

Announcements

- Program #3
 - is available – start before spring break!!
- Reading
 - Notes (Today)
 - Chapter 16, 13.3 (Tuesday (after Spring Break))
 - Skip 16.4-16.6 and 16.9

Representing Characters

- Need
 - Represent common characters
 - Have standards so computers can interoperate
- Common Formats
 - ASCII
 - 7 bits for characters (stored in 8 bits normally)
 - most commonly used
 - UNICODE
 - family of encodings 8, 16, and 32 bits/character
 - allow greater variety of characters
 - Able to represent virtually every character in use today
 - and some no longer in use

ASCII

- Represents normal characters on US keyboards
 - A-Z (101-132)
 - a-z (141-172)
 - 0-9 (60-71)
 - punctuation: !@#\$%^&*()_+--=[]{}|\;:"'<>?.,/
 - space
 - <control-A> - <control-Z>
 - Also have other names <control-M> is CR (\r)

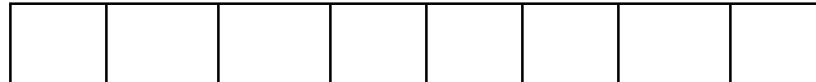
UNICODE

- Unicode Representations

- UTF-32 32 bit representation of all characters
 - all characters are the same size
 - wastes lots of space (2x UTF-16 for most things, 4x ascii for many things)
- UTF-16 16 bit representation of characters
 - some characters are stored in two-character forms
 - popular since most things can be represented in 16 bits
- UTF-8 8 bit representation of characters
 - provides backwards compatibility with ascii
 - low 7 bits are exactly ASCII
 - high bit on indicates part of UNICODE extensions
 - popular for web and other applications

Representing Integers

- All data stored in binary
 - all numbers are 0 or 1
- Unsigned numbers are stored using base 2



128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- possible range 0 to 2^n-1 where n is the number of bits
- Signed numbers are stored using two's complement
 - left most bit indicates if a number is positive or negative
 - 0 is positive
 - 1 is a negative number

Adding Binary Numbers

- Add starting from right
- Carry if the number is too large to represent
 - if it is greater than 1

1001	1001	1011	1101
+ 10	+ 11	+ 11	+111
-----	-----	-----	-----
1011	1100	1110	10100

2's complement representation

- To compute a negative value:
 - flip all the bits of the positive value, add 1
- Allows addition of signed and unsigned numbers
- Valid range of numbers
 - -2^{n-1} to $2^{n-1}-1$ for n bits
 - Example: 16 bit 2's complement -32,768 to 32,767
 - Example: 4 bit 2's complement -8 to 7
- Example: adding 4-bit 2's complement numbers

0010	1101	1101	0101	1001
0011	0001	1110	0111	1010
0101	1110	1011	1100	0011

How do we represent real numbers?

- Each number has two parts
 - mantissa (represents a number between -1 and 1)
 - represented as a binary number i.e. $0.1 = 1/2$
 - exponent (designates the position of the decimal point)
 - uses normal two's complement form
 - number = $m r^e$ where r is the radix
 - $6132.789 = +0.6132789 \times 10^4$
- Normalization
 - convert to number between -1 and 1
 - the most significant digit of mantissa is non-zero

Floating Point Continued

- Computers normally use a radix of 2
- Examples of floating point numbers
 - $1001.11 = .1001110 \times 2^4$
 - $10.5 = 1010.1$ or 10101×2^4
 - $7.4375 = 111.011100 = 1110111 \times 2^3$
- IEEE Floating point standard
 - 32 bit floating point (float)
 - 1 sign bit, 8 bits exponent, 23 bits mantissa
 - 64 bit floating point (double)
 - 1 sign bit, 11 bits exponent, 52 bits mantissa
 - most common for real applications
 - 128 bit floating point (quad)
 - 1 sign bit, 15 bits exponent, 112 bits of mantissa
 - normalization allows for an assumed 1 before the decimal point