CS 31 Homework 4: x86_64 Arithmetic

Due at 11:59pm, Thursday, October 9, 2025

Full Names:

1. Assume the CPU is executing a program and the state of some of its registers is given in the table below. Show how the registers would be updated by the sequence of x86_64 instructions also listed below, i.e. fill in the `Final Value` column. Show your work by listing the intermediate values of the registers.

Register	Initial Value	Final Value
%rax	0	
%rbx	1	
%rcx	2	
%rdx	3	

Here are the x86_64 instructions:

```
add    $20, %rax
add    %rax, %rbx
sub    %rcx, %rbx
add    $3, %rcx
sub    %rdx, %rcx
add    %rdx, %rdx
dec    %rdx
shr    $4, %rbx
and    $0xfffffffe, %rdx # this is tricky
xor    %rax, %rax        # this is tricky
or     $0x0, %rcx
# think about these next two before answering
not    %rbx
add    $1, %rbx
```

2. Assume the CPU is executing a function that has local variables `x`, `y`, and `z` allocated on the stack, and that `x` is allocated at the memory address that is `-24` bytes from the address value stored in register `%rbp`, or `-24(%rbp)`. Assume `y` is stored at `-16(%rbp)`, and `z` is at `-8(%rbp)`.

   For the assembly code listed below and the starting register state listed on the next page:

   (a) In the CPU register and memory figures on the next page, show the values that will be stored in the registers and in memory when execution of these instructions is complete. If the value is unknown, write "?".

   (b) Write a C code translation of the assembly code sequence. You may assume that `x`, `y`, and `z` have already been declared as `int` variables in the C code. You do not need to write the entire function, just the lines of C that might have generated the x86_64 instructions. Hint: our solution is 5 lines of C code.

```
                                                C Code Translation
movq    $2, -8(%rbp)                           -------------------
movq    $3, -16(%rbp)
movq    -8(%rbp), %rdx
movq    -16(%rbp), %rax
addq    %rdx, %rax
movq    %rax, -24(%rbp)
incq    -8(%rbp)
salq    $1, -16(%rbp)
```

Memory Address	Final Value
0xffffff38	
0xffffff40	
0xffffff48	
0xffffff50	
0xffffff58	
0xffffff60	
0xffffff68	

Register	Initial Value	Final Value
%rax	4	
%rdx	7	
%rbp	0xffffff58	