Lecture Midterm Exam
COMP 122/L
Fall 2017

Score breakdown:
Common data representation: 17 Points (~15%)
Common binary operations: 46 Points (~41%)
Floating point representation: 38 Points (~34%)
ARM assembly: 11 Points (~10%)
Total: 112 Points

There are 22 questions in all; there is no need to complete them in order.

Please write your name below, and **wait until told to begin:**

Ground Rules

1. You may have up to two 8 1/2 x 11 inch sheets of paper in front of you containing handwritten notes covering both sides of each sheet.

2. You may have the combined ARM reference card/SWI instruction handout in front of you.

3. You may have a calculator in front of you that can handle exponentiation.

4. If you have a question, raise your hand and I will come to you.

5. You **may not** communicate with anyone else. **Violations of this rule will result in a 0 on the exam.** This exam is purely individual effort.

Common Data Representation

1.) (2 pts) In hexadecimal, how much is a `B` in position `3` worth? Write a formula which evaluates to the answer.

2.) (2 pts) In binary, how much is a `1` in position `7` worth? Write a formula which evaluates to the answer.

3.) (4 pts) Convert decimal `213` into two-digit hexadecimal.

4.) (4 pts) What is decimal `-27` in two's complement notation? Represent your answer using 8 bits.

5.) (5 pts) What is the two's complement number `1101 0001` in decimal? The space in between is only for readability.

Common Binary Operations

6.) (2 pts) Using only the binary shift left operation, how can one multiply a number by decimal 16?

7.) (6 pts) Add the following two 8-bit binary numbers, producing an 8-bit result. Indicate the status of the carry and overflow bits at the end of the addition by circling the appropriate choices. The space in between is only for readability.

```
  1101 0110
+ 1001 0111
-----------
```

Overflow? Yes No

Carry? Yes No

8.) (6 pts) Add the following two 8-bit binary numbers, producing an 8-bit result. Indicate the status of the carry and overflow bits at the end of the addition by circling the appropriate choices. The space in between is only for readability.

```
  1111 0010
+ 1101 0110
-----------
```

Overflow? Yes No
Carry? Yes No

9.) (8 pts) Subtract the second 8-bit binary number from the first 8-bit binary number, producing an 8-bit result. Indicate the status of the carry and overflow bits at the end of the subtraction by circling the appropriate choices. The space in between is only for readability.

```
  1011 0100
- 0101 0000
-----------
```

Overflow? Yes No
Carry? Yes No

10.) (2 pts) What is:

```
  1010 1101
& 1011 1011
-----------
```

11.) (2 pts) What is:

```
  0011 0111
| 1001 1010
-----------
```

12.) (2 pts) What is:

```
  0110 0100
^ 1010 1101
-----------
```

13.) (2 pts) What is:

```
0110 1101 << 2
```

14.) (2 pts) What is:

$$1100\ 1010 >>_{arithmetic} 3$$

15.) (2 pts) What is:

$$1011\ 1111 >>_{logical} 4$$

16.) (6 pts) Consider the following Java-like code:

```
int number = <<read number from user>>;
int mask = MASK;
int result = number OP mask;

if (result != 0) {
  print("Bit 9 was set");
}
```

The above code is supposed to print "`Bit 9 was set`" if bit 9 of `number` was set to `1`. If bit 9 was not set, then this code should not print anything. What hexadecimal value should `MASK` be, and what bitwise operation should `OP` be, for the above code to work correctly?

17.) (6 pts) Consider the following Java-like code:

```
int number = <<read number from user>>;
int mask = MASK;
int numberWithBit11Set = number OP mask;
```

In the above code, the value of `numberWithBit11Set` is supposed to be the same as number, *except* that bit 11 should be unconditionally set to 1. What hexadecimal value should `MASK` be, and what bitwise operation should `OP` be, for the above code to work correctly?

Floating Point Representation

18.) (23 pts) What is the 32-bit binary floating-point representation of decimal `-247.125`? Show all work.

19.) (15 pts) What is the decimal representation of the following 32-bit floating point value? Spaces have been added only for improved clarity. Show all work. You may write a formula that evaluates to the final value instead of the value itself, if you prefer.

```
0100 0101 0000 0000 0001 1100 0000 0000
```

ARM Assembly

20.) (3 pts) What is wrong with the following code, if anything?

```
  .equ Exit, 0x11
  .equ Open, 0x66
  .equ Close, 0x68
  .equ Read_Int, 0x6C

  .data
filename:
  .asciz "myFile.txt"

  .text
  .global _start
_start:
  ldr r0, =filename
  mov r1, #0
  swi Open
  swi Read_Int
  swi Close
  swi Exit
  .end
```

21.) (6 pts) Consider the following code, which sets up a `.data` section:

```
  .data
label1:
  .word 0, 255
label2:
  .asciz "a"
```

Assuming the `.data` section starts at address `0`, how does this look in memory? Use the following table as a template. Each value should be represented with a two-digit hexadecimal number, and the preceding `0x` may be omitted. As a hint, the ASCII value of `'a'` in hexadecimal is `0x61`. If the value at a given index is unknown, mark the position with `?`.

Value												
Index	0	1	2	3	4	5	6	7	8	9	10	11

22.) (2 pts) What value will be in register `r0` at the end of the following code?

```
mov r0, #0
mov r1, #57
cmp r1, #100
addmi r0, r1, #3
```