**Title:- Arithmetic Operations.**

**Assignment Name:- Write an ALP using switch case to perform arithmetic operations.**

%macro IO 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .data

m1 db "enter choice (+,-,\*, /)" ,10 ; 10d -> line feed

l1 equ $-m1

m2 db "Write a switch case driven X86/64 ALP to perform 64-bit hexadecimal arithmetic operations (+,-,\*, /) using suitable macros. Define procedure for each operation." ,10

l2 equ $-m2

m3 db "rahul ghosh 3236" ,10

l3 equ $-m3

madd db "addition here" ,10

l4 equ $-madd

msub db "subtraction here" ,10

l5 equ $-msub

mmul db "multiplication here" ,10

l6 equ $-mmul

mdiv db "division here" ,10

l7 equ $-mdiv

mspace db 10

m\_result db "result is "

m\_result\_l equ $-m\_result

m\_qou db "qoutient is "

m\_qou\_l equ $-m\_qou

m\_rem db "remainder is "

m\_rem\_l equ $-m\_rem

m\_default db "enter correct choice",10

m\_default\_l equ $-m\_default

section .bss

choice resb 2

\_output resq 1

\_n1 resq 1

\_n2 resq 1

temp\_1 resq 1

temp\_2 resq 1

section .text

global \_start

\_start:

IO 1,1,m2,l2

IO 1,1,m3,l3

IO 1,1,m1,l1

IO 0,0,choice,2

cmp byte [choice],'+'

jne case2

call add\_fun

jmp exit

case2:

cmp byte [choice],'-'

jne case3

call sub\_fun

jmp exit

case3:

cmp byte [choice],'\*'

jne case4

call mul\_fun

jmp exit

case4:

cmp byte [choice],'/'

jne case5

call div\_fun

jmp exit

case5:

cmp byte [choice],'a'

jne error

call add\_fun

call sub\_fun

call mul\_fun

call div\_fun

jmp exit

error:

IO 1,1,m\_default,m\_default\_l

jmp exit

exit:

mov rax, 60

mov rdi, 0

syscall

add\_fun:

IO 1,1,madd,l4

mov qword[\_output],0

IO 0,0,\_n1,17

IO 1,1,\_n1,17

call ascii\_to\_hex

add qword[\_output],rbx

IO 0,0,\_n1,17

IO 1,1,\_n1,17

call ascii\_to\_hex

add qword[\_output],rbx

mov rbx,[\_output]

IO 1,1,mspace,1

IO 1,1,m\_result,m\_result\_l

call hex\_to\_ascii

ret

sub\_fun:

IO 1,1,msub,l5

mov qword[\_output],0

IO 0,0,\_n1,17

IO 1,1,\_n1,17

;IO 1,1,mspace,1

call ascii\_to\_hex

add qword[\_output],rbx

IO 0,0,\_n1,17

IO 1,1,\_n1,17

;IO 1,1,mspace,1

call ascii\_to\_hex

sub qword[\_output],rbx

mov rbx,[\_output]

IO 1,1,mspace,1

IO 1,1,m\_result,m\_result\_l

call hex\_to\_ascii

ret

mul\_fun:

IO 1,1,mmul,l6 ; message

IO 0,0,\_n1,17 ; n1 input

IO 1,1,\_n1,17

call ascii\_to\_hex; conversion returns hex value in rbx

mov [temp\_1],rbx ; storing hex in temp\_1

IO 0,0,\_n1,17 ;n2 input

IO 1,1,\_n1,17

call ascii\_to\_hex

mov [temp\_2],rbx ; putting hex of n2 in temp\_2

mov rax,[temp\_1] ; temp\_1->rax

mov rbx,[temp\_2] ;temp\_2->rbx

mul rbx ; multiplication

push rax

push rdx

IO 1,1,mspace,1

IO 1,1,m\_result,m\_result\_l

pop rdx

mov rbx,rdx; setting rbx value for conversion

call hex\_to\_ascii

pop rax

mov rbx,rax; setting rbx value for conversion

call hex\_to\_ascii ; final output

ret

div\_fun:

IO 1,1,mdiv,l7

IO 0,0,\_n1,17 ; n1 input

IO 1,1,\_n1,17

call ascii\_to\_hex; conversion returns hex value in rbx

mov [temp\_1],rbx ; storing hex in temp\_1

IO 0,0,\_n1,17 ;n2 input

IO 1,1,\_n1,17

call ascii\_to\_hex

mov [temp\_2],rbx ; putting hex of n2 in temp\_2

mov rax,[temp\_1] ; temp\_1->rax

mov rbx,[temp\_2] ;temp\_2->rbx

xor rdx,rdx

mov rax,[temp\_1] ; temp\_1->rax

mov rbx,[temp\_2] ; temp\_2->rbx

div rbx ; div

push rax

push rdx

IO 1,1,mspace,1

IO 1,1,m\_rem,m\_rem\_l

pop rdx

mov rbx,rdx

call hex\_to\_ascii; remainder output

IO 1,1,mspace,1

IO 1,1,m\_qou,m\_qou\_l

pop rax

mov rbx,rax

call hex\_to\_ascii; quotient output

ret

ascii\_to\_hex:

mov rsi, \_n1

mov rcx, 16

xor rbx, rbx

next1:

rol rbx, 4

mov al, [rsi]

cmp al,47h

jge error

cmp al, 39h

jbe sub30h

sub al, 7

sub30h:

sub al, 30h

add bl, al

inc rsi

loop next1

ret

hex\_to\_ascii:

mov rcx, 16

mov rsi,\_output

next2:

rol rbx, 4

mov al, bl

and al, 0Fh

cmp al, 9

jbe add30h

add al, 7

add30h:

add al, 30h

mov [rsi], al

inc rsi

loop next2

IO 1,1,\_output,16

IO 1,1,mspace,1

ret