Practical – 4

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

Program:

```
%macro scall 4
 mov rax,%1
 mov rdi,%2
 mov rsi,%3
 mov rdx,%4
 syscall
%endmacro
;------------------------.data section-----------------------------
section .data
 arr dq 000000000000003h,0000000000000002h
 n equ 2
 menu db 10d,13d,"**********MENU**********"
 db 10d,13d,"1. Addition"
 db 10d,13d,"2. Subtraction"
 db 10d,13d,"3. Multiplication"
 db 10d,13d,"4. Division"
 db 10d,13d,"5. Exit"
 db 10d,13d,"Enter your Choice: "
 menu_len equ $-menu
 m1 db 10d,13d,"Addition: "
 l1 equ $-m1
```

```
 m2 db 10d,13d,"Substraction: "
 l2 equ $-m2
 m3 db 10d,13d,"Multiplication: "
 l3 equ $-m3
 m4 db 10d,13d,"Division: "
 l4 equ $-m4

;------------------------.bss section-----------------------------
section .bss
 answer resb 16 ;to store the result of operation
 choice resb 2

;------------------------.text section ----------------------------
section .text
 global _start:
 _start:
 up: scall 1,1,menu,menu_len
 scall 0,0,choice,2

 cmp byte[choice],'1'
 je case1
 cmp byte[choice],'2'
 je case2
 cmp byte[choice],'3'
 je case3
 cmp byte[choice],'4'
```

```
je case4
cmp byte[choice],'5'
je case5

case1: scall 1,1,m1,l1
call addition
jmp up

case2: scall 1,1,m2,l2
call substraction
jmp up
case3: scall 1,1,m3,l3
call multiplication
jmp up
case4: scall 1,1,m4,l4
call division
jmp up

case5: mov rax,60
mov rdi,0
syscall

;procedures for arithmetic and logical operations
addition:
mov rcx,n
dec rcx
```

```
 mov rsi,arr
 mov rax,[rsi]
up1: add rsi,8
 mov rbx,[rsi]
 add rax,rbx
 loop up1
 call display
ret

substraction:

 mov rcx,n
 dec rcx
 mov rsi,arr
 mov rax,[rsi]
up2: add rsi,8
 mov rbx,[rsi]
 sub rax,rbx
 loop up2
 call display
ret

multiplication:
 mov rcx,n
 dec rcx
 mov rsi,arr
```

```
 mov rax,[rsi]
up3: add rsi,8
 mov rbx,[rsi]
 mul rbx
 loop up3
 call display
ret
division:
 mov rcx,n
 dec rcx

 mov rsi,arr
 mov rax,[rsi]
up4: add rsi,8
 mov rbx,[rsi]
 mov rdx,0
 div rbx
 loop up4
 call display
ret
or:
 mov rcx,n
 dec rcx
 mov rsi,arr
 mov rax,[rsi]
up6: add rsi,8
```

```
 mov rbx,[rsi]
 or rax,rbx
 loop up6
 call display
ret
xor:
 mov rcx,n
 dec rcx
 mov rsi,arr
 mov rax,[rsi]
up7: add rsi,8
 mov rbx,[rsi]
 xor rax,rbx
 loop up7
 call display
ret
and:
 mov rcx,n
 dec rcx
 mov rsi,arr
 mov rax,[rsi]
up8: add rsi,8
 mov rbx,[rsi]
 and rax,rbx
 loop up8
 call display
```

```
ret
display:
 mov rsi,answer+15
 mov rcx,16
cnt: mov rdx,0
 mov rbx,16
 div rbx
 cmp dl,09h
 jbe add30
 add dl,07h
add30: add dl,30h
 mov [rsi],dl
 dec rsi
 dec rcx
 jnz cnt
 scall 1,1,answer,16
ret
```

Output:

```
atharva@atharva:~$ gedit lab4.asm
atharva@atharva:~$ nasm -f elf64 lab4.asm
atharva@atharva:~$ ld -o lab4 lab4.o
atharva@atharva:~$ ./lab4
**********MENU**********
1. Addition
2. Subtraction
3. Multiplication
```

4. Division

5. Exit

Enter your Choice: 1

Addition: 0000000000000005

**********MENU**********

1. Addition

2. Subtraction

3. Multiplication

4. Division

5. Exit

Enter your Choice: 2

Substraction: 0000000000000001

**********MENU**********

1. Addition

2. Subtraction

3. Multiplication

4. Division

5. Exit

Enter your Choice: 3

Multiplication: 0000000000000006

**********MENU**********

1. Addition

2. Subtraction

3. Multiplication

4. Division

5. Exit

Enter your Choice: 4

Division: 0000000000000001

**********MENU**********

1. Addition

2. Subtraction

3. Multiplication

4. Division

5. Exit

Enter your Choice: 5