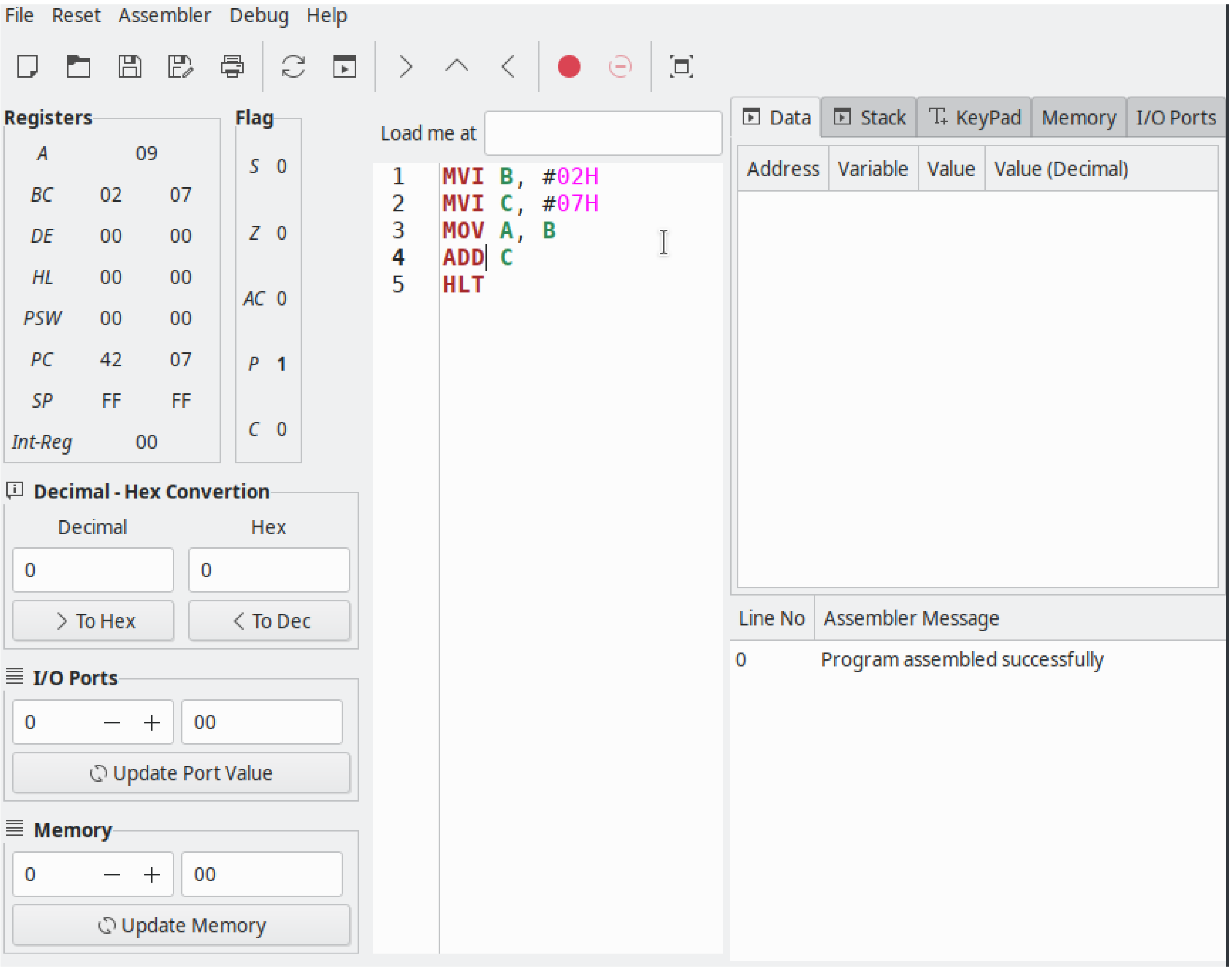
DIGITAL LOGIC AND MICROPROCESSOR

Registration Number:	**18BIT0272**
Name:	PRIYAL BHARDWAJ
Slot:	L7+L8
Experiment Name:	8085 Microprocessor Programming

1. Write an ALP for adding two 8-bit numbers 02H and 07H stored in registers B and C respectively.



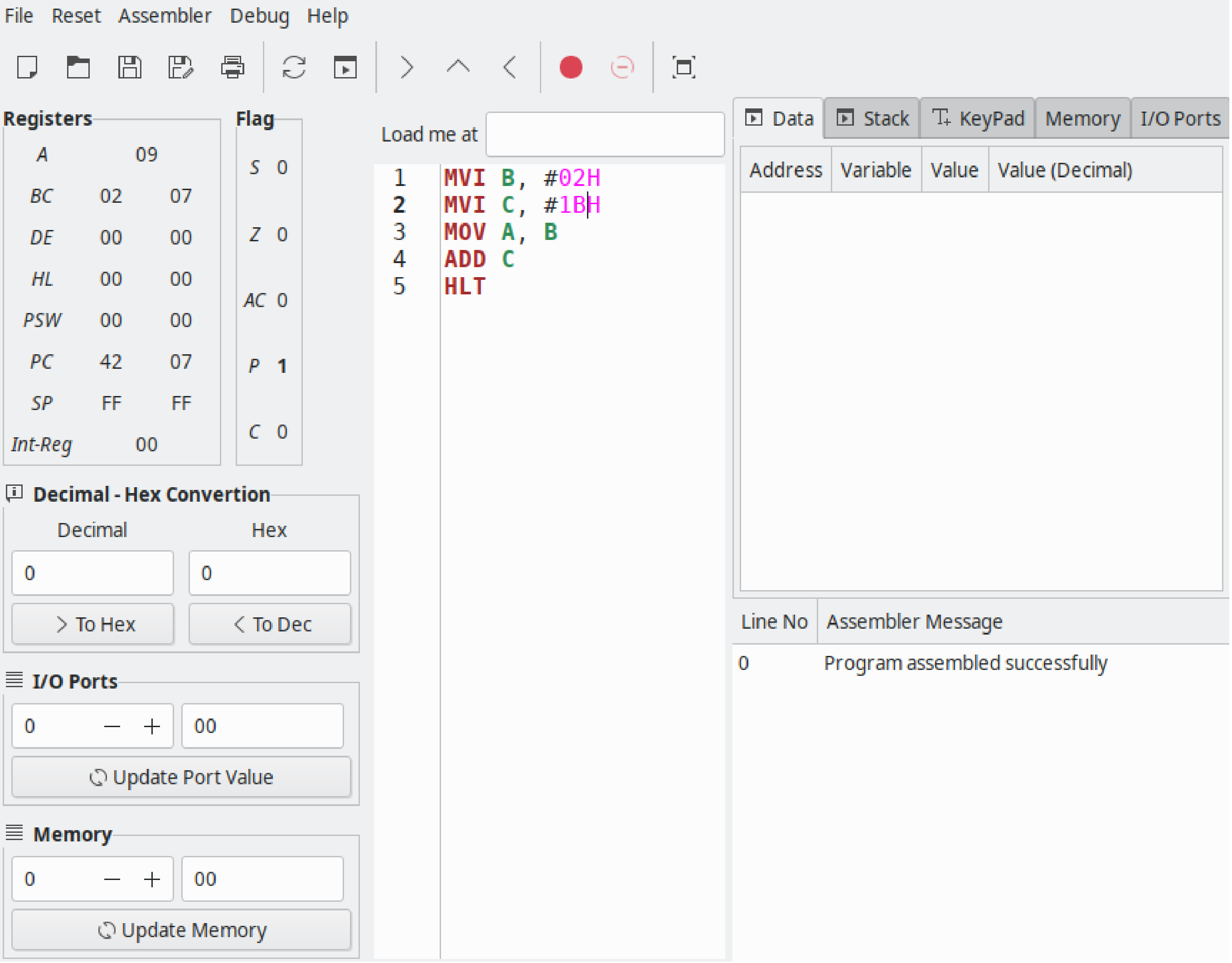
Registers		
A	09	
BC	02	07
DE	00	00
HL	00	00
PSW	00	00
PC	42	07
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	1
C	0

```
MVI B, #02H
MVI C, #07H
MOV A, B
ADD C
HLT
```

Line No	Assembler Message
0	Program assembled successfully

2. Write an ALP for adding two 8-bit numbers 02H and 1BH stored in registers B and C respectively with carry.



Registers

A	09	
BC	02	07
DE	00	00
HL	00	00
PSW	00	00
PC	42	07
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	1
C	0

```
MVI B, #02H
MVI C, #1BH
MOV A, B
ADD C
HLT
```

Decimal: 0 Hex: 0

I/O Ports: 0 00

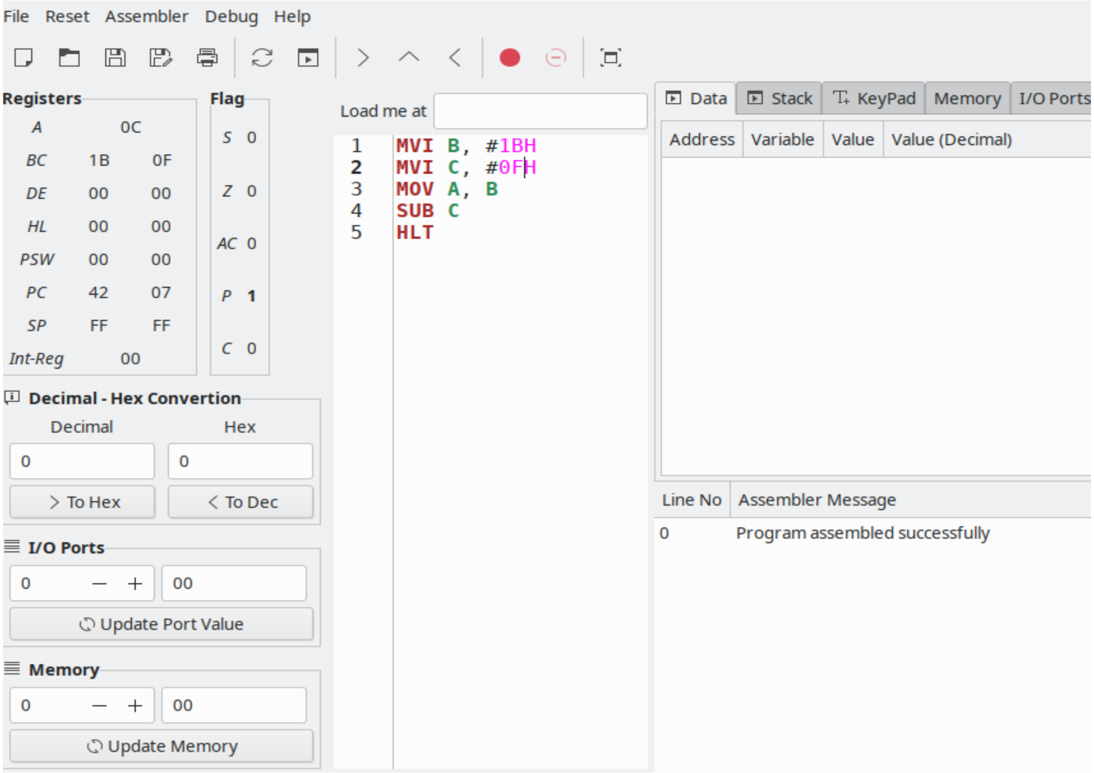
Memory: 0 00

Address	Variable	Value	Value (Decimal)

Line No	Assembler Message
0	Program assembled successfully

3. Write an ALP for subtracting two 8-bit numbers stored in registers with and without borrow.

Without borrow:



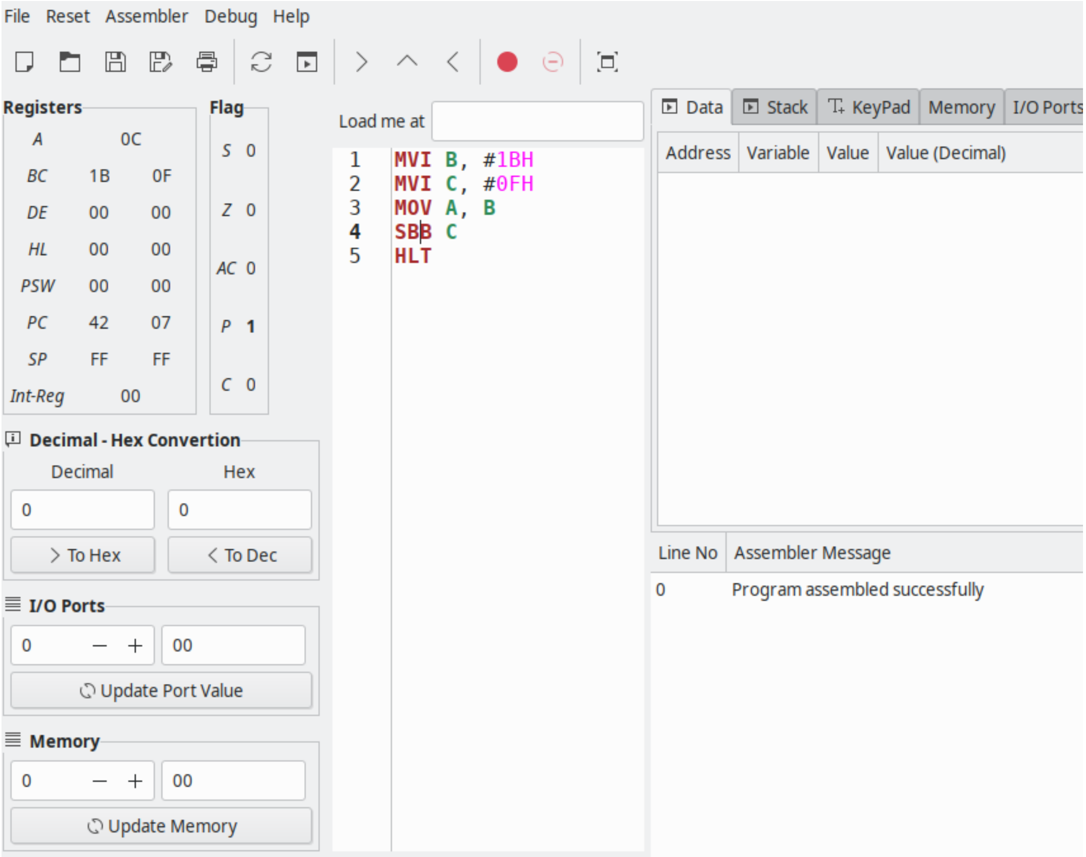
```
MVI B, #1BH
MVI C, #0FH
MOV A, B
SUB C
HLT
```

A	0C	
BC	1B	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	07
SP	FF	FF
Int-Reg	00	

S	0
Z	0
AC	0
P	1
C	0

Line No	Assembler Message
0	Program assembled successfully

With borrow:



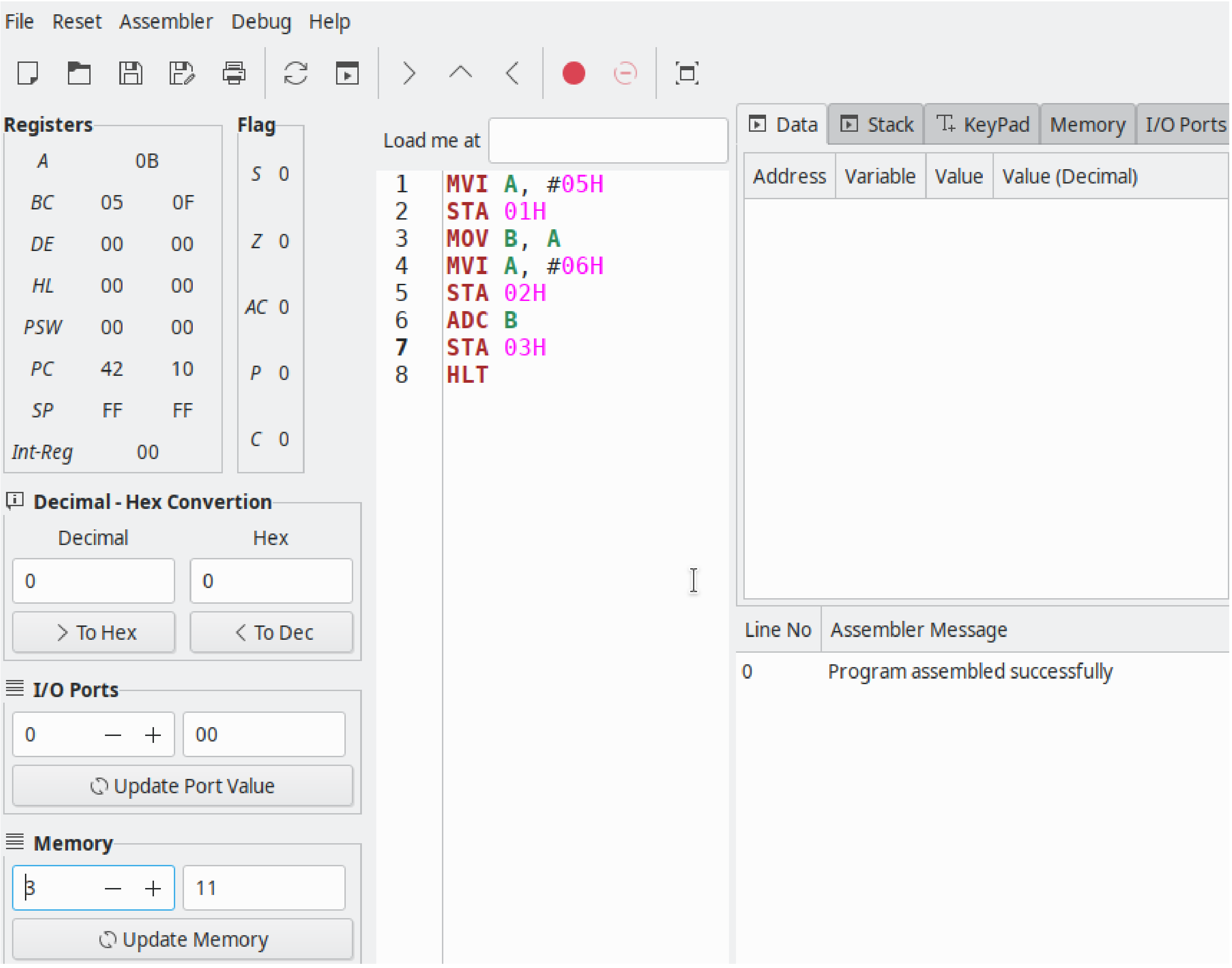
```
MVI B, #1BH
MVI C, #0FH
MOV A, B
SBB C
HLT
```

A	0C	
BC	1B	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	07
SP	FF	FF
Int-Reg	00	

S	0
Z	0
AC	0
P	1
C	0

Line No	Assembler Message
0	Program assembled successfully

4. Write an ALP for storing two 8-bit data in memory addresses 01 and 02 respectively, and perform addition of these 2 data and store it in memory address 03.



```
MVI A, #05H
STA 01H
MOV B, A
MVI A, #06H
STA 02H
ADC B
STA 03H
HLT
```

Registers

A	0B	
BC	05	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	10
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	0

Decimal - Hex Convertion

Decimal	Hex
0	0

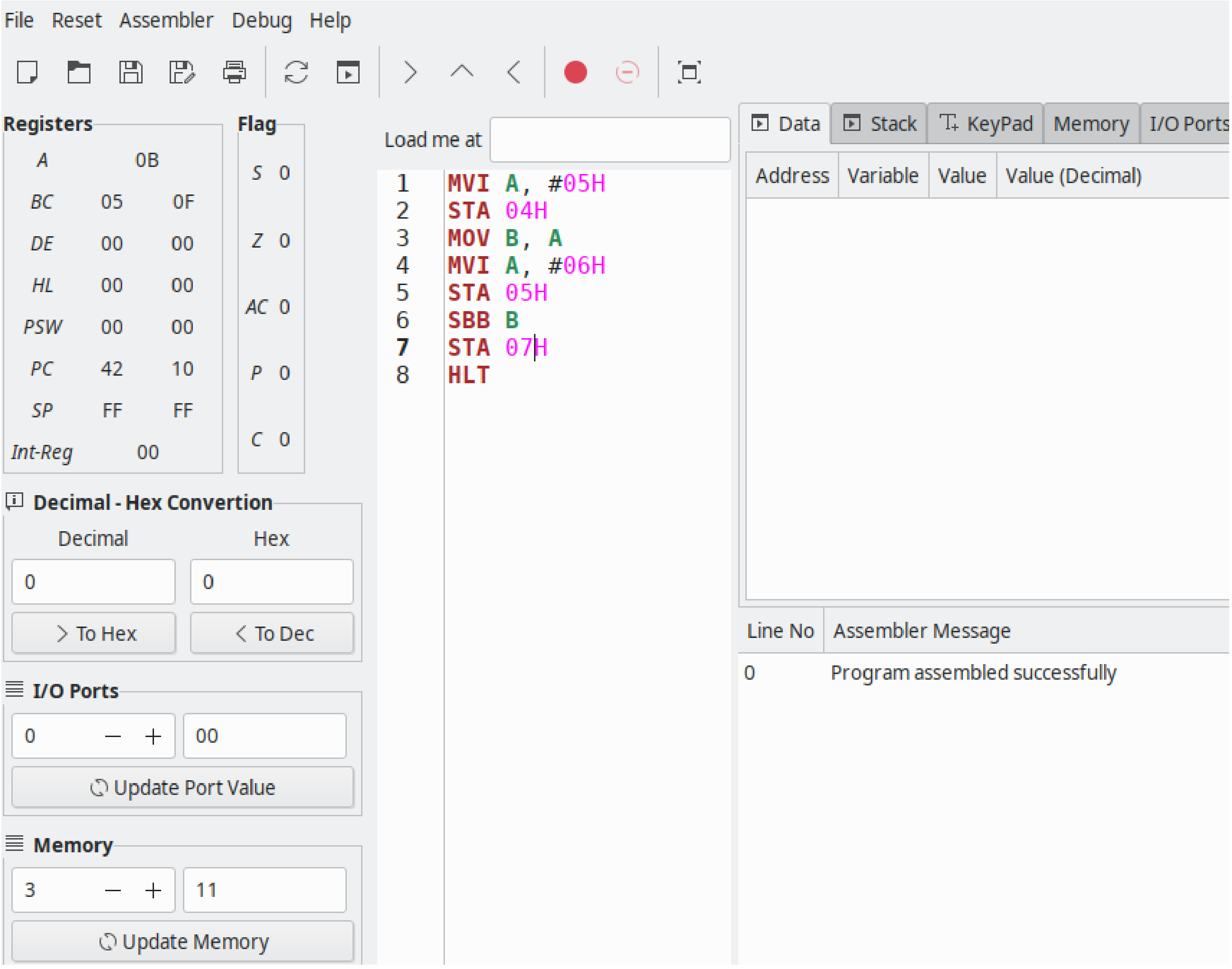
I/O Ports: 0 — + 00

Memory: 3 — + 11

Address	Variable	Value	Value (Decimal)

Line No	Assembler Message
0	Program assembled successfully

5. Write an ALP for storing two 8-bit data in memory addresses 04 and 05 respectively, and perform addition of these 2 data and store it in memory address 07.



Load me at

```
MVI A, #05H
STA 04H
MOV B, A
MVI A, #06H
STA 05H
SBB B
STA 07H
HLT
```

Registers

A	0B	
BC	05	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	10
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	0

Decimal	Hex
0	0

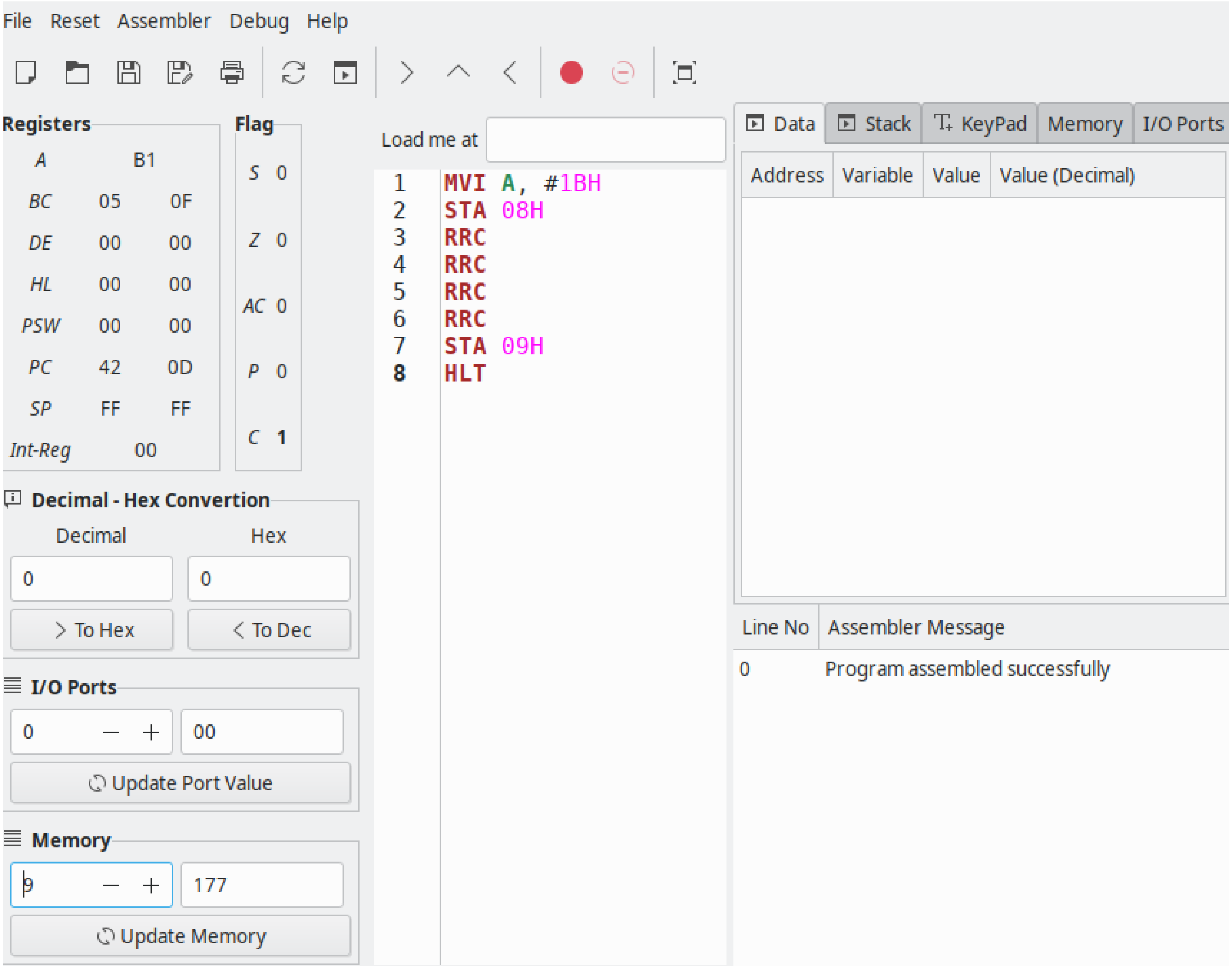
I/O Ports: 0 — + 00

Memory: 3 — + 11

Address	Variable	Value	Value (Decimal)

Line No	Assembler Message
0	Program assembled successfully

6. Write and ALP for rotating the data stored in memory address 08 four times towards the right and store the result in memory address 09.



Registers

A	B1	
BC	05	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	0D
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	1

Load me at

```
MVI A, #1BH
STA 08H
RRC
RRC
RRC
RRC
STA 09H
HLT
```

Address	Variable	Value	Value (Decimal)

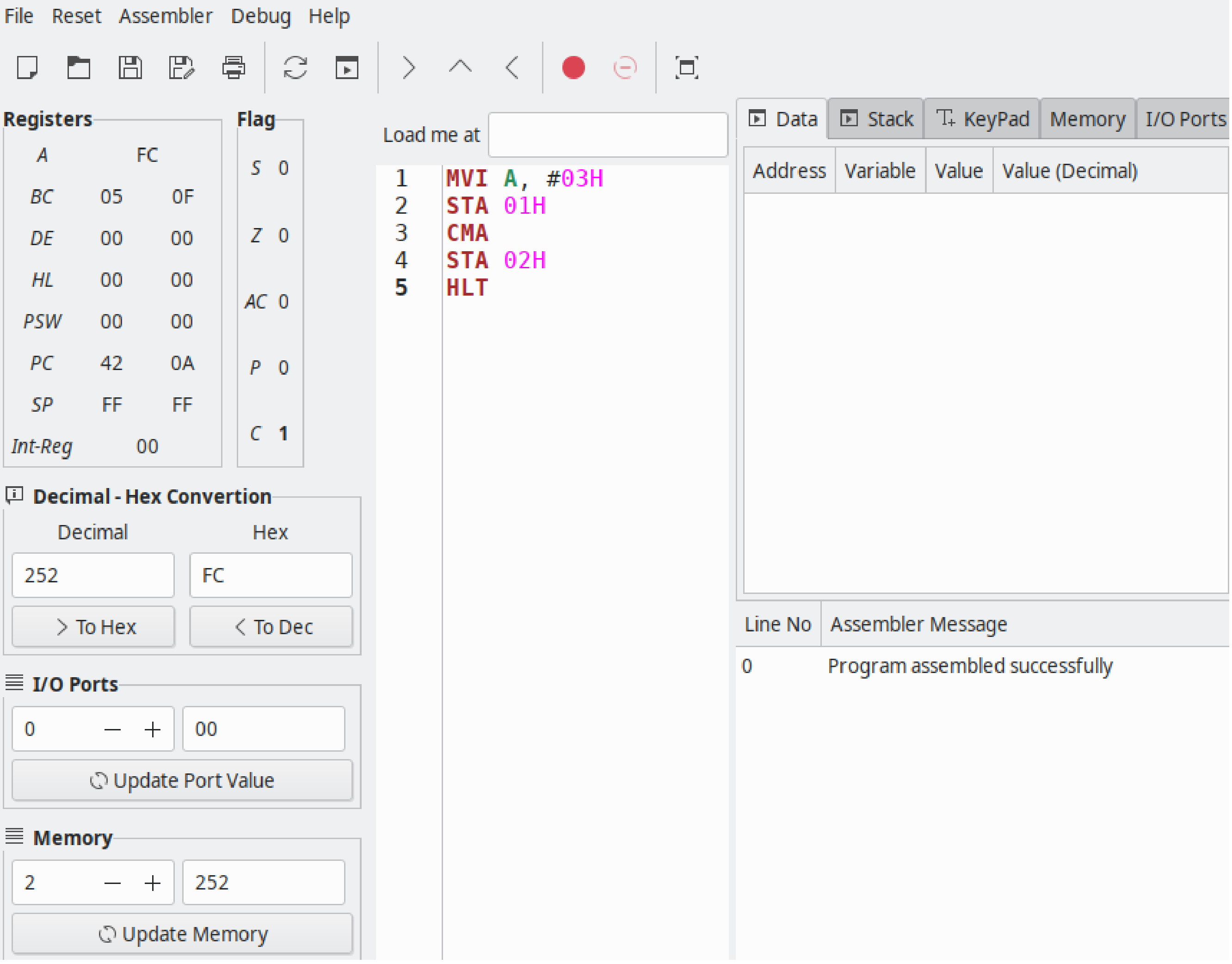
Line No	Assembler Message
0	Program assembled successfully

Decimal	Hex
0	0

I/O Ports: 0 – + 00

Memory: 9 – + 177

7. Write an ALP for finding 1's complement of data stored in an address 01 and store the resulting data in 02.



Registers

A	FC	
BC	05	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	0A
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	1

Load me at

```
MVI A, #03H
STA 01H
CMA
STA 02H
HLT
```

Decimal - Hex Convertion

Decimal	Hex
252	FC

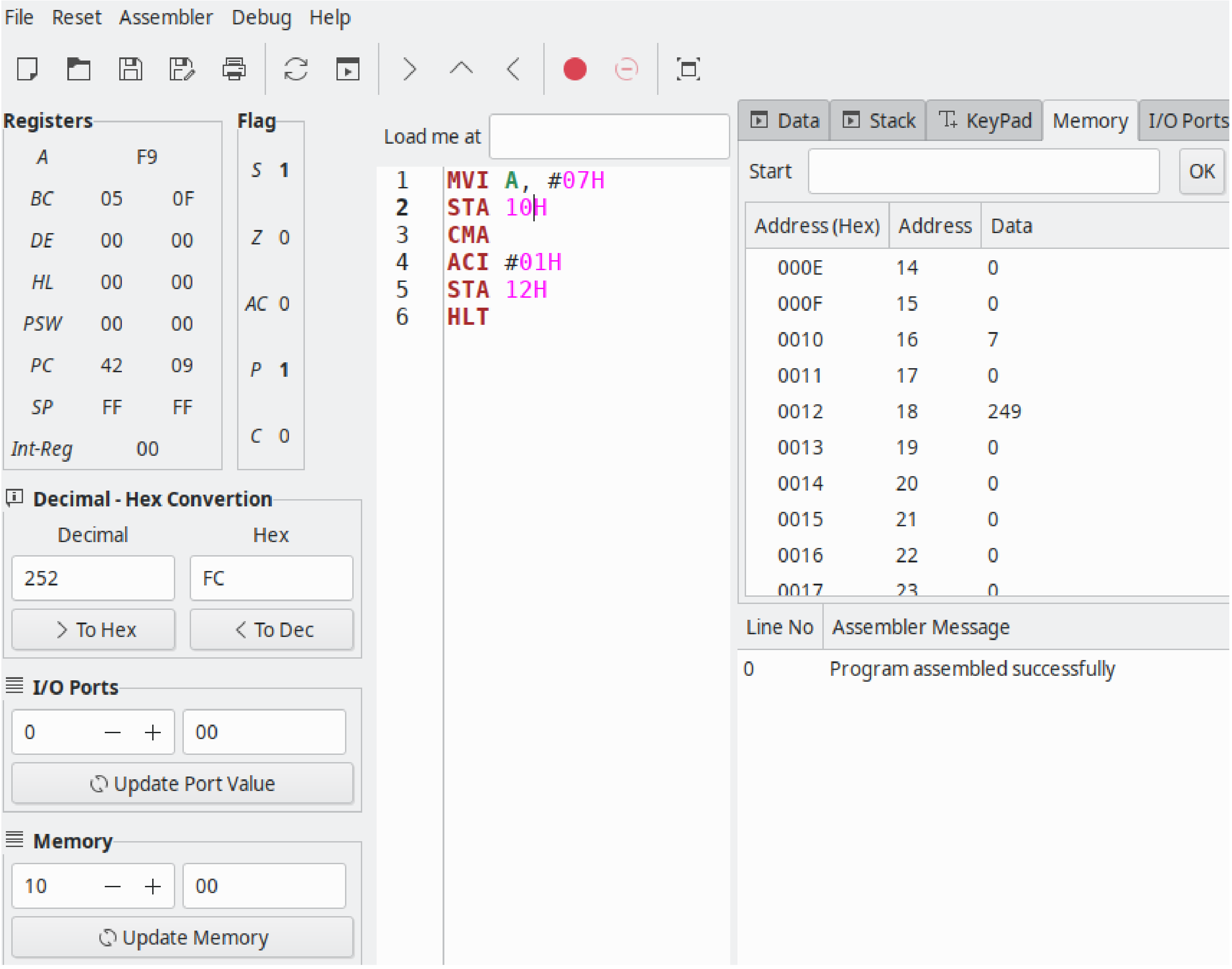
I/O Ports: 0 — + 00

Memory: 2 — + 252

Address	Variable	Value	Value (Decimal)

Line No	Assembler Message
0	Program assembled successfully

8. Write an ALP for finding 2's complement of data stored in an address 10 and store the resulting data in 12.



Load me at

```
MVI A, #07H
STA 10H
CMA
ACI #01H
STA 12H
HLT
```

Registers

A	F9	
BC	05	0F
DE	00	00
HL	00	00
PSW	00	00
PC	42	09
SP	FF	FF
Int-Reg	00	

Flag

S	1
Z	0
AC	0
P	1
C	0

Decimal - Hex Convertion

Decimal	Hex
252	FC

I/O Ports

0 — + 00

Memory

10 — + 00

Start

Address (Hex)	Address	Data
000E	14	0
000F	15	0
0010	16	7
0011	17	0
0012	18	249
0013	19	0
0014	20	0
0015	21	0
0016	22	0
0017	23	0

Line No	Assembler Message
0	Program assembled successfully

(GNU8085) Date ....

Expt. No. 5. 8085 Microprocessor Programming Page No. 20

1. Write an ALP for adding two 8-bit numbers 02H, 07H stored in registers B and C respectively.

```
MVI  B, #02H
MVI  C, #07H
MOV  A, B
ADD  C
HLT
```

2. Write an ALP for adding two 8 bit numbers 02H, 1BH stored in registers B and C with carry.

```
MVI  B, # 02H
MVI  C, # 1BH
MOV  A, C
ADC  B
HLT
```

3. Write an ALP for subtracting two 8 bit numbers stored in registers with and without borrow

```
(a)   MVI  C, #08H            (b)   MVI  C, # 08H
      MVI  B, # 03H                 MVI  B, # 04H
      MOV  A, C                     MOV  A, B
      SUB  B                        SBB  C
      HLT                           HLT

   (without borrow)               (with borrow)
```

Teacher's Signature : ____________

Date ______________

Expt. No. ______________ Page No. __________

4. Write an ALP for storing the 8 bit data in memory address 01 & 02. Then perform addition of these two numbers and store the result in the memory address 03.

```
MVI  A, #05H
STA  01 H
Mov  B, A
MVI  A, # 06H
STA  02H
ADC  B
STA  03H
HLT
```

5. Write an ALP for storing the 8-bit data in memory address 04 & 05. Then perform subtraction of these 2 numbers & store the result in memory address 07.

```
MVI  A, #05H
STA  04 H
Mov  B,A
MVI  A, #06H
STA  05H
SBB  B
STA  07 H
HLT
```

6. Write an ALP for rotating the data available in memory address 08 four times towards right & store the result in memory address 09.

Teacher's Signature ______________

Date ....

Expt. No. .... Page No. ....

→
```
MVI  A, #1BH
STA  08H
RRC
RRC
RRC
RRC
STA 09H
HLT
```

7. Write an ALP for finding the 1s complement of data stored in an address 01, and store the result in memory address 02.

→
```
MVI. A, #03H
STA 01H
CMA
STA 02H
HLT
```

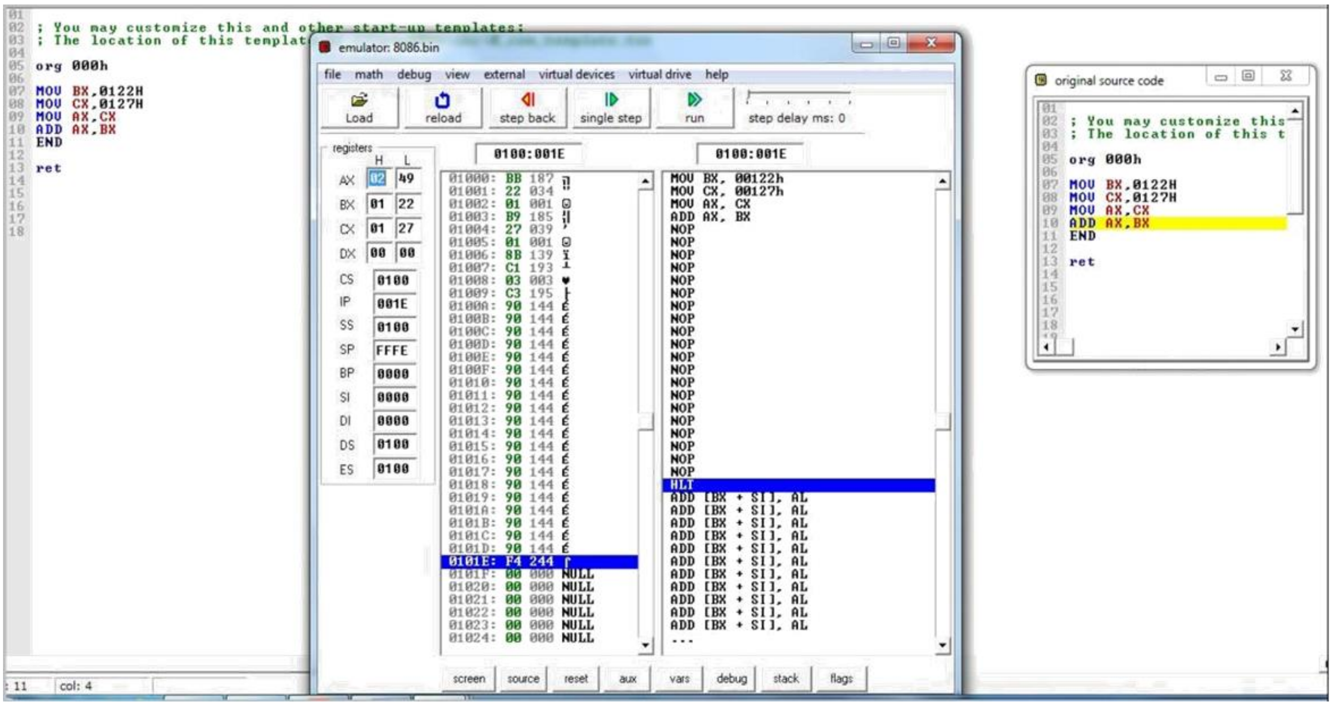
8. Write an ALP for finding the 2s complement of data stored in an address 10 and store the result in memory address 12.

→
```
MVI A, #07H
STA 10H
CMA
ACI, ~~0~~#01H
STA 12H
HLT
```

Teacher's Signature : ____________

Experiment Name: 8086 Microprocessor Programming

1. Write an ALP for adding two 16 bit numbers 0122H, 0127H stored in registers B and C respectively.



```
; You may customize this and other start-up templates;
; The location of this templat

org 000h

MOV BX,0122H
MOV CX,0127H
MOV AX,CX
ADD AX,BX
END

ret
```

emulator: 8086.bin

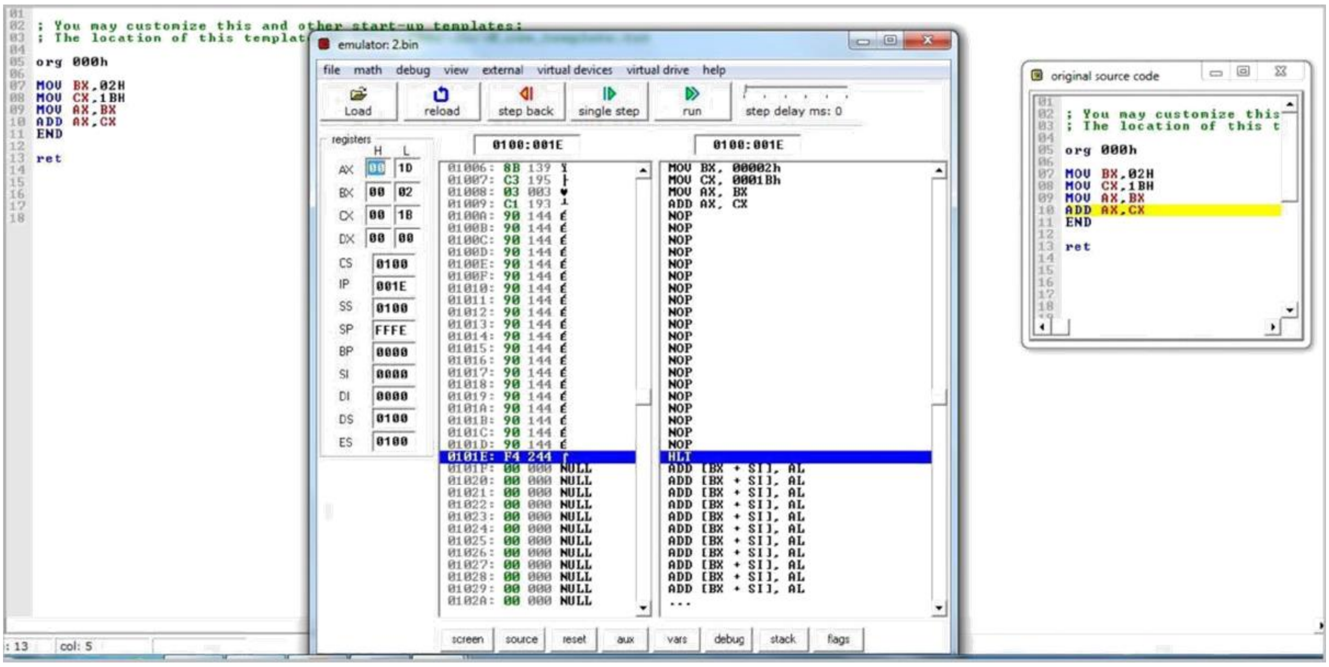
```
    H  L
AX  02 49
BX  01 22
CX  01 27
DX  00 00
CS  0100
IP  001E
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:001E

01000: BB 187 ╗
01001: 22 034 "
01002: 01 001 ☺
01003: B9 185 ╣
01004: 27 039 '
01005: 01 001 ☺
01006: 8B 139 ï
01007: C1 193 ┴
01008: 03 003 ♥
01009: C3 195 ├
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: 90 144 É
0101D: 90 144 É
0101E: F4 244 ⌠
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
```

```
0100:001E

MOV BX, 00122h
MOV CX, 00127h
MOV AX, CX
ADD AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
; You may customize this
; The location of this t

org 000h

MOV BX,0122H
MOV CX,0127H
MOV AX,CX
ADD AX,BX
END

ret
```

2. Write an ALP for adding two 8 bit numbers 02H, 1BH stored in registers B and C.



```
; You may customize this and other start-up templates;
; The location of this templat

org 000h

MOV BX,02H
MOV CX,1BH
MOV AX,BX
ADD AX,CX
END

ret
```

emulator: 2.bin

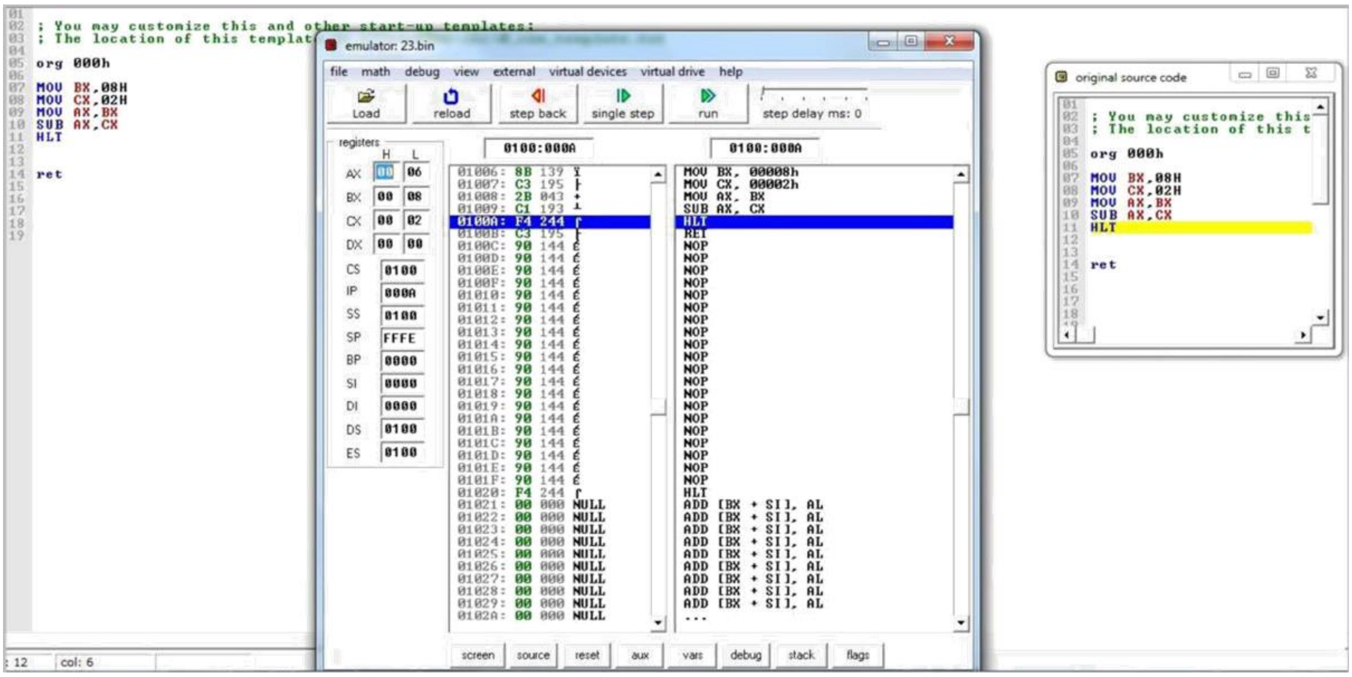
```
registers   H   L
AX         00  1D
BX         00  02
CX         00  1B
DX         00  00
CS         0100
IP         001E
SS         0100
SP         FFFE
BP         0000
SI         0000
DI         0000
DS         0100
ES         0100
```

```
0100:001E                          0100:001E
01006: 8B 139 ï                    MOV BX, 00002h
01007: C3 195 ├                    MOV CX, 0001Bh
01008: 03 003 ♥                    MOV AX, BX
01009: C1 193 ┴                    ADD AX, CX
0100A: 90 144 É                    NOP
0100B: 90 144 É                    NOP
0100C: 90 144 É                    NOP
0100D: 90 144 É                    NOP
0100E: 90 144 É                    NOP
0100F: 90 144 É                    NOP
01010: 90 144 É                    NOP
01011: 90 144 É                    NOP
01012: 90 144 É                    NOP
01013: 90 144 É                    NOP
01014: 90 144 É                    NOP
01015: 90 144 É                    NOP
01016: 90 144 É                    NOP
01017: 90 144 É                    NOP
01018: 90 144 É                    NOP
01019: 90 144 É                    NOP
0101A: 90 144 É                    NOP
0101B: 90 144 É                    NOP
0101C: 90 144 É                    NOP
0101D: 90 144 É                    NOP
0101E: F4 244 ſ                    HLT
0101F: 00 000 NULL                 ADD [BX + SI], AL
01020: 00 000 NULL                 ADD [BX + SI], AL
01021: 00 000 NULL                 ADD [BX + SI], AL
01022: 00 000 NULL                 ADD [BX + SI], AL
01023: 00 000 NULL                 ADD [BX + SI], AL
01024: 00 000 NULL                 ADD [BX + SI], AL
01025: 00 000 NULL                 ADD [BX + SI], AL
01026: 00 000 NULL                 ADD [BX + SI], AL
01027: 00 000 NULL                 ADD [BX + SI], AL
01028: 00 000 NULL                 ADD [BX + SI], AL
01029: 00 000 NULL                 ADD [BX + SI], AL
0102A: 00 000 NULL                 ...
```

original source code

```
; You may customize this
; The location of this t

org 000h

MOV BX,02H
MOV CX,1BH
MOV AX,BX
ADD AX,CX
END

ret
```

3. Write an ALP for subtracting two 8 bit numbers stored in registers



```
; You may customize this and other start-up templates;
; The location of this templat

org 000h

MOV BX,08H
MOV CX,02H
MOV AX,BX
SUB AX,CX
HLT


ret
```

emulator: 23.bin

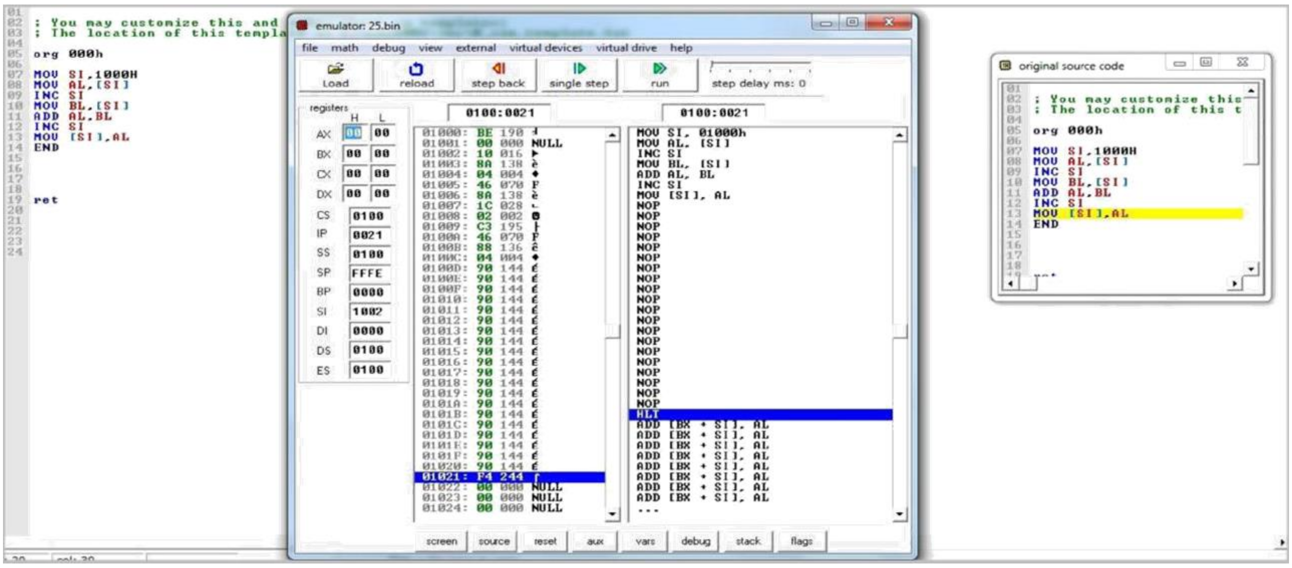
```
registers   H   L
AX         00  06
BX         00  08
CX         00  02
DX         00  00
CS         0100
IP         000A
SS         0100
SP         FFFE
BP         0000
SI         0000
DI         0000
DS         0100
ES         0100
```

```
0100:000A                          0100:000A
01006: 8B 139 ï                    MOV BX, 00008h
01007: C3 195 ├                    MOV CX, 00002h
01008: 2B 043 +                    MOV AX, BX
01009: C1 193 ┴                    SUB AX, CX
0100A: F4 244 ſ                    HLT
0100B: C3 195 ├                    RET
0100C: 90 144 É                    NOP
0100D: 90 144 É                    NOP
0100E: 90 144 É                    NOP
0100F: 90 144 É                    NOP
01010: 90 144 É                    NOP
01011: 90 144 É                    NOP
01012: 90 144 É                    NOP
01013: 90 144 É                    NOP
01014: 90 144 É                    NOP
01015: 90 144 É                    NOP
01016: 90 144 É                    NOP
01017: 90 144 É                    NOP
01018: 90 144 É                    NOP
01019: 90 144 É                    NOP
0101A: 90 144 É                    NOP
0101B: 90 144 É                    NOP
0101C: 90 144 É                    NOP
0101D: 90 144 É                    NOP
0101E: 90 144 É                    NOP
0101F: 90 144 É                    NOP
01020: F4 244 ſ                    HLT
01021: 00 000 NULL                 ADD [BX + SI], AL
01022: 00 000 NULL                 ADD [BX + SI], AL
01023: 00 000 NULL                 ADD [BX + SI], AL
01024: 00 000 NULL                 ADD [BX + SI], AL
01025: 00 000 NULL                 ADD [BX + SI], AL
01026: 00 000 NULL                 ADD [BX + SI], AL
01027: 00 000 NULL                 ADD [BX + SI], AL
01028: 00 000 NULL                 ADD [BX + SI], AL
01029: 00 000 NULL                 ADD [BX + SI], AL
0102A: 00 000 NULL                 ...
```

original source code

```
; You may customize this
; The location of this t

org 000h

MOV BX,08H
MOV CX,02H
MOV AX,BX
SUB AX,CX
HLT


ret
```

4. Write an ALP for storing the 16bit data in memory address 01 and 02. Then perform addition of these two numbers and store the result in the memory address 03.



```
; You may customize this and
; The location of this templa

org 000h

MOV SI,1000H
MOV AL,[SI]
INC SI
MOV BL,[SI]
ADD AL,BL
INC SI
MOV [SI],AL
END




ret
```

```
     H   L
AX  00  00
BX  00  00
CX  00  00
DX  00  00
CS  0100
IP  0021
SS  0100
SP  FFFE
BP  0000
SI  1002
DI  0000
DS  0100
ES  0100
```

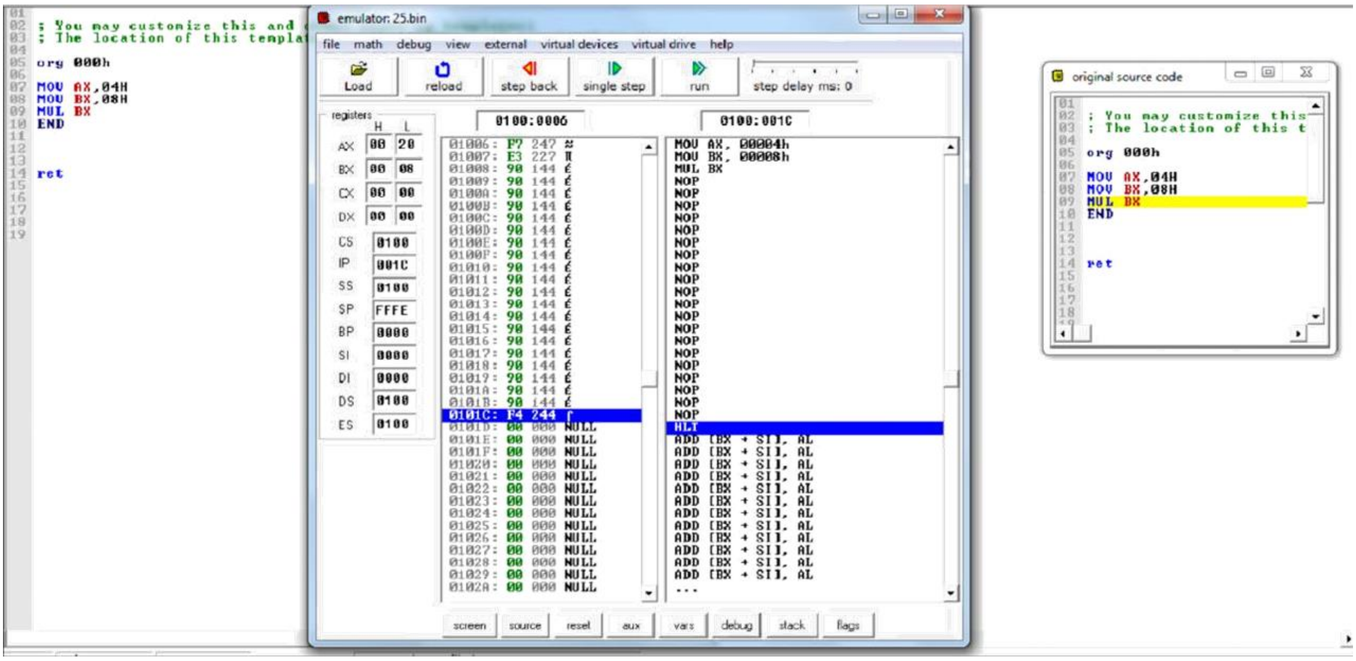
```
0100:0021

01000: BE 190 ╛
01001: 00 000 NULL
01002: 10 016 ►
01003: 8A 138 è
01004: 04 004 ♦
01005: 46 070 F
01006: 8A 138 è
01007: 1C 028 ∟
01008: 02 002 ☻
01009: C3 195 ├
0100A: 46 070 F
0100B: 88 136 ê
0100C: 04 004 ♦
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: 90 144 É
0101D: 90 144 É
0101E: 90 144 É
0101F: 90 144 É
01020: 90 144 É
01021: F4 244 ⌠
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
```

```
0100:0021

MOV SI, 01000h
MOV AL, [SI]
INC SI
MOV BL, [SI]
ADD AL, BL
INC SI
MOV [SI], AL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
; You may customize this
; The location of this t

org 000h

MOV SI,1000H
MOV AL,[SI]
INC SI
MOV BL,[SI]
ADD AL,BL
INC SI
MOV [SI],AL
END
```

5. Write an ALP for performing multiplication and division of 2 16 bit numbers.

Multiplication:



```
; You may customize this and
; The location of this templat

org 000h

MOV AX,04H
MOV BX,08H
MUL BX
END



ret
```

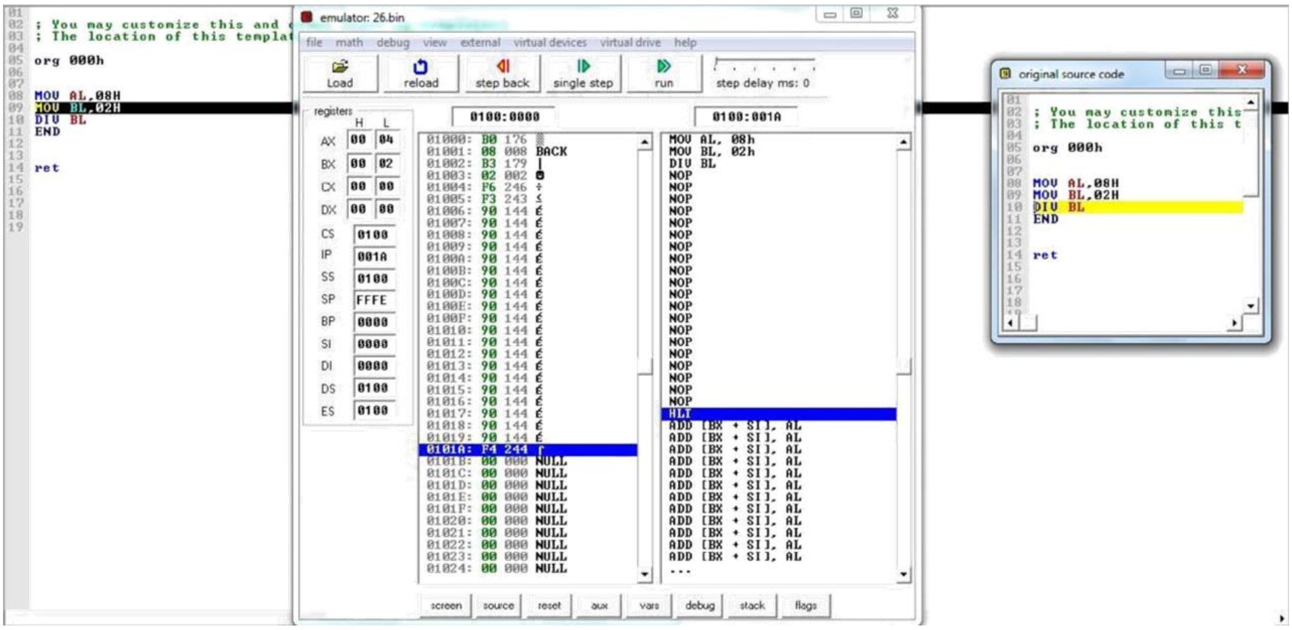
```
     H   L
AX  00  20
BX  00  08
CX  00  00
DX  00  00
CS  0100
IP  001C
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0006

01006: F7 247 ≈
01007: E3 227 π
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: F4 244 ⌠
0101D: 00 000 NULL
0101E: 00 000 NULL
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
01025: 00 000 NULL
01026: 00 000 NULL
01027: 00 000 NULL
01028: 00 000 NULL
01029: 00 000 NULL
0102A: 00 000 NULL
```

```
0100:001C

MOV AX, 00004h
MOV BX, 00008h
MUL BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
; You may customize this
; The location of this t

org 000h

MOV AX,04H
MOV BX,08H
MUL BX
END



ret
```

Division:



```
; You may customize this and 
; The location of this templat

org 000h


MOV AL,08H
MOV BL,02H
DIV BL
END


ret
```

emulator: 26.bin
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0

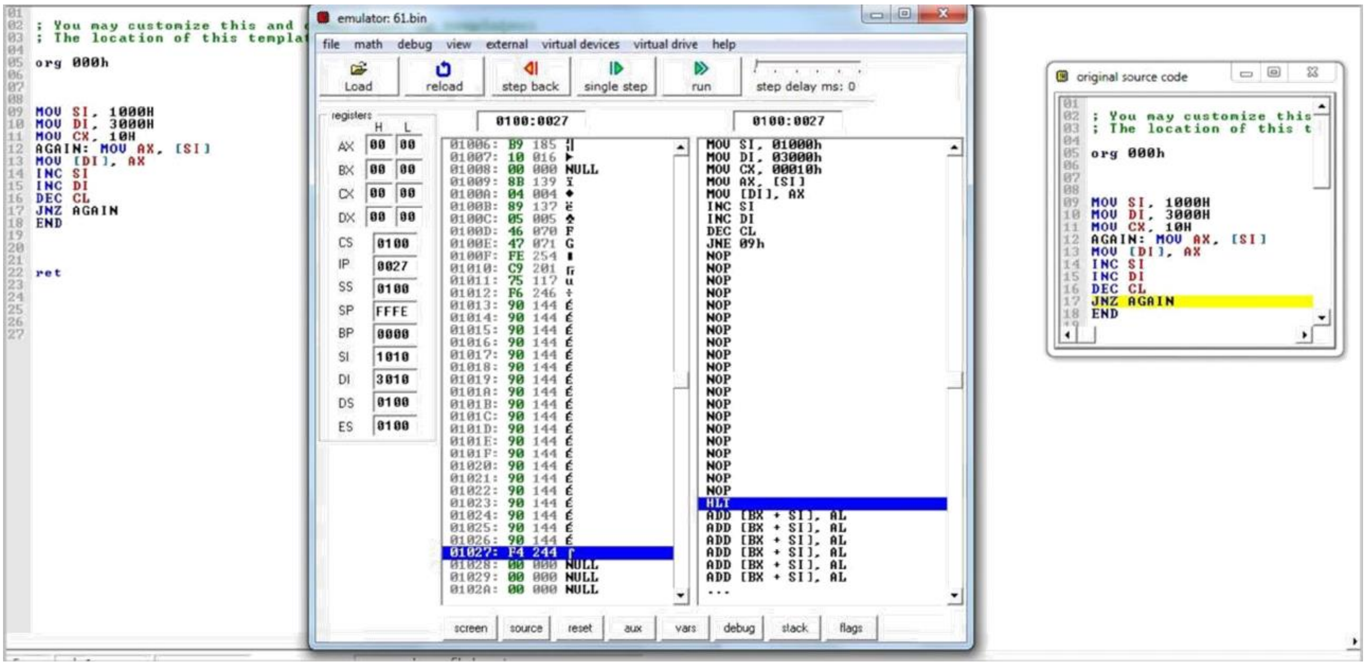
```
registers
     H   L
AX  00  04
BX  00  02
CX  00  00
DX  00  00
CS  0100
IP  001A
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0000
01000: B0 176 ▒
01001: 08 008 BACK
01002: B3 179 │
01003: 02 002 ☻
01004: F6 246 ÷
01005: F3 243 ≤
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: F4 244 ⌠
0101B: 00 000 NULL
0101C: 00 000 NULL
0101D: 00 000 NULL
0101E: 00 000 NULL
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
```

```
0100:001A
MOV AL, 08h
MOV BL, 02h
DIV BL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
; You may customize this
; The location of this t

org 000h


MOV AL,08H
MOV BL,02H
DIV BL
END


ret
```

6. Write an ALP for transferring a block of ten 16 bit data from memory address starting at 1000 to a memory location 3000



```
; You may customize this and 
; The location of this templat

org 000h



MOV SI, 1000H
MOV DI, 3000H
MOV CX, 10H
AGAIN: MOV AX, [SI]
MOV [DI], AX
INC SI
INC DI
DEC CL
JNZ AGAIN
END



ret
```

emulator: 61.bin
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0

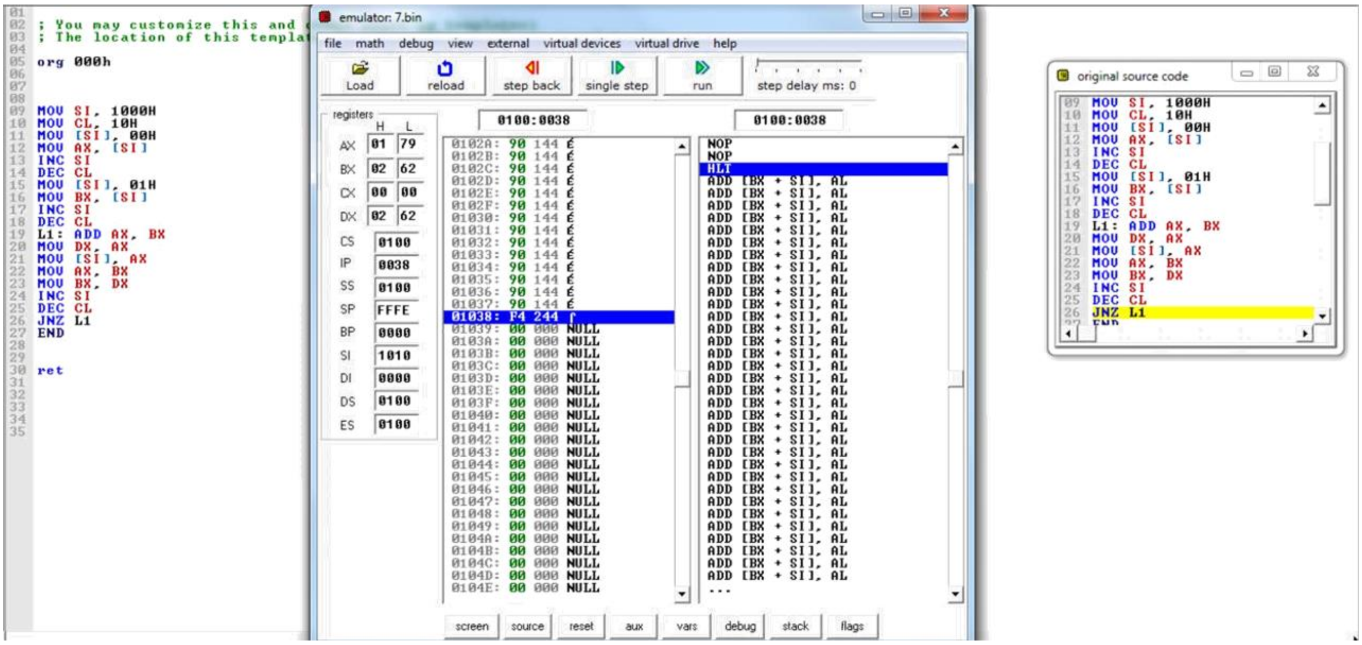
```
registers
     H   L
AX  00  00
BX  00  00
CX  00  00
DX  00  00
CS  0100
IP  0027
SS  0100
SP  FFFE
BP  0000
SI  1010
DI  3010
DS  0100
ES  0100
```

```
0100:0027
01006: B9 185 ╣
01007: 10 016 ►
01008: 00 000 NULL
01009: 8B 139 ï
0100A: 04 004 ♦
0100B: 89 137 ë
0100C: 05 005 ♣
0100D: 46 070 F
0100E: 47 071 G
0100F: FE 254 ■
01010: C9 201 ╔
01011: 75 117 u
01012: F6 246 ÷
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: 90 144 É
0101D: 90 144 É
0101E: 90 144 É
0101F: 90 144 É
01020: 90 144 É
01021: 90 144 É
01022: 90 144 É
01023: 90 144 É
01024: 90 144 É
01025: 90 144 É
01026: 90 144 É
01027: F4 244 ⌠
01028: 00 000 NULL
01029: 00 000 NULL
0102A: 00 000 NULL
```

```
0100:0027
MOV SI, 01000h
MOV DI, 03000h
MOV CX, 00010h
MOV AX, [SI]
MOV [DI], AX
INC SI
INC DI
DEC CL
JNE 09h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
; You may customize this
; The location of this t

org 000h


MOV SI, 1000H
MOV DI, 3000H
MOV CX, 10H
AGAIN: MOV AX, [SI]
MOV [DI], AX
INC SI
INC DI
DEC CL
JNZ AGAIN
END
```

7. Write an ALP to display 13 numbers of the fibonacci series in a memory location.



```
; You may customize this and
; The location of this templat

org 000h


MOV SI, 1000H
MOV CL, 10H
MOV [SI], 00H
MOV AX, [SI]
INC SI
DEC CL
MOV [SI], 01H
MOV BX, [SI]
INC SI
DEC CL
L1: ADD AX, BX
MOV DX, AX
MOV [SI], AX
MOV AX, BX
MOV BX, DX
INC SI
DEC CL
JNZ L1
END


ret
```

emulator: 7.bin
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags

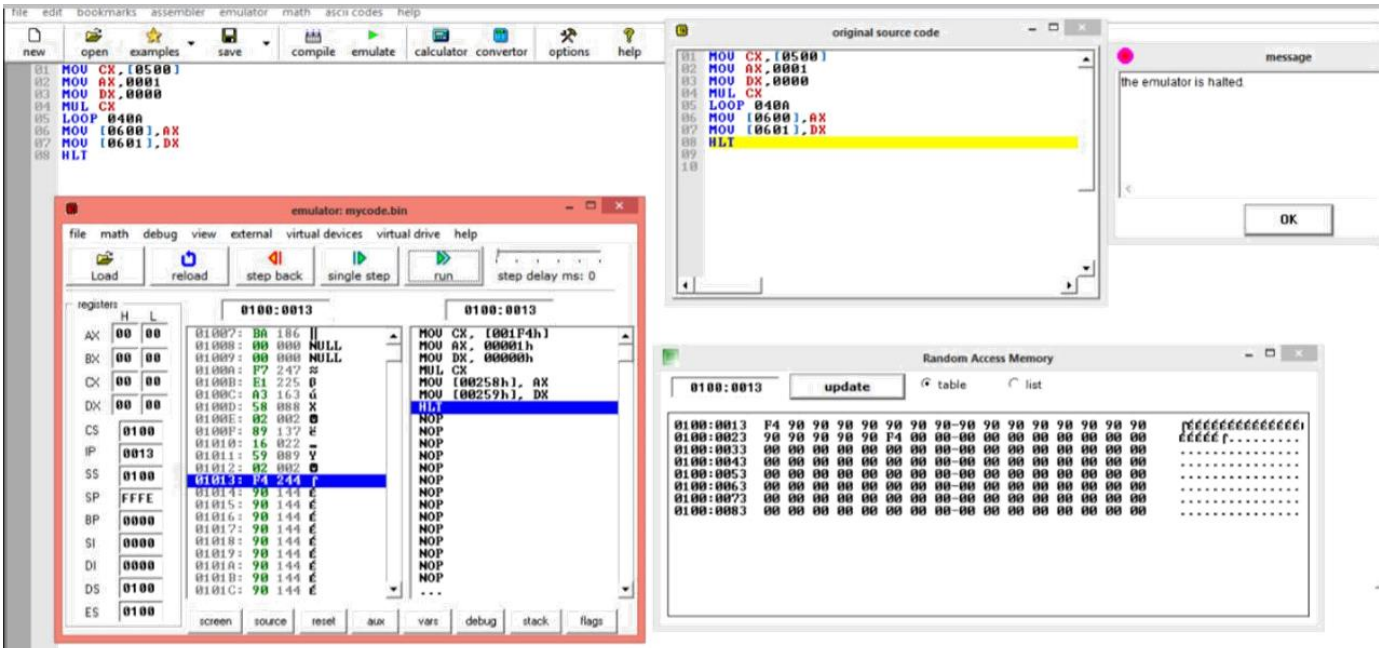
```
      H  L
AX    01 79
BX    02 62
CX    00 00
DX    02 62
CS    0100
IP    0038
SS    0100
SP    FFFE
BP    0000
SI    1010
DI    0000
DS    0100
ES    0100
```

```
0100:0038
0102A: 90 144 É
0102B: 90 144 É
0102C: 90 144 É
0102D: 90 144 É
0102E: 90 144 É
0102F: 90 144 É
01030: 90 144 É
01031: 90 144 É
01032: 90 144 É
01033: 90 144 É
01034: 90 144 É
01035: 90 144 É
01036: 90 144 É
01037: 90 144 É
01038: F4 244 ſ
01039: 00 000 NULL
0103A: 00 000 NULL
0103B: 00 000 NULL
0103C: 00 000 NULL
0103D: 00 000 NULL
0103E: 00 000 NULL
0103F: 00 000 NULL
01040: 00 000 NULL
01041: 00 000 NULL
01042: 00 000 NULL
01043: 00 000 NULL
01044: 00 000 NULL
01045: 00 000 NULL
01046: 00 000 NULL
01047: 00 000 NULL
01048: 00 000 NULL
01049: 00 000 NULL
0104A: 00 000 NULL
0104B: 00 000 NULL
0104C: 00 000 NULL
0104D: 00 000 NULL
0104E: 00 000 NULL
```

```
0100:0038
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
MOV SI, 1000H
MOV CL, 10H
MOV [SI], 00H
MOV AX, [SI]
INC SI
DEC CL
MOV [SI], 01H
MOV BX, [SI]
INC SI
DEC CL
L1: ADD AX, BX
MOV DX, AX
MOV [SI], AX
MOV AX, BX
MOV BX, DX
INC SI
DEC CL
JNZ L1
END
```

8. Write an ALP to find the factorial of a number.



file edit bookmarks assembler emulator math ascii codes help
new
open
examples
save
compile
emulate
calculator
convertor
options
help

```
MOV CX,[0500]
MOV AX,0001
MOV DX,0000
MUL CX
LOOP 040A
MOV [0600],AX
MOV [0601],DX
HLT
```

emulator: mycode.bin
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
      H  L
AX    00 00
BX    00 00
CX    00 00
DX    00 00
CS    0100
IP    0013
SS    0100
SP    FFFE
BP    0000
SI    0000
DI    0000
DS    0100
ES    0100
```

```
0100:0013
01007: BA 186 ‖
01008: 00 000 NULL
01009: 00 000 NULL
0100A: F7 247 ≈
0100B: E1 225 β
0100C: A3 163 ú
0100D: 58 088 X
0100E: 02 002 ☻
0100F: 89 137 ë
01010: 16 022 ▬
01011: 59 089 Y
01012: 02 002 ☻
01013: F4 244 ſ
01014: 90 144 É
01015: 90 144 É
01016: 90 144 É
01017: 90 144 É
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: 90 144 É
```

```
0100:0013
MOV CX, [001F4h]
MOV AX, 00001h
MOV DX, 00000h
MUL CX
MOV [00258h], AX
MOV [00259h], DX
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```
MOV CX,[0500]
MOV AX,0001
MOV DX,0000
MUL CX
LOOP 040A
MOV [0600],AX
MOV [0601],DX
HLT
```

the emulator is halted.

0100:0013 ● table ○ list

```
0100:0013  F4 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90   ſÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ
0100:0023  90 90 90 90 90 F4 00 00-00 00 00 00 00 00 00 00   ÉÉÉÉÉſ..........
0100:0033  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0043  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0053  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0063  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0073  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0083  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

(EMU8086) Date ....

Expt. No. 6 8086 Microprocessor Programming Page No. 25

1. Write an ALP for adding two 16-bit numbers 0122H, 0127H stored in registers B and C respectively.

→

```
MOV  BX, 0122H
MOV  CX, 0127H
MOV  AX, BX
ADD  AX, CX
END
```

2. Write an ALP for adding two 8 bit numbers 02H, 1BH stored in registers B and C

→

```
MOV BL, 02H
MOV CL, 1BH
MOV AL, BL
ADD  AL, CL
END
```

3. Write an ALP for subtracting two 8 bit numbers stored in registers.

→

```
MOV  BL, 08H
MOV  CL, 02H
MOV  AL, BL
SBB  AL, CL
END
```

4. Write an ALP for adding the 16-bit data in memory address 01 and 02. Then perform addition of these two numbers and

Teacher's Signature : ____________

store the result in the memory address 03.

→
```
MOV [1001 H] , 0122 H
MOV [1002H], 0123H
MOV AX, [1001 H]
ADD AX, [1002H]
MOV [1003H], AX
END
```

5. Write an ALP for performing multiplication and division of two 16-bit numbers.

Multiplication:-

→
```
MOV AX, 0122H
MOV BX, 0128H
MUL BX
END
```

Division:-

→
```
MOV AX, 0122H
MOV BX, 0128H
DIV BX
END
```

6. Write an ALP for transferring a block of ten 16-bit data from memory address starting at 1000 to memory location 3000.

→
```
Mov SI , [1000H]
Mov DI, [3000H]
Mov AL, 0AH
LOOP: Mov BL, [SI]
      Mov [DI], BL
      INC SI
      INC DI
      DEC AL
      JNZ LOOP
      END
```

7. Write an ALP to display 13 numbers of the fibonacci series in a memory location.

→
```
MOV  SI, 1000H
MOV  CL, 0010 H
MOV  [SI], 0000H
MOV  AX, [SI]
INC  SI
DEC  CL
MOV  [SI], 0001H
MOV  BX, [SI]
INC  SI
DEC  CL
LOOP :   ADD AX, BX
         MOV DX, AX
         MOV [SI], AX
         MOV AX, BX
         MOV BX, DX
         INC SI
         DEC CL
         JNZ LOOP
         END
```

8. Write an ALP to find the factorial of a number.

```
→   Address      Mnemonics
    0400         MOV CX, [0500]
    0404         MOV AX , 0001H
    0407         MOV DX , 0000H
    040A         MUL CX
    040C         LOOP 040A
    0410         MOV [0600], AX
    0414         MOV [0601], DX
    0418         END
```

23/3/19