Name:_____ UID:_____ Discussion #: _____

Week 2 CS 33 Worksheet

1. Write a function in C that, given a number n, returns another number where the k^{th} (0-indexed) bit from the right is set to 0. Only use `return`, parentheses and binary operations: `+ - * / ~ & | << >>`

Examples:

`killKthBit(37, 2) = 33 because` $37_{10} = 100\mathbf{\underline{1}}01_2 \rightarrow 100\mathbf{\underline{0}}01_2 = 33_{10}$
`killKthBit(37, 3) = 37 because the 3rd bit is already 0.`

```
// 0 <= k < 8 * sizeof(n)
int killKthBit(int n, int k) {




}
```

2. `mov` vs `lea` - What are the final values of `%rax` and `%rbx` (in hex) after the following code is run?

```
movq $0x1000, %rdx
movq $0x12345, %r12
movq %r12, (0x1008)
movq 8(%rdx), %rax
leaq 8(%rdx), %rbx
```

%rax: **%rbx:**

3. Write some assembly code that computes the result of `(%rax + %rcx) * 4`, and stores the result in `%rdx` in no more than 3 instructions. For example, if `%rax = 5`, `%rcx = 10`, `%rdx` should equal 60. Don't modify the value of any register except `%rdx`. Note: there are multiple ways to do this - try and find as many as you can!

4. Invalid `mov` Instructions - Explain why these instructions are invalid in a 64-bit assembly program.

```
a) movl %eax, %rdx
```

```
b) movb %di, 8(%rdx)
```

```
c) movq (%rsi), 8(%rbp)
```

```
d) movw 0xFF, (%eax)
```

5. Consider the following similar functions:

`int f1(int a, int b) {`
 	
`  if ( b < a )`
`  return b;`
`  else`
`  return a;`
`}`
	`int f2(int a, int b) {`
	
	`  if ( a < b )`
	`  return a;`
	`  else`
	`  return b;`
	`}`
		`int f3(int a, int b) {`
		`  unsigned ub = b;`
		`  if ( ub < a )`
		`  return a;`
		`  else`
		`  return ub;`
		`}`

Which of the functions would compile into this assembly code on a 64- bit machine?

```
       movl %esi, %eax
       cmpl %eax, %edi
       jge .L4
       movl %edi, %eax
.L4:   ret
```

6. Review the following assembly code.

```
movq $0xaabbccddeeff1122, %rax
movl $0x12345678, %eax
movw $0x3344, %ax
movb $0x55, %al
movb $0x66, %ah
```

What are the values stored in `%rax` after **each** instruction?

7. Operand Form Practice (see Figure 3.3 in Section 3.4.1 in textbook, 3rd edition)
Assume the following values are stored in the indicated registers/memory addresses, and that all memory reads are 1 byte reads.

Address	Value	Register	Value
0x104	0x34	%rax	0x104
0x108	0xCC	%rcx	0x5
0x10C	0x19	%rdx	0x3
0x110	0x42	%rbx	0x4

Fill in the table for the indicated operands:

Operand	Value	Operand	Value
$0x110	_____	(%rax, %rbx)	_____
%rax	_____	3(%rax, %rcx)	_____
0x110	_____	256(, %rbx, 2)	_____
(%rax)	_____	(%rax, %rbx, 2)	_____
8(%rax)	_____	229(%rdx, %rcx, 8)	_____

8. Condition Codes and Jumps - Assume the addresses and registers are in the same state as in the previous question. Does the following code result in a jump to `.L2`?

```
leaq (%rax, %rbx), %rdi
cmpq $0x100, %rdi
jg .L2
```

9. Summarize what this program does in no more than ~20 words (refer to the input value as x in your summary):

```
     xor  %eax, %eax
     jmp  .btm
.top addq %rdi, %rax
     subq $1, %rdi
.btm cmpq $0x0, %rdi
     jg   .top
     ret
```