Lecture Three
Complements in Digital Computers

1. Why do we use complements?

In digital systems, subtraction is not performed directly like:
A-B
Instead, computers convert subtraction into **addition**, which is easier to implement in hardware:

$$A-B=A+(\text{complement of } B)$$

This simplifies:

- Circuit design
- Speed of operations
- Logical manipulation

2. The negative numbers

There are three types or methods to represent the negative numbers in a computer: -

1. Sign – and – Magnitude (signmagnitude).
2. 1's complement.
3. 2's complement.

Ex: Represent $(+ 12)_{10}$, $(- 12)_{10}$ in signmagnitude, 1's and 2's complement?
Sol :

Signmagnitude 1's comp. 2's comp.
+12 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0
-12 1 1 1 0 0 1 0 0 1 1 1 0 1 0 0

3. Types of Complements in Base r

◆ Two types (for each base-r system)

- ➢ Diminishing radix complement (r-1 complement)
- ➢ Radix complement (r complement)

◆ **For *n*-digit number *N***

$(r^n-1)-N$ → **r-1 complement**

r^n-N → **r complement**

Examples

- 9's complement of 674653
  - 10^6-1-674653 = 999999-674653 = 325346
- 9's complement of 023421
  - 10^6-1-023421 = 999999-023421 = 976578
- 10's complement of 674653
  - 10^6 – 674653= 325347
- 10's complement of 023421
  - 10^6 – 023421= 976579
- 1's complement of 10111001
  - 11111111 – 10111001 = 01000110
  - Simply replace 1's and 0's
- 1's complement of 10100010
  - 01011101
- 2's complement of 10111001
  - 01000110 + 1 = 01000111
  - Add 1 to 1's complement
- 2's complement of 10100010
  - 01011101 + 1 = 01011110

4. Subtraction with Complements of Unsigned

- *M – N*
  - *Add M to r's complement of N*
    - *Sum = M+(r^n – N) = M – N+ r^n*
  - *If M > N, Sum will have an end carry r^n , discard it*
  - *If M<N, Sum will not have an end carry and*
    - *Sum = r^n – (N – M) (r's complement of N – M)*

◆ *So M – N = – (r's complement of Sum)*

ملخص القاعدة في جدول

الحالة	r المتمم الـ	(r−1) المتمم الـ
M ≥ N (يوجد حمل)	**يُهمل الحمل**	**يُضاف الحمل إلى الناتج**(end-around carry)
M < N (لا يوجد حمل)	الناتج سالب = المتمم الـ r لمجموع الجمع	(r−1) الناتج سالب = المتمم الـ لمجموع الجمع

Example:

◆ **65438 – 05623**

In this case 5623 is less than 65438 so we will follow the first rule.

10^5 -05623= 94377

		6	5	4	3	8
+		9	4	3	7	7
	1	5	9	8	1	5
-	1	0	0	0	0	0
		5	9	8	1	5

◆ **5623 – 65438**

In this case 5623 is less than 65438 so we will follow the second rule.

10^5 -65438= 34562

	0	5	6	2	3
+	3	4	5	6	2
	4	0	1	8	5

ناخذ متتمة الناتج

-	5	9	8	1	5	

- **10110010 - 10011111**
- In this case **10011111**is less than **10110010** so we will follow the first rule.
  10011111→ 01100000+1→ 01100001

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

- **10011111-10110010**

  10110010→ 01001101+1→ 01001110

	1	0	0	1	1	1	1	1
+	0	1	0	0	1	1	1	0
	1	1	1	0	1	1	0	1
المتممة	-0	0	0	1	0	0	0	1
	0	0	0	1	0	0	1	1

- Perform the operation $(A\ B\ C\ E\ F)_{16}$ - (4 8 F 9 D)16 using (r – 1)'s comp.?

Solution

اولا نجد متممة 48F9D

Subtract each digit from **F (15 in hex)**

Digit	**F − Digit**	**Result**
4	F − 4	B
8	F − 8	7
F	F − F	0
9	F − 9	6
D	F − D	2

48F9D→B7062

ABCEF

+ B7062

162D51

1

62D52

5. Binary Codes

The digital data is represented, stored and transmitted as group of binary bits (binary code). The binary code is represented by the number as well as alphanumeric letter.

A. Advantages of Binary Codes

Some of the advantages that binary code offers can be summarized as,

- ✓ Binary codes are suitable for the computer applications.
- ✓ Binary codes are suitable for the digital communications.
- ✓ Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- ✓ Since only 0 & 1 are being used, implementation becomes easy.

Binary codes for *decimal digit require a minimum of four bits*. Numerous different codes can be obtained by arranging four or more bits in many distinct possible combinations. A few possibilities are shown in the table:-

Table:

Decimal Digit	(BCD) 8421	Excess-3 (BCD+0011)	8 4 2 1
0	0000	0011	0000
1	0001	0100	0111

Decimal Digit	(BCD) 8421	Excess-3 (BCD+0011)	8 4 2 1
2	0010	0101	0110
3	0011	0110	0101
4	0100	0111	0100
5	0101	1000	1011
6	0110	1001	1010
7	0111	1010	1001
8	1000	1011	1000
9	1001	1100	1111

In BCD code (Binary Coded Decimal) each decimal number is represented in 4 bits. In Excess – 3 Code each decimal number is represented by adding 3 to each number in BCD code.

Note:

With four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

B. Types of Binary Codes

I. BCD (Binary Coded Decimal)

Ex Convert $(13)_{10}$ to Binary, BCD code?

Sol:

- $(13)_{10} = (1101)_2$
- $(13)_{10} = (0001\ 0011)_{(BCD)}$

Ex Decode the following BCD numbers?

1 – $(1000111100010010101)_{(BCD)}$
2 – $(110110.01101)_{(BCD)}$

Sol:
1 – $(0100\ 0111\ 1000\ 1001\ 0101)_{(BCD)} = (47895)_{10}$

Ex Encode the following numbers into BCD code?

$1 - (1110011)_2$

Solution

$1 - 1110011 = 1\times2^0 + 1\times2^1 + 0\times2^2 + 0\times2^3 + 1\times2^4 + 1\times2^5 + 1\times2^6$
$= 1 + 2 + 0 + 0 + 16 + 32 + 64$
$= (115)_{10}$

$(115)_{10} = (0001\ 0001\ 0101)_{(BCD)}$

Ex What are the Ex-3 of the following numbers?

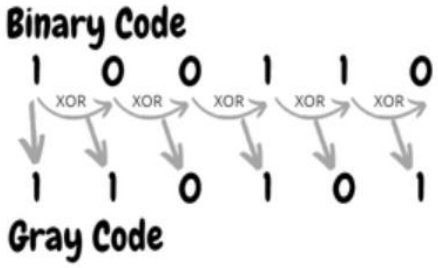
$1 - (365)_{10}$

Sol:

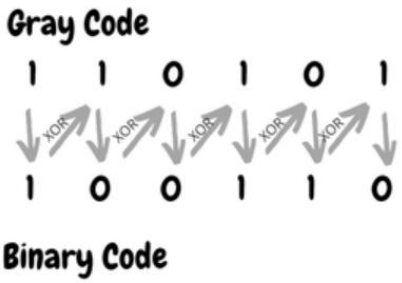
3 6 5
+3 +3 +3
_____
6 9 8 = $(0110\ 1001\ 1000)_{(Ex\text{-}3)}$

II. Gray Code

- Only **one bit changes** between consecutive numbers
  ✓ Used in error reduction systems.
- To convert a number from binary to Gray code the relations must be known:
- $0 \oplus 0 = 0$ $0 \oplus 1 = 1$ $1 \oplus 1 = 0$ $1 \oplus 0 = 1$



Binary code to gray code conversion



Gray code to binary code conversion

III. Alphanumeric Codes

Alphanumeric codes are the codes that represent numbers and alphabetic characters. The following three alphanumeric codes are very commonly used for the data representation:

- American Standard Code for Information Interchange (ASCII).
- Extended Binary Coded Decimal Interchange Code (EBCDIC).
- Five bit Baudot Code.
- ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code. ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.
- **ASCII Code**
- The standard binary code for the alphanumeric characters is called ASCII (American Standard Code for Information Interchange). It uses 7 bits to code 128 characters, as shown in the table. The seven bits of the code are designed by A0 through A6, with A6 being the most significant bit. For example, the letter A is represented in ASCII as (1000001).
- The ASCII code contains **94** characters that can print and **34** nonprinting used in control functions. The printing characters consist of **26** uppercase letters, the **26** lowercase letters, **10** numerals, and **32** special printable character such as %, @ and $.

- **Ex**: What are the characters corresponding to the ASCII code?
- (01001010100111110010001001110010000010001001000101101011O)**ASCII**
- Using the groups from the image's solution:

7-bit Binary	Decimal	Character
1001010	74	**J**
1001111	79	**O**
1001000	72	**H**
1001110	78	**N**
0100000	32	**space**
1000100	68	**D**
1000101	69	**E**
1010110	86	**V**