# **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> | Adds iostream library functionality |
| using namespace std; | Sets the namespace to standard library |
| int main() | Main function for application |
| { | Start of Program with curly bracket |
| int num, i; | Declaration of “num” and “I” integer |
| int product =1; | Declaration of product integer and assign to 1 |
| cout<<"Enter a number:\n"<< endl; | Using “Cout” to print message and print a new line |
| cin>>num; | “Cin” method to obtain user input and assign to the variable labeled num |
| for(i=num;i>0; i--)  product = product \* i; | Loop keeps its execution until elements are null. For each loop set the value of product multiplied by i to the variable product. |
| cout<<"The factorial for " << num << "is: \n"<< product; | Using “Cout” to print message with factorial value of the num variable. |
| return 1; | Returning the value of the function and end of flow within function block. Program is terminated |
| } | Informs Compiler end of function. Exit |

**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 num, i;  int product =1; | movq  %rax, -8(%rbp)  movl $1, -12(%rbp) |
| cout<<"Enter a number:\n"<< endl; | .LC0:  .string "Enter a number:\n"  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movq  \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax  movq  %rax, %rsi  movq  %rdx, %rdi  call  \_ZNSolsEPFRSoS\_E@PLT |
| cin>>num; | movq %fs:40, %rax  movq  %rax, -8(%rbp)  xorl  %eax, %eax  leaq  -20(%rbp), %rax  movq  %rax, %rsi  leaq  \_ZSt3cin(%rip), %rdi  call  \_ZNSirsERi@PLT  xorq %fs:40, %rcx  je  .L5  call  \_\_stack\_chk\_fail@PLT |
| for(i=num;i>0; i--)  product = product \* i; | movl  -20(%rbp), %eax  movl  %eax, -16(%rbp)  .L3:  cmpl  $0, -16(%rbp)  jle .L2  subl  $1, -16(%rbp)  jmp .L3  movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl  %eax, -12(%rbp) |
| cout<<"The factorial for " << num << "is: \n"<< product; | .LC1:  .string "The factorial for "  leaq .LC1(%rip), %rsi  leaq  \_ZSt4cout(%rip), %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movl  -20(%rbp), %eax  movl  %eax, %esi  movq  %rdx, %rdi  call  \_ZNSolsEi@PLT  .LC2:  .string "is: \n"  leaq  .LC2(%rip), %rsi  movq  %rax, %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movl  -12(%rbp), %eax  movl  %eax, %esi  movq  %rdx, %rdi  call  \_ZNSolsEi@PLT |
| return 1; | 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** |
| --- | --- |
| .globl main  .type main, @function  main: | Globl: informs program to add the main label.  Type: Informs program that main is executable code.  Main: Declares the label main as entry point |
| movq  %rax, -8(%rbp)  movl $1, -12(%rbp) | %rbp assigned as integer  %rbp assigned a value of 1 |
| .LC0:  .string "Enter a number:\n"  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movq  \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax  movq  %rax, %rsi  movq  %rdx, %rdi  call  \_ZNSolsEPFRSoS\_E@PLT | .LCO: String used to display message  Pointers: Stored on stack  Cin: method called for user input  Movq: implemented to optimize stack  %rax: takes user input and moves value  End of function is called |
| movq %fs:40, %rax  movq  %rax, -8(%rbp)  xorl  %eax, %eax  leaq  -20(%rbp), %rax  movq  %rax, %rsi  leaq  \_ZSt3cin(%rip), %rdi  call  \_ZNSirsERi@PLT  xorq %fs:40, %rcx  je  .L5  call  \_\_stack\_chk\_fail@PLT | Registers are optimized in stack to implement values within the function.  Xorl: method used to 0 out regester.  Cin: method called to store value  Assembly moves to .L5L: Table used if conditions are met. |
| movl  -20(%rbp), %eax  movl  %eax, -16(%rbp)  .L3:  cmpl  $0, -16(%rbp)  jle .L2  subl  $1, -16(%rbp)  jmp .L3  movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl  %eax, -12(%rbp) | Integer I is assigned to the variable num and is moved throughout.  The process compares the value at %rpb and if it is greater than 1, the code will execute inside the loop. If it hasn’t then value of 1 is subtracted from integer “I”. The program then loops once again.  The value for the product variable is multiplied with the integer “I” and then stored. |
| .LC1:  .string "The factorial for "  leaq .LC1(%rip), %rsi  leaq  \_ZSt4cout(%rip), %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movl  -20(%rbp), %eax  movl  %eax, %esi  movq  %rdx, %rdi  call  \_ZNSolsEi@PLT  .LC2:  .string "is: \n"  leaq  .LC2(%rip), %rsi  movq  %rax, %rdi  call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq  %rax, %rdx  movl  -12(%rbp), %eax  movl  %eax, %esi  movq  %rdx, %rdi  call  \_ZNSolsEi@PLT | String stored in .LC1 and displayed when called.  Values are moved throughout to be used and passed to print method when complete.  String stored in .LC2 and displayed when called.  Values are moved throughout to be used and passed to print method when complete. |