# **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> | Adds iostream library functionality |
| using namespace std; | Sets the namespace to standard library |
| int main() | Main function for application |
| { | Start of Program with curly bracket |
| int width=10; | Declares variable width to integer 10 |
| int height=5; | Declares variable height to integer 5 |
| int area; | Declares variable area |
| area = width \* height; | Declares area to width \* height |
| cout<<endl<< area; | Prints a return line and the variable area |
| return 0; | Returns 0 to end the application and the function is terminated |
| } | Exit |

**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() | .globl main  .type main, @function  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; | subq $16, %rsp  movq ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0ES6\_@GOTPCREL(%rip), %rax  movq %rax, %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZNSolsEPFRSoS\_E@PLT  movq %rax, %rdx  movl -4(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT |
| Return 0; | movl  $0, %eax  leave |

**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** |
| --- | --- |
| .globl main  .type main, @function  main: | Globl: informs program to add the main label.  Type: Informs program that main is executable code.  Main: Declares the label main as entry point |
| movl $10, -12(%rbp) | Declared value: 10 is moved 12 bytes above the starting point |
| movl $5, -8(%rbp) | Declared value: 5 is moved 8 bytes above the starting point |
| movl -12(%rbp), %eax  imull -8(%rbp), %eax  movl %eax, -4(%rbp) | Above values: 10 and 5 taken from the %rbp and moved to eax register. They are multiplied then stored. Stored value is moved to %rbp and stored 4 bytes from the starting point. |
| subq $16, %rsp  movq \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax  movq %rax, %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZNSolsEPFRSoS\_E@PLT  movq %rax, %rdx  movl -4(%rbp), %eax  movl %eax, %esi  movq %rdx, %rdi  call \_ZNSolsEi@PLT | Subq assigns space on the stack. %rsp points to top of stack  Movq stored value is moved from %rip to %rax, %rax is moved to %rsi.  Leaq moves address from %rip to %rsi.  Cout(call) function is then made  More values moved and results are printed |
| movl  $0, %eax  leave | Code Done  Terminate Program |