CS 241 — Lecture 1

Bartosz Antczak *Instructor: Kevin Lanctot* *January 3, 2017*

1.1 Course Info

This course outlines what happens when you compile and run a program. We will be critical of programming languages and also build our own.
For the past 30 years, the two hardest programs you can build are operating systems and compilers (and guess what we're building? A compiler!).
Assignments mainly consist of programming.

1.2 Representing Data

Humans often represent numbers using combinations of 10 different symbols (0 ... 9; referred to as *base 10*); however, there are other number systems. We'll focus on

- Binary — represents numbers using only two symbols (0 and 1). Computers use this number system
- Hexadecimal — represents numbers using sixteen different symbols (0 ... 9 and A ... F)

Computers used to use base 10 symbols, but it was converted to binary in the 1930s because it led to a much simpler design.
To display which number system we're using, we include a subscript on the number. For instance, if we were to write 0111 in binary, we'd represent it as 0111_2 (2 for binary).

Example 1.2.1. *How we write numbers:*

What do we mean when we write 50,320 in base 10?

$$50320_{10} = 5 \cdot 10^4 + 0 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 0 \cdot 10^0$$

Each symbol is in their respective "placeholder" (i.e., in 50320, the number 3 is in the hundred's place). This "placeholder" approach is used to write numbers in every number system.

Example 1.2.2. *Writing and adding binary:*
To add binary digits (remember the only two digits are 0 and 1), we follow four simple rules:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

What is $101_2 + 011_2$*?*

$$\begin{array}{r} 1\ 1 \\ 1\ 0\ 1 \\ +\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

1.2.1 Representing negative numbers

In binary, we initially tried adding an extra digit at the front of the binary string to represent either positive or negative digits (i.e., $\underline{0}001$ is +1, and $\underline{1}001$ is -1). But this didn't work because zero was then represented as two different values (+0 and -0), which is redundant.

Two's compliment

This is a much better method which gets rid of the two 0's issue while still allowing computers to interpret negative numbers. To represent a negative number, invert the bits and add 1.

Example 1.2.3. *Negate* 4_{10}*:*

Binary Representation of 4_{10}	*Invert*	*Add 1*
0100	1011	1100

Thus, $1100 = -4_{10}$. Two's compliment works because of modular arithmetic. More on this in the next lecture.