Joe Puccio

The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2014

Problem Set #2
Issued Friday, 2/7/13; Due Thursday, 2/13/13
(hand in your work at start of the lab hour)

You may not use a calculator (or online tool) to directly perform binary/hex arithmetic or base conversion. Please **enter your answers in the space provided.**

Problem 1: Information Encoding (18 points)

In class we learned that in order to uniquely identify one of **N** equally likely symbols, $\lceil log_2 N \rceil$ bits of information must be communicated. Answer the questions below:

a) How many bits are necessary to encode an integer in the range of 0 to 127 (inclusive)?

Answer: 7

b) How many bits are necessary to encode an integer in the range of 0 to 256 (inclusive)?

Answer: 8

c) How many bits are necessary to encode an integer in the range of -32 to 31 (inclusive)?

Answer: 6

d) Consider a deck of cards with three distinct suits and the numbers 1-9. How many bits are necessary to encode a card's suit (ignoring its number value)?

Answer: 2

e) How many bits are necessary to encode a card's value (ignoring suit)?

Answer: 4

f) How many bits are necessary to encode this deck (considering both suit and number value)?

Answer: 6

Problem 2: Two's Complement Representation (24 points)

Most computers choose a particular *word length* (measured in bits) for representing integers. Many current-generation processors have word lengths of 64 bits.

a) How many different values can be encoded in a 32-bit word if all values are to be positive integers (i.e., unsigned)? Express your answer a power of 2.

Answer:

2^32

b) If the 64-bit word now uses 2's complement representation to allow negative numbers as well, does that impact the total number of distinct values that can be encoded? If so, how many many distinct values can now be encoded?

Answer:

(2^64)-1

Use a ***16-bit 2's complement*** representation to answer the following questions:

c) What is the binary representation for 12942?

Answer:

0 0 1 1 0 0 1 0 1 0 0 0 1 1 1 0

d) What is the binary representation for 1?

Answer:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

e) What is the binary representation for the most negative number (i.e., negative number with the largest magnitude) that can be represented?

Answer:

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

f) What is the decimal value for the most negative number in (e) above?

Answer:

-32768

g) What is the result in binary of negating the most negative number number in (e) above?

Answer:

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Problem 3: Hexadecimal Representation (15 points)

Since writing a long string of binary digits can be tedious, it is often convenient to use the hexadecimal notation, where a single digit in the range 0–9 or A–F is used to represent adjacent groups of 4 bits (starting from the right). Give the corresponding 8-hexit (a 'hexit' = hex digit) encoding for each of the numbers below. (*Hint:* For decimal numbers, you can either convert them first to binary using the successive division method discussed in class and then convert binary to hex, or you could directly convert decimal to hex using successive division by 16. If the decimal number is negative, first convert its positive version, then use 2's-complement negation.)

a) 6834_{ten}

Answer:

0 0 0 0 1 a b 2

b) -2048_{ten}

Answer:

100 0000 0000 which goes to

0100 0000 0000

0 0 0 0 0 8 0 0

c) $1001\ 0110\ 1010\ 1101_{two}$

Answer:

0 0 0 0 9 6 a d

d) $1101\ 1000\ 0110\ 1010\ 10_{two}$

Answer:

0 0 0 3 6 1 a a

e) -3_{ten}

Answer:

0 0 1 1 which goes to

1 1 0 1

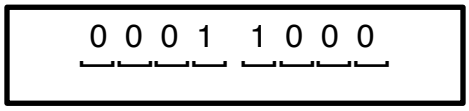
0 0 0 0 0 0 0 d

Problem 4: Binary Arithmetic (27 points)

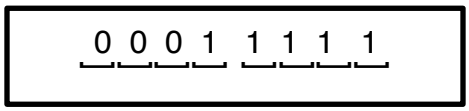
Calculate the following using ***8-bit*** 2's-complement arithmetic (which is just a fancy way of saying to do ordinary addition in base 2 ***keeping only 8 bits of your answer***). Remember that subtraction is performed by negating the second operand to form its 2's complement, and then adding it to the first operand. Give your answers in binary and decimal as indicated.

a) $24_{ten} + 31_{ten}$

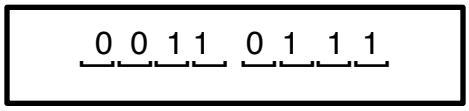
Operand 1_{two}



Operand 2_{two}



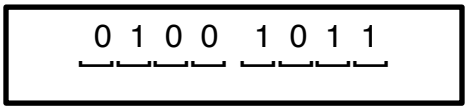
$Result_{two}$



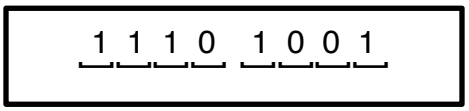
$Result_{ten}$

b) $75_{ten} - 23_{ten}$

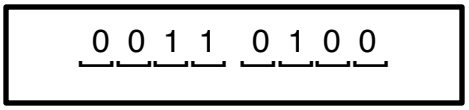
Operand 1_{two}



Operand 2_{two}



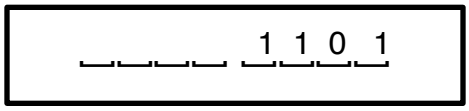
$Result_{two}$



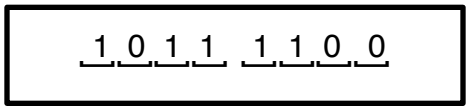
$Result_{ten}$

c) $13_{ten} - 68_{ten}$

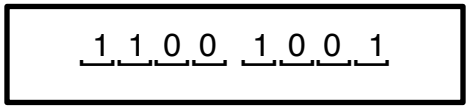
Operand 1_{two}



Operand 2_{two}



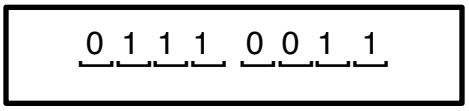
$Result_{two}$



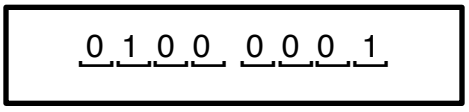
$Result_{ten}$

d) $115_{ten} + 65_{ten}$

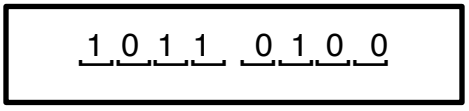
Operand 1_{two}



Operand 2_{two}



$Result_{two}$



$Result_{ten}$

e) Explain in a sentence or two what happened in part (d) above.

Because the resulting number required takes up all 8 bits, we have no bit to represent sign. Therefore, if a computer were to look at our output, it may read it in 2's complement form and assume it is a negative number which it is not meant to be.

Problem 5: Fixed-Point Binary (16 points)

Using a ***16-bit*** fixed-point binary representation, where the leftmost 8 bits are the integer part (i.e., before the binary point), and the rightmost 8 bits are the fractional part, convert the first two decimal numbers below into binary, and the remaining two from binary into decimal.

a) 82.75_{ten}

Answer:

0 1 0 1 0 0 1 0 . 1 1 0 0 0 0 0 0

-15.03125

b) -15.01325_{ten} (*Hint*: First ignore the sign and convert to binary; then perform 2's complement negation.)

Answer:

0 0 0 0 1 1 1 0 . 0 0 0 0 1 0 0 0

1 1 1 1 0 0 0 1 . 1 1 1 1 0 1 1 1

1 1 1 1 0 0 0 0 . 1 1 1 1 1 0 0 0

c) $0110\ 1101\ .\ 0111\ 0100_{two}$

Answer:

109.453125

d) 1111 1110 . 0010 0000two

Answer:

-1.875