E. F. Schwartz

SDS
SCIENTIFIC DATA SYSTEMS

# 920 Computer reference manual

# SDS 920 Computer
# reference manual

SDS
SCIENTIFIC DATA SYSTEMS


SDS

# Contents

# Figures

# I. General Description

## A. INTRODUCTION

The SDS 920 is a high-speed, low-cost, general-purpose digital computer with the following characteristics:

24-bit word plus parity bit

Binary arithmetic

Single address instructions with:

- Index Register
- Indirect Addressing
- Programmed Operators

Basic Core memory 4,096 words expandable to 16,384 words, all addressable

Built-in floating point capability

Multi-precision instructions

Typical execution times (including memory access and indexing):

| | |
|---|---|
| Add: | 16 microseconds |
| Multiply: | 128 microseconds |
| Floating Point Operations: | |
| (24-bit Mantissa plus 9-bit Exponent) | |
| Add: | 192 microseconds |
| Multiply: | 280 microseconds |
| (39-bit Mantissa plus 9-bit Exponent): | |
| Add: | 368 microseconds |
| Multiply: | 560 microseconds |

Program interchangeability with other SDS 900 Series computers

Parity checking of all memory and input/output operations

1024 channels of Priority Interrupt (optional)

Memory non-volatile with power failure

Input/Output:

Standard:

300 character/second Paper Tape Reader
60 character/second Paper Tape Punch
Automatic Typewriter
Dual-channel Priority Interrupt
Display and manual control of internal registers

Optional:

Magnetic Tape Systems (IBM compatible)
Line Printer
Punched Card Equipment
Direct communication with IBM 7090, A/D converters, etc.

Buffered input/output at rates in excess of 60,000 characters/second simultaneous with computation.

FORTRAN II and Symbolic Assembler as part of complete software package

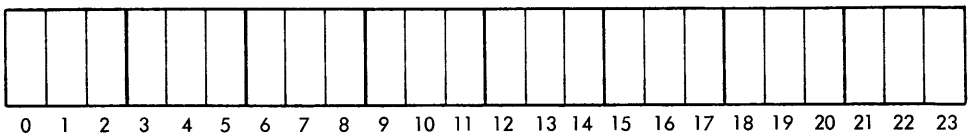
All silicon semiconductors

$0^{\circ}$ to $55^{\circ}$C operating temperature range
Dimensions: 66" x 48" x 27"
Power: 110V, 60 cps, 10 amps

## B. MACHINE ORGANIZATION

### 1. Word Format

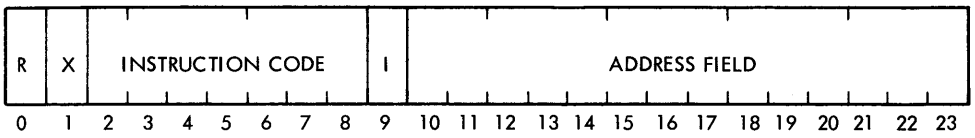
The SDS 920 computer word contains 24 binary digits (bits), numbered from 0 through 23 starting at the left.




Arithmetic is performed using a binary, two's complement number system. The sign bit, then, can be considered to be integral with the data word. For simplicity of description, SDS 920 words are written in octal notation where each octal digit represents 3 binary digits. For example, an SDS 920 data word containing alternate "ones" and "zeros" would be written in the form:

52525252

The instruction word format of the SDS 920 is as follows:




| Bit Number | Function |
|---|---|
| 0 | Relative Address Bit - A "1" in this position causes the location of the instruction to be added to the address at loading. This bit is not used in the logic of the computer. |
| 1 | Index Register Bit - A "1" in this position causes the contents of bits 10 - 23 of the Index Register (X Register) to be added to the address portion of the instruction prior to execution. |
| 2 - 8 | Instruction Code - The contents of bit 2 (Programmed Operator Bit) determine the method of interpretation of the remaining 6 bits. If bit 2 contains an "0", the contents of bits 3 through 8 are decoded as a normal instruction (see Section II). If bit 2 contains a "1", the instruction code is used to determine a subroutine entrance address. |
| 9 | Indirect Address Bit - A "1" causes the computer to interpret bits 10 - 23 of the instruction (possibly modified by indexing) as the memory location where the effective address of the instruction may be found. A "0" causes bits 10 - 23 (possibly modified by indexing) to be interpreted as the effective address of the instruction. |
| 10 - 23 | Address - These bits normally determine the memory address referenced by the Instruction Code. |

For simplicity, SDS 920 instructions are written in the following manner:

| Field (Bit Positions) | | Remarks |
|---|---|---|
| Index (0 - 1) | 0 (or Blank) | No Index, No Relative Address |
| | 1 | Index |
| | 2 | Relative Address |
| | 3 | Both Relative Address and Index |

| Field (Bit Positions) | Remarks |
|---|---|
| Instruction (2 - 8) | Mnemonic (3 letters) or<br>Numeric (3 octal digits) |
| Indirect Address (9) | I = Indirect Address<br>Blank = Direct Address |
| Address (10 - 23) | Blank or<br>1 - 5 Octal Digits (0-37777) or<br>Plus (+) or Minus (-) sign followed by 1 - 5 octal digits of relative address |

For example, the instruction to add to the A Register the contents of memory location defined by the sum of 3742 and the contents of the Index Register is

1 ADD 3742

## 2. Registers

The SDS 920 contains nine arithmetic and control registers. All are full-word registers except as noted:

a. Available to the programmer:

A Register - The A Register is the main accumulator of the SDS 920.

B Register - The B Register is an extension of the A Register. It contains the least significant portion of double length numbers.

X Register - The X Register is the Index Register used in address modification. The number of positions shifted during normalizing are also counted in the X Register. Although the X Register is a full word register and is transferred to and from memory as such, only the least significant 15 bits are used for indexing operations.

P Register - The P Register (14 bits) contains the memory address of the current instruction. Unless modified it is increased by one at the completion of each instruction.

W Register - The W Register is the input/output register of the SDS 920. Input data is automatically assembled in this register and is transferred to memory under program control.

Y Register - (optional) Identical with the W Register. Permits multiple input/output processing.

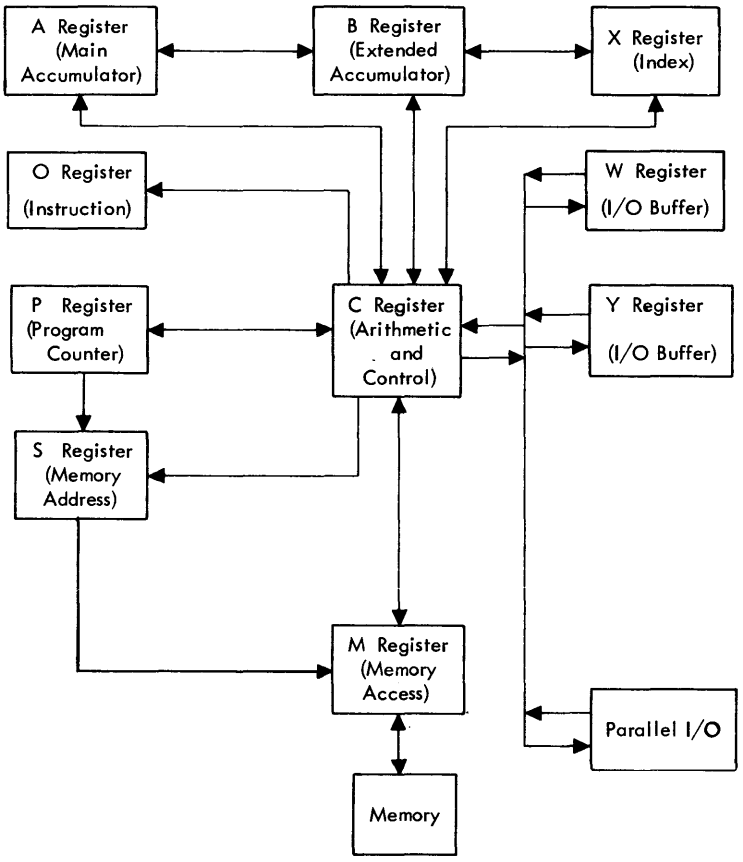



Figure 1-1 Functional Diagram - SDS 920

b. Not available to the programmer:

C Register - The C Register is an arithmetic and control register used in multiply, divide, and other operations. All instructions brought from memory first appear in the C Register before decoding. Address modification and parity generation/detection take place in the C Register.

S Register - The S Register (14 bits) contains the address of the memory location to be accessed (instruction or data).

O Register - The O Register (7 bits) contains the instruction code of the instruction being executed.

M Register - The M Register holds each word as it is brought out of memory. Recopying of the word into memory takes place from the M Register.

## 3. Memory Organization

The main memory of the SDS 920 is composed of from 1 to 4 random access magnetic core modules of 4,096 words each. The maximum memory contains 16,384 directly addressable words. Each word contains 24 data bits plus 1 parity bit. The memory words are identified in octal by addresses 00000 through 37777. The basic computer contains 4,096 memory words identified as locations 00000 through 07777.

Special logic has been included in the SDS 920 to prevent changes in memory from occurring due to power failure (total or transient) or when power is manually shut off. Before each memory word is accessed, power is checked to ensure that the entire read-write cycle can be successfully completed. If the voltage has dropped to the point where success is not assured, the computer will halt.

Even parity is generated for all words stored in memory, i.e., the parity bit is set either to a "0" or "1" so that the sum of "1's" in the 25 bit word is even. All memory words are parity checked when they are brought from memory. A switch on the control planel may be set so that the computer halts automatically if a parity error occurs.

## 4. Overflow

An overflow indicator in the computer permits the detection of arithmetic errors in the program. The overflow indicator is turned on if any of the following occur:

a. Addition or subtraction, resulting in a sum or difference, which cannot be contained within the A Register.

b. Multiplication of N by N where N is the largest negative number that can be represented in an SDS 920 word (40000000). This product cannot be contained within the A and B registers.

c. A division operation where the absolute value of the numerator is equal to or larger than the absolute value of the denominator. The quotient cannot be contained within the A Register.

d. A left shift operation which shifts a bit of absolute magnitude equal to one beyond position 1 of the A Register.

The status of the overflow indicator can be tested using SKIP IF SIGNAL NOT SET (40). When this instruction is executed, the overflow indicator is turned off. The overflow indicator can be turned off, without testing, using ENERGIZE OUTPUT M (02).

Overflow indication is cumulative. If the Overflow Indicator is turned on, it remains on until turned off by the appropriate instruction.

The status of the Overflow Indicator is automatically preserved when subroutines are executed (Programmed Operator, Closed or Interrupt). When a Programmed Operator instruction is executed, the status of the Overflow Indicator is automatically preserved in bit 0 of memory location 00 and the Overflow Indicator is turned off. When MARK PLACE AND BRANCH (43) is executed (during Interrupt or when starting a closed subroutine) the status of the Overflow Indicator is preserved in bit 0 of the location determined by the effective address and the Overflow Indicator is unchanged.

RETURN BRANCH (51) automatically performs a logical "or" of the contents of the Overflow Indicator and the contents of bit 0 of the location determined by the effective address.

## C. SPECIAL CHARACTERISTICS

Certain features have been included in the SDS 920 which simplify programming and provide significant economies in program running time and memory.

### 1. Address Modification

The SDS 920 contains two techniques for the modification of instruction addresses. These techniques, Indexing and Indirect Addressing, may be used singly or in combination to achieve highly flexible program operation. In both, the address is modified after the instruction has been called from memory. The instruction is preserved in memory in its original form.

a. Indexing

The SDS 920 contains an Index Register for instruction modification. If the Index Bit (bit 1) in the instruction is set equal to 1, the contents of bits 10 - 23 of the X Register are added to the address portion of the instruction prior to execution. Instructions are provided for modification and testing of the X Register, transferring information between the X and B Registers, the X and A Registers, and between the X Register and memory.

For example, if the X Register contains the value + 5 and the instruction

1 ADD 200

is executed, the effective address of the instruction is 205.

b. Indirect Addressing

If the instruction contains a "1" in the Indirect Address Bit (bit 9) the address portion of the instruction (possibly modified by indexing) is used to determine the location in memory which contains the effective address of the instruction. Since this technique allows the modification of many instructions using only one constant, it is extremely useful for certain types of indexing operations, table organization and look-up, and program organization.

The process of indirect addressing may be iterative since the memory location containing the indirect address may in turn contain a "1" in the Indirect Address Bit (bit 9). In this case, the computer will repeat the procedure and a third address is located. There is no limit to the number of times this process may be repeated.

Addresses are modified by Indexing before the associated Indirect Address bit is interrogated. Indirect address can be modified by Indexing at any level.

2. Programmed Operator

The use of Programmed Operators enables sub-routines to be referenced with a single instruction of the same form as built-in machine instructions. While instruction codes 00-77 are decoded in a normal manner, codes 100-177 cause the computer to enter a sub-routine uniquely determined by the code. The return address is automatically recorded in memory at location 0000 so that program continuity is maintained. Through the use of indirect addressing, the subroutine can gain access to the effective address of the calling instruction.

Programmed Operator sub-routines may be assigned three letter mnemonic designations in a manner identical with built-in machine instructions described in Section II. These designations are converted

to instruction codes (100-177) when the program is loaded. Subroutine input and connection is also provided by the loading system.

Through use of Programmed Operators, symbolic homogeneity is maintained with other computers in the SDS 900 series. Mnemonic designations are identical in all computers. For example, while the designation "FLA" (for Floating Add) may refer to a built-in machine instruction in one computer, it references a Programmed Operator subroutine in another. This technique preserves the one-to-one instruction relationship and, therefore, programs wirtten for one 900 series computer may be run on any other computer in the series.

A description of the use of Programmed Operators and a list of Programmed Operator instructions supplied with the computer are detailed in Appendix C.

# II. Machine Instructions

## A. INTRODUCTION

A description of all SDS 920 instructions is contained in this section. Instructions are grouped by functional category. Tables of Instructions in numerical and functional order are given in Appendices A and B, respectively.

The following statements apply to the instruction descriptions:

1. All instructions times are in memory cycles where 1 cycle equals 8 microseconds, and include accessing of the instruction. 125KC
2. Octal notation is used throughout.
3. Parentheses are used to denote "contents of." For example, "(A)" denotes "contents of the A Register."
4. Subscripted numbers identify bit positions. For example, "$X_{0-14}$" means "bit positions 0 through 14 of the X Register."
5. Indexing and Indirect Addressing apply to all instructions except as noted. Indexing does not change the instruction execution time. One additional memory cycle is required for each indirect address used.
6. The term "effective address" applies to the instruction address after all modifications (Indexing and Indirect Address) have been completed.
7. Examples are included where clarification is necessary. These have been chosen in order to fully illustrate the instruction rather than for practical importance of the example.

## B. LOAD-STORE INSTRUCTIONS

(76) LOAD A LDA

The contents of the memory location determined by the effective address are loaded into the A Register.

Registers Affected: A Timing: 2

(35) STORE A STA

The contents of the A Register are stored in the memory location determined by the effective address.

Registers Affected: M Timing: 3

(75) LOAD B LDB

The contents of the memory location determined by the effective address are loaded into the B Register.

Registers Affected: B Timing: 2

(36) STORE B STB

The contents of the B Register are stored in the memory location determined by the effective address.

Registers Affected: M Timing: 3

(71) LOAD INDEX LDX

The entire contents (24 bits) of the memory location determined by the effective address are copied into the Index Register.

Registers Affected: X Timing: 2

(34) STORE INDEX STX

The contents of the Index Register are stored in the memory location determined by the effective address. All 24 bits of the Index Register are stored.

Registers Affected: M Timing: 3

(62) EXCHANGE M & A XMA

The contents of the memory location determined by the effective address are loaded into the A Register and the A Register stored in memory.

Registers Affected: A, M Timing: 3

## C. ARITHMETIC INSTRUCTIONS

(55) ADD M TO A ADD

The contents of the memory location determined by the effective address are added to the A Register, and the result appears in A.

Overflow occurs and the overflow indicator is turned on, if both numbers are of the same sign, and the sign of the result is opposite. In this case the sum is incorrect.

This instruction is also used in multi-precision addition [see ADD WITH CARRY (57)]

Registers Affected: A, $X_0$ Timing: 2

(57) ADD WITH CARRY ADC

This instruction is used to perform double precision operations. The lower half of the numbers are added first, using ADD M TO A (55). The carry is automatically retained in the sign position of the X Register. The two upper halves are then added, using this instruction, which is the same as ADD M TO A (55), except that the carry bit previously generated is also added.

Overflow resulting from the addition of the lower half of the numbers is not meaningful and the Overflow Indicator is turned off prior to execution of this instruction.

Overflow can occur as in ADD M TO A (55).

Example:

Assume the A and B Registers contain a double precision number to which the double precision number in Locations 1020 and 1021 is to be added. The least significant half appears in 1020. The program is as follows:

| Location | Instruction | A | B | Carry | 1021 | 1020 |
|---|---|---|---|---|---|---|
| Prior to Execution | | 20314624 | 71510426 | - | 15034166 | 12300000 |
| 2100 | XAB | 71510426 | 20314624 | - | 15034166 | 12300000 |
| 2101 | ADD 1020 | 04010426 | 20314624 | 1 | 15034166 | 12300000 |
| 2102 | XAB | 20314624 | 04010426 | 1 | 15034166 | 12300000 |
| 2103 | ADC 1021 | 35351013 | 04010426 | 0 | 15034166 | 12300000 |

NOTE: Since the process is self-propagating, this instruction may be used in performing additions of any precision.

Registers Affected: A, $X_0$ Timing: 2

(63) ADD A TO M ADM

The contents of the A Register are added to the memory location determined by the effective address, and the result stored in the same memory location.

Overflow can occur in the same manner as ADD M TO A (55).

Registers Affected: M Timing: 3

(61) MEMORY INCREMENT MIN

The contents of the memory location determined by the effective address are increased by one, and the resultant sum replaced in the same memory location. Overflow can occur in the same manner as ADD M TO A (55).

Registers Affected: M Timing: 3

(54) SUBTRACT M FROM A SUB

The contents of the memory location determined by the effective address are subtracted from the A Register and the result appears in A.

Overflow occurs and the overflow indicator is turned on, if both numbers are of different signs and the sign of the result does not agree with the original sign of A. In this case the difference is incorrect.

This instruction is also used in multi-precision subtraction [see SUBTRACT WITH CARRY (56) ]

Registers Affected: A, $X_0$ Timing: 2

(56) SUBTRACT WITH CARRY SUC

This instruction is used to perform double precision operations. The lower half of the numbers are subtracted first, using SUBTRACT M FROM A (54). The carry is automatically retained in the sign position of the X Register. The two upper halves are then subtracted using this instruction, which is the same as SUBTRACT M FROM A (54) except that the carry bit previously generated is used.

Overflow resulting from the subtraction of the lower half of the numbers is not meaningful and the Overflow Indicator is turned off prior to execution of this instruction.

Overflow can occur as in SUBTRACT M FROM A (54).

Example:

Assume the A and B Registers contain a double precision number from which the double precision number in Locations 1075 and 1076 is to be subtracted. The least significant half appears in 1075. The program is as follows:

| Location | Instruction | A | B | Carry | 1021 | 1020 |
|---|---|---|---|---|---|---|
| Prior to Execution | | 36142070 | 31567000 | - | 14236213 | 26120000 |
| 2100 | XAB | 31567000 | 36142070 | - | 14236213 | 26120000 |
| 2101 | SUB 1075 | 03447000 | 36142070 | 1 | 14236213 | 26120000 |
| 2102 | XAB | 36142070 | 03447000 | 1 | 14236213 | 26120000 |
| 2103 | SUC 1076 | 21703655 | 03447000 | 0 | 14236213 | 26120000 |

NOTE: Since the process is self-propagating, this instruction may be used in performing subtractions of any precision.

Registers Affected: A, $X_0$ Timing: 2

(64) MULTIPLY MUL

The A Register is multiplied by the contents of the memory location determined by the effective address. The product is located in the A and B Registers; the most significant portion is in A. The sign of the product is in $A_0$; $B_0$ is part of the product and is not treated as a sign bit. Since the multiplier and multiplicand are treated as fractional quantities, the contents of $B_{23}$ are not meaningful. The original contents of B do not affect the operation and are destroyed. If the contents of both the multiplier and multiplicand have the value 40000000, overflow will occur and the overflow indicator is turned on.

Example:

Multiplying 3 by 3 is represented as follows:

| | A | B | Memory |
|---|---|---|---|
| Before Execution | 00000003 | Not Meaningful | 00000003 |
| After Execution | 00000000 | 00000022 | 00000003 |

Registers Affected: A, B Timing: 16

(65) DIVIDE DIV

The contents of the A and B Registers, treated as a double precision number, are divided by the contents of the memory location determined by the effective address. The quotient appears in the A Register, the remainder in B. The sign of the remainder is identical with the sign of the original contents of the A Register.

Overflow will occur and the overflow indicator turned on, if $1 \leq \frac{(AB)}{M} < -1$

In this case, the results are not arithmetically correct.

Example:

Dividing 7 by 3 is represented as follows:

| | A | B | Memory |
|---|---|---|---|
| Before Execution | 00000000 | 00000016 | 00000003 |
| After Execution | 00000002 | 00000002 | 00000003 |

Registers Affected: A, B Timing: 28

## D. BRANCH-SKIP INSTRUCTIONS

(01) BRANCH UNCONDITIONALLY BRU

The next instruction is taken from the location determined by the effective address.

Registers Affected: None Timing: 1

(41) INCREMENT INDEX & BRANCH BRX

The contents of the Index Register are incremented by one. If the resultant Index Register value contains a "1" in bit 9 of the index (i.e., the address bits are negative) control is transferred to the location

determined by the effective address. If the Index Register is positive, the next instruction is taken in sequence. The most significant bits of the Index Register (bits 0 - 8) have no effect on the execution of this instruction.

Registers Affected: X

Timing: 1, if branch
2, if no branch

(43) MARK PLACE AND BRANCH BRM

The contents of the P Register are stored in the memory location determined by the effective address, and control is transferred to the subsequent memory location. The status of the Overflow Indicator is stored in bit 0 of M. Bits 1 - 9 of M are set to zero.

Example:

| Location | Instruction |
|---|---|
| 01517 | BRM 522 |

| | Overflow Indicator | Location 0522 | P Register |
|---|---|---|---|
| Before Execution | 1 (on) | --- | 01517 |
| After Execution | 1 (on) | 40001517 | 00523 |

Note: This instruction is normally used to enter subroutines where a return to the main program will occur after the subroutine has been completed. RETURN BRANCH (51) is used to return to the main program.

Registers Affected: M

Timing: 2

(51) RETURN BRANCH BRR

The contents of the memory location determined by the effective address are incremented by one and the least significant 14 bits stored in the P Register. A logical "or" is performed between bit 0 and the Overflow Indicator and the result placed in the Overflow Indicator.

Example:

| Location | Contents |
|---|---|
| 2100 | BRR 2000 |
| 2000 | 00003220 |

If the instruction in location 2100 is executed, the next instruction will be taken from location 3221.

Note: This instruction is normally used to return to the main program after completion of a subroutine in conjunction with MARK PLACE AND BRANCH (43).

Registers Affected: None

Timing: 2

(40) SKIP IF SIGNAL NOT SET SKS

This instruction determines the status of one-bit (on/off) signals, the identity of the signals being determined by the effective address. If the instruction is in location L and the signal is set (i.e. = 1), the next instruction will be taken from L + 1. If the signal is not set (i.e. = 0), the next instruction will be taken from L + 2.

Appendix D contains a full description of the signals which may be sensed by this instruction.

Indirect addressing and indexing do not apply to this instruction.

Registers Affected: None

Timing: 1, if no skip
2, if skip

(50) SKIP IF A EQUALS M SKE

The contents of the A Register are compared with the contents of the memory location determined by the effective address. If the instruction is in location L, and (A) $\neq$ (M), the next instruction will be taken from L + 1. If (A) = (M), the next instruction will be taken from L + 2.

Registers Affected: None

Timing: 2, if no skip
3, if skip

(73) SKIP IF A GREATER THAN M SKG

The contents of the A Register are algebraically compared with the contents of the memory location determined by the effective address. If the instruction is in location L, and (A) $\leq$ (M), the next instruction is taken from L + 1. If (A) > (M), the next instruction is taken from L + 2.

Registers Affected: None

Timing: 2, if no skip
3, if skip

(60) REDUCE M, SKIP IF NEGATIVE SKR

The contents of the memory location determined by the effective address are reduced by one and the result replaced in the same memory location. If the instruction is in Location L, and the result is positive, the next instruction is taken from L + 1. If the result is negative, the next instruction is taken from L + 2.

Overflow can occur in the same manner as in SUBTRACT M FROM A (54).

Registers Affected: M Timing: 3

(70) SKIP IF A = M ON B MASK SKM

The contents of the A Register are compared with the contents of the memory location, determined by the effective address, wherever a "1" appears in the corresponding position of the B Register. If the instruction is in Location L, and the selected contents of A and M do not agree, the next instruction is taken from L + 1. If A and M agree, the next instruction is taken from L + 2.

Example:

| A | B | Memory |
|---|---|---|
| 00043007 | 00077000 | 57643240 |

Since a comparison is made in bit positions 9 - 14 only (as determined by B), comparison is found and the skip will occur.

Note: This instruction can be used to search through tabular sets of compressed information where comparison is required in certain bit positions. INCREMENT INDEX AND BRANCH (41) is normally used with this instruction to provide a 3 cycle search loop.

Registers Affected: None Timing: 2, if no skip
3, if skip

(53) SKIP IF M NEGATIVE SKN

If the instruction is in Location L and the contents of the memory location determined by the effective address are positive, the next instruction is taken from L + 1. If the contents are negative, the next instruction is taken from L + 2.

Registers Affected: None Timing: 2, if no skip
3, if skip

(72) SKIP IF M AND A DO NOT COMPARE ONES SKA

The contents of the A Register are compared, bit by bit, with the contents of the memory location determined by the effective address. If the instruction is in Location L and both (A) and (M) have corresponding "1's" in any bit position, the next instruction is taken from L + 1. If A and Memory have no corresponding "1's" in any bit position, the next instruction is taken from L + 2.

The instruction logically "ands" corresponding bits in A and Memory, based on the following table:

| A | Memory | Result |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

If the result produces a "1" in any bit position, a skip will not occur.

Note: Different configurations of the memory word result in a wide variety of conditional instructions for use by the programmer. Some representative ones are:

| Memory Configuration | Instruction |
|---|---|
| 40000000 | Skip if A is Positive |
| 77777777 | Skip if A = 0 |
| 00000001 | Skip if A is Even |

Registers Affected: None

Timing: 2, if no skip
3, if skip

(52) SKIP IF M AND B DO NOT COMPARE ONES SKB

The contents of the B Register are compared, bit by bit, with the contents of the memory location determined by the effective address. If the instruction is in Location L and both B and Memory have corresponding "1's" in any bit position, the next instruction is taken from L + 1. If B and Memory have no corresponding "1's" in any bit position, the next instruction is taken from L + 2.

The instruction logically "ands" corresponding bits in A and Memory, based on the following table:

| B | Memory | Result |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

If the result produces a "1" in any bit position, a skip will not occur.

Note: Different configurations of the memory word result in a wide variety of conditional instructions for use by the programmer. Some representative ones are:

| Memory Configuration | Instruction |
|---|---|
| 40000000 | Skip if B is Positive |
| 77777777 | Skip if B = 0 |
| 00000001 | Skip if B is Even |

Registers Affected: None

Timing: 2, if no skip
3, if skip

(74) DIFFERENCE EXPONENTS AND SKIP SKD

The contents of bits 15 - 23 of the memory location determined by the effective address are subtracted from bits 15 - 23 of the B Register. The absolute magnitude of the difference is stored in bits 15 - 23 of the X Register. If the instruction is in Location L and the difference is positive [ i.e. $(M_{15-23}) \leq (B_{15-23})$ ] the next instruction is taken from L + 1. If the difference is negative [ i.e. $(M_{15-23}) > (B_{15-23})$ ] the next instruction is taken from L + 2.

This instruction is used in floating point addition and subtraction.

Registers Affected: X

Timing: 2, if no skip
3, if skip

## E. LOGICAL INSTRUCTIONS

(14) EXTRACT ETR

A logical "and" is performed between corresponding bits of the A Register and the memory location determined by the effective address. The result appears in A. The operation takes place according to the following table:

| A | M | Results in A |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Example:

| | A | M |
|---|---|---|
| Before Execution | 64231567 | 00777700 |
| After Execution | 00231500 | 00777700 |

Registers Affected: A Timing: 2

(16) MERGE MRG

A logical "inclusive or" is performed between corresponding bits of the A Register and the memory location determined by the effective address. The result appears in A. The operation takes place according to the following table:

| A | M | Result in A |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Example:

| | A | M |
|---|---|---|
| Before Execution | 06446254 | 02340712 |
| After Execution | 06746756 | 02340712 |

Registers Affected: A Timing: 2

(17) EXCLUSIVE OR EOR

A logical "exclusive or" is performed between corresponding bits of A Register and the memory location determined by the effective address. The result appears in A. The operation takes place according to the following table:

| A | M | Result in A |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Example:

| | A | M |
|---|---|---|
| Before Execution | 34165031 | 70077021 |
| After Execution | 44112010 | 70077021 |

Note: Selected bit positions of the A Register may be inverted by proper configuration of the memory word. If all "1's" appear in the memory word, a "1's complement" of A results.

Registers Affected: A

Timing: 2

## F. REGISTER CHANGE INSTRUCTIONS

All of the instructions in this group operate as micro-instructions using the same instruction code. Each of the low order nine bits of the operand determine one operation to be performed. The programmer may specify combinations of these bits to perform simultaneous functions.

The function of each address bit is as follows:

| Address Bit | Function |
|---|---|
| 23 | Clear A |
| 22 | Clear B |
| 21 | Copy (A) into B |
| 20 | Copy (B) into A |
| 19 | Copy (B) into X |
| 18 | Copy (X) into B |
| 17 | Bits 15-23 only * |
| 16 | Copy (X) into A |
| 15 | Copy (A) into X |

*See STORE EXPONENT (4600122) for special function of this bit

Example:

The following instruction copies (A) into B and clears the A Register:

046 00005

Both functions are performed simultaneously, i.e. each bit of A is copied into B and that position of A is cleared.

Indirect addressing and indexing do not apply to these instructions.

These instructions require one machine cycle regardless of the number of functions performed. As an aid to the programmer, the most useful combinations have been assigned mnemonic designations and are recognizable by the loader. These instructions follow.

(4600001) CLEAR A CLA

The contents of the A Register are set to zero.

Registers Affected: A Timing: 1

(4600002) CLEAR B CLB

The contents of the B Register are set to zero.

Registers Affected: B Timing: 1

(4600003) CLEAR AB CLR

The contents of the A and B Registers are both set to zero.

Registers Affected: A, B Timing: 1

(46000004) COPY A INTO B CAB

The contents of the A Register are copied into the B Register

Registers Affected: B Timing: 1

(4600010) COPY B INTO A CBA

The contents of the B Register are copied into the A Register.

Registers Affected: A Timing: 1

(4600014) EXCHANGE A AND B XAB

The contents of the A Register are copied into the B Register and, simultaneously, the B Register is copied into A.

Registers Affected: A, B Timing: 1

(4600200) COPY INDEX INTO A CXA

The contents of the Index Register are copied into the A Register.

Registers Affected: A Timing: 1

(4600400) COPY A INTO INDEX CAX

The contents of the A Register are copied into the Index Register.

Registers Affected: A Timing: 1

(4600600) EXCHANGE INDEX AND A XXA

The contents of the Index Register are copied into the A Register and, simultaneously, the A Register is copied into the Index Register.

Registers Affected: A, X Timing: 1

(4600020) COPY B INTO INDEX CBX

The contents of the B Register are copied into the Index Register.

Registers Affected: X Timing: 1

(400040) COPY INDEX INTO B CXB

The contents of the Index Register are copied into the B Register.

Registers Affected: B Timing: 1

(4600060) EXCHANGE INDEX AND B XXB

The contents of the Index Register are copied into the B Register and, simultaneously, the B Register is copied into the Index Register.

Registers Affected: B, X Timing: 1

(4600122) STORE EXPONENT STE

The least significant 9 bits of the B Register are copied into the Index Register. Bit 15 of the Index Register (the sign of the exponent) is then extended into bit 0. The 9 low order bits of B are then cleared.

Example:

| | B | Index |
|---|---|---|
| Before Execution | 64152713 | - - - |
| After Execution | 64152000 | 77777713 |

Note: This instruction assists in the manipulation of floating point double precision numbers where the fraction is stored in the high order 39 bits and the exponent in the low order 9 bits.

Registers Affected: B, X Timing: 1

(4600140) LOAD EXPONENT LDE

The least significant 9 bits of the Index Register are copied into the least significant 9 bits of the B Register. The previous contents of the 9 least significant bits of B are destroyed.

Example:

| | B | Index |
|---|---|---|
| Before Execution | 34765712 | 00000151 |
| After Execution | 34765151 | 00000151 |

Note: This instruction assists in the manipulation of floating point double precision numbers where the fraction is stored in the high order 39 bits and the exponent in the low order 9 bits.

Registers Affected: B Timing: 1

(4600160) EXCHANGE EXPONENTS XEE

The least significant 9 bits of the B Register and Index Register are exchanged. No information is lost in the exchange. Bit 15 of the Index Register (the sign of the exponent) is then extended into bit 0.

Example:

| | B | Index |
|---|---|---|
| Before Execution | 67142355 | 77777133 |
| After Execution | 67142133 | 00000355 |

Note: This instruction assists in the manipulation of floating point double precision numbers where the fraction is stored in the high order 39 bits and the exponent in the low order 9 bits.

Registers Affected: B, X Timing: 1

## G. SHIFT INSTRUCTIONS

These instructions use the instruction code to determine the direction of shift (66 = right; 67 = left). Bits 10 - 11 of the instruction address determine the method of shifting in accordance with the following Table:

| Contents of Bits 10 - 11 | Method of Shifting |
|---|---|
| 00 | A, B Shift |
| 10 | A, B Cycle |
| 01 | Normalize (Left only) |

The number of shifts is specified in bits 18 - 23 of the instruction address.

Indirect addressing may be used with these instructions, bits 10 and 11 of the effective address determining the method of shifting.

Indexing is performed on the low order six bits of the address only. It is thus possible to index the number of shifts without affecting the method of shifting.

(66000XX) RIGHT SHIFT AB RSH

The contents of the A and B Registers are shifted right N bit positions, N being specified in the low order six bits of the instruction. The sign bit of A is not shifted but its value is copied into vacated bit positions. The sign bit of B takes part in the shifting. Bits shifted beyond the lowest order bit of B are lost.

Example:

If the instruction is: RSH 00022

| | A | B |
|---|---|---|
| Before Execution | 45261237 | 27651260 |
| After Execution | 77777745 | 26123727 |

Note: Scaling of floating point numbers may be performed with this instruction by use of indexing where the difference of exponents is placed in the Index Register as a positive quantity.

Registers Affected: A, B

Timing: If N is the number of shifts,
$2 + \frac{N}{2}$, N even
$2 + \frac{N+1}{2}$, N odd

(66200XX) RIGHT CYCLE AB RCY

The contents of the A and B Registers are shifted right, in a ring fashion, N bit positions. N is specified in the six lowest order bits of the instruction. The signs of A and B participate in the shifting. No information is lost in this operation.

Example:

If the instruction is: RCY 00017

| | A | B |
|---|---|---|
| Before Execution | 61245703 | 41637701 |
| After Execution | 37701612 | 45703416 |

Registers Affected: A, B

Timing: If N is the number of shifts,
$2 + \frac{N}{2}$, N even
$2 + \frac{N+1}{2}$, N odd

(67000XX) LEFT SHIFT LSH

The contents of the A and B Registers are shifted left N bit positions. N is specified in the six lowest order bits of the instruction. The sign bit of A is shifted. Zeros are copied into the vacated bit positions. The sign bit of B takes part in the shifting. Bits shifted beyond position 0 of A are lost.

Overflow occurs and the Overflow Indicator is turned on if a bit is shifted beyond bit position 1 of A that differs from the sign of A.

Example:

If the instruction is: LSH 00022

| | A | B |
|---|---|---|
| Before Execution | 46712370 | 64132711 |
| After Execution | 70641327 | 11000000 |

Registers Affected: A, B

Timing: If N is the number of shifts,

$2 + \frac{N}{2}$, N even

$2 + \frac{N + 1}{2}$, N odd

(67200XX) LEFT CYCLE AB LCY

The contents of the A and B Registers are shifted left, in a ring fashion, N bit positions. N is specified in the six lowest order bits of the instruction. The signs of A and B participate in the shifting. No information is lost in this operation.

Example:

If the instruction is: LCY 00011

| | A | B |
|---|---|---|
| Before Execution | 71432560 | 34156723 |
| After Execution | 32560341 | 56723714 |

Registers Affected: A, B

Timing: If N is the number of shifts,

$2 + \frac{N}{2}$, N even

$2 + \frac{N + 1}{2}$, N odd

(67100XX) NORMALIZE AND DECREMENT X NOD

The contents of the A and B Registers are shifted left and a one is subtracted from the Index Register for each bit position shifted. Shifting terminates when either (a) a bit appears in bit position 1 of A which is opposite to the sign of A, (i.e., $|A| \geq 1/2$), or (b) N shifts have occurred (N being specified in the six lowest order bits of the instruction). The programmer must ensure that N is sufficiently large to permit the normalize to be completed.

The shifting rules are identical with LEFT SHIFT (67000XX).

Decrementing of the Index Register occurs regardless of the contents of the Index Bit.

Example:

If the instruction is: NOD 00030

| | A | B | Index |
|---|---|---|---|
| Before Execution | 00004632 | 76124035 | 00000000 |
| After Execution | 23153705 | 20164000 | 77777765 |

Note: This instruction is normally used in conjunction with RIGHT SHIFT (66000XX) to manipulate floating point numbers.

Registers Affected: A, B, X

Timing: if R is the resultant number of shifts,

$2 + \frac{R}{2}$, R even

$2 + \frac{R + 1}{2}$, R odd

## H. CONTROL INSTRUCTIONS

(00) HALT HLT

The computer halts computation and the HALT light on the control panel is turned on. To resume computation, the operator must first return the "Run-Idle-Step" switch to the "Idle" position. The next instruction will be executed (and the HALT light turned off) if this switch is then placed in either "Run" or "Step." Indirect addressing and indexing do not apply to this instruction.

Registers Affected: None Timing: 1

(20) NO OPERATION NOP

This instruction executes without effect on the A Register, B Register, or memory. Indirect addressing and indexing do not apply to this instruction.

Registers Affected: None Timing: 1

(23) EXECUTE EXU

The instruction in the memory location determined by the effective address is executed without changing the contents of the P Register. If the Execute Instruction is in Location L, and the effective address of this instruction is M, instructions are performed in the following order:

L
M
L + 1

If in the above sequence M contains a Branch instruction, control will not return to L + 1, but to the effective location of the branch. If M contains a Skip instruction, control will return to L + 1 or L + 2, depending on whether a skip takes place.

If M contains another Execute instruction, the process will continue except that control will return to L + 1 after completion of the first normal (Non-Branch, Non-Skip, Non-Execute) instruction.

Registers Affected: None Timing: 1

(77) COPY EFFECTIVE ADDRESS INTO INDEX EAX

The effective address of the instruction is copied into the Index Register. Three distinct types of operations are performed using this instruction as a function of the contents of the Index Bit and Indirect Address Bit. These are summarized as follows:

| Instruction | Function | Description |
|---|---|---|
| EAX M | M → X | The address portion of the instruction is copied into the Index Register. |
| 1 EAX M | M + (X) → X | The address portion of the instruction is added to the contents of the Index Register and the results copied into the Index Register. |
| EAX I M | Effective Address of M → X | The effective address of the instruction is copied into the Index Register. |

The nine highest order bits of X are not affected by this instruction.

Registers Affected: X Timing: 2

## I. INPUT/OUTPUT INSTRUCTIONS

The Input/Output Instructions for the SDS 920 are designed to communicate with external devices either in conjunction with the interrupt system, or in an independent fashion. When the Input/Output Instructions are used with the interrupt system, the interrupt signal itself provides synchronization. When the instructions are used without the interrupt system, the computer provides an automatic delay to synchronize with the external device; the instructions "lock up" until the computer is synchronized with the input or output transfer.

(12) M INTO W BUFFER WHEN READY MIW

The contents of the memory location specified by the effective address are copied into the W Buffer Register. If the interrupt is employed, it synchronizes the transfer; if not, the operation will not take place until the W Buffer Register has been emptied. The computer will lock up until that time.

Registers Affected: None Timing: 2 + wait

(32) W BUFFER INTO M WHEN READY WIM

The contents of the W Buffer Register are transferred to the memory location specified by the effective address. The W Register is cleared and is available for re-loading by the external device. If the interrupt is employed, it synchronizes the transfer; if not, the computer will lock up until the W Register has been filled and then perform the transfer.

Registers Affected: M Timing: 3 + wait

(10) M INTO Y BUFFER WHEN READY MIY

The contents of the memory location specified by the effective address are copied into the Y Buffer Register, an optional Input/Output Register. If the interrupt is employed, it synchronizes the transfer; if not, the operation will not take place until the Y Buffer Register has been emptied. The computer will lock up until that time.

Registers Affected: None Timing: 2 + wait

(30) Y BUFFER INTO M WHEN READY YIM

The contents of the optional Y Buffer Register are transferred to the memory location specified by the effective address. The Y Register is cleared and is available for re-loading by the external device. If

the interrupt is employed, it synchronizes the transfer; if not, the computer will lock up until Y has been filled and then perform the transfer.

Registers Affected: M Timing: 3 + wait

(13) PARALLEL OUTPUT POT

The contents of the effective address are held in the C Register for parallel transfer to an external device. Unless synchronized by the interrupt, the computer locks up until released by the external device. The external device is specified by the 14 bits of the effective address.

The effective address determines both the external device selected and the memory location of the data. Thus information transmitted to an external device is stored in a memory location unique to that device.

Note: A typical use for this instruction is for sending a parallel output to a digital-analog converter.

Registers Affected: None Timing: 2 +wait

(33) PARALLEL INPUT PIN

Unless synchronized by the interrupt, the computer "locks up" until the external device transfers parallel data, up to 24 bits, into the C Register. When the transfer is completed, the external device releases the computer and the contents of the C Register (including computer-generated parity) are stored in the memory location specified by the effective address. The external device is specified by the effective address.

The effective address determines both the external device selected and the memory location of the data. Thus information received from an external device is stored in a memory location unique to that device.

Note: This instruction is used to receive data from such devices as analog-digital converters.

Registers Affected: M Timing: 3 +wait

(02) ENERGIZE OUTPUT M EOM

This instruction provides an 8 microsecond output pulse and certain output control signals. The effective address of the instruction determines the destination of the pulse.

Appendix D contains a full description of the use of this instruction.

Indirect addressing and indexing do not apply to this instruction.

Registers Affected: None Timing: 1

# III. Floating Point Operations

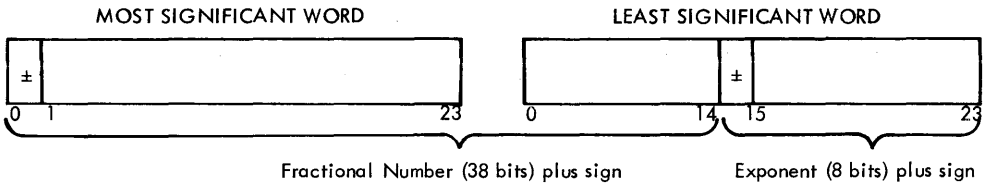
The SDS 920 is designed to manipulate floating point numbers in a fast, efficient manner, due to the inclusion of a special group of instructions in the SDS 920 Instruction List. Floating point relieves the programmer of the burden of maintaining the binary point while arithmetic operations are being performed. The scaling of numbers is automatically maintained by compact Programmed Operator subroutines designed for the SDS 920. Numbers are represented in the form

$$F \times 2^{E}$$

and where F is the fractional number and E is the exponent.

Floating Point operations may be performed in either single or double precision on the SDS 920. Double precision is used when accuracy of approximately 11 decimal digits must be maintained. Single precision permits faster execution times when approximately seven decimal digits of accuracy are required.

## A. DOUBLE PRECISION FLOATING POINT FORMAT

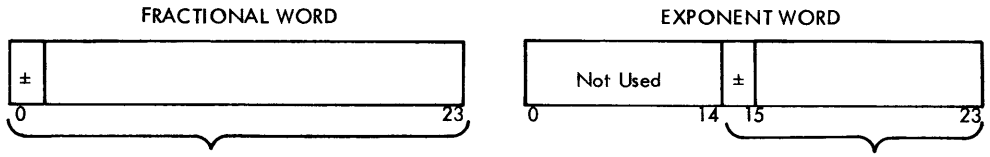



Double Precision Floating Point operations in the SDS 920 are performed using a fractional number of 39 bits (38 bits plus sign) and an exponent of 9 bits (8 bits plus sign). Numbers can be represented with F equal to 11 decimal digits plus sign and multiplier as high as $10^{\pm 73}$.

The Programmed Operator subroutines which perform Double Precision Floating Point operations are:

| Designation | Name | Function | Approx. Execution Time |
|---|---|---|---|
| FLA | Floating Add | Floating (A,B) + (M,M+1) → A,B | 368 μsec |
| FLS | Floating Subtract | Floating (A,B) − (M,M+1) → A,B | 368 μsec |
| FLM | Floating Multiply | Floating (A,B) × (M,M+1) → A,B | 560 μsec |
| FLD | Floating Divide | Floating (A,B) ÷ (M,M+1) → A,B | 800 μsec |

## B. SINGLE PRECISION FLOATING POINT FORMAT




Fractional Number (23 bits) plus sign

Exponent (8 bits) plus sign

Single Precision Floating Point operations in the SDS 920 are performed using a fractional number of 24 bits (23 bits plus sign) and an exponent of 9 bits (8 bits plus sign). Numbers can be represented with F equal to 6 decimal digits plus sign and an exponent as high as $10^{\pm 73}$. The Programmed Operator subroutines which perform Single Precision Floating Point operations are:

| Designation | Name | Function | Approx. Execution Time |
|---|---|---|---|
| FSA | Floating Add, Single Precision | Floating (A) + (M) → A Exponent in B,M + 1 | 192 μsec |
| FSS | Floating Subtract, Single Precision | Floating (A) - (M) → A Exponent in B,M + 1 | 192 μsec |
| FSM | Floating Multiply Single Precision | Floating (A) x (M) → A Exponent in B,M + 1 | 280 μsec |
| FSD | Floating Divide Single Precision | Floating (A) − (M) → A Exponent in B,M + 1 | 368 μsec |

## C. SDS 920 INSTRUCTIONS FOR FLOATING POINT

In order to maintain accuracy in floating point operations, all fractional numbers must be in normalized form, i.e., shifted to the left to eliminate leading insignificant digits. When numbers are inserted into the computer, and when a floating point arithmetic operation has been performed, the fractional number must be normalized and the exponent adjusted to reflect the change in the fractional number. In the SDS 920, NORMALIZE AND DECREMENT X (67100XX) is used to

a) shift the fractional number to the left to eliminate leading insignificant digits and

b) adjust the exponent (contained in the X Register) for each bit position shifted.

When performing floating point addition and subtraction, it is necessary to align the numbers so that the exponents are equal before the arithmetic is performed. In the SDS 920, DIFFERENCE EXPONENTS AND SKIP (74), determines in one instruction

a) which of the numbers is to be shifted and

b) the number of positions to be shifted to align the numbers. Alignment is performed using SHIFT AB (66000XX), Index Bit equal to one, with the number of shifts located in the X Register.

Manipulation of the exponent is required in all floating point operations. Capability is included in the Register Change Instructions to

a) transfer the exponent portion of the word to and from the A, B and X Registers and

b) clear exponent bits when arithmetic is to be performed.

These operations can be performed in effective combinations in one machine cycle (8μsec).

# IV. Input / Output

## A. INTRODUCTION

The SDS 920 is capable of sending and receiving information at character rates of up to 500 kc. Provisions are included for handling the following types of data:

1. Single bit input, such as equipment on/off status and breakpoint switches.

2. Single bit output, such as start/stop signals to external devices.

3. Seven bit parallel input/output where parity generation/detection and character-to-word assembly is automatic. Simultaneous input and output can be performed.

4. Parallel input/output of up to twenty-four bits under clocking control of the external device. Transfers are under program control.

5. Input/output, under external control, of up to twenty-four bits. The external device interlaces input/output operations with computer program operation by having direct control over the computer's main memory.

In addition, a flexible Priority Interrupt System is included which permits efficient input/output control over a large variety of devices.

## B. METHODS OF INPUT/OUTPUT

### 1. Single Bit Input/Output

The SDS 920 is provided with instructions which transmit an eight microsecond pulse to set external signals (ENERGIZE OUTPUT M (02) ) and test the status of these signals (SKIP IF SIGNAL NOT SET (40) ). Since the address portion of each instruction is used to identify the external signal, over 16,000 distinct signals can be connected. Address configurations have been assigned for standard input/out addressing and computer signals. These addresses are described in Appendix D. Addresses not defined may be used for special single-bit communication as required.

## 2. Input/Output Buffer System

The Input/Output Buffer, or W Register, contains a full-word register plus the associated control circuitry needed to transfer full data words to and from the computer's memory and seven-bit characters to and from external devices. Parity detection/generation is automatically performed in the buffer; the parity bit is not part of the computer word.

Input/Output Operations using the Input/Output Buffer are performed in the following manner:

ENERGIZE OUTPUT M (02) sets the Input/Output address, Input or Output Status, and the character count (1 - 4 characters) (see Appendix D). Normally, this activates the external unit and the operation is begun. W BUFFER INTO MEMORY WHEN READY (32) and M INTO W BUFFER WHEN READY (12) are then used to transfer information between memory and the W Register. When the W Register has been filled on input (or emptied on output), an interrupt occurs to cause the computer to unload (or fill) the buffer. The programmer has the option, however, of disabling the interrupt prior to starting if the computer is not to perform simultaneous computation and input/output. In this case, when the input/output instruction is executed, the computer will "lock up" in this instruction until the W Register has been emptied (or filled).

The character counter, as set by the program, determines the number of characters required to fill the buffer. The character counter is set as follows:

| Character Counter | Number of Characters to Fill Buffer |
|---|---|
| 00 | 1 |
| 01 | 2 |
| 10 | 3 |
| 11 | 4 |

If the program sets a character count of "1", the buffer is determined to be full when one character is received from the input device and stored in the least significant six bits of the W Register. If the program sets a character count of "4", four characters are sequentially received from the input device and stored in the W Register; the first character appears in the most significant portion, and the last in the least significant six bits. Output operations are performed in the same manner.

The Input/Output Buffer provides a 5.5 character buffer between the computer and the external device. When the W Register has been emptied (or filled), an additional character may be completely processed before the W Register must be accessed. Typically, the maximum time available between successive input (or output) instructions is C + 1.5 character times, where C is the Character Count.

While the W Register is being emptied (or filled) by the external device, other computation can be performed.

### 3. Parallel Input/Output

Data can be transferred between the SDS 920 and other devices, one word at a time (twenty-four bits in parallel). Two instructions are provided for such parallel data transfers, PARALLEL INPUT (33) and PARALLEL OUTPUT (13).

PARALLEL INPUT (33) and PARALLEL OUTPUT (13) addresses determine both the selected external unit and the memory location accessed. Thus, information to or from a device is stored in a memory location unique to that device.

When PARALLEL INPUT (33) is executed, a "ready" signal plus the 14 address bits are transmitted to the peripheral unit. The address bits uniquely identify the unit to be connected, such as an analog-to-digital converter. The ready signal causes the external unit to transmit up to 24 information bits directly into the C Register. After the information has been transmitted, the external unit sends a "complete" pulse to the computer. When received by the computer, memory parity is generated and the word stored in the memory location determined by the address.

Parallel output functions in a similar manner.

### 4. External Memory Interlace

The SDS 920 is designed so that external devices may communicate directly with the computer's memory while internal computation is being performed.

Information is entered into (or taken from) the W (and/or Y) buffer and computation is automatically halted when the buffer is full (or empty). An externally-held address controls the operation of the memory during this period, which is two machine cycles. The computer is then restarted. The computer can be halted in this manner during a long instruction (over 3 machine cycles). The maximum transfer rate is approximately 200,000 characters per second.

## C. PRIORITY INTERRUPT SYSTEM

The SDS 920 has two priority interrupt channels for handling of asynchronous input/output devices in conjunction with the Input/Output Buffer. The channels are identified as Priority Interrupt Channel 30 and Priority Interrupt Channel 31, the lower-numbered channel having the highest priority. Each channel causes the computer to interrupt to a unique memory location, corresponding to the channel number.

When an interrupt signal is received, the present instruction is completed and control is then transferred to the memory location corresponding to the channel number. The P Register is left unchanged by operation. The next instruction executed is MARK PLACE AND BRANCH (43). This instruction stores the present contents of the P Register and initiates the required subroutine. When the subroutine is completed, control must be returned to the interrupted program using BRANCH UNCONDITIONALLY (01) with indirect addressing. This instruction automatically clears the Interrupt Channel.

The Interrupt System may be enabled or disabled by the program, as required, using ENERGIZE OUTPUT M (02), or may be manually enabled from the Control Panel, overriding the interrupt status set by the program. When each interrupt operation is obeyed, MARK PLACE AND BRANCH (43) is executed immediately during the next machine cycle.

Additional priority interrupt channels are also available (see Section E).

## D. STANDARD INPUT/OUTPUT DEVICES

### 1. Control Panel

The SDS 920 is equipped with a convenient control panel, both to permit operator intervention and to display the internal status of the computer. A diagram of the control panel is shown in Figure IV-1. Contained on the control panel are the following:

a. Displays

(1) PROGRAM LOCATION - 14-bit P Register is continuously displayed.

(2) REGISTER DISPLAY - A 24-bit register display is provided in conjunction with a rotary selection switch. This display will contain values of different internal registers depending on the location of the rotary selection switch. These are:

| Rotary Selection Switch Position | Display |
|---|---|
| C | C Register (which contains the full instruction immediately prior to execution). |
| A | A Register |
| B | B Register |
| X | X Register |

(3) OVERFLOW - The status of the overflow indicator is reflected by this display. A description of the operation of this indicator is contained in Section I.

(4) HALT - This indicator is turned on whenever HALT (00) is executed. It is turned off whenever an instruction other than HALT (00) is executed.

(5) MEMORY PARITY - This indicator is turned on whenever a memory parity error occurs. It is turned off by returning the RUN-IDLE-STEP Switch to the IDLE position. This indicator operates in conjunction with the PARITY Switch described below.

(6) INTERRUPT ENABLED - When this indicator is on, the Interrupt System is enabled.

(7) INPUT/OUTPUT - Six of these indicators reflect the address of the Input/Output Unit presently connected to the W Register (see Appendix D). The seventh indicator reflects the validity of the Input/Output Error data.

Switches

(1) CLEAR - When the computer is in IDLE, depressing this switch clears the contents of the register being displayed (see REGISTER SWITCHES for resetting).

(2) REGISTER SWITCHES - These 24 switches, located directly below the Register Display, permit insertion of information directly into the register being displayed. The computer must be in IDLE and the register cleared. Depressing a switch places a "1" in the corresponding bit position of the register being displayed.

(3) RUN-IDLE-STEP - This is a three-position toggle with two stationary positions and a spring-loaded momentary position in STEP. In the RUN position, computation occurs at machine speed. In the IDLE position, the computer idles immediately after an instruction has been read from memory. If the Register Switch is in the "C" position, the complete instruction may be viewed in the Register Display. In the STEP position, the instruction is executed and the computer returns to the IDLE state. The switch must be released to the IDLE position before another STEP may be performed.

(4) REGISTER - The function of this rotary switch is described under Register Display.

(5) HOLD - If this switch is in the UP position, the P Register will not be stepped. This switch permits execution of additional instructions during checkout without changing the program stored in memory.

(6) MEMORY PARITY - If this switch is in the HALT position, the computer will enter an IDLE state whenever a memory parity error occurs. If this switch is in the CONTINUE position, the computer will not change state when memory parity occurs.

(7) INTERRUPT ENABLED - If this switch is in the COMPUTER position, the Interrupt System may be enabled or disabled under program control. If this switch is in the ENABLE position, the Interrupt System is enabled regardless of program operations.

(8) BREAKPOINT - The status of these four switches may be detected by the program using SKIP IF SIGNAL NOT SET (40). They are used to control pre-determined options within the program.

(9) START - This switch is used to initialize the control section of the computer. The W Buffer is reset, the P Register is cleared, and HALT (00) is automatically executed. The RUN-IDLE-STEP Switch must be in IDLE when this switch is depressed.

(10) FILL - Depressing this switch causes the computer to read one word (4 characters) automatically from Paper Tape Reader No. 1 into memory location 0002 and execute the instruction. Simultaneously, the X Register is set to the value 77777772 (-6). The following short program (called a "bootstrap") can then be loaded and executed without further action by the operator. The bootstrap is capable of loading a binary tape of any length into any portion of memory.

| LOCATION | INSTRUCTION | | |
|---|---|---|---|
| 0002 | 1 | WIM | 0011 |
| 0003 | | BRX | 0002 |
| 0004 | | LDX | 0010 |
| 0005 | 1 | WIM | (ending address of program + 1) |
| 0006 | | BRX | 0005 |
| 0007 | | BRU | EXIT |
| 0010 | | (number of words in program as a negative value) | |

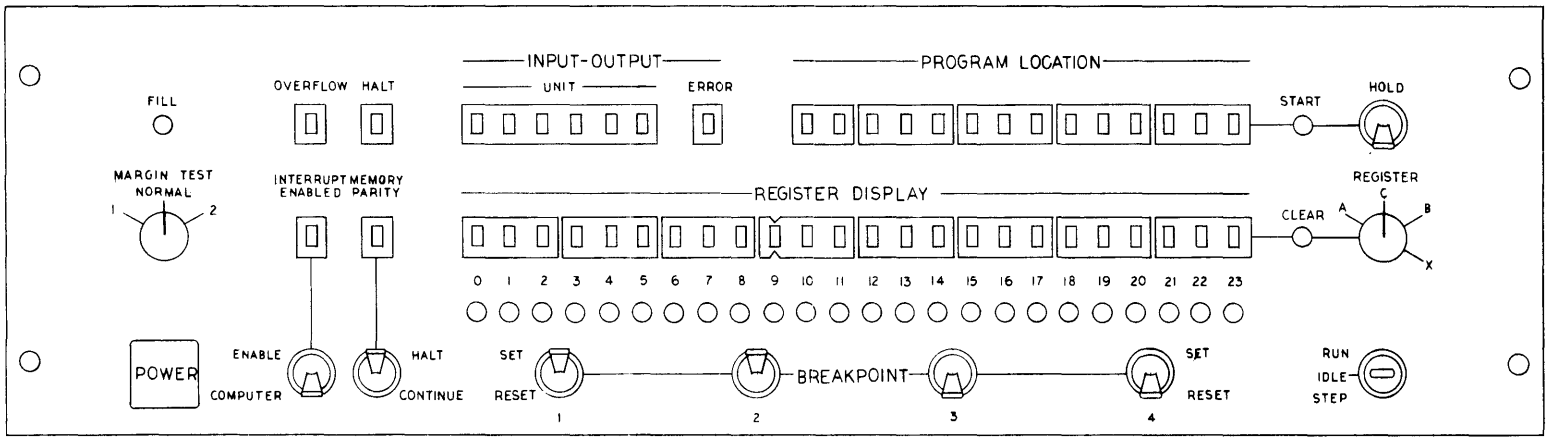



Figure IV-1 Control Panel

The sequence of operations performed by the operator to load a program using the FILL Switch is as follows:

(a) Insert the program to be loaded in Paper Tape Reader No. 1. The initial portion of the tape contains the bootstrap program.

(b) Set the RUN-IDLE-STEP Switch in the IDLE position.

(c) Depress START

(d) Set the RUN-IDLE-STEP Switch in the RUN position.

(e) Depress FILL

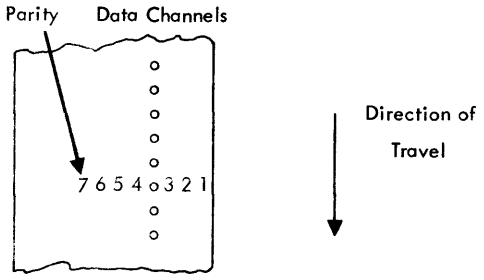
If the loading system is present in memory, only steps (b) and (c) above need be performed. An unconditional branch to the initial location of the loading system can then be executed, using the Register Switches.

(11) POWER - This is a back-lighted switch which applies power to the computer.

(12) MARGIN TEST - This rotary switch is used during maintenance to test computer operation under marginal test conditions.

2. Photo Electric Reader

The SDS 920 Photo Electric Reader reads 7-channel paper tape into the computer at 300 characters per second. A Paper Tape Spooler with two removable 8" NAB tape reels can be provided for tape handling. The paper tape format is as follows:



All characters will contain a parity bit such that the number of holes in each character is odd. Characters at the beginning of a tape which are without holes in any of the seven channels are ignored by the reader and are not transmitted to the computer.

Paper tape information is read into the SDS 920 as follows:

The computer program addresses the reader and sets the character count simultaneously, using ENERGIZE OUTPUT M (02). When addressed, the photo reader automatically turns on and begins reading paper tape. No information is transmitted to the computer until a character is detected with a hole in at least one of the channels. Each character is transferred to the Input/Output Buffer as is detected. At the buffer, parity is checked and the character is shifted serially into the W Register. The character count is reduced by one. If the W Register is full, the information is held in the character flip-flops until the buffer has been emptied. This procedure continues until either leader appears on the tape or the reader is turned off by the program. ENERGIZE OUTPUT M (02) is used to disconnect the reader and simultaneously turn it off by placing an address, different from that of the reader, in the Input/Output address flip-flops.

Parity errors or any other errors detected by the buffer system cause the Input/Output Error Indicator to be turned on. Information transfer, however, continues until either the end of the data or a programmed stop. The Input/Output Error Indicator can be sensed, at any time, by the program. Normally, however, the program will sense errors after a block of information has been read or when an error is suspected (i.e., false codes received).

### 3. Paper Tape Punch

The SDS 920 Paper Tape Punch operates at 60 paper tape characters per second. A removable 8" NAB tape reel is provided for handling the output tape. Seven paper tape channels are punched per character; six data channels and one parity channel generated in the Input/Output Buffer. The paper tape format is shown under Photo Electric Reader.

Addressing (including energizing and de-energizing) the Paper Tape Punch is identical with the Photo Electric Reader.

### 4. Typewriter

The SDS 920 Typewriter is used for both input and output. Data can be manually inserted from the typewriter into the Input/Output Buffer. Information can be transmitted from the Input/Output Buffer directly to the typewriter for printing of information. The typewriter is operated at a 15 character per second printing rate. The typewriter input/output codes are detailed in Appendix E.

A Breakpoint Switch is used to indicate to the computer that entry of information through the typewriter is required. The computer is programmed to periodically sense the Breakpoint switch and, if on, address the typewriter. A light on the typewriter is turned on whenever a typewriter input is requested by the program. When the light is on, the operator may type a key which is transmitted directly to the Input/Output Buffer. The Interrupt System is normally used for typewriter input since the character input rate is extremely slow compared to the computation speed of the SDS 920 (approximately 10,000 instructions may be executed between each character typed on the keyboard).

Output characters are transmitted to the typewriter via the Input/Output Buffer. Up to four output characters can be stored in an SDS 920 computer word for transfer to the W Register. The Interrupt System is normally used in typewriter output.

## E. OPTIONAL INPUT/OUTPUT DEVICES

### 1. Additional Priority Interrupt Channels

Additional priority interrupt channels are available with the SDS 920 in groups of 16 channels, to as many as 1024 channels. Each channel is assigned a priority status - no channels being on the same priority level. A channel of a given priority may not cause an interrupt to occur if an interrupt with higher priority is being obeyed. In this case, the interrupt is held until higher priority interrupts have been completed.

### 2. Additional Input/Output Buffer

The SDS 920 is pre-wired to include an additional Input/Output Buffer identical to the standard buffer. The register contained with this buffer is called the Y Register. The standard SDS 920 instructions M INTO Y BUFFER WHEN READY (10) and Y BUFFER INTO M WHEN READY (30) are used to transfer data between the Y Register and memory.

Input/Output devices may be connected to either the W or the Y Buffer, as desired.

Two additional channels of priority interrupt are supplied with this Input/Output Buffer to facilitate simultaneous operations.

### 3. Paper Tape Spooler

A Paper Tape Spooler is available with the SDS 920 for use with the Paper Tape Reader. The Paper Tape Spooler contains two removable 8" NAB tape reels and take-up arms for controlled tape feeding. Manual switches are included for Forward/Reverse and On/Off.

## 4. Magnetic Tape Systems

Two types of Magnetic Tape Systems are available with the SDS 920. A practically unlimited number of these can be connected to the computer. Both Systems record in standard IBM format with 7 bits per character.

Redundant heads are provided for parity checking by a "read-after-write" system. Longitudinal parity and standard gaps are automatically generated during writing operations. Parity can be detected by the program at any time, although detection is typically performed at the end of reading or writing one record. A Ready signal is provided in each tape unit and this signal can be sensed by the program before any read-write operation. If "Ready" is on, the magnetic tape unit is ready in all respects to send or receive data.

### a. Program-Controlled Buffer System

The Magnetic Tape System, using the Program-Controlled Input/Output Buffer, operates at character transfer rates up to 30 kc in the SDS 920. The computer is available for simultaneous computing for a large portion of the time during read-write operations. At 2 kc, 4.8% of computer time is used for input/output while at 15 kc, 36% is used.

During write operations, the program transmits data, one word at a time, to the Input/Output Buffer. Here characters are formed and parity bits are generated. Each character is transmitted to the Tape Unit in parallel. When the last word of the record has been transferred to the Input/Output Buffer, the program signals the Tape Unit that no further data is to be sent. Longitudinal parity and an end-of-record gap are automatically generated and the Tape Unit stops. When the record has been completed, the computer is interrupted for error-checking purposes.

During read operations, the program controls inputs in a similar manner. When the inter-record gap appears under the read head, data input ceases and the computer is interrupted. This indicates to the program that the record is complete.

The system provides for rapid scanning of the tape in either direction, under program control. When in the scanning mode, the Tape Unit moves in a forward or reverse direction at reading speed. An interrupt is provided to the computer when each inter-record gap is detected. The computer has the option of continuing the scan by signalling the Tape Unit using ENERGIZE OUTPUT M (02). If no signal is transmitted by the computer, the Tape Unit stops. This enables the computer to search for a required record by either counting inter-record gaps, or by record identification. When the computer is interrupted, the last word of the record is present in the Input/Output Buffer. This word may be used to identify the record.

Portions of the tape may be erased, under program control, using a special Input/Output Address (see Appendix D). In this mode, inter-record gap is written on the tape for each character transmitted by the program.

A 15 kc magnetic tape unit is normally employed with the Program-Controlled Buffer System.

b. Automatic External Memory Interlace System

The Magnetic Tape Unit, using external memory interlace, includes storage for memory control information. Information transfer takes place through one of the input/output buffers. Since these operations are not under program control, only a small fraction of the available time (typically 5% at a 15 kc character rate) is used. During the remaining time, input/output and computation proceed simultaneously.

Prior to reading or writing a record, the SDS 920 program transmits the following information to the Tape Unit using PARALLEL OUTPUT (13):

(1) starting memory address
(2) number of words

The computer starts the read-write process using ENERGIZE OUTPUT M (02). The complete operation is then performed without computer intervention. Upon completion of transmission, the computer is interrupted to perform read-write error checking. If, during the read operation, the number of words requested by the computer is smaller than the length of the tape record, an error indicator is turned on. No transmission occurs after the number of words read into the computer are equal to the number requested and the Tape Unit stops when the inter-record gap appears.

5. Other Devices

In addition to the Input/Output units described herein, the SDS 920 is capable of being effectively connected to a wide range of devices. Some of these are:

a. Punched Card Reader
b. Punched Card Punch
c. Line Printers
d. Disc Files
e. Analog-to-Digital Converters
f. Digital-to-Analog Converters
g. Other SDS Computers

Practically any number of these devices can be used in any combination with the SDS 920.

## F. SYSTEM CONSIDERATIONS

The SDS 920 is easily integrated into a wide range of computing systems as well as systems for data acquisition and real time control.

As indicated in this section, the SDS 920 is designed to operate -- without special buffering equipment -- with a wide range of input/output devices. All of these have been designed with standard interface characteristics so that the construction of a complex system simply requires the interconnecting of the various elements rather than any extensive engineering design.

### 1. Computing Systems

Appendix D is a listing of typical devices that can be employed with the SDS 920 to construct comprehensive systems. In addition, another SDS computer can be used as a satellite input/output processor with the SDS 920 so as to minimize the time spent in input/output.

With the addition of a second input/output buffer (see Section E), the SDS 920 can serve as a high-speed universal data converter between various media and various formats. For example, the SDS 920 can convert gapless magnetic tape to gapped tape at character rates of up to 5 kc.

### 2. Data Acquisition and Real Time Control

The SDS 920 as a system component is best discussed in connection with block diagrams.

#### a. Data Acquisition

Figure IV-2 is a typical data acquisition system. Analog signals of various sorts are multiplexed, under the control of the SDS 920, into an analog-to-digital converter. The digital output of the converter is read into the SDS 920 together with any number of other digital inputs and the computer performs one or all of the following functions:

(1) Zero correction
(2) Full-scale correction
(3) Conversion to engineering units
(4) Formatting for recording on magnetic tape
(5) Selection and printing of certain data points
(6) Selection of points for plotting via digital-to-analog converters
(7) Limit comparison and display
(8) Computation of performance variables on the basis of the acquired data
(9) Transmission of some of the data to other computers and systems

A typical system can process 2500 inputs/sec while performing operations (1), (2), (3), and (4).

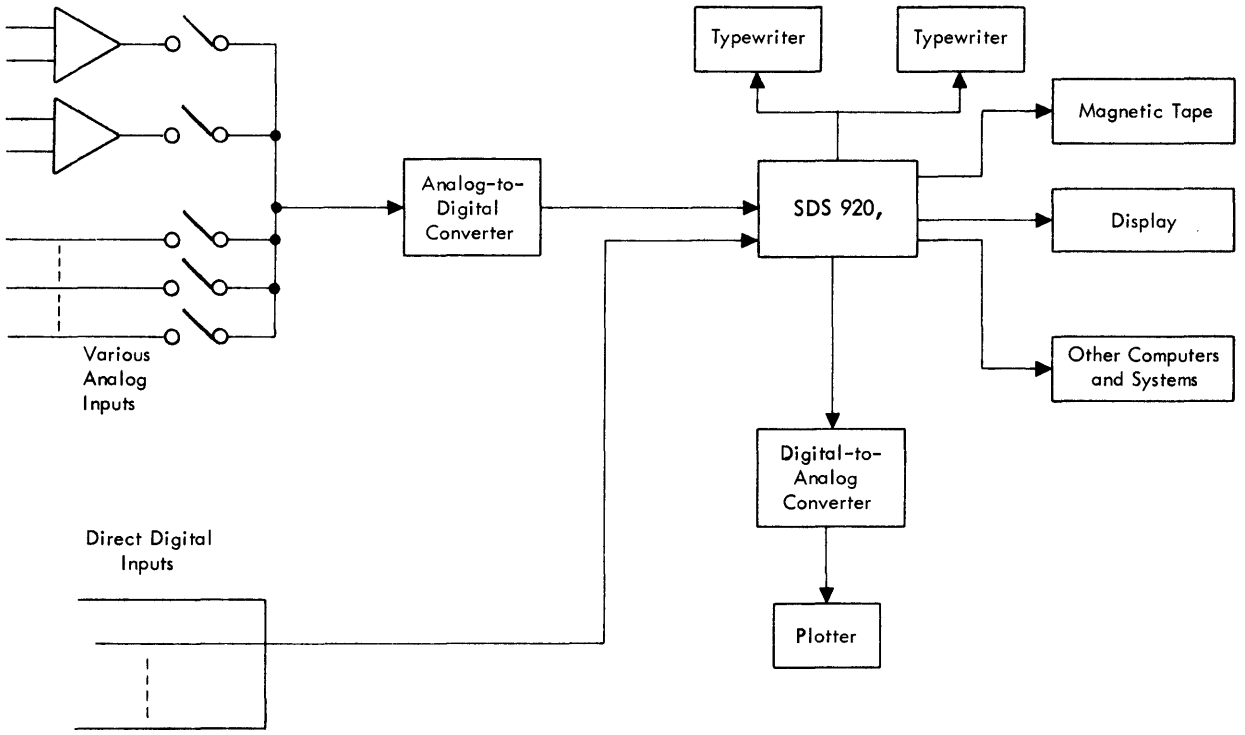



Figure IV-2 Typical Data Acquisition System

b. Real Time Control

Figure IV-3 is a typical control system. Input variables -- which may be digital or analog -- are entered into the SDS 920 through the appropriate input devices together with input control signals. The latter are processed by the Priority Interrupt System that selects the order in which various functions are to be performed. Some of the functions may require special programs to be executed. These are stored on magnetic tape or in a disc file. The computer transforms the incoming data, prints the results, uses some results to generate analog voltages for controlling devices such as valves or other actuators, and generates control signals to start, stop, and signal various other operations. A manual keyboard facilitates changes in limits, scale factor, etc.

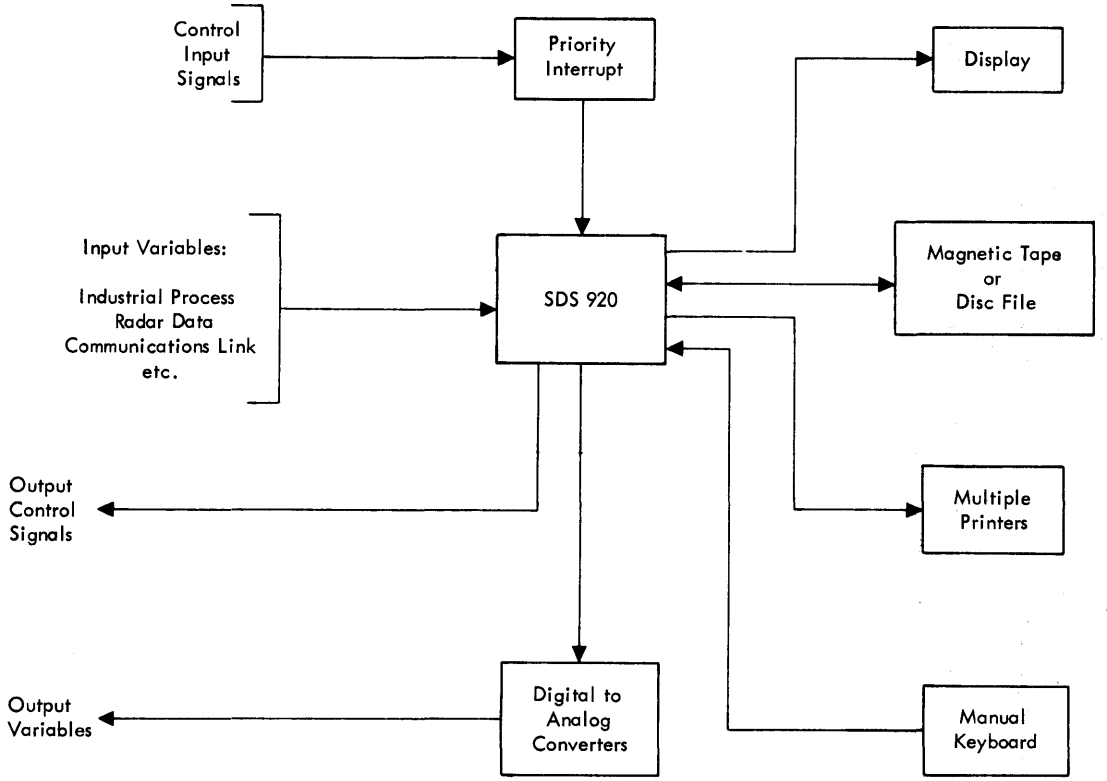



Figure IV-3 Typical Real Time Control System

# V. Programming Examples

The programming examples included in this section make use of many of the powerful instructions included in the SDS 920. Special emphasis has been given to input/output programming. Octal notation is used throughout, except as noted.

## A. EXAMPLE NO. 1 (PAPER TAPE INPUT)

### 1. Problem

Read a binary paper tape (previously generated by the computer) into memory from Paper Tape Reader No. 1. The ending location has been entered through the typewriter and is stored (plus one) as an address in a location internal to the program. The program must be relocatable. The Interrupt System is not used since simultaneous computation is not being performed. The first word on the binary tape contains the total number of words to be read. The last word contains a complement check sum of all preceding data words. The first word is not included in the sum.

### 2. Program

| Location | Instruction | Remarks |
|---|---|---|
| L | EOM 20004 | Disable Interrupt System |
| | EOM 01304 | Address and start reader |
| | 2 WIM +16 | Word count → temp. storage |
| | CLA | O → A |
| | 2 SUB +14 | Complement of Word Count → A |
| | 046 00401 | 0 → A, A → X combined |
| L + 6 | 2 WIMI +13 | ith Word → Memory |
| L + 7 | 2 ADDI +12 | Accumulate check sum |
| L + 10 | 2 BRX -2 | X + 1 → X; complete? |
| L + 11 | SKS 20010 | Complete; Input/Output Error? |
| | 2 BRU ERROR | Yes - to Error Routine |
| L + 13 | 2 SKA +4 | No Error; Check Sum = 0 ? |
| | 2 BRU ERROR | No - to Error Routine |
| | EOM 00000 | Check Sum = 0; Disconnect |
| | BRU REST | To Rest of Program |
| | OCT77777777 | Constant for Skip on Zero |

2. Program

| Location | Instruction | Remarks |
|---|---|---|
| | OCT00000000 | Temporary storage |
| L + 21 | 1 HLT LOC | Ending location plus one with Index Bit equal to 1 |

Storage - $18_{10}$ words

Timing - (136 + 64i) μsec where i = no. of words

3. Explanation

All instruction which refer to memory locations within the program have been assigned to a Relative Address Bit = 1. Starting at L the first word on the input tape is complemented and stored in the Index Register where it is used to count the input words and for address modification. The 3 instruction loop starting at location L + 6 performs the input of all data words. The ending location plus one is stored in location L + 21 with an Index Register Bit = 1. The instructions in locations L + 6 and L + 7 both refer to this location as an indirect address. The effective address of the instructions in L + 6 and L + 7 is the ending location + 1 + Index. During the last complete pass, the Index Register is equal to -1. The effective address, then, is the ending location. The instructions in L + 11 and L + 13 test for Input/Output Error and Check Sum Error, respectively. If an error exists, the program branches to an ERROR routine (not shown).

This input program requires only 64 μsec of computer time for each input word. Since the Paper Tape Reader, operating at 300 characters per second, makes a word available to the SDS 920 every 13.3 m sec, over 99% of the time is available for other computation. It is possible to receive data using this program at character rates over 30 kc.

## B. EXAMPLE NO. 2 (MAGNETIC TAPE OUTPUT)

### 1. Problem

Write a record on Magnetic Tape Unit No. 1 using the W Register: 100 data words starting at location 2000. Computation is performed simultaneously with writing and the interrupt system is enabled. The A, B, and X Registers are not used by this program except during the initializing pass. This permits the main program to continue without being disturbed. Note that a faster and simpler program is possible if this restriction is removed.

### 2. Program

| Location | Instruction | | Remarks |
|---|---|---|---|
| 30 | BRM | 40 | Interrupt Channel #30 Entrance |
| 31 | BRM | 47 | Interrupt Channel #31 Entrance |
| 32 | LDA | 54 | Initial Entrance - |
| 33 | STA | 41 | Set up initial address and word count; |
| 34 | LDA | 55 | Start Magnetic Tape Unit #1 |
| 35 | STA | 56 | |
| 36 | EOM | 01350 | |
| 37 | BRU | OUT | Initial Exit |
| 40 | HLT | 0 | Channel #30 Return Address Storage |
| 41 | (MIW | START) | Word → W Buffer |
| 42 | MIN | 41 | Increase address by one |
| 43 | SKR | 56 | Reduce Word Count by one; Done? |
| 44 | BRU I | 40 | No; Return to Main Program |
| 45 | EOM | 10100 | Yes; send Last Word Signal to W Buffer and |
| 46 | BRU I | 40 | Return to Main Program |
| 47 | HLT | 0 | Channel #31 Return Address Storage |
| 50 | SKS | 20010 | Error ? |
| 51 | BRU | ERROR | Yes; Branch to Error Program |
| 52 | EOM | 00000 | No; Disconnect |
| 53 | BRU I | 47 | Return to Main Program |

2. Program

| Location | Instruction | Remarks |
|---|---|---|
| 54 | MIW 2000 | Initial Address |
| 55 | OCT00000143 | Word Count -1 |
| 56 | OCT00000000 | Temporary Storage |

Storage - $23_{10}$ words

Timing - (152 + 96i) μsec where i = number of words
In this case, where i = 100, computation time = 12,952 μsec
Maximum transfer rate - 41.6 kc
At 15 kc, 64% of time available for other computation

3. Explanation

The program is first entered at location 31 where the starting address and number of words to be transferred are initialized. At location 36, Magnetic Tape Unit No. 1 is started and control is returned to the main program at location 37. Each time the W Buffer is ready to receive a word, Priority Interrupt Channel #30 is energized and the instruction in location 30 is next executed. At location 30, the P Register is stored in location 40 and a branch to location 41 occurs where the output sequence begins. The memory word is transferred to the W Register, the memory address is increased by one, and the word count is reduced by one. The A, B, and X Registers are not changed by these instructions. When a total of 100 words have been transferred, a skip occurs at location 43 and the W Buffer is signaled that the last word has been transmitted.

When the last word has been completely written on Magnetic Tape Unit No. 1, Priority Interrupt Channel #31 is energized and the instruction in location 31 is next executed. The Error Indicator is then checked. If an error has occurred during the write operation, the program branches to an Error Routine (not shown) where corrective action is taken.

Since Priority Interrupt Channel #30 is of a higher priority than Channel #31, an interrupt from Channel #31 cannot take place while a Channel #30 interrupt is being obeyed. The branch instructions in location 44 or 46 automatically clear Priority Interrupt Channel #30.

## C. EXAMPLE NO. 3 (ANALOG INPUT SCANNING)

### 1. Problem

A 12-bit Analog-to-Digital Converter and Multiplexer are connected to the SDS 920. Bring into the computer the digital outputs of the converter and check against upper and lower limits. If within limits, convert to engineering units and store the data. The conversion formula is ax + b, where a and b are unique conversion coefficients and x is the data.

Assume that there are 64 converter inputs. All related information is located in corresponding positions of tables, i.e., Upper Limits (2000 - 2077), Lower Limits (2100 - 2177), "a" coefficients (2200 - 2277), "b" coefficients (2300 - 2377), Converted Data Storage (2400 - 2477) and Raw Input Storage (2500 - 2577).

The program starts at location 1000.

### 2. Program

| Location | Instruction | | Remarks |
|---|---|---|---|
| 1000 | LDX | 1016 | Set Index with $-100_8$ |
| 1001 | EOM | 30001 | Step Multiplexer, Start conversion |
| 1002 | 1 PIN | 2600 | ith Raw Data → Storage |
| 1003 | 1 LDA | 2100 | Upper Limit → A |
| 1004 | 1 SKG | 2600 | Upper Limit → Raw Data? |
| 1005 | BRU | OUT | No. -to Out of Limits Routine |
| 1006 | 1 LDA | 2600 | Yes. Raw Data → A |
| 1007 | 1 SKG | 2200 | Raw Data → Lower Limit |
| 1010 | BRU | OUT | No - to Out of Limits Routine |
| 1011 | 1 MUL | 2300 | a(x) |
| 1012 | 1 ADD | 2400 | ax + b |
| 1013 | 1 STA | 2500 | Converted Data → Storage |
| 1014 | BRX | 1001 | X + 1 → X; Complete? |
| 1015 | BRU | REST | Yes - Proceed |
| 1016 | OCT77777700 | | Complement of 100 |

Storage - $15_{10}$ words

Timing - (24 + 280i) μsec where i = no. of points

2. Proaram

3. Explanation

Since all storage is related to the data point being sampled, the Index Register can be used to gain access to all locations. As the Index Register is changed, a different set of data becomes available.

ENERGIZE OUTPUT M (02) in location 1001 performs the dual function of stepping the Multiplexer and starting the conversion. By proper placement of this instruction, processing of the ith point may be performed in parallel with the conversion of the i + 1 point. Therefore, conversion time is not a factor in calculating program time.

## D. EXAMPLE NO. 4 (BREAKPOINT TEST)

1. Problem

Read the Breakpoint Switches and test as follows:

| Breakpoint Configuration | Result |
|---|---|
| 1111 | Branch to 1000 |
| 0111 | Branch to 1020 |
| 1011 | Branch to 1040 |
| 1101 | Branch to 1060 |
| Rest | Branch to 2000 |

2. Program

| Location | Instruction | | Remarks |
|---|---|---|---|
| L | SKS | 20740 | All on ? |
| | BRU | 1000 | Yes - exit |
| L + 2 | SKS | 20440 | No; #1 and #4 on ? |
| | 2 BRU | +4 | Yes |
| L + 4 | SKS | 20340 | No; #2, #3, and #4 on ? |
| | BRU | 1020 | Yes - exit |
| | BRU | 2000 | No - exit |

| Location | Instruction | | Remarks |
|---|---|---|---|
| | SKS | 20100 | #3 on ? |
| | BRU | 1040 | Yes - exit |
| | BRU | 1060 | No; #4 on - exit |

Storage - $10_{10}$ words

Timing - 16 - 56 μsec

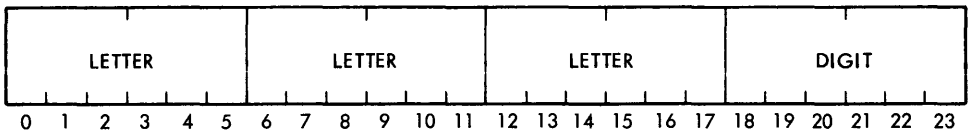
3. Explanation

The Breakpoint Switches may be tested in "or" combinations (see Appendix D). Use was made of this feature to test for all switches ON (at location L), and for other combinations (at locations L + 2 and L + 4).

## E. EXAMPLE NO. 5 (TABLE SEARCH)

1. Problem

A table of data is contained in memory where each word is of the following form:




The table is arranged so that all entries with the same alphabetic characters are grouped together, ordered numerically. Alphabetical order is not maintained.

A subroutine is required which enters new information into this table. The new word, in the above form, is in the A Register upon entry. The subroutine is required to search the table for the correct storage location, store the new entry, and precess the remainder of the words one position.

2. Program

| Location | Instruction | Remarks |
|---|---|---|
| L | EXIT | |
| | 2 LDB | Mask → B |
| | 2 LDX | "Length" → X |
| L + 3 | 1 SKM BEGIN | Letters Identical |
| | 2 BRU +11 | No; Branch to modify X |
| L + 5 | 1 SKG BEGIN | Yes; New > Old ? |
| | 2 BRU +4 | Yes |
| | 2 BRX -2 | No, X + 1 → X, End ? |
| L + 10 | 1 STA BEGIN | Yes; Store Entry |
| | 2 BRR -11 | Exit |
| L + 12 | 1 XMA BEGIN | Precess Table |
| | 2 BRX -1 | X + 1 → X; End ? |
| | 2 BRU -4 | Yes; Store last entry and exit |
| | 2 BRX -12 | X + 1 → X; End? |
| | 2 BRU -6 | Yes; Go to store |
| | OCT 77777700 | Letter Mask |
| | OCT LENGTH | Value for length of table |

Storage - $17_{10}$ words

Timing - (72 + 32i) μsec where i = table length

3. Explanation

At location L + 3, the table is searched for letter agreement. If agreement is not found, the new entry is stored at the end of the table and EXIT takes place. When letter agreement is found, the subroutine enters a numerical comparison at location L + 5. The word is then exchanged with the table word it is to replace at L + 12 and the remainder of the table precessed by one.

# VI. Appendices

# Appendix A

## SDS 920 INSTRUCTION LIST - NUMERICAL ORDER

| Instr. Code | Designation | Name | Function | Timing |
|---|---|---|---|---|
| 00 | HLT | HALT | Halts Computation | 1 |
| 01 | BRU | BRANCH UNCONDITIONALLY | M → P | 1 |
| 02 | EOM | ENERGIZE OUTPUT M | 8 μsec pulse to Point(s) addressed | 1 |
| 10 | MIY | M INTO Y BUFFER WHEN READY | (M) → Y | 2 + wait |
| 12 | MIW | M INTO W BUFFER WHEN READY | (M) → W | 2 + wait |
| 13 | POT | PARALLEL OUTPUT | (M) → Unit M in Parallel | 2 + wait |
| 14 | ETR | EXTRACT | (A) and (M) → A | 2 |
| 16 | MRG | MERGE | (A) or (M) → A | 2 |
| 17 | EOR | EXCLUSIVE OR | $(M)\overline{(A)}$ or $\overline{(M)}(A)$ → A | 2 |
| 20 | NOP | NO OPERATION | ---- | 1 |
| 23 | EXU | EXECUTE | Instr. M is performed, P unchanged | 1 |
| 30 | YIM | Y BUFFER INTO M WHEN READY | (Y) → M | 3 + wait |
| 32 | WIM | W BUFFER INTO M WHEN READY | (W) → M | 3 + wait |
| 33 | PIN | PARALLEL INPUT | (Unit M) → M in Parallel | 3 + wait |
| 34 | STX | STORE INDEX | (X) → M | 3 |
| 35 | STA | STORE A | (A) → M | 3 |
| 36 | STB | STORE B | (B) → M | 3 |
| 40 | SKS | SKIP IF SIGNAL NOT SET | If Signal = 1, P + 1 → P<br>If Signal = 0, P + 2 → P | 1, 2 |
| 41 | BRX | INCREMENT INDEX AND BRANCH | (X) + 1 → X<br>If X Neg., M → P<br>If X Pos., P + 1 → P | 1, 2 |
| 43 | BRM | MARK PLACE AND BRANCH | (P) → M; M + 1 → P | 2 |
| 4600001 | CLA | CLEAR A | "0" → A | 1 |
| 4600002 | CLB | CLEAR B | "0" → B | 1 |
| 4600003 | CLR | CLEAR AB | "0" → A, B | 1 |
| 4600004 | CAB | COPY A INTO B | (A) → B | 1 |

| Instr. Code | Designation | Name | Function | Timing |
|---|---|---|---|---|
| 4600010 | CBA | COPY B INTO A | (B) → A | 1 |
| 4600014 | XAB | EXCHANGE A AND B | (A) ⟷ (B) | 1 |
| 4600020 | CBX | COPY B INTO INDEX | (B) → X | 1 |
| 4600040 | CXB | COPY INDEX INTO B | (X) → B | 1 |
| 4600060 | XXB | EXCHANGE INDEX AND B | (X) ⟷ (B) | 1 |
| 4600122 | STE | STORE EXPONENT | $(B_{15-23}) \rightarrow X_{15-23}$<br>$0 \rightarrow B_{15-23}, X_{15} \rightarrow X_{0-14}$ | 1 |
| 4600140 | LDE | LOAD EXPONENT | $(X_{15-23}) \rightarrow B_{15-23}$ | 1 |
| 4600160 | XEE | EXCHANGE EXPONENTS | $(B_{15-23}) \longleftrightarrow (X_{15-23})$ | 1 |
| 4600200 | CXA | COPY INDEX INTO A | (X) → A | 1 |
| 4600400 | CAX | COPY A INTO INDEX | (A) → X | 1 |
| 4600600 | XXA | EXCHANGE INDEX AND A | (X) ⟷ (A) | 1 |
| 50 | SKE | SKIP IF A EQUALS M | If (A) ≠ (M), P + 1 → P<br>If (A) = (M), P + 2 → P | 2, 3 |
| 51 | BRR | RETURN BRANCH | (M) + 1 → P | 2 |
| 52 | SKB | SKIP IF M AND B DO NOT COMPARE ONES | If (B) (M) ≥ 1, P + 1 → P<br>(B) (M) = 0, P + 2 → P | 2, 3 |
| 53 | SKN | SKIP IF M NEGATIVE | If (M) ≥ 0, P + 1 → P<br>If (M) < 0, P + 2 → P | 2, 3 |
| 54 | SUB | SUBTRACT M FROM A | (A) - (M) → A | 2 |
| 55 | ADD | ADD M TO A | (A) + (M) → A | 2 |
| 56 | SUC | SUBTRACT WITH CARRY | (A) - (M) - Carry → A | 2 |
| 57 | ADC | ADD WITH CARRY | (A) + (M) + Carry → A | 2 |
| 60 | SKR | REDUCE M, SKIP IF NEGATIVE | (M) - 1 → M<br>If M Pos., P + 1 → P<br>If M Neg., P + 2 → P | 3 |
| 61 | MIN | MEMORY INCREMENT | (M) + 1 → M | 3 |
| 62 | XMA | EXCHANGE M AND A | (A) ⟷ (M) | 3 |
| 63 | ADM | ADD A TO M | (A) + (M) → M | 3 |
| 64 | MUL | MULTIPLY | (A) × (M) → A, B | 16 |
| 65 | DIV | DIVIDE | (A) ÷ (M) → A, R → B | 28 |

| | Instr. | | | |
|---|---|---|---|---|
| 66000XX | RSH | RIGHT SHIFT AB | AB Shift Right N Places | $2 + \frac{N}{2}$, Even<br>$2 + \frac{N+1}{2}$, Odd |
| 66200XX | RCY | RIGHT CYCLE AB | AB Cycled Right N Places | " |
| 67000XX | LSH | LEFT SHIFT AB | AB Shift Left N Places | " |
| 67100XX | NOD | NORMALIZE AND DECREMENT X | AB Left and X - 1 → X<br>Until $A_0 \neq A_1$ or N Shifts | " |
| 67200XX | LCY | LEFT CYCLE AB | AB Cycled Left N Places | " |
| 70 | SKM | SKIP IF A = M ON B MASK | If (B) (A) ≠ (B) (M), P + 1 → P<br>If (B) (A) = (B) (M), P + 2 → P | 2, 3 |
| 71 | LDX | LOAD INDEX | (M) → X | 2 |
| 72 | SKA | SKIP IF M AND A DO NOT COMPARE ONES | If (A) (M) ≥ 1, P + 1 → P<br>(A) (M) = 0, P + 2 → P | 2, 3 |
| 73 | SKG | SKIP IF A GREATER THAN M | If (A) ≤ (M), P + 1 → P<br>If (A) > (M), P + 2 → P | 2, 3 |
| 74 | SKD | DIFFERENCE EXPONENTS AND SKIP | $\lvert(B_{15-23}) - (M_{15-23})\rvert \rightarrow X_{15-23}$<br>If difference is positive, P + 1 → P<br>If difference is negative, P + 2 → P | 2, 3 |
| 75 | LDB | LOAD B | (M) → B | 2 |
| 76 | LDA | LOAD A | (M) → A | 2 |
| 77 | EAX | COPY EFFECTIVE ADDRESS INTO INDEX | Effective Address → X | 2 |

# Appendix B

SDS 920 INSTRUCTION LIST - OPERATIONAL CLASS

| Designation | Instr. Code | Name | Function | Timing |
|---|---|---|---|---|
| LOAD-STORE: | | | | |
| LDA | 76 | LOAD A | (M) → A | 2 |
| STA | 35 | STORE A | (A) → M | 3 |
| LDB | 75 | LOAD B | (M) → B | 2 |
| STB | 36 | STORE B | (B) → M | 3 |
| LDX | 71 | LOAD INDEX | (M) → X | 2 |
| STX | 34 | STORE INDEX | (X) → M | 3 |
| XMA | 62 | EXCHANGE M AND A | (A) ↔ (M) | 3 |
| ARITHMETIC: | | | | |
| ADD | 55 | ADD M TO A | (A) + (M) → A | 2 |
| ADC | 57 | ADD WITH CARRY | (A) + (M) + Carry → A | 2 |
| ADM | 63 | ADD A TO M | (A) + (M) → M | 3 |
| MIN | 61 | MEMORY INCREMENT | (M) + 1 → M | 3 |
| SUB | 54 | SUBTRACT M FROM A | (A) - (M) → A | 2 |
| SUC | 56 | SUBTRACT WITH CARRY | (A) - (M) - Carry → A | 2 |
| MUL | 64 | MULTIPLY | (A) x (M) → A, B | 16 |
| DIV | 65 | DIVIDE | (A) ÷ (M) → A, R → B | 28 |
| BRANCH-SKIP: | | | | |
| BRU | 01 | BRANCH UNCONDITIONALLY | M → P | 1 |
| BRX | 41 | INCREMENT INDEX AND BRANCH | (X) + 1 → X<br>If X Neg., M → P<br>If X Pos., P + 1 → P | 1, 2 |
| BRM | 43 | MARK PLACE AND BRANCH | (P) → M; M + 1 → P | 2 |
| BRR | 51 | RETURN BRANCH | (M) + 1 → P | 2 |
| SKS | 40 | SKIP IF SIGNAL NOT SET | If Signal = 1, P + 1 → P<br>If Signal = 0, P + 2 → P | 1, 2 |

| Designation | Instr. Code | Name | Function | Timing |
|---|---|---|---|---|
| BRANCH-SKIP: | | | | |
| SKE | 50 | SKIP IF A EQUALS M | If (A) $\neq$ (M), P + 1 → P<br>If (A) = (M), P + 2 → P | 2, 3 |
| SKG | 73 | SKIP IF A GREATER THAN M | If (A) $\leq$ (M), P + 1 → P<br>If (A) > (M), P + 2 → P | 2, 3 |
| SKR | 60 | REDUCE M, SKIP IF NEGATIVE | (M) - 1 → M<br>If (M) Pos., P + 1 → P<br>If (M) Neg., P + 2 → P | 3 |
| SKM | 70 | SKIP IF A = M ON B MASK | If (B)(A) $\neq$ (B)(M), P + 1 → P<br>IF (B)(A) = (B)(M), P + 2 → P | 2, 3 |
| SKN | 53 | SKIP IF M NEGATIVE | If (M) $\geq$ 0, P + 1 → P<br>If (M) < 0, P + 2 → P | 2, 3 |
| SKA | 72 | SKIP IF M AND A DO NOT COMPARE ONES | If (A)(M) $\geq$1, P + 1 → P<br>If (A)(M) = 0, P + 2 → P | 2, 3 |
| SKB | 52 | SKIP IF M AND B DO NOT COMPARE ONES | If (B)(M) $\geq$ 1, P + 1 → P<br>If (B)(M) = 0, P + 2 → P | 2, 3 |
| SKD | 74 | DIFFERENCE EXPONENTS AND SKIP | $\lvert (B_{15-23}) - (M_{15-23}) \rvert \rightarrow X_{15-23}$<br>If Difference is Pos., P + 1 → P<br>If Difference is Neg., P + 2 → P | 2, 3 |
| LOGICAL: | | | | |
| ETR | 14 | EXTRACT | (A) and (M) → A | 2 |
| MRG | 16 | MERGE | (A) or (M) → A | 2 |
| EOR | 17 | EXCLUSIVE OR | (M)$(\overline{A})$ or $(\overline{M})$(A) → A | 2 |
| REGISTER CHANGE: | | | | |
| CLA | 4600001 | CLEAR A | "0" → A | 1 |
| CLB | 4600002 | CLEAR B | "0" → B | 1 |
| CLR | 4600003 | CLEAR AB | "0" → A, B | 1 |
| CAB | 4600004 | COPY A INTO B | (A) → B | 1 |
| CBA | 4600010 | COPY B INTO A | (B) → A | 1 |
| XAB | 4600014 | EXCHANGE A AND B | (A) $\longleftrightarrow$ (B) | 1 |
| CXA | 4600200 | COPY INDEX INTO A | (X) → A | 1 |
| CAX | 4600400 | COPY A INTO INDEX | (A) → X | 1 |
| XXA | 4600600 | EXCHANGE INDEX AND A | (X) $\longleftrightarrow$ (A) | 1 |
| CBX | 4600020 | COPY B INTO INDEX | (B) → X | 1 |

| Designation | Instr. Code | Name | Function | Timing |
|---|---|---|---|---|
| REGISTER CHANGE: | | | | |
| CXB | 4600040 | COPY INDEX INTO B | $(X) \rightarrow B$ | 1 |
| XXB | 4600060 | EXCHANGE INDEX AND B | $(X) \longleftrightarrow (B)$ | 1 |
| STE | 4600122 | STORE EXPONENT | $(B_{15-23}) \rightarrow X_{15-23}$<br>$0 \rightarrow B_{15-23} X_{15} \rightarrow X_{0-14}$ | 1 |
| LDE | 4600140 | LOAD EXPONENT | $(X_{15-23}) \rightarrow B_{15-23}$ | 1 |
| XEE | 4600160 | EXCHANGE EXPONENTS | $(B_{15-23}) \longleftrightarrow (X_{15-23})$ | 1 |
| SHIFT: | | | | |
| RSH | 66000XX | RIGHT SHIFT AB | AB Shift Right N Places | Even, $2 + \frac{N}{2}$<br>Odd, $2 + \frac{N+1}{2}$ |
| RCY | 66200XX | RIGHT CYCLE AB | AB Cycled Right N Places | " |
| LSH | 67000XX | LEFT SHIFT AB | AB Shift Left N Places | " |
| LCY | 67200XX | LEFT CYCLE AB | AB Cycled Left N Places | " |
| NOD | 67100XX | NORMALIZE AND DECREMENT X | AB Left and X - 1 → X<br>Until $A_0 \neq A_1$, or N Shifts | " |
| CONTROL: | | | | |
| HLT | 00 | HALT | Halts Computation | 1 |
| NOP | 20 | NO OPERATION | ---- | 1 |
| EXU | 23 | EXECUTE | Instruction M is performed, P unchanged | 1 |
| EAX | 77 | COPY EFFECTIVE ADDRESS INTO INDEX | Effective Address → X | 2 |
| INPUT/OUTPUT: | | | | |
| MIW | 12 | M INTO W BUFFER WHEN READY | (M) → W | 2 + wait |
| WIM | 32 | W BUFFER INTO M WHEN READY | (W) → M | 3 + wait |
| MIY | 10 | M INTO Y BUFFER WHEN READY | (M) → Y | 2 + wait |
| YIM | 30 | Y BUFFER INTO M WHEN READY | (Y) → M | 3 + wait |
| POT | 13 | PARALLEL OUTPUT | (M) → Unit M in Parallel | 2 + wait |
| PIN | 33 | PARALLEL INPUT | (Unit M) → M in Parallel | 3 + wait |
| EOM | 02 | ENERGIZE OUTPUT M | 8 μsec Pulse to Point(s) Addressed | 1 |

# Appendix C

## PROGRAMMED OPERATOR INSTRUCTIONS

A library of Programmed Operator subroutines is available to greatly extend the SDS 920 instruction list. A partial list of these subroutines is indicated below. Each subroutine is specified by a unique mnemonic code and represents an available instruction which may be used directly in preparing 920 programs. Up to 64 of these Programmed Operator instructions may be used in preparing any one program.

The program loading system automatically organizes the interconnection between the Programmed Operator instructions and the corresponding subroutines. Each Programmed Operator instruction mnemonic code is converted on input to an instruction code of 100 to 177. A memory location from 100 through 177 corresponding to each assigned instruction code is loaded with an unconditional branch to the corresponding subroutine. For example, if the input program contains "ADP," the Programmed Operator instruction for Add Double Precision, it may be assigned instruction code 107, and the ADP subroutine may be loaded in memory locations 426 through 435. If memory location 1001 contains the instruction ADP 3640, the flow of performing this instruction will be as follows:

| Location | Instruction Code | Instruction | Remarks |
|---|---|---|---|
| 000 | (00) | HLT I ---- | Return Address |
| 1001 | (107) | ADP 3640 | P.O. Add Double Precision |
| 1002 | | | |
| 107 | (01) | BRU 426 | Transfer to ADP Subroutine |
| 426 | (34) | STX 77 | X → Temporary Storage |
| 427 | (74) | EAX I 000 | Effective Address → X |
| 430 | (46) | XAB | A ⟷ B |
| 431 | (55) | 1 ADD 0 | Add Lower |
| 432 | (46) | XAB | A ⟷ B |
| 433 | (57) | 1 ADC 1 | Add Upper |
| 434 | (71) | LDX 77 | Temporary Storage → X |
| 435 | (51) | BRR 000 | Return |

When the 920 Computer detects the instruction code of 107 for the instruction in location 1001, control is automatically transferred to location 107 and the contents of the program counter are automatically stored in location 000. Location 000 also retains an indirect address bit and is changed to:

| Location | Instruction | Remarks |
|---|---|---|
| 000 | HLT I 1001 | Return to 1001 |

The instruction in 107 then transfers control directly to the ADP subroutine. After the contents of the X Register are placed in temporary storage, the EAX 1000 instruction in 427 loads the X Register with the effective address of the Programmed Operator instruction 3640. The required instructions are then performed, based on this address, and the original contents of the X Register are restored. The BRR 000 instruction in 435 then provides a direct transfer of control to location 1002 through the return address in location 000.

Each time the instruction code 107 is detected, indicating an ADP instruction, the ADP subroutine will be entered and performed in the same way. If an ADP instruction is indexed or contains an indirect address, the EAX I 000 instruction will extract the correct effective address.

## SDS 920 PROGRAMMED OPERATOR INSTRUCTIONS

| Mnemonic Code | Name | Function |
|---|---|---|
| ADP | Add Double Precision | (A,B) + (M,M+1) → A,B |
| ASD | ARC SIN - Double Precision | ASIN (A,B) → A,B |
| ASF | ARC SIN - Floating | Floating ASIN (A,B) → A,B |
| ASN | ARC SIN of A | ASIN (A) → A |
| ATD | ARC TAN - Double Precision | ATAN (A,B) → A,B |
| ATF | ARC TAN - Floating | Floating ATAN (A,B) → A,B |
| ATN | ARC-TAN of A | ATAN (A) → A |
| BDF | Binary to Decimal - Floating | Floating $(A,B)_{Binary}$ → $A,B_{Decimal}$ (8 digits)<br>Exponent in X (3 digits) |
| BID | Binary to Decimal in A | $(A)_{Binary}$ → $A,B_{Decimal}$<br>Binary Point at M |
| COS | COS of A | COS (A) → A |
| CSD | COS - Double Precision | COS (A,B) → A,B |
| CSF | COS- Floating | Floating COS (A,B) → A,B |
| DBD | Decimal to Binary-Double Precision | $(A,B)_{Decimal}$ → $A,B_{Binary}$<br>Decimal Point at M |
| DBF | Decimal to Binary - Floating | $(A,B)_{Decimal}$ → Floating $A,B_{Binary}$<br>Decimal Point at M |
| DDP | Divide, Double Precision | (A,B) ÷ (M,M+1) → A,B |
| DIB | Decimal to Binary in A | $(A)_{Decimal}$ → $A_{Binary}$<br>Decimal Point at M |
| EXD | Exponential-Double Precision | $e^{(A,B)}$ → A,B |
| EXF | Exponential - Floating | Floating $e^{(A,B)}$ → A,B |
| EXP | Exponential of A | $e^{(A)}$ → A |
| FFL | Fixed Point to Floating Point | (A,B) Fixed → (A,B) Floating<br>Binary Point at bit M |
| FLA | Floating Add | Floating (A,B) + (M,M+1) → A,B |
| FLD | Floating Divide | Floating (A,B) ÷ (M,M+1) → A,B |
| FLF | Floating Point to Fixed Point | (A,B) Floating → (A,B) Fixed Binary Point at bit M |
| FLM | Floating Multiply | Floating (A,B) x (M,M+1) → A,B |
| FLS | Floating Subtract | Floating (A,B) - (M,M+1) → A,B |
| FSA | Floading Add, Single Precision | Floating (A) + (M) → A<br>Exponent in B, M + 1 |

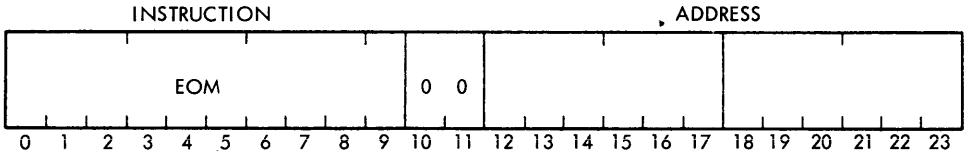
| Mnemonic Code | Name | Function |
|---|---|---|
| FSD | Floating Divide, Single Precision | Floating (A) $\div$ (M) $\rightarrow$ A<br>Exponent in B, M + 1 |
| FSM | Floating Multiply, Single Precision | Floating (A) x (M) $\rightarrow$ A<br>Exponent in B, M + 1 |
| FSS | Floating Subtract, Single Precision | Floating (A) - (M) $\rightarrow$ A<br>Exponent in B, M + 1 |
| LDP | Load Double Precision | (M, M+1) $\rightarrow$ A, B |
| LGD | LOG - Double Precision | $LOG_M$ (A, B) $\rightarrow$ A, B |
| LGF | LOG-Floating | Floating $LOG_M$ (A, B) $\rightarrow$ A, B |
| LOG | LOG of A | $LOG_M$ (A) $\rightarrow$ A |
| MDP | Multiply, Double Precision | (A, B) x (M, M+1) $\rightarrow$ A, B |
| SDP | Subtract Double Precision | (A, B) - (M, M + 1) $\rightarrow$ A, B |
| SIN | SIN of A | SIN (A) $\rightarrow$ A |
| SND | SIN - Double Precision | SIN (A, B) $\rightarrow$ AB |
| SNF | SIN - Floating | Floating SIN (A, B) $\rightarrow$ A, B |
| SQD | Square Root - Double Precision | $\sqrt{(A, B)}$ $\rightarrow$ A, B |
| SQF | Square Root - Floating Point | Floating $\sqrt{A, B}$ $\rightarrow$ A, B |
| SQR | Square Root of A | $\sqrt{(A)}$ $\rightarrow$ A |
| STD | Store Double Precision | (A, B) $\rightarrow$ M, M + 1 |

# Appendix D

## INPUT/OUTPUT ADDRESS CODES

Input/Output Address Codes are used when (a) connecting external units to the Input/Output Buffer (b) controlling external units (c) sensing the status of external units (d) setting and sensing internal computer conditions and (e) setting and sensing internal indicators. Setting is performed using ENERGIZE OUTPUT M (02) while sensing is performed using SKIP IF SIGNAL NOT SET (40). The address portion of these instructions comprises the Input/Output Address.

ENERGIZE OUTPUT M (02) is used to control the Input/Output Buffer. The address portion of this instruction is interpreted as follows:




| Bit Positions | Function | |
|---|---|---|
| 12 | 1 = Reverse | 0 = Forward |
| 13 | 1 = Begin output without leader<br>0 = Begin output with leader | |
| 14 | 1 = Start | 0 = Select |
| 15 | 1 = Y Buffer | 0 = W Buffer |
| 16, 17 | Character Count;<br>01 = 2 characters,<br>11 = 4 characters | 00 = 1 character<br>10 = 3 characters, |
| 18-23 | Unit Address Codes (see below) | |

Examples:

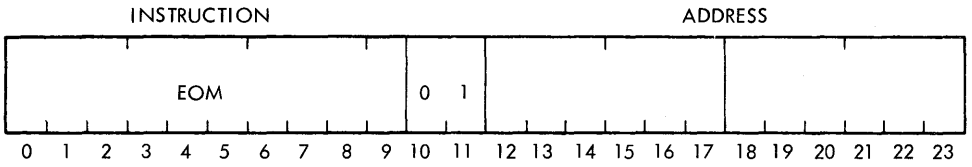
| Instruction | Function |
|---|---|
| EOM 01304 | Start Paper Tape Reader #1, set W Buffer to accept input data and assemble 4 characters per word. |
| EOM 03341 | Start Typewriter #1, output 4 characters per word |
| EOM 01344 | Start Paper Tape Punch #1, automatically generate 6 leader frames; output 4 characters per word. |
| EOM 03344 | Start Paper Tape Punch #1; do not generate leader, output 4 characters per word. |

## UNIT ADDRESS CODES

| | | | |
|---|---|---|---|
| 00 | Disconnect | 40 | |
| 01 | Type Input #1 | 41 | Type Output #1 |
| 02 | Type Input #2 | 42 | Type Output #2 |
| 03 | Type Input #3 | 43 | Type Output #3 |
| 04 | Paper Tape Input #1 | 44 | Paper Tape Punch Output #1 |
| 05 | Paper Tape Input #2 | 45 | Paper Tape Punch Output #2 |
| 06 | Card Reader Input #1 | 46 | Card Punch Output #1 |
| 07 | Card Reader Input #2 | 47 | Card Punch Output #2 |
| 10 | Magnetic Tape Input #1 | 50 | Magnetic Tape Output #1 |
| 11 | Magnetic Tape Input #2 | 51 | Magnetic Tape Output #2 |
| 12 | Magnetic Tape Input #3 | 52 | Magnetic Tape Output #3 |
| 13 | Magnetic Tape Input #4 | 53 | Magnetic Tape Output #4 |
| 14 | Magnetic Tape Input #5 | 54 | Magnetic Tape Output #5 |
| 15 | Magnetic Tape Input #6 | 55 | Magnetic Tape Output #6 |
| 16 | Magnetic Tape Input #7 | 56 | Magnetic Tape Output #7 |
| 17 | Magnetic Tape Input #8 | 57 | Magnetic Tape Output #8 |
| 20 | - | 60 | High Speed Printer Output #1 |
| 21 | - | 61 | High Speed Printer Output #2 |
| 22 | - | 62 | - |
| 23 | - | 63 | - |
| 24 | - | 64 | Incremental Plotter Output #1 |
| 25 | - | 65 | Incremental Plotter Output #2 |
| 26 | Disc File Input #1 | 66 | Disc File Output #1 |
| 27 | Disc File Input #2 | 67 | Disc File Output #2 |
| 30 | Scan Magnetic Tape #1 | 70 | Magnetic Tape Erase #1 |
| 31 | Scan Magnetic Tape #2 | 71 | Magnetic Tape Erase #2 |
| 32 | Scan Magnetic Tape #3 | 72 | Magnetic Tape Erase #3 |
| 33 | Scan Magnetic Tape #4 | 73 | Magnetic Tape Erase #4 |
| 34 | Scan Magnetic Tape #5 | 74 | Magnetic Tape Erase #5 |
| 35 | Scan Magnetic Tape #6 | 75 | Magnetic Tape Erase #6 |
| 36 | Scan Magnetic Tape #7 | 76 | Magnetic Tape Erase #7 |
| 37 | Scan Magnetic Tape #8 | 77 | Magnetic Tape Erase #8 |

## INPUT/OUTPUT UNIT CONTROL

Special control instructions to Input/Output Units are performed using ENERGIZE OUTPUT M (02). The address portion of this instruction is interpreted as follows:



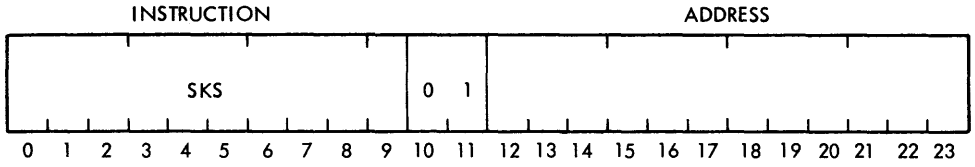

| Bit Positions | Function |
|---|---|
| 12 | 1 = Control Function #1 |
| 13 | 1 = Control Function #2 |
| 14 | 1 = Control Function #3 |
| 15 | 1 = Control Function #4 |
| 16 | 1 = Y Buffer Control |
| 17 | 1 = W Buffer Control |
| 18-23 | Unit Address Code<br>(See Input/Output Buffer Control) |

Examples:

| Instruction | Function |
|---|---|
| EOM 10100 | Terminate output connected to W Buffer |
| EOM 14010 | Rewind Magnetic Tape Unit #1 |

## INPUT/OUTPUT UNIT TEST

The status of Input/Output Units is tested using SKIP IF SIGNAL NOT SET (40). The address portion of the instruction is interpreted as follows:



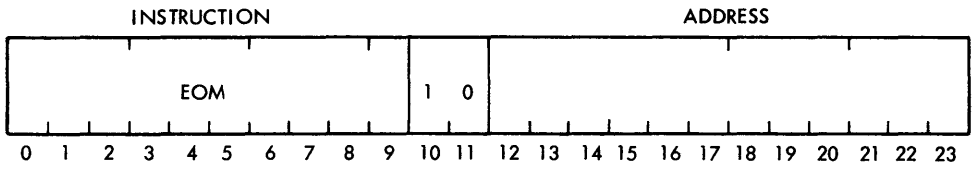

| Bit positions | Function |
|---|---|
| 12 | 1 = Skip if signal No. 1 not set |
| 13 | 1 = Skip if signal No. 2 not set |
| 14 | 1 = Skip if signal No. 3 not set |
| 15 | 1 = Skip if signal No. 4 not set |
| 16 | 1 = Skip if signal No. 5 not set |
| 17 | 1 = Skip if signal No. 6 not set |
| 18-23 | Unit Address Codes<br>(See Input/Output Buffer Control) |

Examples:

| Instruction | Function |
|---|---|
| SKS 10101 | Skip if Typewriter #1 off |
| SKS 10110 | Skip if Magnetic Tape Unit #1 off |
| SKS 10210 | Skip if Magnetic Tape Unit #1 File Protect off |

## SETTING COMPUTER CONDITIONS

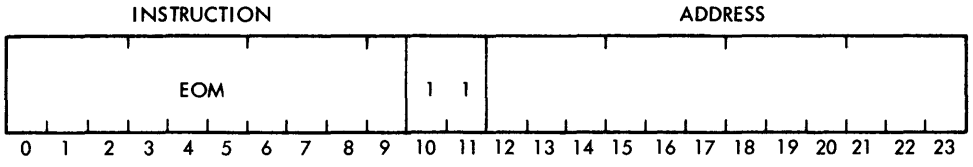
Internal computer conditions are set using ENERGIZE OUTPUT M (02). The address portion of this instruction is interpreted as follows:




| Bit Positions | Function | Instruction |
|---|---|---|
| 12-20 | Unassigned | |
| 21 | Disable Interrupt System | EOM 20004 |
| 22 | Enable Interrupt System | EOM 20002 |
| 23 | Turn OffOverflow | EOM 20001 |

## SETTING SYSTEM CONTROL INDICATORS

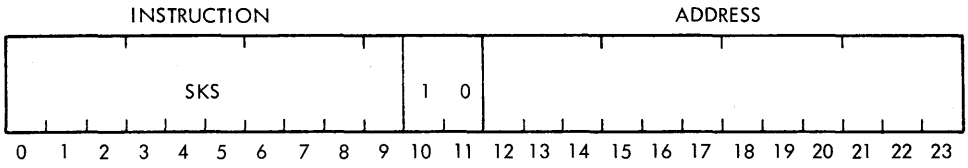
System Control Indicators are set using ENERGIZE OUTPUT M (02). The address portion of this function is interpreted as follows:




| Bit Positions | Function |
|---|---|
| 12-23 | Assigned as required |

## SENSING COMPUTER CONDITIONS

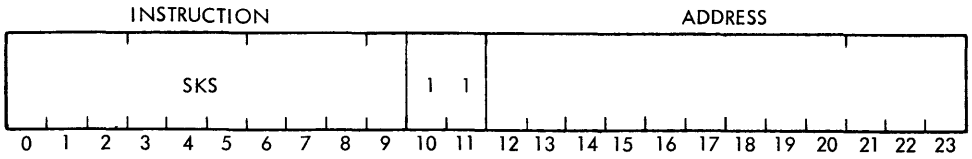
Computer Conditions are sensed using SKIP IF SIGNAL NOT SET (40). The address portion of this instruction is interpreted as follows:




| Bit Position | Function | Instruction |
|---|---|---|
| 12 | Unassigned | |
| 13 | Skip if Y Buffer Ready | SKS 22000 |
| 14 | Skip if W Buffer Ready | SKS 21000 |
| 15 | Skip if Breakpoint #1 off | SKS 20400 |
| 16 | Skip if Breakpoint #2 off | SKS 20200 |
| 17 | Skip if Breakpoint #3 off | SKS 20100 |
| 18 | Skip if Breakpoint #4 off | SKS 20040 |
| 19 | Skip if Y Buffer Error off | SKS 20020 |
| 20 | Skip if W Buffer Error off | SKS 20010 |
| 21 | Skip if Interrupt Enabled | SKS 20004 |
| 22 | Skip if Interrupt Disabled | SKS 20002 |
| 23 | Skip if Overflow off and Turn Off Overflow | SKS 20001 |

## SENSING EXTERNAL SYSTEM INDICATORS

External System Indicators are sensed using SKIP IF SIGNAL NOT SET (40). The Address portion of this instruction is interpreted as follows:




| Bit Positions | Function |
|---|---|
| 12-23 | Assigned as required |

# Appendix E

## TYPEWRITER CODES

| Character | Code | Character | Code |
|---|---|---|---|
| A | 01 | Z | 51 |
| B | 02 | 0 | 60 |
| C | 03 | 1 | 61 |
| D | 04 | 2 | 62 |
| E | 05 | 3 | 63 |
| F | 06 | 4 | 64 |
| G | 07 | 5 | 65 |
| H | 10 | 6 | 66 |
| I | 11 | 7 | 67 |
| J | 21 | 8 | 70 |
| K | 22 | 9 | 71 |
| L | 23 | ≠ | 40 |
| M | 24 | ! | 20 |
| N | 25 | ? | 00 |
| O | 26 | # | 73 |
| P | 27 | / | 41 |
| Q | 30 | , | 53 |
| R | 31 | $ | 33 |
| S | 42 | . | 13 |
| T | 43 | Space | 12 |
| U | 44 | Carriage Return | 17 |
| V | 45 | Upper Case | 15 |
| W | 46 | Lower Case | 14 |
| X | 47 | Tab | 32 |
| Y | 50 | Back Space | 37 |

# NOTES

SDS
SCIENTIFIC DATA SYSTEMS

1542 Fifteenth Street ▫ Santa Monica, California ▫ Upton 0—5471

SDS 5002