# **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> | Includes the iostream header file for inputs and outputs |
| using namespace std; | Used to set the namespace to std. can use cout instead of std::cout |
| int main() | Defines the main function, no arguments and returns an int |
| { | Beginning of function body |
| int width=10; | Declares integer variable width with value of 10 |
| int height=5; | Declares integer variable height with value of 5 |
| int area; | Declares integer variable area |
| area = width \* height; | Assigns area to width \* height |
| cout<<endl<< area; | Outputs an end line and then the area to the console |
| return 0; | Returns a 0 essentially ending the function |
| } | End of function body |

**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** |
| --- | --- |
| #include<iostream> | Gets processed by the preprocessor |
| using namespace std; | Used in compilation but doesn’t have assembly code |
| int main(){ | .text    .globl  main    .type main, @function  main:  .LFB1493:    .cfi\_startproc    pushq %rbp    .cfi\_def\_cfa\_offset 16    .cfi\_offset 6, -16    movq  %rsp, %rbp    .cfi\_def\_cfa\_register 6    subq  $16, %rsp |
|  |  |
| 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; | 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 |
| return 0; } | movl  $0, %eax    leave    .cfi\_def\_cfa 7, 8    ret    .cfi\_endproc |
|  |  |

**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** |
| --- | --- |
| .text    .globl  main    .type main, @function  main:  .LFB1493:    .cfi\_startproc    pushq %rbp    .cfi\_def\_cfa\_offset 16    .cfi\_offset 6, -16    movq  %rsp, %rbp    .cfi\_def\_cfa\_register 6    subq  $16, %rsp | .text is the beginning of the code section  Main is a global symbol  Main is a function  Start of main function  I don’t know exactly what all this does beyond starting the function call and setting everything up  The last line allocates space for the local variables of the function. This is why we don’t have a line of assembly for <int area;> since that is declared here. |
| movl  $10, -12(%rbp) | Moves the value of 10 to an offset of 12 from the base pointer. |
| movl  $5, -8(%rbp) | Moves the value of 5 to an offset of 8 from the base pointer. |
| movl  -12(%rbp), %eax    imull -8(%rbp), %eax    movl  %eax, -4(%rbp) | Moves the value of offset of 12 from base pointer (10) to eax register.  Multiplies the value of offset 8 from base pointer (5) by the eax register (10) and stores in eax register.  Moves the value of eax register (50) into offset of 4 from base pointer. |
| 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 | The beginning of this is for cout and endl. I don’t know exactly what it is doing.  This is where our previous answer is called. It moves the offset of 4 in base pointer (50) into eax register.  It gets attached to the outout stream but I’m not sure exactly how. |
| movl  $0, %eax    leave    .cfi\_def\_cfa 7, 8    ret    .cfi\_endproc | Sets the eax register to 0  Not sure exactly what the rest does except it ends the function and returns 0 |
|  |  |
|  |  |
|  |  |
|  |  |