Submitted by: Raghav Taneja
(Student #) 20058783

CISC-221 Computer Architecture
Assignment #2: Data Representation

Masking, Arithmetic Operations, Expression Evaluation
Fall 2020

Instructor: Dave Dove (dove@cs.queensu.ca).

1 Introduction

This assignment continues to focus on bit-level representation of data but now includes arithmetic operations and expression evaluation as well as more masking practice. It also covers floating point representation which was not required for the RAT but you are responsible for it in a general sense on the first Quiz.

2 Logistics

You must do this assignment individually and submit your solution to the associated drop-box in OnQ under your own name, before the scheduled due date and time.

Clarifications and corrections will be posted to the course onQ site.

3 Instructions

This assignment is formatted as a MS Word Document. Enter your answers in the spaces provided. Save the document and submit as a pdf file to the course web site. Feel free to write your own C programs to test your answers. Use the pi0 – pi29 systems as you did with the previous assignment.
The ARM architecture of a Raspberry Pi is a 32-bit architecture.

PART A. A bit (pun) more masking.

Byte extraction and insertion code is useful in many contexts. Being able to write this sort of code is an important skill to foster. Suppose we number bytes in a 32 bit word from 0 (least significant bits) to 3 (most significant bits).
Write code for the following C function, which will return an unsigned value in which byte *i* of argument *x* has been replaced by byte *b.*
Complete the main() function so that it can be used to test the replace_byte function.

```
unsigned replace_byte (unsigned x, int i, unsigned char b){
/           *your code below*/

            int t = i << 3;

            unsigned m = 0xFF << 1;

            return (x & ~m) | (b << 1);
}
void main(){
//complete the main function here. (use more space if necessary)

            int x = 0x12345678;

            int result = 0x12AB5678;

            x = replace_byte(x, 0xAB, 2);

            if (x == result){

                        printf("Success\n");

            }

            return 0;

}
```

PART B. Expression evaluation.

In the following questions assume the variable *a* is a signed integer and that the machine uses two's complement representation.

Circle the answer that correctly evaluates to the source expression in each question.

1. `a / 4.`

```
    a) 1 + (a << 3) + ~a
    b) (a << 4) + (a << 2) + (a << 1)
    [✓ c)] ((a < 0) ? (a + 3) : a) >> 2
    d) ~((a >> 31) << 1)
```

2. `(a < 0) ? 1 : -1`

```
a)  ~((a | (~a + 1)) >> 31) & 1
```
[✓ b)] `~((a >> 31) << 1)`
```
c)  a >> 2
d)  ((a < 0) ? (a + 3) : a >> 2
```

(the expression (a < 0) ? 1 : -1 can be interpreted as: if (a < 0) return 1; else return -1;)

PART C. Multiplication by constants (revised)
For each of the following values of K, find ways to express int x * K without using the multiply operator, using a minimum number of shift, add, and subtract operations. Consider additions, subtractions, and shifts to have comparable cost. The maximum allowable number of Shift and Add/Sub operations are specified in the table.
See Practice Problem 2.40 in the text.

K	Shifts	Add/Subs	Expression
9	1	1	**(x << 3) + x**
14	1	2	**(x << 4) -x -x**
37	2	2	**(x << 5) + (x << 2) + x**
-110	2	4	**(x << 4) – (x << 7) + x + x**

PART D. IEEE-754 Representation

Manually encode the decimal value 37.625 into an IEEE-754 Single Precision Floating Point value. Express it as a hexadecimal value. Create and run a simple C program to verify your encoding. It’s not necessary to submit your test program.
Note: you can’t print a float value in hex using the %x placeholder. You need to recast it as an int first. You have seen examples of using a union in C to see the same variable from two different perspectives (see a value as a float and as an int in this case)

Answer: 0x**42168000**