COMPSCI 2GA3 Tutorial 6 Note

Note:

This note does NOT cover all the materials in Chapter 3 -- Only the ones rated to sample questions of this tutorial are included.

For any questions about the tutorials and courses, feel free to contact me. (Email: wangm235@mcmaster.ca)

GLHF :)
Mingzhe Wang

Decimal to binary

Given a fraction decimal number n and integer k, convert decimal number n into equivalent binary number up-to k precision after decimal point.

Examples:

```
Input: n = 2.47, k = 5
Output: 10.01111

Input: n = 6.986 k = 8
Output: 110.11111100
```

A) Convert the integral part of decimal to binary equivalent

1. Divide the decimal number by 2 and store remainders in array.
2. Divide the quotient by 2.
3. Repeat step 2 until we get the quotient equal to zero.
4. Equivalent binary number would be reverse of all remainders of step 1.

B) Convert the fractional part of decimal to binary equivalent

1. Multiply the fractional decimal number by 2.
2. Integral part of resultant decimal number will be first digit of fraction binary number.
3. Repeat step 1 using only fractional part of decimal number and then step 2.

C) Combine both integral and fractional part of binary number.

```
Let's take an example for n = 4.47 k = 3

Step 1: Conversion of 4 to binary
1. 4/2 : Remainder = 0 : Quotient = 2
2. 2/2 : Remainder = 0 : Quotient = 1
3. 1/2 : Remainder = 1 : Quotient = 0

So equivalent binary of integral part of decimal is 100.

Step 2: Conversion of .47 to binary
1. 0.47 * 2 = 0.94, Integral part: 0
2. 0.94 * 2 = 1.88, Integral part: 1
3. 0.88 * 2 = 1.76, Integral part: 1

So equivalent binary of fractional part of decimal is .011

Step 3: Combined the result of step 1 and 2.

Final answer can be written as:
100 + .011 = 100.011
```

(source: https://www.geeksforgeeks.org/convert-decimal-fraction-binary-number/?ref=rp)

Binary to decimal

Given an string of binary number **n**. Convert binary fractional **n** into it's decimal equivalent.

Examples:

```
Input: n = 110.101
Output: 6.625

Input: n = 101.1101
Output: 5.8125
```

A) Convert the integral part of binary to decimal equivalent

1. Multiply each digit separately from left side of radix point till the first digit by 2^0, 2^1, 2^2,... respectively.
2. Add all the result coming from step 1.
3. Equivalent integral decimal number would be the result obtained in step 2.

B) Convert the fractional part of binary to decimal equivalent

1. Divide each digit from right side of radix point till the end by 2^1, 2^2, 2^3, ... respectively.
2. Add all the result coming from step 1.
3. Equivalent fractional decimal number would be the result obtained in step 2.

C) Add both integral and fractional part of decimal number.

Let's take an example for n = 110.101

Step 1: Conversion of 110 to decimal

=> $110_2 = (1*2^2) + (1*2^1) + (0*2^0)$

=> $110_2 = 4 + 2 + 0$

=> $110_2 = 6$

So equivalent decimal of binary integral is 6.

Step 2: Conversion of .101 to decimal

=> $0.101_2 = (1*1/2) + (0*1/2^2) + (1*1/2^3)$

=> $0.101_2 = 1*0.5 + 0*0.25 + 1*0.125$

=> $0.101_2 = 0.625$

So equivalent decimal of binary fractional is 0.625

Step 3: Add result of step 1 and 2.

=> 6 + 0.625 = 6.625

(source: https://www.geeksforgeeks.org/convert-binary-fraction-decimal/)

Floating-point number system cont.

- The string of base β digits $d_0 d_1 \cdots d_{t-1}$ is called mantissa or significand
- $d_1 d_2 \cdots d_{t-1}$ is called fraction
- A common way of expressing x is

$$\pm\, d_0.d_1 \cdots d_{t-1} \times \beta^e$$

- A FP number is normalized if d_0 is nonzero
  denormalized otherwise

Floating-point number system cont.

Example 1. Consider the FP $(10, 3, -2, 2)$.

- Numbers are of the form

$$d_0.d_1d_2 \times 10^e, \quad d_0 \neq 0,\ e \in [-2, 2]$$

- largest positive number 9.99×10^2
- smallest positive normalized number 1.00×10^{-2}
- smallest positive denormalized number 0.01×10^{-2}
- denormalized numbers are e.g. 0.23×10^{-2}, 0.11×10^{-2}
- 0 is represented as 0.00×10^0

IEEE 754

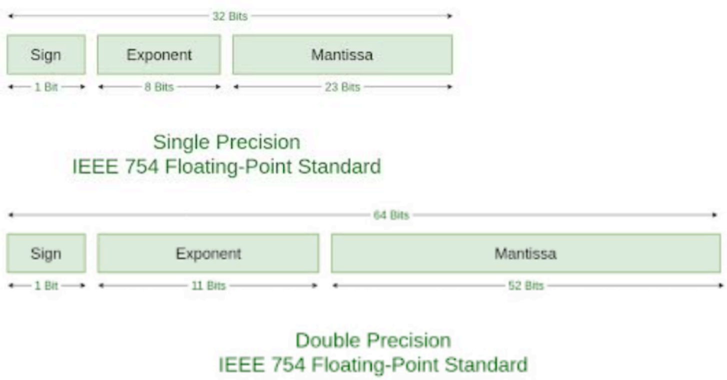
- IEEE 754 standard for FP arithmetic (1985)
- IEEE 754-2008, IEEE 754-2019
- Most common (binary) single and double precision since 2008 half precision

	bits	t	L	U	ϵ_{mach}
single	32	24	-126	127	1.2×10^{-7}
double	64	53	-1022	1023	2.2×10^{-16}

	range	smallest normalized	smallest denormalized
single	$\pm 3.4 \times 10^{38}$	$\pm 1.2 \times 10^{-38}$	$\pm 1.4 \times 10^{-45}$
double	$\pm 1.8 \times 10^{308}$	$\pm 2.2 \times 10^{-308}$	$\pm 4.9 \times 10^{-324}$

(These are $\approx$ values)

IEEE 754 cont.



(From `https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/`)

- sign 0 positive, 1 negative
- exponent is biased
- first bit of mantissa is not stored, sticky bit, assumed 1

IEEE 754 cont.

Single precision

- `Inf`
  - sign: 0 for +`Inf`, 1 for -`Inf`
  - biased exponent: all 1's, 255
  - fraction: all 0's
- `NaN`
  - sign: 0 or 1
  - biased exponent: all 1's, 255
  - fraction: at least one 1
- 0
  - sign: 0 for $+0$, 1 for -0
  - biased exponent: all 0's
  - mantissa: all 0's
- FP numbers
  - biased exponent: from 1 to 254, bias: 127
  - actual exponent: $1 - 127 = -126$ to $254 - 127 = 127$

IEEE 754 cont.

Double precision

- bias 1023
- biased exponent: from 1 to 2046
- actual exponent: from -1022 to 1023
- rest similar to single

One thing to remember

actual exponent = biased exponent – bias

biased exponent = actual exponent + bias

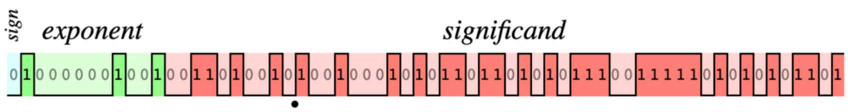
HARD TO LEARN? NO WORRY. A PLAYGROUND FOR FLOATING POINT!

Don’t forget to click “toggle details” :)

https://bartaz.github.io/ieee754-visualization/

1234.5678

toggle details



$n = 1234.5678$

$= +1 \times (2^{10} + 2^{7} + 2^{6} + 2^{4} + 2^{1} + 2^{-1} + 2^{-4} + 2^{-8} + 2^{-10} + 2^{-12} + 2^{-13} + 2^{-15} + 2^{-16} + 2^{-18} + 2^{-20} + 2^{-22} + 2^{-23} + 2^{-24}$

$= -1^{0} \times 2^{10} \times (2^{0} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-9} + 2^{-11} + 2^{-14} + 2^{-18} + 2^{-20} + 2^{-22} + 2^{-23} + 2^{-25} + 2^{-26} + 2^{-28} + 2^{-30} + 2^{-32} +$

$= -1^{0} \times 2^{1033-1023} \times (1 + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-9} + 2^{-11} + 2^{-14} + 2^{-18} + 2^{-20} + 2^{-22} + 2^{-23} + 2^{-25} + 2^{-26} + 2^{-28} + 2^{-30} + 2$

$= -1^{s} \times 2^{e-1023} \times (1 + \sum_{i=0}^{51} b_{52-i}\, 2^{-i})$

Hardware Implementation