IBM

Instruction-Reference
1411 Processing Unit Instructions and Special Features

Preface

This manual contains descriptions of:

1. Central processing unit (CPU) instructions for the IBM 1410 Data Processing System.
2. Three special features (accelerator, priority, and program addressable clock) to the IBM 1411 Processing Unit.

The *IBM 1411 Input-Output Operations,* Customer Engineering Manual of Instruction, Form 223-2692, contains information on the overlap special feature and input-output instructions for the IBM 1410 system.

To supplement descriptions in this manual, examples are employed when necessary, and circuit controls, data flow charts, and timing charts are included. Because several instructions have similar functions and are executed in nearly the same manner, a timing chart for each instruction is not included, and instruction descriptions are not equally detailed. A chart in the "Appendix" lists information contained on each instruction. To fully understand the actions that the CPU performs when executing each instruction, this manual must be used with *IBM 1410 System Fundamentals,* Customer Engineering Instruction-Reference Manual, Form 223-2589.

This manual obsoletes:

1. Pages 80 through 137 and 459 through 463 in *IBM 1410 Data Processing System,* Customer Engineering Manual of Instruction, Form 225-6549-1.
2. *IBM 1410 Data Processing System, CPU Instruction Material,* Customer Engineering Manual of Instruction, Form R23-2587-1.
3. *IBM 1411 Processing Unit Instructions and Special Features,* Customer Engineering Instruction-Reference Manual (Preliminary Edition), Form R23-2698.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Customer Engineering Manuals, Dept. B96, PO Box 390, Poughkeepsie, N. Y.

Contents

Illustrations

The IBM 1410 Data Processing System uses stored program instructions to initiate all system operations. The format and contents of each instruction indicate the operation to be performed and, if required, the storage locations of data to be processed in the operation.

Instruction Formats

The basic 1410 instruction form is divided into four parts – the operation code, the A- or I-address or X-control field, the B-address, and a d-character. Because 1410 instructions are of variable length instruction form, instruction lengths can vary from one to twelve positions. Each instruction in a 1410 program must contain an operation code. However, the instruction may or may not contain other instruction parts as determined by the format requirements for that particular instruction. The following example shows the basic format for a typical 1410 instruction.

Operation Code	A- or I-address or X-control Field	B-address	d-character
x	aaaaa or iiiii or xxx	bbbbb	d

Each of the four parts in the basic 1410 instruction form designates information that the central processing unit (CPU) requires to perform an operation. An explanation of each instruction part follows:

1. The operation code (op code) is always a single character that specifies the basic machine operation to be performed. A word mark must be set over the operation code if the instruction is to be executed.

2. An instruction can contain either an A-address, I-address, or X-control field, but only one of the three. The A-address is always five characters and designates the location of A-field data in storage. The I-address is always five characters and specifies the address of an instruction in storage. The X-control field contains only three characters and is used only for input-output (I-O) operations.

3. The B-address is always five characters and designates the location of B-field data in storage.

4. The d-character is a single character at the end of the instruction used to establish a condition (or conditions) under which the CPU must perform the operation.

The IBM 1410 has a sequential method of program execution; thus, instruction 2 follows instruction 1, and so on, unless special circumstances during processing cause the CPU to alter the sequential execution of instructions.

Each instruction must have a word mark set over the op code, and must not contain word marks in any other position. Also, a word mark must be set in the core storage location immediately to the right of the last character of an instruction; this is normally the word mark associated with the op code of the next sequential instruction.

Instruction length checking is incorporated in the system to insure that each instruction read contains a valid number of characters for the operation code specified.

Valid instruction words vary in length from one to twelve characters depending on the amount of information required for the operation. The general instruction form consists of a single-character operation code followed by one or two five-character addresses, or a three-character input-output operation specification (X-control field), and in some cases, a single-character operation modifier. Valid instruction word lengths are:

O	= 1 position
O d	= 2 positions
O xxx d	= 5 positions
O aaaaa	= 6 positions
O aaaaa d	= 7 positions
O xxx bbbbb d	= 10 positions
O aaaaa bbbbb	= 11 positions
O aaaaa bbbbb d	= 12 positions

The O specifies an operation code. The five a's specify the five-character address of the A-field. The five b's specify the five-character address of the B-field. The three x's specify the x-control field, and the d specifies an operation modifier. Figure 1 lists acceptable lengths for 1410 instructions.

Instruction Decoding and Execution

A program step (an instruction) is read out of storage and decoded, and executed in two phases, instruction phase and execute phase.

Instruction Phase

During instruction phase, called instruction read-out time, the instruction is read out of core storage. Portions of the instruction are stored in various registers in the CPU; for example, the op code is stored in the operation register, addresses in the address registers, and the d-character in the operation modifier register. One storage cycle must be executed to read each character out of storage. Therefore, the number of

Instruction	Function	Acceptable Instruction Length
A (A) (B)	Add	1 6 11
S (A) (B)	Subtract	1 6 11
? (A) (B)	Zero and Add	1 6 11
! (A) (B)	Zero and Subtract	1 6 11
@ (A) (B)	Multiply	1 6 11
% (A) (B)	Divide	1 6 11
D (A) (B) d	Move Data	1 6 12
Z (A) (B)	Move Characters and Suppress Zeros	1 6 11
E (A) (B)	Edit	1 6 11
C (A) (B)	Compare	1 6 11
T (A) (B) d	Table Lookup	1 6 12
J (I) blank	Unconditional Branch	1 7
J (I) d	Test and Branch	1 7
R (I) d	Branch if I-O Channel Status Indicator On (Ch 1)	7
X (I) d	Branch if I-O Channel Status Indicator On (Ch 2)	7
B (I) (B) d	Branch if Character Equal	1 6 12
W (I) (B) d	Branch if Bit Equal	1 6 12
V (I) (B) d	Branch on Word Mark and/or Zone Equal	1 6 12
G (C) d	Store Address Register	7
, (A) (B)	Set Word Marks	1 6 11
¤ (A) (B)	Clear Word Marks	1 6 11
/ (B)	Clear Storage	1 6
/ (I) (B)	Clear Storage and Branch	1 6 11
.	Halt	1
. (I)	Halt and Branch	6
N	No Operation	1

Figure 1. Acceptable Lengths for 1410 CPU Instructions

storage cycles required during instruction phase is equal to the number of characters in the instruction.

Each character position in the instruction has a designated significance. The op code must be the first character in the instruction; the d-character must occupy the last position in the instruction; the A-address must be in the first five positions after the op code, etc. Because characters in the instruction are read from core storage one at a time, the I-ring advances to indicate the character being processed at the beginning of each storage cycle during instruction phase. The I-ring consists of 13 triggers labeled I-ring op, and I-ring 1 through I-ring 12. The op code and the length of the instruction determine the point to which the I-ring advances during instruction phase. Times between I-ring advances during instruction phase are called I-cycles. Figure 2 shows the manner in which the instruction move data (D aaaaa bbbbb d) is processed in instruction phase.

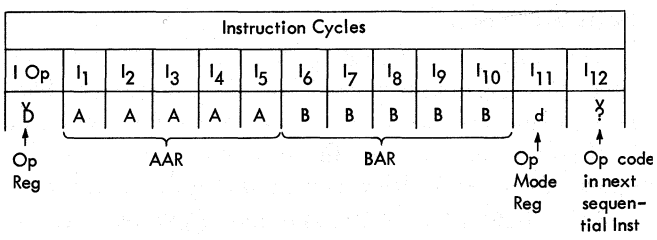


Figure 2. CPU Action in Instruction Phase for Data Move Instruction

During the first I-cycle, the I-ring is set to I-ring op, and the op code character is processed from core storage to the operation register. The single-character op code for each instruction conditions "common op code grouping" lines that:

1. Establish requirements for the length and contents of the instruction.

2. Condition checking circuits to determine whether the length and contents of the instruction are acceptable.

3. Condition and control circuit actions in the execution of the designated operation.

Figure 3 lists operation codes and their corresponding "common op code grouping" lines.

The 1410 System Fundamentals and *1411 Functional Units*, Customer Engineering Instruction-Reference Manual contains more detailed information on instruction read-out operation.

Execute Phase

At the completion of instruction phase, the CPU is conditioned to perform the designated instruction during execute phase. The length and complexity of execute phase is determined by the operation to be performed. In execute phase, the CPU takes a combination or series of A-, B-, C-, D-, E-, and/or F-cycles to read characters out of core storage (only one character is read out per cycle) and perform a step in the designated operation. The length of each cycle varies from 4.5 to 7.5 microseconds (or from 4.0 to 6.67 microseconds with the accelerator feature).

Characters are removed from storage in either ascending or descending order of core storage addresses. When the second scan latch is set, characters are unloaded from the low-order storage position to the high-order storage position or from the high-order position of the field to the low-order position of the field; unloading storage characters in this manner is called reverse scanning. An example of reverse scanning follows:

Assume that the field 93487 is located in storage positions 00100 through 00104; the high-order digit in the field (9) is stored in position 00100; the low-order digit in the field (7) is stored in position 00104. If the field is reverse scanned (second scan latch on), the 9 reads out first, the 3 second, and the 7 last. When the first or third scan latch is set, characters are unloaded from the high-order storage position to the low-order storage position or from the low-order position of the field to the high-order position of the field; unloading characters from storage in this manner is called forward scanning. An example of forward scanning follows:

	COMMON OP CODE GROUPING LINES	?	!	A	S	@	%	E	Z	C	W	V	/	.	,	¤	U	D	J	B	R	X	G	T	M	L	K	F	N
Instruction Read-Out	Percent Type Op Codes																X								X	X			
	Not Percent Type Op Codes	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X		X					
	Addr Dbl Op Codes	X	X	X	X							X		X	X				X		X	X							
	Not Addr Dbl Op Codes					X	X	X	X	X	X	X		X			X	X		X			X	X	X	X	X	X	
	1 Addr Plus Mod Op Codes																X		X		X	X	X						
	2 Addr No Mod Op Codes	X	X	X	X	X	X	X	X	X			X		X	X													
	2 Addr Plus Mod Op Codes										X	X						X		X				X	X	X			
	2 Address Op Codes	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X		X				X	X	X			
	Addr Type Op Codes	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
	2 Char Only Op Codes																										X	X	
	C Cycle Op Codes					X	X																	X					
	No C or D Cy Op Codes							X	X	X	X	X	X	X	X	X		X	X	X									
	No D Cy at I Ring 6 Ops	X	X	X	X			X	X	X	X	X	X	X	X	X		X		X				X					
	No Index On 1st Addr Ops																X						X		X	X			
Operational	Reset Type Op Codes	X	X																										
	Add or Subt Op Codes			X	X																								
	Mpy or Div Op Codes					X	X																						
	Add Type Op Codes	X	X	X	X																								
	Arith Type Op Codes	X	X	X	X	X	X																						
	E or Z Op Codes							X	X																				
	Compare Type Op Codes									X										X				X					
	Branch Type Op Codes *										X	X	X	X					X	X	X	X							
	No Branch Op Codes	X	X	X	X	X	X	X	X	X					X	X	X	X					X	X	X	X	X	X	
	Word Mark Op Codes														X	X													
	M or L Op Codes																								X	X			
Control	1st Scan First Op Codes	X	X	X	X	X	X	X	X	X	X	X	X		X	X				X			X	X					
	A Cy First Op Codes	X	X	X	X	X	X	X	X	X					X	X		X						X					
	Std A Cycle Op Codes	X	X	X	X	X	X	X	X	X								X						X					
	B Cy First Op Codes										X	X	X							X									
	A Reg to A Ch On B Cy Ops	X	X	X	X	X	X	X	X	X			X	X	X	X		X						X					
	Op Mod to A Ch On B Cy Ops										X	X					X		X	X	X	X					X	X	
	Load Mem On B Cy Op Codes	X	X	X	X	X	X	X	X									X											
	Rgen Mem On B Cy Op Codes									X	X	X		X					X	X	X	X		X			X	X	
	Stop at F on B Cy Op Codes	#	#			#	#						X	X				#	X		X	X							
	Stop at H on B Cy Ops										X	X								X									
	Stop at J on B Cy Op Codes	X	X			X	X	X	X									X											
	RO B AR On Scan B Cy Ops							X	X	X	X	X	X	X	X	X		X	X	X	X	X		X					
	RO A AR On A Cy Ops	X	X	X	X			X	X	X					X	X		X											

* Not a Line Name, a Grouping Only. # Indicates Accelerator Feature Timing.

Figure 3. 1410 Common Op-Code Grouping Lines (13.14.01-14)

Assume that the field 93487 is located in storage positions 00100 through 00104; the high-order digit in the field (9) is stored in position 00100; the low-order digit in the field (7) is stored in position 00104. If the field is forward scanned (the first or third scan latch on), the 7 reads out first, the 8 second, and the 9 last.

At the completion of execute phase, the CPU normally returns to instruction phase to read out the next instruction. CPU actions during execute phase are described for specific operations in other sections of this manual.

Questions on 1410 Instruction Formats and Decoding

Answers to these review questions are in the Appendix.

1. List the four parts into which the following 1410 instruction can be divided: B 09000 08000 M

2. What instructions contain X-control fields?

3. Does any 1410 instruction contain both an A- and I-address or both an X-control field and an A- or I-address?

4. Over what position in the instruction must a word mark be set?

5. What purpose does the I-ring serve during instruction phase?

6. Are common op code grouping lines conditioned during instruction phase or execute phase?

The add, subtract, zero and add, zero and subtract, multiply, and divide instructions are used to perform arithmetic operations in the IBM 1410 Data Processing System. The use of add-to-storage logic in the system eliminates the need for special purpose accumulators or counters. Because any group of storage positions can be used as an accumulating field, the capacity for arithmetic functions is not limited by a predetermined number of counter positions.

All arithmetic functions are performed under complete algebraic sign control. The combination of zone bits in the units position of the fields that the arithmetic instruction specifies determines the sign of the factor. Figure 4 shows the four possible combinations of zone bits and the values of the signs they represent. The standard machine method of signing a field is to indicate a positive factor with both A- and B-bits; a B-bit represents a negative factor.

Sign	BCD Code Bit Configuration	Card Code Configuration
Plus	No A- or B-Bit	No Zone
Plus	A- and B-Bits	12 Zone
Minus	B-Bit Only	11 Zone
Plus	A-Bit Only	0 Zone

Figure 4. Bit Equivalents for Signs

The arithmetic operations in the IBM 1410 are performed by using one of two types of add scans incorporated in the system. The two types of add scans are: true add and complement add. The type of add scan performed is determined by the arithmetic operation and the signs of the factors involved. In an algebraic subtract, recall that the sign of the subtrahend (A-field) is changed and added to the minuend (B-field). The sign of the result is the sign of the greater value only after the A-field is considered to have been changed.

In all arithmetic operations, the presence of characters represented by the card codes of blank, 8-3, 8-4, 8-5, 8-6, and 8-7 in the numeric portion of a field are treated as 0, 3, 4, 5, 6, and 7, respectively.

If the result in an arithmetic operation exceeds the B-field limit imposed by the B-field word mark, the carry is lost, and the arithmetic overflow indicator turns on. The test and branch instruction, J(1)Z, tests and turns off the arithmetic overflow indicator.

If the result of any add, subtract, multiply, zero and add, or zero and subtract operation is zero, the zero balance indicator turns on. The next add, subtract, multiply, zero and add, or zero and subtract instruction, that does not result in a zero balance, turns off the zero balance indicator.

Add and Subtract Instructions

Instruction Formats

Formats for the add and subtract instructions are:

Op code	A-address	B-address
Å (add)	xxxxx	xxxxx
Å (add)	xxxxx	
Å (add)		
Š (subtract)	xxxxx	xxxxx
Š (subtract)	xxxxx	
Š (subtract)		

If the add or subtract instruction specifies two addresses (Å or Š xxxxx xxxxx), the numeric data in the A-field is algebraically added to (add operation) or subtracted from (subtract operation) the numeric data in the B-field. The result is stored in true form in the B-field. Except for the sign position which may be changed, B-field zone bits are undisturbed. A-field zone bits are ignored in all other positions except the sign position. A B-field word mark stops the operation and must be set over the high-order position of that field. If the A-field is shorter than the B-field, it, too, must have a defining word mark to stop transmission of data from the A-field to the B-field. When the A-field is shorter than the B-field, the system automatically adds zeros to (add operation) or subtracts zeros from (subtract operation) the extra high-order positions of the B-field until a B-field word mark is detected. If the A-field is longer than the B-field, the high-order positions of the A-field, that exceed the limits imposed by the B-field word marks, are not processed.

If the add or subtract instruction has only an A-address (Å or Š xxxxx), the A-field is added to (add operation) or subtracted from (subtract operation) itself. The result is stored in the A-field. Add operations in which the instructions designate only one field are always executed with true add cycles and the sign bit configurations of the results are always the same as the original sign of the A-field. When the subtract instruction designates only one field, the numeric portion of the A-field is always 0 after the operation, but zones in the A-field remain unchanged; the A-field sign bit configuration is the same as it was before the operation.

If the add or subtract instruction does not designate an A- or B-address (no address chaining), the contents of the AAR from the previous operation specify the A-field, and the contents of the BAR specify the B-field in the add or subtract operation. The operation is executed in the manner described for two address add and subtract instructions.

CPU Operation

During last instruction read-out cycle, the units control, first scan control, and A-cycle control latches are set to initiate the first cycle (A-cycle) of the add or subtract operation. In the first A-cycle, the units position of the A-field reads out of storage and is gated to the A-data register. The operation (op) modifier register is gated to the A- and assembly channels on A-cycles to satisfy the validity check circuits.

During the subsequent B-cycle, the units position of the B-field reads out of storage onto the B-channel, and the A-field character in the A-data register is gated to the A-channel. The signs of the A- and B-channel characters are analyzed to determine whether the A-field should be true or complement added to the B-field. The first B-cycle is one logic gate longer than other B-cycles in the operation to allow time to condition the true or complement add controls.

Whether the system executes a true or complement add scan is determined by the number of minus signs in the factors and the type of operation being performed (Figure 5).

Type of Operation	A-Field Sign	B-Field Sign	Type of Add Cycle	Sign of Result
ADD	+	+	True Add	+
		-	Complement Add	Sign of greater value
	-	+	Complement Add	
		-	True Add	-
SUBTRACT	+	-	True Add	-
		+	Complement Add	Sign of greater value (after A-field sign is changed as a result of the subtract instruction)
	-	-	Complement Add	
		+	True Add	+

Figure 5. Types of Add Cycles and Sign of Result for Add and Subtract Operations

Numeric bits in the A- and B-channel characters are added in the adder unit. The adder output feeds the assembly unit where B-channel zones and word mark and adder numerics are combined. The character is than sent to the B-field location in storage. If there is an adder carry and no A- or B-channel word mark, the carry latch is set and combined with the sum of the binary portions of the A- and B-channel digits on the next B-cycle.

The A- and B-channel characters are added or subtracted until an A- or B-channel word mark is sensed. If an A-channel word mark is detected before a B-channel word mark is sensed, indicating that the B-field is longer than the A-field, the extension latch is set and a series of B-cycles are executed until the operation is complete (B-field word mark). Remaining characters in the B-field are combined with zeros on true add or nines on complement ad scans. The 0 or 9 is inserted directly on the A-channel input to the adder.

If a B-channel word mark is sensed before an A-channel word mark is detected, indicating that the A-field is longer than the B-field, other A-field characters are not processed. The following example illustrates this action:

ADD +1099 (A-field) + +$\overset{v}{1}$00 (B-field) = +1199

A-field 1099

B-field $\overset{v}{1}$00

B-field after add operation is complete. High-order digit (1) in A-field is not processed. B-field answer is incorrect. $\overset{v}{1}$99

When there is an adder carry from the high-order position on a complement add scan, the add or subtract operation ends when a B-channel word mark is sensed as shown in the following example:

ADD −17 (A-field) + +$\overset{v}{9}$00 (B-field) = +883

Because the A-field sign is negative and the B-field sign is positive, the A-field must be complement added to the B-field.

Complement of A-field 982

B-field $\overset{v}{9}$00

Carry latch is always on for units position of complement add 1

B-field after add operation is complete. $\overset{v}{8}$83 + carry

When the B-channel word mark is sensed, the add operation ends and the carry is lost. The operation shown is correct.

An adder carry from the high-order position on a true add scan signifies an overflow and sets the arithmetic overflow latch. Study the following example:

ADD +99 (A-field) + +$\overset{v}{9}$0 (B-field) = 189

A-field 99

B-field $\overset{v}{9}$0

B-field after add operation is complete. Arithmetic overflow latch is set. B-field answer is incorrect. $\overset{v}{8}$9

Because the low-order position of the next field was read out of storage on the last B-cycle of the operation, the digit written in storage on the last B-cycle is not known. The original B-field should have contained three positions: (0$\overset{v}{9}$0).

When the stored result is in complement form, as indicated by no adder carry from the high-order position on a complement add scan, the characters in the

B-field must be converted to true form and the B-field sign changed. The following example illustrates this action:

ADD	−18 (A-field) +	+$\overset{\vee}{0}$12 (B-field) = −006
Complement of A-field		981
B-field		$\overset{\vee}{0}$12
Carry latch is always on for units position of complement add		1
B-field after complement add scan.		$\overset{\vee}{9}$94
The B-field result is in complement form. No adder carry from the high-order position initiates another forward scan by setting the third scan latch. The B-field is converted to true form and added to zeros inserted on the A-channel input to the adder. The B-field sign is inverted.		
B-field after complement add scan		+$\overset{\vee}{9}$94
Recomplement B-field, and change B-field sign		−$\overset{\vee}{0}$05
Insert zeros on A-channel input to the adder		000
Carry latch is always on for units position of complement add		1
B-field at the end of the add operation. The B-field answer is correct.		−$\overset{\vee}{0}$06

Figures 6 and 7 show diagrammed explanations of CPU operation in the execution of the add and subtract instructions.

The following controls are active when the CPU performs the add and subtract operations:

1. Initiate A-cycle and read out first A-field character.

Signal	Control	Logic
Set A Cy Ctrl	A Cycle First Op Codes Last Insn RO Cycle	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read-out AAR on A Cy Ops.	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16

2. Set modifier controls to −1.

Set 1st Scan Ctrl	1st Scan First Op Code Last Insn RO Cy	12.30.05
1st Scan Ctrl Lat	Set 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

3. Set character into A-data register.

Sw B Ch to A Reg	A Cy, LGD	15.38.01

4. Control A-cycle length.

Std A Cy Ops A Cy	Add Type Op Codes, A Cy	13.14.06
Stop at F	Std A Cy Ops A Cy	12.12.30

5. Initiate B-cycle and read out first B-field character.

Set B Cy Ctrl	Std A Cy Ops A Cycle	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Units Ctrl Latch	Last Insn RO Cy, Next to Last LG	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02
Regen Units + Body Ctrl	Std A Cy Ops A Cy	16.30.01
Units Ctrl Latch	Regen Units + Body Ctrl, Next to LLG Units Latch	16.30.02

Signal	Control	Logic
RO D AR * Arith	Units Latch, B Cy Ctrl Arith Type Op Codes	16.41.01
Set Mem AR Gated	LGA, 2nd CP	14.17.16

6. Regenerate modify controls.

Regen 1st Scan Ctrl	Std A Cy Ops, A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

7. Gate A-field character to A-channel.

Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02

8. Set true or complement controls according to type op code, A- and B-field signs.

True Add B	1st Scan, Add or Subt Op Codes	16.20.10
Start Compl Add 1	1st Scan, Units Latch, B Cy Add Op Code, A Ch −, B Ch +	16.20.12
Compl Latch	Start Compl Add 1	16.20.15

This step is the only difference between an add and subtract op code.

9. On complement add set carry latch to correct units position.

Carry Latch	Start Compl Add 1	16.20.21

No carry latch is set if a Start True Add.

10. Gate A-Channel to Adder.

Adder A Ch Use T or C	Units Latch, 1st Scan, B Cy Add Type Op Codes, Not 1401	16.20.11

11. Gate adder output through the assembly to storage.

Use Adder Nu	B Cy, Add or Subt	16.40.02
Use B Ch Zones	Units, 1st Scan, B Cy, Add or Subt Op Codes	16.40.01
Use B Ch WM	B Cy, Arith Type Op Codes	15.49.04
Load Memory	Load Mem on B Cy Op Codes B Cy	12.50.01

12. If there is no A- or B-channel word marks, set the carry latch if there is a carry.

RA + RS + A + S.B. Not BW	Add Type Op Codes, B Cycle B Ch Not WM Bit	16.20.03
Set Carry Latch	RA + RS + S.B. Not BW Adder Carry	16.20.20
Carry Ctrl Latch	Set Carry Latch, Last LG	16.20.21
Carry Latch	Carry Ctrl Latch, LGC	16.20.21

13. Take another A-cycle and read out next A-field character.

RA + RS + A + S.I.B Not BW Not AW	Add Type Op Codes, 1st Scan B Cy, A Ch Not WM, B Ch Not WM	16.20.03
Set A Cy Ctrl * Arith	RA + RS + A + S.I.B. Not BW Not AW	16.42.01
Set A Cy Ctrl	Set A Cy Ctrl * Arith	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read out A AR on A Cy Ops	14.71.30
Set Mem AR Gated	LG, 2nd CP	

14. Regenerate modify controls.

Regen 1st Scan Ctrl* Arith	RA + RS + A + S.B. Not BW	16.43.01

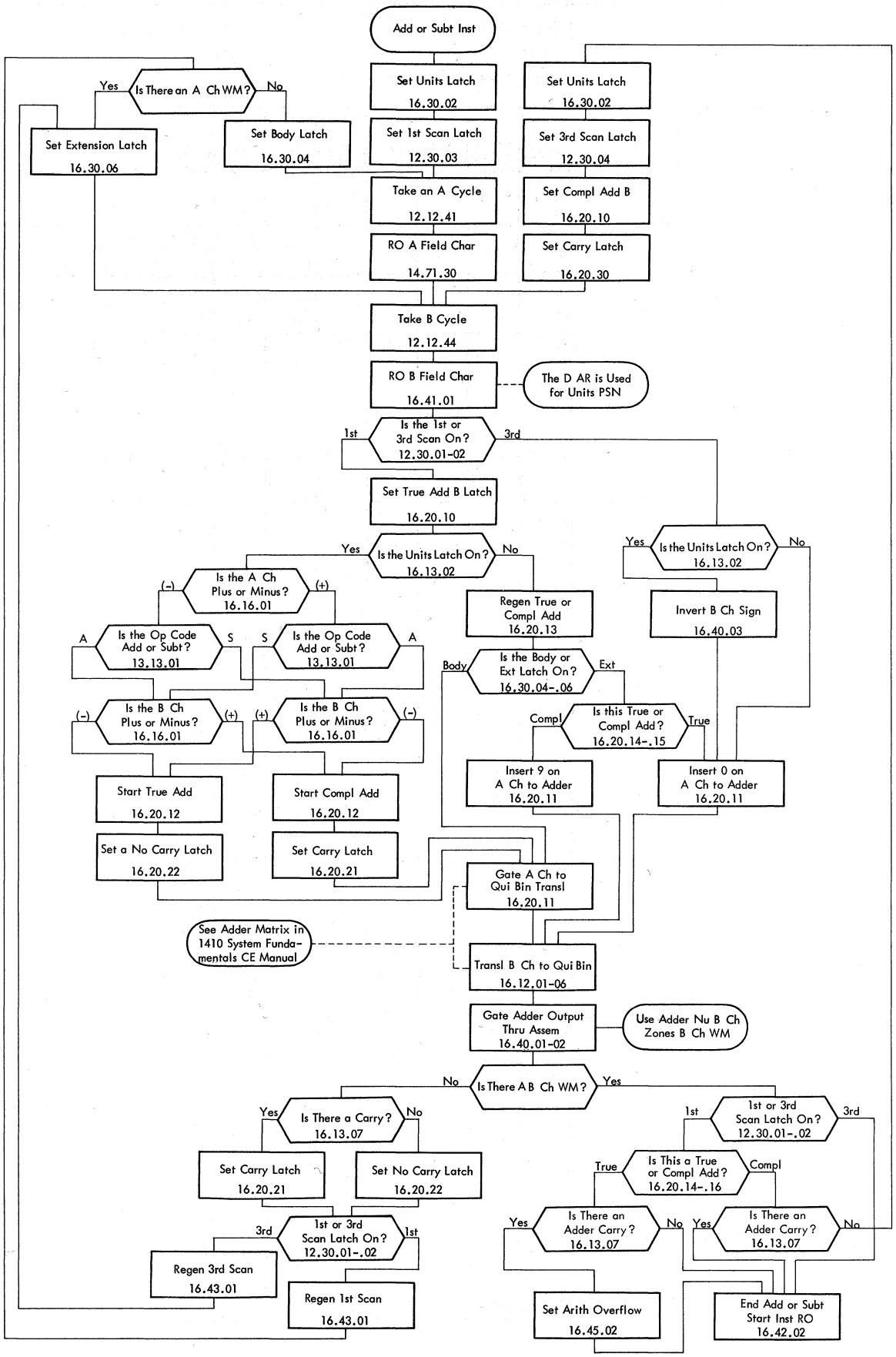


Figure 6. Add or Subtract

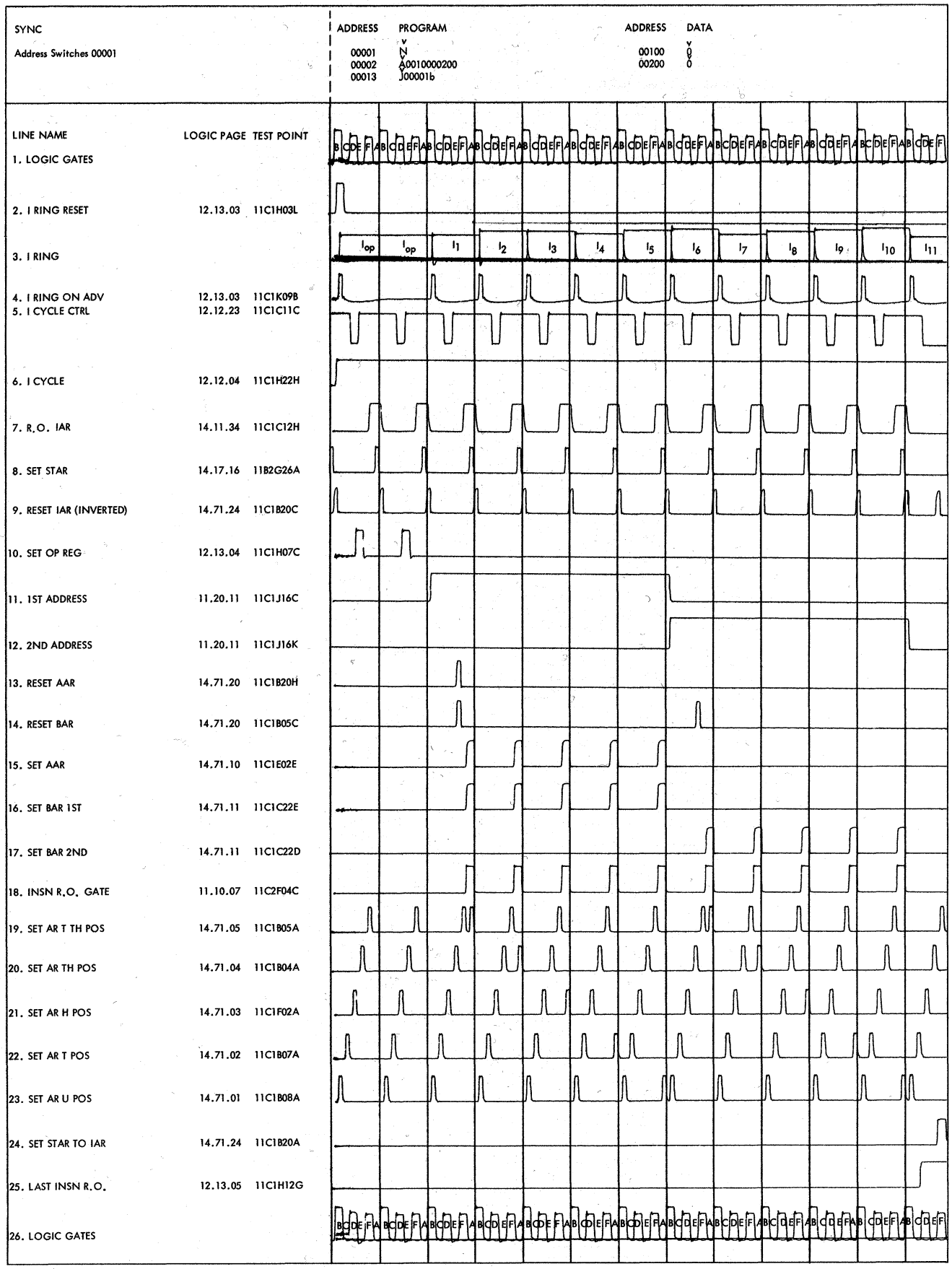


Figure 7A. Add Operation Instruction Read-out

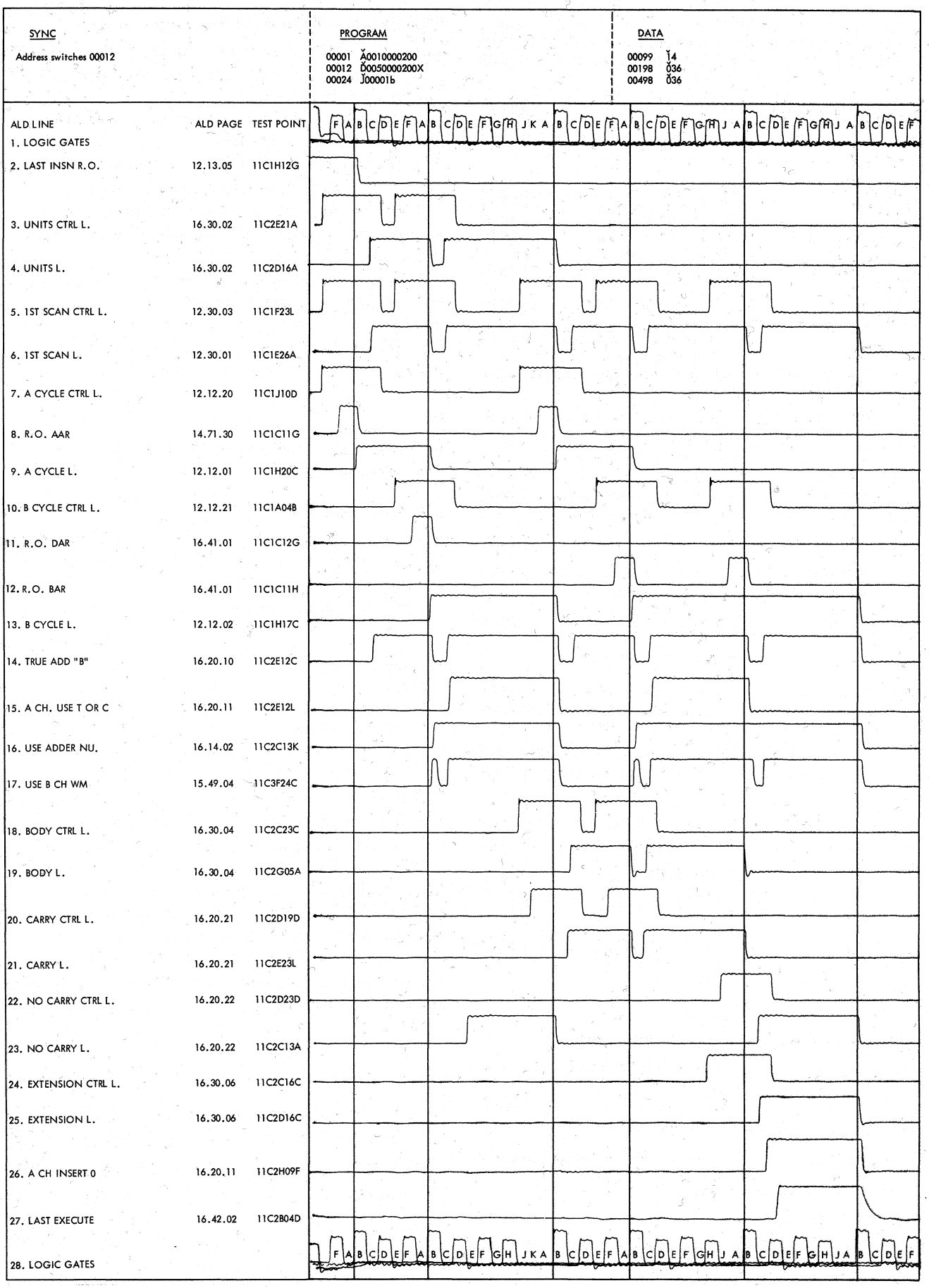


Figure 7B. Execute Phase of Add Operation

Signal	Control	Logic
Regen 1st Scan Ctrl	Regen 1st Scan Ctrl * Arith	12.30.05
1st Scan Ctrl Latch	Regen 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl Lat, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

15. Take another B-cycle and read out next B-field character.

Set B Cy Ctrl	Std A Cy Ops A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Set Body Ctrl Latch	RA + RS +A + S.I.B. Not BW Not AW	16.30.01
Body Ctrl Latch	Set Body Ctrl Latch, Next to Last LG	16.30.04
Body Latch	Body Ctrl Latch, LGC	16.30.04
RO B AR * Arith	B Cy Ctrl, Body Ctrl Latch	16.30.04
	B Cy Ctrl, Body Ctrl Latch Arith Type Op Codes	16.41.01

If there was an A-channel word mark, the extension latch is set on instead of the body latch.

16. During the extension, the adder receives a 0 or 9.

A + S.B.I.S.X	A + S.B. Cy, 1st Scan, Compl Latch, Extension Latch	16.20.05
A Ch Insert +9	A + S.B.I.S.X	16.20.11
RA + RS + A + S.I.B.T.X	True Latch, Extension Latch, Add Type Op Code, 1st Scan, B Cy	16.20.03
A Ch Insert +0	RA + RS +A + S.I.B.T.X	16.20.11

17. A B-channel word mark stops the operation except on complement add with no carry.

A + S.B.I.S.BW. Not RC	Adder No Carry, B Ch WM Bit	16.20.05
	A + S.B. Cy, 1st Scan, Compl Latch	
Set B Cy Ctrl * Arith	A + S.B.I.S. BW. Not RC	16.42.01

Set B Cycle Control * Arithmetic causes another B-cycle during which the B-field is converted to true form.

18. Set up the controls to complement add the B-field to zeros inserted into the adder on the A-field side.

Set Units Ctrl Latch	A + S.B.I.S. BW. Not RC	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch, Next to Last LG	16.30.02
Units Latch	Units Ctrl Latch, LG.C	16.30.02
Set 3rd Scan Ctrl	A + S.B.I.S. BW. Not RC	16.43.01
3rd Scan Ctrl	Set 3rd Scan Ctrl, Next to Last LG	12.30.04
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Compl Add B	3rd Scan, Add Type Op Codes	16.20.10
A Ch Insert +0	3rd Scan, B Cy, Add Type Op Codes	16.20.11

19. Set carry latch to correct units position.

Set Carry Latch	A + S.B.I.S. BW. Not RC	16.20.20
Carry Ctrl Latch	Set Carry Latch, Last LG	16.20.21
Carry Latch	Carry Ctrl Latch, LGC	16.20.21

20. Change sign over units position.

Use Inv B Ch Sign	Units Latch, B Cy, 3rd Scan Add or Subt Op Codes	16.40.03

Use Inv B-channel Sign changes + to − or − to +.

21. End the operation.

Last Execute Cy * Arith	B Cy, 3rd Scan, Add or Subt B Ch WM	16.42.02

Questions on Add and Subtract Operation

Answers to these review questions are in the Appendix.

1. *When the CPU decodes the Ǎ (AAAAA) instruction, what action occurs?*
2. *Are other A-field characters processed after a B-channel word mark is sensed?*
3. *Why is the first B-cycle in the add or subtract operation longer than subsequent B-cycles?*
4. *When the sum in an add operation (or the remainder in a subtract operation) is 0, what indicator turns on?*
5. *When the A-field word mark is sensed before the B-field word mark is detected in a true add scan, what action occurs?*
6. *When the add or subtract instruction specifies A- and B-addresses, where is the result stored?*

Zero and Add and Zero and Subtract Instructions

Instruction Formats

Formats for the zero and add (systems diagrams refer to the instruction as reset add) and zero and subtract (systems diagrams refer to the instruction as reset subtract) are as follows:

Op code	A-address	B-address
? (zero and add)	xxxxx	xxxxx
? (zero and add)	xxxxx	
? (zero and add)		
! (zero and subtract)	xxxxx	xxxxx
! (zero and subtract)	xxxxx	
! (zero and subtract)		

If the zero and add or zero and subtract instruction specifies two addresses (? or ! xxxxx xxxxx), the numeric data in the A-field is stored in the B-field. The sign of the result field (B-field) is the same as the sign of the A-field in the zero and add operation; the sign of the result field is the opposite of the A-field sign in the zero and subtract operation (Figure 8). If, in a zero and add operation, the A-field has no sign and is thus understood to be positive, the system generates an actual positive sign for the B-field by placing A-

A-Field Sign	B-Field Sign At End of Operation
No A-bit and No B-bit (plus)	B-bit (minus)
B-bit (minus)	A-bit and B-bit (plus)
A-bit and B-bit (plus)	B-bit (minus)
A-bit (plus)	B-bit (minus)

Figure 8. Sign Changes for Zero and Subtract (Two Addresses)

and B-bits over the units position. All other zone positions, except the sign position, in the B-field are cleared (no A- nor B-bits) in zero and add and zero and subtract operations. The B-field must have a defining word mark to stop the operation. The A-field requires a word mark only if it is shorter than the B-field, in which case extra high-order B-field positions are set to 0. If the A-field is longer than the B-field, the high-order positions of the A-field that exceed the limits imposed by the B-field word mark are not processed.

If the zero and add or zero and subtract instruction specifies only one address (? or ! xxxxx), numeric data in the A-field does not change. The instruction causes the system to strip the A-field of all zones, except the units (sign) position, and to change non-numeric codes (blanks and 8-3, 8-4, 8-5, 8-6, and 8-7) to their numeric equivalents (0, 3, 4, 5, 6, and 7, respectively). In the zero and add operation, the sign of the A-field is retained, but the bit configuration of the plus sign may change; for example, if the bit configuration of the plus sign is an A-bit or NOT A- and NOT B-bits, the plus sign is changed to the A- and B-bit configuration. In the zero and subtract operation, the A-field sign changes; if the A-field was positive before the operation, it is negative after the operation; if the A-field was negative before the operation, it is positive (A- and B-bits) after the operation. In the one address zero and add or zero and subtract instruction, the A-field requires a word mark in its high-order position.

If the zero and add or zero and subtract instruction does not designate an A- or B-address (no address chaining), the contents of the AAR from the previous operation specify the A-field, and the contents of the BAR specify the B-field in the operation. The zero and add or zero and subtract instruction is then executed in the manner described for two address instructions.

CPU Operation

During the A-cycle that begins execution of the zero and add or zero and subtract instruction, the units position of the A-field reads out of storage and is set in the A-data register. After the A-cycle, the system executes a B-cycle to process the B-field character from storage onto the B-channel and to gate the character in the A-data register to the A-channel. The A-channel character is set in the adder; a 0 is inserted on the B-channel input to the adder. In all cases, the adder true adds the A-channel character to 0. Because the units latch is on (first A- and B-field characters to be processed), the adder sum is combined in the assembly with the sign of the A-field (zero and add) or with the inverted sign of the A-field (zero and subtract). The character and the sign are gated to the units position of the B-field in storage. Until an A- or B-channel word mark is sensed, the system alternately executes A- and B-cycles, adds A-channel numeric bits to 0 in the adder, and stores the adder output in the B-field. If an A-channel word mark is detected before a B-channel word mark is sensed, zeros are stored in the remaining B-field positions. A B-channel word mark terminates the operation.

Figures 9 and 10 show diagrammed explanations of CPU operation in the execution of the zero and add and zero and subtract instructions.

The following controls are active when the CPU performs the zero and add and zero and subtract operations.

SIGNAL	CONTROL	LOGIC
1. Initiate A-cycle and read out first A-field character.		
Set A Cy Ctrl	A Cy First Op Codes Last Insn RO Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read out AAR on A Cy Ops.	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
2. Set modifier controls to −1.		
Set 1st Scan Ctrl	1st Scan First Op Code Last Insn RO Cy	12.30.05
1st Scan Ctrl Lat	Set 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
3. Set Character into A-data register.		
Sw B Ch to A Reg	A Cycle, LGD	15.38.01
4. Control A-cycle length.		
Std A Cy Ops A Cy	Add Type Op Codes, A Cy	13.14.06
Stop at F	Std A Cy Ops A Cy	12.12.30
5. Set true add controls.		
Set True	RA + RS • Last Insn RO Cy	16.20.13
True Ctrl Latch	Set True, Last LG	16.20.14
True Latch	True Ctrl Latch, LGC	16.20.14
Set No Carry	RA + RS • Last Insn RO Cy	16.20.20
No Carry Ctrl Latch	Set No Carry, Last LG	16.20.20
No Carry Latch	No Carry Ctrl Latch, LGC	16.20.22
6. Initiate B-cycle and read out first B-field character.		
Set B Cy Ctrl	Std A Cy Ops A Cycle	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Units Ctrl Latch	Last Insn RO Cy, Next to Last LG	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02
Regen Units + Body Ctrl	Std A Cy Ops A Cy	16.30.01
Units Ctrl Latch	Regen Units + Body Ctrl, Next to Last LG Units Latch	16.30.02
RO D AR * Arith	Units Latch, B Cy Ctrl Arith Type Op Codes	16.41.01
Set Mem AR Gated	LGA, 2nd CP	14.17.16
7. Regenerate modify controls.		
Regen 1st Scan Ctrl	Std A Cy Ops, A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

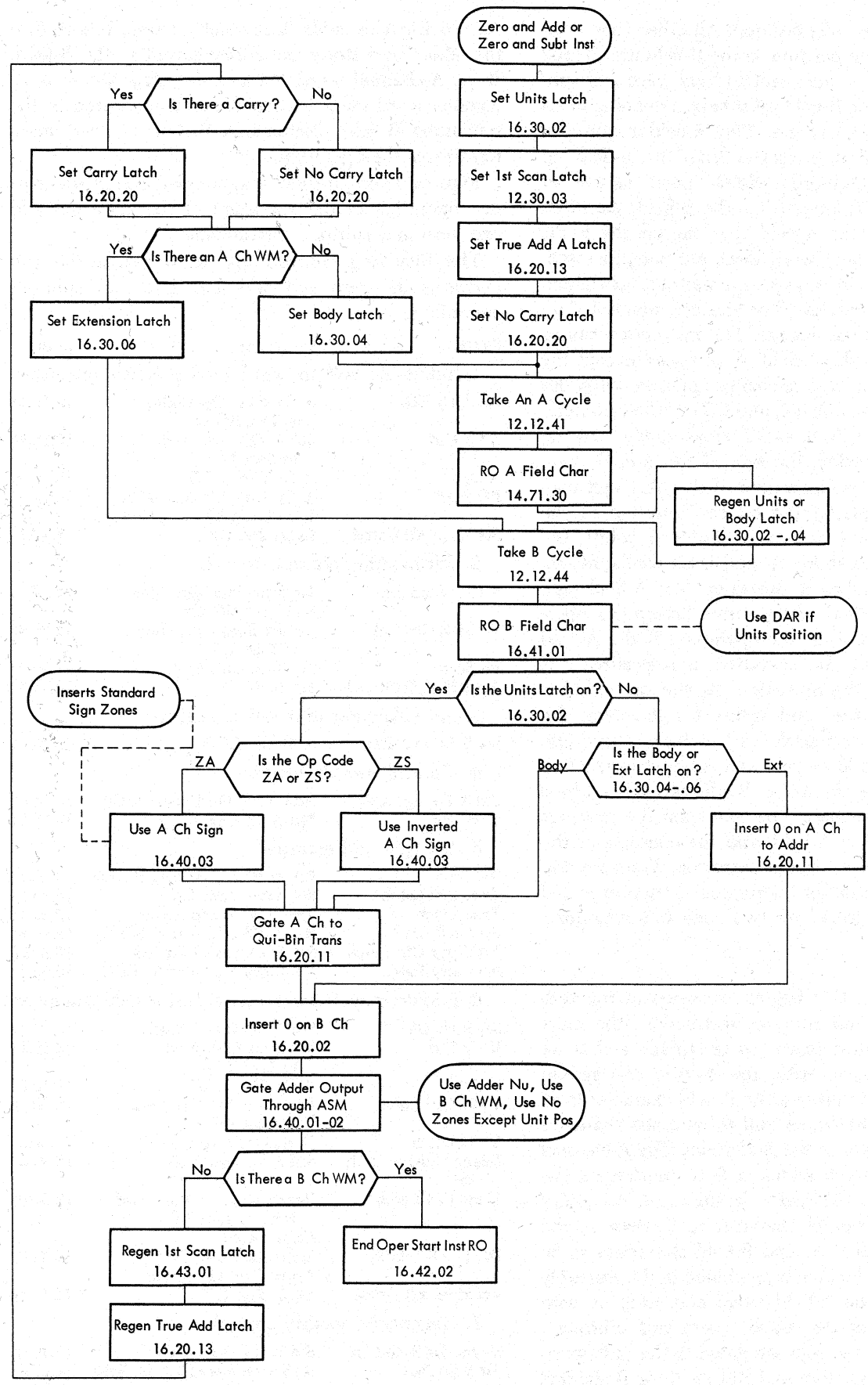


Figure 9. Zero and Add or Zero and Subtract

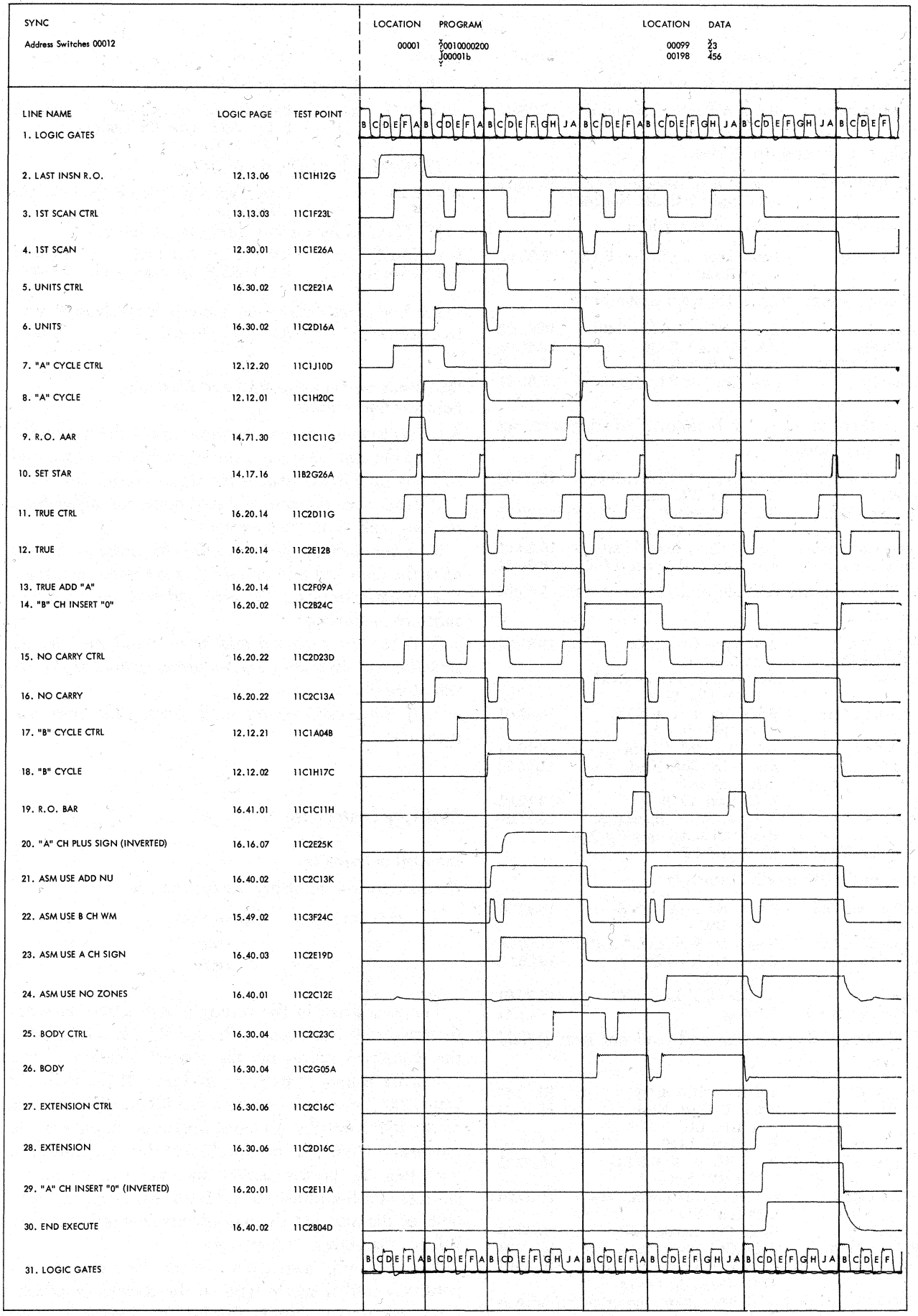


Figure 10. Zero and Add Operation Timings

Signal	Control	Logic
8. Gate A-field character to A-channel.		
Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02
9. Gate A-channel to Adder.		
Adder A Ch Use T or C	Units Latch, 1st Scan, B Cy Add Type Op Codes, Not 1401	16.20.11
10. Insert a 0 in B side of the adder.		
B Ch Insert +0	Reset Type Op Codes, B Cy, 1st Scan	16.20.02
11. Gate adder output through assembly.		
Use Adder Nu	RA + RS • B • Not 1401	16.40.02
Use A Ch Sign	RA • Units • B Cy	16.40.03
or Use Inv A Ch Sign	RS • Units • B Cy	16.40.03
Load Storage	Load Mem on B Cy Op Codes B Cy	12.50.01
12. If there is no A- or B-channel word marks, set the no carry latch.		
RA + RS + A + S.B. Not BW	Add Type Op Codes, B Cy B Ch Not WM Bit	16.20.03
Set No Carry Latch	RA + RS + S.B. Not BW, No Adder Carry	16.20.20
No Carry Ctrl Latch	Set No Carry Latch, Last LG	16.20.21
No Carry Latch	No Carry Ctrl Latch, LGC	16.20.21
13. Take another A-cycle and read out next A-field character.		
RA + RS + A + S.I.B. Not BW Not AW	Add Type Op Codes, 1st Scan B Cycle, A Ch Not WM, B Ch Not WM	16.20.03
Set A Cy Ctrl * Arith	RA + RS + A + S.I.B. Not BW Not AW	16.42.01
Set A Cy Ctrl	Set A Cy Ctrl * Arith	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read Out A AR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	
14. Regenerate modify controls.		
Regen 1st Scan Ctrl * Arith	RA + RS + A + S.B. Not BW	16.43.01
Regen 1st Scan Ctrl	Regen 1st Scan Ctrl * Arith	12.30.05
1st Scan Ctrl Lat	Regen 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl Lat, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
15. Take another B-cycle and read out next B-field character.		
Set B Cy Ctrl	Std A Cy Ops A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Set Body Ctrl Latch	RA + RS + A + S.I.B. Not BW Not AW	16.30.01
Body Ctrl Latch	Set Body Ctrl Latch, Next to Last LG	16.30.04
Body Latch	Body Ctrl Latch, LGC	16.30.04
RO B AR * Arith	B Cy Ctrl, Body Ctrl Latch	16.30.04
	B Cy Ctrl, Body Ctrl Latch Arith Type Op Codes	16.41.01

If there was an A-channel word mark, the extension latch is set on instead of the body latch.

Signal	Control	Logic
16. Gate A-channel to adder.		
Adder A Ch Use L or C	Body Latch, 1st Scan B Cy, Add Type Op Codes Not 1401	16.20.11
17. Gate a 0 in B side of adder.		
B Ch Insert +0	Reset Type Op Codes, B Cy 1st Scan	16.20.02
18. Gate adder output through assembly.		
Use Adder Nu	RA + RS.B. Not 1401	16.40.02
Use No Zones	RA + RS.B. 1st Scan. Body Latch	16.40.01
19. End operation when there is a B-Channel WM.		
Last Execute Cy	RA + RS.I.B. B Ch WM	16.42.02

Questions on Zero and Add and Zero and Subtract Operations

Answers to these review questions are in the Appendix.

1. If the sign of the A-field is positive in a zero and subtract operation, what is the sign of result field?

2. How does the zero and add operation differ from the zero and subtract operation?

3. What positions of the result field contain zone bits after the zero and add or zero and subtract operation?

4. What terminates the zero and add and zero and subtract operations?

5. When the zero and add or zero and subtract instruction specifies only an A-address, where is the result stored?

6. If the A-field contains all zeros, what indicator turns on?

Multiply Instruction

Instruction Formats

Formats for the multiply instruction are:

Op code	A-address	B-address
@	xxxxx	xxxxx
@	xxxxx	
@		

The A-address in the multiply instruction specifies the low-order character in the A-field (multiplicand); the B-address represents the storage location of the low-order B-field character (product). If the multiply instruction does not contain a B-address, the contents of the BAR from the previous operation designate the low-order character in the B-field for the multiply operation. If the instruction does not contain an A-address or a B-address (no address chaining), the contents of the AAR and the BAR specify low-order A- and B-field characters, respectively.

The multiply instruction causes the CPU to repetitively add numeric data in the A-field (multiplicand) to numeric data in the B-field (product), start-

ing with low-order B-field positions. Results of the additions are stored in the B-field.

Multiplier digits must be stored in the high-order B-field positions before the multiply operation begins. However, the mutilplier is eliminated digit-by-digit as the operation progresses. If the multiplier is to be retained, it must be stored in another area in memory.

Word marks must be set in high-order multiplier and multiplicand positions.

B-Field Length

Because the product is developed in the B-field, the field must be long enough to accommodate repetitive additions of the A-field and still not interfere with useful multiplier positions. Therefore the number of digits in the multiplier plus the number of digits in the multiplicand plus one equals the number of positions that the B-field must contain.

Concept of Machine Multiplication

The following example is used to aid in explaining the concept of multiplication on the IBM 1410 Data Processing System:

Multiplier X̆XX̂
Multiplicand YYYYY̅ A-address B-address
Product Field (before multiplier is moved in) ZZZZZZZZZ

Because the multiplier has three positions and the multiplicand has five positions, the product field (B-field) has nine positions. Before the CPU decodes the multiply instruction, an image of the multiplier must be stored in the high-order B-field positions; this is usually accomplished with the zero and add instruction. The product field (B-field) must contain x̆xx̂zzzzzz when the multiply operation begins.

Before the CPU performs the first add scan in the operation:

1. Characters to the right of multiplier digits in the product field are replaced with zeros, and

2. Zone bits in the units multiplicand (A-field) and multiplier digits are analyzed to determine the signs of the respective factors. If signs of both the A-field and the multiplier are alike, a plus sign is set over the units position of the B-field. If signs of the A-field and the mutliplier are different, a minus sign is set over the units position of the B-field.

The B-field in the example becomes x̆xx000000̄.

In other CPU operations in the execution of the multiply instructions:

1. Data in the A-field are true added or complement added into the B-field. Each multiplier digit specifies the number and type of add scans that should be executed (the number of times that the A-field is added into the B-field).

2. The CPU shifts when action that each multiplier digit designates is complete. The shift allows the add scans that the next multiplier digit requires to begin at the next product field position in high-order sequence.

3. Multiplier digits are incremented or decremented by one before or after add scans until all multiplier digits are either reduced to 0 or increased to 9. The multiply operation ends when a 0 with a word mark is sensed in the multiplier.

Rules of Machine Multiplication

To execute a multiply instruction, the CPU applies the following rules:

1. A multiply operation begins with the true add latch set and ends with a true add scan.

2. A multiplier digit of 1, 2, 3, or 4 causes:
   a. The multiplier digit to be decreased by one.
   b. The CPU to perform a true add scan.
   c. The CPU to shift after the multiplier digit is reduced to 0.

3. A multiplier digit of 5, 6, 7, 8, or 9 causes:
   a. The multiplier digit to be increased by one.
   b. The CPU to perform a complement add scan (except as noted in rule 6).
   c. The CPU to shift after the multiplier digit is increased to 9.

4. When a shift and a change from complement add to true add scan occur, the multiplier digit is not decreased before the true add scan.

5. When multiplier digits designate that a true add scan follow a complement add scan (as in the cases of multiplier digits 18, 47, 36, 25), an extra true add scan is taken after the shift ending the last complement add scan.

6. When the multiplier digit 9 follows the multiplier digit 5, 6, 7, 8, or 9 (for example 97, 95, 98, 99), the CPU does not perform a complement add scan when the high-order 9 is sensed; a shift is signalled immediately.

7. When performing complement add scans, the complement of the A-field (multiplicand) is added to the B-field the number of times equal to the complement of the mutliplier digit, except as noted in rule 6; for example, the multiplier digit 8 specifies that the A-field be complement added to the B-field (rule 3b) twice; the tens complement of 8 is 2. However, the multiplier digit is only increased to 9 (rule 3c).

8. The zero balance indicator turns on when the product field is 0 at the end of a multiply operation.

9. A multiplier digit of 0 signals a shift immediately; no add scan is taken.

Machine Multiplication Examples

The following examples illustrate the manner in which the CPU executes specific multiply operations:

EXAMPLE 1

A multiplier units digit of 1, 2, 3, or 4 causes a corresponding number of true additions of the A-field (multiplicand) to low-order positions of the product field (B-field). Before each true add scan, the multiplier digit is decreased by one until the digit is reduced to 0. When the multiplier digit is 0 and the subsequent true add scan is complete, a left shift is forced, causing:

1. The tens position of the multiplier to control the number and type of add scans.

2. The add scans to begin in the tens position of the product field.

This procedure is repeated for each position of the multiplier until the action that each multiplier digit specifies is complete.

Example: 203 × 1625

	Multiplier	B-field
	2 0 3 | 0 0 0 0 0	
Read out 3 and reduce	2 0 2 | 0 0 0 0 0	
True add	0 1 6 2 5	
Read out 2 and reduce	2 0 1 | 0 1 6 2 5	
True add	0 1 6 2 5	
Read out 1 and reduce	2 0 0 | 0 3 2 5 0	
True add	0 1 6 2 5	
Read out 0 and shift	2 0 0 | 0 4 8 7 5	
Read out 0 and shift	2 0 | 0 0 4 8 7 | 5	
Read out 2 and reduce	1 | 0 0 0 4 8 | 7 5	
True add	0 1 6 2 5	
Read out 1 and reduce	0 | 0 1 6 7 3 | 7 5	
True add	0 1 6 2 5	
Read out 0 end op	0 | 0 3 2 9 8 | 7 5	

EXAMPLE 2

A units multiplier digit of 5, 6, 7, 8, or 9 causes the complement of the A-field to be added to low-order positions of the product field. The complement of the multiplier digit determines the number of complement add scans required. The multiplier digit is incremented by one after each complement scan until the digit is increased to 9. When the required number of complement ad scans (as specified by the units multiplier digit) have been taken, a left shift is forced, causing:

1. The tens position of the mutliplier to control the number and type of add scans.

2. The add scans to begin in the tens position of the product field. This procedure is repeated for each position of the multiplier until the action that each multiplier digit specifies is complete.

If the high-order multiplier digit (the digit containing the word mark) is 5, 6, 7, 8, or 9, the CPU performs a true add scan after the last complement add scan and the shift have been taken.

Example: 7 × 1625

	Multiplier
	7 0 0 0 0 0
Add complement of A-field	9 8 3 7 5
Increment by one	8 9 8 3 7 5
Add complement of A-field	9 8 3 7 5
Increment by one	9 9 6 7 5 0
Add complement of A-field	9 8 3 7 5
Shift	9 9 5 1 2 5
True add A-field	1 6 2 5
End op	0 1 1 3 7 5

Complement addition for multiplier digits 5, 6, 7, 8, and 9 saves processing cycles and time. For example, a multiplier digit of 7 causes, instead of 7 true add scans, only four scans total.

EXAMPLE 3

When a multiplier digit of 9 is sensed after a multiplier digit of 5, 6, 7, 8, or 9 (95, 96, 97, 98, 99), the high-order 9 in the multiplier does not require a complement scan. The high-order 9 signals a left shift immediately.

When multiplier digits designate that a true add scan follow a complement add scan, an extra true add scan is taken after the shift ending the last complement add scan. The multiplier digit is not decreased before the true add scan.

A multiplier digit of 0 signals a left shift immediately.

Example: 19910 × 22

	Multiplier
	1 9 9 1 0 0 0
Read out 0 and shift	1 9 9 1 0 0 0
Read out 1 and decrease	1 9 9 0 0 0 0
True add	2 2
Shift	1 9 9 0 0 2 2 0
Read out 9 and comp add	1 9 9 0 0 2 2 0
	9 7 8
Read out 9 and shift (9 follows 9)	1 9 9 9 8 0 2 0
Read out 1 and true add	1 9 9 9 8 0 2 0
(switch from comp add to true add)	2 2
Read out 1 and decrease	0 0 2 1 8 0 2 0
True add	2 2
End op	0 0 4 3 8 0 2 0

Address Registers

In a multiply operation:

a. The AAR scans the A-field.

b. The BAR scans the product field (B-field).

c. The CAR retains the address of the units position of the multiplicand (A-field).

d. The DAR retains the address of the product field position used at the start of each scan. The DAR is modified by −1 on a shift cycle.

At the end of the multiply operation, the AAR contains the address equal to the original A-address minus

the length of the A-field; the BAR contains the address equal to the original B-address minus the length of the B-field.

CPU Operation

To more clearly explain CPU action when performing a multiply operation, execution of the following example is used to supplement descriptions of CPU functions. Figure 11 shows a step-by-step breakdown in the execution of the following example:

Multiplier 82
Multiplicand (A-field) 181
Product Field (B-field when the multiply operation starts) 82XXXX

The CPU must first locate the units position of the multiplier. To accomplish this, the CPU scans through the A- and B-fields and replaces B-field characters with zeros until an A-field word mark is sensed. In the example, three A- and B-cycles are required to read out the A-field word mark; the B-field changes to 82X000.

The A-field word mark sets the extension latch, and the CPU takes another A- and B-cycle. During the A-cycle, the CAR is used to read out the units position of the A-field; the units position A-field character (containing the sign of the A-field) is stored in the A-data register. During the B-cycle, the extra position in the product field (the position that contains neither a 0 nor part of the multiplier) is replaced with a 0. In the example, the B-field changes to 820000.

Another B-cycle follows. The units position of the multiplier (containing the sign of the B-field) is read out of storage and gated onto the B-channel. The A-field character in the A-data register is gated to the A-channel. The MQ latch is set to identify characters on the A- and B-channels as units position characters of the multiplier and the multiplicand. Signs over the A- and B-channel characters are analyzed to determine the sign of the product field. Like signs (as in the example) set the plus latch; unlike signs set the minus latch.

The B-channel character (units multiplier digit) is examined to determine whether true or complement add cycles are to be taken. If the multiplier digit is 1, 2, 3, or 4, the true add latch is set, and the digit is decreased by one and returned to the units multiplier position in storage. To accomplish this, the digit 9 is inserted in the A-channel side of the adder, and the B-channel character (1, 2, 3, or 4) is inserted in the B-channel side of the adder. The two digits are added, and the high-order carry is dropped, effectively subtracting one from the B-channel digit. For example, when the digit 2 is inserted in the B-channel side of the adder, the digit 9 is inserted on the A-channel input to the adder. The digits are added, and the high-order carry is dropped; the adder ouput (1) is stored in the units position of the multiplier in memory. The B-field in the example becomes 810000.

If the multiplier digit is 5, 6, 7, 8, or 9, the complement add latch is set, and the multiplier digit is returned to storage unchanged.

In addition to setting the true add or complement add latch, the units position multiplier digit sets the first, second, or no scan latch for the subsequent D-cycle. If the digit is not a 0, the no scan latch is set; if the digit is a 0 without a word mark, the first scan latch is set; if the digit is a 0 with a word mark, the second scan latch is set, and the operation ends when the subsequent D-cycle is complete. In the example, the units digit in the multiplier is 2, causing the no scan latch to turn on.

When the sign analysis is complete and the no scan and true add or complement add latches are set, the CPU takes a D-cycle to set the sign of the product over the units position of the product field (B-field). The product field in the example then becomes 810000.

The CPU scans the A- and B-fields and true adds or complement adds the A-field to the B-field. The number and type of add scans performed are determined by the numeric value of the multiplier digit. In the example, the units position multiplier digit was 2 before the reduction, indicating that the A-field must be true added to the B-field twice. The CPU takes A- and B-cycles to read out multiplicand and product field characters to execute the addition. Before each A-field to B-field true addition is begun, the multiplier digit is read out and reduced by one. At the end of the second true add scan, the B-field in the example becomes 800362. When the multiplier digit is 5, 6, 7, 8, or 9, the complement of the A-field is added to the B-field the number of times that the complement of the multiplier digit specifies; for example, the multiplier digit 8 indicates that the A-field must be complement added to the B-field twice (the tens complement of 8 is 2). The digit is increased by one after the first complement add scan; in cases of other digits designating complement add scans, the digit is increased by one after each A-field to B-field addition until the multiplier digit is increased to 9.

When the designated number of true add or complement add scans are complete, the next multiplier digit is examined to determine whether true or complement add scans are required.

When action that each multiplier digit specifies is completed, a left shift occurs, causing the subsequent complement or true add cycle to begin at the next digit in high-order sequence. In the example, a left shift occurs when the two true add cycles that the units multiplier digit (2) designates are executed. Actions

MULTIPLY PROBLEM: "B" FIELD V 8 | + 2 | | | | | "A" FIELD V 1 | 8 | − 1

▲ -LOCATION OF D AR

OBJECTIVES	CYCLE	A/R RO	UNITS BODY EXTN MQ	SCAN 1ST 2ND, 3RD NO	STORED "B" FIELD ▼						DIGIT ON ASSEM	ADDER "B"	CARRY NO CARRY	ADDER "A"	TRUE OR COMP	REMARKS
LOCATE- UNITS POS. OF MULTIPLIER	A B	C AR D AR	U	1	V	+				0	0				T	UNITS A UNITS B
ZERO- PRODUCT FIELD	A B	A AR B AR	Y	1					0		0				T	
	A B	A AR B AR	Y	1				0			0				T	
	A B	C AR B AR	X	1			0				0				T	UNITS A
ANALYZE-SIGN & MULT CHAR	B	B AR	MQ	1		1					1	2	C	9	T	REDUCE
STORE-SIGN	D	D AR		N						0	0				T	UNITS B
					V 8	1	0	0	0	0						
TRANSFER- TRUE ADD SCAN	A B	C AR D AR	U	3						1	1	0	C	1	T	UNITS A UNITS B
	A B	A AR B AR	Y	3					8		8	0	C	8	T	
	A B	A AR B AR	Y	3				1			1	0	C	1	T	
	B	B AR	X	3			0				0	0	C	0	T	
	B	B AR	MQ	3		0					0	1	C	9	T	REDUCE
					V 8	0	0	1	8	1						
TRANSFER- TRUE ADD SCAN	A B	C AR D AR	U	3						2	2	1	C	1	T	UNITS A UNITS B
	A B	A AR B AR	Y	3					6		6	8	C	8	T	
	A B	A AR B AR	Y	3				3			3	1	C	1	T	
	B	B AR	X	3			0				0	0	C	0	T	
	B	B AR	MQ	3		0					0	0	C	9	T	
MODIFY D AR -1	D	D AR		3						▼	0				T	SHIFT
ANALYZE-MULT.CHAR.	B	B AR	MQ	3	8						8	8	C	9	T	
					V 8	0	0	3	6	2						
TRANSFER- COMP ADD SCAN	A B	C AR D AR	U	3					5		5	6	C	8	S	UNITS A TENS B
	A B	A AR B AR	Y	3				5			5	3	C	1	S	
	A B	A AR B AR	Y	3			8				8	0	C	8	S	
	B	B AR	X	3		9					9	0	C	9	S	
	B	B AR	MQ	3	9						9	8	C	0	S	INCREASE
					V 9	9	8	5	5	2						
TRANSFER- COMP ADD SCAN	A B	C AR D AR	U	3					4		4	5	C	8	S	UNITS A TENS B
	A B	A AR B AR	Y	3				7			7	5	C	1	S	
	A B	A AR B AR	Y	3			6				6	8	C	8	S	
	B	B AR	X	3		9					9	9	C	9	S	
	B	B AR	MQ	3	9						9	9	C	0	S	
MODIFY D AR -1	D	D AR		3				▼							T	SHIFT & MDL - "ON"
					V 9	9	6	7	4	2						
TRANSFER- TRUE ADD SCAN	A B	C AR D AR	U	3				8			8	7	C	1	T	UNITS A HUNDREDS B
	A B	A AR B AR	Y	3			4				4	6	C	8	T	
	A B	A AR B AR	Y	3		1					1	9	C	1	T	
END OPERATION	B	B AR	X	3	0						0	9	C	0	T	
					V 0	1	4	8	4	2						

Figure 11. Multiply Example

that the next multiplier digit (8) specify must begin in the tens position of the B-field.

The next multiplier digit reads out of storage and sets the true or complement add controls. The digit is decreased by one and returned to storage if the true add latch is set. The digit is returned to storage unchanged if the complement add latch is set. The A-field is then true added or complement added to the B-field as designated by the multiplier digit. The B-field in the example changes to $\overset{\vee}{9}96742\overset{+}{}$ after the shift and two complement add scans are complete.

The multiply operation cannot end with a complement add scan. In the example, the high-order multiplier digit specifies two complement add scans. A shift and a true add scan must be executed after the second complement add scan. The B-field in the example becomes $\overset{\vee}{0}14842\overset{+}{}$. The multiply operation ends when the 0 with a word mark is sensed in the multiplier.

Figures 12 and 13 shows detailed CPU action in the execution of a multiply operation.

Questions on Multiply Operation

Answers to these review questions are in the Appendix.

1. *List the multiplier digits that cause:*
   a. *Complement add scans.*
   b. *True add scans.*
2. *When the multiplier contains three digits and the multiplicand contains five digits, how many storage positions should be reserved for the product field?*
3. *When is the multiply operation terminated?*
4. *Does the CPU move the multiplier to the product field when executing a multiply instruction?*
5. *When the multiplier is* $\overset{\vee}{9}9$*:*
   a. *How many complement add scans are required in the multiply operation?*
   b. *How many true add scans are required in the multiply operation?*
6. *When the multiplier is* $\overset{\vee}{8}28$*:*
   a. *How many complement add scans are required in the multiply operation?*
   b. *How many true add scans are required in the multiply operation?*

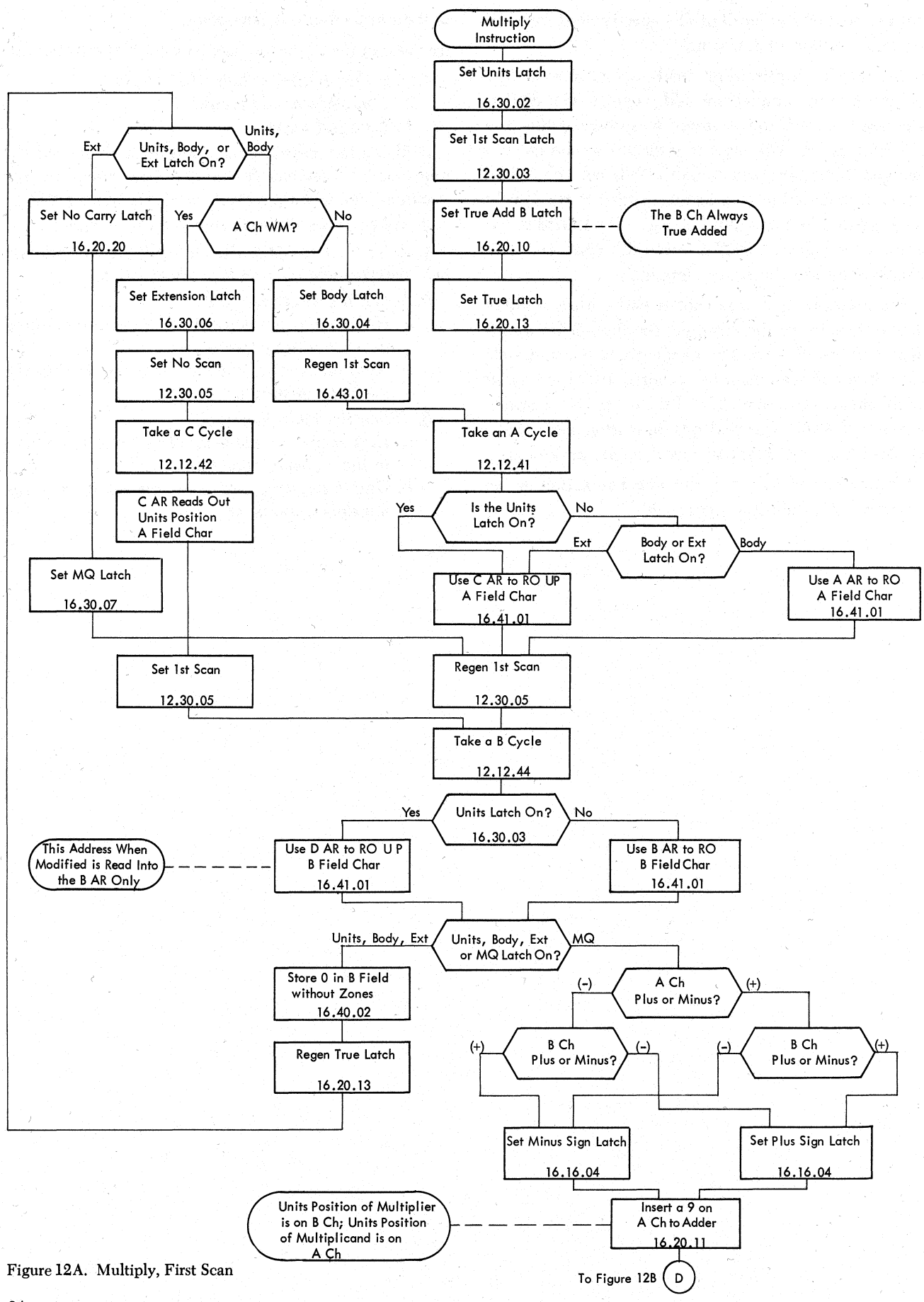


Figure 12A. Multiply, First Scan

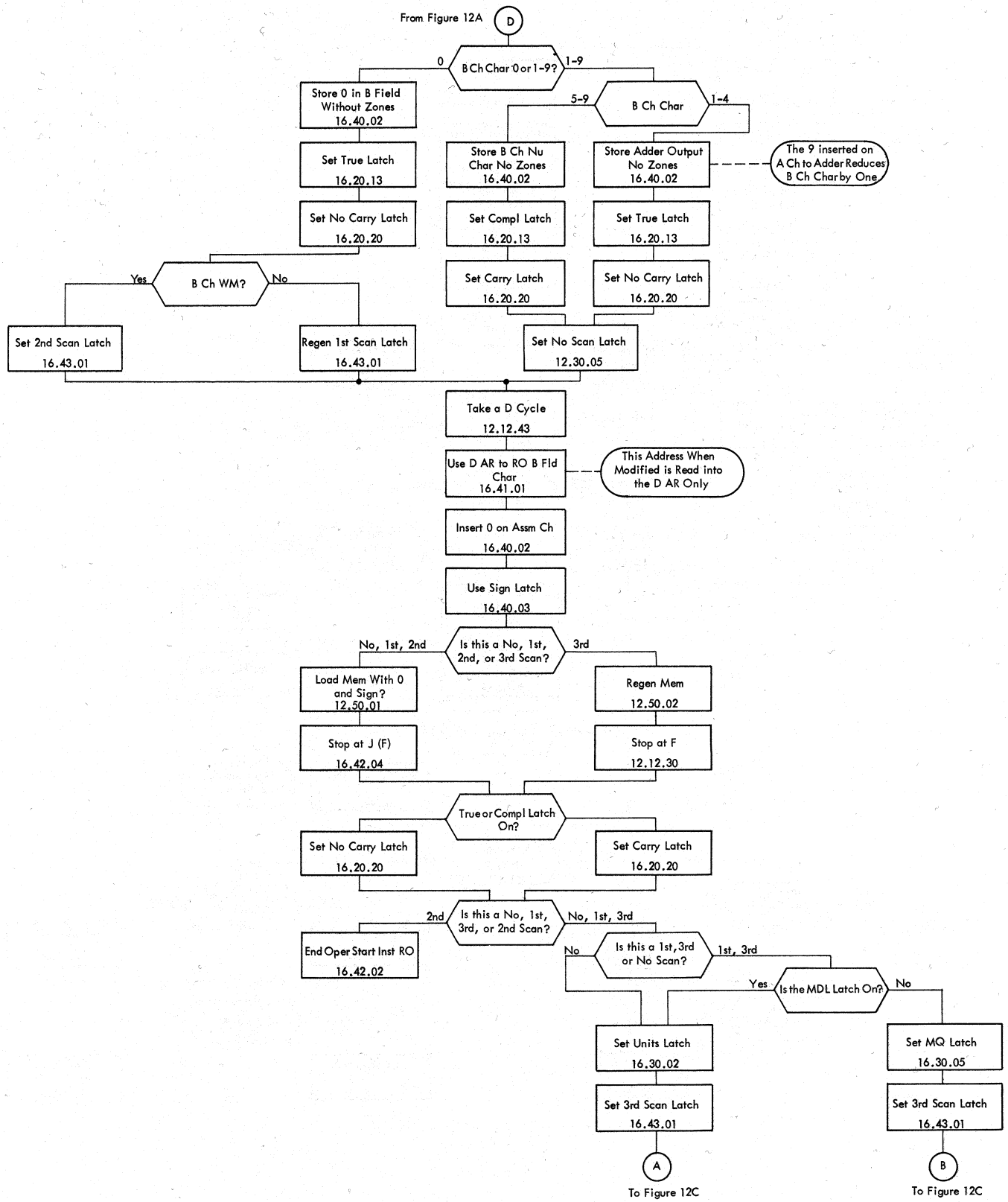


Figure 12B. Multiply, Set Sign and/or Shift

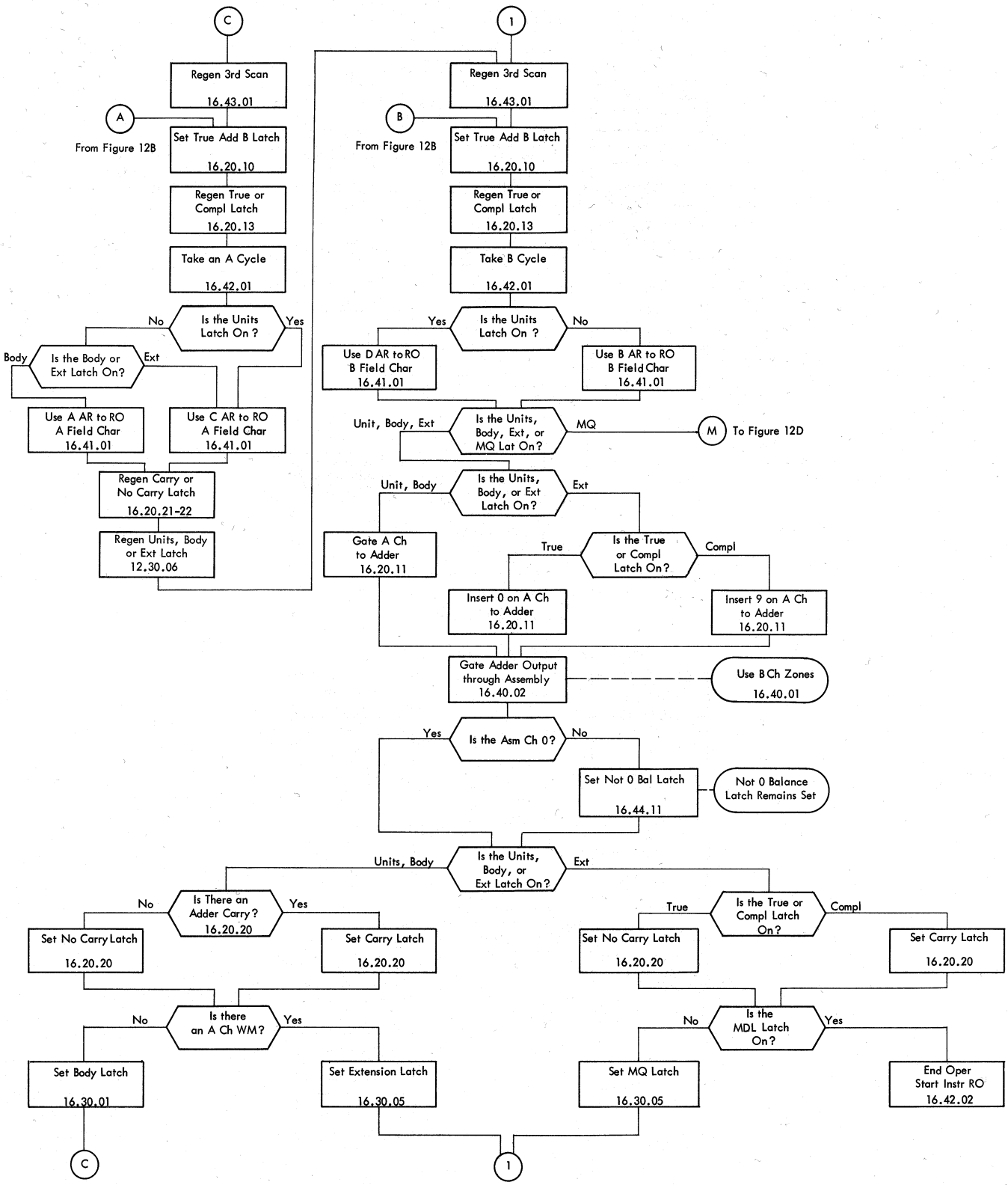


Figure 12C. Multiply, Add A-Field to B-Field

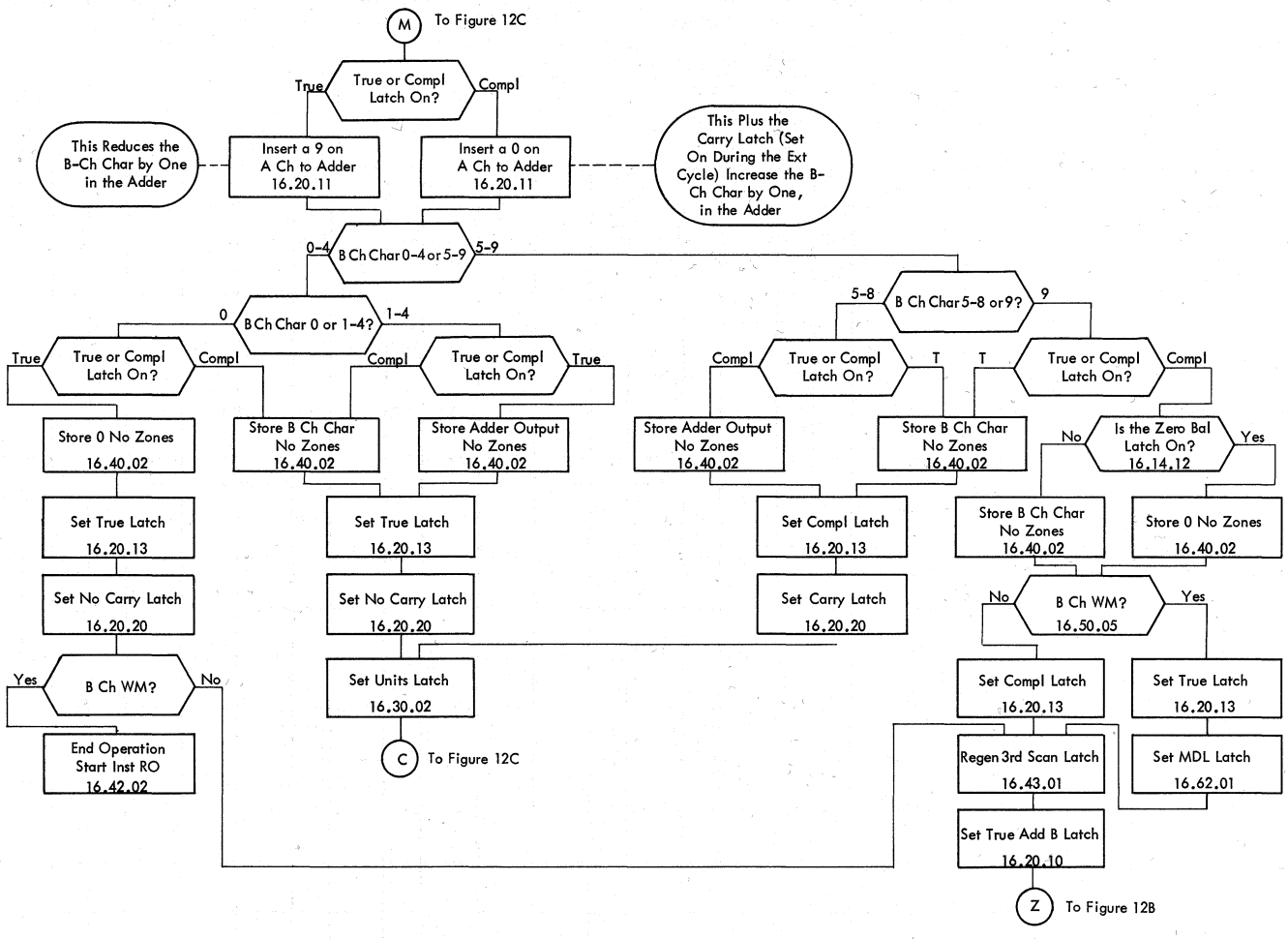


Figure 12D. Multiply, MQ Controls

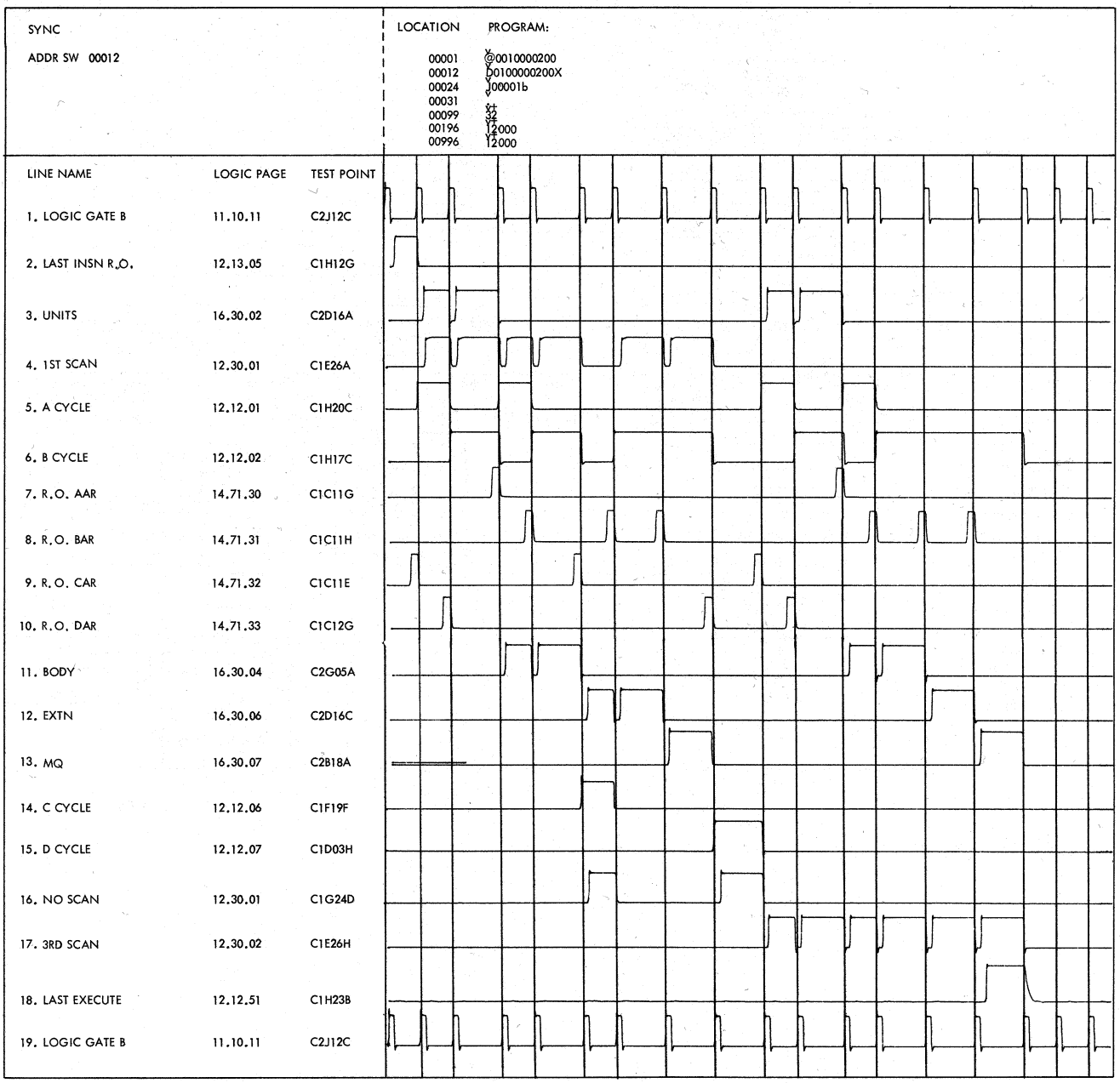
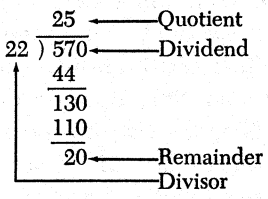


Figure 13. Multiply Operation Timings

Divide Instruction

The following example shows mathematical terms assigned to factors in a divide operation:



Instruction Formats

Formats for the divide instruction are:

OP CODE	A-ADDRESS	B-ADDRESS
%	XXXXX	XXXXX
%	XXXXX	
%		

The A-address in the instruction specifies the units position of the divisor; the B-address designates the storage location containing the high-order position of the dividend. If the divide instruction does not have a B-address, the contents of the BAR from the previous operation locate the high-order position of the dividend. If neither an A- nor B-address is included in the instruction (no address chaining), contents of the AAR and BAR from the previous operation provide A- and B-addresses, respectively, for the divide operation.

The divide instruction causes the dividend (located in the B-field) to be divided by the divisor (located in the A-field). To effect the divide operation, the CPU executes a series of complement add scans and right shifts to repetitively subtract the divisor from the dividend. The quotient is developed in high-order B-field positions; the remainder is located in low-order positions in the B-field; the dividend is destroyed as the divide operation proceeds.

Programming Considerations

The program containing the divide instruction must establish certain conditions before the CPU begins the division. Conditions that must be considered before the divide operation begins are:

Addressing of Factors: The dividend is located in the low-order positions of the B-field. During the divide operation, the quotient is developed in high-order B-field positions, and the dividend is destroyed. If the dividend is to be retained, it must also be stored in another area in memory.

Length of the B-field: Because the quotient is developed in the B-field, the field must be long enough to accommodate repetitive complement additions of the A-field and still not interfere with quotient positions being developed. Therefore, the length of the B-field is determined by adding 1 to the sum of the number of digits in the divisor and dividend; for example, the problem 5001 ÷ 10 requires that the B-field contain seven positions.

4 (positions in dividend) + 2 (positions in divisor) + 1 = 7 (positions in B-field)

Signs: The divisor can either be signed or unsigned. If no zone bits are in the units position of the divisor, the divisor is considered positive. A sign must be set in the units position of the dividend (located in the units position of the B-field) to stop the divide operation. The dividend sign must consist of A- and B-bits for plus, or B-bit for minus. At the end of the operation, the sign of the quotient follows algebraic sign rules (Figure 14), and appears over the units position of the quotient; the sign of the remainder is the sign of the original dividend.

Divisor Sign	+	+	-	-
Dividend Sign	+	-	+	-
Remainder Sign	+	-	+	-
Quotient Sign	+	-	-	+

Figure 14. Divide Sign Control

Zeros: The quotient field (the high-order B-field positions that do not contain the dividend) must contain zeros when the divide operation begins. Moving the dividend into the B-field by means of the zero and add instruction insures both the presence of zeros in high-order (quotient) positions of the B-field and proper signing of the B-field.

Word Marks: A word mark must be set over the high-order position of the divisor. A B-field word mark is not necessary; if, however, a word mark is set in the B-field, it is ignored, but retained.

Concepts of Machine Division

To execute a divide operation, the CPU performs a series of subtractions by complement adding the divisor to the dividend, starting with the high-order dividend position. A carry from the high-order position indicates a successful complement add scan and increases the quotient field by one. Successive complement add scans are performed until no adder carry from the high-order position is detected, indicating that the previous subtraction was unsuccessful (overdraw) and that a correction scan must be taken.

In a correction scan, the divisor is true added to the dividend to restore the dividend to its value preceding the unsuccessful subtraction. If the divisor is not true added to the units dividend position (located in the units position of the B-field), a right shift is taken after the correction scan. The CPU repeats the operation, complement adding the divisor to the dividend,

starting with the next dividend digit in low-order sequence.

When a correction scan causes the divisor to be added to the units dividend position (defined by a B-bit in the standard sign), actions to set the quotient sign and end the divide operation are initiated.

At the end of the operation, high-order B-field positions contain the quotient; low-order B-field positions contain the remainder. Zone bits in the units quotient and remainder positions indicate the signs of the factors. When the divide operation is complete:

1. The number of digits in the divisor (A-field) equals the number of B-field positions containing the remainder.

2. The number of digits in the original dividend equals the number of B-field positions containing the quotient.

3. A numeric zero in the B-field separates the remainder from the quotient; for example, when the system divides 6876×55, the B-field contains 0125001 at the end of the operation. Because the divisor contains two digits (55), the two low-order B-field positions contain the remainder (01). Four digits were in the original dividend (6876), therefore, the four high-order positions in the B-field contain the quotient (0125). A numeric 0 separates the remainder from the quotient.

Examples 1, 2, and 3 following illustrate the theory of machine division:

EXAMPLE 1

$147 \div 12$

	0 0 0 1 4 7 (B-field before division)
Complement add divisor to dividend (high-order position)	9 8 8
No adder carry from high-order position	0 9 8 9 4 7
Correction scan (true add divisor to dividend)	1 2
B-field is restored to value preceding overdraw	0 0 0 1 4 7
Shift right and complement add	9 8 8
Successful subtraction	0 1 0 0 2 7
Complement add	9 8 8
No adder carry from high-order position	0 1 9 9 0 7
Correction scan (true add divisor to dividend)	1 2
B-field is restored to value preceding overdraw	0 1 0 0 2 7
Shift right and complement add	9 8 8
Successful subtraction	0 1 1 0 1 5
Complement add	9 8 8
Successful subtraction	0 1 2 0 0 3
Complement add	9 8 8
No adder carry from high-order position	0 1 2 9 9 1
Correction scan (true add divisor to dividend)	1 2
B-field is restored to value preceding overdraw	0 1 2 0 0 3
Because divisor was true added to units position of B-field (indicated by presence of B-bit), apply sign to quotient and end operation.	0 1 2 0 0 3

Because there are three digits in the dividend (147), the three high-order positions in the B-field at the end of the divide operation contain the quotient.

Because there are two digits in the divisor (12), the two low-order B-field positions contain the remainder at the end of the divide operation.

A 0 separates the quotient from the remainder.

EXAMPLE 2

$14980 \div 65$

	0 0 0 1 4 9 8 0 (B-field before division)
Complement add divisor to dividend (high-order position)	9 3 5
No adder carry from high-order position	0 9 3 6 4 9 8 0
Correction scan (true add divisor to dividend)	6 5
	0 0 0 1 4 9 8 0
Shift right and complement add	9 3 5
No adder carry from high-order position	0 0 9 4 9 9 8 0
Correction scan (true add divisor to dividend)	6 5
	0 0 0 1 4 9 8 0

```
                                                0 0 0 1 4 9 8 0
  Shift right and complement add                      9 3 5
                                                0 0 1 0 8 4 8 0
Successful subtraction
  Complement add                                      9 3 5
                                                0 0 2 0 1 9 8 0
Successful subtraction
  Complement add                                      9 3 5
                                                0 0 2 9 5 4 8 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)        6 5
                                                0 0 2 0 1 9 8 0
  Shift right and complement add                      9 3 5
                                                0 0 2 1 0 3 3 0
Successful subtraction
  Complement add                                      9 3 5
                                                0 0 2 2 0 6 8 0
Successful subtraction
  Complement add                                      9 3 5
                                                0 0 2 3 0 0 3 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)      9 3 5
                                                0 0 2 3 9 3 8 0
  Shift right and complement add                        6 5
                                                0 0 2 3 0 0 3 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)      9 3 5
                                                0 0 2 3 0 9 6 5
Because divisor was true added to units position of B-field
(indicated by presence of B-bit), apply sign to quotient
and end operation.                                    6 5
                                                0 0 2 3 0 0 3 0
                                                Ø 0 2 3 0 | 0 | 3 0
```

Because there are five digits in the dividend (14980), the five high-order positions in the B-field at the end of the divide operation contain the quotient.

Because there are two digits in the divisor (65), the two low-order B-field positions contain the remainder at the end of the divide operation.

A 0 separates the quotient from the remainder.

EXAMPLE 3

$\breve{1}5\bar{0}\bar{0} \div \breve{1}0\bar{0}\bar{0}$

```
                                                Ø 0 0 0 0 1 5 0 0 0̄   (B-field before division)
  Complement add divisor to dividend (high-order)     9 0 0 0
                                                0 0 9 0 0 1 5 0 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)      1 0 0 0
                                                0 0 0 0 0 1 5 0 0
B-field restored to value preceding overdraw
  Shift right and complement add                      9 0 0 0
                                                0 0 0 9 0 1 5 0 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)      1 0 0 0
                                                0 0 0 0 0 1 5 0 0
B-field restored to value preceding overdraw
  Shift right and complement add                        9 0 0 0
                                                0 0 0 0 9 1 5 0 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)        1 0 0 0
                                                0 0 0 0 0 1 5 0 0
B-field restored to value preceding overdraw
  Shift right and complement add                          9 0 0 0
                                                0 0 0 0 1 0 5 0 0
Successful subtraction
  Complement add                                          9 0 0 0
                                                0 0 0 0 1 9 5 0 0
No adder carry from high-order position
  Correction scan (true add divisor to dividend)          1 0 0 0
                                                0 0 0 0 1 0 5 0 0
B-field restored to value preceding overdraw

Because divisor was true added to units position of B-field   Ø 0 0 0 1̇ | 0 | 5 0 0̄
(indicated by presence of B-bit), apply sign to quotient
and end operation.
```

Because there are four digits in the dividend (1500), the four high-order positions in the B-field at the end of the divide operation contain the quotient.

Because there are four digits in the divisor (1000), the four low-order B-field positions contain the remainder at the end of the divide operation.

A 0 separates the quotient from the remainder.

Program Conditions that Cause Divide Overflow

The program designating the divide operation must be constructed to anticipate conditions listed below that can cause a divide overflow indication:

1. If the quotient field is two or more positions short, the divide operation usually results in a divide overflow. If the field is one position too small, no overflow indication is given, even though the units position of the adjacent field is changed. A quotient field too small is a programming error, and is not checked by the system.

2. Division by 0 always results in a divide overflow indication.

3. Because only one quotient digit can be developed at a time, it is important to address the high-order position of the dividend (B-address in the divide instruction). This insures that the first divide operation results in a single high-order quotient digit. An improperly addressed dividend can cause a divide overflow if the result of the first series of complement add scans produces a quotient of greater than 9 (Figure 15).

Example 90 ÷ 9

INCORRECT			CORRECT	
The first complement add scan does not begin at the high-order dividend position.	0090 91 CA* 1 0181 91 CA 2 0272 91 CA 3 0363 91 CA 4 0454 91 CA 5 0545 91 CA 6 0636 91 CA 7 0727 91 CA 8 0818 91 CA 9 0909 91 CA10 1000		The first complement add scan begins at the high-order dividend position.	0090 91 1000 91 1910 9 1000 91 1091 9 1000
DIVIDE OVERFLOW First series of complement add scans produces a quotient of greater than 9				

*CA is abbreviation for complement add scan

Figure 15. Examples of Dividend Addressing

Address Registers

In a multiply operation:

a. The AAR scans the A-field (divisor).

b. The BAR scans the dividend positions used in each scan.

c. The CAR retains the address of the units position of the divisor.

d. The DAR retains the address of the position in the dividend field used at the start of each scan; the DAR is modified by +1 on shift cycles.

At the end of the divide operation, the IAR contains the address of the next sequential instruction; the AAR contains the address equal to the original A-address minus the number of characters in the A-field; the BAR contains the address of the tens position of the quotient field.

CPU Operation

To perform the divide operation, the CPU executes A- and B-cycles to read out divisor (A-field) and dividend (B-field) characters. A-field characters are gated to the A-data register on A-cycles and to the A-channel on succeeding B-cycles. B-field characters are gated to the B-channel on B-cycles. The adder unit true or complement adds the A-channel character to the B-channel character as determined by the ON and OFF states of the true and complement latches. The adder unit output is gated through assembly to storage.

The high-order character in the divisor contains a defining word mark. The A-field character containing the word mark is read from storage and set in the A-data register during a normal A-cycle. On the B-cycle that follows, the high-order divisor digit is gated to the A-channel and combined with the corresponding B-channel character in the adder. The adder output is gated through the assembly unit to storage. Because the last character in the A-field has been processed (as indicated by the A-channel word mark), the extension latch is set, initiating another (the second successive) B-cycle.

End of Correction Scan and Shift Cycle

When the CPU is performing a correction scan (true addition) and the extension latch initiates the second successive B-cycle, the next B-field character in high-order sequence is read out of storage and gated onto the B-channel. The adder unit combines the B-channel character with a 0 automatically inserted on the A-channel input to the adder; the adder output is gated through the assembly unit to storage, ending the true add (correction) scan. Shift and complement add cycles follow.

To accomplish the shift, the CPU sets the second scan latch and takes a D-cycle to increase the DAR address by one. The character read out of storage on the D-cycle is returned to its storage location unchanged. At the end of the D-cycle, the units, complement, and third scan latches are set, initiating a complement add scan. The CPU uses the CAR to read out the units position of the A-field and the DAR (modified by +1 in the preceding shift cycle) to read out the B-field

character. A- and B-field characters are read out of storage on alternate A- and B-cycles until the divisor digit containing the word mark is sensed on the A-channel again.

End of Complement Add Scan

When the CPU is performing a complement add scan and the extension latch initiates the second successive B-cycle, the next B-field character in high-order sequence is read out of storage and gated onto the B-channel. The adder unit combines the B-channel character with a 9 automatically inserted on the A-channel input to the adder; the adder output is gated through the assembly unit to storage. If the sum of the B-channel character and the 9 inserted on the A-channel input to the adder does not require a carry, the reduction scan (complement add scan) was unsuccessful, and the complement add scan ends. The CPU then takes a true ad scan to restore the dividend to its value preceding the unsuccessful subtraction. To initiate the true add scan, the true, units, and third scan latches are set, and the CAR and DAR are used to address storage on the first A- and B-cycles, respectively, in the true add (correction) scan.

If the sum of the B-channel character and the 9 inserted on the A-channel input to the adder requires a carry, the MQ and carry latches are set, and the complement latch is regenerated. The ON states of the complement and MQ latches indicate a successful reduction scan and initiate another (the third successive) B-cycle to increase a quotient digit by one. During the B-cycle, a digit in the quotient field is read out of storage, gated onto the B-channel, and into the adder unit. A 0 is automatically inserted on the A-channel input to the adder. The adder combines the B-channel character (a quotient digit) with the carry that resulted from the addition on the previous B-cycle, increasing the character by one. The character is gated to storage. If the sum of the B-channel character and the carry is greater than 9, requiring the carry latch to be set again, the overflow latch is set, and the divide operation is terminated. No carry should result from the addition performed when the MQ and complement add latches are set. If the addition does not produce a carry, the successful reduction scan ends, and another complement add scan is begun. The MQ latch is reset, the units latch is set, and the CAR and DAR are used to read out A- and B-field characters, respectively.

Set Quotient Sign and End Divide

On the first A- and B-cycles in the last correction scan in the divide operation, characters in the units positions of the A- and B-fields are read out of storage. The character in the units position of the A-field contains the divisor sign. The character in the units position of the B-field contains the dividend sign. Only the units character in the B-field can contain a B-bit. When a B-bit is detected on the B-channel in a true add scan, signs of characters on the A- and B-channels are analyzed, and the plus latch or minus latch and the MDL (multiply divide last) latch are set. The CPU takes A- and B-cycles to execute the true add scan in the normal manner. The first character gated to storage in the scan takes the B-channel sign; the remainder always takes the sign of the original dividend. Because the MDL latch is set, shift and complement add cycles are not taken when the true add scan is complete, but rather the MQ latch is set, and the true latch is regenerated. A B-cycle is taken to read the units digit in the quotient field out of storage and onto the B-channel. The adder unit combines the units quotient field digit with:

1. A 9 automatically inserted on the A-channel input to the adder.
2. The carry left over from the previous correction scan. The carry latch is set at the end of each correction scan, but the carry is used only when the MDL latch is set.

Therefore, when the adder unit completes the addition, the output digit equals the original B-channel input digit, and the carry latch is set.

(Units Quotient Field Digit + Carry + 9 = Units Quotient Field Digit + 9)

The correct quotient sign (as determined by the on state of the plus latch or minus latch) is set over the output digit from the adder unit. The digit and the sign are returned to storage. At the end of the B-cycle, the divide operation ends.

Figure 16 uses a specific example to illustrate the step-by-step action that the CPU performs to execute the divide instruction. Figure 17 is a data flow diagram of the divide operation.

Questions on Divide Operation

Answers to these review questions are in the Appendix.

1. What condition causes the CPU to set the divide overflow latch?

2. a. Why is a correction scan necessary?

b. When does the CPU recognize an unsuccessful reduction?

3. When are D-cycles taken?

4. When is the multiply divide latch set?

5. When are the CAR and DAR used in the divide operation?

6. When does the CPU set the quotient sign?

DIVIDE PROBLEM: B FIELD V 0 0 0 7 2 0 9 + A FIELD V − 8 5

LOCATION OF D AR

OBJECTIVES	CYCLE	A/R RO	UNITS BODY EXTN MQ	SCAN 1ST 2ND, 3RD NO	STORED "B" FIELD	DIGIT ON ASSM	ADDER "B"	CARRY NO CARRY	ADDER "A"	TRUE OR COMP	REMARKS
REDUCTION- COMP ADD	A B	C AR D AR	U	1	V . . 2 . . +	2	7	C	4	S	UNITS A 1000'S B
SCAN	A B	A AR B AR	Y	1	. . 2 . . .	2	0	C	1	S	
OVERDRAW	B	B AR	X	1	. 9	9	0	$\overline{C}$	9	S	
					0 9 2 2 2 0 9						
CORRECTION-TRUE ADD	A B	C AR D AR	U	3	. . . 7 . .	7	2	$\overline{C}$	5	T	UNITS A 1000'S B
SCAN	A B	A AR B AR	Y	3	. . 0 . . .	0	2	$\overline{C}$	8	T	
	B	B AR	X	3	. 0	0	9	C	0	T	
					0 0 0 7 2 0 9						
MODIFY D AR + 1	D	D AR		2	. . . . ▼ . .						SHIFT
REDUCTION- COMP ADD	A B	C AR D AR	U	3	. . . 7 . .	7	2	C	4	S	UNITS A 100'S B
SCAN	A B	A AR B AR	Y	3	. . . 8 . .	8	7	$\overline{C}$	1	S	
OVERDRAW	B	B AR	X	3	. . 9 . . .	9	0	$\overline{C}$	9	S	
					0 0 9 8 7 0 9						
CORRECTION-TRUE ADD	A B	C AR D AR	U	3	. . . . 2 . .	2	7	$\overline{C}$	5	T	UNITS A 100'S B
SCAN	A B	A AR B AR	Y	3	. . . 7 . . .	7	8	C	8	T	
	B	B AR	X	3	. . 0	0	9	C	0	T	
					0 0 0 7 2 0 9						
MODIFY D AR + 1	D	D AR		2	. . . . . ▼						SHIFT
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 5	5	0	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 3 . .	3	2	$\overline{C}$	1	S	
	B	B AR	X	3	. . . 6 . . .	6	7	$\overline{C}$	9		
ACCUMULATE QUOT	B	B AR	MQ	3	. . 1	1	0	C	0	S	
					0 0 1 6 3 5 9						
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 0	0	5	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 5 . .	5	3	C	1	S	
	B	B AR	X	3	. . . 5 . . .	5	6	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3	. . 2	2	1	C	0	S	
					0 0 2 5 5 0 9						
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 5	5	0	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 6 . .	6	5	$\overline{C}$	1	S	
	B	B AR	X	3	. . . 4 . . .	4	5	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3	. . 3	3	2	C	0	S	
					0 0 3 4 6 5 9						
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 0	0	5	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 8 . .	8	6	C	1	S	
	B	B AR	X	3	. . . 3 . . .	3	4	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3	. . 4	4	3	C	0	S	
					0 0 4 3 8 0 9						
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 5	5	0	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 9 . .	9	8	$\overline{C}$	1	S	
	B	B AR	X	3	. . . 2 . . .	2	3	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3	. . 5	5	4	C	0	S	
					0 0 5 2 9 5 9						
REDUCTION- COMP ADD	A B	C AR D AR	U	3	 0	0	5	C	4	S	UNITS A TENS B
SCAN	A B	A AR B AR	Y	3	. . . . 1 . .	1	9	C	1	S	
	B	B AR	X	3	. . . 2 . . .	2	2	C	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3	. . 6	6	5	C	0	S	
					0 0 6 2 1 0 9						

Figure 16A. Divide Example

REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					5	5	0	C	4	S	UNITS A TENS B
	A B	A AR B AR	Y	3				2		2	1	$\overline{C}$	1	S	
	B	B AR	X	3			1			1	2	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3		7				7	6	C	0	S	
					0	0	7	1	2	5	9				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					0	0	5	C	4	S	UNITS A TENS B
	A B	A AR B AR	Y	3				4		4	2	C	1	S	
	B	B AR	X	3			0			0	1	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3		8				8	7	C	0	S	
					0	0	8	0	4	0	9				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					5	5	0	C	4	S	UNITS A TENS B
	A B	A AR B AR	Y	3				5		5	4	$\overline{C}$	1	S	
OVERDRAW	B	B AR	X	3			9			9	0	$\overline{C}$	9	S	
					0	0	8	9	5	5	9				
CORRECTION-TRUE ADD SCAN	A B	C AR D AR	U	3					0	0	5	$\overline{C}$	5	T	UNITS A TENS B
	A B	A AR B AR	Y	3				4		4	5	C	8	T	
	B	B AR	X	3			0			0	9	C	0	T	
					0	0	8	0	4	0	9				
MODIFY D AR + 1	D	D AR		2											SHIFT
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					4	4	9	C	4	S	UNITS A UNITS B
	A B	A AR B AR	Y	3				2		2	0	C	1	S	
	B	B AR	X	3			3			3	4	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3			1			1	0	C	0	S	
					0	0	8	1	3	2	4				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					9	9	4	C	4	S	UNITS A UNITS B
	A B	A AR B AR	Y	3				3		3	2	$\overline{C}$	1	S	
	B	B AR	X	3			2			2	3	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3			2			2	1	C	0	S	
					0	0	8	2	2	3	9				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					4	4	9	C	4	S	UNITS A UNITS B
	A B	A AR B AR	Y	3				5		5	3	C	1	S	
	B	B AR	X	3			1			1	2	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3			3			3	2	C	0	S	
					0	0	8	3	1	5	4				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					9	9	4	C	4	S	UNITS A UNITS B
	A B	A AR B AR	Y	3				6		6	5	$\overline{C}$	1	S	
	B	B AR	X	3			0			0	1	$\overline{C}$	9	S	
ACCUMULATE QUOT	B	B AR	MQ	3			4			4	3	C	0	S	
					0	0	8	4	0	6	9				
REDUCTION- COMP ADD SCAN	A B	C AR D AR	U	3					4	4	9	C	4	S	UNITS A UNITS B
	A B	A AR B AR	Y	3				8		8	6	C	1	S	
OVERDRAW	B	B AR	X	3			9			9	0	$\overline{C}$	9	S	
					0	0	8	4	9	8	4				
CORRECTION-TRUE ADD SCAN	A B	C AR D AR	U	3					9	9	4	$\overline{C}$	5	T	UNITS A UNITS B
	A B	A AR B AR	Y	3				6		6	8	$\overline{C}$	8	T	MDL "ON"
	B	B AR	X	3			0			0	9	C	0	T	
END OPERATION	B	B AR	MQ	3			4			4	4	C	9	T	
					V 0	0	8	4	0	6	+ 9				
					QUOTIENT					REMAINDER					

Figure 16B. Divide Example

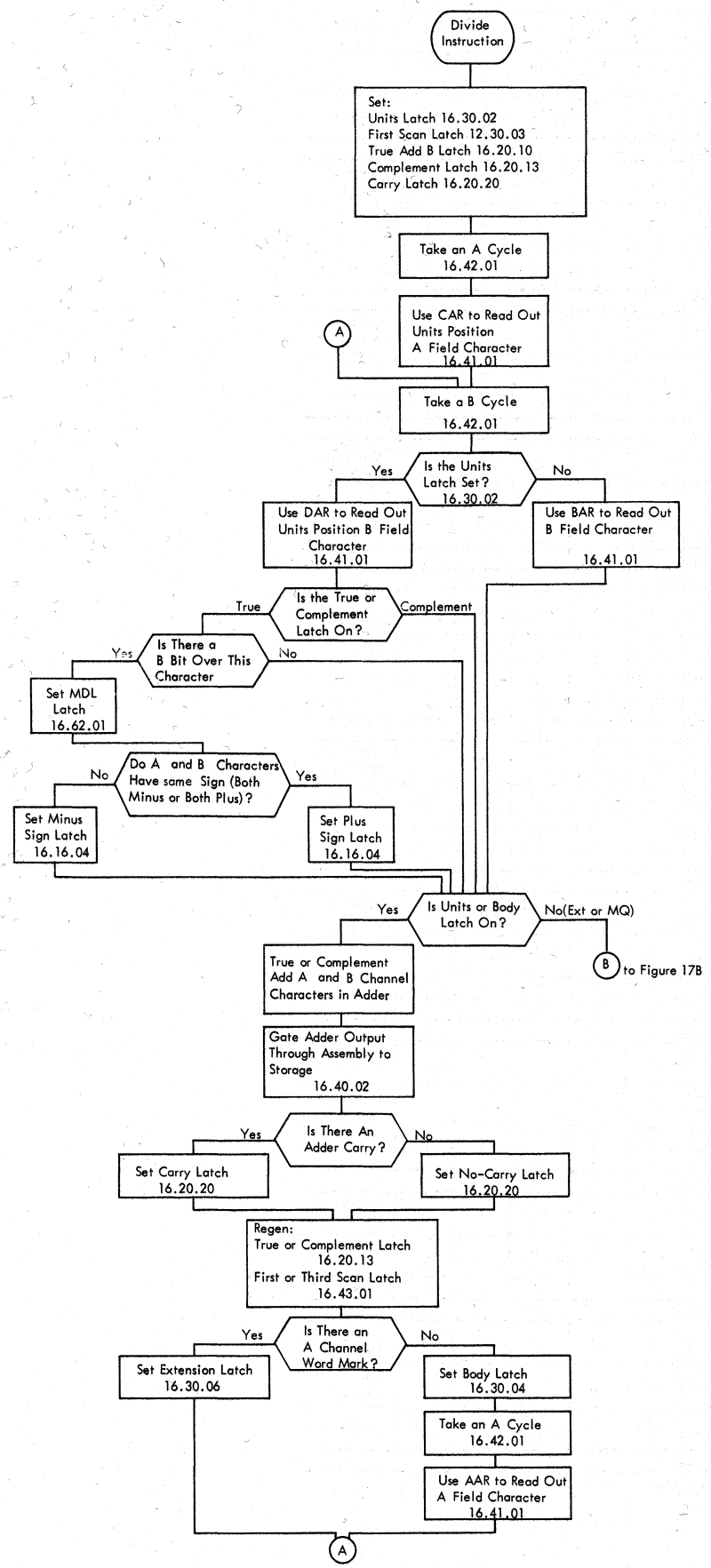


Figure 17A. Divide

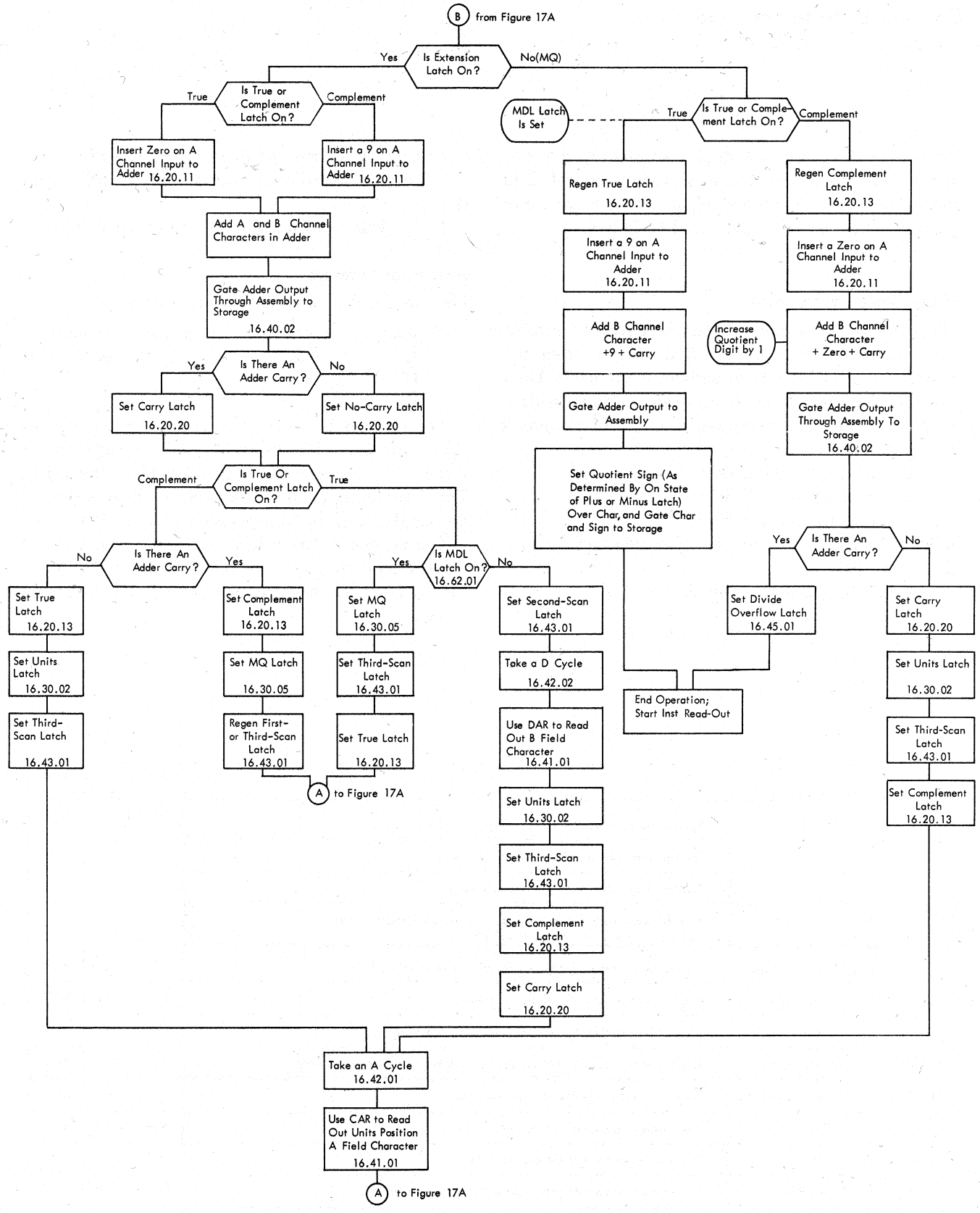


Figure 17B. Divide

General Data Instructions

When executing general data instructions, the CPU manipulates data within core storage. General data instructions include move data, move characters and suppress zeros, edit, compare, and table lookup.

Move Data Instruction

The move data instruction causes the system to transfer data, either left to right or right to left, from the A-field to the B-field with or without word marks. Data are moved either by fields or records. When a data field is moved, the operation can be programmed to stop at:

1. A word mark in the A-field.
2. A word mark in the B-field.
3. A word mark in either field.

When a record is moved, the operation can be programmed to stop at:

1. A record mark in the A-field.
2. A group mark-word mark in the A-field.
3. Either a record mark or group mark-word mark in the A-field.
4. The first word mark sensed in either field.

Instruction Formats

Formats for the move data instruction are:

OP CODE	A-ADDRESS	B-ADDRESS	D-CHARACTER
V D	XXXXX	XXXXX	see Figure 18
V D	XXXXX		
V D			

The move data instruction causes the CPU to move characters from left to right or from right to left, serially by character, from the A-field to the B-field. The d-character in the instruction establishes the conditions that control the operation (Figure 18).

The portion of the A-field transferred replaces only the corresponding portion of the B-field. If data are moved from left to right, the A-address specifies the

d-Character Control Bits		Control	Address Registers after Move Operation**		
			IAR	AAR	BAR
1		Transfer numeric portion of data field (A-field).			
2		Transfer zone portion of data field (A-field).			
4		Transfer word marks from A-field to B-field.			
(No 1, 2, or 4-bits)		Scan for word marks, record marks, or group mark-word marks; transfer no data from A-field or B-field.			
8-bit (Move data left to right)	No A- and no B-bits	Stop transfer or scan at first word mark sensed in either field.	NSI	A+LW	B+LW
	*A-bit only	Stop transfer or scan at A-field record mark.	NSI	A+LA	B+LA
	B-bit only	Stop transfer or scan at A-field group mark-word mark.	NSI	A+LA	B+LA
	A- and B-bits	Stop transfer or scan at A-field record mark or group mark-word mark.	NSI	A+LA	B+LA
No 8-bit (Move data right to left)	No A- and no B-bits	Transfer or scan only one storage position.	NSI	A-1	B-1
	*A-bit only	Stop transfer or scan at A-field word mark.	NSI	A-LA	B-LA
	B-bit only	Stop transfer or scan at B-field word mark.	NSI	A-LB	B-LB
	A- and B-bits	Stop transfer or scan at first word mark sensed in either field.	NSI	A-LW	B-LW

* When the A-bit d-character modifier is used in instructions to write programs on tape, the odd parity mode should be used.
** See Appendix for list of abbreviations.

Figure 18. d-Character Control Bits for Move Data Instructions

high-order position of the A-field; the B-address specifies the high-order B-field position. If data are moved from right to left, the A-address specifies the low-order A-field position; the B-address specifies the low-order position of the B-field. The position that contains the terminating character is moved or replaced as other characters in the field.

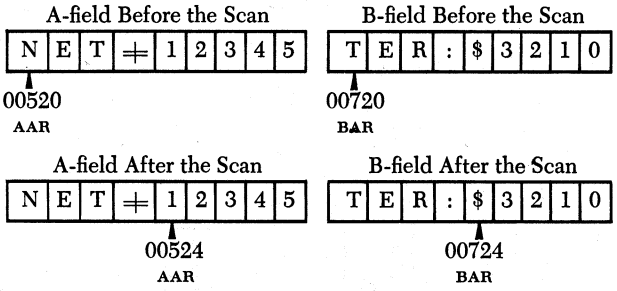
If the move data instruction does not designate an A- or B-field address (no address chained instruction), contents of the AAR, BAR, and op-modifier register specify the A-field, B-field, and d-character, respectively, for the move data operation. If the move data instruction specifies an A-address and no B-address or d-character, the contents of the BAR and the op-modifier register designate the B-address and the d-character, respectively, for the move data operation.

Scan Operation

The move data instruction, with the appropriate d-characters (no 1-, 2-, or 4-bit), is used for scan operations in which no data are transferred from the A-field to the B-field. The following example illustrates a scan operation:

INSTRUCTION: D̆ 00520 00720 Y

The most important results shown are the contents of the address registers after the operation. No data are transferred. The B-address must be part of the instruction, even if, as in the example, the scan is for the first record mark in the A-field exclusively (the d-character, Y, has an A-bit). Because the scan is from left to right (the d-character, Y, has an 8-bit), the A- and B-addresses specify the high-order positions of the respective fields.



CPU Operation

During last instruction read-out cycle:

1. The d-character in the op-modifier register is examined to determine whether the character contains an 8-bit.

2. Op code grouping lines condition controls to execute a standard A-cycle first.

If the d-character in the op-modifier register contains an 8-bit, the second scan control latch is set at next to last logic gate of last instruction read-out cycle. If the d-character in the op-modifier register does not contain an 8-bit, the first scan control latch is set.

During A-cycles in the move data operation, the AAR addresses storage. The A-field character that the AAR specifies reads out of storage and is gated into the A-data register.

A B-cycle follows each A-cycle in the operation. During each B-cycle, the character that the BAR specifies reads out of storage and is gated onto the B-channel; the character in the A-data register is gated to the A-channel. The 1-, 2-, and 4-bits of the character in the op-modifier register condition assembly controls to replace designated portions of the B-channel character with corresponding portions of the A-channel character. The resultant character is stored in the B-field in memory.

During each B-cycle, the 8-, A-, and B-bit positions in the character in the op-modifier are examined to determine whether conditions to end the move data operation are met. If conditions, that the d-character specifies to end the operation are satisfied, more A- and B-cycles are not executed, and the operation terminates when the present B-cycle is complete.

Figures 19 and 20 show diagrammed explanations of CPU operation in the execution of the move data instruction.

The following controls are active when the CPU performs the move data operation:

Signal	Control	Logic
1. Initiate A-cycle and RO first A-field character.		
Set A Cy Ctrl	A Cy First Op Codes Last Insn RO Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
2. Set modifier controls according to 8-bit d-modifier.		
Set 1st Scan Ctrl	Last Insn RO, Data Move Op Code Op Mod Reg Not 8-Bit	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
NOTE: Second scan is set when operation modifier is 8-bit.		
3. Set character into A-data register.		
Sw B Ch to A Reg	A Cy, LGD	15.38.01
4. Control A-cycle length.		
ARS D or T Op Codes	Data Move Op Code	13.14.07
Std A Cy Ops A Cy	ARS D or T Op Codes, A Cy	13.14.06
Stop at F	Std A Cy Ops, A Cy	12.12.30
5. Initiate B-cycle and RO first B-field character.		
Set B Cy Ctrl	Std A Cy Ops, A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02

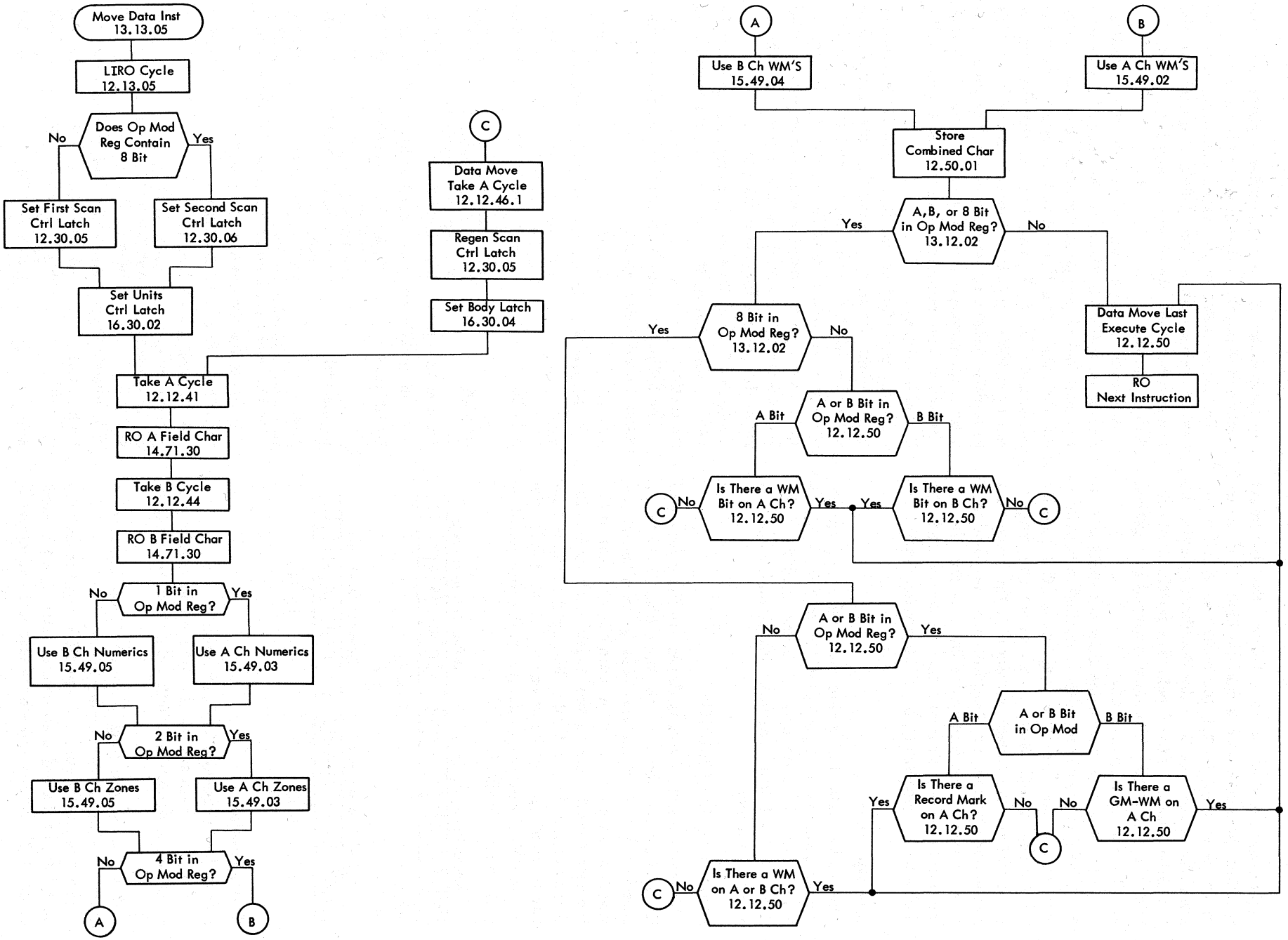


Figure 19. Move Data

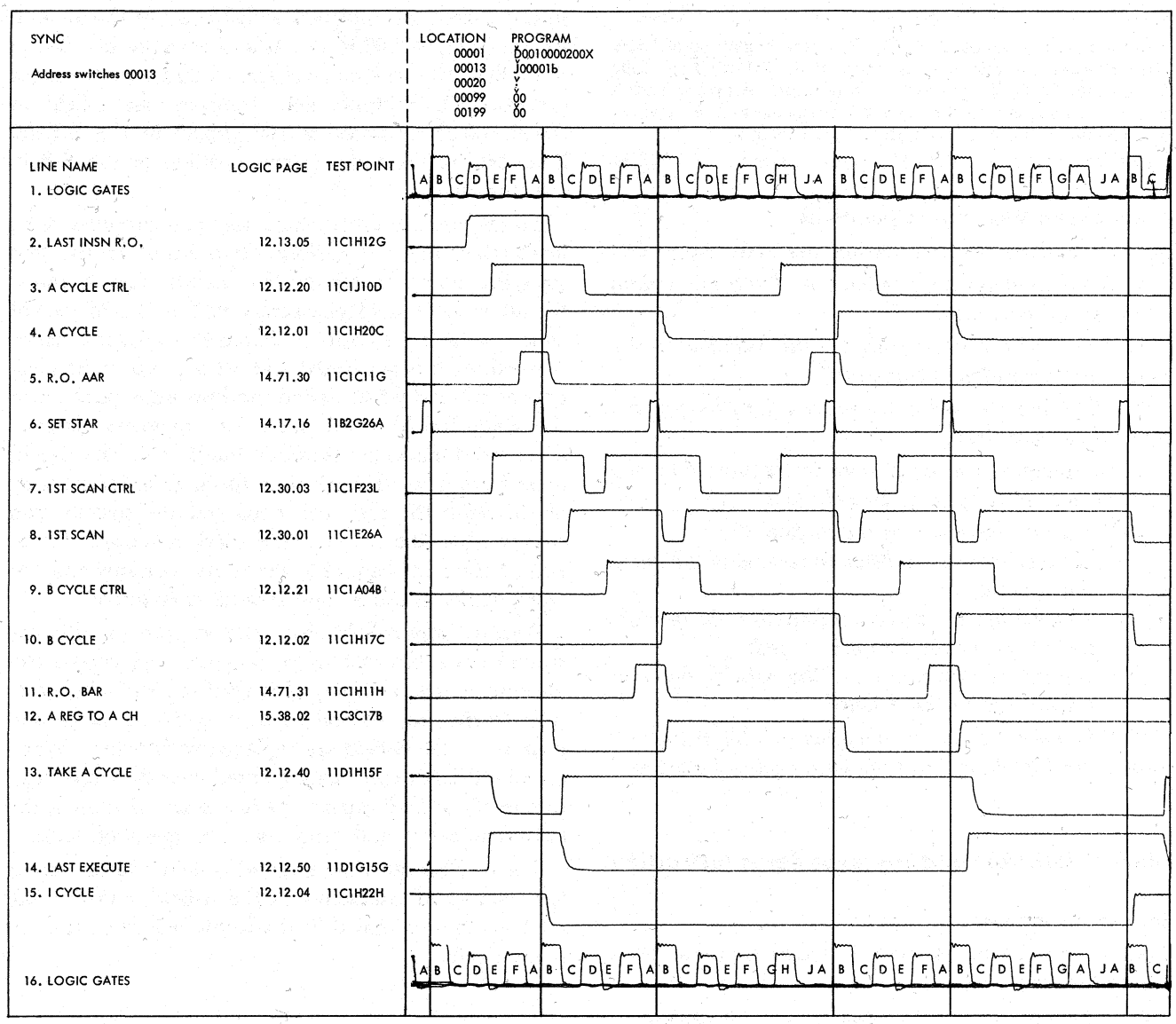


Figure 20. Data Move Operation Timings

Signal	Control	Logic
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl	14.71.31
Set Mem AR Gated	RO B AR on Scan B Cy Ops LGA, 2nd CP	14.17.16

6. Gate A-field character to A-channel.

Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02

7. Set assembly unit controls according to operation modifier 1-, 2-, and 4-bits.

8. Read character into storage.

Load Memory	Load Mem on B Cy Op Codes, B Cy	12.50.01

9. Control B-cycle length.

Signal	Control	Logic
Stop at J	Stop at J on B Cy Op Codes, B Cy	12.12.32

10. Compare the 8-, A-, and B-bits of the operation modifier register with the A- or B-field character to determine end of operation.

Data Move Take A Cy		12.12.40
Data Move A Cy Ctrl Set	Data Move Take A Cy	12.12.41
Set A Cy Ctrl	B Cy, Data Move A Cy Ctrl Set, Data Move Op Code	12.12.20

When the A- or B-field character satisfies the operation modifier character stop condition, Data Move Take A-Cycle

Signal	Control	Logic

drops around the end of Logic Gate C or the beginning of Logic Gate D when the character is set into the B-Data Reg. Data Move Take A-Cycle prevents taking more A-cycles and a similar combination of the op mod character and A- and B-field characters bring up Data Move Last Execute.

Last Execute Cycle	Data Move Last Execute	12.12.51

Questions on Move Data Operations

Answers to these review questions are in the Appendix.

1. *What function does execution of the move data instruction accomplish?*
2. *What characters are stored in the A-field after the move data operation is complete?*
3. *What are the valid instruction lengths for the move data instruction?*
4. *In executing the move data instruction, does the* CPU *execute an A-cycle or B-cycle first?*
5. *What bit position(s) in the d-character:*
   a. *Establishes the condition to terminate the move data operation?*
   b. *Determines the direction (left to right or right to left) in which data are moved?*
   c. *Determines the portion of the A-field character transferred to the B-field?*
6. *When the d-character contains an 8-bit, is the first scan control latch or the second scan control latch set?*

Move Characters and Suppress Zeros Instruction

Instruction Formats

Formats for the move characters and suppress zeros instruction are:

Op code	A-address	B-address
Z	xxxxx	xxxxx
Z	xxxxx	
Z		

The move characters and suppress zeros instruction designates:

1. The addresses of the low-order characters in the A- and B-fields, or
2. The address of the low-order A-field character; the contents of the BAR from the previous operation specify the low-order B-field character, or
3. Neither the A- nor B-field (no address chaining). The contents of the AAR and the BAR from the previous operation specify the low-order A- and B-field characters, respectively.

Description of Operation

The move characters and suppress zeros operation transfers A-field data to the B-field. Zeros and commas to the left of the first (high-order) significant digit in the A-field are replaced with blanks in the B-field; for example, 001206 in the A-field becomes bb 1206 in the B-field after the move character and suppress zeros operation. In multiple field transfers, an A-field of 00106@00.25 becomes bb 106@bb.25 in the B-field. Zone bits in the units (sign) position of the B-field are removed.

To execute the move characters and suppress zeros instruction, the CPU performs two scans. In the first scan, the CPU alternately takes A- and B-cycles to transfer all other A-field characters to the B-field except zones in the units position character and word marks. A word mark is generated and set over the units position of the B-field to define the low-order position of the field; B-field word marks are removed, and no other word marks are stored in the B-field. The A-field must have a word mark over the high-order position of the field; the first scan ends and the second scan begins when the A-field word mark is sensed. At the end of the first scan, only the units positions and the word marks in the A- and B-fields may differ.

Because only B-field characters are processed in the second scan, the CPU takes B-cycles exclusively. Reverse scanning is employed to read out B-field characters. During the second scan, insignificant zeros and commas in the B-field are replaced with blanks. When the one B-field word mark (stored over the units position of the B-field during the first scan) is sensed, the move characters and suppress zeros operation ends.

The AAR contains the original A-field address minus the number of characters in the A-field, and the BAR contains the original B-field address +1 when the operation ends.

CPU Operation

During last instruction read-out cycles, the first scan control, A-cycle control, and units control latches are set. To begin execute phase of the move characters and suppress zeros operation, the CPU takes a standard A-cycle first. The low-order A-field character reads out of storage and is set in the A-data register. On the B-cycle following the first A-cycle, the units position of the B-field reads out of storage onto the B-channel; the character in the A-data register is gated to the A-channel. Because the units control latch is set, A- and B-channel characters are identified as units position characters of the respective fields. The assembly gates A-channel numerics, no zones, and a word mark to the units position of the B-field. The zero suppress latch is set during the first B-cycle, but the latch does not function to blank characters until the second scan.

On B-cycles executed while the first scan latch is set, A-channel characters are examined for a word mark.

If no A-channel word mark is detected during the first B-cycle, the units latch is reset, and the body latch is set. With the first scan and body latches set, the CPU alternately executes A- and B-cycles to read A- and B-field characters out of storage and transfer A-channel zones and numerics to the B-field until an A-channel word mark is detected. B-channel zones, numerics, and word marks are replaced and do not affect the operation.

When the A-channel word mark is detected, indicating that all A-field characters have been processed, the first scan latch is reset; the second scan, MQ, and B-cycle control latches are set to initiate the second scan.

The CPU executes B-cycles exclusively during the second scan, therefore, reverse scanning is employed to read out B-field core storage positions for the second time in the operation. Because the BAR was modified by −1 on the last B-cycle of the first scan, the address in the BAR at the end of the first scan is one less than the address of the high-order B-field character. The addressed character is read from storage onto the B-channel, through the assembly, and back to storage intact on an edit skid cycle. The ON status of the MQ latch identifies this special cycle and prevents the operation from ending at this point. During the skid cycle, the BAR is modified by +1 (second scan) to address the high-order B-field position, the extension latch is set, and the MQ latch is reset.

The CPU takes successive B-cycles to read out B-field characters. The zero suppress latch, set during the first B-cycle of the first scan, causes B-channel zeros and commas read before the first significant digit is detected to be replaced with blanks; other characters are gated back to the B-field without changes. The first significant digit detected on the B-channel resets the zero suppress latch to prevent blanking significant zeros and commas following the digit.

If, after the zero suppress latch is reset, a B-channel character is found to be neither a significant digit (1 through 9) nor a 0, comma, blank, minus sign, or decimal, the zero suppress latch is set again. For example, if the B-field contains the characters 0010¢090, the zero suppress latch is on when the two high order zeros are read, and the zeros are replaced with blanks. The zero suppress latch is reset when the first significant digit (1) is sensed, and the 0 following the 1 is not affected. Because the cent sign (¢) is not a significant digit, 0, comma, blank, minus sign, or decimal, the 0 suppress latch is set again. The 0 following the cent sign is replaced with a blank. When the 9 is sensed, the zero suppress latch is reset, and the low-order 0 following the 9 is not blanked. The B-field changes from 0010¢090 to bb10¢b90 during the second scan.

When the B-field word mark (stored over the units position of the B-field on the first B-cycle of the first scan) is detected, indicating the end of the B-field:

1. The word mark is stripped from the character.
2. The character is processed as other second scan B-field characters.
3. The move characters and suppress zeros operation ends.

Figures 21 and 22 show diagrammed explanations of CPU operation in the execution of the move characters and suppress zeros instruction.

The following controls are active when the CPU performs the move characters and suppress zeros operation:

Signal	Control	Logic
1. Initiate A-cycle and RO first A-field character.		
Set A Cy Ctrl	A Cy 1st Op Codes Last Insn RO Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO A AR	A Cy Ctrl, LG Special A Read out A AR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
2. Set modifier controls to −1.		
Set 1st Scan Ctrl	1st Scan 1st Op Code Last Insn RO Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
3. Set character into A-data register.		
Sw B Ch to A Reg	A Cy, LGD	15.38.01
4. Control A-cycle length.		
Std A Cy Ops A Cy	E or Z Op Code, A Cy	13.14.06
Stop at F	Std A Cy Ops, A Cy	12.12.30
5. Initiate B-cycle and RO first B-field character.		
Set B Cy Ctrl	Std A Cy Ops, A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
6. Regenerate modify controls.		
Regen 1st Scan Ctrl	Std A Cy Ops, A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
7. Gate A-field character to A-channel.		
Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02
8. Set assembly controls to eliminate zones and write a WM over the units position character.		
Units Ctrl Latch	Last Insn RO Cy, Next to Last LG	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02
Regen Units + Body Ctrl	Std A Cy Ops A Cy	16.30.01

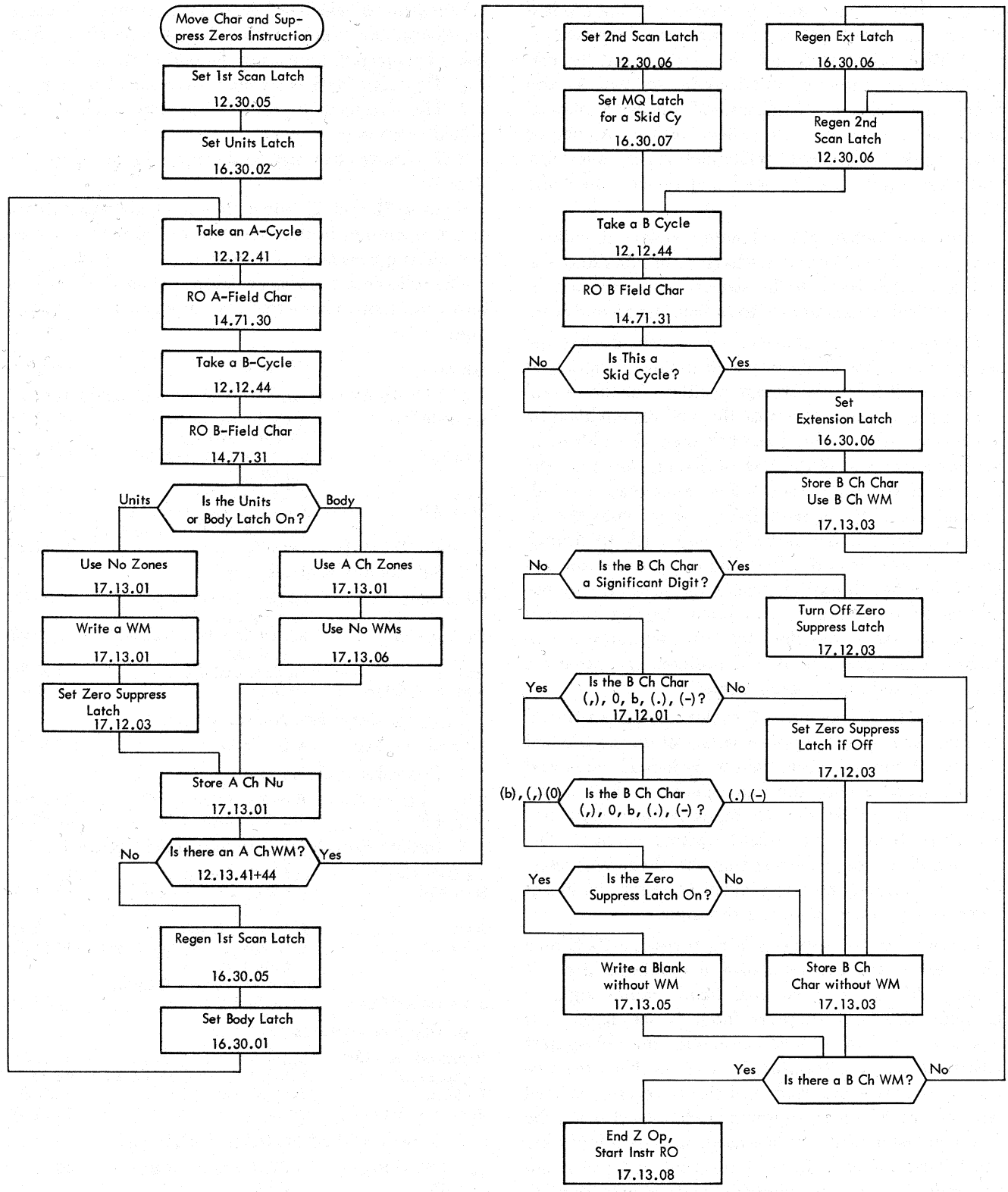


Figure 21. Move Characters and Suppress Zeros

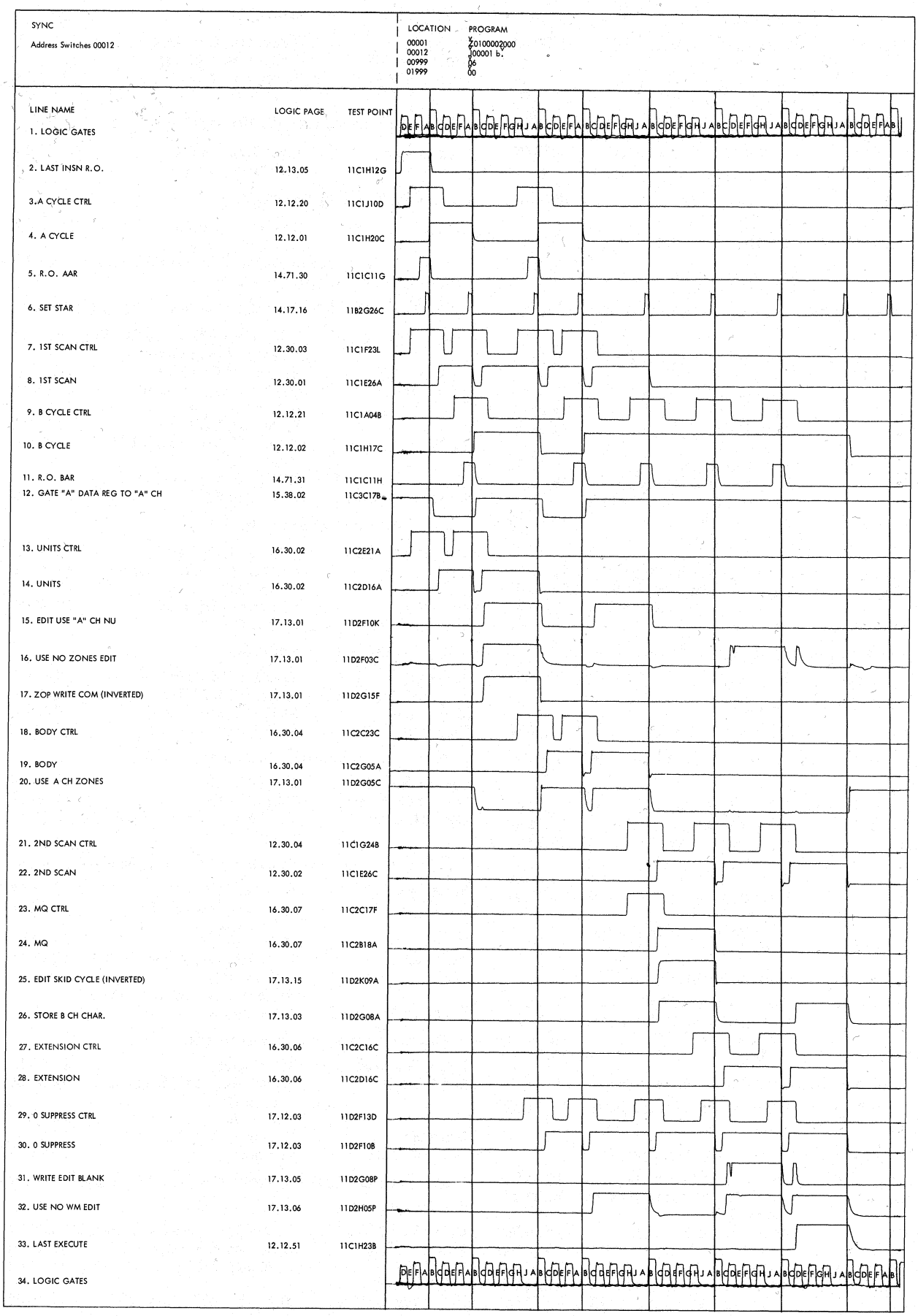


Figure 22. Move Characters and Suppress Zeros Operation Timings

Signal	Control	Logic
Units Ctrl Latch	Regen Units + Body Ctrl, Next to LLG Units Latch	16.30.02
Z Op B Cy	Z Op Code, B Cy	17.11.04
Edit Use A Ch Nu	Z Op B Cy – Units Latch	17.13.01
Use No Zones Edit	Z Op B Cy – Units Latch	17.13.01
Z Op Write WM	Z Op B Cy – Units Latch	17.13.01

9. Read character into storage.

Load Memory	Load Mem on B Cy Op Codes, B Cy	12.50.01

10. Control B Cy Length.

Stop at J	Stop at J on B Cy Op Codes, B Cy	12.12.32

11. Initiate another A-cycle and RO next A-field character.

Set A Cy Ctrl on Z Op	Edit Use A Ch Nu, A Ch Not WM Move Zero Sup Op Code	12.12.41
Set A Cy Ctrl	Set A Cy Ctrl on Z Op	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO A AR	A Cy Ctrl, LG Special A Read Out A AR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd Cp	14.17.16

12. Regenerate modify control.

Regen 1st Scan Ctrl	Set A Cy Ctrl on Z Op	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

13. Set character into A-data register.

Sw B Ch to A Reg	A Cy, LGD	15.38.01

14. Control A-cycle length.

Stop at F	Std A Cy Ops, A Cy	12.12.30

15. Initiate another B-cycle and RO next B-field character.

Std A Cy Ops A Cycle	E or Z Op Code, A Cy	13.14.06
Set B Cy Ctrl	Std A Cy Ops, A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

16. Regenerate modify controls.

Regen 1st Scan Ctrl	Std A Cy Ops, A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

17. Gate A-field character to A-channel.

Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02

18. Set assembly controls for next position.

Set Body Ctrl Latch	Set A Cy Ctrl on Z Op	16.30.01
Body Ctrl Latch	Set Body Ctrl Latch, Next to Last LG	16.30.04
Body Latch	Body Ctrl Latch, LGC	16.30.04
Z Op B Cy	Z Op Code, B Cy	17.11.04
Use a Ch Zones Edit	Z Op B Cy, Body Latch	17.13.01
Edit Use A Ch Nu	Z Op B Cy, Body Latch	17.13.01
0 Suppress Ctrl	Z Op B Cy, Z Op Write WM	17.12.03
0 Suppress	0 Suppress Ctrl, LGC	17.12.03
0 Suppress Ctrl	0 Suppress, 1st Scan	17.12.03
Use No WM Edit	0 Suppress, Z Op B Cy, 1st Scan	17.13.06

19. Read character into storage.

Load Memory	Load Mem on B Cy Op Codes, B Cy	12.50.01

20. Control B-cycle length.

Stop at J	Stop at J on B Cy Op Code, B Cy	12.12.32

If the A-field character has no WM steps 11-20 are repeated until there is an A-Channel WM.

21. Take another B-cycle and RO B-field character.

Edit Set B Cy Ctrl G	Edit Use A Ch Nu, A Ch WM Move Zero Sup Op Code	12.12.44
Set B Cy Ctrl	Edit Set B Cy Ctrl G	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

22. Set modify controls to +1.

Set 2nd Scan Ctrl	Edit Set B Cy Ctrl G	12.30.06
2nd Scan Ctrl	Set 2nd Scan Ctrl, Next to Last LG	12.30.04
2nd Scan	2nd Scan Ctrl, LGC	12.30.02
Addr Mod Set to +1	2nd Scan	14.71.41

23. Set skid cycle controls.

Set M Q Ctrl	Edit Set B Cy Ctrl G	16.30.05
M Q Ctrl Lat	Set M Q Ctrl, Next to Last LG	16.30.07
M Q Latch	M Q Ctrl Lat, LGC	16.30.07
Edit Skid Cy	M Q Latch, Z Op B Cy	17.13.15

24. Store B-channel character.

Use B Ch WM	Edit Skid Cy	15.49.04
Store B Ch Char	Edit Skid Cy	17.13.03
Load Memory	Load Mem on B Cy Op Codes B Cy	12.50.01

25. Take another B-cycle and RO B-field character.

Set B Cy Ctrl	Edit Skid Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.17.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

26. Regenerate modify controls.

Regen 2nd Scan Ctrl	M Q Latch	17.13.15
2nd Scan Ctrl	Next to Last LG, Regen 2nd Scan Ctrl 2nd Scan	12.30.04
2nd Scan	2nd Scan Ctrl, LGC	12.30.02
Addr Mod Set to +1	2nd Scan	14.71.41

27. Set assembly controls to write blank if character is a comma or 0.

Extension Ctrl Lat	Edit Skid Cy, Next to Last LG	16.30.06
Extension Latch	Extension Ctrl Latc, LGC	16.30.06
Not * Fill or Fl Dol Ctrl	Last Insn RO Cy, LLG	17.12.02
Not * Fill or Fl Dol	Not * Fill or Fl Dol Ctrl	17.12.02

Signal	Control	Logic
Not * Fill or Fl Dol Ctrl	(A Cy, Not * Fill or Fl Dol) (Z Op B Cy, Not * Fill or Fl Dol)	17.12.02
Not * Fill	Float Dollar Sign, or Not * Fill or Fl Dol	17.13.04
Blanked 0 or Comma	E or Z, 2nd Scan Extension 0 Suppress, Not Decimal Ctrl Blank 0 or Comma, Not * Fill	17.13.04
Write Edit Blank	Blanked 0 or Comma	17.13.05

If the character is a significant digit the assembly controls are set to use the B-channel character.

Stor B Ch Char	2nd Scan, Z Op B Cy, Not Comma Not Blank, Not 0	17.13.03

When a significant digit is read out, zero suppress control is turned off to prevent any more zeros from being blanked.

Steps 25 through 27 are repeated until a B-channel WM is read out.

28. Initiate Insn RO.

Last Execute Cy	2nd Scan, Extension Latch, E or Z Op Code, Not * Fill or Fl Dol Not Decimal Ctrl, B Ch WM	12.12.51

Questions on Move Characters and Suppress Zeros Operation

Answers to review questions are in the Appendix.

1. What operations are accomplished in the successful execution of a move characters and suppress zeros instruction?

2. When are A-field zones not moved to the B-field?

3. Why is a skid cycle executed?

4. When is the zero suppress latch set during the second scan?

5. When does first scan end and the second scan begin?

6. If a move characters and suppress zeros instruction designates an A-field containing 0̆.0061 and a B-field containing 1̆.169, what characters do the A- and B-field contain at the end of the operation?

Edit Instruction

The edit instruction for the IBM 1410 Data Processing System causes all desired commas, decimals, dollar signs, asterisks, credit symbols, and minus signs to be automatically inserted in the designated numeric output field. The edit operation also eliminates insignificant zeros and commas in the output field. Thus, editing in the 1410 is the automatic control of zero suppression, inserting of identifying symbols, and punctuation of an output field.

To perform the edit operation, two fields are required: the data field and the control field. The data field contains data to be edited for output. The control field establishes the conditions that control the edit operation.

The control field is divided into two parts: the body (used for punctuating the data field) and the status portion (contains the special characters). The body of the control word begins with the rightmost 0 or blank and ends when the data field word mark is sensed. Other characters in the control (before the low-order 0 or blank and after the data field word mark is detected) make up the status portion of the word.

An edit operation requires two instructions. The move data instruction is used to transfer the control word (with word mark) to the output area; the edit instruction moves data to the output area and performs the editing function.

Instruction Formats

Formats for the move characters and edit instruction are:

Op code	A-address	B-address
Ě	xxxxx	xxxxx
Ě	xxxxx	
Ě		

The data field (A-field) is modified by the contents of the edit control field (B-field), and the results are stored in the B-field. A- and B-addresses in the instruction designate memory locations of low-order characters in the respective fields. When the instruction designates only the A-address, the contents of the B-address register from the previous operation specify the low-order character in the B-field to be used in the edit operation. When the instruction does not specify an A- or B-address, the contents of the A-address register and the B-address register from the previous operation specify the low order A- and B-field characters.

Word Marks

Word marks must be set in the high-order A- and B-field positions to define the end of the fields. When the A-field word mark is sensed, the remaining commas in the B-field are set to blanks. No A-field characters are transferred to the B-field after the B-field word mark is sensed. The data field can contain fewer, but should not contain more, positions than the number of blanks and zeros in the body of the control word.

Editing Specifications

All numeric, alphabetic, and special characters can be used in the control word. However, some characters have special meanings:

Control Character	Function
b (blank)	Replaced with the character from the corresponding position of the A-field.
0 (zero)	Used for zero suppression. Replaced with a corresponding character from the A-field. The rightmost 0 in the control word indicates the rightmost limit of zero suppression.
. (decimal)	Remains in the edited field in the position where written, unless decimal control was not in effect, and the data field did not contain a significant digit.
, (comma)	Undisturbed in the output data field in the position where written, unless zero suppression takes place and no significant numeric character is found to the left of the comma.
CR (credit)	Body portion: Undisturbed in the position where written. Status portion: If sign of the data field is positive, the positions containing CR are replaced with blanks. If the sign of the data field is negative, positions containing CR are undisturbed in the output field.
– (minus)	Same as CR (credit).
& (ampersand)	Causes a blank space in the output field; can be used in multiples.
* (asterisk)	Status portion: Undisturbed in the position where written. Body portion: Replaced by an A-field character. If 0 preceded the asterisk, asterisk fill is requested.
$ (dollar)	Status portion: Undisturbed in the position where written. Body portion: Replaced by an A-field character. If 0 preceded the dollar sign, floating dollar operation is requested.

Zero Suppression

Zero suppression is the deletion of unwanted zeros and commas to the left of significant digits in an output field. The following example of zero suppression in editing illustrates this operation:

Example:

A-field	0010900
Control word (B-field)	$bb,bb0.bb
Forward scan	$00,109.00
Reverse scan	$bbb109.00
Results of edit	$ 109.00

A special 0 is placed (in the body of the control word) in the rightmost limit of 0 suppression.

FORWARD SCAN

1. The positions in the output field at the right of this special 0 are replaced by the corresponding digits from the A-field.
2. When the special 0 is detected in the control field, it is replaced by the corresponding digit from the A-field.
3. A word mark is automatically set in this position of the B- (output) field.
4. The scan continues until the B-field (high-order) word mark is sensed and removed.

REVERSE SCAN

1. All zeros and punctuation at the left of the first significant character (up to and including the zero suppression code position) are replaced with blanks in the output field.
2. When the automatically set zero suppression word mark is sensed, it is erased and the operation ends.

Asterisk Protection

When it is necessary to have asterisks appear at the left of significant digits, the asterisk protection feature is used. The control word is written with the asterisk in the body to the left of the zero suppression code (if the asterisk appears in the body to the right of the zero suppression code, it is treated as a blank). The following example illustrates the asterisk protection feature:

Example:

A-field	00257426
Control word (B-field)	bbb,b*0.bb&CR
Forward scan	002,574.26 CR
Reverse scan	**2,574.26 CR
Results of edit	**2,574.26 CR

FORWARD SCAN

1. The normal editing process proceeds until the asterisk is sensed.
2. The asterisk is replaced (in the output field) by the corresponding digit from the A-field.
3. The editing process continues normally until the B-field word mark is sensed and removed.

REVERSE SCAN

1. Zeros, blanks, and punctuation to the left of the first significant digit are replaced by asterisks.
2. The word mark (set during the forward scan) signals the end of the edit operation; the word mark is removed.

Note: Asterisk protection and floating dollar sign cannot be used in the same control word.

Floating Dollar Sign

The floating dollar sign feature causes the insertion of a $ in the position to the left of the first significant digit in an amount. The control word is written with the $ in the body to the left of the zero suppression code (if the $ appears in the body to the right of the zero suppression code, it is treated as a blank). Three scans are necessary to perform the edit operation when the floating dollar sign feature is employed. The following example illustrates the operation of the floating dollar sign feature:

Example:

A-field	00257426
Control word (B-field)	bbbb,b$0.bb
First forward scan	b002,574.26
Reverse scan	bbb2,574.26
Second forward scan	$2,574.26
Results of edit	$2,574.26

FIRST FORWARD SCAN

1. The normal editing process proceeds until the $ is sensed.
2. The $ is replaced (in the output field) by the corresponding digit from the A-field.
3. Editing continues until the B-field work mark is sensed and removed.

REVERSE SCAN

1. Zeros and punctuation to the left of the first significant digit are replaced by blanks.
2. The reverse scan continues until the word mark (set during the first forward scan) signals the start of the second forward scan; the word mark is erased.

SECOND FORWARD SCAN

The second forward scan continues until the first blank position is sensed. This blank position is replaced by $, and the operation ends.

Note: Floating dollar sign cannot be used at the right of the decimal point, or when asterisk protection is used in the control word.

Sign Control Left

CR (credit) or – (minus) symbols can be placed at the left of a negative field. The control word is written with the CR or – symbol in the high-order status position. The following example illustrates sign control left operation:

Example:

A-field	00378940
Control word (B-field)	CR&bbb,bb0.bb
Forward scan	CRb003,789.40
Reverse scan	CRbbb3,789.40
Results of edit	CR 3,789.40

FORWARD SCAN

1. The scan proceeds until the zero suppression code 0 in the control field is sensed.
2. The corresponding character from the A-field is placed in this position of the output field.
3. A word mark is automatically inserted in this position in the output field.
4. The scan proceeds until the B-field word mark is sensed, indicating the end of the body of the control word.
5. The CR (credit) or – (minus) symbols are undisturbed in their corresponding positions in the output field if the sign of the A-field is minus. If the sign of the A-field is plus, the CR or minus symbol is blanked.

REVERSE SCAN

1. Zeros and punctuation are replaced with blanks in the output field. The scan continues until the automatically set word mark is sensed.
2. The automatically set word mark is erased and the operation ends.

Decimal Control

The decimal control feature insures that decimal points print only when there are significant digits in the A-field. The control word is written with a decimal point in the body to the left of the zero suppression code 0. Two scans are sufficient to complete the edit operation using the decimal control feature unless the field contains no significant digits; in this case, three scans are required. The following examples illustrate decimal control operation:

Examples:

1.	A-field	00000
	Control word (B-field)	bbb.b0
	First forward scan	000.00
	Reverse scan	bbb.00
	Second forward scan	bbbbbb
	Results of edit	(blank field)
2.	A-field	29437
	Control word (B-field)	bbb.b0
	Forward scan	294.37
	Reverse scan	294.37
	Results of edit	294.37

FIRST FORWARD SCAN

1. When the zero suppression code 0 is sensed during editing, this position is replaced by the corresponding digit from the A-field.
2. A word mark is set automatically in this position in the B- (output) field.
3. Editing continues normally until the B-field word mark is sensed and erased.

REVERSE SCAN

1. Zeros and punctuation are replaced with blanks in the output field until the decimal point is sensed.
2. The decimal point and the digits at the right are not changed. The automatically set word mark is erased. If there are no significant digits in the field, the second forward scan is initiated; otherwise, the edit operation stops.

SECOND FORWARD SCAN

1. The zeros at the right of the decimal point, and the decimal point itself, are replaced with blanks.
2. The operation stops at the decimal column.

CPU Operation

During last instruction read-out cycle, the first scan and units latches are set, and controls are conditioned to execute a standard A-cycle first. In the first A-cycle, the character stored in the units position of the A-field reads out of memory and is gated into the A-data register. Because this low-order A-field character contains the sign of the A-field, the plus or minus latch is set.

After the A-cycle, the CPU executes a B-cycle to read out the low-order B-field character. If the units position B-field character is not a blank, 0, &, CR, or −, or if the character is CR, or − and the minus latch is set, the character is returned unchanged to the low-order B-field position. If the character is &, a blank is returned to the B-field. Because the B-field character is neither a 0 nor a blank, the character is recognized as part of the status portion of the control word (B-field); another B-cycle is required to read out the next B-field character. If the next character is not a 0 or blank, indicating that it, too, is part of the status portion of the control word, the same conditions determine whether the character or a blank is returned to the B-field. The CPU executes successive B-cycles to read out and return characters to the B-field until the first 0 or blank is sensed.

The first B-field 0 or blank indicates the end of the status portion and the beginning of the body of the control word (a 0 or blank may be sensed on the first B-cycle). The first B-field 0 or blank resets the units latch, sets the body latch, and stores the numerics in the A-field character (set in the A-data register during the first A-cycle) in the B-field. B-field characters read while the body latch is set are in the body of the control word.

When the first B-field 0 is sensed, the zero suppress latch is set, and a word mark is automatically set over the A-field character returned to the B-field during that B-cycle.

With the first scan and body latches set, the CPU:

1. Returns the B-channel character to the B-field if the character is not 0, blank, *, $, or &.

2. Writes a blank in the B-field if the B-channel character is an & sign.

3. Stores the A-channel character if the B-channel character is a blank or 0.

4. Sets the * fill or floating dollar latch if the B-channel character is an * or $ and the zero suppress latch is set. The A-channel character is stored in the B-field.

A word mark is gated to the B-field only when the low-order 0 in the control word is sensed.

If the B-channel character is returned to the B-field on a B-cycle, the CPU executes another B-cycle to read out the next B-field character immediately. If the A-channel character is stored in the B-field on a B-cycle, the CPU executes another A-cycle, then another B-cycle to read the next characters in sequence from the A- and B-fields, respectively. A B-cycle always follows an A-cycle. However, an A-field character stored in the A-data register on an A-cycle might remain in the register until the CPU executes several B-cycles.

If the A-channel word mark, indicating the end of the A-field, is sensed before the B-channel word mark is detected, the body latch is reset, and the extension latch is set. The CPU executes successive B-cycles to read out the remaining characters in the control word (B-field). Either the same characters read or blanks are returned to the B-field. B-channel characters are read while the first scan and extension latches are set in the status portion of the control word. The B-channel word mark indicating the end of the B-field terminates the first scan. Other A-field characters are not processed after the B-channel word mark is detected. For this reason, the A-field should not contain more characters than the number of blanks and zeros in the body of the control word.

If the zero suppress latch is not set when the B-field word mark is sensed, the first scan and the edit operation end (no zeros in the control word). If the zero suppress latch is set when the B-field word mark is detected, the second scan (reverse scan) is initiated.

The CPU executes B-cycles exclusively during the second scan. Because the BAR was modified by −1 on the last B-cycle of the first scan, the BAR addresses the low-order character in the next field when the second scan begins. The CPU performs a skid B-cycle to read out and return the character to its storage position unchanged. The ON states of the second scan and MQ latches, set at the end of the first scan, identify the first B-cycle of the second scan as the skid cycle. The MQ latch is not regenerated when the skid cycle is complete. The extension latch is set during the skid cycle.

In the second scan, all insignificant zeros and commas (zeros and commas to the left of significant digits) are set to blanks or, if the asterisk fill latch is set, replaced with asterisks. The first significant digit (1-9) encountered in the second scan resets the zero suppress latch. If the zero suppress latch is not reset when a decimal is sensed, the decimal control latch is set, canceling the blanking effect of the zero suppress latch. When either the zero suppress latch is reset or the decimal control latch is set, zeros and commas in the B-field are not replaced with blanks or asterisks; for example, a B-field containing 000.01 at the end of the first scan becomes bbb.01 during the second scan rather than bbb.b1. If a character that is not a significant digit (1-9), blank, comma, 0, minus sign or deci-

mal is encountered after the zero suppress latch is reset and before the decimal control latch is set, the zero suppress latch is set again. Zeros and commas sensed before the next significant digit are replaced with * or blanks; for example, a B-field containing 000100bCARSbb00200,000.75 after the first scan becomes 100 CARS 200,000.75 during the second scan if the asterisk fill latch is off.

Recall that in the first scan, a word mark was set over the B-field location containing the low-order 0 in the control word. If the decimal control and zero suppress latches are set when the word mark is sensed and the character read out with the word mark is not a significant digit (1-9), a third scan is started; for example, if the B-field contains 000.00 at the end of the first scan and bbb.00 at the end of the second scan, a third scan is initiated when the B-cycle, during which the 0 is processed, is complete. Remember that the decimal control latch does not reset the zero suppress latch, but rather, it cancels the zero blanking effect of the zero suppress latch.

The edit operation is terminated at the end of the second scan if the floating dollar latch is off and either:

1. The decimal control or zero suppress latches are off when the B-field word mark is sensed, or

2. The character read out of storage with the word mark is a significant digit. If the floating dollar latch was set during first scan, a third scan is required regardless of conditions established during second scan.

The CPU executes B-cycles exclusively during the third scan (forward scan). Because the BAR was modified by +1 on the last B-cycle of the second scan, the BAR addresses the character to the right of the desired position when the third scan begins. The CPU performs a skid B-cycle to read out and return the character to its storage position unchanged. The ON states of the third scan and MQ latches, set at the end of the second scan, identify the first B-cycle of the third scan as the skid cycle. The MQ latch is not regenerated when the skid cycle is complete. The extension latch is set during the skid cycle.

If the zero suppress latch is set, zeros processed during the third scan are replaced with blanks or, if the asterisk fill latch is set, with asterisks.

If the decimal control and zero suppress latches are set when a decimal is sensed during third scan, the decimal is replaced with a blank or asterisk, and the edit operation ends; for example, if the B-field contained 000.00 after the first scan, and bbb.00 after the second scan, the field is blanked (bbbbbb) at the end of the third scan. The edit operation is complete when the decimal is read out and replaced with a blank.

If the decimal control latch is set and the zero suppress latch is reset when a decimal is detected during third scan, the decimal is returned to the B-field, and the edit operation ends.

If the floating dollar latch is set, a $ replaces the first B-field blank detected on third scan and terminates the edit operation; for example, if the floating dollar latch was set during first scan and the B-field contained 100bCARSbbbb200,000.75 at the end of the second scan, the B-field contains 100bCARSbbb$200,000.75 at the end of the third scan. The operation ends when the $ is stored in the B-field.

If both the floating dollar and decimal control latches are set, the third scan and the edit operation are complete when either a decimal or blank is read out of memory and the appropriate character is returned to the B-field.

Figure 23 is a step-by-step editing process of a selected example. Figures 24 and 25 show diagrammed explanations of CPU actions in the execution of the move characters and edit instruction.

The following controls are active in the move characters and edit operation:

SIGNAL	CONTROL	LOGIC
1. Initiate A-cycle and RO first A-field character.		
Set A Cy Ctrl	A Cy First Op Codes Last Insn RO Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
2. Set modifier controls to −1.		
Set 1st Scan Ctrl	1st Scan First Op Code Last Insn RO Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl, Next to Last LG	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
3. Set sign latch.		
Units Ctrl Latch	Last Insn RO Cy, Next to Last LG	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02
Plus/Minus Sign Latch	A Cy, Units Latch, E or Z Op Code, B Ch Plus/ Minus, Last LG	16.16.04
4. Set character into A-data register.		
Sw B Ch to A Reg	A Cy, LGD	15.38.01
5. Control A-cycle length.		
Std A Cycle Ops, A Cy	E or Z Op Code, A Cy	13.14.06
Stop at F	Std A Cy Ops, A Cy	12.12.30
6. Initiate B-cycle and RO first B-field character.		
Set B Cy Ctrl	Std A Cy Ops, A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
7. Regenerate modify controls.		

	Op Code	A-address	B-address
Edit Instruction	E	12163	04685
Storage		A-field (data field) 00257426	B-field (control word) $bbb,bb0.bb&CR&**
Result of Edit Op		A-field 00257426	B-field $ 2,574.26 **

STEP	TYPE OF CYCLE	ADDRESS REGISTERS			DATA REGISTER		PUT BACK INTO STORAGE	B-FIELD AT END OF CYCLE	REMARKS
		I	A	B	B	A			
1	Iop	00002	?????	?????	E	E	E	$bbb,bb0.bb&CR&**	Read Instruction Op Code
2	I1	00003	1????	?????	1	1	1	Same	Load A-address register
3	I2	00004	12???	?????	2	2	2	Same	Load A-address register
4	I3	00005	121??	?????	1	1	1	Same	Load A-address register
5	I4	00006	1216?	?????	6	6	6	Same	Load A-address register
6	I5	00007	12163	?????	3	3	3	Same	Load A-address register
7	I6	00008	12163	0????	0	0	0	Same	Load B-address register
8	I7	00009	12163	04???	4	4	4	Same	Load B-address register
9	I8	00010	12163	046??	6	6	6	Same	Load B-address register
10	I9	00011	12163	0468?	8	8	8	Same	Load B-address register
11	I10	00012	12163	04685	5	5	5	Same	Load B-address register
12	I11	00012	12163	04685	Op	Op	Op	Same	Op Code & next instruction
13	A	00012	12162	04685	6	6	6	Same	Execute EDIT instruction
14	B	00012	12162	04684	*	6	*	Same	
15	B	00012	12162	04683	*	6	*	Same	
16	B	00012	12162	04682	&	6	Blank	$bbb,bb0.bb&CRb**	
17	B	00012	12162	04681	R	6	Blank	$bbb,bb0.bb&Cbb**	
18	B	00012	12162	04680	C	6	Blank	$bbb,bb0.bb&bbb**	
19	B	00012	12162	04679	&	6	Blank	$bbb,bb0.bbbbbb**	
20	B	00012	12162	04678	b	6	6	$bbb,bb0.b6bbbb**	
21	A	00012	12161	04678	2	2	2	Same	
22	B	00012	12161	04677	b	2	2	$bbb,bb0.26bbbb**	
23	A	00012	12160	04677	4	4	4	Same	

Figure 23A. Step-By-Step Editing Process

STEP	TYPE OF CYCLE	ADDRESS REGISTERS			DATA REGISTER		PUT BACK INTO STORAGE	B-FIELD AT END OF CYCLE	REMARKS
		I	A	B	B	A			
24	B	00012	12160	04676	•	4	•	Same	
25	B	00012	12160	04675	0	4	4	$\overset{v}{\$}bbb,bb\overset{v}{4}$.26bbbb**	Zero Suppress
26	A	00012	12159	04675	7	7	7	Same	
27	B	00012	12159	04674	b	7	7	$\overset{v}{\$}bbb,b\overset{v}{7}$4.26bbbb**	
28	A	00012	12158	04674	5	5	5	Same	
29	B	00012	12158	04673	b	5	5	$\overset{v}{\$}$bbb,$\overset{v}{5}$74.26bbbb**	
30	A	00012	12157	04673	2	2	2	Same	
31	B	00012	12157	04672	,	2	,	Same	
32	B	00012	12157	04671	b	2	2	$\overset{v}{\$}$bb2,$\overset{v}{5}$74.26bbbb**	
33	A	00012	12156	04671	0	0	0	Same	
34	B	00012	12156	04670	b	0	0	$\overset{v}{\$}$b02,$\overset{v}{5}$74.26bbbb**	
35	A	00012	12155	04670	$\overset{v}{0}$	$\overset{v}{0}$	$\overset{v}{0}$	Same	
36	B	00012	12155	04669	b	$\overset{v}{0}$	0	$\overset{v}{\$}$002,$\overset{v}{5}$74.26bbbb**	
37	B	00012	12155	04668	$\overset{v}{\$}$	$\overset{v}{0}$	$	$002,$\overset{v}{5}$74.26bbbb**	Sense Word Mark — Rev. Scan
38	Skid B	00012	12155	04669	?	$\overset{v}{0}$	?	$002,$\overset{v}{5}$74.26bbbb**	Units Position of next Field
39	B	00012	12155	04670	$	$\overset{v}{0}$	$	Same	
40	B	00012	12155	04671	0	$\overset{v}{0}$	Blank	$b02,$\overset{v}{5}$74.26bbbb**	
41	B	00012	12155	04672	0	$\overset{v}{0}$	Blank	$bb2,$\overset{v}{5}$74.26bbbb**	
42	B	00012	12155	04673	2	$\overset{v}{0}$	2	Same	
43	B	00012	12155	04674	,	$\overset{v}{0}$	,	Same	
44	B	00012	12155	04675	5	$\overset{v}{0}$	5	Same	
45	B	00012	12155	04676	7	$\overset{v}{0}$	7	Same	
46	B	00012	12155	04677	$\overset{v}{4}$	$\overset{v}{0}$	4	$bb2,574.26bbbb**	

Figure 23B. Step-By-Step Editing Process

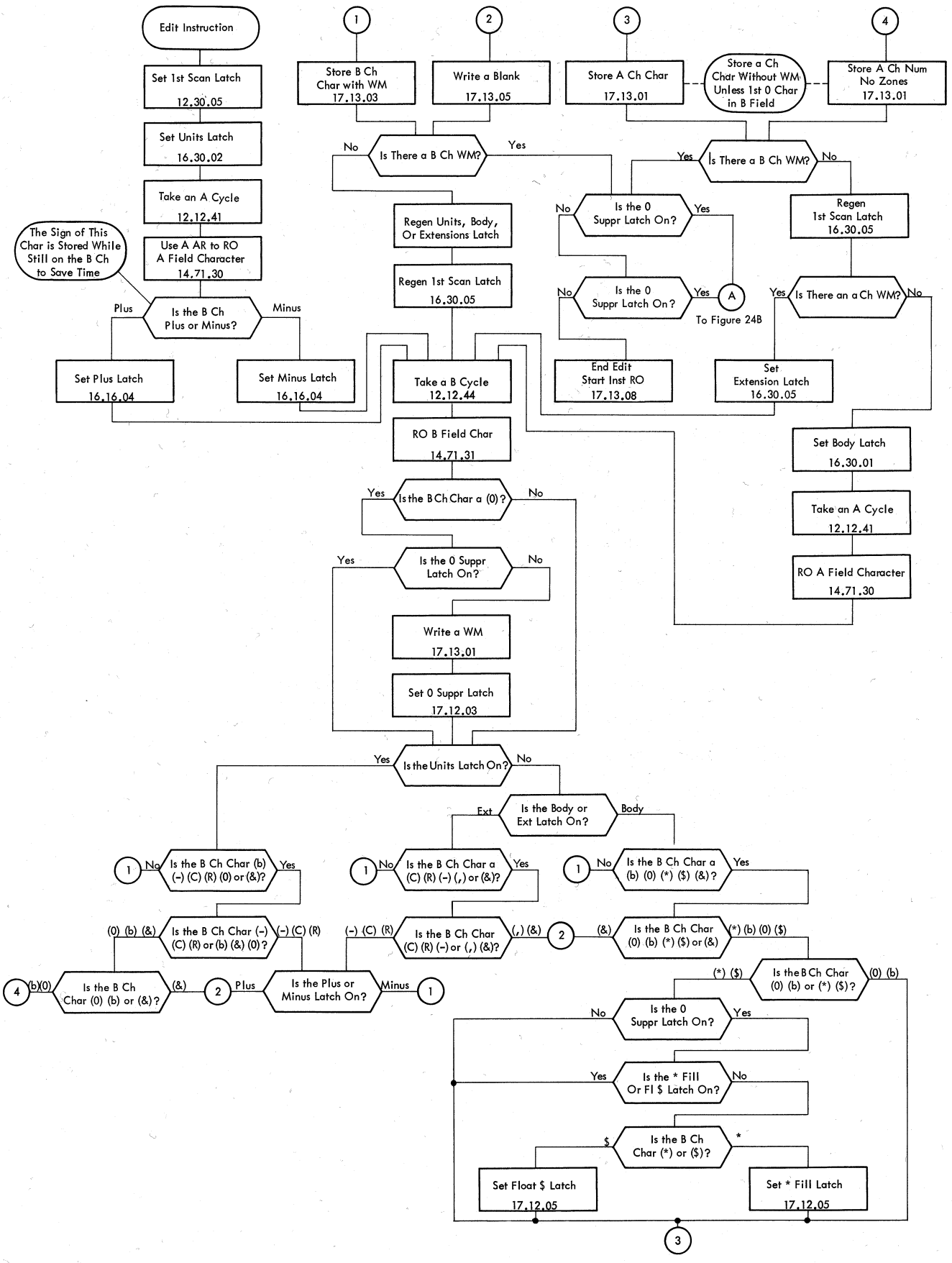
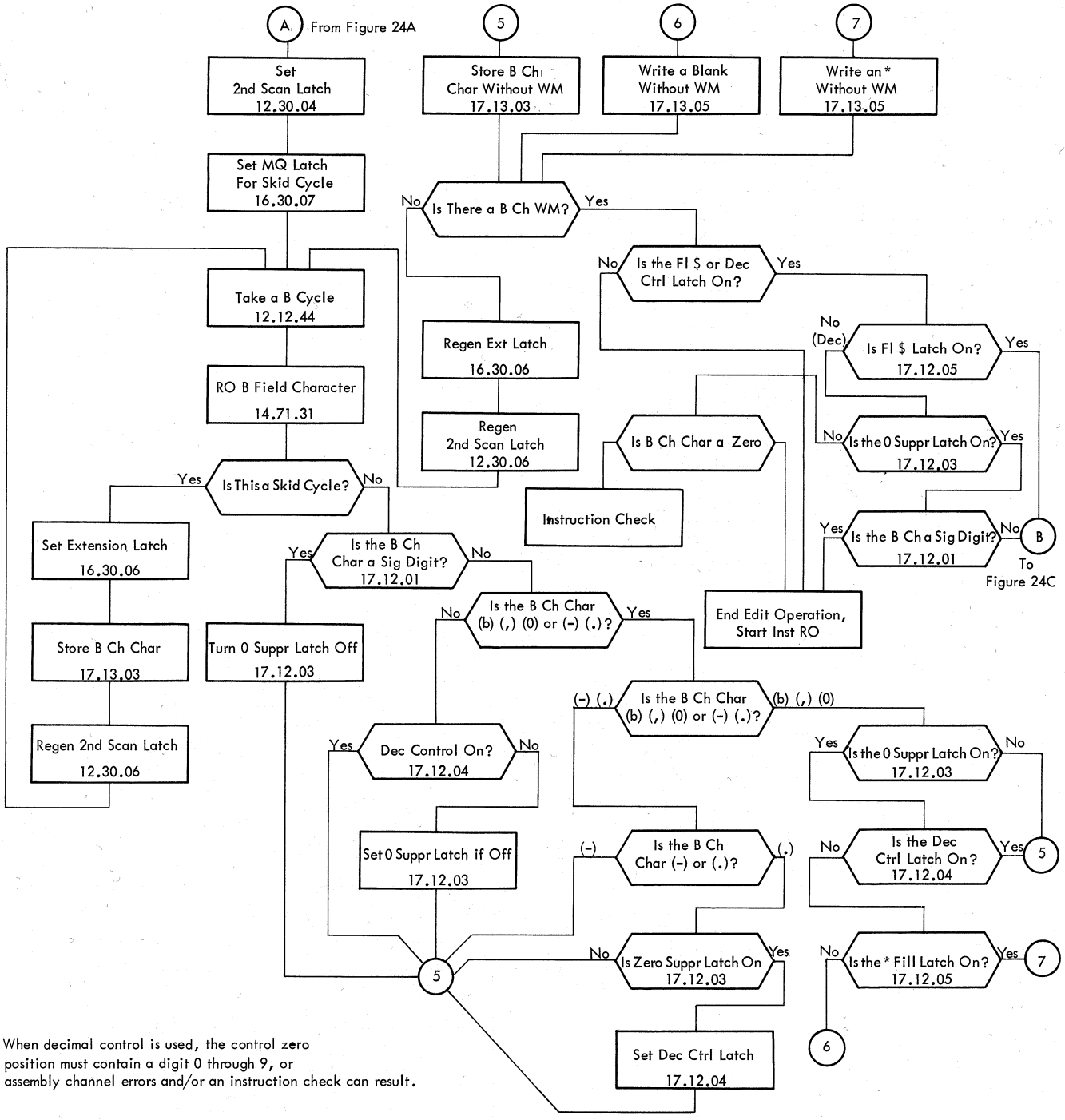
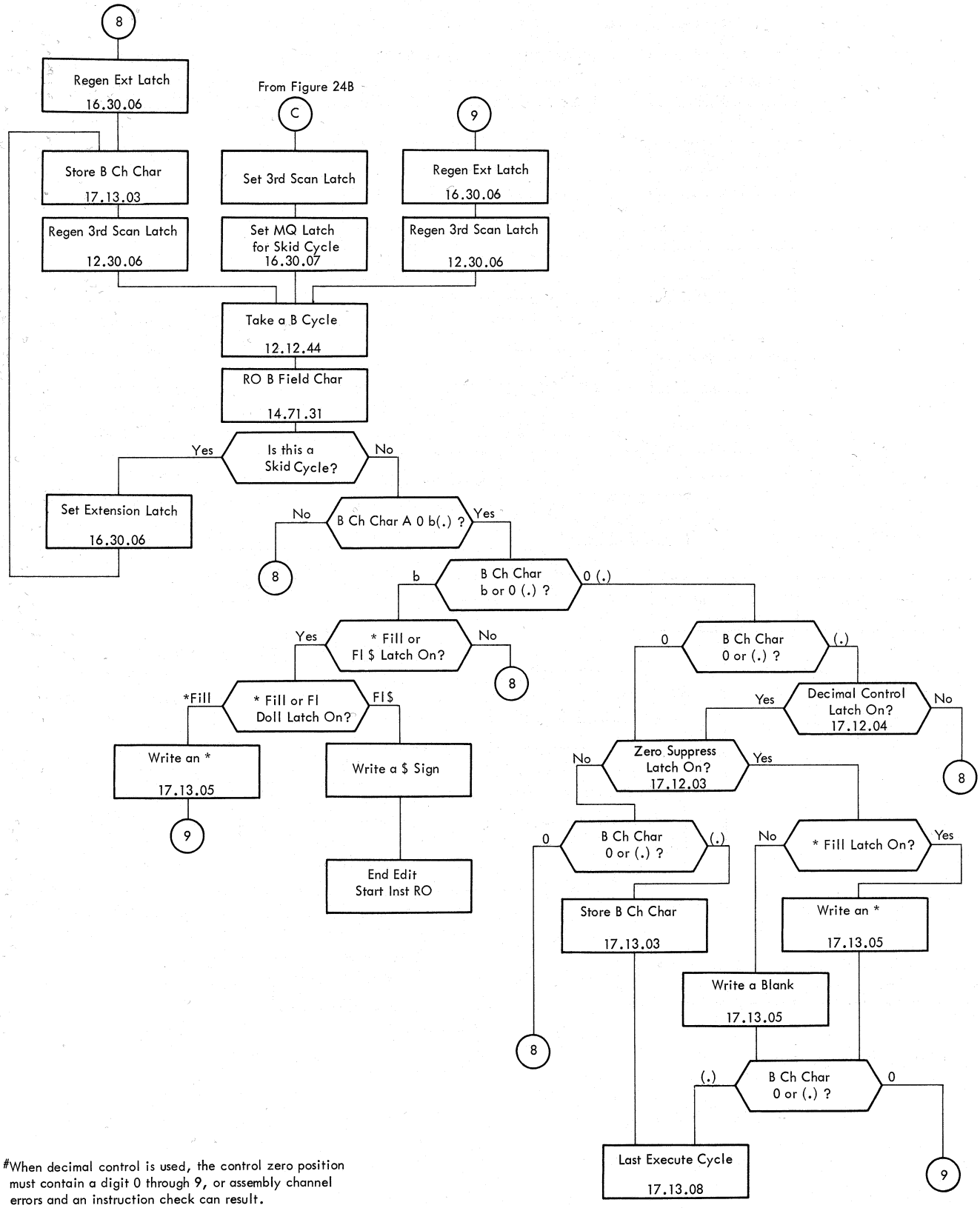


Figure 24A. Edit, First Scan



When decimal control is used, the control zero position must contain a digit 0 through 9, or assembly channel errors and/or an instruction check can result.

Figure 24B. Edit, Second Scan



#When decimal control is used, the control zero position must contain a digit 0 through 9, or assembly channel errors and an instruction check can result.

Figure 24C. Edit, Third Scan

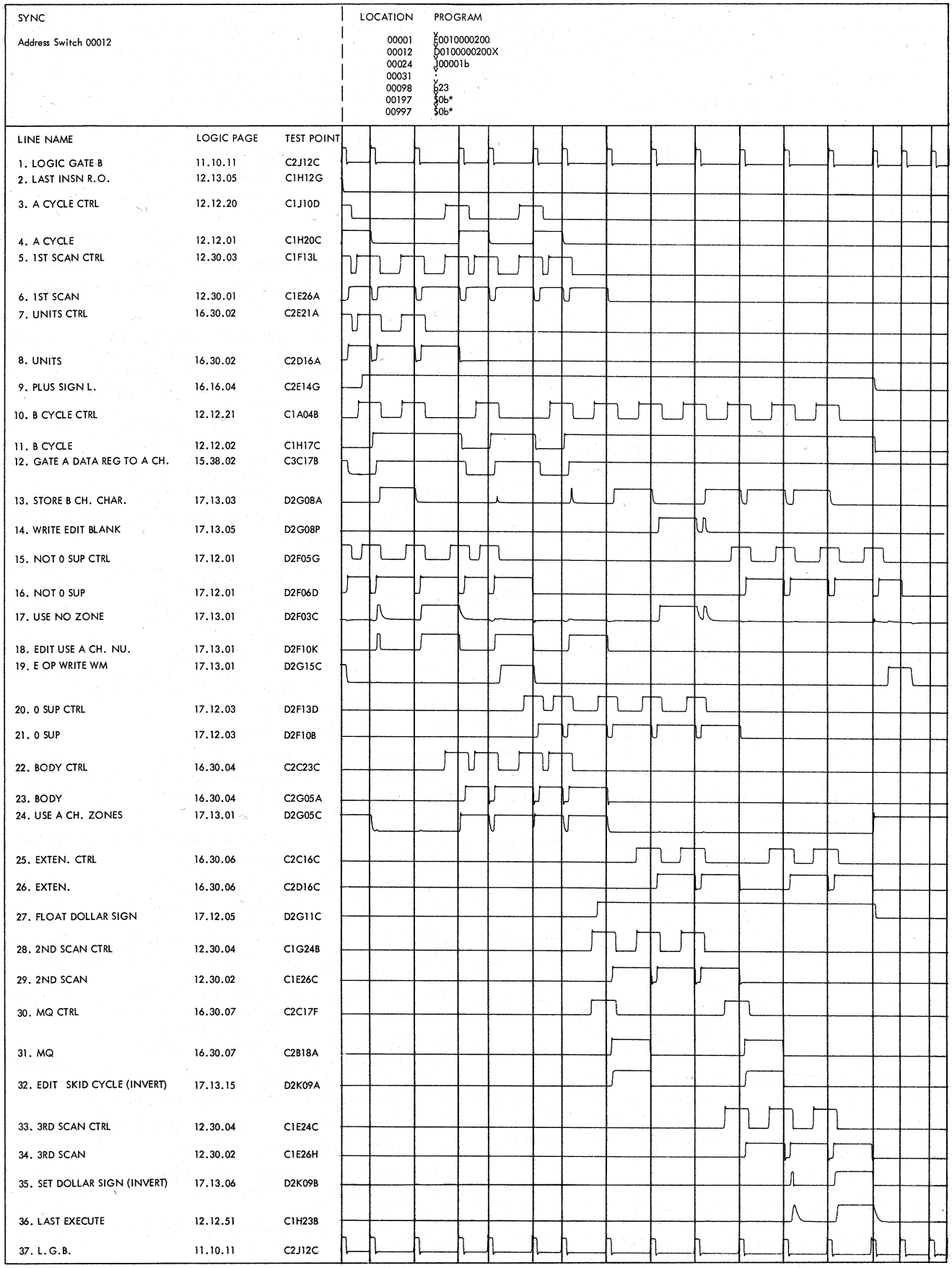


Figure 25. Edit Operation Timings

Signal	Control	Logic
Regen 1st Scan Ctrl	Std A Cy Ops, A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

8. Gate A-field character to A-channel.

Gate A Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02

9. Set assembly controls according to B-channel character.

Regen Units + Body Ctrl	Std A Cy Ops A Cy	16.30.01
Units Ctrl Latch	Regen Units + Body Ctrl, Next to Last LG Units Latch	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02

If B-channel character is not a blank, minus sign, C, R, &, or 0; store the B-channel character.

Credit or Not U Ctrl Char	Units Latch, E Op • B Cy Not Blank, Not Ctrl 0, Not C Char, Not R Char, Not Minus Symbol, Not Space	17.13.02
Store B Ch Character	Credit or Not U Ctrl Char	17.13.03

If B-channel character is a C, R, or Minus sign, the sign latch determines what is stored.

Credit or Not U Ctrl Char	Units Latch, E Op • B Cy, 1st Scan, C or R or Minus, Minus Sign Latch	17.13.02
or Store B Ch Char	Credit or Not U Ctrl Char	17.13.03
or Blanked Credit Symbol	C or R or Minus, Plus Sign Latch, 1st Scan, E Op • B Cy, Units Latch	17.13.02
Write Edit Blank	Blanked Credit Symbol	17.13.05

If the B-channel character is &, it means space and a blank is stored.

Space	E Op B Cy, 1st Scan, B Ch (B, A, 8, 1, 2, 4)	17.11.07
Write Edit Blank	Space	17.13.05

10. Any of the conditions in step 9 cause another B-cycle with the Units latch regenerated to identify the status portion of the control word.

Write B Char or Spec Char	Store B Ch Character, or Write Edit Blank	17.13.07
Edit Set B Cy Ctrl C	Write B Char or Spec Char 1st or 2nd Scan, B Ch Not WM Bit	17.13.09
Set B Cy Ctrl	Edit Set B Cy Ctrl C	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl, RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

11. Regenerate modify controls.

Regen 1st Scan Ctrl	Edit Set B Cy Ctrl C	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

12. Regenerate units latch.

Regen Units + Body Ctrl	Edit Set B Cy Ctrl C	16.30.01
Units Ctrl Latch	Regen Units + Body Ctrl, Next to Last Logic Gate, Units Latch	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02

13. Set assembly controls according to B-channel character; this character can be the same as in step 9 in which case steps 9-12 are repeated.

If the B-channel character is 0 or blank the character on the A-channel (from the last A-cycle) is stored.

Edit Use A Ch Nu	E Op • B Cy, Units Latch, Blank or 0	17.13.01
Use No Zones * Edit	E Op • B Cy, Units Latch, Blank or 0	17.13.01

14. If the character is a 0, a word mark is stored over the A-channel character to identify the end of zero suppress on the second scan; also, the zero suppress latch is set on.

Not Zero Suppress Ctrl	Last Insn RO Cy, Last LG	17.12.01
Not Zero Suppress	Not Zero Suppress Ctrl, LGC	17.12.01
Not Zero Suppress Ctrl	Not Zero Suppress, Last LG, A Cy/1st Scan, E Op • B Cy, Not Ctrl 0	17.12.01
E Op Write WM	E Op • B Cy, 1st Scan, Ctrl Zero Not 0 Suppress	17.13.01
0 Suppress Ctrl	E Op Write WM, E or Z Op • B Cy, Last LG	17.12.03
0 Suppress	0 Suppress Ctrl, LGC	17.12.03

15. After the A-channel character is stored another A-cycle must be taken.

Edit Set A Cy Ctrl	Edit Use A Ch Nu, A Ch WM, B Ch WM	12.12.41
Set A Cy Ctrl	Edit Set A Cy Ctrl	12.12.41
A Cy Ctrl	Set A Cy Ctrl, Next to Last LG	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
RO AAR	A Cy Ctrl, LG Special A Read out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16

16. Regenerate modify control.

Regen 1st Scan Ctrl	Edit Set A Cy Ctrl	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

17. Set assembly controls according to B-channel character in body of control word.

Set Body Ctrl Latch	Edit Set A Cy Ctrl	16.30.01
Body Ctrl Latch	Set Body Ctrl Latch, Next to Last LG	16.30.04
Body Latch	Body Ctrl Latch, LGC	16.30.04

If B-channel character is not a blank, 0, *, $, or &, store the B-channel character.

Store B Ch Char	Body Latch, E Op • B Cy, Not Blank, Not Ctrl 0, Not *, Not $, Not Space	17.13.03

If B-channel character is & write a blank.

Write Edit Blank	Space	17.13.05

Again, either of these conditions cause another B-cycle as in steps 10-12 except that the body latch is regenerated.

18. If the B-channel character is a 0, blank, *, or $ the A-channel character is stored.

Edit Use A Ch Nu and Use A Ch Zones * Edit	E Op • B Cy, Body Latch, Blank or 0	17.13.01
Edit Use A Ch Nu and Use A Ch Zones * Edit	E Op • B Cy, Body Latch, * or $	17.13.01

A 0 sets the zero suppress latch if it had not been set before.

Signal	Control	Logic

19. An * or $ sets the asterisk fill or float dollar latch if the zero suppress latch had been previously set.

Not * Fill or Fl Dol Ctrl	Last Insn RO Cy, LLG	17.12.02
Not * Fill or Fl Dol	Not * Fill or Fl Dol Ctrl	17.12.02
Not * Fill or Fl Dol Ctrl	(A Cy, Not * Fill or Fl Dol) or E Op • B Cy, Body Latch, Not *, Not $	17.12.02 17.12.02
Asterisk Fill or Fl Dollar Sign	E Op • B Cy, Not * Fill or Fl Dol, 0 Suppress, Body Latch, and */$	17.12.05

20. Edit use A-channel Number again causes A-cycles unless the last A-field character has a word mark which causes a B-cycle.

Edit Set B Cy Ctrl A	Edit Use A Ch Nu, E Op Code A Ch WM Bit, B Ch Not WM Bit	12.12.44
Set B Cy Ctrl	Edit Set B Cy Ctrl A	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

21. Regenerate modify controls.

Regen 1st Scan Ctrl	Edit Set B Cy Ctrl A	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

22. Set assembly control according to B-channel character.

Set Extension Ctrl Latch	Edit Set B Cycle Ctrl A	16.30.05
Extension Ctrl Lat	Set Extension Ctrl Latch, Next to LLG	16.30.06
Extension Latch	Extension Ctrl Lat, LGC	16.30.06

If B-channel character is not a C, R, minus sign, comma or &, store the B-channel character.

Store B Ch Char	Extension Latch, E Op B Cy 1st Scan, Not C Char, Not R Char, Not Minus Symbol, Not Space, Not Comma	17.13.03

If B-channel character is an & or a comma, a blank is stored.

Write Edit Blank •	Space or E Op B Cy, Extension Latch, Comma	17.13.05

If the B-channel character is a C, R or minus sign, the sign latch determines what is stored.

Credit or Not U Ctrl Char	Extension Latch, E Op B Cy, 1st Scan, C or R or Minus, Minus Sign Latch	17.13.02
Store B Ch Character	Credit or Not U Ctrl Char	17.13.03
or Blanked Credit Symbol	C or R or Minus, Plus Sign Latch, 1st Scan, E Op B Cy, Extension Latch	17.13.02
Write Edit Blank	Blanked Credit Symbol	17.13.05

23. The scan continues through the status portion of the control word until a word mark; if the zero suppress latch is on the second scan is initiated otherwise the operation ends.

Edit Set B Cy Ctrl B	E Op B Cy 1st Scan, 0 Suppress, B Ch WM Bit	17.13.08
Set B Cy Ctrl	Edit Set B Cy Ctrl B	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

24. Set modify controls to +1.

Set 2nd Scan Ctrl	Edit Set B-Cy Ctrl B	12.30.06
2nd Scan Ctrl	Set 2nd Scan Ctrl, Next to LLG	12.30.04
2nd Scan	2nd Scan Ctrl, LGC	12.30.02
Addr Mod Set to +1	2nd Scan	14.71.41

25. Set skid cycle controls.

Set MQ Ctrl	Edit Set B Cy Ctrl B	16.30.05
MQ Ctrl Latch	Set MQ Ctrl, Next to Last LG	16.30.07
MQ Latch	MQ Ctrl Latch, LGC	16.30.07
Edit Skid Cy	MQ Latch, E or Z Op B Cy	17.13.15

26. Store B-channel character.

Use B-Ch WM	Edit Skid Cy	15.49.04
Store B-Ch Char	Edit Skid Cy	17.13.03
Load Memory	Load Memory on B Cy Op Codes B Cy	12.50.01

27. Take another B-cycle and read out B-field character.

Set B Cy Ctrl	Edit Skid Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Read Out BAR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops	14.17.31
Set Mem AR Gated	LGA, 2nd Clock Pulse	14.17.16

28. Regenerate modify controls.

Regen 2nd Scan Ctrl	MQ Latch	17.13.15
2nd Scan Ctrl	Next to Last LG, Regen 2nd Scan Ctrl 2nd Scan	12.30.04
2nd Scan	2nd Scan Ctrl, LGC	12.30.02
Addr Mod Set to +1	2nd Scan	14.71.41

29. Set assembly controls according to B-channel character.

Extension Ctrl Latch	Edit Skid Cy, Next to Last LG	16.30.06
Extension Latch	Extension Ctrl Latch, LGC	16.30.06

If B-channel character is any other character except 0, comma, or blank, the B-channel character is stored.

Store B Ch Char	E or Z Op B Cy, 2nd Scan, Not Ctrl 0, Not Blank, Not Comma	17.13.03

If B-channel character is a 0, blank, or comma, the character is either stored as is (zero suppress off), blanked (zero suppress on), or an * is stored (zero suppress and asterisk fill latches on). Zero suppress is the same as in Z op code.

Decimal Control

30. Set decimal controls when a decimal is sensed.

Decimal Ctrl	2nd Scan, Extension Latch, E or Z Op Code, Decimal, Last LG	17.13.03

31. After the decimal, store all B-channel characters.

2nd Scan Sig Char	E or Z Op • 2nd Scan • Extension, Blank 0 or Comma, Decimal Ctrl	17.13.04
Store B Ch Char	E or Z Op • B Cy, 2nd Scan, Not Blank, Not Comma, Not 0 or 2nd Scan Sig Char	17.13.03

Signal	Control	Logic
32. Take another B-cycle and read out B-field character if a third scan is required.		
3rd Scan Cond	Decimal Ctrl, 0 Suppress Not Sig Digit	17.13.07
Edit Set B Cy Ctrl D	3rd Scan Cond, E or Z Op Code 2nd Scan Extension, B Ch WM	17.13.09
Set B Cy Ctrl	Edit Set B Cy Ctrl D	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
33. Set modify controls to −1.		
Set 3rd Scan Ctrl	Edit Set B Cy Ctrl D	12.30.06
3rd Scan Ctrl	Set 3rd Scan Ctrl, Next to Last LG	12.30.04
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Addr Mod Set to −1	3rd Scan	14.71.41
34. Set skid cycle controls.		
Set M Q Ctrl	Edit Set B Cy Ctrl D	16.30.05
M Q Ctrl Latch	Set M Q Ctrl, Next to Last LG	16.30.07
M Q Latch	M Q Ctrl Latch, LGC	16.30.07
Edit Skid Cy	M Q Latch, Z Op B Cy	12.13.15
35. Store B-channel character.		
Use B Ch WM	Edit Skid Cy	15.49.04
Store B Ch Char	Edit Skid Cy	17.13.03
Load Memory	Load Mem on B Cy Op Codes, B Cy	12.50.01
36. Take another B-cycle and read out B-field character.		
Set B Cy Ctrl	Edit Skid Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
37. Regenerate modify controls.		
Regen 3rd Scan Ctrl	M Q Latch	17.13.15
3rd Scan Ctrl	Next to LLG, Regen 3rd Scan Ctrl, 3rd Scan	12.30.04
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Addr Mod Set to −1	3rd Scan	14.71.41
38. Set assembly controls to write blank if character is a decimal or 0.		
Extension Ctrl Latch	Edit Skid Cy, Next to Last LG	16.30.06
Extension Latch	Extensions Ctrl Latch, LGC	16.30.06
Not * Fill or Fl Dol Ctrl	Last Insn RO Cy, LG	17.12.02
Not * Fill or Fl Dol	(A Cy, Not * Fill or Fl Dol)	17.12.02
Not * Fill or Fl Dol Ctrl	(Z Op B Cy, Not * Fill or Fl Dol)	17.12.02
Write Edit Blank	Not * Fill, 0 Suppress E or Z, 2nd Scan, Extension, 0 or Decimal	17.13.05
Store B Ch Char	Not Decimal, Not 0, 3rd Scan	17.13.03
39. If the character is not a decimal, another B-cycle is taken.		
Write B Char or Spec Char	Write Edit Blank	17.13.07
Edit Set B Cy Ctrl F	Not Decimal, E or Z 3rd Scan ,Extension, Write B Char or Spec Char	17.13.09
Set B Cy Ctrl	Edit Set B Cy Ctrl F	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
40. Regenerate modify controls.		
Regen 3rd Scan Ctrl	Edit Set B Cy Ctrl F	12.30.06
3rd Scan Ctrl	Next to LLG, Regen 3rd Scan Ctrl 3rd Scan	12.30.04
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Addr Mod Set to −1	3rd Scan	14.71.41
41. Set assembly controls to write blank if character is a decimal or 0.		
Regen Ext Ctrl	Edit Set B Cy Ctrl F	16.30.05
Extension Ctrl Latch	Regen Ext Ctrl	16.30.06
Extension Latch	Extension Ctrl Latch, LGC	16.30.06
Not * Fill or Fl Dol Ctrl	Last Insn RO Cy, LG	17.12.02
Not * Fill or Fl Dol	Not * Fill or Fl Dol Ctrl	17.12.02
Not * Fill or Fl Dol Ctrl	(A Cy, Not * Fill or Fl Dol) (Z Op B Cy, Not * Fill or Fl Dol)	17.12.02
Write Edit Blank	Not * Fill, 0 Suppress E or Z, 3rd Scan, Extension 0 or Decimal	17.13.05
42. If the character is a decimal, the operation is ended.		
Last Execute Cy	E or Z • 3rd Scan Extension Decimal Ctrl, Decimal * Fill	17.13.08
43. Write * if B-channel character is 0, blank, or comma.		
Write Edit *	E or Z • 2nd Scan • Extension, Not Decimal Ctrl, 0 Suppress, Blank 0 or Comma, * Fill	17.13.05

The second scan continues until the word mark, set on the first scan, is read out and removed. Here the operation is either ended or a third scan is taken.

44. Take another B-cycle and read out B-field character if a third scan is required.		
3rd Scan Cond	Float Dollar Sign	17.13.07
Edit Set B Cy Ctrl D	3rd Scan Cond, E or Z Op Code 2nd Scan • Extension, B Ch WM	17.13.09
Set B Cy Ctrl	Edit Set B Cy Ctrl D	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
45. Set modify controls to −1.		
Set 3rd Scan Ctrl	Edit Set B Cy Ctrl D	12.30.06
3rd Scan Ctrl	Set 3rd Scan Ctrl, Next to Last LG	12.30.04

Signal	Control	Logic
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Addr Mod Set to −1	3rd Scan	14.71.41

46. Set skid cycle controls.

Set M Q Ctrl	Edit Set B Cy Ctrl D	16.30.05
M Q Ctrl Latch	Set M Q Ctrl, Next to Last LG	16.30.07
M Q Latch	M Q Ctrl Latch, LGC	16.30.07
Edit Skid Cy	M Q Latch, Z Op • B Cy	12.13.15

47. Store B-channel character.

Use B Ch WM	Edit Skid Cy	15.49.04
Store B Ch Char	Edit Skid Cy	17.13.03
Load Memory	Load Mem on B Cy Op Codes, B Cy	12.50.01

48. Take another B-cycle and read out B-field character.

Set B Cy Ctrl	Edit Skid Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, Next to Last LG	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
RO B AR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl RO B AR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16

49. Regenerate modify controls.

Regen 3rd Scan Ctrl	M Q Latch	17.13.15
3rd Scan Ctrl	Next to LLG, Regen 3rd Scan Ctrl 3rd Scan	12.30.04
3rd Scan	3rd Scan Ctrl, LGC	12.30.02
Addr Mod Set to −1	3rd Scan	14.71.41

50. The assembly controls cause the B-channel character to be stored until a blank is read out; the blank causes a $ to be stored and the operation is ended.

Set $ * Edit	E or Z • 3rd Scan • Extension, Float $, Blank	17.13.06
Last Execute Cy	Set $ * Edit	12.12.51

Questions on Edit Operation

Answers to review questions are in the Appendix.

1. How many times can the MQ *latch be set in an edit operation?*

2. What condition causes the edit operation to end when the first scan is complete? What conditions cause the edit operation to end when the second scan is complete?

3. If the B-field contains $\dot{0}0.000$ *at the end of the first scan and the floating dollar latch is not set, will the* CPU *initiate a third scan when the second scan is complete?*

4. Does the CPU *take A- and B-cycles during the first, second, and third scans?*

5. Are B-field characters read during the first scan when the extension latch is set in the body or the status portion of the control word?

6. What limitation is imposed on the length of the A-field?

Compare Instruction

Instruction Formats

Formats for the compare instruction are:

Op code	A-address	B-address
$\dot{C}$	xxxxx	xxxxx
$\dot{C}$	xxxxx	
$\dot{C}$		

The compare instruction causes B-field data to be compared to data in the A-field. (The comparison is always B to A, never A to B). Execution of the compare instruction does not change data stored in either field. The result of the compare sets the compare high (B>A), compare low (B<A), or compare equal (B=A) latch. A subsequent test and branch instruction checks the state of the latches.

When the A-field is shorter than the B-field, the compare high latch is set, designating the B-field as the greater of the two fields, regardless of their values. If, however, the B-field is shorter than the A-field, the low latch is not set unconditionally at the end of the compare operation.

At the end of the operation, the A- and B-address registers contain the original A- and B-addresses minus the length of the A- or B-field, whichever is shorter.

CPU Operation

During last instruction read-out cycle, the units, first scan, and A-cycle control latches are set. To execute the compare instruction, the CPU takes an A-cycle to read out the A-field character that the AAR specifies; the character is stored in the A-data register. The CPU then takes a B-cycle to read out and gate the B-field character onto the B-channel. The character in the A-data register is gated to the A-channel while the B-field character is gated to the B-channel. A- and B-channel characters are compared in the adder and compare units. The output of the compare matrix sets the high, low, or equal latch. The equal latch can be set only when the units positions of the fields are compared. If all A- and B-field characters are equal, the equal latch is not reset in the compare operation.

The CPU alternately executes A- and B-cycles, comparing A- and B-field characters and setting or resetting the high and low latches as required, until a word mark is sensed in either the A- or the B-field. The CPU terminates the compare operation when either an A- or B-channel word mark is detected. The last compare latch set (high, low, or equal) remains on to indicate the result of the operation.

Figures 26 and 27 show detailed operation in the execution of the compare instruction.

Examples 1 and 2 following illustrate the compare operation.

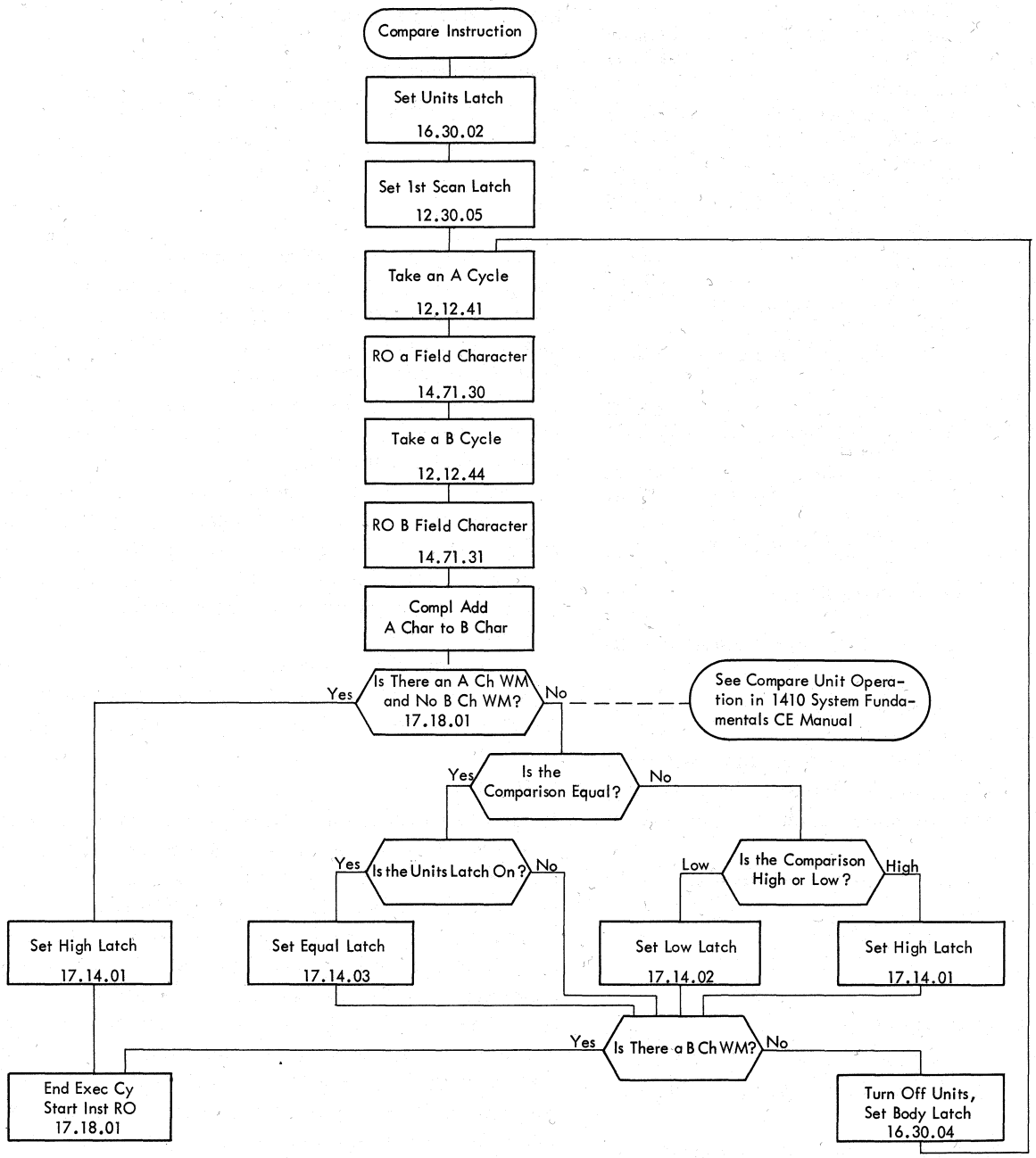


Figure 26. Compare

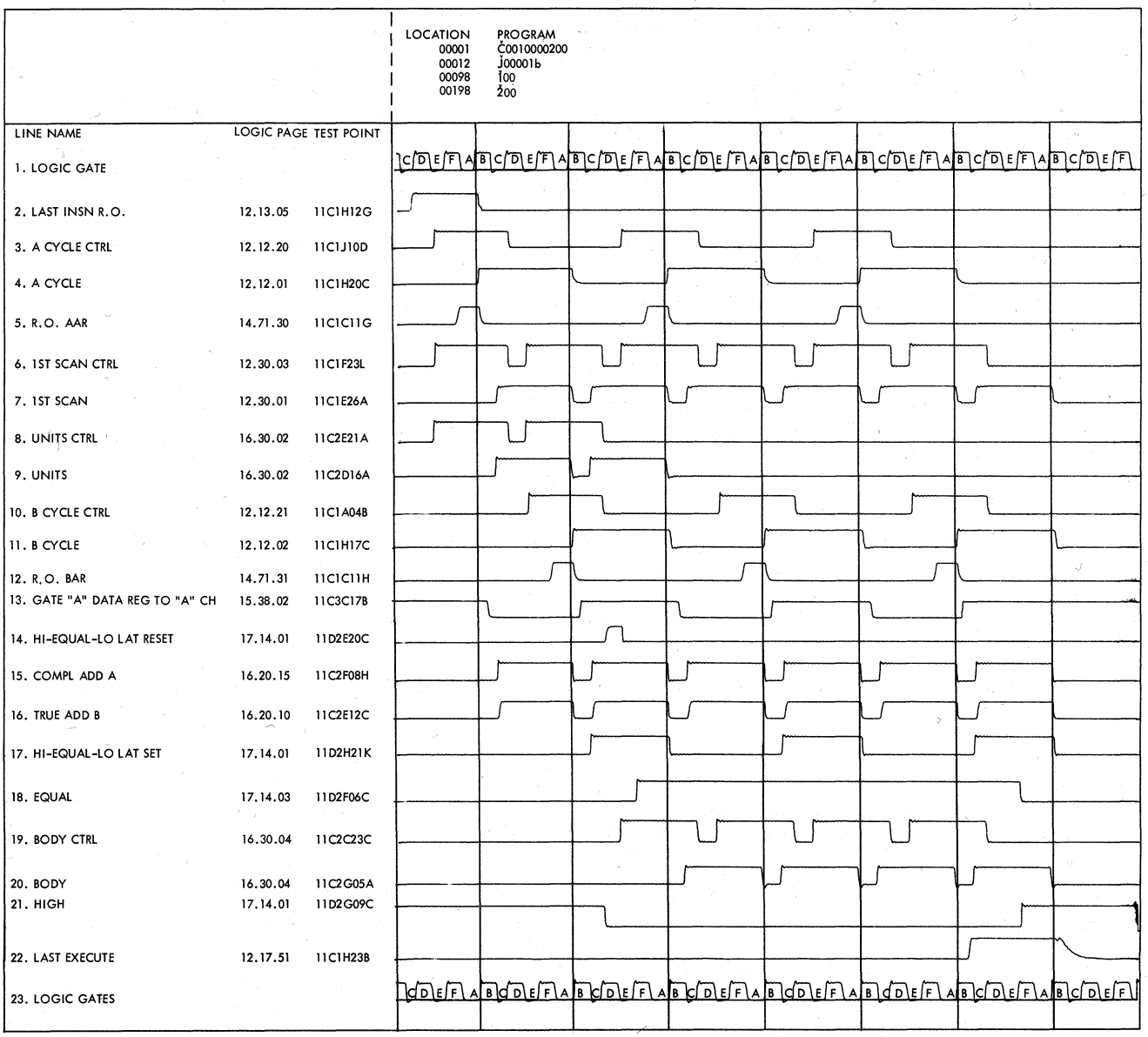
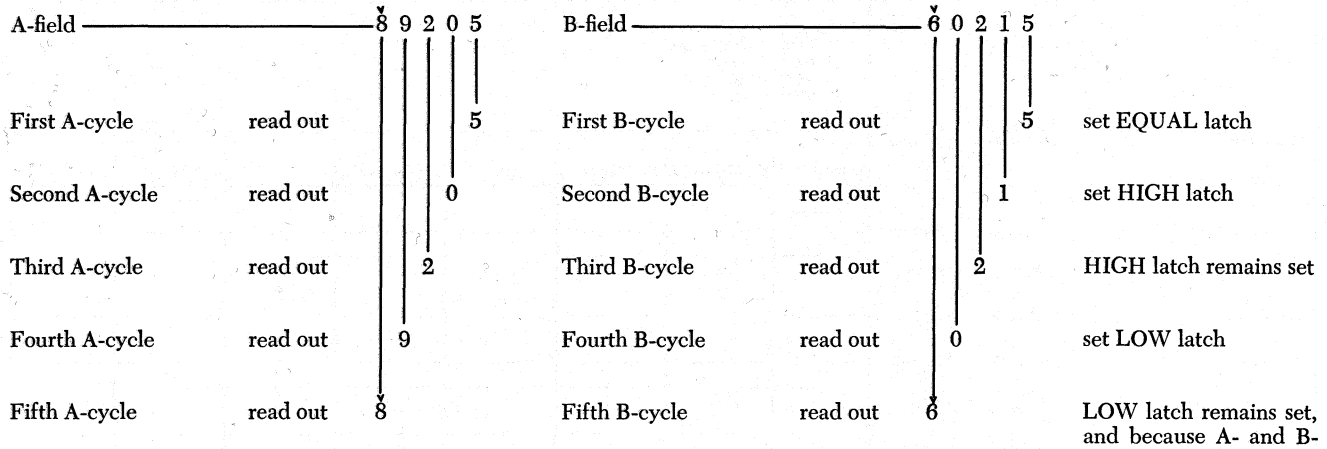


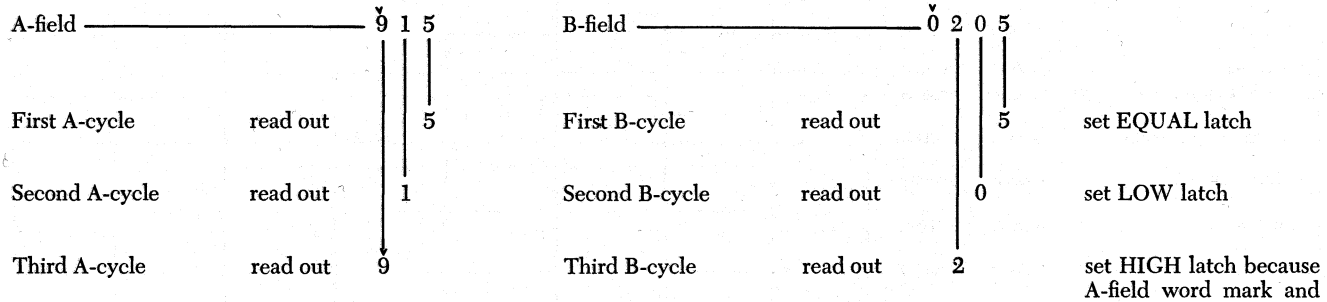
Figure 27. Compare Operation Timings

EXAMPLE 1



The comparison designates the A-field as the greater of the two fields as indicated by the LOW latch being set when the comparison is terminated.

EXAMPLE 2



Although A-field is numerically greater than B-field, the comparison ends with the HIGH latch set, designating B-field as the greater of the two fields, because A-field is shorter than B-field.

The following controls are active when the CPU performs a compare operation:

SIGNAL	CONDITION	LOGIC
1. Initiate A-cycle and read out first A-field character.		
Set A Cy Ctrl	A Cy First Op Codes Last Instruction Read Out Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl Next to Last Logic Gate	12.12.20
A Cy	A Cy Ctrl, LGB	12.12.01
Read Out AAR	A Cy Ctrl, Logic Gate Special A, Read Out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
2. Set modifier controls.		
Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
3. Set character into A-data register.		
Sw B Ch to A Reg	A Cy, LGD	15.38.01
4. Initiate B-cycle and read out first B-field character.		
A Cy Ops A Cy	Compare Op Code, A Cy	13.14.06
Stop at F	A Cy Ops A Cy	12.12.30
5. Initiate B-cycle and read out first B-field character.		
Set B Cy Ctrl	A-Cy Ops, A-Cy	12.12.44
B Cy Ctrl	Set B-Cy Ctrl, Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl	12.12.02
Read Out BAR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops	14.17.31
Set Mem AR Gated	LGA 2nd CP	14.17.16
6. Gate A-field character to A-channel.		
Gate A Data Reg to A Ch	B-Cy, A Reg to A Ch on B Cy Ops	15.38.02
7. Control B-cycle length.		
Stop at F	B Cy, Compare Op Code	12.12.30
8. Set high latch if there is an A-channel WM with no B-channel WM.		
Set High Cy	B Cy, Compare Op Code, A Ch WM, B Ch Not WM	17.18.01
High	Set High Cy, LGF	17.14.01
Last Execute Cy * TLU	Set High Cy	17.18.01

Signal	Condition	Logic
9. Set equal latch if compare result is equal for the units positions.		
Units Ctrl Latch	Last Instruction Read Out	16.30.02
Units Latch	Units Ctrl Latch, LGC	16.30.02
Regen Units and Body Ctrl	Standard A Cy Ops A Cy	16.30.01
Units Ctrl Latch	Regen Units and Body Ctrl Next to Last Logic Gate	16.30.02
Equal Low Latches Set	B Cycle, Compare Op Code Units Latch	17.14.01
Equal	Units Latch, Equal Low Latches Set Compare Equal, Not Set High Cy	17.14.03
10. Take another A-cycle and read out next A-field character.		
Compare Mode Set A Cy Ctrl	B Cycle, Compare Op Code A Ch Not WM, B Ch Not WM	17.18.01
Set A Cy Ctrl	Compare Mode Set A Cy Ctrl	12.12.41
A Cy Ctrl	Set A Cy Ctrl Next to Last Logic Gate	12.12.20
A Cy	A Cy Ctrl, Logic Gate Special A Read Out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	LGA, 2nd CP	14.17.16
11. Set body latch.		
Set Body Ctrl Latch	Compare Mode Set A Cy Ctrl	16.30.01
Body Ctrl Latch	Set Body, Ctrl Latch Next to Last Logic Gate	16.30.04
Body Latch	Body Ctrl Latch	16.30.04
Regen Units and Body Ctrl	Standard A Cy * A Cy	16.30.02
12. Regen modify controls.		
Regen 1st Scan Ctrl	Standard A Cy Ops A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl, 1st Scan, Next to Last Logic Gate	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41
13. Set character into A-data register.		
Sw B Ch to A Reg	A Cy, LGD	15.38.01
14. Take another B-cycle and read out next B-field character.		
Standard A Cy Ops A Cy	Compare Op Code, A Cy	13.14.06
Set B Cy Ctrl	Standard A Cy Ops, A Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl, L Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Read Out BAR	B Cy Ctrl, 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd CP	14.17.16
15. Control A-cycle length.		
Stop at F	Standard A Cy Ops, A Cy	12.12.30
16. Gate A-field character to A-channel.		
Gate A-Data Reg to A Ch	B Cy, A Reg to A Ch on B Cy Ops	15.38.02
17. Control B-cycle length.		
Stop at F	B Cy, Compare Op Code	12.12.30
18. Set high or low latch.		
Equal Low Latches Set	Body Latch, Compare Mode B Cy	17.14.01
High Latch	Compare High, Equal Low	17.14.01
Low Latch	Compare Low, Not Set High Cy, Equal Low Latches Set, LGF	17.14.02
19. Initiate instruction read out.		
Last Execute Cy TLU	B Ch WM, B Cy, Compare Op Code	17.18.01

Note: Steps 10 through 18 are repeated until a WM is read out.

Questions on Compare Operation

Answers to review questions are in the Appendix.

1. The compare instruction causes the CPU to compare B-field characters to characters in the A-field. Are contents of either field changed as a result of the compare operation?

2. What conditions terminate the operation?

3. Is the compare high, compare low, or compare equal latch set unconditionally if the B-field is longer than the A-field?

4. What action occurs if the A-field is longer than the B-field?

5. What conditions must exist before the compare equal latch can be set?

6. Interpret the result of the comparison when the compare low latch is set at the end of the compare operation.

Table Lookup Instruction

Definitions

The table lookup instruction causes the CPU to search through a previously prepared table in core storage to locate a specific function.

A function is either a segment of actual data or the storage address of a segment of data compiled in the table; for example, if multiples of five are listed in the table, 5, 10, 15, 20, etc. are table functions.

To recognize the desired function, the system requires two arguments, the table argument and the search argument. Each function in the table has a prefix called a table argument to provide separate identification for each function; for example, if 5, 10, 15, 20, and 25 are table functions, 501, 1002, 1503, 2004, and 2505 might appear in the table; the two low-order digits preceding each function make up a table argument. Each table argument must contain the same number of characters. A function and corresponding table argument make up a table field.

To locate the specific function in the table, the CPU compares a search argument with table arguments. The search argument is identification data read in from an input device or generated by the program. Although the search argument is stored in core storage, it is not

part of the table. The table lookup instruction designates the search argument to be used in the operation. The instruction specifies the condition (high, low, or equal) sought in the search argument-table argument comparisons.

The time required to execute the table lookup instruction is determined by the number of table fields and the number of characters in each table field encountered before the desired function is found. Therefore, functions should contain the least possible number of characters when adequate core storage space is available. If segments of data compiled in the table contain five characters or less, the data are stored in the table; in this case, the function is actual data. If segments of data compiled in the table contain more than five characters, the data are usually located in another area of core storage. In this case, the function is the 5-character address of the desired segment of data.

Instruction Formats

Formats for the table lookup instruction are:

OP CODE	A-ADDRESS	B-ADDRESS	d-CHARACTER
T	xxxxx	xxxxx	See Figure 28
T	xxxxx		
T			

The table lookup instruction causes the CPU to search for a table argument that is equal to, lower, or higher than the search argument as specified by the d-character (Figure 28) in the instruction. The A-address in the instruction specifies the address of the low-order position in the search argument. The B-address is the location of the low-order character in the table. The d-character specifies the condition to stop the table search. If the instruction does not contain a B-address or d-character, the contents of the BAR and operation-modifier register from the previous operation are substituted for the B-address and d-character, respectively. If the instruction does not contain an A-address, the contents of the AAR from the previous operation become the A-address in the table lookup operation.

Description	d-Char	Bits	Stop if Table Argument is:
Lookup Equal	2	2	Equal to Search Argument
Lookup Low	1	1	Lower than Search Argument
Lookup High	4	4	Higher than Search Argument
Lookup Low or Equal	3	2,1	Equal to or Lower than Search Argument
Lookup Equal or High	6	4,2	Equal to or Higher than Search Argument
Lookup Low or High	5	4,1	Lower than or Higher than Search Argument

Figure 28. d-Characters for Table Lookup Operation

The table argument and the search argument must contain the same number of characters. However, the table field (the table argument and the function) is longer than the search argument.

Description of Operation

The search argument (A-field) must have a word mark set to define its high-order position. The table field must have a defining word mark in the high-order function position. The CPU compares the search argument against the table argument until an A-field word mark is sensed in the search argument. If the condition established by the d-character in the instruction is not met and a B-field word mark is not sensed before the A-field word mark is detected, the CPU takes B-cycles to skip through the corresponding function (the CPU takes a B-cycle for each character in the function; for this reason, the function should contain only the necessary number of characters). The search argument (A-field) is compared against the next table argument when the B-field word mark indicating the end of the previous table field is sensed. A B-field word mark, denoting the end of a table field, encountered before the A-field word mark is found, indicates the end of the table. This condition causes the high-compare indicator to turn on unconditionally, and the table lookup operation terminates immediately. In the case of a single character A-field, the condition specified by the d-character must be met to stop the operation.

Figure 29 shows an example of the table search in locating the square of a number (search argument). A table of squares is stored in core storage. The table includes the number (table argument) followed by the square of the number (function). In this case, the function is a segment of actual data. The end of the table is designated by a B-field shorter than the corresponding A-field. Figures 30 and 31 show CPU operation and timings in the execution of the table lookup instruction.

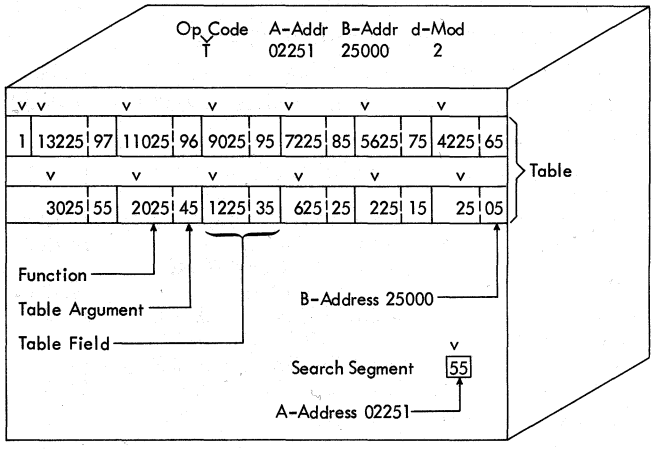


Figure 29. Storage Table for Table Lookup Operation

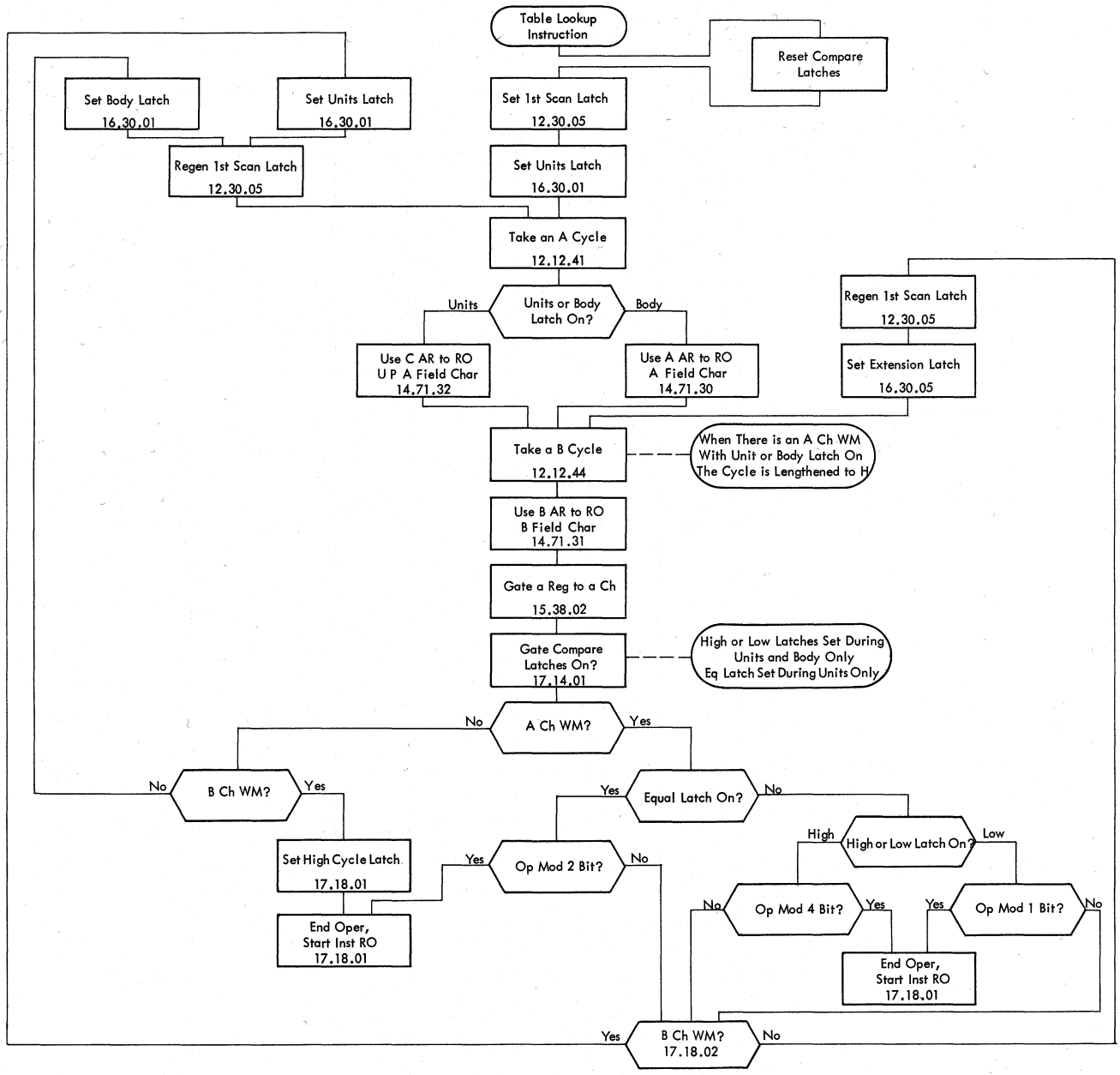


Figure 30. Table Lookup

The following controls are active when the CPU performs a table lookup operation:

Signal	Control	Logic
1. Set controls for forward scan; start with the units position of the arguments.		
Set 1st Scan Ctrl	Last Instruction Read Out First Scan First Op Codes	12.13.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.13.03
1st Scan	1st Scan Ctrl LGC	12.13.01
Regen 1st Scan Ctrl	A Cy Standard A Cy Ops	12.30.05
Set Units Ctrl Latch	Last Instruction Read Out	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02
Regen Units and Body Ctrl	Standard A Cy Ops A Cy	16.30.01
2. Take an A-cycle, and use CAR to read out units position of search argument.		
Set A Cy Ctrl	Last Instruction Read Out Cy A Cy First Op Codes	12.12.41
A Cy Ctrl	Set A Cy Ctrl Next to Last Logic Gate	12.12.20
A Cy	A Cy Ctrl LGB	

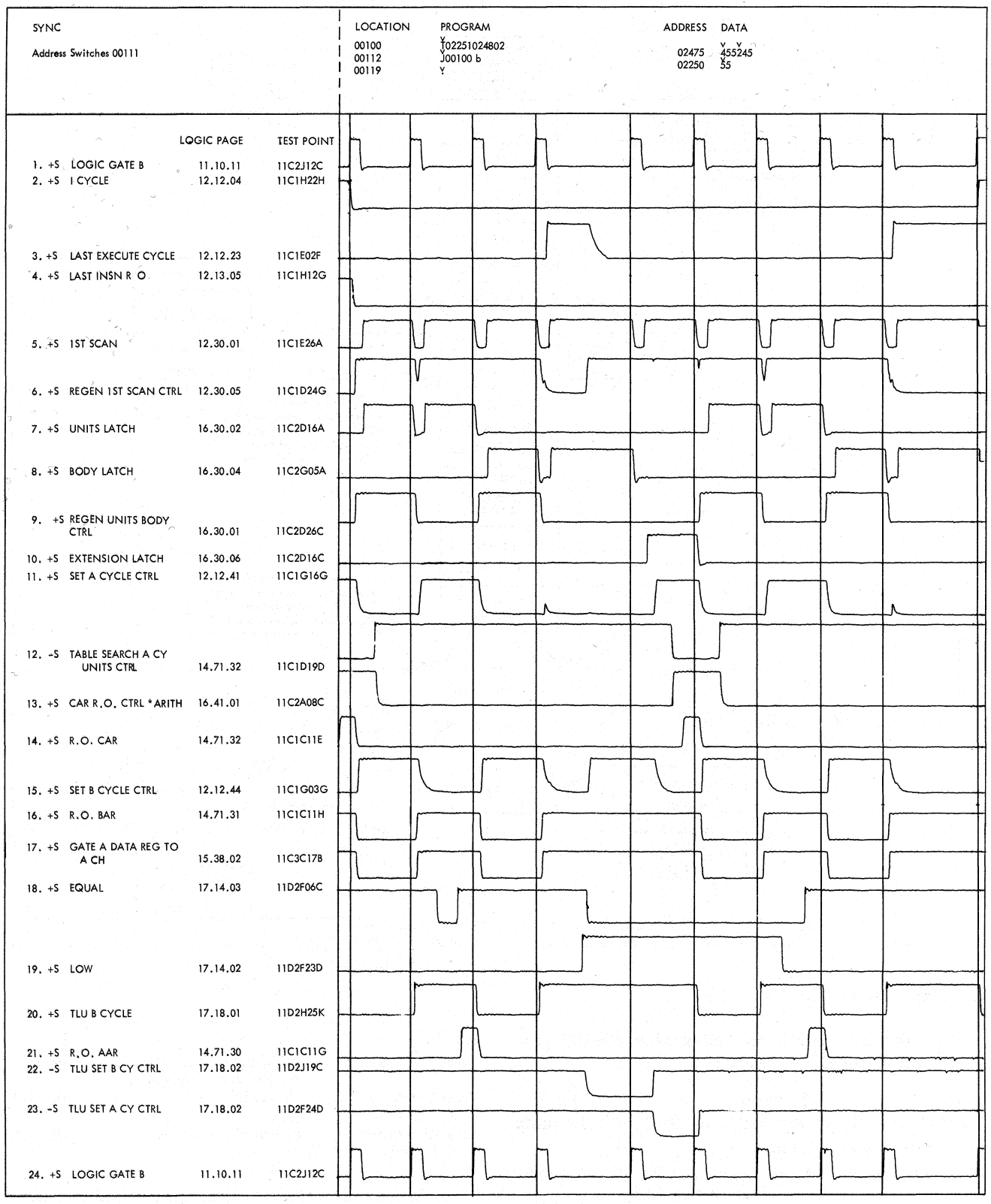


Figure 31. Table Lookup Operation Timings

Signal	Control	Logic
Table Search A Cy Units Ctrl	A Cy Ctrl	14.71.32
	Units Ctrl Latch	
	Table Search Op Code	
CAR Read Out Ctrl	Table Search A Cy Units Ctrl	16.41.01
* Arith		
Read Out CAR	CAR Read Out Ctrl	14.71.32
	* Arith	
	Logic Gate Special A	

3. Take a B-cycle and use BAR to read out units position of table argument.

Set B Cy Ctrl	Standard A Cy Ops	12.12.44
	A Cy	
B Cy Ctrl	Set B Cy Ctrl	12.12.21
	Next to Last Logic Gate	
B Cy	B Cy Ctrl	12.12.02
	LGB	
Read Out BAR	B Cy Ctrl	14.71.31
	1st + 2nd + 3rd Scan Ctrl	
	Logic Gate Special A	
	Not Console Inhibit AR Read Out	

4. Compare A-channel with B-channel, and set high, equal, or low compare latch.

Gate A Data Reg to A Ch	B Cy	15.38.02
	A Reg to A Ch on B Cy Ops	
CMP Mode B Cy	B Cycle	17.18.01
	Table Search Op Code	
Equal Low Latches Set	CMP Mode B Cy	17.14.01
	Units Latch	
Equal	Equal Low Latches	17.14.03
	CMP Equal	
	Not Set High Cy	
	Units Latch	

5. Check for A-channel or B-channel word mark; no A-channel or B-channel word mark initiates another A-cycle; read out AAR (tens position search argument).

CMP Mode Set A Cy Ctrl A	A Ch Not WM Bit	17.18.01
	B Ch Not WM Bit	
	Table Search Op Code	
Set Body Ctrl Latch	CMP Mode Set A Cy Ctrl A	16.30.01
Body Ctrl Latch	Set Body Ctrl Latch	16.30.04
	Next to Last Logic Gate	
Body Latch	Body Ctrl Latch	16.30.04
	LGC	
Set A Cy Ctrl	CMP Mode Set A Cy Ctrl A	17.18.01
A Cy Ctrl	Set A Cy Ctrl	12.12.20
	Next to Last Logic Gate	
A Cy	A Cy Ctrl	12.12.01
	LGB	
Read Out AAR	A Cy Ctrl	14.71.30
	Body Ctrl Latch	
	Table Search Op Code	
	Not Console Inhibit AR Read Out	
	Logic Gate Special A	

6. Take a B-cycle. Use BAR to read out tens position of table argument; an A-channel word mark with units or body latch on extends this B-cycle to H; set the high, low, or equal compare latch.

Set B Cy Ctrl	Standard A Cy Ops	12.12.44
	A Cy	
B Cy Ctrl	Set B Cy Ctrl	12.12.21
	Next to Last Logic Gate	
B Cy	B Cy Ctrl	12.12.02
	LGB	
Read Out BAR	B Cy Ctrl	14.71.31
	1st + 2nd + 3rd Scan Ctrl	
	Logic Gate Special A	
TLU B Cy	B Cy	17.18.01
	Table Search Op Code	
Stop at H	TLU B Cy	17.18.02
	A Ch WM Bit	
	Units or Body Latches	
CMP Mode B Cy	B Cy	17.18.01
	Table Search Op Code	
Equal Low Latches Set	CMP Mode B Cy	17.14.01
	Body Latch	
Low	Equal Low Latches Set	17.14.02
	Not Set High Cy	
	CMP Low	
	LGF	

7. The next step to be taken is now determined by:
   a. A-channel word mark and no B-channel word mark.
   b. Compare latch set (low).
   c. Op modifier register character (2).

Note: the extension of B-cycle to H provides the necessary time to check these conditions.

TLU Set B Cy Ctrl	Low	17.18.02
	Op Mod Reg Not 1 Bit	
	TLU B Cy	
	A Ch WM Bit	
	B Ch Not WM Bit	
Set Extn Ctrl Latch	TLU Set B Cy Ctrl	16.30.05
Extn Ctrl Latch	Set Extn Ctrl Latch	16.30.06
	Next to Last Logic Gate	
Extn Latch	Extn Ctrl Latch	16.30.06
	LGC	

8. Take continuous B-cycles until a B-channel word mark (End of Function Field) is detected; initiate an A-cycle for units position of next table argument.

Set B Cy Ctrl	Standard A Cy Ops	12.12.44
	A Cy	
B Cy Ctrl	Set B Cy Ctrl	
	Next to Last Logic Gate	
B Cy	B Cy Ctrl	
	LGB	
TLU Set A Cy Ctrl B	TLU B Cy	17.18.02
	Low	
	Not 1 Bit	
	A Ch WM Bit	
	B Ch WM Bit	
Set Units Ctrl Latch	TLU Set A Cy Ctrl B	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch	16.30.02
	Next to Last Logic Gate	
Units Latch	Units Ctrl Latch	
	LGC	
Set A Cy Ctrl	TLU Set A Cy Ctrl B	12.12.41

Questions on Table Lookup Operation

Answers to review questions are in the Appendix.

1. When is the table lookup operation terminated?

2. To locate the desired function, what comparisons are made?

3. When the CPU executes the following instruction, what address does the BAR contain when the operation is complete?

Ť 02000 12345 2
Search Argument 0̌10
Table 00500100500850041000025̌0001

4. Is storage data altered as a result of the execution of a table lookup instruction?

5. Does the CPU *perform A-cycles or B-cycles to bypass functions in the table?*

6. Does the A-field contain the table argument or the search argument?

The IBM 1410 Data Processing System can be programmed to test for conditions that can occur during processing, and transfer (branch) to a predetermined set of instructions or sub-routines as a result of the specific test. This is called the logical ability of the system. A transfer (branch) from one instruction to another (or set of instructions) to alter the sequential execution of program steps is called a program branch. A branch instruction can be:

1. An uncondtiional branch that occurs as a direct result of the execution of the branch instruction (no special condition, other than the execution of that program step is needed to transfer the program out of its normal sequential execution); or,

2. A conditional branch that occurs as a result of a particular condition such as an arithmetic overflow, zero balance, etc. If the condition is present at the time that a conditional branch instruction is executed, sequential execution of program steps immediately following is bypassed. The program branches to the address of the instruction specified by the I-address of the conditional branch instruction. If the condition is not present, the CPU executes the next sequential instruction; no branch occurs.

All branch instructions have:

1. A d-character to specify the condition necessary for a program transfer.

2. An I-address instead of an A-field address. The I-address is stored in the AAR during instruction read out. The I-address represents the location of the op code in the instruction to which the program branches if conditions for the branch are met.

The CPU executes all branch operations in the same manner; the various branch instructions merely establish specific conditions which cause the CPU to perform the branch operation. To execute a program branch, the CPU takes a B-cycle; the IAR which contains the address of the next sequential instruction, reads out to the STAR and through the modifier to the BAR. The no-scan latch is set, causing a modify-by-zero condition in the modifier.

Note: The B-cycle to transfer the contents of the IAR (containing the address of the next sequential instruction) to the BAR permits the program to return to the point of interruption when the branched-to routine is completed. To effect the return, the first instruction in the branched-to routine must be the store B address register instruction.

Completion of the successful branch is accomplished during the subsequent instruction read-out operation when the AAR (containing the branch I-field) sets STAR rather than the IAR.

The following controls are active in the execution of a successful branch operation:

SIGNAL	CONDITION	LOGIC
1. Set modify controls for transfer of IAR to BAR.		
Set No Scan Ctrl * Br Ops	(branch condition)	12.60.04
Set No Scan Ctrl	Set No Scan Ctrl * Br Ops	12.30.05
No Scan Ctrl	Set No Scan Ctrl Next to Last Logic Gate	12.30.03
No Scan	No Scan Ctrl	12.30.01
Addr Mod Set to 0	No Scan Ctrl	14.71.41
2. Set controls to read out AAR during next instruction read out cycle (first instruction of branch routine).		
Br to A Conds	(branch condition)	12.60.04
Br to AAR Latch	Br to A Conds Next to Last Logic Gate Not 1401 Mode	12.60.14
3. Take a B-cycle to read out IAR to BAR; modify by 0.		
Set B Cy Ctrl * BR Ops	(branch condition)	12.60.04
Set B Cy Ctrl	Set B Cy Ctrl * Br Ops	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out IAR	B Cy Ctrl No Scan Ctrl LG Special A Not Console Inhibit AR Read Out	14.71.34
Set BAR	B Cy LGD or LGE or LGF	
4. Initiate instruction read out.		
Last Execute Cy * Br Cond	B Cy	12.60.08
	No Scan Br Type Op Code	
Last Execute Cy	Last Execute Cy * Br Cond	12.12.51
I Cy Ctrl	Set I Cy Ctrl Next to Last Logic Gate	12.12.23
5. With branch to AAR latch set, the program moves to the first instruction in the branch routine.		
Read Out AAR	I Cy Ctrl Br to AAR Latch LG Special A	14.71.30

Unconditional Branch Instruction

The unconditional branch instruction causes the CPU to branch from the routine to the I-address; no condition must be satisfied to execute the transfer. The I-

address is the location of the op code in the next instruction to be executed; for example, the instruction J(05000) causes the CPU to unconditionally execute the program step beginning in storage position 05000.

The format for the unconditional branch instruction is:

OP CODE	I-ADDRESS	d-CHARACTER
J	xxxxx	blank

Figures 32 and 33 show detailed operation in the execution of the unconditional branch instruction.

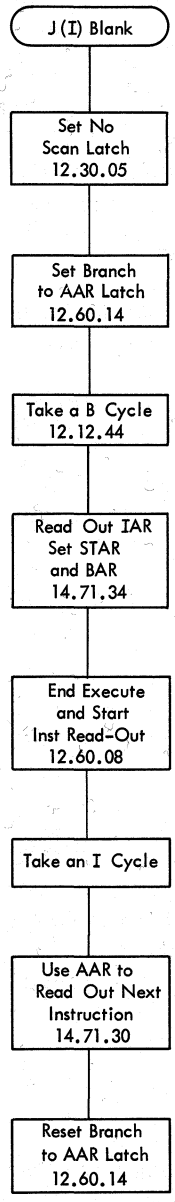


Figure 32. Unconditional Branch

Conditional Branch Instructions

Test and Branch

The test and branch instruction allows the program to test for conditions that can occur during processing and cause the CPU to execute a branch operation if the specific conditions exists. The format for the test and branch instruction is:

OP CODE	I-ADDRESS	d-CHARACTER
J	xxxxx	see Figure 34

The d-character specifies the internal indicator to be examined. If the indicator is on, the program branches to the I-address for the next instruction. If the indicator is off, the CPU executes the next instruction in sequence. The various indicators and the d-characters used to test them are shown in Figure 34. If an arithmetic or divide overflow has occurred when the J(I)Z(branch on arithmetic overflow) or J(I)W (branch on divide overflow) instruction is executed, the respective overflow indicator is reset. Other test and branch d-characters do not reset their corresponding indicators.

When performing a test and branch operation, the d-character in the instruction is set in the op-modifier register at I-ring 6 time of instruction read out. Depending on whether the indicator specified by the d-character is on or off, either the branch or no-branch condition is brought up at I-ring 7 time (last instruction read-out cycle). The branch condition indicates that the designated internal indicator is on, and the CPU performs the branch to the I-address. The no branch condition indicates that the indicator tested is off and brings up last execute cycle; the CPU does not perform the branch operation, but begins the instruction read-out phase of the next instruction in sequence. Figure 35 shows a diagrammed explanation of the test and branch operation.

The following controls are among the controls operative in the execution of the test and branch instruction (CPU operations in performing a successful branch are described earlier in the section):

SIGNAL	CONDITION	LOGIC
1. Op modifier register set with d-character.		
Set Op Mod Reg	I Ring 6 Time 1 Addr Plus Mod Op Codes LGE	15.38.04
2. Test internal indicator specified by d-character (Figure 34); if indicator is on, a program branch is executed.		
2nd Cond A Br Gated	2nd Cond A Br	12.60.02

Branch If I-O Channel Status Indicator On

The format for the branch if I-O channel status indicator on instruction is:

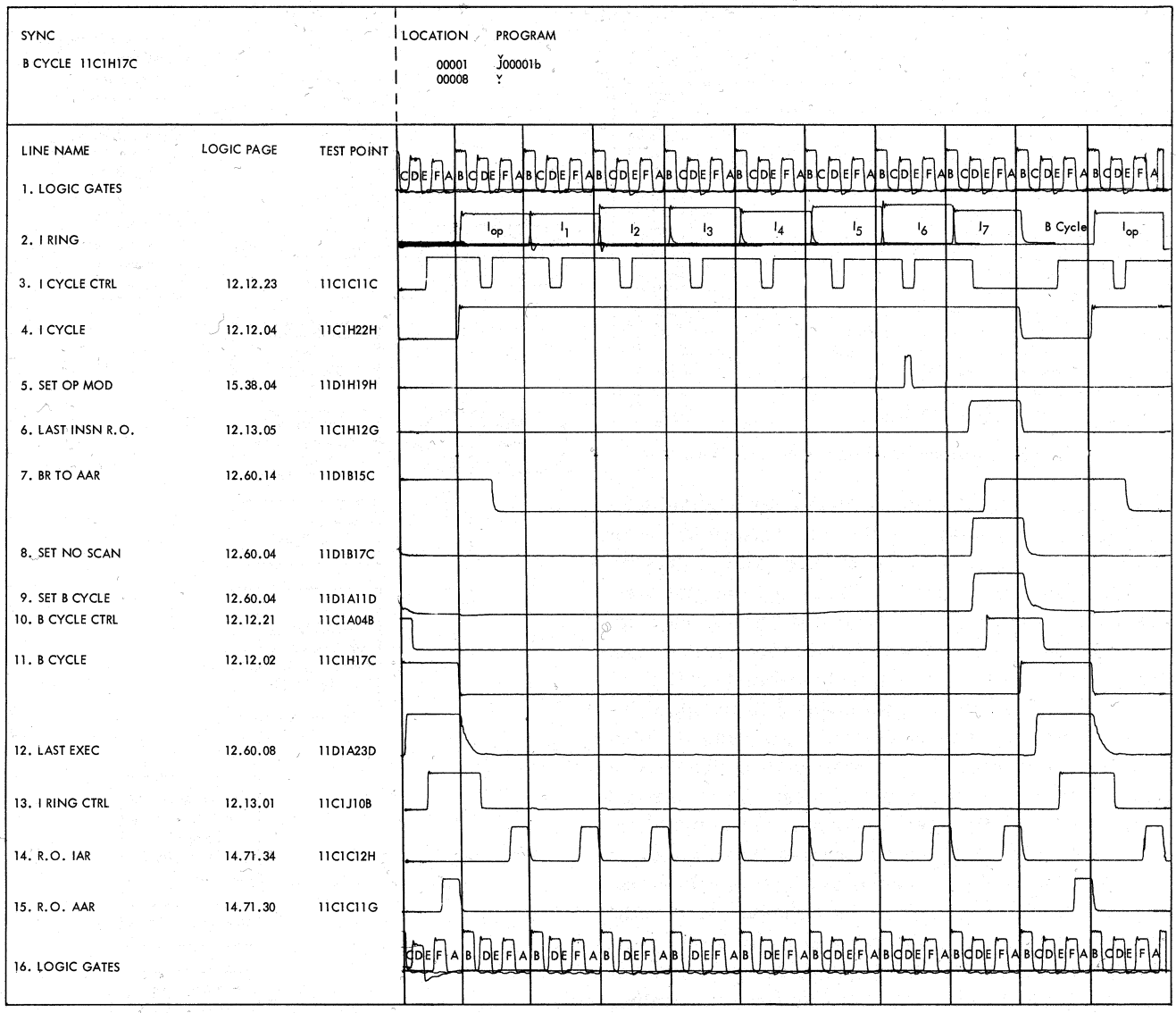


Figure 33. Unconditional Branch Timings

Op code	I-address	d-character
R̆ (Channel 1)	xxxxx	see Figure 36
X̆ (channel 2)	xxxxx	see Figure 36

The CPU executes a program branch to the I-address if the I-O channel status indicator(s) specified by the op code and the d-character in the instruction is on. I-O channel status indicators turn on as a result of conditions that occur during operation of any of the input-output units serviced by the channel.

The bit configuration of the d-character used in the branch if I-O channel status indicator on instruction designates the indicators to be tested. This instruction with a group mark (≡) d-character must be given prior to the execution of another input or output instruction on the same channel to avoid interlocking the system. The system is interlocked if the status test is not satisfied. The status test is satisfied if:

1. A branch if any I-O channel status indicator on instruction, R̆ or X̆ (I) ≡, is given prior to the next I-O unit instruction; or,

2. A specific R̆ or X̆ (I) d instruction (Figure 36) is given and results in a successful branch prior to the next I-O unit instruction.

If an R̆ (I) ≡ (B, A, 8, 4, 2, and 1 bits in the d-character) instruction is given following a channel 1 input-output operation, the instruction tests all channel 1 I-O channel status indicators, and if any indicators are on, the program branches to the specified I-address.

d-Char	Indicator	Logic
Blank	Unconditional Branch	12.60.02
9	Carriage Channel 9	12.61.13
@	Carriage Overflow (Channel 12)	12.60.01
/	Compare Unequal	12.60.01
S	Compare Equal (B=A)	12.60.01
T	Compare Low (B < A)	12.60.01
U	Compare High (B > A)	12.60.01
V	Zero Result	12.60.01
W	Divide Overflow	12.60.01
Z	Arithmetic Overflow	12.60.02
1	Overlap in Process on Channel 1	12.60.15
2	Overlap in Process on Channel 2	12.60.15
R	Printer Carriage Busy	12.61.13
Q	Branch Inquiry	12.60.02
K	Branch on Tape Mark Record	12.61.13

Figure 34. Branch Conditions for Test and Branch Instruction

The program can then test the indicators (individually or in groups, depending on the bit structure of the d-character) to determine the exact condition present.

If the system is equipped with the input-output overlap feature, the program should first test the overlap-in-process indicator with a test and branch instruction. This insures that the overlapped I-O function is complete before the branch if I-O channel status indicator on instruction is given. A branch if I-O channel status indicator on instruction executed while the machine is performing an I-O overlap operation causes the machine to interlock until the overlapped function is complete.

The test and branch on I-O channel status indicator on instructions are quite similar in operation. However, in the execution of the branch if I-O channel status indicator on instruction, if the channel to be tested is in process at I-ring 6 time, the disable-compute cycle is brought up to stop the CPU clock and prevent testing the indicators until the operation is complete. Figure 37 shows a diagrammed explanation of the branch if I-O channel status indicator on instruction.

The following controls are active in the execution of the branch if I-O channel status indicator on instruction (CPU operations in performing a successful branch are described earlier in the section):

Signal	Condition	Logic
1. Op mod register set with d-character.		
Set Op Mod Reg	I Ring 6 Time 1 Addr Plus Mod Op Codes LGE	15.38.04
2. Check I-O channel for in process condition. When I-O channel is in process, bring up compute disable.		
E Ch In Process	(Overlap operation on channel)	13.60.04
Compute Disable Cy	E Ch In Process Br On Status Channel 1	12.12.60
3. Test external indicator specified by d-character; if indicator is on, execute a program branch.		
2nd Cond A Br Gated	Last Instruction Read Out Cy (Op Mod Reg)	12.60.02

Note: I-O status latches for a given channel reset at I ring 4 time of a M/L instruction that uses that channel or at I ring 1 time for two character only ops.

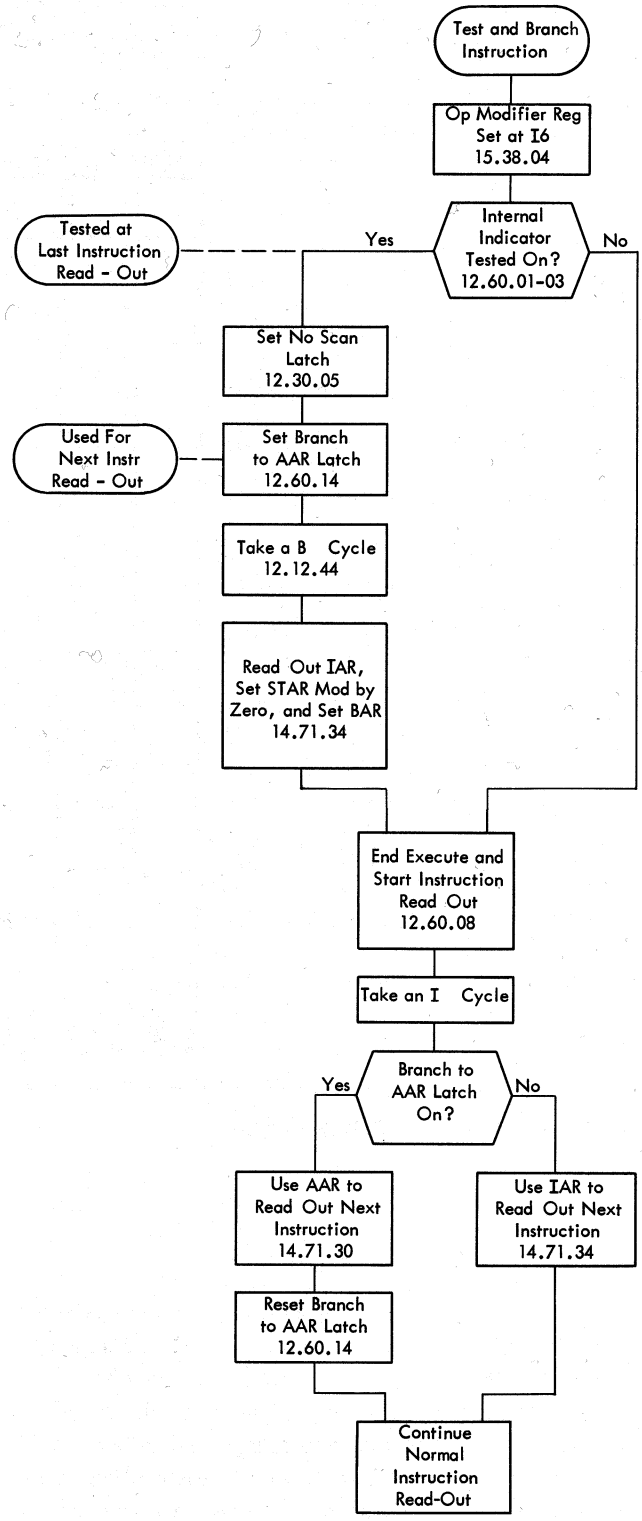


Figure 35. Test and Branch

Test Description	Status Latch Involved	d-Character Bits	ALD	Locations	
Branch if I-O unit not ready	Not Ready	1	12.60.02	12.60.15	The indicator is internally set during instructions involving I-O devices if these devices or their associated synchronizers are in a not ready condition, but before data transfers are taken. If the indicator is on, the operation is terminated; no data are transferred.
Branch if I-O unit busy	Busy	2	12.60.02	12.60.15	The indicator is internally set during instructions involving I-O devices if these devices or their associated synchronizers are in a busy condition, but before data transfers are taken. If the indicator is on, the operation is terminated; no data are transferred.
Branch if I-O data check	Data Check	4	12.60.02	12.60.15	The indicator is set on after data transfers to or from I-O devices, their associated synchronizers, or the CPU if a parity error was detected during the transfer.
Branch if I-O condition	Condition	8	12.60.02	12.60.15	The indicator is normally set during the move or load instruction before data transfers occur. For example, the indicator is set on if an end of file (last card stacked) has occurred in the card reader. When the indicator is on, the operation ends.
Branch if I-O no transfer	No Transfer	A	12.60.02	12.60.15	The no transfer indicator is set when no data are available to be transferred.
Branch if I-O wrong length record	Wrong Length Record	B	12.60.02	12.60.15	The wrong length record indicator is set when the record written in storage or read from storage is not the correct length.
Branch if any I-O channel status indicator on	Any		12.60.02	12.60.15	Branch to I-address if any I-O channel status indicators are on.

Figure 36. Branch Conditions for Branch if I-O Channel Status Indicator on Instruction

Branch If Character Equal

Formats for the branch if character equal instruction are:

OP CODE	I-ADDRESS	B-ADDRESS	d-CHARACTER
B	xxxxx	xxxxx	any BCD character
B	xxxxx		
B			

The branch if character equal instruction causes the character in the storage position designated by the B-address to be compared to the d-character in the instruction. If both characters (the character in storage and the d-character) have the same bit configuration, the CPU executes a program branch to the instruction beginning at the I-address. If the two characters are not exactly the same, the CPU executes the next instruction in sequence. If the instruction specifies only the op code, the contents of the AAR, BAR, and op mod register from the previous operation designate the I-address, B-address, and d-character, respectively. Word marks do not affect this operation.

When the branch if character equal instruction is executed, the compare high, low, or equal indicator is set. The high indicator is set if the B-address character is higher than the d-character (collating sequence).

The branch if character equal instruction causes the CPU to take a B-cycle to read out the character at the B-field address onto the B-channel. The op modifier register is gated to the A-channel. A comparison between the A- and B-channel characters is made in the compare and adder units. If the equal latch is set, the program branches to the I-address in the AAR. In all other cases, the CPU executes the next sequential instruction. Figures 38 and 39 show diagrammed operation of the branch if character equal instruction.

The following controls are active in the execution of the branch if character equal instruction (CPU actions in performing a successful branch are described earlier in the section):

SIGNAL	CONDITION	LOGIC
1. Set op mod reg with d-character.		
Set Op Mod Reg	I Ring 11 Time 2 Addr Plus Mod Op Codes LGE	15.38.04
2. Prepare CPU to read out character designated by BAR.		
Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05

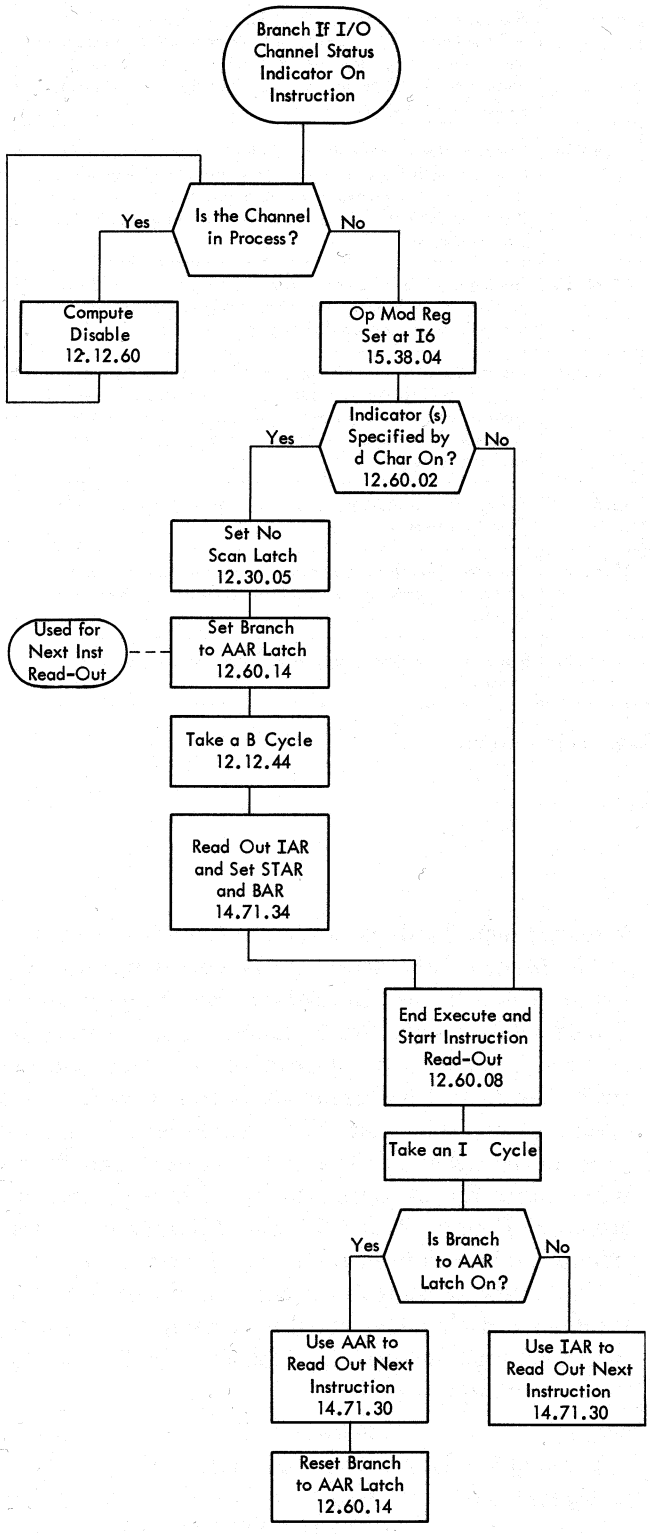


Figure 37. Branch if I-O Channel Status Indicator On

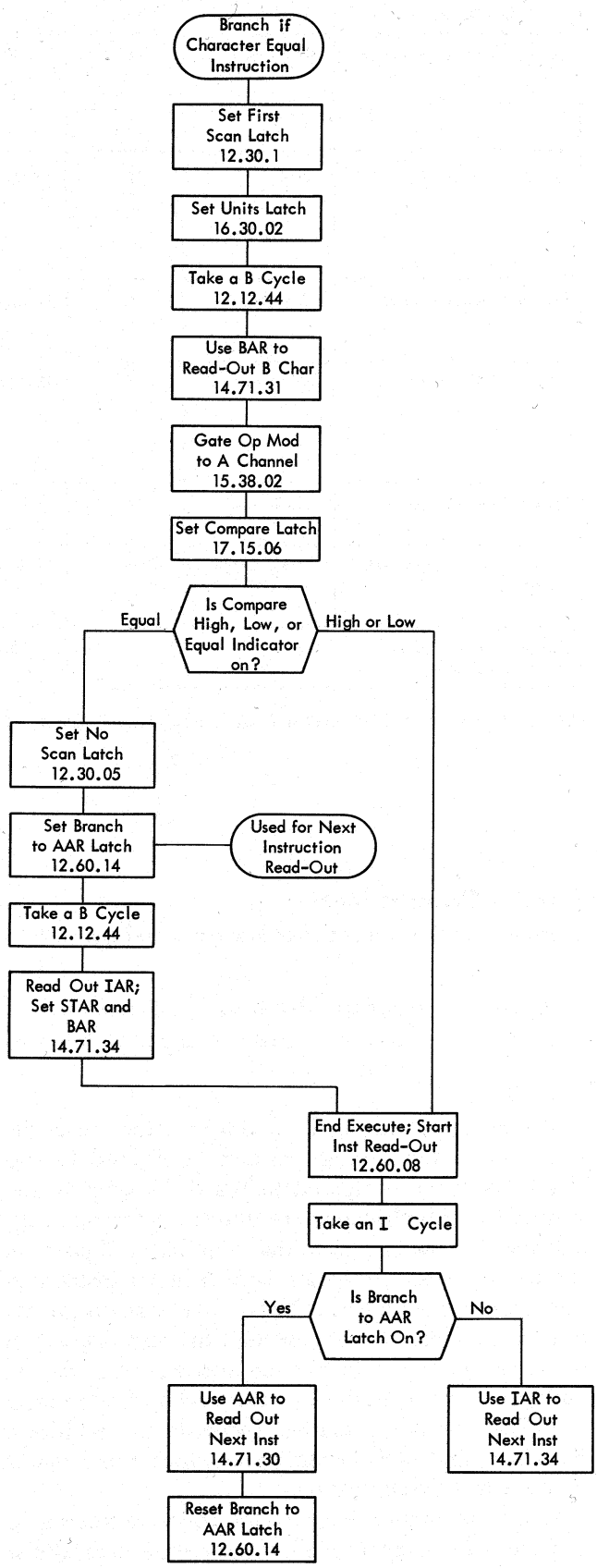


Figure 38. Branch if Character Equal

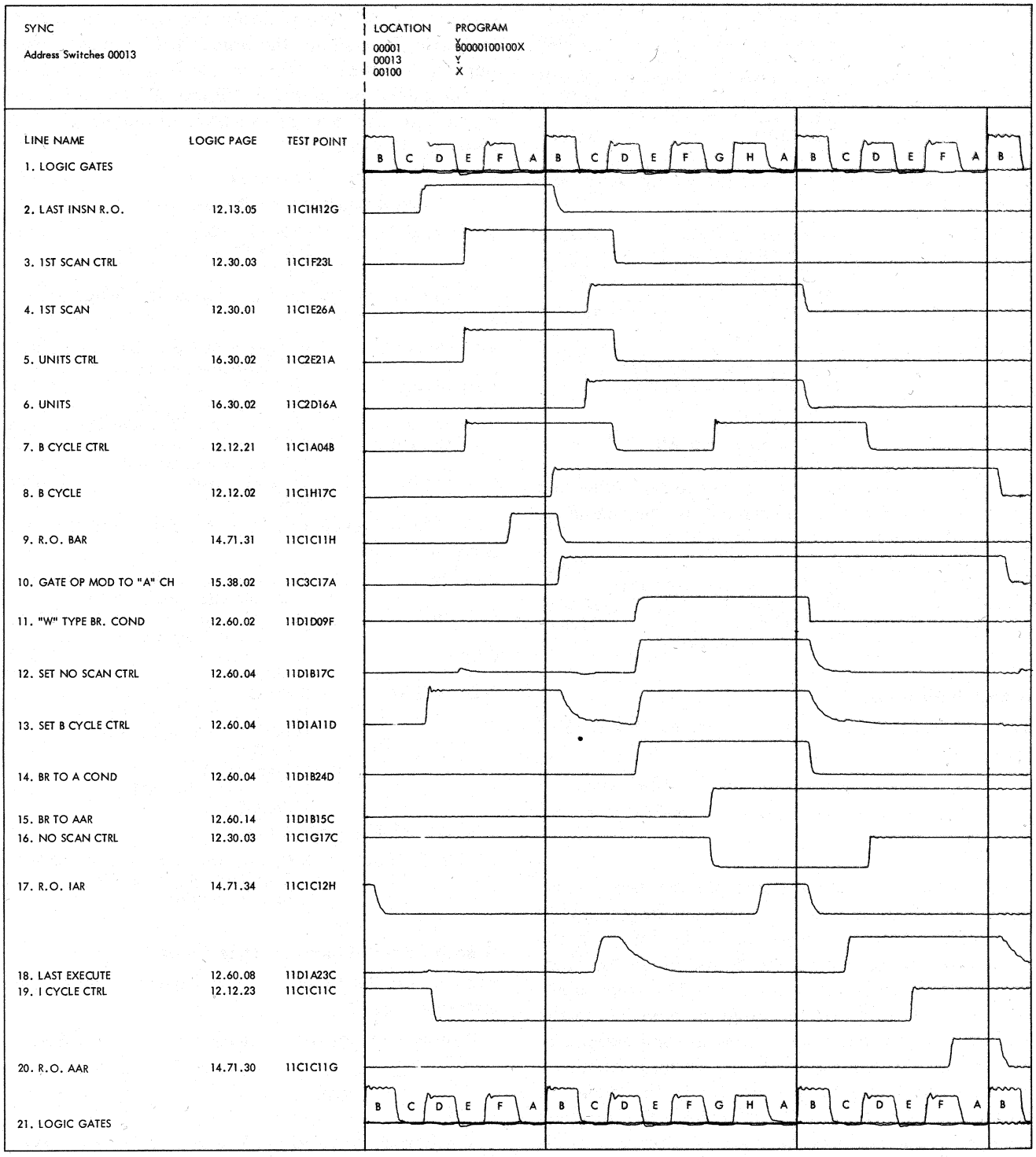


Figure 39. Branch if Character Equal Timings

Signal	Control	Logic
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
Set Units Ctrl Latch	Last Instruction Read Out Cy	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02

3. Take a B-cycle to read out B-field character onto B-channel.

Set B Cy Ctrl	B Cy First Op Codes Last Instruction Read Out Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out Bar	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops LG Special A Not Console Inhibit AR Read Out	14.71.31

4. Gate op mod register character to A-channel and compare with B-channel character; if the characters are equal, execute a program branch.

Gate Op Mod Reg to A Ch	B Cy Op Mod to A Ch on B Cy Ops	15.38.02
Comp Equal	Adder Equal	17.15.06
1st Cond A Br Gated	Comp Equal Char Test Br Op Code	12.60.03

Branch If Bit Equal

The instruction formats for the branch if bit equal instruction are:

Op code	I-address	B-address	d-character
W	xxxxx	xxxxx	any BCD character
W	xxxxx		
W			

The branch if bit equal instruction causes the character located in the storage position designated by the B-address to be compared, bit by bit, with the d-character in the instruction. If any bit in the character at the B-address matches any bit in the configuration of the d-character, the program branches to the I-address. Word marks and C-bits are not compared. For example, if position 06779 (B-address) contains a Z (CBA81 bits) and the d-character in the branch if bit equal instruction is 3 (C21 bits), the program branches to the I-address because the 1 bit is common in the d-character and the B-address character.

The branch if bit equal instruction causes the CPU to take a B-cycle to read out the character at the B-field address onto the B-channel. The d-character is gated to the A-channel. Bits in the characters on the A- and B-channels are compared, but the compare unit is not used. If any bit in the character on the A-channel matches a bit in the character on the B-channel, the CPU executes a program branch. If the characters on the A- and B-channels do not contain at least one common bit, the CPU performs the next instruction in sequence. Timings for the branch if bit equal instruction are identical to those in the branch if character equal instruction shown in Figure 39. The following controls are active in the execution of the branch if bit equal instruction:

Signal	Condition	Logic
1. Op mod register set with d-character.		
Set Op Mod Reg	I ring 11 Time 2 Addr Plus Mod Op Codes LGE	15.38.04
2. Prepare CPU to read out character designated by BAR.		
Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
Set Units Ctrl Latch	Last Instruction Read Out Cy	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02
3. Take a B-cycle to read out B-field character onto B-channel.		
Set B Cy Ctrl	B Cy First Op Codes Last Instruction Read Out Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out BAR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops LG Special A Not Console Inhibit AR Read Out	14.71.31
4. Branch if any bits equal.		
1st Cond A Br Gated	B Cy, 1st Scan (and A and B Ch bits equal), Bit Test Branch Op Code	12.60.03

Branch On Word Mark Or Zone Equal

Formats for the branch on word mark or zone equal instruction are:

Op code	I-address	B-address	d-character
V	xxxxx	xxxxx	see Figure 40
V	xxxxx		
V			

The branch on word mark or zone equal instruction causes the CPU to examine the character located in the storage position designated by the B-address for the zone or word mark combinations specified by the d-character. If conditions for the branch are satisfied, the CPU executes the program branch to the instruction at the I-address.

A one-bit in the d-character examines the B-address character for a word mark. A two-bit in the d-character compares the zone bits in the B-address character with the zone bits in the d-character. A combination of one-

and two-bits in the d-character allows either a word mark or an equal zone bit comparison to initiate a branch to the specified I-address. If the program does not branch to the I-address, the CPU executes the next instruction in sequence. The d-character and the conditions they test are shown in Figure 40.

Significant Bit in d-Character	d-Character	Condition for Branch	Logic
1-bit	1	Branch on word mark	12.60.03
2-bit	2, B, S, or K	Branch on zone equal	12.60.03
1- and 2-bits	T, L, C, or 3	Branch on zone equal or Word Mark	12.60.03

Figure 40. Conditions for Branch on Word Mark or Zone Equal Instruction

When the d-character in the branch on word mark or zone equal instruction contains a two-bit, the CPU takes a B-cycle to read out the character at the B-field address onto the B-channel. The op modifier register is gated to the A-channel. Zone bits in the two characters are compared in the compare unit. The high, low, and equal compare latches are not disturbed as a result of the comparison. If the zone bits in the two characters are equal, the program branches to the I-address. When the d-character in the branch on word mark or zone-equal instruction contains a one-bit, the character located in the position designated by the B-address is read out of storage and examined for a word mark in the B-data register. If the character contains a word mark, the program branches to the I-address. Figure 41 shows CPU action in the branch on word mark or zone-equal instruction. Timings for the instuction are identical to those in the branch if character equals instruction shown in Figure 39.

The following controls are active when the CPU performs the branch on word mark or zone equal instruction:

Signal	Condition	Logic
1. Op mod register set with d-character.		
Set Op Mod Reg	I Ring 11 Time 2 Addr Plus Mod Op Codes LGE	15.38.05
2. Prepare CPU to read out character designated by BAR.		
Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
Set Units Ctrl Latch	Last Instruction Read Out Cy	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02
3. Take a B-cycle to read out B-field character to B-channel.		
Set B Cy Ctrl	B Cy First Op Codes Last Instruction Read Out Cy	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out BAR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops Logic Gate Special A Not Console Inhibit AR Read Out	14.71.31
4. Op mod register containing a 1-bit is switched with B-channel word mark bit to determine branch or no branch condition.		
1st Cond A Br Gated	Op Mod Reg 1 Bit B Ch Word Mark Bit Zone or Word Mark Test Br Op Code	12.60.03
5. Gate the op modifier character to the A-channel.		
Gate Op Mod Reg to A Ch on B Cy	Op Mod to A Ch on B Cy Ops	15.38.02
6. Op mod register containing a 2-bit is combined with compare zone equal to initiate branch.		
Comp Zone Equal 1st Cond A Br Gated	(Compare Matrix) Op Mod Reg 2 bit, Comp Zone Equal, B Cy, 1st Scan, Zone or Word Mark, Test Br Op Code	17.15.04

Questions on Branch Operations

Answers to review questions are in the Appendix.

1. What is the basic difference between conditional and unconditional branch instructions?

2. Where is the I-address in a branch instruction stored during instruction read out?

3. When is the completion of a successful branch operation accomplished?

4. If the system is equipped with the I-O *overlap feature, why should the program first test the overlap-in-process indicator before the branch if* I-O *channel status indicator on instruction is executed?*

5. List the branch instructions that can be chained.

6. When a chained branch instruction specifies only the op code, what I-address, B-address, and d-character is used in the operation?

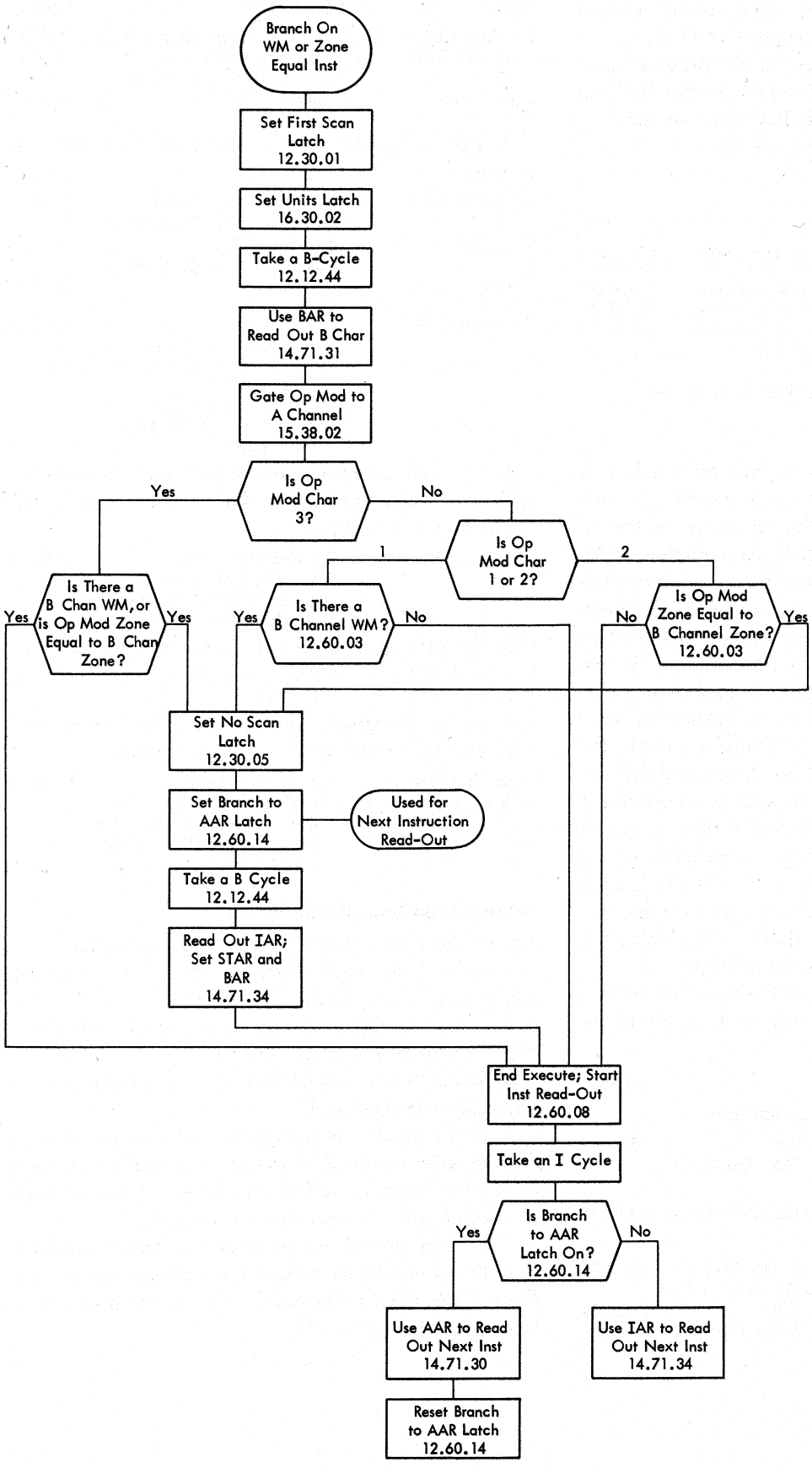


Figure 41. Branch on Word Mark or Zone Equal

Instructions that cause miscellaneous operations are: store address register, set word mark, clear word mark, clear storage, clear storage and branch, halt, halt and branch and no operation. These instructions are used to facilitate programming and prepare storage areas for processing data fields.

Store Address Register Instruction

The format for the store address register instruction is:

OP CODE	C-ADDRESS	d-CHARACTER
G	xxxxx	A, B, E, or F (see Figure 42)

The store address register instruction causes the CPU to store the contents of the address register designated by the d-character in the C-field. The C-address specifies the storage location in which the units position of the register contents are to be stored. Word marks in the C-field have no effect on the operation. An example of the use of the store address register instruction is:

Operation	d-Character
Store contents of A-address register	A
Store contents of B-address register	B
Store contents of E-address register	E
Store contents of F-address register	F

Figure 42. Store Address Register Operations and d-Characters

Before the CPU executes a successful branch operation, the address of the next sequential instruction is stored in the B-address register. If the first instruction after the branch operation is the G (xxxxx) B instruction, causing the contents of the B-address register to be stored, the program can return to the point of interruption by retrieving the stored contents of the BAR.

To execute the store address register instruction, the CPU takes five C-cycles. The C-address register is modified by −1 during each C-cycle. The A-ring advances to A6 to serially gate each position of the selected address register to the address exit channel. Data on the address exit channel are switched to the A-data register, to the A-channel, through the assembly to the storage location addressed by the CAR. The units position of the selected address register contents is stored in the C-address position at A-ring 2 time. The operation ends after the 10,000 position of the selected address register is stored at A-ring 6 time. Figures 43 and 44 show detailed operation in the execution of the store address register instruction.

The following controls are active in the execution of the store address register instruction:

SIGNAL	CONTROL	LOGIC
1. Develop store address registers.		
Store Addr Regs Op Code	Op DCDR 4, 2, 12 Op DCDR 8, A, Not B Op Reg Ars Not C Bit	13.13.07
2. Set CAR during instruction read out.		
Set CAR	Store Addr Regs Op Code 1st Address B Ch Not WM Bit Instruction Read Out Gate	14.71.12
3. Modify by −1.		
G Op Set C Cy Ctrl A	Store Addr Reg Op Code Last Instruction Read Out Cy	12.12.42
Set 1st Scan Ctrl	G Op Set C Cy, Ctrl A	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
4. Set A-ring 1 time.		
Set A Ring 1 Trigger	Store Addr Reg Op Code Last Instruction Read Out Cy LGF	14.70.10
A Ring 1 Time	Set A Ring 1 Trigger	14.70.01
5. Take a C-cycle.		
Set C Cy Ctrl	G Op Set C Cy Ctrl A	12.12.42
C Cy Ctrl	Set C Cy Ctrl Next to Last Logic Gate	12.12.20
6. Address storage with CAR and modify CAR.		
Read Out CAR	Store Addr Regs Op Code C Cy Ctrl Logic Gate Special A	14.71.32
Reset CAR	C Cy Ctrl Logic Gate Early B	14.71.22
Set CAR	C Cy Ctrl, LGB or LGC C Cy Ctrl, LGD or LGE or LGF	14.71.12
7. Advance A-ring.		
A Ring Adv	LGB	14.70.11
8. Read out address register that op mod character indicates.		
Store Addr Reg Ops Req Gate	Store Addr Regs Op Code C Cy LGB to Last Logic Gate	14.71.30
Read Out AAR	Store Addr Reg Ops Req Gate A Symbol Op Modifier	14.71.30
or Read Out BAR	Store Addr Reg Ops Req Gate B Symbol Op Modifier	14.71.31
or Read Out EAR	Store Addr Reg Ops Req Gate E Symbol Op Modifier	14.71.35
or Read Out FAR	Store Addr Reg Ops Req Gate F Symbol Op Modifier	14.71.36
9. Scan out positions of selected address register.		

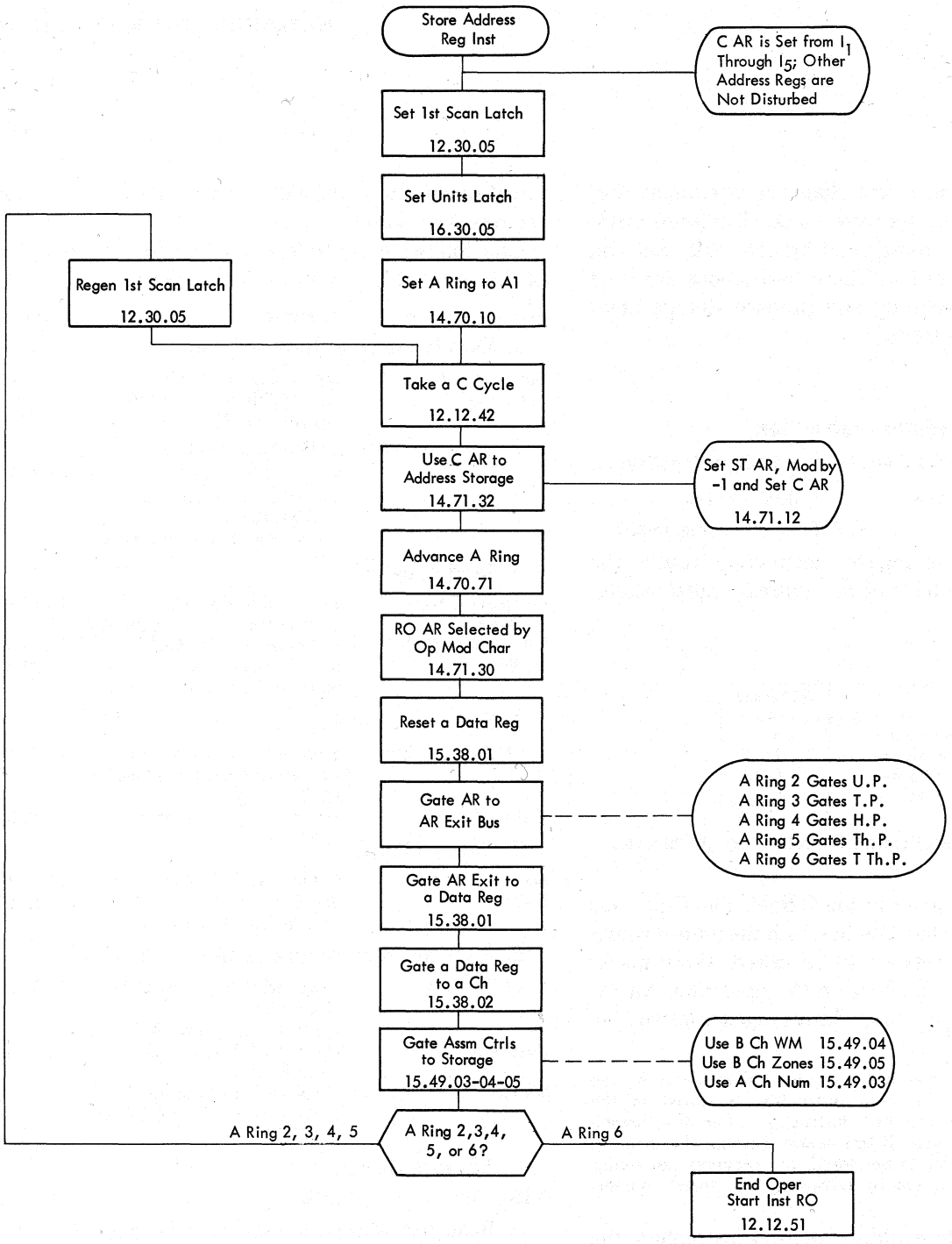


Figure 43. Store Address Register

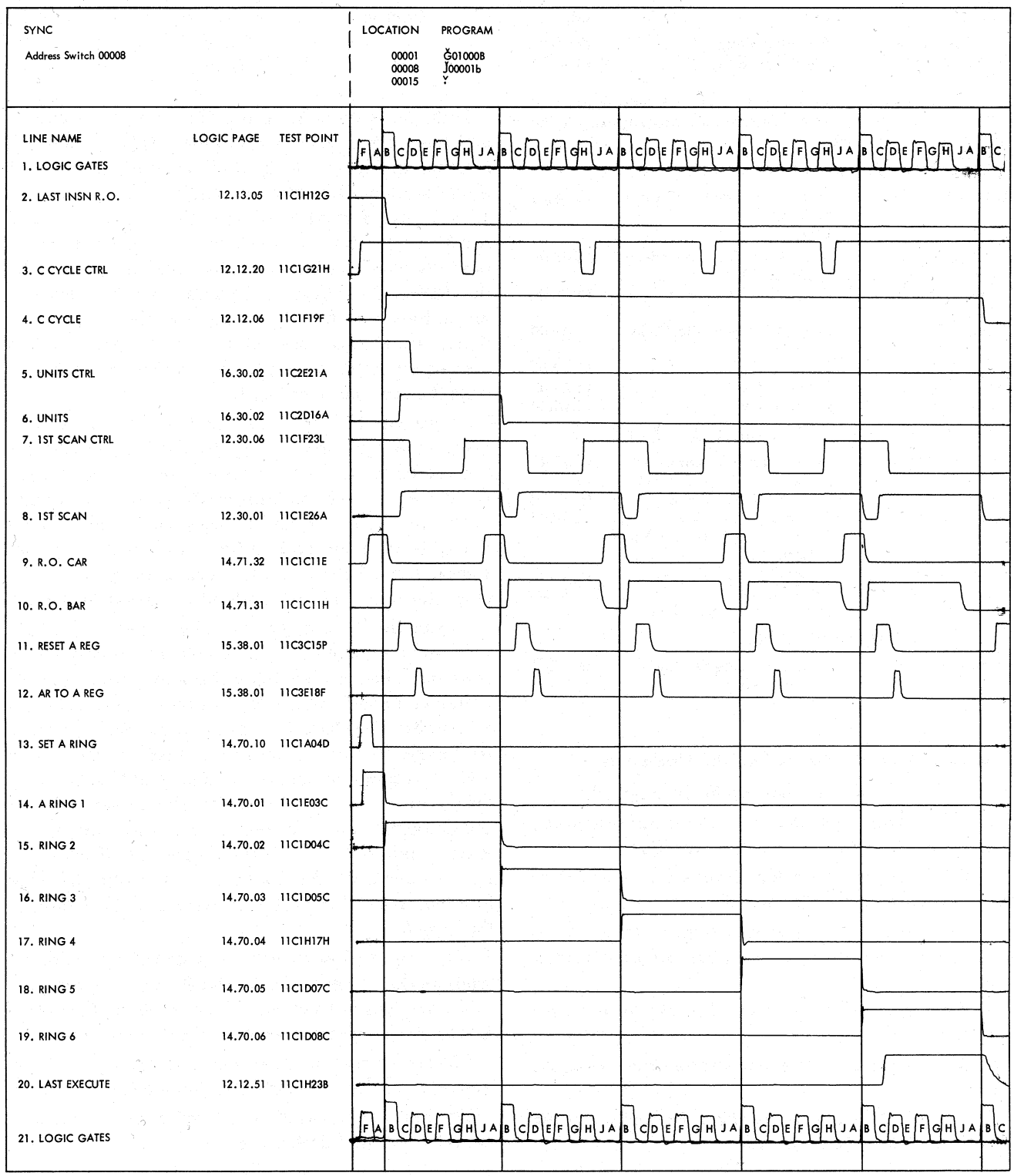


Figure 44. Store Address Register Operation Timings

Signal	Control	Logic
Addr Scnr 2 Pos	A-Ring 2 Time	14.70.02
AR Bus Gtd Out UP Bits	Addr Scnr 2 Pos	14.17.01

A similar circuit is developed to read out tens position at A-Ring 3 time, etc. to put the selected address register bits on the address exit channel.

10. Switch address exit bits to storage.

Reset A Data Reg	Store Addr Reg Op Code C Cy Ctrl LGC	15.38.01
Sw AR Exit Ch to A Reg	Store Addr Reg Op Code C Cy, LGD	15.38.01
Gate A Data Reg to A Ch	C Cy, Not CR Disable	15.38.02
G Op, C Cy	Store Addr Regs Op Code C Cy	15.49.04
Use A Ch Nu	G Op, C Cy	15.49.03
Use B Ch Zones	G Op, C Cy	15.49.05
Use B Ch WM	G Op, C Cy	15.49.04

11. Repeat steps 5-10 if A-ring is 2, 3, 4, or 5.

G Op Set C Cy Ctrl B	Store Addr Regs Op Code A Ring 2, 3, 4, or 5 Time	12.12.42
Regen 1st Scan Ctrl	G Op Set C Cy Ctrl B	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl 1st Scan, Next to Last Logic Gate	12.30.03
Set C Cy Ctrl	G Op Set C Cy B	12.12.42
C Cy Ctrl	Set C Cy Ctrl Next to Last Logic Gate	12.12.20

12. Signal last execute cycle.

Last Execute Cy	Store Addr Reg Op Code A Ring 6 Time 1st Scan	12.12.51

Questions on Store Address Register Operation

Answers to review questions are in the Appendix.

1. What storage position does the C-address in the store address register instruction specify?

2. What purpose does the d-character in the store address register instruction serve?

3. List the address registers whose contents can be stored by executing the store address register instruction.

4. What effect do C-field word marks have on the operation?

5. When does the store address register operation end?

6. When is the character in the units position of the selected address register stored?

Set Word Marks Instruction

The set word mark instruction causes the CPU to store word marks in designated core storage locations. Data in the addressed storage position are not disturbed.

The set word mark instruction can have one of the following formats.

Op code	A-address	B-address
;	xxxxx	xxxxx
;	xxxxx	
;		

If the set word mark instruction contains A- and B-addresses, a word mark is set in the specified A-address location and in the designated B-address position. If the set word mark instruction contains only one address (A-address), a word mark is set twice in the specified A-address location. If the address instruction is indexed, a word mark is set at the location specified by the indexed A-address. If the instruction is given with no address specified (a no-address chained instruction), word marks are set in the address locations designated by the A- and B-address registers (contents from the previous operation).

To execute the set word mark instruction, the CPU takes an A-cycle and a B-cycle. During the A-cycle, the A-address character is read from storage to the B-channel. A word mark bit is added to the character in the assembly unit, and a check bit is added or removed to maintain odd parity in the eight bit planes (seven BCD planes and the word mark plane). The character and the word mark bit are gated to storage.

During the B-cycle, the character at the B-address is read from storage and gated onto the B-channel. A word mark bit is added to the character in the assembly unit, and a check bit is added or removed to maintain odd parity. The character and the word mark are gated to storage. If only one address is specified, the A-data address is stored in both the A- and B-address registers during instruction read out. Therefore, when execute phase begins for a single address instruction, the A-data address is in both the A- and B-address registers, causing a word mark to be set twice in the same location: once during the A-cycle and once during the B-cycle.

If the set word mark instruction contains no address, word marks are set in the A- and B-addresses specified in the previous instruction. The CPU executes the operation as if the instruction contained two addresses.

Figures 45 and 46 show detailed operation in the execution of the set word mark instruction.

The following controls are active in the execution of the set word mark instruction:

Signal	Control	Logic

1. Set modifier ctrls to −1.

Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
1st Scan	1st Scan Ctrl LGC	12.30.01
Addr Mod Set to −1	1st Scan Ctrl	14.71.41

2. Set the units latch.

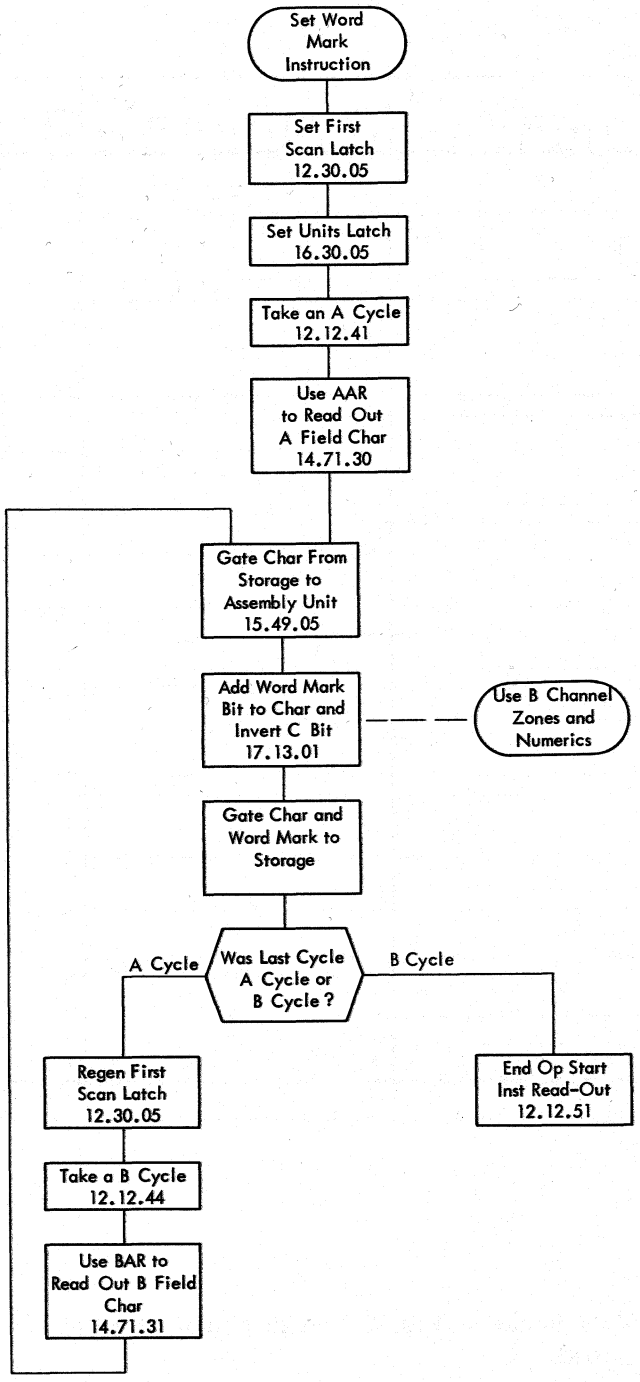


Figure 45. Set Word Marks

Signal	Control	Logic
Set Units Ctrl Latch	Last Instruction Read Out Cy	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02

3. Initiate an A-cycle, and read out first A-field character.

Control	Logic Signal	
Set A Cy Ctrl	A Cy First Op Codes Last Instruction Read Out Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl Next to Last Logic Gate	12.12.01
A Cy	A Cy Ctrl LGB	12.12.01
Read Out AAR	A Cy Ctrl Read Out AAR on A Cy Ops Logic Gate Special A	14.71.30
Set Mem AR Gated	LGA, 2nd Clock Pulse	14.17.16

4. Gate character through assembly to storage; set word mark, and add or remove C-bit to maintain odd parity (15.50.07).

Use B Ch Zones; use B Ch Nu	Word Mark Op Code A or B Cy	15.49.05
Set WM	Set WM Op Code A or B Cy	17.13.01
Asm Ch WM Bits	Set WM	15.50.08
Load Memory	Adr B Cy WM Op Code	12.50.01

5. Control A-cycle length.

Word Mark Op A Cy	Word Mark Op Codes A Cy	12.12.44
Stop at J	Word Mark Op A Cy	12.12.32

6. Regen 1st scan latch.

Regen 1st Scan Ctrl	Word Mark Op A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl Next to Last Logic Gate 1st Scan	12.30.03
1st Scan	1st Scan Ctrl LGC	12.30.01
Addr Mod Set to −1	1st Scan Ctrl	14.71.41

7. Initiate a B-cycle, and read out first B-field character.

Set B Cy Ctrl	A Cy Word Mark Op Codes	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out BAR	Logic Gate Special A B Cy Ctrl 1st, 2nd, or 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd Clock Pulse	14.17.16

8. Gate character through assembly to storage; repeat step 4.

9. Control B-cycle length; end operation.

Word Mark Op B Cy	Word Mark Op Codes B Cy	12.12.51
Stop at J	Word Mark Op B Cy	12.12.32

10. Initiate instruction read out.

Last Execute Cy	Word Mark Op B Cy	12.12.50

Questions on Set Word Marks Operation

Answers to review questions are in the Appendix.

1. What is the purpose of the set word marks instruction?

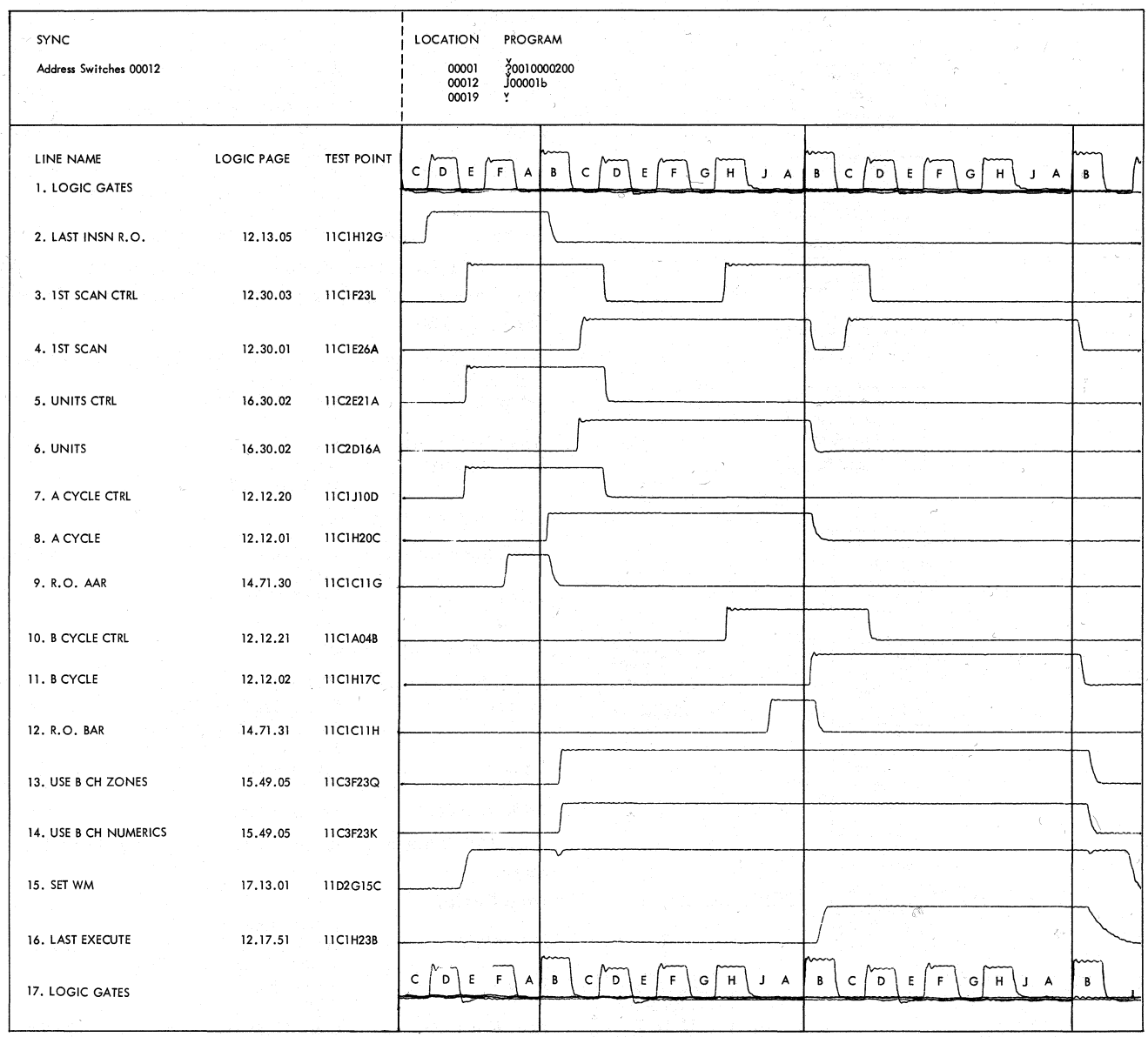


Figure 46. Set Word Mark Operation Timings

2. Is the data character in the specified storage location altered when the set word marks operation is performed?

3. If the set word marks instruction contains only one address, what action occurs?

4. When a word mark is added to a character, how is parity maintained?

Clear Word Mark Instruction

The clear word mark instruction causes the CPU to clear word marks in designated core storage locations. Data in the addressed storage position are not disturbed.

The clear word mark instruction can have one of the following formats.

OP CODE	A-ADDRESS	B-ADDRESS
⊓	XXXXX	XXXXX
⊓	XXXXX	
⊓		

If the clear word mark instruction contains A- and B-addresses, word marks are cleared from the specified A-address location and the designated B-address position. If the clear word mark instruction contains

only one address (A-address), the word mark is cleared from the specified A-address location. If the instruction is given with no address specified (a no-address chained instruction), word marks are cleared from the address locations designated by the A- and B-address registers (contents from the previous operation).

To execute the clear word mark instruction, the CPU takes an A-cycle and a B-cycle. During the A-cycle, the A-address character is read from storage and gated onto the B-channel. The word mark bit is stripped from the character in the assembly unit, and a check bit is added or deleted to maintain parity in the eight bit planes (seven BCD planes and the word mark plane). The character (without the word mark) is gated to storage.

During the B-cycle, the character at the B-address is read from storage and gated onto the B-channel. The word mark bit is stripped from the character in the assembly unit, and a check bit is added or removed to maintain odd parity. The character (without the word mark) is gated to storage. If only one address is specified, the A-data address is stored in both the A- and B-address registers during instruction read out. Therefore, when execute phase begins for a single address instruction, the A-data address is in both the A- and B-address registers, causing the CPU to execute the procedure to strip the word mark bit twice from the same location, once during the A-cycle and once during the B-cycle.

If the clear word mark instruction contains no address, word mark bits are removed from the A- and B-addresses specified in the previous instruction. The CPU executes the operation as if the instruction contained two addresses.

Figure 47 shows CPU data flow in the execution of the clear word mark instruction.

The following controls are active in the execution of the clear word mark instruction:

SIGNAL	CONTROL	LOGIC
1. Set modifier controls to −1.		
Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
1st Scan	1st Scan Ctrl LGC	12.30.01
Addr Mod Set to −1	1st Scan Ctrl	14.71.41
2. Set the units latch.		
Set Units Ctrl Latch	Last Instruction Read Out Cy	16.30.01
Units Ctrl Latch	Set Units Ctrl Latch Next to Last Logic Gate	16.30.02
Units Latch	Units Ctrl Latch LGC	16.30.02

3. Initiate an A-cycle, and read out first A-field character.

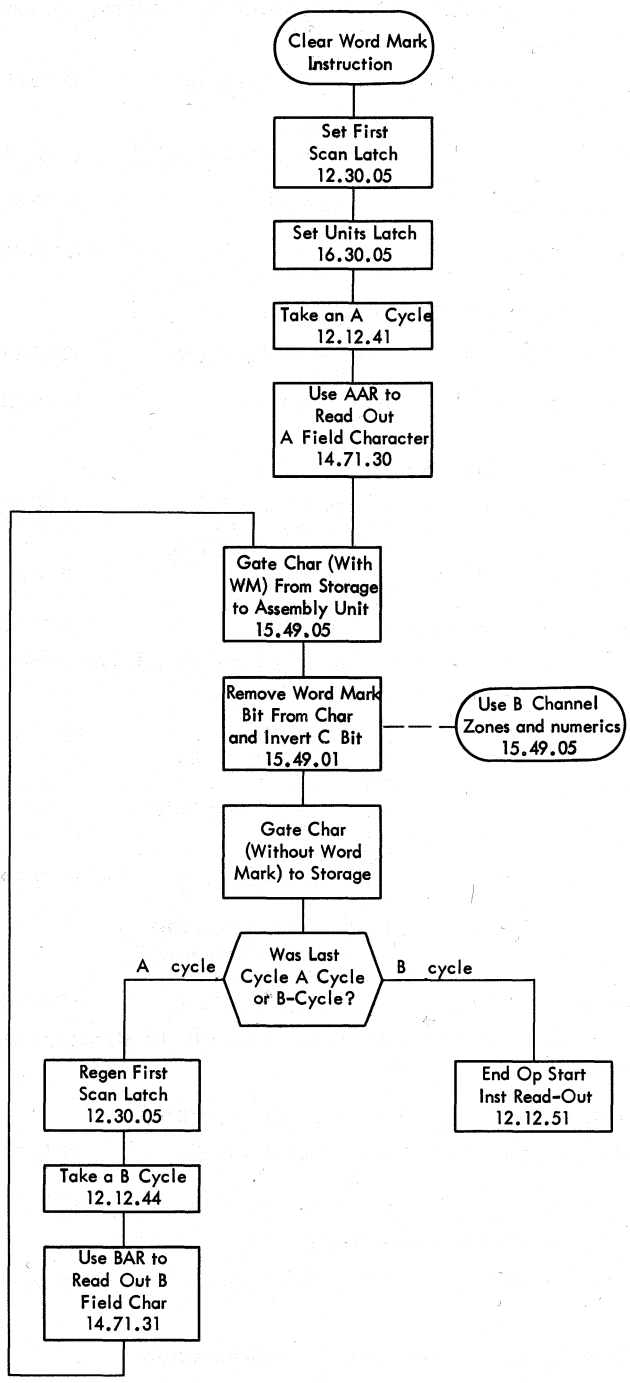


Figure 47. Clear Word Mark

SIGNAL	CONTROL	LOGIC
Set A Cy Ctrl	A Cy First Op Codes Last Instruction Read Out Cy	12.12.41
A Cy Ctrl	Set A Cy Ctrl Next to Last Logic Gate	12.12.20
A Cy	A Cy Ctrl LGB	12.12.01
Read Out AAR	A Cy Ctrl Read Out AAR on A Cy Ops	14.71.30
Set Mem AR Gated	Logic Gate Special A LGA, 2nd Clock Pulse	14.17.16

Signal	Control	Logic
4. Gate character through assembly to storage; clear word mark.		
Use B Ch Zones; use B Ch Nu	Word Mark Op Codes A or B Cy	15.49.05
Use No WM	Clear Word Mark Op Code A or B Cy	15.49.01
Assembly Ch Not WM Bit	Use No WM	15.50.08
Load Memory	A or B Cy Word Mark Op Codes	12.50.01
5. Control A-cycle length.		
Word Mark Op A Cy	Word Mark Op Codes, A Cy	12.12.44
Stop at J	Word Mark Op A Cy	12.12.32
6. Regen 1st scan latch.		
Regen 1st Scan Ctrl	Word Mark Op A Cy	12.30.05
1st Scan Ctrl	Regen 1st Scan Ctrl Next to Last Logic Gate	12.30.03
1st Scan	1st Scan Ctrl LGC	12.30.01
Addr Mod Set to −1	1st Scan Ctrl	14.71.41
7. Initiate a B-cycle and read out first B-field character.		
Set B Cy Ctrl	A Cy Word Mark Op Codes	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Read Out BAR	Logic Gate Special A B Cy Ctrl 1st, 2nd, or 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops	14.71.31
Set Mem AR Gated	LGA, 2nd Clock Pulse	14.17.16
8. Gate character through assembly to storage; repeat step 4.		
9. Control B-cycle length; end operation.		
Word Mark Op B Cy	Word Mark Op Codes B Cy	12.12.51
Stop at J	Word Mark Op B Cy	12.12.32
10. Initiate instruction read out.		
Last Execute Cy	Word Mark Op B Cy	12.12.50

Questions on Clear Word Marks Operation

Answers to review questions are in the Appendix.

1. What is the purpose of the clear word marks instruction?

2. Is the data character in the specified storage location altered when the clear word marks operation is performed?

3. If the clear word marks instruction contains only one address, what action occurs?

4. When a word mark is stripped from a character, how is parity maintained?

Clear Storage Instruction

The formats for the clear storage instruction are:

Op code	B-address
⌄ / ⌄	xxxxx

The clear storage instruction causes the CPU to clear characters and word marks from an area in storage, right to left, from the specified B-address through the nearest hundreds position. The cleared area is set to blanks (C-bits). The number of positions cleared can be calculated by adding 1 to the last two digits in the B-address; for example, the instruction ⌄/12999 causes the CPU to clear storage from positions 12999 through 12000; 100 positions are set to blanks (99 + 1 = 100). The instruction ⌄/15000 causes the CPU to clear storage location 15000; one position is cleared (00 + 1 =1).

A clear storage instruction with a B-address conditions the address-doubler op code line, and the B-address in the instruction is loaded in the AAR and BAR during instruction read out time. To execute the operation, the CPU takes a series of B-cycles. During each B-cycle, the storage location designated by the address in the BAR is cleared, and the address in the BAR is reduced by one. Storage locations are cleared in successive B-cycles to the nearest even hundreds position. The −1 condition ON at logic gate D time, indicating a borrow one from the hundreds position, defines the even hundreds position; the even hundreds latch is set, and execute phase ends. At the end of the operation, the BAR contains 1 minus the address of the last storage location cleared (xxx99). The AAR, unchanged after instruction read out, contains the original B-address specified in the instruction.

If the clear storage instruction does not contain a B-address (a no-address chained instruction), the contents of the BAR from the previous operation are used as the B-address, and the CPU takes B-cycles to perform the operation in the normal manner. In this case, however, the AAR is not loaded during instruction read out time and is unchanged during the execution of the clear storage instruction.

Figure 48 shows CPU operation in the execution of the clear storage instruction.

The following controls are active in the execution of the clear storage instruction:

Signal	Control	Logic
1. Initiate last instruction read out cycle; set no branch latch.		
Last Instruction Read Out Cy	I-Ring 6 Time No D Cy at I-Ring 6 Ops B Ch WM Bit	12.13.05
No Branch Conditions	Clear Op Code I-Ring 1 or 6 Time Last Instruction Read Out Cy	12.60.08

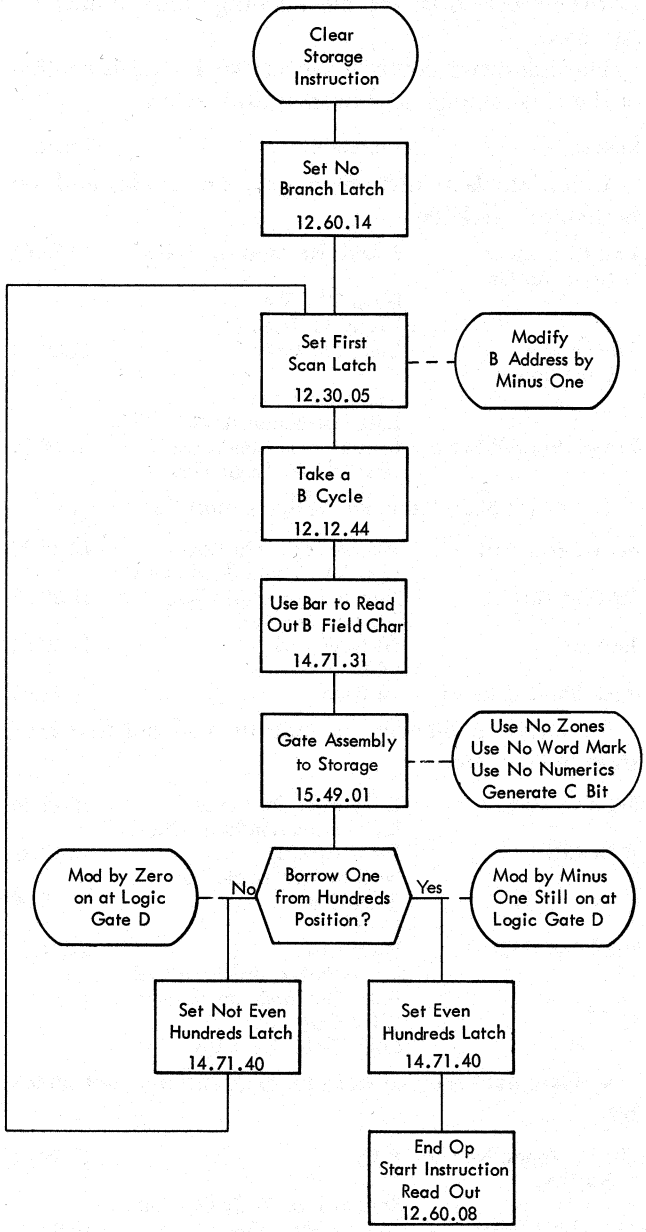


Figure 48. Clear Storage

SIGNAL	CONTROL	LOGIC
No Branch Latch	No Branch Conditions LGZ	12.60.14

2. Set 1st scan latch; set address modifier to −1.

Set 1st Scan Ctrl	1st Scan First Op Codes Last Instruction Read Out Cy	12.30.05
1st Scan Ctrl	Next to Last Logic Gate Set 1st Scan Ctrl	12.30.03
1st Scan	1st Scan Ctrl, LGC	12.30.01
Addr Mod Set to −1	1st Scan	14.71.41

3. Initiate a B-cycle; use BAR to read out first character from storage.

Set B Cy Control	B Cy First Op Codes Last Instruction Read Out Cy	12.12.44
B Cy Control	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl, LGB	12.12.02
Read Out BAR	B Cy Ctrl 1st + 2nd + 3rd Scan Ctrl Read Out BAR on Scan B Cy Ops Logic Gate Special A	14.71.31

4. Gate assembly to storage; clear storage; set check-bit.

Use No Zones, No Nu, No WM	B Cy Stop at F on B Cy Op Codes	15.49.01
Assembly Ch Nu C Bit	Use No Numerics	15.50.09
Assembly Ch Zone C Bit	Use No Zones	15.50.10
Assembly Ch Not WM Bit	Use No WM	15.50.08
Assembly Ch C Char Bit	Assembly Ch Nu C Bit Assembly Ch Zone C Bit Not Assembly Ch WM Bit	15.50.07
Load Memory	Clear Op Code 1st Scan B Cy	12.50.01

5. Test for even or not even hundreds address; set appropriate latch.

Not Even Hundreds Addr	Zero Latch 2nd Clock Pulse, LGD	14.71.40
Clear Op Take 1st Scan	Not Even Hundreds Addr Clear Op Code B Cy 1st Scan	12.60.04
Set 1st Scan Ctrl	Clear Op Take 1st Scan	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl, Next to Last LG	12.30.03
Addr Mod Set to −1	1st Scan Ctrl	14.71.41

6. Take another B-cycle; read out next core storage character.

7. Repeat step 4.

8. Test for an even hundreds address; if not an even hundreds address, continue to take B-cycles and clear storage until an even hundreds address is detected; when even hundreds address is detected, set even hundreds address latch.

Borrow Latch	Minus One 2-8 Line, 2nd Clock Pulse	14.30.08
Even Hundreds Addr	Borrow Latch, 2nd Clock Pulse, LGD	14.71.40

9. End execute cycle; initiate I-cycle.

Last Execute Cy * Br Cnd	Even Hundreds Addr No Branch Latch B Cy 1st Scan Clear Op Code	12.60.08
Last Execute Cy	Last Execute Cy * Br Cnd	12.12.51
Set I Cy Ctrl	Last Execute Cy	12.13.02
I Cy Ctrl	Set I Cy Ctrl Next to Last Logic Gate	12.12.23

10. Use IAR to read out next instruction.

Questions on Clear Storage Operation

Answers to review questions are in the Appendix.

1. If the B-address in the clear storage instruction is 09050, what is the number of storage positions cleared in the clear storage operation?

2. If the clear storage instruction does not contain a B-address, is the AAR *loaded during instruction read out?*

3. If the clear storage instruction contains a B-address, what address is stored in the AAR *when the clear storage operation ends?*

4. When is the clear storage operation terminated?

Clear Storage and Branch Instruction

The format for the clear storage and branch instruction is:

OP CODE	I-ADDRESS	B-ADDRESS
/	xxxxx	xxxxx

The clear storage and branch instruction causes the CPU to clear characters and word marks from an area in storage, right to left, from the specified B-address through the nearest hundreds position and branch to the instruction at the I-address after the clear operation. The clear storage and branch instruction combines the clear storage and unconditional branch instructions.

During instruction read out, the I-address in the instruction is stored in the AAR; the B-address is stored in the BAR. Because the last instruction read out cycle occurs at I-ring 11 time, the branch to AAR latch is set.

To clear storage, the CPU takes a series of B-cycles. During each B-cycle, the storage location designated by the address in the BAR is cleared, and the address in the BAR is reduced by one. Storage locations are cleared in successive B-cycles to the nearest even hundreds position. The −1 condition ON at logic gate D time, indicating a borrow one from the hundreds position, defines the even hundreds position; the even hundreds latch is set, and execute phase ends. At the end of the clear operation, the BAR contains 1 minus the address of the last storage location cleared (xxx99).

Because the branch to AAR and even hundreds latches are set, the no-scan latch is set, causing a modify by 0 condition in the modifier. The CPU takes a B-cycle. The IAR, which contains the address of the next sequential instruction, reads out to the STAR, through the modifier to the BAR. The completion of the branch operation is accomplished during the subsequent instruction read out operation at which time the AAR (containing the address of the next instruction to be executed) sets the STAR.

Figures 49 and 50 show CPU timings and operation in the execution of the clear storage and branch instruction.

The following controls are active in the execution of the clear storage and branch instruction:

SIGNAL	CONTROL	LOGIC
1. Initiate last instruction read out cycle, and set branch to AAR latch.		
Last Instruction Read Out Cy	2 Addr No Mod Op Codes	12.13.05
	B Ch WM Bit	
	I-Ring 11 Time	
Branch to A Conditions	Clear Op Code	12.60.04
	I-Ring 11 Time	
	Last Instruction Read Out Cy	
Branch to AAR Latch	Branch to A Conditions	12.60.14
	Next to Last Logic Gate	
2. Set 1st Scan latch; set address modifier to −1.		
Set 1st Scan Ctrl	1st Scan First Op Codes	12.30.05
	Last Instruction Read Out Cy	
1st Scan Ctrl	Next to Last Logic Gate	12.30.03
	Set 1st Scan Ctrl	
1st Scan	1st Scan Ctrl	12.30.01
	LGC	
Addr Mod Set to −1	1st Scan	14.71.41
3. Initiate a B-cycle; use BAR to read out first core storage character.		
Set B Cy Ctrl	B Cy First Op Codes	12.12.44
	Last Instruction Read Out Cy	
B Cy Ctrl	Set B Cy Ctrl	12.12.21
	Next to Last Logic Gate	
B Cy	B Cy Ctrl	12.12.02
	LGB	
Read Out BAR	B Cy Ctrl	14.71.31
	1st + 2nd + 3rd Scan Ctrl	
	Read Out BAR on Scan B Cy Ops	
	Logic Gate Special A	
4. Gate assembly to storage; clear storage; set check bit.		
Use No Zones, No Nu, No WM	B Cy	15.49.01
	Stop at F on B Cy Op Codes	
Assembly Ch Nu C Bit	Use No Numerics	15.50.09
Assembly Ch Zone C Bit	Use No Zones	15.50.10
Assembly Ch Not WM Bit	Use No WM	15.50.08
Assembly Ch C Char Bit	Assembly Ch Nu C Bit	15.50.07
	Assembly Ch Zone C Bit	
	Not Assembly Ch WM Bit	
Load Memory	Clear Op Code	12.50.01
	1st Scan	
	B Cy	
5. Test for even and not even hundreds address; set appropriate latch.		
Not Even Hundreds Addr	Zero Latch	14.71.40
	2nd Clock Pulse, LGD	
Clear Op Take 1st Scan	Not Even Hundreds Addr	12.60.04

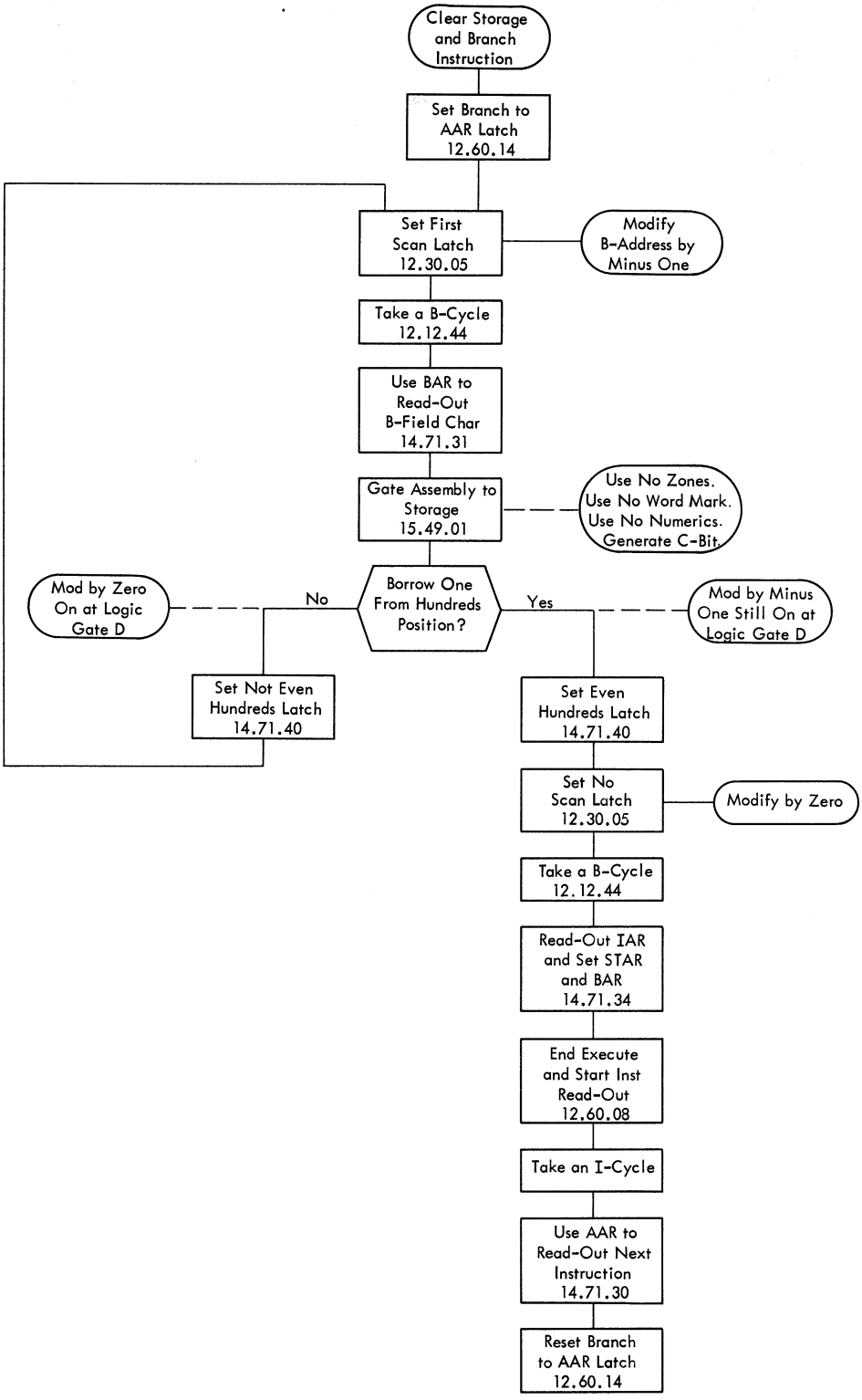


Figure 49. Clear Storage and Branch

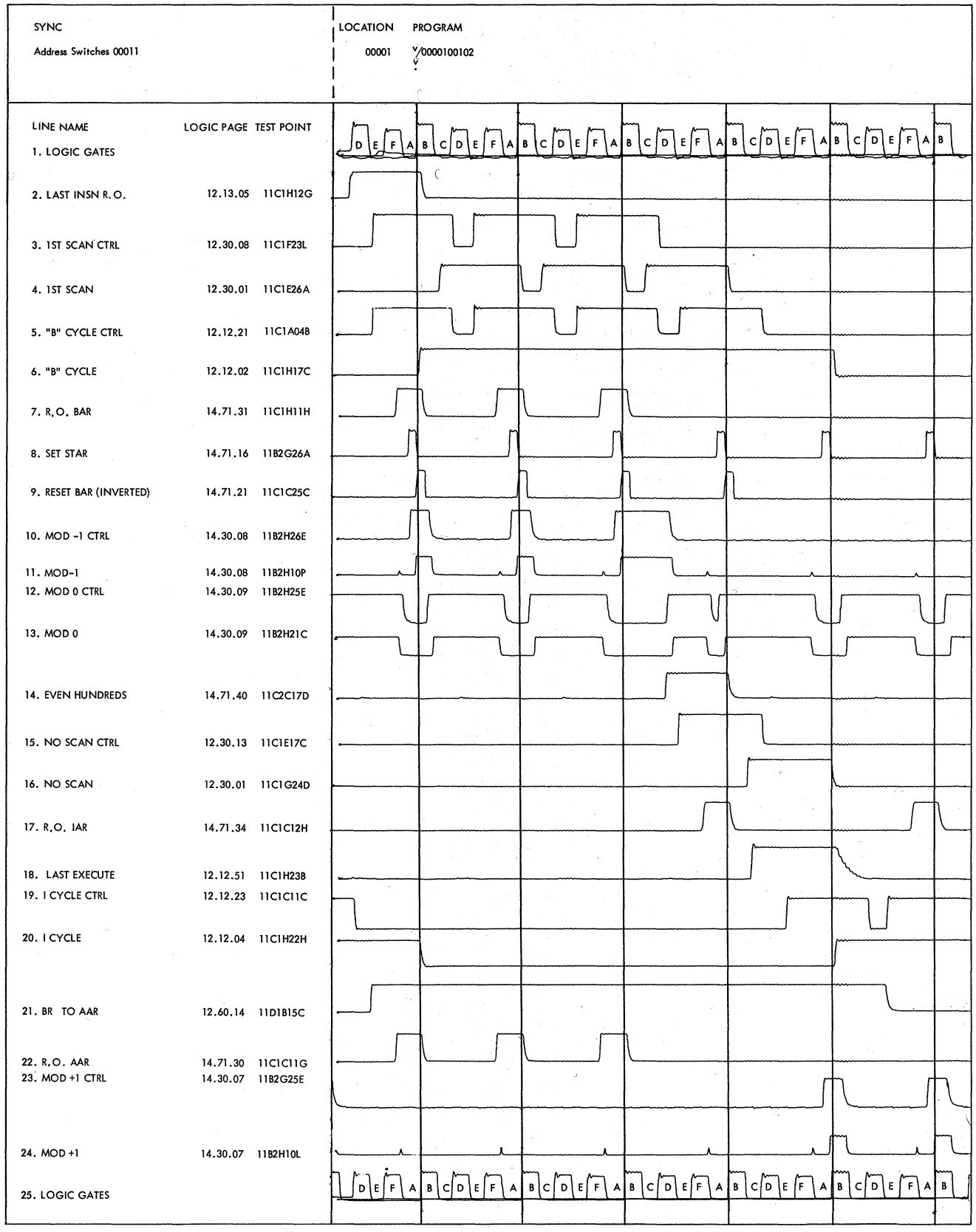


Figure 50. Clear Storage and Branch Operation Timings

Signal	Control	Logic
	Clear Op Code B Cy 1st Scan	
Set 1st Scan Ctrl	Clear Op Take 1st Scan	12.30.05
1st Scan Ctrl	Set 1st Scan Ctrl Next to Last Logic Gate	12.30.03
Addr Mod Set to −1	1st Scan Ctrl	14.71.41

6. Take another B-cycle; read out next character from core storage.

7. Repeat step 4.

8. Test for an even hundreds address; if not an even hundreds address, continue to take B-cycles and clear storage until an even hundreds address is detected; when even hundreds address is found, set even hundreds address latch.

Borrow Latch	Minus One 2-8 Line, 2nd Clock Pulse	14.30.08
Even Hundreds Addr	Borrow Latch 2nd Clock Pulse, LGD	14.71.40

9. Initiate a branch operation.

Set No Scan Ctrl * Br Ops	Clear Op Code B Cy 1st Scan Even Hundreds Addr Branch to AAR Latch	12.60.04
Set B Cy Ctrl * Br Ops	Clear Op Code B Cy 1st Scan Even Hundreds Addr Branch to AAR Latch	12.60.04
Set B Cy Ctrl	Set B Cy Ctrl * Br Ops	12.12.44
B Cy Ctrl	Set B Cy Ctrl Next to Last Logic Gate	12.12.21
B Cy	B Cy Ctrl LGB	12.12.02
Set No Scan Ctrl	Set No Scan Ctrl * Branch Ops	12.30.05
No Scan	Set No Scan Ctrl Next to Last Logic Gate	12.30.03
Addr Mod Set to 0	No Scan	14.71.41

10. Read Out IAR to STAR and BAR.

Read Out IAR	B Cy Ctrl No Scan Ctrl Logic Gate Special A	14.71.34
Mem AR Set	LGA or LGR, 2nd Clock Pulse	14.17.16
Set BAR	LGB or LGC B, E, or F Cy Ctrl	

11. End execute, and initiate I-cycle.

Last Execute Cy * Br Cnd	Branch Type Op Codes No Scan B Cy	12.60.08
Last Execute Cy	Last Execute Cy * Br Cnd	12.12.51
Set I Cy Ctrl	Last Execute Cy	12.13.02

12. Use AAR to read out next instruction.

Read Out AAR	I Cy Ctrl Branch to AAR Latch Logic Gate Special A	14.71.30

Questions on Clear Storage and Branch Operation

Answers to review questions are in the Appendix.

1. How do the clear storage and clear storage and branch operations differ?

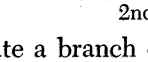
2. Where is the I-address in the clear storage and branch instruction stored during instruction read out?

3. Is the branch operation that the clear storage and branch operation causes a conditional or unconditional branch?

4. When is the completion of the branch operation accomplished?

Halt Instruction

The format for the halt instruction is:



The halt instruction causes the logic clock in the CPU to stop. Pressing the start key starts system operation with the next sequential instruction. Because the halt and halt and branch instructions have the same operational code (.), a word mark must be present in the storage location immediately to the right of the halt op code to distinguish the halt instruction from the halt and branch instruction.

The last instruction read-out cycle (determined by the presence of a B-channel word mark at I-ring 1 time) causes a simultaneous last execute cycle; this condition sets the stop latch. When the stop latch is set, the logic clock stops, initiating a console stop print-out operation if:

1. The print control switch is not set to inhibit; and,
2. The mode switch is in neither the display nor alter position.

If the console stop print-out operation is complete when the start key is pressed, the stop latch resets, allowing the logic clock to run again; the CPU executes the next sequential instruction. If the start key is pressed before the console stop print-out operation ends, no action occurs until the print-out stops.

Figure 51 shows CPU operation in the execution of the halt instruction.

The following controls are active when the CPU performs the halt operation:

Signal	Control	Logic

1. Set the no branch latch, and initiate the last execute cycle.

Last Instruction Read Out Cy	No C or D Cy Op Codes I Ring 1 time B Ch WM Bit	12.13.05
No Branch Cnds	I Ring 1 time Stop or Branch Op Code Last Instruction Read Out Cy	12.60.08

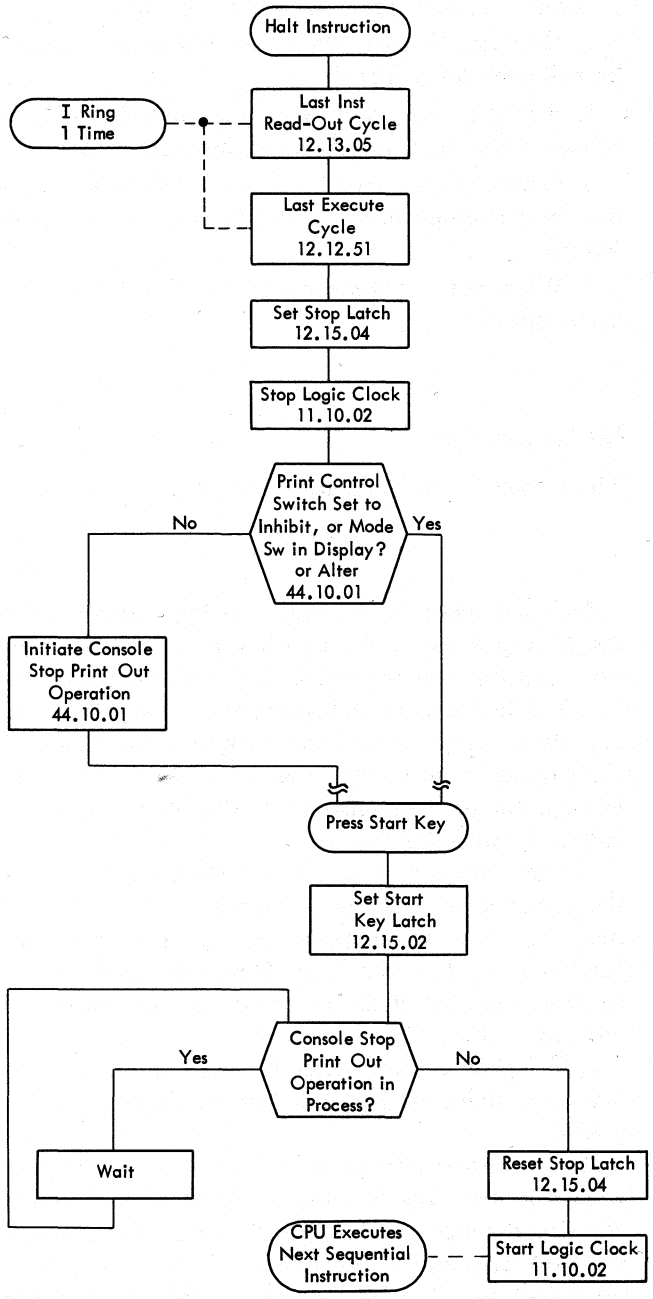


Figure 51. Halt

Signal	Control	Logic
No Branch Latch	No Branch Cnds Next to Last Logic Gate	12.60.14
Last Execute Cy * Br Cnd	Last Instruction Read Out Cy I Ring 1 time Stop or Br Op Code	12.60.08
Last Execute Cy	Last Executive Cy * Br Cnd	12.12.51

2. Set the stop latch, and stop the logic clock.

Stop Latch	LGZ, 2nd Clock Pulse Last Execute Cy Stop or Branch Op Code	12.15.04
Clock Stopped	Stop Latch	11.10.02

Signal	Control	Logic

3. Initiate stop print-out operation.

Console Stop Print Out Op	Stop or Branch Op Code Last Execute Cy	12.15.04
Console Enable Stop Print Out	Clock Stopped Console Stop Print Out Cnd	13.42.10

Questions on Halt Operation

Answers to review questions are in the Appendix.

1. The halt instruction causes the logic clock in the CPU to stop; when system operation resumes, what instruction is executed first?

2. How is the halt instruction distinguished from the halt and branch instruction during instruction read out?

3. When is the console stop print-out operation started?

4. Does the logic clock start automatically when the stop print-out operation ends?

Halt and Branch Instruction

The format for the halt and branch instruction is:

Op code	I-address
•	xxxxx

The halt and branch instruction causes the logic clock in the CPU to stop. When the start key is pressed, system operation resumes with the instruction at the designated I-address.

At the end of the instruction read-out cycle, the branch to AAR latch is set. By taking a branch type B-cycle, the AAR is loaded with the I-address in the instruction. The IAR, which contains the address of the next sequential instruction, reads out to the STAR, through the modifier (previously set to modify by 0 condition) to the BAR. Last execute cycle becomes active at this time. The completion of the branch operation is accomplished during the instruction read-out cycle (initiated when the start key is depressed after the stop) at which time the AAR (containing the address of the next instruction to be executed) sets the STAR.

The stop latch is set during last execute cycle, causing the logic clock to stop. A console stop print-out operation is initiated if:

1. The print control switch is not set to inhibit, and
2. The mode switch is in neither the display nor alter position.

If the console stop print-out operation is complete when the start key is pressed, the stop latch resets, allowing the logic clock to run again. The CPU completes execution of the branch instruction and performs the operation specified by the instruction at the I-address of the halt and branch instruction. If the start

key is pressed before the console stop print-out operation ends, no action occurs until the print out stops.

CPU operation in the execution of the halt and branch instruction is shown in Figure 52.

The following controls are active when the CPU performs the halt and branch operation:

SIGNAL	CONTROL	LOGIC
1. Set the branch to AAR latch.		
2nd Cnd A Branch Gated	Last Instruction Read Out Cy Stop or Branch Op Code I Ring 6 Time	12.60.02
Branch to A Cnds	2nd Cnd A Branch Gated	12.60.04
Branch to AAR Latch	Branch to A Cnds Next to Last Logic Gate	12.60.14
2. Initiate a B-cycle; set address modifier to 0.		
Set B Cy Ctrl * Br Ops	2nd Cnd A Branch Gated	12.60.04
Set B Cy Ctrl	Set B Cy Ctrl Br Ops	12.12.44
B Cy Ctrl	B Cy Ctrl LGB	12.12.02
Set No Scan Ctrl	Set No Scan Ctrl * Br Ops	12.30.05
No Scan	Set No Scan Ctrl Next to Last Logic Gate	12.30.03
Addr Mod Set to 0	No Scan	14.71.41
3. Read out IAR to STAR and BAR.		
Read Out IAR	No Scan Ctrl B Cy Ctrl Logic Gate Special A	14.71.34
Mem AR Set	LGA or LGR, 2nd Clock Pulse	14.17.16
Set BAR	B, E, or F Cy Ctrl LGB or LGC	14.71.11
4. Terminate execute phase.		
Last Execute Cy * Br Cnd	No Scan Branch Type Op Codes B Cy	12.60.08
Last Execute Cy	Last Execute Cy * Br Cnd	12.12.51
5. Set the stop latch, and stop the logic clock.		
Stop Latch	LGZ, 2nd Clock Pulse Last Execute Cy Stop or Branch Op Code	12.15.04
Clock Stopped	Stop Latch	11.10.02
6. Stop the logic clock at logic gate F.		
Stop at F on B Cy Op Codes	Stop or Branch Op Codes	12.14.08
Stop at F	Stop at F on B Cy Op Codes B Cy	12.12.30
Last Logic Gate	Stop at F LGF	12.12.31
7. Initiate stop print-out operation.		
Console Stop Print Out Cnd	Stop or Branch Op Code Last Execute Cy	12.15.04
Console Enable Stop Print Out	Clock Stopped Console Stop Print Out Cnd	13.42.10 13.42.10

Questions on Halt and Branch Operation

Answers to review questions are in the Appendix.

1. The halt and branch instruction causes the logic clock in the CPU to stop. When the system resumes operation, what instruction is executed first?

2. Is the branch to AAR latch set, initiating the branch operation, before the logic clock stops or after the logic clock starts again?

3. If the start key is pressed before the stop-print out operation is complete, is the stop-print out operation terminated immediately?

4. Is the branch operation, which the halt and branch instruction designates, an unconditional branch or a conditional branch?

No Operation Instruction

No format for the no operation instruction is:

OP CODE
N (any number of characters can follow the N op code if none of the characters contain a word mark bit)

The N op code performs no operation; it can be substituted for the op code in any instruction to make that instruction ineffective.

During I-op time, the N op code is set in the operation register, preventing the I-ring from advancing. Characters following the N op code are read out in successive I-cycles and ignored until a B-channel word mark is detected. Therefore, the no operation instruction causes the CPU to skip one or several instructions as determined by the location of the first B-channel word mark after the N op code is decoded. The B-channel word mark causes the op code of the next instruction to be loaded in the op register, and instruction read out continues. The I-ring, clamped at I-op time during execution of the no-op instruction, is allowed to advance in the normal manner.

CPU action in the execution of the no operation instruction is shown in Figures 53 and 54.

The following controls are active when the CPU executes the no operation instruction:

SIGNAL	CONTROL	LOGIC
1. Decode the N op code, and prevent the I-ring from advancing.		
ARS No Op	Op Dcdr 4, Not 2 and 1 Op Dcdr B and Not A and Not 8 Op Reg ARS Not C-Bit	13.13.08
Not I Ring Advance	ARS No Op	
2. Repeat I-cycle; read out IAR; modify by +1.		
Set I Cy Ctrl	ARS No Op	12.13.02
I Cy Ctrl	Set I Cy Ctrl Next to Last Logic Gate	12.12.23
I Cy	I Cy Ctrl LGB	12.12.04
Read Out IAR	I Cy Ctrl No Branch Latch Logic Gate Special A	14.71.34
Set Mem AR Gated	LGA or LGR, 2nd Clock Pulse	14.17.16

3. If there is no B-Channel WM, repeat step 2 until a B-Channel WM is detected.

4. If there is a B-Channel WM, set op code into op reg, and continue instruction read out.

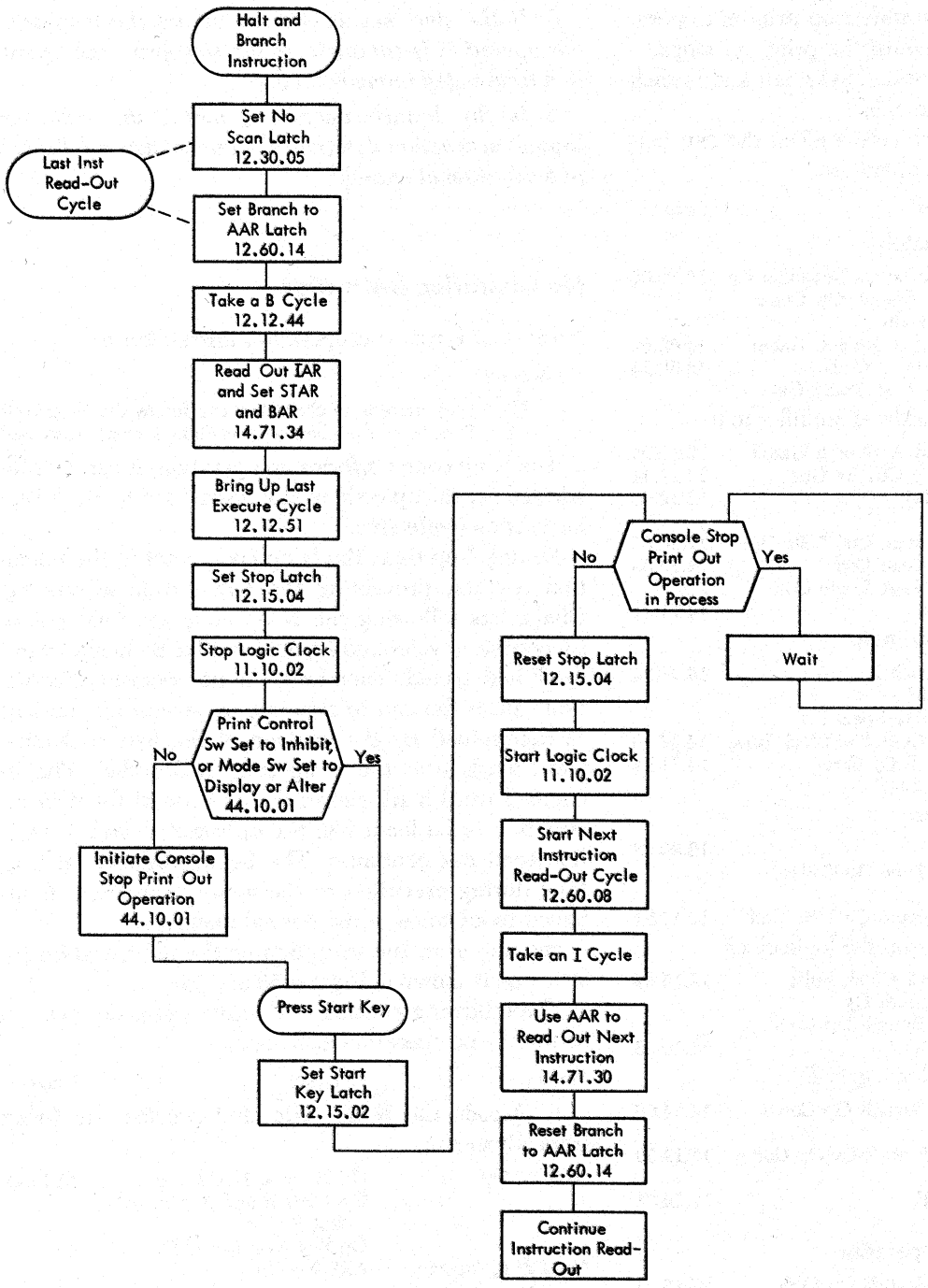


Figure 52. Halt and Branch

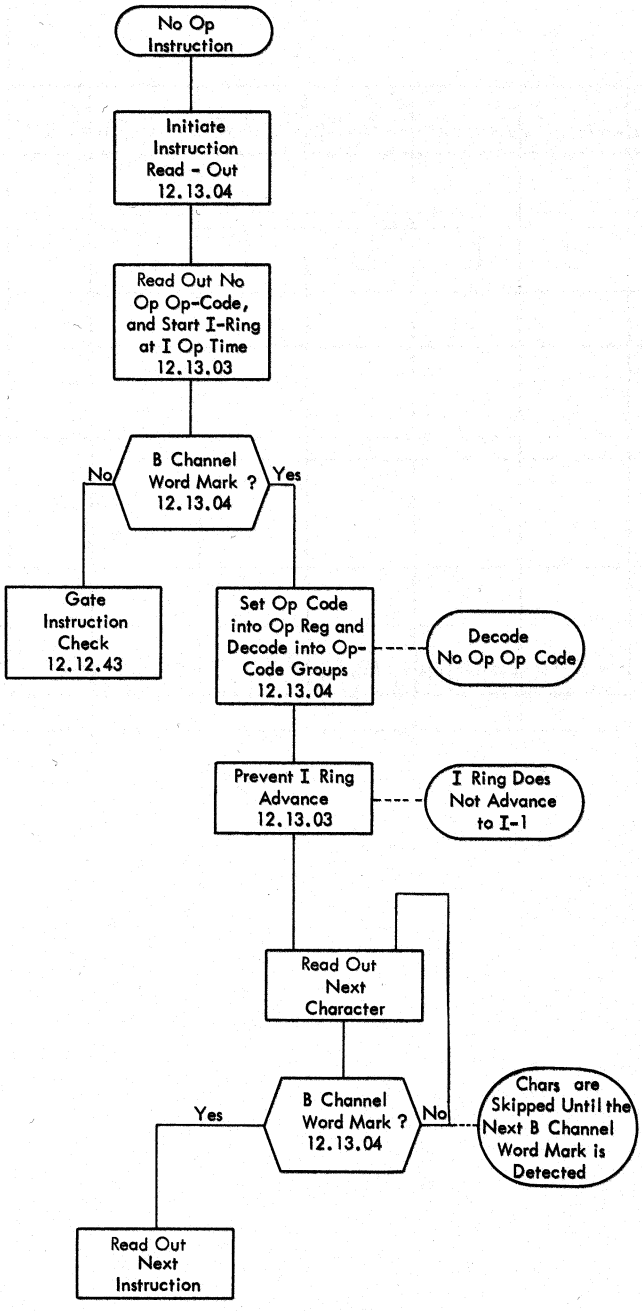


Figure 53. No Operation

Signal	Control	Logic
Set Op Register	B Ch WM Bit I Cy LGF I Ring Op Time	12.13.04

Questions on No Operation Instruction

Answers to review questions are in the Appendix.

1. To what point does the I-ring advance when the N op code is set in the operation register?

2. When the N op code is detected, are other characters read out of core storage?

3. If a program contains N00599 06500 A 07700 07720 Š 09000 08000, will the add operation be performed?

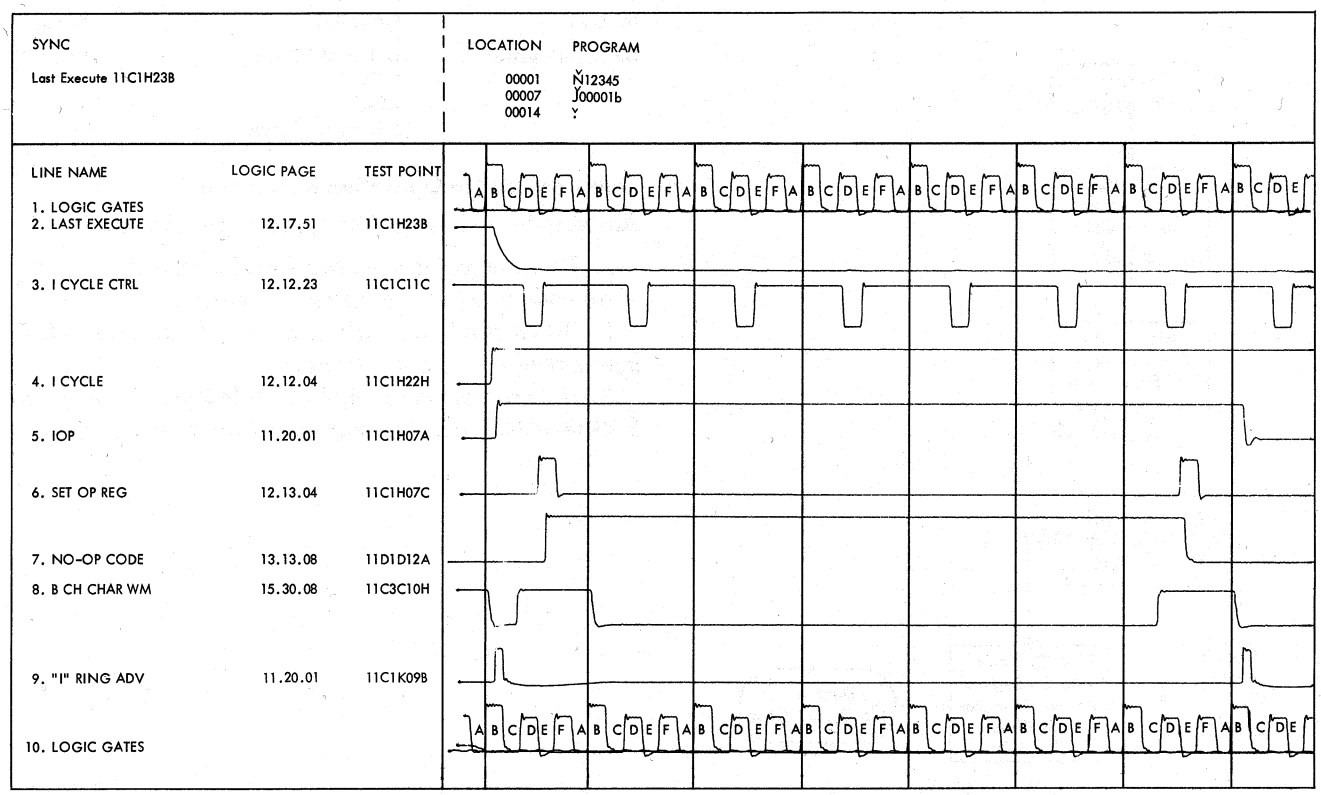


Figure 54. No Op Timings

This section describes the accelerator, priority processing, and program addressable clock special features for the IBM 1410 Data Processing System.

1410 Accelerator Feature

The IBM 1410 Accelerator Feature (Special Feature 1007) increases the processing speed of the IBM 1411 Processing Unit. Compute operations are performed at speed increases of approximately 15 to 23 percent; input-output data transfer rates are not changed. Because the same logic operations are performed faster, all programs that operate on 1410 systems without the accelerator feature can be executed without modification when the feature is installed.

The accelerator feature:

1. Reduces the range of storage cycle duration from 4.5 to 7.5 microseconds to 4.0 to 6.66 microseconds.
2. Shortens execute cycle lengths by two logic gates in three operations.

Results of Speed Increases

By replacing the standard 1.33 megacycle ocillator in the logic clock circuits with a 1.5 megacycle oscillator, an effective speed increase is made in the storage cycle. The durations of clock pulses are reduced from 0.375 to 0.333 microseconds, resulting in 0.666 microseconds logic gate pulses. Although the logic gate ring contains the same triggers (A through K), the shortest cycle (stop at F) is reduced to approximately 4.0 microseconds ($6 \times .666 = 3.996$); the longest cycle (stop at K) is reduced to approximately 6.66 microseconds ($10 \times 0.666 = 6.660$). The shorter storage cycle requires a TNL card to replace the DFT card containing STAR latches (14.17.01 – 14.17.15 accelerator feature systems pages).

In addition to the changes required to shorten the length of storage cycles, the accelerator feature makes significant changes in the adder and assembly units and their associated translating and control circuits. Faster circuits in these units shorten execute cycles in multiply, divide, and data move operations by two logic gates. The "stop at F on B-cycle" common op code grouping line is conditioned rather than "stop at J on B-cycle" as in normal operation.

Component Circuit Changes

Diffused diode transistor logic (DDTL) circuits are used to obtain the faster processing speeds that the accelerator feature provides. DDTL circuits require approximately 15 to 100 nanoseconds delay per decision. The 1411 CPU is primarily composed of saturated drift transistor resistor logic (SDTRL) and saturated drift transistor diode logic (SDTDL) requiring approximately 50 to 150 nanoseconds delay per decision. DDTL circuits replace only selected cards in the CPU to decrease the time required to execute specific logical functions. The following simplified example illustrates the manner in which this is accomplished:

To obtain an output at P, simultaneous inputs to block Y are required. Assume that the circuit delays in path 2 are three times longer than the delays required in path 1. Pin a on block Y is conditioned three times sooner than pin b.

The circuit delay in path 2 can be decreased without changing the logical functions that blocks 3 through 9 perform by using faster circuits. If blocks 3 through 9 were replaced with circuits three times faster, the delay in path 2 would be decreased by two-thirds. The new circuit in path 2 makes the output at pin P available in one-third the original time.

DDTL circuits use B levels ($+B = +6$ volts; $-B = 0$ volts). High speed converter circuits transform S levels used in SDTRL and SDTDL circuits to B levels or B levels to S levels. The *DDTL Component Circuits CE Manual of Instruction* (R23-2618) describes diffused diode transistor logic in detail.

Priority Processing Feature

The priority feature is an interrupt system that provides an automatic branch to a fixed memory location (00101) when a channel or an I-O device indicates cer-

tain conditions. These conditions set indicators to allow the CPU to determine the specific cause of the interruption. I-O devices can automatically interrupt the main routine either with a request for service or by indicating that certain conditions exist in the I-O operation.

Priority processing provides increased efficiency in the use of system units. Virtually all waiting can be eliminated when transferring data from CPU to an output device or from an input device to CPU. The priority feature allows CPU to continue processing until the I-O device completes an operation. The channel (or the I-O device) signals the CPU when the I-O operation ends or certain conditions occur, allowing the CPU to begin another operation (or to continue the present operation) on that channel as soon as the main routine can be interrupted. The CPU need not interrogate indicators repetitively to determine when the operation ends.

An interrupt transfers system control from the main routine to a priority routine. With the priority feature, the CPU can process two independent programs; one program is being executed while the other program is waiting for an I-O operation to end.

The following conditions are necessary to effect an interruption:

1. The instruction in the main routine to be interrupted must meet the requirements of interruptible instructions.
2. The CPU must be operating in the priority alert mode.
3. The CPU must receive an interrupt request signal.

Interruptible Instructions

The actual interruption of the main routine program occurs during instruction read out at I-ring 6 time. An operation is defined as a two-address op code if no B-channel word mark is detected at I-ring 6 time. Consequently, chained instructions cannot be interrupted. Neither the store address register instruction nor I-O instructions can be interrupted. Figure 55 shows the instructions that can be interrupted and the number of characters that these instructions must contain.

Priority Alert Mode

Installation of the priority feature does not require that the CPU be subject to interruptions by I-O devices at all times; the CPU can operate in either priority alert mode or normal mode. In normal mode, no interruptions can occur. In priority alert mode, interruptions are permitted.

The CPU can enter the priority alert mode, allowing program interruptions, only when the Y(I)E instruction is executed. The Y(I)E instruction causes the CPU to branch unconditionally to the storage location designated by the I-address in the instruction and to operate in the priority alert mode. When the CPU is in the priority alert mode, the priority alert mode indicator on the IBM 1415 console turns on.

Operation	Op Code	Length	Non-Interruptible Length
Zero and Add	?	11	1,6
Zero and Subtract	!	11	1,6
Add	A	11	1,6
Subtract	S	11	1,6
Multiply	@	11	1,6
Divide	%	11	1,6
Move Characters and Edit	E	11	1,6
Move Characters and Suppress Zones	Z	11	1,6
Compare	C	11	1,6
Clear Storage and Branch	/	11	1,6
Set Word Mark	,	11	1,6
Clear Word Mark		11	1,6
Branch if Bit Equal	W	12	1,6
Branch on Word Mark or Zone Equal	V	12	1,6
Move Data	D	12	1,6
Branch if Character Equal	B	12	1,6
Table Lookup	T	12	1,6
Test and Branch	J	7	1
Branch if I-O Chan Status Ind On (Ch 1)	R	7	---
Branch if I-O Chan Status Ind On (Ch 2)	X	7	---
Parity Test Branch	Y	--	1,7
No Operation	N	--	Any
Move (I-O Operation)	M	--	10
Load (I-O Operation)	L	--	10
Store Address Register	G	--	7
Halt	.	--	1,6
Unit Control	U	--	2
Control Carriage	F	--	2
Stacker Select Feed	K	--	2

Figure 55. Interruptible and Non-Interruptible Operation Codes

Priority alert mode is reset, returning the CPU to normal mode when either:

1. An I-O device interrupts the main routine; or,
2. The Y(I)X instruction (branch unconditionally and exit from priority alert mode) is executed.

After each program interrupt, the CPU must perform the Y(I)E instruction to return to priority alert mode.

Interrupt Request

Each channel can have six priority request indicators: overlap, I-O unit, inquiry, outquiry, seek, and attention. Interrupt conditions occurring in a channel or I-O unit are signalled by turning on a particular priority request indicator. When the CPU recognizes the request of a priority request indicator the priority alert mode is reset, and an unconditional branch to storage location 00101 is executed.

Interrupt requests occur on a real time basis. Therefore, a priority request may occur after I-ring 6 time in instruction read out of an interruptible instruction. Because the priority request sets the corresponding priority request indicator, the interrupt will occur during instruction read out of the next interruptible instruction in all other cases except when an overlap

priority request occurs too late (after I-ring 6 time in instruction read out of an interruptible instruction) to cause an interrupt, and the next instruction is a branch if I-O status indicator on, the overlap priority request indicator set previously is reset, and no interruption occurs. The resetting of the overlap priority request indicator under these circumstances is a normal function of the branch if I-O status indicator on instruction.

Figure 56 shows priority request indicators and the conditions to set and reset the indicators.

OVERLAP PRIORITY REQUEST INDICATOR

The channel 1 or channel 2 overlap priority request indicator is turned on at the completion of an overlapped read, write or write check operation on that channel. The indicator can also be turned on by performing a seek operation in overlap mode (7631/1301). In this case, the indicator turns on at the completion of the address transfer to the IBM 7631 File Control prior to the actual movement of the access mechanism.

The priority test and branch instructions, Y(I)1 (channel 1) and Y(I)2 (channel 2) are used to determine the status of the overlap priority request indicators on the respective channels. Execution of the R(I)d (channel 1) or X(I)d (channel 2) instruction resets the corresponding overlap priority request indicator.

I-O UNIT PRIORITY REQUEST INDICATOR

Two priority select switches on the IBM 1415 Console and the I-O devices that the switches select control the channel 1 and channel 2 I-O unit priority request indicators. One priority select switch represents channel 1 operation; one switch represents channel 2 operation. Each switch has five positions: OFF, CARD READER, CARD PUNCH, PRINTER, AND PAPER TAPE READER. A double position priority select key on the IBM 1415 Console controls each priority select switch. When a priority select key is in the off position, the corresponding priority select switch is ineffective; when a key is set on, the I-O device selected on the corresponding switch is permitted to set the I-O unit priority request indicator. When a priority select key is initially set to the on position, the I-O unit priority request indicator turns on automatically, regardless of either priority select switch setting. This automatic turn on of the I-O unit priority request indicator allows the CPU to initially enter the interrupt routine. Succeeding interruptions that the I-O unit priority request indicator cause are authentic indications that the selected I-O device requests service or signals a certain condition in the I-O operation.

If the priority select switch for channel 1 is selecting CARD READER, the I-O unit priority request indicator for channel 1 is set each time that the card reader completes a read operation.

Execution of the Y(I)U (channel 1) or Y(I)F (channel 2) instruction causes the CPU to test and reset the corresponding I-O unit priority request indicator.

The card reader, card punch, and printer can initiate an interrupt by setting either the overlap or I-O unit priority request indicators.

INQUIRY REQUEST INDICATOR

The channel 1 or channel 2 inquiry request indicator is set when one of the following conditions occur:

1. The inquiry request key on the IBM 1415 Console (for channel 1 only) is pressed.
2. Any other serial input buffer in the IBM 1414, Model 4 or 5, Input/Output Synchronizer except the paper tape reader is loaded with 80 characters or receives an end of message character.
3. An error occurs in transmission from an input TELE-PROCESSING® unit to the 1414, Model 4 or 5, input buffer.

Pressing the release key on the console resets the inquiry request indicator if the inquiry request key caused the inquiry request status. If a teleprocessing unit(s) produced the inquiry request status, the inquiry request indicator is automatically reset when all inquiry requests are serviced by a proper read operation directed to the unit(s) in inquiry request status.

Inquiry request interrupts caused by the console and teleprocessing buffers can be differentiated by issuing an I-O NOP instruction to the console. If the inquiry request was not caused by the console, the no transfer indicator is set; if the inquiry request was caused by the console, the no transfer indicator is not set.

OUTQUIRY REQUEST INDICATOR

The outquiry request indicator for channel 1 or channel 2 is set:

1. When eighty characters have been processed out of an output optional buffer in the 1414, Model 4 or 5, or the last character in the message has been transmitted to a Tele-Processing unit.
2. When an error occurs in transmission from an optional buffer in the 1414, Model 4 or 5, to an output device.

Execution of the Y (I) N (channel 1) or Y (I) ≠ (channel 2) instruction tests and resets the outquiry request indicator.

SEEK PRIORITY REQUEST INDICATOR

When any access mechanism on the channel is in seek complete status, the seek priority request indicator for channel 1 or channel 2 is set. The Y (I) S (channel 1) or Y (I) T (channel 2) instruction is used to test the

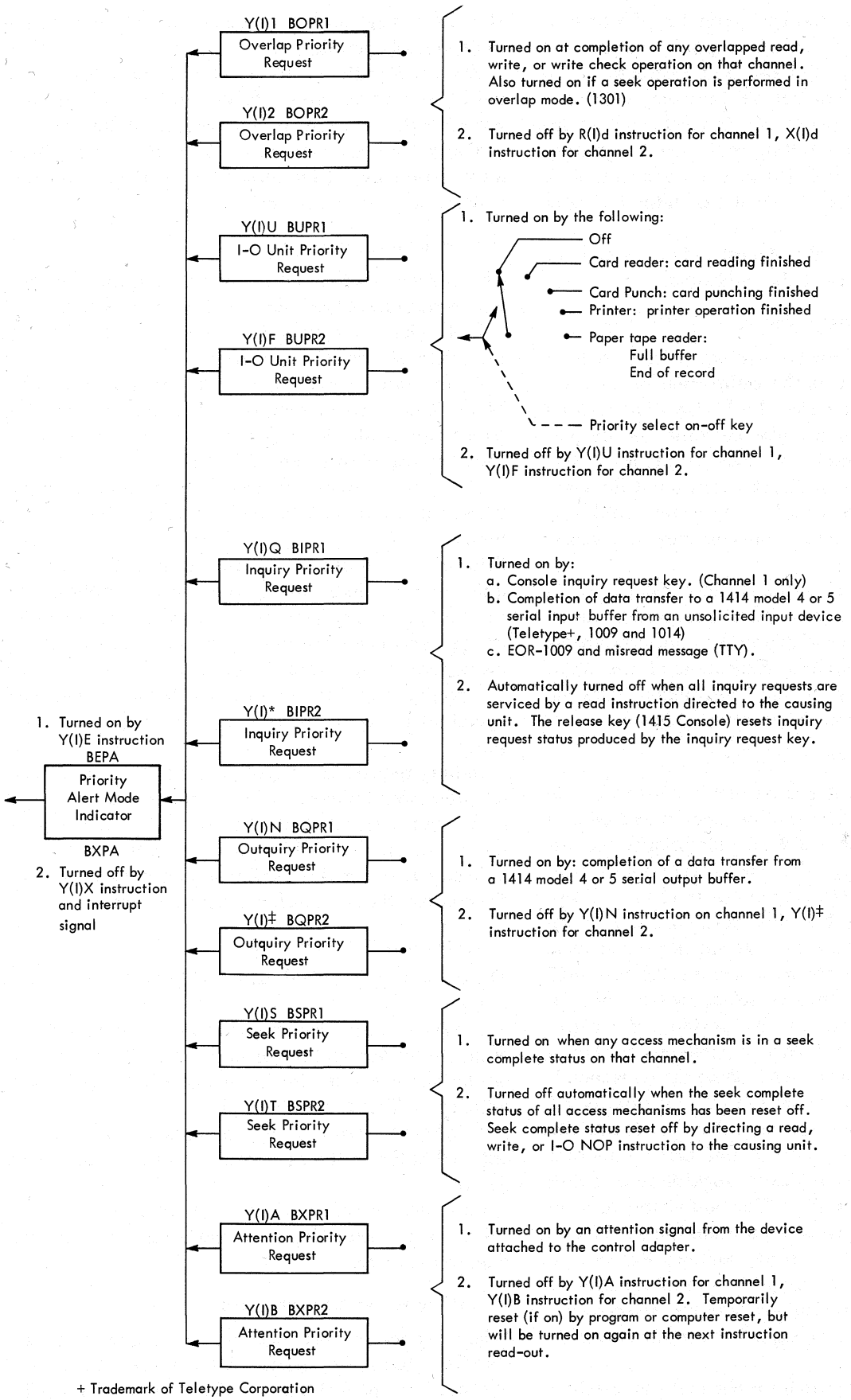


Figure 56. Priority Request Indicators

seek priority indicator. A read, write, or I-O NOP instruction directed to the unit or units in seek complete status resets the seek complete status, causing the seek priority request indicator to turn off.

ATTENTION PRIORITY REQUEST INDICATOR

An attention signal from any device attached to the channel control adapter sets the channel 1 or channel 2 attention priority request indicator. Execution of the Y (I) A (channel 1) or Y (I) B (channel 2) instruction tests and resets the corresponding indicator.

Programming

When an interrupt occurs, the CPU branches unconditionally to storage location 00101. When the priority routine is complete, program control must be returned to the interrupted instruction in the main routine. The store B-address register instruction (G ccccc B) should be the first instruction in the interrupt routine so that the address of the interrupted instruction in the main routine is retained. Because the interrupt occurred at I-ring 6 time, the contents of the BAR after the unconditional branch to storage position 00101 is 6 plus the address of the op code in the interrupted instruction. One method of returning to the storage location containing the op code of the interrupted instruction is:

1. To use the C-address in the store B-address register instruction (G ccccc B) to designate the I-address in the unconditional branch instruction (J iiiii b or Y iiiii E) at the end of the interrupt routine, and
2. Index the unconditional branch instruction at the end of the interrupt routine by −6.

The priority routine must save the reading of the arithmetic condition latches (arithmetic overflow, divide overflow, zero balance, compare high, compare low, and compare equal latches) if they are used in the priority routine. The priority routine must also determine the cause of the interruption.

The following program demonstrates a program interrupt. The main routine is a single add instruction that repetitively adds 1 to the B-field. The priority routine prints a line by interrupting the main routine. The priority select ON/OFF key on the console is set in the PRINTER position, and the priority select key, also on the console, is in the on position.

		MAIN ROUTINE	
00001	Y	00008 E	Turn on priority alert
00008	A	01000 02010	Add 1 to B-field
00018	J	00008 b	Repeat the addition
		INTERRUPT ROUTINE	
00101	G	00137 B	Store contents of BAR
00108	M	%20 02000 W	Print a line
00118	R	00125 ≠	Interlock
00125	Y	00132 U	
00132	Y	000 ≠0 E	Turn on priority alert and branch back to main routine
		CONSTANTS	
01000	I		A-field
02000	0000000000		B-field
02132	≠		Group mark-word mark
00025	bbbbO		Index register 1 (−6)

The program starts at storage position 00001. The Y(00008)E instruction causes the CPU to set the priority alert mode latch (Systems page 19.10.07). The CPU then executes the add instruction. The unconditional branch instruction, J(00008)b, causes the CPU to repetitively perform the add instruction until the first interrupt is requested.

When the priority select key on the console is set in the on position, the first interrupt occurs. With the priority switch latch on, the next console clock 3 pulse sets the I-O unit priority request latch (Systems page 19.10.06). The I-O unit interrupt latch allows logic gate E of I-ring op time to set the delayed interrupt latch.

In the program example, only the add and unconditional branch instructions can be interrupted. With E-channel not in process, "selected I-O unit interrupt condition" is brought up, conditioning "interrupt request." "Interrupt request" conditions "start interrupt" at I-ring 6 time in instruction read out of the add or unconditional branch instruction (Systems page 19.10.03).

"Start interrupt" allows logic gate E of I-ring 6 to turn on the interrupt branch latch (Systems page 19.10.02). The sequence then follows a standard branch. The I-ring is reset, and a B-cycle follows to set the contents of the IAR into the BAR. At the end of the B-cycle, the fixed address 00101 is generated and set directly into STAR, allowing I-ring op time to begin the priority routine.

The Y(00132)U instruction resets the interrupt request latch to allow the program to return to the main routine.

The G(00137)B instruction stores the contents of the BAR (00014 in this example) at location 00137. The print and R(I)≠ instructions follow.

The Y(000≠0)E instruction has now been changed to Y(000/4)E. The record mark (A-, 8-, and 2-bits) was changed to a slash (C-, A-, and 1-bits) by the G op code at 00101. When the Y(000/4)E is executed, the branch is to 00008 because the A-bit in the tens position indexes the address by −6 (index register 1).

The add routine is repeated until the fall of busy from the printer causes another interrupt. The entire cycle then continues indefinitely.

Figures 57, 58, and 59 show CPU circuit operations and timings in an interrupt operation.

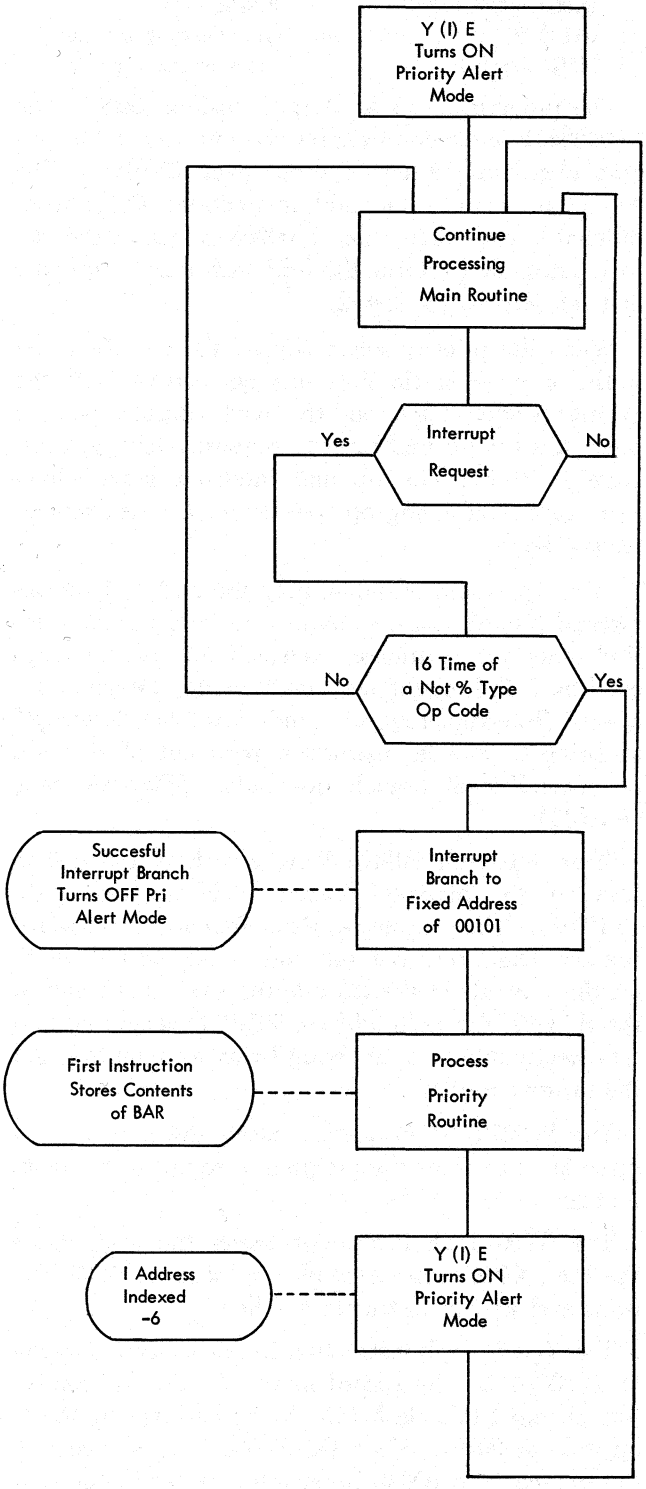


Figure 57. Interrupt Data Flow

Program Addressable (Real Time) Clock

The program addressable clock for the IBM 1410 Data Processing System provides a method of establishing an accurate log of system usage time. Under stored program control, the immediate time registered by the clock can be obtained for recording or processing. (ALD's refer to the program addressable clock as the real time clock.)

The program addressable clock is located on the IBM 1415 Console and is powered from a voltage source not affected by the normal power off controls of the IBM 1410 Data Processing System. Emergency power off, however, removes power from the clock.

The clock indicates the time of day in continental (24 hours) notation to hundredths of an hour; for example, the time 3:26 PM is indicated as 15:43; 15 represents the 15th hour of the day (3:00 PM); 43 represents the fraction of the hour in hundredths past the 15th hour (43/100 is approximately equal to 26/60). Although the clock advances once per minute, it counts in hundredths of an hour. The units position of the hundredths of an hour is expressed by the digit 0, 2, 3, 5, 7, or 8 (Figure 60).

Programming

Read out of clock time is accomplished by the store address register instruction, G(C)T. Execution of the G(C)T instruction causes the CPU to place four digits representing clock time and a special identifier digit in core storage beginning at the location specified by the C-address. The five digits are stored right to left beginning at the units position of the hundredths of an hour. The identifier digit is stored to the left of the last time digit in the low order core storage position. Any zones or word marks in the core storage position in which time data are to be stored are not affected.

The clock advances once each minute; the real time clock busy signal, active for 345 ± 115 milliseconds, prevents clock time from being stored during this interval. If the real time clock busy signal is active when the CPU executes the G(C)T instruction, five 9's are read into core storage in the positions designated for time data, indicating that the clock is busy. The program must instruct the CPU to perform the operation again to obtain correct clock time. Program use of the clock should provide for the possibility that the real time clock busy signal will be active when clock time is requested. The G(C)T instruction should be followed by a branch if character equal instruction, B(I) (B)9. If the identifier position of the clock data contains a 9, the program branches to the I-address specified in the instruction. The B-address is the core storage position in which the identifier digit is located (the C-address −4). If the real time clock busy signal is not active when the CPU executes the G(C)T instruction, four time digits and a zero identifier digit are read from the clock and loaded in core storage.

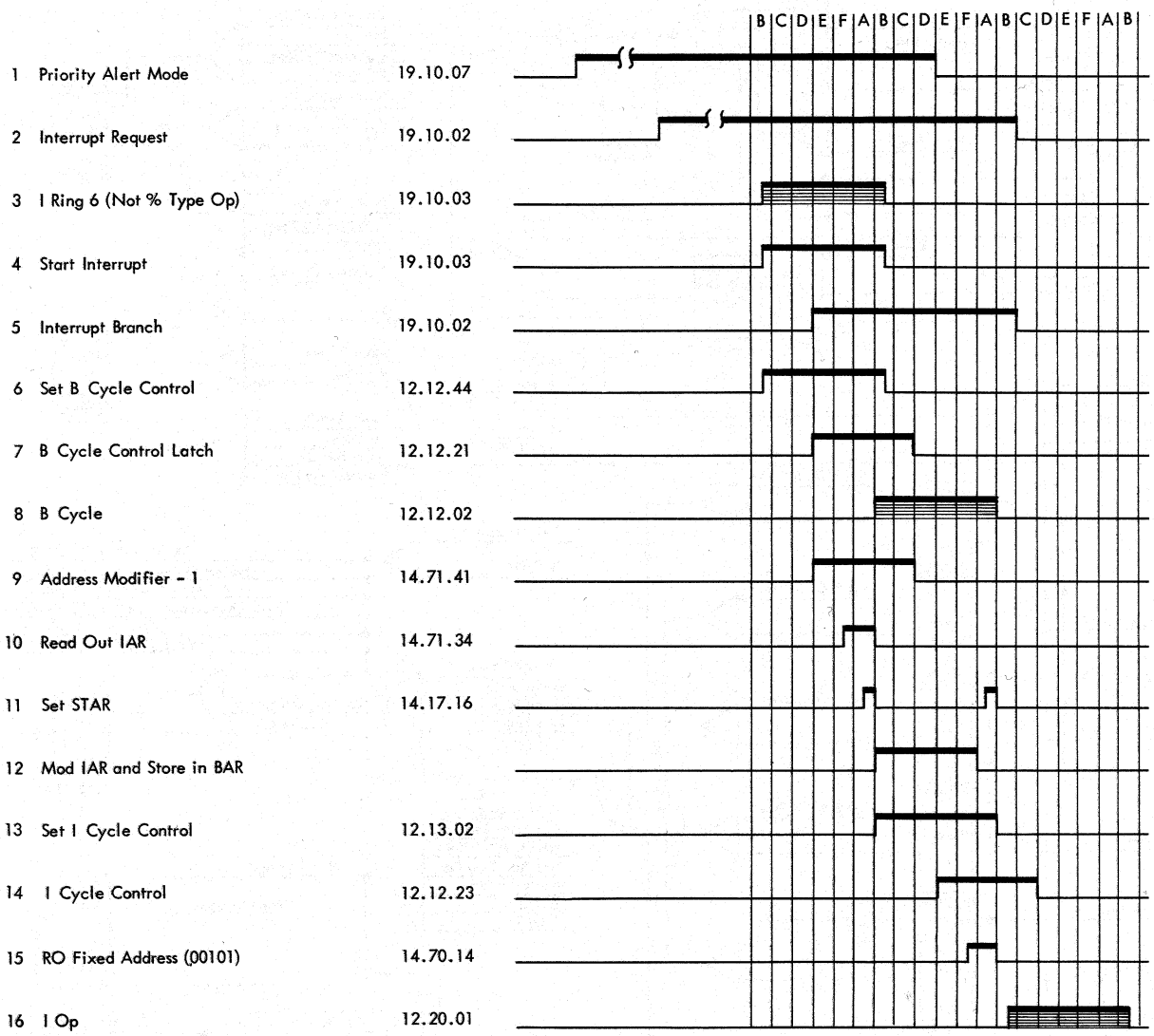


Figure 58. CPU Timings in Interrupt Operation

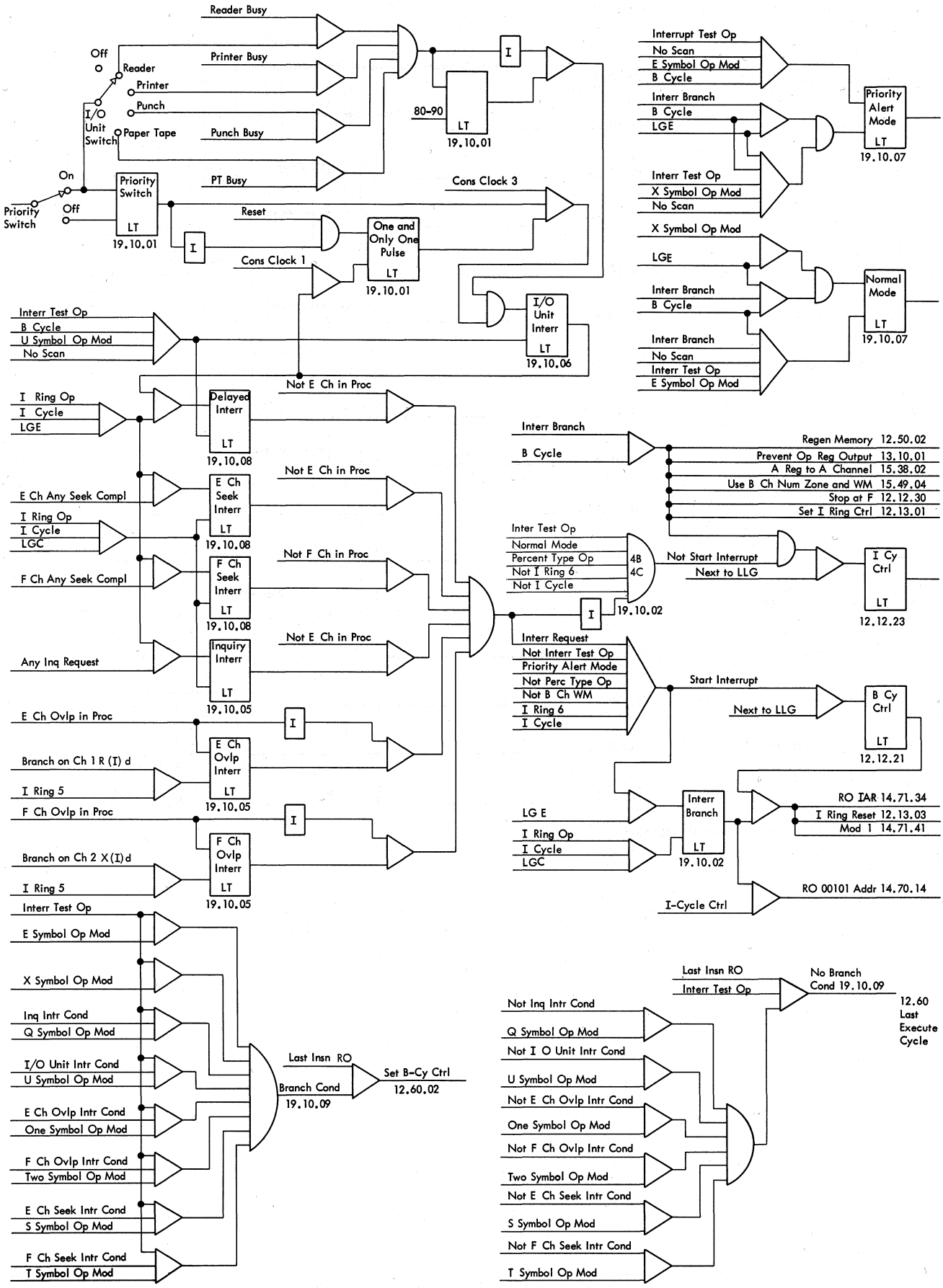


Figure 59. Interrupt Controls

TIME		TIME		TIME		TIME	
Min's	Hund's	Min's	Hund's	Min's	Hund's	Min's	Hund's
01	02	16	27	31	52	46	77
02	03	17	28	32	53	47	78
03	05	18	30	33	55	48	80
04	07	19	32	34	57	49	82
05	08	20	33	35	58	50	83
06	10	21	35	36	60	51	85
07	12	22	37	37	62	52	87
08	13	23	38	38	63	53	88
09	15	24	40	39	65	54	90
10	17	25	42	40	67	55	92
11	18	26	43	41	68	56	93
12	20	27	45	42	70	57	95
13	22	28	47	43	72	58	97
14	23	29	48	44	73	59	98
15	25	30	50	45	75	00	00

Figure 60. Time Derivation Table

If any position of the clock fails to read out when time data are requested and the clock is not busy, a 9 is stored in the failing position and in positions to the left of the failing digit; for example, 99950 read into core storage as the result of the G(C)T instruction indicates that a failure occurred in the first hour position. When the CPU executes the second branch if character equal instruction 460 milliseconds after the first test revealed that the clock was busy, correct clock data should be loaded in the specified area in core storage. If the second test also indicates that the clock is busy, a failure has occurred.

Figures 61, 62, and 63 show CPU logic operation and timings when executing the G(C)T instruction.

Procedures to Set and Adjust Clock

Figure 64 shows an exploded view of the program addressable clock; refer to this figure for parts locations when making clock adjustments.

SET TIME

To set the clock, press each of the four clock levers on the console to advance the corresponding commutator disc until the correct numeric indication is reached. Operate the levers one at a time. The digit setting of each disc can be seen through the aperture above the lever.

A commutator disc cannot be advanced when the cam unit holds down the disc lever. To advance the clock when this action occurs, press the adjacent lever to the right until the cam unit releases the lever. The low-order disc of the hundredths clock position (the disc at the extreme right) advances once every minute and must be set immediately after advancing.

FEED AND DETENT PAWLS ADJUSTMENT

If the clock fails to advance properly:

1. Remove the power plug to stop the clock motor immediately after the clock drive lever drops off of the high dwell of the cam.
2. Rotate the commutator disc backward until it is against the detent.
3. Check for 0.005 inch to 0.025 inch clearance between the detent pawl face and the adjusting stud (Figure 65).
4. Position the adjusting stud to obtain the clearance specified in step 3.
5. Check all commutator teeth.

BUSY SWITCH ADJUSTMENT

The duration of the busy signal is established by the difference in length of the long and short cam follower levers. The busy switch adjustment only assures reliable operation of the switch. The point at which the switch transfers defines the beginning of the busy signal.

1. Immediately after the short cam follower lever drops off the high dwell of the cam, remove the power plug to stop the clock motor.
2. Loosen the busy switch mounting screws.
3. With a stiff wire, hold the short cam follower lever half the distance between the high and low dwells of the cam; move the busy switch so that its contacts transfer at midpoint.
4. Tighten the mounting screws.
5. Plug-in the clock power cord to allow the clock motor to run. When both the short and long cam follower levers are on the high dwell of the cam, remove the clock power plug.
6. Check the movement of the busy switch actuating arm by inserting a 0.025 inch feeler gage between the roller and the arm. The arm should move at least 0.025 inch before the switch contacts transfer. If the switch contacts transfer too soon, plug the power cord, and perform steps 1 through 5 again.
7. Using a positive sync, scope point 11B2C24F to determine the length of the busy signal. If the busy signal is not active between 230 and 460 milliseconds, recheck the busy switch adjustment.

COMMUTATOR CONTACT TIMING ADJUSTMENT

The clock advances when the busy signal is active. When advancing, the commutator brush breaks contact with one commutator segment and moves to a second commutator segment (Figure 66). In the process of moving, however, contact overlap occurs as the brush

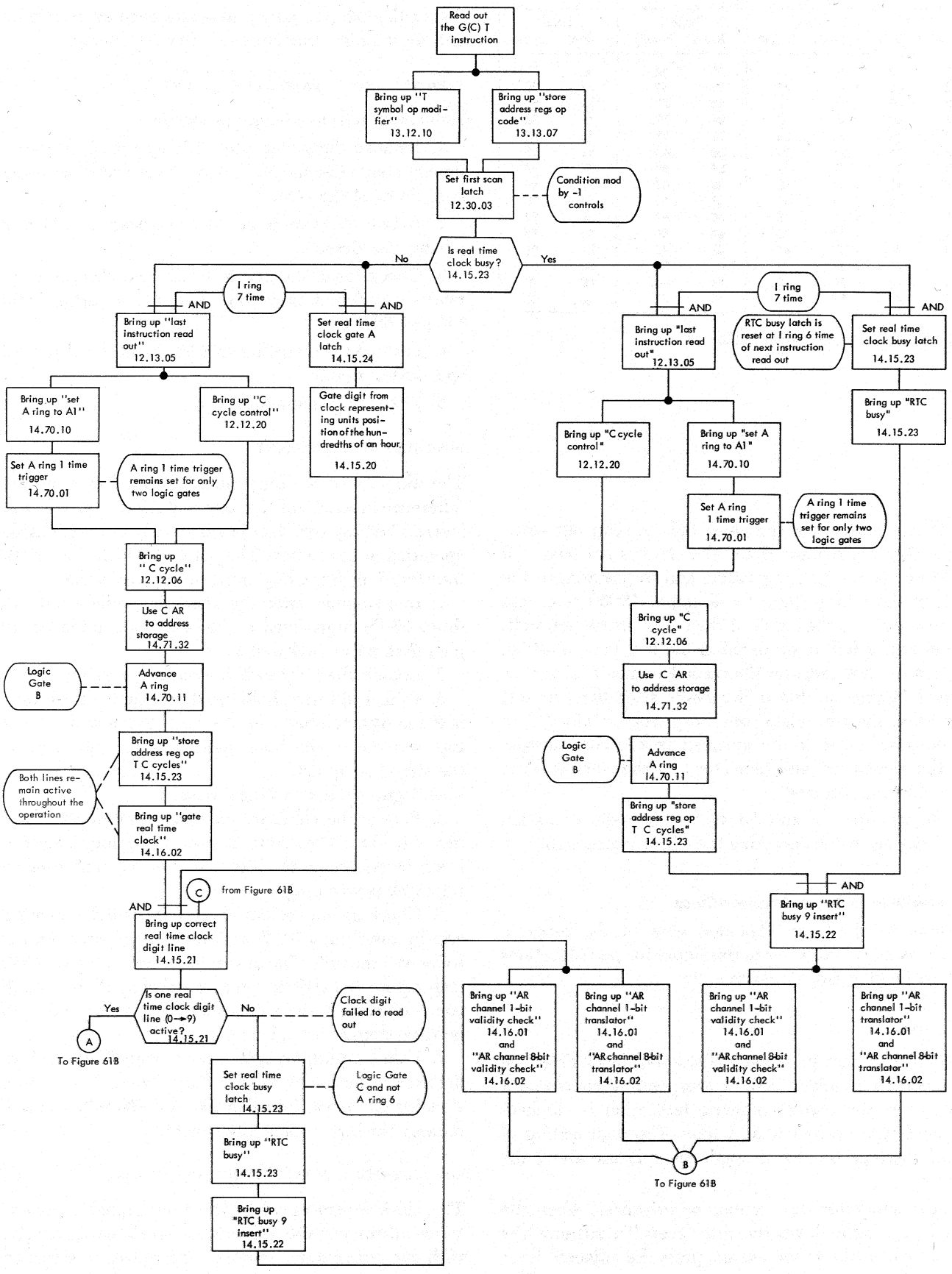


Figure 61A. CPU Execution of G(C)T Instruction

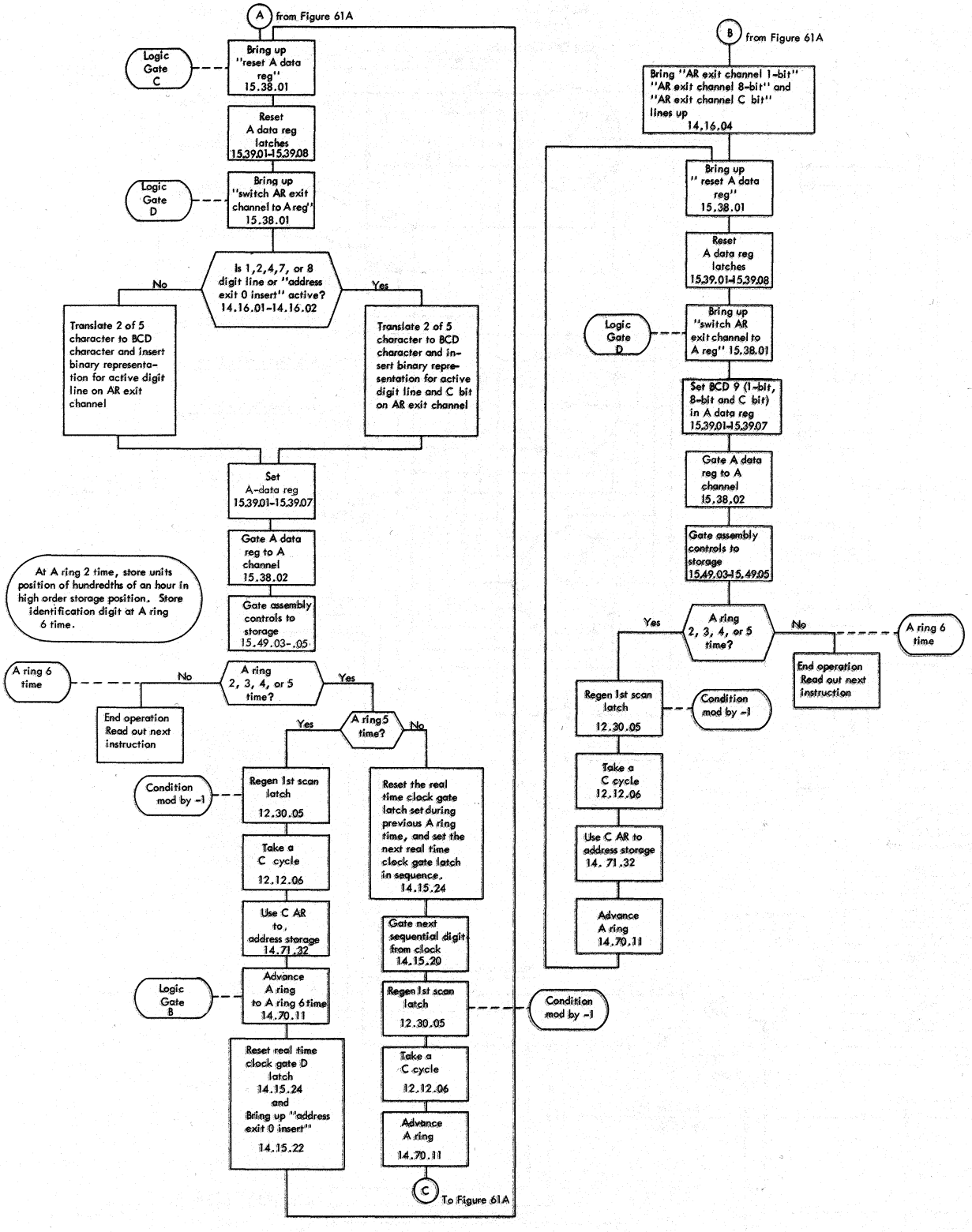


Figure 61B. CPU Execution of G(C)T Instruction

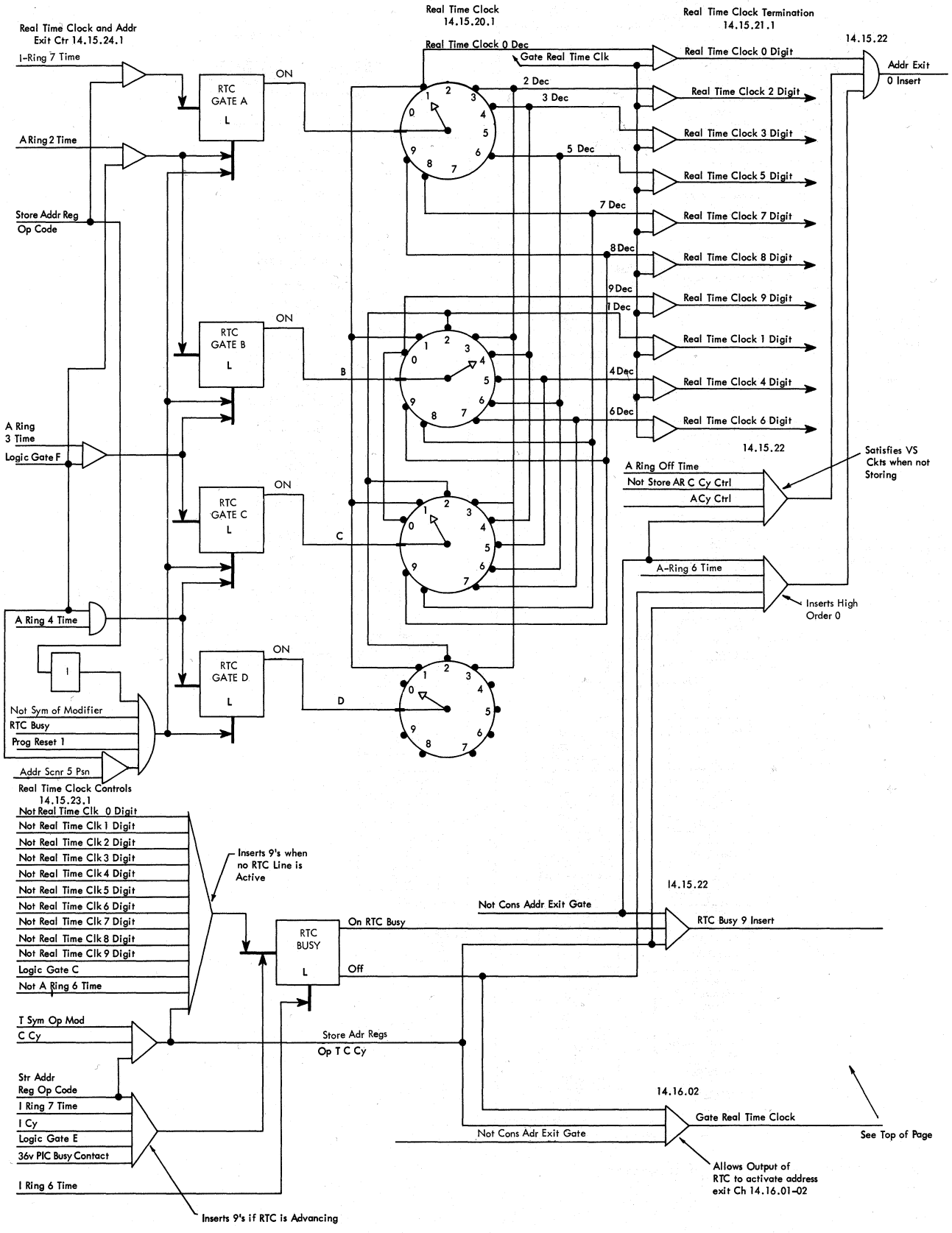


Figure 62. CPU Timings in Execution of G(C)T Instruction

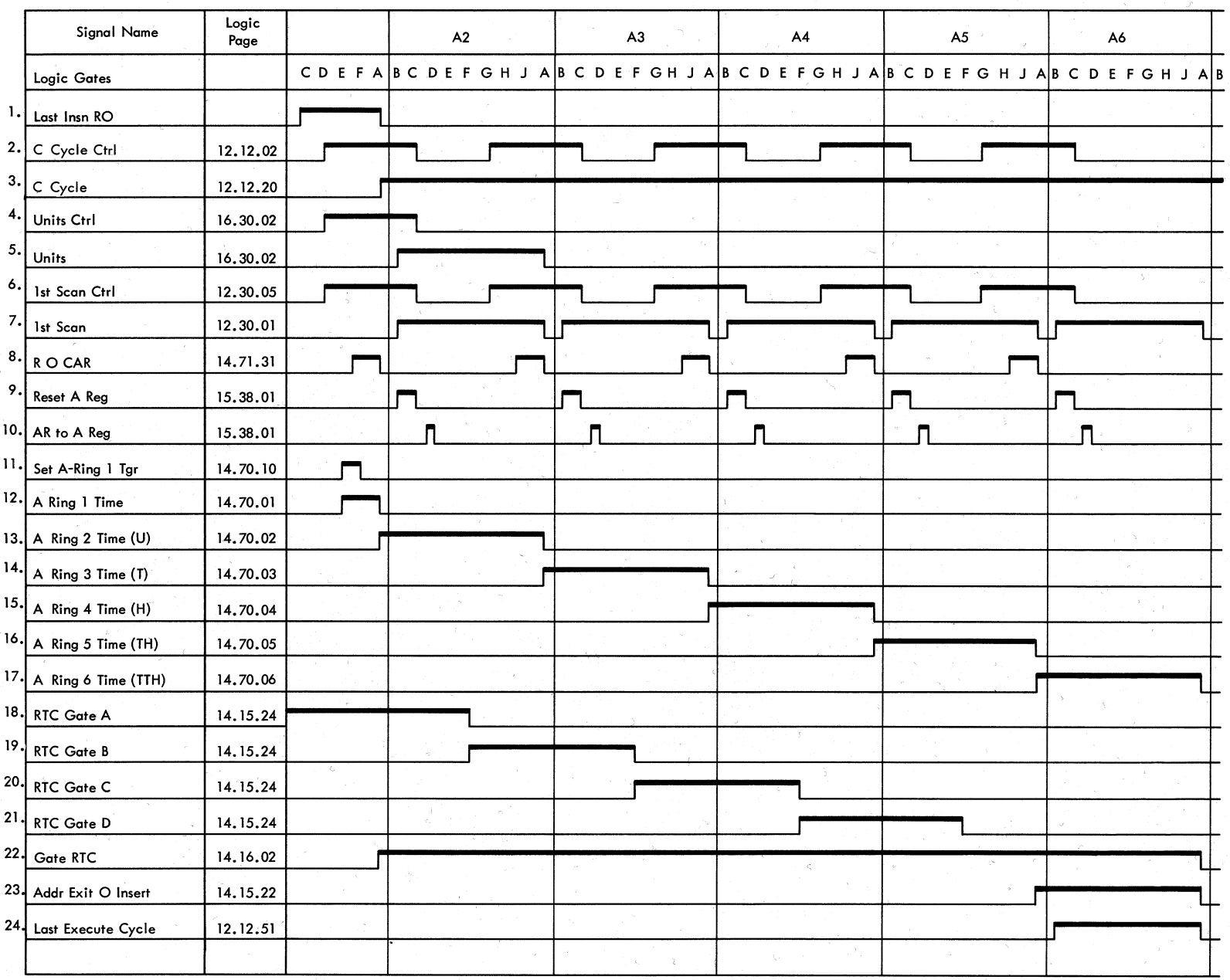


Figure 63. Program Addressable (Real Time) Clock

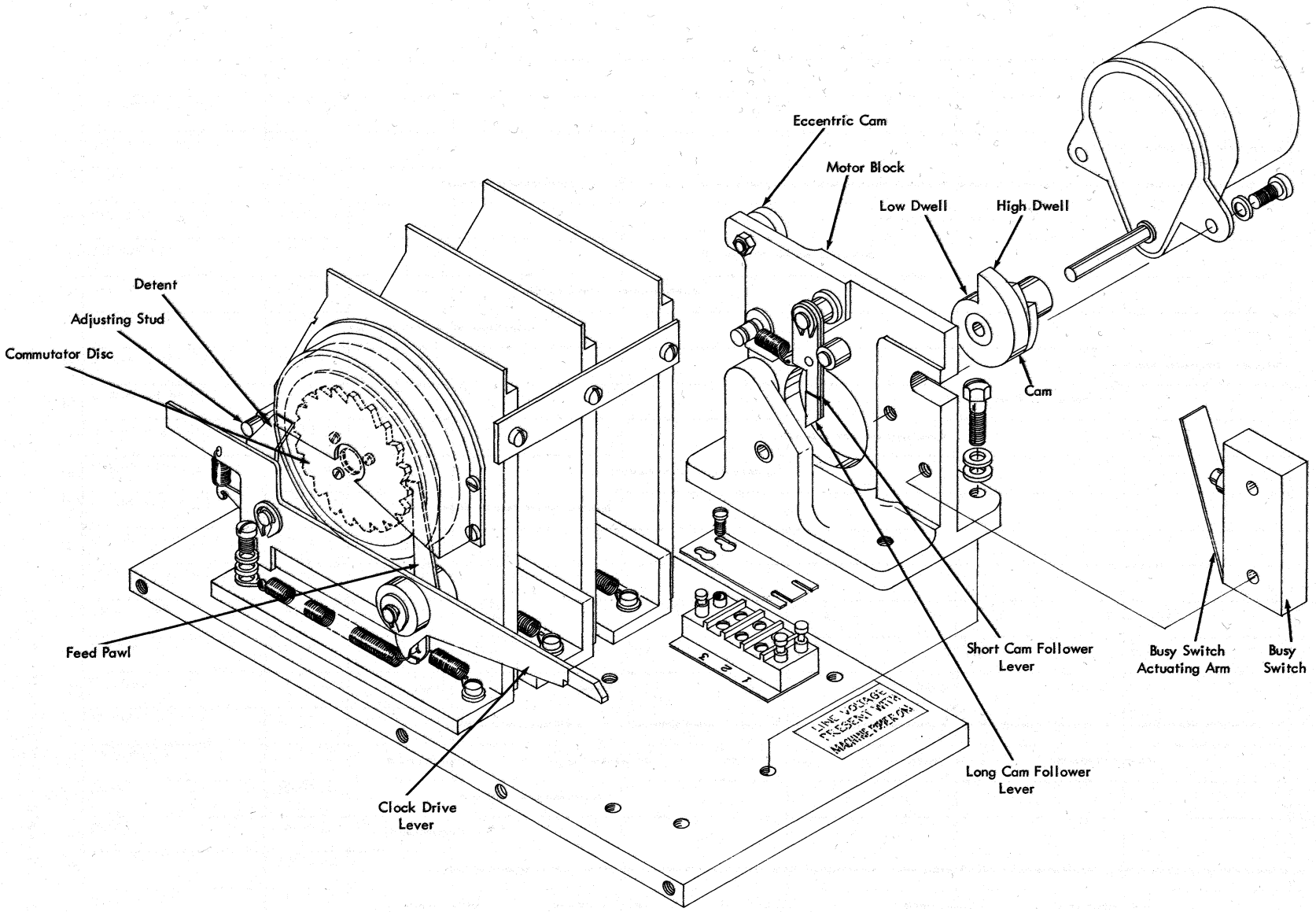


Figure 64. Exploded View of Program Addressable Clock

makes contact with the second commutator segment before it breaks contact with the first commutator segment (Figure 66c). The transfer is complete when the brush breaks contact with the first segment (Figure 66d). However, the busy signal does not drop when the transfer ends. To check the operation for the hundredths position:

1. Turn on system power, and check to see that the clock motor is running.
2. Sync scope at point 11B2C24F (+s RTC Busy) using a 50 millisecond division sweep; use external sync.
3. Connect point 11B2C26F to ground through a 1K ½ watt resistor.
4. Set preamplifier to 1 volt/division, and connect scope probe to point 11B2C26F (RTC gate A driver). The level should be approximately −7 volts. Because of contact overlap, voltage will drop to approximately −9 volts during the transfer and return to −7 volts when the transfer is complete. The transfer should be complete between 100 and 170 milliseconds after the busy signal begins.
5. To adjust, loosen the motor block mounting screws and adjust the eccentric cam against the commutator plate to obtain the timing specified in step 4 (Figure 67).
6. Tighten the motor block mounting screws.

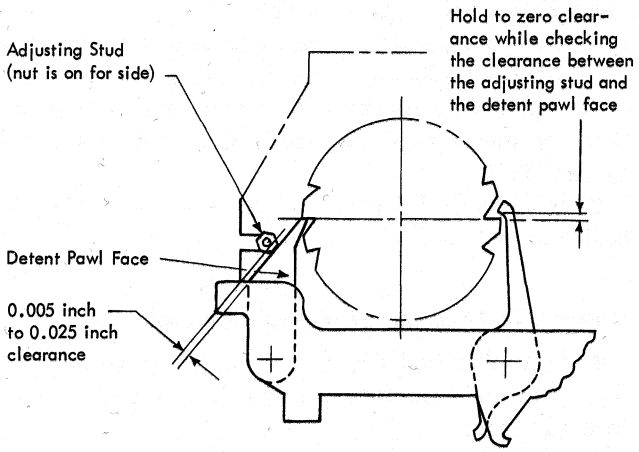


Figure 65. Parts Location for Feed and Detent Pawl Adjustment

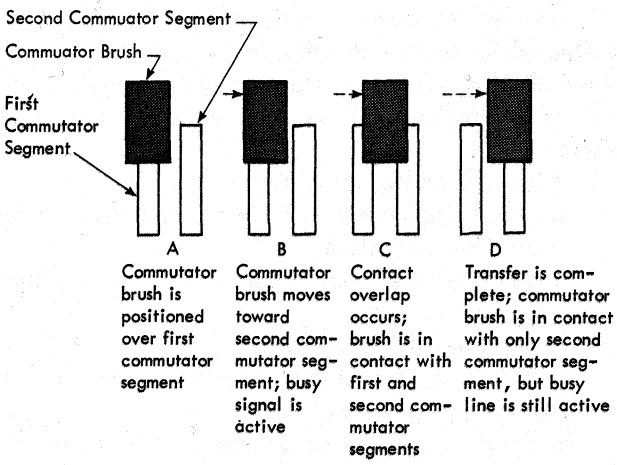


Figure 66. Clock Transfer

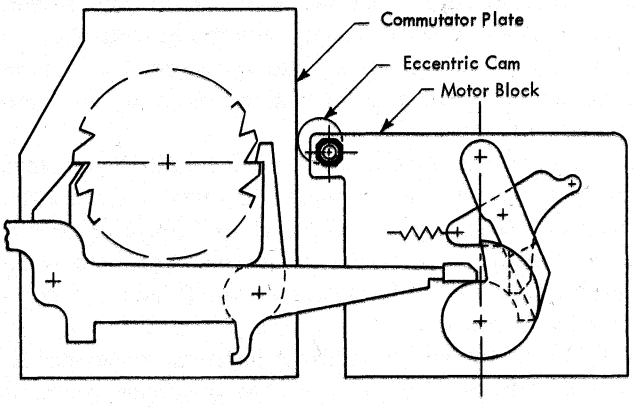


Figure 67. Parts Location for Commutator Contact Timing Adjustment

Appendix

Answers to Questions on 1410 Instruction Formats and Decoding

1. The instruction B 09000 08000 M can be divided into four parts: op code (B); A-address (09000); B-address (08000); d-character (M).
2. Only instructions that control I-O devices contain X-control fields.
3. No, a 1410 instruction can contain an A-address or an I-address or an X-control field.
4. A word mark must be set over the op code in the instruction. The storage position to the right of the last character in the instruction must also contain a word mark.
5. The I-ring indicates the instruction character being processed from storage during instruction phase.
6. Common op code grouping lines are conditioned during instruction phase.

Answers to Questions on Add and Subtract Operations

1. When the add instruction specifies only one address, the contents of the A-field are doubled.
2. No, other A-field characters are not processed after a B-channel word mark is sensed.
3. The first B-cycle in the add or subtract operation is one logic gate longer than other B-cycles in the operation to allow sufficient time to analyze A- and B-field signs and set up true or complement add controls.
4. When the sum in an add operation (or the remainder in a subtract operation) is zero, the zero balance indicator turns on.
5. When the A-field word mark is detected before the B-field word mark is sensed in a true add scan, the extension latch is set, and a series of B-cycles are executed until the operation is complete (B-field word mark). Remaining characters in the B-field are combined with zeros.
6. When the add or subtract instruction specifies A- and B-addresses, the result of the operation is stored in the B-field.

Answers to Questions on Zero and Add and Zero and Subtract Operations

1. If the sign of the A-field is positive in a zero and subtract operation, the sign of the result field is negative.
2. In the zero and add operation, the sign of the result field is the same as the sign of the A-field. In the zero and subtract operation, the sign of the result field is opposite the sign of the A-field.
3. After the zero and add or zero and subtract operation, only the units position of the result field contains zone bits. The zone bits in the units position of the result field are the sign of the field.
4. The zero and add or zero and subtract operation ends when a B-channel word mark is sensed.
5. When the zero and add or zero and subtract instruction specifies only an A-address, the result is stored in the A-field.
6. If the A-field contains all zeros, the zero balance indicator turns on.

Answers to Questions on Multiply Operation

1. Multiplier digits 5, 6, 7, 8, and 9 cause complement add scans; multiplier digits 1, 2, 3, and 4 cause true add scans.
2. When the multiplier contains three digits and the multiplicand contains five digits, the product field should have nine positions (multiplier digits + multiplicand digits + 1).
3. The multiply operation ends when a 0 with a word mark is sensed in the multiplier.
4. No, the CPU does not move the multiplier to the product field when executing a multiply instruction. The multiplier must be moved to the high-order product field positions before the multiply operation begins.
5. When the multiplier is 99:
   a. One complement add scan is required in the multiply operation.
   b. One true add scan is required in the multiply operation.
6. When the multiplier is 828:
   a. Four complement add scans are required in the multiply operation.
   b. Four true add scans are required in the multiply operation.

Answers to Questions on Divide Operation

1. The divide overflow latch is set when an adder carry is detected while the MQ and complement latches are set.
2. a. A correction scan is necessary to restore the dividend to its value preceding the unsuccessful subtraction.

b. The CPU recognizes an unsuccessful reduction when the complement addition performed with the extension and complement latches set produces no carry.

3. D-cycles are taken after all other correction scans except the last correction scan in the divide operation.

4. The multiply divide last latch is set on the first B-cycle of the last correction scan in the divide operation (when character containing dividend sign is detected on the B-channel).

5. The CAR and DAR are used on the first A- and B-cycles, respectively, in each complement add and correction scan (when the units latch is set). The DAR is also used on D-cycles.

6. The CPU sets the quotient sign on a special B-cycle taken when the last correction scan in the divide operation is complete. On the special B-cycle, the units position quotient digit is read out of storage and combined with the correct quotient sign. The sign and the digit are gated to storage.

Answers to Questions on Move Data Operations

1. A-field characters are moved to the B-field.

2. The move data operation does not alter data in the A-field. Characters in the A-field before the operation are in the A-field when the move data operation is complete.

3. Ď(A)(B)d Ď(A) Ď

4. The CPU executes an A-cycle first.

5. a. The 8-, A-, and B-bit positions in the d-character establish the conditions to terminate the move data operation.
   b. The 8-bit position in the d-character determines whether data are moved from left to right or from right to left.
   c. The 1-, 2-, and 4-bit positions in the d-character determine the portion of the A-field character transferred to the B-field.

6. The second scan control latch is set when the d-character contains an 8-bit.

Answers to Questions on Move Characters and Suppress Zeros Operation

1. A-field characters are transferred to the B-field, and zeros and commas to the left of the first significant digit are replaced with blanks.

2. Zone bits in the units A-field position are not transferred to the B-field.

3. During the last B-cycle of the first scan, the BAR was modified by −1. Therefore, when the first scan ends and the second scan begins, the BAR addresses the character to the left of the desired B-field position. A skid cycle is executed to read the addressed character out of storage and return the character to its memory location unchanged. The BAR is modified by +1 during the skid cycle so that the desired high-order B-field character is addressed.

4. When a significant digit is sensed during the second scan, the zero suppress latch is reset. If a character that is not a significant digit, comma, 0, decimal, blank, or minus sign is detected after the zero suppress latch is reset, the zero suppress latch is set again.

5. The first scan ends and the second scan begins when an A-field word mark is sensed.

6. After the move characters and suppress zeros operation, the A-field contains ˇ0061 and the B-field contains .bb61.

Answers to Questions on Edit Operation

1. The MQ latch can be set two times during an edit operation requiring three scans, one time during an edit operation requiring two scans.

2. If the zero suppress latch is not set during the first scan, indicating that the control word contains no zeros, the operation ends when the first scan is complete. The edit operation ends when the second scan is complete if the floating dollar latch was not set during the first scan and either:
   a. The decimal control latch or the zero suppress latch is reset when the second scan ends, or
   b. The last character processed during the second scan is a significant digit.

3. Yes, the CPU will initiate a third scan because the zero suppress latch and the decimal control latch will be set when the second scan ends, and the last character to be processed in the second scan is not a significant digit.

4. No, the CPU takes A- and B-cycles during the first scan, and B-cycles exclusively during the second and third scans.

5. B-field characters read during the first scan while the extension latch is set are in the status portion of the control word.

6. The A-field should not contain more positions than the number of blanks and zeros in the body of the control word.

Answers to Questions on Compare Operation

1. No, the comparisons do not alter data in either the A- or B-field.

2. The compare operation ends when either an A- or B-field word mark is sensed.

3. If the B-field is longer than the A-field, the compare high latch is set, designating the B-field as the greater of the two fields, regardless of their values.

4. If the A-field is longer than the B-field, only A-field characters read out before the B-field word mark is sensed are compared to B-field characters. The com-

pare latch set at the end of the operation, indicating the result of the B-field to A-field comparison, is determined by actual comparisons made (i.e. was the B-field greater than, equal to, or less than the part of the A-field read out before the B-field word mark was detected).

5. The compare equal latch can be set only when the A- and B-channel characters compared on the first B-cycle are equal. The ON state of the units latch identifies the first B-cycle.

6. The A-field is greater than the B-field.

Answers to Table Lookup Operation

1. The table lookup operation ends when the condition that the d-character specifies is satisfied or when the end of the table is sensed.

2. The CPU compares a search argument to table arguments to locate the desired function.

3. The BAR contains 12322 when the operation is complete.

4. No, the table lookup operation does not alter storage data.

5. The CPU performs successive B-cycles to bypass table functions.

6. The A-field contains the search argument.

Answers to Questions on Branch Operations

1. An unconditional branch instruction causes the CPU to perform a branch operation as a direct result of the execution of the branch instruction. A conditional branch instruction causes the CPU to perform a branch operation only when the conditon that the instruction specifies is met.

2. The I-address in a branch instruction is stored in the AAR during instruction read out.

3. A successful branch operation is completed during the subsequent instruction read-out cycle when the AAR (containing the branch I-field) sets STAR.

4. If the branch if I-O channel status indicator on instruction is executed while the machine is performing an I-O overlap operation, the machine interlocks until the overlapped function is complete.

5. The only branch instructions that can be chained are: branch if bit equal, branch if character equal, and branch on word mark or zone equal.

6. When a chained branch instruction specifies only the op code, the contents of the AAR, BAR and op modifier register from the previous operation designate the I-address, B-address, and the d-character, respectively.

Answers to Questions on Store Address Register Operation

1. The C-address in the store address register instruction is the address of the units position of the C-field, the location in which the units position of the designated address register is stored.

2. The d-character in the store address register instruction designates the address register whose contents are to be stored.

3. Contents of the AAR, BAR, EAR, and FAR can be stored in store address register operations.

4. C-field word marks have no effect on the store address register operation.

5. The store address register operation ends at A-ring 6 time after ten thousands position of the selected address register is stored.

6. The character in the units position of the selected address register is stored at A-ring 2 time.

Answers to Questions on Set Word Marks Operation

1. The set word marks instruction causes the CPU to store word marks in designated core storage locations.

2. Execution of the set word marks instruction does not alter any core storage character.

3. If the set word marks instruction contains only one address, the same address is set in the AAR and the BAR, causing a word mark to be set twice in the same location.

4. When a word mark is added to a character, a check bit is added or removed to maintain parity.

Answers to Questions on Clear Word Marks Operation

1. The clear word marks instruction causes the CPU to strip word marks in designated core storage locations.

2. Execution of the clear word marks instruction does not alter any core storage character.

3. If the clear word marks instruction contains only one address, the word mark is stripped twice from the same location.

4. When a word mark is stripped from a character, a check bit is added or removed to maintain parity.

Answers to Questions on Clear Storage Operation

1. If the B-address in the clear storage instruction is 09050, the clear storage operation clears 51 storage positions.

2. If the clear storage instruction does not contain a B-address, the AAR is not loaded during instruction read out.

3. If the clear storage instruction contains a B-address, the B-address is loaded in the AAR and the BAR during instruction read out. However, during execute phase of the clear storage operation, the AAR is not used, and contains the original B-address when the clear storage operation ends.

4. The minus one condition ON at logic gate D time, indicating a borrow one from the hundreds position, defines the even hundreds position; the even hundreds latch is set, and the clear storage operation ends.

Answers to Questions on Clear Storage and Branch Operation

1. The clear storage and branch operation causes the CPU to branch to the specified I-address when storage positions through the nearest hundreds address have been cleared.
2. The I-address is loaded in the AAR during instruction read out.
3. The clear storage and branch operation causes an unconditional branch.
4. Completion of the branch operation is accomplished during the subsequent instruction read-out operation.

Answers to Questions on Halt Operation

1. When system operation resumes after the logic clock stops, the next sequential instruction is executed.
2. A word mark must be present in the storage location immediately to the right of the halt op code to distinguish the halt instruction from the halt and branch instruction during instruction read out.
3. The stop print-out operation begins when the logic clock stops if:
   a. The print control switch is not set to inhibit, and
   b. The mode switch is not in the display or alter position.
4. No, the logic clock does not start automatically when the print-out operation ends.

Answers to Questions on Halt and Branch Operation

1. When system operation resumes after the logic clock stops, the CPU executes the instruction that the I-address in the halt and branch instruction designates.
2. The branch to AAR latch is set, initiating the branch operation, at the end of instruction read out.
3. If the start key is pressed before the stop print out operation ends, no action occurs until the print-out is complete.
4. The branch operation that the halt and branch operation causes is an unconditional branch.

Answers to Questions on No Operation Instruction

1. When the N op code is set in the operation register at I-op time, the I-ring does not advance.
2. When the N op code is detected, other characters are read out of core storage and ignored until a B-channel word mark is detected.
3. Because the add op code does not contain a word mark, the add instruction will not be executed.

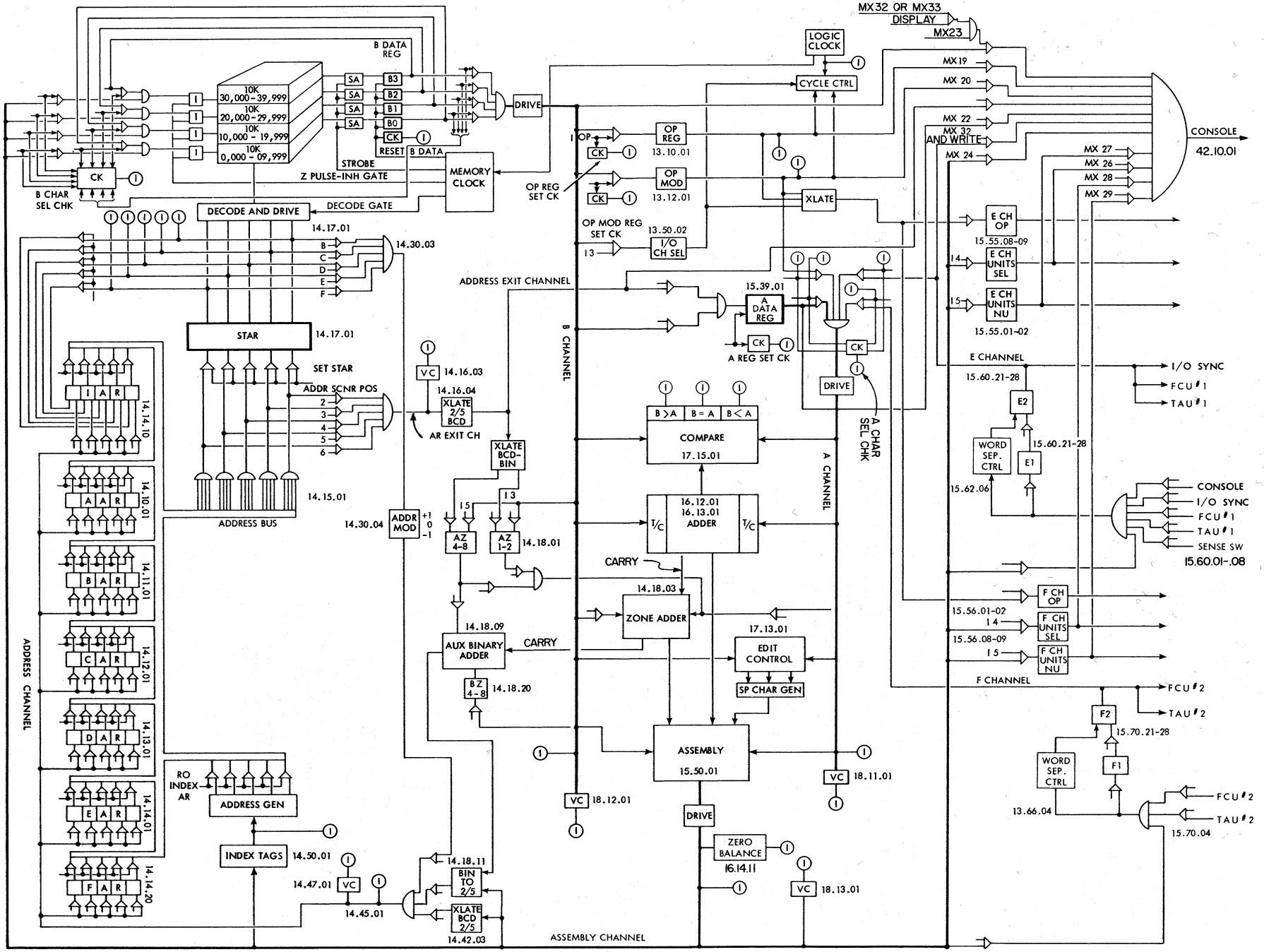
Reference Index

Instruction Type	Instruction	Op-Code*	Instruction Format	Data Flow Chart	Timing Chart	Circuit Controls	Acceptable Instruction Lengths	Address	Registers IAR	After AAR	Operation BAR	Comments
LOGIC OPERATION INSTRUCTIONS	Unconditional Branch	v J	Page 71	Page 72	Page 73	Page 71	7		NSIB	BI	NSI	Causes a program branch to the storage location specified by 1-address.
	Test and Branch	v J	72	74	Not Included	71 72	7	Branch	NSIB	BI	NSI	CONDITIONAL BRANCH INSTRUCTIONS: Cause program branch to the storage location specified by the 1-address when the condition designated by the d-character in the instruction is satisfied. If the condition designated by the d-character in the instruction is not satisfied, the system executes the next sequential instruction. Figures 34, 36 and 40 list special d-characters for the test and branch, branch if I-O channel status indicator on, and branch on word mark or zone equal instructions, respectively.
								No Branch	NSI	BI	BI	
	Branch if I-O Channel Status Indicator On	v R (ch 1) v X (ch 2)	73	75	77	71 74	7	Branch	NSIB	BI	NSI	
								No Branch	NSI	BI	BI	
	Branch if Character Equal	v B	75	76	77	71 75	1 6 12	Branch	NSIB	BI	NSI	
								No Branch	NSI	BI	B-1	
	Branch if Bit Equal	v W	78	Not Incl.	77	71 78	1 6 12	Branch	NSIB	BI	NSI	
								No Branch	NSI	BI	B-1	
	Branch on Word Mark or Zone Equal	v V	78	79	Not Included	71 80	1 6 12	Branch	NSIB	BI	NSI	
								No Branch	NSI	BI	B-1	
MISCELLANEOUS INSTRUCTIONS	Store Address Register	v G	81	82	83	81	7		NSI	Ap	Bp	Causes the system to take C-cycles to store the contents of the AAR, BAR, EAR, or FAR as designated by the d-character in the instruction.
	Set Word Mark	v '	84	85	86	84	1	No-address	NSI	Ap-1	Bp-1	Sets word marks in storage locations designated by AAR and BAR (loaded in previous operation).
							6	1-address	NSI	A-1	A-1	Sets word marks in storage location designated by A-address in instruction.
							11	2-addresses	NSI	A-1	B-1	Sets word marks in storage locations designated by A- and B-address in instruction.
	Clear Word Mark	v ¤	86	87	86	87	1	No-address	NSI	Ap-1	Bp-1	Removes word marks in storage locations designated by AAR and BAR (loaded in previous operation).
							6	1-address	NSI	A-1	A-1	Removes word mark in storage location designated by A-address in instruction.
							11	2-addresses	NSI	A-1	B-1	Removes word marks in storage locations designated by A- and B-addresses in instruction.
	Clear Storage	v /	88	89	92	88	1	No-address	NSI	Ap	bbb00-1	Clears data and word marks (right-to-left) from the core storage location specified by the BAR to, and including the nearest hundreds position.
							6	1-address	NSI	B	bbb00-1	
	Clear Storage and Branch	v /	90	91	92	90	11		NSIB	BI	NSI	Clears data and word marks (right-to-left) from the core storage location specified by the BAR to, and including the nearest hundreds position and branches unconditionally to the location specified by the I-address in the instruction.
	Halt	v .	93	51	Not Included	93	1		NSI	Ap	Bp	Causes the system to stop. Pressing the start key starts the system with the next sequential instruction.
	Halt and Branch	v .	94	96	Not Included	95	6		NSIB	BI	NSI	Causes the system to stop. Pressing the start key starts the system with the instruction designated by the I-address in the instruction.
	No-Operation	v N	95	97	98	95	1		NSI	Ap	Bp	The N operation code can be substitued for the operation code of any instruction to make that instruction ineffective. Instructions following the N operation code are skipped until a word mark in storage is detected.

A A-address of the instruction
B B-address of the instruction
NSI Address of the next sequential instruction
BI Address of the next instruction if a branch is taken
NSIB Address of the next instruction in normal sequence after a branch has been taken
LA The number of characters in the A-field
LB The number of characters in the B-field
LW The number of characters in the A- or B-field, whichever is shorter
Ap The previous contents of the A-address register
Bp The previous contents of the B-address register
* a word mark must be set over the op-code in an instruction

Instruction Type	Instruction	Op-Code* type	Instruction Format	Data Flow Chart	Timing Chart	Circuit Controls	Acceptable Instruction Lengths	Address	Registers	After	Operation	Comments
									IAR	AAR	BAR	
ARITHMETIC INSTRUCTIONS	Add	v A	Page 8	Page 11	Page 12 13	Page 10	1 6 11	2-fields	NSI	A-LW	B-LB	Algebraically adds numeric data in A-field to numeric data in B-field. Sum stored in B-field.
								1-field	NSI	A-LA	A-LA	Algebraically doubles numeric data in A-field. Sum stored in A-field.
	Subtract	v S	8	11	7A 7B	10	1 6	2-fields	NSI	A-LW	B-LB	Algebraically subtracts numeric data in A-field from numeric data in B-field; difference stored in B-field.
							11	1-field	NSI	A-LA	A-LA	Algebraically subtracts numeric data in A-field from itself; difference is stored in A-field.
	Zero and Add	v ?	14	16	17		1 6	2-fields	NSI	A-LW	B-LB	Numeric data in A-field is stored in B-field. Sign of result field (B-field) is same as sign of A-field.
							11	1-field	NSI	A-LA	A-LA	Strips A-field of all other zones except in units (sign) position. Sign of A-field is unchanged.
	Zero and Subt	v !	14	16	17	15	1 6	2-fields	NSI	A-LW	B-LB	Numeric data in A-field is stored in B-field. Sign of results field (B-field) is opposite sign of A-field.
							11	1-field	NSI	A-LA	A-LA	Strips A-field of all other zones except in units (sign) position. Sign of A-field is inverted.
	Multiply	v @	18	24,25 26,27	23	Not Included	1 6 11		NSI	A-LA	B-LB	Repetitively adds numeric data in A-field (multiplicand) and stores result in B-field (product), starting with low-order positions.
	Divide	v %	28	35 36	Not Included	Not Included	1 6 11		NSI	A-LA	tens position of quotient field	Causes dividend (in B-field) to be divided by divisor (in A-field). Quotient is stored in high-order B-field positions.
GENERAL DATA INSTRUCTIONS	Move Data	v D	37	39 40	41	38	1 6 12	SEE FIGURE 18				Moves data serially by character from A-field to B-field under control of the d-character.
	Move Char and Suppress Zeros	v Z	42	44	45	43	1 6 11		NSI	A-LA	B+1	Moves A-field data to B-field. Insignificant zeros and commas are replaced with blanks; zone bits in units position of B-field are removed. A-field data is unchanged after the operation.
	Edit	v E	47	54 55 56	57	51	1 6 11		NSI	A-LA	varies with result of edit	Edit control field (B-field) modifies the data field (A-field). Results stored in the B-field.
	Compare	v C	61	62	63	64	1 6 11		NSI	A-LW	B-LW	Compares data in the B-field to A-field data (the comparison is always B to A). All other bits except C and WM-bits in each character are compared.
	Table Lookup	v T	66	67	68	67	1 6 11		NSI	A-LA	address of function at immediate left of table argument that stopped the operation	Causes the system to search through a previously prepared table in core storage and find the desired factor or the address of the desired factor.

A A-address of the instruction
B B-address of the instruction
NSI Address of the next sequential instruction
BI Address of the next instruction if a branch is taken
NSIB Address of the next instruction in normal sequence after a branch has been taken
LA The number of characters in the A-field
LB The number of characters in the B-field
LW The number of characters in the A- or B-field, whichever is shorter
Ap The previous contents of the A-address register
Bp The previous contents of the B-address register
* A word mark must be set over the op-code in an instruction


MX32 OR MX33 DISPLAY
MX23
LOGIC CLOCK
CYCLE CTRL
MX19
MX 20
MX 22
MX 32 AND WRITE
MX 24
MX 27
MX 26
MX 28
MX 29
CONSOLE 42.10.01
B DATA REG
10K 30,000-39,999
10K 20,000-29,999
10K 10,000-19,999
10K 0,000-09,999
SA
B3
B2
B1
B0
CK
RESET
B DATA
STROBE
Z PULSE-INH GATE
DRIVE
MEMORY CLOCK
B CHAR SEL CHK
DECODE AND DRIVE
DECODE GATE
14.17.01
14.30.03
I OP
OP REG 13.10.01
OP REG SET CK
OP MOD 13.12.01
OP MOD REG SET CK
13.50.02
I/O CH SEL
XLATE
E CH OP 15.55.08-09
E CH UNITS SEL
E CH UNITS NU 15.55.01-02
ADDRESS EXIT CHANNEL
15.39.01
A DATA REG
A REG SET CK
A CHAR SEL CHK
B CHANNEL
A CHANNEL
E CHANNEL 15.60.21-28
I/O SYNC
FCU#1
TAU#1
E2
E1
WORD SEP. CTRL
15.62.06
CONSOLE
I/O SYNC
FCU#1
TAU#1
SENSE SW
15.60.01-.08
STAR
SET STAR
ADDR SCNR POS
14.15.01
ADDRESS BUS
14.16.03
14.16.04
XLATE 2/5 BCD
AR EXIT CH
XLATE BCD-BIN
B>A
B=A
B<A
COMPARE 17.15.01
16.12.01 16.13.01 ADDER
ADDR MOD
14.30.04
AZ 4-8
AZ 1-2
14.18.01
CARRY
14.18.03
ZONE ADDER
17.13.01
EDIT CONTROL
SP CHAR GEN
14.18.09
AUX BINARY ADDER
BZ 4-8
14.18.20
F CH OP 15.56.01-02
F CH UNITS SEL
F CH UNITS NU 15.56.08-09
F CHANNEL
FCU#2
TAU#2
F2
15.70.21-28
F1
WORD SEP. CTRL
13.66.04
15.70.04
ADDRESS CHANNEL
IAR 14.14.10
AAR 14.10.01
BAR 14.11.01
CAR 14.12.01
DAR 14.13.01
EAR 14.14.01
FAR 14.14.20
RO INDEX AR
ADDRESS GEN
INDEX TAGS 14.50.01
14.47.01
14.45.01
14.18.11
BIN TO 2/5
XLATE BCD 2/5
14.42.03
ASSEMBLY 15.50.01
VC 18.12.01
VC 18.11.01
VC 18.13.01
DRIVE
ZERO BALANCE 16.14.11
ASSEMBLY CHANNEL

COMMENT SHEET

IBM 1411 PROCESSING UNIT INSTRUCTIONS AND SPECIAL FEATURES

CUSTOMER ENGINEERING INSTRUCTION–REFERENCE, FORM 223–2698

FROM

NAME _____

OFFICE NO. _____

FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

FOLD

☐ SUGGESTED ADDITION (PAGE , TIMING CHART, DRAWING, PROCEDURE, ETC.)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

CUT ALONG LINE

FOLD

FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY
IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N.Y.

ATTN: CE MANUALS, DEPARTMENT B96

CUT ALONG LINE

FOLD FOLD

STAPLE STAPLE

223-2698

Printed in U.S.A. 223-2698

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York