— BILL FLANAGAN

1219 COMPUTER
MAINTENANCE
STUDENT STUDY GUIDE

13 JANUARY 1967

PX 3814-0-1

Prepared by: Defense Systems Training

UNIVAC
DIVISION OF SPERRY RAND CORPORATION | DEFENSE SYSTEMS DIVISION

LIST OF EFFECTIVE PAGES

PAGE NUMBER	CHANGE IN EFFECT	PAGE NUMBER	CHANGE IN EFFECT
Title	Original	5.24-1 thru 5.24-6	Original
ii thru viii	Original	5.25-1 thru 5.25-6	Original
1.1-1 thru 1.1-2	Original	5.26-1 thru 5.26-4	Original
2.1-1 thru 2.1-6	Original	6.1-1 thru 6.1-18	Original
3.1-1 thru 3.1-2	Original	6.2-1 thru 6.2-10	Original
3.2-1 thru 3.2-2	Original	6.3-1 thru 6.3-14	Original
3.3-1 thru 3.3-10	Original	6.4-1 thru 6.4-6	Original
4.1-1 thru 4.1-10	Original	6.5-1 thru 6.5-12	Original
5.1-1 thru 5.1-8	Original	6.6-1 thru 6.6-8	Original
5.2-1 thru 5.2-12	Original	6.7-1 thru 6.7-8	Original
5.3-1 thru 5.3-6	Original	7.1-1 thru 7.1-14	Original
5.4-1 thru 5.4-8	Original	7.2-1 thru 7.2-14	Original
5.5-1 thru 5.5-4	Original	7.3-1 thru 7.3-6	Original
5.6-1 thru 5.6-6	Original	7.4-1 thru 7.4-12	Original
5.7-1 thru 5.7-6	Original	7.5-1 thru 7.5-4	Original
5.8-1 thru 5.8-6	Original	7.6-1 thru 7.6-8	Original
5.9-1 thru 5.9-6	Original	8.1-1 thru 8.1-14	Original
5.10-1 thru 5.10-4	Original	8.2-1 thru 8.2-4	Original
5.11-1 thru 5.11-6	Original	8.3-1 thru 8.3-4	Original
5.12-1 thru 5.12-8	Original	8.4-1 thru 8.4-20	Original
5.13-1 thru 5.13-6	Original	8.5-1 thru 8.5-18	Original
5.14-1 thru 5.14-6	Original	8.6-1 thru 8.6-24	Original
5.15-1 thru 5.15-10	Original	8.7-1 thru 8.7-6	Original
5.16-1 thru 5.16-6	Original	8.8-1 thru 8.8-10	Original
5.17-1 thru 5.17-6	Original	8.9-1 thru 8.9-6	Original
5.18-1 thru 5.18-4	Original	8.10-1 thru 8.10-6	Original
5.19-1 thru 5.19-8	Original	8.11-1 thru 8.11-6	Original
5.20-1 thru 5.20-4	Original	8.12-1 thru 8.12-12	Original
5.21-1 thru 5.21-4	Original	8.13-1 thru 8.13-4	Original
5.22-1 thru 5.22-4	Original	8.14-1 thru 8.14-6	Original
5.23-1 thru 5.23-6	Original		

PREFACE

The purpose of this study guide is to provide a text for study of the maintenance of the UNIVAC® 1219 computer. The information herein supplements that which is presented in class and that contained in the UNIVAC® 1219 Technical Manual.

Training on the UNIVAC 1219 computer is necessarily complemented by operation descriptions, charts, block diagrams, simplified logic diagrams, and study questions. In this study guide most of these training aids are available for reference in the sequential order of course development. The use of this information is determined by the instructor.

Each section contains a list of references which are portions of the UNIVAC 1219 Technical Manual, PX 3316. If no reference is made to this material within the study guide text, the student should read the information at the completion of the study guide section.

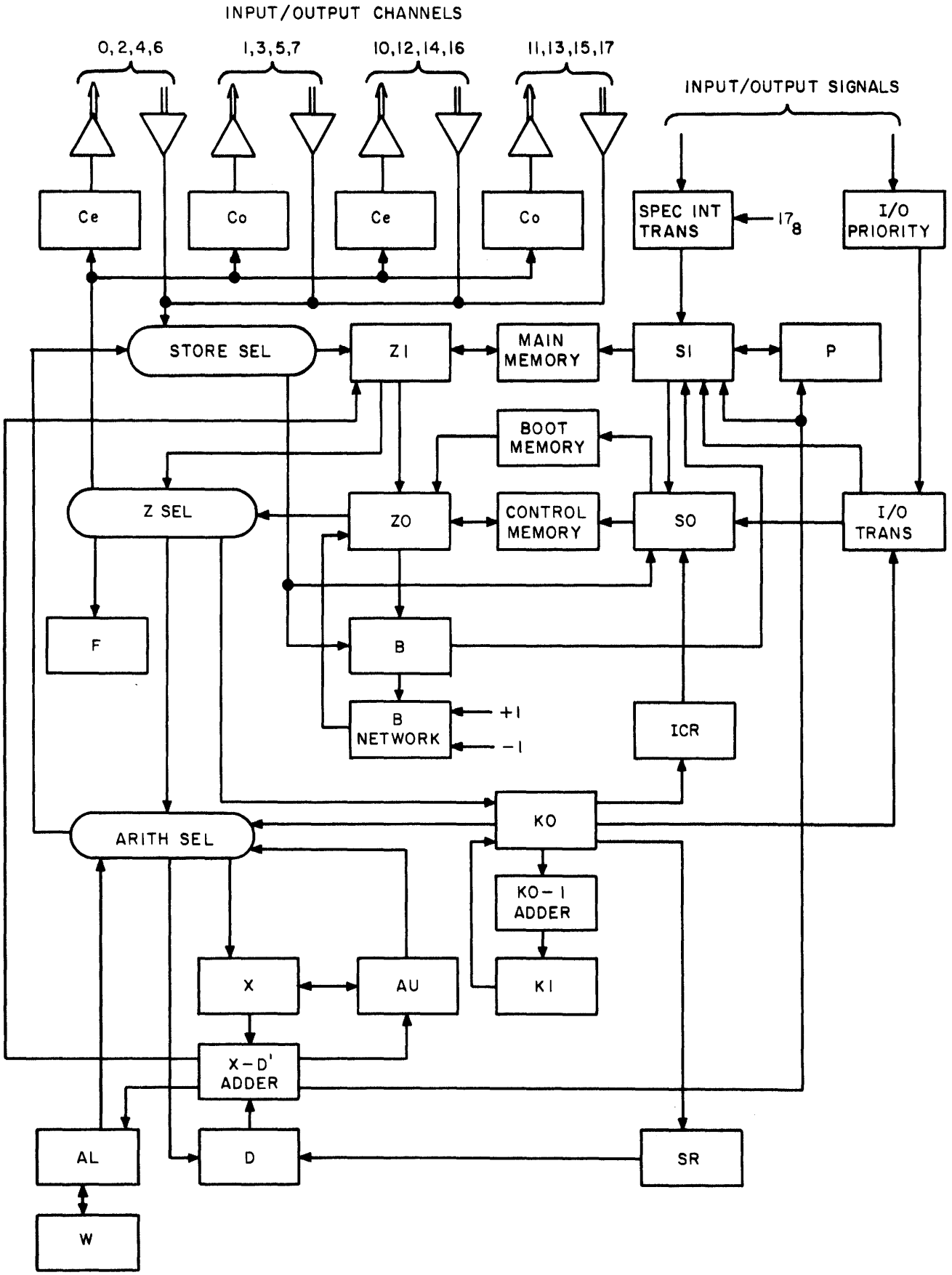
®- Registered Trademark of Sperry Rand Corporation

OUTLINE OF CONTENTS

SECTION 6 - ARITHMETIC SECTION

SECTION 7 - MEMORY SECTION

SECTION 8 - INPUT/OUTPUT SECTION



1219 Computer Block Diagram

SECTION 1 - GENERAL INTRODUCTION

1.1. GENERAL DESCRIPTION

1.1-1. OBJECTIVES

To present the general characteristics and descriptions of the computer sections and features of those sections.

1.1-2. INTRODUCTION

None.

1.1-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume I, Section 1.

1.1-4. INFORMATION

UNIVAC 1219 Technical Manual, Volume I, Section 1.

1.1-5. SUMMARY

None.

SECTION 2 - PROGRAMING

2.1. INSTRUCTION WORDS

2.1-1. OBJECTIVES

To present the instruction word format, interpretation of its designators, definition of terms specified by the instruction, and description of the instruction repertoire.

2.1-2. INTRODUCTION

The instruction word controls the machine operations to perform a predetermined function according to its bit configuration. There are 102_{10} different instructions.

2.1-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume I, Paragraphs 3-6a, b, and c.

2.1-4. INFORMATION

a. General Description. The instruction word is an 18-bit number which controls the computer operations to achieve some predetermined function. This function could be an arithmetic type such as addition or subtraction, a logical type such as logical multiplication (AND function), or a data transfer operation causing a binary word to be brought from memory and placed into a register. The operation to be performed is specified by the bit configuration (coding) of the instruction word.

An over-all computer operation such as navigation, target position prediction, etc., is performed by the sequential execution of selected instructions. This arrangement of instructions is referred to as a program. The instructions are stored in memory along with 18-bit data words. There are no restrictions imposed on the bit configuration of a data word and it may therefore be identical in content to an instruction word. Distinction between these word types is possible, however, because the addresses of the memory locations containing the instructions are kept track of in the P register. When a word is extracted from memory, the computer can distinguish an instruction from data because the time during which the instruction is obtained is different from that involving data word extraction from memory, and because the instruction is placed in a special register.

The memory origin of the instructions executed are controlled such that they are obtained from memory at sequential addresses. The addresses are formulated by incrementing the content of P by +1 after each instruction is obtained. Therefore, as an instruction is being executed, P holds the address of the next instruction.

b. Format 1 Instructions.

1. Instruction Word Configuration. The format 1 instructions are identified by the 6 most significant binary bits of the word forming an octal code of other than 50_8. These octal digits comprise the function code (also referred to as the f designator). Refer to figure 2.1-1 for the format 1 configuration.

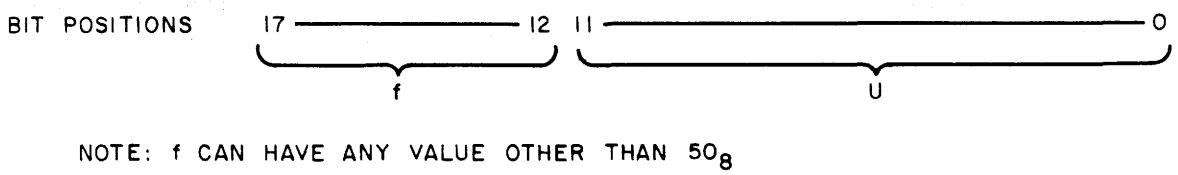


Figure 2.1-1. Format 1 Instruction Word Configuration

The function code specifies the operation to be performed. The lower 12 bits (U) are used as determined by the function code. If the operation requires a memory reference, U formulates a portion of the memory address. In other cases, U may be the actual number handled by the instruction. Regardless of the origin of this number (memory or U), the number is referred to as the operand.

2. Even-Numbered Function Codes or f > 50_8 Instructions.

a) Use of U as a Constant. There are only four format 1 instructions which use the U portion as the operand (also referred to as constant). These function codes are f = 36, 37, 70, 71_8. In order to provide an 18-bit operand value, these instructions extend the sign of U (U_{11}) into the upper 6 bits. This value, referred to as XU, is formulated in a register separate from that which holds the function code. Refer to figure 2.1-2 for the XU configuration.

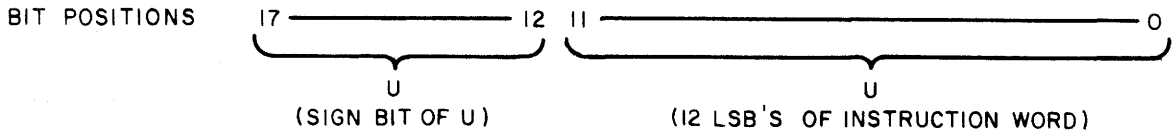


Figure 2.1-2. XU Configuration for f = 36, 37, 70, 71

As an example of the use of XU, consider the instruction word 704065_8. The function code of 70 (ENTALK) places the constant XU in the AL register. The initial AL content would be cleared and the XU value of 774065_8 would be placed in AL. No memory reference other than that necessary to obtain the instruction word is required.

b) Use of U as an Address. All format 1 instructions other than f = 36, 37, 70, 71 require a memory reference either to obtain an operand or to store a value into memory. U is used as the 12 least significant bits of the address. Sixteen bits are necessary to allow addressing for the complete $65,536_{10}$-word memory. The upper 4 bits are supplied from either the P register (P_{15-12}) or the special register ($SR_{4,2-0}$). The resulting address is referred to as either Up or U_{SR} depending upon which is used. The origin of these 4 bits is determined by the function code and SR_3. SR_3 is considered to be the SR active bit. If SR is inactive ($SR_3 = 0_2$), SR is not used. Refer to table 2.1-1 for the conditions necessary to use Up and U_{SR}.

TABLE 2.1-1. U_P AND U_{SR} CONDITIONS

CONDITIONS.	RESULTING ADDRESS*
$(SR_3 = 0_2)$ or (f = 30, 31, 34, 35, 51-67, 72-76	$U_P = \underbrace{15 \text{——} 12}_{P_{15-12}} \; \underbrace{11 \text{————} 0}_{U}$
$(SR_3 = 1_2)$ & (f = 01-27, 32, 33, 40-47	$U_{SR} = \underbrace{15 \text{——} 12}_{SR_{4,2-0}} \; \underbrace{11 \text{————} 0}_{U}$

* If interrupt sequence is in effect, upper 4 bits of address are 0's.

Address = $\underbrace{15 \text{——} 12}_{0's} \; \underbrace{11 \text{————} 0}_{U}$

All format 1 instructions other than those listed in Table 2.1-1 use XU as a constant. The interrupt sequence is discussed in a later sheet.

There are instructions which can be used to change the contents of SR to any configuration. SR_3 is not considered as an address bit; its only use is to specify the activeness of SR.

If P_{15-12} is used, the address used by the instruction will be in the same area of memory from which the instruction word itself is obtained. Realize that the instruction is obtained from the address held by P. P is advanced to the address of the next instruction while the instruction is being brought from memory. During the instruction execution, P holds the address from which the current instruction was obtained plus one. If the current instruction was extracted from address 017777_8 which is in bank 1, the upper 4 bits of Up will be 0010_2 which will reference bank 2. Thus, care must be taken in executing instructions from the last address of a bank.

3. Odd Numbered Function Codes and $f < 50_8$ Instructions. For instructions with odd-numbered function codes less than 50_8, the XU, Up, or U_{SR} value is modified. These function codes require the addition of one of the control memory B (index) registers to the address or constant formed with U (Up, USR, or XU). The B register to be used is specified by the 3-bit index control register (ICR). There are instructions which can be used to change the contents of ICR. ICR controls all

instruction references to a B register. The modifications of U_P + B, U_{SR} + B, and XU + B are performed in an end-around-carry manner.

c. Format 2 Instructions. The format 2 instructions are identified by the 6 most significant binary bits of the word forming the octal code of 50_8. Refer to figure 2.1-3 for the format 2 configuration.

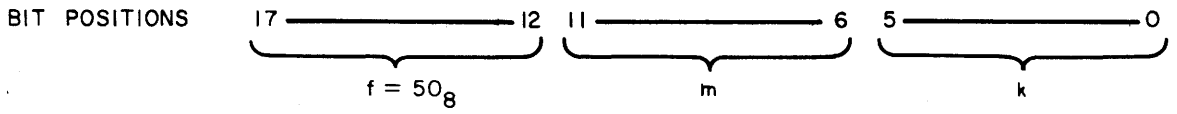


Figure 2.1-3. Format 2 Instruction Word Configuration

For these instructions, f only indicates the format 2 configuration. The m designator is the effective function code (sometimes referred to as minor function code). It specifies the operation to be performed and is often referred to as f. The m designator is placed in the same register as is f for format 1 instructions. The machine is able to determine which designator this register holds by inspecting the format 2 flip-flop. This flip-flop is set if the upper 6 bits of the instruction word indicate format 2 (50_8).

The remaining 6 bits comprise the k designator. The use of k is determined by m. Most of the format 2 instructions are of the input/output type, and k indicates the I/O channel number to be affected. In other cases, k represents a shift count and a value (constant) to be placed in ICR or SR. None of the format 2 instructions use any of the instruction bits to form a memory address.

d. Fault Function Codes. There are three format 1 function codes which cause an instruction fault condition. These are f = 00, 01, and 77_8. This fault condition interrupts the program and jumps the program to either address 00000 (control memory) or address 00500_8 (bootstrap) depending upon the position of the AUTOMATIC RECOVERY switch. This switch and its effect are discussed in a later sheet. There are other instructions which are illegal; however, they do not cause the fault interrupt condition.

e. Repertoire of Instructions (non-I/O Type). There are 3 basic groups of instructions. Some instructions require an operand from memory. Others store the contents of AU, AL, or B into memory. The remaining instructions do not use memory. Refer to table 2.1-2 for a listing of the non-I/O instructions. The important feature of this table shows the use of XU, U_P, U_{SR}, k and B modification.

Refer to the UNIVAC 1219 Technical Manual, paragraph 3-6c for a more exact description of the instructions. The input/output instructions are discussed in later sheets.

TABLE 2.1-2. 1219 COMPUTER REPERTOIRE OF INSTRUCTIONS (NON-I/O)

ENTER (Mem → Register)		
10	ENTAU	Address = U_P,
11	*ENTAUB	U_{SR}, U_P+B, or
12	ENTAL	U_{SR}+B
13	*ENTALB	
32	ENTB	
33	*ENTBB	
ENTER (non-memory)		
36	ENTBK	(XU → B)
37	*ENTBKB	(XU+B → B)
70	ENTALK	(XU → AL)
50:72	ENTICR	(k → ICR)
50:73	ENTSR	(k → SR)
STORE (registr → Mem)		
40	CL	0's → Mem at
41	*CLB	Address U_P,
42	STRB	U_{SR}, U_P+ B, or
43	*STRBB	U_{SR}+B
44	STRAL	Storage Address = U_P,
45	*STRALB	U_{SR}, U_P + B,
46	STRAU	or U_{SR} + B
47	*STRAUB	
72	STRICR	Storage ad-
74	STRADR	dress = U_P
75	STRSR	
ADD (Reg + Mem → register)		
14	ADDAL	Address = U_P,
15	*ADDALB	U_{SR}, U_P + B, or
20	ADDA	U_{SR} + B
21	*ADDAB	
ADD (AL + XU → AL)		
71	ADDALK	no memory

SUBTRACT (Reg - Mem → Reg)		
16	SUBAL	Address = U_P,
17	*SUBALB	U_{SR}, U_P+B,
22	SUBA	or U_{SR}+B
23	*SUBAB	
SELECTIVE (logical operation with Reg & Mem)		
04	SLSU	Address = U_P,
05	*SLSUB	U_{SR}, U_P + B,
51	SLSET	or U_{SR} + B
52	SLCL	Address = U_P
53	SLCP	
COMPLEMENT (Reg, no Mem)		
50:61	CPAL	
50:62	CPAU	k not used
50:63	CPA	
ROUND (Reg, no Mem)		
50:60	RND	k not used
MULTIPLY & DIVIDE (Reg & Mem)		
24	MULAL	
25	*MULALB	Address = U_P,
26	DIVA	U_{SR}, U_P + B
27	*DIVAB	or U_{SR} + B
SHIFT & SCALE FACTOR (no Mem)		
50:41	RSHAU	
50:42	RSHAL	
50:43	RSHA	k = shift count
50:44	**SF	
50:45	LSHAU	
50:46	LSHAL	
50:47	LSHA	

COMPARE (Reg & Mem)		
02	CMAL	Address =
03	*CMALB	U_P, U_{SR},
06	CMSK	U_P + B, or
07	*CMSKB	U_{SR} + B
SKIP		
56	BSK	Mem used,
57	ISK	Address = U_P
50:50	SKP	k = console skip key
50:51	SKPNBO	
50:52	SKPOV	
50:53	SKPNCV	k not used
50:54	SKPODD	
50:55	SKPEVN	
DIRECT JUMP		
34	JP	jump address = U_P
35	*JPB	or U_P + B
60	JPAUZ	
61	JPALZ	
62	JPAUNZ	
63	JPALNZ	jump address = U_P
64	JPAUP	
65	JPALP	
66	JPAUNG	
67	JPALNG	
73	BJP	
INDIRECT JUMP		
54	IJPEI	jump address in mem at address U_P
55	IJP	
DIRECT RETURN JUMP		
76	RJP	jump address = U_P+1
INDIRECT RETURN JUMP		
30	IRJP	jump add. =
31	*IRJPB	NO.+1 at U_P or U_P + B
STOP		
50:56	STOP	k = console stop key

NOTES: *Odd function codes less than 50_8 add B_{ICR} to U value.

**Scale factor instruction (f = 50:44) stores a shift count in a fixed memory location.

2.1-5. SUMMARY

There are 2 basic instruction types, Format 1 and Format 2. Format 1 instructions ($f \neq 50_8$) use the lower 12 bits (U) either as an address (U_P or U_{SR}) or an operand (constant, XU). Some Format 1 instructions specify modification of the value formed with U by the addition of B_{ICR}.

Format 2 instructions ($f = 50_8$) use the m designator as the actual function code. The lower 6 bits (k) is used as a constant.

SECTION 3 - OPERATION

3.1. CONTROLS AND INDICATORS

3.1-1. OBJECTIVES

To present the general description of the controls and indicators, their functions, and their uses.

3.1-2. INTRODUCTION

Although the computer is basically automatic, there are provisions for control of the computer. Indicators provide a means of monitoring the operations. Switches are used to control a part or all of the computer operation, to provide jump and stop conditions, and to govern the speed of operation for maintenance purposes. There are also pushbutton/indicators which monitor and allow the manual setting of flip-flops, and pushbuttons with which an entire register can be cleared.

3.1-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume I, Paragraphs 3-1 and 3-2.

3.1-4. INFORMATION

Refer to the UNIVAC 1219 Technical Manual, Volume I, paragraphs 3-1 and 3-2 for the general description of controls and indicators.

3.1-5. SUMMARY

None.

SECTION 3 - OPERATION

3.2. MANUAL INSTRUCTION EXECUTION

3.2-1. OBJECTIVES

To present the operating procedures involved with the manual execution of an instruction.

3.2-2. INTRODUCTION

For program loading, program debugging, and for maintenance purposes, control of the computer can be effected to cause the computer to execute one instruction which has been loaded into the F register.

3.2-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume I, Paragraphs 3-3a and b.

3.2-4. INFORMATION

a. General Description. Through the use of the function repeat operation in conjunction with the operation step mode, the operator can execute one instruction repeatedly at a manually controlled rate. The function code portion of the instruction must be set in F, and the address used by the instruction must be set in P. Each depression of the START STEP/RESTART switch causes one execution of the instruction. During each execution, P is incremented by +1 so as to create sequentially increasing addresses to be used by the instruction.

b. Operating Procedures. Refer to the UNIVAC 1219 Technical Manual, Volume I, paragraphs 3-3a and b. These paragraphs describe the procedures for manually executing the ENTAL (f = 12) and STRAL (f = 44) instructions. These instructions provide for reading from memory and writing into memory operations.

The procedures described in the above reference can be modified to allow the internal, low-speed restart oscillator to effect the stepping action. If the START STEP/RESTART switch is placed in the up (locked) position instead of the momentary down (spring loaded) position, each pulse from the restart oscillator causes the instruction to be executed once and thus simulates one manual depression of the switch. The rate of executions can be governed by the RESTART SPEED CONTROL potentiometer which varies the frequency of the oscillator from 2 to 200 pulses per second.

3.2-5. SUMMARY

None.

SECTION 3 - OPERATION

3.3. UTILITY PACKAGE I

3.3-1. OBJECTIVES

To present the capabilities and use of the Utility Package I service routines.

3.3-2. INTRODUCTION

Utility Package I contains operator service routines used to perform memory loads and dumps in various formats and to change the contents of memory locations by computer panel control.

3.3-3. REFERENCES

None.

3.3-4. INFORMATION

a. General Description. Operator service routines are those routines which perform handling service to the user; however, they do not become integrated into his program. Utility Package (UPAK) I is a paper tape utility package which loads assembled program tapes and makes memory dumps on paper tapes. The package provides other console conveniences which are inspect-and-change memory cell contents and store constant into memory.

UPAK I is a collection of seven subroutines punched on paper tape in relocatable bioctal format. These subroutines provide paper tape input/output functions and examination and alteration of memory for program debugging. The following is a listing of the UPAK I subroutines.

1. Load Absolute Flexowriter/Field Data/ASCII Code*
2. Load Absolute Bioctal Code
3. Load Relocatable Bioctal Code**
4. Dump Absolute Flexowriter/Field Data/ASCII Code*
5. Dump Absolute Bioctal Code

*This type of code is determined by the type of paper tape punch (Flexowriter, 1232 I/O Console (field data), or 1532 I/O Console (ASCII) that the particular UPAK I specifies.

**Relocatable (relative) bioctal code is generated by the TRIM Assemblers.

6. Inspect and Change

7. Store Constant in Memory

UPAK I may be loaded anywhere in computer memory above address 01000_8 with the single restriction that the entire package must be contained within a single memory bank. The paper tape load and dump routines are operable under program or manual control; the inspect-and-change and store-constant-in-memory functions are operable from the computer control panel only.

UPAK I has been designed for computers with a maximum of 32,768 storage locations. For this reason, the dump and load subroutines (with the exception of load relocatable bioctal code) cannot be used to either load or dump a tape above address 77777_8. Those installations possessing larger than a 32,768-location computer should use UPAKM for the full addressing capability.

UPAK I occupies $1,031_8$ memory locations, exclusive of addresses 00540_8 through 00777_8 which are used by the UPAK I loader. Entrance addresses to the seven subroutines are assigned relative to the UPAK I base address. The base address is specified by the operator prior to loading UPAK I. This operation is discussed later in this sheet. Refer to table 3.3-1 for the subroutine starting addresses.

TABLE 3.3-1. UPAK I SUBROUTINE ENTRANCE ADDRESSES

ADDRESS	SUBROUTINE ENTRANCE
Base*+ 0	Program entrance to paper tape load**
Base + 2	Program entrance to dump absolute Flexowriter/field data/ASCII code
Base + 4	Program entrance to dump absolute bioctal code
Base + 6	Manual entrance to paper tape load**
Base + 7	Manual entrance to dump absolute Flexowriter/field data/ASCII code
Base + 10_8	Manual entrance to dump absolute bioctal code
Base + 11_8	Manual entrance to inspect and change
Base + 12_8	Manual entrance to store constant in memory

NOTES: *Base refers to the base address of UPAK I in memory.
**These paper tape operations are for all formats:
Absolute Flexowriter/field data/ASCII
Absolute bioctal
Relocatable (relative) bioctal

UPAK I subroutines use the currently active B register, but they store and restore its original value. Subroutines entered under program control also store and restore the Special Register (SR) value used by the program.

b. Description of Subroutines and Formats

1. Load Absolute Flexowriter/Field Data/ASCII Code. A particular UPAK I can accept one of these three codes, depending on the type of UPAK I.

Refer to figure 3.3-1 for the input tape format.

A carriage return followed by either an 8 or an 88 and another carriage return activates the load subroutine. A single 8 indicates the input tape has no checksum; an 88 indicates that the input tape has a checksum. The subroutine ignores any data which may precede either of these two combinations. Each computer instruction word on tape is preceded by a five-digit address which need not be in sequential order; all five digits must be present. The next six digits constitute the instruction word to be loaded; all six digits must be present. The load subroutine, in effect, accumulates the first 11 octal digits following a carriage return as address and instruction, and ignores all other character codes including notes. A carriage return signals the end of the instruction. If less than 11 octal digits are accumulated, the instruction will not be loaded. A final carriage return followed by a double period (..) terminates the load and if required, initiates a check verification.

When the checksum verification is correct, the load terminates with AU and AL both containing 0's. When it is incorrect, the load subroutine terminates with AU = computed checksum and AL = tape checksum.

If Skip Key 1 is set, the load subroutine will perform a checksum verification without loading the tape into memory.

No Checksum	Checksum
*8	*88
*AAAAA IIIIII	*AAAAA IIIIII
*AAAAA IIIIII	*AAAAA IIIIII
*AAAAA IIIIII	*AAAAA IIIIII
*AAAAA IIIIII	*AAAAA IIIIII
*..	*..
	*CCCCCC

NOTES: * indicates carriage return. For 1232 and 1532 I/O Consoles (field data and ASCII codes, respectively), line feed is required after carriage return.
A indicates octal character address digit.
I indicates octal character instruction or constant digit.
C indicates octal character checksum digit.

Figure 3.3-1. Flexowriter/Field Data/ASCII Code Format for Input via UPAK I As It Appears Printed When Punched on Tape

When the checksum verification is correct, the load terminates with AU and AL both containing 0's. When it is incorrect, the load subroutine terminates with AU = computed checksum and AL = tape checksum.

If Skip Key 1 is set, the load subroutine will perform a checksum verification without loading the tape into memory.

2. Load Absolute Bioctal Code. Refer to figure 3.3-2 for the input tape format.

The absolute bioctal tape must begin with a 76 (code for absolute bioctal tape). Immediately after the 76 code are the initial and final addresses consisting of five digits each. Six-digit instruction words follow without further addressing, and the tape is terminated by a six-digit checksum.

If Skip Key 1 is set, the absolute bioctal load subroutine will perform a checksum verification without loading the tape into memory.

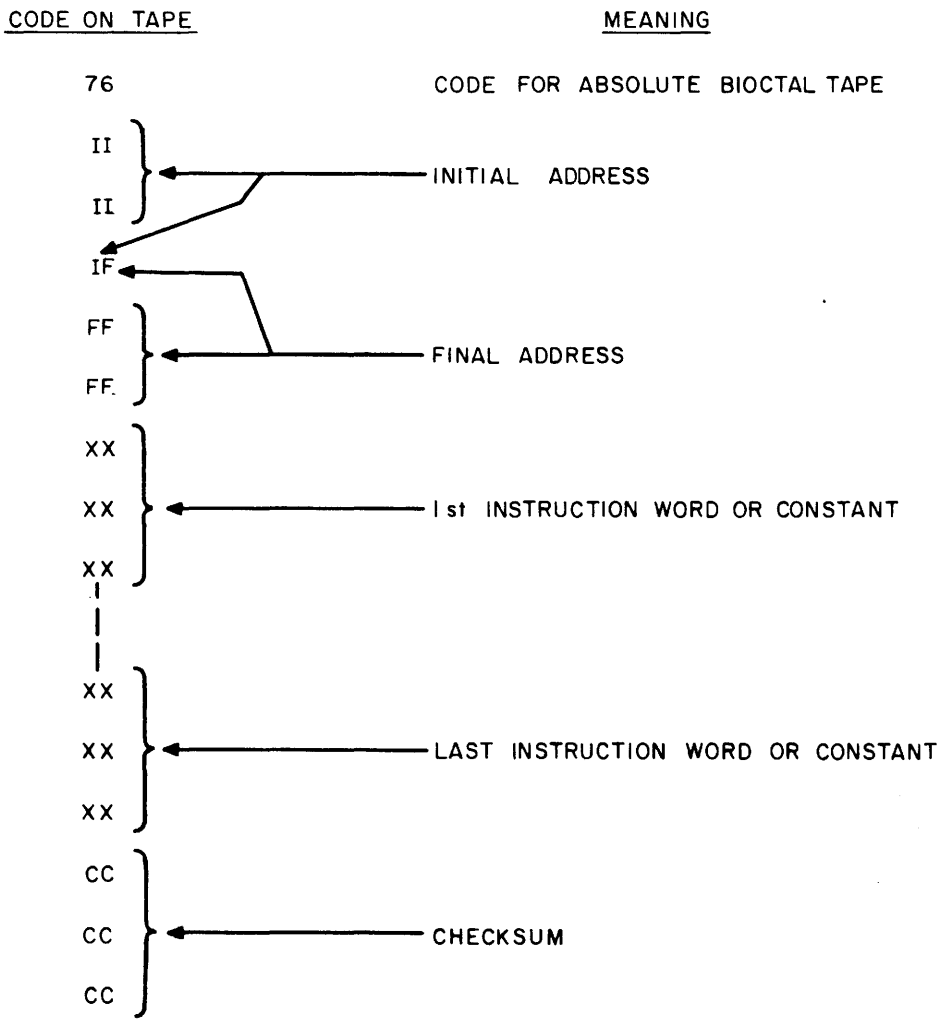


Figure 3.3-2. Absolute Bioctal Code Format for Input via UPAK I
As It Appears On Punched Tape

3. Load Relocatable (Relative) Bioctal Code. Refer to figure 3.3-3 for the input tape format. The relocatable bioctal tape must begin with a 75 (code for relocatable bioctal tape). The entire program must be relative to base zero. Six-digit instructions follow the 75 code without addressing. Each instruction is preceded by a one-digit code which specifies to the subroutine how to modify the instruction for storage. The tape is terminated by a six-digit checksum preceded by a code of 7. The computer operator specifies the load base address in AU; the load subroutine then uses this information to accomplish the tape load. The tape can be loaded anywhere in computer memory.

Refer to table 3.3-2 for the interpretations of the modification code.

This tape format is generated by the TRIM assemblers. If Skip Key 1 is set, the relocatable bioctal load subroutine will perform a checksum verification without loading the tape into memory.

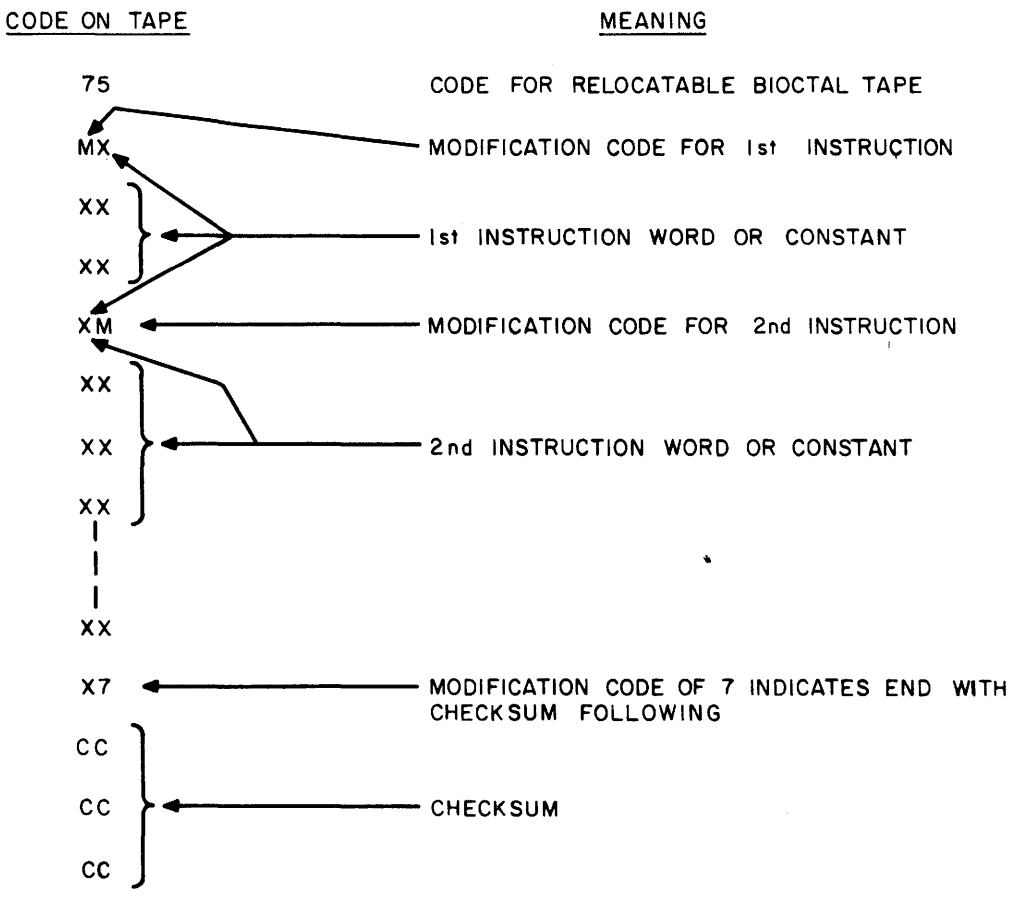


Figure 3.3-3. Relocatable Bioctal Code Format For Input via UPAK I As It Appears On Punched Tape

TABLE 3.3-2. RELOCATABLE BIOCTAL MODIFICATION CODE INTERPRETATION

CODE	MEANING
0	No modification of following instruction word or constant.
1	Add base address to bits 11-0 of following instruction word or constant.
2	No modification of following instruction word or constant.
3	Add base address to bits 14-0 of following instruction word or constant.
4	Add to current load address the following constant (creates a break in the storage addresses; i.e., the next instruction will be stored at the current address plus the constant following the 4 code).
5,6	Not used.
7	Following is checksum.

4. Dump Absolute Flexowriter/Field Data/ASCII Code. Depending on the type of UPAK I, the dump is in one of these three codes. The dump is initiated manually at the computer or under program control for output on punched paper tape. The 88 format (with checksum at the end) is the only one dumped. The output tape includes both the addresses and the contents of memory location being dumped. The format is that shown in figure 3.3-1.

5. Dump Absolute Bioctal Code. The absolute bioctal dump is initiated manually at the computer or under program control. The output on punched paper tape is the 76 code followed by the initial and final addresses, the contents of the inclusive memory locations, and the checksum. The format is that shown in figure 3.3-2. If more than one program area is dumped successively on the same tape, the format for each such area is as just described.

6. Inspect and Change. The inspect-and-change subroutine causes the contents of the memory location specified in AU to be displayed in AL. The content of AL may then be changed manually. The content of AL is then returned to the memory address from which it was taken. The inspection address must only be entered the first time since the content of AU is incremented by 1, and the content of the next sequential address is brought into AL with each successive performance of the inspect-and-change function. If the operator wishes to inspect the content of some addresses other than the next sequential address, he may do so by setting the new address in AU before returning AL into memory.

7. Store Constant in Memory. The store-constant-in-memory function permits the operator to load a specified area of memory with a value manually entered into AU. If AU = 0, the area is cleared.

c. Operating Procedures.

1. Loading UPAK I. UPAK I is provided on punched paper tape. The tape is subdivided into 2 parts, the loader which is in absolute bioctal format and

UPAK I which is in relocatable bioctal format. The loader itself performs a relocatable bioctal load which loads UPAK I. The loader is brought into memory by bootstrap. UPAK I loading procedure is as follows.

STEP 1. Place UPAK I tape in reader.

STEP 2. Master clear computer.

STEP 3. Press LOAD button to activate paper tape bootstrap.

STEP 4. Start computer (depress START STEP/RESTART switch). AU and AL should contain 0's.

STEP 5. Set AU to the desired UPAK I base address.

STEP 6. Start computer (depress START STEP/RESTART switch). The computer will stop after UPAK I is in memory. AU and AL should contain 0's.

After UPAK I has been loaded into memory, addresses 00540_8 through 00777_8, which are occupied by the loader for UPAK I, are available for other use. It should be noted, however, that any subsequent loading of UPAK I will use these addresses to accomplish the load.

2. Error Detection. Automatic checksum verification by the paper tape load subroutines is the only error detection performed by UPAK I. If the tape checksum and the checksum formulated from the loaded information agree, the computer will perform a normal stop with AU and AL both containing 0's. If they do not agree, the computer will stop with AU = computed checksum and AL = tape checksum.

Checksum verification is also performed on the loader for UPAK I. Immediately after this loader has been inputted, control is given to a temporary checksum verification subroutine which is part of the inputted loader. This subroutine verifies the bioctal entry of the loader.

3. Use of UPAK I. Paragraphs 3.3-4c3a) through 3.3-4c3e) list the procedures of operation for the UPAK I subroutines. These procedures assume UPAK I to be in memory.

a) Paper Tape Load (All Formats).

1) Manual Initiation.

STEP 1. Place tape in reader.

STEP 2. Set P to UPAK I base address +6.

Step 3. Set Skip Key 1 if checksum verification only is required.

Step 4. For relocatable bioctal load only, set starting address in AU. (Not required if Skip Key 1 is set.)

Step 5. Start computer. (Depress START STEP/RESTART switch.) Computer will stop after load. AU and AL should contain 0's. Successive tapes may be loaded without resetting P.

2) Program Initiation.

STEP 1. Place tape in reader.

STEP 2. For relocatable bioctal load only, enter AU with starting address.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address.

b) Dump Absolute Flexowriter/Field Data/ASCII Code.

1) Manual Initiation.

STEP 1. Set P to UPAK I base address +7.

STEP 2. Set AU to first address to be dumped.

STEP 3. Set AL to last address to be dumped.

STEP 4. Start computer. (Depress START STEP/RESTART switch.) Computer will stop after dump. Successive dumps may be performed by starting at Step 2.

2) Program Initiation.

STEP 1. Enter AU with first address to be dumped.

STEP 2. Enter AL with last address to be dumped.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address +2.

c) Dump Absolute Bioctal Code.

1) Manual Initiation.

STEP 1. Set P to UPAK I base address $+10_8$.

STEP 2. Set AU to first address to be dumped.

STEP 3. Set AL to last address to be dumped.

STEP 4. Start computer. (Depress START STEP/RESTART switch.) Computer will stop after dump. Successive dumps may be performed by starting at Step 2.

2) Program Initiation.

STEP 1. Enter AU with first address to be dumped.

STEP 2. Enter AL with last address to be dumped.

STEP 3. Execute a return jump or indirect return jump to UPAK I base address +4.

d) Inspect and Change (Manual Initiation Only).

STEP 1. Set P to UPAK I base address $+11_8$.

STEP 2. Set AU to desired memory address.

STEP 3. Start computer. (Depress START STEP/RESTART switch.) Computer will stop with the address in AU and its contents in AL. The address in AU and/or the contents in AL may now be changed.

STEP 4. Start computer. (Depress START STEP/RESTART switch.)

If contents only were altered, they will be stored at the original address and computer will stop with the next sequential address in AU and its contents in AL.

If the address only was changed, the contents of AL will be restored to their proper memory location and computer will stop with the new address in AU and its contents in AL.

If both the address in AU and its contents in AL were changed, the contents of AL will be stored at the original address and computer will stop with the new address in AU and its contents in AL.

Any number of such sequences may be executed starting with Step 4.

e) Store Constant in Memory (Manual Initiation Only).

STEP 1. Set P to UPAK I base address $+12_8$.

STEP 2. Set first storage address in AU.

STEP 3. Set last storage address in AL.

STEP 4. Start computer. (Depress START STEP/RESTART switch.) UPAK I will record these addresses and computer will stop with AU cleared.

STEP 5. Set desired constant in AU.

STEP 6. Start computer. (Depress START STEP/RESTART switch.) UPAK I will store the contents of AU at successive memory locations within, and including, the limits established in Steps 2 and 3. Additional entrances may be made starting at Step 2.

3.3-5. SUMMARY

None.

SECTION 4 - LOGIC INTRODUCTION

4.1. LOGIC CIRCUITS, SYMBOLOGY, AND COMPONENT NOTATION

4.1-1. OBJECTIVES

To present the logic symbology of the 1219, a brief analysis of electronic logic gate operation, and use of the logic notation.

4.1-2. INTRODUCTION

The electronic diagrams for the 1219 use logic symbols for each circuit. The student needs only to be able to interpret the logical function indicated by each symbol. With the exception of some of the memory circuitry, all circuitry will be discussed in its symbol form after this sheet.

4.1-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Section 8.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

4.1-4. INFORMATION

a. Logic Levels. The two internal logic voltage levels are referred to as high (H) and low (L). The high level is 0 volts, ground. The low level is -4.5 volts. Except for the memory section, all decision-making gates operate with these levels. If a low level is needed to satisfy a particular logical function, a flag is drawn on the input line. If a high level is needed, there is no flag.

In the same manner, the output of the logical function indicates the logic level present if the function is satisfied. The presence of the flag on the output lead implies a low level exists if the function is satisfied. No flag indicates a high level. Refer to figure 4.1-1 for the flag notation.

b. AND Function.

1. General Description. The AND logical function is performed by a circuit referred to as an AND gate. This gate is identified by a special symbol. To satisfy an AND gate, all of the inputs must be present at the indicated levels (flag notation). When the gate is satisfied, the indicated output level is present. Refer to figure 4 1-2 for the four basic AND configurations and truth tables.

These examples only use two inputs. There can be many more inputs. Regardless of the number of inputs, they must all be at the indicated level to satisfy the AND gate.

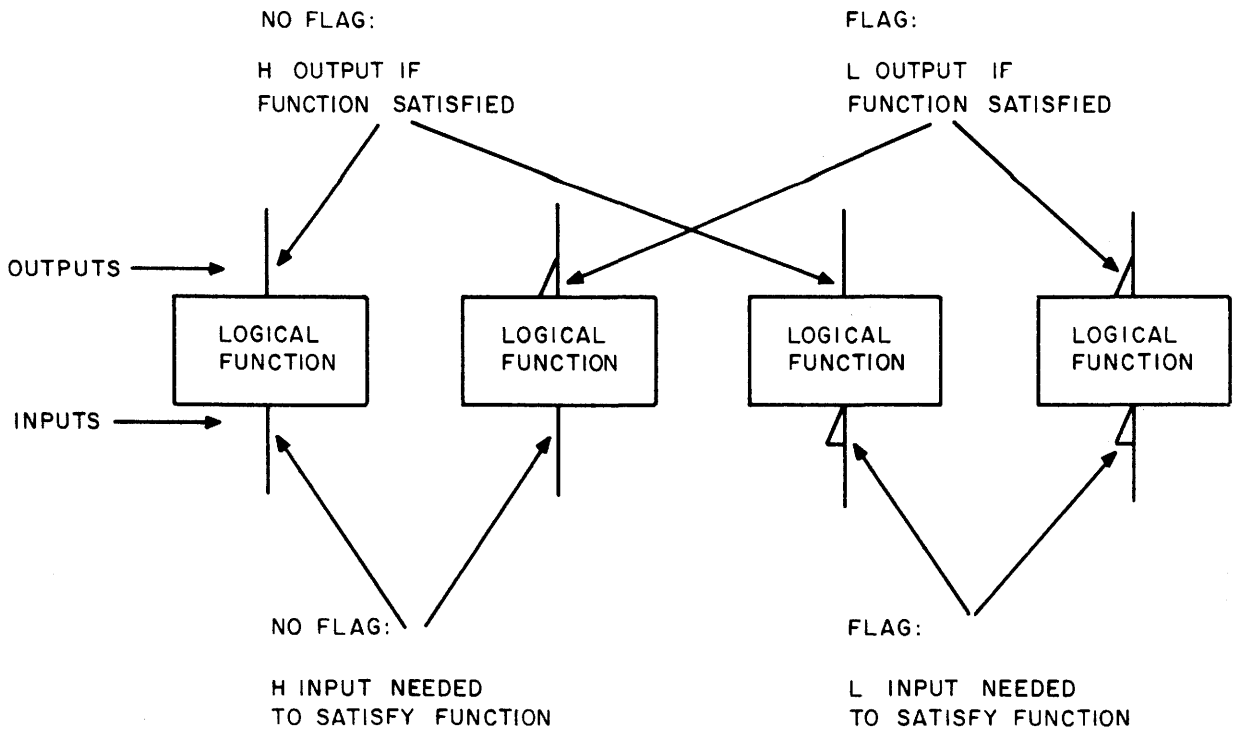


Figure 4.1-1. Illustration of Flag Level Notation

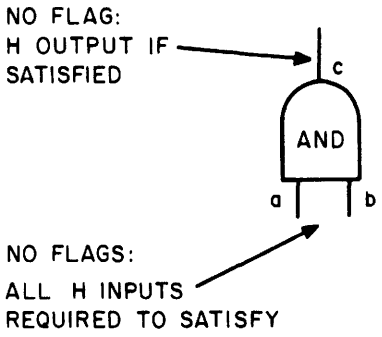
2. Detailed Analysis. AND gate c of figure 4.1-2 is used for an example. Refer to figure 4.1-3 for the schematic.

Current flows from the -15 volt supply through the 2.2k, 470 ohm, and 5.6k resistors to the +15 volt supply. A voltage of the polarity shown is developed across the 470 ohm resistor. If both inputs A and B are at low levels, the voltage applied to the transistor base with respective to ground (emitter) is negative which forward biases the transistor. As a result, current flows from the -4.5 volt supply through the collector to emitter to ground. In this condition, the output is at ground level due to the low internal resistance of the collector to emitter path.

If either the A or the B input (or both) is at a high level (ground), current flows from the -15 volt supply through the 2.2k resistor and input diode. Because of the low internal resistance of the input diode, an effective ground level is present on the left side of the 470 ohm resistor. The voltage drop across this resistor (of the polarity shown) then becomes the transistor bias which cuts off the transistor. Depending upon the load, approximately -4.5 volts is present at the output.

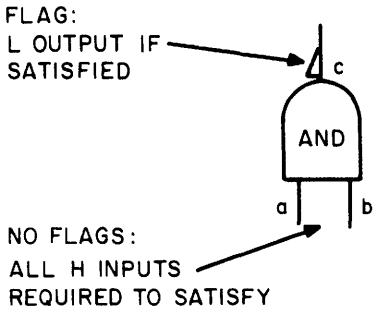
c. OR Function.

1. General Description. The OR logic function is performed by a circuit referred to as an OR gate. This gate is identified by a special symbol. To satisfy an OR gate, at least one of the inputs must be present at the indicated level. When the gate is satisfied, the indicated output level is present. An OR gate can be drawn as an equivalent AND gate. Refer to figure 4.1-4 for the four basic configurations, truth tables, and equivalent AND gates.



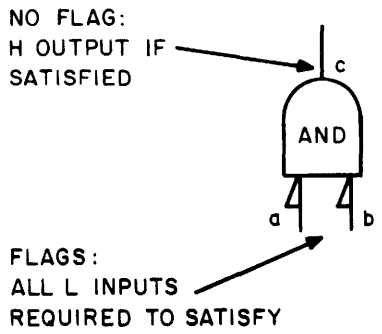
INPUTS		OUTPUT
a	b	c
L	L	L
L	H	L
H	L	L
H	H	H

a. HIGH INPUTS, HIGH OUTPUT



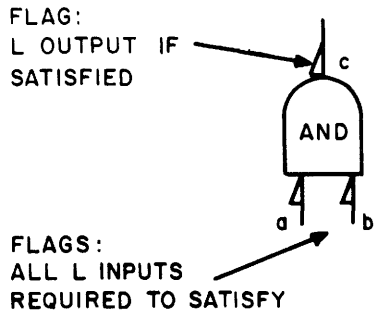
INPUTS		OUTPUT
a	b	c
L	L	H
L	H	H
H	L	H
H	H	L

b. HIGH INPUTS, LOW OUTPUT



INPUTS		OUTPUT
a	b	c
L	L	H
L	H	L
H	L	L
H	H	L

c. LOW INPUTS, HIGH OUTPUT



INPUTS		OUTPUT
a	b	c
L	L	L
L	H	H
H	L	H
H	H	H

d. LOW INPUTS, LOW OUTPUT

Figure 4.1-2. Basic AND Gates

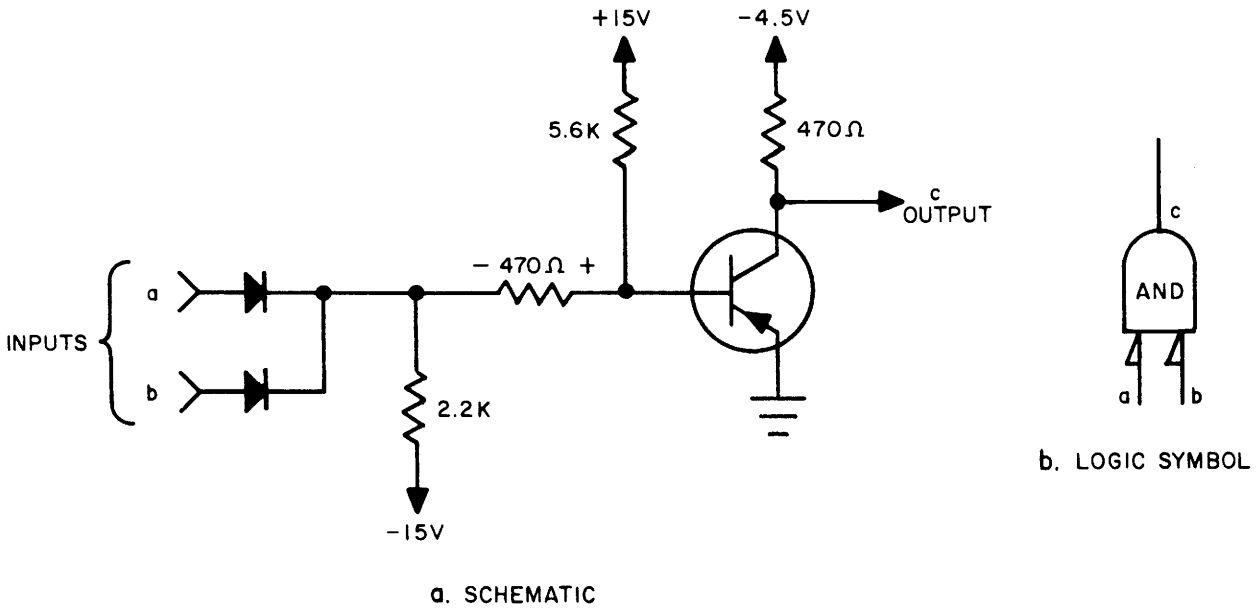


Figure 4.1-3. Low Inputs, High Output AND Gate

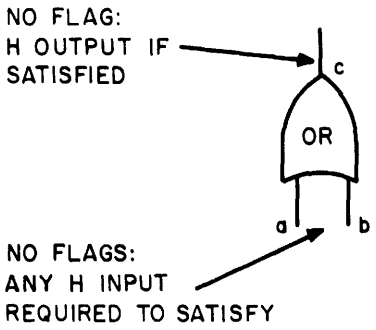
These examples use only two inputs; there can, however, be many more inputs. Regardless of the number of inputs, at least one of them must be at the indicated level to satisfy the OR gate.

2. Detailed Analysis. OR gate c of figure 4.1-4 is used for an example. Refer to figure 4.1-5 for the schematic.

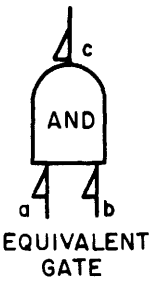
Current flows from the -15 volt supplies through the 470 ohm resistor to the +15 volt supply. A voltage of the polarity shown is developed across the 470 ohm resistor. If both inputs A and B are at high levels (ground), current flows from the -15 volt supplies through the 2.4k resistors and the input diodes. This ground level is applied to the left side of the 470 ohm resistor which causes its voltage drop to be the transistor bias (base to emitter). The transistor is cut off and the -4.5 volt supply is felt as the output.

If either input is at a low level, that low level is applied to the left side of the 470 ohm resistor. If a ground is applied to the other input, its coupling diode to the 470 ohm resistor is reverse biased and cut off. The resulting negative voltage applied to the transistor forward biases it to cause collector-to-emitter current flow from the -4.5 volt supply. The output is a ground level because of the relatively low internal resistance of the transistor.

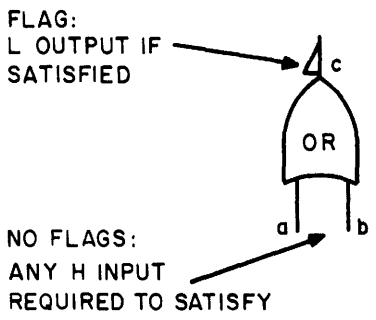
d. AND/OR Combination Function. AND and OR circuits can be combined into one gate. Refer to figure 4.1-6 for an example.



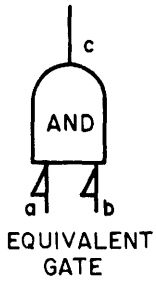
INPUTS		OUTPUT
a	b	c
L	L	L
L	H	H
H	L	H
H	H	H



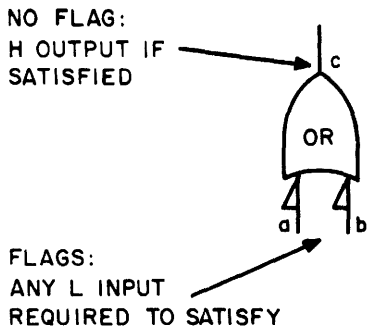
a. HIGH INPUTS, HIGH OUTPUT



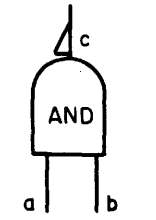
INPUTS		OUTPUT
a	b	c
L	L	H
L	H	L
H	L	L
H	H	L



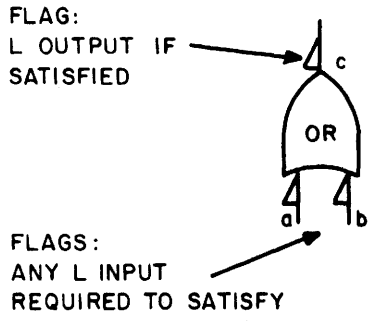
b. HIGH INPUTS, LOW OUTPUT



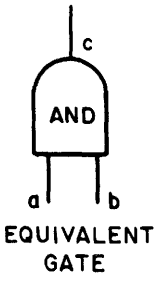
INPUTS		OUTPUT
a	b	c
L	L	H
L	H	H
H	L	H
H	H	L



c. LOW INPUTS, HIGH OUTPUT



INPUTS		OUTPUT
a	b	c
L	L	L
L	H	L
H	L	L
H	H	H



d. LOW INPUTS, LOW OUTPUT

Figure 4.1-4. Basic OR Gates

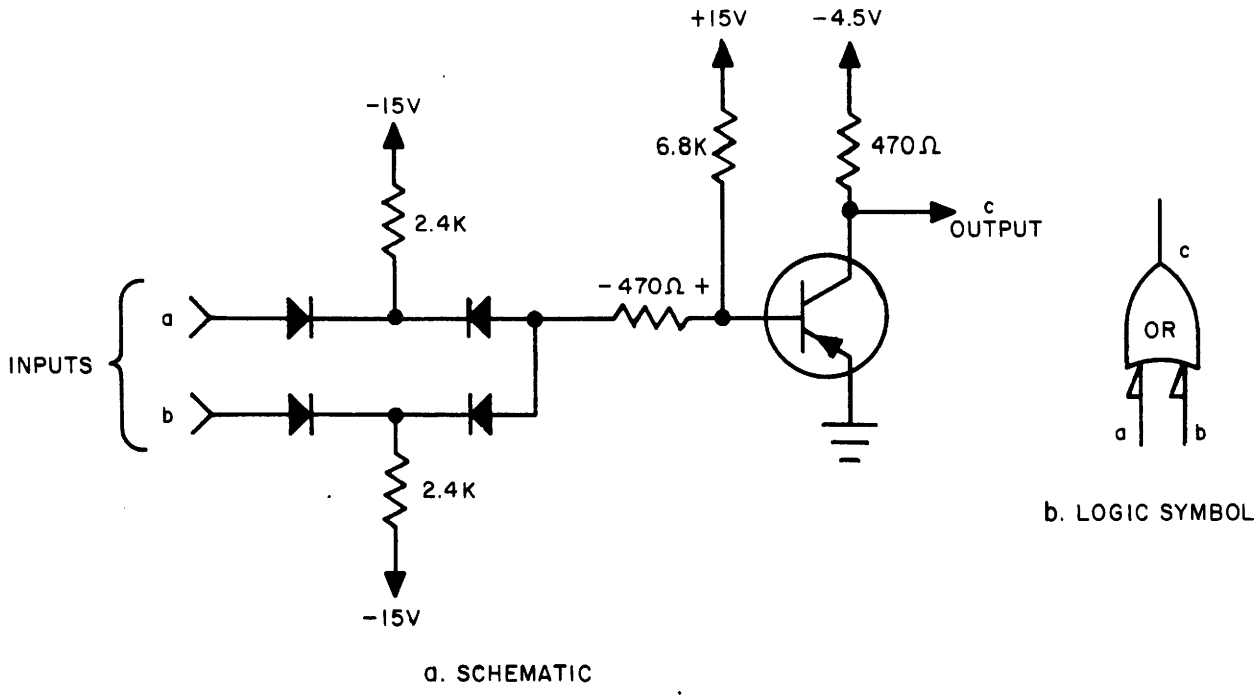


Figure 4.1-5. Low Inputs, High Output OR Gate

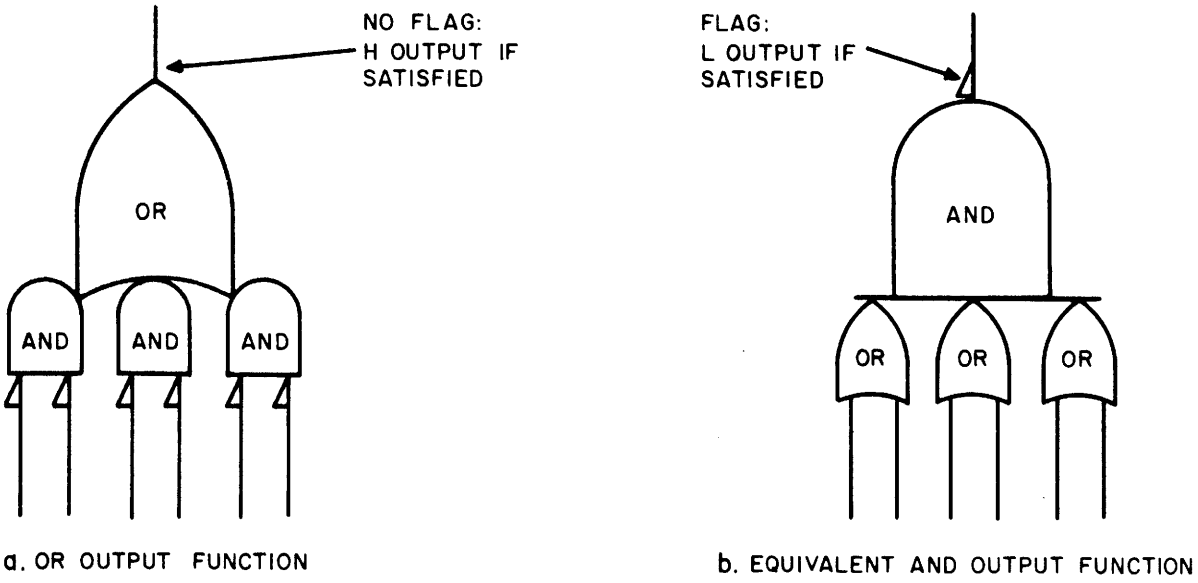


Figure 4.1-6. AND/OR Combination Gate

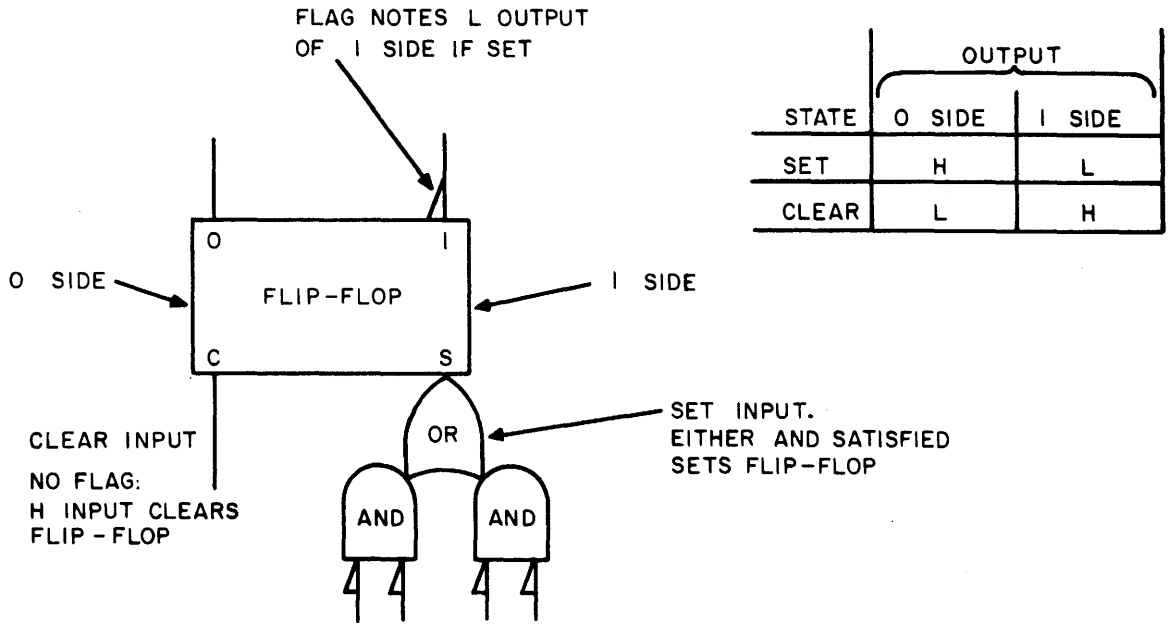
Both logic symbols represent the same circuits. Circuit A performs an OR function of the AND gate inputs. That is, only one input AND gate must be satisfied (both inputs at low levels) to satisfy the OR function and produce that high level output.

Circuit B performs the same logical operation as circuit A; however, if B is satisfied (low level output), A is not satisfied (low level output). In circuit B, the input gates are represented as OR functions. All input OR gates must be satisfied (one input at high level for each gate) to satisfy the output AND function.

The input AND/OR gates are comprised of diode circuitry.

e. Flip-Flop. The flip-flop is a bistable device. Its two stable states are referred to as set and clear. Flip-flops can be used as temporary storage elements since they can retain either of these states. The set/clear condition represents the information stored. Several flip-flops (register) can be used to hold a complete binary word. Refer to figure 4.1-7 for an example.

There are several variations concerning the set and clear inputs. If the clear input is flagged, a low level would be required to clear. The clear input may also have an AND gate such that the coincidence of several conditions may be necessary to perform the clearing. If the flip-flop is to hold a binary bit, it is usally cleared first (cleared state represents 0_2) and then set (set state represents 1_2). The output conditions for the two states hold true for any flip-flop, regardless of the set and clear input configuration. The 1 side output lead is always flagged to indicate a low level when set.



	OUTPUT	
STATE	0 SIDE	1 SIDE
SET	H	L
CLEAR	L	H

Figure 4.1-7. Flip-Flop

f. Component Notation. Each circuit has a unique reference number. The printed circuit card type number and card location within the machine also is noted. Refer to figure 4.1-8 for an example.

In most cases, the unique circuit number identifies the logic group with which the circuit is associated. In this example, the letter A indicates probable association with either the AU or AL register. If it does pertain to a selector or register (i.e., a device with which the gate might be associated with one particular bit position), the last two digits of the unique number indicate the bit position. Since AU is numbered with bit positions 235 through 218, this gate in all probability, pertains to AL_{00}.

The unique circuit number for a flip-flop is slightly different. Refer to figure 4.1-9 for an example.

All flip-flops have the "X" in their unique terms. In this example, the 1 and 0 sides are referred to by the terms 01A00 and 00A00, respectively. This particular flip-flop is stage 2^0 of the AL register.

Refer to UNIVAC 1219 Technical Manual, Volume I, Section 8-3 through 8-5 and logic diagrams, figure 9-1.

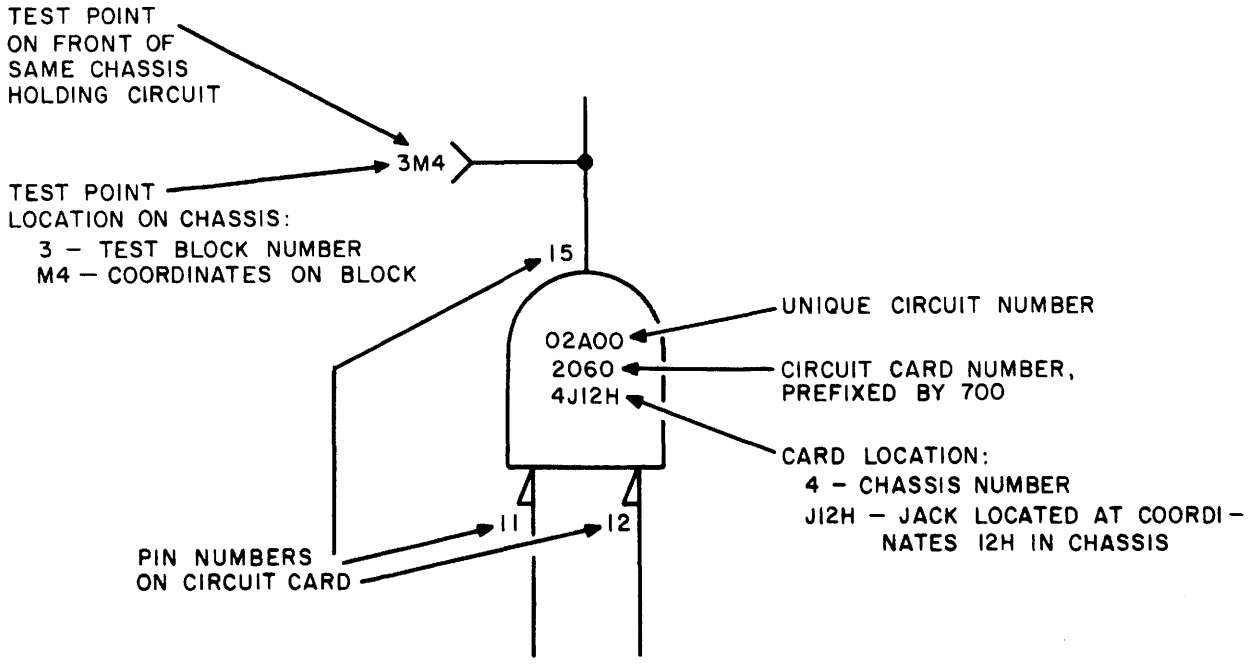


Figure 4.1-8. Logic Gate Notation

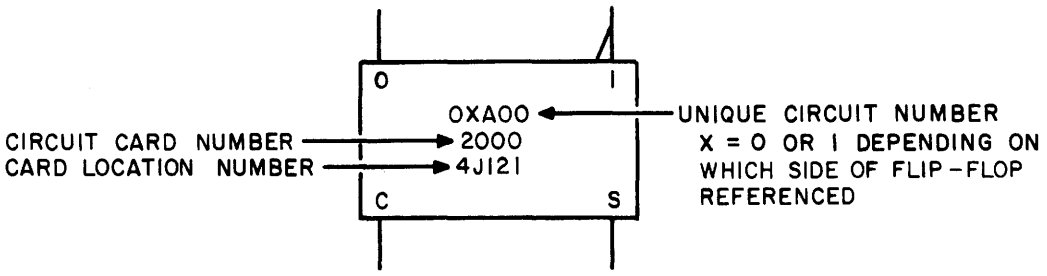


Figure 4.1-9. Flip-Flop Notation

4.1-5. SUMMARY

All circuitry is analyzed in the logic symbol form. The three basic circuits are the AND gate, the OR gate, and the flip-flop. Each logic symbol has a unique circuit number, circuit card type, and physical location code.

SECTION 5 - CONTROL SECTION

5.1. MASTER CLOCK, MODE CONTROL, PHASE STEP MODE, PHASE REPEAT

5.1-1. OBJECTIVES

To present the detailed theory of operation involved in the master clock, mode control, phase step mode, and phase repeat.

5.1-2. INTRODUCTION

The master clock is the source of all computer timing. Phase step mode and phase repeat allow the operation of the clock to be manually controlled for maintenance purposes. One of four modes can be manually selected to effect the desired operations.

5.1-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraphs 4-2a(5) and 4-2b(1), (2) and (4).

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.1-4. INFORMATION

a. Master Clock.

1. General Description. In order to effect the necessary data transfers, etc. under the strict assurance that they will be executed with a predetermined time relationship to other operations, the computer utilizes a common timing source. This source is the master clock. The basic component of the master clock is a free-running delay line oscillator which produces a symmetrical square wave.

The master clock output consists of four signals, referred to as phases. Each phase is enabled by one alternation of the delay line oscillator. These four phase outputs, represented by Ø1, Ø2, Ø3, and Ø4, provide timing to all computer operations. Clock cycle time (all four phases) is 500 nanoseconds.

2. Detailed Analysis. This analysis is of the clock operation when running at normal high speed. Manual control of the clock is described later in this sheet. Refer to logic diagrams, figure 9-4.

The basic component of the master clock is the oscillator comprised of the three delay lines shown. The oscillator will run freely if 24J06 outputs a constant high level. Except during phase mode, this high level exists because of the pin 5 input to 24J06.

If both inputs to 01DD01 are high levels, a low level is applied to the series of delay lines. One of the 01DD01 inputs is taken from a tap of delay 03DY01 such that after some time period, this input will become a low level. The resulting high level applied to the delay lines will eventually appear at the input to 01DD01 and cause its output to again go to a low level. This continuous action causes symmetrical square wave signals to appear at the delay line taps and at the output of 01DD01.

Because of the inherent circuit delay of 01DD01, each square wave alternation has a duration of approximately 125 nanoseconds. Each alternation is used to generate one clock phase. Thus, it requires two oscillator cycles to complete the 4-phase clock cycle.

Outputs of the square wave oscillator are used to develop the outputs of 63C13 and 63C24. A low level from 63C13 enables Ø1 and Ø3. A low level from 63C24 enables Ø2 and Ø4. These low levels occur alternately. Other enables for Ø generation are from flip-flops 7XC24 and 7XC13. These flip-flops change their set/clear configuration on every alternation of the square wave oscillator. The third set of enables for Ø generation are from 21J06, 21J07, 21J08, and 21J09. Their outputs are constant low levels if the Phase Mode is not in effect. The output of 71C14 is discussed later in this sheet. Refer to table 5.1-1 for a description of clock phase generation.

5.1-1. NORMAL MASTER CLOCK OPERATION

63C13	63C24	7XC24 ff	7XC13 ff	Generated Ø
L	H	Set	Clear	Ø1
H	L	Set	Set	Ø2
L	H	Clear	Set	Ø3
H	L	Clear	Clear	Ø4
L	H	Set	Clear	Ø1

Refer to the UNIVAC 1219 Technical Manual, Volume I, Section 4, figures 4-5 and 4-6 for a more exact timing description of the master clock operation.

For marginal check conditions, the pulse width of the phases can be varied with the CLOCK switches shown in logic diagrams, figure 9-4.

b. Mode Control.

1. General Description. One of four modes can be manually selected by depressing the corresponding mode indicator/switch. These modes are run, operation and phase step, and load. The phase step mode is the only one which affects the master clock so as to allow manual control of phase generation. Otherwise, the clock runs at normal speed. The use of the other three modes is discussed in later sheets.

2. Detailed Analysis. Refer to the UNIVAC 1219 Technical Manual, Volume I, Section 4, figure 4-3 and logic diagrams, figure 9-3.

The computer can be in one of the four modes shown. The desired mode is initiated by depressing the associated mode button. This action grounds the output of the corresponding 00J0- gate to enable that mode. The mode gates are cross-coupled such that once a mode is manually selected it will be held in control after the mode button is released. The computer itself can initiate some of these modes by other inputs to these control gates.

c. Phase Step Mode Without Phase Repeat.

1. General Description. The generation of the clock phases can be manually controlled such that each depression of the START STEP/RESTART switch will produce one phase. Each manipulation of this switch will produce the next phase. The produced phase occurs once and has its normal pulse width.

The sequence of events for manual phase generation is as follows.

STEP 1. Depress PHASE STEP MODE button.

STEP 2. Depress MANUAL CLEAR PHASE button.

STEP 3. Depress one PHASE button (first phase desired).

STEP 4. Depress START STEP/RESTART switch for each phase desired.

With the START STEP/RESTART switch in the locked up position, phase generation is controlled by the low-speed oscillator. This is an internal oscillator whose frequency can be varied from approximately 2 to 200 pps by the RESTART SPEED CONTROL potentiometer. Each oscillator cycle produces the next clock phase, thus simulating repeated depressions of the START STEP/RESTART switch.

2. Detailed Analysis.

a) START STEP/RESTART Switch Logic. Refer to figure 5.1-1 and logic diagrams, figure 9-3.

Remote control logic is not shown in figure 5.1-1 and the signal labels assume local control. The START STEP/RESTART switch is shown in the neutral position. The up position is locked and the down position is spring loaded to return to neutral when the switch is released.

In order to better describe the logic operation with signal labels, the output of the low speed oscillator is referred to as "LS pulse" when at a high level. This output is only used with the switch in the locked up position.

The purpose of the start flip-flop is to allow the switch to be honored only once for each depression or for each LS pulse when the switch is in the up position.

Develop and verify each of the signal labels shown in figure 5.1-1.

Refer to the output labels for both output gates of this logic group (23J10 and 35J10). Notice that the occurrence of the LS pulse with the switch in the up position has the same effect as does the switch being depressed. Also, the absence of this signal (LS pulse) with the switch in the up position has the same effect as does the switch being returned to the neutral position. Therefore, the up position

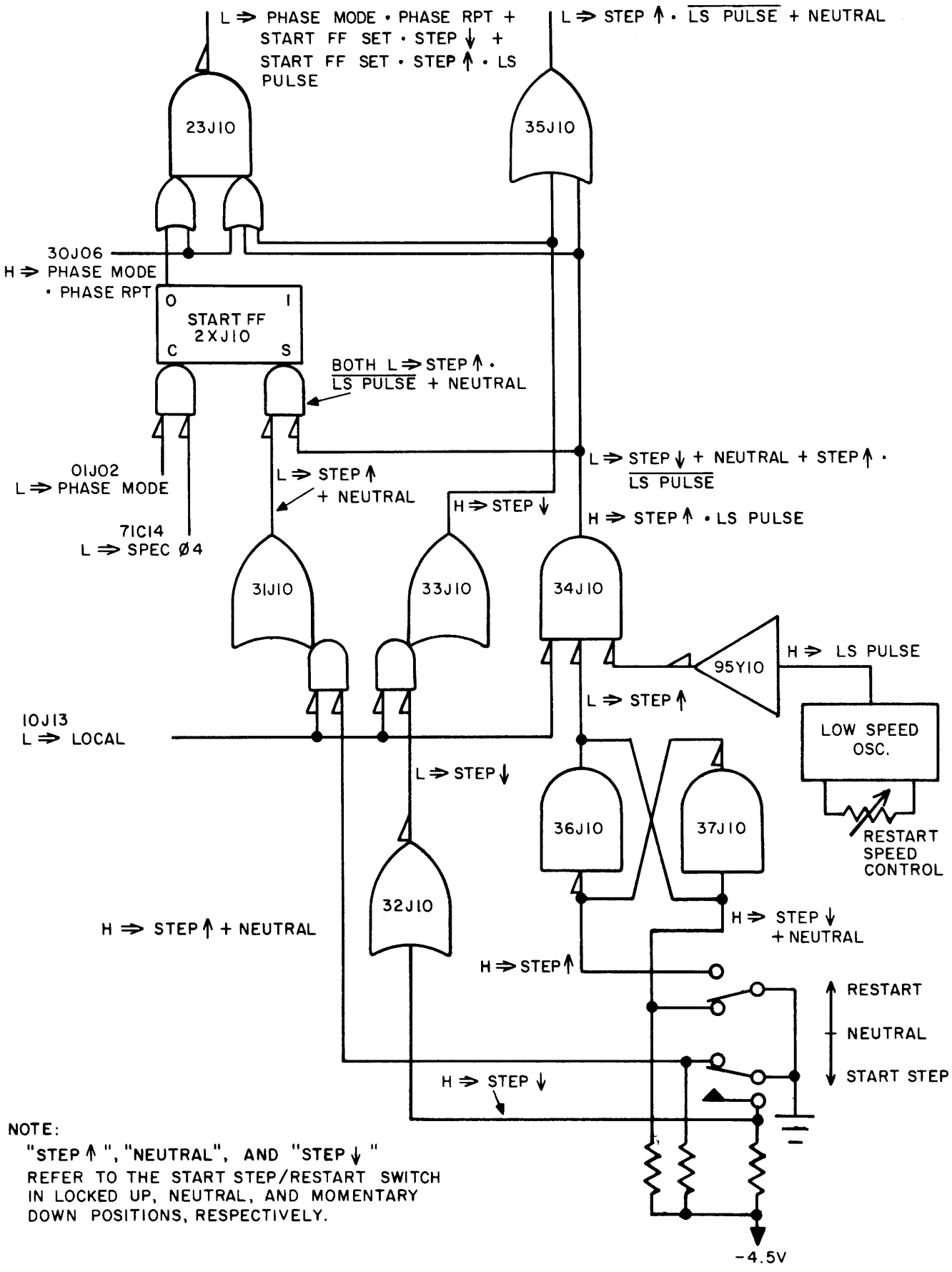


Figure 5.1-1. START STEP/RESTART Switch Logic

enables the low-speed oscillator to continually simulate the START STEP/RESTART switch being depressed and released to neutral.

b) Phase Generation. Refer to logic diagrams, figures 9-3 and 9-4.

With the phase step mode in effect, the generation of a particular phase is dependent upon the corresponding 1XJO- flip-flop. Each time that the START STEP/RESTART switch is released to the neutral position from being depressed or during LS pulse with this switch in the locked up position, the 1XJO- flip-flop which was set is cleared and the next 1XJO- flip-flop is set to enable generation of the next phase.

Refer to table 5.1-3 for the sequence of events for the phase step mode. This operation is described for repeated depression and releasing of the START STEP/RESTART switch. The events are the same for the up position of this switch except those events caused by depression of the switch are caused by the LS pulse and those events caused by releasing the switch to neutral are caused by LS pulse.

d. Phase Step Mode With Phase Repeat.

1. General Description. Phase repeat can only be achieved with the phase step mode in control. With the PHASE REPEAT switch on (up position), the manually selected phase will be continually generated at its normal rate. The START STEP/RESTART switch is not effective. The selection of a phase or combination of phases require the manipulation of the MANUAL PHASE CLEAR and PHASE buttons.

2. Detailed Analysis. Refer to logic diagrams, figures 9-3 and 9-4.

During the phase repeat operation, the clock oscillator is continually running. 24J06 has a constant low level input from 23J10.

30J06 outputs a constant high level to disable 32J06. Thus, none of the 0XJO- flip-flops can change state automatically. Likewise, none of the 1XJO- flip-flops change state. To select a phase or a combination of phases to be generated, the 0XJO- flip-flops must be cleared by the MANUAL PHASE CLEAR switch. The desired phases can be selected by depressing the associated indicator/switches which set the corresponding 0XJO- flip-flops. The proper 1XJO- flip-flops are immediately set to enable phase generation.

5.1-5. SUMMARY

The master clock is a four-phase timing source. It is free running if the phase step mode is not in effect. With the phase step mode (non-phase repeat) in effect phase generation is manually or low-speed-oscillator controlled as determined by the START STEP/RESTART switch. With phase-repeat and phase-step modes, the clock is free running. However, phase generation is selected by the PHASE indicator/switches.

TABLE 5.1-3. PHASE STEP MODE SEQUENCE OF EVENTS WITHOUT PHASE REPEAT
(Initially not in phase mode, clock running)

SWITCH ACTION	PHASE FLIP-FLOPS									Start ff	Clock
	1XJ06	1XJ07	1XJ08	1XJ09	0XJ06	0XJ07	0XJ08	0XJ09	24J06		
STEP NEUTRAL	**	**	**	**	**	**	**	**	H	Set	Ø1
	**	**	**	**	**	**	**	**	H	Set	Ø2
	**	**	**	**	**	**	**	**	H	Set	Ø3
	**	**	**	**	**	**	**	**	H	Set	Ø4, Spec Ø4*
Depress PHASE MODE	**	**	**	**	**	**	**	**	L^1	Set	Stop clock^2
Depress MAN CLR PHASE	Clr^2	Clr^2	Clr^2	Clr^2	Clr^1	Clr^1	Clr^1	Clr^1	L^1	Set	Stop clock^2
Depress Ø1	Set^2	Clr^2	Clr^2	Clr^2	Set^1	Clr^1	Clr^1	Clr^1	L^1	Set	Stop clock^2
Depress STEP	Set^2	Clr^2	Clr^2	Clr^2	Set^1	Clr^1	Clr^1	Clr^1	H^1	Set	En clock^2
	Set^2	Clr^2	Clr^2	Clr^2	Set^1	Clr^1	Clr^1	Clr^1	H^1	Set	Ø1
	Set^2	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	H^1	Clr^2	Spec Ø4*^{1}
	Set^2	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	H^1	Clr^2	
STEP neutral	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	L^1	Set^1	Stop clock^2
	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	L^1	Set^1	Stop clock^2
Depress STEP	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	H^1	Set^1	En clock^2
	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	Clr^1	H^1	Set^1	Ø2
	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	H^1	Clr^2	Spec Ø4*^{1}
	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	H^1	Clr^2	
STEP neutral	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	L^1	Set^1	Stop clock^2
	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	L^1	Set^1	Stop clock^2
Depress STEP	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	H^1	Set^1	En clock^2
	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	Clr^1	H^1	Set^1	Ø3
	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	H^1	Clr^2	Spec Ø4*^{1}
	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Clr^2	Set^2	H^1	Clr^2	
STEP neutral	Clr^1	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	L^1	Set^1	Stop clock^2
	Clr^1	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	L^1	Set^1	Stop clock^2
Depress STEP	Clr^1	Clr^1	Clr^1	Set^1	Clr^2	Clr^2	Clr^2	Set^2	H^1	Set^1	En clock^2
	Clr^1	Clr^1	Clr^1	Set^1	Set^2	Clr^2	Clr^2	Clr^2	H^1	Clr^2	Ø4, Spec Ø4*^{1}
	Clr^1	Clr^1	Clr^1	Set^1	Set^2	Clr^2	Clr^2	Clr^2	H^1	Clr^2	
STEP neutral	Set^1	Clr^1	Clr^1	Clr^1	Set^2	Clr^2	Clr^2	Clr^2	L^1	Set^1	Stop clock^2
	Set^1	Clr^1	Clr^1	Clr^1	Set^2	Clr^2	Clr^2	Clr^2	L^1	Set^1	Stop clock^2
Depress STEP	Set^1	Clr^1	Clr^1	Clr^1	Set^2	Clr^2	Clr^2	Clr^2	H^1	Set^1	En clock^2
	Set^1	Clr^1	Clr^1	Clr^1	Set^2	Clr^2	Clr^2	Clr^2	H^1	Set^1	Ø1
	Set^1	Clr^1	Clr^1	Clr^1	Clr^2	Set^2	Clr^2	Clr^2	H^1	Clr^2	Spec Ø4*^{1}
	Set^1	Clr^1	Clr^1	Clr^1	Clr^2	Set^2	Clr^2	Clr^2	H^1	Clr^2	
STEP neutral	Clr^1	Set^1	Clr^1	Clr^1	Clr^2	Set^2	Clr^2	Clr^2	L^1	Set^1	Stop clock^2

NOTES: * "Spec Ø4" refers to the low level output of gate 71C14.

** Phase flip-flops are not used if not in Phase Mode. Their states are not necessarily known.

"STEP" refers to the START STEP/RESTART switch.

Numbers shown in listings indicates the sequence of events for the particular time period.

NAME: _____

5.1-6. STUDY QUESTIONS

a. Refer to logic diagrams, figure 9-4.

What is the approximate time duration the 7XC24 flip-flop is set when the clock oscillator is free running?

_____ nanoseconds

b. Given: 35J10 grounded output (logic diagrams, figure 9-3).

Considering this malfunction, explain its effect upon the following operating modes.

1. Free running clock (non-phase step).

2. Phase step mode (not phase repeat).

c. Given: 03J02 grounded output (logic diagrams, figure 9-3).

Considering this malfunction, explain its effect upon phase step mode operation (non-phase repeat).

SECTION 5 - CONTROL SECTION

5.2. MAIN TIMING, INSTRUCTION SEQUENCER, OPERATION STEP MODE, SEQUENCE STEP, AND STOP OPERATIONS

5.2-1. OBJECTIVES

To present the detailed theory of operation involved in the timing logic which is controlled by the master clock.

5.2-2. INTRODUCTION

Main timing is derived from a series of flip-flops. The progression of this flip-flop timing chain is controlled by the master clock. Each time through the chain comprises a main timing cycle. Each of these cycles is under control of a sequence which specifies the basic operation performed during that cycle. Computer sequencing can be interrupted and manually controlled.

5.2-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraphs 4-2a(2), 4-2b(3), and 4-2c(4) and (5).

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.2-4. INFORMATION

a. General Description. In order to effectively expand the clock phase outputs, a group of 16 flip-flops, referred to as the main timing chain, is used. These flip-flops change their set/clear configuration on every clock phase. There are 16 individual configurations that occur during four 4-phase clock cycles. These four clock cycles comprise one main timing cycle. The flip-flops are arranged sequentially, such that each clock phase sets the next sequential flip-flop and clears some previously set flip-flop. Each flip-flop is set for the duration of one 4-phase clock cycle. The timing flip-flops are numbered according to the setting sequence. When the last flip-flop is set, it enables the setting of the first flip-flop on the next clock phase for recycling.

The necessary sequence of events to execute an operation is timed by the combination of the timing chain flip-flop outputs and the clock phases. Then, as each sequential flip-flop is set, a new step (data transfer to a register, etc.) may be performed as a part of an instruction execution. As discussed in later sheets, more than one timing chain cycle may be necessary to complete the execution. For example, one cycle could be used to obtain an instruction from memory; and the next cycle could execute that instruction.

b. Detailed Analysis. Refer to logic diagrams, figures 9-8 through 9-11.

The main timing flip-flops are arranged in four groups. Each group specifies one master clock cycle. Main timing will continue to cycle if 03T11, the setting gate for the T12 flip-flop, is enabled. This discussion assumes the timing cycle to be constantly reoccurring. Refer to table 5.2-1 for the flip-flop setting/clearing sequence. The time notations are discussed later in this sheet.

TABLE 5.2-1. MAIN TIMING

CLOCK PHASE	FLIP-FLOP ACTION	TIME NOTATION
Ø2	Set T11	
Ø3	Set T12	
Ø4	Set T13	
Ø1	Set T14	T1.1
Ø2	Set T21; clear T11	T1.2
Ø3	Set T22; clear T12	T1.3
Ø4	Set T23; clear T13	T1.4
Ø1	Set T24, clear T14	T2.1
Ø2	Set T31, clear T21	T2.2
Ø3	Set T32, clear T22	T2.3
Ø4	Set T33, clear T23	T2.4
Ø1	Set T34, clear T24	T3.1
Ø2	Set T41, clear T31	T3.2
Ø3	Set T42, clear T32	T3.3
Ø4	Set T43, clear T33	T3.4
Ø1	Set T44, clear T34	T4.1
Ø2	Set T11, clear T41	T4.2
Ø3	Set T12, clear T42	T4.3
Ø4	Set T13, clear T43	T4.4
Ø1	Set T14, clear T44	T1.1

NOTES: Timing continues as long as recycle (setting T12) is enabled.
T52 flip-flop is not shown.

The T52 flip-flop is set for only two special operations and is discussed in a later sheet.

The time notation specifies the current clock cycle of the main timing cycle (one of four) and the current clock phase. For example, T1.2 time exists during Ø2 in the first complete four-phase clock cycle after the setting of T11. Since four clock cycles occur for each main timing cycle, the main timing cycle is two microseconds in duration.

Refer to the UNIVAC 1219 Technical Manual, Volume I, Section 4, figure 4-4 for a waveform timing diagram of the main timing cycle.

The main timing cycle duration is directly associated with the two-microsecond main memory cycle. Each time that a main memory reference is needed to obtain a word or store a word, a main timing cycle is required to control the transfer of data to and from memory.

c. Instruction Sequencer.

1. General Description.

a) Purpose of Sequences. During the execution of an instruction, the main timing cycle is always under control of a sequence (I, R1, R2, or W for non-I/O instructions). These sequences determine the use of the timing signals which may be taken from the main timing chain flip-flops. Different sequences in control will cause different operations to be performed. In special cases, two sequences run in parallel. The instruction itself selects (via the function code translator) the sequences to be used. The number of sequences used determines the instruction execution time.

b) Sequence Types.

1) I Sequence. This sequence is used when the computer is ready to execute a new instruction. This new instruction is obtained from memory at the address specified by the program. If required, B modification is performed. The necessary value of U_P, U_{SR}, or XU is formulated.

Many instructions require only the I sequence and are therefore executed in only two microseconds. For these instructions, special additional I sequence operations are performed.

2) R1 Sequence. This sequence obtains an operand (value to be operated on) from memory at the address specified in the instruction. Some instructions are completed by the R1 sequence.

3) R2 Sequence. This sequence is only used by f = 20-23 instructions. A second operand is obtained from memory.

4) W Sequence. This sequence performs storage of values into memory at the address specified in the instruction.

2. Detailed Analysis. Each sequence effects control by two flip-flops which may be referred to as initial and final; e.g., I sequence$_i$ and I sequence$_f$. When set they cause the execution of their designated operations as timed by main timing and master clock.

Refer to logic diagrams, figures 9-12 and 9-13.

Except for the I/O sequence flip-flops, the upper rank contains the initial sequence flip-flops and the lower rank contains the sequence final flip-flops. Notice that of the sequences listed above, the initial sequence flip-flops are cleared at T4.1 time and are enabled to be set at T4.2 time. The final flip-flops are cleared at T2.1 time and are enabled to be set at T2.2 time.

The final flip-flop set at T2.2 time corresponds to the initial flip-flop which has been set at T4.2 time. The setting of the initial flip-flops is determined by the instruction and its use of the main timing cycle. Refer to table 5.2-2 for the conditions to set the initial sequence flip-flops.

TABLE 5.2-2. INITIAL SEQUENCE FLIP-FLOP SETTING CONDITIONS (NON-I/O)

I_i ff		setting Int_i ff
	or	(Int_i ff set + Wait Seq ff clear) · (not setting $Wait_i$, W_i, and $R1_i$ ff's)
$R1_i$ ff		I_f ff set · format 1 · f = 02-33, 50-57
	or	$R1_f$ ff set · $R2_f$ ff clear · f = 20-23
$R2_i$ ff		$R1_f$ ff set · $R2_f$ ff clear · f = 20-23
W_i ff		I_f ff set · f = 50:44
	or	I_f ff set · format 1 · f = 40-47, 72, 74-76
	or	$R1_f$ ff set · f = 30, 31, 57, 76

NOTE: Setting time is T4.2

Each main timing cycle is under control of a sequence. The initial sequence flip-flops change (advance) their configuration once every main timing cycle. The final flip-flops follow the initial flip-flops. Each sequence, therefore, has a duration of one main timing cycle (two microseconds). Refer to figure 5.2-1 for an example of the sequencing operation.

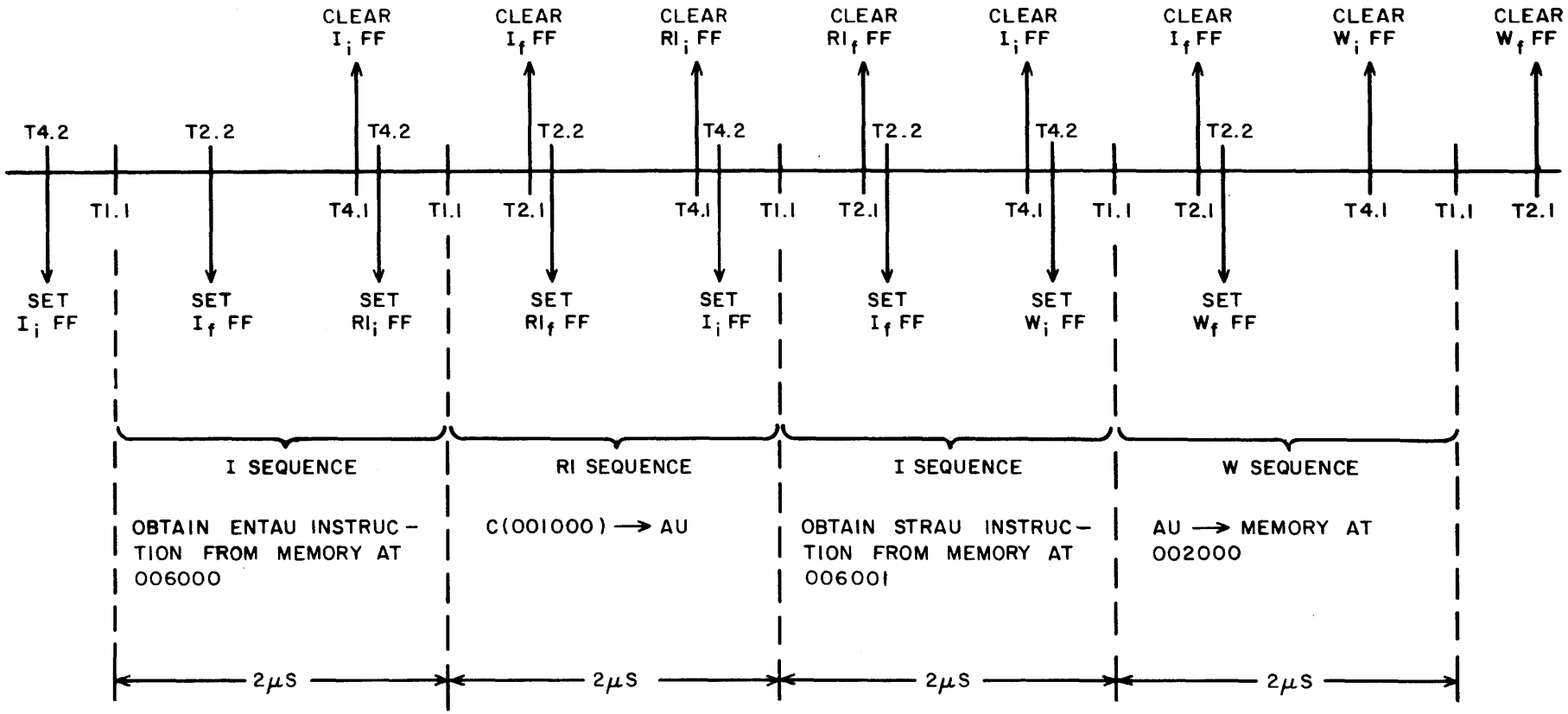
The detailed descriptions of operations performed during the sequences are presented in later sheets concerned with individual instruction executions.

d. Operation Step Mode and Sequence Step.

1. General Description. The operation step mode can only be manually initiated. This mode is put into effect and other modes are disabled by depressing the OPERATION STEP MODE indicator/switch. Without sequence step operation, each depression of the START STEP/RESTART switch causes the next instruction to be executed and then the computer stops. The stop is accomplished by stopping the main timing chain after the next I-sequence. Therefore, one instruction is

PROGRAM

ADDRESS	INSTRUCTION		
006000	101000	ENTAU	C(001000) → AU (4μS)
006001	462000	STRAU	AU → MEMORY AT 002000 (4μS)



NOTE: OTHER SEQUENCES ARE IN CONTROL PRIOR TO AND AFTER THE EXECUTIONS OF THE ABOVE INSTRUCTIONS BUT ARE NOT SHOWN.

Figure 5.2-1. Example of Sequencing Operation

executed and the following one is read from memory. The sequence indicators will specify the sequence following the I-sequence which will be executed upon the next switch depression.

Refer to figure 5.2-2 for an example of the operation step mode. This is the same example considered in figure 5.2-1 which shows the uninterrupted sequencing operation.

If the STOP/SEQUENCE STEP switch is in the locked up position, sequence step operation is performed. Each depression of the START STEP/RESTART switch causes the next sequence to be executed. With this operation, the results of each sequence of an instruction may be observed.

Automatic stepping can be achieved by placing the START STEP/RESTART switch in the locked up position. As was described for phase stepping, this switch position allows the low-speed oscillator to simulate continual depressing and releasing to neutral the START STEP/RESTART switch.

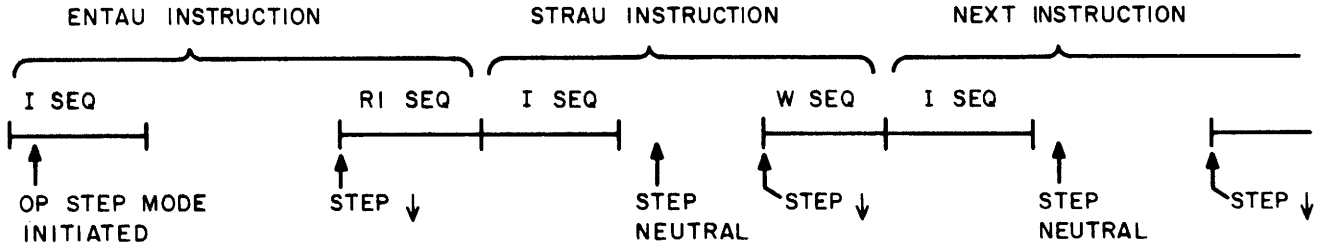
2. Detailed Analysis.

a) Instruction Stepping (not Sequence Step). Refer to figure 5.2-2 and logic diagrams, figure 9-3.

The STOP/SEQUENCE STEP switch is in the neutral position. Remote control logic is not shown in figure 5.2-3 and the signal labels assume local control. Verify each of the labels in figure 5.2-3. If necessary, refer to study guide sheet No. 5.1, figure 5.1-1 to develop the labels which refer to the START STEP/RESTART switch.

PROGRAM

ADDRESS		INSTRUCTION		
006000	=	101000	ENTAU	(I & RI SEQUENCES, 4μS)
006001	=	462000	STRAU	(I & W SEQUENCES, 4μS)



NOTE: "STEP ↓" AND "NEUTRAL" REFER TO THE START STEP/RESTART SWITCH IN THE NEUTRAL AND MOMENTARY DOWN POSITIONS, RESPECTIVELY.

Figure 5.2-2. Example of Operation Step Mode Without Sequence Step

Stopping is effected by the clearing of the run 1 flip-flop which prevents the setting of the T12 flip-flop via 07J10. Therefore, main timing stops. The run 1 flip-flop is cleared by 24J10 via 25J10. Notice that 24J10 is satisfied during the I-sequence if the B1 and B2 sequences are not running in parallel. The B1 and B2 sequences are used for certain I/O instructions and are described in a later sheet. The run 1 flip-flop, then, is cleared at T4.1 time of the I-sequence.

The computer remains stopped until the run 1 flip-flop is again set. The setting is accomplished by manipulation of the START STEP/RESTART switch via 23J10. The START flip-flop prevents more than one step operation for each depression of this switch. Refer to table 5.2-3 for the sequence of events.

With the START STEP/RESTART switch in the locked up position, depressing and releasing to neutral of this switch are simulated by LS pulse and $\overline{\text{LS pulse}}$, respectively. This can be seen by examining the inputs to 23J10 and the set side of the Start flip-flop as shown in figure 5.2-3.

b) Sequence Step. Refer to figure 5.2-3.

With the STOP/SEQUENCE STEP switch in the locked up position, which selects sequence step operation, 24J10 is constantly enabled. Therefore, the run 1 flip-flop is cleared at every T4.1 time to effect a sequence stop. Operations are similar to those for operation step previously described. Main timing is stopped after the setting of the T11 flip-flop and is not restarted until manipulation of the START STEP/RESTART switch.

e. Stop Operations.

1. General Description. The disadvantage in stopping the computer by means of the operation step mode is that main timing is stopped and not only the program but also any current I/O operations stop. Other program stops can be effected without affecting main timing. Depressing the STOP/SEQUENCE STEP switch causes a program stop at the completion of the current sequence. Execution of the f = 50:56 instruction also causes a program stop if the stop condition is satisfied. This instruction is described in more detail in a later sheet.

With either of these two methods, the program stop is effected by disabling the control of the sequences. No operation is performed without sequence control. The I/O sequences are not disabled and main timing continues unaffected, thus allowing I/O operations. The computer mode is not altered.

Once stopped, manipulation of the START STEP/RESTART switch will effect restart and the computer will continue operation in the same mode which had control prior to the stop.

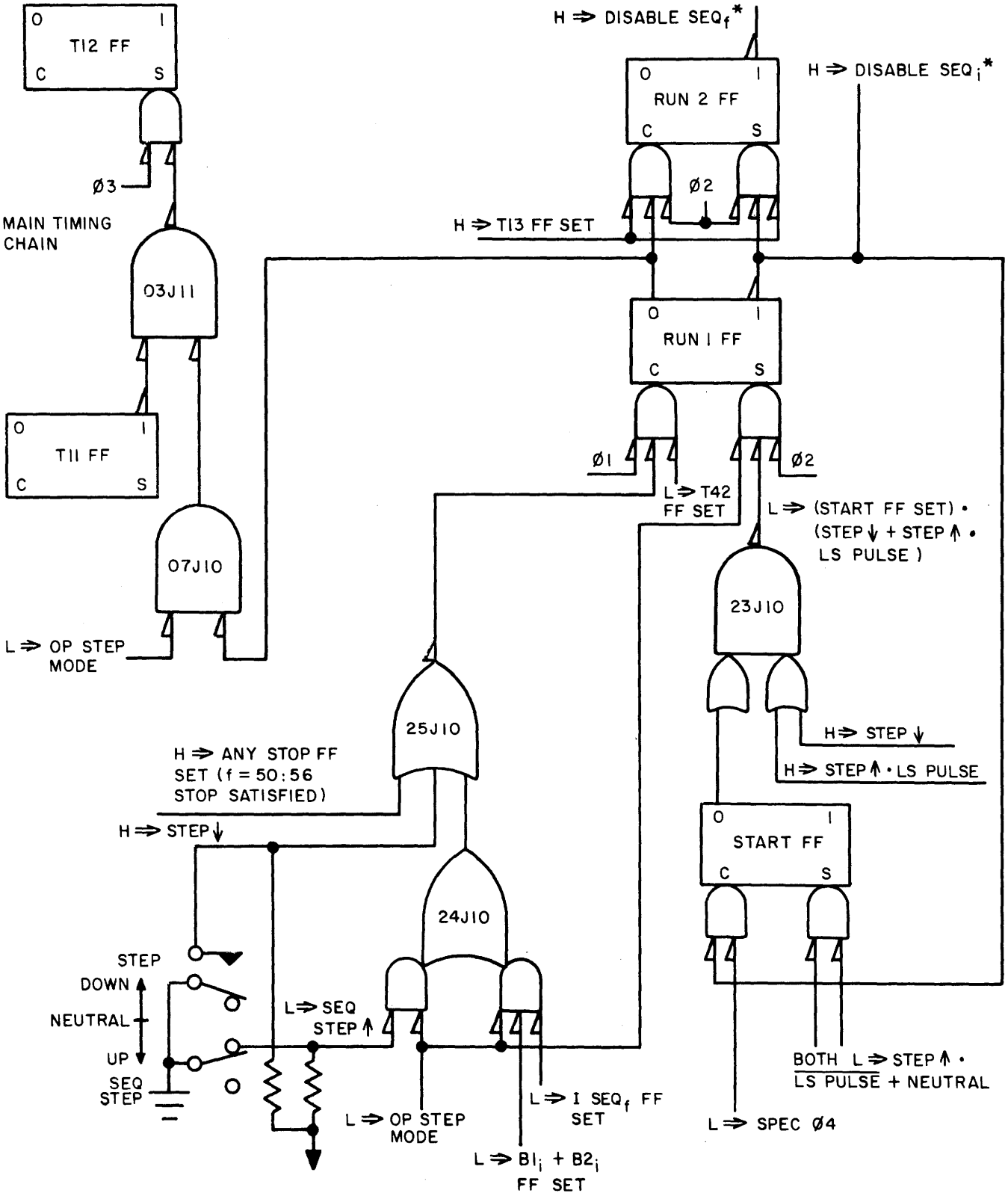
2. Detailed Analysis. (Refer to figure 5.2-3).

25J10 is used to clear the Run 1 flip-flop. This gate is enabled by depressing the STOP/SEQUENCE STEP switch or by any of the stop flip-flops being set. One or more of these flip-flops are set if the f = 50:56 instruction stop condition is satisfied.

TABLE 5.2-3. OPERATION STEP MODE SEQUENCE OF EVENTS WITHOUT SEQUENCE STEP (INITIALLY IN RUN MODE, COMPUTER RUNNING)

SWITCH ACTION	TIME NOTATION	START FF	RUN 1 FF	23J10 OUTPUT	OTHER CONDITIONS
		Set	Set	H	
Depress OP STEP MODE		Set	Set	H	
	T2.2	Set	Set	H	Set I seq_f ff
	T4.1	Set	Clear	H	Clear I seq_i ff
	T4.2	Set	Clear	H	Set next seq_i ff, set T11 ff
	Ø3	Set	Clear	H	Cannot set T12 ff (stop main timing)
Depress STEP		Set	Clear	L	
	Next Ø2	Set	Set	L	
	Ø3	Set	Set	L	Set T12 ff (enable main timing)
	Ø4, Spec Ø4	Clear	Set	H	Set T13 ff
		Clear	Set	H	
		Clear	Set	H	
	T2.2	Clear	Set	H	Set I seq_f ff
	T4.1	Clear	Clear	H	Clear I seq_i ff Set next seq_i ff Set T11 ff
	Ø3	Clear	Clear	H	Cannot set T12 ff (stop main timing)
STEP neutral		Set	Clear	H	

NOTE: "STEP" refers to the START STEP/RESTART switch.



NOTES: *ALL SEQUENCES DISABLED EXCEPT I/O.
"STEP↑", "NEUTRAL", AND "STEP↓" REFER TO THE START STEP/RESTART SWITCH IN LOCKED UP, NEUTRAL, AND MOMENTARY DOWN POSITIONS, RESPECTIVELY.

Figure 5.2-3. Operation Step, Sequence Step, and Stop Logic

Notice that main timing is not stopped unless the operation step mode is in control. The Run 2 flip-flop follows the state of the Run 1 flip-flop at T1.2 time. When both Run flip-flops are clear, all sequence control is disabled except for the I/O sequences. Refer to logic diagrams, figure 9-3 to follow the outputs of the Run flip-flops to verify their sequence disable function.

As discussed in a later sheet, the timing for the f = 50:56 instruction is such that the program stop which it causes occurs at the beginning of the I-sequence. If restarted, the computer would read from memory the instruction which followed the stop instruction and then execute it.

Notice that the manual stop is not timed by a sequence. If the STOP/SEQUENCE STEP switch is depressed, the stop occurs at the completion of the current sequence.

5.2-5. SUMMARY

The main timing cycle is two microseconds in duration and is created by a chain of 16 flip-flops. Four master clock cycles are necessary to complete the main timing cycle of two microseconds. Recycling of main timing is automatic. All computer operations are controlled by this cycle.

Each main timing cycle is under control of a sequence which determines how the cycle is used in performing part of the current instruction. Manual control of the main timing cycle and/or sequences can be effected by use of the operation step mode, sequence step operation, and programed or manual stop conditions.

NAME: _____

5.2-6. STUDY QUESTIONS

a. Refer to logic diagrams, figure 9-24. In the space provided below, draw the waveform of the output of 31T31. Start the waveform at T1.1 time and continue it for six microseconds. Indicate the proper time notation (such as T1.1) at each point where the wave form level changes. Assume no signal rise and fall times.

H (0v) ----

L (-4.5v)

Time Notation --- T1.1 ↑ T1.1 ↑ T1.1 ↑ T1.1 ↑

b. Given: 25J10 constant low level output (logic diagrams, figure 9-3).

Describe the effect that this malfunction would have upon the operation step mode with STOP/SEQUENCE STEP switch in the neutral position.

SECTION 5 - CONTROL

5.3. INSTRUCTION EXECUTION TECHNIQUES

5.3-1. OBJECTIVES

To present the general techniques employed for instruction execution.

5.3-2. INTRODUCTION

As each instruction is obtained, it is translated and executed. The execution is effected by data transfers among the various registers and memory locations. These transfers are performed in a predetermined sequence as controlled by the four-phase clock, main timing cycle, and timing sequences (I, R1, R2, W, B1, and B2).

5.3-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.3-4. INFORMATION

a. Main Memory Reference. The operations involved in referencing memory are accomplished during what is termed as the memory cycle. This two microsecond cycle is effected by a separate memory timing unit which reads information from the specified location and then writes that same or different information back into the same location. The read operation leaves the particular location cleared to 0's, thus requiring the write portion to restore the information.

There are several sections of memory referred to as banks. Each bank contains 4,096 addresses. The referenced address is put into the S1 register. S1 directs the operation to the particular group of 18 storage units which comprise the specified location. The Z1 register holds the information read from or written into main memory.

Each instruction requires at least the one memory cycle necessary to obtain that instruction. An instruction may require additional memory cycles during its execution. For example, a store instruction causes the contents of a register to be stored in memory. Refer to figure 5.3-1 for a block diagram description of the main memory operation as used to extract and store information.

b. Control Memory Reference. Any time that a memory reference is needed and the address is one which is assigned to control memory, the control memory cycle is initiated in place of main memory. This 500-nanosecond cycle is effected by a separate timing unit which has a read operation and a write operation like that for main memory.

S0 is the control memory address register. The Z0 register holds the information read from, or written into, control memory.

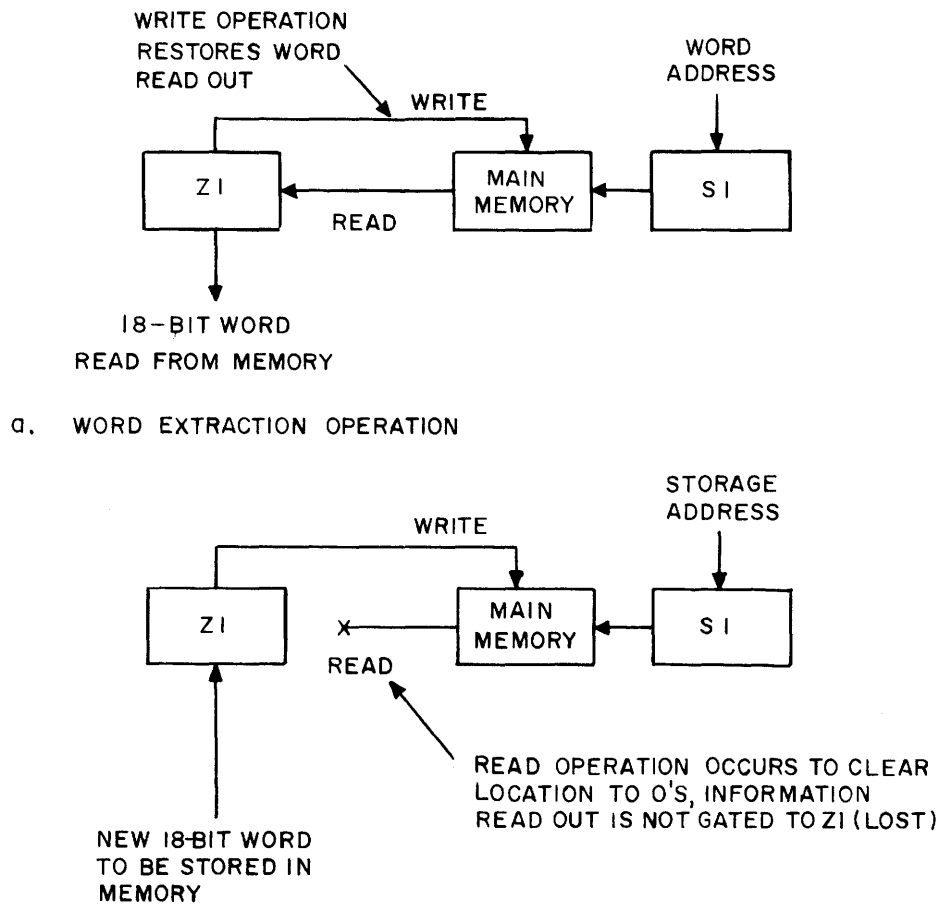


Figure 5.3-1. Main Memory Data Flow

d. Register Data Flow. The data flow necessary to obtain instructions and execute them is directed to the proper registers by means of register entrance gates controlled by timing signals referred to as commands. Each register has circuitry which generates commands to enable the clearing of the register and data entry into the register from the proper source. Refer to figure 5.3-2 for an example showing one bit position of the AU register.

The clear-AU command would occur prior to the data entry command. The AU flip-flop can be set from either X or the X-D' Adder. This flip-flop is in the logic diagrams, figure 9-97.

d. Selectors. Selectors are comprised of logic gates which have many input data sources and can have many output destinations. As for registers, each selector has logic circuitry which generates commands to enable the various input sources to apply their data to the selector gates. The selector actually serves the same function as the data entry gates to a register. The outputs of the selector can be one of the inputs to the register entry gates, thereby providing more data sources to the register than can be accommodated with the number of entry

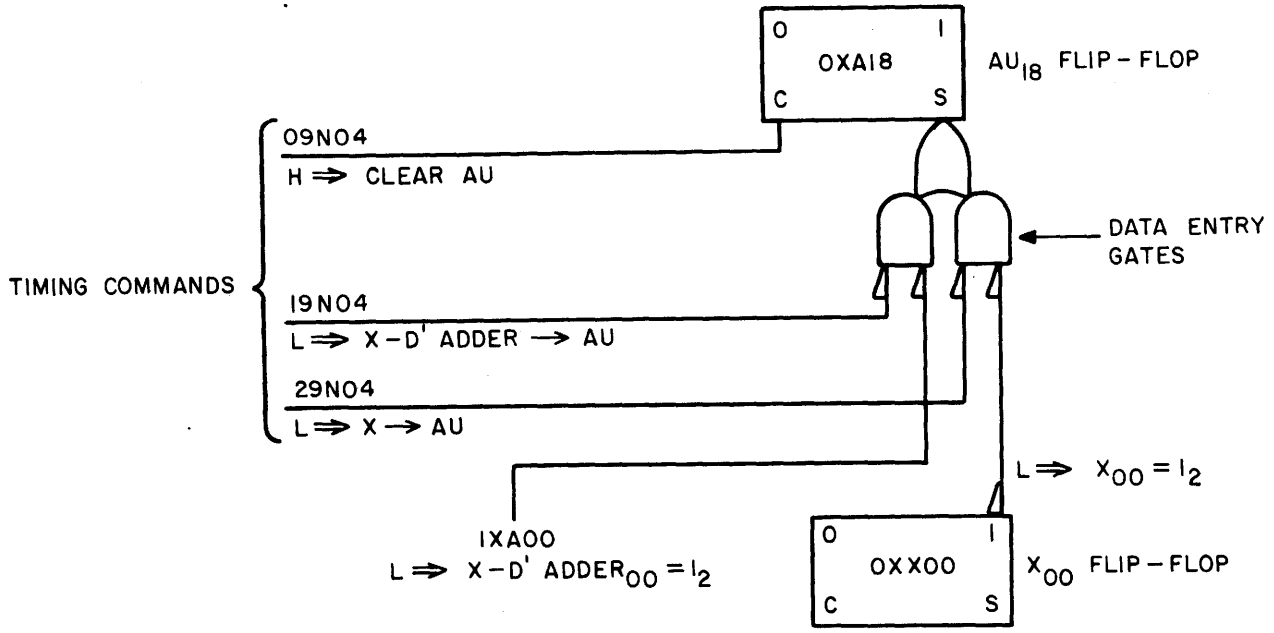


Figure 5.3-2. AU, Bit 18, Commands and Inputs

gates on the flip-flop logic card. Refer to figure 5.3-3 for an example showing one bit position of arithmetic select.

Notice that inputs to the selector can be other selectors as well as registers. Since the selector is comprised simply of gates, the input data source must be applied for the period of time that the data is needed at the output of the selector. The portion of the arithmetic select shown is in the logic diagrams, figure 9-82.

Usually, the data is applied to the selector for one clock cycle (500 nanoseconds). During this time period, the output of the selector can be used to set a register. Refer to table 5.3-1 for an example of data transfer from one register to another by way of a selector.

TABLE 5.3-1. TRANSFER OF AL TO X VIA ARITHMETIC SELECT

TIME NOTATION	COMMANDS
T1.1	AL ⟶ Arith Sel
T1.2	
T1.3	Clear X
T1.4	Arith Sel ⟶ X (X = AL)
T2.1	Drop AL ⟶ Arith Sel

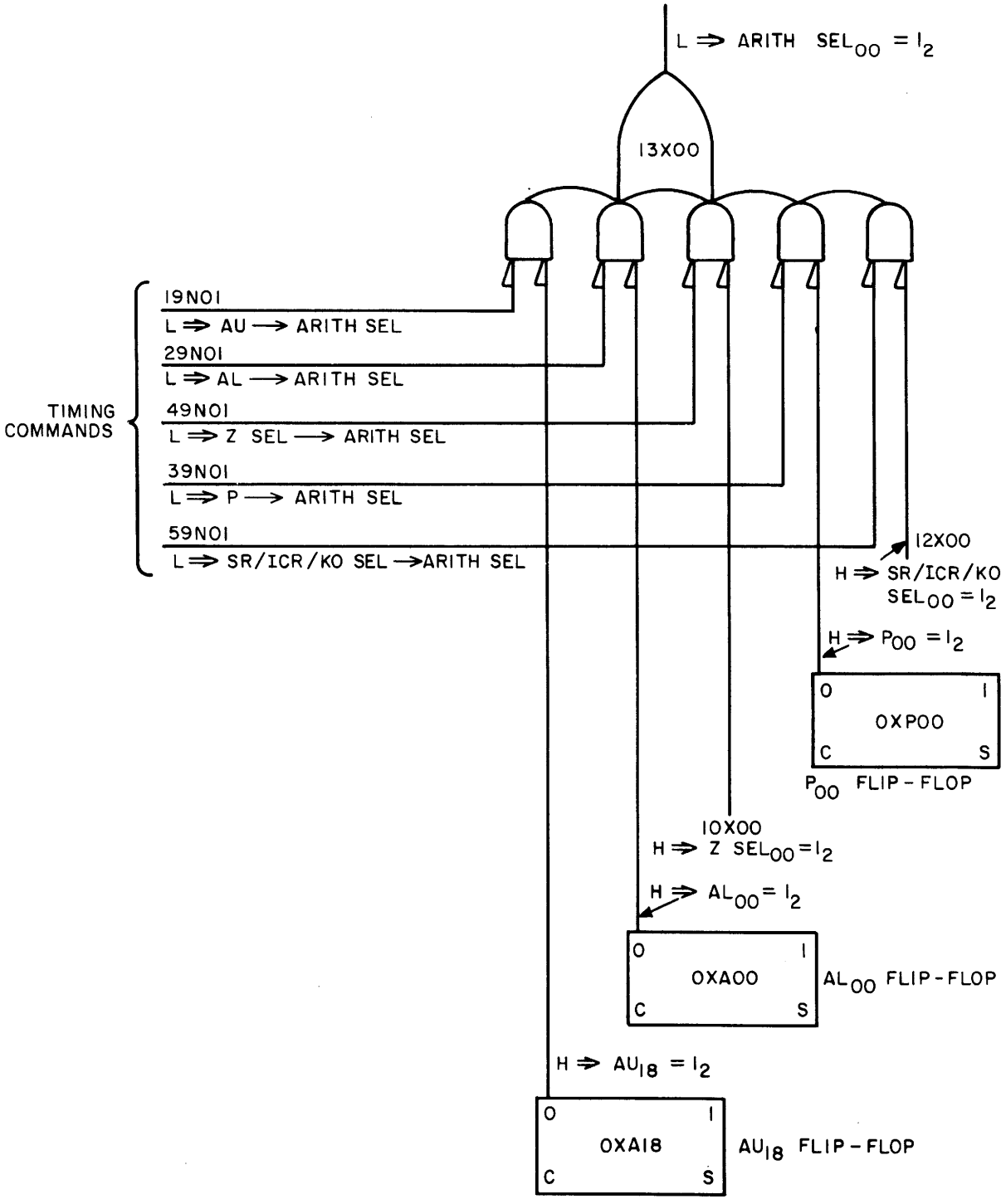


Figure 5.3-3. Arithmetic Select, Bit 00, Commands and Inputs

5.3-5. SUMMARY

None.

SECTION 5 - CONTROL

5.4. I-SEQUENCE

5.4-1. OBJECTIVES

To present the detailed theory of operations involved in the I-sequence.

5.4-2. INTRODUCTION

The I-sequence is used to obtain the next instruction from memory and formulate the value of U, XU, U_P, or U_{SR} with or without B as required by the instruction. Some instructions are able to complete their operations within the two microsecond I-sequence.

5.4-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Section 4-7, Table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.4-4. INFORMATION

a. General Description. The I-sequence is used to obtain the instruction to be executed. The instruction is extracted from memory at an address which is dependent upon the previous instruction. If that instruction did not specify a program jump or skip, the next instruction is obtained from the next sequential address of the program. Refer to table 5.4-1 for a list of the basic operations performed during the I-sequence.

TABLE 5.4-1. BASIC FUNCTIONS OF I-SEQUENCE

NUMBER	FUNCTION
1	S is set from P to address of instruction.
2	Memory cycle is initiated referencing address in S.
3	Instruction word is read from memory and put in F, KO, and D.
4	P is incremented by +1 to formulate address for next instruction.
5	B (index register) is obtained from control memory at address in ICR.
6	Value of U_P or U_{SR} is formulated, if required.
7	B is added to U_P or U_{SR} if required.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.4-1 for a block diagram description of the I-sequence operations.

S1 receives the instruction address from P at T1.1 time. A memory cycle obtains this instruction which is put in Z1 and applied to Z select. From Z select, the various portions of the instruction word are sent to three destinations. The six most significant bits of the word are examined from Z select to determine the format. If the instruction is format 1, bits 17-12 are considered the function code and are placed in F. If the instruction is format 2, bits 11-6 are placed in F as the function code. The function code is translated by the function code translator which controls the remaining operations to execute the instruction.

K0 receives the six least significant bits of the instruction word. Although the transfer into K0 is unconditional, K0 is not used by all of the instructions.

The entire instruction word is applied to arithmetic select from Z select. However, bits 17-12 are masked out by 0's. The remaining 12 bits are placed in D. This value is referred to as U. Depending upon the instruction, D can receive bits from either SR or P so as to formulate U_P or U_{SR} in D. For certain function codes, $SR_{4,2-0}$ is conditionally set in D_{15-12}. This transfer occurs only if $SR_3 = 1_2$, which indicates that SR is active. With exceptions, for these function codes, P_{15-12} is set in D_{15-12} if SR is inactive ($SR_3 = 0_2$).

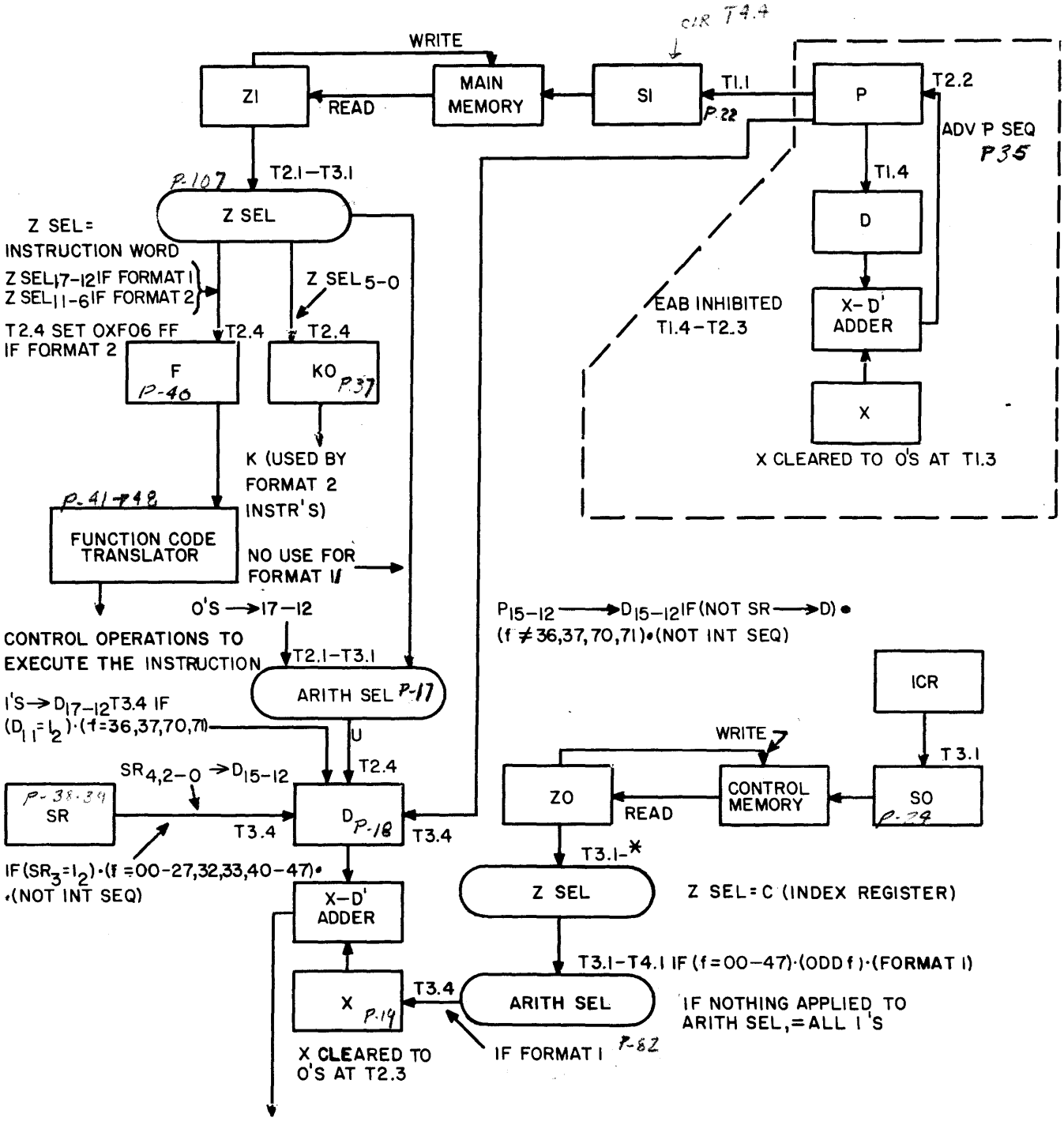
If the interrupt sequence is active, the instruction is obtained from an interrupt entrance address due to the detection of an interrupt. The value of U contained in D is not modified by SR or P if the interrupt sequence is active. Therefore, if the instruction which is executed from an interrupt entrance address due to an interrupt references memory, the memory address will be in bank 0 ($00000-07777_8$). Usually the instruction contained in an interrupt address is a jump instruction. If this instruction performs a direct jump, it will jump to bank 0. If the instruction performs an indirect jump, the address of the jump address is in bank 0.

The value in D (U, U_{SR}, or U_P) is modified by an index register for format 1 instructions whose function codes are less than 50_8 and are odd numbered. The index register is referred to as B and is the content of one of 8 addresses ($00001-00010_8$) in control memory. The address is specified by the content of ICR. Except for the value of 000_2, the value of ICR is the exact address of B. The value of 000_2 specifies the B register at address 00010_8.

If B is to be used, it is placed in X. The X-D' adder output of B-D' is effectively B + D. If B is not used, the value in D (U, U_{SR}, or U_P) is outputted by the adder without modification. If used, the adder output is either the operand or the address of the operand depending upon the instruction.

The advance P subsequence is used to increment P by +1. This operation sets P to the next consecutive address in preparation for obtaining the next sequential instruction of the program. If the instruction obtained by the I-sequence performs a program jump or skip, P will later be changed to the address of the desired next instruction.

Notice that if U is modified by P_{15-12}, D is set to these bits at T3.4 time. This time is after P has been advanced by +1. Therefore, if the current instruction was obtained from the last address of a memory bank, the value U_P will specify the



NOTES: * ZO → Z SEL OCCURS FOR THE DURATION OF THE T24 FF CLEAR
OPERAND = 18 BITS
ADDRESS = 16 BITS
ONLY THOSE EVENTS COMMON TO MOST INSTRUCTIONS ARE SHOWN

Figure 5.4-1. I-Sequence Data Flow

next bank. Obviously, special attention must be given to the use of the last bank addresses.

Some instructions cause other events to occur during the last portion of the I-sequence. These additional operations are presented in later sheets.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.4-2 for a sequential list of essential-I sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

TABLE 5.4-2. I-SEQUENCE ESSENTIAL COMMANDS

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P → S1, Init Memory, *set Incr P ff
T1.3	*Clear D, *Clear X, Clear F, Clear Z1, *set OXL11 ff
T1.4	*P_L → D_L, *P_U → D_U, Clear K0, *set Inhib EAB ff
T2.1	Z1 → Z Sel, Z Sel → Arith Sel, 0's → Arith Sel_{17-12}, *Clear P *Clear Incr P ff *Adder → P
T2.3	Clear X, Clear D, *Clear OXL11 ff, *Clear Inhib EAB ff
T2.4	Z Sel_{17-12} → F if format 1**, Z Sel_{11-6} → F & set OXF06 ff if format 2**
	Arith Sel → D, Z Sel_{5-0} → K0
T3.1	ICR → S0, Init CM, Z0 → Z Sel***
	Z Sel → Arith Sel if (f = 00-47) · (odd f) · (format 1); otherwise drop Z Sel → Arith Sel
	Drop Z1 → Z Sel, drop 0's → Arith Sel_{17-12}
T3.4	$SR_{4,2-0}$ → D_{15-12} if (SR_3 = 1) · (f = 00-27, 32, 33, 40-47)· (not Int Seq.)
	P_{15-12} → D_{15-12} if (not SR → D) · (f ≠ 36, 37, 70, 71) · (not Int Seq.)
	Arith Sel → X if format 1
T4.1	Drop Z Sel → Arith Sel

*These events are concerned with or are controlled by the advance P subsequence.

**The format of the instruction is sensed from Z Sel_{17-12}.

***Z0 → Z Sel occurs for the duration of the T24 ff clear.

Only those events common to most instructions are listed.

5.4-5. SUMMARY

The I-sequence obtains from memory the instruction to be executed and formulates the value of U, XU, U_P, or U_{SR} with or without B depending upon the requirements of the instruction. One of these values is available at the output of the X-D' adder at the end of the I-sequence. Some instructions require only the I-sequence for their executions.

NAME: _____

5.4-6. STUDY QUESTIONS

a. Given: Pin 6 of OXF06 ff grounded (logic diagrams, figure 9-40)
instruction in memory = 507203

Explain briefly what would happen if the computer attempted to read this instruction from memory and execute it. What instruction would actually be executed?

b. Given: 70N01 constant low level output (logic diagrams, figure 9-17)
instruction in memory = 100500
SR = 01010_2

During the execution of this instruction, the content of some memory location is put into AU. Considering the given conditions, what will be the address from which the operand is obtained?

Address of operand = _____

SECTION 5 - CONTROL

5.5. FUNCTION CODE TRANSLATOR

5.5-1. OBJECTIVES

To present the detailed theory of operation involved in the function code translator.

5.5-2. INTRODUCTION

The function code translator interprets the bit configuration of the function code in the F register and controls operations to execute the instruction.

5.5-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraphs 4-2c (2), and 4-2c (3).

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.5-4. INFORMATION

a. General Description. The function code translator translates the function code portion of the instruction word from the F register. F contains bits 17 through 12 of the instruction word for format 1 and bits 11 through 06 for format 2. Format must be known to determine which of these instruction word bits are being translated. For example, if the translator should output a signal indicating "f = 30", the instruction could be either f = 30 or f = 50:30 depending upon the format. Format is specified by F_{06}.

Separate translations are performed on the two octal digits comprising the function code. The results of these translations are combined to determine the six-bit function code.

Refer to the UNIVAC 1219 Technical Manual, Volume I, Section 4, figure 4-7 for a block diagram.

b. Detailed Analysis

1. Subtranslator 1. Refer to logic diagrams, figure 9-41. This logic group translates the more significant octal digits of the function code from F_{05-03}. The gates which perform this translation are 08F00 through 08F30, 08F60, 09F40, 09F50, and 09F70. The last two numbers of the logic gate number indicate the octal digit for which the particular gate translates. For example, 08F00 outputs a high level if bits 05 through 03 of F contain 0's. Thus, this high level would indicate the presence of any function code whose more significant octal digit is 0_8 which is f = 00-07.

Other gates in this figure translate for larger function code groups. For example, a low level output of 91F00 indicates f = 00-07, 10-17, 20-27, 30-37, 40-47 or f = 00-47.

2. Subtranslator 2. Refer to logic diagrams, figure 9-42.

This logic group translates the less significant octal digit of the function code. Only four of the eight possible conditions are indicated because only bits $F_{02,01}$ are tested. For example, 09F01 outputs a low level if $F_{02,01} = 00_2$. F_{00} is not tested and could contain 0_2 or 1_2. Therefore, this low level indicates f = X0, X1. The "X" means the more significant octal digit can be anything.

3. Subtranslator 3. Refer to logic diagrams, figure 9-43.

This logic group simply provides many logic outputs to indicate the conditions of F_{06} and F_{00}. If $F_{00} = 1_2$, the function code must be an odd number. If combined with the logic outputs of subtranslator 2, the F_{00} indication can specify the exact less significant octal digit of the function code. For example, if subtranslator 2 indicates f = X4, X5 and subtranslator 3 indicates f = even, then f = X4.

F_{06} indicates the format of the instruction. If $F_{06} = 1_2$, the instruction is format 2. From this logic, format indication is combined with other function code translations to completely describe the instruction.

4. Translator f = 02-77. Refer to logic diagrams, figures 9-44 through 9-48.

These logic groups combine various translator outputs to indicate groups of function codes.

5.5-5. SUMMARY

The function code translator indicates the operation to be performed as specified by the instruction. Function code translation is effected by combining several subtranslations of the bit configuration in F.

NAME:_____

5.5-6. STUDY QUESTIONS

a. Given: 20F70 constant low level output (logic diagrams, figure 9-48)

Considering this malfunction, list all function codes which will cause 41F72 (figure 9-48) to output a low level.

b. Given: 30F00 constant low level output (logic diagrams, figure 9-48)

Considering this malfunction, list all function codes which will cause 41F72 (figure 9-48) to output a low level.

c. Refer to logic diagrams, figure 9-41.

List all function codes which will cause 94F00 to output a high level (no malfunctions).

d. Given: 08F40 constant low level output (logic diagrams, figure 9-41)

Considering this malfunction, list all function codes which will cause 93F00 (figure 9-41) to output a low level.

e. Given: 08F00 grounded output (logic diagrams, figure 9-41)

Considering this malfunction, list all function codes which will cause 93F00 (figure 9-41) to output a low level.

f. Given: 08F40 grounded output (logic diagrams, figure 9-41)

Considering this malfunction, list all function codes which will cause 95F00 (figure 9-41) to output a low level.

SECTION 5 - CONTROL

5.6. INSTRUCTION EXECUTION OF STOP

5.6-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of the STOP instruction, f = 50:56.

5.6-2. INTRODUCTION

This instruction allows the program to conditionally or unconditionally stop itself.

5.6-3. REFERENCES

UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.6-4. INFORMATION

a. General Description

1. Instruction Interpretation. The STOP instruction, f = 50:56, conditionally stops the computer. A stop is effected if any 1_2 in k corresponds in bit position number to a manually selected STOP switch. If $k_{05} = 1_2$, an unconditional stop is performed. The computer is stopped by disabling the control of all sequences except I/O. Main timing is not affected; therefore, I/O operations can continue.

After the stop, the STOP indicator which is lighted specifies which bit of k caused the stop. These indicators are extinguished upon computer restart.

2. Execution Sequence (I). All operations are performed within the I-sequence. Only the one memory reference to obtain the instruction is necessary.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to Figure 5.6-1 for a block diagram description of the execution of f = 50:56.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

At T2.4 time, KO receives k of the instruction word. According to the bits set and STOP switches selected, the stop flip-flops are set. If any stop flip-flop is set at T4.1 time, the computer is stopped; i.e., sequences are disabled. If necessary, refer to study guide sheet number 5.2 for a review of the stop operation.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.6-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.6-5. SUMMARY

The STOP instruction allows the program to unconditionally stop itself with $k_5 = 1_2$. The other k bit positions set allow preselected manual stops. Stopping is effected by disabling the sequences. Main timing is not affected.

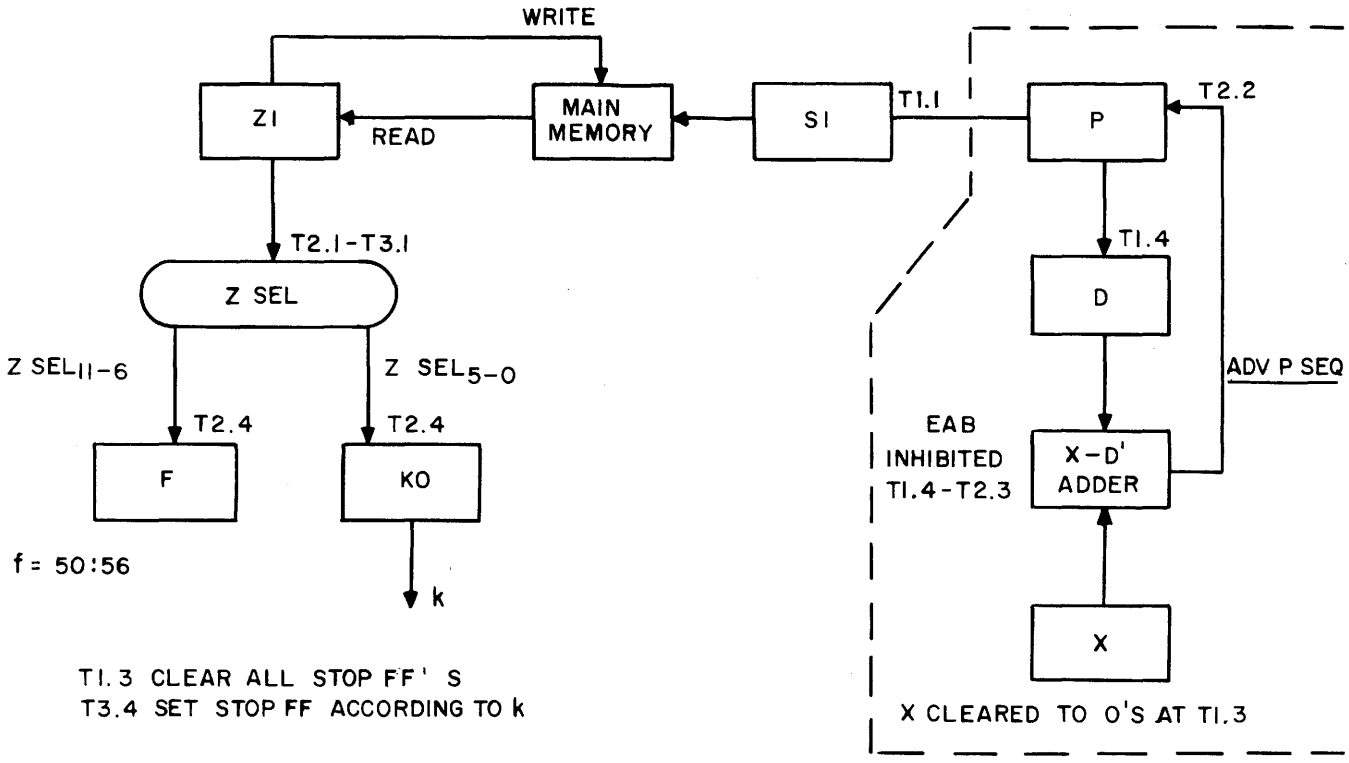


Figure 5.6-1. I-Sequence Data Flow For f = 50:56

TABLE 5.6-1. I-SEQUENCE ESSENTIAL COMMANDS FOR f = 50:56

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P $\longrightarrow$ S1, Init Memory, *set Incr P ff
T1.3	*Clear D, *Clear X, Clear F, Clear Z1, *set OXL11 ff, clear all Stop ff's
T1.4	*$P_L \longrightarrow D_L$, *$P_U \longrightarrow D_U$, Clear KO, *set Inhib EAB ff
T2.1	Z1 $\longrightarrow$ Z Sel, *Clear P, *clear Incr P ff
T2.2	*Adder $\longrightarrow$ P
T2.3	*Clear OXL11 ff, *clear Inhib EAB ff
T2.4	Z $Sel_{11-6} \longrightarrow$ F, set OXF06 ff, Z $Sel_{5-0} \longrightarrow$ KO
T3.4	Set stop ff corresponding to KO bit = 1_2 and selected STOP switch
T4.1	Clear run 1 ff if any stop ff set

*These events are concerned with, or are controlled by, the advance P subsequence.

NAME:_____

5.6-5. STUDY QUESTIONS

a. Given: OXG60 flip-flop, pin 12 grounded (logic diagrams, figure 9-31)

Describe the effect that this malfunction will have upon computer operations.

SECTION 5 - CONTROL

5.7. INSTRUCTION EXECUTION OF ENTALK, ADDALK

5.7-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instruction with f = 70, 71.

5.7-2. INTRODUCTION

These instructions enter AL with either the value of XU or XU + AL.

5.7-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.7-4. INFORMATION

a. General Description

1. Instruction Interpretation

a) ENTALK, f = 70. This instruction obtains the lower 12 bits of the instruction word, which is referred to as U, and extends its sign to formulate an 18-bit word. This value is placed in AL. The original contents of AL are destroyed (cleared).

b) ADDALK, f = 71. This instruction is similar to f = 70. It formulates an 18-bit value by extending the sign of U. XU is then added to AL and their sum is placed in AL. The original content of AL is destroyed. The Overflow flip-flop is set if an overflow occurs.

2. Execution Sequence (I). All operations are performed within the I-sequence. Since the operand Y does not come from memory, only the memory reference to obtain the instruction is necessary.

b. Detailed Analysis

1. Data Flow Block Diagram. Refer to figure 5.7-1 for a block diagram description of the execution of f = 70, 71.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

The operand Y is formulated in D. D receives U, the lower 12 bits of the instruction word, from arithmetic select. The sign bit of U is extended from D_{11}. For D, the operand XU is applied to one side of the X-D' Adder. If f = 70, X contains -0. If f = 71, X receives AL via arithmetic select. The final content of AL is X-D' which is effectively X + D. Therefore, for f = 70, AL receives -0 + XU. For f = 71, AL receives AL_1 + XU.

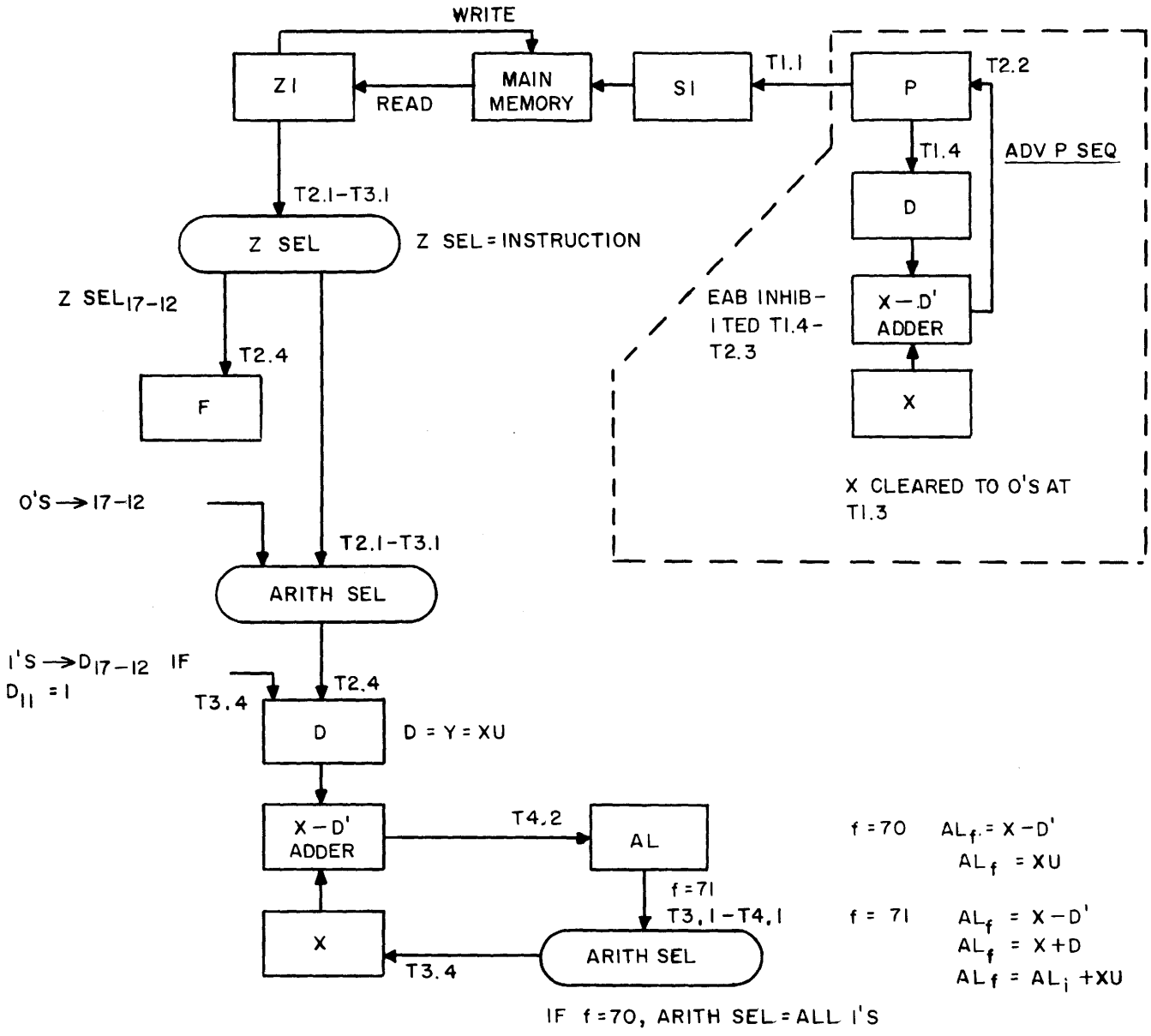
The setting of the overflow flip-flop for f = 71 is conditioned by the X-D' adder. The X-D' adder is analyzed in a later sheet.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.7-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.7-5. SUMMARY

The ENTALK and ADDALK instructions both use the value XU. The sign extension is formulated in D after U has been set into its lower 12 bits.



T4.2 SET OVERFLOW ff IF (f = 71)·(SIGN OF $AL_f \neq$ SIGN OF AL_i)*

NOTE: * THE OVERFLOW CONDITION IS INDICATED FROM THE X−D' ADDER BY:
($X_{17} = 0_2 \cdot D_{17}' = 1_2$ · NO BORROW REQUEST → BIT 17) +
($X_{17} = 1_2 \cdot D_{17}' = 0_2$ · BORROW REQUEST → BIT 17)

Figure 5.7-1. I-Sequence Data Flow For f = 70, 71

TABLE 5.7-1. I-SEQUENCE ESSENTIAL COMMANDS FOR f = 70, 71

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P $\longrightarrow$ S1, Init Memory, *Set Incr P ff
T1.3	Clear Z1, *set OXL11 ff, *clear D, *clear X, clear F
T1.4	$*P_L \longrightarrow D_L$, $*P_U \longrightarrow D_U$, *set Inhib EAB ff
T2.1	*Clear P, Z1 $\longrightarrow$ Z Sel, Z Sel $\longrightarrow$ Arith Sel, 0's $\longrightarrow$ Arith Sel_{17-12} *Clear Incr P ff
T2.2	*Adder $\longrightarrow$ P
T2.3	Clear X, *clear Inhib EAB ff, *clear OXL11 ff
T2.4	A $Sel_{17-12} \longrightarrow$ F, Arith Sel $\longrightarrow$ D
T3.1	AL $\longrightarrow$ Arith Sel if f = 71, drop Z1 $\longrightarrow$ Z Sel drop Z Sel $\longrightarrow$ Arith Sel, drop 0's $\longrightarrow$ Arith Sel_{17-12}
T3.4	1's $\longrightarrow D_{17-12}$ if D_{11} = 1, Arith Sel $\longrightarrow$ X
T4.1	Clear AL, drop AL $\longrightarrow$ Arith Sel
T4.2	Adder $\longrightarrow$ AL, set Overflow ff if (f = 71)$\cdot$(sign of $AL_f \neq$ sign of AL_i)

*These events are concerned with, or are controlled by, the advance P subsequence.

NAME:_____

5.7-6. STUDY QUESTIONS

a. Given: instruction = 714000
AL_i = 004000

Assume each of the following malfunctions to occur individually. Indicate AL_f for each of these conditions.

1. 13X11 grounded output (logic diagrams, figure 9-83)
AL_f = _____

2. 50NO2 constant low level output (logic diagrams, figure 9-18)
AL_f = _____

3. Pin 13 of 0XD11 ff grounded (logic diagrams, figure 9-86)
AL_f = _____

4. Pin 13 of 0XX11 ff grounded (logic diagrams, figure 9-88)
AL_f = _____

5. Pin 15 of 0XA11 ff grounded (logic diagrams, figure 9-100)
AL_f = _____

6. 70NO1 constant low level output (logic diagrams, figure 9-17)
AL_f = _____

b. Given: 20NO1 grounded output (logic diagrams, figure 9-17)
instruction = 712005
SR = 01100_2
AL_i = 600731
content of address 42005 = 045007

Give the final content of AL.
AL_f = _____

SECTION 5 - CONTROL

5.8. INSTRUCTION EXECUTION OF ENTICR, ENTSR

5.8-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 50:72, 50:73.

5.8-2. INTRODUCTION

These instructions enter either ICR or SR with the lower bits of the instruction word.

5.8-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.8-4. INFORMATION

a. General Description

1. Instruction Interpretation

a) ENTICR, f = 50:72. This instruction places the three least-significant bits of the instruction word in ICR. The original content of ICR is destroyed. ICR is used to specify a B register as used by certain other instructions.

b) ENTSR, f = 50:73. This instruction places the five least-significant bits of the instruction word in SR. The original content of SR is destroyed. If SR is active, its content is used by certain other instructions with bits 11-0 of these instruction words to formulate a memory address. SR is considered active if its bit position 3 is equal to 1_2.

2. Execution Sequence (I). All operations are performed within the I-sequence. Only the one memory reference to obtain the instruction is necessary.

b. Detailed Analysis

1. Data Flow Block Diagram. Refer to 5.8-1 for a block-diagram description of the execution of f = 50:72, 50:73.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

KO is set to the six least-significant bits of the instruction word from Z-select. If f = 50:72, ICR is cleared and receives bits 2-0 of KO which are the lower three bits of the instruction word. If f = 50:73, SR is cleared and receives bits 4-0 of KO. If bit position 3 of the instruction word is equal to 1_2, this value entered in SR makes SR active.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.8-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.8-5. SUMMARY

The ENTICR and ENTSR instructions are both format 2 and use the value k. The k value is available in KO after T2.4 time. Only the I-sequence is required to complete the execution of these instructions.

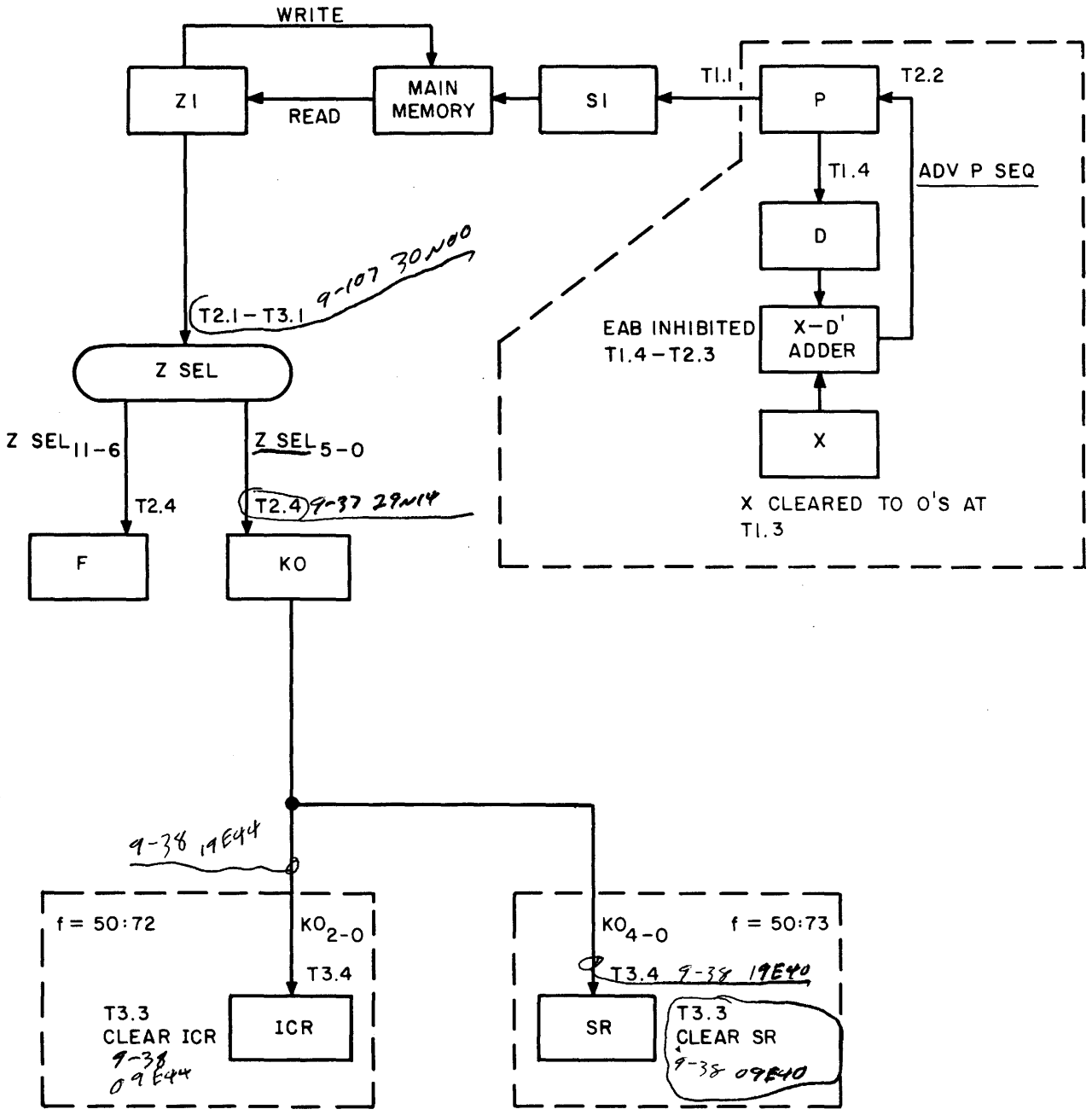


Figure 5.8-1. I-Sequence Data Flow For f = 50:72, 50:73

TABLE 5.8-1. I-SEQUENCE ESSENTIAL COMMANDS FOR f = 50:72, 50:73

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P ⟶ S1, Init Memory, *set Incr P ff
T1.3	*Clear D, *clear X, clear Z1, clear F, *set OXL11 ff
T1.4	$*P_L \longrightarrow D_L$, $*P_U \longrightarrow D_U$, clear KO, *set Inhib EAB ff
T2.1	*Clear P, Z1 ⟶ Z Sel, *clear Incr P ff
T2.2	*Adder ⟶ P
T2.3	*Clear OXL11 ff, clear Inhib EAB ff
T2.4	$Z\ Sel_{11-6} \longrightarrow F$, set OXF06 ff, $Z\ Sel_{5-0} \longrightarrow KO$
T3.1	drop Z1 ⟶ Z Sel
T3.3	Clear ICR if f = 50:72, clear SR if f = 50:73
T3.4	$KO_{2-0} \longrightarrow$ ICR if f = 50:72, $KO_{4-0} \longrightarrow$ SR if f = 50:73

*These events are concerned with or are controlled by the advance-P subsequence.

NAME: _____

5.8-6. STUDY QUESTIONS

a. Given:

Address	Instruction
006000	507203

This instruction is part of a program that is currently being executed. Assume that at T1.3 time of the I-sequence for this instruction, pin 8 of ÖXG32 flip-flop (logic diagrams, figure 9-27) is shorted to ground and remains grounded. Describe the effect that this malfunction would have upon the execution of the program.

SECTION 5 - CONTROL SECTION

5.9. INSTRUCTION EXECUTION OF ENTBK, ENTBKB

5.9-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 36, 37.

5.9-2. INTRODUCTION

These instructions enter B with either the value of XU or XU + B.

5.9-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.9-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) ENTBK, f = 36. This instruction obtains the lower 12 bits of the instruction word, U, and extends its sign to formulate an 18-bit word. This value is placed in the B register specified by the content of ICR. The original content of B is destroyed.

b) ENTBKB, f = 37. This instruction is similar to f = 36. It formulates an 18-bit value by extending the sign of U. XU is then added to the content of the B register specified by ICR and their sum is placed in the same B register. The original content of B is destroyed.

2. Execution Sequences.

a) I-Sequence. The operand Y (XU or XU + B) is formulated within the same I-sequence which obtains the instruction from memory.

b) Next I-Sequence. The storage of Y into B is performed during the first portion of the I-sequence for the next sequential instruction.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.9-1 for a block diagram description of the execution of f = 36, 37.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

The value of XU is formulated in D from where it is applied to one side of the X-D' adder. A control memory reference is used to obtain the content of the B register specified by ICR. ICR selects the B register by setting SO to the proper address ($00001 - 00010_8$).

If f = 37, the content of the B register is placed in X. Z1 receives the value of X-D' which is effectively X + D. For f = 36, Z1 receives -0 + XU. For f = 37, Z1 receives B + ZU.

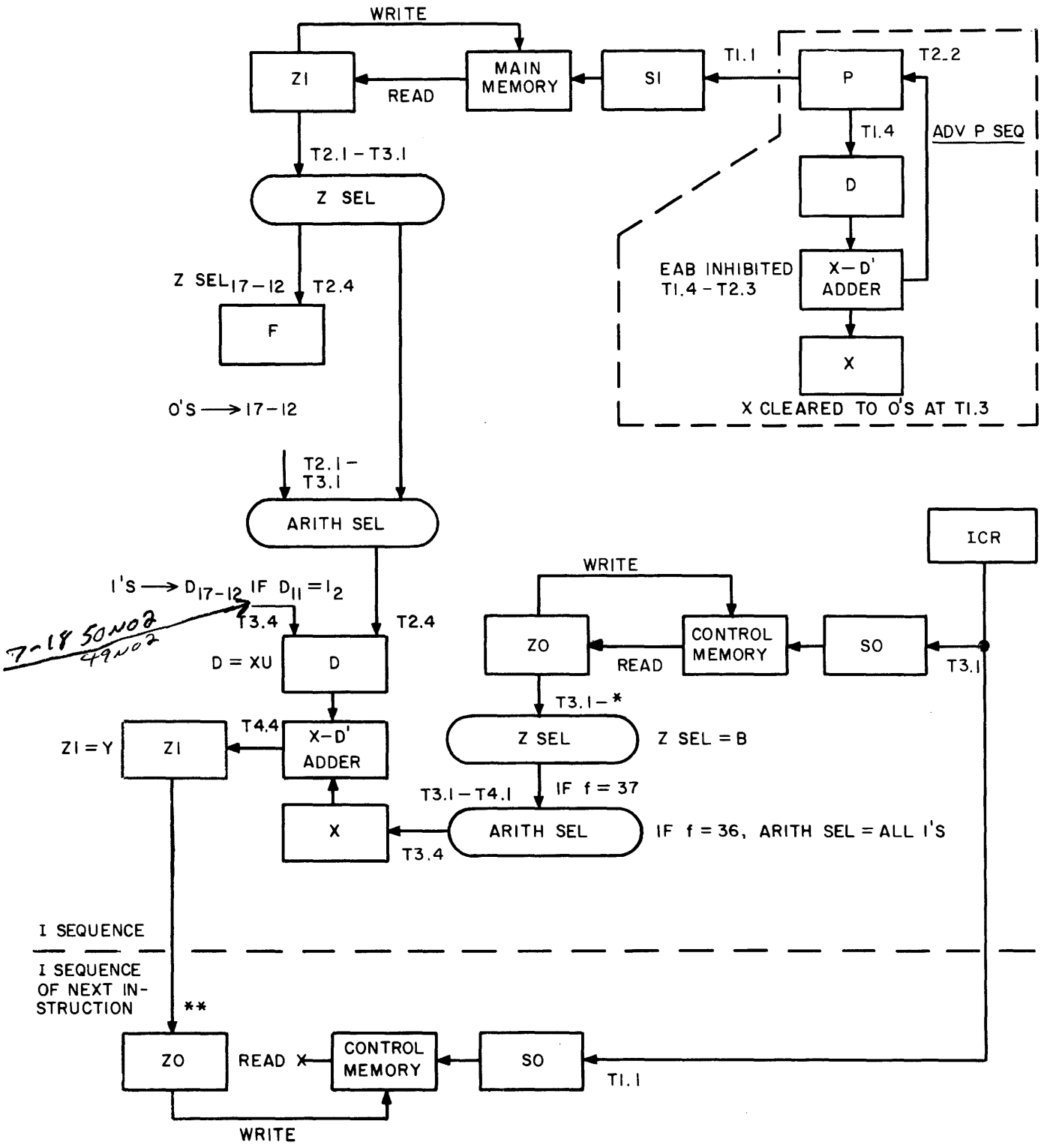
During the next I-sequence, another control memory reference is used, with the address supplied by ICR, to store Z1 in the B register location via ZO. The previous content of B is destroyed because the gating of control memory to ZO is disabled during the read portion of the memory cycle.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.9-1 for a sequential list of essential I and next I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.9-5. SUMMARY

The ENTBK and ENTBKB instructions both use the value XU which is formulated in D. Only the I-sequence and the first portion of the next I-sequence are required to complete the executions of these instructions.



NOTES: *ZO ⟶ Z SEL OCCURS FOR DURATION OF T24 FF CLEAR.
**ZI ⟶ ZO IS TIMED BY CM TIMING.

Figure 5.9-1. I and Next I-Sequence Data Flow for f = 36, 37

TABLE 5.9-1. I AND NEXT I-SEQUENCE ESSENTIAL COMMANDS FOR f = 36, 37

TIME NOTATION	COMMANDS
	I-SEQUENCE
T4.4	Clear S1
T1.1	P → S1, Init Memory, *set Incr P ff
T1.3	Clear Z1, *set OXL11 ff, *clear D, *clear X, clear F
T1.4	$*P_L \rightarrow D_L$, $*P_U \rightarrow D_U$, *set Inhib EAB ff
T2.1	*Clear P, Z1 → Z Sel, Z Sel → Arith Sel, 0's → Arith Sel_{17-12}, *clear Incr P ff
T2.2	*Adder → P
T2.3	Clear X, *clear Inhib EAB ff, *clear OXL11 ff
T2.4	Z Sel_{17-12} → F
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel, drop 0's → Arith Sel_{17-12}, ICR → S0, Init CM, Z0 → Z Sel **, Z Sel → Arith Sel if f = 37
T3.4	1's → D_{17-12} if D_{11} = 1, Arith Sel → X
T4.1	Drop Z Sel → Arith Sel
T4.3	Clear Z1
T4.4	Adder → Z1***, disable CM → Z0
	I-SEQUENCE OF NEXT INSTRUCTION
T1.1	ICR → S0, Init CM
T1.4	Drop disable CM → Z0

* These events are concerned with, or are controlled by, the advance-P subsequence.
** Z0 → Z Sel occurs for duration of T24 ff clear.
*** Z1 → Z0 is timed by control memory timing.

NAME: _____

5.9-6. STUDY QUESTIONS

a. Given: 10N13 constant low level output (input to 11N11, logic diagrams, figure 9-25)

instruction = 376054
ICR = 011_2
B3 = 323142

Considering the given conditions, what is the final content of B3 (control memory address 000003)?

$B3_f$ = _____

SECTION 5 - CONTROL SECTION

5.10. INSTRUCTION EXECUTION OF ENTB, ENTBB

5.10-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 32, 33.

5.10-2. INTRODUCTION

These instructions enter B with the content of memory.

5.10-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.10-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) ENTB, f = 32. This instruction obtains the operand Y from memory. The memory address is U_P if SR is inactive or U_{SR} is active. Y is placed in the B register specified by the content of ICR. The original content of B is destroyed.

b) ENTBB, f = 33. This instruction is similar to f = 32. Y is obtained from memory at address $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR. Y is placed in this same B register. The original content of B is destroyed.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of Y is formulated from U, P, SR, and B.

b) R1-Sequence. The R1-sequence is used to obtain the operand Y with another memory reference.

c) Next I-Sequence. The storage of Y into B is performed during the first portion of the I-sequence for the next sequential instruction.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address of Y.

2. Data Flow Block Diagram. Refer to figure 5.10-1 for a block diagram description of the execution of f = 32, 33.

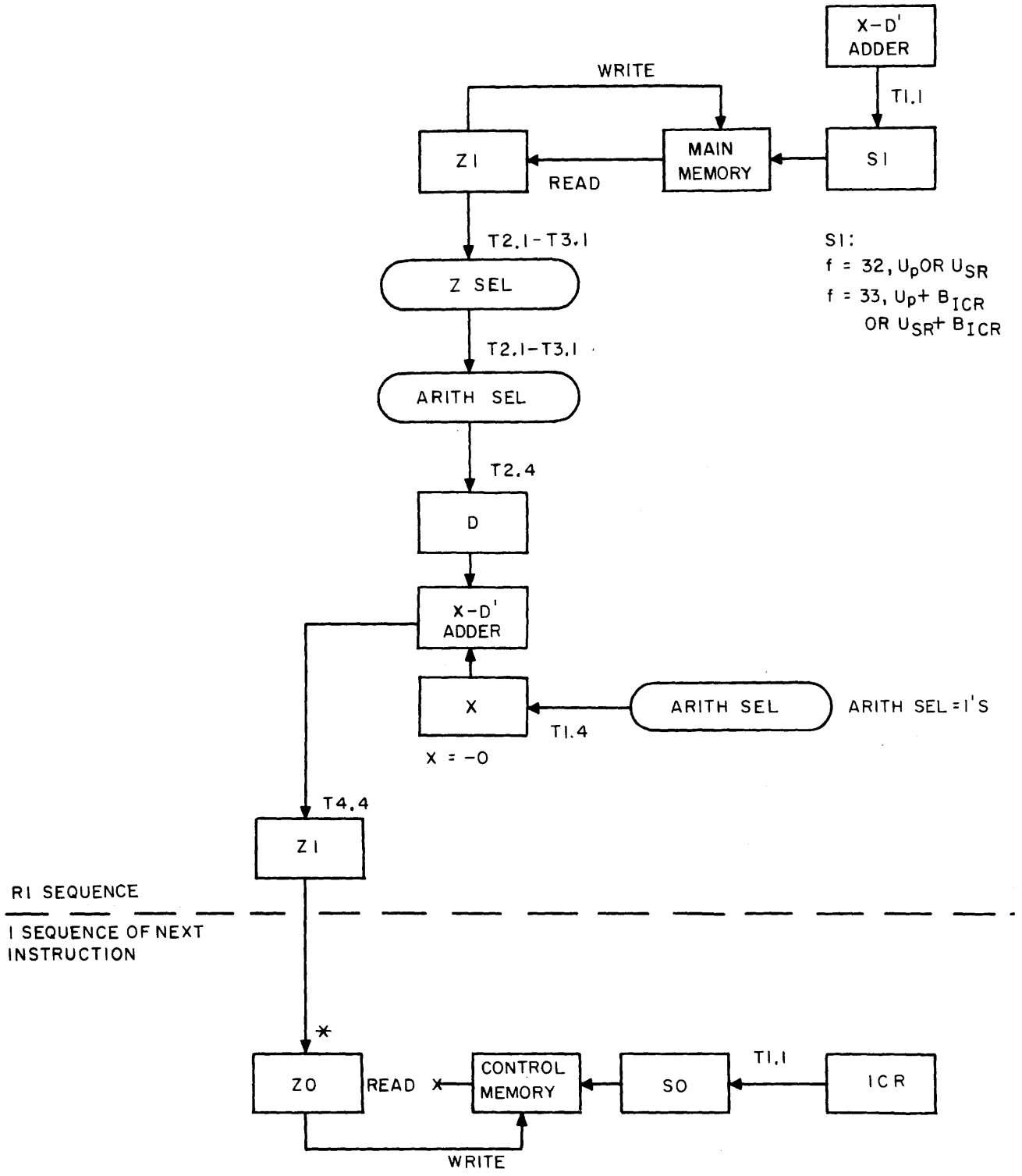
The R1-sequence uses a memory reference to obtain Y. Y is placed in D from where it is applied to one side of the X-D' adder. X is set to all 1's. Therefore, the X-D' adder outputs (-0) - Y' which is effectively Y. During the next I-sequence, Z0 receives Y from Z1 and a control memory reference is used to store this value in B. The B register address is put in S0 from ICR. The previous content of B is destroyed because the gating of control memory to Z0 is disabled during the read portion of the memory cycle.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.10-1 for a sequential list of essential R1- and I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.10-5. SUMMARY

The ENTB and ENTBB instructions both use the value U_P or U_{SR} which is formulated in D during the I-sequence. The R1-sequence and the first portion of the next I-sequence are required to complete the executions of these instructions.



NOTE: * Z1 ⟶ Z0 IS TIMED BY CM TIMING.

Figure 5.10-1. R1 and Next I-Sequence Data Flow for f = 32, 33

TABLE 5.10-1. R1 AND NEXT I-SEQUENCE ESSENTIAL COMMANDS FOR f = 32, 33

TIME NOTATION	COMMANDS
	R1 SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init Memory
T1.3	Clear X, clear Z1
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
T4.3	Clear Z1
T4.4	Adder → Z1*, disable CM → Z0
	I SEQUENCE OF NEXT INSTRUCTION
T1.1	ICR → S0, Init CM
T1.4	Drop Disable CM → Z0

*Z1 → Z0 is timed by control memory timing.

SECTION 5 - CONTROL SECTION

5.11. INSTRUCTION EXECUTION OF ENTAU, ENTAUB, ENTAL, ENTALB, ADDAL, ADDALB, SUBAL, SUBALB

5.11-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 10-17.

5.11-2. INTRODUCTION

These instructions enter either AU or AL with the content of memory alone, +AU, +AL, -AU, or -AL.

5.11-3. REFERENCES

a. UNIVAC 1219 Technial Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.11-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) ENTAU, f = 10. This instruction obtains the operand Y from memory. The memory address is U_P if SR is inactive or U_{SR} if SR is active. Y is placed in AU. The original content of AU is destroyed.

b) ENTAUB, f = 11. Except for the memory address, this instruction is the same as f = 10. The address of Y is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR.

c) ENTAL, f = 12. Except for the destination of Y, this instruction is the same as f = 10. Y is placed in AL, and the original content of AL is destroyed. AU is not disturbed.

d) ENTALB, f = 13. Except for the address of Y, this instruction is the same as f = 12. The address of Y is the same as for f = 11.

e) ADDAL, f = 14. This instruction obtains the operand Y from memory. The memory address is U_P if SR is inactive or U_{SR} if SR is active. Y is added to AL and their sum is placed in AL. The original content of AL is destroyed. The Overflow flip-flop is set if an overflow occurs.

f) ADDALB, f = 15. Except for the memory address, this instruction is the same as f = 14. The address of Y is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR. The Overflow flip-flop is set if the sign of $AL_f \neq$ sign of AL_i.

g) SUBAL, f = 16. Except for the arithmetic operation performed, this instruction is the same as f = 14. Y is subtracted from AL and their difference is placed in AL. The Overflow flip-flop is set if an overflow condition occurs.

h) SUBALB, f = 17. Except for the address of Y, this instruction is the same as f = 16. The address of Y is the same as for f = 15.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of Y is formulated from U, P, SR, and B.

b) R1-Sequence. The R1-sequence is used to obtain the operand from memory, perform the necessary arithmetic operation, and place the result in either AU or AL.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X - D' adder is outputting the address of Y.

2. Data Flow Block Diagram. Refer to figure 5.11-1 for a block diagram description of the execution of f = 10-17.

The R1-sequence uses a memory reference to obtain Y. If f = 10-15, D receives Y. If f = 16, 17 D receives Y'. X is set to AL for f = 14-17; otherwise, X contains -0. The adder output of X - D' is placed in either AU or AL.

If f = 10-13, the adder output is actually -0-Y' which is effectively -0+Y. If f = 14, 15, the adder output is actually AL - Y' which is effectively AL + Y. If f = 16, 17, the adder output is actually AL - Y' which is effectively AL-Y.

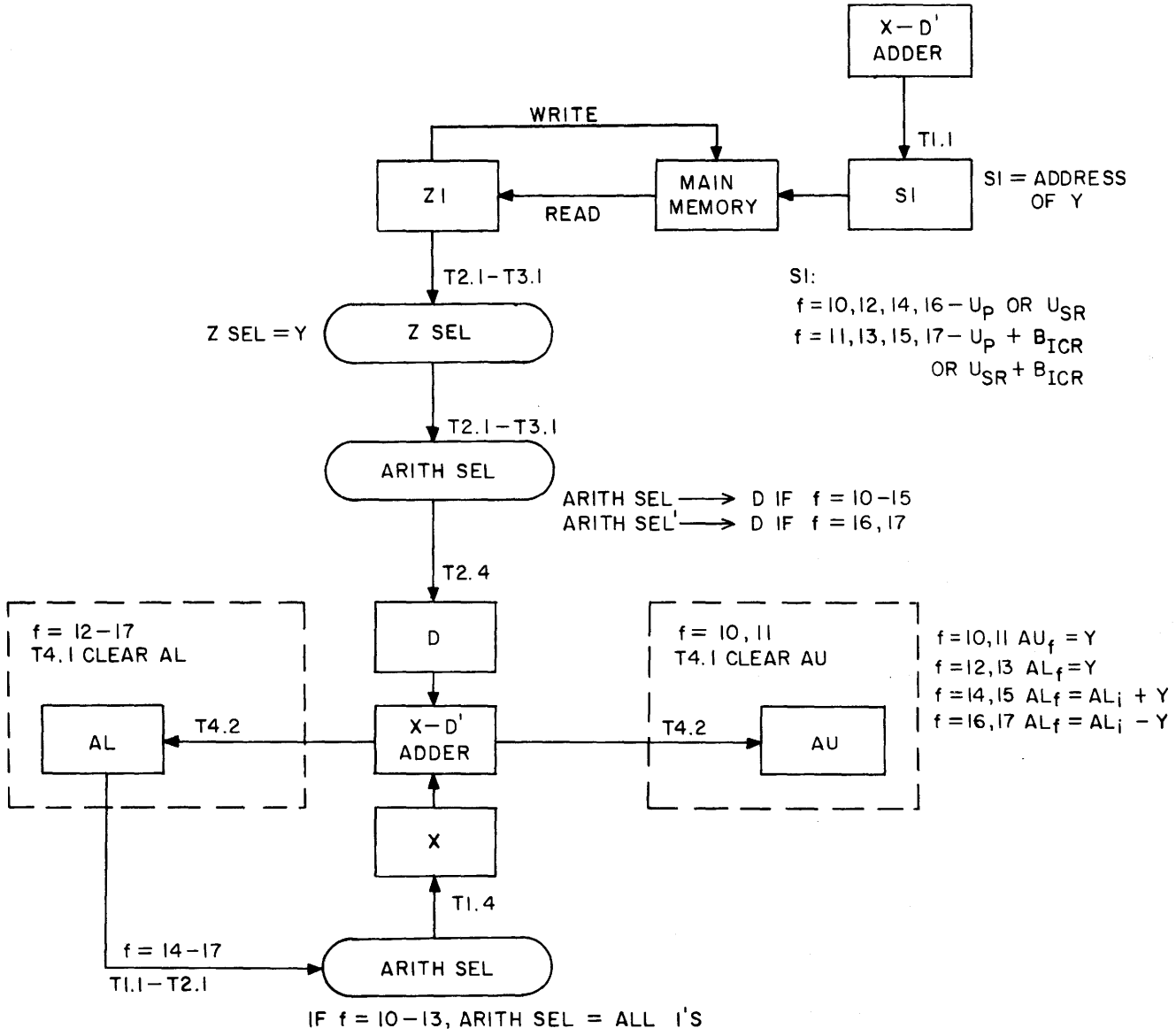
For f = 14-17, the overflow flip-flop is used to record that the sign of $AL_i \neq AL_f$. The state of this flip-flop can be later sensed by the execution of f = 50:52 or 50:53 instruction. The setting of the Overflow flip-flop is conditioned by the X - D' adder. The X - D' adder is analyzed in a later sheet.

As discussed in a later sheet, the operand Y could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.11-1 for a sequential list of essential R1-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.11-5. SUMMARY

The f = 10-17 instructions use the value U_P or U_{SR} which is formulated in D during the I-sequence. The R1-sequence is required to complete the executions of these instructions.



T3.2 SET OVERFLOW FF IF $(f = 14-17) \cdot (\text{SIGN OF } AL_f \neq \text{SIGN OF } AL_i)$*

NOTE: * THE OVERFLOW CONDITION IS INDICATED FROM THE X − D' ADDER BY:
$(X_{17} = 0_2 \cdot D_{17}' = 1_2 \cdot$ NO BORROW REQUEST ⟶ BIT 17) +
$(X_{17} = 1_2 \cdot D_{17}' = 0_2 \cdot$ BORROW REQUEST ⟶ BIT 17)

Figure 5.11-1. R1-Sequence Data Flow for f = 10-17

TABLE 5.11-1. R1-SEQUENCE ESSENTIAL COMMANDS FOR f = 10-17

TIME NOTATION	COMMANDS f =	10,11	12,13	14,15	16,17
T4.4	Clear S1	X	X	X	X
T1.1	Adder → S1, Init memory	X	X	X	X
	AL →Arith Sel			X	X
T1.3	Clear X, clear Z1	X	X	X	X
T1.4	Arith Sel →X	X	X	X	X
T2.1	Z1 →Z Sel, Z Sel →Arith Sel	X	X	X	X
	Drop AL →Arith Sel			X	X
T2.3	Clear D	X	X	X	X
T2.4	Arith Sel → D	X	X	X	
	Arith Sel' → D				X
T3.1	Drop Z1 →Z Sel, drop Z Sel → Arith Sel	X	X	X	X
T3.2	Set Overflow ff if sign of $AL_f \neq$ sign of AL_i			X	X
T4.1	Clear AU	X			
	Clear AL		X	X	X
T4.2	Adder →AU	X			
	Adder →AL		X	X	X

NAME: _____

5.11-6. STUDY QUESTIONS:

a. Given: instruction = 140500
SR = 010012_2
content of address 010500 = 000003
AL_i = 000050

Assume each of the following malfunctions to occur individually immediately at the completion of the I-sequence. Indicate AL_f (after R1-sequence) for each of these conditions.

1. Pin 8 of 0XF01 ff grounded (logic diag., figure 9-40)
AL_f = _____

2. 17N02 grounded output (logic diag., figure 9-18)
AL_f = _____

3. 10N05 constant low level output (logic diag., figure 9-20)
AL_f = _____

4. 00N05 grounded output (logic diag., figure 9-20)
AL_f = _____

5. 00N03 grounded output (logic diag., figure 9-19)
AL_f = _____

6. 28N02 constant low level output (logic diag., figure 9-18)
AL_f = _____

7. 29N01 constant low level output (logic diag., figure 9-17)
AL_f = _____

b. Given: 06G22 grounded output (logic diag., figure 9-22)
SR = 010011_2
content of address 015000 = 326374

Address	Instruction
006000	105000
006001	125000

Immediately after the execution of the instruction at address 006000, what is the content of AU?
AU_f = _____

c. Given: 09N13 grounded output (logic diag., figure 9-23)
SR = 010002_2
content of address 002450 = 000023

Address	Instruction
006000	102450

After the execution of the instruction at address 006000, what is the final content of AU? Assume that Z1 is initially cleared.

AU_f = _____

d. Given: 10N02 constant low level output (logic diag., figure 9-18)
SR = 10000_2
content of address 002300 = 560143

Address	Instruction
001000	102300

After the execution of the instruction at address 001000, what is the final content of AU?

AU_f = _____

SECTION 5 - CONTROL SECTION

5.12. INSTRUCTION EXECUTION OF ADDA, ADDAB, SUBA, SUBAB

5.12-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 20-23.

5.12-2. INTRODUCTION

These instructions add to or subtract from the 36-bit value held in AU and AL the 36-bit value formulated by the content of two consecutive addresses.

5.12-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, Tables 4-11, 4-12, and 4-13.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.12-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) ADDA, f = 20. This instruction obtains a 36-bit operand from two consecutive memory addresses. The address of the least significant 18 bits is U_P if SR is inactive or U_{SR} if SR is active. The address of most significant 18 bits is the next sequential address. The 36-bit operand is added to the 36-bit value of AU and AL together with AU being the more significant half. The sum is placed in AU and AL. Their original values are destroyed.

An overflow condition from AU will not cause an end-around effect upon AL, but this overflow condition will cause the Borrow Test flip-flop to be set. The Overflow flip-flop is set if an overflow occurs. The conditions of these flip-flops can be later sensed by the f = 50:51, 50:52, or 50:53 instructions which can cause the program to execute the necessary operations.

If the address of the least significant 18 bits of the operand is not an even number, this same address will be used to obtain the most significant 18 bits of the operand. In this case, the same 18-bit number will be added to both AU and AL.

b) ADDAB, f = 21. Except for the memory addresses, this instruction is the same as f = 20. The address of the 18 least significant operand bits is either U_P + B or U_{SR} + B depending upon the activeness of SR. The B register is specified by ICR.

c) SUBA, f = 22. Except for the arithmetic operation performed, this instruction is the same as f = 20. The arithmetic operation is the same as for f = 22.

d) SUBAB, f = 23. Except for the arithmetic operation performed, this instruction is the same as f = 21. The arithmetic operation is the same as for f = 22.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of the 18 most significant bits of the operand is formulated from U, P, SR, and B.

b) R1-Sequence. The R1-sequence is used to obtain the 18 least significant bits of the operand from memory, perform the necessary arithmetic operation with AL, and place the result in AL. The end-around effect of an AL overflow is prevented but is recorded.

c) R1 and R2-Sequences in Parallel. R1 and R2-sequences are executed together to operate upon AU. The 18 most significant bits of the operand are obtained. The necessary arithmetic operation is performed with AU, and the result is placed in AU. An overflow from AL is considered during the arithmetic operation. An overflow from AU is prevented but recorded.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address of the 18 least significant bits of the operand.

2. Data Flow Block Diagram. Refer to figure 5.12-1 for a block diagram description of the execution of f = 20-23.

The R1-sequence uses a memory reference to obtain the 18 least significant bits of the operand. Either this 18-bit value or its complement is placed in D depending upon the arithmetic operation to be performed. X receives AL. The X-D' adder output is placed in AL. For f = 20,21, AL receives AL_i - Y' which is effectively AL_i + Y. For f = 22, 23, AL receives AL_i - Y'' which is effectively AL_i - Y.

End-around borrow is inhibited since a borrow from AL is to be taken from AU. The Borrow Test flip-flop is set to record the end-around borrow and apply it to the operation involving AU.

The R1 and R2-sequences in parallel use a memory reference to obtain the 18 most significant bits of the operand. The memory address is one greater than the address used by the previous memory reference. This address is formulated by not clearing out the previous address in S1 and setting $S1_{00}$ to a 1_2. If the first address in S1 is not an even number, $S1_{00}$ would already contain a 1_2 and both memory references would use the same address.

The arithmetic operation performed during the R1 and R2-sequences is the same as executed during the previous R1-sequence except that AU is used. An end-around borrow is inserted in the X-D' adder if the Borrow Test flip-flop is set, which would indicate that the AL operation produced an end-around borrow. The inserted

end-around borrow effectively subtracts 1 from the difference of X-D'.

End-around borrow from AU is inhibited although it is recorded by the Borrow Test flip-flop being set. The Overflow flip-flop is set if an overflow occurs. The setting of this flip-flop is conditioned by the X-D' Adder. The X-D' adder is analyzed in a later sheet. If overflow occurs, it is necessary to sense this condition with an f = 50:52 or 50:53 instruction. The f = 50:51 instruction can be executed to sense the end-around borrow which is inhibited. The program can insert this borrow by executing another add or subtract A (AU and AL combined) instruction with the 36-bit operand being +1.

As discussed in a later sheet, the operand could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.12-1 for a sequential list of essential R1 and R1/R2-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.12-5. SUMMARY:

The f = 20-23 instructions use the value U_P or U_{SR} which is formulated in D during the I-sequence. The R1-sequence and R1 in parallel with the R2-sequence are required to complete the executions of these instructions.

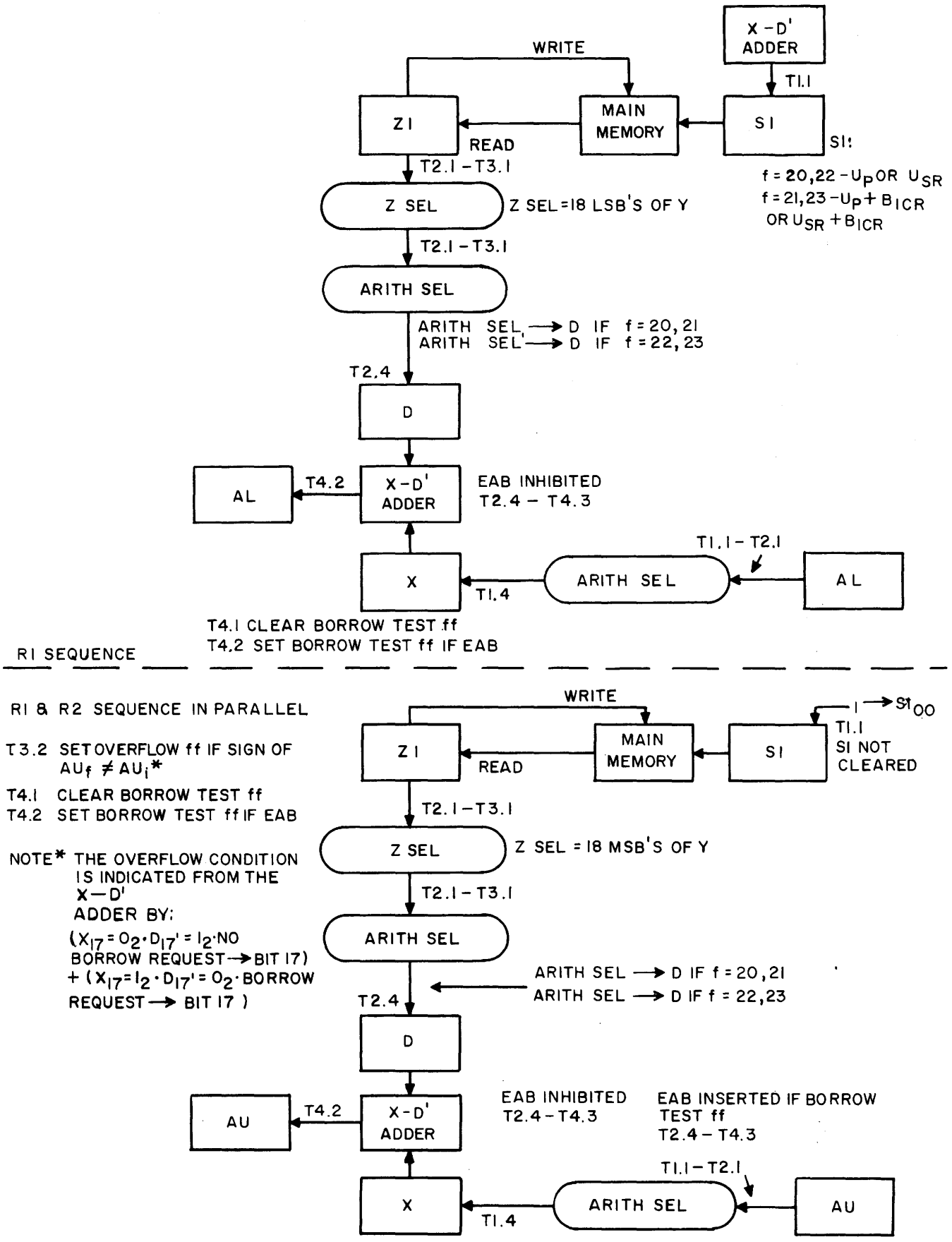


Figure 5.12-1. R1 and R1/R2-Sequence Data Flow for f = 20-23

TABLE 5.12-1. R1 AND R1/R2 ESSENTIAL COMMANDS FOR f = 20-23

TIME NOTATION	COMMANDS
	R1 SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init memory, AL → Arith Sel
T1.3	Clear X, clear Z1
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel, drop AL → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D if f = 20,21, Arith Sel' → D if f = 22,23 set Inhib EAB ff
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
T4.1	Clear AL, clear Borrow Test ff
T4.2	Adder → AL, set Borrow Test ff if EAB
T4.3	Clear Inhib EAB ff
	R1 AND R2 SEQUENCES IN PARALLEL
T1.1	1 → $S1_{00}$, Init memory, AU → Arith Sel
T1.3	Clear X, clear Z1
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel, drop AU → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D if f = 20, 21; Arith Sel' → D if f = 22,23 set Insert EAB ff if Borrow Test ff set, set Inhib EAB ff
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
T3.2	Set Overflow ff if sign of $AU_f \neq$ sign of AU_i
T4.1	Clear AU, clear Borrow Test ff
T4.2	Adder → AU, set Borrow Test ff if EAB
T4.3	Clear Insert EAB ff, clear Inhib EAB ff

NAME: _____

5.12-6. STUDY QUESTIONS:

a. Given: 21E51 grounded output (logic diagrams, Figure 9-30)

content of address 021000 = 450000
content of address 021001 = 670002
AU_i = 130015
AL_i = 457675
SR = 00000_2

Address	Instruction
020000	201000

After the execution of the instruction at address 020000, what is the content of AU and AL?

AU_f = _____ AL_f = _____

SECTION 5 - CONTROL SECTION

5.13. INSTRUCTION EXECUTION OF CL, CLB, STRB, STRBB, STRAL, STRALB, STRAU, STRAUB

5.13-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 40-47.

5.13-2. INTRODUCTION

These instructions store in memory either 0's, B, AL or AU.

5.13-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-14.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.13-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) CL, f = 40. This instruction clears the content of memory address U_P if SR is inactive or U_{SR} if SR is active. The clearing is effected by storing 0's. The original content of the memory address is destroyed.

b) CLB, f = 41. Except for the memory address, this instruction is the same as f = 40. The storage address is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR.

c) STRB, f = 42. This instruction stores the content of B in memory at address U_P if SR is inactive or U_{SR} if SR is active. The B register is specified by ICR. The original content of the memory address is destroyed. The content of B is not disturbed.

d) STRBB, f = 43. Except for the memory address, this instruction is the same as f = 42. The storage address is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register which is used to formulate the storage address is the same one which is stored and is specified by ICR.

e) STRAL, f = 44. This instruction stores the content of AL in memory at address U_P if SR is inactive or U_{SR} if SR is active. The original content of the memory address is destroyed. The content of AL is not disturbed.

f) STRALB, f = 45. Except for the memory address, this instruction is the same as f = 44. The storage address is either U_P + B or U_{SR} + B depending upon the activeness of SR. The B register is specified by ICR.

g) STRAU, f = 46. This instruction stores the content of AU in memory at address U_P if SR is inactive or U_{SR} if SR is active. The original content of the memory address is destroyed. The content of AU is not disturbed.

h) STRAUB, f = 47. Except for the memory address, this instruction is the same as f = 46. The storage address is either U_P + B or U_{SR} + B depending upon the activeness of SR. The B register is specified by ICR.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the storage address is formulated from U, P, SR, and B.

b) W-Sequence. The W-sequence performs the storage of 0's , B, AL or AU by executing another memory reference.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the storage address.

2. Data Flow Block Diagram. Refer to figure 5.13-5 for a block diagram description of the execution of f = 40-47.

The W-sequence uses a memory reference to perform the storage. The previous content of the memory address is destroyed because the gating of memory to Z1 is disabled during the read portion of the memory cycle. The value to be stored is placed in Z1 from store-select.

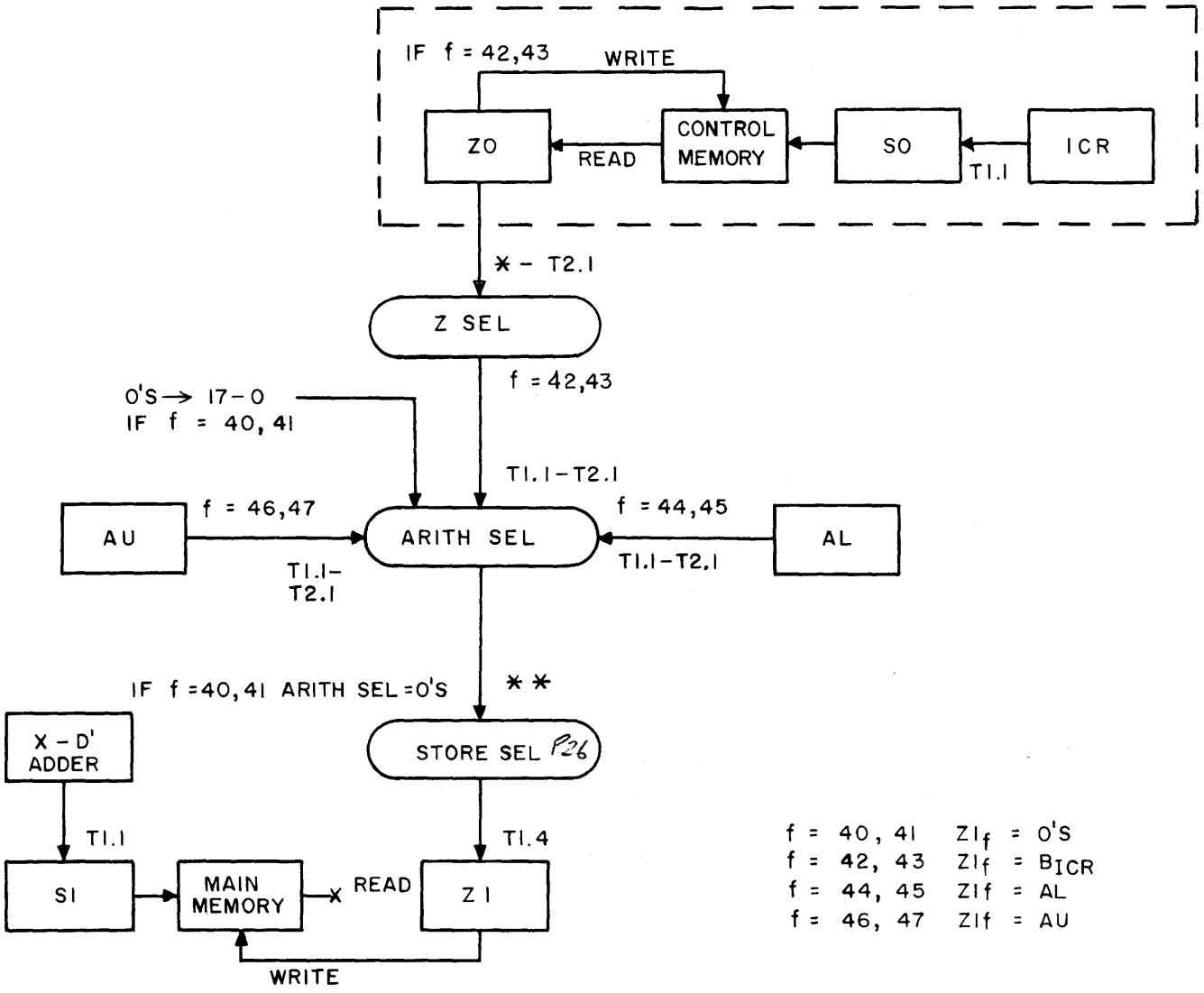
If f = 42, 43, a control memory reference is used to obtain the content of B. SO is set to the B register address from ICR.

As discussed in a later sheet, the storage address could be in bootstrap or control memory. Since the content of bootstrap memory cannot be altered by the program, storage into its addresses is useless.

3. Essential Commands. Refer to table 5.13-1 for a sequential list of essential W-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.13-5. SUMMARY

The f = 40-47 instructions use U_P or U_{SR} which is formulated in D during the I-sequence. The W-sequence is required to complete the execution of these instructions.



SI:

f=40,42,44,46-U_P OR U_{SR}

f=41,43,45,47-U_P + B_{ICR} OR U_{SR} + B_{ICR}

NOTES: * ZO → Z SEL OCCURS FOR DURATION OF T24 CLEAR.

** ARITH SEL → STORE SEL OCCURS FOR ENTIRE W SEQUENCE.

Figure 5.13-1. W-Sequence Data Flow for f = 40-47

TABLE 5.13-1. W-SEQUENCE ESSENTIAL COMMANDS FOR f = 40-47

TIME NOTATION	COMMANDS f =	40, 41	42, 43	44, 45	46, 47
T4.4	Clear S1	X	X	X	X
T1.1	Adder → S1, Init Memory	X	X	X	X
	ICR → S0, Init CM, *Z Sel → Arith Sel		X		
	0's → Arith Sel	X			
	AL → Arith Sel			X	
	AU → Arith Sel				X
T1.3	Clear Z1	X	X	X	X
T1.4	**Store Sel → Z1, disable Mem → Z1	X	X	X	X
T2.1	Drop Z Sel → Arith Sel		X		
	Drop 0's → Arith Sel	X			
	Drop AL → Arith Sel			X	
	Drop AU → Arith Sel				X
T2.4	Drop disable Mem → Z1	X	X	X	X

NOTES: *Z0 → Z Sel occurs for duration of T24 ff clear.

**Arith Sel → Store Sel occurs for entire W-sequence.

NAME:_____

5.13-6. STUDY QUESTIONS

a. Given: 11F75 constant low level output (logic diagrams, figure 9-48)
instruction = 402000
SR = 11001_2

After the execution of the given instruction, what are the contents of SR and memory at address 112000?

SR_f = _____
$Memory_f$ at address 112000 = _____

b. Given: 10N13 constant low level output (logic diagrams, figure 9-23)
instruction = 462000
SR = 11001_2
AU = 342015
initial content of address 112000 = 411312

After the execution of the given instruction, what is the content of memory at address 112000?

$Memory_f$ at address 112000 = _____

c. Given: 93F02 constant low level output (logic diagrams, figure 9-44)
instruction = 462000
SR = 01000_2
AU = 321456
AL = 753663

After the execution of the given instruction, what is the content of memory at address 002000?

$Memory_f$ at address 002000 = _____

SECTION 5 - CONTROL SECTION

5.14. INSTRUCTION EXECUTION OF STRICR, STRADR, STRSR

5.14-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 72, 74, 75.

5.14-2. INTRODUCTION

These instructions store the content of SR, ICR, or the lower 12 bits of AL in memory.

5.14-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-14.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.14-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) STRICR, f = 72. This instruction stores the content of ICR in memory at address U_P. The initial six least significant bits of memory are destroyed. The initial 12 most significant bits of memory are retained. If ICR = 000_2, bit 3 in memory is set to 1_2.

b) STRADR, f = 74. This instruction stores the 12 least significant bits of AL in memory at address U_P. The initial 12 least significant bits of memory are destroyed. The initial six most significant bits of memory are retained. AL is not disturbed.

c) STRSR, f = 75. This instruction stores the content of SR in memory at address U_P. The initial six least significant bits of memory are destroyed. The initial 12 most significant bits of memory are retained. After storage, SR is cleared which deactivates it.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the storage address is formulated by U and P.

b) W-Sequence The W-sequence performs the storage into memory at address U_P.

b. Detailed Analysis

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the storage address U_P.

2. Data Flow Block Diagram. Refer to figure 5.14-1 for a block diagram description of the execution of f = 72, 74, 75.

The W-sequence uses a memory reference to perform the storage. The value to be stored (ICR, AL, or SR) is applied to arithmetic-select. The bit positions not to be stored are masked-out in arithmetic-select with 0's. Z1 is set by arithmetic-select. Those original bits in memory which are to be replaced by the stored value are destroyed by not being gated to Z1 during the read portion of the memory cycle.

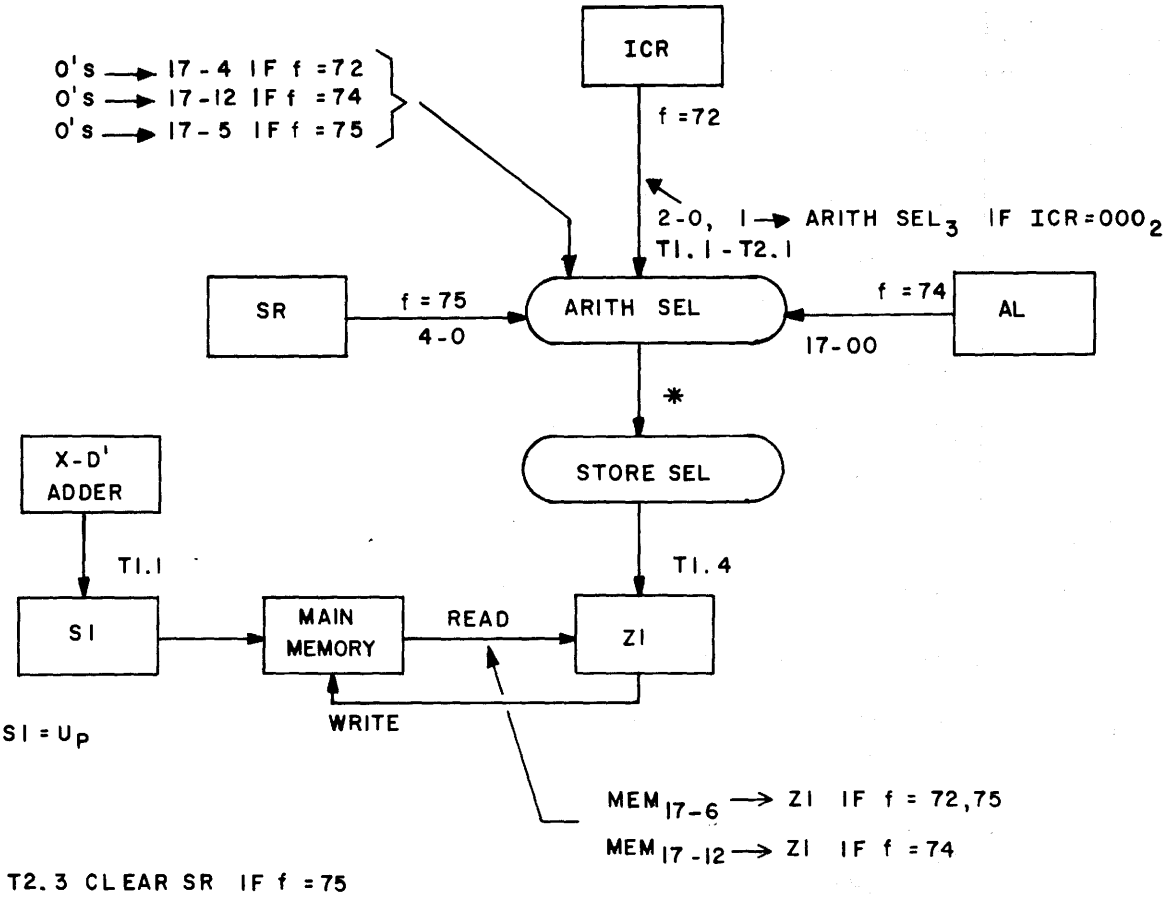
If f = 75, SR is cleared after it is stored. This clearing deactivates SR.

As discussed in a later sheet, the storage address could be in bootstrap or control memory. Since bootstrap has nondestructive read-out, storage into this memory is useless.

3. Essential Commands. Refer to table 5.14-1 for a sequential list of essential W-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.14-5. SUMMARY

The STRICR, STRADR, and STRSR instructions use the value U_P which is formulated in D during the I-sequence. The W-sequence is required to complete the executions of these instructions.



NOTE: * ARITH SEL → STORE SEL OCCURS FOR ENTIRE W SEQUENCE.

$f = 72$ $MEMORY_f = \underbrace{17-6}_{MEMORY_i}\ \underbrace{5-4}_{0's}\ 3\ \underbrace{2\ 0}_{ICR}$

0 IF $ICR \neq 000_2$

1 IF $ICR = 000_2$

$f = 74$ $MEMORY_f = \underbrace{17-12}_{MEMORY_i}\ \underbrace{11-0}_{AL_{11-0}}$

$f = 75$ $MEMORY_f = \underbrace{17-6}_{MEMORY_i}\ 5\ \underbrace{4-0}_{SR}$

0

Figure 5.14-1. W-Sequence Data Flow for f = 72, 74, 75

TABLE 5.14-1.W SEQUENCE ESSENTIAL COMMANDS FOR f = 72, 74, 75

TIME NOTATION	COMMANDS f =	72	74	75
T4.4	Clear S1	X	X	X
T1.1	Adder → S1, Init memory	X	X	X
	ICR → Arith Sel_{2-0}, 0's → Arith Sel_{17-4}, 1 → Arith Sel_3 if ICR = 000_2	X		
	AL → Arith Sel, 0's → Arith Sel_{17-12}		X	
	SR → Arith Sel_{4-0}, 0's → Arith Sel_{17-5}			X
T1.3	Clear Z1	X	X	X
T1.4	*Store Sel → Z1, disable Mem_{5-0} → Z1	X	X	X
	Disable Mem_{11-6} → Z1		X	
T2.1	Drop ICR → Arith Sel_{2-0}, drop 0's → Arith Sel_{17-4}, drop 1 → Arith Sel_3	X		
	Drop AL → Arith Sel, drop 0's → Arith Sel_{17-12}		X	
	Drop SR Arith Sel_{4-0}, drop 0's Arith Sel_{17-5}			X
T2.3	Clear SR			X
	Drop disable Mem_{5-0} → Z1	X	X	X
	Drop disable Mem_{11-6} → Z1		X	

* Arith Sel → Store Sel occurs for entire W-sequence.

NAME: _____

5.14-6. STUDY QUESTIONS

a. Given: 10G45 grounded output (logic diagrams, figure 9-39)
ICR = 100_2

Address	Instruction
033356	726632

After the execution of the given instruction, what is the content of memory at address 036632?

$Memory_f$ at address 036632 = _____

SECTION 5 - CONTROL SECTION

5.15. INSTRUCTION EXECUTION OF SLSU, SLSUB, SLSET, SLCL, SLCP

5.15-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 04, 05, 51-53.

5.15-2. INTRODUCTION

These instructions selectively control the value placed in each bit of AL. As determined by the instruction, the final content of AL is controlled by the content of memory AL_i, and AU. In this sheet, the "+" sign if used to indicate the logical inclusive OR function.

5.15-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragrpah 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logical diagrams).

5.15-4. INFORMATION

a. General Description

1. Instruction Interpretation

a) SLSU, f = 04. This instruction selectively substitutes the bits of the operand Y for the bits of AL as controlled by AU. The origin of Y is memory at address U_P if SR is inactive or U_{SR} if SR is active. Where AU is a 1_2, the corresponding bit of Y is put in AL in place of the initial bit of AL. Where AU is a 0_2, the corresponding bit of AL is retained unchanged. AU is not disturbed.

The logical term for AL_f $(AU' \cdot AL_i) + (AU \cdot Y)$.

b) SLSUB, f = 05. Except for the memory address, this instruction is the same as f = 04. The address of Y is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR.

c) SLSET, f = 51. This instruction selectively sets the bits of AL as controlled by the operand Y. The origin of Y is memory at address U_P. Where Y is a 1_2, the corresponding bit of AL is set to a 1_2. Where Y is a 0_2, the corresponding bit of AL is retained unchanged.

The logical term for AL_f is $AL_i + Y$.

d) SLCL, f = 52. This instruction selectively clears the bits of AL as controlled by the operand Y. The origin of Y is memory at address U_P. Where Y is a 0_2, the corresponding bit of AL is cleared to a 0_2. Where Y is a 1_2, the corresponding bit of AL is retained unchanged.

The logical term for AL_f is $AL_i \cdot Y$.

e) SLCP, f = 53. This instruction selectively complements the bits of AL as controlled by the operand Y. The origin of Y is memory at address U_P. Where Y is a 1_2, the corresponding bit of AL is complemented. Where Y is a 0_2, the corresponding bit of AL is retained unchanged.

The logical term for AL_f is $(AL_i' \cdot Y) + AL_i \cdot Y')$.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of Y is formulated from U, P, SR, and B.

b) R1-Sequence. The R1-sequence is used to obtain the operand from memory, perform the necessary operation, and place the result in AL.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously decribed. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D'Adder is outputting the address of the operand.

2. Data Flow Block Diagram.

a) SLSU and SLSUB Instructions, f = 04, 05. Refer to figure 5.15-1 for a block diagram description of the execution of f = 04, 05.

The R1-sequence uses a memory reference to obtain the operand Y. D receives the complement of AU from arithmetic-select at T1.4 time. Without clearing, D also receives Y at T2.4 time. D therefore applies to one side of the X-D' adder the inclusive OR function value of AU' + Y.

X receives AU at T1.4 time and also, without clearing, AL at T3.4 time. X therefore applies to the other side of the X-D' adder the inclusive OR function value of AU + AL.

AL is cleared and set to the value of X-D' from the adder which is actually $(AU + AL_i) - (AU' + Y)'$. This term can best be expressed as $(AU + AL_i) - (AU \cdot Y')$.

This final result in AL should be such that each bit position is set to the corresponding bit of either AL_i or Y depending upon the value in AU. No bit position value is in any way dependent upon any other bit position. So as to use the arithmetic adder, inter-action among the bit positions must be prevented by insuring that borrow conditions will not occur. A borrow condition exists when in the same bit position the subtraction is $0_2 - 1_2$. For these instructions the adder subtracts $(AU + AL_i) - (AU \cdot Y')$. If the value of $AU + AL_i$ is equal to 0_2, AU must

be equal to 0_2 and, therefore, the value of AU·Y' must also be equal to 0_2. Thus, whenever X is equal to 0_2, D' is also equal to 0_2. A borrow condition can never exist during this logical operation, and no bit position can be affected by another position.

To analyze the value in AL_f of $(AU + AL_i) - (AU \cdot Y')$, refer to table 5.15-1. Develop the AL_f values in this table by considering each configuration of values for AU, AL_i, and Y.

TABLE 5.15-1. TRUTH TABLE OF $(AU + AL_i) - (AU \cdot Y') = AL_f$

AU	AL_i	Y	AL_f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

This same truth table also holds true for the AL_f value of $(AU' \cdot AL_i) + (AU \cdot Y)$. This term satisfies the expression "substitute Y for AL where AU is a 1_2". If AU if a 0_2, the value in the corresponding AL_f bit position is the same as the initial AL value.

As discussed in a later sheet, the operand could be obtained from bootstrap or control memory.

b) SLSET and SLCL Instructions, f = 51, 52. Refer to figure 5.15-2 for a block diagram description of the execution of f = 51, 52.

The R1-sequence uses a memory reference to obtain the operand Y. Y is applied to arithmetic-select from Z-select. If f = 52, AL is also applied to arithmetic-select which formulates the logical AND function value of $AL_i \cdot Y$. This value is placed in D from where it is applied to one side of the X-D' adder. The other side of the adder senses -0 from X. AL is cleared and receives the X-D' value of $(-0)-(AL_i \cdot Y)'$ which is effectively $AL_i \cdot Y$. This term satisfies the expression "clear AL where Y is a 0_2". If Y is a 1_2, the value in the corresponding AL_f bit position is the same as the initial AL value.

If f = 51, operations are very similar to those for f = 52. The logical AND function is not used. D simply receives Y and applies it to one side of the X-D' Adder. AL is not cleared of its initial value and receives the X-D' value of (-0)-Y' which is

effectively Y. The inclusion OR function of AL_i + Y is formulated in AL_f. This term satisfies the expression "set AL to a 1_2 where Y is a 1_2". If Y is a 0_2, the value in the corresponding AL_f bit position is the same as the initial AL value.

As discussed in a later sheet, the operand could be obtained from bootstrap or control memory.

c) SLCP Instruction, f = 53. Refer to figure 5.15-3 for a block diagram description of the execution of f = 53.

The R1-sequence uses a memory reference to obtain the operand Y. D receives the complement of AL_i from arithmetic-select at T1.4 time. Without clearing, D also receives the complement of Y at T2.4 time. D therefore applies to one side of the X-D' adder the inclusive OR function value of Y' + AL_i'.

X receives AL at T1.4 time and also, without clearing, Y at T2.4 time. X therefore applies to the other side of the X-D' Adder the inclusive OR function value of Y + AL_i.

AL is cleared and set to the value of X-D' from the adder which is actually (Y + AL_i')'. This term can best be expressed as (Y + AL_i) - (Y·AL_i).

Borrow conditions in the adder are prevented to insure that any AL bit position value is not affected by any other bit position. A borrow condition exists when in the same bit position the subtraction is 0_2-1_2. For this instruction the adder subtracts (Y + AL_i) - (Y · AL_i). If the value of Y + AL_i is equal to 0_2, AL_i must be equal to 0_2; and, therefore, the value of Y · AL_i must also be equal to 0_2. Thus, whenever X is equal to 0_2, D' is also equal to 0_2. A borrow condition can never exist during this logical operation, and no bit position can be affected by another bit position.

To analyze the value in AL_f of (Y + AL_i) - (Y · AL_i), refer to table 5.15-2. Develop the AL_f values in this table by considering each configuration of values for AL_i and Y.

TABLE 5.15-2. TRUTH TABLE OF (Y + AL_i) - (Y·AL_i) = AL_f

AL_i	Y	AL_f
0	0	0
0	1	1
1	0	1
1	1	0

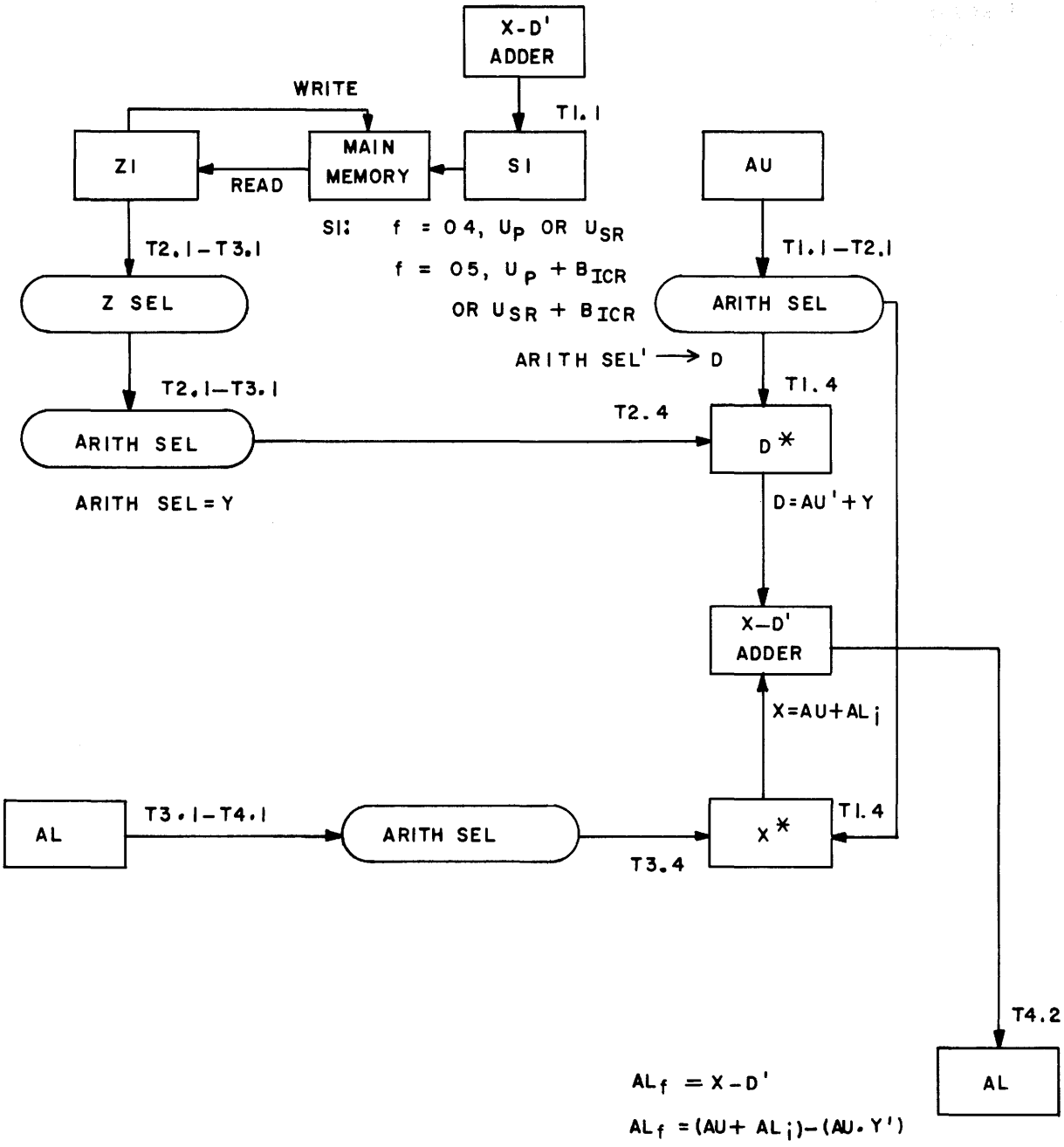
This same truth table also holds true for the AL_f value of $(AL_i' \cdot Y) + (AL_i \cdot Y')$. This term satifies the expression "complement AL where Y is a 1_2". If Y is a 0_2, the value in the corresponding AL_f bit position is the same as the initial AL value.

As discussed in a later sheet, the operand could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.15-3 for a sequential list of essential R1-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

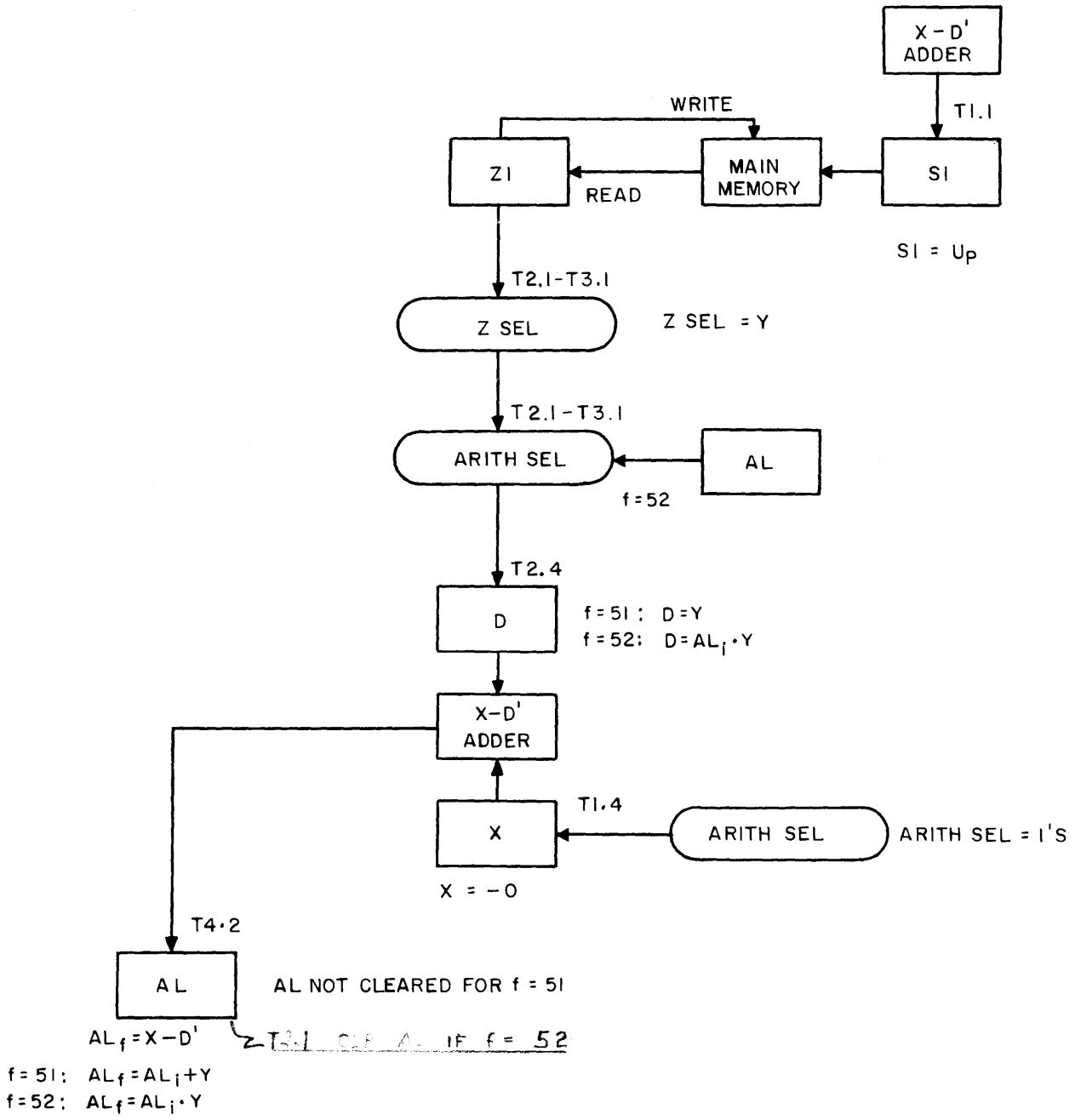
5.15-5. SUMMARY

The f = 04, 05 instructions use the value U_P or U_{SR}. The f = 51-53 instructions use only the value U_P. These values are formulated in D during the I-sequence. The R1-sequence is required to complete the executions of these instructions.



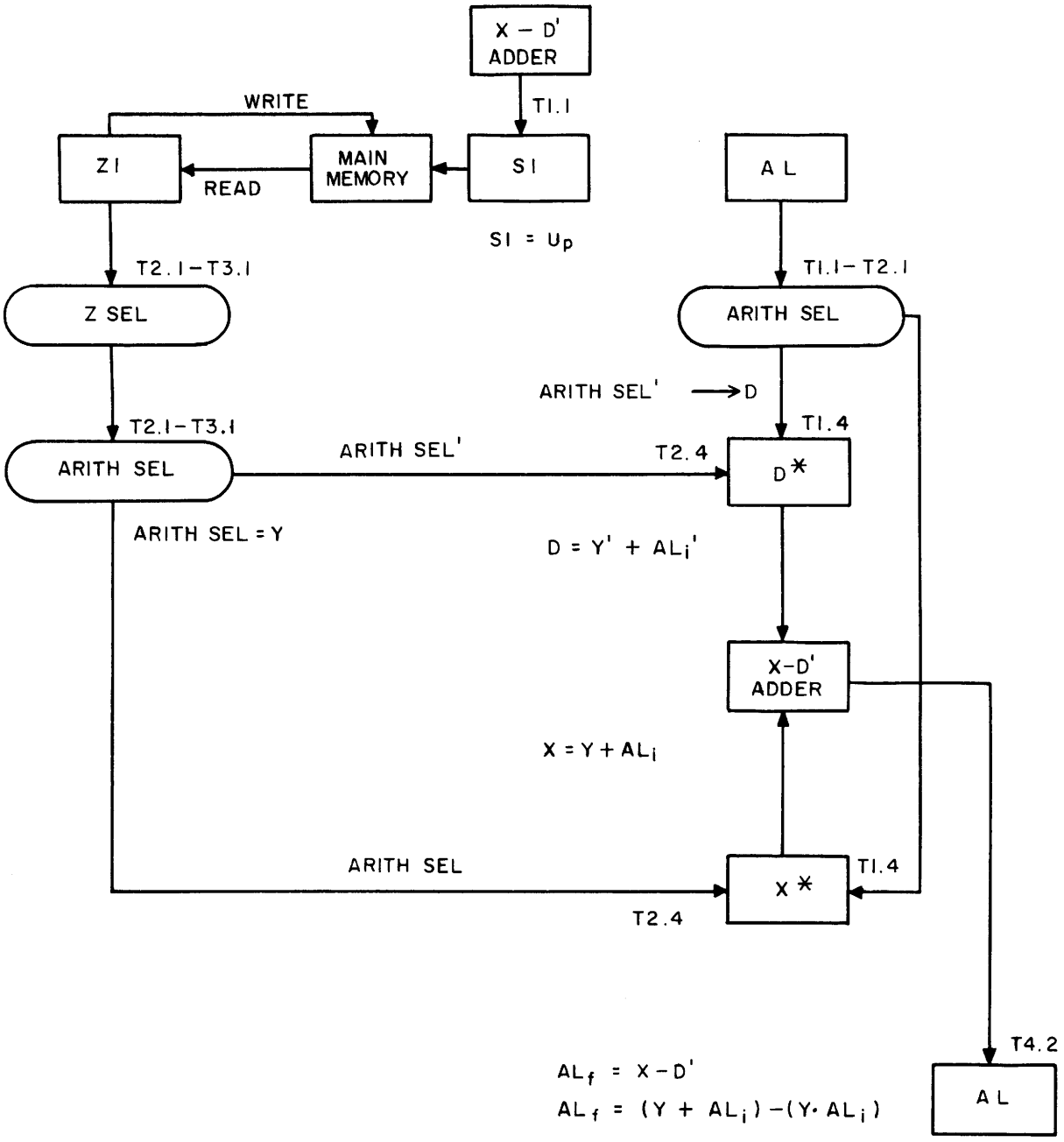
NOTES: * X AND D ARE NOT CLEARED AFTER T1.4 ENTRY
THE "+" SIGN INDICATES LOGICAL INCLUSIVE OR FUNCTION.

Figure 5.15-1. R1-Sequence Data Flow for f = 04, 05



NOTE: THE "+" SIGN INDICATES LOGICAL INCLUSIVE OR FUNCTION.

Figure 5.15-2. R1-Sequence Data Flow for f = 51, 52



NOTES: * X AND D ARE NOT CLEARED AFTER T1.4 ENTRY,
THE "+" SIGN INDICATES LOGICAL INCLUSIVE OR FUNCTION.

Figure 5.15-3. R1-Sequence Data Flow for f = 53

TABLE 5.15-3. R1 SEQUENCE ESSENTIAL COMMANDS FOR f = 04, 05, 51-53

TIME NOTATION	COMMANDS f =	04, 05	51	52	53
T4.4	Clear S1	X	X	X	X
T1.1	Adder → S1, init Memory	X	X	X	X
	AU → Arith Sel	X			
	AL → Arith Sel				X
T1.3	Clear Z1, clear X	X	X	X	X
	Clear D	X			X
T1.4	Arith Sel → X	X	X	X	X
	Arith Sel' → D	X			X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel	X	X	X	X
	AL → Arith Sel			X	
	Drop AU → Arith Sel	X			
	Drop AL → Arith Sel				X
T2.3	Clear D		X	X	
T2.4	Arith Sel → D	X	X	X	
	Arith Sel' → D, Arith Sel → X				X
T3.1	AL → Arith Sel	X			
	Drop Z1 → Z Sel, drop Z Sel → Arith Sel	X	X	X	X
	Drop AL → Arith Sel			X	
T3.4	Arith Sel → X	X			
T4.1	Clear AL	X		X	X
	Drop AL → Arith Sel	X			
T4.2	Adder → AL	X	X	X	X

SECTION 5 - CONTROL SECTION

5.16. INSTRUCTION EXECUTION OF CPAL, CPAU, CPA

5.16-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 50:61, 50:62, 50:63.

5.16-2. INTRODUCTION

These instructions complement the value in either AL, AU, or AL and AU together.

5.16-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.16-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) CPAL, f = 50:61. This instruction takes the 1's complement of the content of AL and places this value in AL. The initial content of AL is destroyed. If AL initially equals +0, AL is not changed.

b) CPAU, f = 50:62. This instruction takes the 1's complement of the content of AU and places this value in AU. The initial content of AU is destroyed. If AU initially equals +0, AU is not changed.

c) CPA, f = 50:63. This instruction takes the 1's complement of the combined contents of AU and AL and places this 36-bit value in AU and AL. The initial contents of AU and AL are destroyed. If the initial 36-bit value equals +0, AU and AL are not changed.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the complementing of AU occurs for f = 50:62, 50:63.

b) Next I-Sequence. During the first portion of the I-sequence for the next instruction, the complementing of AL occurs for f = 50:61, 50:63.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.16-1 for a block diagram description of the execution of f = 50:61, 50:62, 50:63.

Most of the I-sequence operations are as previously described. These instructions are of the format 2 group. The lower six bits of the instruction word are not used. If necessary refer to study guide sheet number 5.4 for a detailed description.

If f = 50:62, 50:63, D receives AU' at T3.4 time from arithmetic-select and applies this value to one side of the X-D' adder. According to the instruction, if the selected portion of A (AL, AU, or AL and AU) does not equal all 0's, X is set by both arithmetic-select and its complement. The resulting logical inclusive OR function of AU v AU' in X causes X to be set to all 1's.

If f = 50:62, 50:63, AU is cleared and receives X - AU'' which is effectively X + AU_i'. Unless AU_i equals +0, X is set to all 1's or -0 which does not alter the value of AU_i' as it is passed through the adder. If AU_i equals +0, X is left cleared to +0. AU then receives the adder (+0) - (+0)'' which is effectively (+0) - (+0). The result in AU would be +0, the same as its initial value.

The complementing of AL for f = 50:61, 50:63 is executed in the same manner as for AU. AU does not receive its final value from the adder until T1.2 of the I-sequence of the next instruction.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.16-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

An additional main timing flip-flop (T52) is used for f = 50:61, 50:63, to provide the timing for the clear-AL and adder $\longrightarrow$ AL commands. Refer to figure 9-11 in the logic diagrams for this flip-flop.

5.16-5. SUMMARY

The CPAL, CPAU, and CPA instructions are all format 2 and do not use the value k. Only the I-sequence and the first portion of the next I-sequence are required to complete the executions of these instructions.

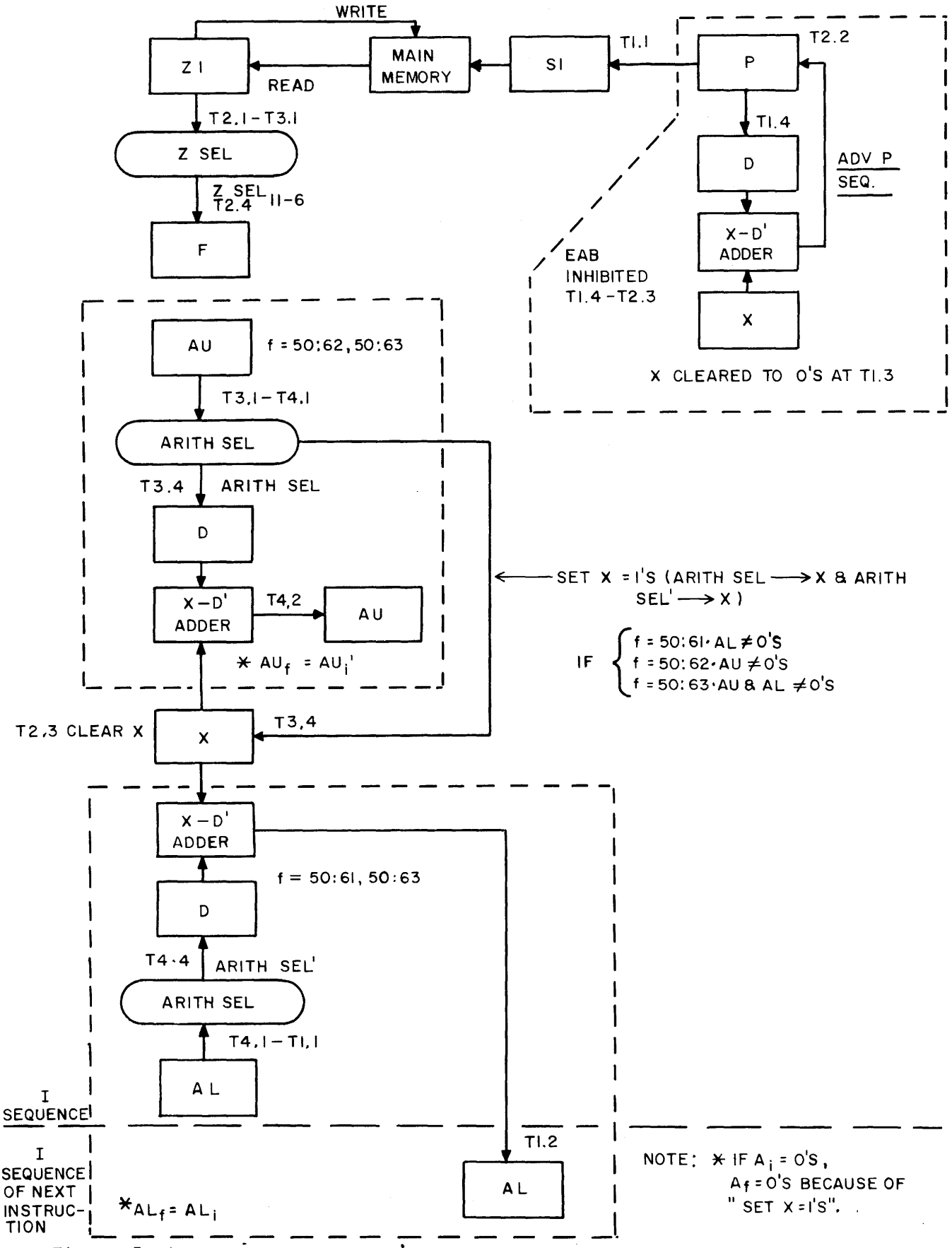


Figure 5.16-1. I and Next I Sequence Data Flow for f = 50:61, 50:62, 50:63

TABLE 5.16-1. I AND NEXT I SEQUENCE ESSENTIAL COMMANDS
FOR f = 50:61, 50:62, 50:63

TIME NOTATION	COMMANDS f =	50:61	50:62	50:63
	I SEQUENCE			
T4.4	Clear S1	X	X	X
T1.1	P → S1, Init Memory, *set Incr P ff	X	X	X
T1.3	Clear Z1, *set OXL11 ff	X	X	X
	*Clear D, *Clear X, Clear F	X	X	X
T1.4	$*P_L \rightarrow D_L$, $*P_U \rightarrow D_U$	X	X	X
	*Set Inhib EAB ff	X	X	X
T2.1	*Clear P, Z1 → Z Sel	X	X	X
	*Clear Incr P ff	X	X	X
T2.2	*Adder → P	X	X	X
T2.3	Clear X, *clear Inhib EAB ff	X	X	X
	*Clear OXL11 ff	X	X	X
T2.4	$Z\ Sel_{11-6} \rightarrow$ F, set 0XF06 ff	X	X	X
T3.1	Drop Z1 → Z Sel	X	X	X
	AU → Arith Sel		X	X
T3.3	Clear D		X	X
T3.4	Arith Sel' → D		X	X
	**Arith Sel → X & Arith Sel' → X if AL≠0's	X		
	**Arith Sel → X & Arith Sel' → X if AU≠0's		X	
	**Arith Sel → X & Arith Sel' → X if AU & AL≠0's			X
T4.1	Clear AU, drop AU → Arith Sel		X	X
	AL → Arith Sel	X		X
T4.2	Adder → AU		X	X
T4.3	set T52 ff (Main Timing), Clear D	X		X
T4.4	Arith Sel' → D	X		X
T1.1	Clear AL, drop AL → Arith Sel	X		X
T1.2	Adder → AL	X		X
T1.3	Clear T52 ff (Main Timing)	X		X

*These events are concerned with or are controlled by the advance-P subsequence.

**Arith Sel → X and Arith Sel' → X occurring simultaneously cause the function of set X = 1's.

NAME:_____

5.16-6. STUDY QUESTIONS.

a. Given: 10N01 constant low level output (logic diagrams, figure 9-17)
instruction = 506200
AU_i = 345670

After the execution of the given instruction, what is the content of AU?

AU_f = _____

b. Given: 91A00 constant low level output (logic diagrams, figure 9-29)
instruction = 506100

Describe any effect that this malfunction would have upon the execution of the given instruction and upon the final content of AL.

SECTION 5 - CONTROL SECTION

5.17. INSTRUCTION EXECUTION OF CMAL, CMALB, CMSK, CMSKB

5.17-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 02, 03, 06, 07.

5.17-2. INTRODUCTION

These instructions compare the content of memory with either AL or the logical product of AU·AL.

5.17-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.17-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) CMAL, f = 02. This instruction compares the operand Y with AL. The origin of Y is memory at address U_P if SR is inactive or U_{SR} if SR is active. The Compare flip-flop is set. The Equal flip-flop is set if AL = Y. The Greater flip-flop is set if AL $\geq$ Y algebraically. AL is considered greater if it is more positive than Y. AL is more positive than Y if AL is positive and Y is negative or if both AL and Y have the same signs and the absolute value of AL is greater than Y.

Neither AL nor Y is disturbed. The conditions of the Equal and Greater flip-flops can be later sensed by the execution of f = 60-67. One of these instructions must be the next sequential instruction after f = 02, 03, 06, 07 to obtain the results of the comparison. Any instruction other than f = 60-67 will clear the Compare, Equal, and Greater flip-flops.

b) CMALB, f = 03. Except for the memory address, this instruction is the same as f = 02. The address of Y is either U_P + B or U_{SR} + B depending upon the activeness of SR. The B register is specified by ICR.

c) CMSK, f = 06. This instruction is similar to f = 02. The origin of Y is the same. The logical product (AND function) of AU·AL is compared with the logical product of Y·AU.

The value in AU is such that it masks out the desired portion of AL. Where AU contains a 1_2, the corresponding bit position in AL will be used in the comparison with Y. If AU contains a 0_2, the corresponding logical product bit will be a 0_2 and that AL bit value will not be compared with Y.

AL, AU, and Y are not disturbed.

d) CMSKB, f = 07. Except for the memory address, this instruction is the same as f = 06. The address of Y is either $U_P + B$ or $U_{SR} + B$ depending upon the activeness of SR. The B register is specified by ICR.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of Y is formulated by U, P, SR, and B.

b) R1-Sequence. The R1-sequence is used to obtain the operand from memory, perform the comparison using Y, AL, and possibly AU, and record the results in the comparison flip-flops.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address of the operand.

2. Data Flow Block Diagram. Refer to figure 5.17-1 for a block diagram description of the execution of f = 02, 03, 06, 07.

The R1-sequence uses a memory reference to obtain the operand Y. X receives either AL or AU·AL from arithmetic-select at T1.4 time. D receives the complement of Y or Y·AU from arithmetic-select at T2.4 time. The X-D' adder is used to perform the comparison by subtracting either AL - Y or (AU·AL) - Y depending upon the instruction.

As discussed in a later sheet, the operand could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.17-1 for a sequential list of essential R1-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams. A detailed analysis of the compare logic is presented later in this sheet.

4. Compare Logic.

a) Compare and Equal Flip-flops. Refer to logic diagrams, figure 9-29 for flip-flops.

For f = 02, 03, 06, 07, the Compare flip-flop is set by gate 10E34 unconditionally at T3.2 time of the R1-sequence. At this same time gate 81A00 sets the Equal flip-flop if X= D'. 81A00 is conditioned by the AND function of the outputs of gates 80A00, 80A04, 80A09, and 80A13. These gates use the outputs of certain one of the X-D' adder gates to determine the equality of a portion of the X and D' values.

For example, the 10A00 input to 80A00 is at a low level if $X_{00} = D'_{00}$. Refer to logic diagrams, figure 9-94 for the analysis of the operation of gate 10A00.

10A00 has inputs from bit position 00 of the X and D registers; however, its output is described in terms of X and D'. A low level output of 10A00 indicates $X_{00} = D'_{00}$. Refer to table 5.17-2 for the conditions which produce a low level from 10A00.

TABLE 5.17-2. CONDITIONS FOR LOW LEVEL OUTPUT OF 10A00

CONDITION	INPUTS TO 10A00			
	00D00	00X00	01D00	01X00
$X_{00} = 0$ & $D'_{00} = 0$	H	L	L	H
$X_{00} = 1$ & $D'_{00} = 1$	L	H	H	L

Each of the other 10A--gates of the Adder is used to determine equality in its particular bit position of X and D'.

b) Greater Flip-Flop. Refer to logic diagrams, figure 9-29 for this flip-flop.

When the Compare flip-flop is set, the Greater flip-flop is also set if X is more positive than D' as indicated by a low level output of gate 70A17. 70A17 compares X and D' by sensing the signs of these values from the adder bit position 17 and the end-around borrow condition involved in the X-D' subtraction.

Input 10A17 to 70A17 is like 10A00 which was discussed previously. 10A17 outputs a high level if $X_{17} \neq D'_{17}$. The condition for this high level can best be expressed as X and D' signs unlike. The 11A17 input to 70A17 is inversion of 10A17. Therefore, the condition for a high level at this input can be expressed as X and D' signs like. Refer to table 5.17-3 for the conditions which produce a low level from 70A17.

TABLE 5.17-3. CONDITIONS FOR LOW LEVEL OUTPUT OF 70A17 (SET GREATER FF)

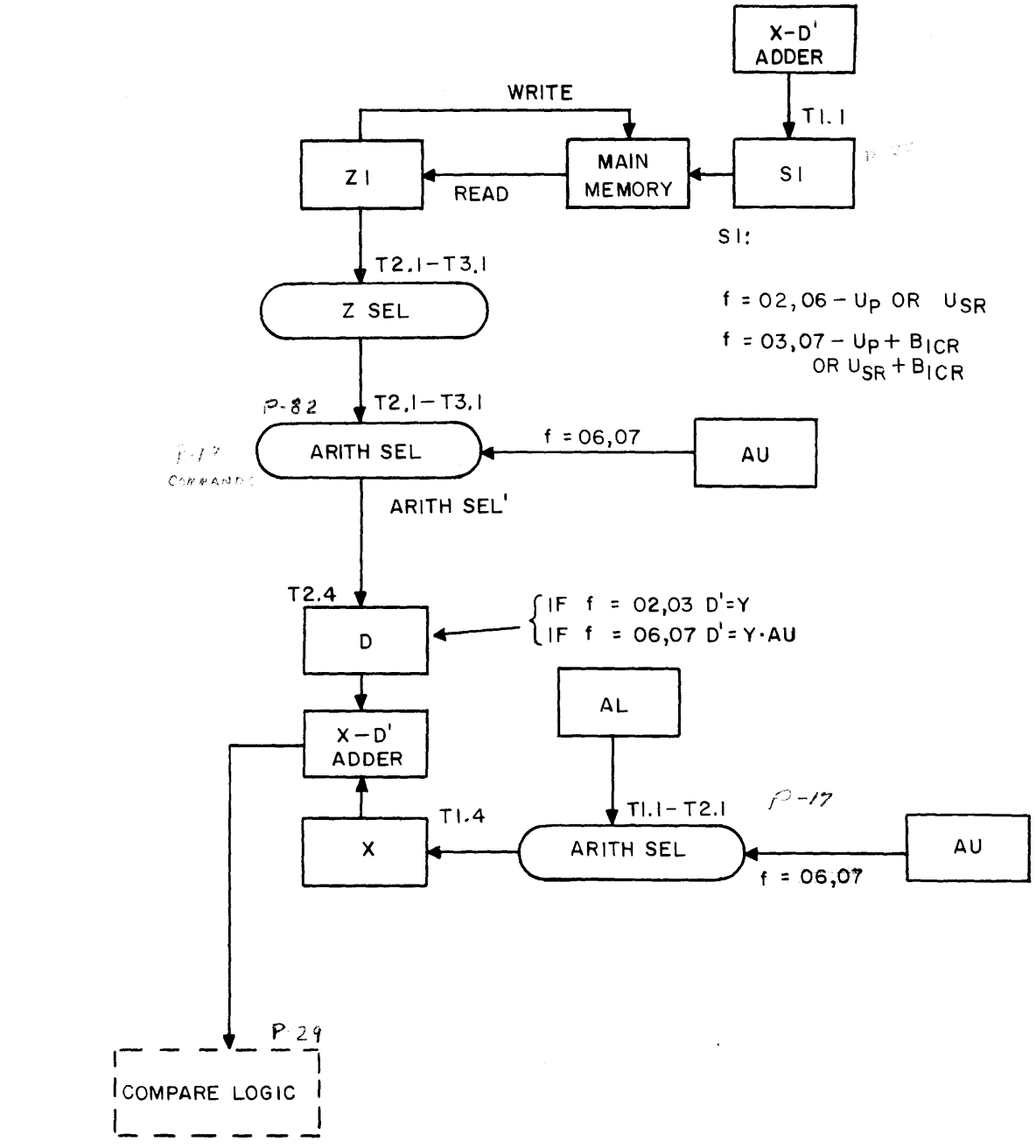
CONDITION	INPUTS TO 70A17			
	12A00	11A17	13A00	10A17
unlike signs · EAB (X pos & D neg)	H	L	L	H
like signs · No EAB (/X/≥/D'/)	L	H	H	L

c) Clearing of Comparison Flip-Flops. Refer to logic diagrams, figure 9-29.

The Greater, Compare, and Equal flip-flops are cleared by a high level from gate 00E34. 00E34 clears these flip-flops at T4.1 time of the I-sequence for any instruction other than f = 60-67. The f = 60-67 instructions test the states of these flip-flops. Therefore, these instructions must immediately follow the f = 02, 03, 06, 07 instructions in order to detect the result of the comparison.

5.17-5. SUMMARY

The f = 02, 03, 06, 07 instructions use the value Up or U_{SR} which is formulated in D during the I-sequence. The R1-sequence is required to complete the execution of these instructions.



T3.2 SET COMPARE ff
SET EQUAL ff IF X = D' *
SET GREATER ff IF X ≥ D' ALGEBRAICALLY **

NOTES: * X = D' MEANS: f = 02,03 (AL=Y)
f = 06,07 (AL·AU)=(Y·AU)
** X ≥ D' MEANS: f = 02,03 (AL ≥ Y)
f = 03,07 (AL·AU) ≥ (Y·AU)

Figure 5.17-1. R1 Sequence Data Flow for f = 02, 03, 06, 07

TABLE 5.17-1. R1 SEQUENCE ESSENTIAL COMMANDS FOR f = 02, 03, 06, 07

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	Adder → S1, Init Memory, AL → Arith Sel
	AU → Arith Sel if f = 06, 07
T1.3	Clear Z1, clear X
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel, drop AL Arith Sel, drop AU → Arith Sel, AU → Arith Sel if f = 06, 07
T2.3	Clear D
T2.4	Arith Sel' → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel, drop AU → Arith Sel
T3.2	Set Compare ff, set Equal ff if X = D'
	Set Greater ff if X ≥ D' algebraically

SECTION 5 - CONTROL SECTION

5.18. INSTRUCTION EXECUTION OF RND

5.18-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of the instruction with f = 50:60.

5.18-2. INTRODUCTION

This instruction rounds the 36-bit number in AU and AL to formulate an 18-bit number.

5.18-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.18-4. INFORMATION

a. General Description.

1. Instruction Interpretation. This instruction, RND, considers the combined contents of AU and AL as a 36-bit number. AU contains the more significant half. This value is rounded so as to formulate an 18-bit value which is placed in AL. AU is not disturbed. The original content of AL is destroyed.

2. Execution Sequence (I). All operations are performed within the I-sequence. Only the one memory reference to obtain the instruction is necessary.

b. Detailed Analysis.

1. Effect of AL_{17}. The round operation involves the modification of AU by AL_{17}. The bit value in AL_{17} is simply added to AU. AU, therefore, is made greater by 1_2 or left unchanged. The operation is performed in the X-D' adder and the result is placed in AL.

The sign of the 36-bit AU and AL number must be considered. With AU positive, 0_2 is added to AU if $AL_{17} = 0$ and 1_2 is added to AU if $AL_{17} = 1$. The addition of AU + 1 is accomplished by placing AU in D, setting X to +0, and inhibiting the the end around borrow. Refer to table 5.18-1 for the adder conditions which are set up when AU is positive.

Since X contains all 0's for $AL_{17} = 1$, the subtraction of X-D' by the adder causes an end-around borrow for any positive value of AU. The addition of +1 is effected simply by inhibiting this borrow.

TABLE 5.18-1. X-D' ADDER CONDITIONS FOR f = 50:60

AU	AL_{17}	D	X	Other	Adder Output	AL_f
pos	0	AU	-0	none	(-0) - (AU)'	AU + 0
pos	1	AU	+0	EAB inhibited	(+) - (AU)' +1	AU + 1
neg	0	AU	-0	EAB inhibited*, EAB inserted	(-0) - (AU)' -1	AU - 1
neg	1	AU	+0	none	(+0) - (AU)'	AU - 0

* The inhibiting of the EAB with these conditions is of no use, since no EAB will occur.

If AU is negative and $AL_{17} = 0_2$, this 0_2 is actually the negative representation of 1_2 and AU must be made more negative. AU is made more negative by subtracting 1. Refer to table 5.18-1 for the adder conditions which are set up when AU is negative.

Since X contains all 1's for $AL_{17} = 0$, the subtraction of X-D' by the adder causes no end-around borrow. The subtraction of -1 is effected simply by inserting an end-around borrow.

2. Data Flow Block Diagram. Refer to figure 5.18-1 for a block diagram description of the execution of f = 50:60.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

D receives AU at T3.4 time. X is cleared to +0 and is set to -0 if $AL_{17} = 0$. Setting X to all 1's (-0) is effected by entering X with both arithmetic-select and its complement.

The addition and subtraction of 1_2 is effected by the manipulation of the adder end-around borrow.

As discussed in a later sheet, the instruction could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.18-2 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.18-5. SUMMARY

The RND instruction is format 2 and does not use the value k. Only the I-sequence is required to complete the execution of this instruction.

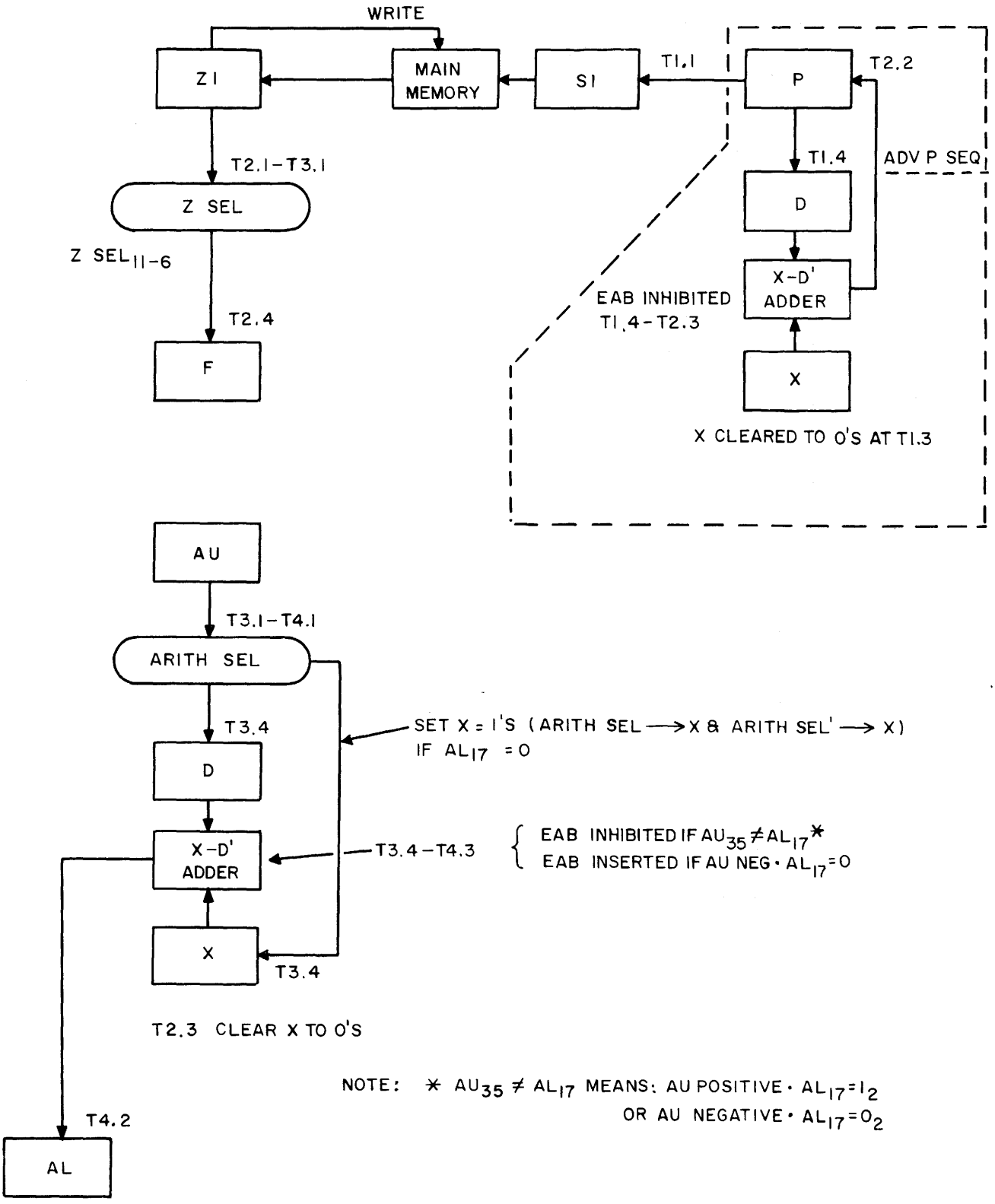


Figure 5.18-1. I-Sequence Data Flow for f = 50:60

TABLE 5.18-2. I-SEQUENCE ESSENTIAL COMMANDS FOR f = 50:60

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P ⟶ S1, Inite Memory, *set Incr P ff
T1.3	*Clear D, *Clear X, Clear Z1, Clear F, *set OXL11 ff
T1.4	*P_L ⟶ D_L, *P_U ⟶ D_U, *set Inhib EAB ff
T2.1	*Clear P, Z1 ⟶ Z Sel, *clear Incr P ff
T2.2	*Adder ⟶ P
T2.3	Clear X, *clear OXL11 ff, *clear Inhib EAB ff
T2.4	Z Sel_{11-6} ⟶ F, set OXF06 ff
T3.1	AU ⟶ Arith Sel, drop Z1 ⟶ Z Sel
T3.3	Clear D
T3.4	Arith Sel ⟶ D, Arith Sel ⟶ X & Arith Sel' ⟶ X (Set X = 1's) if AL_{17} = 0 Set Inhib EAB ff if $AU_{35} \neq AL_{17}$, set Insert EAB ff if AU neg·AL_{17}=0
T4.1	Clear AL, drop AU ⟶ Arith Sel
T4.2	Adder ⟶ AL
T4.3	Clear Inhib EAB ff, clear Insert EAB ff

* These events are concerned with or are controlled by the advance-P subsequence.

SECTION 5 - CONTROL SECTION

5.19. INSTRUCTION EXECUTION OF JP, JPB, JPAUZ, JPALZ, JPAUNZ, JPALNZ, JPAUP, JPALP, JPAUNG, JPALNG

5.19-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 34, 35, 60-67.

5.19-2. INTRODUCTION

These instructions either unconditionally or conditionally perform a program jump to the address U_P or $U_P + B$.

5.19-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.19-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) JP, f = 34. This instruction unconditionally performs a program jump to the address U_P.

b) JPB, f = 35. This instruction unconditionally performs a program jump to the address $U_P + B$. The B register is specified by ICR.

c) JPAUZ, f = 60. This instruction performs a program jump to the address U_P if the Compare flip-flop is clear and AU = +0 or the Compare flip-flop is set and the Equal flip-flop is set.

d) JPALZ, f = 61. This instruction performs a program jump to the address U_P if the Compare flip-flop is clear and AL = +0 or the Compare flip-flop is set and the Equal flip-flop is set.

e) JPAUNZ, f = 62. This instruction performs a program jump to the address U_P if the Compare flip-flop is clear and AU $\neq$ +0 or the Compare flip-flop is set and the Equal flip-flop is clear.

f) JPALNZ, f = 63. This instruction performs a program jump to the address U_P if the Compare flip-flop is clear and AL $\neq$ 0 or the Compare flip-flop is set and the Equal flip-flop is clear.

g) JPAUP, f = 64. This instruction performs a program jump to the address Up if the Compare flip-flop is clear and AU is positive or the Compare flip-flop is set and the Greater flip-flop is set.

h) JPALP, f = 65. This instruction performs a program jump to the address Up if the Compare flip-flop is clear and AL is positive or the Compare flip-flop is set and the Greater flip-flop is set.

i) JPAUNG, f = 66. This instruction performs a program jump to the address Up if the Compare flip-flop is clear and AU is negative or the Compare flip-flop is set and the Greater flip-flop is clear.

j) JPALNG, f = 67. This instruction performs a program jump to the address Up if the Compare flip-flop is clear and AL is negative or the Compare flip-flop is set and the Greater flip-flop is clear.

2. Execution Sequence (I). All operations are performed within the I-sequence. Only the one memory reference to obtain the instruction is necessary.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.19-1 for a block diagram description of the execution of f = 34, 35, 60-67.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

The value of Up is formulated in D from where it is applied to one side of the X-D' adder. If f = 35, the content of the B register specified by ICR is placed in X from arithmetic-select. Otherwise, X receives all 1's from arithmetic-select. The adder output is either Up or Up + B depending upon the instruction.

If the jump condition is satisfied, P is cleared and is set to the jump address from the adder. Instructions with f = 34, 35, perform unconditional jumps, The program jump is actually effected by the following I-sequence when it obtains the next instruction from the address contained in P.

As discussed in a later sheet, the instruction word could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.19-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams. A detailed analysis of the jump evaluation logic is presented later in this sheet.

3. Jump Evaluation Logic.

a) Clear P and Adder $\longrightarrow$P Command Enable. Refer to logic diagrams, Figure 9-21 for the enable logic for the commands clear P and adder $\longrightarrow$P.

For these instructions, gate 10N07 is used to enable the clear P and adder $\longrightarrow$P commands. For f = 34, 35, 10N07 unconditionally outputs a high level when the T42 flip-flop is set during the I-sequence which causes these commands. Input 13G34 is at a constant low level for f = 34, 35.

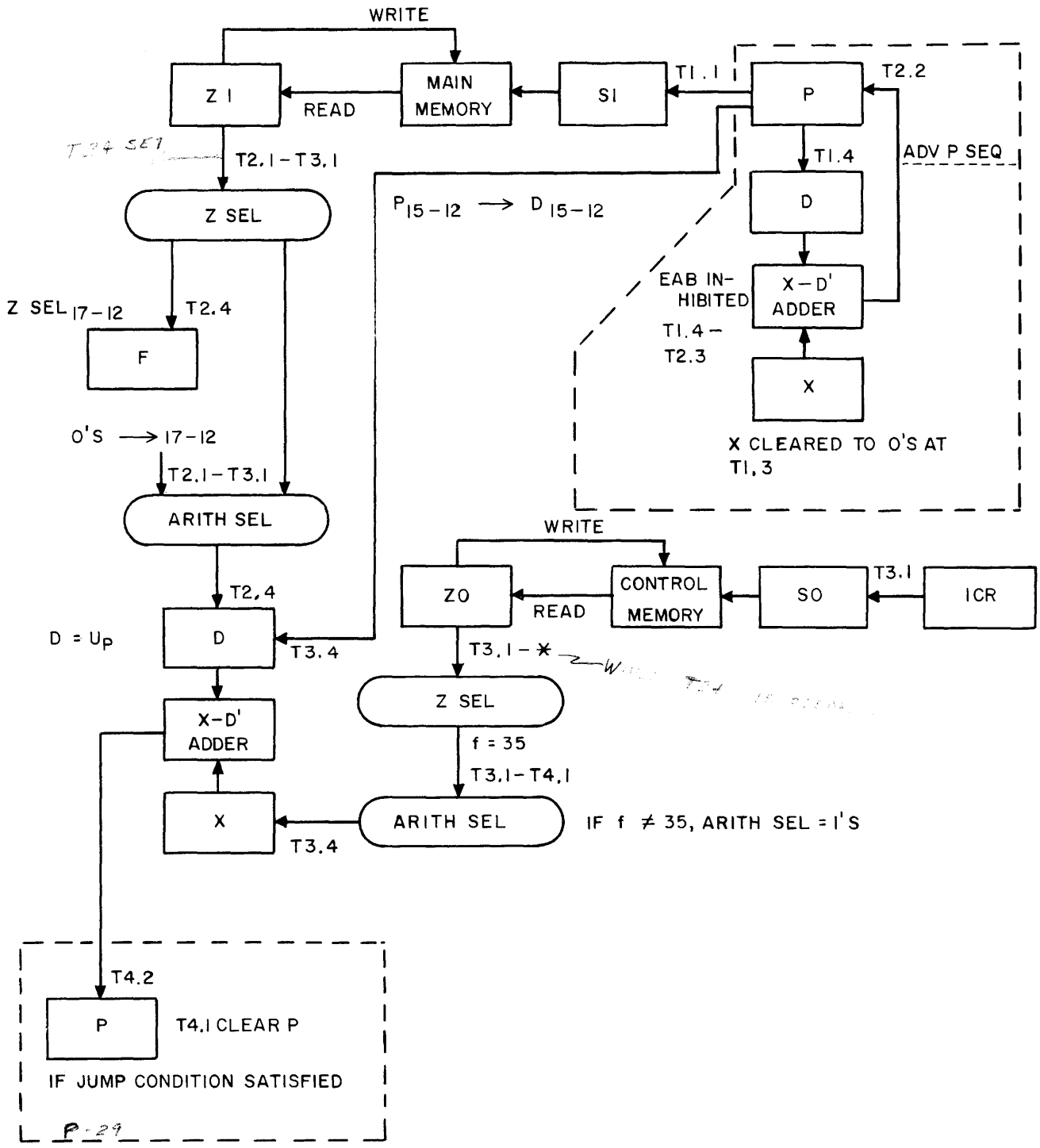


Figure 5.19-1. I Sequence Data Flow for f = 34, 35, 60-67

TABLE 5.19-1. I SEQUENCE ESSENTIAL COMMANDS FOR f = 34, 35, 60-67

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P → S1, Init Memory, *set Incr P ff
T1.3	*Clear D, *clear X, clear Z1, clear F, *set OXL11 ff
T1.4	*P_L → D_L, *P_U → D_U, *set Inhib EAB ff
T2.1	*Clear P, Z1 → Z Sel, Z Sel → Arith Sel, 0's → Arith Sel_{17-12}
	*Clear Incr P ff
T2.2	*Adder → P
T2.3	Clear X, clear D, *clear OXL11 ff, *clear Inhib EAB ff
T2.4	Z Sel_{17-12} → F, Arith Sel → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel, drop 0's → Arith Sel_{17-12}
	INCR → S0, Init CM, Z0 → Z Sel**, Z Sel → Arith Sel if f = 35
T3.4	P_{15-12} → D_{15-12}, Arith Sel → X
T4.1	***Clear P if jump satisfied, drop Z Sel → Arith Sel
T4.2	***Adder → P if jump satisfied

NOTES: * These events are concerned with or are controlled by the advance-P subsequence.
** Z0 → Z Sel occurs for duration of T24 ff clear.
*** Jump condition is satisfied if:
f = 34, 35 unconditionally
f = 60 Compare ff clear · AU = +0 or Compare ff set · Equal ff set
f = 61 Compare ff clear · AL = +0 or Compare ff set · Equal ff set
f = 62 Compare ff clear · AU ≠ +0 or Compare ff set · Equal ff clear
f = 63 Compare ff clear · AL ≠ +0 or Compare ff set · Equal ff clear
f = 64 Compare ff clear · AU pos or Compare ff set · Greater ff set
f = 65 Compare ff clear · AL pos or Compare ff set · Greater ff set
f = 66 Compare ff clear · AU neg or Compare ff set · Greater ff clear
f = 67 Compare ff clear · AL neg or Compare ff set · Greater ff clear

If f = 60-67, the logic level of input 13G34 is conditioned by the jump evaluation logic. Refer to logic diagrams, figure 9-29 for the jump evaluation logic.

For f = 60-67, gate 13G34 has a high input level on pin 6. Each of the two remaining input OR gates to 13G34 must be satisfied with at least one high level input in order to produce a low level output of 13G34 and cause the program jump.

b) Compare Flip-Flop Clear. If the Compare flip-flop is clear, it applies a high level input to pin 13 of 13G34. In this case, gate 24G34 must apply a high level to pin 8 of 13G34 in order to cause a program jump. The output of 24G34 is conditioned by the values in AU and AL which are to be tested by the f = 60-67 instructions when the Compare flip-flop is clear. Refer to table 5.19-2 for the conditions necessary to produce a high level output of 24G34.

c) Compare Flip-Flop Set. If the Compare flip-flop is set, it applies a high level input to pin 9 of 13G34. In this case, gate 12G34 must apply a high level to pin 12 of 13G34 in order to cause a program jump. The output of 12G34 is conditioned by the states of the Greater and Equal flip-flops which are to be tested by the f = 60-67 instructions when the Compare flip-flop is set. Refer to table 5.19-3 for the conditions necessary to produce a high level output of 12G34.

TABLE 5.19-2. CONDITIONS TO PRODUCE HIGH LEVEL FROM 24G34

	GATE OUTPUT LEVELS					
Conditions	91A18	91A00	20G34	21G34	22G34	23G34
f = 60 · AU = +0	L	L or H	H	H	L	H
f = 61 · AL = +0	L or H	L	H	H	L	H
f = 62 · AU ≠ +0	H	L or H	L	L or H	H	L
f = 63 · AL ≠ +0	L or H	H	L	L or H	H	L
f = 64 · AU pos	L or H	L or H	L or H	H	L	H
f = 65 · AL pos	L or H	L or H	L or H	H	L	H
f = 66 · AU neg	H	L or H	L	L	H	L
f = 67 · AL neg	L or H	H	L	L	H	L

TABLE 5.19-3. CONDITIONS TO PRODUCE HIGH LEVEL FROM 12G34

CONDITIONS	GATE OUTPUT LEVELS	
	10G34	11G34
f = 60, 61 · Equal ff set	H	L
f = 62, 63 · Equal ff clear	L	H
f = 64, 65 · Greater ff set	H	L
f = 66, 67 · Greater ff clear	L	H

5.19-5. SUMMARY

The f = 34, 35, 60-67 instructions use the value Up which is formulated in D during the I-sequence. Only the I-sequence is required to complete the executions of these instructions.

NAME: _____

5.19-6. STUDY QUESTIONS

a. Given: instruction = 611000
AL = 340000

When executed, the given instruction performs a program jump. Assume each of the following malfunctions to occur individually. Indicate for each of these conditions if it would cause the erroneous program jump. Refer to logic diagrams, figure 9-29.

would cause jump?

1. 91A00 constant low level output _____
2. 22G34 constant low level output _____
3. 13G34 constant low level output _____
4. 10G34 constant low level output _____
5. 10G34 grounded output _____

SECTION 5 - CONTROL SECTION

5.20 INSTRUCTION EXECUTION OF IJPEI, IJP

5.20-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 54, 55.

5.20-2. INTRODUCTION

These instructions perform an unconditional program jump to the address which is contained in memory. Interrupts are enabled by f = 54 clearing the All Interrupt Lockout flip-flop.

5.20-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.20-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) IJPEI, f = 54. This instruction performs an unconditional program jump. The jump address is contained in the 15 least significant bit positions of memory at the address U_P. The All Interrupt Lockout flip-flop is cleared to enable the honoring of a future interrupt.

This instruction is usually executed at the end of subroutine which is initiated at the occurrence of an interrupt. The program jump would be executed to return to the main program and allow the detection of another interrupt.

b) IJP, f = 55. Except for the interrupt enable feature, this instruction is the same as f = 54. The All Interrupt Lockout flip-flop is not affected.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of the jump address is formulated from U and P. The All Interrupt Lockout flip-flop is cleared if f = 54.

b) R1-Sequence. The R1-sequence is used to obtain the jump address from memory and place it in P.

b. Detailed Analysis.

1. I-Sequence. Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address (Up) of the jump address.

In addition to the normal I-sequence operations, for f = 54, the All Interrupt Lockout flip-flop is cleared at T3.4 time. The effect of the All Interrupt Lockout flip-flop upon interrupts is analyzed in a later sheet.

2. Data Flow Block Diagram. Refer to figure 5.20-1 for a block diagram description of the execution of f = 54, 55.

The R1-sequence uses a memory reference to obtain the jump address. This address is applied to one side of the X-D' adder from D. X applies -0 to the other side of the adder which it received from arithmetic-select. Arithmetic-select indicates all 1's when nothing is applied to it.

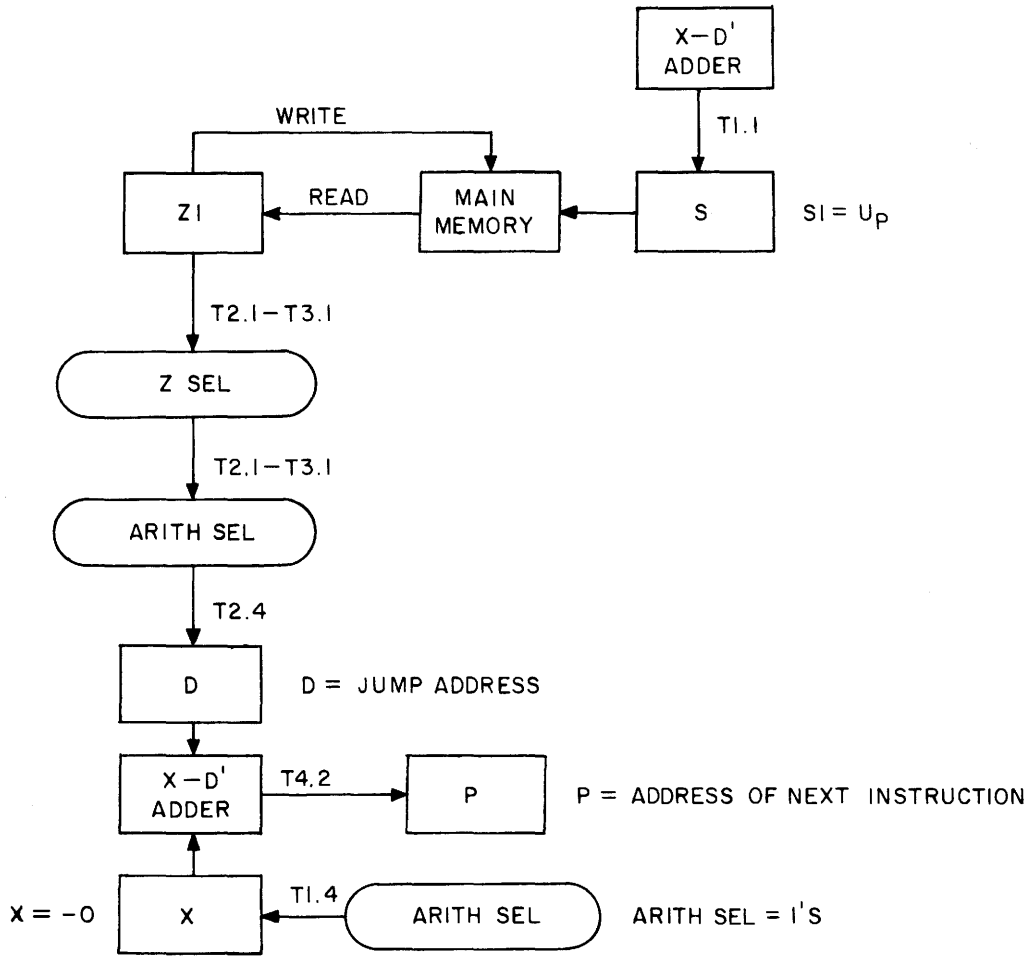
The adder output of X-D' is placed in P. P actually receives (-0) - (jump address)' which is effectively the jump address not modified. The following I-sequence actually effects the jump when it obtains the next instruction from the address contained in P.

As discussed in a later sheet, the jump address could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.20-1 for a sequential list of essential R1-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.20-5. SUMMARY

The IJPEI and IJP instructions use the value U_P which is formulated in D during the I-sequence. The R1-sequence is required to complete the execution of these instructions.



NOTE: FOR f = 54, CLEAR ALL INTERRUPT LOCKOUT FF AT T3.4 OF 1 SEQUENCE.

Figure 5.20-1. R1-Sequence Data Flow for f = 54, 55

TABLE 5.20-1. R1-SEQUENCE ESSENTIAL COMMANDS FOR f = 54, 55

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	Adder → S1, Init Memory
T1.3	Clear Z1, clear X
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
T4.1	Clear P
T4.2	Adder → P

SECTION 5 - CONTROL SECTION

5.21. INSTRUCTION EXECUTION OF RJP

5.21-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of the instruction with f = 76.

5.21-2. INTRODUCTION

This instruction stores P in memory and performs a direct program jump.

5.21-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-14.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.21-4. INFORMATION

a. General Description.

1. Instruction Interpretation. This instruction, RJP, stores the content of P in memory at the address U_P and performs a program jump to the address $U_P + 1$. Just prior to the execution of this instruction, P contains the address of the normally next sequential instruction in the program. The storage of this address allows a later return to the proper point in the main program.

If this instruction is executed from an interrupt entrance address due to an interrupt, the storage address for P is U, and the jump address is U +1.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the storage address for P is formulated from U and P.

b) W-Sequence. The W-sequence uses a memory reference to store the content of P and set P to the jump address.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed

description. At the end of the I-sequence, the X-D' adder is outputting the address U_P.

2. Data Flow Block Diagram. Refer to figure 5.21-1 for a block diagram description of the execution of f = 76.

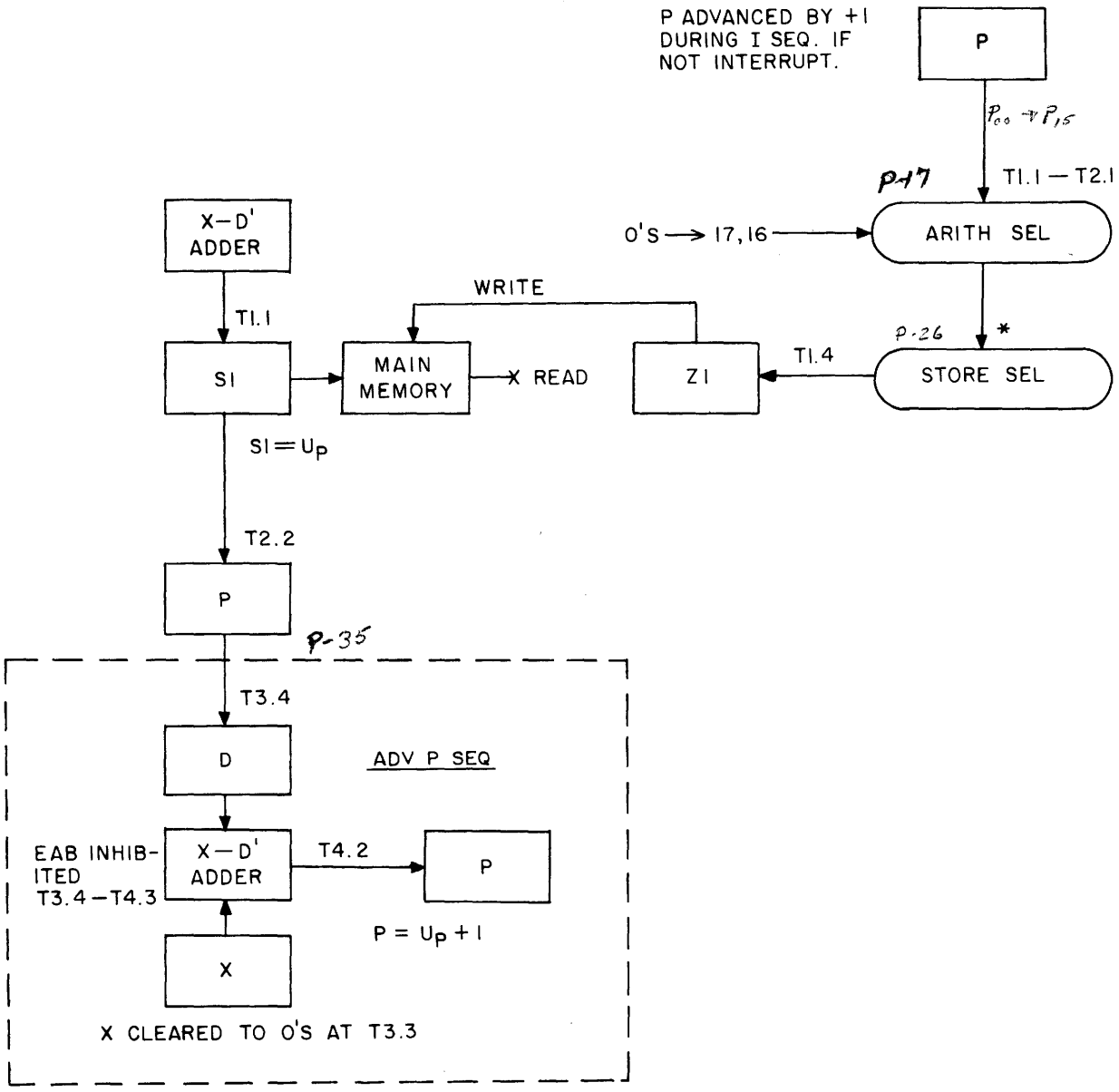
The W-sequence uses a memory reference to store P. Z1 is set to P from where it is stored by the write portion of the memory cycle. The gating of memory to Z1 is disabled during the read portion of the memory cycle which destroys the original memory content.

P is cleared and, at T2.2 time, receives the value U_P from S1. The advance-P subsequence is used to increment U_P by +1. The following I-sequence actually effects the jump by obtaining the next instruction from the address contained in P.

3. Essential Commands. Refer to table 5.21-1 for a sequential list of essential W-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.21-5. SUMMARY

The RJP instruction uses the value U_P which is formulated in D during the I-sequence. The W-sequence is required to complete the execution of this instruction.



NOTE: *ARITH SEL ⟶ STORE SEL OCCURS DURING ENTIRE W SEQUENCE.

Figure 5.21-1. W Sequence Data Flow for f = 76

TABLE 5.21-1. W SEQUENCE ESSENTIAL COMMANDS FOR f = 76

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	Adder → S1, Init memory, P → Arith Sel, 0's → Arith Sel$_{17, 16}$
T1.3	Clear Z1
T1.4	* Store Sel → Z1, disable Mem → Z1
T2.1	Clear P, drop P → Arith Sel, drop 0's → Arith Sel$_{17, 16}$
T2.2	S1 → P
T2.4	Drop disable Mem → Z1
T3.1	**set Incr P ff
T3.3	**Clear D, **clear X, **set OXL11 ff
T3.4	**P_L → D_L, **P_U → D_U, *set Inhib EAB ff
T4.1	**Clear P, **clear Incr P ff
T4.2	** Adder → P
T4.3	**Clear OXL11 ff, **clear Inhib EAB ff

* Arith Sel → Store Sel occurs during entire W-sequence.

** These events are concerned with or are controlled by the advance-P subsequence.

SECTION 5 - CONTROL SECTION

5.22. INSTRUCTION EXECUTION OF IRJP, IRJPB

5.22-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 30, 31.

5.22-2. INTRODUCTION

These instructions store P in memory and perform an indirect program jump.

5.22-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11, 4-12, and 4-14.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.22-4. INFORMATION

a. General Description

1. Instruction Interpretation.

a) IRJP, f = 30. This instruction stores the content of P in memory. The storage address is obtained from memory at the address U_P. A program jump is then performed to this storage address incremented by +1. Just prior to the execution of this instruction, P contains the address of the normally next sequential instruction in the program. The storage of this address allows a later return to the proper point in the main program.

If this instruction is executed from an interrupt entrance address due to an interrupt, the address of the P storage is U, rather than U_P.

b) IRJPB, f - 31. Except for the memory address of the storage address for P, this instruction is the same as f = 30. The storage address for P is obtained from memory at the address $U_P + B$. The storage address is incremented by +1 to produce the jump address.

2. Execution Sequences.

a) I Sequence. During the I-sequence which obtains the instruction from memory, the address of the P storage address is formulated from U and P.

b) R1-Sequence. The R1-sequence uses a memory reference to obtain the storage address for P.

c) W-Sequence. The W-sequence uses a memory reference to store the content of P and set P to the jump address.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address U_P.

2. Data Flow Block Diagram. Refer to figure 5.22-1 for a block diagram description of the execution of f = 30, 31.

The R1-sequence obtains from memory the storage address for P. D applies this address to one side of the X-D' adder. X receives -0 from arithmetic-select. The adder outputs (-0) - (P storage address)' which is effectively the P storage address not modified.

The W-sequence uses the storage address from the Adder to store P in memory. The gating of memory to Z1 is disabled during the read portion of the memory cycle which destroys the original memory content.

P is cleared and, at T2.2 time, receives the storage address from S1. The advance-P subsequence is used to increment this storage address by +1. The following I-sequence actually effects the jump by obtaining the next instruction from the address contained in P.

As discussed in a later sheet, the P storage address could be obtained from bootstrap or control memory. The value of the storage address could also specify bootstrap or control memory. Since bootstrap has nondestructive readout, storage into this memory is useless.

3. Essential Commands. Refer to table 5.22-1 for a sequential list of essential R1 and W-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.22-5. SUMMARY

The IRJP and IRJPB instructions use the value U_P which is formualted in D during the I-sequence. The R1 and W-sequences are required to complete the executions of these instructions.

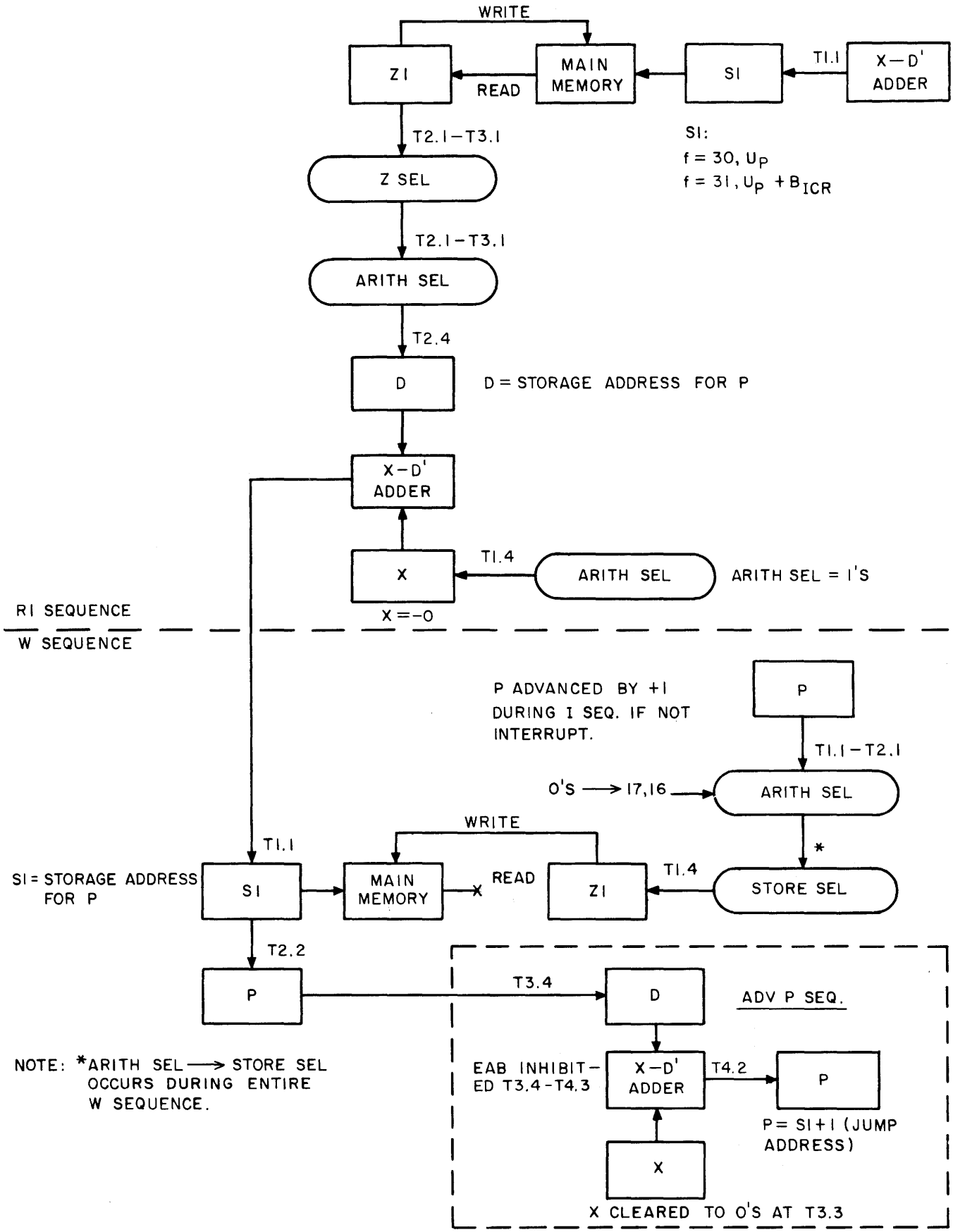


Figure 5.22-1. R1 and W-Sequence Data Flow for f = 30, 31

TABLE 5.22-1. R1 AND W-SEQUENCE ESSENTIAL COMMANDS FOR f = 30, 31

TIME NOTATION	COMMANDS
	R1 SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init Memory
T1.3	Clear Z1, clear X
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
	W SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init Memory, P → Arith Sel, 0's → Arith $Sel_{17,16}$
T1.3	Clear Z1
T1.4	*Store Sel → Z, disable Mem → Z1
T2.1	Clear P, drop P → Arith Sel, drop 0's → Arith $Sel_{17,16}$
T2.2	S1 → P
T2.4	Drop disable Mem → Z1
T3.1	**Set Incr P ff
T3.3	**Clear D, **Clear X, **set OXL11 ff
T3.4	**P_L → D_L, **P_U → D_U, **set Inhib EAB ff
T4.1	**Clear P, **clear Incr P ff
T4.2	**Adder → P
T4.3	**Clear OXL11 ff, **clear Inhib EAB ff

* Arith Sel → Store Sel occurs during entire W Sequence.

** These events are concerned with or are controlled by the Advance P Subsequence.

SECTION 5 - CONTROL SECTION

5.23. INSTRUCTION EXECUTION OF SKP, SKPNBO, SKPOV, SKPNOV

5.23-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 50:50 - 50:53.

5.23-2. INTRODUCTION

These instructions either unconditionally or conditionally perform a program jump of the next sequential instruction.

5.23-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.23-4. INFORMATION

a. General Description.

1. Instruction Interpretation.

a) SKP, f = 50:50. This instruction performs a program skip if either bit position 5 of the instruction word equals 1 or if any bit position from 4 through 0 of the instruction word which equals a 1 corresponds to a skip key which is manually selected.

b) SKPNBO, f = 50:51. This instruction performs a program skip if the Borrow Test flip-flop is clear. The Borrow Test flip-flop is not affected. This flip-flop is set by the execution of f = 20-23 to indicate that an end-around borrow condition occurred during the arithmetic operation involving AU. The f = 20-23 instructions prevent the end around borrow and if this condition is detected by f = 50:51, a f = 22, 23 instruction should be executed with Y = +1 so as to provide the effect of the necessary end-around borrow.

c) SKPOV, f = 50:52. This instruction performs a program skip if the Overflow flip-flop is set. After its condition is sensed, the Overflow flip-flop is cleared.

d) SKPNOV, f = 50:53. This instruction peforms a program skip if the Overflow flip-flop is cleared. After its condition is sensed, the Overflow flip-flop is cleared.

2. Execution Sequence (I). All operations are performed with the I-sequence. Only the one memory reference to obtain the instruction is necessary.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.23-1 for a block diagram description of the execution of f = 50:50 - 50:53.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed analysis.

D is cleared and, at T3.4 time, receives the content of P. At this time P has been incremented by +1. Therefore, D contains the address of the current instruction plus 1 which is the address of the instruction which might be skipped.

X is cleared to 0's at T3.3 time. The X-D' adder is used to increment the content of D by +1 just as is done by the advance-P subsequence. If the skip condition is satisfied, P is cleared and receives the adder value of $P_i + 2$. The next sequential instruction will therefore be skipped.

As discussed in a later sheet, the instruction word could be obtained from bootstrap or control memory.

2. Essential Commands. Refer to table 5.23-1 for a sequential list of essential I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams. The analysis of the skip evaluation logic is presented later in this sheet.

3. Skip Evaluation Logic.

a) Clear P and Adder → P Command Enable

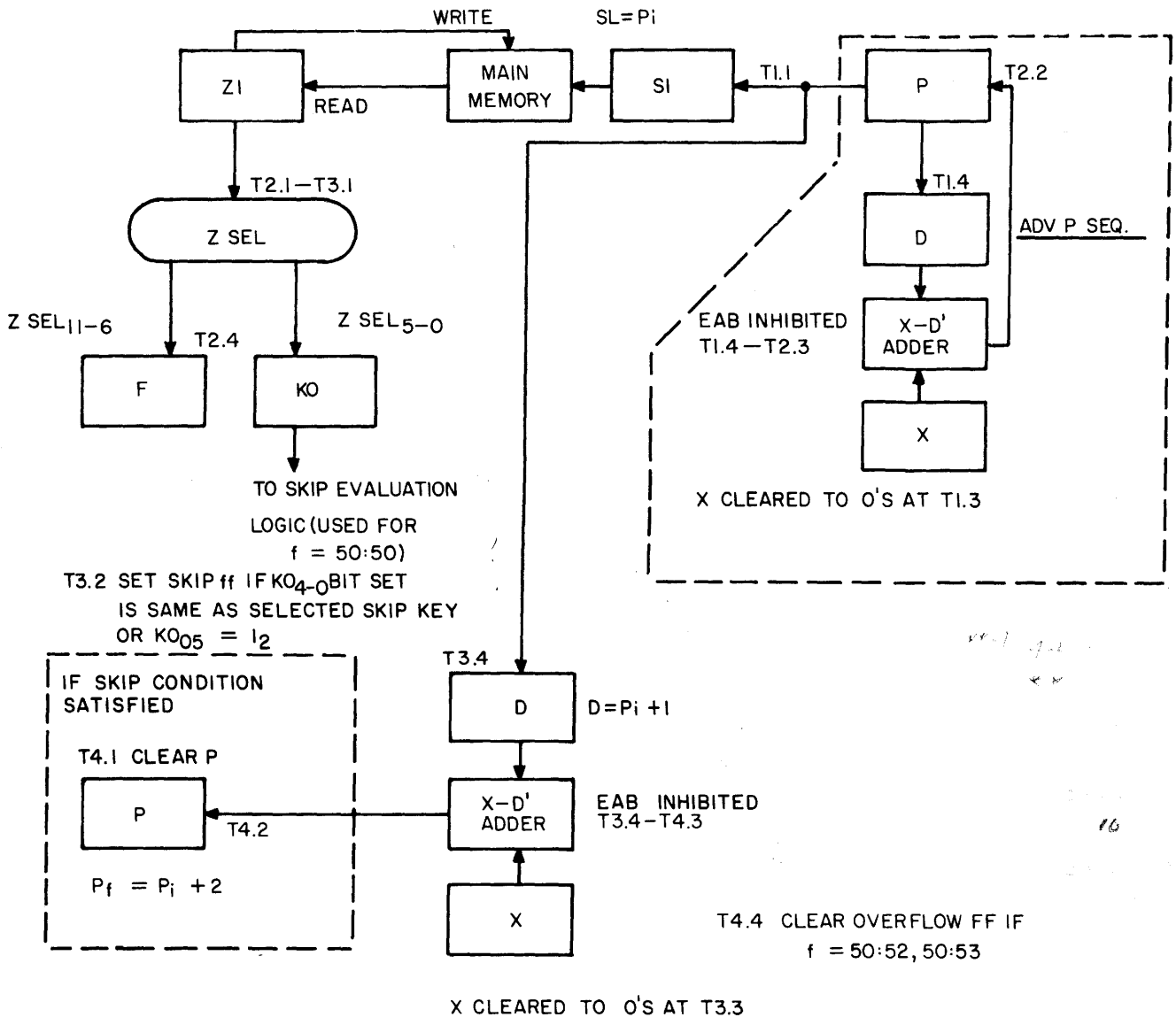
Refer to logic diagrams, figure 9-21 for the clear-P and Adder → P command logic.

When the T42 flip-flop is set during the I-sequence for these instructions, gate 10N07 outputs a high level if its input 20G50 is at a low level. The high level from 10N07 allows the program skip by setting P to the new address. The logic level of input 20G50 is conditioned by the skip evaluation logic.

b) SKP Instruction, f = 50:50. Refer to logic diagrams, figure 9-30 for the skip evaluation logic.

For f = 50:50, gate 20G50 has high levels applied to its input, pins 5, 7, 9, and 11. Input pin 13 is at a low level. If input pin 14 is at a high level, the resulting low level from 20G50 will cause a program skip. Pin 14 is at a low level if the Skip flip-flop is set. Refer to logic diagrams, figure 9-31 for the Skip flip-flop.

The Skip flip-flop is set at T3.2 time if gate 16G50 outputs a high level or if bit 5 of KO equals 1_2. KO_5 corresponds to bit 5 of the instruction word. When set, this bit then causes an unconditional skip.



NOTE: * SKIP IF f = 50:50, SKIP FF SET
f = 50:51, BORROW TEST FF CLEAR
f = 50:52, OVERFLOW FF SET
f = 50:53, OVERFLOW FF CLEAR

Figure 5.23-1. I-Sequence Data Flow For f = 50:50 - 50:53

TABLE 5.23-1. I SEQUENCE ESSENTIAL COMMANDS FOR f = 50:50 - 50:53

TIME NOTATION	COMMANDS
T4.4	Clear S1
T1.1	P → S1, Init memory, *set Incr P ff
T1.3	*Clear D, *clear X, clear Z1, clear F, *set OXL11 ff
T1.4	*P_L → D_L, * P_U → D_U, clear KO, *set Inhib EAB ff
T2.1	*Clear P, Z1 → Z Sel, *clear Incr P ff
T2.2	*Adder → P
T2.3	*Clear OXL11 ff, *clear Inhib EAB ff
T2.4	Z Sel_{11-6} → F, set OXF06 ff, Z Sel_{5-0} → KO
T3.1	Clear Skip ff, drop Z1 → Z Sel
T3.2	Set Skip ff if KO_{05} = 1 or KO_{04-00} bit set same as selected skip key
T3.3	Clear D, clear X
T3.4	P_L → D_L, P_U → D_U, set Inhib EAB ff
T4.1	Clear P if skip condition satisfied**
T4.2	Adder → P if skip condition satisfied**
T4.3	Clear Inhib EAB ff
T4.4	Clear Overflow ff if f = 50:52, 50:53

* These events are concerned with or are controlled by the advance-P subsequence.

** Skip condition satisfied if: f = 50:50, Skip ff set
f = 50:51, Borrow Test ff clear
f = 50:52, Overflow ff set
f = 50:53, Overflow ff clear

16G50 outputs a high level if any bit of KO_{4-0} is set and the corresponding skip key has been manually selected.

c) SKPNBO Instruction, f = 50:51. Refer to logic diagrams, figure 9-30.

For f = 50:51, gate 20G50 has high levels applied to its input pins 5, 7, 11, and 13. Input pin 9 is at a low level. If input pin 10 is at a high level, the resulting low level from 20G50 will cause a program skip. Pin 10 is at a high level if the Borrow Test flip-flop is clear, indicating a no borrow condition from the previous f = 20-23 instruction.

d) SKPOV and SKPNOV Instructions, f = 50:52, 50:53. Refer to logic diagrams, Figure 9-30.

For f = 50:52, 50:53, gate 20G50 has high levels applied to its input pins 7, 9, 11, and 13. Input pin 5 is at a low level. If input pin 6 is at a high level, the resulting low level from 20G50 will cause a program skip. The level on pin 6 is conditioned by the Overflow flip-flop. Refer to table 5.23-2 for the conditions necessary to produce a high level from gate 12G52 for f = 50:52, 50:53.

TABLE 5.23-2. CONDITIONS TO PRODUCE HIGH LEVEL FROM 12G52 FOR f = 50:52, 50:53

CONDITIONS	GATE OUTPUT LEVELS 10G52	11G52
f = 50:52·Overflow ff set	H	L
f = 50:53·Overflow ff clear	L	H

5.23-5. SUMMARY

The f = 50:50 - 50:53 instructions are format 2. The f = 50:50 instruction is the only one of this group that uses the value k. The k value is available in KO after T2.4 time. Only the I-sequence is required to complete the executions of these instructions.

SECTION 5 - CONTROL SECTION

5.24. INSTRUCTION EXECUTION OF BSK

5.24-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of the instruction with f = 56.

5.24-2. INTRODUCTION

This instruction performs a program skip if B equals the operand from memory. If they are not equal, B is incremented by +1.

5.24-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11 and 4-12.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.24-4. INFORMATION

a. General Description.

1. Instruction Interpretation. This instruction, BSK, obtains the operand Y from memory at the address U_P and compares this value with B. register is specified by ICR. If B = Y, a program skip is performed. If B $\neq$ Y, the content of B is incremented by +1.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of the operand is formulated from U and P.

b) R1-Sequence. The R1-sequence uses a memory reference to obtain the operand and compares Y with B. P is modified to effect a skip if B = Y.

c) Next I-Sequence. If B $\neq$ Y, the value of B + 1 is stored in control memory during the first portion of the I-sequence for the next instruction.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address U_P.

2. Data Flow Block Diagram. Refer to figure 5.24-1 for a block diagram description of the execution of f = 56.

The R1-sequence obtains the operand Y from memory and places its complement in D. A control memory reference is used to obtain the content of the B register specified by ICR which is placed in X. The adder is used to compare X and D'. The Equal flip-flop is set if X = D' (B = Y).

If X = D', the advance-P subsequence is used to increment P by +1. This incrementing changes P from the address of the instruction to be skipped to the following instruction.

A second control memory reference is used to again obtain the content of the index register. This value is placed in B and is incremented by the B-network. The incremented value of B + 1 is used only if the Equal flip-flop is clear which indicates that B $\neq$ Y.

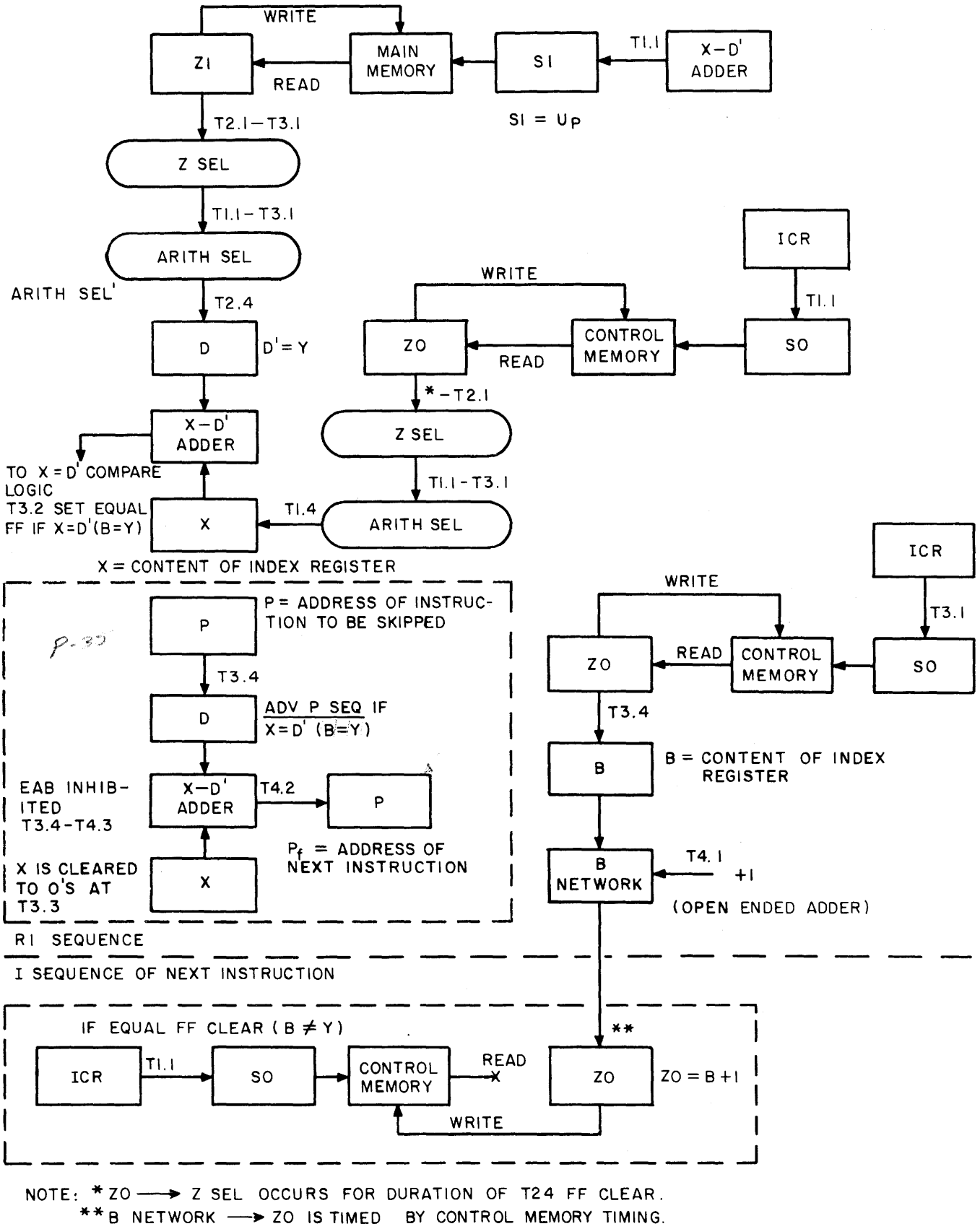
During the I-sequence, for the next instruction to be executed, another control memory reference is performed if the Equal flip-flop is clear. The value of B + 1 is stored in the B register address from Z0. The gating of control memory to Z0 is prevented during the read portion of the memory cycle which destroys the original memory content.

As discussed in a later sheet, the operand Y could be obtained from bootstrap or control memory.

3. Essential Commands. Refer to table 5.24-1 for a sequential list of essential R1 and next I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.24-5. SUMMARY

The BSK instruction uses the value Up which is formulated in D during the I-sequence. The R1-sequence and first portion of the next I-sequence are required to complete the execution of this instruction.



NOTE: * ZO ⟶ Z SEL OCCURS FOR DURATION OF T24 FF CLEAR.
** B NETWORK ⟶ ZO IS TIMED BY CONTROL MEMORY TIMING.

Figure 5.24-1. R1 and Next I-Sequence Data Flow for f = 56

TABLE 5.24-1. R1 AND NEXT I-SEQUENCE ESSENTIAL COMMANDS FOR f = 56

TIME NOTATION	COMMANDS
	R1 SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init Memory, ICR → SO, Init CM, *Z Sel → Arith Sel
T1.3	Clear Z1, clear X
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, *drop Z0 → Z Sel
T2.3	Clear D
T2.4	Arith Sel' → D
T3.1	ICR → SO, Init CM, set Incr P ff, drop Z1 → Z Sel, drop Z Sel → Arith Sel
T3.2	Clear B, set Equal ff if X = D', clear Incr P ff if X ≠ D'
T3.3	**Clear D, **clear X, **set OXL11ff
T3.4	**P_L → D_L, **P_U → D_U, **set Inhib EAB ff, Z0 → B
T4.1	**Clear P, **clear Incr P ff, clear B±1 ff (+1 → B Network
T4.2	**Adder → P
T4.3	**Clear OXL11 ff, **clear Inhib EAB ff
T4.4	Disable CM → Z0**
	I=SEQUENCE OF NEXT INSTRUCTION
T1.1	ICR → SO & Init CM if Equal ff clear
T1.4	Drop disable CM → Z0

* Z0 → Z Sel occurs for duration of T24 ff clear.

** These events are concerned with or are controlled by the advance-P subsequence which is disabled if X ≠ D' (B ≠ Y) at T3.2 time.

*** B-network → Z0 is timed by control memory timing.

NAME: _____

5.24-6. STUDY QUESTIONS

a. It is determined that the f = 50:50 - 50:53 instructions are able to perform program skips properly. However, it is found that the f = 56 instruction cannot execute a skip. Indicate which of the following malfunctions might be suspected.

Suspect?

1. 1OL1O constant low level output (logic diagrams, figure 9-35) _____

2. 1ON07 constant low level output (logic diagrams, figure 9-21) _____

3. Pin 15 of OXG33 ff grounded (logic diagrams, figure 9-27) _____

SECTION 5 - CONTROL SECTION

5.25. INSTRUCTION EXECUTION OF ISK

5.25-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 57.

5.25-2. INTRODUCTION

This instruction performs a program skip if the content of memory Y equals +0. If not, Y is decremented by 1.

5.25-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, tables 4-11, 4-12, and 4-14.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.25-4. INFORMATION

a. General Description.

1. Instruction Interpretation. This instruction, ISK, obtains the operand Y from memory at the address Up. If Y = +0, a program skip is performed. If Y $\neq$ +0, Y is decremented by 1 and is replaced in memory.

2. Execution Sequences.

a) I-Sequence. During the I-sequence which obtains the instruction from memory, the address of the operand is formulated from U and P.

b) R1-Sequence. The R1-sequence uses a memory reference to obtain the operand and examines it. If Y $\neq$ +0, it is decremented by -1.

c) W-Sequence. The W-sequence uses a memory reference to store either Y or Y - 1. P is modified to effect a skip if Y_i = +0.

b. Detailed Analysis.

1. I-Sequence. The I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description. At the end of the I-sequence, the X-D' adder is outputting the address.

2. Data Flow Block Diagram. Refer to figure 5.25-1 for a block diagram description of the execution of f - 57.

The R1-sequence obtains the operand Y from memory and places it in D from where it is applied to one side of the X-D' adder. X receives -0 from arithmetic-select which has no input. If X = D', then Y' = -0 or Y = +0 and the adder output is (-0) - (-0) or +0 which is Y. Since X is set to all 1's, no end-around borrow can occur. If X ≠ D', an end-around borrow is inserted which causes the adder output to be Y - 1.

During the W-sequence, Z1 receives the adder output which is stored in memory at the same address from where Y was obtained. This address was placed in S1 by the R1 Sequence and is not cleared out. The gating of memory to Z1 is prevented during the read portion of the memory cycle which destroys the original memory content (Y).

If X = D' (Y_i = +0), the next sequential instruction of the program is to be skipped. The advance-P subsequence is used to increment P. This incrementing changes P from the address of the skipped instruction to the address of the following instruction.

As discussed in a later sheet, the operand Y could be obtained from bootstrap or control memory. Since bootstrap has nondestructive read-out, storage of the adder output into this memory would have no effect. Bootstrap would still retain the original Y value.

3. Essential Commands. Refer to table 5.25-1 for a sequential list of essential R1 and W-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.25-5. SUMMARY

The ISK instruction uses the value U_P which is formulated in D during the I-sequence. The R1 and W-sequences are required to complete the execution of this instruction.

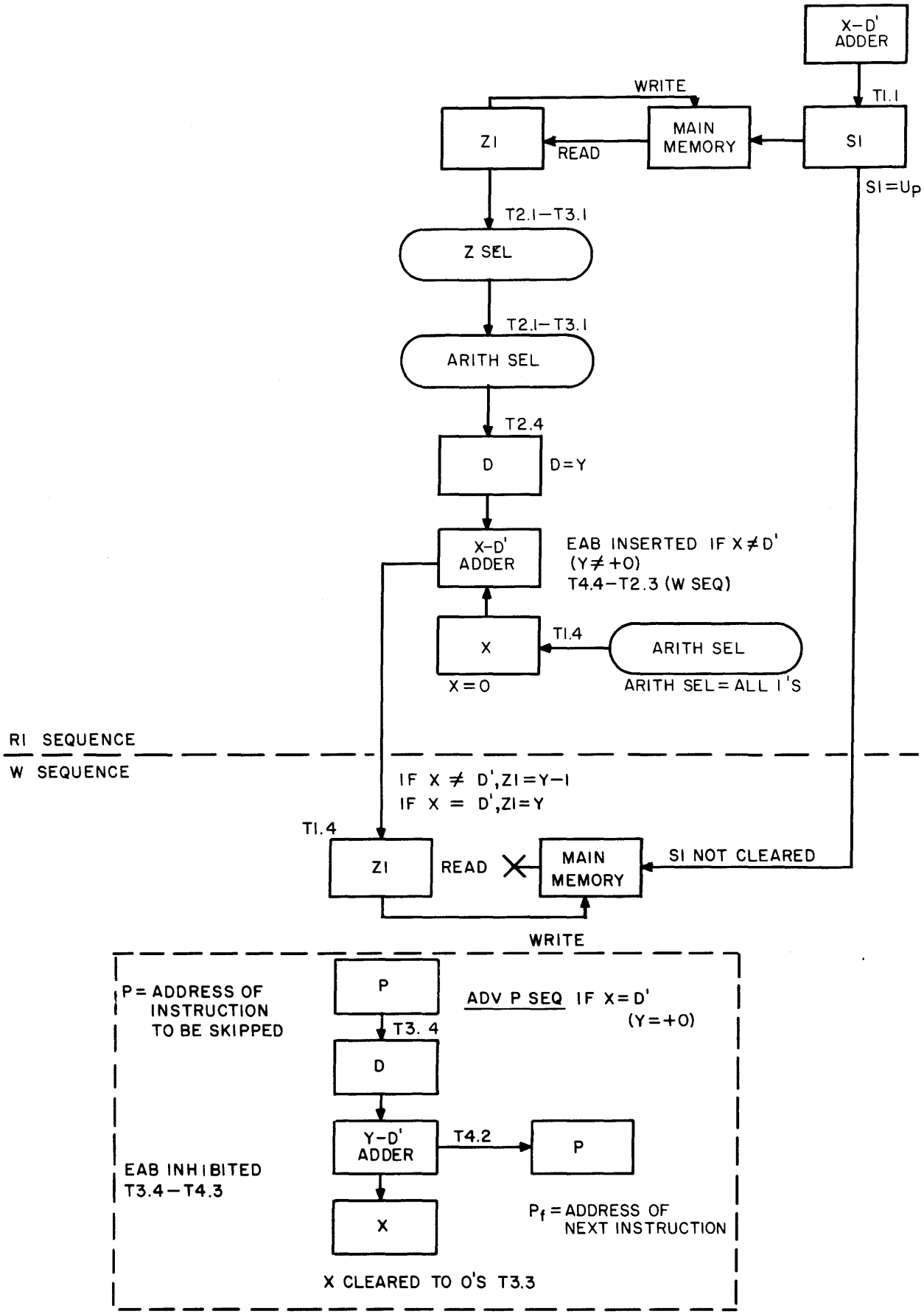


Figure 5.25-1. R1 and W Sequence Data Flow for f = 57

TABLE 5.25-1. R1 AND W SEQUENCE ESSENTIAL COMMANDS FOR f = 57

TIME NOTATION	COMMANDS
	R1 SEQUENCE
T4.4	Clear S1
T1.1	Adder → S1, Init Memory
T1.3	Clear Z1, Clear X
T1.4	Arith Sel → X
T2.1	Z1 → Z Sel, Z Sel → Arith Sel
T2.3	Clear D
T2.4	Arith Sel → D
T3.1	Drop Z1 → Z Sel, drop Z Sel → Arith Sel
T4.4	Set Insert EAB ff if X ≠ D'
	W SEQUENCE
T1.1	Init Memory
T1.3	Clear Z1
T1.4	Adder → Z1, disable Mem → Z1
T2.3	Clear Insert EAB ff
T2.4	Drop disable Mem → Z1
T3.1	*Set Incr P ff if X = D'
T3.3	*Clear D, *Clear X, *set OXL11 ff
T3.4	$*P_L \rightarrow D_L$, $*P_U \rightarrow D_U$, * set Inhib EAB ff
T4.1	*Clear P, *clear Incr P ff
T4.2	*Adder → P
T4.3	*Clear OXL11 ff, *clear Inhib EAB ff

* These events are concerned with or are controlled by the advance-P subsequence which is initiated only if X = D' ($Y_i = +0$).

NAME: _____

5.25-6. STUDY QUESTIONS

a. Given: 30N13 constant low level output (logic diagrams, figure 9-23)
initial content of address 034400 = 056000

Address	Instruction
032000	574400

After the execution of the instruction at address 032000, what is the content of memory at address 034400?

$Memory_f$ at address 034400 = _____

SECTION 5 - CONTROL SECTION

5.26. INSTRUCTION EXECUTION OF BJP

5.26-1. OBJECTIVES

To present the detailed theory of operation involved in the execution of instructions with f = 73.

5.26-2. INTRODUCTION

This instruction performs a program jump if B $\neq$ +0 and then decrements B by 1. Nothing is affected if B = +0.

5.26-3. REFERENCES

a. UNIVAC 1219 Technical Manual, Volume I, Paragraph 4-7, table 4-11.

b. UNIVAC 1219 Technical Manual, Volume II, Section 9 (logic diagrams).

5.26-4. INFORMATION

a. General Description.

1. Instruction Interpretation. This instruction, BJP, obtains the content of the B register specified by ICR. If B = +0, nothing is effected. If B $\neq$ +0, B is decremented by 1 and a program jump is executed to the address U_P.

2. Execution Sequences.

a) I Sequence. During the I-sequence which obtains the instruction from memory, B is obtained and examined. P is modified to effect a skip if B = +0.

b) Next I Sequence. If B $\neq$ 0, the storage of B-1 back into control memory is performed during the first portion of the I-sequence for the next instruction.

b. Detailed Analysis.

1. Data Flow Block Diagram. Refer to figure 5.26-1 for a block diagram description of the execution of f = 73.

Most of the I-sequence operations are as previously described. If necessary, refer to study guide sheet number 5.4 for a detailed description.

The value of U_P is formulated in D from where it is applied to one side of the X-D' adder. X receives -0 from arithmetic-select which has no input. The adder output of (-0) - $(U_P)'$ if effectively U_P.

A control memory reference is used to obtain the content of the B register specified by ICR. If B = +0, P is cleared and receives the jump address U_P The following I-sequence actually effects the jump by obtaining the next instruction from the address contained in P.

During the first portion of the next I-sequence, another control memory reference is used to store the value of B-1 back into the B register address if $B \neq +0$. The gating of control memory to ZO is prevented during the read portion of the memory cycle which destroys the original memory content.

2. Essential Commands. Refer to table 5.26-1 for a sequential list of essential I and next I-sequence events. Develop these commands by referring to the proper enable pages in the logic diagrams.

5.26-5. SUMMARY

The BJP instruction uses the value U_P which is formulated in D during the I-sequence. The I-sequence and first portion of the next I-sequence are required to complete the execution of this instruction.

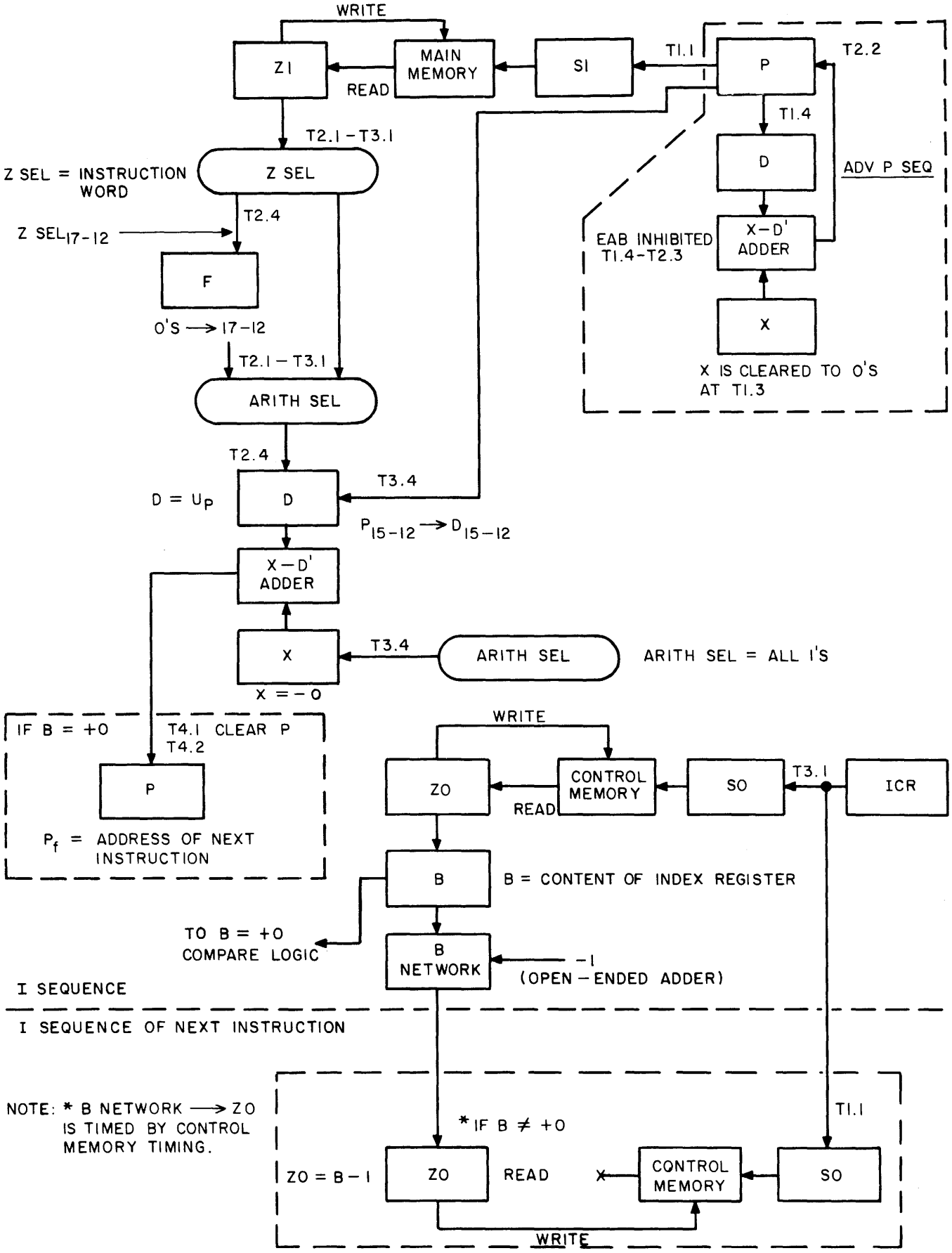


Figure 5.26-1. I and Next I-Sequence Data Flow For f = 73

TABLE 5.26-1. I AND NEXT I SEQUENCE ESSENTIAL COMMANDS FOR f = 73

TIME NOTATION	COMMANDS
	I SEQUENCE
T4.4	Clear S1
T1.1	P → S1, Init Memory, *set Incr P ff
T1.3	*Clear D, *clear X, clear F, clear Z1, *set OXL11 ff
T1.4	$*P_L \rightarrow D_L$, $*P_U \rightarrow D_U$, *set Inhib EAB ff
T2.1	*Clear P, Z1 → Z Sel, Z Sel → Arith Sel, 0's → Arith Sel_{17-12} *clear Incr P ff
T2.2	*Adder → P
T2.3	Clear D, clear X, *clear OXL11 ff, *clear Inhib EAB ff
T2.4	Z $Sel_{17-12} \rightarrow$ F, Arith Sel → D
T3.1	ICR → S0, Init CM, drop Z1 → Z Sel, drop Z Sel > Arith Sel
	drop 0's → Arith Sel_{17-12}
T3.2	Clear B
T3.4	$P_{15-12} \rightarrow D_{15-12}$, Arith Sel → X, Z0 → B
T4.1	Clear P if B ≠ +0, clear B± 1 ff
T4.2	Adder → P if B ≠ +0, set B±1 ff (-1 → B Network)
T4.4	Disable CM → Z0**
	I SEQUENCE OF NEXT INSTRUCTION
T1.1	ICR → S0 & Init CM if B ≠ +0
T1.4	Drop disable CM → Z0

* These events are concerned with or are controlled by the advance-P subsequence.

** B network → Z0 is timed by control memory timing.