23/11/2021

MICROPROCESSOR

8085 Stack operation

→ The stack is a portion of read/write memory set aside by the user for the purpose of storing information temporarily.

→ When the information is written on stack the operation is called PUSH.

→ When the information is read from stack the operation is called POP.

→ The stack operates in FIRST IN LAST OUT (FILO) fashion. This means that the first information pushed on to the stack is the last information popped off from the stack.

→ IN 8085, stack is implemented using STACK POINTER register. SP is a 16-bit register which gives the address of memory where the information is to be stored or to be read.

→ The memory location currently pointed by stack pointer is called top of stack.

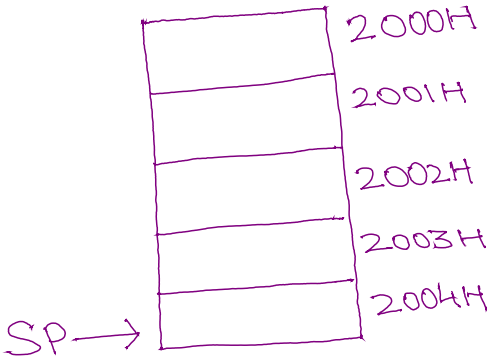
→ IN 8085, there are 2 instructions that work with stack.

① PUSH rp ② POP rp

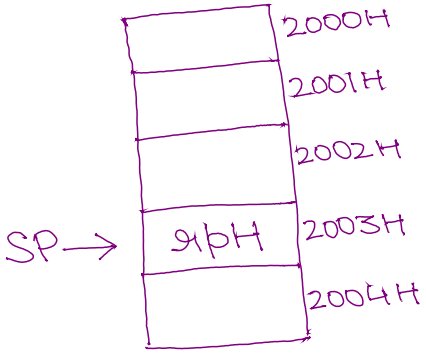
rp → means 16-bit register pair (BC, DE, HL)

PUSH rp : This instruction decrements stack pointer by one and copies the higher byte of the register pair mentioned in the instruction into the memory location pointed by stack pointer. It then decrements the stack pointer again by one and copies the lower byte of the register pair into memory location pointed by stack pointer.

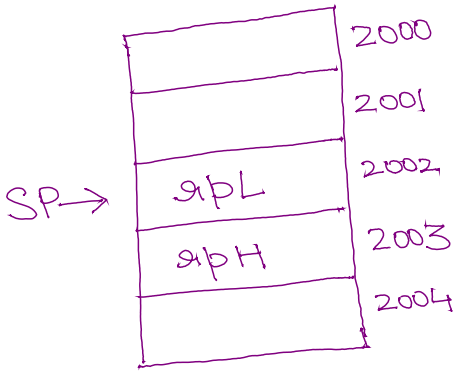
Operation : SP ← SP − 1 ; (SP) ← rpH ; SP ← SP − 1 ; (SP) ← rpL



(a) Initial position



(b) Decrements SP by 1 and store lower byte of rp



(c) Decrement SP by 1 and store higher byte of rp.

Example :

PUSH B ← Store BC rp onto stack

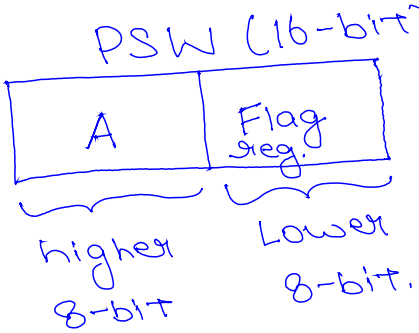
PUSH D ← Store DE rp onto stack

PUSH H ← Store HL rp onto stack

PUSH PSW ← Store PSW onto stack.

↳ Program Status Word.

PSW : Program Status Word is the combination of accumulator and Flag register.



POP rp : This instruction copies the contents of memory location pointed by the stack pointer into the lower byte of the specified register pair and increments the stack pointer by one. Then it copies the contents of memory location pointed by stack pointer into the higher byte of the specified register pair and increments the stack pointer again by one.

operation :

rpL ← (SP)

SP ← SP+1

rpH ← (SP)

SP ← SP+1

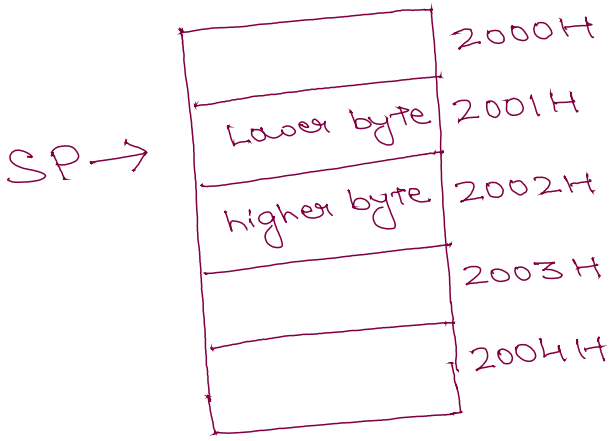
Examples :

POP B ← read into BC rp

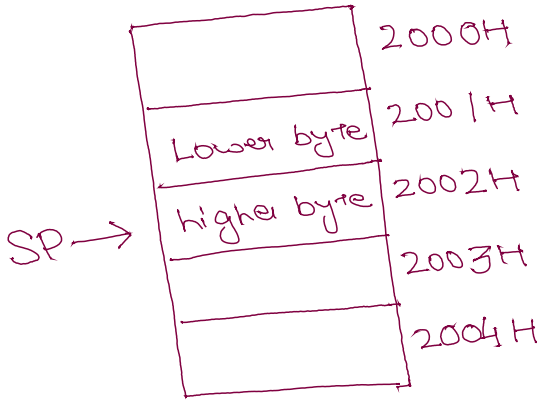
POP D ← read into DE rp

POP H ← read into HL rp

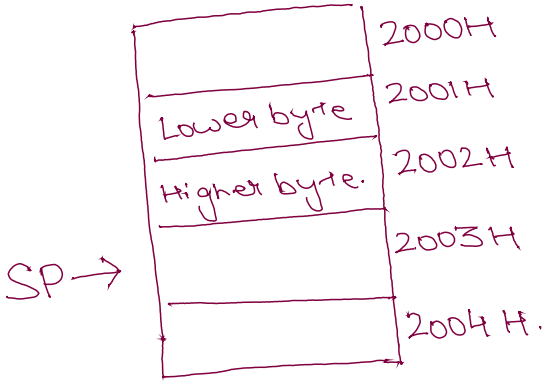
POP PSW ← read into PSW



(a) Initial position



(b) Read lower byte into rpL then increment SP by 1



(c) Read higher byte into rpH then increment SP by 1.

→ Program to swap the contents of BC and DE register pair using stack.

```
LXI SP, 2004H   ← SP is initialized.
LXI B, 4444H    } random values are
LXI D, 5555H.   } stored in BC, DE rp.
PUSH B  ← Contents of BC is stored on stack
PUSH D  ← Contents of DE is stored on stack
POP B   ← Contents of stack read to BC
POP D   ← Contents of stack read to DE
HLT.
```

23/11/2021

MICROPROCESSORS

SUBROUTINES IN 8085

→ A Subroutine is a group of instructions written seperately from the main program to perform a function that occurs repeatedly in main program.

→ 8085 µp has two instructions to implement subroutines :-

(1) CALL instruction ↓ Call subroutine from main program

(2) RET instruction ↓ return to main program from subroutine.

→ CALL instruction is used in the main program to call a subroutine

→ RET instruction is used at the end of the subroutine to return to the main program.

⟹ When a subroutine is called, the contents of Program counter (PC register) which is the address of the instruction following the CALL instruction, is stored on the stack by the µp automatically and the program execution is transferred to the subroutine address.

⇒ When the RET instruction is executed at the end of the subroutine, the memory address stored on the stack during CALL is retrieved back to the Program Counter (PC register) so that the sequence of execution is resumed in the main program.

⇒ Hence, to use subroutines, stack must be initialized in the beginning of main program always.
(using LXI SP, 16-bit address)

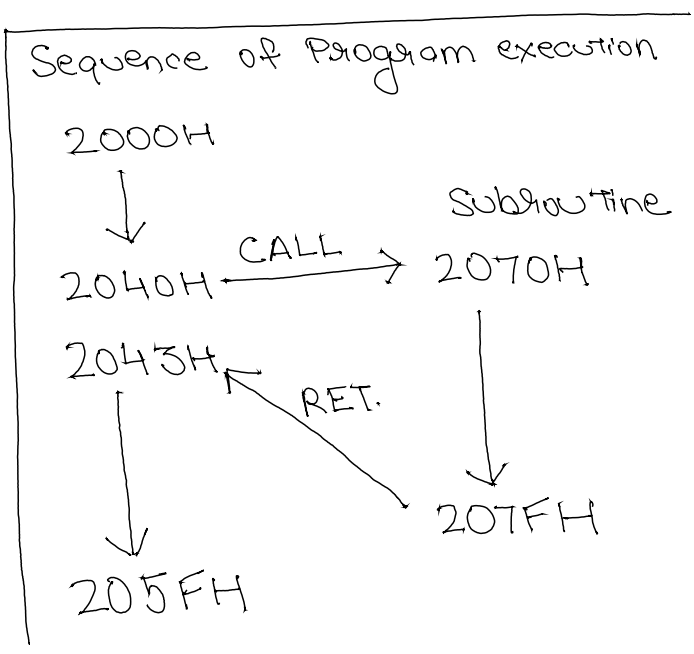
Example:

MAIN PROGRAM

2000H: LXI SP, 2400H ← Stack Initialize
=
=
=
2040H: CALL 2070H ← Call a subroutine which starts at address 2070H.
2043H: CMA
=
=
205FH: HLT.

SUBROUTINE

2070H: First instruction
=
=
=
207FH: RET



How µp will know where to come back after subroutine?.

→ Due to this, PC value was stored on stack after CALL before going to subroutine. Hence, when RET occurs in subroutine, value stored on the stack helps to come back to main program.

→ In the example :-

When CALL 2070H is executed the address of the next instruction CMA which is 2043H is stored on the stack. So in the subroutine when RET instruction is executed the address 2043H is retrieved back from stack to continue the execution of main program.

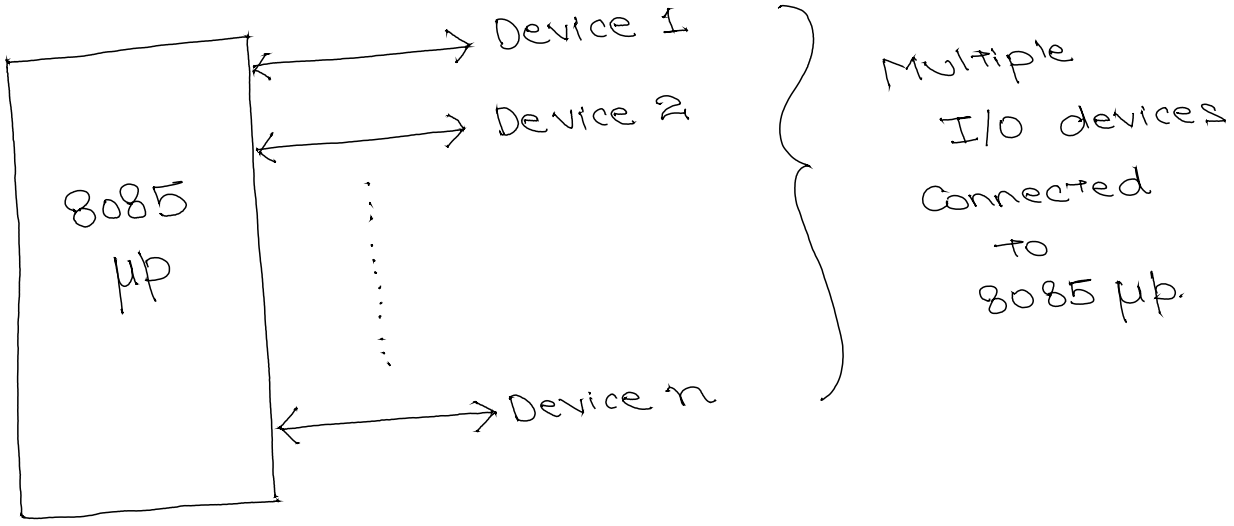
Syntax of CALL instruction :-

CALL 16-bit memory address of subroutine

Example : CALL 4000H.

23/11/2021 Microprocessors

8085 Interrupts



→ When one or more I/O devices are connected to a microprocessor system any one of them demand service or attention of μp at any time.

→ Microprocessor can observe and provide attention or service to multiple I/O devices in one of the two ways:-

① POLLING
↓
μp will check each device one by one for service.

- Disadvantage is μp is not able to do other processing

② INTERRUPTS
↓
Better options is μp will do some task and if any device require service it will inform μp

→ Interrupt is a process of data transfer whereby an external device can inform the μp that it is ready for communication and hence it requests attention.

→ The interrupt process allows the μp to respond to these external requests for attention on a demand basis and leaves the μp free to perform other tasks.

→ Interrupt request are classified in two categories :-

① MASKABLE INTERRUPT

↓

μp can ignore or delay a maskable interrupt request, if it is performing some critical task.

② NON-MASKABLE INTERRUPT.

↓

μp cannot ignore or delay non-maskable interrupt request. It has to be serviced immediately.

→ 8085 μp has 5 hardware interrupts

① TRAP ② RST 7.5 ③ RST 6.5

④ RST 5.5 ⑤ INTR

→ TRAP IS NON-MASKABLE INTERRUPT

→ The other 4 are MASKABLE INTERRUPTS.

Q How Interrupts are serviced by 8085 μp?

⇒ To service the request of an interrupt, a subroutine is written by the programmer. The subroutine written for an interrupt is known as INTERRUPT SERVICE ROUTINE (ISR).

Q IS CALL instruction is used for calling ISR?.

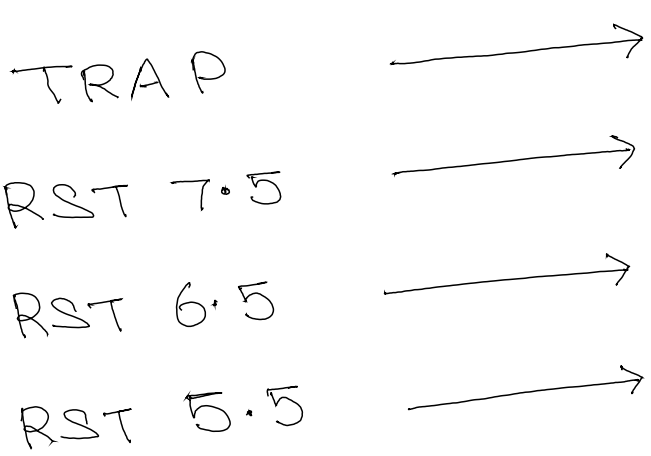
⇒ NO, ISR is automatically called by μp once it receives and acknowledges the interrupt request.

VECTORED INTERRUPTS

When an external device raises an interrupt request, μp has to execute ISR associated with the interrupt. If the internal circuit of the μp produces a CALL to a pre-determined memory location which is the starting address of ISR then that address is called vector address and such interrupts are called vectored interrupts.

8085 Vectored Interrupts

VECTOR ADDRESS



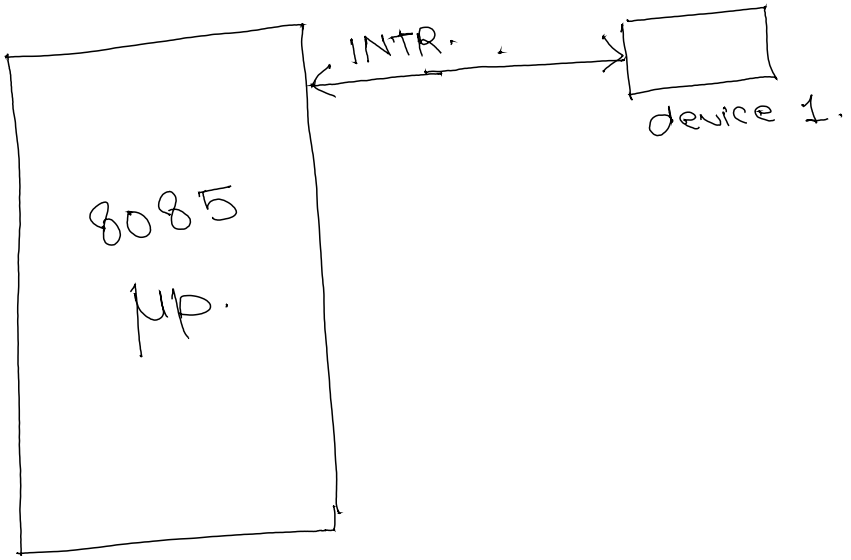
AUTOMATIC CALL LOCATIONS

EI : Enable Interrupt Instruction.

DI : Disable Interrupt Instruction.

↑ These two are machine control instruction which can enable/disable use of interrupts.

INTR → is a non-vectored interrupt.

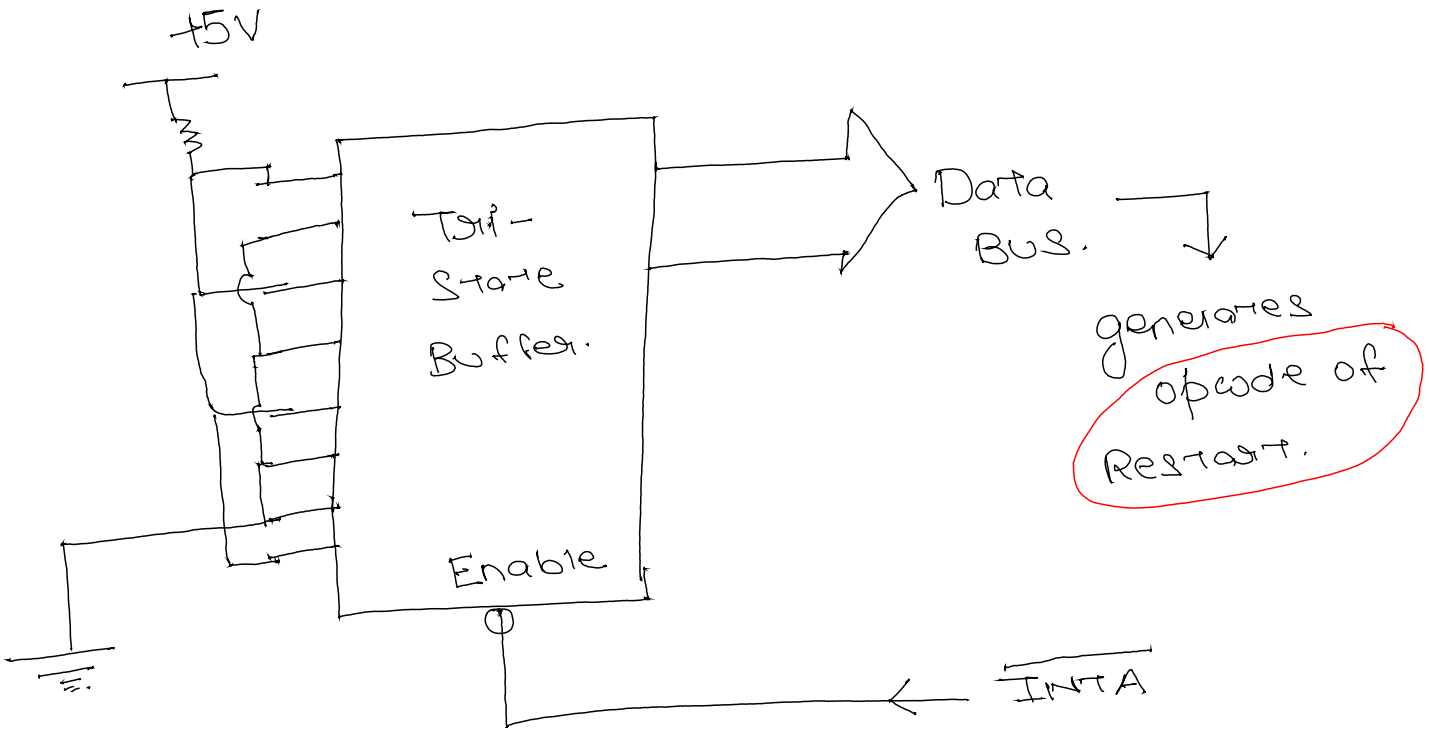


→ In general 8085 will perform some task (executing main program).

→ At some instant, intr signal becomes active. This means 8085 receives a interrupt request by device 1.

→ µp will ~~halt~~ pause the main program and execution will begin from memory location of ISR written for INTR interrupt.

→ After executing the ISR the µp will return to the main program.



Restart Instruction → kind of software interrupt.

RST 0 → take program execution to 0000H.

RST 1 → 0008H

RST 2 → 0010H

RST 3 → 0018H

RST 4 → 0020H

RST 5 → 0028H

RST 6 → 0030H

RST 7 → 0038H.