Name: UID:

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

```
movq (%rdx), %rax
leaq (%rdx), %rax
```

2. Invalid `mov` instructions: explain why these instructions would not be found in an assembly program.

   (a) `movl %eax, %rdx`
   (b) `movb %di, 8(%rdx)`
   (c) `movq (%rsi), 8(%rbp)`
   (d) `movw $0xFF, (%eax)`

3.

   (a) What would be the corresponding instruction to move 64 bits of data from register `%rax` to register `%rcx`?

   (b) What would be the corresponding instruction to move 64 bits of data from the memory location stored in register `%rax` to register `%rcx`?

4. 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	_____	3(%rax, %rcx)	_____
%rax	_____	256(, %rbx, 2)	_____
0x110	_____	(%rax, %rbx, 2)	_____
(%rax)	_____		
8(%rax)	_____		
(%rax, %rbx)	_____		

5. Condition codes and jumps: assume the addresses and registers are in the same state as in the previous problem. Does the following code result in a jump to `.L2`?

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

6. Which of the functions `cool1`, `cool2`, or `cool3` would compile into this assembly code?

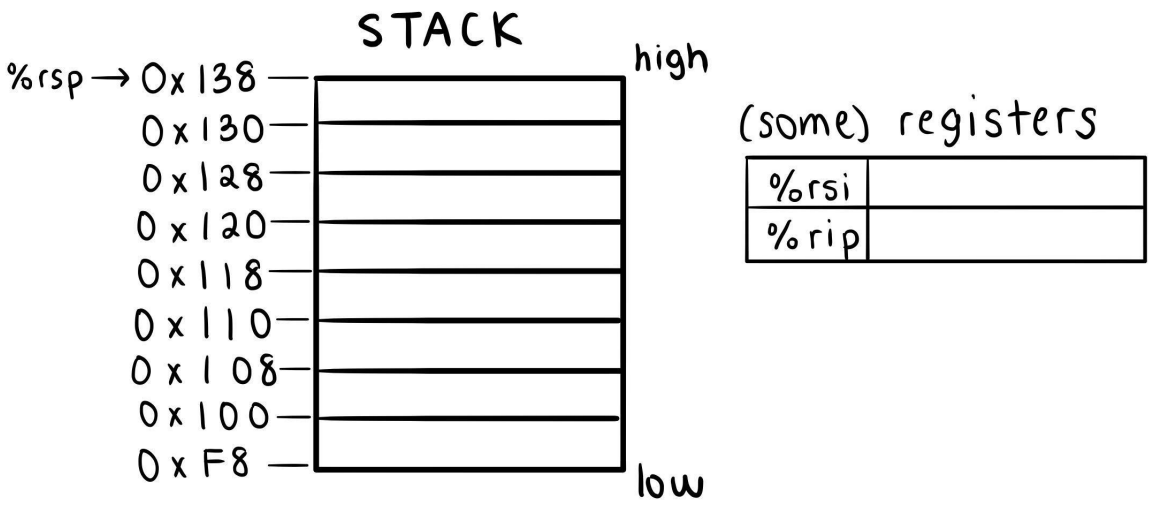
```
     movl %esi, %eax
     cmpl %eax, %edx
     jge .L4
     movl %edx, %eax
.L4:
     ret
```

```
int cool1(int a, int b) {
    if ( b < a )
     return b;
    else
     return a;
}

int cool2(int a, int b) {
    if ( a < b )
     return a;
    else
     return b;
}

int cool3(int a, int b) {
    unsigned ub = (unsigned) b;
    if ( ub < a )
     return a;
    else
     return ub;
}
```

7. Consider the following disassembled function:

```
000000000040102b <phase_2>:
  40102b:  55                    push   %rbp
  40102c:  53                    push   %rbx
  40102d:  48 83 ec 28           sub    $0x28,%rsp
  401031:  48 89 e6              mov    %rsp,%rsi
  401034:  e8 e3 03 00 00        callq  40141c <read_six_numbers>
  401039:  83 3c 24 01           cmpl   $0x1,(%rsp)
  …
```

(a) Assume `%rsp` initially has a value of `0x138`. Draw the stack (see example diagram below) for the execution of `<phase_2>`, updating the stack and register values as necessary after each line is executed.



(b) Right after the `callq` instruction has been executed, what are the values of `%rsp`, `%rsi`, and `%rip`?