Digital Logic Design

Digital Logic Design

- **Description**:
  - *This course include: Number Systems and Binary Codes; Boolean Algebra and Basic Results; Switching Functions; Minimization Techniques; Analysis and Design of Combinational and Sequential Logic circuits.*

- **Course Objectives** :

  A student who successfully fulfils the course requirements will have demonstrated an ability to:
  - *Perform arithmetic operations in many number systems*
  - *Manipulate Boolean algebraic structures*
  - *Analyze and design various combinational logic circuits.*
  - *Analyze and design clocked sequential circuits*

Digital Logic Design

- **Textbook:**
  - *Digital Design - Morris Mano – 4th Edition.*

- **References :**
  - *Thomas L. Floyd, Digital Fundamentals , Merrill, imprint of Macmillan Publishing Company New York, 1994.*
  - *M. M. Mano & C. R. Kime, Logic and Computer Design Fundamentals, Prentice-Hall 2001.*
  - *B. Stephen & V. Zvonko, Fundamentals of Digital Logic with VHDL Design , McGraw-Hill 2000*

Digital Logic Design

Laboratory Works: *:*

- *Laboratory sessions are organized in parallel to theoretical study given in classrooms. Student required to perform at least 7 experiments and submit reports for evaluation.*

Grading System *:*

- *Continuous Assessment (**30%**)*
  - *Quizzes, Homeworks and Lab work* **20**
  - *Midterm Exam* **10**
- *Final examinations (**70%**)*
  - *Lab Exam* **25**
  - *Final Exam* **45**

Digital Logic Design

Chapter 1
Digital Systems and Binary Numbers

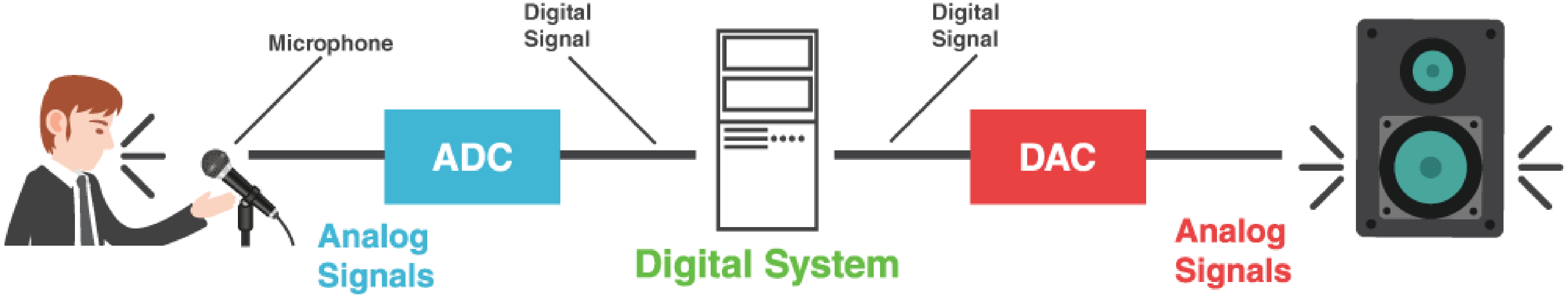
Outline of Chapter 1

- *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*

1.1 Digital Systems and Binary Numbers

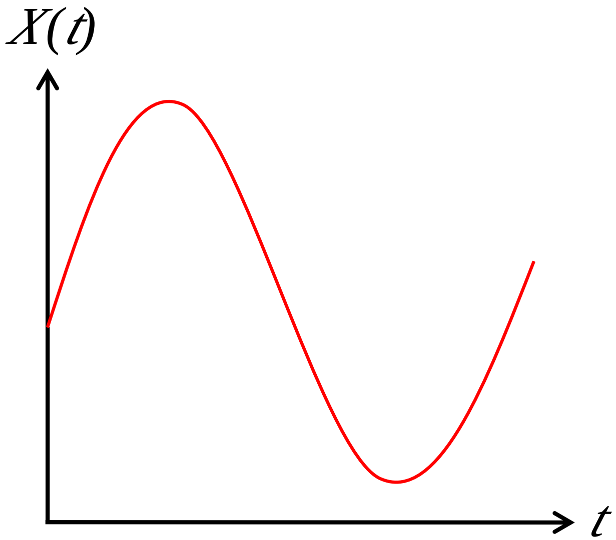
- *Digital computers*
  - *General purposes*
  - *Many scientific, industrial and commercial applications*
- *Digital systems (Embedded)*
  - *Telephone switching exchanges*
  - *Digital camera*
  - *Electronic calculators, PDA's*
  - *Digital TV*
- *Discrete information-processing systems*
  - *Manipulate discrete elements of information (Factories)*

Analog and Digital Signal

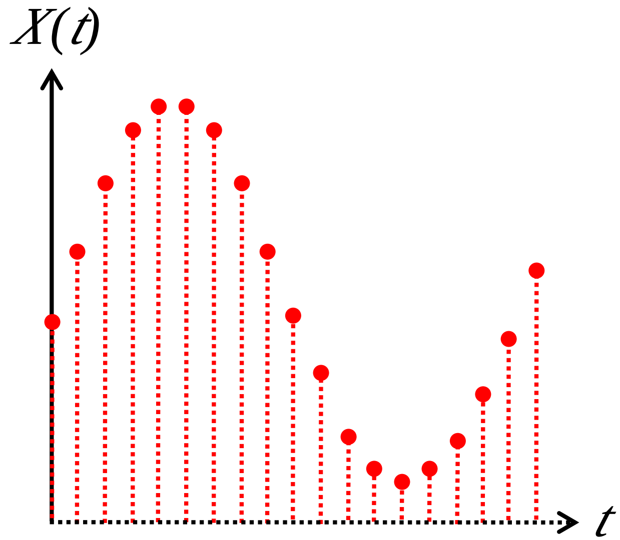


Analog and Digital Signal

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



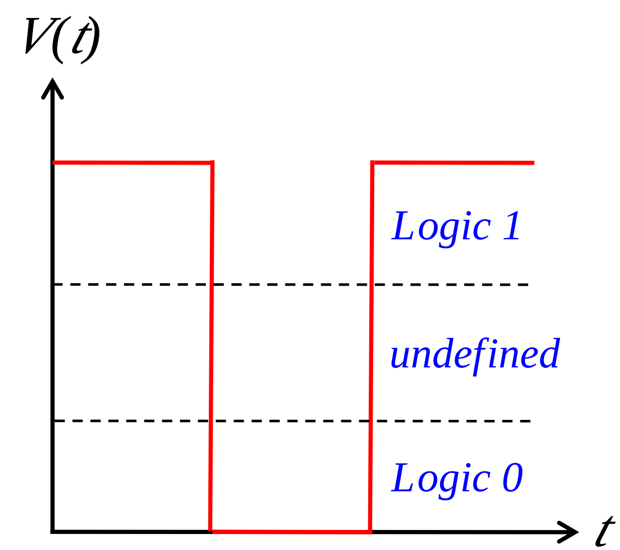
Analog signal



Digital signal

Binary Digital Signal

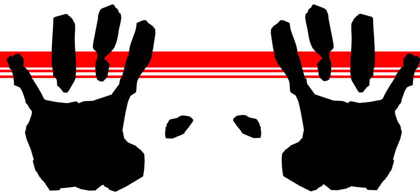
- *An information variable represents a physical quantity.*
- *For digital systems, the variable takes on discrete values.*
  - *Two level, or binary values are the most prevalent values.*
- *Binary values are represented abstractly by:*
  - *Digits 0 and 1*
  - *Words (symbols) False (F) and True (T)*
  - *Words (symbols) Low (L) and High (H)*
  - *And words On and Off*
- *Binary values are represented by values or ranges of values of physical quantities.*



Binary digital signal

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*



2	1	0		-1	-2
5	1	2	.	7	4
100	10	1		0.1	0.01
500	10	2		0.7	0.04

$$d_2 * B^2 + d_1 * B^1 + d_0 * B^0 + d_{-1} * B^{-1} + d_{-2} * B^{-2}$$

$$(512.74)_{10}$$

Octal Number System

- **Base = 8**
  - 8 digits { 0, 1, 2, 3, 4, 5, 6, 7 }
- **Weights**
  - Weight = (Base)^{Position}
- **Magnitude**
  - Sum of "Digit x Weight"
- **Formal Notation**

64	8	1		1/8	1/64
5	1	2	.	7	4
2	1	0		-1	-2

$$5 * 8^{2} + 1 * 8^{1} + 2 * 8^{0} + 7 * 8^{-1} + 4 * 8^{-2}$$

$$=(330.9375)_{10}$$

$$(512.74)_{8}$$

1.2 Binary Number System

- *Base = 2*
  - *2 digits { 0, 1 }, called binary digits* or "*bits*"
- *Weights*
  - *Weight = (Base)^{Position}*
- *Magnitude*
  - Sum of "*Bit x Weight*"
- *Formal Notation*
- *Groups of bits*   *4 bits = Nibble*

  8 bits = Byte

4	2	1		1/2	1/4
1	0	1	.	0	1
2	1	0		-1	-2

$$1*2^2+0*2^1+1*2^0+0*2^{-1}+1*2^{-2}$$

$$=(5.25)_{10}$$

$$(101.01)_2$$

1 0 1 1

1 1 0 0 0 1 0 1

The Power of 2

n	2^n
0	$2^0=1$
1	$2^1=2$
2	$2^2=4$
3	$2^3=8$
4	$2^4=16$
5	$2^5=32$
6	$2^6=64$
7	$2^7=128$

n	2^n	
8	$2^8=256$	
9	$2^9=512$	
10	$2^{10}=1024$	**Kilo**
11	$2^{11}=2048$	
12	$2^{12}=4096$	
20	$2^{20}=1M$	**Mega**
30	$2^{30}=1G$	**Giga**
40	$2^{40}=1T$	**Tera**

1.3 Hexadecimal Number System

- *Base = 16*
  - *16 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }*
- *Weights*
  - *Weight = (Base)* ^{Position}
- *Magnitude*
  - Sum of "*Digit x Weight*"
- *Formal Notation*

256	16	1		1/16	1/256
1	E	5	•	7	A
2	1	0		-1	-2

$$1*16^2+14*16^1+5*16^0+7*16^{-1}+10*16^{-2}$$

$$=(485.4765625)_{10}$$

$$(1E5.7A)_{16}$$

Binary Addition

- *Column Addition*

	1	1	1	1	1	1		
		1	1	1	1	0	1	= 61
+			1	0	1	1	1	= 23
	1	0	1	0	1	0	0	= 84

$\geq (2)_{10}$

Binary Subtraction

- Borrow a "Base" when needed

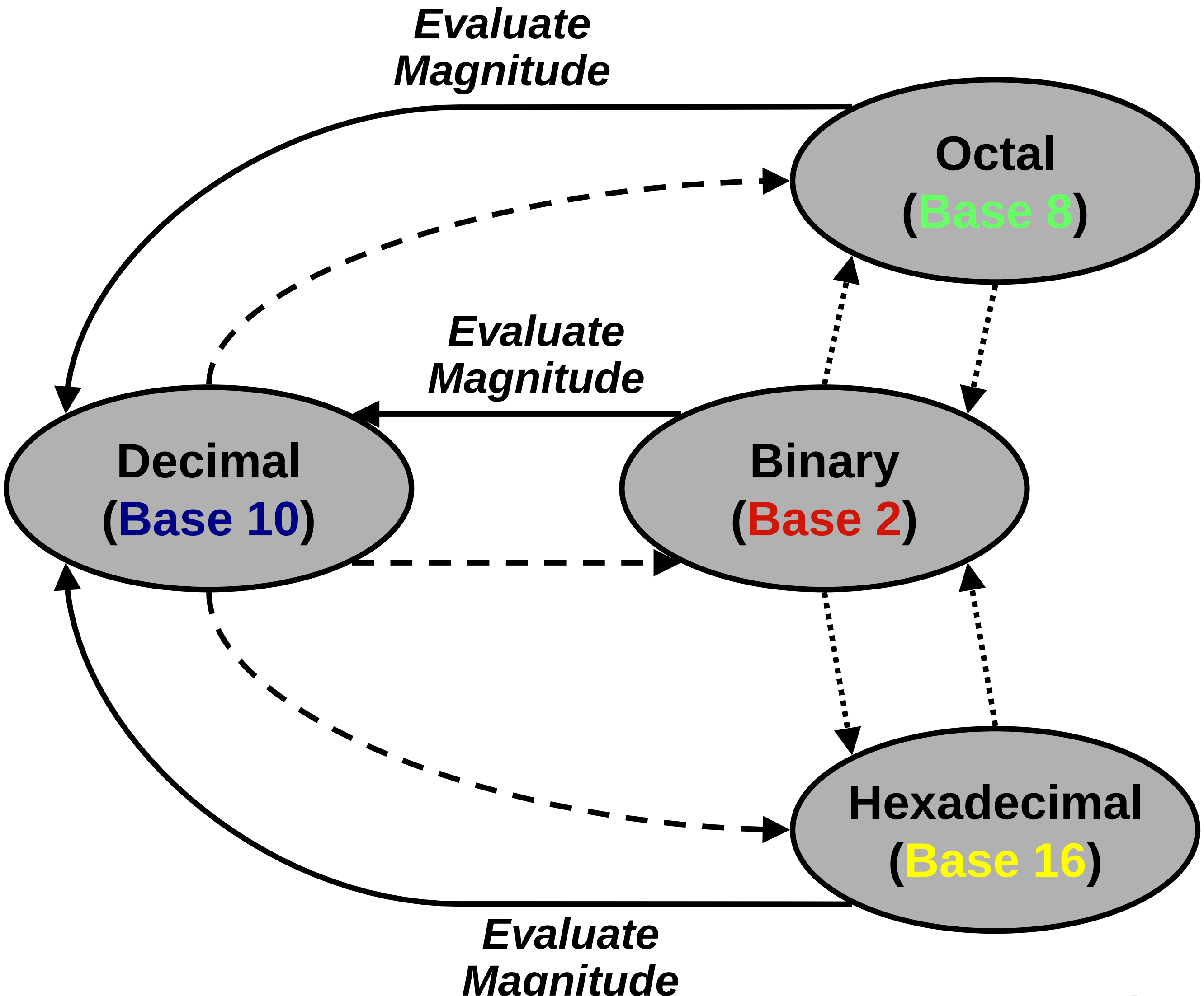
		1			2			$= (2)_{10}$
	0	~~2~~	2	0	0	2		
	~~1~~	0	0	~~1~~	~~1~~	0	1	= 77
−			1	0	1	1	1	= 23
	0	**1**	**1**	**0**	**1**	**1**	**0**	= 54

Binary Multiplication

- *Bit by bit*

				1	**0**	**1**	**1**	**1**
x					1	0	1	0
				0	**0**	**0**	**0**	**0**
			1	**0**	**1**	**1**	**1**	
		0	**0**	**0**	**0**	**0**		
	1	**0**	**1**	**1**	**1**			
	1	**1**	**1**	**0**	**0**	**1**	**1**	**0**

1.4 Number Base Conversions



Decimal (*Integer*) to Binary Conversion

- *Divide* the number by the 'Base' *(=2)*
- *Take the remainder (either 0 or 1) as a coefficient*
- *Take the quotient and repeat the division*

Example: $(13)_{10}$

	Quotient	Remainder	Coefficient
13 / 2 =	6	1	$a_0 = 1$
6 / 2 =	3	0	$a_1 = 0$
3 / 2 =	1	1	$a_2 = 1$
1 / 2 =	0	1	$a_3 = 1$

Answer: $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

MSB LSB

Decimal (*Fraction*) to Binary Conversion

- *Multiply* the number by the 'Base' *(=2)*
- *Take the integer (either 0 or 1) as a coefficient*
- *Take the resultant fraction and repeat the division*

Example: $(0.625)_{10}$

		Integer		Fraction	Coefficient
0.625	* 2 =	1	.	25	$a_{-1} = 1$
0.25	* 2 =	0	.	5	$a_{-2} = 0$
0.5	* 2 =	1	.	0	$a_{-3} = 1$

Answer: $(0.625)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_2 = (0.101)_2$

MSB LSB

Decimal to Octal Conversion

Example: $(175)_{10}$

	Quotient	Remainder	Coefficient
175 / 8 =	21	7	$a_0 = 7$
21 / 8 =	2	5	$a_1 = 5$
2 / 8 =	0	2	$a_2 = 2$

Answer: $(175)_{10} = (a_2\, a_1\, a_0)_8 = (257)_8$

Example: $(0.3125)_{10}$

	Integer		Fraction	Coefficient
0.3125 * 8 =	2	.	5	$a_{-1} = 2$
0.5 * 8 =	4	.	0	$a_{-2} = 4$

Answer: $(0.3125)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_8 = (0.24)_8$

Binary – Octal Conversion

- *$8 = 2^3$*
- *Each group of 3 bits represents an octal digit*

Example:

Assume Zeros

$$(\,10110.01\,)_2$$

$$(\,2\;\;6\,.\,2\,)_8$$

Octal	*Binary*
0	*0 0 0*
1	*0 0 1*
2	*0 1 0*
3	*0 1 1*
4	*1 0 0*
5	*1 0 1*
6	*1 1 0*
7	*1 1 1*

Works both ways (*Binary* to *Octal* & *Octal* to *Binary*)

Binary − Hexadecimal Conversion

- $16 = 2^4$
- *Each group of 4 bits represents a hexadecimal digit*

Example:

Assume Zeros

$(\ 1\ 0\ 1\ 1\ 0\ .\ 0\ 1\ )_2$

$(\ 1\quad 6\ .\ 4\ )_{16}$

Hex	*Binary*
0	*0 0 0 0*
1	*0 0 0 1*
2	*0 0 1 0*
3	*0 0 1 1*
4	*0 1 0 0*
5	*0 1 0 1*
6	*0 1 1 0*
7	*0 1 1 1*
8	*1 0 0 0*
9	*1 0 0 1*
A	*1 0 1 0*
B	*1 0 1 1*
C	*1 1 0 0*
D	*1 1 0 1*
E	*1 1 1 0*
F	*1 1 1 1*

Works both ways (*Binary* to *Hex* & *Hex* to *Binary*)

Octal − Hexadecimal Conversion

- *Convert to Binary as an intermediate step*

Example:

$$(\;2\quad 6\;.\;2\;)_8$$

Assume Zeros

$$(\;0\,1\,0\,1\,1\,0\,.\,0\,1\,0\;)_2$$

Assume Zeros

$$(\;1\quad 6\;.\;4\;)_{16}$$

Works both ways (*Octal* to *Hex* & *Hex* to *Octal*)

Decimal, Binary, Octal and Hexadecimal

Decimal	Binary	Octal	Hex
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

1.5 Complements

- *There are* **two types** *of complements for each base-**r** system: the radix complement and diminished radix complement.*
- **Diminished Radix Complement - (r-1)'s Complement**
  - *Given a number N in base r having n digits, the* $(r-1)$'s complement *of N is defined as:*

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

- **Example for 6-digit <u>decimal</u> numbers***:*
  - *9*'s complement is $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
  - *9*'s complement of *546700 is 999999–546700 = 453299*
- **Example for 7-digit <u>binary</u> numbers:**
  - *1*'s complement is $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
  - *1*'s complement of *1011000 is 1111111–1011000 = 0100111*

Complements in Binary Numbers

- *Observations:*
  - *Subtraction from (r^n – 1) will never require a borrow*
  - *Diminished radix complement can be computed digit-by-digit*
  - *For binary: 1 – 0 = 1 and 1 – 1 = 0*
- *1*'s Complement (*Diminished Radix* Complement)
  - All '*0*'s become '*1*'s
  - All '*1*'s become '*0*'s

Example $(10110000)_2$

⇨ $(01001111)_2$

If you add a number and its 1's complement …

$$
\begin{array}{r}
1\,0\,1\,1\,0\,0\,0\,0 \\
+\ 0\,1\,0\,0\,1\,1\,1\,1 \\
\hline
1\,1\,1\,1\,1\,1\,1\,1
\end{array}
$$

Complements

- *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 512398 is 487602
  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 Binary Numbers

- 2's Complement (*Radix Complement*)
  - *Take 1*'s complement then add *1*

OR

  - Toggle all bits to the left of the first '*1*' from the right

Example:

1's *Comp. of*:

	1 0 1 1 0 0 0 0	**1 0 1 1 0 0 0 0**
	0 1 0 0 1 1 1 1	
+	**1**	
	0 1 0 1 0 0 0 0	**0 1 0 1 0 0 0 0**

Lecture One

تم بحمد الله

Complements

- **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

- *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

- *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.*

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

$$
\begin{array}{lrr}
\text{(b)} & Y = & 1000011 \\
 & \text{2's complement of } X = & +\,\underline{0101100} \\
 & \text{Sum} = & 1101111
\end{array}
$$

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

1.6 Signed Binary Numbers

- *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

- *Table 1.3 lists all possible four-bit signed binary numbers in the three representations.*

Signed Binary Numbers

- *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.*
  - *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:*

+ 6	00000110	− 6	11111010
+13	00001101	+13	00001101
+ 19	00010011	+ 7	00000111
+ 6	00000110	− 6	11111010
−13	11110011	−13	11110011
− 7	11111001	− 19	11101101

Signed Binary Numbers

- *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)*

1.7 Binary Codes

- *BCD Code*
  - *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

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

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

- *BCD addition*

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

Binary Code

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

BCD	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

Binary Codes

- *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

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 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.*

Binary Codes

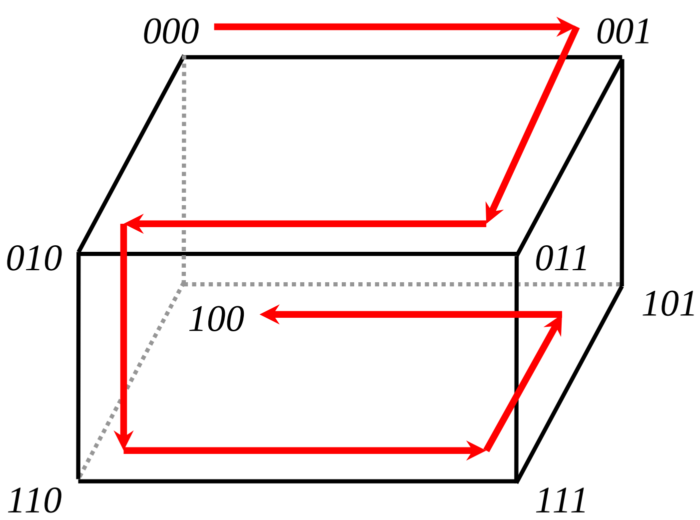
- *Other Decimal Codes*

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)

- *Gray Code*
  - *The advantage is that only bit in the code group changes in going from one number to the next.*
    - » *Error detection.*
    - » *Representation of analog data.*
    - » *Low power design.*



1-1 and onto!!

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

- *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:*

	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 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.
  - *Example:*

Message A with even parity : *110001001*

Message A with odd parity: *010001001*

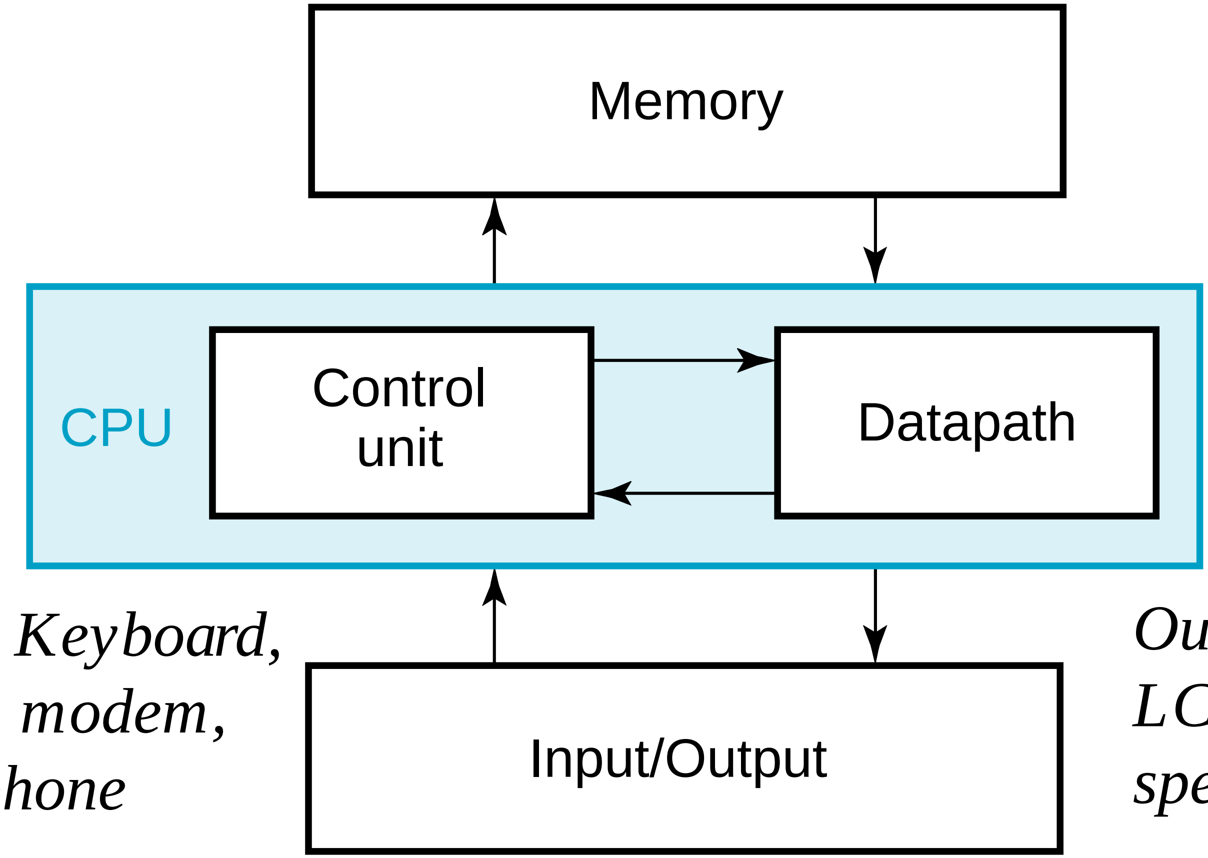
1.8 Binary Storage and Registers

- *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 $\rightarrow$ 2^n possible states

- *A binary cell*
  - *Two stable state*
  - *Store one bit of information*
  - *Examples: flip-flop circuits, 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.*

A Digital Computer Example



Inputs: Keyboard, mouse, modem, microphone

Outputs: CRT, LCD, modem, speakers

Synchronous or Asynchronous?

Transfer of information

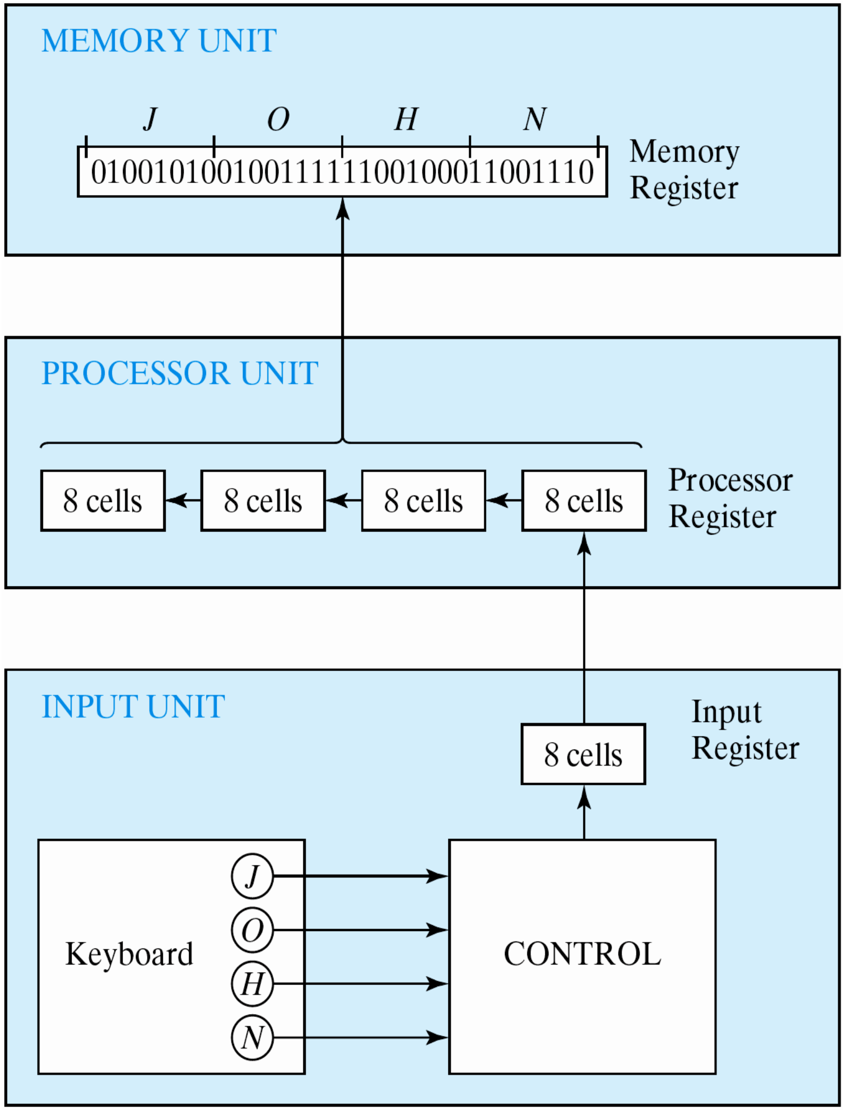


Figure 1.1 Transfer of information among register

Transfer of information

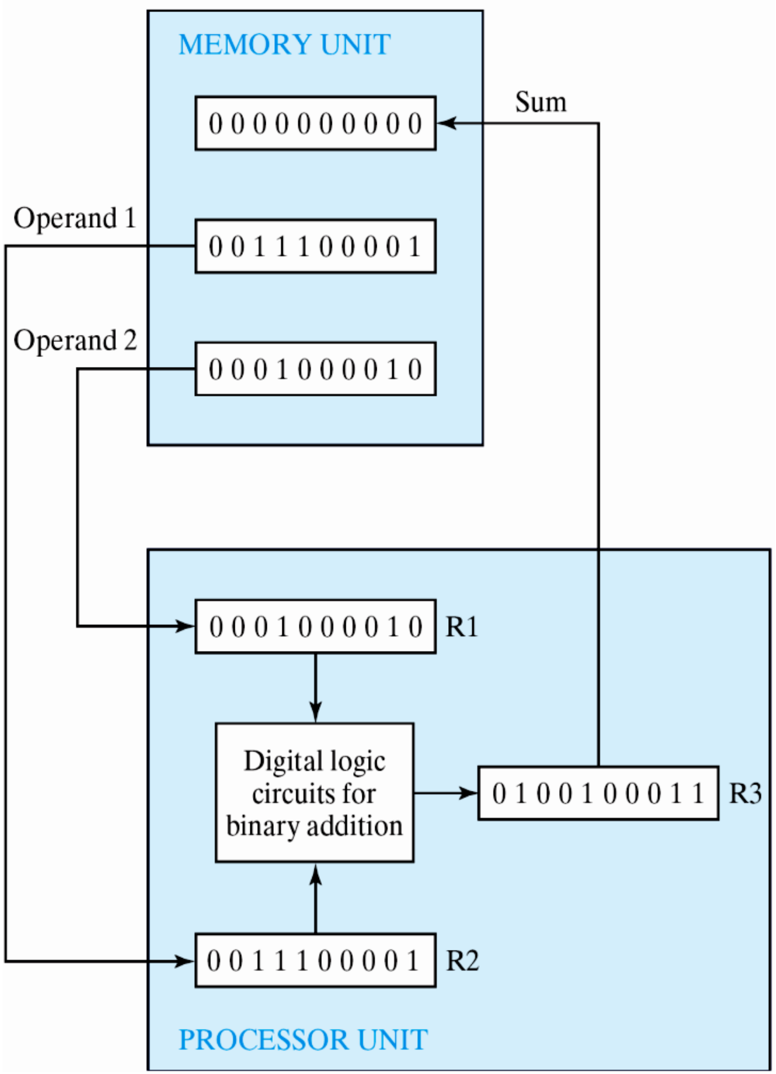


Figure 1.2 Example of binary information processing

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

LOAD R1;
LOAD R2;
ADD R3, R2, R1;
STORE R3;

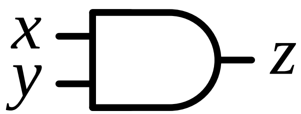
Binary Logic

- *Truth Tables, Boolean Expressions, and Logic Gates*

AND

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

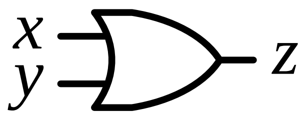
$$z = x \bullet y = x\,y$$



OR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	1

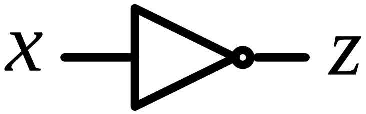
$$z = x + y$$



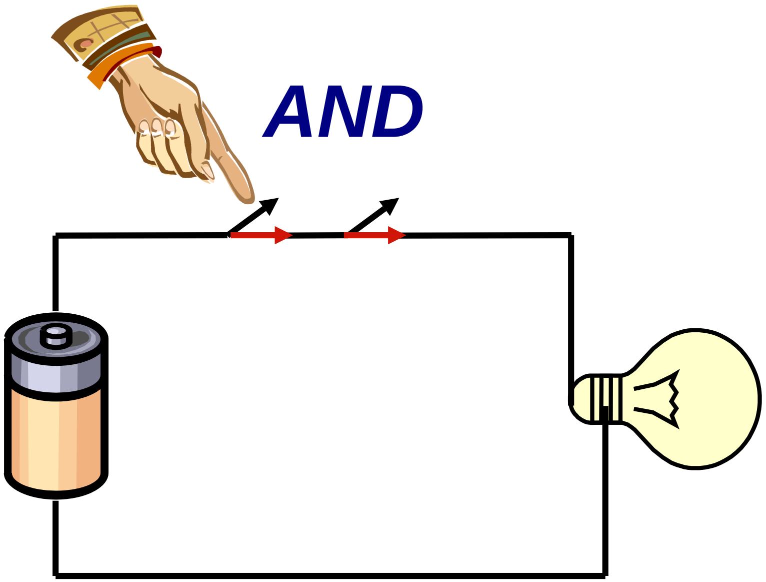
NOT

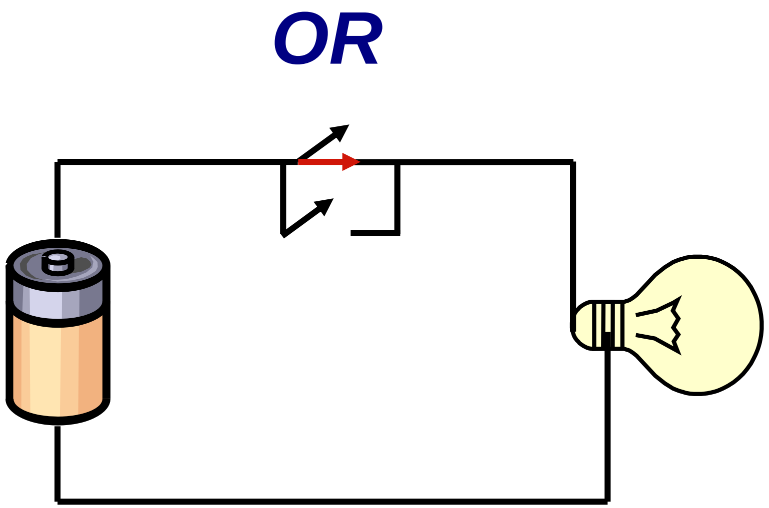
x	*z*
0	1
1	0

$$z = \overline{x} = x'$$



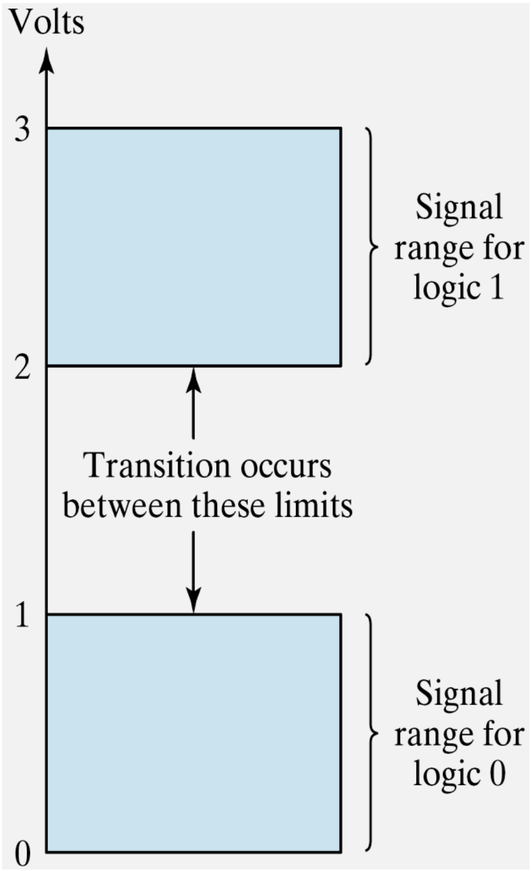
Switching Circuits





Binary Logic

- **Logic gates**
  - *Example of binary signals*



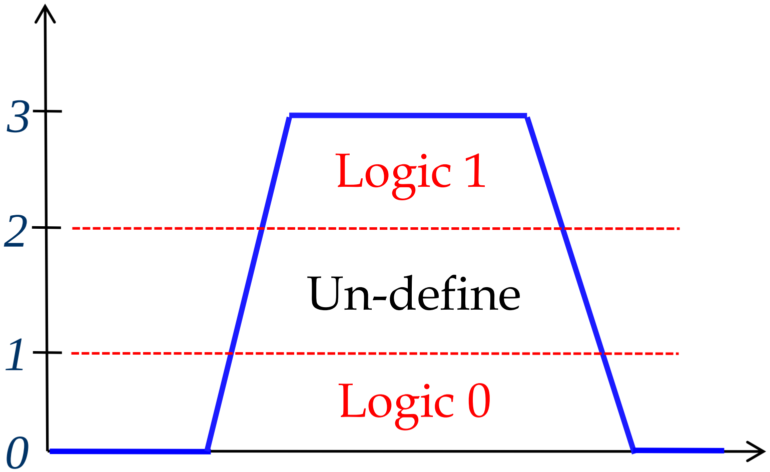


Figure 1.3 Example of binary signals

Binary Logic

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

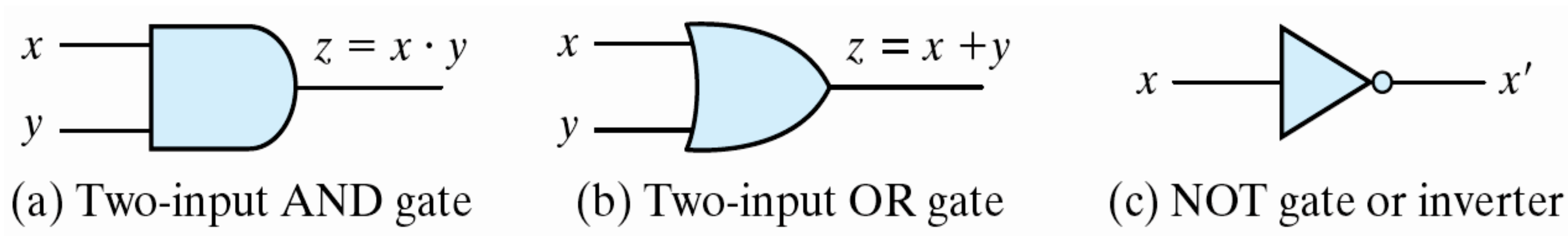


Fig. 1.4 Symbols for digital logic circuits

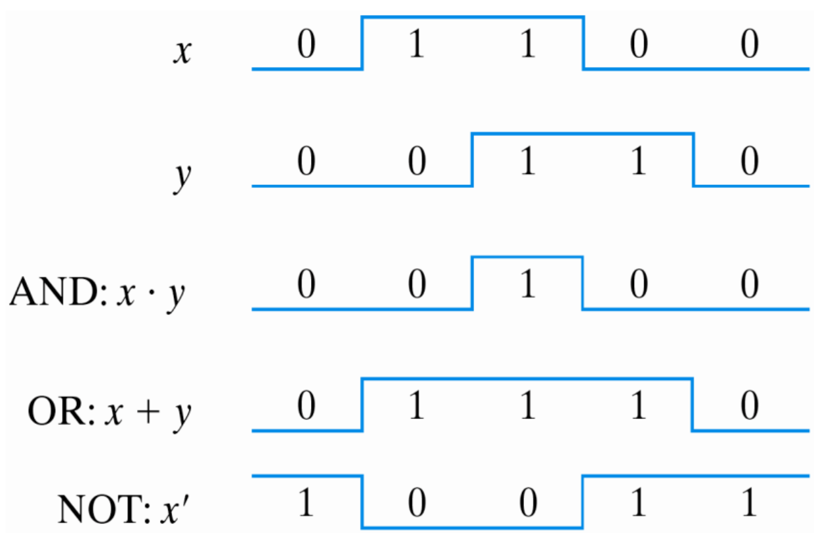
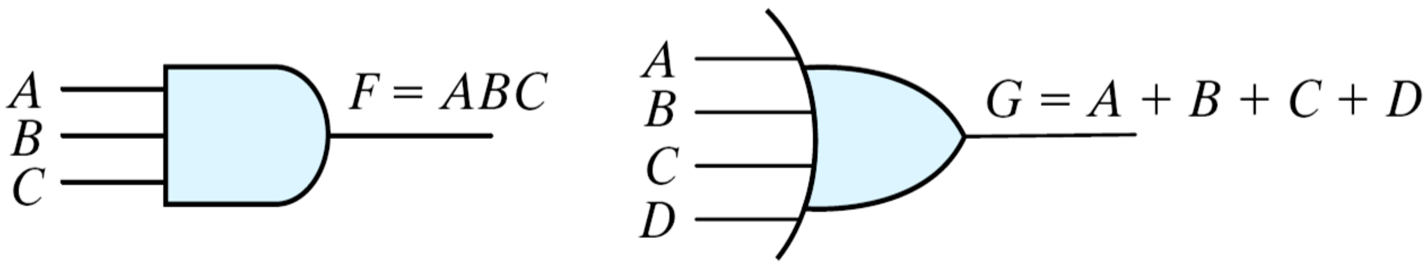


Fig. 1.5 Input-Output signals for gates

Binary Logic

- *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

THE END

HomeWork (1):

*Digital Design (**4th**)- Morris Mano-Page 32-*

Problems:

1.8

1.9

1.13

1.14

1.18

THE END

HomeWork (2):

*Digital Design (**4th**)- Morris Mano-Page 32- Problems:*

1.20

1.22

1.29

1.30

1.34