SAP-3

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

Introduction

- 8 bit microcomputer
- Upward-compaitible with 8085
- Includes all the instructions of SAP-2
- New instructions included

Model

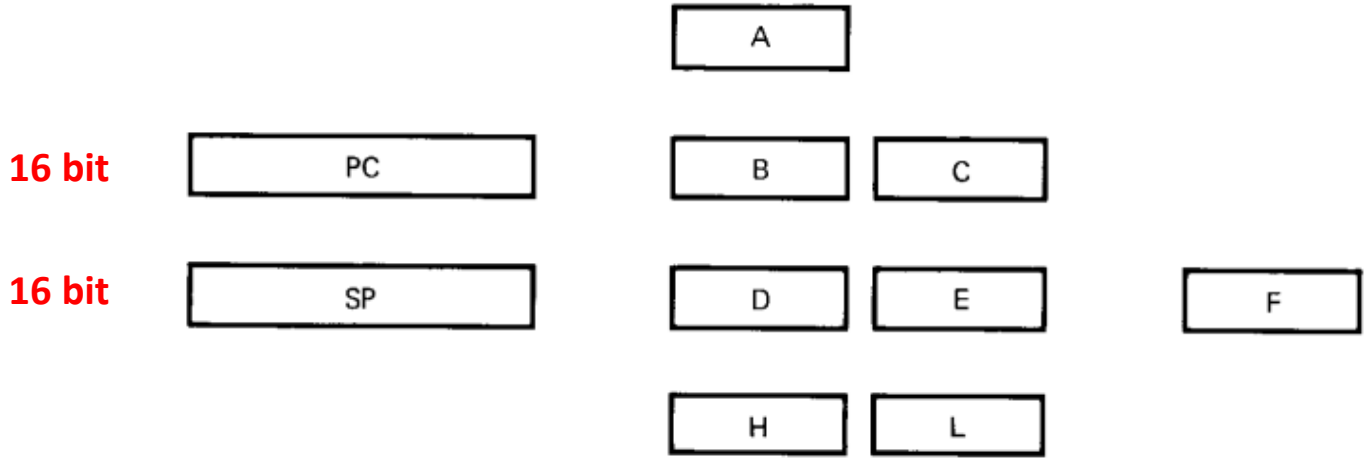


Fig. 12-1 SAP-3 programming model.

Instructions-
MOV & MVI

- Same as SAP-2
- Only difference is that more registers to choose from

Arithmetic instructions

- Carry flag

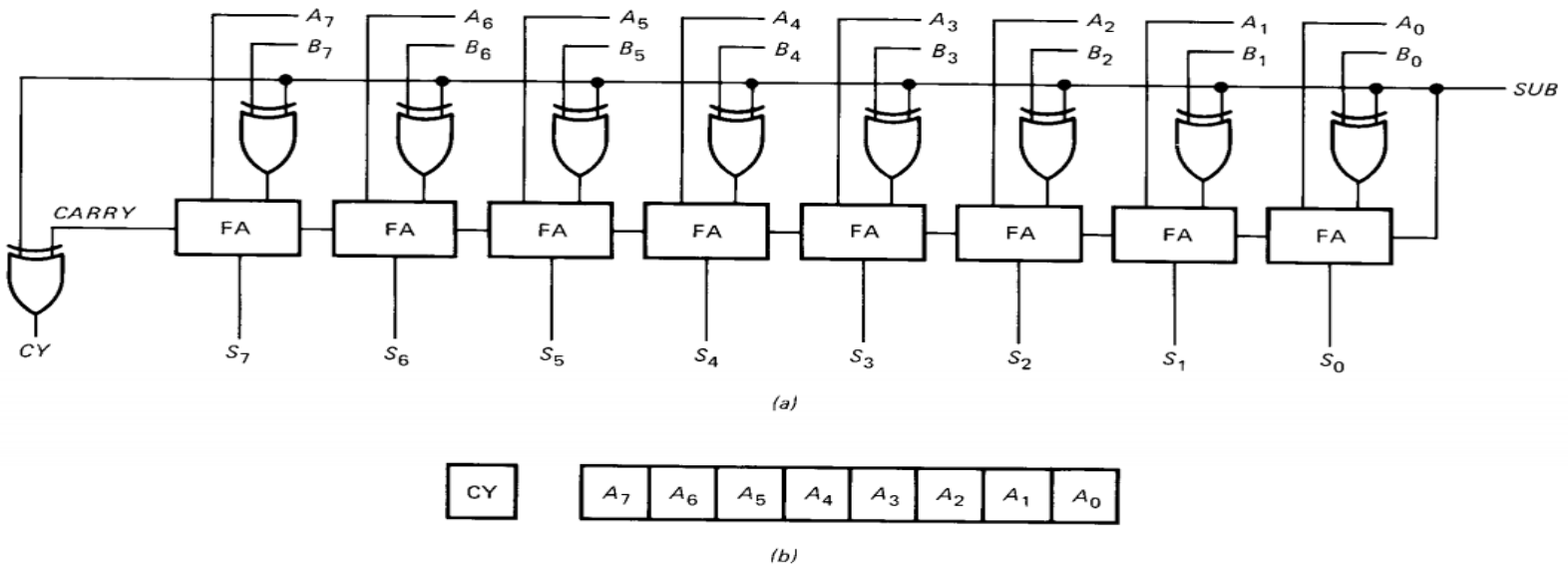


Fig. 12-2 (*a*) SAP-3 adder-subtractor (*b*) carry flag and accumulator.

- Carry flag instructions
  - STC (Set the carry flag)
  - CMC (complements the carry flag)

Present status of CY is unknown, you want to reset the flag what will you do?

Arithmetic instructions
ADD & ADC

- ADD instruction:
- ADC instruction:
  - Add with carry
  - For example: A= 1000 0011, E= 0001 0010, CY=1
  - After execution of ADC E
  - What will happen?

Arithmetic instructions
SUB & SBB

- SUB instruction:
  - During SUB, CY flag acts like a **borrow flag.**
- SBB instruction:
  - Subtract with borrow
  - For example: A= 1111 1111, E= 0000 0010, CY=1
  - After execution of SBB E
  - What will be the value of A and CY?

Example

- Show a SAP-3 program that adds 700 and 900, store the final answer in the H and L register.

700D=02BCH= 0000 0010 1011 1100
900D=0384H= 0000 0011 1000 0100

INCREMENTS, DECREMENTS

- INR reg
  - No effect on carry flag
  - Suppose, B=1111 1111 , S=1, Z=0, CY=0
  - After INR B, B=0000 0000, S=0, Z=1, CY=0
- DCR reg
  - No effect on carry flag

Rotate

- Rotate All Left (RAL)
- Rotate All Right (RAR)
  - including CY

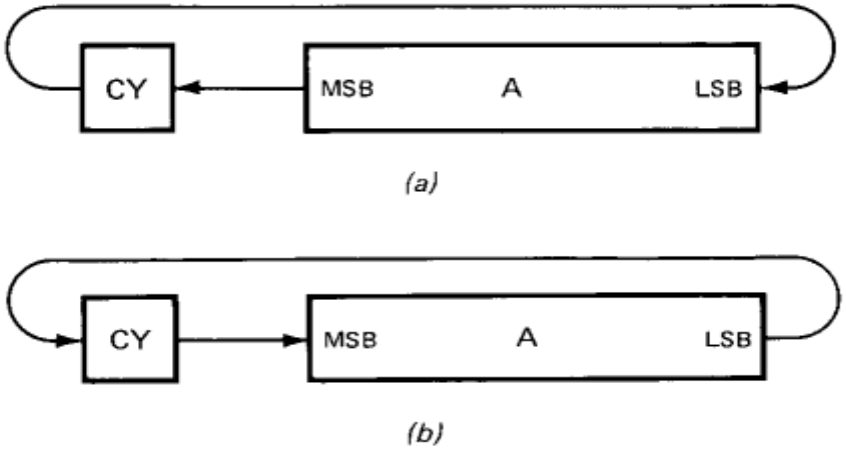


Fig. 12-3 (*a*) RAL; (*b*) RAR.

Rotate

- Rotate Left with Carry (RLC)
- Rotate right with Carry (RRC)

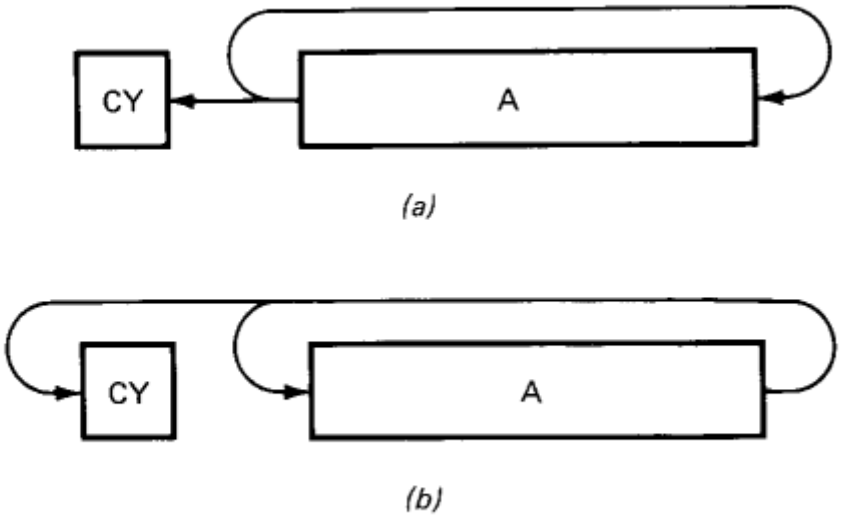


Fig. 12-4 (*a*) RLC; (*b*) RRC.

Rotate

- Multiply and divide by 2
  - With Carry flag reset, an RAL has the effect of multiplying by 2
  - CY=0, A=0000 0111
  - After RAL, CY=0, A=0000 1110
  - RAR, divides by 2.
  - After RAR, CY=0, A=0000 0111

Logic Instructions

- Similar as SAP-2
- ANA reg
- ORA reg
- XRA reg
- CMP reg (new in SAP-3)
  - Compares the specified register with the contents of accumulator
  - $Z = \begin{cases} 1, if\ A = reg \\ 0, if\ A \neq reg \end{cases}$
  - Contents of A reg is copied into a temp reg
  - Then contents of specifies reg is subtracted from the contents of temp reg
  - If the values are equal Zero flag is set
  - If unequal, zero flag is reset
  - If, A=F1H, E=F1H, Z=0, after CMP E, Z=1, A & D will be unaffected.

Immediates

- ANI byte – AND immediate byte with Accumulator
- ORI byte – OR immediate byte with Accumulator
- XRI byte – XOR immediate byte with Accumulator
- ADI byte - Add immediate byte with Accumulator
- ACI byte – Add immediate byte plus CY with Accumulator
- SUI byte – Subtract immediate byte from Accumulator
- SBI byte- – Subtract immediate byte and CY from Accumulator
- CPI byte – Compare immediate byte with Accumulator

Example

- Subtract 700 from 900 using SAP-3 program.

700D=02BCH= 0000 0010 1011 1100
900D=0384H= 0000 0011 1000 0100

Jump Instructions

- JMP address
- JM address
- JZ address
- JNZ address
- JP address
  - Jump if positive (including 0)
  - Opposite of JM
- JC & JNC
- JPE & JPO
  - Checks parity
  - If result has even number of 1's PF is set
  - If odd number of 1's, PF is reset

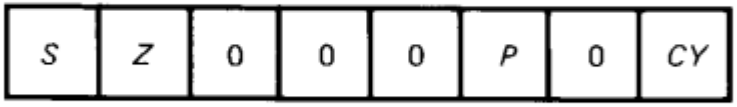


Fig. 12-5 F register stores flags.

Extended Register Instructions

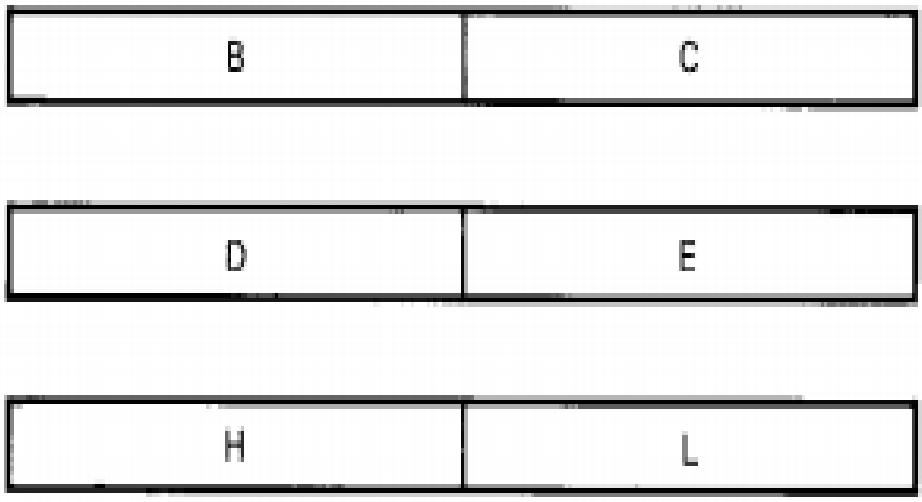


Fig. 12-6 Register pairs.

Extended Register Instructions

- Load Extended Immediate

LXI B,dble
LXI D,dble
LXI H,dble

where B stands for BC
D stands for DE
H stands for HL
dble stands for double byte

- Suppose, LXI B, 65FAH
- B=65H, C=FAH

Extended Register Instructions

- DAD Instruction (Double ADD)

```
DAD B
DAD D
DAD H

where B stands for BC
      D stands for DE
      H stands for HL
```

- DAD adds the contents of the specified register pair to the contents of HL register pair.
- Affects carry flag.

Extended Register Instructions

- INX & DCX

```
INX B
INX D
INX H

where B stands for BC
      D stands for DE
      H stands for HL
```

- DCX also works same as INX.

Indirect Instructions/Addressing

- HL acts as *data pointer*
- Indirect Read
  - MOV reg, M , **where $M=M_{HL}$**
  - HL=3020H, M_{HL}=34H
  - Now, MOV A, M
  - A=34H

Indirect Instructions/Addressing

- Indirect Write
  - MOV M, reg , **where $M=M_{HL}$**
  - HL=3020H, M_{HL}=34H, A=22H
  - Now, MOV M,A
  - M_{3020H}=22H

Why these are called indirect addressing?

Indirect-Immediate Instructions

- MVI M, byte , **where $M=M_{HL}$**
- HL=3020H, M_{HL}=34H
- Now, MVI M, ACH
- M_{3020H}=ACH

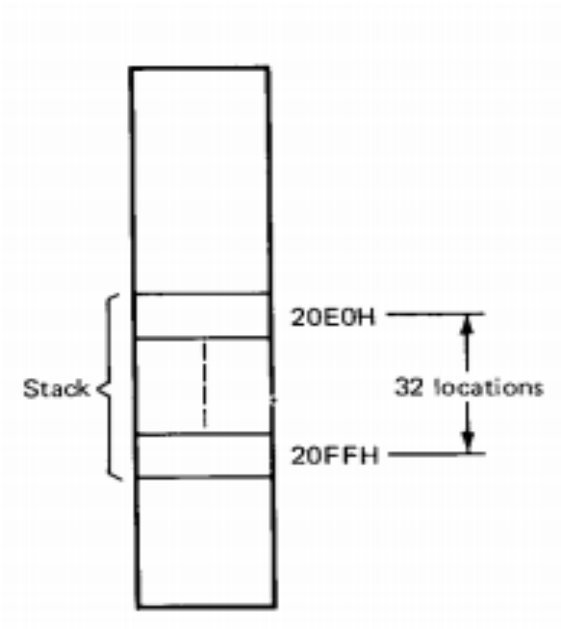
- ADD M
- ADC M
- SUB M
- SBB M
- INR M
- DCR M
- ANA M
- ORA M
- XRA M
- CMP M

Example

- Suppose 256 bytes of data are stored in memory addresses 2000H and 20FFH. Show a program that will copy all these 256 bytes at memory location 3000H and 30FFH.

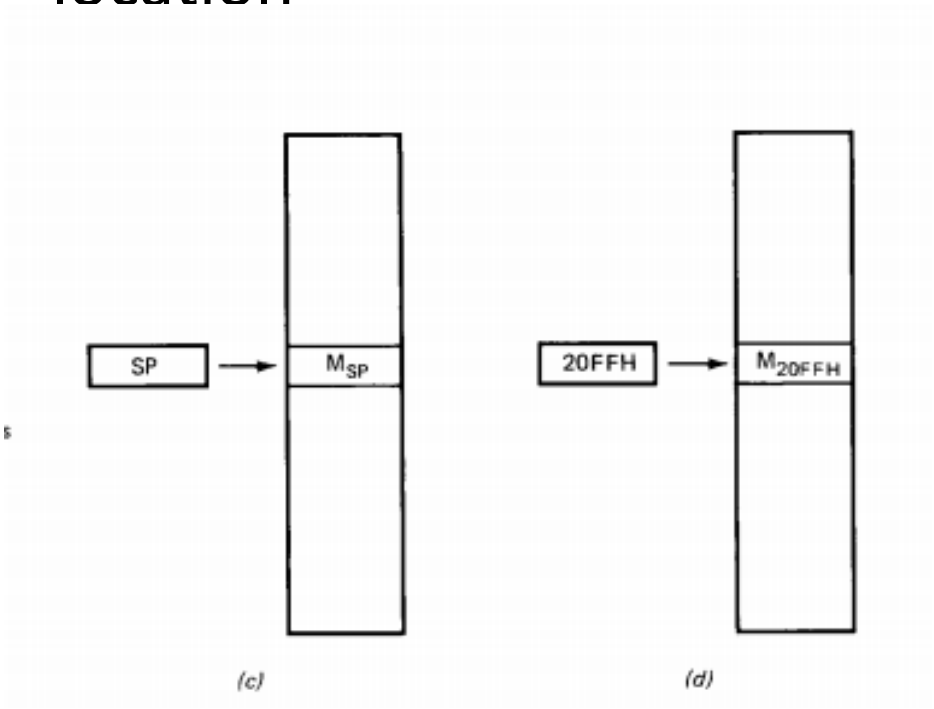
Stack Instructions

- The stack
  - Programmer decide where to locate the stack and how big it will be.
  - They can locate the stack anywhere they want
  - After setting up the stack, they no longer use that portion of memory for program or data



Stack Instructuions

- Stack Pointer
  - 16 bit register SP holds the desired memory location



To load the SP,
LXI SP, dble

LXI SP, 20FFH

Stack Instruction

- PUSH

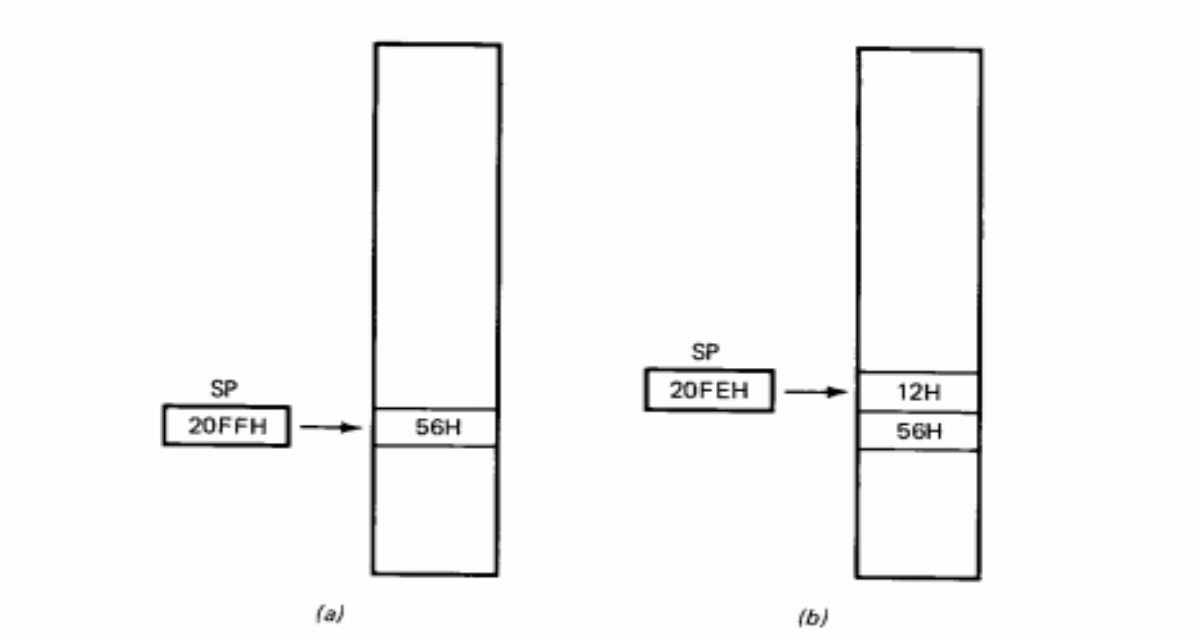
```
PUSH B
PUSH D
PUSH H
PUSH PSW

where B stands for BC
      D stands for DE
      H stands for HL
    PSW stands for program status word
```

  - PSW means program status word
  - PSW=AF
  - A=contents of Accumulator
  - F=contents of flag register

PUSH

- SP is decremented by 1
- High byte in the specified register is stored at M_{SP-1}
- SP is again decremented
- Low byte in the specified register is stored at memory location M_{SP-2}
- Suppose, B=5612H, SP=2100H



POP instructions

- The lower byte is read from the memory location pointer by SP and stored in the lower half of the register pair.
- SP is incremented.
- The higher byte is read and stored in the upper half of the register pair.
- SP is again incremented.