# **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 |
| { |  |
| int width=10; | Sets variable width to integer 10 |
| int height=5; | Sets variable height to integer 5 |
| int area; | Declares variable area |
| area = width \* height; | Sets area to width \* height |
| cout<<endl<< area; | Prints a return line and the variable area |
| return 0; | Returns 0 to end the application |
| } |  |

**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\_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  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: | The global directive tells the assembler to add the main label. The type directive tells the assembler that main is executable code. Main: is the entry point for the function. |
| movl $10, -12(%rbp) | Value 10 is moved 4 bytes above the starting point on the register |
| movl $5, -8(%rbp) | Value 5 is moved 8 bytes above the starting point on the register |
| movl -12(%rbp), %eax  imull -8(%rbp), %eax  movl %eax, -4(%rbp) | The two earlier values are taken from the rbp register and moved to eax where they are multiplied and the value is stored. That value is then moved to register rbp and stored 12 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  leave | Subq allocates space on the stack. Movq then moves the what is stored at %rip to %rax. %rax is also moved to %rsi. Leaq then moves the address at to %rip to %rsi. A call to a function is then made (either cout or endl). More movement is done and then a second function call that is also cout or endl depending on what the first was. The function main function then exits. |