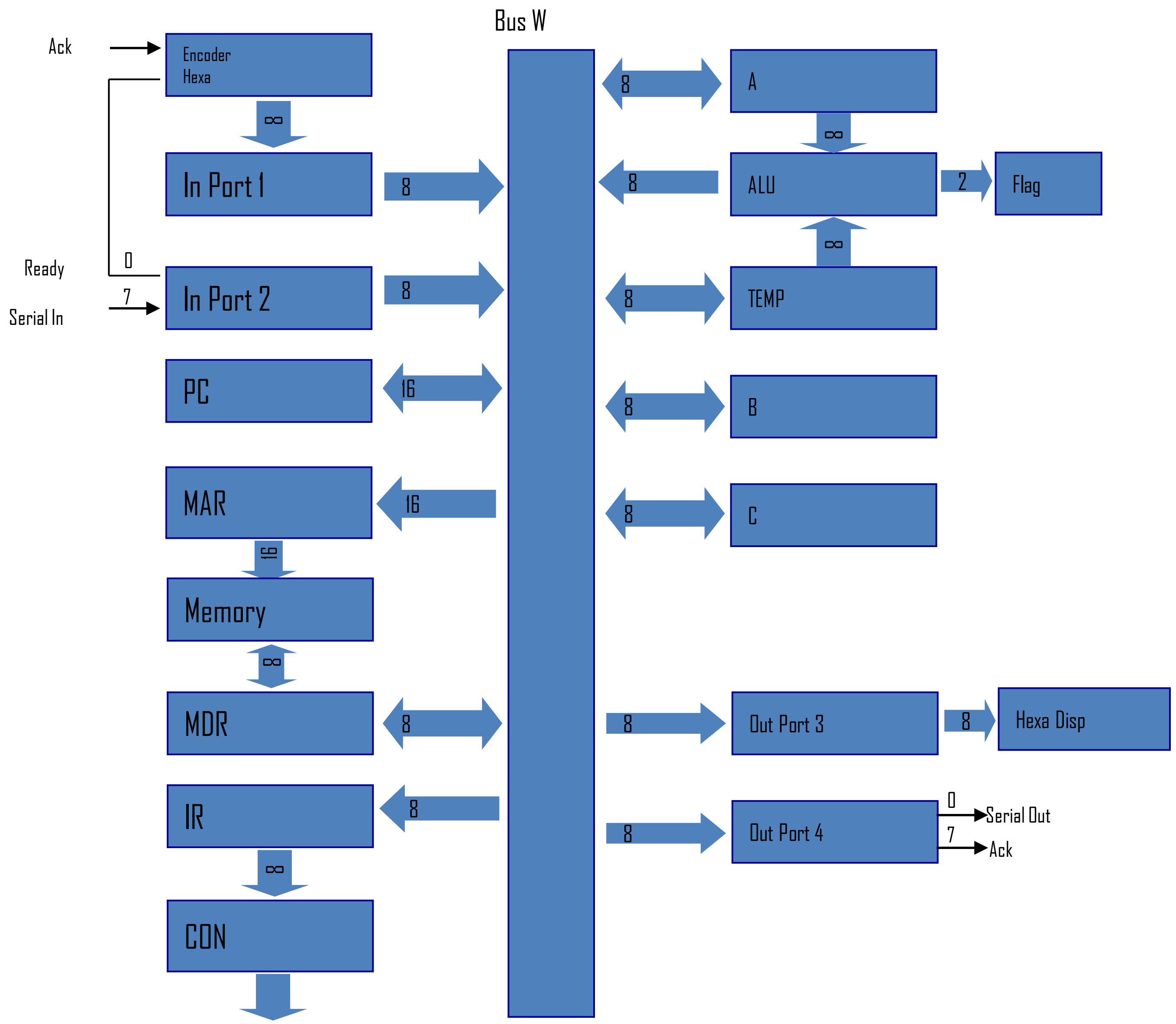
SAP-3

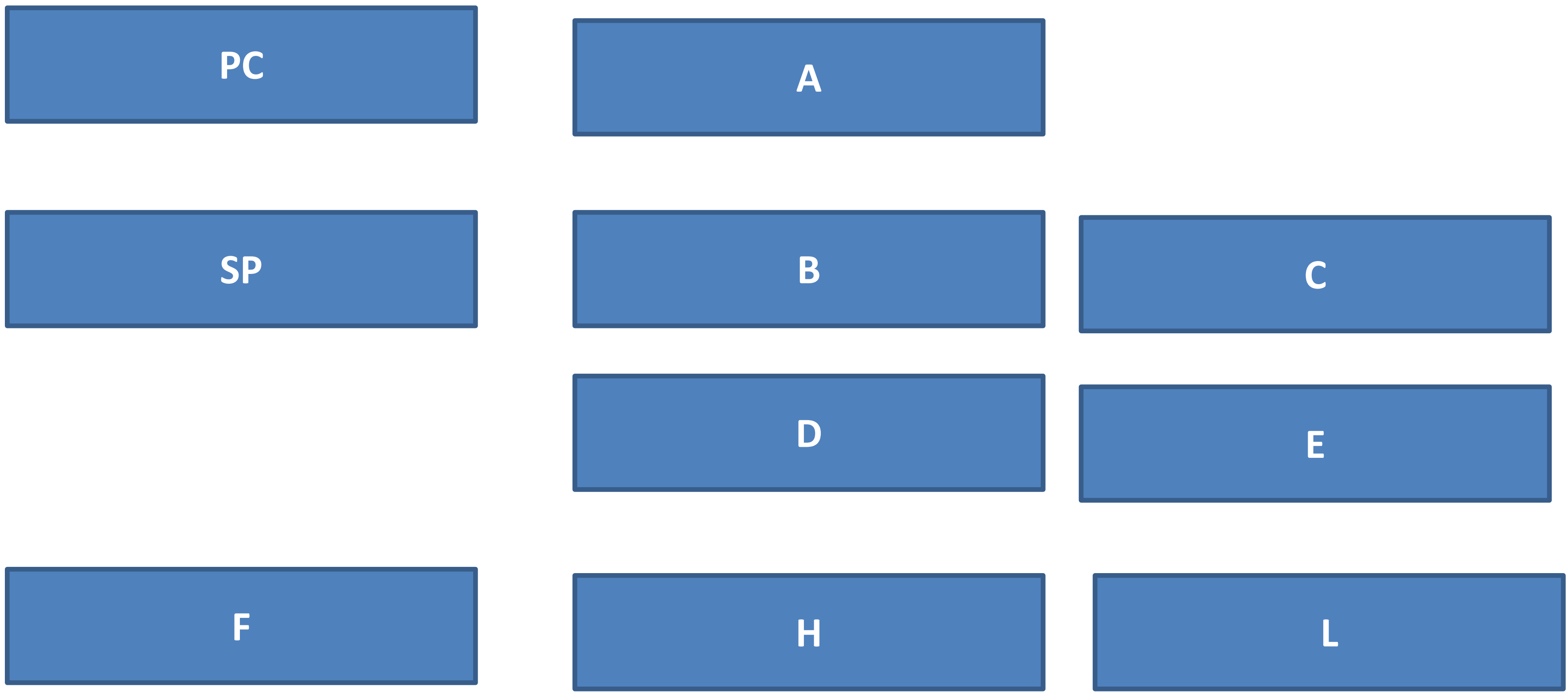
Introduction

- SAP-3 is an 8-bit microcomputer that is upward compatible with 8085.
- SAP3 includes all SAP2 features.
- It includes **stack** operations.

SAP2 - Architecture



Architecture (Like as SAP2)



Program Counter

16 bit address

Thus can count from

PC= 0000 0000 0000 0000

PC= 1111 1111 1111 1111(FFFFH)

Memory Data Register

- 8-bit Register
- Output setup RAM
- Receives data from the bus before write operation
- Data to the bus after read operation

Instruction Register

- 8-bit op code
- Can accommodate 256 instruction
- Around 246 instructions

Controller Sequencer

As usual → Like SAP2

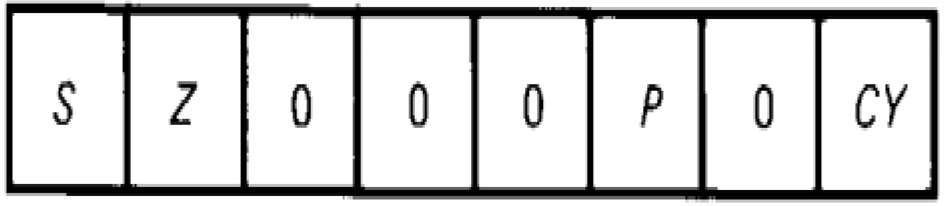
Accumulator

Same as SAP-1

ALU and Flags

- ALU: Includes both arithmetic and logical operation
- 4 or more control bits for determining the operation to be performed

- Flag: Represent the status of the arithmetic and logical operation
- 8-bit Register Use
    - Zero Flag(Z)
    - Sign Flag(S)
    - Carry Flag(CY)
    - Parity Flag (P)



Temp

- **Temporary register (TEMP) → like SAP2**

Microprocessor Instructions

MOV & MVI

MOV reg1, reg2

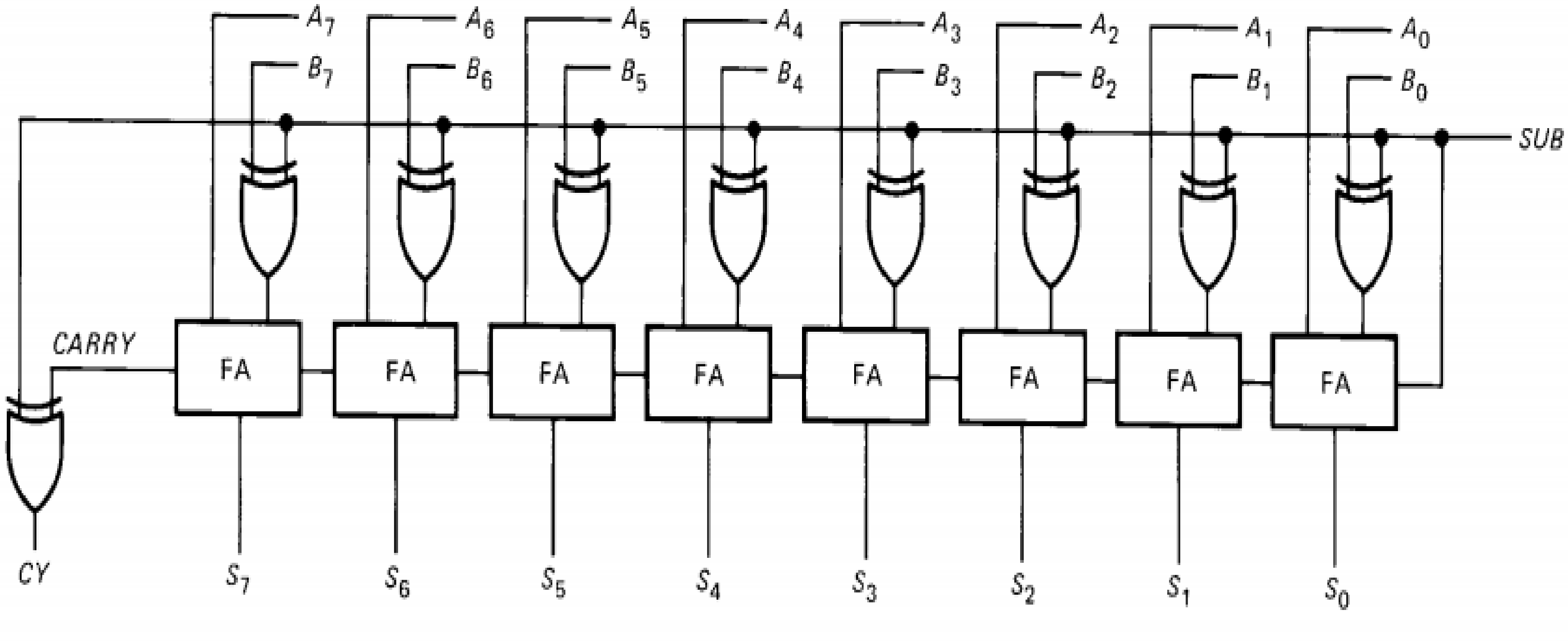
reg1 = A, B, C, D, E, H, L

reg2 = A, B, C, D, E, H, L

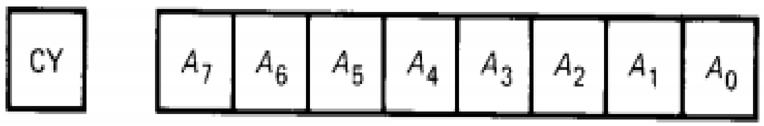
MVI reg, byte

reg = A, B, C, D, E, H, L

Carry Flag (CY)



(a)



Carry Flag (CY) Contd.

$$CY = \begin{cases} CARRY & \text{for ADD instructions} \\ \overline{CARRY} & \text{for SUB instructions} \end{cases}$$

1.Add → Carry

2.Sub → Borrow

Carry Flag Instructions

- STC → **S**e**TC**arry (CY =1)
- CMC → **C**omple**M**ent**C**arry
    - CY = $\overline{CY}$

ADD

ADD reg

reg = A, B, C, D, E, H, L

ADC

ADC → ADD with Carry (CY).

ADC reg

reg = A, B, C, D, E, H, L

Example: A = 1111 1111 E = 0000 0000 CY =1.

```
ADC E ➔         1111 1111
              + 0000 0001
------------------------------------------------
               10000 0000
```

→ At the End, CY = 1. A = 0000 0000

SUB

SUB reg

reg = A, B, C, D, E, H, L

SBB

SBB → Subtract with borrow (CY).

SUB reg

reg = A, B, C, D, E, H, L

Example: A = 1111 1111 E = 0000 0010 CY =1.

```
SBB E →             1111 1111
                  - 0000 0011
------------------------------------------
                    1111 1100
```

Increment

INC reg

reg = A, B, C, D, E, H, L

- INC has no effect on the carry.

Decrement

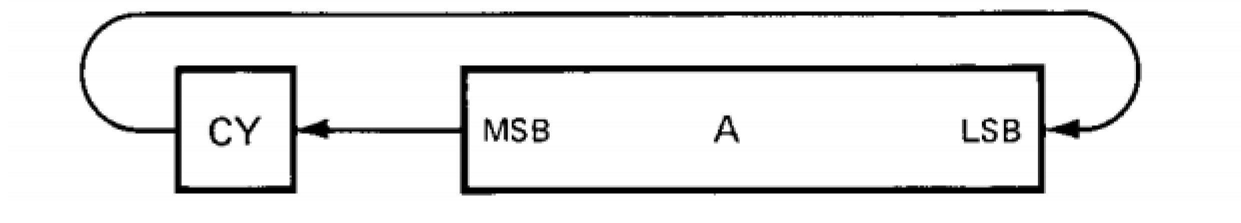
DEC reg

reg = A, B, C, D, E, H, L

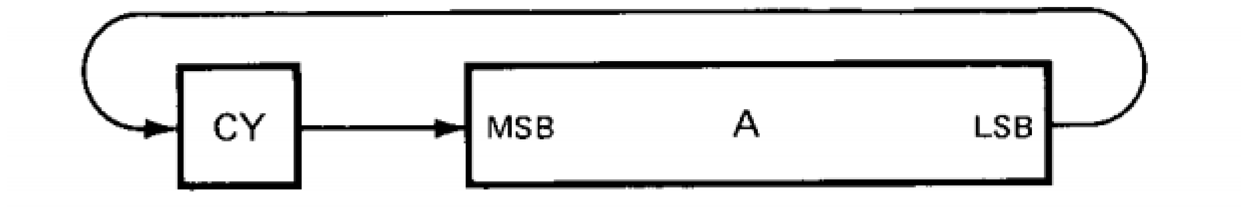
- DEC has no effect on the carry.

RAL & RAR

- RAL → Rotate All Left (a)
- RAR → Rotate All Right (b)



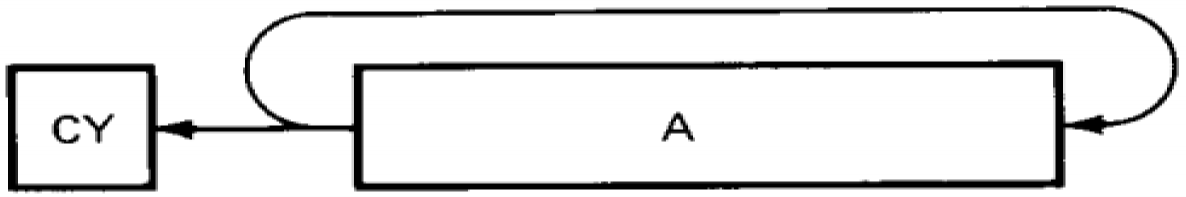
(a)



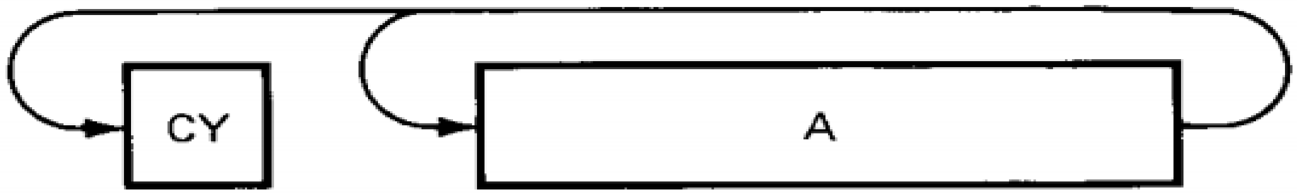
(b)

RLC & RRC

- RLC → Rotate Left with Carry (a)
- RRC → Rotate Right with Carry (b)
- Multiplication & Division



(a)



(b)

Logic Instructions

ANA reg

ORA reg

XRA reg

reg = A, B, C, D, E, H, L

Compare Instruction

CMP reg

reg = A, B, C, D, E, H, L

- Z flag effected after the operation

Arithmetic & Logic Immediate

ANI byte
ORI byte
XRI byte

ADI byte → ADD immediate
ACI byte → ADD with carry immediate
SUI byte → SUB immediate
SBI byte → SUB with borrow immediate
CPI byte → Compare Immediate

Parity Flag (P)

- P = 1, if A has EVEN number of ‘1’
- P = 0, otherwise

Jump And Call Instruction

SAP2

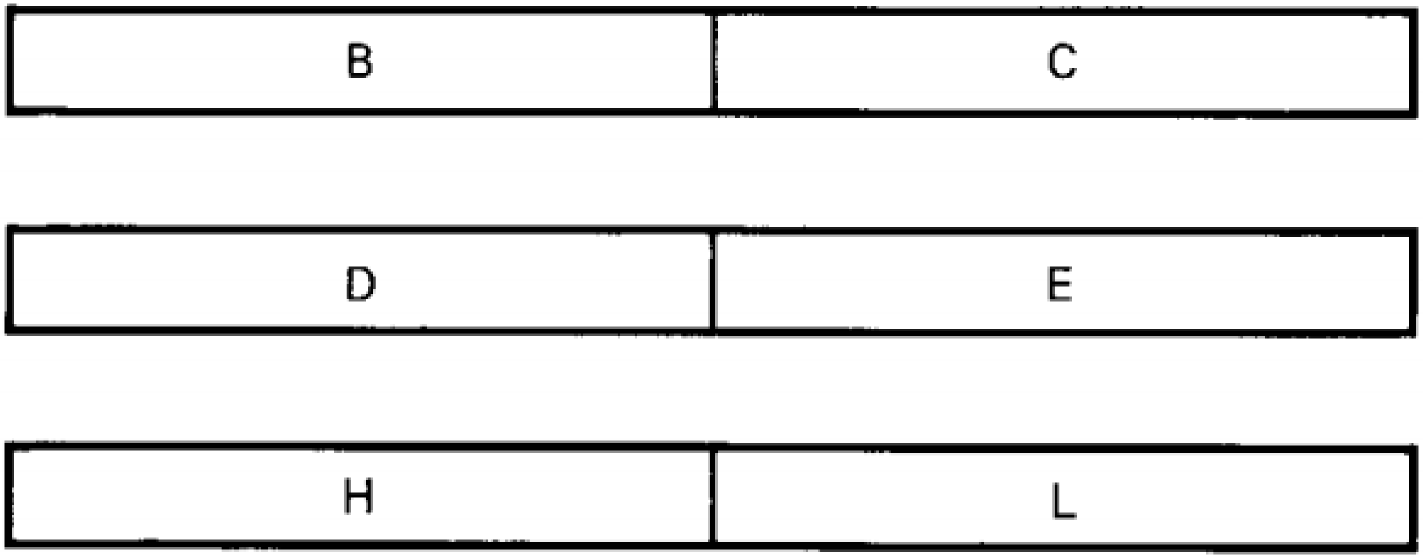
JMP address
JM (Jump if Minus)
JZ(Jump if zero)
JNZ(Jump if not zero)

SAP3

- JP (Jump if positive)
- JC (Jump id Carry)
- JNC (Jump if not Carry)
- JPE (Jump if Even Parity)
- JPO (Jump if Odd Parity)

Extended Register

- Register pairs → 16 bits
- 3 pairs (BC, DE and HL)



Extended Instructions

- X → for Extended instruction

LXI B, dble

LXI D, dble

LXI H, dble

Example: LXI B, 90FFH

B = 90H, C = FFH

Extended Instructions

- DAD instruction → Double ADD

DAD B

DAD D

DAD H

Here, default register Pair is **HL**, works like A for extended ADD.

Extended Instructions

- INX & DCX instruction

INX B, INX D, INX H

DCX B, DCX D, DCX H

- No Flags will be effected after these operations.

Indirect Instructions

- The HL register pair points to the memory locations where data is stored, means HL is a ***data pointer***.
- **LDA address and STA address**
- HL = address
- Use HL pair to access memory location

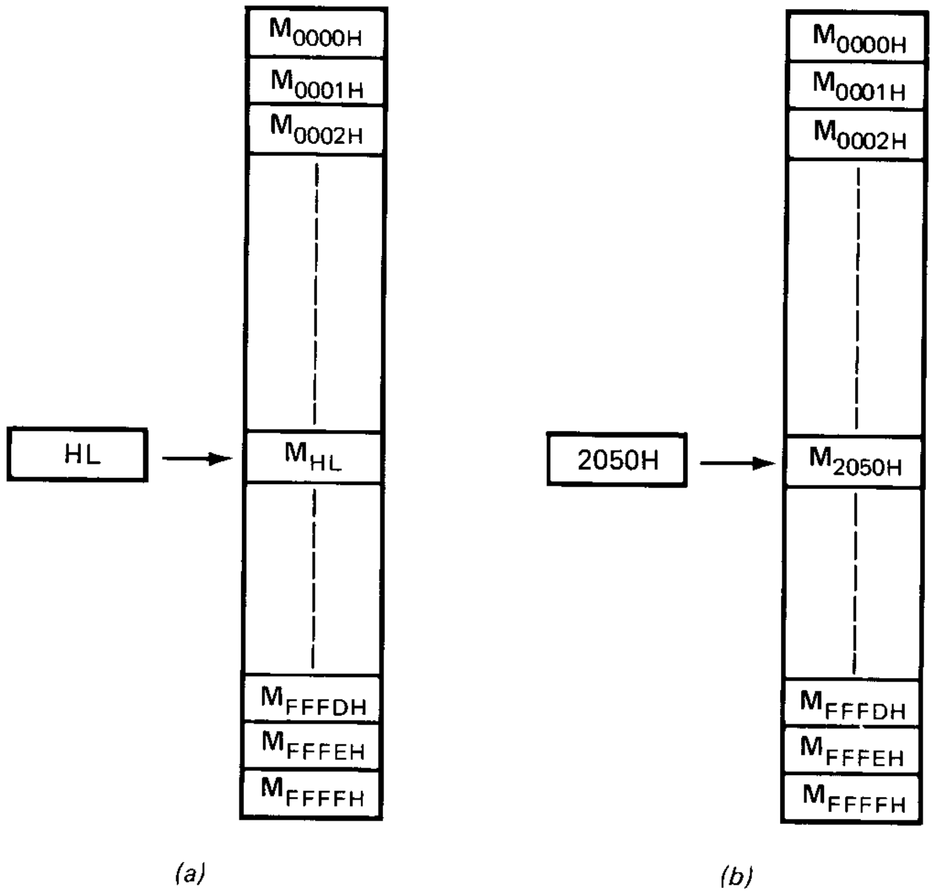
Indirect Read

MOV reg, M

reg = A, B, C, D, E, H, L

M = memory address

HL ← M

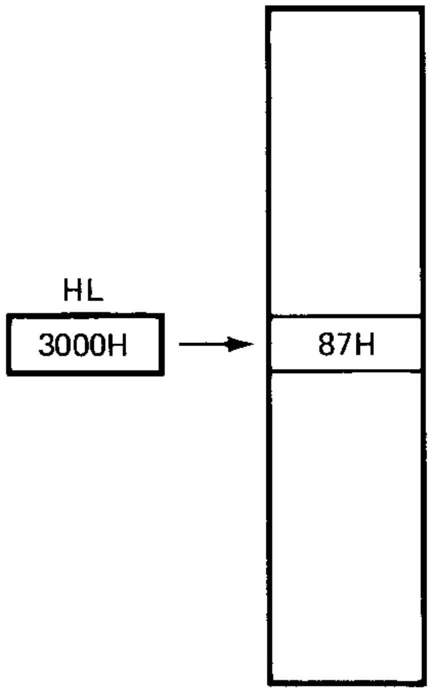


Indirect Read

MOV C , M

HL = 3000H

C = 87H



Indirect Write

MOV M, reg

reg = A, B, C, D, E, H, L

M = memory address

HL $\leftarrow$ M

Indirect Immediate Instructions

MVI M, byte

M = memory address

HL $\leftarrow$ M

Other Instructions by HL pointer

- ADD M
- ADC M
- SUB M
- SBB M
- INC M
- DEC M
- ANA M
- ORA M
- XRA M
- CMP M

STACK Instructions

- Begin Address: 20FFh
- End Address: 20E0h

- PUSH, POP
- Before call we need to store registers & Flags

PUSH B (BC)
PUSH D (DE)
PUSH H (HL)
PUSH PSW

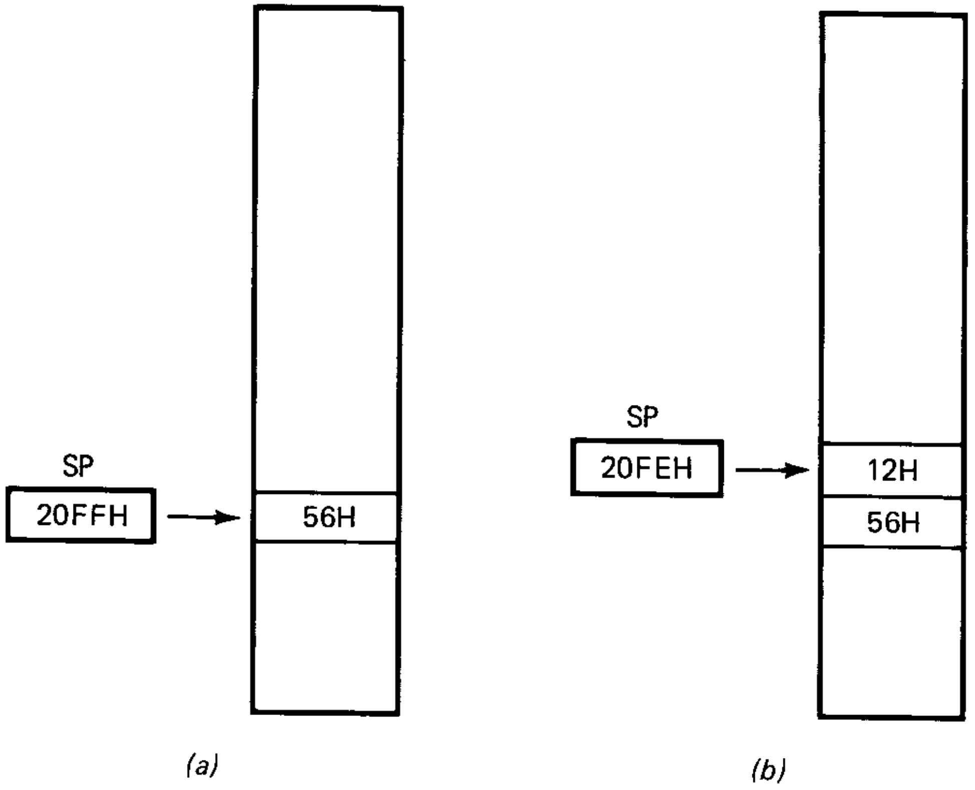
PUSH operation

- When PUSH instruction is executed, the following things happen:
  - The SP is decremented to get a new value of SP-1
  - The high byte in the specified register pair is stored in M[SP-1]
  - The SP is decremented again to get SP-2
  - The low byte in the specified register pair is stored in M[SP-2]

PUSH Operation
Example

BC = 5612 H

SP = 2100 H ; PUSH B



POP Operation

- When POP is executed, the following happens:
  - REVESE of PUSH !!

Call & RET Instructions

CALL

Subroutine ????

Call is used to call the subroutine

Ret

Return back from subroutine

Program Counter contents and other values (FLAGS) ????

-----stored in the STACK

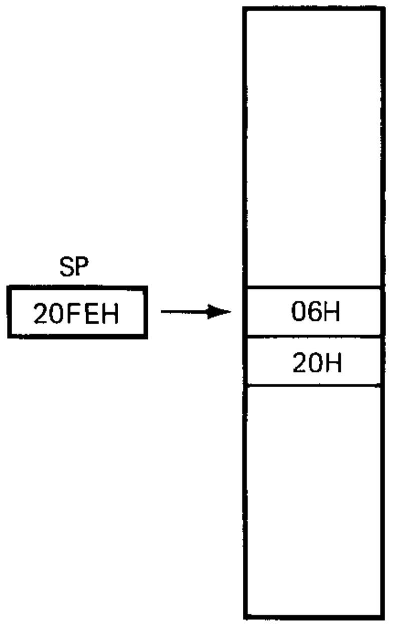
CALL Execution Example

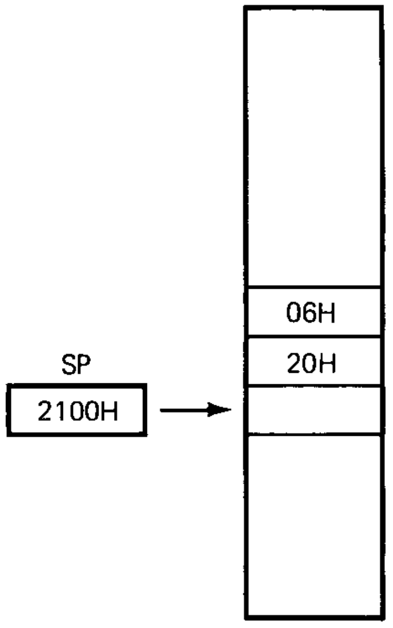
```
Address     Instruction

2000H       LXI SP,2100H
2001H
2002H

Address     Instruction

2003H       CALL 8050H
2004H
2005H
2006H       MVI A,0EH
  .             .
  .             .
  .             .
20FFH       HLT
  .
  .
  .
8050H           .
  .             .
  .             .
8059H       RET
```





Conditional CALLs

- CNZ address
- CZ address
- CNC address
- CC address
- CPO address
- CPE address
- CP address
- CM address

Conditional RETURNs

- RNZ
- RZ
- RNC
- RC
- RPO
- RPE
- RP
- RM

Acknowledgement

Md. Iftekharul Islam Sakib

Lecturer

CSE, BUET

Thank you