CSCI-UA.0201-003

Computer Systems Organization

Midterm Exam Fall 2015 (time: 60 minutes)

Last name: First name:

Notes:

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly.**
- **Questions vary in difficulty; it is strongly recommended that you do not spend too much time on any one question.**

1. [2 points] What would have happened if we didn't have linkers? State 2 consequences.

- Cannot use libraries.
- Must have our programs written as one huge file. Cannot have multi-file programs.

2. [5 points] Fill in the blanks in the following table, assume the instructions are executed sequentially (i.e. instruction 1 executed, then instruction 2, then instruction 3. Assume rax and rbx are holding signed numbers.

instruction#	**instruction**	**rax**	**rbx**	**CF**	**ZF**	**SF**	**OF**
Initially		0xFFFFFFFF	0x00000001	0	0	0	0
1	addq %rbx, %rax	0x000000000	0x00000001	1	1	0	0
2	testq %rbx, %rax	0x000000000	0x00000001	0	1	0	0
3	cmpq %rbx, %rax	0x000000000	0x00000001	0	0	1	0

3. [5 points] For the following assembly code and its corresponding C code, fill-in the blanks in the C code assuming x will go into %rbx and y will go into %rax;

```
        cmpq %rax, %rbx            unsigned long  x, y;  /* declaration of x and y;
        ja  L1                     if ( x > y )
        subq, %rbx, %rax            {
        jmp L2                             y += 10;
L1:     addq $10, %rax              }
L2:     addq %rbx, %rax            else
                                   {
                                           y -= x;
                                   }
                                   y += x;
```

4. [2 points] Can the carry flag (CF) and the overflow flag (OF) be both 1 at the same time? If yes, give an example of an operation that does this (no need for assembly code, just describe the operation). If not, explain why not.

Yes they can, because the processor does not differentiate between signed and unsigned.
example:

```
       1 0 1 1 1 1 1…1 1 1
     + 1 0 1 1 1 1 1…1 1 1
     1 0 1 1 1 1 1 1…1 1 0
```

So, two negative numbers yield a positive result (OF is set to 1), and there is a carry from the most significant bit (CF is set to 1). The compiler however, won't care about CF if the two numbers are signed, and won't care about OF if the two number are unsigned. But in both cases, the example shown above will yield an overflow whether numbers are signed or not.

5. [2 points] State two reasons for why do we need an assembler and not making the compiler generate the binary presentation right away.

- To be able to compare assembly code with HLL code. This enables professional programmers to enhance the quality (i.e. performance, etc) of their code. So, we need assembly code to be generated.
- The compiler is already a complicated piece of software. We don't want to add another task to it.
- Compiler and assembler are doing two different tasks, hence each one can be optimized differently.

6. Suppose x is an integer (i.e. 4 bytes). We want to test whether the most significant bit of x is 1 or not (i.e. the left most bit), so we wrote the expression:

if((x & mask) != 0)

a. [1 point] What is the value of mask, both in binary and hexadecimal?

10000000000000000000000000000000
0x80000000

b. [2 points] Which of the following expressions generate correct mask? Circle ALL correct answers. There may be more than one correct answer, or there may be none!

- 0x1FFFFFFF << 3
- 0x1FFFFFFF << 2
- two's complement of 0xFFFFFFFC
- two's complement of (-2)

c. [1 point] Please give the expression that sets the most significant bit of x to 1 and leave all the other bits unchanged.

x |= 0x80000000;