# **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 input;  int output; | Move contents of the  −8(%rbp) to %eax  take the value of %eax  multiply it by 8 and shift  3 bits to the left  subtract 3 from the  contents of %eax  Move contents of the %eax to  −4(%rbp) |
| 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) | output = ((input \* 8) – 1) / 8; | Move contents of the  −8(%rbp) to %eax  take the value of %eax  multiply it by 4  shift  2 bits to the left  subtract 1 from the  contents of %eax  load the memory  address of 7(%rax) into %edx. Chekcs next line  %eax is above zero or not  statement is a conditional  move that checks if the negative is  for %edx to %eax  the value of %eax  divide by 8 and shift 3 bits to  the right side.  Move contents of the %eax to  −4(%rbp) |
| 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) | output = output + ((input \* 4) + 2) / 8; | Move contents of the  −8(%rbp) to %eax  put the memory  address of 7(%rax) into %edx  Check next line  %eax is above zero or not.  conditional  move that checks if the negative is  for %edx to %eax  take the value of %eax  divide it by 8 and shift 3 bits to  the right side  Move contents of the  −8(%rbp) to %eax.  take the value of %edx  multiply it by 4 and it will shift  2 bits to the left  add %edx from the  contents of %eax  Move contents of the %eax to  −4(%rbp) |