Section 2

The SDS 910-9300 series, a planned family

The Scientific Data System 900-9000 series consists of the SDS 910, 920, 925, 930, 940, 945, and 9300 computers. The series includes capabilities and features found in most 24-bit machines. The design implementation is among the best for 24-bit machines, as measured by equipment utilization, the processor state, implementation technology, and ease of use.

The first delivery dates for the members of the series are 910 (August, 1962), 920 (September, 1962), 925 (February, 1965), 930 (June, 1964), 940 (April, 1966), 945 (~1968), and 9300 (December, 1964).

The 910 and 920 were designed at the same time as a planned series of compatible computers which spanned a range of performance. The 910 has instructions which facilitate defining 920 instructions by software. For examole, these include the multiply and divide step[1] (see page 544) instructions in the 910 for programming the multiply and divide instruction in the 920.

The I/O facility evolved to a clean structure, with the potential for having a high degree of T and Ms data-transfer concurrency at a comparatively low cost. The IBM 7094 should be studied for a contrasting (more expensive) approach.

The instructions which help manipulate floating-point data are interesting and useful. The machine's ability to execute closed floating-point arithmetic subroutines is fairly good considering that the instructions are not hardwired.

The Programmed Operator (POP) instructions provide the ability to define an instruction set for efficient encoding. The idea appeared earlier in Atlas. However, the POP instruction calls subprograms in primary memory, instead of in fixed memory like Atlas.

A nice scheme[1] is described for increasing the memory address space from 16,384 to 32,768 words. Other schemes which switch memory banks, like those in the PDP-8 (Chap. 5) and in the 65,384-word 7094 II (Chap. 41), tend to be less desirable and flexible.

The SDS 930 was used at the University of California (Berkeley) as the base machine for the design of the Berkeley Time Sharing System (Chap. 24). SDS later marketed the system as the SDS 940.

The 9300 was not a member of the original 910-930 series. There is almost symbolic language program compatibility. Several registers and extra memory transfer paths were added to form the 9300 from the 930. The power of the 9300 is only a factor of 2 times the 930 for simple instructions. However, the hardwired floating-point instructions in the 9300 increases the power over the 930 by a factor of almost 10 for arithmetic problems. It is hard to believe that the incompatible 9300 was a wise choice. (We suggest a more reasonable alternative could have been a two-processor 930'. The 930' processor would be a 930 but with hardwired floating-point arithmetic instructions.) The 9300 has interesting twin-mode instructions for simultaneously operating on 12-bit data pairs. The 24-bit fixed-point word is sufficient for the real-time applications for which the computer was designed.

A flaw in the series is the sharing of K's among peripheral T's and Ms's. This problem can be seen by looking at the PMS structure (Chap. 42, Fig. 2, page 546). The connection to the peripheral K from K('Channel) requires a continuous connection during the data-transfer dialogue to Mp. This structure is especially bad in the case of a slow T, for example, a typewriter. A single character transmission requires that K('W, 'Y) be assigned to the typewriter during the complete message transmission (at a connected time of 100 milliseconds/character). The problem can be avoided by placing a character memory in each slow KT. Multiple devices could then run concurrently without requiring the elaborate K('W, 'Y) to be attached to them. The structure does not preclude such an improvement.

A complete description of the input/output and interrupt system is given and should be read carefully.

[1] We believe this appeared originally in the DEC PDP-1 introduced in November, 1960.

Chapter 42

The SDS[1] 910-9300 series

Introduction

The SDS 910, 920, 925, and 930 form a compatible series of computers. The 9300, though not compatible with the series, was an outgrowth of it. The 9300 uses the Ms and T devices of the 930. The 940 was designed initially at the University of California, Berkeley (see Chap. 24) for time sharing, and the 945 is a successor to the 940. The word length is 24 bits, and one single address instruction is encoded per word. The state of the machine consists of Mp(2048 ~ 32768 w) and Mps('P/Program Counter, 'A/Accumulator, 'B/Extended Accumulator, 'X/Index register).

These computers have been designed to process data originating from physical processes in real time. This design goal leads to a priority interrupt system with many (1,024) levels. The multiple interrupts facilitate programming and decrease the interrupt response time. A 24-bit word or two 12-bit words are a reasonable size for the problem types encountered. A multiple of 6 bits was chosen because of the (then) standard 6-bit magnetic-tape character. The relatively efficient storage representation and processing of floating-point data allow these computers to be used for general-purpose computation. However, only the 9300 has built-in floating-point operations. The 9300 has extensive capability for more general-purpose use. It is also used for operations on half-length data.

The data types processed by the 910-930 include words, integers, addresses, and boolean vectors. Several special instructions aid processing of types floating-point and double-length integers. The 9300 processes the additional data-types single- and double-length floating point. The 9300 has twin-mode instructions which operate on two half-length data (12 b) simultaneously. The two's complement representation is used for negative numbers.

The multiply, divide, and several other instructions are not wired into the 910, and compatibility between the 910 and 920-930 cannot be completely obtained by programming, although the 910 is a subset of the 920-930. Likewise, a smaller minimum Mp is available on the 910 (2,048 word versus 4,096 word). The 920 and 930 have identical instruction sets and differ in memory and logic performance. The 930 has a t.cycle: 1.75 μs, and the 910-920 has t.cycle: 8 μs. The more elaborate PMS structure of the 930 allows for greater growth, (e.g., by having more access ports to Mp).

The 9300's instruction set is different from the 930's. There are three index registers. The PMS structure is similar (and nearly compatible) with the 930. There are more (and better) working registers in the 9300 Pc to increase performance. The 9300 has two memory-access links, and the Pc can fetch instructions and data simultaneously. The instructions in the various C's appear in Table 1 for comparison purposes.

The SDS 925, a 1.75-μs version of the SDS 910, was available only for a brief time and will not be discussed further.

The machines process instructions (operations to the accumulator) in the following times (microseconds):

Instruction	*910*	*920*	*930*	*9300*
Fixed-Point Add	16	16	3.5	1.75
Fixed-Point Multiply	248	32	7.0	7.0
Floating-Point Add	896	384	92	14.0
Floating-Point Multiply	1696	656	147	12.25

Structure

The structure of these computers is given with PMS and conventional diagrams in Figs. 1 to 4.

The SDS channel is a Kio('Channel) and not a Pio, since it has no program counter and uses Pc. However, it can be as effective as a Pio. Of course, the cost is lower since Pc is shared. If K('W, 'Y) requires memory accesses, they must wait until suitable times in the Pc instruction-interpretation process to communicate with memory (Fig. 1).

The PMS structural detail (Fig. 2) does not show the algorithm by which simultaneous Kio('W, 'Y, 'C, 'D) and Pc requests for Mp are resolved. K has the highest priority, and further resolution among K's is determined by the K with the fullest buffer memory. Thus the priority is variable.

There are three basic K types, or channels (Fig. 2), in the 930 and 9300:

1 K('Time Multiplexed Communications Channel/TMCC)

2 K('Direct Access Communications Channel/DACC)

3 K('Data Subchannel/DSC)

[1] Scientific Data Systems merged with Xerox Corporation in 1969. The divisional name became Xerox Data Systems (XDS).

Table 1 SDS 910, 920, 930 and 9300 instruction sets[1]

Mnemonic		*Name*
LOAD/STORE		
+LDA	M, T	Load A
+STA	M, T	Store A
+LDB	M, T	Load B
+STB	M, T	Store B
LDP	M, T	Load Double Precision
STD	M, T	Store Double Precision
LDS	M, T	Load Selective (Masked)
STS	M, T	Store Selective (Masked)
+LDX	M, T	Load Index X
+STX	M, T	Store Index X
+EAX	M, T	Copy Effective address into Index Register 1
STZ	M	Store Zero
+XMA	M, T	Exchange M and A
XMB	M, T	Exchange M and B
XMX	M, T	Exchange M and Index Register
ARITHMETIC		
+ADD	M, T	Add M to A
DPA	M, T	Double Precision Add
+SUB	M, T	Subtract M from A
DPS	M, T	Double Precision Subtract
MPO	M, T	M Plus One
∘MIN	M, T	M Increment (M + 1)
MPT	M, T	M Plus Two
‡+ADM	M, T	Add to Memory
‡+MUL	M, T	Multiply
‡+DIV	M, T	Divide
TMU	M, T	Twin Multiply
DPN	M, T	Double Precision Negate
×MUS	M, T	Multiply Step
×DIS	M, T	Divide Step
‡∘SUC	M, T	Subtract with Carry
‡∘ADC	M, T	Add with Carry*
×∘MDE	M, T	M Decrement
ARITHMETIC, FLOATING-POINT (OPTIONAL)		
FLA	M, T	Floating Add
FLS	M, T	Floating Subtract
FLM	M, T	Floating Multiply
FLD	M, T	Floating Divide
LOGICAL		
+ETR	M, T	Extract
+MRG	M, T	Merge
REGISTER CHANGE		
RCH	M, T	Register Change
AXB	M, T	Address to Index Base
‡∘CLA		Clear A*
‡∘CLB		Clear B*
∘CLR		Clear AB
‡∘CAB		Copy A into B*
∘ABC		Copy A into B, Clear A
‡∘CBA		Copy B into A*
∘BAC		Copy B into A, Clear B
∘XAB		Exchange A and B
‡∘CBX		Copy B into Index*
‡∘CXB		Copy Index into B*
‡∘XXB		Exchange Index and B*
‡∘STE		Store Exponent*
‡∘LDE		Load Exponent*
‡∘XEE		Exchange Exponents*
‡∘CXA		Copy Index into A*
‡∘CAX		Copy A into Index*
∘XXA		Exchange Index and A*
‡∘CNA		Copy Negative into A*
∘CLX		Clear X
COPY		Copy
BRANCH		
+BRU	M, T	Branch Unconditionally
+BRX	M, T	Increase Index and Branch
+BRM	M, T	Mark Place and Branch
BRC	M, T	Branch and Clear Interrupt
BMA	M, T	Branch and Mark Place or Argument Address
+BRR	M, T	Return Branch
TEST/SKIP		
‡+SKE	M, T	Skip if A Equals M
+SKG	M, T	Skip if A Greater than M
SKL	M, T	Skip if A Less than M
+SKM	M, T	Skip if A equals M on B Mask
SKU	M, T	Skip if A Unequal M
SKQ	M, T	Skip if Masked Quantity in A Greater than M
SKF	M, T	Skip if Floating Exponent in B is Greater than or Equal
+SKA	M, T	Skip if A and M do not Compare Ones Anywhere
‡+SKB	M, T	Skip if B and M do Compare Ones Anywhere
+SKN	M, T	Skip if M is Negative
‡+SKR	M, T	Reduce M, Skip if Negative

Table 1 SDS 910, 920, 930 and 9300 instruction sets (Continued)

Mnemonic		*Name*
+EOR	M, T	Exclusive OR
REGISTER SHIFT		
SHIFT	M, T	Shift
ARSA	N, T	Arithmetic Right Shift A
ARSB	N, T	Arithmetic Right Shift B
∘RSH,		Arithmetic Right Shift AB
ARSD	N, T;	Arithmetic Right Shift Double
ARST	N, T	Arithmetic Right Shift Twin
LRSA	N, T	Logical Right Shift A
LRSB	N, T	Logical Right Shift B
∘LRSH	(930 only),	Logical Right Shift AB
LRSD	N, T;	Logical Right Shift Double
LRST	N, T	Logical Right Shift Twin
CRSA	N, T	Circular Right Shift A
CRSB	N, T	Circular Right Shift B
∘RCY,		Circular Right Shift AB
CRSD	N, T;	Circular Right Shift Double
CRST	N, T	Circular Right Shift Twin
∘LSH;		Arithmetic Left Shift AB
ALSA	N, T	Arithmetic Left Shift A
ALSB	N, T	Arithmetic Left Shift B
ALSD	N, T	Arithmetic Left Shift Double
ALST	N, T	Arithmetic Left Shift Twin
LLSA	N, T	Logical Left Shift A
LLSB	N, T	Logical Left Shift B
LLSD	N, T	Logical Left Shift Double
LLST	N, T	Logical Left Shift Twin
CLSA	N, T	Circular Left Shift A
CLSB	N, T	Circular Left Shift B
∘LCY,		Circular Left Shift AB
CLSD	N, T;	Circular Left Shift Double
CLST	N, T	Circular Left Shift Twin
NORA	N, T	Normalize A
∘NOD	N, T	Normalize; Decrement X
NORD	N, T;	Normalize Double
CONTROL		
+HLT		Halt
+NOP	M, T	No Operation
+EXU	M, T	Execute
INR	M, T	Interpret
REP	M, T	Repeat
SKP	M, T	Skip if Bit Sum Even
+SKS	M, T	Skip if Signal Not Set
∘SKD	M, T	Difference Exponents; Skip*
FLAG REGISTER		
FRTS	M	Flag Indicator Reset Test/Set
FLAG	M	Flag
FIRS	M	Flag Indicator Reset/Set
FSTR	M	Flag Indicator Set Test/Reset
FRST	M	Flag Indicator Reset/Set Test
SWT	M	SENSE Switch Test
INTERRUPTS		
+EIR		Enable Interrupts
+DIR		Disable Interrupts
+EIT		Interrupt Enabled Test
+IDT		Interrupt Disabled Test
+AIR		Arm Interrupts
MEMORY EXTENSION (930 ONLY)		
∘		Set Extension Register
∘		Extension Register Test
BREAKPOINT TESTS (SENSE SWITCHES IN 9300)		
∘BPT	4	Breakpoint No. 4 Test
∘BPT	3	Breakpoint No. 3 Test
∘BPT	2	Breakpoint No. 2 Test
∘BPT	1	Breakpoint No. 1 Test
OVERFLOW (FLAG IN 9300)		
∘ROV		Reset Overflow
∘REO		Record Exponent Overflow
∘OVT		Overflow Test; Reset
PROGRAMMED OPERATORS		
∘POP	M, T	Programmed Operator (64 instructions)

[1] M-Memory or Memory Address; N-number of shifts; T-tag field; +-also in the 910, 920 and 930; x-910 only; ∘-not in the 9300; ‡-not in the 910.

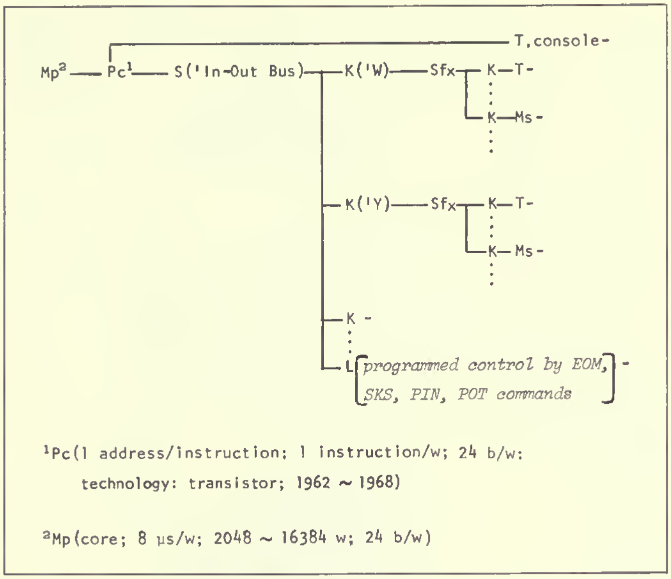

T.console-
Mp²
Pc¹
S('In-Out Bus)
K('W)
Sfx
K—T-
K—Ms-
K('Y)
Sfx
K—T-
K—Ms-
K -
L[programmed control by EOM, SKS, PIN, POT commands] -
¹Pc(1 address/instruction; 1 instruction/w; 24 b/w: technology: transistor; 1962 ~ 1968)
²Mp(core; 8 μs/w; 2048 ~ 16384 w; 24 b/w)

Fig. 1. SDS 910 and 920 PMS diagram.

The links between KT or KMs and any one of K('TMCC), K('DACC), and K('DSC) are identical. The KT or KMs assembles/disassembles characters into/from words and transmits/receives them to/from the Kio('Channel). The channel communicates with Mp or Pc for data transmission and finally communicates with Pc at task completion (the block of data transferred). Task alarms may cause Kio to interrupt Pc. Each Kio('Channel) can assemble data on a 6-, 12-, or 24-bit basis for Mp accesses. A K('Channel) recognizes two types of information: data being transmitted between Mp and the peripheral K, and initialization or controlling information from Pc.

In the 930 or 9300 K's the principal distinction is that the actual data-path switching routes differ. From a program operation and control viewpoint the Time Multiplexed and the Direct Access Communication Channels (TMCC and DACC) and the Data Subchannels (DSC) behave almost identically. The TMCC and DSC differ from DACC in that the block control information (number of words and location in memory) for the channel may be either in primary memory or in local hardware memory associated with the channel hardware.

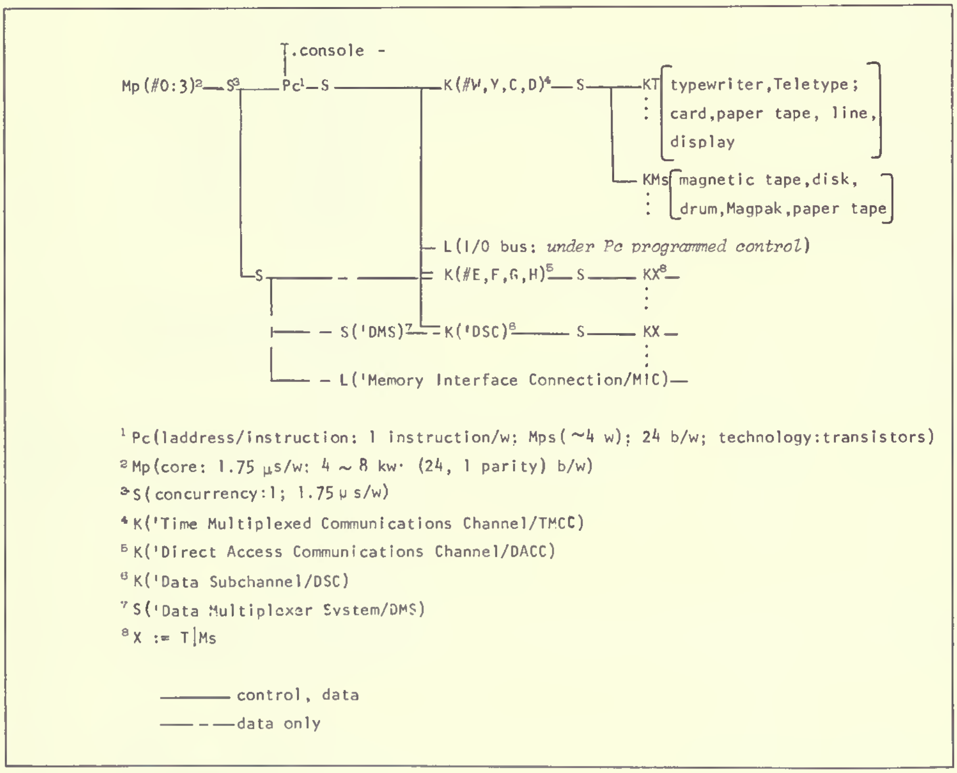

T.console -
Mp(#0:3)²
S³
Pc¹
S
K(#W,Y,C,D)⁴
S
KT[typewriter,Teletype; card,paper tape, line, display]
KMs[magnetic tape,disk, drum,Magpak,paper tape]
L(I/O bus: under Pc programmed control)
S
K(#E,F,G,H)⁵
S
KX⁸
S('DMS)⁷
K('DSC)⁶
S
KX
L('Memory Interface Connection/MIC)
¹Pc(1address/instruction: 1 instruction/w; Mps(~4 w); 24 b/w; technology:transistors)
²Mp(core: 1.75 μs/w: 4 ~ 8 kw· (24, 1 parity) b/w)
³S(concurrency:1; 1.75 μ s/w)
⁴K('Time Multiplexed Communications Channel/TMCC)
⁵K('Direct Access Communications Channel/DACC)
⁶K('Data Subchannel/DSC)
⁷S('Data Multiplexer System/DMS)
⁸X := T|Ms
control, data
data only

Fig. 2. SDS 930 PMS diagram.

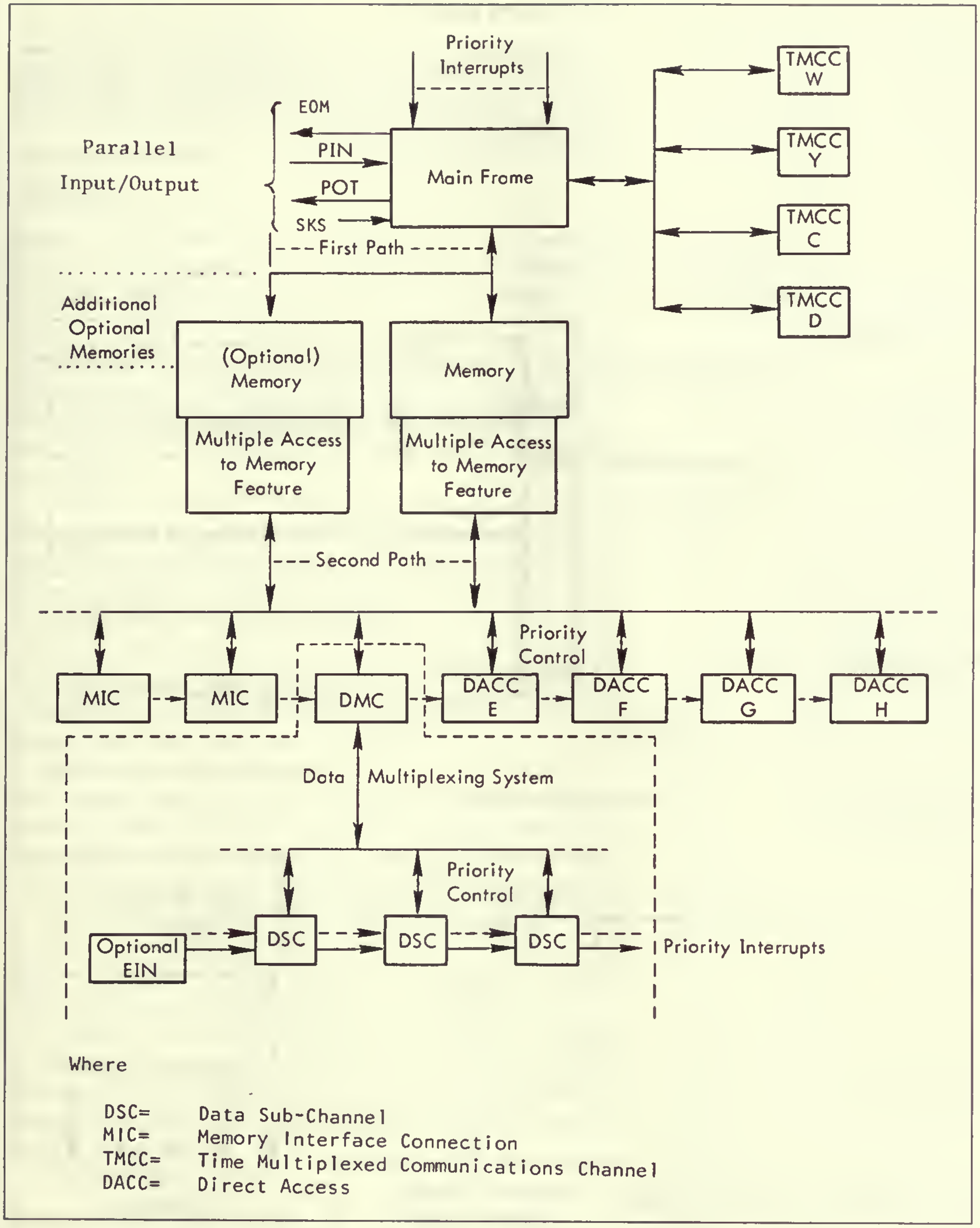


Fig. 3. SDS 390 computer-configuration diagram. (*Courtesy of Scientific Data Systems.*)

The 9300 structure, though not given in the PMS diagram, is essentially that of the 930 (Figs. 3 and 4). In the 9300, Mp has three access ports or a S('Memory-Processor; 8 Mp; 3 P,K). The Pc('9300) requires two of the access ports for independent access of instructions and data, leaving one for K transfer to Ms and T.

Instruction-set processor

The interesting parts of the ISP are discussed informally below. The formal ISP description given in Appendix I of this chapter should be read. The descriptions are partially taken from the SDS Programming Reference Manuals.

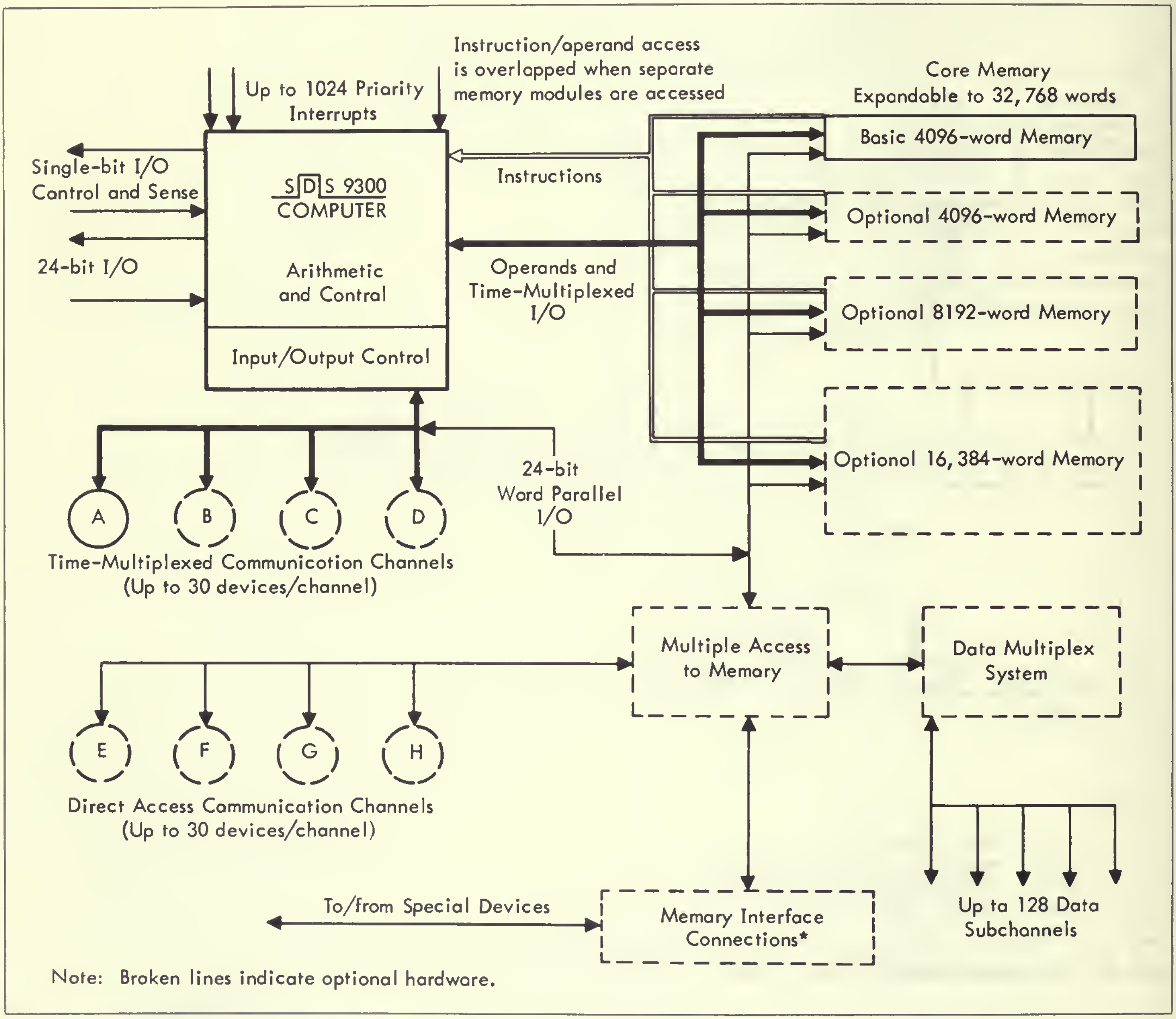

Up to 1024 Priority Interrupts
Instruction/operand access is overlapped when separate memory modules are accessed
Core Memory Expandable to 32,768 words
Single-bit I/O Control and Sense
SDS 9300 COMPUTER
Arithmetic and Control
Input/Output Control
24-bit I/O
Instructions
Operands and Time-Multiplexed I/O
Basic 4096-word Memory
Optional 4096-word Memory
Optional 8192-word Memory
Optional 16,384-word Memory
24-bit Word Parallel I/O
A
B
C
D
Time-Multiplexed Communication Channels (Up to 30 devices/channel)
Multiple Access to Memory
Data Multiplex System
E
F
G
H
Direct Access Communication Channels (Up to 30 devices/channel)
To/from Special Devices
Memory Interface Connections*
Up to 128 Data Subchannels
Note: Broken lines indicate optional hardware.

Fig. 4. SDS 9300 computer-configuration diagram. (*Courtesy of Scientific Data Systems.*)

Registers and memory (930)

The Pc state is declared in the ISP description. The ISP registers are A, B, X, P, M, and miscellaneous bits for overflow, carry, etc. Overflow can be turned on for arithmetic overflow in addition, subtraction, multiplication, division, and left-shift instructions.

Data formats

General. A computer word, W, is 24 binary digits (bits) or 8 octal digits. A word is numbered $W\langle 0:23\rangle$ from left to right or alternatively $W\langle 0:7\rangle_8$.

Fixed-point data format. Fixed-point numbers are represented in two's complement form with the sign at $W\langle 0\rangle$. A 23-bit fraction $W\langle 1:23\rangle$ can be assumed. The binary point is to the left of bit position 1 ($W\langle 1\rangle$). For integers, the binary point is to the right of $W\langle 23\rangle$.

Floating-point data format. Subroutines perform double- and single-precision floating-point arithmetic. A floating-point word is defined as $f\langle 0:47\rangle := W[n:(n+1)]\langle 0:23\rangle$. Of course, single-precision floating point requires less processing time.

The fractional portion (mantissa), $f\langle 0:38\rangle$, of a double-precision floating-point number is a 39-bit proper fraction with the leading bit being the sign bit and the binary point located to the left of the most significant magnitude bit, $f\langle 1\rangle$.

The floating-point exponent is a 9-bit integer, $f\langle 39:47\rangle$, with the leading bit being the sign, $f\langle 39\rangle$. The standard routines operate on both fraction and exponent in two's complement form. If F represents the contents of the fractional field and E represents the contents of the exponent field, the number has the form $F \times 2^E$.

Standard subroutines assume that the more significant word is in the A register and that the less significant word is in the B register. Correspondingly for Mp, the more significant word is in Mp[x] and the least significant word in Mp[x + 1].

The single-precision floating-point representation is identical to that of double-precision floating point; i.e., it takes two words. However, the least significant bits of the mantissa, $f\langle 24:38\rangle$, are not processed; thus there is a saving in time but not in space for using single precision.

Instruction word format (930)

The computer instruction word format is given in Fig. 5.

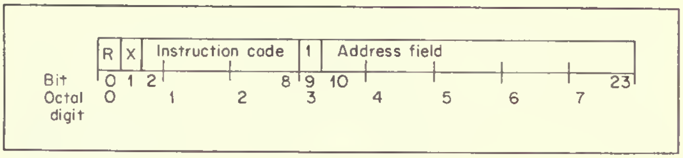


Fig. 5. SDS 930 instruction-format diagram.

$W\langle 0\rangle$ is the Relative Address bit, R. Standard software loading programs use this bit; central processor decoding logic does not use or sense this bit. A 1 in $W\langle 0\rangle$ causes some loading programs to add the assigned location of the instruction to the address field contents prior to actual storage into the assigned location.

$W\langle 1\rangle$ is the Index Register bit, X. It determines whether or not the index register will be added to calculate the effective address.

$W\langle 2:8\rangle$ is the Instruction Code field and determines the operation to be performed. The Programmed Operator facility is selected by $W\langle 2\rangle$; it is part of the Tag field $W\langle 0:2\rangle$.

$W\langle 9\rangle$ is the Indirect Address bit, I. It determines whether or not e or M[e] is to be used as the effective address (see below).

$W\langle 10:23\rangle$ is the Address field and for most instructions represents the location of the operand called for by the instruction code.

Address modification. Index and indirect addressing, used singly or in combination, perform address modification after bringing the instruction from memory but before executing it. The instruction remains in memory in its original form. The results of indexing and/or indirect addressing form the "effective address," e.

INDEXING If the content of the index bit in an instruction is a 1, prior to execution the computer adds the contents $X\langle 10:23\rangle$, of the index register to the contents of the address field of the instruction. This addition does not keep any overflow or carry beyond the fourteenth address bit. This addition occurs prior to any indirect action.

INDIRECT ADDRESSING A 1 in the indirect address bit causes the computer to decode the contents of the effective address, accessed as described above, as if it were an instruction without an instruction code; that is, the address logic reinitiates address decoding, using the word in the effective location (the memory cell whose address is the effective address). This is an iterative process and provides multilevel indirect and indexed addressing. Each level of indirect addressing adds an additional cycle time to the instruction execution time.

930 memory extension control registers. Core memory in the 930 is expandable to 32,768 words. However, the address field in the instruction format is 14 bits long, allowing direct access of only up to 16,384 words. Memory extension in the 930 contains two 3-bit memory extension registers, EM2 and EM3, and allows addressing of memories of 32,768 words. The program loads either or both of the registers and activates them as desired. Each register can become the most significant digit (fifth octal) of any operand address.

The program uses the first extension register, EM3, by calling for an address with an 11_2 in the most and next most significant address bits, respectively (a 3 for the most significant octal digit). The program calls for EM2, the second extension register, by setting the same two address bits to 10_2 (a 2 for the most significant octal digit). In this way, normal addressing compatible with the 910 and 920 occurs by setting a 3 in EM3, and a 2 in EM2.

910-930 instructions

Programmed Operators (POP's) enable subroutines to be called with a single instruction. This provides definable instructions of the same form as built-in machine instructions. The computer decodes the operation codes $100_8 \sim 177_8$ as special instructions and transfers to a subroutine whose address is uniquely determined by the code. The computer records the address of the POP instruction at location 0 together with an indirect address bit so that the program continuity may be maintained. By indirect addressing which refers to location 0, which in turn refers to the POP instruction, the subroutine can gain access to the effective address of the operand associated with the POP instruction.

The instruction set for the computers in this series is listed in Table 1. The table should be used to compare the machines.

There are two instructions in the 910 which are not in the 920 or 930: Multiply Step and Divide Step. These instructions facilitate writing subroutines for multiplication and division. The Multiply Step (MUS) instruction is defined:

$$\text{MUS} \rightarrow (B\langle 23\rangle \rightarrow A \leftarrow A + M[e];\ \text{next } AB \leftarrow AB/2);$$

9300 instructions

The instruction word format in the central processor is shown in Fig. 6.

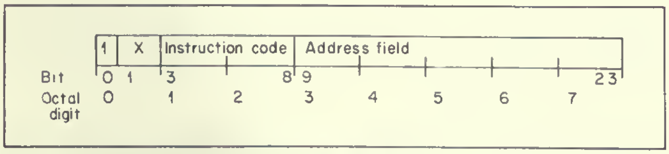


Fig. 6. SDS 9300 instruction-format diagram.

$W\langle 0\rangle$ contains the Indirect Address bit I.

$W\langle 1{:}2\rangle$ contains the Index Register bits $X\langle 0{:}1\rangle$.

$W\langle 0{:}2\rangle$ is called the Tag field.

$W\langle 3{:}8\rangle$ contains the Instruction code; the contents of this field determine the operation to be performed.

$W\langle 9{:}23\rangle$ contains the Address; for most instructions, the contents of this field represent the memory location of the operand called for by the instruction code.

Address modification. Each index register contains an unsigned base address of 15 magnitude bits and a signed increment of 9 bits. The increment contains 8 magnitude bits and a sign bit and is held in two's complement form.

Index registers are modified by adding the signed-increment value to the base address using two's complement arithmetic. Since the increment and base address fields are of unequal lengths, the sign bit (bit 0) of the increment field is extended six positions to the left prior to the addition. This 15-bit sum is then stored in the base address field of the index register. The index register may be incremented by any value from -256_{10} to 255_{10} using a single instruction. Incrementing and testing for a "terminal condition" is done by the instruction Increase Index And Branch (BRX), as follows:

If the index register has been negatively incremented, a terminal condition exists when the base address has been reduced below the zero value.

If the index register has been positively incremented, a terminal condition exists when the resultant base address has been increased beyond the maximum address value (077777_8).

If the terminal condition exists, the next instruction is taken in sequence. If the terminal condition does not exist, program control is transferred to the location specified.

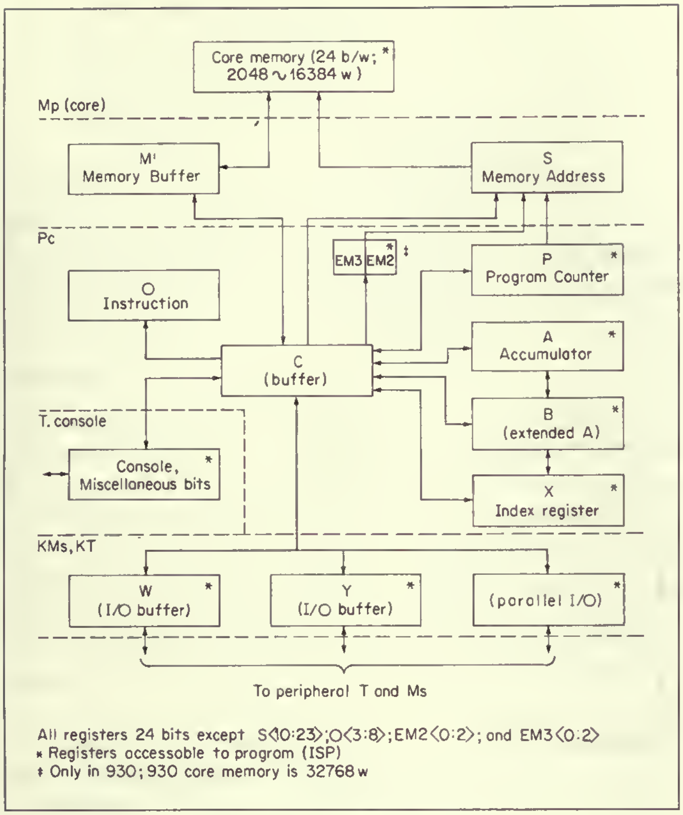
The instruction set for the 9300 is given in Table 1.

Pc implementation

All the processors of the series have basically similar register configurations because of the common Instruction-set Processor. However, the increasing complexities of the machines can be seen by comparing the register structures of the 910-930 (Fig. 7) with the 9300 (Fig. 8). The figures show both the registers accessible to the program or defined by the ISP (denoted by *) and the temporary registers which are necessary for the implementation.

910, 920, 930 registers (Fig. 7)

ISP registers (*). The A register is the main accumulator of the computer. The B register is an extension of the A register. The



All registers 24 bits except S⟨10:23⟩; O⟨3:8⟩; EM2⟨0:2⟩; and EM3⟨0:2⟩
* Registers accessable to program (ISP)
‡ Only in 930; 930 core memory is 32768 w

Fig. 7. SDS 910, 920, and 930 registers diagram.

B register contains the less significant portion of double-length numbers. Overflow and carry bits are used with A and B operations.

The index register X, used in address modification, is a full-word register. Index-register operations use the least significant 14 bits.

The P register is a 14-bit register that contains the memory address of the current instruction. Unless modified by the program, the contents of P increase by 1 at the completion of each instruction.

The memory extension registers, EM3 and EM2, are 3-bit registers that specify the portion of extended memory being used. They exist only in the 930.

Hardware registers not in the ISP. The S register is a 14-bit register that contains the address of the memory location to be accessed for instructions or data. The 15-bit address is formed by S and one of the memory extension registers.

The 24-bit C register communicates with memory. Instructions are temporarily held in C before instruction decoding. It is used as an arithmetic and control register in multiply, divide, and other operations. Address modification and parity generation/detection use the C register.

The O register is a 6-bit register that contains the instruction or operation code of the instruction being executed.

The M' register is a 24-bit register that holds each word as it comes from memory. Recopying of a word into memory takes place from the M' register.

9300 registers (Fig. 8)

ISP registers (°). The A and B registers of the 9300 are the same as in the 900 series computers; however, the P register is P⟨9:23⟩.

There are three 24-bit index registers, X[1:3]. Each index register is composed of a base address of 15 bits and a signed increment of 9 bits.

The Flag register, F, is a 6-bit register that may be set and/or sensed by the program. The first bit position of this register is the overflow indicator.

Hardware registers not in the ISP. The C register holds the 24-bit operand word as it is transmitted to, or received from, memory.

The D register holds the next 24-bit instruction word as it is received from memory.

The 15-bit S register contains the address of the memory location to be accessed for either instruction or operand.

The 6-bit O register contains the instruction code of the instruction being executed.

The A' register is an optional 15-bit register used for the floating-point option. It temporarily extends the A register during the execution of floating-point instructions.

The B' register is an optional 15-bit register which temporarily extends the B register during the execution of floating-point instructions.

Instruction interpretation in the 900 series

The instruction-interpretation process can be explained in terms of the processor's registers (Fig. 7). The ADD instruction execution (not including memory mapping) defined in ISP as $A \leftarrow A + M[e]$ is interpreted as

$S \leftarrow P$; $P \leftarrow P + 1$; next   *fetch the instruction*

$M' \leftarrow \text{Memory}[S]$; next

$C \leftarrow M'$; next

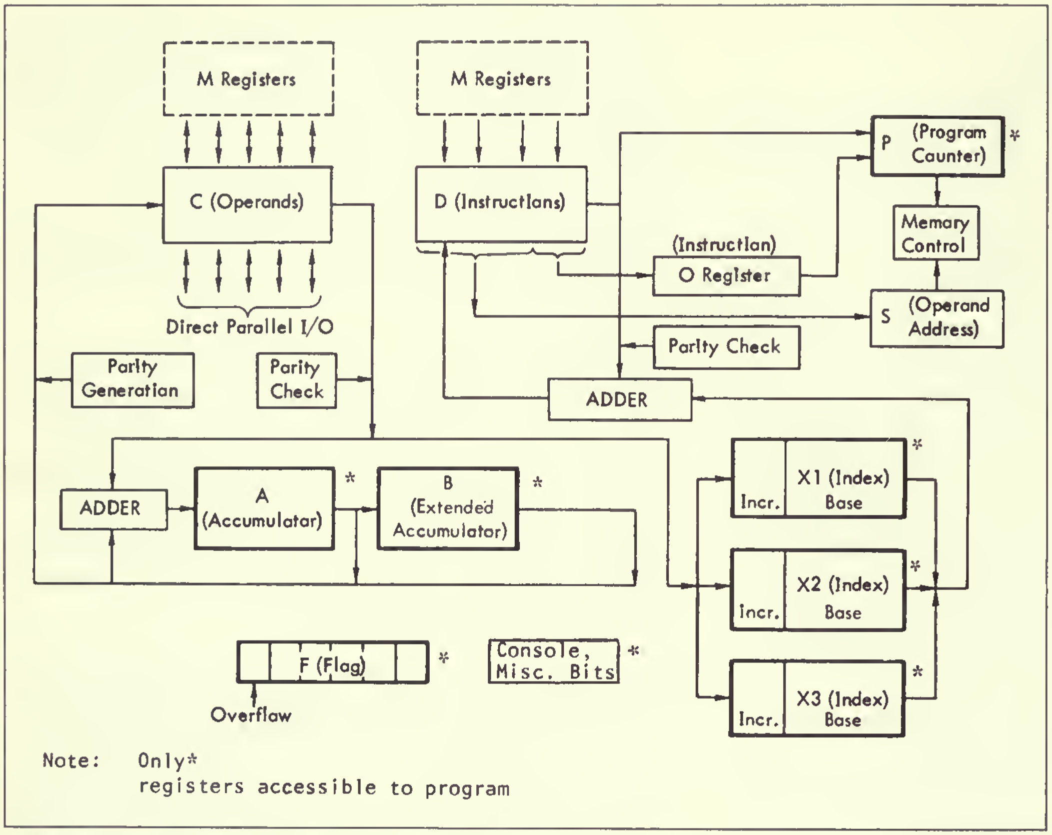


Fig. 8. SDS 9300 registers diagram. (*Courtesy of Scientific Data Systems.*)

```
O ← C⟨0:5⟩; next
(O = 05) → (                  ADD execution
                              operand effective-address-cal-
                                culation process (including
                                indexing and indirect ad-
                                dressing)
    S ← C⟨10:23⟩; next        final operand fetch
    M' ← Memory[S]; next
    C ← M'; next
    A ← A + C)                add operation
```

Input/output processing

Introduction

There are several methods of transferring data between Mp and the K's. These methods will be described independently, and in order of increasing complexity. They are:

1*a* Single bit sent to a selected K (EOM instruction).

1*b* Single bit sense (or bit detection) from a K (SKS instruction).

2 Word parallel to/from a K (POT/PIN instruction).

3 Interrupt from one of 1,024 K's on a priority basis to Pc. K can signal Pc to execute a particular program.

4*a* Time Multiplexed Communication Channel/TMCC (Internal Interlace[1] feature).

4*b* Time Multiplexed Communication Channel (External Interlace[2]).

5 Direct Access Communication Channel/DACC (External Interlace).

[1] The control information for the location of the next word transferred and the number of words to transfer are kept in Mp.

[2] The control information is taken from registers within K.

6a Data Subchannel/DSC (Internal Interlace).

6b Data Subchannel (External Interlace).

7 Memory Interface Connection/MIC link. A component has a link to Mp.

Methods 1 to 3 above are completely under control of a program and are simple time-independent instructions (or methods) of transferring data to K's (and onto KT or KMs). The ISP description (Appendix 1 of this chapter) has a detailed description of the I/O devices and these I/O instructions.

Single-bit control and sense

Two instructions provide for single-bit ON/OFF control signals. The first, EOM, transmits a control signal and a 14-bit address to an external device or a function within the computer. The second, SKS, selects an external device or computer function and skips in response to a false (0) signal. Up to 16,384 control signals can be sent and 16,384 input signals tested theoretically. (A more reasonable number of physical destinations would be 50.) Execution of an EOM causes a signal of approximately 1.4 microseconds duration to be transmitted.

EOM instruction format. EOM is used to select a specific I/O device by placing a 1 in its select register. EOM requires one cycle.

W⟨2⟩ = 0.

W⟨0:1⟩ is reserved for special system address bits.

W⟨3:8⟩ contains the EOM instructions code, 02.

W⟨10:11⟩ contains the system mode specifier.

W⟨12:23⟩ contains the 12-bit address field that specifies the special system destinations.

SKS format. The SKS instruction format has each corresponding bit field identical to the system EOM format. Execution of an SKS causes a 14-bit address to be presented to all K's; the K being addressed responds and is tested. If the addressed external K supplies a "set" signal to the central processor, the computer executes the next instruction in sequence from the SKS. If no signal is set, the computer skips the next instruction in sequence and executes the following instruction. No registers are affected except the P register. SKS requires two or three Mp cycles if no skip or skip, respectively, is executed.

Word parallel instructions

Two instructions, Parallel Output (POT) and Parallel Input (PIN), permit any word in Mp to be presented in parallel on a physical connector to a K or, inversely, permit signals sent from a K to be stored in Mp. The execution of a POT or PIN instruction sends a signal to the external device involved in the input/output operation, which notifies the device to send its data word as soon as it is operational. When the device becomes operational during a Read or PIN operation, it transmits a Ready signal to the central processor while at the same time presenting a data word to Pc.

During the execution of a POT instruction, the central processor transmits a signal to the external device, alerting it to receive a data word. When the device becomes operational, it transmits a Ready signal to the central processor, which releases the data word to the external device.

Selective input/output with these devices is accomplished by preceding POT or PIN with an EOM to alert (select) the desired device by a specific address. By preceding the POT or PIN with an SKS, the Ready signal of the special device can be tested after the execution of the EOM but prior to execution of the parallel transfer instruction; a possible Pc "hangup" can thus be avoided. The Ready signal can also set one of the priority interrupts.

PIN stores the contents of 24 input lines in parallel in the effective-memory location. PIN or POT requires four cycles plus any waiting time for Ready.

Interrupt

The interrupt provides program control of input/output operations, aids in programming simultaneous input/output and compute operations, and allows immediate recognition of special external conditions by causing Pc to execute an instruction in a selected Mp location at the end of the execution cycle of the current instruction. Without disturbing the program register, the processor executes an instruction in one of a selected set of memory locations. A Mark Place and Branch (BRM) instruction in this location saves the contents of the program register, EM3, EM2, and overflow indicator and transfers to the particular interrupt servicing routine required. To exit from the interrupt service routine, a Branch Unconditionally (BRU) instruction using indirect addressing returns control to the next instruction in proper sequence in the main program; it also clears the interrupt. Processor state (that is, A, B, Overflow, and X) must be preserved and restored by the program if the registers are used by the program.

The priority interrupt system has up to 1,024 interrupts arranged in levels. The levels have priority according to a priority number; the higher priority levels have a smaller number. Interrupt channels are installed in Pc in groups of 16. The assignment of physical memory locations to interrupt levels is shown in Appendix 1 of this chapter; the assignment is in order of decreasing priority from location 200_8 (highest) to 1477_8 (lowest). Interrupt requests can also be programmed. The power fail-safe (for power supply off) interrupts and out-of-order interrupts have the highest priority.

Besides the interrupt mechanism just discussed, there is also a single instruction interrupt. This permits the execution of only one instruction before automatically being cleared and returning to the program that was interrupted. For example, if an external clock source is connected to the computer so that it pulses an interrupt line at set intervals, the program can maintain a programmed real-time clock. Each time the external pulse causes an interrupt, the program executes the single instruction, Memory Increment (MIN), to add 1 to the memory word selection for use as a programmed real-time clock. (The main program can examine this memory location whenever necessary to determine how many time increments have elapsed since the clock was started.)

Interrupts can be single or normal-instruction interrupts in any combination desired.

An interrupt has three operational states: inactive, waiting, and active states.

In the inactive state, no interrupt signal has been received into the level and none is currently being processed by its interrupt servicing subroutine.

In the waiting state, an interrupt has been received but is not being processed. This situation may arise when an interrupt of higher priority is being processed. When all higher waiting interrupts have been processed, this level goes to the active state.

In the active state, the interrupt has caused the main program to recognize its presence and has transferred to its assigned interrupt location where it is being processed.

Two program control features are Arm/Disarm and Enable/Disable. Arm/Disarm controls whether an interrupt can proceed from the inactive state to the waiting state. When armed, an interrupt signal sets the interrupt to the waiting state. Enable/

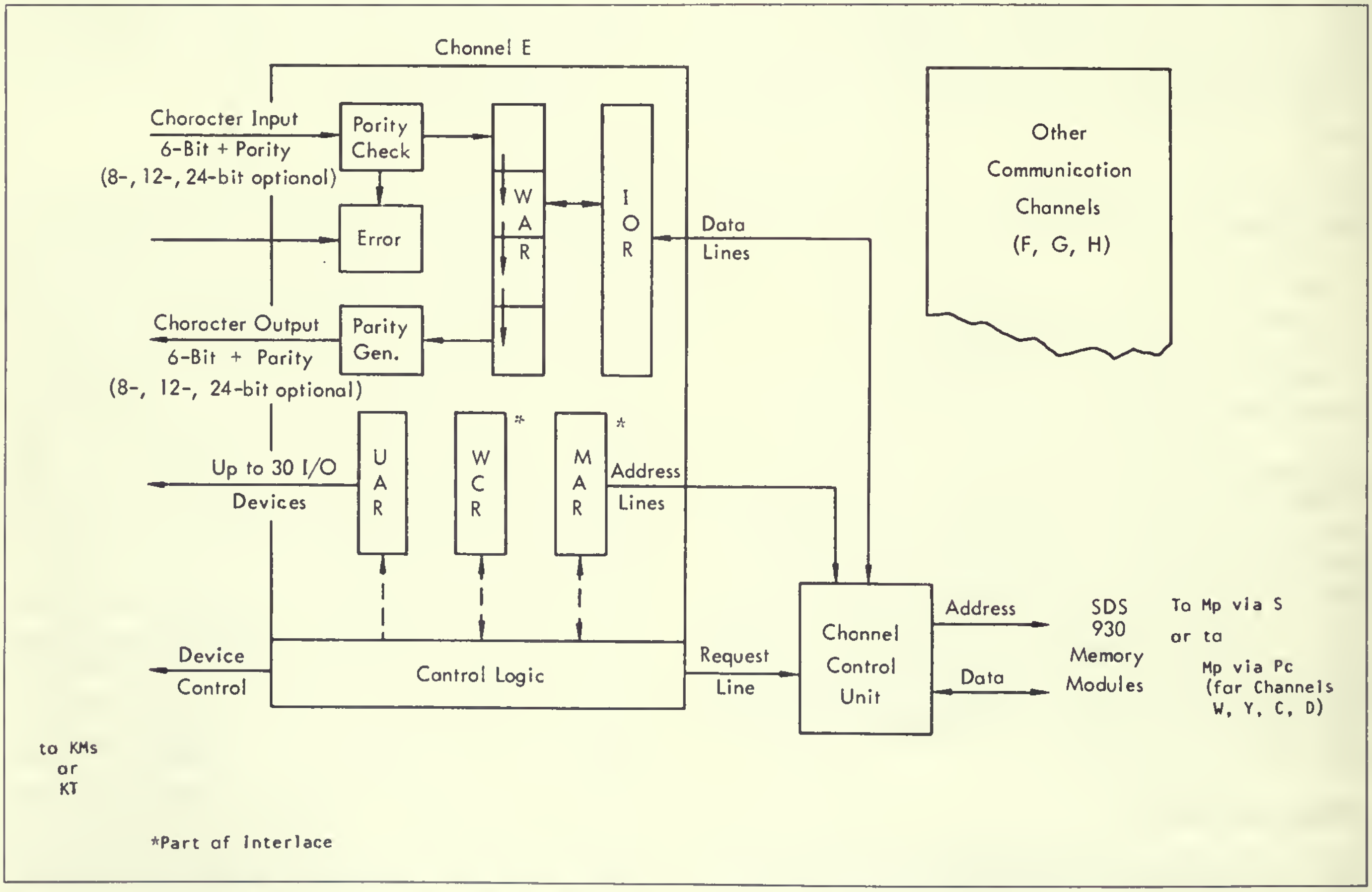


Fig. 9. SDS 930 direct-access communication-channel register diagram. (*Courtesy of Scientific Data Systems.*)

Disable operates on the entire interrupt system. (When the interrupt system is enabled, interrupts can occur.)

Communications channels—Kio('Channel)'s

Kio('Communication Channels) provide buffering, input/output control, and data transmission simultaneously with computation. There can be up to eight independent communication channels and a large number of subchannels in a single system. Figure 9 shows the registers in a K('Channel).

Each channel can control up to 30 KT's or KMs's. The channel handles character, word assembly and disassembly, input/output parity detection and generation, data transmission to and from memory, and end-of-transmission detection.

All channels are bidirectional and can communicate with 6-bit character devices or word devices in 6, 12, and 24 bits. The main program that initializes a K specifies the number of characters to be contained in each word during the transmission.

The channel interlace controls the transfer of the data words going through the associated channel buffer, supplies the memory address of data coming from or going to memory, and maintains the word count determining the number of words transferred. This interlace information can be either in K hardware (external interlace) or in Mp (internal interlace). The terminal interrupts, End of Record and Zero Word Count, come from the interlace and are under its control.

The time-multiplexed channels use the memory-access logic of Pc to transmit input and output of data words and require two memory cycles (see Fig. 2). Each direct-access channel has independent memory-access logic and requires one memory cycle (see Fig. 2).

Communication-channel description. Up to 30 peripheral devices (K's for T or Ms) may be connected to one K('Channel) (Fig. 9). Each device has a unique, 2-digit, octal address by which it is selected for an input/output operation. To select the peripheral device, the program loads the proper unit address into the 6-bit Unit Address Register (UAR) in the channel. This address selects both the device and, if appropriate, the function to be performed. Placing a nonzero unit address in the unit address register connects the peripheral unit addressed to the channel, and the unit becomes active. When the UAR contains a zero address, or any time that a terminal or initial condition clears the contents of UAR, the channel becomes inactive.

The 24-bit data Word Assembly Register (WAR) contains the data word actively being received or transmitted during an input or output operation. During input, 6-bit characters (plus parity) enter the Single-Character Register (SCR) where the channel buffer assembles them, one at a time, into the WAR.

The channel interlace contains two working registers: the Word Count Register (WCR) and the Memory Address Register (MAR). A channel may have these registers either in K or in Mp. In the setup sequence for an interlaced input/output operation, the POT instruction transmits to the interlace a data word made up of the word count (that is, length) and the starting address of the data block. The 15-bit Word Count Register (WCR) contains the data word count during a data transfer. The number of data words is decremented by 1, and the new count replaces the old one in the WCR for each word transmitted.

The Memory Address Register (MAR) contains the starting destination or source address in memory of the transmitted data. The memory locations to or from which data words are to be transmitted enter the MAR at the same time the word count does. During transmission of data, the interlace increments the MAR after each word as it decrements the contents of the WCR. These two registers provide the interlace control of block transmissions. Obviously, if the interlace control registers are in Mp, then two extra accesses are required for each word transferred.

Memory interface connection link

Once a computer is equipped with a multiple-access-to-memory feature, one or more Memory Interface Connections (MIC) can be attached. The MIC is a general interface to the computer that allows special devices to access Mp. It preserves the integrity of the memory by generating the parity of incoming data words and checking the parity of words read from memory to indicate memory failures. The device that is connected to the MIC must hold both the data and the address until the transmission to/from memory is completed (that is, MIC does not have registers).

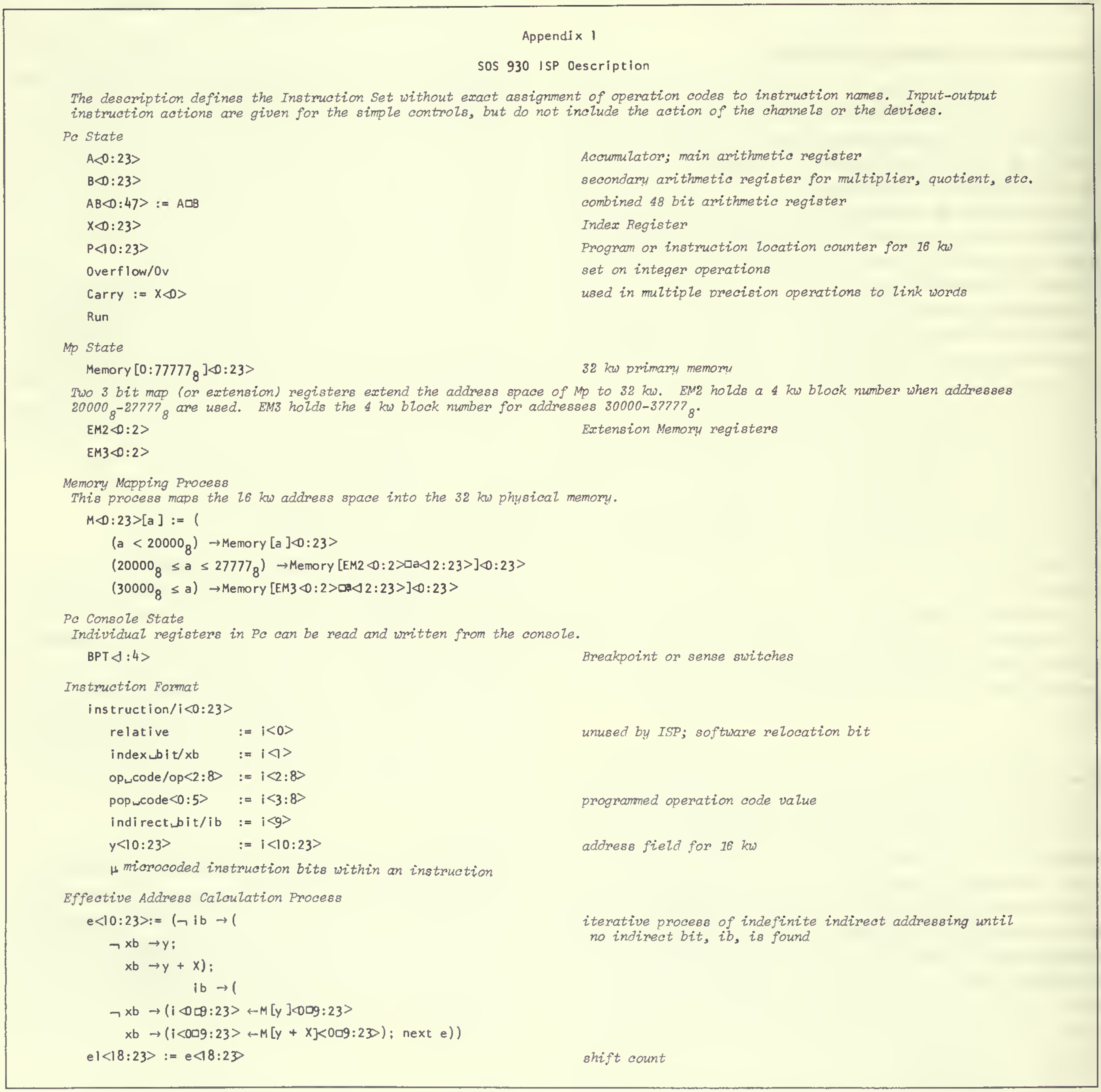
Conclusions

The SDS computers appear to be the first attempt to design several computers at the same time with a common ISP. Over a longer time span other compatible computers were added to the original 910 and 920 as technology (and marketing) dictated. The series is characteristic of well-designed typical 24-bit computers. By increasing the arithmetic capability, the series could also be used more generally.

References

Scientific Data Systems Reference Manuals for the 930 and 9300 computers

APPENDIX 1 SDS 930 ISP DESCRIPTION



Appendix 1

SDS 930 ISP Description

The description defines the Instruction Set without exact assignment of operation codes to instruction names. Input-output instruction actions are given for the simple controls, but do not include the action of the channels or the devices.

```
Pc State
   A<0:23>                                              Accumulator; main arithmetic register
   B<0:23>                                              secondary arithmetic register for multiplier, quotient, etc.
   AB<0:47> := A□B                                      combined 48 bit arithmetic register
   X<0:23>                                              Index Register
   P<10:23>                                             Program or instruction location counter for 16 kw
   Overflow/Ov                                          set on integer operations
   Carry := X<0>                                        used in multiple precision operations to link words
   Run

Mp State
   Memory[0:$77777_8$]<0:23>                            32 kw primary memory
 Two 3 bit map (or extension) registers extend the address space of Mp to 32 kw.  EM2 holds a 4 kw block number when addresses
 $20000_8$-$27777_8$ are used.  EM3 holds the 4 kw block number for addresses 30000-$37777_8$.
   EM2<0:2>                                             Extension Memory registers
   EM3<0:2>

Memory Mapping Process
 This process maps the 16 kw address space into the 32 kw physical memory.
   M<0:23>[a] := (
      (a < $20000_8$) → Memory[a]<0:23>
      ($20000_8$ ≤ a ≤ $27777_8$) → Memory[EM2<0:2>□a<12:23>]<0:23>
      ($30000_8$ ≤ a) → Memory[EM3<0:2>□a<12:23>]<0:23>

Pc Console State
 Individual registers in Pc can be read and written from the console.
   BPT<1:4>                                             Breakpoint or sense switches

Instruction Format
   instruction/i<0:23>
      relative          := i<0>                         unused by ISP; software relocation bit
      index␣bit/xb      := i<1>
      op␣code/op<2:8>   := i<2:8>
      pop␣code<0:5>     := i<3:8>                       programmed operation code value
      indirect␣bit/ib   := i<9>
      y<10:23>          := i<10:23>                     address field for 16 kw
      μ microcoded instruction bits within an instruction

Effective Address Calculation Process
   e<10:23>:= (¬ ib → (                                 iterative process of indefinite indirect addressing until
      ¬ xb → y;                                          no indirect bit, ib, is found
        xb → y + X);
                 ib → (
      ¬ xb → (i<0□9:23> ← M[y]<0□9:23>
        xb → (i<0□9:23> ← M[y + X]<0□9:23>); next e))
   el<18:23> := e<18:23>                                shift count
Instruction Interpretation Process
   ¬ Interrupt␣interpretation → (                              normal interpretation
      instruction ←M[P]; P ←P + 1; next                        fetch
         Instruction␣execution);                               execute
     Interrupt␣interpretation → (                              interrupt interpretation
      instruction ←M[$200_8 + 20_8 \times$ K␣address + I␣address]: next
         Instruction execution)

Instruction Set and Instruction Execution Process
   Instruction␣execution := (
   Load and Store Group
      LDA → (A ←M[e]);                                         load A
      STA → (M[e] ←A);                                         store A
      LDB → (B - M[e]);                                        load B
      STB →M[e]←B);                                            store B
      LDX → (X ←M[e]);                                         load index
      STX → (M[e] ←X);                                         store index
      EAX → (X ←e);                                            load index from e
      XMA → (M[e] ←A; A ←M[e]);                                exchange A and M
   Arithmetic Group
      SUB → (Ov,Carry□A ←A - M[e]);                            subtract
      ADD → (Ov,Carry□A ←A + M[e]);                            add
      SUC → (Ov,Carry□A ←A - M[e] - Carry);                    subtract with Carry
      ADC → (Ov,Carry□A ←A + M[e] + Carry);                    add with Carry
      MIN → (Ov,M[e] ← M[e] + 1);                              memory increment
      ADM → (Ov,M[e] ← M[e] + A);                              add to memory
      MUL → (Ov,AB ←A x M[e]);                                 multiply
      DIV → (Ov,B ←AB/M[e]; A ←AB mod M[e]);                   divide
   Logical Group
      ETR → (A ←A ∧ M[e]);                                     extract
      MRG → (A ←A ∨ M[e]);                                     merge
      EOR → (A ←A ⊕ M[e]);                                     exclusive or

Microcoded Register Exchange Instruction
 Each instruction can be formed from a series of microprogrammed operations.  Compound microcoded instructions are shown below
 without a μ).
      CLA → (A ←0);                                            μ, clear A
      CLB → (B ←0);                                            μ, clear B
      CLR → (AB ←0);                                           clear A and B
      CLX → (X ←0);                                            μ, clear X
      CAB → (B ←A);                                            μ, copy A into B
      CBA → (A ←B);                                            μ, copy B into A
      XAB → (A ← B; B ←A);                                     exchange A and B
      CXB → (B ←X);                                            μ, copy X into B
      CBX → (X ←B);                                            μ, copy B into X
      XXB → (X ←B; B ←X);                                      exchange X and B
      CAX → (X ←A);                                            μ, copy A into X
```

```
    CXA → (A ← X);                                                     μ, copy X into A
    XXA → (A ← X; X ← A);                                              exchange X and A
    CNA → (A ← ¬ A);                                                   μ, not A
    BAC → (A ← B; B ← 0);                                              copy B into A, clear B
    ABC → (B ← A; A ← 0);                                              copy A into B, clear A
    STE → (X<15:23> ← B<15:23>; X<0:14> ← sign␣extend(B<15>));         μ, store exponent: exponent control bit
           B<15:23 > ← 0);
    LDE → (B<15:23> ← X<15:23>);                                       load exponent
    XEE → (<B15:23> ← X<15:23>; X<15:23> ← B<15:23>;                   exchange exponent
           X<0:14> ← sign␣extend(B<15>));
    End of microcoded instruction group
  Shift Group
     LRSH → (AB ← AB / 2^el {logical});                                logical right shift
     RSH → (AB ← AB / 2^el);                                           right shift
     RCY → (AB ← AB / 2^el {rotate});                                  right cycle
     LSH → (Ov,AB ← AB x 2^el);                                        left shift
     LCY → (AB ← AB x 2^el {rotate});                                  left cycle
     NOD → (X ← X - normalize␣exponent(AB)                             normalize, decrease X
            AB ← normalize(AB));
  Skip Test Group
    SKE → ((A = M[e]) → (P ← P + 1));                                  skip if A = M
    SKB → ((M[e] ∧ B) = 0) → (P ← P + 1);                              skip if B and M don't compare 1's
    SKN → (M[e]<0> → (P ← P + 1});                                     skip if M negative
    SKR → (Ov,M[e] ← M[e] - 1; next M[e]<0> → (P ← P + 1});            reduce M, skip < 0
    SKM → ((M[e] ∧ B) = (A ∧ B)) → (P ← P + 1);                        skip on masked M
    SKG → (A > M[e]) → (P ← P + 1);                                    skip if greater than M
    SKO → (XR<0:23> ← abs(B<15:23> - M[e]<15:23>};                     difference exponents and skip
           (M[e]<15:23> > B<15:23>) → (P ← P + 1)):
    SKA → ((M[e] ∧ A) = 0) → (P ← P + 1);                              skip if A and M don't compare 1's
  Branch Group
    BRU → (P ← e);                                                     branch unconditionally
    BRX → (X ← X + 1; X<9> → P ← e);                                   increment Index, Branch
    BRM → (M[e]<0> ← Ov; M[e]<3:5> ← EM3; M[e]<1,2,9> ← 0;             mark place and branch
           M[e]<6:8> ← EM2: M[e]<10:23> ← P; next                      used to call subroutines
           P ← e + 1);
    BRR → (P ← M[e] + 1: Ov ← Ov ∨ M[e]<0>);                           branch return; used in terminating subroutines
  Control Group
    HLT → (Run ← 0):                                                   halt
    NOP → :                                                            no operation
    EXU → (instruction ← M[e]:                                         execute
           instruction␣execution);
  Overflow Test Group
    OVT → (Ov → (P ← P + 1); (Ov ← 0));                                overflow test
    ROV → (Ov ← 0);                                                    reset overflow
    REO → (X<14> ⊕ X<15>) → (Ov ← 1);                                  record exponent
    Breakpoint Test Group
       ((BPT 1 ∧ BPT<1>) ∨ (BPT 2 ∧ BPT<2>) ∨ (BPT 3 ∧ BPT<3>) ∨ (BPT 4 ∧ BPT<4>)) → (P ← P + 1);
    Memory Extension Register Control Group
       SET → (instruction<17> → (EM2 ← instruction<21:23>);
              instruction<16> → (EM3 ← instruction<18:20>));
       EXT → condition → (P ← P + 1);
             condition := ((instruction<22> ∧ (EM2 = 2)) ∧ (instruction<23> ∧ (EM3 = 3)))
       POP → (M[0]<0,9:23> ← 0v□1□P; P ← 100_8 + pop␣code);         programmed operator; 64 user defined instructions called via
                                                                     subroutine link in M[0]
       EOM → IO␣instruction␣execution;                                see the definition of the IO instruction set below
       POT → IO␣instruction␣execution;
       PIN → IO␣instruction␣execution;
       SKS → IO␣instruction␣execution;
                   )                                                  end Instruction␣execution; not including Input Output
                                                                       instructions
 Input-Output Control from the Pc

 KT and KMs State
  Devices consist  of the following parts:
    IO␣Device[0:77777_8]                                              name (or address) of a specific IO device; the EOM command
                                                                       is first given to select the specific device; subsequent
                                                                       commands are implicitly to the selected device
    IO␣Output[0:77777_8]<0:23>                                        Input and Output Data buffers associated with specific
    IO␣input[0:77777_8]<0:23>                                          devices
    IO␣Ready[0:77777_8]                                               bit for each device to denote when device is ready to trans-
                                                                      mit data
    IO␣Select[0:77777_8]                                              a bit within each device denoting it has been selected for
                                                                      an operation
    io␣unit<0:14>                                                     the particular io device selected by the EOM command;
 IO Instruction Set
    EOM → (io␣unit ← e);                                              command to select or address the device; energize output M
    POT → (IO␣Select[io␣unit] ∧ IO␣Ready[io␣unit] → (                 output data command
             IO␣Output[io␣unit] ← M[e]; io␣unit ← 0);
             IO␣Select[io␣unit] ∧ ¬ IO␣Ready[io␣unit] → (POT));        wait until ready
    PIN → (IO␣Select[io␣unit] ∧ IO␣Ready[io␣unit] → (                 input data command
             M[e] ← IO␣Input[io␣unit]; io␣unit ← 0);
             IO␣Select[io␣unit] ∧ ¬ IO␣Ready[io␣unit] → (PIN));        wait until ready
    SKS → (io␣unit ← e; next                                          skip if signal is not set
            (IO␣select[io␣unit] ∧ IO␣Ready[io␣unit] → (
             P ← P + 1);
             io␣unit ← 0);
 Interrupt System States
    Interrupt                                                         controls whether interrupts will be processed
    I␣RQ[0:63]<0:15>                                                  array of 1024 interrupt requests
    I␣ON[0:63]<0:15>                                                  array of interrupt enable to enable or inhibit interrupt
                                                                       requests
    I␣Signal[0:63]<0:15> := I␣RQ[0:63]<0:15> ∧ I␣ON[0:63]<0:15>
    K␣address<0:5>                                                    group number
    I␣address<0:3>                                                    level number within a group of the active interrupt
```

The I_address and K_address combine ($200_8 + 20_8 \times$ K address + I_address) to establish an interrupt address, 200_8 is the highest priority and $200_8 + 1477_8$ the lowest priority.

Interrupt_level_state[0:63]<0:15>$_3$

There are three states associated with each interrupt, Inactive, Waiting, and Active:
Inactive means no I_signal is present.
Waiting means the I_signal has been received but is waiting to be processed.
Active means the interrupt has caused the main program to recognize its presence.

The instruction in M[$200_8 + 20_8 \times$ K_address + I_address] is executed upon interrupt. There are two kinds of interrupts: single instruction allows one instruction to be executed and the interrupt level state is changed from active to inactive; and normal requires that a mark place and branch, BRM, instruction to be executed to save P. At the completion of the interrupt program, a branch unconditional (BRU) indirectly via the BRM instruction restores the interrupt level. (That is, the Interrupt_level_state is changed from Active to Inactive, and another I_Signal can be processed.)

Interrupt_interpretation

A state denoting that an interrupt is to be processed or the interrupt level state to be changed from Waiting to Active for normal interrupts and Waiting to Active to Inactive for single interrupts. The interrupt processed is the highest of those waiting provided there are no interrupts of highest level in the Active state.

Interrupt Control Instructions

```
  EIR → (Interrupt ← 1);                       enable interrupt: turn on mode
  DIR → (Interrupt ← 0);                       disable interrupt; turn off
  IET → (Interrupt → P ← P + 1);               interrupt test: skip if on
  IDT → (¬ Interrupt → P← P + 1);              interrupt disable test; skip if off
```

POT instruction to control the Interrupt System. EOM[20020] is first given to select the Interrupt Suetem.

```
 (POT ∧ IO_Ready[20020]) → (                           interrupt control instructions
     (c = 1) → I_ON[a]<0:15> ← I_ON[a]<0:15> ∨ B<0:15>;       arm a channel level group
     (c = 2) → I_ON[a]<0:15> ← I_ON[a]<0:15> ∨ ¬ B<0;15>;     disarm a channel level group
     (c = 3) → I_ON[a]<0:15> ← b<0:15>);                      set a channel level group
        a<0:5> := M[e]<0:5>                                   group select or K_address
        b<0:15>:= M[e]<8:23>                                  data for I_address
        c<0:1> := M[e]<6:7>                                   command control bits
```