Stacks & Subroutines

Stack:

→ The stack in 8085 microprocessor is a set of memory locations in the R/W memory, specified by the programmer in a main program.

These memory locations are used to store the binary information (bytes) temporarily during the execution of a program.

→ The instruction LXI SP, 16 bit addr

loads a 16 bit memory address in the stack pointer register of the microprocessor.

→ Once the stack location is identified, storing of data bytes begins at memory address that is one less than the address in stack pointer register.

Ex:- If LXI SP, 2099H

storing of data bytes begins at the address 2098H. and continues in reverse numerical order 2098, 2097, 2096 etc.

→ As a general practice, stack is initialized at the highest available memory location.

→ Data bytes in register pairs of microprocessor can be stored on the stack (two at a time) in reverse order (decreasing memory address) by using the instruction PUSH.

→ Data bytes can be transferred from the stack to the respective register by using the instruction POP.

→ Because two data bytes are being stored at a time, the 16 bit memory address in stack pointer register is decremented by two.

Similarly, when data bytes are retrieved, the address is incremented by two.

Instructions

LXI SP, 16 bit → Load stack pointer
→ Load stack pointer with a 16 bit address.

PUSH Rp → Store register pair on stack
→ 1 byte instruction.
→ copies the content of specified reg. pair on the stack

Ex

PUSH B

→ SP is decremented
→ contents of B reg is loaded into the address in SP
→ SP is decremented
→ contents of C reg. is loaded into the address in SP.

PUSH D,
PUSH H,
PUSH PSW.

POP Rp

→ Retrieves reg. pair from Stack.
→ 1 byte instruction
→ copies the content of top two memory locations of the stack into specified register pair.

POP B

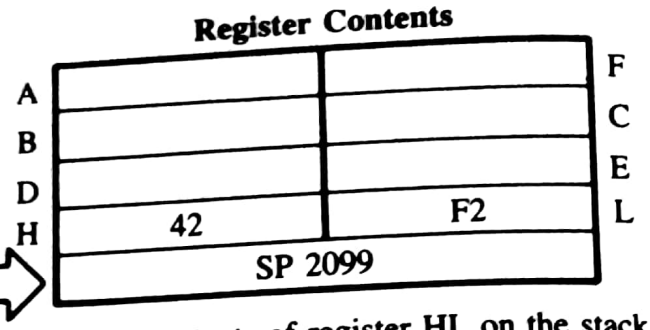
→ contents of memory location specified by SP is moved into reg. C.
SP is incremented by 1.
contents of memory location specified by SP is moved into reg. B.
SP is again incremented by 1

Eg:- POP D
POP H
POP PSW.

Ex:-

In the following set of instructions, the SP is initialized and the contents of register pair HL are stored on the stack by using the instructions PUSH. HL register pair is used for the delay counter; at the end of the delay counter, the contents of HL are retrieved by using the instructions. POP. Assuming the available user memory ranges from 2000H to 20FFH, illustrate the contents of various registers when PUSH and POP instructions are executed.

Memory location	Mnemonic
2000	LXI SP, 2099H
2003	LXI H, 42F2H
2006	PUSH H
2007	DELAY COUNTER
200F	↓
2010	POP H.



Memory Location	Mnemonics	
2000	LXI SP,2099H;	
2003	LXI H,42F2H;	
2006	PUSH H	;Store contents of register HL on the stack
2007	DELAY COUNTER	;The register pair HL can be used by the Delay Counter if necessary
200F	↓	
2010	POP H	;Load HL registers with the contents of the two top locations of the stack

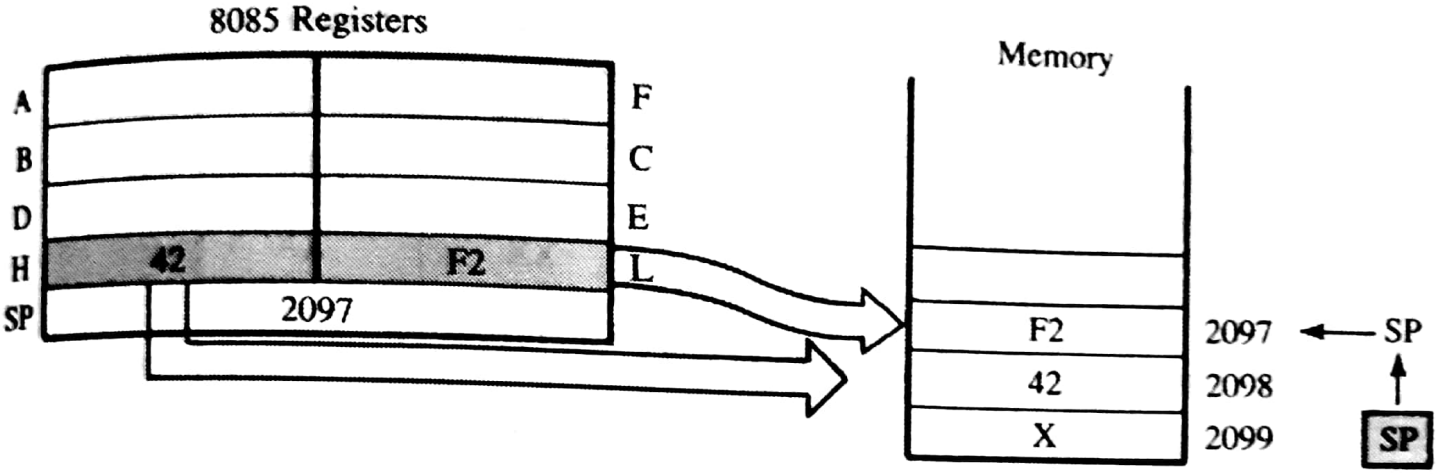


FIGURE 9.2
Contents on the Stack and in the Registers After the PUSH Instruction

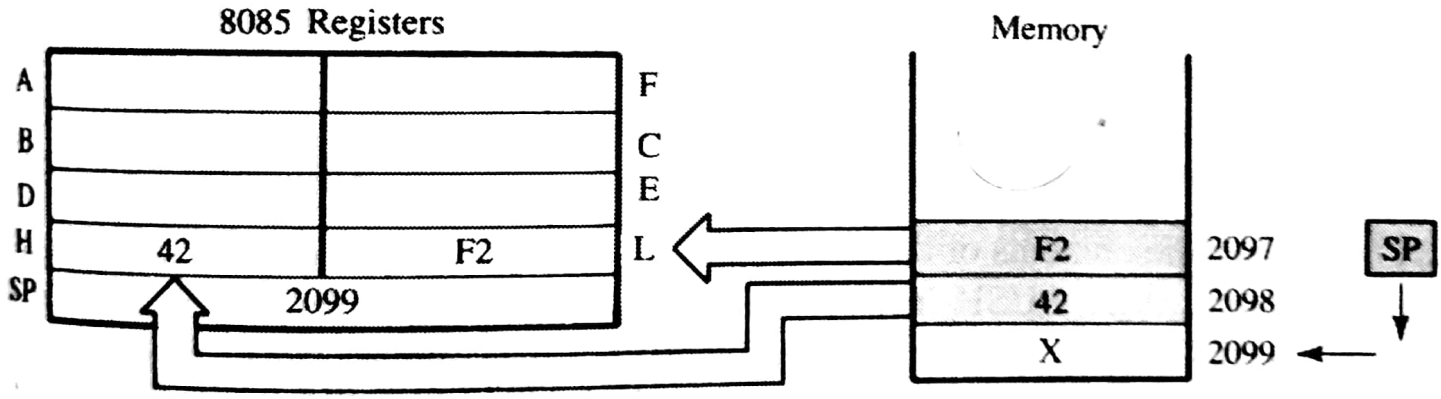


FIGURE 9.3
Contents on the Stack and in the Registers After the POP Instruction

(SP) → 2099 H

(H) → 42

(L) → F2

After the PUSH H instruction

(H) → 42

(L) → F2

SP → SP-1 (2098)

(2098) → 42

SP → SP-1 (2097)

(2097) → F2

program then goes through a delay.

When POP happens

(SP) → 2097

(L) ← F2

SP → SP+1

(SP) → 2098

(H) ← 42

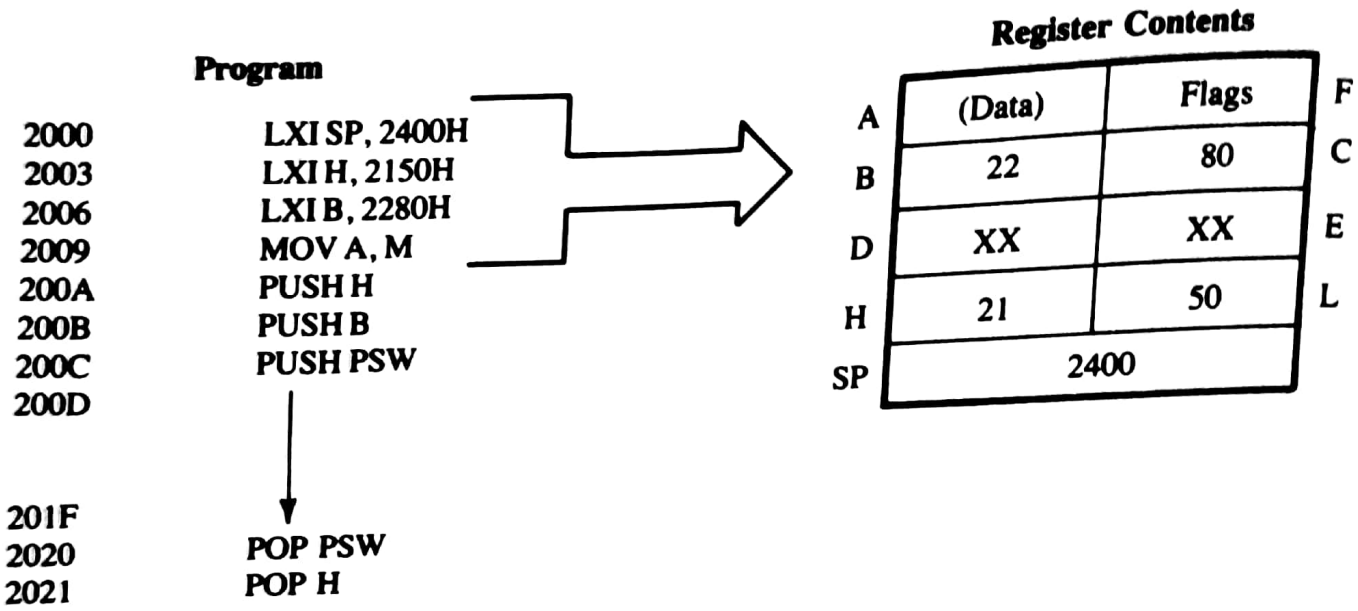
SP → SP+1

(SP) → 2098

Ex:-

The available user memory ranges from 2000H to 2BFFH. A program of data transfer and arithmetic operations is stored in memory locations from 2000H to 2050H, and the stack pointer is initialized at location 2400H. Two sets of data are stored, starting at memory locations 2150H and 2280H. Registers HL, and BC are used as memory pointers to data locations. A segment of program is shown below

1. Explain how SP can be initialized at one memory location beyond the available user memory.
2. Illustrate the contents of stack memory, and registers when PUSH and POP instructions are executed. and explain how memory pointer are exchanged.
3. Explain the various contents of memory.



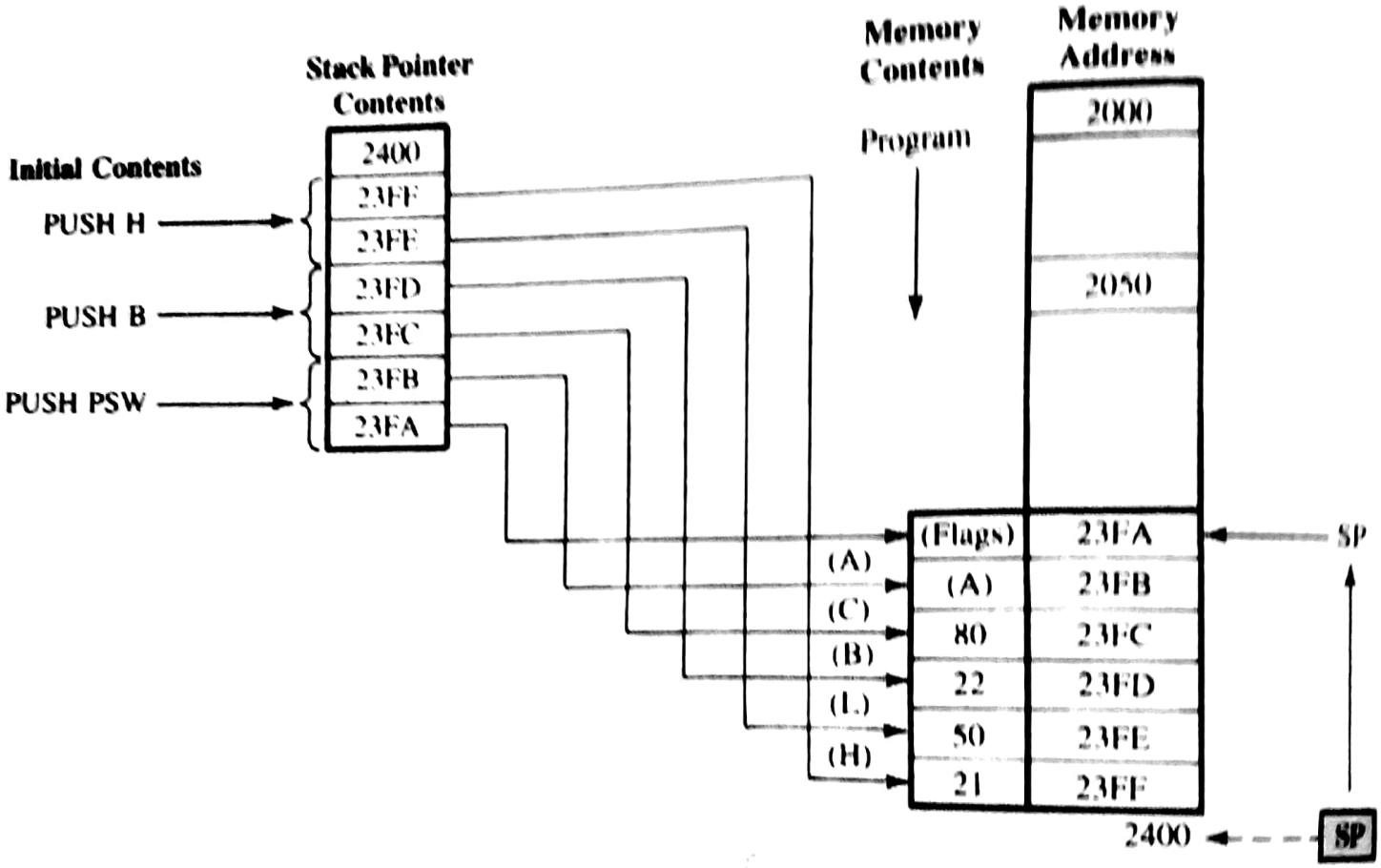


FIGURE 9.5
Stack Contents After the Execution of PUSH Instructions

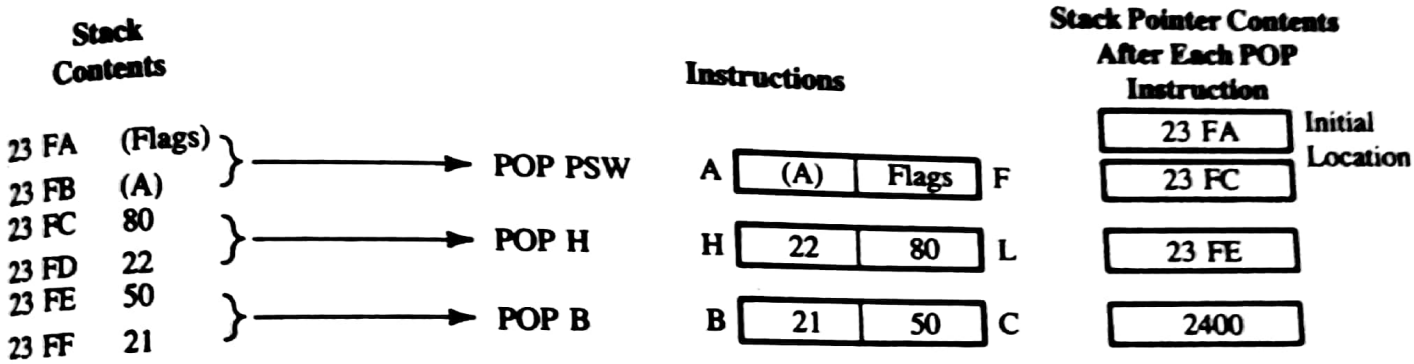
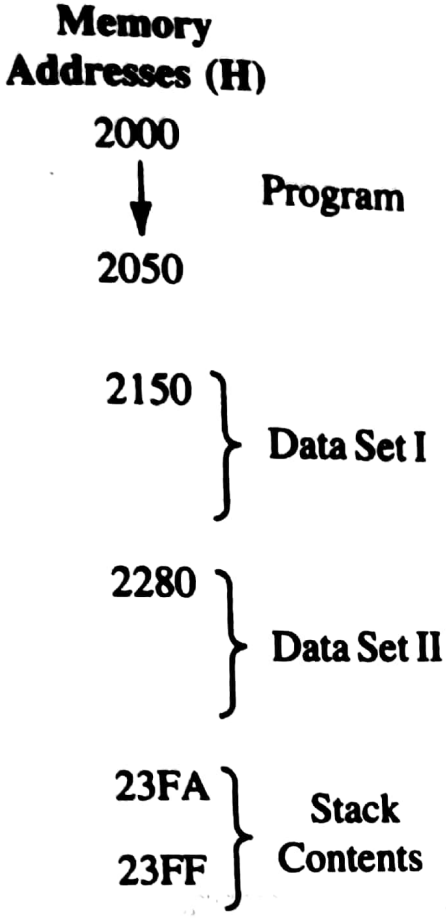


FIGURE 9.6
Register Contents After the Execution of POP Instructions

FIGURE 9.7
R/W Memory Contents



Ex:- Illustrative program: Resetting and displaying flags.

Problem Statement:

Write a program to perform the following functions:

1. clear all flags
2. Load 00H into ACC. and demonstrate that Zero flag is not affected by the data transfer instruction.
3. Logically OR the accumulator with itself to set the Zero flag and display the flag at PORT1, or store all flags on stack.

Soln

```
LXI SP, XX 99H      ; initialize stack
MVI L, 00H          ; clear L
PUSH H              ; place (L) on stack
POP PSW             ; clear flags
MVI A, 00H          ; Load 00H in (A)
PUSH PSW            ; Save flags on stack
POP H               ; Retrieve flags in L
MOV A, L            ; L.
OUT PORT0           ;
MVI A, 00H          ; load 00H again
ORA A               ; set flags & reset CY, AC
PUSH PSW            ; Save flags on stack
POP H               ; Retrieve flags in L
```

PROGRAM

Memory Address	Machine Code	Instructions	Comments
XX00	31	LXI SP,XX99H	;Initialize the stack
01	99		
02	XX		
03	2E	MVI L,00H	;Clear L
04	00		
05	E5	PUSH H	;Place (L) on stack
06	F1	POP PSW	;Clear flags
07	3E	MVI A,00H	;Load 00H
08	00		
09	F5	PUSH PSW	;Save flags on stack
0A	E1	POP H	;Retrieve flags in L
0B	7D	MOV A,L	
0C	D3	OUT PORT0	;Display flags
0D	PORT0		
0E	3E	MVI A,00H	;Load 00H again
0F	00		
10	B7	ORA A	;Set flags and reset CY, AC
11	F5	PUSH PSW	;Save flags on stack
12	E1	POP H	;Retrieve flags in L
13	7D	MOV A,L	
14	E6	ANI 40H	;Mask all flags except Z
15	40		
16	D3	OUT PORT1	
17	PORT1		
18	76	HLT	;End of program

Storing in Memory: Alternative to Output Display

XX0A	3E	MVI A,00H	;Load 00H again
0B	00		
0C	B7	ORA A	;Set flags and reset CY and AC
0D	F5	PUSH PSW	;Save flags on stack
0E	76	HLT	;End of program

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	0	0	1	0	0	= 40H
S	Z		AC		P		CY	

Subroutine:

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

Ex:- If a time delay is required between three successive events, three delays can be written in the main program.
To avoid repetition of the same delay instructions, the subroutine technique can be used.
Delay instructions are written once, seperately from the main program and are called by the main program when needed.

→ 8085 has two instructions to implement subroutines:

CALL (call a subroutine)
RET (return to main program from subroutine)

CALL is used in the main program to call a subroutine.
RET is used at the end of the subroutine to return to the main program.

→ When a subroutine is called, contents of PC, which is the address of the instruction following CALL instruction is stored on the stack and program execution is transferred to subroutine.

→ When RET is executed at the end of subroutine the memory address stored on stack is retrieved and sequence of execution is resumed in the main program.

INSTRUCTIONS

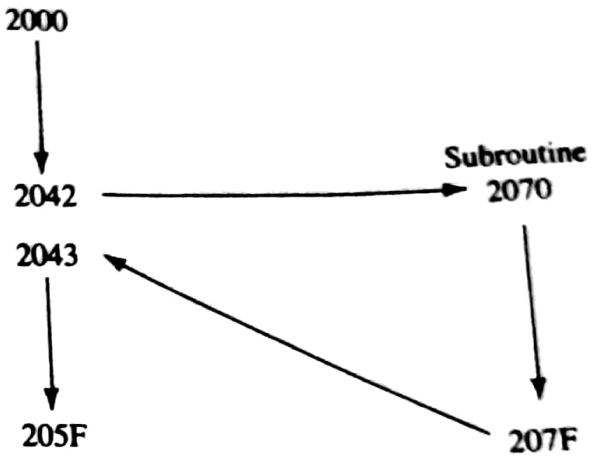
Opcode	Operand	
CALL	16-bit memory address of a subroutine	Call Subroutine Unconditionally ☐ This is a 3-byte instruction that transfers the program sequence to a subroutine address ☐ Saves the contents of the program counter (the address of the next instruction) on the stack ☐ Decrements the stack pointer register by two ☐ Jumps unconditionally to the memory location specified by the second and third bytes. The second byte specifies a line number and the third byte specifies a page number ☐ This instruction is accompanied by a return instruction in the subroutine
RET		Return from Subroutine Unconditionally ☐ This is a 1-byte instruction ☐ Inserts the two bytes from the top of the stack into the program counter and increments the stack pointer register by two ☐ Unconditionally returns from a subroutine

Illustrate the exchange of information between the stack and the program counter for the following program if the available user memory ranges from 2000H to 23FFH.

Memory Address		
2000	LXI SP,2400H	;Initialize the stack pointer at 2400H
↓	↓	
2040	CALL 2070H	;Call the subroutine located at 2070H. This is
2041		; a 3-byte instruction
2042		
2043	NEXT INSTRUCTION	;The address of the next instruction following
		; the CALL instruction
↓	↓	
205F	HLT	;End of the main program
2070	First Subroutine	;Beginning of the subroutine
	Instruction	
↓	↓	
207F	RET	;End of the subroutine
2080	↓	
	Other Subroutines	
↓	↓	
2398	Empty Space	
23FF	↓	
2400		;The stack is initialized at 2400H

STACK

FIGURE 9.9
Subroutine Call and Program Transfer



The program execution begins at 2000_H, continues until the end of CALL 2042_H, and transfers to the subroutine at 2070_H. At the end of he subroutine, after executing the RET instruction, it comes back to the main program at 2043H and continues.

CALL EXECUTION

Memory Address	Machine Code	Mnemonics	Comments
2040	CD	CALL 2070H	;Call subroutine located at the memory
2041	70		; location 2070H
2042	20		
2043	NEXT	INSTRUCTION	

The sequence of events in the execution of the CALL instruction by the 8085 is shown in Figure 9.10. The instruction requires five machine cycles and eighteen T-states. The sequence of events in each machine cycle is as follows.

1. M_1—Opcode Fetch: In this machine cycle, the contents of the program counter (2040H) are placed on the address bus, and the instruction code CD is fetched using the data bus. At the same time, the program counter is upgraded to the next memory address, 2041H. After the instruction is decoded and executed, the stack pointer register is decremented by one to 23FFH.
2. M_2 and M_3—Memory Read: These are two Memory Read cycles during which the 16-bit address (2070H) of the CALL instruction is fetched. The low-order address 70H is fetched first and placed in the internal register Z. The high-order address 20H is fetched next, and placed in register W. During M_3, the program counter is upgraded to 2043H, pointing to the next instruction.
3. M_4 and M_5—Storing of Program Counter: At the beginning of the M_4 cycle, the normal operation of placing the contents of the program counter on the address bus is suspended; instead, the contents of the stack pointer register 23FFH are placed on the address bus. The high-order byte of the program counter (PCH = 20H) is placed on the data bus and stored in the stack location 23FFH. At the same time, the stack pointer register is decremented to 23FEH.

   During machine cycle M_5, the contents of the stack pointer 23FEH are placed on the address bus. The low-order byte of the program counter (PCL = 43H) is placed on the data bus and stored in stack location 23FEH.

Memory Address	Code (H)
2040	CD
2041	70
2042	20

Instruction: CALL 2070H

Machine Cycles	Stack Pointer (SP) 2400	Address Bus (AB)	Program Counter (PCH) (PCL)	Data Bus (DB)	Internal Registers (W) (Z)
M_1 Opcode Fetch	23FF (SP–1)	2040	20 41	CD Opcode	—
M_2 Memory Read		2041	20 42	70 Operand	→ 70
M_3 Memory Read	23FF	2042	20 43	20 Operand	→ 20
M_4 Memory Write	23FE (SP–2)	23FF	20 43	20 (PCH)	
M_5 Memory Write	23FE	23FE	20 43	43 (PCL)	(20) (70)
M_1 Opcode Fetch of Next Instruction		20 70 → (W)(Z)	→ 2071		(2070) → (W)(Z)

FIGURE 9.10
Data Transfer During the Execution of the CALL Instruction

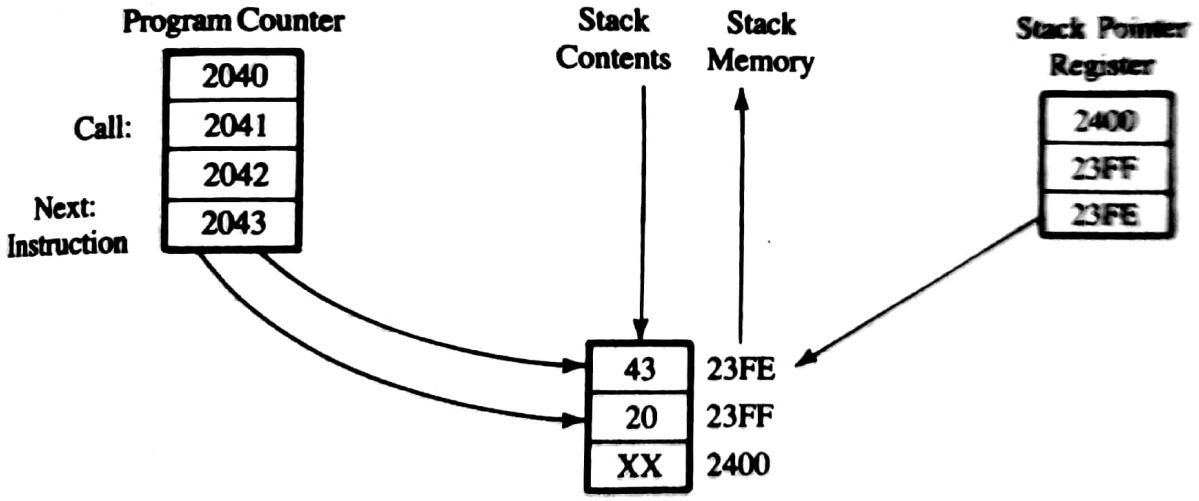


FIGURE 9.11
Contents of the Program Counter, the Stack Pointer, and the Stack During the Execution of the CALL Instruction

Instruction: RET

Memory Address	Code (H)
207F	C9

Contents of Stack Memory	
23FE	43
23FF	20

Machine Cycles	Stack Pointer (23FE)	Address Bus (AB)	Program Counter	Data Bus (DB)	Internal Registers (W) (Z)
M_1 Opcode Fetch	23FE	207F	2080	C9 Opcode	
M_2 Memory Read	23FF	23FE		43 (Stack) →	→ 43
M_3 Memory Read	2400	23FF		20 (Stack−1) →	→ 20
M_1 Opcode Fetch of Next Instruction		2043 (W) (Z) ←	2044		→ 2043 (W) (Z)

FIGURE 9.12
Data Transfer During the Execution of the RET Instruction

Restart, conditional call and Return instructions:

Restart (RST) Instructions

→ RST instructions are 1 byte call instruction that transfer the program execution to a specific location on page 00H.

→ These are executed in the same way as CALL instruction.

→ There are 8 RST instructions

RST 0	CALL 0000H
RST 1	CALL 0008H
RST 2	CALL 0010H
RST 3	CALL 0018H
RST 4	CALL 0020H
RST 5	CALL 0028H
RST 6	CALL 0030H
RST 7	CALL 0038H.

conditional Call and Return Instructions

→ These are based on four data conditions (flags): carry, zero, sign, parity.

conditional Call

CC → Call subroutine if CY=1

CNC → Call subroutine if CY=0

CZ → Call subroutine if Z=1

CNZ → Call subroutine if Z=0

CM → Call subroutine if S=1

CP → Call subroutine if S=0

CPE → Call subroutine if P=1 (even)

CPO → Call subroutine if P=0 (odd)

Conditional Return

RC

RNC

RZ

RNZ

RM

RP

RPE

RPO

Note: Based on above concept, a subroutine can have multiple conditional returns.

Advanced Instructions:

→ The instructions explained previously deal with 8 bit data (except LXI)

However, in some cases, data larger than 8 bits must be manipulated especially in arithmetic manipulations and stack operations.

→ Even if 8085 is an 8 bit microprocessor, its architecture allows specific combinations of two 8-bit registers to form 16 bit registers.

We also have several instruction to manipulate 16 bit data.

16 bit Data Transfer & Data Exchange group:

LHLD : Load HL registers direct
- → 3byte instruction
- → second and third bytes specify a memory location
- → Transfers the contents of specified memory location into L register
- → Transfer the contents of next memory location to H registers

SHLD : Store HL register direct
- → 3 byte instructions
- → second and third bytes specify a memory location
- → Stores the content of L register in the specified memory location.
- → Stores the content of H register in the next memory location.

XCHG : Exchange the contents of HL and DE
→ 1 byte instruction
→ contents of H reg exchanged with contents of D reg
contents of L reg. exchanged with content of E reg.

Ex:- Memory locations 2050H and 2051H contain 3FH and 42H respectively. Register pair DE contains 856FH. Write instructions to exchange the contents of DE with the contents of memory locations.

Sol

```
LHLD 2050H
XCHG
SHLD 2050
```

(E) = 3F
(D) = 42
(H) = 85 (D) = 42
(L) = 6F (E) = 3F
(2050) = 6F
(2051) = 85

Arithmetic Group:

Operation: Add with carry

ADC R
ADC M
ACI 8 bit

} These instructions add the contents of the operand, the carry and accumulator. All flags are affected.

Operation: Subtraction with carry

SBB R
SBB M
SBI 8 bit

} These instructions subtract the contents of operand and borrow from the content of accumulator.

Ex:- Registers BC contain 2793H and registers DE contain 3182H. Write instructions to add these two 16 bit numbers and place the sum in memory location 2050H and 2051H

Sol

Before instructions:

B | 27 | 93 | C

D | 31 | 82 | E

MOV A, C	A | 93 | F	
ADD E	A | 15 | CY	93 H 82 H 1|15 H
MOV L, A	H | 15 | L	
MOV A, B	A | 27 |	1 CY 27 H 31 H 59 H
ADC D	A | 59 | CY=0 F	
MOV H, A	H | 59 | 15	
SHLD 2050H	(2050) → 15 (2051) → 59	

Ex:- Registers BC contain 8538H and registers DE contain 62A5H. Write instructions to subtract the contents of DE from the contents of BC and place the result in BC.

Sol

```
MOV A, C
SUB E
MOV C, A
MOV A, B
SBB D
MOV B, A
```

(B) 85 | 38 (C)

(D) 62 | A5

-1 | 1|93

(B) 22 | 93 (C)

operation : Double Register Add

DAD Rp

Add register pair to register H L

→ 1 byte instruction

→ Adds the contents of operand (register pair or stack pointer) to the contents of HL register

→ The result is placed in HL register

→ The CY flag is altered to reflect the result of 16 bit addition.
No other flags are affected

→ The instruction set includes 4 instructions

DAD B
DAD D
DAD H
DAD SP

Ex:- Write instruction to display the contents of stack pointer register at output ports :

```
LXI H, 0000H    ; clear HL
DAD SP          ; place stack pointer contents in HL
MOV A, H        ; place higher order address of SP in Acc.
OUT PORT1
MOV A, L        ; place lower order address of SP in Acc
OUT PORT2
```

Instructions related to Stack pointer and the program counter

XTHL : Exchange top of the stack with H & L

→ The contents of L are exchanged with the contents of memory location shown by SP and the contents of H are exchanged with the contents of memory location SP+1.

SPHL : copy H and L registers into stack pointer register

→ The contents of H specify the higher order byte and contents of L specify the low-order byte.

→ The contents of HL registers are not affected.

PCHL : copy H and L registers into program counter

→ The contents of H specify the higher order byte and contents of L specify the low order byte

CMC : complement the carry flag (CY)

STC : Set the carry flag