DHAKA UNIVERSITY OF ENGINEERING & TECHNOLOGY, GAZIPUR-1707

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course No: CSE 3812

Course Title: Microprocessor and Interfacing Sessional

Lab 5

Report Name: Logic, Shift and Rotate Instructions & Multiplication and Division Instructions in EMU8086.

Date of Allocation: 10/11/2024 Date of Submission: 23/11/2024

Submitted To:

Dr. Md. Jakirul Islam
Associate Professor, Department of CSE
Dhaka University of Engineering & Technology, Gazipur

Sumaya Khatun
Lecturer, Department of CSE
Dhaka University of Engineering & Technology, Gazipur

Submitted By:

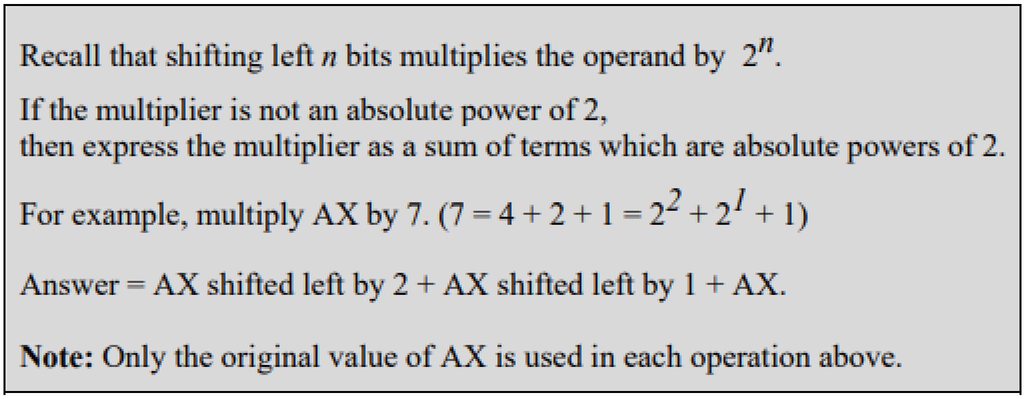
Name: Imran Raza
Student ID: 214097
Year/Semester: 3rd Year/1st Semester
Section: B

Objectives:

To understanding Logic, Shift and Rotate Instructions & Multiplication and Division Instructions in EMU8086.

Problem Statements:

1. Write a program to multiply AX by 27 using only Shift and Add instructions. You should not use the MUL instruction.



Recall that shifting left *n* bits multiplies the operand by 2^n.

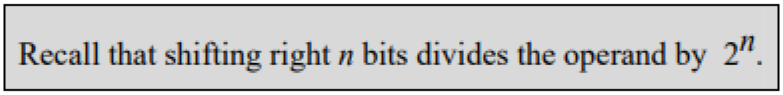
If the multiplier is not an absolute power of 2,
then express the multiplier as a sum of terms which are absolute powers of 2.

For example, multiply AX by 7. ($7 = 4 + 2 + 1 = 2^2 + 2^1 + 1$)

Answer = AX shifted left by 2 + AX shifted left by 1 + AX.

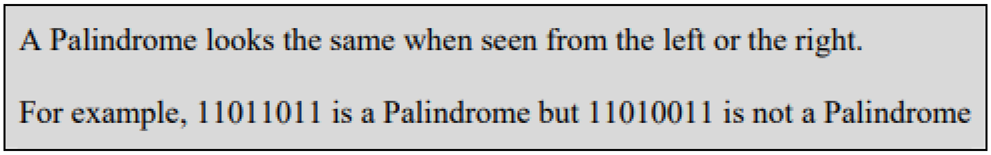
Note: Only the original value of AX is used in each operation above.

2. Write a program to divide AX by 8 using Shift instructions. You should not use the DIV instruction. Assume AX is a multiple of 8.



Recall that shifting right *n* bits divides the operand by 2^n.

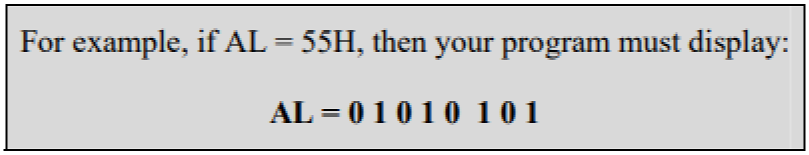
3. Write a program to check if a byte is a Palindrome. [Hint: Use Rotate instructions]. If the byte is a Palindrome, then move AAh into BL. Otherwise move 00h in BL.



A Palindrome looks the same when seen from the left or the right.

For example, 11011011 is a Palindrome but 11010011 is not a Palindrome

4. Write a program to display the bits of a register or memory location. Use the INT 21H interrupts to display data on the display monitor. [Hint: Use logical shift instruction to move data bit into the carry flag]



For example, if AL = 55H, then your program must display:

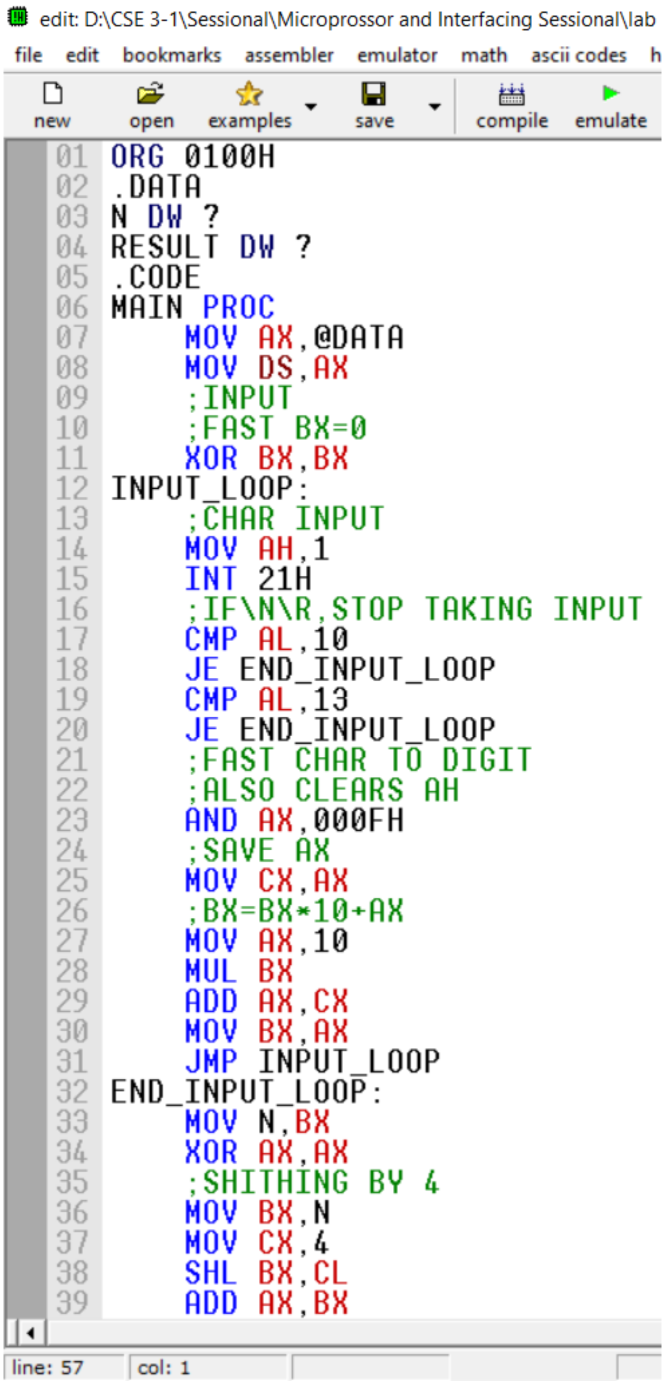
AL = 0 1 0 1 0 1 0 1

5. Write assembly code for each of the following high-level language assignment statements. Suppose that A, B, and C are word variables and all products will fit in 16 bits. Use IMUL for multiplication. It's not necessary to preserve the contents of variables A, B, and C.
   a. A = 5 x A – 7
   b. B = (A-B) x (B-10)

Problem 1:

```
ORG 0100H
.DATA
N DW ?
RESULT DW ?
.CODE
MAIN PROC
   MOV AX,@DATA
   MOV DS,AX
   ;INPUT
   ;FAST BX=0
   XOR BX,BX
INPUT_LOOP:
   ;CHAR INPUT
   MOV AH,1
   INT 21H
   ;IF\N\R,STOP TAKING INPUT
   CMP AL,10
   JE END_INPUT_LOOP
   CMP AL,13
   JE END_INPUT_LOOP
   ;FAST CHAR TO DIGIT
   ;ALSO CLEARS AH
   AND AX,000FH
   ;SAVE AX
   MOV CX,AX
   ;BX=BX*10+AX
   MOV AX,10
   MUL BX
   ADD AX,CX
   MOV BX,AX
   JMP INPUT_LOOP
```

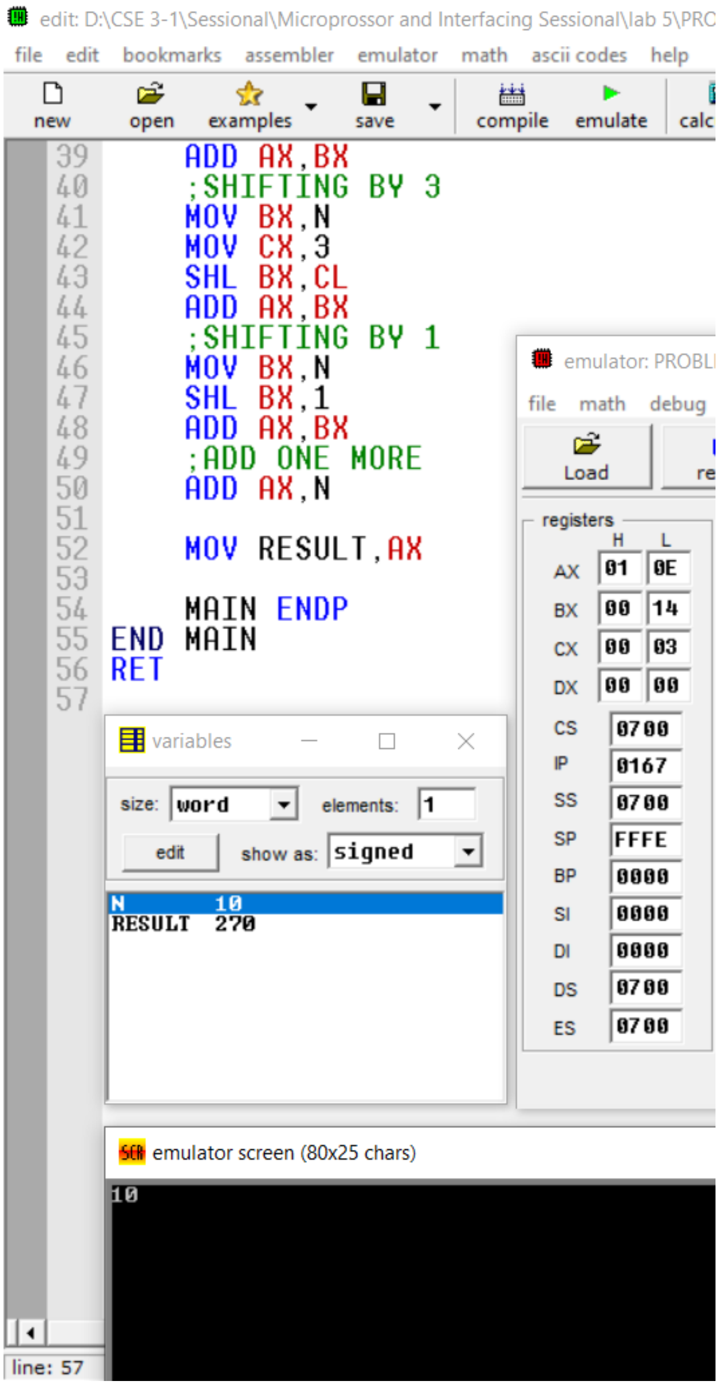
```
END_INPUT_LOOP:
    MOV N,BX
    XOR AX,AX
    ;SHITHING BY 4
    MOV BX,N
    MOV CX,4
    SHL BX,CL
    ADD AX,BX
    ;SHIFTING BY 3
    MOV BX,N
    MOV CX,3
    SHL BX,CL
    ADD AX,BX
    ;SHIFTING BY 1
    MOV BX,N
    SHL BX,1
    ADD AX,BX
    ;ADD ONE MORE
    ADD AX,N

    MOV RESULT,AX

    MAIN ENDP
END MAIN
RET
```



edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab
file edit bookmarks assembler emulator math ascii codes h
new open examples save compile emulate

```
ORG 0100H
.DATA
N DW ?
RESULT DW ?
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    ;INPUT
    ;FAST BX=0
    XOR BX,BX
INPUT_LOOP:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP
    CMP AL,13
    JE END_INPUT_LOOP
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP
END_INPUT_LOOP:
    MOV N,BX
    XOR AX,AX
    ;SHITHING BY 4
    MOV BX,N
    MOV CX,4
    SHL BX,CL
    ADD AX,BX
```

line: 57 col: 1



edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab 5\PRC
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calc

```
    ADD AX,BX
    ;SHIFTING BY 3
    MOV BX,N
    MOV CX,3
    SHL BX,CL
    ADD AX,BX
    ;SHIFTING BY 1
    MOV BX,N
    SHL BX,1
    ADD AX,BX
    ;ADD ONE MORE
    ADD AX,N

    MOV RESULT,AX

    MAIN ENDP
END MAIN
RET
```

line: 57

emulator: PROBL
file math debug
Load re

registers

	H	L
AX	01	0E
BX	00	14
CX	00	03
DX	00	00
CS	0700	
IP	0167	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

variables

size: word elements: 1
edit show as: signed

```
N       10
RESULT  270
```

emulator screen (80x25 chars)

```
10
```

Problem 2:

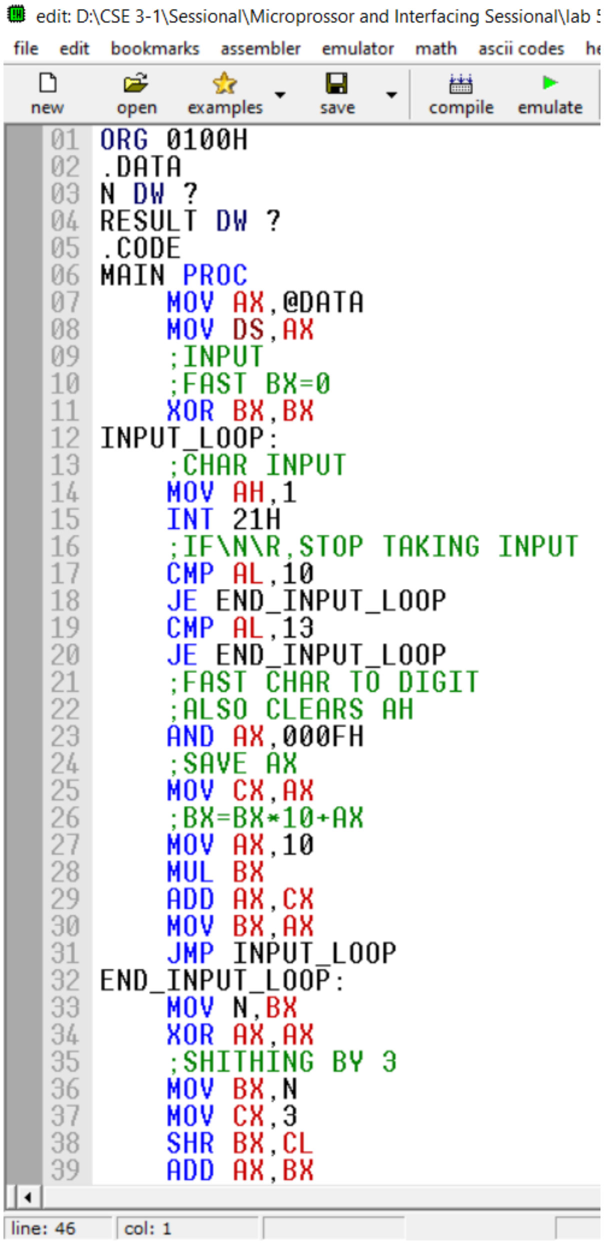
```
ORG 0100H
.DATA
N DW ?
RESULT DW ?
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    ;INPUT
    ;FAST BX=0
    XOR BX,BX
INPUT_LOOP:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP
    CMP AL,13
    JE END_INPUT_LOOP
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP
END_INPUT_LOOP:
    MOV N,BX
    XOR AX,AX
    ;SHITHING BY 3
    MOV BX,N
    MOV CX,3
    SHR BX,CL
    ADD AX,BX
```

MOV RESULT,AX

MAIN ENDP

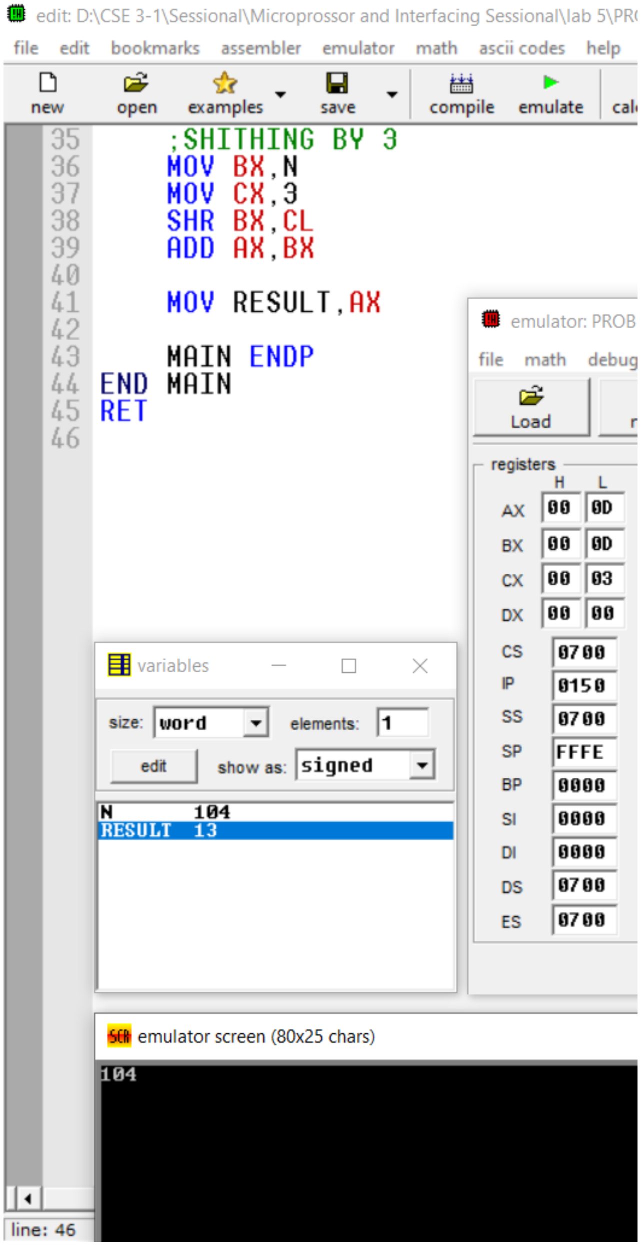
END MAIN

RET


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab 5
file edit bookmarks assembler emulator math ascii codes he
new open examples save compile emulate

```
ORG 0100H
.DATA
N DW ?
RESULT DW ?
.CODE
MAIN PROC
     MOV AX,@DATA
     MOV DS,AX
     ;INPUT
     ;FAST BX=0
     XOR BX,BX
INPUT_LOOP:
     ;CHAR INPUT
     MOV AH,1
     INT 21H
     ;IF\N\R,STOP TAKING INPUT
     CMP AL,10
     JE END_INPUT_LOOP
     CMP AL,13
     JE END_INPUT_LOOP
     ;FAST CHAR TO DIGIT
     ;ALSO CLEARS AH
     AND AX,000FH
     ;SAVE AX
     MOV CX,AX
     ;BX=BX*10+AX
     MOV AX,10
     MUL BX
     ADD AX,CX
     MOV BX,AX
     JMP INPUT_LOOP
END_INPUT_LOOP:
     MOV N,BX
     XOR AX,AX
     ;SHITHING BY 3
     MOV BX,N
     MOV CX,3
     SHR BX,CL
     ADD AX,BX
```

line: 46 col: 1


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab 5\PR
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate cal
emulator: PROB
file math debug
Load
variables
edit
emulator screen (80x25 chars)

```
     ;SHITHING BY 3
     MOV BX,N
     MOV CX,3
     SHR BX,CL
     ADD AX,BX

     MOV RESULT,AX

     MAIN ENDP
END MAIN
RET
```

registers

	H	L
AX	00	0D
BX	00	0D
CX	00	03
DX	00	00
CS	0700	
IP	0150	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

size: word elements: 1

show as: signed

```
N       104
RESULT  13
```

```
104
```

line: 46

Problem 3:

ORG 0100H

.DATA

N DB 11011011B

RESULT DB ?

.CODE

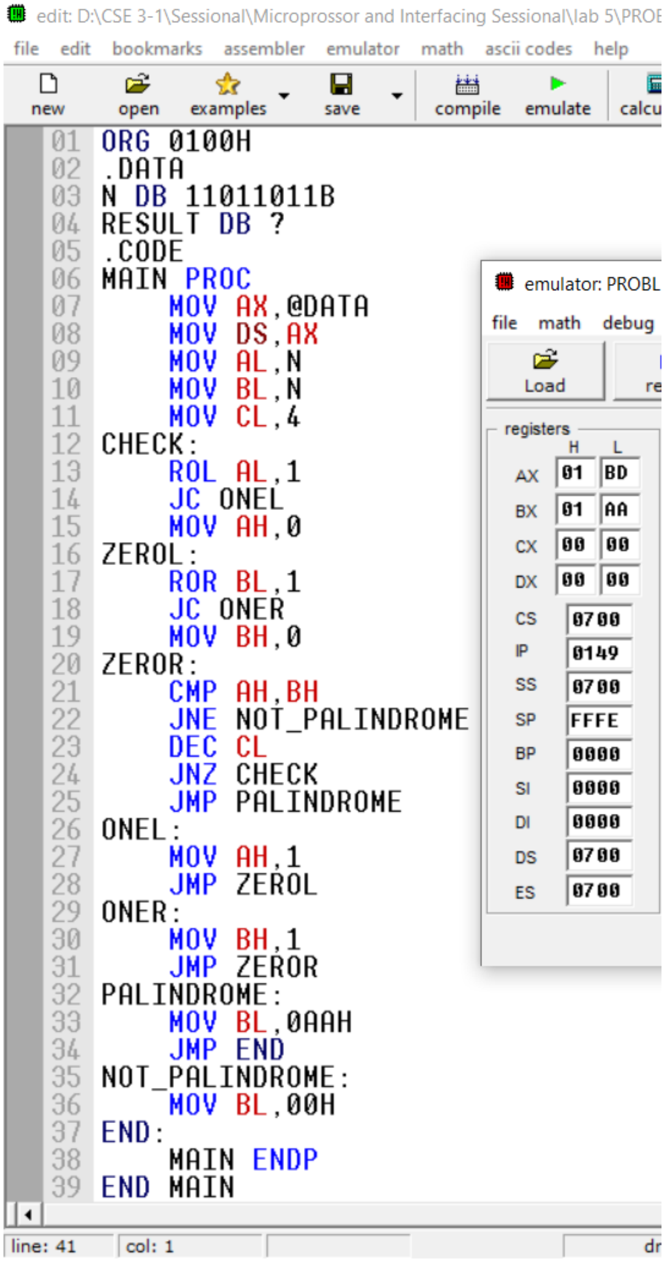
```
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX

    MOV AL,N
    MOV BL,N
    MOV CL,4
CHECK:
    ROL AL,1
    JC ONEL
    MOV AH,0
ZEROL:
    ROR BL,1
    JC ONER
    MOV BH,0
ZEROR:
    CMP AH,BH
    JNE NOT_PALINDROME
    DEC CL
    JNZ CHECK
    JMP PALINDROME
ONEL:
    MOV AH,1
    JMP ZEROL
ONER:
    MOV BH,1
    JMP ZEROR

PALINDROME:
    MOV BL,0AAH
    JMP END
NOT_PALINDROME:
    MOV BL,00H
```

END:

MAIN ENDP

END MAIN

RET


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab 5\PROE
file edit bookmarks assembler emulator math ascii codes help
new
open
examples
save
compile
emulate
calcu
emulator: PROBL
file math debug
Load
re
registers
H
L
dr

```
ORG 0100H
.DATA
N DB 11011011B
RESULT DB ?
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    MOV AL,N
    MOV BL,N
    MOV CL,4
CHECK:
    ROL AL,1
    JC ONEL
    MOV AH,0
ZEROL:
    ROR BL,1
    JC ONER
    MOV BH,0
ZEROR:
    CMP AH,BH
    JNE NOT_PALINDROME
    DEC CL
    JNZ CHECK
    JMP PALINDROME
ONEL:
    MOV AH,1
    JMP ZEROL
ONER:
    MOV BH,1
    JMP ZEROR
PALINDROME:
    MOV BL,0AAH
    JMP END
NOT_PALINDROME:
    MOV BL,00H
END:
    MAIN ENDP
END MAIN
```

```
AX  01  BD
BX  01  AA
CX  00  00
DX  00  00
CS  0700
IP  0149
SS  0700
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

line: 41 col: 1

Problem 4:

ORG 0100H

.DATA

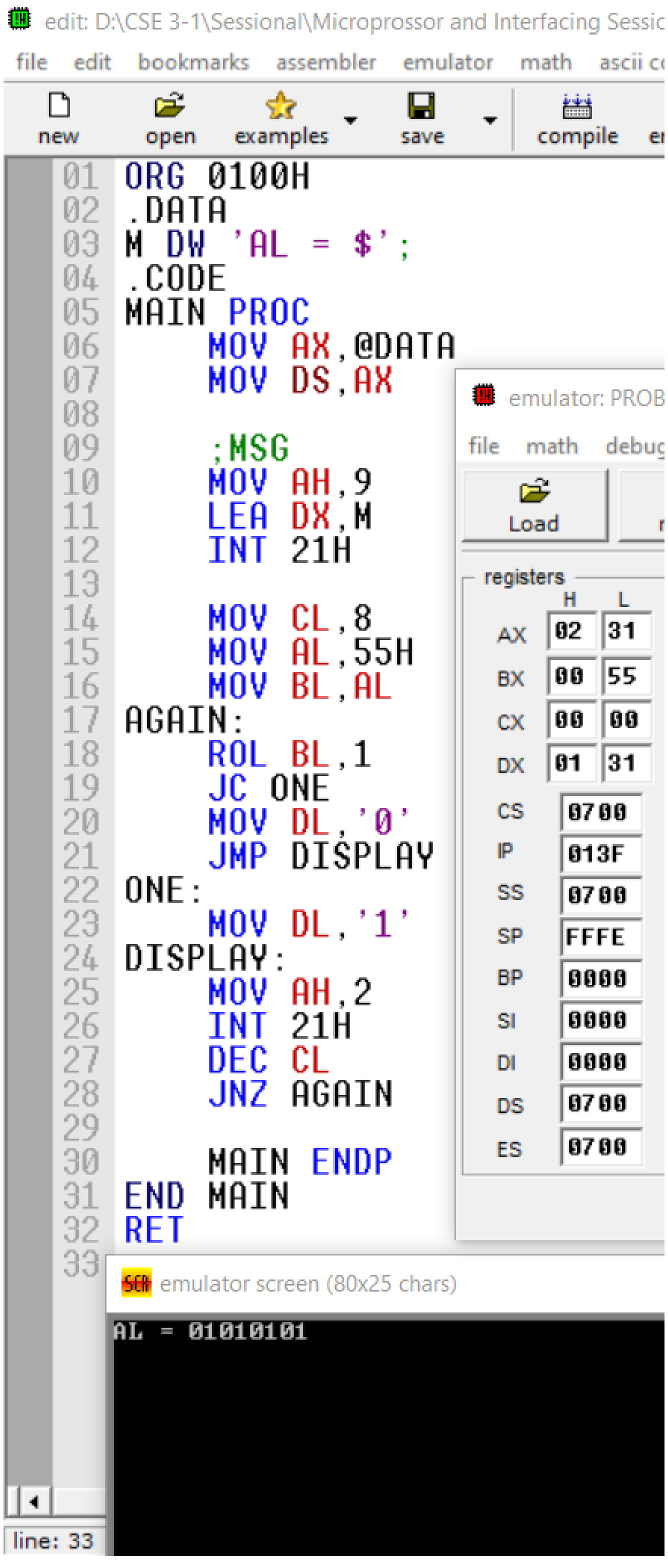
M DW 'AL = $'

.CODE

MAIN PROC

MOV AX,@DATA

```
    MOV DS,AX


    ;MSG
    MOV AH,9
    LEA DX,M
    INT 21H


    MOV CL,8
    MOV AL,55H
    MOV BL,AL
AGAIN:
    ROL BL,1
    JC ONE
    MOV DL,'0'
    JMP DISPLAY
ONE:
    MOV DL,'1'
DISPLAY:
    MOV AH,2
    INT 21H
    DEC CL
    JNZ AGAIN


    MAIN ENDP
END MAIN
RET
```


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessic
file edit bookmarks assembler emulator math ascii c
new
open
examples
save
compile
emulator: PROB
file math debug
Load
registers
H
L
emulator screen (80x25 chars)
line: 33

```
ORG 0100H
.DATA
M DW 'AL = $';
.CODE
MAIN PROC
     MOV AX,@DATA
     MOV DS,AX

     ;MSG
     MOV AH,9
     LEA DX,M
     INT 21H

     MOV CL,8
     MOV AL,55H
     MOV BL,AL
AGAIN:
     ROL BL,1
     JC ONE
     MOV DL,'0'
     JMP DISPLAY
ONE:
     MOV DL,'1'
DISPLAY:
     MOV AH,2
     INT 21H
     DEC CL
     JNZ AGAIN

     MAIN ENDP
END MAIN
RET
```

	H	L
AX	02	31
BX	00	55
CX	00	00
DX	01	31
CS	0700	
IP	013F	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
AL = 01010101
```

Problem 5:

ORG 0100H

.DATA

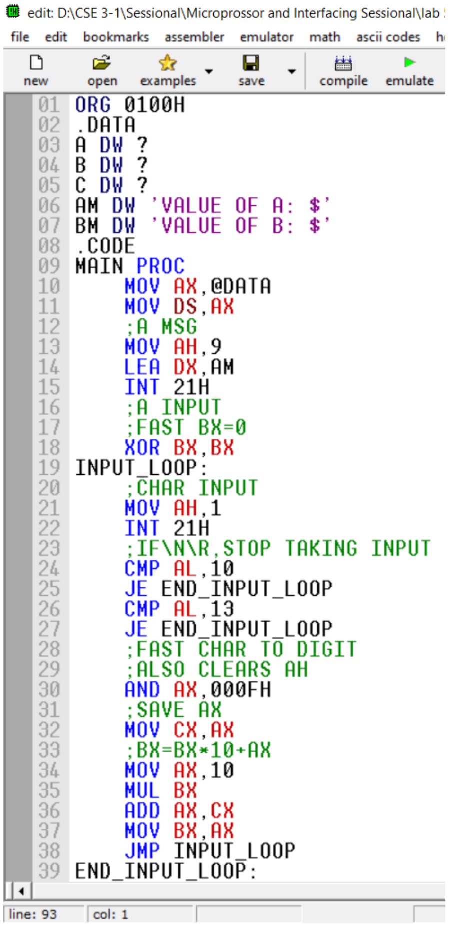
A DW ?

B DW ?

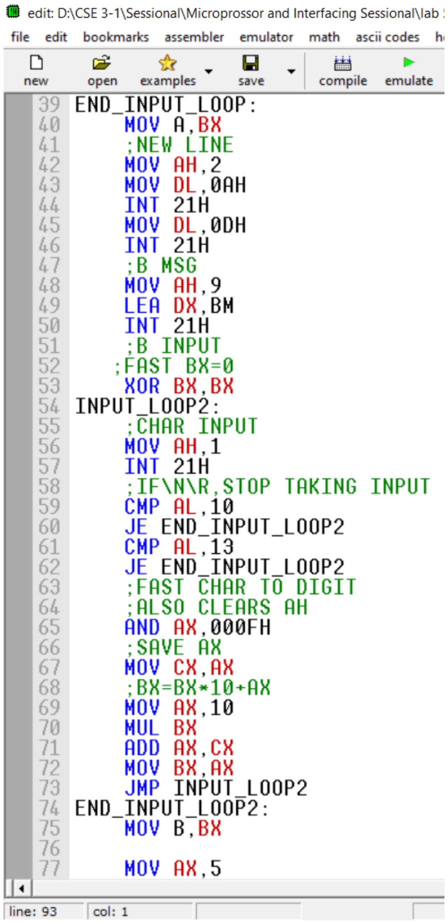
C DW ?

```
AM DW 'VALUE OF A: $'
BM DW 'VALUE OF B: $'
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    ;A MSG
    MOV AH,9
    LEA DX,AM
    INT 21H
    ;A INPUT
    ;FAST BX=0
    XOR BX,BX
INPUT_LOOP:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP
    CMP AL,13
    JE END_INPUT_LOOP
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP
```

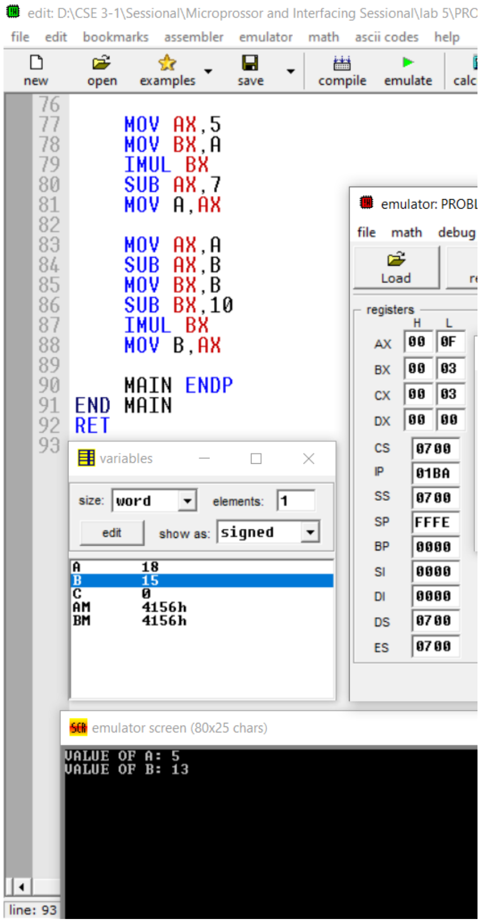
```
END_INPUT_LOOP:
    MOV A,BX
    ;NEW LINE
    MOV AH,2
    MOV DL,0AH
    INT 21H
    MOV DL,0DH
    INT 21H
    ;B MSG
    MOV AH,9
    LEA DX,BM
    INT 21H
    ;B INPUT
   ;FAST BX=0
    XOR BX,BX
INPUT_LOOP2:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP2
    CMP AL,13
    JE END_INPUT_LOOP2
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP2
END_INPUT_LOOP2:
    MOV B,BX

    MOV AX,5
    MOV BX,A
    IMUL BX
    SUB AX,7
    MOV A,AX

    MOV AX,A
    SUB AX,B
    MOV BX,B
    SUB BX,10
    IMUL BX
    MOV B,AX

    MAIN ENDP
END MAIN
RET
```


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab
file edit bookmarks assembler emulator math ascii codes h
new open examples save compile emulate
line: 93 col: 1

```
ORG 0100H
.DATA
A DW ?
B DW ?
C DW ?
AM DW 'VALUE OF A: $'
BM DW 'VALUE OF B: $'
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    ;A MSG
    MOV AH,9
    LEA DX,AM
    INT 21H
    ;A INPUT
    ;FAST BX=0
    XOR BX,BX
INPUT_LOOP:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP
    CMP AL,13
    JE END_INPUT_LOOP
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP
END_INPUT_LOOP:
```


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab
file edit bookmarks assembler emulator math ascii codes h
new open examples save compile emulate
line: 93 col: 1

```
END_INPUT_LOOP:
    MOV A,BX
    ;NEW LINE
    MOV AH,2
    MOV DL,0AH
    INT 21H
    MOV DL,0DH
    INT 21H
    ;B MSG
    MOV AH,9
    LEA DX,BM
    INT 21H
    ;B INPUT
   ;FAST BX=0
    XOR BX,BX
INPUT_LOOP2:
    ;CHAR INPUT
    MOV AH,1
    INT 21H
    ;IF\N\R,STOP TAKING INPUT
    CMP AL,10
    JE END_INPUT_LOOP2
    CMP AL,13
    JE END_INPUT_LOOP2
    ;FAST CHAR TO DIGIT
    ;ALSO CLEARS AH
    AND AX,000FH
    ;SAVE AX
    MOV CX,AX
    ;BX=BX*10+AX
    MOV AX,10
    MUL BX
    ADD AX,CX
    MOV BX,AX
    JMP INPUT_LOOP2
END_INPUT_LOOP2:
    MOV B,BX

    MOV AX,5
```


edit: D:\CSE 3-1\Sessional\Microprossor and Interfacing Sessional\lab 5\PRC
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calc
emulator: PROBL
file math debug
Load
registers
variables
emulator screen (80x25 chars)
line: 93

```

    MOV AX,5
    MOV BX,A
    IMUL BX
    SUB AX,7
    MOV A,AX

    MOV AX,A
    SUB AX,B
    MOV BX,B
    SUB BX,10
    IMUL BX
    MOV B,AX

    MAIN ENDP
END MAIN
RET
```

registers

	H	L
AX	00	0F
BX	00	03
CX	00	03
DX	00	00

CS	0700
IP	01BA
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

size: word elements: 1

edit show as: signed

```
A       18
B       15
C       0
AM      4156h
BM      4156h
```

```
VALUE OF A: 5
VALUE OF B: 13
```

Discussion:

Finally it can be said that, we learned about so many new instruction called Logic, Shift and Rotate Instructions & Multiplication and Division Instructions. By using those instruction, we are able solve new logical problem like Palindrome Check or Byte Display etc.