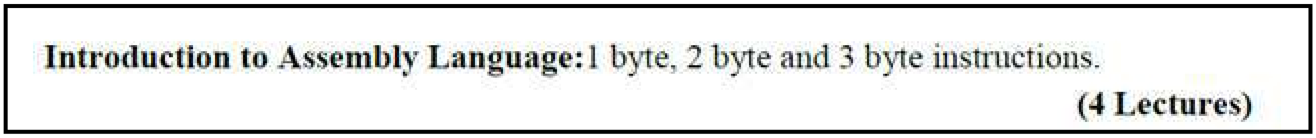
Digital Systems & Applications

Chapter - 13

13. Intel 8085 Microprocessor Architecture

Chapter – 13

Introduction to Assembly Language



Introduction to Assembly Language:1 byte, 2 byte and 3 byte instructions.

(4 Lectures)

Que1. What is an instruction?

Ans. An instruction is a command given to the microcomputer to perform a specific task or function on a given data.

Que2. What is meant by instruction set?

Ans. An instruction set is a collection of instructions that the microprocessor is designed to perform.

Que3. In how many categories the instructions of 8085 be classified?

Ans. Functionally, the instructions can be classified into five groups:

z data transfer (copy) group
z arithmetic group
z logical group
z branch group
z stack, I/O and machine control group.

Que4. What are the different types of data transfer operations possible?

Ans. The different types of data transfer operations possible are cited below:

z Between two registers.
z Between a register and a memory location.
z A data byte can be transferred between a register and a memory location.
z Between an I/O device and the accumulator.
z Between a register pair and the stack.

The term 'data transfer' is a misnomer—actually data is not transferred, but copied from source to destination.

Que5: What is a three-byte instruction?

Ans: In a three-byte instruction, the first byte specifies the Opcode, and the

following two bytes specifies the 16- bit address.

Important Instructions of 8085

Table below gives important instructions for 8085 which will be useful for writing assembly language programs.

Control Instructions

https://alllabexperiments.com

Opcode	Operand	Explanation of Instruction	Description
NOP	**none**	No operation	No operation is performed. The instruction is fetched and decoded. However no operation is executed. `Example: NOP`
HLT	**none**	Halt and enter wait state	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. `Example: HLT`

LOGICAL INSTRUCTIONS

Opcode	Operand	Explanation of Instruction	Description
CMP	**R** **M**	Compare register or memory with accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < (reg/mem): carry flag is set if (A) = (reg/mem): zero flag is set if (A) > (reg/mem): carry and zero flags are reset `Example: CMP B or CMP M`

CPI	**8-bit data**	Compare immediate with accumulator	The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < data: carry flag is set if (A) = data: zero flag is set if (A) > data: carry and zero flags are reset `Example: CPI 89H`
ANA	**R** **M**	Logical AND register or memory with accumulator	The contents of the accumulator are logically ANDed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set. `Example: ANA B or ANA M`
ANI	**8-bit data**	Logical AND immediate with accumulator	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set. `Example: ANI 86H`
XRA	**R** **M**	Exclusive OR register or memory with accumulator	The contents of the accumulator are Exclusive ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. `Example: XRA B or XRA M`
XRI	**8-bit data**	Exclusive OR immediate with accumulator	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. `Example: XRI 86H`
ORA	**R** **M**	Logical OR register or memory with	The contents of the accumulator are logically ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location,

		accumulator	its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. `Example: ORA B or ORA M`
ORI	**8-bit data**	Logical OR immediate with accumulator	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. `Example: ORI 86H`
RLC	**none**	Rotate accumulator left	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected. `Example: RLC`
RRC	**none**	Rotate accumulator right	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected. `Example: RRC`
RAL	**none**	Rotate accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. `Example: RAL`
RAR	**none**	Rotate accumulator right through carry	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. `Example: RAR`
CMA	**none**	Complement accumulator	The contents of the accumulator are complemented. No flags are affected. `Example: CMA`
CMC	**none**	Complement carry	The Carry flag is complemented. No other flags are affected.

			Example: CMC
STC	**none**	Set Carry	Set Carry Example: STC

BRANCHING INSTRUCTIONS

<table>
<tr><th>Opcode</th><th>Operand</th><th>Explanation of Instruction</th><th>Description</th></tr>
<tr><td><b>JMP</b></td><td><b>16-bit address</b></td><td>Jump unconditionally</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
Example: JMP 2034H or JMP XYZ</td></tr>
<tr><td><table>
<tr><td>Opcode</td><td>Description</td><td>Flag Status</td></tr>
<tr><td><b>JC</b></td><td><b>Jump on Carry</b></td><td><b>CY = 1</b></td></tr>
<tr><td><b>JNC</b></td><td><b>Jump on no Carry</b></td><td><b>CY = 0</b></td></tr>
<tr><td><b>JP</b></td><td><b>Jump on positive</b></td><td><b>S = 0</b></td></tr>
<tr><td><b>JM</b></td><td><b>Jump on minus</b></td><td><b>S = 1</b></td></tr>
<tr><td><b>JZ</b></td><td><b>Jump on zero</b></td><td><b>Z = 1</b></td></tr>
<tr><td><b>JNZ</b></td><td><b>Jump on no zero</b></td><td><b>Z = 0</b></td></tr>
<tr><td><b>JPE</b></td><td><b>Jump on parity even</b></td><td><b>P = 1</b></td></tr>
<tr><td><b>JPO</b></td><td><b>Jump on parity odd</b></td><td><b>P = 0</b></td></tr>
</table></td><td><b>16-bit address</b></td><td>Jump conditionally</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.
Example: JZ 2034H or JZ XYZ</td></tr>
<tr><td><table>
<tr><td>Opcode</td><td>Description</td><td>Flag Status</td></tr>
<tr><td>CC</td><td>Call on Carry</td><td>CY = 1</td></tr>
<tr><td>CNC</td><td>Call on no Carry</td><td>CY = 0</td></tr>
<tr><td>CP</td><td>Call on positive</td><td>S = 0</td></tr>
</table></td><td><b>16-bit address</b></td><td>Unconditional subroutine call</td><td>The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents</td></tr>
</table>

<table>
<tr>
<td>
<table>
<tr><td>CM</td><td>Call on minus</td><td>S = 1</td></tr>
<tr><td>CZ</td><td>Call on zero</td><td>Z = 1</td></tr>
<tr><td>CNZ</td><td>Call on no zero</td><td>Z = 0</td></tr>
<tr><td>CPE</td><td>Call on parity even</td><td>P = 1</td></tr>
<tr><td>CPO</td><td>Call on parity odd</td><td>P = 0</td></tr>
</table>
</td>
<td></td>
<td></td>
<td>of the program counter) is pushed onto the stack.
Example: CALL 2034H or CALL XYZ</td>
</tr>
<tr>
<td>RET</td>
<td>none</td>
<td>Return from subroutine unconditionally</td>
<td>The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter,and program execution begins at the new address.
Example: RET</td>
</tr>
</table>

Arithmetic Instructions

Opcode	Operand	Explanation of Instruction	Description
ADD	R M	Add register or memory, to accumulator	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M
ADC	R M	Add register to accumulator with carry	The contents of the operand (register or memory) and M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADC B or ADC M

ADI	**8-bit data**	Add immediate to accumulator	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. `Example: ADI 45H`
ACI	**8-bit data**	Add immediate to accumulator with carry	The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. `Example: ACI 45H`
LXI	**Reg. pair, 16-bit data**	Load register pair immediate	The instruction loads 16-bit data in the register pair designated in the operand. `Example: LXI H, 2034H or LXI H, XYZ`
DAD	**Reg. pair**	Add register pair to H and L registers	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected. `Example: DAD H`
SUB	**R** **M**	Subtract register or memory from accumulator	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. `Example: SUB B or SUB M`
SBB	**R** **M**	Subtract source and borrow from accumulator	The contents of the operand (register or memory) and M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. `Example: SBB B or SBB M`
SUI	**8-bit data**	Subtract immediate from accumulator	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags

			are modified to reflect the result of the subtraction. Example: SUI 45H
SBI	**8-bit data**	Subtract immediate from accumulator with borrow	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example: XCHG
INR	**R** **M**	Increment register or memory by 1	The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: INR B or INR M
INX	**R**	Increment register pair by 1	The contents of the designated register pair are incremented by 1 and the result is stored in the same place. Example: INX H
DCR	**R** **M**	Decrement register or memory by 1	The contents of the designated register or memory are M decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: DCR B or DCR M
DCX	**R**	Decrement register pair by 1	The contents of the designated register pair are decremented by 1 and the result is stored in the same place. Example: DCX H
DAA	**none**	Decimal adjust accumulator	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the

			high-order four bits. Example: DAA

Data Transfer Instructions

Opcode	Operand	Explanation of Instruction	Description
MOV	**Rd, Rs** **M, Rs** **Rd, M**	Copy from source(Rs) to destination(Rd)	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. `Example: MOV B, C or MOV B, M`
MVI	**Rd, data** **M, data**	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. `Example: MVI B, 57H or MVI M, 57H`
LDA	**16-bit address**	Load accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. `Example: LDA 2034H`
LDAX	**B/D Reg. pair**	Load accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. `Example: LDAX B`

LXI	**Reg. pair, 16-bit data**	Load register pair immediate	The instruction loads 16-bit data in the register pair designated in the operand. `Example: LXI H, 2034H or LXI H, XYZ`
LHLD	**16-bit address**	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. `Example: LHLD 2040H`
STA	**16-bit address**	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. `Example: STA 4350H`
STAX	**Reg. pair**	Store accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. `Example: STAX B`
SHLD	**16-bit address**	Store H and L registers direct	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. `Example: SHLD 2470H`
XCHG	**none**	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. `Example: XCHG`
SPHL	**none**	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H

			and L registers are not altered. `Example: SPHL`
XTHL	**none**	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered. `Example: XTHL`
PUSH	**Reg. pair**	Push register pair onto stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the highorder register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. `Example: PUSH B or PUSH A`
POP	**Reg. pair**	Pop off stack to register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. `Example: POP H or POP A`
OUT	**8-bit port address**	Output data from accumulator to a port with 8-bit address	The contents of the accumulator are copied into the I/O port specified by the operand. `Example: OUT F8H`
IN	**8-bit port address**	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. `Example: IN 8CH`

Que6: Memory location 3050 H is specified by HL pair and contains data FE H. Accumulator contains 14 H. Add the contents of memory location with accumulator. Store the result in 2050 H.

Ans:

The program is written as follows:

```
        LXIH, 3050 H  ⇒  HL register pair points 3050 H memory location
             ADD M    ⇒  Add contents of 3050 H with accumulator
          STA 2050 H  ⇒  Result in accumulator is stored in 2050 H
```

Prior to execution of the program, memory location 3050 H is loaded with data FE H and accumulator with 14 H.

Que7: Write down the result of EX–OR operation on accumulator and register B. Assume ACC = 18 H and B = 27 H.

Ans:

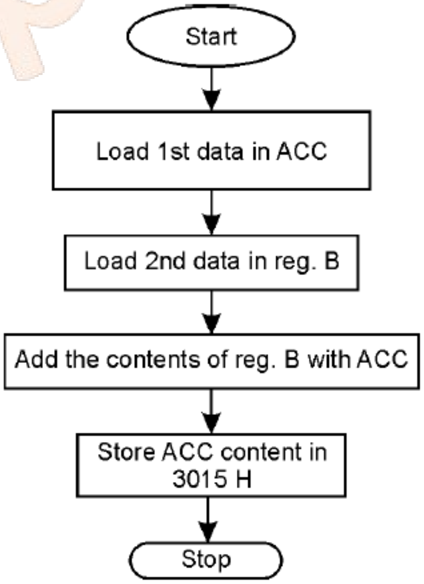
The instruction used for this operation is XRA B.

ACC	=	0	0	0	1	1	0	0	0
B	=	0	0	1	0	0	1	1	1
EX–OR	⇒	0	0	1	1	1	1	1	1
	=	3F H							

The result 3F H of EX–OR is stored in accumulator.

Que8: Draw the flow chart and write down the program to add two numbers 16 H and D2 H. Store the result in memory location 3015 H.

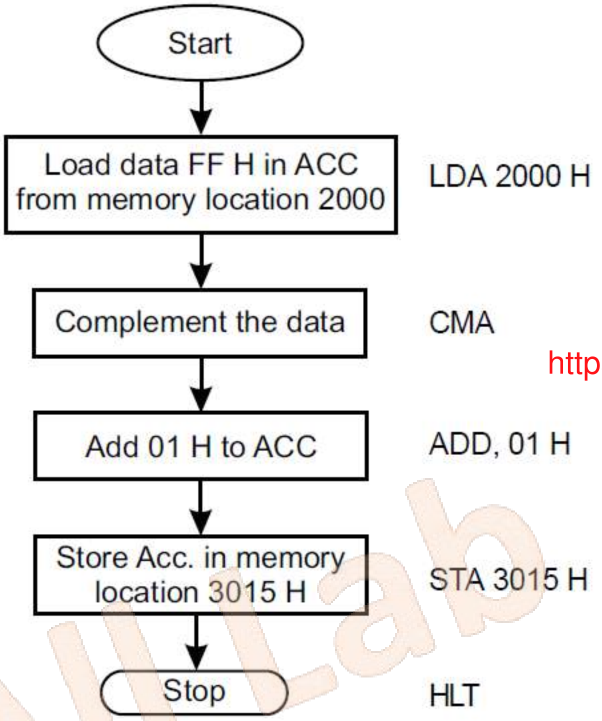
Ans. The flowchart for the above would look as follows.


Start
Load 1st data in ACC
Load 2nd data in reg. B
Add the contents of reg. B with ACC
Store ACC content in 3015 H
Stop

```
MVI A, 16 H

MVI B, D2H

ADD B

STA 3015 H

HLT
```

Que9: Write a program, along with flowchart, to find the 2's complement of the number FF H, stored at memory location 2000 H. Store the result in memory location 3015 H.

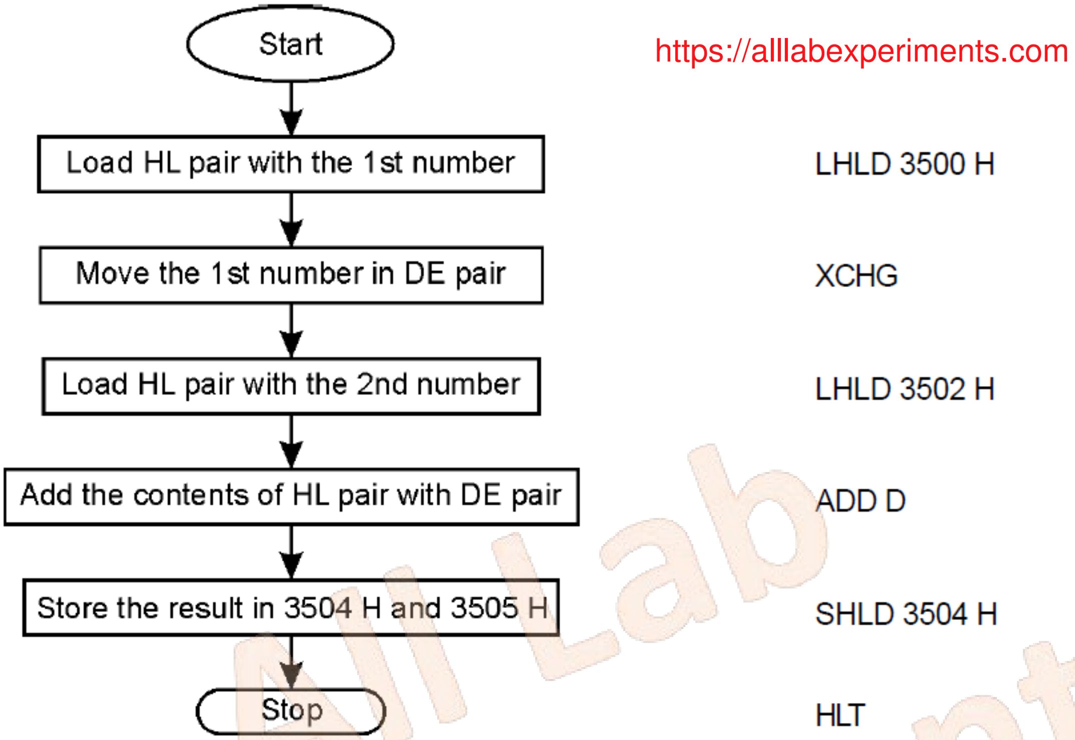
Ans. The flowchart, along with the program, is shown below:


Start
Load data FF H in ACC from memory location 2000
LDA 2000 H
Complement the data
CMA
Add 01 H to ACC
ADD, 01 H
Store Acc. in memory location 3015 H
STA 3015 H
Stop
HLT

Memory location 2000 H contains data FF H.

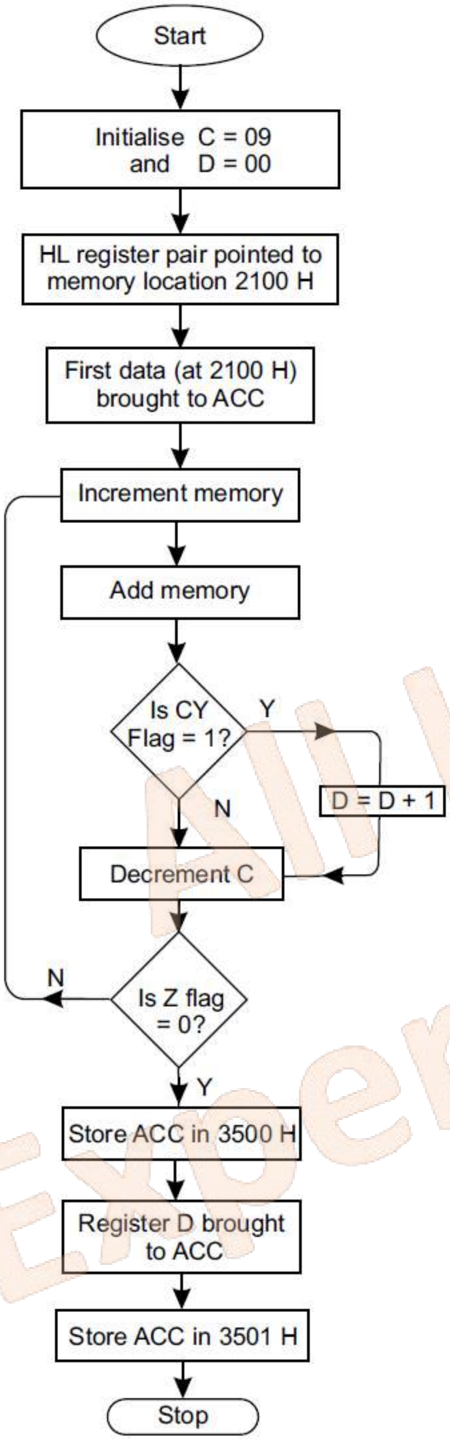
Que10: Write down the program to add two sixteen bit numbers. Draw the corresponding flowchart also.

Ans. Let the two sixteen- bit numbers are stored at memory locations 3500 H to 3503 H. The result of addition is to be stored at memory locations 3504 H and 3505 H. The flowchart and the program will look like as follow:


Start
Load HL pair with the 1st number
Move the 1st number in DE pair
Load HL pair with the 2nd number
Add the contents of HL pair with DE pair
Store the result in 3504 H and 3505 H
Stop
LHLD 3500 H
XCHG
LHLD 3502 H
ADD D
SHLD 3504 H
HLT

Que11: Ten 8-bit numbers are stored starting from memory location 2100 H. Add the numbers, store the result at 3500 H memory location and carry at 3501 H. Draw the flowchart also.

Ans. Initially, C register is stored with 09 H to take care of ten 8-bit data. Register D is initialised to 00 and stores the subsequent carry as addition is carried on. Final sum is stored at 3500 H w The corresponding flowchart is also shown. Ten numbers of 8-bit data are stored starting from memory location 2100 H.hile carry is stored at 3501 H.



```
MVI C, 09
MVI D, 00
LXI H, 2100 H
MOV A, M
* INX H
ADD M
JNC **
INR D
** DCR C
JNZ *
STA 3500 H
MOV A, D
STA 3501 H
HLT
```