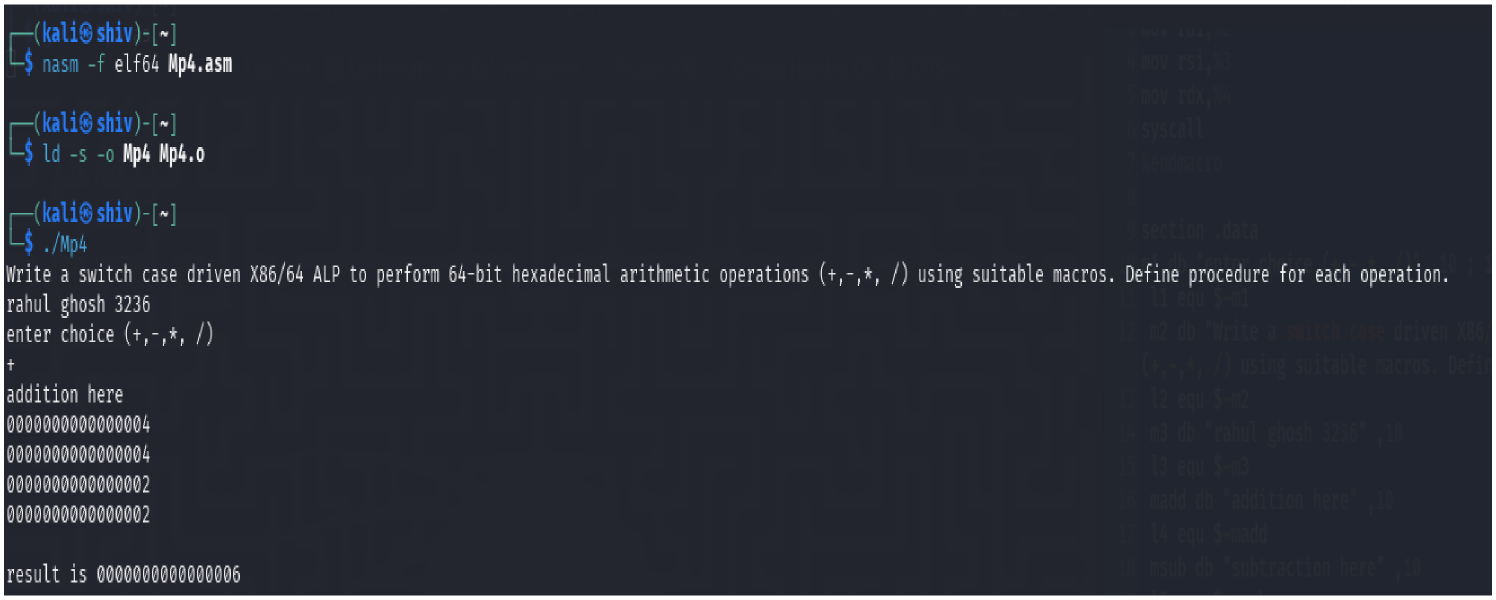
```
%macro IO 4
    mov rax,%1
    mov rdi,%2
    mov rsi,%3
    mov rdx,%4
    syscall
%endmacro

section .data
    m1 db "enter choice (+,-,*, /)" ,10         ; 10d -� line feed
    l1 equ $-m1
    m2 db "Write a switch case driven X8G/G4 ALP to perform G4-bit hexadecimal arithmetic operations (+,-,*, /) using suitable macros. Define procedure for each operation." ,10
    l2 equ $-m2
    m3 db "rahul ghosh 323G" ,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
    lG 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
    call ascii_to_hex
    add qword[_output],rbx
    IO 0,0,_n1,17
    IO 1,1,_n1,17
    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,lG
    IO 0,0,_n1,17
    IO 1,1,_n1,17
    call ascii_to_hex
    mov [temp_1],rbx
    IO 0,0,_n1,17
    IO 1,1,_n1,17
    call ascii_to_hex
    mov [temp_2],rbx
    mov rax,[temp_1]
    mov rbx,[temp_2]
    mul rbx
    push rax
    push rdx
    IO 1,1,mspace,1
    IO 1,1,m_result,m_result_l
    pop rdx
    mov rbx,rdx
    call hex_to_ascii
    pop rax
    mov rbx,rax
    call hex_to_ascii
    ret

```

```
div_fun:
    IO 1,1,mdiv,l7
    IO 0,0,_n1,17
    IO 1,1,_n1,17
    call ascii_to_hex
    mov [temp_1],rbx
    IO 0,0,_n1,17
    IO 1,1,_n1,17
    call ascii_to_hex
    mov [temp_2],rbx
    mov rax,[temp_1]
    mov rbx,[temp_2]
    xor rdx,rdx
    mov rax,[temp_1]
    mov rbx,[temp_2]
    div rbx
    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
    IO 1,1,mspace,1
    IO 1,1,m_qou,m_qou_l
    pop rax
    mov rbx,rax
    call hex_to_ascii
    ret

ascii_to_hex:
    mov rsi, _n1
    mov rcx, 1G
    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, 1G
    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,1G
    IO 1,1,mspace,1
    ret
```

Output :



```
┌──(kali㉿shiv)-[~]
└─$ nasm -f elf64 Mp4.asm

┌──(kali㉿shiv)-[~]
└─$ ld -s -o Mp4 Mp4.o

┌──(kali㉿shiv)-[~]
└─$ ./Mp4
Write a switch case driven X86/64 ALP to perform 64-bit hexadecimal arithmetic operations (+,-,*, /) using suitable macros. Define procedure for each operation.
rahul ghosh 3236
enter choice (+,-,*, /)
+
addition here
0000000000000004
0000000000000004
0000000000000002
0000000000000002

result is 0000000000000006
```