



Representations 2

Prof. David August
COS 217



Today

- Unsigned Multiplication
- Fixed Point
- Floating Point

Multiplication



Computing Exact Product of w -bit numbers x, y

- Need $2w$ bits

Unsigned: $0 \leq x * y \leq (2^w - 1)^2 = 2^{2w} - 2^{w+1} + 1$

Two's Complement:

min: $x * y \geq (-2^{w-1})(2^{w-1}-1) = -2^{2w-2} + 2^{w-1}$

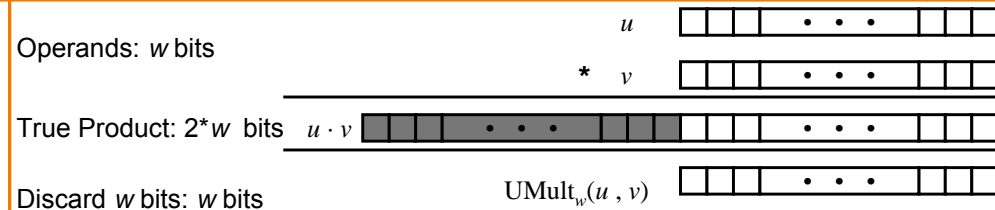
max: $x * y \leq (-2^{w-1})^2 = 2^{2w-2}$

- Maintaining Exact Results

- Need unbounded representation size
- Done in software by *arbitrary precision* arithmetic packages
- Also implemented in Lisp, ML, and other languages



Unsigned Multiplication in C



- Standard Multiplication Function
 - Ignores high order w bits
- Implements Modular Arithmetic
 - $UMult_w(u, v) = u \cdot v \text{ mod } 2^w$
- What about unsigned integer division?

Unsigned Multiplication



Binary makes it easy:

- 0 => place 0 (0 x multiplicand)
- 1 => place a copy (1 x multiplicand)

Key sub-parts:

- Place a copy or not
- Shift copies appropriately
- Final addition

5



Representations



What can be represented in N bits?

Unsigned: $0 \rightarrow 2^n - 1$

Signed: $-2^{n-1} \rightarrow 2^{n-1} - 1$

What about:

Very large numbers? 9,349,787,762,244,859,087,678

Very small numbers? 0.000000000000000000004691

Rationals? $2/3$

Irrationals? $\text{SQRT}(2)$

Transcendentals? e, PI

7

Interpretations



Bit Pattern	Method 1	Method 2	Method 3
000	0	0	0
001	1	1	0.1
010	e	2	0.2
011	pi	4	0.3
100	4	8	0.4
101	-pi	16	0.5
110	-e	32	0.6
111	-1	64	0.7

What should we do? Another method?

The Binary Point

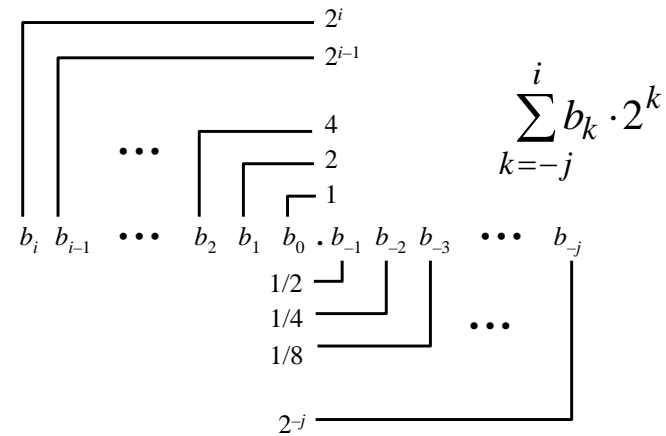


$$101.11_2 = 4 + 1 + \frac{1}{2} + \frac{1}{4} = 5.75$$

Observations:

- Divide by 2 by shifting point left
- $0.111111\dots_2$ is just below 1.0
- Some numbers cannot be exactly represented well
 $1/10 \rightarrow 0.0001100110011[0011]^* \dots_2$

Obvious Approach: Fixed Point



Fixed Point



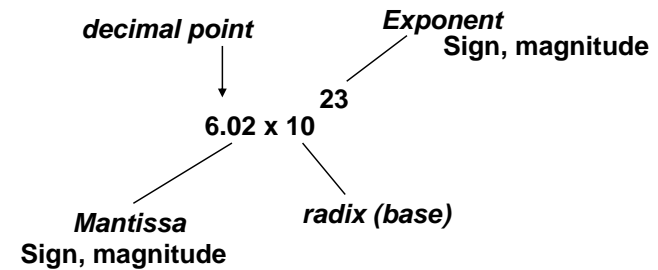
In w-bits ($w = i + j$):

- use i-bits for left of binary point
- use j-bits for right of binary point

Qualities:

- Easy to understand
- Arithmetic relatively easy to implement...
- Precision and Magnitude:
 $16\text{-bits, } i=j=8: 0 \rightarrow 255.99609375$
 Step size: 0.00390625

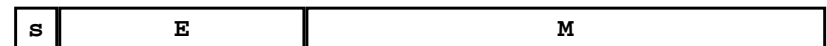
Another Approach: Scientific Notation



- In Binary:

$$\text{radix} = 2$$

$$\text{value} = (-1)^s \times M \times 2^E$$



- How is this better than fixed point?

IEEE 754 Floating Point



- Established in 1980 as uniform standard for floating point arithmetic
- Supported by all major CPUs
- In 99.999% of all machines used today

Driven by Numerical Concerns

- Standards for rounding, overflow, underflow
- Primarily numerical analysts rather than hardware types defined standard

This is where it gets a little involved...

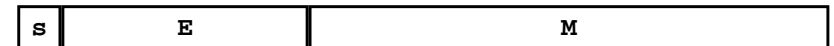
13

IEEE 754 Floating Point Standard



- Single precision: 8 bit exponent, 23 bit significand
- Double precision: 11 bit exponent, 52 bit significand
- Significand M normally in range $[1.0, 2.0)$ → Implied 1
- Exponent E biased exponent → B is bias ($B = 2^{|E|-1} - 1$)

$$N = (-1)^s \times 1.M \times 2^{E-B}$$



- Bias allows integer comparison (almost!)
 - 0000...0000 is most negative exponent
 - 1111...1111 is most positive exponent

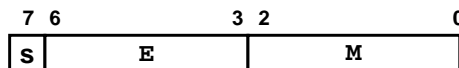
IEEE 754 Floating Point Example



Define Wimpy Precision as:

1 sign bit, 4 bit exponent, 3 bit significand, $B = 7$

Represent: -0.75



IEEE 754 Floating Point



There's more!

Normalized: $E \neq 000...0$ and $E \neq 111...1$

- Recall the implied 1.xxxxxx

Special Values: $E = 111...1$

- $M = 000...0$:
 - Represents $\pm \infty$ (infinity)
 - Used in overflow
 - Examples: $1.0/0.0 = +\infty$, $1.0/-0.0 = -\infty$
 - Further computations with infinity possible
 - Example: $X/0 > Y$ may be a valid comparison

16

IEEE 754 Special Exponents



Normalized: $E \neq 000\dots 0$ and $E \neq 111\dots 1$

Special Values: $E = 111\dots 1$

- $M \neq 000\dots 0$:
 - Not-a-Number (NaN)
 - Represents invalid numeric value or operation
 - Not a number, but not infinity (e.g. $\text{sqrt}(-4)$)
 - Examples: $\text{sqrt}(-1)$, $\infty - \infty$
 - NaNs propagate: $f(\text{NaN}) = \text{NaN}$

17

IEEE 754 Special Exponents



Normalized: $E \neq 000\dots 0$ and $E \neq 111\dots 1$

- Recall the implied $1.xxxxxx$

Denormalized: $E = 000\dots 0$

- $M = 000\dots 0$
 - Represents value 0
 - Note the distinct values $+0$ and -0

18

IEEE 754 Special Exponents



Normalized: $E \neq 000\dots 0$ and $E \neq 111\dots 1$

- Recall the implied $1.xxxxxx$

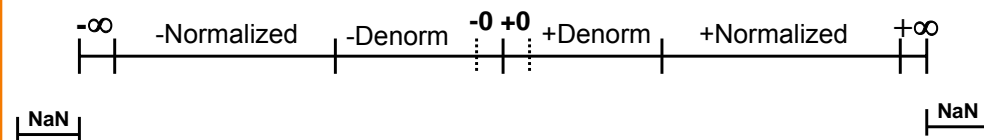
Denormalized: $E = 000\dots 0$

- $M \neq 000\dots 0$
 - Numbers very close to 0.0
 - Lose precision as magnitude gets smaller
 - “Gradual underflow”

Exponent $-Bias + 1$
Significand $0.xxxx\dots x_2$

19

Encoding Map



20

Dynamic Range



	S	E	M	exp	value	
Denormalized numbers	0	0000	000	n/a	0	
	0	0000	001	-6	1/512	← closest to zero
	0	0000	010	-6	2/512	
	...					
	0	0000	110	-6	6/512	
	0	0000	111	-6	7/512	← largest denorm

Normalized numbers	0	0001	000	-6	8/512	← smallest norm
	0	0001	001	-6	9/512	
	...					
	0	0110	110	-1	28/32	
	0	0110	111	-1	30/32	← closest to 1 below
	0	0111	000	0	1	
	0	0111	001	0	36/32	← closest to 1 above
0	0111	010	0	40/32		
...						
0	1110	110	7	224		
0	1110	111	7	240	← largest norm	

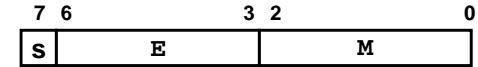
0	1111	000	n/a	inf		

Wimpy Precision



Define Wimpy Precision as:

1 sign bit, 4 bit exponent, 3 bit significand, B = 7



E = 1-14: Normalized

E = 0: Denormalized

E = 15: Infinity/ NaN