Chap. 3 Data Representation

- 3-1 Data Types

  - Binary information is stored in ***memory*** or ***processor registers***
  - Registers contain either ***data*** or ***control information***
    - ***Data*** are numbers and other binary-coded information
    - ***Control information*** is a bit or a group of bits used to specify the sequence of command signals
  - Data types found in the registers of digital computers
    - ***Numbers*** used in arithmetic computations
    - ***Letters*** of the alphabet used in data processing
    - ***Other discrete symbols*** used for specific purpose
  - Number Systems
    - ***Base*** or ***Radix r system*** : uses distinct symbols for ***r digits***
    - Most common number system :Decimal, Binary, Octal, Hexadecimal
    - Positional-value(weight) System : $\mathbf{r^2\ r^1 r^0 . r^{-1}\ r^{-2}\ r^{-3}}$
      - » Multiply each digit by an integer power of r and then form he sum of all weighted digits

- Decimal System/Base-10 System
  - Composed of 10 symbols or numerals(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
- Binary System/Base-2 System
  - Composed of 10 symbols or numerals(0, 1)
  - Bit = Binary digit
- Hexadecimal System/Base-16 System : Tab. 3-2
  - Composed of 16 symbols or numerals(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)
- Binary-to-Decimal Conversions

$1011.101_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^o) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

$= 8_{10} + 0 + 2_{10} + 1_{10} + 0.5_{10} + 0 + 0.125_{10}$

$= 11.625_{10}$

- Decimal-to-Binary Conversions

Repeated division(See p. 69, ***Fig. 3-1***)

37 / 2 = 18 remainder 1 (binary number will end with 1) : **LSB**

18 / 2 = 9 remainder 0

9 / 2 = 4 remainder 1

4 / 2 = 2 remainder 0

2 / 2 = 1 remainder 0

1 / 2 = 0 remainder 1 (binary number will start with 1) : **MSB**

Read the result upward to give an answer of $37_{10} = 100101_2$

0.375 x 2 = 0.750 integer 0 **MSB**

0.750 x 2 = 1.500 integer 1 .

0.500 x 2 = 1.000 integer 1 **LSB**

Read the result downward $.375_{10} = .011_2$

Hex-to-Decimal Conversion

$2AF_{16} = (2 \times 16^2) + (10 \times 16^1) + (15 \times 16^o)$

$= 512_{10} + 160_{10} + 15_{10}$

$= 687_{10}$

Decimal-to-Hex Conversion

423_{10} / 16 = 26 remainder 7 (Hex number will end with 7) : **LSB**

26_{10} / 16 = 1 remainder 10

1_{10} / 16 = 0 remainder 1 (Hex number will start with 1) : **MSB**

Read the result upward to give an answer of $423_{10} = 1A7_{16}$

Table 3-2

Hex	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Hex-to-Binary Conversion

$9F2_{16} = 9 \quad F \quad 2$

$\downarrow \quad \downarrow \quad \downarrow$

$= 1001 \quad 1111 \quad 0010$

$= 100111110010_2$

Binary-to-Hex Conversion

$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{0\,0\,1\,1}_{3}\ \underbrace{1\,0\,1\,0}_{A}\ \underbrace{0\,1\,1\,0}_{6}$

$= 3A6_{16}$

Binary-Coded-Decimal Code

- Each digit of a decimal number is represented by its binary equivalent

8	7	4	**(Decimal)**
↓	↓	↓	
1000	0111	0100	**(BCD)**

- Only the four bit binary numbers from 0000 through 1001 are used
- Comparison of BCD and Binary

137_{10} = 10001001_2 **(Binary) -** *require only 8 bits*

137_{10} = 0001 0011 0111_{BCD} **(BCD) -** *require 12 bits*

Alphanumeric Representation

- Alphanumeric character set
  - » 10 decimal digits, 26 letters, special character($, +, =,….)
- ASCII :
  - » Standard alphanumeric binary code.

- Complements
  - ***Complements*** are used in digital computers for simplifying the ***subtraction operation*** and for logical manipulation.
  - There are two types of complements for base r system
    - 1) r's complement 2) (r-1)'s complement
      - » Binary number : 2's or 1's complement
      - » Decimal number : 10's or 9's complement
  - (r-1)'s Complement
    - (r-1)'s Complement of N = $(r^n-1)-N$
      - » 9's complement of N=***546700***

        $(10^6-1)-546700= (1000000-1)-546700= 999999-546700$

        $= \mathbf{453299}$
      - » 1's complement of N=***101101***

        $(2^6-1)-101101= (1000000-1)-101101= 111111-101101$

        $= \mathbf{010010}$
  - r's Complement
    - r's Complement of N = r^n-N
      - » 10's complement of ***2389***= 7610+1= ***7611***
      - » 2's complement of ***1101100***= 0010011+1= ***0010100***

N : given number
r : base
n : digit number

546700(N) + 453299(9's com)
=999999

101101(N) + 010010(1's com)
=111111

* ***r's Complement***
(r-1)'s Complement +1 $=(r^n-1)-N+1= r^n-N$

- **Subtraction of Unsigned Numbers** (M-N), N≠0
  - 1) M + (r^n-N)
  - 2) M ≥ N : Discard end carry, Result = M-N
  - 3) M < N : No end carry, Result = - r's complement of (N-M)

» *Decimal Example)*

M ≥ N

72532(M) - 13250(N) = 59282

  72532
+ 86750 (10's complement of 13250)

1 59282

Discard End Carry

Result = *59282*

M < N

13250(M) - 72532(N) = -59282

  13250
+ 27468 (10's complement of 72532)

0 40718

No End Carry

Result = -(10's complement of 40718)
= -(59281+1) = *-59282*

» *Binary Example)*

X ≥ Y

1010100(X) - 1000011(Y) = 0010001

  1010100
+ 0111101 (2's complement of 1000011)

~~1 0010001~~

Result = *0010001*

X < Y

1000011(X) - 1010100(Y) = -0010001

  1000011
+ 0101100 (2's complement of 1010100)

0 1101111

Result = -(2's complement of 1101111)
= -(0010000+1) = *-0010001*

- Integer Representation

  - Signed-magnitude representation
  - Signed-1's complement representation
  - Signed-2's complement representation

+14	-14
0 0001110	1 0001110
0 0001110	1 1110001
0 0001110	1 1110010

- Arithmetic Addition
  - Addition Rules of Ordinary Arithmetic
    - » The signs are ***same*** : sign= ***common*** sign, result= ***add***
    - » The signs are ***different*** : sign= ***larger*** sign, result= ***larger-smaller***
  - Addition Rules of the signed 2's complement
    - » Add the two numbers including their sign bits
    - » Discard any carry out of the sign bit position

(-12) + (-13) = -25
(+12) + (+13) = +25

(+25) + (-37)
= 37 - 25 = -12

***Addition Exam)**

+ 6	00000110	- 6	11111010
+ 13	00001101	+ 13	00001101
+ 19	00010011	+ 7	00000111
+ 6	00000110	- 6	11111010
- 13	11110011	- 13	11110011
- 7	11111001	- 19	11101101

- Arithmetic Subtraction
  - Subtraction is changed to an Addition
    - » (± A) - (+ B) = (± A) + (- B)
    - » (± A) - (- B) = (± A) + (+ B)

*** Subtraction Exam)** (- 6) - (- 13) = +7
11111010 - 11110011 = 11111010 + 2's comp of 11110011
= 11111010 + 00001101
= 1 00000111 = +7

Discard End Carry

- Overflow
  - Two numbers of n digits each are added and the sum occupies n+1 digits
  - n + 1 bit cannot be accommodated in a register with a standard length of n bits.

Overflow

- An overflow may occur if the two numbers added are both positive or both negative
  - When two unsigned numbers are added
    - an overflow is detected from the end carry out of the MSB position
  - When two signed numbers are added
    - the MSB always represents the sign

      - the sign bit is treated as part of the number

      - the end carry does not indicate an overflow

*** Overflow Example)**

	out in		*out in*
carries	0 1	carries	1 0
+ 70	0 1000110	- 70	1 0111010
+ 80	0 1010000	- 80	1 0110000
+ 150	1 0010110	- 150	0 1101010

- 3-4 Floating-Point Representation

  - The floating-point representation of a number h
    - 1) Mantissa : signed, fixed-point number
    - 2) Exponent : position of binary(decimal) point

* Decimal + 6132.789

Fraction	*Exponent*
+0.6132789	+4

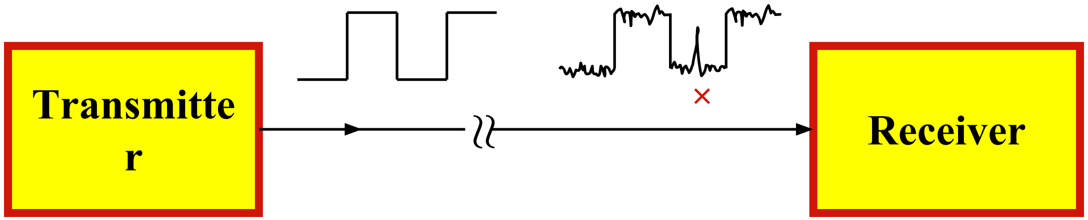
  - Scientific notation : $m \times r^e$ (+0.6132789 x 10^{+4}
    - ***m*** : mantissa, ***r*** : radix, ***e*** : exponent

Fraction	*Exponent*
01001110	000100

  - Example : $m \times 2^e = +(.1001110)_2 \times 2^{+4}$

  - Normalization
    - Most significant digit of mantissa is ***nonzero***

- 3-6 Error Detection Codes
  - Binary information transmitted through some form of communication medium is subject to external noise



  - Parity Bit
    - An extra bit included with a binary message to make the total number of 1's either odd or even.
  - Even-parity method
    - The value of the parity bit is chosen so that the total number of 1s *(including the parity bit)* is an ***even*** number

1 1 0 0 0 0 1 1

Added parity bit

  - Odd-parity method
    - Exactly the same way except that the total number of 1s is an ***odd*** number

1 1 0 0 0 0 0 1

Added parity bit