.
D	Displacement Address (8 bits) – Direct address (+ or –) to words relative to the address of the instruction. May be expressed as P (Program Counter) +D, where D is the range, –128 through +127. An exception, if the absolute-address mode of a Control instruction is specified, then D directly addresses the first 256 words in memory.
B	Byte Indicator (1 bit) – Specifies whether the memory operand is a word (B=0) or a byte (B=1). Field of a Control instruction specifies the Relative (B=1) or Absolute (B=0) address mode.
OP	Operation Indicator (3 bits) – Defines 1 of 8 operations available to certain classes. Several classes use the same set of operations, as explained in greater detail under Instruction Descriptions.
I	Indirect Addressing Indicator (1 bit) – Specifies first level of indirect addressing if I=1.
AR	Accumulator Register Designator (3 bits) – Designates 1 of 8 general registers as an A-Register during instruction execution.
E	Extended-Address Indicator (1 bit) – Indicates (when 1) that the word following the instruction will be accessed as an extended-address part of the instruction.
XR	Index Register Designator (3 bits) – Designates 1 of 7 general registers as an X-Register during instruction execution.
K	Constant (4 bits) – Designates length of a Shift command, or an immediate constant. **Also used to enable interrupts.**
F	Status Bit-Pattern (7 bits) – Comprises the bit pattern for changing control states for certain Control instructions.

ADDRESSING

SUE 1110 Processor develops a 16-bit operand address based on the mode that is selected by the instruction class code and other fields of the instruction word format.

BYTE-WORD ADDRESSING

A bit (B) in the instruction word specifies if the operand is to be a byte (8 bits) or a word (16 bits) in general register instructions. If B = 1 and bit zero of the effective operand address is ZERO, the left byte (bits 15 through 8) is used; the right byte (bits 7 through 0) is used if bit zero is ONE. If B = 0, a word operand is requested and the address of the word is treated as an even-numbered byte address.

ABSOLUTE AND RELATIVE ADDRESSING

Branch instructions use the relative displacement method to develop the branch address. The D field of the instruction is an 8-bit (7 bits plus sign) number that specifies a branch within +127 or -128 words (not bytes) from the current location. Negative numbers are represented in twos complement form.

Control instructions use the relative displacement as well as the absolute addressing modes. Bit B of the instruction word, when set to a ONE, selects the relative mode and, when ZERO, selects the absolute mode. In the absolute address mode the D field of the instruction is an 8 bit number that specifies direct address of the first 256 words (not bytes) of memory.

EXTENDED ADDRESSING

When the E bit of the instruction is a ONE, the word following the instruction becomes the base address and is used to develop the operand address. When E = 0 the base address is ZERO.

INDEXING

Content of one of the seven general registers may be selected as an index register. The XR field of the instruction selects the register. When the XR field is all zeros, no indexing is specified. If neither extended addressing nor indexing is called for (i.e. bits 3-0 all ZEROs) then no address is specified and an unimplemented instruction trap is generated.

Two types of indexing are used:

Base Relative Indexing (indexing relative to the base address of the computer or user program). — In this type of indexing, the index register contains the complete address of the desired memory location. Base relative indexing together with autoincrement or autodecrement provide generalized push down and pop up stack processing capabilities.

Table Indexing (indexing relative to the base address of a table). — In this type of indexing the index register contains the variable n to fetch the quantity located at TABLE + n.

AUTO INCREMENTING AND DECREMENTING

Within the general register instructions, separate class codes are used to provide the option for automatic increment or decrement of the index register selected by the XR field of the instruction. When autodecrement is specified, the content of the index register is decremented before the operand address is generated. When autoincrement is specified, the content of the selected index register is incremented after the operand address is generated.

ONE is subtracted or added to the content of the index register when the instruction specifies a byte operand with autodecrement or autoincrement. TWO is subtracted or added when the instruction specifies a word operand with autodecrement or autoincrement.

INDIRECT ADDRESSING

If indirect bit I of the instruction is set to a ONE, the address developed by the processor points to the address of the operand.

Multi-level indirect addressing is provided in the word mode only. The processor tests the least significant bit of the indirect address. If this bit is a ONE, and the word mode is specified, the word pointed-to is also treated as an indirect address. If the least significant bit of the address is a ZERO, the processor stops the multi-level indirect addressing for this instruction. If the processor counts up to 16 levels of indirect addressing, an unimplemented instruction self-interrupt is generated and the instruction is trapped.

Only single level indirect addressing is available in the byte mode because the least significant bit of the operand address specifies left or right byte.

REGISTER, IMMEDIATE AND LITERAL OPERANDS

The data-to-accumulator (class code 4) general instruction provides for selection of register, literal or immediate operands. The register operand is the register specified by the XR field, and can be the program counter if XR = 0. The literal operand may be the 16-bit word following the instruction or the 16-bit word following the instruction plus the contents of XR. An immediate operand is the 4-bit value in the instruction's K field.

COMBINATION ADDRESSING MODES

In most general register instructions, combinations of addressing modes may be specified to yield fourteen useful functions for memory operand selection. The processor develops addresses in combinations of the following in the sequence shown:

> Extended Address
> Autodecrement the Index
> Indexed
> Indirect
> Autoincrement the Index

Autodecrement and autoincrement functions apply to the contents of the general register selected by the XR field of the instruction.

On autodecrement the content of the index register is decremented by one for byte addresses or by two for word addresses before the index register contents is used as an index value. On autoincrement the content of the index register is incremented by one or two after it is used as an index value.

If the XR field of an instructions is all ZEROs, no indexing is specified. However, auto-increment or auto-decrement specified with a ZERO XR field affects the program counter.

Table 1 contains a summary of the fourteen combinational addressing modes.

Table 1. Combination Addressing Modes

Address Mode	M Effective Address	XR Index Register	Assembler Mnemonic
Extended	A	-	A
Extended, Indexed	A + X	-	A(R)
Extended, Indexed, Autoincrement	A + X	X + e	A(R+)
Extended, Autodecrement, Indexed	A + X - e	X - e	A(-R)
Indexed	X	-	(R)
Indexed, Autoincrement	X	X + e	(R+)
Autodecrement, Indexed	X - e	X - e	(-R)
Extended, Indirect	[A]	-	*A
Extended, Indexed, Indirect	[A + X]	-	*A(R)
Extended, Indexed, Autoincrement, Indirect	[A + X]	X + e	*A(R+)
Extended, Autodecrement, Indexed, Indirect	[A + X - e]	X - e	*A(-R)
Indexed, Indirect	[X]	-	*(R)
Indexed, Autoincrement, Indirect	[X]	X + e	*(R+)
Autodecrement, Indexed, Indirect	[X - e]	X - e	*(-R)

NOTES: A - 16-bit word following instruction

X - Content of General register selected by XR field

e - A ONE if byte address, a TWO if word address

[ ] - 16-bit word at address specified in brackets.

SPECIAL ADDRESSES

Even addresses 61,440 to 65,534 (hexadecimal F000 to FFFE) are reserved for addressing of system hardware registers within SUE system modules. The odd numbered addresses in this range are not used. Each system module is assigned a set of even (word) addresses as shown in table 2.

Addressing a system register for either a read or write function is allowed by master modules. The slave module always transmits or receives 16 data bits. If the selected register is less than 16 bits in length, the data is transmitted in the least significant bit positions and the most significant, unused, bit positions are ZEROs.

Table 2. Special Addresses

Addresses (Hexadecimal)	Module Assignment
F000-F7FE	Reserved for special memory assignments
F800	I/O Device Controller #1, Status Register
F802	I/O Device Controller #1, BTA Address Register
F804	I/O Device Controller #1, BTA Block Length Register
F806	I/O Device Controller #1, Control Register
F808	I/O Device Controller #1, Data Register
F80A-F80E	Reserved for I/O Device Controller #1
F810-F81E	I/O Device Controller #2 as in #1
F820-FAFE	Reserved for I/O Device Controllers as in #1. (see Appendix C)
FB00-FBFE	Auto Load Memory
FC00-FEFE	Reserved for Auto Load
FF00	Central Processor (#0) Register 0, (Program Counter)
FF02-FF0E	Central Processor (#0), General Registers 1-7
FF10	Central Processor (#0), Status Indicators
FF12	Central Processor (#0), Instruction Register
FF14-FF1C	Reserved for Central Processor #0
FF1E	Central Processor (#0), Control Flip-Flops
FF20-FF3E	Processor #1, same set as #0
FF40-FF5E	Processor #2, same set as #0
FF60-FF7E	Processor #3, same set as #0
FF80	Control Panel #1 Address Register-Attention Interrupt
FF82	Control Panel #1 Data Register
FF84-FF86	Control Panel #2 as in #1
FF88-FF8A	Control Panel #3 as in #1
FF8C-FF8E	Control Panel #4 as in #1
FF90-FFFF	Reserved for other System Modules to be assigned.

STATUS INDICATORS

SUE 1110 Processor has a 16-bit status indicator register. Status indicators may be affected by execution of general register and shift instructions. This is indicated by their symbol in INSTRUCTION DESCRIPTIONS. The status indicators may also be set or reset with special control instructions.

The status bit position within the status register, symbol, name, and description are as follows:

Bit	Symbol	Name and Description
0	EQ	Equal — In a compare operation, the source operand equals the target operand.
1	GT	Greater-Than — In a compare operation, the source operand is greater than the target operand.
2	OV	Overflow — Set during Add, Subtract, or Arithmetic Left Shift if the Carry out of bit 15 is different than the Carry in to bit 15. If the set condition is not caused, V remains unchanged.
3	CY	Carry — Receives the Carry out of bit 15 during an Add, Subtract, Arithmetic Left Shift, or Left Linked Shift. Reset during an Arithmetic Right Shift. Receives bit 0 shifted out from a Right Linked Shift.
4	F1	
5	F2	Flags 1, 2, or 3 — Programmable flag bits.
6	F3	
7	LP	Loop Complete — Set if content of register selected by XR field equals ZERO at the completion of an Autoincrement or Autodecrement instruction. Reset if content of XR is NOT ZERO.
8	OD	Odd — For all general register instructions except Compare, the Odd indicator receives the least significant bit of the result.
9	ZE	Zero — For all general register instructions except Compare, set if the result is ZERO and reset if NOT ZERO.
10	NG	Negative — Receives the most significant bit of the result of any general register instruction except Compare.
11	A	Active — Indicates that the processor is executing instructions. A is set unless the processor is quiescent.
12	M1	Interrupt Mask — Bits M1 through M4 correspond to system interrupts 1 through 4. When any bit is set or reset, respectively, the Bus Controller is requested to ignore or allow interrupt requests for the corresponding vector.
13	M2	
14	M3	
15	M4	

INSTRUCTION DESCRIPTIONS

GENERAL REGISTER INSTRUCTIONS

Class codes 1 through 7 specify the general register instructions. They are
all two-operand instructions with one set of eight general operations. In the
definitions of these operations, the terms target (T) and source (S) are used.
The target is the register or memory cell to be modified, the source is the
register or memory cell used as an operand that is to remain unchanged.

GENERAL OPERATIONS. — The OP field of the instruction selects the operation
for each class of general register instruction as follows:

OP Code (Hexadecimal	Operation	Description	Status Indicators Affected
0	MOVe	Transfer the source operand to the target operand. $(S) \rightarrow (T)$	NG, ZE, OD
1	SUBtract	Subtract the source operand from the target operand and store the result in the target operand. $-(S) + (T) \rightarrow (T)$	CY, OV, NG, ZE, OD
2	ADD	Form the sum of the source (S) and target (T) operands and store in (T). $(S) + (T) \rightarrow (T)$	CY, OV, NG, ZE, OD
3	AND	Form the logical product of the source and target operands and store the result in the target operand. $(S).AND. (T) \rightarrow (T)$	NG, ZE, OD
4	Inclusive OR	Form the logical sum of the source and target operands and store in the target operand. $(S).OR. (T) \rightarrow (T)$	NG, ZE, OD

| 5 | Exclusive OR | Form the logical difference of the source and target operands and store in the target operand.
(S).EOR. (T) → (T) | NG, ZE, OD |

| 6 | CoMPare | Compare logical, the source operand to the target operand. Register contents and memory contents are not affected. | GT, EQ |

	GT	EQ
If (S) < (T)	0	0
If (S) = (T)	0	1
If (S) > (T)	1	0

NOTE

Bit 15 of each word is considered a magnitude bit, not a sign bit. The compare result is unsigned based on the 16-bit magnitude.

| 7 | TeST | Form the logical product of the source and target operands. Register and memory contents are not affected. | NG, ZE, OD |

If (S) .AND. (T) = 0, SET ZE, RESET NG, OD

If (S) .AND. (T) ≠ 0, RESET ZE

If (S) .AND. (T) is odd, SET OD (odd implies bit 0 is set)

If (S) .AND. (T) is negative, SET NG (negative implies bit 15 is set)

GENERAL REGISTER INSTRUCTION WORD FORMATS. — The instruction word formats used for the general register instructions is shown in table 3.

Table 3. General Register Instruction Word Formats

General Register Classes*		H_1 15 14 13 12	H_2 11 10 9 8		H_3 7 6 5 4		H_4 3 2 1 0	
Accumulator	Auto Decrement	C=1	B	OP	I	AR	E	XR
To	Auto Increment	C=2	B	OP	I	AR	E	XR
Memory		C=3	B	OP	I	AR	E	XR
Jump to Subroutine		C=4	0	0	I	AR	E	XR
Jump		C=4	0	0	I	0	E	XR
Data to	Literal/Register	C=4	1	OP	0	AR	E	XR
Accumulator	Immediate Data	C=4	1	OP	1	AR	K	
Memory	Auto Decrement	C=5	B	OP	I	AR	E	XR
To	Auto Increment	C=6	B	OP	I	AR	E	XR
Accumulator		C=7	B	OP	I	AR	E	XR

NOTES:

C   Class Codes 1-7

OP   Operation Code: 0   MOV   Move
                              1   SUB   Subtraction
                              2   ADD   Addition
                              3   AND   Logical Product
                              4   IOR   Logical Inclusive OR
                              5   EOR   Logical Exclusive OR
                              6   CMP   Compare
                              7   TST   Test

B   Word when 0, Byte when 1

I   Indirect when 1

AR   Accumulator Register designator (0-7)

E   Extended or two-word instruction when 1

XR   Index Register designator (0-7), no indexing when 0

K   4-bit Immediate data constant

For E=0 and XR $\neq$ 0, XR provides the entire operand address. If no index
register is selected (XR=0), and E=0, an instruction trap occurs, except
for class 4, Register, where the PC is the source operand.

For E=1, XR=0, the next word provides the entire operand address. If E=1
and XR $\neq$ 0, indexing operation is specified. In this case, the content of (XR)
is added to the next word to produce the effective address of the memory
operand or an indirect address.

*SUE 1112 Instruction Set contains more instructions in class code 4.

The following additional symbols are used in the instruction definitions:

( )   Contents of

M   effective operand address

PC   Program Counter, general register 0.

P   Current instruction address

General Register Instruction Definitions

ACCUMULATOR TO MEMORY, AUTO DECREMENT

C	B	OP	I	AR	E	XR
1						

15   11 10   7   6   3   2   0

```
+-------------------------------+
|                               |
|       Extended Address        |
|                               |
+-------------------------------+
```
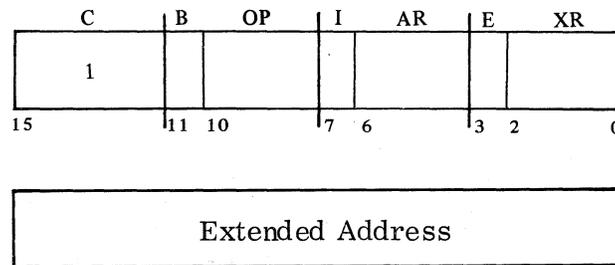
(AR) operates on (M). (XR) is decremented before use.

ACCUMULATOR TO MEMORY, AUTO INCREMENT

C	B	OP	I	AR	E	XR
2						

15   11 10   7   6   3   2   0

```
+-------------------------------+
|                               |
|       Extended Address        |
|                               |
+-------------------------------+
```
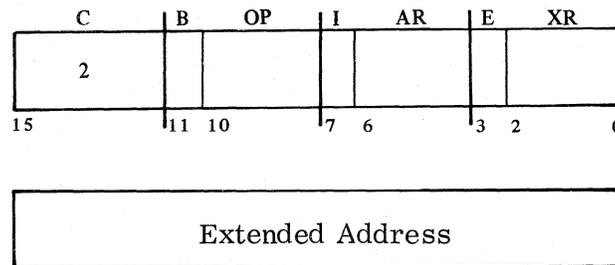
(AR) operates on (M). (XR) is incremented after use.

ACCUMULATOR TO MEMORY

C	B	OP	I	AR	E	XR
3						

15   11 10   7   6   3   2   0

Extended Address
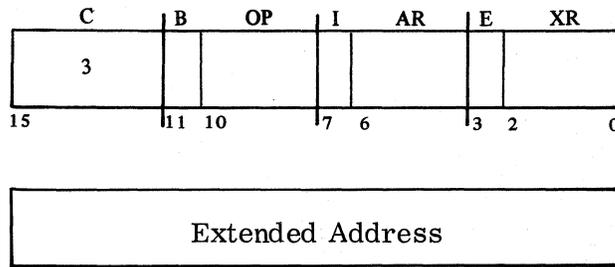
(AR) operates on (M).   (XR) is not affected.

JUMP TO SUBROUTINE   JSBR

C	B	OP	I	AR	E	XR
4	0	0		1-7		

15   11 10   7   6   3   2   0

Extended Address
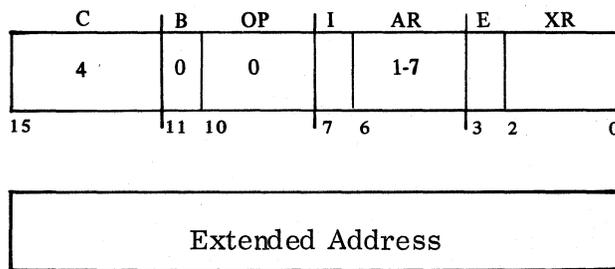
P + 2 replaces (AR) for E = 0 and P +4 replaces (AR) for E = 1.
(M) operates on PC (content of general register 0). Thus, the return
address is stored in AR and PC is set to the jump-location address.

JUMP   JUMP

C		OP	I	AR	E	XR
4	0	0		0		

15   11 10   7   6   3   2   0

Extended Address
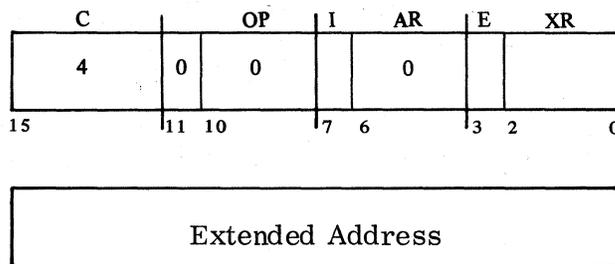
(M) operates on PC, setting it to the jump-location address. The
jump function is the same as a MOV (M) to PC, but does not affect
status indicators.

DATA TO ACCUMULATOR, INDEX REGISTER

C	B	OP	I	AR	E	XR
4	1		0		0	

15   11 10   7 6   3 2   0
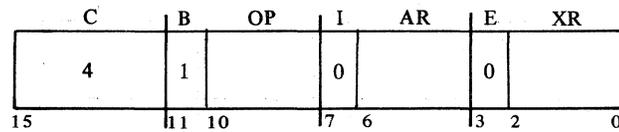
A register-to-register instruction.   (XR) operates on (AR).

DATA TO ACCUMULATOR, LITERAL

C	B	OP	I	AR	E	XR
4	1		0		1	

15   11 10   7 6   3 2   0
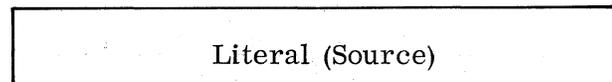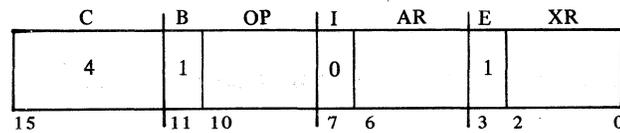
Literal (Source)

The word following the instruction is the literal source operand. It operates on (AR). If XR is not 0, then (XR) is added to the literal before operating on (AR).

DATA TO ACCUMULATOR, IMMEDIATE

C	B	OP	I	AR	K
4	1		1		

15   11 10   7 6   3   0
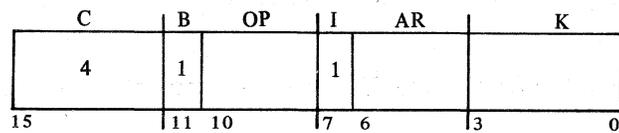
K operates on (AR).   K is the 4-bit immediate constant operand.

MEMORY TO ACCUMULATOR, AUTO DECREMENT

C	B	OP	I	AR	E	XR
5						

15   11 10   7 6   3 2   0

Extended Address
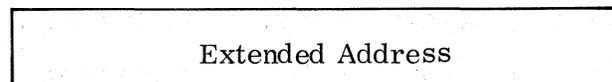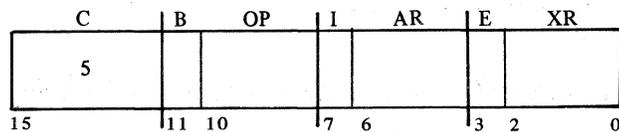
(M) operates on (AR).   (XR) is decremented before use.

MEMORY TO ACCUMULATOR, AUTO INCREMENT

	C	B	OP	I	AR	E	XR
	6						

15   11   10   7   6   3   2   0

Extended Address
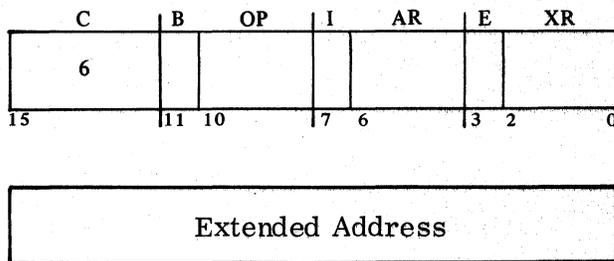
(M) operates on (AR). (XR) is incremented after use.

MEMORY TO ACCUMULATOR

	C	B	OP	I	AR	E	XR
	7						

15   11   10   7   6   3   2   0

Extended Address
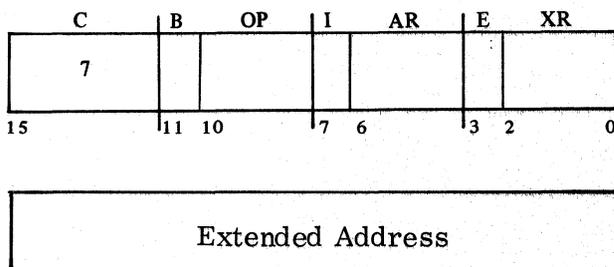
(M) operates on (AR). (XR) is not affected.

GENERAL REGISTER INSTRUCTION TIMES. - Instruction execution times
depend on:

> Operand addressing modes
> General operation code
> Program memory access and cycles
> INFIBUS availability

Table 4 contains a summary of typical general register instruction times assum-
ing the INFIBUS is available to the processor and a SUE 3311 Core Memory is
used for instruction and data storage. A memory cycle time of 850 nanoseconds,
read access time of 750 nanoseconds, and a write access time of 550 nanosec-
onds is used. Access is the total time to access both the bus scheduler and
memory. Microprogram steps of 160 nanoseconds are used for arithmetic
operations and 130 nanoseconds for non-arithmetic operations.

Table 4. SUE 1110 (Basic) General Register Instruction Times

General Instruction		Time (Microseconds)		
		Indexed	Auto-Increment	Auto-Decrement
ACCUMULATOR TO MEMORY	Class Codes	3	2	1
Logical: MOV, AND, IOR, EOR Op Codes: 0 3 4 5		3.94	4.81	4.81
Arithmetic: SUB, ADD Op Codes: 1 2		4.03	4.90	4.90
Compare: CMP Op Code: 6		3.70	4.57	4.57
Test: TST Op Code: 7		3.35	4.22	4.22

Address Modes:
  For Extended, add 0.13
  For Indirect, add 1.14 for first level, add 1.01
    for each additional level
  For Extended, Indirect, add 1.40 for first level,
    add 1.01 for each additional level				

JUMP, JUMP TO SUBROUTINE	Class Code	4	–	–
Instruction: JUMP, JSBR Op Code: 0, AR = 0, AR ≠ 0		2.79	–	–
Address Modes: For Extended, add 0.06		2.85	–	–
For Indirect, add 1.14 for first level, add 1.01 for each additional level		3.93	–	–
For Extended, Indirect add 1.33 for first level, add 1.01 for each additional level		4.12	–	–
DATA TO ACCUMULATOR	Class Code	4	–	–
Logical: MOV, AND, IOR, EOR Op Codes: 0 3 4 5 Register to		2.50	–	–
Arithmetic: SUB, ADD Register or Op Codes: 1 2 Immediate		2.79	–	–
Compare: CMP Op Code: 6		2.69	–	–
Test: TST Op Code: 7		2.50	–	–

Address Modes:
  For Literal add 0.68
  For Literal Indexed add 0.84				

MEMORY TO ACCUMULATOR	Class Codes	7	6	5
Logical: MOV, AND, IOR, EOR Op Codes: 0 3 4 5		3.35	4.09	4.09
Arithmetic: SUB, ADD Op Codes: 1 2		3.64	4.38	4.38
Compare: CMP Op Code: 6		3.67	4.41	4.41
Test: TST Op Code: 7		3.35	4.09	4.09

Address Modes:
  For Extended add 0.13
  For Indirect add 1.14 for first level, add 1.01 for
    each additional level
  For Extended, Indirect add 1.40 for first level,
    1.01 for each additional level				

NOTE: All times are in microseconds.				

To compute the actual instruction execution time, it is necessary to add the time increments shown in Table 4 for each selected addressing mode. The minimum times shown in the table assume an indexed addressing mode. A more complete table of general instruction times is given in Appendix A.

For example, an ADD register-to-register instruction requires 2.79 microseconds with the SUE core memory. An ADD memory-to-accumulator instruction requires 3.64 microseconds when the operand address is held in an index register. If the address is located in the next word location (extended instruction mode), the time is 3.77 microseconds. Indexing the extended address does not add time to the instruction. Indirect addressing adds 1.14 microseconds for the first level and 1.01 for each subsequent level.

BRANCH CONDITIONAL INSTRUCTIONS

Thirteen conditions can be tested by branch conditional (TRUE or FALSE) instructions. Each condition can be tested to produce a branch or a fall-through to the next instruction for either state (TRUE for class code 9 and FALSE for class code 8). The condition status is determined by testing the status indicators and programmable flags affected by the last operation.

BRANCH CONDITIONS. — Following is a list of the 13 branch conditions and their meaning when TRUE.

| T Field	Condition	Symbol	Meaning (TRUE Condition)
0	Unconditional	UN	The branch is made unconditionally.
1	Equal	EQ	The latest compare operation found the two operands to be equal to each other.
2	Greater-Than	GT	The latest compare operation found the source operand to be greater than the target operand.
3	Overflow	OV	An add, subtract, or shift operation produced a result outside of the range $-2^{15} \le R \le + (2^{15} - 1)$ since overflow was last reset.
4	Carry	CY	The latest add, subtract, or shift operation produced a carry out of the most significant end of the arithmetic unit.
5	Flag 1	F1	These three programmable flags can be
6	Flag 2	F2	set or reset by a set or reset status
7	Flag 3	F3	indicator instruction.
8	Loop Complete	LP	This indicator is set if the result of the latest autoincrement or autodecrement of any index register equals zero; otherwise it is reset.
9	Odd	OD	The result of the latest general operation (except compare), or shift operation is an odd number (Bit 0 = 1).
A	Zero	ZE	The latest general operation (except compare), or shift operation results in all zeros.
B	Negative	NG	Result of the latest general operation (except compare), or shift operation is a negative number (Bit 15 = 1).
C	Less-Than	LT	In the latest compare operation, the source operand was less than the target operand.
D, E, F			Cause an unimplemented instruction trap.

BRANCH INSTRUCTION WORD FORMATS. –

	H_1				H_2			H_3				H_4			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

No Operation	C= 8	0	////////
Branch Unconditional	C= 9	0	D
Branch False	C= 8	T	D
Branch True	C= 9	T	D

D   Displacement word address in twos complement form.

T   The T-Field specifies each Branch test. That is, which processor
     status indicator (if any) is to be tested. A list of each indicator, the
     corresponding value for T, and the operator assembler-mnemonics
     follows (z = T for true and F for false):

T (hexadecimal)	Indicator	Assembler Mnemonic
		xx ↓↓
1	Equal	BEQz
2	Greater Than	BGTz
3	Overflow	BOVz
4	Carry	BCYz
5	Flag 1	BF1z
6	Flag 2	BF2z
7	Flag 3	BF3z
8	Loop Complete	BLPz
9	Odd	BODz
A	Zero	BZEz
B	Negative	BNGz
C	Less Than	BLTz
D, E, F,	(Instruction is trapped)	

Branch Instruction Definitions

NO OPERATION   NOPR

C	T	D
8	0	////////
15	11 7	0
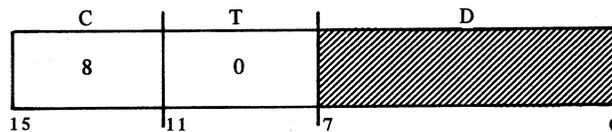
A one-word NO-OP which does not affect status indicators.

BRANCH UNCONDITIONAL   BRUN

| C	T	D
9	0	
15	11	7 0
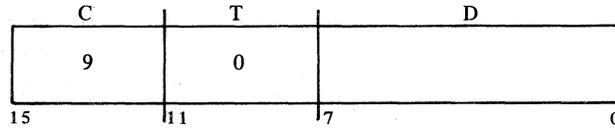
An unconditional (no testing) branch is made to the relative address specified by D.   PC + 2 x D replaces PC.   D is in twos complement form with sign extended to represent a 16 bit number.

BRANCH FALSE   BxxF

C	T	D
8		
15	11	7 0
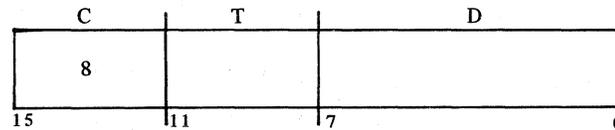
A Branch is made to the relative address specified by D if the indicator specified by T is false, or 0; otherwise, the next instruction in sequence is accessed.

BRANCH TRUE   BxxT

C	T	D
9		
15	11	7 0
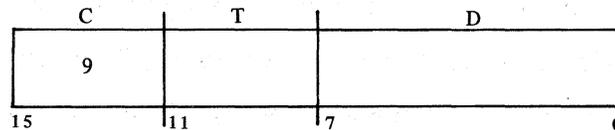
A Branch is made to the relative address specified by D if the indicator specified by T is true, or 1; otherwise, the next instruction in sequence is accessed.

BRANCH INSTRUCTION TIMES. – Branch instruction execution times depend on whether or not the branch occurs, or the next instruction in sequence is executed. The branch-on-less-than operation ($T = C$) has different timing than the branch on other status bits. The branch instruction times are shown in Table 5.

Table 5. Branch Instruction Times

Instruction	Assembler Mnemonic	Time (microseconds)			
		Next Word		Branch	
		1110 Basic	All Others**	1110 Basic	All Others**
No Operation	NOPR	1.78	1.75	–	–
Branch Unconditional	BRUN	–	–	2.72	2.82
Branch True	BxxT*	1.78	1.75	2.72	2.82
Branch False	BxxF*	1.78	1.75	2.72	2.82
Branch Less Than True	BLTT	1.75	1.75	3.08	3.21
Branch Less Than False	BLTF	1.88	1.88	3.08	3.21

*where xx = EQ, GT, OV, CY, F1, F2, F3, LP, OD, ZE, NG.
**includes Processors 1110A and B, 1111A and B, and 1112A and B.

SHIFT INSTRUCTIONS

Class code A (hexadecimal) specifies a shift instruction. Up to 15 bit-position shifts may be specified in a single shift instruction. Two formats are provided to allow an option on the location of the shift count. When bit 7 of the instruction is a ZERO, the least significant four bits of the general register selected by XR, contains the shift count. When bit 7 is a ONE, the K field of the instruction word specifies the shift count.

SHIFT INSTRUCTION WORD FORMATS. -

Two single-word formats are used. The formats illustrated are for the shift

count defined by (XR) or K, respectively.

	H$_1$				H$_2$				H$_3$				H$_4$		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Shift Indexed	A$_{16}$				0	OP			0	AR			0	XR		
Shift Immediate	A$_{16}$				0	OP			1	AR			K			

AR   Accumulator Register designator (to be shifted).
K   Shift Count
XR   Shift Count Source Register.
OP   Shift Operation Code:

| OP	Bits			Operation
	10	9	8	
0	0	0	0	Single Left Arithmetic Open
1	0	0	1	Single Left Logical Linked
2	0	1	0	Single Left Logical Open
3	0	1	1	Single Left Logical Closed
4	1	0	0	Single Right Arithmetic Open
5	1	0	1	Single Right Logical Linked
6	1	1	0	Single Right Logical Open
7	1	1	1	Single Right Logical Closed

NOTE

SUE 1112 Processor provides double length
shifts and also normalize instructions in
addition to these basic single shifts.

Shift Instruction Definitions

SINGLE LEFT ARITHMETIC OPEN SLAO

OP = 0
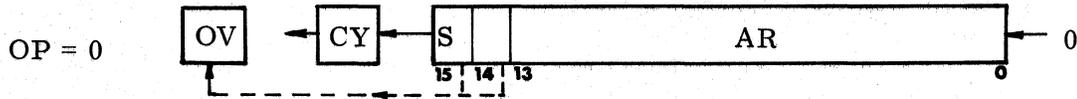
(AR) bits are shifted left out of $(AR)_{15}$ to the carry (CY) and zeros are shifted to $(AR)_0$. If any $(AR)_{14}$ bit is different than $(AR)_{15}$ preceding a shift, then the overflow indicator, OV, is set. Operation affects status indicators: CY, OV, NG, ZE, OD.

SINGLE LEFT LOGICAL LINKED SLLL

OP = 1
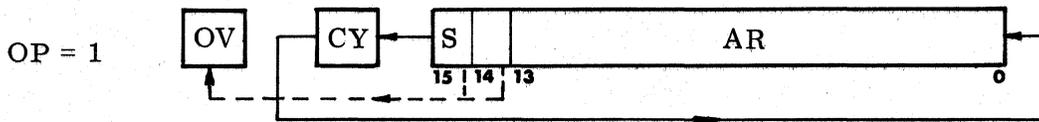
Carry (CY) is shifted into $(AR)_0$ and $(AR)_{15}$ is shifted into CY. If any $(AR)_{14}$ bit is different than $(AR)_{15}$ preceding a shift, then the overflow indicator, OV, is set. Operation affects status indicators: CY, OV, NG, ZE, OD.

SINGLE LEFT LOGICAL OPEN SLLO

OP = 2
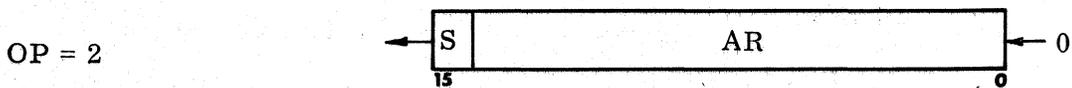
(AR) is shifted left. For each bit shifted, $(AR)_{15}$ is lost and $(AR)_0$ equals 0. Operation affects status indicators: NG, ZE, OD.

SINGLE LEFT LOGICAL CLOSED SLLC

OP = 3
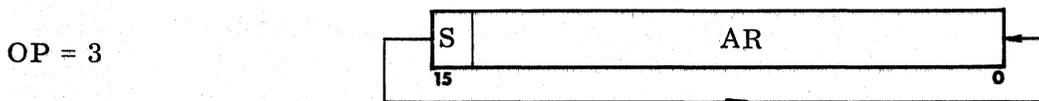
(AR) is shifted left. $(AR)_{15}$ is shifted into $(AR)_0$. Operation affects status indicators: NG, ZE, OD.

SINGLE RIGHT ARITHMETIC OPEN   SRAO

OP = 4
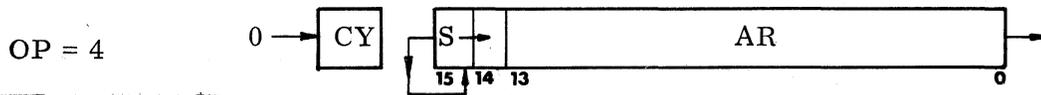
(AR) is shifted right. $(AR)_{15}$, the sign bit, remains the same and is shifted into $(AR)_{14}$. $(AR)_0$ bits shifted out are lost. Carry (CY) is reset. Operation affects status indicators: CY, NG, ZE, OD.

SINGLE RIGHT LOGICAL LINKED   SRLL

OP = 5
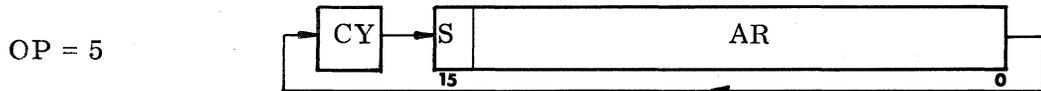
Carry (CY) is shifted into $(AR)_{15}$, and $(AR)_0$ is shifted into CY. Operation affects status indicators: CY, NG, ZE, OD.

SINGLE RIGHT LOGICAL OPEN   SRLO

OP = 6
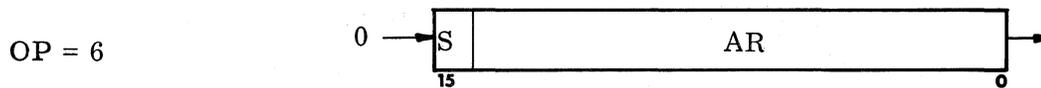
(AR) is shifted right. For each bit shifted, $(AR)_0$ is lost and $(AR)_{15}$ equals 0. Operation affects status indicators: NG, ZE, OD.

SINGLE RIGHT LOGICAL CLOSED   SRLC

OP = 7
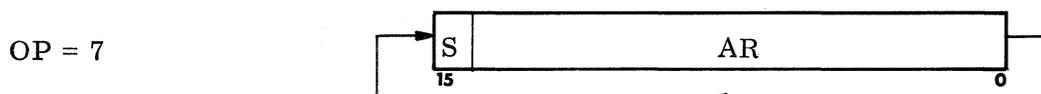
(AR) is shifted right. $(AR)_0$ is shifted into $(AR)_{15}$. Operation affects status indicators: NG, ZE, OD.

SHIFT INSTRUCTION TIMING. - Shift instruction execution times depend on the number of single bit shifts (N) specified in either the K field (immediate) or the selected register, XR. The time is calculated by the formula:

$$T_s = 2.76 + (0.26)N$$

where N = 0, 1, ..., 15.

CONTROL INSTRUCTIONS

Class code 0^1 specifies a group of instructions that provide control of processor operation in a system. The instructions provide control of system interrupts, and storing and restoring status indicators and general registers.

CONTROL INSTRUCTION WORD FORMATS. -

Formats				Assembler Mnemonic	Instructions
H_1	H_2	H_3	H_4		
15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0		
C= 0	0	/////	/////	HALT	Halt
C= 0	2	0	F	RSTS	Reset Programmable Status Indicators
C= 0	2	1	F	SETS	Set Programmable Status Indicators
C= 0	8	0	K	ENBL	Enable Interrupts
C= 0	8	4	K	ENBW	Enable and Wait
C= 0	8	8	K	DSBL	Disable Interrupt
C= 0	8	C	K	DSBW	Disable and Wait
C= 0	8	4/C	0	WAIT	Wait
C= 0	B	1	D	STSM	Status to Memory
C= 0	B	3	D	REGM	Registers to Memory
C= 0	B	4	D	RETN	Return from Interrupt
C= 0	B	5	D	MSTS	Memory to Status
C= 0	B	7	D	MREG	Memory to Registers

Notes:

Shaded areas are ignored.

F - Programmable status bits to be reset (RST) or set (SET) (bit positions correspond with status register).

K - Interrupt mask bits to be reset (ENBL) or set (DSBL) (corresponding mask bits in status register).

B - Address mode, absolute when 0 or relative when 1.

D - Address field (words), twos complement form for relative.

---

[1]More instructions in class code 0 are described under SUE 1110A and B, and 1112A and B Instruction Sets.

Control Instruction Definitions

HALT   HALT

| C	T	D
0	0	/////

15   11   7   0
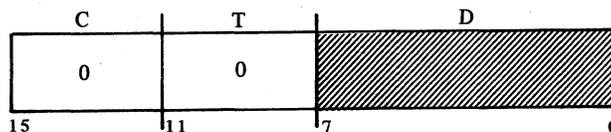
Further instruction execution ceases.   Execution resumes if the RUN
switch on the control panel is pressed or if RUN is enabled by another
processor.   Return from a HALT is to the next instruction in sequence.
Interrupts cause no resumption.

RESET PROGRAMMABLE STATUS INDICATORS   RSTS

C	T	I	F
0	2	0	Mask

15   11   7 6   0
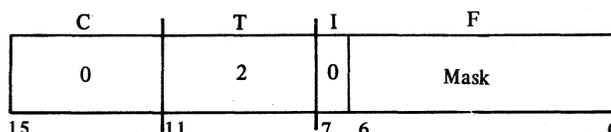
F is a mask for resetting status indicators.   For each corresponding
bit of F and the least significant seven bits of the status indicator word,
if F is ONE, the indicator is reset to ZERO.

SET PROGRAMMABLE STATUS INDICATORS   SETS

C	T	I	F
0	2	1	Mask

15   11   7 6   0
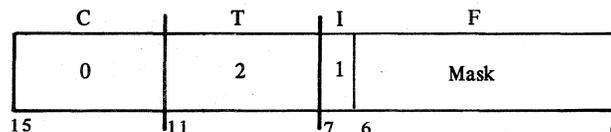
F is a mask for setting status indicators.   For each corresponding bit
of F, and the least significant seven bits of the status indicator word,
if F is ONE, the indicator is set to ONE.

F Bits for RSTS and SETS

| F bits set	Status Indicators
0	EQ EQUAL
1	GT GREATER THAN
2	OV OVERFLOW
3	CY CARRY
4	F1 FLAG 1
5	F2 FLAG 2
6	F3 FLAG 3

NOTE

In the following five interrupt control instruc-
tions, the K field (bits 3 through 0) corresponds
to status register bits 15 through 12 that enable
or disable interrupts 4 through 1, respectively.

ENABLE INTERRUPTS   ENBL
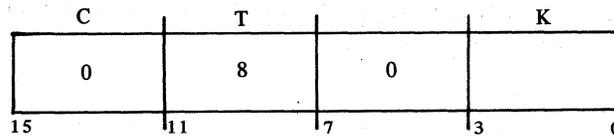
C	T		K
0	8	0	

15   11   7   3   0

Each ZERO in the K field is ignored, each ONE in the K field
enables the corresponding interrupt.

ENABLE and WAIT   ENBW
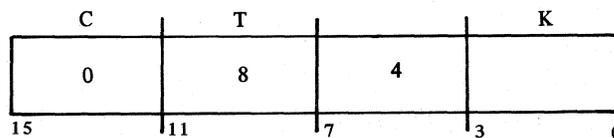
C	T		K
0	8	4	

15   11   7   3   0

Each ZERO in the K field is ignored, each ONE in the K field
enables the corresponding interrupt. The processor enters
the WAIT state until an enabled interrrupt occurs. If the
enabled interrupt is 4, the interrupt is processed in the
normal manner. If the enabled interrupt is 1, 2, or 3,
execution continues at the next instruction in sequence.
If no interrupts are enabled an instruction trap occurs.

DISABLE INTERRUPTS   DSBL

C	T		K
0	8	8	

15   11   7   3   0
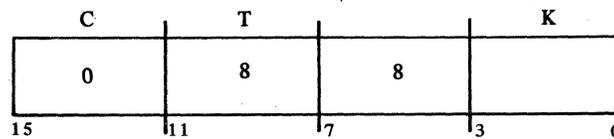
Each ZERO in the K field is ignored, each ONE causes an interrupt
disable for the corresponding interrupt.

DISABLE and WAIT   DSBW

C	T		K
0	8	C	

15   11   7   3   0
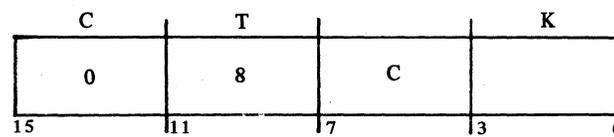
Each ZERO in the K field is ignored, each ONE in the K field
disables the corresponding interrupt. The processor enters
the WAIT state until a non-disabled interrupt occurs. If the
enabled interrupt is 4, the interrupt will be processed in the
normal manner. If the interrupt is 1, 2, or 3, execution
continues at the next instruction in sequence. If no interrupts
are enabled, an instruction trap occurs.

WAIT   WAIT

C	T	K	
0	8	4/C	0

15   11   7   3   0
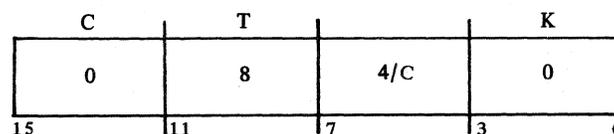
When the K-field is zero, no interrupt masking takes place, and the
processor enters the WAIT state until an interrupt occurs. If the
interrupt is level 4, it will be taken. The normal return after a
level 4 interrupt is to the WAIT instruction.

If the interrupt level is 1, 2, or 3, execution continues at the
next instruction in sequence. If no interrupts are enabled, an
instruction trap occurs.

STATUS TO MEMORY   STSM

C	B	OP	D
0		1	

15   11 10   7   0
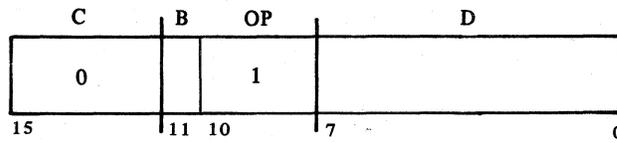
The content of the status indicator register replaces M. Relative
(B = 1) or absolute (B = 0) addressing is used to determine M.

REGISTERS TO MEMORY   REGM

C	B	OP	D
0		3	

15   11 10   7   0
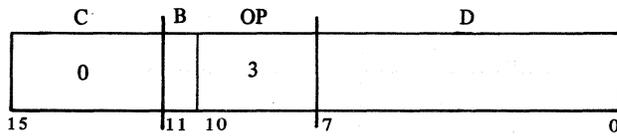
The general registers 1, 2, 3, 4, 5, 6, and 7 are stored into memory
in words M, M + 2, ..., M + 12. Relative (B = 1) or absolute (B = 0)
addressing is used to determine M.

RETURN FROM INTERRUPT   RETN

C	B	OP	D
0		4	

15   11 10   7   0
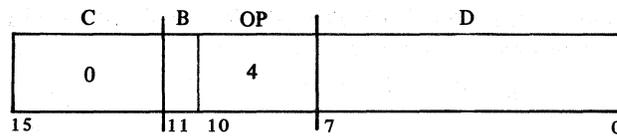
(M) replaces the status indicator register, and M + 2 replaces the Pro-
gram Counter, PC. Relative (B = 1) or Absolute (B = 0) addressing is
used to determine M.

MEMORY TO STATUS   MSTS

C	B	OP	D
0		5	

15   11 10   7   0
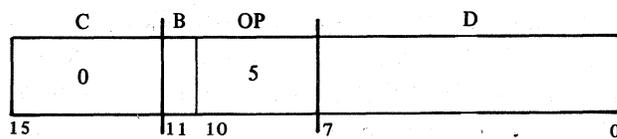
(M) replaces the content of the status indicator register. Relative
(B = 1) or absolute (B = 0) addressing is used to determine M.

MEMORY TO REGISTERS   MREG

C	B	OP	D
0		7	
15	11	10	7 0
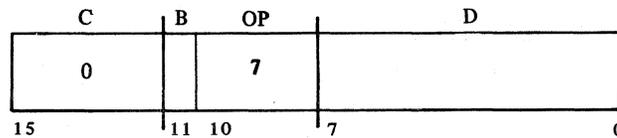
General registers 1, 2, 3, 4, 5, 6, and 7 are loaded from memory, from words M, M + 2, ..., M + 12. Relative (B = 1) or absolute (B = 0) addressing is used to determine M.

CONTROL INSTRUCTION TIMES. — Table 6 contains a list of the instruction execution times for control instructions.

Table 6. SUE 1110 (basic) Control Instruction Times

Instruction	Assembler Mnemonic	Time (microseconds)
Halt	HALT	1.01 + time to restart
Reset Programmable Status Indicators	RSTS	1.59
Set Programmable Status Indicators	SETS	1.72
Enable Interrupts	ENBL	1.85
Enable and Wait	ENBW	2.80 + time to interrupt
Disable Interrupts	DSBL	1.98
Disable and Wait	DSBW	2.80 + time to interrupt
Wait	WAIT	2.80 + time to interrupt
Status to Memory	STSM	2.14 Absolute, 2.46 Relative
Registers to Memory	REGM	7.24 Absolute, 7.56 Relative
Return from Interrupt	RETN	4.26 Absolute, 4.58 Relative
Memory to Status	MSTS	2.47 Absolute, 2.79 Relative
Memory to Registers	MREG	7.93 Absolute, 8.25 Relative

UNIMPLEMENTED INSTRUCTIONS

Instruction class codes B_{16} to F_{16} (11-15) have not been implemented for SUE 1110 Processors. They are reserved for instruction set expansion, and some have been expanded to accommodate Processors SUE 1111A and B, and 1112A and B, as described in the last four sections of this bulletin. Use of these class codes for SUE 1110 causes the instruction to be trapped for software interpretation of the instruction. Trapping an instruction refers to the action taken on unimplemented instructions. When an unimplemented bit combination is detected, a transfer is made to an interpretive subroutine that can either simulate the instruction execution or perform some specialized system functions.

INPUT/OUTPUT INSTRUCTIONS

There are no dedicated input-output instructions. The upper 4K addresses (out of a total 64K) are reserved for device addresses, control words, status words, etc. Input/output functions may be accomplished by ordinary general instructions, and status checking by test instructions.

SUE 1110A INSTRUCTION SET

INTRODUCTION

Processor SUE 1110A, an improved design of SUE 1110, provides the capability to set the key bits of the address bus. The SUE 1110A instruction set includes all of the instructions for SUE 1110 (basic) plus the Store Key, SKEY, instruction.

STORE KEY INSTRUCTION

A subclass of class code 0, the Store Key instruction has the following format:

C	T		K	
0	8	1	0 | 0	

15   11   7   3 2 1   0
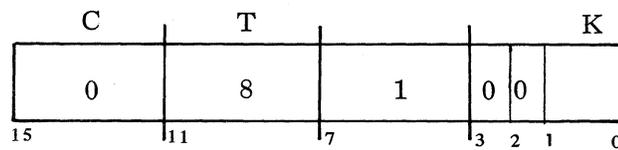
K – A two bit value to be stored into the key bits.

Refer to Appendix A for instruction times.

SUE 1110B INSTRUCTION SET

INTRODUCTION

SUE 1110B Instruction Set includes all of the instructions performed by Processors SUE 1110 (basic), and 1110A, and the two Fetch and Clear (word or byte) instructions described below.   Refer to Appendix A for instruction times.

FETCH AND CLEAR INSTRUCTIONS

A subclass of class code 0, Fetch and Clear allows implementation of multi-processor systems with shared resources.

FETCH AND CLEAR INSTRUCTION WORD FORMAT

	C	B	OP	I	AR	E	XR
FCLW FCLB	0		6				

15   11 10   7 6   3 2   0

```
┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
│           Extended Address      │
└ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
```

    B - Word when 0 (FCLW), byte when 1 (FCLB)

     I - Indirect when 1

  AR - Accumulator register designator (0-7)

     E - Extended or two-word instruction when 1

  XR - Index register designator (0-7), no indexing when 0

FETCH AND CLEAR OPERATION

This instruction reads and clears the designated memory word or byte and places the previous contents into the designated register. In particular, it allows a processor to read a memory operand without allowing another processor to read the same memory operand before it has been cleared by the first processor.

NOTE

Both the memory cell and the designated register are cleared by these instructions when performed by SUE 1110A, 1111A and 1112A processors. SUE 1110 (basic) processor traps on this instruction.

SUE 1111A INSTRUCTION SET

INTRODUCTION

The SUE 1111A Instruction Set includes all instructions described for processors
SUE 1110 (basic), SUE 1110A, and instructions described in this section under
SUE 1111A. Processor SUE 1111A can perform the following decimal arithmetic
instructions:

Instruction	Mnemonic	Operation Code	Description
Zero and Add	ZADD	2	Move decimal field
Add Decimal	ADDD	3	Add decimal fields
Subtract Decimal	SUBD	4	Subtract decimal fields
Compare Decimal	CMPD	5	Compare decimal fields
Shift Right	SFTR	8	Shift decimal field right
Move Right	MOVR	9	Move field right to left
Shift Left	SFTL	A	Shift decimal field left
Move Left	MOVL	B	Move field left to right
Compare Field	COMP	C	Compare fields

TEMPORARY STORAGE

The decimal instructions implemented in SUE 1111 use the storage locations
associated with the unimplemented instruction trap as temporary storage.
For example, locations 20, 22, and 24 for the CPU have the contents of regis-
ters 5, 6, and 7 during and after completion of a decimal instruction. If an
unimplemented instruction routine is to use a decimal instruction, it must save
and restore these locations.

INSTRUCTION FORMAT

All decimal instructions except Shift Right (SFTR) and Shift Left (SFTL) are accommodated by one standard format as follows:

	H1				H2				H3				H4			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	C				OP				0	R1			0	R2		
Word 1	L1				L2				0	X1			0	X2		

A modification of this format accommodates SFTR and SFTL instructions:

	H1				H2				H3				H4			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	C				OP				0	R1			0			
Word 1	L1				S				0	X1			0	X2		

C   Class Code – All instructions use class code C_{16}

OP   Operation Codes 2_{16} through 5_{16} and 8_{16} through C_{16}
(see instruction description)

R1   Register designator (1-7) for address of the most significant byte of the source field

R2   Register designator (1-7) for the address of the most significant byte of the destination field

L1   Source field length, minus 1 (bytes)

L2   Destination field length, minus 1 (bytes)

S   Number of decimal digit positions to be shifted.

X1   Index register, content of which is appended to the L1 field, (only if X1 is non-zero). Length is then L1 + (X1) + 1.

X2   Index register, content of which is appended to the L2 or S field, (only if X2 is non-zero). Length is then L2 + (X2) + 1. For Shift instructions the content of X2 is appended to the S field. The total digit positions shifted is then S + (X2).

NOTES

All seven general registers may be used as
the source for parameters. If zero is specified
as the register for X1 or X2, then only the L
or S field is used. R1 and R2 cannot be zero.

The target field must be large enough to hold
each operation result, else the result is trun-
cated without overflow status being set.

DECIMAL DATA FORMAT

Decimal data is formatted with two digits (4 bit fields or nibbles) per byte with
the right-most nibble taken as a sign. The valid decimal digits are 0 to 9_{16}
with A_{16} to F_{16} giving invalid results. Valid signs are C for plus and D for
minus.

Two examples of decimal data format required for operation codes 2, 3, 4, 5,
8, and A follow:

Example 1

0 1	3 8	5 C
Byte 1	Byte 2	Byte 3

Example 1 represents the number +1385.

Example 2

3 5	5 7	3 D
Byte 1	Byte 2	Byte 3

Example 2 represents the number −35573.

CHARACTER DATA FORMAT

The character data format used with operation codes 9, B, and C follow:

A	B	C
Byte 1	Byte 2	Byte 3

SYMBOLIC CODING FOR OPERANDS

Symbolic coding of the operand field for all instructions except Shift is: (R1), L1(X1), (R2), L2(X2). For Shift instructions, SFTR and SFTL, the coding is: (R1), L(X1), S(X2).

INSTRUCTIONS

ZERO AND ADD   ZADD

OP = 2

Zero And Add moves the source field to the destination field. If the destination field is longer than the source field, the excess high-order bytes are filled with zeros. If the destination field is shorter than the source field, the excess data is truncated. The move takes place from the right end first so that truncated data is in the most significant portion of the field.

ADD   ADDD

OP = 3

Add Decimal numerically adds the signed source field to the signed destination field with the sum placed into the destination field. If the sum is larger than the destination field, the excess high-order bytes are truncated. If the sum is smaller than the destination field, the excess high-order destination field bytes are filled with zeros.

Addition occurs from right to left one byte at a time. The sum contains the correct sign following the normal rules of algebra. If the result of the addition is zero, the result has the same sign as the original destination field.

SUBTRACT   SUBD

OP = 4

Subtract Decimal numerically subtracts the signed source field from the destination field with the difference placed into the destination field. If the difference is larger than the destination field, the excess high-order bytes are truncated. If the difference is smaller than the destination field, the excess high-order destination field bytes are filled with zeros.

Subtraction occurs from right to left one byte at a time. The difference contains the correct sign following the normal rules of algebra. If zero, the difference has the same sign as the original destination field.

<div style="text-align:center">COMPARE DECIMAL   CMPD</div>

OP = 5

Compare Decimal numerically compares the signed source field with the signed destination field. Status bits EQ (bit 0) and GT (bit 1) indicate the results of the compare. The status bits are affected as follows:

	GT	EQ
Source = Destination	0	1
Source > Destination	1	0
Source < Destination	0	0

Commensurate with the rules of order, positive and negative zeros are equal by the Decimal Compare instruction, and large negative numbers are smaller than small negative numbers.

The source and destination fields need not be the same length. The numeric values of the two fields are compared and leading zeros are ignored. Neither the source nor the destination field is altered by this instruction. To provide a faster decision algorithm, comparison is made first on the sign and units digits, then on digits in order of most significance.

<div style="text-align:center">SHIFT RIGHT   SFTR</div>

OP = 8

Shift Right performs a decimal digit shift to the right. Digit positions on the left end of the field are filled with zeros as shifting proceeds. Digits on the right end are shifted out around the sign and lost, leaving the sign unchanged. Note that the number of shifts refers to the digit positions shifted and not the number of bytes. Also note that the true number of shifts is given and not the number minus 1. Therefore, zero shifts can be specified causing no operation to be performed. Shift Right provides a fast way to divide by a power of ten, and can be used for decimal point alignment, etc.

MOVE RIGHT   MOVR

OP = 9

Move Right transfers the source field to the destination field. The right-most
byte (units/sign byte in decimal field) is moved first. Then, the move proceeds
to the left one byte at a time.

If the source field is smaller than the destination field, the remaining left end
of the destination field is unchanged. If the destination field is smaller than the
source field, the move proceeds to the left until the destination field is full; then
the move aborts so that all right-end bytes are transferred correctly. The
source field is left unchanged unless it overlaps the destination field.

SHIFT LEFT   SFTL

OP = A

Shift Left performs a decimal digit shift to the left. Digit positions on the right
end of the field, except the sign position, are filled with zeros as shifting pro-
ceeds. The sign is not shifted and left unaltered. Digits on the left end are
shifted out and lost.

Note that the number of shifts refers to the digit positions shifted and not the
number of bytes. Also note that the true number of shifts is given and not the
number minus 1. Therefore, zero shifts can be specified causing no operation
to be performed. Shift Left provides a fast way to multiply by a power of ten,
and can be used for decimal point alignment, etc.

MOVE LEFT   MOVL

OP = B

Move Left transfers the source field to the destination field. The left-most byte
is moved first, then the move proceeds to the right one byte at a time.

If the source field is smaller than the destination field, the remaining right end
of the destination field is left unchanged. If the destination field is smaller than
the source field, the move proceeds to the right until the destination field is

full; then the move aborts so that all left-end bytes are transferred correctly. The source field is left unchanged unless it overlaps the destination field.

COMPARE FIELD   COMP

OP = C

Compare Field compares the source field with the destination field. The compare operates from left to right. If either field is shorter than the other, the shorter field is considered to be extended to the right with ASCII blanks (A0) during the compare operation. Status bits EQ (bit 0) and GT (bit 1) indicate the compare results and are affected as follows:

	GT	EQ
Source = Destination	0	1
Source > Destination	1	0
Source < Destination	0	0

The compare assumes the collating sequence of ASCII (i. e. binary values of 8-bit characters as stored internally in core).

## SUE 1111B INSTRUCTION SET

INTRODUCTION

The SUE 1111B Instruction Set includes all instructions performed by processors SUE 1110 (basic), 1110A, and 1111A, and the two Fetch and Clear instructions described below. (Fetch and Clear, described for SUE 1110B and 1112B processors, is repeated here for programmer convenience.)

Refer to Appendix A for instruction times.

FETCH AND CLEAR INSTRUCTIONS

A subclass of class code 0, Fetch and Clear allows implementation of multi-processor systems with shared resources.

### FETCH AND CLEAR INSTRUCTION WORD FORMAT

```
              C    B   OP  I   AR   E   XR
          ┌──────┬──┬──────┬──┬─────┬──┬─────┐
FCLW      │  0   │  │  6   │  │     │  │     │
FCLB      │      │  │      │  │     │  │     │
          └──────┴──┴──────┴──┴─────┴──┴─────┘
          15     11 10     7  6     3  2     0
          ┌ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
          │        Extended Address          │
          └ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
```
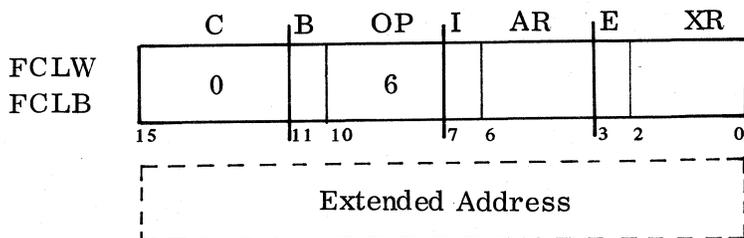
    B - Word when 0 (FCLW), byte when 1 (FCLB)

    I - Indirect when 1

  AR - Accumulator register designator (0-7)

    E - Extended or two-word instruction when 1

  XR - Index register designator (0-7), no indexing when 0

FETCH AND CLEAR OPERATIONS

This instruction reads and clears the designated memory word or byte and places the previous contents into the designated register. In particular it

allows a processor to read a memory operand without allowing another processor to read the same memory operand before it has been cleared by the first processor.

NOTE

Both the memory cell and the designated
register are cleared by this instruction when
performed with SUE 1110A, 1111A, and 1112A
processors. SUE 1110 (basic) processor
traps on this instruction.

SUE 1112A INSTRUCTION SET

INTRODUCTION

The SUE 1112A Double Precision Processor performs all instructions described in this bulletin for SUE 1110 (basic), 1110A, and the extended instructions described in this section under SUE 1112A. Following is a list of the 1112A extended instructions and their class codes:

Instruction	Class Codes (Hexadecimal)
Bit Manipulation	4
Move	4
Normalize and Count	A
Double-Length Shift	A
Single Precision Fixed Point	B
Double Precision Fixed Point	B
Control	0

DOUBLE PRECISION DATA FORMAT

Double-precision data operations are accommodated by the following format:

Word 0	S	Data (AR, even)
Word 1	S'	Data (AR+1, odd)

15   14   0

Word 0 - Most significant fifteen bits of the fixed-point number

Word 1 - Least significant fifteen bits of the fixed-point number

Range - $-(2^{30}) \leq$ Number $< +(2^{30})$

S - Sign bit of the 32-bit twos complement fixed-point number

S' - Sign bit extension in the least significant word

Upon termination of all

> double-length arithmetic normalize,
>
> double-length arithmetic shift, and
>
> double-precision add, subtract, and multiply

instructions, sign S' is adjusted to reflect sign S unless bits 14 through 0 of

word 1 are all zeros. In that event, sign S' also is zero. Also, on these

double-length instructions, and double-length Load And Store: the zero indi-

cator (ZE) reflects the condition of both words, the sign indicator (NG) reflects

the sign of the most significant word, and the odd indicator (OD) reflects the

value of the least significant bit of the least significant word.

NOTE

> To take the two's complement of a double
> precision number, use the following proce-
> dure: If the least significant word is not
> zero, take the two's complement of the least
> significant word and the one's complement of
> the most significant word. If the least signifi-
> cant word is zero, take the two's complement
> of the most significant word.

INSTRUCTION TIMES

Refer to Appendix A for timing of SUE 1112A arithmetic instructions.

BIT MANIPULATION INSTRUCTIONS

Bit Manipulation Instructions use a subclass of class code 4. Each of six

operations in the bit manipulation instructions may alter status indicators NG,

ZE, OD, and CY, for subsequent testing by Branch Conditional instructions.

OV is unaffected. CY = 0 indicates the designated bit or bits are all ZEROs.

## BIT MANIPULATION INSTRUCTION FORMATS

Three instruction word formats are used by bit manipulation instructions:

Single Bit Addressed by (XR)

C	OP	I	AR	E	XR
4	1-6	0		0	

15   11   7 6   3 2   0
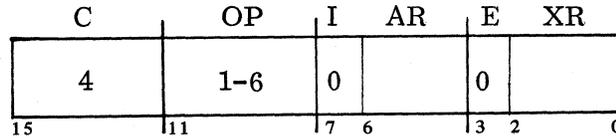
OP – Operation code 1-6

AR – Accumulator register (0-7) that contains the operation result

XR – Index register of which the least significant four bits contain the bit number to be tested and modified.

Single Bit Explicitly Designated by K
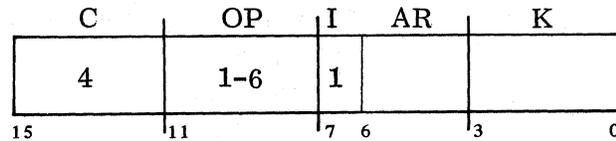
C	OP	I	AR	K
4	1-6	1		

15   11   7 6   3   0

OP – Operation code 1-6

AR – Designator (0-7) for the accumulator register that contains the operation result

K – A four-bit value that specifies the bit number to be tested.

Multiple Bits Selected by the Mask and (XR)
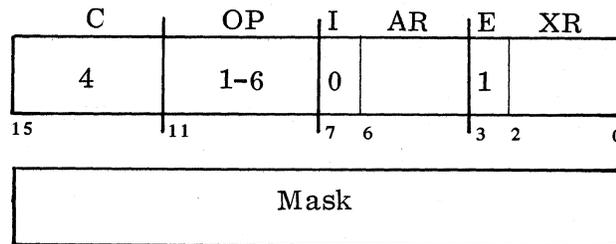
C	OP	I	AR	E	XR
4	1-6	0		1	

15   11   7 6   3 2   0

Mask

OP – Operation code 1-6

AR – Designator (0-7) for the accumulator register that contains the operation result

Mask – Second word of extended instruction to select the bits for testing and modification.

XR – If 0, only the mask field is used to determine the bits to be tested; if 1-7, the mask is ANDed with the content of XR to select the bits to be tested and modified.

BIT MANIPULATION OPERATIONS

Following are bit manipulation operations. Each of the operations (OP codes 1-6) tests the designated bit or bits and sets CY if any are not zero; resets CY if all are zero.

Operation Code	Mnemonic	Description
1	RBIT	Make the designated bit or bits a 0, all others unchanged
2	SBIT	Make the designated bit or bits a 1, all others unchanged
3	CBIT	Change the designated bit or bits, all others unchanged
4	IBIT	Isolate the designated bit or bits, all others reset to 0
5	TSBT	Test the designated bit or bits and shift (AR) left one. The bit shifted out of $(AR)_{15}$ is lost and a zero is shifted into $(AR)_0$
6	TBIT	Only test the designated bit or bits

MOVE INSTRUCTIONS

Another subclass of class code 4 is used by the Move instructions. There are four operations, each of which may alter status indicators NG, ZE, and OD for subsequent testing by Branch Conditional instructions. CY and OV are unaffected.

MOVE INSTRUCTION FORMAT

C	OP	AR	XR
4	7	I	J

15   11   7   6   3   2   0
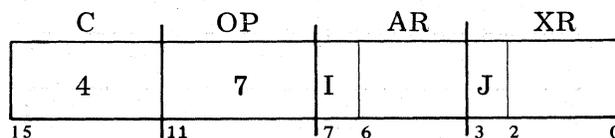
AR – Destination register designator

XR – Source register designator

MOVE OPERATIONS

Following are Move instruction operations:

I-J Codes	Mnemonic	Description
I=0, J=0	NEGT mov T	Move the twos complement value of register XR to register AR. The content of XR is unchanged unless XR = AR.
I=0, J=1	CPLM mov D	Move the ones complement value of register XR to register AR. The content of XR is unchanged.
I=1, J=0	MOVP	Move the positive magnitude of register XR to register AR. The content of XR is unchanged.
I=1, J=1	MOVM	Move the negative magnitude of register XR to register AR. The content of XR is unchanged.

NORMALIZE AND COUNT INSTRUCTIONS

A subclass of class A (Shift) instructions is used by the Normalize And Count instructions. There are 8 instructions, 4 single-, and 4 double-length.

NORMALIZE AND COUNT INSTRUCTION FORMAT

C		OP	AR		XR
A_{16}	B	Even	0	1	

15   11 10   7 6   3 2   0
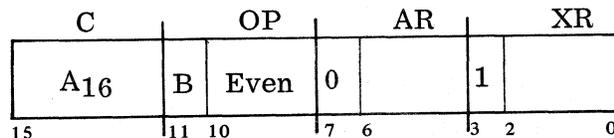
B - 0 = single length shift, 1 = double length shift

AR - Designator of register to be shifted

XR - Designator of register to be incremented by shift count

NORMALIZE AND COUNT OPERATIONS

Following are Normalize And Count operations:

Operation Code	Mnemonic	Description
Single Length		
0	SLAN	Single left arithmetic normalize
2	SLLN	Single left logical normalize
4	SRAN	Single right arithmetic normalize
6	SRLN	Single right logical normalize

Operation Code	Mnemonic	Description
Double Length		
0	DLAN	Double left arithmetic normalize
2	DLLN	Double left logical normalize
4	DRAN	Double right arithmetic normalize
6	DRLN	Double right logical normalize

The register (AR), or pair of registers, (AR and AR+1, where AR is even) is shifted in the direction indicated until the requested condition is met. The count of positions shifted is added to (XR). A maximum shift of 15 (31 for double) may occur. If the register or registers indicated for normalize contain zero, the instruction sets only the status bits; AR and XR are not altered. Also, on double arithmetic normalize, the sign position of the odd register is ignored by the zero test; but the position is set to the sign of the even register, or it is cleared if the odd register bits 14 to 0 are all ZERO. Double-length arithmetic format is described under INTRODUCTION at the beginning of this section.

SINGLE LEFT ARITHMETIC NORMALIZE   SLAN

OP = 0
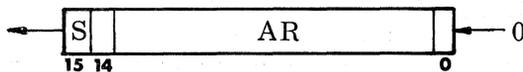
$$\left[\text{Shift stops when bits } (AR)_{15} \text{ and } (AR)_{14} \text{ are different}\right]$$

If any $(AR)_{14}$ is the same as $(AR)_{15}$, shift until these bits differ. Then add the shift count to (XR). ZEROs are shifted into $(AR)_0$ and bits shifted out of $(AR)_{15}$ are lost. Status bits NG, ZE, OD are affected accordingly. CY and OV are unaffected.

SINGLE LEFT LOGICAL NORMALIZE   SLLN

OP = 2
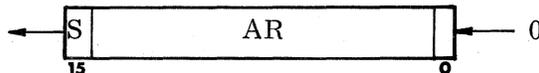
$$\left[\text{Shift stops when } (AR)_{15} = 1\right]$$

$(AR)_{15}$ is tested for a set condition; if not set, (AR) is shifted left. Zero is shifted into $(AR)_0$. When $(AR)_{15}$ is set, the operation terminates and the shift count is added to (XR). Status bits NG, ZE, and OD reflect the shift result. CY and OV are unaffected.

SINGLE RIGHT ARITHMETIC NORMALIZE   SRAN

OP = 4
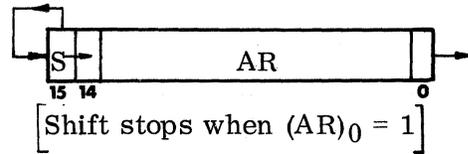
$$\left[\text{Shift stops when } (AR)_0 = 1\right]$$

$(AR)_0$ is tested for a set condition. If not set, (AR) is shifted right. $(AR)_{15}$, the sign bit, remains the same and is shifted into $(AR)_{14}$. When $(AR)_0$ is set, the operation terminates and the shift count is added to (XR). Status bits NG, ZE, OD reflect the shift result. CY and OV are unaffected.

SINGLE RIGHT LOGICAL NORMALIZE   SRLN

OP = 6   0 ⟶

$$\left[\text{Shift stops when } (AR)_0 = 1\right]$$
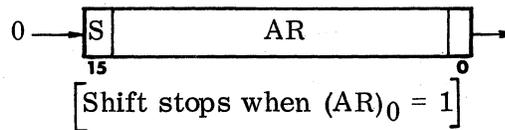
$(AR)_0$ is tested for a set condition. If not set, (AR) is shifted right. Zero bits are shifted into $(AR)_{15}$. When $(AR)_0$ is set, the operation terminates and the shift count is added to (XR). Status bits NG, ZE, and OD reflect the shift result. CY and OV are unaffected.

DOUBLE LEFT ARITHMETIC NORMALIZE   DLAN

OP = 2
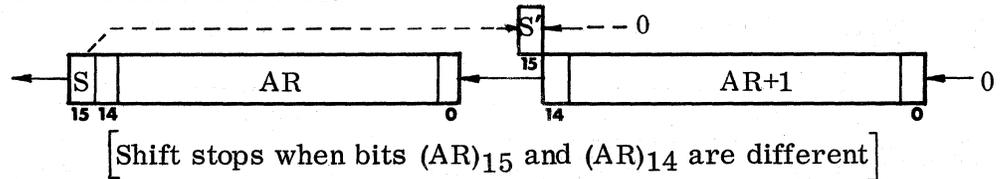
$$\left[\text{Shift stops when bits } (AR)_{15} \text{ and } (AR)_{14} \text{ are different}\right]$$

If any $(AR)_{14}$ is the same as $(AR)_{15}$, shift until these bits differ. Then add the shift count to (XR). ZEROs are shifted into $(AR+1)_0$ and $(AR+1)_{14}$ is shifted into $(AR)_0$. When shift stops: $(AR+1)_{15} = 0$ if $(AR+1)_{14-0} = 0$; if not, $(AR+1)_{15} = (AR)_{15}$. Status bits NG, ZE, and OD are affected accordingly. CY and OV are unaffected.

DOUBLE LEFT LOGICAL NORMALIZE   DLLN

OP = 2

$$\left[\text{Shift stops when } AR_{15} = 1\right]$$
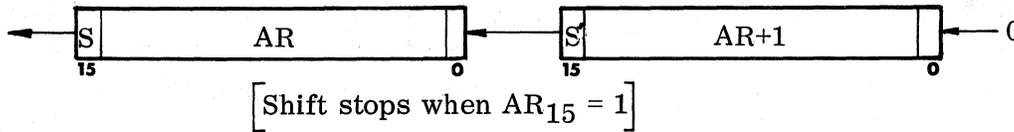
$(AR)_{15}$ is tested for a set condition; if not set, (AR) is shifted left. ZEROs are shifted into $(AR+1)_0$. $(AR+1)_{15}$ is shifted into $(AR)_0$. When $(AR)_{15}$ is set, the operation terminates and the shift count is added to (XR). Status bits NG, ZE, and OD reflect the shift result. CY and OV are unaffected.

DOUBLE RIGHT ARITHMETIC NORMALIZE   DRAN

OP = 4
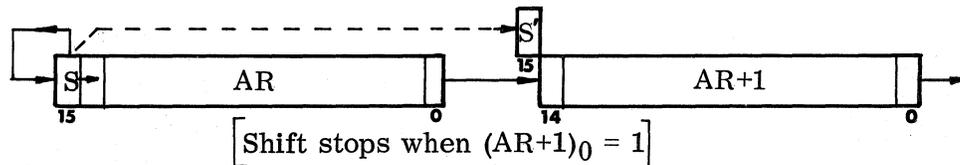
$$\left[\text{Shift stops when } (AR+1)_0 = 1\right]$$

$(AR+1)_0$ is tested for a set condition. If not set, (AR) and (AR+1) are shifted right. $(AR)_{15}$, the sign bit, remains the same and is shifted into $(AR)_{14}$. $(AR)_0$ is shifted into $(AR+1)_{14}$. When $(AR+1)_0$ is set, the operation terminates. Upon termination, $(AR+1)_{15}$ is adjusted to reflect the sign of the even register.

DOUBLE RIGHT LOGICAL NORMALIZE   DRLN

OP = 6

$$\left[\text{Shift stops when } (AR+1)_0 = 1\right]$$
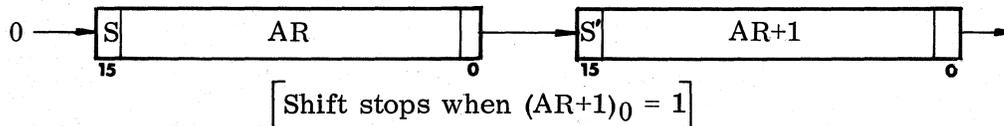
Zero is shifted into $(AR)_{15}$. $(AR+1)_0$ is tested for a set condition. If not set, (AR) and (AR+1) are shifted right and $(AR)_0$ is shifted into $(AR+1)_{15}$. When $(AR+1)_0$ is set, the operation terminates and the shift count is added to XR. Status bits NG, ZE, and OD reflect the shift result. CY and OV are unaffected.

DOUBLE LENGTH SHIFT INSTRUCTIONS

A subclass of class A (Shift) instructions is used by the Double-Length Shift Instructions. There are 8 instructions, corresponding to the 8 single-length shifts.

DOUBLE-LENGTH SHIFT INSTRUCTION FORMAT

Two formats are shown below. The shift count is contained either in the register designated by XR (bit 7 = 0), or in instruction bits 4 through 0, designated by K (bit 7 = 1).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Shift Indexed	A_{16}				1		OP		0	AR		▨	0		XR	
Shift Immediate	A_{16}				1		OP		1	AR			K			

AR – Designates the register pair (2, 3; 4, 5; or 6, 7) to be shifted

XR – Designates register containing the shift count

K – Shift count for immediate. Note that the field contains 5 bits, extending into bit position 4 (which is not used by AR).

OP – Shift operation code

DOUBLE-LENGTH SHIFT OPERATIONS

The double-length shift operations are similar to the single-length shift subclass, and are defined as follows:

| Operation Code	Mnemonic	Operation
0	DLAO	Double Left Arithmetic Open
1	DLLL	Double Left Logical Linked
2	DLLO	Double Left Logical Open
3	DLLC	Double Left Logical Closed
4	DRAO	Double Right Arithmetic Open
5	DRLL	Double Right Logical Linked
6	DRLO	Double Right Logical Open
7	DRLC	Double Right Logical Closed

On arithmetic shifts, the inter-register shift coupling is between bit 0 of the even register and bit 14 of the odd register.

## DOUBLE LEFT ARITHMETIC OPEN   DLAO

OP = 0
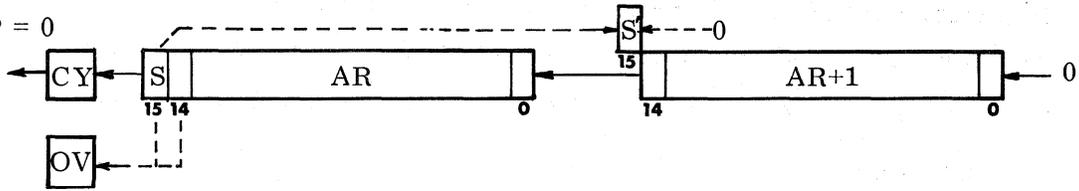
(AR) and (AR+1) are shifted left. $(AR)_{15}$ is shifted to carry (CY), and zeros are shifted into $(AR+1)_0$. If any $(AR)_{14}$ bit shifted is different than $(AR)_{15}$, overflow indicator, OV, is set. When shift stops: $(AR+1)_{15} = 0$ if $(AR+1)_{14-0} = 0$; if not, $(AR+1)_{15} = (AR)_{15}$. Operation affects status indicators CY, OV, NG, ZE, OD.

## DOUBLE LEFT LOGICAL LINKED   DLLL

OP = 1
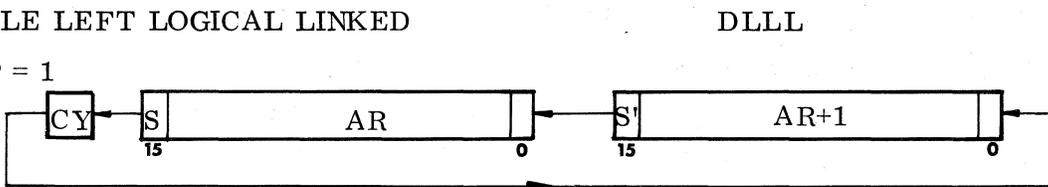
Carry (CY) is shifted into $(AR+1)_0$. $(AR)_{15}$ is shifted into CY. Operation affects status indicators: CY, NG, ZE, OD. (OV cannot set as in SLLL.)

DOUBLE LEFT LOGICAL OPEN   DLLO

OP = 2
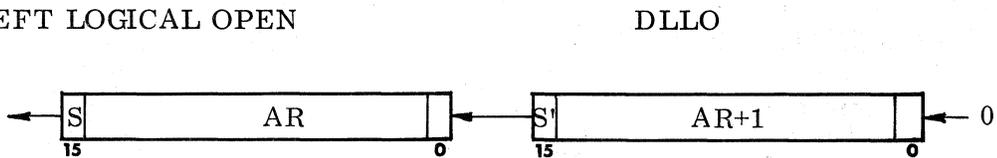
(AR) and (AR + 1) are shifted left. For each bit shifted, $(AR)_{15}$ is lost and $(AR+1)_0$ equals 0. Operation affects status indicators: NG, ZE, OD.

DOUBLE LEFT LOGICAL CLOSED   DLLC

OP = 3
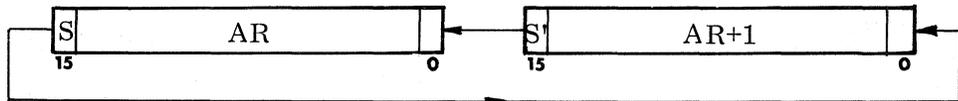
(AR) and (AR + 1) are shifted left. $(AR)_{15}$ is shifted into $(AR + 1)_0$. Operation affects status indicators: NG, ZE, OD.

## DOUBLE RIGHT ARITHMETIC OPEN DRAO

OP = 4
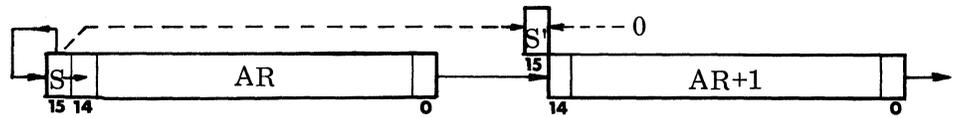
(AR + 1) and (AR) are shifted right. Sign bit $(AR)_{15}$ remains the same and is shifted into $(AR)_{14}$. $(AR+1)_0$ bits shifted out are lost. When shift stops: $(AR+1)_{15} = 0$ if $(AR+1)_{14-0} = 0$; if not, $(AR+1)_{15} = (AR)_{15}$. Operation affects status indicators: NG, ZE, OD.

DOUBLE RIGHT LOGICAL LINKED DRLL

OP = 5
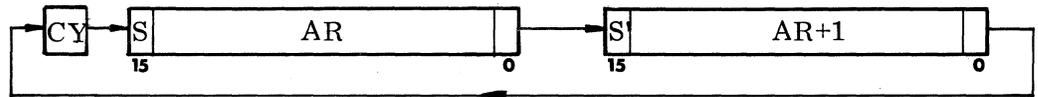
Carry (CY) is shifted into $(AR)_{15}$. $(AR+1)_0$ is shifted into CY. Operation affects status indicators CY, NG, ZE, OD.

DOUBLE RIGHT LOGICAL OPEN DRLO

OP = 6
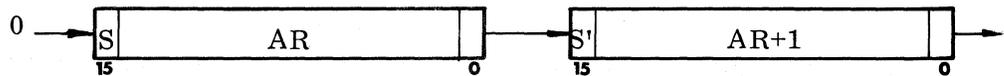
(AR) and (AR + 1) are shifted right. For each bit shifted, $(AR+1)_0$ is lost, and $(AR)_{15}$ equals 0. Operation affects status indicators: NG, ZE, OD.

DOUBLE RIGHT LOGICAL CLOSED DRLC

OP = 7
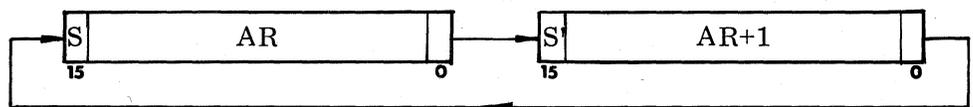
(AR) and (AR + 1) are shifted right. $(AR+1)_0$ is shifted into $(AR)_{15}$. Operation affects status indicators: NG, ZE, OD.

CLASS B INSTRUCTION SET

The Class B instruction set contains two sets of arithmetic instructions: single-precision fixed point and double-precision fixed point.

CLASS B INSTRUCTION FORMAT

```
                    OP      AR      XR
  ┌───────────┬──┬─────┬──┬─────┬──┬─────┐
  │    B16    │0 │     │I │     │E │     │
  └───────────┴──┴─────┴──┴─────┴──┴─────┘
  15         11 10     7  6    3  2      0
  ┌───────────────────────────────────────┐
  │                   M                     │
  └───────────────────────────────────────┘
```
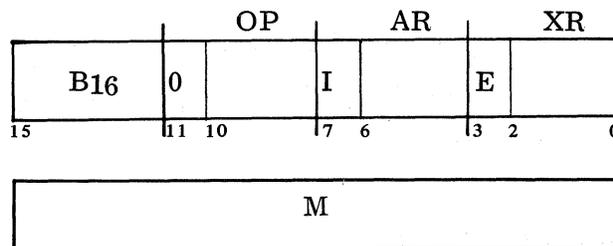
OP – Operation code

  I – Indirect address indicator

AR – Accumulator register pair for fixed point instructions (See definition below)

  E – Extended address indicator

XR – Index register designator

  M – Extended address (if required)

ACCUMULATOR REGISTERS

A register pair beginning with an even-numbered register, such as (R2, R3), (R4, R5) or (R6, R7) are defined as one accumulator for some single-precision and all double-precision fixed point instructions. An attempt to use R0 or R1 as an accumulator causes a level-5 interrupt.

SINGLE-PRECISION FIXED POINT INSTRUCTIONS

Three single-precision fixed point instructions supplement the basic instruction set. These extended instructions have a one-word memory operand and a two-register accumulator operand.

ADDRESSING MODES. – Standard memory-to-register addressing is permitted within this subclass. The accumulator registers are designated as a pair of registers addressed by the even-numbered register of the pair. The even-numbered register contains the most significant data.

ONE-WORD OPERAND FORMAT. - Following is the one-word operand format:

S	M
15 14	0
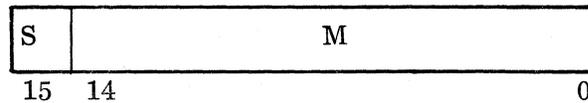
S - Sign bit of the 16-bit twos complement fixed point number

M - Remaining 15 bits of the fixed point number

$$-(2^{15}) \leq NUMBER < +(2^{15})$$

Format for double-length word is described under <u>INTRODUCTION</u> to this section.

SINGLE-PRECISION FIXED-POINT OPERATIONS

Three single-precision operations are described as follows:

Opera- tion Code	AR	Mnemonic and Operation	Description	Status Indicators Affected
3	2, 4, 6	MLTA (Multiply, Add)	Multiply the data in the odd-numbered register by the effective operand, and add the contents of the even-numbered register. A two-word product is formed in the combined registers.	NG, ZE, OD, CY
3	3, 5, 7	MULT (Multiply)	Multiply the data in the odd-numbered register by the effective operand to form a two-word product in the even-odd register pair.	NG, ZE, OD
4	2-3, 4-5, 6-7	DIVD (Divide)	Divide the data in the two-register accumulator by the effective operand. A properly-signed quotient results in the odd-numbered register, with the remainder (in the even-numbered register) having the same sign as the original dividend.	OV, NG, ZE, OD

NOTES

a. If register 0 or 1 is specified as AR, the instruction traps as an unimplemented instruction.

b. A multiplier or divisor of 8000 (i. e., - 65,536, the most negative number) has the same effect as if zero; results in setting OV on divide.

c. A multiplicand or addend of 8000 is treated as -65,536.

d. A dividend of 8000 0000 (i. e., most negative double-precision number) causes a divide check result.

DOUBLE PRECISION FIXED-POINT INSTRUCTIONS

Four double-precision, fixed-point instructions are provided in the extended instructions. Each has a two-word memory operand and a two-register accumulator operand. Format for double precision fixed point words is described under INTRODUCTION to this instruction set.

ADDRESSING MODES. - Standard memory-to-register addressing is permitted within the extended class. No other addressing modes are permitted. The effective memory address is the address of two consecutive memory words, the first containing the most significant data. The accumulator registers are designated as a pair of registers, addressed by the even-numbered register of the pair (e. g. R2 of the R2, R3 pair). The even-numbered register contains the most significant data.

DOUBLE PRECISION FIXED POINT OPERATIONS

Four operations are described as follows:

| Opera-tion Code	AR	Mnemonic and Operation	Description	Status Indicators Affected
5	2, 4, 6	DLOD (Double Load)	Move the contents of the two consecutive words located at the effective address to the combined registers.	NG, ZE, OD
0	2, 4, 6	DSTA (Double Store)	Move the contents of the two registers to the two consecutive words located at the effective address.	NG, ZE, OD
2	2, 4, 6	DADD (Double Add)	Add the contents of the consecutive words located at the effective address to the two registers.	CY, OV, NG, ZE, OD
1	2, 4, 6	DSUB (Double Subtract)	Subtract the contents of the two consecutive words located at the effective address from the two registers.	CY, OV, NG, ZE, OD

NOTE

If register 0 is specified as AR, the instruction traps as an unimplemented instruction.

CONTROL INSTRUCTIONS

Four control instructions (mnemonics SKEY, JKEY, LCPU, LKEY) are included in the SUE 1112A Instruction Set. Instruction Store Key (SKEY), described in the SUE 1110A Instruction Set, is repeated here for programming convenience.

CONTROL INSTRUCTION FORMATS

A subclass of class code 0, the control instructions use the following format:

```
             C        T            E    K
SKEY  ┌──────────┬──────────┬──────┬──┬──────┐
JKEY  │    0     │    8     │  1   │0 │0 │     │
      └──────────┴──────────┴──────┴──┴──────┘
      15        11         7      3  2  1     0
```
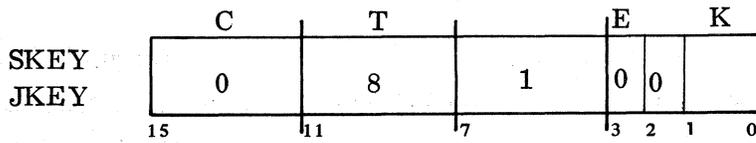
Jump Address

E - See description below

K - A two-bit value to be stored into the key bits of the address bus

Mnemonic	E	Description
SKEY	0	Store the value K into the key bits
JKEY	1	Store the value K into the key bits and the jump address into the program counter (i.e. jump)

```
             C        T       OP       XR
LCPU  ┌──────────┬──────────┬──────┬──┬──────┐
LKEY  │    0     │    8     │ 2-3  │0 │      │
      └──────────┴──────────┴──────┴──┴──────┘
      15        11         7      3  2       0
```
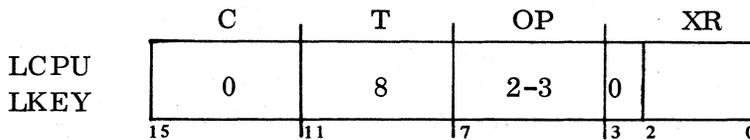
OP - Operation code 2-3

XR - Index register designator

| Operation Code	Mnemonic	Description
2	LCPU	Load the processor number into (XR) 5, 6; all other bits in (XR) are cleared.
3	LKEY	Load the Lkey value into (XR) right justified.

SUE 1112B INSTRUCTION SET

INTRODUCTION

SUE 1112B Instruction Set includes all of the instructions performed by processors SUE 1110 (basic), 1110A, 1112A and the two Fetch and Clear (word or byte) instructions described below. The Fetch and Clear Instructions, described also in the SUE 1110B and 1111B Instruction Sets, are repeated here for programmer convenience.

FETCH AND CLEAR INSTRUCTIONS

A subclass of class code 0, Fetch and Clear allows implementation of multi-processor systems with shared resources.

FETCH AND CLEAR INSTRUCTION WORD FORMAT

	C	B	OP	I	AR	E	XR
FCLW FCLB	0		6				

15   11 10   7 6   3 2   0

```
┌─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┐
│            Extended Address          │
└─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
```

   B - Word when 0 (FCLW), byte when 1 (FCLB)

   I - Indirect when 1

   AR - Accumulator register designator (0-7)

   E - Extended or two-word instruction when 1

   XR - Index register designator (0-7), no indexing when 0

FETCH AND CLEAR OPERATION

This instruction reads and clears the designated memory word or byte and

places the previous contents into the designated register. In particular it

allows a processor to read a memory operand without allowing another proces-

sor to read the same memory operand before it has been cleared by the first

processor.

NOTE

Both the memory cell and the designated
register are cleared by this instruction when
performed with SUE 1110A, 1111A and 1112A
processors. SUE 1110 (basic) processor
traps on this instruction.

Table A-1. SUE 1110 (Basic) General Register Instruction Times

General Instruction Class	Class Code	Address Mode	Assembler Mnemonic	Operation Word or Byte, Time in Microseconds							
				MOV OP = 0	SUB OP = 1	ADD OP = 2	AND OP = 3	IOR OP = 4	EOR OP = 5	CMP OP = 6	TST OP = 7
ACCUMULATOR TO MEMORY, AUTO DECREMENT	1	Indexed	R, (-R)	4.81	4.93	4.90	4.81	4.81	4.81	4.57	4.22
		Extended, Indexed	R, A(-R)	4.94	5.06	5.03	4.94	4.94	4.94	4.70	4.35
		Indexed, Indirect	R, *(-R)	5.95	6.07	6.04	5.95	5.95	5.95	5.71	5.36
		Extended, Indexed, Indirect	R, *A(-R)	6.23	6.33	6.30	6.23	6.23	6.23	5.97	6.21
ACCUMULATOR TO MEMORY, AUTO INCREMENT	2	Indexed	R, (R+)	4.81	4.93	4.90	4.81	4.81	4.81	4.57	4.22
		Extended, Indexed	R, A(R+)	4.94	5.06	5.03	4.94	4.94	4.94	4.70	4.35
		Indexed, Indirect	R, *(R+)	5.95	6.07	6.04	5.95	5.95	5.95	5.71	5.36
		Extended, Indexed, Indirect	R, *A(R+)	6.23	6.33	6.30	6.23	6.23	6.23	5.97	6.21
ACCUMULATOR TO MEMORY	3	Indexed	R, (R)	3.94	4.06	4.03	3.94	3.94	3.94	3.70	3.35
		Extended	R, A	4.07	4.19	4.16	4.07	4.07	4.07	3.83	3.48
		Extended, Indexed	R, A(R)	4.07	4.19	4.16	4.07	4.07	4.07	3.83	3.48
		Indexed, Indirect	R, *(R)	5.08	5.20	5.17	5.08	5.08	5.08	4.84	4.49
		Extended, Indirect	R, *A	5.34	5.46	5.43	5.34	5.34	5.34	5.10	5.34
		Extended, Indexed, Indirect	R, *A(R)	5.34	5.46	5.43	5.34	5.34	5.34	5.10	5.34
DATA TO ACCUMULATOR	4	Register to Register	R, R	2.50	2.79	2.79	2.50	2.50	2.50	2.69	2.50
		Immediate to Register	=H)X, R	2.50	2.79	2.79	2.50	2.50	2.50	2.69	2.50
		Literal to Register	=H)XXXX, R	3.18	3.47	3.47	3.18	3.18	3.18	3.37	3.18
		Literal, Indexed to Register	=H)XXXX(R), R	3.34	3.63	3.63	3.34	3.34	3.34	3.53	3.34
MEMORY TO ACCUMULATOR, AUTO DECREMENT	5	Indexed	(-R), R	4.09	4.38	4.38	4.09	4.09	4.09	4.41	4.09
		Extended, Indexed	A(-R), R	4.22	4.51	4.51	4.22	4.22	4.22	4.54	4.22
		Indexed, Indirect	*(-R), R	5.23	5.52	5.52	5.23	5.23	5.23	5.55	5.23
		Extended, Indexed, Indirect	*A(-R), R	5.49	5.78	5.78	5.49	5.49	5.49	5.81	5.49
MEMORY TO ACCUMULATOR, AUTO INCREMENT	6	Indexed	(R+), R	4.09	4.38	4.38	4.09	4.09	4.09	4.41	4.09
		Extended, Indexed	A(R+), R	4.22	4.51	4.51	4.22	4.22	4.22	4.54	4.22
		Indexed, Indirect	*(R+), R	5.23	5.52	5.52	5.23	5.23	5.23	5.55	5.23
		Extended, Indexed, Indirect	*A(R+), R	5.49	5.78	5.78	5.49	5.49	5.49	5.81	5.49
MEMORY TO ACCUMULATOR	7	Indexed	(R), R	3.35	3.64	3.64	3.35	3.35	3.35	3.67	3.35
		Extended	A, R	3.48	3.77	3.77	3.48	3.48	3.48	3.80	3.48
		Extended, Indexed	A(R), R	3.48	3.77	3.77	3.48	3.48	3.48	3.80	3.48
		Indexed, Indirect	*(R), R	4.49	4.78	4.78	4.49	4.49	4.49	4.81	4.49
		Extended, Indirect	*A, R	4.75	5.04	5.04	4.75	4.75	4.75	5.07	4.75
		Extended, Indexed, Indirect	*A(R), R	4.75	5.04	5.04	4.75	4.75	4.75	5.07	4.75

Table A-2. SUE 1110A/B, 1111A/B, and 1112A/B General Register Instruction Times

General Instruction Class	Class Code	Address Mode	OP Code	Operands	MOV (W/B) OP = 0	SUB (W/B) OP = 1	ADD (W/B) OP = 2	AND (W/B) OP = 3	IOR (W/B) OP = 4	EOR (W/B) OP = 5	CMP (W/B) OP = 6 Result ≤	CMP Result >	TST (W/B) OP = 7
ACCUMULATOR TO MEMORY, AUTO DECREMENT	1	Register Address (Indexed)	----	R, (-R)	4.73	4.86	4.76	4.73	4.73	4.73	4.34	4.43	4.31
		Extended (with/without Indexing)	----	R, A(-R)	4.73	4.89	4.79	4.76	4.76	4.76	4.37	4.46	4.34
		Register Address, Indirect (Word)[1]	---W	R, *(-R)	6.71	6.87	6.77	6.74	6.74	6.74	6.35	6.44	6.32
		(Byte)	---B	R, *(-R)	5.96	6.12	6.02	5.99	5.99	5.99	5.60	5.69	5.57
		Extended, Indirect (Word)[1]	---W	R, *A(-R)	6.32	6.48	6.38	6.35	6.35	6.35	5.96	6.05	5.93
		(Byte)	---B	R, *A(-R)	5.70	5.86	5.76	5.73	5.73	5.73	5.34	5.43	5.31
ACCUMULATOR TO MEMORY, AUTO INCREMENT	2	Register Address (Indexed)	----	R, (R+)	4.73	4.86	5.76	4.73	4.73	4.73	4.34	4.43	4.31
		Extended (with/without Indexing)	----	R, A(R+)	4.80	4.96	4.86	4.86	4.86	4.86	4.44	4.53	4.41
		Register Address, Indirect (Word)[1]	---W	R, *(R+)	6.71	6.87	6.77	6.74	6.74	6.74	6.35	6.44	6.32
		(Byte)	---B	R, *(R+)	5.96	6.12	6.02	5.99	5.99	5.99	5.60	5.69	5.57
		Extended, Indirect (Word)[1]	---W	R, *A(R+)	6.39	6.55	6.45	6.42	6.42	6.42	6.03	6.12	6.00
		(Byte)	---B	R, *A(R+)	5.77	5.93	5.83	5.80	5.80	5.80	5.41	5.50	5.38
ACCUMULATOR TO MEMORY	3	Register Address (Indexed)	----	R, (R)	3.76	4.22	4.12	4.09	4.09	4.09	3.70	3.79	3.67
		Extended (with/without Indexing)	----	R, A(R) or R, A	3.83	4.16	4.06	4.03	4.03	4.03	3.64	3.73	3.61
		Register Address, Indirect (Word)[1]	---W	R, *(R)	5.74	6.07	5.97	5.94	5.94	5.94	5.55	5.64	5.52
		(Byte)	---B	R, *(R)	4.99	5.32	5.22	5.19	5.19	5.19	4.80	4.89	4.77
		Extended, Indirect, (Word)[1]	---W	R, *A(R) or R, *A	5.42	5.75	5.65	5.62	5.62	5.62	5.23	5.32	5.20
		(w/wo Indexing) (Byte)	---B	R, *A(R) or R, *A	4.80	5.13	5.03	5.00	5.00	5.00	4.61	4.70	4.58
ACCUMULATOR/EXPLICIT DATA TO ACCUMULATOR	4	Register to Register	----	R, R	2.24	2.66	2.53	2.24	2.24	2.24	2.40	2.49	2.24
		Indexed, Literal	----	=H)XXXX, R	3.12	3.54	3.41	3.12	3.12	3.12	3.28	3.37	3.12
		Literal (Full Word)	----	=H)XXXX(R), R	2.96	3.38	3.25	2.96	2.96	2.96	3.12	3.21	2.96
		Immediate (Four-bits)	----	=H)X, R	2.24	2.66	2.53	2.24	2.24	2.24	2.40	2.49	2.24
MEMORY TO ACCUMULATOR, AUTO DECREMENT	5	Register Address (Indexed)	----	(-R), R	4.05	4.47	4.34	4.05	4.05	4.05	4.21	4.30	4.05
		Extended (with/without Indexing)	----	A(-R), R	4.08	4.50	4.37	4.08	4.08	4.08	4.24	4.33	4.08
		Register Address, Indirect (Word)[1]	---W	*(-R), R	6.06	6.48	6.35	6.06	6.06	6.06	6.22	6.31	6.06
		(Byte)	---B	*(-R), R	5.31	5.73	5.60	5.31	5.31	5.31	5.47	5.56	5.31
		Extended, Indirect (Word)[1]	---W	*A(-R), R	5.67	6.09	5.96	5.67	5.67	5.67	5.83	5.92	5.67
		(Byte)	---B	*A(-R), R	5.05	5.47	5.34	5.05	5.05	5.05	5.21	5.30	5.05
MEMORY TO ACCUMULATOR, AUTO INCREMENT	6	Register Address (Indexed)	----	(R+), R	4.05	4.47	4.34	4.05	4.05	4.05	4.21	4.30	4.05
		Extended (with/without Indexing)	----	A(R+), R	4.15	4.57	4.44	4.15	4.15	4.15	4.31	4.40	4.15
		Register Address, Indirect (Word)[1]	---W	*(R+), R	6.06	5.48	6.35	6.06	6.06	6.06	6.22	6.31	6.06
		(Byte)	---B	*(R+), R	5.31	5.73	5.60	5.31	5.31	5.31	5.47	5.56	5.31
		Extended, Indirect (Word)[1]	---W	*A(R+), R	5.74	6.16	6.03	5.74	5.74	5.74	5.90	5.99	5.74
		(Byte)	---B	*A(R+), R	5.12	5.54	5.41	5.12	5.12	5.12	5.28	5.37	5.12
MEMORY TO ACCUMULATOR	7	Register Address (indexed)	----	(R), R	3.54	3.96	3.83	3.54	3.54	3.54	3.70	3.79	3.54
		Extended (with/without Indexing)	----	A(R), R or A, R	3.48	3.90	3.77	3.48	3.48	3.48	3.64	3.73	3.48
		Register Address, Indirect (Word)[1]	---W	*(R), R	5.39	5.81	5.68	5.39	5.39	5.39	5.55	5.64	5.39
		(Byte)	---B	*(R), R	4.64	5.06	4.93	4.64	4.64	4.64	4.80	4.89	4.64
		Extended, Indirect, (Word)[1]	---W	*A(R), R or *A, R	5.07	5.49	5.36	5.07	5.07	5.07	5.23	5.32	5.07
		(w/wo Indexing) (Byte)	---B	*A(R), R or *A, R	4.45	4.87	4.74	4.45	4.45	4.45	4.61	4.70	4.45
JUMP OR CALL SUBROUTINE	4	Register Address (Indexed)	JUMP / JSBR	R / R, R	3.01								
		Extended (with/without Indexing)	JUMP / JSBR	A or A(R) / A, R or A(R), R	3.04								
		Register Address, Indirect[1]	JUMP / JSBR	*R / *R, R	5.02								
		Extended, Indirect[1] (w/wo Indexing)	JUMP / JSBR	*A or *A(R) / *A, R or *A(R), R	4.63								
FETCH AND CLEAR	0	Register Address (Indexed)	----	(R), R	5.02								
		Extended (with/without Indexing)	----	A(R), R or A, R	4.96								
		Register Address, Indirect (Word)[1]	FCLW	*(R), R	6.87								
		(Byte)	FCLB	*(R), R	6.12								
		Extended, Indirect (Word)[1]	FCLW	*A(R), R or *A, R	6.55								
		(Byte)	FCLB	*A(R), R or *A, R	5.93								

NOTE

[1] Add 1.01 microseconds for each additional level of indirect.

Table A-3. SUE 1110 (Basic), 1110A/B, 1111A/B and 1112A/B
Control Instruction Times

Instruction	Assembler Mnemonic	Time (microseconds)
Halt	HALT	1.01 + time to restart
Reset Programmable Status Indicators	RSTS	1.59
Set Programmable Status Indicators	SETS	1.72
Enable Interrupts	ENBL	1.85
Enable and Wait	ENBW	2.80 + time to interrupt
Disable Interrupts	DSBL	1.98
Disable and Wait	DSBW	2.80 + time to interrupt
Wait	WAIT	2.80 + time to interrupt
Status to Memory	STSM	2.14 Absolute, 2.46 Relative
Registers to Memory	REGM	7.24 Absolute, 7.56 Relative
Return from Interrupt	RETN	4.26 Absolute, 4.58 Relative
Memory to Status	MSTS	2.47 Absolute, 2.79 Relative
Memory to Registers	MREG	7.93 Absolute, 8.25 Relative
Store Key	SKEY*	2.6
*Not available in SUE 1110 (Basic) Processor		

SINGLE SHIFT INSTRUCTION TIMING FOR SUE 1110 (Basic), 1110A/B, 1111A/B, 1112A/B

Shift instruction execution times depend on the number of single bit shifts specified in either the K field (immediate) or the selected register, XR. The time is calculated by the formula:

$$T_s = 2.76 + (0.26)N$$

where N = 0, 1, ..., 15.

Table A-4. SUE 1110 (Basic) Branch Instruction Times

Instruction	Assembler Mnemonic	Time (microseconds)	
		Next Word	Branch
No Operation	NOPR	1.78	-
Branch Unconditional	BRUN	-	2.72
Branch True	BxxT*	1.78	2.72
Branch False	BxxF*	1.78	2.72
Branch Less Than True	BLTT	1.75	3.08
Branch Less Than False	BLTF	1.88	3.08
*where xx = EQ, GT, OV, CY, F1, F2, F3, LP, OD, ZE, NG.			

Table A-5. SUE 1110A/B, 1111A/B and 1112A/B
Branch Instruction Times

Instruction	Assembler Mnemonic	Time (microseconds)	
		Next Word	Branch
No Operation	NOPR	1.75	-
Branch Unconditional	BRUN	-	2.82
Branch True	BxxT*	1.75	2.82
Branch False	BxxF*	1.75	2.82
Branch Less Than True	BLTT	1.75	3.21
Branch Less Than False	BLTF	1.88	3.21
*where xx = EQ, GT, OV, CY, F1, F2, F3, LP, OD, ZE, NG.			

SUE 1111A/B CLASS C INSTRUCTION TIMES

DECIMAL AND CHARACTER INSTRUCTIONS

DECIMAL ADD AND SUBTRACT TIMING

Both the decimal add and subtract operations are started with a sign analysis to determine whether true addition (ADDD with like signs or SUBD with unlike signs), or true subtraction (ADDD with unlike signs or SUBD with like signs) is to occur. Timing for both situations can be calculated using the following case formats and procedures:

TRUE ADDITION CASE FORMATS

a. Propagate
    if final carry
    out of L

					L
XX	XX	<99	=99	=99	DL

$P \geq 0$ |←—P—→| SL

$DL > SL = L$

b. Propagate and
    overflow if
    final carry
    out of L

			L
=99	=99	=99	DL

|←— P —→| SL

$DL - SL = P > 0$ $DL > SL = L$

c. No propagate and
    overflow, if
    final carry
    out of L

L
DL
SL

$P = 0$ $DL = SL = L$

d. No propagate and
    overflow, if
    final carry
    out of L

	L
	DL
SL	

$P = 0$ $SL > DL = L$
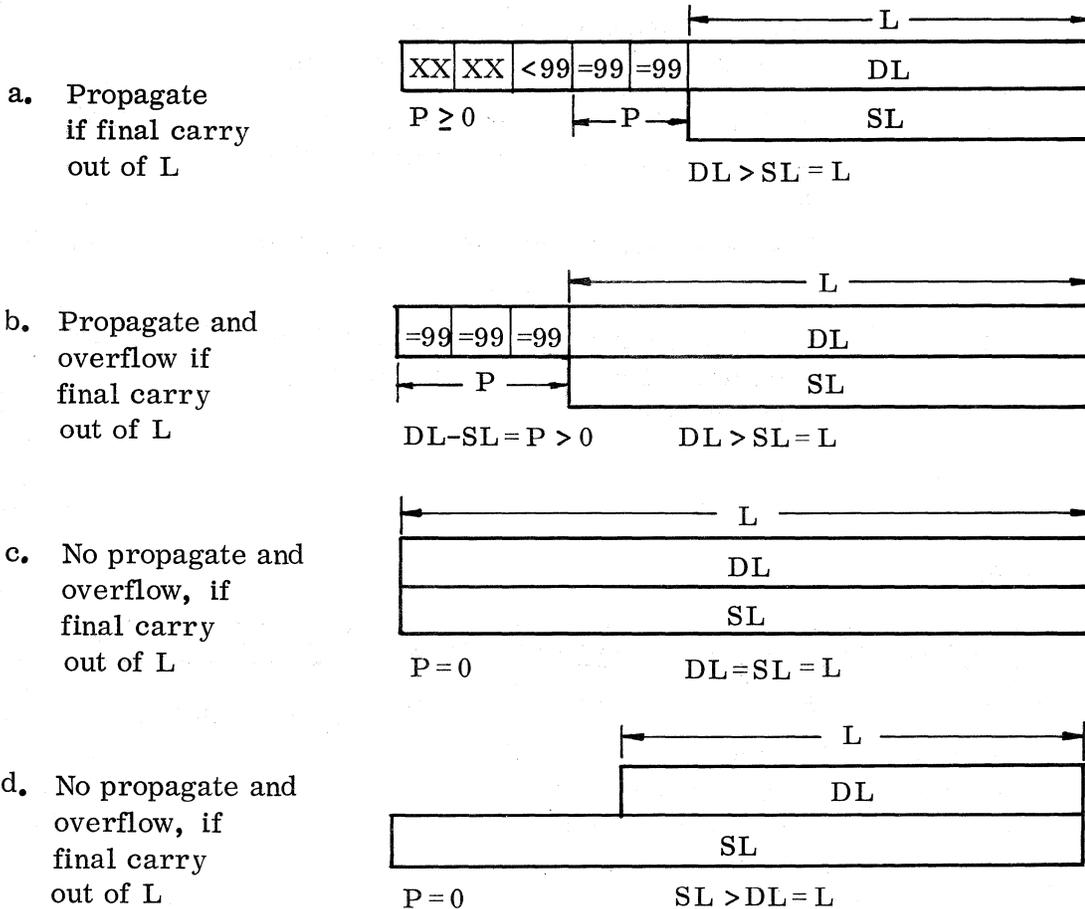
TRUE ADDITION TIMING PROCEDURES

Step	For Case Formats	Conditions (Times in Microseconds)	Times (Microseconds)
1	a, b, c, d	Start with the basic time of 10.70 and go to Step 2.	10.70
2	a, b, c, d	For each digit pair to be added (including units and signs) add in 3.21 and go to Step 3.	3.21L
3	a, b, c, d	STOP if no final carry from the summation; otherwise take Step 4.	+0
4	c, d	Add in 0.26 and STOP if $SL \geq DL$; otherwise take Step 5.	+0.26
5	a, b	For each digit pair in the extension field (i.e. where $DL > SL$ or $P > 0$) that equals 99, add in 2.7 until a non-99 pair is encountered (take Step 6), or the extension field runs out (take Step 8).	2.70P
6	a	Add in 2.30 and STOP if the LSD of an extension pair is not 9; otherwise take Step 7.	+2.30
7	a	Add in 2.33 and STOP when the MSD of an extension pair is not 9.	+2.33
8	b	Add in 0.26 and STOP when the extension field runs out (i.e. only 99 is encountered).	+0.26

Example

| 07 | 99 | 74 | 3+ | minus | 42 | 6- | equals | 08 | 00 | 16 | 9+ |

|—P—| P=1, |— L —| L=2,
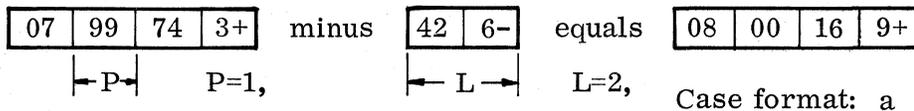
Case format: a

```
time = 10.70   basic
   3 x 3.21    summation
        2.70   propagate carry
    +   2.30   terminate carry
       ─────
       22.12   microseconds total
```

TRUE SUBTRACTION CASE FORMATS

e. Propagate
if final borrow
out of L

XX	XX	>0	00	00	DL

|←—P—→| SL |

$P \geq 0$ $DL > SL = L$

f. Propagate and
recomplement
if final borrow
out of L

00	00	00	DL

|←——P——→| SL |

$DL - SL = P$ 0 $DL > SL = L$

g. No propagate and
recomplement
if final borrow
out of L

DL
SL

$P = 0$ $DL = SL = L$

h. No propagate and
recomplement
if final borrow
out of L

DL

SL

$P = 0$ $SL > DL = L$
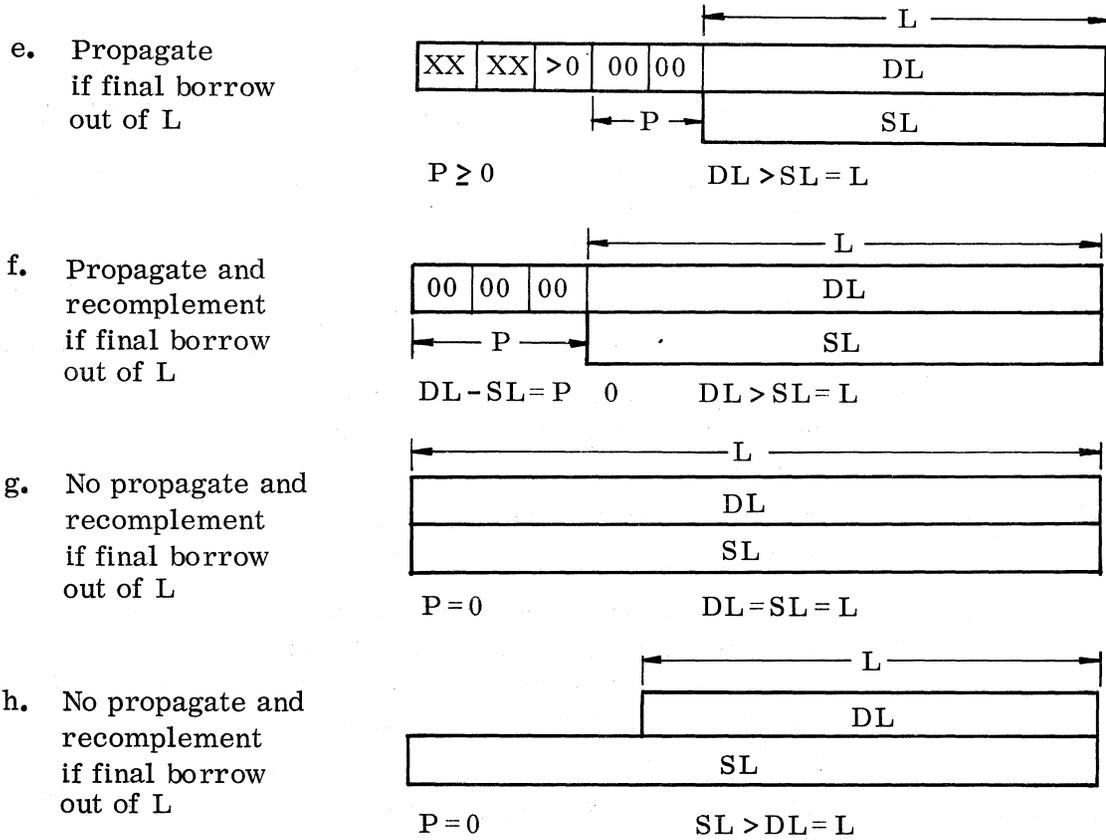
RECOMPLEMENT CASE FORMATS

j. Units digit $\neq$ 0,
start nine's complement

	>0±

$N = DL - 1$

k. Short ten's
complement

XX	XX	XX	XX	XX	XX	XX	XX	XX	>0	0±

$N = DL - 2$

l. Extended ten's
complement

XX	XX	XX	XX	XX	XX	XX	>0	00	00	0±

$N = DL - 2 - T$ T
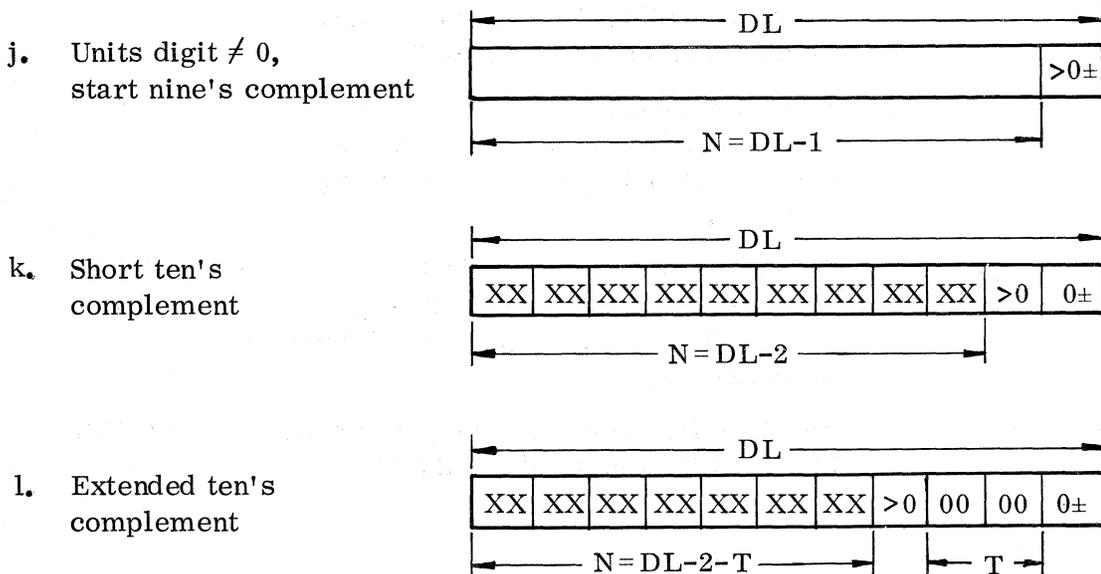
TRUE SUBTRACTION TIMING PROCEDURE

Step	For Case Formats	Conditions (Times in Microseconds)	Times (Microseconds)
1	e, f, g, h	Start with the basic time of 10.99 and go to Step 2.	10.99
2	e, f, g, h	Add in 3.21 for each digit pair to be subtracted (including units and signs) and go to Step 3.	3.21L
3	e, f, g, h	STOP if there is no final borrow from the subtraction process; otherwise take Step 4.	+0
4	g, h	Go to Step 7 if SL $\geq$ DL; otherwise take Step 5.	
5	e, f	Add in 2.01 for each digit pair in the extension field (i.e. where DL > SL or P > 0) until a non-zero is encountered (take Step 6), or the extension field runs out (take Step 7).	2.01P
6	e	Add 2.01 and STOP when a non-zero digit is encountered in the extension field.	+2.01
7	j, k, l	Add 2.43 (for first step of recomplement); go to Step 10 if the units digit is not zero; otherwise go to Step 8.	2.43
8	l	Add 1.33 for each digit pair in the destination field that is zero until a non-zero pair is encountered, then go to Step 9.	+1.33T
9	k, l	Add 2.14 if the LSD of a digit pair is not zero; add 2.40 if the LSD is zero and the MSD of a digit pair is not zero. Go to Step 10.	2.14 2.40
10	j, k, l	Add 1.75 for each remaining digit pair in the destination field and STOP.	+1.75N

Four Examples of Subtract Timing.

1. No Borrow:

| 78 | 29 | 4- | plus | 18 | 2+ | equals | 28 | 11 | 2- |

⊢— L —⊣   L = 2

Case format: e, f, g, or h

time = 10.99   basic
2 x   3.21   subtraction
+   0.00   no borrow
17.41   microseconds total

2. Borrow Without Recomplement:

| 02 | 00 | 22 | 64 | 5+ | minus | 72 | 70 | 5+ | equals | 01 | 99 | 99 | 94 | 0+ |

—⊢ P ⊢—   P = 1   ⊢—— L ——⊣   L = 3

Case format: e

time = 10.99   basic
3 x   3.21   subtraction
2.01   propagate borrow
+   2.01   terminate borrow
24.64   microseconds total

3. Recomplement:

| 01 | 07 | 23 | 5- | minus | 00 | 01 | 09 | 23 | 5- | equals | 00 | 02 | 00 | 04 |

⊢—— L ——⊣   L = 4 $N = 1, T = 1$ —⊢ N ⊢— —⊢ T ⊢—

Case format: g or h

time = 10.99   basic
4 x   3.21   subtraction
2.43   tens complement units and sign
1.33   tens complement zero
2.14   tens complement
+   1.75   nines complement
31.48   microseconds total

4. Borrow and Recomplement:

| 00 | 00 | 00 | 72 | 9+ | plus | 00 | 73 | 8- | equals | 00 | 00 | 00 | 00 | 9- |

⊢— P —⊣   P = 2   ⊢—— L —⊣   ⊢——— N ———⊣

$L = 3, N = 4, T = 0$

Case format: f

time = 10.99   basic
3 x   3.21   subtraction
2 x   2.01   propagate borrow
2.43   tens complement units and sign
+ 4 x   1.75   nines complement
34.07   microseconds total

DECIMAL SHIFT TIMING

Table A-6 lists times for Shift Left (SFTL) and Shift Right (SFTR) instructions.

Table A-6. Decimal Shift Timing Chart

Shift Left Time in Microseconds

Shift Count	1	2	3	4	5	6	7
0	●———— 11.14 ————●						
1	●	16.51	18.27	20.03	21.79	23.55	25.31
2		15.30	19.34	21.10	22.86	24.62	26.38
3		●	17.36	19.12	20.88	22.64	24.40
4			16.15	20.19	21.95	23.71	25.47
5			●	18.21	19.97	21.73	23.49
6				17.00	21.04	22.80	24.56
7	13.43			●	19.06	20.82	22.58
8		14.28			17.85	21.89	23.65
9			15.13		●	19.91	21.67
10				15.98		18.70	22.74
11					16.83	●	20.76
12							19.55
13						17.68	18.53
14	●	●	●	●	●	●	18.53

Shift Right Time in Microseconds

Shift Count	1	2	3	4	5	6	7
0	●———— 11.64 ————●						
1	●	17.06	19.17	21.28	23.39	25.50	27.61
2		16.38	18.37	20.13	21.89	23.65	25.41
3		●	17.91	20.02	22.13	24.24	26.35
4			17.23	19.22	20.98	22.74	24.50
5			●	18.76	20.87	22.98	25.09
6				18.08	20.07	21.83	23.59
7	13.14			●	19.61	21.72	23.83
8		13.99			18.93	20.92	22.68
9			14.84		●	20.46	22.57
10				15.69		19.78	21.77
11					16.54	●	21.31
12						17.39	20.63
13							18.24
14	●	●	●	●	●	●	18.24

Extrapolation Formulae:

Add 0.85 for each additional shift in zero character.

Add 1.76 for each additional left-shifted character.

Add 1.76 for each additional right-shifted character when shift count is even.

Add 2.11 for each additional right-shifted character when shift count is odd.

Note: Field length (SL) is in actual number of characters (i.e. digit pairs) and will be 1 greater than the length specified value in the instruction.

CLASS C NON-IMPLEMENTED OP-CODES

All SUE 1111A and 1111B, class C instructions (Operation Codes 0, 1, 6, 7, D, E and F) trap. Time to trap takes 16.28 microseconds.

MOVE TIMING

Time (T) for MOVR, MOVL, and ZADD is calculated using the following formulae:

Mnemonic	Formula
MOVR or MOVL	$T = 11.36 + 1.76N$
ZADD	$T = 11.36 + 1.76N + 0.85Z$

where:

   N is the number of characters (i.e. DL or SL, whichever is smaller)

   Z is the number of zeros ($Z = DL - SL$ if $DL > SL$).

COMPARE-FIELD TIMING

The following general format applies to Compare-Field timing calculations.
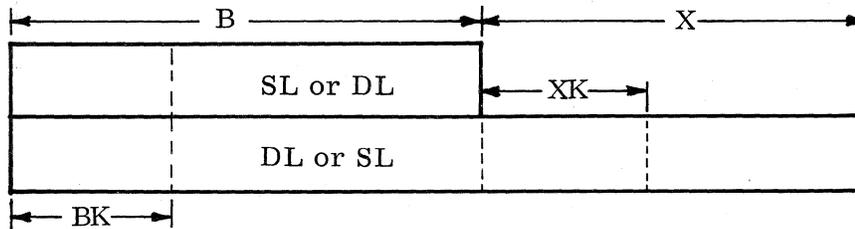
where:

   SL = Source Length

   DL = Destination Length

    X = Difference in Length in characters, SL − DL

    B = Length of the shorter field, SL or DL

   XK = Number of leading blank characters (Ƅ) in the extension field (X) before a non-blank character is encountered.

   BK = Number of character pairs in the two body fields that are in corresponding positions, before a non-equal pair is encountered.

COMPARE FIELD TIME CALCULATIONS

Time (T) in microseconds can be calculated for the six distinct cases as follows:

Body Fields Equal, SL = DL

A	B	C
A	B	C
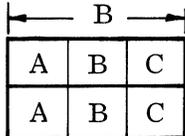
$X = XK = 0$
$B = BK > 0$

$T = 11.35 + 2.4B$

Unequal Pair in Body Scan

Z	Y	X	W
Z	Y	A	B
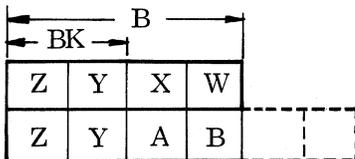
$B > BK \geq 0$

$T = 13.72 + 2.4BK$

Body Fields Equal, SL > DL, or SL < DL, Extension Field Blank

Z	Y	X
Z	Y	X
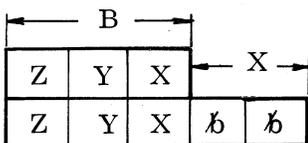
$B = BK > 0$
$X = XK > 0$

SL > DL:
$\quad T = 11.64 + 2.4B + 1.2X$

SL < DL:
$\quad T = 11.51 + 2.4B + 1.2X$

Body Fields Equal, SL > DL or SL < DL, Non-blank in Extension Field

A	B	XK
A	B	D
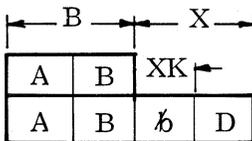
$B = BK > 0$
$X > XK \geq 0$

SL > DL:
$\quad T = 13.39 + 2.4B + 1.2XK$

SL < DL:
$\quad T = 13.26 + 2.4B + 1.2XK$

DECIMAL COMPARE TIMING

Decimal comparison is done in a manner to provide the fastest comparison result. The fields are scanned in the following order of significance: signs and units digits, extension fields, body fields. The extension field (of length X) is the most significant portion of the longer comparand and includes all digits of more significance than the most significant digit of the shorter comparand. The body fields (of length B) are those portions of the comparands remaining (i. e. not units or signs, or extension). Comparison may take one of four basic formats based on the relative lengths of the fields:

1. Source length = destination length = 1
    (both fields are units digits and signs).

U	S
U	S
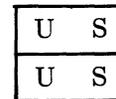
2. Source length ≠ destination length, but one field = 1
    (one comparand is units and sign).
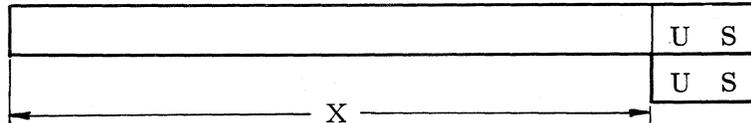
3. Source length = destination length > 1
    (both fields are equal length and more than units and sign).
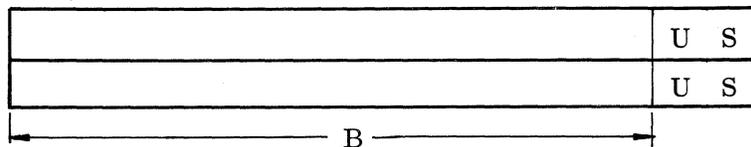
4. Source length ≠ destination length; both fields > 1
    (not equal length but both fields are more than units and sign).
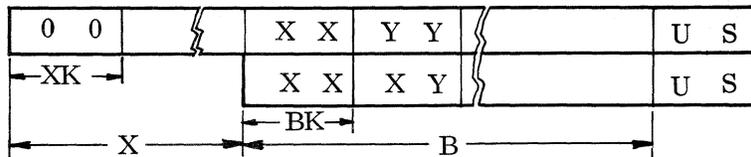
where:
    X is the difference in field lengths in characters (digit pairs).
    B is the length of the shorter field minus one.
    XK is the number of leading zero characters in the extension field.
    BK is the number of leading characters in the body fields which are
        equal in corresponding positions (or are both zero in the case of
        Zero Scan), before non-equal (or non-zero) pair is encountered.

DECIMAL COMPARE TIME CALCULATIONS

Time (T) in microseconds can be calculated for three cases as follows:

Case 1 - The signs are not equal and the units digits are not both zero.

	0	C
	2	D

$X \geq XK \geq 0$
$B \geq BK \geq 0$

$T = 13.08$

Case 2 - The signs are not alike and both units digits are zero. A zero scan is evoked.

Case 2 Formats:

a.

0D
0C

$X = XK = B = BK = 0$

b.

00	00	00	0C
			0D

$\longleftarrow$ X $\longrightarrow$

$X = XK > 0$
$B = BK = 0$

c.

00	00	00	00	0D
	00	00	00	0C

$\leftarrow$X$\rightarrow$
$\longleftarrow$ B $\longrightarrow$

$X = XK \geq 0$
$B = BK > 0$

d.

00	00	05	0C
			0D

$\leftarrow$XK$\rightarrow$
$\longleftarrow$ X $\longrightarrow$

$X > XK > 0$
$B = BK = 0$

e.

00	00	00	10	0D
	00	02	00	0C

$\leftarrow$X$\rightarrow$BK$\leftarrow$
$\longleftarrow$ B $\longrightarrow$

$X = XK \geq 0$
$B > BK \geq 0$

DECIMAL COMPARE TIMING PROCEDURE FOR CASE 2

Step	For Case Formats	Conditions (Times in Microseconds)	Times (Microseconds)
1	a, b, c, d, e	Start with basic time of 14.40 and go to Step 2.	14.40
2	a, b, c, d, e	Add 0.13 if destination comparand sign is positive, and go to Step 3.	0.13
3	b, c, d, e	Add 0.03 if SL > DL, and go to Step 4.	0.03
4	b, c, d, e	Add 1.17 for each leading zero character in the extension field and go to Step 5.	1.17 XK
5	d	Add 1.69 and STOP (Note 1) if there is a non-zero character in the extension field; otherwise go to Step 6.	+1.69
6	a, b	Add 0.42 and STOP (Note 2) if B=0 (i.e. no body) and if either the extension field is all zero ([X]=0) or there is no extension field (X=0); otherwise go to Step 7.	+0.42
7	c, e	Add 2.53 for each pair of leading zero characters in the body field. Go to Step 8.	2.53BK
8	e	Add 3.18 and STOP (Note 1) for the first non-zero character encountered in either body field; otherwise go to Step 9.	+3.18
9	c	Add 0.81 and STOP (Note 2) if both body fields are all zero (i.e. +0 and −0).	+0.81

Note 1 - The result is Greater-Than or Less-Than depending upon the signs.

Note 2 - The result is Equal (and specifically a positive zero equals a negative zero).

Case 3 - The signs are alike. A compare scan is evoked.

Case 3 Formats:

f.

7D
3D

$X = XK = B = BK = 0$

g.

00	00	00	2C

←———— X ————→ | 9C |

$X = XK > B = BK = 0$

h.

00	02	37	5C
	02	37	4C

←X→ ←— B —→

$X = XK \geq 0$

$B = BK > 0$

j.

00	90	73	8D
			8D

→ XK ←
←———— X ————→

$X > XK \geq B = BK = 0$

k.

00	02	39	7D
	02	30	2D

←X→ BK ←
←—— B —→

$X = XK \geq 0$

$B > BK \geq 0$

DECIMAL COMPARE TIMING PROCEDURE FOR CASE 3

Step	For Case Formats	Conditions (Times in Microseconds)	Times (Microseconds)
1	f, g, h, j, k	Start with basic time of 14.01 and go to Step 2.	14.01
2	f, g, h, j, k	Add 0.13 if the destination comparand sign is positive and go to Step 3.	0.13
3	g, h, j, k	Add 0.03 if SL > DL and go to Step 4.	0.03
4	g, h, j, k	Add 1.17 for each leading zero character in the extension field and go to Step 5.	1.17XK
5	j	Add 1.98 and STOP (Note 3) if there is a non-zero character in the extension field; otherwise go to Step 6.	+1.98
6	f, g	Add 0.42 and STOP (Note 4) if B = 0 (i.e. no body) and if either the extension field is all zero ([X] = 0) or there is no extension field (X = 0); otherwise, go to Step 7.	+0.42
7	h, k	Add 2.40 for each corresponding equal pair of leading characters in the body field and go to Step 8.	2.40BK
8	k	Add 4.34 and STOP (Note 5) for the first corresponding un-equal character pair in the body field; otherwise go to Step 9.	+4.34
9	h	Add 0.81 and STOP (Note 4) if all corresponding characters in the body fields are equal.	+0.81

Note 3 - Not equal: compare based on sign and which field is longer.
Note 4 - Compare based on units and signs only
Note 5 - Not equal: compare based on sign and un-equal body characters.

Table A-7. SUE 1112A/B Instruction Times

Instruction		Execution Time in Microseconds	
Bit Manipulation		Selected by	
		XR or K	Mask w/wo XR
RBIT	Make the designated bit a zero	3.24	3.30
SBIT	Make the designated bit a one	3.24	3.30
CBIT	Change (complement) the designated bit	3.24	3.30
IBIT	Isolate (extract) the designated bit	3.24	3.30
TSBT	Test the designated bit and shift left	3.27	3.33
TBIT	Only test the designated bit	3.24	3.30
Move		Operand Value	
		Positive	Negative
NEGT	Move the twos complement value	2.49	2.49
CPLM	Move the ones complement value	2.49	2.49
MOVP	Move the positive magnitude	2.75	2.88
MOVN	Move the negative magnitude	2.75	2.88
Normalize and Count			
SxxN	Single Normalize, AC = 0	2.82	
SLAN	Single Left Arithmetic Normalize	3.24 + 0.32 per shift	
SLLN	Single Left Logical Normalize	2.95 + 0.29 per shift	
SRAN	Single Right Arithmetic Normalize	2.95 + 0.29 per shift	
SRLN	Single Right Logical Normalize	2.95 + 0.29 per shift	
DxxN	Double Normalize, both AC's = 0	3.89	
DLAN	Double Left Arithmetic Normalize	5.21 + 0.61 per shift	
DLLN	Double Left Logical Normalize	4.89 + 0.48 per shift	
DRAN	Double Right Arithmetic Normalize	5.18 + 0.97 per shift	
DRLN	Double Right Logical Normalize	4.31 + 0.81 per shift	

Table A-7. SUE 1112A/B Instruction Times (continued)

Instruction		Execution Time in Microseconds
Double-Length Shift		
DLAO	Double Left Arithmetic Open With Zero Count	5.08 + 0.61 per shift 3.95
DLLL	Double Left Logical Linked With Zero Count	4.50 + 0.32 per shift 3.89
DLLO	Double Left Logical Open	3.95 + 0.48 per shift
DLLC	Double Left Logical Closed	4.11 + 0.64 per shift
DRAO	Double Right Arithmetic Open	4.11 + 0.97 per shift
DRLL	Double Right Logical Linked	3.89 + 0.81 per shift
DRLO	Double Right Logical Open	3.50 + 0.81 per shift
DRLC	Double Right Logical Closed	3.76 + 1.07 per shift

Table A-8. SUE 1112A/B Single- and Double-Precision
Fixed-Point Instruction Times

	Store	Load	Add	Sub	Multiply (See notes)		Divide (See notes)
	DSTA	DLOD	DADD	DSUB	MLTA	MULT	DIVD
Register Address (Indexed)	5.06	4.44	5.96	5.96	≈16.87	≈15.35	≈15.29
Extended (w/wo Indexing)	5.13	4.54	6.06	6.06	≈16.97	≈15.45	≈15.39
Register Address, Indirect*	7.04	6.45	7.97	7.97	≈18.88	≈17.36	≈17.30
Extended, Indirect*	6.72	6.13	7.65	7.65	≈18.56	≈17.04	≈16.98

Notes:

*Add 1.01 for each additional level of indirect

DIVIDE

1. Assumes divisor is positive and quotient is positive and even

2. Time is 4.10 if divide check occurs

3. Add 0.29 if divisor is negative

4. Subtract 0.06 if quotient is negative

5. Subtract 0.13 if quotient is odd

6. Total range (except divide check) for extended direct divide is 15.68 to 15.20

MULTIPLY

7. Assumes typically seven 'one' bits in the absolute value of the multiplier; if more (or less) add (or subtract) $N \times 0.03$ to the time, where N is the additional number of significant multiplier bits.

8. Add 0.19 if the product is negative

9. Total range for extended direct multiply (w/o accumulate) is 15.24 to 15.88

APPENDIX B

INSTRUCTION SUMMARY AND INDEX

SUE 1110 (BASIC) INSTRUCTIONS SUMMARY

Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4	L3	L2	L1	A	N	Z	O	LP	F3	F2	F1	C	V	G	E

Control Instructions — Class O

15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	MNEMONIC
0	0	x x x x	x x x x	HALT
0	2	0 F3 F2 F1	C V G E	RSTS
0	2	1 F3 F2 F1	C V G E	SETS
0	8	0	L4 L3 L2 L1	ENBL
0	8	4	L4 L3 L2 L1	ENBW
0	8	8	L4 L3 L2 L1	DSBL
0	8	C	L4 L3 L2 L1	DSBW
0	B 1		D	STSM
0	B 3		D	REGM
0	B 4		D	RETN
0	B 5		D	MSTS
0	B 7		D	MREG

B — 0 if Absolute, 1 if Relative.
D — Address field (words), two's complement form for Relative.

General Register Instructions — Classes 1-7

		15 14 13 12	11	10 9 8	7	6 5 4	3	2 1 0
Accumulator To Memory	⌐ Auto Decrement	1	B	OP	I	AR	E	XR
	⌐ Auto Increment	2	B	OP	I	AR	E	XR
		3	B	OP	I	AR	E	XR
	Jump	4	0	0	I	0	E	XR
	Jump to Subroutine	4	0	0	I	AR	E	XR
	Data to	4	1	OP	0	AR	E	XR
	Accumulator	4	1	OP	1	AR	K
Memory To Accumulator	⌐ Auto Decrement	5	B	OP	I	AR	E	XR
	⌐ Auto Increment	6	B	OP	I	AR	E	XR
		7	B	OP	I	AR	E	XR

Sequence: −, E, XR, I, B, +.

OP — Operation Code:

0	MOV	Move
1	SUB	Subtraction
2	ADD	Addition
3	AND	Logical Product
4	IOR	Logical Inclusive Or
5	EOR	Logical Exclusive Or
6	CMP	Compare
7	TST	Test

B   Word mode if 0, byte mode if 1.
I   Indirect when 1.
AR   Accumulator Register designator (0-7).
E   Extended or two word instruction if 1.
XR   Index Register designator (1-7), no indexing if 0.
K   4 bit literal constant data.

Branch Instructions — Classes 8-9

	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
No Operation	8	0	x x x x x x x x
Branch Unconditional	9	0	D
Branch False	8	T	D
Branch True	9	T	D

D — Displacement word address in two's complement form.

T	INDICATOR	MNEMONIC
1	Equal	BEQx
2	Greater Than	BGTx
3	Overflow	BOVx
4	Carry	BCYx
5	Flag 1	BF1x
6	Flag 2	BF2x
7	Flag 3	BF3x
8	Loop Complete	BLPx
9	Odd	BODx
A	Zero	BZEx
B	Negative	BNGx
C	Less Than	BLTx
D, E, F,	Unimplemented	

X = T or F

Shift Instructions — Class A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A				B	L/R	OP		0		AR		X		XR	
A				B	L/R	OP		1		AR			K		

B   0 = single length, 1 = double length (not implemented on 1110).
AR   Accumulator Register designator (to be shifted).
K   Shift Count
XR   Shift Count Source Register.
L/R   Left (L) when 0, Right (R) when 1.

OP   Shift Operation Code: 00 = Arithmetic
                 01 = Logical Linked
                 10 = Logical Open
                 11 = Logical Closed

SUE 1110 (BASIC) INSTRUCTION INDEX

D.   SHIFT INSTRUCTIONS

E.   CONTROL INSTRUCTIONS

SUE 1112A INSTRUCTIONS

Includes all instructions listed under SUE 1110 (Basic), 1110A,
and the following:

A.   BIT MANIPULATION INSTRUCTIONS

B.   MOVE INSTRUCTIONS

C.   NORMALIZE AND COUNT INSTRUCTIONS

D.   DOUBLE LENGTH SHIFT

E.   SINGLE PRECISION FIXED POINT

F.   DOUBLE PRECISION FIXED POINT

G.   CONTROL INSTRUCTIONS

SUE 1112B INSTRUCTIONS

Includes all instructions listed under SUE 1110 (Basic), 1110A,
1112A, and the following:

APPENDIX C

INPUT/OUTPUT ADDRESSES

Table C-1. Input-Output Device Addresses*

Address (Hex)	Input/Output Device Controller
F800	Teletypewriter No. 1
F810	Teletypewriter No. 2
F820	High Speed Paper Tape Reader No. 1
F830	High Speed Paper Tape Punch No. 1
F840	High Speed Paper Tape Reader No. 2
F850	High Speed Paper Tape Punch No. 2
F860	Card Reader No. 1
F870	Card Reader No. 2
F880	Card Punch No. 1
F890	Card Punch No. 2
F8A0	Line Printer No. 1
F8B0	Line Printer No. 2
F8C0	Magnetic Tape No. 1 (handles 4 Drives)
F8D0	Magnetic Tape No. 2 (handles 4 Drives)
F8E0	Bulk File No. 1 (Fixed Head)
F8F0	Bulk File No. 2 (Fixed Head)
F900	Disk File Unit No. 1 (Fixed and Removable)
F910	Disk File Unit No. 2 (Fixed and Removable)
F920	Cassette No. 1
F930	Cassette No. 2
FA00	CRT Display, Alphanumeric No. 1
FA10	CRT Display, Alphanumeric No. 2
⋮	⋮
FAF0	CRT Display, Alphanumeric No. 16
FF90	Input Keyboard No. 1, Business System
FFA0	CRT Display No. 1, Business System

*Note: Device address assignment is variable by jumper wires connected on each controller. The addresses shown are recommended and are subject to change.

APPENDIX D

SELF-INTERRUPT AND SYSTEM INTERRUPT
EXECUTIVE SPACE

	ADDRESS THAT CAUSED ABORT	STATUS	ABORTED INSTRUCTION ADDRESS	SERVICE ROUTINE VECTOR
6	ADDR00 0028 ADDR10 0048 ADDR01 0038 ADDR11 0058	ADDR00 002A ADDR10 004A ADDR01 003A ADDR11 005A	ADDR00 002C ADDR10 004C ADDR01 003C ADDR11 005C	ADDR00 002E ADDR10 004E ADDR01 003E ADDR11 005E
5	UNIMPLEMENTED INSTRUCTION ADDR00 0020 ADDR10 0040 ADDR01 0030 ADDR11 0050	STATUS ADDR00 0022 ADDR10 0042 ADDR01 0032 ADDR11 0052	UNIMPLEMENTED INSTRUCTION ADDRESS ADDR00 0024 ADDR10 0044 ADDR01 0034 ADDR11 0054	SERVICE ROUTINE VECTOR ADDR00 0026 ADDR10 0046 ADDR01 0036 ADDR11 0056
4	MODULE ADDRESS 0018	STATUS 001A	PROGRAM COUNTER 001C	SERVICE ROUTINE VECTOR 001E
3	MODULE ADDRESS 0010	STATUS 0012	PROGRAM COUNTER 0014	SERVICE ROUTINE VECTOR 0016
2	MODULE ADDRESS 0008	STATUS 000A	PROGRAM COUNTER 000C	SERVICE ROUTINE VECTOR 000E
1	MODULE ADDRESS 0000	STATUS 0002	PROGRAM COUNTER 0004	SERVICE ROUTINE VECTOR 0006

SELF INTERRUPT: rows 6, 5

SYSTEM INTERRUPT: rows 4, 3, 2, 1

1110-R03-72

APPENDIX E

USASCII CHARACTER SET AND HEXADECIMAL CODES

HEX	CHARACTER	HEX	CHARACTER
A0	space	C1	A
A1	!	C2	B
A2	"	C3	C
A3	#	C4	D
A4	$	C5	E
A5	%	C6	F
A6	&	C7	G
A7	' (apostrophe)	C8	H
A8	(	C9	I
A9	)	CA	J
AA	*	CB	K
AB	+	CC	L
AC	, (comma)	CD	M
AD	-	CE	N
AE	. (period)	CF	O
AF	/	D0	P
B0	0	D1	Q
B1	1	D2	R
B2	2	D3	S
B3	3	D4	T
B4	4	D5	U
B5	5	D6	V
B6	6	D7	W
B7	7	D8	X
B8	8	D9	Y
B9	9	DA	Z
BA	:	DB	[left bracket
BB	;	DC	\ back slash
BC	< less than	DD	] right bracket
BD	=	DE	↑ up arrow
BE	> greater than	DF	← left arrow
BF	?		
C0	@	87	bell
		8A	line feed
		8D	carriage return