Computer Math

Document License

Introduction

Intent / Goals

This document was started after observing a large number of questions on newsgroups that related to how "computer math" actually operated. This document doesn't aim to reproduce a normal math textbook but instead focus on areas that are specific to computers.

@@EXPAND@@

Notations & Conventions

@TODO:

I. Binary Math

There's a common reference that "Computers think in 1's and 0's" - this is not exactly true (The computers operate based on electrical signals that people equate into 1's and 0's - electronically logical high and logical lows).

The Binary number system has what's called a base of "2". A base refers to the number of digits allowed in a number system hence the only possible digit representations allowed for binary are 0 and 1. Numbers greater than this require additional digits to the left. The following table gives an example of how the numbers 0 through 9 in decimal are written in binary.

Decimal	**Binary**
0_{10}	0000_2
1_{10}	0001_2
2_{10}	0010_2
3_{10}	0011_2

4_{10}	0100_2
5_{10}	0101_2
6_{10}	0110_2
7_{10}	0111_2
8_{10}	1000_2
9_{10}	1001_2

table I-A

Note that it's conventional to show binary numbers as having a length that is a multiple of either 4 or 8 binary digits. Any unfilled positions are represented by prefixing 0's at the beginning as shown in the table I-A above.

Interesting properties of Binary "base 2" representation:

- All even numbers, including 0, have a 0 in their rightmost digit position.
- All odd numbers have a 1 in their rightmost digit position.
- All powers of 2 have a leading 1 followed by the number of zeros by which 2 is raised.
  Example: 2^3 (= 8_{10}) meaning that it is represented in binary as a 1 followed by 3 zeros: 1000_2

Representation

Base 2 numbers are typically represented as the number with a subscript of 2. In the table above the decimal number 4 could be written in binary as $100_{2.}$ Similairly, decimal numbers may also be written with a subscript representing their base. The lack of a numerical subscript is understood to imply a normal "base 10" decimal number. Eg: 4_{10} = 100_2.

Application

@@TODO: Application

II. Number Systems

While computers can be said to represent the number systems, because of various issues you'll often find other numbers systems used when dealing with computers - typically when programming at a very low level. These reasons are typically for historic and convention reasons.

Hexadecimal Number System

One of the main ones used is that of the Hexadecimal system or base 16. Hexadecimal ('Hex') uses the first 6 digits of the alphabet to represent the additional 6 digits of the system:

Hexadecimal	Binary	Decimal
00_{16}	0000_2	0_{10}
01_{16}	0001_2	1_{10}
02_{16}	0010_2	2_{10}
03_{16}	0011_2	3_{10}
04_{16}	0100_2	4_{10}
05_{16}	0101_2	5_{10}
06_{16}	0110_2	6_{10}
07_{16}	0111_2	7_{10}
08_{16}	1000_2	8_{10}
09_{16}	1001_2	9_{10}
$0A_{16}$	1010_2	10_{10}
$0B_{16}$	1011_2	11_{10}
$0C_{16}$	1100_2	12_{10}
$0D_{16}$	1101_2	13_{10}
$0E_{16}$	1110_2	14_{10}
$0F_{16}$	1111_2	15_{10}
10_{16}	00010000_2	16_{10}

FF_{16}	11111111_2	255_{10}

table I-B

Unlike binary numbers, hexadecimal digits are usually represented as having a length that is a multiple of 2 although this is not typically followed as strictly as the prefixing for binary numbers. See table I-B above.

Positive and Negative

@@todo

One / Two's Complement

@@todo

Binary Coded Decimal (BCD) Math

Binary Coded Decimal (BCD) is another numeric notation that uses 4 bit positions to represent each digit in a decimal number. The number 190_{10} is represented as 000110010000_{bcd}. If this number is broken up into 4 bit chunks (also known as a nibble) each chunk represents the corresponding number: 1, 9 and then 0 in this case.

Some processors support performing BCD math natively without having to go through a conversion process (eg: converting to their native binary format). The main advantage of BCD is that entered ascii characters can rapidly be converted to BCD; the main disadvantage is that the amount of values that can be represented in a byte (8 bits) shrinks from binary's 256 to 100 (including zero).

BCD	**Binary**	**Decimal**
0_{bcd}	0000_2	0_{10}
1bcd	0001_2	1_{10}
2bcd	0010_2	2_{10}
3_{bcd}	0011_2	3_{10}
4_{bcd}	0100_2	4_{10}
5_{bcd}	0101_2	5_{10}
6_{bcd}	0110_2	6_{10}
7_{bcd}	0111_2	7_{10}

8_{bcd}	1000_2	8_{10}
* INVALID *	1001_2	9_{10}
* INVALID *	1010_2	10_{10}
* INVALID *	1011_2	11_{10}
* INVALID *	1100_2	12_{10}
* INVALID *	1101_2	13_{10}
* INVALID *	1110_2	14_{10}
* INVALID *	1111_2	15_{10}
10_{bcd}	00010000_2	16_{10}
* INVALID *	11111111_2	255_{10}

table II-B

Number System Conversion

http://www.permadi.com/tutorial/numHexToBin/index.html

Adding, Subtracting

Boolean Logic

Multiplication

Division

Errors

Approximations

@@todo

error referred to here are the errors that involve a lack of precision in the value being represented. A fractional value of 1/3 is unable to be represented in a binary form. With an implied leading decimal point 1/3 closest approximation is 01010101_2 (ie : 85/256); expressed in decimal form this value is 0.332 - an error of 0.00133333. Going to 16 bits results in an approximation of 21845/65536 which is 0.33332824 in decimal. Here, an error is still seen.

Growth of Errors

@@TODO : Discuss recurent operations errors.

Boolean Logic

@@TODO: True/False

@@TODO: And

@@TODO: Or/Xor

@@TODO: Not

@@TODO: Nor/Nand

@@TODO: Implication

@@TODO: Boolean Laws : DeMorgans

@@TODO: Other laws

@@TODO: Boolean Logic Proofs

Set Operations

@TODO: Implication

@TODO: Equivlance

@TODO: Union

Square Root

A quick algorithm to calculate integral square roots can be found by the observation that the count of all successive odd numbers that total up to the number is the integral square root.

Example:

25 = 1 + 3 + 5 + 7 + 9

36 = 1 + 3 + 5 + 7 + 9 + 11 / sqrt = 6

Note 11+1 = 12, 3+9=12,

@@EXPLAIN MATH@@

Fixed Point Arithmetic

Geometric Functions

Algorithms / Applications

@@TODO: Breshem's Line drawing algorithm (?? name)
@@TODO: Power Approximations
@@TODO: Table Lookup Approximations (eg: Sine / Cosine etc via table of powers)