

# Digital Logic Design

## lecture 1 :Number Systems

Eng .Shada Mohammed

Lecture contents Collected From Different Recourses

# Learning Objectives

---

- Represent data in different digital formats:
  - Positional number system
  - Binary coded decimal (BCD)
  - Hexadecimal systems
- Negative number representation
- Convert from one format to another
- Perform basic mathematical operations on data represented in different format.

# Digital Logic Design

## Introduction

A digital computer stores data in terms of digits (numbers) and proceeds in discrete steps from one state to the next. The states of a digital computer typically involve binary digits which may take the form of the presence or absence of magnetic markers in a storage medium, on-off switches or relays. In digital computers, even letters, words and whole texts are represented digitally.

Digital Logic is the basis of electronic systems, such as computers and cell phones. Digital Logic is rooted in binary code, a series of zeroes and ones each having an opposite value. This system facilitates the design of electronic circuits that convey information, including logic gates. Digital Logic gate functions include and, or and not. The value system translates input signals into specific output. Digital Logic facilitates computing, robotics and other electronic applications.

Digital Logic Design is foundational to the fields of electrical engineering and computer engineering. Digital Logic designers build complex electronic components that use both electrical and computational characteristics. These characteristics may involve power, current, logical function, protocol and user input. Digital Logic Design is used to develop hardware, such as circuit boards and microchip processors. This hardware processes user input, system protocol and other data in computers, navigational systems, cell phones or other high-tech systems.

## Common Number Systems

System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, ... 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, ... 7	No	No
Hexa-decimal	16	0, 1, ... 9, A, B, ... F	No	No

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

## Weighted Positional Notation

---

- Use the position of the symbol to indicate the value
- By assigning each position the appropriate power of the base, we can get a unique representation of numbers in that base
- People are comfortable with a Decimal System (Base-10 or radix-10)

Given a sequence of n digits  $D = d_{n-1}d_{n-2} \cdots d_1d_0$

$$V(D) = d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \cdots + d_1 \times 10^1 + d_0 \times 10^0$$

$V(D)$  : The value of number D

## Weighted Positional Notation (cont.)

---

- Binary integer: base-2 or radix-2

$$B = b_{n-1}b_{n-2} \cdots b_1b_0$$

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \cdots + b_1 \times 2^1 + b_0 \times 2^0 = \sum_{i=0}^{n-1} b_i 2^i$$

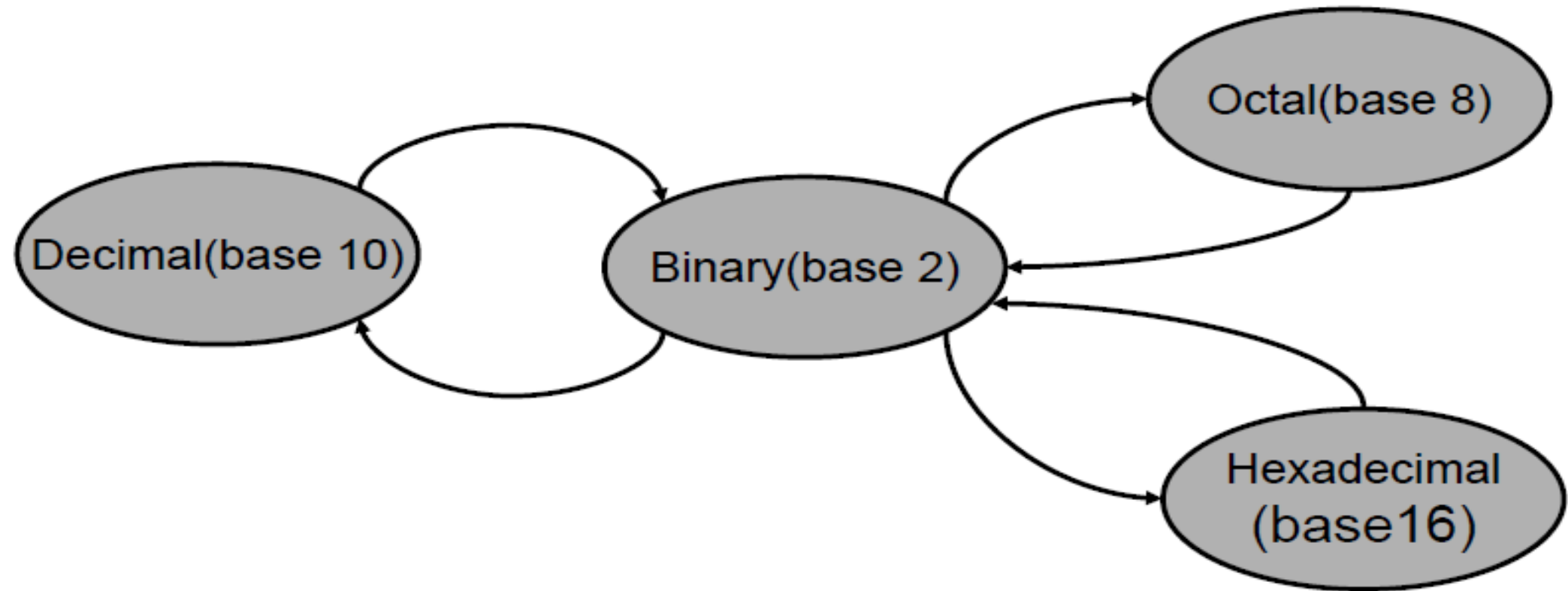
- Octal and Hexadecimal

$$K = k_{n-1}k_{n-2} \cdots k_1k_0$$

$$V(K) = \sum_{i=0}^{n-1} k_i r^i$$

- Octal (r=8), Hex (r=16)

# Conversion Between Number Bases



# Binary to Decimal

---

- Technique

- Multiply each bit by  $2^n$ , where  $n$  is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results



## Example

MSB: Most Significant Bit

LSB: Least Significant Bit

$b_0$

$101011_2 \Rightarrow$

$B = b_{n-1}b_{n-2} \cdots b_1b_0$

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

1	x	$2^0$	=	1
1	x	$2^1$	=	2
0	x	$2^2$	=	0
1	x	$2^3$	=	8
0	x	$2^4$	=	0
1	x	$2^5$	=	32
				<hr/>
				43 <sub>10</sub>

32 16 8 4 2 1  
x x x x x x  
1 0 1 0 1 1

## Octal to Decimal

---

### ■ Technique

- Multiply each bit by  $8^n$ , where  $n$  is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

### Example

---

$$\begin{array}{rcll} 724_8 \Rightarrow & 4 & \times 8^0 & = & 4 \\ & 2 & \times 8^1 & = & 16 \\ & 7 & \times 8^2 & = & 448 \\ & & & & \hline & & & & 468_{10} \end{array} +$$

## Hexadecimal to Decimal

### ■ Technique

- Multiply each bit by  $16^n$ , where  $n$  is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right
- Add the results

### Example

$$\begin{array}{rcll} ABC_{16} & \Rightarrow & C \times 16^0 & = 12 \times 1 = 12 \\ & & B \times 16^1 & = 11 \times 16 = 176 \\ & & A \times 16^2 & = 10 \times 256 = 2560 \\ & & & \hline & & & 2748_{10} \end{array}$$

A = 10

B = 11

C = 12

D = 13

E = 14

F = 15

# Decimal to Binary

---

## ■ Technique

- Divide by two, keep track of the remainder
- First remainder is bit 0 ( $b_0$ ) (LSB, least-significant bit)
- Second remainder is bit 1 ( $b_1$ )

## Example

---

$$125_{10} = ?_2$$

2		125	
2		62	1
2		31	0
2		15	1
2		7	1
2		3	1
2		1	1
		0	1

$125_{10} = 1111101_2$

LSB: Least Significant Bit

↓  
MSB

Decimal Number –  $29_{10}$

Calculating Binary Equivalent

Step	Operation	Result	Remainder
Step 1	$29 / 2$	14	1
Step 2	$14 / 2$	7	0
Step 3	$7 / 2$	3	1
Step 4	$3 / 2$	1	1
Step 5	$1 / 2$	0	1

As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the least significant digit (LSD) and the last remainder becomes the most significant digit (MSD).

Decimal Number –  $29_{10}$  = Binary Number –  $11101_2$

## Octal to Binary

---

- Technique

- Convert each octal digit to a 3-bit equivalent binary representation

### Example

---

$$\begin{array}{rccccccc} 7 & 0 & 5 & & & & \\ \downarrow & \downarrow & \downarrow & & & & \\ 111 & 000 & 101 & & & & \\ 705_8 & = & 111000101_2 & & & & \end{array}$$

## Binary to Octal

---

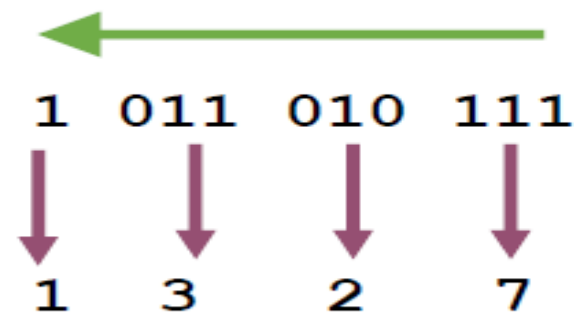
- Technique

- Group bits in threes, **starting on right**
- Convert to octal digits

### Example

---

$$1011010111_2 = ?_8$$



$$1011010111_2 = 1327_8$$

## Hexadecimal to Binary

---

- Technique

- Convert each hexadecimal digit to a 4-bit equivalent binary representation

### Example

---

$$10AF_{16} = ?_2$$

1	0	A	F
↓	↓	↓	↓
0001	0000	1010	1111

$$10AF_{16} = 0001000010101111_2$$



## Binary to Hexadecimal

---

- Technique
  - Group bits in fours, starting on **right**
  - Convert to hexadecimal digits

### Example

---

$$\begin{array}{ccccccc} & & & 10 & 1011 & 1011 & \\ & & & \downarrow & \downarrow & \downarrow & \\ & & & 2 & B & B & \\ 1010111011_2 & = & ?_{16} & & & & \\ & & & 1010111011_2 & = & 2BB_{16} & \end{array}$$

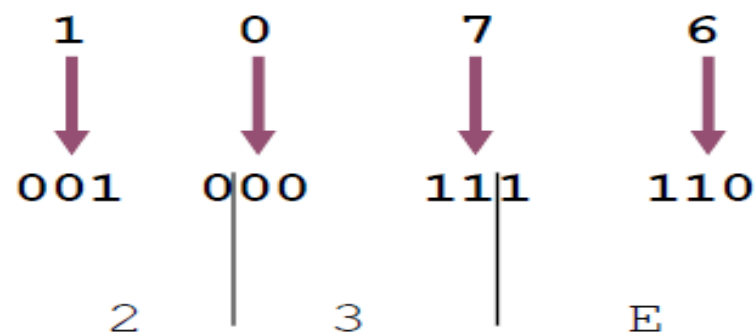
## Octal to Hexadecimal

## Hexadecimal to Octal

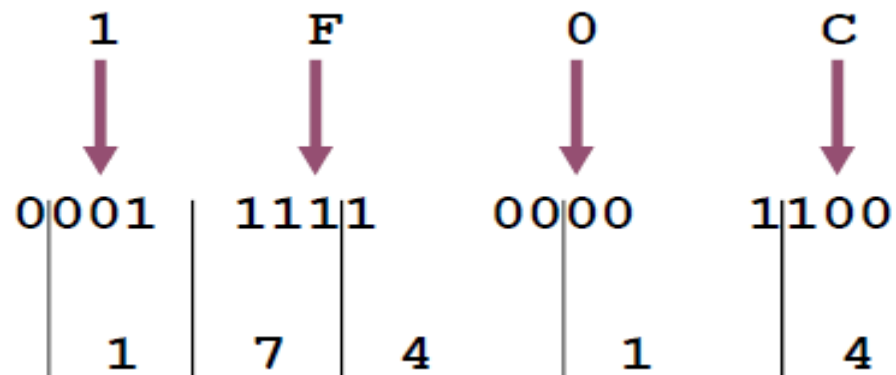
- Technique
  - Use binary as an intermediary

### Example

$$1076_8 = ?_{16}$$
$$1076_8 = 23E_{16}$$



$$1F0C_{16} = ?_8$$
$$1F0C_{16} = 17414_8$$



## Signed Integers

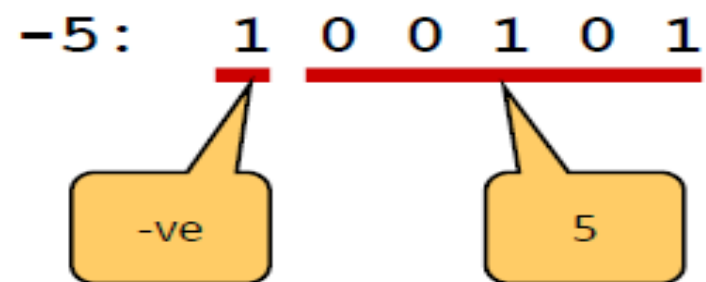
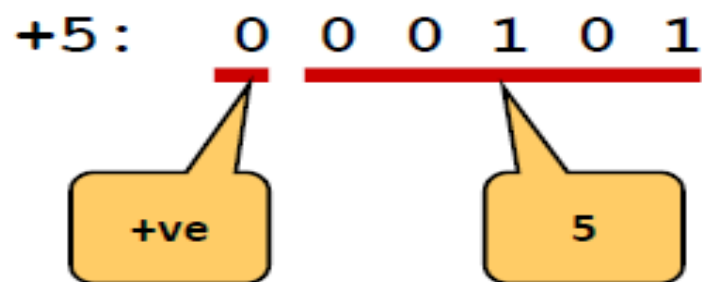
---

- Previous examples were for “unsigned integers” (positive values only!)
- Must also have a mechanism to represent “signed integers” (positive **and** negative values!)
- E.g.,  $-5_{10} = ?_2$
- **Two common** schemes: *sign-magnitude* and *twos complement*

## Sign-Magnitude

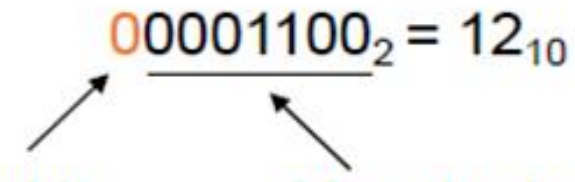
---

- Extra bit on left to represent **sign**
  - 0 = positive value
  - 1 = negative value
- E.g., 6-bit sign-magnitude representation of +5 and -5:

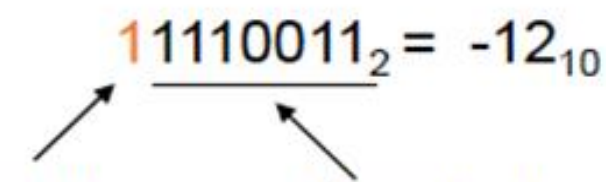


# One's Complement Representation

- The one's complement of a binary number involves inverting all bits.
  - 1's comp of 00110011 is **11001100**
  - 1's comp of 10101010 is **01010101**
- To find negative of 1's complement number take the 1's complement.


$$\underline{00001100}_2 = 12_{10}$$

Sign bit      Magnitude


$$\underline{11110011}_2 = -12_{10}$$

Sign bit      Magnitude

---

## Twos Complement Example

- Represent -5 in binary using 2's complement notation
  1. Decide on the number of bits 6 (for example)
  2. Find the binary representation of the +ve value in 6 bits
  3. Flip all the bits

4. Add 1

000101  
111010

+5

111010  
+        1  
-----  
111011

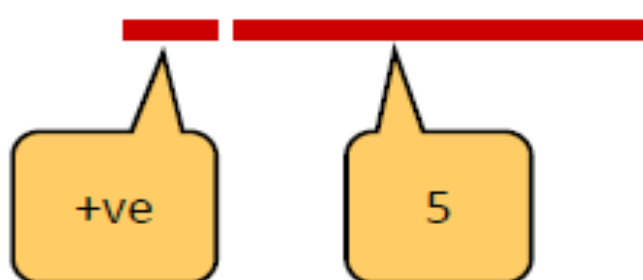
-5

## Sign Bit

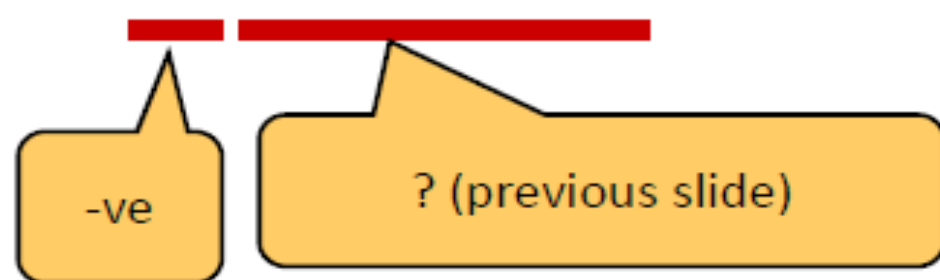
---

- In 2's complement notation, the MSB is the sign bit (as with sign-magnitude notation)
  - 0 = positive value
  - 1 = negative value

+5: 0 0 0 1 0 1



-5: 1 1 1 0 1 1



**Table 3.2** Interpretation of four-bit signed integers.

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1




## What is -5 plus +5?


---

- Zero, of course, but let's see

Sign-magnitude

$$\begin{array}{r} -5: \quad 10000101 \\ +5: \quad +00000101 \\ \hline 10001010 \end{array}$$


Twos-complement

$$\begin{array}{r} \phantom{-5:} \quad \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\ -5: \quad 11111011 \\ +5: \quad +00000101 \\ \hline 00000000 \end{array}$$


Ignore last carry

## 2's Complement Subtraction

- Easy
- No special rules
- Just subtract, well ... actually ... just add!

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

add

2's complement of B

$$10 - 3 = 10 + (-3) = 7$$

+3:	000011
1s C:	111100
+1:	<u>          1</u>
-3:	111101

$$\begin{array}{r} 001010 \\ +111101 \\ \hline 000111 \end{array}$$

# Twos Complement Examples

---

Here we'll use 4-bit values

$$\begin{array}{r} (+5) \\ + (+2) \\ \hline (+7) \end{array}$$

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

$$\begin{array}{r} (-5) \\ + (+2) \\ \hline (-3) \end{array}$$

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

$$\begin{array}{r} (+5) \\ + (-2) \\ \hline (+3) \end{array}$$

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

  
ignore

$$\begin{array}{r} (-5) \\ + (-2) \\ \hline (-7) \end{array}$$

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

  
ignore

The result is always correct since a carry from the sign-bit can be simply ignored.