

CSE 140 Homework One

August 8, 2013

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on August 15, 2013 (Thursday) at 4:00pm.*

1 Problem Set Part A

- Roth&Kinney, 6th Ed: 1.1
- Roth&Kinney, 6th Ed: 1.3
- Roth&Kinney, 6th Ed: 1.4
- Roth&Kinney, 6th Ed: 1.5
- Roth&Kinney, 6th Ed: 1.7
- Roth&Kinney, 6th Ed: 1.9
- Roth&Kinney, 6th Ed: 1.15
- Roth&Kinney, 6th Ed: 1.19
- Roth&Kinney, 6th Ed: 1.29
- Roth&Kinney, 6th Ed: 1.31

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) $(BAD)_{16} - (123321)_4 = (\quad)_8$

(Part B) $(uvw)_3 \times (81)_{10} = (\quad)_3$ (u, v, w are three variables each denoting a single digit value. Your answer to this part of the question can contain any or all of these three variables.)

(Part C) $(29)_{x^2} = (\frac{x}{2}1)_6 + (130)_x \quad x = ?$

(Part D) $(a4)_{r^2} = (2b1)_r \quad a, b, r = ?$

2 (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	$\bar{1}\bar{1}100000$ 00000000 00000000
-7	00000010	$\bar{1}\bar{1}100000$ 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 “<<” = left shift, “>>” = right shift.	Operations on E
A. $M \ll 1$	a. $E + 1$
B. $M \ll 2$	b. $E + 2$
C. $M \ll 3$	c. $E + 3$
D. $M \ll n$, where $n > 3$	d. $E + n$, where $n > 3$
E. $M \gg 1$	e. $E - 1$
F. $M \gg 2$	f. $E - 2$
G. $M \gg 3$	g. $E - 3$
H. $M \gg n$, 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		

3 (Hamming Codes) As you hopefully remember, standard Hamming codes (detailed below), are single-bit error correction, double bit error-detection codes. By using 3 parity equations and the associated correction equations, Hamming codes manage to correct single bit errors (and detect double bit errors). While you were impressed by the ease with which the failing bit was identified by looking at $C_0C_1C_2$ as an unsigned integer index of the failing bit, you quickly realize that other ways of pinpointing the failing bit are perfectly acceptable in a digital design implementation since the hardware does not so much care about the nice visual aid that we all enjoyed in class.

Armed with this perspective, you decide to investigate whether alternative formats of the parity and correction equations can do just as effective of a job at uniquely identifying the location of a single bit error (once you are satisfied that the set of equations does have the property of single-bit error correction, you will of course delegate responsibility to another guy in your company to deal with the subsequent implementation). Below are some groups of equations which you have been mulling over. In case you have forgotten, we give you below the standard Hamming bit positions and associated equations.

Bit	m_0	m_1	m_2	p_0	m_3	p_1	p_2
Position	7	6	5	4	3	2	1

$$\begin{array}{ll}
 p_0 = m_0 \oplus m_1 \oplus m_2 & C_0 = p_0 \oplus m_0 \oplus m_1 \oplus m_2 \\
 p_1 = m_0 \oplus m_1 \oplus m_3 & C_1 = p_1 \oplus m_0 \oplus m_1 \oplus m_3 \\
 p_2 = m_0 \oplus m_2 \oplus m_3 & C_2 = p_2 \oplus m_0 \oplus m_2 \oplus m_3
 \end{array}$$

(Part A) For a start, here are **two sets** of non-standard correction equations. For each set of equations, determine whether or not it is possible to perform single-bit error correction using only the result of the correction equations. Explain why it is or is not possible.

$$\begin{array}{llllllll}
 \text{Set A} & C_0 & = & m_0 & & \oplus & m_2 & \oplus & m_3 & & \oplus & p_1 \\
 & C_1 & = & m_0 & & \oplus & m_2 & & & \oplus & p_0 & \oplus & p_2 \\
 & C_2 & = & m_0 & \oplus & m_1 & & & \oplus & m_3 & & \oplus & p_1 & \oplus & p_2
 \end{array}$$

$$\begin{array}{llllllllll}
 \text{Set B} & C_0 & = & m_0 & & & & \oplus & p_0 & \oplus & p_1 & \oplus & p_2 \\
 & C_1 & = & & & m_2 & \oplus & m_3 & \oplus & p_0 & \oplus & p_1 \\
 & C_2 & = & & m_1 & \oplus & m_2 & & & & \oplus & p_1 & \oplus & p_2
 \end{array}$$

(Part B) The following **two sets** of equations only contain two correction equations. Explain whether or not either of these equation sets can be used to perform single-bit error correction.

$$\begin{array}{lcl} \text{Set A} & C_0 & = m_0 \oplus m_1 \oplus m_3 \oplus p_0 \oplus p_2 \\ & C_1 & = m_0 \oplus m_2 \oplus m_3 \oplus p_1 \oplus p_2 \end{array}$$

$$\begin{array}{lcl} \text{Set B} & C_0 & = m_1 \oplus m_2 \oplus m_3 \oplus p_0 \oplus p_2 \\ & C_1 & = m_0 \oplus p_0 \oplus p_1 \oplus p_2 \end{array}$$

(Part C) The following **two sets** of equations utilize four correction equations. For each set of equations, explain whether or not it is possible to perform single-bit error correction using only the result of the correction equations.

$$\begin{array}{lcl} \text{Set A} & C_0 & = m_0 \oplus m_2 \oplus p_0 \oplus p_1 \\ & C_1 & = m_0 \oplus m_1 \oplus p_0 \oplus p_1 \\ & C_2 & = m_1 \oplus m_3 \oplus p_1 \oplus p_2 \\ & C_3 & = m_0 \oplus m_2 \oplus m_3 \oplus p_0 \end{array}$$

$$\begin{array}{lcl} \text{Set B} & C_0 & = m_0 \oplus m_3 \oplus p_0 \\ & C_1 & = m_0 \oplus m_2 \oplus p_1 \\ & C_2 & = m_0 \oplus m_1 \oplus p_2 \\ & C_3 & = m_1 \oplus m_2 \oplus m_3 \end{array}$$

(Part D) Using these modified parity and correction equations, position m_0, m_1, m_3 , and p_1 so that the bit-string $C_0C_1C_2$ specifies the location of any 1-bit error.

$$\begin{aligned} p_0 &= m_0 \oplus m_2 \oplus m_3 & C_0 &= m_0 \oplus m_3 \oplus p_1 \oplus p_2 \\ p_1 &= m_1 \oplus m_2 \oplus m_3 & C_1 &= m_0 \oplus m_1 \oplus m_2 \oplus p_2 \\ p_2 &= m_0 \oplus m_1 \oplus m_2 & C_2 &= m_0 \oplus m_2 \oplus m_3 \oplus p_0 \end{aligned}$$

Bit		p_2			m_2		p_0
Position	7	6	5	4	3	2	1

(Part E) Here the bit positions have been negated. Specify C_3, C_2, C_1 , and C_0 such that $C_0C_1C_2C_3$ is the 2's complement representation of the position of any single-bit error, and 0 if there is no error.

Bit	m_0	m_1	m_2	p_0	m_3	p_1	p_2
Position	-7	-6	-5	-4	-3	-2	-1

$$C_0 =$$

$$C_1 =$$

$$C_2 =$$

$$C_3 =$$