Signed Binary Representations and Arithmetic

Dr. Debapriya Roy

Motivation

- Computers use binary to represent data and perform arithmetic.
- To handle **positive and negative numbers**, we use signed representations.
- Three main systems:
    1. Sign-Magnitude
    2. 1's Complement
    3. 2's Complement
- We will discuss how they represent numbers and perform arithmetic operations.

Sign-Magnitude Representation

- MSB is the **sign bit**:
  - $0 =$ Positive
  - $1 =$ Negative
- Remaining bits represent magnitude (as unsigned binary).
- **Example (4 bits):**
  - $+5 = 0101$, $-5 = 1101$

Issues:

- Two zeros: 0000 (+0), 1000 (-0)
- Arithmetic requires extra logic for comparing and handling sign.

1's Complement Representation

- Positive numbers: normal binary
- Negative numbers: invert (flip) all bits of the positive number
- **Example (4 bits):**
  - $+5 = 0101$
  - $-5 = 1010$ (1's complement of 0101)

Features:

- Two zeros: 0000 (+0), 1111 (-0)
- Subtraction becomes addition by complementing
- **End-around carry** must be added back

2's Complement Representation

- Positive numbers: unchanged
- Negative numbers: flip all bits and add 1
- **Example (4 bits):**
  - $+5 = 0101$
  - -5: flip $\rightarrow$ 1010, add 1 $\rightarrow$ 1011

Advantages:

- Only one zero (0000)
- Simple arithmetic – same circuit for add/sub
- Most widely used method in computers today

Sign-Magnitude Arithmetic

Rules:

- Same signs $\rightarrow$ add magnitudes, keep the sign
- Different signs $\rightarrow$ subtract smaller from larger, use larger's sign

Example: $-5 + 3$

- $-5 = 1101$, $+3 = 0011$
- Magnitude: 5 - 3 $= 2$
- Result: $-2 = 1010$

Note: Requires magnitude comparison and conditional logic

1's Complement Arithmetic

Rules:

- Add both numbers (including sign bits)
- If there is a carry from MSB, add it back (end-around carry)

Example: $-3 + 2$ (4-bit)

- $+2 = 0010$
- $-3 = 1100$ (1's complement of 0011)
- Add: $0010 + 1100 = 1110$ (no carry)
- Result: $1110 = -1$

Note: Still has two zeros. End-around carry must be handled.

1's Complement Arithmetic (Example 2 with End-Around Carry)

Example 2: $-3 + 4$ (4-bit)

- $+4 = 0100$
- $-3 = 1100$ (1's complement of 0011)
- Add: $0100 + 1100 = 10000$ (5 bits: carry out is 1)
- End-around carry: $0000 + 1 = 0001$
- Result: $0001 = +1$

Note: End-around carry must be added back for correct result.

1's Complement Arithmetic (Example 3 with End-Around Carry)

Goal: Add $-2 + (-1)$ using 4-bit 1's complement

- -2:
  - $+2 = 0010 \rightarrow$ 1's complement $=$ $\mathbf{1101}$
- -1:
  - $+1 = 0001 \rightarrow$ 1's complement $=$ $\mathbf{1110}$
- Add: $1101 + 1110 = \mathbf{1\ 1011}$ (5 bits; carry $= 1$)
- End-around carry: $1011 + 1 = \mathbf{1100}$
- Result: $\mathbf{1100} = -3$ (1's complement of 0011)

Note: End-around carry must be added back. MSB $= 1$ indicates a negative result.

1's Complement: Circular Number Line (Part 1)

Why Circular? In 1's complement, the number line wraps around — overflow loops back to the start like a circle.

4-bit 1's Complement Representations:

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

Note: Two representations for zero: `0000` (+0) and `1111` (-0)

1's Complement: Circular Number Line (Part 2)

Circular Behavior in Action:

- 0111 (+7) + 0001 = 1000 (-7) — wraps around after +7
- 1111 (-0) + 0001 = 0000 (+0) — end-around behavior

Why Add Carry Back?

- Carry out from MSB is not overflow — it's a valid part of the result.
- 1's complement uses **modulo** $(2^n - 1)$ arithmetic.
- To stay on the circular number line, the carry must be **added back** (end-around carry).

Conclusion:

1's complement arithmetic only gives correct results when the end-around carry is included.

1's Complement: Circular Number Line Visualization

1's Complement wraps like a circle — values move clockwise through positives and wrap to negatives.

$$
\begin{array}{ccc}
 & +0 \rightarrow +1 \rightarrow +2 \rightarrow \ldots \rightarrow +7 & \\
\uparrow & & \downarrow \\
 & -0 \leftarrow -1 \leftarrow -2 \leftarrow \ldots \leftarrow -7 &
\end{array}
$$

Explanation:

- After $+7$ (0111), next value is 1000 (-7)
- Carry from MSB wraps to LSB (end-around carry)
- Ensures arithmetic continuity on a circular number line

Problem with Two Representations of Zero in 1's Complement

Key Issue: In 1's complement, zero has two representations:

- Positive zero: 0000 0000
- Negative zero: 1111 1111

Example: Adding +4 and -4 (4-bit representation)

$$+4 = 0100$$
$$-4 = 1011 \quad \text{(1's complement of 0100)}$$
$$\text{Sum} = 0100 + 1011 = 1111 \quad (Negativezero)$$

Problems Caused:

- Ambiguity in checking for zero: 0000 vs 1111
- Comparison logic must account for both
- Requires **end-around carry** in addition
- One representation wasted

Disadvantages of 1's Complement Representation

- **Two representations of zero:**
  - Positive zero: 0000 0000
  - Negative zero: 1111 1111
  - Leads to ambiguity and extra handling in comparisons.
- **Extra logic for equality checks:**
  - Both 0000 and 1111 must be treated as zero.
- **Addition requires end-around carry:**
  - If there is a carry from the MSB, it must be added to the LSB.
  - Slows down arithmetic operations.
- **Wasted encoding:**
  - One bit pattern wasted due to dual zero representation.
  - Only $2^n - 1$ distinct integers can be represented.
- **Sign bit interpretation issues:**
  - MSB is 1 for both negative zero and negative numbers.
  - Negative zero looks like a negative number.

2's Complement Arithmetic

Rules:

- Add both numbers directly
- Ignore any final carry out of MSB

Example: $-3 + 2$ (4-bit)

- $+2 = 0010$
- $-3 \rightarrow 3 = 0011 \rightarrow \text{flip} = 1100 \rightarrow \text{add } 1 = 1101$
- Add: $0010 + 1101 = 1111$
- Result: $1111 = -1$

Benefit: No special handling; works like unsigned addition.
Note: In 2's complement, **ignore the carry-out**. No end-around carry is needed, unlike 1's complement.

2's Complement Arithmetic (Carry Ignored)

Example: $-5 + 7$ (4-bit)

- $+7 = 0111$
- -5:
  - $5 = 0101$
  - Flip: 1010
  - Add 1: **1011**
- Add: $0111 + 1011 = \mathbf{1\ 0010}$ (carry out $= 1$)
- Ignore carry-out $\rightarrow$ result $= \mathbf{0010}$
- Final result: $+2$

Note: Carry-out from MSB is discarded. Result is still correct.

Comparison Summary

Property	Sign-Magnitude	1's Comp	2's Comp
Two zeros	Yes	Yes	No
Easy arithmetic	No	No	Yes
Used today	No	Rarely	Yes
End-around carry	No	Yes	No
Same HW for add/sub	No	No	Yes