Exercise 1

PROBLEM 1

Each of the following lines of code generates an error message when we invoke the assembler. Explain what is wrong with each line.

```
movw %(%rax), 4(%rsp, %rsi, 8)
movb %al, %sl
movq %rdi, $0x111
movl %r8, (%rdx)
movl %ecx, %rdx
movb %si, 8(%rbp)
movb $0xF, (%ebx)
```

PROBLEM 2

Assume variables `sp` and `dp` are declared with types

```
src_t *sp;

dest_t *dp;
```

where `src_t` and `dest_t` are data types declared with `typedef`. We wish to use the appropriate pair of data movement instructions to implement the operation

```
*dp = (dest_t)*sp;
```

Assume that the values of `sp` and `dp` are stored in registers `%rdi` and `%rsi`, respectively. For each entry in the table, show the two instructions that implement the specified data movement. The first instruction in the sequence should read from memory, do the appropriate conversion, and set the appropriate portion of register `%rax`. The second instruction should then write the appropriate portion of `%rax` to memory. In both cases, the portions may be `%rax`, `%eax`, `%ax`, or `%al`, and they may differ from one another.

Recall that when performing a cast that involves both a size change and a change of “signedness” in C, the operation should change the size first (Section 2.2.6).

src_t	dest_t	Instruction
long	long	movq (%rdi), %rax movq %rax, (%rsi)
char	int	
char	unsigned	
unsigned char	long	
int	char	
unsigned	unsigned char	

char	short	

PROBLEM 3

Suppose a 64-bit little endian machine has the following memory and register status. Fill in the blanks using 8-byte value and in hex notation. NOTE: **Instructions are independent**.

Memory status:

Address	Value
0x100	0xf0f0f0f0
0x104	0x78563412
0x108	0x1000

Register status:

Register	Value
%rax	0x104
%rbx	0x1
%rcx	0xffffffff fffffffc
%rdx	0x87654321

Fill in the blanks below:

Operation	Destination	Value
subq (%rax), %rdx		
imulq $2, (%rax, %rbx, 4)		
notq (%rax, %rcx)		
leaq 8(%rax, %rbx, 4), %rdx		
movb $0x1, %al		

PROBLEM 4

Indicate the status (0, 1 or unchanged) of the following flags after each instruction, please write "——" if the flag doesn't change. Assume 3 in %rax, -8 in %rbx. NOTE: **Each instruction works independently and would NOT affect each other**.

Instruction	OF	SF	ZF	CF
addq %rbx, %rax				
subq %rax, %rbx				
leaq (%rax, %rax, 2), %rax				
xorq %rax, %rax				
salq $2, %rbx				
cmpq %rax, %rbx				
testq %rax, %rbx				

PROBLEM 5

The C code

```
int comp(data_t a, data_t b) {
  return a COMP b;
}
```

shows a general comparison between arguments a and b, where data_t, the data type of the arguments, is defined (via typedef) to be one of the integer data types listed in Figure 3.1 and either signed or unsigned. The comparison COMP is defined via #define.

Suppose a is in some portion of %rdi while b is in some portion of %rsi. For each of the following instruction sequences, determine which data types data_t and which comparisons COMP could cause the compiler to generate this code.

```
A. cmpl %esi, %edi
   setl %al
B. cmpw %si, %di
   setge %al
C. cmpb %sil, %dil
   setbe %al
D. cmpq %rsi, %rdi
   setne %al
```