CS 429 Quiz 2: July 3, 2018

Name/EID: ____________________________________________

Note that this quiz has two sides.

1. (5 points) Assume the values given in the table.

<table>
<thead>
<tr>
<th>Address</th>
<th>value</th>
<th>Register</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0xFF</td>
<td>%rax</td>
<td>0x100</td>
</tr>
<tr>
<td>0x104</td>
<td>0xAB</td>
<td>%rcx</td>
<td>0xCD</td>
</tr>
<tr>
<td>0x108</td>
<td>0x13</td>
<td>%rdx</td>
<td>0x4</td>
</tr>
<tr>
<td>0x10C</td>
<td>0x11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

State (in hex) what value is in %rax after each statement is executed, if there’s enough information to tell. If not, write “can’t determine.” Note that each statement is independent; this is not a sequential program. Register values are 64-bits, but you can assume leading zeros.

A. ______________ leaq (%rax,%rdx,4), %rax

B. ______________ leaq 10(%rax,%rdx), %rax

C. ______________ addq %rcx, %rax

D. ______________ leaq 0x108(%rdx), %rax

E. ______________ movq 0x108(%rdx), %rax

2. (2 points) By default in x86-64, the stack:

A. is located in the heap
B. is located at the “bottom” of memory
C. grows up toward larger addresses
D. grows down toward smaller addresses
3. (4 points total) For the following code fragment, assume that it’s running on an x86-64 bit machine. State in decimal what’s in the indicated register following execution:

```assembly
movl $31, %ecx
pushq %rcx
movl $10, %ecx
pushq %rcx
addq $3, 8(%rsp)
popq %rax
popq %rbx
```

A. What’s in %rax? _____________________________________________

B. What’s in %rbx? _____________________________________________

4. _______ (2 points) Suppose we invent a new technology that allows storing 3 states, not just 2. Instead of bits (0, 1), we have trits (-1, 0, +1). What’s the maximum number of integers we could represent in n trits?

A. $2^n$ B. $3n$ C. $3!$ (3 factorial) D. $3^n$ E. $2^{n+3}$

5. _______ (2 points) On the x86-64, the 32-bit string 0x400EBC5A could not represent which of these?

A. two’s complement integer
B. floating point number
C. pointer
D. unsigned integer
E. it could be any of them.

6. _______ (2 points) Arguments passed to functions in x86-64 are passed via:

A. main memory
B. the stack
C. registers
D. a combination of stack and registers
E. None of the above.

Page total: ______/10
7. (10 points) For each of the following determine whether it is true or false to assert that this is a legal x86-64 assembly statement. Please write on the line T if it is legal or F if it is not legal.

A. _____ movq %al, %rcx

B. _____ movq (%rax, %rcx, 3), %rbx

C. _____ movq %rax, (%rbx)

D. _____ leaq (%rax, %rcx, 4), %rbx

E. _____ movl %rax, %rbx

F. _____ movq (%rax), (%rbx)

G. _____ movq (%rax), %rsi

H. _____ movq (%rax,4), %rbx

I. _____ movzbl %al, %eax

J. _____ movzb %rax, %al
8. (9 points total) The following x86-64 assembly code is the compiled image of a C subroutine `mystery`.

```assembly
mystery:
    movl $0, %edx
    movl $0, %eax
    jmp .L2
.L3:
    addq %rax, %rax
    movq %rdi, %rcx
    andl $1, %ecx
    orq %rcx, %rax
    shrq %rdi
    addl $1, %edx
.L2:
    cmpl $63, %edx
    jle .L3
ret
```

A. (7 points) Fill in the missing C code that might have generated the code above:

```c
long mystery( long x ) {
    long y = __________;
    int i;

    for ( __________; __________; __________) {
        __________;

        __________;
    }

    __________;
}
```

B. (2 points) What does this function do?
9. (8 points total) The following x86-64 assembly code is the compiled image of a C subroutine **mystery**.

```
mystery:
    testl %edx, %edx
    jle .L3
    pushq %rbx
    movq %rdi, %rax
    xorq %rsi, %rax
testb $1, %al
    sete %bl
    movzbl %bl, %ebx
    subl $1, %edx
    shrq $1, %rsi
    shrq $1, %rdi
call mystery
    addl %ebx, %eax
    jmp .L2
.L3:
    movl $0, %eax
    ret
.L2:
    popq %rbx
    ret
```

Fill in the missing C code that might have generated the code above. Use the template; don’t add any variables or extra lines. *Be sure to use correct C syntax.*

```c
int mystery( unsigned long x, unsigned long y, int n ) {
    if ( ________________ )
        return ________________;
    else {
        return ________________;
        + ________________;
    }
}}
```
```c
long reverse(unsigned long x)
// Return a version of x with the bits reversed
{
    long y = 0;
    int i;
    for (i = 0; i < 64; i++) {
        y = ((y << 1) | (x & 1));
        x = (x >> 1);
    }
    return y;
}

int countmatches(unsigned long x, unsigned long y, int n)
// Return the number of bit positions that match in the last
// n bits of x and y.
{
    if (n <= 0)
        return 0;
    else {
        return
            ( (x & 1) == (y & 1) )
            + countmatches(x >> 1, y >> 1, n - 1);
    }
}
```