CS 241: Foundations of Sequential Programs

Chris Thomson

Winter 2013, University of Waterloo

Notes written from Gordon Cormack's lectures.

1 Introduction & Character Encodings

1.1 Course Structure

The grading scheme is 50% final, 25% midterm, and 25% assignments. There are eleven assignments. Don't worry about any textbook. See the course syllabus for more information.

1.2 Abstraction

Abstraction is the process of removing or hiding irrelevant details. Everything is just a sequence of bits (binary digits). There are two possible values for a bit, and those values can have arbitrary labels such as:

- Up / down.
- Yes / no.
- 1 / 0.
- On / off.
- Pass / fail.

Let's say we have four projector screens, each representing a bit of up/down, depending on if the screen has been pulled down or left up (ignoring states between up and down). These screens are up or down independently. There are sixteen possible combinations:

Screen 1	Screen 2	Screen 3	Screen 4
Up (1)	Down (0)	Up (1)	Down (0)
Down (0)	Down (0)	Down (0)	Up (1)
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Note that there are sixteen combinations because $k = 4$, and there are always 2^k combinations since there are two possible values for each of k screens.

1.3 Endianness

Let's consider the sequence 1010. This sequence of bits has a different interpretation when following different conventions.

- **Unsigned, little-endian**: $(1 \times 2^0) + (0 \times 2^1) + (1 \times 2^2) + (0 \times 2^3) = 1 + 4 = 5$.
- **Unsigned, big-endian**: $(0 \times 2^0) + (1 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) = 2 + 8 = 10$.
- **Two's complement, little-endian**: $5 - 16 = -10$.
- **Two's complement, big-endian**: $10 - 16 = -6$.
- **Computer terminal**: LF (line feed).

Note that a two's complement number n will satisfy $-2^{k-1} \leq n < 2^{k-1}$.

1.4 ASCII

ASCII is a set of meanings for 7-bit sequences.

Bits	ASCII Interpretation
0001010	LF (line feed)
1000111	G
1100111	g
0111000	8

In the latter case, 0111000 represents the character '8', not the unsigned big- or little-endian number 8.

ASCII was invented to communicate text. ASCII can represent characters such as A-Z, a-z, 0-9, and control characters like ();!. Since ASCII uses 7 bits, $2^7 = 128$ characters can be represented with ASCII. As a consequence of that, ASCII is basically only for Roman, unaccented characters, although many people have created their own variations of ASCII with different characters.

1.5 Unicode

Unicode was created to represent more characters. Unicode is represented as a 32-bit binary number, although representing it using 20 bits would also be sufficient. The ASCII characters are the first 128, followed by additional symbols.

A 16-bit representation of Unicode is called **UTF-16**. However, there's a problem: we have *many* symbols ($> 1M$) but only $2^16 = 65,536$ possibilities to represent them. Common characters are represented directly, and there is also a 'see attachment' bit for handling the many other symbols that didn't make the cut to be part of the $65,536$. Similarly, there is an 8-bit representation of Unicode called **UTF-8**, with the ASCII characters followed by additional characters and a 'see attachment' bit.

The bits themselves do not have meaning. Their meaning is in your head – everything is up for interpretation.