EED2003 Digital Design
Presentation 1:
Digital Computers and Number Systems

Asst. Prof.Dr. Ahmet ÖZKURT
Asst. Prof.Dr Hakkı T. YALAZAN

Based on the notes of Jaeyoung Choi, choi@comp.ssu.ac.kr fall 2000

1.1 Digital Computers

- Digital Computers
  - *'information age'*
    - a prominent and growing role in modern society
  - *'generality'*
    - follow a sequence of instruction, called a program, that operates on given data
    - perform a variety of information-processing tasks

- Digital computer
  - the best-known example of a digital system
  - manipulate discrete elements of information

  (ex) 10 decimal digits, 26 letters of the alphabets, ...

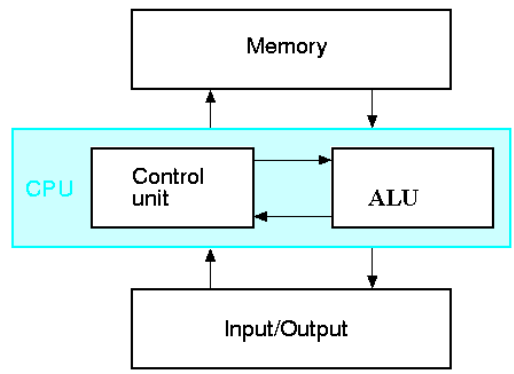
1.1 Digital Computers

- Signals
  - electrical signals such as voltages and currents
  - two discrete values
    - High (output) 4.5~5.5 (input) 3.0~5.5
    - Low (output) -0.5~1.0 (input) -0.5~2.0
  - High & Low (H & L), True & False, 1 & 0

- Binary Number System
  - a binary digit is called a *bit*
  - information is represented in group of bits
  - use various coding techniques

1.1 Digital Computers

- Computer Structure



1.1 Digital Computers

- Basic Structure
  - memory unit: stores programs, input, output, data
  - ALU Unit (Arithmetic Logic Unit): performs arithmetic and other dataprocessing operations, as specified by the program
  - control unit: supervises the flow of information between units

    (CPU = control unit + data path)
  - input device: key board
  - output device: CRT, LCD

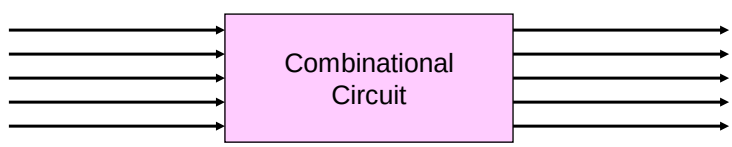
- More
  - FPU (floating-point unit)
  - MMU (memory management unit)

    (Memory: MMU + internal cache + external cache + RAM)

Combinational Logic

- logic circuits for digital systems: ***combinational*** vs ***sequential***

- Combinational Circuit
  - outputs are determined by the present applied inputs
  - performs an operation, which can be specified logically by a set of Boolean expressions



1.2 Number Systems

- decimal number (base 10 or radix 10)
  - $724.5 = 7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$
  - In general,
    - $A_nA_{n-1}....A_1A_0.A_{-1}A_{-2}....A_{-m+1}A_{-m}$
    - Each A_i coefficient is (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

- base r or radix r
  - expressed with a power series in r

    $A_n\ r^n + A_{n-1}\, r^{n-1} + ... + A_1\, r^1 + A_0\, r^0 + A_{-1}\, r^{-1} + A_{-2}\, r^{-2} + ... + A_{-m+1}\, r^{-m+1} + A_{-m}\, r^{-m}$
  - expresses in positional notation
    - $A_nA_{n-1}....A_1A_0.A_{-1}A_{-2}....A_{-m+1}A_{-m}$
  - . is called radix point

1.2 Number Systems

- A_n is the most significant digit (msd)
- A_{-m} is the least significant digit (lsd)
- enclose coefficients in parentheses and place a subscript

  $(312.4)_5 = 3 \times 5^2 + 1 \times 5^1 + 2 \times 5^0 + 4 \times 5^{-1}$

  $= 75 + 5 + 2 + 0.8 = (82.8)_{10}$
- in computer work, binary, octal, and hexadecimal is popular

- Binary Numbers
  - base 2 with two digits: 0 & 1

    $(11010)_2 = 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (26)_{10}$
  - digits in a binary numbers are called bits
  - powers of two are listed in Table 1-1.

1.2 Number Systems

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

- Table 1-1: Powers of Two

2^{10} = 1024 referred to as K (Kilo); 2^{20} as M (Mega); 2^{30} as G (Giga)

1.2 Number Systems

- conversion of decimal to binary
  - successively subtracts powers of two from the decimal number
  - $(625)_{10}$ = (?)$_2$

625 - 512 = 113	$512 = 2^9$
113 - 64 = 49	$64 = 2^6$
49 - 32 = 17	$32 = 2^5$
17 - 16 = 1	$16 = 2^4$
1 - 1 = 0	$1 = 2^0$

$(625)_{10} = 2^9 + 2^6 + 2^5 + 2^4 + 2^0 = (1001110001)_2$

1.2 Number Systems

- Octal and Hexadecimal Number
  - octal number - base 8 (0, 1, ..., 6, 7)

    $(127.4)_8 = 1x8^2 + 2x8^1 + 7x8^0 + 4x8^{-1} = (87.5)_{10}$

    hexadecimal number - base 16 (0,1,....,9,A,B,C,D,E,F)

    $(B65F)_{16} = 11x16^3 + 6x16^2 + 5x16^1 + 15x16^0 = (46687)_{10}$
  - 1 octal digit = 3 binary digits

    1 hexa digit = 4 binary digits
  - conversion

    (0010 1100 0110 1011. 1111 0000 0110)2 = $(2C6B.F06)_{16}$

    $(3A6.C)_{16}$ = 0011 1010 0110. 1100 = $(1110100110.11)_2$

1.2 Number Systems

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 1-2 Numbers with Different Bases

1.3 Arithmetic Operations

- addition, subtraction, and multiplication
  - same as for decimal numbers

(Ex1.1) $(59F)_{16} + (E46)_{16}$

				(1)
5 9 F		5	9	15
E 4 6		14	4	6
1 3 E 5		19	14	21
		1	3	E	5

(Ex1.2) $(762)_8 \times (45)_8$

(Octal)	(Octal)	(Decimal)	(Octal)
7 6 2	5 x 2	= 10 = 8 + 2	= 12
4 5	5 x 6 + 1	= 31 = 24 + 7	= 37
4 6 7 2	5 x 7 + 3	= 38 = 32 + 6	= 46
3 7 1 0	4 x 2	= 8 = 8 + 0	= 10
4 3 7 7 2	4 x 6 + 1	= 25 = 24 + 1	= 31
	4 x 7 + 3	= 31 = 24 + 7	= 37

1.3 Arithmetic Operations

- Conversion from Decimal to Other Base

(Ex1.3) $(153)_{10} = (\ ?\ )_8$

153/8 = 19 + 1/8 1

19/8 = 2 + 3/8 3

2/8 = 0 + 2/8 2

$(153)_{10} = (231)_8$

(Ex1.5) $(0.6875)_{10} = (\ ?\ )_2$

0.6875 x 2 = 1.375 1

0.375 x 2 = 0.75 0

0.75 x 2 = 1.5 1

0.5 x 2 = 1.0 1

$(0.6875)_{10} = (0.1011)_2$

1.4 Decimal Codes

- decimal number system (people are accustomed to)
  (vs) binary number system (natural for computer)
- 2 ways
  - convert decimal numbers to binary
    - perform all arithmetic calculation in binary
      and then convert the binary results back to decimal
  - perform the arithmetic operations with decimal numbers
    when they are stored in coded form
- n-bit binary code
  - a group of n bits up to 2n distinct combinations of 1's & 0's
  - 2-bit binary code: 00, 01, 10, 11

1.4 Decimal Codes

- 10 decimal digits
  - 4-bit binary code (6 are unassigned)
  - numerous different binary codes
  - BCD (Binary Coded Decimal)

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

  - $(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (101110001)_2$
  - BCD numbers are decimal numbers, not binary numbers

1.4 Decimal Codes

- BCD Addition

```
  4  0100        4  0100          8   1000
 +5  0101       +8  1000         +9   1001
  9  1001       12  1100         17  1 0001
                    +0110             +0110
                 1  0010          1  0111
```

  - add two BCD numbers as if two binary numbers
  - if sum is greater than or equal to 1010, add 0110

1.5 Alphanumeric Codes

- handle of data of numbers and letters
  - set of elements include 10 digits, 26 letters, special characters
  - 36 ~ 64 letters if only capital letters: need 6 bits
    64 ~ 128 letters if upper/lower letters: need 7 bits

- ASCII Character Code
  - standard binary code is ASCII (Table 1.4)
  - ASCII contain 94 graphic chars + 34 control chars

- Parity Bit
  - ASCII is a 7-bit code + 1 bit => 8-bit (1 byte)
    \---- used for specific purpose

1.5 Alphanumeric Codes

- parity bit: total number of 1 is even (even parity)
  total number of 1 is odd (odd parity)

		(even parity)	(odd parity)
ASCII A =	1000001	01000001	11000001
ASCII T =	1010100	11010100	01010100

- helpful in detecting errors during the transmission of information

- Unicode
  - a new standard for 16-bit alphanumeric codes
  - referred to as Unicode/10646
  - 16 bits provide 65,536 code words,
    - represent the symbols and ideographs of the world's languages
  - 16 bits, implemented in computers by 2 bytes
    - little-endian vs big-endian

Complements

- Complements
  - 2 types:
    - radix complement: r's complement
    - diminished radix complement: (r-1)'s complement
      - 2's & 1's for binary numbers
      - 10's & 9's for decimal numbers
  - 1's complement of N (binary number): $(2^n - 1) - N$
    - 1's comp of 1011001 ==> 0100110
    - 1's comp of 0001111 ==> 1110000

Complements

- 2's complement of N: 2^n - N for N != 0, 0 for N = 0
  - add 1 to the 1's complement
  - 2's comp of 101100 ==> 010011 + 1 ==> 010100
  - leaving all least significant 0's and the first 1 unchanged then replacing 1's with 0's, 0's with 1's
  - 2's comp of 1101100 ==> 0010100
  - 2's complement of N is 2^n – N
    & the complement of the complement is 2^n - (2^n-N) = N

Complements

- Subtraction with Complements
  - (M - N)
    1) add 2's comp of the subtrahend N to the minuend M
    M + (2^n-N) = M - N + 2^n
    2) if M ≥ N, the end cary is discarded
    3) if M < N, the result is 2^n - (N - M)
    take the 2's complement of the sum & place a minus sign
  - avoid *overflow problem* to accomodate the sum

- Ex3.6 X = 1010100, Y = 1000011

X =	1010100
2's comp of Y =	0111101
Sum =	10010001
Discard end carry 2^7 =	- 10000000
Answer X - Y =	0010001

Y =	1000011
2's comp of X =	0101100
Sum =	1101111

Answer Y - X = -0010001