**COAL LAB 12**

Q.1)

.asm

.686

.MODEL FLAT, C

.STACK 2048

.DATA

.CODE

clear PROC

xor eax, eax

xor ebx, ebx

ret

clear ENDP

END

.cpp

#include <stdio.h>

extern "C" void clear();

int threeprod(int , int , int);

int main()

{

clear();

unsigned int num1, num2, num3, result;

printf("Enter 3 integers: \n");

scanf\_s("%d %d %d", &num1, &num2, &num3);

result=threeprod(num1,num2, num3);

printf("\nResult = %d\n ",result);

return 0;

}

int threeprod(int num1, int num2, int num3) {

int result;

\_asm

{

MOV EAX, num1

MOV EBX, num2

MUL EBX

MOV EBX, num3

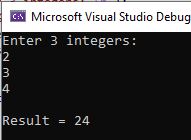
MUL EBX

MOV result, EAX

}

return result;

}



Q.2)

.asm

.686

.MODEL FLAT, C

.STACK 2048

.DATA

.CODE

clear PROC

xor eax, eax

xor ebx, ebx

ret

clear ENDP

GCD PROC uses ebx ,a:dword,b:dword

mov eax,a

mov ebx,b

cmp eax,0

jne cont

mov eax,ebx

ret

cont:

cmp ebx,0

jne contin

ret

contin:

cmp eax,ebx

jne continue

ret

continue:

cmp eax,ebx

jle recursion

sub eax,ebx

INVOKE GCD,eax,ebx

ret

recursion:

sub ebx,eax

INVOKE GCD,eax,ebx

ret

GCD endp

END

.cpp

#include <stdio.h>

extern "C" void clear();

extern "C" int GCD(int,int);

int gcd(int, int);

int main()

{

clear();

unsigned int num1, num2, result;

printf("Enter 2 integers: \n");

scanf\_s("%d%d", &num1, &num2);

result = gcd(num1, num2);

printf("\nGCD = %d\n ", result);

return 0;

}

int gcd(int num1, int num2)

{

int result;

return result=GCD(num1,num2);

}

