HOW COMPUTERS CALCULATE-BINARY ARITHMETIC AND BOOLEAN LOGIC

Paul Bouthellier
Department of Computer Science and Mathematics
University of Pittsburgh-Titusville
Titusville, PA 16354
pbouthe@pitt.edu

Any student who uses a computer or calculator to analyze a mathematical or statistical problem should have a good idea of how the machine actually computes the results. Such knowledge is necessary to guard against (and deal with) approximation errors, overflows, underflows, wrap-around-infinities, and a hosts of related problems. It also shows students that behind every button push is not "magic" but just some rather clever mathematics and engineering. The main results in this paper are broken into three parts: how and why machine calculations go wrong, how binary addition and subtraction is done, and how logic gates are used to perform addition and subtraction in calculators and computers.

We begin by looking at problems such as approximation errors and wrap-around infinites. At best, when these problems occur an error will be reported and the calculations halted. At worst, no error will be reported and an incorrect result will be returned.

Next we will discuss the mathematical theory behind how computers perform calculations (the software). Addition is straightforward. Subtraction is done by a technique called the "two's complement method" which turns a subtraction problem into one of addition.

Finally we will discuss the engineering (hardware) which implements the software. The hardware consists of circuits called half-adders and adders which are based on the Boolean operators XOR, AND, and NOT. Examples combining the software and hardware results will be given to show how calculators and computers actually compute.

Computing Follies: The Not Good, The Bad and the Truly Ugly

Some reasons for problems with calculations as done by calculators and computers are as follows:

- Calculators & Computers Problems:
  - Limited Internal Accuracy
  - Limited Number of On-Screen Decimals
  - Representation Problems via Bases
  - Inherent Instability of Certain Problems
    - Root Finding
    - Time-Varying Dynamical Systems

- Limitations due to Sampling
  - Graphing
- Accuracy of Numerical Methods Problems:
  - Integration
  - Differentiation
  - R-K Methods
  - Root-Finding

Examples of such problems are given as follows:

Lack of Sufficient Internal Digits [1]

```
run:
10^20+4+10^20 = 0.0
10^20-10^20+4 = 4.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 1

The above was done in Java, which generally only carries 15 decimal places.

The same problem is illustrated in GeoGebra 5 below:

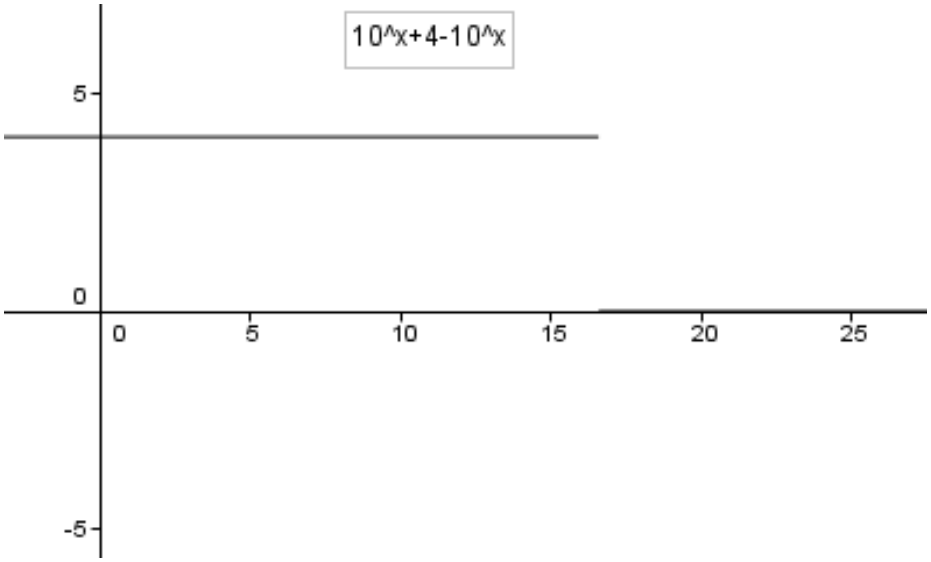

Figure 2

A related problem is illustrated in Maple:

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^{x}=1$$

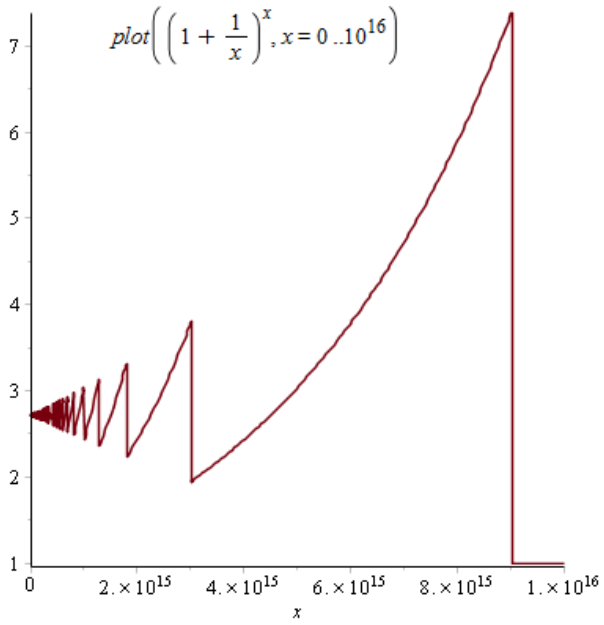

Figure 3

The Lack of Sufficient Internal Digits to Represent Numbers Often Causes Wraparounds:

```
run:
-128-1 is 127
-32768 -1 is 32767
127+1 is -128
32767+1 is -32768
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4-Using Java

The problem with these is that no error message is generated-just an incorrect answer.

These examples illustrate why students need to know the basics of how computers and calculators generate their results.

Binary Math [1], [2]

First we shall consider the unsigned case.

Consider the 8-bit case

0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Smallest Number:

0*2^7+0*2^6+0*2^5+0*2^4+0*2^3+0*2^2+0*2^1+0=0

Largest Number:

1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+10=255

Binary Addition

AND	0	1
0	0	1
1	1	10

70+30=100

0	1	0	0	0	1	1	0

+

0	0	0	1	1	1	1	0

=

0	1	1	0	0	1	0	0

=2^6+2^5+2^2=100

If there is not enough space to hold the resulting number the calculation is said to overflow.

Binary Subtraction-The Two's Complement Method

To represent negative numbers:

- Compute the binary representation of the number
- Switch the 0's and 1's
- Add 1 to the result

70=01000110

To compute -70:

1	0	1	1	1	0	0	1

+

0	0	0	0	0	0	0	0	1

=

1	0	1	1	1	0	1	0

To represent -30:

30=00011110

1	1	1	0	0	0	0	1

+

0	0	0	0	0	0	0	1

=

1	1	1	0	0	0	1	0

Binary Subtraction-Turning Subtraction Into Addition

Consider Y-X

where X and Y >0 are integers

1. Replace X with its two's complement
2. Binary add Y and the two's complement of X

- If 9th-bit is =1 Y-X>0 and
  - Y-X=8-bit representation
- If 9th-bit is=0 Y-X<0 and
  - Y-X=-The two's complement of the 8-bit representation

To compute 70-30:

70=01000110
-30=11100010

0	1	0	0	0	1	1	0

+

1	1	1	1	0	0	0	1	0

=

0	0	1	0	1	0	0	0

As 9th-bit=1:

70-30=2^5+2^3=32+8=4

To compute 30-70

30=00011110
-70=10111010

0	0	0	1	1	1	1	0

+

1	0	1	1	1	0	1	0

=

1	1	0	1	1	0	0	0

As 9^{th} -bit=0:

Take the two's complement of 11011000=00101000

Y-X=-(2^5+2^3)=-40

The signed case uses the first bit of a number's representation to denote whether the number is positive or negative:

- 0 implies positive
- 1 implies negative
  - 10000000=-128
  - 1xxxxxxx=-128+the binary value of xxxxxxxx

Such a scheme allow 8-bits to represent the numbers -128 to 127 with a unique representation of 0.

Binary addition and subtraction are then done exactly as illustrated above using binary addition and the two's complement.

Implementing Binary Addition via Logic Gates [1], [2]

Need three logic gates:

- AND
- OR
- XOR

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

These Boolean functions are implemented in computers via circuitry referred to as AND gates, OR gates, and XOR gates respectively. These gates are then combined into circuits called half-adders and adders as illustrated below.

Adders and Half-Adders: The four-bit case

$$x_1x_2x_3x_4+y_1y_2y_3y_4$$

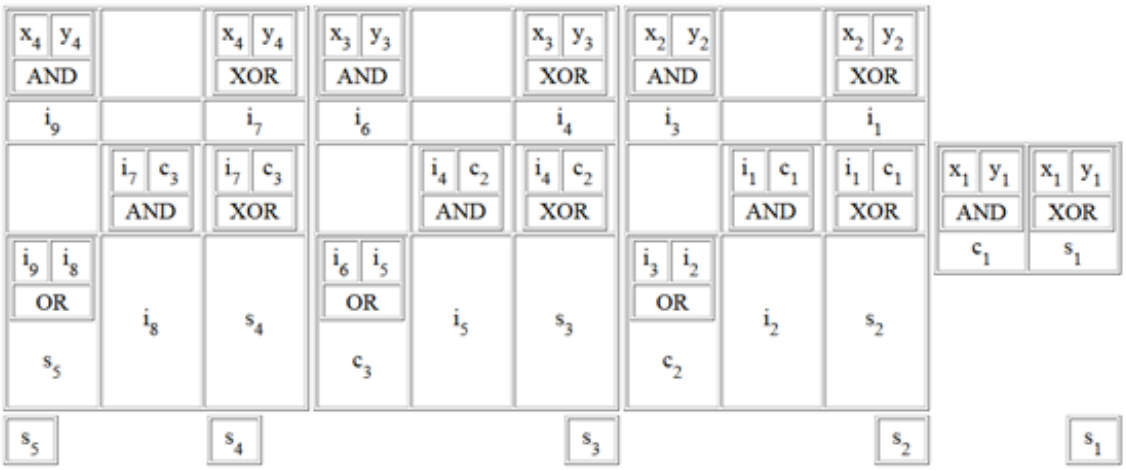

Figure 5

The s, c, and i terms represent the sums, carry-over digits, and the intermediate calculations respectively.

The rightmost table is called a half-adder and is used to add the two rightmost digits in a binary addition. The larger boxes on the left are called full-adders and are used to add digitsand deal with any carry-over calculations necessary.

An example is given as follows:

Here we shall use boolean logic to compute

12+11

1100+1011

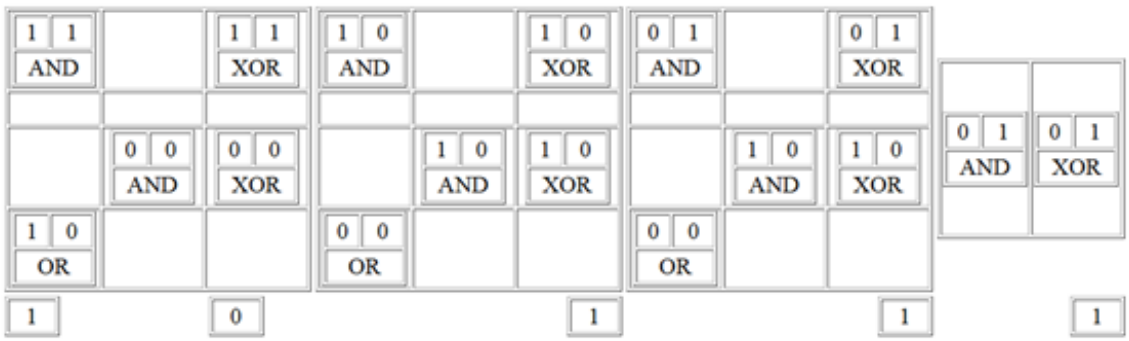

Figure 6

Hence:

12+11=10111=23

References:

[1] Mark Howell, *Lies My Calculator Told Me,* http://apcentral.collegeboard.com.apc/members/courses/teachers_corner/11703.html

[2] Gerald R. Rising, *Inside Your Calculator* Hoboken, NJ,Wiley, 2007.

[3] Clive Maxwell and Alvin Brown, *The Definitive Guide To How Computers Do Math* Hoboken, NJ, Wiley, 2005.