Computer Science 240
Binary Operations
Assignment for Lab 2

Submit hardcopy of completed exercises at the beginning of lab.

Also, submit a hardcopy of your answers from lecture Assignment Zero for the "Make Nothing from Something" section (solutions to some simple bit puzzles).

Representation of numbers
The following is an overview of signed vs. unsigned representation of numbers (which you will also discuss in lecture). Read carefully before doing the problems at the end.

Given n bits, 2^n different values can be represented.

Using **unsigned representation**, the range of values is $0 \rightarrow 2^n - 1$ (all values are positive. Zero is considered a positive number, as well)

Using **signed representation**, the range of values is $-2^{n-1} \rightarrow 2^{n-1} - 1$, (half the values are negative, half are positive)

Two's Complement is a form of signed representation:

The ost significant /leftmost bit is 0 if the number is positive, and 1 if it is negative.

Example: given a fixed number of 4 bits:
1000_2 is negative.
0111_2 is positive.

Overflow
Given a fixed number of n available bits:
Overflow occurs if a value cannot fit in n bits.

Example: given 4 bits:
The largest negative value we can represent is -8_{10} (1000_2)
The largest positive value we can represent is $+7_{10}$ (0111_2)

Overflow when Adding

An overflow occurs when adding two n-bit numbers if the result will not fit in n bits.

When using unsigned representation, a carry-out from the most significant pair of bits is the same as an overflow.

However, when using two's complement representation, a carry-out does **not** necessarily mean that is also an overflow.

An overflow can be detected when:
-Two positive numbers added together yield a negative result, or
-Two negative numbers added together yield a positive result.

An overflow cannot result if a positive and negative number are added.

Example: given 4 bit representation:

$$\begin{array}{rl} 0111_2 & (7_{10}) \\ \underline{+\ 0001_2} & (7_{10}) \\ 1000_2 & (-8_{10})\text{, which is an overflow} \quad \textbf{NOTE:}\text{ there is not a carry-out!} \end{array}$$

Example: given 4 bits,

$$\begin{array}{r} 1001_2\ (-7_{10}) \\ \underline{+\ 1111_2\ (-1_{10})} \\ 1\ 1000_2\ (-8_{10}) \end{array}$$ no overflow, even though there **is** a carry-out

For each of the following problems, perform addition on the given values (assume **two's complement** representation). Indicate whether there is a carry-out or an overflow for each addition.

For the first 2 calculations, assume **16- bit representation.** Do the calculation using the binary values.

Then, convert the result to hexadecimal notation. To convert, divide the 16 binary digits of the result into groups of 4, and translate each group to the corresponding hexadecimal value. Note that if there is a carry-out, that is the 17th bit, and it is not used in result or in the hexadecimal translation!

Carry-Out? Overflow?

1.

$$\begin{array}{r} 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1_2 \\ +\ \underline{1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1_2} \end{array}$$

Result in binary :

Result in hexadecimal:

Carry-Out? Overflow?

2.

$$\begin{array}{r} 0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0_2 \\ \underline{+\ \ 0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1_2} \end{array}$$

Result in binary :

Result in hexadecimal:

Now, assume 32-bit representation, using hexadecimal notation, and specify result in hexadecimal.

Carry-Out? Overflow?

3.

$$\begin{array}{r} \text{A A F F 9 0 1 4}_{16} \\ +\ \underline{\text{A A E 3 C D 1 2}}_{16} \end{array}$$

Result in hexadecimal:

Carry-Out? Overflow?

4.

$$\begin{array}{r} \text{7 F A A 3 2 7 8}_{16} \\ +\ \underline{\text{6 0 2 4 C D 1 2}}_{16} \end{array}$$

Result in hexadecimal: