Prepared by: J. C. Hartman
Date: 3 June 1965

Subject: CP-823/U (Central Processor Operation

The operational configuration of the 1830 computer is divided into three parts: (1) I/O, (2) Memory, and (3) Central Processor. Both I/O and the Central Processor time share memory, however, I/O is given priority over the Central Processor.

The Central Processor (CP) unit contains all of the logic for arithmetic, computer control and memory access control. Figure 1 illustrates registers, basic translation networks, and communication paths within the central processor. For detailed information on basic sequences and the registers, logic networks, and communication paths used for events during a typical sequence, see Figure 1 and the sequence timing charts included in this note.

Sequence timing charts are presented for the basic A, A^b, B and the C sequences for a right shift instruction. Table 1 indicates the general tasks accomplished during each sequence.

TABLE 1

A Sequence - Increment Program Address by 1; Instruction from Memory to U.

A^b Sequence - Add the B register specified by the b designator to the lower half of U.

A^j Sequence - This sequence is used to read the B registers, to increment or decrement a B register when required and to test for the completion of the repeat operation.

B Sequence - Transfers $U_L \rightarrow S$ to Read (Y) from memory; initiates the C sequence.

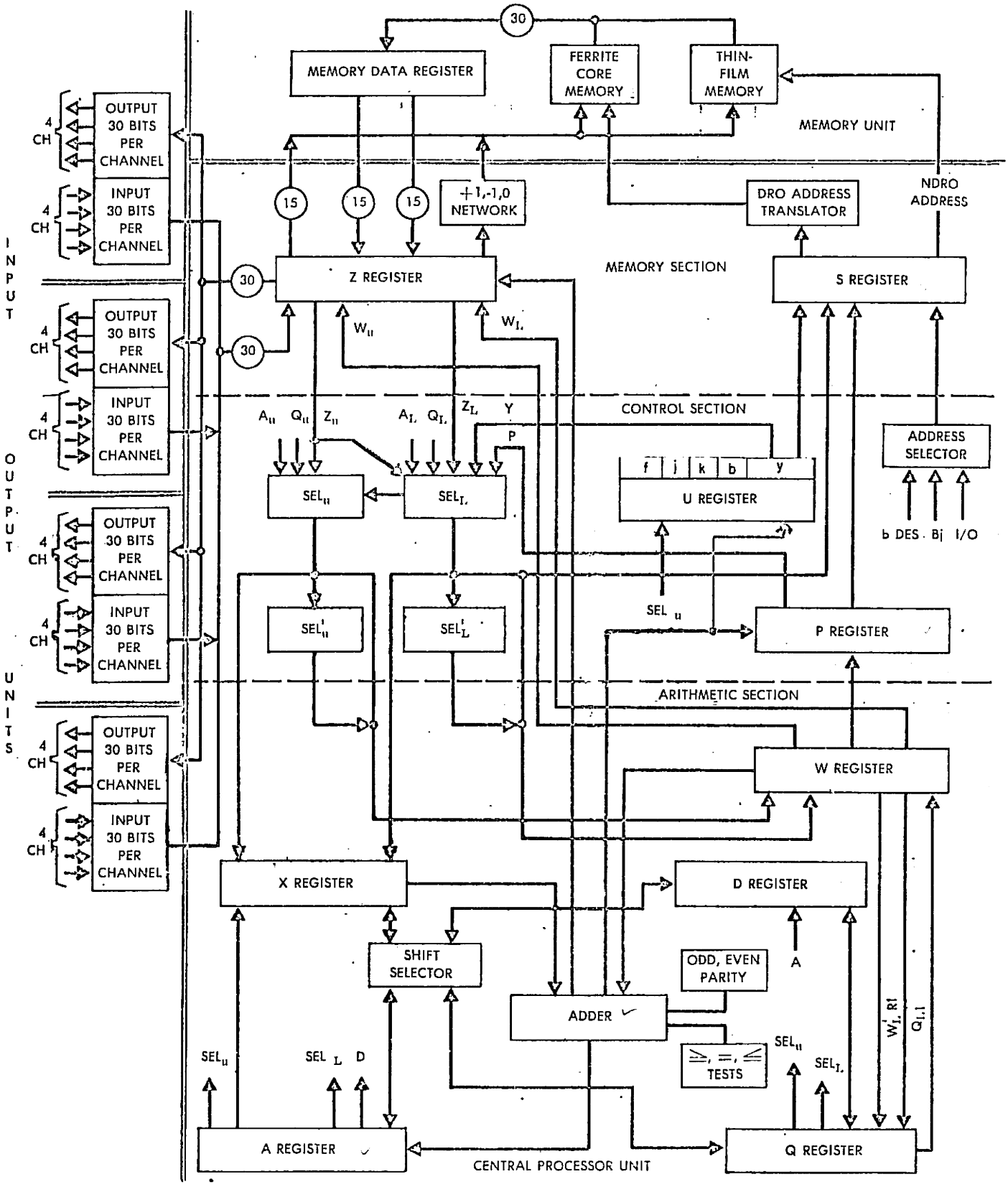
B^j Sequence - This sequence is used whenever information is to be stored into a B register.

C Sequence - This is the sequence during which the operand is transferred to or from memory, and the logic of the instruction is carried out (i.e. addition, subtraction, logical product, etc.).

D Sequence - This sequence is executed for special store data operations; for example, it occurs during a replace operation when the data is to be replaced into memory.

Main timing

On each of the sequence charts, the time intervals noted as T11 through T64 are generated from the main timing chain. The timing interval T11 is initiated at the beginning of clock phase 2 and terminated at the end of



NOTE: Units enclosed within double lines are separately packaged.

Figure 1. CP-823/U Computer — Block Diagram with Full 16 Channels Input and 16 Channels Output

clock phase 1. On each succeeding clock phase a new interval is generated. It can be seen that intervals that end in 1 (T11, 21, 31, etc.) terminate at the end of phase 1, those intervals that end in 3 terminate at the end of phase 3. Each interval lasts for 667 nanoseconds, and the main timing chain reinitiates itself every 4 microseconds.

Instruction Sequence

The instruction sequences are logic chains designed to take advantage of the 4 microsecond access of memory. Once a sequence is initiated, the chain of major events for the sequence must be completed before the next sequence (involving a memory reference) is initiated. As shown on the A and A^b Sequence Timing Chart the A^b Sequence is initiated a short time before the A Sequence is completed, but no major logic chain of the A Sequence is affected.

I/O operations, although having priority to access memory, are prevented from using memory until the sequence being executed is completed. However, I/O will inhibit further instruction sequences until I/O has satisfied all current memory demands. Upon completion of I/O, the instruction sequences will be reinitiated by main timing at the point I/O interrupted.

A Sequence

The A Sequence is common to all instructions. In the A Sequence the P register is incremented by 1 and the instruction to be executed is read from memory and is transmitted to the U register. As shown in the A sequence timing chart, the contents of the P register is transmitted to the S register to provide the address from which the instruction is read. The address in P is then gated into the X register and a +1 is sent to the W register. X and W are then added together in the adder and the sum (address of next instruction to be read) is gated into the P register. (For all of the normal operations requiring a transmission into a register the register is cleared - set to $\emptyset$ - before the information is gated into the register). While P is being incremented by one (1), the memory read cycle is occurring. Shortly after the next instruction address is placed in the P register, the instruction to be executed is gated from the memory data register into the Z register. The upper half of the Z register (15 bits) is then transmitted through the selector (a transmission path switching network) to the upper half of the U register. The instruction in the Z register is also transmitted through the selector to the W register. The X register (which has been cleared) and the W register are then summed in the adder and the lower 15 bits are transmitted into the lower half of the U register. The instruction to be executed is now contained in the U register.

Associated with the A sequence, is an Abort function (abort flip-flop), which under special conditions will prevent the execution of the instruction just placed in the U register. Under normal operations, the abort condition will be clear and the instruction will continue to completion. However if the abort condition does exist, an inhibit is imposed on all sequences, that might normally follow the A sequence, to prevent instruction execution, and to allow another A sequence to be initiated. Shortly after the next A sequence is initiated, the abort condition is cleared to allow normal instruction sequencing (the abort flip-flop is <u>always</u> cleared near the end

of each A sequence regardless if it had been clear or set). The result of the abort condition being set is that two A sequences will be executed in succession and the instruction placed in the U register during the first A sequence will be skipped.

The abort condition is established for the following conditions:

1. A normal instruction j test - The j test is usually performed and when necessary the abort condition is set in the C sequence.

2. Jump, Return Jump, Arithmetic Jump, and Arithmetic Return Jump j test (f = 61,. 65, 60 or 64) - The j test is performed early in the A sequence. If the test is affirmative, the abort condition will be established to prevent execution of the jump instructions in the U register and initiate another A sequence, to execute the next instruction.

Another special operation performed in the A sequence is associated with the Terminate Input (f=66) or Terminate Output (f=67) instructions. These instructions utilize only the A sequence (See Sequence chart page 17) and to prevent execution of other instruction sequences an inhibit is placed on the strobe signal used to initiate the other sequences. Because no other sequence exist, another A sequence will be initiated to execute the next instruction (Note - the abort condition is not used).

A^b Sequence

The A^b Sequence is always executed when the operand in U_L (Lower 15 bits) is to be incremented by an index register (B register). Initiation of the A^b sequence is normally from the A sequence, but during a repeat mode the A^b sequence is initiated by the B sequence for a B^b index (RPT Mode & r = 4 - 7*) requirement and by the A^j sequence for a B^b index operation (Rpt Mode & r = 3 or 7*) (See the sequence chart pages 16,17). As shown in the A^b sequence timing chart, U_L is transmitted to the X register via the SEL_L network while the contents of the index register is being read from memory. The contents of the index register is transmitted to the W register via the Z register and the Selector (Sel) network. Addition is carried out in the adder and the result is transmitted to U_L for use as an operand or an operand address.

Three sequences, which are the B-C sequence, D sequence, and A^j sequence, can be initiated from the A^b sequence. The most common path is to initiate the B-C sequence for normal instruction execution. The D sequence is entered for a replace operation during the repeat mode with r equal to 3 or 7, and the A^j sequence is initiated from A^b for a B register store instruction (f16), a B skip (f71), or a B jump (f72).

*The Repeat Mode and the R register are established by the Repeat Instruction (f=70) and the j designator. The j designator for the Repeat Instruction is translated directly into the type of repeat to be performed and is held by the R register until the repeat is terminated (See the B sequence for more information).

A^j Sequence

The A^j sequence is primarily used to read the content of a B register from memory into the Z register, and to restore the B register unchanged, decremented or incremented through the 0, +1, -1 network. Other operations occur also but they are instruction oriented and are discussed below. Instructions that use the A^j sequence are Store B (f = 16), B Skip (f = 71), B Jump (f = 72), and all instructions being repeated. Following is more specific information on each instruction:

Store B (f = 16) - The A^j sequence is used to read the content of the B register (defined by the b-designator) for it to be stored during the B and C sequences. The content of the B register is restored unchanged, and the content of the B register is saved for the C sequence by transmitting the Z register to the W register through the selector network. See the sequence chart page 14 to see how the A^j sequence is included in the sequence logic chain.

B Skip (f = 71) - The A^j sequence is used to make the contents of the B register available for the $(B^b) - Y = 0$ test in the C sequence and to increment the B register by one. When the B register is restored from Z, the +1 network is used to increment the register. The content of the Z register is transmitted through the SEL network to set the complement of Z in the W register in preparation for use in the C sequence. (See the sequence chart page 19 to see when the sequence is used for this instruction).

B Jump (f = 72) - The A^j sequence is used to read the content of a B register, to decrement the content, and to perform a test for the B register equal to zero. The B register is read into the Z register from memory, and it is decremented by one using the -1 network in the restore operation. A test is made, during the A^j sequence, for b = 0, by transferring the content of Z through the SEL to the W register (complement of Z to W); then X (X = 0 at the time) and W registers are added, using the adder network, and the result is tested for zero. If a zero (0) condition exists, an inhibit is established to prevent the initiation of the B and C sequences and to enable the initiation of the B^j sequence (the B^j sequence is used to clear the B register since the restore operation would have decremented it from 0 to 77777). See page 19 for the sequence chart showing how this instruction used the A^j sequence).

Repeated Instruction - The A^j sequence is executed each time an instruction is repeated to decrement the content of B^7 and to test B^7 for a value of one (1). Depending on the type of repeat, the A^j sequence may also be used to decrement (r = 2) or increment (r = 1) the lower half of the U register. B^7 is decremented by one using the -1 network during the restore cycle. The test for $B^7 = 1$ is made in the adder network after the content of the Z register has been loaded in the X register, through the SEL network, and -1 has been loaded in the W register. If the result of the X and W addition is zero (0), the repeat condition is terminated and the next instruction will be allowed to initiate in a normal manner. See the sequence charts to see how the A^j sequence is used when an instruction is being repeated.

B Sequence

The B sequence is always run in parallel with the C sequence, and it is used to do odd bookkeeping tasks while the C sequence is performing the logical instruction operations. The B sequence will normally initiate the memory cycle used by the C sequence (see the shift instruction timing charts). When a computer operation is going to require more than 4 usec, the B sequence contains the logic to sense this need and to hold (inhibit) an A sequence from being initiated. The first time an instruction is repeated, after the rpt instruction (f70) is executed, the B sequence will set a rpt condition by setting bit 3 of the R register.

B^j Sequence

The B^j sequence is used when data is to be entered into a B register. Instructions ENTER B (f = 12), REPEAT (f = 70), B SKIP (f = 71) and B JUMP (f = 72) utilize the B^j sequence. Following is a more detailed explanation for each instruction.

ENTER B (f = 12) - The B^j sequence is used to enter a B register with the operand specified by U_L and the k designator.

REPEAT (f = 70) - The B^j sequence is used to load the B^7 register with the value of the repeat count.

B SKIP (f = 71) - The B^j sequence is used to clear the B register specified by the b-designator and increment the P register by one (1), skip the next instruction. To increment the P register, P is transmitted through the SEL network to the X register, a +1 is loaded into the W register and the output of the adder network is gated back into the P register.

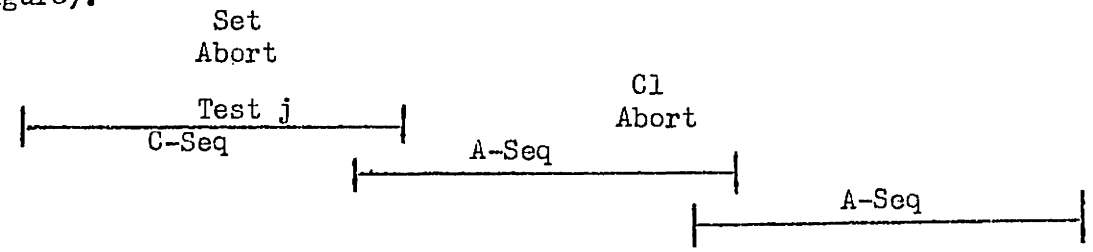
B JUMP (f = 72) - The B^j sequence is used to clear the B register specified by the b-designator after the A^j sequence tests that a B register 0 condition does exist. The B^j sequence must clear the B register since the A^j sequence decremented the register from 0 to 77777.

C Sequence

The C sequence is the sequence in which most of the logic to execute the instruction is executed. Many basic operations such as Read Y, Store Y, Multiply-Divide/Sq.Rt.-Shift, and C extension are treated as subsequences and are initiated out of the C sequence; each subsequence has its own timing chain that operates in unison with the C sequence (or any sequence that initiates it). The timing chart for the C sequence, of a shift instruction, is shown since all subsequences are initiated except the Store Y subsequence. As shown on the C sequence timing chart, the Read Y subsequence is initiated at the time the memory transfer to the Z register is enabled. The Read Y subsequence will provide the enable to allow the operand in the Z register to enter the selector network; the operand will also be controlled by the k designator. The C sequence will then provide the output at time T33 to transmit the selector to the K register on Ø3. Initiation of the shift subsequence (this is the Multiplay, Divide/Sq Rt, Shift Sequence) will begin the timing and logic that controls the shift operation (see further information below on the shift operation). When a shift is being executed, the B sequence will initiate a hold condition to prevent an A sequence from being started, because the shift execution will require more than the

4 microseconds allotted to the C sequence. The shift subsequence will maintain control until the K register is decremented to zero, and then the C sequence extension is enabled to perform the j Test and to clear the hold condition.

The j Test is used each time an instruction requires a test of A or Q (defined by the j designator), and if the result is positive, the next instruction will be skipped by initiating the Abort function. The Abort will generate an inhibit to prevent the A^b, A^j, and B sequences from being initiated, but will allow an A sequence to be initiated (see Abort sequence figure).



ABORT SEQUENCES EXAMPLE

The Abort flip-flop is cleared during the final stages of the first A-sequence but after the next A-sequence has been initiated. Normal sequencing will occur from the second A-sequence. An instruction skip will occur since the first A sequence will place an instruction in the U register and will increment the P register, and the second A-sequence will insert another instruction into U and will again increment P. This action will effectively skip the instruction placed in U on the first A-sequence.

Shift. - The actual data shift operation occurs during the C-sequence and consists of two data transfers which depend upon the contents of the K register (6 bits - See Shift Operation Figure). One data transfer is from the A, Q, or AQ registers to the X, D, or XD register. The transfer will allow a shift of one bit right or left; a direct transfer will occur if the one's digit of the K register (low 3 bits) is zero. A second data transfer will place the data back in A, Q, or AQ shifted (left or right) 8 bits, if the tens digit of the K register (upper 3 bits) is unequal to zero; a zero in the upper octal digit of K will cause a direct data transfer to the A, Q or AQ registers with no shift. Each time a transfer occurs the K register is decremented; the shift cycle will continue until the K register has been decremented to zero. It can be seen that, since a complete cycle must occur, a shift of 1, 8 or 9 bits will take the same amount of time; a shift of 7_{10} or 56_{10} thru 63_{10} will require 7 cycles (for example, a shift of 57_{10} will require 7 cycles - however the shift will be composed of 6 cycles for a shift of 8 bits each cycle and one cycle for a shift of 9 bits. On the six (6) 8-bit shifts, the data transfer from A, Q, or AQ to X, D, or XD will be direct since the lower digit of K would be a zero after the first cycle when a shift of 9 occurred).

D Sequence

The D sequence is executed when it is necessary to store an operand, in memory, that has been provided during the B, C sequences. Instructions that use the D sequence are all Replace Operations, Return Jump (both manual and Arithmetic), and Input/Output operations. Specific information on each instruction type is discussed below.

Replace Instructions. - The D sequence is used to store the content of the A register into the address specified by U_L and the k designator (data was previously entered into the A register during the B, C sequences).

A standard store operation is executed as follows: See the sequence charts to see how the D sequence fits into the sequence logic chain. Page

1. U_L is transmitted to S for the storage address.

2. A is transmitted to the SEL and the SEL is transmitted to W (if the word is to be complimented SEL^1 will be used).

3. Memory Data Register and W are transmitted to Z (the memory data register contains the information currently contained in the address being changed). If the whole word is being stored (k = 3 or 7), the memory data register to Z transmission is inhibited; if an upper half word is being stored (k = 2 or 6), the upper half of W and the lower half of the memory data register is transmitted into Z; if a lower half word is being stored (k = 1 or 5), the lower half of the memory data register and the upper half of W is transmitted into Z.

4. The content of Z is then written into the storage address.

Return Jump Instructions. - The D sequence is used to store the return address (an address one (1) greater than the address of the Return Jump) into the address specified by the return jump instruction. Before the D sequence is executed, the B, C sequences will establish the following conditions:

- The X register holds the previous content of the P register (this is the address of the instruction following the Return Jump instruction).

- The P register contains the first address of the program to be executed (specified by U_L and the k designator).

The D sequence performs the following logic:

1. The P register is transmitted to S ($P \rightarrow S$). This will initiate a memory cycle to read and write information into the first instruction of the routine to be executed.

2. The X and W (clear) registers are added in the adder network and the result is transmitted to U_L (this action will place the return address in UL).

3. U_L is then transmitted through the SEL network to the W register.

4. The W register is then transmitted to the Z register to be stored during the memory restore period.

5. After the W register has been loaded into Z, the P register is transmitted to the X register, through the SEL network, and a +1 is transmitted to the W register. The X and W registers are combined in the adder network and the result is then transmitted back to P. The P register now contains an address one (1) greater than the subprogram entrance address, and the next A sequence will initiate the first instruction of the subprogram.

See the sequence charts pages 17, 18, to obtain composite sequence picture.

Input/Output Instructions. - The input/output instructions (f = 73, 74, 75 & 76) use the D sequence for several events including a store operation. Prior to execution of the D sequence, the B and C sequences were used to read the buffer limits and store the information in the W register. D sequence events are listed below:

1. $\hat{j}$ is transmitted to the S register through the address selector at the beginning of the D sequence (this action will place the address of the buffer control word -100 thru 117 input; 120 thru -137 output - in the S register).

2. The D sequence will also provide an output to the I/O Box, associated with the channel, to translate the function code (f = 73, 74, 75 or 76) and to make the channel active.

3. The W register (buffer limit) is transmitted to the Z register to be stored during the memory write period.

See the sequence chart page 19 to see how the D sequence is used for these instructions.

Composite Sequence Charts

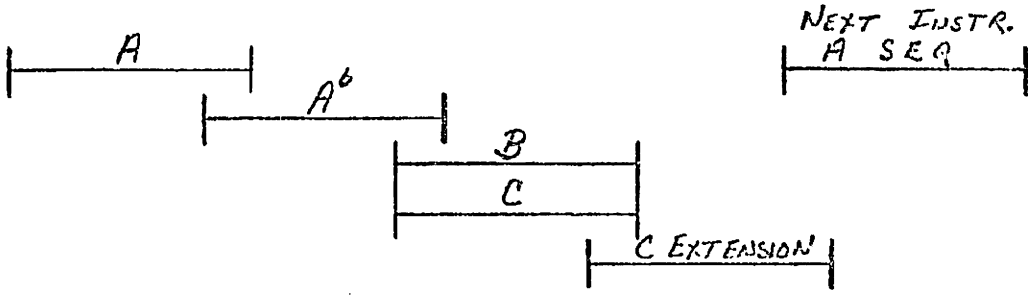
The instruction composite sequence charts present the logical sequences for each instruction. All instructions have been presented, and some instructions have more than one chart since they have different options available.

All sequence charts, when applicable, show an A^b sequence. The A^b sequence is only used if a B register is being used as an index. If the b designator of an instruction is zero (0), then the A^b sequence is skipped and the next sequence executed; the net result will be a four (4) microsecond gain in the instruction execution time.

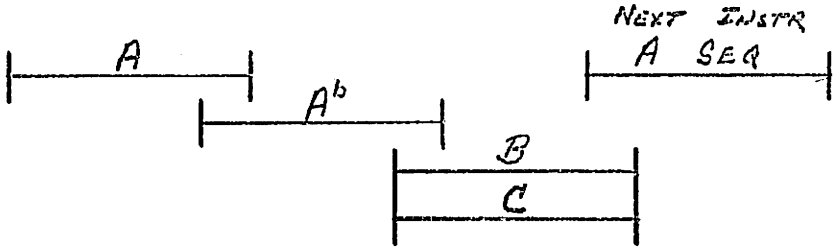
Each sequence is four (4) microseconds long except for some special sequence extending instructions like shift, multiply, divide and square root. For the special instructions, the lengthened sequence consist of multiples of four (4) microsecond periods.

INSTRUCTION COMPOSITE SEQUENCE CHARTS

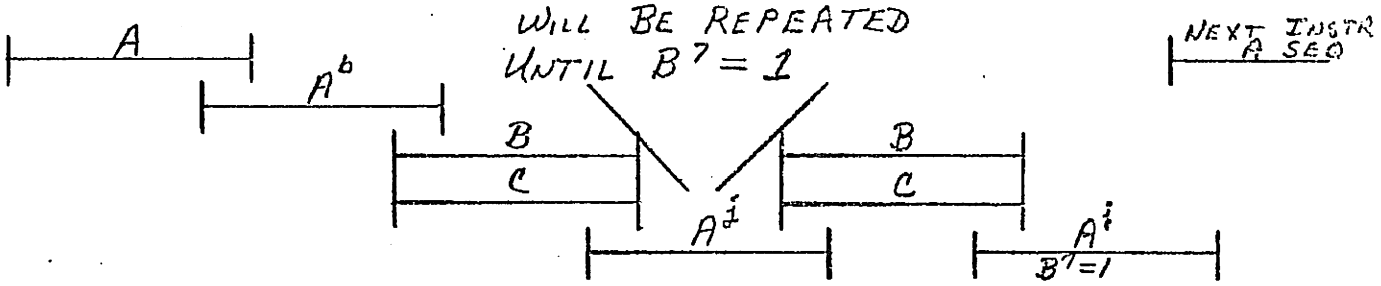
f = 01, 02, 03, 05, 06, 07 (SHIFT INSTRUCTIONS)



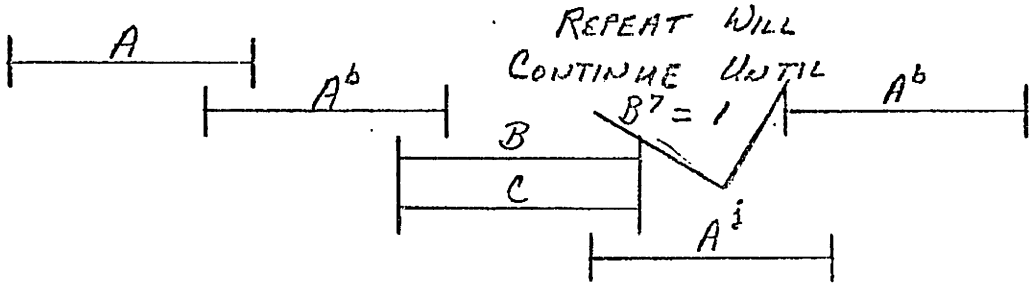
f = 04, 10, 11, 13, 14, 15, 20, 21, 26, 27, 30–33, 40–43, 47, 50–53.



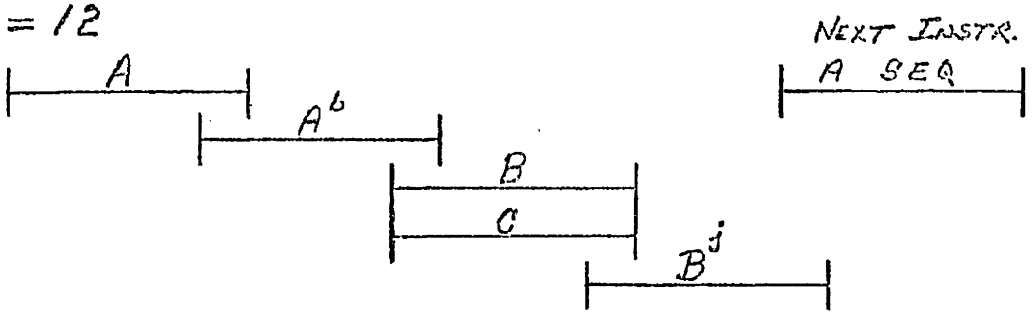
f = 04, 10, 11, 14, 15, 20, 21, 26, 27, 30–33, 40–43, 47, 50–53 (REPEAT MODE r = 0, 1, 2)



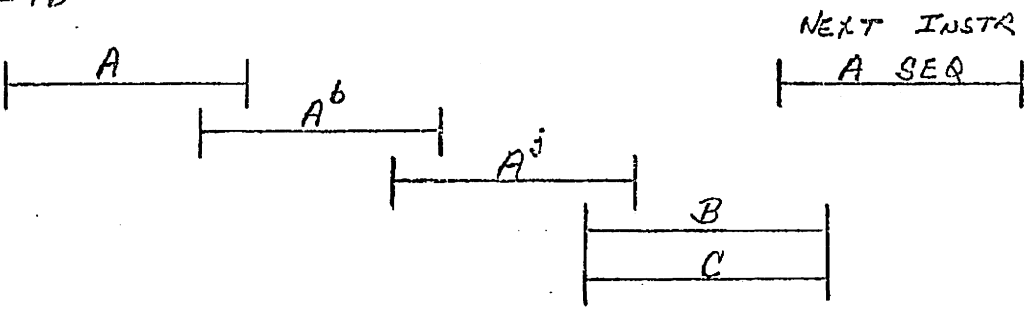
f = 04, 10, 11, 14, 15, 20, 21, 26, 27, 30–33, 40–43, 47, 50–53 (REPEAT MODE, r = 3)



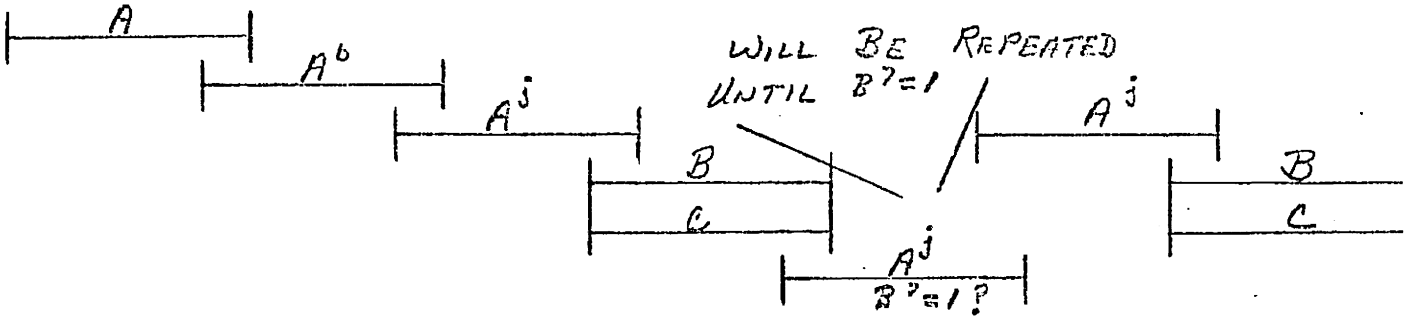
f = 12



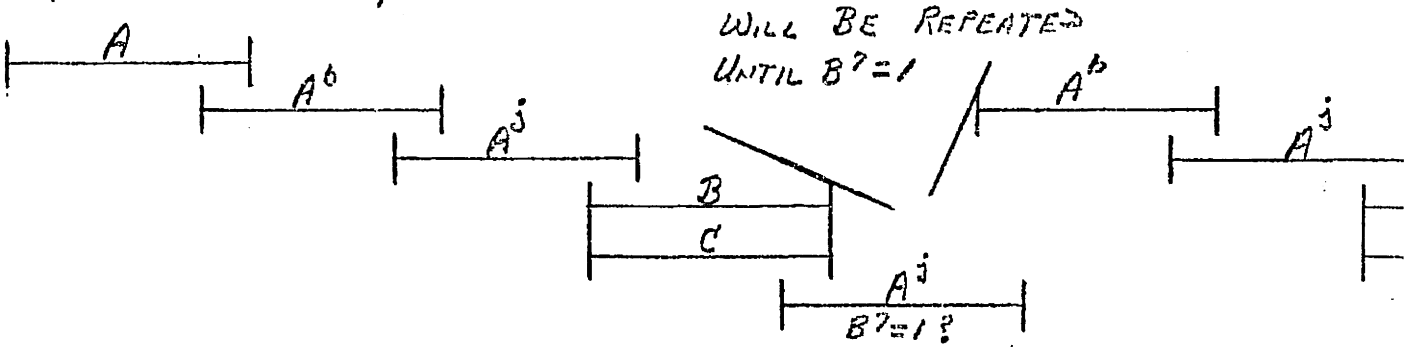
f = 16



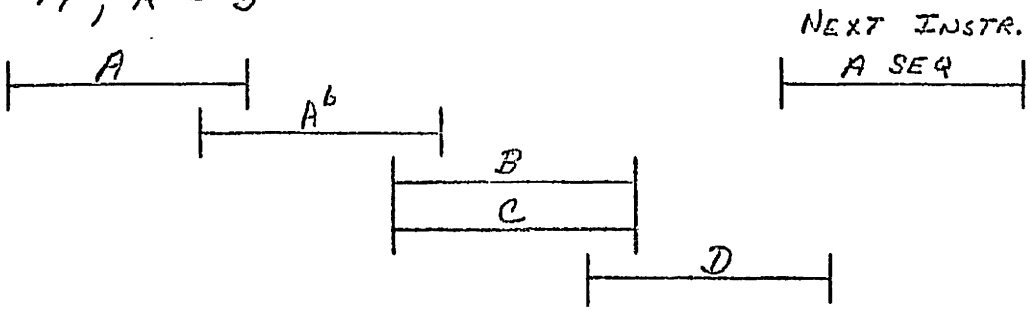
f = 16, REPEAT MODE, r = 0, 1, 2



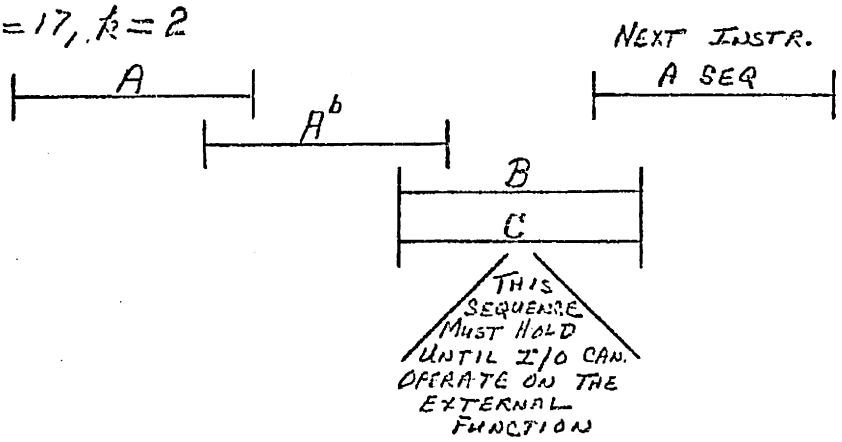
f = 16, REPEAT MODE, r = 3



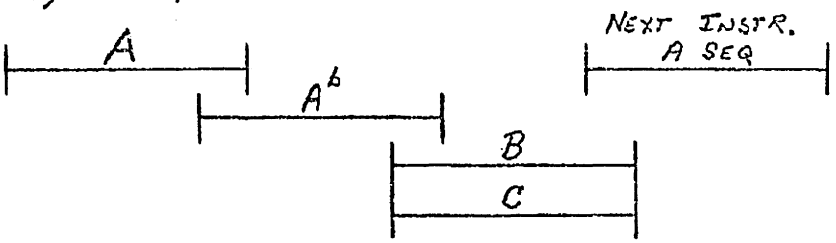
f = 17, k = 3



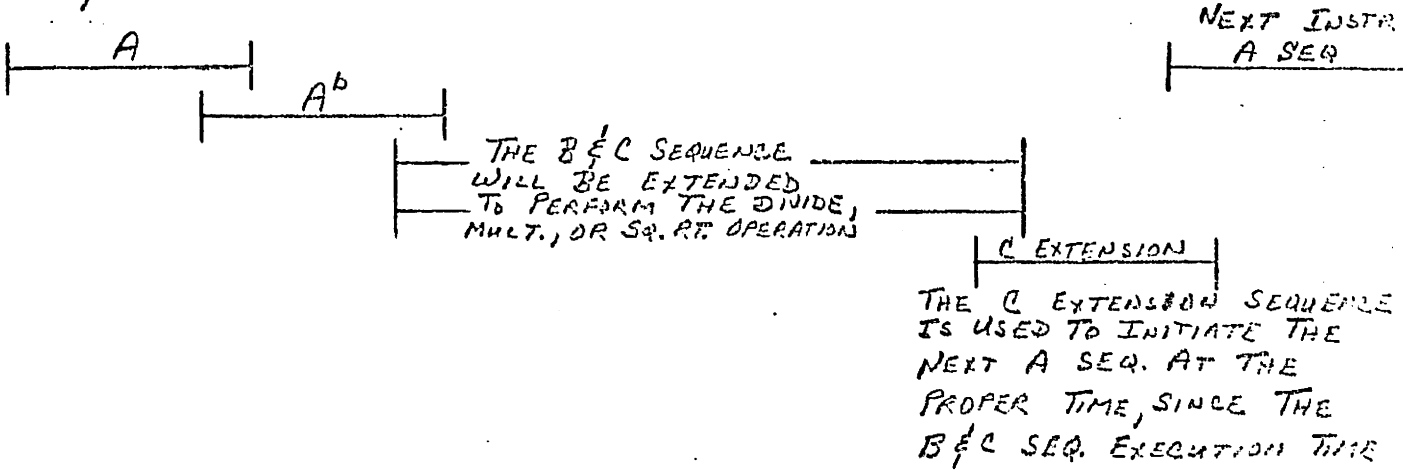
f = 17, k = 2



f = 17, k = 0,1

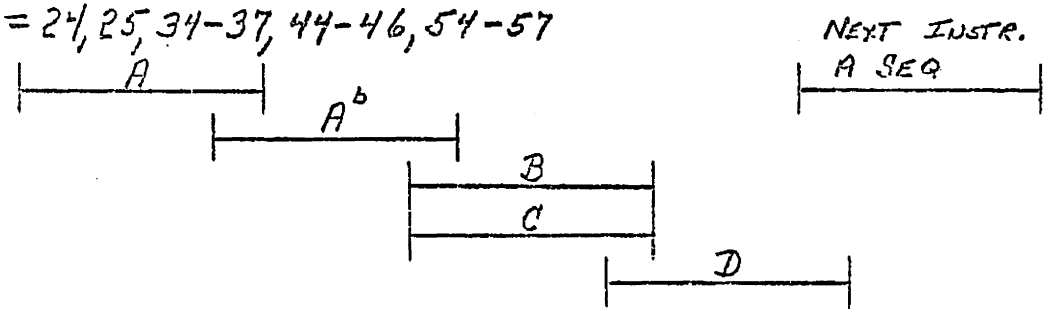


f = 22, 23

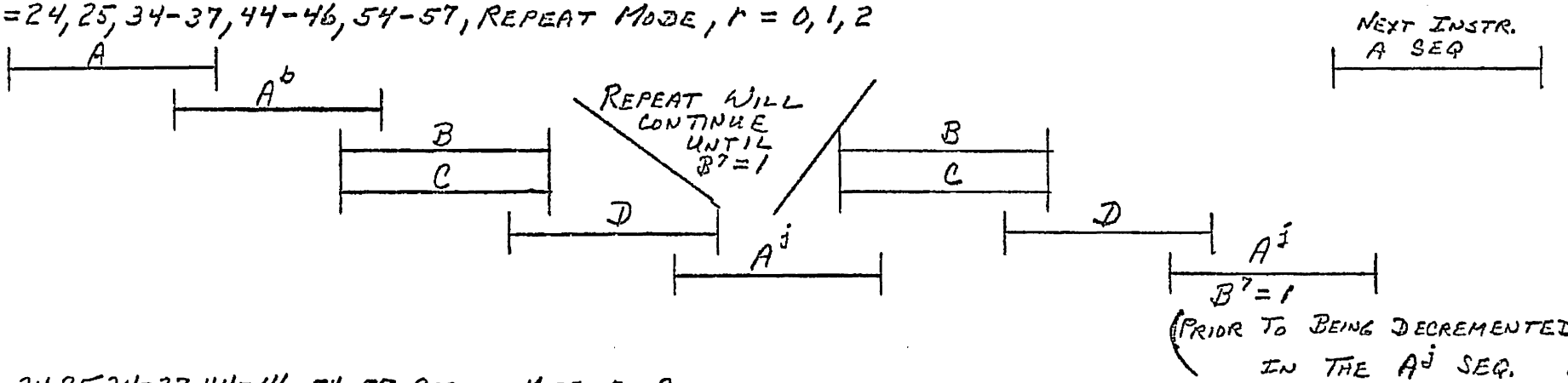


THE C EXTENSION SEQUENCE IS USED TO INITIATE THE NEXT A SEQ. AT THE PROPER TIME, SINCE THE B & C SEQ. EXECUTION TIME

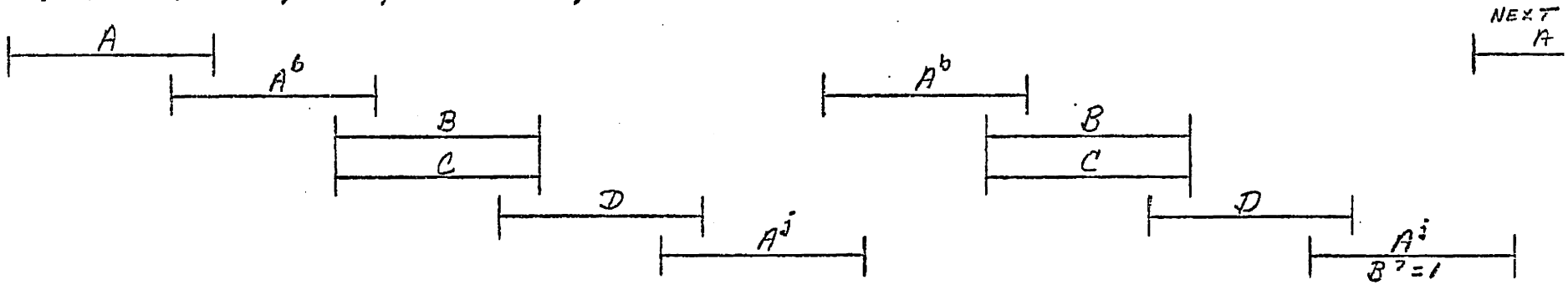
f = 24, 25, 34-37, 44-46, 54-57



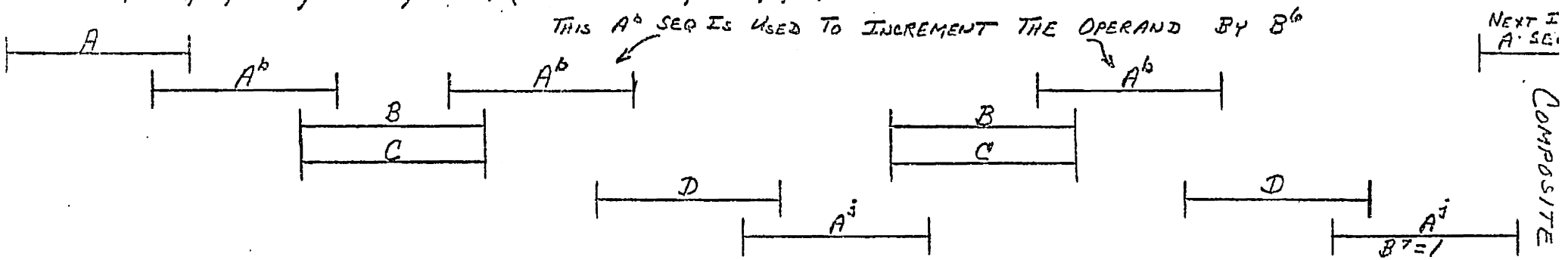
f = 24, 25, 34-37, 44-46, 54-57, REPEAT MODE, r = 0, 1, 2



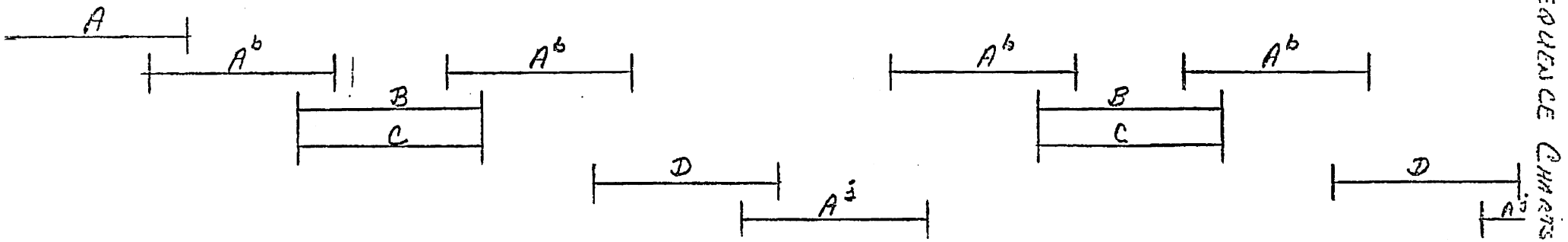
f = 24, 25, 34-37, 44-46, 54-57, REPEAT MODE, r = 3



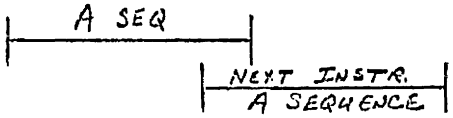
f = 24, 25, 34-37, 44-46, 54-57 (REPEAT MODE, r = 4, 5, 6)



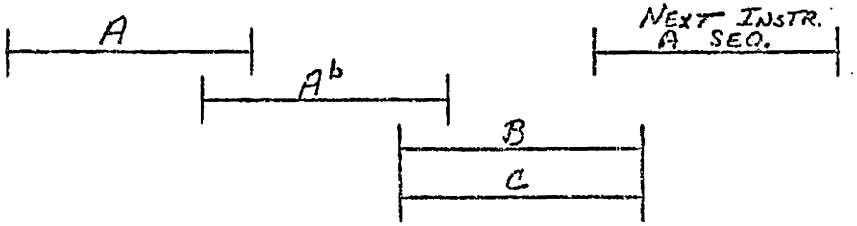
f = 24, 25, 34-37, 44-46, 54-57 (REPEAT MODE, r = 7)



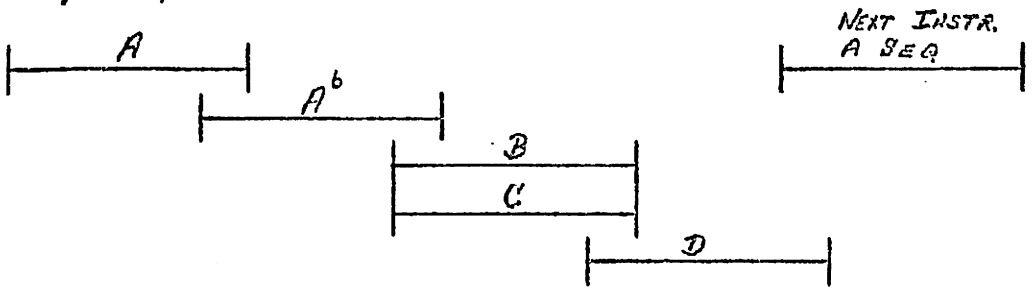
f = 60, 61, 64, 65 & NO JUMP OR RETURN JUMP COND. OR f = 66, 67



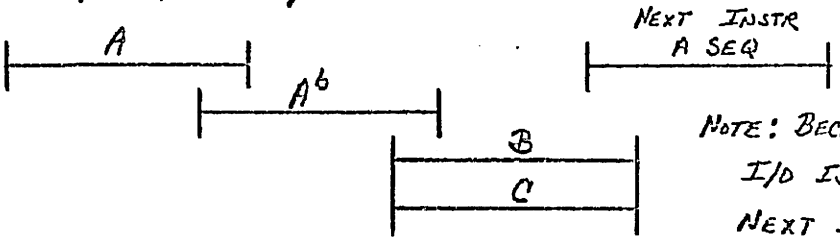
f = 60, 61 & JUMP CONDITION SATISFIED, f = 62, 63 & INPUT, OUTPUT ACTIVE



f = 64, 65 ¢ JUMP CONDITION SATISFIED

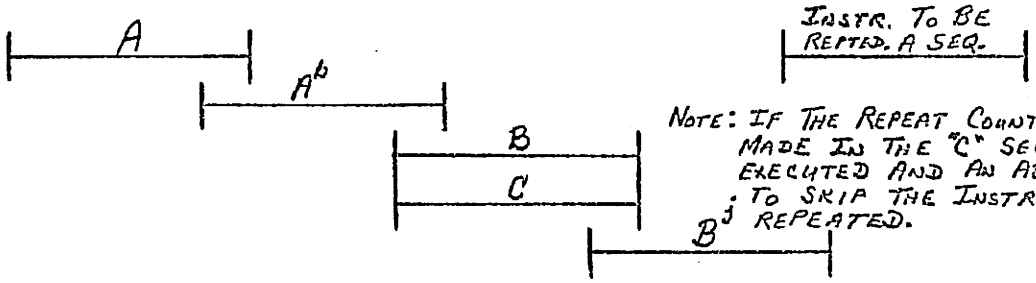


f = 62, 63 ¢ INPUT, OUTPUT NOT ACTIVE



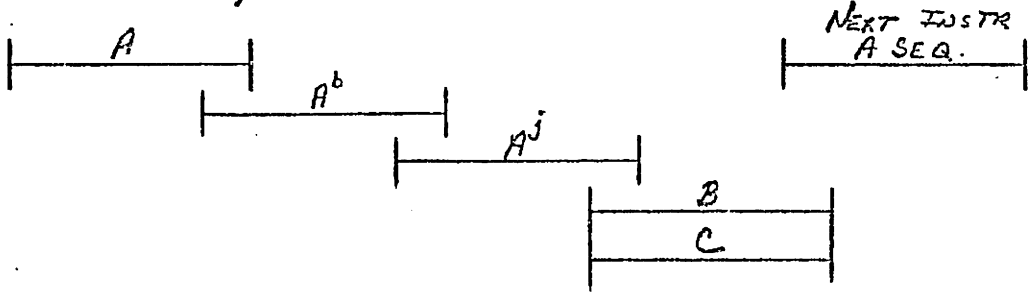
NOTE: BECAUSE THE ACTIVE RESPONSE FROM I/O IS TOO LATE TO PREVENT THE NEXT INSTR. SEQUENCE, THE ABORT CONDITION CAN'T BE USED AND THE A^b, B, AND C SEQUENCE ARE EXECUTED. TO PREVENT THE JUMP, WHEN NOT ACTIVE, AN INHIBIT IS PLACED ON CLEARING THE P REGISTER AND THE JUMP ADDRESS IS NOT TRANSMITTED TO P (W → P IS INHIBITED)

f = 70

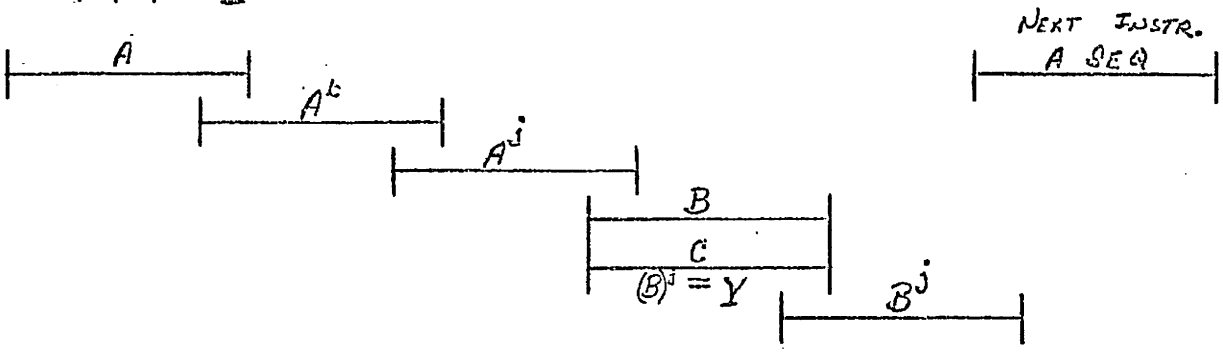


NOTE: IF THE REPEAT COUNT IS ZERO (0) (TEST IS MADE IN THE "C" SEQUENCE), THE B^j IS NOT EXECUTED AND AN ABORT CONDITION IS SET TO SKIP THE INSTRUCTION TO HAVE BEEN B^j REPEATED.

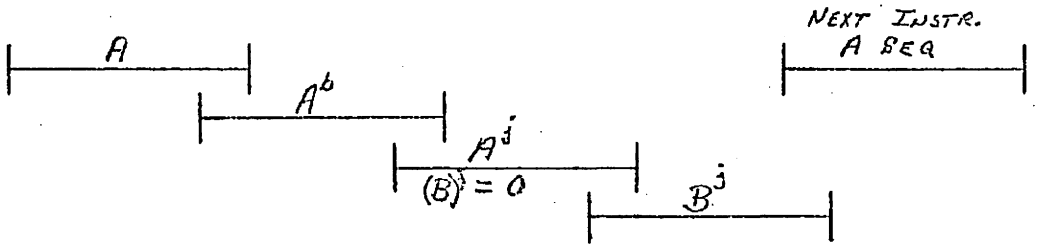
f = 71 ¢ $(B)^j \neq Y$, 72 ¢ $(B)^j \neq 0$



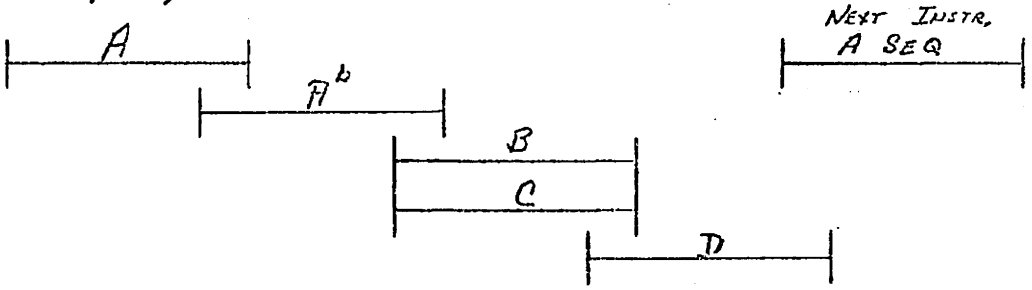
f = 71 $\xi(B)^j = Y$

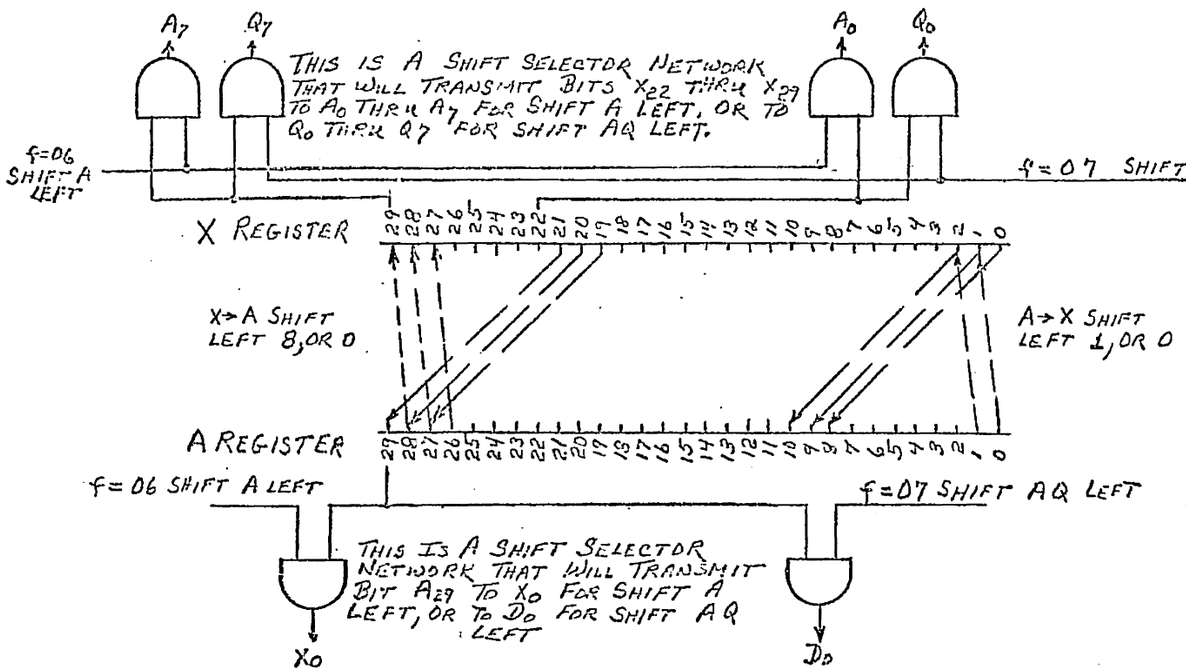


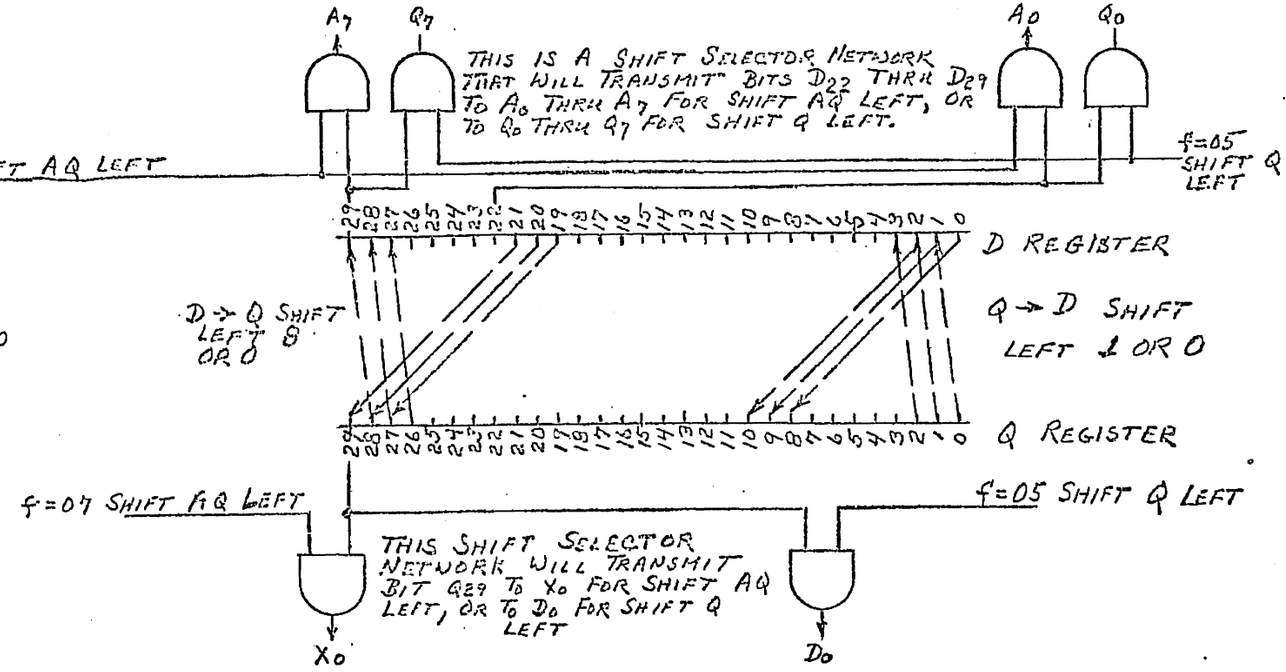
f = 72 $\xi(B)^j = 0$



f = 73, 74, 75, 76



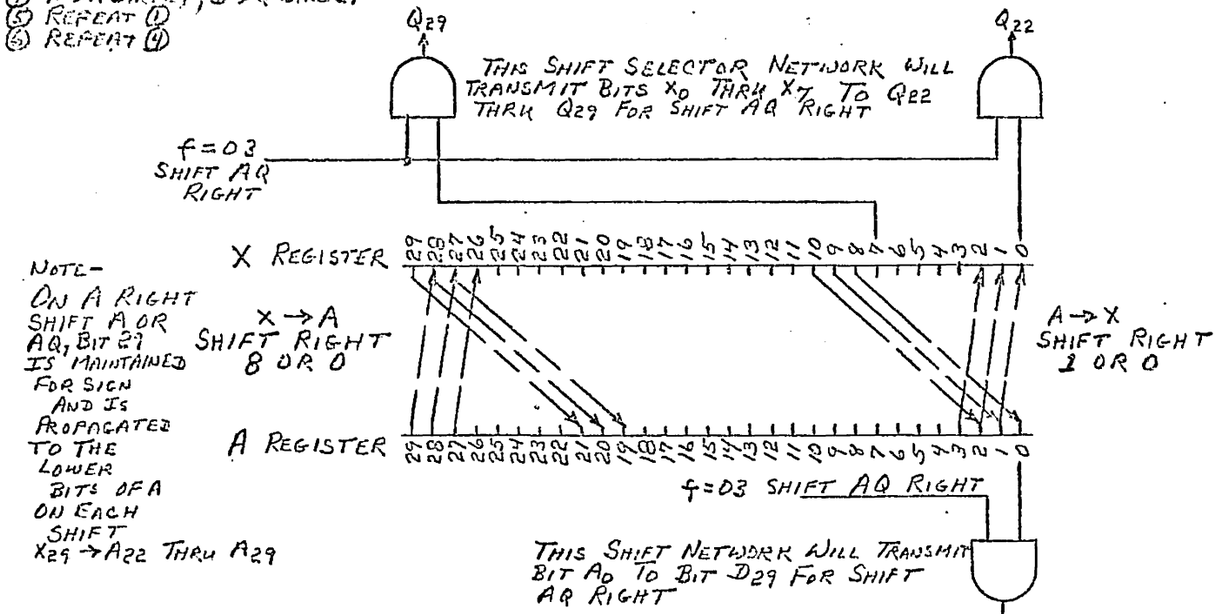




EXAMPLE SHIFT AQ LEFT 13_8

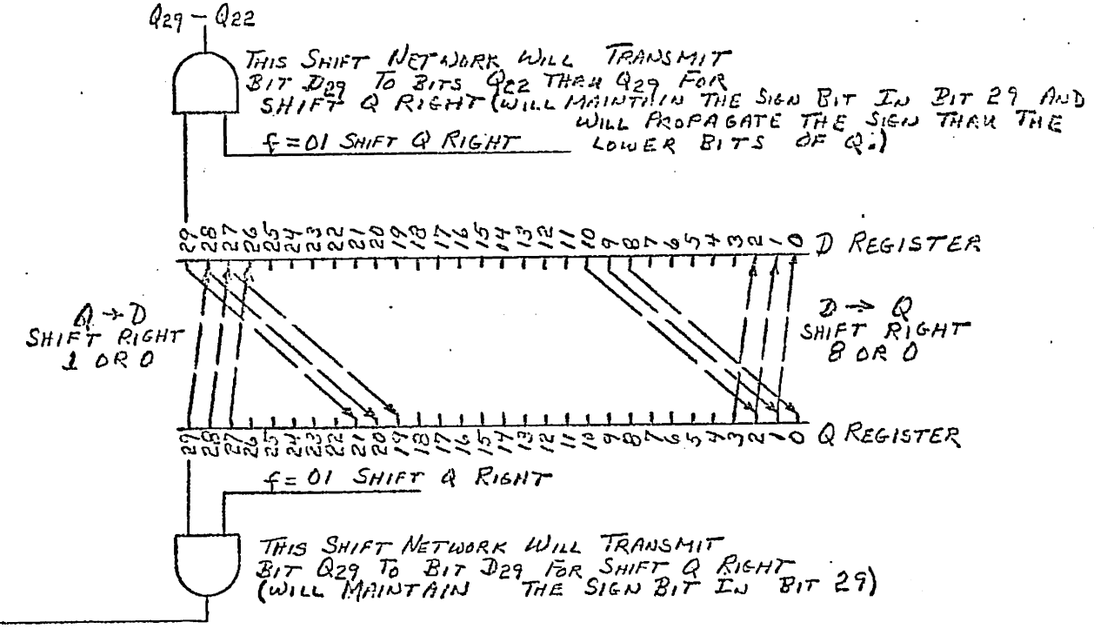
① A→X LEFT 1, Q→D LEFT 1, Q_{29}→X_0, A_{29}→Q_0
② X→A LEFT 8, D→Q LEFT 8, D_{29-22}→A_{7-0}, X_{29-22}→Q_{7-0}
③ A→X LEFT 1, Q→D LEFT 1, Q_{29}→X_0, A_{29}→D_0
④ X→A DIRECT, D→Q DIRECT
⑤ REPEAT ①
⑥ REPEAT ④

SHIFT A, Q, OR AQ LEFT



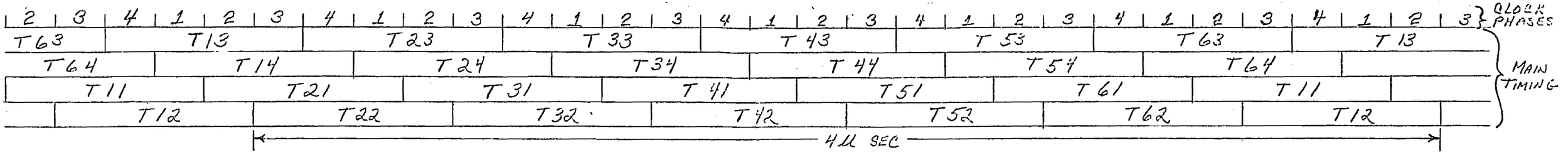
EXAMPLE SHIFT (12_8) RIGHT SHIFT A

11_8 A→X RT 1 X→A RT 10_8
12_8 A→X RT 1 X→A DIRECT

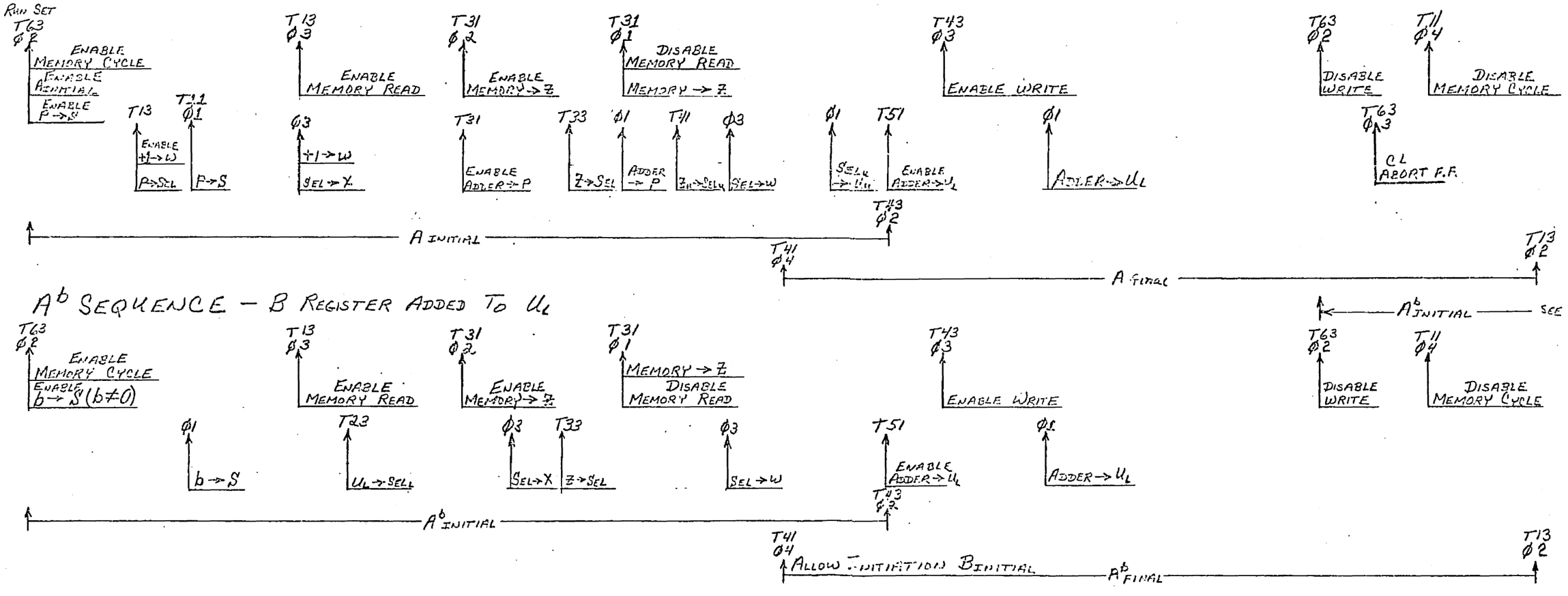


*THIS OR GATE IS USED TO SET BIT D29 AND IS TYPICAL OF ALL THE SELECTOR NETWORKS ILLUSTRATED ON THIS PAGE

SHIFT A, Q, OR AQ RIGHT

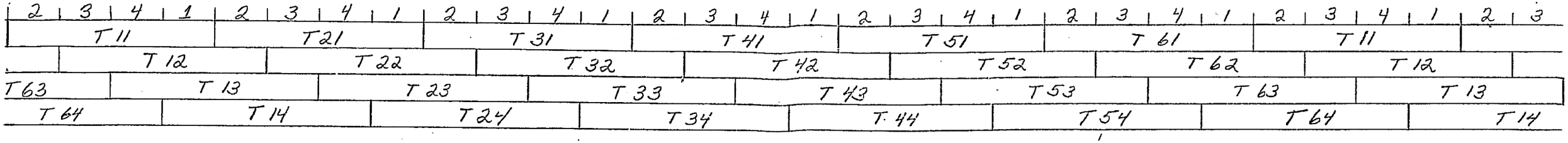


A SEQUENCE — INCREMENT P BY 1, INSTRUCTION TO U

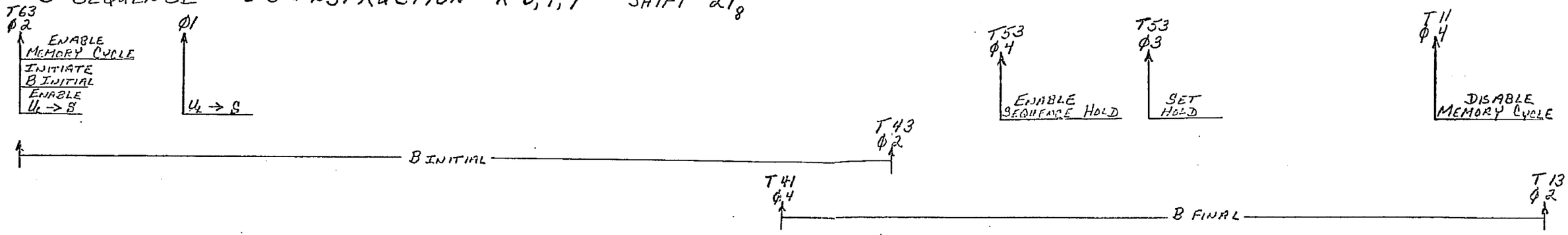


A AND A^b SEQUENCE TIMING CHART

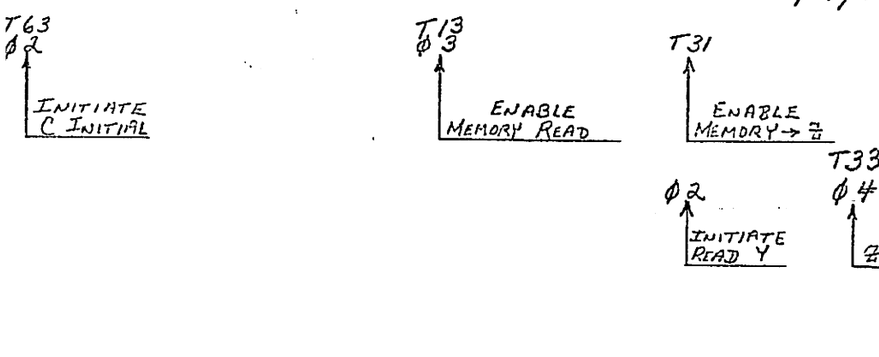
3-12-65 JCH



B SEQUENCE – 03 INSTRUCTION k $\overline{0,4,7}$ SHIFT 21_8



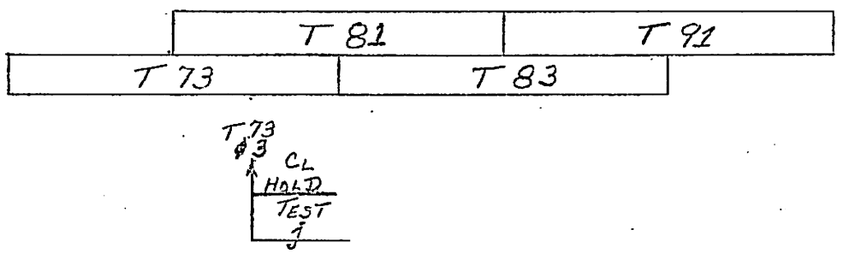
C SEQUENCE – 03 INSTRUCTION k $\overline{0,4,7}$



THE SHIFT LOGIC WILL NOT ALLOW C SEQUENCE EXTENSION UNTIL K1=0

TEST j (POSITIVE RESULTS) – LOGIC WILL BE ESTABLISHED TO INITIATE AN "A" SEQUENCE AND INHIBIT ALL FOLLOWING SEQUENCES (A^b, A^j, B, ETC.). AT THE END OF THE NEXT A SEQUENCE (T63 φ3) THE SEQUENCE INHIBIT (ABORT) IS CLEARED. THE NEXT SEQUENCE WILL BE ANOTHER A SEQUENCE TO INITIATE THE PROPER SEQUENCE CHAIN. THE FIRST A SEQUENCE WILL PLACE THE INSTRUCTION TO BE SKIPPED IN U, BUT IT WILL NOT BE EXECUTED SINCE AN INHIBIT HAS BEEN PLACED ON THE FOLLOWING SEQUENCES. THE SECOND A SEQUENCE WILL PLACE THE NEXT INSTRUCTION TO BE EXECUTED IN U AND SINCE THE SEQUENCE ABORT WAS CLEARED AT THE END OF THE FIRST A SEQUENCE, NORMAL INSTRUCTION EXECUTION IS ALLOWED.

TEST j (NEGATIVE RESULTS) – NORMAL SEQUENCING WILL OCCUR



B, C AND EXTENDED C SEQUENCE TIMING CHART