

Department of Computer Science Faculty of Computing UNIVERSITI TEKNOLOGI MALAYSIA

SUBJECT NAME:	COMPUTER ORGANIZATION AND ARCHITECTURE	
SUBJECT CODE:	SECR 1033	
SEMESTER:	2 – 2023/2024	
LAB TITLE:	Lab 2: Arithmetic Equations & Operations	
	Execute the lab in group of two.	
	Student 1 Student 2 No. 1, 3, 5 No 2, 4, 6	
	110. 1, 5, 5	
	Name 1: <u>TEH RU QIAN</u>	
STUDENT INFO:	Metric No: <u>A23CS0191</u>	
	Link for Video Demo:	
	https://youtu.be/z2edwpnfJbU	
	Name 2: <u>GOE JIE YING</u>	
	Metric No: <u>A23CS0224</u>	
	Link for Video Demo: _https://youtu.be/z2edwpnfJbU_	
	Duration of submission:-	
	2 weeks	
SUBMISSION DATE	Submission items in elearning:-	
& ITEMS:	1. Lab 2 exercise sheet/file (in .pdf), with the links for demo	
	video 5 - 10min, on the cover page.	
	2. The assembly programs (in .asm).	

MARKS:	

1

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 word1
var2 word 9
.code
main PROC
      mov ax, var1 ; LINE1
      mov bx, var2
                   ; LINE2
                   ; LINE3
      xchg ax, bx
      call DumpRegs
      exit
main ENDP
END main
```

Answer Q1

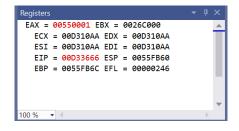
a) Fill (Write) in the contents for the related register in each line:

Table 1

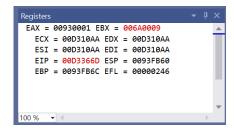
LINE1	AX = 0001h var1 = 0001h	Move the value of var1 (1d) into register AX
LINE2	BX = 0009h var2 = 0009h	Move the value of var2 (9d) into register BX
LINE3	AX = 0009 BX = 0001h	Exchange the value of register AX and BX
LINE4	AX = 0009h var1 = 0009h	Move the value of register AX into variable var1
LINE5	BX = 0001h var2 = 0001h	Move the value of register BX into variable var2

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

LINE1:



LINE2:



LINE3:

LINE4:

LINE5:

```
Registers

EAX = 006F0009 EBX = 00510001

ECX = 00D310AA EDX = 00D310AA

ESI = 00D310AA EDI = 00D310AA

EIP = 00D3367C ESP = 006FF878

EBP = 006FF884 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

		·
LINE 1	EAX = 0000001Ah	Move the value of Xval (26d)
	Xval = 0000001Ah	into register EAX
LINE 2	EAX = FFFFFE6h	Negate register EAX with 2's
	EAA – FFFFFEOII	complement value
LINE 3	EBX = 0000001Eh	Move the value of Yval (30d)
	Yval = 0000001Eh	into register EBX
LINE 4	EBX = FFFFFF6h	Subtract the value of register
	Zval = 00000028h	EBX with the Zval (40d)
LINE 5	EAX = FFFFFFDCh	Add the value of register EBX to
	$\mathbf{EBX} = \mathbf{FFFFFF6h}$	register EAX
LINE 6	$\mathbf{EAX} = \mathbf{FFFFFDDh}$	Increment of EAX by 1
LINE 7	EAX = FFFFFFDDh	Move the value of EAX into
	Rval = FFFFFDDh	variable Rval

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

LINE1:

```
Registers

EAX = 0000001A EBX = 00F08000 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A1015 ESP = 0115FF44

EBP = 0115FF50 EFL = 00000246
```

LINE2:

```
Registers

EAX = FFFFFE6 EBX = 00F08000 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A1017 ESP = 0115FF44

EBP = 0115FF50 EFL = 00000293

0x005A4008 = 0000001E
```

LINE3:

```
Registers

EAX = FFFFFE6 EBX = 0000001E ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A101D ESP = 0115FF44

EBP = 0115FF50 EFL = 000000293

0x005A400C = 00000028
```

LINE4:

```
Registers

EAX = FFFFFFE6 EBX = FFFFFFF6 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A1023 ESP = 0115FF44

EBP = 0115FF50 EFL = 00000287
```

LINE5:

```
Registers

EAX = FFFFFFDC EBX = FFFFFFF6 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A1025 ESP = 0115FF44

EBP = 0115FF50 EFL = 00000283
```

LINE6:

```
Registers

EAX = FFFFFFDD EBX = FFFFFFF6 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A1026 ESP = 0115FF44

EBP = 0115FF50 EFL = 00000287

0x005A4000 = 000000000
```

LINE7:

```
Registers

EAX = FFFFFFDD EBX = FFFFFF6 ECX = 005A1005 EDX = 005A1005

ESI = 005A1005 EDI = 005A1005 EIP = 005A102B ESP = 0115FF44

EBP = 0115FF50 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

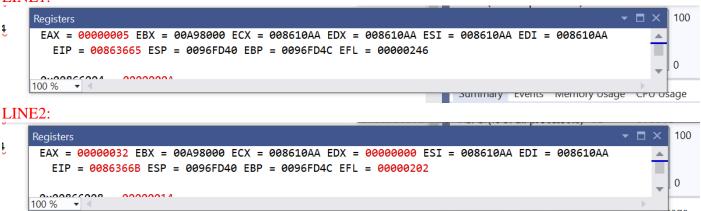
a) Fill (Write) in the contents for the related register in each line:

Table 3

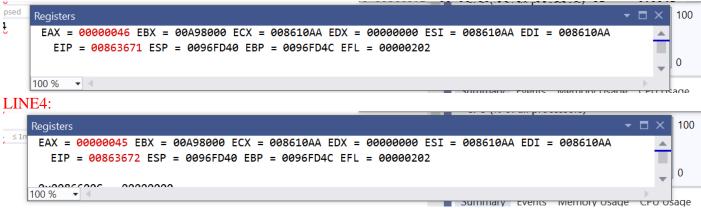
LINE1	EAX = 00000005h	Move the value of var1 (5d)
	var1 = 00000005h	into register EAX
LINE2	EAX = 0000 0032h	Multiply the value of var2(10d)
	var2 = 0000 000Ah	to register EAX
LINE3	EAX = 0000 0046h	Add the value of var3(20d) to
	var3 = 0000 0014h	register EAX
LINE4	EAX = 0000 0045h	Move the value of register EAX
	var4 = 0000 0045h	into variable var4

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

LINE1:







- 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
                      ; LINE1
     mov ax, var1
     mov bx,5
                       ; LINE2
                      ; LINE3
     mul bx
     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 immediate value (5d) into register BX
LINE3	AX = 00C8h BX = 0005h	Multiply the value of register BX with register AX and store the value in register DX
LINE4	BX = 000Ah var2 = 000Ah	Move the value of val2 (10d) into register BX
LINE5	BX = 0007h	Subtract the value in register BX with an immediate value (3d)
LINE6	AX = 001Ch BX = 0007h DX = 0004h	Divide the register AX by register BX, the quotient store in register AX and remainder store in register DX
LINE7	AX = 001Ch var4 = 001Ch	Move the value of register AX into variable var4

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

LINE1:

```
Registers
                                                                           ▼ 🗖 X
 EAX = 00EF0028 EBX = 00DAC000 ECX = 00BE1005 EDX = 00BE1005
   ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE1016 ESP = 00EFFD68
   EBP = 00EFFD74 EFL = 00000246
120 % ▼ <
LINE2:
Registers
                                                                           ▼ 🗖 X
  EAX = 00EF0028 EBX = 00DA0005 ECX = 00BE1005 EDX = 00BE1005
    ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE101A ESP = 00EFFD68
    EBP = 00EFFD74 EFL = 00000246
120 %
LINE3:
Registers
 EAX = 00EF00C8 EBX = 00DA0005 ECX = 00BE1005 EDX = 00BE0000
    ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE101D ESP = 00EFFD68
    EBP = 00EFFD74 EFL = 00000202
```

LINE4:

120 % ▼

```
Registers
  EAX = 00EF00C8 EBX = 00DA000A ECX = 00BE1005 EDX = 00BE0000
    ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE1024 ESP = 00EFFD68
    EBP = 00EFFD74 EFL = 00000202
120 %
LINE5:
Registers
                                                                             \square \times
  EAX = 00EF00C8 EBX = 00DA0007 ECX = 00BE1005 EDX = 00BE0000
    ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE1028 ESP = 00EFFD68
    EBP = 00EFFD74 EFL = 00000202
120 %
LINE6:
Registers
 EAX = 00EF001C EBX = 00DA0007 ECX = 00BE1005 EDX = 00BE0004
   ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE102B ESP = 00EFFD68
   EBP = 00EFFD74 EFL = 00000202
120 %
LINE7:
Registers
 EAX = 00EF001C EBX = 00DA0007 ECX = 00BE1005 EDX = 00BE0004
   ESI = 00BE1005 EDI = 00BE1005 EIP = 00BE1031 ESP = 00EFFD68
   EBP = 00EFFD74 EFL = 00000202
120 % ▼
```

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

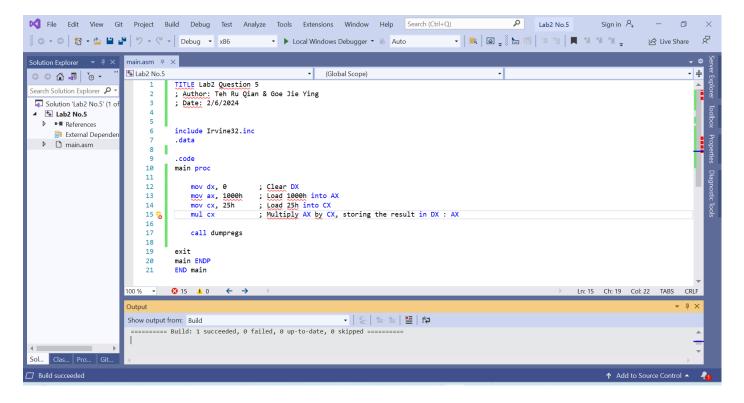
MOV AX, 1000h ; Load 1000h into AX

MOV CX, 25h ; Load 25h into CX

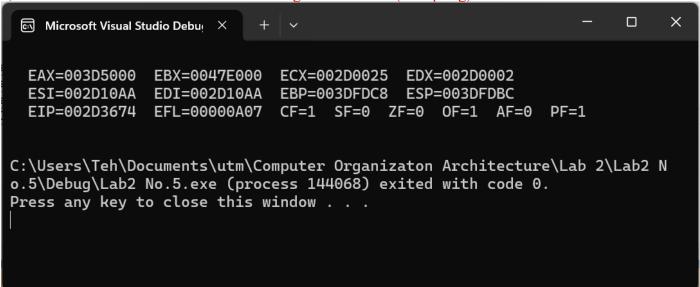
MUL CX ; Multiply AX by CX, storing the result in DX:AX
```

Answer Q5

a) Screenshot of full program (.asm):



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



- 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

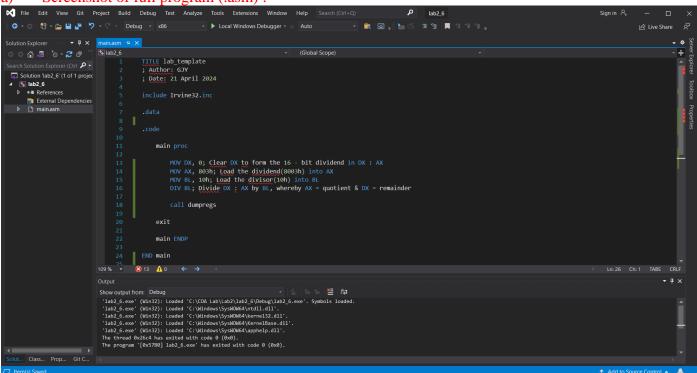
MOV AX, 803h ; Load the dividend (8003h) into AX

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):



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

```
EAX=01370380 EBX=01134010 ECX=006010AA EDX=00600000 ESI=006010AA EDI=0137F8E4 ESP=0137F8D8 EIP=00603671 EFL=00000246 CF=0 SF=0 ZF=1 OF=0 AF=0 PF=1

C:\COA Lab\Lab2\lab2_6\Debug\lab2_6\exe (process 17012) exited with code 0.
To automatically close the console when debugging stops, enable Tools=>Options=>Debugging=>Automatically close the console when debugging stops.

Press any key to close this window . . .
```