**Topic: Logical binary shifts**

Reading Time: 20 mins

**·        Note\* Highlight important/core points while reading**

·        Read the content and write the answers given in the document in your words, to get the solid grip on topic.

**Logical Binary Shifts**

**What is a Logical Binary Shift?**

A **logical binary shift** is a bitwise operation used in computing to move binary digits **left or right** within a number. It is commonly used for **multiplication, division, and bitwise operations** in programming and hardware design.

There are two types of logical shifts:

1. **Logical Left Shift (<<)**
2. **Logical Right Shift (>>)**

**1. Logical Left Shift (<<)**

A **logical left shift** moves all the bits **one or more places to the left**, filling the empty rightmost bits with **0s**.

**Effect of a Left Shift**

* Each **left shift by one place** multiplies the number by **2**.
* If shifting causes a **1 to move out of the leftmost bit**, it is lost (**overflow**).

**Example of Left Shift by 1 bit:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary (Before)** | **0** | **0** | **1** | **0** | **1** | **1** | **(11 in decimal)** |
| Left Shift (<<) | 0 | 1 | 0 | 1 | 1 | 0 | (22 in decimal) |

**Binary calculation:**

 001011  (11 in decimal)

<< 1

------------

 010110  (22 in decimal)

* 11₁₀ × 2 = 22₁₀

**Example of Left Shift by 2 bits:**

   001011  (11 in decimal)

<< 2

------------

   101100  (44 in decimal)

* **11₁₀ × 2² = 44₁₀**

**Key Points of Logical Left Shift:**

·         Each shift **doubles** the number.

·         Empty rightmost bits are **filled with 0s**.

·         If a **1 moves out of the leftmost bit**, **overflow occurs**.

**2. Logical Right Shift (>>)**

A **logical right shift** moves all the bits **one or more places to the right**, filling the empty leftmost bits with **0s**.

**Effect of a Right Shift**

* Each **right shift by one place** **divides the number by 2** (ignoring the remainder).

**Example of Right Shift by 1 bit:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Binary (Before)** | **1** | **0** | **1** | **1** | **0** | **0** | **(44 in decimal)** |
| Right Shift (>>) | 0 | 1 | 0 | 1 | 1 | 0 | (22 in decimal) |

**Binary calculation:**

   101100  (44 in decimal)

>> 1

------------

   010110  (22 in decimal)

* 44₁₀ ÷ 2 = 22₁₀

**Example of Right Shift by 2 bits:**

   101100  (44 in decimal)

>> 2

------------

   001011  (11 in decimal)

* 44₁₀ ÷ 2² = 11₁₀

**Key Points of Logical Right Shift:**

·         Each shift **divides** the number by **2**.

·         Empty leftmost bits are **filled with 0s**.

·         If the **number is odd, the remainder is lost** (e.g., 5 >> 1 becomes 2, not 2.5).

**A-Rated Questions/Answers By Examiner**

**Q1: What happens when you perform a logical left shift on a binary number?**

**Answer:** A logical left shift **moves all bits to the left** and **fills the rightmost bit with 0**. Each shift **multiplies the number by 2**.

**Q2: Perform a left shift by 2 bits on 00011001₂. What is the new value in decimal?**

**Answer:**

* **00011001₂ (25 in decimal)**
* Left shift by 2 bits: **01100100₂**
* **01100100₂ = 100₁₀**
* Answer: **100 in decimal**

**Q3: Perform a right shift by 1 bit on 00101100₂. What is the new value in decimal?**

**Answer:**

* **00101100₂ (44 in decimal)**
* Right shift by 1 bit: **00010110₂**
* **00010110₂ = 22₁₀**
* Answer: **22 in decimal**

**Q4: What happens if a logical left shift causes a 1 to move out of the leftmost bit?**

**Answer:** **Overflow occurs**, and the lost bit cannot be recovered.

**Q5: Explain why a logical right shift of an odd number results in data loss.**

**Answer:** Since **binary division by 2 ignores remainders**, shifting an odd number **removes the least significant bit (1)**, causing data loss.

### Write your Answers on your Notebook and Verify it on Next Screen

**Q6: What is the result of performing a left shift by 3 bits on 00001011₂?**

**Q7: How does a logical left shift affect overflow in a fixed-bit system?**

**Q8: Why is a logical right shift not suitable for signed numbers?**

**Q9: Perform a right shift by 3 bits on 10101000₂. What is the new value in decimal?**

**Q10: What is the mathematical effect of shifting a binary number left by N bits?**

**6. Answer:**

* 00001011₂ = **11** in decimal.
* Left shift by 3 bits: **01011000₂**.
* **01011000₂ = 88₁₀**.
* **Answer:** **88 in decimal**.

**7. Answer:**

* A logical left shift **doubles the number** with each shift.
* If the leftmost bit is **1**, it may **move out of the bit limit**, causing **overflow**.
* Example: In an **8-bit system**, shifting **11000000₂ (192₁₀) left by 1** results in **10000000₂ (128₁₀)**, which can cause an **overflow error**.

**8. Answer:**

* A logical right shift **fills empty leftmost bits with 0**, which can **change the sign** of a signed number.
* Example:
  + **Signed 8-bit representation of -5** = **11111011₂**.
  + Right shift by 1 → **01111101₂** (**now positive!**).
* This incorrect behavior makes logical right shifts unsuitable for signed numbers.

**9. Answer:**

* **10101000₂** = **168₁₀**.
* Right shift by **3** bits: **00010101₂**.
* **00010101₂ = 21₁₀**.
* **Answer:** **21 in decimal**.

**10. Answer:**

* A left shift by **N** bits **multiplies** the number by **2ⁿ**.
* Example:
  + **5₁₀ (00000101₂) << 3** → **00101000₂ = 40₁₀**.
  + **5 × 2³ = 40**.
* **Answer:** **Shifting left by N bits multiplies the number by 2ⁿ**.