Logic Design

Dr. Yosry A. Azzam

Binary systems

Chapter 1

- Agenda

Binary Systems :
Binary Numbers,
Binary Codes,
Binary Logic

- ASCII Code (American Standard Code for Information Interchange)
- Boolean Algebra (Basic Theorems, Property of Boolean Algebra, Boolean Functions)
- Logic Gates

- Readings
  - Mano: Ch 1 & 2 (until 2-4)
- Objectives
  - Understand Bit & Byte as the foundation of data representation
  - Understand the Binary System, it’s operations, conversions and negative number representation
  - Understand the Logic Gates & Binary Logics, which they based on

Data Representation

- The complex computer system is built on a

2-states system (on/off) : The Binary System.

- Binary system is a 2 base numbering system: 0 and 1

- Each 0 and 1 is called "BIT" (BInary digiT)

Bits & Bytes

- Bit (0 or 1)
  - Off/On
    for positive logic
  - On/Off
    for negative logic

Dec (Bin)

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

Bits & Bytes (cont'd)

- Byte : a group of 8 bits, represent :
  - ASCII characters (1 byte is 1 character)

  Refer to ASCII Table p : 23
  - Unicode
  - There are other format of data representation discussed later in the course.

- A (0100 0001)
- B (0100 0010)
- ...
- Z (0101 1010)
- ...
- 0 (0011 0000)
- 1 (0011 0001)
- ...
- 9 (0011 1001)

Binary Systems

- Binary Numbers
- Binary Codes
- Binary Logic

Binary and Decimal Numbers

- Binary
  - $1010 = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0$
  - 0, 1, 10, 11 …
  - Called "Base-2"
- Decimal
  - $7392 = 7x10^3 + 3x10^2 + 9x10^1 + 2x10^0$
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 …
  - Called "Base-10"
- Octal
  - Based-8 : (0, 1, 2, 3, 4, 5, 6, 7)
- Hexadecimal
  - Based-16 : (0, 1, 2, 3, 4, 5, 6, 7, 8 ,9 ,A ,B ,C ,D ,E ,F)

Reading : Mano. Chapter 1

Binary Systems and Number Base Conversion:

Decimal Numbers (Base-10):

0,1,2,3,4,5,6,7,8, and 9 Ten Digits

Binary Numbers (Base-2): 0 and 1

Two Digits

Octal Numbers (Base-8): 0..7

Eight Digits

Hexadecimal No. (Base-16):

0,…,9,A,B,C,D,E,F 16 Digits

and so on.

1.3 Number Base Conversion

(1): $(7392)_{10} = 7x10^3 + 3x10^2 + 9x10^1 + 2x10^0$

(2): $(1010.011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3} = (10.375)_{10}$

(3): $(4021.2)_5 = 4x5^3 + 0x5^2 + 2x5^1 + 1x5^0 + 2x5^{-1} = (511.4)_{10}$

(4): Convert decimal 41 to binary, i.e., $(41)_{10} = (¿?)_2$

Solution:

Divide by 2	Integer quotent	Remainder	Coefficient	
41/2	=20	+1	1	LSB
20/2	=10	+0	0	
10/2	= 5	+0	0	
5/2	= 2	+1	1	
2/2	= 1	+0	0	
1/2	= 0	+1	1	MSB

OR= 101001

Divide by 2	Remainder	
41		
20	1	LSB
10	0	
5	0	
2	1	
1	0	MSB
0	1	→ Answer=101001

(5): Convert $(0.6875)_{10}$ to binary.

Multiply by 2	Integer quotient	fraction	Coefficient	
0.6875x2	=1	0.3750	1	MSB
0.3750x2	=0	0.7500	0	
0.7500x2	=1	0.5000	1	
0.5000x2	=1	0.0000	1	LSB

- Answer: $(0.6875)_{10} = (0.1011)_2$

(6): Convert decimal 153 to octal, i.e., $(153)_{10} = (\ ¿?)_8$

Solution:

Divide by 8	Remainder	
153		
19	1	LSB
2	3	
0	2	MSB
		→ Answer=231

➔ $(153)_{10} = (\ 231)_8$

(7): Convert $(0.513)_{10}$ to octal, to seven significant figures

Multiply by 8	Integer quotient	fraction	Coefficient	
0.513x8	=4	0.104	4	MSB
0.104x8	=0	0.832	0	
0.832x8	=6	0.656	6	
0.656x8	=5	0.248	5	
0.248x8	=1	0.984	1	
0.984x8	7	0.872	7	LSB

Answer: $(0.513)_{10} = (0.406517.....)_8$

(8): Convert decimal 153.513 to octal,

since we know that $(153)_{10} = (231)_8$

and $(0.513)_{10} = (0.406517)_8$

Then $(\underline{153}.\underline{513})_{10} = (\underline{231}.\underline{406517})_8$

1.4 Octal and Hexadecimal Numbers

Since $2^3=8$ and $2^4=16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds to four binary digits.

Examples:

convert the binary 10110001101011.111100000110 to octal.

Answer: $(10\ 110\ 001\ 101\ 011\ .\ 111\ 100\ 000\ 110)_2$

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

convert the binary 10110001101011.111100000110 to Hexadecimal

Answer: $(10\ 1100\ 0110\ 1011\ .\ 1111\ 0000\ 0110)_2$

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

Binary Numbers : Conversions 2

- Octal ($2^3 = 8$)
  - $(10110001101011.11110000011)_2$

10 110 001 101 011.111 100 000 110
2 6 1 5 3 . 7 4 0 6

  - $(26153.7406)_8$
- Hexadecimal ($2^4 = 16$)
  - $(10110001101011.11110000011)_2$

10 1100 0110 1011.1111 0000 0110
2 C 6 B . F 0 6

  - $(2C6B.F06)_{16}$

Binary Numbers : Operations

- Summation

```
 101101
+100111
----------
1010100
```

- Subtraction

```
 101101
-100111
----------
 000110
```

- Multiplication

```
      1011
       101
----------
      1011
    0000 .
   1011 . .
----------
    110111
```

Two's complement notation systems

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Diminished Radix Complements

- Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.
- 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 $(10^n - 1) - N = 999..99 - N$
- For binary numbers, $r=2$ and $r-1=1$ so,
- The 1's complement of N is $(2^n - 1) - N = 111\ldots111 - N$

Radix Complements

- The radix complement of an n-digit number N in base r is defined as r^n-N for N$\neq$0 and 0 for N=0. *i.e.* the radix complement= diminished radix complement +1

Complements

- The complement of 012398 is
  - 9′s complement (diminished radix complement)
    - $(999999)_{10}-(012398)_{10} = (987601)_{10}$
  - 10′s complement (radix complement)
    - $(987602)_{10} = (987601)_{10} + 1=(987602)_{10}$ or:
    - $(1000000)_{10}-(012398)_{10}=(987602)_{10}$
- The complement of 1101100 is
  - 1′s complement (diminished radix complement)
  - $(1111111)_2- (1101100)_2=$ **0010011**
  - 2′s complement (radix complement)
  - $(10000000)_2- (1101100)_2=$ **0010100**

Complements (cont'd.)

- The *(r-1)'s* complement of octal or hexadecimal numbers is obtained by subtracting each digit from 7 or F (decimal 15) respectively

Examples:

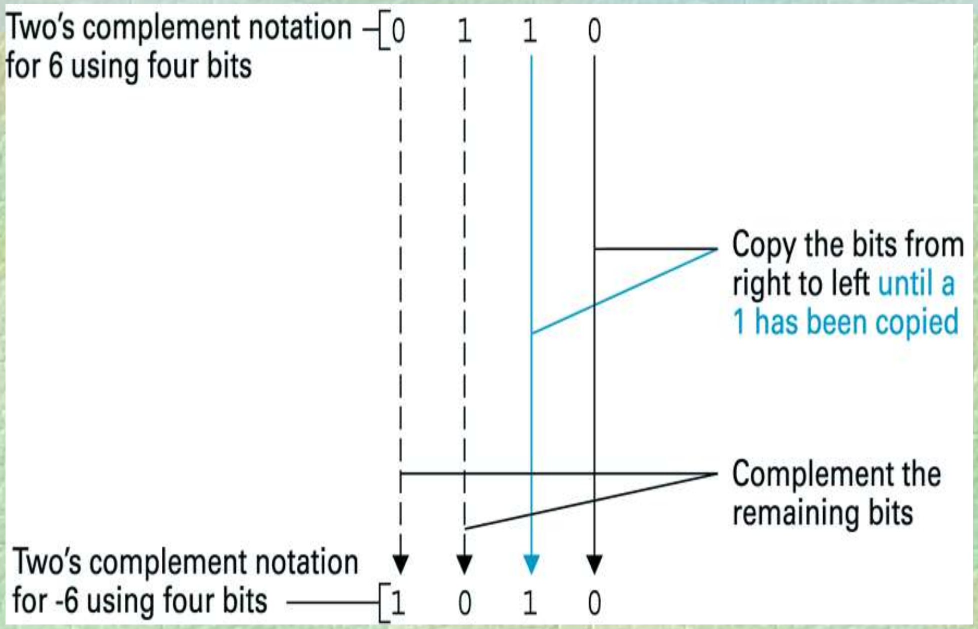
(1): 10's complement of $(52520)_{10} = 10^5 - 52520 = 47480$

(2): 10's complement of $(246700)_{10}$ is 753300

(3): 10's complement of $(0.3267)_{10} = 1.0 - 0.3267 = 0.6733$

(4): 2's complement of $(101100)_2 = (2^6)_{10} - (101100)_2 = (1000000)_2 - (101100)_2$
$= (010100)_2$

(5): 2's complement of $(0.0110)_2 = (2^0)_{10} - (0.0110)_2 = (1 - 0.0110)_2 = (0.1010)_2$.



Subtraction with Complement

- 10’s complement
  - Subtract 72532 – 3250

72532
10’s complement: +96750

Sum: 169282
Remove end carry: -100000

Answer: 69282

- 2’s complement
  - Subtract 1010100 - 1000011

1010100
2’s complement: +0111101

Sum: 10010001
Remove end carry: -10000000

Answer: 0010001

Signed Binary Numbers 1

- Due to hardware limitation of computers, we need to represent the negative values using *bits*. Instead of a "+" and "-" signs.
- Conventions:
  - **0** for positive
  - **1** for negative

Signed Binary Numbers 2

- $(9)_{10} = (0000\ 1001)_2$
- 1. Signed magnitude (used in ordinary arithmetic):
  - $(-9)_{10} = (1000\ 1001)_2$
  - Changing the first ‟sign bit” to negative
- 2. Signed 1′s complement:
  - $(-9)_{10} = (1111\ 0110)_2$
  - Complementing all bits including sign bit
- 3. Signed 2′s complement:
  - $(-9)_{10} = (1111\ 0111)_2$
  - Taking the 2′s complement of the positive number

Signed Binary Numbers 3

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	—	—

Arithmetic Addition and Subtraction

Problem in base ten		Problem in two's complement		Answer in base ten
3 + 2	→	0011 + 0010 = 0101	→	5
−3 + −2	→	1101 + 1110 = 1011	→	−5
7 + −5	→	0111 + 1011 = 0010	→	2

+6	00000110
+13	00001101
+19	00010011

- 6	11111010
+13	00001101
+7	00000111

+ 6	00000110
- 13	11110011
- 7	11111001

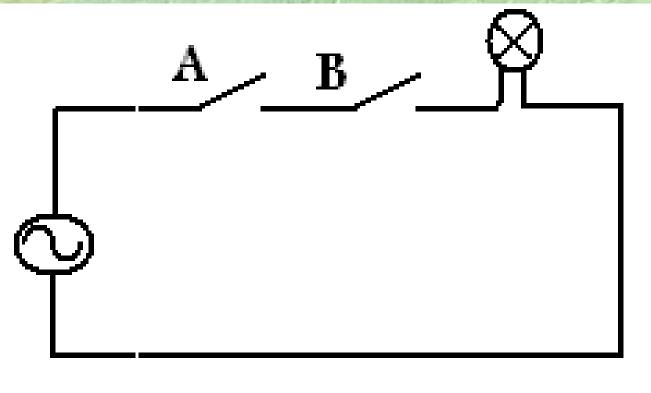
- 6	11111010
- 13	11110011
- 19	11101101

Binary Logic

- Binary Logic: Consists of Binary Variables and Logical Operations
- Basic Logical Operations:
  - AND
  - OR
  - NOT
- Truth tables: Table of all possible combinations of variables to show relation between values

Logical Operation: AND

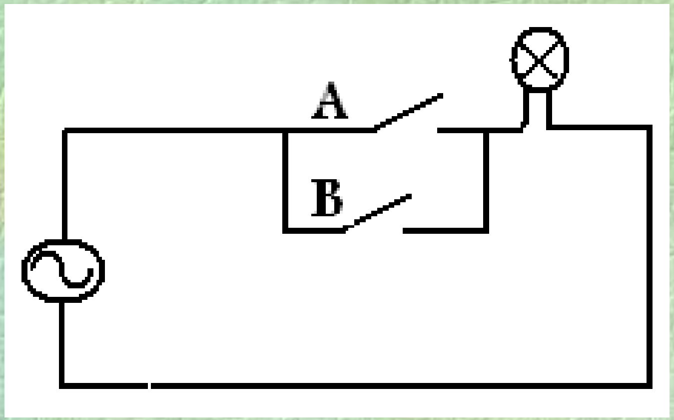
- Value "1" only if all inputs are "1"
- Acts as electrical switches in series
- Denote by " . "



X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

Logical Operation: OR

- Value "1" if any of the inputs is "1"
- Acts as electrical switches in parallel
- Denote by "+"



X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

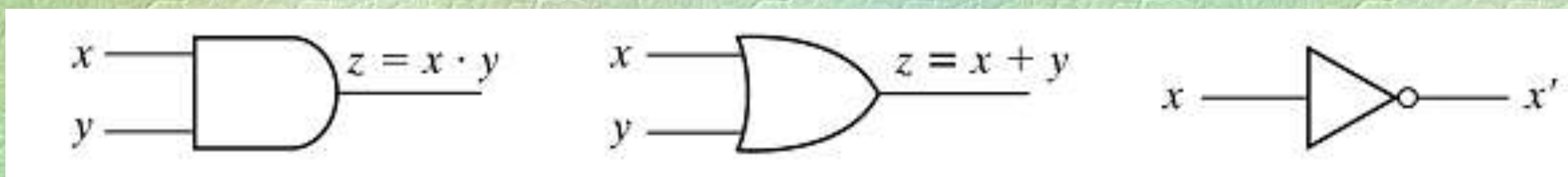
Logical Operation: NOT

- Reverse the value of input
- Denote by complement sign (!x or x′ or $\bar{x}$).
- Also called "inverter"

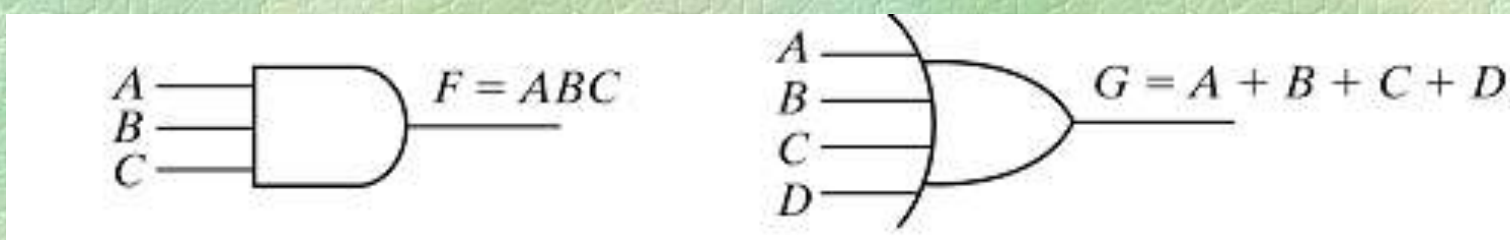
X	X′
0	1
1	0

Logic Gates

- Is electronic digital circuits (logic circuits)
  [Mano p.29-30]
- Is blocks of hardware Called "digital circuits", "switching circuits", "logic circuits" or simply "gates"

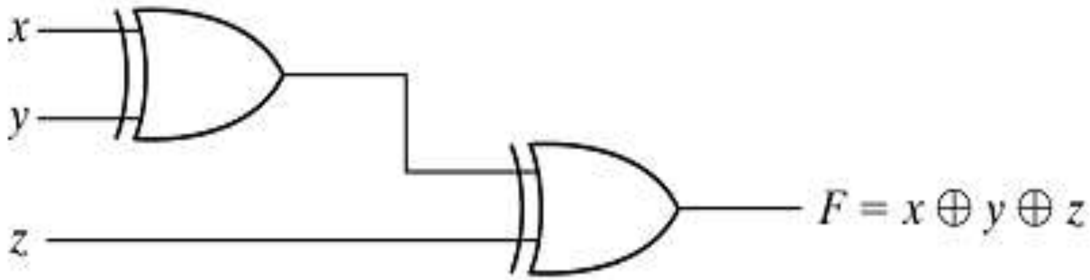


(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

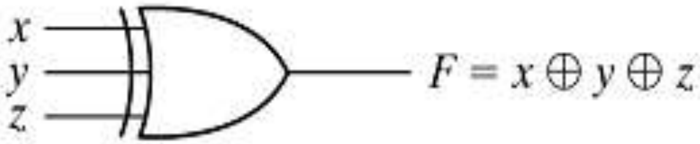


(a) Three-input AND gate (b) Four-input OR gate

X-OR Gates



(a) Using 2-input gates

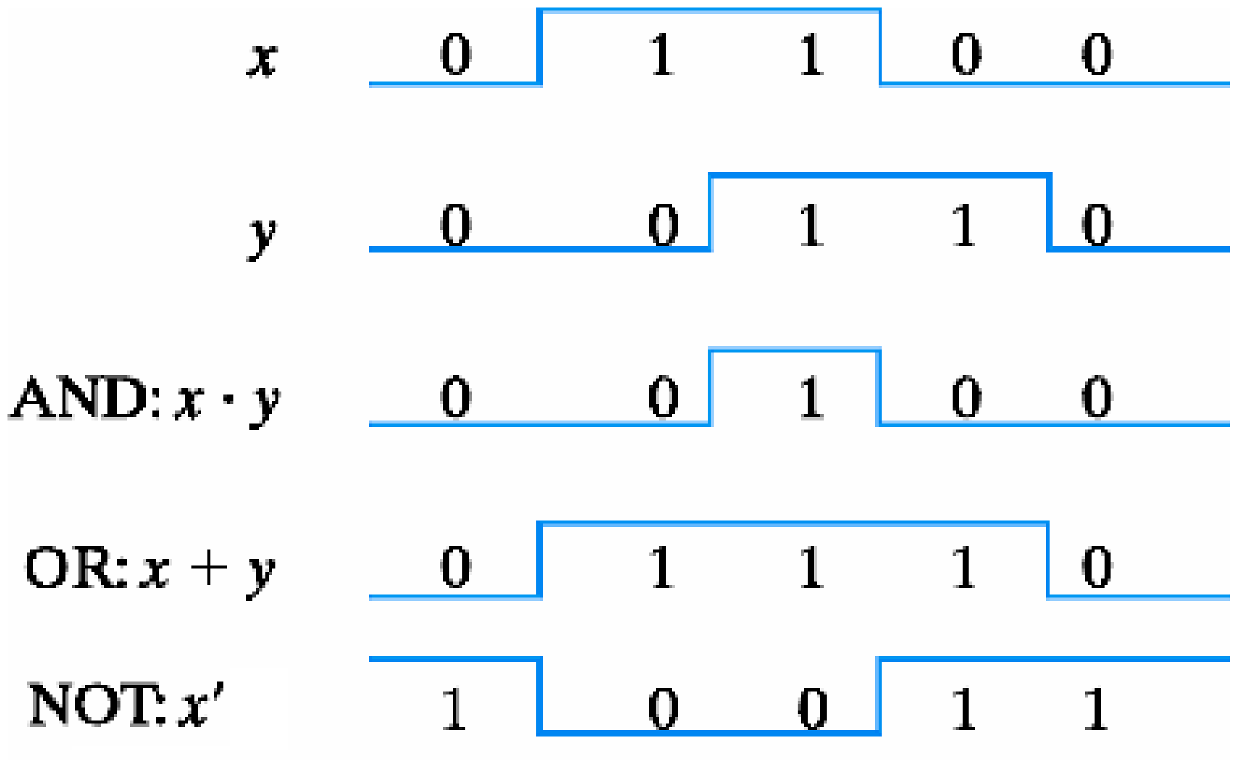


(b) 3-input gate

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

Input-Output Signals



Input-output signals for gates

Binary Signals Levels

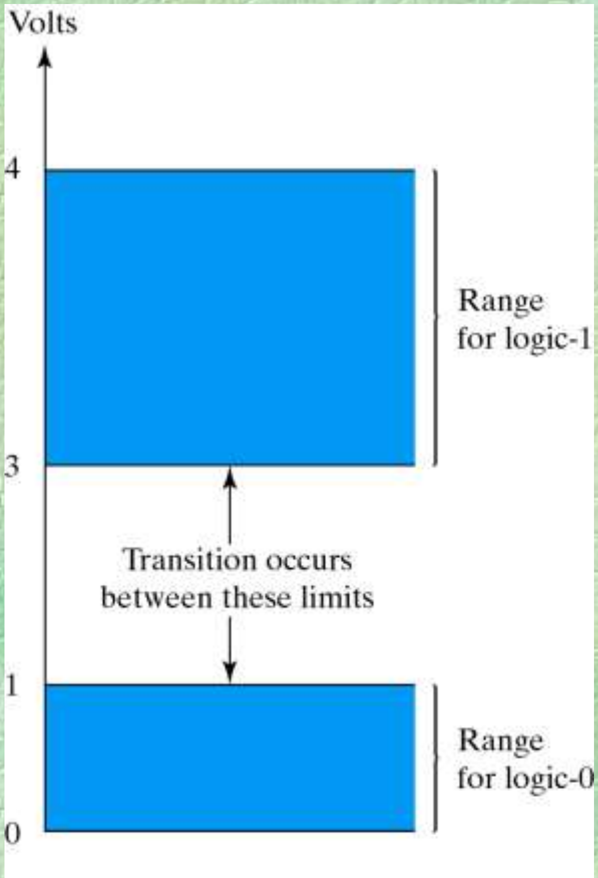
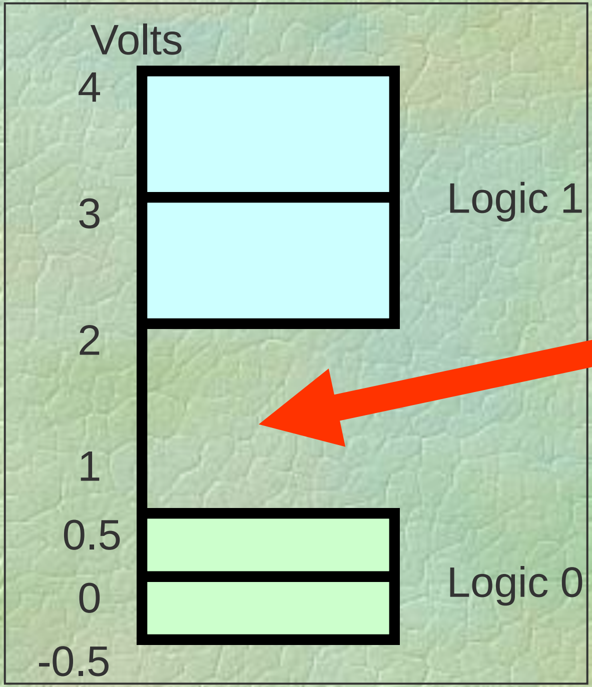


Fig. 1-3 Example of binary signals



- Acceptable level of deviation
- Nominal level
- State of transition

Positive and negative logic

(a) Positive logic (b) Negative logic

BCD Code

- Although the binary number system is the most natural system for a computer, most people are more accustomed to decimal system.
- Convert decimal numbers to binary, perform all arithmetic calculations in binary and then convert the binary results back to decimal.
- So, we represent the decimal digits by means of a code that contains 1′s and 0′s.
- Also possible to perform the arithmetic operations directly with decimal numbers when they are stored in coded form.

BCD Code

- Ex1: BCD for $(396)_{10}$ is $(0011\ 1001\ 0110)_{BCD}$
- Ex2: $(185)_{10}=(0001\ 1000\ 0101)_{BCD} = (10111001)_2$
- So, the BCD has 12 bits, but binary equivalent has 8 bits

BCD Addition

4	0100	4	0100	8	1000
+ 5	0101	+8	1000	+9	1001
9	1001	12	1100	17	10001
			+ 0110		+ 0110
			1 0010		1 0111

Binary 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

Decimal Arithmetic of BCD

- Add (+375) + (-240)= +135

	0 375
Complement of 240	+ 9 760
Discard the end carry	0 135

The 9 in the leftmost position of the second number represents a minus

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	0 0 0 0
1	0001	0001	0100	0 1 1 1
2	0010	0010	0101	0 1 1 0
3	0011	0011	0110	0 1 0 1
4	0100	0100	0111	0 1 0 0
5	0101	1011	1000	1 0 1 1
6	0110	1100	1001	1 0 1 0
7	0111	1101	1010	1 0 0 1
8	1000	1110	1011	1 0 0 0
9	1001	1111	1100	1 1 1 1
	1010	0101	0000	0 0 0 1
Unused	1011	0110	0001	0 0 1 0
bit	1100	0111	0010	0 0 1 1
combi-	1101	1000	1101	1 1 0 0
nations	1110	1001	1110	1 1 0 1
	1111	1010	1111	1 1 1 0

Gray Code

Binary		Reflected Code (Gray code)	Decimal Digit
0000		0000	0
0001	1-bit change	0001	1
0010	…	0011	2
0011		0010	3
0100		0110	4
0101		0111	5
0110	1-bit change	0101	6
0111		0100	7
1000		1100	8
1001		1101	9
1010		1111	10
1011		1110	11
1100		1010	12
1101		1011	13
1110		1001	14
1111		1000	15

ASCII Character Code

- The ASCII (American Standard Code for Information Interchange)
  - 7 bits per character to code 128 characters including special characters ($ = 0100010)
  - It uses 94 graphic characters that can be printed and 34 non-printing characters used for control functions.
  - There are 3 types of control characters: format effectors, information separators, and communication control characters.

01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

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

ASCII 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

ASCII Character Code (Contd.)

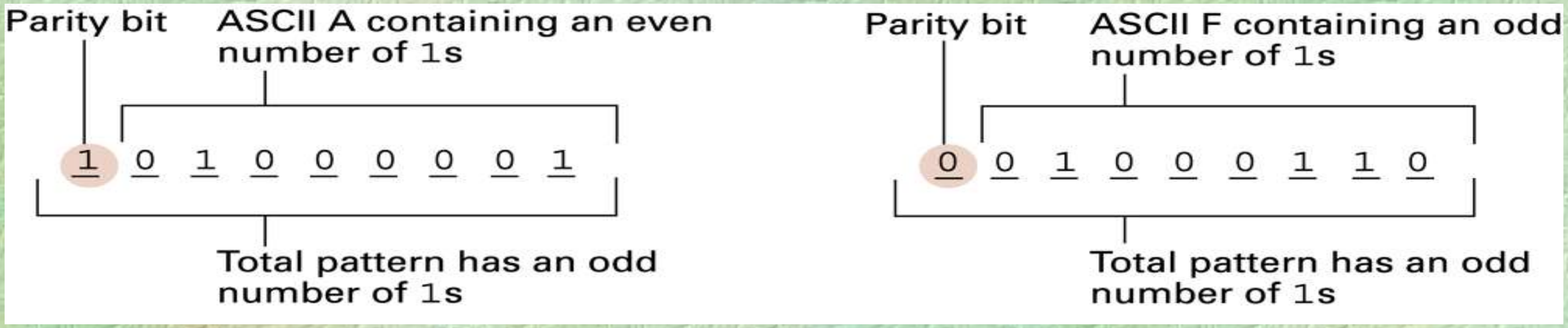
- Although ASCII code is a 7-bitcode, ASCII characters are most often stored one per byte.
- The extra bit are used for other purposes, depending on the application.
- For Ex., some printers recognize 8-bit ASCII characters with the MSB set to 0.
- Additional 128 8-bit characters with the MSB set to 1 are used for other symbols such as the Greek alphabet or italic type font.

Error Detecting Code

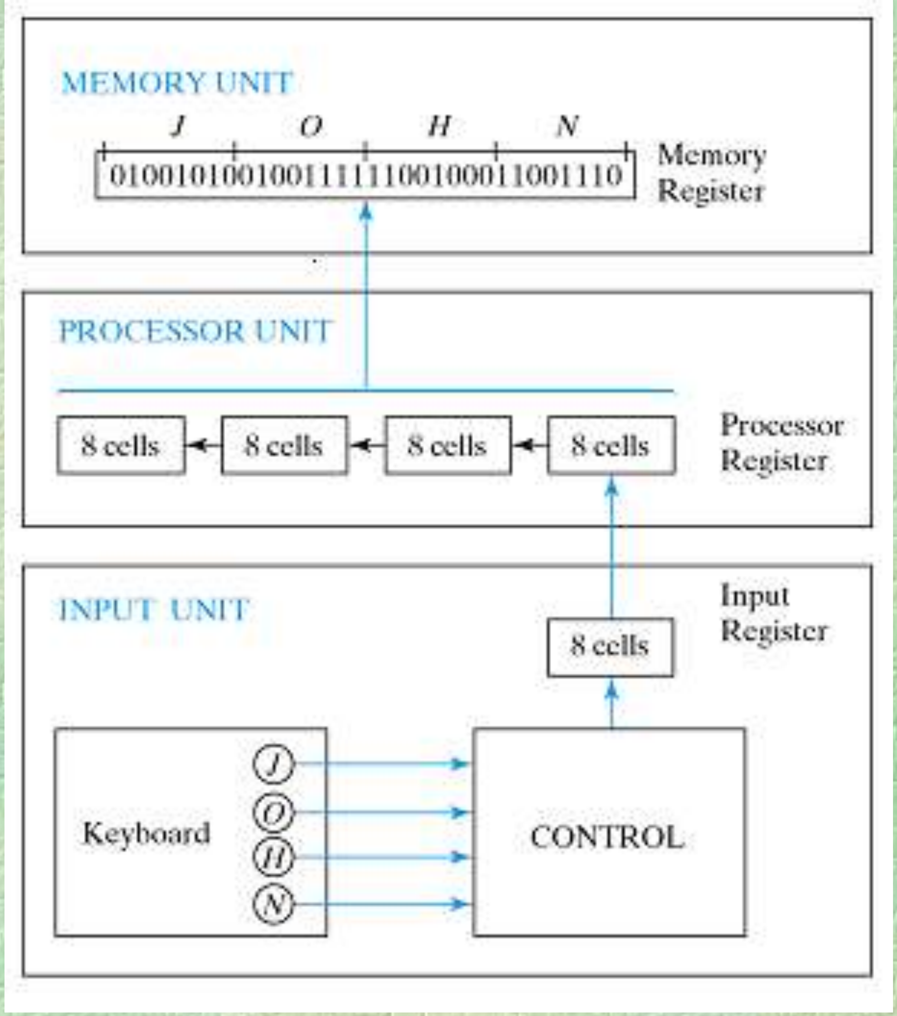
- To detect errors in data communication and processing, the eighth bit is used to indicate parity.
- This parity bit is an extra bit included with a message to make the total number of 1's either even or odd.

		with even parity	with odd parity
Ex:	ASCII A = 1000001	01000001	11000001
	ASCII T = 1010100	11010100	01010100

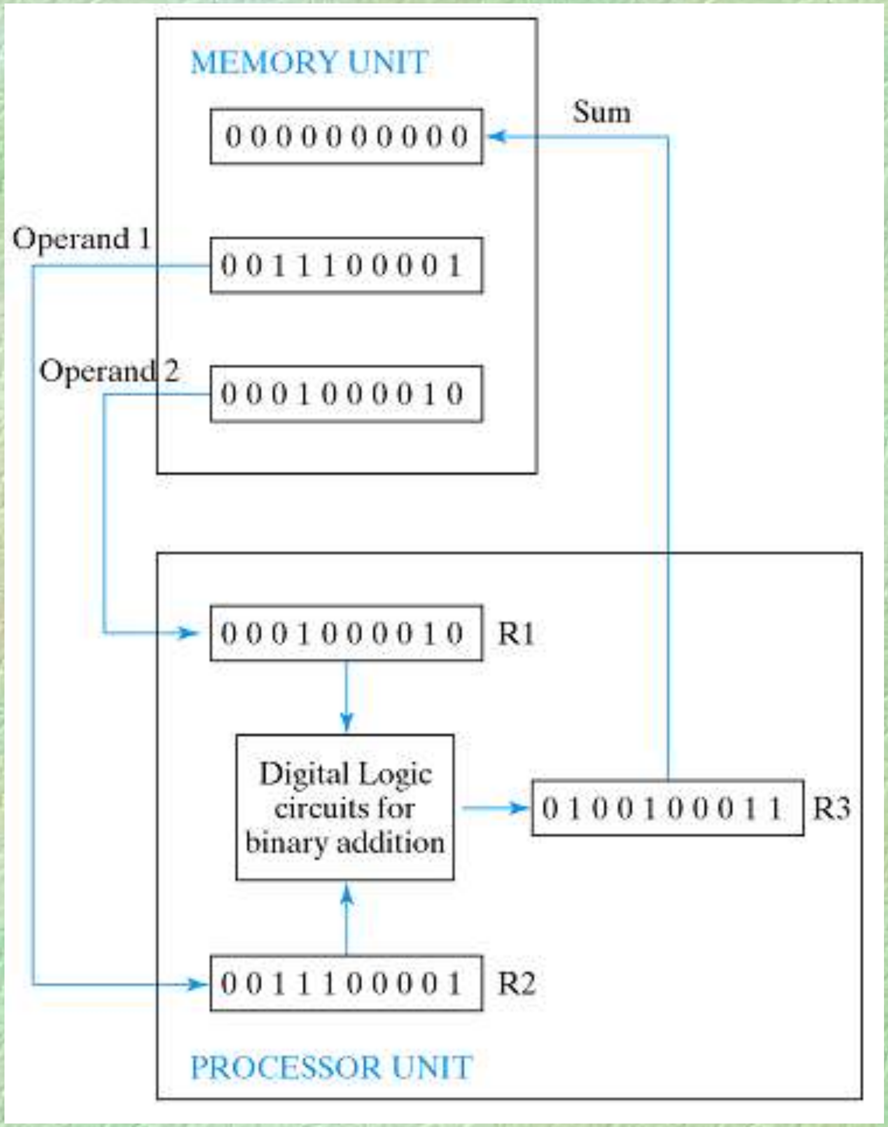
The ASCII codes for the letters A and F adjusted for odd parity



Transfer of information with registers



Example of Binary information system



Exercises

- Problem 1-2
- Problem 1-3
- Problem 1-10
- Problem 1-16

- My Advise :

  Do all problems p: 30-31,