**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: December 12, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

**Computer Organization and**

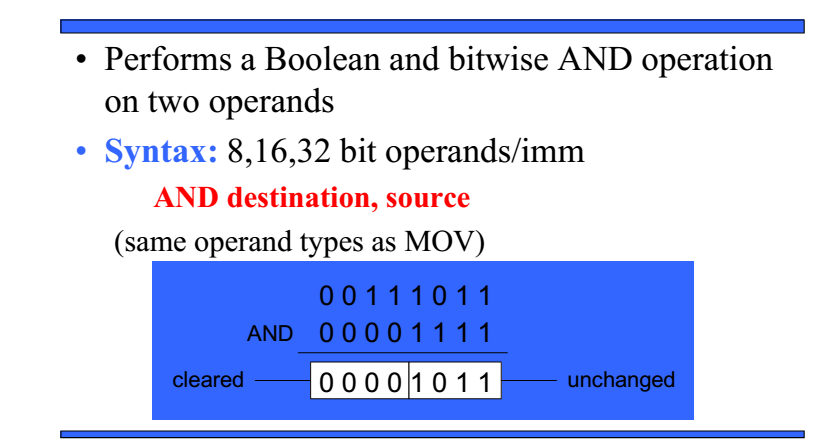
**Assembly Language (CS235)**

**Lab #8 Bit wise and Shift operations in Assembly Language**

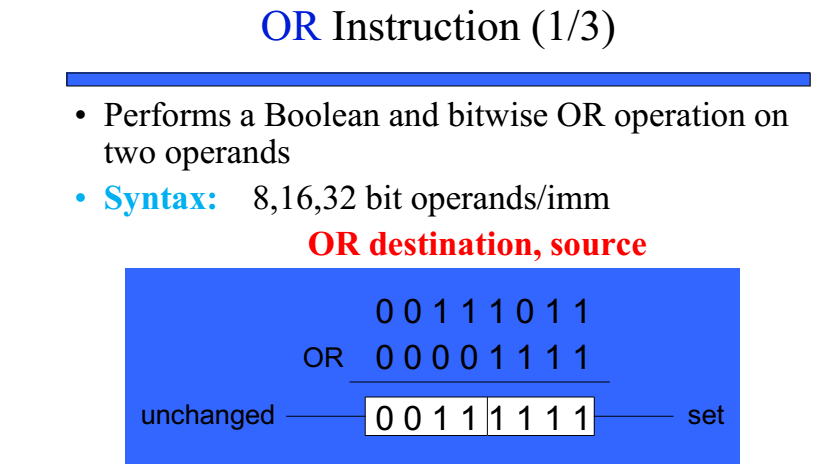
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics 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 bitwise operator for multiplication in Assembly language

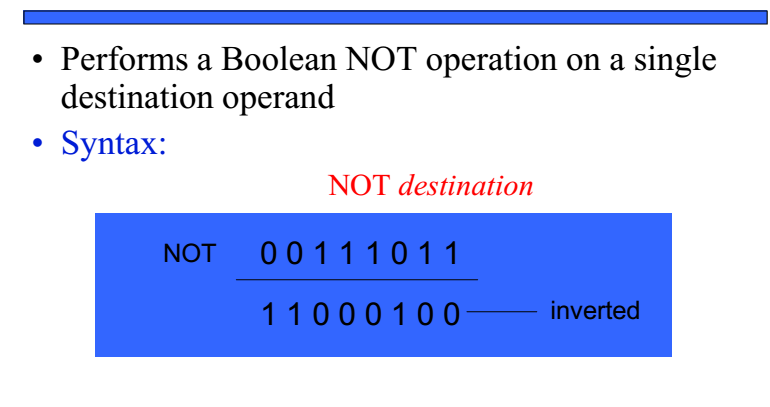














**Mask**: A mask is a bit pattern that is defined by a programmer, which allows specific bits in a piece of data to be set, cleared, tested or altered.



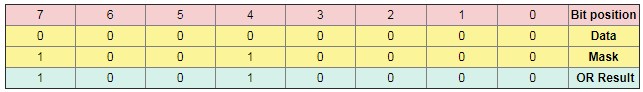
**Setting Bits to 1:**

To turn on a specific bit, we can use the OR bitwise operation and a suitable mask.



For example, if we need to turn on Bit 4 and Bit 7 of a byte (remember that the bit on the right hand side is Bit 0), you can use the mask 1001 0000 and the OR bitwise operation.



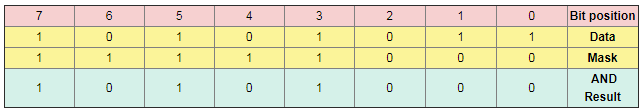


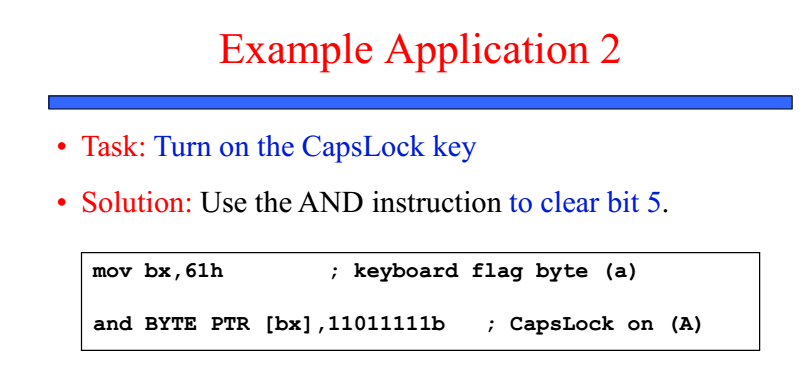
**Resetting/Clear bits to 0**  
We can't force a bit to be 0 using the OR command. We can use the bitwise command AND with a suitable mask,



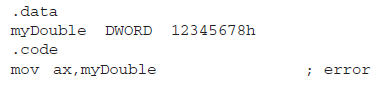
For example, we want to reset Bits 0, 1 and 2 in a byte but leave all the other bits as they were. You would use the mask 1111 1000 along with the AND bitwise operator.













 ;





**The TEST instruction**

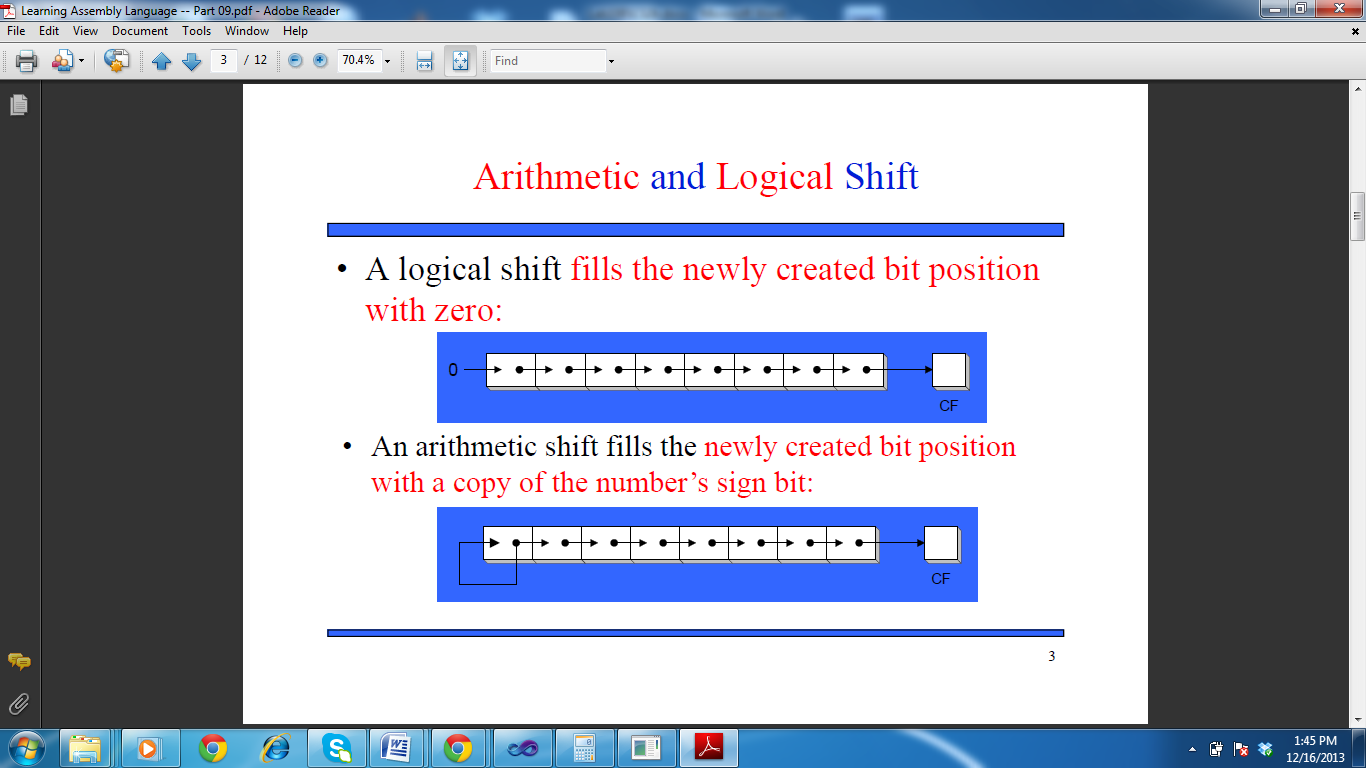


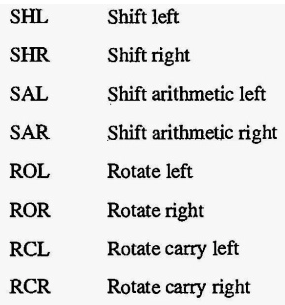
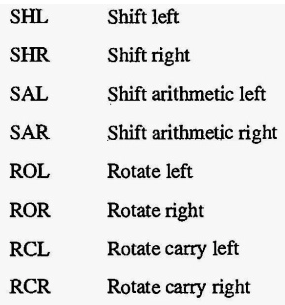
The test [instruction](https://en.wikipedia.org/wiki/Instruction_(computing)) performs a [bitwise AND](https://en.wikipedia.org/wiki/Bitwise_AND) on two [operands](https://en.wikipedia.org/wiki/Operand). The [flags](https://en.wikipedia.org/wiki/FLAGS_register) [SF](https://en.wikipedia.org/wiki/Sign_flag), [ZF](https://en.wikipedia.org/wiki/Zero_flag), [PF](https://en.wikipedia.org/wiki/Parity_flag) are modified while the result of the [AND](https://en.wikipedia.org/wiki/Bitwise_AND) is discarded.



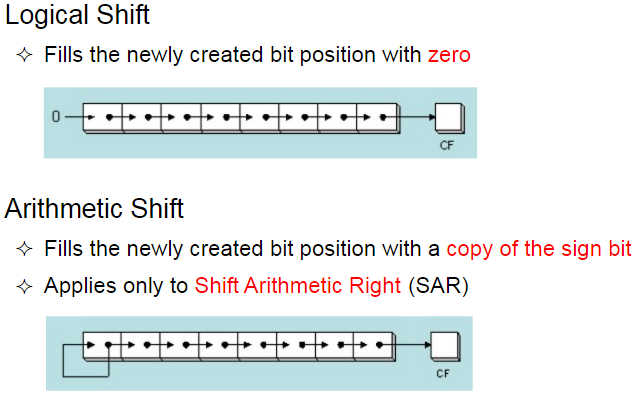
* **TEST *destination, source***
  + Performs AND, **does not** store the result
  + **Flags are set**



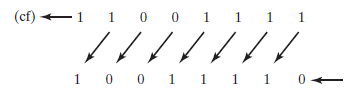
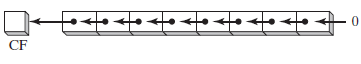


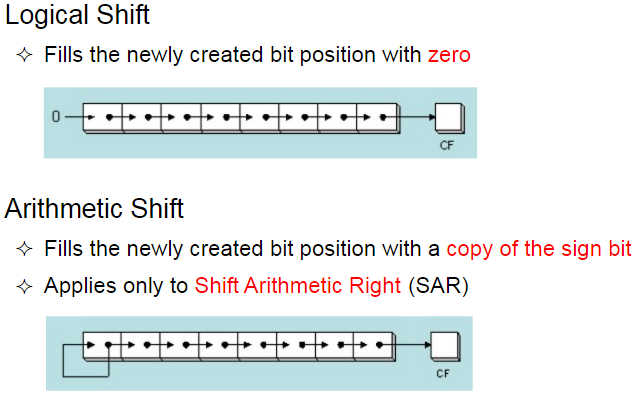




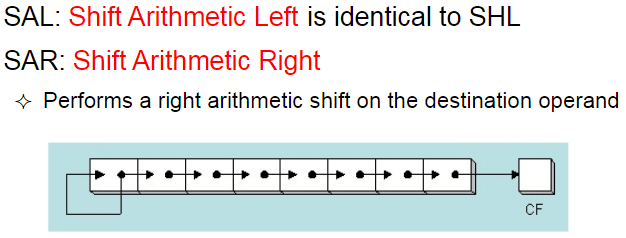


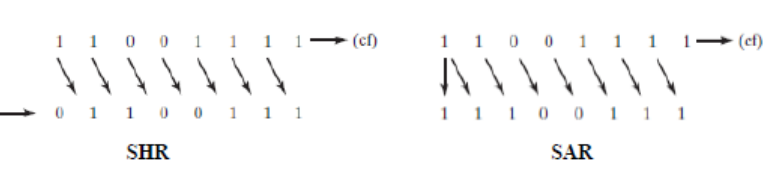


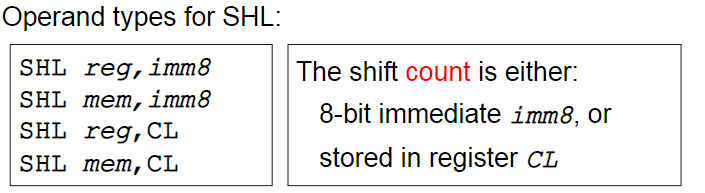


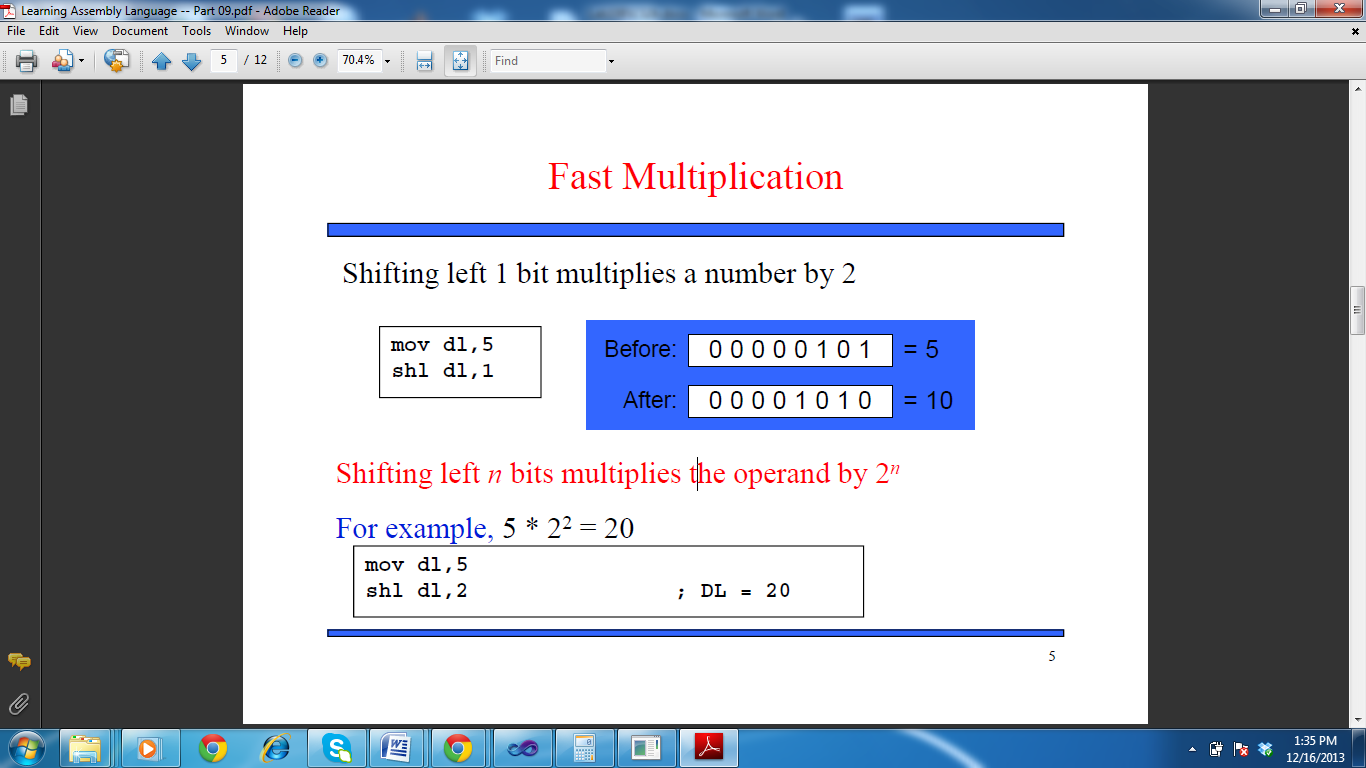
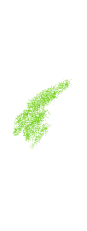
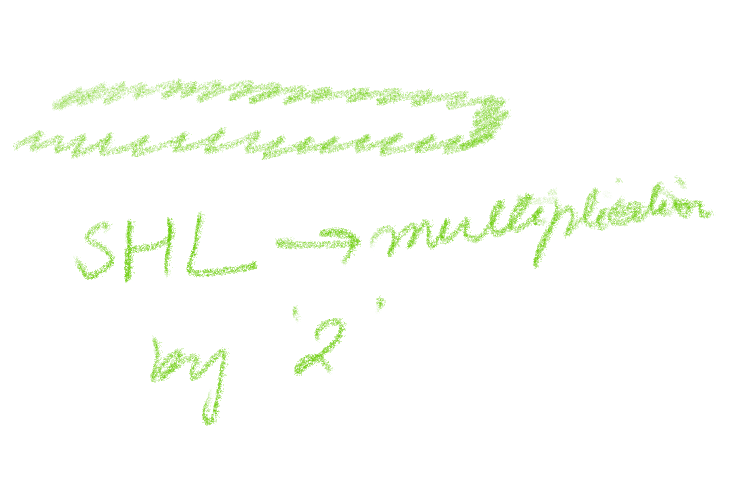
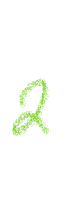
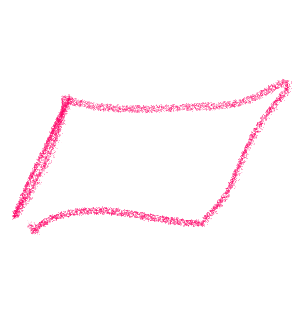














**Q#1** Write assembly code to find number of ones in32-bit hex number using bit wise operation.

**Code:**

TITLE Number of Ones in 32-bit Hex Number

INCLUDE irvine32.inc

.data

sInput BYTE "Enter a 32-bit Hex Number: ", 0



sOutput BYTE "Number of ones in 32-bit Hex Number are ",0

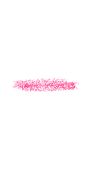
.code

main proc

mov edx, offset sInput

call WriteString ; prompt for input

call ReadHex



mov ebx,eax

call NumberOfOnes ; method call

exit

main endp

NumberOfOnes proc uses ebx

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

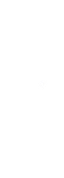
; Receives 32-bit hex number in ebx

; Returns number of ones in eax



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



mov eax,0 ; initialize ecx



NotZero:



shr ebx,1 ; shift right and check CF



 jnc L ; if carry flag is not 1 jump

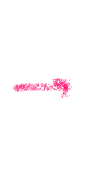


inc al ; else increment eax



L:



 cmp ebx,0 ; loop until ebx becomes 0



jnz NotZero



mov edx, offset sOutput

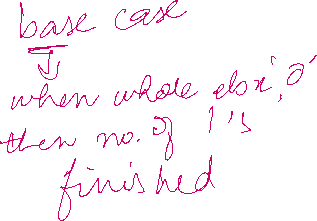


call WriteString

call WriteDec



call crlf

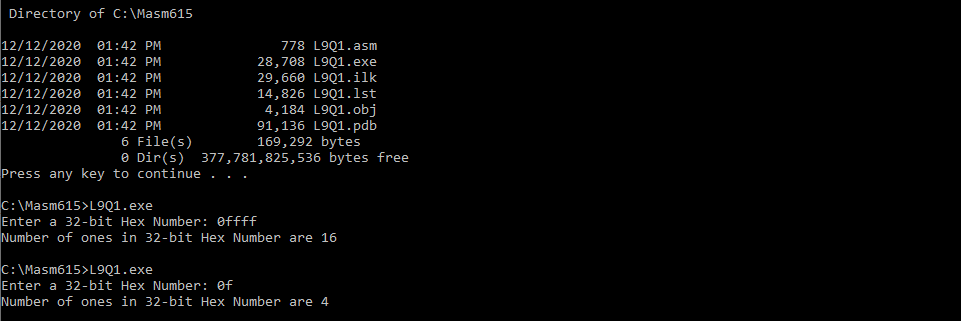


ret

NumberOfOnes endp

end main

**Output:**





**Q#2** Write equivalent assembly code of given C++ program

void main(){

Int a,b,result;

printf("nEnter the numbers to be multiplied :");

scanf("%d%d",&a,&b); // Provide input such that a>b

result=0; // if b>a , result is 0

while(b != 0) // Iterate the loop till b==0



{



if (b&01) // Bitwise & of the value of b with 01

{



result=result+a; // Add a to result if b is odd .



}

a<<=1; // Left shifting the value contained in 'a' by 1



// multiplies a by 2 for each loop



b>>=1; // Right shifting the value contained in 'b' by 1.

}



printf("nResult:%d",result); //display result using writeint comand

}

**Code:**

TITLE Multiply 2 Numbers

INCLUDE irvine32.inc

.data

a DWORD ?

b DWORD ?

result DWORD 0

sInput BYTE "Enter the numbers to be multiplied: ", 13, 10, 0

sOutput BYTE "Result: ", 0

.code

main proc

mov edx, offset sInput

call WriteString ; Prompt user

call ReadInt ; Read 1st integer

mov a, eax



call ReadInt ; Read 2nd integer

mov b, eax



call Mult

exit

main endp

Mult proc uses eax

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

; Receives the numbers to be multiplied in a and b

; Returns answer in 'result'

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

; to be used for storing result as

; add instruction cannot have memory

; location as both destination and source

mov eax, 0

cmp b, 0



jz L ; Do not loop if b = 0



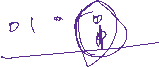
bNotZero:



test b, 01



jz bNotOdd



add eax, a



bNotOdd:



shl a, 1 ; Left shifting the value in 'a' by 1



shr b, 1 ; Right shifting the value in 'b' by 1



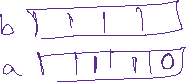
cmp b, 0



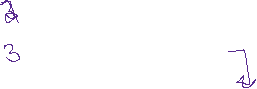
jnz bNotZero ; Iterate the loop till b = 0



L:



mov edx, offset sOutput



call WriteString ; Display output

call WriteInt



; move eax to result



mov result, eax ; Move eax to result

call crlf

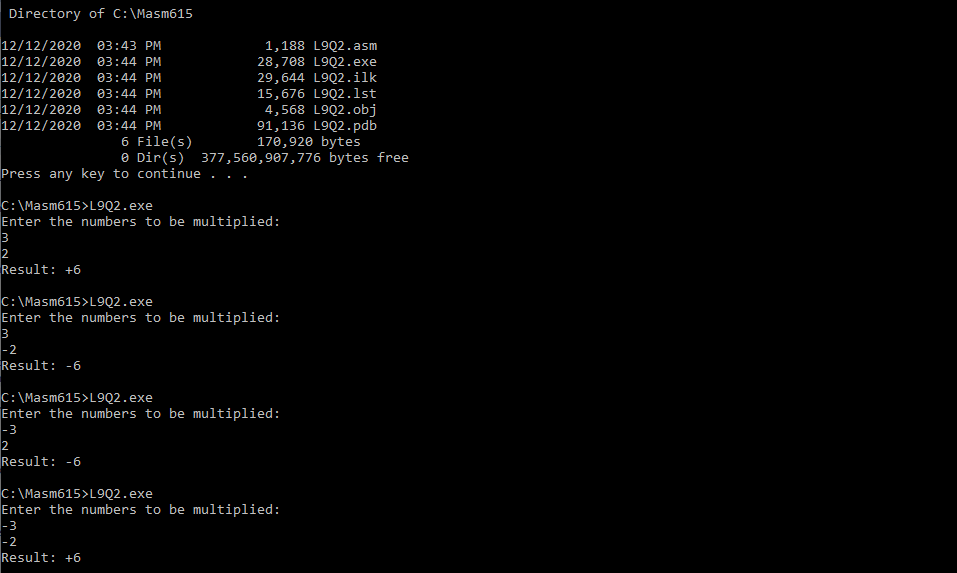


ret

Mult endp

end main

**Output:**





**Q#3** Write assembly code to take a string from user and convert the string into all capital**.**

(Print your name, do not use the spaces)

**Code:**

TITLE Convert the String into all CAPITAL

INCLUDE irvine32.inc

.data

string BYTE ?

sInput BYTE "Enter a string without space to convert to UPPERCASE: ",13,10,9,0

sOutput BYTE "In uppercase: ",13,10,9,0

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

; 13 = carriage return



; 10 = line field



; 9 = horizontal tab



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

.code

main proc

mov edx, offset sInput ; prompt user

call WriteString

mov edx, offset string ; read string

mov ecx, 64 ; maximum number of characters + 1 for string-end indicator



call readString ; returns length of entered string in al

; input string should not contain space

; because AND-ing mask 11011111b

; with 00100000b (ascii for space) will

; result into 00000000b i.e. NULL

; the end-string indicator

mov ecx, lengthof string

call allCapital ; method call

mov edx, offset sOutput

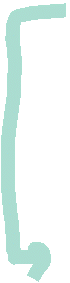
call WriteString ; display output

mov edx, offset string

call WriteString

call crlf

exit



main endp

allCapital proc uses edx

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

; Recieves edx = offset of 'string' variable

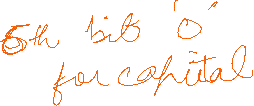
; al = Number of characters entered by user

; Returns string in upper case in same variable

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

movzx ecx, al

; 11011111b is the MASK to turn caps lock on



L:

and BYTE PTR [edx], 11011111b

inc edx



loop L

ret

allCapital endp

end main

**Output:**

