# **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> | Include the main header file for input/output operations |
| using namespace std; | Tell the compilers to use standard namespace so we don’t need to add namespace for each input/output function |
| int main() | The entry point of the program |
| { | Opening braces for the main function |
| int width=10; | Declare and assign 10 to width variable |
| int height=5; | Declare and assign 5 to height |
| int area; | Declare area variable (unsigned) |
| area = width \* height; | Assign the multiplication of width and height to area variable |
| cout<<endl<< area; | Print empty line and area |
| return 0; | Return 0 to the caller |
| } | Close braces for the main program |

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

**movl $10, -12(%rbp)**

**movl $5, -8(%rbp)**

**movl -12(%rbp), %eax**

**imull -8(%rbp), %eax**

**movl %eax, -4(%rbp)**

**call endl**

**movl %eax, %esi**

**call cout**

**movl $0, %eax**

**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** |
| **#include<iostream>** |  |
| **using namespace std;** |  |
| **int main()** |  |
| **{** |  |
| **int width=10;** | **movl $10, -12(%rbp)** |
| **int height=5;** | **movl $5, -8(%rbp)** |
| **int area;** |  |
| **area = width \* height;** | **movl -12(%rbp), %eax**  **imull -8(%rbp), %eax**  **movl %eax, -4(%rbp)** |
| **cout<<endl<< area;** | **call endl**  **movl %eax, %esi**  **call cout** |
| **return 0;** | **movl $0, %eax** |
| **}** |  |

**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** |
| **movl $10, -12(%rbp)** | Push the 10 value 12 bytes above %rbp (which is general purpose register) and starts with r because it’s 64bit architecture. |
| **movl $5, -8(%rbp)** | Push the 8 value 8 bytes above the base stack %rbp |
| **movl -12(%rbp), %eax** | Move 10 from register rbp to eax register |
| **imull -8(%rbp), %eax** | Multiple 5 (-8 %rbp) by register eax and store the result in eax |
| **movl %eax, -4(%rbp)** | Move eax value to register -4 rbp which is area variable |
| **call endl** | Call endl function from iostream lib address |
| **movl %eax, %esi** | Move the value of eax register to esi in order to print the area |
| **call cout** | Print out the value in esi |
| **movl $0, %eax** | Return 0 to main caller |
|  |  |