Subject: Microprocessor Lab

Assignment No. 05

Roll No: **21118**

Batch: E-1

Problem Statement:

Write a switch case driven ALP to perform 64-bit hexadecimal arithmetic operations (+, -, \*, %) using macros. Define procedure for each operation.

Hardware of PC:

* Manufacturer and model: Acer Swift-3
* Processor: Intel core i5 – 8265U @1.60 GHz
* Memory: 8GB of DDR4 RAM and 512GB of ROM
* System Type: 64-bit OS, x-64 based PC

Software Used:

* Operating system: Ubuntu 20.04 LTS on oracle virtual machine
* Text editor: Gedit (version: 3.36.2)
* Assembler: NASM (version: 2.14.02)

Theory:

*Instructions:*

1. *Add – Integer Addition:*
   1. This is going to add the number in source with number in destination. The result is stored in destination.
   2. Syntax:

add <reg>, <reg>

add <reg>, <mem>

add <mem>, <reg>

add <reg>, <con>

add <mem>, <con>

1. *Sub – Integer Subtraction:*
   1. The sub instruction stores in the value of its first operand the result of subtracting the value of its second operand from the value of its first operand.
   2. Syntax:

sub <reg>, <reg>

sub <reg>, <mem>

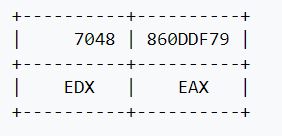
sub <mem>, <reg>

sub <reg>, <con>

sub <mem>, <con>

1. *Mul:*
   1. The mul instruction is used to perform a multiplication. Always multiplies EAX by a value.
   2. The result of the multiplication is stored in a 64-bits value across EDX (most significant 32 bits of the operation) and EAX (least significant 32 bits of the operation).
   3. Syntax: mul value
   4. Example:

mul 0x10 - Multiplies EAX by 0x10 and stores the result in EDX:EAX.



1. *Div:*
   1. The div instruction is used to perform a division. Always divides the 64 bits value across EDX:EAX by a value. The result of the division is stored in EAX and the remainder in EDX.
   2. Syntax: div value

*Procedure:*

* Procedures in assembly are equivalent to functions in c++.
* Syntax:

*proc\_name:*

// procedure body

// ..

ret

* Procedure can be called from another procedure by

CALL *proc\_name*

* The called procedure returns the control to the calling procedure by using the RET instruction.

Algorithm:

* *Hex to ascii conversion:*
  1. Initialize counter to 2
  2. Set rsi pointer to slen
  3. Use rol instruction to reverse the number (rol al,04h).
  4. Move al to bl.
  5. Perform and operation on bl with 0Fh
  6. Compare bl with 09h
  7. If bl is less than or equal to 09h then goto step 9
  8. Else add 07h to bl
  9. Add 30h to bl and move the content of bl in rsi.
  10. Increment rsi and decrement counter
  11. if counter is not zero then goto step 3
  12. Else display the content of slen
* *Arithematic Operations:*
  1. Ask user for its choice.
  2. If choice is 1, call addition procedure.
  3. If choice is 2, call subtraction procedure.
  4. If choice is 3, call multiplication procedure.
  5. If choice is 4, call division procedure.
  6. If choice is different the exit from the program.
  7. Display the result as per user choice

Program:

; Shubham (21118)

; MPL Assignment 05

%macro rwm 3

mov rax, %1

mov rdi, 01

mov rsi, %2

mov rdx, %3

syscall

%endmacro

section .data

opnd1 dq 0000000000000007 ; operands for binary arithematic operations

opnd2 dq 0000000000000003

menu: db "Enter", 0xA, "1 for Addition", 0xA, "2 for Subtraction", 0xA, "3 for Multiplication", 0xA, "4 for Division", 0xA, "0 to Exit", 0xA, ": "

menul equ $-menu

q db "Quotient : "

m1: equ $-q

nwln db 0xA

section .bss

ascii\_num resb 16 ; array for storing ascii of hex

choice resb 02 ; users choice to perform operation

section .text

global \_start

\_start:

rwm 01, menu, menul

rwm 00, choice, 02 ; getting operation choice

cmp byte[choice], 30h

JE exit ; je -> jmp if equal

cmp byte[choice], 31h

JE op1

cmp byte[choice], 32h

JE op2

cmp byte[choice], 33h

JE op3

cmp byte[choice], 34h

JE op4

JMP exit

op1 :

call Addition

JMP exit

op2 :

call Subtraction

JMP exit

op3 :

call Multiplication

JMP exit

op4 :

call Division

JMP exit

exit:

mov rax,60

mov rdi,0

syscall

; Procedures for arithematic operations

Addition:

mov rax,[opnd1]

mov rbx,[opnd2]

add rax,rbx

call conv\_and\_display

rwm 01, nwln, 01

ret

Subtraction:

mov rax,[opnd1]

mov rbx,[opnd2]

sub rax,rbx

call conv\_and\_display

rwm 01, nwln, 01

ret

Multiplication:

mov rax,[opnd1]

mov rbx,[opnd2]

mul rbx

mov r9,rax

xor rax,rax ; clearing rax

mov rax,rdx

call conv\_and\_display ; displaying rdx

xor rax,rax ; clearing rax

mov rax,r9

call conv\_and\_display ; displaying rax

rwm 01, nwln, 01

ret

Division:

mov rax,[opnd1]

mov rbx,[opnd2]

xor rdx,rdx ; clearing rdx

div rbx

mov r9,rax

rwm 01, q, m1

mov rax,r9

call conv\_and\_display

rwm 01, nwln, 01

ret

; Procedure for hex to ascii converion

conv\_and\_display:

mov rsi, ascii\_num+15

mov rcx, 16

again:

mov rdx,0

mov rbx,16h ; 16 in hex == 10 in decimal

;(quotient and rem will be stored in rax and rdx resp)

div rbx ; on divide rem will be last digit

cmp dl,09h

jbe add30

add dl,07h

add30:

add dl,30h

mov [rsi],dl

dec rsi

dec rcx

jnz again

rwm 01, ascii\_num, 16

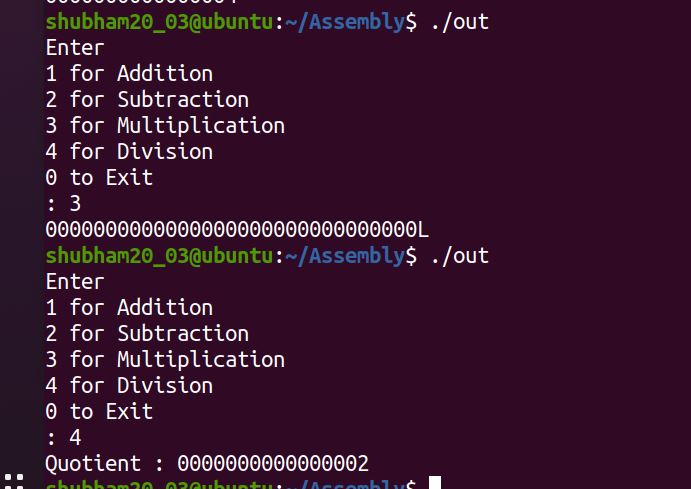
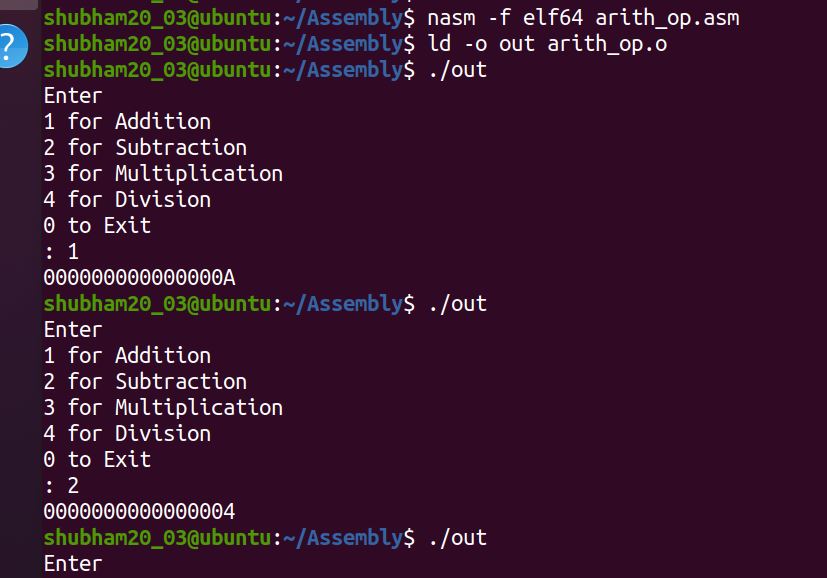
ret

mov rax,60

mov rdi,00

Syscall

Output:



Conclusion:

In this assignment I learned arithmetic operations on 64-bit numbers by using switch case, macro and procedure in assembly language and written the assembly program for the same.