Block Two: The Information Layer

Yangtao Ge

June 5, 2019

1 Chapter 2: Binary Value and Number System

Abstract

This chapter describes binary values – the way in which computer **hardware** represents and manages information. It also puts the binary value in all number system.

1.1 Number and Computing

Some definitions of Numbers:

- Number: A unit of an abstract mathematical system subject to the laws of arithmetic (succession, addition and multiplication).
- Natural number: The number **0** and any number obtained by reaptedly adding to 1 to 1.
- Negative number: A value less than zero and with a sign oppsite to its **positive counterpart**
- Rational number: An integer or the quotient of two integers (division by zero included)

1.2 Positional Notation

Some definitions of Base:

- Base: The foundational value of a number system, which dictates **number digits** and the **value of digit Position**
- Positional notation: A way of expressing number in different base system in a following way:

  $$d_n * R^{n-1} + d_{n-1} * R^{n-2} + ... + d_2 * R + d_1 \tag{1}$$

  where **Base-R** has n digits and d_i represents the digit in the ith position

Watch out the digit in a number. e.g. 2074 does not have base **less than Base-8** because digit 7 is used here.

2 digits is needed to represent the base value. e.g. 10 is <u>ten</u> in decimal. 10 is <u>eight</u> in base 8. 10 is <u>two</u> in binary.

Carry and borrow system is also applied to other base system. However, the value represented binary these carries and borrows means the **value of the base**.

All power of 2 number system can be transfered to **binary**, then to **decimal**. Examples are as follows:

$$\underline{\text{count every 4 digits for Hex}}$$

$$1010110 = 101(5)\ \&\ 0110(6)$$

$$\underline{\text{count every three digits for Oct}}$$

$$101010111100 = 101(5)\ \&\ 010(2)\ \&\ 111(7)\ \&\ 100(4)$$

Algorithm for Base 10 to Other Bases is as follows:

```
WHILE (the quotient is not zero):
    Divide the decimal number by the new base
    Make the reminder the next digit to the left in the answer
    Replace the decimal number with the quotient
```

This algorithm shows that:

- The production of new number is **from right to left**
- Quotient is repeatedly used, reminder is the **answer**

some definitions about bit:

- binary digit: A digit in the **binary number** system
- bit: Binary digit
- byte: **Eight** binary digits
- word: A group of one or more bytes
  the number of bits in a word = word length of the computer

2 Chapter 3: Data Representation

Abstract

This chapter includes how to store a certain type of information and represent in a computer environment

2.1 Data and Computers

Some definitions related to data:

- Data: basic value and facts
- Information: Organized data and can provide **useful solutions** to problems
- Multimeadia: Sevral different media types i.e. Numbers, Text, Audio, images and etc.
- Bandwidth: The number of bits or bytes that can be transmitted from one place to another within a fixed time

- Data compression: shrink the size of the data
- Compression ratio:

$$Ratio = \frac{Compressed\ Size}{Original\ Size} \tag{2}$$

  $0 < Ratio < 1$, closer to zero $\rightarrow$ tighter the compression

- Lossless: Without any Loss in the process of compaction
- Lossy: Is lost in the process of compaction

Real world is **infinite**, but computer is **finite**

Some definitions about types of data:

- Analog data: A **continuous** representation of data e.g. mercury thermometer (smooth wave)
- Digital data: A **discrete** representation of data e.g. button (square wave)

In computer:

- Analog Data $\xrightarrow{\text{digitize}}$ Digital Data
- use **binary** system to represent them

Degraded: Electronic signals degrades as they move down a line (**Threshold**)

Some definitions about Digital signals:

- Pulse-Code Modulation (PCM): Variation in a signal that jumps sharply between two **extremes**
- Reclocked: The act of reasserting an original digital signal before **too much degreadation occurs**

Analog vs Digital: (need review)

Analog degrades $\rightarrow$ in-range value $\rightarrow$ valid $\rightarrow$ information lost

Digital degrades $\rightarrow$ PCM $\rightarrow$ high to low $\rightarrow$ reclocked $\rightarrow$ information saved

n bits can represent 2^n things.
Increase the number of bits by 1 $\Rightarrow$ **double** the number of things we can represent

2.2 Representing Numeric Data

2.2.1 Negative Values

<u>The work flow is:</u>
Sign-Magnitude Representation $\rightarrow$ Fixed-sized Numbers $\rightarrow$ Two's Complement

- Sign-Magnitude Representation: "value + sign"
  Problem: Will have **two** representation of 0 (+0 & -0)

- Fixed-sized Numbers: use half of the integers to represent negatives
  Method: Add the number together and **dicard** any carries

$$Negative(I) = 10^k - I \tag{3}$$

  Problem: Can't be represnet in computer

- Two's Complement: use certain number of bits to represent a integer and <u>leftmost</u> one bit for representing **sign** e.g. -(2) is 11111110
  Method: **invert** the bits and **add 1**

$$Negative(I) = 2^k - I \tag{4}$$

Overflow occurs when the value that we compute cannot fit into <u>the number of bits</u> we have allocated for the result
e.g. 01111111(127) + 00000011(3) = 10000010(-126) is not +130

2.2.2 Real Numbers

Different from Math: all noninteger values $\Leftrightarrow$ Real Number

Radix means the **dot** that separates the whole part from the fractional part in a real number in *any base*

Floating Point means a representation of a real number that keeps track of the **sign**, **mantissa**, and **exponent**

Base-10:

$$R = sign * mantissa * 10^{exp} \tag{5}$$

Base-2:

$$R = sign * mantissa * 2^{exp} \tag{6}$$

Floating Point needs 64 bits: $64 = 1(sign) + 11(exponent) + 52(mantissa)$ i.e. double precision

Algorithm Converting fractional parts from base-10 to other:

```
WHILE (the fractional part is not zero):
    Multiply the fractional part by the new base
    Make the whole part the next digit to the left in the answer
    Replace the fractional part with the result of multiplication
```

Noticed that:

- it is possible that the loop will **never end** → precision problems
- instead of division, **multiplication** is used here
- More detail method of computing the Floating point is **NOT** included in this book