COPY # 10

UNIVAC ® CALCULATING TABULATOR

PROCESSOR MANUAL

SECTIONS: 4 & 5
APPENDIX: C

REMINGTON RAND UNIVAC
DIVISION OF SPERRY RAND CORP.
Philadelphia, Pa.
March 1958

SECTION IV

TABLE OF CONTENTS

ILLUSTRATIONS

TABLES

SECTION IV

THEORY OF OPERATION

4-1. Section II, by explaining the use of logical symbols and elements, enables the reader to interpret the logical circuitry of the processor. Section III, by describing the logical components of the processor, familiarizes the reader with the specific circuits encountered in this section, section IV, and in the Analysis of UCT Instructions manual. The purpose of this section is to apply the knowledge gained in the foregoing sections to actual computer operations in which more than just an understanding of the logical circuits is necessary. Types of instructions not detailed in this section are treated on the more general level in section V. All instructions are treated in complete detail in the Analysis of UCT Instructions manual.

4-2. BASIC OPERATION CYCLE

4-3. The basic operation cycle consists of the steps necessary to perform any instruction completely. All instructions require at least three basic steps. Instructions requiring only three steps are those in which no operand is involved, such as a simple register-to-register transfer. Most instructions, however, require four basic steps because of the need for an operand. For this reason, a basic operation cycle consists of four steps. In instructions requiring only three steps the third or search-for-operand step is bypassed. The four steps of the normal cycle, with the search for the instruction as the starting point, are:

(1) Search for the instruction (Sc).

(2) Staticize the instruction (Sz).

(3) Search for the operand (Sc-m).

(4) Execute the instruction (Ex).

4-4. When an instruction is completed, an ending pulse is generated which clears the static register and starts the next cycle by initiating a search for the next instruction in the program (search for the instruction).

When an instruction has been located on the drum it is read from storage by the read circuits and is stored in the static register and register C (rC) (staticize the instruction). If the instruction requires an operand another search is made for this quantity (search for the operand). When the operand is located, the first step of the instruction is executed (execute the instruction). Any further steps in the instruction are executed in order after the first execution step. At the completion of the execution of the instruction, an ending pulse is produced to start the sequence once more.

4-5. SEARCH-FOR-INSTRUCTION STEP

4-6. After an instruction has been completed a signal is applied to the ending pulse buffer of the static register. As shown in figure 4-1, the ending-pulse buffer generates the EP signal which is applied to the restore gates of the seven static-register flip-flops. The high EP signal restores all seven flip-flops to their barred outputs, which indicate 0 bits at t11B of the execute step of the preceding instruction. The new outputs of the static register drive the instruction decoder.

4-7. Only the search (Sc) gateline of the instruction decoder is permissive to the 0 bits. The search gate interprets the two 0's in the two lowest-order bit positions, STR1 and STR2, and generates the 1A function signal. Function signal (FS) 1A drives the function encoder and generates function signals 1, 58+, 63 and 74. These four function signals and FS 1A control the search-for-instruction step.

4-8. In the search-for-instruction step, the storage address specified by the c address in rC must be located, the specified band selected, and the memory-read circuits alerted to read the instruction word contained in the address.

4-9. LOCATING THE MEMORY ADDRESS. In the search for a memory address the following must be determined: the word channel and drum quadrant containing the desired address, the proper band, and the read-write head which is to be energized to read the contents of the location. One of four heads must be selected in reading from a fast-access band. In normal access, however, only one head can be

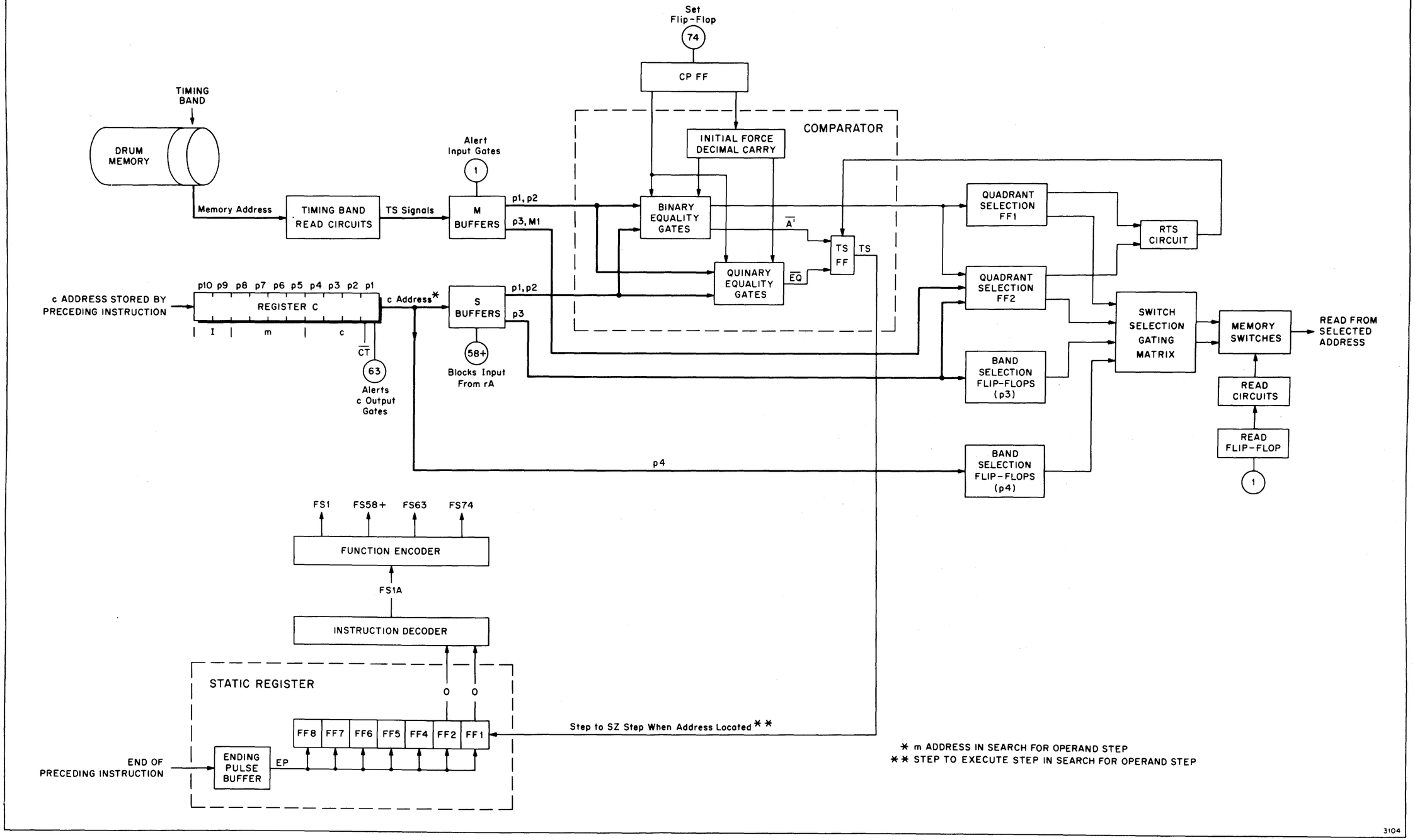


Figure 4-1. Search Step of Basic Operation Cycle

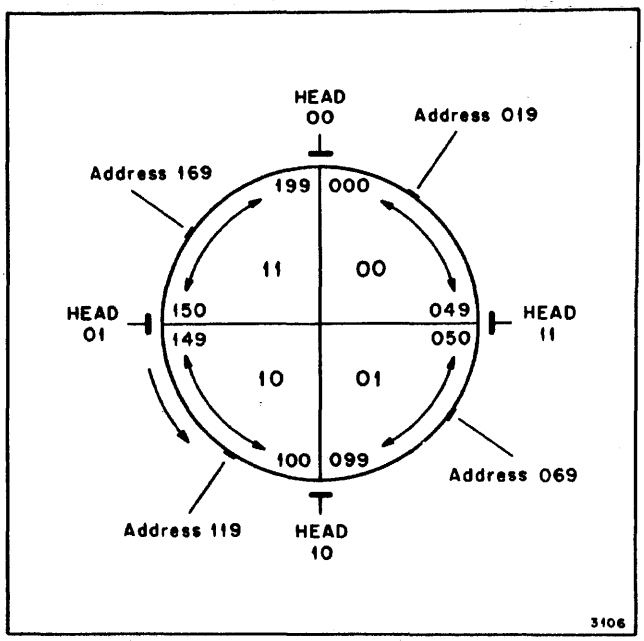
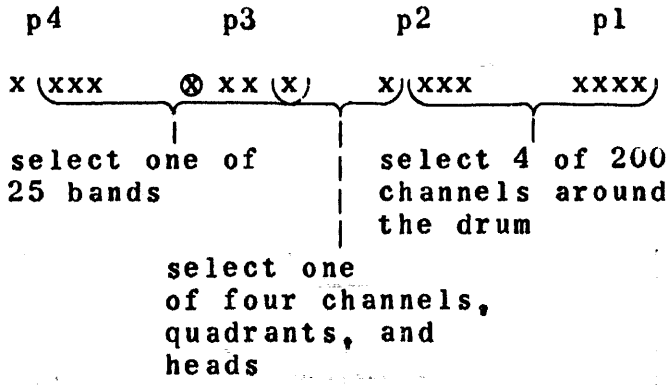


Figure 4-2. Drum Quadrants

selected since there is only one head per track. In a search for an address, the bits of the four c-address digits stored in rC are used as follows:



4-10. Because there are four heads spaced at 90 degree intervals around the drum in fast-access storage, it is necessary to theoretically divide the surface or circumference of the drum into quadrants. It is necessary to refer to quadrants in normal access because both types of access storage use the same circuits. One of the four fast-access heads must be energized to read or write at the time the quadrant containing the desired memory address is approaching that head, minimizing the time required to read from or write onto the drum.

4-11. The p1 digit and the three lowest-order bits of p2 partially represent the channel across the drum in which the correct address is found. If the complete address is 4469, p1 represents the 9 digit. However, the 3 lowest-order bits of p2 do not fully represent the 6 digit; they merely show that the digit is either 1 or 6. Since addresses 000 through 199 are found within a band, the word can be in any one of channels 019, 069, 119, or 169. These four possible combinations for the time selection bits are shown in figure 4-2. The most significant bit (MSB) of p2 and the MSB and least significant bit (LSB) of p3 choose one of the quadrants, completing the selection of a channel and also selecting the head to be energized for reading. The remaining bits of digit p3 and the bits of digit p4 select the proper band.

4-12. Digit p2 of 4469 is 6,1001 in biquinary code. The three lowest-order bits, 001, are common to both 1 and 6 in biquinary code. Therefore if the MSB of p2 is a 0, the address is either 0$\underline{1}$9, or 1$\underline{1}$9; if it is a 1, the address is either 0$\underline{6}$9 or 1$\underline{6}$9. In address 4469 the MSB of p2 is 1; therefore the address is in either quadrant 01 or 11 (addresses 169 or 069) as shown in figure 4-3.

Table 4-1. Oddness and Evenness of Biquinary Combinations

	Biquinary Code MSB LSB	Decimal Equivalent	Even or Odd[a] MSB LSB
	0 0 0 0	0	0 and 0 = 0
	0 0 0 1	1	0 and 1 = 1
	0 0 1 0	2	0 and 0 = 0
	0 0 1 1	3	0 and 1 = 1
p3 ——>	0 1 0 0	4	0 and 0 = 0
	1 0 0 0	5	1 and 0 = 1
	1 0 0 1	6	1 and 1 = 0
	1 0 1 0	7	1 and 0 = 0
	1 0 1 1	8	1 and 1 = 0
	1 1 0 0	9	1 and 0 = 1

[a] 0 = even; 1 = odd

4-13. Address 069 has an even-hundreds digit; address 169 has an odd-hundreds digit. The least-significant bit of p3 designates one of the two quadrants selected by the MSB of p2. Selection is accomplished by combining the LSB of p3 with the MSB of p3 to ascertain the evenness or oddness of p3.

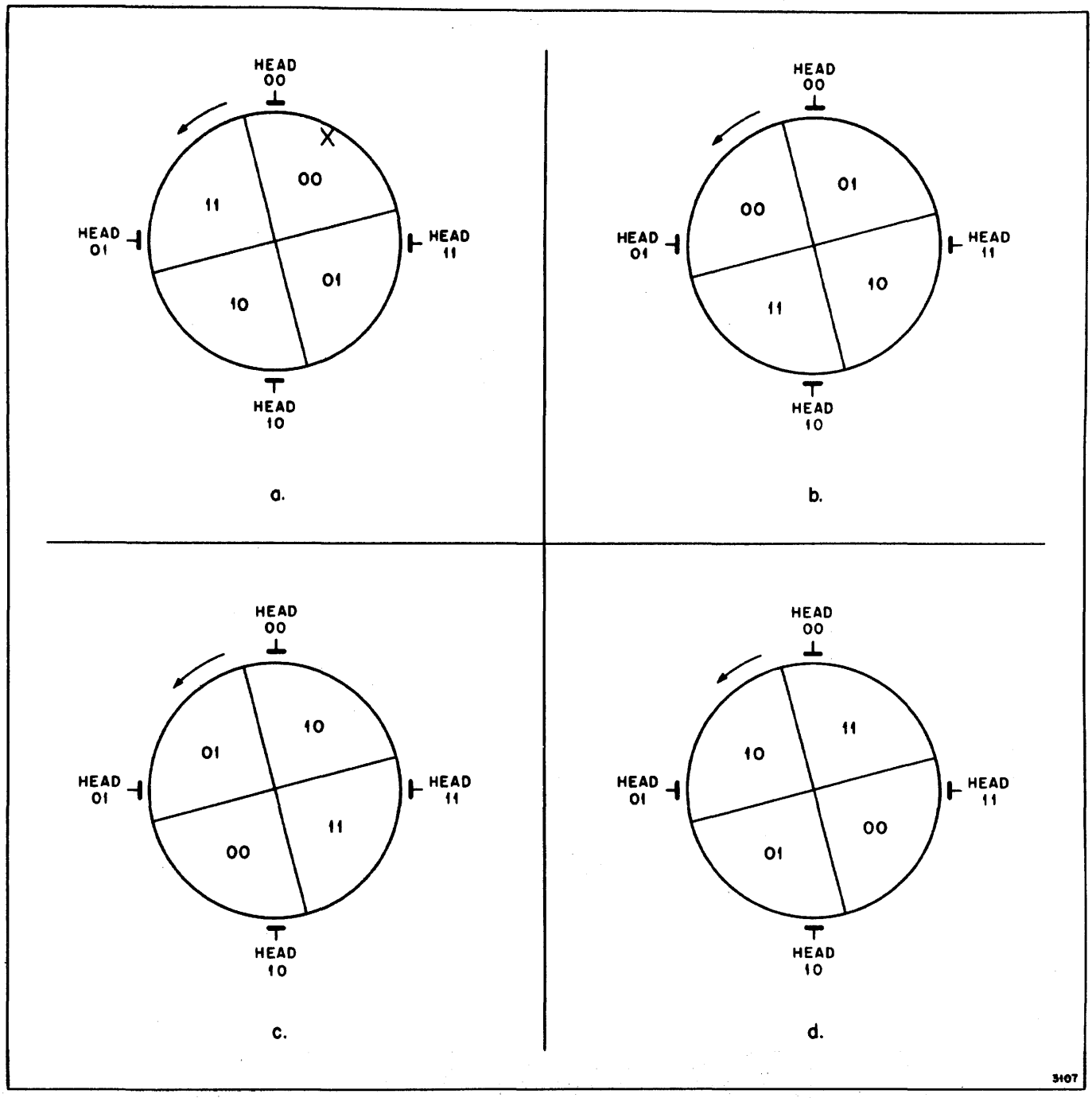


Figure 4-3. Drum Quadrants under Fast-Access Heads

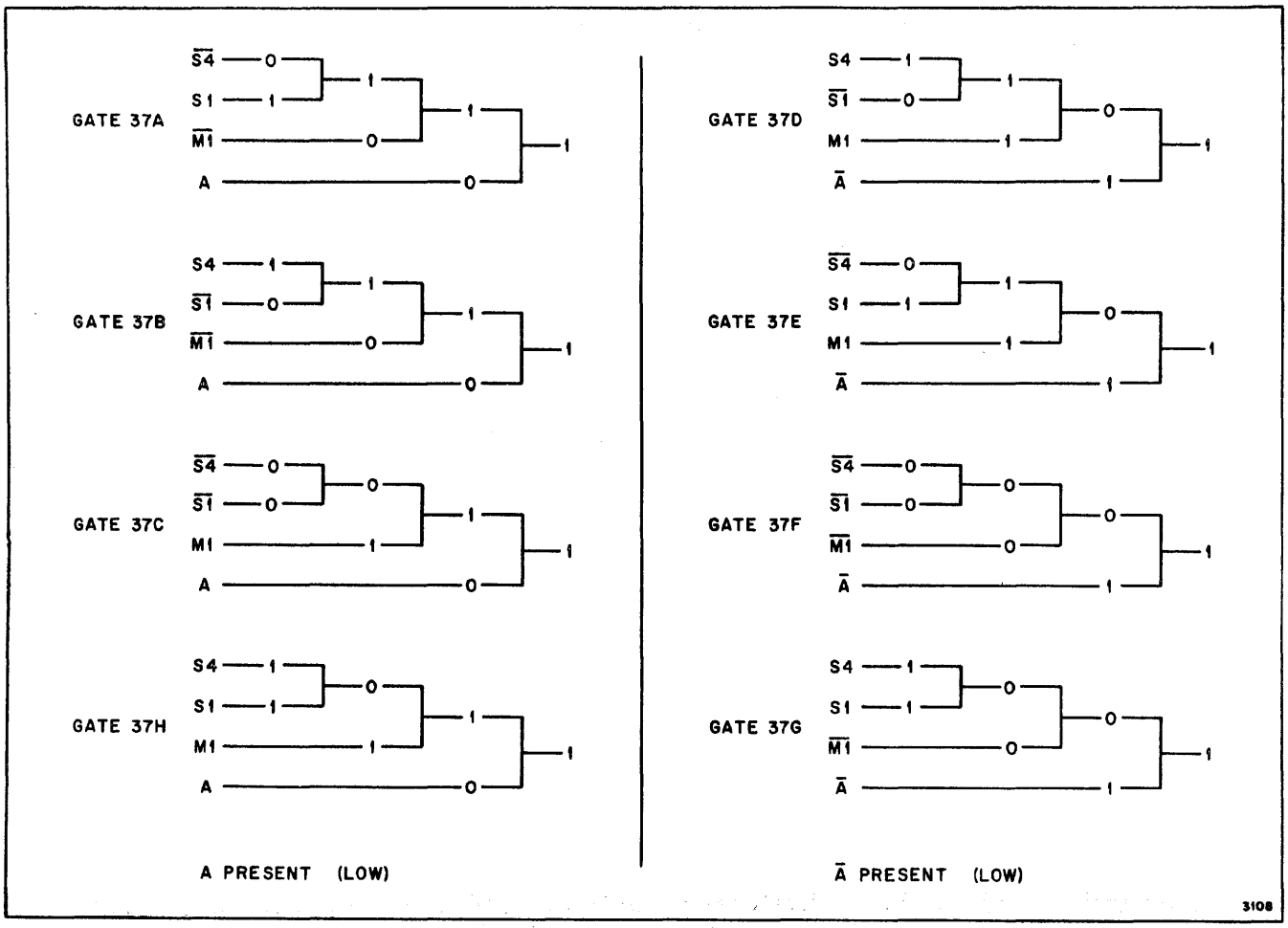


Figure 4-4. Quarter Addition of QS2 FF Inputs

4-14. This combination of the MSB and LSB of p3 is necessary because the least-significant bits of the biquinary code do not form a pattern by which the oddness or evenness of a digit can be determined. The first column of table 4-1 shows that the LSB of p3 (0), if taken alone could indicate either an odd or even decimal digit. The last column shows that by combining the LSB and MSB of the combinations for each digit, a definite pattern is formed. When the combined bits equal 0, the digit is even; when they equal 1, the digit is odd.

4-15. In the 4469 address digit, p3 is 0100 (4) which is an even-digit combination (0). Therefore digit p3 signifies that the address is contained in a location having an even hundreds digit, such as $\underline{0}69$, $\underline{2}69$, $\underline{4}69$, $\underline{6}69$, or $\underline{8}69$. The MSB of p2 already has designated either the $\overline{169}$ or 069 channel. The p3 digit, being even, specifies the 069 channel and therefore quadrant 01.

4-16. The MSB of p2 and the LSB of p3, both of which determined the correct quadrant, also select one of the four heads in a fast-access band. With the 000 address in a band under the 00 head, the four quadrants and four heads are as shown in figure 4-2. To determine the head to be energized, the MSB of p2 and the LSB of p3 from rC and the timing combination from memory are quarter added. Table 4-2 lists the possible combinations of these bits. The four conditions shown under table 4-2a refer to the drum position in figure 4-3a; that is, quadrant 00 is under head 00. The conditions under table 4-2b refer to the possible conditions when quadrant 01 is approaching head 00. Similarly, table 4-2c and figure 4-3c show the possible conditions when quadrant 10 is approaching head 00; table 4-2d shows the possible conditions when quadrant 11 is approaching head 00.

4-17. The cutaway view of the drum in figure 4-4a shows the conditions specified in table 4-2a. It must be remembered that the quadrant called for in the c address is represented by the two quadrant selection bits of rC, that is, the MSB of p2 and the LSB of p3. If the 00 quadrant is under the 00 head, as in all four cases of table 4-2a, and the 00 quadrant is called for by rC, combination of the two quadrants results in selection of the 00 head. If the desired address is X in quadrant 00, the 00 head must be selected as the drum rotates in a counterclockwise direction. If the drum quadrants are in the same position and quadrant 11 is specified in the c address, the 01 head must be selected.

Table 4-2. Head Selection

Timing Band Combinations p3 M1	p2 M4	c Address p3 S1 S4	p2 S4	Head Selected	Table Section
0	0	0	0	00	a
0	0	0	1	11	
0	0	1	0	10	
0	0	1	1	01	
0	1	0	0	01	b
0	1	0	1	00	
0	1	1	0	11	
0	1	1	1	10	
1	0	0	0	10	c
1	0	0	1	01	
1	0	1	0	00	
1	0	1	1	11	
1	1	0	0	11	d
1	1	0	1	10	
1	1	1	0	01	
1	1	1	1	00	

4-18. With any of the other three quadrants under the 00 head, as in figure 4-3a, 4-3b and 4-3c, head selection again depends on the quadrant to be read. For example, in figure 4-3c, the 10 quadrant is under the 00 head. Therefore, if the 11 quadrant is to be read, the 11 head, which is most immediately in the path of counterclockwise rotation of the drum and quadrant 11, is selected. The same conditions in table 4-2c (last line) verify the head selected in figure 4-3c.

4-19. Thus far, digits p1, p2, and part of p3 have selected the proper memory channel, the quadrant containing the address, and one of four heads to be energized in reading that address. Digit p3 and the three LSB's of p4 determine which of five fast-access bands contain the address. Digit p3 determines the third digit, the hundreds digit, of the address; the three lowest-order bits of p4 determine the fourth, or thousands digit of the address. A detailed explanation of the memory search process is given in paragraph 4-20.

4-20. COMPARISON AND HEAD·SELECTION. The comparison operation (figure 4-1) begins with the comparison of the timing band channel addresses with the address contained in rC. As shown in the figure, the timing band signals enter the comparator on the M lines from the M buffers; the c-address signals from rC enter on the S lines from the S buffers. (Figure 4-1 and the logical diagrams in Appendix A are pertinent throughout this description.)

4-21. Function signal 63 alerts the c-address output gates of rC. The gates are made permissive by the $\overline{CT}$ signal from the conditional transfer flip-flop (CT FF). The CT FF is restored to $\overline{CT}$ by the RCT outputs of the static register at the end of every search operation. Digits p1, p2, and p3 of the c address in rC are transferred, through the permissive output gates, to the S buffers. The p4 digit goes directly from rC to the band-selection flip-flops. The output signals of the four output gates of rC (figure A-1) are C12, C22, C32, and C42. The c address from rC can enter the S buffers and the S lines because FS 58+ has blocked the (rA) outputs which normally enter the S buffers. The p1 and p2 digits of the c address are transferred on the S lines into the quinary and binary equality gates of the comparator.

4-22. Function signal 1 alerts the input gates of the M buffers from t0 to t2B. The timing band combinations signals TS1 through TS4 from the timing-band read circuit enter the M buffers and the M lines.

4-23. Function signal 74 sets the complement (CP) FF of the sign-and-control circuit to produce a complement (CP) signal for the comparison operation. The CP signal alerts the binary and quinary equality gates of the comparator. Function signal 74 also generates a CP5 signal from the CP FF for use in the comparison operation. The CP5 signal drives the initial force-decimal-carry circuit which generates the A and C decimal-carry signals. These signals alert the quinary equality gates and binary carry gates of the computer.

4-24. The p1 digits from rC and memory arrive on the M and S lines at time t1B. The quinary bits of p1 (M1, M2, M3, and S1, S2, S3) are compared in the quinary equality gates of the comparator. Equality between the three M bits and the three S bits makes one of the eight gates permissive and causes a high output from one of the quinary equality gates. The high output goes to a non-complementing amplifier which produces a high $\overline{EQ}$ output at t2B.

4-25. Before the comparison operation at t11B of the previous word time the time selection flip-flop (TS FF) (figure 4-1) was set. The set state of the TS FF indicates equality of the M and S bits. Inequality of the M and S bits restores the TS FF, indicating that the timing-band address under the TS head at that instant does not match the c address in rC. At various times the TS output is sampled to see whether a match has been obtained. Restoring the TS FF causes another search cycle to be initiated at the time TS is sampled. If the flip-flop remains set, it indicates equality of the timing address bits with the c-address bits. The TS FF input gate 21 (figure A-12) restores the flip-flop to $\overline{TS}$ only when $\overline{EQ}$ is low at time t2B or t3B. The $\overline{EQ}$ output is low only when the quinary bits are unequal, resulting in a low output from the quinary equality gates. The high $\overline{EQ}$ signal blocks input gate 21 to keep the TS FF set.

4-26. The binary bits of p1 also are compared in the binary equality gates of the comparator at the same time (t1B) as the quinary bits are compared. Equality of bits M4 and S4 in the gates alerted by CP produces a high output signal which is buffed into a non-complementing

amplifier to generate a high $\overline{A'}$ at t2B. This indication of equal binary bits goes to gate 20, an input gate of the TS FF. The high $\overline{A'}$ signal blocks the gate and the TS FF remains set.

4-27. Inequality of M4 and S4 cause the binary equality circuit to produce a low output to the non complementing amplifiers. The output of the amplifiers is a low $\overline{A'}$ signal. Input gate 20 of the TS FF is permissive to low signals at time t2B, and the low $\overline{A'}$ signal restores the TS FF to indicate inequality.

4-28. At time t2B, the quinary bits of digit p2 are compared for equality in the quinary equality gates. The comparison is accomplished in the same manner as for the quinary bits of p1. The high $\overline{EQ}$ signal, indicating equality of the quinary bits of p2, occurs at t3B and keeps the TS FF set.

4-29. The binary bits of digit p2, M4 and S4, are compared in the binary equality gates which are alerted by the low $\overline{CP}$ signal at t2B. Equality of the bits produces a high $\overline{A'}$ signal. The most significant bit of p2 indicates whether the desired address is above or below 50; it therefore contributes to quadrant selection.

4-30. The high $\overline{A'}$ signal goes to the quadrant selection 1 flip-flop (QS1 FF) of the memory selection circuit (figure A-18). The signal blocks gate 33 of the QS1 FF, keeping the flip-flop restored to a low $\overline{QS1}$ output. The low $\overline{QS1}$ output indicates a 0 in the LSB of the quadrant address: either 00 or 10. If the $\overline{A'}$ signal on gate 33 is low at time St3B, the gate operates to set the flip-flop to a low QS1 output. The QS1 signal indicates a 1 in the LSB of the quadrant address: either 01 or 11. The QS1 FF output goes to the switch-selection gating matrix.

4-31. Also at St3B, the input gates of the QS2 FF sample the conditions which affect the selection of the MSB of the quadrant address. The output of the QS2 FF is the MSB of the two-digit head address shown in table 4-2.

4-32. All of the input gates of QS2 FF logically quarter add their inputs. For example, gate 37E of the flip-flop is permissive only to low signals $\overline{S4}$, S1, M1, and $\overline{A}$ (figure 4-4). The $\overline{S4}$ and S1 signals are the MSB and LSB

of the p3 digit. Quarter addition of these two bits produces a 1. The M1 signal indicates a 1 bit, and it is quarter added to the result of the quarter addition of $\overline{S4}$ and S1 bits. Quarter addition of a 1 bit ($\overline{S4}$, S1) to a 1 bit (M1) again produces a 0 bit. The $\overline{A}$ signal, which has a value of 1, is quarter added to the previous bit result to produce a final result of 1. The QS2 FF is, as a result, set to a QS2 output to select a quadrant address with a MSB of 1. If either bit of the quadrant address selected by the quadrant selection flip-flops is a 1, the fast-access bands are involved. If the 00 quadrant is selected, normal access may be involved.

4-33. The low $\overline{A}$ signal has a value of 1 when the p2 digit from the timing band is less than 5 and the p2 digit of rC is 5 or larger. The low A signal has a value of 0 under all conditions except those associated with the $\overline{A}$ signal. Gates 28, 29 and 31 of the comparator (figure A-12) determine from comparison of the MSB's of p2 whether the A or the $\overline{A}$ signal is low. If one of the inputs to each one of these gates is high, the $\overline{A}$ signal is low. This can occur only when the M4 bit is a 0 and the S4 bit is a 1. If this condition is not present, one of the gates is permissive to the low inputs, generating a low A signal. The low A signal is present when the MSB of the timing-band p2 digit is equal to the MSB of the rC p2 digit or if the MSB of the timing-band p2 digit is a 1 and the MSB of the rC p2 digit is a 0.

4-34. The conditions necessary to set the QS2 FF to a QS2 (1) output are shown in figure 4-4. If none of the eight gates is permissive, the flip-flop is restored to a $\overline{QS2}$ (0) output. The outputs of the QS1 and QS2 FF's go to the memory switch selection matrix together with the outputs of the band-selection flip-flops.

4-35. If the quadrant address selected by the outputs of QS1 FF and QS2 FF is 10, 01, or 11, the input gate which produced the 1 output (a high output is necessary for 1) also causes a high signal to go to amplifier 39 to generate a low RTS signal. The low RTS signal goes to gate 22 of the TS FF (figure A-12).

4-36. The signals which make gate 22 permissive are low $\overline{S3}$ and $\overline{S4}$ which indicate 0's in the S3 and S4 bit positions of p4. When S3 and S4 are 0, addresses 0000 through 3999 (normal access) are involved; when S3 is 1, addresses 4000 through 4999 (fast access) are involved.

4-37. If either of the quadrant selection flip-flops has, by generating RTS, indicated a fast-access head and the low $\overline{S3}$ and $\overline{S4}$ bits indicate a normal-access location, gate 22 (figure A-18) restores the TS FF. When the QS1 and QS2 flip-flops select a 00 head, the RTS signal is high. The high RTS signal blocks gate 22, keeping the TS FF set so that the selected head can be energized to read or write.

4-38. When the search operation is unsuccessful, a high TS signal is generated by the restored TS FF and sent to input gate 7 of static register flip-flop 1. The high input blocks the gate to keep the flip-flop set to a 0 bit. The flip-flop cannot be stepped to a 1 bit output until the TS input is low at t5B or t10B.

4-39. During instructions H, X, and J, which involve writing on the drum because they are transfer-to-memory instructions, the TS signal is sampled at gate 8 of static-register flip-flop 1. A low TS signal indicates that the timing band combination matches that of the c or m address in rC.

4-40. BAND SELECTION. The final step in locating an address is the selection of the band specified by the p3 and p4 digits in rC. The correct band is selected by digit p3 and the three lowest-order bits of p4. These seven bits indicate the hundreds and thousands digits of the address and whether the address is in fast- or normal-access storage.

4-41. The band is selected by the band selection flip-flops. See figures 4-1 and A-18. The operation of these flip-flops is explained in paragraphs 3-224 through 3-227. The input gates of the band selection p3 and p4 flip-flops are alerted at St3B, the same time as the quadrant selection flip-flops are alerted. At this time, the three highest-order bits of the p3 digit from rC are on the S lines. The S inputs to the p3 flip-flops generate MS signals which indicate one of five possible band addresses specified by the hundreds digit of the address. Digit p3 specifies one of the following band addresses: 000 through 199, 200 through 399, 400 through 599, 600 through 799, 800 through 999. The quadrant selection circuits determine, at the same time, a specific quadrant of such a band address.

4-42. The band selection p4 flip-flops sample the two lowest-order bits of digit p4 to determine the thousands digit of the address. At time St3B, direct outputs of rC are sampled by the MS10 and MS20 flip-flops. The C signals are sent directly to the flip-flops without being delayed at the S buffers so that all of the band and quadrant selection operations are initiated at time St3B.

4-43. The fast/normal flip-flop determines whether the desired address is in fast- or normal-access storage. At St3B, the third lowest-order bit of digit p4 is sampled by the input gates of the flip-flop. The C33 and C34 signals from rC indicate whether the desired address is in normal-access locations 0000 to 3999 or fast-access locations 4000 to 4999. For normal-access selection the flip-flop generates the NM signal; for fast-access selection the flip-flop generates the FM signal. These signals go to the switch selection matrix and to the read-write circuits to control reading and writing by fast- or normal-access heads.

4-44. The outputs from all the band selection and quadrant selection circuits appear at the switch selection matrix at t4B. The clear-band selection circuit, which is explained in paragraphs 3-228 and 3-229 clears all of the band selection flip-flops and also the quadrant selection flip-flops if the p1 and p2 digits are found to be unequal.

4-45. If the search operation has located the correct address, as indicated by a low TS signal, FS1 makes gate 42A (figure A-19) of the read flip-flop permissive at t8B. The permissive gate sets the flip-flop, the output of which goes to gates 45A and 45B. One of these gates is made permissive at t9B by the FM or NM signal from the fast/normal selection flip-flop (figure A-18). If gate 45B is made permissive, the low RGF signal alerts the fast-access read output gates (figure A-20) to enable the instruction word in the specified fast-access storage location to be read from the drum onto the DM lines. If gate 45A is made permissive, a low signal (figure A-19) alerts the normal-access read-output gates to enable the instruction word to be read from normal-access storage onto the DM' lines. The instruction word thus is read from the memory to be stored in the static register and rC, during the staticize step. (See section 6-239, for a discussion of read circuits in detail.)

4-46. Paragraphs 3-231 through 3-234 explain switch selection and the operation of the memory switch circuits.

4-47. STATICIZE-INSTRUCTION STEP

4-48. The primary function of the staticize step (figure 4-5) is to store the two instruction digits of the instruction word so that they can control the execution of the instruction. Other functions of the staticize step (SZ) are to store the entire ten digits of the instruction word in rC and to store the p7 digit of the instruction word in the multiplier/quotient counter. The word stored in rC, in addition to containing the two instruction digits, contains the storage address of the operand to be used in the instruction (m address) and the address of the next instruction word (c address). The p7 digit of the instruction word specifies the number of shifts necessary in a shift instruction and the number of the output stacker to be selected in the card reader in a select stacker instruction. It is stored in the MQC on every instruction so that no special preparation is necessary if a shift instruction is called for by the program.

4-49. At t10B of the search step, the output of the TS FF is sampled to make certain that the search step was successful. If the search was successful, the TS signal is low at gate 7 of STR FF1 (figure A-2). Function signal 1 alerts this gate which also samples the $\overline{OF}$ output of the overflow FF at t10B of the search step. Because the conditions necessary to generate a high $\overline{OF}$ signal are explained in paragraph 3-143, it is sufficient to say here that the low $\overline{OF}$ signal indicates that no abnormal conditions are present. When gate 7 is permissive to these signals, it sets FF 1 to a low STR1 output, which indicates a 1 bit, and generates a high RCT1 signal which restores the CT FF (figure A-12).

4-50. Every staticize step causes the CT FF to be restored to $\overline{CT}$. This condition causes the c address to be read unless at a later time the CT FF is set to CT. As a result, an instruction which requires an operand sets the CT FF at the end of the staticize step; otherwise the next search is for the c address.

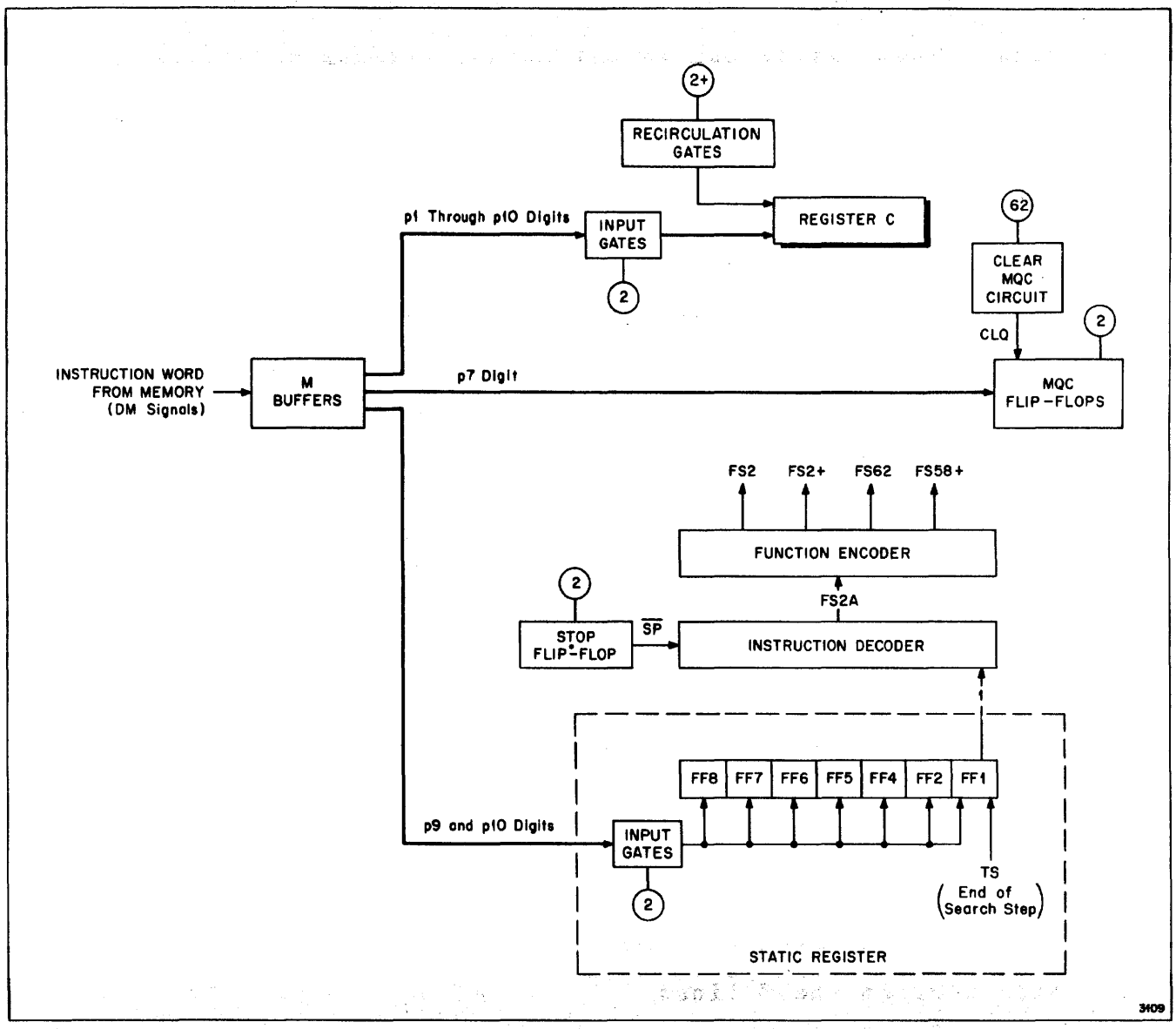


Figure 4-5. Staticize Step

4-51. When FF1 is set to a 1 output (figure 4-5), the input to the instruction decoder is 0000 0X01, since only that flip-flop has changed. The decoder converts this combination into FS 2A, which drives the function encoder to generate function signals 2, 2+, 62, and 58+. These signals control the functions of the staticize step.

4-52. STORING THE p7 DIGIT. Before storing the p7 digit in the four flip-flops of the MQC, the flip-flops must be cleared of previous information. At t0B of the staticize step, FS 62 at gate 19 of the clear-MQC circuit (figures 4-5 and A-14) generates the CLQ signal. The CLQ signal restores the four flip-flops to 0 at gates 5, 6, 7, 8.

4-53. Function signal 2 alerts the input gates of the flip-flops at t7B, the time interval during which digit p7 is on the M lines. The four bits of digit p7 enter the flip-flops to be stored for future use if the staticized instruction is a shift or select stacker instruction.

4-54. STORING THE INSTRUCTION WORD. At t0A of the staticize step, the p0 digit of the instruction word from memory is present on the DM or DM' lines of the read-output circuits (figures A-19 and A-20). These signals go directly to the M buffer and are on the M lines at t0B. Function signal 2+ blocks the recirculation gates of the four subregisters of rC at t11B+ of the search step, clearing the register. Thus, at t0B of the staticize step, the p0 digit of the instruction word enters the permissive input gates of rC. Function signal 2 alerts the input gates to the digits of the instruction word on the M lines.

4-55. STATICIZING THE INSTRUCTION. Once they have performed their functions, the function signals generated by the staticize step must be removed. At t8B of the staticize step, FS 2 is low at gate 114 of the static register (figure A-2). This restores STR FF1 to a 0 output, which causes the function signals of the staticize step to be lost at t11B.

4-56. At t9B the p9 digit of the instruction word is present on the M lines as shown in figure 4-5. Function signal 2 alerts the input gates of FF1, FF2, and FF4 at t9B so that the p9 digit can enter. (Only three bits of the p9 digit are of use in the processor.) To make certain

that no function signals are generated in the one pulse time between the time digit p9 is staticized (t9B) and digit p10 is staticized (t10B), the high $\overline{SP}$ signal blocks the instruction decoder for one pulse time. At t9B, function signal 2 sets the stop flip-flop; gate 20 of the stop flip-flop generates a high $\overline{SP}$ signal (figure A-5). The high $\overline{SP}$ signal blocks the instruction decoder.

4-57. At t10B, the p10 digit is present on the M lines. Function signal 2 also alerts the input gates of flip-flop 5, flip-flop 6, flip-flop 7 and flip-flop 8 at t10B so that the p10 digit can enter. By t11B the two instruction digits are completely staticized in the static register. Function signals required to execute the staticized instruction are available from the instruction decoder at t0A, and from the function encoder at t0B.

4-58. SEARCH-FOR-OPERAND STEP

4-59. In the search for an operand step, the memory address designated by the m address of the instruction word is searched for in the same manner as in the search-for-instruction step. When located, the operand is read from memory in time to be used in the execution step of the instruction. In some instructions, the m address does not contain an operand. It may instead contain one of the following: a location in memory to which information must be transferred, the number of shifts in a shift instruction, a stacker number in a select-stacker instruction, or the number of lines to be advanced in the print instruction.

4-60. The search-for-operand step is not required for all instructions. It is necessary only in instructions which require one of the following: an operand from memory, a memory location to be read from or written onto, shift instructions, or select-stacker instructions. The arithmetic instructions require an operand as do the instructions which transfer information from or to memory.

4-61. INSTRUCTION CODE CHARACTERISTICS. Instructions requiring operands can be identified by their instruction code. When the LSD (digit p9) of a staticized instruction is a decimal 0 or 5, a search for operand step is necessary. The two lowest-order bits of decimal 0 and decimal 5 are 00. The 00 bits indicate to the control unit that the memory location designated by the m address of the instruction word in rC either contains an operand or is

the location into which information is to be transferred. The least significant digits of instructions which do not involve an operand or storage location are other than 0 or 5; therefore the two lowest-order bits of such instructions are not 00, but 01 or 10. Table 4-3 shows the characteristic codes of instructions requiring a search-for-operand step.

4-62. LOCATING THE OPERAND. The search process required to locate an operand in memory is identical to the search-for-instruction process. Control of the operand search, however, differs from control of the instruction search. Figure 4-1 is also a block diagram of the search-for-operand step. Differences between the instruction search and operand search in the figure are indicated by the notes. In locating an instruction, the c address in rC is compared with the timing band addresses; in locating an operand, the m address in rC is compared with the timing band addresses.

4-63. Although FS 63 alerts both the m-address output gates and the c-address output gates of rC, the CT FF determines which address is to be read out and compared. When an instruction is staticized, the output of STR FF2 determines whether an operand is necessary. Table 4-3 shows that the STR2 position of the staticized instruction is always 0 when a search for operand is required.

4-64. The $\overline{\text{STR2}}$ output of the static register is sampled by gate 12 of the CT FF (figure A-12) at t10B of the staticize step. If the $\overline{\text{STR2}}$ signal is low, indicating that a search must be made for the m address, the gate is permissive. When gate 12 is permissive the CT FF is set to CT, which controls readout of the m address from rC. When gate 12 is blocked by high signal $\overline{\text{STR2}}$, the CT FF is restored to $\overline{\text{CT}}$, which controls readout of the c address. The staticized instruction, however, has generated new function signals for executing the instruction so that the $\overline{\text{CT}}$ signal has no effect on rC until after the execution step.

4-65. The CT output of the CT FF is available at the m-address output gates of rC at t0B. The LSD of the m address, p5, goes from rC to the S buffers and the S lines and is available at the quinary and binary equality gates of the comparator (figure A-12) at t1B. The comparison and memory selection processes for the search-for-operand step are the same as those of the search-for-instruction step, (paragraph 4-5).

4-66. The H, X, and J transfer instructions require a search-for-operand step, but the operand is a memory location into which information must be written. Because these three instructions entail writing in the memory, time must be allowed for the write circuits to be prepared. For this reason, the H, X, and J instructions generate FS 3 for use in the search operation in addition to the normal search (Sc) function signals. Function signal 3 alerts gate 8 of STR FF1 to sample the TS signal at t5B. (In normal search, the TS signal is sampled three pulse times later, at t8B.) Sampling at t5B allows the write circuits to prepare for writing by the time the execute step begins. If the TS signal is low at t5B, the STR FF1 is set to a 1, initiating the execute step of the instruction stored in the static register.

4-67. EXECUTE-INSTRUCTION STEP

4-68. The execute-instruction step is initiated when the seven bits of the two instruction digits are staticized in the static register and either of the two lowest-order bits of the p9 digit is a 1 (search-for-operand step is unnecessary), or when the search for-operand step has been completed and the STR FF1 has been stepped to a 1 output. In both cases the combination of bits stored in the STR represents a specific instruction and can be decoded by the instruction decoder. The combination in the static register goes to the instruction decoder and function encoder to generate one or more function signals which execute the staticized instruction.

4-69. The Analysis of UCT Instructions manual lists the function signals generated for each instruction and the purpose of each of these signals. Sections 4-74 through 4-199, and the input-output manuals explain the details of the execution steps of typical instructions. The instructions not covered in detail in these sections and the input-output manuals are treated in functional block diagram form in Section V.

4-70. TWO-EXECUTION-STEP INSTRUCTIONS. Many instructions involve two execution steps. The add instruction (70), for example, adds two quantities in the first execution step and complements if necessary the sum of the addition in the second execution step. The function signals generated for the first step (A1) of the add instruction include FS 64. At t10B of the A1 step FS 64 at the stepping gate of the STR (figure A-2) restores FF1 and FF2.

Flip-flop 1 had been previously set to a 1 to accomplish the A1 step. Setting FF2 to a 1 output sets up the combination in the static register for step A2; as a result, a new set of function signals is generated to control the A2 step. The stepping gate similarly controls FF1 and FF2 during many other instructions.

4-71. Table 4-3 shows the code combinations for the steps of all UCT instructions. The table is divided into instructions which require only one execution step and those which require more than one. It is subdivided further into instructions which require a search-for-operand step and those which do not. The third column, STR code, lists the code combinations for each step of each instruction after it has been staticized in the static register. The code positions marked by X are bits which are unnecessary to the staticize step. The X in the LSD - STR3 position is included only to balance each combination with the eight bits of the biquinary code.

4-72. TIMING OF THE BASIC OPERATION CYCLE

4-73. The four steps of the basic operation cycle require a minimum of four word times to complete. Instructions requiring no search-for-operand step, take only three word times. Instructions with more than one execution step require more than four word times. The timing of the basic operation cycle with and without a search for operand step is shown in figure 4-6. The timing of the various operations of the search-for-operand step is identical to that of the search-for-instruction step. Therefore the search-for-operand step shows only the timing of operations unique to it. Although many other operations occur during each word time, only the ones necessary to explain the timing of the whole cycle are included.

4-74. PROCESSING A TYPICAL INSTRUCTION

4-75. This section applies the previously discussed details of the basic operation cycle to the processing of an actual instruction. In the search-for-instruction step, the operation of the memory read circuits was explained. Use of the H(60) instruction in this section permits explanation of the memory write circuits because the instruction requires a transfer of information in a register to a storage location.

4-76. In the search-for-instruction step, the H(60) instruction was located in the memory. In the staticize step the instruction word was stored in rC, and

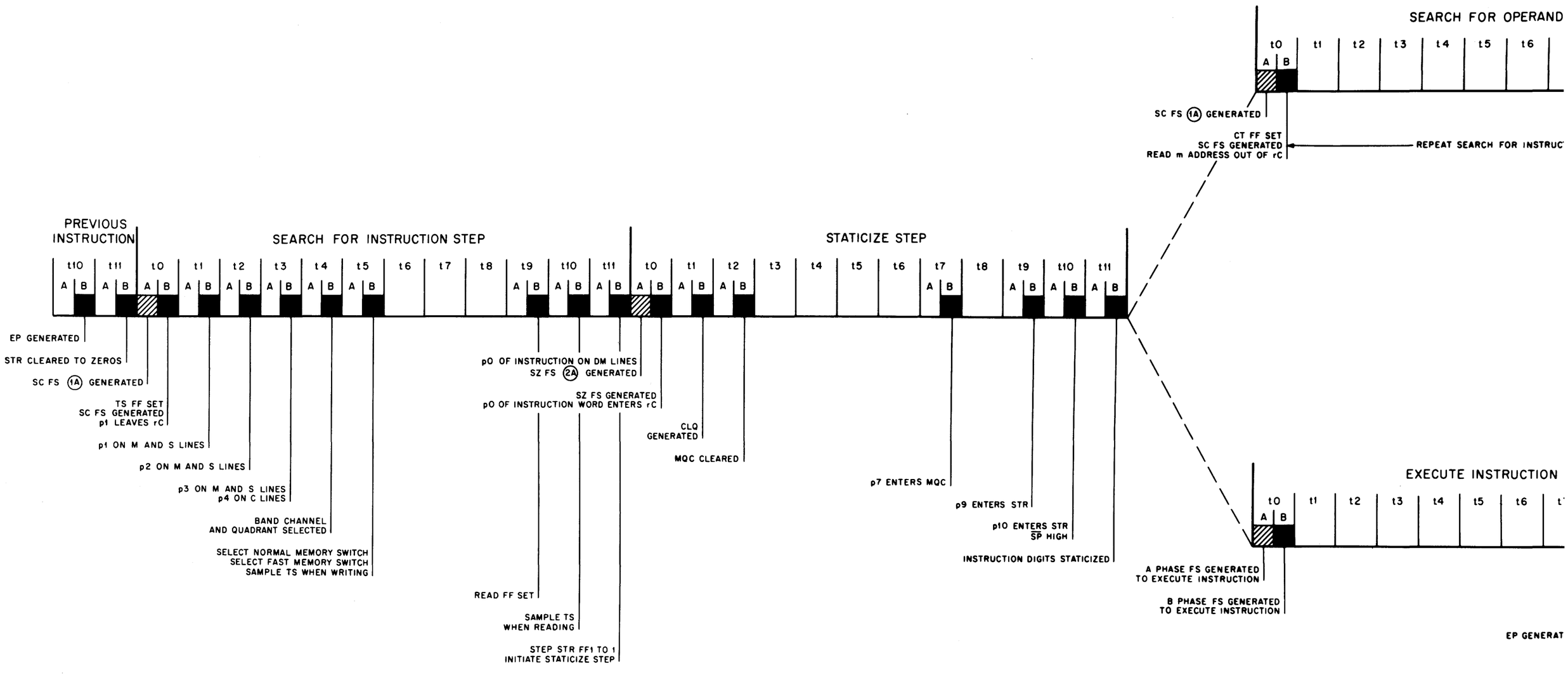


Figure 4-6. Timing of Basic Operation Cycle

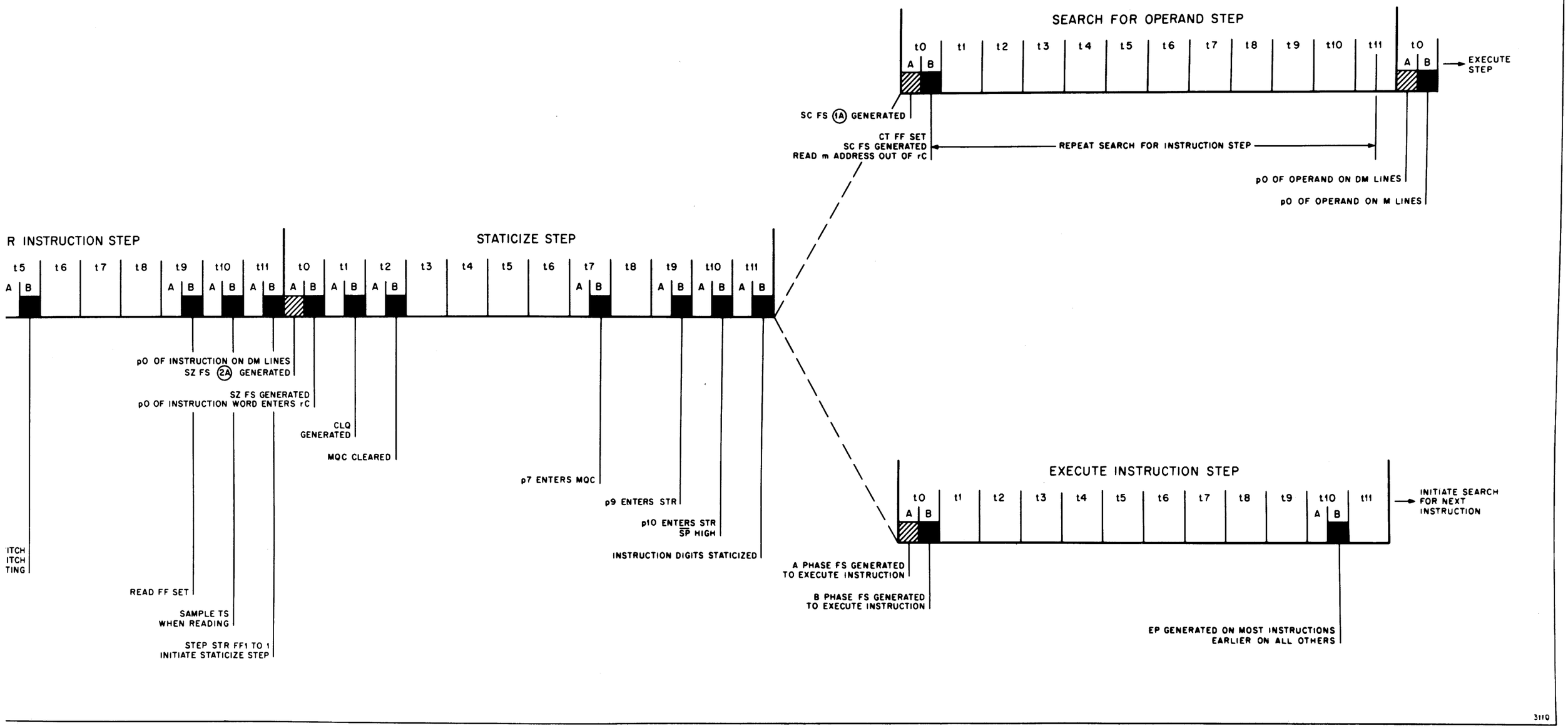


Figure 4-6. Timing of Basic Operation Cycle

Table 4-3. Instruction Code Combinations

INSTRUCTIONS WITH ONE EXECUTE STEP (SEARCH)

Instruction	Biquinary	STR Code S S S S S S S T T T T T T T R R R R R R R 8 7 6 5 4 2 1	Steps
05 (Y): (m) ——> rX	0 0 0 0 1 0 0 0	X X X X X X 0 0	(Sc 00)
		0 0 0 0 1 X 0 1	(Y 06)
20 (P): Superimpose (m) and	0 0 1 0 0 0 0 0	X X X X X X 0 0	(Sc 00)
(rA) ——> rA		0 X 1 0 0 X 0 1	(P 21)
25 (B): (m) ——>rA	0 0 1 0 1 0 0 0	X X X X X X 0 0	(Sc 00)
		0 X 1 0 1 X 0 1	(B 26)
30 (L): (m) ——> rL	0 0 1 1 0 0 0 0	X X X X X X 0 0	(Sc 00)
		0 X 1 1 0 X 0 1	(L 31)
35 (E): Logically multiply	0 0 1 1 1 0 0 0	X X X X X X 0 0	(Sc 00)
(m) and (rA) ——> rA		0 X 1 1 1 X 0 1	(E 36)
50 (J): (rL) ——> m	1 0 0 0 0 0 0 0	1 0 0 0 0 X 0 0	(J-Sc 50)
		1 0 0 0 0 X 0 1	(J-Ex 51)
60 (H): (rA) ——> m	1 0 0 1 0 0 0 0	1 X 0 1 X X 0 0	(H-Sc 60)
		1 X 0 1 0 X 0 1	(H-Ex 61)
65 (X): (rX) ——> m	1 0 0 1 1 0 0 0	1 X 0 1 X X 0 0	(X-Sc 65)
		1 X 0 1 1 X 0 1	(X-Ex 66)

INSTRUCTIONS WITH ONE EXECUTE STEP (NO SEARCH)

Instruction	Biquinary	STR Code	Steps
		S S S S S S S T T T T T T T R R R R R R R 8 7 6 5 4 2 1	
12 (G): RR ——→UCT (rA) + (rX) ——→rA	0 0 0 1 0 0 1 0	0 X 0 1 0 X 1 0	(G 12)
17 (R): UCT ——→RR (rA) ——→rA + rX	0 0 0 1 1 0 1 0	0 X 0 1 1 X 1 0	(R 17)
47 (Z2): Select output stacker of fast reader	0 1 0 0 1 0 1 0	0 1 0 0 1 X 1 0	(Z2 47)
67 Stop	1 0 0 1 1 0 1 0	1 X 0 1 1 X 1 0	(Stop 67)
72 (CC): Feed one card fast reader	1 0 1 0 0 0 1 0	1 X 1 0 0 X 1 0	(CC 72)
77 (K): (rA) ——→rL	1 0 1 0 1 0 1 0	1 X 1 0 1 X 1 0	(K 77)

82 (Q): (rA) : (rL)
If (rA) = (rL), go to m
If (rA) ≠ (rL), go to c
	1 0 1 1 0 0 1 0	1 X 1 0 0 X 1 0	(Q 82)

87 (T): (rA) : (rL)
If (rA) > (rL), go to m
If (rA) ≤ (rL), go to c
	1 0 1 1 1 0 1 0	1 X 1 1 1 X 1 0	(T 87)

INSTRUCTIONS WITH TWO OR MORE EXECUTE STEPS (SEARCH)

Instruction	Biquinary	STR Code S S S S S S S S T T T T T T T T R R R R R R R R 8 7 6 5 4 3 2 1	Steps
55 (D): (m) ÷ (rL) ⟶ rA	1 0 0 0 1 0 0 0	X X X X X X 0 0	(Sc 00)
remainder ⟶ rX		1 0 0 0 1 X 0 1	(D1 56)
		1 0 0 0 1 X 1 1	(D2 58)
		1 0 0 1 1 X 1 1	(D3 68)
70 (A): (m) + (rA) ⟶ rA	1 0 1 0 0 0 0 0	X X X X X X 0 0	(Sc 00)
		1 X 1 0 X X 0 1	(A1 71)
		1 X 1 0 X X 1 1	(A2 73)
75 (S): (rA) - (m) ⟶ rA	1 0 1 0 1 0 0 0	X X X X X X 0 0	(Sc 00)
		1 X 1 0 X X 0 1	(S1 76)
		1 X 1 0 X X 1 1	(S2 78)
85 (M): (m) x (rL) ⟶ rA+rX	1 0 1 1 1 0 0 0	X X X X X X 0 0	(Sc 00)
		1 X 1 1 1 X 0 1	(M1 86)
		1 X 1 1 1 X 1 1	(M2 88)

INSTRUCTIONS WITH TWO OR MORE EXECUTE STEPS (NO SEARCH)

Instruction	Biquinary	STR Code S S S S S S S T T T T T T T R R R R R R R 8 7 6 5 4 2 1	Steps
11 (PR1): memory ——>print buffer	0 0 0 1 0 0 0 1	0 X 0 1 X X 0 1	(PR1 11)
		0 X 0 1 X X 1 1	(PR2 13)
16 (PF1): paper advance	0 0 0 1 1 0 0 1	0 X 0 1 X X 0 1	(PF1 16)
		0 X 0 1 X X 1 1	(PF2 18)
22 (I11): Test read punch unit card buffer	0 0 1 0 0 0 1 0	0 X 1 0 0 X 1 0	(I11 22)
If loaded, go to m		0 X 1 0 X X 1 1	(I12 23)
If not loaded, go to c			
27 (I21): If printing or paper feeding, go to c	0 0 1 0 1 0 1 0	0 X 1 0 1 X 1 0	(I21 27)
If not printing or paper feeding, go to m; (rC) ——> rA		0 X 1 0 X X 1 1	(I12 28)
32 (N): Right circular shift |rA| ——>rX; |rX| ——>rA n places	0 0 1 1 0 0 1 0	0 X 1 1 X X 1 0	(N1 32)
		0 X 1 1 0 X 1 1	(N2 33)
37 (V): Left shift rA n places Lose MSD Replace LSD with 0	0 0 1 1 1 0 1 0	0 X 1 1 X X 1 0	(V1 37)
		0 X 1 1 1 X 1 1	(V2 38)
42 (I31): Test card buffer fast reader	0 1 0 0 0 0 1 0	0 1 0 0 0 X 1 0	(I31 42)
If loaded, go to m		0 X 1 0 X X 1 1	(I12 23)
If not loaded, go to c			

INSTRUCTIONS WITH TWO OR MORE EXECUTE STEPS (NO SEARCH) (cont)

Instruction	Biquinary	STR Code	Steps
		S S S S S S S T T T T T T T R R R R R R R 8 7 6 5 4 2 1	
46 (W11): Read punch unit card buffer ⟶	0 1 0 0 1 0 0 1	0 1 0 0 1 X 0 1	(W11 46)
main memory		0 1 0 0 1 X 1 1	(W12 48)
62 (ZS1): Zero suppress Rem Rand word in rA	1 0 0 1 0 0 1 0	1 X 0 1 0 X 1 0	(ZS1 62)
		1 X 0 1 0 X 1 1	(ZS2 63)
81 (W21): Start read punch unit	1 0 1 1 0 0 0 1	1 X 1 1 0 X 0 1	(W21 81)
and main memory ⟶		1 X 1 1 0 X 1 1	(W22 83)
Read punch unit card buffer			
96 (W31): Fast reader card	1 1 0 0 1 0 0 1	1 1 0 0 1 X 0 1	(W31 96)
buffer main memory		1 1 0 0 1 X 1 1	(W32 98)

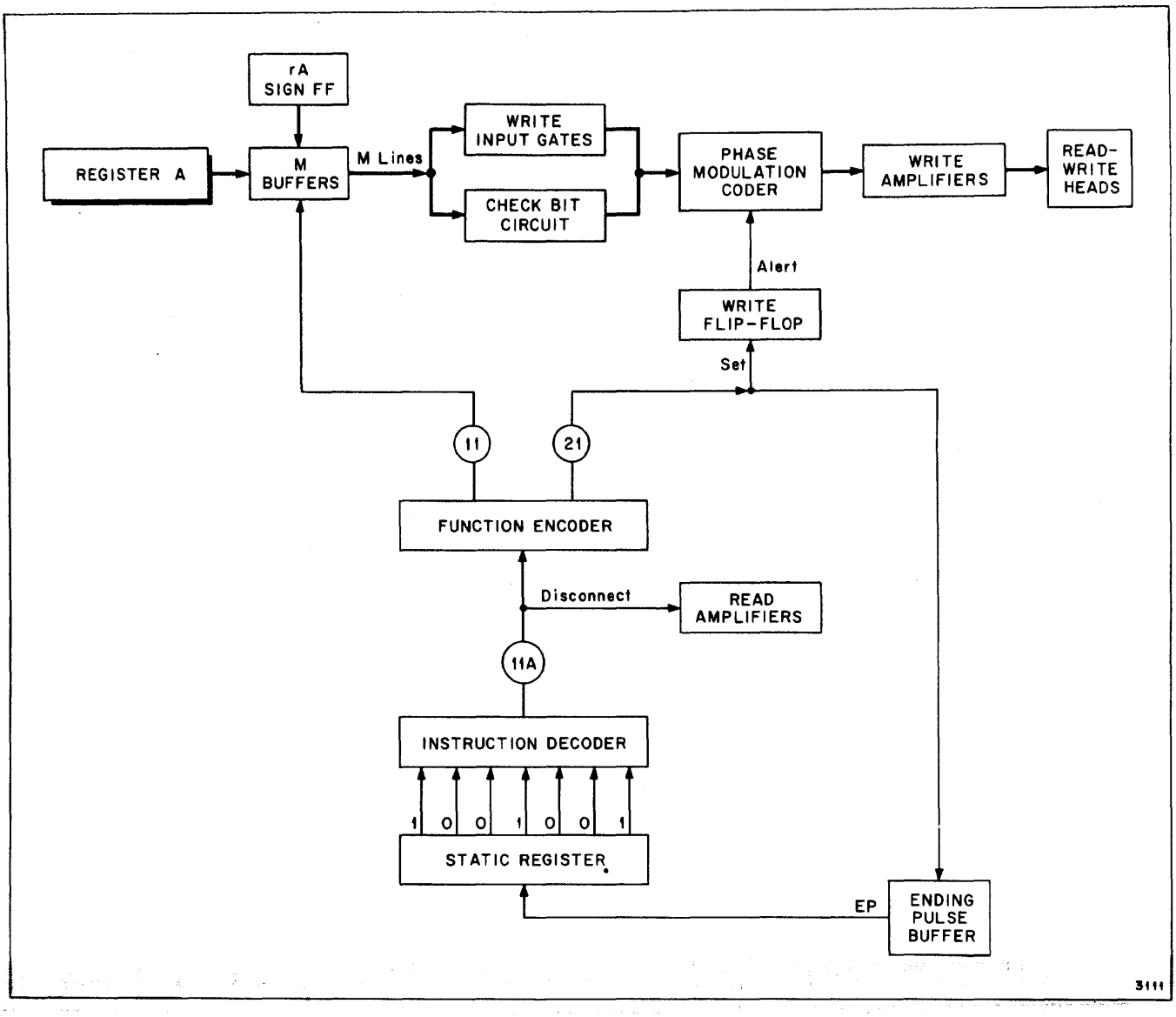


Figure 4-7. Execute Step of H (60) Instruction

the two instruction digits, 60, were staticized in the static register. In the H instruction the contents of rA must be transferred to a memory location. The address of that location is the operand in the instruction. The execute step of the H instruction controls the writing of the word from rA into the specified memory location on the drum.

4-77. SEARCH FOR OPERAND

4-78. The H(60) instruction is staticized in the static register flip-flops by the staticize step as 1001 0X00. The two lowest-order bits of the p9 digit are 0's, indicating a search. This combination is picked up by the H-X-J-Sc and Sc gate lines of the instruction decoder and generates FS 1A and FS 3A. (The search steps of the H, X, and J instructions require the same function signals; therefore, all are on a common instruction decoder gate line.)

4-79. Function signal 1A generates FS 58+, FS 74, FS 63 and FS 1. Function signal 58+ blocks readout from rA onto the S lines. Function signal 74 sets the CP FF to produce a CP signal. Function signal 63, in conjunction with the CT signal, causes readout of the m address from rC onto the S lines. Function signal 1 alerts the input gates of the M buffer to enable the timing band signals from the drum to enter the M lines.

4-80. The 1001 0X00 STR output combination also generates FS 3A (H-X-J Sc gate) which generates FS 3 to alert the STR FF1 stepping gate. At t5B of the search-for-operand step, the output of the TS FF is sampled by gate 8 of FF1. A low TS signal and FS 3 make gate 8 permissive and step FF1 from a 0 output to a 1 output. The new STR output combination of 1001 0X01 (61) is present at the instruction decoder at t6B.

4-81. At t6B, FS 3 sets the write FF (figure A-19) at gate 5B to prepare the write circuits for the execute step of the instruction.

4-82. When gate 8 of STR FF1 is permissive at t6A, a high RCT2 signal is generated which goes to the input gates of the CT FF and to buffer 104 of the write circuits. The high RCT2 signal restores the CT FF, generating the low CT signal which controls readout of the c address from rC and generates the write pedestal (paragraph 6-243). The new STR combination causes the function signals for search to end at t7A. The search-for-operand step has located the operand, which is the storage location designated by the m address.

4-83. EXECUTE THE H(60) INSTRUCTION

4-84. In the execute step of the H(60) instruction (figure 4-7), the contents of rA is recorded in the storage address located by the search-for-operand step. During the execution step, certain function signals are generated by the new combination in the static register. These signals control the transfer of the contents of rA onto the M lines and into the memory write circuits, the preparation of the write circuits to record on the drum, and eventually the ending of the instruction.

4-85. At t7A of the search step the H gate line of the instruction decoder generates FS 11A which generates FS 11 and FS 21. These signals control the writing of the contents of rA onto the memory drum.

4-86. Function signal 11 alerts input gates 7A, 7B, 7C, and 7D of the M buffers (figure A-6). These gates are permissive to the four bits of the p0 digit (A1M, A2M, A3M, and A4M) from rA at t7B, and the contents of rA enters onto the M lines. Gate 6 of the M buffers, also alerted by FS 11, also is permissive at t7B to a low A- signal. This condition indicates that the sign of the contents of rA, which has been stored in the rA sign flip-flop, is minus. If the sign of rA is plus, low A- is absent from gate 06. The sign position of the word (t7B) on the A1M lines contains 0, indicating a plus sign. The word and its sign are sent from rA onto the M lines and into the write circuits (figures A-19 and A-20).

4-87. At t4B of the execute step FS 21 operates gate 19 of the static register to generate the EP and Reset-TS signals. Because the H, X, and J instructions involve writing on the drum and therefore require less time for execution than most instructions, the EP signal is generated early. The function signals for the instruction are lost at t7B. The TS FF is restored so that the set output of the flip-flop, TS, cannot set the read flip-flop. Normally, at t8B, the read flip-flop (figure A-19) is set by FS 1, TS and $\overline{OF}$. All these signals will be present at t8B because the function signals for search will be available at t7B. To keep the read flip-flop restored until reading is required, the Reset-TS signal restores the TS FF, keeping the TS signal high. High TS blocks gate 42A of the read flip-flop, keeping it restored. The zero combination in the STR initiates a search for the next instruction.

4-88. MEMORY WRITE OPERATION

4-89. During a write operation, information and check bits are sent on the M lines to the phase-modulation coder where each input signal is coded into a phase-modulated pulse representation for memory recording. The resulting pulse is amplified in the write amplifier and drives the write heads.

4-90. At t7A, FS 11A electronically disconnects the read amplifiers from the write circuits by generating the write pedestal (paragraph 6-243). Control signal RCT2, present for only one pulse time, has generated the write pedestal from t6A until t7A. For writing on the fast-access bands the IR output of the write-pedestal generator disconnects the read amplifiers of the fast-access heads.

4-91. The bits of each digit to be written are sent in parallel to the check-bit circuit and to the write input circuits. The check-bit circuit is explained in paragraph 4-202. Each of the four bits of the digit to be written is delayed a pulse time by two complementing amplifiers before going to the input gates of the phase-modulation coder, so that the check bit, and the information bits can be written simultaneously. Computing the check bit requires a full pulse time. At t9B the information bits and a check bit are present at the input gates to the phase-modulation coder. The write flip-flop output alerts the input gates at t8B and for a full word time thereafter. The operation of the phase-modulation coder is explained in paragraphs 3-208 through 3-212. The bits are converted into a phase-modulated waveshape which goes directly to the read-write heads and is written on the drum in the location specified by the m address.

4-92. ARITHMETIC OPERATIONS

4-93. ADDITION AND SUBTRACTION

4-94. ADD INSTRUCTION. The functions of the add instruction are: add algebraically the quantity in the storage location designated by m to the contents of rA, store the result in rA, compute the sign of the sum and store the sign in the rA sign flip-flop. The quantity in the memory is the addend; the quantity in rA is the augend.

4-95. Before addition takes place:

(1) The search for the instruction has been completed.

(2) The add instruction (70) has been staticized.

(3) The staticized instruction has initiated a search for the operand (m address), which in this instruction is the addend.

(4) The augend has been stored in rA by a previous instruction and its sign has been stored in the register A sign flip-flop. The add instruction requires two execution steps, A1 and A2.

4-96. A-1 STEP. In the first step of addition, A1, (figure 4-8a), the signs of the augend and addend are compared at the input gates of the CP FF. The gates determine from the two signs whether the augend is to be complemented (CP) or not complemented ($\overline{CP}$) before addition. This information controls the complementing circuit of the quinary adder. The sign of the augend is compared with the CP output of the complement flip-flop at the input gates of the rA sign flip-flop to determine the sign of the sum. Once determined, the sign of the sum is stored in the rA sign flip-flop. The augend in rA is sent on the S lines into the binary and quinary adders. The addend from the memory location designated by the m address is sent on the M lines into the binary and quinary adders. The adders add the two quantities digit by digit and transfer the sum on the 0 lines into rA.

4-97. The staticized digits of the A1 step (71) generate function signals 4, 50, 55+, 64, and 75 at t0B to execute the A1 step. Function signal 4 alerts the input gates of the complement (CP) FF. Gates 8 and 9 of the CP FF (paragraph 3-138) operate during the add instruction to sample the signs of the two quantities. If the signs are unlike, one of the gates operates to set the CP FF to CP, which will cause the augend digits to be complemented. If the signs are alike, gates 8 and 9 are blocked, the flip-flop remains restored to $\overline{CP}$, and the augend is not complemented.

4-98. Function signal 55+ blocks the recirculation gates of rA, and the digits of the augend go to the S buffers and the S lines. Function signal 50 alerts the quinary and binary adders gates at t0B.

4-99. The quinary bits of the p1 digit of the addend (M1, M2, and M3) go to the input gates of the decimal carry adder (figure A-11) and the quinary carry gates (figure A-12). The input gates of the decimal carry adder sample the A and C signals from the binary and quinary circuits to determine whether a carry bit is to be added to the two digits being added. Because the p1 digits are the first to be added there can be no carry, and the $\overline{A}$ or $\overline{C}$ signal, or both, will be low. The outputs of the decimal carry adder go to the quinary adder. They represent the quinary bits of each digit of the addend.

4-100. The quinary bits of the p1 digit of the augend (S1, S2, and S3) go to the input gates of the complementing circuit (figure A-11). If the CP (complement)

signal alerts the gates, the circuit produces the complement of the input bits. If the $\overline{CP}$ (no complement) signal alerts the gates, the quinary S bits pass through the circuit unchanged.

4-101. The quinary S bits and M bits also are sent in parallel to the quinary carry circuit. Any combination of quinary bits which indicates the need for quinary carry generates C and C' signals from the quinary carry circuit. The gates are blocked by combinations which require no quinary carry, generating the low $\overline{C}$ signal. The low C signal, if generated, is used in conjunction with the A signal to indicate a decimal carry into the addition of the next two digits. The low C' signal indicates the presence of quinary carry and causes a carry bit to be added in the addition of the binary bits in the binary adder.

4-102. The quinary outputs of the decimal carry adder and those of the complementing circuit are added in the quinary adder gates. One or two of the adder gates is permissive to the input combinations, producing a high $\underline{0}$ output signal. A high $\underline{0}$ output signal indicates a 1 bit, a low $\underline{0}$ output signal indicates a 0 bit.

4-103. The binary bits of the two quantities are added in the binary adder (figure A-12) at the same time as the quinary bits. Initially, the M4 and S4 binary bits are compared in the binary equality gates. The gates, alerted by $\overline{CP}$, sample the two binary bits during a normal addition. If either bit is a 1, a signal indicating the 1 bit goes to the binary carry circuit to generate a binary carry signal A, which will be used with the C signal, if present, in the addition of the next two digits. The carry circuit also sends the A' signal to gate 6 of the binary adder. If the $\overline{C}$ signal is low, gate 6 is permissive to the A' signal and the $\underline{0}42$ output is high, indicating a 1 bit in the MSB of the sum digit. No decimal carry will be generated if the $\overline{C}$ signal is low.

4-104. If the M4 and the S4 bits each has a value of 1, gate 30 of the force decimal carry circuit is permissive and the A and C signals are generated to force an indication of a decimal carry to the next addition.

105. If the M4 and the S4 bits each has a value of 0, the binary equality gates are blocked, generating low signals $\underline{A}$ and $\underline{A}'$ which indicate no binary carry. The low $\underline{A}'$ signal goes to gate 7 of the binary adder which is permissive only if quinary carry is present, a condition indicated by the low C' signal. If the gate is permissive, the $\underline{041}$ output is high, indicating a 1 in the MSB position of the sum digit. If quinary carry is absent, the C' signal is high blocking gate 7 and producing a low $\underline{041}$ bit. A low $\underline{041}$ output signal indicates a 0 in the MSB of the sum digit. The $\underline{0}$ signal outputs of the quinary adder and the binary adder are sent to the sum input buffers of rA. The sum digits are stored in rA during the A1 step.

4-106. If the sum of two quantities with unlike signs is zero, the sign of that sum must be forced to plus. The TS FF is used as a control circuit to indicate this condition to the rA sign flip-flop. Function signal 75 alerts restore gates 16 and 17 of the TS FF during the A1 step. If the signs of the two quantities being added are unlike, the CP FF is set to CP. The CP, A, and C signals are present during addition when the signs are unequal. These signals alert the quinary-equality gates, which compare for equality the digits being added in the adders. The CP signal alerts the binary-equality gates which also compare for equality. If the first nine digits of the quantities being added are equal, the TS FF remains set to TS. The TS signal goes to the rA sign flip-flop to force the sign of the sum to plus if the sum of the quantities is zero.

4-107. At t11B, FS 4 alerts the input gates of the rA sign flip-flop and the overflow flip-flop. Gates 3, 6, and 7 of the rA sign FF compute the sign of the sum (paragraph 3-126) and store it in the flip-flop. During addition, the overflow flip-flop (OF FF) is set at gate 29 when the input signals indicate that the sum has exceeded the capacity of rA. When this possibility is expected, the programmer provides for it by programming a routine in the c+1 address, to be used only if overflow occurs. The set output of the OF FF, OF, sets the overflow delay flop to OF2+ during the next search-for-instruction step. The OF2+ signal keeps the TS FF set at buffer 82 to delay for one word time the reading of the selected instruction word address. This causes the c+1 address to be read instead of the c address.

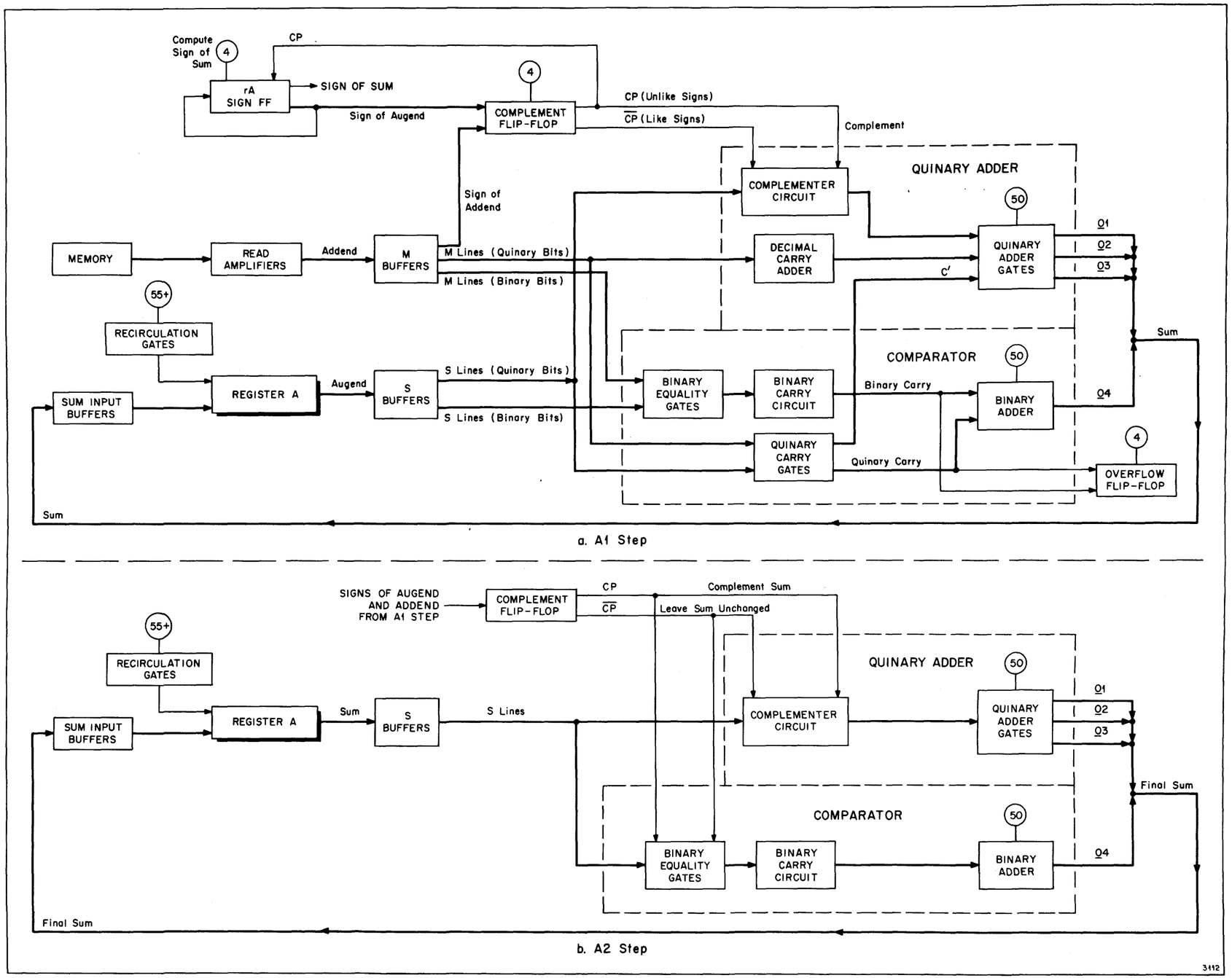


Figure 4-8. Add Instruction

4-108. Function signal 64 alerts the STR stepping gate at t10B of the A1 step. If the EP signal is low, the gate steps FF1 and FF2 to 1 outputs. At this point, however, FF1 is already generating a 1 bit output, so that, in effect, only FF2 is stepped. The new 11 reading in the lowest-order STR bits is converted into function signals for the A2 step by the instruction decoder and function encoder. The function signals generated for the A2 step are 50, 55+ and 67.

4-109. A2 STEP. If the sum determined during A1 is the complement of the true sum, it must be complemented to its true form in the A2 step (figure 4-8). The sum determined during A1 and stored in rA reads out onto the S lines. Register A is cleared by FS 55+ to make room for the true sum which will be determined in the A2 step.

4-110. Function signal 50 alerts the gates of the binary and quinary adders. The sum on the S lines goes to the complementing circuit and the binary equality gates for complementing if necessary. If complementing is unnecessary (paragraph 3-138), the $\overline{CP}$ signal is present and the sum merely passes through the complementing circuit. It also passes through the quinary and binary adders, as in the A1 step, and returns to rA. No addition takes place in the adders because there is no quantity on the M lines to be added. (In subtraction, however, a 1 is automatically added to the quantity from rA in the decimal carry adder.)

4-111. Function signal 67 operates gate 17 of the ending-pulse buffer circuit at t9B. At t10B the EP signal is generated to clear the static register of the add instruction and initiate a search for the next instruction.

4-112. SUBTRACTION. The execution of add and subtract instructions is identical; the two instructions differ only in control functions. The function of the subtract instruction is to subtract algebraically the contents of the storage location designated by the m address from the contents of rA, store the difference in rA, compute the sign of the difference and store it in the rA sign flip-flop. The quantity in the memory is the subtrahend; the quantity in rA is the minuend. The subtract instruction also requires two execution steps, S1 and S2.

4-113. Although the instruction code for subtract is 75, the staticized digits cause the same function signals to be generated as those generated for the staticized add instruction. The STR FF4 output for subtract is an STR4 signal. As a result, gates 10 and 11 of the CP FF operate during subtraction to determine whether the subtrahend is to be complemented.

4-114. When quantities with unlike signs are subtracted, the process is the same as when quantities with like signs are added. The sign of the subtrahend is changed, and the two quantities are added. These conditions cause the CP FF to be restored to $\overline{CP}$ so that no complementing takes place.

4-115. A special circuit in the function encoder operates during addition and subtraction to control the CP FF and complementing in the A2 or S2 step. The operation of this circuit is described in paragraph 4-123.

4-116. SAMPLE PROBLEMS

4-117. The many similarities and the few dissimilarities of addition and subtraction may be illustrated by two sample problems. In the sample two digits are first added and then subtracted.

4-118. ADDITION. In this sample addition problem column (A) indicates the digits, (B) indicates their biquinary equivalents, (C) indicates the bits of the digits as signals on the M and S lines, and (D) represents the outputs of the decimal carry adder and complementing circuits. The $\overline{M4}$ and $\overline{S4}$ bits are not included because they are added within only the binary adder.

	(A)	(B)	(C)					(D)	
ADD:	+2 ADDEND (m)	0010	$\overline{M4}$	$\overline{M3}$	M2	$\overline{M1}$	$\overline{M3U}$	M2U	$\overline{M1U}$
	=	=					=		
	+1 AUGEND (rA)	0001	$\overline{S4}$	$\overline{S3}$	$\overline{S2}$	S1	$\overline{S3C}$	$\overline{S2C}$	S1C
	+3								

	LOW	LOW	HIGH	HIGH		
SUM AFTER A1 and A2:	04	03	02	01	=	0011 or +3

4-119. Like signs in the add instruction restore the CP FF to $\overline{CP}$. The quinary bits of the augend, M3, M2, and M1, go to the decimal carry adder which does not change their value. The outputs of the circuit are M3U, M2U, and M1U which have the same quinary value (010) as the inputs. The quinary bits of the addend $\overline{S3}$, $\overline{S2}$, and S1 go to the complementer circuit which does not complement

because the $\overline{CP}$ signal is low. The outputs of the complementer circuit, $\overline{S3C}$, $\overline{S2C}$, and S1C, also have the same value (001) as the inputs. The binary bits of the two digits, $\overline{M4}$ and $\overline{S4}$, go to the binary carry circuit. These two bits and $\overline{CP}$ generate a low $\underline{0}4$ output from the binary adder. A low output signal on the $\underline{0}$ lines is a 0 bit, which goes to the A4 subregister of rA. The outputs of the decimal carry adder and complementer circuits go to the quinary adder. Only gate 34 of the adder is permissive to the input combination. Gate 34 produces a high $\underline{0}12$ output which goes to the A1 and A2 subregisters of rA. Hence the two lowest order bits of the sum are 1's. Because only gate 34 is permissive, the other gates are blocked. As a result, inputs to the A3 subregister from the quinary adder are low $\underline{0}$ signals, indicating a 0 bit. The sum determined during the A1 step, therefore, is 0011, or 3. In the A2 step this sum recirculates unchanged through the complementer circuit because the CP FF remains restored to $\overline{CP}$. The rA sign flip-flop determines from the signs of the two quantities that the sign of the sum is plus.

4-120. SUBTRACTION. The same quantities which were added in the sample problem shown in paragraph 4-118 are subtracted in the sample problem at the end of this paragraph. In the first step of subtraction, S1, the input M and S bits indicated by (C) which go to the decimal carry adder and the complementer are the same as in addition. In subtraction, however, the decimal carry adder adds a 1 to the subtrahend and the complementer produces the 9's complement of the minuend if the CP signal is present. (Paragraph 4-122 explains the complementing.)

S1:	(A)		(B)		(C)				(D)
SUBTRACT:	+2 MINUEND (rA)	=	0010	=	$\overline{S4}$ $\overline{S3}$ S2 $\overline{S1}$	--> complement -->	$\overline{S3C}$	S2C	$\overline{S1C}$
	+1 SUBTRAHEND (m)		0001		$\overline{M4}$ $\overline{M3}$ $\overline{M2}$ M1	--> add 1 ------->	$\overline{M3U}$	M2U	$\overline{M1U}$

	HIGH	HIGH	LOW	LOW		
SUM AFTER S1:	$\underline{0}4$	$\underline{0}3$	$\underline{0}2$	$\underline{0}1$	=	1100 or 9

S2:	(E)		(F)		(G)
1100 (from rA)	=	S4 S3 $\overline{S2}$ $\overline{S1}$	---> complement --->	$\overline{S3C}$	$\overline{S2C}$ $\overline{S1C}$
0000 (on M lines)	=	$\overline{M4}$ $\overline{M3}$ $\overline{M2}$ $\overline{M1}$	---> add 1 -------->	$\overline{M3U}$	$\overline{M2U}$ M1U

	LOW	LOW	LOW	HIGH		
SUM AFTER S2:	$\underline{0}4$	$\underline{0}3$	$\underline{0}2$	$\underline{0}1$	=	0001 or 1

4-121. At t0B of the first step of subtraction (S1) the plus signs of the two digits and the STR4 signal set the CP FF to CP at gate 11. At t1A, high signal CP4 is generated for one pulse time. Signal CP4 goes to the initial force-decimal-carry circuit to force generation of the decimal-carry signals, A and C. The A and C signals represent the 1 bit which is to be added to the subtrahend in the decimal carry adder. Signals A and C, because they were generated by the CP4 signal, last for only one pulse time. The quinary M inputs to the decimal carry adder represent a decimal 1; the outputs (D) represent a 2. The reason for the addition of a 1 to the subtrahend is explained in paragraph 3-119.

4-122. Complementing of the S input combination to the complementing circuit is not evident from the output combination (D) which retains the same quinary value. The result of complementing, however, is evident at the outputs of the adders. Only gate 46 of the adder is permissive to the outputs of the decimal-carry adder and complementer circuits; it produces high $\underline{O}3$ and low $\underline{O}2$, and $\underline{O}1$ outputs. The M4 and S4 bits, with CP, send a low signal to gate 6 of the binary adder. None of the quinary carry gates alerted by CP is permissive to the M and S inputs. As a result, a low C signal alerts gate 6 to produce a high $\underline{O}4$ output, indicating a binary 1 bit. The $\underline{O}$ input bits to rA at the end of the S1 step therefore have a value of 1100 or 9. The decimal 9 is the 10's complement of the true answer.

4-123. Because addition of the two combinations in the S1 step produced no decimal carry, the $\overline{A}$ and $\overline{C}$ signals from the comparator (figure A-12) are low at t11B of the S1 step. Gates 62 and 63 of the function encoder (figure A-4), alerted by the CP signal, are permissive at t11B of the S1 step, to the low $\overline{A}$ and $\overline{C}$ signals. When the gates are permissive, low FS 74 is generated. This function signal keeps the CP FF set to CP and also generates the CP5 signal for one pulse time. The CP5 signal forces decimal carry signals A and C to low from t1B until t2B of the S2 step.

4-124. In the S2 step, the 1100(E) (paragraph 4-120) combination returns to the S lines from register A as S4, S3, $\overline{S2}$, $\overline{S1}$ (F). The three lowest order S bits go to the complementer which produces a $\overline{S3C}$, $\overline{S2C}$, $\overline{S1C}$ (G) output combination.

4-125. During the S2 step there is no information on the M lines, and the barred M signals are all low (F) (paragraph 4-120). The A and C signals, which are present for

one pulse time, however, cause a 1 to be added to the 0 value of the M bits. The resulting output of the decimal carry adder is M3U, M2U, M1U (G) (paragraph 4-120).

4-126. The outputs of the decimal carry adder and complementer go to the quinary adder. Only gate 32 is permissive to the input combination. Gate 32 produces a high $\underline{0}1$ output, while the $\underline{0}2$ and $\underline{0}3$ outputs of the quinary adder and the $\underline{0}4$ output of the binary adder are low. The output combination of the quinary and binary adders, 0001 or one, returns to the sum input gates of rA. The plus sign of the result is computed by the rA sign flip-flop (paragraph 3-128).

4-127. MULTIPLICATION

4-128. Multiplication in the UCT system is accomplished by iterative additions and shifts of the multiplicand. The multiplicand is added to itself the number of times specified by each digit of the multiplier. The multiplication process has two steps, M1 and M2. The processor can multiply a ten-digit multiplicand by a ten-digit multiplier. In such a case the 20-digit product will be stored in rA and rX. The least significant part of the product is stored in rX; the most significant part of the product is stored in rA.

4-129. Before multiplication, the multiplicand is placed in rL by a programmed transfer instruction. In the first step of multiplication (M1) the multiplier is transferred from its location in the memory to rX. A code combination, the multiplier sentinel, is placed in the LSD position of rA. The sentinel will signal the end of multiplication and cause the multiplication process to end.

4-130. In the second step, M2, the multiplicand in rL is added to itself in the adder the number of times indicated by the LSD of the multiplier. The LSD of the multiplier is stored in the multiplier quotient counter (MQC) which counts the number of additions. The sum of these additions is returned to rA. The contents of rA and rX are next shifted right one digit position, placing the LSD of rA into the most significant digit (MSD) position of rX. The LSD in rA before each shift becomes the LSD of the product. This process of adding and shifting continues until the least significant digit of the product is the LSD of rX.

4-131. The sentinel also is shifted during the shift operations. Initially it is shifted from the LSD position in rA to the MSD position of rX. Every ensuing shift operation shifts the sentinel one more place to the right until it reaches the LSD position of rX. It is then shifted into the MQC where it ends the multiplication process.

4-132. The M2 step consists of two phases, designated IER and $\overline{\text{IER}}$.

4-133. During each IER (shift) phase, the contents of rA and rX are shifted one place to the right, the sentinel is shifted one place, and the LSD of rX, which contains the multiplier, is shifted into the MQC to control the number of additions.

4-134. During the $\overline{\text{IER}}$ (add) phase, the multiplicand is added to the sum of additions from the previous IER phase the number of times specified by the multiplier digit in MQC. The new sum is returned to rA for the next $\overline{\text{IER}}$ phase.

4-135. GENERAL DESCRIPTION. The multiplication method used in the processor is illustrated by the sample problem in figure 4-9. In the problem, the number 1234 is multiplied by 4321. Although the processor can multiply two ten-digit numbers, the sample problem is confined to two four-digit numbers. Figure 4-9a shows the method used by the computer; figure 4-9b shows the method as used in the processor circuits. The circled digits and the final four digits in figure 4-9a are the product digits. The first computed product digit is 4 and it is the LSD of the final product.

4-136. During the M1 step, the multiplication sentinel (S) is placed in the LSD of rA, and the multiplier is placed in rX. The M2 step begins with a shift phase in which the sentinel is shifted from rA to the MSD position of rX. The multiplier in rX is also shifted right, causing the LSD, 1, to be transferred to the MQC. The next phase of M2 is the add phase in which the multiplicand from rL is added to the contents of rA the number of times specified by the digit stored in MQC. The MQC digit is 1; therefore the multiplicand 1234 is added to the contents of rA, which at this point is 0000. The sum of the addition, 1234, is returned to rA. The MQC counts down to 0 after the necessary number of additions has occurred, and a zero digit in MQC initiates the next shift phase.

4-137. In the next shift phase, the contents of rA (1234) is shifted to the right, causing the LSD of the first partial product, 4, to be shifted into rX. At the same time that rA and rX are shifted, the sentinel advances to the right one more place. The LSD of the multiplier in rX, 2, is shifted into MQC to control the number of additions in the next add phase. The multiplicand, 1234, is added to the shifted contents of rA, 0123, twice as specified by the MQC. The two additions produce the second partial product, 2591. The second partial product is also shifted, the LSD, 1, becoming the second quotient digit when placed in the MSD of rX. The product digits and sentinel in rX again shift to the right, creating the MSD vacancy into which the partial product digit, 1, is placed. The LSD of rX, 3, is transferred to the MQC. The MQC therefore permits three additions of the multiplicand, after which a zero reading starts the next shift phase. This process of alternating shift and add phases continues until the sentinel reaches the LSD position of rX and is shifted into MQC. The sentinel combination in MQC ends the multiplication with the most significant part of the product in rA and the least significant part in rX.

4-138. DETAILED DESCRIPTION. The following preparations for multiplication take place before the M1 step:

(1) The multiplicand is placed in rL and its sign in the rL sign flip-flop by a transfer instruction.

(2) The instruction word is located in the search-for-instruction step.

(3) The multiply instruction, 85, is staticized in the static register.

(4) The multiplier in the storage location designated by m is read from the memory in the search-for-operand step.

(5) The STR flip-flop 1 is jammed to 1. The combination in the STR is 1011 1X01, which generates function signals at t0A and t0B to carry out the M1 step.

4-139. M1 STEP. The sign of the final product in multiplication is determined during the M1 step (figure 4-10a) by the rA sign flip-flop, which compares the signs of the multiplier and multiplicand. The sign of the multiplicand already has been stored in the rL sign flip-flop during the transfer instruction. The sign of the multiplier enters the

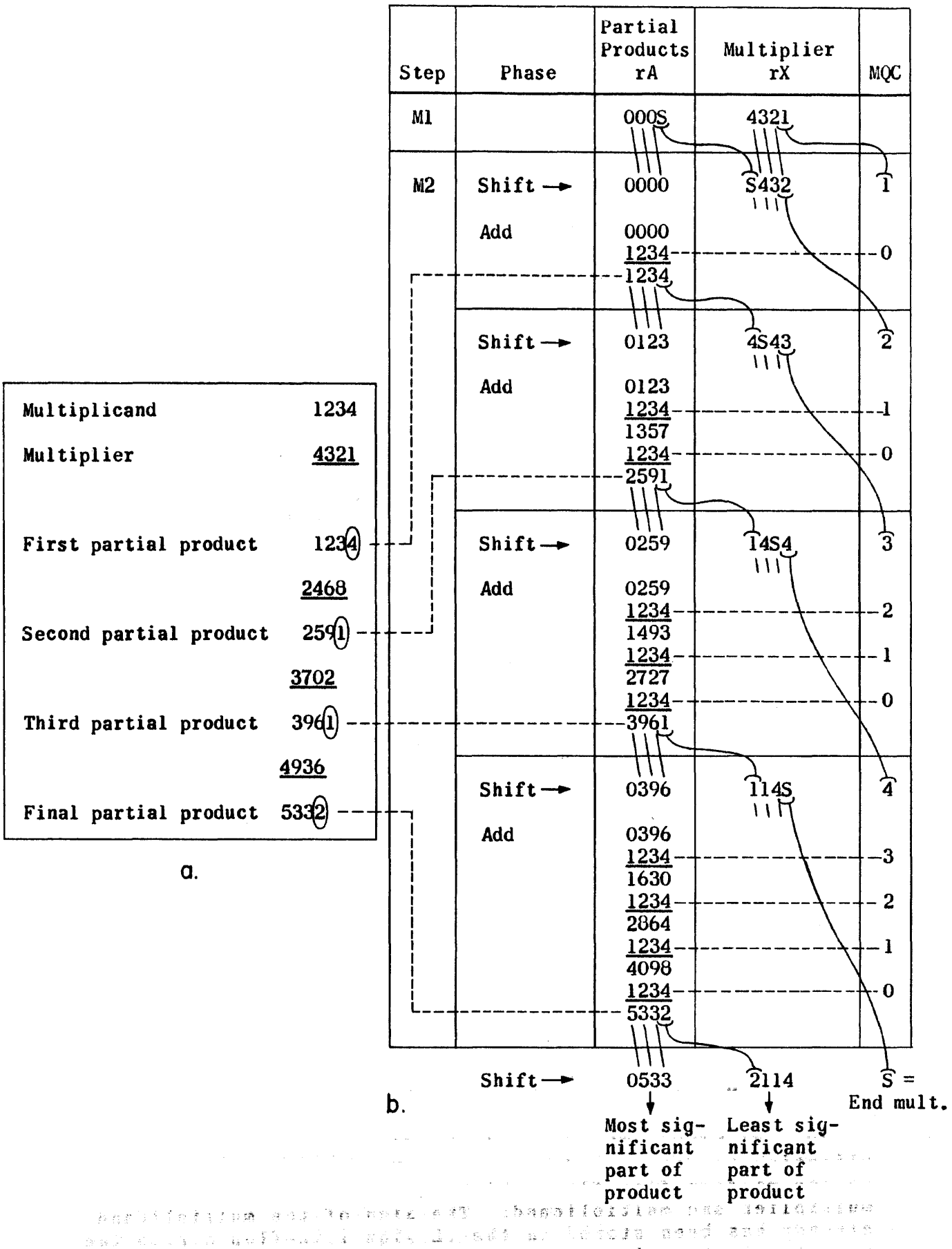


Figure 4-9. Multiplication Process 3113

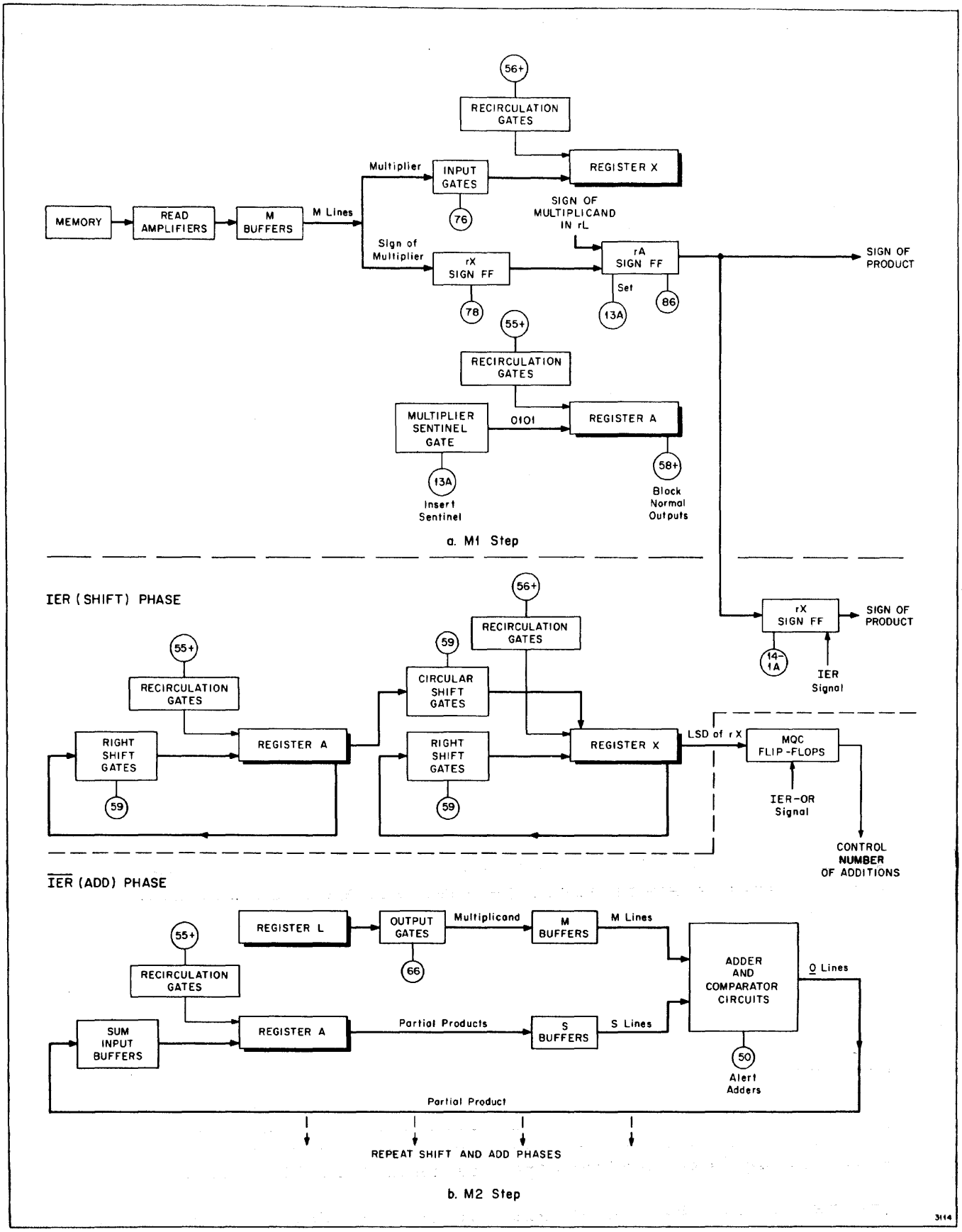


Figure 4-10. Multiply Instruction

rX sign flip-flop at t0B at gates 34 and 35, which are alerted by function signal 78. At t0A, the rA sign flip-flop is set to A- by FS 13A to prepare for the comparison of the signs of the multiplier and multiplicand. At t11B of the M1 step, FS 86 alerts gates 1 and 2 of the rA sign flip-flop. If the signs of the multiplier and multiplicand are alike (X+ and L+, or X- and L-), the flip-flop is restored to A+. If the signs are unlike, gates 1 and 2 are blocked and the flip-flop remains set to A-.

4-140. At t0A, FS 62 operates gate 19 of the clear-MQC circuit. At t1A the CLQ signal clears the MQC FF's which receive the LSD of the multiplier in the M2 step. At t0B, FS 56+ blocks the recirculation gates of rX, and FS 76 alerts the input gates to the multiplier which enters on the M lines.

4-141. Register A is cleared during the M1 step so it can receive the partial products during the M2 step. At t0B, FS 55+ blocks the recirculation gates of rA. At t1B, FS 13 operates the multiplier sentinel gate of rA. When this gate operates, it jams 1's into the A1 and A3 subregisters. As a result, the LSD of rA is the multiplier sentinel combination, 0101. Register A now contains the sentinel and nine zero digits. Function signal 58+ blocks the normal output gates of rA so that the sentinel remains within the register.

4-142. At t10B of the M1 step, FS 64 alerts the STR stepping gate which jams STR FF2 to a 1 output to initiate the M2 step. Function signals for the M2 step are generated at t0A and t0B of the M2 step.

4-143. At t11B, gate 11 of the IER FF samples the STR output signals which indicate that the M2 step has been staticized, and the $\overline{Q}$ (0) outputs of the MQC FF's, which indicate that the circuit has been cleared. If these two conditions are met, the IER FF is set at t0B to initiate the IER phase of the M2 step. The flip-flop generates low control signals IER at t0B and IER-OR at t1B, high control signals IERA1+ from t0A to t0B only, IERA2+ at t1A, and IER+ at t1B. These signals control the IER phase of the M2 step.

4-144. M2 STEP. The M2 step staticized in the static register generates FS 55+ and $\overline{66}$. These signals are present during both the IER and $\overline{\text{IER}}$ phases of M2. Other function signals, however, are generated during each of the two phases by the outputs of the IER FF.

4-145. IER PHASE. At the beginning of the IER shift phase, control signal IERA1+ generates FS 59 for one pulse time from t0B to t1B. Beginning at t1B, control signal IERA2+ is high, and it generates FS 59 and 56+.

4-146. At t0A of the IER phase, FS 14-1A jams the rX sign flip-flop to X+. At t0B of the IER phase, the low IER control signal alerts rX sign flip-flop, gate 33 which samples the sign of rA from the rA sign flip-flop. If the sign of rA is minus, the rX sign flip-flop is restored to X-; if it is plus, the flip-flop remains set to X+. Thus the signs of both parts of the final product will be identical.

4-147. Function signal 59 alerts the right shift gates of rA and rX and the circular shift gates of rX so that a right shift of both registers can take place. Function signal 56+ blocks the recirculation gates of rX; function signal 55+ blocks the recirculation gates of rA. During the first IER phase, the right shift operation causes the multiplication sentinel, in the LSD of rA, to be shifted into the MSD position of rX. The LSD of the multiplier in rX is at the same time shifted into the MQC FF's. The IER-OR control signal alerts the input gates of the four MQC FF's to receive the outputs of the rX subregisters.

4-148. Function signal 66 alerts the output gates of rL, which stores the multiplicand during multiplication. The multiplicand in rL is transferred to the M lines and into the adder circuits in preparation for the add phase ($\overline{\text{IER}}$). The partial products stored in rA after every shift phase are on the S lines and are also inputs to the adder circuits.

4-149. During every IER shift phase, the IER control signal alerts gate 18 of the ending pulse-circuit to sample for the 0101 sentinel in MQC. When the sentinel is present in the MQC FF's, the gate is permissive and the EP signal is generated to end multiplication.

4-150. $\overline{\text{IER}}$ PHASE. During this phase the contents of rA are added in the adder to the multiplicand from rL. The $\overline{\text{IER}}$ add phase is initiated at t11B of the IER phase by the t11B+ timing signal at gate 40 of the IER FF. The signal restores the flip-flop at t0A, and the polarities of the signals generated by the flip-flop during the IER phase are reversed for the $\overline{\text{IER}}$ phase. The IER+ output, which is now low, goes to the instruction decoder. If the staticized M2 signals are present and the IER+ signal is low, FS 14-2A is generated. Function signal 14-2A in turn generates FS's 50 and 61. Function signal 50 alerts the binary and quinary adders. Function signal 61 alerts the countdown gates of the MQC.

4-151. The partial product in rA, after every shift phase, and the multiplicand are added in the binary and quinary adders, just as in addition (paragraph 4-94). The sum of each addition is returned to rA. The MQC countdown circuit counts down to zero from the multiplier digit stored in it during the shift phase. When the reading in the MQC FF's is zero, the IER FF is set to initiate the IER shift phase again.

4-152. After the last multiplier digit has been processed and the necessary additions have been performed, the sentinel is shifted into the MQC FF's. In the next shift phase, the IER signal alerts the ending pulse gate 18. If the sentinel (0101) is stored in MQC, the Q1 and Q3 signals from the MQC FF's are low. Only the sentinel has 1 bits in these positions. The EP is generated and, at t0B, a search for the next instruction is initiated.

4-153. DIVISION

4-154. In the method of non-restoring division used in the UCT system the programmer scales the dividend so that it will be smaller than the divisor. Within the processor the dividend is logically multiplied by ten and complemented. The divisor is added to this new form of the dividend until addition produces an overflow carry. The remainder is complemented and multiplied by ten (shifted left), and the divisor is again added to this new dividend.

4-155. The number of divisor-to-dividend additions is monitored for each step and these numbers (or their complements) become the final quotient digits. The end of the division process is signaled by a sentinel that has been stored in the least significant digit position of a register and is shifted with each step toward the most significant digit position. When the sentinel becomes the most significant digit, the division process ends.

4-156. GENERAL DESCRIPTION. In the first phase of the division process, the dividend is shifted left and complemented. In the next phase, the divisor is added to the complemented dividend. Because the dividend is in its complemented form, the resulting quotient digit is also in complemented form. The quantity which remains after the divisor-to-complemented-dividend additions is the complemented remainder as shown:

```
                   complemented quotient
          ___________________________
divisor  ) complemented dividend
         / +      divisor
          ___________________________
           complemented remainder
```

The complemented remainder becomes the dividend of the next phase in which the next quotient digit will be determined. Before being used as the next dividend, however, this complemented remainder is complemented again to return it to its original or true form as shown:

```
                    quotient
                ____________
divisor        )dividend
               /+ divisor
                ___________
                remainder
```

4-157. In the next succeeding phase the dividend, quotient, and remainder again become the complemented form of their actual value. The cycles of the shift and complement phases alternate throughout the division process, and are differentiated as even cycles and odd cycles.

4-158. The odd cycles of the shift and complement phase always precede the even cycles. In the odd cycles the dividend, the quotient, and the remainder less the divisor are the 9's complement of their real value. In the even cycles the three quantities are represented in their uncomplemented form.

4-159. It must be remembered that although the quantity contained in rA at the end of every add phase is the remainder of that phase it becomes the new dividend necessary for the determination of the second quotient digit. Therefore, to arrive at a five-digit quotient, there must be five dividends, each dividend being the remainder of the previous division.

4-160. The quotient digits derived from each cycle are stored in rX, the contents of which is left shifted and complemented until division ends. The quotient digit derived during an odd cycle is complemented an odd number of times so that when it is transferred to rA it is in its true form. The quotient digit derived during an even cycle is complemented in rX an even number of times so that when transferred to rA it too is in its true form.

4-161. In the sample division problem (figure 4-11) the divisor, 0.133, is divided into the dividend, 0.032. The MQC column shows the state of the MQC countdown circuit at each successive addition. The Phase column shows whether the computer is in the shift-complement phase (OR) or the add phase ($\overline{OR}$). The distinction between odd and even OR phases is explained in paragraph 4-174.

Step	MQC	Phase	rA	rX	rL
D1	0		003200	000001	013300
D2	10	OR odd	//// 967990	///// 999980	013300
	9	$\overline{OR}$	+ 133 981290		
	8		+ 133 994590		
	7		+ 133 C007890 ////	999980 ////	
	10	OR even	921000	000120	
	9	$\overline{OR}$	+ 133 934300		
	8		+ 133 947600		
	7		+ 133 960900		
	6		+ 133 974200		
	5		+ 133 987500		
	4		+ 133 C000800 ////	000120 ////	
	10	OR odd	991990	998750	
	9	$\overline{OR}$	+ 133 C005290 ////	998750 ////	
	10	OR even	947000	012400	
	9	$\overline{OR}$	+ 133 960300		
	8		+ 133 973600		
	7		+ 133 986900		
	(6)		+ 133 C000200	012400	
D3	0		024060 Final Quotient	000200 Carry	013300 Divisor

OR = shift and complement phase $\overline{OR}$ = add phase

```
          .2406
.133:   .0320000
          266
           540
           532
             800
             798
              02
```

Figure 4-11. Division Process 3115

4-162. For simplification the division process described is for a six-digit register rather than the 12-digit register used in the UCT system. The least-significant digit position in each register is the sign position and remains unchanged throughout the process. The most significant digit is the SBW position.

4-163. In the D1 step the following takes place:

(1) MQC is cleared to 0.

(2) The dividend is placed in rA.

(3) The division sentinel (0001) is stored in LSB position of rX.

4-164. The first phase of D2 is the odd OR phase, in which the contents of rA and rX are individually shifted left and complemented. The 003200 combination in rA therefore becomes the quantity 967990. Similarly, the 1 sentinel and the contents of rX become the quantity 999980.

4-165. In the next phase, the $\overline{OR}$ phase, the divisor is added to the dividend. The 133 divisor can be added to the complemented dividend three times before a decimal carry occurs. Meanwhile, the MQC counts down from ten to seven. This ends the $\overline{OR}$ phase with a 7 digit in MQC, the remainder in rA, and the rX sentinel shifted one digit position to the left.

4-166. In the next phase, the even OR, the remainder is shifted left and complemented to the quantity 921000, which becomes the new dividend for the computation of the next quotient digit. In the left shift operation, a 0 replaces, in the LSD position, the MSD shifted to the left out of the register. The quotient digit 7, stored in MQC, is placed in the LSD position of rA after the sentinel is shifted left, and both are complemented. The quantity in rA becomes 000120, of which the 1 is the sentinel and 2 is the first quotient digit. At the end of the OR phase the MQC is jammed to ten.

4-167. In the next $\overline{OR}$ phase a count of the divisor is again added to the dividend. Six such additions take place before a decimal carry occurs. As a result, the MQC counts down to 4 and the $\overline{OR}$ phase ends.

4-168. The next OR phase causes the remainder, 000800, to

be complemented and shifted, the sentinel to be shifted, the digit 4 from MQC to be placed in rX, and the contents of rX to be complemented. The MQC is again jammed to ten.

4-169. Addition of divisor to dividend (remainder) again takes place in the $\overline{OR}$ phase. Only one addition takes place before decimal carry occurs; thus, the digit in MQC is 9.

4-170. In the next OR phase, the remainder 005290 is shifted and complemented to a quantity of 947000, the new dividend for the last add phase. The sentinel is shifted into the next MSB position. The quotient digit 9, computed in previous add phase, enters the MSB position of rX, and the contents of rX is complemented. The quantity in rX is now 012400. The 0 is the SBW, 1 is the sentinel, 2 is the first final quotient digit, 4 is the second final quotient digit, and 0 is the sign position.

4-171. The final addition phase, $\overline{OR}$, steps MQC down to 6 and leaves a remainder in rA of 000200.

4-172. In the previous OR phase the 1 sentinel was shifted to the MSB position of rX, signifying the end of division. The $\overline{OR}$ add phase takes place as described, but the sentinel has begun to initiate the D3 step, which starts as $\overline{OR}$ ends.

4-173. In the D3 step, the contents of rA and rX are interchanged; the final quotient is stored in rA and the final remainder in rX. The quotient digit, 6, derived from the last set of additions, goes uncomplemented to the LSD position of rA to become part of the quotient. The sign of the quotient also is computed during D3 in the rA sign flip-flop.

4-174. Examination of the problem shows that the quotient digits resulting from an odd OR phase are complemented an odd number of times; the quotient digits resulting from an even OR phase are complemented an even number of times. This process ensures that the final quotient, transferred to rA, is the true quotient.

4-175. DETAILED DESCRIPTION. The following assumptions are made in the discussion of the D instruction:

(1) The search-for-instruction step has taken place.

(2) The instruction has been staticized.

(3) The dividend (operand) has been located by the second search step.

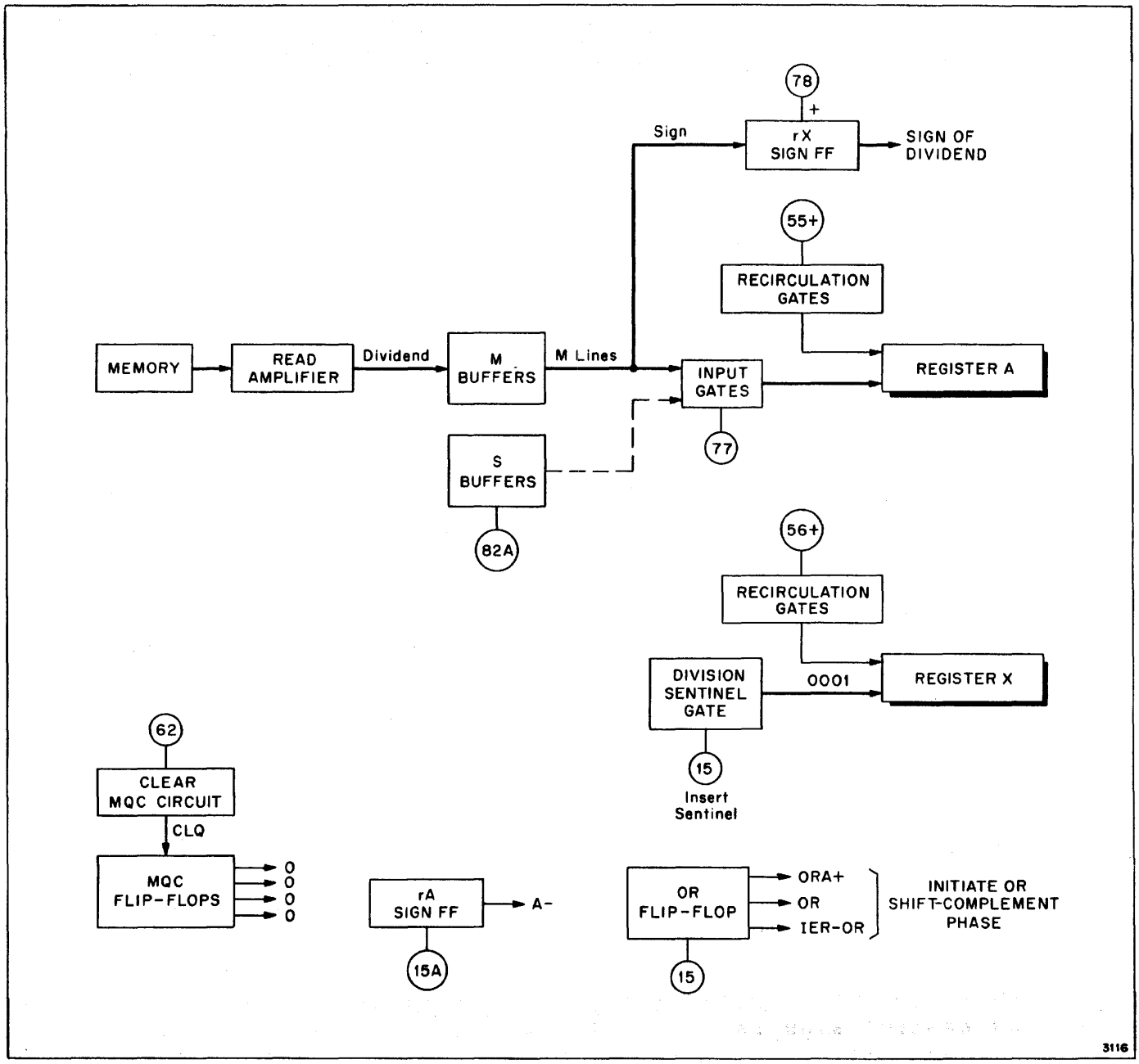


Figure 4-12. D1 Step of Division

(4) The divisor has been placed in rL by a previous instruction, either an L or K instruction.

4-176. D1 STEP. In the first execution step of division, D1, (figure 4-12) registers A and X are prepared for division,

the rX and rA sign flip-flops are prepared for the computation of the sign of the quotient and the sign of the remainder, MQC is cleared, the division sentinel is placed in rX, and at the end of D1 the static register is stepped to D2. The divisor has been stored in rL by a previous instruction.

4-177. Function signal 55+ blocks the recirculation gates of rA to clear the register of its contents. Function signal 77 alerts the input gates of rA, making them permissive to the M and S lines. However, since only the dividend on the M lines is to be read into rA at this time, FS 82A holds the outputs of the S buffers low so that the S lines cannot read into rA.

4-178. Function signal 56+ blocks the recirculation gates of rX to clear the register. The division sentinel is generated by FS 15 at the division sentinel gate to place an 0001 combination in the LSD position of rX.

4-179. Function signal 62 alerts gate 19 of the clear MQC circuit at t0B to generate the CLQ signal at t1B. The CLQ signal clears the MQC FF's to zeros.

4-180. The sign of the divisor has been stored in the rL sign flip-flop by a previous instruction. At the beginning of D1, the rX sign flip-flop is set to X+ at t0B by FS 78 at gate 35. If the sign of the dividend is minus, gate 34 is permissive, causing an X- to be stored. The signs of the divisor and dividend are sampled in the D3 step to determine the sign of the quotient.

4-181. At t11B of the D1 step FS 15 alerts gate 9 to set the OR FF, (figure A-15) initiating the first OR cycle of D2 at t0B. The flip-flop generates the IER-OR, ORA+, and OR control signals.

4-182. At t10B of D1 FS 64 alerts the STR stepping gate to step STR flip-flops 1 and 2 to outputs STR1 and STR2. The combination in STR is now 1000 1X11 which generates the D2 function signals.

4-183. D2 STEP. The D2 step consists of alternating shift and complement phases (OR) and add phases ($\overline{OR}$). The first phase of D2 is always the OR phase and is followed by the $\overline{OR}$ phase.

4-184. OR PHASE. The OR phase (figure 4-13) is initiated by the setting of the OR FF at the end of D1 (figure 4-12). The D1 function signal 15 sets the flip-flop at t11B at gate 9.

4-185. The ORA+ control signal, generated during D1, is a high signal which sets the CP FF, producing a low CP output. Signal ORA+ also generates a CLQ signal at the clear-MQC circuit at buffer 42. The CLQ signal clears the MQC flip-flops.

4-186. COMPLEMENTING AND LEFT SHIFT OF rA. The contents of rA (dividend) normally reads onto the S lines; when normal recirculation of the register is blocked by FS 55+, the register is cleared. The first three bits of the dividend (S1, S2, and S3) on the S lines are read into the quinary-adder complementer circuit. The CP signal, which alerts the complementing gates, causes the first three bits of the dividend to be complemented. The quinary bits (S1C, S2C, and S3C) return to rA at the left shift gates. In recirculating through the complementer circuit the first three bits are delayed one pulse time, the time necessary to effect a left shift of rA. The fourth bit of rA is delayed (shifted left) within rA and recirculated to the left shift gates as an A"4 signal. The fourth bit is complemented by gate 42 which is alerted by the CP signal. Function signal 71 alerts the rA left-shift gates, and the left-shifted, complemented dividend enters and is stored.

4-187. COMPLEMENTING AND LEFT SHIFT OF rX. Function signal 56+ blocks the recirculation gates of rX, and the contents of the register including the sentinel enter the input gates of the MQC FF's. The IER-OR control signal, generated during D1, alerts these gates. The MQC FF's, which have been cleared by the CLQ signal, supply one pulse time of delay for the X1, X2, X3, and X4 outputs of rX. One pulse time after entering the flip-flops the X input bits become the Q outputs of the flip-flops and return to rX at the quotient complementing gates. The OR control signal alerts these gates, which complement the Q inputs from MQC. In the first OR cycle only the sentinel is stored in rX whereas in succeeding OR cycles the sentinel and quotient digits are stored in rX. The entire contents of rX is complemented and shifted left.

4-188. COMPLEMENTING THE LSD OF rA. In the first OR phase the original dividend is complemented and left shifted.

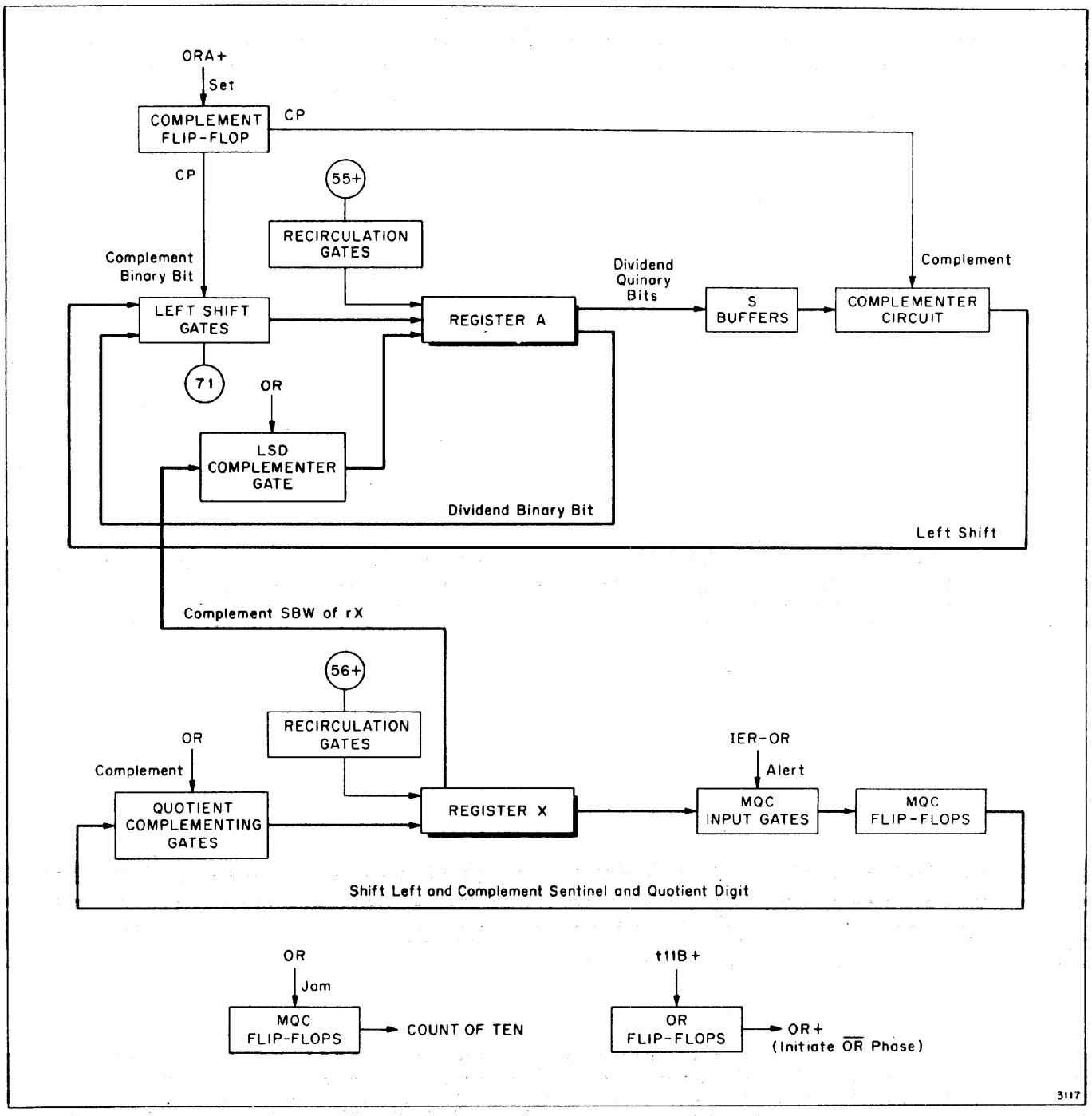


Figure 4-13. D2 Step, OR Phase

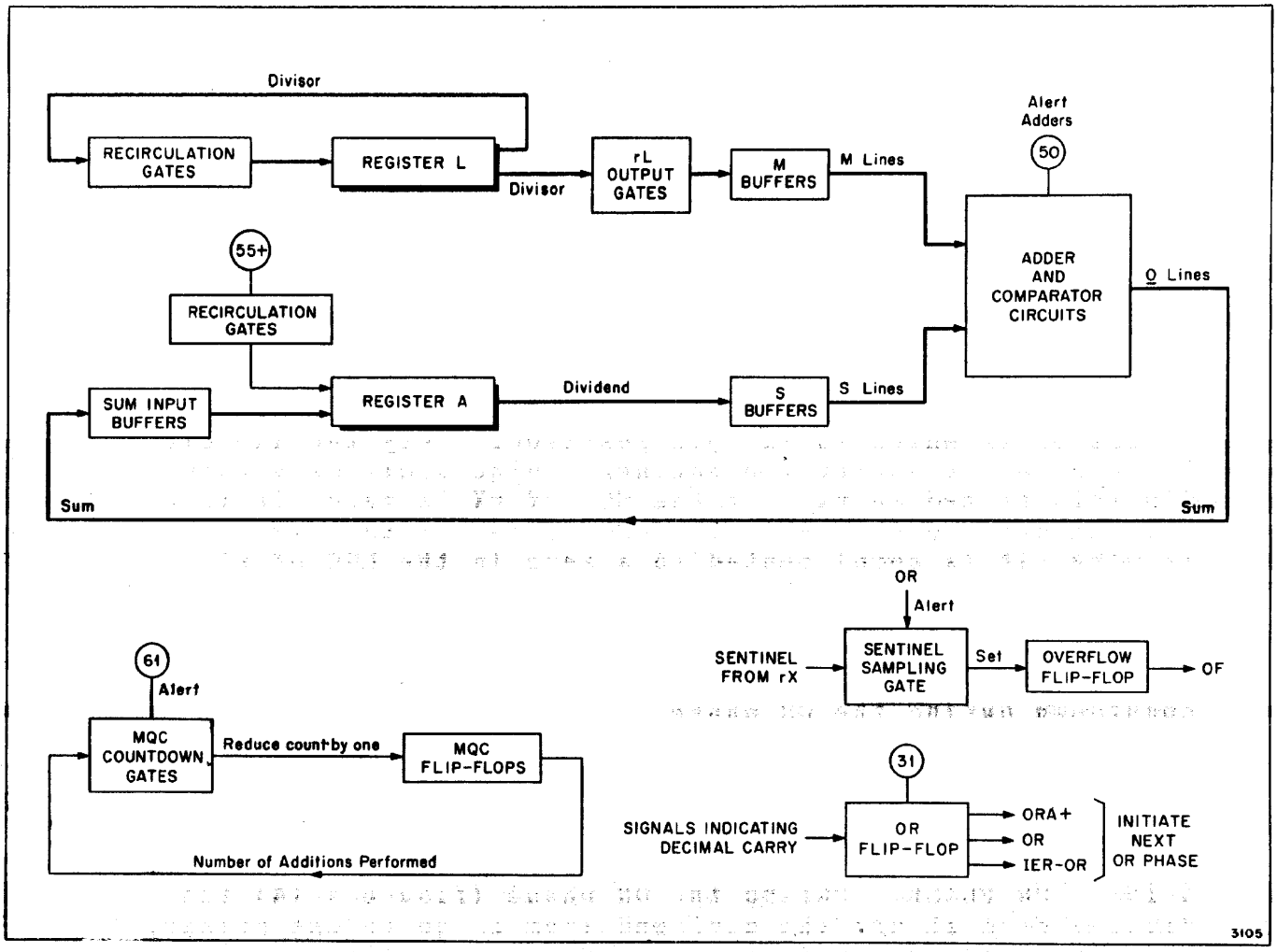


Figure 4-14. D2 Step, $\overline{OR}$ Phase

In an ordinary left shift of rA instruction, the vacancy created in the LSD position is filled with a 0. However, since in division the dividend is complemented, the digit which becomes the LSD also must be complemented. On alternate phases, therefore, the LSD of rA is either a 9 or a 0. During the odd OR phases (shift and complement) the LSD of rA is 9. During the even OR phases (complement and shift) the LSD of rA is 0.

4-189. The LSD complementer gate of rA is alerted by the OR output of the OR FF. At t1B the gate samples the X4D output of rX which is the SBW position. Only the X4D bit is necessary to determine whether a nine digit or a zero digit is stored in rX. If the SBW of rX is zero, it is complemented to a nine in the LSD of rA; if the SBW of rX is nine, it is complemented to a zero in the LSD of rA.

4-190. JAM MQC. At t11B of each OR phase the OR signal makes gate 1 permissive, jamming ones into MQC FF's 1, 2, and 4. This action stores a ten combination in MQC to allow countdown during the $\overline{OR}$ phase.

4-191. INITIATE $\overline{OR}$ PHASE. At the end of the OR phase a t11B+ high signal restores the OR FF to an output of OR+, initiating the $\overline{OR}$ phase.

4-192. $\overline{OR}$ PHASE. During the $\overline{OR}$ phase (figure 4-14) the divisor from rL and the dividend from rA go to the quinary and binary adder circuits where the two quantities are added. The MQC countdown circuit monitors the number of additions. The sum of the divisor-to-dividend additions returns to rA for temporary storage.

4-193. ADDITION. The divisor, already placed in rL by a previous instruction, is recirculating within rL. Function signal 66 and the OR FF output, OR+, which is low at this time, make the rL output gates permissive. The contents of rL continues to recirculate but also reads onto the L lines to the M buffer and the M lines. The M lines go to the quinary and binary adders. Function signal 55+ blocks recirculation of rA, and the dividend reads into the S buffers and the S lines. The S4 bits of the dividend on the S lines read into the binary adder circuit. The three lowest-order bits read into the quinary adder circuit, as in the addition process (paragraph 4-94). Function signal 50 alerts the quinary and binary adders to the M and S inputs. The divisor and dividend are added and the sum emerges on the O lines. With each addition, the MQC countdown gates,

alerted by FS 61, reduce the number stored in the MQC FF's by one every t1B of the OR phase. When decimal carry occurs, the remainder is still on the 0 lines and the MQC contains the number of additions performed. Function signal 31 alerts gate 10 of the OR FF throughout the D2 step to sample the $\overline{OF}$, A, C, S3, and S4 signals. The A and C signals at t11B indicate that the addition of the p10 digit has produced a decimal carry to the p11 digit. The S3 and S4 signals (1100=9) indicate that the p11 digit is a nine. If a decimal carry is added to the final digit to be added, a nine, a carry condition, will occur. The $\overline{OF}$ FF is restored during the add phases, generating low $\overline{OF}$. If present, these signals set the flip-flop to the OR phase by generating the OR, ORA+, and IER-OR control signals.

4-194. END OF D2 STEP. During every OR phase of D2, the division sentinel advances toward the MSD position of rX. At t9B of the tenth OR cycle the sentinel sampling gate 27 examines the X1 bit of rX which, if present, indicates the sentinel in the MSB position of rX. If the gate is alerted by OR, indicating that the last addition has ended, the OF FF is set, generating OF. The OF signal makes it possible to step to D3, if the last $\overline{OR}$ phase is complete.

4-195. D3 STEP. In the D3 Step (figure 4-15) the contents of rA and rX are interchanged placing the final quotient in rA and the remainder (carry) in rX. The sign of the quotient also is computed during D3 and the instruction is ended.

4-196. INTERCHANGING rA AND rX. Function signal 31 alerts input gate 001 of the D3 FF (figure A-2) throughout the D2 step. When the following signals are present on the gate, the D3 FF is set, initiating the third step of division, D3:

(1) OF, indicating that the sentinel is the MSD of rX.

(2) A and C, which indicate that a carry or remainder is present.

(3) S3 and S4, which indicate the presence of the digit 9 in the SBW of rA.

(4) STR8 which indicates that improper division has not occurred. (Improper division generates EP, which clears all STR FF's to zero.)

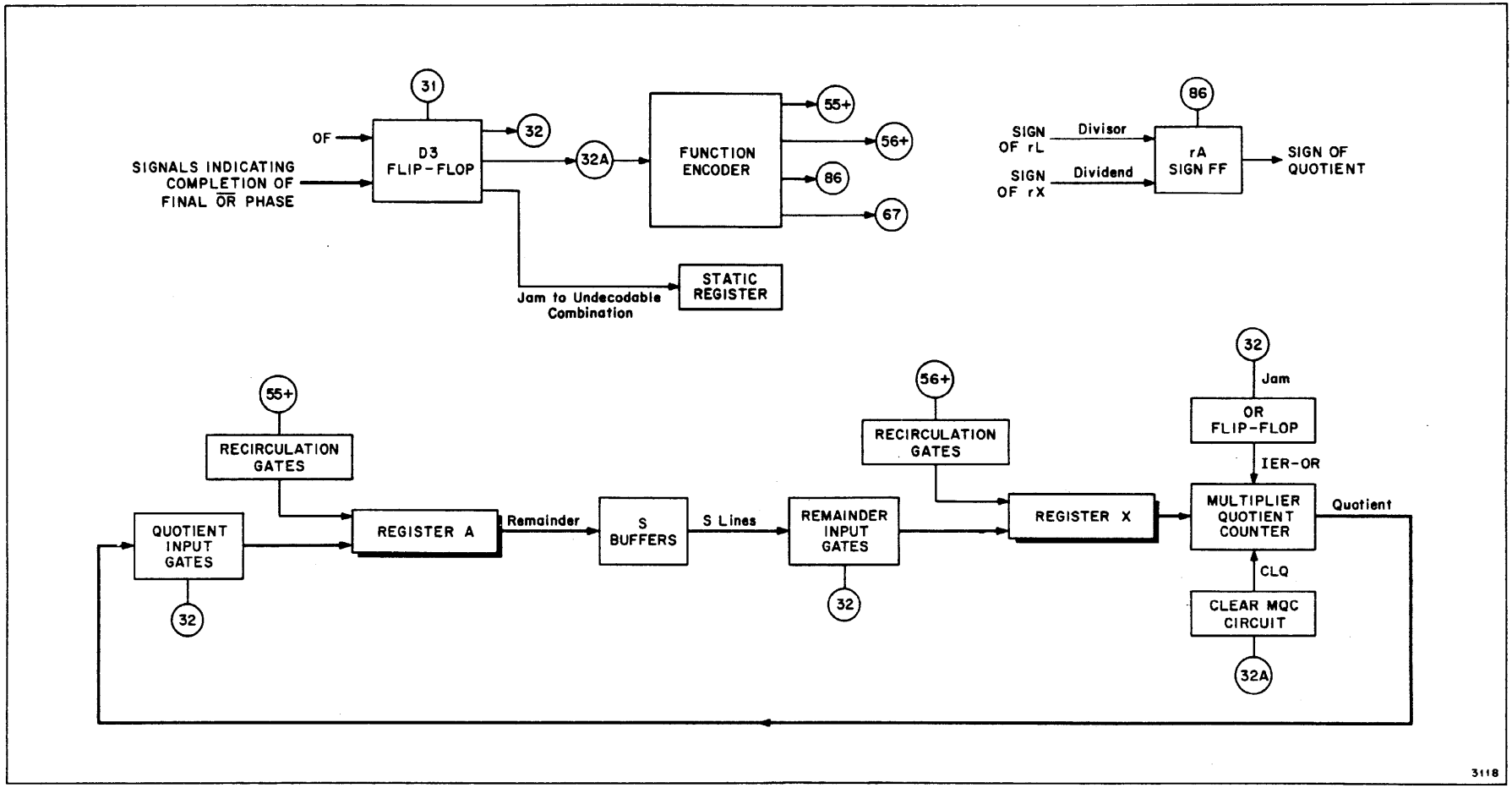


Figure 4-15. D3 Step

Because OF is low, $\overline{OF}$ is high, blocking gate 10 of the OR FF, so that the flip-flop remains restored. Coincidence of these signals at t11B of the 10th $\overline{OR}$ phase sets the D3 FF, which generates FS 32 at t0B. Function signal 32A is generated at t1A. When the D3 FF is set, it also steps FF5 to a 1 output, resulting in a combination that cannot be decoded by the instruction decoder. As a result, no function signals are generated by the instruction decoder since the D3 FF is now generating its own function signal. The flip-flop is restored every word time by the t11B+ signal.

4-197. Function signal 32A drives the function encoder to generate FS 55+, 56+, 86, and 67. Function signal 55+ blocks recirculation of rA. The remainder, from rA, goes to the S buffer and the S lines. Function signal 32 alerts the remainder of rX input gates to the S inputs. (Function signal 56+ blocks recirculation in rX. Function signal 32 also generates the IER-OR control signal which alerts the input gates of the MQC FF's to the X lines from rX. Function signal 32A clears the MQC FF's every pulse time by generating the CLQ clear signal at the clear MQC circuit.)

4-198. As the quotient in rX is shifted through the one pulse time of delay supplied by the MQC FF's, the remainder on the S lines reads into rX. The outputs of rX are read through the MQC FF's. Function signal 32 alerts the quotient input gates of rA to the quotient on the Q lines. Thus the remainder is transferred into rX and the quotient is transferred into rA. Function signal 67 ends the instruction at t11B of the D3 step by generating the EP signal at t9B.

4-199. SIGN OF THE QUOTIENT. During the D3 step the sign of the quotient is computed at gates 01 and 02 of the rA sign flip-flop. Function signal 86 alerts these gates to sample the sign of the divisor, which is stored in the rL sign flip-flop, and the sign of the dividend, which is stored in the rX sign flip-flop. The rA sign flip-flop already has been set to A-. If the X and L signs signify a minus sign for the quotient, the rA sign flip-flop remains set to A-. If the X and L outputs signify a plus sign, the flip-flop is restored to store an A+ for the sign of the quotient.

4-200. ERROR CIRCUITS

4-201. A constant check on system errors or abnormal conditions in the input-output devices is maintained by four flip-flops: the memory-read error flip-flop, the timing-error flip-flop, the cycling-unit error flip-flop, and the

I/O (input-output) abnormal-condition flip-flop. The signals produced by the setting of all of these flip-flops except the I/O abnormal-condition flip-flop, set the stop flip-flop. The signal generated by the stop flip-flop blocks the gates of the instruction decoder so that the next instruction staticized in the static register will not be executed until the cause of the condition is remedied and the operator restarts the computer. An error or abnormal condition which develops in one of the input-output devices sets the I/O abnormal-condition flip-flop.

4-202. MEMORY-CHECK FLIP-FLOP

4-203. When information is written into the main memory the check-bit computer (figure A19) ensures that an odd number of 1 bits is recorded on the drum for each digit. The check-bit computer performs the same check on information read from memory. If the digit read from memory contains an even number of 1 bits, a low CK signal is generated. If the digit contains an odd number of 1 bits, a low $\overline{CK}$ signal is generated. The CK and $\overline{CK}$ signals go to the memory check flip-flop to indicate whether a 0 or a 1 check bit was necessary when written on the drum. The value of the check bit being read from the drum is indicated by the DM5 signal output of the read circuits. The check bit is found on the DM55 line when reading from fast memory. The memory-check flip-flop is set to a high MRE output (Memory Read Error) to indicate an error under two conditions:

(1) If a 0 checkbit is present, as indicated by the low $\overline{CK}$ signal, and there is a 1 check bit being read as indicated by the low DM5 signal, or

(2) if a 1 check bit is present as indicated by a low CK signal and a 0 check bit indicated by the low $\overline{DM5}$ signal.

4-204. The memory-check flip-flop can be set to a high MRE output only if the check-timing flip-flop is set. This check-timing flip-flop is set at t0B if the RD signal is present, indicating that the main memory read flip-flop is set. The low output of the check timing flip-flop alerts gates 53A and 53B of the memory-check flip-flop to sample for check bit error condition.

4-205. The high MRE signal lights the MAIN STORAGE and PROCESSOR/OFF NORMAL indicators on the control panel and sets the stop flip-flop (figure A-5). The high $\overline{SP}$ output of the stop flip-flop goes to the instruction decoder

to block the generation of function signals for the execution of the next instruction until the error condition is remedied. The operator must press the start button generating a high ST (start) signal. This signal restores the memory read error flip-flop to a low MRE output and the stop flip-flop to a low $\overline{SP}$ output. The low $\overline{SP}$ signal enables the instruction decoder to generate function signals which enable the processor to resume operations.

4-206. TIMING-ERROR FLIP-FLOP

4-207. The timing-error flip-flop (figure A-17) samples the output digits of the timing band read circuits. When originally recorded on the timing band of the drum, each permanent timing combination contains an odd number of 1 bits. The timing error flip-flop maintains a constant check on the digit outputs of the timing-band read circuits to ensure that each digit contains an odd number of 1 bits. Thus, the operation of the memory drum and reading circuits is constantly checked.

4-208. Gates 2, 3, 4, and 5 of the timing error flip-flop test the TS1, TS2, and TS3 outputs of the timing band read circuits. Gates 6 and 7 test the TS4 and TS5 outputs. An even combination of 1 bits alerts two of the gates, setting the flip-flop to a high TE (timing error) output. For example, gates 2 and 6 operate if the TS combination is 0 0000. This combination contains an even number (zero) of 1 bits; therefore an error exists. Gate 2 operates to send a high signal to buffer 14 and then a low signal to gate 9. Gate 6 operates to send a high signal to buffer 16 and then a low signal to gate 9. Both inputs to gate 9 are therefore low, setting the flip-flop to a high TE output.

4-209. The timing-error flip-flop is set to a high TE output, indicating an error, under two conditions:

(1) If each of the two groupings of gates receives an even combination, (even + even = even) or,

(2) if each of the two groupings receives an odd combination (odd + odd = even).

A normal-odd input combination in one grouping plus an even input combination in the other grouping (odd + even = odd) puts a high on both set gates and the timing error flip-flop output, TE, remains low. If an error exists the high TE signal lights the TIMING ERROR and PROCESSOR OFF NORMAL indicators on the control panel and sets the stop flip-flop (paragraph 4-202).

4-210. CYCLING-UNIT ERROR FLIP-FLOP

4-211. The cycling unit error flip-flop (figure A-16) constantly checks the synchronization of the timing combination input to the cycling unit. This combination, TS4, TS3, $\overline{TS2}$ and TS1, must be present at t6B of every word time to properly synchronize processor and input-output operations. If this combination is absent at t6B, the cycling unit error flip-flop generates a signal that stops the processor and indicates the condition on the control panel for remedial action.

4-212. Gate 3 of the flip-flop is blocked by the high t6B+ signal at t6B, which is the exact time that the timing combination should be present. At any time other than t6B, the t6B+ signal is low. Therefore, if the bits of the timing combination are low at the gate at any time other than t6B, the gate is permissive, setting the flip-flop. Gates 4, 5, 6, and 7 check the individual bits of the timing combination for synchronization. These gates are alerted by the t6B- signal at t6B. A malfunction is present if any one of the $\overline{TS1}$, TS2, $\overline{TS3}$, and $\overline{TS4}$ signals is low at t6B, since only the timing combination signals should be low at that time. If the input signal to one of these gates is low at t6B, the gate is permissive. This condition again sets the flip-flop to a high CUE output. The high CUE signal lights the CYCLING UNIT ERROR and PROCESSOR OFF NORMAL indicators on the control panel and sets the stop flip-flop.

4-213. INPUT-OUTPUT

4-214. Any abnormal operation of one of the three input-output devices produces a signal that goes to one of the three set gates of the master I/O abnormal-condition flip-flop (figure A-2). On the next card cycle or print instruction, the flip-flop is set by the function signal generated by the instruction and the signal that indicates the abnormal condition. For example, abnormal operation of the printer causes a low AOP (Abnormal Operation Printer) signal to be generated. The next print instruction (16 or 11) in the program generates FS 41, which alerts gate 100 to the AOP indication. The permissive gate sets the I/O abnormal-condition flip-flop, generating high Jam I2A and Jam I2B signals. The Jam I2A signal sets FF 1, 2, and 6, to 1 outputs. The Jam I2B signal restores FF 5 and 8 to 0 outputs. This staticizes the second step of a test instruction. The abnormal condition signals, AOT and AOR,

from the read-punch and card-reader respectively, similarly affect the I/O abnormal-condition flip-flop when the instruction involving these devices is staticized. These signals set the flip-flop, which in turn, jams the STR FF's to the second stage of a test instruction.

4-215. This new reading generates function signals that transfer the unexecuted instruction still in rC to rA for storage. The second stage of a test instruction normally transfers the contents of rC to rA and also sets the CT FF so the next instruction is taken from m instead of c. (See I12, under instruction 22, Analysis of UCT Instructions.) This setting of the CT FF is prevented by an inhibiting Jam I2B signal which blocks the gate.

4-216. The contents of rC is transferred to rA for future use by the programmer. This transfer takes place without clearing rC. Jam I2A ensures that the next instruction is taken from c+1 address by setting the OF FF and restoring the CT FF. The assumption is made that the programmer has stored, in the c+1 address, a routine which brings the computer to a stop.

4-217. MANUALLY-CONTROLLED OPERATIONS

4-218. The computer can operate either on a continuous or a non-continuous basis. Although the computer normally functions in the continuous mode; facilities are provided, mainly for troubleshooting and testing procedures, to operate the computer in the non-continuous or step-by-step mode.

4-219. The operator controls the mode of operation of the computer by depressing one of the six pushbutton lights on the control panel under OPERATION. These buttons light when pushed and are mechanically interlocked so that only one can be pushed at a time. When the CONTINUOUS button is pushed, the computer operates continuously until a programmed stop instruction is staticized in the static register, until an error occurs, or until the operator stops computation by pushing the STOP button. To place the computer in a non-continuous mode one of the five buttons marked ONE LINE PRINT, ONE CARD RPU, ONE CARD HSR, ONE INSTRUCTION, or COMPARISON STOP must be pushed. The logical operation of the non-continuous modes are described in paragraphs 4-220 through 4-234.

4-220. ONE-LINE PRINT

4-221. When the operator depresses the ONE LINE PRINT button, the computer stops on the first step of the next

advance-and-print instruction. When this button is pushed an EPR signal is generated and sent to gate 25 of the stop flip-flop (figure A-5). The EPR signal remains at gate 25 until the conditions indicating that an advance instruction has been staticized are present. Function signal 41 alerts the gate when the advance-and-print instruction is staticized and the $\overline{STR4}$ signal signifies that an advance-and-print instruction has been called for rather than the paper-advance instruction, both of which generate FS 41. The low $\overline{ST}$ signal indicates that the start flip-flop has been set.

4-222. The one remaining signal necessary to make gate 25 permissive is the low output of the IOS FF. Because this signal is necessary in all of the non-continuous operations, the explanation of the one-line-print operation applies also to the four other operations.

4-223. When any one of the non-continuous pushbutton switches is energized, the NC (non-continuous) signal is generated. The NC signal is used to synchronize the control panel input signal with internal computer signals. It alerts gate 1 of the synchronizing flip-flop which is permissive at tOB if an EW signal is present (paragraph 3-194). Because the NC signal is generated by a manually-controlled switch, it may be a weak or partial signal at tOB. The synchronizing flip-flop stores the NC signal for two word times, during which the signal either dies out or builds up into a strong signal.

4-224. If the original NC signal dies out, the synchronizing flip-flop is restored and then set again with a strong signal when the next EW signal is available. At the end of the two word times, the signal goes to gate 9 of the IOS FF. Gate 9 is alerted by the EW signal and is permissive at tOB if the input signal is low, setting the flip-flop. The IOS FF generates a signal (IOS) which is sent to set gates 19, 22, 23, 24, and 25 of the stop flip-flop. The setting of the IOS FF generates a high $\overline{IOS}$ signal which restores the synchronizing flip-flop.

4-225. When gate 25 is permissive, it sets the stop flip-flop to low SP and high $\overline{SP}$ outputs. The $\overline{SP}$ signal blocks the instruction decoder so that no function signals are generated. When the operator pushes the START button a high ST signal is generated to restore the stop flip-flop.

4-226. ONE-CARD RPU

4-227. When the ONE CARD RPU button is pushed the computer stops on the first step of the next card cycle instruction involving the read-punch unit. When this button is pushed, a TECC signal is generated and sent to gate 19 of the stop flip-flop. Gate 19 is alerted by FS 29 during the first step of the next card-cycle instruction for the read-punch unit. If the $\overline{ST}$ and IOS signals are present the gate operates to set the stop flip-flop (paragraphs 4-223 through 4-225.)

4-228. ONE-CARD HSR

4-229. When the ONE CARD HSR button is pushed the computer stops on the first step of the next card-cycle instruction for the card reader. When this button is pushed, an RECC signal is generated and is sent to gate 22 of the stop flip-flop. Gate 22 is alerted by FS 28 only during the first step of the next card-reader card-cycle instruction. The $\overline{ST}$ signal makes the gate permissive, setting the stop flip-flop (paragraphs 4-223 through 4-225.)

4-230. ONE INSTRUCTION

4-231. When the ONE INSTRUCTION button is pressed, the computer stops after the next instruction is staticized. The operator uses this mode of operation to type information into the computer from the keyboard. When this button is pushed, a staticize-every-minus (SZE-) signal is generated and sent to gate 23 of the stop flip-flop. Function signal 2 alerts gate 23 during the staticize step of the next instruction. When the gate is permissive the stop flip-flop is set (paragraphs 4-223 through 4-225.)

4-232. COMPARISON STOP

4-233. The fifth non-continuous button, COMPARISON STOP, differs from the other four in that it stops the computer after a Q or T instruction has been executed but before the computer searches for the next instruction. The function of both the Q and T instructions is to compare two quantities and, on the basis of this comparison, to determine whether the next instruction should be taken from the m or c address. Two illuminated pushbutton switches on the control panel indicate to the operator whether the next instruction to be searched for will be

in the m or c address. When the operator has depressed the COMPARISON STOP button, the computer stops after the comparison is made and the next address has been selected. If the operator wishes to override the address determined by the comparison the correct button is pushed. The m or c button changes the state of the CT FF and lights the lamp associated with the chosen address. The RUN button is then pushed and the computer searches for the next instruction.

4-234. When the COMPARISON STOP button is pushed a QT Stop signal is generated and sent to gate 24 of the stop flip-flop. The STR signals that make gate 24 permissive are common only to the Q and T comparison instructions. If the inputs to gate 24 are low, the gate is permissive at t10B, setting the stop flip-flop. The t10B signal ensures that the comparison instruction will be completed before the stop flip-flop is set (paragraphs 4-223 through 4-225.)

4-235. KEYBOARD INPUT

4-236. The control panel keyboard enables the operator to change the contents of the arithmetic registers (A, X, or L), by typing in new information. An instruction word also is typed into rC at the keyboard to load the program into the memory at the c address specified in the instruction word. The keyboard consists of ten keys (0 through 9), a KEYBOARD ALERT pushbutton to energize the keyboard, a KEYBOARD READY light, which indicates that the keyboard is ready for use, and two WORD RELEASE pushbuttons labeled + and -. With the latter two pushbuttons, the sign of the word is inserted and the typed word is released from temporary storage into the selected register.

4-237. MANUAL OPERATION

4-238. At all times during computation, the contents of one of the four registers, A, C, X, or L, is displayed in lights on the control panel. Forty lights labelled REGISTER CONTENTS display the biquinary combinations of each of the 10 digits contained in the register. Two sign lights (+ and -) directly below these 40 lights indicate the sign of the word in the register. The operator selects the register to be displayed by pushing one of four illuminated pushbuttons on the control panel labelled REGISTER SELECTOR. These four buttons, C, A, X, and L, light when depressed and are mechanically interlocked so that only one can be operated at a time.

4-239. To type either instruction or data information into the computer, the operator selects a register and stops the computer by pushing the STOP button. When the computer stops, the operator pushes the ONE INSTRUCTION and KEYBOARD ALERT buttons. After a delay of approximately one second, the KEYBOARD READY lamp lights to signal the operator that the keyboard is ready for use. The operator can now type in the word, MSD first. When the complete word is typed, the operator pushes one of the word-release buttons, either + or -. The word is shifted into the selected register and its sign is placed in the proper sign flip-flop. The operator then pushes the CONTINUOUS button and the RUN button.

4-240. If a typing error is made, the operator can push one of the word-release buttons to release the word into the register, push the KEYBOARD ALERT button once again, and retype the word.

4-241. The operator can depress two keys at the same time to form bit combinations for characters not represented by keys on the keyboard. An alternate method is to type in two complete words. The operator then depresses one of the word-release buttons and the combination of the first and second shifts into the selected register. Here is a list of key combinations that the operator can use:

Biquinary Code	Key Combinations
0101	1 + 4
0110	2 + 4
0111	3 + 4
1101	9 + 1
1110	9 + 2
1111	4 + 8

4-242. The word typed into the computer is stored temporarily in four shift registers—read-punch shift register, card-reader shift register, card-reader group counter, and read-punch group counter (figure 4-16.) Each shift register stores 10 bits of the typed 40-bit word. The LSD of the word reads out of the shift registers first, although the operator types in the MSD of the word first. A CLIA-signal, which clears all four shift registers, is generated when the (keyboard alert) button is pushed. Signals necessary to shift the word are generated every time a key is released. When the entire word is typed and stored in the shift registers, the operator pushes one of the two word-release keys to supply the sign of the word and open the path into the selected registers.

4-243. DEPRESSING A KEY

4-244. Each time the operator depresses a key to place one digit into the four shift registers, the four-bit code for the desired digit is generated by the keyboard encoder matrix shown in figure A-5. The four bits of each digit are distributed onto four output lines. The SCI1 line goes into the read-punch shift register, the SCI2 line goes to the read-punch group counter, the SCI3 line goes to the card-reader shift register, and the SCI4 line goes to the card-reader group counter. Each bit is stored in first flip-flop of each register. Table 4-4 shows the outputs of the keyboard encoder which is controlled by the keyboard keys.

Table 4-4. Keyboard Encoder

Keyboard Key	Card-Reader Group Counter SCI_4	Card-Reader Shift Register SCI_3	Read-Punch Group Counter SCI_2	Read-Punch Shift Register SCI_1	UCT Code
0					0000
1				x	0001
2			x		0010
3			x	x	0011
4		x			0100
5	x				1000
6	x			x	1001
7	x		x		1010
8	x		x	x	1011
9	x	x			1100

4-245. A high SCI+ signal is generated from the beginning to the end of the keyboard input. The SCI+ signal goes to the card-reader group counter to block the setting of the sample-pulse 1 flip-flop and to the read-punch group counter to block the stepping of the row counter.

4-246. RELEASING A KEY

4-247. Each time a key is released a sprocket signal is generated. This signal is synchronized with the timing of computer signals by the EW and t4B- signals which set the sprocket-synchronizing flip-flop. This flip-flop generates a low SPR signal. This SPR signal sets the keyboard input flip-flop (figure A-27) at t4B, gate 218 when the EW signal is present. Three signals are generated—a low SCISB, a high SCISA, and a high RSP. The SCISA signal is available for 9 pulse times (t6A to t3A) and in all four registers it shifts each bit of information 9 places to the right. The SCISB signal supplies a recirculation path to the two group counters. The other two shift registers are already recirculating. The RSP signal restores the sprocket synchronizing flip-flop.

4-248. As a result of the operation of a key the four bits of the first digit stored in the four registers are shifted nine places to the right in all shift registers by the SCISA signal. This signal sends a high signal to the restore gates and a low to the set gates of the ten flip-flops of each register. Section 2-97 explains the operation of a shift register. As a result of this shifting, the four bits of the first digit are placed in the LSD position of the four registers. When the next key is depressed, the four bits of the next digit are stored in the first flip-flop of each register. When the key is released the SCISA signal again shifts the new bits nine places to the right and shifts the previously inserted bits the same number of places. The SCISB signal, sent to the group counters, permits the first bit in each counter to recirculate and become the second lowest order bit of the register. All ten digits of a word are read into the registers and shifted in the same manner. As a result, all digits of the word are in position in the register to be read out LSD first. See table 4-4.

4-249. SIGNING AND RELEASING THE WORD

4-250. After all ten digits of a word have been typed, the operator provides the word with a sign and releases the word into the previously selected register by depressing either of the two word-release buttons, labeled + and -. If the plus button is depressed, a WR+ signal is generated. If the minus button is depressed, a WR- signal is generated. The WR+ or WR- signal is stored temporarily in the word-release initial storage flip-flop and then is synchronized with computer timing by the setting of the word-release synchronizing flip-flop. The synchronized signal generated from this flip-flop sets the word-release flip-flop. When set, this flip-flop generates two signals, a high WRA and a low WRB. The WRB signal generates function signals that will open up the path into the selected register on the M lines and gate the information stored in all four registers onto the M lines. The WRA signal supplies the shift pulses needed to shift information out of the four registers.

4-251. The WRB signal alerts the DM gates 56A, 56B, 56C, and 56D (figure A-5) so the four output lines of shift registers read the contents onto the DM and M lines as follows:

Register	Output	DM line	M line
Read-Punch shift register	SR2	DM11	M1
Read-Punch group counter	TG10	DM21	M2
Card-Reader shift register	SR1	DM31	M3
Card-Reader group counter	G10	DM41	M4

The WRB signal also goes to the generate-instruction gates 57, 58, 59 and 60 (figure A-5), only one of which has been alerted. One of the RSC, RSA, RSL, or RSX signals alerts one of the gates, depending on which register has been selected by the operator. If the RSC signal has been generated, a generate-beta signal is produced. If the RSX signal has been generated, a generate-Y signal is produced. If the RSL signal has been generated, a generate-L signal is produced and if RSA signal has been generated, a generate-B signal is produced.

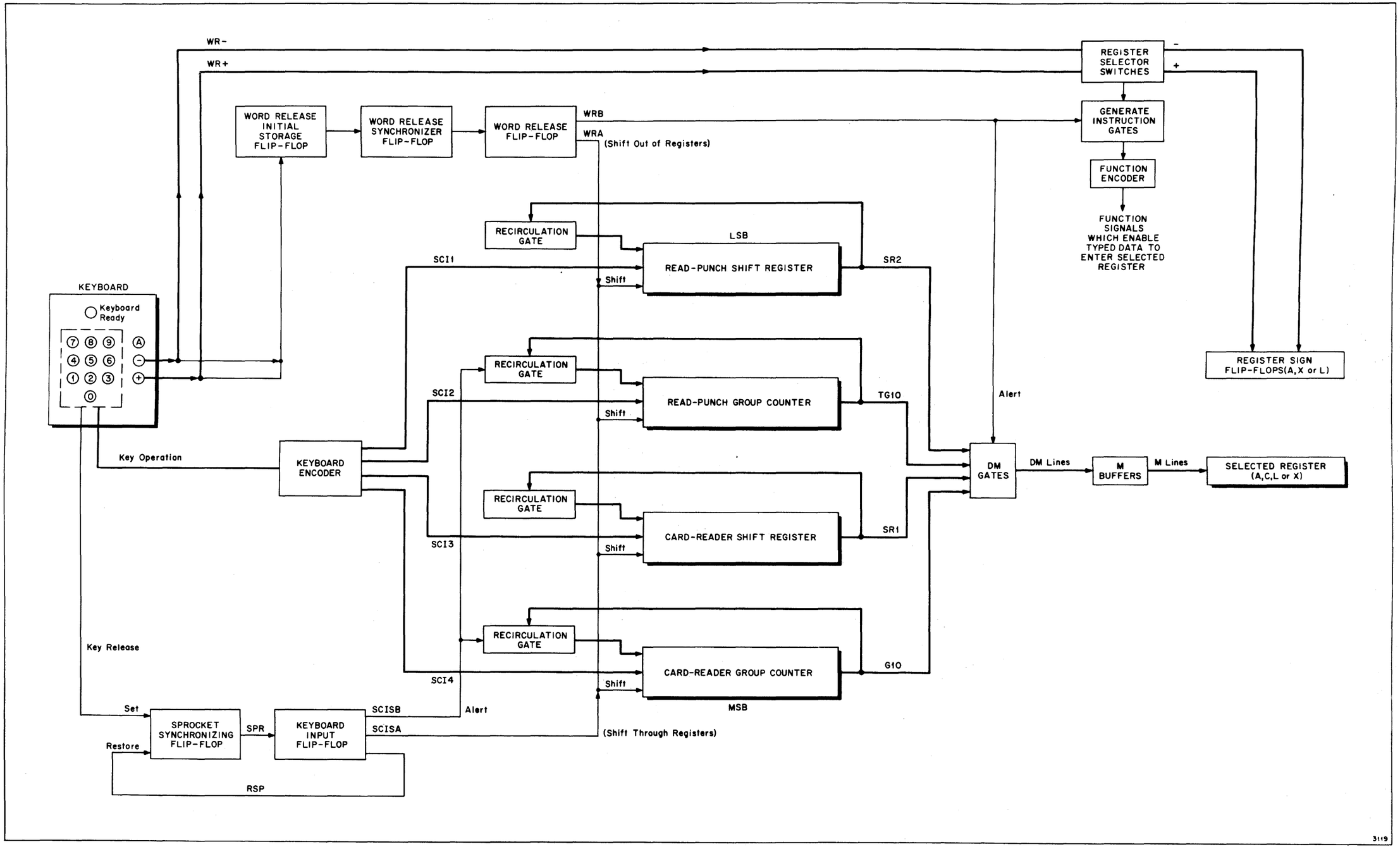


Figure 4-16. Keyboard Input Operation

One of these four signals is sent to the instruction decoder to generate function signals that perform the instruction required to store the input data in the selected register. (Function signal 67, however, is not generated because the static register is not involved.) The WRB signal also is sent to gate 15 on the static register to generate an ending pulse if FS 2 is present.

4-252. The WRA signal is sent to all four shift registers to place a high signal on the restore gates and a low on the set gates to shift the contents out of the registers. The WRA signal also is sent to the keyboard input flip-flop to generate an RSP signal which restores the word-release initial-storage and word-release synchronizing flip-flops.

4-253. The signal generated when an arithmetic register is selected (RSA, RSX, or RSL) is combined with either the WR+ or WR- signal to generate one of the following signals: WRA-, WRA+, WRX-, WRX+, WRL-, WRL+. These signals go directly to the rA, rX, or rL sign flip-flops to store the correct sign in the correct flip-flops. If rC is selected to receive the word typed in, no sign is involved.

SECTION V

TABLE OF CONTENTS

ILLUSTRATIONS

TABLE

SECTION V

INSTRUCTIONS

5-1. INTRODUCTION

5-2. There are 32 instructions in the UCT repertoire. Eight are input-output instructions described in other manuals. The remaining instructions, concerned with operations internal to the processor, are described in this section and in section 4 of this manual. Appendix C lists all the instructions numerically.

5-3. OUTLINE OF SECTION

5-4. The descriptions of instructions in this section are arranged as follows:

(1) Input-output instructions

(2) Arithmetic instructions

(3) Transfer instructions

(4) Translate instructions

(5) Miscellaneous instructions

(6) Comparison instructions

(7) Test instructions

5-5. OTHER REFERENCES

5-6. SECTION IV. Section IV of this manual, Theory of Operations, describes in detail the search and staticize instruction steps that are common to all instructions. Section 4 also explains the execution steps of the 60(H) instruction, transfer (rA) to m, and the arithmetic instructions: 70(A), add; 75(S), subtract; 85(M), multiply; and 55(D), divide. A thorough knowledge of the five instructions and the basic search and staticize sequence should provide the reader with the necessary background for understanding the less detailed descriptions in this section. The execution sequence for the remaining six transfer instructions and the translation, comparison, miscellaneous, and input-output test instructions are described in this section at a block diagram level. The

reader may assume that the initial search and staticize instruction steps have been completed and that both the text and the figures in this section refer to the execution sequences.

5-7. ANALYSIS OF UCT INSTRUCTIONS. To trace in detail the execution of the instructions described in this section, refer to Analysis of UCT Instructions, a manual which describes briefly the steps in each instruction and the effect of the control signals generated by the instructions.

5-8. SECTION III. Section III of this manual, Description of Processor Components, gives a functional description of the logical circuits that are mentioned both in this section and in the manual Analysis of UCT Instructions.

5-9. NOTATION

5-10. Instructions are designated in this manual by both an alphabetic notation, used on the logical drawings, and a numeric notation, used chiefly in programming; for example, instructions are designated: 70(A), 85(M).

5-11. DESCRIPTION OF INSTRUCTIONS

5-12. INPUT-OUTPUT INSTRUCTIONS

5-13. Input-output instructions are used to control the transfer of information to and from the input-output devices, as well as to control the movement of cards, the selection of output stackers, and so forth. Because a thorough understanding of these instructions requires knowledge of the input-output devices, each input-output instruction is described in detail in the separate manual for the device with which it is associated, as follows:

Read-Punch Unit Manual

46(W11) instruction
57(Z1) instruction
81(WZ1) instruction

Card Reader Manual

47(Z2) instruction
72(CC) instruction
96(W31) instruction

11(PR1) instruction
16(PF1) instruction

For convenience, these instructions are listed numerically in appendix C of this manual.

5-14. ARITHMETIC INSTRUCTIONS

5-15. The four arithmetic instructions, 70(A), 75(S), 85(M), and 55(D), are described in detail in section 4 of this manual.

5-16. TRANSFER INSTRUCTIONS

5-17. There are seven UCT transfer instructions that are internal to the processor. They are the 60(H), 65(X), 50(J), 25(B), 05(Y), 30(L), and 77(K) instructions.

5-18. THE 60(H), 65(X), AND 50(J) INSTRUCTIONS. These instructions transfer information from a register in the processor to a main memory location. The 60(H) instruction, which transfers the contents of register A to a main memory location, is described in detail in section 4. The 65(X) and 50(J) instructions are similar to the 60(H) instruction except that they transfer the contents of register X and register L, respectively, to main memory locations.

5-19. THE 25(B), 05(Y), AND 30(L) INSTRUCTIONS. These instructions transfer information from a main memory location to a register in the processor. The 25(B) instruction, which transfers the contents of a main memory location to register A, is described in paragraph 5-20. The 05(Y) and 30(L) instructions are similar to the 25(B) instruction except that they transfer the contents of a main memory location to register X and register L, respectively.

5-20. During the execution step of the 25(B) order (figure 5-1), the recirculation gates of register A are blocked so that the new word from memory can enter the register. The word from memory is read by the read amplifiers, goes through the M buffers, and enters the register-A input gates. A blocking signal into the S buffers forces the S lines low so that the input gates are permissive to information on the M lines only. The sign of the word from memory enters the rA sign flip-flop as shown in figure 5-1. When the entire word from memory is in the register, the blocking signal on the recirculation gates is removed so that the word can recirculate in the register.

5-21. THE 77(K) INSTRUCTION. The 77(K) instruction transfers the contents of register A to register L and is the only register-to-register transfer instruction. Because this instruction does not involve a search for an operand, the m-address portion of the instruction is ignored.

5-22. TRANSLATE INSTRUCTIONS

5-23. There are two translate instructions. The 12(G) instruction translates one word from card code to UCT code, and the 17(R) instruction translates one word from UCT code to card code. The input-output devices operate with card code, while the processor performs arithmetic operations only with UCT code. See section I, Introduction, for details about the codes.

5-24. THE 12(G) INSTRUCTION. The 12(G) instruction translates the contents of registers A and X from card code to UCT code. If it is desired to perform arithmetic operations on information read by the card reader or read-punch device, the 12(G) instruction must first be given in order to translate the information from the card code used by the input-output devices to the UCT code used by the processor.

5-25. Information read in card code from punched cards by the card read or read-punch device is eventually stored in main memory. Each word of information in the 6-bit card code must be divided into two parts, called the primed and unprimed parts, in order to store it on the memory drum, which normally stores the 4-bit UCT code. The unprimed part is stored in one location which normally stores a complete UCT-coded word, and the primed part is stored in another location. The division of each word in card code into primed and unprimed parts is accomplished automatically when the programmer transfers the information from the card-buffer band to main memory. This operation is described in detail in the Read-Punch Manual and in the Card Reader Manual.

5-26. In order to translate a word from card code to UCT code, the programmer gives the appropriate transfer instructions to transfer the unprimed part of the word to be translated from its main memory location to register A and the primed part from its main memory location to register X. When a 12(G) instruction is given, the unprimed part of the word in register A is sent to the S buffer and then to the translation gates of rA; this information does not recirculate in rA because the recirculation gates are blocked. At the same time, the primed part of the word

in rX is sent to the M buffers and then to the translation gates of rA. The recirculation gates of rX are also blocked. See figure 5-2.

5-27. After the complete card-coded word is translated into UCT code, the blocking signal on the recirculation gates of rA is removed to allow the newly formed UCT word to recirculate in rA. Register X remains cleared to all zeros. The sign of the UCT word in rA after translation is the sign of the unprimed part of the card-coded word that was in rA before translation.

5-28. THE 17(R) INSTRUCTION. The 17(R) instruction translates the contents of register A from UCT code to card code. The programmer places the UCT-coded word which is to be translated in register A. When a 17(R) instruction is given, the word is sent to the S buffers and then to the translation gates of registers A and X (figure 5-3), but it does not recirculate in register A because the recirculation gates are blocked. The unprimed part of the translated word goes to register A; the primed part of the translated word goes to register X. After the complete word is translated, the blocking signals on the recirculation gates of both registers are removed so that the unprimed and primed parts of the word can circulate in rA and rX respectively. The programmer then transfers the primed and unprimed parts of the word to main memory by giving the appropriate transfer instructions. During the translation process, both the rA and rX sign flip-flops are forced to a plus condition.

5-29. MISCELLANEOUS INSTRUCTIONS

5-30. Six instructions have been grouped in the miscellaneous category. These are: the 20(P) superimpose instruction; the 35(E) extract instruction; the 32(N) shift-right instruction; the 37(V) shift-left instruction; the 62(ZS) zero-suppress instruction; and the 67(STOP) instruction.

5-31. THE 20(P) SUPERIMPOSE INSTRUCTION. The 20(P) instruction superimposes the 1-bits of the word in the m-address location on the word in register A. The result is a new word in register A containing a 1-bit where either the original word in rA or the word from memory had a 1-bit. The sign of the original word in rA is not changed.

5-32. This instruction is used when it is desired to combine the information in one word with the information in another word so that the new word contains the information of both. Often the information in one word must be combined with many other words, so that the information to be superimposed is placed in memory as a constant.

5-33. As shown in the block diagram of the 20(P) instruction (figure 5-4), the word of information in rA is allowed to circulate in register A through the recirculation gates. At the same time, the word from memory enters rA on the M lines from the M buffers through the input gates of rA. The S buffers are blocked so that the rA input gates are permissive to information on the M lines only. Thus, the 1-bits in register A are allowed to recirculate into the register through the recirculation gates and the 1-bits in the word from memory are allowed to enter the register through the input gates.

5-34. THE 35(E) EXTRACT INSTRUCTION. The 35(E) instruction changes 1-bits to binary zeros in each decimal digit of the word in register A whenever there is a binary zero in the corresponding bit position of the word in the main memory location designated by the m address. The sign of rA is not changed.

5-35. This instruction is referred to as the extract instruction because it is used to extract certain digits from a word in rA. In the extract word placed in memory by the programmer, each bit of each digit contains a 1-bit where the corresponding digit in rA is to be unchanged, and a 0-bit where it is to be changed to zero.

5-36. For example, to extract the five most significant digits of the word in rA by changing the five least significant digits to zeros, the programmer places the extract word .....00000 in the memory. When an extract instruction that contains this extract word as the m address is given, the 0-bits in each bit position of the five least significant digit positions change these digits to zeros in the word in rA. Since the UCT code for a period is 1111, or four 1-bits, there are no zero bits in the five most significant digit positions of the extract word, and the corresponding digits in the original word in rA remain unchanged.

5-37. As figure 5-5 shows, this instruction is similar to the 20(P) instruction. In the 35(E) instruction, however, the word in rA does not recirculate because the recirculation gates are blocked. Instead, it is sent to the S buffer, so that it is on the S lines at the same time that the extract word from memory is on the M lines. Corresponding digits of both words, therefore, are at the input gates of register A at the same time. (See section 3-34.) The new word formed in rA as a result of this instruction contains 1-bits only in the bit positions where the S and M lines had 1-bits in corresponding bit positions.

5-38. THE 32(N) SHIFT-RIGHT INSTRUCTION. The 32(N) instruction shifts the contents of register A to the right a designated number of places into rX and at the same time shifts the contents of rX to the right into register A. The number of places shifted can vary from zero through ten. A single digit in the p7 digit position of the m address of the instruction word indicates the number of places to be shifted. (The p7 position is the next to most significant digit of the m address.) The bit-code combination 1101 is used to indicate 10 in this instruction.

5-39. During the staticize step of every instruction, the p7 digit of the m address is placed in the multiplier-quotient counter (MQC), so that it is available if the instruction is a shift instruction.

5-40. There are two execution steps to the 32(N) instruction: N1 and N2. During the N1 step (figure 5-6), the digit in MQC (the p7 digit of the m address) is examined. If the digit in MQC is a zero, an ending pulse clears the static register to all zeros so that there is a search for the next instruction. If the digit in MQC is not a zero, the static register is stepped to the N2 step of the 32(N) instruction.

5-41. During the N2 step of this instruction, the contents of rA and rX are shifted to the right by blocking the normal recirculation gates of rA and rX, and circulating the contents of rA and rX through a shortened loop. During the N2 stage the contents of rA and rX are circulated through the shortened loop the number of times indicated by the digit in MQC. During each circulation, the contents of both rA and rX are shifted one place to the right (figure 5-6). A circular shift operation also takes place: the least significant digit (LSD) of rA becomes the most significant digit (MSD) of rX and the LSD of rX becomes the MSD of rA.

During each shift operation the count-down gates of MQC count down one until the digit in MQC is zero. A zero in MQC generates an ending pulse.

5-42. THE 37(V) SHIFT-LEFT INSTRUCTION. The 37(V) instruction shifts the contents of rA to the left a designated number of places. The most significant digits are lost and zeros are placed in the least significant digit places. The number of places shifted can vary from zero through ten. The code combination 1101 is used to indicate 10 in this instruction.

5-43. There are two steps to the 37(V) instruction: V1 and V2. During the V1 step (figure 5-7), the digit in MQC is examined. The digit in MQC is the p7 digit of the m address, placed in MQC during every staticize instruction step. If the digit in MQC is a zero, an ending pulse clears the static register so that there is a search for the next instruction. If the digit in MQC is not a zero, the static register is stepped to the V2 step of the 37(V) instruction.

5-44. During the V2 step, the normal recirculation gates of rA are blocked and the contents of rA is shifted to the left by circulating it through a lengthened loop. The contents of rA is delayed one pulse time in the loop before it reaches the left shift gates, so that the entire contents of rA shifts one place to the left each time it circulates. The binary bit of each digit is delayed by adding one pulse time in subregister 4 of rA. The quinary bits are delayed one pulse time by sending them through the S buffers and the complementing circuit, and then back into rA. (The bits are only delayed in the complementing circuit, not complemented.) As a result of each left shift of one digit position, the most significant digit is lost and a zero appears in the least significant digit position. During each shift operation the count-down gates of MQC count down one until the digit in MQC is zero. A zero in MQC generates an ending pulse.

5-45. THE 62(ZS) ZERO-SUPPRESS INSTRUCTION. The 62(ZS) instruction suppresses all the card-coded punching or printing zeros and commas (and only zeros and commas) to the left of the most significant digit and replaces them with non-punching or non-printing zeros. The m portion of the instruction word is ignored because there is no search for an operand in this instruction. Suppressing means that a code combination which would cause a zero

or comma to be punched on a card or printed by the printer is changed so that the code combination causes a space on the card or the printed paper.

5-46. All output information which is to be printed by the printer or punched by the read-punch device must be in card code. When information is translated from UCT code to card code before going to the input-output devices, all UCT-coded zeros are translated into card-coded punching or printing zeros. The card code for a punching or printing zero is 000001. (See paragraph 1-37 for explanation of the card code.) The combination 000001 causes a hole to be punched in the zero row of the card, or causes the printer to print a zero. The zero suppress instruction changes the binary one of the combination 000001 to zero, creating the non-printing and non-punching combination 000000.

5-47. In addition to suppressing all zeros to the left of the most significant digit, the 62 (ZS) instruction suppresses all commas to the left of the most significant digit. The instruction changes 000011, the code combination for a comma, to 000000. Since a result may have a large number of digits, commas are often programmed into words. For example, 75875875 is printed or punched as 75,875,875. However, if the number 00,000,875 is to be printed or punched, a zero suppress instruction removes the five zeros and two commas to the left of the most significant digit, and 875 is printed or punched.

5-48. During a zero suppress instruction, the unprimed part of the word to be suppressed must be in register A. Since each digit of the unprimed part consists of the four lower order bits, the unprimed part of the word fills register A. However, each digit of the primed part consists only of the two higher order bits and these occupy subregisters 1 and 2 of rX. Because subregister 3 is not used to circulate information, it is used temporarily as a special storage device during the execution of the zero suppress instruction. Register X, therefore, should contain only the primed part of the word.

5-49. Figure 5-8 shows the two steps of the zero-suppress instruction: ZS1 and ZS2. During ZS1, the unprimed part of the word in rA is sent through the zero-suppress and comma-suppress gates of rX. The gates, which are connected only to subregister 3 of rX, place a binary zero in subregister 3 wherever there is a punching or printing zero (000001) or comma (000011) in the corresponding digit position of rA. They place a binary one in subregister 3

for every character other than a zero or a comma. When the entire contents of rA has been examined, the static register is stepped to ZS2.

5-50. During the ZS2 step, the word in rA recirculates through the normal recirculation gates. Subregister 3 of rX is designed so that the outputs of each of the ten bits in the subregister are available in parallel. These ten outputs are inputs to the zero-suppress gate of rA. The output of the rA zero-suppress gate is an input to the normal recirculation gates of rA. As long as a binary one in subregister 3 of rX indicates that there are still significant digits to enter rA, the zero-suppress gate of rA is blocked so that the output of this gate keeps the normal recirculation of rA permissive to the recirculating digits. A blocking signal on the normal recirculation qate of subregister 3 of rX prevents the bits from reentering the subregister. When there are no longer any binary ones in the subregister, the zero-suppress gate of rA is made permissive and its output blocks the normal recirculation gates of rA so that all the 1-bits of the remaining digits of the word in rA (which are zeros or commas) are changed to 0-bits. That is, all remaining zeros and commas are suppressed.

5-51. THE 67(STOP) INSTRUCTION. The 67(STOP) instruction stops the processor. When this instruction is in the static register, it generates a signal which sets the stop flip-flop and clears the static register to zeros. The stop flip-flop generates a signal which is sent to the instruction decoder and prevents the generation of any function signals.

5-52. With the static register cleared to zeros and no function signals being generated, the 67 instruction word (which includes the m and c address) continues to circulate in register C. The processor is started again by pressing the RUN button, one of the two COMPUTATION buttons on the control panel. The next instruction is at the m address if the NEXT ADDRESS button for the m address is lit; at the c address if that NEXT ADDRESS button is lit. The operator may select the next address by pressing either NEXT ADDRESS button, however. The buttons control the conditional-transfer (CT) flip-flop.

5-53. COMPARISON INSTRUCTIONS

5-54. There are two comparison instructions which compare the contents of register A and register L and, as a result of the comparison, direct the processor either to the m address or to the c address. The 82(Q) instruction compares for the equality or inequality of the contents of registers A and L. The 87(T) instruction compares for greater-than and less-than conditions.

5-55. The processor can be stopped after the execution of an 82(Q) or an 87(T) instruction by pressing the COMPARISON STOP button on the control panel. When this button is pressed, one of the NEXT ADDRESS pushbuttons lights to indicate whether the next instruction is to be taken from the main memory location indicated by the m address or the c address. The operator can change the address determined by the comparison, by pushing the other NEXT ADDRESS button. The comparison stop operation is explained in detail in paragraph 4.5.5.

5-56. THE 82(Q) INSTRUCTION. The 82(Q) instruction compares the contents of register A with the contents of register L. If (rA) and (rL) are algebraically equal, the next instruction is in the memory location designated by the m address. If (rA) and (rL) are not algebraically equal, the next instruction is in the memory location designated by the c address.

5-57. Figure 5-9 is a block diagram of the 82(Q) instruction. The state of the conditional-transfer (CT) flip-flop determines which address is read from the memory. If the CT flip-flop is set, the m address is read from memory; if the CT flip-flop is restored, the c address is read from memory. At the beginning of the instruction, the CT flip-flop is set. If (rA) and (rL) are equal, the CT flip-flop remains set at the end of the comparison and the m address is read from memory. If, however, (rA) and (rL) are not equal, the CT flip-flop is restored and the c address is read from memory.

5-58. As shown in figure 5-9, (rA) and (rL) are sent to the binary and quinary equality gates on the M and S lines, (rA) on the S lines and (rL) on the M lines. The binary equality gates sample for the equality of the binary bits of each digit on the M and S lines, while the quinary equality gates sample for equality of the quinary bits of each digit on the M and S lines. See paragraphs 3-82 and 3-95 for a detailed explanation of these circuits.

5-59. In addition to the M and S lines, the binary and quinary equality circuits of the comparator have, as an input, the CP signal from the complement flip-flop. The complement flip-flop is set at the beginning of the comparison operation so that the CP signal alerts the binary and quinary equality gates during the word time that (rA) and (rL) are compared.

5-60. The quinary equality gates also have the A and C signals as inputs. When the complement flip-flop is set at the beginning of the comparison operations, it generates a CP5 signal for one pulse time. The CP5 signal is sent to the initial force-decimal-carry circuit (paragraph 3-99), where it initially generates the A and C signals so that the quinary equality gates are permissive to the bits of the first digit to be compared for equality on the M and S lines. For example, suppose the words being compared are as follows:

(rL) on M Lines 0000006917

(rA) on S Lines 0000006917

The initial generation of the A and C signals by the initial force-decimal-carry circuit makes the quinary equality gates permissive to the two sevens, the first two digits to be compared. Because they are equal, the comparison of the two seven digits results in the generation of the A and C signals again. Because all succeeding digits are equal, the A and C signals continue to be generated and are present at the end of the comparison.

5-61. There are four gates which can restore the CT flip-flop during the 82(Q) instruction. (See Analysis of UCT Instructions for gate numbers.) The outputs of the binary and quinary equality circuits (the $\overline{EQ}$ and $\overline{A^{\prime}}$ signals) are sent to two of the four restore gates, to restore the CT flip-flop if either the binary or the quinary part is not equal. During this instruction, the other twc restore gates of the CT flip-flop have as inputs the outputs of the rA and rL sign flip-flops, to restore the CT flip-flop if the signs of (rL) and (rA) are not the same.

5-62. THE 87(T) INSTRUCTION. The 87(T) instruction compares the contents of registers A and L. If the contents of register A is algebraically greater than the contents of register L, the next instruction is in the memory location designated by the m address. If (rA) is algebraically less than or equal to (rL), the next instruction is in the memory location designated by the c address.

5-63. Figure 5-10 is a block diagram of the 87(T) instruction. The condition of the CT flip-flop determines which address is read from memory. If the CT flip-flop is set, the m address is read from memory; if the CT flip-flop is restored, the c address is read from memory. The CT flip-flop is set at the beginning of the comparison operation. If (rA) is greater than (rL), the CT flip-flop remains set at the end of the comparison and the memory location designated by the m address is read. If, however, (rA) is less than or equal to (rL), the CT flip-flop is restored and the memory location designated by the c address is read from memory.

5-64. The contents of rA and rL are compared during the 87(T) instruction in the quinary equality gates (paragraph 3-82), the binary equality gates (paragraph 3-95), the force decimal carry gates (paragraph 3-91) and part of the quinary carry circuit (paragraph 3-86). The M and S lines are inputs to all of these circuits. During the T instruction, the contents of rA is sent to the circuits on the S lines and the contents of rL is sent on the M lines. Several of the circuits have the CP signal as an input. Since the CP flip-flop is set at the beginning of the comparison operation, the circuits which have CP as an input are permissive to information on the M and S lines.

5-65. The binary and quinary equality gates sample for the equality of digits on the M and S lines in this instruction as they do in the 82(Q) instruction. Gate 31 of the force-decimal-carry circuit (figure A12) samples for whether the binary bit on the M line is greater than the binary bit on the S line. The part of the quinary circuit consisting of gates 33, 34, and 46 through 56 samples for whether the digit on the M lines is greater than the digit on the S lines.

5-66. Table 5-1 shows all conditions for comparing (rA) and (rL). Whether the CT flip-flop remains set or is restored depends upon whether the A and C signals are present at the completion of the comparison. The restore gates of the CT flip-flop during the 87(T) instruction have as inputs the A and C signals (gate 13) or the $\overline{C}$ signal (gate 14) or the $\overline{A}$ signal (gate 15). (See Analysis of UCT Instructions.) Paragraph 3-107 explains which gate operates to restore the CT flip-flop under each of the conditions in table 5-1.

5-67. As indicated in table 5-1, gate 32 of figure A12 initially generates the decimal-carry signals (A and C) if the sign of (rA) is plus. If (rA) is minus the A and C signals are not initially generated. The initial generation of the A and C signals when (rA) is plus but not when (rA) is minus affects the conditions under which the CT flip-flop remains set or is restored.

5-68. As shown in table 5-1, there are two conditions under which the CT flip-flop is restored when rA is plus: when (rA) equals (rL), and when (rA) is less than (rL). If (rA) equals (rL), the binary and quinary equality gates continue to generate the A and C signals, which restore the CT flip-flop at the end of the comparison.

5-69. If (rA) is less than (rL), the A and C signals are again present at the end of the comparison, but they are generated by the gates which indicate (rA) is less than (rL) (paragraph 3-86). For example, suppose the two words being compared are as follows:

(rL) on M lines 0000007809

(rA) on S lines 0000007431

When the nine and the one are compared, A and C signals are generated because rA is less than rL. When the next digits, the zero and the three, are compared, the A and C signals are lost because rA is greater than rL and there are no gates to generate A and C signals under this condition. When the two most significant non-equal digits, the 8 and 4, are compared, A and C signals are again generated by the gates of the quinary-carry circuit

which indicate rA is less than rL. The A and C signals thus generated make the quinary equality gates permissive so that the succeeding digits, which are all equal, continue to generate the A and C signals, which restore the CT flip-flop at the end of the comparison.

5-70. Table 5-1 shows that when (rA) is minus, there is only one condition for which the CT flip-flop remains set: when both (rA) and (rL) are minus, and (rA) is less than (rL). When (rA) is less than (rL), the gates which indicate the contents of rA is less than the contents of rL generate the A and C signals and the CT flip-flop is not restored. For example, suppose the two words being compared are as follows:

(rL) on M lines 0000005924

(rA) on S lines 0000005384

The A and C signals are not generated until the two most significant non-equal digits, the nine and the three, are compared, when the gates which indicate that rA is less than rL generate the signals. Because the A and C signals are not initially generated, the equality gates are blocked and the A and C signals are not generated when the two fours are compared. They are not generated when the eight and the two are compared, because there are no gates which indicate (rA) is greater than (rL). However, when the nine and the three generate the A and C signals, the biquinary equality gates are made permissive and the A and C signals continue to be generated because all remaining digits of this example are equal.

5-71. TEST INSTRUCTIONS

5-72. Three input-output test instructions are provided in the UCT, one for each of the three input-output devices. The programmer uses these instructions to determine whether a particular input-output device is available or whether it is still processing a previous instruction. For example, the input-output test for the printer tests to determine if the printer is free to accept a print order or is still processing a previous print order. If the printer is not available to process a print order, the processor is free to continue the main program, instead of waiting until the printer is available. This is a saving in computing time.

The instructions are distributed throughout the program to test the availability of the input-output devices at convenient intervals.

5-73. Each of the three input-output instructions has two stages. The first stage tests whether the device is available. If it is not available, an ending pulse is generated and the processor goes to the c address, the address of the next instruction. Because each of the three instructions tests a different device, the first stage of each instruction is different.

5-74. If the test performed during the first stage shows that the device is available, the static register is stepped to the second stage. The second stage of each test instruction is identical. (See figure 5-11.) During this stage, the contents of register C is transferred to register A, and the next instruction is read from the m address. Register C contains the c address, the address of the next instruction of the main program. It is put in register A because the processor must return to it in order to continue to process the program. The m address to which the processor is directed can contain a subroutine which transfers the c address of the test instruction from register A to memory, gives one of the input-output instructions, and returns the processor to the main program.

5-75. THE 22 (I 11) INSTRUCTION. The 22 (I 11) instruction tests to determine whether the card buffer band is loaded with information from the read-punch device. If the card buffer band is not loaded from the read-punch device, the next instruction is in the memory location designated by the c address. If the card buffer band is loaded from the read-punch unit, the next instruction is in the memory location designated by the m address, and the contents of register C is transferred to register A.

5-76. Item 1 of the instruction analysis for the 22 (I 11) instruction lists the signals in the first stage, during which the test is made to determine whether or not the buffer is loaded from the read-punch device. If the buffer band is not loaded from the read-punch device, an ending pulse is generated. If the buffer band is loaded from the read-punch device, the static register is stepped to the second

stage of the instruction (items 2 through 9 of the instruction analysis). The second stage of this instruction, which is common to all the input-output test instructions, is shown in block diagram form in figure 5-11.

5-77. THE 27 (I 21) INSTRUCTION. The 27 (I 21) instruction tests to determine whether the printer is printing or feeding paper. If the printer is printing or feeding paper, the next instruction is in the memory location designated by the c address. If the printer is not printing or feeding paper, the next instruction is in the memory location designated by the m address and the contents of register C is transferred to register A.

5-78. Item 1 of the instruction analysis of the 27 (I 21) instruction lists the signals in the first stage, during which the test is made to determine whether or not a printing or paper feeding operation is taking place. If the printer is occupied, an ending pulse is generated. If the printer is available, the static register is stepped to the second stage of the instruction (items 2 through 9 of the instruction analysis). The second stage, which is common to all the input-output test instructions, is shown in block diagram form in figure 5-11.

5-79. THE 42 (I 31) INSTRUCTION. The 42 (I 31) instruction tests to determine whether the card buffer is loaded with information read by the card reader. If the buffer band is not loaded from the card reader, the next instruction is in the memory location designated by the c address. If the card buffer band is loaded from the card reader, the next instruction is in the memory location designated by the m address and the contents of register C is transferred to register A.

5-80. Item 1 of the instruction analysis lists the signals in the first stage, during which the test is made to determine whether or not the buffer band is loaded from the card reader. If the buffer band is not loaded from the card reader, an ending pulse is generated. If the card buffer band is loaded, the static register is stepped to the second stage of the instruction (items 2 through 9 of the instruction analysis). The second stage, which is common to all the input-output instructions, is shown in block diagram form in figure 5-11.

Table 5-1. State of Conditional-Transfer Flip-Flop After Comparison Operation

When the flip-flop is set, the m address is read. When the flip-flop is restored, the c address is read.

Sign*		Absolute Value		
(rA)	(rL)	(rA) > (rL)	(rA) = (rL)	(rA) < (rL)
+	+	set	restored	restored
+	-	set	set	set
-	+	restored	restored	restored
-	-	restored	restored	set

*When the sign of (rA) is plus, gate 32 generates initial decimal-carry signals. When the sign of (rA) is minus, no initial decimal-carry signals are generated.

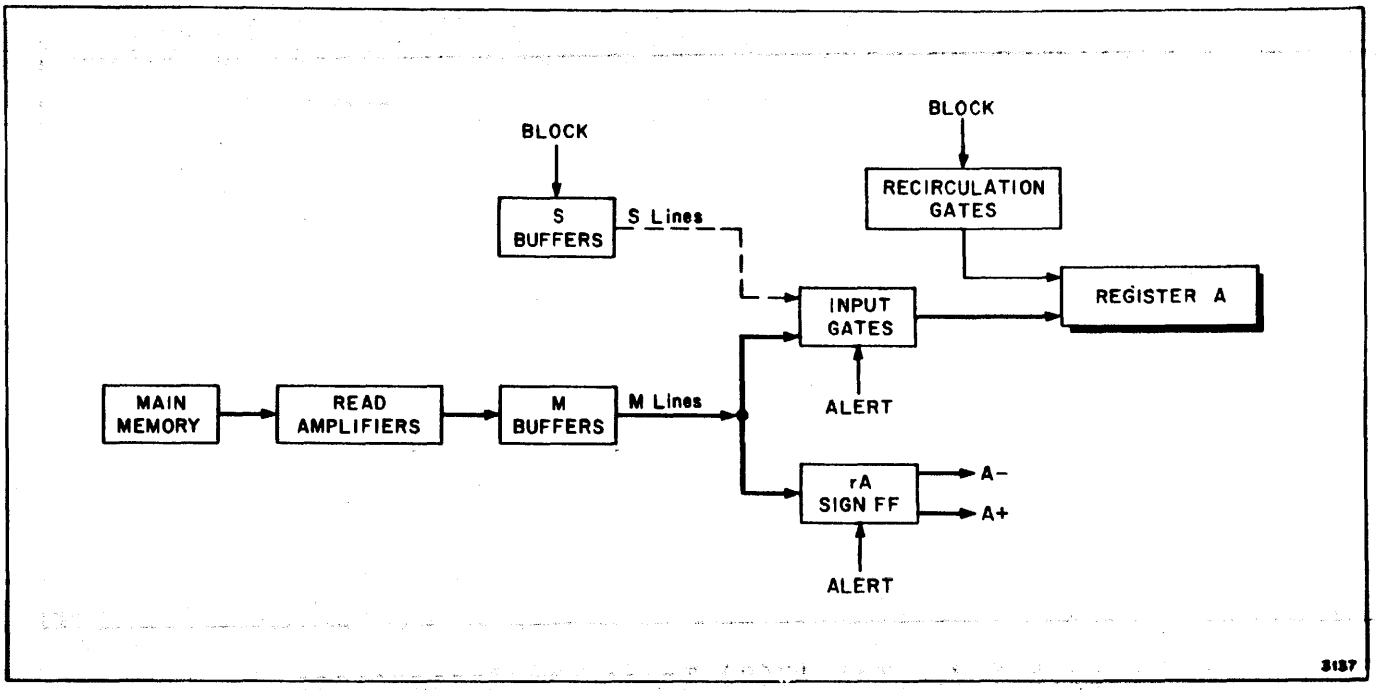


Figure 5-1. The 25(B) Transfer Instruction ((m)--->rA)

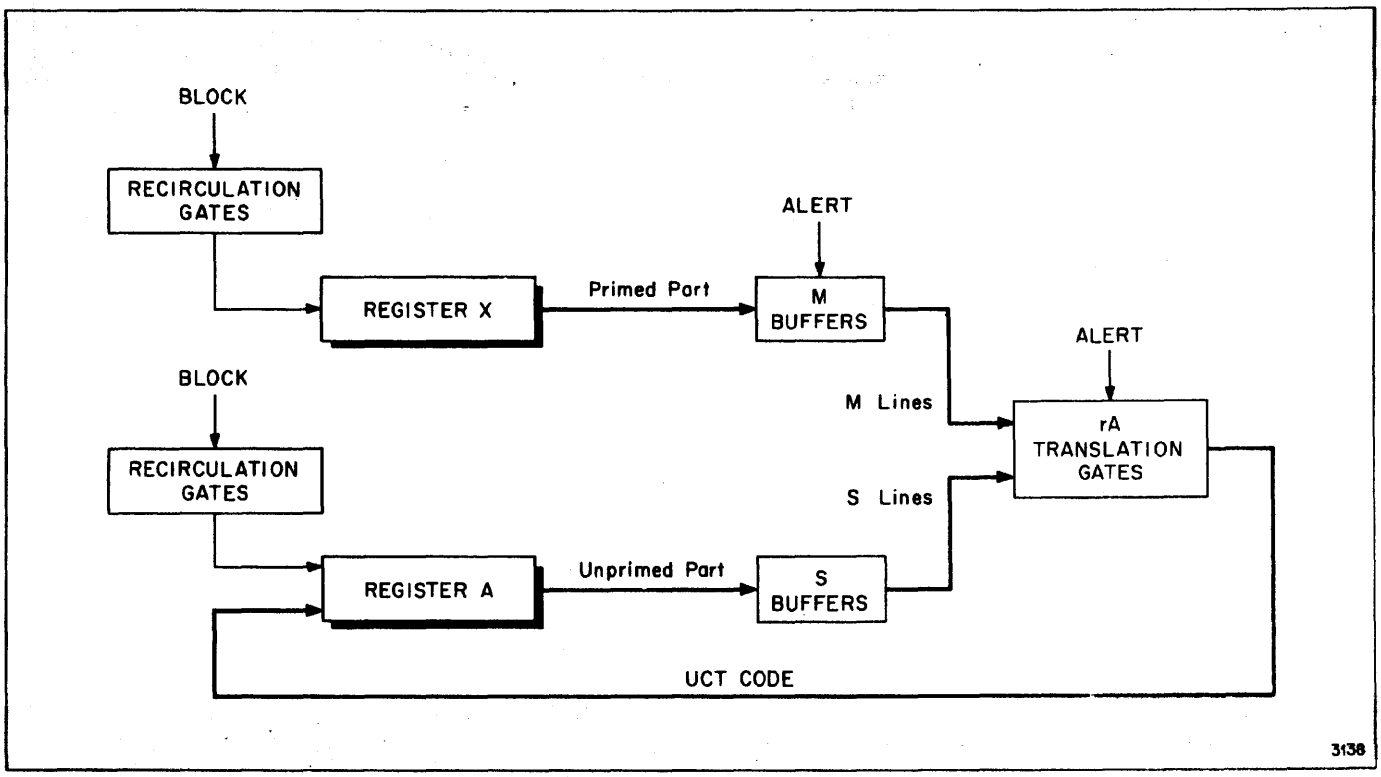


Figure 5-2. The 12(G) Translate Instruction
(Card Code to UCT Code)

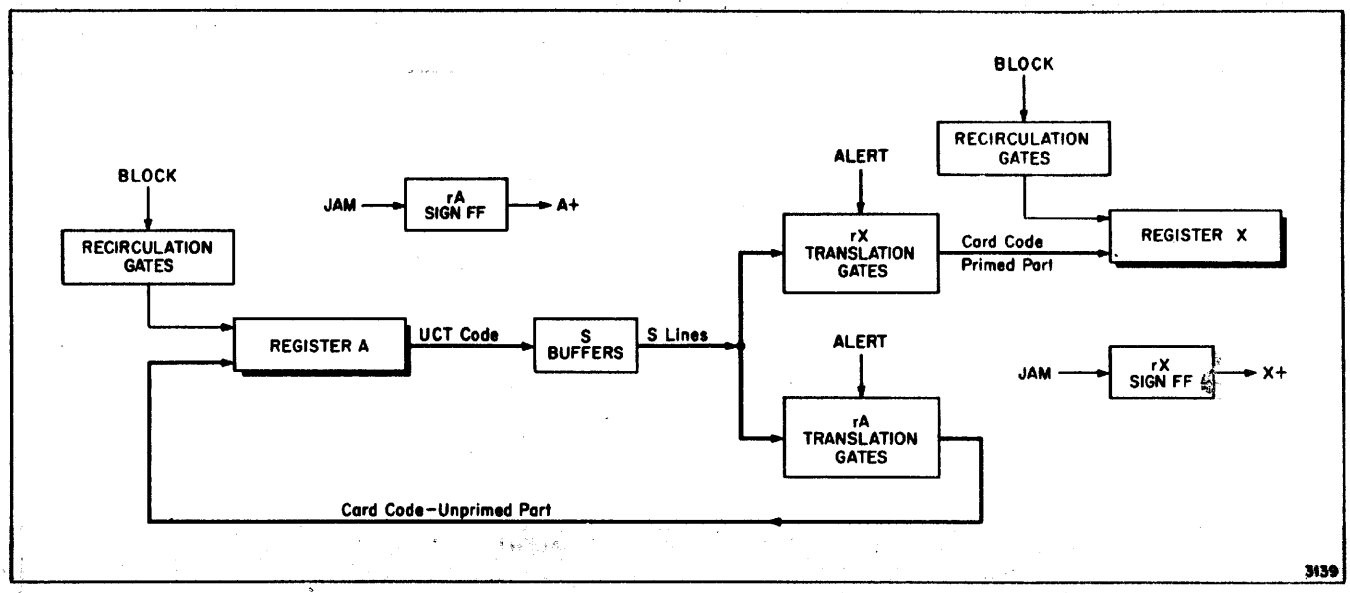


Figure 5-3. The 17(R) Translate Instruction (UCT Code to Card Code)

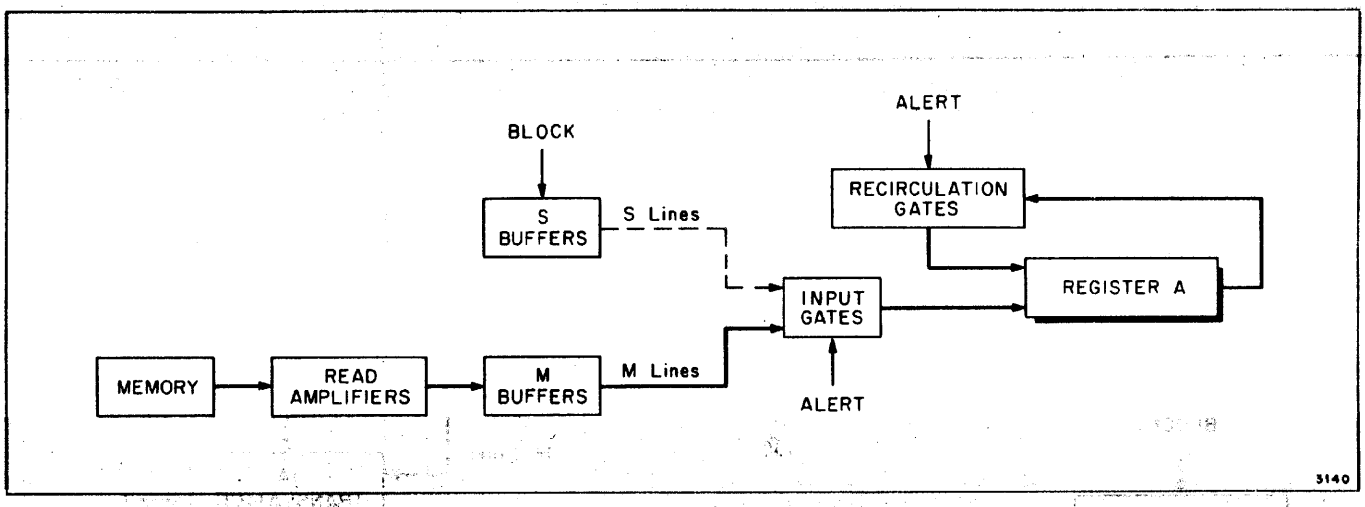


Figure 5-4. The 20(P) Superimpose Instruction

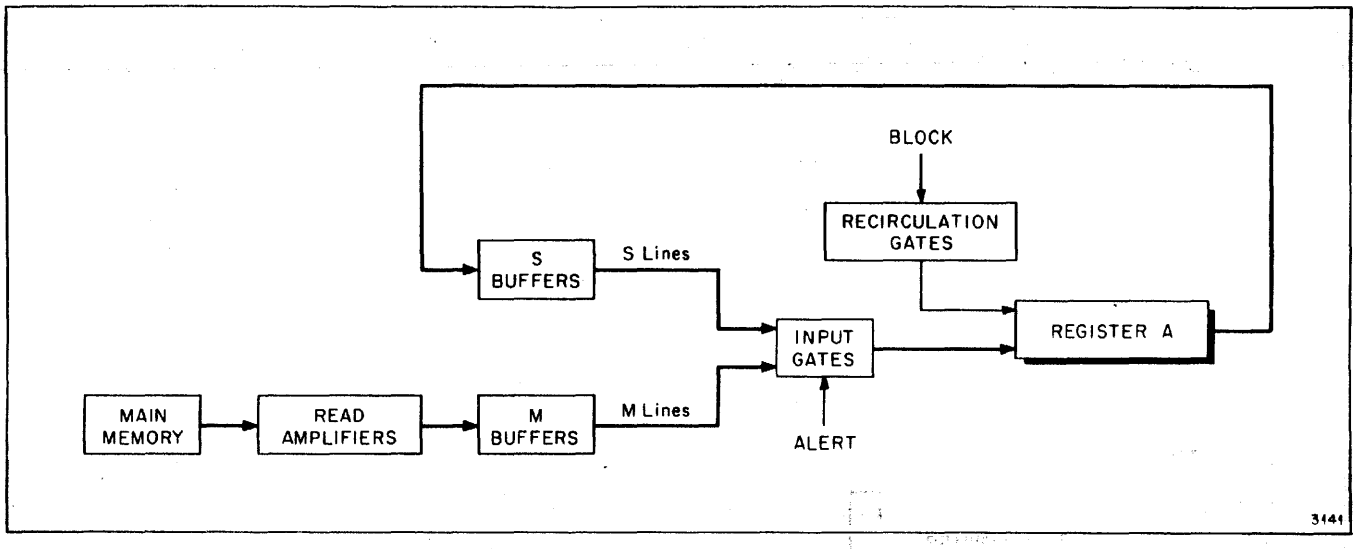


Figure 5-5. The 35(E) Extract Instruction

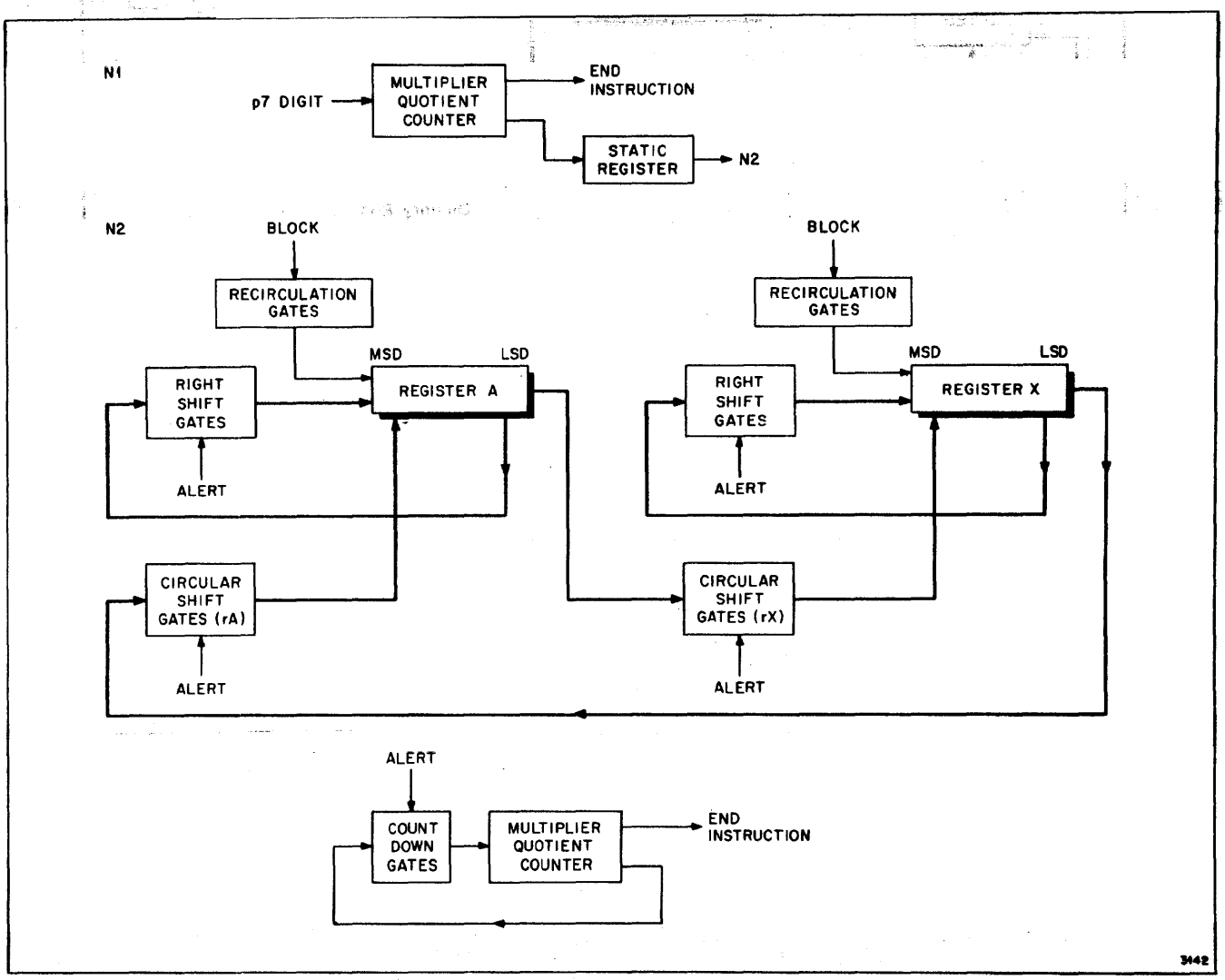


Figure 5-6. The 32(N) Right-Shift Instruction

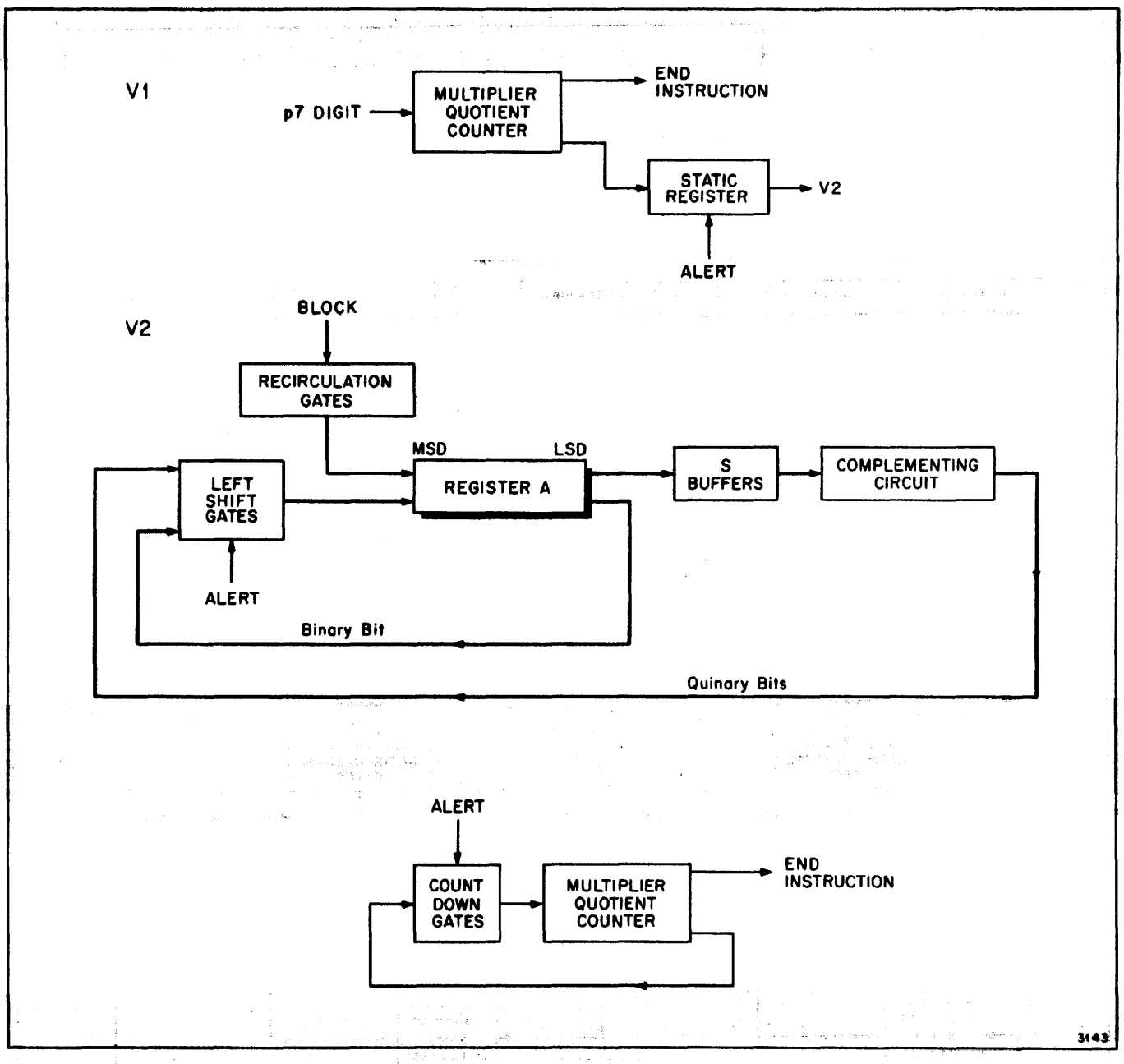


Figure 5-7. The 37(V) Left-Shift Instruction

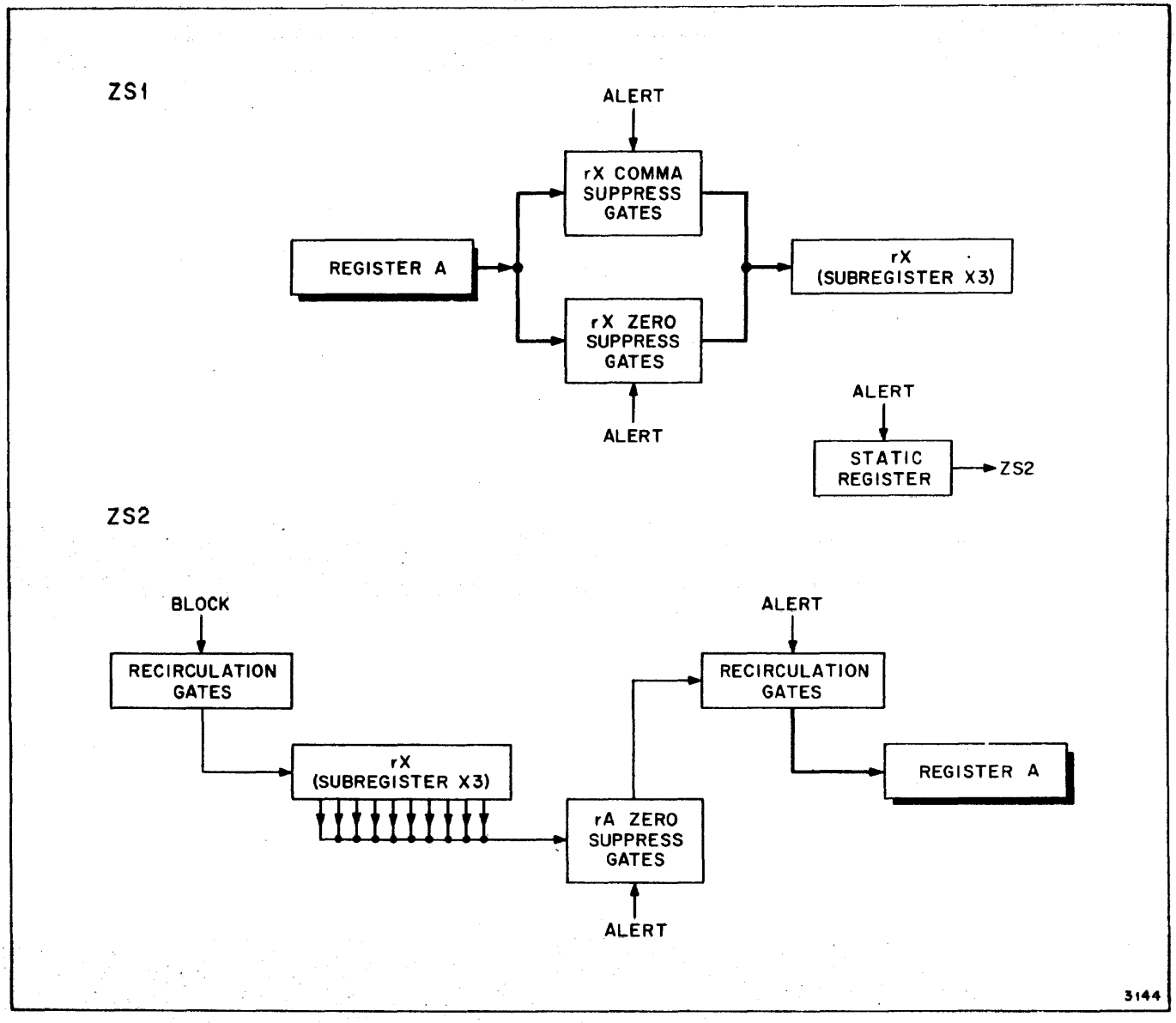


Figure 5-8. The 62(ZS1) Zero-Suppress Instruction

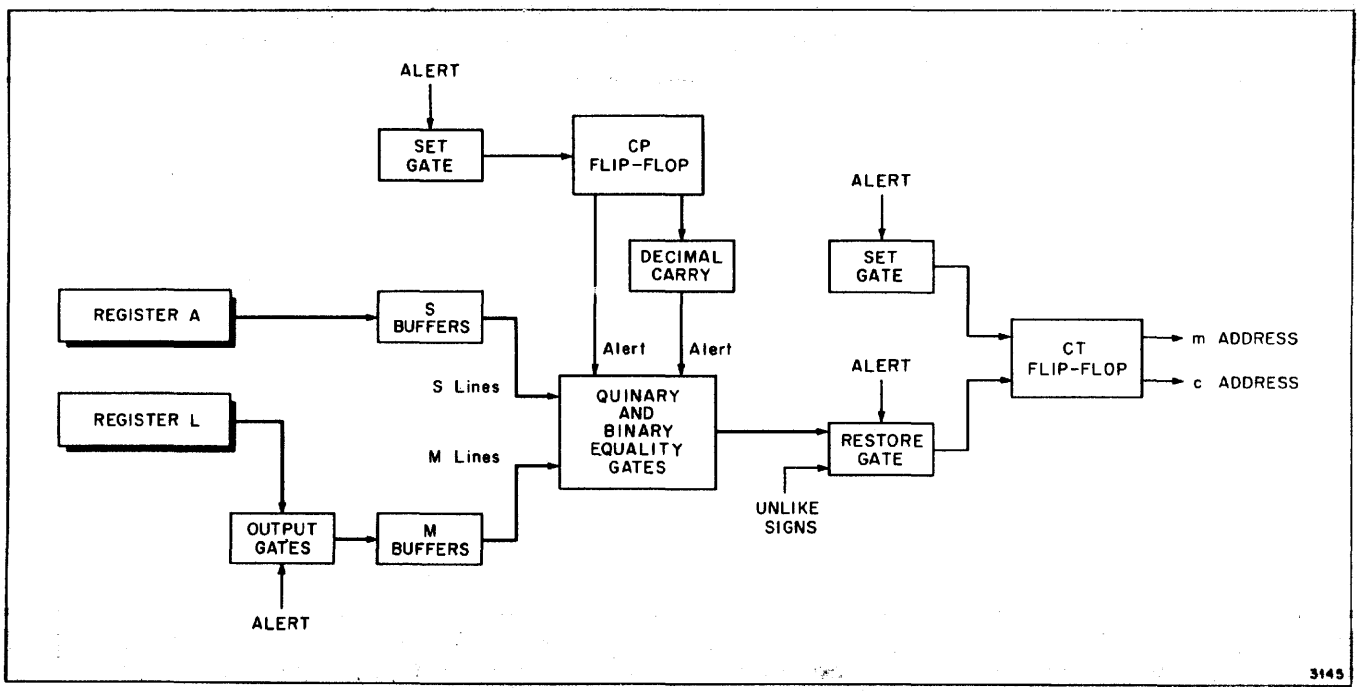


Figure 5-9. The 82(Q) Equality-Comparison Instruction

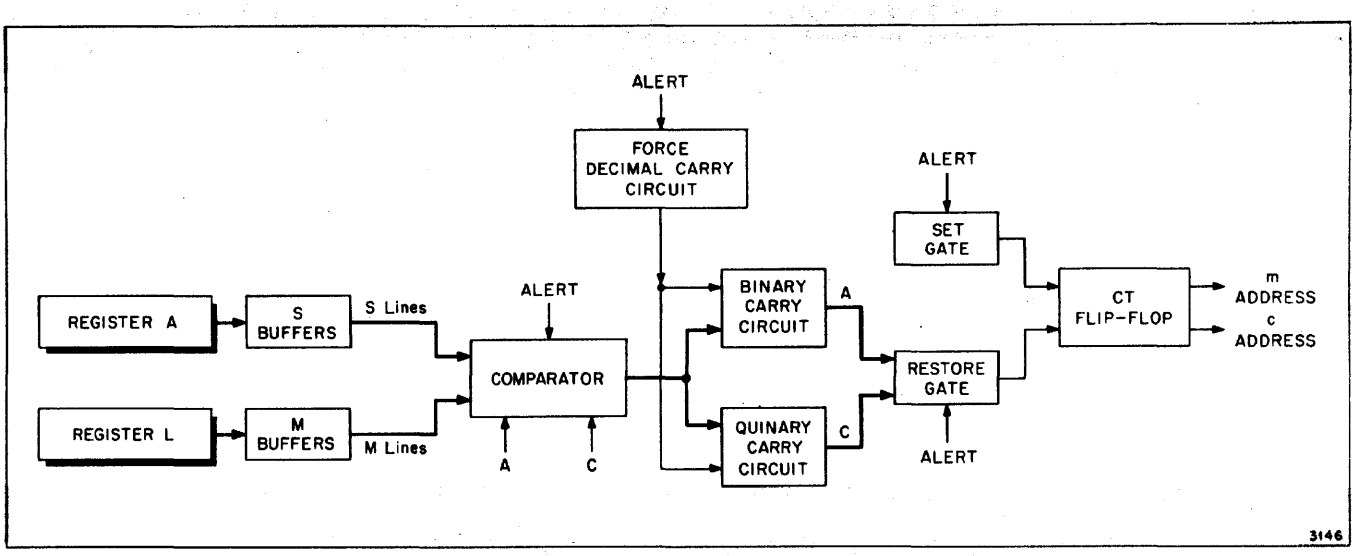


Figure 5-10. The 87(T) Comparison Instructions

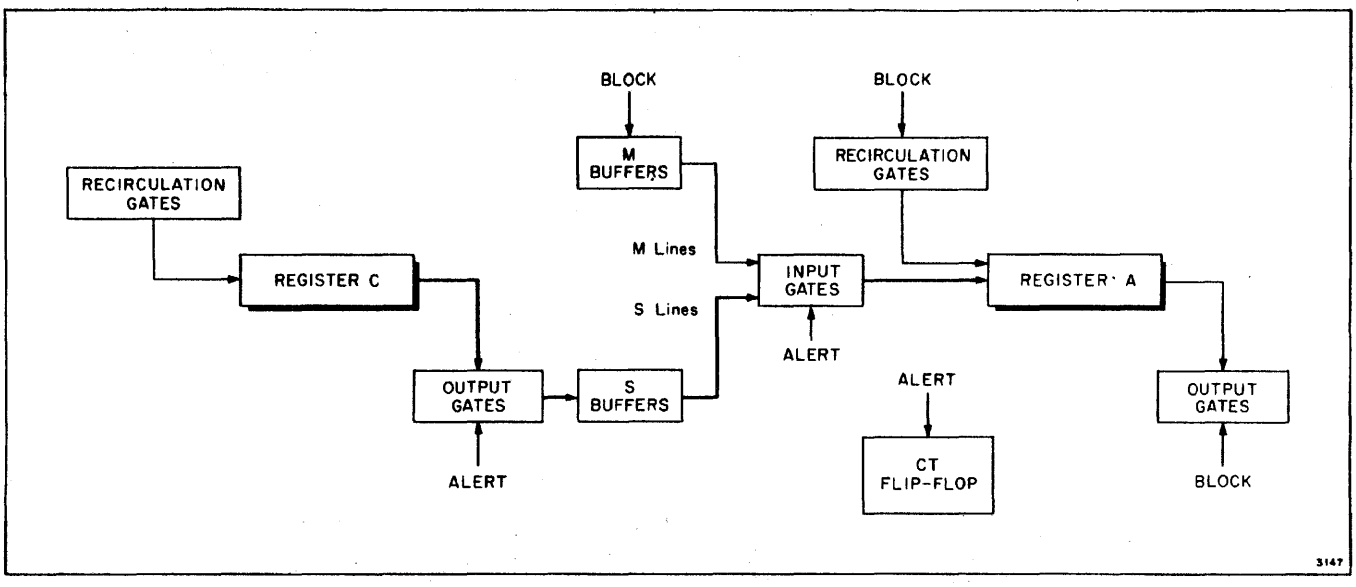


Figure 5-11. Second Stage of 22(I 11), 27(I 21), and 42(I 31) Test Instructions

APPENDIX C

INSTRUCTION LIST

APPENDIX C

INSTRUCTION LIST

C-1. This appendix contains a list of all the instructions provided for UCT. These instructions are divided into seven groups:

(1) Arithmetic Instructions
(2) Transfer Instructions
(3) Translating Instructions
(4) Miscellaneous Instructions
(5) Comparison Instructions
(6) Input-Output Test Instructions
(7) Input-Output Instructions

The four arithmetic instructions in group 1 and the 60 instruction of group 2 are explained in detail in Section IV of this manual. The input-output instructions (group 7) are explained in detail in the separate manuals for the three input-output devices. All the remaining instructions are explained on the block diagram level in Section V of this manual.

C-2. The letter m refers to the m address of the instruction word, the letter c to the c address. A dotted line in place of the m address means that any digits in this part of the instruction word are ignored by the computer. The parenthesis enclosing letters means "contents of." For example, (rA) means contents of register A.

C-3. The UCT instructions are:

Table C-1. UCT Instruction List

INSTRUCTION	DESCRIPTION	TIMING (No. of word times for minimum latency)
	Arithmetic Instructions	
70 m c	Add algebraically (m) to (rA) and put result into rA	5
75 m c	Subtract algebraically (m) from (rA) and put result into rA	5
85 m c	Multiply (rL) by (m) and put 20 digit product into rA (most significant half) and rX (least significant half), each half having sign of product	5 plus the number of digits in the multiplier plus the sum of the multiplier digits.
55 m c	Divide (m) by (rL) and put unrounded result and sign of the quotient in rA and remainder in rX. Remainder has sign of dividend. If divisor = dividend or divisor = 0, an overflow to c+1 will result	20 plus the sum of the odd digits of the quotient plus the sum of the tens complements of the even digits. MSD is #1 (odd).
	Transfer Instructions	
25 m c	Transfer (m) to rA	4
60 m c	Transfer (rA) to m	4
05 m c	Transfer (m) to rX	4
65 m c	Transfer (rX) to m	4
30 m c	Transfer (m) to rL	4
50 m c	Transfer (rL) to m	4
77...c	Transfer (rA) to rL	3

	Translation Instructions	
12...c	Send (rA) and (rX) through the card code to UCT translator and deposit result into rA and clear rX. [(rA) contained the unprimed word of the card image.] Sign of the result is the sign of the unprimed word in rA	3
17...c	Send (rA) through the UCT-to-card code translator and deposit the two words in rA and rX. [(rA) contains the unprimed word and rX the primed.] Signs of results are positive. All UCT code zeros are translated to Remington Rand code punching zeros	3
	Miscellaneous Instructions	
20 m c	Superimpose the 1 bits of (m) on to (rA) and leave the result in rA. Sign of rA is undisturbed	4
35 m c	Change the bits in each decimal digit of (rA) to binary zero wherever (m) has a binary zero in the corresponding bit position. Sign of rA is undisturbed	4
32 n c	Shift (rA) to the right n places into rX which also is shifting to the right into rA. The sign positions are not involved in this shift. n can vary between 0 and 10 and is a single digit inserted in the next to most significant digit position of m	3 + n

Table C-1. UCT Instruction List (cont)

Miscellaneous Instructions (cont)		
37 n c	Shift (rA) to the left n places losing the most significant digits and bringing in zeros in the least significant places on the right. n can vary from 0 to 10 and is a single digit inserted in the next to most significant digit position of m. The sign digit is undisturbed	3 + n
62...c	"Zero Suppress" the word in rA. All Remington Rand punching zeros and commas (and any other character containing a one in the least significant bit position) to the left of the most significant digit are replaced by non-punching zeros	4
67...c	Stop the computer	Indefinite
Comparison Instructions		
82 m c	If (rA) equals (rL), the next instruction is in m; if not, the next instruction is in c	3
87 m c	If (rA) is algebraically greater than (rL), the next instruction is in m; if not, the next instruction is in c	3
Input-Output Test Instructions		
22 m c	If the read-punch unit buffer is loaded, transfer (rC) to rA and go to m for the next instruction. If not, go to c for the next instruction	3

Input-Output Test Instructions (cont)		
42 m c	If the high speed reader unit buffer is loaded, transfer (rC) to rA and go to m for the next instruction. If not, go to c for the next instruction	3
27 m c	If a print or advance operation is in progress (print flip-flop or paper feed flip-flop is reset), go to c for the next instruction. If not, (i.e. print flip-flop and paper feed flip-flop are set), transfer (rC) to rA and go to m for the next instruction	3
Input-Output Instructions		
	Read-Punch Unit	
81 m c	Transfer the output card images from the band designated by m to the buffer band. When the memory to buffer transfer is completed, the computer is free to operate on other instructions	203
46 m c	Wait until the buffer is loaded, then transfer the input card images from the read-punch buffer to the input band designated by m (i.e. 0000, 0200, 0400, etc.)	203
57 m c	Select output stacker. If m = 0000, stacker #0 is selected, if m = 0100, stacker #1 is selected	3

Table C-1. UCT Instruction List (cont)

	Input-Output Instructions (cont)	
	Card Reader	
72...c	Feed one card into the continuously moving rollers of the feed. The card will be read at each station in turn, and the data stored in the buffer band	3
96 m c	Wait until the card-reader buffer is loaded, then transfer this data from the buffer to the memory band designated by m (i.e. 0000, 0200, 0400, etc.)	203
47 m c	Select output stacker. If m = 0000, stacker #0 is selected; if m = 0100, stacker #1 is selected; if m = 0200, stacker #2 is selected	3
	Printer	
16 m c	Advance. Wait until the previous advance or print operation is completed, then move the paper "m" lines as indicated by the two least significant digits of m. Once the paper movement is started, the computer is free to operate on other instructions	3
11 m c	Advance and print. Wait until the previous advance (16) or print (11) operation is completed, then start to move paper "m" lines, as indicated by the two least significant digits of m. The two most significant digits of m indicate the print-interlace band	591