CS 410 Assembly to C++ Activity

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 = (input << 3) - 3;
		1. Move contents of -8(%rbp) to %eax
		2. Take value of %eax and shift 3 bits to the left
		3. Subtract 3 from the value of %eax
		4. Move the contents of %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)
	int input;
	int output = ((input << 2) – 1) >> 3;
		1. Move contents of -8(%rbp) to %eax
		2. Take value of %eax and shift 2 bits to the left
		3. Subtract 1 from the value of %eax
		4. Load effective address - put memory address of 7(%rax) into %edx
		5. Test %eax to see if it's above zero (AND)
		6. Conditional move if negative for %edx to %eda
		7. Take value of %eax and shift 3 bits to the right
		8. Move the contents of %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)
	int input;
	int output = (input >> 3) + (input << 2);
		1. Move contents of -8(%rbp) to %eax
		2. Load effective address - put memory address of 7(%rax) into %edx
		3. Test %eax to see if it's above zero (AND)
		4. Conditional move if negative for %edx to %eda
		5. Take value of %eax and shift 3 bits to the right
		6. Move contents of -8(%rbp) to %eax
		7. Take value of %edx and shift 2 bits to the left
		8. Add the values of %edx and %eax
		9. Move the contents of %eax to -4(%rbp)