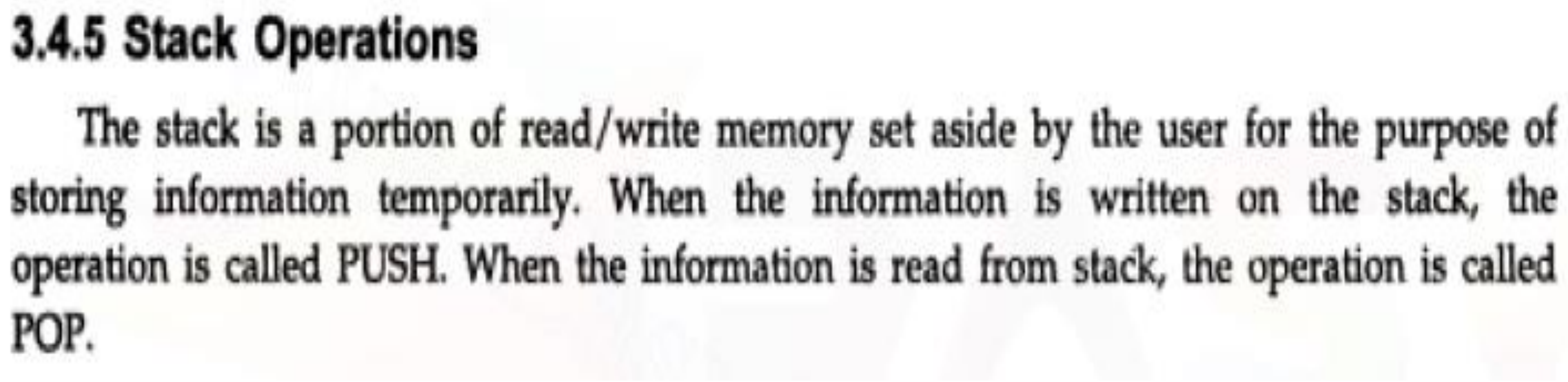
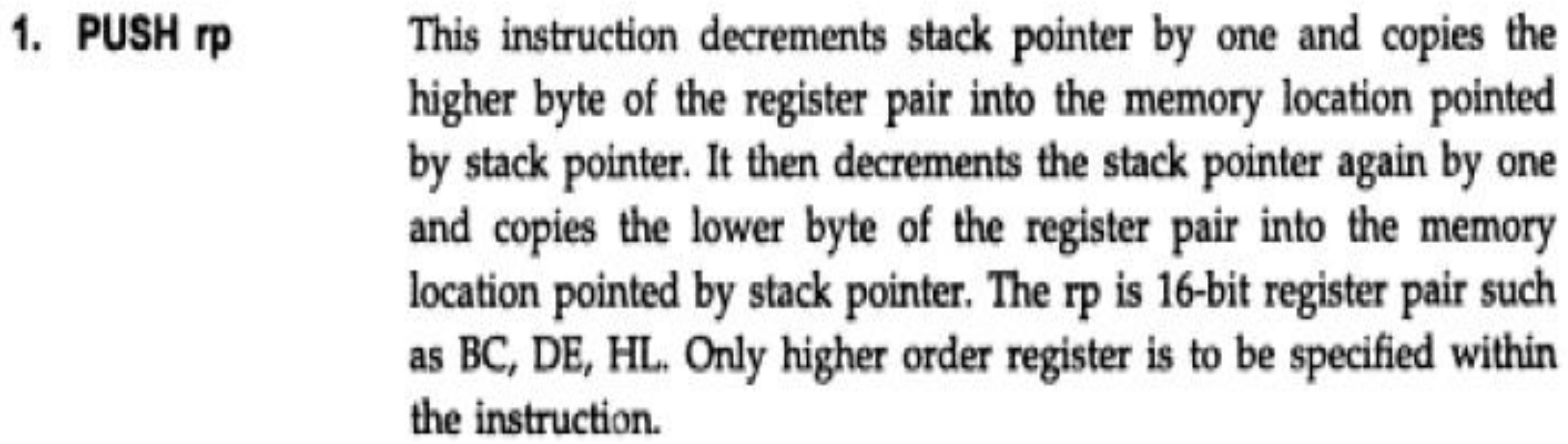
Microprocessor……..



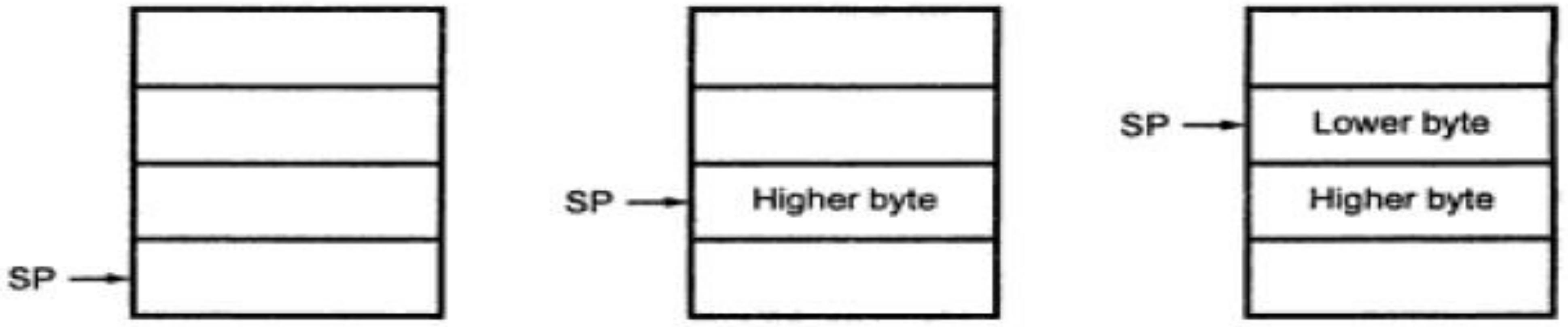
3.4.5 Stack Operations

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 the stack, the operation is called PUSH. When the information is read from stack, the operation is called POP.



1. **PUSH rp** This instruction decrements stack pointer by one and copies the higher byte of the register pair 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 the memory location pointed by stack pointer. The rp is 16-bit register pair such as BC, DE, HL. Only higher order register is to be specified within the instruction.

Operation : SP ← SP − 1, (SP) ← rpH, SP ← SP − 1, (SP) ← rpL.



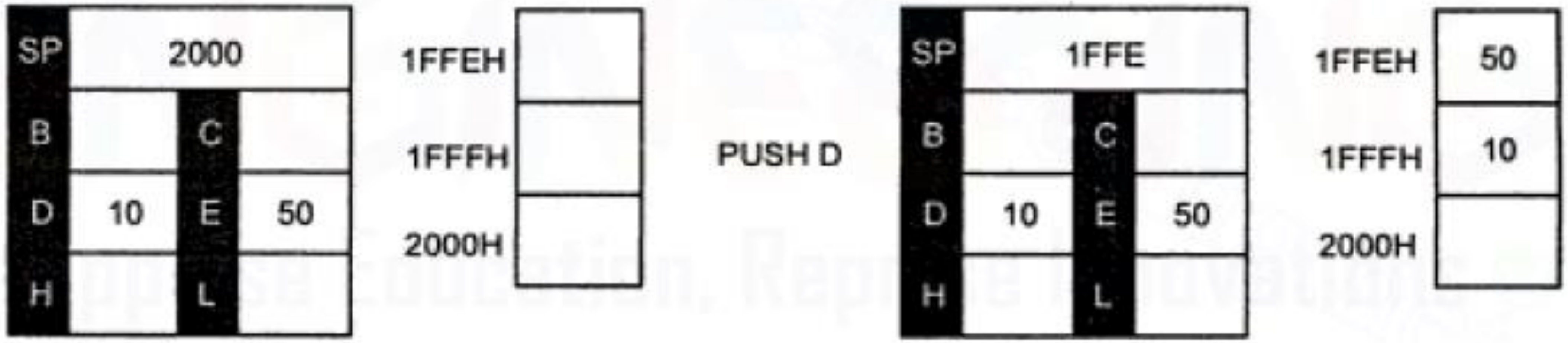
(a) Initial position **(b) Decrements SP & stores higher byte** **(c) Decrements SP & stores lower byte**

Fig. 3.5 Steps involved in PUSH Operation

Example : SP = 2000H, DE = 1050H.

PUSH D ; This instruction will decrement the stack pointer (2000H) by one (SP = 1FFFH) and copies the contents of D register (10H) into the memory location 1FFFH. It then decrements the stack pointer again by one (SP = 1FFEH) and copies the contents of E register (50H) into the memory location 1FFEH.

Microprocessor……..



2. PUSH PSW This instruction decrements stack pointer by one and copies the accumulator contents into the memory location pointed by stack pointer. It then decrements the stack pointer again by one and copies the flag register into the memory location pointed by the stack pointer.

Operation : $SP \leftarrow SP - 1$

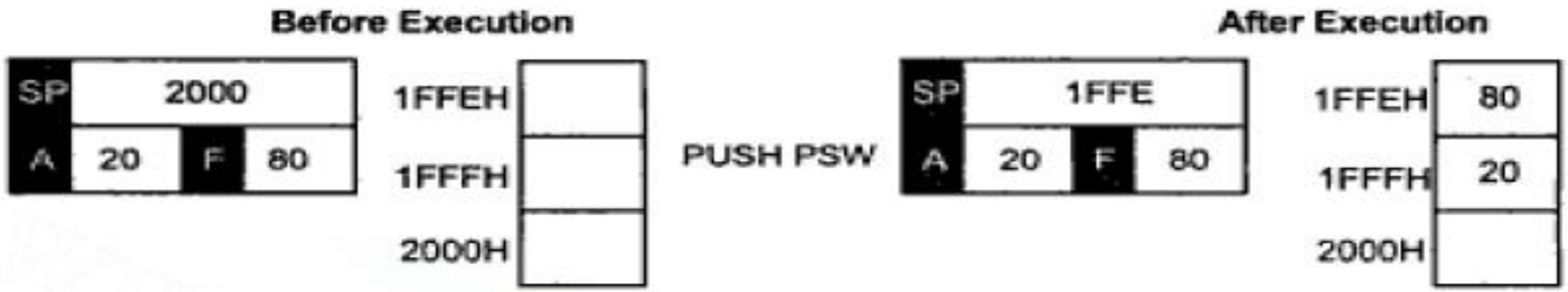
$(SP) \leftarrow A$

$SP \leftarrow SP - 1$

$(SP) \leftarrow$ Flag register

Example : SP = 2000H, A = 20H, Flag register = 80H

PUSH PSW ; This instruction decrements the stack pointer (SP = 2000H) by one (SP = 1FFFH) and copies the contents of the accumulator (20H) into the memory location 1FFFH. It then decrements the stack pointer again by one (SP = 1FFEH) and copies the contents of the flag register (80H) into the memory location 1FFEH.



3. POP rp

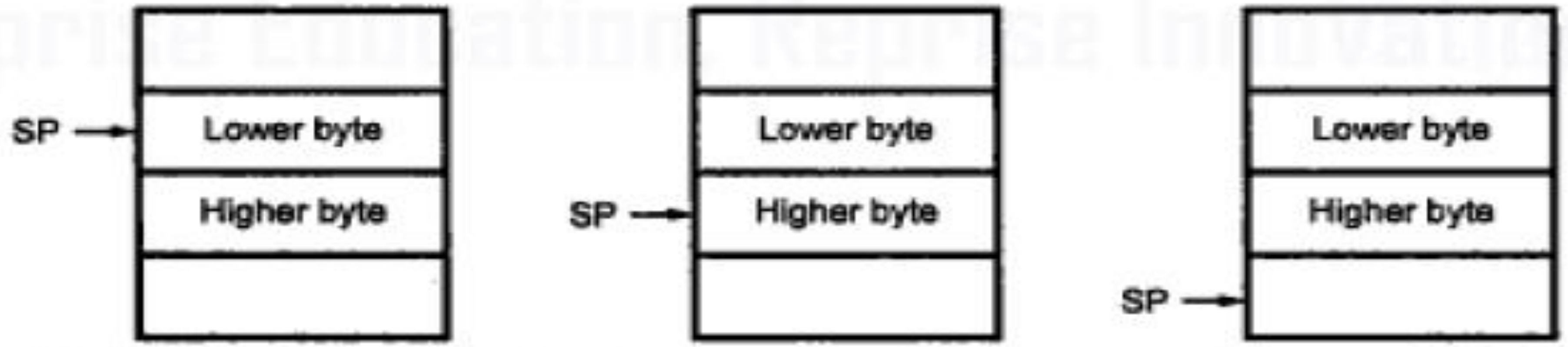
This instruction copies the contents of memory location pointed by the stack pointer into the lower byte of the specified register pair and increments the stack pointer by one. It then copies the contents of memory location pointed by stack pointer into the higher byte of the specified register pair and increments the stack pointer again by one. The rp is 16-bit register pair such as BC, DE, HL. Only higher order register is to be specified within the instruction.

Microprocessor……..

Operation : rpL ← (SP)

SP ← SP + 1

rpH ← (SP), SP ← SP + 1

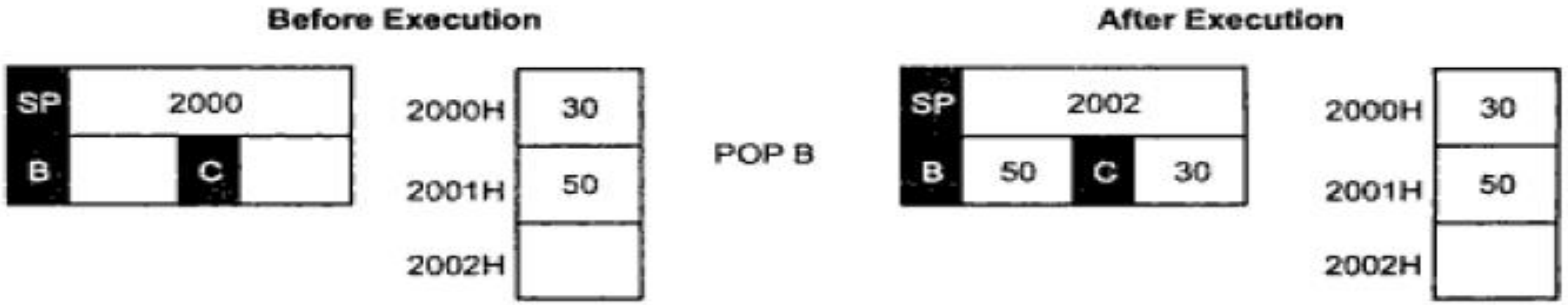


(a) Initial position, **(b) Increments SP and reads lower byte** **(c) Increments SP reads higher byte**

Fig. 3.6 Steps Involved in POP Operation

Example : SP = 2000H, (2000H) = 30H, (2001H) = 50H

POP B ; This instruction will copy the contents of memory location pointed by stack pointer, 2000H (i.e. data 30H) into the C register. It will then increment the stack pointer by one, 2001H and will copy the contents of memory location pointed by stack pointer, 2001H (i.e. data 50H) into B register, and increment the stack pointer again by one.



4. POP PSW

This instruction copies the contents of memory location pointed by the stack pointer into the flag register and increments the stack pointer by one. It then copies the contents of memory location pointed by stack pointer into the accumulator and increments the stack pointer again by one.

Operation : Flag register ← (SP)

SP ← SP + 1

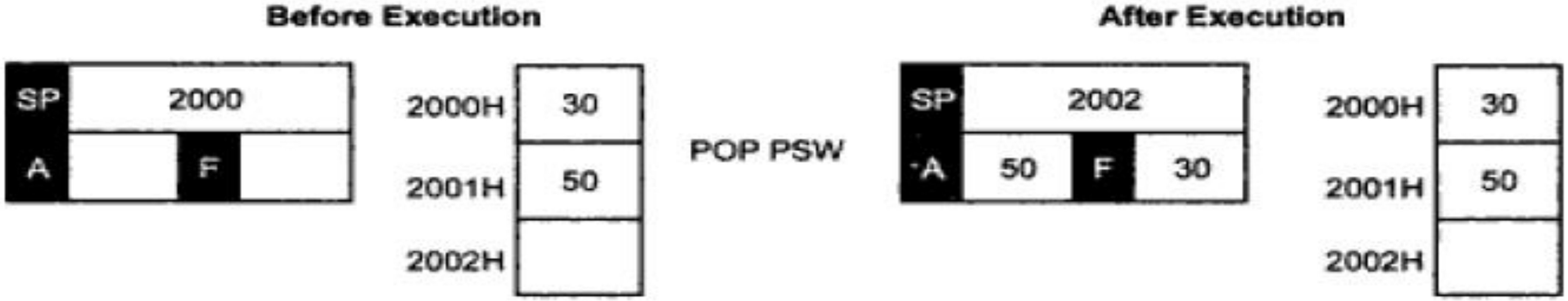
A ← (SP)

SP ← SP + 1

Microprocessor……..

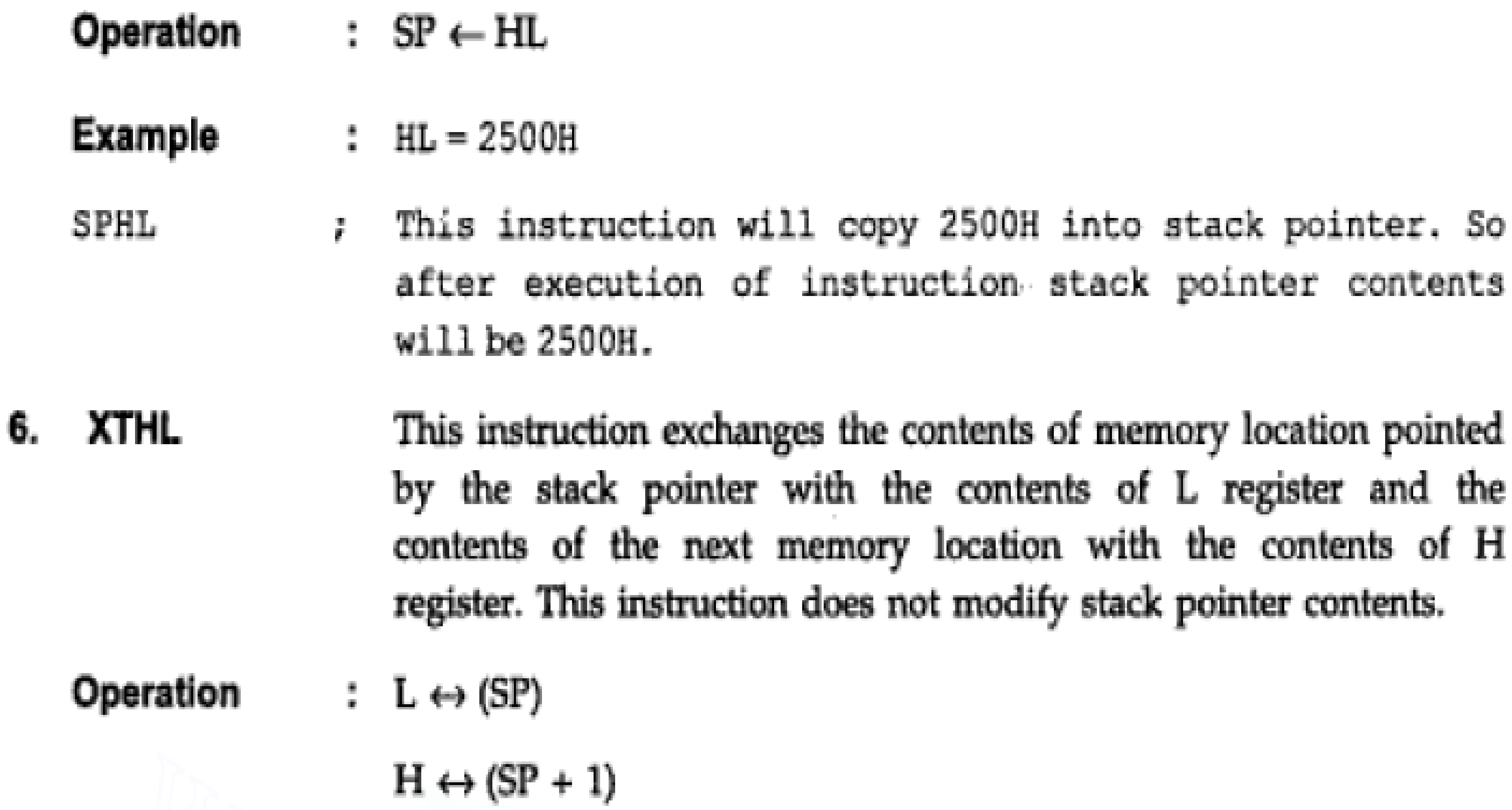
Example : SP = 2000H, (2000H) = 30H, (2001H) = 50H

POP PSW ; This instruction will copy the contents of memory location pointed by the stack pointer, 2000H (i.e. data 30H) into the flag register. It will then increment the stack pointer by one, 2001H and will copy the contents of memory location pointed by stack pointer into the accumulator and increment the stack pointer again by one.



SPHL This instruction copies the contents of HL register pair into the stack pointer. The contents of H register are copied to higher order byte of stack pointer and contents of L register are copied to the lower byte of stack pointer.

Microprocessor……..



Operation : SP ← HL

Example : HL = 2500H

SPHL ; This instruction will copy 2500H into stack pointer. So after execution of instruction stack pointer contents will be 2500H.

6. XTHL This instruction exchanges the contents of memory location pointed by the stack pointer with the contents of L register and the contents of the next memory location with the contents of H register. This instruction does not modify stack pointer contents.

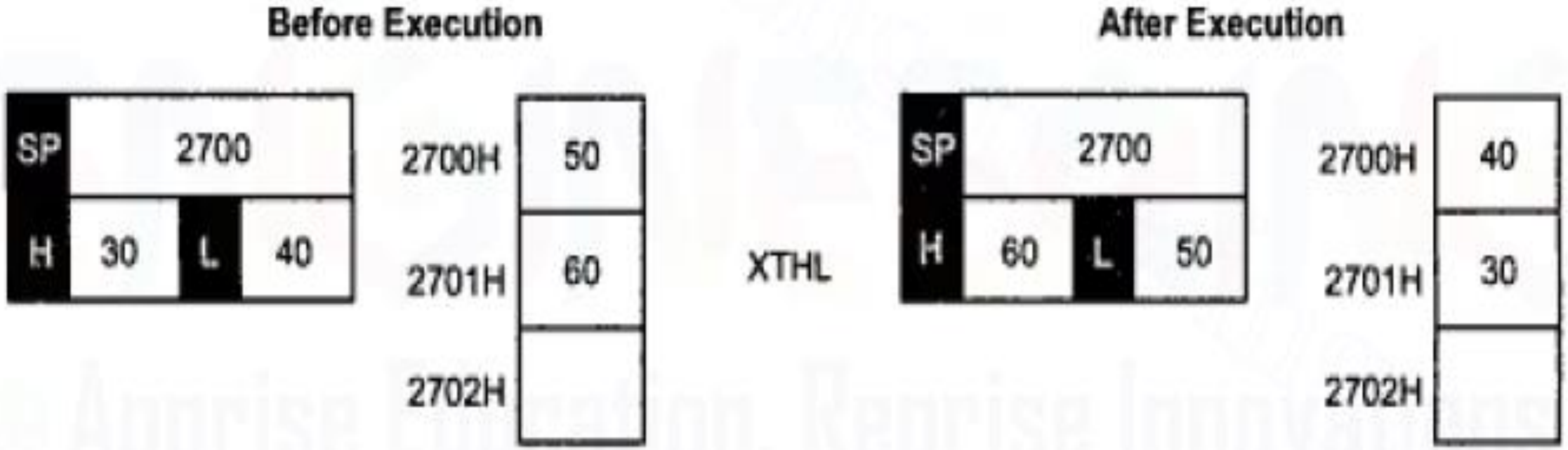
Operation : L ↔ (SP)

H ↔ (SP + 1)

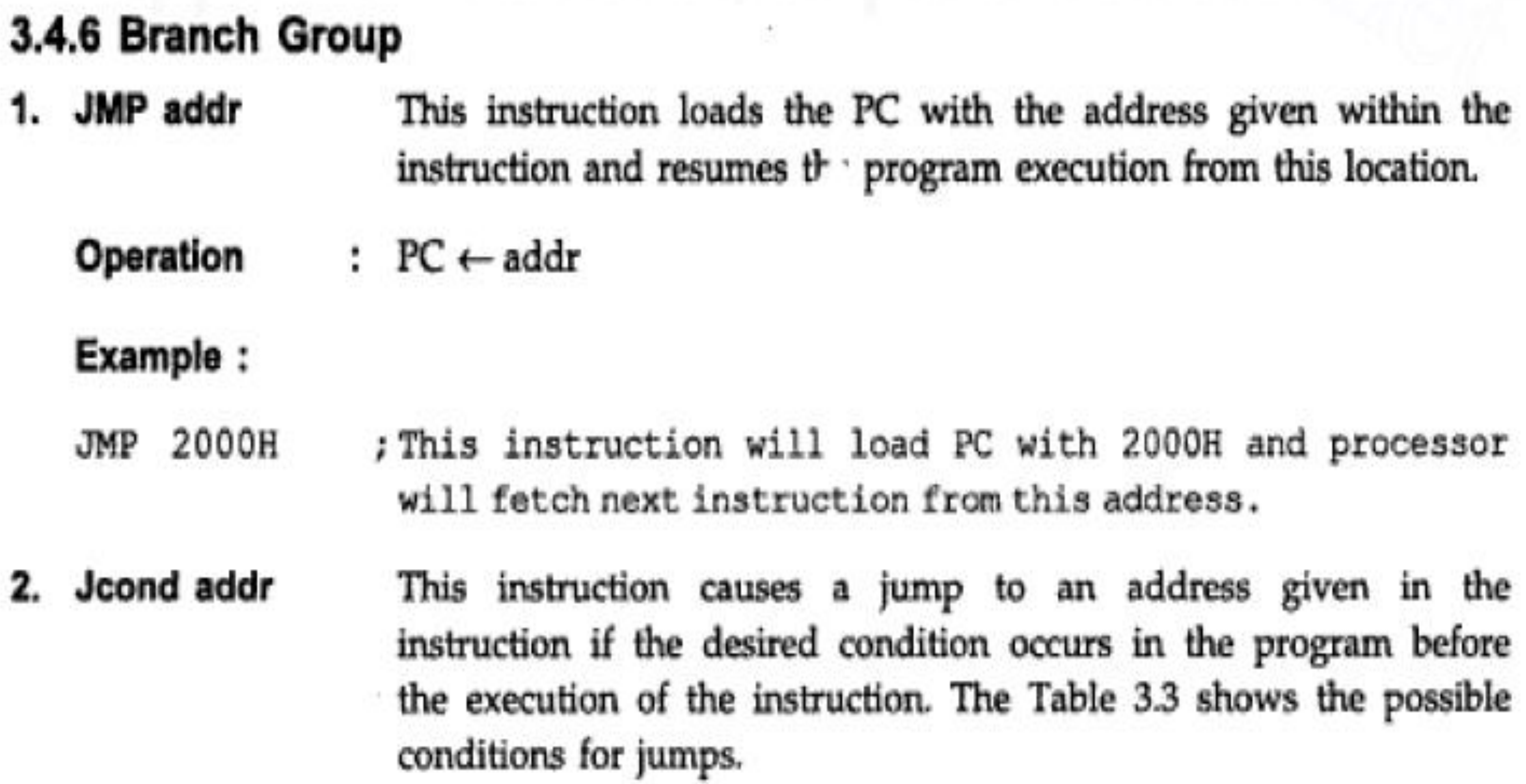
Microprocessor……..

Example : HL = 3040H and SP = 2700H, (2700H) = 50H, (2701H) = 60H

XTHL This instruction will exchange the contents of L register(40H) with the contents of memory location 2700H (i.e. 50H) and the contents of H register (30H) with the contents of memory location 2701H (i.e. 60H).



Microprocessor……..



3.4.6 Branch Group

1. JMP addr This instruction loads the PC with the address given within the instruction and resumes th program execution from this location.

Operation : PC ← addr

Example :

```
JMP 2000H     ;This instruction will load PC with 2000H and processor
               will fetch next instruction from this address.
```

2. Jcond addr This instruction causes a jump to an address given in the instruction if the desired condition occurs in the program before the execution of the instruction. The Table 3.3 shows the possible conditions for jumps.

Microprocessor……..

Instruction code	Description	Condition for jump
JC	Jump on carry	CY = 1
JNC	Jump on not carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0
JZ	Jump on zero	Z = 1
JNZ	Jump on not zero	Z = 0

Table 3.3 Conditional jumps

Example : Carry flag = 1

JC 2000H : This instruction will cause a jump to an address 2000H i.e. program counter will load with 2000H since CF =1.

Microprocessor……..

3. CALL addr

A subroutine is a group of instructions, performs a particular subtask which is executed number of times. It is written separately. The microprocessor executes this subroutine by transferring program control to the subroutine program. After completion of subroutine program execution, the program control is returned back to the main program.

The CALL instruction is used to transfer program control to a subprogram or subroutine. This instruction pushes the current PC contents onto the stack and loads the given address into the PC. Thus the program control is transferred to the address given in the instruction. Stack pointer is decremented by two.

When the subroutine is called, the program control is transferred from calling program to the subroutine. After execution of subroutine it is necessary to transfer program control back to the calling program. To do this processor must remember the address of the instruction next to the CALL instruction. Processor saves this address on the stack when the CALL instruction is executed.

Note : The stack is a part of read/write memory set aside for storing intermediate results and addresses.

Operation : $(SP - 1) \leftarrow PC_H$

$(SP - 2) \leftarrow PC_L$

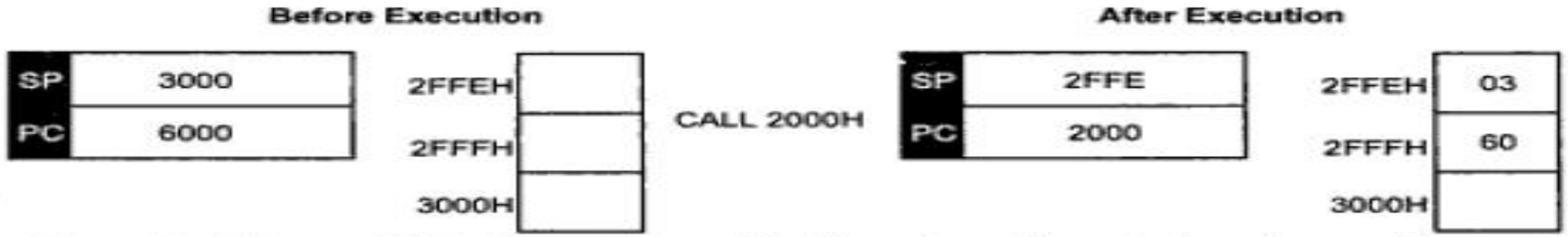
$SP \leftarrow SP - 2$

$PC \leftarrow addr$

Example : `Stack pointer = 3000H.`

Microprocessor……..

```
6000H  CALL  2000H   ; This  instruction  will  store  the  address  of
                       instruction  next  to   CALL  (i.e.  6003H)  on  the
                       stack and load PC
6003H    ---       with 2000H.
```



4. C cond addr This instruction calls the subroutine at the given address if a specified condition is satisfied. Before call it stores the address of instruction next to the call on the stack and decrements stack pointer by two. The Table 3.4 shows the possible conditions for calls.

Instruction code	Description	Condition for CALL
CC	Call on carry	CY = 1
CNC	Call on not carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0
CZ	Call on zero	Z = 1
CNZ	Call on not zero	Z = 0

Table 3.4 Conditional calls

Operation : If condition true $(SP - 1) \leftarrow PCH$

$(SP - 2) \leftarrow PCL$

$PC \leftarrow addr$

else $PC \leftarrow PC + 3$

Example : Carry flag = 1, stack pointer = 4000H.

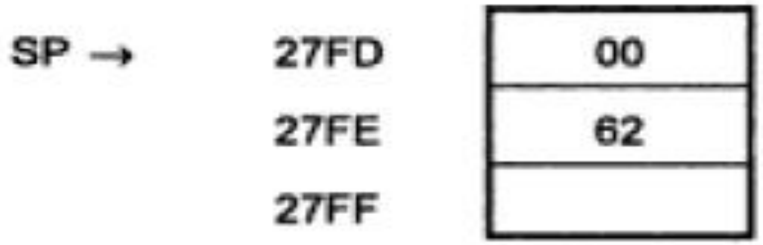
2000H CC 3000H ; This instruction will store the address of the next instruction i.e. 2003H on the stack and load the program counter with 3000H, since the carry flag is set.

5. RET This instruction pops the return addr (address of the instruction next to CALL in the main program) from the stack and loads program counter with this return address. Thus transfers program control to the instruction next to CALL in the main program.

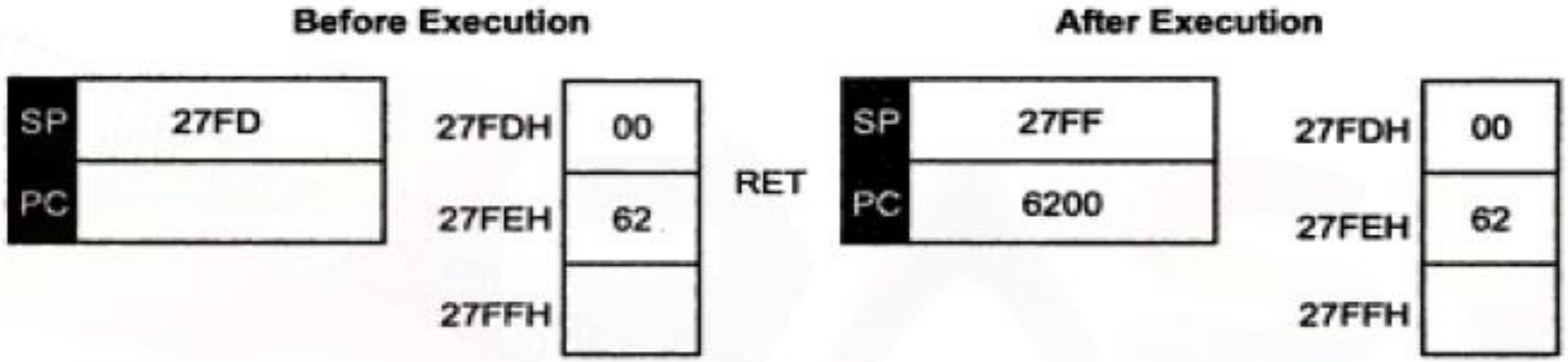
Microprocessor……..

Operation : $PC_L \leftarrow (SP)$
$PC_H \leftarrow (SP+1)$
$SP \leftarrow SP + 2$

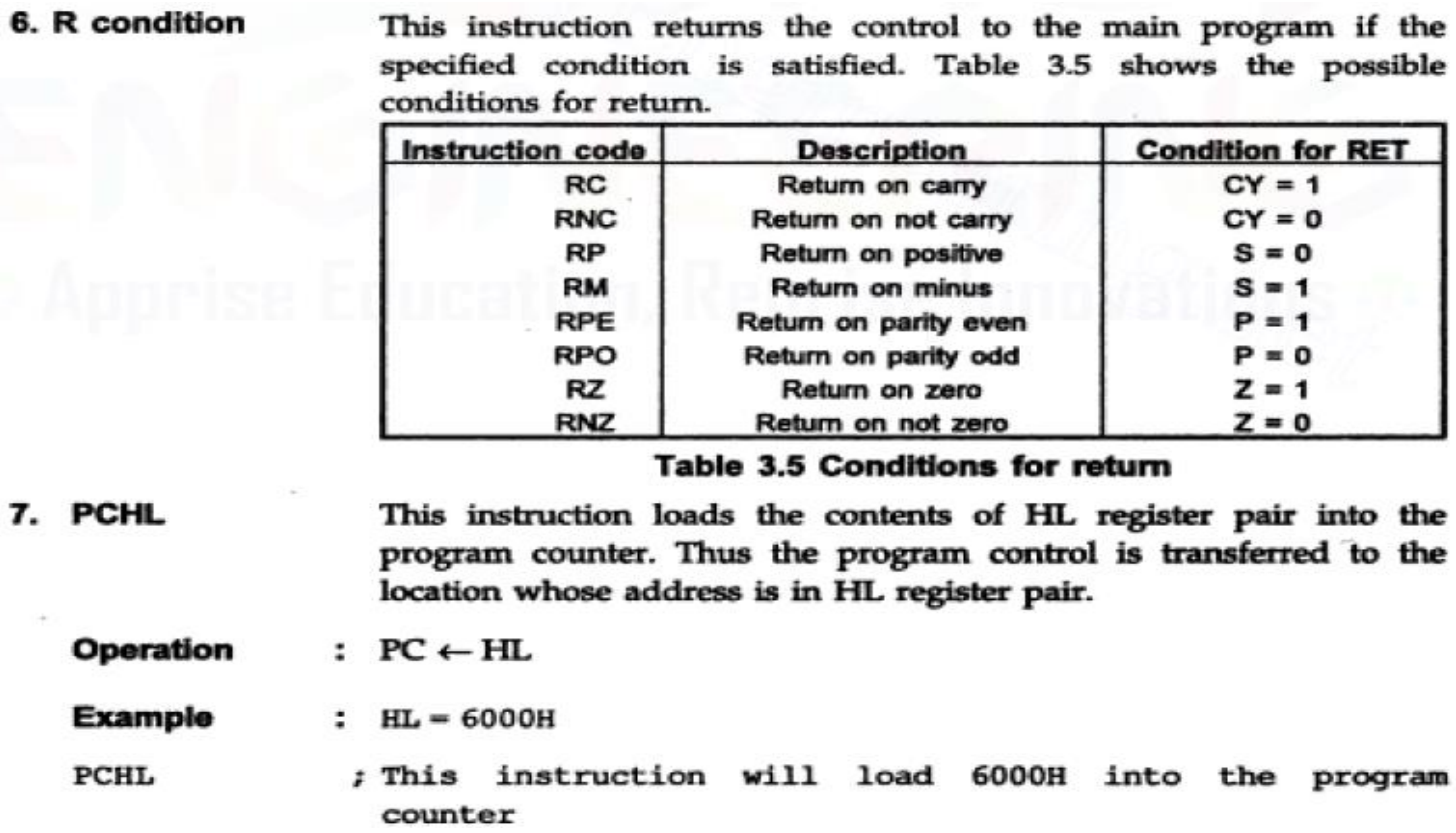
Example : If SP = 27FDH and contents on the stack are as shown then



```
RET            ;This instruction will load PC with 6200H and it will
                transferprogram control to the address 6200H. It will
                also increment the stack pointer by two.
```



Microprocessor……..



6. R condition This instruction returns the control to the main program if the specified condition is satisfied. Table 3.5 shows the possible conditions for return.

Instruction code	Description	Condition for RET
RC	Return on carry	CY = 1
RNC	Return on not carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0
RZ	Return on zero	Z = 1
RNZ	Return on not zero	Z = 0

Table 3.5 Conditions for return

7. PCHL This instruction loads the contents of HL register pair into the program counter. Thus the program control is transferred to the location whose address is in HL register pair.

Operation : PC ← HL

Example : HL = 6000H

```
PCHL        ; This instruction will load 6000H into the program
              counter
```

Microprocessor……..

8. RST n

This instruction transfers the program control to the specific memory address as shown in Table 3.6. This instruction is like a fixed address CALL instruction. These fixed addresses are also referred to as vector addresses. The processor multiplies the RST number by 8 to calculate these vector addresses. Before transferring the program control to the instruction following the vector address RST instruction saves the current program counter contents on the stack like CALL instruction

Instruction code	Vector address
RST 0	0×8 = 0000H
RST 1	1×8 = 0008H
RST 2	2×8 = 0010H
RST 3	3×8 = 0018H
RST 4	4×8 = 0020H
RST 5	5×8 = 0028H
RST 6	6×8 = 0030H
RST 7	7×8 = 0038H

Table 3.6 Vector addresses for return instructions

Operation : $(SP - 1) \leftarrow PC_H$

$(SP - 2) \leftarrow PC_L$

$SP \leftarrow SP - 2$, $PC \leftarrow (n \times 8)$ in hex

Example : SP = 3000H

2000H RST 6; This instruction will save the current contents of the program counter (i.e. address of next instruction 2001H) on the stack and it will load the program counter with vector address 6 × 8= 48_{10} = 30H) 0030H.

3.4.7 Input/Output

1. **IN addr(8-bit)** This instruction copies the data at the port whose address is specified in the instruction into the accumulator.

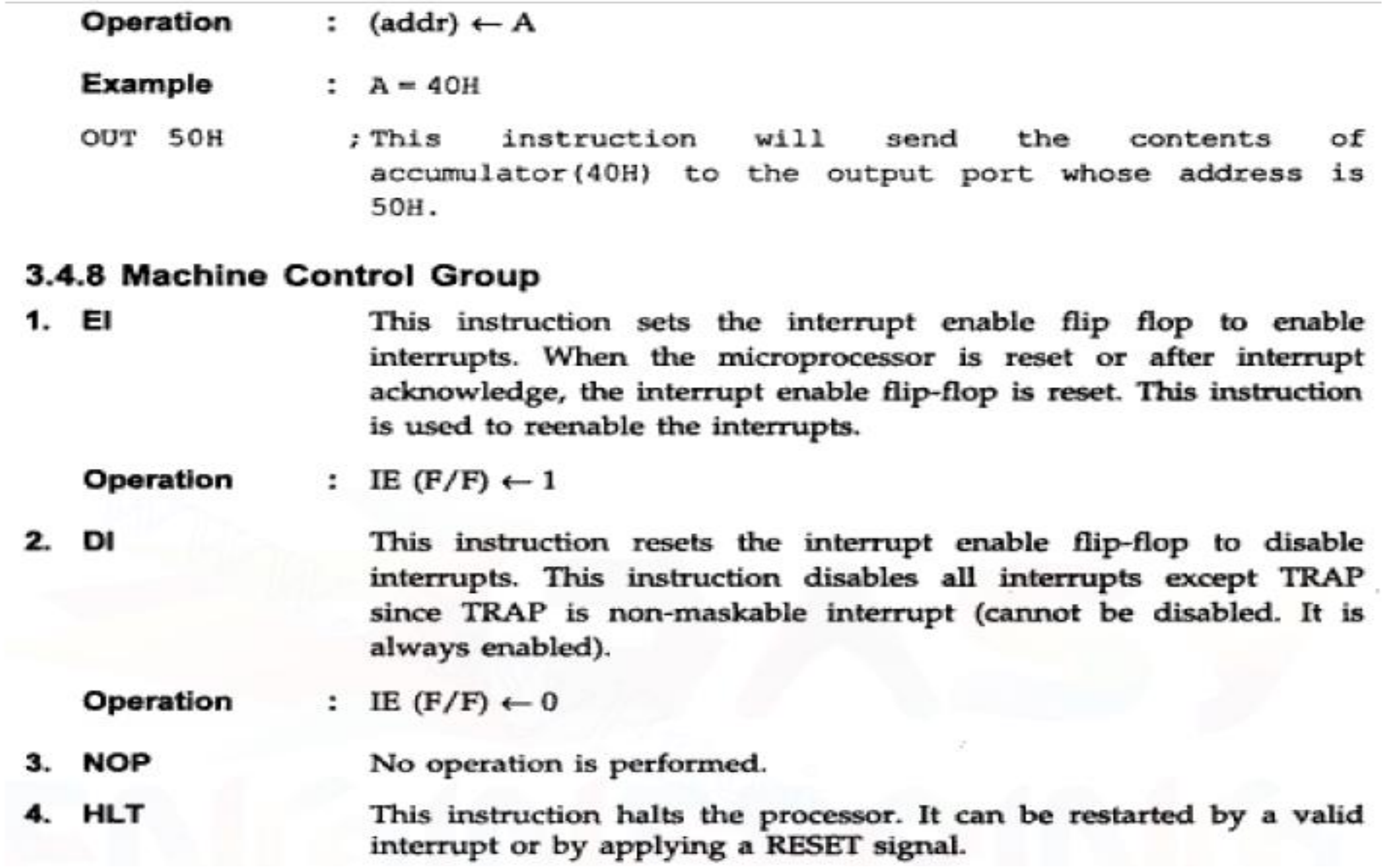
Operation : A ← (addr)

Example : Port address = 80H, data stored at port address 80H, (80H) = 10H

IN 80H ; This instruction will copy the data stored at address 80H, i.e. data 10H in the accumulator.

2. **OUT addr(8-bit)** This instruction sends the contents of accumulator to the output port whose address is specified within the instruction.

Microprocessor……..



Operation : (addr) ← A

Example : A = 40H

```
OUT  50H        ;This instruction will send the contents of
                 accumulator(40H) to the output port whose address is
                 50H.
```

3.4.8 Machine Control Group

1. EI This instruction sets the interrupt enable flip flop to enable interrupts. When the microprocessor is reset or after interrupt acknowledge, the interrupt enable flip-flop is reset. This instruction is used to reenable the interrupts.

Operation : IE (F/F) ← 1

2. DI This instruction resets the interrupt enable flip-flop to disable interrupts. This instruction disables all interrupts except TRAP since TRAP is non-maskable interrupt (cannot be disabled. It is always enabled).

Operation : IE (F/F) ← 0

3. NOP No operation is performed.

4. HLT This instruction halts the processor. It can be restarted by a valid interrupt or by applying a RESET signal.