# Digital Systems and Binary Numbers

COE 202

Digital Logic Design

Dr. Abdulaziz Tabbakh

College of Computer Sciences and Engineering

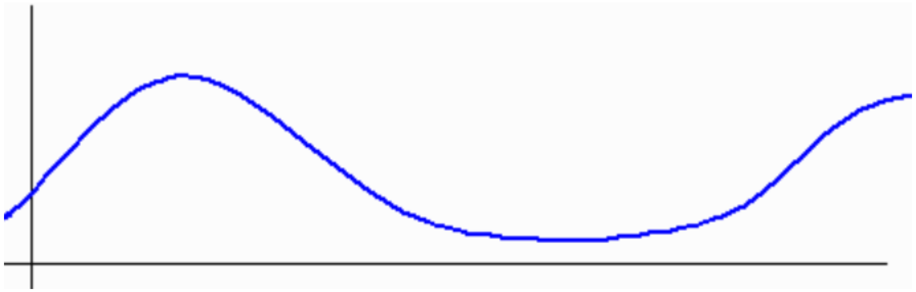
King Fahd University of Petroleum and Minerals

# Outline

- Introduction
- Numbering Systems
- Weighted Number Systems
- Binary, Octal & Hexadecimal Systems
- Number Base Conversions
- Binary Addition, Subtraction, Multiplication
- Binary Codes and Binary Logic

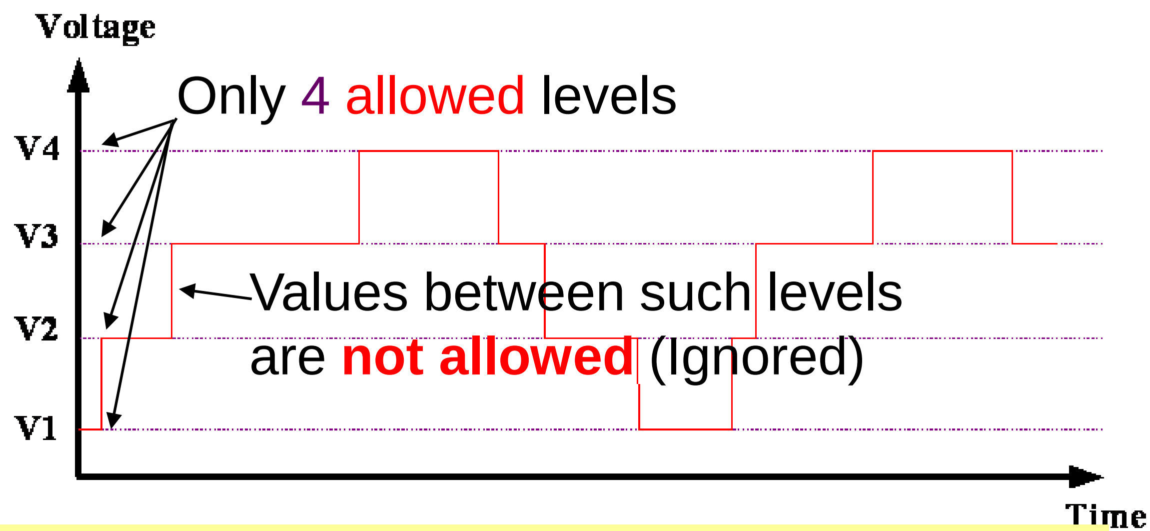
# Introduction

- The natural world around us is predominantly Analog
- Analog means Continuous (both in time and amplitude)
  - Value changes smoothly over time
  - Have a continuous (infinite) range of amplitudes (Values)
- Examples:
  - Sound
  - Temperature
  - Speed
  - Voltage/Current



# Why Digital ?

- ❖ Digital systems are everywhere
- ❖ Finite (discrete) number of possible states (values)
  - ✧ Can present and manipulate information and elements
  - ✧ Dealing with finite states is easier than a infinite number of states
- ❖ Examples:
  - ✧ Alphabet
  - ✧ Playing cards
  - ✧ Chessboard




**Special Case of Digital: only two signal levels → Binary signal**

# Signals in Digital Systems

- Information is represented in digital systems by signals
- Most common types of signals are currents and voltages
- Signals in digital systems can have two possible values:
  - 0 (OFF)
  - 1 (ON)
- These digital systems are called binary systems
- Each digit is called bit and a group of bits is called binary code

# Numbering Systems

- Numbering systems are characterized by their base (radix) number.
- In general a numbering system with a base *r* will have *r* different digits (including the 0) in its number set. These digits will range from *0 to r-1.*
- The most widely used numbering systems are listed in the table below:

| Numbering System | Base | Digits Set |
| --- | --- | --- |
| Binary | 2 | 1 0 |
| Octal | 8 | 7 6 5 4 3 2 1 0 |
| Decimal | 10 | 9 8 7 6 5 4 3 2 1 0 |
| Hexadecimal | 16 | F E D C B A 9 8 7 6 5 4 3 2 1 0 |

# Weighted Number Systems

- A number D consists of *n* digits with each digit having a particular *position*.

$$D = d_{n-1}\ d_{n-2}\ \ldots\ldots\ d_2\ d_1\ d_0$$

*Position n-1* | *Position 2* | *Position 1* | *Position 0*

- Every digit *position* is associated with a *fixed weight*.
- If the weight associated with the *ith* position is $w_i$, then the value of D is given by:

$$D = d_{n-1}\, w_{n-1} + d_{n-2}\, w_{n-2} + \ldots + d_2\, w_2 + d_1\, w_1 + d_0\, w_0$$

# Example of Weighted Number Systems

- The Decimal number system is a weighted system.
- For integer decimal numbers, the weight of the rightmost digit (*at position 0*) *is* 1, the weight of *position 1* digit is 10, that of *position 2* digit is 100, *position 3* is 1000, etc.
- Thus, $w_0 = 1$, $w_1 = 10$, $w_2=100$, $w_3 = 1000$, etc.
- Example:
- Show how the value of the decimal number 9375 is estimated.

First Position Index

First Position Index (0)

| Position | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Number | 9 | 3 | 7 | 5 |
| Weight | 1000 | 100 | 10 | 1 |
| Value | 9 x 1000 | 3x100 | 7x10 | 5x1 |
| Value | 9000 + | 300 + | 70 + | 5 |

# The Radix (Base)

- For *digit position i*, most weighted number systems use weights ($w_i$) that are powers of some constant value called the radix (*r*) or the base such that $w_i = r^i$.
- A number system of radix *r*, typically has a set of r allowed digits $\in \{0,1, \ldots,(r-1)\}$.
- The leftmost digit has the highest weight → Most Significant Digit (MSD).
- The rightmost digit has the lowest weight → Least Significant Digit (LSD).

# The Radix (Base)

- Example: Decimal Number System
- 1. Radix (Base) = *Ten*
- 2. Since $w_i = r^i$, *then*
  - $w_0 = 10^0 = 1,$
  - $w_1 = 10^1 = 10,$
  - $w_2 = 10^2 = 100,$
  - $w_3 = 10^3 = 1000$, *etc.*
- 3. Number of Allowed Digits is Ten:
  - {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

MSD LSD

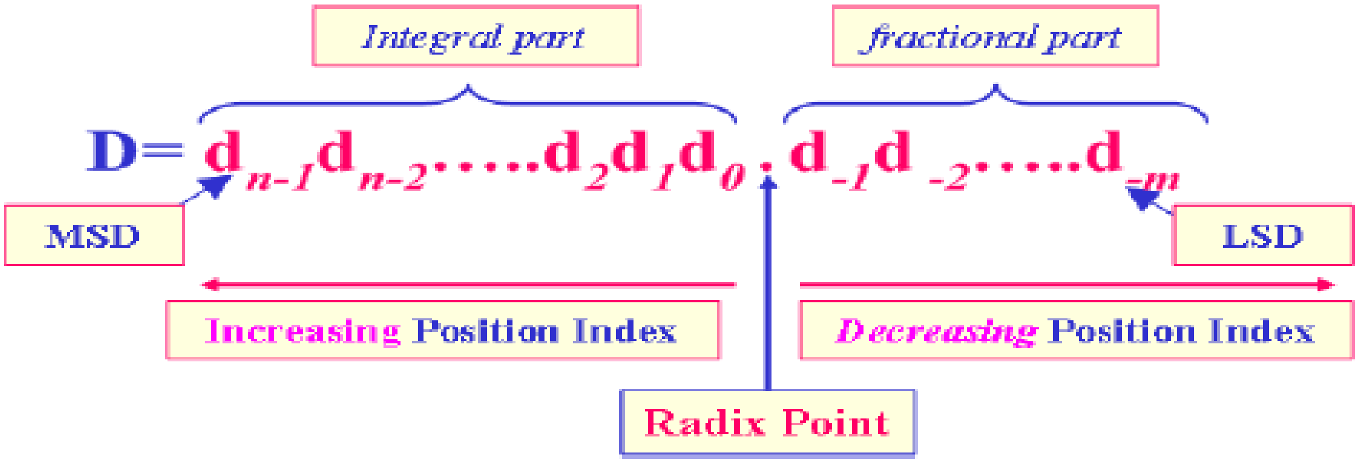
$$9375 = 5\text{x}10^0 + 7\text{x}10^1 + 3\text{x}10^2 + 9\text{x}10^3$$

$$= 5\text{x}1 + 7\text{x}10 + 3\text{x}100 + 9\text{x}1000$$

| Position | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Weight | 1000 $= 10^3$ | 100 $= 10^2$ | 10 $= 10^1$ | 1 $= 10^0$ |

# The Radix Point

❖ A number D of *n* integral digits and *m* fractional digits is represented as shown:




❖ Digits to the left of the radix point (*integral digits) have positive* position indices, while digits to the right of the radix point (*fractional digits*) *have negative position indices.*

# The Radix Point

- Position indices of digits to the left of the radix point (the integral part of D) start with a 0 and are incremented as we move left ($d_{n-1}d_{n-2}.....d_2d_1d_0$).
- Position indices of digits to the right of the radix point (the fractional part of D) start with a -1 and are decremented as we move right($d_{-1}d_{-2}.....d_{-m}$).
- The weight associated with digit *position i* is given by $w_i = r^i$, where *i* is the position index $\forall i = -m, -m+1, \ldots, -2, -1, 0, 1, \ldots\ldots, n-1$.
- The Value of D is Computed as:

$$D = \sum_{i=-m}^{n-1} d_i \, r^i$$

# The Radix Point

❖ Example: Show how the value of the decimal number 52.946 is estimated.

$$D = 5\ 2\ .\ 9\ 4\ 6$$

$d_1$ $d_0$ . $d_{-1}$ $d_{-2}$ $d_{-3}$

| Number | 5 | 2 | . | 9 | 4 | 6 |
|---|---|---|---|---|---|---|
| Position | 1 | 0 | . | -1 | -2 | -3 |
| Weight | $10^1 = 10$ | $10^0 = 1$ | . | $10^{-1} = 0.1$ | $10^{-2} = 0.01$ | $10^{-3} = 0.001$ |
| Value | 5 x 10 | 2 x 1 | . | 9 x 0.1 | 4 x 0.01 | 6 x 0.001 |
| Value | 50 + | 2 + | | 0.9 + | 0.04 + | 0.006 |

$$D = 5x10^1 + 2x10^0 + 9x10^{-1} + 4x10^{-2} + 6x10^{-3}$$

# Notation

- Let $(D)_r$ denote a number D expressed in a number system of radix r.
- In this notation, r will be expressed in decimal.
- Examples:
- $(29)_{10}$ Represents a decimal value of 29. The radix “10” here means ten.
- $(100)_{16}$ is a Hexadecimal number since r = “16” here means sixteen. This number is equivalent to a decimal value of $16^2=256$.
- $(100)_2$ is a Binary number (radix =2, i.e. two) which is equivalent to a decimal value of $2^2 = 4$.

# Binary System

- r=2
- Each digit (bit) is either 1 or 0
- Every binary number is a sum of powers of 2

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

$$\mathbf{D} = \mathbf{d}_{n-1}\,\mathbf{w}_{n-1} + \mathbf{d}_{n-2}\,\mathbf{w}_{n-2} + \ldots + \mathbf{d}_2\,\mathbf{w}_2 + \mathbf{d}_1\,\mathbf{w}_1 + \mathbf{d}_0\,\mathbf{w}_0$$

**Table 1-3** Binary Bit Position Values.

| $2^n$ | Decimal Value | $2^n$ | Decimal Value |
|---|---|---|---|
| $2^0$ | 1 | $2^8$ | 256 |
| $2^1$ | 2 | $2^9$ | 512 |
| $2^2$ | 4 | $2^{10}$ | 1024 |
| $2^3$ | 8 | $2^{11}$ | 2048 |
| $2^4$ | 16 | $2^{12}$ | 4096 |
| $2^5$ | 32 | $2^{13}$ | 8192 |
| $2^6$ | 64 | $2^{14}$ | 16384 |
| $2^7$ | 128 | $2^{15}$ | 32768 |

# Binary System

❖ Examples: Find the decimal value of the two Binary numbers $(101)_2$ and $(1.101)_2$

MSB LSB

$$(1\ 0\ 1)_2 = 1x2^0 + 0x2^1 + 1x2^2$$
$$= 1x1 + 0x2 + 1x4$$
$$= (5)_{10}$$

MSB LSB

$$(1\ .\ 1\ 0\ 1)_2 = 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$$
$$= 1 + 0.5 + 0 + 0.125$$
$$= (1\ .\ 6\ 2\ 5)_{10}$$

# Octal System

- $r = 8$ (Eight = $2^3$ )
- Eight allowed digits {0, 1, 2, 3, 4, 5, 6, 7}
- Examples: Find the decimal value of the two Octal numbers $(375)_8$ and $(2.746)_8$

$$\underbrace{(\,3}_{\text{MSD}}7\underbrace{5}_{\text{LSD}}\,)_8 = 5x8^0 + 7x8^1 + 3x8^2$$

$$= 5x1 + 7x8 + 3x64$$

$$= (\,253\,)_{10}$$

$$(\underbrace{2}_{\text{MSD}}.74\underbrace{6}_{\text{LSD}}\,)_8 = 2x8^0 + 7x8^{-1} + 4x8^{-2} + 6x8^{-3}$$

$$= (2.94921875\,)_{10}$$

# Hexadecimal System

- r = 16 (Sixteen = $2^4$)
- Sixteen allowed digits {0-to-9 and A, B, C, D, E, F}
- Where: A = Ten, B = Eleven, C = Twelve, D = Thirteen, E = Fourteen & F = Fifteen.
- Examples: Find the decimal value of the two Hexadecimal numbers $(9E1)_{16}$ and $(3B.C)_{16}$

MSD LSD

$$
\begin{aligned}
(9E1)_{16} &= 1\text{x}16^0 + \text{E}\text{x}16^1 + 9\text{x}16^2 \\
&= 1\text{x}1 + 14\text{x}16 + 9\text{x}256 \\
&= (2529)_{10}
\end{aligned}
$$

MSD LSD

$$
\begin{aligned}
(3B.C)_{16} &= \text{C}\text{x}16^{-1} + \text{B}\text{x}16^0 + 3\text{x}16^1 \\
&= 12\text{x}16^{-1} + 11\text{x}16^0 + 3\text{x}16 \\
&= (59.75)_{10}
\end{aligned}
$$

# Binary, Octal, and Hexadecimal

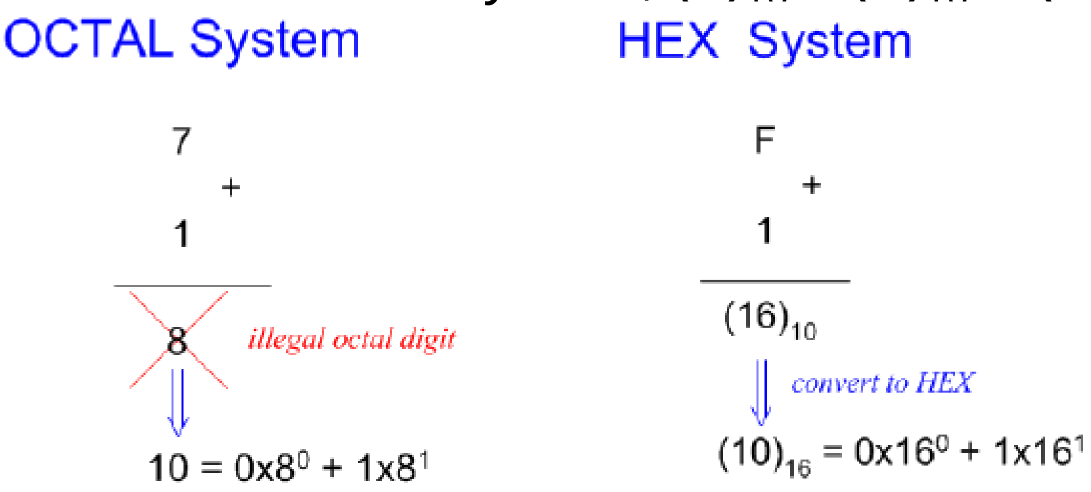
| Decimal | Binary | Octal | Hexa decimal | Decimal | Binary | Octal | Hexa decimal |
|---|---|---|---|---|---|---|---|
| 0 | 0000 | 00 | 0 | 8 | 1000 | 10 | 8 |
| 1 | 0001 | 01 | 1 | 9 | 1001 | 11 | 9 |
| 2 | 0010 | 02 | 2 | 10 | 1010 | 12 | A |
| 3 | 0011 | 03 | 3 | 11 | 1011 | 13 | B |
| 4 | 0100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

# Important Properties

- The Largest value that can be expressed in n integral digits is ($r^n$-1).
- The Largest value that can be expressed in m fractional digits is (1-$r^{-m}$).
- The Largest value that can be expressed in n integral digits and m fractional digits is ($r^n - r^{-m}$)
- Total number of values (patterns) representable in n digits is $r^n$.

# Important Properties

- Q. What is the result of adding 1 to the largest digit of some number system??
  - For the decimal number system, $(1)_{10} + (9)_{10} = (10)_{10}$
  - For the binary number system, $(1)_2 + (1)_2 = (10)_2 = (2)_{10}$
  - For the octal number system, $(1)_8 + (7)_8 = (10)_8 = (8)_{10}$
  - For the hexadecimal system, $(1)_{16} + (F)_{16} = (10)_{16} = (16)_{10}$

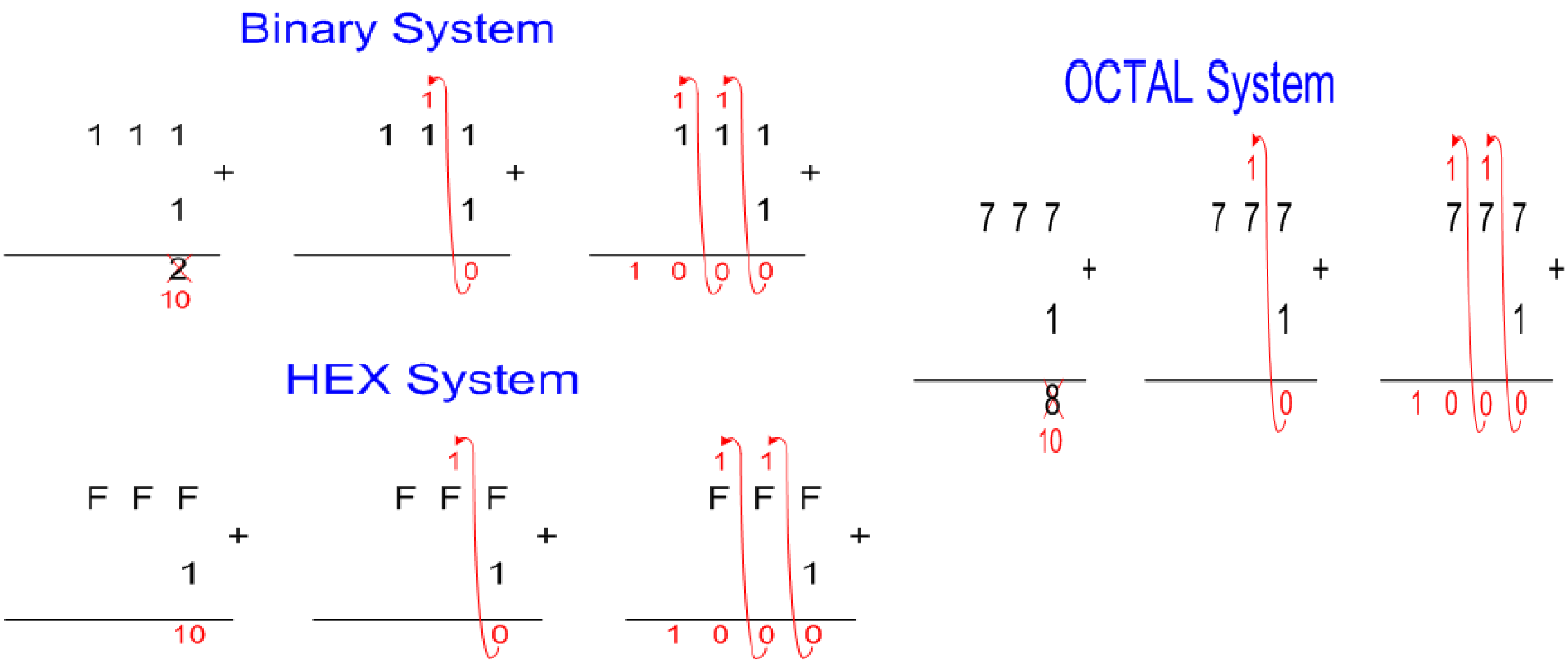


# Important Properties

- Q. What is the largest value representable in 3-integral digits?
- A. The largest value results when all 3 positions are filled with the largest digit in the number system.
  - For the decimal system, it is $(999)_{10}$
  - For the octal system, it is $(777)_8$
  - For the hex system, it is $(FFF)_{16}$
  - For the binary system, it is $(111)_2$
- Q. What is the result of adding 1 to the largest 3-digit number?
  - For the decimal system, $(1)_{10} + (999)_{10} = (1000)_{10} = (10^3)_{10}$
  - For the octal system, $(1)_8 + (777)_8 = (1000)_8 = (8^3)_{10}$

# Important Properties

- In general, for a number system of radix r, adding 1 to the largest *n-digit* number = $r^n$.
- Accordingly, the value of largest *n-digit number* = $r^n - 1$.



# Number Base Conversion

- Given the representation of some number $(X_B)$ in a number system of radix $B$, we need to obtain the representation of the same number in another number system of radix $A$, i.e. $(X_A)$.
- For a number that has both integral and fractional parts, conversion is done separately for both parts, and then the result is put together with a system point in between both parts.
- Converting Whole (Integer) Numbers
  - Assume that $X_B$ has $n$ digits $(b_{n-1} \ldots\ldots\ldots b_2\ b_1\ b_0)_B$, where $b_i$ is a digit in radix $B$ system, i.e. $b_i \in \{0, 1, \ldots.., \text{"B-1"}\}$.
  - Assume that $X_A$ has $m$ digits $(a_{m-1} \ldots\ldots\ldots a_2\ a_1\ a_0)_A$, where $a_i$ is a digit in radix $A$ system, i.e. $a_i \in \{0, 1, \ldots.., \text{"A-1"}\}$.

# Converting Whole (Integer) Numbers

❖ Dividing $X_B$ by A, the remainder will be $a_0$.

$$X_B = \overbrace{(b_{n-1}\ldots\ldots\ldots b_2\, b_1\, b_0)_B}^{\text{Knowns}} \Rightarrow \underbrace{(a_{m-1}\ldots\ldots\ldots a_2\, a_1\, a_0)_A}_{\text{Unknowns}}$$

$$X_B = \underbrace{a_{m-1}*A^{m-1}+\ldots\ldots+a_2*A^2+a_1*A^1}_{\text{Divisible by A}} + \underbrace{a_0*A^0}_{\text{Not Divisible by A}}$$

❖ In other words, we can write $X_B = Q_0.A+a_0$

# Converting Whole (Integer) Numbers

Where, $Q_0 = \underbrace{a_{m-1}*A^{m-2} + \ldots + a_2*A^1}_{\text{Divisible by A}} + \underbrace{a_1*A^0}_{\text{Not Divisible by A}}$

$\mathbf{Q_0 = Q_1 A + a_1}$

$\mathbf{Q_1 = Q_2 A + a_2}$

……………………………..

$\mathbf{Q_{m-3} = Q_{m-2} A + a_{m-2}}$

$\mathbf{Q_{m-2} = a_{m-1} < A}$ **(not divisible by A)**

$\mathbf{\quad = Q_{m-1} A + a_{m-1}}$

Where $\mathbf{Q_{m-1}} = 0$

# Converting Whole (Integer) Numbers

- This division procedure can be used to convert an integer value from some radix number system to any other radix number system.
- The first digit we get using the division process is $a_0$, then $a_1$, then $a_2$, till $a_{m-1}$
- Example: Convert $(53)_{10}$ to $(?)_2$

| Division Step | Quotient | Remainder | |
|---|---|---|---|
| 53 ÷ 2 | $Q_0 = 26$ | 1 = $a_0$ | LSB |
| 26 ÷ 2 | $Q_1 = 13$ | 0 = $a_1$ | |
| 13 ÷ 2 | $Q_2 = 6$ | 1 = $a_2$ | |
| 6 ÷ 2 | $Q_3 = 3$ | 0 = $a_3$ | |
| 3 ÷ 2 | $Q_4 = 1$ | 1 = $a_4$ | |
| 1 ÷ 2 | 0 | 1 = $a_5$ | MSB |

Stopping Point

Thus $(53)_{10} = (110101.)_2$

# Converting Whole (Integer) Numbers

❖ Since we always divide by the radix, and the quotient is re-divided again by the radix, the solution table may be compacted into 2 columns only as shown:

**Binary Point**

| | | | | | |
|---|---|---|---|---|---|
| 51 | ● | | 53 | ● | |
| 25 | 1 | $a_0$ | 26 | 1 | ← **LSB** |
| 12 | 1 | | 13 | 0 | |
| 6 | 0 | | 6 | 1 | |
| 3 | 0 | | 3 | 0 | |
| 1 | 1 | | 1 | 1 | |
| 0 | 1 | $a_m$ | 0 | 1 | ← **MSB** |

$$(51)_{10} = (110011.)_2$$

$$(53)_{10} = (110101.)_2$$

# Converting Whole (Integer) Numbers

❖ Example: Convert $(755)_{10}$ to $(?)_8$

| Division Step | Quotient | Remainder | |
|---|---|---|---|
| 755 ÷ 8 | $Q_0 = 94$ | $3 = a_0$ | LSB |
| 94 ÷ 8 | $Q_1 = 11$ | $6 = a_1$ | |
| 11 ÷ 8 | $Q_2 = 1$ | $3 = a_2$ | |
| 1 ÷ 8 | 0 | $1 = a_3$ | MSB |

| | |
|---|---|
| 755 | ● |
| 94 | 3 |
| 11 | 6 |
| 1 | 3 |
| 0 | 1 |

$(755)_{10} \Rightarrow (1363.)_8$

❖ Example: Convert $(1606)_{10}$ to $(?)_{12}$

| | | |
|---|---|---|
| 1606 ÷12 | ● | |
| 133 ÷12 | 10 = A | LSB |
| 11 ÷12 | 1 | |
| 0 | 11 = B | MSB |

For radix twelve, the allowed digit set is:

{0-9, A, B}

$(1606)_{10} \Rightarrow (B1A.)_{12}$

# Another Procedure for Converting from Decimal to Binary

- Start with a binary representation of all 0's
- Determine the highest possible power of two that is less or equal to the number.
- Put a 1 in the bit position corresponding to the highest power of two found above.
- Subtract the highest power of two found above from the number.
- Repeat the process for the remaining number

# Another Procedure for Converting from Decimal to Binary

- Example: Converting $(76)_{10}$ to Binary
  - The highest power of 2 less or equal to 76 is 64, hence the seventh (MSB) bit is 1

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | . | . | . | . | . | . |

  - Subtracting 64 from 76 we get 12.
  - The highest power of 2 less or equal to 12 is 8, hence the fourth bit position is 1

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | . | . | . |

  - We subtract 8 from 12 and get 4.
  - The highest power of 2 less or equal to 4 is 4, hence the third bit position is 1

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | . | . |

  - Subtracting 4 from 4 yield a zero, hence all the left bits are set to 0 to yield the final answer

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |

# Converting Binary to Decimal

❖ Weighted positional notation shows how to calculate the decimal value of each binary bit:

$$Decimal = (d_{n-1} \times 2^{n-1}) + (d_{n-2} \times 2^{n-2}) + ... + (d_1 \times 2^1) + (d_0 \times 2^0)$$

$d$ = binary digit

❖ binary 10101001 = decimal 169:

$(1 \times 2^7) + (1 \times 2^5) + (1 \times 2^3) + (1 \times 2^0) = 128+32+8+1=169$

# Binary to Octal Conversion

- Each octal digit corresponds to 3 binary bits.

$$(b_n \ldots\ldots b_5\ b_4\ b_3\ b_2\ b_1\ b_0 \,.\, b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}\ b_{-5} \ldots\ldots)_2 \Rightarrow (\ ?\ )_8$$

$$(b_n \ldots\ldots \underbrace{b_5\ b_4\ b_3}_{\text{3-bits}}\ \underbrace{b_2\ b_1\ b_0}_{\text{3-bits}} \,.\, \underbrace{b_{-1}\ b_{-2}\ b_{-3}}_{\text{3-bits}}\ \underbrace{b_{-4}\ b_{-5} \ldots}_{\text{3-bits}})_2$$

**Starting Point**

- Example: Convert $(1110010101.1011011)_2$ into Octal.

$$\underbrace{001}_{1}\_\_\underbrace{110}_{6}\_\_\underbrace{010}_{2}\_\_\underbrace{101}_{5}\_.\_\underbrace{101}_{5}\_\_\underbrace{101}_{5}\_\_\underbrace{100}_{4} = (1625.554)_8$$

# Binary to Hexadecimal Conversion

❖ Each hexadecimal digit corresponds to 4 binary bits.

$$(b_n \ldots\ldots b_5\, b_4\, b_3\, b_2\, b_1\, b_0\, .\, b_{-1}\, b_{-2}\, b_{-3}\, b_{-4}\, b_{-5} \ldots\ldots)_2 \Rightarrow (\ ?\ )_{16}$$

**Starting Point**

$$(b_n \overbrace{\ldots\ldots b_5\, b_4}^{4\text{-bits}}\, \overbrace{b_3\, b_2\, b_1\, b_0}^{4\text{-bits}}\, .\, \overbrace{b_{-1}\, b_{-2}\, b_{-3}\, b_{-4}}^{4\text{-bits}}\, b_{-5})_2$$

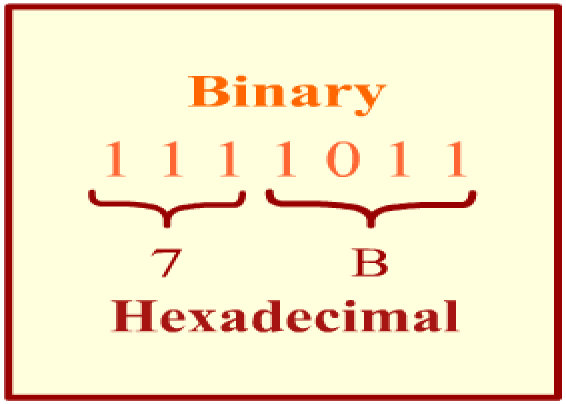
❖ Example: Convert $(1110010101.1011011)_2$ into hex.

$$\underbrace{0011}_{3}\_\_\underbrace{1001}_{9}\_\_\underbrace{0101}_{5}\_.\_\underbrace{1011}_{B}\_\_\underbrace{0110}_{6} = (395.B6)_{16}$$

# Binary to Hexadecimal Conversion

- Example: Translate the binary integer 000101101010011110010100 to hexadecimal

| | | | | | |
|---|---|---|---|---|---|
| 1 | 6 | A | 7 | 9 | 4 |
| 0001 | 0110 | 1010 | 0111 | 1001 | 0100 |




M1023.swf

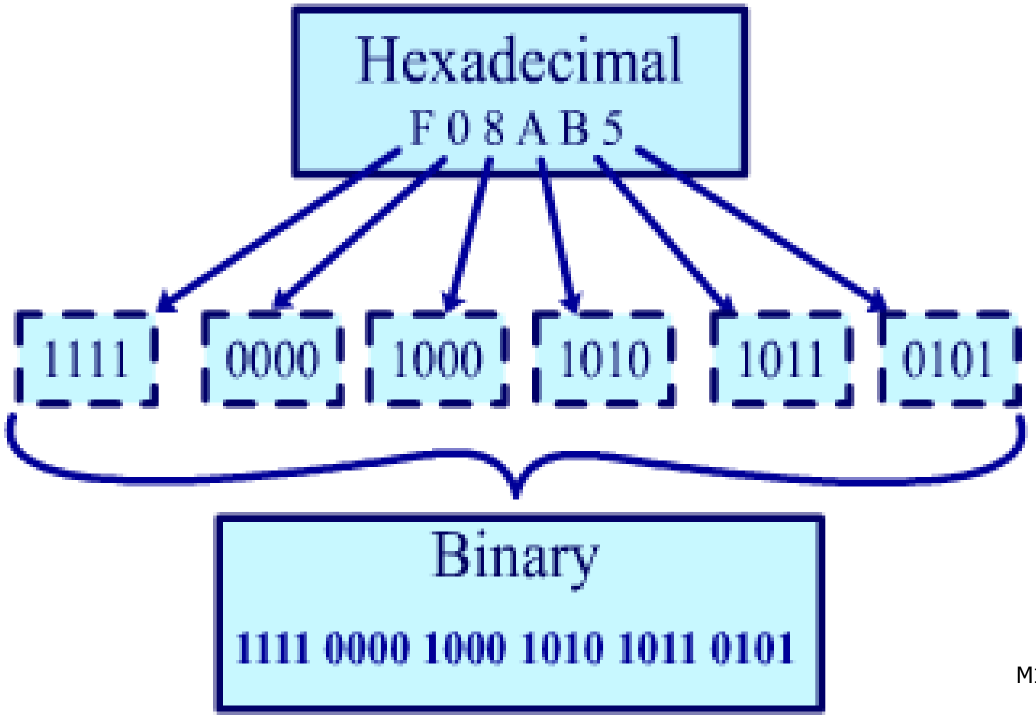
# Binary, Octal, & Hexadecimal Conversions

- Octal (or Hexadecimal) to Binary:
  - Express each octal (hexadecimal) digit as three (four) binary bits starting at the radix point and going <u>both ways</u>.
- Binary to Octal (or Hexadecimal):
  - Group the binary bits into three (four) bit groups starting at the radix point and going <u>both ways</u>, padding LH zeros in the integer part and RH zeros in the fractional part as needed
  - Replace each group of three (four) bits with the equivalent octal (hexadecimal) digit
- Octal ⇔ Hexadecimal
  - Go through **binary** as an intermediate step (use above)

  e.g. Octal ←→ ***<u>Binary</u>*** ←→ Hexadecimal

# Converting Hexadecimal to Binary

❖ Each Hexadecimal digit can be replaced by its 4-bit binary number to form the binary equivalent.




M1021.swf

# Converting Hexadecimal to Decimal

❖ Multiply each digit by its corresponding power of 16:

$\text{Decimal} = (d_3 \times 16^3) + (d_2 \times 16^2) + (d_1 \times 16^1) + (d_0 \times 16^0)$

d = hexadecimal digit

❖ Examples:

- $(1234)_{16} = (1 \times 16^3) + (2 \times 16^2) + (3 \times 16^1) + (4 \times 16^0) =$ $(4{,}660)_{10}$
- $(3BA4)_{16} = (3 \times 16^3) + (11 * 16^2) + (10 \times 16^1) + (4 \times 16^0) =$ $(15{,}268)_{10}$

# Converting Fractions

- Assume that $X_B$ has n digits, $X_B = (0.b_{-1}\ b_{-2}\ b_{-3}\ldots\ldots b_{-n})_B$
- Assume that $X_A$ has m digits, $X_A = (0.a_{-1}\ a_{-2}\ a_{-3}\ldots\ldots a_{-m})_A$

$$\text{Thus, } X_B = \overbrace{(0.b_{-1}\ b_{-2}\ b_{-3}\ldots\ldots b_{-n})_B}^{\text{Knowns}} \Rightarrow \underbrace{(0.a_{-1}\ a_{-2}\ a_{-3}\ldots\ \ldots a_{-m})_A}_{\text{Unknowns}}$$

$$X_B = a_{-1} * A^{-1} + a_{-2} * A^{-2} + \ldots\ldots\ldots a_{-m} * A^{-m}$$

$$X_B * A = \overset{\text{Integer}}{a_{-1}} + \overset{\text{Fraction}}{X_{B1}}$$

$$X_{B1} * A = a_{-2} + X_{B2}$$

$$X_{Bm-2} * A = a_{-m-1} + X_{Bm-1}$$

$$X_{Bm-1} * A = a_{-m}$$

# Converting Fractions

❖ Example: Convert $(0.731)_{10}$ to $(?)_2$

Binary Point

$0.731*2=\mathbf{1}.462$

$0.462*2=\mathbf{0}.924$

$0.924*2=\mathbf{1}.848$

$0.848*2=\mathbf{1}.696$

$0.696*2=\mathbf{1}.392$

$0.392*2=\mathbf{0}.784$

$0.784*2=\mathbf{1}.568$

$$(0.731)_{10} = (.1011101)_2$$

# Converting Fractions

❖ Example: Convert $(0.731)_{10}$ to $(?)_8$

Octal Point

$8*0.731 = 5.848$

$8*0.848 = 6.784$

$8*0.784 = 6.272$

$8*0.272 = 2.176$

$(0.731)_{10} = (0.5662)_8$

❖ Example: Convert $(0.357)_{10}$ to $(?)_{12}$

System Point

$12*0.357 = 4.284$

$12*0.284 = 3.408$

$12*0.408 = 4.896$

$12*0.896 = 10.752 \Rightarrow A = 10$

$10 = A$

$(0.357)_{10} \Rightarrow (0.434A)_{12}$

# Binary Addition

- 1 + 1 = 2, but 2 is not allowed digit in binary
- Thus, adding 1 + 1 in the binary system results in a Sum bit of 0 and a Carry bit

**Example**

Carries

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| + | | 1 | 1 | 1 | 1 | | |
| | 1 | 0 | 1 | 1 | 0 | 1 | |
| | 1 | 0 | 0 | 1 | 1 | 1 | + |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | |

**Binary Addition Table**

| | Carry | Sum |
|---|---|---|
| Weight | $2^1$ | $2^0$ |
| 0 + 0 | 0 | 0 |
| 0 + 1 | 0 | 1 |
| 1 + 0 | 0 | 1 |
| 1 + 1 | 1 | 0 |

$\equiv 1 \times 2^1$ $\equiv 0 \times 2^0$

$\equiv +2$

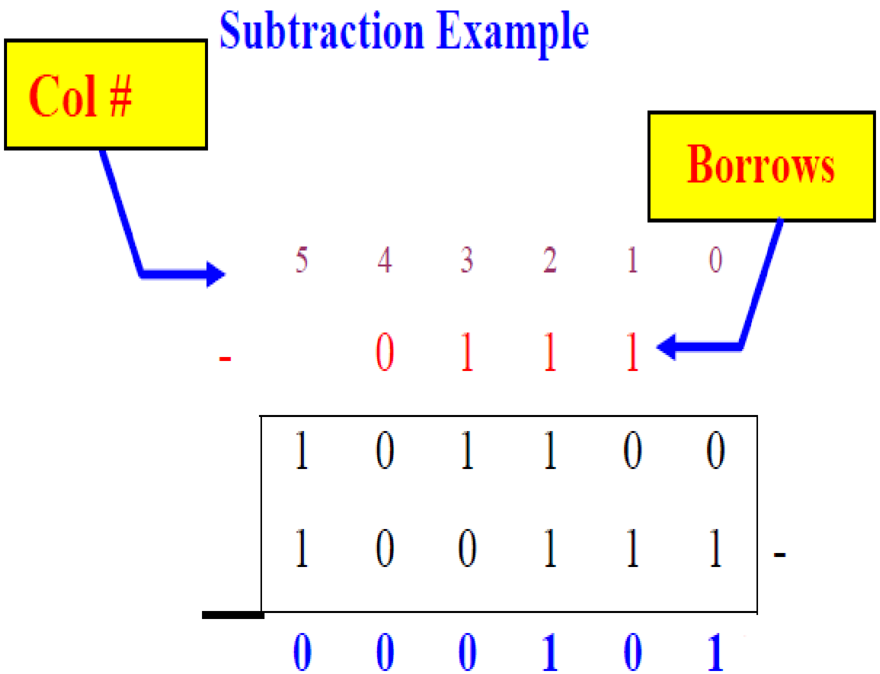
# Binary Addition

- Start with the least significant bit (rightmost bit)
- Add each pair of bits
- Include the carry in the addition, if present

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| carry: | | | | | 1 | | | | |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | (4) |
| + | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | (7) |
| | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | (11) |
| **bit position:** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | |

# Binary Subtraction

- The borrow digit is negative and has the weight of the next higher digit.




| | Borrow | Difference |
|---|---|---|
| Weight | $-2^1$ | $+2^0$ |
| 0 - 0 | 0 | 0 |
| 1 - 1 | 0 | 0 |
| 1 - 0 | 0 | 1 |
| 0 - 1 | 1 | 1 |

$\equiv 1x(-2^1)$ $\equiv +1x2^0$

$\equiv -1$

# Binary Multiplication

- Binary multiplication is performed similar to decimal multiplication.
- Example: 11 * 5 = 55

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Multiplicand** | | | 1 | 0 | 1 | 1 | |
| **Multiplier** | | | | 1 | 0 | 1 | x |
| | | | 1 | 0 | 1 | 1 | |
| | | 0 | 0 | 0 | 0 | | + |
| | 1 | 0 | 1 | 1 | | | + |
| | 1 | 1 | 0 | 1 | 1 | 1 | |

# Hexadecimal Addition

- Divide the sum of two digits by the number base (16). The quotient becomes the carry value, and the remainder is the sum digit.

| | | | |
|---|---|---|---|
| | | 1 | 1 |
| 36 | 28 | 28 | 6A |
| 42 | 45 | 58 | 4B |
| 78 | 6D | 80 | B5 |

21 / 16 = 1, remainder 5

# Binary Codes

- Digital systems and circuits work with signals that have only one of two states corresponding to digital 1 and 0.
- Any discrete element of information among a group of quantities (elements) can be represented by a binary code.
- One bit can represent up to two elements (1 or 0).
- A binary code is a group of bits (1's and 0's)
- A Byte is a binary code of 8 bits
- A group of $2^n$ distinct elements requires a minimum of n bits
- The bit combination of an n-bit code is determined from the count in binary from 0 to $2^n$-1. Each element is assigned a unique binary bit combination, and no two elements can have the same code to remove ambiguity.
- To code a group of m elements, we need to use n bits such that: $2^n \geq m$

# Using Binary Bits to Represent Information: Numbers and Codes

- We can assign any combination of binary bits (called a code word) to represent any information item as long as data is uniquely represented
- Two Basic Types of Information:
  - Numeric (e.g. signed or unsigned integers)
    - Code tied to binary numbers
    - Better use codes that allow simple implementation of common arithmetic operations
  - Non-numeric (e.g. symbols, colors, the alphabet)
    - Code not tied to binary numbers
    - Greater flexibility: No arithmetic operations involved

# Binary Codes

- Examples:
  - A group of four elements can be represented by two bit code [00, 01, 10, and 11].
  - A binary code to represent the decimal digits [0-9], must contain at least 4 bits because $(2^4=16) \geq 10 \geq (2^3=8)$.

# Binary Coded Decimal

- Computer systems (binary), People (Decimal)
  - Decimal → Binary → calculation → Decimal
- Store numbers in Decimal.
  - ➔ code decimal numbers by binary codes
- $(2^4=16) \geq 10 \geq (2^3=8)$ ➔ 4 bits needed.
- 6 codes are not used.
- A decimal number in BCD is the same as its equivalent binary number.

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

| ecimal Digit | BCD 8421 |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

# 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 |

# ASCII Code

- American Standard Code for Information Interchange (ASCII)
- Represents numbers (10), letter (26), special characters (%, *, and $) and controlls
- This type of code is called alpha-numeric code.
- 7-bits   $b_7b_6b_5b_4b_3b_2b_1$ to code 128 characters
- The ASCII code also contains 94 graphic characters that can be printed and 34 nonprinting characters used for various control functions

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 |

**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 Code

A = 100 0001

a = 110 0001

DEL = 111 1111

# UNICODE

- Extends ASCII to 65,536 universal characters codes
  - 2 byte (16-bit) code words
  - For encoding characters in world languages
  - Used in many modern applications
  - Arabic codes: from 0600 to 06FF (hex)

# Gray Code

- ❖ Used in applications that require encoding of continuous data and its transmission
- ❖ Only one bit is changing at a time
- ❖ Eliminates the problem of wrong intermediate coding when multiple bits are changed for a step transition,
- ❖ A change from 0111→1000! may produce an intermediate erroneous number if some bits takes longer to change than others.

**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 |

# Error-Detection Code

- The parity bit is an <u>extra bit</u> included in the message to make to total number of 1's EVEN or ODD.
- Parity bits are used to detect errors encountered during data transmission, the transmitter inserts the parity bit, and the receiver checks against it. If an error is detected, the data is retransmitted.
- Parity can detect error in 1, 3 or any odd combination of errors.

| | Even Parity | Odd Parity |
|---|---|---|
| A = 1000001 | <u>0</u>1000001 | <u>1</u>1000001 |
| T = 1010100 | <u>1</u>1010100 | <u>0</u>1010100 |