Chapter 2
Number Systems, Operations, and Codes

- What does 5132.13 really mean?
- Depends on the number base!
- Assuming base 10:

$$5132.13_{10} = \mathbf{5}\text{x}10^{3} + \mathbf{1}\text{x}10^{2} + \mathbf{3}\text{x}10^{1} + \mathbf{2}\text{x}10^{0} + \mathbf{1}\text{x}10^{-1} + \mathbf{3}\text{x}10^{-2}$$

- Assuming base 6:

$$5132.13_{6} = \mathbf{5}\text{x}6^{3} + \mathbf{1}\text{x}6^{2} + \mathbf{3}\text{x}6^{1} + \mathbf{2}\text{x}6^{0} + \mathbf{1}\text{x}6^{-1} + \mathbf{3}\text{x}6^{-2}$$

- We often use a subscript to indicate the base.

2.1 Decimal Numbers

- The position of each digit in a decimal number indicates the magnitude of the quantity represented and can be assigned a weight
- Example
  - 5236.71

$= 5\times1000+2\times100+3\times10+6\times1+7\times0.1+1\times0.01$

$= 5\times10^3+2\times10^2+3\times10^1+6\times10^0+7\times10^{-1}+1\times10^{-2}$

For an arbitrary decimal number

$$N = a_{n-1}a_{n-2}\cdots a_1 a_0 \bullet a_{-1}a_{-2}\cdots a_{-m}$$

$$= a_{n-1}\times 10^{n-1} + a_{n-2}\times 10^{n-2} + \cdots + a_1\times 10 + a_0\times 10^0 + a_{-1}\times 10^{-1}$$

$$+ a_{-2}\times 10^{-2} + \cdots + a_{-m}\times 10^{-m}$$

$$= \sum a_i \times 10^i$$

2.2 Binary Numbers

- The binary system with its two digits is a base-two system

DECIMAL NUMBER	BINARY NUMBER			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

$$N = a_{n-1}a_{n-2}\cdots a_1 a_0 \bullet a_{-1}a_{-2}\cdots a_{-m}$$

$$= a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \cdots + a_1\times 2^1 + a_0\times 2^0$$

$$+ a_{-1}\times 2^{-1} + a_{-2}\times 2^{-2} + \cdots + a_{-m}\times 2^{-m}$$

$$= \sum a_i \times 2^i$$

A simple binary counting application

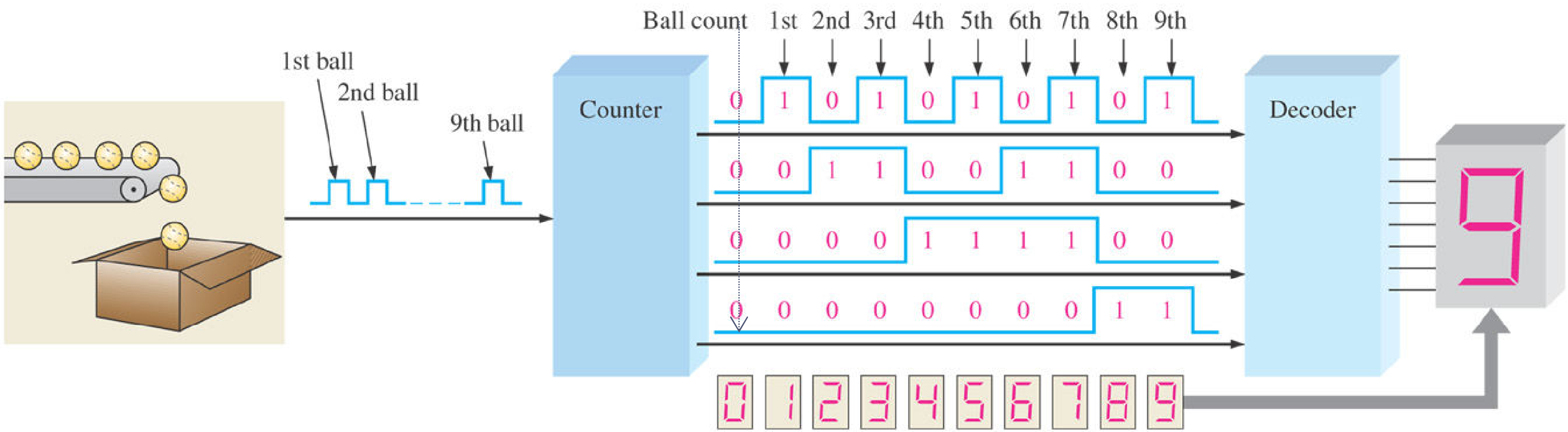


Figure 2–1 Illustration of a simple binary counting application.

- Hexadecimal Numbers
  - Widely used in computer and microprocessor applications
  - A(10),B(11),C(12),D(13),E(14),F(15);

$$N = a_{n-1}a_{n-2}\cdots a_1a_0 . a_{-1}a_{-2}\cdots a_{-m}$$

$$= a_{n-1}\times 16^{n-1} + a_{n-2}\times 16^{n-2} + \cdots + a_1\times 16 + a_0\times 16^0 + a_{-1}\times 16^{-1} + a_{-2}\times 16^{-2}$$

$$+ \cdots + a_{-m}\times 16^{-m}$$

$$= \sum a_i \times 16^i$$

Conversions

- Binary-to-decimal conversions
  - Adding the weights of all 1s in a binary number to get the decimal value
- Decimal-to-binary conversions

$$(S)_{10} \cdots \Rightarrow k_n k_{n-1} \ldots k_1 k_0 .\, k_{-1} k_{-2} \ldots k_{-m+1} k_{-m}$$

Decimal numbers

$$(S)_{10} = k_n 2^n + k_{n-1} 2^{n-1} + \cdots + k_1 2^1 + k_0 2^0$$

$$= 2(k_n 2^{n-1} + k_{n-1} 2^{n-2} + \cdots + k_1) + k_0$$

Decimal Fractions

$$(S)_{10} = k_{-1} 2^{-1} + k_{-2} 2^{-2} + \cdots + k_{-m} 2^{-m}$$

$$2(S)_{10} = k_{-1} + (k_{-2} 2^{-1} + k_{-3} 2^{-2} + \cdots + k_{-m} 2^{-m+1})$$

Conversion from Binary

Example

-- Convert 101011.11_2 to base 10:

$\mathbf{101011.11_2}$

$\mathbf{= 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}}$

= 32 + 0 + 8 + 0 + 2 + 1 + ½ + ¼

$\mathbf{= 43.75_{10}}$

Conversion from decimal to binary

2	173	1
2	86	0
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
	0	

$$(173)_{10} = (10101101)_2$$

$$\begin{array}{r} 0.8125 \\ \times\quad 2 \\ \hline 1.6250 \end{array} \qquad 1 = k_{-1}$$

$$\begin{array}{r} 0.6250 \\ \times\quad 2 \\ \hline 1.2500 \end{array} \qquad 1 = k_{-2}$$

$$\begin{array}{r} 0.2500 \\ \times\quad 2 \\ \hline 0.5000 \end{array} \qquad 0 = k_{-3}$$

$$\begin{array}{r} 0.5000 \\ \times\quad 2 \\ \hline 1.0000 \end{array} \qquad 1 = k_{-4}$$

$(0.8125)_{10}=(0.1101)_2$

$$(173.8125)_{10} = (10101101.1101)_2$$

- Binary-to-hexadecimal conversion

$$\begin{array}{ccccc} & (0101 & 1110\bullet & 1011 & 0010)_2 \\ & \downarrow & \downarrow & \downarrow & \downarrow \\ = & (5 & E\bullet & B & 2)_{16} \end{array}$$

- Hexadecimal-to-binary conversion

$$\begin{array}{cccccc} & (8 & F & A\bullet & C & 6)_{16} \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ = & (1000 & 1111 & 1010\bullet & 1100 & 0110)_2 \end{array}$$

2.3 Signed Numbers （符号数）

- **Sign-magnitude form （原码/符号数值形式）**
- **1’s complement form（反码形式）**
- **2’s complement form（补码形式）**

Sign-magnitude Form

- A signed number consists of both sign and magnitude information
- The sign indicates whether a number is positive or negative
- The magnitude is the value of the number
- The sign bit
  - A ‘0’(zero) sign bit indicates a positive number and a ‘1’ sign bit indicates a negative number

$$\underset{\uparrow \atop \text{Sign}}{0}\underbrace{0011001}_{\text{Magnitude}} \qquad \underset{\uparrow \atop \text{Sign}}{1}\underbrace{0011001}_{\text{Magnitude}}$$

Example

Decimal number	Sign-magnitude form
25	0001 1001
-25	1001 1001
39	0010 0111
-39	1010 0111
0	0000 0000 1000 0000

Example

$$10100111 = ?$$

$$00100111 = ?$$

$$10100111 = (-1)^1 \times \left(1\times 2^5 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0\right) = -39$$

$$00100111 = (-1)^0 \times \left(1\times 2^5 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0\right) = 39$$

Range of Singed Integer Numbers That Can be represented

For signed-magnitude numbers, the range of values for n-bit numbers is

$$-\left(2^{n-1}\right)+1 \sim +\left(2^{n-1}-1\right)$$

WHY?

Example

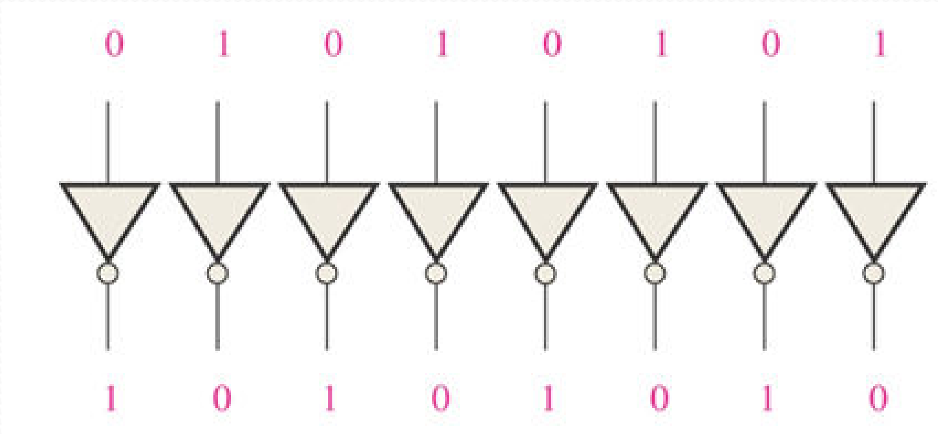
$$n=8: \quad -127 \sim 127$$

$$11111111 \sim 10000000 \quad 01111111 \sim 00000000$$

$$-127 \sim -0 \qquad\qquad 127 \sim +0$$

1's Complement Form of Signed Numbers

- Positive numbers: the same as the positive sign-magnitude numbers
- Negative numbers: the 1's complements of the corresponding positive numbers
  - Change all 1s to 0s and all 0s to 1s



Example

Decimal number	Sing-magnitude form	1' s complement form
25	0001 1001	0001 1001
-25	1001 1001	1110 0110
39	0010 0111	0010 0111
-39	1010 0111	1101 1000
0	0000 0000 1000 0000	0000 0000 1111 1111

Example $11011000_{1C} = ?$

$00100111_{1C} = ?$

$$11011000_{1C} = 1\times\left(-2^7\right) + 1\times 2^6 + 1\times 2^4 + 1\times 2^3 + 1 = -39$$

$$00100111_{1C} = 0\times\left(-2^7\right) + 1\times 2^5 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0 = 39$$

Range of Singed Integer Numbers That Can be represented

For 1's complement signed numbers, the range of values for n-bit numbers is

$$-\left(2^{n-1}\right)+1 \sim +\left(2^{n-1}-1\right)$$ **WHY?**

Example

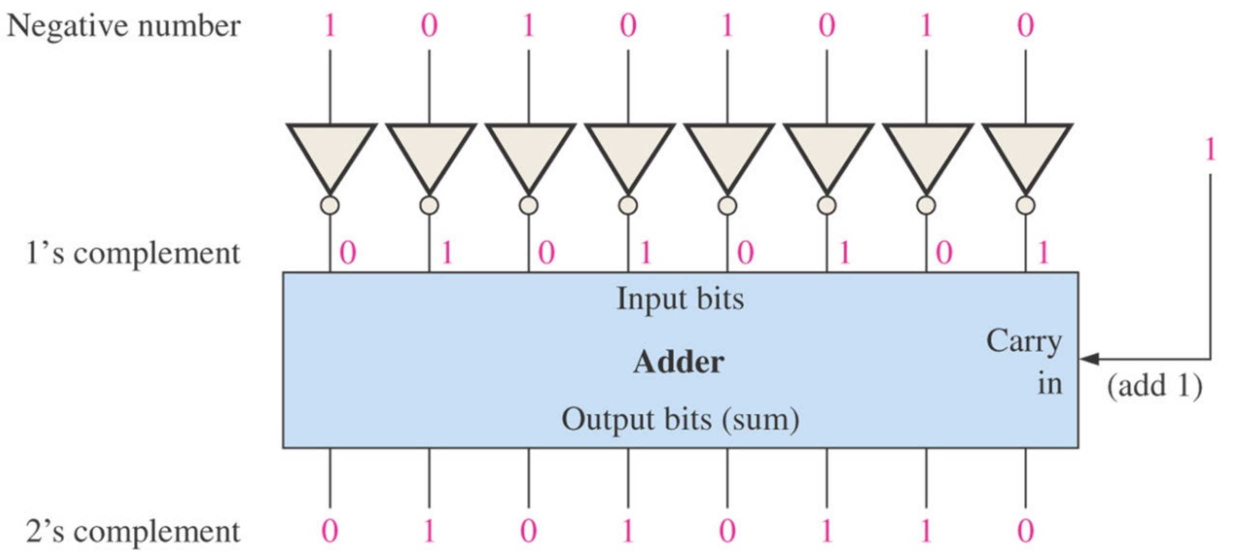
$$n=8: \quad -127 \sim 127$$

$$11111111 \sim 10000000 \qquad 01111111 \sim 00000000$$

$$-0 \sim -127 \qquad\qquad 127 \sim +0$$

2's Complement Form of Signed Numbers

- Positive numbers: the same as the sign magnitude and 1's complement forms
- Negative numbers: the 2's complements of the corresponding positive numbers
  - Adding 1 to the LSB of the 1's complement
  - 2's complement = (1's complement) +1



Example

Decimal number	Sing-magnitude form	1' s complement form	2's complement form
25	0001 1001	0001 1001	0001 1001
-25	1001 1001	1110 0110	1110 0111
39	0010 0111	0010 0111	0010 0111
-39	1010 0111	1101 1000	1101 1001
0	0000 0000 1000 0000	0000 0000 1111 1111	0000 0000

Example

$$11011001_{2C} = ?$$

$$00100111_{2C} = ?$$

$$11011001_{2C} = 1\times\left(-2^7\right) + 1\times 2^6 + 1\times 2^4 + 1\times 2^3 + 1\times 2^0 = -39$$

$$00100111_{2C} = 0\times\left(-2^7\right) + 1\times 2^5 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0 = 39$$

Range of Singed Integer Numbers That Can be represented

For 2's complement signed numbers, the range of values for n-bit numbers is

$$-\left(2^{n-1}\right) \sim +\left(2^{n-1}-1\right)$$

WHY?

Example

$$n = 8: \quad -128 \sim 127$$

$$11111111 \sim 10000000 \quad 01111111 \sim 00000000$$

$$-1 \sim -128 \qquad\qquad 127 \sim 0$$

Unsigned integer numbers (magnitude)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Signed integer numbers (sign-magnitude form)

-7	-6	-5	-4	-3	-2	-1	-0	0	1	2	3	4	5	6	7
1111	1110	1101	1100	1011	1010	1001	1000	0000	0001	0010	0011	0100	0101	0110	0111

Signed integer numbers (1's complement form)

-7	-6	-5	-4	-3	-2	-1	-0	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Signed integer numbers (2's complement form)

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Floating-point Numbers—IEEE754

- A floating-point number consists of two parts plus a sign.
- The ***mantissa***（*尾数部分*）: the part of a floating-point number that represents the magnitude of the number.
- The ***exponent***（*指数部分*）: the part of a floating-point number that represents the number of places that the decimal point (or binary point) is to be moved.
- Single-precision: 32 bits
- Double-precision: 64 bits
- Extended-precision: 80 bits

Example

1 0110 1001 0001 = 1.0110 1001 0001 x 2^{12}

(Assuming this is a positive number)

Exponent 12+**127**=139 ---------->1000 1011

Mantissa 0110 1001 0001

(The 1 left of the binary point is not included)

S	Exponent	Mantissa (Fraction)
0	1000 1011	01101001000100000000000
1 bit	8 bit	23 bit

Example

Determine the binary value of the following floating-point binary number.

S	Exponent	Mantissa (Fraction)
1	10010001	10001110001000000000000

$$\text{Number} = (-1)^S (1+F)\left(2^{E-127}\right)$$

$$= (-1)^1 (1.10001110001)\left(2^{145-127}\right)$$

$$= -1100011100010000000$$

S	Exponent	Mantissa (Fraction)
0/1	00000000	0000000000000000000000

S	Exponent	Mantissa (Fraction)
0/1	1111111111	0000000000000000000000

S	Exponent	Mantissa (Fraction)
0/1	1111111111	non zero

+/- 0

S	Exponent	Mantissa (Fraction)
0/1	00000000	0000000000000000000000

+/- inf

S	Exponent	Mantissa (Fraction)
0/1	1111111111	0000000000000000000000

NaN
Not a Number

S	Exponent	Mantissa (Fraction)
0/1	1111111111	non zero

2.4 Arithmetic Operations with Binary Numbers

- Addition
- Subtraction
- Multiplication
- Division

For Unsigned Numbers

Binary Addition

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 and carry **1** to the next column

Examples:

```
  0101     (5_10)
+ 0010     (2_10)
  ----
  0111     (7_10)
```

```
 1 1 1  <-------- Carries
  0101     (5_10)
+ 0011     (3_10)
  ----
  1000     (8_10)
```

$0101 \quad (5_{10})$
$+\ 0010 \quad (2_{10})$
$0111 \quad (7_{10})$

Carries: $1\ 1\ 1$
$0101 \quad (5_{10})$
$+\ 0011 \quad (3_{10})$
$1000 \quad (8_{10})$

Binary Addition

Add 6-bit numbers 45_{10} and 44_{10} in binary

```
  1  1 1   <-------------------- Carries
   101101      (45₁₀)
+  101100      (44₁₀)
  -------
  1011001      (89₁₀)
```

If the operands are *unsigned*, you can use the final carry-out as the MSB of the result.

Adding 2 *k*-bit numbers ⇔ *k*+1 bit result

More Binary Addition

```
  1 1 1 1
   1111    (15₁₀)
+  0001    (1₁₀)
   ----
   0000    (0₁₀)
```

If you don't want a 5-bit result, just keep the lower 4 bits.

Here, 4 bits is insufficient to hold the result (16).

It *rolls over* back to 0.

For Signed Numbers

Negative Binary Numbers

- Several ways of representing negative numbers
- Most obvious is to add a sign (+ or -) to the binary integer

Sign-Magnitude

0 is '+'

1 is '-'

Number	Sign	Magnitude	Full Number
+1	0	0001	00001
-1	1	0001	10001
+5	0	0101	00101
-5	1	0101	10101
+0	0	0000	00000
-0	1	0000	10000

Easy to interpret number value

Sign-Magnitude Examples

```
      0 0 0
  0 0011      (+3₁₀)
+ 0 0100      (+4₁₀)
  ------
  0 0111      (+7₁₀)
```

$$\begin{array}{rrl} & {\scriptstyle 0\,0\,0} & \\ & 0\ 0011 & (+3_{10}) \\ + & 0\ 0100 & (+4_{10}) \\ \hline & 0\ 0111 & (+7_{10}) \end{array}$$

$$\begin{array}{rrl} & {\scriptstyle 0\,0\,0} & \\ & 1\ 0011 & (-3_{10}) \\ + & 1\ 0100 & (-4_{10}) \\ \hline & 1\ 0111 & (-7_{10}) \end{array}$$

Signs are the same, just add the magnitudes

Another Sign-Magnitude Example

Signs are different: determine which has larger magnitude

$$\begin{array}{rl} 0\ 0101 & (+5_{10}) \\ \underline{+\ 1\ 0011} & (-3_{10}) \end{array}$$

Put larger magnitude number on top

$$\begin{array}{rl} 0\ 0101 & (+5_{10}) \\ \underline{-\ 1\ 0011} & (-3_{10}) \\ 0\ 0010 & (+2_{10}) \end{array}$$

Subtract

Result has sign of larger magnitude number…

Yet Another Sign-Magnitude Example

Signs are different: determine which has larger magnitude

```
  0 0010     (+2₁₀)
+ 1 0101     (-5₁₀)
```

$$\begin{array}{rl} 0\ 0010 & (+2_{10}) \\ \underline{+\ 1\ 0101} & (-5_{10}) \end{array}$$

Put larger magnitude number on top

Subtract

$$\begin{array}{rl} 1\ 0101 & (-5_{10}) \\ \underline{-\ 0\ 0010} & (+2_{10}) \\ 1\ 0011 & (-3_{10}) \end{array}$$

Result has sign of larger magnitude number…

Sign-Magnitude Form

- Addition requires two separate operations
  - addition
  - subtraction
- Several decisions:
  - Signs same or different?
  - Which operand is larger?
  - What is sign of final result?
- Two zeroes (+0, -0)

Sign-Magnitude

- Advantages
  - Easy to understand
- Disadvantages
  - Two different 0s
  - Hard to implement in logic

One’s Complement

- Positive numbers are the same as sign-magnitude
- -N is represented as the 1’s complement of N:

$$-N = N'$$

Number	1's Complement
+1	0001
-1	1110
+5	0101
-5	1010
+0	0000
-0	1111

One's Complement Examples

```
  000101     (5₁₀)
+ 010100     (20₁₀)
  011001     (25₁₀)
```

$$\begin{array}{rl} 000101 & (5_{10}) \\ +\ 010100 & (20_{10}) \\ \hline 011001 & (25_{10}) \end{array}$$

$$\begin{array}{rl} {\scriptstyle 1\,1\,0\,0} & \\ 0101 & (5_{10}) \\ +\ 1100 & (-3_{10}) \\ \hline 0001 & \\ +\ 1 & \\ \hline 0010 & (+2_{10}) \end{array}$$

If there is a carry out on the left, it must be *wrapped around* and added back in on the right

One's Complement

- Addition complicated by end around carry
- No decisions (unlike sign-magnitude)
- Still two zeroes (+0, -0)

One’s Complement

- Advantages
  – Easy to generate –N
  – Only one addition process
- Disadvantages
  – End around carry
  – Two different 0s

Two's Complement

- Treat positional digits differently

$$0111_{2C} = -0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 7_{10}$$

$$1111_{2C} = -1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = -1_{10}$$

Most significant bit (MSB) given negative weight…
Other bits same as in unsigned

Two's Complement

Number	2's Complement
+1	0001
-1	1111
+5	0101
-5	1011
+0	0000
-0	none

Sign-Extension in 2's Complement

- To make a *k*-bit number wider
  - replicate sign bit

$110_{2C} = -1 \times 2^2 + 1 \times 2^1 = -2_{10}$

$1110_{2C} = -1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = -2_{10}$

$11110_{2C} = -1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = -2_{10}$

More Sign-Extension

$010_{2C} = 1 \times 2^1 = 2_{10}$

$0010_{2C} = 1 \times 2^1 = 2_{10}$

$00010_{2C} = 1 \times 2^1 = 2_{10}$

Works for both positive and negative numbers!

Negating a 2's Complement Number

1. Invert all the bits
2. Add 1

$$+2_{10} = 0010_{2C} \rightarrow 1101 + 1 = 1110_{2C} = -2_{10}$$

$$-2_{10} = 1110_{2C} \rightarrow 0001 + 1 = 0010_{2C} = +2_{10}$$

Two's Complement Addition

```
 0 0 1
  0101     (+5₁₀)
+ 0001     (+1₁₀)
  ----
  0110     (+6₁₀)
```

```
 1 1 1
  1011     (-5₁₀)
+ 1111     (-1₁₀)
  ----
  1010     (-6₁₀)
```

```
 1 1 0
  0110     (+6₁₀)
+ 1111     (-1₁₀)
  ----
  0101     (+5₁₀)
```

Operation is same as for unsigned.
Same rules, same procedure.

Interpretation of operands and results are different.

Two's Complement

- Addition always the same
- Only 1 zero
- Negation somewhat complicated

- ***The representation of choice***

2.5 Binary Coded Decimal (BCD)

- A way to express each of the decimal digits with a binary code
- The 8421 Code
    - 0 --------> 0000
    - 1 --------> 0001
    - …
    - 9 --------> 1001

Decimal Digit	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Convert $\mathbf{2496}_{10}$ to BCD Code

2	**4**	**9**	**6**
↓	↓	↓	↓
0 0 1 0	**0 1 0 0**	**1 0 0 1**	**0 1 1 0**

Note this is very different from converting to binary which yields:

$$\mathbf{1\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0}_2$$

十进制的数字：123

- **BCD的表示形式**：0001 0010 0011，12bit = 1.5个字节
- int i = 123，那么int是4个字节数字，在内存表示形式：
  0x0000007b = 0000 0000 0000 0000 0000 0000 0111 1011
- 字符形式表示：char s[] = "123";即s[0] = '1',s[1] = '2' ,s[2] = '3'。那么在内存中表示形式:
  '1'：0001 0011
  ‘2'：0010 0011
  ‘3'：0011 0011
  字符形式“123”，共3个字节。

因此，BCD 编码是一种压缩算法，减少了数字在内存中的占有量

ASCII Code

- ASCII ➔ *American Standard Code for Information Interchange*
- ASCII is a 7-bit code used to represent letters, symbols, and terminal codes
- There are also Extended ASCII codes, represented by 8-bit numbers
- Terminal codes include such things as:

    Tab (TAB)
    Line feed (LF)
    Carriage return (CR)
    Backspace (BS)
    Escape (ESC)
    And many more!

ASCII Code

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Extended ASCII Code

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

ASCII Code (partial)

Character	ASCII Code
c	1 1 0 0 0 1 1
d	1 1 0 0 1 0 0
e	1 1 0 0 1 0 1
f	1 1 0 0 1 1 0
g	1 1 0 0 1 1 1
h	1 1 0 1 0 0 0
I	1 1 0 1 0 0 1
j	1 1 0 1 0 1 0
k	1 1 0 1 0 1 1
l	1 1 0 1 1 0 0
m	1 1 0 1 1 0 1
n	1 1 0 1 1 1 0
o	1 1 0 1 1 1 1
p	1 1 1 0 0 0 0
q	1 1 1 0 0 0 1

Convert “help” to ASCII

h	**e**	**l**	**p**
↓	↓	↓	↓
1101000	**1100101**	**1101100**	**1111000**
↓	↓	↓	↓
0x68	**0x65**	**0x6C**	**0x70**

2.6 Digital Codes

- The Gray Code
  - Unweighted and not an arithmetic code
  - No specific weights assigned to the bit positions
  - Important feature: exhibits only a single bit change from one code word to the next in sequence

Table 2-6 Four-bit Gray code

DECIMAL	BINARY	GRAY CODE	DECIMAL	BINARY	GRAY CODE
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

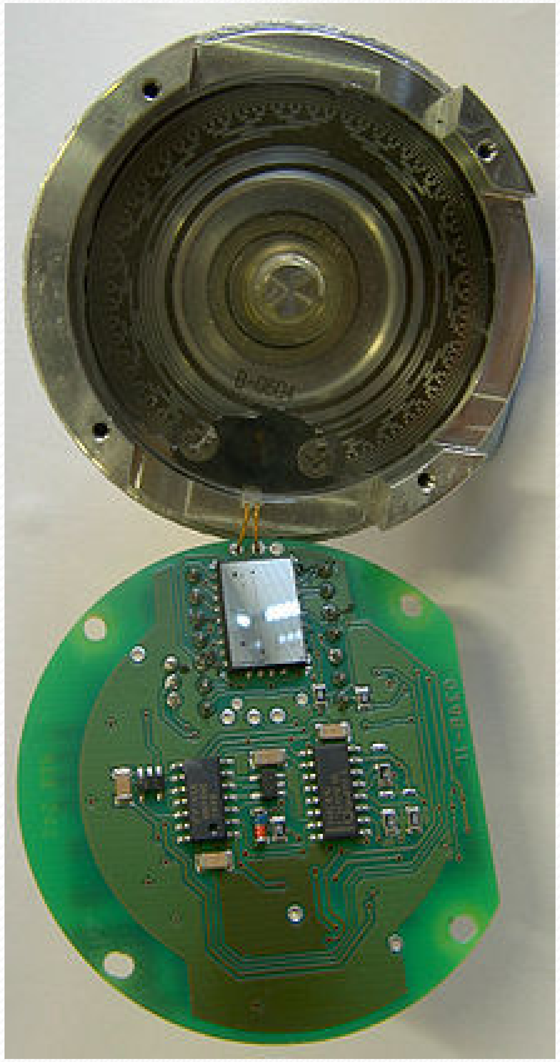
- Binary-to-Gary Code Conversion
  - The MSB in the Gray code is the same as the corresponding MSB in the binary number
  - **Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit**
  - Discard carries
  - Example

10110 (binary) ---------> 11101 (Gray code)

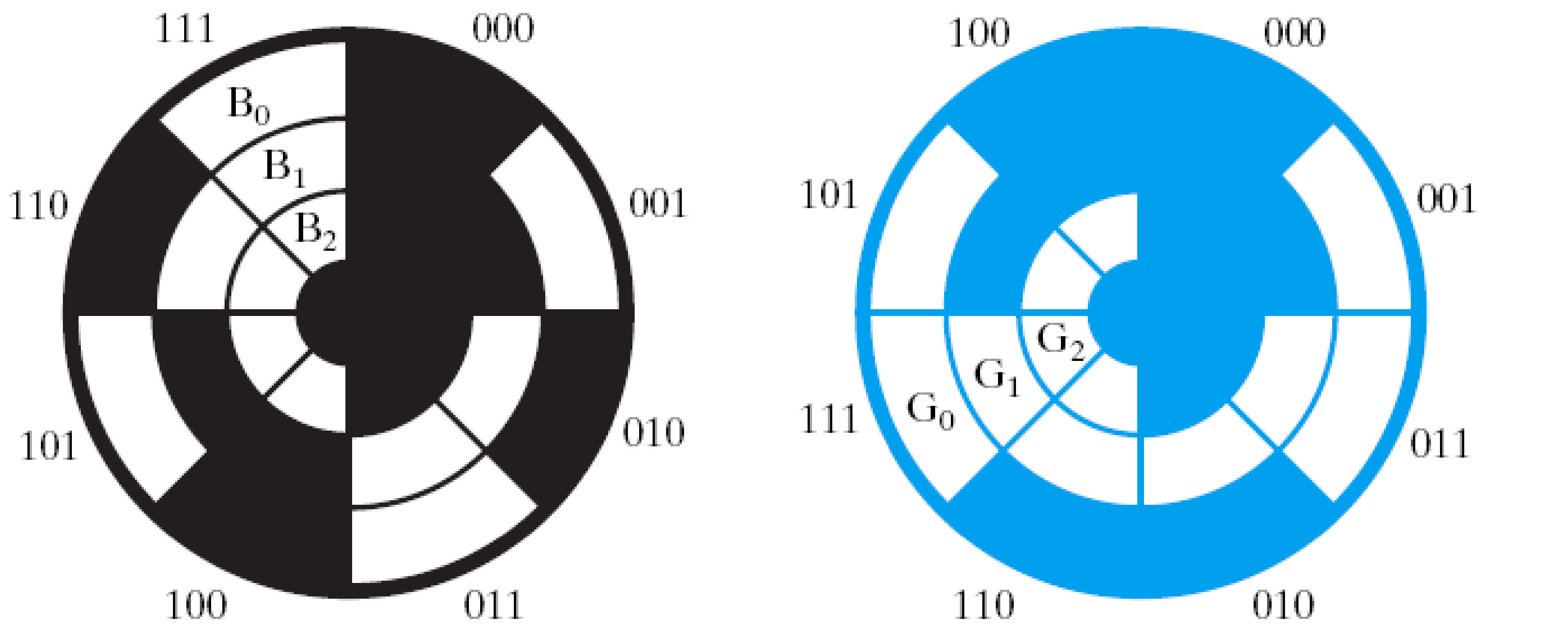
- Gray-to-Binary Conversion
  - The MSB in the binary code is the same as the corresponding bit in the Gray code
  - **Add each binary code bit generated to the Gray code bit in the next adjacent position**
  - Discard carries
  - Example

11011 (Gray code) --------> **10010** (binary)

An application of Gray Code



Rotary encoder （旋转编码器）



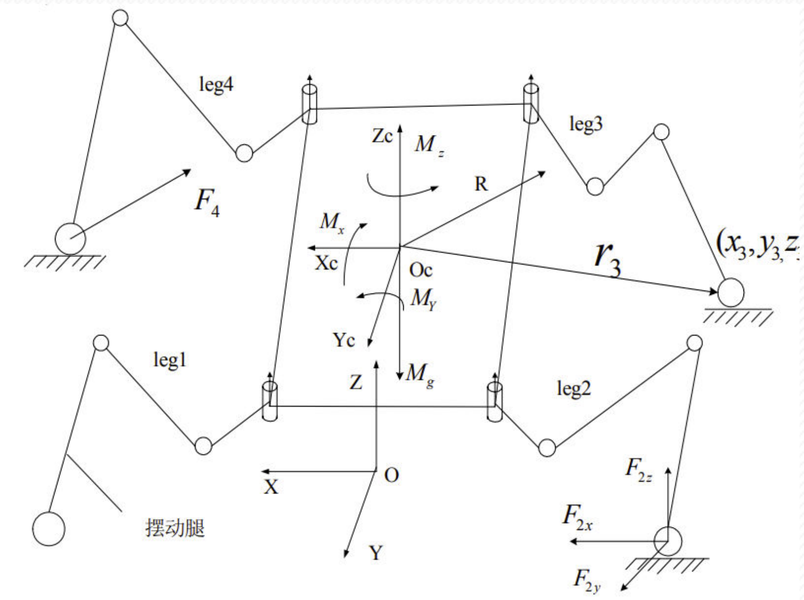
(a) Binary Code for Positions 0 through 7 (b) Gray Code for Positions 0 through 7

Standard Binary Encoding				
Sector	Contact 1	Contact 2	Contact 3	Angle
0	off	off	off	0° to 45°
1	off	off	ON	45° to 90°
2	off	ON	off	90° to 135°
3	off	ON	ON	135° to 180°
4	ON	off	off	180° to 225°
5	ON	off	ON	225° to 270°
6	ON	ON	off	270° to 315°
7	ON	ON	ON	315° to 360°

Gray Coding				
Sector	Contact 1	Contact 2	Contact 3	Angle
0	off	off	off	0° to 45°
1	off	off	ON	45° to 90°
2	off	ON	ON	90° to 135°
3	off	ON	off	135° to 180°
4	ON	ON	off	180° to 225°
5	ON	ON	ON	225° to 270°
6	ON	off	ON	270° to 315°
7	ON	off	off	315° to 360°

旋转编码器：在电梯运行过程中，通过旋转编码器检测、软件实时计算以下信号——电梯所在层楼位置、换速点位置、平层点位置，从而进行楼层计数、发出换速信号和平层信号。

步进电机



四足步行机器人力学模型

Summary

- Number systems
  - Binary, decimal, hexadecimal
- Singed number
- Arithmetic operation
- Codes

HW (Edition11) HW (Edition10)

- Page 54~57
  - 11
  - 15
  - 22
  - 23
  - 31
  - 51
  - 56

- Page 59~62
  - 11
  - 15
  - 22
  - 23
  - 31
  - 51
  - 56