CSCI 2330 – x86-64 Assembly Exercises

1. For each of the following x86-64 instructions, rewrite as a C-style command using assignment (=), pointer dereferencing (*), and regular arithmetic operators (+, –, etc). You can use register names as subexpressions – e.g., "**movq %rax, %rcx**" could be rewritten as "**rcx = rax**". Assume no data size scaling for C pointer arithmetic.

(a) `addq %rax, %rcx`

(b) `movq %rax, (%rcx)`

(c) `subq (%rax), %rcx`

(d) `leaq (%rax), %rcx`

(e) `leaq 9(%rax, %rdx), %rbx`

(f) `addq 9(%rax, %rdx), %rbx`

2. Assuming **func1** and **func2** are non-void functions that each take two arguments, rewrite the following x86-64 instructions as a series of C-style function calls. You can define and use any variables you wish.

```
movq    $5, %rsi
movq    $8, %rdi
callq   func1
movq    %rax, %rsi
callq   func2
movq    %rax, %rdi
callq   func1
```

3. Rewrite the x86-64 instructions below as a C function, which you can assume is non-void and takes one argument. Remember to specify the appropriate types of the argument and return value. You can use any local variables you wish. **Do not use any goto statements in your function.**

```
compute:
     movq    $0, %rax
     movq    $1, %rbx
     jmp     .L2
.L1: addq    $1, %rax
     salq    $1, %rbx        # left shift
.L2: cmpq    %rdi, %rbx
     jl      .L1             # jump if less
     ret
```