**2-1 Activity: C++ to Assembly**

Dylan Jeffery

SNHU

CS-410

Brian West

1/15/2022

# **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** |
| --- | --- |
| int num, i; | Initialize integers ‘num’ and ‘I’ |
| int product =1; | Initialize integer product and set it to 1 |
| cout<<"Enter a number:\n"<< endl; | Prompt user for a number |
| cin>>num; | Read user input and map it to variable num |
| for(i=num;i>0; i--)          product = product \* i; | Decremental for loop finds factorial of number provided by user (User number is multiplied by the integer right below it, this process repeats until i = 1) |
| cout<<"The factorial for " << num << "is: \n"<< product; | Print out the result, formatted for readability. |
| return 1; | Indicates a successful run |

**Step 2:** Convert the C++ file into assembly code.

**.file "assignment2\_1.cpp"**

**.text**

**.section .rodata**

**.type \_ZStL19piecewise\_construct, @object**

**.size \_ZStL19piecewise\_construct, 1**

**\_ZStL19piecewise\_construct:**

**.zero 1**

**.local \_ZStL8\_\_ioinit**

**.comm \_ZStL8\_\_ioinit,1,1**

**.LC0:**

**.string "Enter a number:\n"**

**.LC1:**

**.string "The factorial for "**

**.LC2:**

**.string "is: \n"**

**.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 $32, %rsp**

**movq %fs:40, %rax**

**movq %rax, -8(%rbp)**

**xorl %eax, %eax**

**movl $1, -12(%rbp)**

**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**

**leaq -20(%rbp), %rax**

**movq %rax, %rsi**

**leaq \_ZSt3cin(%rip), %rdi**

**call \_ZNSirsERi@PLT**

**movl -20(%rbp), %eax**

**movl %eax, -16(%rbp)**

**.L3:**

**cmpl $0, -16(%rbp)**

**jle .L2**

**movl -12(%rbp), %eax**

**imull -16(%rbp), %eax**

**movl %eax, -12(%rbp)**

**subl $1, -16(%rbp)**

**jmp .L3**

**.L2:**

**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**

**movl $1, %eax**

**movq -8(%rbp), %rcx**

**xorq %fs:40, %rcx**

**je .L5**

**call \_\_stack\_chk\_fail@PLT**

**.L5:**

**leave**

**.cfi\_def\_cfa 7, 8**

**ret**

**.cfi\_endproc**

**.LFE1493:**

**.size main, .-main**

**.type \_Z41\_\_static\_initialization\_and\_destruction\_0ii, @function**

**\_Z41\_\_static\_initialization\_and\_destruction\_0ii:**

**.LFB1982:**

**.cfi\_startproc**

**pushq %rbp**

**.cfi\_def\_cfa\_offset 16**

**.cfi\_offset 6, -16**

**movq %rsp, %rbp**

**.cfi\_def\_cfa\_register 6**

**subq $16, %rsp**

**movl %edi, -4(%rbp)**

**movl %esi, -8(%rbp)**

**cmpl $1, -4(%rbp)**

**jne .L8**

**cmpl $65535, -8(%rbp)**

**jne .L8**

**leaq \_ZStL8\_\_ioinit(%rip), %rdi**

**call \_ZNSt8ios\_base4InitC1Ev@PLT**

**leaq \_\_dso\_handle(%rip), %rdx**

**leaq \_ZStL8\_\_ioinit(%rip), %rsi**

**movq \_ZNSt8ios\_base4InitD1Ev@GOTPCREL(%rip), %rax**

**movq %rax, %rdi**

**call \_\_cxa\_atexit@PLT**

**.L8:**

**nop**

**leave**

**.cfi\_def\_cfa 7, 8**

**ret**

**.cfi\_endproc**

**.LFE1982:**

**.size \_Z41\_\_static\_initialization\_and\_destruction\_0ii, .-\_Z41\_\_static\_initialization\_and\_destruction\_0ii**

**.type \_GLOBAL\_\_sub\_I\_main, @function**

**\_GLOBAL\_\_sub\_I\_main:**

**.LFB1983:**

**.cfi\_startproc**

**pushq %rbp**

**.cfi\_def\_cfa\_offset 16**

**.cfi\_offset 6, -16**

**movq %rsp, %rbp**

**.cfi\_def\_cfa\_register 6**

**movl $65535, %esi**

**movl $1, %edi**

**call \_Z41\_\_static\_initialization\_and\_destruction\_0ii**

**popq %rbp**

**.cfi\_def\_cfa 7, 8**

**ret**

**.cfi\_endproc**

**.LFE1983:**

**.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 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 num, i; | movl $0, %eax (this is removed when next line is added and code is recompiled) |
| int product =1; | movl $1, -4(%rbp)  movl $1, %eax |
| cout<<"Enter a number:\n"<< endl; | + 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  + movl $1, -12(%rbp)  …  + leaq -16(%rbp), %rax  + movq %rax, %rsi  + leaq \_ZSt3cin(%rip), %rdi  + call \_ZNSirsERi@PLT  movl $1, %eax  + movq -8(%rbp), %rcx  + xorq %fs:40, %rcx  + je .L3  + call \_\_stack\_chk\_fail@PLT |
| for(i=num;i>0; i--)          product = product \* i; | leaq -20(%rbp), %rax  movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax  movl %eax, -16(%rbp)  .L3:  cmpl $0, -16(%rbp)  jle .L2  movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp)  subl $1, -16(%rbp)  jmp .L3  .L2:  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx  je .L5  call \_\_stack\_chk\_fail@PLT  .L5:  leave |
| cout<<"The factorial for " << num << "is: \n"<< product; | + 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 |

**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** |
| --- | --- |
| .LC0:  .string "Enter a number:\n"  .LC1:  .string "The factorial for "  .LC2:  .string "is: \n" | Store these strings for later |
| subq $32, %rsp | This is a compiler optimization |
| movq %fs:40, %rax  movq %rax, -8(%rbp)  xorl %eax, %eax | Standard prologue, xorl line initializes a register. |
| movl $1, -12(%rbp) | Move 1, 12 bytes above register %rgp  (int product = 1) |
| leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi | Load effective address lines for pointers |
| 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  leaq -20(%rbp), %rax  movq %rax, %rsi | Call <std::cout> and other library functions related to prompting user for a number  Move contents of %rax and %rdx  Leaq is moving some value (from 20 bytes above %rbp) to %rax which is then moved to %rsi |
| leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax  movl %eax, -16(%rbp) | Read user input into register which pointer was set previously.  Move value (from 20 bytes above %rbp) to %eax and then move that to 16 bytes above %rbp  The last two lines mirror the last two lines of the last block |
| .L3:  cmpl $0, -16(%rbp)  jle .L2 | What is the value of 0 with respect to the value 16 bytes above %rbp (our loop decrementor)  If !(i > 0) jump to .L2  jle is a conditional jump that happens after a comparison |
| movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp)  subl $1, -16(%rbp)  jmp .L3 | If the loop continues, move value of product into place.  Multiply product and i  Move our new value 12 bytes above %rbp and subtract 1 from decrementor.  Jump back to beginning of .L3 |
| .L2:  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  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx  je .L5  call \_\_stack\_chk\_fail@PLT  .L5:  leave | This sequence of assembly code is triggered by the termination of the for loop.  First leaq statement reads one of the pre-defined strings. Second one is related to cout pointer.  Call to std::cout  Move contents of %rax to %rdx  Move contents of 20 bytes above %rbp to %eax  Move Contents of %eax to %esi  Move contents of %rdx to %rdi  Call to pull final value of product variable for console output  Move contents of %rax to %rdi  More cout calls  Move commands and more cout calls to finish the formatted output.  xorq check if equal to original value  then jump to .L5  if not call stack fail  Exit the program |
|  |  |