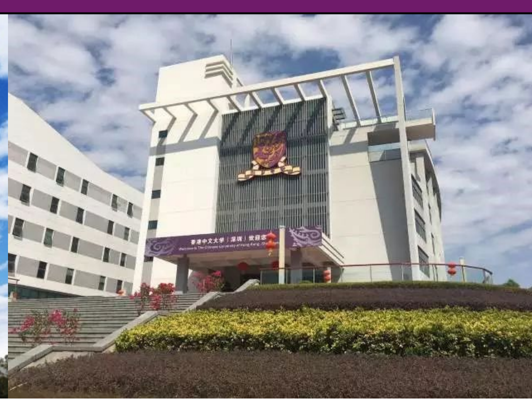
EIE 2050 Digital Logic and Systems

Chapter 2 : Number Systems, Operations, and Codes

Simon Pun, Ph.D.

Announcements

- ❑ NO tutorials and homework for the second week EITHER;

Last Week

- ❑ Analog versus Digital
- ❑ Bits (Binary digits), Logic Levels and Digital Waveforms
- ❑ Basic logic functions: NOT, AND and OR
- ❑ Combinational & sequential logic functions: comparator, adder, encoder/decoder, (de)multiplexer, flip-flops, registers, counter
- ❑ Integrated circuit (IC): Programmable versus Fixed-function
  - ◆ Package: Surface-mounted and Through-hole
  - ◆ Programmable: PLD (SPLD and CPLD) and FPGA
  - ◆ Fixed-function : SSI/MSI/VLSI/ULSI
- ❑ Instruments: Oscilloscope, logic analyzer, signal/waveform gen., digital multimeter, DC power supply.

Decimal versus Binary Numbers

- Weighted number systems : Decimal, Binary, Hexadecimal and Octal

Decimal Numbers

Each digit takes a value btw 0~9

The digit 2 has a weight of 10 in this position.

The digit 3 has a weight of 1 in this position.

2 3

2×10 + 3×1

20 + 3

23

Fractional numbers

$$10^2\ 10^1\ 10^0.10^{-1}\ 10^{-2}\ 10^{-3}\ \ldots$$

Decimal point

$$\begin{aligned}568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2})\\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01)\\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03}\end{aligned}$$

Binary Numbers

Each digit takes a value either 0 or 1

The digit ~~2~~ 1 has a weight of ~~10~~ 2^1 in this position.

The digit ~~3~~ 1 has a weight of ~~1~~ 2^0 in this position.

1 1

1×2^1 + 1×2^0

2 + 1

3

Fractional numbers

$$2^{n-1}\ \ldots\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}\ \ldots\ 2^{-n}$$

Binary point

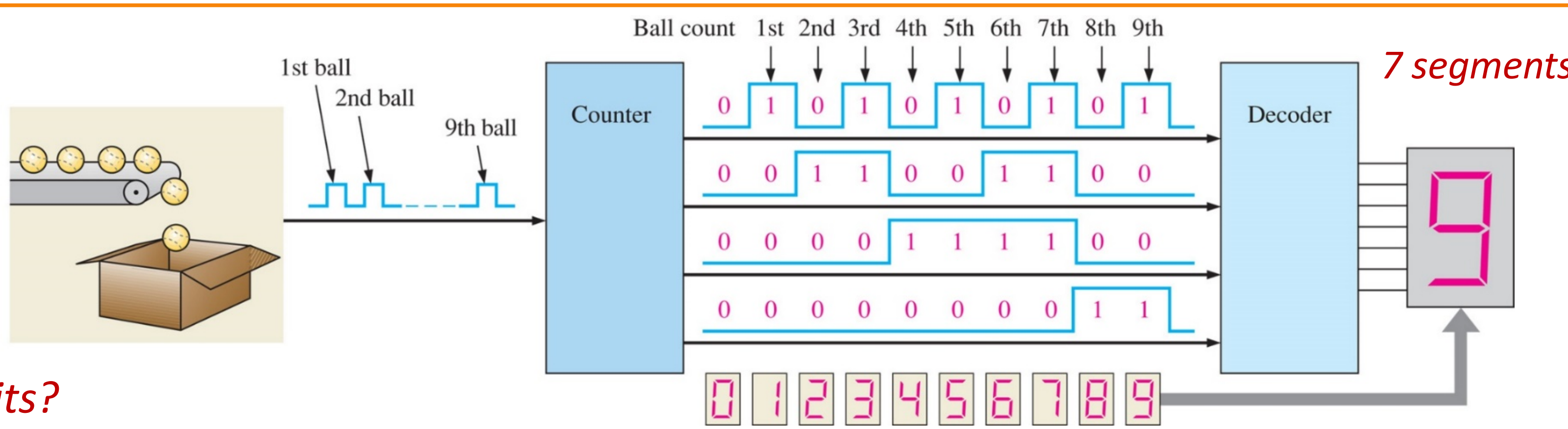
$$\begin{aligned}0.1011_2 &= 2^{-1} + 2^{-3} + 2^{-4}\\ &= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}\end{aligned}$$

Counting in Binary

- Largest decimal number can be represented by n bits : $2^n - 1$
- MSB: Most Significant Bit
- LSB : Least Significant Bit
- Example: Putting 9 balls in each box with four bits while displaying the # of balls in the current box

TABLE 2–1

Decimal Number	MSB	Binary Number		LSB
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



Why 4 bits?

Decimal-to-Binary Conversion

❑ Sum-of-Weights Method

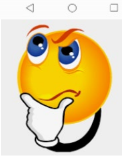
$19 = 16 + 2 + 1$

$= 2^4 + 2^1 + 2^0 \rightarrow 10011$

$0.625 = 0.5 + 0.125$

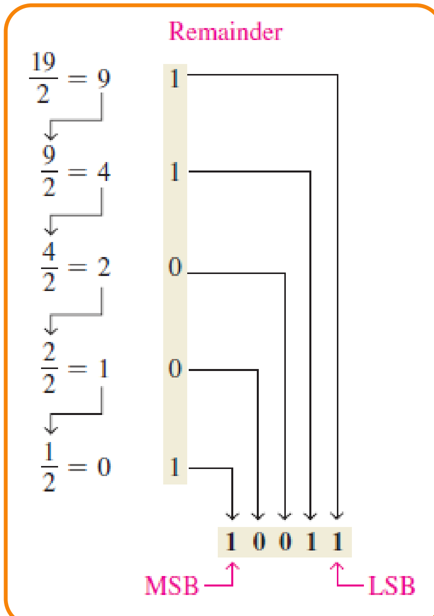
$= 2^{-1} + 2^{-3}$

$= 0.101_2$



What about 19.625 and 0.626?

❑ Repeated Division/Multiplication-by-2 Method



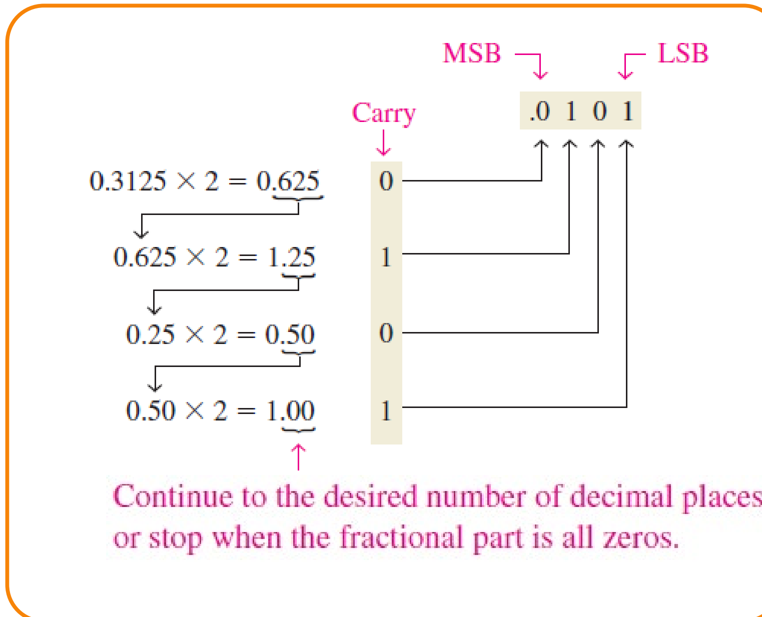


TABLE 2–2

Binary weights. https://www.rapidtables.com/convert/number/decimal-to-binary.html

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 **0.0625**	1/32 0.03125	1/64 0.015625

Binary Arithmetic

- Basic rules

Addition

$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1

Subtraction

$0 - 0 = 0$	
$1 - 1 = 0$	
$1 - 0 = 1$	
$10 - 1 = 1$	$0 - 1$ with a borrow of 1

Multiplication

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Addition

$\boxed{11_2} + \boxed{1_2}$

	Carry	Carry	
	1	1	
	0	1	1
+	0	0	1
	1	0	0

Subtraction

$$\begin{array}{r} 101 \\ -011 \\ \hline \mathbf{010} \end{array}$$

Multiplication

$$\begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline \mathbf{100011} \end{array}$$

Partial products

Division

$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array}$$

Complements of Binary Numbers

- 1's Complements & 2's Complements

By definition

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	*Changing all 1s to 0s and all 0s to 1s,*
0 1 0 0 1 1 0 1	1's complement
+ 1	Add 1
01001110	2's complement

- Alternative method of finding the 2's compliment (**recommended**)
  - Start at the right with the LSB and write the bits as they are up to and including the **first 1**
  - Take the 1's complements of the remaining bits.

1011$\boxed{1}$000	Binary number
$\underbrace{\mathbf{0100}}\underbrace{\mathbf{1000}}$	2's complement

1's complements of original bits

These bits stay the same.

Signed Numbers (I)

- Sign-Magnitude Form
  - Sign bit: 0 → positive; 1 → negative
  - Sign bit and Magnitude bits.

0 0 0 1 1 0 0 1

Sign bit   Magnitude bits

- Representation Forms
  - 1's Complement

Using 8 bits to represent -25

25: 0 0 0 1 1 0 0 1

-25: 1 1 1 0 0 1 1 0

  - 2's Complement

-25: 1 1 1 0 0 1 1 1

Why this method gives the right answer?

$$2^n + 2^{n-1} + \ldots + 2^0 = 2^{n+1} - 1$$

- EXAMPLE 2–16： Determine the decimal value of the signed binary number : 11101000 expressed in 1's complement

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

x

-128 + 64 + 32 + 8 = -24

Adding 1 to the result, -24 + 1 = **-23**

$$\begin{array}{rl} 1\,1\,0\,1\,0\,0\,0 & x \\ +\ 0\,0\,1\,0\,1\,1\,1 & y \\ \hline 2^7-1 & \end{array}$$

y is the 1's complement of x

$x + y = 2^7 - 1$

$-y = -2^7 + x + 1$

Signed Numbers (II)

- EXAMPLE 2–17： Determine the decimal value of the signed binary number : 10101010 expressed in 2's complement

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

$\Rightarrow$ -128 + 32 + 8 + 2 = -86

Compared to the 1's complement, the add-one step is skipped

- Range of Signed Integer Numbers
  - n-bit numbers $[-2^{n-1}, +2^{n-1}-1]$
  - e.g. 8-bit numbers [-128, +127]

    (8 bits = 1 byte)

Brutal approach

$$
\begin{array}{rl}
 & \boxed{0\,1\,0\,1\,0\,1\,0} \quad x \\
- & 1 \\
\hline
 & 0\,1\,0\,1\,0\,0\,1 \\
 & \Downarrow \text{Complement} \\
y & \boxed{1\,0\,1\,0\,1\,1\,0} = 86_{10} \Rightarrow = -86_{10}
\end{array}
$$

Textbook approach

$$
\begin{array}{rll}
 & 0\,1\,0\,1\,0\,1\,0 & x \\
+ & 1\,0\,1\,0\,1\,0\,1 & y' \\
\hline
 & 1\,1\,1\,1\,1\,1\,1 &
\end{array}
$$

Flip all bits

1's Complement

$y = y' + 1 \Rightarrow -y = -y' - 1 = -2^7 + x$

Addition with Signed Numbers (I)

- 3-step procedures:
  1. Convert the decimal numbers into the binary form with negative numbers expressed in the 2's compliment form
  2. Perform binary addition
  3. Sign: If both numbers positive (negative) → the result positive (negative). Otherwise, **overflow** indicates the result is positive.

Both numbers positive

$$\begin{array}{r} 00000111 \\ +\ 00000100 \\ \hline 00001011 \end{array} \qquad \begin{array}{r} 7 \\ +\ 4 \\ \hline 11 \end{array}$$

Both numbers negative

$5_{10} = 00000101_2$

$9_{10} = 00001001_2$

2's complement

$$\begin{array}{r} 11111011 \\ +\ 11110111 \\ \hline 1\ \ 11110010 \end{array} \qquad \begin{array}{r} -5 \\ +\ -9 \\ \hline -14 \end{array}$$

Discard carry

$00001110_2 = 14_{10}$

If the sign is incorrect, then overflow has occurred!

8-bit numbers [-127, +128]

$$\begin{array}{r} 01111101 \\ +\ 00111010 \\ \hline 10110111 \end{array} \qquad \begin{array}{r} 125 \\ +\ 58 \\ \hline 183 \end{array}$$

incorrect

Addition with Signed Numbers (II)

One negative number

$$\begin{array}{r} 00010000 \\ +\ 11101000 \\ \hline 11111000 \end{array} \qquad \begin{array}{r} 16 \\ +\ -24 \\ \hline -8 \end{array}$$

1. $16_{10} = 00010000_2$; $24_{10} = 00011000_2$ → $-24_{10} = 11101000_2$
2. 11111000
3. 2's complement: $00001000_2 = 8_{10}$ → -8

No overflow indicates the result is negative

$6_{10}=00000110_2$

$$\begin{array}{rr} & 00001111 \\ + & 11111010 \\ \hline \text{Carry} \rightarrow 1 & 00001001 \end{array} \qquad \begin{array}{r} 15 \\ +\ -6 \\ \hline 9 \end{array}$$

1. $15_{10}=00001111_2$; $6_{10}=00000110_2$ → $-6_{10}=11111010_2$
2. $+\ 11111010$
3. $00001001_2 = 9_{10}$

Overflow indicates the result is positive

Why Complement + Adding One Works?

Recall "borrowing one's"

Not an easy task for computers

Decimal

$$\begin{array}{r} {}^{1}\ \ \\ 3 \\ -\ \ 8 \\ \hline -5 \end{array}$$

3-8

$3_{10} = 00011_2$

$8_{10} = 01000_2$ → $-8_{10} = 11000_2$

2's complement

$= 11011_2$

2's complement $00101_2 = 5_{10}$

Refer to the NO overflow case on the previous page

Binary

$$\begin{array}{r} {}^{1\ 1\ 1\ 1} \\ 0000 \\ -\ 0011 \\ \hline 1101 \end{array}$$

10-01

Using 0-3 as an example

2's complement of 0000

$$\begin{array}{r} {}^{1\ 1\ 1\ 1} \\ \boxed{1}0000 \\ -\ 0011 \\ \hline 1101 \end{array}$$

$=1111 + 1$

10-01

1 **Inversion** of 0011

2 **Add one**

$$\begin{array}{r} \boxed{+\ 1} \\ 1111 \\ -\ 0011 \\ \hline 1101 \end{array}$$

$\boxed{\begin{array}{r} 1111 \\ -\ 0011 \end{array}} = 1100$

10-01

→ Avoid the borrowing process with 2's complement

Subtraction with Signed Numbers

- Change the sign of the subtrahend and add the numbers
- If the subtrahend is in the binary format → 2's compliment

EXAMPLE 2–20

Perform each of the following subtractions of the signed numbers:

(a) 00001000 – 00000011 (b) 00001100 – 11110111

(c) 11100111 – 00010011 (d) 10001000 – 11100010

Solution

Like in other examples, the equivalent decimal subtractions are given for reference.

(a) In this case, $8 - 3 = 8 + (-3) = 5$.

	00001000	Minuend (+8)
	+ 11111101	2's complement of subtrahend (–3)
Discard carry →	1 00000101	Difference (+5)

(b) In this case, $12 - (-9) = 12 + 9 = 21$.

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

Multiplication with Signed Numbers

- ❑ Direct addition :
  - ◆ Very lengthy if the multiplier is a large number
  - ◆ both numbers must be in true (uncomplemented) form.
- ❑ Partial product
  - ◆ Same sign → Positive;
  - ◆ Different signs → Negative

A Decimal Example

$$
\begin{array}{r}
239 \\
\times\ 123 \\
\hline
717 \\
478 \\
+\ 239 \\
\hline
29{,}397
\end{array}
$$

EXAMPLE 2–22

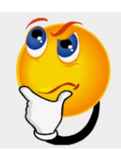
01010011 (multiplicand) and $\boxed{11000101}$ (multiplier).

83 -59

	1010011	Multiplicand
59	× $\boxed{0111011}$	Multiplier
	1010011	1st partial product
	+ 1010011	2nd partial product
	11111001	Sum of 1st and 2nd
	+ 0000000	3rd partial product
	011111001	Sum
	+ 1010011	4th partial product
	1110010001	Sum
	+ 1010011	5th partial product
	100011000001	Sum
	+ 1010011	6th partial product
	1001100100001	Sum
	+ 0000000	7th partial product
4897	01001100100001	Final product
	10110011011111	2's complement
	-4897	

Division with Signed Numbers

- ❑ Division operation in computers is accomplished using subtraction
- ❑ Quotient and Reminder
  - ◆ When to stop the subtraction: the reminder is a zero or a negative number
  - ◆ Quotient: # of times of subtraction performed
- ❑ Sign of the quotient
  - ◆ Same sign → Positive;
  - ◆ Different signs → Negative



What about dividing 100 by -25?

EXAMPLE 2–23 Divide 01100100 by 00011001

100 25

Step 1: 00011001 → 11100111

2's complement

Step 2:

```
  01100100
+ 11100111
----------
  01001011
```

Add 1 to quotient

Step 3:

```
  01001011
+ 11100111
----------
  00110010
```

Add 1 to quotient

Step 4:

```
  00110010
+ 11100111
----------
  00011001
```

Add 1 to quotient

Step 5:

```
  00011001
+ 11100111
----------
  00000000
```

Add 1 to quotient

Quotient=4_{10}=0100

Floating-Point Numbers

- A floating-point number expressed with mantissa and exponent $\boxed{0.2415068} \times 10^{\boxed{9}}$
- Single-precision Floating-point binary numbers

S	Exponent (E)	Mantissa (fraction, F)
1 bit	8 bits	23 bits

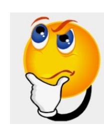
(32 bits)

◆ Biased exponent : adding 127 to the actual exponent

EXAMPLE 2–18： Convert 3.248 x 10^4 to a single-precision floating-point binary number

$3.248 \times 10^4 = 111111011100000_2 = \cancel{1}.\boxed{11111011}_2 \times 2^{14}$

Biased exponent = 14 + 127 = 141 = $\boxed{10001101}_2$



Mantissa =11111011100000000000000 (23 bits)

0	10001101	11111011100000000000000

32480	Reminder
16240	0
8120	0
4060	0
2030	0
1015	0
507	1
253	1
126	1
63	0
31	1
15	1
7	1
3	1
1	1
0	1

LSB

MSB

The range of the biased exponent [-127, +128]

Extremely large or Small numbers can be represented in this way

Hexadecimal Numbers

- A number system of a base of sixteen composed of 10 numeric digits and 6 alphabetic characters
- Binary-to-Hexadecimal Conversion
  - Divide the binary number into 4-bit groups, starting at the right-most bit
  - Replace each 4-bit group with the equivalent hexadecimal symbol

TABLE 2–3

Decimal	Binary	Hexadecimal
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
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

EXAMPLE 2–24

(a) $$\underbrace{1100}_{\downarrow\ \mathrm{C}}\underbrace{1010}_{\downarrow\ \mathrm{A}}\underbrace{0101}_{\downarrow\ 5}\underbrace{0111}_{\downarrow\ 7} = \mathbf{CA57}_{16}$$

(b) $$\underbrace{0011}_{\downarrow\ 3}\underbrace{1111}_{\downarrow\ \mathrm{F}}\underbrace{0001}_{\downarrow\ 1}\underbrace{0110}_{\downarrow\ 6}\underbrace{1001}_{\downarrow\ 9} = \mathbf{3F169}_{16}$$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Hexadecimal-based Conversion

❑ Hexadecimal-to-Binary Conversion

$$\underbrace{1100}_{C}\underbrace{1111}_{F}\underbrace{1000}_{8}\underbrace{1110}_{E}$$

$CF8E_{16}=1100111110001110_2$

TABLE 2-3

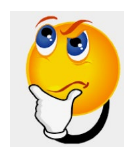
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

❑ Hexadecimal-to-Decimal Conversion

$$E5_{16} = (E \times \overset{16^1}{16}) + (5 \times \overset{16^0}{1})$$

$$= (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$$

❑ Decimal-to-Hexadecimal Conversion



◆ Successive division by 16 with the 1st remainder being the least significant digit (LSD) $650_{10}=28A_{16}$

$$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 = A$$

Why this must be an integer?

$$\frac{40}{16} = 2.5 \rightarrow 0.5 \times 16 = 8 = 8$$

$$\frac{2}{16} = 0.125 \rightarrow 0.125 \times 16 = 2 = 2$$

LSD

MSD

Hexadecimal Operations

10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- Hexadecimal addition

EXAMPLE 2–29

(d)

$$\begin{array}{r} DF_{16} \\ +\ AC_{16} \\ \hline \mathbf{18B_{16}} \end{array}$$

right column: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$

$27_{10} - 16_{10} = 11_{10} = B_{16}$ with a 1 carry

left column: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$

$24_{10} - 16_{10} = 8_{10} = 8_{16}$ with a 1 carry

- Hexadecimal subtraction
  - 2's compliment of the hexadecimal subtrahend before addition

2A → 00101010 → 11010110 → D6

84_{16} - $2A_{16}$

(a) $2A_{16} = 00101010$

2's complement of $2A_{16} = 11010110 = D6_{16}$ (using Method 1)

$$\begin{array}{r} 84_{16} \\ +\ D6_{16} \\ \hline \cancel{1}5A_{16} \end{array}$$

Add

Drop carry, as in 2's complement addition

The difference is $\mathbf{5A_{16}}$.

$C3_{16} - 0B_{16}$

(b) $0B_{16} = 00001011$

2's complement of $0B_{16} = 11110101 = F5_{16}$

$$\begin{array}{r} C3_{16} \\ +\ F5_{16} \\ \hline \cancel{1}B8_{16} \end{array}$$

Add

Drop carry

The difference is $\mathbf{B8_{16}}$.

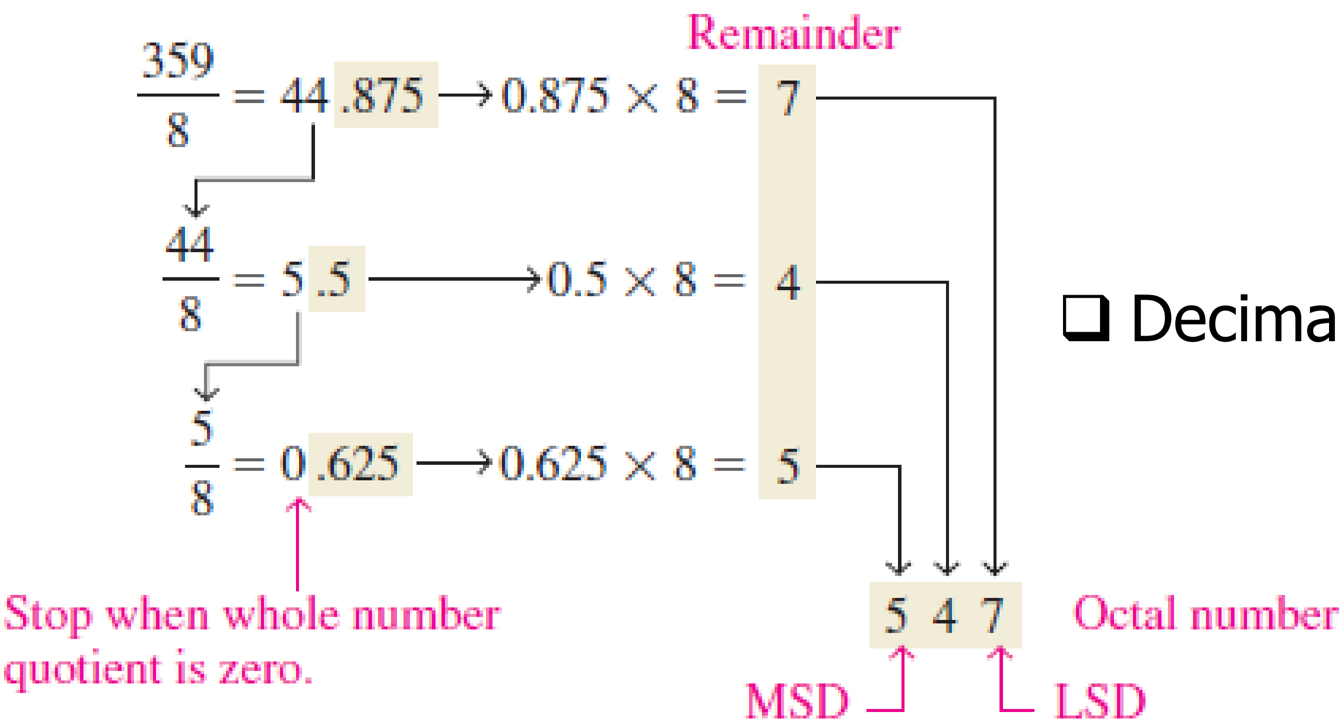
Octal Numbers (I)

- A number system of a base of eight composed of 3 digits
- Octal-to-Decimal Conversion

Weight: $8^3\ 8^2\ 8^1\ 8^0$

Octal number: $2\ \ 3\ \ 7\ \ 4$

$$\begin{aligned} 2374_8 &= (2\times 8^3) + (3\times 8^2) + (7\times 8^1) + (4\times 8^0) \\ &= (2\times 512) + (3\times 64) + (7\times 8) + (4\times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$



- Decimal-to-Octal Conversion

Octal Numbers (II)

- Octal-to-Binary Conversion
  - Replace each octal digit with three bits

TABLE 2–4

Octal/binary conversion.

Octal Digit	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

EXAMPLE 2–31

Convert each of the following octal numbers to binary:

(a) 13_8 (b) 25_8 (c) 140_8 (d) 7526_8

Solution

(a) $\underbrace{001}_{1}\underbrace{011}_{3}$ = 001011

(b) $\underbrace{010}_{2}\underbrace{101}_{5}$ = 010101

(c) $\underbrace{001}_{1}\underbrace{100}_{4}\underbrace{000}_{0}$ = 001100000

(d) $\underbrace{111}_{7}\underbrace{101}_{5}\underbrace{010}_{2}\underbrace{110}_{6}$ = 111101010110

- Binary-to-Octal Conversion
  - Convert each 3-bit group to the equivalent octal digit

Binary Coded Decimal (BCD)

- Express each of the decimal digits with a binary code.
- The 8421 code

TABLE 2–5

Decimal/BCD conversion.

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

EXAMPLE 2–33

170_{10}

(c) $\overbrace{0001}^{1}\overbrace{0111}^{7}\overbrace{0000}^{0}$

2469_{10}

(d) $\overbrace{0010}^{2}\overbrace{0100}^{4}\overbrace{0110}^{6}\overbrace{1001}^{9}$

- BCD-to-Decimal Conversion
  - Starting from the right-most bit and divide the code into groups of four bits

BCD Addition

1. Add the two BCD numbers, using the rules for binary addition
2. If a 4-bit sum ≤ 9 → Valid BCD number
3. Otherwise, add 6 (0110) to the 4-bit sum

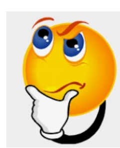
TABLE 2–5

Decimal/BCD conversion.

Decimal Digit	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

EXAMPLE 2–36

$$\begin{array}{rrll}
 & 1001 & & 9 \\
 & +\ 0100 & & +4 \\
\hline
 & \boxed{1101} & \rightarrow \text{Invalid BCD number } (>9) & 13 \\
 & +\ 0110 & \text{Add 6} & \\
\hline
\underbrace{0001}_{\downarrow\ 1} & \underbrace{0011}_{\downarrow\ 3} & \text{Valid BCD number} &
\end{array}$$



Why adding 6? Think about what 1111 stands for.

Digital Codes: The Gray Code (I)

- 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

Digital Codes: The Gray Code (II)

EXAMPLE 2–37

(a) Convert the binary number 11000110 to Gray code.

(b) Convert the Gray code 10101111 to binary.

Solution

(a) Binary to Gray code:

Start from the MSB

1 — + → 1 — + → 0 — + → 0 — + → 0 — + → 1 — + → 1 — + → 0

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

1 0 1 0 0 1 0 1

(b) Gray code to binary:

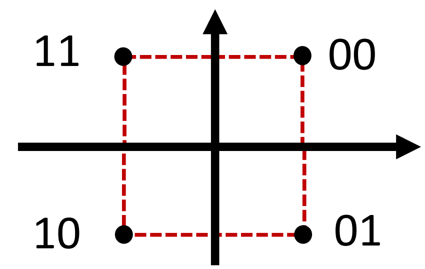
Start from the MSB

1 0 1 0 1 1 1 1

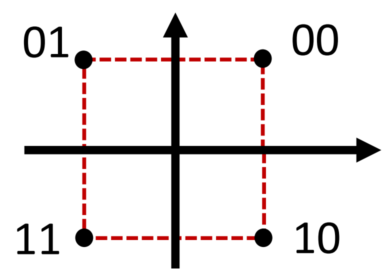
↓ + ↗ ↓ + ↗ ↓ + ↗ ↓ + ↗ ↓ + ↗ ↓ + ↗ ↓ + ↗ ↓

1 1 0 0 1 0 1 0

Without gray code

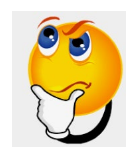


With gray code



Encoding transmit symbols with the Gray Code

- Transmitter sends one of the four symbols;
- Transmitted symbols are distorted by noise;
- The receiver tries to decode the distorted symbols;
- The probability of making a symbol detection error is ***inversely*** proportional to the symbol distance



Why Gray Code has an advantage?

Digital Codes: Alphanumeric Codes

❑ ASCII : American Standard Code for Information Interchange

TABLE 2–7

American Standard Code for Information Interchange (ASCII).

Control Characters				Graphic Symbols											
Name	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**	**Symbol**	**Dec**	**Binary**	**Hex**
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	−	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

Error Codes :Parity Bit for Error Detection

- A parity bit is attached to a group of bits to make the total number of 1's in a group always even or always odd.

TABLE 2–8

The BCD code with parity bits.

Even Parity *P*	Even Parity BCD	Odd Parity *P*	Odd Parity BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

EXAMPLE 2–40

An odd parity system receives the following code groups: 10110, 11010, 110011, 110101110100, and 1100010101010. Determine which groups, if any, are in error.

	# of 1's
10110	3
11010	3
110011	4
110101110100	7
1100010101010	6

Since odd parity is required, any group with an even number of 1s is incorrect. The following groups are in error: 110011 and 1100010101010.

Error Codes :Cyclic Redundancy Check

- Cyclic Redundancy Check (CRC) : an error detection method that can detect multiple errors in data blocks
- At the sending end, a checksum is appended to a block of data.
- At the receiving end, the check sum is generated and compared to the sent checksum. If the check sums are the same, no error is detected.

EXAMPLE 2–41

D: 11010011 Data

G: 1010 Generator code

Step 1: Append 0000 to the data

D' = 110100110000

Step 2: $\dfrac{D'}{G} = \dfrac{110100110000}{1010}$

```
110100110000
1010
 1110
 1010
  1000
  1010
   1011
   1010
      1000
      1010
       100
```

Modulo-2 operation (XOR)

One-bit error at the receiving end

```
100100110100
1010
  1100
  1010
   1101
   1010
    1111
    1010
     1010
     1010
        0100
```

Non-zero remainder
Error detected!

Remainder (Checksum) : 0100

Step 3: The transmitted CRC is 110100110100

Chapter Review

- Binary, Decimal, Hexadecimal and Octal numbers
- Binary/Hexadecimal/Octal/Decimal Conversion
  - Binary/Hexadecimal/Octal-to-Decimal Conversion
  - Decimal-to-Binary/Hexadecimal/Octal Conversion: Repeated Division/Multiplication-by-2/16/8 (LSD→ MSD)
- Most Significant Digit (MSD) and Least Significant Digit (LSD)
- Floating-Point numbers: Mantissa and (biased) Exponent
- Binary Arithmetic
  - Addition, Subtraction, Multiplication and Division
  - Unsigned versus Signed numbers
- Binary coded decimal (BCD)
- Digital codes: Gray/Alphanumeric/Error Codes

True/False Quiz

- The octal number system is a weighted system with eight digits.
- The binary number system is a weighted system with two digits.
- MSB stands for most significant bit.
- In hexadecimal, 9 + 1 = 10.
- The 1's complement of the binary number 1010 is 0101.
- The 2's complement of the binary number 1111 is 0000.
- The right-most bit in a signed binary number is the sign bit.
- The hexadecimal number system has 16 characters, six of which are alphabetic characters.
- BCD stands for binary coded decimal.
- An error in a given code can be detected by verifying the parity bit.
- CRC stands for cyclic redundancy check.
- The modulo-2 sum of 11 and 10 is 100.

True/False Quiz

- ✓ The octal number system is a weighted system with eight digits.
- ✓ The binary number system is a weighted system with two digits.
- ✓ MSB stands for most significant bit.
- ✗ In hexadecimal, 9 + 1 = 10.
- ✓ The 1's complement of the binary number 1010 is 0101.
- ✗ The 2's complement of the binary number 1111 is 0000.
- ✗ The right-most bit in a signed binary number is the sign bit.
- ✓ The hexadecimal number system has 16 characters, six of which are alphabetic characters.
- ✓ BCD stands for binary coded decimal.
- ✓ An error in a given code can be detected by verifying the parity bit.
- ✓ CRC stands for cyclic redundancy check.
- ✗ The modulo-2 sum of 11 and 10 is 100.