Logical Instruction(affect flags)

- **ANDing (performs bitwise ANDing)**
  - Accumulator(A) is an implicit operand in this instruction and result is always stored in A
  - Flags: CY=0, AC=1, other flags as per result
  - Instruction: **ANA** reg/M; A <- A and reg/M

    (**1 Byte Instruction**)

    e.g. ANA C; A <- A and C

    ANA L; A <- A and L

    ANA M; A <- A and [HL]

  - Instruction: **ANI** 8-bit value; A <- A and value

    (**2 Bytes Instruction**)

    e.g. ANI 8DH; A <- A and 8DH

  (Note: AND operation is used to reset(make 0) and mask the individual bit without affecting other bits)

Logical Instruction(affect flags)

- **ANDing (performs bitwise ANDing)**

  Write a program in 8085 to reset bit D3 of an accumulator without affecting other bits.

  ANI F7H

  HLT

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	1	1	1
D7	D6	D5	D4	0	D2	D1	D0

Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Write a program in 8085 to mask bit D3 and D7 of an accumulator.

ANI 88H

HLT

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	0
D7	0	0	0	D3	0	0	0

Logical Instruction(affect flags)

- **ORing (performs bitwise ORing)**
  - Accumulator(A) is an implicit operand in this instruction and result is always stored in A
  - Flags: CY=0, AC=0, other flags as per result
  - Instruction: **ORA** reg/M; A <- A or reg/M

    (**1 Byte Instruction**)

    e.g. ORA D; A <- A or D

    ORA E; A <- A or E

    ORA M; A <- A or [HL]

  - Instruction: **ORI** 8-bit value; A <- A or value

    (**2 Bytes Instruction**)

    e.g. ORI 1FH; A <- A or 1FH

  (Note: OR operation is used to set(make 1) the individual bit without affecting other bits)

Logical Instruction(affect flags)

- **ORing (performs bitwise ORing)**

  Write a program in 8085 to set bit D5 of an accumulator without affecting other bits.

  ORI 20H

  HLT

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	0	0	0
D7	D6	1	D4	D3	D2	D1	D0

Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Logical Instruction(affect flags)

- **XORing (performs bitwise XORing)**
  - Accumulator(A) is an implicit operand in this instruction and result is always stored in A
  - Flags: CY=0, AC=0, other flags as per result
  - Instruction: **XRA** reg/M; A <- A xor reg/M

    (**1 Byte Instruction**)

    e.g. XRA D; A <- A xor D

    XRA E; A <- A xor E

    XRA M; A <- A xor [HL]

  - Instruction: **XRI** 8-bit value; A <- A xor value

    (**2 Bytes Instruction**)

    e.g. XRI 1FH; A <- A xor 1FH

  (Note: XOR operation is used to complement the individual bit without affecting other bits)

Logical Instruction(affect flags)

- **XORing (performs bitwise XORing)**

  Write a program in 8085 to complement(toggle) bit D1 of an accumulator without affecting other bits.

  XRI 02H

  HLT

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0
D7	D6	D5	D4	D3	D2	D1’	D0

Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Logical Instruction(affect flags)

- **Complement Accumulator**
  - No flags are affected
  - Instruction: **CMA**; A <- A'

    (**1 Byte Instruction**)

    e.g. CMA; A <- A'

Write a program in 8085 to read the value from memory 8080H. Complement the value and store the result at memory 8099H.

```
LDA 8080H;   A <- [8080H]
CMA
STA 8099H;   [8099H] <- A
HLT
```

Logical Instruction(affect flags)

Write a program in 8085 to read the value from memory 6600H. Reset the bit D6 and set bit D2 and store the result at memory 7700H.

```
LDA 6600H
ANI BFH
ORI 04H
STA 7700H
HLT
```

	D7	D6	D5	D4	D3	D2	D1	D0
ANDing	1	0	1	1	1	1	1	1
	D7	0	D5	D4	D3	D2	D1	D0
ORing	0	0	0	0	0	1	0	0
	D7	0	D5	D4	D3	1	D1	D0

Write a program in 8085 to read the value from memory 9000H. Complement bit D5 and then add 49H and store the result at memory 9005H.

```
LDA 9000H
XRI 20H
ADI 49H;   A <- A+49H
STA 9005H ;  [9005H] <- A
HLT
```

	D7	D6	D5	D4	D3	D2	D1	D0
XORing	0	0	1	0	0	0	0	0
	D7	D6	D5’	D4	D3	D2	D1	D0
ADD	0	1	0	0	1	0	0	1

Logical Instruction(affect flags)

- **PSW (Program Status Word)**
  - It is the combination of two registers Accumulator and Flag in respective order

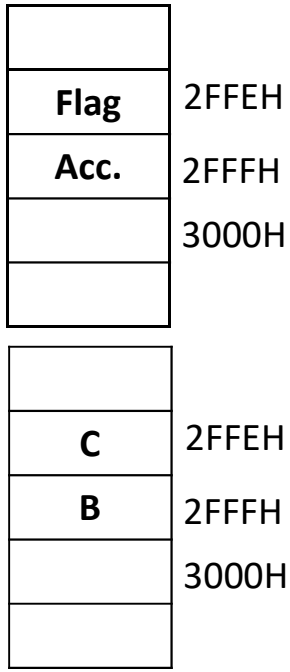
(Note: ***There are only two instructions PUSH and POP which operate on PSW)***

PSW

A	F

Write a program in 8085 to set sign and carry flag and reset parity flag without affecting other flags

```
LXI SP, 3000H
PUSH PSW
POP B;   C <- Flag,  B <- Acc.
MOV A, C
ORI 81H
ANI FBH
MOV C, A
PUSH B
POP PSW; F <- C, A <- B
HLT
```



	S	Z	X	AC	X	P	X	CY
ORing	1	0	0	0	0	0	0	1
	1	Z	X	AC	X	P	X	1
ANDing	1	1	1	1	1	0	1	1
	1	Z	X	AC	X	0	X	1