SAP-2

Chapter-11

Prepared By: Lec Tasmiah Tamzid Anannya, CSE Dept, MIST

Introduction

- SAP-2 is the next step of SAP-1 in the evolution toward modern computer.
- It includes more instructions than SAP-1.
- These new instructions force computer to repeat or skip part of a program which opens up a new world for computer program.

SAP-1 Architecture

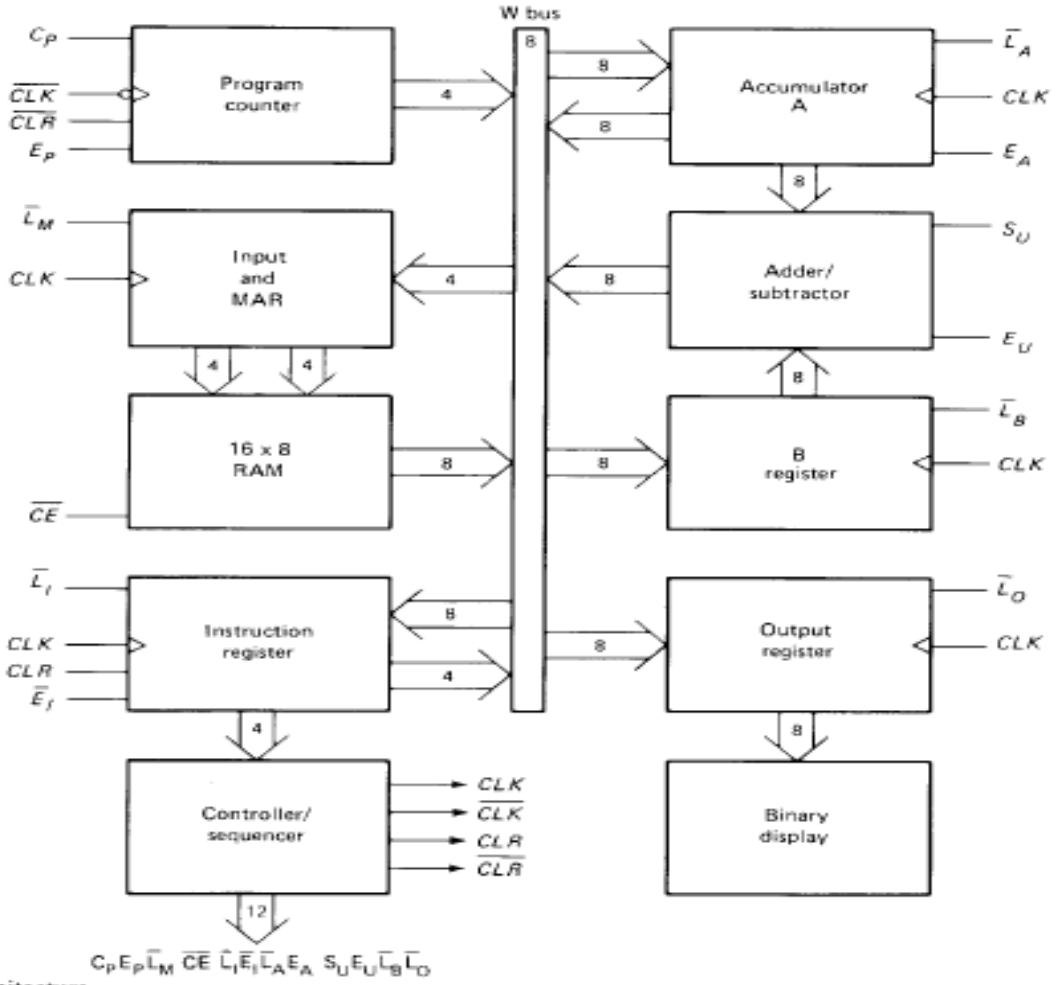


Fig. 10-1 SAP-1 architecture.

Bidirectional Registers

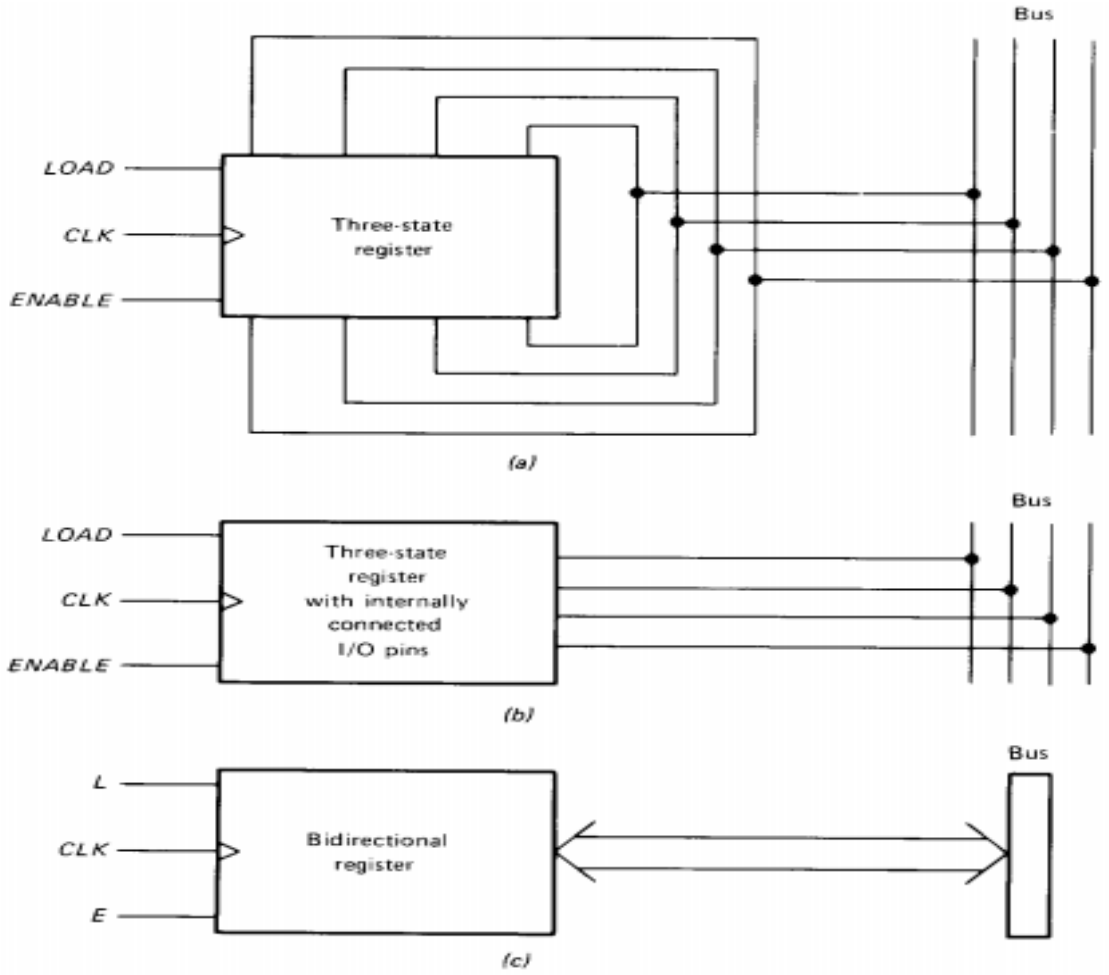


Fig. 11-1 Bidirectional register.

Architecture

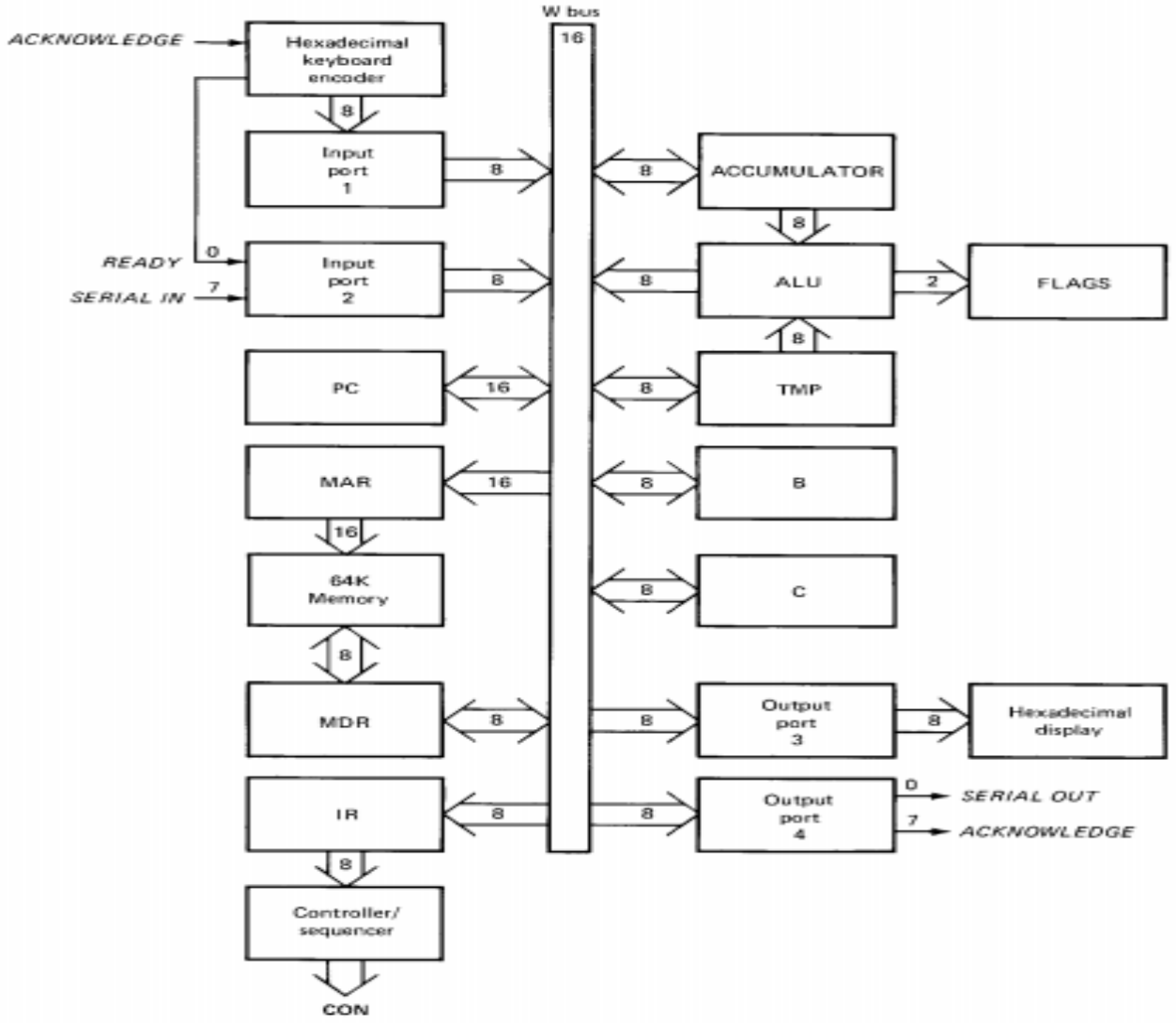


Fig. 11-2 SAP-2 block architecture.

Memory Reference Instructions

- SAP-2 fetch cycle as same as SAP-1.
  - T1 is address state
  - T2 is increment state
  - T3 memory state

- During the execution cycle, memory or memory may not be used.
- Memory reference instruction(MRI) is one that uses the memory during the execution cycle.

Memory Reference Instructions LDA and STA

- LDA has the same meaning as before.
- LDA 2000H means to load the accumulator with the contents of memory location 2000H.
- STA means store the accumulator.
- STA 7FFFH means to store the accumulator content at memory location 7FFFH.
- If A=8AH, after execution of STA 7FFFH, memory address 7FFFH will store 8AH.

Memory Reference Instructions
MVI

- MVI is the mnemonic for move immediate.
- It tells the computer to load a designated register with the byte that immediately follows up.

MVI A, 37H

- It tells the computer to load the accumulator with 37H.

MVI A, byte

MVI B, byte

MVI C, byte

Opcodes

TABLE 11-1. SAP-2 OP CODES

Instruction	Op Code	Instruction	Op Code
ADD B	80	MOV B,A	47
ADD C	81	MOV B,C	41
ANA B	A0	MOV C,A	4F
ANA C	A1	MOV C,B	48
ANI byte	E6	MVI A,byte	3E
CALL address	CD	MVI B,byte	06
CMA	2F	MVI C,byte	0E
DCR A	3D	NOP	00
DCR B	05	ORA B	B0
DCR C	0D	ORA C	B1
HLT	76	ORI byte	F6
IN byte	DB	OUT byte	D3
INR A	3C	RAL	17
INR B	04	RAR	1F
INR C	0C	RET	C9
JM address	FA	STA address	32
JMP address	C3	SUB B	90
JNZ address	C2	SUB C	91
JZ address	CA	XRA B	A8
LDA address	3A	XRA C	A9
MOV A,B	78	XRI byte	EE
MOV A,C	79		

Example

- Show the mnemonics for a program that loads the accumulator with 49H, B register with 4AH, C register with 4BH, then store the accumulator content at location 6285H.

```
MVI A, 49H
MVI B, 4AH
MVI C, 4BH
STA 6285H
HLT
```

Example

- Translate the previous program into machine language using opcode at address 2000H.

MVI A, 49H
MVI B, 4AH
MVI C, 4BH
STA 6285H
HLT

Opcodes:
MVI A-3EH
MVI B-06H
MVI C-OEH
STA-32H
HLT-76H

Address	Contents
2000H	
2001H	
2002H	
2003H	
2004H	
2005H	
2006H	
2007H	
2008H	
2009H	

Example

- Translate the previous program into machine language using opcode at address 2000H.

MVI A, 49H
MVI B, 4AH
MVI C, 4BH
STA 6285H
HLT

Opcodes:
MVI A-3EH
MVI B-06H
MVI C-OEH
STA-32H
HLT-76H

Address	Contents
2000H	3EH (Opcode of MVI A)
2001H	49H
2002H	06H (Opcode of MVI B)
2003H	4AH
2004H	0EH (Opcode of MVI C)
2005H	4BH
2006H	32H (opcode of STA)
2007H	85H
2008H	62H
2009H	76H (opcode of HLT)

Register Instructions
MOV

- Memory reference instructions are relatively slow because they require more than one memory access during the instruction cycle.
- Furthermore, we often want to move data directly from one register to another not going through memory.
- MOV stands for move.
- It tells computer to move data from one register to another.

```
MOV A,B
MOV A,C
MOV B,A
MOV B,C
MOV C,A
MOV C,B
```

Suppose,
A=34H
B=05H
After MOV A,B
A=05H
B=05H

Register Instructions
ADD & SUB

- ADD stands for add the data in the designated register to the contents of accumulator.

ADD B

ADD C

- Similarly, SUB means subtract the data in the designated register from the accumulator content.
- SUB C will subtract the data of C register from the accumulator.

SUB B

SUB C

Register Instructions
INR & DCR

- INR A
- INR B
- INR C
- DCR A
- DCR B
- DCR C

Example

- Show the mnemonic for adding decimal 23 and 45. the answer is to be stored at location 5600H. Also increment the answer by 1 and store it in C register.

```
MVI A, 17H
MVI B, 2DH
ADD B
STA 5600H
INR A
MOV C,A
```

Example

```
MVI A, 17H
MVI B, 2DH
ADD B
STA 5600H
INR A
MOV C,A
```

- Hand assemble means to translate a source program to machine language program by hand rather than machine.
- Hand-assemble the previous program.

Address	Contents
2000H	
2001H	
2002H	
2003H	
2004H	

Address	Contents
2005H	
2006H	
2007H	
2008H	
2009H	
200AH	

Example

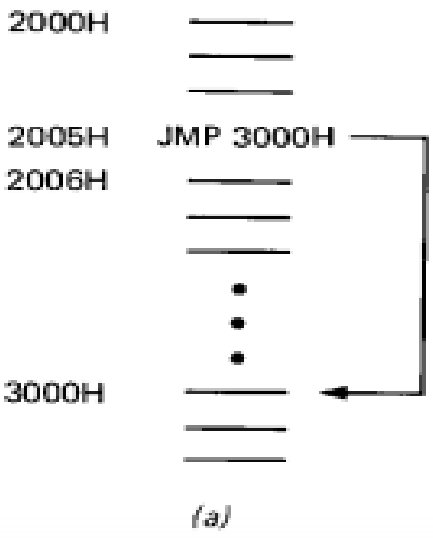
- Hand assemble means to translate a source program to machine language program by hand rather than machine.
- Hand-assemble the previous program.

Address	Contents
2000H	Opcode of MVI A
2001H	17H
2002H	Opcode of MVI B
2003H	2DH
2004H	Opcode of ADD B

Address	Contents
2005H	Opcode of STA
2006H	00H
2007H	56H
2008H	Opcode of INR
2009H	Opcode of MOV C,A
200AH	Opcode of HLT

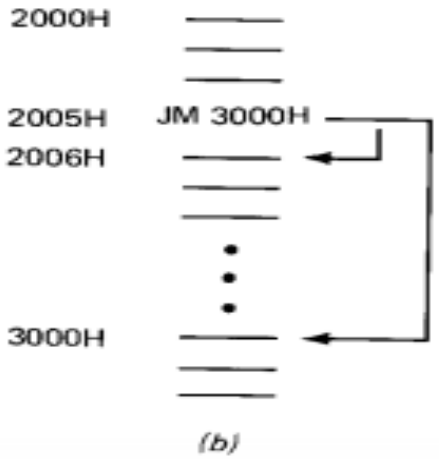
JUMP Instructions
JMP

- JMP is the mnemonic for Jump
- It tells the computer to get the next instruction from the designated memory location.
- Every JMP instruction includes an address that is loaded into the program counter.
- JMP 3000H tells the computer to get the next instruction from memory location 3000H.



JUMP Instructions
JM

- During the execution of a instruction, if the accumulator contents become negative, the sign flag will be set.
- $S = \begin{cases} 0, if\ A \geq 0 \\ 1, if\ A < 0 \end{cases}$
- JM is the mnemonic for jump if minus
  - The computer will jump to designated address if and only if sign flag (S) is set (1).



JUMP Instructions
JZ & JNZ

- During execution, if accumulator becomes zero, the zero flag will be set.

$$Z = \begin{cases} 0, when\ A \neq 0 \\ 1, when\ A = 0 \end{cases}$$

- JZ is the mnemonic for jump if zero.
- JNZ stands for jump if not zero.

- JM, JZ, JNZ are conditional jumps because the program jump occurs if certain conditions are satisfied.
- On the other hand, JMP is unconditional, always jumps to the specified address.

CALL & RET

- CALL is the mnemonic for call the subroutine.
- A subroutine is a program stored in memory for possible use by various programs.
- Like finding sines, cosines, tangents, square roots etc.
- Every call instruction must include the starting address of the desired subroutine.

CALL 5000H

- RET stands for return.
- It is used at the end of the subroutine.
- When CALL is executed automatically PC contents are stored in FFFEH and FFFFH (last two memory locations).
- CALL is unconditional.

Example

- Show a program that multiple decimal 12 and 8.

```
        MVI A,00H
        MVI B,0CH
        MVI C, 08H
REPEAT: ADD B
        DCR C
        JZ DONE
        JMP REPEAT
DONE:   HLT
```

```
        MVI A,00H
        MVI B,0CH
        MVI C, 08H
REPEAT: ADD B
        DCR C
        JNZ REPEAT
        HLT
```

Example

- Hand-assemble the program starting at address 2000H.

Address	Contents
2000H	
2001H	
2002H	
2003H	
2004H	
2005H	
2006H	
2007H	
2008H	
2009H	

Address	Contents
200AH	
200BH	
200CH	
200DH	
200EH	

```
        MVI A,00H
        MVI B,0CH
        MVI C, 08H
REPEAT: ADD B
        DCR C
        JZ DONE
        JMP REPEAT
DONE:   HLT
```

Example

- Hand-assemble the program starting at address 2000H.

Address	Contents
2000H	Opcode of MVI A
2001H	00H
2002H	Opcode of MVI B
2003H	0CH
2004H	Opcode of MVI C
2005H	08H
2006H	Opcode of ADD B
2007H	Opcode of DCR C
2008H	Opcode of JZ
2009H	0EH

Address	Contents
200AH	20H
200BH	Opcode of JMP
200CH	06H
200DH	20H
200EH	Opcode of HLT

Example

- Change the multiplication part into a subroutine located at starting address F006H.

Address	Contents
F006H	
F007H	
F008H	
F009H	
F00AH	
F00BH	

```
REPEAT:  ADD B
         DCR  C
         JNZ REPEAT
         RET
```

Example

- Change the multiplication part into a subroutine located at starting address F006H.

Address	Contents
F006H	Opcode of ADD B
F007H	Opcode of DCR C
F008H	Opcode of JNZ
F009H	06H
F00AH	F0H
F00BH	Opcode of RET

```
REPEAT:  ADD B
         DCR  C
         JNZ REPEAT
         RET
```

Main Program using this subroutine:

```
MVI A, 00H
MVI B,0CH
MVI C, 08H
CALL F006H
HLT
```

Logic Instructions

- CMA
  - Complement the accumulator
  - Inverts each bit of the accumulator
- ANA
  - And the accumulator content with the designated register
  - ANA B & ANA C
- ORA
  - Or the accumulator content with the designated register
  - ORA B & ORA C
- XRA
  - Xor the accumulator content with the designated register
  - XRA B & XRA C

Logic Instructions

- ANI
  - And immediate
  - And the accumulator contents with the byte that immediately follows the opcode.
  - ANI C7H (1100 0111)
  - Suppose, A=0101 1110
  - Then the new content of A will be 0100 0110
- ORI
  - Or immediate
  - Or the accumulator contents with the byte that immediately follows the opcode.
- XRI
  - Xor immediate
  - Xor the accumulator contents with the byte that immediately follows the opcode.

Other Instructions

- NOP
  - Stands for no operation.
  - During execution of NOP, all T states do nothings, no register changes occur.
- HLT
  - It ends data processing
- IN
  - Takes input
  - Tells computer to take data from designated port to the accumulator-
  - IN byte(port number)
  - IN 01H & IN 02H
- OUT
  - Gives output
  - Accumulator word is loaded into the designated output port.
  - OUT byte(port number)
  - OUT 03H & OUT 04H

Other Instructions

- RAL
  - Rotate the accumulator left
  - Will shift all bits to the left and move MSB into LSB.
  - If A= 1011 1101
  - After RAL, A= 0111 1011
- RAR
  - Rotate the accumulator right.
  - If A= 1011 1101
  - After RAR, A= 1101 1110

Example

- Take input from port 2 and determine whether the LSB is 0 or 1.
  - If 0, then load ASCII value of Y to the accumulator
  - else load ASCII value of N to accumulator

```
        IN 02H
        ANI 01H
        JNZ YES
        MVI A, 4EH  (4EH is ASCII of N)
        JMP DONE
YES:    MVI A, 59H  (59H is ASCII of Y)
DONE:   OUT 03H
        HLT
```

Exampele

- Modify the previous program if we want serial output at port 4 instead of parallel output at port 3.

```
        IN 02H
        ANI 01H
        JNZ YES
        MVI A, 4EH  (4EH is ASCII of N)
        JMP DONE
YES:    MVI A, 59H  (59H is ASCII of Y)
DONE:   MVI C, 08H
AGAIN:  OUT 04H
        RAR
        DEC C
        JNZ AGAIN
        HLT
```

Handshaking in SAP-2

- Handshaking is an interaction between a CPU and a peripheral device during an I/O data transfer.

- The sequence of SAP-2 handshaking:
  - i. READY bit goes high.
  - ii. Input the data from port 1.
  - iii. ACK bit goes high to reset the READY bit.
  - iv. Reset the ACK bit.

Example

- Take a input from port 1 using handshaking.

```
AGAIN:  IN 02H
        ANI 01H
        JZ AGAIN
        IN 01H
        MOV B,A
        MVI A, 80H
        OUT 04H
        MVI A, 00H
        OUT 04H
        HLT
```

T states

- Variable machine cycle.
- For example, ADD C takes 4 T states to execute and ANI byte takes 7 T states, CALL takes 18 T states to execute.
- JM/JZ/JNZ has T states of 10/7.
  - 10 T states when jump occurs
  - 7 T states when jump does not occur.

Flags

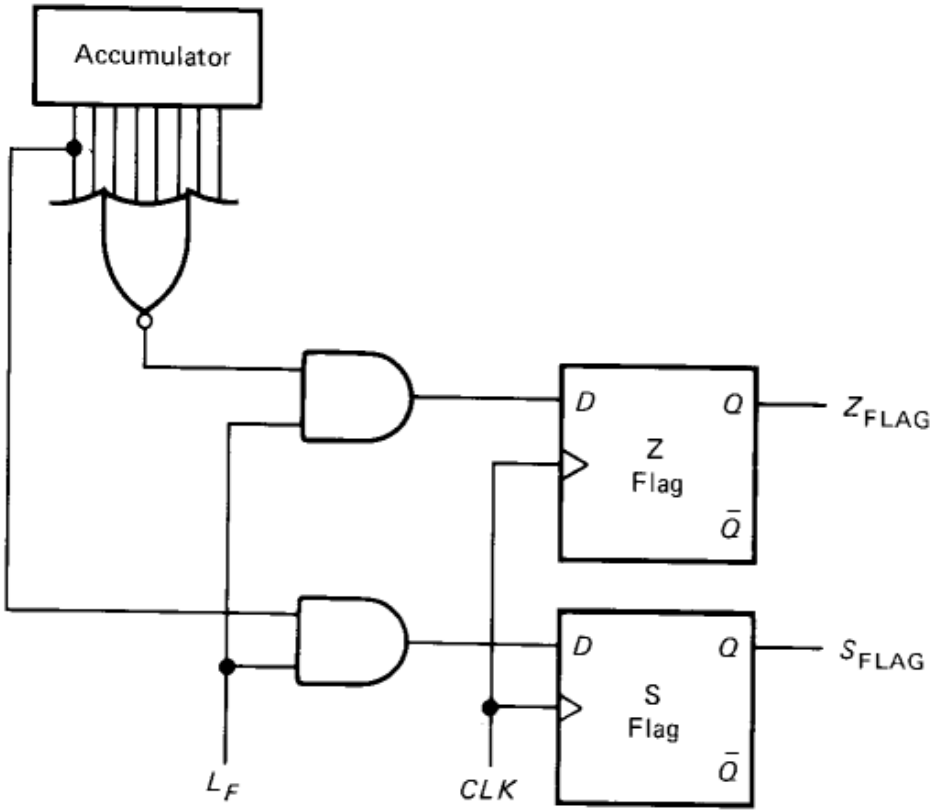


Fig. 11-8 Setting the flags.

Addressing modes

- Direct addressing
  - LDA/STA/IN/OUT
- Immediate addressing
  - MVI/ANI/ORI/XRI/CALL/JMP/JZ/JNZ/JM
- Register addressing
  - ADD/SUB/INR/DCR/MOV/ANA/ORA/XRA
- Implied addressing
  - CMA/RAR/RAL/RET

Bytes

- Each SAP instruction occupies a number of bytes in the memory.
- SAP-2 instructions are either 1/2/3 bytes long.
  - ADD instructions are 1 byte long
  - ANI instruction is 2 bytes long
  - CALL instruction is 3 bytes long

Example

- SAP-2 has a clock frequency of 1 MHz. This means each T state is 1 microsecond long.
- How does it take to execute the following SAP-2 subroutine?

```
        MVI C, 46H (decimal 70)
AGAIN:  DCR C
        JNZ AGAIN
        NOP
        RET
```

MVI: 1x7x1 μs
DCR: 70x4x1 μs
JNZ: 69x10x1 μs (jump)
1x7x1 μs $(no\ jump)$
NOP: 1x4x1 μs
RET: 1x10x1 μs

Total: 998 μs $(\text{almost } 1 \text{ ms})$

Bytes??

Example

```
        MVI B, 0AH
LOOP 1: MVI C, 47H (decimal 71)
LOOP 2: DCR C
        JNZ LOOP2
        DCR B
        JNZ LOOP1
        RET
```