



Digital Systems- ITC131

Dr. Alaa Eldeen S. Mohamed

Lecture 02: Number Systems(Part 2)

Part1-Contents

- Digital Vs Analog
- Numbering Systems [binary, octal and hexadecimal]
- Base Conversion
- Information Representation
- Binary Additions
- Complement

Part2-Contents

- Signed Integers
- Two's complement representation
- Subtraction with complements
- Decimal Codes [BCD, weighted codes]
- Logic Gates

Signed Integers

- Several ways to represent a signed number
 - Sign-Magnitude
 - 1's complement
 - 2's complement
- Divide the range of values into 2 equal parts
 - First part corresponds to the positive numbers (≥ 0)
 - Second part correspond to the negative numbers (< 0)
- Focus will be on the 2's complement representation
 - Has many advantages over other representations
 - Used widely in processors to represent signed integers

Sign Magnitude Representation

- The sign-magnitude binary format is the simplest conceptual format.
- In this method of representing signed numbers, the most significant digit (MSD) takes on extra meaning.
- If the MSD is a 0, indicates that the number is treated as positive.
- If the MSD is a 1, this indicates that the number is negative.
- The other bits indicate the magnitude (absolute value) of the number.

Number Systems (Part 2)

4-bit Binary value	Unsigned value	Signed value
0000	0	0
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7
1000	8	-0
1001	9	-1
1010	10	-2
1011	11	-3
1100	12	-4
1101	13	-5
1110	14	-6
1111	15	-7

One's Complement Representation

- Signed numbers representation using 1's complement:
- The role of sign bit is the same as of the sign magnitude form.
- Sign bit 0 and 1 represent the positive number and negative number respectively.
- The magnitude part represents the 1's complement of the actual number.

• Example 1:

• $+10 = 0\ 1\ 0\ 1\ 0$

• $-10 = 1\ 0\ 1\ 0\ 1$

Example 2:

$+5 = 0\ 1\ 0\ 1$

$-5 = 1\ 0\ 1\ 0$

- Properties of 1's complement representation
- With the n bits, the range of numbers that can be represented in 1's complement is:
- $[-2^{n-1}-1 \text{ to } +2^{n-1}-1]$
- **Example:**
- If we consider 4 bits numbers, we can represent range from (-7 to +7)
- If we consider 8 bits numbers, we can represent range from (-127 to +127)

Number Systems (Part 2)

4 -bit Binary value	Unsigned value	Signed value
0000	0	0
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7
1000	8	-7
1001	9	-6
1010	10	-5
1011	11	-4
1100	12	-3
1101	13	-2
1110	14	-1
1111	15	-0

Number Systems (Part 2)

4 -bit Binary value	Unsigned value	Signed value
0000	0	0
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Two's Complement Representation

❖ Positive numbers

✧ Signed value = Unsigned value

❖ Negative weight for MSB

✧ Another way to obtain the signed value is to assign a negative weight to most-significant bit

$$= -128 + 32 + 16 + 4 = -76$$

1	0	1	1	0	1	0	0
-128	64	32	16	8	4	2	1

- Exercise:

Represent the 1111010 using the signed-2's complement System.

1 1 1 1 0 1 0

- 0 0 0 1 1 0

compute 2's complement

- 6

- Exercise:

Represent the decimal -46 and +29 to binary. using the signed-2'a-complement System.

+ 29 → We need 0 11101

-46 → first we get the value of 46 then compute the 2's complement

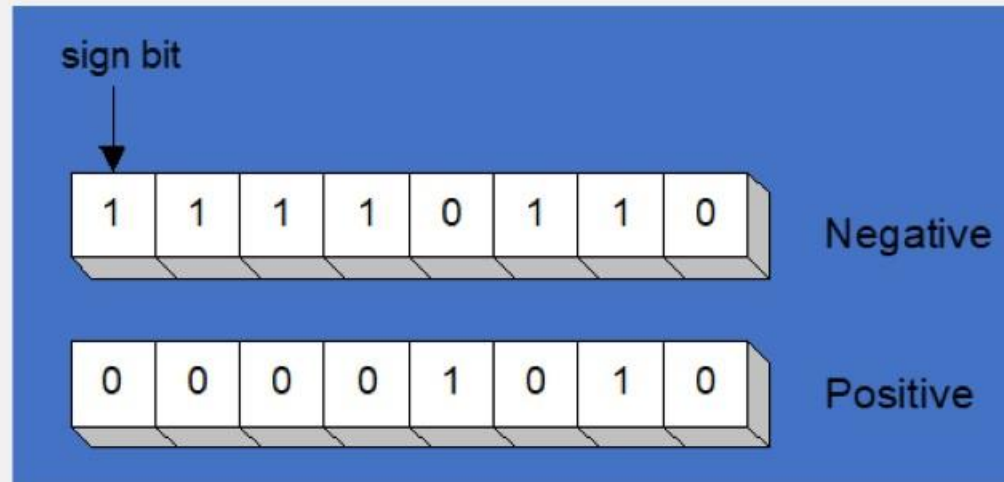
+46 = 0 101110

Then -46 = 1 010010

Number Systems (Part 2)

Sign Bit

Highest bit indicates the sign. 1 = negative, 0 = positive



If highest digit of a hexadecimal is > 7 , the value is negative

Examples: 8A and C5 are negative bytes

A21F and 9D03 are negative words

B1C42A00 is a negative double-word

Sign Extension

Step 1: Move the number into the lower-significant bits

Step 2: Fill all the remaining higher bits with the sign bit

- This will ensure that both magnitude and sign are correct
- Examples

- Sign-Extend 10110011 to 16 bits

10110011 = -77 →

11111111 10110011 = -77

- Sign-Extend 01100010 to 16 bits

01100010 = +98 →

00000000 01100010 = +98

- Infinite 0s can be added to the left of a positive number
- Infinite 1s can be added to the left of a negative number

- Exercise:

Find the 2's complement of B2FA.

Convert B2FA to binary ---- 1011 0010 1111 1010

The 2's complement of B2FA = 0100 1101 0000 0110

4 D 0 6 =(4D06)₁₆

Subtraction with Complements

- The subtraction of two n -digit unsigned numbers $M - N$ in base 2 can be done as follows:
1. Add the minuend, M , to the 2's complement of the subtrahend, N .
 2. If $M \geq N$, the end carry is discarded, what is left in the result is $M - N$.
 3. If $M < N$, there is no end carry. Then take the 2's complement of the result and place a negative sign in front.

Number Systems (Part 2)

Example 1

Compute: $1010100 - 1000011$

1). Form 2's complement of 1000011:

1000011
0111101

2). Add the two:

1010100
0111101
(+) -----
1 0010001 (a carry)

3). Since $M \geq N$, discard the carry. Ans. = 0010001

Number Systems (Part 2)

Example 2

Compute: $1000011 - 1010100$

- | | |
|--|--------------------|
| | 1010100 |
| 1). Form 2's complement of 1010100: | 0101100 |
| 2). Add the two: | 1000011 |
| | 0101100 |
| | (+) ----- |
| | 1101111 (no carry) |
| 3). Since $M < N$, complement the result. | |
| | Ans. = (-) 0010001 |

• Exercise:

Perform subtraction on the unsigned binary using the 2's complement subtrahend (show the steps)

1. $10011 - 10001 \rightarrow$ first compute the 2's complement Of $10001 = 01111$

2. 10011

01111

$100010 \rightarrow$ the result = 00010

Number Systems (Part 2)

Binary Codes

- Binary Coded Decimal (BCD 4-bit)
 - $(4267)_{10} = (0100\ 0010\ 0110\ 0111)_{BCD}$
 - $(9803)_{10} = (1001\ 1000\ 0000\ 0011)_{BCD}$
- The following configurations: 1010, 1011, 1100, 1101, 1110, and 1111 are not authorized in BCD
- Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.
 - $13_{10} = 1101_2$ (conversion) $\Leftrightarrow 0001 \mid 0011$ (coding)

➤ BCD Addition

- The **addition of BCD numbers** is slightly different from **binary addition**. Here, the rules of binary addition are partially applicable only to the individual 4-bit groups.
- Get the binary sum, if $\text{sum} > 9$, add 6 to the sum
- Obtain the correct BCD digit sum and a carry

Number Systems (Part 2)

BCD Addition

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

$$184 + 576 = 760$$

BCD carry	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

Number Systems (Part 2)

Weighted Codes

➤ The BCD code is the 8,4,2,1 code, and 8, 4, 2, and 1 are weights; so, BCD is a *weighted code*

Decimal	8,4,2,1	Excess3	8,4,-2,-1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0100
2	0010	0101	0110	0101
3	0011	0110	0101	0111
4	0100	0111	0100	0110
5	0101	1000	1011	0010
6	0110	1001	1010	0011
7	0111	1010	1001	0001
8	1000	1011	1000	1001
9	1001	1100	1111	1000

➤ Exercise

➤ Represent the unsigned decimal numbers 842 and 537 in BCD. and then show the steps needed to form their sum.

➤ 842 → (1000 0100 0010)_{bcd}
 ➤ 537 → (0101 0011 0111)_{bcd}

1000 0100 0010

0101 0011 0111

 1101 0111 1001

0110

0001 0011 0111 1001

➤ Exercise

➤ Represent the unsigned decimal numbers 537 in BCD.

Represent the decimal number 537 in

a. BCD → 0101 0011 0111

b. Excess 3 Code. → 1000 0110 1010

Logic Gates

- The building blocks used to create digital circuits are logic gates.
- There are three elementary logic gates and a range of other simple gates.
- Each gate has its own logic symbol which allows complex functions to be represented by logic diagrams.
- The function of each gate can be represented by a truth table or using Boolean notation.

- The table used to represent the Boolean expression of a logic gate function is commonly called a Truth Table.
- A logic gate truth table shows each possible input combination to the gate or circuit with the resultant output depending upon the combination of these input(s).

- For example, consider a single 2-input logic circuit with input variables labelled as A and B.
- There are "four" possible input combinations or 2^2 of "OFF" and "ON" for the two inputs.
- However, when dealing with Boolean expressions and especially logic gate truth tables, we do not general use "ON" or "OFF" but instead give them bit values which represent a logic level "1" or a logic level "0" respectively.



➤ Then the four possible combinations of A and B for a 2-input logic gate is given as:

0 0

0 1

1 0

1 1

➤ a 3-input logic circuit would have 8 possible input combinations or 2^3 and a 4-input logic circuit would have 16 or 2^4 , and so on as the number of inputs increases.

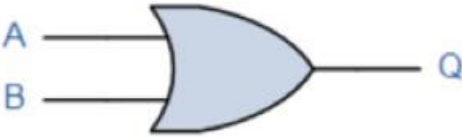
AND Gate

- Boolean Algebra Truth Tables For A 2-input AND Gate
- Note that the Boolean Expression for a two input AND gate can be written as: $A.B$ or just simply AB without the decimal point.

Symbol	Truth Table		
 2-input AND Gate	A	B	Q
	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = A.B$		Read as A AND B gives Q	

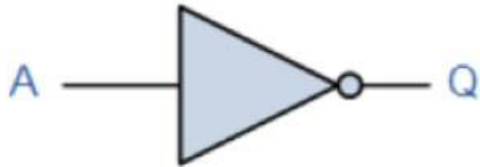
OR Gate

- Boolean Algebra Truth Tables For A 2-input OR Gate
- Note that the Boolean Expression for a two input OR gate can be written as: $A+B$.

Symbol	Truth Table		
 2-input OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression $Q = A+B$		Read as A OR B gives Q	

Inverter (Not Gate)

➤ For a single input NOT (Inverter) gate, the output Q is ONLY true when the input is "NOT" true, the output is the inverse or complement of the input giving the Boolean Expression of: ($Q = \text{NOT } A$).

Symbol	Truth Table	
 Inverter or NOT Gate	A	Q
	0	1
	1	0
Boolean Expression $Q = \text{NOT } A$ or $\bar{A}$		Read as inversion of A gives Q

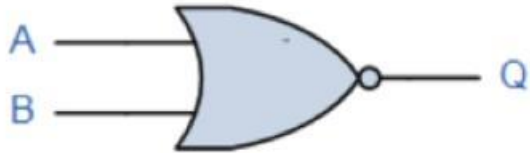
NAND (Not AND) Gate

➤ For a 2-input NAND gate, the output Q is NOT true if BOTH input A and input B are true, giving the Boolean Expression of: ($Q = \text{not}(A \text{ AND } B)$).

Symbol	Truth Table		
 2-input NAND Gate	A	B	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = \overline{A \cdot B}$	Read as A AND B gives NOT-Q		

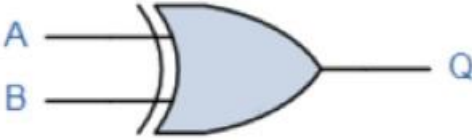
NOR (Not OR) Gate

➤ For a 2-input NOR gate, the output Q is true if BOTH input A and input B are NOT true, giving the Boolean Expression of: ($Q = \text{not}(A \text{ OR } B)$).

Symbol	Truth Table		
 2-input NOR Gate	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	0
Boolean Expression $Q = \overline{A+B}$		Read as A OR B gives NOT-Q	

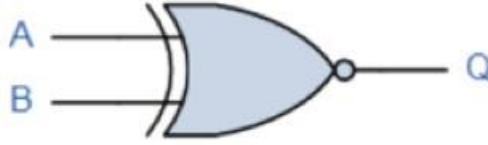
XOR (Exclusive OR) Gate

➤ For a 2-input Ex-OR gate, the output Q is true if EITHER input A or if input B is true, but NOT both giving the Boolean Expression of: ($Q = (A \text{ and NOT } B) \text{ or } (\text{NOT } A \text{ and } B)$).

Symbol	Truth Table		
 2-input Ex-OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A \oplus B$			

XNOR (Exclusive NOR) Gate

➤ For a 2-input Ex-NOR gate, the output Q is true if BOTH input A and input B are the same, either true or false, giving the Boolean Expression of: ($Q = (A \text{ and } B) \text{ or } (\text{NOT } A \text{ and } \text{NOT } B)$).

Symbol	Truth Table		
 2-input Ex-NOR Gate	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = \overline{A \oplus B}$			

Number Systems (Part 2)

Conclusion

- Most used number systems are decimal, binary and hexadecimal.
- Computers store data in the form of bits, bytes, and words using the binary numbering system.
- Hexadecimal numbers are formed using four-bit groups.
- Signed integers can be stored in one's complement, two's complement, or signed magnitude representation.
- Logic Gates are The building blocks used to create digital circuits

END OF Lecture 02