# **CS 410 C++ to Assembly With Loops Activity Template**

**Step 1:** Explain the functionality of the C++ code.

## C++ Code Functionality

| **C++ Line of Code** | **Explanation of Functionality** |
| --- | --- |
| #include<iostream> | Include input output library |
| using namespace std; | Declare namespace std |
| int main() | Declare main class type integer |
| { | Begin Main class action |
| int num, i; | Declare num and i as integers |
| int product =1; | Declare product as integer with value 1 |
| cout<<"Enter a number:\n"<< endl; | Print “Enter a Number” and new line |
| cin>>num; | Accept input to variable num |
| for(i=num;i>0; i--) | For loop with condition that i is set to the value of num, runs while i is greater than 0, and decrement i by 1 each time the loop runs |
| product = product \* i; | Product is set to current product value multiplied by i |
| cout<<"The factorial for " << num << "is: \n"<< product; | Print “The factorial for”, the value of num, “is:”, the value of product |
| return 1; | Return 1 as return value |
| } | End main activity |

**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; |  |
| int main() | 17 .globl main  18 .type main, @function |
| { | 21 .cfi\_startproc |
| int num, i; | 28 movq %fs:40, %rax  29 %rax, -8(rbp) |
| int product =1; | 31 movl $1, -12(%rbp) |
| cout<<"Enter a number:\n"<< endl; | 32 leaq .LC0(%rip), %rdi  33 leaq \_ZSt4cout(%rip), %rdi  34 call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT |
| cin>>num; | 42 leaq \_ZST3cin(%rip), %rdi  43 call \_ZNSirsERi@PLT |
| for(i=num;i>0; i--) | 47 cmpl $0, -16(%rbp)  49 movl -12(%rbp), %eax  52 subl $1, -16(%rbp) |
| product = product \* i; | 50 imull -16(%rbp), %eax |
| cout<<"The factorial for " << num << "is: \n"<< product; | 55 leaq .LC1(%rip), %rsi  56leaq \_ZSt4cout(%rip), %rdi  57call\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  59movl -20(%rbp), %eax  62call \_ZNSolsEi@PLT  63leaq .LC2(%rip), %rsi  65call\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  67movl -12(%rbp), %eax  70call \_ZNSolsEi@PLT |
| return 1; | 79 ret |
| } | 80 .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** |
| --- | --- |
| 17 .globl main | Begin the main class |
| 18 .type main, @function | Define main as fucntion |
| 21 .cfi\_startproc | Start the main activity |
| 28 movq %fs:40, %rax | Moves num to rax |
| 29 %rax, -8(rbp) | Move I to rbp register |
| 31 movl $1, -12(%rbp) | Moves 1 to product |
| 32 leaq .LC0(%rip), %rdi | References string at .LC0 |
| 33 leaq \_ZSt4cout(%rip), %rdi | Sets the cout for call |
| 34 call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT | Calls the cout |
| 42 leaq \_ZST3cin(%rip), %rdi | preps the cin for call |
| 43 call \_ZNSirsERi@PLT | Calls the cin |
| 47 cmpl $0, -16(%rbp) | Checks if I is greater than 0 |
| 49 movl -12(%rbp), %eax | Sets I = to num |
| 52 subl $1, -16(%rbp) | Decrements the value of I by 1 |
| 50 imull -16(%rbp), %eax | Multiplies product times I and stores to product |
| 55 leaq .LC1(%rip), %rsi | References string at LC1 |
| 56leaq \_ZSt4cout(%rip), %rdi | Prepares cout for call |
| 57call\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT | Calls cout print string |
| 59movl -20(%rbp), %eax | Moves vale to num |
| 62call \_ZNSolsEi@PLT | Calls next part of cout statement print variable |
| 63leaq .LC2(%rip), %rsi | References string at LC2 |
| 65call\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT | Calls cout to print string |
| 67movl -12(%rbp), %eax | Moves value of product |
| 70call \_ZNSolsEi@PLT | Calls cout to print variable |
| 79 ret | Returns statement |
| 80 .cfi\_endproc | Ends the process |