

## CPSC 213, Winter 2016, Term 1

### Midterm I Sample Questions

#### Exercise 1

What is the value of `i` after the following Java statements execute?

```
byte b = 0x84;

int i = b << 8;
```

Recall that `int`'s are 4-bytes long. Give your answer as a single number in hex.

#### Exercise 2

Consider the following content of a portion of memory (in the form of address: value):

```
0x1000: 0x12
0x1001: 0x34
0x1002: 0x56
0x1003: 0x78
```

What is the little endian value of the 4-byte integer at address 0x1000? Give your answer as a single value in hex.

#### Exercise 3

Consider the following C code with three global variables, `a`, `b`, and `c`, that are stored at addresses 0x1000, 0x2000, 0x3000, respectively, and a procedure `foo()` that accesses them.

```
int a[1]; // at address 0x1000
int b[1]; // at address 0x2000
int* c;    // at address 0x3000
void foo() {
    a[0] = 1;
    b[0] = 2;
    c = a;
    c[0] = 3;
    c = b;
    *c = 4;
}
```

Describe what you know about the content of memory following the execution of `foo()` on a 32-bit Little Endian processor. List only memory locations whose address and value you know. List each byte of memory separately using the form “byte\_address: byte\_value”. List all numbers in hex.

#### Exercise 4

Consider the following C code.

```
int a[10] = {0,1,2,3,4,5,6,7,8,9}; // i.e., a[i] = i
int* b = a+4;
int foo (int* x, int* y, int* z) {
    *x = *x + *y;
    *x = *x + *z;
    return *x;
}
int bar () {
    return foo (b - 2, a + (b - a) + (&a[7] - &a[6]), a + 2);
}
```

What value does `bar()` return? Justify your answer (1) by simplifying the description of the arguments to `foo()` as much as possible so that the relationship among them, if any, is clear and (2) by carefully explaining what happens when `foo()` executes.

#### Exercise 5

Consider the following C global variable declarations.

```
int a[10];
int* b;
int i;
```

Give the SM213 assembly code the compiler might generate for the following statements that access these variables. You may use labels `a`, `b`, and `c` for addresses. You may not assume anything about the value of registers. Comment every line.

#### Exercise 6

Consider the following C code.

```
int* b;
void set (int i) {
    b[i] = i;
}
```

There is a dangerous bug in this code. Carefully describe what it is. Assume that `b` was assigned a value somewhere else in the program.

#### Exercise 7

What is the value of the register `r0` after the following program executes?

```
ld $0x2004, r0    # r0 = 0x2004
ld (r0), r0       # r0 = m[r0]

.pos 0x2000
.long 0           # value at address 0x2000
.long 1           # value at address 0x2004
.long 2           # value at address 0x2008
.long 3           # value at address 0x200c
```