Prgram No.4

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

```
%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
```

Output:

```
^C
student@student-Vostro-3902:~/Downloads/Ratnapal$ clear




student@student-Vostro-3902:~/Downloads/Ratnapal$ nasm -f elf64 mp4.asm
student@student-Vostro-3902:~/Downloads/Ratnapal$ ld -s -o mp4 mp4.o
student@student-Vostro-3902:~/Downloads/Ratnapal$ ./mp4
Write a switch case driven X86/64 ALP to perform 64-bit hexadecimal arithmetic operations (+,-,*, /) using suitable macros. Define procedure fo
r each operation.
enter choice (+,-,*, /)
+
addition here
2
2
5
5

result is 1FBBBBBBBBBBBA0
student@student-Vostro-3902:~/Downloads/Ratnapal$ ./mp4
Write a switch case driven X86/64 ALP to perform 64-bit hexadecimal arithmetic operations (+,-,*, /) using suitable macros. Define procedure fo
r each operation.
enter choice (+,-,*, /)
*
multiplication here
3
3
6
6
◆
result is 01B77C048D159E23
3B72EA61D950C900
student@student-Vostro-3902:~/Downloads/Ratnapal$
```