Binary shifts

Logical (unsigned numbers)

Arithmetic (signed numbers)

Binary shifts move the bits in a number left or right, multiplying (left) or dividing (right) by powers of 2. Eg. shift 1 place, multiply or divide by 2; shift 2 places multiply or divide by; shift 3 places, multiply or divide by 8 etc.

Left = multiply Right = divide

Logical shifts are applied to **UNSIGNED** numbers. These fill empty spaces with 0s.

Arithmetic shifts are applied to **SIGNED** numbers. A left arithmetic shift acts in the same way as a left logical shift, filling empty spaces with 0s. A right arithmetic keeps the MSB (the left hand bit which holds the sign) and fills all spaces **with copies of the MSB**.

Advantages and disadvantages of binary shifts

Advantage	Explanation	Disadvantage	Explanation
Efficiency	Binary shifts are very efficient operations for computers to perform. Shifting bits requires simpler logic compared to multiplication or division algorithms making them faster to execute.	**Limited Functionality**	They only work for powers of 2.
Simplicity	They are relatively simple to understand and to code especially for logical shifts.	**Overflow/Underflow –**	For arithmetic shifts, there's a risk of overflow (positive number becomes too large) or underflow (negative number becomes too small) if not handled carefully.
Power of 2 operations	They are great at multiplying or dividing by powers of 2 which are common operations in various computing tasks like memory addressing and image editing	**Potential loss of precision**	When right shifting (dividing) there is a potential loss of precision depending on the original number because decimal numbers aren't represented. So odd numbers are rounded down.
		Sign Bit Consideration	For arithmetic right shifts, it's important to consider the sign bit to preserve the original number's positive or negative value.

Questions

1. A **logical binary shift** moves the bits in a binary number to the left or right, filling the empty spaces with:

(a) Random bits

(b) 1s

(c) 0s

(d) The sign bit

2. What is the effect of a **logical left shift** 1 place on the value of a binary number?

(a) No change

(b) Reduces the value by half

(c) Increases the value by half

(d) Doubles the value

3. In an **arithmetic right shift**, the empty space created on the left side is filled with:

(a) Random bits

(b) 0s

(c) 1s

(d) The value of the most significant bit

4. An **arithmetic right shift** of 2 places on a signed binary number is equivalent to:

(a) Multiplying by 2

(b) Dividing by 2

(c) Multiplying by 4

(d) Dividing by 4

5. Which type of shift (logical or arithmetic) would be most appropriate when multiplying an unsigned binary number by 2?

(a) Arithmetic right shift

(b) Logical right shift

(c) Logical left shift

(d) Arithmetic left shift

6. Explain the difference between a logical left shift and a logical right shift in terms of how they affect the bits in a binary number.

7. Why is it important to consider the sign bit (MSB) when performing an arithmetic right shift?

8. In what scenario might you use a logical shift operation in your program?

ANSWERS

1. C) The spaces in a logical shift bitwise operation are filled with 0s.

2. D) Left shift by 1 place will multiply the value by 2.

3. D) The MSB is copied into empty bits in a right arithmetic shift.

4. D) Right shifts divide, whether logical or arithmetic. By 2 places, means divide by 4

5. C) Logical left shift. Logical because the number is unsigned. Left because the question asks about multiplication

6. A logical shift is used on an unsigned number. A left shift results in the multiplication of the binary number. In a left shift all the bits are moved to the left by the stated number of places. The empty bits on the right are filled with 0. A right shift results in division of the binary number. All of the bits are shifted to the right leaving spaces on the left-hand side. These spaces are filled with 0s.

7. An arithmetic shift is performed on a signed binary number. The MSB of a signed number holds the sign; 0 for positive and 1 for negative. It is important that the sign doesn't change when performing a binary shift because this would incorrectly change a negative number into a positive and vice versa.

8. A left logical shift would be used in a program when a binary number needed to be multiplied by a power of 2. An example of when this could be used would be in image editing. For example, doubling the size of an image.