# **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 input/output stream library for input/output functionality |
| using namespace std; | Makes use of the standard namespace |
| int main() | Declares a function named main, the entry point of the program. |
| { | An opening brace to indicate where the logic inside the main function begins |
| int width=10; | Declaration of the variable “width” of data type integer initialized to 10. |
| int height=5; | Declaration of the variable “height” of data type integer initialized to 5 |
| int area; | Declaration of variable area of data type integer |
| area = width \* height; | Assigns variable area equal to the variable width multiplied by the variable height |
| cout<<endl<< area; | Outputs to the console a new line followed by the variable area |
| return 0; | Returns 0 because the main function is of type integer and expects an integer value to be returned. |
| } | A closing brace to indicate where the logic inside the main function ends. |

**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> | .type \_ZStL19piecewise\_construct, @object  .size \_ZStL19piecewise\_construct, 1  \_ZStL19piecewise\_construct:    .zero 1    .local  \_ZStL8\_\_ioinit    .comm \_ZStL8\_\_ioinit,1,1 |
| using namespace std; | none |
| 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; | movl  -12(%rbp), %eax |
| area = width \* height; | imull -8(%rbp), %eax  movl %eax, -4(%rbp) |
| cout <<endl<< area; | movq  \_ZSt4endlIcSt11char\_traitsIcEERSt13basi  c\_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 | ret |
| } |  |

**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** |
| --- | --- |
| 1. .type \_ZStL19piecewise\_construct, @object 2. .size \_ZStL19piecewise\_construct, 1 3. \_ZStL19piecewise\_construct: 4. .zero 1 5. .local \_ZStL8\_\_ioinit 6. .comm \_ZStL8\_\_ioinit,1,1 | (1-4) Declares \_ZStL19piecewise\_construct as an object with a size of 1 byte and initializes it to zero. This object contains all of the source code from the iostream header file  (5-6) Declares \_ZStL8\_\_ioinit as a local symbol and allocates 1 byte for it. |
| none | N/A |
| 1. .text 2. .globl  main 3. .type main, @function | 1. This tells the assembler that the following code is part of the text/code segment. Executable instructions for the program are stored here. 2. Declares main as global. 3. Indicates that main’s type is a function |
| 1. main: 2. .LFB1493: 3. .cfi\_startproc 4. pushq %rbp 5. .cfi\_def\_cfa\_offset 16 6. .cfi\_offset 6, -16 7. movq  %rsp, %rbp 8. .cfi\_def\_cfa\_register 6 9. subq  $16, %rsp | 1. This is a label indicating the start of the main function 2. This is a local label used by the assembler for internal purposes 3. This marks the beginning of the function. 4. Pushes the current value of the base pointer register (%rbp) onto the stack. 5. Offsets CFA 16 bytes from the current stack pointer 6. Specifies that the %rbp register is located at an offset of -16 from the CFA 7. Moves the stack pointer to the base pointer (setting up the new stack frame) 8. This changes the CFA register to %rbp (register number 6) 9. Subtracts 16 from the stack pointer allocating 16 bytes |
| 1. movl  $10, -12(%rbp) | 1. Move the value 10 into the memory location -12(%rbp) – 12 bytes above our register %rbp |
| 1. movl  $5, -8(%rbp) | 1. Move the value of 5 into the memory location -8(%rbp) – 8 bytes above our register %rbp |
| (1) movl  -12(%rbp), %eax | (1) Move the value at location -12(%rbp) into  the eax register |
| 1. imull -8(%rbp), %eax 2. movl %eax, -4(%rbp) | (1) Multiply the value in eax by the value stored  at memory location -8(%rbp) and store the  result in the eax register  (2) Move the value in the eax register into the  memory location -4(%rbp) |
| 1. movq  \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax 2. movq  %rax, %rsi 3. leaq  \_ZSt4cout(%rip), %rdi 4. call  \_ZNSolsEPFRSoS\_E@PLT 5. movq  %rax, %rdx 6. movl  -4(%rbp), %eax 7. movl  %eax, %esi 8. movq  %rdx, %rdi 9. call  \_ZNSolsEi@PLT 10. movl $0, %eax | (1) Move the address of the endl function into  the rax register  (2) Move the value in the rax register into the rsi  Register  (3) Load the address of the cout function into the  rdi register  (4) Calls the function to output endl to cout  (5) Move the value in the rax register to the rdx  register  (6) Move the value at memory location -4(%rbp)  into the eax register  (7) Move the value from the eax register into the  esi register  (8) Move the value in the rdx register to the rdi  Register  (9) Call the function to output the value in esi to  cout  (10) Move the value 0 into the eax register. This  is used to indicate successful completion |
| 1. leave 2. ret | (1)  Cleans up the stack frame  (2) Returns from the function |