Department of Computer Science
Faculty of Computing
UNIVERSITI TEKNOLOGI MALAYSIA

<table>
<tr><td><b>SUBJECT NAME:</b></td><td><b>COMPUTER ORGANIZATION AND ARCHITECTURE</b></td></tr>
<tr><td><b>SUBJECT CODE:</b></td><td><b>SECR 1033</b></td></tr>
<tr><td><b>SEMESTER:</b></td><td><b>2 – 2023/2024</b></td></tr>
<tr><td><b>LAB TITLE:</b></td><td><b>Lab 2: Arithmetic Equations & Operations</b></td></tr>
<tr><td><b>STUDENT INFO :</b></td><td>Execute the lab in group of two.
<table>
<tr><th>Student 1</th><th>Student 2</th></tr>
<tr><td><i>No. 1, 3, 5</i></td><td><i>No 2, 4, 6</i></td></tr>
</table>
<b>Name 1:</b> <b>NUR FIRZANA BINTI BADRUS HISHAM</b>
<b>Metric No:</b> <b>A23CS0156</b>
<b>Name 2:</b> <b>LUBNA AL HAANI BINTI RADZUAN</b>
<b>Metric No:</b> <b>A23CS0107</b></td></tr>
<tr><td><b>SUBMISSION DATE:</b></td><td>22/5/2024</td></tr>
</table>

MARKS:

____

Arithmetic Equation Coding in Assembly Language

Q1. Execute the program below. Determine output of the program by inspecting the content of the related registers.
a) Fill in Table 1 with the content of each register or variable on every LINE, in **Hexadecimal** (as per the output). Please complete the comments for every LINE.
b) Paste the screenshot of all registers' content after each LINE is executed.

```
INCLUDE Irvine32.inc
.data
var1 word 1
var2 word 9

.code
main PROC
        mov ax, var1     ; LINE1
        mov bx, var2     ; LINE2
        xchg ax, bx      ; LINE3
        mov var1, ax     ; LINE4
        mov var2, bx     ; LINE5
        call DumpRegs
        exit
main ENDP
END main
```

Answer Q1

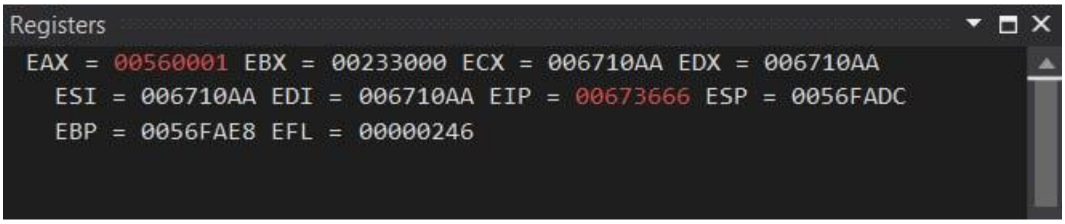
a) Fill (Write) in the contents for the related register in each line:

Table 1

LINE1	**AX = 0001h** **var1 = 0001h**	Move the value of var1 (1d) into register AX
LINE2	**BX =** 0009h **var2 =** 0009h	Move the value of var2 into register EAX
LINE3	**AX =** 0009h **BX =** 0001h	Exchange the value ax to bx and vice versa
LINE4	**AX =** 0009h **var1 =** 0009h	Move the value of register ax into var1
LINE5	**BX =** 0001h **var2 =** 0001h	Move the value of register bx into var2

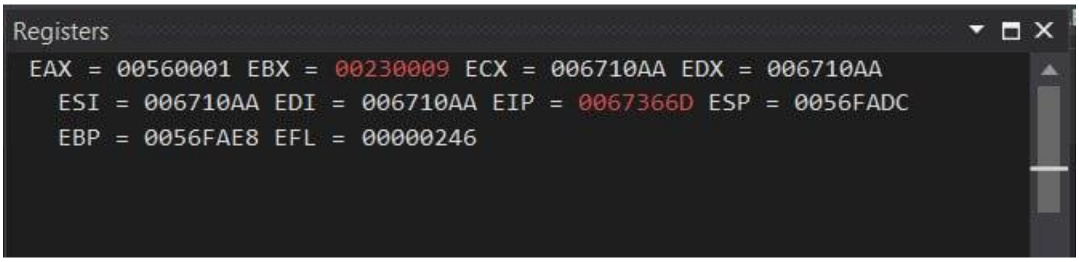
b) Paste here screenshot of all registers' content after each LINE is executed:

LINE1:



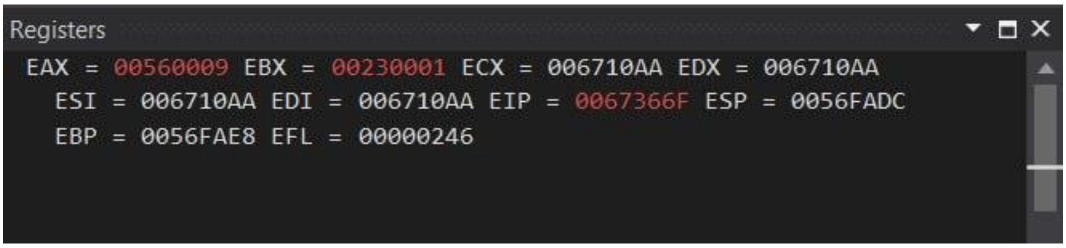
```
 EAX = 00560001 EBX = 00233000 ECX = 006710AA EDX = 006710AA
   ESI = 006710AA EDI = 006710AA EIP = 00673666 ESP = 0056FADC
   EBP = 0056FAE8 EFL = 00000246
```

LINE2:



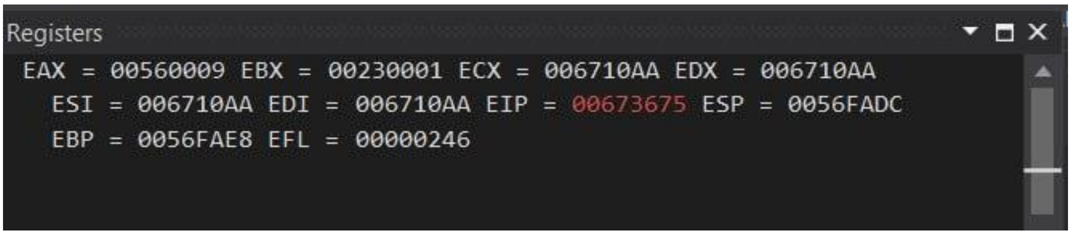
```
 EAX = 00560001 EBX = 00230009 ECX = 006710AA EDX = 006710AA
   ESI = 006710AA EDI = 006710AA EIP = 0067366D ESP = 0056FADC
   EBP = 0056FAE8 EFL = 00000246
```

LINE3:



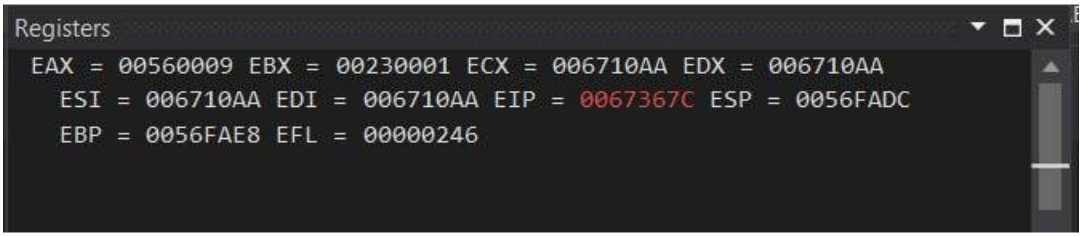
```
EAX = 00560009 EBX = 00230001 ECX = 006710AA EDX = 006710AA
  ESI = 006710AA EDI = 006710AA EIP = 0067366F ESP = 0056FADC
  EBP = 0056FAE8 EFL = 00000246
```

LINE4:



```
EAX = 00560009 EBX = 00230001 ECX = 006710AA EDX = 006710AA
  ESI = 006710AA EDI = 006710AA EIP = 00673675 ESP = 0056FADC
  EBP = 0056FAE8 EFL = 00000246
```

LINE5:



```
EAX = 00560009 EBX = 00230001 ECX = 006710AA EDX = 006710AA
  ESI = 006710AA EDI = 006710AA EIP = 0067367C ESP = 0056FADC
  EBP = 0056FAE8 EFL = 00000246
```

Q2. Execute the program below. Determine output of the program by inspecting the content of the related registers and watches.

a) Fill in Table 2 with the content of each register or variable on every LINE, in **Hexadecimal** (as per the output). Please complete the comments for every LINE.

b) Paste the screenshot of all registers' content after each LINE is executed.

Arithmetic expression: Rval = (-Xval + (Yval – Zval)) + 1

```
include irvine32.inc

.data
Rval DWORD ?
Xval DWORD 26
Yval DWORD 30
Zval DWORD 40

.code
main proc
       mov eax,Xval        ; LINE1
       neg eax             ; LINE2
       mov ebx,Yval        ; LINE3
       sub ebx,Zval        ; LINE4
       add eax,ebx         ; LINE5
       inc eax             ; LINE6
       mov Rval,eax        ; LINE7
       exit
main endp
end main
```

Answer Q2

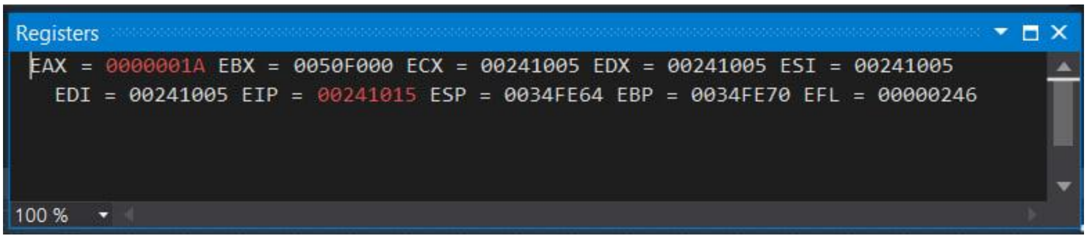
a) Fill (Write) in the contents for the related register in each line:

Table 2

LINE1	**EAX = 0000001Ah** **Xval = 0000001Ah**	Move the value of Xval (26d) into register EAX
LINE2	**EAX =** FFFFFFE6h	Change the sign of the register eax value
LINE3	**EBX =** 0000001Eh **Yval =** 0000001Eh	Move the value of Yval into ebx register
LINE4	**EBX =** FFFFFFF6h **Zval =** 00000028h	Substract the value of Zval from ebx
LINE5	**EAX =** FFFFFFDCh **EBX =** FFFFFFF6h	Add the value of ebx into eax
LINE6	**EAX =** FFFFFFDDh	Increment of register eax value
LINE7	**EAX =** FFFFFFDDh **Rval =** FFFFFFDDh	Move the value of register eax into Rval

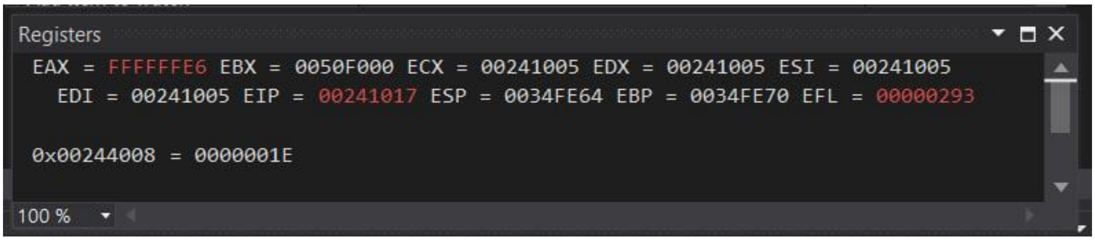
b) Paste here screenshot of all registers' content after each LINE is executed:

LINE1:



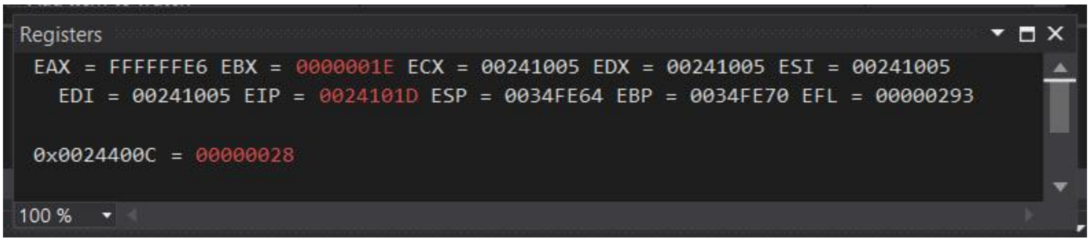
```
EAX = 0000001A EBX = 0050F000 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 00241015 ESP = 0034FE64 EBP = 0034FE70 EFL = 00000246
```

LINE2:



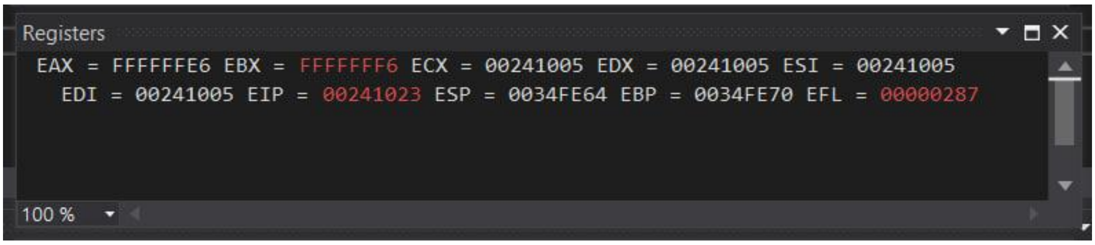
```
EAX = FFFFFFE6 EBX = 0050F000 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 00241017 ESP = 0034FE64 EBP = 0034FE70 EFL = 00000293

0x00244008 = 0000001E
```

LINE3:



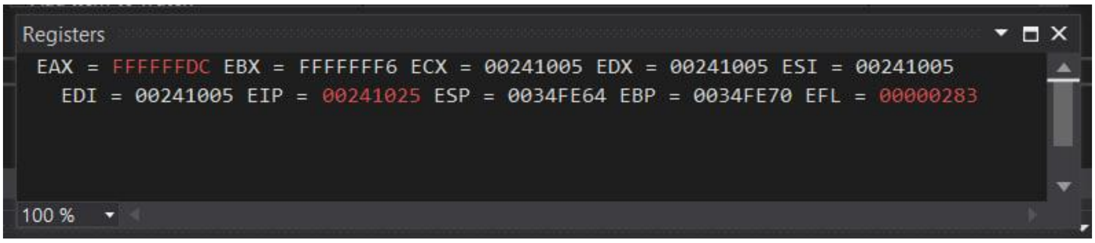
```
EAX = FFFFFFE6 EBX = 0000001E ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 0024101D ESP = 0034FE64 EBP = 0034FE70 EFL = 00000293

0x0024400C = 00000028
```

LINE4:



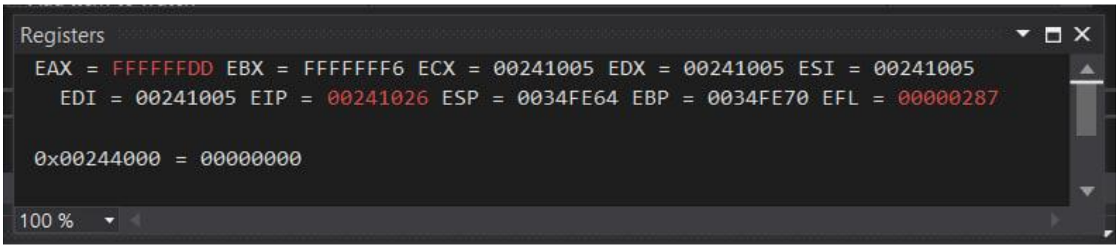
```
EAX = FFFFFFE6 EBX = FFFFFFF6 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 00241023 ESP = 0034FE64 EBP = 0034FE70 EFL = 00000287
```

LINE5:



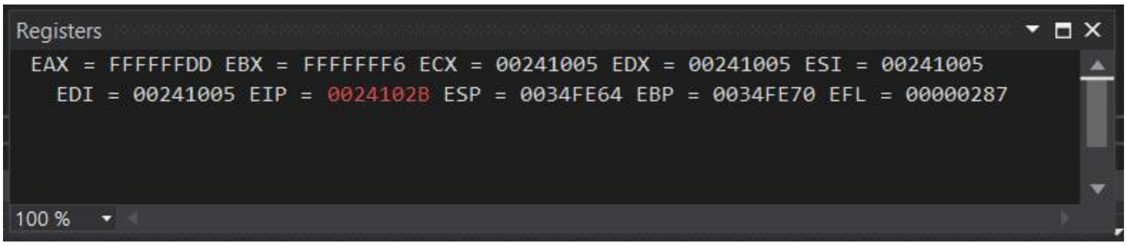
```
EAX = FFFFFFDC EBX = FFFFFFF6 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 00241025 ESP = 0034FE64 EBP = 0034FE70 EFL = 00000283
```

LINE6:



```
EAX = FFFFFFDD EBX = FFFFFFF6 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 00241026 ESP = 0034FE64 EBP = 0034FE70 EFL = 00000287

0x00244000 = 00000000
```

LINE7:



```
EAX = FFFFFFDD EBX = FFFFFFF6 ECX = 00241005 EDX = 00241005 ESI = 00241005
  EDI = 00241005 EIP = 0024102B ESP = 0034FE64 EBP = 0034FE70 EFL = 00000287
```

Q3. Execute the program below. Determine output of the program by inspecting the content of the related registers.

a) Fill in Table 3 with the content of each register or variable on every LINE, in **Hexadecimal** (as per the output). Please complete the comments for every LINE.

b) Paste the screenshot of all registers' content after each LINE is executed.

Arithmetic expression: var4 = [(var1 * var2) + var3] - 1

```
include irvine32.inc

.data
var1 DWORD 5
var2 DWORD 10
var3 DWORD 20
var4 DWORD ?

.code
main proc
      mov eax, var1          ; LINE1
      mul var2               ; LINE2
      add eax, var3          ; LINE3
      dec eax                ; LINE4
      exit
main endp
end main
```

Answer Q3

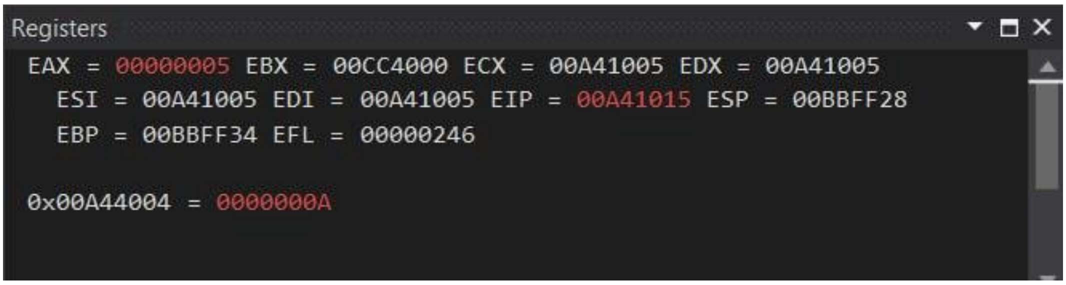
a) Fill (Write) in the contents for the related register in each line:

Table 3

LINE1	**EAX = 00000005h** **var1 = 00000005h**	Move the value of var1 (5d) into register EAX
LINE2	**EAX =** 00000032h **var2 =** 0000000Ah	multiply value of var2 with value in EAX and store in register
LINE3	**EAX =** 00000046h **var3 =** 00000014h	add value of var3 into register EAX
LINE4	**EAX =** 00000045h **var4 =** 00000045h	decrement value of register EAX and store in var4

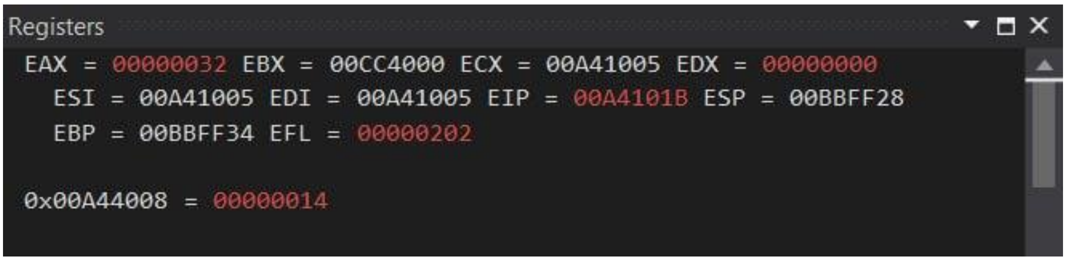
b) Paste here screenshot of all registers' content after each LINE is executed:

LINE1:



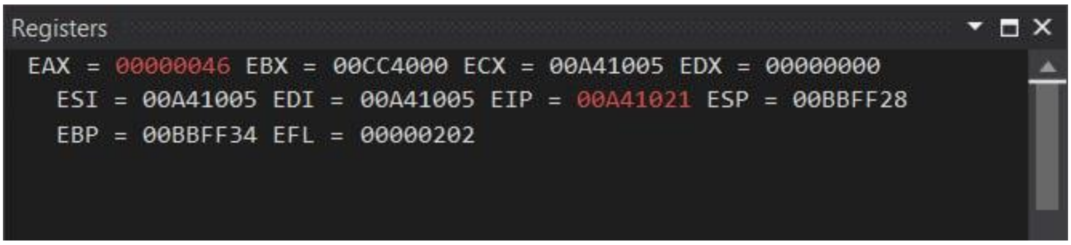
```
Registers
EAX = 00000005 EBX = 00CC4000 ECX = 00A41005 EDX = 00A41005
  ESI = 00A41005 EDI = 00A41005 EIP = 00A41015 ESP = 00BBFF28
  EBP = 00BBFF34 EFL = 00000246

0x00A44004 = 0000000A
```

LINE2:



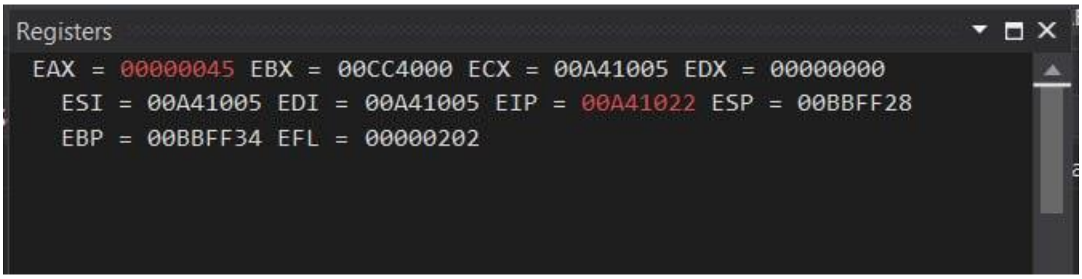
```
Registers
EAX = 00000032 EBX = 00CC4000 ECX = 00A41005 EDX = 00000000
  ESI = 00A41005 EDI = 00A41005 EIP = 00A4101B ESP = 00BBFF28
  EBP = 00BBFF34 EFL = 00000202

0x00A44008 = 00000014
```

LINE3:



```
Registers
EAX = 00000046 EBX = 00CC4000 ECX = 00A41005 EDX = 00000000
  ESI = 00A41005 EDI = 00A41005 EIP = 00A41021 ESP = 00BBFF28
  EBP = 00BBFF34 EFL = 00000202
```

LINE4:



```
Registers
EAX = 00000045 EBX = 00CC4000 ECX = 00A41005 EDX = 00000000
  ESI = 00A41005 EDI = 00A41005 EIP = 00A41022 ESP = 00BBFF28
  EBP = 00BBFF34 EFL = 00000202
```

Q4. Execute the program below. Determine output of the program by inspecting the content of the related registers.
a) Fill in Table 4 with the content of each register or variable on every LINE, in Hexadecimal (as per the output). Please complete the comments for every LINE.
b) Paste the screenshot of all registers' content after each LINE is executed.

Arithmetic expression: var4 = (var1 * 5) / (var2 – 3)

```
include irvine32.inc
.data
      var1 WORD 40
      var2 WORD 10
      var4 WORD ?
.code
main proc
      mov ax,var1       ; LINE1
      mov bx,5          ; LINE2
      mul bx            ; LINE3
      mov bx,var2       ; LINE4
      sub bx,3          ; LINE5
      div bx            ; LINE6
      mov var4,ax       ; LINE7
      exit
main endp
end main
```

Answer Q4

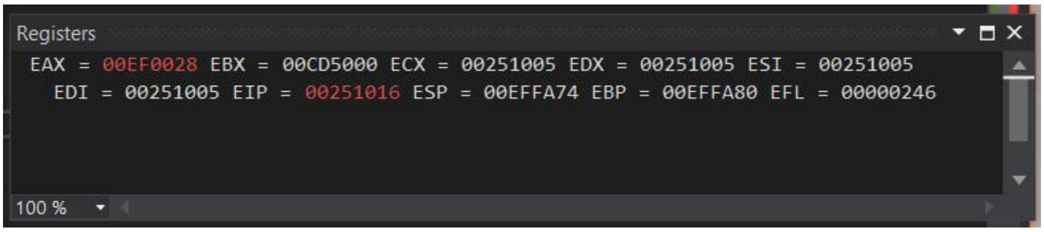
a) Fill (Write) in the contents for the related register in each line:

Table 4

LINE1	**AX = 0028h** **var1 = 0028h**	Move the value of var1 (40d) into register AX
LINE2	**BX =** 0005h	Move the value of 5 into register BX
LINE3	**AX =** 00C8h **BX =** 0005h	Multiply the value of register AX with value of register BX
LINE4	**BX =** 000Ah **var2 =** 000Ah	Move the value of var2 into register BX
LINE5	**BX =** 0007h	Subtract the value of register BX with 3
LINE6	**AX =** 001Ch **BX =** 0007h **DX =** 0004h	Divide the value of register AX with the value of register BX and remainder go into register DX
LINE7	**AX =** 001Ch **var4 =** 001Ch	Move the value of register AX into var4

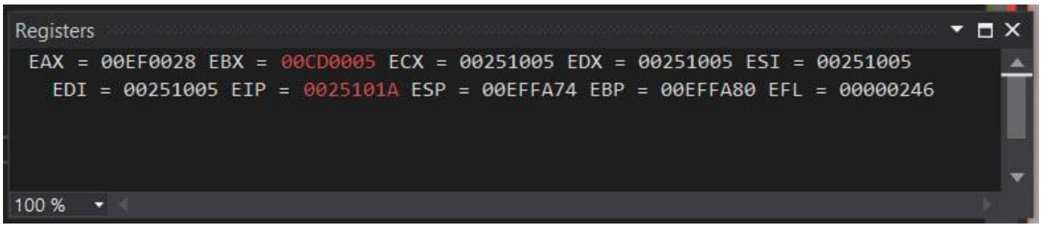
b) Paste here screenshot of all registers' content after each LINE is executed:

LINE1:



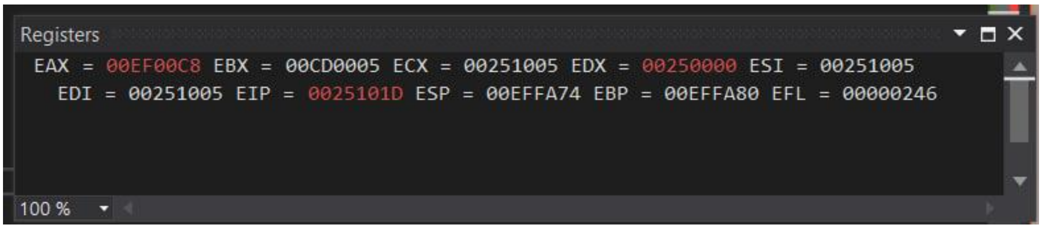
```
Registers
EAX = 00EF0028 EBX = 00CD5000 ECX = 00251005 EDX = 00251005 ESI = 00251005
  EDI = 00251005 EIP = 00251016 ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000246
```

LINE2:



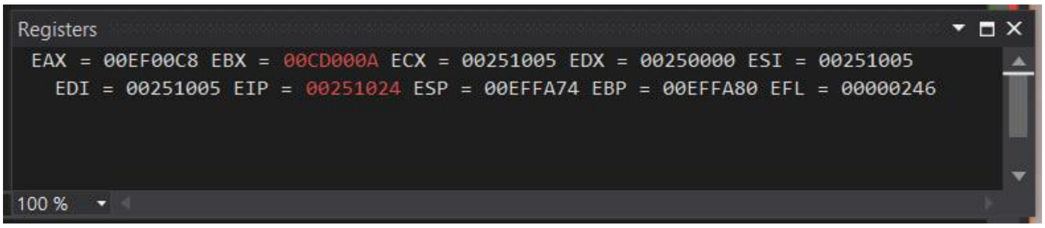
```
Registers
EAX = 00EF0028 EBX = 00CD0005 ECX = 00251005 EDX = 00251005 ESI = 00251005
  EDI = 00251005 EIP = 0025101A ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000246
```

LINE3:

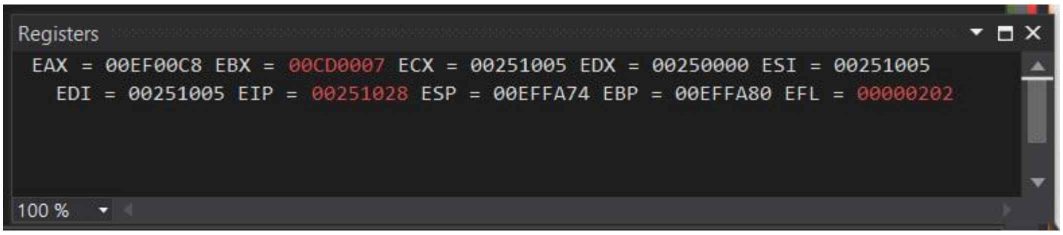


```
Registers
EAX = 00EF00C8 EBX = 00CD0005 ECX = 00251005 EDX = 00250000 ESI = 00251005
  EDI = 00251005 EIP = 0025101D ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000246
```

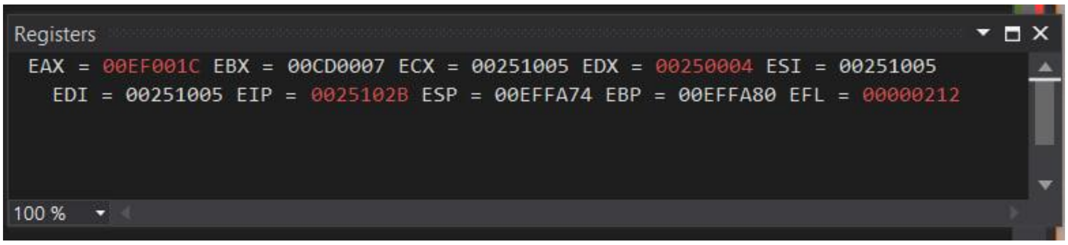
LINE4:



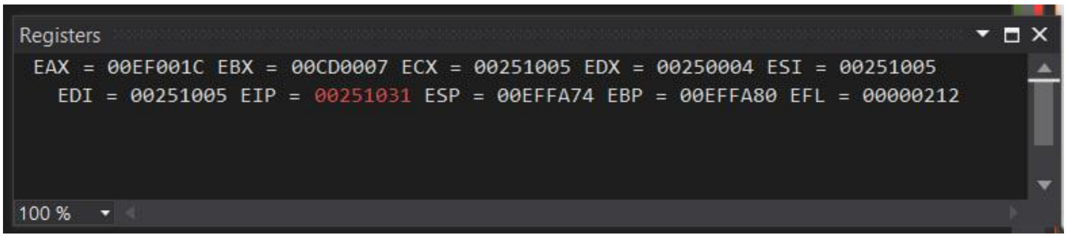
```
Registers
EAX = 00EF00C8 EBX = 00CD000A ECX = 00251005 EDX = 00250000 ESI = 00251005
  EDI = 00251005 EIP = 00251024 ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000246
```

LINE5: 

```
EAX = 00EF00C8 EBX = 00CD0007 ECX = 00251005 EDX = 00250000 ESI = 00251005
  EDI = 00251005 EIP = 00251028 ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000202
```

LINE6: 

```
EAX = 00EF001C EBX = 00CD0007 ECX = 00251005 EDX = 00250004 ESI = 00251005
  EDI = 00251005 EIP = 0025102B ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000212
```

LINE7: 

```
EAX = 00EF001C EBX = 00CD0007 ECX = 00251005 EDX = 00250004 ESI = 00251005
  EDI = 00251005 EIP = 00251031 ESP = 00EFFA74 EBP = 00EFFA80 EFL = 00000212
```

Short Notes for MUL CX and DIV BL:

MUL CX

a. MUL always uses AX (or its extended versions EAX or RAX) as the implicit destination register.

b. The operand size determines the size of the result:

   i. Byte-sized operand: Result in AX

   ii. Word-sized operand: Result in DX:AX

   iii. Doubleword-sized operand (32-bit mode): Result in EDX:EAX

   iv. Quadword-sized operand (64-bit mode): Result in RDX:RAX

c. The upper half of the result (DX or EDX or RDX) holds any overflow bits.

d. The Carry Flag (CF) is set if the upper half of the product is non-zero.

DIV BL

a. DIV always uses the DX:AX or EDX:EAX pair as the implicit dividend register.

b. The divisor is specified as the operand of the DIV instruction.

c. The quotient is stored in AX (for 16-bit division) or EAX (for 32-bit division).

d. The remainder is stored in DX.

e. Clear DX (or EDX for 32-bit division) before division to ensure a correct 16-bit or 32-bit dividend.

f. If the divisor is 0, a division error occurs.

g. The Overflow Flag (OF) is set if the quotient is too large to fit in the destination register.

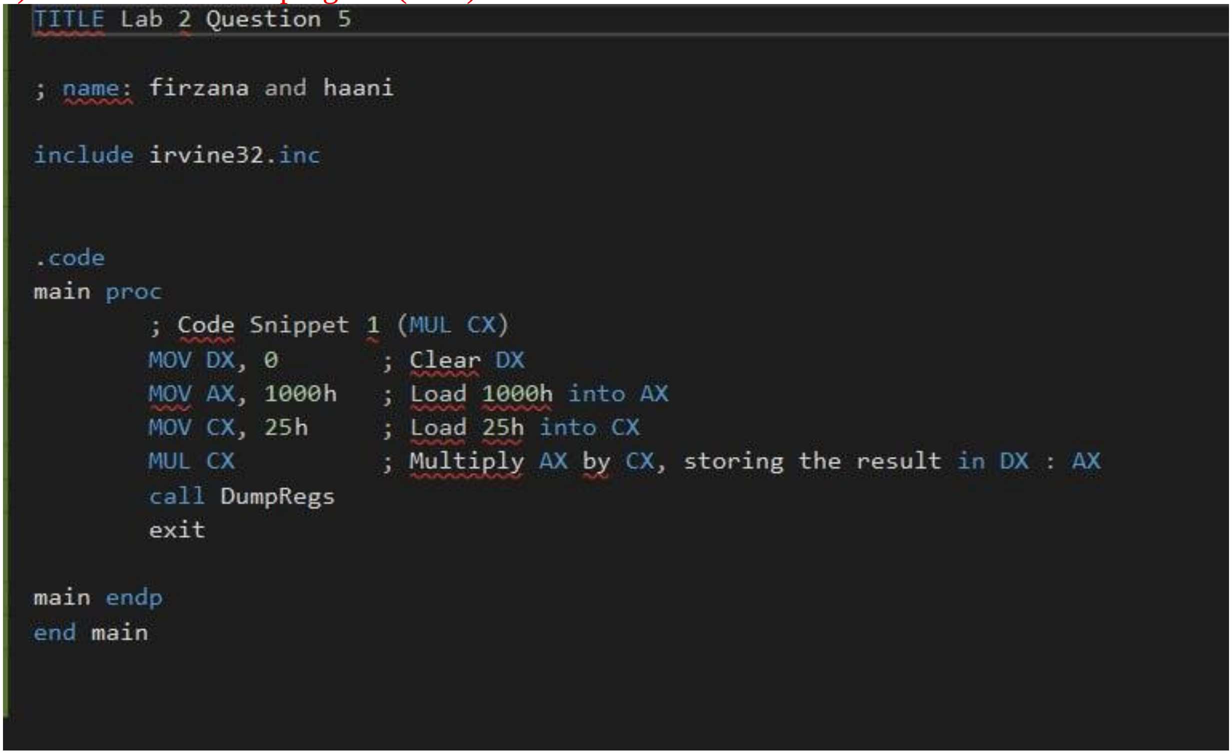
Q5. Given the following instructions as is Code Snippet 1.

a) Write a full program to execute the Code Snippet 1.
b) What are the contents of the related registers after Code Snippet 1 is executed? Paste the screenshot of DumpReg.

```
; Code Snippet 1 (MUL CX)

MOV DX,0         ; Clear DX
MOV AX,1000h     ; Load 1000h into AX
MOV CX, 25h      ; Load 25h into CX
MUL CX           ; Multiply AX by CX, storing the result in DX:AX
```

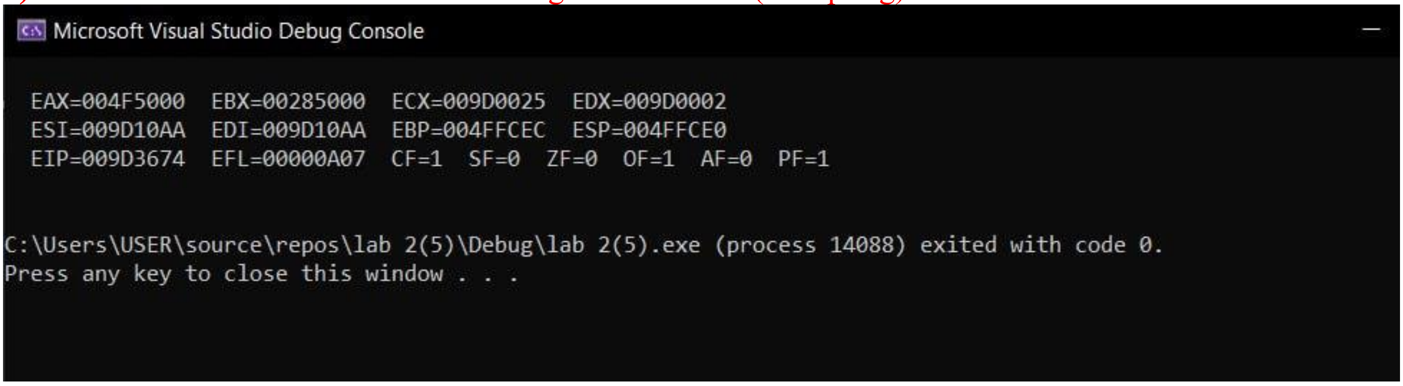
Answer Q5

a) Screenshot of full program (.asm) :



```
TITLE Lab 2 Question 5

; name: firzana and haani

include irvine32.inc


.code
main proc
        ; Code Snippet 1 (MUL CX)
        MOV DX, 0        ; Clear DX
        MOV AX, 1000h    ; Load 1000h into AX
        MOV CX, 25h      ; Load 25h into CX
        MUL CX           ; Multiply AX by CX, storing the result in DX : AX
        call DumpRegs
        exit

main endp
end main
```

b) Paste here the screenshot of the final registers' content (DumpReg):



```
  EAX=004F5000  EBX=00285000  ECX=009D0025  EDX=009D0002
  ESI=009D10AA  EDI=009D10AA  EBP=004FFCEC  ESP=004FFCE0
  EIP=009D3674  EFL=00000A07  CF=1  SF=0  ZF=0  OF=1  AF=0  PF=1


C:\Users\USER\source\repos\lab 2(5)\Debug\lab 2(5).exe (process 14088) exited with code 0.
Press any key to close this window . . .
```

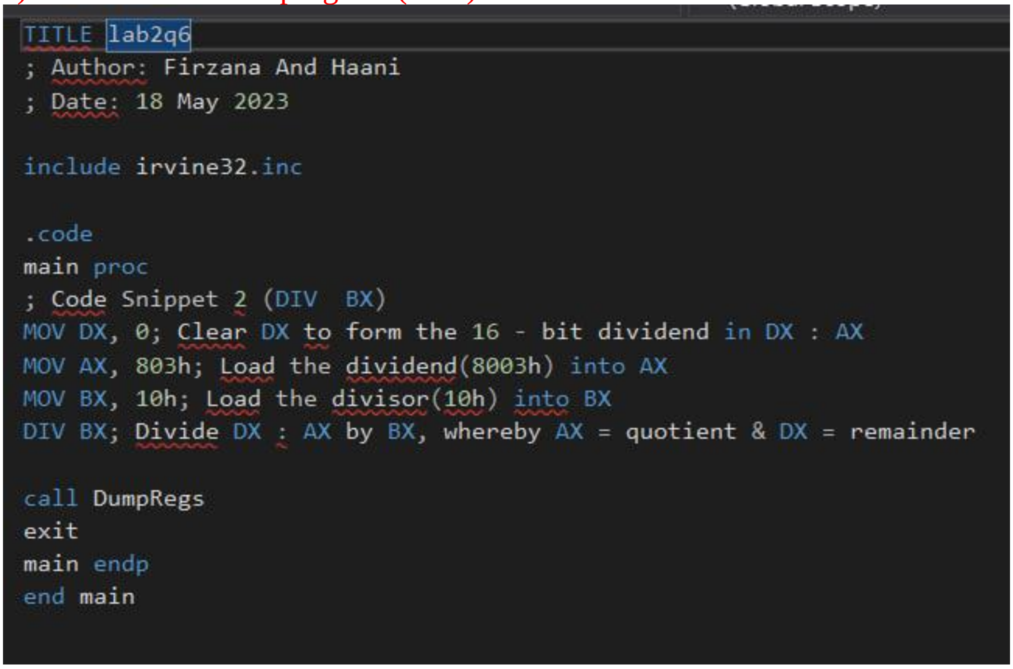
Q6. Given the following instructions as is Code Snippet 2.

a) Write a full program to execute the Code Snippet 2.
b) What are the contents of the related registers after Code Snippet 2 is executed? Paste the screenshot of DumpReg.

```
; Code Snippet 2 (DIV  BL)
MOV DX, 0      ; Clear DX to form the 16-bit dividend in DX:AX
MOV AX, 803h   ; Load the dividend (8003h) into AX
MOV BL, 10h    ; Load the divisor (10h) into BL
DIV BL         ; Divide DX:AX by BL, whereby AX=quotient & DX=remainder
```

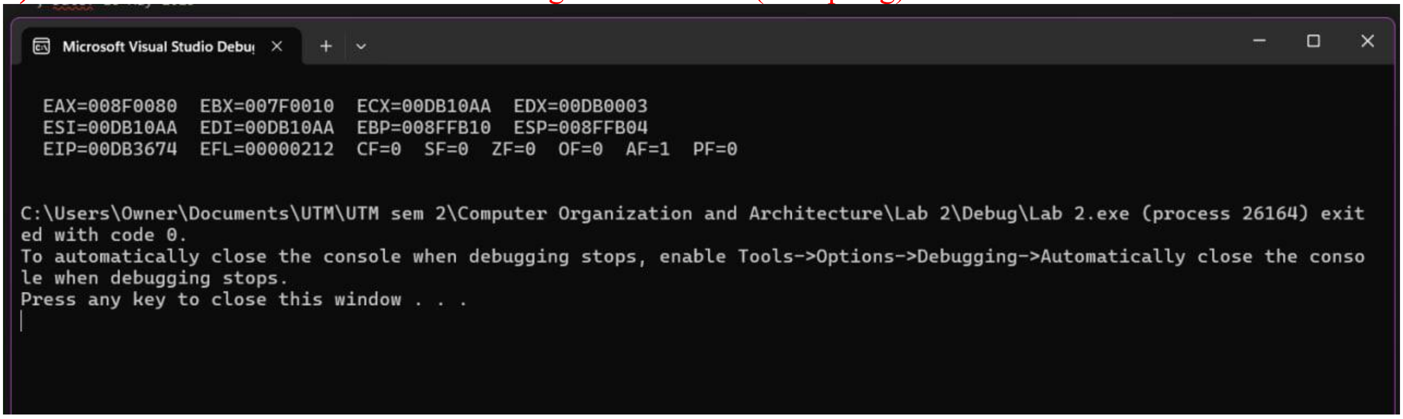
Answer Q6

a) Screenshot of full program (.asm) :



```
TITLE lab2q6
; Author: Firzana And Haani
; Date: 18 May 2023

include irvine32.inc

.code
main proc
; Code Snippet 2 (DIV  BX)
MOV DX, 0; Clear DX to form the 16 - bit dividend in DX : AX
MOV AX, 803h; Load the dividend(8003h) into AX
MOV BX, 10h; Load the divisor(10h) into BX
DIV BX; Divide DX : AX by BX, whereby AX = quotient & DX = remainder

call DumpRegs
exit
main endp
end main
```

b) Paste here the screenshot of the final registers' content (DumpReg):



```
  EAX=008F0080  EBX=007F0010  ECX=00DB10AA  EDX=00DB0003
  ESI=00DB10AA  EDI=00DB10AA  EBP=008FFB10  ESP=008FFB04
  EIP=00DB3674  EFL=00000212  CF=0  SF=0  ZF=0  OF=0  AF=1  PF=0


C:\Users\Owner\Documents\UTM\UTM sem 2\Computer Organization and Architecture\Lab 2\Debug\Lab 2.exe (process 26164) exit
ed with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```