# Data Representation

COE 202

Digital Logic Design

Dr. Aiman El-Maleh

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

# Outline

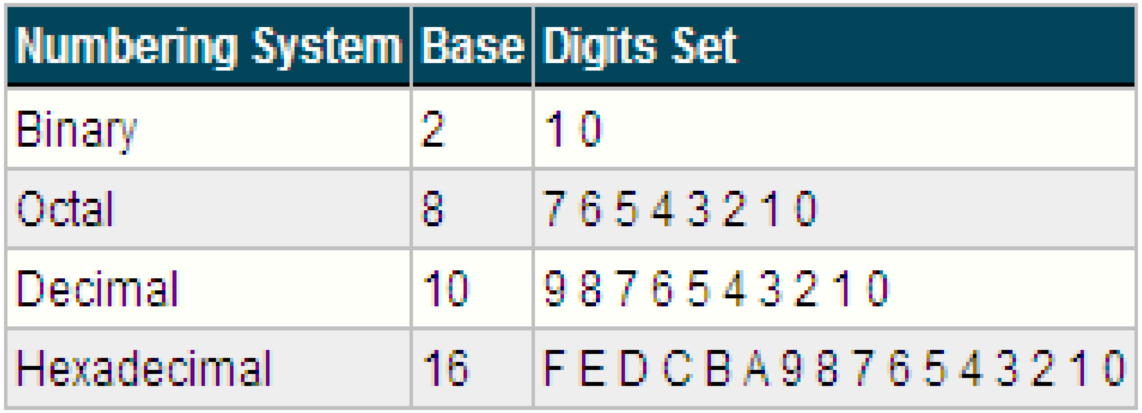
- Introduction
- Numbering Systems
- Binary & Hexadecimal Numbers
- Base Conversions
- Binary Addition, Subtraction, Multiplication
- Hexadecimal Addition
- Binary Codes for Decimal Digits
- Character Storage

# Introduction

- Computers only deal with binary data (0s and 1s), hence all data manipulated by computers must be represented in binary format.
- Machine instructions manipulate many different forms of data:
  - Numbers:
    - Integers: 33, +128, -2827
    - Real numbers: 1.33, +9.55609, -6.76E12, +4.33E-03
  - Alphanumeric characters (letters, numbers, signs, control characters): examples: A, a, c, 1 ,3, ", +, Ctrl, Shift, etc.
  - Images (still or moving): Usually represented by numbers representing the Red, Green and Blue (RGB) colors of each pixel in an image,
  - Sounds: Numbers representing sound amplitudes sampled at a certain rate (usually 20kHz).
- So in general we have two major data types that need to be represented in computers; numbers and characters.

# Numbering Systems

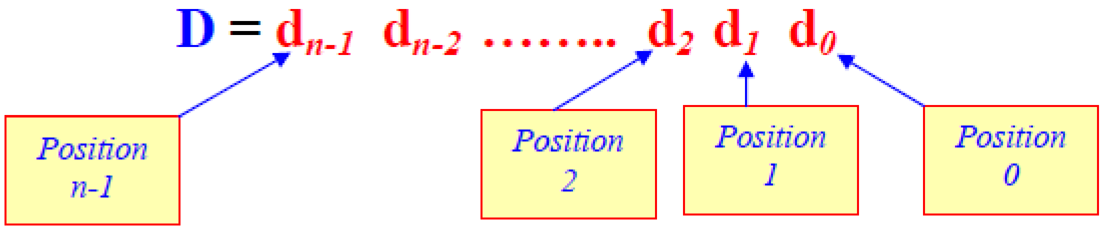
- ❖ Numbering systems are characterized by their base number.
- ❖ In general a numbering system with a base *r* will have *r* different digits (including the 0) in its number set. These digits will range from *0 to r-1.*
- ❖ The most widely used numbering systems are listed in the table below:



| 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*.



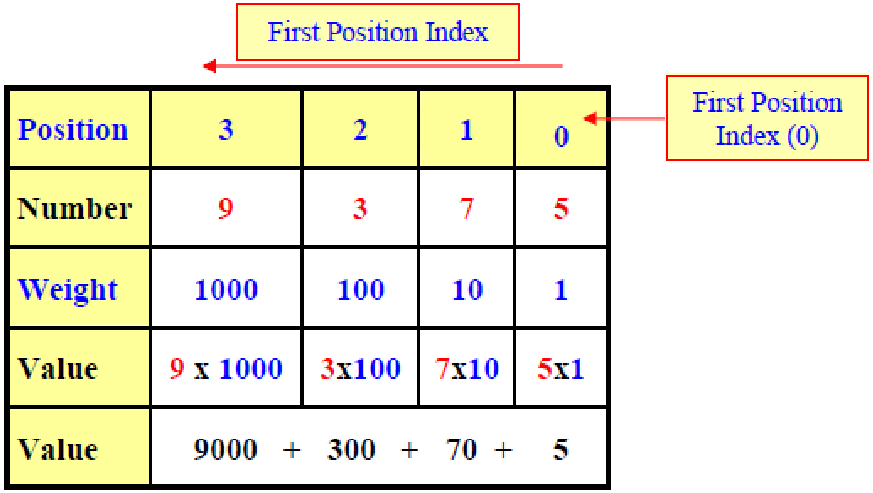
$$D = d_{n-1}\ d_{n-2}\ \ldots\ldots\ d_2\ d_1\ d_0$$

- 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 (النظام العشري) 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)

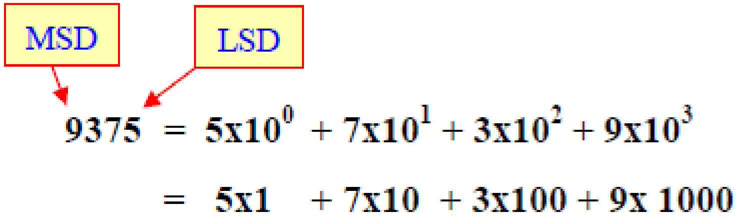
| Position | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Number | 9 | 3 | 7 | 5 |
| Weight | 1000 | 100 | 10 | 1 |
| Value | 9 x 1000 | 3x100 | 7x10 | 5x1 |
| Value | 9000 + | 300 + | 70 + | 5 |

# The Radix (Base)

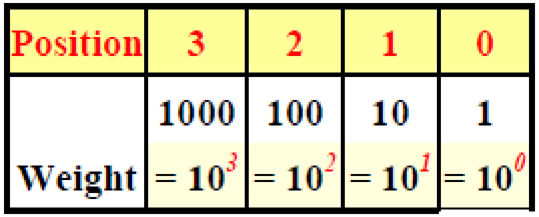
- For *digit position i*, most weighted number systems use weights ($w_i$) that are powers of some constant value called the radix (*r*) or the base such that $w_i = r^i$.
- A number system of radix *r*, typically has a set of r allowed digits ∈ {0,1, …,(r-1)}.
- The leftmost digit has the highest weight → 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}



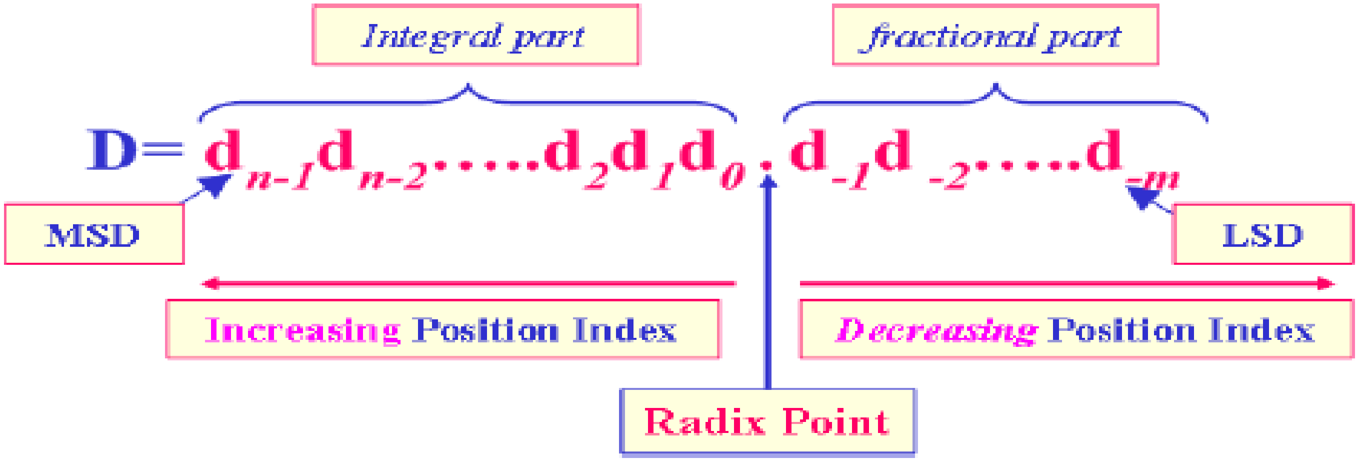
$$9375 = 5x10^0 + 7x10^1 + 3x10^2 + 9x10^3$$
$$= 5x1 + 7x10 + 3x100 + 9x\ 1000$$



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

# The Radix Point

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



Integral part
fractional part
$D = d_{n-1}d_{n-2}\ldots d_2d_1d_0 . d_{-1}d_{-2}\ldots d_{-m}$
MSD
LSD
Increasing Position Index
Decreasing Position Index
Radix Point

- 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.*

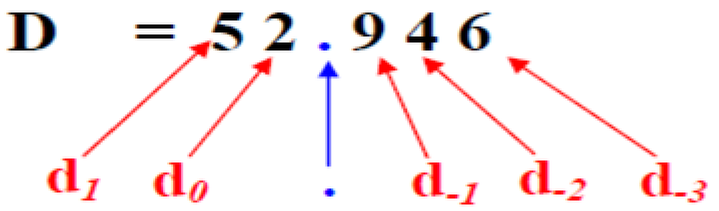
# The Radix Point

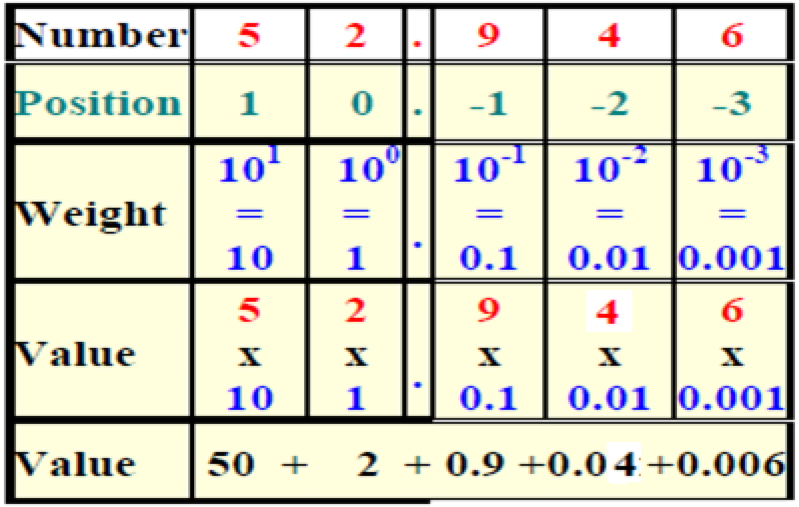
- Position indices of digits to the left of the radix point (the integral part of D) start with a 0 and are incremented as we move left ($d_{n-1}d_{n-2}.....d_2d_1d_0$).
- Position indices of digits to the right of the radix point (the fractional part of D) start with a -1 and are decremented as we move 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.





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

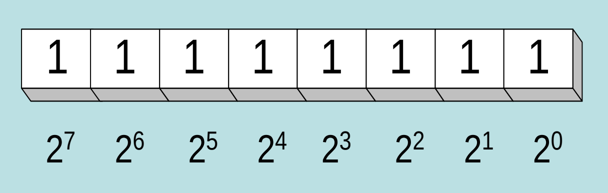
$$D = 5x10^1 + 2x10^0 + 9x10^{-1} + 4x10^{-2} + 6x10^{-3}$$

# 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.

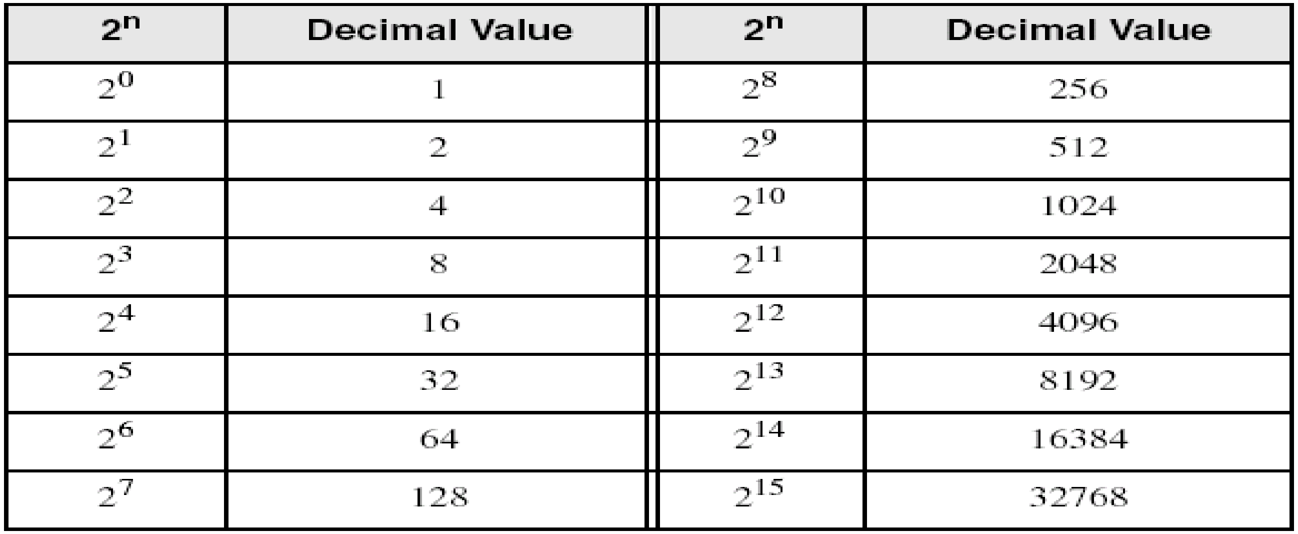
# Binary System

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



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

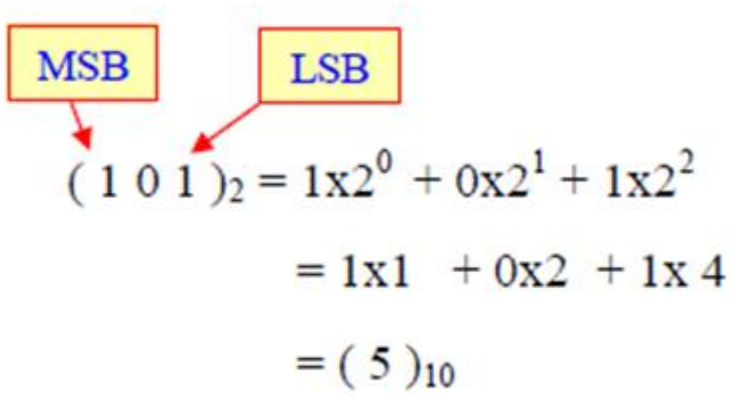
**Table 1-3** Binary Bit Position Values.



| $2^n$ | 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 $(1.101)_2$



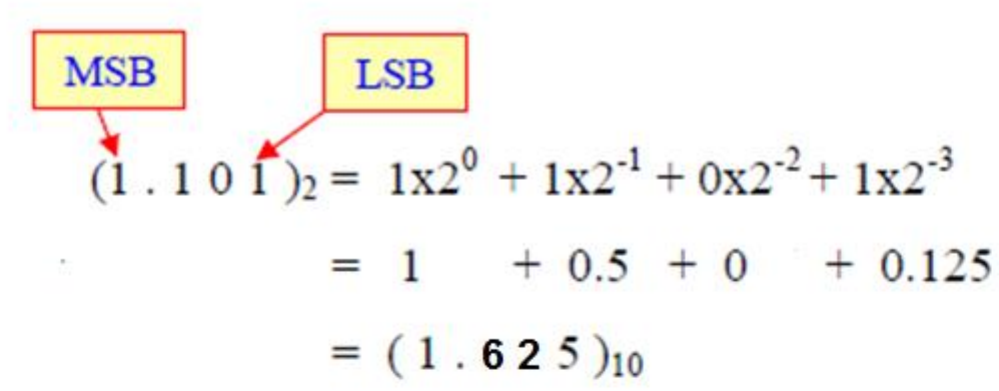
MSB

LSB

$$(1\ 0\ 1)_2 = 1x2^0 + 0x2^1 + 1x2^2$$

$$= 1x1 \quad + 0x2 \quad + 1x\,4$$

$$= (\ 5\ )_{10}$$



MSB

LSB

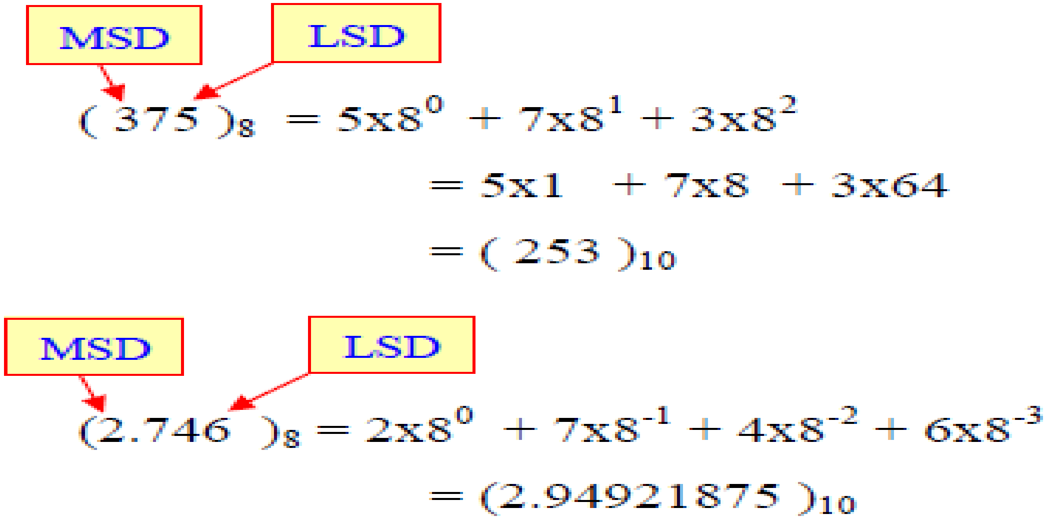
$$(1\ .\ 1\ 0\ 1\ )_2 = 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$$

$$= 1 \quad + 0.5 \quad + 0 \quad + 0.125$$

$$= (\ 1\ .\ 6\ 2\ 5\ )_{10}$$

# 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 $(2.746)_8$



MSD

LSD

$$(375)_8 = 5x8^0 + 7x8^1 + 3x8^2$$

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

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

MSD

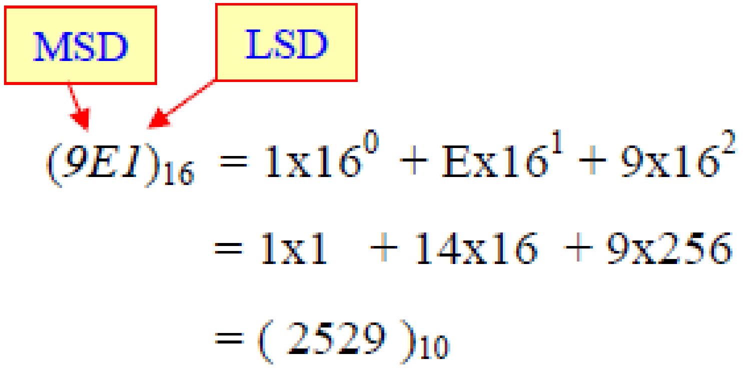
LSD

$$(2.746)_8 = 2x8^0 + 7x8^{-1} + 4x8^{-2} + 6x8^{-3}$$

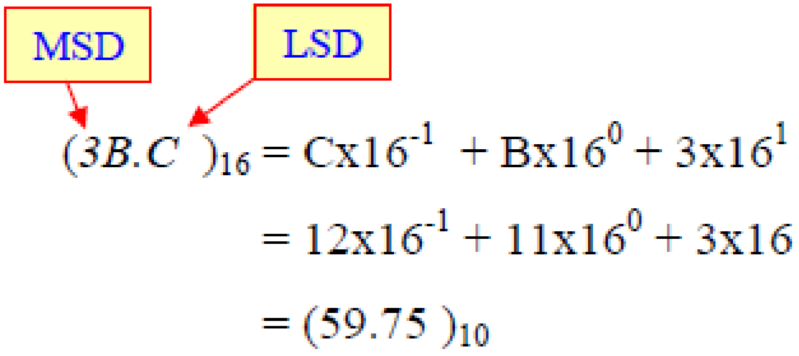
$$= (2.94921875)_{10}$$

# 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 $(3B.C)_{16}$



$$(9E1)_{16} = 1x16^0 + Ex16^1 + 9x16^2$$
$$= 1x1 + 14x16 + 9x256$$
$$= (2529)_{10}$$



$$(3B.C)_{16} = Cx16^{-1} + Bx16^0 + 3x16^1$$
$$= 12x16^{-1} + 11x16^0 + 3x16$$
$$= (59.75)_{10}$$

# Hexadecimal Integers

- Binary values are represented in hexadecimal.

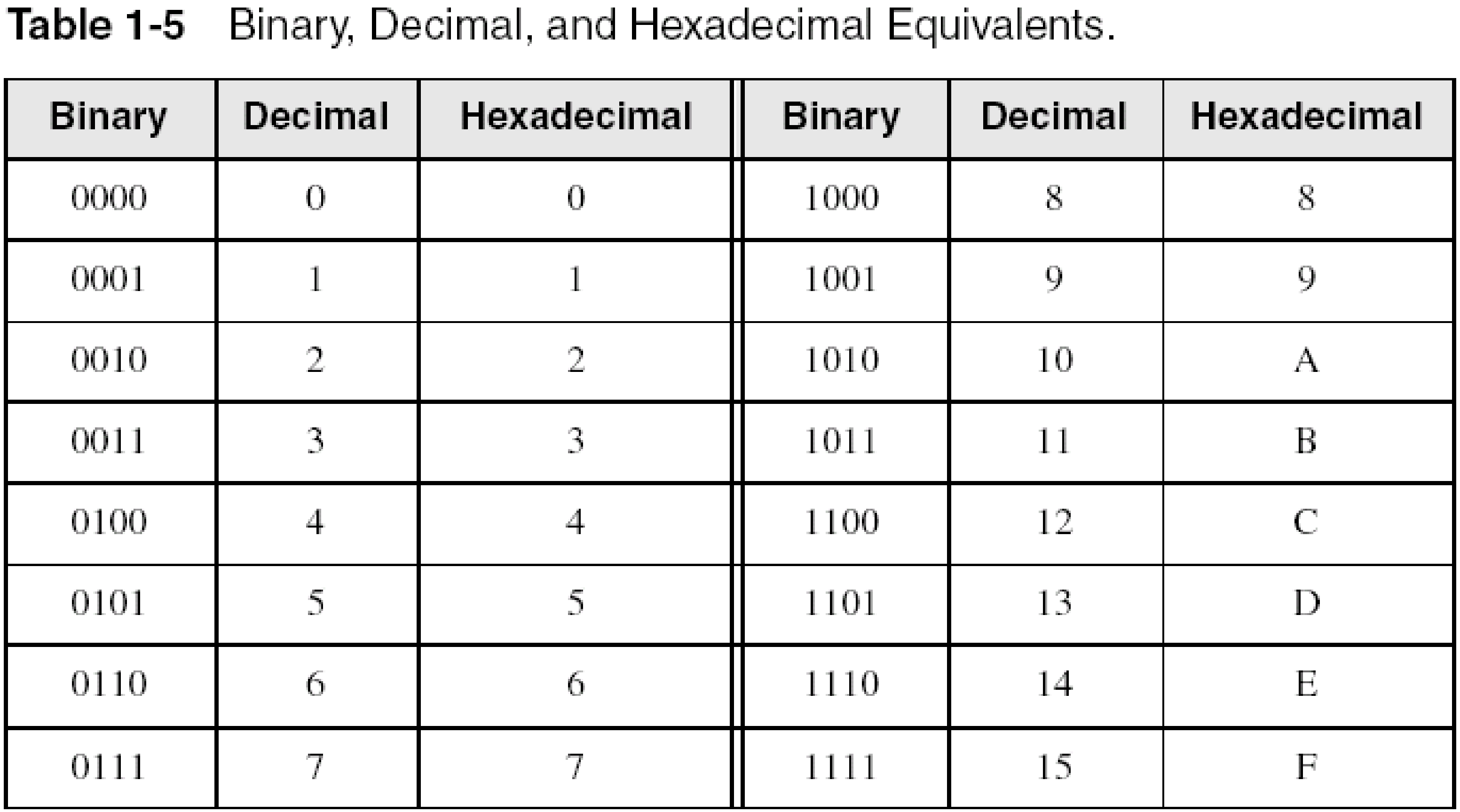


**Table 1-5** Binary, Decimal, and Hexadecimal Equivalents.

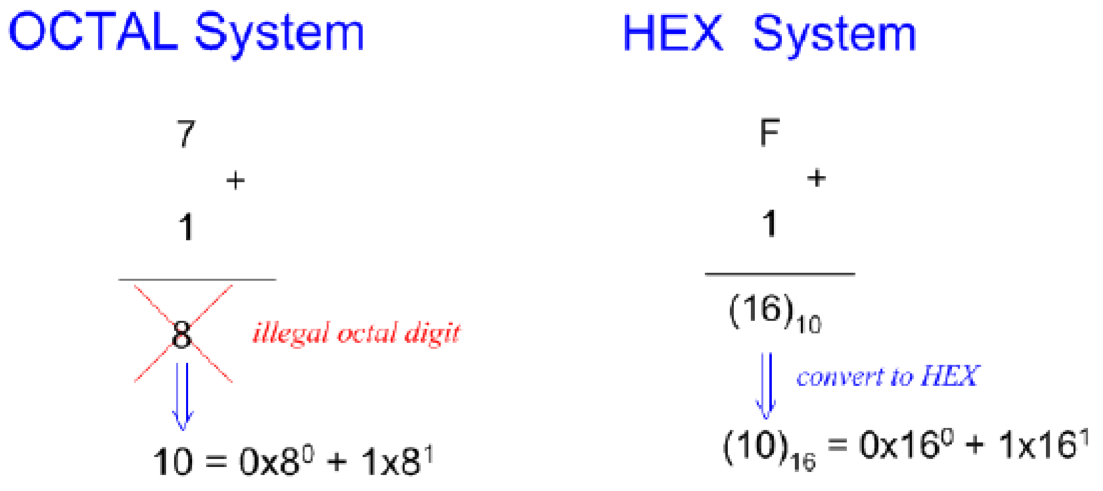
| Binary | Decimal | Hexadecimal | Binary | Decimal | Hexadecimal |
|---|---|---|---|---|---|
| 0000 | 0 | 0 | 1000 | 8 | 8 |
| 0001 | 1 | 1 | 1001 | 9 | 9 |
| 0010 | 2 | 2 | 1010 | 10 | A |
| 0011 | 3 | 3 | 1011 | 11 | B |
| 0100 | 4 | 4 | 1100 | 12 | C |
| 0101 | 5 | 5 | 1101 | 13 | D |
| 0110 | 6 | 6 | 1110 | 14 | E |
| 0111 | 7 | 7 | 1111 | 15 | 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, $(1)_{16} + (F)_{16} = (10)_{16} = (16)_{10}$



OCTAL System

$7 + 1 = \cancel{8}$

*illegal octal digit*

$10 = 0x8^0 + 1x8^1$

HEX System

$F + 1 = (16)_{10}$

*convert to HEX*

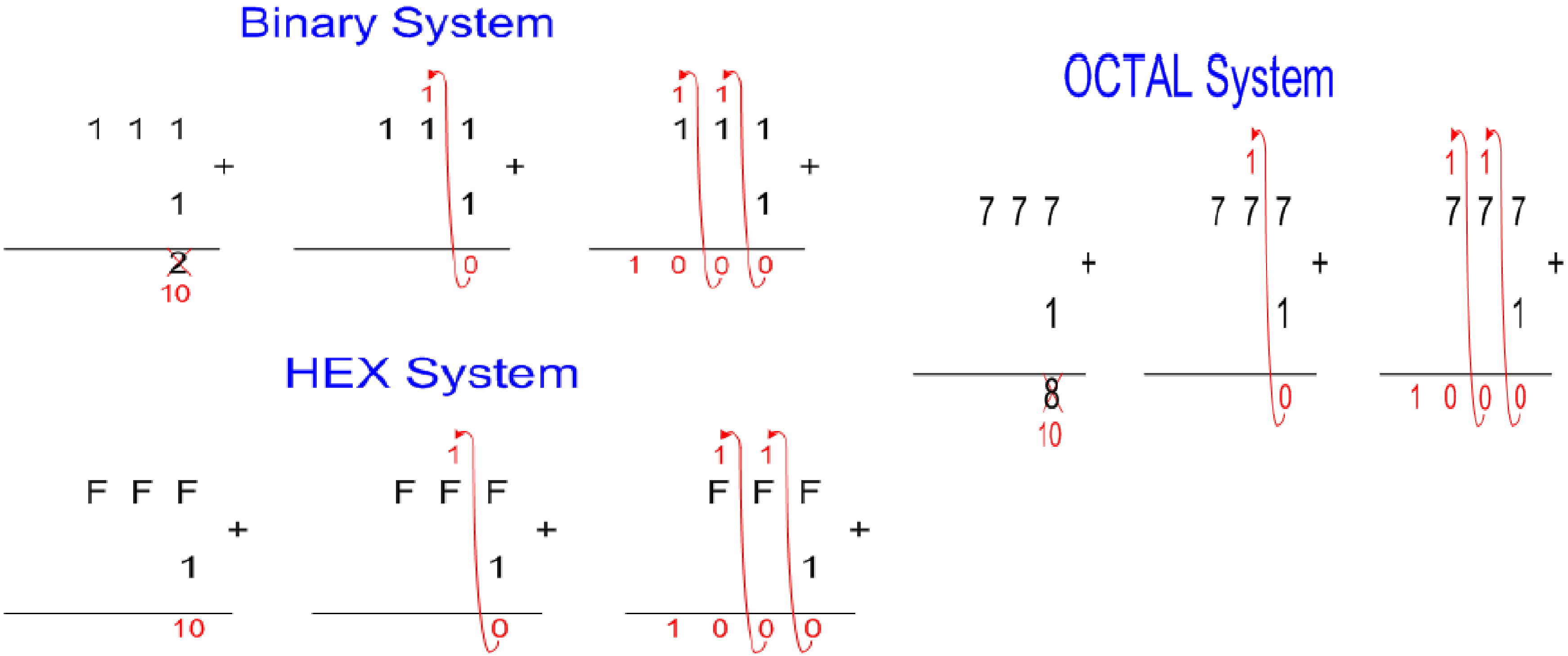
$(10)_{16} = 0x16^0 + 1x16^1$

# 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* = $r^n - 1$.

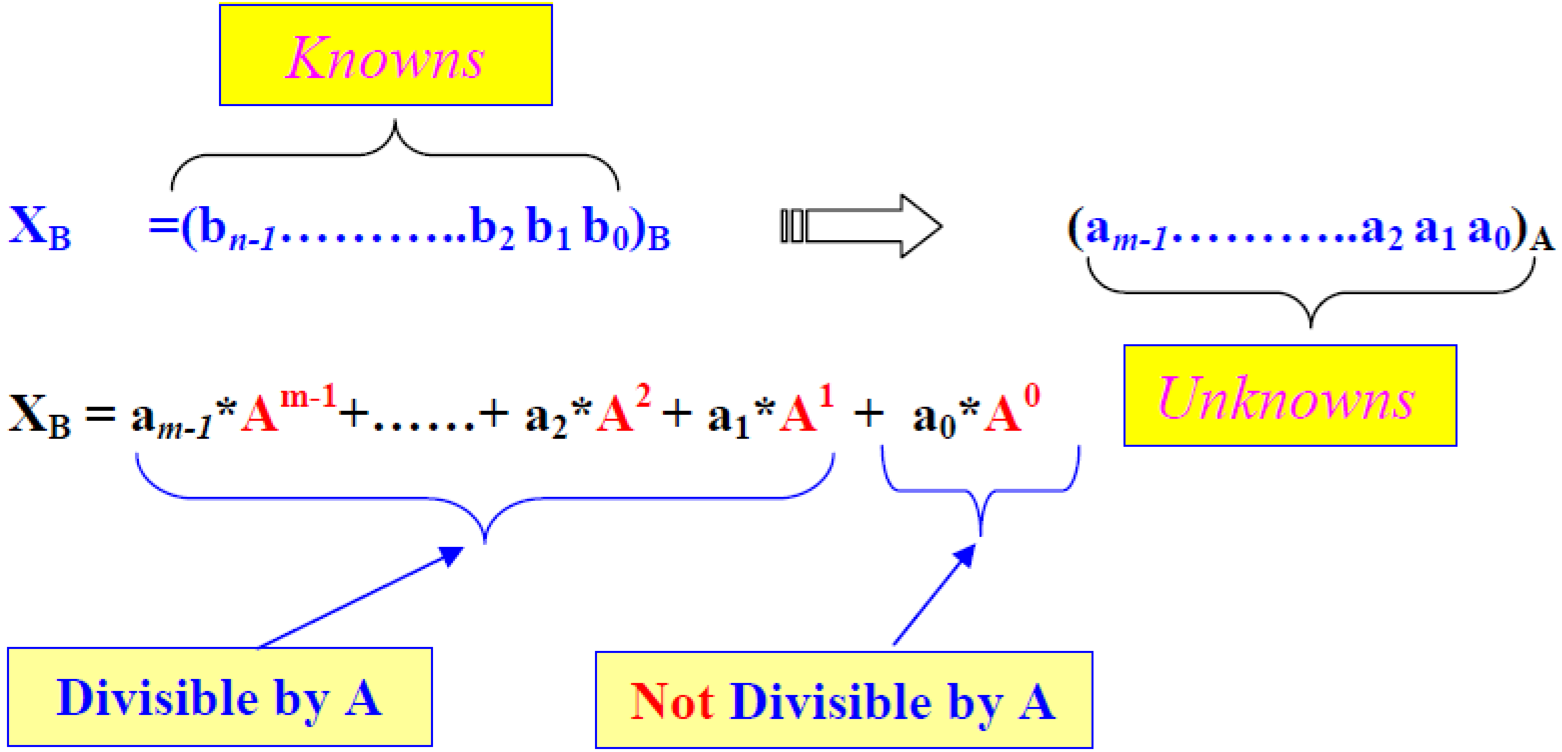


# Number Base Conversion

- 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 $(b_{n-1}.........b_2\ b_1\ b_0)_B$, where $b_i$ is a digit in radix B system, i.e. $b_i \in \{0, 1, ....., \text{“B-1”}\}$.
  - Assume that $X_A$ has m digits $(a_{m-1}.........a_2\ a_1\ a_0)_A$, where $a_i$ is a digit in radix A system, i.e. $a_i \in \{0, 1, ....., \text{“A-1”}\}$.

# Converting Whole (Integer) Numbers

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



Knowns

$X_B \quad =(b_{n-1}..........b_2\, b_1\, b_0)_B \Rightarrow (a_{m-1}..........a_2\, a_1\, a_0)_A$

Unknowns

$X_B = a_{m-1}*A^{m-1}+......+ a_2*A^2 + a_1*A^1 + a_0*A^0$

Divisible by A

Not Divisible by A

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

# Converting Whole (Integer) Numbers

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

$Q_0 = Q_1A+a_1$

$Q_1 = Q_2A+a_2$

....................................

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

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

$=Q_{m-1}A+a_{m-1}$

Where $Q_{m-1} = 0$

# 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 $(?)_2$

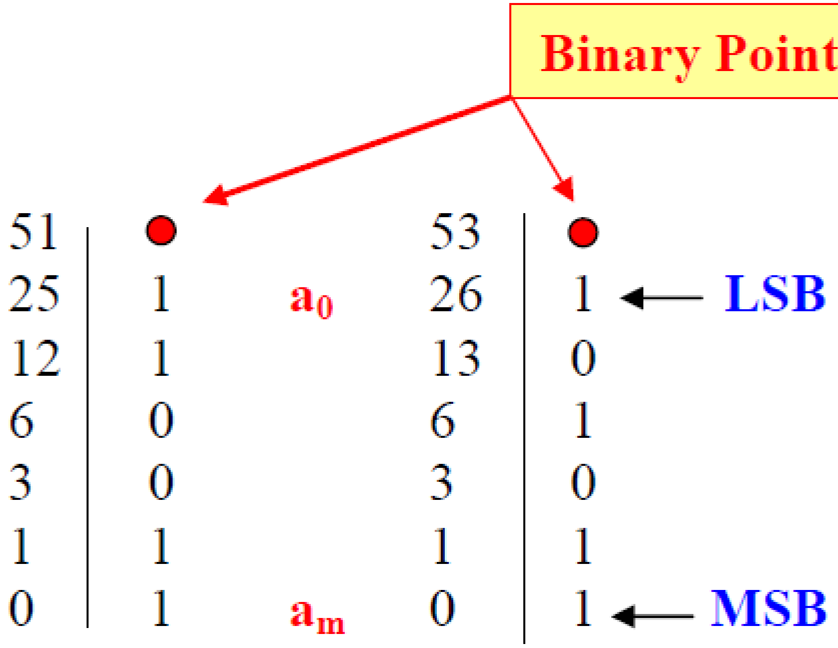
| Division Step | Quotient | Remainder | |
|---|---|---|---|
| 53 ÷ 2 | $Q_0 = 26$ | $1 = a_0$ | LSB |
| 26 ÷ 2 | $Q_1 = 13$ | $0 = a_1$ | |
| 13 ÷ 2 | $Q_2 = 6$ | $1 = a_2$ | |
| 6 ÷ 2 | $Q_3 = 3$ | $0 = a_3$ | |
| 3 ÷ 2 | $Q_4 = 1$ | $1 = a_4$ | |
| 1 ÷ 2 | 0 | $1 = a_5$ | MSB |

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:



|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 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 |

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

- ❖ 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}

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

# Converting Binary to Decimal

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

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

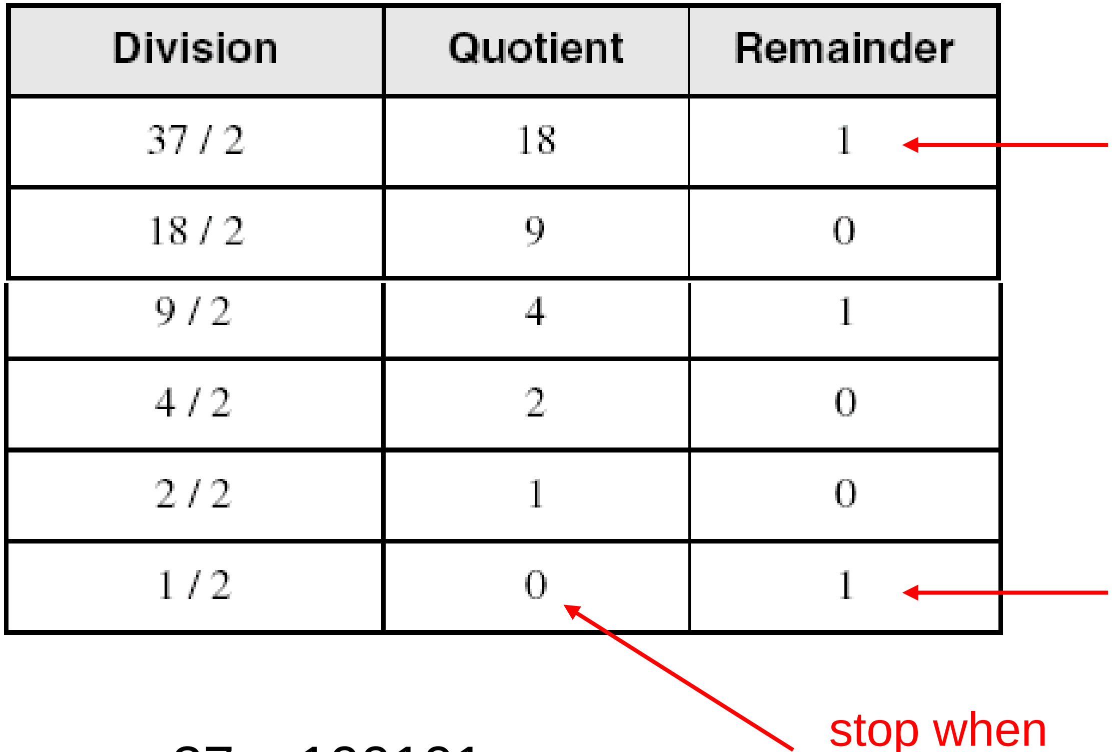
$d$ = binary digit

- binary 10101001 = decimal 169:

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

# Convert Unsigned Decimal to Binary

- Repeatedly divide the decimal integer by 2. Each remainder is a binary digit in the translated value:



| Division | Quotient | Remainder |
|---|---|---|
| 37 / 2 | 18 | 1 |
| 18 / 2 | 9 | 0 |
| 9 / 2 | 4 | 1 |
| 4 / 2 | 2 | 0 |
| 2 / 2 | 1 | 0 |
| 1 / 2 | 0 | 1 |

least significant bit

most significant bit

stop when quotient is zero

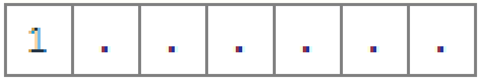
37 = 100101

# 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

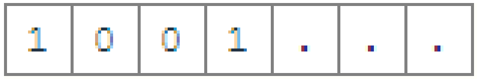
# Another Procedure for Converting from Decimal to Binary

- 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



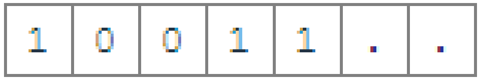
| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | . | . | . | . | . | . |

  - 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



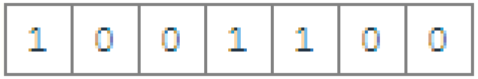
| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | . | . | . |

  - 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

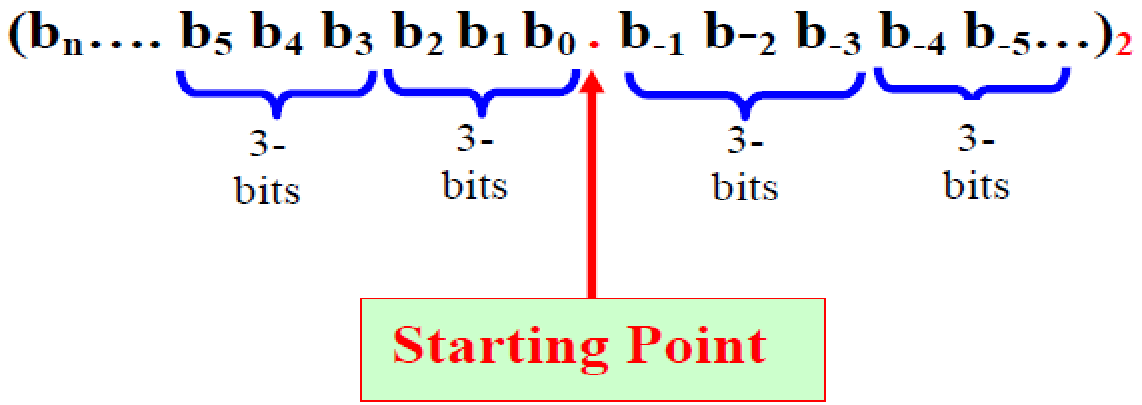


| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |

# Binary to Octal Conversion

- Each octal digit corresponds to 3 binary bits.

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



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

Starting Point

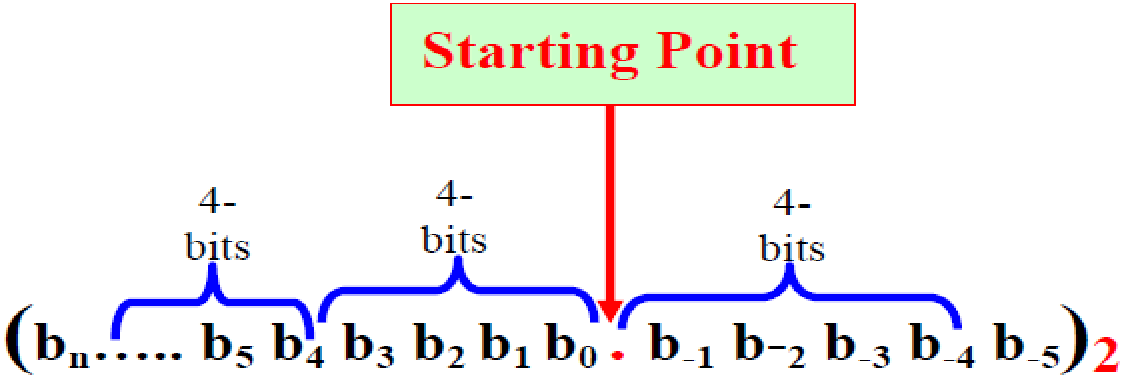
- Example: Convert $(1110010101.1011011)_2$ into Octal.

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

# Binary to Hexadecimal Conversion

- ❖ Each hexadecimal digit corresponds to 4 binary bits.

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



Starting Point

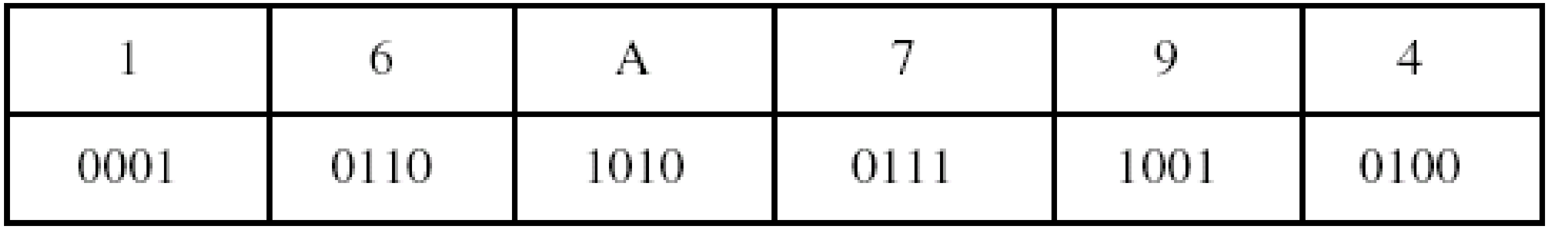
$$(\overbrace{b_n.....\ b_5\ b_4}^{\text{4-bits}}\ \overbrace{b_3\ b_2\ b_1\ b_0}^{\text{4-bits}}\ .\ \overbrace{b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}}^{\text{4-bits}}\ b_{-5})_2$$

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

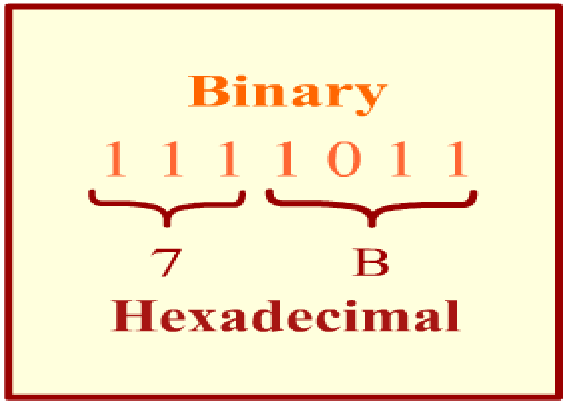
$$\underbrace{0011}_{3}\_\_\underbrace{1001}_{9}\_\_\underbrace{0101}_{5}\_.\_\underbrace{1011}_{B}\_\_\underbrace{0110}_{6} \quad = (395.B6)_{16}$$

# Binary to Hexadecimal Conversion

- Example: Translate the binary integer 000101101010011110010100 to hexadecimal



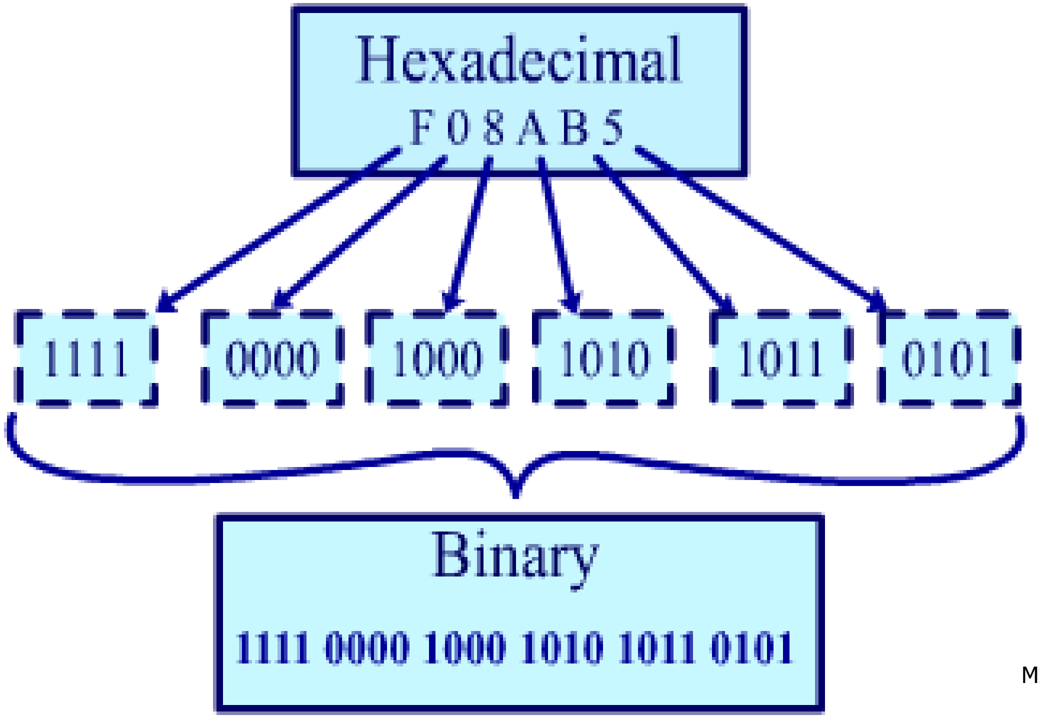
| | | | | | |
|---|---|---|---|---|---|
| 1 | 6 | A | 7 | 9 | 4 |
| 0001 | 0110 | 1010 | 0111 | 1001 | 0100 |



M1023.swf

# 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:

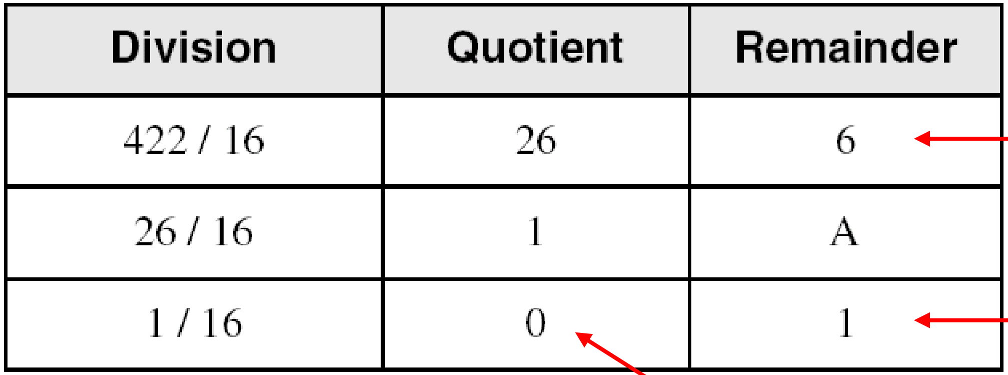
  Decimal = (d3 × $16^3$) + (d2 × $16^2$) + (d1 × $16^1$) + (d0 × $16^0$)

  d = hexadecimal digit

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

# Converting Decimal to Hexadecimal

- Repeatedly divide the decimal integer by 16. Each remainder is a hex digit in the translated value:



| Division | Quotient | Remainder |
|---|---|---|
| 422 / 16 | 26 | 6 |
| 26 / 16 | 1 | A |
| 1 / 16 | 0 | 1 |

least significant digit

most significant digit

stop when quotient is zero

$$(422)_{10} = (1A6)_{16}$$

# 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, $X_A = (0.a_{-1}\ a_{-2}\ a_{-3}\ldots\ldots a_{-m})_A$

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

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

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

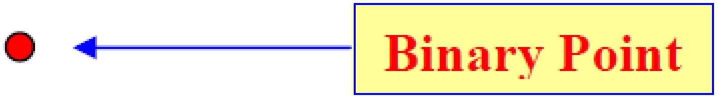
$$X_{B1}*A = a_{-2} + X_{B2}$$

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

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

# Converting Fractions

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



Binary Point

$0.731*2=1.462$

$0.462*2=0.924$

$0.924*2=1.848$

$0.848*2=1.696$

$0.696*2=1.392$

$0.392*2=0.784$

$0.784*2=1.568$

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

# 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 ⇒ A=10

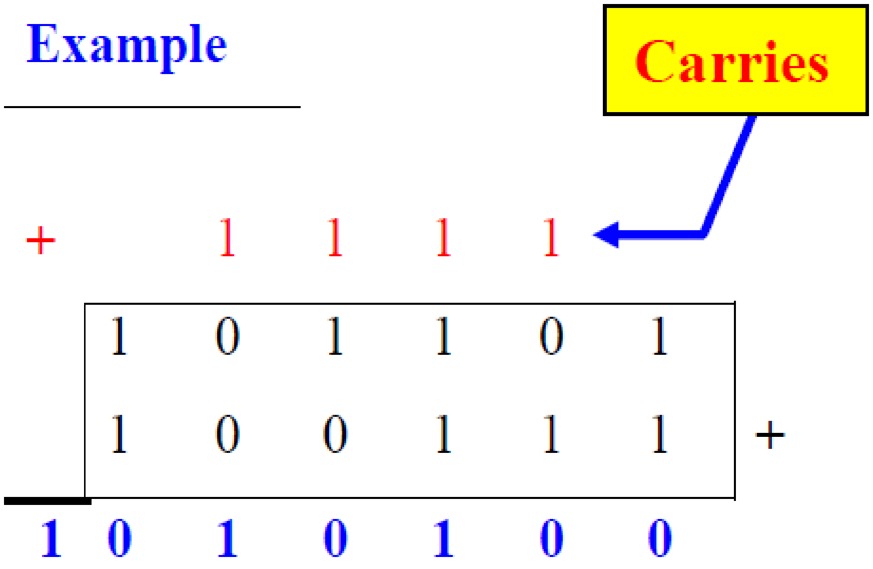
=A

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

# 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

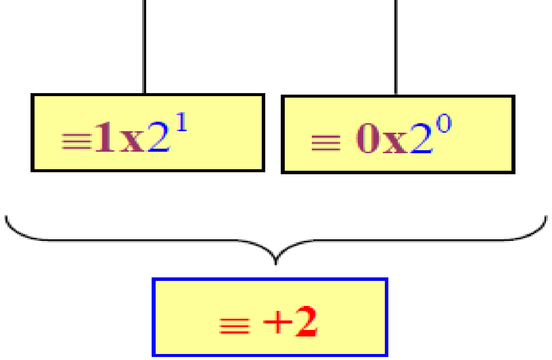


Carries

```
+       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 |



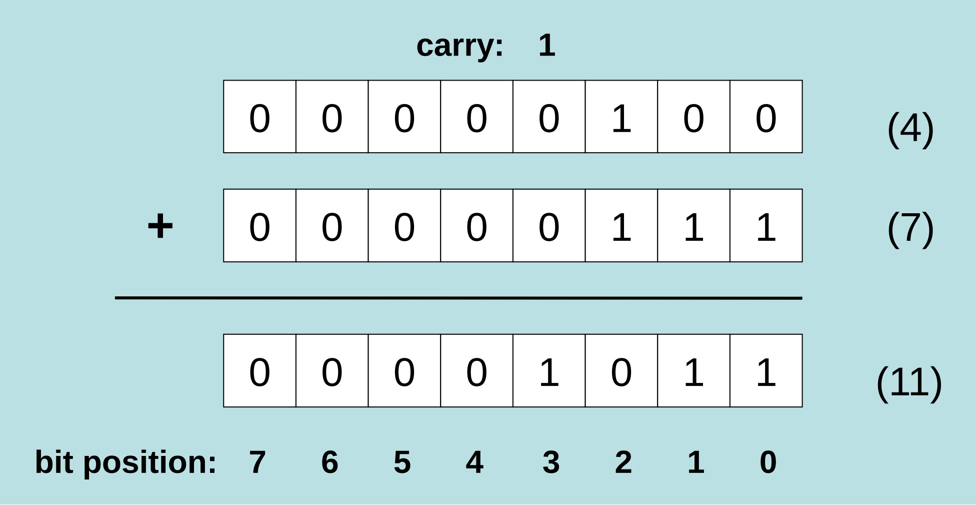
$\equiv 1\text{x}2^1$

$\equiv 0\text{x}2^0$

$\equiv +2$

# Binary Addition

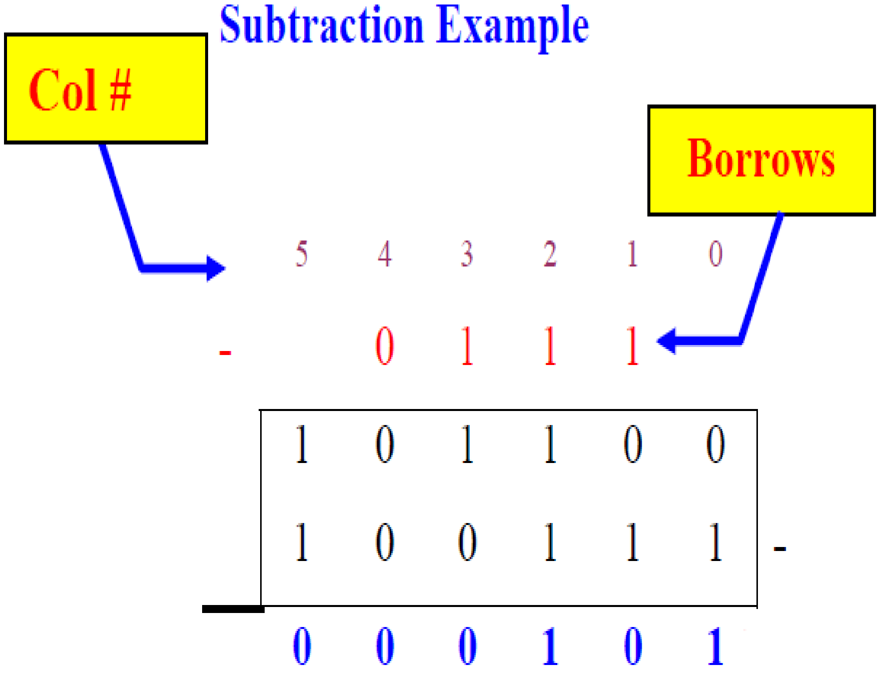
- Start with the least significant bit (rightmost bit)
- Add each pair of bits
- Include the carry in the addition, if present



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 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.



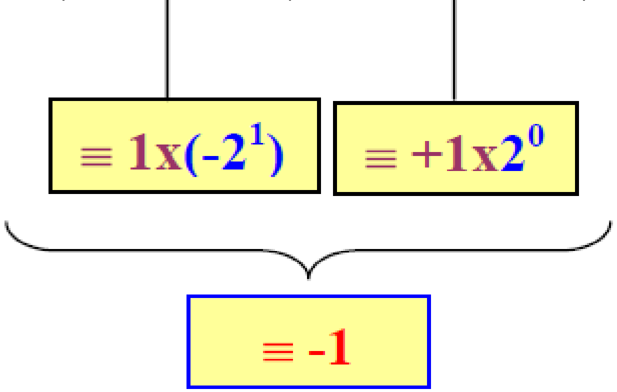
Subtraction Example

Col #

Borrows

```
      5   4   3   2   1   0
  -       0   1   1   1
      1   0   1   1   0   0
      1   0   0   1   1   1   -
      0   0   0   1   0   1
```

|  | 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.

$$\begin{array}{cccc} & & \overset{1}{\phantom{0}}\phantom{0} & \overset{1}{\phantom{0}}\phantom{0} \\ 36 & 28 & 28 & 6A \\ 42 & 45 & 58 & 4B \\ \hline 78 & 6D & 80 & B5 \end{array}$$

21 / 16 = 1, remainder 5

# Binary Codes for Decimal Digits

- Internally, digital computers operate on binary numbers.
- When interfacing to humans, digital processors, e.g. pocket calculators, communication is decimal-based.
- Input is done in decimal then converted to binary for internal processing.
- For output, the result has to be converted from its internal binary representation to a decimal form.
- To be handled by digital processors, the decimal input (output) must be coded in binary in a digit by digit manner.

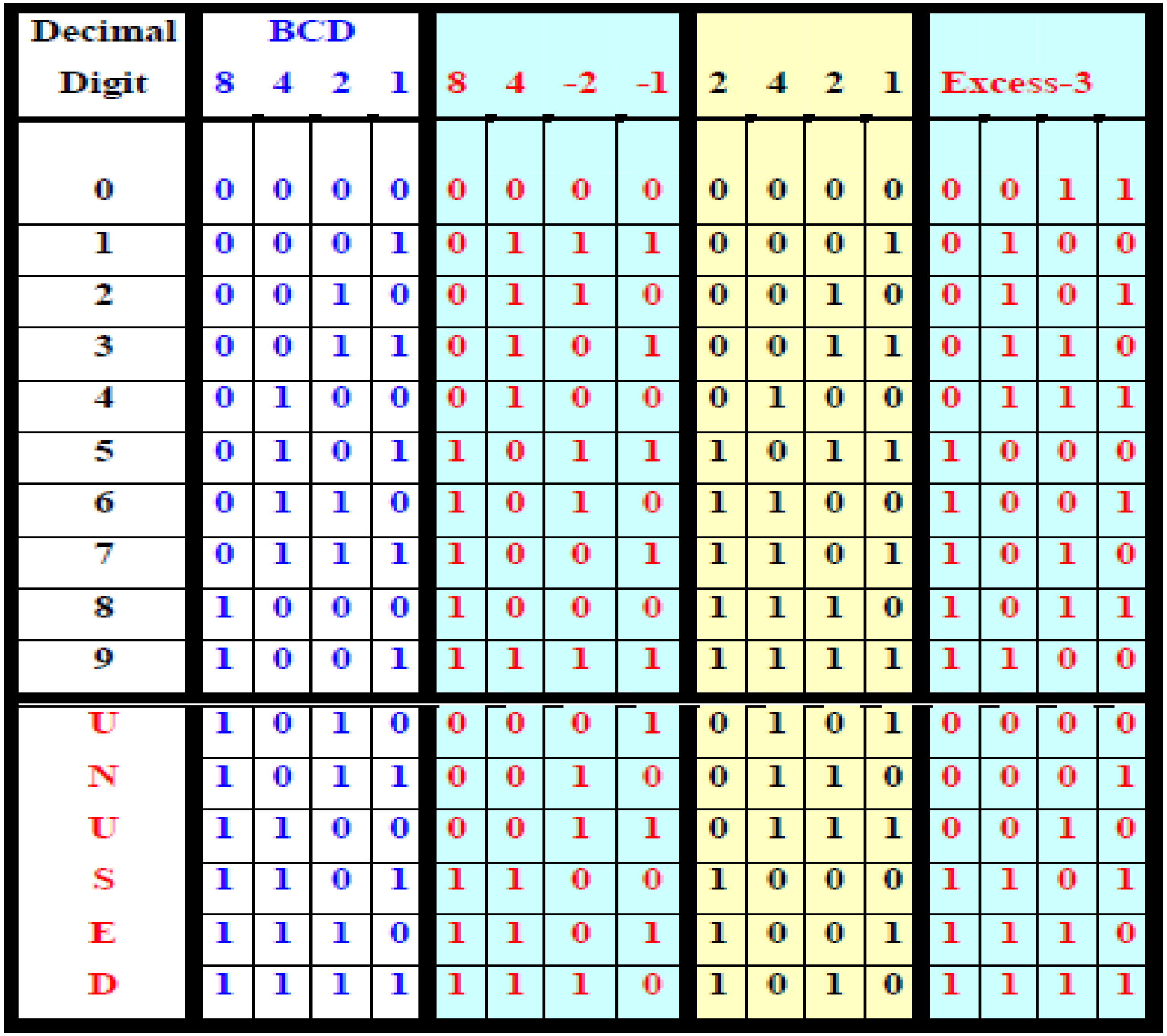
# Binary Codes for Decimal Digits

- For example, to input the decimal number 957, each digit of the number is individually coded and the number is stored as 1001_0101_0111.
- Thus, we need a specific code for each of the 10 decimal digits. There is a variety of such decimal binary codes.
- One commonly used code is the Binary Coded Decimal (BCD) code which corresponds to the first 10 binary representations of the decimal digits 0-9.
  - The BCD code requires 4 bits to represent the 10 decimal digits.
  - Since 4 bits may have up to 16 different binary combinations, a total of 6 combinations will be unused.
  - The position weights of the BCD code are 8, 4, 2, 1.

# Binary Codes for Decimal Digits

- Other codes use position weights of
  - 8, 4, -2, -1
  - 2, 4, 2, 1.
- An example of a non-weighted code is the excess-3 code
  - digit codes are obtained from their binary equivalent after adding 3.
  - Thus the code of a decimal 0 is 0011, that of 6 is 1001, etc.

# Binary Codes for Decimal Digits



| Decimal | BCD | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Digit | 8 | 4 | 2 | 1 | 8 | 4 | -2 | -1 | 2 | 4 | 2 | 1 | Excess-3 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| U | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| N | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| U | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| S | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| E | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| D | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

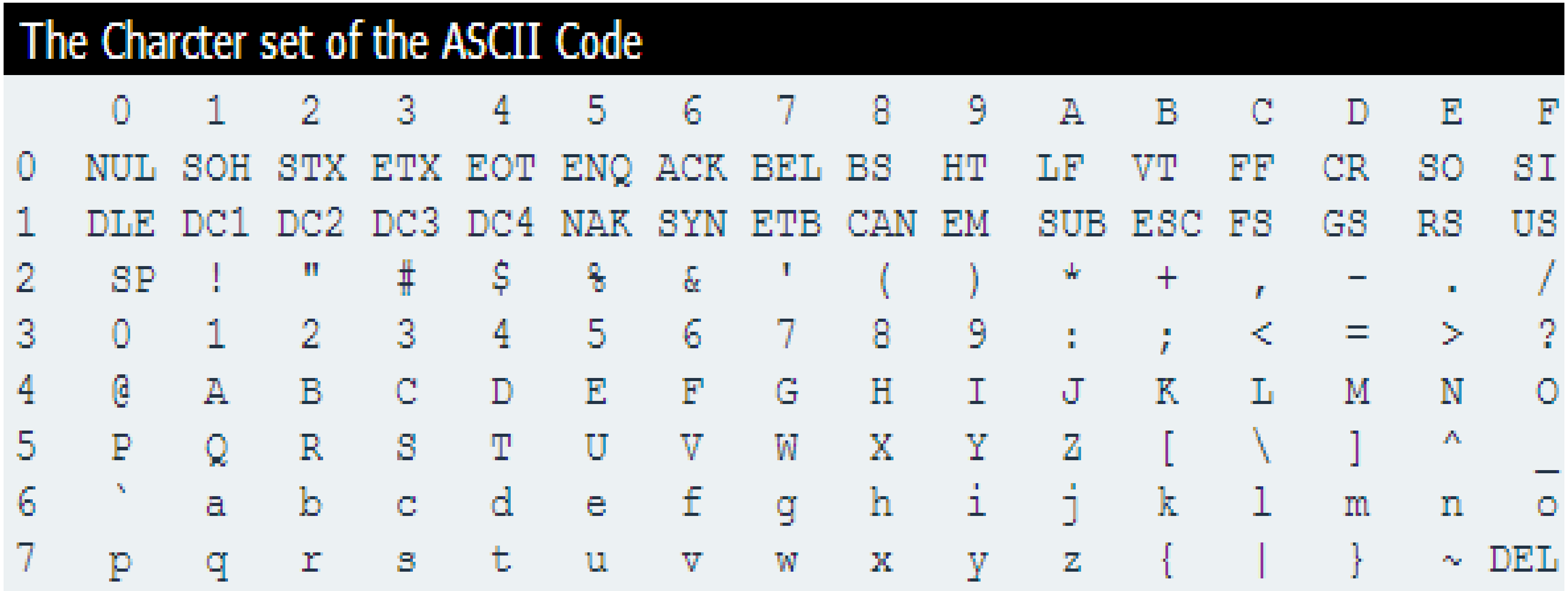
# Number Conversion versus Coding

- Converting a decimal number into binary is done by repeated division (multiplication) by 2
- Coding a decimal number into its BCD code is done by replacing each decimal digit of the number by its equivalent 4 bit BCD code.
- Example: Converting $(13)_{10}$ into binary, we get 1101, coding the same number into BCD, we obtain 00010011.
- Exercise: Convert $(95)_{10}$ into its binary equivalent value and give its BCD code as well.
- Answer: $(1011111)_2$, and 10010101.

# Character Storage

- Character sets
  - Standard ASCII: 7-bit character codes (0 – 127)
  - Extended ASCII: 8-bit character codes (0 – 255)
  - Unicode: 16-bit character codes (0 – 65,535)
  - Unicode standard represents a universal character set
    - Defines codes for characters used in all major languages
    - Used in Windows-XP: each character is encoded as 16 bits
    - Arabic codes: from 0600 to 06FF (hex)
  - UTF-8: variable-length encoding used in HTML
    - Encodes all Unicode characters
    - Uses 1 byte for ASCII, but multiple bytes for other characters

# ASCII Codes



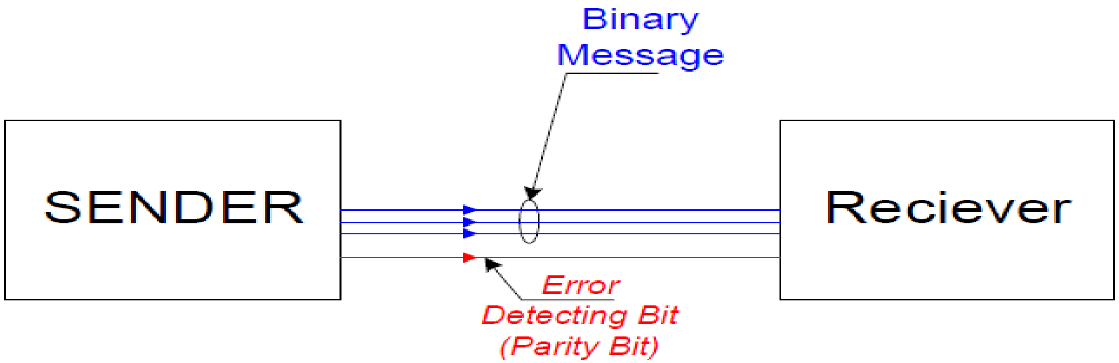
The Charcter set of the ASCII Code

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 2 | SP | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | DEL |

- Examples:
  - ASCII code for space character = 20 (hex) = 32 (decimal)
  - ASCII code for ‘A' = 41 (hex) = 65 (decimal)
  - ASCII code for 'a' = 61 (hex) = 97 (decimal)

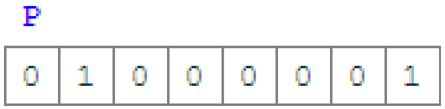
# Error Detection

- Binary information may be transmitted through some communication medium, e.g. using wires or wireless media.
- A corrupted bit will have its value changed from 0 to 1 or vice versa.
- To be able to detect errors at the receiver end, the sender sends an extra bit (parity bit) with the original binary message.



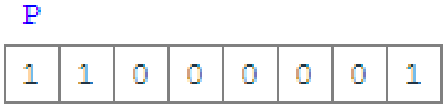
# Parity Bit

- A parity bit is an extra bit included with the n-bit binary message to make the total number of 1's in this message (including the parity bit) either odd or even.
- The 8th bit in the ASCII code is used as a parity bit.
- There are two ways for error checking:
  - Even Parity**:** Where the 8th bit is set such that the total number of 1s in the 8-bit code word is even.



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

  - Odd Parity**:** The 8th bit is set such that the total number of 1s in the 8-bit code word is odd.



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