Digital Logic Design

Lecture :04

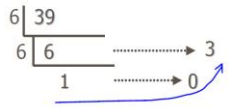
Number Examples with Different Bases

- Decimal (base-10)
  - $(982)_{10}$
- Binary (base-2)
  - $(01111010110)_2$
- Octal (base-8)
  - $(1726)_8$
- Hexadecimal (base-16)
  - $(3d6)_{16}$
- Others examples:
  - base-9 = $(1321)_9$
  - base-11 = $(813)_{11}$
  - base-17 = $(36d)_{17}$

Convert between different bases

- Convert a number base-x to base-y, e.g. $(0100111)_2$ to $(?)_6$
  - First, convert from base-x to base-10 if $x \neq 10$
  - Then convert from base-10 to base-y

$0100111 = 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 39$



$\therefore (0100111)_2 = (103)_6$

Information Representation (cont.)

- Various Codes used in Computer Industry
  - Number-only representation
    - BCD (4 bits per decimal number)
  - Alpha-numeric representation
    - ASCII (7 bits)
  - Unicode (16 bit)

Negative Number Representation

- Options
  - Sign-magnitude
  - One's Complement
  - Two's Complement (we use this in this course)

Sign-magnitude

- Use the most significant bit (MSB) to indicate the sign
  - 0: positive, 1: negative
- Problem
  - Representing zeros?
  - Do not work in computation

+0	000
+1	001
+2	010
+3	011
-3	111
-2	110
-1	101
-0	100

One's Complement

- Complement (flip) each bit in a binary number
- Problem
  - Representing zeros?
  - Do not always work in computation
    - Ex: 111 + 001 = 000 → Incorrect !

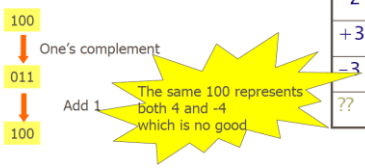
+0	000
+1	001
+2	010
+3	011
-3	100
-2	101
-1	110
0	111

Two's Complement

- Complement (flip) each bit in a binary number and add 1, with overflow ignored
- Work in computation perfectly

Two's Complement

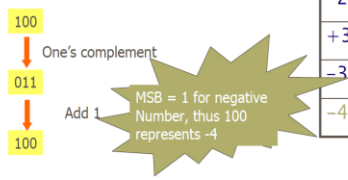
- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !



0	000
+1	001
-1	111
+2	010
-2	110
+3	011
-3	101
??	100

Two's Complement

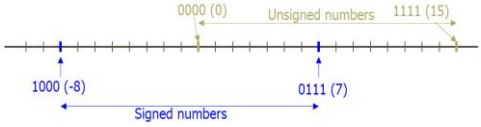
- Complement (flip) each bit in a binary number and adding 1, with overflow ignored
- Work in computation perfectly
- We will use it in this course !



0	000
+1	001
-1	111
+2	010
-2	110
+3	011
-3	101
-4	100

Range of Numbers

- An N-bit number
  - Unsigned: 0 .. $(2^N - 1)$
  - Signed: -2^{N-1} .. $(2^{N-1} - 1)$
- Example: 4-bit



Binary Arithmetic Addition

	Sum	Carry
0 + 0	= 0	0
0 + 1	= 1	0
1 + 1	= 0	1
1+ 0	= 1	1

Binary Arithmetic Subtraction

Consider the Computation of 7- 5
It will be as 7 + (-5) = 2 (using 2's Complement)
5 has Binary = 0101
7 has binary = 0111

```
  0 1 1 1 (7)
- 1 0 1 1 (-5)
(1)0 0 1 0      (Ignoring OverFlow )
```

Binary Computation

```
010001 (17=16+1)
001011 (11=8+2+1)
---------------
011100 (28=16+8+4)
```

```
Unsigned arithmetic
010001 (17=16+1)
101011 (43=32+8+2+1)
---------------
111100 (60=32+16+8+4)
```

```
Signed arithmetic (w/ 2's complement)
010001 (17=16+1)
101011 (-21: 2's complement=010101=21)
---------------
111100 (2's complement=000100=4, i.e. -4)
```

Binary Computation

The carry is discarded

```
Unsigned arithmetic
101111 (47)
011111 (31)
---------------
001110 (78??  Due to overflow, note that
              78 cannot be represented
              by a 6-bit unsigned number)
```

The carry is discarded

```
Signed arithmetic (w/ 2's complement)
101111 (-17  since 2's complement=010001)
011111 (31)
---------------
001110 (14)
```