Twos Complement and Floating Point

CSCI 23000

Andy Harris
Indiana University / Purdue University - Indianapolis

Adding Binary Numbers

- 101 + 111 = ?
- Just like base ten
- ```
   11
   101
  +111
  ====
  1100
  ```

Subtracting binary numbers

- Also like base ten
- Use borrowing: 101 - 11
- ```
   101       5
   -11       3
   ===
    10       2
  ```
- Sometimes you'll need to borrow from several digits
- ```
   1000      8
   -   1     1
   ====
    111      7
  ```
  ```

Multiplying binary numbers

- 101 * 101
- Actually simpler than in base 10!
- ```
    101      5
   *101      5
   ====
    101
   0000
  10100
  =====
  11001     25
  ```

Problems with unsigned numbers

- finite space in memory
- There will be a number to big to store
- No negative numbers
- No fractional or real values

Possible solutions

- Sign / Magnitude
- Ones complement
- Twos complement

Sign / Magnitude

- first digit is sign
- rest is magnitude
- two zeros (?!?)
- 5 + (-3) doesn't work

Ones Complement

- Just reverse all values
- If first value is 1:
  - invert everything
  - make it negative
- Still has two zeros
- 5 + (-5) works, but 5 + (-3) doesn't
  ```

Twos complement

- All digits have ordinary magnitude
- leftmost digit is negative
- (trick - invert and add one)
- Only one zero
- Adding negatives works!

Two's Complement examples

- Leftmost digit is negative
-

-8	4	2	1
1	0	0	1

- -8 + 1 = -7
- If leftmost digit is zero, number is positive
- If leftmost digit is one, number is negative

The strange behavior of negative numbers

- What is 10 in two's comp? (-2)
- What is 110? (-4 + 2 = -2)
- 1110? (-8 + 4 + 2 = -2)
- 11110? (-16 + 8 + 4 + 2 = -2)
- A negative number in two's complement has infinite leading ones!

Adding a negative

- 5 + -3
-

-8	4	2	1
0	1	0	1
1	1	0	1
0	0	1	0

- Negative numbers have infinite leading 1s!

Real numbers in binary

- How could you handle fractions?
- What about the decimal (binary?) point
- Does .1 have meaning in binary?
- How about .01?

Reviewing binary notation

- Values to the left of the point are two raised to a positive exponent
- Values to the right of the point are two raised to a negative exponent
-

2^{-1}	2^{-2}	2^{-3}	2^{-4}
1/2	1/4	1/8	1/16
.5	.25	.125	.0625

- .011 (binary) = .25 + .125 = .375 (decimal)

Converting binary to decimal

- Convert .101 (binary) to decimal
- Just like converting whole numbers:
-

.5	.25	.125	.0625
1	0	1	0
.5	0	.125	0

- .5 + .125 = .625

Converting Decimal to Binary

- Convert .6 (decimal) to binary
- Just like integers:
-

.5	.25	.125	.0625	.03125	.015625	.0078125	.00390625
1	0	0	1	1	0	0	1

- 1/2 + 1/16 + 1/32 + 1/256 = 0.59765625

Real numbers and error

- Binary conversion is not exact
- Many fractions (like .6) are repeating
- Limited precision
- Working with floating numbers frequently introduces error

Floating point notation

- Two main approaches:
- Fixed: easier but less flexible
- Floating: More flexible but more complex
- Fixed: point is always in the same place
- Floating: point can move around.
- Floating is now standard

Reviewing Scientific notation

- Avagadro's Number
- $6.02 * 10^{23}$
- Scientific notation descrives large and small numbers
- It stores a real number as two integers
- 6.02 is *Mantissa*
- 10 is *Base*
- 23 is *Exponent*

Devising a floating notation

- Scientific notation can be modified for binary
- Store mantissa and exponent as integers
- Use 2 as the base
- EG .01 binary could be represented like this:
- $0001 * 2^{-2}$

A representative notation

- for simplicity, we'll devise a format
- ABBBBCDD
- A = sign of mantissa
- B = magnitude of mantissa
- C = sign of exponent
- D = magnitude of exponent

Using our notation

- Start with a floating number:

1	0	1	0	1	0	1	1

- Break it into pieces:

sm	mantissa				se	exponent	
1	0	1	0	1	0	1	1

- Resolve all but mantissa:

sm	mantissa				se	exponent	
1	0	1	0	1	0	1	1
-	0	1	0	1	+	3	

Conversion continued

- Begin with mantissa in binary
- move binary point

$$-0101 * 2^{3} = -0101000$$

- Solve as binary problem
- Don't forget sign of mantissa
-

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	0

- -40

Working with negative exponents

- same, but move binary point to left

sm	mantissa				se	exponent	
1	0	1	0	1	1	1	1
-	0	1	0	1	-	3	

- $-0101 * 2^{-3} = -0.101$

- (1/2) + (1/8) = .5 + .125 = .625
- -.625

Decimal to float conversion

- Start with a decimal value

```
-.375
```

- Find the binary approximation

.5	.25	.125
0	1	1

- Convert to 4 digit mantissa

- `-0011 * 2`$^{-3}$

- Build ABBBBCDD chart

sm	mantissa				se	exponent	
1	0	0	1	1	1	1	1
-	0	0	1	1	-	3	

Notes about floating point

- There can be more than one 'right' answer
- convention is to use the smallest integer mantissa
- mantissa is stored as an integer (point on right)
- mantissa and exponent in sign / magnitude format
- There are four zeros!
- limited precision means possiblity (likelihood) of error

"Real" floating point notation

- The ABBBBCDD format is an approximation
- Numerous real formats exist
- All work the same way
- All have many more digits
- Basic idea is the same