

# CSE 140 Homework One

July 1, 2010

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on July 12, 2010 (Monday) at 4:00pm.*

## 1 Problem Set Part A

- textbook 1.3
- textbook 1.5(c)(d)
- textbook 1.6(c)(f)
- textbook 2.1(c)
- textbook 2.4(a)
- textbook 2.5(c)
- textbook 2.6(b)(d)
- textbook 2.7(b)
- textbook 2.8(a)(c)
- textbook 2.13(a)(c)
- textbook 2.14(a)
- textbook 2.15(a)(b)
- textbook 2.16(a)(c)

## 2 Problem Set Part B

### 1 (Number System Conversions)

This question concerns the positional number system, namely, conversions between positional number systems of different radices, and additions in positionally weighted codes.

(Part A)  $(0.FACE)_{16} + (0.132103)_4 = ( \quad )_8$

(Part B)  $(BAD)_{15} = ( \quad )_5$

(Part C) For all the following equations, find the minimum *non-zero* integers of  $x, y$ , and  $r$ , if any, that satisfy the equation. If you think no value combination of  $x, y$ , and  $r$  can satisfy this equation, please give a brief reasoning. (If you cannot find the minimum values of  $x, y$ , and  $r$ , please provide any values you think may satisfy for partial credit.)

$$(xy)_r = (yx)_{r+1}$$

$$\left(\frac{x}{2}y\right)_r = \left(\frac{y}{2}x\right)_{2r}$$

## 2 (Division algorithms)

This question concerns both *restoring* and *non-restoring* division algorithms for **unsigned binary numbers**, wherein the dividend is  $m$  bits and the divisor is  $n$  bits ( $m > n$ ). We have performed divisions for a number of dividends and divisors (which we are not specifying), and got as result a number of **10-bit** quotients.

(Part A) In this part, we are providing you with some information of *non-restoring* division in terms of the number of subtraction and addition operations performed during the division process. You are asked to figure out the number of subtraction and addition operations to be performed, if the *restoring* algorithm is used for the division.

| Non-restoring     |                | Restoring         |                |
|-------------------|----------------|-------------------|----------------|
| # of subtractions | # of additions | # of subtractions | # of additions |
| 6                 | 4              |                   |                |
| 5                 | 6              |                   |                |
| 4                 | 7              |                   |                |
| 5                 | 5              |                   |                |

(Part B) Assume that only the number of **subtraction** operations in the *non-restoring* algorithm is known. Is it possible to figure out the number of additions in the *restoring* algorithm? Is it possible to figure out the number of subtractions in the *restoring* algorithm? Please give a brief explanation for both of your answers.

(Part C) Assume that only the number of **addition** operations in the *non-restoring* algorithm is known. Is it possible to figure out the number of additions in the *restoring* algorithm? Is it possible to figure out the number of subtractions in the *restoring* algorithm? Please give a brief explanation for both of your answers.

### 3 (Floating Point Representation)

This question concerns the extension of floating point representation for *balanced ternary* numbers. In an  $n$ -bit balanced ternary number, the  $i^{th}$  bit has a weight of  $3^i$  ( $0 \leq i \leq n - 1$ ). Each bit can have a value of 0, 1, or  $\bar{1}$ , with  $\bar{1}$  denoting -1. Accordingly, the number **5** will be represented as  $1 \bar{1} \bar{1}$ , while the number **-7** will be represented as  $\bar{1} 1 \bar{1}$ .

In this question, we define a floating point representation for *balanced ternary* numbers, as shown in the following formula:

$$N = 3^E * (0.M)$$

with both **E** and **M** being *balanced ternary* numbers. The exponent **E** is an 8-bit value, while the mantissa **M** is a 24-bit value. In the normalized format, the most significant bit of **M** cannot be 0. As an example, the normalized representations of the numbers **5** and **-7** are shown as follows:

| Value | Exponent | Mantissa                                           |
|-------|----------|----------------------------------------------------|
| 5     | 00000010 | $\bar{1}\bar{1}\bar{1}00000 \ 00000000 \ 00000000$ |
| -7    | 00000010 | $\bar{1}1\bar{1}00000 \ 00000000 \ 00000000$       |

(Part A) Convert  $-19_{10}$  to this *balanced ternary* floating point number representation, by filling in the following table.

| Value | Exponent | Mantissa |
|-------|----------|----------|
| -19   |          |          |

(Part B) In this normalized floating point representation of *balanced ternary* numbers, what is the **largest positive** value of the mantissa and what is the **smallest positive** value of the mantissa? (*Hint: both values should be smaller than 1, as the ternary point is to the left of the most significant bit of M.*) Please convert both values to a decimal representation.

(Part C) In this normalized representation, what is the total number of distinct values that the mantissa can represent? Please give a brief reasoning.

(Part D) Would it be possible to change the outlined representation to have a hidden ‘1’ in the mantissa, such as in the IEEE754 representation? If it is possible, would it be beneficial in terms of increasing representational capability? Please give a brief reasoning.

(Part E) For the *balanced ternary* floating point numbers, addition/subtraction and multiplication/division can be performed in the same way as for floating point numbers in the IEEE754 single precision format. In these algorithms, the last step is always a possible normalization of the resulting exponent and mantissa. We list below two sets of candidate normalization operations to **E** and **M** for the last normalization step of the algorithms. However, only some of them can actually occur in the computation.

| Operations on $M$                      | Operations on $E$          |
|----------------------------------------|----------------------------|
| “<<” = left shift, “>>” = right shift. |                            |
| A. $M << 1$                            | a. $E + 1$                 |
| B. $M << 2$                            | b. $E + 2$                 |
| C. $M << 3$                            | c. $E + 3$                 |
| D. $M << n$ , where $n > 3$            | d. $E + n$ , where $n > 3$ |
| E. $M >> 1$                            | e. $E - 1$                 |
| F. $M >> 2$                            | f. $E - 2$                 |
| G. $M >> 3$                            | g. $E - 3$                 |
| H. $M >> n$ , where $n > 3$            | h. $E - n$ , where $n > 3$ |
| I. Do nothing                          | i. Do nothing              |

Given a pair of *balanced ternary* floating point numbers  $N_1$  and  $N_2$ , we are going to perform addition, subtraction, multiplication, and division on them. Unfortunately, the *only* piece of information we know is that the signs of these two numbers are **identical**.

For the various operations specified below, fill in each box of the table with some combination of the operations on **M** (Mantissa) and **E** (Exponent) which exactly describes all the possibilities that can occur in the last normalization step. *Please note that multiple answers may hold for each case, in which case for full credit, please supply all correct answers.*

|             | Operations on $M$ | Operations on $E$ |
|-------------|-------------------|-------------------|
| $N_1 + N_2$ |                   |                   |
| $N_1 + N_1$ |                   |                   |
| $N_1 - N_2$ |                   |                   |
| $N_1 * N_2$ |                   |                   |
| $N_1 / N_2$ |                   |                   |