KERNEL MASTERS Lab Assignment

NAME: VIGNESH.B	KMID: KM40BESD01

ALP LAB Assignments:

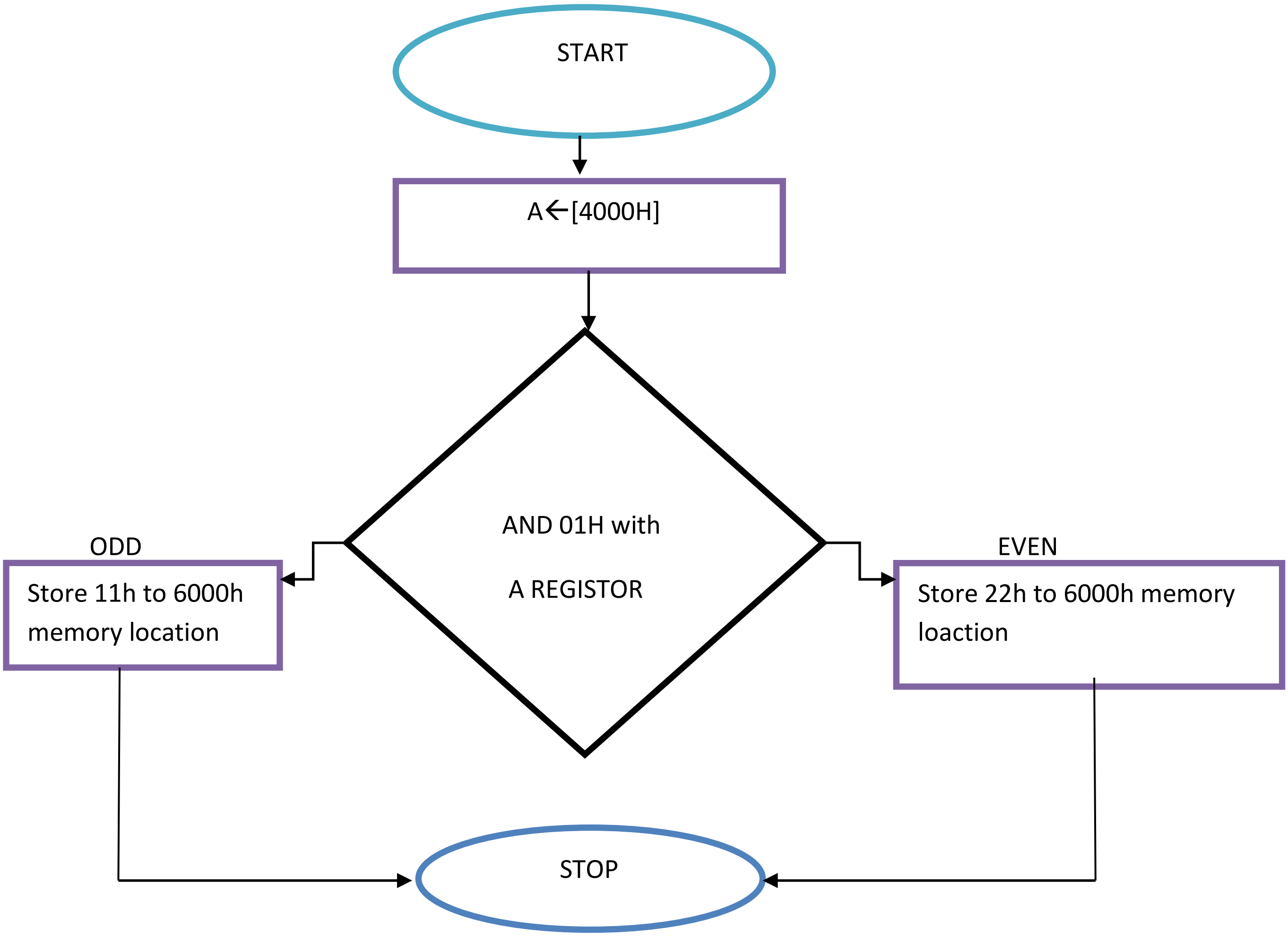
1. To find Odd No or Even No:

Write an ALP to find given number is odd or even and load number in 4000H memory location the result stored in 6000H. If even, store 22H at memory location 6000 otherwise store 11H at memory location 6000.

STEP 1: Pseudo Code

- Load data from the memory location 4000h to microprocessor
- Check the loaded data Even or Odd.
- LSB bit indicates the data odd or even.
- So perform AND operation with the data.
- If the result is 00 then its EVEN else its ODD.
- If its even then store 22h data to the 6000h memory location.
- And if the data is odd then store 11H data to the 6000h memory location.

STEP 2: Flow Chart

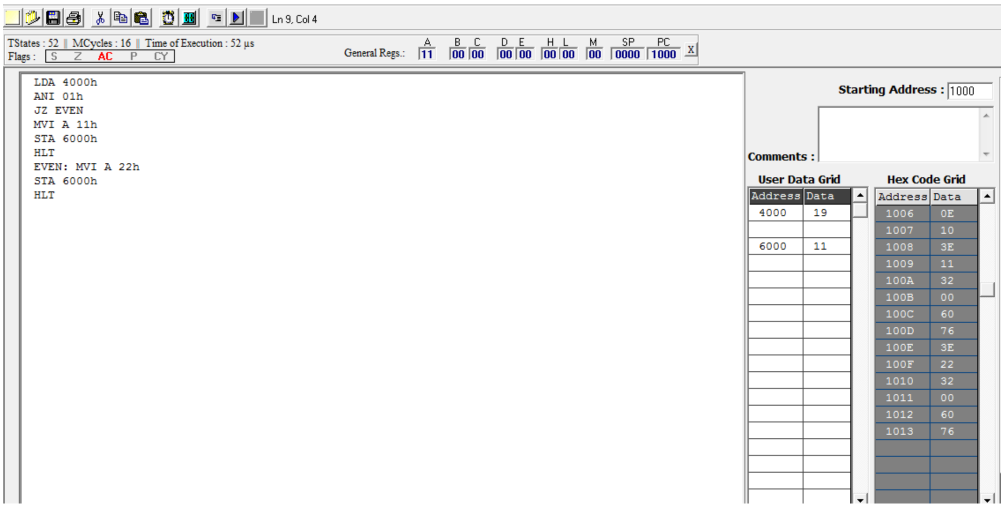

START
A←[4000H]
AND 01H with
A REGISTOR
ODD
Store 11h to 6000h memory location
EVEN
Store 22h to 6000h memory loaction
STOP

STEP 3: ALP PROGRAM

Memory address	Hexa code	Label	Opcode	Operand	Comments
1000h	XX		LDA	4000H	A←[4000H]
1001h	00H				
1002h	40H				
1003h	XX		ANI	01H	A←A (AND) 01H
1004h	01H		JZ	EVEN	Jump to EVEN label if zero flag set Zf=1
1005h	XX		MVI	A,11H	A←11H
1007H	22H				
1008H	XX		STA	6000H	[6000H]←A
1009H	00H				
100AH	60H				
100BH	XX		HLT		EXIT
100CH	XX	EVEN:	MVI	A,22H	A←22H
100DH	11H				
100EH	XX		STA	6000H	[6000H]←A
100FH	00H				
1010H	60H				
1011H	XX		HLT		EXIT

STEP 4: EXECUTION IN SIMULATOR

IF ODD NUMBER ENTERED:



TStates : 52 ‖ MCycles : 16 ‖ Time of Execution : 52 µs

Flags : S Z AC P CY

General Regs.:

A	B	C	D	E	H	L	M	SP	PC
11	00	00	00	00	00	00	00	0000	1000

```
LDA 4000h
ANI 01h
JZ EVEN
MVI A 11h
STA 6000h
HLT
EVEN: MVI A 22h
STA 6000h
HLT
```

Starting Address : 1000

Comments :

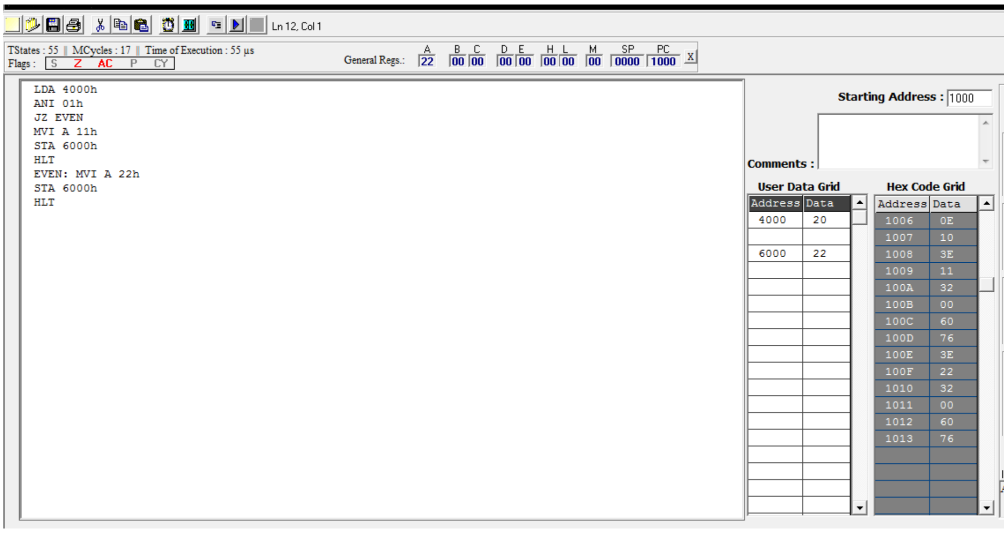
User Data Grid

Address	Data
4000	19
6000	11

Hex Code Grid

Address	Data
1006	0E
1007	10
1008	3E
1009	11
100A	32
100B	00
100C	60
100D	76
100E	3E
100F	22
1010	32
1011	00
1012	60
1013	76

IF EVEN NUMBER ENTERED:



Ln 12, Col 1

TStates : 55 | MCycles : 17 | Time of Execution : 55 µs

Flags : S Z AC P CY

General Regs.: A 22 | B 00 | C 00 | D 00 | E 00 | H 00 | L 00 | M 00 | SP 0000 | PC 1000

```
LDA 4000h
ANI 01h
JZ EVEN
MVI A 11h
STA 6000h
HLT
EVEN: MVI A 22h
STA 6000h
HLT
```

Starting Address : 1000

Comments :

User Data Grid

Address	Data
4000	20
6000	22

Hex Code Grid

Address	Data
1006	0E
1007	10
1008	3E
1009	11
100A	32
100B	00
100C	60
100D	76
100E	3E
100F	22
1010	32
1011	00
1012	60
1013	76

2. Data transfer memory to memory:

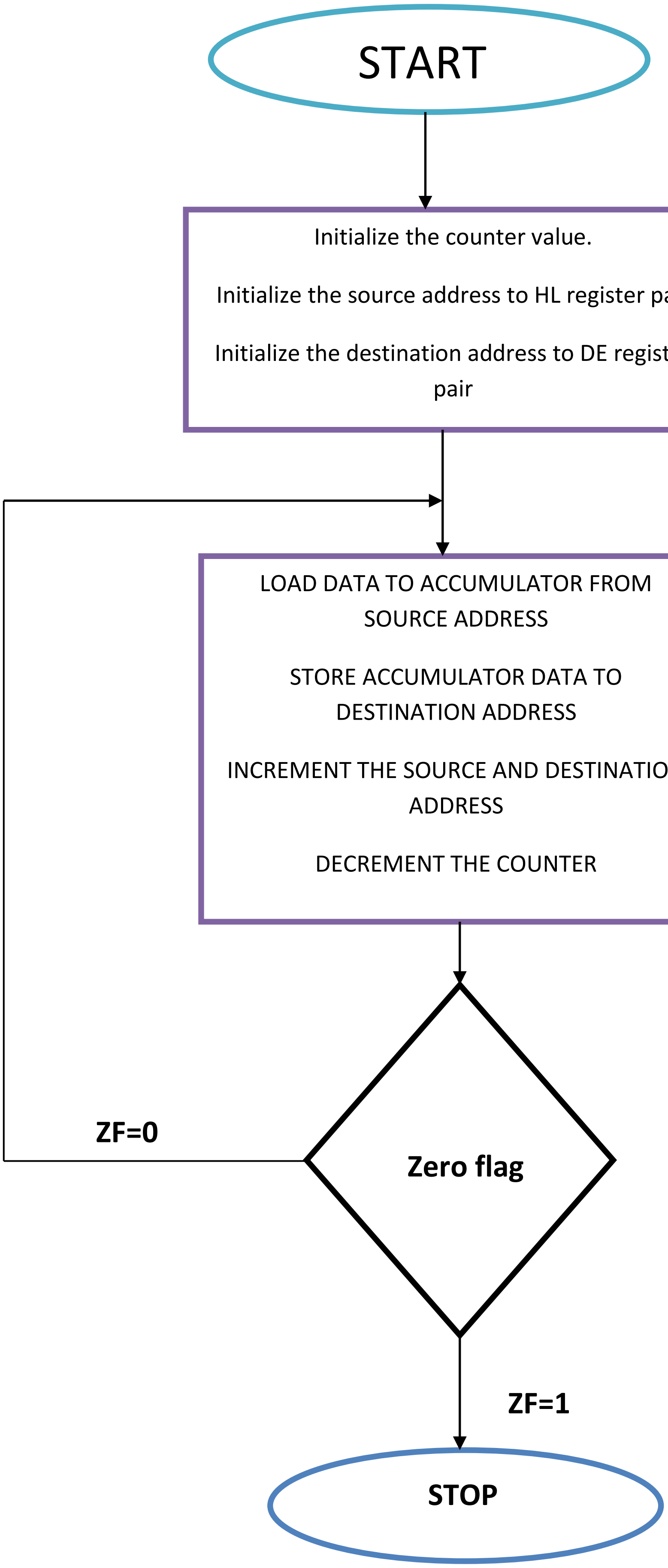
Write an ALP to 16 bytes of data stored in memory locations at 2000H to 200FH. Transfer the entire block of data to new memory locations starting at 4000H.

STEP 1: Pseudo code

STEP 1: Pseudo code

- initialize the counter
- Move the source and destination memory address as data to two register pair.
- Move the content of the one data pair to another data pair
- increment both the register pair address
- decrement the counter
- repeat the step 3 until the counter becomes zero

STEP 2: FLOW CHART


START
Initialize the counter value.
Initialize the source address to HL register pair
Initialize the destination address to DE register pair
LOAD DATA TO ACCUMULATOR FROM SOURCE ADDRESS
STORE ACCUMULATOR DATA TO DESTINATION ADDRESS
INCREMENT THE SOURCE AND DESTINATION ADDRESS
DECREMENT THE COUNTER
ZF=0
Zero flag
ZF=1
STOP

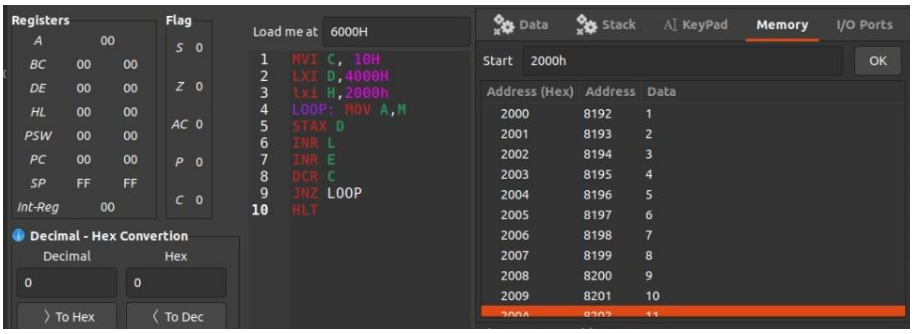
STEP 3: ALP

Memory location	Hexa code	label	Opcode	operand	comment
1001H	XX		MVI	C,10h	C<-- 16
1002H	10H				
1003H	XX		LXI	D,4000h	D<-- 4000H
1004H	00H				
1005H	40H				
1006H	XX		LXI	H,2000H	H<-- 2000H
1007H	00H				
1008H	20H				
1009H	XX	LOOP:	MOV	H,M	A<--[HL]
100AH	XX		STAX	D	[DE]<--A
100BH	XX		INR	L	INCREMENT SOURCE ADDRESS
100CH	XX		INR	E	INCREMENT THE DESTINATION ADDRESS
100DH	XX		DCR	C	DECREMENT COUNTER
100EH	XX		JNZ	LOOP	GOTO LOOP LABEL IF C=00H
100FH	XX		HLT		EXIT

STEP 4: EXECUTION IN SIMULATION

BEFORE PROGRAM EXECUTION :

STARTING MEMORY ADDRESS: (2000H)



Registers		
A	00	
BC	00	00
DE	00	00
HL	00	00
PSW	00	00
PC	00	00
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

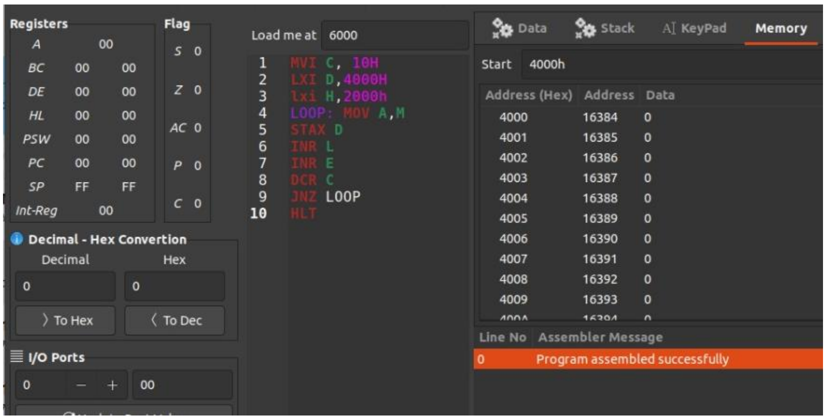
Load me at 6000H

```
MVI C, 10H
LXI D,4000H
lxi H,2000h
LOOP: MOV A,M
STAX D
INR L
INR E
DCR C
JNZ LOOP
HLT
```

Start 2000h

Address (Hex)	Address	Data
2000	8192	1
2001	8193	2
2002	8194	3
2003	8195	4
2004	8196	5
2005	8197	6
2006	8198	7
2007	8199	8
2008	8200	9
2009	8201	10
200A	8202	11

DESTINATION ADDRESS: (4000H)



Registers		
A	00	
BC	00	00
DE	00	00
HL	00	00
PSW	00	00
PC	00	00
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

Load me at 6000

```
MVI C, 10H
LXI D,4000H
lxi H,2000h
LOOP: MOV A,M
STAX D
INR L
INR E
DCR C
JNZ LOOP
HLT
```

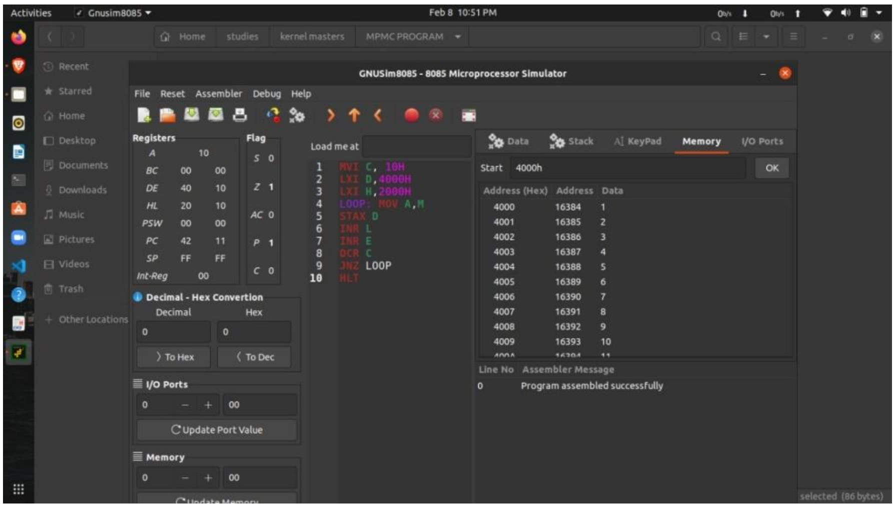
Start 4000h

Address (Hex)	Address	Data
4000	16384	0
4001	16385	0
4002	16386	0
4003	16387	0
4004	16388	0
4005	16389	0
4006	16390	0
4007	16391	0
4008	16392	0
4009	16393	0

Line No	Assembler Message
0	Program assembled successfully

AFTER EXECUTING PROGRAM:

DESTINATION ADDRESS: (4000H)



Registers		
A	10	
BC	00	00
DE	40	10
HL	20	10
PSW	00	00
PC	42	11
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Load me at

```
MVI C, 10H
LXI D,4000H
LXI H,2000H
LOOP: MOV A,M
STAX D
INR L
INR E
DCR C
JNZ LOOP
HLT
```

Start 4000h

Address (Hex)	Address	Data
4000	16384	1
4001	16385	2
4002	16386	3
4003	16387	4
4004	16388	5
4005	16389	6
4006	16390	7
4007	16391	8
4008	16392	9
4009	16393	10

Line No	Assembler Message
0	Program assembled successfully

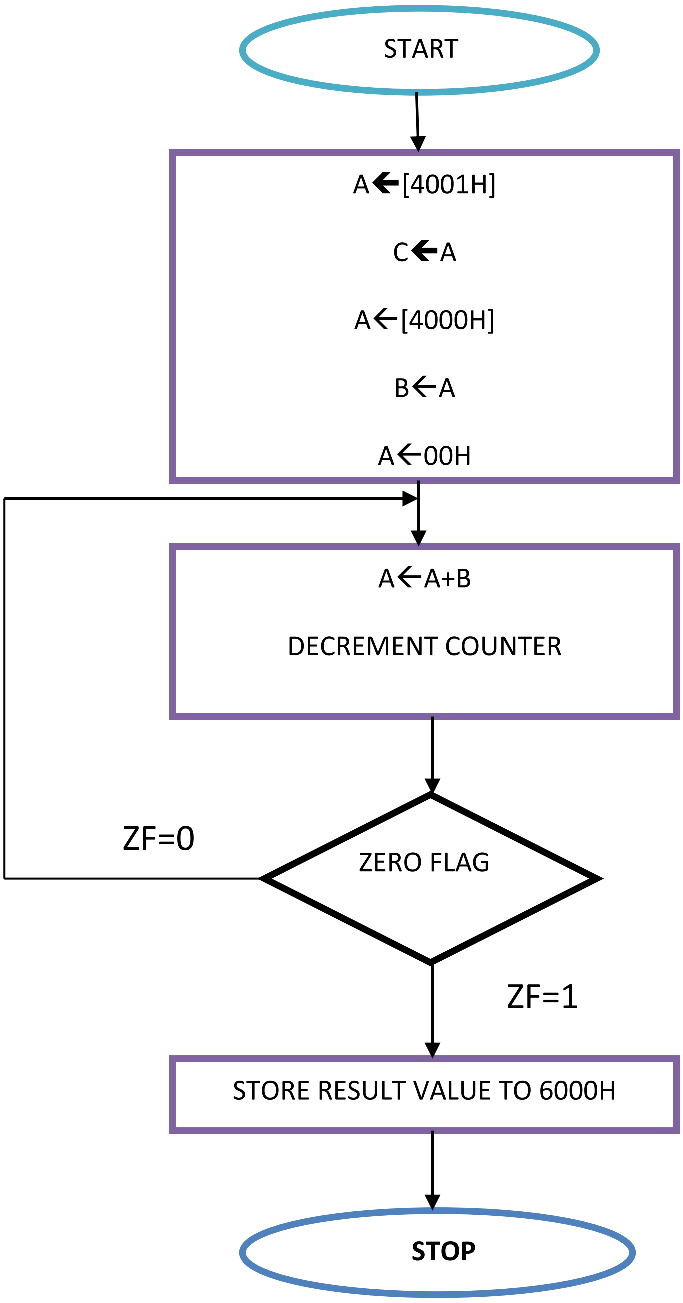
3. To Perform Multiplication without using MUL instruction:

Write an ALP to perform multiplication of two numbers without using MUL instruction first & second number stored in 4000H & 4001H memory locations respectively and the result stored in 6000H?

STEP 1: Pseudo code

1. Load the content of the 4001h to accumulator.
2. Move the accumulator value to counter.
3. Load the content of the 4000h to accumulator.
4. Add the accumulator value with SAME DATA store the value in the accumulator.
5. Decrement counter value
6. Repeat step 4&5 until counter become zero.
7. Store the result in 6000h.

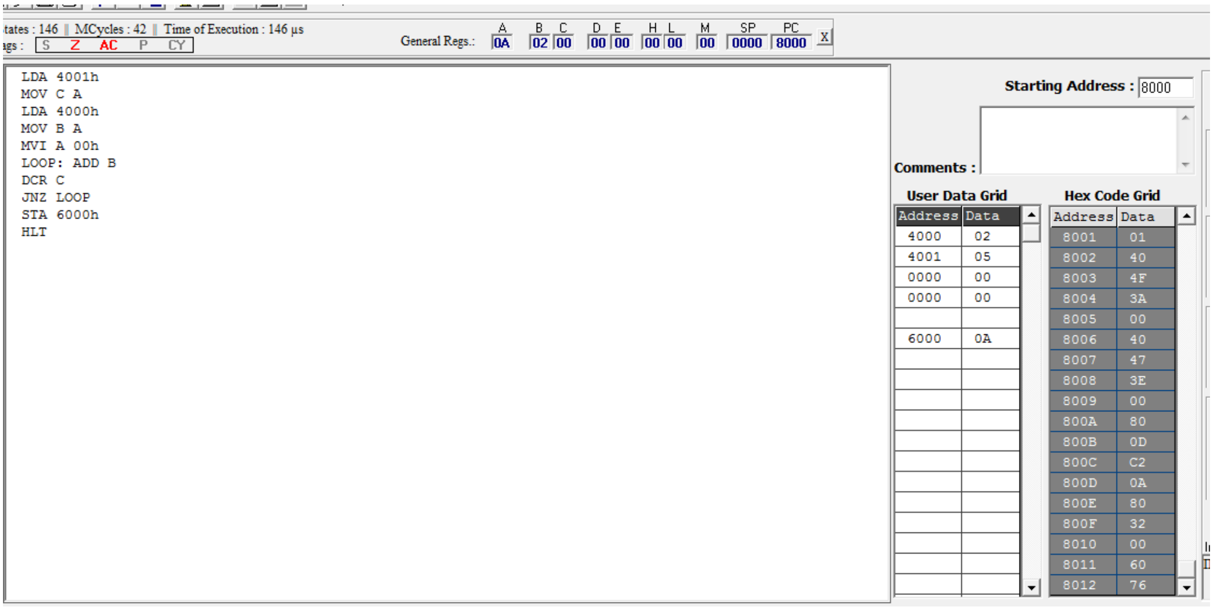
STEP 2: FLOW CHART



STEP 3:ALP

MEMORY ADDRESS	HEXA CODE	LABEL	OPCODE	OPERAND	COMMENT
1000H	XX		LDA	4001H	A←[4001]
1001H	01H				
1002H	40H				
1003H	XX		MOV	C, A	C←A
1004H	XX		LDA	4000H	A←[4000H]
1005H	00H				
1006H	40H				
1007H	XX		MOV	B, A	B←A
1008H	XX		MVI	A,00H	A←00H
1009H	00H				
100AH	XX	LOOP:	ADD	B	A←B
100BH	XX		DCR	C	C←C-1
100CH	XX		JNZ	LOOP	JUMP TO LOOP LABEL IF NO ZERO FLAG=0, ELSE EXIT
100DH	XX		STA	6000H	[6000H]←A
100EH	00H				
100FH	60H				
1010H	XX		HLT		EXIT

STEP 4 : EXECUTION



tates : 146 ‖ MCycles : 42 ‖ Time of Execution : 146 µs
ags : S Z AC P CY

General Regs.: A 0A | B 02 | C 00 | D 00 | E 00 | H 00 | L 00 | M 00 | SP 0000 | PC 8000

```
LDA 4001h
MOV C A
LDA 4000h
MOV B A
MVI A 00h
LOOP: ADD B
DCR C
JNZ LOOP
STA 6000h
HLT
```

Starting Address : 8000

Comments :

User Data Grid

Address	Data
4000	02
4001	05
0000	00
0000	00
6000	0A

Hex Code Grid

Address	Data
8001	01
8002	40
8003	4F
8004	3A
8005	00
8006	40
8007	47
8008	3E
8009	00
800A	80
800B	0D
800C	C2
800D	0A
800E	80
800F	32
8010	00
8011	60
8012	76

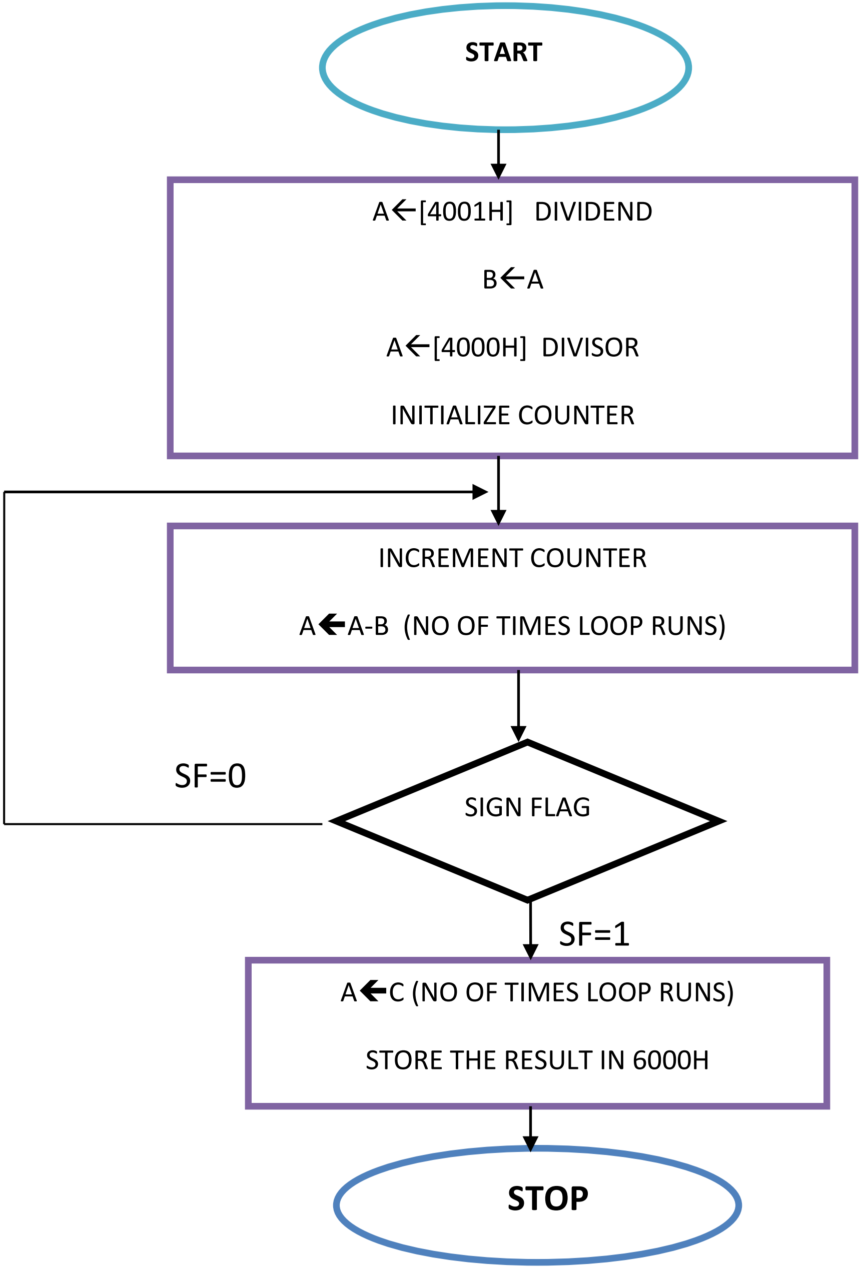
4. To Perform Division without using DIV instruction:

Write an ALP to perform division of two numbers without using DIV instruction first & second number stored in 4000H & 4001H memory locations respectively and the result stored in 6000H?

STEP 1:Pseudo code

```
1. Load the dividend to the accumulator 4001h
2. Store dividend in any register
3. Load divisor to accumulator 4000h
4. Initialize counter
5. Increment counter
6. Subtract dividend by divisor and store result in accumulator
7. Goto step 4 until the value become negative
8. Store the counter in accumulatror
9. Store the no of times loop run in 6000h
```

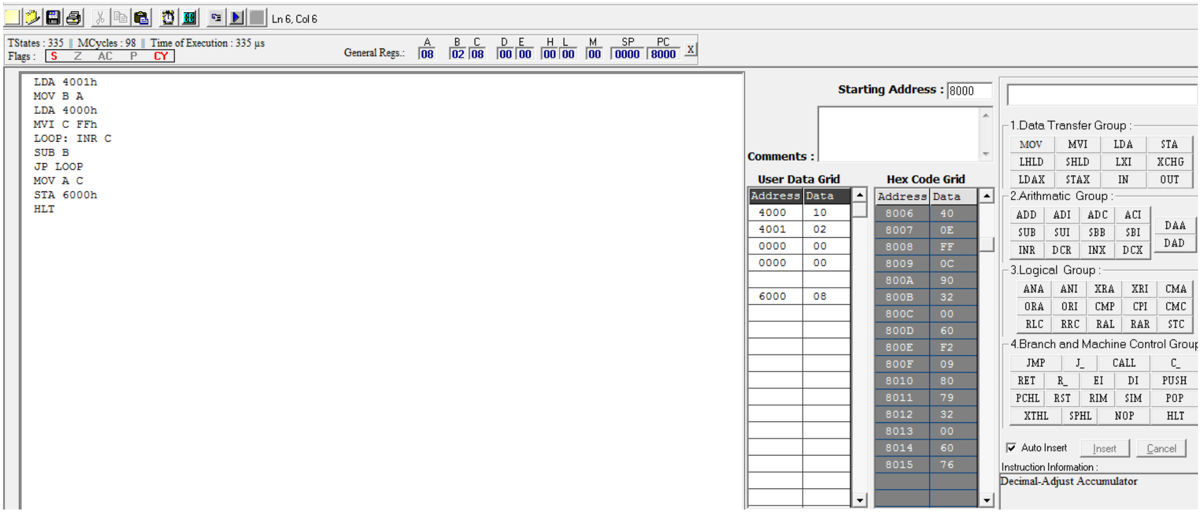
STEP2 :FLOW CHART



STEP3: ALP

MEMORY ADDRESS	HEXA CODE	LABEL	OPCODE	OPERAND	COMMENT
1000H	XX		LDA	4001H	A←[4001]
1001H	01H				
1002H	XX		MOV	B, A	B←A
1003H	XX		LDA	4000H	A←[4000H]
1004H	00H				
1005H	40H				
1006H	XX		MVI	C, FFH	C←FFH
1007H	FFH				
1008H	XX	LOOP:	INR	C	C=C+1
1009H	XX		SUB	B	A←A-B
100AH	XX		JP	LOOP	GOTO LOOP LABEL IF THE OUTPUT IS POSITIVE
100BH	XX		MOV	A, C	A←C
100CH	XX		STA	6000H	[6000]←A
100DH	00H				
100EH	60H				
100FH	XX		HLT		EXIT

STEP 4: EXECUTION IN SIMULATION



TStates : 335 | MCycles : 98 | Time of Execution : 335 µs
Flags : S Z AC P CY

General Regs.:

A	B	C	D	E	H	L	M	SP	PC
08	02	08	00	00	00	00	00	0000	8000

```
LDA 4001h
MOV B A
LDA 4000h
MVI C FFh
LOOP: INR C
SUB B
JP LOOP
MOV A C
STA 6000h
HLT
```

Starting Address : 8000

Comments :

User Data Grid

Address	Data
4000	10
4001	02
0000	00
0000	00
6000	08

Hex Code Grid

Address	Data
8006	40
8007	0E
8008	FF
8009	0C
800A	90
800B	32
800C	00
800D	60
800E	F2
800F	09
8010	80
8011	79
8012	32
8013	00
8014	60
8015	76

☑ Auto Insert

Instruction Information :
Decimal-Adjust Accumulator

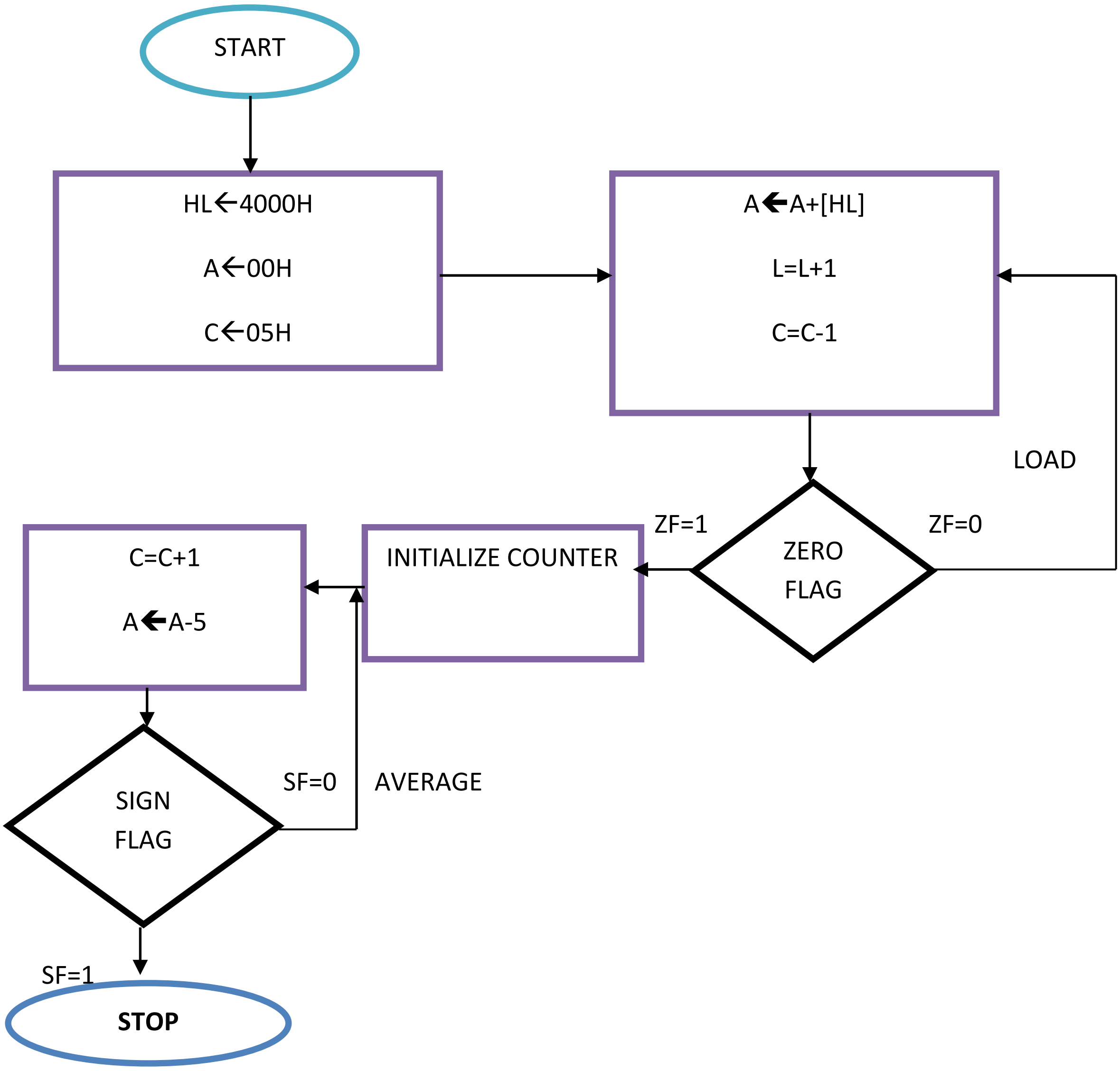
5. To find average of 5 Numbers:

Write an ALP average of 5 numbers takes the numbers from 4000H to 4004H location and store the result in 4005H?

Step 1: Pseudo code

```
1. Initialize HL register with 4000h ,accumulator and counter
2. Add content pointing by the HL pair register with accumulator
3. Increment HL pair
4. Goto step 2 for 5 times to load data
5. Initialize counter
6. Increment counter
7. A←A-05
8. Incase sign flag active exit, else goto step 6 until it became negative.
9. Store the no of times loops runs to the 4005H memory location.
```

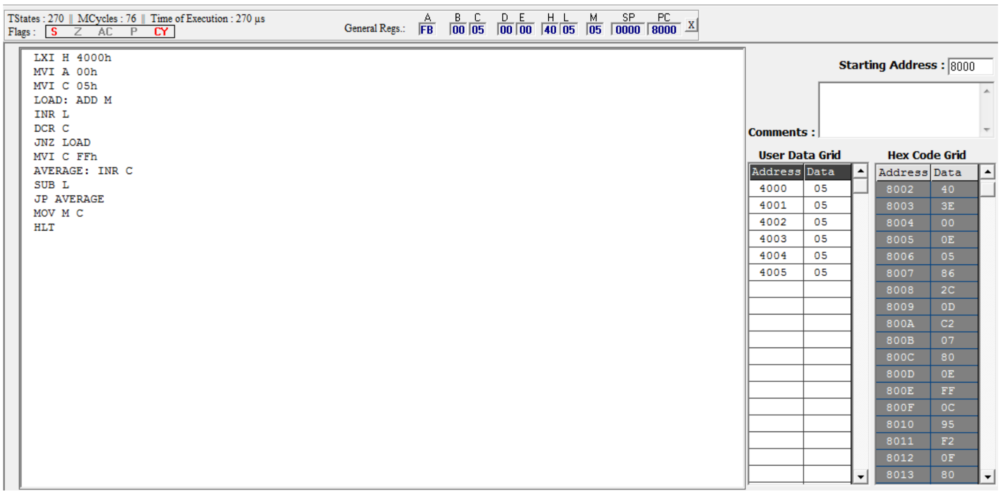
Step 2: flow chart



STEP 3: ALP

MEMORY ADDRESS	HEXA CODE	LABEL	OPCODE	OPERAND	COMMENT
1000H	XX 00H 40H		LXI	H, 4000H	H←[4000H]
1003H	XX 00H		MVI	A, 00H	A←00H
1005H	XX 05H		MVI	C, 05H	C←05H
1007H	XX	LOAD:	ADD	M	A←A+[HL]
1008H	XX		INR	L	INCREMENT MEMORY ADDRESS
1009H	XX		DCR	C	C=C-1
100AH	XX		JNZ	LOAD	JUMP TO LOAD IF COUNTER NOT ZERO ELSE EXIT
100BH	XX FFH		MVI	C, FFH	COUNTER INITIALIZE
100DH	XX	AVERAGE:	INR	C	C=C+1
100EH	XX		SUB	L	A←A-05
100FH	XX		JP	AVERAGE	
1010H	XX		MOV	M, C	STORE THE NO OF TIMES EXECUTED IN THE 4005H
1011H	XX		HLT		

STEP 5: EXECUTION IN SIMULATION



TStates : 270 ‖ MCycles : 76 ‖ Time of Execution : 270 µs

Flags : S Z AC P CY

General Regs.:	A	B	C	D	E	H	L	M	SP	PC
	FB	00	05	00	00	40	05	05	0000	8000

```
LXI H 4000h
MVI A 00h
MVI C 05h
LOAD: ADD M
INR L
DCR C
JNZ LOAD
MVI C FFh
AVERAGE: INR C
SUB L
JP AVERAGE
MOV M C
HLT
```

Starting Address : 8000

Comments :

User Data Grid

Address	Data
4000	05
4001	05
4002	05
4003	05
4004	05
4005	05

Hex Code Grid

Address	Data
8002	40
8003	3E
8004	00
8005	0E
8006	05
8007	86
8008	2C
8009	0D
800A	C2
800B	07
800C	80
800D	0E
800E	FF
800F	0C
8010	95
8011	F2
8012	0F
8013	80

6 · Step 1: ~~Pseedu~~Pseado Code ·

* Initialize 16-bit data in the register pair

* Load stack pointer address in the HL reg pair

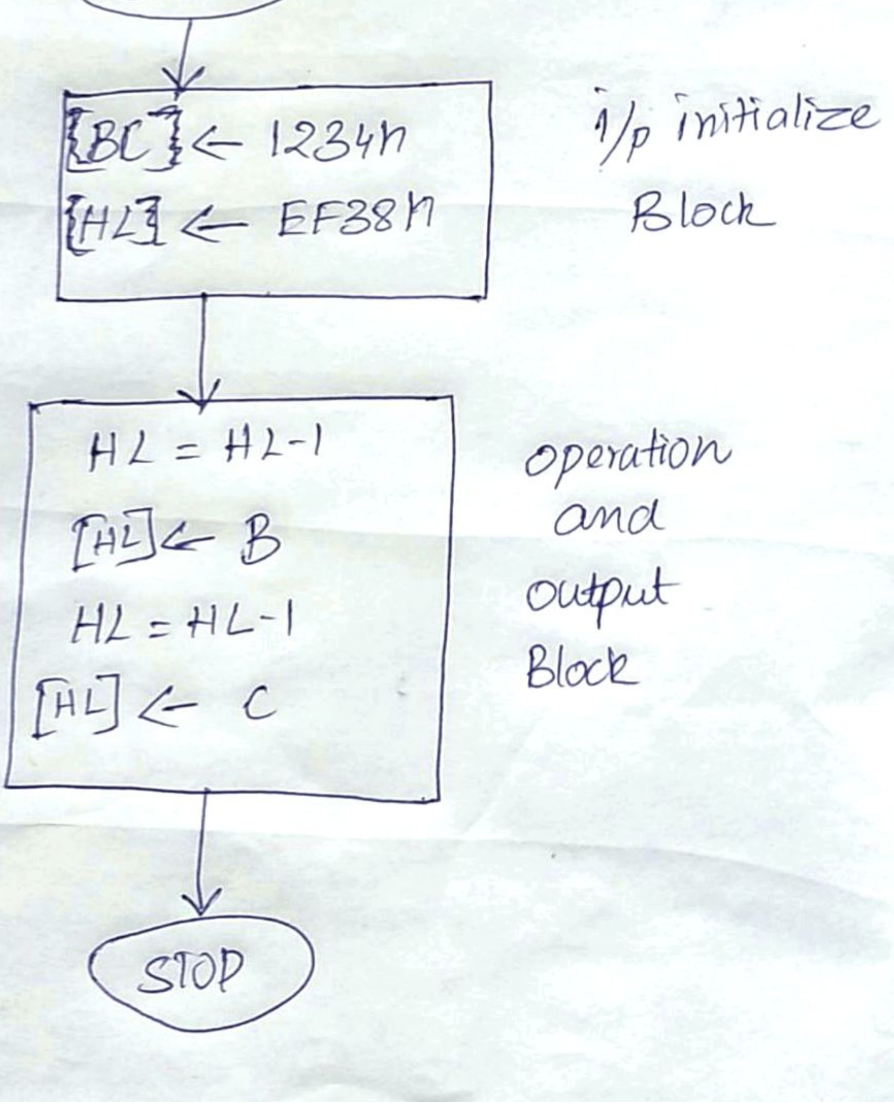
* Decrement HL pair

* Load higher order byte

* Decrement HL pair

* Load lower order byte.

step2: Flow chart.



Step 3:- A2P

Mem Address	Hex Code	Label	Opcode	operand	Comment
1000h	XX		LXI	b,1234H	[BC]← 1234H
1001h	34H				
1002H	12H				
1003h	XX		LXI	H, EF38H	{HL}← EF38H
1004h	38H				
1005h	EF				
1006h	XX		dcx	H	HL= HL-1
1007H	XX		MOV	m, b	[HL]← B
1008H	XX		dcx	h	HL= H2-1
1009H	XX		mov	m, c	[H2]∈C

7. Count No of 1's

Step 1:- Initialize Pseudo Code

1. Initialize Counter and load memory address data stored.

2. Move the content to the register.

3. Perform the OR operation to set the flag, so that the zero flag affected.

4. If stored data zero exit the loop and terminate the program store zero at destination register.

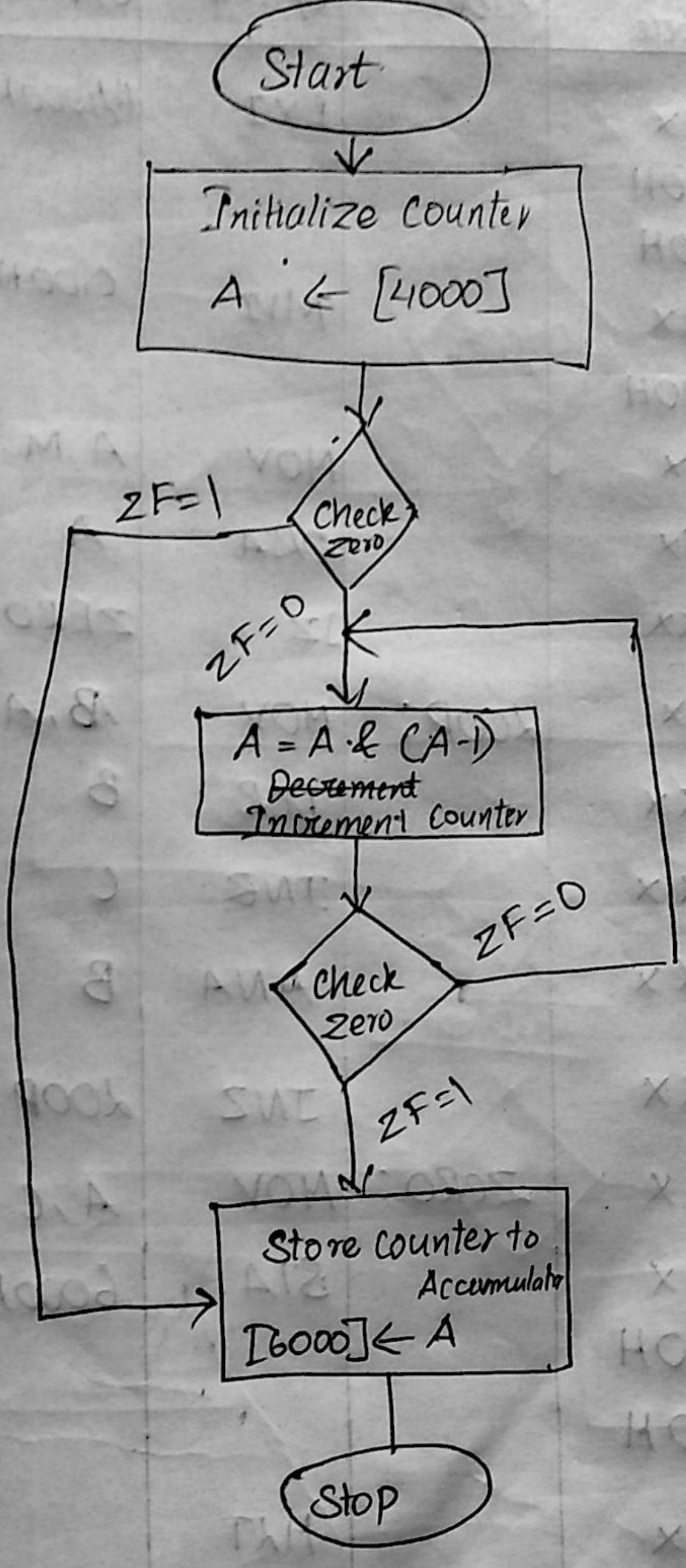
5. Else store the data to another register and decrement it.

6. Using the formula $n = n \& (n-1)$ count the loop until its break.

7. If output zero exit loop, else goto step 5.

8. After loop terminated no of times loop executed are stored in destination

Step:2 Flow Chart.



Step 3: ALP

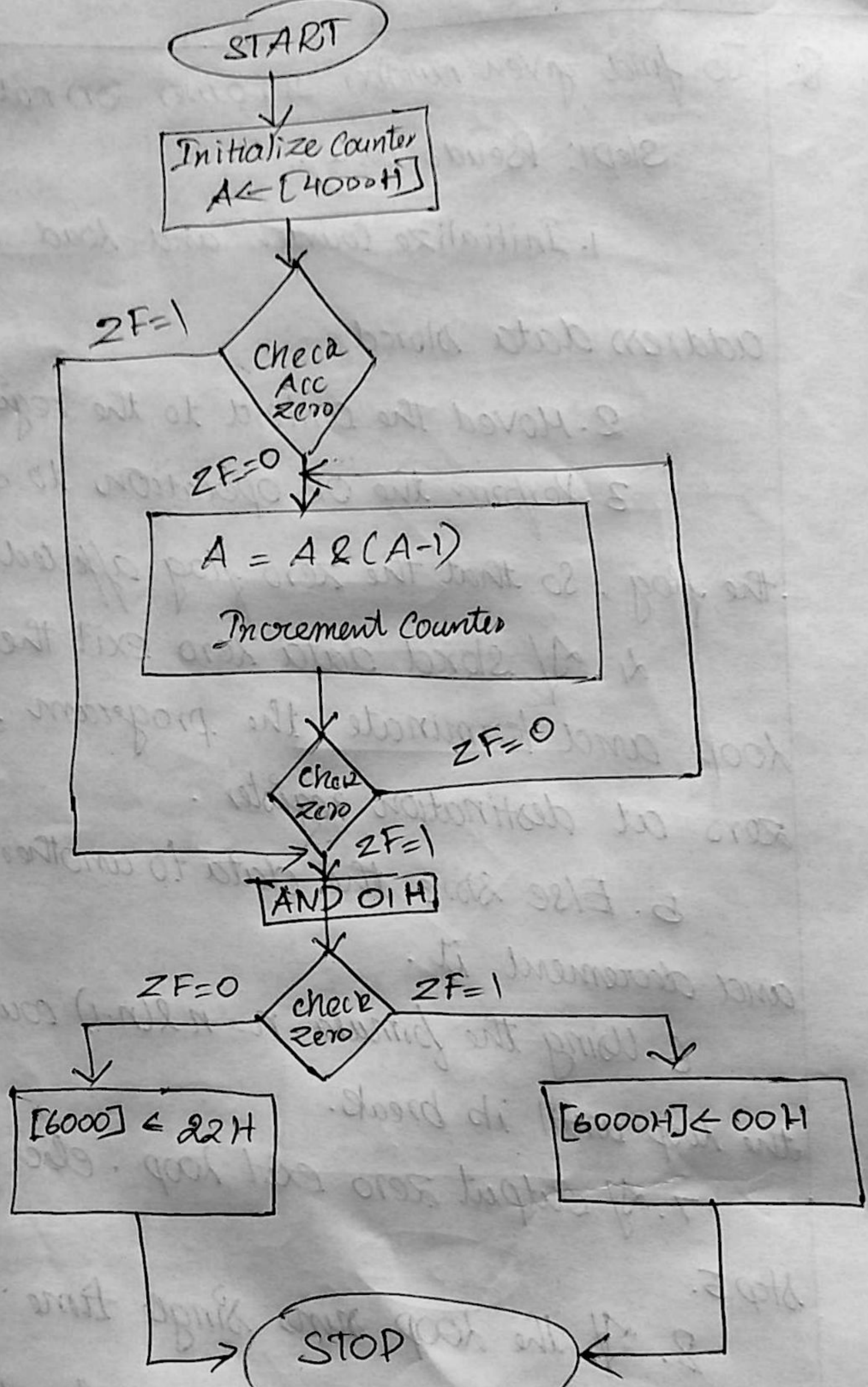
Memory Address	Hexa Code	Label	opcode	OPERAND	COMMENT
1000H	XX		LXI	H,4000H	A←[4000
1001H	00H				
1002H	40H		MVI	C,00H	C←00H
1003H	XX				
1004H	00H				
1005H	XX		MOV	A,M	A←[HL]
1006H	XX		ORA	A	
1007H	XX		JZ	ZERO	
1008H	XX	LOOP:	MOV	B, A	(n-1)
1009H	XX		DCR	B	
100AH	XX		INR	C	
100BH	XX		ANA	B	n=n&(n-1)
100CH	XX		JNZ	LOOP	
100DH	XX	ZERO:	MOV	A, C	no of time LOOP Run
100EH	XX		STA	6000H	
100FH	00H				
1010H	60H				
1011H	XX		HLT		

8. To find given number 2 power (or) not.

Step1: Pseudo code.

```
1. Initialize Counter and load memory address data stored.
2. Moved the contend to the register.
3. Perform the OR operation to set the flog, So that the zero flag affected.
4. If stored data zero exit the loop and terminate the program store zero at destination register.
5. Else store the data to another register and decrement it.
6. Using the formula $n = n \& (n-1)$ count the loop until its break.
7. If output zero exit loop. else goto step 5.
8. If the loop runs single time then store 22h to 6000H memory location else clear it.
```

Step 2: Flow chart .



Step 3: ALP

~~Hex~~ Address	Hexa code	Label	OPCODE	OPERAND	COMMENT
1000H	XX		LXI	H,4000H	
1001H	00H				
1002H	40H				
1003H	XX		MVI	C,00H	
1004H	00H				
1005H	XX		MOV	A,M	
1006H	XX		OR	A	
1007H	XX		JZ	ZERO	
1008H	XX	LOOP:	MOV	B,A	
1009H	XX		DCR	B	(n-1)
100AH	XX		INR	C	
100BH	XX		ANA	B	n=n&(n-1)
100CH	XX		JNZ	LOOP	
100DH	XX	ZERO:	MOV	A,C	
100EH	XX		ANI	01H	
100FH	01H				
1010H	XX		JZ	EXIT	
1011H	XX		MVI	A,22H	A←22H If power of 2
1012H	22H		~~ST~~		
1013H	XX		STA	6000H	
1014H	00H				
1015H	60H				
1016H	XX		HLT		
1017H	XX	EXIT:	MVI	A,00H	A←00H If not p of 2
1018H	00H				
1019H	XX 00H 60H		STA	6000H	
101CH	XX		HLT		