C++ to Assembly Activity

CS-410-R4890 Software Reserve Engineering

Raphael Coloma

March 10, 2024

# **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 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 width=10; | Initialize integer variable width and assign the value 10 to the variable |
| int height=5; | Initialize integer variable height and assign the value 5 to the variable |
| int area; | Initialize integer variable area |
| area = width \* height; | Multiple the values in variables width and height and assign the total to the variable area |
| cout<<endl<< area; | Print to screen a newline character then the value stored in variable area |
| 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:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp 6 |
| 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\_traitsIcEERSt13basic\_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;  } | movl $0, %eax  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:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp | Pushes the values of the registers onto the stack and uses the registers to run the logic. |
| movl $10, -12(%rbp) | The value 10 was pushed 12 bytes above the base pointer register %rbp. |
| movl $5, -8(%rbp) | The value 5 was pushed 8 bytes above base pointer register %rbp. |
| movl -12(%rbp), %eax | The value stored 12 bytes above register %rbp (10) is moved to register %eax. |
| imull -8(%rbp), %eax | The value stored 8 bytes above register %rbp (5) is multiplied by the value in register %eax (10) and stored in register %eax. |
| movl %eax, -4(%rbp) | The value in register %eax (50) is moved 4 bytes above register %rbp. |
| movq \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax | The value in the variable is moved to register %rax. |
| movq %rax, %rsi | Prepares read register for read variable |
| leaq \_ZSt4cout(%rip), %rdi | Makes %rdi point to variable to print |
| call \_ZNSolsEPFRSoS\_E@PLT | Call print function to print newline character |
| movq %rax, %rdx | The value in register %rax is moved to register %rdx. |
| movl -4(%rbp), %eax | The value stored 4 bytes above register %rbp (50) is moved to register %eax. |
| movl %eax, %esi | The value in register %eax (50) is moved to register %esi. |
| movq %rdx, %rdi | Prepares read register for read variable |
| call \_ZNSolsEi@PLT | Call print function for variable |
| movl $0, %eax  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc | The value 0 is moved to register %eax.  The program is terminated |