LAB 9

Questions:

Task#1

Write ASM instructions that calculate EAX * 21 using binary multiplication.

Hint: 21 = 2^4+2^ + 2^0

Task#2

Give an assembly language program to move -128 in ax and expand eax. Using shift and rotate instruction.

Task#3

Write a series of instructions that shift the lowest bit of AX into the highest bit of BX without using the SHRD instruction. Next, perform the same operation using SHRD.

Task#4

Implement the following C++ expression in assembly language, using 32-bit signed operands:

val1 = (val2 / val3) * (val1 / val2);

Task#5

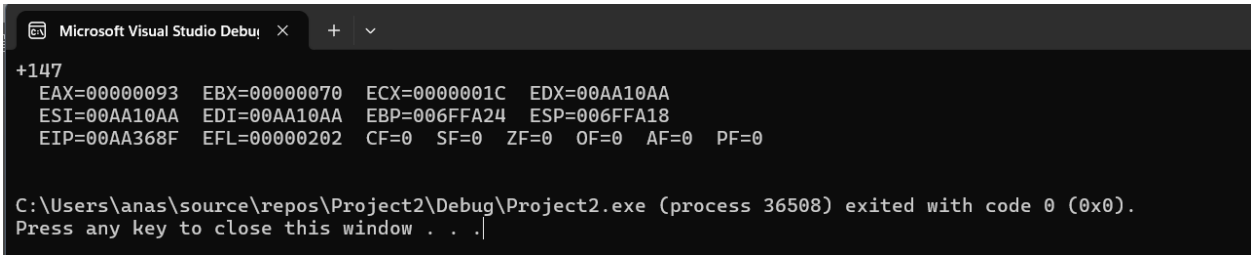
Create a procedure Extended_Add procedure to add two 64-bit (8-byte) integers.

Q1

Code:

```
INCLUDE Irvine32.inc

.data
var DWORD 7 ; value to store in eax

.code
main PROC
mov eax , 0
mov eax , var

; eax * 21
shl eax , 4 ; eax = eax * 2^4
mov ebx , eax
mov eax , var
shl eax , 2 ; eax = eax * 2^2
mov ecx , eax
mov eax , var
shl eax , 0
```

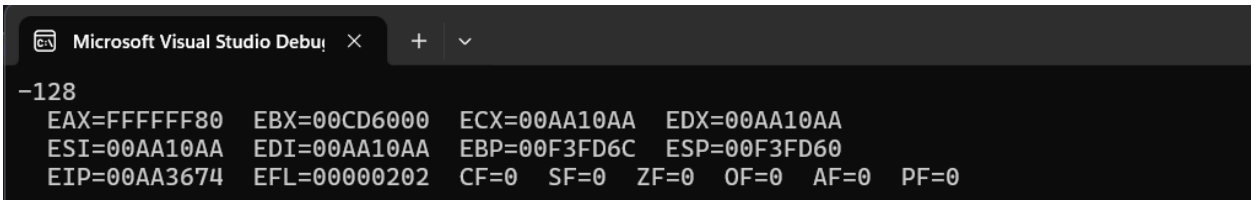
```
add eax , ecx
add eax , ebx

call Writeint

call Dumpregs
exit
main ENDP
END main
```



Q2

Code:

```
INCLUDE Irvine32.inc

.data


.code
main PROC

mov ax , -128 ; high half of eax is filled with zeros
shl eax , 16 ; shift the FF80h(-128) 16 bits to the left filling 16bits rights
filled with zero
sar eax , 16 ; performs airthmetic shift to the right while conserving
sign(ffffff80h)


call Writeint

call Dumpregs
exit
main ENDP
END main
```



Q3

Code:

```
INCLUDE Irvine32.inc

.data


.code
main PROC
mov ax , 12

and ax , 1 ; isolating lowest bit
shl ax , 15 ; loswest bit move to highest bit of ax

mov bx , 0
or bx , ax ; bx highest bit is set according to ax's lowest bit

call Dumpregs
exit
main ENDP
END main
```

USING SHRD INSTRUCTION

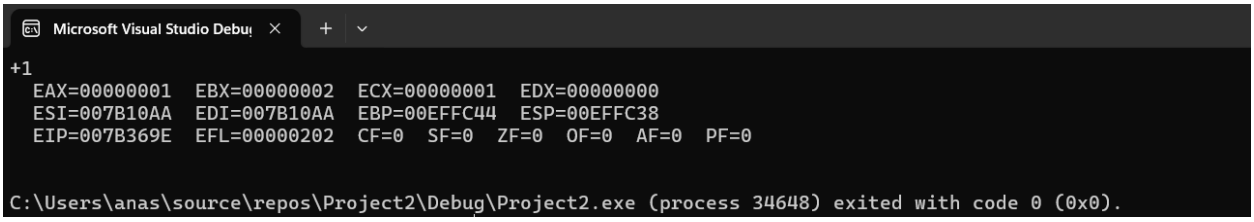
```
shrd ax , bx , 1
```

Q4

Code:

```
INCLUDE Irvine32.inc

.data
val1 DWORD 2 ; can be changed later
val2 DWORD 2 ; can be changed later
val3 DWORD 2 ; can be changed later

.code
main PROC

mov eax , 0

mov eax ,  val2
cdq ; Sign-extend EAX into EDX for division to preserve sign
mov ebx , val3
idiv ebx ; eax = val2/val3

mov ecx , eax ; to store result of val2/val3 in ecx

mov eax , 0
mov ebx , 0
mov eax , val1
cdq
mov ebx , val2
idiv ebx ; eax = val1/val2
```

```
imul ecx ; eax = (val2/val3) * (val1/val2)

mov val1 , eax


call Dumpregs
exit
main ENDP
END main
```



Q5

Code:

```
INCLUDE Irvine32.inc

.data
low1 DWORD ? ; Lower 32 bits of the first integer
high1 DWORD ? ; Upper 32 bits of the first integer
low2 DWORD ? ; Lower 32 bits of the second interger
high2 DWORD ? ; Upper 32 bits of the second integer

.code
main PROC
mov eax , 0
call Extended_Add
main ENDP
exit

Extended_Add PROC
mov eax , low1
mov ebx , low2
add eax , ebx

;now add the carry(if any) from the lower bits of integer to higher bits
mov eax , high1
mov ebx , high2
adc eax , ebx ; eax = high1 + high2 + carry

call Writeint

ret
Extended_Add ENDP


call Dumpregs
exit
main ENDP
END main
```

Assigned some values to low1,high1,low2,high2 for output