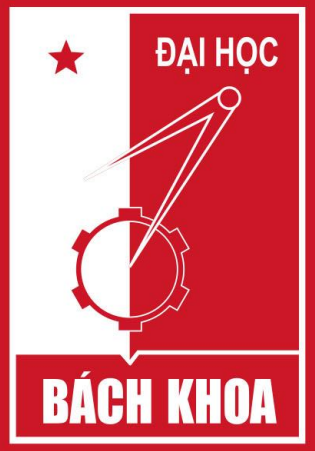


ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Electronics for Information Technology

(Điện tử cho Công nghệ Thông tin)

IT3420E

Đỗ Công Thuần

Department of Computer Engineering

Email: thuandc@soict.hust.edu.vn

General Information

- Course: **Electronics for Information Technology**
- ID Number: IT3420
- Credits: 2 (2-1-0-4)
- Lecture/Exercise: 32/16 hours (48 hours, 16 weeks)
- Evaluation:
  - Midterm examination and weekly assignment: **50%**
  - Final examination: **50%**
- Learning Materials:
  - Lecture slides
  - Textbooks
    - ***Introductory Circuit Analysis*** (2015), 10th – 13th ed., Robert L. Boylestad
    - ***Electronic Device and Circuit Theory*** (2013), 11th ed., Robert L. Boylestad, Louis Nashelsky
    - ***Microelectronics Circuit Analysis and Design*** (2006), 4th ed., Donald A. Neamen
    - ***Digital Electronics: Principles, Devices and Applications*** (2007), Anil K. Maini

Contact Your Instructor

- You can reach me through office in **Room 802, B1 Building**, HUST.
  - You should make an appointment by email before coming.
  - If you have urgent things, just come and meet me!
- You can also reach me at the following **email** any time. This is the best way to reach me!
  - thuandc@soict.hust.edu.vn

Course Contents

- The Concepts of Electronics for IT
- **Chapter 1**: Passive Electronic Components and Applications
- **Chapter 2**: Semiconductor Components and Applications
- **Chapter 3**: Operational Amplifiers
- **Chapter 4**: Fundamentals of Digital Circuits
- **Chapter 5**: Logic Gates
- **Chapter 6**: Combinational Logic
- **Chapter 7**: Sequential Logic

Chapter 4:
Fundamentals of Digital Circuits

1. Introduction to Number Systems
2. Number Systems
3. Digital Arithmetic

References:

Digital electronics: Principles, Devices, and Applications, Anil Kumar Maini 2007 John Wiley & Sons

Fundamentals of Logic Design, Seventh Edition, Charles H. Roth, Jr. and Larry L. Kinney

Digital Fundamentals, Thomas L. Floyd, Eleventh Edition, Pearson Education Limited 2015

Chapter 4:
Fundamentals of Digital Circuits

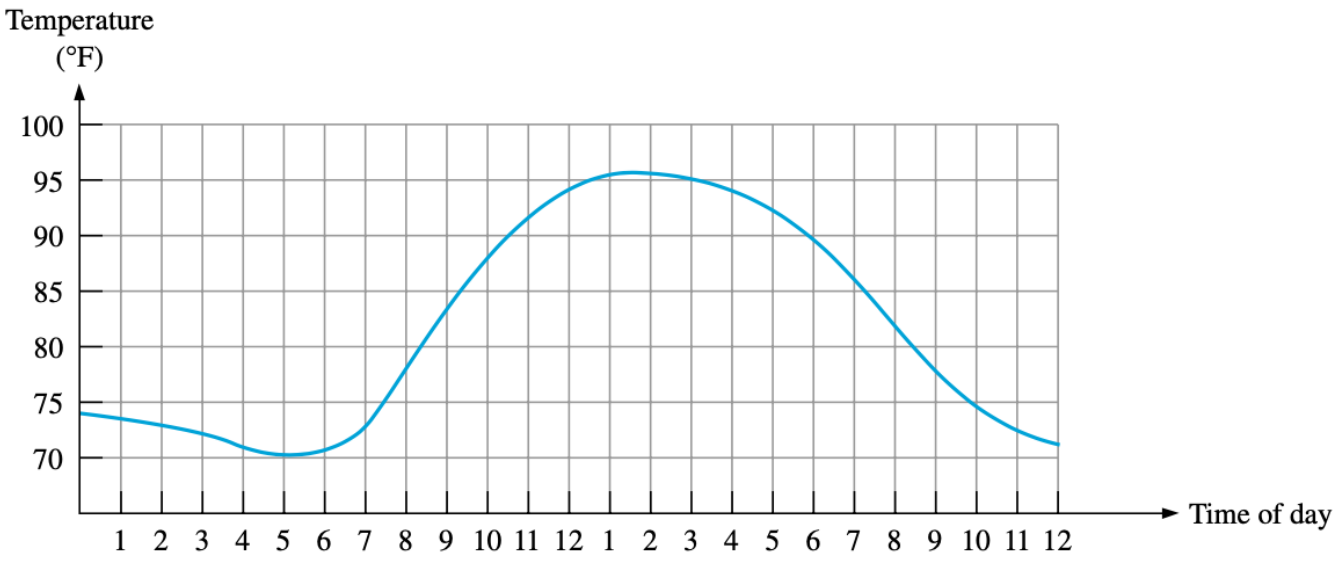
1. Introduction to Number Systems
2. Number Systems
3. Digital Arithmetic

Introduction to Number Systems

- Electronic circuits can be divided into 2 broad categories: **digital** and **analog**.
- Digital electronics involves quantities with descrete values.
- Analog electronics involves quantities with continuous values.

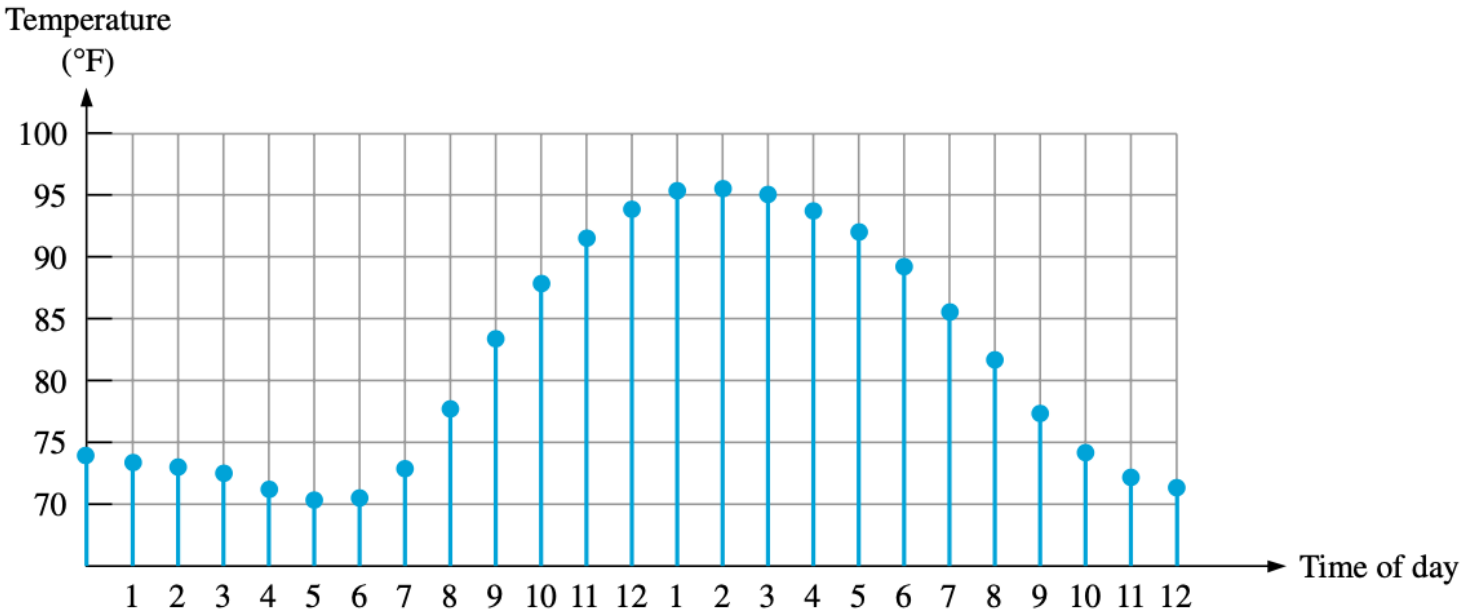
Analog Quantity

- An analog quantity is one having continuous value.
- Example: the air temperature changes over a continuous range of values.
  - Infinite values between 70º - 71º



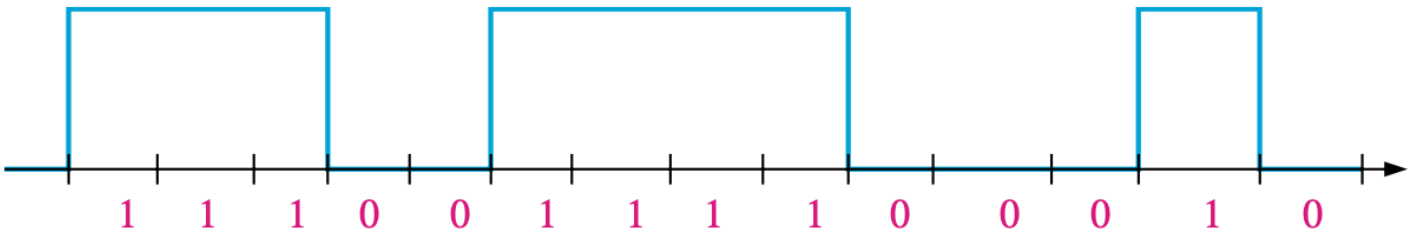
Digital Quantity

- A digital quantity is one having a discrete set of values.
- Example: taking a temperature reading every hour



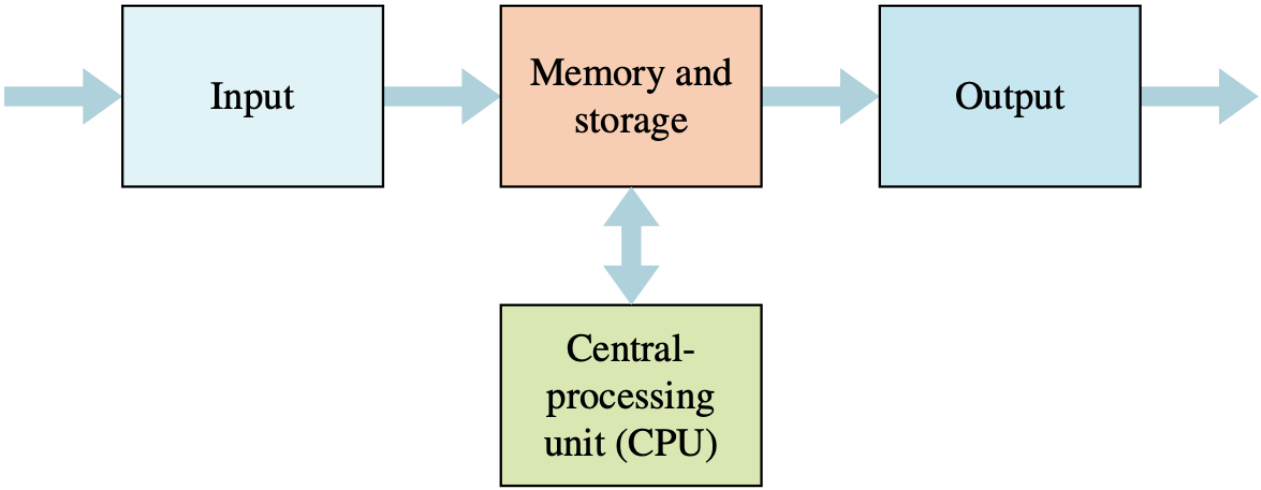
Digital Signal and Modulation

- A digital signal is encoded into sequences of 0's and 1’s.
- The sequencs is then transmitted as a sequence of voltage pulses of equal width that represent bit ‘1’ and bit ‘0’.



Digital System

- A digital system contains devices that process the physical quantities represented in digital form.
- Example: personal computer (PC)



The Digital Advantage

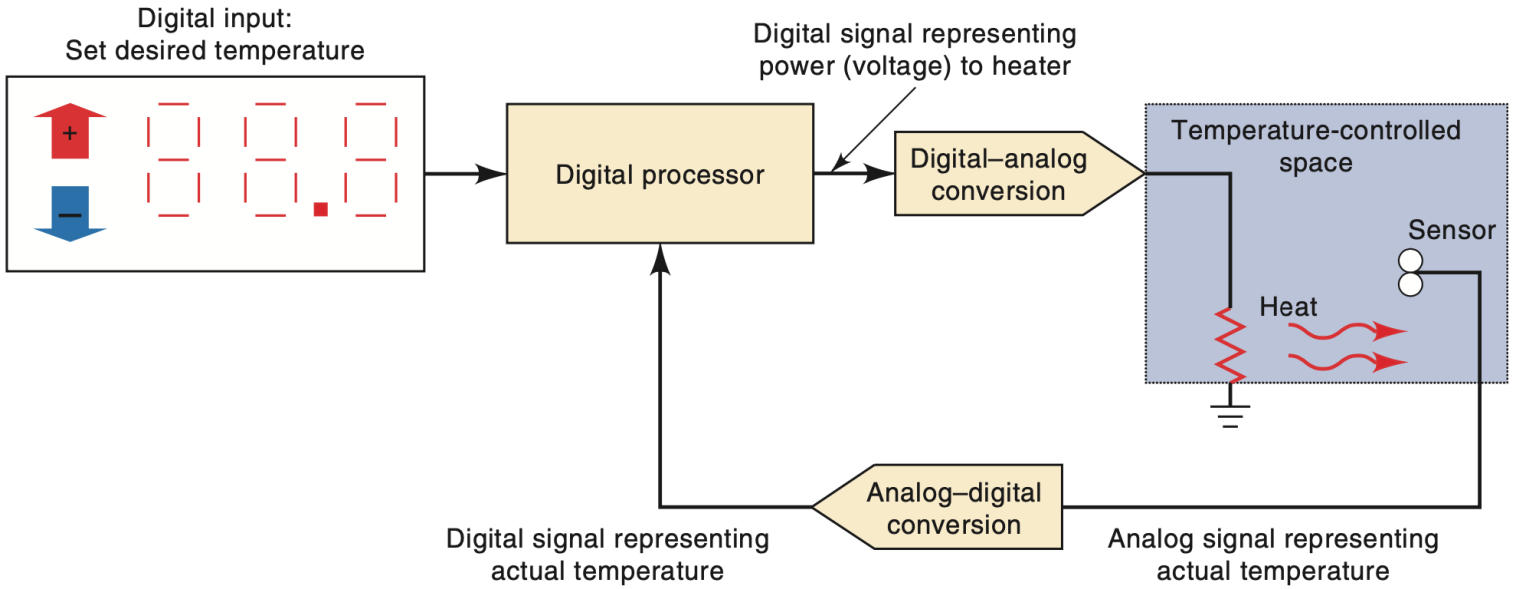
- Digital data can be processed and transmitted more efficiently and reliably than analog data.
- Digtal data can be stored more compactly and preprocued with great accuracy and clarity.
- Noise does not affect digital data as much as analog data.
- Digital systems are more programmable and much easier to design.

The Digital Disadvantage

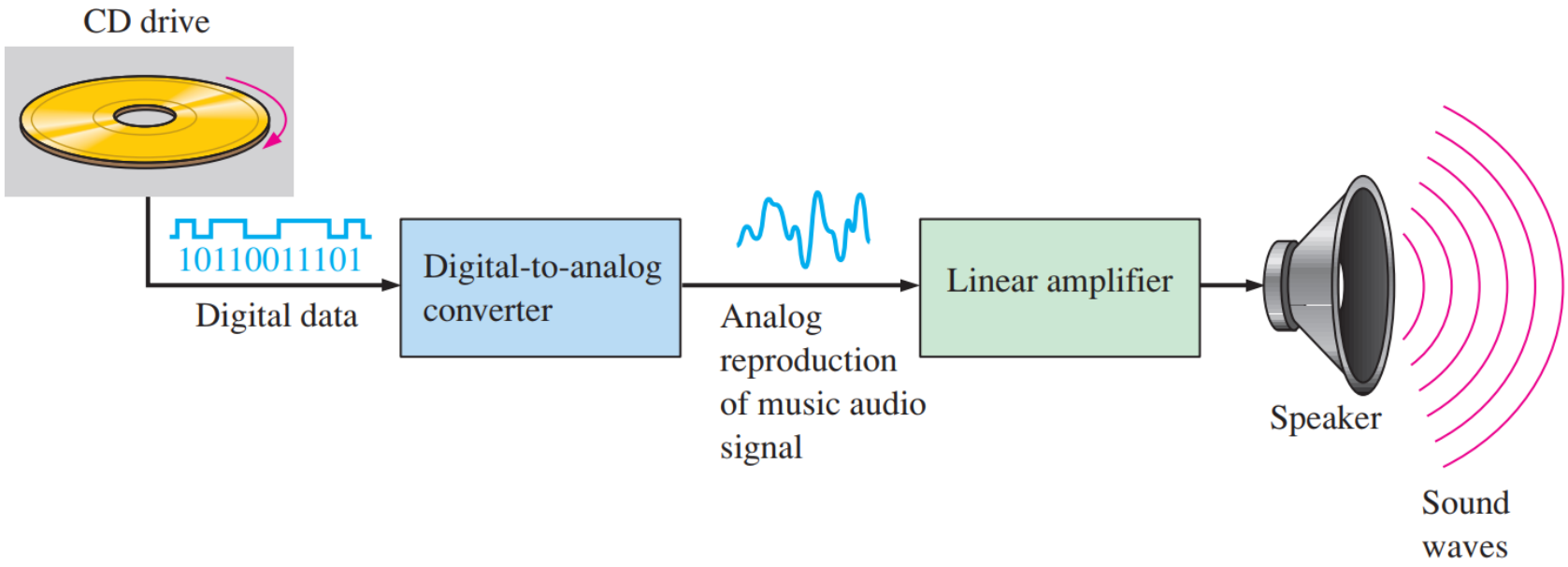
- Most physical quantities (e.g., position, velocity, acceleration, force, pressure, temperature and flowrate, …) are analogue in nature.
- A/D conversion takes time and causes error!
  - Physical quantities (variables) to analog signal (voltage, current) conversion
  - Analog-to-Digital conversion
  - Digital processing
  - Digital-to-Analog conversion

Combining Digital and Analog Methods

- Most of real applications use both digital and analog systems.

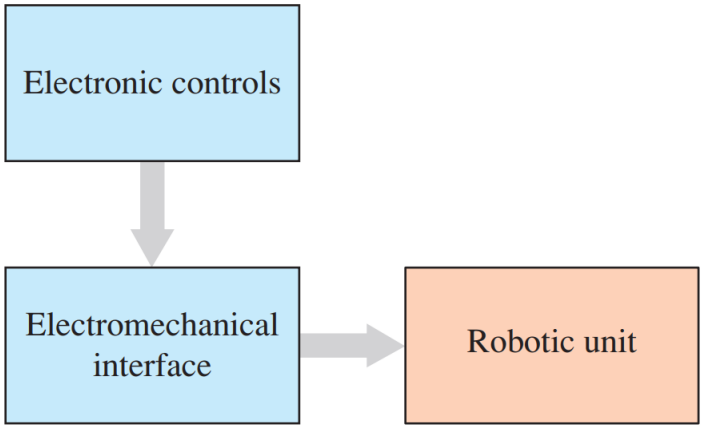


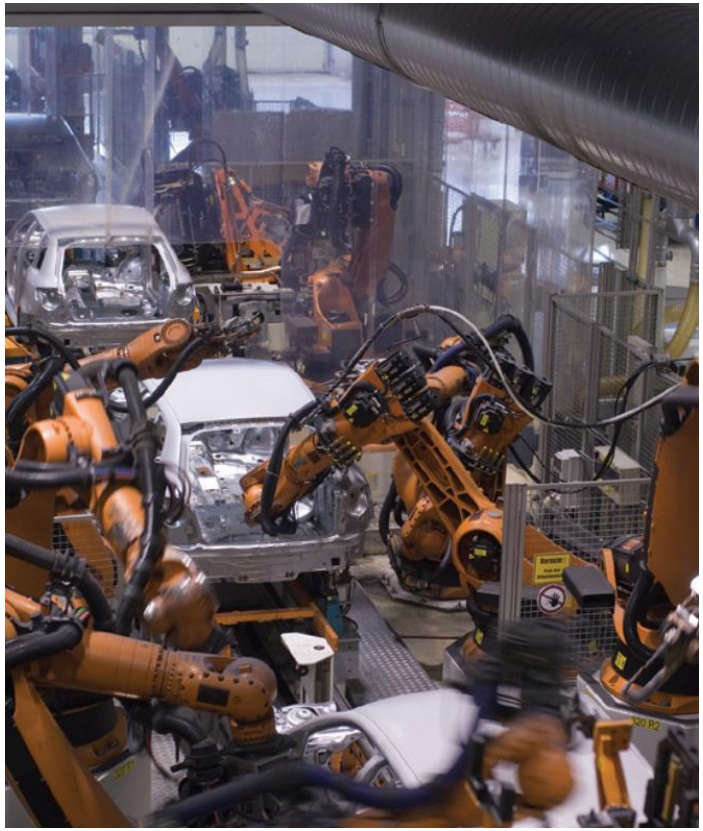
Combining Digital and Analog Methods



Basic block diagram of a CD player

Combining Digital and Analog Methods





A mechatronic system: Robotic arm

Chapter 4:
Fundamentals of Digital Circuits

1. Introduction to Number Systems
2. Number Systems
3. Digital Arithmetic

Number System

- A number system includes the number of independent digits (or symbols) used in the number system, the place values of the different digits constituting the number, and the maximum numbers that can be written with the given number of digits .
  - Symbol in Roman numerals: I, V, X, L, C,..
  - Standard form: IX, XV, XXX
- Tthe number of independent digits used in the number system is called radix or base (denoted as **r**).
  - **Example:** the decimal system has a radix of 10 (0,1,2…,9).

Number System

- Base-***r*** Number System
- Common Number Systems
- Representation of Negative Numbers
- Conversion between Two Number Systems
- Floating-Point Numbers

Base-***r*** Number System

- A radix of ***r*** digits, 0 → (***r***-1)
- A number, N, in the base-***r*** number system is represented in the form: $a_5\, a_4\, a_3\, a_2\, a_1\, a_0\, .\, a_{-1}\, a_{-2}\, a_{-3}$
- In general, N is written as:

$$N = a_{n-1} \times r^{n-1} + \cdots + a_0 \times r^0 + a_{-1} \times r^{-1} + \cdots + a_{-m} \times r^{-m}$$

$$= \sum_{n-1}^{-m} a_i \times r^i$$

- Weights of whole numbers: r^0, r^1, r^2…
- Weights of fractional numbers: r^{-1}, r^{-2}, r^{-3}…

Common Number Systems

- Decimal System

  → For human beings

- Binary System

  → For machine (e.g. PC)

- Octal System

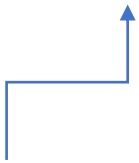
  → A convenient way to abbreviate binary numbers

- Hexadecimal System

  → A convenient way to abbreviate binary numbers

Decimal System (base 10)

- A radix of 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Weights of whole numbers: 10^0, 10^1, 10^2, 10^3…
- Weights of fractional numbers: 10^{-1}, 10^{-2}, 10^{-3}…
- Standard form: $(3456.265)_{10}$



Decimal point

Binary System (base 2)

- A radix of 2 digits: 0, 1
- Weights of whole numbers: 2^0, 2^1, 2^2, 2^3…
- Weights of fractional numbers: 2^{-1}, 2^{-2}, 2^{-3}…
- Standard form: $(0011.0111)_2$

Octal System (base 8)

- A radix of 8 digits: 0, 1, 2, 3, 4, 5, 6, 7
- Weights of whole numbers: 8^0, 8^1, 8^2, 8^3…
- Weights of fractional numbers: 8^{-1}, 8^{-2}, 8^{-3}…
- Standard form: $(123)_8$

Hexadecimal System (base 16)

- A radix of 16 digits: 0→9, A, B, C, D, E, F
- Weights of whole numbers: 16^0, 16^1, 16^2, 16^3…
- Weights of fractional numbers: 16^{-1}, 16^{-2}, 16^{-3}…
- Standard form: $(2ABE)_{16}$

Basic Concepts – Binary System

- Bit: 0 and 1, the smallest unit of information.
- Byte: a string of 8 bits, the basic unit of data operated as a single unit in computers.
- Word: a string of bits whose size is fixed for a specified computer (1 byte, 2 bytes, 3 bytes, 4 bytes or even larger).
- 1's complement of a binary number is obtained by complementing all its bits.
  - E.g.: 1's complement of $(10010110)_2$ is $(01101010)_2$
- 2's complement of a binary number is obtained by adding '1' to its 1's complement.
  - E.g.: 2's complement of $(10010110)_2$ is $(01101011)_2$

Basic Concepts – Decimal System

- 9's complement of a given decimal number is obtained by subtracting each digit from 9.
  - E.g.: 9's complement of $(2496)_{10}$ is $(7503)_{10}$

- 10's complement of a given decimal number is obtained by adding '1' to its 9's complement.
  - E.g.: 10's complement of $(2496)_{10}$ is $(7504)_{10}$

Basic Concepts – Octal System

- 7's complement of a given octal number is obtained by subtracting each digit from 7.
  - E.g.: 7's complement of $(562)_8$ is $(215)_8$

- 8's complement of a given decimal number is obtained by adding '1' to its 7's complement.
  - E.g.: 8's complement of $(562)_8$ is $(216)_8$

Basic Concepts – Hexadecimal System

- 15's complement of a given octal number is obtained by subtracting each digit from 15.
  - E.g.: 15's complement of $(3BF)_{16}$ is $(C40)_{16}$

- 16's complement of a given decimal number is obtained by adding '1' to its 15's complement.
  - E.g.: 16's complement of $(2AE)_{16}$ is $(D52)_{16}$

Example 4.1

- The 7's complement of an octal number is $(5264)_8$. Determine the binary and hexa equivalent of the octal number.
  - The 7's complement: $(5264)_8$
  - The octal number = $(2513)_8$
  - The binary number = $(010\ 101\ 001\ 011)_2$
    = $(10101001011)_2$
  - The hexa decimal number = $(10101001011)_2$
    = $(101\ 0100\ 1011)_{16}$
    = $(54B)_{16}$

Conversion btw. Two Number Systems

The integer and fractional parts of the given number should be treated separatedly.

1. To find the decimal equivalent of a number in a base ***r***:
   - Represent the integer part in the form: r^0, r^1, r^2, r^3…
   - Represent the fractional part in the form: r^{-1}, r^{-2}, r^{-3}…

Binary-to-Decimal Conversion

- Example: $(1101.0101)_2 = (13.3125)_{10}$
- The whole part: $(1101)_2$
  - The decimal equivalent :

    $= 1x2^0 + 0x2^1 + 1x2^2 + 1x2^3$

    $= 1 + 0 + 4 + 8 = 13$
- The fractional part: $(.0101)_2$
  - The decimal equivalent:

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

  $= 0 + 0.25 + 0 + 0.0625 = 0.3125$

Octal-to-Decimal Conversion

- Example: $(135.21)_8 = (93.265625)_{10}$
- The whole part: $(135)_8$
  - The decimal equivalent:

    $= 5x8^0 + 3x8^1 + 1x8^2$

    $= 5 + 24 + 64 = 93$
- The fractional part: $(.21)_8$
  - The decimal equivalent:

    $= 2x8^{-1} + 1x8^{-2}$

    $= 0.25 + 0.015625 = 0.265625$

Hexadecimal-to-Decimal Conversion

- Example: $(1BF.A1)_{16} = (447.\ 62890625)_{10}$
- The whole part: $(1BF)_{16}$
  - The decimal equivalent:

    $= 15x16^0 + 11x16^1 + 1x16^2$

    $= 15 + 176 + 256 = 447$
- The fractional part: $(.A1)_{16}$
  - The decimal equivalent:

    $= 10x16^{-1} + 1x16^{-2}$

    $= 0.625 + 0.00390625 = 0.62890625$

Conversion btw. Two Number Systems

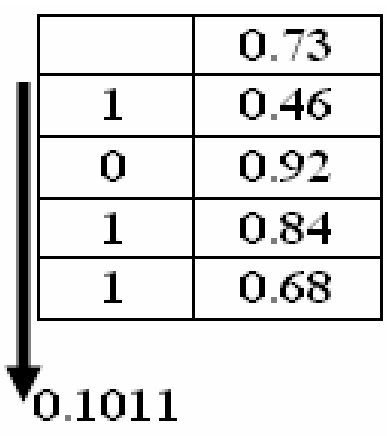
The integer and fractional parts of the given number should be treated separatedly.

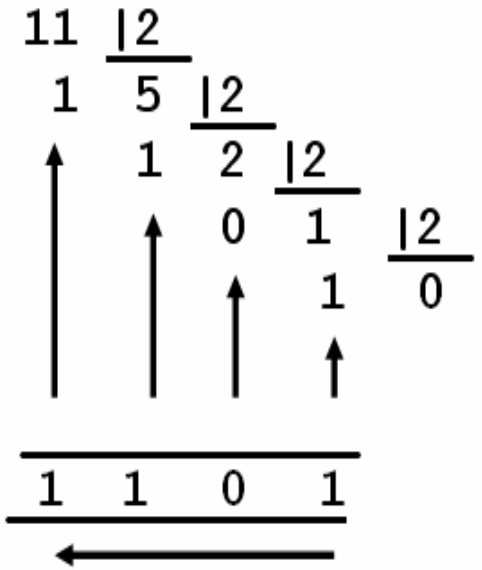
1. To find the decimal equivalent of a number in a base ***r***:
   - Represent the integer part in the form: r^0, r^1, r^2, r^3…
   - Represent the fractional part in the form: r^{-1}, r^{-2}, r^{-3}…
2. To convert a decimal number to its equivalent in a base ***r***:
   - For the integer part, successively divide the integer part by ***r*** → record the remainders until the quotient becomes '0'. The remainders written in reverse order constitute the equivalent (in a base ***r***).
   - For the fractional part, successively multiply the fractional part by ***r*** → record the carry until the result of multiplication is '0'. The carry sequence written in forward order constitutes the equivalent (in a base ***r***).

Decimal-to-Binary Conversion

- Example: $(11.73)_{10} = (1011.1011)_2$

- The whole part: $(11)_{10}$
- The fractional part: $(0.73)_{10}$





Decimal-to-Octal Conversion

- Example: $(23.75)_{10} = (27.6)_8$

- The whole part: $(23)_{10}$
  - The octal equivalent: $(27)_8$
- The fractional part: $(0.75)_{10}$
  - The octal equivalent: $(0.6)_8$

Decimal-to-Hexadecimal Conversion

- Example: $(23.75)_{10} = (17.12)_{16}$

- The whole part: $(23)_{10}$
  - The hexadecimal equivalent: $(17)_{16}$
- The fractional part: $(0.75)_{10}$
  - The hexadecimal equivalent: $(0.12)_{16}$

Conversion btw. Two Number Systems

The integer and fractional parts of the given number should be treated separatedly.

1. To find the decimal equivalent of a number in a base ***r***:
   - Represent the integer part in the form: r^0, r^1, r^2, r^3…
   - Represent the fractional part in the form: r^{-1}, r^{-2}, r^{-3}…
2. To convert a decimal number to its equivalent in a base ***r***:
   - For the integer part, successively divide the integer part by ***r*** → record the remainders until the quotient becomes '0'. The remainders written in reverse order constitute the equivalent (in a base ***r***).
   - For the fractional part, successively multiply the fractional part by ***r*** → record the carry until the result of multiplication is '0'. The carry sequence written in forward order constitutes the equivalent (in a base ***r***).
3. For octal–binary/binary–octal conversion:
   - Replace each octal digit with its 3-bit binary equivalent.
   - Split the integer and fractional parts into groups of 3 bits, starting from the binary point on both sides. The 0s can be added to complete the outside groups if needed.
4. For hexadecimal–octal/octal–hexadecimal conversion:
   - The hex number → its binary equivalent → its octal equivalent. (*An alternative approach: the given hex number → its decimal equivalent → its octal equivalent.*)
   - The octal number → its binary equivalent → its hex equivalent.

Binary-to-Octal/Octal-to-Binary Conversion

- Example: $(375.2)_8 = (?)_2$

  $(1100110001.0111)_2 = (?)_8$

- Octal-to-Binary conversion: $(375.2)_8 = (?)_2$

  $(375.2)_8 = (011\ 111\ 101.010)_2$

  $= (011111101.010)_2$

  $= (11111101.01)_2$

- Binary-to-Octal conversion:

  $(1100110001.0111)_2 = (1\ 100\ 110\ 001.011\ 1)_2$

  $= (001\ 100\ 110\ 001.011\ 100)_2$

  $= (1461.34)_8$

Hex-to-Octal/Octal-to-Hex Conversion

- Example: $(375.2)_8 = (?)_{16}$

  $(3BF.1A)_{16} = (?)_8$

- Octal-to-Hex conversion: $(375.2)_8 = (?)_{16}$

  $(375.2)_8 = (011\ 111\ 101.010)_2$

  $= (011111101.010)_2$

  $= (1111\ 1101.0100)_2 = (FD.4)_{16}$

- Hex-to-Octal conversion: $(3BF.1A)_{16} = (?)_8$

  $(3BF.1A)_{16} = (0011\ 1011\ 1111.0001\ 1010)_2$

  $= (1110111111.0001101)_2$

  $= (001\ 110\ 111\ 111.000\ 110\ 100)_2$

  $= (1677.064)_8$

Representation of Negative Numbers

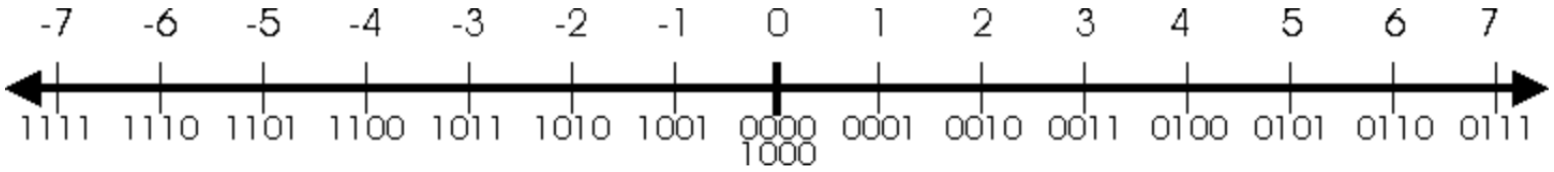
- Several most common methods:
  - Sign and magnitude
  - 2's complement
  - 1's complement
- In each of these methods, the leftmost bit of a number is 0 for positive numbers and 1 for negative numbers.

Sign and Magnitude

- In an n-bit *sign and magnitude* system, a number is represented by a sign bit (0 for '−' and 1 for '+'), followed by n-1 bits that represent the magnitude of the number.
- Example: in an 8-bit sign & magnitude number system
  - +9 is represented by $(00001001)_2$
  - −9 is represented by $(10001001)_2$
- Range of representation: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$, including a positive and negative zero (+0 & −0)
  - Example: in an 8-bit S&M number system, the range is from -127_{10} to $+127_{10}$, including 00000000 (+0) and 10000000 (−0)

Sign and Magnitude

- **Advantage**: easily determine whether a number is negative or positive by testing the MSB
- **Disadvantage** :
  - 2 zero representations (-0 and +0)



  - Binary addition and subtractions are difficult!

1's Complement

- A negative number (-N) is represented by the 1's complement of its positive number (N).
- Example: in an 8-bit 1's complement number system
  - +9 is represented by $(00001001)_2$
  - −9 is represented by $(11110110)_2$
- Range of representation: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$, including a positive and negative zero (+0 & −0)
  - Example: in an 8-bit 1's complement number system, the range is from −127 to +127

1's Complement

- **Advantage**:
  - Easily determining whether a number is negative or positive by testing the MSB
  - Quite convenient for the computer to perform arithmetic
- **Disadvantage**:
  - 2 zero representations (−0 and +0)
  - Binary addition and subtractions are difficult!

```
 −3      1100
 −4      1011
     (1) 0111
         └──→1   (end-around carry)
         ----
         1000    (correct answer, no overflow)
```

1's Complement

- **Advantage**:
  - Easily determining whether a number is negative or positive by testing the MSB
  - Quite convenient for the computer to perform arithmetic
- **Disadvantage**:
  - 2 zero representations (−0 and +0)
  - Binary addition and subtractions are difficult!

```
−5        1010
−6        1001
      (1) 0111
          └──→1    (end-around carry)
          0100     (wrong answer because of overflow)
```

2's Complement

- A negative number (-N) is represented by the 2's complement of its positive number (N):
  - The MSB is the sign bit (0 = '+' & 1 = '–')
  - The following bits are used to represent the magnitude.
- Example: in an 8-bit 2's complement number system
  - +9 is represented by $(00001001)_2$
  - –9 is represented by $(11110111)_2$
- Range of representation: $-(2^{n-1})$ to $+(2^{n-1}-1)$, including only one zero representation.
  - Example: in an 8-bit 2's complement number system, the range is from −128 đến +127

2's Complement

- **Advantage**:
  - Only one representation of zero, 000…0 (all zeroes)
- **Disadvantage**:
  - Not work for multiplication and division
- Binary subtraction:
  - Converting subtraction to addition of 2's complement numbers
  - Discarding the last carry

$$\begin{array}{rll} 11111000 & (-8) \\ \underline{00010011} & \underline{+19} \\ (\cancel{1})00001011 & = +11 \\ \uparrow\!\!\_\_ \text{(discard last carry)} & \end{array}$$

Floating-Point Numbers

- Floating-point notation can be used conveniently to represent both large as well as small fractional or mixed numbers.
- Floating-point numbers are in general expressed in the form: $N = M \times b^E$

  M: **M**antissa (bits)

  b: **b**ase (bits)

  E: **E**xponent (bits)

- Decimal system: $N = M \times 10^E$
- Hexadecimal system: $N = M \times 16^E$
- Binary system: $N = M \times 2^E$

Chapter 4:
Fundamentals of Digital Circuits

1. Introduction to Number Systems
2. Number Systems
3. Digital Arithmetic

Digital Arithmetic

- Basic Rules of Binary Addition and Subtraction
- Addition of Larger-Bit Binary Numbers
- Subtraction of Larger-Bit Binary Numbers
- Binary Multiplication
- Binary Division
- Floating-Point Arithmetic

Basic Rules of Binary Addition

1. 0 + 0 = 0
2. 0 + 1 = 1
3. 1 + 0 = 1
4. 1 + 1 = 0 (with a carry of ‘1’ to the next more significant bit)
5. 1 + 1 + 1 = 1 (with a carry of ‘1’)

Addition of Larger-Bit Binary Numbers

$$A + B + C_{in}$$

A	B	Carry-in (C_{in})	Sum	Carry-out (C_o)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Addition Using the 2's Complement Method

1. Both the numbers are positive
2. The larger of the two numbers is positive
3. The larger of the two numbers is negative
4. Both numbers are negative

Addition Using the 2's Complement Method

- **Case 1**: Consider $+37_{10}$ and $+18_{10}$
  - 2's complement representation of +37 in 8 bits = 00100101
  - 2's complement representation of +18 = 00010010
  - Result = $(00110111)_2 = (+55)_{10}$

- **Case 2**: Consider $+37_{10}$ and -18_{10}
  - 2's complement representation of + 37 = 00100101
  - 2's complement representation of -8 = 11101110
  - Result = $(00010011)_2 = (+19)_{10}$

Addition Using the 2's Complement Method

- **Case 3**: Consider $+18_{10}$ and $\mathbf{-37_{10}}$
  - 2's complement representation of -37 = 11011011
  - 2's complement representation of +18 = 00010010
  - Result = $(11101101)_2 = (-19)_{10}$

- **Case 4**: Consider $\mathbf{-18_{10}}$ and $\mathbf{-37_{10}}$
  - 2's complement representation of -18 = 11101110
  - 2's complement representation of -37 = 11011011
  - Result = $(11001001)_2 = (-55)_{10}$

Addition Using 2's Complement Arithmetic

Steps to do addition in 2's complement arithmetic:

1. Represent the two numbers to be added in 2's complement form
2. Do the addition using basic rules of binary addition
3. Disregard the final carry, if any
4. The result of addition is in 2's complement form

Addition Using the 2's Complement Method

- When overflow occurs, the result is wrong:

$$
\begin{array}{rl}
 & X = 0100\ 1011 = +75 \\
+ & Y = 0101\ 0001 = +81 \\
\hline
 & S = 1001\ 1100 \neq +156
\end{array}
$$

$(S = -2^7 + 2^4 + 2^3 + 2^2 = -100)$

$$
\begin{array}{rl}
 & X = 1001\ 1000 = -104 \\
+ & Y = 1011\ 0110 = -74 \\
\hline
 & S = 0100\ 1110 \neq -178
\end{array}
$$

$C_{out} = 1 \rightarrow$ **Discard**

$(S = 2^6 + 2^3 + 2^2 + 2^1 = 78)$

Basic Rules of Binary Subtraction

1. 0 - 0 = 0
2. 1 - 0 = 1
3. 1 - 1 = 0
4. 0 - 1 = 1 (with a borrow of 1 from the next more significant bit)

Subtraction of Larger-Bit Binary Numbers

$$A - B - B_{in}$$

Inputs			Outputs	
Minuend (A)	Subtrahend (B)	Borrow-in (B_{in})	Difference (D)	Borrow-out (B_o)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Subtraction Using 2's Complement Arithmetic

1. Both minuend and subtrahend are positive. The subtrahend is the smaller of the two.
2. Both minuend and subtrahend are positive. The subtrahend is the larger of the two.
3. The minuend is positive. The subtrahend is negative and smaller in magnitude.
4. The minuend is positive. The subtrahend is negative and greater in magnitude.
5. Both minuend and subtrahend are negative. The minuend is the smaller of the two.n
6. Both minuend and subtrahend are negative. The minuend is the larger of the two.

Subtraction Using 2's Complement Arithmetic

- **Case 1**: +24 – (+14)
  - 2's complement representation of +24 = 00011000
  - 2's complement representation of +14 = 00001110
  - 2's complement of the subtrahend (i.e. +14) = 11110010 (i.e. -14)
  - Result = $(00001010)_2 = (+10)_{10}$
- **Case 2**: +14 – (+24)
  - 2's complement representation of +24 = 00011000
  - 2's complement representation of +14 = 00001110
  - 2's complement of the subtrahend (i.e. +24) = 11101000 (i.e. -24)
  - Result = $(11110110)_2 = (-10)_{10}$

Subtraction Using 2's Complement Arithmetic

- **Case 3**: +24 – (-14)
  - 2's complement representation of +24 = 00011000
  - 2's complement representation of -14 = 11110010
  - 2's complement of the subtrahend (i.e. -14) = 00001110 (i.e. +14)
  - Result = $(00100110)_2 = (+38)_{10}$
- **Case 4**: +14 – (-24)
  - 2's complement representation of -24 = 11101000
  - 2's complement representation of +14 = 00001110
  - 2's complement of the subtrahend (i.e. -24) = 00011000 (i.e. +24)
  - Result = $(00100110)_2 = (+38)_{10}$

Subtraction Using 2's Complement Arithmetic

- **Case 5**: -24 – (-14)
  - 2's complement representation of -24 = 11101000
  - 2's complement representation of -14 = 11110010
  - 2's complement of the subtrahend (i.e. -14) = 00001110 (i.e. +14)
  - Result = $(11110110)_2 = (-10)_{10}$
- **Case 6**: -14 – (-24)
  - 2's complement representation of -14 = 11110010
  - 2's complement representation of -24 = 11101000
  - 2's complement of the subtrahend (i.e. -24) = 00011000 (i.e. +24)
  - Result = $(00001010)_2 = (+10)_{10}$

Subtraction Using 2's Complement Arithmetic

Steps to do substraction in 2's complement arithmetic:

1. Represent the minuend and subtrahend in 2's complement form
2. Find the 2's complement of the subtrahend
3. Add the 2's complement of the subtrahend to the minuend
4. Disregard the final carry, if any
5. The result is in 2's complement form
6. 2's complement notation can be used to perform subtraction when the expected result of subtraction lies in the range from -2^{n-1} to $+(2^{n-1}-1)$, n being the number of bits used to represent the numbers.

Binary Multiplication

- The basic rules of multiplication:

1. 0 x 0 = 0
2. 0 x 1 = 0
3. 1 x 0 = 0
4. 1 x 1 = 1

- Microprocessors and microcomputers the following algorithms to do binary multiplication:

1. *Repeated Left-Shift and Add Algorithm*
2. ***Repeated Right-Shift and Add Algorithm***

Repeated Left-Shift and Add Algorithm

$$
\begin{array}{rl}
 & 1\ 0\ 1\ 1\ 1 \ldots\ldots\ldots\ldots\ (23)_{10} \\
\times & 1\ 1\ 0 \ldots\ldots\ldots\ldots\ (6)_{10} \\
\hline
 & 0\ 0\ 0\ 0\ 0 \\
 & 1\ 0\ 1\ 1\ 1 \\
 & 1\ 0\ 1\ 1\ 1 \\
\hline
 & 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0
\end{array}
$$

Repeated Add and Right-Shift Algorithm

1 0 1 1 1 1 1 0	Multiplicand Multiplier
0 0 0 0 0 + 0 0 0 0 0	Start
0 0 0 0 0	Result of first addition
0 0 0 0 0 + 1 0 1 1 1	0 (Result of addition shifted one bit to right)
1 0 1 1 1	Result of second addition
0 1 0 1 1 + 1 0 1 1 1	10 (Result of addition shifted one bit to right)
1 0 0 0 1 0	Result of third addition
0 1 0 0 0 1	010 (Result of addition shifted one bit to right)

Binary Division

- Binary division can be performed by using either:

1. *Repeated Right-Shift and Substract Algorithm*, or
2. *Repeated Left-Shift and Substract Algorithm*

Repeated Right-Shift and Substract Algorithm

- Dividend: 100110
- Divisor: 1100

	Quotient		
First step	0	1 0 0 1 1 0	Dividend
		−1 1 0 0	Divisor
Second step	1	1 0 0 1 1	First five MSBs of dividend
		−1 1 0 0	Divisor shifted to right
		0 1 1 1	First subtraction remainder
Third step	1	0 1 1 1 0	Next MSB appended
		−1 1 0 0	Divisor right shifted
		0 0 1 0	Second subtraction remainder

Repeated Left-Shift and Substract Algorithm

Quotient	1 0 0 1 −1 1 0 0	1 0
0	1 1 0 1 +1 1 0 0	Borrow exists
	1 0 0 1	Final carry ignored
	1 0 0 1 1 −1 1 0 0	Next MSB appended
1	0 1 1 1	No borrow
	0 1 1 1 0 −1 1 0 0	Next MSB appended
1	0 0 0 1 0	No borrow

Floating-Point Arithmetic

- **Homework**: Read textbooks!