Floating Point

15-213: Introduction to Computer Systems
Recitation 2: Monday, Jan 27th, 2014

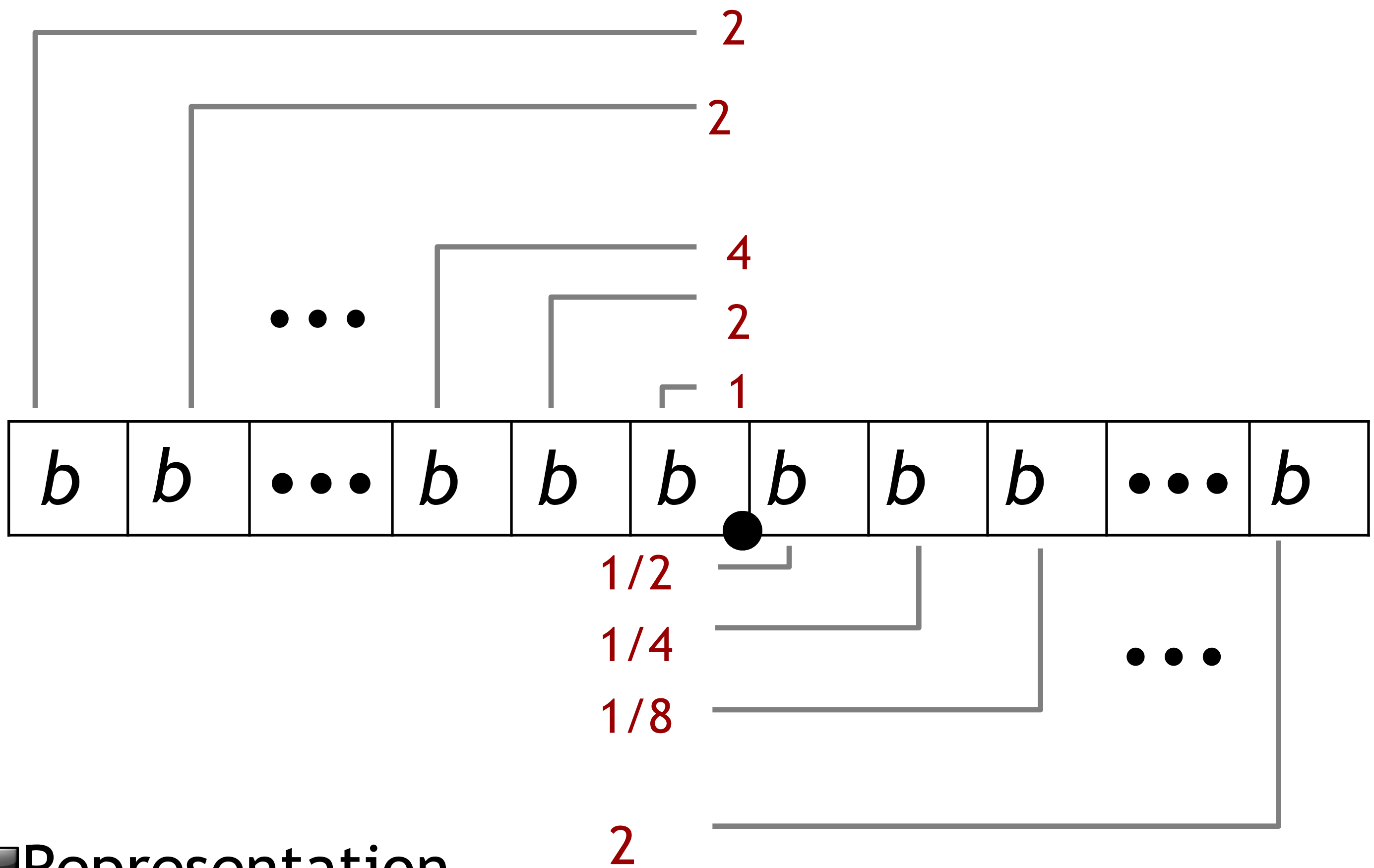
Agenda

- Floating point representation
  - Binary fractions
  - IEEE standard
  - Example problems

Reminder

- Data Lab is due Thursday, Jan 30^{th}

Floating Point - Fractions in Binary



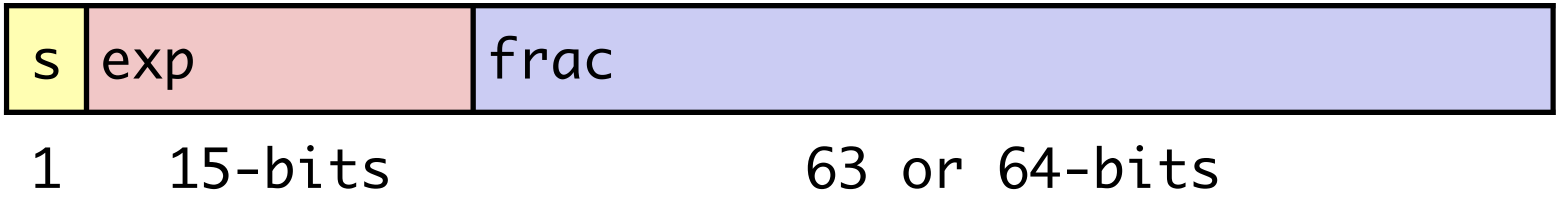
- Representation
  - Bits to right of “binary point” represent fractional powers of 2
  - Represents rational number: $\sum_{k=-j}^{i} b_k \times 2^k$

Floating Point - IEEE Standard

- Single precision: 32 bits

- Double precision: 64 bits

- Extended precision: 80 bits (Intel only)



Floating Point - IEEE Standard

- What does this mean?
  - We can think of floating point as binary scientific notation
  - The number represented is *essentially* (sign * frac * 2^{exp})
- Example:
  - Assume our floating point format has **no sign bit, k = 3 exponent bits, and n=2 fraction bits**
  - What does 0b10010 represent?

Floating Point - IEEE Standard

- What does this mean?
  - We can think of floating point as binary scientific notation
  - The number represented is *essentially* (sign * frac * 2^{exp})
- Example:
  - Assume our floating point format has **no sign bit, k = 3 exponent bits, and n=2 fraction bits**
  - What does 0b10010 represent? **3**

Floating Point - IEEE Standard

- Bias
  - exp is unsigned; needs a bias to represent negative numbers
  - Bias = 2^{k-1} - 1, where k is the number of exponent bits
  - Can also be thought of as bit pattern 0b011…111

Normalized	Denormalized
0 < exp < (2	exp = 0
Implied leading 1	Leading 0
E = exp - Bias	E = 1 - Bias.
Denser near origin	Evenly spaced
Represents large numbers	Represents small numbers

  - When converting frac/int => float, assume normalized until proven otherwise

Floating Point - IEEE Standard

- Special Cases ($exp = 2^k-1$)
  - Infinity
    - Result of an overflow during calculation or division by 0
    - $exp = 2^k-1$ (i.e. 1111…1), frac = 0
  - Not a Number (NaN)
    - Result of illegal operation (sqrt(-1), inf – inf, inf * 0)
    - $exp = 2^k-1$, frac != 0
  - Keep in mind these special cases are not the same

Floating Point - IEEE Standard

- Round to even
  - Why? Avoid statistical bias of rounding up or down on half.
  - How? Like this:

`1.01`	truncate	`1.01`
`1.01`	below half; round down	`1.01`
`1.01`	interesting case; round to even	`1.10`
`1.01`	above half; round up	`1.10`
`1.10`	truncate	`1.10`
`1.10`	below half; round down	`1.10`
`1.10`	Interesting case; round to even	`1.10`
`1.10`	above half; round up	`1.11`
`1.11`	truncate	`1.11`

Rounding

1.BBGRXXX

Guard bit: LSB of result

Round bit: 1st bit removed

Sticky bit: OR of remaining b

- Round up conditions
  - Round = 1, Sticky = 1 → > 0.5
  - Guard = 1, Round = 1, Sticky = 0 → Round to even

Value	*Fraction*	*GRS*	*Incr?*	*Rounded*
128	1.0000000	000	N	1.000
15	1.1010000	100	N	1.101
17	1.0001000	010	N	1.000
19	1.0011000	110	Y	1.010
138	1.0001010	011	Y	1.001
63	1.1111100	111	Y	10.000

Number to Float (S EEEE FFF, 8 bit FP)

- Convert: 27

Number to Float

- Convert: 27
- Positive so we know $S = 0$
- Turn 27 to bits:

$$27_{10} = 11011_2$$

- Normalized value so lets fit in the leading 1

$$1.1011_2$$

- But we only have 3 fraction bits so we must round. Digits after rounding equal half, last rounding digit is 1 so we round up.

$$1.1011_2 = 1.110_2$$

- Thus $F = 110$

Number to Float

- Calculate the exponent :

$$11100_2 \rightarrow 1.1100_2\, x\, 2^4$$

- Calculate the exponent bits:

$$E = Exponent - Bias \rightarrow 4 = Exponent - 7 \rightarrow Exponent = 4 + 7 = 11$$

- So the exponent is 11, in bits:

$$Exponent = 1011_2$$

- Answer: $0\ 1011\ 110_2$

Float to Number

- Convert: $1\ 1001\ 010_2$

Float to Number

- Convert: $1\ 1001\ 010_2$
  - Sign bit tells us it is negative
  - We know it is normalized (non-zero exponent) so lets figure out the exponent:

$$1001_2 = 9_{10}$$

$$E = Exponent - Bias \rightarrow 9 - 7 = 2$$

  - Now the fraction (remember the leading 1):
  - Put it all together: 1.010_2
  - Answer: -5 $1.010_2\ x\ 2^2 = 101_2 = 5_{10}$

Float to Number

- Convert: 0 0000 110

Float to Number

- Convert: 0 0000 110
  - Sign bit tells us its positive
  - It is denormalized because of the 0 exponent so lets figure out the exponent:

$$E = -Bias + 1 \rightarrow -7 + 1 = -6$$

  - Now the fraction (remember the leading 0):

$$0.110 \; x \; 2^{-6}$$

  - Put it all together:

$$0.110_2 \; x \; 2^{-6} = 0.000000110_2$$

Floating Point - Example

- For EEE FF, 5 bit FP, complete the following table:

Value	Floating Point	Rounded Value
9/32		
8		
9		
	000 10	
19		

Floating Point - Example

- For EEE FF, 5 bit FP, complete the following table:

Value	Floating Point	Rounded Value
9/32	**001 00**	**1/4**
8	**110 00**	**8**
9	**110 00**	**8**
1/8	000 10	
19	**111 00**	**inf**

Floating point encoding. In this problem, you will work with floating point numbers based on the IEEE floating point format. We consider two different 6-bit formats:

Format A:

- There is one sign bit s.
- There are $k = 3$ exponent bits. The bias is $2^{k-1} - 1 = 3$.
- There are $n = 2$ fraction bits.

Format B:

- There is one sign bit s.
- There are $k = 2$ exponent bits. The bias is $2^{k-1} - 1 = 1$.
- There are $n = 3$ fraction bits.

For formats A and B, please write down the binary representation for the following (use round-to-even). Recall that for denormalized numbers, $E = 1 - \text{bias}$. For normalized numbers, $E = e - \text{bias}$.

Value	Format A Bits	Format B Bits
Zero	0 000 00	0 00 000
One		
1/2		
11/8		

Solution

```
      | A        | B
One   | 0 011 00 | 0 01 000   Exact in both formats
1/2   | 0 010 00 | 0 00 100   Exact in both formats, norm in A, denorm in B
11/8  | 0 011 10 | 0 01 011   Format A round to even, format B exact
```

Floating Point Recap

- Floating point = $(-1)^s\, M\, 2^E$
- MSB is sign bit `s`
- Bias = $2^{(k-1)} - 1$ (k is num of `exp` bits)
- Normalized
  - `exp` ≠ 000…0 and `exp` ≠ 111…1
  - M = 1.`frac`
  - E = `exp` - Bias
- Denormalized
  - `exp` = 000….0
  - M = 0.`frac`
  - E = - Bias + 1

Floating Point Recap

- Special Cases
  - +/- Infinity: `exp` = 111…1 and `frac` = 000…0
  - +/- NaN: `exp` = 111…1 and `frac` ≠ 000…0
  - +0: `s` = 0, `exp` = 000…0 and `frac` = 000…0
  - -0: `s` = 1, `exp` = 000…0 and `frac` = 000…0
- Round towards even when half way (i.e. when LSB of `result` = 0)

Questions/comments?