ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Digital Systems and Binary Numbers

Computer Organization
502044

Acknowledgement

This slide show is intended for use in class, and is not a complete document. Students need to refer to the book to read more lessons and exercises. Students have the right to download and store lecture slides for reference purposes; Do not redistribute or use for purposes outside of the course.

[1] Morris R. Mano (Author), Michael D. Ciletti, [2019] **Digital Design: With an Introduction to the Verilog HDL,** chapter 1 - Digital system and Binary numbers, 5th Edition.

🖂 **trantrungtin.tdtu.edu.vn**

Chapter Objectives

1. Understand binary number system.
2. Know how to convert between binary, octal, decimal, and hexadecimal numbers.
3. Know how to take the complement and reduced radix complement of a number.
4. Know how to form the code of a number.
5. Know how to form the parity bit of a word.

Outline

- Digital Systems
- Binary Numbers
- Binary Arithmetic
- Number-base Conversions
- Octal & Hexadecimal Numbers
- Complements
- Signed Binary Numbers
- Binary Codes
- Binary Storage & Registers
- Binary Logic

Syllabus

1.1 Digital system

- Digital age and information age
- Digital computers
  - General purposes
  - Many scientific, industrial and commercial applications

- Digital systems
  - Telephone switching exchanges
  - Digital camera, Digital TV
  - Electronic calculators, PDA's
- Discrete information-processing systems
  - Manipulate discrete elements of information
  - For example, {1, 2, 3, …} and
  - {A, B, C, …}…

1.1 Digital system

- Digital systems have ability to represent and manipulate discrete elements of information.
    - 10 decimal digits,
    - the 26 letters of the alphabet,
    - the 52 playing cards,
    - and the 64 squares of a chessboard.
- Early digital computers were used for numeric computations. In this case, the discrete elements were the digits.
    - From this application, the term digital computer emerged.

1.1 Digital system

- Discrete elements of information are represented in a digital system by physical quantities called signals. Electrical signals such as voltages and currents are the most common.
- Electronic devices called transistors predominate in the circuitry that implements these signals. The signals in most present-day electronic digital systems use just two discrete values and are therefore said to be binary.

1.1 Digital system

- A binary digit, called a bit, has two values: 0 and 1.
- Discrete elements of information are represented with groups of bits called binary codes.
    - For example, the decimal digits 0 through 9 are represented in a digital system with a code of four bits.
    - e.g., the number 7 is represented by 0111.
    - we could write (0111)2
    - Then 01112 = 710

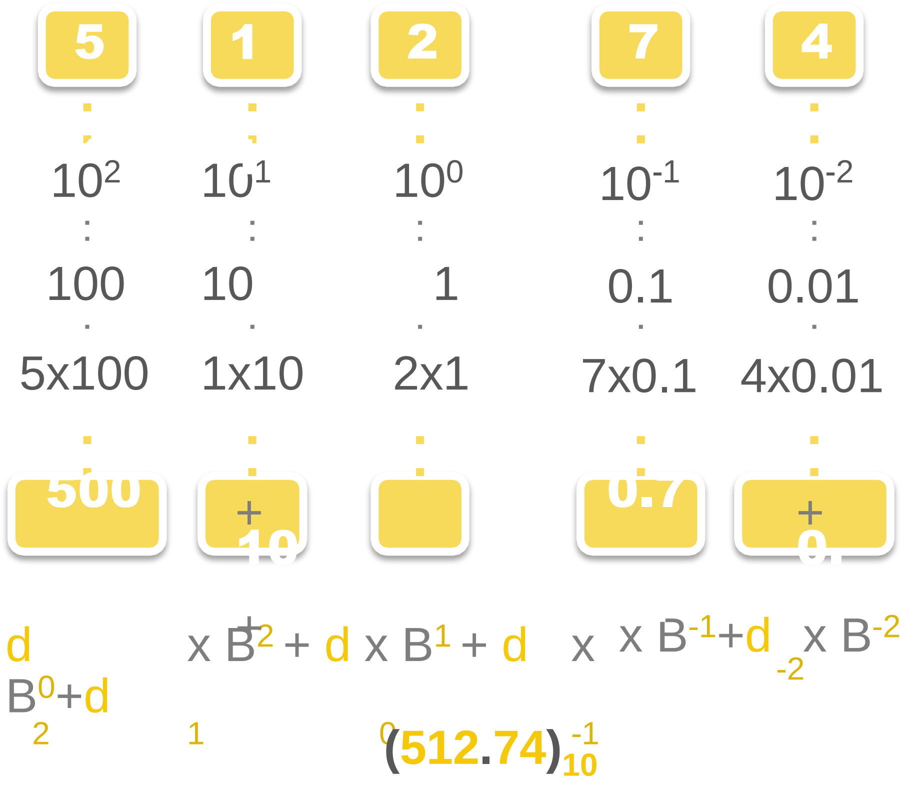
1.1 Digital system

- A digital system is an interconnection of digital modules. To understand the operation of each digital module, it is necessary to have a basic knowledge of digital circuits and their logical function.

1.2 Binary numbers

Decimal Number System

- Base (also called radix) = 10
  - 10 digits
  - { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Digit Position
  - Integer & fraction
- Digit Weight
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation



1.2 Binary numbers

- A weighted-positional number system
  - Base or radix is 10 (the base or radix of a number system is the total number of symbols/digits allowed in the system)
  - Symbols/digits = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
  - Position is important, as the value of each symbol/digit is dependent on its type and its position in the number
  - Example, the 9 in the two numbers below has different values:
    - $(7594)_{10} = (7 \times 10^3) + (5 \times 10^2) + (9 \times 10^1) + (4 \times 10^0)$
    - $(912)_{10} = (9 \times 10^2) + (1 \times 10^1) + (2 \times 10^0)$
  - In general:

$$(a_n a_{n-1} \ldots a_0 \,.\, f_1 f_2 \ldots f_m)_{10} = (a_n \times 10^n) + (a_{n-1} \times 10^{n-1}) + \ldots + (a_0 \times 10^0) + (f_1 \times 10^{-1}) + (f_2 \times 10^{-2}) + \ldots + (f_m \times 10^{-m})$$

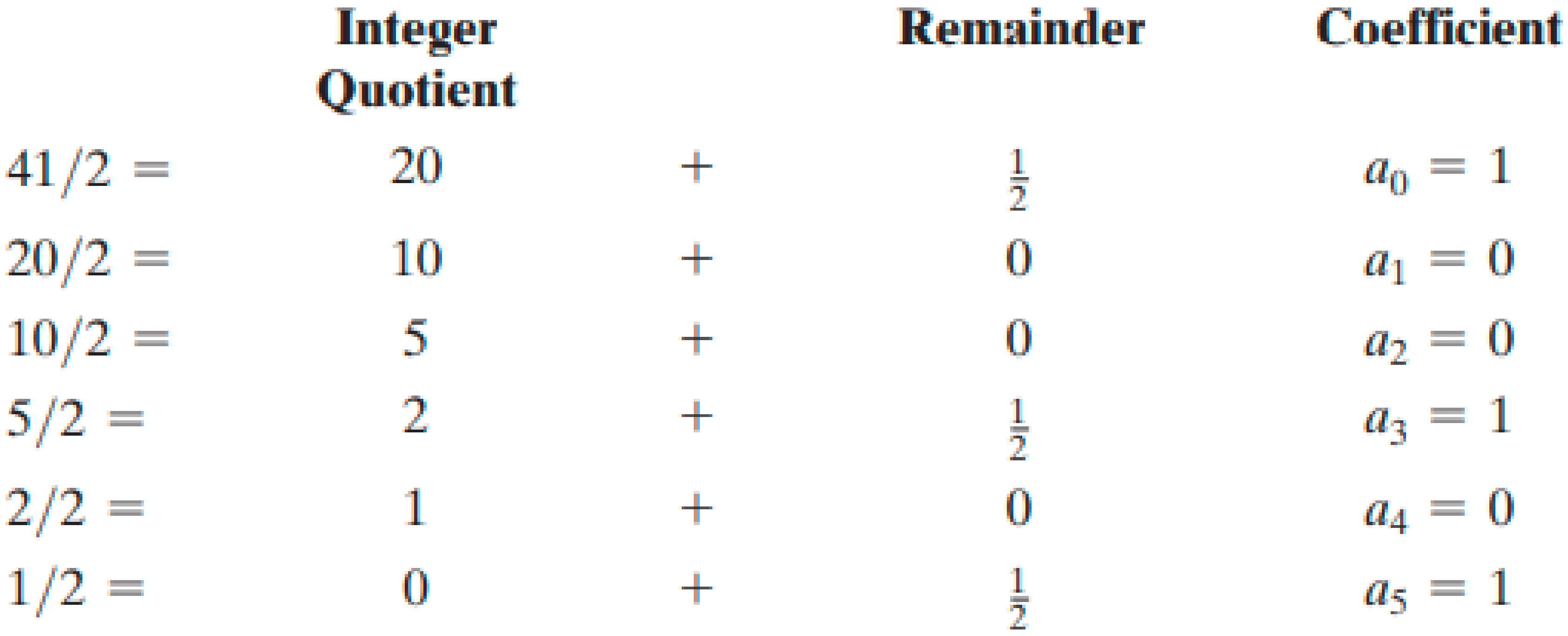
1.2 Binary numbers

- Binary (base 2)
  - Weights in powers of 2
  - Binary digits (bits): 0, 1
- Octal (base 8)
  - Weights in powers of 8
  - Octal digits: 0, 1, 2, 3, 4, 5, 6, 7.
- Hexadecimal (base 16)
  - Weights in powers of 16
  - Hexadecimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Base/radix R:
  - Weights in powers of R

1.3 Number-Base Conversions

$$\begin{tikzcd}
\text{Base-2} \arrow[ddr] & & \text{Base-2} \\
\text{Base-3} \arrow[dr] & & \text{Base-3} \\
\text{Base-4} \arrow[r] & \text{Decimal} \arrow[uur] \arrow[ur] \arrow[r] \arrow[ddr] & \text{Base-4} \\
\cdots & & \cdots\cdot \\
\text{Base-}R \arrow[uur] & & \text{Base-}R
\end{tikzcd}$$

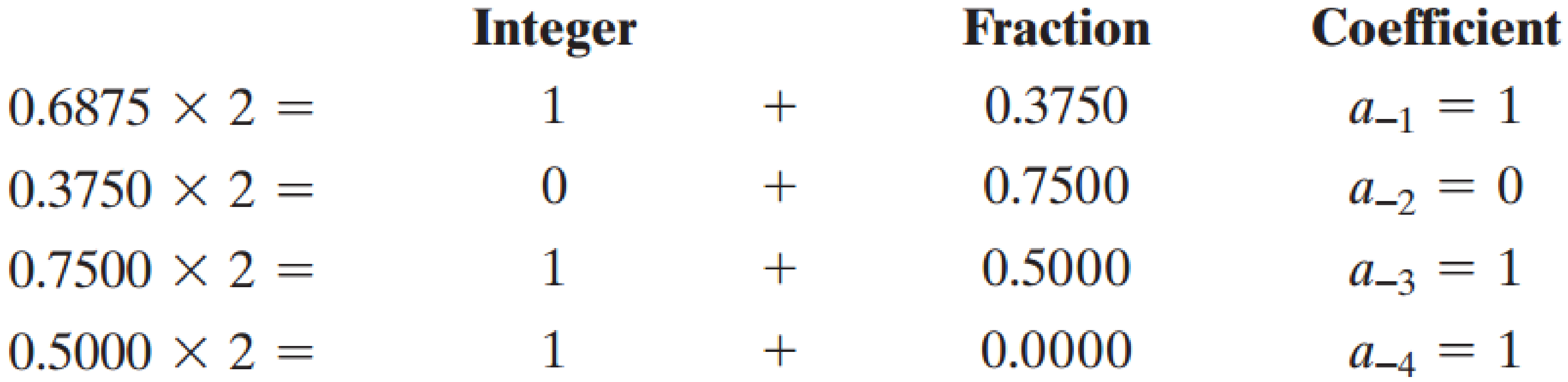
1.3 Example 1



	Integer Quotient		Remainder	Coefficient
$41/2 =$	20	$+$	$\frac{1}{2}$	$a_0 = 1$
$20/2 =$	10	$+$	0	$a_1 = 0$
$10/2 =$	5	$+$	0	$a_2 = 0$
$5/2 =$	2	$+$	$\frac{1}{2}$	$a_3 = 1$
$2/2 =$	1	$+$	0	$a_4 = 0$
$1/2 =$	0	$+$	$\frac{1}{2}$	$a_5 = 1$

1.3 Example 2

- $(0.6875)_{10} = (0.a^{-1}a^{-2}\ a^{-3}\ a^{-4})_2 = (0.1011)_2$



	Integer		Fraction	Coefficient
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

1.3 Example 3

- Conversion from **decimal** integers to any **base-R** system is similar to this example, except that division is done by **R** instead of **2**

153	
19	1
2	3
0	$2 = (231)_8$

1.3 Example 4

- $(0.513)_{10} = (0.406517\ \ldots)_8$

$$\begin{aligned}
0.513 \times 8 &= 4.104 \\
0.104 \times 8 &= 0.832 \\
0.832 \times 8 &= 6.656 \\
0.656 \times 8 &= 5.248 \\
0.248 \times 8 &= 1.984 \\
0.984 \times 8 &= 7.872
\end{aligned}$$

1.4 Octal and Hexadecimal Numbers

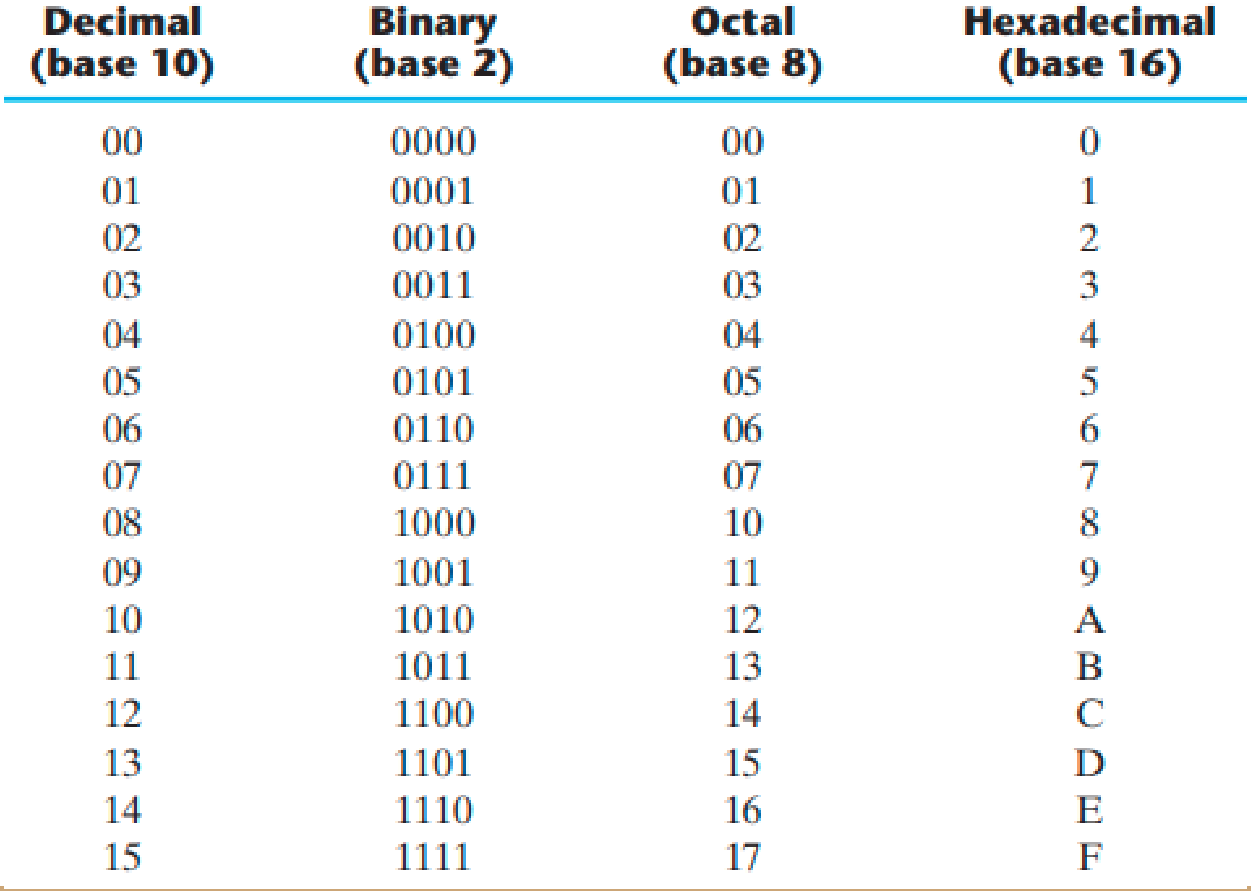
- The conversion from and to binary, octal, and hexadecimal plays an important role in digital computers.
- B/c shorter patterns of hex characters are easier to recognize than long patterns of 1's and 0's.

$$\begin{array}{ccccccccccl}
(10 & 110 & 001 & 101 & 011 & \cdot & 111 & 100 & 000 & 110)_2 & = (26153.7406)_8 \\
2 & 6 & 1 & 5 & 3 & & 7 & 4 & 0 & 6 &
\end{array}$$

1.4 Octal and Hexadecimal Numbers

- Retains the binary system in the computer, but reduces the number of digits the human must consider, utilizes the relationship between the binary number system and the octal or hexadecimal system.
- Most computer manuals use either octal or hexadecimal numbers to specify binary quantities

1.4 Numbers with Different bases



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

1.4 Convert in HEX and OCT

- Binary number is divided into groups of four / three digits

$$\begin{array}{ccccccc} (10 & 1100 & 0110 & 1011 & \cdot & 1111 & 0010)_2 = (2C6B.F2)_{16} \\ 2 & C & 6 & B & & F & 2 \end{array}$$

$$\begin{array}{ccccccc} (673.124)_8 = (110 & 111 & 011 & \cdot & 001 & 010 & 100)_2 \\ 6 & 7 & 3 & & 1 & 2 & 4 \end{array}$$

$$\begin{array}{ccccc} (306.D)_{16} = (0011 & 0000 & 0110 & \cdot & 1101)_2 \\ 3 & 0 & 6 & & D \end{array}$$

1.5 Complements of Numbers

- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.

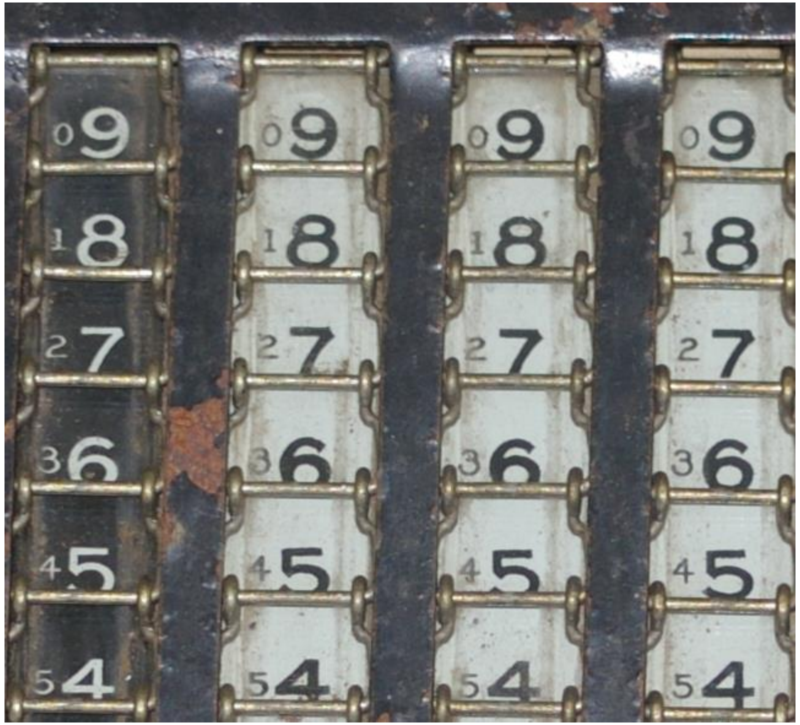
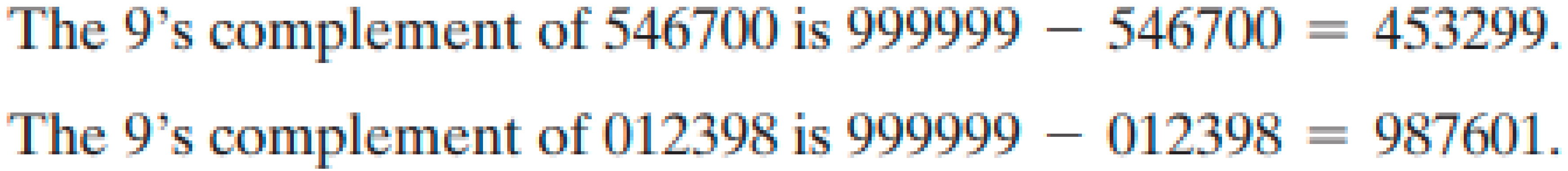


Fig. Complement numbers on an adding machine c. 1910. The smaller numbers, for use when subtracting, are the nines' complement of the larger numbers, which are used when adding

1.5 Diminished Radix Complement

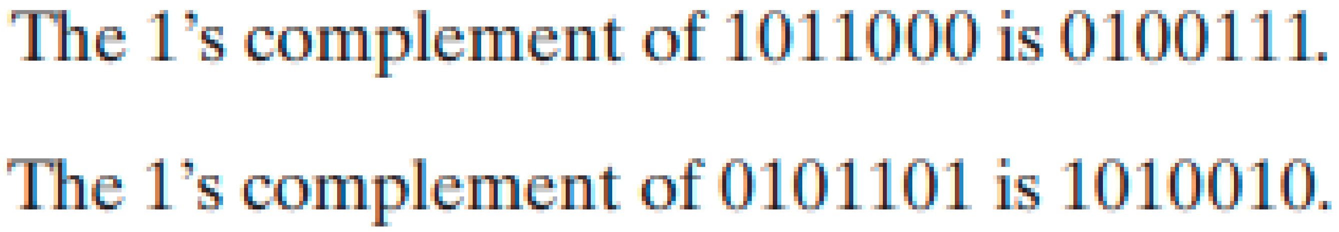
- Given a number **N** in base **R** having **n** digits, the (**R - 1**)'s complement of **N**
- Diminished radix complement, is defined as (**R^n - 1**) - **N**


The 9's complement of 546700 is $999999 - 546700 = 453299$.

The 9's complement of 012398 is $999999 - 012398 = 987601$.

1.5 Diminished Radix Complement of Binary

- The 1’s complement of a binary number is formed by changing 1’s to 0’s and 0’s to 1’s.

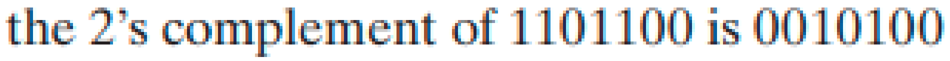


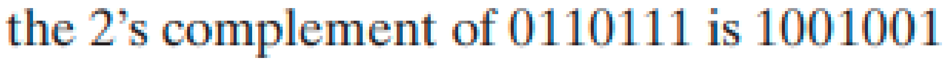
The 1's complement of 1011000 is 0100111.

The 1's complement of 0101101 is 1010010.

1.5 Radix Complement

- The **R**'s complement of an **n-digit** number **N** in base **R** is defined as
  - $\mathbf{R^n - N}$ for N $\neq 0$
  - **0** for N = 0


the 2's complement of 1101100 is 0010100


the 2's complement of 0110111 is 1001001

1.5 Subtraction with Complements

- **n-digit** unsigned numbers **M - N** in base **R** can be done as follows:
- $M + (R^n - N) = M - N + R^n$
  - If M > N, the sum will produce an end carry R^n, which can be discarded; what is left is the result M - N
  - If M < N, the sum does not produce an end carry and is equal to $R^n - (N - M)$, take the R’s complement of the sum and place a negative sign in front.

1.5 Example 1

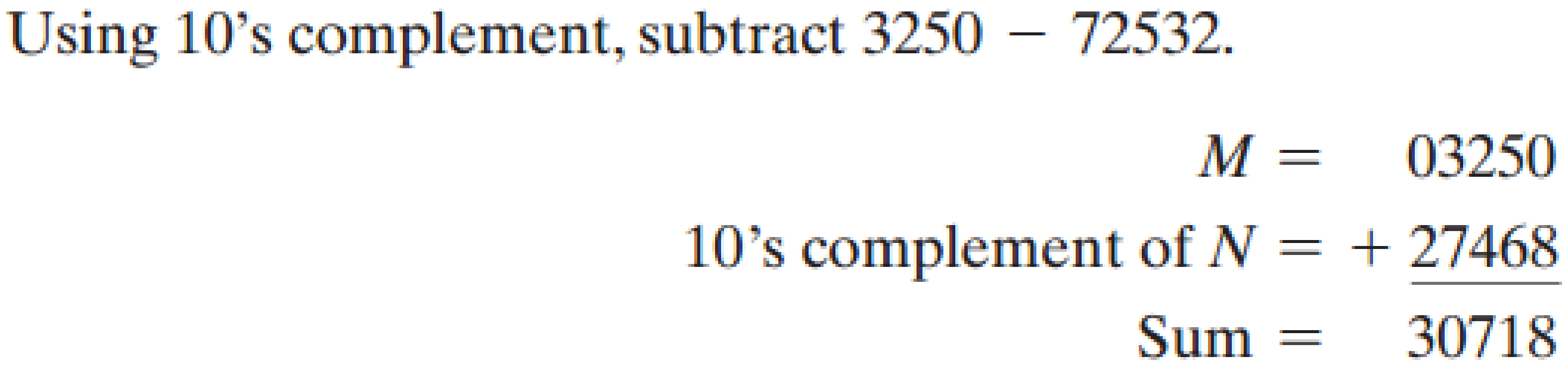
Using 10's complement, subtract 72532 − 3250.

$$
\begin{aligned}
M &= 72532 \\
\text{10's complement of } N &= +\ \underline{96750} \\
\text{Sum} &= 169282 \\
\text{Discard end carry } 10^5 &= -\ \underline{100000} \\
\textit{Answer} &= 69282
\end{aligned}
$$

1.5 Example 2

- The answer is

- (10’s complement of 30718) = - 69282



Using 10’s complement, subtract 3250 − 72532.

$$\begin{array}{rr} M = & 03250 \\ \text{10's complement of } N = & +\underline{27468} \\ \text{Sum} = & 30718 \end{array}$$

1.5 Example 3

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction **(a)** $X - Y$ and **(b)** $Y - X$ by using 2's complements.

(a)

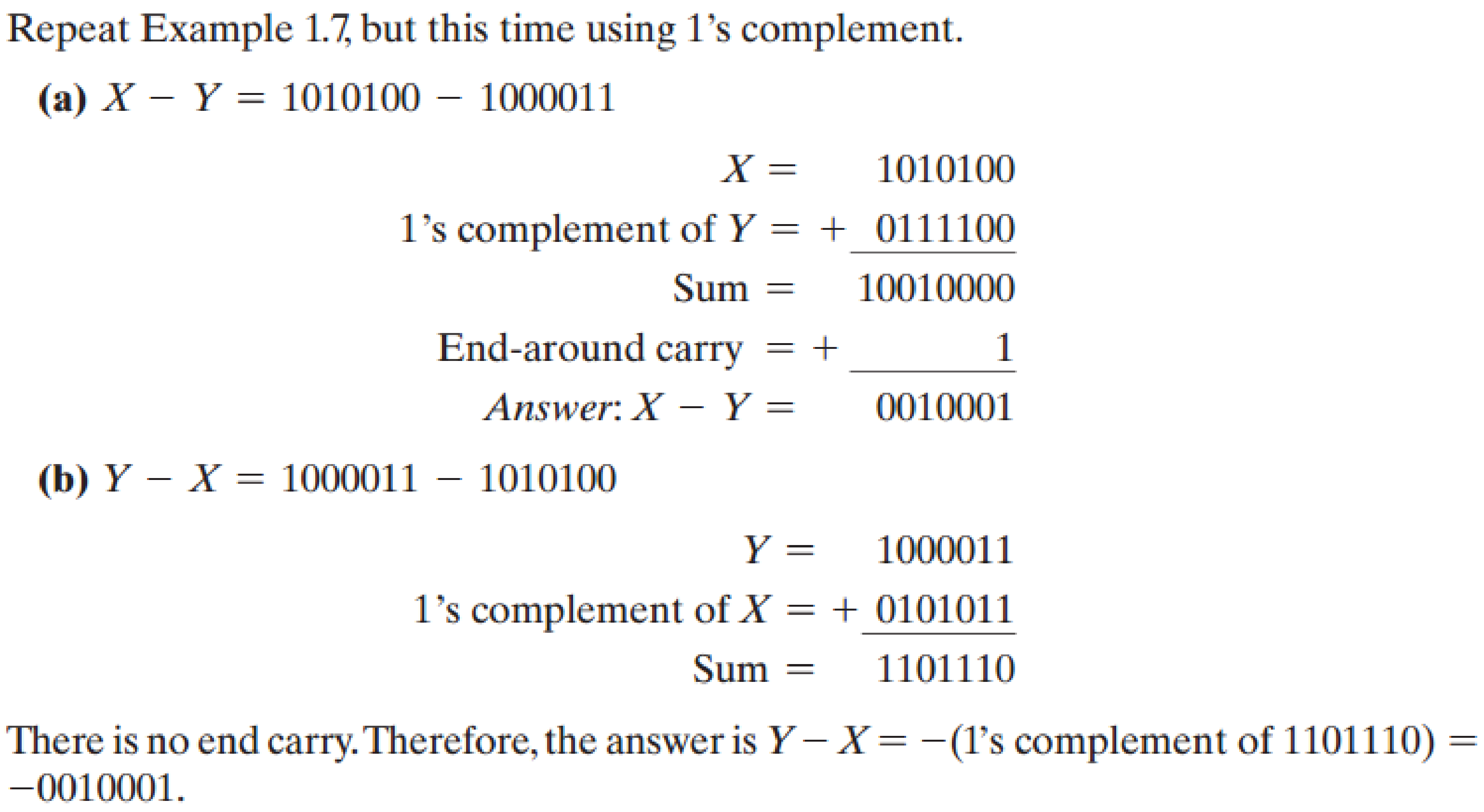
$$\begin{array}{rr} X = & 1010100 \\ \text{2's complement of } Y = & + \underline{0111101} \\ \text{Sum} = & 10010001 \\ \text{Discard end carry } 2^7 = & -\underline{10000000} \\ \textit{Answer: } X - Y = & 0010001 \end{array}$$

(b)

$$\begin{array}{rr} Y = & 1000011 \\ \text{2's complement of } X = & + \underline{0101100} \\ \text{Sum} = & 1101111 \end{array}$$

There is no end carry. Therefore, the answer is $Y - X = -(2\text{'s complement of } 1101111) = -0010001$.

1.5 Example 4



Repeat Example 1.7, but this time using 1's complement.

(a) $X - Y = 1010100 - 1000011$

$$\begin{array}{rr}
X = & 1010100 \\
\text{1's complement of } Y = & +\underline{\;0111100} \\
\text{Sum} = & 10010000 \\
\text{End-around carry} = & +\underline{\qquad\quad 1} \\
\textit{Answer}: X - Y = & 0010001
\end{array}$$

(b) $Y - X = 1000011 - 1010100$

$$\begin{array}{rr}
Y = & 1000011 \\
\text{1's complement of } X = & +\underline{\;0101011} \\
\text{Sum} = & 1101110
\end{array}$$

There is no end carry. Therefore, the answer is $Y - X = -(\text{1's complement of } 1101110) = -0010001$.

1.6 Signed Binary Numbers

- Positive integers (including zero) can be represented as unsigned numbers. To represent negative integers, we need a notation for negative values.
- BOTH of them consist of **a string of bits** when represented in a computer.
- Three representations for signed binary numbers:
  - Sign-and-Magnitude
  - 1s Complement
  - 2s Complement

1.6 Sign-Magnitude

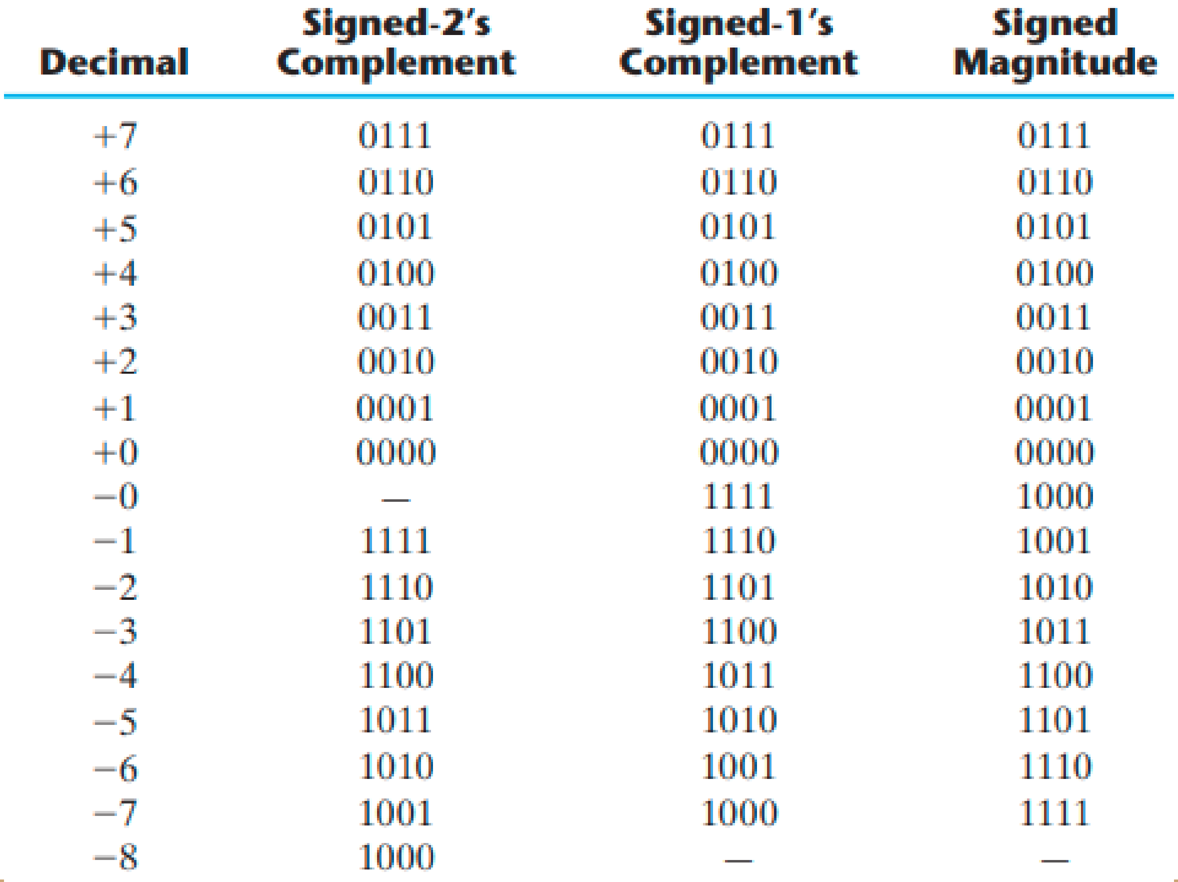
- If the binary number is signed, then the leftmost bit represents the sign and the rest of the bits represent the number (magnitude).
- For example,

Bit string	Unsigned	Sign and Magnitude
01001	9	9
11001	25	-9

1.6 Signed- complement

- More convenient to implement arithmetic operations.
- A negative number is indicated by its complement.
- Three different ways to represent -9 with eight bits:
  - signed-magnitude representation: 10001001
  - signed-1's-complement representation: 11110110
  - signed-2's-complement representation: 11110111

1.6 Signed Binary Numbers



Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

1.6 Arithmetic Addition

- If the signs are the same, we add the two magnitudes and give the sum the common sign.
- If the signs are different, we subtract the smaller magnitude from the larger and give the difference the sign of the larger magnitude

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

- In signed-complement system does not require a comparison or subtraction, but **only addition**.

1.6 Arithmetic Addition

- The addition of two signed binary numbers with negative numbers represented in signed-2's-complement form is obtained from the addition of the two numbers, including their sign bits. A **carry out** of the sign-bit position is discarded.

$$\begin{array}{rr} +\ 6 & 00000110 \\ +13 & 00001101 \\ \hline +19 & 00010011 \end{array} \qquad \begin{array}{rr} -\ 6 & 11111010 \\ +13 & 00001101 \\ \hline +\ 7 & 00000111 \end{array}$$

$$\begin{array}{rr} +\ 6 & 00000110 \\ -13 & 11110011 \\ \hline -\ 7 & 11111001 \end{array} \qquad \begin{array}{rr} -\ 6 & 11111010 \\ -13 & 11110011 \\ \hline -19 & 11101101 \end{array}$$

1.6 Arithmetic Subtraction

- $(\pm A) - (+B) = (\pm A) + (-B)$
- $(\pm A) - (-B) = (\pm A) + (+B)$

- Therefore, computers need **only one common hardware circuit** to handle both types of arithmetic .

1.6 Overflow

- Signed numbers are of a fixed range. If the result of addition/subtraction goes beyond this range, an overflow occurs.
- Overflow can be easily detected:
  - positive add positive □ negative **OR** negative add negative □ positive
- Example: 4-bit 2's-complement system
  - Range of value: -8_{10} to 7_{10}
  - $0101_{2s} + 0110_{2s} = 1011_{2s}$
    $5_{10} + 6_{10} = -5_{10}$?! (overflow!)
  - $1001_{2s} + 1101_{2s} = 10110_{2s}$ (discard end-carry) $= 0110_{2s}$
    $-7_{10} + -3_{10} = 6_{10}$?! (overflow!)

1.6 Overflow

- Which of the above is/are overflow(s)?

```
  +3         0011
+ +4       + 0100
----       --------
  +7         0111
----       --------
```

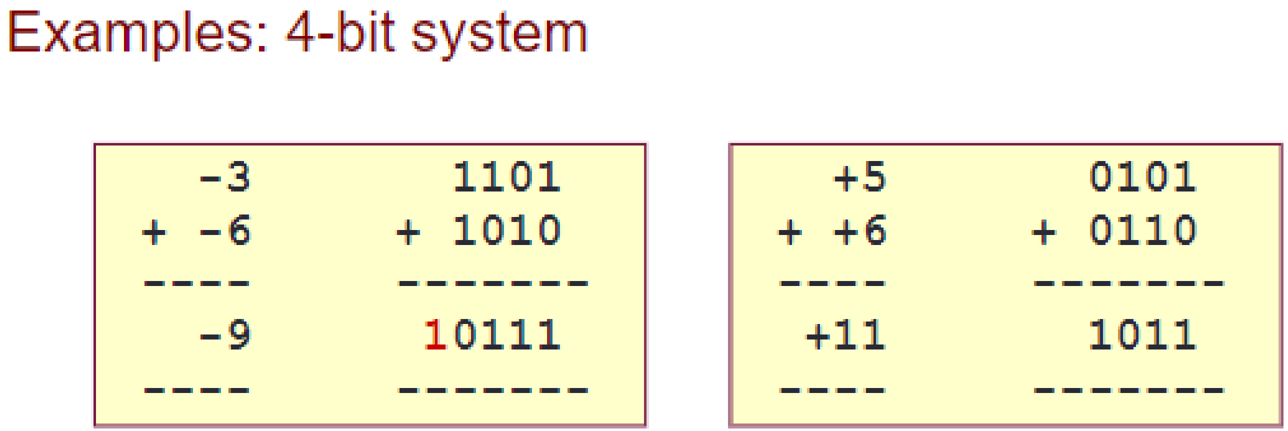
```
  -2         1110
+ -6       + 1010
----       --------
  -8        11000
----       --------
```

```
  +6         0110
+ -3       + 1101
----       --------
  +3        10011
----       --------
```

```
  +4         0100
+ -7       + 1001
----       --------
  -3         1101
----       --------
```

1.6 Overflow (cont.)

- Which of the above is/are overflow(s)?



Examples: 4-bit system

```
  -3         1101
+ -6       + 1010
----       --------
  -9        10111
----       --------
```

```
  +5         0101
+ +6       + 0110
----       --------
 +11         1011
----       --------
```

1.7 Binary Codes

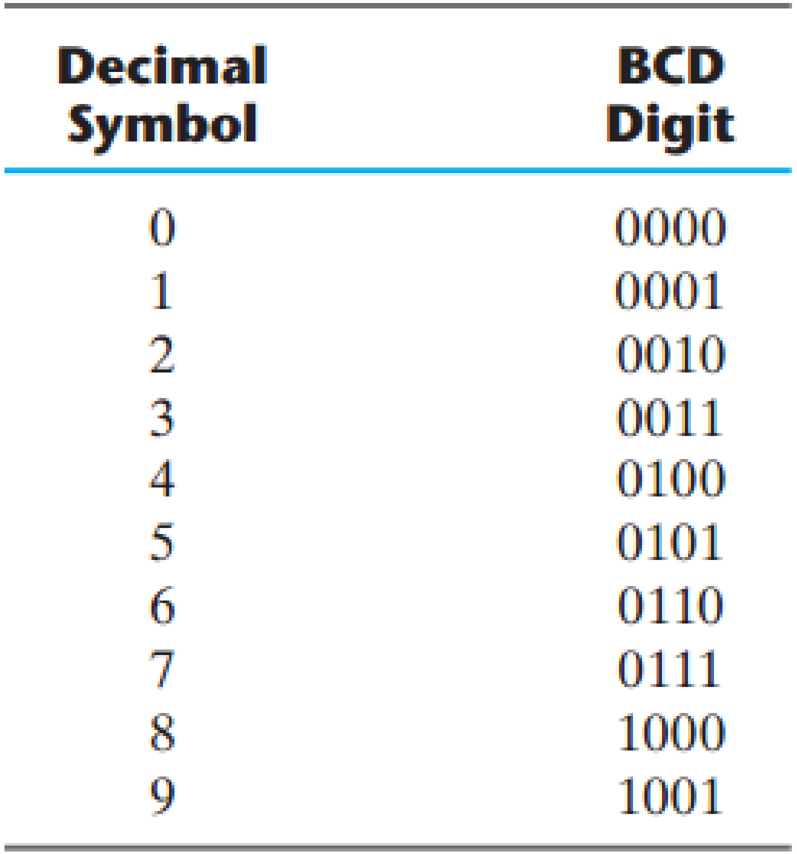
- Binary-Coded Decimal Code
- Excess-3
- 84-2-1
- Gray code
- ASCII
- Error detect code

1.7 Binary-coded decimal

- Each group of 4 bits representing one decimal digit

  $396_{10} = 0011\ 1001\ 0110_2$

- The binary combinations 1010 through 1111 are not used and have no meaning in BCD.



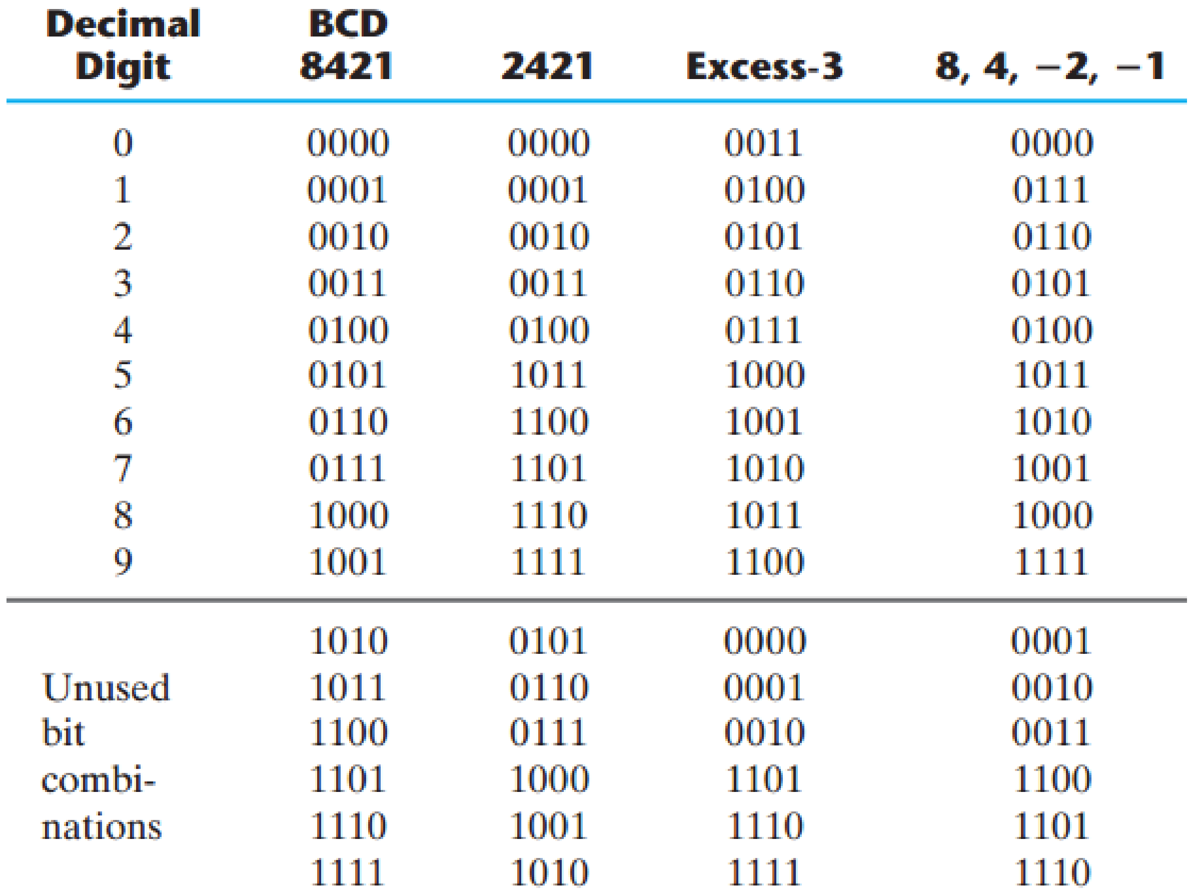
Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

1.7 BCD Addition

$$\begin{array}{rrrrrr}
4 & 0100 & 4 & 0100 & 8 & 1000 \\
+5 & +0101 & +8 & +1000 & +9 & 1001 \\
\hline
9 & 1001 & 12 & 1100 & 17 & 10001 \\
 & & & +0110 & & +0110 \\
\cline{4-4}\cline{6-6}
 & & & 10010 & & 10111
\end{array}$$

1.7 Other Decimal Codes

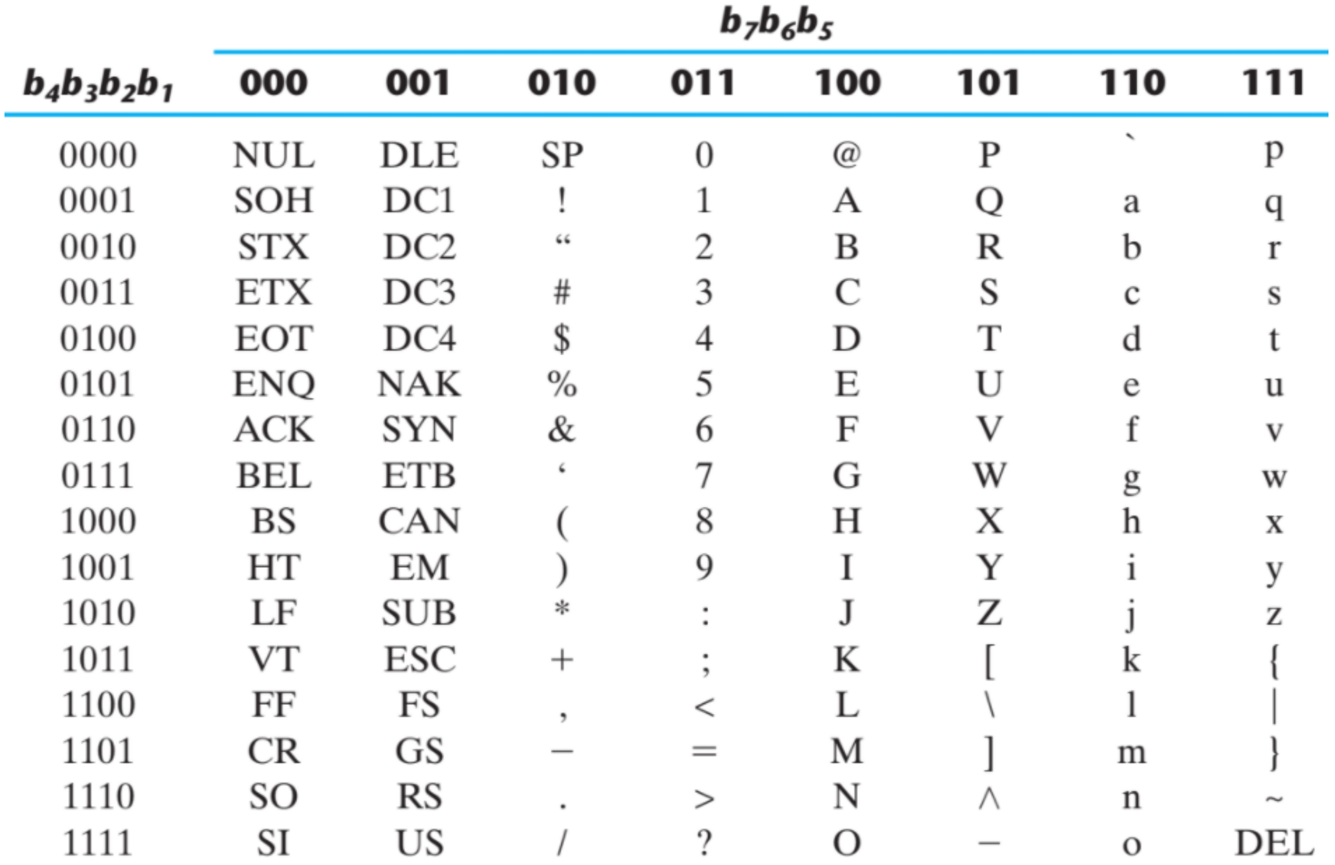
- Which is/are self-complementing code(s)?



Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

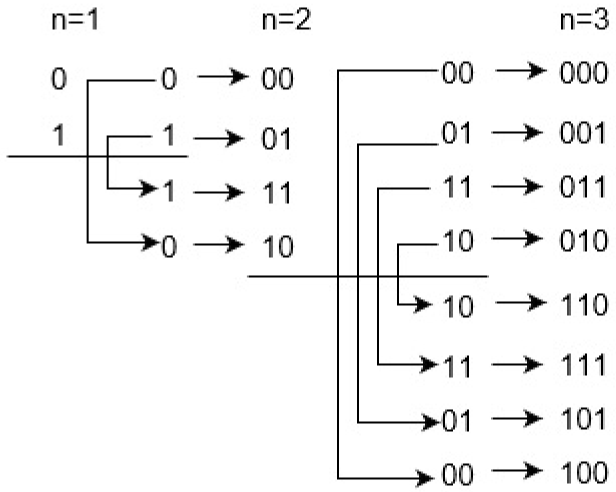
1.7 ASCII

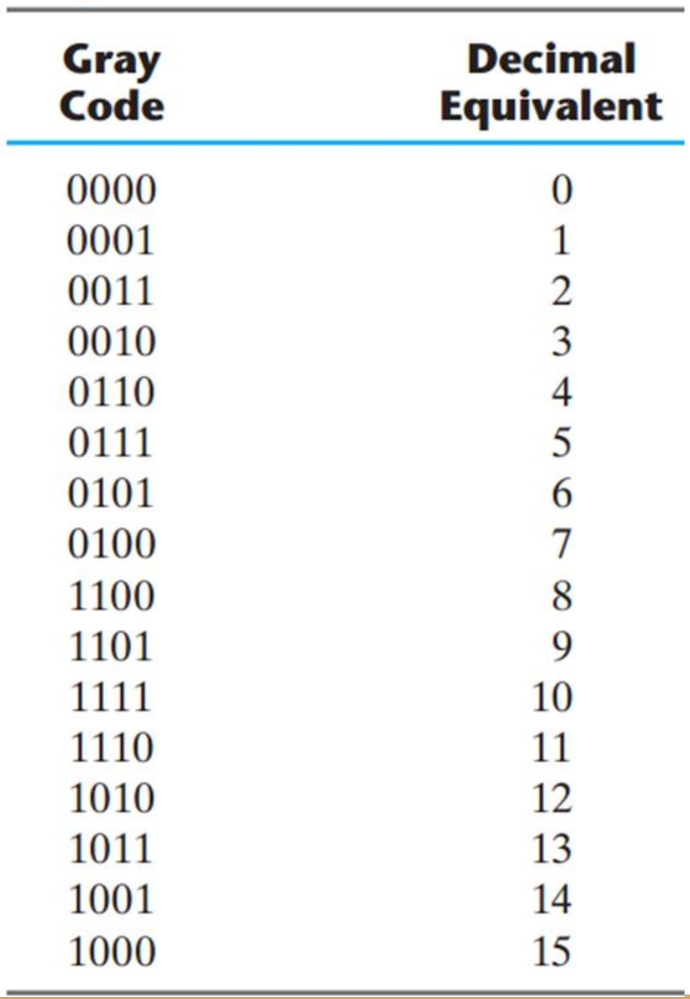
- American Standard Code for Information Interchange
- 94 graphic characters that can be printed
- 34 non-printing characters



	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	–	o	DEL

1.7 Gray code

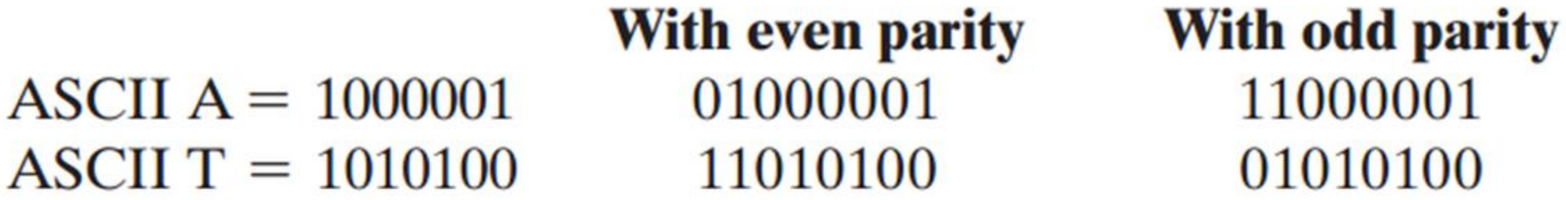




Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

1.7 Error-Detecting Code

- How to detect an error in received string of bits?



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

1.8 Binary Storage and Registers

- The binary information in a digital computer must have a physical existence in some medium for storing individual bits.
- A binary cell is a device that possesses two stable states and is capable of storing one bit (0 or 1) of information
- The information stored in a cell is 1 when the cell is in one stable state and 0 when the cell is in the other stable state.

1.8 Register

- A register is a group of binary cells. A register with **n-cells** can store any discrete quantity of information that contains **n-bits.**
- 16-bit register with the following binary content:

1100 0011 1100 1001

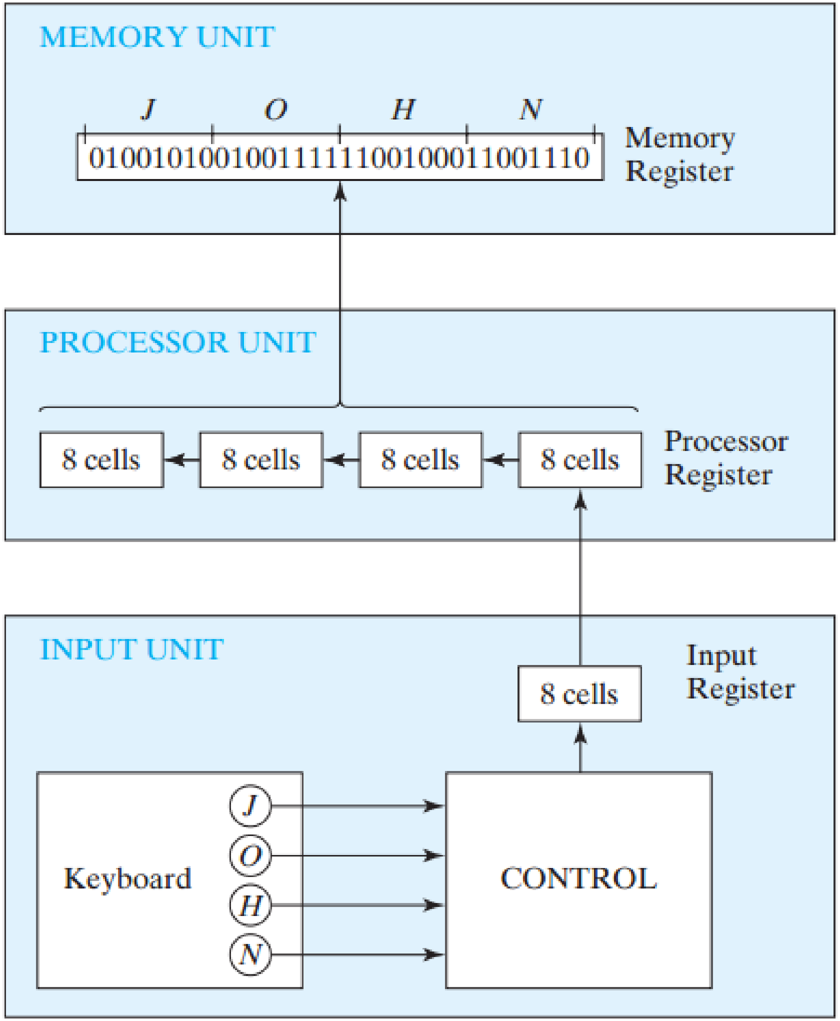
- any binary number from 0 to $2^{16} - 1$
- two meaningful characters
- may be interpreted differently for different types of data

1.8 Register Transfer

- In digital systems, a register transfer operation is a basic operation that consists of a transfer of binary information from one set of registers into another set of registers. The transfer may be direct, from one register to another, or may pass through data-processing circuits to perform an operation.

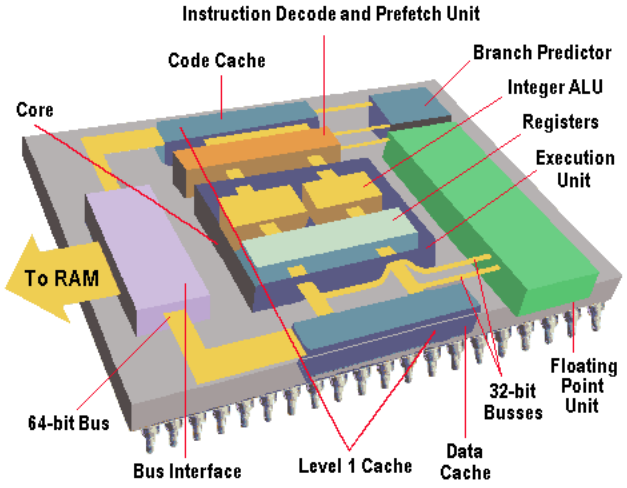
1.8 Reg. Transfer

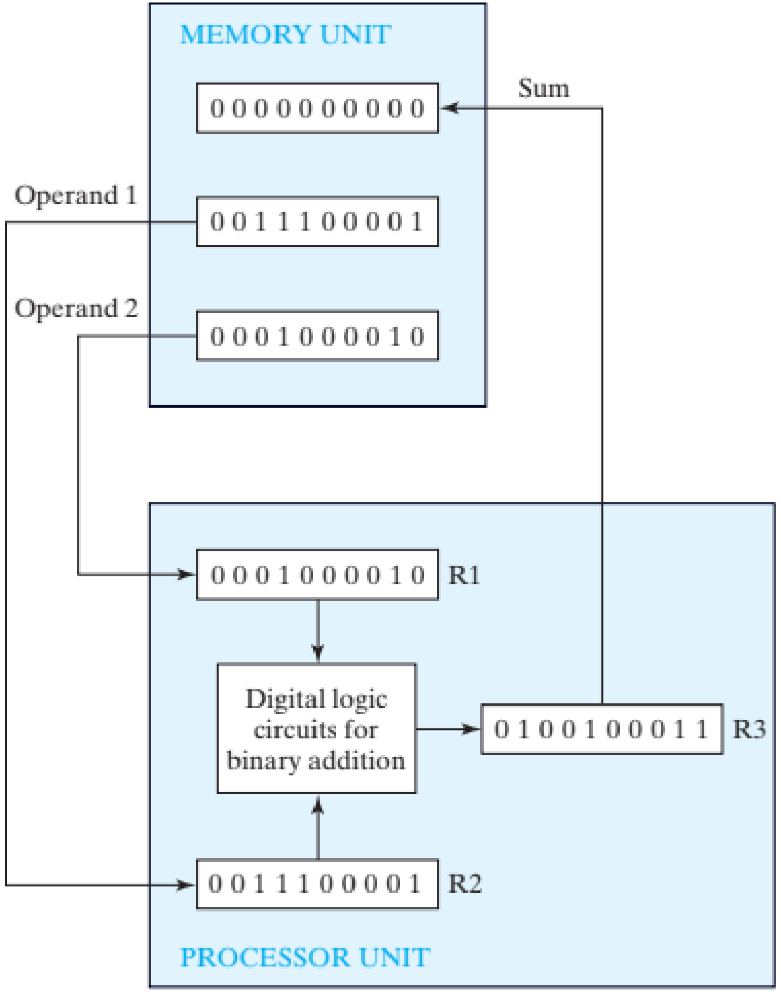
- In digital systems, a register transfer operation is a basic operation that consists of a transfer of binary information from one set of registers into another set of registers.
  - may be direct
  - may pass through data-processing circuits



1.8 Reg. Transfer

- The device most commonly used for holding data is a register.

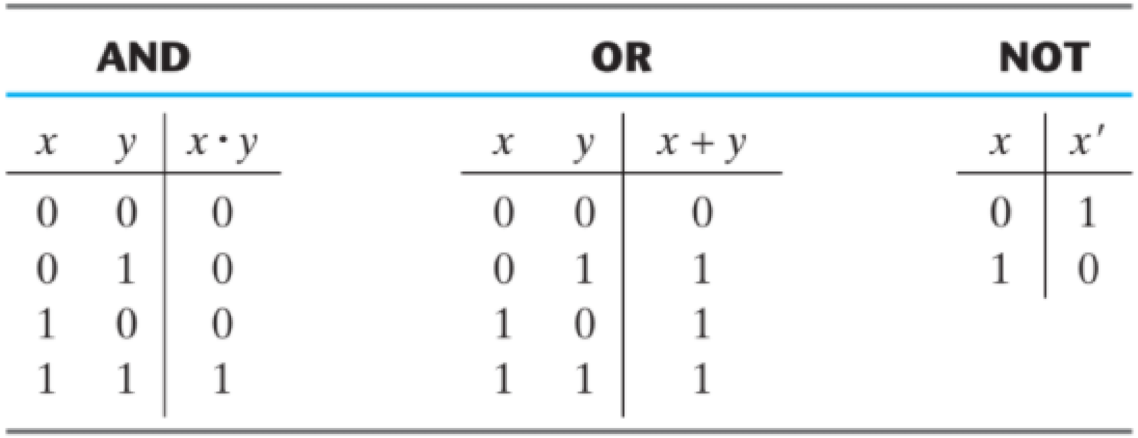




1.9 Binary Logic

- Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning.
- Binary logic should not be confused with binary arithmetic.
- There are three basic logical operations: AND, OR, and NOT.

Truth Tables of Logical Operations



AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

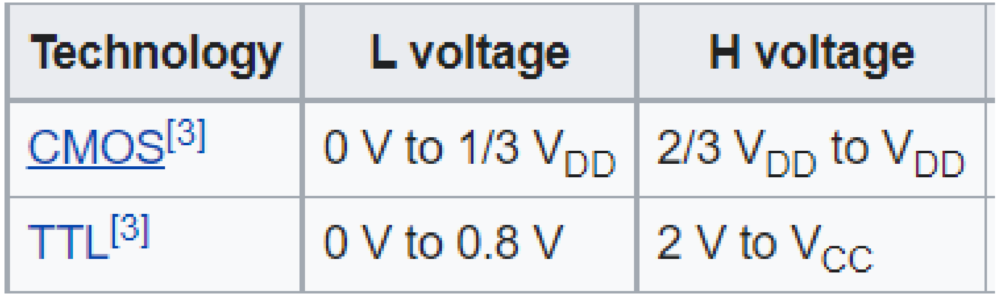
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

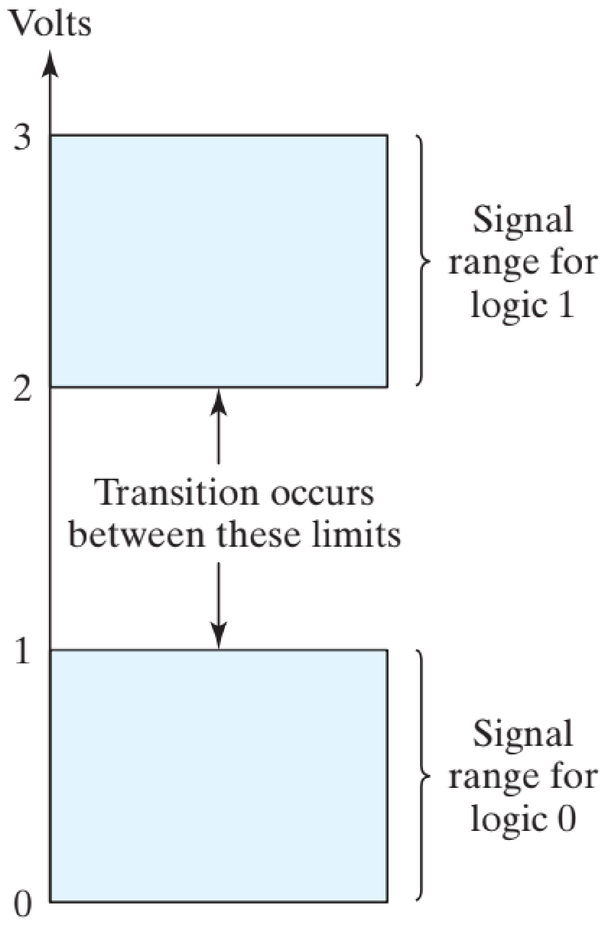
x	x'
0	1
1	0

1.9 Binary Logic

- Logic voltage levels may be vary.



Technology	L voltage	H voltage
CMOS[3]	0 V to 1/3 V_{DD}	2/3 V_{DD} to V_{DD}
TTL[3]	0 V to 0.8 V	2 V to V_{CC}



1.10 Summary

- Our computers are digital systems, and implemented into Personal computers, Servers and Embedded computers.
- Binary numbers / codes are suitable for digital circuits which works on LOW / HIGH signals.
- Decimal, Binary, Octal and Hexadecimal are radixes using in computer science.
- Numbers for calculating, Codes for transferring.
- Register, memory are physical devices that store the binary information.

1.11 Further topics

- BCD code
- ASCII
- Storage register
- Binary logic
- BCD addition
- Binary codes
- Binary numbers
- Excess-3 code