

# نمایش اعداد

# سیستم نمایش اعداد

## • مبنا (base):

➤ مبنا  $r$ : ارقام محدود به  $[0, r-1]$

- دسیمال:  $(379)_{10}$

- باینری:  $(01011101)_2$

- اکتال:  $(372)_8$

- هگزادسیمال:  $(23D9F)_{16}$

## • نیازها:

➤ محاسبات در هر سیستم

➤ تبدیل از یک سیستم به سیستم دیگر

# سیستم نمایش اعداد (دسیمال)

◀ اعداد دسیمال:

– دو بخش صحیح و اعشاری

$$A_{n-1} A_{n-2} \dots A_1 A_0 . A_{-1} A_{-2} \dots A_{-m+1} A_{-m}$$

که  $A_i$  عددی بین 0 تا 9 و با وزن  $10^i$  است.

# سیستم نمایش اعداد (دسیمال)

The value of

$$A_{n-1} A_{n-2} \dots A_1 A_0 . A_{-1} A_{-2} \dots A_{-m+1} A_{-m}$$

is calculated by

$$\sum_{i=n-1..0} (A_i * 10^i) + \sum_{i=-m..-1} (A_i * 10^i)$$

مثال:

$$(126.53)_{10}$$


$$= 1*10^2 + 2*10^1 + 6*10^0 + 5*10^{-1} + 3*10^{-2}$$

# سیستم نمایش اعداد (حالت کلی)

- “base”  $r$

- $$N = A_{n-1} * r^{n-1} + A_{n-2} * r^{n-2} + \dots + A_1 * r + A_0 + A_{-1} * r^{-1} + A_{-2} * r^{-2} + \dots + A_{-m} * r^{-m}$$

Most  
Significant  
Digit (MSD)



Least  
Significant  
Digit (LSD)



## سیستم نمایش اعداد (حالت کلی)

• مثال:  $r = 6$

$$(312.4)_6 = 3*6^2 + 1*6^1 + 2*6^0 + 4*6^{-1} \\ = (116.66)_{10}$$

◀ تبدیل از مبنای  $r$  به مبنای 10 با رابطه بالا انجام می شود.

# اعداد باینری (مبنای 2)

کامپیوترها داده ها را به صورت رشته ای از “بیت ها” نمایش می دهند.

- بیت: 0 یا 1

مبنای 2: ارقام 0 یا 1

• مثال:

$$(101101.10)_2 = 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2}$$

$$\begin{aligned} \text{(in decimal)} &= 32 + 0 + 8 + 4 + 0 + 1 + \frac{1}{2} + 0 \\ &= (45.5)_{10} \end{aligned}$$

# اعداد باینری (مبنای 2)

• مثال:

$$(1001.011)_2 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 + \\ 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$$

$$\text{(in decimal)} = 8 + 1 + 0.25 + 0.125 \\ = (9.375)_{10}$$



# اعداد باینری

$$\begin{array}{cccccc} 32 & 16 & 8 & 4 & 2 & 1 & .5 & .25 & .125 & .0625 \\ ( 1 & 1 & 0 & 1 & 0 & 1 & . & 1 & 0 & 1 & 1 )_B = ( 53.6875 )_D \end{array}$$

## توان های 2

n	2 <sup>n</sup>	n	2 <sup>n</sup>	n	2 <sup>n</sup>
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

**Memorize** at least through 2<sup>12</sup>

# اعداد اکتال (مبنای 8)

- مبنای 8:

◀ ارقام 0 تا 7

- مثال:

$$\begin{aligned}(762)_8 &= 7*8^2 + 6*8^1 + 2*8^0 \\ \text{(in decimal)} &= 448 + 48 + 2 \\ &= (498)_{10}\end{aligned}$$

# اعداد هگزادسیمال (مبنای 16)

• مبنای 16:

ارقام 0, ..., 9, A, B, C, D, E, F

A=10, B=11, ..., F = 15

• مثال:

$$\begin{aligned}(3FB)_{16} &= 3 * 16^2 + 15 * 16^1 + 11 * 16^0 \\ (\text{in decimal}) &= 768 + 240 + 11 \\ &= (1019)_{10}\end{aligned}$$

# تبدیل میناها

➤ هر مینا (۲) ← دسیمال (گفته شده)

➤ دسیمال ← هر مینای ۲

➤ اکتال ← باینری و برعکس

➤ هگزادسیمال ← باینری و برعکس

# تبدیل دسیمال به هر مبنای r

$$34,761_{10} = (?)_{16}$$

• بخش صحیح:

- تقسیم متوالی بر r تا زمانی که خارج قسمت برابر با صفر شود
- خواندن باقیمانده ها به بالا.

$$\begin{array}{r} 16 \overline{) 34,761} \\ 16 \overline{) 2,172} \text{ rem } 9 \\ 16 \overline{) 135} \text{ rem } 12 = C \\ 16 \overline{) 8} \text{ rem } 7 \\ 0 \text{ rem } 8 \end{array}$$

Read up

$$34,761_{10} = 87C9_{16}$$

# تبدیل دسیمال به هر مبنای r

• بخش اعشاری:

• ضرب متوالی در r تا زمانی که بخش اعشاری صفر شود

• خواندن بخشهای صحیح رو به پایین  
 $0.78125_{10} = (?)_{16}$

$$0.78125 \times 16 = 12.5 \quad \text{int} = 12 = C$$

$$0.5 \times 16 = 8.0 \quad \text{int} = 8$$

Read  
down

$$0.78125_{10} = 0.C8_{16}$$

# تبدیل دسیمال به هر مبنای r

• مثال دیگر

$$0.1_{10} = (?)_2$$

$$0.1 \times 2 = 0.2 \quad \text{int} = 0$$

$$0.2 \times 2 = 0.4 \quad \text{int} = 0$$

$$0.4 \times 2 = 0.8 \quad \text{int} = 0$$

$$0.8 \times 2 = 1.6 \quad \text{int} = 1$$

$$0.6 \times 2 = 1.2 \quad \text{int} = 1$$

$$0.2 \times 2 = 0.4 \quad \text{int} = 0$$

$$0.4 \times 2 = 0.8 \quad \text{int} = 0$$



Read  
down

$$0.1_{10} = 0.\overline{00011}_2$$



## اعداد در مبناهاي مختلف

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

**Memorize** at least Binary and Hex

# باینری به اکتال

## باینری به هگزادسیمال

### • باینری به اکتال

$$8 = 2^3 \blacktriangleleft$$

← هر ۳ بیت باینری به یک رقم اکتال تبدیل می شود.

### • باینری به هگزادسیمال

$$16 = 2^4 \blacktriangleleft$$

← هر ۴ بیت باینری به یک رقم هگزادسیمال تبدیل می شود.

# Binary $\leftrightarrow$ Octal

$(11010101000.1111010111)_2$

$(011|010|101|000|.|111|101|011|100)_2$

$(\begin{array}{ccccccc} \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ 3 & 2 & 5 & 0 & . & 7 & 5 & 3 & 4 \end{array})_8$

# Binary $\leftrightarrow$ Hex

$(110\ 1010\ 1000\ .\ 1111\ 0101\ 11)_2$

$(0110|1010|1000|.|1111|0101|1100)_2$



$(6\ A\ 8\ .\ F\ 5\ C)_{16}$

# Octal ↔ Hex

• از طریق باینری انجام دهید:

Hex → Binary → Octal

Octal → Binary → Hex

# تبدیل ها (مثال)

• جدول را پر کنید:

Decimal	Binary	Octal	Hex
329.3935	?	?	?
?	10101101.011	?	?
?	?	336.5	?
?	?	?	F9C7.A

# اعمال ریاضی باینری: جمع

• قوانین: مانند جمع دسیمال

• با این تفاوت که  $1+1 = 10$  ← تولید نقلی

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

$0+0 = 0(c0)$  (sum 0 with carry 0) <

$0+1 = 1+0 = 1(c0)$  <

$1+1 = 0(c1)$  <

$1+1+1 = 1(c1)$  <

نقلی	1	1	1	1	1	0
مضاف	0	0	1	0	0	1
مضاف الیه	0	1	1	1	1	1
نتیجه	1	0	1	0	0	0

# اعمال ریاضی باینری: تفریق

• قوانین:

$0-0 = 1-1 = 0$  (b0) (result 0 with borrow 0)  $\swarrow$

$1-0 = 1$  (b0)  $\swarrow$

$0-1 = 1$  (b1)  $\swarrow$

...  $\swarrow$

$X$	229	1	1	1	0	0	1	0	1
$Y$	- 46	-	0	0	1	0	1	1	0
$X - Y$	183	1	0	1	1	0	1	1	1

رقم قرصی	1	1	0	0	
مفروق	1	1	0	1	1
مفروق الیه	0	1	1	0	1
نتیجه	0	1	1	1	0



# نمایش اعداد

- نمایش اعداد مثبت:

➤ در بیشتر سیستم ها یکسان است.

- نمایش اعداد منفی:

➤ اندازه-علامت (Sign magnitude)

➤ مکمل ۱ (1's complement)

➤ مکمل ۲ (2's complement)

- در بیشتر سیستم ها: مکمل ۲

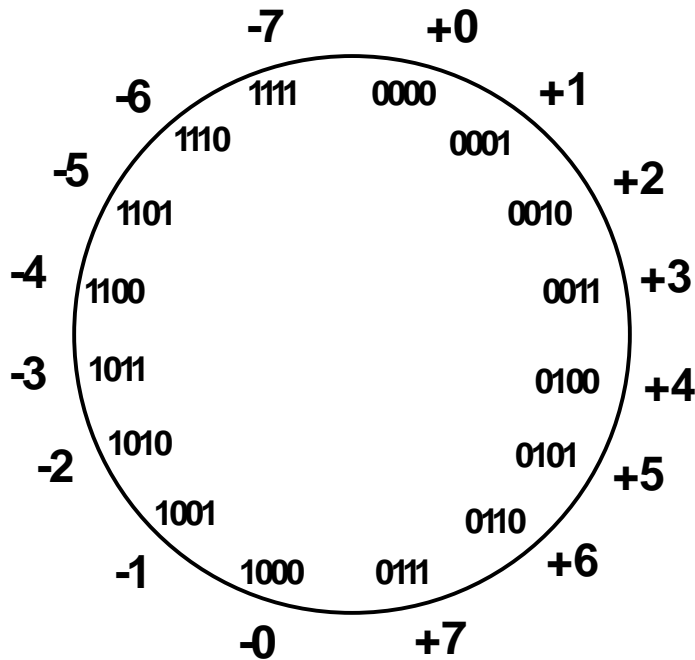
- فرض در اسلایدهای بعدی:

➤ ماشین با کلمه های ۴ بیتی:

- ← ۱۶ مقدار مختلف قابل نمایش.

- تقریباً نیمی مثبت، نیمی منفی.

# نمایش اعداد



اندازه-علامت:

0 100 = +4

1 100 = -4

High order bit is sign: 0 = positive (or zero), 1 = negative

Three low order bits show the magnitude: 0 (000) thru 7 (111)

Number range for n bits =  $-(2^{n-1} - 1)$  to  $(2^{n-1} - 1)$

Two representations for 0

# نمایش اعداد

مکمل ۱:

If  $N$  is a positive number, then its 1's complement,  $\overline{N}$ , is defined as:

$$\overline{N} = (2^n - 1) - N$$

Example: 1's complement of 7

$$2^4 = 10000$$

$$\begin{array}{r} - \\ 1 \end{array} = \underline{00001}$$

1111

$$\begin{array}{r} - \\ 7 \end{array} = \underline{0111}$$

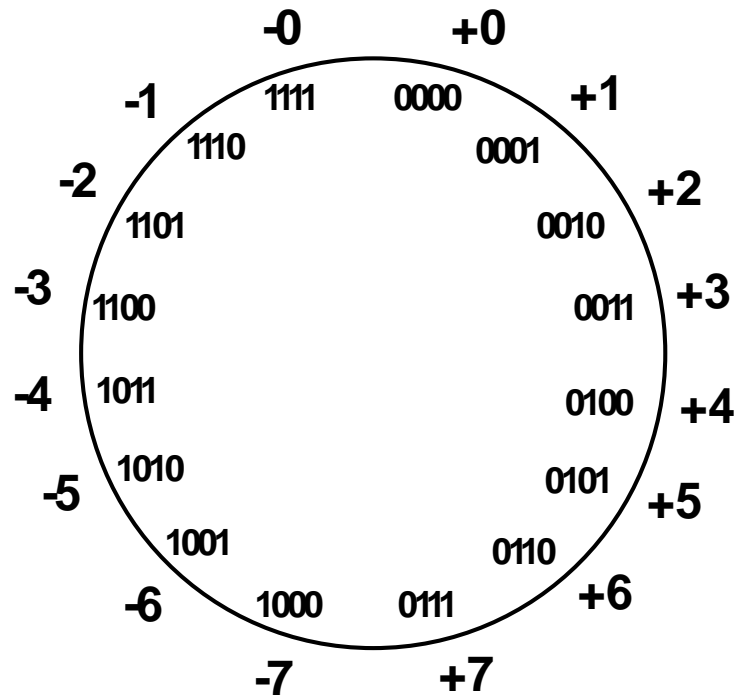
1000 = -7 in 1's comp.

Shortcut method:

Simply perform bit-wise complement

0111  $\rightarrow$  1000

# نمایش اعداد



مکمل ۱:

0 100 = +4

1 011 = -4

-

Subtraction implemented by addition & 1's complement

Still two representations of 0! This causes some problems

Some complexities in addition

# نمایش اعداد

If N is a positive number, then its 2's complement,  $N^*$ , is defined as:

$$N^* = 2^n - N$$

Example: 2's complement of 7

$$2^4 = 10000$$

$$\text{sub } 7 = \underline{0111}$$

$$1001 = \text{repr. of } -7$$

مکمل ۲:

Example: 2's complement of -7

$$2^4 = 10000$$

$$\text{sub } -7 = \underline{1001}$$

$$0111 = \text{repr. of } 7$$

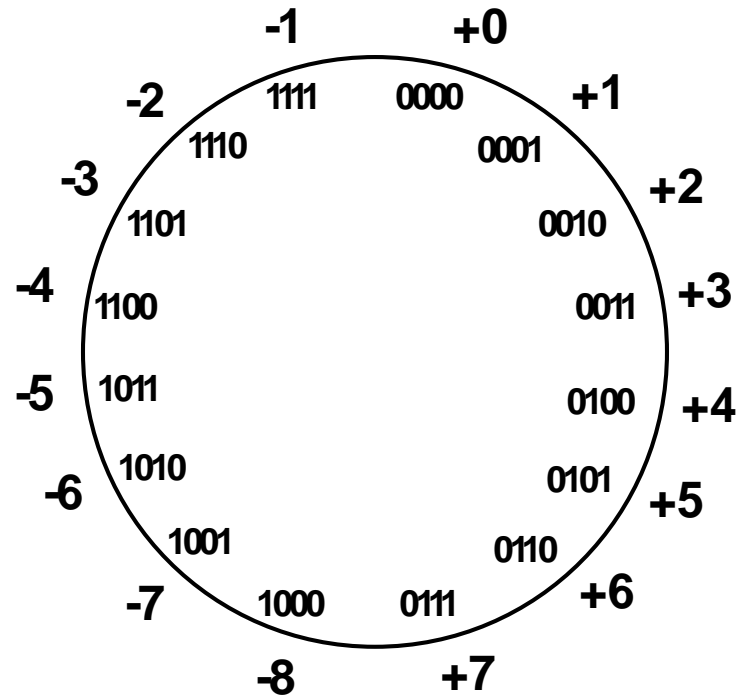
Shortcut method:

2's complement = bit-wise complement + 1

0111  $\rightarrow$  1000 + 1  $\rightarrow$  1001 (representation of -7)

1001  $\rightarrow$  0110 + 1  $\rightarrow$  0111 (representation of 7)

# نمایش اعداد



مکمل ۲:

0 100 = +4

1 100 = -4

Only one representation for 0

One more negative number than positive number (-8)

Number range for n bits =  $-(2^{n-1})$  to  $(2^{n-1} - 1)$

# جمع و تفریق مکمل ۲

4	0100	-4	1100
<u>+ 3</u>	<u>0011</u>	<u>+ (-3)</u>	<u>1101</u>
7	0111	-7	<b>1</b> 1001

4	0100	-4	1100
<u>- 3</u>	<u>1101</u>	<u>+ 3</u>	<u>0011</u>
1	<b>1</b> 0001	-1	1111

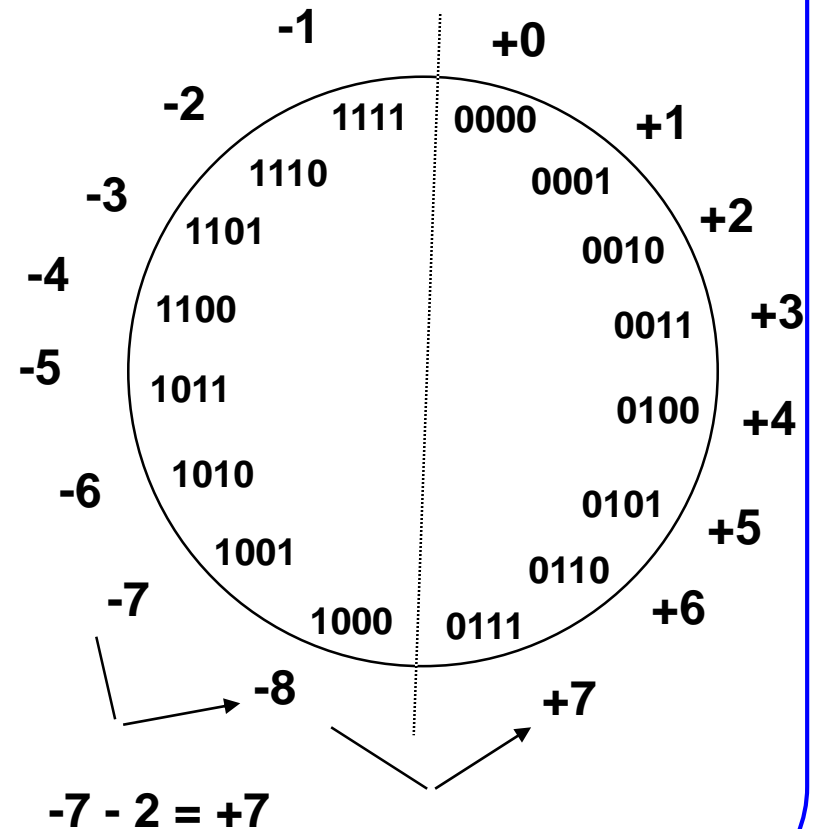
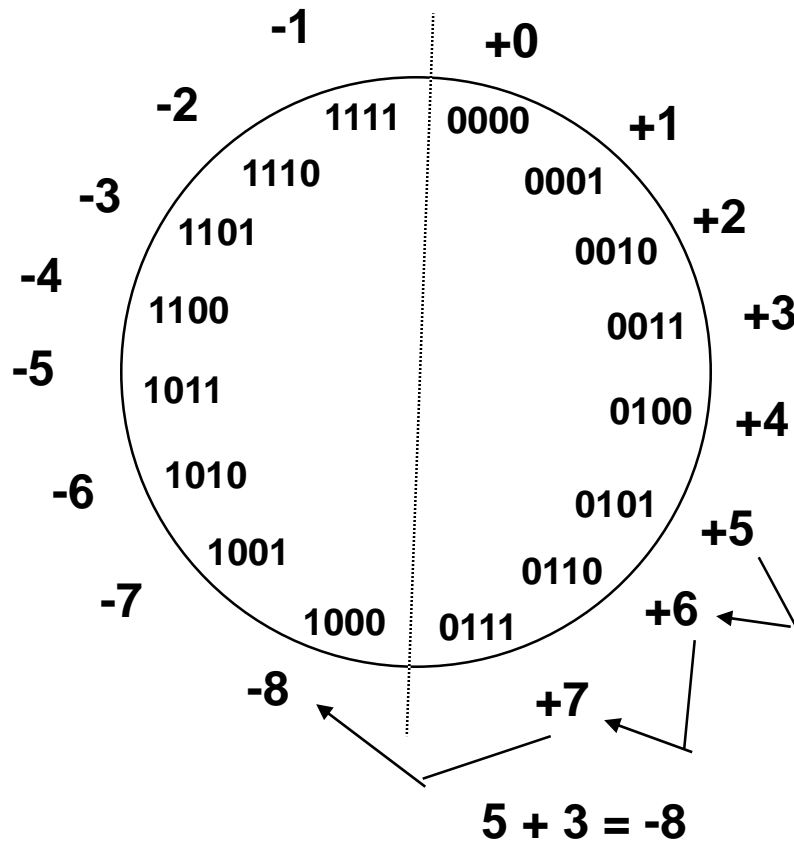
Simpler addition scheme makes 2's complement the most common choice for integer number systems within digital systems

## Overflow Conditions:

سریز

1. Adding two positive numbers to get a negative number:

2. Adding two negative numbers to get a positive number:





# سریز

## Overflow Conditions:

$$\begin{array}{r} 5 \quad \quad 0111 \\ \quad \quad 0101 \\ \hline 3 \quad \quad 0011 \\ -8 \quad \quad 1000 \end{array}$$

Overflow

$$\begin{array}{r} -7 \quad \quad 1000 \\ \quad \quad 1001 \\ \hline -2 \quad \quad 1110 \\ 7 \quad \quad 10111 \end{array}$$

Overflow

$$\begin{array}{r} 5 \quad \quad 0000 \\ \quad \quad 0101 \\ \hline 2 \quad \quad 0010 \\ 7 \quad \quad 0111 \end{array}$$

No overflow

$$\begin{array}{r} -3 \quad \quad 1111 \\ \quad \quad 1101 \\ \hline -5 \quad \quad 1011 \\ -8 \quad \quad 11000 \end{array}$$

No overflow

**Method 1: Overflow when the carry into sign  $\neq$  carry out**

**Method 2: Overflow when  $\text{sign}(A) = \text{sign}(B) \neq \text{sign}(\text{result})$**

## ضرب باینری

- Shift-and-add algorithm, as in decimal

M'cand	0	0	0	1	1	0	1
M'plier	0	0	0	0	1	1	0
(1)			0	0	0	0	0
(2)		0	1	1	0	1	
(3)	0	1	1	0	1		
Sum	1	0	0	1	1	1	0

- Check:  $13 * 6 = 78$

# Binary-Coded Decimal (BCD)

- ◀ A decimal code:  
Decimal numbers (0..9)  
are coded using 4-bit  
distinct binary words
- ◀ Observe that the codes  
1010 .. 1111 (decimal  
10..15) are NOT  
represented (invalid  
BCD codes)

□ **TABLE 1-3**  
**Binary-Coded Decimal (BCD)**

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

**Table 1-3 Binary-Coded Decimal (BCD)**

# Binary-Coded Decimal

- To code a number with  $n$  decimal digits, we need  $4n$  bits in BCD

e.g.  $(365)_{10} = (0011\ 0110\ 0101)_{\text{BCD}}$



- This is different from converting to binary, which is  $(365)_{10} = (101101101)_2$
- Clearly, BCD requires more bits. BUT, it is easier to understand/interpret

# BCD Addition

Case 1:

$$\begin{array}{r} 0001 \quad 1 \\ 0101 \quad 5 \\ \hline (0) 0110 \quad (0) 6 \end{array}$$

Case 2:

$$\begin{array}{r} 0110 \quad 6 \\ 0101 \quad 5 \\ \hline (0) 1011 \quad (1) 1 \end{array}$$

**WRONG!**

Case 3:

$$\begin{array}{r} 1000 \quad 8 \\ 1001 \quad 9 \\ \hline (1) 0001 \quad (1) 7 \end{array}$$

Note that for cases 2 and 3, adding number 6 (0110) gives us the correct result.

**How can we identify the wrong cases?**

## BCD Addition (cont.)

- **BCD addition is therefore performed as follows:**

1) Add the two BCD digits together using normal binary addition

2) Check if correction is needed

a) 4-bit sum is in range of 1010 to 1111

**OR**

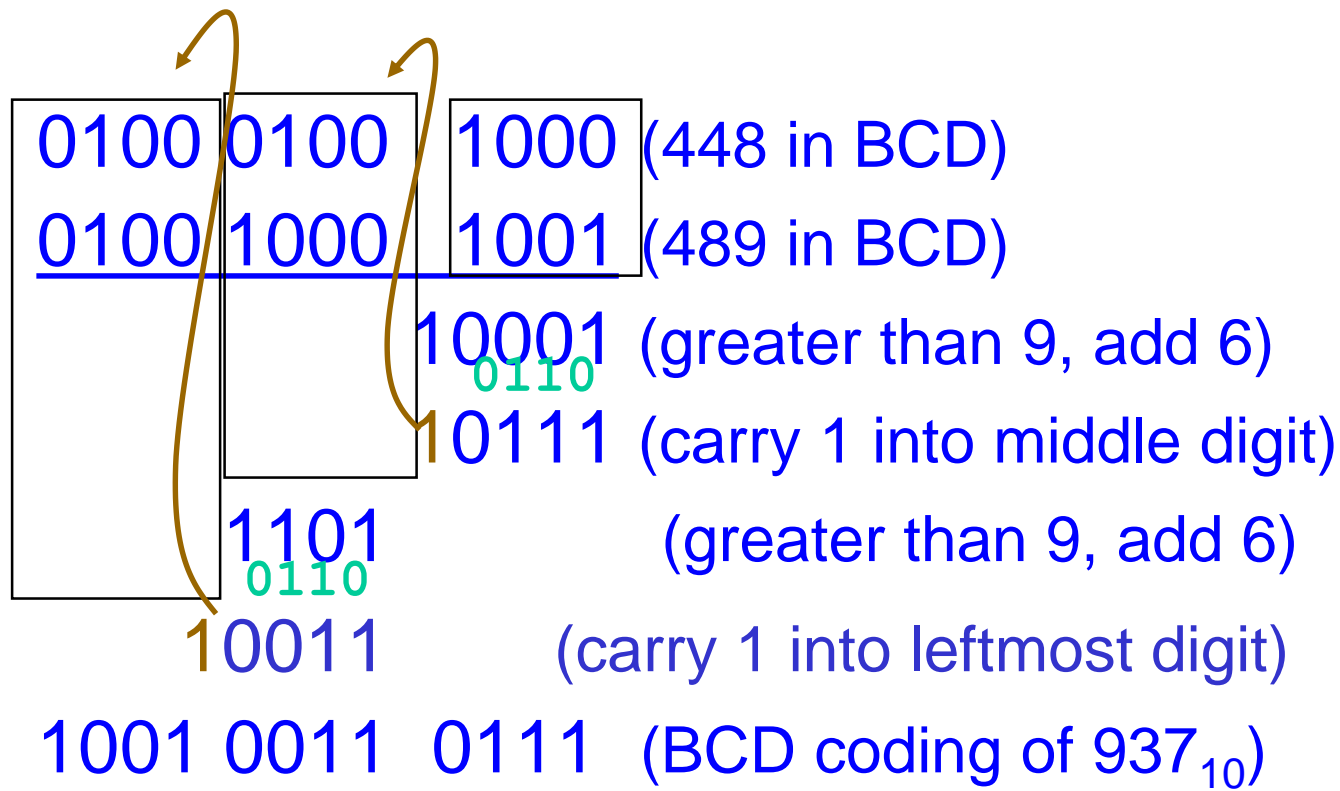
b) carry out of MSB = 1

3) If correction is required, add 0110 to 4-bit sum to get the correct result

→ **BCD carry out = 1**

# BCD Addition (cont.)

- **Example: Add 448 and 489 in BCD.**



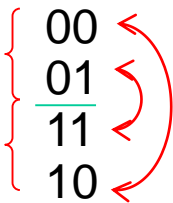
# Gray Codes

- Gray codes are *minimum change codes*
  - ◀ From one numeric representation to the next, only one bit changes
  - ◀ Applications:
    - Later.

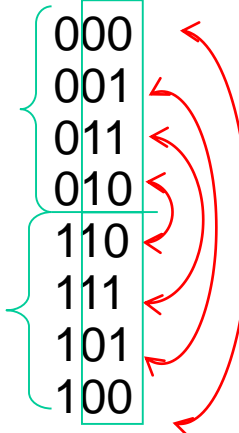


# Gray Codes (cont.)

Binary	Gray
00	00
01	01
10	11
11	10



Binary	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100



Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

