STACK, I/O AND MACHINE CONTROL GROUP

Special Note:
In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

STACK INSTRUCTIONS

1) PUSH Rp

It pushes the given register pair into the stack.

Eg: PUSH B ; SP ← SP – 1
[SP] ← B
SP ← SP - 1
[SP] ← C

Addr. Mode	Flags Affected	Cycles	T-States
Register	**None**	3	**12**

Special Notes:

All Stack operations (Push and Pop) are compulsorily **16-bit operations**.
We can push register pairs, and not individual 8-bit registers.

Push and Pop only support **Register Addressing Mode**. Hence if we want to push a number like 2000H, we must first move it into a register pair and then push it.

Eg: LXI B, 2000H
Push B

2) POP Rp

It pops the top 2 elements (2 × 8-bit ∴ 16-bit) from the stack into the given register pair.
The lower byte comes out first as the higher byte was pushed in first and stack operates in LIFO manner.

Eg: POP B ; C ← [SP]
SP← SP + 1
B ← [SP]
SP← SP + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	**None**	3	**10**

3) PUSH PSW

It pushes the PSW (Program Status Word) into the stack.
The PSW is the combination of the accumulator and the Flag register, accumulator being the higher byte.
∴ **PSW ➔ AF**
PSW can **ONLY** be used in PUSH and POP instructions.

Eg: PUSH PSW ; SP ← SP – 1
[SP] ← A
SP ← SP - 1
[SP] ← F

Addr. Mode	Flags Affected	Cycles	T-States
Register	**None**	3	**12**

4) POP PSW

It pops the top 2 elements (2 × 8-bit ∴ 16-bit) from the stack into the PSW.

Eg: POP PSW ; F ← [SP]
SP← SP + 1
A ← [SP]
SP← SP + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	**None**	3	**10**

Please Note: There are other instructions like XTHL, SPHL, LXI SP, CALL RET etc which directly or indirectly affect the stack and are already included in various groups above.

Special Note:
PSW is only created to allow push and pop of Accumulator and Flags.
Hence PSW can only be used in Push and Pop instructions.

I/O INSTRUCTIONS

5) IN 8-bit I/O Port address

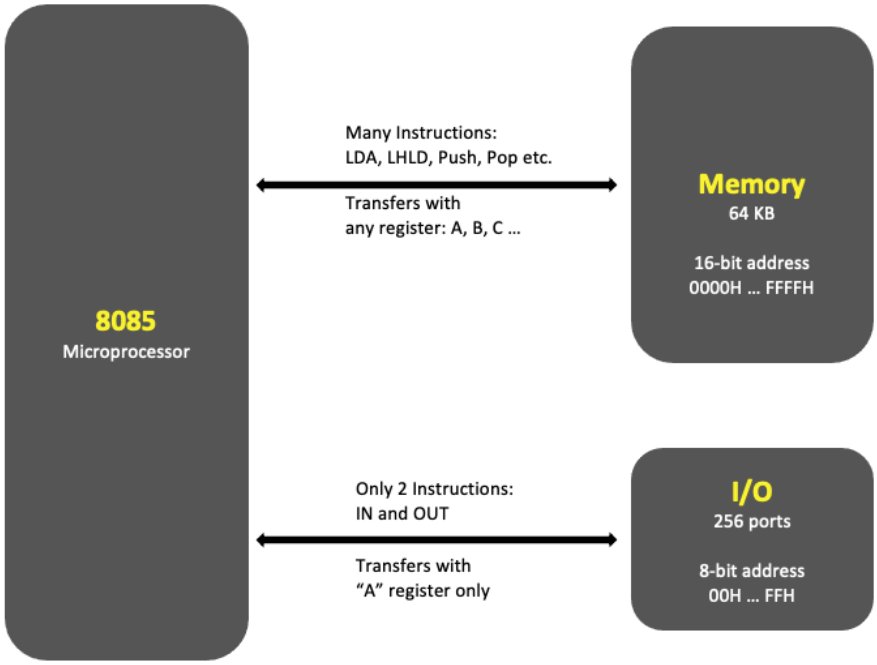
8085 has 256 I/O Ports having 8-bit addresses 00H … FFH.
This instruction is used to read data from an I/O Port, whose address is given in the instruction.
This data can be read into the Accuulator ONLY.
Eg: IN 80 ; A ← $[80]_{I/O}$

Addr. Mode	Flags Affected	Cycles	T-States
Direct	**None**	3	**10**

6) OUT 8-bit I/O Port address

This instruction is used to send data from the accumulator to an I/O Port, whose address is in the instruction.
This data can be sent from the Accuulator ONLY.
Eg: OUT 80; $[80]_{I/O}$ ← A

Addr. Mode	Flags Affected	Cycles	T-States
Direct	**None**	3	**10**



MACHINE CONTROL INSTRUCTIONS

7) SIM (Set Interrupt Mask)

This instruction is used to set the interrupt masking pattern for the µP.
The appropriate bit pattern is **loaded into** the accumulator an then this instruction is executed.
It is basically used to **mask/unmask** the **interrupts** except TRAP and INTR.
It can also be used to send a '**bit**' out through the serial out pin **SID**.
Eg: SIM ; µP accepts the masking pattern through the accumulator.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1	**4**

8) RIM (Read Interrupt Mask)

This instruction is used to read the interrupt masking pattern for the µP.
After executing this instruction, the µP loads the bit pattern **into** the accumulator.
It can also be used to receive a '**bit**' out through the serial in pin **SID**.
Eg: RIM ; µP loads the masking pattern into the accumulator.

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1	**4**

Please Note: For the bit patterns of SIM and RIM instruction, please refer the chapter on Interrupts.

9) EI (Enable Interrupts)

This instruction is used to enable the interrupts in the µP.
This instrusetion sets the INTE flip-flop (Interrupt Enable Flip-Flop).
This instruction effects all the interrupts except TRAP.
Eg: EI ; $INTE_{F/F} \leftarrow 1$

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1	**4**

10) DI (Disable Interrupts)

This instruction is used to disable the interrupts in the µP.
This instrusetion resets the INTE flip-flop.
This instruction effects all the interrupts except TRAP.
Eg: DI ; $INTE_{F/F} \leftarrow 0$

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1	**4**

11) NOP (No Operation)

This instruction performs no operation, but consumes time of the μP.
It is the simplest method of causing a software delay.
Eg: NOP ; ----------

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1	**4**

12) HLT (Halt)

This instruction signifies the end of the program.
It causes the μP to stop fetching any further instruction, hence program execution is stopped.
It makes the Halt Flip Flop inside the μP = 1.
μP checks the Halt Flip Flop in the beginning (1st T-State) of the next Machine Cycle and Stops all operations.

Eg: HLT ; Halt Flip-Flop ← 1

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**None**	1 + 1T	**5**