**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: December 18, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

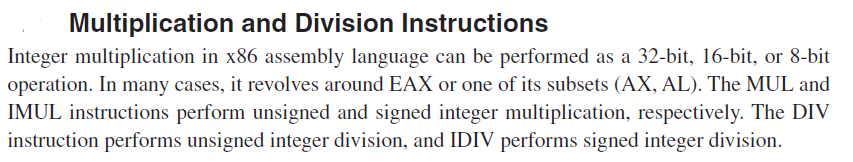
**Computer Organization and**

**Assembly Language (CS235)**

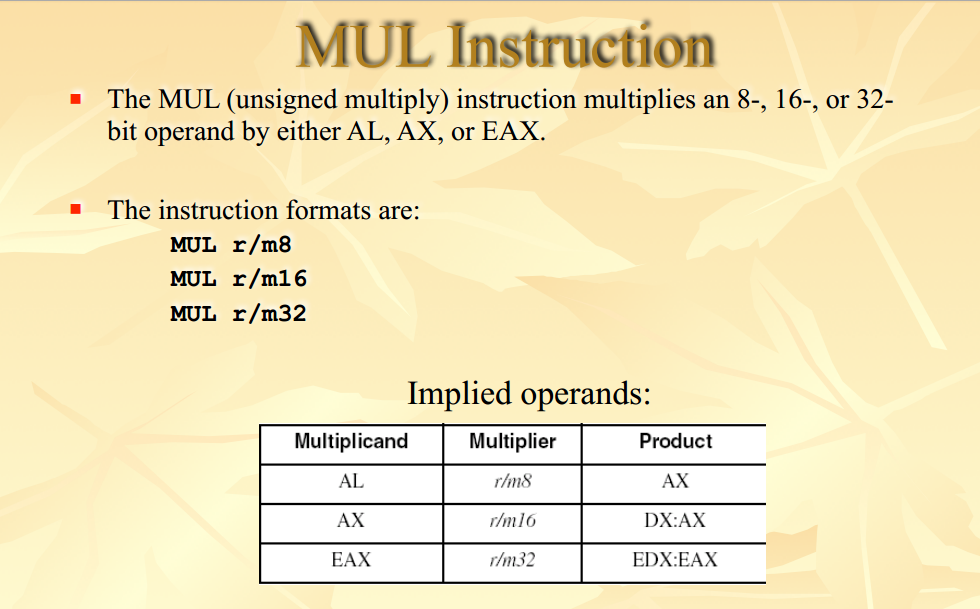
**Lab #10 Multiplication and division in Assembly Language**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team work**  **5 marks** | **Total**  **25 marks** |
| **Fatima Seemab** | **291310** |  |  |  |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |  |  |  |

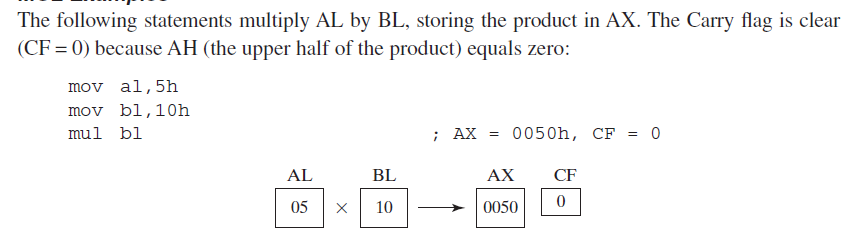
**Objective**: the objective of this lab is to Use **MUL** and **DIV** instruction for multiplication and division in assembly language.

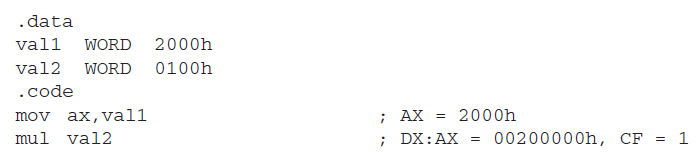


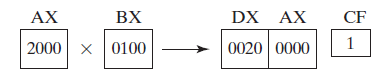


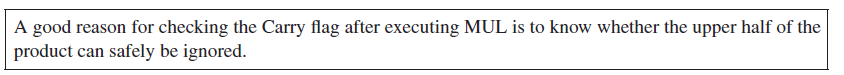




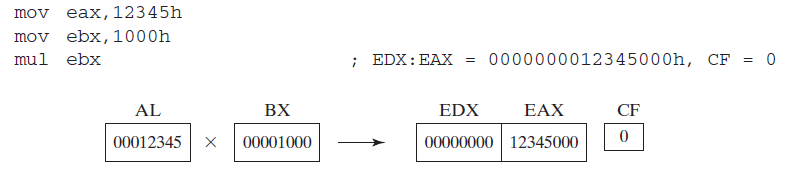




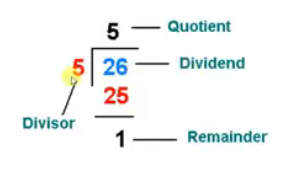


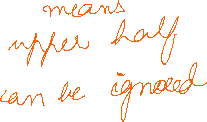


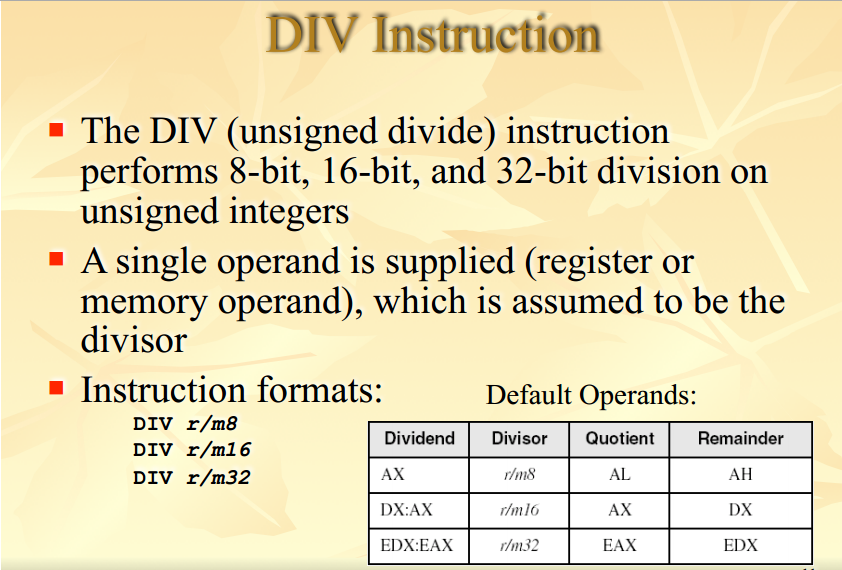
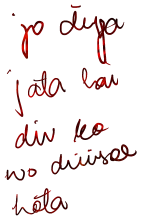
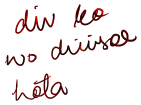




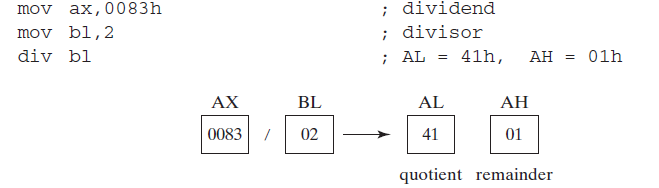
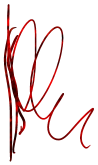
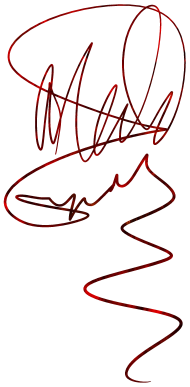
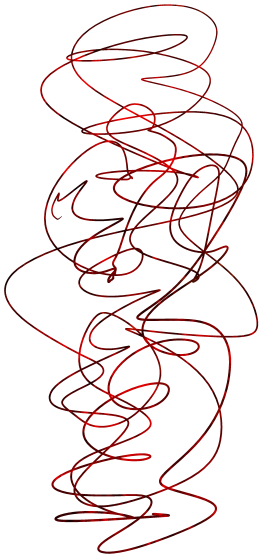
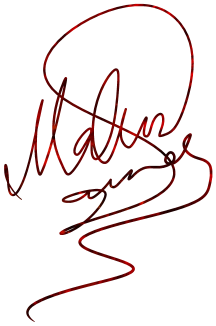


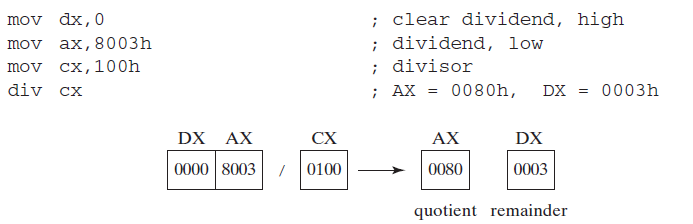


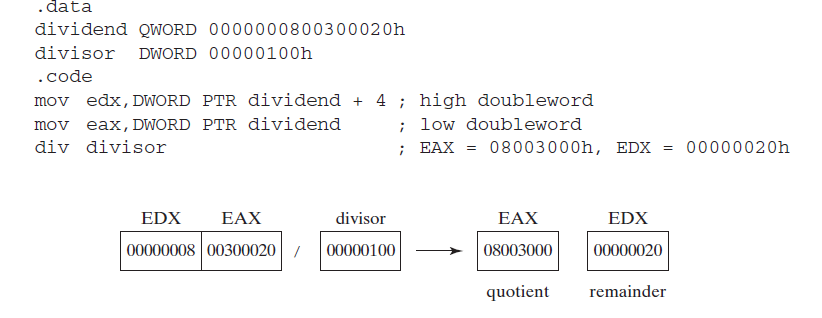












**Lab Task 1:** Write an assembly code using **MUL** to calculate factorial of 16-bit unsinged number.

**Code:**

TITLE factorial of 16-bit unsigned number

INCLUDE Irvine32.inc

.data

sInput BYTE "Enter a 16-bit unsinged number: ",0

sfact BYTE "Factorial: ",0

value WORD ?

.code

main proc

mov edx, offset sInput

call writeString

call readInt

mov value, ax ; move 16-bit value

movzx ecx, value ; loop value times

COMMENT!

16-bit multiplication correctly calculates till 8 factorials

As numbers become bigger factorial become enormous which cannot be multiplied

using mul instruction as it only allows multiplicand to be 32-bit at most.

Even using 32-bit multiplication we can correctly calculate factorial till 12 only.

!

mov eax, 1

l1:

mul cx

loop l1

call dumpregs ; ax contains factorial

mov edx, offset sfact

call writestring ; display factorial in decimal

call writeDec

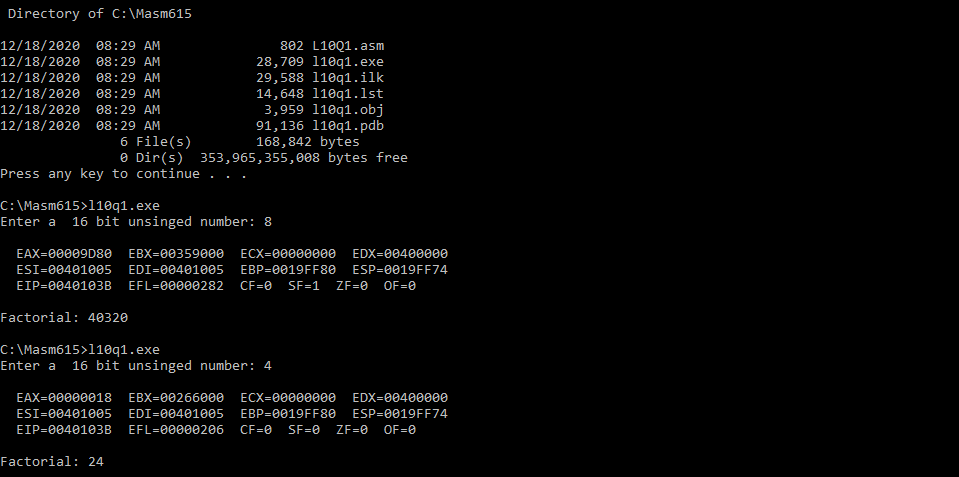
call crlf

exit

main endp

end main

**Output:**



**Lab Task 2:** Write an assembly code which ask user to input time in secs and convert that time into corresponding hrs., min and secs.

**Code:**

TITLE secs to hours, min, secs

INCLUDE irvine32.inc

.data

sInput BYTE "Enter the time in seconds to get corresponding hrs, min and secs: ",0

shour BYTE " hr ",0

smin BYTE " min ",0

ssec BYTE " sec ",0

hr WORD ?

min BYTE ? ; as minutes and second are always less than 60

sec BYTE ?

seconds DWORD ?

.code

main proc

mov edx, offset sInput

call writeString ; prompt user

call readInt ; take input

mov seconds, eax

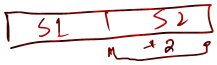


call SecToHours ; method call

exit

main endp





SecToHours proc uses ax cx dx

;-----------------------------------------------

; calculates hrs. min and sec

; Receives seconds in eax



; calls printTime proc

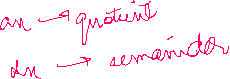


;-----------------------------------------------

; for hours



mov dx, WORD PTR seconds + 2



mov ax, WORD PTR seconds



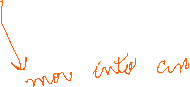
mov cx, 3600 ; 32-bit dividend, 16-bit divisor

div cx

; ax contains hours



; dx contains remainder



; for minutes

mov hr, ax

mov ax, dx

mov cl, 60

div cl ; 16-bit dividend, 8-bit divisor



; al contains minutes

; ah contains seconds



mov min, al

mov sec, ah

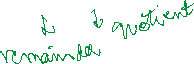
call printTime

ret



SecToHours endp

printTime proc



;-----------------------------------------------



; uses variables hr, min and sec

; prints time on console

;-----------------------------------------------

movzx eax, hr

call writedec ; print hours

mov edx, offset shour

call writeString

movzx eax, min ; print minutes

call writedec

mov edx, offset smin

call writeString

movzx eax, sec ; print seconds

call writedec

mov edx, offset ssec

call writeString

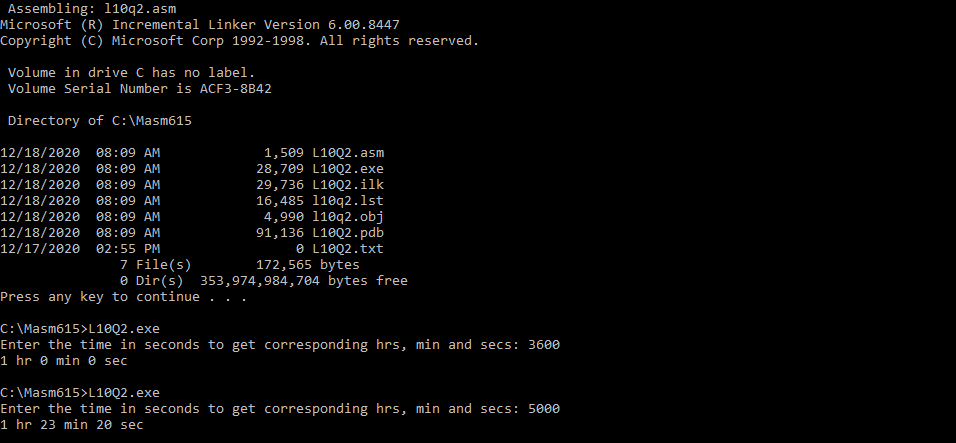
call crlf

ret

printTime endp

end main

**Output:**

****