Chapter 1
Digital System and Binary Numbers

Outline

- Digital Systems
- Binary Numbers
- Number-Base Conversions
- Octal and Hexadecimal Numbers
- Complements
- Signed Binary Numbers
- Binary Codes
- Binary Storage and Registers
- Binary Logic

Digital Systems

- Why do we need to learn digital systems? Digital systems are everywhere in modern life: (3C and more...)
  - Computers, PDA (personal digital assistant)
  - Consumer Electronics: Digital Camera, MP3 Player, Digital TVs, DVD...
  - Communications: Telephone switching, Internet, Cellular phones
  - Traffic control Systems
  - Medical Treatments: computer tomography...
- Characteristics of digital systems
  - Elements are in discrete levels. Ex: integer, alphabet, porker cards...
  - A digital system is an integration of many digital modules which manipulate the digital elements.
  - Current digital systems are based on “logic design.”
- After learning “Logic Design,” you can understand how a digital system operates in each basic logic module.

Number Systems

- Decimal Number:

  $7392 = 7 \times 1000 + 3 \times 100 + 9 \times 10 + 2 \times 1$

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

  where base (radix) = 10 and (7, 3, 9, 2) are coefficients with respect to different powers of 10.
- Any number can be written with a positive radix r by a string of digits:

  $$a_{n-1}a_{n-2} \ldots a_1a_0 \,.\, a_{-1}a_{-2} \ldots a_{-m+1}a_{-m}$$

  where $0 \le a_i < r$ and . is the *radix point*.
- The string of digits represents the power series:

  $$(\text{Number})_r = \left(\sum_{i=0}^{n-1} a_i \cdot r^i\right) + \left(\sum_{j=-m}^{-1} a_j \cdot r^j\right)$$

  Integer Fraction

Number Systems

		General	Decimal	Binary	Hexadecimal
Radix		r	10	2	16
Coefficients (Digits)		0, 1, 2, … , r - 1	0, 1, 2, … , 9	0, 1	0, 1, 2, … , 9, A, B, C, D, E, F
Power of Radix	0	r^0	$10^0 = 1$	$2^0 = 1$	$16^0 = 1$
	1	r^1	$10^1 = 10$	$2^1 = 2$	$16^1 = 16$
	2	r^2	$10^2 = 100$	$2^2 = 4$	$16^2 = 256$
	3	r^3	$10^3 = 1000$	$2^3 = 8$	$16^3 = 4096$
	4	r^4	$10^4 = 10000$	$2^4 = 16$	$16^4 = 65536$
	⋮	⋮	⋮	⋮	⋮
	-1	r^{-1}	$10^{-1} = 0.1$	$2^{-1} = ½$	16^{-1}
	-2	r^{-2}	$10^{-2} = 0.01$	$2^{-2} = ¼$	16^{-2}
	-3	r^{-3}	$10^{-3} = 0.001$	$2^{-3} = 1/8$	16^{-3}
	-4	r^{-4}	$10^{-4} = 0.0001$	$2^{-4} = 1/16$	16^{-4}

Examples

- Decimal number

  $(7,392)_{10} = 7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

- Base-5 number

  $(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1}$

  $= (511.4)_{10}$

- Binary number

$(110101)_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

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

- Hexadecimal number

  $(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0$

  $= (46,657)_{10}$

- Base-8 number: $(127.4)_8 = (87.5)_{10}$

Binary Special Units

- 2^{10} (1,024) is Kilo, denoted "K", ~ 10^3
- 2^{20} (1,048,576) is Mega, denoted "M", ~ 10^6
- 2^{30} (1,073,741,824) is Giga, denoted "G", ~ 10^9
- 2^{40} (1,099,511,627,776) is Tera, denoted "T", ~ 10^{12}

- A 512 M-byte memory is actually a 536,870,912 (512 x 1048576) byte memory.
- Hard drive companies use "real" bytes as their product capacity.
  - 120 G-byte HD ⇒ 111.75 G-bytes shown on your computer.

Arithmetic Operations

Addition:

Augend: 101101
Addend: +100111
Sum: 1010100

Subtraction:

Minuend: 101101
Subtrahend: - 100111
Difference: 000110

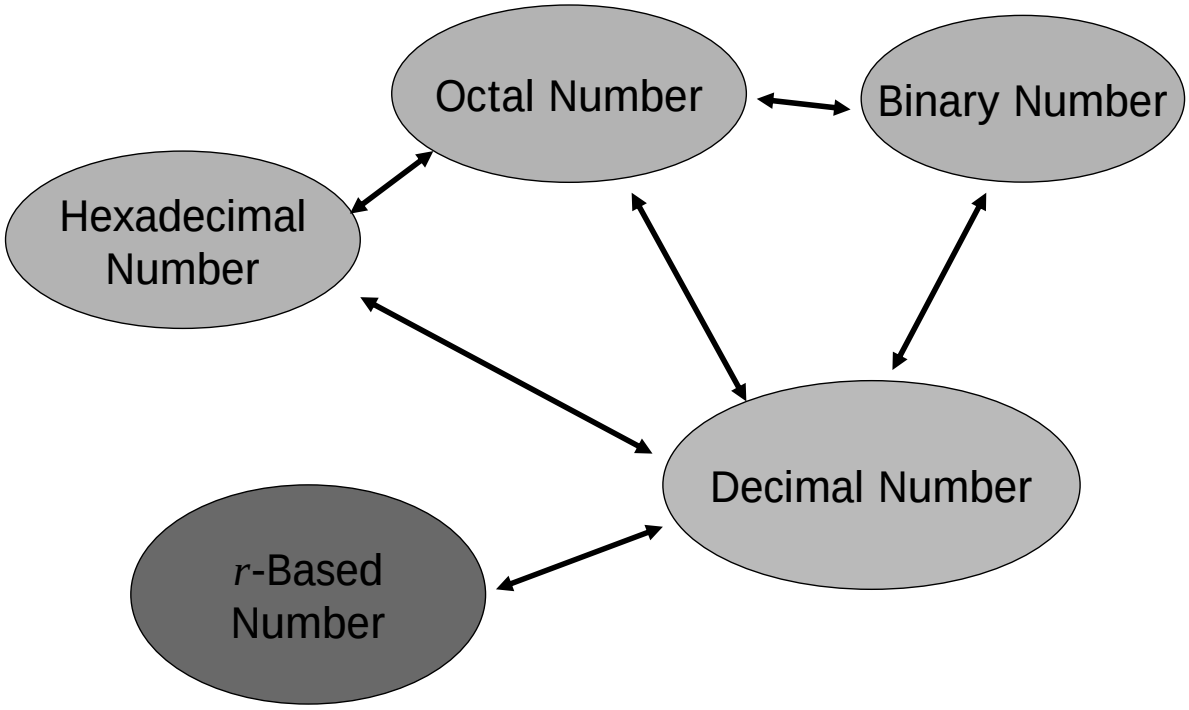
Multiplication:

Multiplicand: 1011
Multiplier: x 101

1011
0000
1011

Product: 110111

Number-Base Conversions



Conversion between Decimal and *r*-Based Number (Integer)

- r-based integer to decimal integer

$$(a_n a_{n-1} \ldots a_1 a_0)_r = (a_n r^n + a_{n-1} r^{n-1} + \ldots + a_1 r^1 + a_0 r^0)_{10}$$

- Decimal integer to r-based integer
  - By dividing the number and all successive integer quotients by r and collecting the remainders as its coefficients.

by using division process on $(N)_{10}$:

$$\frac{N}{r} = a_n r^{n-1} + a_{n-1} r^{n-2} + \cdots + a_2 r^1 + a_1 + \frac{a_0}{r} = Q_1 + \frac{a_0}{r} \quad \text{Residual } a_0$$

$$\frac{Q_1}{r} = a_n r^{n-2} + a_{n-1} r^{n-3} + \cdots + a_3 r^1 + a_2 + \frac{a_1}{r} = Q_2 + \frac{a_1}{r} \quad \text{Residual } a_1$$

$$\frac{Q_2}{r} = a_n r^{n-3} + a_{n-1} r^{n-4} + \cdots + a_4 r^1 + a_3 + \frac{a_2}{r} = Q_3 + \frac{a_2}{r} \quad \text{Residual } a_2$$

.....

$$N_{10} = (a_n a_{n-1} \ldots a_1 a_0)_r$$

Example

- From $(41)_{10}$ to $(a_n a_{n-1} \ldots a_1 a_0)_2$:

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

$(41)_{10} = (a_n a_{n-1} \ldots a_1 a_0)_2 = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = (101001)_2$

Conversion between Decimal and r-Based Number (Fraction)

- r-based fraction to decimal number

$$(0.a_{-1}a_{-2}\ldots a_{-m+1}a_{-m})_r = (a_{-1}r^{-1} + a_{-2}r^{-2} + \ldots + a_{-m+1}r^{-m+1} + a_{-m}r^{-m})_{10}$$

- Decimal fraction to r-based number
  - By multiplying the number and all successive fractions by r and accumulating the integers.

$$N \times r = a_{-1} + a_{-2}r^{-1} + \cdots + a_{-m+1}r^{-m+2} + a_{-m}r^{-m+1} = a_{-1} + M_1 \quad \text{Integer } a_{-1}$$

$$M_1 \times r = a_{-2} + a_{-3}r^{-1} + \cdots + a_{-m+1}r^{-m+3} + a_{-m}r^{-m+2} = a_{-2} + M_2 \quad \text{Integer } a_{-2}$$

$$M_2 \times r = a_{-3} + a_{-4}r^{-1} + \cdots + a_{-m+1}r^{-m+4} + a_{-m}r^{-m+3} = a_{-3} + M_3 \quad \text{Integer } a_{-3}$$

$N_{10} = (0.a_{-1}a_{-2}\ldots a_{-m+1}a_{-m}\ldots)_r$

May have infinite integers

Example

- From $(0.6875)_{10}$ to $(0.a_{-1}a_{-2}...a_{-m+1}a_{-m})_2$

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

stop

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

Example

- From $(0.513)_{10}$ to $(0.a_{-1}a_{-2}...a_{-m+1}a_{-m}\ldots)_8$

		Integer		Fraction	Coefficient
0.513 x 8	=	4	+	0.104	$a_{-1} = 4$
0.104 x 8	=	0	+	0.832	$a_{-2} = 0$
0.832 x 8	=	6	+	0.656	$a_{-3} = 6$
0.656 x 8	=	5	+	0.248	$a_{-4} = 5$
0.248 x 8	=	1	+	0.984	$a_{-5} = 1$
0.984 x 8	=	7	+	0.872	$a_{-6} = 7$

⋮

To seven significant digits:

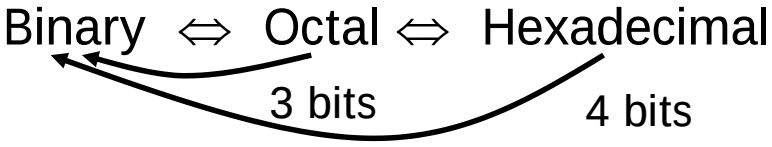
$$(0.513)_{10} = (0.a_{-1}a_{-2}...a_{-m+1}a_{-m}\ldots)_8 = (0.406517...)_8$$

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

- It's important in digital computer to convert



- $(\underline{101}\,\underline{100}\,\underline{011}\,\underline{010}\,\underline{11}.\underline{111}\,\underline{000}\,\underline{011}\,\underline{0})_2$

  $= (2\ \ 6\ \ 1\ \ 5\ \ 3\ .\ 7\ \ 4\ \ 0\ \ 6)_8$

  $= (2\ \ C\ \ \ 6\ \ \ B\ .\ F\ \ \ 0\ \ \ 6)_{16}$

- $(673.124)_8 = (110\ 111\ 011.001\ 010\ 100)_2$
  $6\ \ \ 7\ \ \ 3\ \ \ 1\ \ \ 2\ \ \ 4$
- $(306.D)_{16} = (0011\ 0000\ 0110.1101)_2$
  $3\ \ \ 0\ \ \ 6\ \ \ D$

Complements

- Complements are used in digital computers to simplify the subtraction operations and for logical manipulations, so that the corresponding circuits can be simpler and less expensive.
- There are two types of complements for each base-r system:
  - r's (radix) complement
  - (r–1)'s (diminished radix) complement
  - Examples:

$r = 2$ { 2's complement; 1's complement } $r = 10$ { 10's complement; 9'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

$$(r^n - 1) - N$$

{
base 10: $9999....9 - N$
base 2: $1111....1 - N$
base 8: $7777....7 - N$
base 16: FFFF....F $- N$ = (15, 15, 15, ..., 15) $- N$
}

- Examples:

The 9's complement of 546700 is 999999 – 546700 = 453299
The 9's complement of 012398 is 999999 – 012398 = 987601
The 1's complement of 1011000 is 1111111 – 1011000 = 0100111
The 1's complement of 0101101 is 1111111 – 0101101 = 1010010

Radix Complement

- The r's complement of an n-digit number N in base r is defined as:

$$\begin{cases} r^n - N & \text{for } N \neq 0 \\ 0 & \text{for } N = 0 \end{cases}$$

- The r's complement = (r–1)'s complement + 1
- Examples:

The 10's complement of 546700 is 1000000 – 546700 = 453300
The 10's complement of 012398 is 1000000 – 012398 = 987602
The 2's complement of 1011000 is 10000000 – 1011000 = 0101000
The 2's complement of 0101101 is 10000000 – 0101101 = 1010011

Complements

- How if the number contains a radix point?
  - If number N has a radix point, the point should be removed temporarily to do r's or (r–1)'s complement and then restoring the radix point to the complemented number in the same position.
  - Example: $N = (213.61)_{10} \Rightarrow N' = 21361$

    Take $\begin{cases} \text{10's complement of } N\text{': } 78639 \\ \text{9's complement of } N\text{': } 78638 \end{cases} \Rightarrow \begin{cases} 786.39 \\ 786.38 \end{cases}$

- Note: the complement of the complement is equal to the original number.

Subtraction with Complements

- In elementary school math, we use the "borrow" concept to do subtraction: Not efficient in digital hardware implementations!! $\Rightarrow$ use complements!!
- The subtraction of two n-digit unsigned numbers $M - N$ in base r:
  1. Add the minuend M, to the r's complement of the subtrahend N. That is, $M + (r^n - N) = M - N + r^n$.
  2. If $M \geq N$, the sum will produce an end carry r^n, which can be discarded to get the result $M - N$.
  3. If $M < N$, the sum will be equal to $r^n - (N - M)$, which is the r's complement of $N - M$.

  $\Rightarrow$ To obtain the answer in a familiar form, take the r's complement of the sum and place a "negative" sign in front.

Examples

- Using 10's complement to compute 72532 – 3250

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

- Using 10's complement to compute 3250 – 72532

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

$\because M < N$, the answer is - (10's complement of 30718) = - 69282.

Examples

- X= 1010100, Y = 1000011 to calculate (a) $X - Y$ and (b) $Y - X$ using 2's complement

(a) $X - Y$

X =	1010100
2's complement of Y =	+ 0111101
Sum =	10010001
Discard end carry 2^7 =	–10000000
Answer =	0010001

(b) $Y - X$

Y =	1000011
2's complement of X =	+ 0101100
Sum =	1101111

$\because$ $Y < X$, the answer is - (2's complement of 1101111)
= - 0010001

Subtraction using (r-1)'s Complements

- When performing (r-1)'s complement subtraction, if there's an end carry, the result will be one less than the correct answer. Removing the end carry and adding 1 to the sum is referred to as "end-around carry."
- Examples: X = 1010100, Y = 1000011

(a) $X - Y$

X =	1010100
1's comp't of Y =	+ 0111100
Sum =	10010000
End-around carry =	+ 1
Answer =	0010001

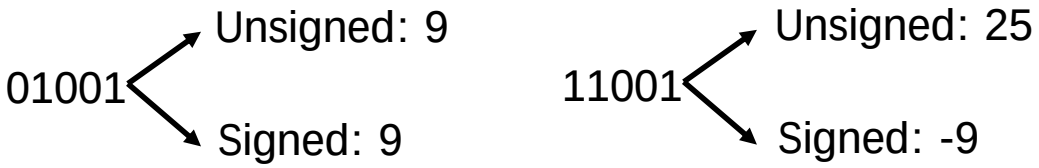
(b) $Y - X$

Y =	1000011
1's comp't of X =	+ 0101011
Sum =	1101110

There's no end carry, the answer is - (1's complement of 1101110) = - 0010001

Signed Binary Numbers

- Traditionally, positive number has no sign or a "plus, +" sign and negative number has a "minus, –" sign before the number.
- In computer the sign is represented by the leftmost bit: "0" is for positive and "1" is for negative.
- In computer, the difference between signed and unsigned numbers is important.
  - Example:



Signed Binary Numbers

- Signed-magnitude representation
  - Negative number is obtained by changing the sign bit in the leftmost position from 0 to 1
    - +9 000001001
    - -9 100001001
- Signed-complement representation
  - In 2's complement, negative number is presented by taking the 2's complement of the positive number
  - 2's complement -- 1's complement

    +9 000001001 +9 000001001

    -9 111110111 -9 111110110
- What is decimal value of 101011011 in (a) signed magnitude, (b) 2's complement, and (c) 1's complement representations?

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	-	-

Arithmetic Addition

- Signed-magnitude addition:
  - If the signs are the same, we add the two magnitudes and give the sum a common sign.
  - If the signs are different, subtract larger magnitude by smaller magnitude. Then give the difference the sign of the larger number to get the resultant sign.
- Signed-complement addition:
  - We do not need sign comparison and magnitude comparison.
  - Only direct addition is required.

Example (2's Complement Addition)

+ 6	00000110
+ 13	00001101
+ 19	00010011

sign

– 6	11111010
+ 13	00001101
+ 7	100000111

overflow
(discard for 2's comp't)

How about 1's comp't?

+ 6	00000110
– 13	11110011
– 7	11111001

– 6	11111010
– 13	11110011
– 19	111101101

overflow

Arithmetic Subtraction

- Signed-magnitude subtraction:
  - Change the sign of subtrahend first.
  - Do the same step as signed-magnitude addition

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

- Signed-complement subtraction:
  - Reverse subtrahend by taking signed 2's complement
  - Do the same step as signed-complement addition (only direct addition)

Binary Code

- Digital systems use signals that have two distinct values and circuit elements that have two stable states (bi-stable states).
- An n-bit binary code is a group of n bits that assumes up to 2^n distinct combinations of 1's and 0's, with each combination representing one element state.
- Binary Codes:
  - Binary Coded Decimal (BCD) Code
  - BCD-2421 Code
  - Excess-3 Code
  - 84-2-1 Code
  - Gray Code
  - ASCII Character Code
  - Error Detecting Code

Binary Coded Decimal (BCD)

- BCD represents the decimal system using binary number:
  - Using 4-bit to represent 0-9 in the decimal system.
  - A-F in the binary numbers are discarded.
  - Example:

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

    $= (10111001)_2$
- Important facts:
  - A BCD number needs more bits than its equivalent binary value. (12 bits v.s. 8 bits in the example)
  - BCD numbers are decimal numbers, not binary numbers.

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

BCD Addition

- Example: 184 + 576 = 760 in BCD

BCD Carry	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary Sum	0111	10000	1010	
Add 6		0110	0110	
BCD Sum	0111	0110	0000	760

- The plus 6 and minus 10 are equivalent in carry-propagation operation.

Decimal Arithmetic

- Signed decimal number in BCD is similar to the signed binary number.
  - First digit: 0 (0000) for "+" and 9 (1001) for "–"
- 10's complement, is obtained by taking the 9's complement first and adding 1 to the least significant digit, which is the one most often used.
- Example: (+375) + (-240) = +135

Decimal	
0	375
+9	760
0	135

BCD			
0000	0011	0111	0101
+1001	0111	0110	0000
1010	1011	1101	0101
0110	0110	0110	
10000	10001	10011	0101

Other Decimal Codes

Decimal Digit	BCD 8421 (weighted)	2421 (weighted)	Excess-3 (self-comp't)	8, 4, -2, -1 (weighted)
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010 (1000)	0101	0110
3	0011	0011 (1001)	0110	0101
4	0100	0100 (1010)	0111	0100
5	0101	1011 (0101)	1000	1011
6	0110	1100 (0110)	1001	1010
7	0111	1101 (0111)	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combinations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Gray Code

- Many physical systems' output data are continuous quantities.
- Continuous/Analog information is converted to digital form by ADC.
- The advantage of Gray code is that only one bit in the code group changes when going from one number to the next.
- Gray code is used in applications in which the normal sequence of binary numbers may produce an error or ambiguity during the transition.

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

ASCII Character Code

- Digital computers require to handle not only numbers, but also characters and symbols.
- To contain all uppercase and lowercase letters, numerals and special characters, 7-bit binary codes are required.
- ASCII (American Standard Code for Information Interchange) code uses 7 bits, b_1 to b_7, to code 128 characters:
  - 94 graphic characters: 26 uppercase, 26 lowercase, 10 numerals, and 32 special characters.
  - 34 control characters: backspace (BS), file separator (FS), start of text (STX), delete (DEL), etc.
- Most computers use "byte" (containing 8 bits). A byte contains an ASCII code, with one more bit for other purposes.

ASCII 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

ASCII Code's Control Characters

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

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

	Even Parity	Odd Parity
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

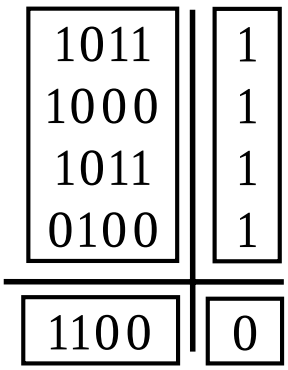
Error-Detecting Code

- 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 some 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.
- Error detecting/correcting code is important in modern communication systems.

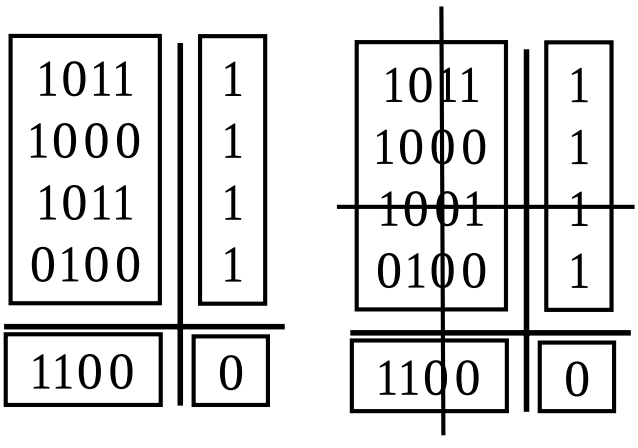
Example of Error Detecting/Correcting Code

Raw Data

1011100010110100



Data Corrected by FEC



1011100010110100 1111 1100 0

Data after FEC

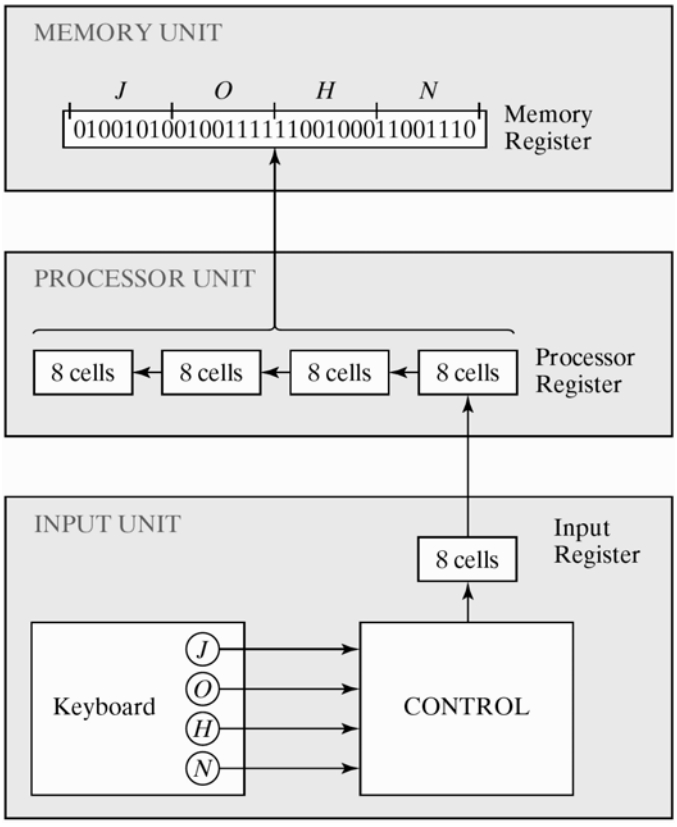
Binary Storage and Registers

- 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-bit register, containing n cells, has 2^n states.
  - Examples: 16-bit register
    - Binary: 1100 0011 1100 1001
    - Decimal number: 50121
    - ASCII Code: C I characters
    - Excess-3 Code: 9096
    - BCD: Meaningless because of 1100

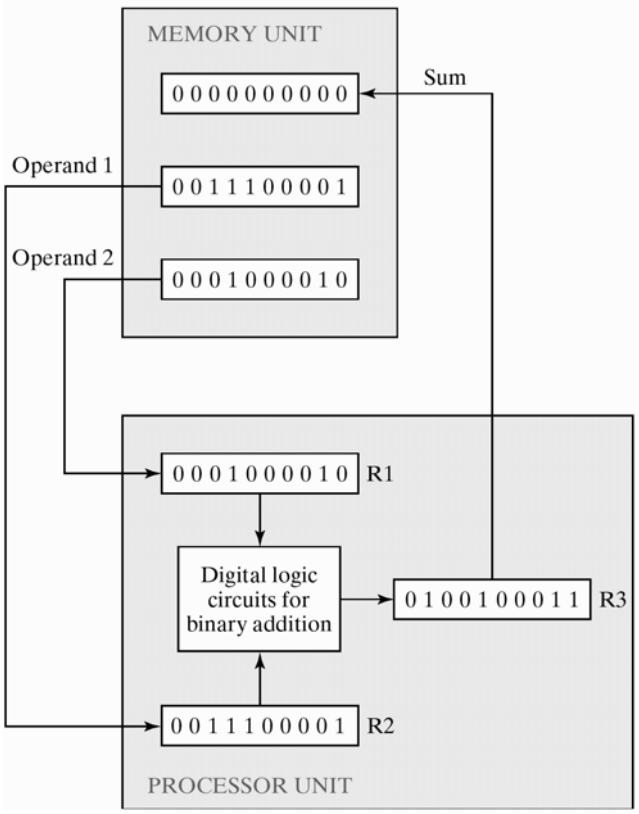
In a register, the same bit configuration may be interpreted differently for different types of data format.

Register Transfer

- Register Transfer
  - A transfer of the information stored in one register to another
  - One of the major operations in digital system
  - Example:



Binary Information Processing



Binary Logic

- Binary logic consists of binary variables and a set of logical operations.
  - Each variable has two and only two distinct possible values: 1 and 0.
  - There are three basic logical operations: AND, OR, and NOT.
- Definitions of binary logic/operation:
  - AND: $x \cdot y = z$, $z = 1$ if and only if $x = 1$ and $y = 1$;
    $z = 0$ otherwize
  - OR: $x + y = z$, $z = 1$ if and only if $x = 1$ and/or $y = 1$;
    $z = 0$ otherwize
  - NOT: x' = z, $z = 1$ if $x = 0$
    $z = 0$ if $x = 1$

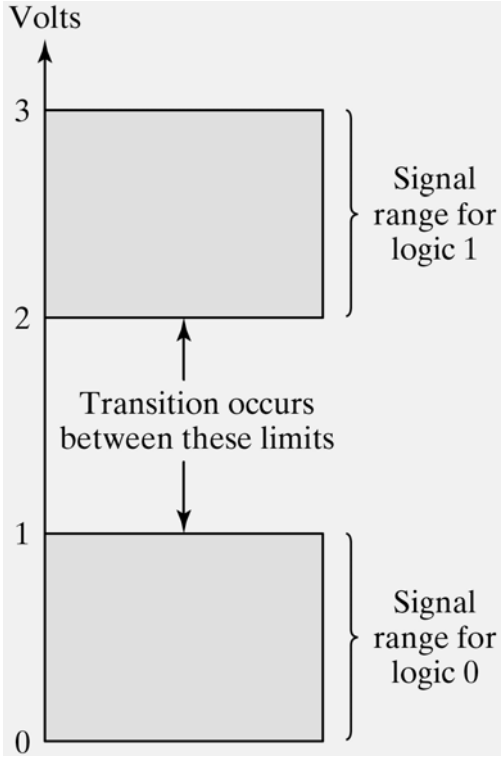
Truth Table

Table 1.8
Truth Tables of Logical Operations

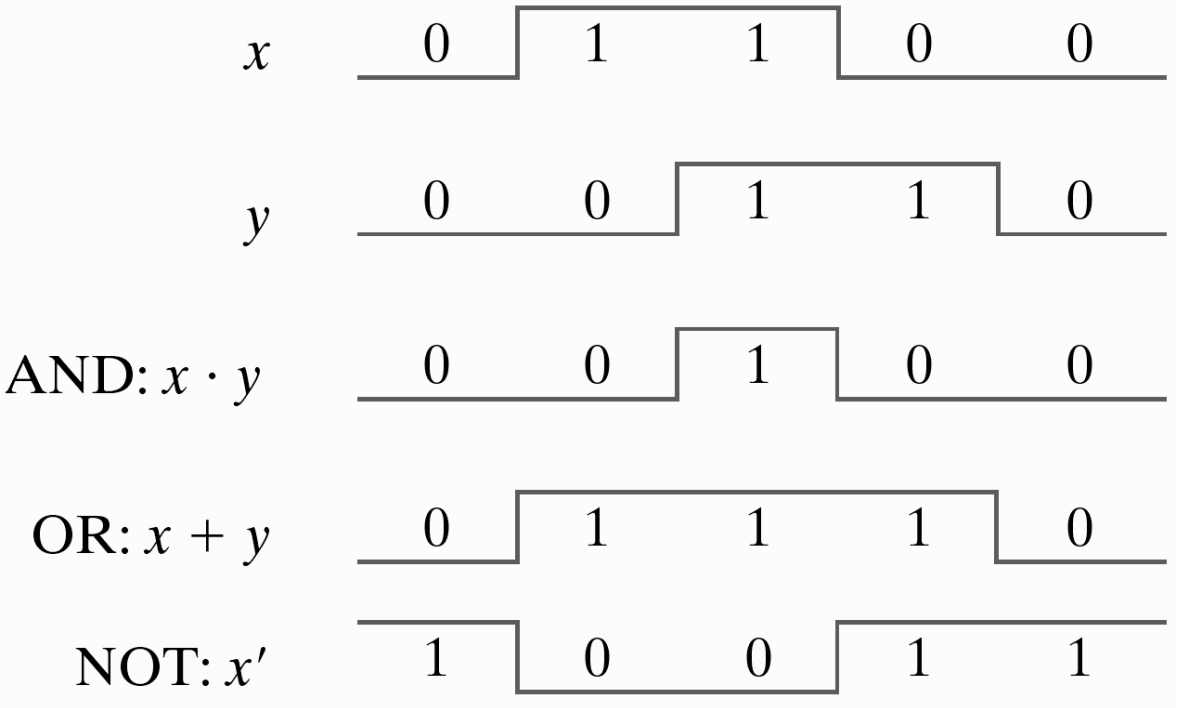
AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Binary Signal Representation

- Logic gates are electronic circuits that operate at two separate voltage levels representing binary variable to logic-1 and logic-0.
- The allowable range defines legal voltage level for logic-1 or logic-0.
- The intermediate range between allowed ranges is crossed only during a state transition.

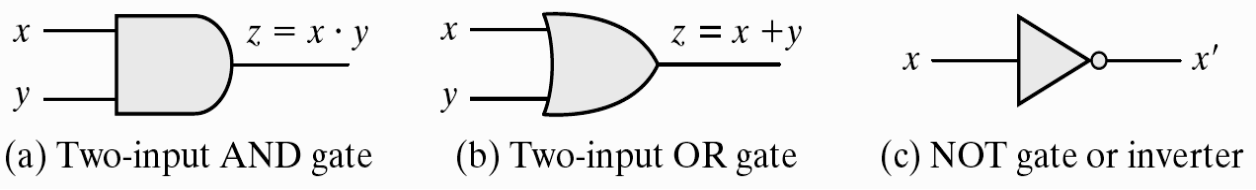


Example of Input-Output Signals for Three Gates



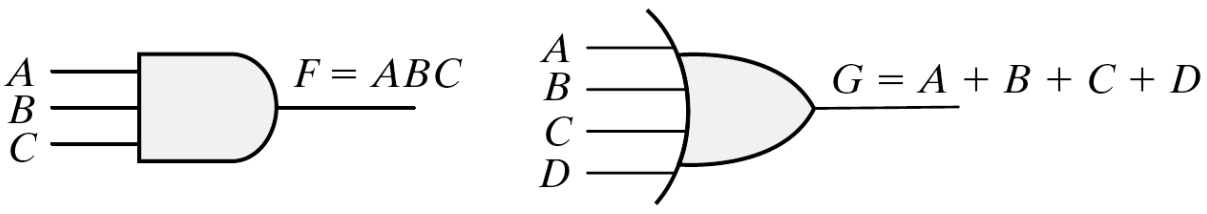
Graphic Symbols for Digital Logic Circuit

- Graphic symbols of AND, OR and Not gates:



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

- Gates with multiple inputs:



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