| Experiment No. 2 |
| --- |
| Implement a program to obtain 2’s complement of a number |
| Date of Performance: |
| Date of Correction: |

**Aim:** To implement a program that computes the 2’s complement of a given binary number.

**Objective:** To understand and apply the concept of 2’s complement for representing and manipulating signed binary numbers.

**Theory:**

In digital systems, particularly in computer architecture, numbers can be either unsigned (only positive) or signed (both positive and negative). While unsigned binary numbers are straightforward, representing negative numbers requires a special system. The 2’s complement system is the most widely used method for representing signed integers in binary.

**Why Use 2’s Complement?**

1. **Unified Arithmetic:** Addition, subtraction, and other operations can be performed using the same hardware logic for both positive and negative numbers.
2. **Single Representation of Zero:** Unlike 1’s complement, which has two representations of 0 (+0 and -0), 2’s complement has only one.
3. **Efficient Hardware Implementation:** Simplifies ALU design in CPUs and supports overflow detection.

**How 2’s Complement Works:**

**To find the 2’s complement of a binary number:**

1. Invert all bits (change 0 to 1 and 1 to 0) — this gives the 1’s complement.
2. Add 1 to the result — this gives the 2’s complement.

**Example:  
Let’s find the 2’s complement of 00010110 (which is +22 in decimal):**

* Step 1: Invert all bits → 11101001
* Step 2: Add 1 → 11101010

**The result, 11101010, is the 2’s complement binary representation of -22.**

**Applications of 2’s Complement:**

* Used in Arithmetic Logic Units (ALUs) of processors to perform subtraction.
* Essential in assembly language programming and low-level software development.
* Helps in handling negative values efficiently in hardware design, compilers, and system programming.

**Key Points:**

* Most Significant Bit (MSB) is used as the sign bit in 2’s complement (0 for positive, 1 for negative).
* In an n-bit system, the range of representable numbers is from −2ⁿ⁻¹ to (2ⁿ⁻¹ − 1).
  + For 8-bit numbers: Range is −128 to +127.
* Overflow and underflow conditions can occur and must be handled carefully.

**Solution:**

**Conclusion:** We understood the concept of 2’s complement and how it is used to represent negative numbers in binary. This gave us insight into how subtraction is performed in digital systems.