Q. How can we interchange the

$$\boxed{\begin{array}{cc} 5 & 3 \\ 3 & 5 \end{array}}$$

→ rotate it 4 times.

```
MOV  CL, 4
ROL  BL, CL
```

not with carry

/ Machine

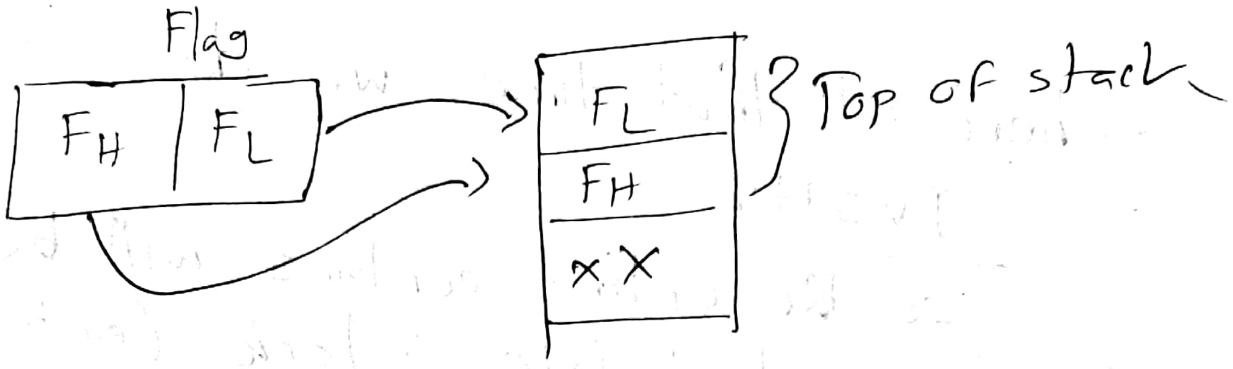
$\boxed{b}$ Processor control loops

via these instruction we can directly operate on flags.

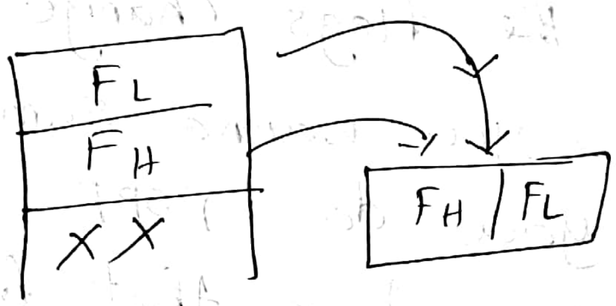
1) Push F → push
2) Pop F
3) LAHF
4) SAHF

When we do arithmetic/logical instructions like : ADD BL, DL . by product they are getting effected.

/ Push F



/ POP F



/ LAHF = Load AH from Flag

AH ← Flag (F_L)

lower byte of flag register loaded on AH.

SAHF = Store AH back into flag.

AH → Flag

Use : for some reason, during computation you want ~~to~~ the same flags back after 20/30 instructions, (you cannot assume after 20 instructions flags will contain same) flag will change after ~~at~~ every arithmetic & logical instructions.

→ here, at that time we do

PUSHF

so, the entire contains will be pushed into stack. (entire info is saved in stack)

now, let the flags change 100 times.

whenever, you wanna same contains back, you do POPF.

POPF will restore that back from stack into the flag register.

So, we can restore that value to the previous value.

For whole flag reg.

For lower byte of flag : we do

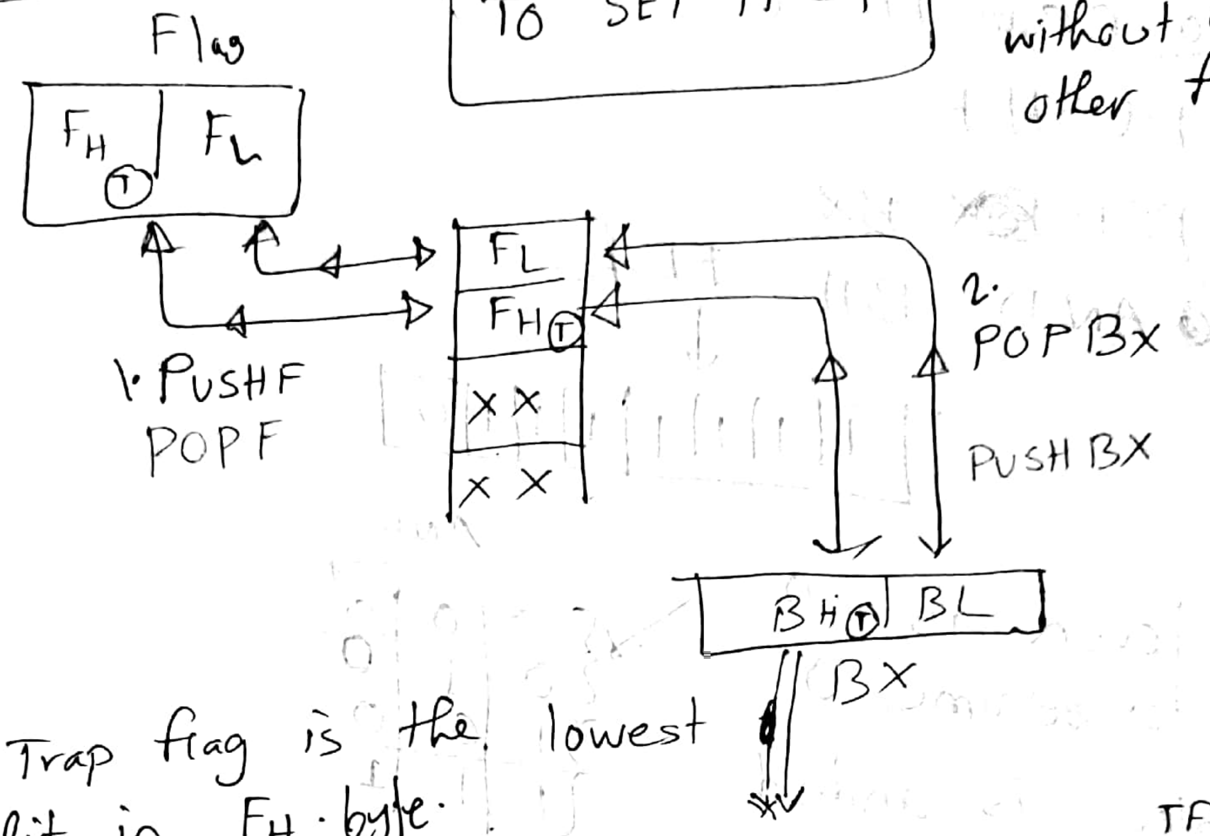
LAHF & SAHF

Scanned with CamScanner

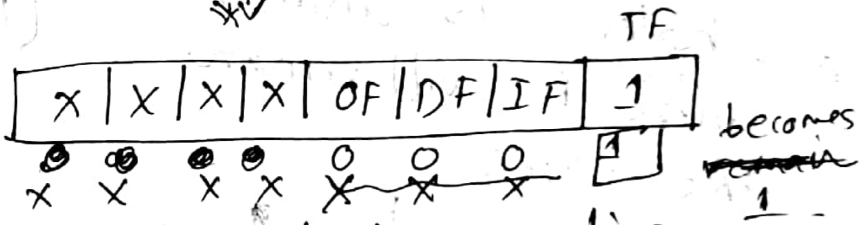
TF

TO SET TF = 1

without effecting other flags.



// Trap flag is the lowest bit in F_H byte.



// now, we want to SET/1 to TF, we need a logic operation we need to do OR operation. since

// anything OR with 0, will remain same

	OR	
0	0 →	0
0	1 →	1
1	0 →	1
1	1 →	1

// anything OR with 1, becomes 1.

OR BH, 01H

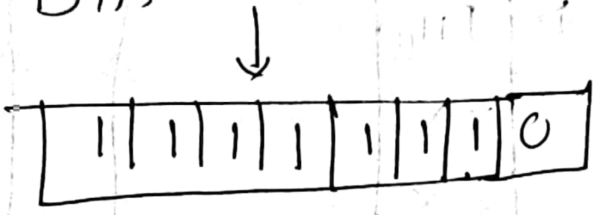
0	0	0	0	0	0	0	1

TF

Then, PUSH BX
POPF

To clear TF = 0

```
PUSH F
POP ~~BX~~ BX
AND BH, FEh
```

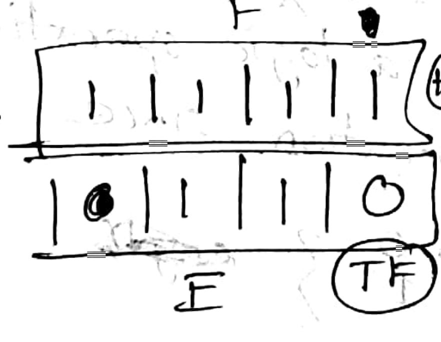


"whatever is 0 has to become 0,

"whatever is anded with 1, will remain the same.

AND		y
0	0	0
0	1	0
1	0	0
1	1	1

→ so,



(H) ANDED will remain same.

(L) ANDED with 0, has to become 0,

```
PUSH BX
POP F
```

pe 2)

Processor Control / Machine Control Instructions

(these are instructions that directly operate on Flag Reg)

[redacted]

For Carry Flag

1) **STC**
This instruction **sets** the **Carry Flag**. No Other Flags are affected. CF=1

2) **CLC**
This instruction **clears** the **Carry Flag**. No Other Flags are affected. CF=0

3) **CMC**
This instruction **complements** the **Carry Flag**. No Other Flags are affected. CF = ~~1~~0
CF = ~~0~~1

For Direction Flag

4) **STD**
This instruction **sets** the **Direction Flag**. No Other Flags are affected.

5) **CLD**
This instruction **clears** the **Direction Flag**. No Other Flags are affected.

For Interrupt Enable Flag

6) **STI**
This instruction **sets** the **Interrupt Enable Flag**. No Other Flags are affected.

7) **CLI**
This instruction **clears** the **Interrupt Enable Flag**. No Other Flags are affected.

Note: There is no direct way to alter TF. It can be altered through program as follows:

To set TF:

```
PUSHF           ; push contents of Flag register into the stack
POP BX          ; pop contents of flag reg from the stack-top into BX
OR BH, 01H      ; set the bit corresponding to TF, in the BH register
PUSH BX         ; push the modified BX register into the stack
POPF            ; pop the modified contents into flag register.
```

To reset TF:

```
PUSHF           ; push contents of Flag register into the stack
POP BX          ; pop contents of flag reg from the stack-top into BX
AND BH, 0FEH    ; reset the bit corresponding to TF, in the BH register
PUSH BX         ; push the modified BX register into the stack
POPF            ; pop the modified contents into flag register.
```