Assignment no.4

Name:- Gaurav Patil

Roll No. - 21259

Batch – G2

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* C O D E \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%macro scall 4

mov eax,%1

mov ebx,%2

mov ecx,%3

mov edx,%4

int 80h

%endmacro

section .data

arr dq 0000003h,00000003h

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

section .bss

answer resb 8 ;to store the result of operation

choice resb 2

section .text

global \_start:

\_start:

up: scall 4,1,menu,menu\_len

scall 3,0,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 4,1,m1,l1

call addition

jmp up

case2: scall 4,1,m2,l2

call substraction

jmp up

case3: scall 4,1,m3,l3

call multiplication

jmp up

case4: scall 4,1,m4,l4

call division

jmp up

case5: mov eax,1

mov ebx,0

int 80h

;procedures for arithmetic and logical operations

addition:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up1: add esi,8

mov ebx,[esi]

add eax,ebx

loop up1

call display

ret

substraction:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up2: add esi,8

mov ebx,[esi]

sub eax,ebx

loop up2

call display

ret

multiplication:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up3: add esi,8

mov ebx,[esi]

mul ebx

loop up3

call display

ret

division:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up4: add esi,8

mov ebx,[esi]

mov edx,0

div ebx

loop up4

call display

ret

or:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up6: add esi,8

mov ebx,[esi]

or eax,ebx

loop up6

call display

ret

xor:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up7: add esi,8

mov ebx,[esi]

xor eax,ebx

loop up7

call display

ret

and:

mov ecx,n

dec ecx

mov esi,arr

mov eax,[esi]

up8: add esi,8

mov ebx,[esi]

and eax,ebx

loop up8

call display

ret

display:

mov esi,answer+7

mov ecx,8

cnt: mov edx,0

mov ebx,16

div ebx

cmp dl,09h

jbe add30

add dl,07h

add30: add dl,30h

mov [esi],dl

dec esi

dec ecx

jnz cnt

scall 4,1,answer,8

ret

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* O U T P U T \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

