TP 1086

RCA 4100

PROGRAMMERS' REFERENCE MANUAL

DATA SYSTEMS DIVISION
DEFENSE ELECTRONIC PRODUCTS
RADIO CORPORATION OF AMERICA

RCA 4100

PROGRAMMERS' REFERENCE MANUAL

Data Systems Division
Defense Electronic Products
Radio Corporation of America

4100 REFERENCE MANUAL

ERRATA

PAGE	CORRECTION
2-1	The fourth line in the fourth paragraph should be amended to read, "content of the selected B-box, denoted as C(B), is algebraically added -- modulo 2^{14} --"
6-2	The fifth line below the instruction word format which reads, "replaces the $C(AC)_0$." should be deleted.
8-2	The first sentence under the SKP instruction word format should be amended to read, "This instruction tests the C(L) as specified by the C-code of the instruction word."
9-3	The first line of the first sentence in the second paragraph should read as follows: "If the selected buffer is not available, it does not accept the control word nor . . .".

CONTENTS

SECTION 1

GENERAL INFORMATION

The RCA 4100 is a family of general-purpose digital computers, providing a high degree of flexibility and system adaptability at moderate cost. The members of the 4100 family (Figure 1-1, 1-2 and 1-3) are interchangeable in their electrical, logical, and mathematical characteristics.

The block diagram of the 4100 Basic Processor (Figure 1-4) illustrates the inherent simplicity of the computer. Three 30-bit registers (AC, R, and G), one 30-bit adder (S), four 15-bit registers (F, I, L, and P), the random-access core memory, and the control unit comprise the entire main frame.

Of these items, the accumulator register (AC), the flag register (F), the overflow indicator (V), and the core-storage cells are accessible to the programmer. Dual significance is attached to the block of 128 core-storage words at the upper end of the memory, termed the "executive" space. Although these locations can be addressed and treated as normal storage space, extreme caution must be exercised because they are also used to control the sixteen programs which the computer is capable of handling. (See Section 2 on Priority Interrupt and Input-Output.) This block is subdivided into sixteen sections of eight words each. The first of these is the instruction location counter, IL, which contains the location of the current instruction in the associated program. The other seven are the corresponding B-boxes and work space, as indicated in Table 1-1.

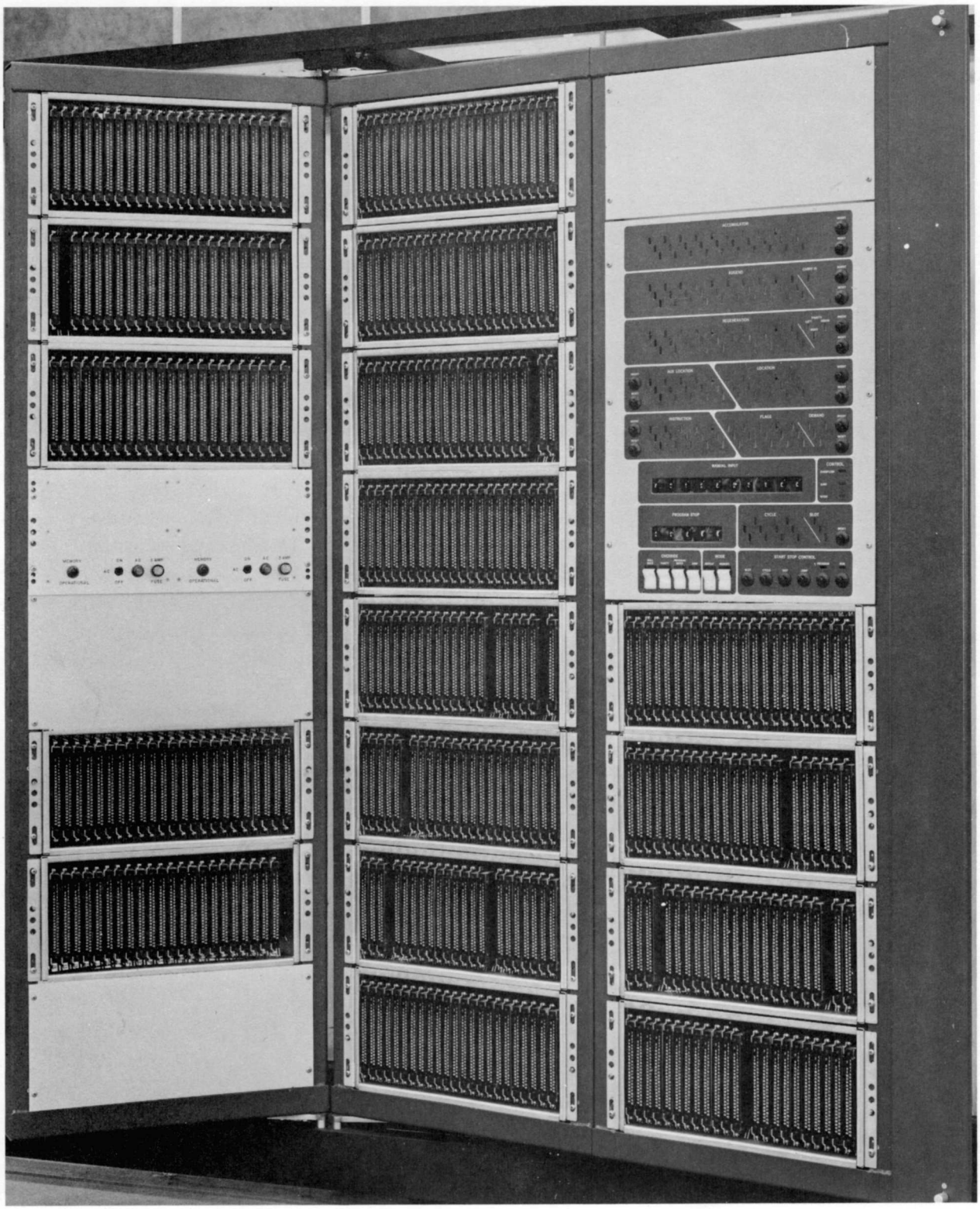


Figure 1-1. 4101 Computer

Figure 1-2. Mobile Version of 4100 or 4150 Computer

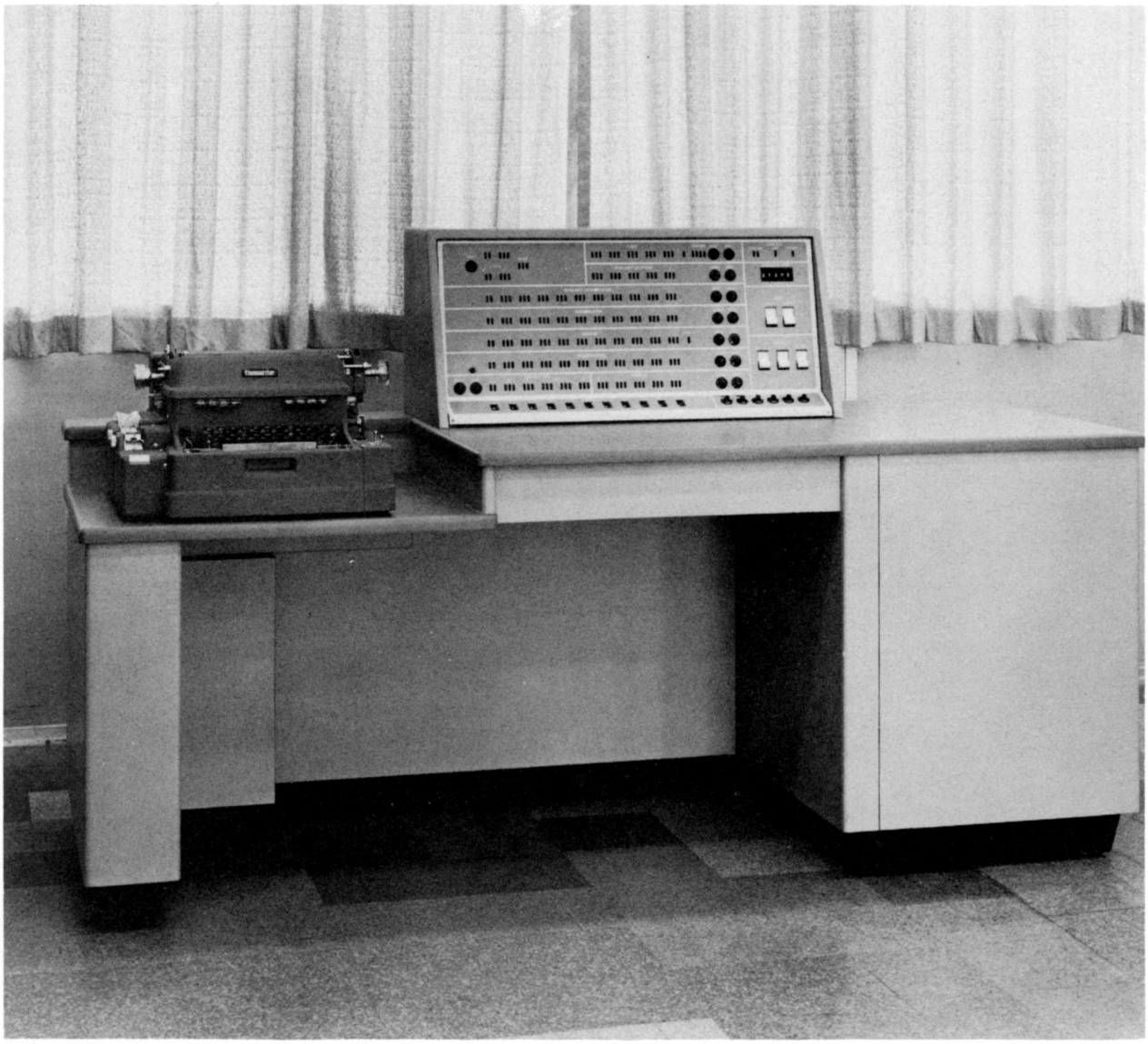


Figure 1-3. Desk Version of 4100 or 4150 Computer

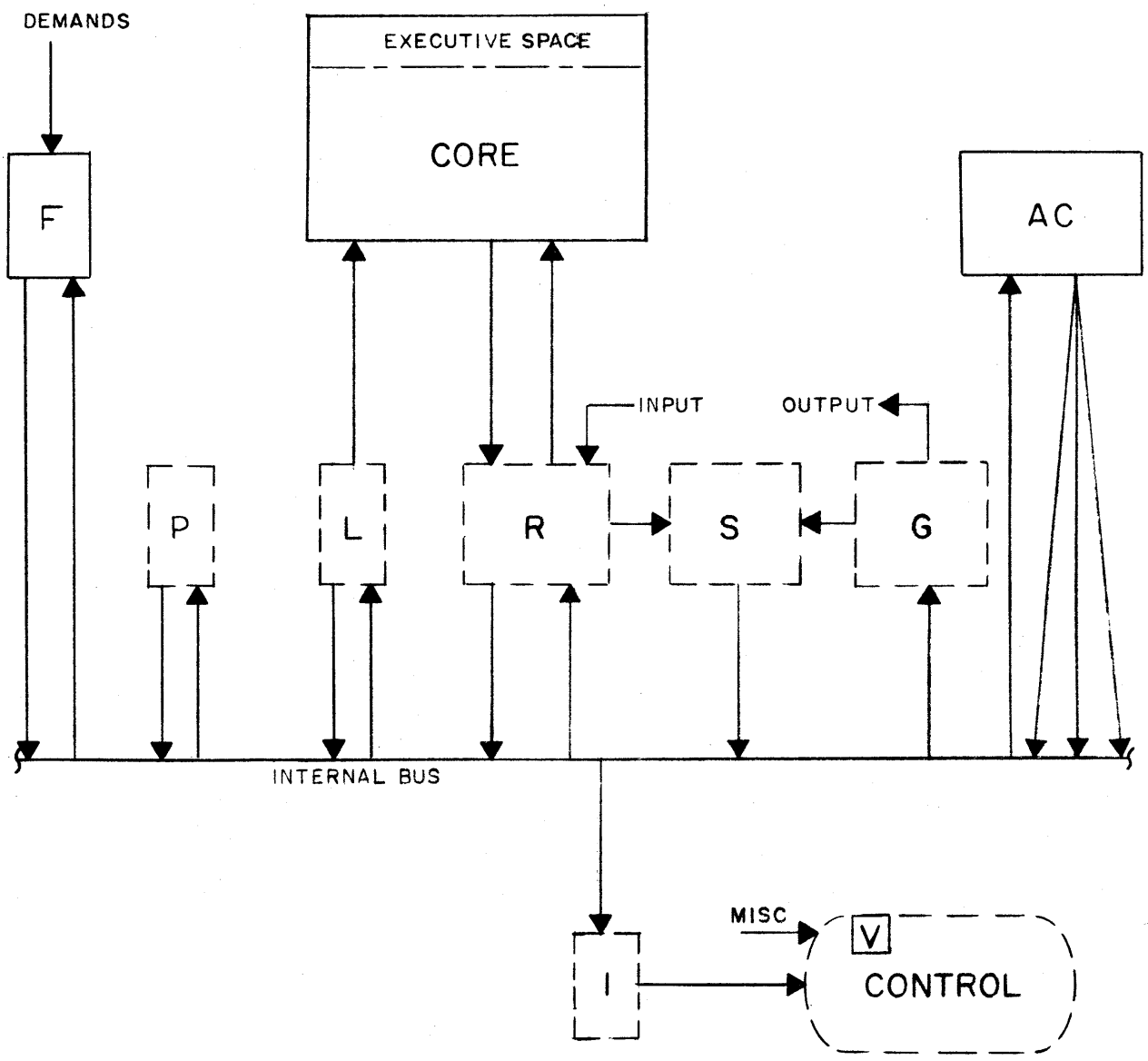


Figure 1-4. Block Diagram of 4100 Basic Processor

TABLE 1-1

LOCATIONS IN EXECUTIVE SPACE

Priority	IL	B-box 6	B-box 5	B-box 4	B-box 3	B-box 2	B-box 1	Work Space
Lowest	37777	37776	37775	37774	37773	37772	37771	37770
	37767	37766	37765	37764	37763	37762	37761	37760
	37757	37756	37755	37754	37753	37752	37751	37750
	37747	37746	37745	37744	37743	37742	37741	37740
	37737	37736	37735	37734	37733	37732	37731	37730
	37727	37726	37725	37724	37723	37722	37721	37720
	37717	37716	37715	37714	37713	37712	37711	37710
	37707	37706	37705	37704	37703	37702	37701	37700
	37677	37676	37675	37674	37673	37672	37671	37670
	37667	37666	37665	37664	37663	37662	37661	37660
	37657	37656	37655	37654	37653	37652	37651	37650
	37647	37646	37645	37644	37643	37642	37641	37640
	37637	37636	37635	37634	37633	37632	37631	37630
	37627	37626	37625	37624	37623	37622	37621	37620
	37617	37616	37615	37614	37613	37612	37611	37610
Highest	37607	37606	37605	37604	37603	37602	37601	37600

Normally, the IL counter is decremented by one, following the execution of each instruction. This causes the instructions to be performed in descending sequential order. Deviations from this sequence are discussed in later chapters.

The 4100 performs all arithmetic operations in two's complement notation. The most significant bit of any word is its sign (i.e., "0" for a positive number and "1" for a negative number). For example, + .25 would be represented by the binary form 00100000000000000000000000000 and - .25 would be represented by the binary form 11100000000000000000000000000.

It is, therefore, obvious that the range of numbers that can be represented by a machine word extends from - 1.0 (octal representation 4000000000) to + $1.0-2^{-29}$ (octal representation 3777777777). The octal representation of the number "zero" is always 0000000000 and can never have a negative sign.

Whenever the result of an arithmetic operation exceeds these limits, the overflow indicator V is set. Unless this indicator is reset prior to the completion of the next instruction, the computation is interrupted and control is transferred to a pre-assigned routine. (See pp 3-1, -2.)

Examples

1. Addition of two positive numbers

```
1230000000                    2340000000
2340000000                    3620000000
3570000000  (no overflow)     6160000000  (overflow)
```

(Note that the first octal digit of each word contains the sign bit).

2. Addition of two negative numbers

```
7120000000                    4000000000
6300000000                    5000000000
5420000000  (no overflow)     1000000000  (overflow)
```

3. Addition of numbers of unlike sign (No overflow possible)

```
4010000000                    4110000000
3700000000                    3700000000
7710000000                    0010000000
```

```
4100000000
3700000000
0000000000
```

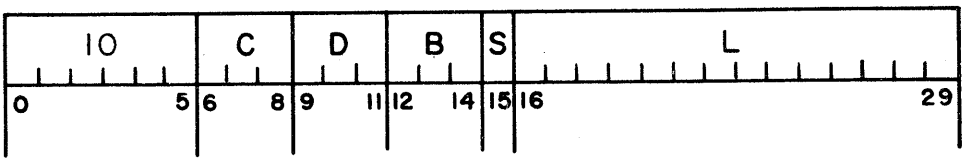
SECTION 2

INSTRUCTION FORMAT

The following sections define all the internal instructions of the 4100 and describe their execution and the time required by each.

A diagram representing the format of the instruction word is given for each instruction. (See following illustration.) Preceding this diagram is the unique three-letter code abbreviation of the instruction name, followed by the instruction name, itself, and the basic execution time. The numerical operation code, bits 0-5, is given in the octal number system, easily convertible to binary for reference to the bit pattern

ADD - Add 19.2 μs



interpreted by the machine. The numbers appearing beneath the diagram indicate the specific bit positions of each part of the instruction word.

The symbol L appearing in the diagram denotes the address part of the instruction and is usually interpreted as the address of a word in core storage. Other interpretations are described in connection with specific instructions.

The presence of the symbol B in any instruction word denotes the availability of a "tag" code, used for selecting a B-box, or index register. Except for instructions primarily concerned with the changing, testing, or storing of B-boxes (specifically noted), the content of the selected B-box, denoted as C(B), is algebrically added -- module 2^{14} -- to the value of L to determine the "effective address" of the instruction. In other words, L+C(B) is substituted for L throughout the description of all instructions except where noted. When the B-field is empty (B=0), no address modification takes place. When B=7, the location of the current instruction, C(IL), is added to L to obtain the effective address. (This type of address modification is defined as "relative addressing".) Intermediate values of B are used to specify any of the six B-boxes.

The symbols C and D, in the instruction word format, comprise a secondary instruction, executed simultaneously with the primary instruction denoted by the operation code. This secondary instruction is used as a program branching control, except as will be noted in

conjunction with specific primary instructions. The C code, in normal usage, specifies the condition for branching and/or subsidiary functions (See Table 2-1). The D code specifies the address, relative to the current instruction location, from which the next instruction will be obtained if the C condition is satisfied.

TABLE 2-1

NORMAL C INTERPRETATION		
C - Value	Branch Condition	Subsidiary Function
0 (D even)	No jump	None
0 (D odd)	No jump	Reset V
1	Initial $C(B)_{16-29} \neq 0$	Replace C(B) by C(B) - 1
2	Initial $C(B)_{16-29} \neq 0$	Replace C(B) by C(B) - 1; Do not add C(B) to L
3	Final $C(AC) \geq 0$	None
4	Final $C(AC) > 0$	None
5	Final $C(AC) \neq 0$	None
6	Final $C(AC) = 0$	None
7	Final $C(AC) < 0$	None

The symbol S, in the instruction word format, is the "suicide" code. The bit in the flag resister, F, corresponding to the priority level of the currently active program, is reset if, and only if, this bit contains a one.

Judicious use of these symbols in conjunction with the associated primary instructions can materially reduce the coding and running time of any program.

The description accompanying an instruction defines the manner in which it is executed when its tag B, conditional branch code C, destination code D, and suicide bit S, are equal to zero.

The shaded areas in the instructions represent fields that are not interpreted by the 4100. Unless otherwise noted, all instructions are subject to the variations delineated above.

Descriptions of the instructions will use the following special terms and definitions:

1. C(L) denotes the content of location L, where L refers to some location in storage. Similarly, C(AC) denotes the content of the accumulator register, and C(B) denotes the content of the B-box specified by B.

2. Subscripts refer to individual bit positions of a register or cell in core. For example, $C(B)_{16-29}$ is read "the contents of the bit positions 16-29 inclusive in the B-box core location".

3. The magnitude of a number is the number itself if it is positive and its two's complement if it is negative.

4. The word "place" in the title of an instruction always implies the transmission of a word from or to some location in core storage to or from some register (e.g., the accumulator or the B-boxes).

5. In all logical operations, the number is treated as a full 30-bit unsigned binary integer.

SECTION 3

4100 INPUT-OUTPUT

PRIORITY PROGRAM - INTERRUPT

The 4100 provides for servicing a multiplicity of input-output channels on a priority basis. With this "priority program interrupt" arrangement, peripheral devices having short holding times can be accommodated with no waste of computer time.

Associated with each priority level is a set of eight pre-assigned memory locations, as described in Section 1. For each priority level, the instruction location register contains the address of the next instruction that the 4100 is to execute when it enters that level. If a "demand" signal arrives from a channel while the computer is performing a lower priority program, it causes the computer to interrupt the active program and jump to the higher priority program. The machine logic ensures that the address of the next instruction of the interrupted program is saved in the IL register.

To maintain the integrity of interrupted programs, it is only necessary that each program detect no discontinuity in the C(AC). Therefore, each program which changes the C(AC) is responsible for preserving and restoring the original C(AC).

When the high priority demand is satisfied and control returns to the interrupted program, the latter proceeds oblivious to the interruption. In this way, the priority program interrupt feature enables the 4100 to interleave a number of different programs.

FLAG AND DEMAND REGISTERS

The priority program interrupt is implemented by means of two flip-flop registers, called the flag (F) and the demand registers. The flag register is composed of fifteen flip-flops, each of which functions as a priority request indicator, or flag. An implied sixteenth stage is permanently "set" and assumes control whenever no higher priority request is present.

In the recommended normal mode of operation, two priorities are reserved for service routines. A high-priority flag, selected by the operator, is set whenever:

1) the overflow indicator, V, has been set, and

2) V is not reset by the instruction which caused it to be set (page 2-2), and

3) V is not reset by the next instruction in the same priority level (page 8-1).

This feature can be used as an immediate automatic fail-safe program jump to an error analysis or alarm analysis routine, when unanticipated overflows occur during arithmetic operations. The lowest priority program, which is automatically activated when the flag register is cleared, is normally a self-check routine, which automatically exercises and tests the computer during "idle" time.

The remaining flags may be associated with input-output channels and may be set either under program control or at the command of a peripheral device. For the latter case, provisions are made at the computer terminal strip to connect the demand signal from each peripheral device to the "set" terminal of any particular priority flag. A flag can be reset only by an instruction in the program of corresponding priority.

The demand register has four stages. It is connected by logic to the flag register and always contains the binary representation of the priority of the program currently in effect (active). The demand register has two major purposes: its content is used in addressing the core memory executive space; and it is used in addressing external equipment. For the latter function, one lead from each demand register stage is brought out to the terminal strip. Each peripheral device decodes and responds to one of the sixteen possible combinations of states of these four wires. When the decoder is connected to the demand register at the terminal strip, the priority becomes an external address for its accompanying peripheral device.

INPUT-OUTPUT OPERATIONS

The transfer of information between the 4100 and ancillary equipment takes place through an input and an output bus. Each consists of thirty data lines that connect to the peripheral equipment at the computer terminal strip. Internally, the input data is received from the bus by the R-register, while output data is transmitted to the output bus by the G-register.

The 4100 exercises control over input and output by means of three instructions: PIM, PMO, and IOC. The PIM instruction transfers data from the input bus to memory. The PMO instruction transfers data from memory to the output bus (Exception: when B = 7, the PMO instruction transfers the accumulator content to the output bus). The IOC instruction transmits the instruction word to the output bus for peripheral device control.

When either a PIM or a PMO is executed, the C-code behaves as a subsidiary peripheral equipment address. (During any instruction, the C-code is held in stages 6, 7, and 8 of the I-register. One lead from each of these stages is brought out to the terminal strip. By connecting an external decoder to these terminals, these stages of I become part of the external address register.) This feature permits the computer to discriminate among several devices attached to the same priority flag, thus increasing the

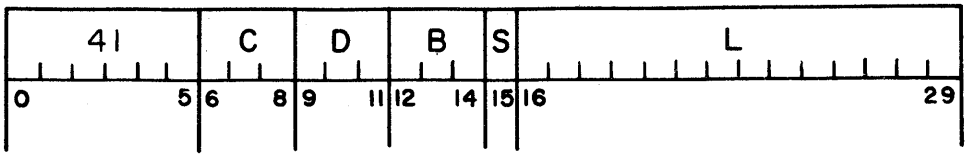
number of devices that the computer can service. It also allows the computer to transmit different instructions to one device.

The interconnection of the 4100 and any peripheral device involves the input or output bus, the two external address registers and three control signals. One signal is the demand signal from each external device to the flag register. A second signal is that generated by the computer during the PMO instruction, informing all peripheral devices that the computer has put information onto the output bus. The third control signal, emitted during execution of the PIM instruction, straddles the computer's internal strobe of the data from the input bus to the R-register. It is used to inhibit changes in the input data being transmitted.

SECTION 4

WORD TRANSMISSION OPERATIONS

PAM -- Place Accumulator in Memory 19.2 μs

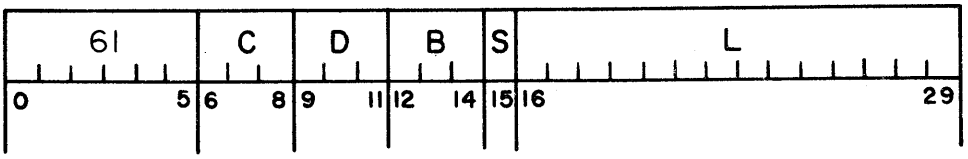


This instruction replaces the C(L) with the C(AC). Overflow is not possible. The C(AC) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

PMA -- Place Memory in Accumulator 19.2 μs

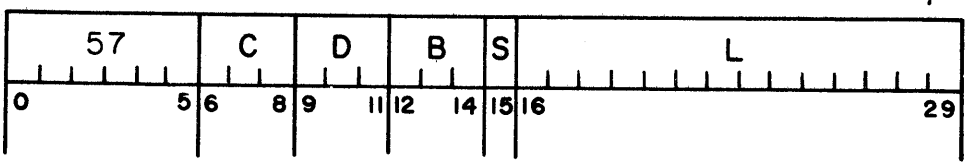


This instruction replaces the C(AC) with the C(L). Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

IMA -- Interchange Memory and AC 21.2 μs

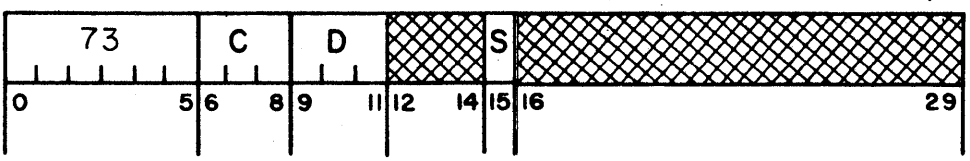


This instruction replaces the C(L) with the initial C(AC) and replaces the C(AC) with the initial C(L). Overflow is not possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

PZA -- Place Zero in Accumulator 12.8 μs

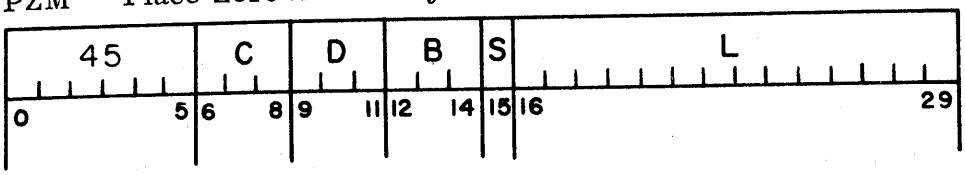


This instruction resets the C(AC) to zero. Overflow is not possible. Bits 16-29 of this instruction are not interpreted by the 4100.

Variations

C - Normal, except C = 1 or 2 is meaningless
D - Normal
B - Not interpreted
S - Normal

PZM -- Place Zero in Memory 19.2 μs

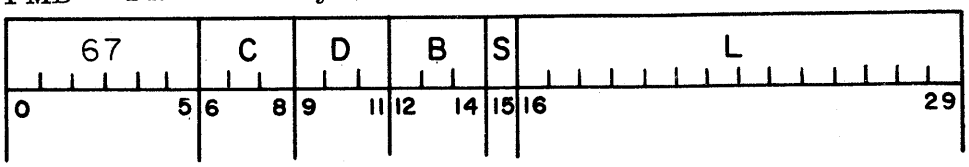


This instruction resets the C(L) to zero. Overflow is not possible. The C(AC) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

PMB -- Place Memory in B-Box 25.6 μs

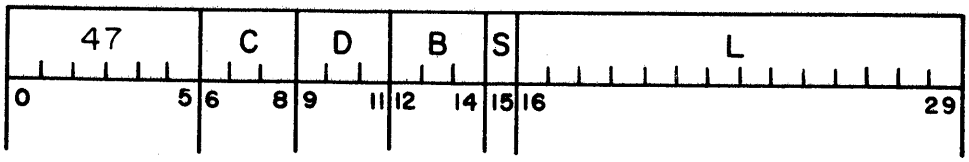


This instruction replaces the $C(B)_{0-29}$ with the $C(L)_{0-29}$. Overflow is not possible. The C(AC) and the C(L) are unchanged.

Variations

C - Normal, except C = 1 or 2 is meaningless
D - Normal
B - Specifies B-box to be changed
S - Normal

PBM -- Place B-Box in Memory 25.6 μs

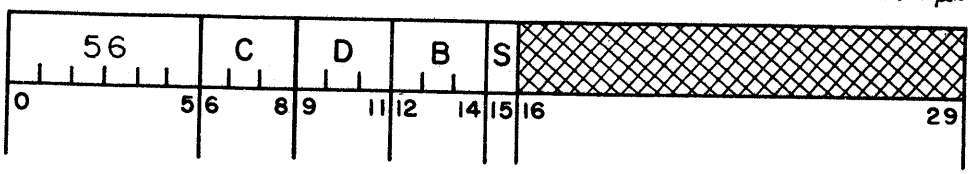


This instruction replaces the C(L) $_{0-29}$ with the C(B) $_{0-29}$. Overflow is not possible. The C(AC) and the C(B) are unchanged.

Variations

C - Normal, except C = 1 or 2 is meaningless
D - Normal
B - Specifies B-Box to be stored (B = 0 or 7 is meaningless)
S - Normal

PAB -- Place Accumulator in B-Box 19.2 μs

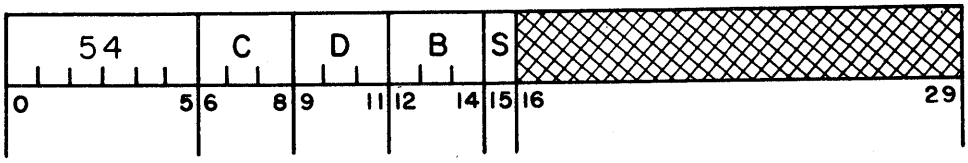


This instruction replaces the C(B) $_{0-29}$ with the C(AC) $_{0-29}$. Overflow is not possible. The C(AC) is unchanged. Bits 16-29 of this instruction are not interpreted by the 4100.

Variations

C - Normal, except C = 1 or 2 is meaningless
D - Normal
B - Specifies B-Box to be changed (B = 0 or 7 is meaningless)
S - Normal

PBA -- Place B-Box in Accumulator 19.2 μs



This instruction replaces the $C(AC)_{0-29}$ with the $C(B)_{0-29}$. Overflow is not possible. The C(B) is unchanged. Bits 16-29 of this instruction are not interpreted by the 4100.

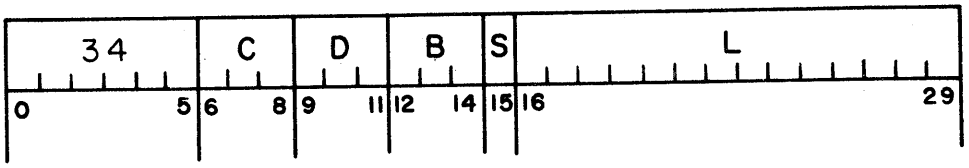
Variations

C - Normal, except C = 1 or 2 is meaningless
D - Normal
B - Specifies B-box to be fetched (B = 0 or 7 is meaningless)
S - Normal

SECTION 5

LOGICAL OPERATIONS

LAN -- Logical "AND" to Accumulator 19.2 μs

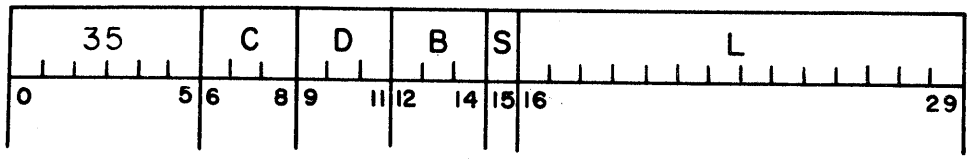


This instruction matches each bit of the C(AC) with the corresponding bit of the C(L). When the corresponding bits of both the C(L) and the C(AC) are ones, a one replaces that bit of the C(AC). When the corresponding bit of the C(L), the C(AC), or both is a zero, a zero replaces that bit of the C(AC). Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LAR -- Logical "AND" and Replace 19.2 μs

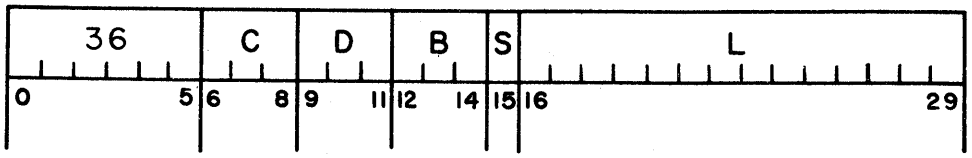


This instruction performs a LAN except that the result replaces both the C(AC) and the C(L). Overflow is not possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LEØ -- Logical Exclusive "ØR" 19.2 μs

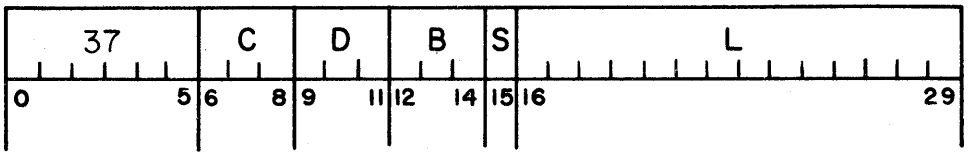


This instruction matches each bit of the C(AC) with the corresponding bit of the C(L). When the corresponding bits of the C(AC) and the C(L) are equal, a zero replaces that bit of the C(AC). When the corresponding bits of the C(AC) and the C(L) are unequal, a one replaces that bit of the C(AC). Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LER -- Logical Exclusive "ØR" and Replace 19.2 μs

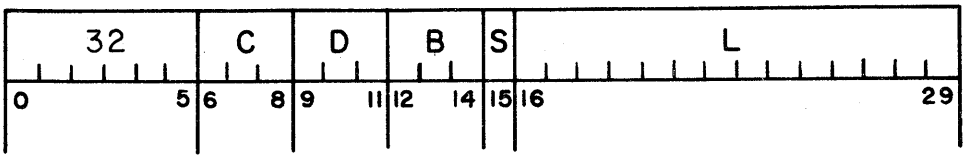


This instruction performs an LEØ except that the result replaces both the C(AC) and the C(L). Overflow is not possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LIØ -- Logical Inclusive "ØR" 19.2 μs

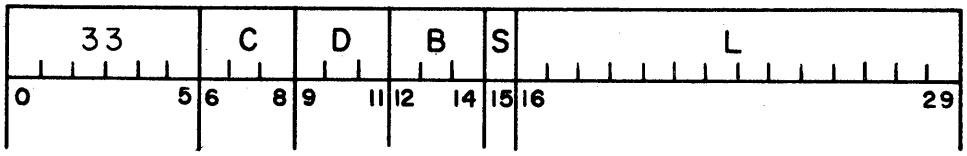


This instruction matches each bit of the C(AC) with the corresponding bit of the C(L). If the corresponding bit of the C(L), the C(AC), or both contains a one, a one replaces that bit of the C(AC). If the corresponding bits of both the C(L) and the C(AC) contain zeros, a zero replaces that bit of the C(AC). Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LIR -- Logical Inclusive "ØR" and Replace 19.2 μs

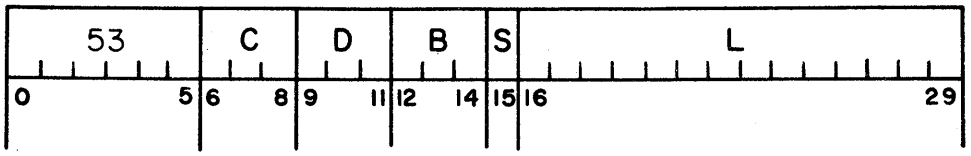


This instruction performs an LIØ except that the result replaces both the C(AC) and the C(L). Overflow is not possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

LIM -- Logical Inclusive "ØR" to Memory 19.2 μs



This instruction matches each bit of the C(AC) with the corresponding bit of the C(L). If the corresponding bit of the C(L), the C(AC), or both contains a one, a one replaces that bit of the C(L). If the corresponding bits of both the C(L) and the C(AC) contain zeros, a zero replaces that bit of the C(L). Overflow is not possible. The C(AC) is unchanged.

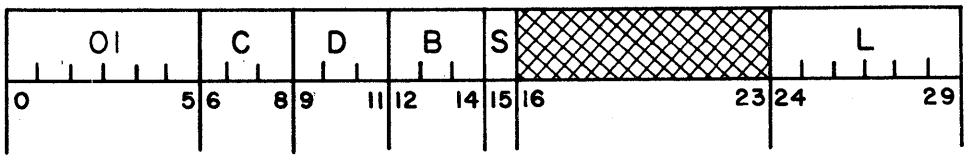
Variations

C - Normal
D - Normal
B - Normal
S - Normal

SECTION 6

SHIFTING OPERATIONS

SAL -- Shift Accumulator, Logical 16.8-76.8 μs



This instruction shifts the $C(AC)_{0-29}$ right or left L places. Bit 24 of the instruction word is interpreted as the algebraic sign of L.

If bit 24 is a one, bits 25-29 are interpreted as the two's complement of the negative (left) shift distance. Bits shifted out of the $C(AC)_0$ are discarded; vacated positions of the C(AC) are filled in with zeros. Overflow is not possible.

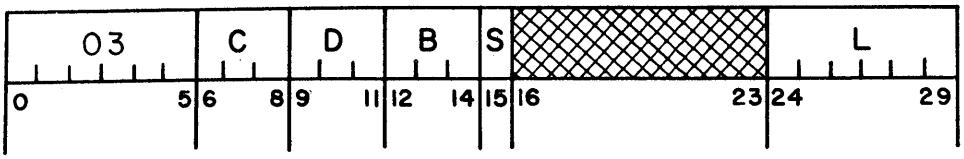
If bit 24 is a zero, bits 25-29 are interpreted as the positive (right) shift distance. Bits shifted out of the $C(AC)_{29}$ are discarded; vacated positions of the $C(AC)_{0-29}$ are filled in with zeros. Overflow is not possible.

Execution time is 16.8 μs for shifting one or two places left or one place right. Each additional two places, or portion thereof, requires 4.0 μs.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SAC -- Shift Accumulator, Circular 16.8-76.8 μs



This instruction rotates the $C(AC)_{0-29}$ right or left L places. The interpretation of L and the execution timing are as described under the SAL instruction. Overflow is not possible.

When rotating left, the $C(AC)_0$ replaces the $C(AC)_{29}$.

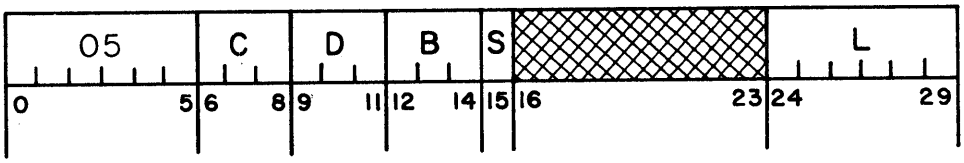
replaces the $C(AC)_0$.

When rotating right, the $C(AC)_{29}$ replaces the $C(AC)_0$.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SAA -- Shift Accumulator, Algebraic 16.8-76.8 μs



This instruction shifts the $C(AC)_{0-29}$ right or left L places. The interpretation of L and the execution timing are as described under the SAL instruction.

When shifting left, bits shifted out of the $C(AC)_0$ are discarded; vacated positions of the C(AC) are filled in with zeros. The overflow indicator is set if the $C(AC)_0$ changes during the process. In the absence of overflow, the action is equivalent to multiplying the C(AC) by a power of two.

When shifting right, bits shifted out of the $C(AC)_{29}$ are discarded; the $C(AC)_0$ is unchanged; vacated positions of the C(AC) are replaced by the $C(AC)_0$. Overflow is not possible. The action is equivalent to dividing the C(AC) by a power of two.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

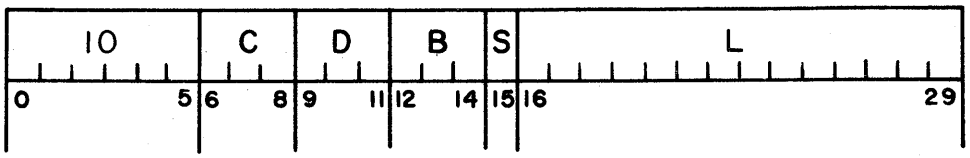
Shift examples

Operation	C(AC)	Notes
Initial	110 101 010 000 000 000 000 011 100 101	
SAL +3	000 110 101 010 000 000 000 000 011 100	
SAL -6	101 010 000 000 000 000 011 100 000 000	
SAC +12	011 100 000 000 101 010 000 000 000 000	
SAC -3	100 000 000 101 010 000 000 000 000 011	$-.7765377772_8$
SAA +3	111 100 000 000 101 010 000 000 000 000	$-.0776540000_8$
SAA -9	000 101 010 000 000 000 000 000 000 000	overflow
SAA -3	101 010 000 000 000 000 000 000 000 000	overflow

SECTION 7

ARITHMETIC OPERATIONS

ADD -- Add 19.2 μs

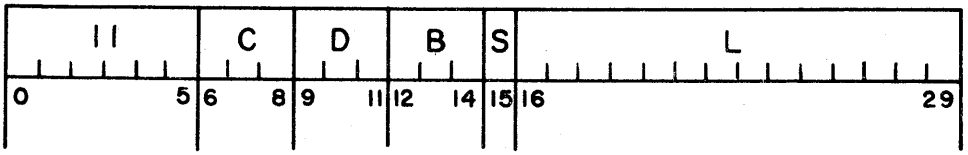


This instruction algebraically adds the C(L) to the C(AC) and replaces the C(AC) with the resulting sum. Overflow is possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

ADR -- Add and Replace 19.2 μs

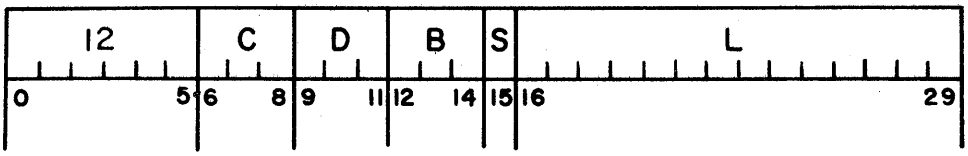


This instruction performs an ADD except that the result replaces both the C(AC) and C(L). Overflow is possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

ADM -- Add Magnitude 19.2 μs

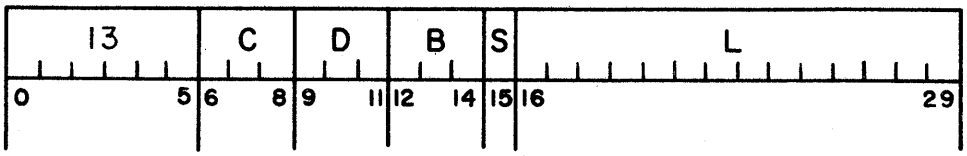


This instruction forms the absolute magnitude of the C(AC) and then performs an ADD. Overflow is possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

AMR -- Add Magnitude and Replace 19.2 μs

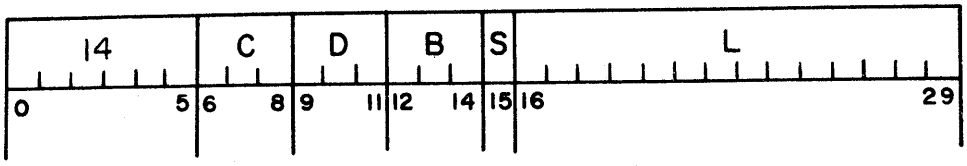


This instruction performs an ADM except that the result replaces both the C(AC) and C(L). Overflow is possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SBT -- Subtract 19.2 μs

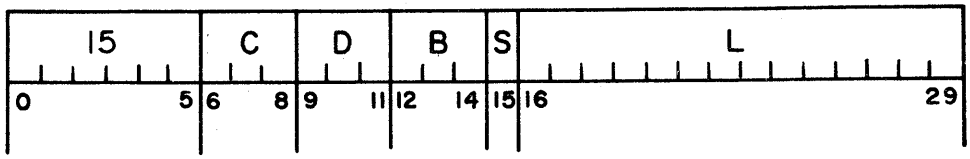


This instruction forms the two's complement of the C(AC) and adds it to the C(L), replacing the C(AC) with the resulting sum. Overflow is possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SBR -- Subtract and Replace 19.2 μs

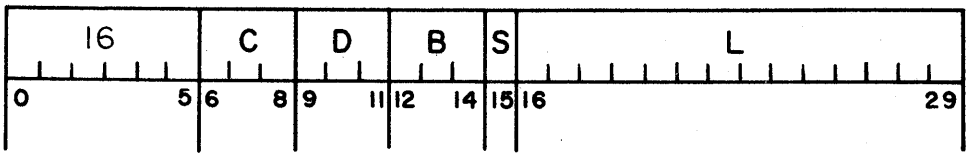


This instruction performs an SBT except that the result replaces both the C(AC) and the C(L). Overflow is possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SBM -- Subtract Magnitude 19.2 μs

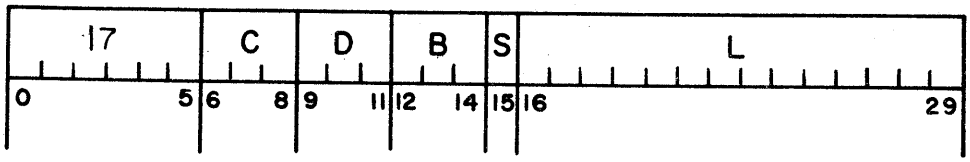


This instruction forms the two's complement of the absolute magnitude of the C(AC) and then adds it to the C(L). The resulting sum replaces the C(AC). Overflow is possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

SMR -- Subtract Magnitude and Replace 19.2 μs

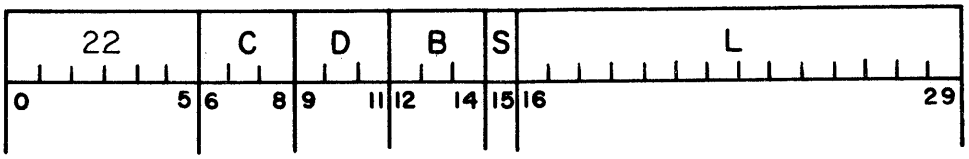


This instruction performs an SBM except that the result replaces both the C(AC) and the C(L). Overflow is possible.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

MPY -- Multiply 137.2 μs

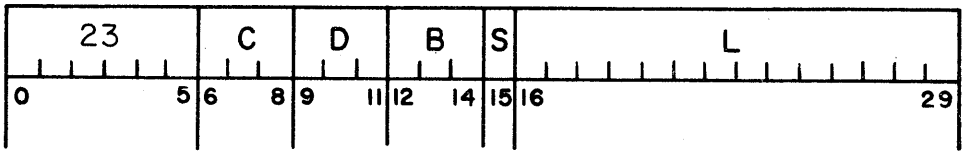


This instruction multiplies the C(AC) by the C(L) and replaces the C(AC) with the most significant 30 bits of the product. The less significant part of the product is discarded. Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

MPH -- Multiply Half-word 72.2 μs

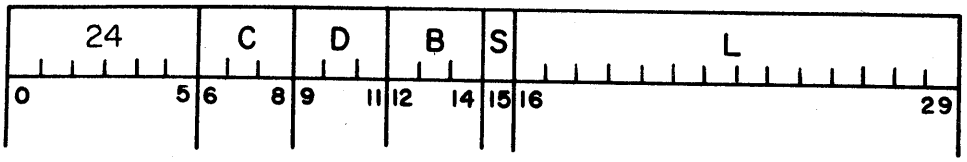


This instruction multiplies the $C(AC)_{0-14}$ by the C(L). The most significant 30 bits of the product replace the C(AC). The less significant part of the product is discarded. Overflow is not possible. The C(L) is unchanged.

Variations

C - Normal
D - Normal
B - Normal
S - Normal

DVD -- Divide 133.2 μs



This instruction divides the C(AC) by the C(L) and replaces the C(AC) with the resulting quotient. The C(L) is unchanged.

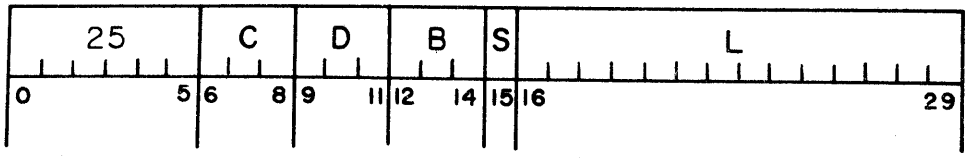
If the magnitude of the C(AC) exceeds the magnitude of the C(L), the quotient sign replaces the $C(AC)_0$, the $C(AC)_{1-29}$ is meaningless, and the overflow indicator V is set.

If the magnitudes of the C(L) and the C(AC) are equal, the quotient replaces the C(AC). The overflow indicator is set if the C(L) is negative. (The quotient is 1.000 000 000 000 000 000 000 000 000 00 if the operand signs were unlike; the quotient is 0.111 111 111 111 111 111 111 111 111 11 if the operand signs were like.)

Variations

C - Normal
D - Normal
B - Normal
S - Normal

DVH -- Divide Half-word 77.2 μs



This instruction performs a DVD except that the most significant 15 bits of the quotient replaces the $C(AC)_{0-14}$ and zeros replace the $C(AC)_{15-29}$. The C(L) is unchanged. This instruction has the same limitations as to overflow, etc., as has the DVD.

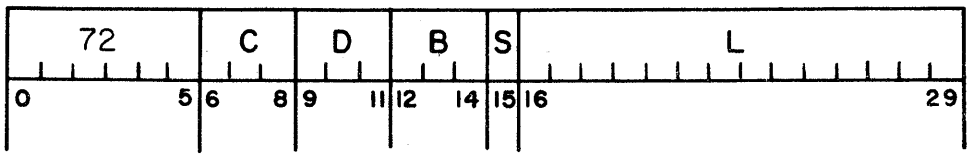
Variations

C - Normal
D - Normal
B - Normal
S - Normal

SECTION 8

CONTROL OPERATIONS

JMP -- Jump 12.8 μs



This instruction examines the condition specified by bits 6, 7, 8, and 11 of the instruction word. If the specified condition is satisfied, the 4100 takes its next instruction from the C(L) and proceeds from there. If the condition is not satisfied, the 4100 proceeds to the next instruction in sequence.

If bit 10 contains a one, and if the jump condition is satisfied, the jump destination is not modified by the C(B) and the location of this instruction, C(IL), replaces the $C(B)_{16-29}$. The $C(B)_{0-15}$ are meaningless.

Overflow is not possible. The C(AC) and the C(L) are unchanged.

Variations

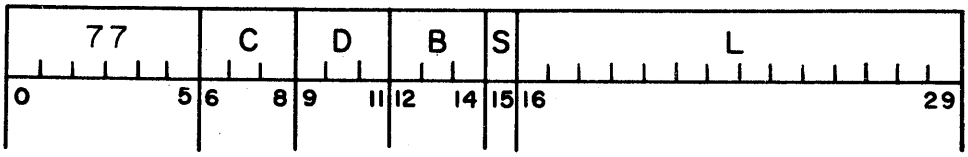
C - Normal, except C = 0, D even, specifies unconditional jump
C = 0, D odd, specifies jump if V = 1 and resets V

D - Bit 9 not interpreted
Bit 10 controls replacing the C(B) with the C(IL)
Bit 11 not interpreted, except if C = 0

B - If C = 1 or 2, or if bit 10 contains a one, B specifies B-box to be changed (B = 0 or 7 is meaningless) Otherwise, B interpretation is normal, and all values are permitted.

S - Normal

TST -- Test 19.2 μs

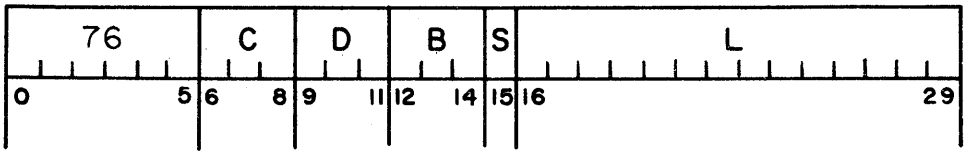


This instruction tests the C(L) as specified by the C-code of the instruction word. If the condition is satisfied, the 4100 jumps back D instructions and proceeds from there. If the condition is not satisfied, the 4100 proceeds to the next instruction in sequence. Overflow is not possible. The C(AC) and the C(L) are unchanged.

Variations

C - See Table 8-1
D - Normal
B - Normal
S - Normal

SKP -- Skip 19.2 μs



This instruction tests the C(L) as specified by the C-code (bits 6-8 of the instruction word). If the condition is satisfied, the 4100 skips the next D instructions and proceeds from there. If the condition is not satisfied, the 4100 proceeds to the next instruction in sequence. Overflow is not possible. The C(AC) and the C(L) are unchanged.

Variations

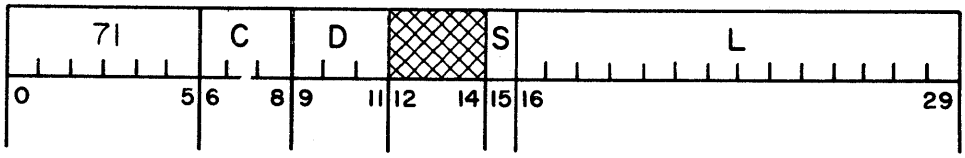
C - See Table 8-1
D - See preceding description
B - Normal
S - Normal

TABLE 8-1

SKP/TST C-CODES

C-value	Branch Condition
0	C(L) > 0
1	C(L) ≠ 0
2	C(L) = 0
3	C(L) < 0
4	C(L) > C(AC)
5	C(L) ≠ C(AC)
6	C(L) = C(AC)
7	C(L) < C(AC)

FLS -- Flag Set 12.8 μs

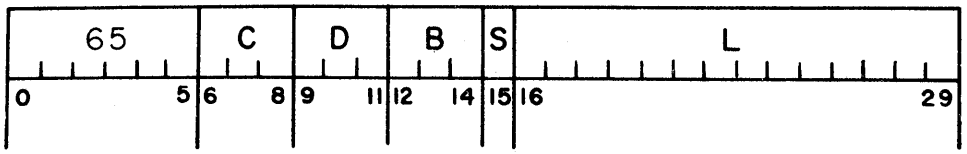


This instruction matches the bits L_{16-29} with the priority flags, F_{1-14}. For each bit of L which contains a zero, the corresponding bit of F is unchanged. For each bit of L which contains a one, a one replaces the corresponding bit of F. The bit of L corresponding to the priority flag of the current program has no effect. Overflow is not possible. The C(AC) and the C(L) are unchanged.

Variations

C - Normal
D - Normal
B - Not interpreted
S - Normal

ALB -- Add L to B-Box 19.2 μs



This instruction adds L (interpreted as a 14-bit unsigned integer) to the $C(B)_{16-29}$ and replaces the $C(B)_{1-14}$ and the $C(B)_{16-29}$ with the resulting sum. Overflow is not possible. The C(AC) and the C(L) are unchanged. The $C(B)_0$ and the $C(B)_{15}$ are meaningless.

If the sum is not less than 16384, the excess over 16384 replaces the $C(B)_{1-14}$ and the $C(B)_{16-29}$, yielding an effective subtraction.

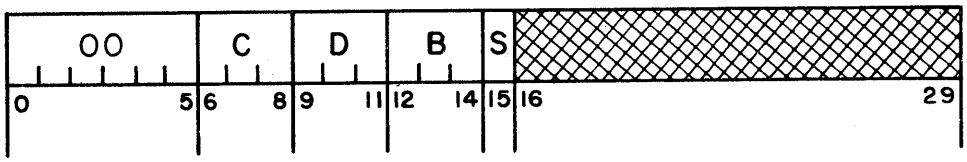
With C = 1, if the sum is not less than 16384, the 4100 loops back D instructions and proceeds from there. Otherwise, the 4100 proceeds to the next instruction in sequence.

Other C-codes are interpreted normally.

Variations

C - Normal, except C = 1 described above (C = 2 is meaningless)
D - Normal
B - Specifies B-box to be changed (B = 0 or B = 7 is meaningless)
S - Normal

HLT -- Halt

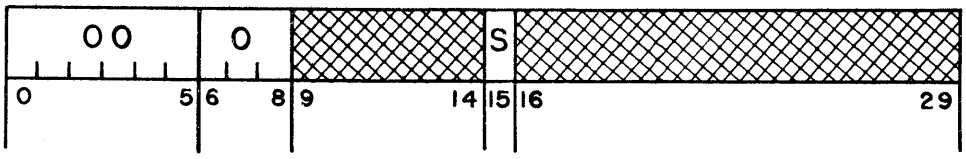


This instruction stops the 4100 if the rollback switch is off (see RLB operation). When started again, the 4100 proceeds to the next instruction in sequence, unless the C-code has caused looping or the S-code has caused suicide of this priority level. Bits 16-29 of the instruction word are not interpreted. Overflow is not possible. The C(AC) is unchanged.

Variations

C - Normal, except C = 1 is meaningless
D - Normal
B - May specify a B-box to be tested by C = 2
S - Normal

RLB -- Rollback Variable Time



This instruction reads one record from the rollback unit if the rollback switch is on (see HLT instruction), and stores it in sequential core locations. The first word replaces the content of the highest location in core storage; successive words enter progressively lower locations until the end of the record is detected. Since this action re-initializes the various instruction location counters, the next instruction to be executed will be obtained from a core location dependent on the status of the priority flags and on the initial value of the appropriate instruction location counter. (Because of the relatively long execution time of the RLB instruction, special attention should be devoted to the probability that external devices may set their priority flags during this interval.)

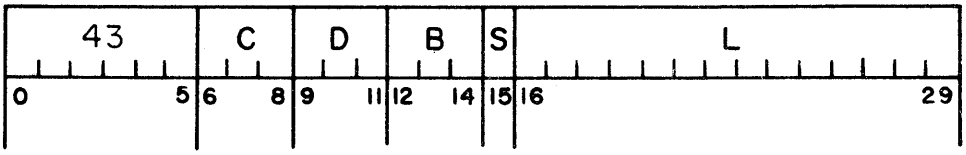
Variations

C - Must be zero
D - Not interpreted
B - Not interpreted
S - Normal

SECTION 9

INPUT-OUTPUT OPERATIONS

PIM -- Place Input Bus in Memory 19.2 μs



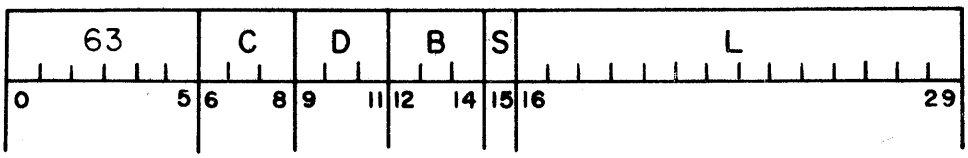
This instruction replaces the C(L) with the content of the input bus. Overflow is not possible. The C(AC) is unchanged.

If B $\neq$ 0 or 7, the $C(B)_{16-29}$ is tested for zero, and the $C(B)_{16-29}$-1 replaces the $C(B)_{16-29}$. If the initial $C(B)_{16-29} \neq 0$, the 4100 loops back D instructions. If the initial $C(B)_{16-29}$=0, or if B=0 or 7, the 4100 proceeds to the next instruction in sequence.

Variations

C - Subsidiary peripheral equipment address (see page 3-2)
D - Normal
B - Normal
S - Normal

PMO -- Place Memory on Output Bus 27.2 μs



This instruction transmits the C(L) to the output bus. Overflow is not possible. The C(AC) and the C(L) are unchanged.

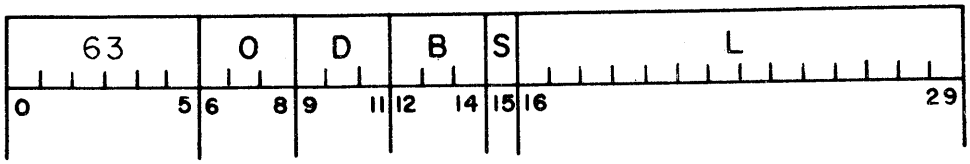
If B = 7, the C(AC) is transmitted to the output bus, and L is not interpreted. The C(AC) and the C(L) are unchanged.

If B ≠ 0 or 7, the $C(B)_{16-29}$ is tested for zero, and the $C(B)_{16-29}$-1 replaces the $C(B)_{16-29}$. If the initial $C(B)_{16-29} \neq 0$, the 4100 loops back D instructions. If the initial $C(B)_{16-29} = 0$, or if B = 0 or 7, the 4100 proceeds to the next instruction in sequence.

Variations

C - Subsidiary peripheral equipment address (C cannot be zero)
D - Normal
B - Normal, except B = 7 transmits the C(AC)
S - Normal

IOC -- Input-Output Control 20.8 μs



This instruction transmits itself to the output bus. Each peripheral buffer examines bits 16-21 of L to determine whether it is the buffer being addressed and, if so, to store, in the buffer, such remaining bits of the instruction word as it needs. Of those remaining, bits 12-14 are normally used to specify one of the peripheral devices attached to this buffer, and bits 22-29 (as required) control specific peripheral actions, such as advance, rewind, feed, backspace, etc. The buffer transmits an "accept" pulse to the 4100.

If the selected buffer is not available, it does not accept the control word or transmit an "accept" signal. Failure to receive an "accept" signal during any IOC instruction causes the 4100 to skip the next D instructions.

Variations

C - Must be zero
D - See above description
B - See above description
S - Normal

SECTION 10

PROGRAMMING EXAMPLE

The following example illustrates a number of the programming techniques discussed in the foregoing chapters.

FIXED-POINT SQUARE-ROOT SUBROUTINE

The method used to compute the square root of a fixed-point argument is a modified Newton-Raphson technique:

$$Y_{n+1} = Y_n - (Y_n - X/Y_n)/2,$$

where X is the argument,

$$Y_0 = (X + 1)/2,$$

and the iteration ceases when the quantity in the first parenthesis becomes less than 2^{-29}

The subroutine (included following this section) is entered by placing the argument, X, in the AC and executing the first instruction of the calling sequence. This transfers control to the square-root routine after first storing, in B-box No. 6, the location of this jump instruction.

The first instruction executed in the subroutine, PZM, clears the work space (SQRT - 21), simultaneously checking the argument for a negative sign (C = 7).

If the argument is negative, the square root of the absolute value is computed. The program jumps back two instructions (D = 2), forms the absolute magnitude of the argument by subtraction from zero, increments B-box No. 6 by one, and returns to the PZM instruction. The modification of B-box No. 6 prepares for the "error return" at the conclusion of the subroutine.

The next instruction executed, PBM, saves the content of B-box No. 5 in the work space (SQRT-22), simultaneously checking for zero argument (C = 6).

If the argument is zero, the program jumps back four instructions (D = 4) and executes a "normal return" with the correct root, zero, in the AC.

With a positive argument, the program next initializes B-box No. 5 to zero and enters a normalization loop: instructions SQRT-3, 4, 5.

During each iteration of the loop, the C(AC) is shifted left two bits and the C(B5) is incremented by one. When the C(AC) becomes greater than the content of (SQRT-19) -- i.e., one-quarter or greater --, the SKP instruction causes an exit from the loop and the next instruction stores the normalized argument in a work space, (SQRT-20).

The next three instructions form the initial value:

$$Y_0 = - (-1/2 - X/2)$$

The four instructions SQRT-10, 11, 12, 13 compute the correction term:

$$(Y_n - X/Y_n)/2$$

The C and D codes of the instruction at SQRT-13 cause an iteration, refining the estimated root until the correction term is less than 2^{-29}.

The resultant root is then placed in the AC and shifted right (using B-box No. 5) to compensate for the previous normalization procedure, leaving the desired square root in the AC.

B-box No. 5 is now restored to its original value and control is returned to the main program.

4100 SYMBOLIC CODING FORM

PROGRAM SQUARE RØØT SUBROUTINE (FIXED PØINT)

PROGRAMMER WEW | DATE 1961-10-3 | PRIORITY — | PAGE 1 OF 2

SYMBOL		OPN	S	ADDRESS ± MODIFIER	B	CD	COMMENTS (NO TAPE)
		JMP		SQRT	6	02	CALLING SEQUENCE
			E	RRØR RETURN			↓
			N	ØRMAL RETURN			
	+3	JMP		-2	6	00	
	+2	SBT		SQRT-2,1			
	+1	ALB		+1	6		
SQRT		PZM		SQRT-2,1		72	
	-1	PBM		SQRT-22	5	64	
	-2	PMB		SQRT-21	5		
	-3	SKP		-16	7	72	
	-4	SAA		-2			
	-5	ALB		+1	5	42	
	-6	PAM		SQRT-20			

RCA

4100 SYMBOLIC CODING FORM

PROGRAM SQRT (CONT'D)

PROGRAMMER WEW | DATE 1961-10-3 | PRIORITY — | PAGE 2 OF 2

SYMBOL		OPN	S	ADDRESS ± MODIFIER	B	CD	COMMENTS (NO TAPE)
	-7	SAA		+1			
	-8	SBT		-10	7		
	-9	SBR		SQRT-21			
	-10	PMA		SQRT-20			
	-11	DVD		SQRT-21			
	-12	SBT		SQRT-21			
	-13	SAA		+1		44	
	-14	PMA		SQRT-21			
	-15	SAA		0	5		
	-16	PMB		SQRT-22	5		
	-17	JMP		-2	6	00	
	-18	ØCT		2000000000			
	-19	ØCT		0777777777			
	-20						
	-21						