Aim:

(a) Introduction to Kit and simulator.

(b) Demonstration of a simple program

(a) Introduction to Kit and simulator.

(i) Introduction to Viny tics-VMC 8501 kit:

- VMC – 8501 is a single board MICROPROCESSOR TRAINING/ DEVELOPEMNT KIT configured around the most widely used Microprocessor Intel-8085.

- The VMC – 8501 communicates with the outside world through a key board having 28 keys and seven segment hexadecimal display.

- VMC – 8501 provides 8K bytes of RAM and 8K bytes of EPROM.

- The monitor program is from 0000h to 1FFFh locations and 2000h to 3FFFh locations are available to the user to write the program and to store required data.

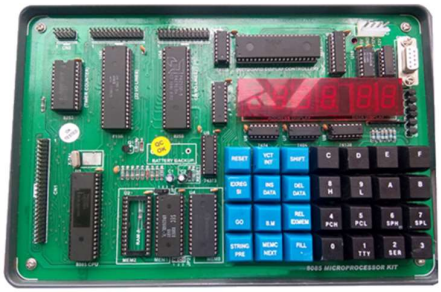
- On the board we have an 8279 chip which controls all keyboard and display operations, an 8255 to provide programmable I/O ports, an 8253 to facilitate timers and counters, RS232 compatibility to connect the kit directly to PC. Other add-on cards can also be connected to this kit.

(ii) 8085 Microprocessor Kit

- It was first designed in the year 1977 by Intel. 8085 is an 8-bit modified form of a microprocessor. It processes the binary data as per the instruction stored in memory and gives a suitable outcome.

- One can find its applications in ovens, gadgets, washing machines, and various other devices. The microprocessor kit is a single-board computer that helps in practicing 8085 microprocessors.

- The kit costs quite little and is good for the learning process. It allows one to start from a low-level language and gradually reach high-level programming.



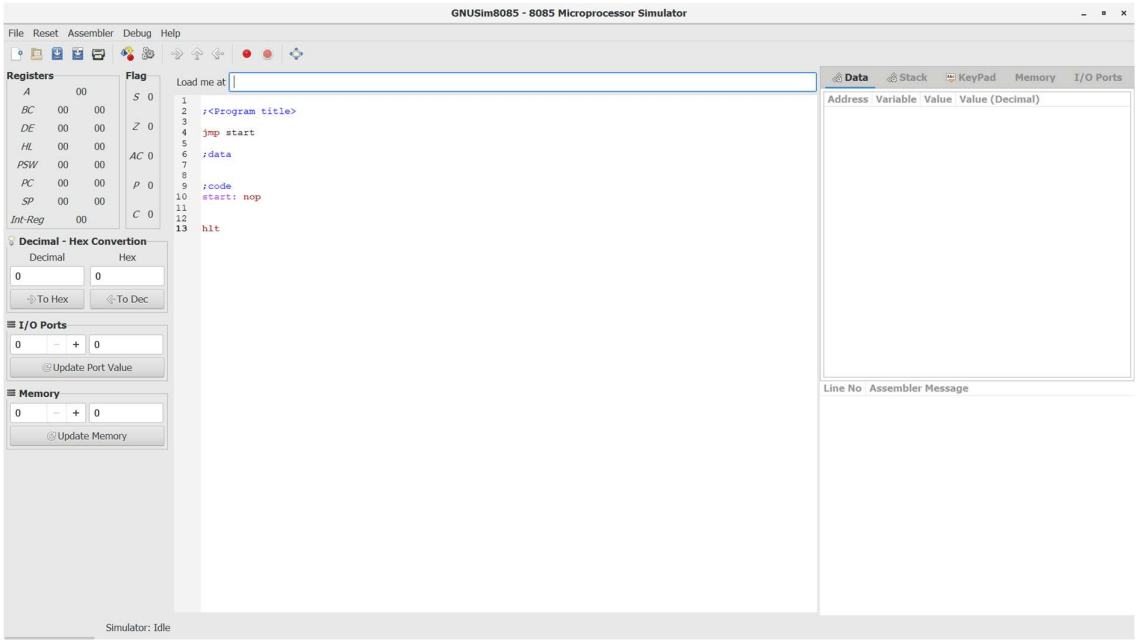
Key features of 8085 Microprocessor Trainer Kit

- Devices : 8085(Intel)
- Clock : 6.144MHz crystal
- 32KB-SRAM for user Data
- 16KB-EEPROM for Monitor Program
- 2×16 Char LCD display
- 48 Programmable I/O Pins for (2 x 8255)
- Three 16-bit programmable timer (8253/8254)
- 40-Pin FRC connector for Bus Extension
- 20-Pin FRC connector Add-on Interface from 8255
- 9-pin DB connector for UART (RS232) interface
- Built-in assembler and dis-assembler.
- 101-PC Type keyboard to enter user address/data commands
- Software Monitor for loading and executing programs with break point facility

iii) GNU Simulator 8085

- GNU Simulator 8085 is a software simulator and debugger for the Intel 8085 microprocessor.
- It allows users to develop, test, and debug software for the 8085 microprocessors without the need for physical hardware.
- It replicates the behaviour of the 8085 microprocessor and its associated peripherals, including memory, input/output ports, and interrupt controllers.
- Users can write and assemble 8085 assembly language programs using a built-in text editor or an external editor and then load the program into the simulator for testing and debugging.
- It provides a graphical user interface that allows users to view and manipulate the memory and registers of the simulated microprocessor.
- It supports single-stepping through the program and setting breakpoints for debugging purposes.

- It is available for a variety of platforms, including Windows, Linux, and macOS.
- It is released under the GNU General Public License (GPL), which means that it is free to use, modify, and distribute.
- It is a powerful tool for developing and testing software for the Intel 8085 microprocessor, and its open-source nature makes it a popular choice among hobbyists, students, and professionals alike.



(b) Demonstration of simple program.

Address	Mnemonics	Op-code	Comments
2000H	MVI A,12H	3EH	;This instruction loads accumulator with 12H
2001H		12H	
2002H	MVI B,34H	06H	;This instruction loads register-B with 34H
2003H		34H	
2004H	MOV C,B	48H	;Loads C with contents of B.
2005H	LXI H,3000H	21H	; Loads Register H with 30H and Register L with 00H
2006H		00H	
2007H		30H	
2008H	ADD B	80H	; Adds the contents of register B to accumulator and stores the result in Acc.
2009H	MOV M, A	77H	; Stores the result in accumulator to memory location specified by HL pair.
200AH	MOV D, A	57H	; Stores the result in accumulator to register-D.
200BH	HLT	76H	; Halt the execution.

Input Data: 12H and 34H

Result:

on Address 3000 H = 46 H

In register D = 46H

Practical 2

Aim:

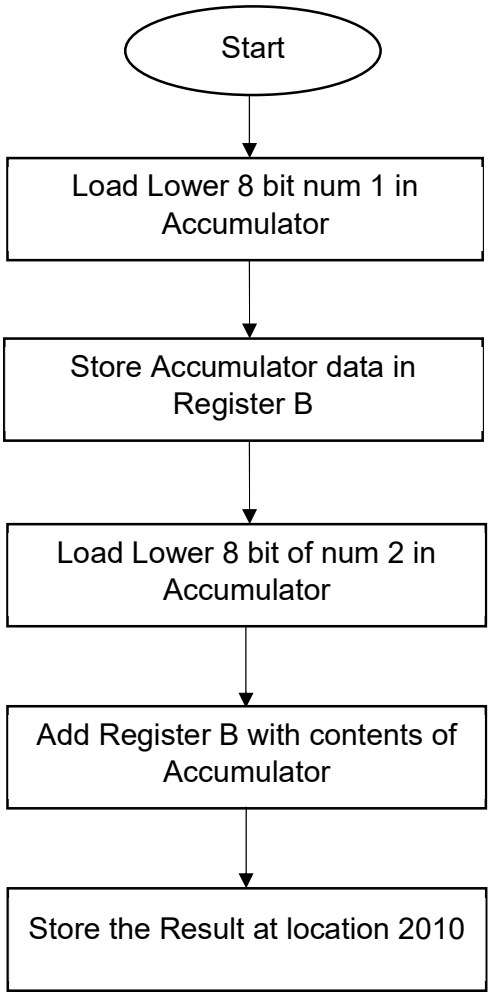
(a) Write a program to add two 16 bit numbers

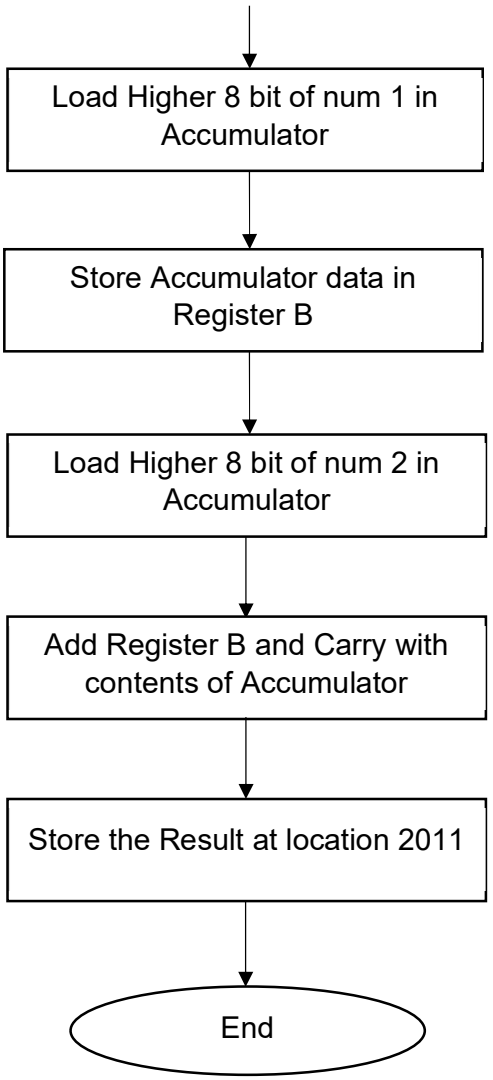
(b) Write a program to subtract two 16 bit numbers

(a) Write a program to add two 16 bit numbers

Statement: Two bit of one 16 bit number is stored at location 2001 (Low Order Bit) & 2002 (Higher Order Bit), another 16 bit number is stored at location 2003 (High Order Bit) & 2004 (Low Order Bit). Result must be stored at location 2010(Low Order Bit) & 2011 (Higher Order Bit).

Flowchart:




Load Higher 8 bit of num 1 in Accumulator
Store Accumulator data in Register B
Load Higher 8 bit of num 2 in Accumulator
Add Register B and Carry with contents of Accumulator
Store the Result at location 2011
End

Program:

Assembly Language Program	Comment
LDA 2001	Load lower 8 bit digits of number 1 in accumulator
MOV B, A	Move contents of A to B
LDA 2003	Load lower 8 bit digits of number 2 in accumulator
ADD B	Add contents of register B with contents of register A
STA 2010	Store result at location 2010
LDA 2002	Load higher 8 bit digits of number 1 in accumulator
MOV B, A	Move contents of A to B
LDA 2004	Load higher 8 bit digits of number 2 in accumulator
ADC B	Add contents of register B with Carry and Content of register A
STA 2011	Store result at location 2011
HLT	Terminate program execution

Result:

(2001) = FF

(2002) = 66

(2003) = 22

(2004) = 33

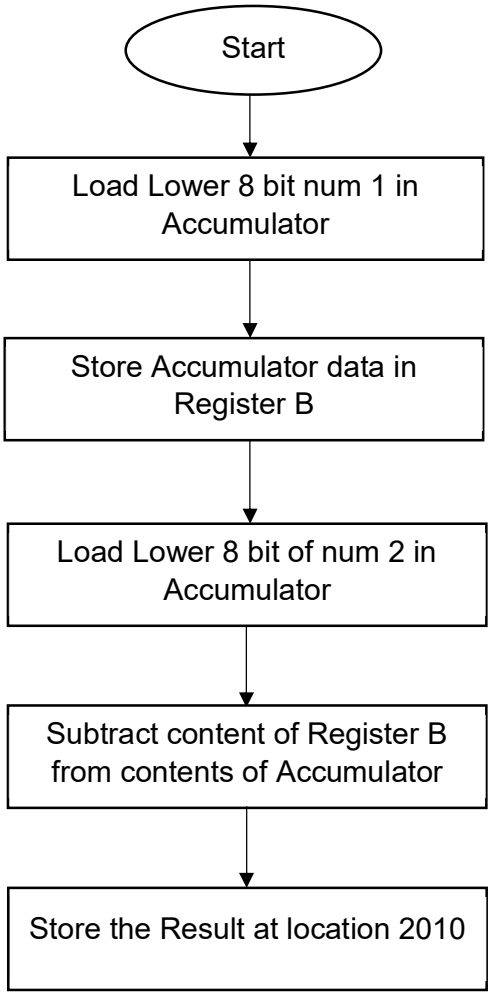
(2001) = 21

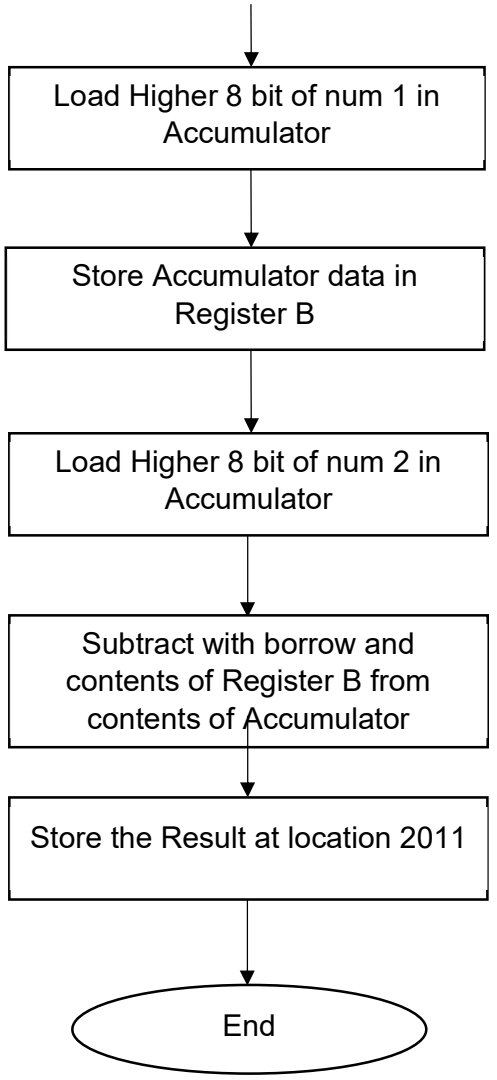
(2001) = 9A

(b) Write a program to subtract two 16 bit numbers

Statement: Two bit of one 16 bit number is stored at location 2001 (Low Order Bit) & 2002 (Higher Order Bit), another 16 bit number is stored at location 2003 (High Order Bit) & 2004 (Low Order Bit). Result must be stored at location 2010(Low Order Bit) & 2011 (Higher Order Bit).

Flowchart:




Load Higher 8 bit of num 1 in Accumulator
Store Accumulator data in Register B
Load Higher 8 bit of num 2 in Accumulator
Subtract with borrow and contents of Register B from contents of Accumulator
Store the Result at location 2011
End

Program:

Assembly Language Program	Comment
LDA 2001	Load lower 8 bit digits of number 1 in accumulator
MOV B,A	Move contents of A to B
LDA 2003	Load lower 8 bit digits of number 2 in accumulator
SUB B	Add contents of register B with contents of register A
STA 2010	Store result at location 2010
LDA 2002	Load higher 8 bit digits of number 1 in accumulator
MOV B,A	Move contents of A to B
LDA 2004	Load higher 8 bit digits of number 2 in accumulator
SBB B	Add contents of register B with Carry and Content of register A
STA 2011	Store result at location 2011
HLT	Terminate program execution

Result:

(2001) = 25

(2002) = 12

(2003) = 35

(2004) = 43

(2001) = 10

(2001) = 31

Practical 3

Aim:

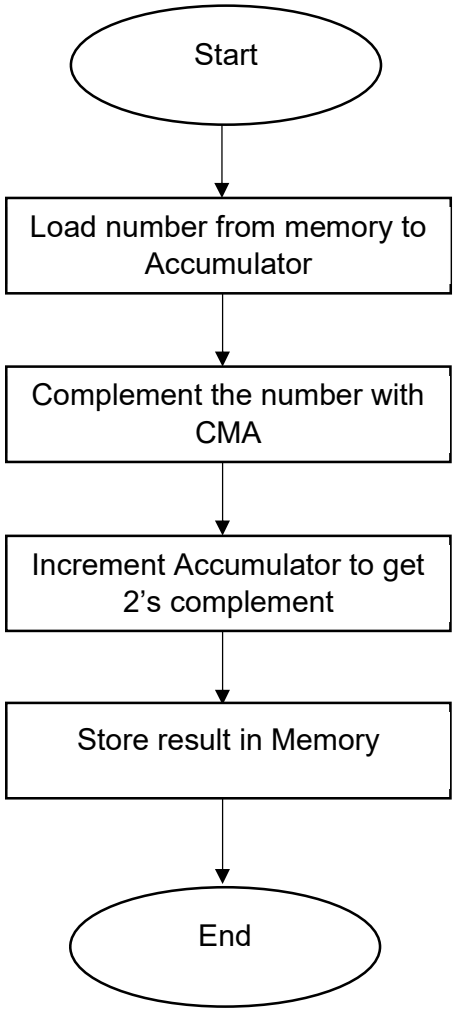
(a) Write a program to Find two's complement of number and store the answer in memory

(b) Write a program to Calculate the sum of series of numbers.

(a) Write a program to Find two's complement of number and store the answer in memory

Statement: Load data from memory in Accumulator. Find 2's complement of number and store in memory.

Flowchart:



Program:

Assembly language program	Comment
LDA 2001	Load the number to Accumulator
CMA	Complement Accumulator
INR A	Increment Accumulator
STA 2002	Store result in accumulator in location 2002
HLT	Terminate program execution

Result:

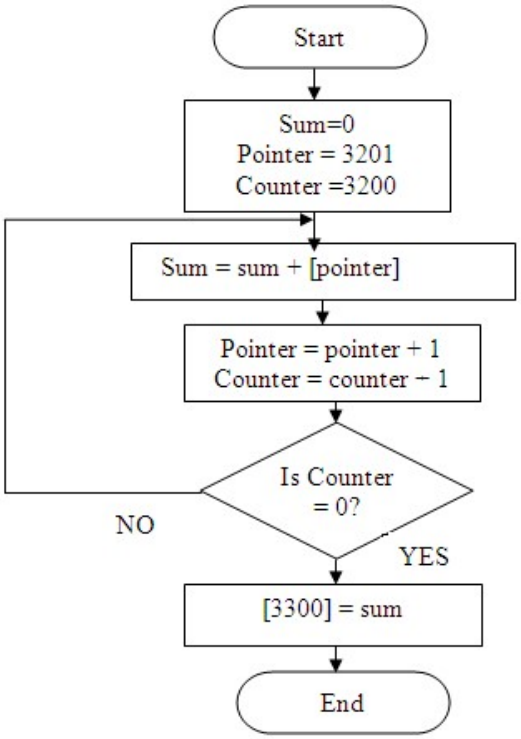
(2001) = FF

(2002) = 01

(b) Write a program to Calculate the sum of series of numbers.

Statement: Calculate the sum of series of numbers. The length of the series is in memory location 3200H and the series begins from memory location 3201H.\

Flowchart:



Program:

Assembly language program	Comments
LDA 3200	
MOV C, A	Initialize Counter
SUB A	sum = 0
LXI H,3201	Initialize pointer
BACK: ADD M	SUM = SUM + data
INX H	increment pointer
DCR C	Decrement counter
JNZ BACK	if counter 0 repeat
STA 3300	Store sum
HLT	Terminate program execution

Result:

3200H = 04H

3201H = 10H

3202H = 45H

3203H = 33H

3204H = 22H

10 +45 + 33 + 22 = AA H

3300H = AA H

Practical 4

Aim:

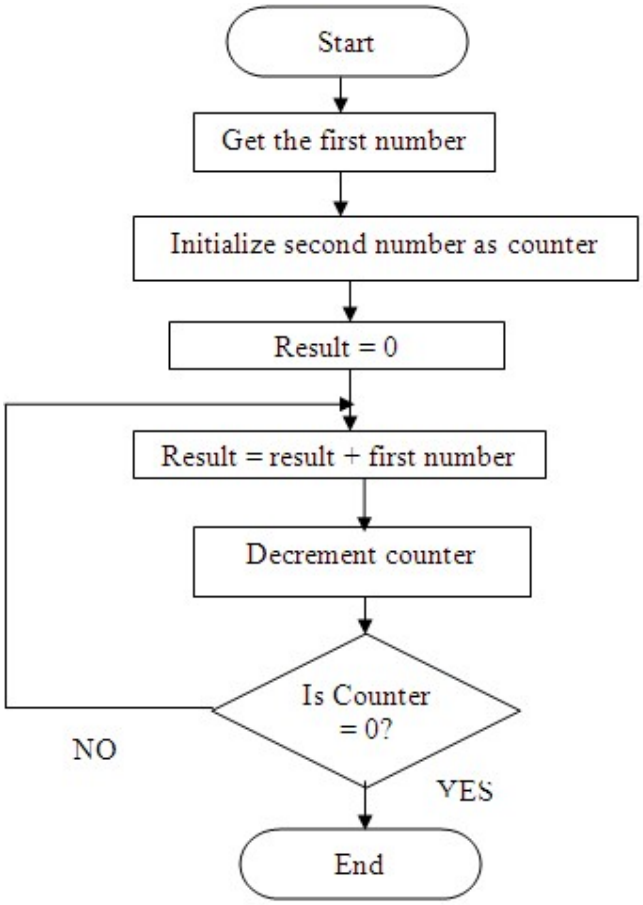
(a) Write a program to multiply two 8 bit numbers.

(b) Write a program to divide two 8 bit no and obtain quotient and remainder.

(a) Write a program to multiply two 8 bit numbers.

Statement: Multiply two 8-bit numbers stored in memory locations 2200H and 2201H by repetitive addition and store the result in memory locations 2300H and 2301H.

Flowchart:



Program:

Assembly Language Program	Comments
LDA 2200H	
MOV E, A	Copy contents of A in E
MVI D, 00	Get the first number in DE register pair
LDA 2201H	
MOV C, A	Initialize counter
LX I H, 0000 H	Result = 0
BACK: DAD D	Result = result + first number
DCR C	Decrement count
JNZ BACK	If count 0 repeat
SHLD 2300H	Store result
HLT	Terminate program execution

Result:

(2200H) = 03H

(2201H) = B2H

B2H + B2H + B2H = 216H

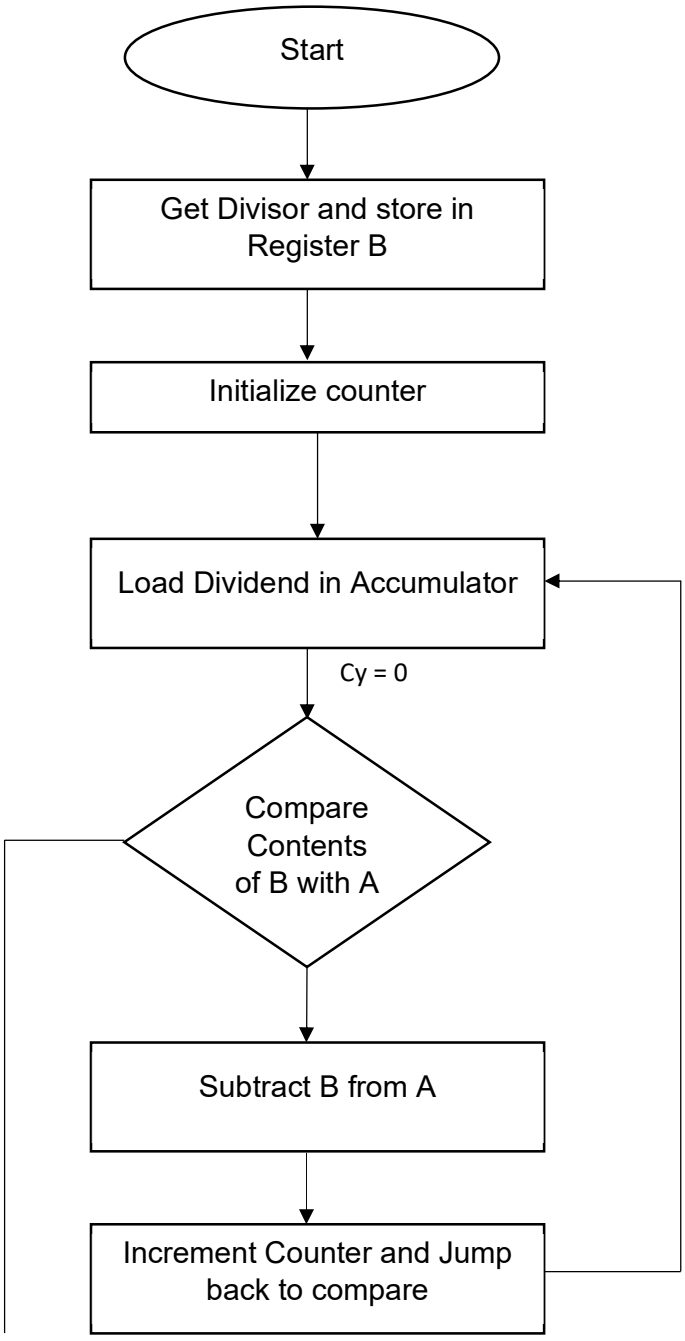
(2300H) = 16H

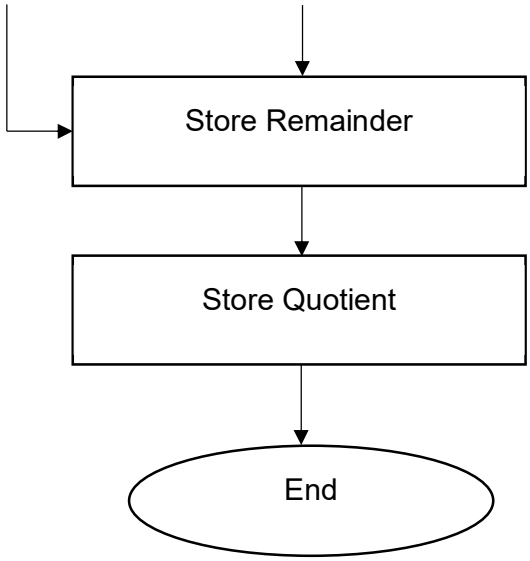
(2301H) = 02H

(b) Write a program to divide two 8 bit no and obtain quotient and remainder.

Statement: Divide 2 8-bit numbers stored at 2000 (Divisor) and 2001 (Dividend). Store quotient at 2003 and remainder at 2002

Flowchart:





Program:

Assembly language Program	Comments
LXI H, 2000	Load 2000 in Register pair HL
MOV B, M	Load divisor from location pointed by HL pair
MVI C, 00	Initialize counter
INX H	Increment register pair HL
MOV A, M	Load dividend in A
label2: CMP B	Compare contents of A with B
JC label1	If Cy = 1 them jump to label 1
SUB B	Subtract contents of B from A
INR C	Increment counter
JMP label2	Jump to label 2
label1: STA 2002	Store remainder at 2002
MOV A, C	Move counter to A
STA 2003	Store Quotient at 2003
HLT	Terminate program execution

Result:

(2000) = 02H

(2001) = 05H

05 – 02 = 03

03 – 02 = 01 (remainder)

(2002) = 01H

(2003) = 02H

Practical 5

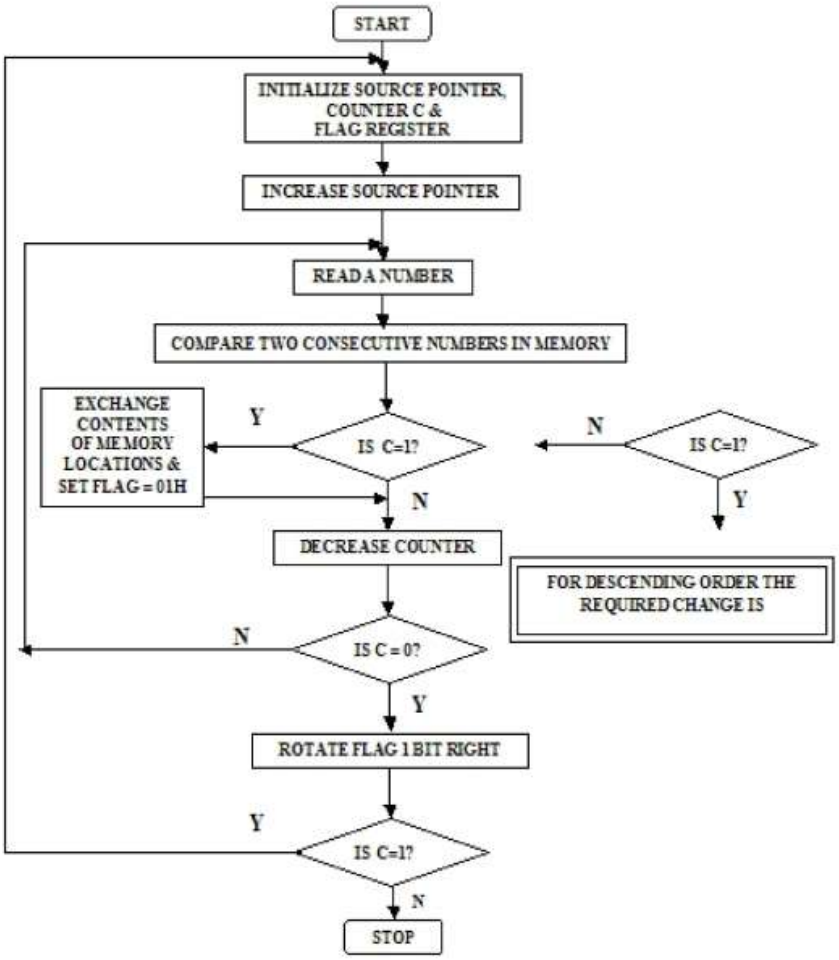
Aim:

(a) Write a program to Arrange the array in ascending order.
(b) Write a program to count number of zero, positive and negative numbers from an array.

(a) Write a program to Arrange the array in ascending order.

Statement: Arrange the array, the total size of array is store at 2000. The array starts from 2001.

Flowchart:



Program:

Assembly language program	Comments
START: LXI H,2000	Load the address where size of array is stored
MVI D, 00	Initialize counter
MOV C, M	Load size of array in Register C
DCR C	Decrement C
INX H	Increment register pair HL
LOOP: MOV A, M	Move contents of Address pointed by HL to accumulator
INX H	Increment register pain HL
CMP M	Compare with data stored at next location
JC SKIP	If Cy = 1 then jump to SKIP
MOV B, M	Move content of M to B
MOV M, A	Move content of A to M
DCX H	Decrement register pair HL
MOV M,B	Move content of B to M
INX H	Increment register pair HL
MVI D, 01H	Move 01H in D
SKIP: DCR C	Decrement C
JNZ LOOP	If z != 0 then jump to LOOP
MOV A,D	Move content of D to A
RRC	Rotate Accumulator right through carry
JC START	If Cy = 1 then jump to START
HLT	Terminate program execution

Result:

(2000) = 5 (size of array)

(2001) = 5

(2002) = 4

(2003) = 3

(2004) = 2

(2005) = 1

After ascending the array

(2001) = 1

(2002) = 2

(2003) = 3

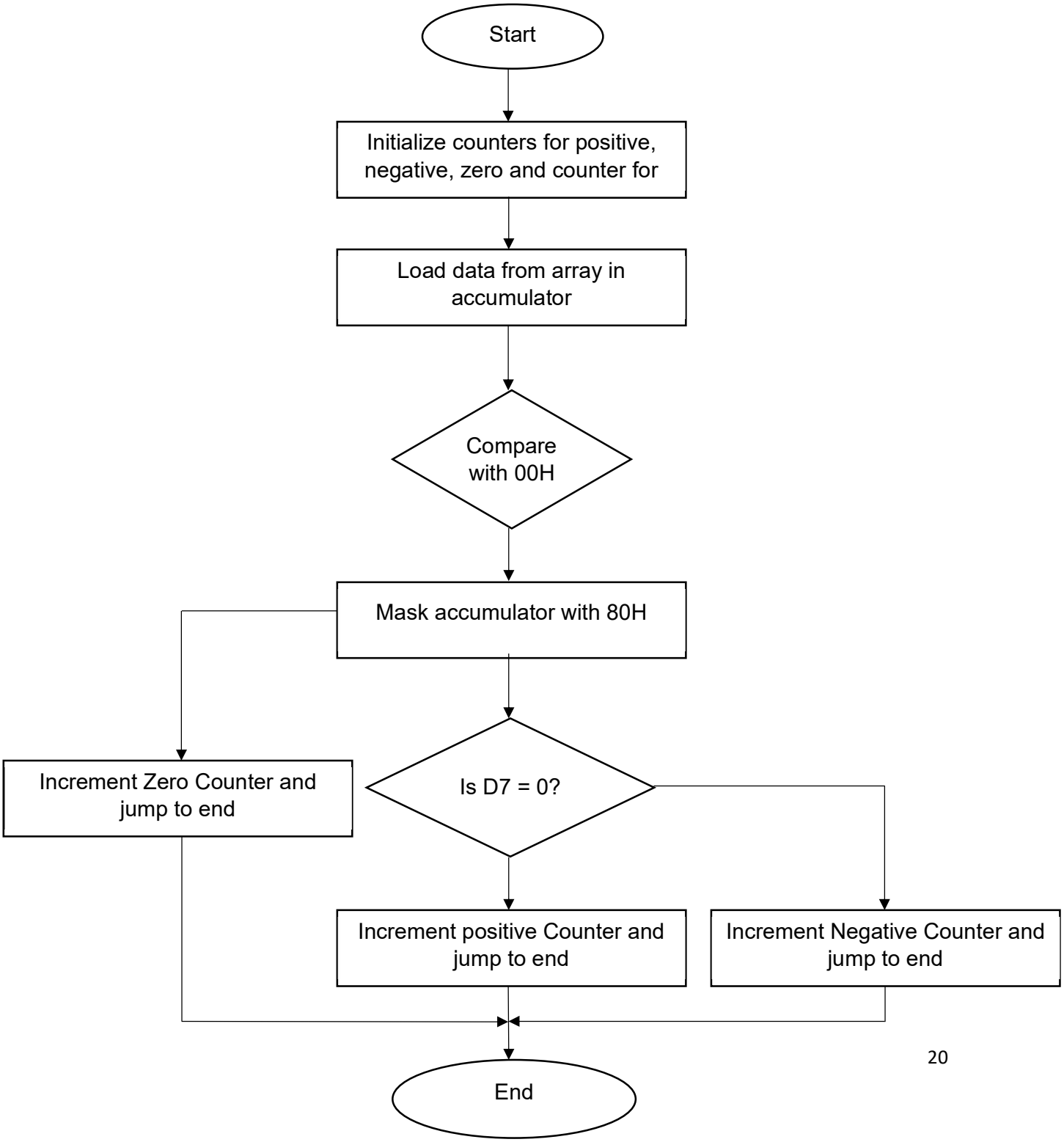
(2004) = 4

(2005) = 5

(b) Write a program to count number of zero, positive and negative numbers from an array.

Statement: Array starts from location 2000. Store count of positive number in D, Store count of negative number in B, and store count of zeros in E. Load to size of array in C.

Flowchart:



Program:

Assembly language program	Comments
LXI H, 2000	Load the starting address of array
MVI C, 05	Load total elements in array
MVI D, 00	Positive number counter
MVI B, 00	Negative number counter
MVI E, 00	Zero counter
BEGIN: MOV A, M	Move contents of M to A
CPI 00H	Compare with 00H to check for Zero
JZ ZERO	If Z = 0 then jump to ZERO
ANI 80H	Mask all bits of accumulator except D7
JNZ NEG	If Z = 0 then jump NEG
INR D	Increment counter for Positive number
JMP LAST	Jump to LAST
ZERO: INR E	Increment counter for Zero
JMP LAST	Jump to LAST
NEG: INR B	Increment counter for Negative number
LAST: INX H	Increment register pair HL for next number in array
DCR C	Decrement counter for size of array
JNZ BEGIN	If C != 0 then jump to BEGIN
HLT	Terminate program execution

Result:

(2000) = 80H

(2001) = 90H

(2002) = 0

(2003) = 3

(2004) = 4

D = 2 (positive number)

B = 2 (negative number)

D = 1 (Zero)

Practical 6

Aim:

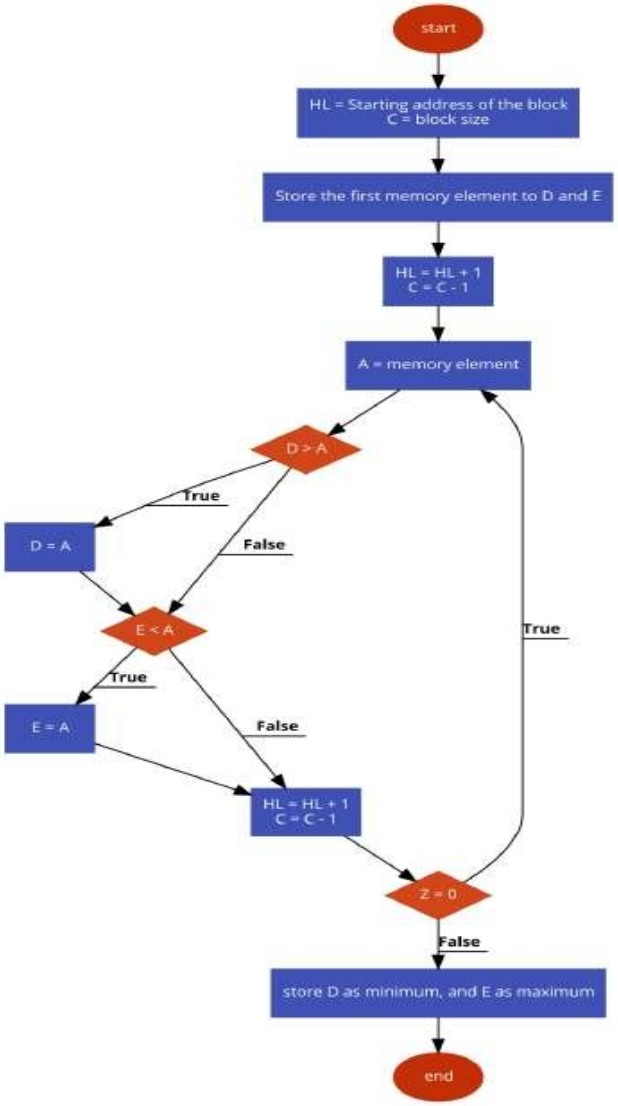
(a) Write a program to find the maximum and minimum number from given numbers.

(b) Write a program to Count Number of one's and zero's in a number.

(a) Write a program to find the maximum and minimum number from given numbers.

Statement: Write 8085 Assembly language program to find the maximum and minimum number in a block often 8-bit numbers.

Flowchart:



Program :

Assembly language Program	Comments
LXI H,8000H	Load the initial address
MVI C,0AH	Load the count of numbers
MOV D,M	Load the first number from memory
MOV E,D	Also load the first number to E
INX H	Point to next location
DCR C	Decrease the count
LOOP :MOV A,M	Load the number from memory to A
CMP D	Compare D with A
JNC SKIP	If CY = 0, A is not smaller
MOV D,A	Update D with A
SKIP :CMP E	Compare E with A
JC DO	if CY = 1, A is not larger
MOV E,A	Update E with A
DO :INX H	Point to next location
DCR C	Decrease C by1
JNZ LOOP	Go to loop
LXI H,8050H	Point to destination address
MOV M,D	Store the smallest number
INX H	Point to next location
MOV M,E	Store the largest number
HLT	Terminate the program

Input:

(8000) : 55H

(8001) : 22H

(8002) : 88H

(8003) : 77H

(8004) : 11H

(8005) : 99H

(8006) : 44H

(8007) : AAH

(8008) : 33H

(8009) : 66H

Result :

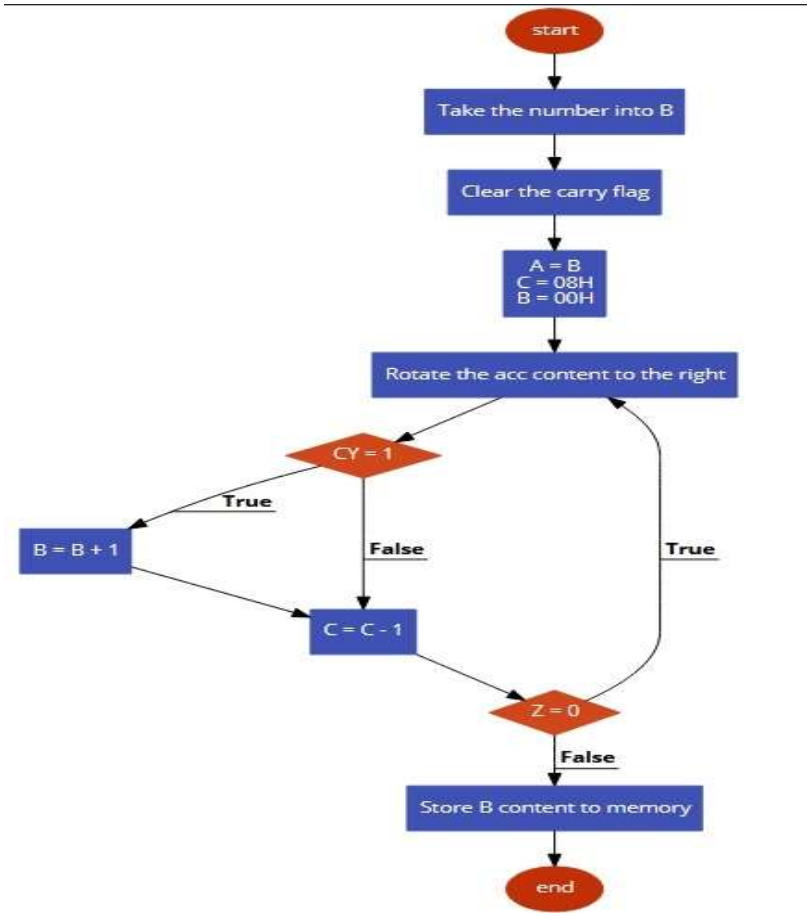
(8050) = 11H

(8051) = AAH

(b) Write a program to Count Number of one's and zero's in a number.

Statement: Write 8085 Assembly language program to count number of one's and zero's in a number.

Flowchart:



Program :

Label	Assembly Lang. Program	Comments
	LXI H, 2100H	Initialize the HL pair to point to the firs data
	MVI C,08H	Set the counter
	MVI D,00H	Initialize the Reg. B
	MOV A,M	Transfer toAccumulator
LABEL1:	RLC	Rotate left with carry (Acc. data)
	JNC LABEL2	If carry is not set then decrement the counter
	INR D	No. of 1's
LABEL2:	DCR C	Decrement the counter
	JNZ LABEL1	If counter is not zero then go to Acc. For rotete the Acc. data
	MVI A,08H	Initialize the Reg. A
	SUB D	No. of 0's
	STA 2101H	Store the ‘0’s
	MOV A,D	Transfer the data
	STA 2102H	Store the ‘1’s
	HLT	END the program

INPUT: 55H

Result:

Number of ‘0’ = 4

Number of ‘1’ = 4

(2101) = 4H

(2102) = 4H

Practical 7

Aim:

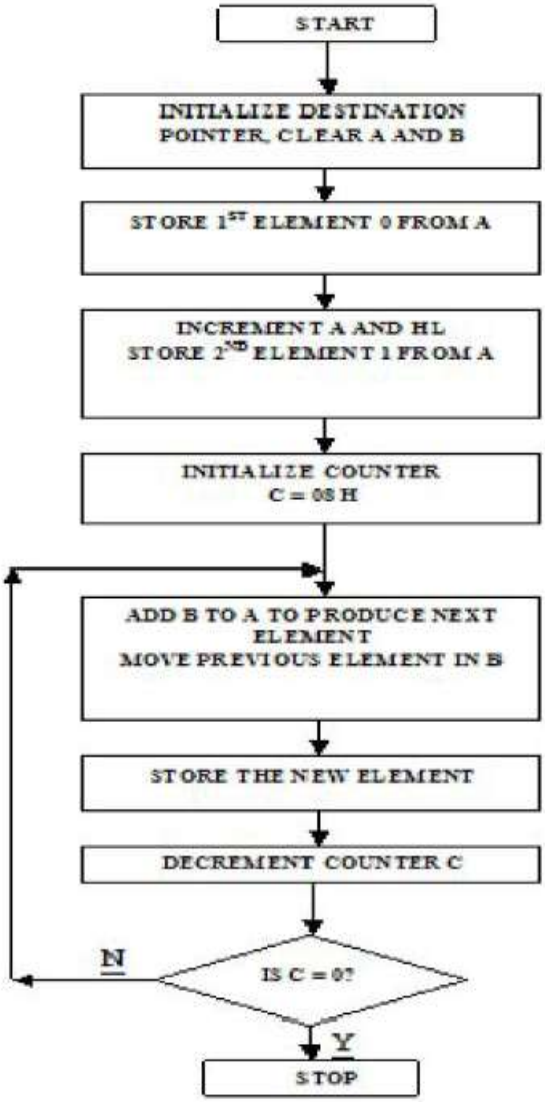
(a) Write a program to generate Fibonacci number.

(b) Write a program to transfer 16 bytes of data stored at one location to another location

a) Write a program to generate Fibonacci number.

Statement: Write an assembly language program to generate Fibonacci series.

Flowchart:



Program:

Assembly Language Program	Comments
LXI H,3000H	Store the Fibonacci series
MVI D, 08H	Initialize counter
MVI B, 00	Initialize variable to store previous number
MVI C, 01	Initialize variable to store current number
MOV M,A	Store 00 at 3000H
INX H	
MOV M,B	Store 01 at 3001H
MOV A, B	[Add two numbers]
BACK: ADD C	[Add two numbers]
INX H	
MOV M,A	Store next element of series
MOV B, C	Current number is now previous number
MOV C, A	Save result as a new current number
DCR D	Decrement count
JNZ BACK	if count 0 go to BACK
HLT	Stop

Result :

(3000) = 00H

(3001) = 01H

(3002) = 01H

(3003) = 02H

(3004) = 03H

(3005) = 05H

(3006) = 08H

(3007) = 0DH

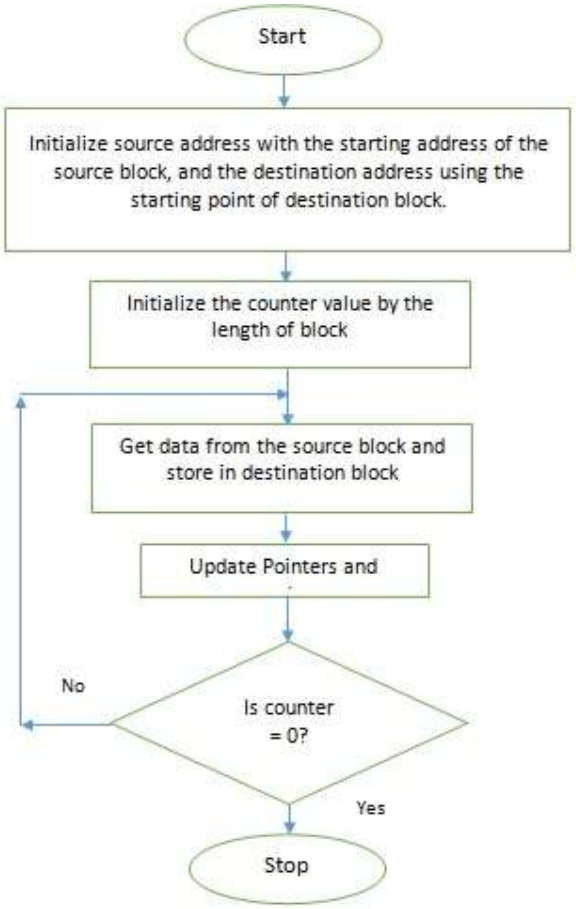
(3008) = 15H

(3009) = 22H

b) Write a program to transfer 16 bytes of data stored at one location to another location

Statement: Write an assembly language program to transfer 16 bytes of data stored at one location to another location.

Flowchart:



Program :

Assembly Lang. Program	Comments
LXI H, 8050H	Initialize source memory pointer 300FH
LXI D, 8070H	Initialize destination memory pointer
MVI C, 10H	Initialize counter
LABEL 1 : MOV A,M	Get byte from source memory block
STAX D	Store byte in the destination memory block
INX H	Decrement source memory pointer

INX D	Decrement destination memory pointer
DCR C	Decrement counter
JNZ LABEL1	If counter not 0 repeat
HLT	Stop execution

Result with inputs as follows :

Input		Output	
Address	**Value**	**Address**	**Value**
8050H	00	8070H	00
8051H	11	8071H	11
8052H	22	8072H	22
8053H	33	8073H	33
8054H	44	8074H	44
8055H	55	8075H	55
8056H	66	8076H	66
8057H	77	8077H	77
8058H	88	8078H	88
8059H	99	8079H	99
805AH	AA	807AH	AA
805BH	BB	807BH	BB
805CH	CC	807CH	CC
805DH	DD	807DH	DD
805EH	EE	807EH	EE
805FH	FF	807FH	FF

Practical 8

Aim:

(a) Write a program to Generate and display BCD up counter with frequency 1 Hz.

(b) Write a program to Generate and display the content of decimal counter.

a) Write a program to Generate and display BCD up counter with frequency 1 Hz.

Statement: Write a program for displaying BCD up counter. Counter should count numbers from 00 to 99H and it should increment after every 1 sec.

Assume operating frequency of 8085 equal to 3MHz. Display routine is available.

Operating frequency=3MHz

Time of 1 T state=$1/3MHz = 0.333\mu sec$

Required time/time required for 1 T-state =3×10^6=1/0. 333μsec

Let us take multiplier count = 3

No. of T state required by inner loop= $3\times10^6 / 3=1\times10^6$

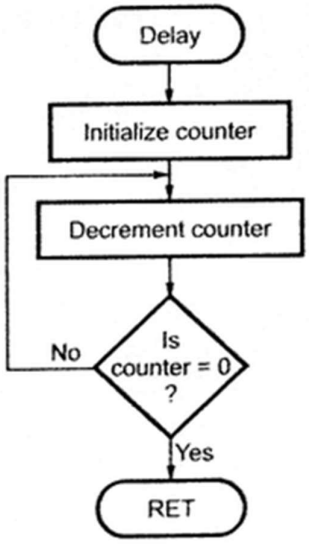
$1\times10^6 = 10 + (count-1) \times 24+ 21$

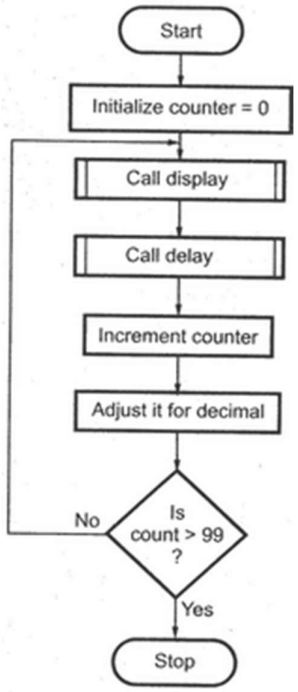
Count= $((1\times10^6- 31)/24) +1>>41666_{10}$

Count = 41666_{10}

=A2C2H

Flowchart:





Program :

Assembly Language Program	**Comments**
LXI SP, 27FFH	Initialize stack pointer
MVI C, 00H	Initialize counter
BACK: CALL Display	Call display subroutine
CALL Delay	Call delay subroutine
MOV A, C	
ADI A, 01	Increment counter
DAA	Adjust it for decimal
MOV C,A	Store count
CPI ,00	Check count is > 99
JNZ BACK	If not, repeat
HLT	Stop

Delay Routine :

Assembly Language Program	Comments
Delay:MVI B, Multiplier count	Initialize multiplier count
BACK 1LXI D, Initialize Count	
BACK:DCX D	Decrement count
MOV A, E	
ORA D	Logically OR D and E
JNZ BACK	If result is not a, repeat
DCR B	Decrement multiplier count
JNZ BACK 1	If not zero, repeat

Result :

The numbers are storing into memory location 0000H.

(b) Write a program to Generate and display the content of decimal counter.

Statement: Write assembly language program to with proper comments for the following: To display decimal decrementing counter (99 to 00) at port 05 H with delay of half seconds between .each count. Write as well the delay routine giving delay of half seconds. Operating frequency of microprocessor is 3.072 MHz. Neglect delay for the main program.

Operating frequency=3.072MH_Z

Time of 1 T state=$1/3.072MH_Z = 3.2552\times10^{-7}$

Number of T-states required=$0.5sec/3.2552\times10^{-7}=1.536\times10^{6}$

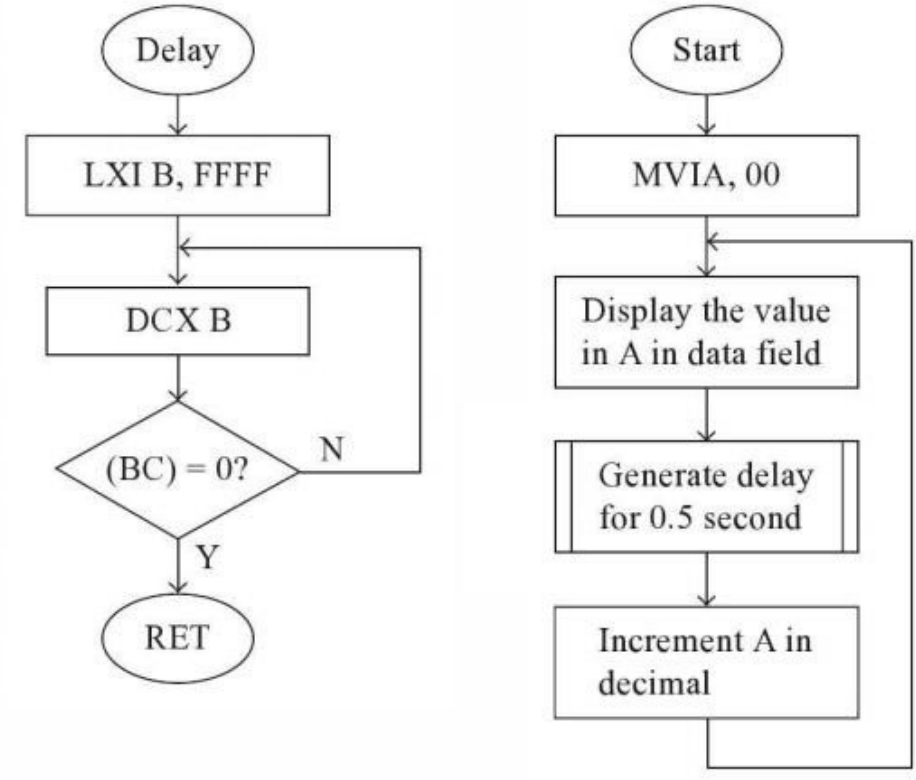
1.536×10^{6} = 10 + (count-1) ×24+21

Count= $((1.536\times10^{6}- 31)/24) +1$

Count = 63999.708_{10}

=FA00H

Flowchart:



Program:

Assembly Language Program	Comments
MVI C, 99H	Initialize counter
BACK: MOV A, C	
ANI OF	Mask higher nibble
CPI OF	
JNZ SKIP	
MOV A, C	
SUI 06	Subtract 6 to adjust decimal count
MOV D, A	
SKIP: MOV A, C	
OUT 05	send count on output port
CALL Delay	Wait for 0.5 seconds
DCR C	decrement count
MOV A, C	
CPI FF	
JNZ BACK	If not zero, repeat
HLT	Stop execution

Delay subroutine:

Assembly Language Program	COMMENT
Delay: LXI D, Count	
Back: DCX D	6 T-states
MOV A, D	4 T-states
ORA E	4 T-states
JNZ Back	10 T-states
RET	

Result :

The numbers are displaying from memory location 0000H.

Practical 9

Aim:

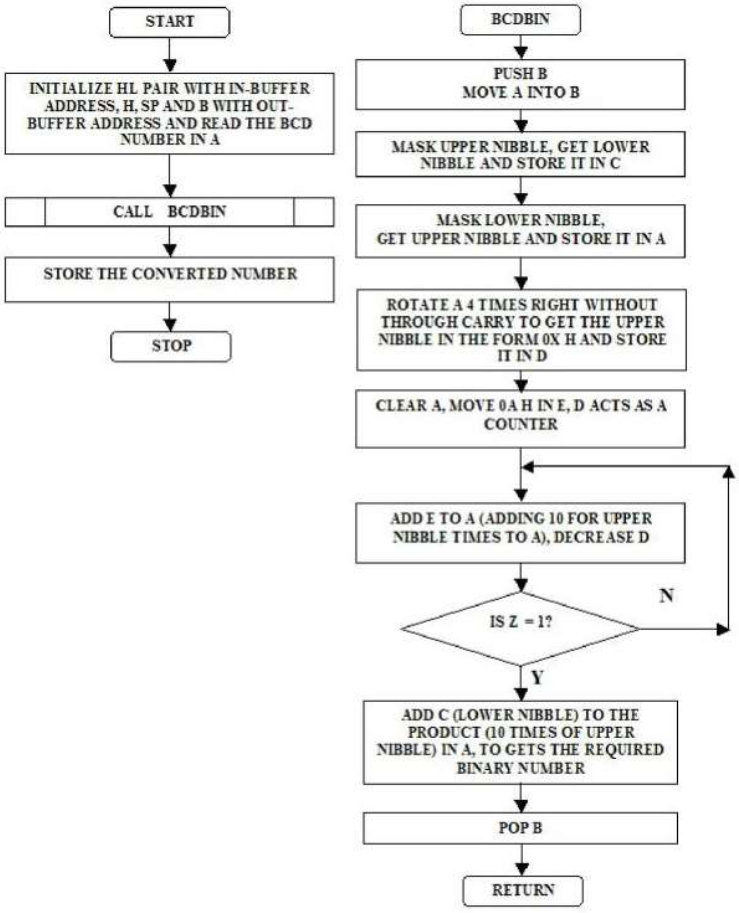
(a) Write a program to convert BCD into equivalent Binary.

(b) Write a program to convert BCD into 7 segment code.

(a) Write a program to convert BCD into equivalent Binary.

Statement: Convert a 2-digit BCD number stored at memory address 2200H into its binary equivalent number and store the result in a memory location 2300H.

Flowchart:



Program:

Assembly Language Program	COMMENT
LDA 2200H	Get the BCD number
MOV B, A	Save it
ANI OFH	Mask most significant four bits
MOV C, A	Save unpacked BCDI in C register
MOV A, B	Get BCD again
ANI FOH	Mask least significant four bits
RRC	Convert most significant four bits into unpacked BCD2
RRC	
RRC	
RRC	
MOV B, A	Save unpacked BCD2 in B register
XRA A	Clear accumulator (sum = 0)
MVI D, 0AH	Set D as a multiplier of 10
Sum: ADD D	Add 10 until (B) = 0
DCR B	Decrement BCD by one
JNZ SUM	Is multiplication complete? i if not, go back and add again
ADD C	Add BCD1
STA 2300H	Store the result
HLT	Terminate program execution

Result :

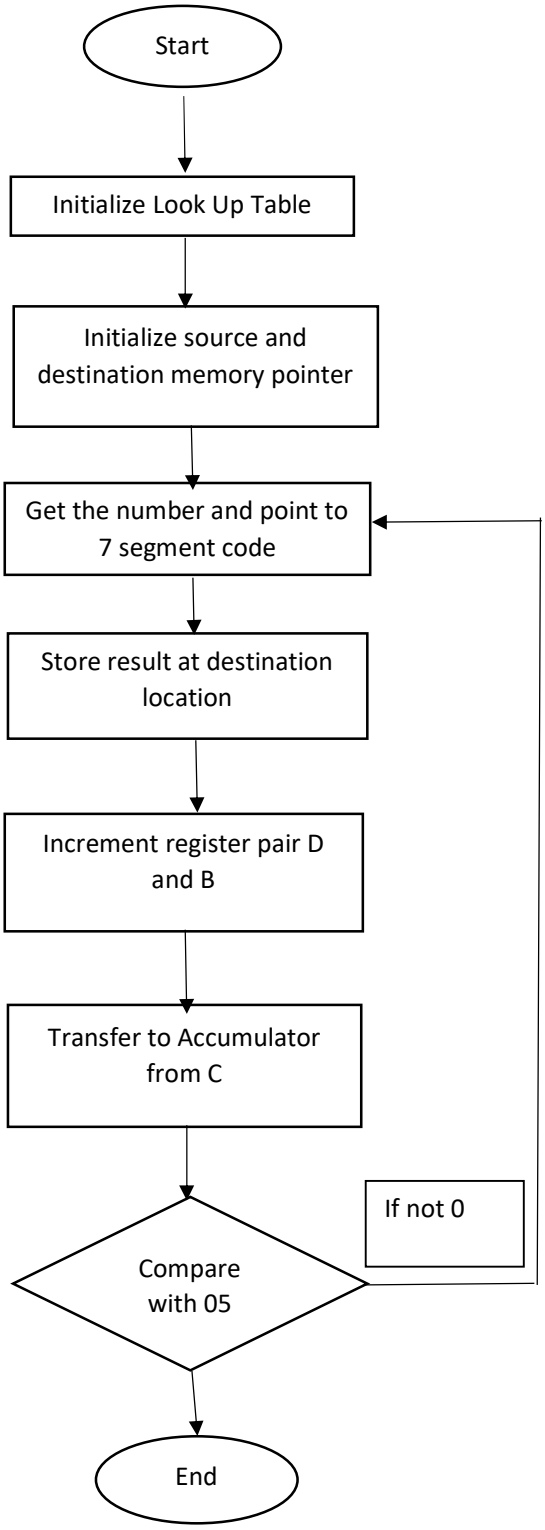
(2200)H: 52

(2300)H: 34

(b) Write a program to convert BCD into 7 segment code.

Statement: Write a main program and a conversion subroutine to convert the five binary numbers stored from 3000H into its equivalent BCD number. Store the result from memory location 3100H**.**

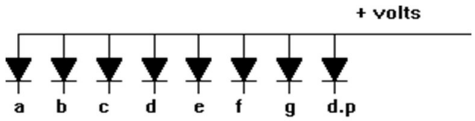
Flowchart:

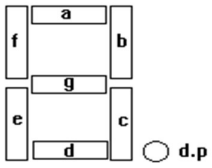


Program :

Assembly Language Program	Comments
LXI H, 3200H	Initialize lookup table pointer
LXI D, 3000H	Initialize source memory pointer
LXI B, 3100H	Initialize destination memory pointer
BACK: LDAX D	Get the number
MOV L, A	A point to the 7-segment code
MOV A, M	Get the 7-segment code
STAX B	Store the result at destination memory location
INX D	Increment source memory pointer
INX B	Increment destination memory pointer
MOV A, C	
CPI O5H	Check for last number
JNZ BACK	If not repeat
HLT	End of program

Result :





Practical 10

Aim:

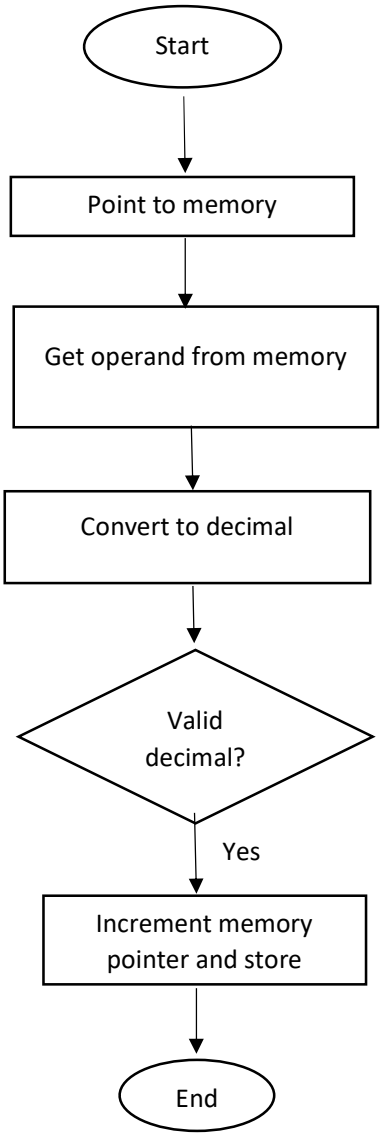
(a) Write a program to convert ASCII to decimal.

(b) Write a program to alter the contents of the flag register in 8085

(a) Write a program to convert ASCII to decimal.

Statement: Convert the ASCII number in memory to its equivalent decimal number.

Flowchart:



Program:

Assembly Language Program	Comments
LXI H, 3000H	Point to data
MOV A, M	Get operand
SUI 30	convert to decimal
CPI 0A	Check whether it is valid decimal number
JC LOOP	yes, store result
MVI A, FF	No, make result=FF
LOOP: INX H	
MOV M, A	(A) = (3001)H
HLT	

Result:

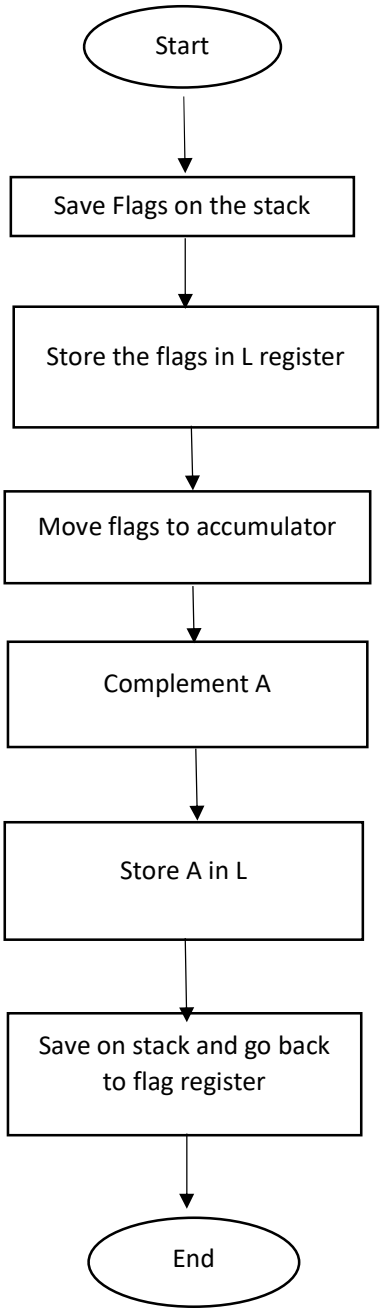
(3000)H: 36

(3001)H: 06

(b) Alter the contents of the flag register in 8085.

Statement: Write a set of instructions to alter the contents of flag register in 8085.

Flowchart:



Program:

Assembly Language Program	COMMENT
PUSH PSW	Save flags on stack
POP H	Retrieve flags in 'L'
MOV A, L	Flags in accumulator
CMA	Complement accumulator
MOV L, A	Accumulator in 'L'
PUSH H	Save on stack
POP PSW	Back to flag register
HLT	Terminate program execution

Result :

F: 10110100

F: 01001011