Tutorial 03: Addition/Subtraction using 2's Complement

Computer Science Department
CS2208: Introduction to Computer Organization and Architecture
Winter 2020-2021
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Binary Arithmetic

❑ These tables cover the fundamental arithmetic operations.

Addition	**Subtraction**	**Multiplication**
0 + 0 = 0 (carry 0)	0 – 0 = 0 (borrow 0)	0 × 0 = 0
0 + 1 = 1 (carry 0)	0 – 1 = 1 (borrow 1)	0 × 1 = 0
1 + 0 = 1 (carry 0)	1 – 0 = 1 (borrow 0)	1 × 0 = 0
1 + 1 = 0 (carry 1)	1 – 1 = 0 (borrow 0)	1 × 1 = 1

Addition (three bits)	**Subtraction (three bits)**
0 + 0 + 0 = 0 (carry 0)	**0 – 0** – 0 = 0 (borrow 0)
0 + 0 + 1 = 1 (carry 0)	**0 – 0** – 1 = 1 (borrow 1)
0 + 1 + 0 = 1 (carry 0)	**0 – 1** – 0 = 1 (borrow 1)
0 + 1 + 1 = 0 (carry 1)	**0 – 1** – 1 = 0 (borrow 1)
1 + 0 + 0 = 1 (carry 0)	**1 – 0** – 0 = 1 (borrow 0)
1 + 0 + 1 = 0 (carry 1)	**1 – 0** – 1 = 0 (borrow 0)
1 + 1 + 0 = 0 (carry 1)	**1 – 1** – 0 = 0 (borrow 0)
1 + 1 + 1 = 1 (carry 1)	**1 – 1** – 1 = 1 (borrow 1)

Sign and Magnitude Addition/Subtraction

- The operations are carried out similar to normal math calculations
- The resultant sign is arranged separately
  - □ The sign of A − B depends on the values of A and B
  - □ If B > A, the answer will be calculated as −(B − A), O.W., it is +(A − B)
- The location of the radix points needs to be aligned before performing the operation.

- If the provided number of bits are not enough to hold the result, it means an overflow occurred.

2's Complement Addition/Subtraction

- A subtraction operation is converted to an addition operation (after performing the *2's complement* to the operand appearing after the negative sign)

- When adding two *positive* numbers and finding the result is *negative*, this means an *overflow occurred*.

- When adding two *negative* numbers and finding the result is *positive*, this means an *overflow occurred*.

- Overflow will *never occur* when adding a positive number to a negative number, or vice versa.

- How about
    - ☐ subtracting a negative number from a positive number?
    - ☐ subtracting a positive number from a negative number?

2's Complement Addition/Subtraction

- ***Example 1***:
Perform $20_{10} - 10_{10}$ using 2's complement 6-bit system

- $20_{10} \rightarrow 10100_2$

- $10_{10} \rightarrow 1010_2$

- $20_{10} - 10_{10} \rightarrow 10100_2 - 1010_2$

  $\rightarrow 010100_2 - 001010_2$

  $\rightarrow 010100_2 + (-001010_2)$

  $\rightarrow 010100_2 + 110110_2$

  $\rightarrow 001010_2$

  $\rightarrow +10_{10}$

Carry out to be ignored

$C_{out} == C_{in}$

```
 11 1
  010100₂
+110110₂
1001010₂
```

Overflow can not occur

This is the answer in 2's complement

This is the answer in decimal to verify

2's Complement Addition/Subtraction

- **_Example 2_**:
  Perform $10_{10} - 20_{10}$ using 2's complement 6-bit system

- 10_{10} ➜ 1010_2

- 20_{10} ➜ 10100_2

- $10_{10} - 20_{10}$ ➜ $1010_2 - 10100_2$

  ➜ $001010_2 - 010100_2$

  ➜ $001010_2 + (-010100_2)$

  ➜ $001010_2 + 101100_2$

  ➜ 110110_2

  ➜ -001010_2

  ➜ -10_{10}

This is the answer in 2's complement

This is the answer in decimal to verify

No carry out

$C_{out} == C_{in}$

Overflow can not occur

$$\begin{array}{r} 1 \\ 001010_2 \\ +101100_2 \\ \hline 110110_2 \end{array}$$

6

2's Complement Addition/Subtraction

- *Example 3*:
  Perform $20_{10} + 10_{10}$ using 2's complement 6-bit system

- $20_{10} \rightarrow 10100_2$

- $10_{10} \rightarrow 1010_2$

- $20_{10} + 10_{10} \rightarrow 10100_2 + 1010_2$
  $\rightarrow 010100_2 + 001010_2$
  $\rightarrow 011110_2$
  $\rightarrow +30_{10}$

This is the answer in 2's complement

This is the answer in decimal to verify

No carry out

$C_{out} == C_{in}$

$$010100_2$$
$$+001010_2$$
$$\overline{011110_2}$$

Overflow might occur, but it did not in this case

2's Complement Addition/Subtraction

- ***Example 4***:

Perform $-20_{10} - 10_{10}$ using 2's complement 6-bit system

- $20_{10} \rightarrow 10100_2$

- $10_{10} \rightarrow 1010_2$

- $-20_{10} - 10_{10} \rightarrow -10100_2 - 1010_2$

    $\rightarrow -010100_2 - 001010_2$

    $\rightarrow (-010100_2) + (-001010_2)$

    $\rightarrow 101100_2 + 110110_2$

    $\rightarrow 1\,00010_2$

    $\rightarrow -011110_2$

    $\rightarrow -30_{10}$

This is the answer in 2's complement

This is the answer in decimal to verify

Carry out to be ignored

$C_{out} == C_{in}$

1111
101100_2
$\underline{+110110_2}$
1100010_2

Overflow might occur, but it did not in this case

8

CS 2208: Introduction to Computer Organization and Architecture

2's Complement Addition/Subtraction

- *Example 5*:

  Perform $20_{10} + 20_{10}$ using 2's complement 6-bit system

- $20_{10} \rightarrow 10100_2$

- $20_{10} + 20_{10} \rightarrow 10100_2 + 10100_2$

  $\rightarrow 010100_2 + 010100_2$

$C_{out} \neq C_{in}$

No carry out

Overflow might occur, **and** *indeed it did in this case*

```
  1 1
  010100₂
+ 010100₂
 101000₂
```

2's Complement Addition/Subtraction

- *Example 6*:
  Perform $-20_{10} - 20_{10}$ using 2's complement 6-bit system

- 20_{10} ➜ 10100_2

- $-20_{10} - 20_{10}$ ➜ $-10100_2 - 10100_2$

  ➜ $-010100_2 - 010100_2$

  ➜ $(-010100_2) + (-010100_2)$

  ➜ $101100_2 + 101100_2$

Carry out to be ignored

$C_{out} \neq C_{in}$

Overflow might occur, and *indeed it did in this case*

```
  1 11
   101100₂
 +101100₂
 ‾‾‾‾‾‾‾‾
 1011000₂
```

10

2's Complement Addition/Subtraction

- **Example 7**:
  Perform $20_{10} - 20_{10}$ using 2's complement 6-bit system
- $20_{10} \rightarrow 10100_2$
- $20_{10} - 20_{10} \rightarrow 10100_2 - 10100_2$

  $\rightarrow 010100_2 - 010100_2$

  $\rightarrow 010100_2 + (-010100_2)$

  $\rightarrow 010100_2 + 101100_2$

  $\rightarrow 000000_2$

  $\rightarrow 0_{10}$

This is the answer in 2's complement

This is the answer in decimal to verify

Carry out to be ignored

$C_{out} == C_{in}$

Overflow can not occur

```
  1111
   010100_2
 +101100_2
 1000000_2
```

2's Complement Addition/Subtraction

- *Example 8*:

Perform $31_{10} + 1_{10}$ using 2's complement 6-bit system

- $31_{10} \rightarrow 11111_2$

- $1_{10} \rightarrow 1_2$

- $31_{10} + 1_{10} \rightarrow 11111_2 + 1_2$

$\rightarrow 011111_2 + 000001_2$

No carry out

$C_{out} \neq C_{in}$

```
  11111
 011111₂
+000001₂
 100000₂
```

Overflow might occur, **and** *indeed it did in this case*

2's Complement Addition/Subtraction

- ***Example 9***:

Perform $-31_{10} - 1_{10}$ using 2's complement 6-bit system

- $31_{10} \rightarrow 11111_2$

- $1_{10} \rightarrow 1_2$

Carry out to be ignored

- $-31_{10} - 1_{10} \rightarrow -11111_2 - 1_2$

  $\rightarrow (-011111_2) + (-000001_2)$

  $\rightarrow (100001_2) + (111111_2)$

  This is the answer in 2's complement

  $\rightarrow 100000_2$

  $\rightarrow -100000_2$

  This is the answer in decimal to verify

  $\rightarrow -32_{10}$

$C_{out} == C_{in}$

$$\begin{array}{r} 111111 \\ 100001_2 \\ +111111_2 \\ \hline 1100000_2 \end{array}$$

Overflow might occur, but it did not in this case

CS 2208: Introduction to Computer Organization and Architecture

2's Complement Addition/Subtraction

- **Example 10**:

Encode -3.25_{10} using 2's complement 6-bit system

- $3.25_{10} \rightarrow 11.01_2$
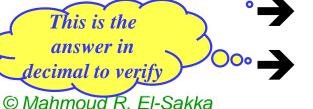- $-3.25_{10} \rightarrow -0011.01_2$
  - $\rightarrow 1100.11_2$

> **Carry out to be ignored**

You can also look at it as if it is $-3_{10} - 0.25_{10}$

> $C_{out} == C_{in}$

- $-3_{10} - 0.25_{10} \rightarrow -11_2 - 0.01_2$
  - $\rightarrow (-000011_2) + (-0000.01_2)$
  - $\rightarrow (111101_2) + (1111.11_2)$
  - $\rightarrow 111100.11_2$
  - $\rightarrow 1100.11_2$
  - $\rightarrow -3.25_{10}$

> *This is the answer in 2's complement*

> *This is the answer in decimal to verify*

> **Overflow might occur, but it did not in this case**

> **Binary points MUST be aligned**

$$\begin{array}{r} 111111 \\ 111101.00_2 \\ +111111.11_2 \\ \hline 1111100.11_2 \end{array}$$