REFERENCE MANUAL

MODEL 21 PROCESSOR

SYSTEM TEN COMPUTER BY SINGER

SINGER
BUSINESS MACHINES

REFERENCE MANUAL

MODEL 21 PROCESSOR

SYSTEM TEN COMPUTER BY SINGER

Publication No. 42-2022
July, 1973

SINGER
BUSINESS MACHINES
2350 WASHINGTON AVE.
SAN LEANDRO, CALIF. 94577

PRINTED IN U.S.A.

The Model 21 Processor is an expanded version of the Model 20. This manual explains the differences between the two and covers the programming considerations required to utilize the new capabilities of the Model 21.

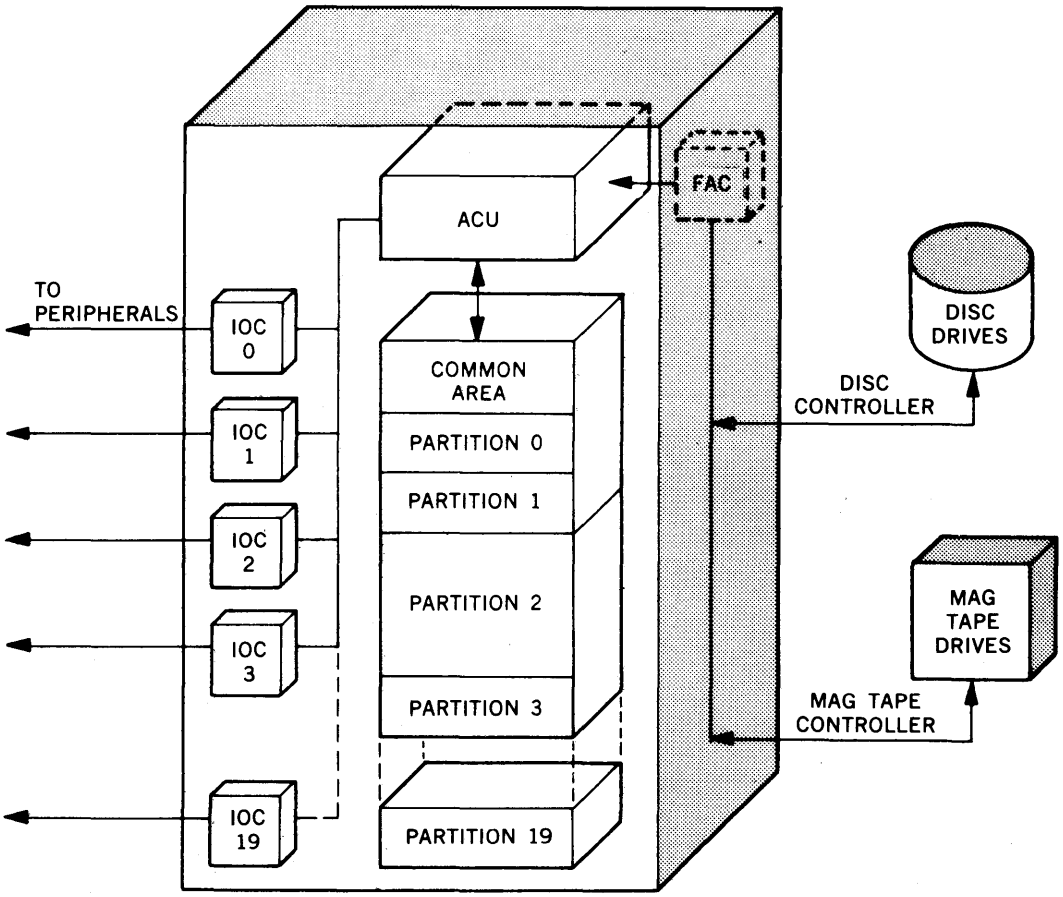


Figure 1. Processor (Block Diagram)

The Model 21 Processor of the SYSTEM TEN*computer by Singer is an enhanced version of the Model 20 Processor. The new capabilities are:

Expanded Common Core

Single level indirect addressing

A new instruction, Move Address

The privileged Common feature of the Model 20 is not available on the Model 21. The following discussion of processor operation applies to both models of the CPU unless specifically stated otherwise.

OPERATING SYSTEM

The central processor is a fixed partition, multiprogramming unit with a maximum of 20 partitions in the system. Each program is assigned to a fixed-size partition and each may use an area of core called Common. A software operating system as such is not required because processing is controlled by ACU hardware; however, a software program such as the Disc Management Facility (DMF) can be utilized for software control of programs. Partitions are processed in sequence, each being given a nominal 40 ms of ACU processing time. When this time has elapsed and a successful Branch instruction has been executed, the hardware switches control to the next higher partition. Partition switching can also be caused by a program instruction and by the initiation of an I/O operation in the host partition (the partition in control of the ACU at any given time). Figure 1 shows a schematic view of processor organization.

MEMORY ORGANIZATION

Core memory is purchased in 10K modules, each location accommodating one 6-bit character. The System Ten computer uses the 64-character ASCII subset shown in Appendix A. Core can be from 10K to 110K in size. Partition size can range from 1K to 10K in 1K increments. Common can be set at from 1K to 10K in the Model 20, from 1K to 65K in the Model 21.

Every location in memory is addressable. The lowest 300 locations in Common can be read but not written to; they contain the registers that are used by the hardware operating system (Figures 2 and 3). The lowest address in Common and in each partition is 0000; the highest is 9999 except for the Model 21, which uses 5-digit addresses in expanded Common.

*A trademark of THE SINGER COMPANY

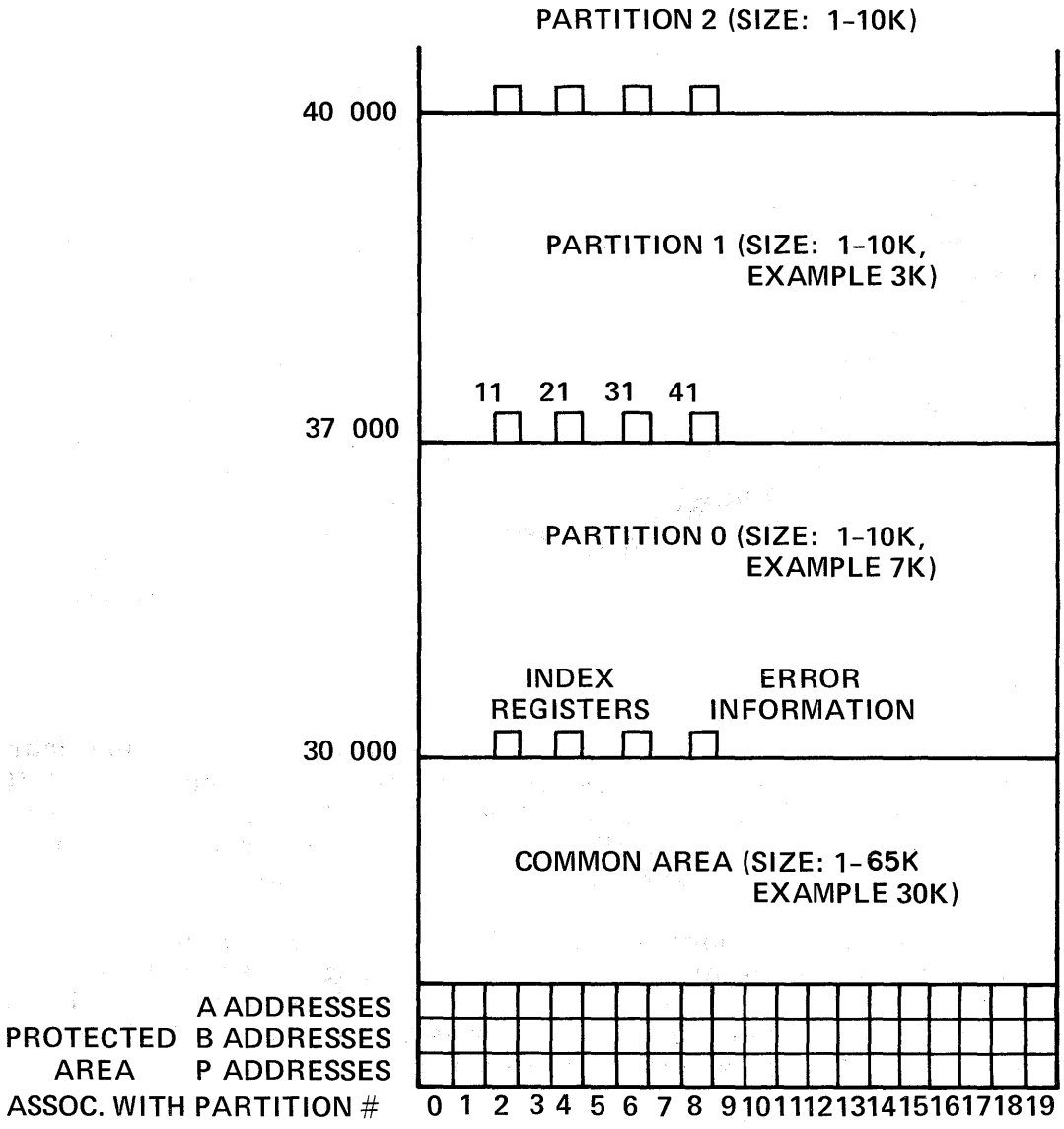


Figure 2. Memory Map

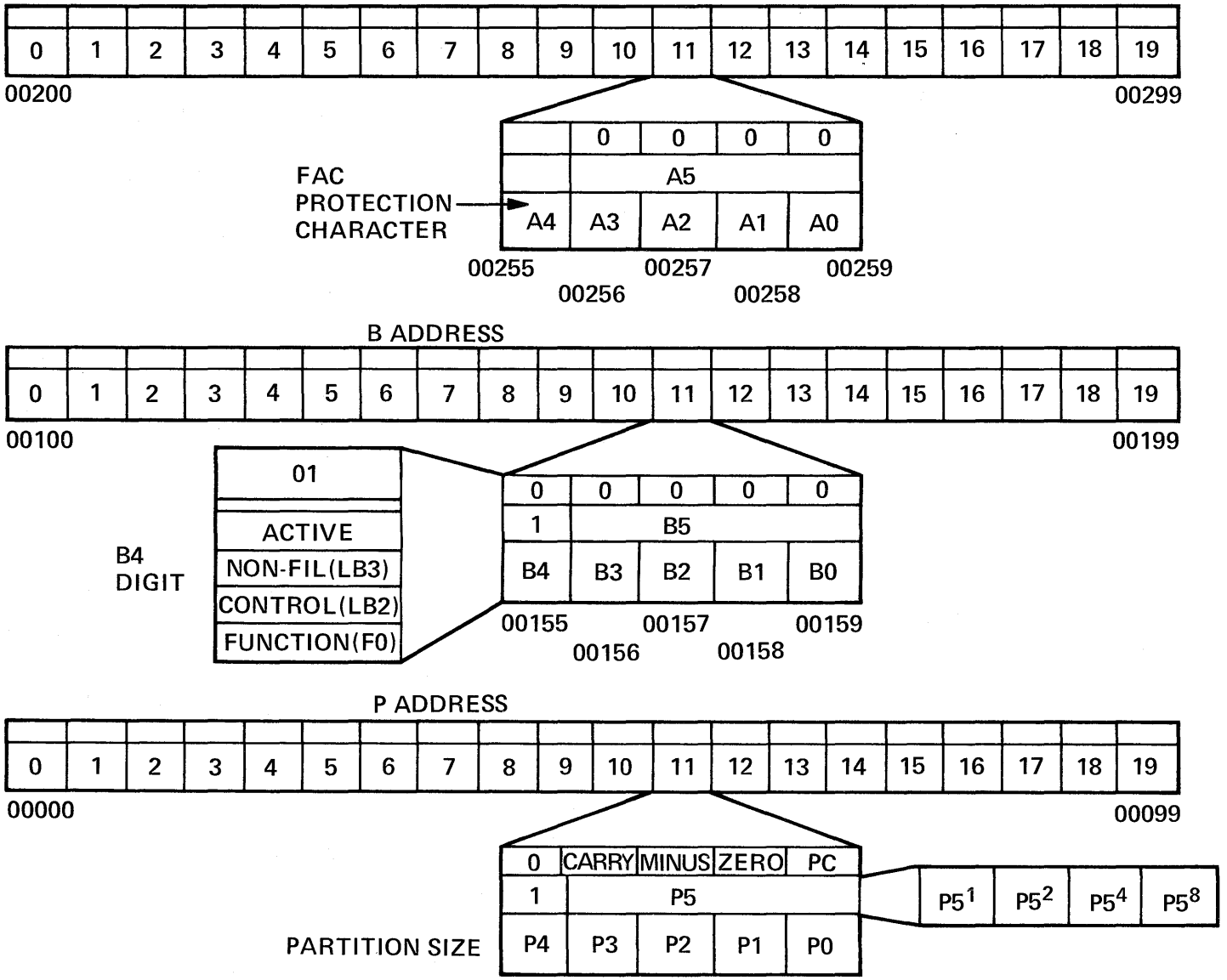


Figure 3. Hardware Protected Memory Area In Common

IOC/FAC OPERATION

The Multi-terminal Input/Output Channels (MTIOC) control all the peripheral devices except tape and disc units. There is one IOC per partition, with a maximum of ten devices per IOC. Each device on an IOC has a unique 1-digit address from 0-9; device 0 (zero) can be used to initiate program loading on an MTIOC partition. Each device can be addressed only by the program residing in the associated partition and only when that partition is in control of the ACU.

When a partition initiates an I/O operation, the hardware immediately switches control to the next higher partition. ACU processing time is not allotted to the partition initiating the I/O operation again until that operation is completed.

The actual transferring of data is handled on an "interrupt" basis. When a character is ready to be transferred between an IOC and the associated partition, the IOC interrupts normal processing long enough to complete the transfer, after which normal processing continues. When several partitions have I/O operations in progress simultaneously, it is possible for some of them to attempt to interrupt normal processing at the same instant in time. If this occurs, the hardware responds to the interrupts in ascending sequence starting with the host partition plus 1.

IOC input devices transmit data to the IOC in ASCII (ASCII code chart is shown in Appendix A). Generally speaking, the IOC converts the ASCII codes by eliminating bit 6 and passing bits 1, 2, 3, 4, 5, and 7 to main memory. During output, the ACU converts each internal character to its corresponding ASCII code. An exception to this occurs when the Write Control instruction is executed. In this case, the characters shown in columns 4 and 5 of the ASCII chart are converted to the ASCII codes for the corresponding characters shown in columns 0 and 1 respectively; characters shown in columns 2 and 3 are converted without changing identity.

The File Access Channel (FAC) controls up to four magnetic tape drives and up to ten logical disc drives (one Model 44 Disc Drive represents four logical drives). The FAC is a shared facility; the devices attached to it are available to all programs in all partitions.

Unlike IOC operations, FAC data transfer operations are not overlapped with normal ACU processing. When an I/O instruction has been sent to the disc controller or to the tape controller and has been acknowledged, the processor switches partitions and continues processing in the normal partition cycle until the FAC device is ready for data transfer. As soon as the data can be transferred (for instance, disc heads are on cylinder), the ACU is devoted to servicing existing IOC transfers and, on a cycle-stealing basis, the FAC until the entire FAC transfer is completed. During the short period of actual data transfer (approximately 500 µs for disc), the ACU does not service the IOC transfers; while the ACU is waiting for the data transfer, it services existing IOC data transfers on an interrupt basis.

Character conversion during FAC I/O operations differs from IOC conversion. For disc operations and for 7-track tape, no character conversion is necessary because data is stored by these devices in the 6-bit main memory form. Data for 9-track magnetic tape output operations is converted to ASCII and then written on the tape in a format compatible with that used by most other tape drives.

INSTRUCTION WORD FORMAT

Instruction word format for the Model 21 differs from that of the Model 20 only in bit 5 of each character; all other fields remain the same. Figures 4 and 5 show the two formats.

IDA and IDB indicate indirect addressing for the A and/or B addresses. An example is given in PROGRAMMING CONSIDERATIONS. Lack of a 5 bit (IDA or IDB = 0) will cause indirect addressing of the associated operand. Individual instruction execution time is increased 45.1 ms for each operand so addressed in an instruction.

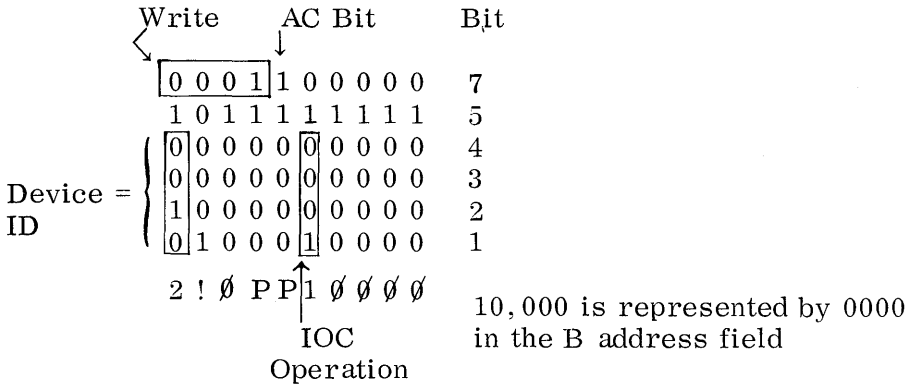
A5 and B5 are used to establish the ten-thousands digit for addresses in Common from 10,000 to 64,999. The three bits in A5 and B5 represent a binary number with a bit weight assignment as shown in Figure 5. Note that the values are inverted in the instruction word; that is, 50, which is 101, is written as 010 in core. The 5-bits over A0 and B0 are reserved for future expansion. The Move Address instruction will transfer the unit position 5-bit in the Model 21 Processor, but this bit may not be transferred in future processors. No programs should depend on this transfer occurring.

The Common core address 27,356 would be derived as follows. 7356 is specified in the normal way in bits 1-4; 20,000 is specified by the three 5-bits of the A5 or B5 field of the instruction word.

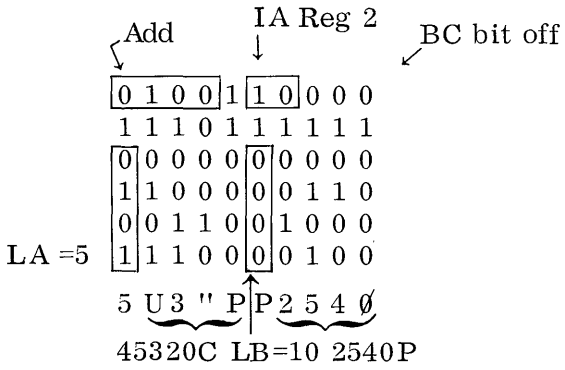
		Bits
20,000	1 0 1	5
	0 0 0 0	4
	1 0 1 1	3
	1 1 0 1	2
	1 1 1 0	1
	7 3 5 6	

20,000 is specified by 2 in the 10-thousands digit. With a bit weight of 124, this is actually 010. However, because the bits are inverted, it is 101 in the instruction word in core.

Example 1: A write to IOC device 2 from location 11,000 in Common for a length of 10,000 (the maximum value for an IOC data transfer); no indexing, no indirect addressing.



Example 2: Add Operand A, 5 characters from Common core location 45,320 as modified by the contents of register 2, to Operand B, the 10 characters located at partition address 2540.



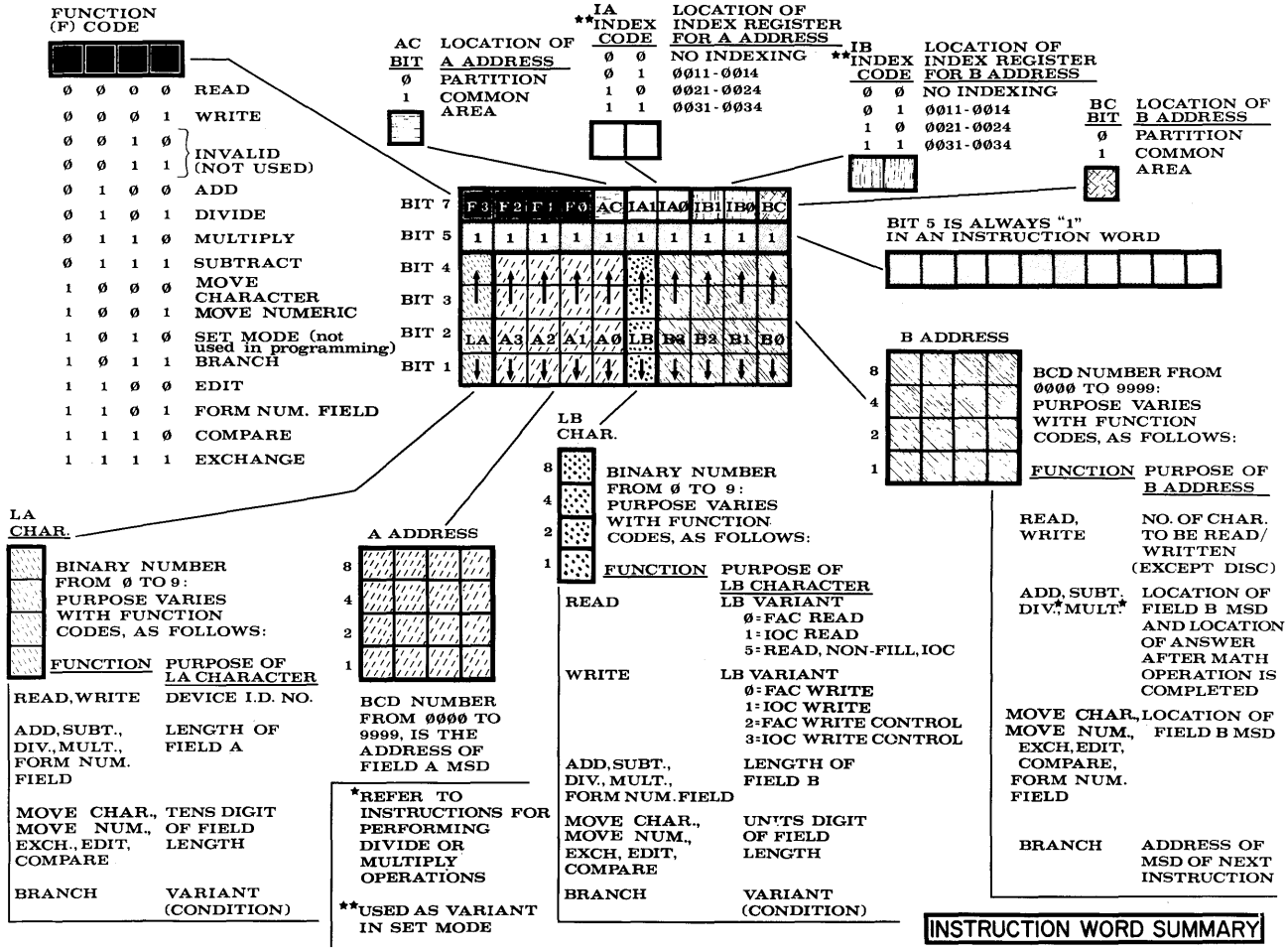


Figure 4. Model 20 Instruction Word

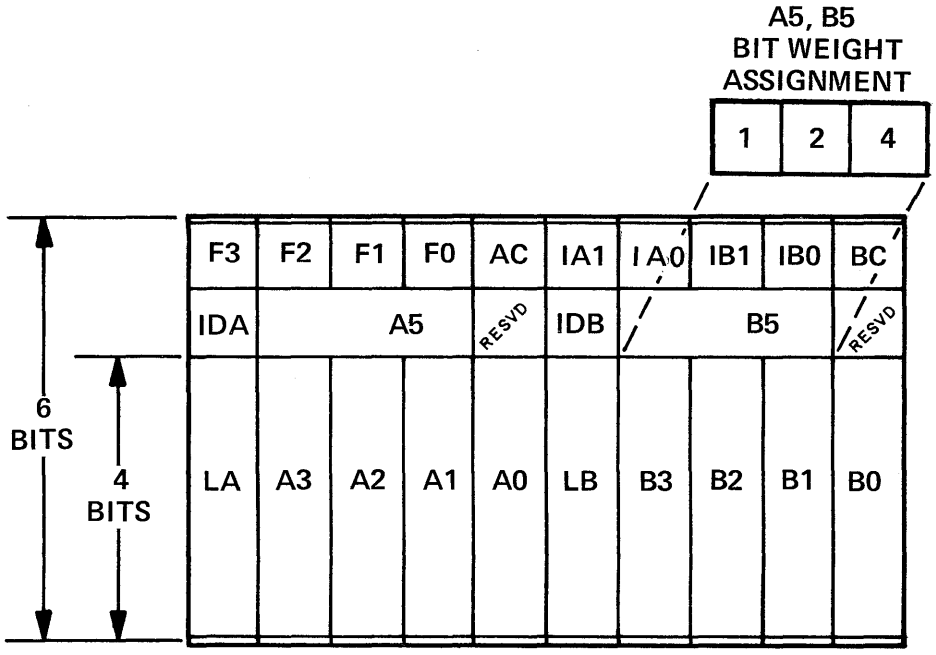


Figure 5. Model 21 Instruction Word

The Model 21 will affect programs in five ways.

1. A and B fields in instructions can refer to 5-digit addresses in expanded Common.
2. Direct or indirect addressing can be used for the A and B fields in instructions.
3. A new instruction, Move Address, is available.
4. There is no program check if a "1" is missing in a 5 bit of an instruction word.
5. Priviledged Common feature is not available.

FIVE-DIGIT ADDRESSES

The A and B fields in the instruction word can be expanded to five digit addresses by utilizing the A5 and B5 bits as shown in Figure 5. These five-digit addresses are restricted to expanded Common; they cannot exist in partition because the maximum partition size in 10K, locations 0000 - 9999.

Read/Write Operations

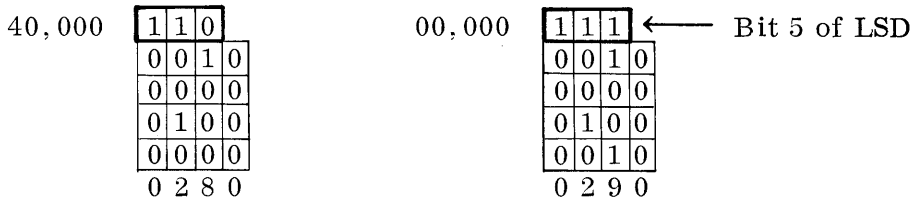
The B field of an I/O instruction contains the number of characters to read or write or the address in core (can be in expanded Common) that contains the six-digit disc address involved in the operation. The maximum number of characters allowed in an I/O data transfer is 10K except for disc where the number of characters transferred is automatically 100. 10K in the B field of the instruction word is 0000; the B5 field is 111.

Arithmetic Manipulation of A and B Addresses

Add/Subtract. When either of these operations is executed, only the numeric bits of the operands are acted upon, with the exception of the rightmost digit (LSD) of operand B of the instruction. The 7-bit of the LSD is the algebraic sign of the sum, and the 5-bit of the LSD is forced to one. Because the other zone bits of operand B are unchanged, there can be no automatic carry to the B5 character.

If the 5-bit of the LSD is part of an address in expanded Common, the address could be changed to a value quite different from that intended. The following examples clarify this possibility; Example 2 illustrates the recommended approach.

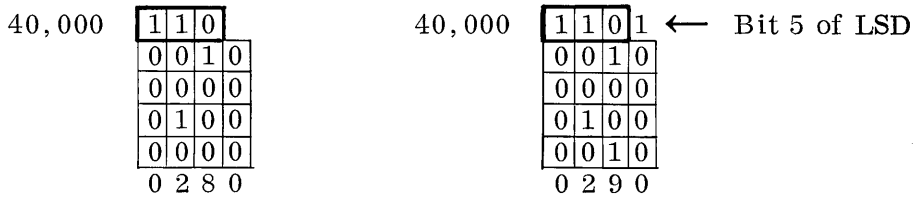
Example 1. Increment an address from 40,280 to 40,290 by adding 1 to the tens position. (The address to be modified is the B operand of the Add instruction.)



Before Addition After Addition

Note how the value of the B5 character has been changed because the 5-bit of the LSD of the B operand of the Add instruction was forced to 1.

Example 2. Increment an address from 40,280 to 40,290 by adding 10 to the units position.



Before Addition After Addition

In this case, forcing the 5-bit of the LSD of operand B to 1 has no effect on the B5 character and therefore does not result in a wrong address. This approach should always be used when modifying addresses in Common over 9,999.

Adding to or subtracting from an address whether by a specific instruction or by modifying it with an index register must always be restricted to within a 10K boundary because only the numeric portion of the address (bits 1-4) is acted on; there is no carry to the B5 character. For instance, if 43,560 is increased by a value of 8000, you will not get the desired address, 51,560.

43,560C Increased by 8000

B5	1 1 0	= 40,000	1 1 0	= 40,000 (unchanged)
4	0 0 0 0		0 0 0 0	
3	0 1 1 0	= 3560	0 1 1 0	= 1560
2	1 0 1 0		0 0 1 0	
1	1 1 0 0		1 1 0 0	

Address	18000	11000	12500
Added value	4000	9000	500
Address after addition	12000	10000	13000

Common core addresses are arithmetically negative in the instruction word because the AC or BC bit will be on to indicate a Common location; bit 7 on in the LSD of an arithmetic operand indicates a negative number. Thus, to add to such an address, you must add a negative value or subtract a positive one.

However, if the B address represents the number of characters to read from or write to an address in Common, bit 7 (the BC bit) can be either on or off, depending on where the instruction is assembled. If the instruction is assembled in Common, the BC bit will be on; if assembled in partition, the BC bit will be off even though referring to a location in Common.

When an index register is used to increment an address, the programmer does not have to be concerned with sign; the hardware operates on the numeric bits only. For instance, if register 1 contains 0005 and the operand is written 1000C(,1), the modified address is 1005C, not 9995C.

Multiply/Divide. During execution of these instructions, all the 5 bits of the B operand are forced to 1; A5 or B5 = 111 represent binary 000. Expanded Common addresses in the B field of a multiply or divide instruction will not be retained.

INDIRECT ADDRESSING

Indirect addressing is indicated in a machine instruction when the IDA and/or IDB bit is in an "OFF' state. In the Assembler II language, indirect addressing is indicated by prefixing a symbolic label with an ampersand (&).

When indirect addressing is indicated, the indirect address specified in the instruction (AØ through A4, A5 if Common, or BØ through B4, B5 if Common) is used to fetch another four-character address from memory. This address immediately replaces the initial indirect address. If indexing is also specified, the index value is added to the final address just prior to beginning normal instruction execution.

The AC/BC bit of the address fetched is also gathered. Therefore, a Common-resident instruction may indirectly address a partition location that contains a Common or extended Common address; the index is applied to this value (post indexing).

Add 45.1 ms to instruction execution time for each operand using indirect addressing.

INDIRECT ADDRESSING EXAMPLE:

```
EXIT        B       &BUCKET
            :
            :
BUCKET      DM      A'ENTRY'
            :
ENTRY       MC      A,B
```

The branch at symbolic location "EXIT" specifies an indirect branch to symbolic location "BUCKET". This causes the four character address of symbolic location "ENTRY" stored beginning at "BUCKET" to replace the original indirect address. In this example the branch passes control to symbolic location "ENTRY".

MOVE ADDRESS INSTRUCTION

Function Code	0011
LA, LB	Number of address characters to be transferred. LA is tens digit and LB is units digit. LA=0, LB=0 is a length of 100, the maximum.
A	Address of MSD of field to be moved
B	Address of MSD of field to which A is moved
AC/BC	"1" indicates a Common address
IA/IB	Indicate indexing
IDA/IDB	"Ø" indicates indirect addressing

The instruction is written

```
MA    A(L,IA),B(IB)
```

where

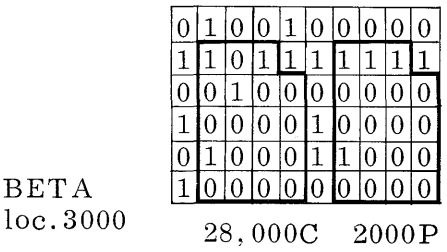
A is the field to be moved
B is the area to which the field is moved
L is the length (100 maximum)
IA/IB indicate indexing
Indirect addressing is indicated by an '&' in front of A and/or B as appropriate

Condidtion code 1 is set if a Common address is transferred; 3 if an address in partition. A branch on condition code 2 or 4 will never succeed.

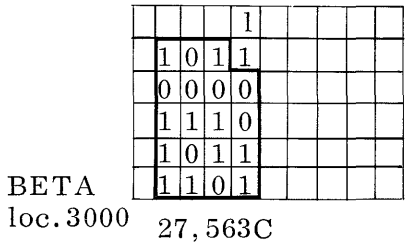
The Move Numeric instruction cannot be used to move expanded core addresses because it transfers only the lower 4 bits of each character; the A5 / B5 character would not be moved, nor would the AC/BC bit. The Move Address instruction includes these characters.

The transfer operation proceeds from left to right, transferring bits 1 through 5 only of each character in the A field to the corresponding position in the B field until the count is exhausted. All bits of the LSD (last character) of the A field are moved intact to B; if the AC or BC bit is in the last character moved, it will be preserved. Although the instruction will transfer 100 characters, four is the usual length. (Future hardware may not transfer the LSD 5-bit in the Move Address instruction, so such a transfer should not be depended upon when programming.)

```
27563C    TAX      DM    N5'0'
28000C    PRICE    DM    N5'0'
29000C    ALPHA    DM    A'TAX' (27563C stored in core in four-character
                         address format)
 2000     TOTAL    DM    N6'0'
 3000     BETA     A     PRICE (5), TOTAL(6)
 4000              MA    ALPHA (4), BETA+1
```



BETA before MA instruction



BETA+1

BETA after MA instruction

The instruction at BETA becomes essentially A TAX(5), TOTAL. Note that ALPHA is an address constant containing the address of TAX. Were the instruction written MA TAX(4), BETA+1, the contents of TAX rather than its address would be moved. The address of PRICE (28000C) is stored in BETA+1 in the instruction word (see Figure 4); the first character of the instruction word is LA. The MA instruction causes that address to be replaced with the address of TAX(27563C) so that the instruction will now add TAX to TOTAL rather than PRICE to TOTAL. The only 7 bit moved is that of the last character in the transfer; this does not affect the function code field of BETA but does transfer the AC bit of 27,563C, the address of TAX.

The following simplified example shows the use of the Move Address instruction in a Branch-and-Link operation.

```
52000C          BC    31(6),RTN(5)     The address of the next instruction
52010                                  (52010) is stored in Register 3
. . .
. . .
. . .
. . .
52100   RTN     MA    31(4),BETA+1     Contents of Register 3 (52010) are
. . .                                  stored in the instruction word at BETA
. . .
        BETA    BC    *(5)             The program returns to 52010, the
                                       instruction following the Branch and
                                       Link
```

DATA FAULT ANALYSIS

Because the fifth bit in an instruction word character for the Model 21 does not have to be "1", some unusual symtoms can occur if the programmer does not consider all aspects of this change.

1. Valid check characters are limited to those characters with a numeric binary value greater than nine.

2. When the processor is started up after having been powered down, it ordinarily continues processing where it left off. However, being powered down while in a load condition can cause difficulties, especially in 10K partitions, if the programmer does not take certain precautions.

When a 10K partition is in a Load Condition, the P Address register is pointing to location 9990 rather than 0000. A power down will cause the ACU to "forget" that a load request operation was in processes; when the processor is started up again, the ten characters at 9990 will be pulled for processing and will be executed if they are a valid instruction word. Because a 10K partition wraps around, the next instruction pulled will be that at 0000 - which may or may not be the desired bootstrap.

To be certain that the partition will go back to a load condition after power down, a check character such as a semi-colon should be placed at location 9990. A check character will automatically cause the partition to revert to Load Condition, ready to execute the bootstrap instruction when it is entered.

3. Previous Model 20 programs utilizing blanks as check characters should be modified (to semicolons, for example) to prevent possible program errors when run on the Model 21.

SOFTWARE: COMPATIBILITY AND MODIFICATIONS

Programs written for the Model 20 Processor may be executed on the Model 21 without change, except as noted above.

When writing or modifying programs using extended Common, the following points should be remembered. LIOCS makes extensive use of address modification and the relative to zero plus index technique of addressing. LIOCS does not support such fields as file control blocks and I/O work areas that are resident in Common. Therefore, all communications from the user program to LIOCS and from LIOCS to the user program must be via partition memory.

Because Assembler I will not support the assembly of indirect addressing or the Move Address instruction, it is suggested that all systems with the Model 21 Processor use Assembler II. Release F (and subsequent releases) of Assembler II supports the assembly of indirect addressing and the Move Address instruction when executing on a Model 21 with a minimum of 11K Common. Assembler II executes within its 9K-minimum partition; the minimum of 11K Common is required to verify that the CPU is indeed a Model 21.

DMF II basic software should be used to take full advantage of the Model 21. DMF II provides a "Common Memory Allocator:, which will make dynamic allocations of Common memory at user request. Macros are provided for this task.

ASCII CODE CHART

b7 → / b6 → / b5 → / BITS					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	b3	b2	b1	Column / Row	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

SYSTEM TEN INSTRUCTION TIMES

Execution Time (T) in Microseconds	
Key: TIX = 0.0, if IA and IB are both zero. TIX =58.9, if IA and IB are both non-zero. TIX =31.1, if IA or IB is non-zero. Add 45.1 for each operand using indirect addressing.*	
ADD	T = 42.2 + 3.3 (LA) + 10.0 (LB) + TIX + TOD, if LA is equal to or less than LB. T = 42.2 +11 (LA) + 12.2 (LB) + TIX + TOD, if LA is greater than LB. TOD = 0.0, if an overdraft does not occur. TOD = 10.0 (LB), if an overdraft occurs. An overdraft will always occur when the absolute value of Operand-A exceeds the absolute value of Operand-B and they have unlike signs.
BRANCH	T = 37.8 for no branch. T = 27.8 for branch to Address-A. T = 44.4 for branch to Address-B (except variants 6.7). T = 75.5 for "Link" (variant 6). T = 51.1 for "Branch on Service Request" (variant 7).
COMPARE	T = 40.0 + 7.78 (10LA + LB) + TIX, if the operands are identical. T = 48.9 + 7.78 (Y) + TIX, if the operands differ. Y = the number of equal comparisons.
DIVIDE	T = 46.67 + 1.11 (LA) + 26.67 (LB) + 22.22 (LA) (LB) + (10.0 + 11.1 (LA)) (S) + TIX. S = Sum of digits in quotient.
EDIT	T = 41.1 + 10.0 (LA + LB) + 6.67 (X1) + 3.33 (X2) + 2.22 (X3) + 2.22 (X4) + TIX. Key X1 = Number of '∂' signs in mask plus number of periods (.), commas (,), slash (/), and minus (-) signs before significance in Operand-B mask. X2 = Number of periods (.), Commas (,), slash (/), and minus (-) signs after significance in Operand-B mask. X3 = Number of significant digits in Operand-A. X4 = 0 for a negative operand. 1 for a positive operand. TIX = 0.0, if IA and IB are both zero. TIX = 58.9, if IA and IB are both non-zero. TIX = 31.1, if IA or IB is non-zero.
EXCHANGE	T = 38.9 + 13.3 (10LA + LB) + TIX.
FORM NUM	T = 43.3 + 3.33 (LA) + 7.78 (LB) + 2.22 (Z) + TIX, if LA - Z is equal to or less than LB. T = 45.55 + 1.11 (LA) + 10.0 (LB) + 4.44 (Z') + TIX, if LA - Z is greater than LB, causing an improper overflow. Key Z = Number of non-numeric characters in Operand-A. Z' = Number of non-numeric characters encountered in Operand-A before LB is filled.
MOVE ADR*	T = 38.9 + 11.1 (10LA + LB) + TIX
MOVE CHAR	T = 40.0 + 11.1(10LA + LB) + TIX
MOVE NUM	T = 40.0 + 11.1(10LA + LB) + TIX
MULTIPLY	T = 47.8 + 6.67 (LA) + 10.0 (LB) = ((10.0 + 11.1 LA) (S)) = TIX. S = Sum of digits in Operand-B.
READ	T = 91.1 + TIX for an Input/Output Channel (IOC). T = 73.3 + TIX for a File Access Channel (FAC).
SUBTRACT	T = 42.2 + 3.3 (LA) + 10.0 (LB) + TIX + TOD, if LA is equal to or less than LB. T = 42.2 + 11 (LA) + 12.2 (LB) + TIX + TOD, if LA is greater than LB. TOD = 0.0, if an overdraft does not occur. TOD = 10.0 (LB), if an overdraft occurs. An overdraft will always occur when the absolute value of Operand-A exceeds the absolute value of Operand-B and they have like signs.
WRITE	T = 91.1 + TIX for an Input/Output Channel (IOC). T = 73.3 + TIX for a File Access Channel (FAC).

*Model 21 Only

Touch & Know
Business Machines by
SINGER

Publication No. 42-2022