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

Step 1: Convert the assembly code into C++ code.

Step 2: Explain the function of the converted C++ code.

| **Assembly Code** | **C++ Code** | **Explanation of Functionality** |
| --- | --- | --- |
| movl −8(%rbp), %eax sall $3, %eax subl $3, %eax movl %eax, −4(%rbp) | Int x,y;  X = x\*8;  X = x-3;  Int y = x; | **Assembly**   * Call a variable, x, and store the value of x at address of 8 bits in register rbp, in register eax. * The value of register eax is then shifted left 3 bits. * The value of register eax is then subtracted by 3. * Call a value, y, and store the value at 4 bits in rbp, in register eax.   **C++**   * The code calls a variable x of type integer. * Since x is shifted left 3 bits in Assembly, x will be set to the value of x\*8. This is due to each shift to times larger per bit space. This will then make 2 to the power of 3 multiplied by x. * With eax representing x in Assembly, the c++ will then call x and set it to x subtracted by 3, as shown in the Assembly code. * The address at 4 bits of rbp set to eax in Assembly then addresses c++ as calling variable yof type int and setting it to x. |
| movl −8(%rbp), %eax sall $2, %eax subl $1, %eax leal 7(%rax), %edx testl %eax, %eax cmovs %edx, %eax sarl $3, %eax  movl %eax, −4(%rbp) | Int x, y ;  X = x\*4;  X = x-1;  X = x / 8;  y = x; | **Assembly**   * Call a variable x and set register eax to the value of 8 bits in register rbp, in eax. * Shift value of register eax left 2 bits * Subtract the value of register eax by 1. * Store in register edx, the value at rax + 7.This sets the flags. * Test to see if eax is equal to 0. If so, then register edx equals register eax value. * If eax is not equal to 0, then eax will be shifted right by 3 bits. * Call a variable z with a value of register eax and place in rbp at 4 bits.   C++   * Variables x, and y of type int are declared. * X is moved 3 bits left in assembly which is equivalent to x being set to x multiplied by 2 to the 2nd power (4). * X is then set to the value x – 1. * X is then shifted right by 3 bits which is equivalent to x being divided by 2 to the 3rd power (8). * Variable y is then set to x. |
| movl −8(%rbp), %eax leal 7(%rax), %edx testl %eax, %eax cmovs %edx, %eax sarl $3, %eax movl −8(%rbp), %edx sall $2, %edx addl %edx, %eax  movl %eax, −4(%rbp) | Int x, y;  X = x /8;  Y = y \* 4;  X = x + y; | **Assembly**   * Call variable x and assign value at address 8 bits in rbp to eax. * Set edx register with the value at rbp + 7 * Check for eax to be above zero. * The statement is a conditional move that checks if the negative is for %edx to %eax. * Move value of register eax right 3 bits * Call register edx and set its value to address rbp at 8 bits * Move register edx left 2 bits * Add registers edx and eax * Set value at register eax to rbp at 4 bits.   **C++**   * Declare variables x and y of type integer * When eax is shifted right 3 bits it is equivalent to x being set to x divided by 2 to the 3rd power (8) * Variable y is then shifted to the left 2 bits which is equivalent to x being set to x multiplied by 2 to the 2nd power (4). * Since x is represented in assembly by eax, it is then set to the sum of x and y. |