# Lab #4
# Memory and Bit Manipulation

Michaelmas Term 2019, Weeks 11 and 12

Working individually, complete all of the exercises below. You will be graded on sections 2, 3 and 4 of this exercise. Your grade will be based on the work you show during your scheduled lab classes. You may show your work during either the first or second week of each two-week cycle.

If you show your work during the first week of a two-week cycle and receive a grade of 4 or 5, you do not need to attend the second week. If you receive a grade less than 4 and do not show your work again in the second week, you will not receive a grade for the exercise.

You must also submit your work to Blackboard. If you fail to submit your work on Blackboard, the grade you receive in class will not be counted.

Attendance will not be taken during lab classes. Receiving a grade for the work you have shown will indicate your participation in the lab exercise. If you fail to participate in more than one exercise during the semester, you may be returned as "non-satisfactory" for the semester.

## 1 Bit Manipulation

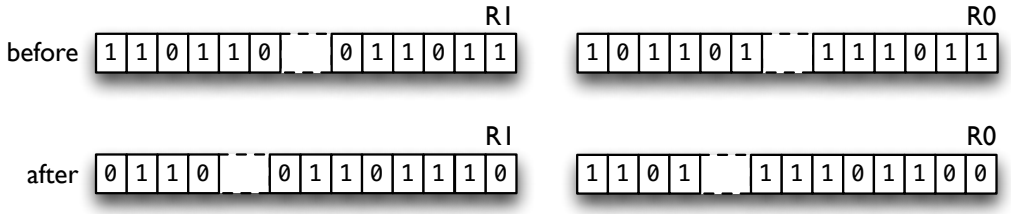
Translate the pseudo-code below into ARM Assembly Language. The program implements unsigned division. Verify this by testing your program with different values for a and b.

```
quotient = 0;
remainder = 0;
mask = 0x80000000;
while (mask != 0) {
    remainder = remainder << 1;

    if ( (a & mask) != 0)
    remainder = remainder | 1;

    if (remainder >= b) {
        remainder = remainder - b;
        quotient = quotient | mask;
    }

    mask = mask >> 1;
}
```

# 2 64-bit Shift [GRADED]

Design and write an ARM Assembly Language program that will perform a logical shift operation on a 64-bit value stored in registers `R0` and `R1`. Assume that the least-significant 32-bits are stored in `R0` and the most-significant 32-bits are stored in `R1`. The count of the number of bits to shift is stored in `R2`. If the count in `R2` is negative, your program should shift left. Conversely, if the count is positive, your program should shift right.

The following example shows how your program should logically shift left the 64-bit value stored in `R0` and `R1` when `R2` contains -2 (`0xFFFFFFFD` using the 2s Complement representation of negative values.)




Test your solution.

# 3 Pseudo-Random Numbers [GRADED]

Design and write an ARM Assembly Language program that will store a specified number of pseudo-random numbers in memory, beginning at address 0x40000000.

# 4 Anagrams [GRADED]

Design and write and ARM Assembly Language program that will determine whether two strings stored in memory are anagrams of each other. Your program should store 1 in R0 if the strings are anagrams and 0 if they are not anagrams.