Digital Logic Design

Hajar Falahati

Department of Computer Engineering
IRAN University of Science and Technology

hfalahati@iust.ac.ir

Number System

- Positioned number

$$N = (a_{n-1}a_{n-2} \ldots a_1a_0 \,.\, a_{-1}a_{-2} \ldots a_{-m})_r$$

  - . = radix point
  - r = radix or base
  - n = number of integer digits to the left of the radix point
  - m = number of fractional digits to the right of the radix point
  - a_{n-1} = most significant digit (MSD)
  - a_{-m} = least significant digit (LSD)

- Polynomial notation
  - Series representation

$$N = a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots + a_0 \times r^0 + a_{-1} \times r^{-1} \ldots + a_{-m} \times r^{-m}$$

$$\sum_{i=-m}^{n-1} a_i r^i$$

Binary Arithmetic

- Consider two binary numbers (A, B) and a carry bit C_i
- How can we add these two binary numbers with a carry bit?

$$\begin{array}{rr} & C_i \\ & A \\ + & B \\ \hline & C\ S \end{array}$$

$$\begin{array}{rr} & A \\ - & B \\ \hline & O \end{array}$$

$$\begin{array}{rr} & A \\ \times & B \\ \hline & O \end{array}$$

$$\begin{array}{rr} & A \\ \div & B \\ \hline & O \end{array}$$

Outline

- Signed and Unsigned Numbers

Signed & Unsigned Numbers

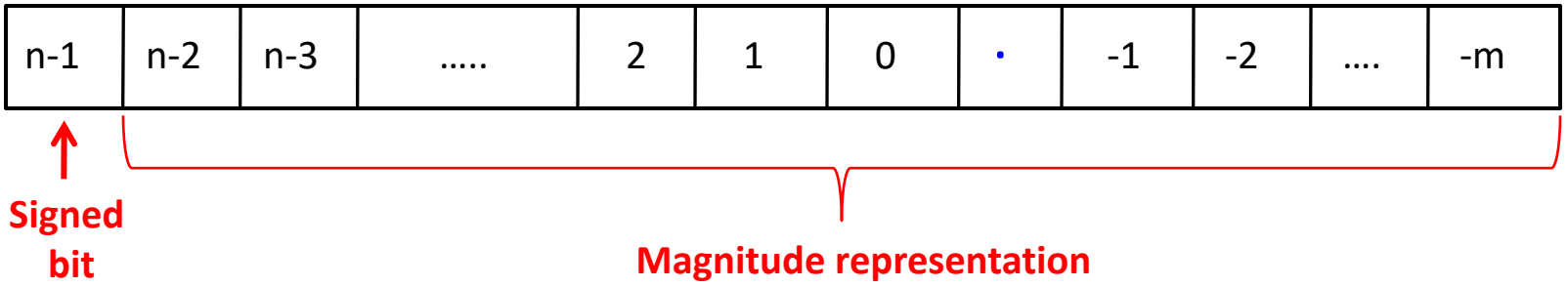
Signed Numbers

- Negative numbers representation
  - -4, -3.02,
- Signed numbers
  - Sign is **important**
  - Both negative and positive numbers
- Unsigned numbers
  - Sign is not important
  - Only positive numbers

Signed Numbers Representation

- Three ways
  - Sign-magnitude
  - 2’s complement
  - 1’s complement

Signed Magnitude Method



- $N = \pm (a_{n-1} \ldots a_1 a_0 . a_{-1} \ldots a_{-m})_r$
- $N = (s\ a_{n-1} \ldots a_1 a_0 . a_{-1} \ldots a_{-m})_{rsm}$
  - $S = r - 1$
  - $r = 2 \rightarrow s = 1$
  - $r = 10 \rightarrow s = 9$

Signed Magnitude Method (cont'd)

- $N = -(13)_{10}$
- $N = +(18)_{10}$
- $N = -(10)_{10}$

Signed Magnitude Method (cont'd)

- $N= -(13)_{10}$
  - Binary: $N= -(1101) = (1,1101)_{2sm} = (11101)_{2sm}$
  - Decimal: $N= -(13)_{10} = (9,13)_{10sm} = (913)_{10sm}$

- $N= +(18)_{10}$
  - Binary: $N= +(10010) = +(010010)_{2sm}$
  - Decimal: $N= +(18)_{10} = (018)_{10sm}$

- $N= -(10)_{10}$
  - Binary: $N= -(1010) = (1,1010)_{2sm} = (11010)_{2sm}$
  - Decimal: $N= -(10)_{10} = (9,10)_{10sm} = (910)_{10sm}$

Signed Magnitude Method (cont'd)

- Advantages
  - **Simple** and **easy** for humans understand

- Disadvantages
  - **Complicates** computer arithmetic units
    - Two representations of zero, +0 and -0
    - Need both an adder and a subtractor

2's Complement

- Let N = $(a_{n-1} \ldots a_0)_2$
  - If $N \geq 0$, it is represented by $(0a_{n-1} \ldots a_0)_2$
  - If $N < 0$, it is represented by $[0a_{n-1} \ldots a_0]_2$

- 2's complement $[N]_2 = 2^n - (N)_2$
  - n: the number of digits in $(N)_2$
  - $(01100101)_2$
  - $[N]_2 = [01100101]_2 = 2^8 - (01100101)_2 = (100000000)_2 - (01100101)_2 = (10011011)_2$,

2's Complement (cont'd)

- Consider 8-bit numbers
- 2's complement of $(11010100)_2$
- $(N)_2 + [N]_2 = 0$?

2's Complement (cont'd)

- $(11010100)_2$
  - $[N]_2 = [11010100]_2$
  - $[N]_2 = 2^8 - (11010100)_2$
  - $[N]_2 = (100000000)_2 - (11010100)_2$
  - $[N]_2 = (00101100)_{2'}$
- $(N)_2 + [N]_2 = 0$?
  - $[N]_2 + (N)_2 = 00101100 + 11010100$
  - $[N]_2 + (N)_2 = 00101100 + 11010100 = 100000000$

2's Complement: Algorithm 1

- Find $[N]_2$ given $(N)_2$
  - Copy the bits of N
  - Proceeding from LSD to MSD
  - Find first 1 bit -> Do nothing
  - Change remaining digits: $a_j \rightarrow 1 - a_j$
  - Example:
    - 2's complement for (11010100)2

	MSD							LSD
Digit notation	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
Number	1	1	0	1	0	1	0	0
2's complement	0	0	1	0	1	1	0	0

2's Complement: Algorithm 1 (cont'd)

$(10000101)_2$ ⟺ $(01111011)_{2cns}$

$(00010000)_2$ ⟺ $(11110000)_{2cns}$

2's Complement: Algorithm 2

- Find $[N]_2$ given $(N)_2$.
  - Change each digit: $a_j \rightarrow 1 - a_j$
  - Add by 1
  - Example:
    - 2's complement for $(11010100)_2$

	MSD							LSB
Digit notation	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
Number	1	1	0	1	0	1	0	0
1st step	0	0	1	0	1	0	1	1
2nd step: +1	0	0	0	0	0	0	0	1
2's complement	0	0	1	0	1	1	0	0

2's Complement: Algorithm 2: Sample 1

- 8-bit numbers

- $(10110101)_2$ ⟺ $(01001010)_2$ ⟺ $(01001011)_2$
- $(01110000)_2$ ⟺ $(10001111)_2$ ⟺ $(10010000)_2$

2’s Complement: Algorithm 2: Sample 2

- 3-bit numbers

• $(181)_{10}$	⟺	• $1000 - (181)_{10}$	⟺	• $(819)_{10}$
• $(112)_{10}$	⟺	• $100 - (112)_{10}$	⟺	• $(888)_{2}$

2's Complement (cont'd)

- **Two's complement number system**
  - Positive number :
    - $N = +(a_{n-2}, \ldots, a_0)_2 = (0, a_{n-2}, \ldots, a_0)_{2cns}$,

      where $0 \leq N \leq 2^{n-1} - 1$
  - Negative number:
    - $N = (a_{n-1}, a_{n-2}, \ldots, a_0)_2$
    - $-N = [a_{n-1}, a_{n-2}, \ldots, a_0]_2$ (two's complement of N),

      where $-1 \geq N \geq -2^{n-1}$

6 bits → +31 to -32

8 bits → +127 to -128

16 bits → +65535 to -65536

2’s Complement : Sample 3

- Present these numbers and 2’s complement in 6 bits
  - $(1)_{10}$
  - $(5)_{10}$
  - $(10)_{10}$
  - $(20)_{10}$
  - $(30)_{10}$
  - $(31)_{10}$
  - $(32)_{10}$

2's Complement : Sample 3

- Present these numbers and 2's complement in 6 bits
  - $(1)_{10}$
  - $(5)_{10}$
  - $(10)_{10}$
  - $(20)_{10}$
  - $(30)_{10}$
  - $(31)_{10}$
  - $(32)_{10}$
  - $(000001)_2$
  - $(000101)_2$
  - $(001010)_2$
  - $(010100)_2$
  - $(011110)_2$
  - $(011111)_2$
  - $(100000)_2$
  - $(0100000)_2$
  - $-(1)_{10} = (111111)_2$
  - $-(5)_{10} = (111011)_2$
  - $-(10)_{10} = (110110)_2$
  - $-(20)_{10} = (101100)_2$
  - $-(30)_{10} = (100010)_2$
  - $-(31)_{10} = (100001)_2$
  - $-(32)_{10} = (100000)_2$
  - $-(32)_{10} = (1100000)_2$

2's Complement: Sample 4

- $(10110101)_2 = (A)_{10}$
- $(01110000)_2 = (B)_{10}$

- $[10110101]_2 = [C]_{10}$
- $[01110000]_2 = [D]_{10}$

- $2^8 - (A)_{10} = (E)_{10}$
- $2^8 - (B)_{10} = (F)_{10}$

2's Complement: Sample 4

- $(10110101)_2 = (181)_{10}$
- $(01110000)_2 = (112)_{10}$

- $[10110101]_2 = (01001011)_2$
- $[01110000]_2 = (10010000)_2$

- $2^8 - (181)_2 = (75)_{10}$
- $2^8 - (112)_2 = (144)_{10}$

2's Complement: Sample 4

- $(10110101)_2 = (181)_{10}$
- $(01110000)_2 = (112)_{10}$

- $[10110101]_2 = (01001011)_2 = (75)_{10}$
- $[01110000]_2 = (10010000)_2 = (144)_{10}$

- $2^8 - (181)_2 = (75)_{10}$
- $2^8 - (112)_2 = (144)_{10}$

2's Complement: Extension

- Extend to more number
  - Todays computer support 32-bit or 64-bit integers
- To extend precision
  - Sign bit extension

$12_{10} = 001100_2$

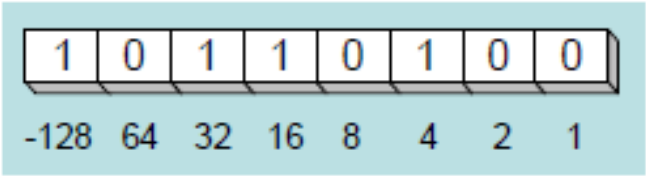
$-12_{10} = 110100_2$

$12_{10} = 000000001100_2$

$-12_{10} = 111111110100_2$

Unsigned and Signed Value

- Positive numbers
  - Signed value = Unsigned value
- Negative numbers
  - Signed value = Unsigned value – 2^n
- Negative weight for MSB



= -128 + 32 + 16 + 4 = -76

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	1
00000010	2	2
....	....	....
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
...	...	...
11111110	254	-2
11111111	255	-1

2’s Complement: Subtraction

- A- B ?

```
  0 1 0 0 1 1 0 1
- 0 0 1 1 1 0 1 0
-----------------
```

```
borrow:   0 110   0 10
  0 1 0 0 1 1 0 1
- 0 0 1 1 1 0 1 0
-----------------
  0 0 0 1 0 0 1 1
```

2’s Complement: Subtraction (cont’d)

- Converts B to its 2's complement
- Add A to (2’s complement of B)
- Ignore final carry

```
borrow:   0 110   1 10
          0 1 0 0 1 1 0 1
        -
          0 0 1 1 1 0 1 0
        ---------------
          0 0 0 1 0 0 1 1
```

```
carry:    1 1     1 1
          0 1 0 0 1 1 0 1
          1 1 0 0 0 1 1 0   (2's complement)
        ---------------
          0 0 0 1 0 0 1 1   Same result
```

2’s Complement: Summary

- 2’s complement of N
- N + 2’s complement of N
  - Zero
  - Final carry is ignored
- Only one zero

2's Complement: Summary (cont'd)

- Range n-bit:
  - **Unsigned** integers: 0 to $(2^n - 1)$
  - **Signed** integers: Range is -2^{n-1} to $(2^{(n-1)} - 1)$
    - **Positive** range: 0 to $(2^{n-1} - 1)$
    - **Negative** range: -2^{n-1} to -1

Storage Size	Unsigned Range	Signed Range
8 bits (byte)	0 to $(2^8 - 1) = 255$	$-2^7 = -128$ to $(2^7 - 1) = +127$
16 bits	0 to $(2^{16} - 1) = 65,535$	$-2^{15} = -32,768$ to $(2^{15} - 1) = +32,767$
32 bits	0 to $(2^{32} - 1) = 4,294,967,295$	$-2^{31} = -2,147,483,648$ to $(2^{31} - 1) = +2,147,483,647$
64 bits	0 to $(2^{64} - 1) =$ 18,446,744,073,709,551,615	$-2^{63} = -9,223,372,036,854,775,808$ to $(2^{63} - 1) = +9,223,372,036,854,775,807$

Thank You