OC EXPERIMENT LAB 5

TITLE: Writing assembly language programs

NAME: Yash Gupta

ROLL NO: S20200010234

OBSERVATION: In this lab I learnt how to write 8086 assembly language programs of

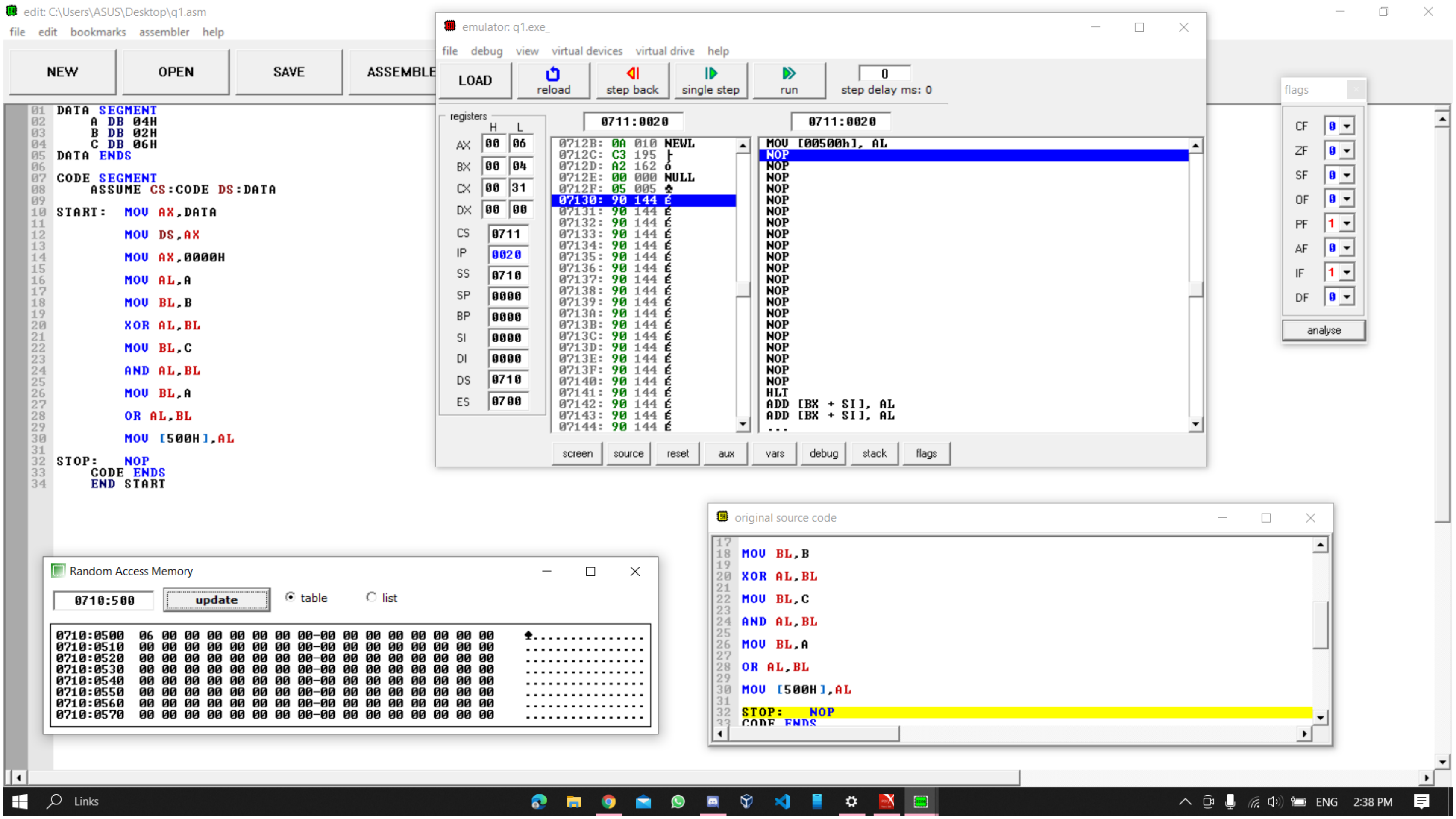
doing basic arithmetic and logical operations on two numbers using different addressing

modes.

QUESTION 1:- Write the 8086 assembly language program to implement the following expression.

f = (a □ b)c + a

Assume a, b, c are the 8/16 bit binary numbers



edit: C:\Users\ASUS\Desktop\q1.asm

```
DATA SEGMENT
     A DB 04H
     B DB 02H
     C DB 06H
DATA ENDS

CODE SEGMENT
     ASSUME CS:CODE DS:DATA

START:  MOV AX,DATA

        MOV DS,AX

        MOV AX,0000H

        MOV AL,A

        MOV BL,B

        XOR AL,BL

        MOV BL,C

        AND AL,BL

        MOV BL,A

        OR AL,BL

        MOV [500H],AL

STOP:   NOP
     CODE ENDS
     END START
```

emulator: q1.exe_

registers

	H	L
AX	00	06
BX	00	04
CX	00	31
DX	00	00
CS	0711	
IP	0020	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

```
0711:0020

0712B: 0A 010 NEWL
0712C: C3 195 ├
0712D: A2 162 ó
0712E: 00 000 NULL
0712F: 05 005 ♣
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
07136: 90 144 É
07137: 90 144 É
07138: 90 144 É
07139: 90 144 É
0713A: 90 144 É
0713B: 90 144 É
0713C: 90 144 É
0713D: 90 144 É
0713E: 90 144 É
0713F: 90 144 É
07140: 90 144 É
07141: 90 144 É
07142: 90 144 É
07143: 90 144 É
07144: 90 144 É
```

```
0711:0020

MOV [00500h], AL
NOP
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
...
```

flags

CF	0
ZF	0
SF	0
OF	0
PF	1
AF	0
IF	1
DF	0

original source code

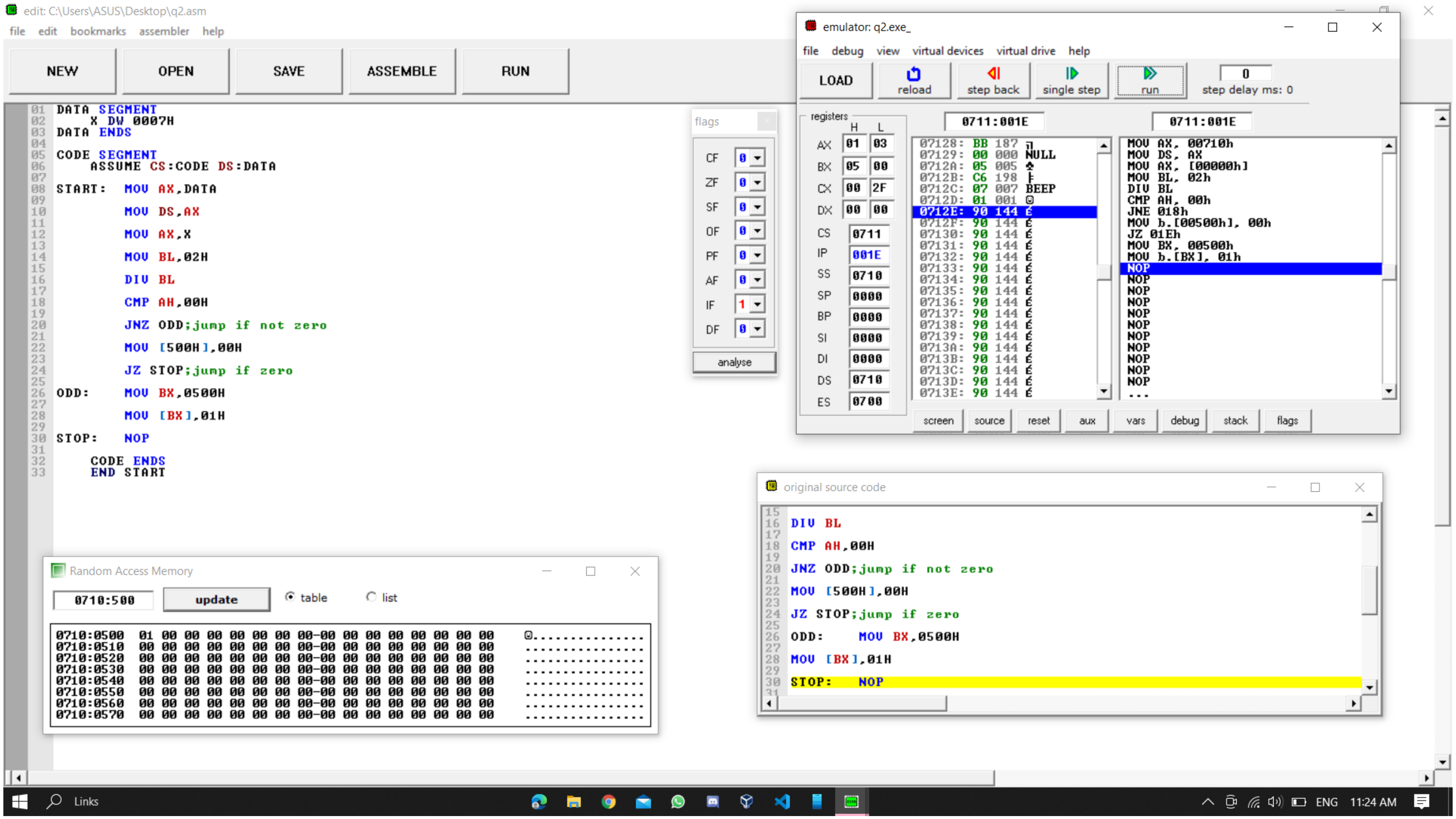
```
MOV BL,B

XOR AL,BL

MOV BL,C

AND AL,BL

MOV BL,A

OR AL,BL

MOV [500H],AL

STOP:   NOP
CODE ENDS
```

Random Access Memory

0710:500 ● table ○ list

```
0710:0500  06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ♠...............
0710:0510  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0520  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0530  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0540  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0550  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0560  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0570  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

QUESTION 2. **Write the 8086 assembly language programme to detect whether a given number is even or odd.**



```
DATA SEGMENT
     X DW 0007H
DATA ENDS

CODE SEGMENT
     ASSUME CS:CODE DS:DATA

START:  MOV AX,DATA

        MOV DS,AX

        MOV AX,X

        MOV BL,02H

        DIV BL

        CMP AH,00H

        JNZ ODD;jump if not zero

        MOV [500H],00H

        JZ STOP;jump if zero

ODD:    MOV BX,0500H

        MOV [BX],01H

STOP:   NOP

     CODE ENDS
     END START
```

CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0

step delay ms: 0

registers

	H	L
AX	01	03
BX	05	00
CX	00	2F
DX	00	00
CS	0711	
IP	001E	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

```
0711:001E
07128: BB 187 ╗
07129: 00 000 NULL
0712A: 05 005 ♣
0712B: C6 198 ╞
0712C: 07 007 BEEP
0712D: 01 001 ☺
0712E: 90 144 É
0712F: 90 144 É
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
07136: 90 144 É
07137: 90 144 É
07138: 90 144 É
07139: 90 144 É
0713A: 90 144 É
0713B: 90 144 É
0713C: 90 144 É
0713D: 90 144 É
0713E: 90 144 É
```

```
0711:001E
MOV AX, 00710h
MOV DS, AX
MOV AX, [00000h]
MOV BL, 02h
DIV BL
CMP AH, 00h
JNE 018h
MOV b.[00500h], 00h
JZ 01Eh
MOV BX, 00500h
MOV b.[BX], 01h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

0710:500 ● table ○ list

```
0710:0500  01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ☺...............
0710:0510  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0520  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0530  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0540  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0550  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0560  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0570  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

original source code

```
DIV BL

CMP AH,00H

JNZ ODD;jump if not zero

MOV [500H],00H

JZ STOP;jump if zero

ODD:    MOV BX,0500H

MOV [BX],01H

STOP:   NOP
```

THANK YOU