CS204: 數位系統設計

Digital Systems and Binary Numbers

Outline of Chapter 1

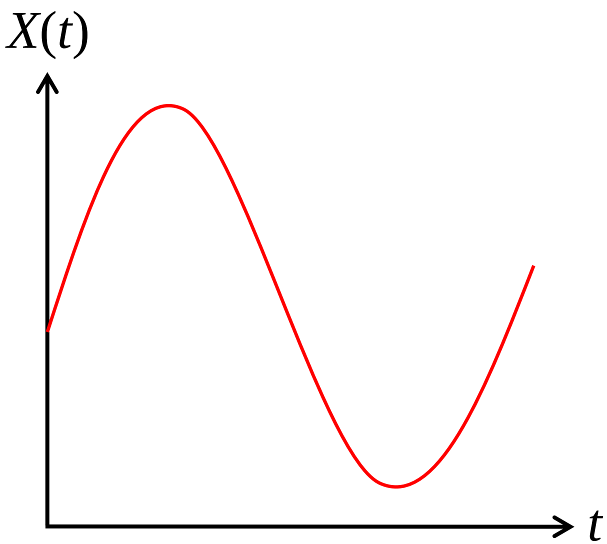
- 1.1 Digital Systems
- 1.2 Binary Numbers
- 1.3 Number-base Conversions
- 1.4 Octal and Hexadecimal Numbers
- 1.5 Complements
- 1.6 Signed Binary Numbers
- 1.7 Binary Codes
- 1.8 Binary Storage and Registers
- 1.9 Binary Logic

Questions

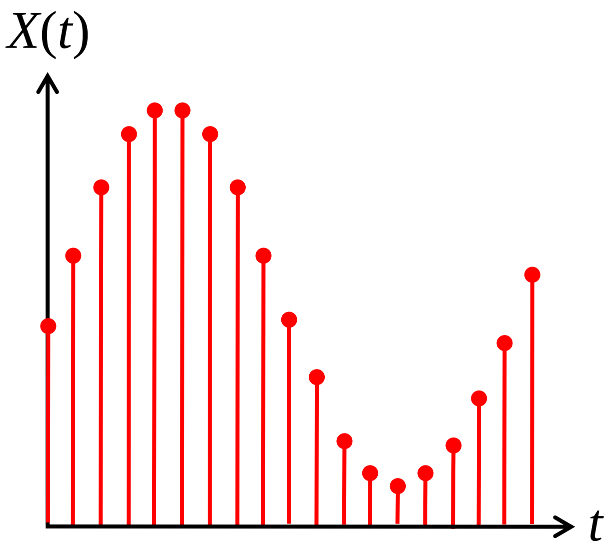
- **What is Digital System?**
- **What is Digital Signal?**

1.1 Digital Systems

- **Digital signal**
  - **The physical quantities or signals can assume only discrete values**
- **Analog signal**
  - **The physical quantities or signals may vary continuously over a specified range**



Analog signal



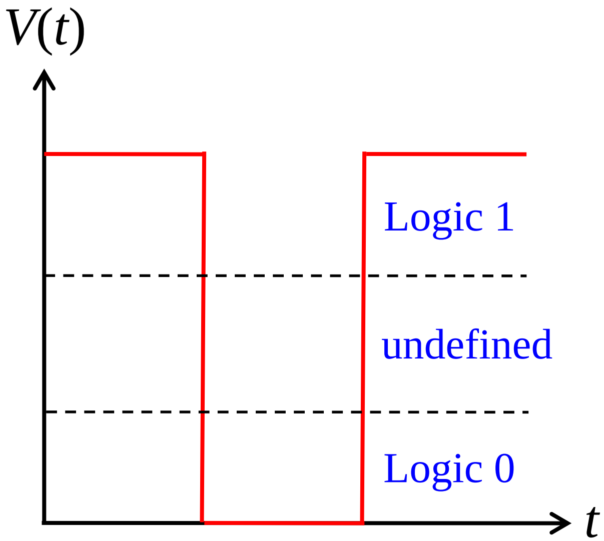
Digital signal

Digital Systems

- A system that manipulates discrete elements of information
  - A finite number of elements: e.g. {1, 2, 3, …} and {A, B, C, …}
- Digital system examples
  - Telephone switching exchanges
  - Digital camera
  - Electronic calculators, PDA's
  - Digital TV
- Digital computers
  - General purposes
  - Many scientific, industrial and commercial applications

Binary Digital Signal

- Binary values are represented abstractly by:
  - Digits 0 and 1
  - Words (symbols) False (F) and True (T)
  - Words (symbols) Low (L) and High (H)
  - And words On and Off
- The most prevalent discrete values
- Binary values are represented by values or ranges of values of physical quantities



Binary digital signal

Question

- How to represent discrete values by 0 and 1?

- If 000 -> 0 then 011 -> ?
  010 -> 2
  100 -> 4
  110 -> 6

- How about symbols?
  - A, B, C, …, ~, @, ..

1.2 Binary Numbers

Decimal number

$$\ldots a_5 a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \ldots$$

Decimal point

$$a_j$$

Power

$$\cdots + 10^5 a_5 + 10^4 a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} + 10^{-2} a_{-2} + 10^{-3} a_{-3} + \cdots$$

Example:

$$7,329 = 7 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$$

General form of base-r system

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \cdots + a_{-m} \cdot r^{-m}$$

Coefficient: $a_j = 0$ to $r - 1$

Binary Numbers

Example: Base-2 number

$$(110101)_2 = 32 + 16 + 4 + 1 = (53)_{10}$$

Special Powers of 2

- **2^{10} (1024) is Kilo, denoted "K"**
- **2^{20} (1,048,576) is Mega, denoted "M"**
- **2^{30} (1,073,741,824) is Giga, denoted "G"**

Powers of two

Table 1.1
Powers of Two

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

Binary Arithmetic

Addition

Augend:	101101
Addend:	+100111
Sum:	1010100

Subtraction

Minuend:	101101
Subtrahend:	- 100111
Difference:	0000110

Multiplication

Multiplicand	1011
Multiplier	× 101
Partial Products	1011
	0000
	1011
Product	110111

Arithmetic operations with numbers in base-*r* follow the same rules as decimal numbers.

1.3 Number-Base Conversions (p.22)

Name	Radix	Digits
Binary	**2**	**0,1**
Octal	**8**	**0,1,2,3,4,5,6,7**
Decimal	**10**	**0,1,2,3,4,5,6,7,8,9**
Hexadecimal	**16**	**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**

- **The six letters (in addition to the 10 integers) in hexadecimal represent: 10 (A), 11 (B), 12 (C), 13 (D), 14 (E), and 15 (F), respectively.**

Number-Base Conversions (p.22)

Example 1.1

- Convert decimal 41 to binary. The process is continued until the *integer quotient* becomes 0.

	Integer Quotient	Remainder	Coefficient
41/2=	**20**	**1**	$a_0 = 1$
20/2=	**10**	**0**	$a_1 = 0$
10/2=	**5**	**0**	$a_2 = 0$
5/2=	**2**	**1**	$a_3 = 1$
2/2=	**1**	**0**	$a_4 = 0$
1/2=	**0**	**1**	$a_5 = 1$

$$(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$$

Number-Base Conversions (p.23, 24)

Example 1.2

- Convert decimal 153 to octal. The required base *r* is 8.

Integer	Remainder
153	
19	1
2	3
0	2

$= (231)_8$

Number-Base Conversions

- **Example 1.3**
  - **Convert $(0.6875)_{10}$ to binary.**
  - **The process is continued until the fraction becomes 0 or until the number of digits has sufficient accuracy.**

	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	$a_{-4} = 1$

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

Number-Base Conversions

- To convert a decimal fraction to a number expressed in base *r*, a similar procedure is used.
- However, multiplication is by *r* instead of 2, and the coefficients found from the integers may range in value from 0 to $r - 1$ instead of 0 and 1.

Number-Base Conversions (p.24)

- **Example1.4**
  - **Convert $(0.513)_{10}$ to octal.**

$$0.513 \times 8 = 4.104$$
$$0.104 \times 8 = 0.832$$
$$0.832 \times 8 = 6.656$$
$$0.656 \times 8 = 5.248$$
$$0.248 \times 8 = 1.984$$
$$0.984 \times 8 = 7.872$$

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

- **From Examples 1.1 and 1.3:** $(41.6875)_{10} = (101001.1011)_2$
- **From Examples 1.2 and 1.4:** $(153.513)_{10} = (231.406517)_8$

1.4 Octal and Hexadecimal Numbers (p.24)

- Numbers with different bases: Table 1.2.

Table 1.2
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

Octal and Hexadecimal Numbers (p.25)

- Conversion from binary to octal can be done by positioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right.

(10	110	001	101	011	·	111	100	000	110)$_2$	= $(26153.7406)_8$
2	6	1	5	3		7	4	0	6	

- Conversion from binary to hexadecimal is similar, except that the binary number is divided into groups of four digits:

(10	1100	0110	1011	·	1111	0010)$_2$	= $(2C6B.F2)_{16}$
2	C	6	B		F	2	

- Conversion from octal or hexadecimal to binary is done by reversing the preceding procedure.

$(673.124)_8$ =	(110	111	011	·	001	010	100)$_2$
	6	7	3		1	2	4

$(306.D)_{16}$ =	(0011	0000	0110	·	1101)$_2$
	3	0	6		D

1.5 Complements (p.26)

- There are two types of complements for each base-*r* system: the radix complement and diminished radix complement.

  → the r's complement and the (r – 1)'s complement.

- Diminished Radix Complement

  Given a number N in base r having n digits, the $(r-1)$'s complement of N is defined as $(r^n - 1) - N$. For decimal numbers, $r = 10$ and $r - 1 = 9$, so the 9's complement of N is $(10^n - 1) - N$.

- Example:

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

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

  For binary numbers, $r = 2$ and $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$.

- Example:

  The 1's complement of 1011000 is 0100111

  The 1's complement of 0101101 is 1010010

Complements (p.27)

Radix Complement

The r's complement of an n-digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and as 0 for $N = 0$. Comparing with the $(r-1)$'s complement, we note that the r's complement is obtained by adding 1 to the $(r-1)$'s complement, since $r^n - N = [(r^n - 1) - N] + 1$.

Example: Base-10

The 10's complement of 012398 is 987602
The 10's complement of 246700 is 753300

Example: Base-2

The 2's complement of 1101100 is 0010100
The 2's complement of 0110111 is 1001001

Complements (p.28)

Subtraction with Complements

- **The subtraction of two *n*-digit unsigned numbers *M* – *N* in base *r* can be done as follows:**

1. Add the minuend M to the r's complement of the subtrahend N. Mathematically, $M + (r^n - N) = M - N + r^n$.
2. If $M \geqq N$, the sum will produce and end carry r^n, which can be discarded; what is left is the result $M - N$.
3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r's complement of $(N - M)$. To obtain the answer in a familiar form, take the r's complement of the sum and place a negative sign in front.

Complements (p.28, 29)

Example 1.5

- Using 10's complement, subtract 72532 – 3250.

	$M =$	72532
10's complement of	$N =$	+ 96750
	Sum =	169282
	Discard end carry 10^5 =	– 100000
	Answer =	69282

Example 1.6

- Using 10's complement, subtract 3250 – 72532.

	M =	03250
10's complement of	N =	+ 27468
	Sum =	30718

There is no end carry.

Therefore, the answer is – (10's complement of 30718) = – 69282.

Complements (p.29)

Example 1.7

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

(a)

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

(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's complement of 1101111) = – 0010001.

Complements (p.30)

- Subtraction of unsigned numbers can also be done by means of the (r – 1)'s complement. Remember that the (r – 1) 's complement is one less then the r's complement.
- Example 1.8
  - Repeat Example 1.7, but this time using 1's complement.

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

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

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

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

There is no end carry, Therefore, the answer is Y – X = – (1's complement of 1101110) = – 0010001.

1.6 Signed Binary Numbers (p.30)

- To represent negative integers, we need a notation for negative values.
- It is customary to represent the sign with a bit placed in the leftmost position of the number since binary digits.
- The convention is to make the sign bit 0 for positive and 1 for negative.
- Example:

Signed-magnitude representation:	10001001
Signed-1's-complement representation:	11110110
Signed-2's-complement representation:	11110111

Four-bit Signed Binary Numbers (p.32)

Table 1.3
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	—	—

Signed Binary Numbers (p.32)

- **Arithmetic addition**
  - **The addition of two numbers in the signed-magnitude system follows the rules of ordinary arithmetic.** 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.** **(e.g., (+25) + (-37) = - (37 – 25) = -12)**
  - **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.**
- **Example:**

$$\begin{array}{r} +\ \ 6 \\ \underline{+13} \\ +19 \end{array} \qquad \begin{array}{r} -6 \\ \underline{+13} \\ +7 \end{array}$$

$$\begin{array}{r} +\ \ 6 \\ \underline{-13} \\ -7 \end{array} \qquad \begin{array}{r} -6 \\ \underline{-13} \\ -19 \end{array}$$

Signed Binary Numbers (p.33)

Arithmetic Subtraction

- **In 2's-complement form:**

1. Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including sign bit).
2. A carry out of sign-bit position is discarded.

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

Example:

(– 6) – (– 13) → (11111010 – 11110011)

→ (11111010 + 00001101)

→ 00000111 (+ 7)

Question

- **How about the addition and subtraction of signed-1's-complement system?**

1.7 Binary Codes (p.34)

BCD Code (Binary-coded decimal)

- A number with k decimal digits will require 4k bits in BCD.
- Decimal 396 is represented in BCD with 12bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.
- The binary combinations 1010 through 1111 are not used and have no meaning in BCD.

Table 1.4
Binary-Coded Decimal (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

Binary Code (p.35, 36)

- **Example:**
  - **Consider decimal 185 and its corresponding value in BCD and binary:**

$$\boxed{(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2}$$

- **BCD addition**

$10 = (1010)_2$

$-10 = (0101 + 1 = 0110)_2$

$-N$

$+(r^n - N)$

$$
\begin{array}{rrrrrr}
4 & 0100 & 4 & 0100 & 8 & 1000 \\
\underline{+5} & \underline{+0101} & \underline{+8} & \underline{+1000} & \underline{+9} & \underline{+1001} \\
9 & 1001 & 12 & 1100 & 17 & 10001
\end{array}
$$

Binary Code (p.37)

- **Example:**
  - **Consider the addition of 184 + 576 = 760 in BCD:**

BCD				
	0001	1000	0100	184
	+ 0101	0111	0110	+576
Binary sum				
Add 6				
BCD sum				760

Binary Codes (p.38)

weighted codes

Other Decimal Codes

Table 1.5
Four Different Binary Codes for the Decimal Digits

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

Binary Codes (p.39, 40)

Gray Code

- The advantage is that only one bit in the code group changes in going from one number to the next
  - » Representation of analog data
    - – Continuous change
  - » Spurious output prevention
    - – 0111 -> 1001 rather than 1000
  - » Low power design

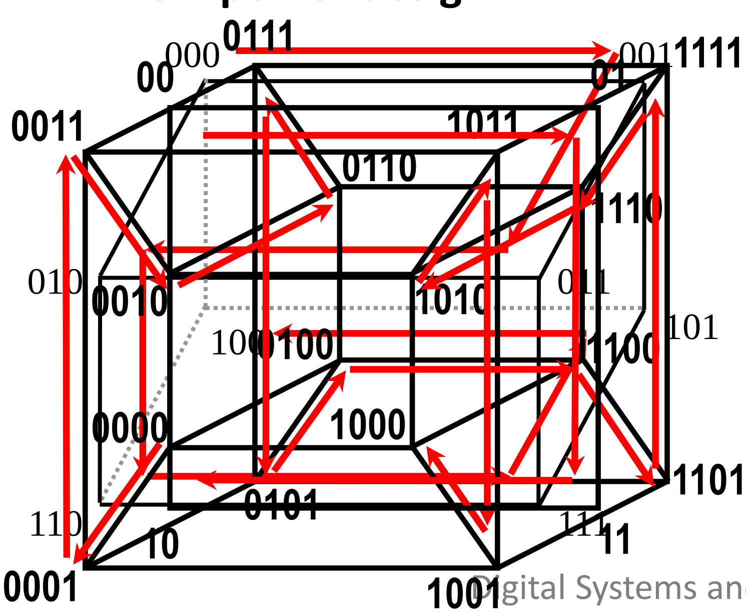


Table 1.6
Gray Code

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

Binary Codes (p.40, 41)

- **American Standard Code for Information Interchange (ASCII) Character Code**

Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**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

Binary Codes (p.41)

ASCII Character Code

Control characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

ASCII Character Codes

- American Standard Code for Information Interchange (Refer to Table 1.7)
- A popular code used to represent information sent as character-based data.
- It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters.
- Some non-printing characters are used for text format (e.g., BS = Backspace, CR = carriage return).
- Other non-printing characters are used for record marking and flow control (e.g., STX and ETX start and end text areas).

ASCII Properties

- ASCII has some interesting properties:
  - Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
  - Upper case A-Z span 41_{16} to $5A_{16}$
  - Lower case a-z span 61_{16} to $7A_{16}$
    - Lower to upper case translation (and vice versa) occurs by flipping bit 6.

Extended ASCII Codes

128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Binary Codes (p.42)

Error-Detecting Code

- To detect errors in data communication and processing, an eighth bit is sometimes added to the ASCII character to indicate its parity.
- A parity bit is an extra bit included with a message to make the total number of 1's either even or odd.
- ACK/NAK for acknowledge/negative acknowledge

Example:

- Consider the following two characters and their even and odd parity:

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

Binary Codes

Error-Detecting Code

- Redundancy (e.g., extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is parity, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single-bit errors and all odd combinations of multiple-bit errors.
- A code word has even parity if the number of 1's in the code word is even.
- A code word has odd parity if the number of 1's in the code word is odd.
- Example:

Message A: 110001001 (even parity)

Message B: 010001001 (odd parity)

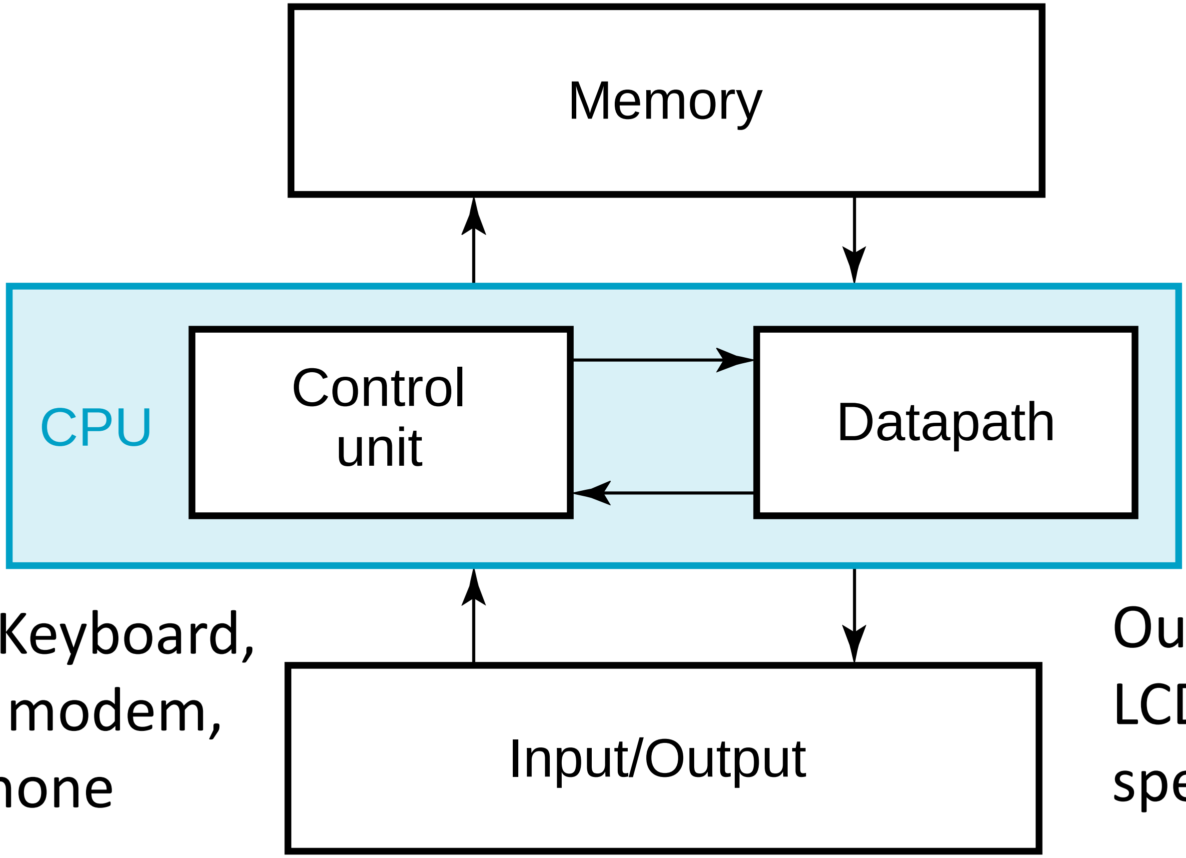
1.8 Binary Storage and Registers (p.43)

- **Registers**
  - **A binary cell is a device that possesses two stable states and is capable of storing one of the two states.**
  - **A register is a group of binary cells. A register with *n* cells can store any discrete quantity of information that contains *n* bits.**

n cells → 2^n possible states

- **A binary cell**
  - **Two stable state**
  - **Store one bit of information**
  - **Examples: flip-flop circuits and capacitor**
- **A register**
  - **A group of binary cells**
  - **AX in x86 CPU**
- **Register Transfer**
  - **A transfer of the information stored in one register to another**
  - **One of the major operations in digital system**
  - **An example in next slides**

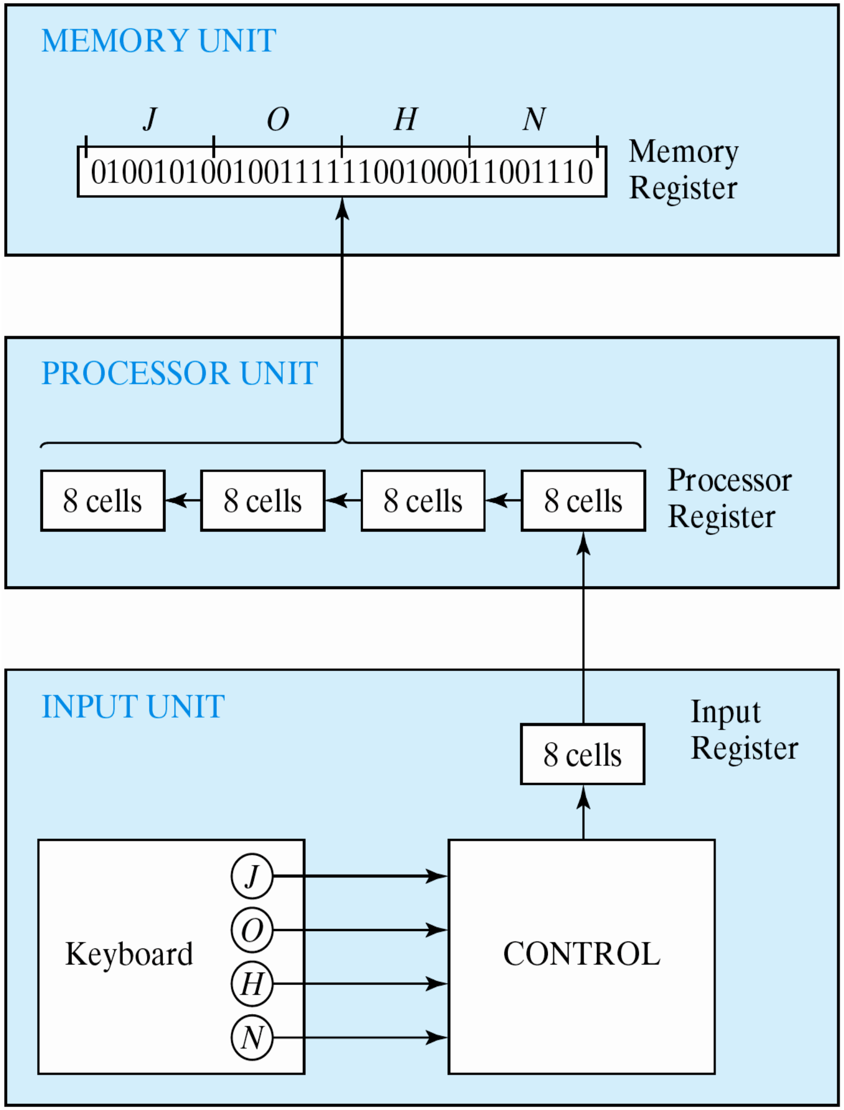
A Digital Computer Example



Inputs: Keyboard, mouse, modem, microphone

Outputs: CRT, LCD, modem, speakers

Transfer of Information (p.44)



Synchronous or Asynchronous?

Figure 1.1 Transfer of information among register

Information Processing (p.45)

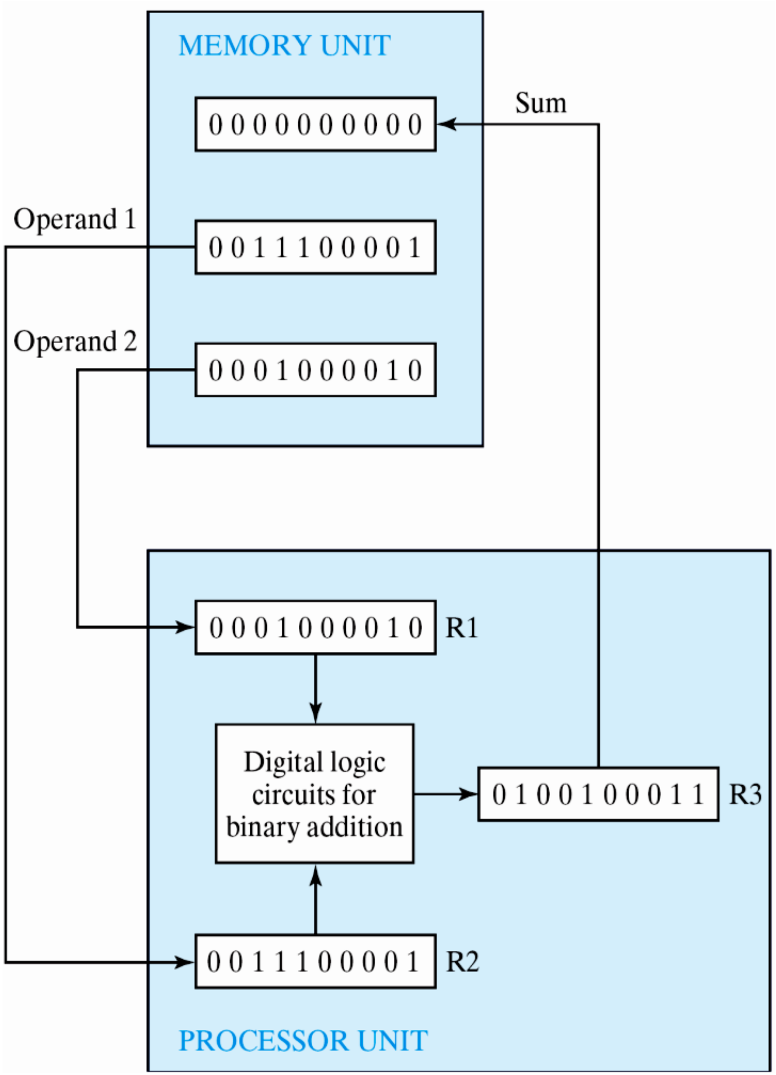


Figure 1.2 Example of binary information processing

- **The other major component of a digital system**
  - **Circuit elements to manipulate individual bits of information**
  - **Load-store machine**

LD	**R1;**
LD	**R2;**
ADD	**R3, R2, R1;**
SD	**R3;**

1.9 Binary Logic (p.46)

Definition of Binary Logic

- **Binary logic consists of binary variables and a set of logical operations.**
- **The variables are designated by letters of the alphabet, such as *A, B, C, x, y, z*, etc, with each variable having two and only two distinct possible values: 1 and 0,**
- **Three basic logical operations: AND, OR, and NOT.**

1. AND: This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "x AND y is equal to z," The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y, and z are binary variables and can be equal either to 1 or 0, and nothing else.)
2. OR: This operation is represented by a plus sign. For example, $x + y = z$ is read "x OR y is equal to z," meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. NOT: This operation is represented by a prime (sometimes by an overbar). For example, $x' = z$ (or $\bar{x} = z$) is read "not x is equal to z," meaning that z is what z is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$, The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

Binary Logic (p.47)

- The truth tables for AND, OR, and NOT are given in Table 1.8.

Table 1.8
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

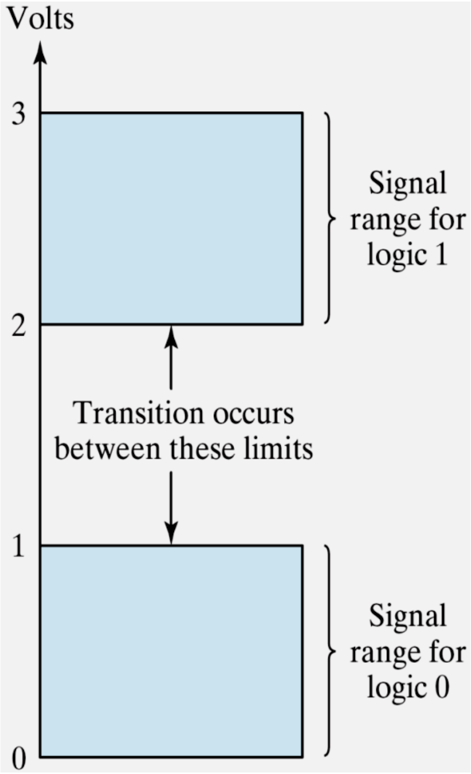
NOT

x	x'
0	1
1	0

Binary Logic (p.47)

Logic gates

- Electronic circuits that operate on one or more input signals to produce an output signal



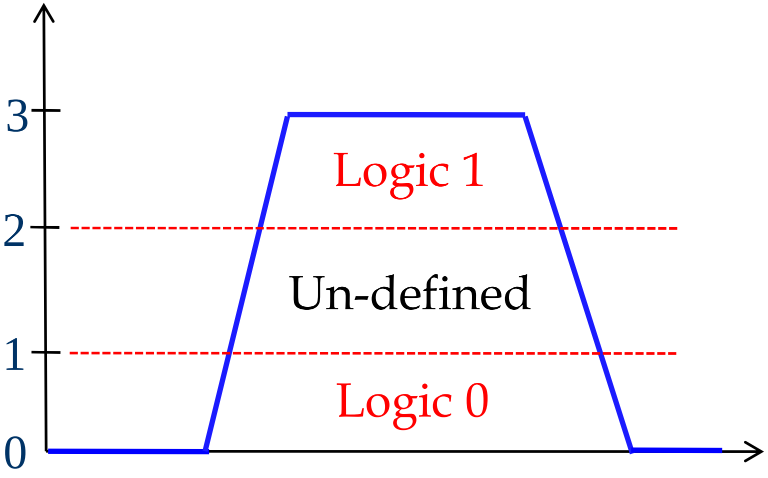
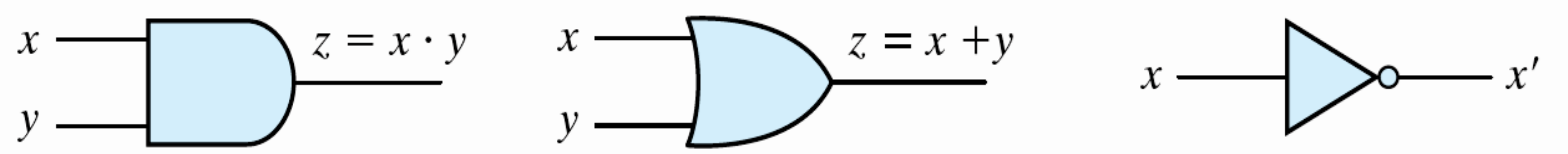


Figure 1.3 Example of binary signals

Binary Logic (p.48)

Logic gates

- **Graphic Symbols and Input-Output Signals for Logic gates:**



(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

Fig. 1.4 Symbols for digital logic circuits

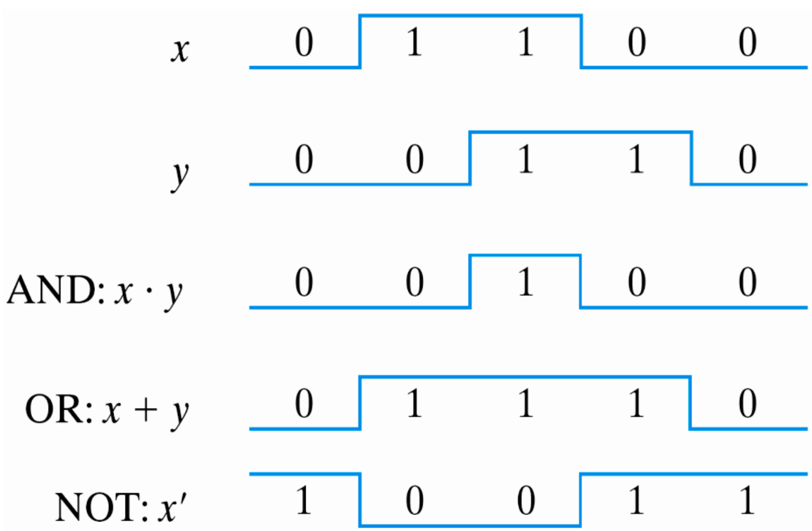
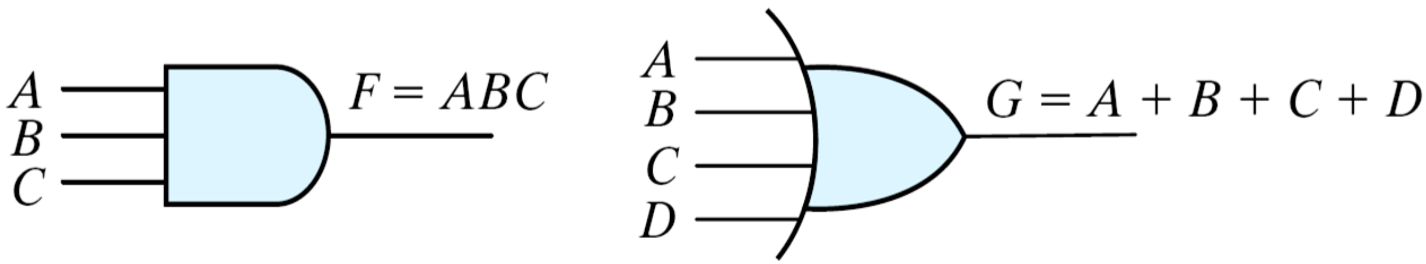


Fig. 1.5 Input-Output signals for gates

Binary Logic (p.49)

Logic gates

- **Graphic Symbols and Input-Output Signals for Logic gates:**



(a) Three-input AND gate (b) Four-input OR gate

Fig. 1.6 Gates with multiple inputs

Summary

- **Digital signal**
  - The physical quantities or signals can assume only discrete values
  - Binary digital signal: 0 and 1
- **Digital system**
  - A system that manipulates discrete elements of information (digital signals)
- **Binary, octal and hexadecimal numbers**
- **Number-base conversions**
- **Complements**
  - r's complement, (r-1)'s complement, and subtraction
- **Signed binary numbers**
  - Signed-magnitude, signed-1's-complement, signed-2's-complement

Summary

- **Binary codes**
  - **BCD, Gray code, ASCII**
- **Binary storage and registers**
  - **Register transfer**
- **Binary logic and logic gates**