EASTERN MEDITERRANEAN UNIVERSITY • DOĞU AKDENİZ ÜNİVERSİTESİ • 1979

INTRODUCTION TO LOGIC DESIGN

Chapter 1

Digital Systems and Binary Numbers

gürtaçyemişçioğlu

OUTLINE OF CHAPTER 1

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

1.1 DIGITAL SYSTEMS

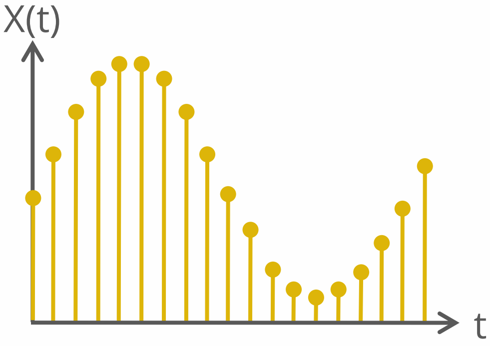
DIGITAL SYSTEMS

- Digital age and information age
- Digital computers
  - General purposes
  - Many scientific, industrial and commercial applications
- Digital systems
  - Telephone switching exchanges
  - Digital camera
  - Electronic calculators, PDA's
  - Digital TV
- Discrete information-processing systems
  - Manipulate discrete elements of information
  - For example, {1, 2, 3, …} and {A, B, C, …}…

DIGITAL SYSTEMS

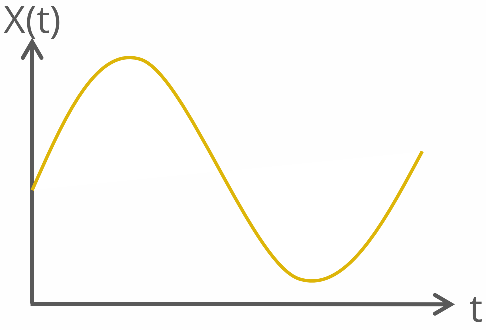
DIGITAL SIGNAL

- The physical quantities or signals can assume only discrete values.
- Greater accuracy



ANALOG SIGNAL

- The physical quantities or signals may vary continuously over a specified range.

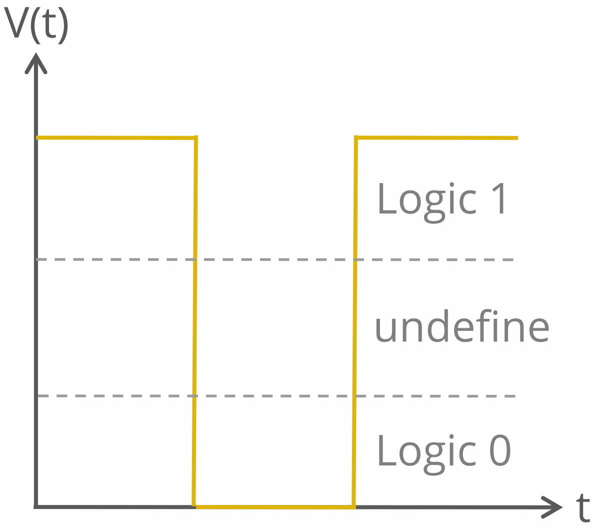


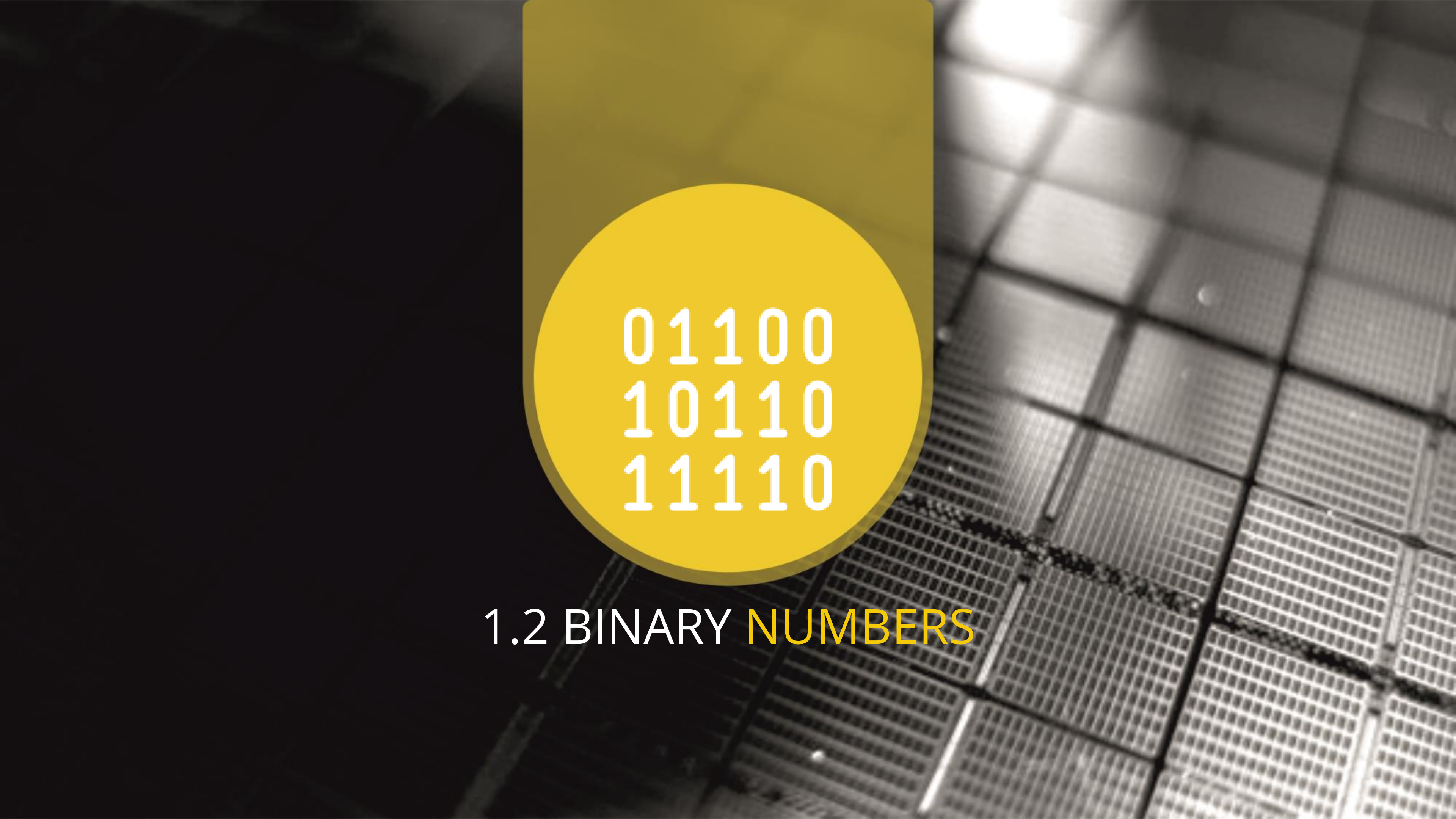
DIGITAL SYSTEMS

- Binary digital signal
  - An information variable represented by 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

DIGITAL SYSTEMS

- Binary values are represented by values or ranges of values of physical quantities.





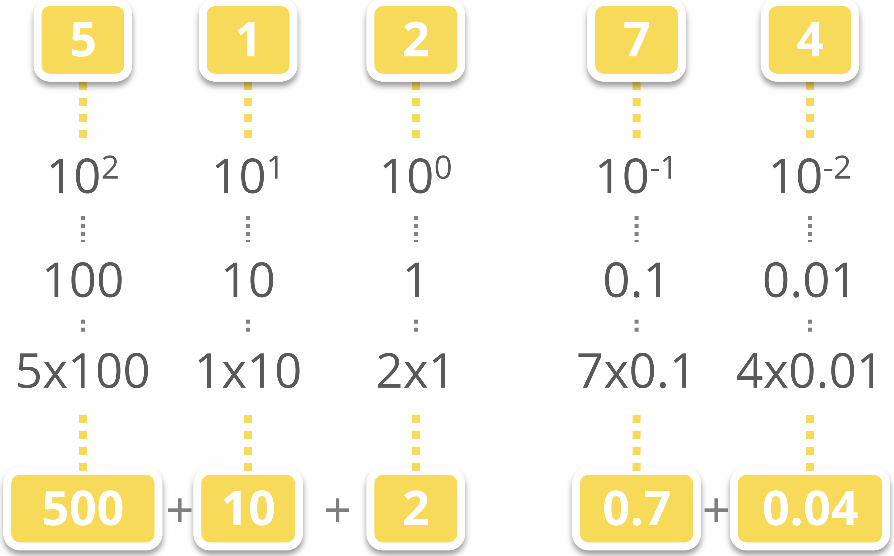
01100
10110
11110

1.2 BINARY NUMBERS

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



5	1	2	7	4
10^2	10^1	10^0	10^{-1}	10^{-2}
100	10	1	0.1	0.01
5x100	1x10	2x1	7x0.1	4x0.01

500 + 10 + 2 0.7 + 0.04

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

$$(512.74)_{10}$$

BINARY NUMBERS

Octal Number System

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

5	**1**	**2**	**7**	**4**
8^2	8^1	8^0	8^{-1}	8^{-2}
64	8	1	1/8	1/64
5x64	1x8	2x1	7x1/8	4x1/64

320 + **8** + **2** **0.875** + **0.0625**

$$o_2 \times B^2 + o_1 \times B^1 + o_0 \times B^0 + o_{-1} \times B^{-1} + o_{-2} \times B^{-2}$$

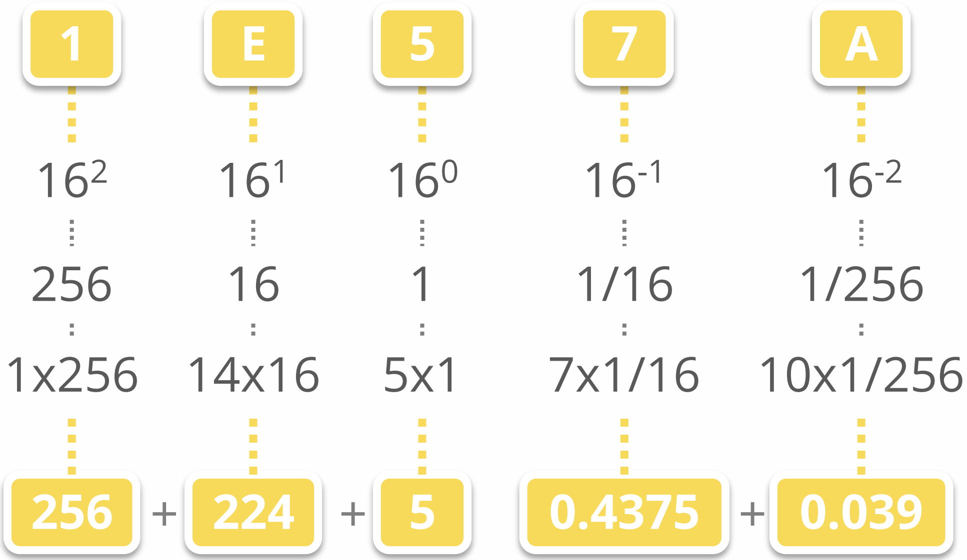
$$(330.9375)_{10}$$

$$(512.74)_8$$

BINARY NUMBERS

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 "*bit* x *Weight*"
- Formal Notation



1	E	5	7	A
16^2	16^1	16^0	16^{-1}	16^{-2}
256	16	1	1/16	1/256
1x256	14x16	5x1	7x1/16	10x1/256
256	224	5	0.4375	0.039

256 + 224 + 5 0.4375 + 0.039

$$H_2 \text{ x } B^2 + H_1 \text{ x } B^1 + H_0 \text{ x } B^0 + H_{-1} \text{ x } B^{-1} + H_{-2} \text{ x } B^{-2}$$

$(458.4765625)_{10}$

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

BINARY NUMBERS

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*

1	**0**	**1**	**0**	**1**
2^2	2^1	2^0	2^{-1}	2^{-2}
4	2	1	0.5	0.25
1x4	0x2	1x1	0x0.5	1x0.25
4 +	**0** +	**1**	**0** +	**0.25**

$$b_2 \times B^2 + b_1 \times B^1 + b_0 \times B^0 + b_{-1} \times B^{-1} + b_{-2} \times B^{-2}$$

$(\mathbf{5.25})_{10}$

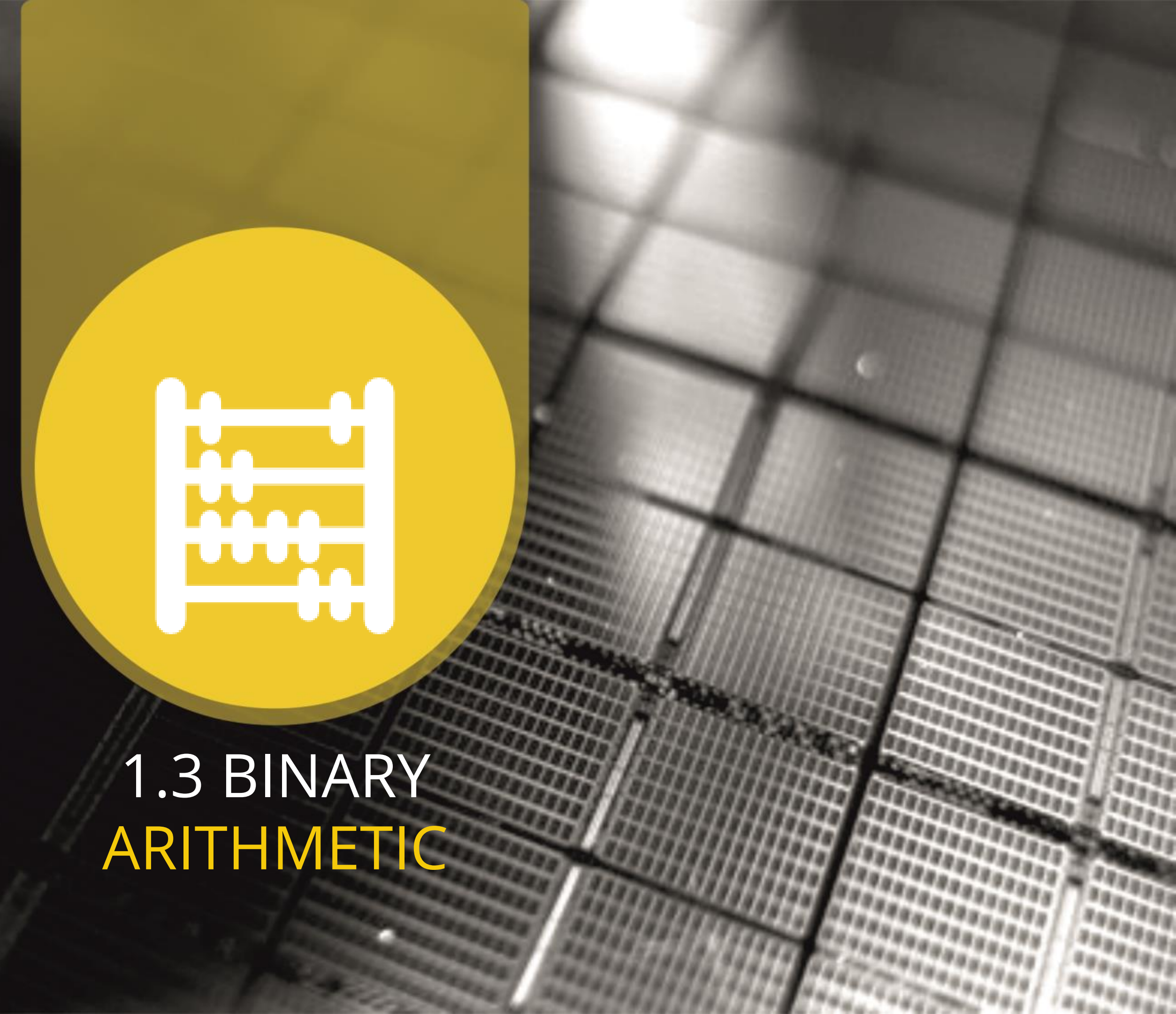
$(\mathbf{101.01})_2$

BINARY NUMBERS

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	$2^{10}=1024$
11	$2^{11}=2048$
12	$2^{12}=4096$
20	$2^{20}=1M$
30	$2^{30}=1G$
40	$2^{40}=1T$



1.3 BINARY ARITHMETIC

BINARY ARITHMETIC

ADDITION

Decimal Addition

$$\begin{array}{cccc} & 1 & 1 & \\ & & 5 & 5 \\ + & & 5 & 5 \\ \hline & 1 & 1 & 0 \end{array}$$

← **Carry**

= ***Ten ≥ Base***

➔ **Subtract a Base**

BINARY ARITHMETIC

ADDITION

Binary Addition - Column Addition

$$\begin{array}{ccccccccc} & 1 & 1 & 1 & 1 & 1 & 1 & & \\ & & 1 & 1 & 1 & 1 & 0 & 1 & = 61 \\ + & & & 1 & 0 & 1 & 1 & 1 & = 23 \\ \hline & 1 & 0 & 1 & 0 & 1 & 0 & 0 & = 84 \end{array}$$

$\geq (2)_{10}$

BINARY ARITHMETIC

SUBTRACTION

Binary Subtraction - Borrow a “Base” when needed

$$
\begin{array}{cccccccc|l}
 & & 1 & & & 2 & & & = (10)_2 \\
 & 0 & \cancel{2} & 2 & 0 & 0 & 2 & & \\
 & \cancel{1} & 0 & 0 & \cancel{1} & \cancel{1} & 0 & 1 & = 77 \\
- & & & 1 & 0 & 1 & 1 & 1 & = 23 \\
\hline
 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & = 54
\end{array}
$$

BINARY ARITHMETIC

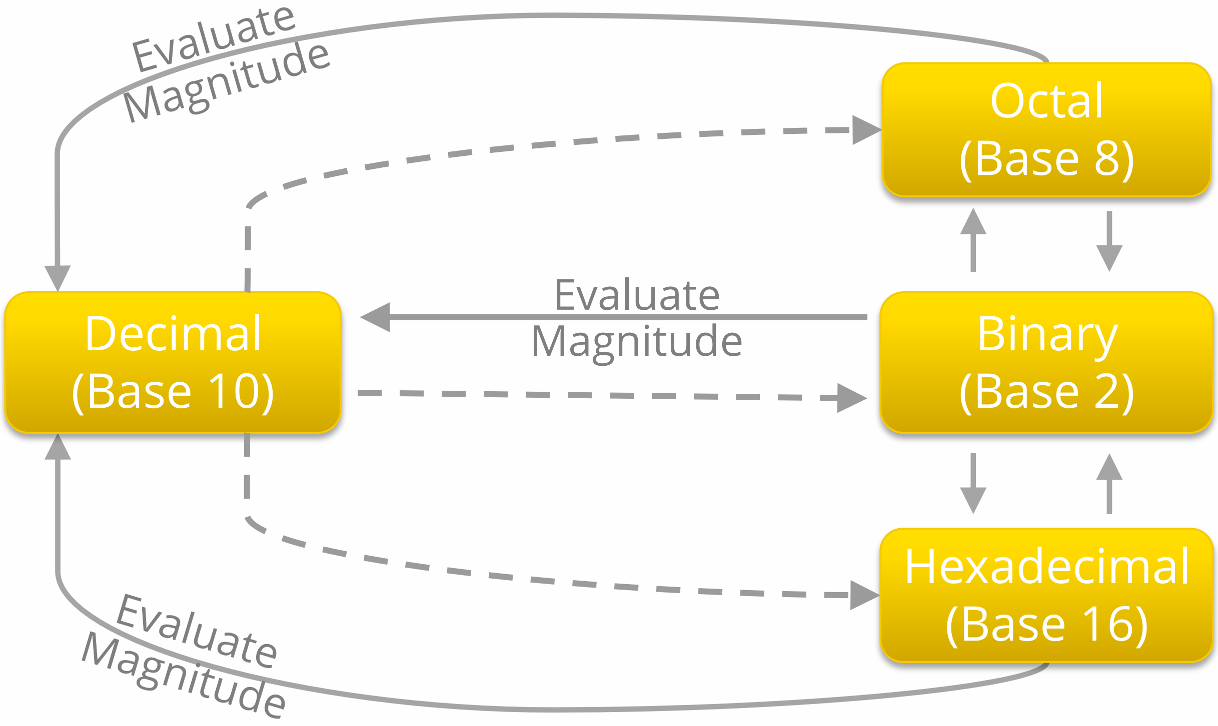
MULTIPLICATION

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 CONVERSION

NUMBER BASE CONVERSION



NUMBER BASE CONVERSION

Decimal 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

NUMBER BASE CONVERSION

Decimal (integer) to binary conversion

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: $$(13)_{10} = (\underset{\substack{\uparrow \\ \textbf{MSB}}}{a_3}\ a_2\ a_1\ \underset{\substack{\nwarrow \\ \textbf{LSB}}}{a_0})_2 = (1101)_2$$

NUMBER BASE CONVERSION

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

NUMBER BASE CONVERSION

Decimal (fraction) to binary conversion

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**

NUMBER BASE CONVERSION

Decimal (integer) 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: $(175)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (257)_8$

MSB **LSB**

NUMBER BASE CONVERSION

Decimal (fraction) to octal conversion

Example: $(0.3125)_{10}$

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

Answer: $(0.3125)_{10} = (0.a_{-1}\ a_{-2}\ a_{-3})_2 = (0.24)_8$

MSB **LSB**

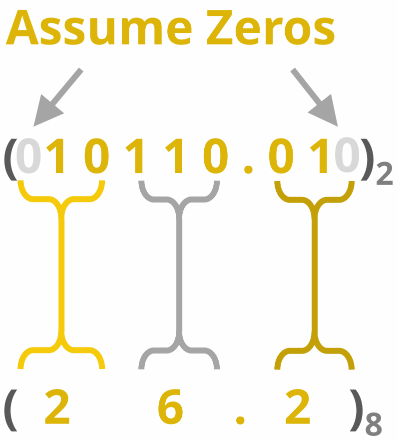
8

1.5 OCTAL & HEXADECIMAL NUMBERS

OCTAL & HEXADECIMAL NUMBERS

- Binary to octal conversion
- 8 = 2^3
- Each group of 3 bits represents an octal digit

Example:



Assume Zeros

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

$(2 \quad 6 . 2)_8$

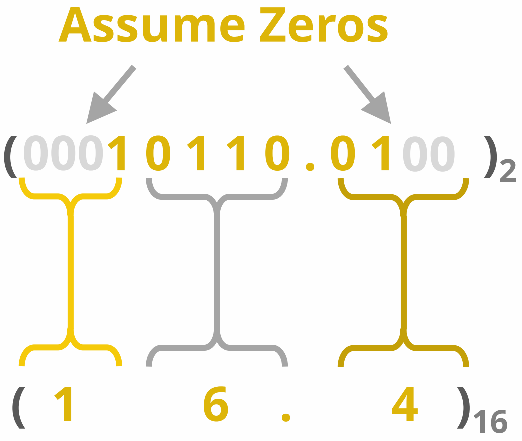
Works both ways (*Binary* to *Octal* & *Octal* to *Binary*)

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

OCTAL & HEXADECIMAL NUMBERS

- Binary to hexadecimal conversion
- $16 = 2^4$
- Each group of 4 bits represents a hexadecimal digit

Example:



Assume Zeros

$$(\underbrace{0001}_{1}\;\underbrace{0110}_{6}\;.\;\underbrace{0100}_{4}\;)_2$$

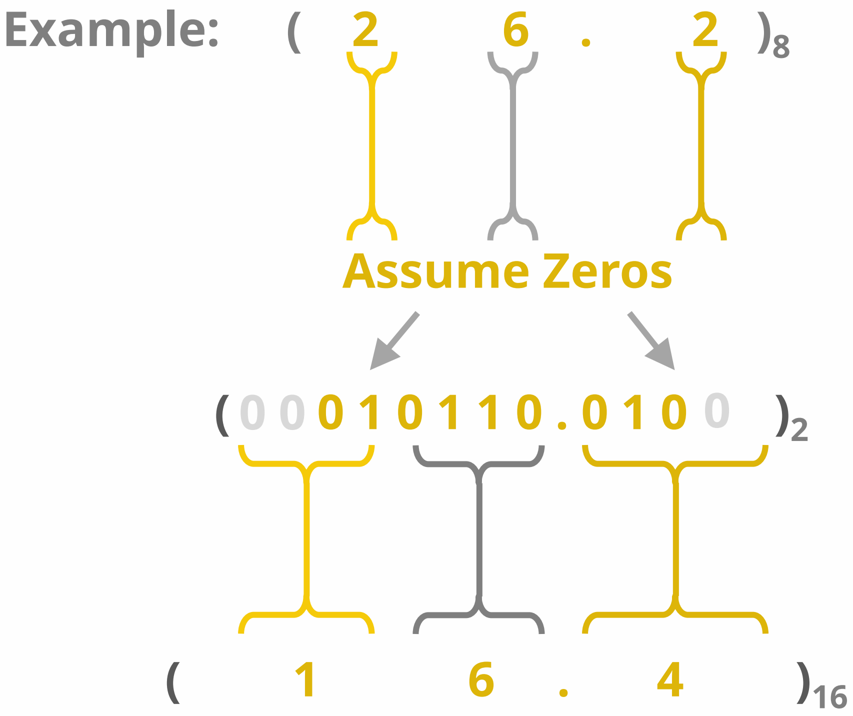
$$(\;1\quad 6\;.\;4\;)_{16}$$

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

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

OCTAL & HEXADECIMAL NUMBERS

- Octal to hexadecimal conversion
- Convert to binary as an intermediate step



Example: $(2\;6\;.\;2)_8$

Assume Zeros

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

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

OCTAL & HEXADECIMAL NUMBERS

Decimal	Octal	Hex	Binary
00	00	0	0000
01	01	1	0001
02	02	2	0010
03	03	3	0011
04	04	4	0100
05	05	5	0101
06	06	6	0110
07	07	7	0111
08	10	8	1000
09	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

1.6 COMPLEMENTS

COMPLEMENTS

- Complements are used in digital computers to simplify the subtraction operation and for logical manipulation. Simplifying operations leads to simpler, less expensive circuits to implement the operations.
- There are two types of complements for each base-*r* system:
  - diminished radix complement.
  - the radix complement

COMPLEMENTS

- **Diminished Radix Complement (r-1)'s Complement**
  - Given a Number = N, base = r, digits = n,
  - The $(r-1)$'s complement *of* N is defined as:

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

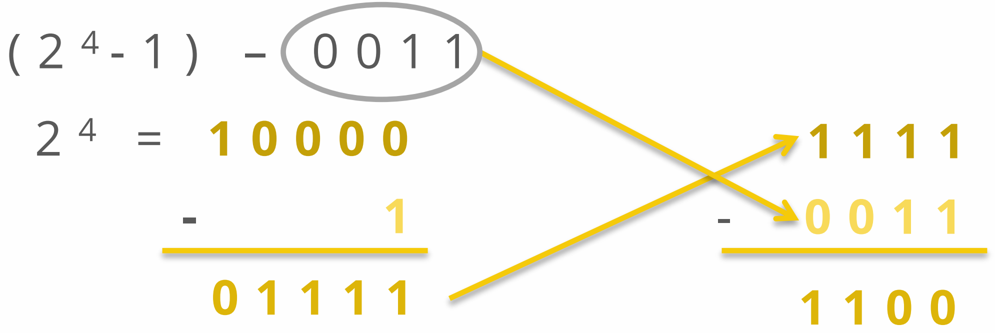
- **Example for 6-digit <u>decimal</u> numbers**:
  - 9's complement is $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
  - 9's complement of 546700 is 999999–546700 = 453299

COMPLEMENTS

- **For decimal numbers r = 10 and**
  - $(r\text{-}1)$ = 9, this is called 9’s complement of N.
  - $(10^n\text{-}1)\text{-}N$.
  - 10^n represents a number that consist of a single 1 followed by n 0’s.
  - $10^n\text{-}1$ is a number represented by n 9’s.
- **For binary numbers r = 2 and**
  - r-1 = 1, this is called 1’s complement of N.
  - $(2^n\text{-}1)\text{-}N$.
  - 2^n represents a binary number that consist of a 1 followed by n 0’s.

COMPLEMENTS

- **For Example: 0011**



$(2^4 - 1) - 0011$

$$2^4 = 10000$$

$$\begin{array}{r} 10000 \\ -\quad\quad 1 \\ \hline 01111 \end{array} \qquad \begin{array}{r} 1111 \\ -\ 0011 \\ \hline 1100 \end{array}$$

- When subtracting binary digit from 1 we can have either 1-0 =1 or 1-1 =0 which causes bit to change from 0 to 1 or from 1 to 0.
- 1's complement of a binary is formed by changing 1's to 0's and 0's to 1's.
- Example: 0 0 1 1 0 1 1 → 1 1 0 0 1 0 0

COMPLEMENTS

- **Example for 7-digit binary numbers:**
  - 1's complement is $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
  - 1's complement of 1011000 is 1111111–1011000 = 0100111
- **Observation:**
  - Subtraction from (r^n – 1) will never require a borrow
  - Diminished radix complement can be computed digit-by-digit

COMPLEMENTS

1’s Complement (*Diminished Radix* Complement)

- All ‘0’s become ‘1’s
- All ‘1’s become ‘0’s
- 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**
- Given a Number = *N*, base = *r*, digits = n,
  - The *r*'s complement *of N* is defined as:

$$r^n - N, \text{ for } N \neq 0 \text{ and } 0 \text{ for } N = 0.$$

- The r's complement is obtained by adding 1 to the (r – 1) 's complement, since
  - $= [(r^n - \cancel{1}) - N] + \cancel{1}.$
  - $= r^n - N$

COMPLEMENTS

- **10’s complement of N can be formed:**
  - By leaving all least significant 0’s unchanged.
  - By subtracting first non-zero least significant digit by 10
  - By subtracting all higher significant digits from 9.
- **2’s complement of N can be formed:**
  - By leaving all least significant 0’s and the first 1 unchanged and replacing 1’s with 0’s and 0’s with 1’s in all other higher significant digits.

COMPLEMENTS

- Example: Base-10
  - The 10's complement of 012398:

$$\begin{array}{ccccccc} & 9 & 9 & 9 & 9 & 9 & 10 \\ - & 0 & 1 & 2 & 3 & 9 & 8 \\ \hline & 9 & 8 & 7 & 6 & 0 & 2 \end{array}$$

  - The 10’s complement of 246700:

$$\begin{array}{ccccccc} & 9 & 9 & 9 & 10 & 0 & 0 \\ - & 2 & 4 & 6 & 7 & 0 & 0 \\ \hline & 7 & 5 & 3 & 3 & 0 & 0 \end{array}$$

COMPLEMENTS

- 2's Complement (*Radix* Complement)
  - Take 1's complement then add 1
  - Toggle all bits to the left of the first '1' from the right

Example:

Number: 1 0 1 1 0 0 0 0

1's Comp.:

1 1 1 1

0 1 0 0 1 1 1 1

\+ 1

0 1 0 1 0 0 0 0

or

1 0 1 1 0 0 0 0

0 1 0 1 0 0 0 0

COMPLEMENTS

- Example: Base-2
  - The 2's complement of 1101100 is 0010100

$$\begin{array}{cccccccc}
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
 & & & & & 1 & 1 & \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
+ & & & & & & & 1 \\
\hline
 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}$$

or

$$\begin{array}{ccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 \\
 & & & & & & \\
0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}$$

COMPLEMENTS

- Example: Base-2
  - The 2's complement of 0110111 is 1001001

$$\begin{array}{rr} & 0\ 1\ 1\ 0\ 1\ 1\ 1 \\ & 1\ 0\ 0\ 1\ 0\ 0\ 0 \\ + & 1 \\ \hline & 1\ 0\ 0\ 1\ 0\ 0\ 1 \end{array}$$

or

0 1 1 0 1 1 1

1 0 0 1 0 0 1

COMPLEMENTS

- Complement of the complement restores the number to its original value
  - r's complement of N is r^n-N
  - The complement of the complement is
  - $r^n - (r^n - N) = N$ → original number

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.
     - N r's complement = r^n-N
     - $= M + r^n - N$
     - $= M - N + r^n$
  2. If $M \geq N$, the sum will produce an end carry r^n, which can be discarded; 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 negative sign in front.

COMPLEMENT

- Example 1.5
  - Using 10's complement, subtract 72532 – 3250.

M = 72532 N = 3250

Step1: Take 10’s complement of N

$$\begin{array}{cccccc} & 9 & 9 & 9 & 10 & 0 \\ - & & 3 & 2 & 5 & 0 \\ \hline & 9 & 6 & 7 & 5 & 0 \end{array}$$

COMPLEMENT

Step2: To find the SUM, Add 10’s complement of N to M

$$\begin{array}{rr} & 72532 \\ + & 96750 \\ \hline \text{SUM=} & 169282 \end{array}$$

Step3: To find the ANSWER discard end carry 10^5, Subtract r^n from SUM

$$\begin{array}{rr} & 169282 \\ - & 100000 \\ \hline \text{ANSWER=} & 69282 \end{array}$$

COMPLEMENT

- Example 1.6
  - Using 10's complement, subtract 3250 - 72532.

M = 3250 N = 72532

Step1: Take 10’s complement of N

$$\begin{array}{rccccc} & 9 & 9 & 9 & 9 & 10 \\ - & 7 & 2 & 5 & 3 & 2 \\ \hline & 2 & 7 & 4 & 6 & 8 \end{array}$$

COMPLEMENT

Step2: To find the SUM, Add 10’s complement of N to M

$$\begin{array}{lrr} & & 03250 \\ & + & 27468 \\ \hline \text{SUM=} & & 30718 \end{array}$$

There is no end carry!

Step3: To find the ANSWER take the –(10’s complement of SUM)

$$\begin{array}{rccccc} & 9 & 9 & 9 & 9 & 10 \\ - & 3 & 0 & 7 & 1 & 8 \\ \hline & 6 & 9 & 2 & 8 & 2 \end{array}$$

ANSWER= - 6 9 2 8 2

COMPLEMENT

Example 1.7: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X – Y and (b) Y – X using 2’s complements.

(a) Step1: Take the 2’s complement of Y

1 0 0 0 0 1 1

0 1 1 1 1 0 1

Step2: Add 2’s complement of Y to X

$$\begin{array}{cccccccc} & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ + & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

There is an end carry!

COMPLEMENT

Step3: Discard the end carry.

  1 0 0 1 0 0 0 1

+ 1 0 0 0 0 0 0 0

ANSWER= 0 0 0 1 0 0 0 1

(b) Y – X: Step1: Take the 2’s complement of X

1 0 1 0 1 0 0

0 1 0 1 1 0 0

COMPLEMENT

Step2: Add 2's complement of X to Y

```
          1   0   0   0   0   1   1
      +   0   1   0   1   1   0   0
      ---------------------------------
SUM=      1   1   0   1   1   1   1
```

There is no end carry!

Step3: To find the answer Y – X = - (2's complement of SUM)

```
          1   1   0   1   1   1   1

        - 0   0   1   0   0   0   1
```

COMPLEMENTS

- 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(84 – 67 = 17)

Step1: Take the 1’s complement of Y

1 0 0 0 0 1 1

0 1 1 1 1 0 0

COMPLEMENTS

Step2: Add1’s complement of Y to the X

$$\begin{array}{rcccccccccl} & & 1 & 0 & 1 & 0 & 1 & 0 & 0 & & \\ & + & 0 & 1 & 1 & 1 & 1 & 0 & 0 & & \\ \hline \text{SUM=} & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & = 16 \end{array}$$

There is an end carry!
Sum is 1 less than the correct difference when an end carry occurs!

Step3: Remove the end carry and add 1 (End-around carry)

$$\begin{array}{rcccccccl} & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ & + & & & & & & & 1 & \\ \hline \text{ANSWER=} & & 0 & 0 & 1 & 0 & 0 & 0 & 1 & = 17 \end{array}$$

COMPLEMENTS

Example 1.8 :Repeat Example 1.7, but this time using 1's complement.

(a) Y – X = 1000011 – 1010100 (67 – 84 = -17)

Step1: Take the 1’s complement of X

1 0 1 0 1 0 0

0 1 0 1 0 1 1

COMPLEMENTS

Step2: Add1's complement of X to the Y

	1	0	0	0	0	1	1
+	0	1	0	1	0	1	1
SUM=	1	1	0	1	1	1	0

There is NO end carry!

Step3: To find the answer Y – X = - (1's complement of SUM)

	1	1	0	1	1	1	0	
ANSWER=	- 0	0	1	0	0	0	1	= -17

1.7 SIGNED BINARY NUMBERS

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 left most 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

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

- 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.

SIGNED BINARY NUMBERS

  - 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	0 0 0 0 0 1 1 0	- 6	1 1 1 1 1 0 1 0
+13	+ 0 0 0 0 1 1 0 1	+13	+ 0 0 0 0 1 1 0 1
+19	0 0 0 1 0 0 1 1	+ 7	0 0 0 0 0 1 1 1

SIGNED BINARY NUMBERS

- Example:

+ 6	0 0 0 0 0 1 1 0	- 6	1 1 1 1 1 0 1 0
- 13	+ 1 1 1 1 0 0 1 1	- 13	+ 1 1 1 1 0 0 1 1
- 7	1 1 1 1 1 0 0 1	- 19	1 1 1 0 1 1 0 1

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.

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

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

SIGNED BINARY NUMBERS

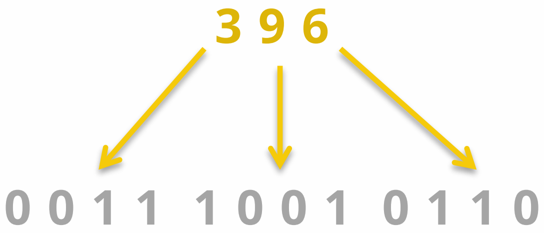
- Example:
    - (-6) – (-13) → (11111010 – 11110011)

-		6		1	1	1	1	1	0	1	0
+	1	3	-	0	0	0	0	1	1	0	1
+		7		0	0	0	0	0	1	1	1

1.8 BINARY CODES

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:



3 9 6

0 0 1 1 1 0 0 1 0 1 1 0

  - Each group of 4 bits representing one decimal digit.

BINARY CODES

- 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.

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 CODES

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

$$(185)_{10}$$

$$(0001\ 1000\ 0101)_{BCD}$$

$$=$$

$$(10111001)_2$$

BINARY CODES

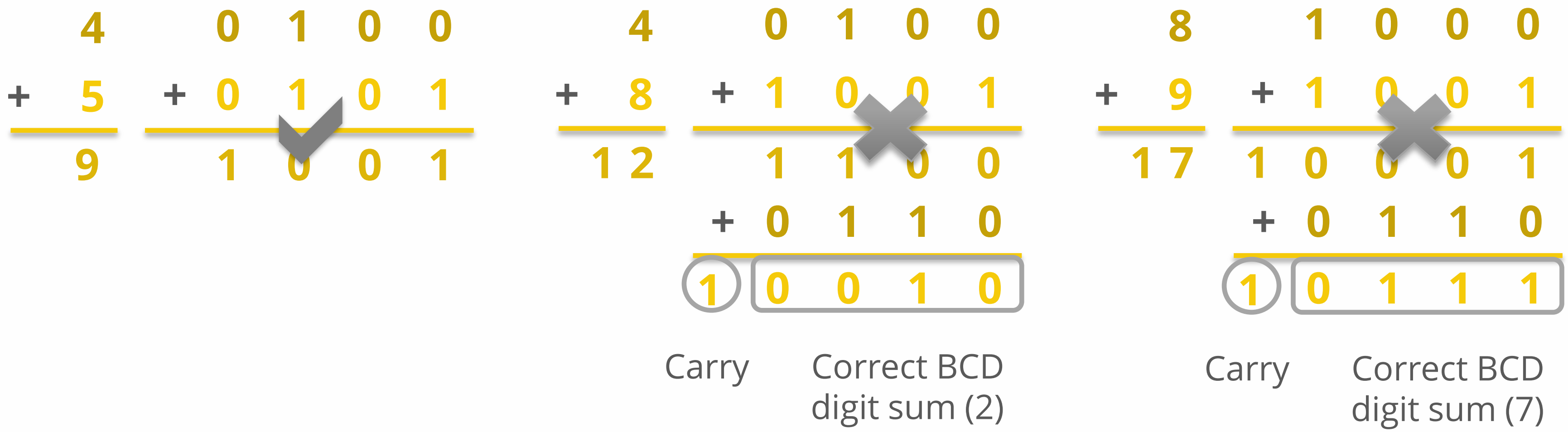
- BCD Addition
  - Consider the addition of two decimal digits in BCD
  - Since each digit does not exceed 9
  - The sum cannot be greater than 9 + 9 + 1 = 19
    - the 1 in the sum being a previous carry
  - Suppose we add the BCD digits as if they were binary numbers
  - The binary sum will produce a result in the range from 0 to 19
  - In binary this will be from 0000 to 10011, but in BCD it is from 0000 to 1 1001
    - The first 1 being a carry and the next four bits being the BCD digit sum

BINARY CODES

- BCD Addition
  - When binary sum is equal to or less than 1001 (without a carry)
    - The corresponding BCD digit is correct
  - When the binary sum is greater than or equal to 1010
    - The result is an invalid BCD digit
  - The addition of 6 = $(0110)_2$ to the binary sum converts it to the correct digit and also produces a carry as required
  - This is because the difference between a carry in the most significant bit position of the binary sum and a decimal carry.
    - 16 – 10 = 6

BINARY CODES

- BCD Addition
  - Consider the following three addition of two decimal digits in BCD



$$\begin{array}{rr} & 4 \\ + & 5 \\ \hline & 9 \end{array} \qquad \begin{array}{rrrrr} & 0 & 1 & 0 & 0 \\ + & 0 & 1 & 0 & 1 \\ \hline & 1 & 0 & 0 & 1 \end{array} \checkmark$$

$$\begin{array}{rr} & 4 \\ + & 8 \\ \hline & 12 \end{array} \qquad \begin{array}{rrrrr} & 0 & 1 & 0 & 0 \\ + & 1 & 0 & 0 & 1 \\ \hline & 1 & 1 & 0 & 0 \\ + & 0 & 1 & 1 & 0 \\ \hline \textcircled{1} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} \end{array} \times$$

Carry

Correct BCD digit sum (2)

$$\begin{array}{rr} & 8 \\ + & 9 \\ \hline & 17 \end{array} \qquad \begin{array}{rrrrr} & 1 & 0 & 0 & 0 \\ + & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 \\ + & 0 & 1 & 1 & 0 \\ \hline \textcircled{1} & \boxed{0} & \boxed{1} & \boxed{1} & \boxed{1} \end{array} \times$$

Carry

Correct BCD digit sum (7)

BINARY CODES

- The addition of two n-digit unsigned BCD numbers follows the same procedure.
  - Consider the addition of 184 + 576 = 760 in BCD

					1					1								
		0	0	0	1		1	0	0	0		0	1	0	0		1	8	4
	+	0	1	0	1	+	0	1	1	1	+	0	1	1	0	+	5	7	6
Binary Sum		0	1	1	1	1	0	0	0	0		1	0	1	0				
Add 6						+	0	1	1	0	+	0	1	1	0				
BCD Sum		0	1	1	1		0	1	1	0		0	0	0	0		7	6	0

BINARY CODES

- The representation of signed decimal numbers in BCD is similar to the representation of signed numbers in binary.
- We can use either the familiar sign and magnitude system or the signed-complement system.
- The sign of a decimal number is usually represented with four bits to conform to the 4-bit code of the decimal digits.
  - "**+**" with **0 0 0 0** and "**-**" with **1001** (BCD equivalent of 9)

BINARY CODES

- The signed-complement system can be either the 9’s or the 10’s complement.
  - But the 10’s complement is the one most often used.
- To obtain the 10’s complement of a BCD number:
  - First take the 9’s complement
    - 9’s complement is calculated from the subtraction of each digit from 9.
  - Then add one to the least significant digit

BINARY CODES

- The procedures developed for the signed-2's complement system in the previous section apply also to the signed-10's complement system for decimal numbers.
- Addition is done by:
  - adding all digits,
  - including the sign digit and
  - discarding the end carry.
- This assumes that all negative numbers are in 10's complement form.

BINARY CODES

- Consider the addition (+375) + (-240) = +135
- Step1: Find the 10’s complement of (-240)

$$\begin{array}{rrrr} & 9 & 10 & 0 \\ - & 2 & 4 & 0 \\ \hline & 7 & 6 & 0 \end{array}$$

- Step 2: Add 10’s complement of (-240) to 375 and discard the end carry.

$$\begin{array}{rrr} & 0 & 375 \\ + & 9 & 760 \\ \hline & 0 & 135 \end{array}$$

BINARY CODES

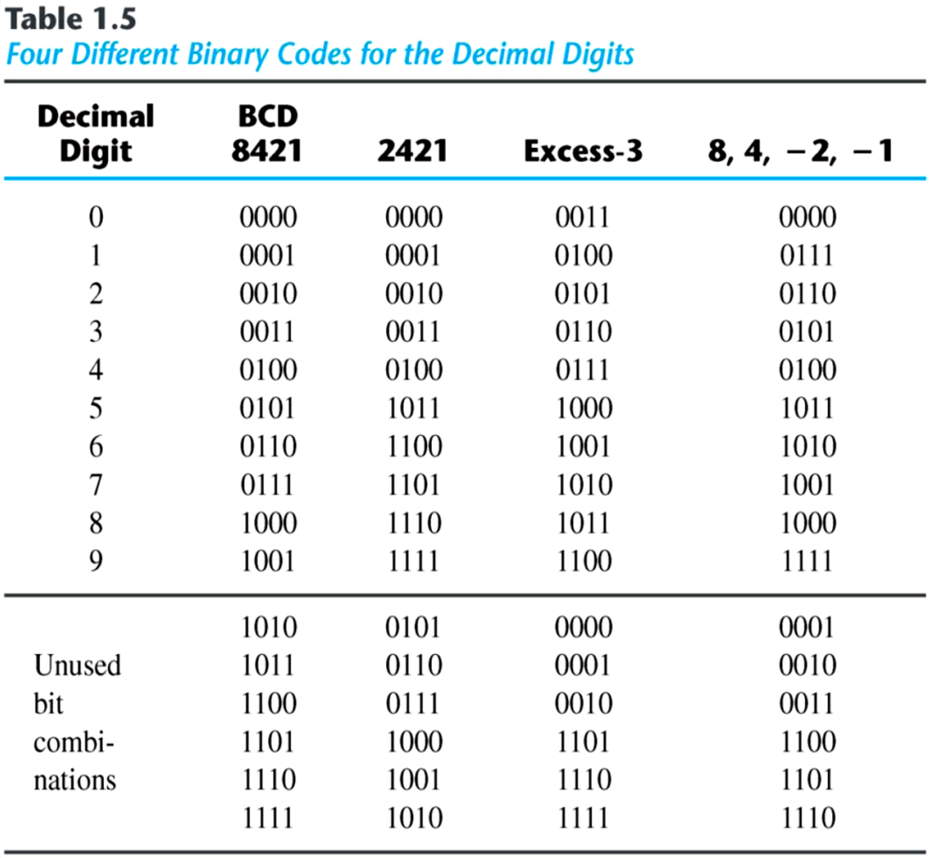


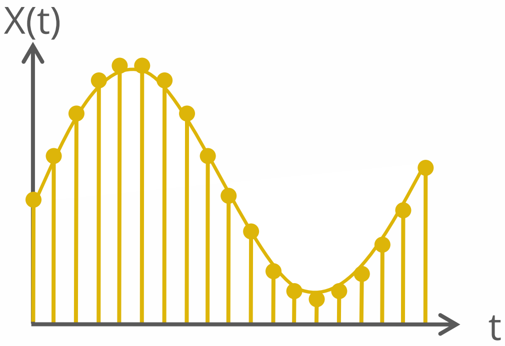
Table 1.5

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
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

BINARY CODES

- Gray code:
  - The output data of many physical systems produce quantities that are continuous.
  - These data must be converted into digital form before they are applied to a digital system.
  - Continuous or analog information is converted into digital form by means of an analog-to-digital converter.



BINARY CODES

- Gray code:
  - It is convenient to use gray code to represent the digital data when it is converted from analog data.
  - 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.

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

- American Standard Code for Information Interchange (ASCII) Character Code (Refer to Table 1.7)
- A popular code used to represent information sent as character-based data.
- It uses 7-bits to represent 128 characters:
  - 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).

BINARY CODES

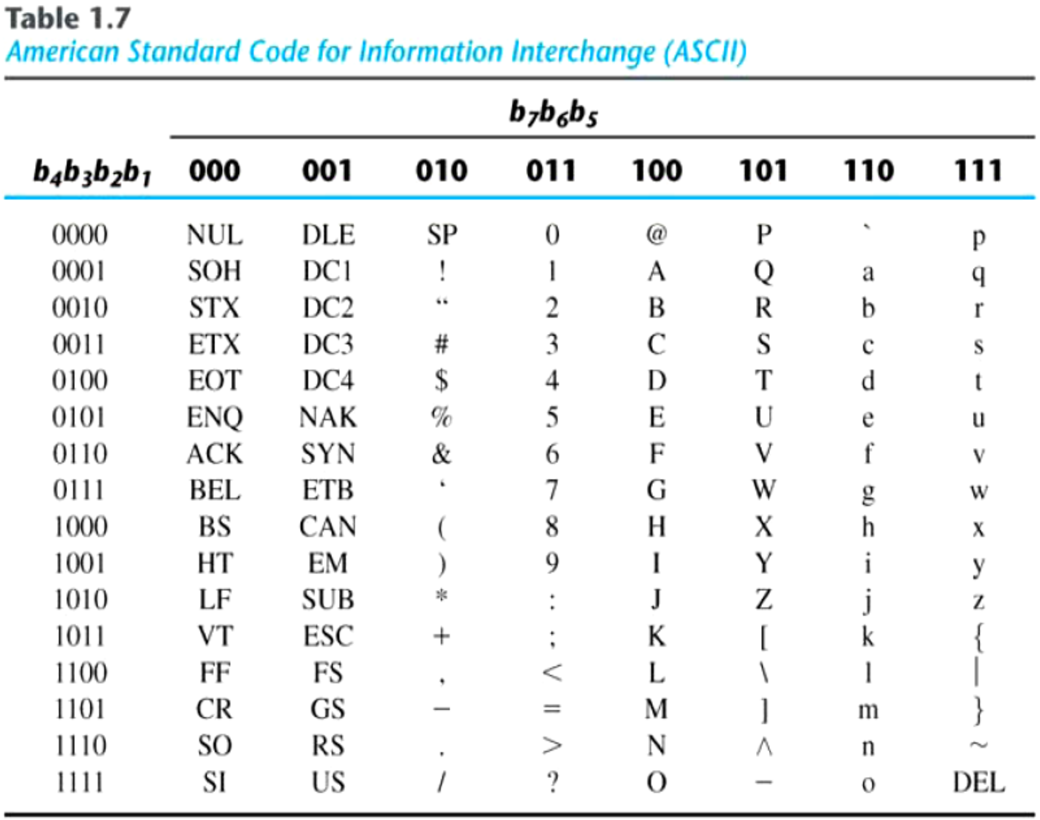
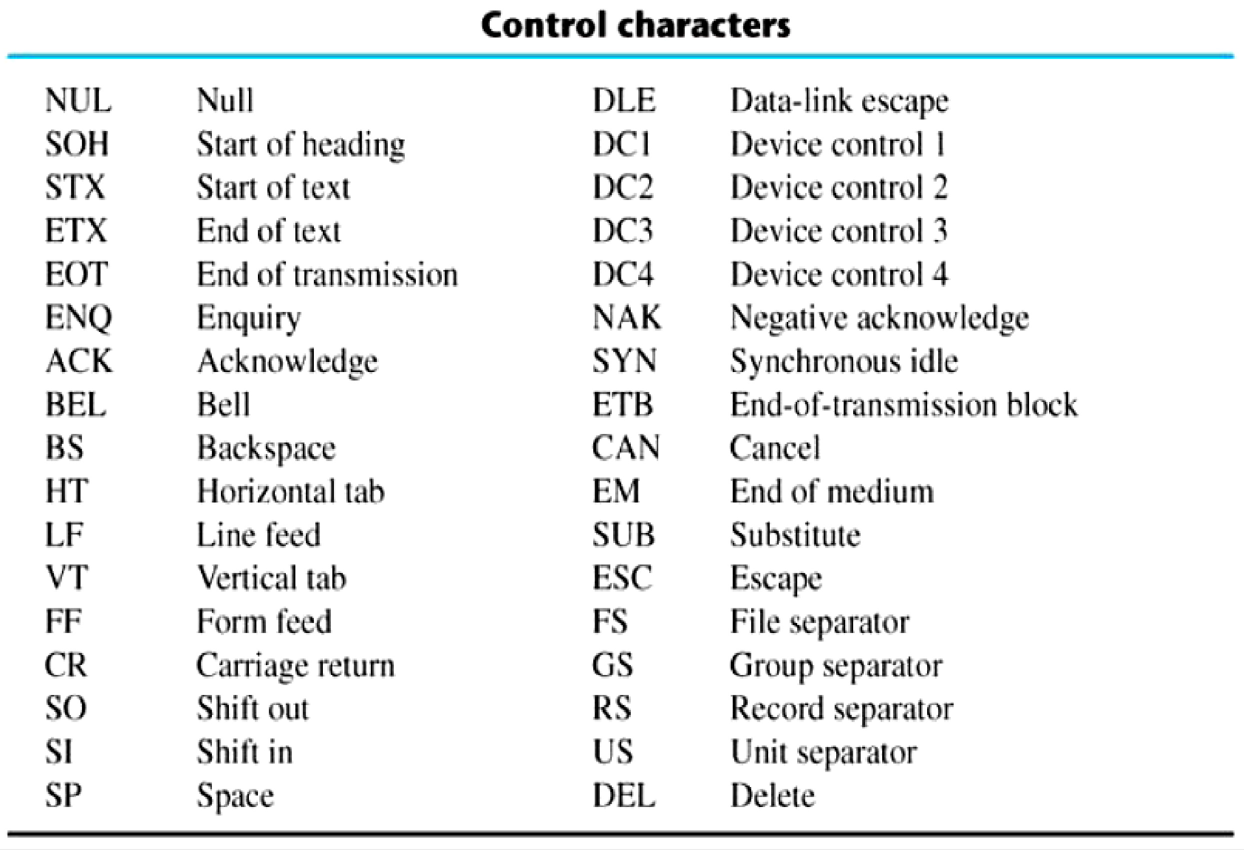


Table 1.7

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



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

BINARY CODES

- 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

- 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: 10001001 1 (even parity)
    - Message B: 10001001 0 (odd parity)

1.9 BINARY STORAGE & REGISTERS

BINARY STORAGE & REGISTERS

- Registers
  - A **binary cell** is a device that possesses two stable states and is capable of storing one of the two states.

1/0

  - A **register** is a group of binary cells. A register with *n* cells can store any discrete quantity of information that contains *n* bits.

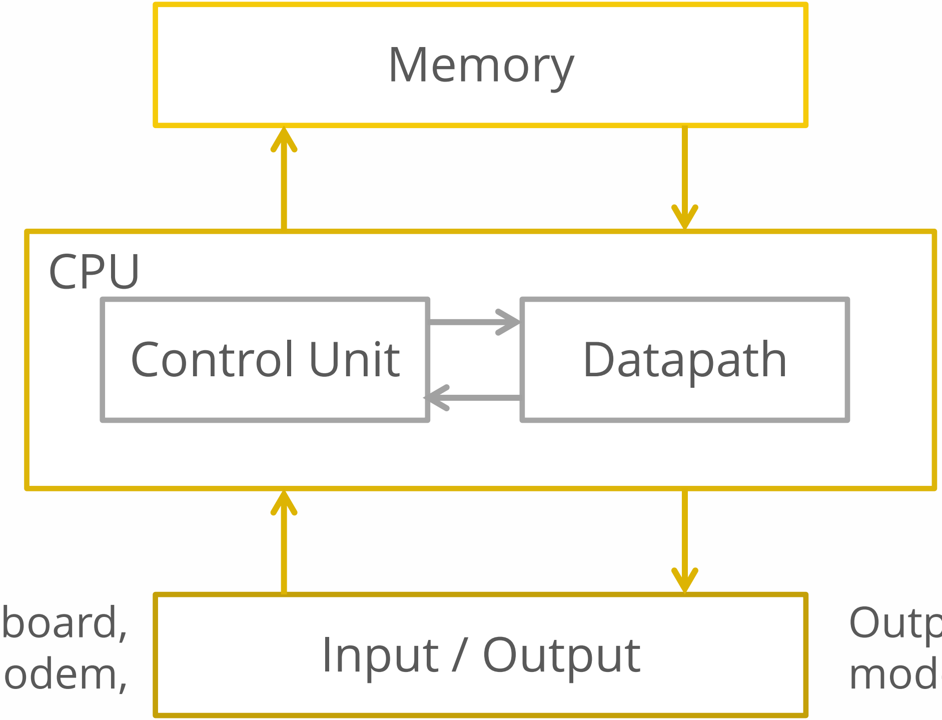
0 1 1 0 1 0 1 1

n cells → 2^n possible states

BINARY STORAGE & REGISTERS

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

BINARY STORAGE & REGISTERS



Memory

CPU

Control Unit

Datapath

Input / Output

Inputs: Keyboard, mouse, modem, microphone

Outputs: CRT, LCD, modem, speakers

Synchronous or Asynchronous?

BINARY STORAGE & REGISTER

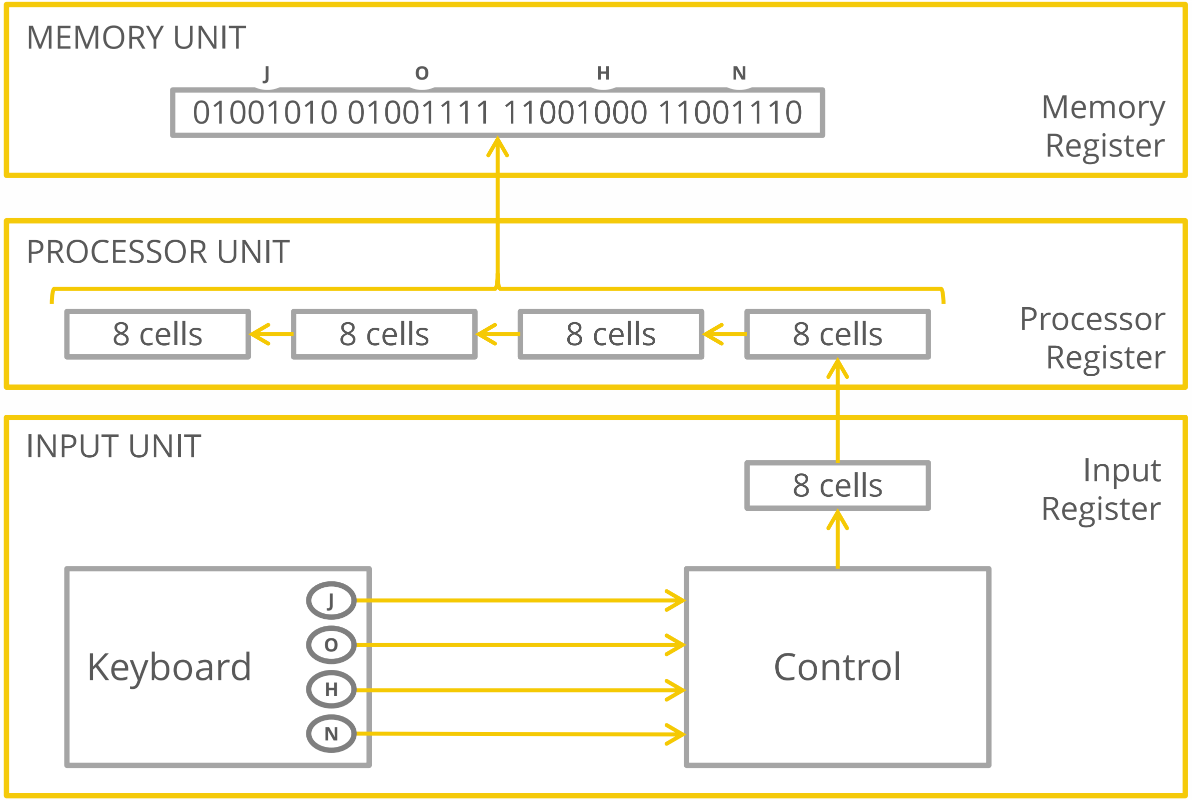


Figure 1.1 Transfer of information among register

BINARY STORAGE & REGISTERS

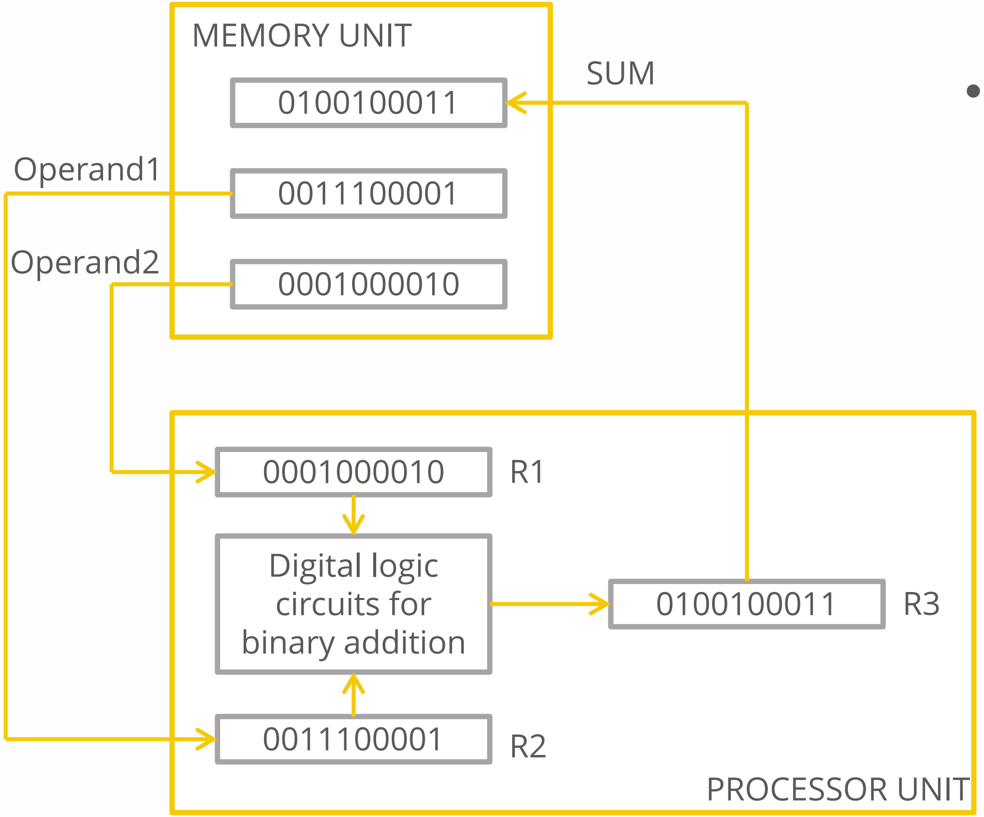
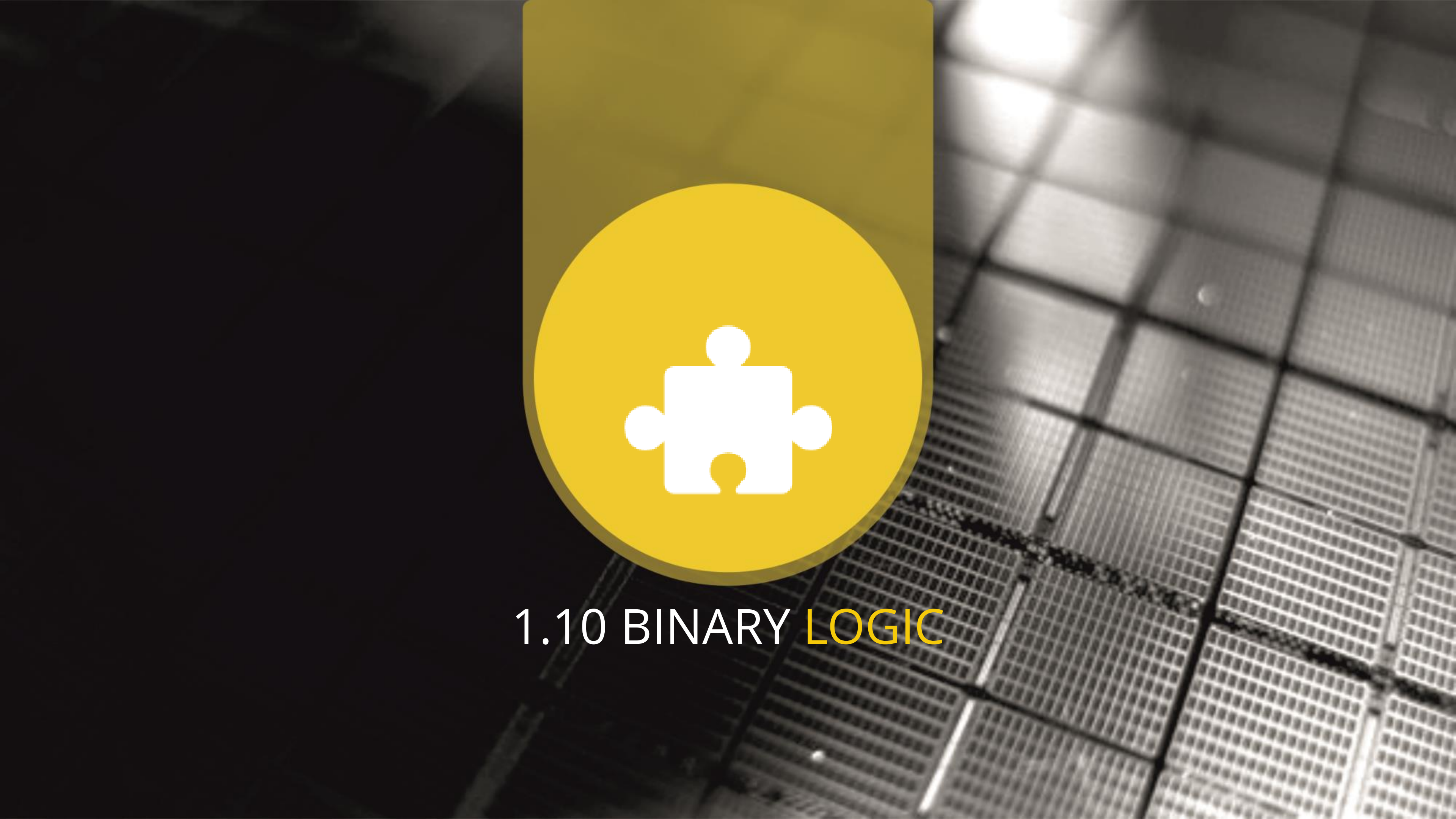


Figure 1.2 Example of binary information processing

- 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;



1.10 BINARY LOGIC

BINARY LOGIC

- 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.

BINARY LOGIC

1. AND operation is represented by a dot (·) or by the absence of an operator.
    - Example: **X · Y** = **Z** or **X Y** = **Z**
        - "**X AND Y** is equal to **Z**".
        - **Z** =**1** if only **X** = **1** and **Y** = **1**; otherwise **Z** = **0**.
            - **X**, **Y** and **Z** are binary variables and can be equal either to **1** or **0**, nothing else.

BINARY LOGIC

2. OR operation is represented by a plus (**+**).
   - Example: **X + Y** = **Z**
     - "**X OR Y** is equal to **Z**".
     - **Z** =**1** if **X** = **1** OR **Y** = **1** or if both **X** = **1** OR **Y** = **1.**
     - If both **X** = **0** OR **Y** = **0**, then **Z** =**0**.

BINARY LOGIC

3. NOT operation is represented by a prim (’) sometimes by an overbar (¯)
   - Example: **X’** = **Z** or $\overline{X}$ = **Z**
     - “**NOT X** is equal to **Z**”.
     - 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**.

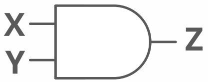
BINARY LOGIC

- Truth tables, Boolean expressions and Logic Gates

AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

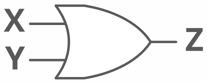
$$Z = X \cdot Y$$



OR

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

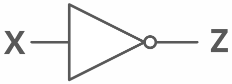
$$Z = X + Y$$



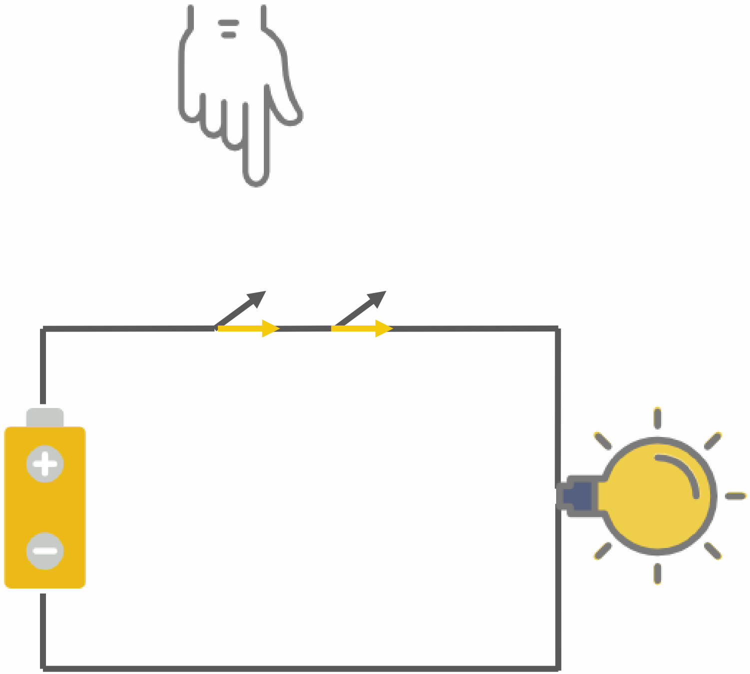
NOT

X	Z
1	0
0	1

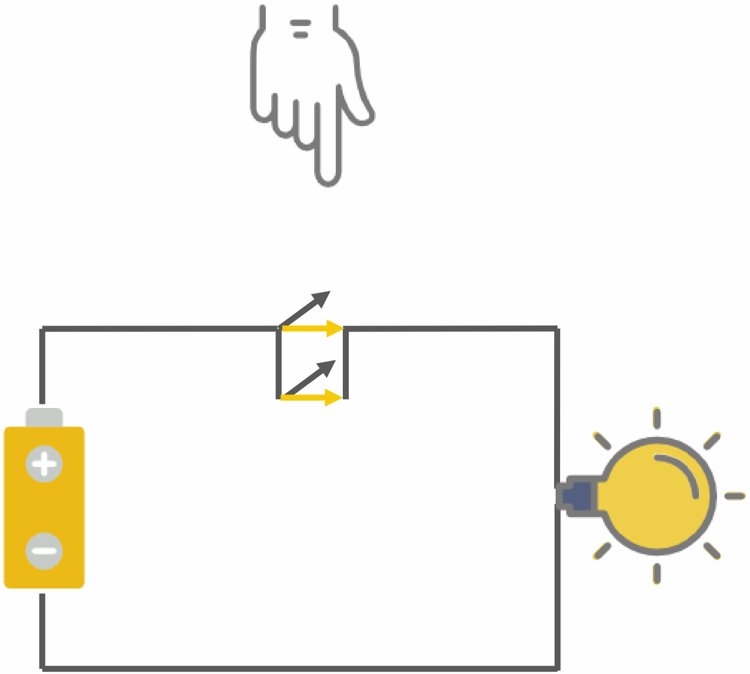
$$\overline{X} = Z$$



BINARY LOGIC



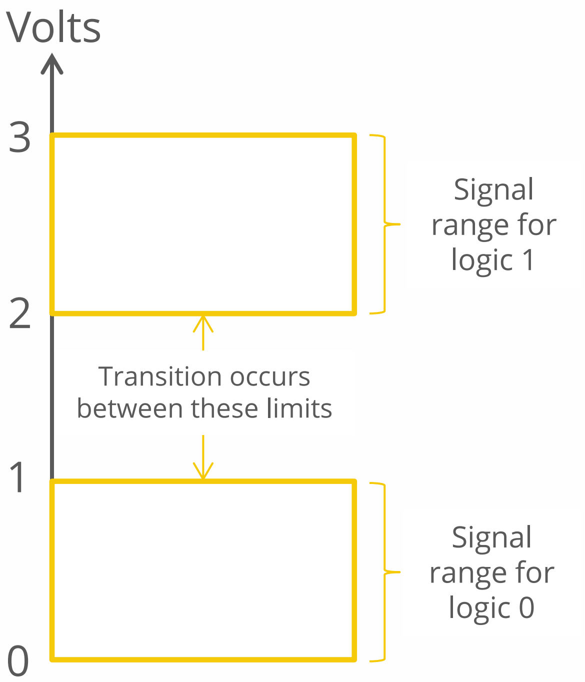
AND



OR

BINARY LOGIC

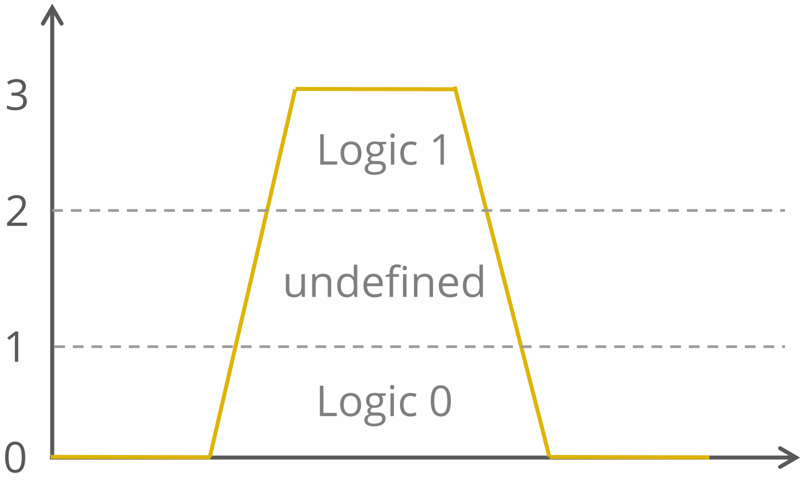
- Example of binary signals



Signal range for logic 1

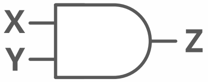
Transition occurs between these limits

Signal range for logic 0

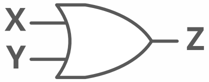


BINARY LOGIC

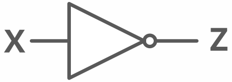
- Graphic Symbols and Input-Output Signals for Logic gates:



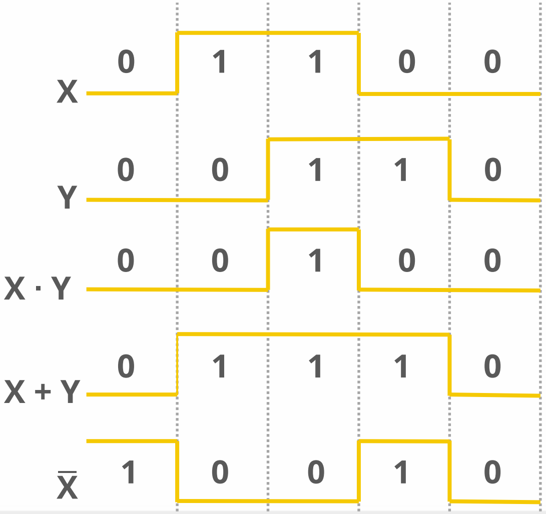
(a) two input AND gate



(b) two input OR gate

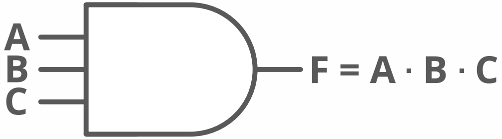


(c) NOT gate or Inverter

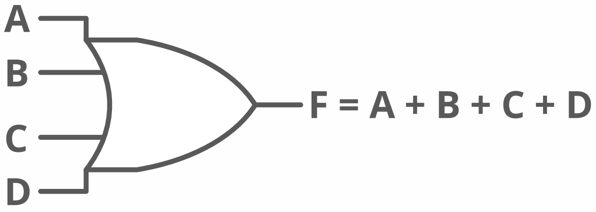


BINARY LOGIC

- Graphic Symbols and Input-Output Signals for many input logic gates:



(a) three input AND gate



(b) four input OR gate