CSE 260 : **Digital Logic Design**

Number Systems and Codes

Binary Coded Decimal (BCD)

- Decimal numbers are more natural to humans. Binary numbers are natural to computers. Quite expensive to convert between the two.
- If little calculation is involved, we can use some *coding schemes* for decimal numbers.
- One such scheme is BCD, also known as the *8421* code.
- Represent each decimal digit as a 4-bit binary code.

Binary Coded Decimal (BCD)

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

- Also known as the *8421* code.
- Represent each decimal digit as a 4-bit binary code.
- Some codes are unused, eg: $(1010)_{BCD}$, $(1011)_{BCD}$, …, $(1111)_{BCD}$. These codes are considered as errors.
- Easy to convert, but arithmetic operations are more complicated.
- Suitable for interfaces such as keypad inputs and digital readouts.

Binary Coded Decimal (BCD)

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

- Examples:

  $(234)_{10} = (0010\ 0011\ 0100)_{BCD}$
  $(7093)_{10} = (0111\ 0000\ 1001\ 0011)_{BCD}$
  $(1000\ 0110)_{BCD} = (86)_{10}$
  $(1001\ 0100\ 0111\ 0010)_{BCD} = (9472)_{10}$

Notes: BCD is not equivalent to binary.

Example: $(234)_{10} = (11101010)_2$

Binary Codes

- Other Codes

Decimal Digit	BCD 8421	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100
	1010	0000
Unused	1011	0001
bit	1100	0010
combi-	1101	1101
nations	1110	1110
	1111	1111

Negative Numbers Representation

- There are three common ways of representing signed numbers (positive and negative numbers) for binary numbers:
  - Sign-and-Magnitude
  - 1s Complement
  - 2s Complement

Sign-and-Magnitude

- Negative numbers are usually written by writing a minus sign in front.
  - Example:

    $-(12)_{10}$, $-(1100)_2$

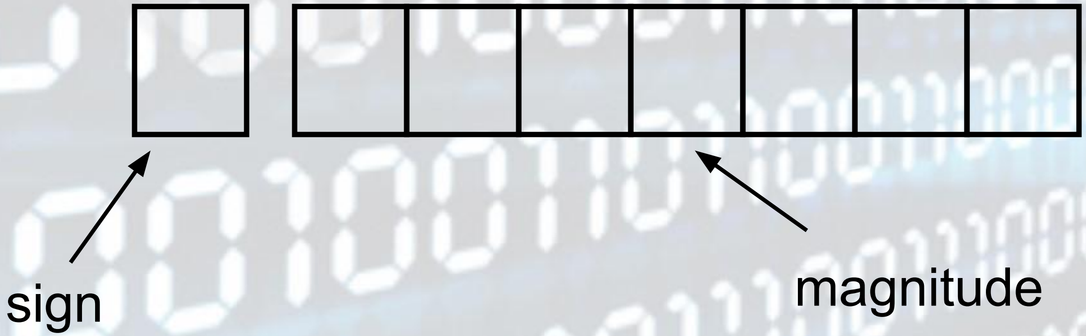
- In computer memory of fixed width, this sign is usually represented by a bit:

  0 for +

  1 for −

Sign-and-Magnitude

- Example: an 8-bit number can have 1-bit sign and 7-bits magnitude.



Signed magnitude representation

- Examples:

$1101_2 = 13_{10}$ (a 4-bit unsigned number)

0 $1101 = +13_{10}$ (a positive number in 5-bit signed magnitude)

1 $1101 = -13_{10}$ (a negative number in 5-bit signed magnitude)

$0100_2 = 4_{10}$ (a 4-bit unsigned number)

0 $0100 = +4_{10}$ (a positive number in 5-bit signed magnitude)

1 $0100 = -4_{10}$ (a negative number in 5-bit signed magnitude)

Sign-and-Magnitude

- Largest Positive Number: 0 1111111 $+(127)_{10}$
- Largest Negative Number: 1 1111111 $-(127)_{10}$
- Zeroes: 0 0000000 $+(0)_{10}$
  1 0000000 $-(0)_{10}$
- Range: $-(127)_{10}$ to $+(127)_{10}$
- Signed numbers needed for negative numbers.
- Representation: Sign-and-magnitude.

1s Complement

- Given a number *x* which can be expressed as an *n*-bit binary number (*i.e. integer part has n digits and fraction has m digit*), its negative value can be obtained in 1s-complement representation using:

+7	0111	-7	1000
+6	0110	-6	1001
+5	0101	-5	1010
+4	0100	-4	1011
+3	0011	-3	1100
+2	0010	-2	1101
+1	0001	-1	1110
+0	0000	-0	1111

1's complement

$$-(00001100)_2$$

inverse

$$(11110011)_2$$

1s Complement

- Essential technique: invert all the bits.

  Examples: 1s complement of 00000001 = $(11111110)_{1s}$

  1s complement of 01111111 = $(10000000)_{1s}$
- Range [in 8 bits]: $-(127)_{10}$ to $+(127)_{10}$
- General Formula=$-(2^{n-1} -1)$ to $+ (2^{n-1} -1)$
- The most significant bit still represents the sign:

  0 = +ve; 1 = -ve.

Note: Range for n bit no. is –(2^(n-1)-1) to (2^(n-1)-1)

1s Complement

- Examples (assuming 8-bit binary numbers):

$(14)_{10} = (00001110)_2 = (00001110)_{1s}$

$-(14)_{10} = -(00001110)_2 = (11110001)_{1s}$

$-(80)_{10} = -(\ ?\ )_2 = (\ ?\ )_{1s}$

2s Complement

- Given a number *x* which can be expressed as an *n*-bit (*i.e. integer part has n digits and fraction has m digit*) number , its negative number can be obtained in 2s-complement representation using:

  $$-x = 2^n - x$$

  Example: With an 8-bit number 00001100, its negative value in 2s complement is thus:

  $$\begin{aligned} -(00001100)_2 &= -(12)_{10} \\ &= (2^8 - 12)_{10} \\ &= (244)_{10} \\ &= \mathbf{(11110100)_{2s}} \end{aligned}$$

2s Complement

- **Method 1:** Essential technique: invert all the bits and add 1.

  Official method!

  Examples:

  2s complement of

  $(00000001)_{2s} = (11111110)_{1s}$ (invert i.e 1's complement)

  $= (11111111)_{2s}$ (add 1)

  2s complement of

  $(01111110)_{2s} = (10000001)_{1s}$ (invert i.e 1's complement)

  $= (10000010)_{2s}$ (add 1)

- **Method 2:** Keep unchanged till 1st occurrence of 1 from LSB and invert remaining 1's into 0's and 0's into 1's till MSB

  $(01111110)_{2s} = (10000010)_{2s}$

  Unchanged

  Inverted

  Unofficial method!

2s Complement

- Range in 8 bits: $-(128)_{10}$ to $+(127)_{10}$

- General Formula= $-(2^{n-1})$ to $+ (2^{n-1}-1)$
- The most significant bit still represents the sign:

  0 = +ve; 1 = -ve.

Note: Range for n bit no. is -2^(n-1) to 2^(n-1)-1

2s Complement

- Examples (assuming 8-bit binary numbers):

$(14)_{10} = (00001110)_2 = (00001110)_{2s}$

$-(14)_{10} = -(00001110)_2 = (11110010)_{2s}$

$-(80)_{10} = -(\ ?\ )_2 = (\ ?\ )_{2s}$

Comparisons of Sign-and-Magnitude and Complements

- Example: 4-bit signed number (*positive values*)

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
+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

Note: Signed magnitude cannot be used for arithmetic calculations

Comparisons of Sign-and-Magnitude and Complements

- Example: 4-bit signed number (*negative values*)

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
-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	-	-	1000

- Note: Signed magnitude cannot be used for arithmetic calculations

Exercise:

1. For 2's complement binary numbers, the range of values for 5-bit numbers is

   a. 0 to 31   b. -8 to +7   c. -8 to +8

   d. -15 to +15   e. -16 to +15

2. In a 6-bit 2's complement binary number system, what is the decimal value represented by $(100100)_{2s}$?

   a. -4   b. 36   c. -36   d. -27   e. -28

Solution:

1) Following numbers are in 1's complement system. Turn them to their no. negative representation

A. 1010101

B. 0111000

C. 0000001

D. 00000

2) Now perform 2's complement

UNSIGNED NO. ARTHEMATIC OPERATION!

Binary Arithmetic Operations for Unsigned numbers

- ADDITION

- Like decimal numbers, two numbers can be added by adding each pair of digits together with carry propagation.

$$\begin{array}{r} (11011)_2 \\ +\ (10011)_2 \\ \hline (101110)_2 \\ \hline \end{array} \qquad \begin{array}{r} (647)_{10} \\ +\ (537)_{10} \\ \hline (1184)_{10} \\ \hline \end{array}$$

Binary Arithmetic Operations for Unsigned Numbers

- SUBTRACTION
- Two numbers can be subtracted by subtracting each pair of digits together with borrowing, where needed.

$$\begin{array}{r} (11001)_2 \\ -\ (10011)_2 \\ \hline (00110)_2 \\ \hline \end{array} \qquad \begin{array}{r} (627)_{10} \\ -\ (537)_{10} \\ \hline (090)_{10} \\ \hline \end{array}$$

Overflow!

Conditions of Overflow Flag :

1. Addition of numbers with same sign,

→ $(+A) + (+B) =$ '+' then $OF = 0$

'−' " $OF = 1$

→ $(-A) + (-B) =$ '−' then $OF = 0$

$(-A) + (-B) =$ '+' " $OF = 1$

* ২টি same sign -এর number add করলে ওই sign না থেকে অন্যটা থাকলে OF হয় that means $OF = 1$ otherwise $OF = 0$.

2. <u>Subtraction</u> of numbers with <u>same sign</u>

২ টি same sign এর number subtract (বিয়োগ) করলে কখনো overflow হয় না that means OF = 0 [(+A) − (+B) = no overflow; (−A) − (−B) = no overflow

3. <u>Addition</u> of numbers with <u>different sign</u>

২ টি different sign (+ or −) number যোগ করলে OF = 0 হবে always মানে overflow হবে না।

(−A) + (+B) = no overflow

(+A) + (−B) = no overflow.

4. <u>Subtraction</u> of numbers with <u>different sign</u>

⇒ $(+A) - (-B) = A + B$ — এটাকে simplify করার পর এক ধরনের form — এ চলে আসে, তাহলে $A+B$ — এর result যদি '+' — আসে — তাহলে $OF = 0$ — আর যদি result '−' — আসে then $OF = 1$.

⇒ $(-A) - (+B) = -A - B = (-A) + (-B)$ — এটাও simplify এর পর এক ধরনের form — এ চলে আসে, — তাহলে ২টো same sign — এর number add করলে যদি — সেই sign — আসে then $OF = 0$ — আর যদি opposite sign আসে তাহলে $OF = 1$.

ARITHMETIC OPERATIONS ON SIGNED NUMBER

2s Complement Addition/Subtraction

- Algorithm for addition, A + B:

1. Perform binary addition on the two numbers.
2. Ignore the carry out of the MSB (most significant bit).
3. Check for overflow: Overflow occurs if the 'carry in' and 'carry out' of the MSB are different, or if result is opposite sign of A and B.

- Algorithm for subtraction, A – B:

  A – B = A + (–B)

1. Take 2s complement of B by inverting all the bits and adding 1.
2. Add the 2s complement of B to A.

2s Complement Addition/Subtraction

- Examples: 4-bit binary system

```
  +3          0011
+ +4        + 0100
----        --------
  +7          0111
----        --------
```

```
  -1          1111
+ -5        + 1011
----        --------
  -6         11010
----        --------
```

```
  +6          0110
+ -3        + 1101
----        --------
  +3         10011
----        --------
```

```
  +4          0100
+ -7        + 1001
----        --------
  -3          1101
----        --------
```

- Which of the above is/are overflow(s)?

2s Complement Addition/Subtraction

- More examples: 4-bit binary system

```
  -3         1101
+ -6       + 1010
----       --------
  -9        10111
----       --------
```

```
  +5         0101
+ +6       + 0110
----       --------
 +11         1011
----       --------
```

- Which of the above is/are overflow(s)?

1s Complement Addition/Subtraction

- Algorithm for addition, A + B:

1. Perform binary addition on the two numbers.
2. If there is a carry out of the MSB, add 1 to the result.
3. Check for overflow: Overflow occurs if result is opposite sign of A and B.

- Algorithm for subtraction, A – B:

  A – B = A + (–B)

1. Take 1s complement of B by inverting all the bits.
2. Add the 1s complement of B to A.

1s Complement Addition/Subtraction

- Examples: 4-bit binary system

```
  +3          0011
+ +4        + 0100
----        --------
  +7          0111
----        --------
```

```
  +5          0101
+ -5        + 1010
----        --------
  -0          1111
----        --------
```

```
  -2        1101
+ -5    +   1010
----        -------
  -7       10111
----      +     1
            -------
            1000
```

```
  -3          1100
+ -7        + 1000
----        --------
 -10         10100
----        +    1
            --------
              0101
```

Sample Math:

Add -3 with -6 in 6 bits using 2's complement number system and justify whether there is overflow or not.

Solution:

Unsigned 3 = 11

+3 = 011

+3 in 6 bits = 000011

To get -3 in binary, we need to calculate 2's complement.

So, 111100 (1's complement)

+1 (Adding 1)

-3 in 6 bits = 111101 (2's complement) -------- (1)

Unsigned 6 = 110

+6 = 0110

+6 in 6 bits = 000110

To get -6 in binary, we need to calculate 2's complement.

So, 111001 (1's complement)

+1 (Adding 1)

-6 in 6 bits = 111010 (2's complement) -------- (2)

Now, adding (1) and (2),

111101

111010

1110111

Ignoring the carry, final result is 110111

Here, we are adding two negative numbers and the result is also a negative number. Hence, there is no overflow.