NAMAN MODANI
005-816-643

1. Write a function that, given a number n, returns another number where the k^th bit from the right is set to to 0.
Examples:
`killKthBit(37, 3) = 33` because $37_{10} = 100\mathbf{1}01_2 \rightsquigarrow 100\mathbf{0}01_2 = 33_{10}$
`killKthBit(37, 4) = 37` because the 4^{th} bit is already 0.

```
int killKthBit(int n, int k) {
    return (n & ~(1 << (k - 1)));
}
```

2. `mov` vs `lea` - describe the difference between the following:

`movq (%rdx), %rax` → takes contents of whatever is stored in %rdx and moves it to %rax
`leaq (%rdx), %rax`
↳ computes load effective address and stores in %rax.

3. Invalid `mov` Instructions - Explain why these instructions would not be found in an assembly program.

a) `movl %eax, %rdx`
destination operand has incorrect size.

b) `movb %di, 8(%rdx)`
instruction suffix and register size do not match.

c) `movq (%rsi),8(%rbp)`
source and destination cannot both be memory references.

d) `movw 0xFF, (%eax)`
%eax cannot be used as an address register (not 64 bits)

4. What would be the corresponding instruction to move 64 bits of data from register `%rax` to register `%rcx`?

movq (%rax), %rcx

5. Operand Form Practice (see page 181 in textbook)
Assume the following values are stored in the indicated registers/memory addresses.

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`	0x110	`3(%rax, %rcx)`	0x19
`%rax`	0x104	`256(, %rbx, 2)`	0xCC
`0x110`	0x42	`(%rax, %rbx, 2)`	0x19
`(%rax)`	0x34		
`8(%rax)`	0x19		
`(%rax, %rbx)`	0xCC		

6. Condition Codes and Jumps - Assume the addresses and registers are in the same state as in Problem 6. Does the following code result in a jump to `.L2`?

```
leaq (%rax, %rbx), %rdi   - 1
cmpq $0x100, %rdi   - 2
jg .L2   - 3
```

$\boxed{\text{YES}}$

1 0x104 + 0x04 = 0x108 → %rdi

2 Sets codes according to 0x108 − 0x100, no codes set.

3 jg is calculated as ~(SF^OF) & ~ZF, i.e. ~(0^0) & ~0 = 1&1 = 1. Hence jump.