**Assignment No-08**

|  |
| --- |
| **SUBJECT: MICROPROCESSOR LAB (MPL)**  **NAME: PRIYANKA SALUNKE** |
| **CLASS: SE COMP A ROLL NO.: F19111151** |
| **SEMESTER: SEM-IV YEAR: 2020-21** |
| **DATE OF PERFORMANCE: DATE OF SUBMISSION:** |
| **EXAMINED:** |

**Title:-**Multiplication

**Assignment Name: -** Write X86/64 ALP to perform multiplication of two 8-bit hexadecimal numbers. Use successive addition and add and shift method. Accept input from the user.

**Objective-**

* To understand the different algorithm for multiplication.
* To understand how to write procedure.

**Outcome-**

* Students will be able to write code for doing multiplication.

### [Prerequisite](http://dictionary.reference.com/browse/prerequisite) -

System call of Unix for Assembly language Program.

**Hardware Requirement-**

Desktop PC

**Software Requirement-**

Ubuntu 14.04,

Assembler: NASM version 2.10.07

Linker: ld

**Introduction:-**

**Guidelines for the algorithm:**

1. Display the menu.

Enter “1” – “ADD AND SHIFT METHOD.”

Enter “2” – “SUCCESSIVE ADDITION METHOD”.

Enter “3” – EXIT

2) Take choice from user then go to the respective subroutines.

ADD AND SHIFT METHOD

1. Initialize code and bss sections.
2. Accept multiplier and multiplicand variables in data segment.
3. Initialize product variable to zero.
4. Set count as number of bits in operand, which is 8.
5. Shift product to left by 1 bit and insert zero as LSB.
6. Transfer MSB of multiplier to carry flag by rotating it to left.
7. Check if carry flag is set or not. If yes add multiplicand to product.
8. Decrement count by 1.
9. Check count=0 else repeat step 5 through step 9 till count=0.
10. Display the final product.

#### SUCCESSIVE ADDITION METHOD

1. Define product=0.
2. Set count=multiplicand.
3. Add product=product + multiplier.
4. Decrement count.
5. Repeat step 3 and 4 till count=0
6. Display product variable value as final product.

**Conclusion: -** Hence we implemented an ALP to do multiplication.

**Questions:-**

1. Explain successive addition algorithm with example?
2. Explain what is Interrupt?

**Program:-**

%macro dispmsg 2 ;macro for display

mov rax,1 ;standard ouput

mov rdi,1 ;system for write

mov rsi,%1 ;display message address

mov rdx,%2 ;display message length

syscall ;interrupt for 64-bit

%endmacro ;close macro

%macro exitprog 0 ;macro for exit

mov rax,60 ;system for exit

mov rdx,0

syscall ;interrupt for 64-bit

%endmacro ;close macro

%macro gtch 1 ;macro for accept

mov rax,0 ;standard input

mov rdi,0 ;system for read

mov rsi,%1 ;input the message

mov rdx,1 ;message length

syscall ;interrupt for 64-bit

%endmacro ;close macro

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

section .data

nwline db 10

m0 db 10,10,"Program to multiply two numbers using successive addition and add-and-shift method"

l0 equ $-m0

m1 db 10,"1. Successive Addition method",10,"2. Add-and-Shift method",10,"3. Exit",10,10, "Enter your choice (1/2/3 <ENTER>): "

l1 equ $-m1

m2 db 10,"Enter multiplicand (2 digit HEX no) : "

l2 equ $-m2

m3 db 10,"Enter multiplier (2 digit HEX no) : "

l3 equ $-m3

m4 db 10,"The Multiplication is : "

l4 equ $-m4

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

section .bss

mcand resq 1 ;reserve 1 quad for multiplicand

mplier resq 1 ;reserve 1 quad for multiplier

input resb 1 ;reserve 1 byte for input

output resb 1 ;reserve 1 byte for output

choice resb 1 ;reserve 1 byte for choice

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

section .text

global \_start ;starting of main program

\_start :

dispmsg m0,l0 ;Displaying the menu

back:

dispmsg m1,l1 ;Displaying the first message

gtch input ;To read and discard ENTER key pressed.

mov al, byte[input] ;Get choice

mov byte[choice],al

gtch input ;To read and discard ENTER key pressed.

mov al, byte[choice]

cmp al, '1' ;compare contents of al with 1

je succ\_add ;if equal the jump to succ\_add procedure

cmp al, '2' ;compare the contents of al with 2

je shft\_add ;if equal the jump to shft\_add procedure

cmp al, '3' ;compare the contents of al with 3

jnz back ;if not zero then jump to back

exitprog ;exit program

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

; SUCCESSIVE ADDITION

succ\_add: ;succ\_add procedure

dispmsg m2,l2 ;Displaying the second message

call getnum ;call getnum procedure

mov [mcand],rax ;mov contents of rax(multiplicand) into mcand buffer

gtch input ;To read and discard ENTER key pressed.

dispmsg m3,l3 ;Displaying the third message

call getnum ;call getnum procedure

mov [mplier],rax ;mov contents of rax(multiplier) into mplier buffer

gtch input ;To read and discard ENTER key pressed.mov rax,0

dispmsg m4,l4 ;Displaying the fourth message

mov rax,0 ;clearing rax register

cmp qword[mplier],0 ;compare contents of mplier buffer in qword with 0

jz ll5 ;if zero jump to loop 5

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

ll1: ;loop 1

add rax,qword[mcand] ;add contents of mcand buffer in qword to contents of rax register

dec qword[mplier] ;decrement contents of mplier buffer

jnz ll1 ;if not zero jump to loop 1

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

ll5: ;loop 5

call disphx16 ;call disphx16 procedure to displays a 8 digit hex number in rax

jmp back ;jump to back after execution

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

; ADD & SHIFT

shft\_add: ;shft\_add procedure

dispmsg m2,l2 ;Displaying the second message

call getnum ;call getnum procedure

mov [mcand],rax ;mov contents of rax(multiplicand) into mcand buffer

gtch input ;To read and discard ENTER key pressed.

dispmsg m3,l3 ;Displaying the third message

call getnum ;call getnum procedure

mov [mplier],rax ;mov contents of rax(multiplier) into mplier buffer

gtch input ;To read and discard ENTER key pressed.

mov rax,0 ;clearing the rax register

dispmsg m4,l4 ;Displaying the fourth message

mov rax,0 ;clearing the rax register

mov rcx,8 ;taking count of 8 in rcx register

mov rdx,qword[mplier] ;multiplier is 8 bits so it occupies dl

mov rbx,qword[mcand] ;mupltiplicand is 8 bits so it occupies bl

;we will put Q in higher 8 bits of ax (i.e. ah)

;and multipler in lower 8 bits of ax (i.e. al)

mov ah,0 ;clearing ah register

mov al,dl ;ah already 0 and al now contains multiplier

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

ll3: ;loop 3 (s3)

mov dh,al ;mov contents of al into dh as dh is used as temporary

and dh,1 ;check d0 bit of multiplier

jz ll8 ;if d0 bit was zero, Z flag will be set (s2)(if zero jmp to loop 8)

add ah, bl ;d0 bit of multiplier is set

;so add multiplicand to Q(add bl into ah)

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

ll8: ;loop 8 (s2)

shr ax,1 ;shift both Q (ah) and muplitiplier (al) right 1 bit

dec rcx ;decrement contents of rcx

jnz ll3 ;if not zero then jump to loop 3 (s3)

call disphx16 ;call procedure disphx16

jmp back ;jump to back

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

getnum: ;Procedure to get a 2 digit hex no from user

; number returned in rax

mov cx,0204h ;02 digits to display and 04 count to rotate

mov rbx,0 ;clearing rbx register

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

ll2: ;loop 2

push rcx ;syscall destroys rcx.Rest all regs are preserved

gtch input ;To read and discard ENTER key pressed.

pop rcx ;pop the contents of rcx

mov rax,0 ;clearing the contents of rax

mov al,byte[input] ;Get choice

sub rax,30h ;subtract 30h from contents of rax

cmp rax,09h ;compare the contents of rax register with 09h

jbe skip1 ;if equal then jump below to skip1 label

sub rax,7 ;subtract 7 from contents of rax register

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

skip1: ;skip1 label

shl rbx,cl ;shift multiplicand and count to the left

add rbx,rax ;add contents of rax register to the contents of rbx register

dec ch ;decrement the contents of ch register

jnz ll2 ;if not zero then jump to loop 2

mov rax,rbx ;mov contents of rbx register into rax register

ret ;return

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

disphx16: ;Displays a 16 digit hex number passed in rax

mov rbx,rax ;move contents of rax register into rbx register

mov cx,1004h ;16 digits to display and 04 count to rotate

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

ll6: ;loop 6

rol rbx,cl ;rotate multiplicand and count to the left

mov rdx,rbx ;mov contents of rbx register into rdx register

and rdx,0fh ;anding contents of rdx register with 0fh

add rdx,30h ;adding contents of rdx register with 30h

cmp rdx,039h ;comparing the contents of rdx register with 39h

jbe skip4 ;if equal then jump below to skip4 label

add rdx,7 ;add 7 to the contents of rdx register

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

skip4: ;skip4 label

mov byte[output],dl ;mov contents of dl register into output buffer in bytes

push rcx ;push the contents of rcx register

dispmsg output,1 ;Displaying the output

pop rcx ;pop the contents of rcx

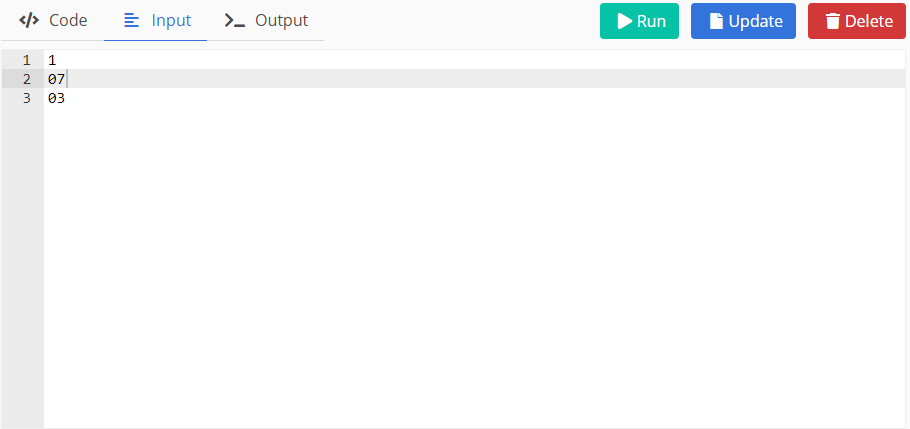
dec ch ;decrement the count(contents of ch)

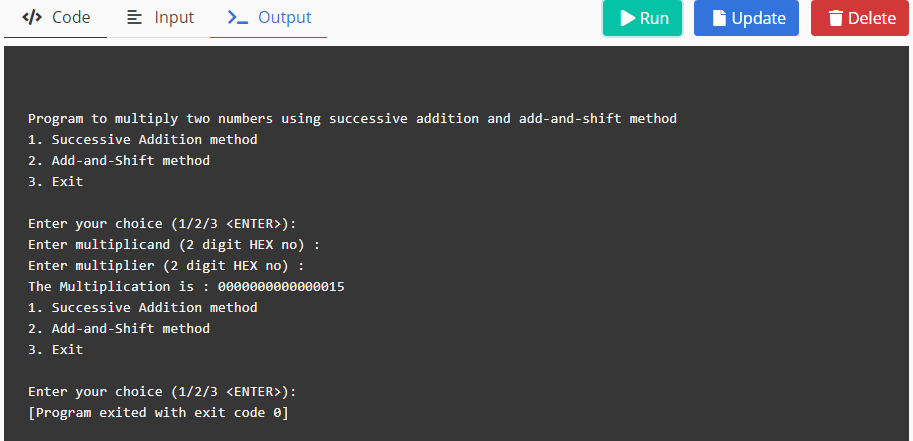
jnz ll6 ;if not zero the jump to loop 6

ret ;return

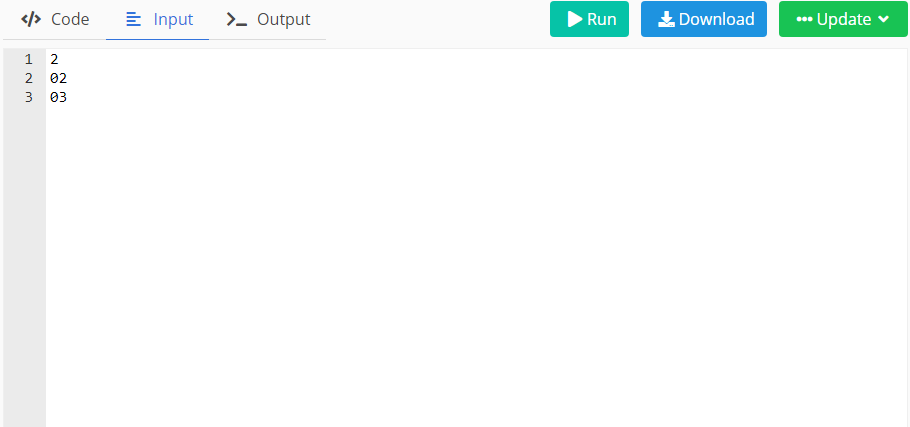
**Output:-**

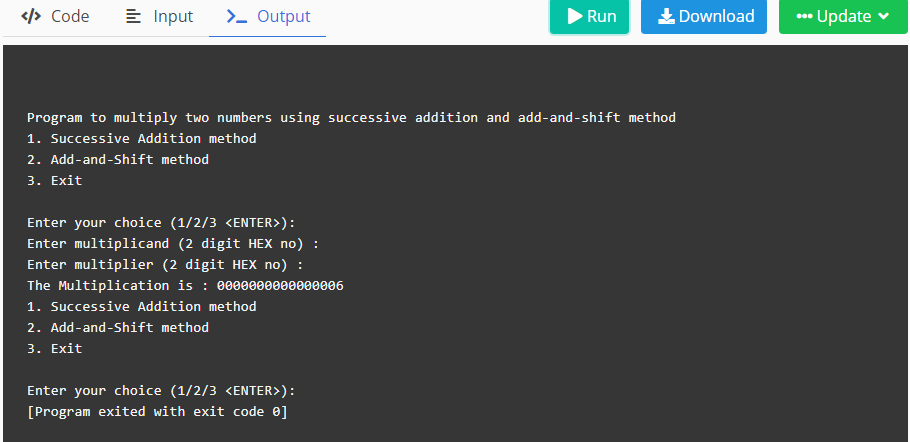
1)





2)





2)