Intro to Software Engineering
EK327

Giacomo Cappelletto

21/1/25

Contents

Chapter 1

Computer Architecture

1.1 Adding Two Single-Bit Binaries Using XOR and AND Gates

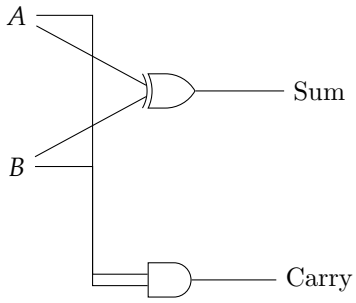
To add two single-bit binary numbers A and B, we need to calculate: 1. The **sum bit**, which is the XOR of A and B. 2. The **carry bit**, which is the AND of A and B.

Logic - Sum Bit: Sum = $A \oplus B$ - **Carry Bit:** Carry = $A \cdot B$

The following truth table summarizes the operation:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The addition is implemented using XOR and AND gates as shown below:



The XOR gate produces the sum bit, while the AND gate produces the carry bit. This forms the fundamental building block of a full binary adder.

1.2 Adding Two Single-Byte Binary Numbers Using XOR and AND

Binary addition can be performed on two single-byte numbers using bitwise operations. Specifically: - The **XOR (Exclusive OR)** operation gives the sum of two bits without considering the carry. - The **AND** operation identifies where a carry will occur. - The carry is then shifted left by one position and added to the result in subsequent iterations.

Algorithm: 1. Compute the sum without carry using XOR:

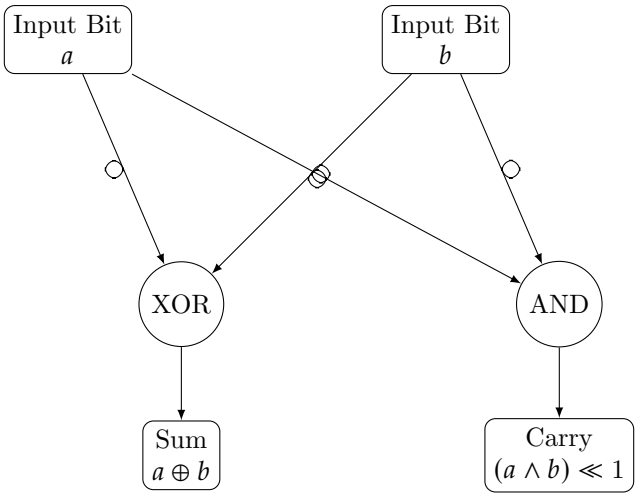
$$\text{sum} = a \oplus b$$

2. Compute the carry using AND and shift it left by one bit:

$$\text{carry} = (a \wedge b) \ll 1$$

3. Repeat the process by adding the carry to the sum until there is no carry left.

Diagram: Below is a diagram that illustrates the process for a single bit addition.



Example: Consider two single-byte binary numbers:

$$a = 01101101, \quad b = 10101001$$

1. Compute the XOR for the sum:

$$\text{sum} = a \oplus b = 11000100$$

2. Compute the AND for the carry and shift left:

$$\text{carry} = (a \wedge b) \ll 1 = 00001010$$

3. Add the sum and carry: - New sum: sum = $11000100 \oplus 00001010 = 11001110$ - New carry: carry = $(11000100 \wedge 00001010) \ll 1 = 00000000$

4. Final result: 11001110.

1.3 One's Complement for a Single Byte

The **one's complement** of a binary number is obtained by flipping all the bits, changing every `1` to `0` and every `0` to `1`. This operation is commonly used in binary arithmetic, particularly in representing negative numbers in early computing systems.

Steps to Compute One's Complement: 1. Write down the binary number. 2. Flip all bits: - Change each `0` to `1`. - Change each `1` to `0`.

Example: Consider the binary number $a = 01101101$.

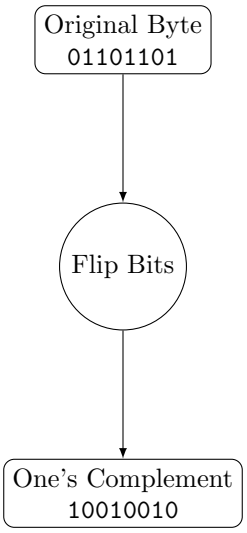
1. Original binary number:

$$a = 01101101$$

2. One's complement (flip all bits):

$$\text{one's complement of } a = 10010010$$

Diagram: The following diagram illustrates the transformation:



Properties of One's Complement: - A binary number and its one's complement always sum to all 1s (i.e., 11111111 for a single byte). - One's complement is useful in representing signed integers: - Positive numbers are represented as-is. - Negative numbers are represented by the one's complement of their positive counterparts.

Example with Signed Integers: For a single byte: 1. +5 in binary: 00000101 2. −5 in one's complement: 11111010

Verification: Adding +5 and −5 in one's complement arithmetic:

$$00000101 + 11111010 = 11111111 \quad \text{(all bits are 1, representing zero in one's complement arithmetic).}$$

1.4 Carrying When Adding Negatives in One's Complement

In one's complement representation, negative numbers are represented by flipping all the bits of their positive counterparts. When adding two negative numbers, a carry might be generated, which needs to be added back to the result to obtain the correct answer.

Example: Adding −3 and −4.

1. Represent −3 and −4 in one's complement for a single byte:

$$+3 = 00000011, \quad -3 = \text{one's complement of } +3 = 11111100$$

$$+4 = 00000100, \quad -4 = \text{one's complement of } +4 = 11111011$$

2. Add the two numbers:

$$\begin{array}{r} 11111100 \\ +11111011 \\ \hline 11110111 \end{array}$$

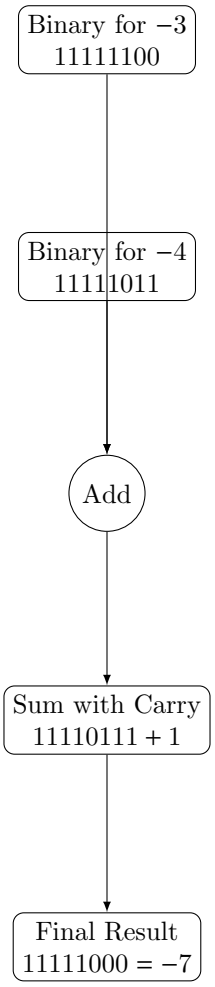
3. Handle the carry: - The result of the addition is 11110111, with a carry bit of 1. - Add the carry back to the least significant bit:

$$11110111 + 1 = 11111000$$

4. Final result:

$$11111000 = \text{one's complement of } +7 = -7$$

Explanation: - When the sum generates a carry, it must be added back to the result to comply with one's complement rules. - The result of $-3 + -4 = -7$, as expected.



Note: This approach works for signed integers in one's complement and highlights the importance of handling the carry bit to ensure accurate results.

1.5 Two's Complement for a Single Byte

The **two's complement** of a binary number is obtained by flipping all the bits (as in one's complement) and then adding 1 to the result. This method is widely used in modern computer systems to represent signed integers.

Steps to Compute Two's Complement: 1. Write down the binary number. 2. Flip all bits (as in one's complement). 3. Add 1 to the flipped number.

Example: Consider the binary number $a = 01101101$.

1. Original binary number:

$$a = 01101101$$

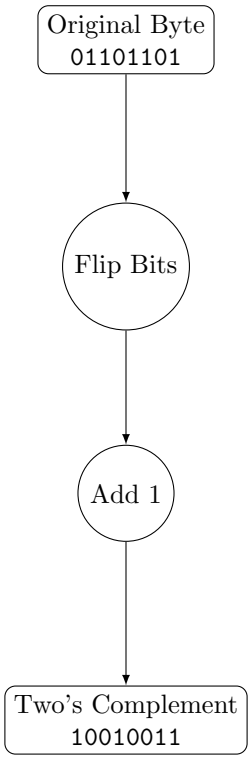
2. Flip all bits (one's complement):

$$10010010$$

3. Add 1 to the result:

$$\text{two's complement of } a = 10010010 + 1 = 10010011$$

Diagram: The following diagram illustrates the transformation:



Properties of Two's Complement: - The two's complement of 0 is 0, and the two's complement of the maximum negative value is itself. - Negative numbers are represented by their two's complement. - Addition and subtraction with two's complement do not require separate subtraction logic, simplifying arithmetic operations.

1.6 Carrying When Adding Negatives in Two's Complement

In two's complement representation, negative numbers are represented by flipping all bits of the positive number and adding 1. When adding two negative numbers, the carry generated during addition is discarded.

Example: Adding -3 and -4.

1. Represent -3 and -4 in two's complement for a single byte:

$$+3 = 00000011, \quad -3 = \text{two's complement of } +3 = 11111101$$

$$+4 = 00000100, \quad -4 = \text{two's complement of } +4 = 11111100$$

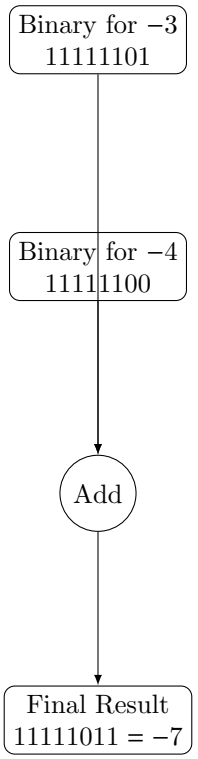
2. Add the two numbers:

$$\begin{array}{r} 11111110 \\ +11111100 \\ \hline 11111011 \end{array}$$

3. Handle the carry: - The result of the addition is 11111011. - In two's complement, the carry is ignored, so the final result is:

$$11111011 = -7$$

Explanation: - In two's complement, the carry bit is discarded, unlike in one's complement. - The result of $-3 + -4 = -7$, as expected.



Note: Two's complement simplifies arithmetic operations by eliminating the need to add back the carry. It is the most commonly used method for representing signed integers in modern computing systems.