# **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 will include the input and output stream |
| using namespace std; | Can omit the namespace qualifier because it is brought into programs scope |
| int main() | Main method |
| { | Opening bracket for the main method |
| int width=10; | Variable width with int type has a value of 10 |
| int height=5; | Variable height with int type has a value of 5 |
| int area; | Variable area is declare as an int type |
| area = width \* height; | Calculates the width and height variables and sets it to the area variable |
| cout<<endl<<area; | Print out a newline that is followed by the new values stored in variable area |
| return 0; | This return the value of zero which means success |
| } | Closes the main method |

**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>   using namespace std; | .text  .globl main  .type main, @function |
| int main()  { | 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; | movl  %eax, -4(%rbp)    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    movl  $0, %eax    leave |
| return 0; | ret    .cfi\_endproc  .LFE1980:    .size \_GLOBAL\_\_sub\_I\_main, .-\_GLOBAL\_\_sub\_I\_main    .section  .init\_array,"aw"    .align 8    .quad \_GLOBAL\_\_sub\_I\_main    .hidden \_\_dso\_handle    .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"    .section  .note.GNU-stack,"",@progbits |
| } |  |
|  |  |
|  |  |
|  |  |

**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 | Text section will store executable code  Globl main is where the assembler tells to add the label to the lists of labels  .type main,@function lets you know the label main is executable code |
| 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 | Defines the main function  Label that refers to the start of the function  .cfi\_startproc is the call frame information directive  pushq %rbp is the standard part of a function introduction in assembly code and saves the rbp register.  .cfi\_def\_cfa\_offset 16 and .cfi\_offset 6, -16 is Dwarf debugging information.  movq %rsp %rbp sets rbp to rsp. |
| movl $10, -12(%rbp) | The value 10 was moved 12 bytes above register %rbp |
| movl $5, -8(%rbp) | The value 5 was moved 8 bytes above register %rbp |
| movl -12(%rbp), %eax    imull -8(%rbp), %eax    movl  %eax, -4(%rbp) | Moved 12 bytes in register to eax  8 bytes in registe imull  Multiply rbp and eax and place result in rbp |
|  |  |
| ret   leave | These two is same as return 0 |
|  |  |
|  |  |
|  |  |