Computer Organization and Architecture

Conrad A. Mearns

January 24, 2017

Turing Machines

1. Turing's Thesis: Every computation can be represented with a Turing Machine.
2. Turing Machine: A mathematical model of a device that can preform any computation.
3. Universal Turing Machine: A machine to implement any and all Turing Machines.

Beyond models, real world constraints include time, financial cost, power, security, thermal dissapation, space, etc.

Bits, Data Types, and Operators

The electro-magnetic field is not digital, yet all of modern computing is represented digitally. To compromise, 0 is a representation of the absence of voltage and 1 is a representation of the presence of voltage.

0V 0.5V "Illegal" 2.4V 2.9V

Signed Binary Arithmetic

Binary, without the addition of extra mathematical symbols, can only represent positive whole integers. Signed numbers like -5 require a formal system of representation in order to be used. A binary number can represent 2^n values for n bits. The objective of signed numbers is to partition half of those values for negative number representation ($-2^{n-1} - 1 \rightarrow -1$) and the other half to positive numbers ($1 \rightarrow 2^{n-1}$) while leaving zero and potentially one other number available.

1. Sign-Magnitude
   The most significant bit is 0 for positive values and 1 for negative values.
   $00101 = 5$ and $10101 = -5$
2. One's Complement
   All bits are inversed to represent negative numbers. Like
   Sign-Magnitude, the most significant bit will tell you whether a number is positive or negative.
   $00101 = 5$ and $11010 = -5$

3. Two's Complement (currently in use)
   For each positive number A, it's negative number (B) satisfies the equation $A + B = 0$ when the final carried bit is dropped. To get this number, take the One's Complement of A and add 1. $00101 = 5$ and the One's Complement is 11010. So the Two's Complement is 11011. As proof, $00101 + 11011 = 100000$ but the last carried one is dropped, leaving 00000.

Arithmetic and Logical Operations

Arithmetic operations

1. Addition
   Just addition, regardless if signed or not. Ignore the final carry-out.

2. Subtraction
   First negate the second operand ($5 \rightarrow -5 for example$), then use addition.

3. Sign Extension
   To add numbers, they must have the same number of bits. This is because of signed numbers, and storage.

Overflow occurs when

1. Signs of the operands are the same

2. The sign of the sum is different

The issue can be tested for by examining the most significant bit's sign between the operands and the result.
Logical operations

1. AND
   The result is true if and only if both operands are true.
   Useful for clearing bits, a mask of 1's signify keep.

2. OR
   The result is true if either operand is true.
   Useful for setting bits. 1's in the second operand copy to the result.

3. NOT
   The result is true if and only if the operand is false.

Each operation is executed on each bit individually.

Fractions, Floating, and Fixed-Point Values

A "binary" point is abstractly added to the value. To the left of the point, each bit is worth is 2^{-n} where n is the place left of the point. For example, 101.11. For large numbers, we use scientific notation. $Sign * (Fraction * 2^{Exponent})$
IEEE 754 Floating Point Standard for 32-bits signifies 1 bit for sign, 8 bits for

the exponenant, and 23 bits for the fraction.
$10111111010000000000000000000000 = -1.5 * 2^{???}$

Other Date Types

1. Single characters use ASCII to map 128 characters to 7-bit code.
2. Text strings are sequences of characters often with a NULL to terminate. No hardware support.
3. Images are arrays of images. Often has hardware support.
4. Sound is a sequence of fixed-point numbers.