Signed binary numbers & Binary Codes

Lecture 2

Dronacharya Group of Institutions

Signed Integer Representation

- Fractional number representation (10.3456) or floating point representation (i.e. 9.23×10^{13}) is not the prime discussion here.

- **Signed integer representation** will be discussed.

- The ***PROBLEM*** with signed integers (- 45, + 27, -99) is the SIGN! How do we encode the sign?

- The sign is an extra piece of information that has to be encoded in addition to the magnitude.

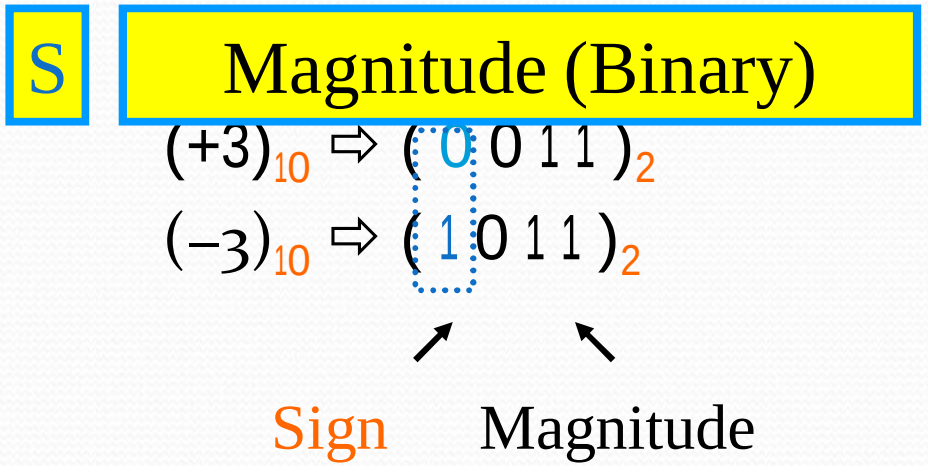
- **what can we do??**

Representation of Negative Numbers

- Signed-Magnitude Representation: Negates a number by changing its sign.
- Complement Number Systems: negates a number by taking its complement.
  - One's-Complement
  - Two's-Complement

Signed Magnitude Representation

- Magnitude is magnitude, *does not change with sign*



- Can't include the *sign bit* in 'Addition'

$$\begin{array}{rl} & 0\ 0\ 1\ 1 \Rightarrow (+3)_{10} \\ + & 1\ 0\ 1\ 1 \Rightarrow (-3)_{10} \\ \hline & 1\ 1\ 1\ 0 \Rightarrow (-6)_{10} \end{array}$$

Signed Magnitude Representation

Signed Magnitude (SM) is a method for encoding signed integers.

The Most Significant Bit is used to represent the sign. ‘1’ is used for a ‘-’ (negative sign), a ‘0’ for a ‘+’ (positive sign).

The format of a SM number in 8 bits is:

Smmmmmmm

where ‘s’ is the sign bit and the other 7 bits represent the magnitude.

NOTE: for positive numbers, the result is the same as the unsigned binary representation.

Signed Magnitude Examples (8 bits)

$-5 = 1\ 0000101_2 = 85_{16}$

$+5 = 0\ 0000101_2 = 05_{16}$

$+127 = 0\ 1111111_2 = 7F_{16}$

$-127 = 1\ 1111111_2 = FF_{16}$

$+0 = 0\ 0000000_2 = 00_{16}$

$-0 = 1\ 0000000_2 = 80_{16}$

For 8 bits, can represent the signed integers -127 to +127.

For N bits, can represent the signed integers

$$-(2^{(N-1)} - 1) \quad \text{to} \quad +(2^{(N-1)} - 1)$$

Signed Magnitude comments

Signed magnitude easy to understand and encode. Is used today in some applications.

One problem is that it has two ways of representing 0 (-0, and +0) . Mathematically speaking, no such thing as two representations for zeros.

Another problem is that addition of K + (-K) does not give Zero!

$$-5 + 5 = 85_{16} + 05_{16} = 8A_{16} = -10 \text{ !!!}$$

Have to consider the sign when doing arithmetic for signed magnitude representation.

Complement Number Systems

- Two numbers in a complement number system can be added/subtracted directly without the sign and magnitude checks.
- Fixed number of digits, n
  - $D=d_{n-1}d_{n-2}...d_1d_0$
- One's Complement
- Two's Complement

One's Complement Representation

To encode a negative number, get the binary representation of its magnitude, then COMPLEMENT each bit. Complementing each bit mean that 1s are replaced with 0s, 0s are replaced with 1s.

What is -5 in Ones Complement, 8 bits?

The magnitude 5 in 8-bits is $00000101_2 = 05_{16}$

Now complement each bit: $11111010_2 = FA_{16}$

FA_{16} is the 8-bit, ones complement number of -5.

NOTE: positive numbers in 1s complement are simply their binary representation.

One's Complement Examples

-5	=	11111010_2	=	FA_{16}
+5	=	00000101_2	=	05_{16}
+127	=	01111111_2	=	$7F_{16}$
-127	=	10000000_2	=	80_{16}
+ 0	=	00000000_2	=	00_{16}
- 0	=	11111111_2	=	FF_{16}

For 8 bits, can represent the signed integers -127 to +127.

For N bits, can represent the signed integers

$$-(2^{(N-1)} - 1) \quad \text{to} \quad +(2^{(N-1)} - 1)$$

One's Complement Comments

Still have the problem that there are two ways of representing 0 (-0, and +0) . Mathematically speaking, no such thing as two representations for zeros.

However, addition of K + (-K) now gives Zero!

$$-5 + 5 = FA_{16} + 05_{16} = FF_{16} = -0 \text{ !!!}$$

Unfortunately, K + 0 = K only works if we use +0, does not work if we use -0.

$$5 + (+0) = 05_{16} + 00_{16} = 05_{16} = 5 \text{ (ok)}$$

$$5 + (-0) = 05_{16} + FF_{16} = 04_{16} = 4 \text{ !!! (wrong)}$$

Two's Complement Representation

To encode a negative number, get the binary representation of its magnitude, COMPLEMENT each bit, then ADD 1. (get Ones complement, then add 1).

What is -5 in Twos Complement, 8 bits?

The magnitude 5 in 8-bits is $00000101_2 = 05_{16}$

Now complement each bit: $11111010_2 = FA_{16}$
Now add one: $FA_{16} + 1 = FB_{16}$
FB_{16} is the 8-bit, twos complement representation of -5.

NOTE: positive numbers in 2s complement are simply their binary representation.

Two's Complement Examples

-5	=	11111011_2	=	FB_{16}
+5	=	00000101_2	=	05_{16}
+127	=	01111111_2	=	$7F_{16}$
-127	=	10000001_2	=	81_{16}
-128	=	10000000_2	=	80_{16} (note the extended range!)
+ 0	=	00000000_2	=	00_{16}
- 0	=	00000000_2	=	00_{16} (only 1 zero!!!)

For 8 bits, can represent the signed integers -128 to +127.

For N bits, can represent the signed integers

$$-2^{(N-1)} \quad \text{to} \quad +(2^{(N-1)} - 1)$$

Note that negative range extends one more than positive range.

Two's Complement Comments

Two's complement is the method of choice for representing signed integers.

It has none of the drawbacks of Signed Magnitude or Ones Complement.

There is only one zero, and K + (-K) = 0.

$$-5 + 5 = FB_{16} + 05_{16} = 00_{16} = 0 \text{ !!!}$$

Normal binary addition is used for adding numbers that represent twos complement integers.

Sign Extended 2's Complement

•What happens if we need to represent the number in with more bits? Lets say 10 bits.

What is -5 in Twos Complement, 8 bits?
The magnitude 5 in 10-bits is $0000000101_2 = 005_{16}$

Now complement each bit: $1111111010_2 = 3FA_{16}$
Now add one: $3FA_{16} + 1 = 3FB_{16}$
$3FB_{16}$ is the 10-bit, twos complement representation of -5.

NOTE: The 8 bit representation of -5 is 11111011_2

The 10 bit representation of -5 is 1111111011_2

Sign Extended 2's Complement

- 12-bit representation of the same numbers

$$
\begin{aligned}
-5 &= 111111111011_2 = FFB_{16} \\
+5 &= 000000000101_2 = 005_{16} \\
+127 &= 000001111111_2 = 07F_{16} \\
-127 &= 111110000001_2 = F81_{16} \\
-128 &= 111110000000_2 = F80_{16} \\
+0 &= 000000000000_2 = 000_{16} \\
-0 &= 000000000000_2 = 000_{16}
\end{aligned}
$$

Compare 12 bits vs. 8 bits

	12 bits	8 bits
-5	111111111011_2	11111011_2
+5	000000000101_2	00000101_2
+127	000001111111_2	01111111_2
-127	111110000001_2	10000001_2
-128	111110000000_2	10000000_2
+ 0	000000000000_2	00000000_2
- 0	000000000000_2	00000000_2

Binary Codes

- Group of *n* bits
  - Up to 2^n combinations
  - Each *combination* represents *an element* of information
- Binary Coded Decimal (BCD)
  - Each Decimal Digit is represented by 4 bits
  - (0 – 9) ⇨ Valid combinations
  - (10 – 15) ⇨ Invalid combinations

Decimal	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

BCD Addition

- One decimal digit + one decimal digit
  - If the result is 1 decimal digit (≤ 9), then it is a simple binary addition

    Example:

$$\begin{array}{r} 5 \\ +\ 3 \\ \hline 8 \end{array} \quad \Leftrightarrow \quad \begin{array}{r} 0\,1\,0\,1 \\ +\ 0\,0\,1\,1 \\ \hline 1\,0\,0\,0 \end{array}$$

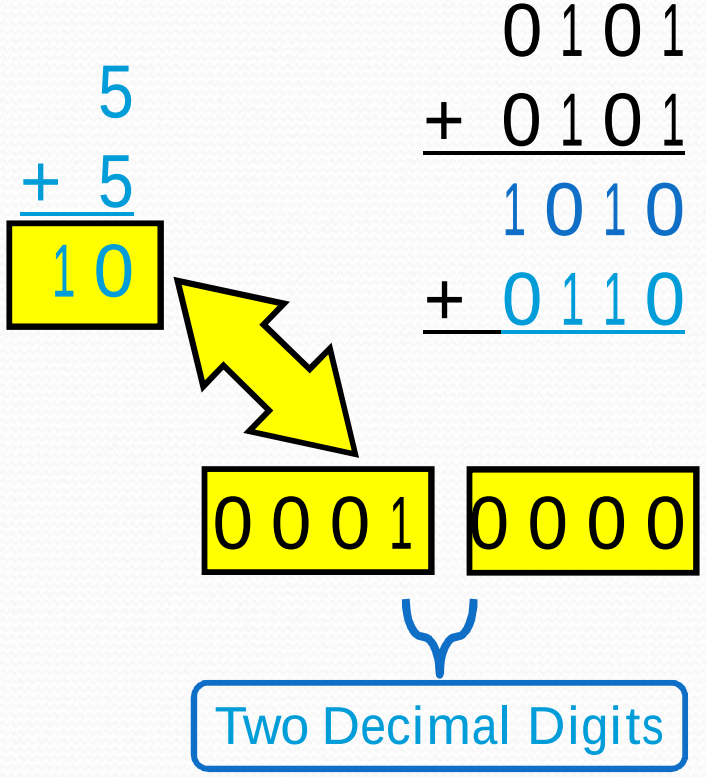
  - If the result is two decimal digits (≥ 10), then binary addition gives invalid combinations

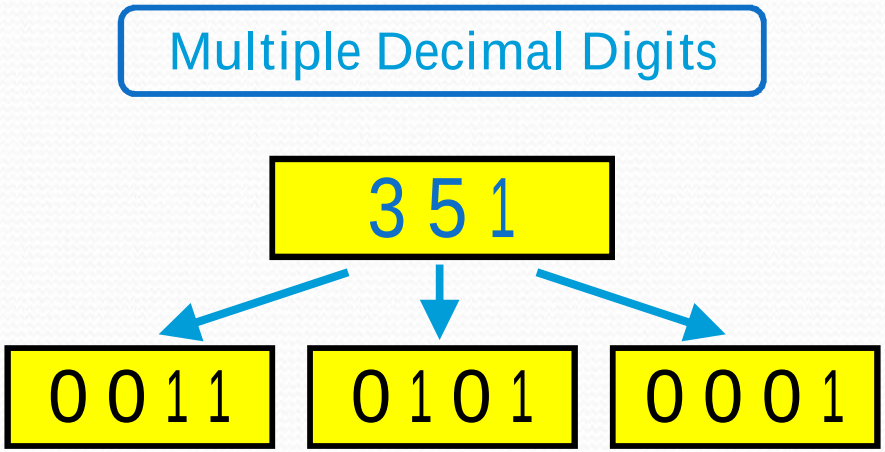
    Example:

$$\begin{array}{r} 5 \\ +\ 5 \\ \hline 10 \end{array} \quad \Leftrightarrow \quad \begin{array}{r} 0\,1\,0\,1 \\ +\ 0\,1\,0\,1 \\ \hline 1\,0\,1\,0 \end{array} \quad \Leftrightarrow \quad 0\,0\,0\,1 \;\; 0\,0\,0\,0$$

BCD Addition

- If the binary result is greater than 9, correct the result by adding 6





Gray Code

- One bit changes from one code to the next code
- Different than Binary

Decimal	Gray	Binary
00	0000	0000
01	0001	0001
02	0011	0010
03	0010	0011
04	0110	0100
05	0111	0101
06	0101	0110
07	0100	0111
08	1100	1000
09	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

ASCII Code

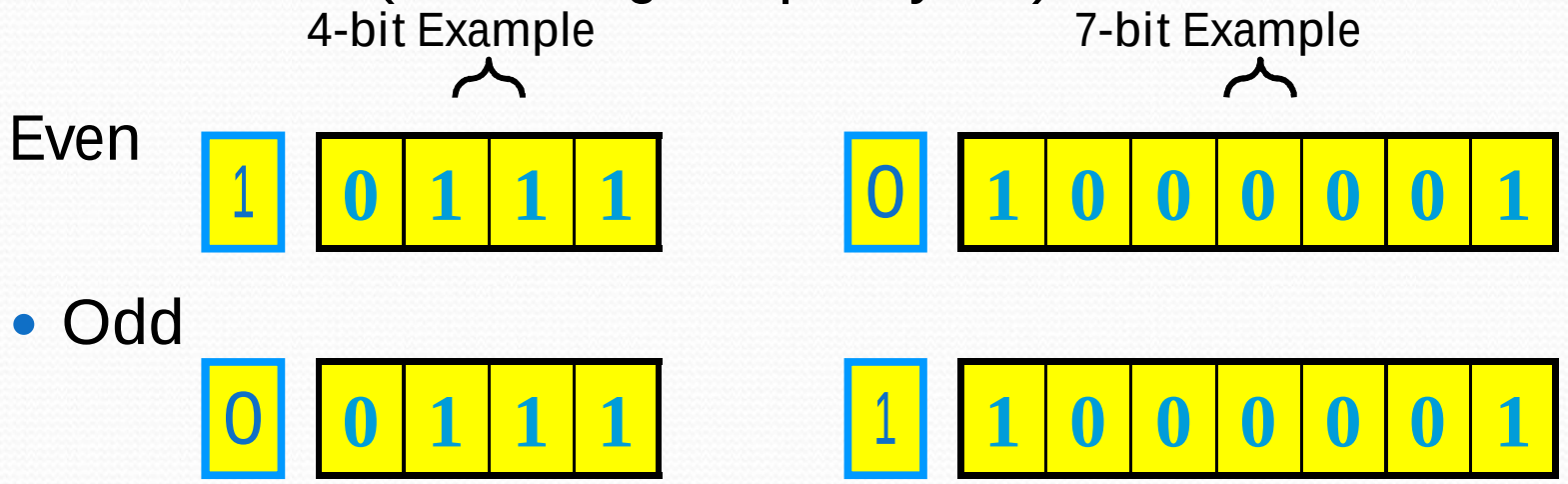
American Standard Code for Information Interchange

Info	7-bit Code
A	1000001
B	1000010
• • •	• • •
Z	1011010
a	1100001
b	1100010
• • •	• • •
z	1111010
@	1000000
?	0111111
+	0101011

Error Detecting Codes

- Parity

  One bit added to a group of bits to make the total number of '1's (including the parity bit) *even* or *odd*



- Good for checking single-bit errors

Summary

- Signed-magnitude, two's complement, one's complement
- Different for negatives numbers
- Representations of positive numbers are SAME.
- 0 may have different representations.
- Sign bit: 0 for positive, 1 for negative
- To use more bits, extend the sign