# **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 a, b;  b = (a \* 8) - 3; | There are 2 variables that are in rbp at -8 and -4. I named those a(-8) and b(-4). A gets moved into the eax register. The register is left shift by 3 which is the equivalent of multiplying by 8. 3 is then subtracted from eax and then stored in eax. Finally eax is stored in b (-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 a, b;  if ((a \* 4) – 1 < 0) {  b = ((a \* 4) – 1 + 7) / 8;  }  else {  b = (a \* 4) – 1) / 8;  } | There are 2 variables that are in rbp at -8 and -4. I named those a(-8) and b(-4). ). A gets moved into the eax register. The register is left shift by 2 which is the equivalent of multiplying by 4. 1 is then subtracted from eax and then stored in eax. Next it takes rax (64 bits) which if I read correctly is the same as eax (lower 32 bits), adds 7 and stores in edx. Testl used with both argument as eax sets a zero flag to 1 if eax is 0 and sets a sign flag to 1 if eax is negative. Cmovs then uses the sign flag and if it is 1 (eax is negative) moves edx (eax + 7) to eax. Eax is then right shifted by 3 (divided by 8) . Eax is then stored in b (-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 a, b;  if (a < o) {  b = ((a + 7) / 8) + (a \* 4);  }  else {  b = (a / 8) + (a \* 4);  } | There are 2 variables that are in rbp at -8 and -4. I named those a(-8) and b(-4). ). A gets moved into the eax register. Next it takes rax (64 bits) which if I read correctly is the same as eax (lower 32 bits), adds 7 and stores in edx. Testl used with both argument as eax sets a zero flag to 1 if eax is 0 and sets a sign flag to 1 if eax is negative. Cmovs then uses the sign flag and if it is 1 (eax is negative) moves edx (eax + 7) to eax. Eax is then right shifted by 3 (divided by 8). Then a is loaded from -8(%rbp) into edx. This is then left shift by 2 (multiply by 4). Edx (a\*4) is added to eax (if a is negative (a+7)/8, if not (a/8)) and stored in eax. Eax is then stored in b (-4%rbp). |