CS 410 C++ to Assembly Activity Template

Step 1: Explain the functionality of the C++ code.

C++ Code Functionality

C++ Line of Code	Explanation of Functionality
#include<iostream>	This line includes the <iostream> header file, which allows input and output operations in C++.
using namespace std;	This line declares that the std namespace will be used. The std namespace contains many standard C++ functions and objects.
int main() {	This line marks the beginning of the main() function, which is the entry point of a C++ program. The execution of the program starts from here.
int width = 10;	This line declares and intalizes an integer variable named “width” with a value of 10
int height = 5;]	This line declares and intalizes an integer variable named “height” with a value of 5
int area;	This line declares an integer variable named area without initializing it.
area = width * height;	This line calculates the product of width and height and assigns the result to the area variable.
cout << endl << area;	This line prints the value of area to the standard output (console). The cout object is used for output operations. The << operator is used to chain multiple outputs together. endl is used to insert a newline character after printing area.
return 0;	This line ends the main() function and returns 0 to the operating system, indicating successful program execution.

Step 2: Convert the C++ file into assembly code.

Step 3: Align each line of C++ code with the corresponding blocks of assembly code.

C++ to Assembly Alignment

C++ Line of Code	Blocks of Assembly Code

int main(){
}
	pushq %rbp #
	movq %rsp, %rbp #,
	subq $16, %rsp #,

int width=10;	movl $10, -12(%rbp) #, width

int height=5;	movl $5, -8(%rbp) #, height

area = width * height;	movl -12(%rbp), %eax # width, tmp91
	imull -8(%rbp), %eax # height, tmp90
	movl %eax, -4(%rbp) # tmp90, area

cout<<endl<< area;	Movq _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL(%rip), %rax #, tmp92
	movq %rax, %rsi # tmp92,
	leaq _ZSt4cout(%rip), %rdi #,
	call _ZNSolsEPFRSoS_E@PLT #
	movq %rax, %rdx #, _1
	movl -4(%rbp), %eax # area, tmp93
	movl %eax, %esi # tmp93,
	movq %rdx, %rdi # _1,
	call _ZNSolsEi@PLT #

return 0;	movl $0, %eax #, _9

Step 4: Explain how the blocks of assembly code perform the same tasks as the C++ code.

Assembly Functionality

Blocks of Assembly Code	Explanation of Functionality

pushq %rbp #
movq %rsp, %rbp #,
subq $16, %rsp #,
	pushq %rbp instruction pushes the value of the %rbp register onto the stack. This is commonly done at the beginning of a function to save the previous value of the base pointer register.
	
	movq %rsp, %rbp instruction copies the value of the stack pointer (%rsp) into the base pointer register (%rbp). It establishes the stack frame for the current function.
	
	subq $16, %rsp instruction subtracts 16 bytes from the stack pointer (%rsp). It allocates space on the stack for local variables or temporary storage.

movl $10, -12(%rbp) #, width	moves the immediate value 10 into the memory location specified by -12(%rbp). This instruction initializes a variable named width with the value of 10.
movl $5, -8(%rbp) #, height	moves the immediate value 5 into the memory location specified by -8(%rbp). This instruction initializes an integer variable named height with the value of 5.

movl -12(%rbp), %eax # width, tmp91
imull -8(%rbp), %eax # height, tmp90
movl %eax, -4(%rbp) # tmp90, area
	movl -12(%rbp), %eax moves the value from the memory location specified by -12(%rbp) into the %eax register. This instruction retrieves the value of the width variable.
	
	imull -8(%rbp), %eax multiplies the value from the memory location specified by -8(%rbp) (value of the height variable) with the value in the %eax register (value of the width variable). The result is stored in %eax.
	
	movl %eax, -4(%rbp) moves the value in the %eax register (the computed area) into the memory location specified by -4(%rbp). This instruction stores the result in a variable named area.

Movq _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL(%rip), %rax #, tmp92
movq %rax, %rsi # tmp92,
leaq _ZSt4cout(%rip), %rdi #,
call _ZNSolsEPFRSoS_E@PLT #
movq %rax, %rdx #, _1
movl -4(%rbp), %eax # area, tmp93
movl %eax, %esi # tmp93,
movq %rdx, %rdi # _1,
call _ZNSolsEi@PLT #
	Movq _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL(%rip), %rax moves the address of the endl function into the %rax register. The endl function is used for printing a newline character.
	
	movq %rax, %rsi moves the value in %rax (address of endl) into the %rsi register.
	
	leaq _ZSt4cout(%rip), %rdi loads the address of the cout object into the %rdi register. cout is the standard output stream used for printing.
	
	call _ZNSolsEPFRSoS_E@PLT calls the << operator function to print the endl to the standard output stream (cout).
	
	movq %rax, %rdx moves the value returned by the previous function call into the %rdx register.
	
	movl -4(%rbp), %eax moves the value of the area variable (stored at -4(%rbp)) into the %eax register.
	
	movl %eax, %esi moves the value in %eax (the area variable) into the %esi register.
	
	movq %rdx, %rdi moves the value in %rdx (address of endl) into the %rdi register.
	
	call _ZNSolsEi@PLT calls the << operator function to print the value of area to the standard output stream (cout).

movl $0, %eax	moves the immediate value 0 into the %eax register. The %eax register is commonly used to store the return value of a function.