

CSCI 2330 – x86-64 Assembly Exercises

1. Assuming `func1` and `func2` are non-void functions that each take two **long** arguments and return a **long**, rewrite the following x86-64 instructions as a series of **three** function calls in C. Assume that the size of a **long** is 8 bytes. You may define variables to save and reuse the return values of these function calls if you wish (but you should still only write three lines of code, each containing one function call).

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

2. Consider the x86-64 instructions below for a function named `check`. Rewrite this code as a C function, which you can assume takes two **int** arguments and returns an **int**. You can use any variable names desired in your C function.

`check`:

```
subl    $4, %edi
cmpl    %edi, %esi
setl    %al
movzbl  %al, %eax
ret
```

3. Assume that `x` is a 4-byte int that starts in the register `%ebx`, `compute` is a function that takes and returns an **int**, and `verify` is a function that takes a **char**. Translate the following snippet of C code into equivalent x86-64 assembly instructions. You may assume that the value of `x` is still in `%ebx` after `compute` returns. **Hint:** Remember to use appropriately-sized registers. The 1-byte virtual register of `%rdi` is called `%dil`.

```
int result = compute(x);
verify(x > result);
```