COAL LAB#12-Inline Assignment

Name: Anusha Saad

ID: 19K-0281

Section: H

Task#1:

.686 .MODEL FLAT, C

.STACK 2048

.code

main PROC

ENTER 0,0

mov eax, 0

mov ecx, 0

mov val1, eax

mov ecx, eax

mov val2, eax

INVOKE CalcGCD, val1, val2

mov eax, GCD

add esp, 4

leave

ret

exit

main ENDP

CalcGCD PROC, X:DWORD, Y:DWORD

push ebp

mov ebp, esp

mov edx,0

mov eax, val1

div ecx

cmp edx,0

jne L1

mov edx,0

mov eax, val2

div ecx

cmp edx,0

je L2

L1: dec ecx

INVOKE CalcGCD, X, Y

L2: mov GCD, ecx

pop ebp

add esp,4

ret

CalcGCD endp

.cpp code:

#include "stdafx.h"

#include "pch.h"

#include <iostream>

using namespace std;

extern "C" void clear();

extern "C" int GCD(int val1, int val2);

int main()

{

clear();

int n1, n2, result;

cout << "Enter two numbers for GCD: ";

cin >> n1;

cin >> n2;

result = GCD(n1, n2);

cout << "GCD of (" << n1 << "," << n2 << "): " << result << endl;

return 0;

}

TASK 2:

.686

.MODEL FLAT, C

.STACK 2048

.data

mul PROTO

ThreeAdd PROTO, temp1:SDWORD, temp2:SDWORD, temp3:SDWORD

.code

mul PROC

xor eax, eax

xor ecx, ecx

ret

mul endp

ThreeAdd PROC, n1:DWORD, n2:DWORD, n3:DWORD

mov eax, n1

mov ecx, n2

add eax,ecx

add eax, n3

ret

ThreeAdd endp

End

.cpp

#include "stdafx.h"

extern "C" void mul();

extern "C" int Add\_Three(int val1, int val2, int val3);

int main()

{

clear();

int n1, n2, n3, result;

cout << "Enter 3 no: ";

cin >> 1;

cin >> n2;

cin >> n3;

res = ThreeAdd(n1, n2, n3);

cout << result << endl;

return 0;

}

TASK 3:

.686

.MODEL FLAT, C

.STACK 2048

.data

Mul PROTO

.code

clear PROC

xor eax, eax

xor ecx, ecx

ret

mul endp

.cpp code

#include "stdafx.h"

extern "C" void mul();

int main()

{

clear();

int n1, n2, result;

cout << "Enter 3 nos: ";

cin >> n1;

cin >> n2;

\_asm

{

mov eax, n1

mov ecx, n2

imul ecx

mov res, eax

}

cout<< result << endl;

return 0;

}

TASK 4:

.cpp code

int main()

{

max();

int min, max, a[10];

cout << "Enter 10 nos: " << endl;

for (int i = 0; i < 10; i++)

{

cin >>a[i];

}

MM(a);

\_asm

{

mov min, eax

mov max, ebx

}

cout << min << endl << max << endl;

return 0;

}