

Quinn Roemer

CISP - 440

Assignment 0.2

9/6/2018

## Part 0 - Signed and Unsigned Binary Numbers.

### Description:

The goal for this section of the assignment was to perform several conversion on signed and unsigned numbers. This included converting decimal numbers to signed or unsigned binary numbers and finding the binaries 2's complement.

### Problem 1:

What is the range of unsigned 16 bit numbers in decimal and in binary?

The range of a sixteen digit binary would be from 0 to  $(2^{16}) - 1$ . Which would equal 65535.

Thus in decimal the range would be:

**0 -> 65535** which is a total of 65536 different combinations.

In binary that would be:

**0000 0000 -> 1111 1111** which is a total of 65536 different combinations.

### Problem 2:

Convert decimal 101 to an 8-bit binary number.

To perform this conversion we will refer to the following chart and add up the necessary values:

128	64	32	16	8	4	2	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$64 + 32 + 4 + 1 = 101$$

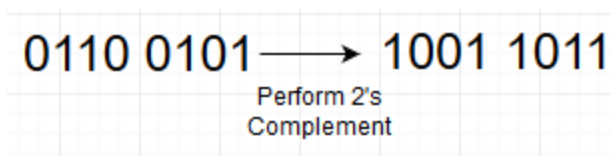
Thus the answer would be:

**0110 0101**

**Problem 3:**

Convert decimal -101 as an 8 bit 2's complement number.

Since we already know the decimal representation for positive 101 we can simply perform the 2's complement operation on the existing binary number.



Thus the answer would be:

**1001 1011**

**Problem 4:**

What is the decimal value of the 2's complement number 11011001?

First we recognize that the binary number is negative because the left-most bit is 1. To perform this conversion we can simply treat the last bit as negative and subtract the value of all other bits from it to get the answer.

$$-128 + 64 + 16 + 8 + 1 = -39$$

Thus the answer would be:

**-39**

**Problem 5:**

What are the largest and smallest decimal values representable in 2's complement using 8 bits?

The largest value in 2's complement 8 bit that can be represented without error is 0111 1111 which would be:

$$64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$$

The smallest value in 2's complement 8 bit that can be represented without error is 1000 0000 which would be:

$$-128 + 0 = -128$$

Thus the answers would be:

**Largest = 127 & Smallest = -128**

### Problem 6:

What are the largest and smallest decimal values representable in 2's complement using 32 bits?

The largest value in 2's complement 32 bit that can be represented without error is 0111 1111 1111 1111 1111 1111 1111 1111 which would be:

2,147,483,647

The smallest value in 2's complement 32 bit that can be represented without error is 1000 0000 0000 0000 0000 0000 0000 0000 which would be:

-2,147,483,648

Thus the answers would be:

**Largest = 2,147,483,647 & Smallest = -2,147,483,648**

### Problem 7:

How many values can be represented with 42 bits using 2's complement?

Because the left-most bit of this number will represent the sign of the recorded value we can safely say that the number of values that this number can represent will equal the range of a 41 bit binary number. We can find this as follows.

0 to  $2^{41}$

Thus the answer would be:

**0 -> 2,199,023,255,552** which is a total of **2,199,023,255,553** combinations

**Problem 8:**

Convert decimal 48 to an 8 bit binary number.

To perform this conversion we will simply look at the following chart and determine the bit pattern.

128	64	32	16	8	4	2	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$32 + 16 = 48$$

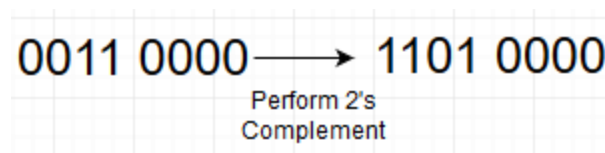
Thus the answer would be:

**0011 0000**

**Problem 9:**

Convert decimal -48 to an 8 bit 2's complement number.

Since we already know the decimal representation for positive 48 we can simply perform the 2's complement operation on the existing binary number.



Thus the answer would be:

**1101 0000**

**Problem 10:**

Convert decimal 38 to an 8 bit binary number.

To perform this conversion we will simply look at the following chart and determine the bit pattern.

128	64	32	16	8	4	2	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$32 + 4 + 2 = 38$$

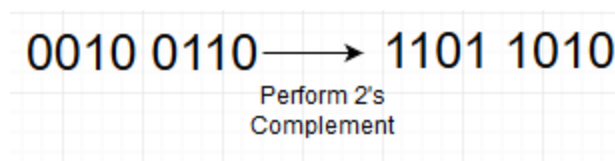
Thus the answer would be:

**0010 0110**

### Problem 11:

Convert decimal -38 to an 8 bit 2's complement number.

Since we already know the decimal representation for positive 38 we can simply perform the 2's complement operation on the existing binary number.



Thus the answer would be:

**1101 1010**

### Problem 12:

What is the decimal value of the 8 bit 2's complement number 1101 1001?

First we recognize that the binary number is negative because the left-most bit is 1. To perform this conversion we can simply treat the last bit as negative and subtract the value of all other bits from it to get the answer.

$$-128 + 64 + 16 + 8 + 1 = -39$$

Thus the answer would be:

**-39**

### **Problem 13:**

What is the decimal value of the 8 bit 2's complement number 1001 1000?

First we recognize that the binary number is negative because the left-most bit is 1. To perform this conversion we can simply treat the last bit as negative and subtract the value of all other bits from it to get the answer.

$$-128 + 16 + 8 = -104$$

Thus the answer would be:

**-104**

### **Problem 14:**

What is the decimal value of the 16 bit 2's complement number 1101 1001 1001 1000?

First we recognize that the binary number is negative because the left-most bit is 1. To perform this conversion we can simply treat the last bit as negative and subtract the value of all other bits from it to get the answer.

$$-32,768 + 16,384 + 4096 + 2048 + 256 + 128 + 16 + 8 = -9832$$

Thus the answer would be:

**-9832**

### **Problem 15:**

What is the decimal value of the 16 bit 2's complement number 0101 1001 1001 1000?

First we recognize that the binary number is positive because the left-most bit is 0. To perform this conversion we can simply treat the last bit as nothing and add all of the values of all other bits to the right to get the answer.

$$16,384 + 4096 + 2048 + 256 + 128 + 16 + 8 = 22,936$$

Thus the answer would be:

**22,936**

**Problem 16:**

What is the decimal value of the 8 bit 2's complement number 0000 0101?

First we recognize that the binary number is positive because the left-most bit is 0. To perform this conversion we can simply treat the last bit as nothing and add all of the values of all other bits to the right to get the answer.

$$4 + 1 = 5$$

Thus the answer would be:

**5**



## Part 1 - Floating Point Formats.

### Description:

The goal for this section of the assignment was to answer questions about floating point numbers and how they are packed. In addition we were to perform several conversion between decimal numbers and a float.

### Problem 1:

The IEEE standard uses excess 127 format for exponents. What would be the decimal exponent value of 0110 1011?

First, let us convert 0110 1011 to decimal.

0110 1011 equals 107 in decimal.

To get the decimal component we must subtract 127 from 107.

$$107 - 127 = -20$$

Thus the answer would be:

**-20**

### Problem 2:

The IEEE standard uses excess 127. Would excess 128 be better? Explain.

No excess 128 would reduce the possible positive numbers that can be represented in this 8 bit form. By having an excess of 127 the possible negative and positive numbers that can be represented are pretty much equal. This is the the best excess to use for this size of binary number.

### Problem 3:

What is the range of exponent values possible using 6 exponent bits? What is the value of k in this "Excess k" format using the IEEE excess mapping scheme?

First let us discover the range of a 6 bit number.

0 to  $(2^6) - 1$

Thus the range would be:

0 -> 63 with a total of 64 possible combinations.

Using the excess mapping scheme format from IEEE we know that the excess for a 6 bit binary number would be:

01 1111 which equals 31.

Thus the answers are:

**Range = 0 -> 63** with a possible **64** combinations

**Excess = 01 1111** which equals **31** using IEEE format

#### Problem 4:

Consider the following floating point format: 1 sign bit, 4 exponent bits in excess 7 format, and 5 mantissa bits. What are the decimal values represented by the following:

1 1011 11100

0 0100 01011

1 1111 11000

0 0110 01001

We will refer to this chart in the following calculations:

.1	.01	.001	.0001	.00001	.000001	.0000001	.00000001
.5	.25	.125	.0625	.03125	.015625	.0078125	.00390625

For the first number we look at the sign bit and determine it is negative. We then look at the exponent bits and determine it is 11 in decimal. We then subtract 7 from 11 and get 4. Thus we have:

-1.11100 \*  $2^4$  which equals:

-11110.0 which equals:  
-30.0

For the second number we look at the sign bit and determine it is positive. We then look at the exponent bits and determine it is 4 in decimal. We then subtract 7 from 4 and get -3. Thus we have:

1.01011 \*  $2^{-3}$  which equals:  
0.00101011 which equals:  
0.125 \* .03125 \* .0078125 \* .00390625 which equals:  
0.00000011920929

For the third number we look at the sign bit and determine it is negative. We then look at the exponent bits and determine it is 15 in decimal. We then subtract 7 from 15 and get 8. Thus we have:

-1.11000 \*  $2^8$  which equals:  
-111000000.0 which equals:  
-448.0

For the fourth number we look at the sign bit and determine it is positive. We then look at the exponent bits and determine it is 6 in decimal. We then subtract 7 from 6 and get -1. Thus we have:

1.01001 \*  $2^{-1}$  which equals:  
0.101001 which equals:  
0.5 \* .125 \* .015625 which equals:  
0.0009765625

Thus the answers are:

**First: -30.0**

**Second: 0.00000011920929**

**Third: -448.0**

**Fourth: 0.0009765625**

**Problem 5:**

Convert decimal 85.0 to this format.

The decimal 85.0 can be converted to a float via the following processes.

Convert decimal to binary.

1010101.0

Convert to scientific notation.

$1.0101010 \times 2^6$

Convert exponent to binary and add 7.

$6 + 7 = 13 = 1101$

Thus the packed float would look like: (note, we lost some precision)

Sign	Exponent	Mantissa
0	1101	01010

**Problem 6:**

Convert decimal 25.0 to this format.

The decimal 25.0 can be converted to a float via the following processes.

Convert decimal to binary.

11001.0

Convert to scientific notation.

$1.10010 \times 2^4$

Convert exponent to binary and add 7.

$4 + 7 = 11 = 1011$

Thus the packed float would look like:

Sign	Exponent	Mantissa
0	1011	10010

### Problem 7:

Convert decimal 2.25 to this format.

The decimal 2.25 can be converted to a float via the following processes.

Convert decimal to binary.

10.01

Convert to scientific notation.

$1.001 \times 2^1$

Convert exponent to binary and add 7.

$1 + 7 = 8 = 1000$

Thus the packed float would look like:

Sign	Exponent	Mantissa
0	1000	001

### Problem 8:

Convert decimal 7.125 to this format.

The decimal 7.125 can be converted to a float via the following processes.

Convert decimal to binary.

111.001

Convert to scientific notation.

$$1.11001 * 2^2$$

Convert exponent to binary and add 7.

$$2 + 7 = 8 = 1001$$

Thus the packed float would look like:

Sign	Exponent	Mantissa
0	1001	11001

### Problem 9:

Consider the following floating point format: 1 sign bit, 4 mantissa bits, and 3 exponent bits in excess 4 format. What are the decimal values represented by the following:

1 1011 111  
 0 0100 010  
 1 1111 110  
 0 0110 010

We will refer to this chart in the following calculations:

.1	.01	.001	.0001	.00001	.000001	.0000001	.00000001
.5	.25	.125	.0625	.03125	.015625	.0078125	.00390625

For the first number we look at the sign bit and determine it is negative. We then look at the exponent bits and determine it is 7 in decimal. We then subtract 4 from 7 and get 3. Thus we have:

-1.1011 \* 2<sup>3</sup> which equals:

-1101.1 which equals:

-13.5

For the second number we look at the sign bit and determine it is positive. We then look at the exponent bits and determine it is 2 in decimal. We then subtract 4 from 2 and get -2. Thus we have:

$1.0100 * 2^{-2}$  which equals:

0.010100 which equals:

$0.25 * .0625$  which equals:

0.015625

For the third number we look at the sign bit and determine it is negative. We then look at the exponent bits and determine it is 5 in decimal. We then subtract 4 from 5 and get 1. Thus we have:

$-1.1111 * 2^1$  which equals:

-11.111 which equals:

$-3.5 * .25 * .125$  which equals:

-3.015625

For the fourth number we look at the sign bit and determine it is positive. We then look at the exponent bits and determine it is 2 in decimal. We then subtract 4 from 2 and get -2. Thus we have:

$1.0110 * 2^{-2}$  which equals:

0.010110 which equals:

$0.25 * .0625 * .03125$  which equals:

0.00048828125

Thus the answers are:

**First: -13.5**

**Second: 0.015625**

**Third: -3.015625**

**Fourth: 0.00048828125**

### **Problem 10:**

Convert decimal 2.25 to this format.

The decimal 2.25 can be converted to a float via the following processes.

Convert decimal to binary.

10.01

Convert to scientific notation.

$1.001 \times 2^1$

Convert exponent to binary and add 4.

$1 + 4 = 5 = 101$

Thus the packed float would look like:

Sign	Mantissa	Exponent
0	0010	101

### Problem 11:

Convert decimal 0.625 to this format.

The decimal 0.625 can be converted to a float via the following processes.

Convert decimal to binary.

0.1010000000

Convert to scientific notation.

$1.0100000000 \times 2^{-1}$

Convert exponent to binary and add 4.

$-1 + 4 = 3 = 011$

Thus the packed float would look like: (lost precision)

Sign	Mantissa	Exponent
0	0100	011



**Problem 12:**

Convert decimal 0.06640625 to this format.

The decimal 0.06640625 can be converted to a float via the following processes.

Convert decimal to binary.

0.00010001000

Convert to scientific notation.

$1.0001000 \times 2^{-4}$

Convert exponent to binary and add 4.

$-4 + 4 = 0 = 000$

Thus the packed float would look like: (lost precision)

Sign	Mantissa	Exponent
0	0001	000

**Problem 12:**

Convert decimal 5.125 to this format.

The decimal 5.125 can be converted to a float via the following processes.

Convert decimal to binary.

101.001

Convert to scientific notation.

$1.01001 \times 2^2$

Convert exponent to binary and add 4.

$2 + 4 = 6 = 110$

Thus the packed float would look like: (lost precision)

Sign	Mantissa	Exponent
0	0100	110

## Part 2 - Integer Arithmetic.

### Description:

The goal for this section of the assignment was to answer questions about about arithmetic with binary numbers. In this section we were to perform addition, multiplication, and division. In addition, we were to indicate if certain flags would be tripped.

### Problem 1:

Add  $1101\ 1100 + 0100\ 0101$  Interpret the answer as a signed 2's complement value and as an unsigned value. Indicate if the answer overflows for signed as well as unsigned interpretations.

Since this math proved difficult to write on a computer I will be inserting a picture of my work.

Handwritten work for Problem 1:

$$\begin{array}{r}
 1101\ 1100 \\
 + 0100\ 0101 \\
 \hline
 1\ 0010\ 0001
 \end{array}$$

Last carry = 1 = Overflow  
 Last 2 carries are the same  
 So signed answer is valid  
 Answer = 1 0010 0001

Thus the answer is:

**1 0010 0001**

**For unsigned we have an overflow**

**For signed the answer is valid**

### Problem 2: //I assumed subtract since or else it would be identical to the above question

Subtract  $1101\ 1100 - 0100\ 0101$  Indicate if the answer overflows for signed as well as unsigned Interpretations.

Since this math proved difficult to write on a computer I will be inserting a picture of my work.



Thus the answer is:

**0011 1011 0100 1100**

**Overflow occurs**

#### **Problem 4:**

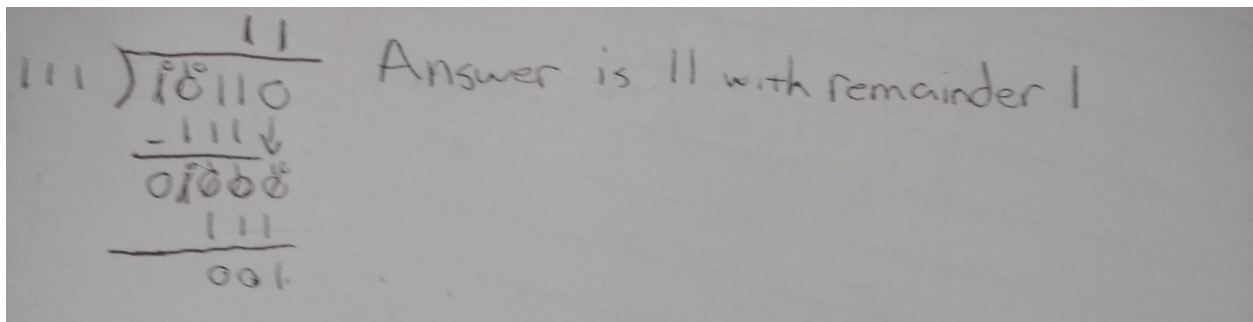
Divide the 22 binary by 7 binary using integer binary long division.

First let us convert 22 and 7 into binary.

$$22 = 1\ 0110$$

$$7 = 111$$

Since this math proved difficult to write on a computer I will be inserting a picture of my work.



Thus the answer is:

**11 with remainder 1**

#### **Problem 5:**

Divide the 1 binary by 5 binary using integer binary long division.

First let us convert 1 and 5 into binary.

$$1 = 1$$

$$5 = 101$$

Since this math proved difficult to write on a computer I will be inserting a picture of my work.

Handwritten binary long division of 1 by 101. The dividend is 1.001100110011001... and the divisor is 101. The quotient is 0.001100110011001... The division shows a repeating pattern of 001 in the quotient, indicating that the division never terminates.

This division  
Never terminates.

Thus the answer is:

**0.0011001100...**

**This division never terminates**

### Problem 6:

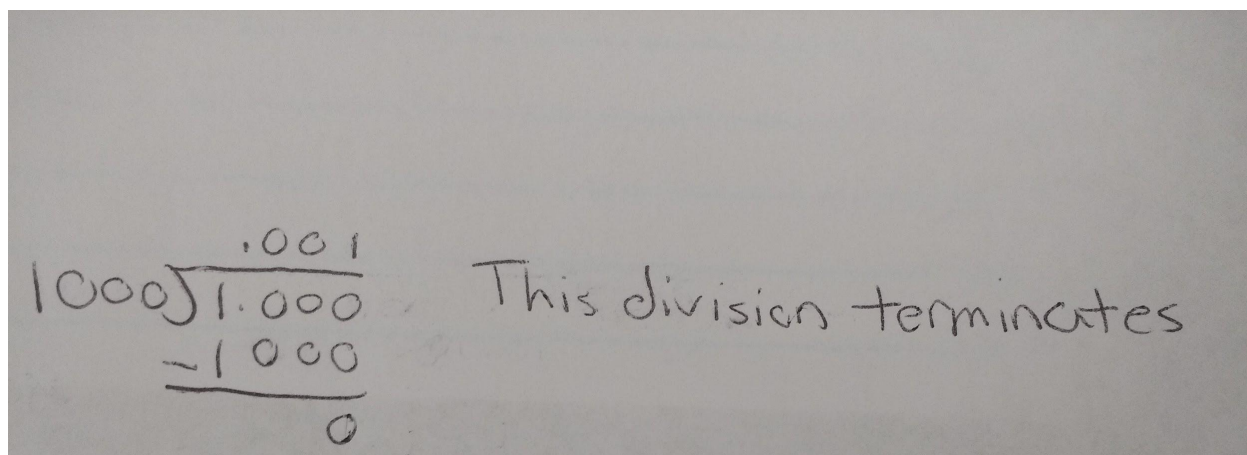
Divide the 1 binary by 7 binary using integer binary long division.

First let us convert 1 and 5 into binary.

$$1 = 1$$

$$8 = 1000$$

Since this math proved difficult to write on a computer I will be inserting a picture of my work.



Handwritten long division showing  $1000 \overline{) 1.000}$ . The quotient is  $0.001$  and the remainder is  $0$ . The text "This division terminates" is written next to the calculation.

Thus the answer is:

**0.001**

**This division terminates**

## Part 3 - Floating Point Arithmetic.

### Description:

The goal for this section of the assignment was to answer questions about arithmetic with floating point numbers. In this section we were to perform addition, multiplication, of packed float numbers.

Please note, this entire section was not taught in class as of the due date of this homework. As a result, I am unsure of some of my answers as it proved challenging to find good examples online.

### Problem 1:

Consider the following floating point format: 1 sign bit, 4 mantissa bits, and 3 exponent bits in excess 4 format.

Add

1 1011 111

0 0100 010

First let us take both numbers out of packed form and put them into scientific notation.

$$-1.1011 * 2^3$$

$$1.0100 * 2^{-2} = 0.00010100 * 2^3$$

Next we will add the mantissa of both numbers and normalize.

$$1011 + 0001 = 1100$$

Thus our answer is:

Sign	Mantissa	Exponent
1	1100	100

Multiply

1 1111 110

0 0110 010



First let us take both numbers out of packed form and put them into scientific notation.

$$-1.1111 * 2^2$$

$$1.0110 * 2^{-2}$$

Next we will multiply the mantissa of both numbers and add the exponents.

$$1111 * 0110 = 101\ 1010$$

$$2 + -2 = 0$$

Thus our answer is:

Sign	Mantissa	Exponent
0	1011	100

### Problem 2:

Consider the following floating point format: 1 sign bit, 4 mantissa bits, and 3 exponent bits in excess 4 format.

Add

$$1\ 1111\ 110$$

$$0\ 0110\ 010$$

First let us take both numbers out of packed form and put them into scientific notation.

$$-1.1111 * 2^2$$

$$1.0110 * 2^{-2} = 0.00010110 * 2^2$$

Next we will add the mantissa of both numbers and normalize.

$$1111 + 0001 = 1\ 000$$

Thus our answer is:

Sign	Mantissa	Exponent
1	1000	110

Multiply

1 1011 111

0 0100 010

First let us take both numbers out of packed form and put them into scientific notation.

$-1.1011 * 2^3$

$1.0100 * 2^{-2}$

Next we will multiply the mantissa of both numbers and add the exponents.

$1011 * 0100 = 10\ 1100$

$3 + -2 = 1$

Thus our answer is:

Sign	Mantissa	Exponent
0	1011	101

For the next 3 problems, calculate how many seconds from Noon, Sept 1 2018 until the start of the next total solar eclipse in the continental USA on April 8, 2024 in Dallas, TX.

Answer = 176,774,400 seconds

= 2046 days

= 5.61 years

### Problem 3:

How many bits are required to store this number of seconds as an unsigned integer?

To figure out how many bits it would take we can simply find the largest  $2^n$  power that is smaller than 176,774,400.

$2^{27} = 134,217,728$ .

Thus the answer is:

**28 bits**

**Problem 4:**

How many mantissa and exponent bits are required to store this same time period, but in YEARS as a float, without data loss? Use “excess zero” value for your float format. That is, do not add any excess value to the exponent.

There are 5.61 years in this time period. First we will convert this number into binary.

101.10011100001

Next, let us convert the number into scientific notation.

$1.0110011100001 * 2^2$

To successfully pack this number with no data loss we would need the following:

**13 Mantissa bits**

**2 Exponent bits**

**Problem 5:**

How many mantissa and exponent bits are required to store this same time period, but in DAYS as a float, without data loss? Use “excess zero” value for your float format.

There 2046.0 days in this time period. First we will convert this number into binary.

11111111110.0

Next, let us convert the number into scientific notation.

$1.11111111 * 2^{10}$

To successfully pack this number with no data loss we would need the following:

**9 Mantissa bits**

**4 Exponent bits**

### **Conclusion**

This assignment was difficult. The latter half of the work was not gone over in class and forced me to do my own research. However, my research did not prove to be very successful resulting in further confusion on my part. Yet, I did the best that I could. Hopefully the next assignment will have some programming. If so, I am looking forward to that!