CSCI 250: EVERYTHING IS A NUMBER PART II

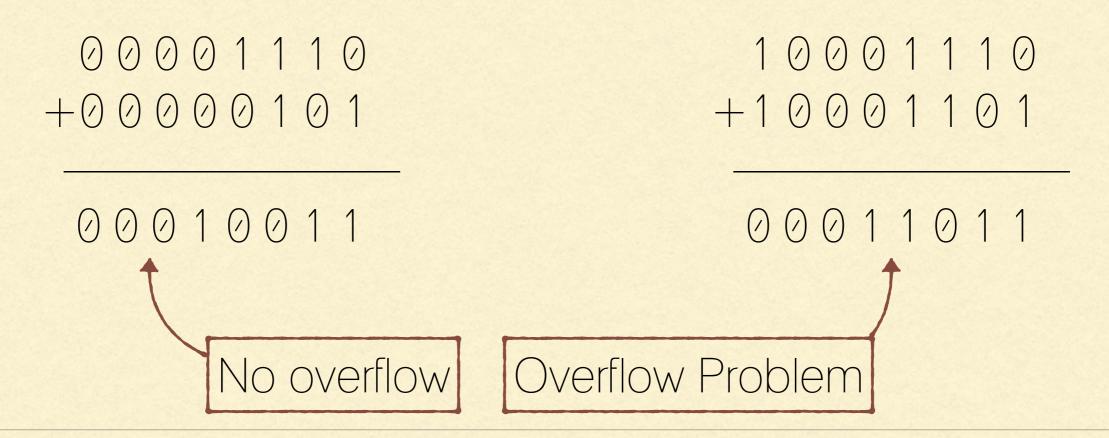
Ayman Hajja, PhD

BINARY ADDITION

Four possible binary addition combinations:

ADDING BINARY — OVERFLOW

More complicated binary additions are also done just like we do decimal addition. Let's do an 8-bit unsigned binary addition:



REPRESENTING NEGATIVE VALUES. WAY 1: SIGN & MAGNITUDE

- Use one bit (the left-most/most-significant) to indicate the sign.
 - "0" indicates a positive integer,
 - "1" indicates a negative integer.
- Question: With 8-bit sign-magnitude representation, what positive integers can be represented and what negative integers can be represented?

REPRESENTING NEGATIVE VALUES. WAY 1: SIGN & MAGNITUDE

- Use one bit (the left-most/most-significant) to indicate the sign.
 - "0" indicates a positive integer,
 - "1" indicates a negative integer.
- Question: With 8-bit sign-magnitude representation, what positive integers can be represented and what negative integers can be represented?

-127 ... 0 ... 127

WAY 1: SIGN & MAGNITUDE

- There are several problems with sign-magnitude. It works well for representing positive and negative integers (although the two zeros are bothersome). But it does not work well in computation.
- A good representation method (for integers or for anything) must not only be able to represent the objects of interest, but must also support operations on those objects.
- (4 bit binary) Can the "binary addition algorithm" be used with sign-magnitude representation? Try adding +7 with -4?

The answer is no

- The one's complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number (swapping 0s for 1s and vice versa).
 - The number 7 is represented as: 00000111
 - The number -7 is represented as: 11111000
- The one's complement of the number then behaves like the negative of the original number in some arithmetic operations.

- To convert a positive decimal number to one's complement, we simply convert the number to its unsigned binary representation:
 - 3 is represented as 00000011 (given we're using 8 bits)
 - Note that we will need to know the number of bits we're converting to
- To convert a negative decimal number to one's complement representation, we:
 - 1. Convert its magnitude to an unsigned binary
 - 2. Swap the 0s for 1s and vice versa

 (1 byte) Using one's complement representation. Represent the decimal 9 in binary:

 (1 byte) Using one's complement representation. Represent the decimal 9 in binary:

Since 9 is positive, we simply convert it to its unsigned binary representation:

0000 1001

 (1 byte) Using one's complement representation. Represent the decimal -11 in binary:

 (1 byte) Using one's complement representation. Represent the decimal -11 in binary:

Since -11 is negative, we first need to convert its magnitude to its unsigned binary representation:

11 (decimal) is 0000 1011 (unsigned binary)

Next, we swap the 0s for 1s and vice versa:

The answer is 1111 0100

- (1 byte) Using one's complement representation. Represent the decimal 20 in binary:
 - A. 00001010
 - B. 10100
 - C. 00010100
 - D. Both B & C are correct
 - E. 11101011

- (1 byte) Using one's complement representation. Represent the decimal 20 in binary:
 - A. 00001010
 - B. 10100
 - C. <u>00010100</u>
 - D. Both B & C are correct
 - E. 11101011

- (1 byte) Using one's complement representation. Represent the decimal -10 in binary:
 - A. 00001010
 - B. 11010
 - C. 10001010
 - D. Both A & C are correct
 - E. 11110101

- (1 byte) Using one's complement representation. Represent the decimal -10 in binary:
 - A. 00001010
 - B. 11010
 - C. 10001010
 - D. Both A & C are correct
 - E. <u>11110101</u>

- To convert from one's complement representation to decimal we look at the MSB and determine whether it's a positive number or negative: <u>0</u>110 (positive number) <u>1</u>010 (negative number)
 - 1. If it's a positive number, we simply treat it as an unsigned binary and convert it to its decimal representation
 - 2. If it's a negative number, we swap the 0s with 1s (and vice versa), then convert the result to decimal treating it as an unsigned binary, and finally add the negative sign

• (1 byte) Using one's complement representation. Convert 00001011 to decimal:

 Using one's complement representation. Convert 00001011 to decimal:

Since the most significant bit is 0, that means the number is positive

Therefore, we simply treat it as an unsigned binary and convert it to decimal:

00001011 (in one's complement) is 11 in decimal

 Using one's complement representation. Convert 11111011 to decimal:

 Using one's complement representation. Convert 11111011 to decimal:

Since the most significant bit is 1, that means the number is negative; therefore, we swap the 0s for 1s (and vice versa):

00000100

Then we treat it as an unsigned binary and convert it to decimal: 00000100 (in unsigned binary) is 4 in decimal Finally, we add the negative sign:

The final answer is -4

- Using one's complement representation. Convert 11110000 to decimal:
 - A. 20
 - B. 15
 - C. -15
 - D. -20
 - E. None of the answers above

- Using one's complement representation. Convert 11110000 to decimal:
 - A. 20
 - B. 15
 - C. <u>-15</u>
 - D. -20
 - E. None of the answers above

WAY 2: ONE'S COMPLEMENT ITS RANGE

Question: With a nibble (4-bit) one's complement representation, what positive integers can be represented and what negative integers can be represented?

WAY 2: ONE'S COMPLEMENT ITS RANGE

Question: With a nibble (4-bit) one's complement representation, what positive integers can be represented and what negative integers can be represented?

Similar to sign-magnitude representation: -7 ... 0 ... 7

WAY 2: ONE'S COMPLEMENT ITS RANGE

Question: With a nibble (4-bit) one's complement representation, what positive integers can be represented and what negative integers can be represented?

Similar to sign-magnitude representation: -7 ... 0 ... 7

We still have the two representations of zero problem:

1111 1111 0000 0000

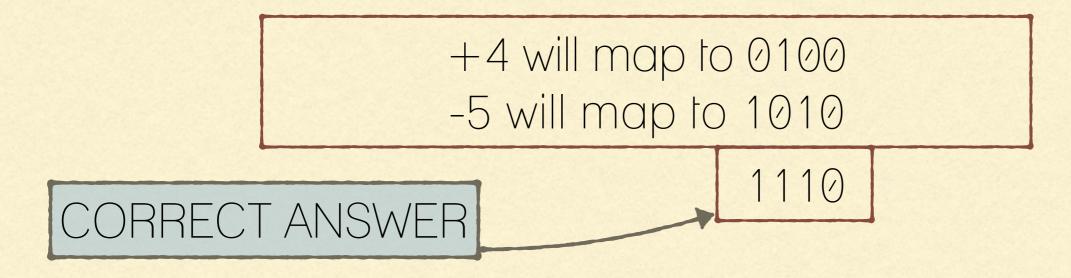
 (4 bit binary) Can the "binary addition algorithm" be used with one's complement representation? Try adding +4 with -5?

+4 will map to 0100

-5 will map to 1010

If we add them together, we should get -1

 (4 bit binary) Can the "binary addition algorithm" be used with one's complement representation? Try adding +4 with -5?



If we add them together, we should get -1

 (4 bit binary) Can the "binary addition algorithm" be used with one's complement representation? Try adding +7 with -4?

+7 will map to 0111

-4 will map to 1011

If we add them together, we should get 3

 (4 bit binary) Can the "binary addition algorithm" be used with one's complement representation? Try adding +7 with -4?

+7 will map to 0111
-4 will map to 1011

INCORRECT ANSWER

0010

If we add them together, we should get 3

When adding with one's complement:

If the carry extends past the end of the bit sequence, then one bit must be added to the result

- First let's detect which of these examples will cause an overflow!
 - Remember that overflow means the resulting answer cannot be presented using the number of bits we have

Example 1

1100
+0111

1000 +0111 1000 +1001

- The two's complement of binary sequence is equivalent to taking the one's complement and then adding one to it.
 - The number 7 is represented as: 00000111
 - (1's complement) The number -7 is represented as: 111111000
 - (2's complement) The number -7 is represented as: 111111001

(4 bits: nibble) Using two's complement representation.
 Represent the decimal 3 in binary:

(4 bits: nibble) Using two's complement representation.
 Represent the decimal 3 in binary:

Since 3 is positive, we simply convert it to its unsigned binary representation:

0011

(4 bits: nibble) Using two's complement representation.
 Represent the decimal -6 in binary:

(4 bits: nibble) Using two's complement representation.
 Represent the decimal -6 in binary:

Since -6 is negative, we first need to convert its magnitude to its unsigned binary representation:

6 (decimal) is 0110 (unsigned binary)

Next, we swap the 0s for 1s and vice versa:

1001

And finally we add 1:

The final answer is 1010

• (4 bits: nibble) Convert from the two's complement 1101 to decimal:

 (4 bits: nibble) Convert from the two's complement 1101 to decimal:

If the number is negative, we first swap the 0s with 1s (and vice versa): 1101 becomes 0010

Now we add 1 to the resulting sequence:

0011

And finally we convert the sequence to decimal and add the negative sign:

The final answer is -3

- Two special numbers in the two's complement representation:
 - A. 0000 (zero). If we get the two's complement for 0000, then we would end up with 0000; which means that we only have one zero in two's complement representation
 - B. 1000. If we get the two's complement for 1000, we would also end up with 1000, which is 8 in unsigned binary. The sequence 1000 in two's complement is equivalent to -8 in decimal

- Two special numbers in the two's complement representation:
 - A. 0000 (zero). If we get the two's complement for 0000, then we would end up with 0000; which means that we only have one zero in two's complement representation
 - B. 1000. If we get the two's complement for 1000, we would also end up with 1000, which is 8 in unsigned binary. The sequence 1000 in two's complement is equivalent to -8 in decimal

The range of values for a 4-bit two's complement is:

 (4 bit binary) Can the "binary addition algorithm" be used with two's complement representation? Try adding +7 with -4?

The answer is Yes, & without any modifications

- First let's detect which of these examples will cause an overflow!
 - Remember that overflow means the resulting answer cannot be presented using the number of bits we have

1100 +0111

1000 +0111 1000 +1001

Example 4

0100
+0101

There's a quick way to determine if the addition will cause an overflow of not:

If the carry-in to the sign bit is not equal to the carry-out from the sign bit then there's an overflow

