RUTGERS

CS211 Computer Architecture

Fall 2020

Recitation 6

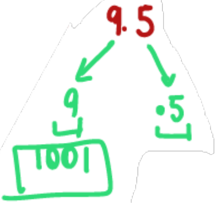
Floating Point Representation

- Used to represent more precise values (decimals)
- In order to do this, we must be able to know how to convert fractions into binary

Fractions in Binary

Ex: 9.5

- Separate the values before and after the decimal and convert



if numbers before decimal place is 2^0, what about after?

$2^{-1}, 2^{-2}, \dots \; 2^{-n}$

Fractions in Binary

Ex: 9.5

9.5

9 → 1001

.5

$0.5 \times 2 = \boxed{1}.0$

Bit value

1

$0.0 \times 2 = 0.0$

Stop here

to convert whole numbers into binary, we divided. For decimals, we multiply.

Multiply the decimals by 2, the corresponding bit value is the number to the left of the decimal place

Note: we do NOT reverse the order of bits as we did for whole numbers. So we are left with **0b.10** and you can drop the 0s after the 1 to get **0b0.1**

$0.5 = \boxed{.1}$

Fractions in Binary

Ex: 7.625

7.625

$\boxed{111}$

$.625 \times 2 = \underline{1}.\underline{25}$

$.25 \times 2 = \underline{0}.\underline{5}$

$.5 \times 2 = \underline{1}.0$

done!

Bit value
1
0
1

$\boxed{= .101}$

Fractions in Binary

Ex: 7.625

$$\boxed{= .\overset{-1}{1}\,\overset{-2}{0}\,\overset{-3}{1}}$$

verify

$$2^{-1} + 0\cdot 2^{-2} + 2^{-3}$$

$$= \frac{1}{2} + \frac{1}{8} = .5 + .125 = .625$$

7.625 = **0b111.101**

IEEE Floating Point

- Let's focus on **single precision** – [redacted]
  - Normalized

- Anatomy of Floating Point
  - sign bit – 1
  - exponent bias – 8
  - mantissa – 23

[redacted]

Steps for General Floating Point Rep

General steps:

1. notice sign bit, then ignore it
2. convert to binary (magnitude)
3. move decimal point before first 1 bit (keep track of how many spots you move)
4. identify mantissa as everything after decimal
5. Bias = $2^{n-1} - 1$ (n will be given - usually 8)
6. Put it all together

Example

4/2 = 2 0
2/2 = 1 0
1/2 = 0 1

100.11

.75x2
1.5>1

11

129

1.0011 x 10^2

0 10000001 0011

129/2 = 64 1
64/2 = 32 0
32/2 = 16 0
16/2 = 8 0
8/2 = 4 0
4/2 = 2 0
2/2 = 1 0
1/2 = 0

Ex: -4.75 in IEEE single-precision

1. negative = 1

2. 4.75 = 100.11

	Bit
.75 x 2 = 1.50	1
.50 x 2 = 1.0	1

Example

Ex: -4.75 in IEEE single-precision

③ $100.11 \rightarrow 1.\underbrace{0011}\times 2^{2}$ ← moved 2 places

④ mantissa = 0011

Example

Ex: -4.75 in IEEE Single precision

Bias is 8 bit as we use 8 bits to represent the exponent

5 exponent = bias + decimal places moved

Bias, n=8

$2^{8}-1 = 127$

$127 + 2 = \underline{129}$

convert to Binary in 8 bits (n) = 10000001

Example

Ex: -4.75 in IEEE single-precision

$$\begin{array}{r} 23 \\ -\ 4 \\ \hline 19 \end{array}$$

(b)

sign bit	exponent	mantissa
1	10000001	0011000...0

7 6 5 4 3 2 1 0

2^7 2^0

$\underbrace{0011000\ldots0}_{23\text{ bits}}$

→ answer

Example

Ex: -4.75 in IEEE single-precision

how do you check? work backwards ☺

- exponent -127 = decimal places moved = e
- $1.[\text{mantissa}] \times 2^{e}$
- sign = 1 if negative
  0 if positive

Recap

- what does IEEE 754 floating point mean?

different types of representation
- single pt - 32 bit
- double pt - 64 bit

↳ 32 bits in answer

↳ 1 sign bit

↳ 8 exponent

↳ 32-1-8 = 23 for mantissa (just add 0s)

Quick Note - Normalized vs Denormalized

- The range of values you can represent in floating point can be defined as **normalized** or **denormalized**
- **Normalized**
  - Our exponent bias = exponent – bias
  - Value represented leading with 1 (1.[mantissa])
- **Denormalized**
  - Our exponent bias = 1 – bias (exponent = 0)
  - Value represented leading with 0 (0.[mantissa])
  - Usually represents numbers very close to 0