C++ to Assembly Activity

CS-410-R4890 Software Reserve Engineering

Raphael Coloma

March 15, 2024

# **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> | Includes input output libraries for allow use of those function calls |
| using namespace std; | Defines the std namespace for the scope of the program |
| int main()      { | Initialize integer class main  Beginning of main class code block |
| int num, i; | Initialize integer variables num and i. |
| int product=1; | Initialize integer variable product and assign the value 1 to the variable. |
| cout<<"Enter a number:\n"; | Print to screen the character string “Enter a number:” and start a newline. |
| cin>>num; | Accept the user input and store the value in the variable num. |
| for (i=num;i>0; i--) | For loop declaration. Assign value in num variable to variable i. Continue loop while the value in variable i is greater than 0. After each iteration, decrease the value in variable i by 1. |
| product = product \* i; | Multiply the value in the variable product by the value in variable i. Then store that value in the variable product. |
| cout<<"The factorial for "<<num << "is: "<<  product<< endl; | First, print to screen the character string “The factorial for ”. Then print to the screen the value in variable num. Next, print the character string “is: “. Then print the value in variable product. Lastly, start a newline. |
| return 0;  } | Ends the main class and returns 0 to signify success.  Ending of the main class code block |

**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 $32, %rsp |
| int num, i; | movq %rax, -8(%rbp) |
| int product=1; | movl $1, -12(%rbp) |
| cout<<"Enter a number:\n"; | .LC0:  .string "Enter a number:\n"  …  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT |
| cin>>num; | leaq -20(%rbp), %rax  movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax |
| for (i=num;i>0; i--) | movl %eax, -16(%rbp)  cmpl $0, -16(%rbp)  jle .L2  …  subl $1, -16(%rbp)  jmp .L3 |
| product = product \* i; | movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp) |
| cout<<"The factorial for "<<num << "is: "<<  product<< endl; | .LC1:  .string "The factorial for "  .LC2:  .string "is: \n"  …  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  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 0;  } | 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 | Defines main as a function of 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 $32, %rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| movq %rax, -8(%rbp) | Reserve 8 bytes above base pointer register %rbp by pushing value of register %rax |
| movl $1, -12(%rbp) | Push the value 1 to 12 bytes above the base pointer register %rbp. |
| .LC0:  .string "Enter a number:\n"  …  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT | Declare sting “Enter a number:” end with newline.  …  Place address of string at .LCO into register %rsi.  Call print function to print string |
| leaq -20(%rbp), %rax  movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax | Place the address of the variable at 20 bytes above register %rbp into register %rax.  Move value in register %rax to register %rsi.  Call cin function to read value from the user.  Move read user value from 20 bytes above register %rbp to register %eax. |
| movl %eax, -16(%rbp)  cmpl $0, -16(%rbp)  jle .L2  …  subl $1, -16(%rbp)  jmp .L3 | Loop conditions  Move read user value from register %eax to 16 bytes above register %rbp.  Compare 0 and read user value.  Exit loop if comparison fails.  After iteration, subtract 1 from read user value.  Jump to start of loop |
| movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp) | Move the value stored 12 bytes above register %rbp to register %eax.  Multiplied the value in register %eax by the value stored at 16 bytes above register %rbp and stored the answer in register %eax.  Move the result from register %eax to 4 bytes above register %rbp. |
| .LC1:  .string "The factorial for "  .LC2:  .string "is: \n"  …  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  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 | Declare string “The factorial for ”.  Declare string “is “ ending with a newline.  …  Place address of string at .LC1 into register %rsi.  Call print function to print string.  Move value from 20 bytes above register %rbp to register %eax.  Call print function to print value of num variable.  Place address of string at .LC2 into register %rsi.  Call print function to print string.  Move value from 12 bytes above register %rbp to register %eax.  Call print function to print value of product variable. |
| leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc | Leave function.  End process.  The program is terminated. |