* Signed Binary numbers

Computer must represent everything with binary digits. It is customary to represent the sign with a bit placed in the leftmost position of the number.

0 for positive(+)

1 for negative(−)

* There is only one way to represent positive numbers in Computer.

The signed magnitud number

To represent +9 in Computer we need at least 5 bits

4 for the number (magnitude)

1 for the sign

$$\underset{\substack{\uparrow \\ +}}{\underline{0}}\ \underbrace{\underline{1}\ \underline{0}\ \underline{0}\ \underline{1}}_{\text{magnitude}}$$

for unsigned 9 we need only 4 bits

$$\underline{1}\ \underline{0}\ \underline{0}\ \underline{1}$$

$$\#\text{ of bits} = \lceil \log_2(\text{number}) \rceil$$

ceil

$\log_2(9) = 3.1699$

$\Rightarrow \#$ of bits $= 4$

signed

$$(\underbrace{0}_{+}\underbrace{1011}_{11}) \longrightarrow (+11)_d$$

To represen +9 using 8 bits

$$\underset{(+)}{0}\ 0\ 0\ 0\ \underbrace{1\ 0\ 0\ 1}_{\text{magnitude}}$$

* There are three ways to represent negative numbers

① Signed magnitude

to represent −9 we need at least 5 bits

$$\underset{(-)}{1}\ \underbrace{1\ 0\ 0\ 1}_{\text{magnitude}}$$

to represent −9 in 8-bit representation

$$\underset{(-)}{1}\ 0\ 0\ 0\ \underbrace{1\ 0\ 0\ 1}_{\text{magnitude}}$$

② First Complement (1's complement)

we first write +9 0 1 0 0 1

then find the 1's complement of (0 1 0 0 1)

$$\underbrace{1\ 0\ 1\ 1\ 0}_{-9}$$

③ Two's complement (2's complement)

we add 1 to the first complement

−9 in 2's complement representation

$$\begin{array}{r} 10110 \\ +\quad 1 \\ \hline 10111 \end{array} \qquad \underbrace{10111}_{-9}$$

To find the number that can be represented as 2's complement as 11011 we should go through the following

$\underset{=}{1}1011$
(−)

(1) Since the 2's complement starts with 1 then the number is negative

(2) find the firs complement of the number

11011
00100

(3) find the two's complement by adding 1

$$\begin{array}{r} 00100 \\ +\quad 1 \\ \hline \underbrace{00101}_{\text{magnitude}} \end{array} \longrightarrow \text{this number equals } 3$$

Then 11011 is equal to −3 in 2's complement representation

To Sum :-

for positive numbers the signed magnitud, 1's complement and 2's complement have the same representation

for negative numbers we have three different representation for 4-bit numbers the following table summarize the range and the different representations for all positive and negative numbers

Decimal	signed magnitud	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	—
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	— need 5 bits	—	1000

unsigned	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
—	—

The range for usigned number $(0) - (2^n - 1)$ (0)-(15)

The range for Signedmagnitude $(-(2^{n-1}-1) - (2^{n-1}-1))$ (-7)-(+7)

The rang for 1's complement $(-(2^{n-1}-1) - (2^{n-1}-1))$ (-7)-(+7)

The range for 2's complement $(-(2^{n-1}-1) - (2^{n-1}))$ (-7)-(+8)

Example: given that A = -15, B = 25, do the following operations using 2's complement in 6-bit representation

① A + B

(-15) + (25)

To represent (-15) in 2's complement we first write +15

0 0 1 1 1 1

1's complement 1 1 0 0 0 0

2's complement 1 1 0 0 0 1 → (-15)

+25 in 2's complement is the same as +25 in sign magnitude

	16	8	4	2	1
0	1	1	0	0	1

≡ 2's complement of (+25)

similar

1 1 0 0 0 0

∴ A + B

```
   110001
 + 011001
 --------
   001010
```

→ the final answer

If the last two carries are similar to each other, then there is no overflow and carry is discarded

② A − B

-15 -25 ≡ (-15) + (-25) = 2's complement of (+15) + 2's complement of (+25)

not similar to each other

1 0 0 1 1 1

```
   110001    (-15)
 + 100111    (-25
 --------
 1 011000
```

overflow

If the last two carries are not similar then there is overflow and the final answer = (2's complement (answer))

-40

The final answer will be 2's complement of (011000)

$$-\begin{array}{cccccc} 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

$= -40$

* Binary Codes:-

- Suppose we have language with 4 symbols (A, B, C, D)

\# of bits to code $= \lceil \log_2(4) \rceil = 2$

Symbol:-	A	B	C	D
Code:-	00	01	10	11

- Suppose we have language with 8 symbols (A, B, C, D, 0, 1, 2, 3)

Symbol	A	B	C	D	0	1	2	3
Code	000	001	010	011	100	101	110	111

- ASCII Code

standard ASCII code:- 7-bit (0-127)

extended ASCII code:- 8-bit (0-255)

please refer to the table page 64 in the slides

Examples:-

$(13)_d \longrightarrow (\quad\quad)_2$ Conversion

$(13)_d \longrightarrow (\quad\quad)_{ASCII}$ Coding

2	13
1	6
0	3
1	1
1	0

$(13)_d \longleftrightarrow (1101)_2$

to code (13)

1 should be represented in 8-bits

3 " " " " "

from the table pag 64 in the slides

$\underset{ASCII}{1} \Longleftrightarrow (31)_H \Longleftrightarrow (00110001)_2$

$\underset{ASCII}{3} \Longleftrightarrow (33)_H \Longleftrightarrow (00110011)$

$\therefore (13)_d \Longleftrightarrow (\underbrace{00110001}_{1}\,\underbrace{00110011}_{3})_{ASCII}$

- BCD Code (Binary Coded Decimal)

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	XXXX
11	XXXX
12	XXXX
13	XXXX
14	XXXX
15	XXXX

unused (6 codes are unused)

$(13)_d \xleftrightarrow{\text{conversion}} (1101)_2$

$(13)_d \xleftrightarrow{\text{coding}} (\underbrace{0001}_{1}\,\underbrace{0011}_{3})_{BCD}$

$(10)_d \xleftrightarrow{\text{coding}} (00010000)_{BCD}$

$(185)_d \xleftrightarrow{\text{coding}} (\underbrace{0001}_{1}\,\underbrace{1000}_{8}\,\underbrace{0101}_{5})_{BCD}$

$(10)_d \xleftrightarrow{\text{conversion}} (1010)_2$

$(185)_d \longleftrightarrow (10111001)_2$

- Addition with BCD

4 + 4

$$\begin{array}{r} 0100 \\ +\ 0100 \\ \hline \underbrace{1000}_{8} \end{array}$$

valid because less then 10

(2) 4 + 5

$$\begin{array}{r} 0100 \\ +\ 0101 \\ \hline \underbrace{1001}_{9} \end{array}$$

Valid less than 10

(3) 5 + 5

$$\begin{array}{r} 0101 \\ +\ 0101 \\ \hline \underbrace{1010}_{10\ \text{not included in BCD}} \end{array}$$

→ not valid = 10 So to make it valid we should add 6 (unused codes)

$$\begin{array}{r} ^{1\,1\,1} \\ 1010 \\ +\ 0110 \\ \hline \underbrace{\underbrace{0001}_{1}\ \underbrace{0000}_{0}}_{10} \end{array}$$

10 final answer

(4) 6 + 6

$$\begin{array}{r} ^{1\,1} \\ 0110 \\ +\ 0110 \\ \hline \underbrace{1100}_{12} \end{array}$$

not valid so we add 6

$$\begin{array}{r} ^{1\,1} \\ 1100 \\ +\ 0110 \\ \hline \underbrace{0001}_{1}\ \underbrace{0010}_{2} \end{array}$$

the final answer is $\boxed{12}$

Example:- using BCD find

```
   1 8 5
 + 9 2 5
```

	1	8	5
	0001	1000	0101
+	1001	0010	0101
	1011	1011	1010
+	0110	0110	0110
1	0001	0001	0000
0001			
1	1	1	0

final answer 1110

- Gray Code

The Gray Code (Reflected Binary Code) is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit)

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

- Conversion from Binary to gray

$(100110)_2 \longleftrightarrow (\quad\quad)_{gray}$

The most significant bit of the gray code is always equals to the most significant bit in the binary code

other bits of the gray code can be obtained by XOR binary code bits at the index and previous index

binary: 1 0 0 1 1 0

XOR

gray: 1 1 0 1 0 1

XOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

odd function

$(110101)_{gray}$

- from gray to binary

The most significant Bit (MSB) of the binary code is always equals to the MSB in the gray code.

other bits can be obtained by checking gray code bit at the index. If current gray code bit is 0, then copy previous binary code bit, else copy the invert of previous binary code bit

$(1\ 1\ 0\ 1\ 0\ 1)_{gray} \longleftrightarrow (\quad\quad)_2$

if 0 copy

If 1 invert

invert copy invert

1 0 0 1 1 0

$(100110)_2 \longrightarrow (38)_d$

* Other Decimal Codes

Decimal	BCD 8 4 2 1	5 4 2 1	2 4 2 1	8 4 -2 -1	Excess-3
0	0000	0000	0000	0000	0011
1	0001	0001	0001	0111	0100
2	0010	0010	0010	0110	0101
3	0011	0011	0011	0101	0110
4	0100	0100	0100	0100	0111
5	0101	1000	1011	1011	1000
6	0110	1001	1100	1010	1001
7	0111	1010	1101	1001	1010
8	1000	1011	1110	1000	1011
9	1001	1100	1111	1111	1100
unused	–	–	–	–	–

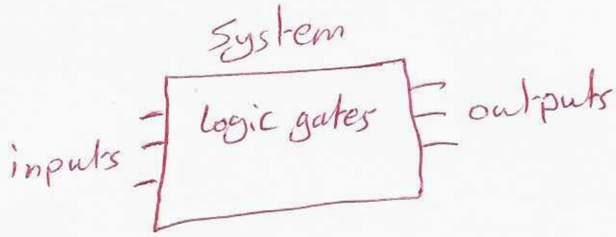
excess-3 = BCD + 3

BCD, 5421, 2421 and 84-2,-1 are weighted codes

2421, 84 -2 -1, and EXCess-3 are self complementary codes (each codeword is its own complement when read in reverse order)

0011 is the complement of 1100
first code — last code

Binary logic Gates



① AND Gate

And gate has 2 or more inputs

Graphical symbol

Truth table

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

mathematical expression

$Z = X$ and Y

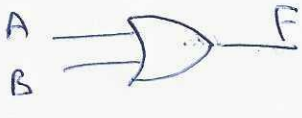
$X \cdot Y$

Z equals to 1 when all inputs equal to 1

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$F = A \cdot B \cdot C$

② OR Gate



$F = A + B$

A OR B

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

F equals to 0 when all inputs equal to 0

(3) Buffer

$F = A$

A	F
0	0
1	1

To delay an input

(4) NOT (inverter)

$F = \bar{A}$

$= A'$

A^c

A	F
0	1
1	0

(5) NAND Gate

Not AND

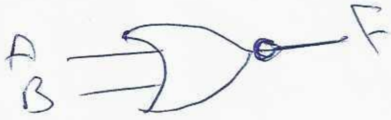
$F = \overline{(A \cdot B)}$

$= \bar{A} + \bar{B}$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

(6) NOR Gate

Not OR

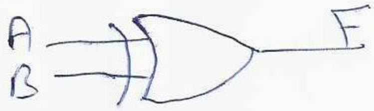


$F = \overline{(A + B)}$

$= \bar{A} \cdot \bar{B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

(7) XOR Gate



$F = \cancel{A}$ XOR B

$= A \oplus B$

$= \bar{A}B + A\bar{B}$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Odd function

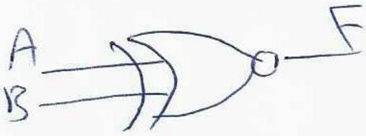
F = 1 if # of 1 are odd

$F = A \oplus B \oplus C$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(8) XNOR gate

Not XOR (even ~~f~~ function)



$F = \overline{(A \oplus B)}$

$= AB + \bar{A}\bar{B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1