Name : Mangesh Chandrawanshi
Roll No. : 1801CS16
Date : 29/08/2020
Course : CS322 Computer Architectue Lab
Submission : Lab1
FileName : 1801CS16_Lab1.pdf

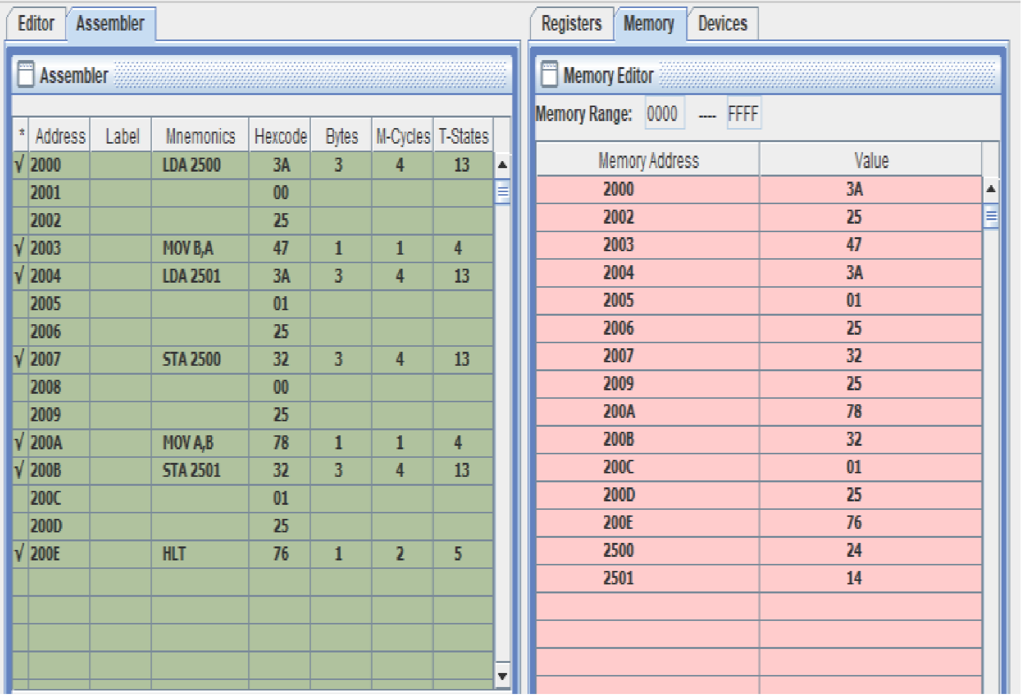
```
# ORG is a standard (almost universal) command that tells the assembler where the
program is to reside in memory. It is the address of the first instruction (or
data) of the program (the ORiGin.)
# DB reserves one byte of memory and initialize the byte with the specified value
```

P1) Program to swap two 8-bit numbers

Algorithm/Comments :
Load value at 2500H in A
Move it to B
Load value at 2501H in A
Store it to 2500H
Move value in B to A
Store it to 2501H

Code:

```
# ORG 2000H
# BEGIN 2000H

LDA 2500H
MOV B,A
LDA 2501H
STA 2500H
MOV A,B
STA 2501H
HLT

# ORG 2500H
# DB 14H,24H
```



Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LDA 2500	3A	3	4	13
	2001			00			
	2002			25			
√	2003		MOV B,A	47	1	1	4
√	2004		LDA 2501	3A	3	4	13
	2005			01			
	2006			25			
√	2007		STA 2500	32	3	4	13
	2008			00			
	2009			25			
√	200A		MOV A,B	78	1	1	4
√	200B		STA 2501	32	3	4	13
	200C			01			
	200D			25			
√	200E		HLT	76	1	2	5

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	3A
2002	25
2003	47
2004	3A
2005	01
2006	25
2007	32
2009	25
200A	78
200B	32
200C	01
200D	25
200E	76
2500	24
2501	14

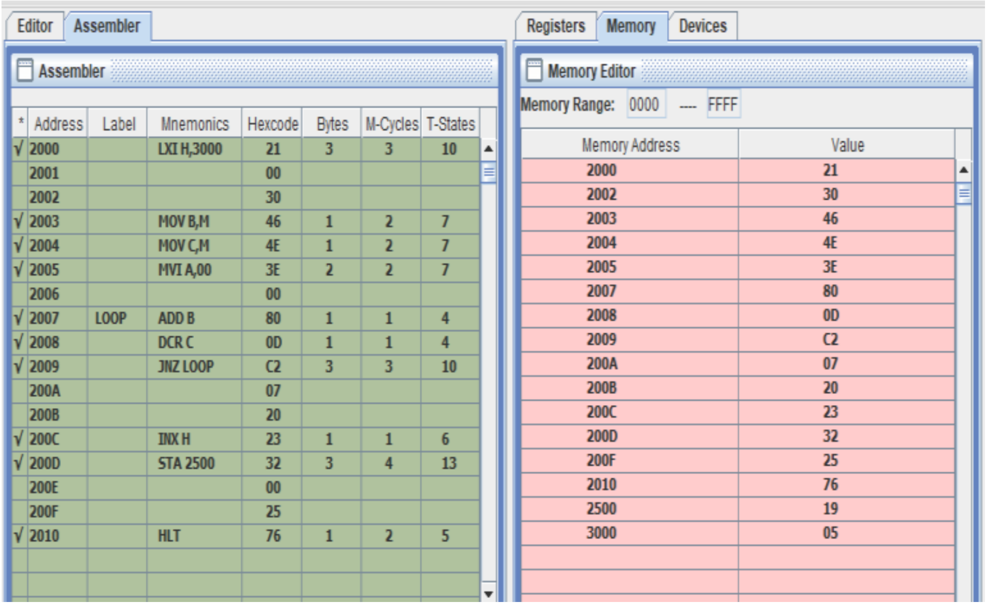
```
// INPUT -> 2500H = 14H, 2501H = 24H
// OUTPUT -> 2500H = 24H, 2501H = 14H
```

P2) Program to square of an 8-bit number (Only if square can be stored in 8-bit)

Algorithm/Comments :
Load Memory Location of the given number in HL pair register
Move the value to B and C from M
Initialize A to zero
Add value in B to A , Decrement C continue this till C is non-zero
Now A has squared value, store it in Memory(Here at 2500H)

Code :

```
# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MOV B,M
            MOV C,M
            MVI A,00H

LOOP:       ADD B
            DCR C
            JNZ LOOP

            INX H
            STA 2500H
            HLT

# ORG 3000H
# DB 05H

// INPUT -> 3000H = 05H
// OUTPUT -> 2500H = 19H
```



Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV B,M	46	1	2	7
√	2004		MOV C,M	4E	1	2	7
√	2005		MVI A,00	3E	2	2	7
	2006			00			
√	2007	LOOP	ADD B	80	1	1	4
√	2008		DCR C	0D	1	1	4
√	2009		JNZ LOOP	C2	3	3	10
	200A			07			
	200B			20			
√	200C		INX H	23	1	1	6
√	200D		STA 2500	32	3	4	13
	200E			00			
	200F			25			
√	2010		HLT	76	1	2	5

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	46
2004	4E
2005	3E
2007	80
2008	0D
2009	C2
200A	07
200B	20
200C	23
200D	32
200F	25
2010	76
2500	19
3000	05

P3) Program to find Largest of two 8-bit numbers

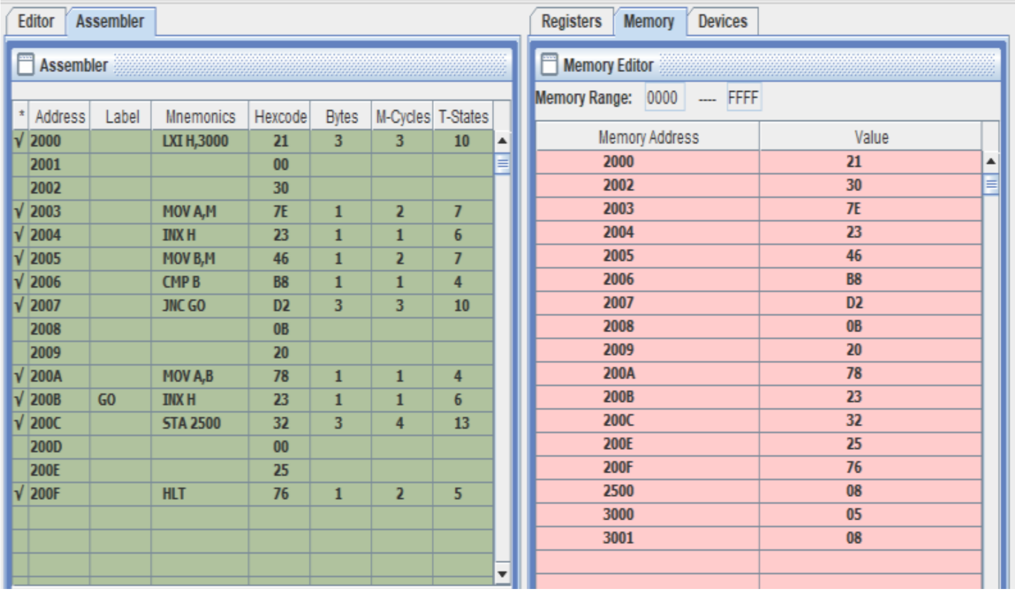
Algorithm/Comments :
Load Memory Location of the first number in HL pair register
Move value to A
Increment H for accessing second number
Move it to B
Compare value in B with A
if it is greater, move it to A
else keep as it is
Store in the value in A (largest number) at desired Memory Location (Here at 2500H)

/*CMP instruction :

This is a 1-byte instruction. It compares the data byte in the register or memory
with the contents of accumulator.
If A less than (R/M), the CY(carry) flag is set and Zero flag is reset.
If A equals to (R/M), the Zero flag is set and CY flag is reset.
If A greater than (R/M), the CY and Zero flag are reset.*/

Code :

```
# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MOV A,M
            INX H
            MOV B,M
            CMP B
            JNC GO

            MOV A,B
GO:         INX H
            STA 2500H
            HLT

# ORG 3000H
# DB 05H,08H

// INPUT -> 3000H = 05H, 3001H = 08H
// OUTPUT ->  2500H = 08H
```



Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV A,M	7E	1	2	7
√	2004		INX H	23	1	1	6
√	2005		MOV B,M	46	1	2	7
√	2006		CMP B	B8	1	1	4
√	2007		JNC GO	D2	3	3	10
	2008			0B			
	2009			20			
√	200A		MOV A,B	78	1	1	4
√	200B	GO	INX H	23	1	1	6
√	200C		STA 2500	32	3	4	13
	200D			00			
	200E			25			
√	200F		HLT	76	1	2	5

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	7E
2004	23
2005	46
2006	B8
2007	D2
2008	0B
2009	20
200A	78
200B	23
200C	32
200E	25
200F	76
2500	08
3000	05
3001	08

P4) Program to find smallest from an Array

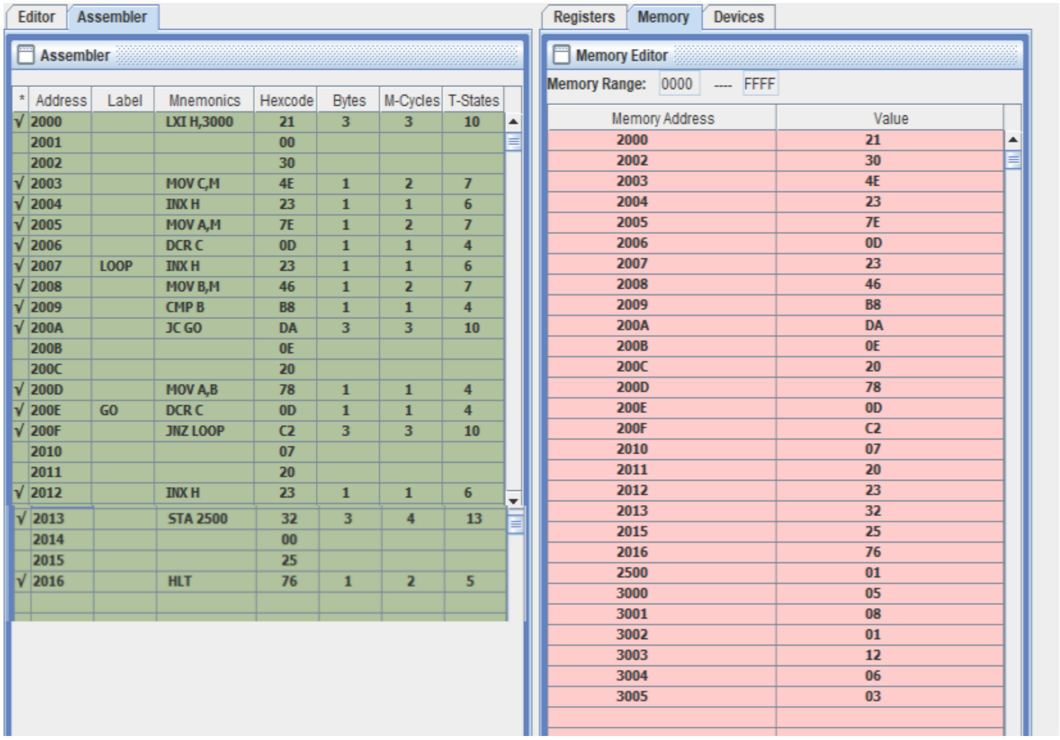
Algorithm/Comments :
Load Memory Location of the size of array in HL pair register
Move the value (size) to C
Increment H to access array elements
Move first element to accumulator
Decrement C (working as a counter)

Starting of Loop - Increment H (next element)
Move the value to B
Compare it with value in A
if value in B is smaller, move it to A
else keep as it is
Decrement C and loop till C is non-zero

Smallest element is at A
Store it in Desired Memory Location (Here at 2500H)

Code:

```
# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MOV C,M
            INX H
            MOV A,M
            DCR C

LOOP: INX H
            MOV B,M
            CMP B
            JC GO
            MOV A,B

GO:     DCR C
            JNZ LOOP
            INX H
            STA 2500H
            HLT

# ORG 3000H
# DB 05H,08H,01H,12H,06H,03H
```



Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV C,M	4E	1	2	7
√	2004		INX H	23	1	1	6
√	2005		MOV A,M	7E	1	2	7
√	2006		DCR C	0D	1	1	4
√	2007	LOOP	INX H	23	1	1	6
√	2008		MOV B,M	46	1	2	7
√	2009		CMP B	B8	1	1	4
√	200A		JC GO	DA	3	3	10
	200B			0E			
	200C			20			
√	200D		MOV A,B	78	1	1	4
√	200E	GO	DCR C	0D	1	1	4
√	200F		JNZ LOOP	C2	3	3	10
	2010			07			
	2011			20			
√	2012		INX H	23	1	1	6
√	2013		STA 2500	32	3	4	13
	2014			00			
	2015			25			
√	2016		HLT	76	1	2	5

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	4E
2004	23
2005	7E
2006	0D
2007	23
2008	46
2009	B8
200A	DA
200B	0E
200C	20
200D	78
200E	0D
200F	C2
2010	07
2011	20
2012	23
2013	32
2015	25
2016	76
2500	01
3000	05
3001	08
3002	01
3003	12
3004	06
3005	03

// INPUT -> 3000H = 05H (size of array), (elements in array) 3001H = 08H, 3002H = 01H, 3003H = 12H, 3004H = 06H 3005H = 03H
// OUTPUT -> 2500H = 01H

P5) Program to Sort the array in Descending Order (in place sorting)

Algorithm/Comments :
Load Memory Location of the size of array in HL pair register
Move the value (size) to C

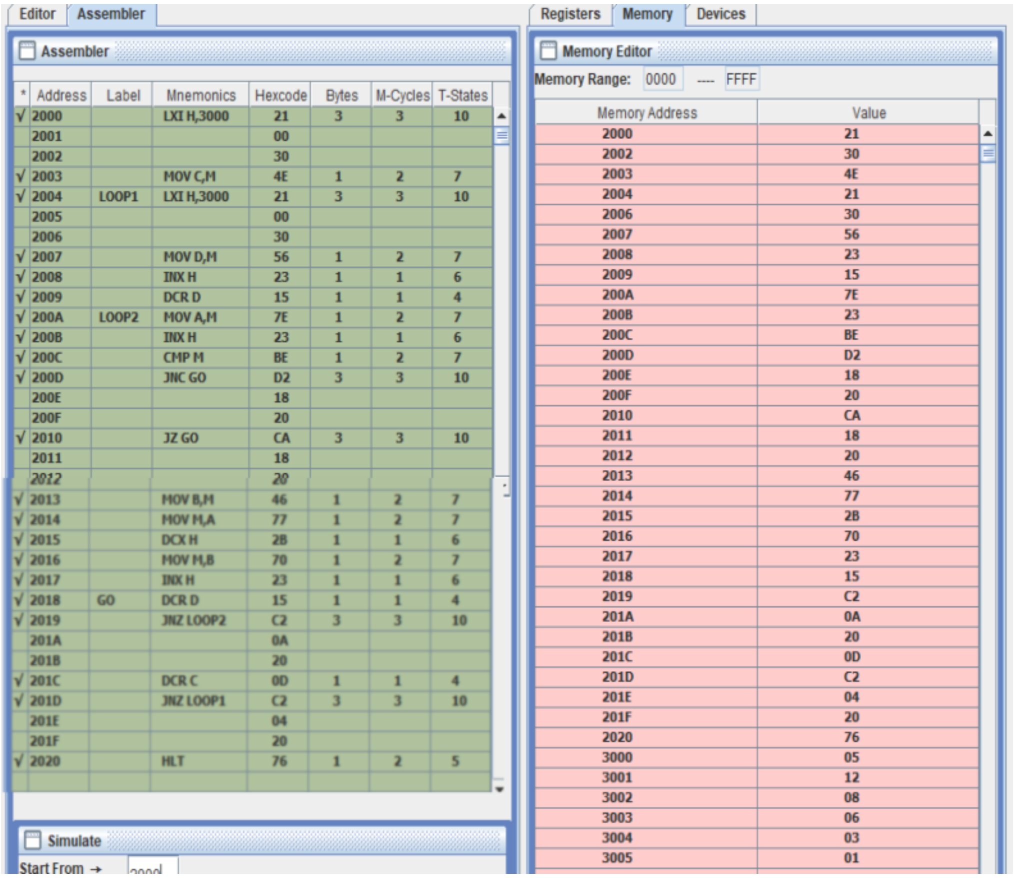
Starting of Loop1 - Load Memory Location of the size of array in HL pair register
Move the value (size) to D
Increment H to access array elements
Decrement D (working as a counter)

Starting of Loop2 - Move first element to accumulator
Increment H (next element)
Compare value in M (at H) with value in A
if value in A is smaller, swap them
else keep as it is
Decrement D and loop till D is non-zero

Decrement C and loop till C is non-zero

Code :

```
# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MOV C,M
LOOP1:      LXI H,3000H
            MOV D,M

            INX H
            DCR D
LOOP2:      MOV A,M
            INX H
            CMP M
            JNC GO
            JZ GO

            MOV B,M
            MOV M,A
            DCX H
            MOV M,B
            INX H

GO:     DCR D
            JNZ LOOP2
            DCR C
            JNZ LOOP1
            HLT
```



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV C,M	4E	1	2	7
√	2004	LOOP1	LXI H,3000	21	3	3	10
	2005			00			
	2006			30			
√	2007		MOV D,M	56	1	2	7
√	2008		INX H	23	1	1	6
√	2009		DCR D	15	1	1	4
√	200A	LOOP2	MOV A,M	7E	1	2	7
√	200B		INX H	23	1	1	6
√	200C		CMP M	BE	1	2	7
√	200D		JNC GO	D2	3	3	10
	200E			18			
	200F			20			
√	2010		JZ GO	CA	3	3	10
	2011			18			
	2012			20			
√	2013		MOV B,M	46	1	2	7
√	2014		MOV M,A	77	1	2	7
√	2015		DCX H	2B	1	1	6
√	2016		MOV M,B	70	1	2	7
√	2017		INX H	23	1	1	6
√	2018	GO	DCR D	15	1	1	4
√	2019		JNZ LOOP2	C2	3	3	10
	201A			0A			
	201B			20			
√	201C		DCR C	0D	1	1	4
√	201D		JNZ LOOP1	C2	3	3	10
	201E			04			
	201F			20			
√	2020		HLT	76	1	2	5

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	4E
2004	21
2006	30
2007	56
2008	23
2009	15
200A	7E
200B	23
200C	BE
200D	D2
200E	18
200F	20
2010	CA
2011	18
2012	20
2013	46
2014	77
2015	2B
2016	70
2017	23
2018	15
2019	C2
201A	0A
201B	20
201C	0D
201D	C2
201E	04
201F	20
2020	76
3000	05
3001	12
3002	08
3003	06
3004	03
3005	01

```
# ORG 3000H
# DB 05H,08H,01H,12H,06H,03H
```

// INPUT -> 3000H = 05H (size of array), (elements in array) 3001H = 08H, 3002H = 01H, 3003H = 12H, 3004H = 06H, 3005H = 03H
// OUTPUT -> Sorted array in Descending Order(12H,08H,06H,03H,01H)

P6) Program to convert HEX to BCD

Algorithm/Comments :
Load Memory Location of the number in HL register pair
Move value to C

Resultant BCD representation will be stored in A(LS) and B(MS) registers

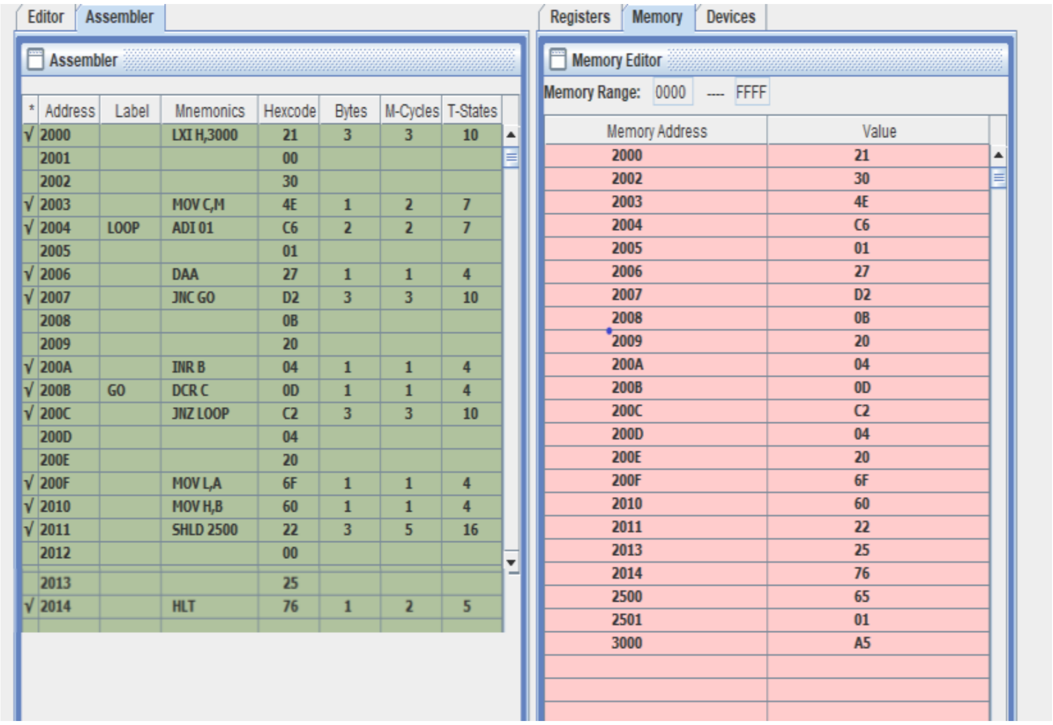
Starting of Loop - Add 01H to A
DAA instruction is applied to get BCD sum as Outcome

if carry flag is set , increment B
else do nothing
Decrement C and continue loop till C is non-zero

Move value in A to L and value in B to H
Store the BCD representation (in HL register pair) to desired memory location
(Here at 2500H)

Code :

```
# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MOV C,M

LOOP: ADI 01H
            DAA
            JNC GO
            INR B

GO:         DCR C
            JNZ LOOP

            MOV L,A
            MOV H,B
            SHLD 2500H
            HLT

# ORG 3000H
# DB A5H
```



Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV C,M	4E	1	2	7
√	2004	LOOP	ADI 01	C6	2	2	7
	2005			01			
√	2006		DAA	27	1	1	4
√	2007		JNC GO	D2	3	3	10
	2008			0B			
	2009			20			
√	200A		INR B	04	1	1	4
√	200B	GO	DCR C	0D	1	1	4
√	200C		JNZ LOOP	C2	3	3	10
	200D			04			
	200E			20			
√	200F		MOV L,A	6F	1	1	4
√	2010		MOV H,B	60	1	1	4
√	2011		SHLD 2500	22	3	5	16
	2012			00			
	2013			25			
√	2014		HLT	76	1	2	5

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	4E
2004	C6
2005	01
2006	27
2007	D2
2008	0B
2009	20
200A	04
200B	0D
200C	C2
200D	04
200E	20
200F	6F
2010	60
2011	22
2013	25
2014	76
2500	65
2501	01
3000	A5

```
// INPUT -> 3000H = A5H
// OUTPUT -> 2050H = 65H (LS), 2051H = 01H (MS) (i.e. 0000 0001 0110 0101)
```

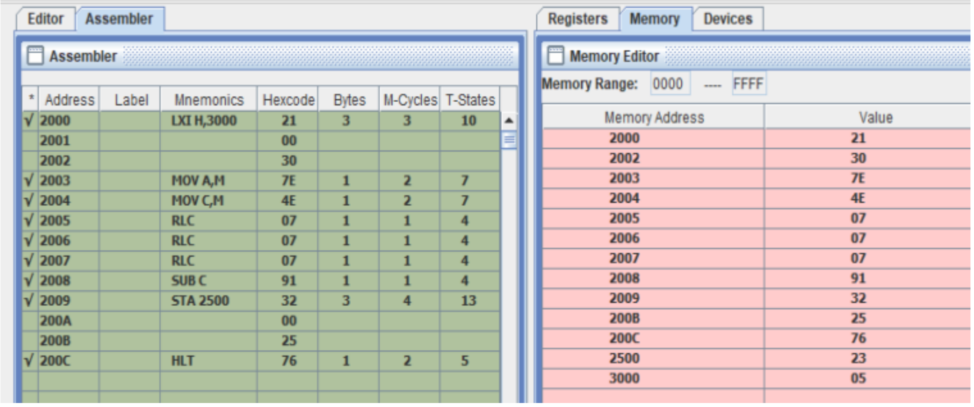
P7) Write a Program to multiply a Number by 7 (Without actually adding or multiplying) (works only if on multiplying by 7 number can be represented using 8 bits)

Algorithm/Comments :
Load Memory Location of the given number in HL register pair
Move the number to A and C
Apply instruction RLC 3 times
It will shift contents in A to the left 3 times (equivalent to multiply by 8)
Subtract value in C from A
Store value in A(7 multipled by original number) to desired memory location (Here at 2500H)

Code :

```
# ORG 2000H
# BEGIN 2000H

LXI H,3000H
MOV A,M
MOV C,M
RLC
RLC
RLC
```

```
SUB C
STA 2500H
HLT

# ORG 3000H
# DB 05H

// INPUT -> 3000H = 05H
// OUTPUT -> 2500H = 23H
```



Editor | Assembler

Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV A,M	7E	1	2	7
√	2004		MOV C,M	4E	1	2	7
√	2005		RLC	07	1	1	4
√	2006		RLC	07	1	1	4
√	2007		RLC	07	1	1	4
√	2008		SUB C	91	1	1	4
√	2009		STA 2500	32	3	4	13
	200A			00			
	200B			25			
√	200C		HLT	76	1	2	5

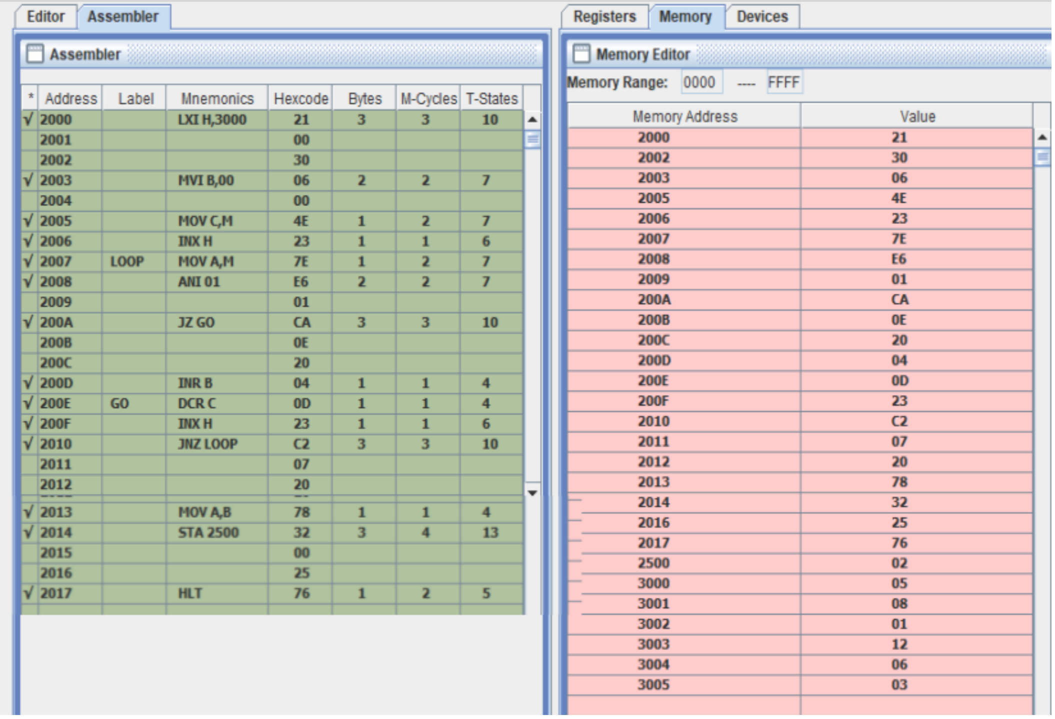
Registers | Memory | Devices

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	7E
2004	4E
2005	07
2006	07
2007	07
2008	91
2009	32
200B	25
200C	76
2500	23
3000	05

```
P8) Count total odd numbers in an array

Algorithm/Comments :
Load Memory Location of the size of array in HL register pair
Initialize B with 0
Move size of array value (in M) to C
Increment H (to access array elements)

Starting of loop - Move element to A
Apply AND with 01H on A
if A is zero do nothing
else increment B (as current accessed element is odd)
Decrement C , Increment H continue loop till C is non-zero

Move B to A
Store value in A(count of odd numbers) to desired memory location (Here at 2500H)

Code :

# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MVI B,00H
            MOV C,M

            INX H
LOOP: MOV A,M
            ANI 01H
            JZ GO
            INR B

GO:         DCR C
            INX H
            JNZ LOOP

            MOV A,B
            STA 2500H
            HLT
```



Editor | Assembler

Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MVI B,00	06	2	2	7
	2004			00			
√	2005		MOV C,M	4E	1	2	7
√	2006		INX H	23	1	1	6
√	2007	LOOP	MOV A,M	7E	1	2	7
√	2008		ANI 01	E6	2	2	7
	2009			01			
√	200A		JZ GO	CA	3	3	10
	200B			0E			
	200C			20			
√	200D		INR B	04	1	1	4
√	200E	GO	DCR C	0D	1	1	4
√	200F		INX H	23	1	1	6
√	2010		JNZ LOOP	C2	3	3	10
	2011			07			
	2012			20			
√	2013		MOV A,B	78	1	1	4
√	2014		STA 2500	32	3	4	13
	2015			00			
	2016			25			
√	2017		HLT	76	1	2	5

Registers | Memory | Devices

Memory Editor

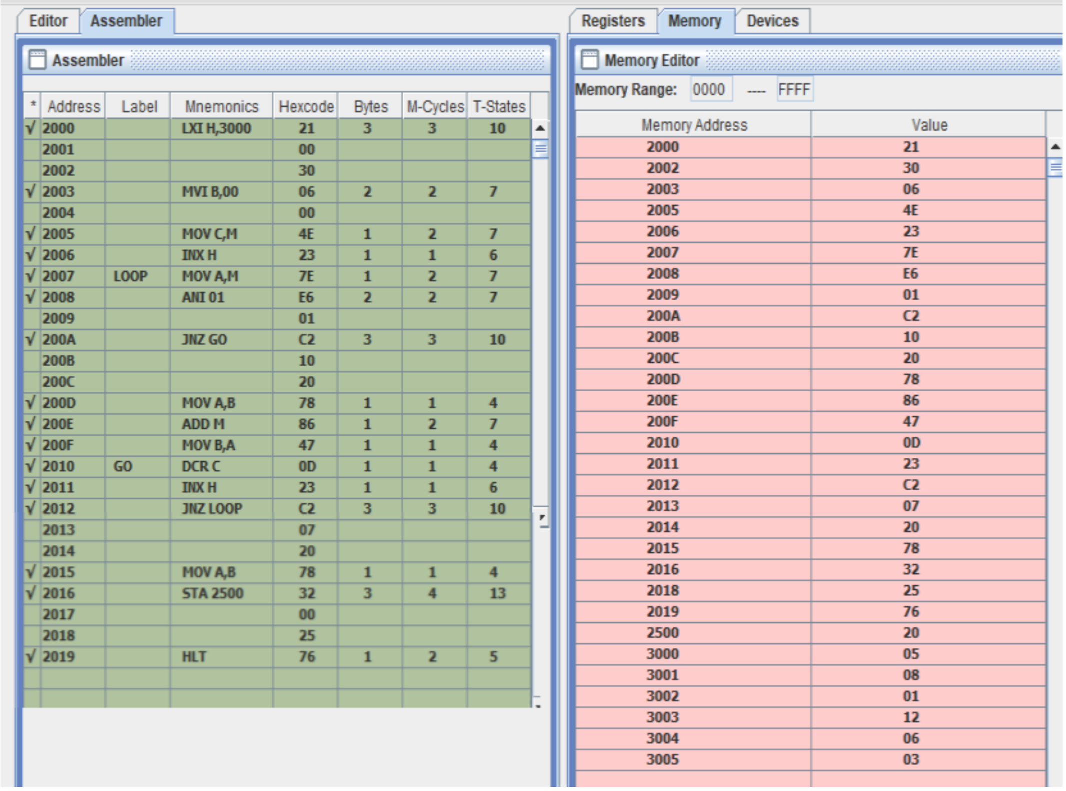
Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	06
2005	4E
2006	23
2007	7E
2008	E6
2009	01
200A	CA
200B	0E
200C	20
200D	04
200E	0D
200F	23
2010	C2
2011	07
2012	20
2013	78
2014	32
2016	25
2017	76
2500	02
3000	05
3001	08
3002	01
3003	12
3004	06
3005	03

```
# ORG 3000H
# DB 05H,08H,01H,12H,06H,03H

// INPUT -> 3000H = 05H (size of array), (elements in array) 3001H = 08H, 3002H =
01H, 3003H = 12H, 3004H = 06H 3005H = 03H
// OUTPUT -> 2500H = 02H
```

```
P9) Calculate sum of all even numbers in the given array

Algorithm/Comments :
Load Memory Location of the size of array in HL register pair
Initialize B with 0
Move size of array value (in M) to C
Increment H (to access array elements)

Starting of loop - Move element to A
Apply AND with 01H on A
if A is zero add current element to B
else do nothing
Decrement C , Increment H continue loop till C is non-zero

Move B to A
Store value in A(sum of even numbers) to desired memory location (Here at 2500H)
```

```
Code :

# ORG 2000H
# BEGIN 2000H

            LXI H,3000H
            MVI B,00H
            MOV C,M

            INX H
LOOP: MOV A,M
            ANI 01H
            JNZ GO
            MOV A,B
            ADD M
            MOV B,A

GO:         DCR C
            INX H
            JNZ LOOP

            MOV A,B
            STA 2500H
            HLT
```



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MVI B,00	06	2	2	7
	2004			00			
√	2005		MOV C,M	4E	1	2	7
√	2006		INX H	23	1	1	6
√	2007	LOOP	MOV A,M	7E	1	2	7
√	2008		ANI 01	E6	2	2	7
	2009			01			
√	200A		JNZ GO	C2	3	3	10
	200B			10			
	200C			20			
√	200D		MOV A,B	78	1	1	4
√	200E		ADD M	86	1	2	7
√	200F		MOV B,A	47	1	1	4
√	2010	GO	DCR C	0D	1	1	4
√	2011		INX H	23	1	1	6
√	2012		JNZ LOOP	C2	3	3	10
	2013			07			
	2014			20			
√	2015		MOV A,B	78	1	1	4
√	2016		STA 2500	32	3	4	13
	2017			00			
	2018			25			
√	2019		HLT	76	1	2	5

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	06
2005	4E
2006	23
2007	7E
2008	E6
2009	01
200A	C2
200B	10
200C	20
200D	78
200E	86
200F	47
2010	0D
2011	23
2012	C2
2013	07
2014	20
2015	78
2016	32
2018	25
2019	76
2500	20
3000	05
3001	08
3002	01
3003	12
3004	06
3005	03

```
# ORG 3000H
# DB 05H,08H,01H,12H,06H,03H

// INPUT -> 3000H = 05H (size of array), (elements in array) 3001H = 08H, 3002H =
01H, 3003H = 12H, 3004H = 06H 3005H = 03H
// OUTPUT ->  2500H = 20H (08H + 12H + 06H)
```

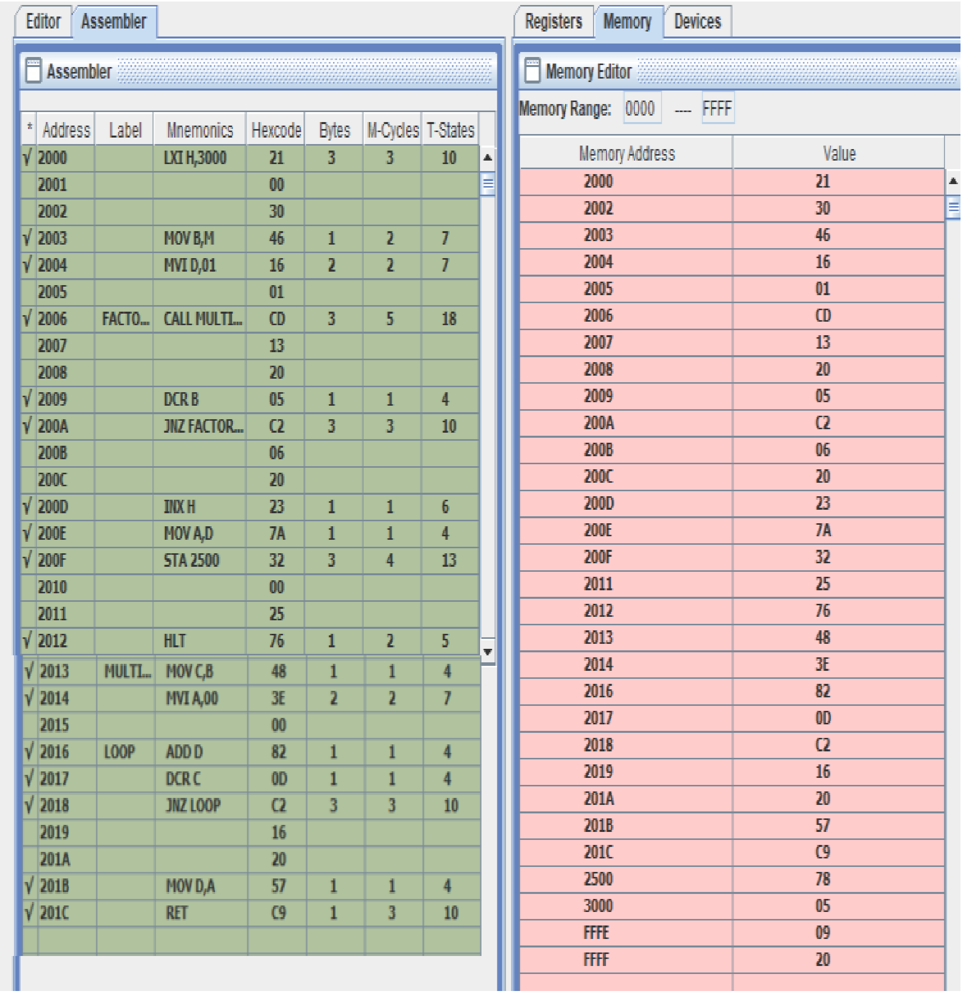
```
P10) Find Factorial of a number

Algorithm/Comments :
Load Memory Location of the number in HL register pair
Move the value to B
Initialize D with 01H

Start of factorial - call MULTIPLY ( like function calling in C or C++)
```

Decrement B and keep going to factorial till B is non-zero
Move value in D to A
Store value in A(factorial) to the desired location(Here at 2500H)

// Description of MULTIPLY
Move value in B to C
Initialize A with 00H
Start of loop - Add D to A
decrement C and loop till C is non-zero
Move A to D
return

Code :

```
# ORG 2000H
# BEGIN 2000H

                  LXI H,3000H
                  MOV B,M
                  MVI D,01H

FACTORIAL:  CALL MULTIPLY
                  DCR B
                  JNZ FACTORIAL
                  INX H
                  MOV A,D
                  STA 2500H
                  HLT

MULTIPLY:   MOV C,B
                  MVI A,00H

LOOP:         ADD D
                  DCR C
                  JNZ LOOP
                  MOV D,A
                  RET

# ORG 3000H
# DB 05H

// INPUT -> 3000H = 05H
// OUTPUT -> 2500H = 78H
```



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI H,3000	21	3	3	10
	2001			00			
	2002			30			
√	2003		MOV B,M	46	1	2	7
√	2004		MVI D,01	16	2	2	7
	2005			01			
√	2006	FACTO...	CALL MULTI...	CD	3	5	18
	2007			13			
	2008			20			
√	2009		DCR B	05	1	1	4
√	200A		JNZ FACTOR...	C2	3	3	10
	200B			06			
	200C			20			
√	200D		INX H	23	1	1	6
√	200E		MOV A,D	7A	1	1	4
√	200F		STA 2500	32	3	4	13
	2010			00			
	2011			25			
√	2012		HLT	76	1	2	5
√	2013	MULTI...	MOV C,B	48	1	1	4
√	2014		MVI A,00	3E	2	2	7
	2015			00			
√	2016	LOOP	ADD D	82	1	1	4
√	2017		DCR C	0D	1	1	4
√	2018		JNZ LOOP	C2	3	3	10
	2019			16			
	201A			20			
√	201B		MOV D,A	57	1	1	4
√	201C		RET	C9	1	3	10

Memory Range: 0000 ---- FFFF

Memory Address	Value
2000	21
2002	30
2003	46
2004	16
2005	01
2006	CD
2007	13
2008	20
2009	05
200A	C2
200B	06
200C	20
200D	23
200E	7A
200F	32
2011	25
2012	76
2013	48
2014	3E
2016	82
2017	0D
2018	C2
2019	16
201A	20
201B	57
201C	C9
2500	78
3000	05
FFFE	09
FFFF	20