Shift Micro Operations

Shift micro-operations are used when data is stored in registers and are commonly applied in the serial transmission of data. These operations involve moving bits of data either to the left or right within a register. They are often combined with arithmetic and logic operations for efficient data processing. Shift micro-operations are essential in various tasks like multiplying or dividing binary numbers, performing logical operations and manipulating data in registers.

There are three main types of shift micro-operations:

* Logical Shift
* Arithmetic Shift
* Circular Shift

**Logical Shift**

A logical shift micro-operation involves moving the bits of data in a register either to the left or right, with zeroes being introduced into the empty positions. This operation is used primarily in binary arithmetic and data processing tasks.

There are two main types of logical shift micro-operations:

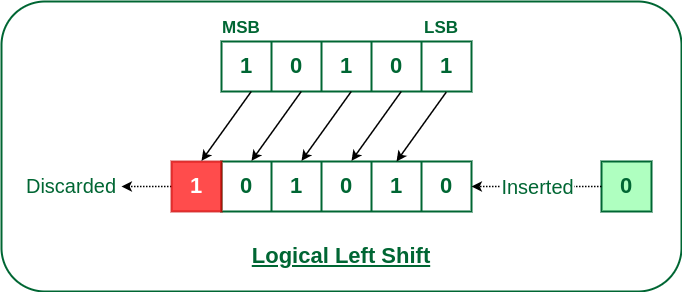
**1. Logical Left Shift**

In this shift, each bit is moved to the left by one position. The Empty least significant bit (LSB) is filled with zero (i.e. the serial input), and the most significant bit (MSB) is rejected or discarded. The left shift operator is denoted by the double left arrow key (<<).

The general syntax for the left shift is

*"shift-expression" << k*

**Note:**For unsigned numbers, every time we shift a number towards the left by 1 bit it multiplies that number by 2.

Logical Left Shift

**Usage:**Used in multiplication of unsigned binary numbers.

**Example:**Let's take an 8-bit unsigned binary number 01010011 (which is 83 in decimal). If we perform a logical shift left:

* **Before Shift:** 01010011 (83 in decimal)
* **After Logical Shift Left:**10100110 (166 in decimal)

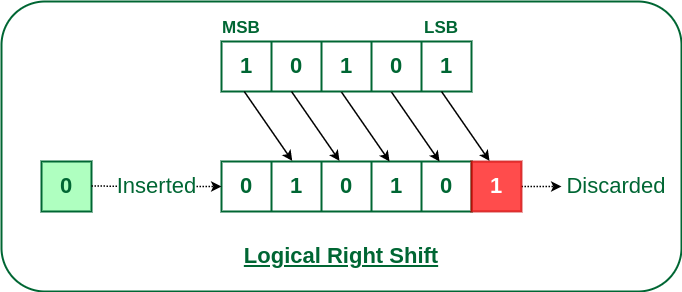
**2. Logical Right Shift**

In this shift, each bit moves to the right one by one. The least significant bit(LSB) is discarded and empty MSB is filled with zero. The right shift operator is denoted by the double right arrow key (>>).

The general syntax for the right shift is

*“shift-expression" >> k*

**Note:** For unsigned numbers, every time we shift a number towards the right by 1 bit it divides that number by 2.

Logical Right Shift

**Usage:** Used in division of unsigned binary numbers.

**Example:** Let's take the same 8-bit binary number 01010011 (83 in decimal). If we perform a logical shift right:

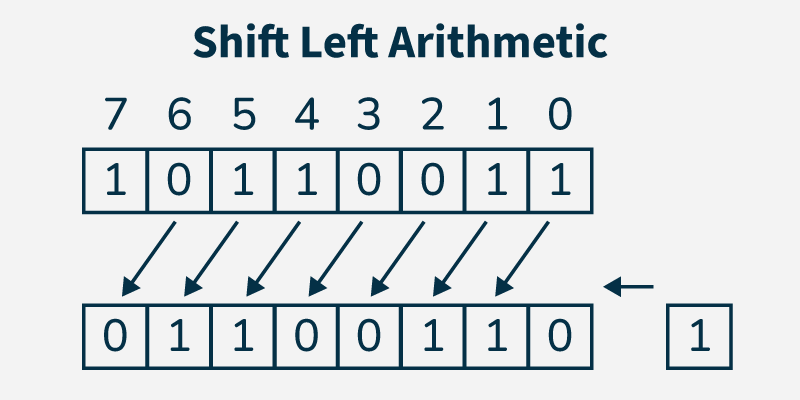
* **Before Shift:** 01010011 (83 in decimal)
* **After Logical Shift Right:**00101001 (41 in decimal)

**Arithmetic Shift**

The arithmetic shift micro-operation moves the signed binary number either to the left or to the right position. Following are the two ways to perform the arithmetic shift:

**1. Arithmetic Left Shift**

In this shift, each bit is moved to the left one by one. The empty least significant bit (LSB) is filled with zero, and the most significant bit (MSB) is rejected same as the Left Logical Shift. However, arithmetic left shift differs conceptually in that it is specifically interpreted as multiplication by two, especially regarding signed integer operations, whereas logical left shift does not inherently assume numeric interpretation.

Arithmetic Shift Left

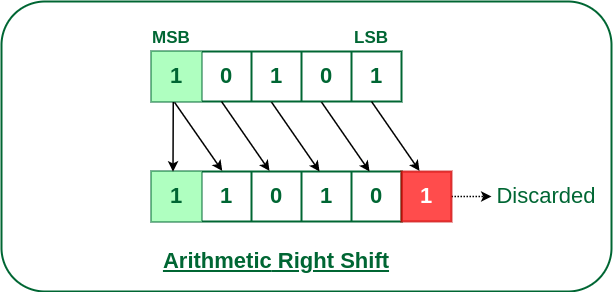
**Usage:** Used for multiplying signed binary numbers by powers of 2.

**Example:**Let's take a signed 8-bit binary number in two's complement form: 11111111 (which represents -1 in decimal). If we perform an arithmetic shift left:

* **Before Shift:**11111111 (-1 in decimal)
* **After Arithmetic Shift Left:**11111110 (-2 in decimal)

**2. Arithmetic Right Shift**

In this shift, each bit is moved to the right one by one and the least significant(LSB) bit is rejected and the empty most significant bit(MSB) is filled with the value of the previous MSB.

Arithmetic Right Shift

**Usage:** Used for dividing signed binary numbers by powers of 2.

**Example:**Let's take a signed 8-bit binary number in two's complement form: 11111111 (which represents -1 in decimal). If we perform an arithmetic shift right:

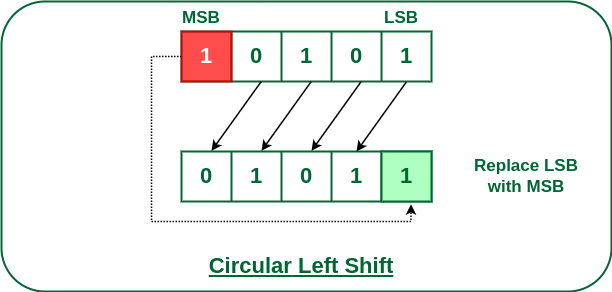
* **Before Shift:**11111111 (-1 in decimal)
* **After Arithmetic Shift Right:**11111111 (-1 in decimal)

**Circular Shift**

The circular shift circulates the bits in the sequence of the register around both ends without any loss of information. Following are the two ways to perform the circular shift:

**1. Circular Left Shift**

In this micro shift operation each bit in the register is shifted to the left one by one. After shifting, the LSB becomes empty, so the value of the MSB is filled in there.

Circular Left Shift

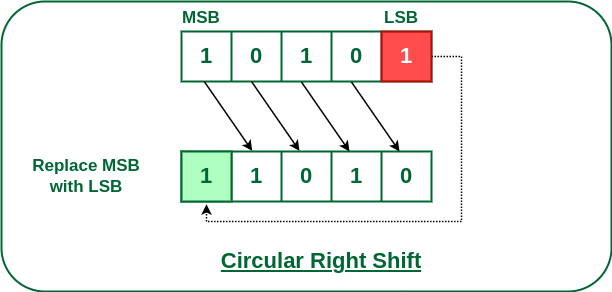
**Usage:** Used in data encryption algorithms and certain arithmetic operations, where the shift should be cyclic.

**Example:**Let's take an 8-bit binary number 11010011. If we perform a circular shift left:

* **Before Shift:** 11010011
* **After Circular Shift Left:** 10100111

**2. Circular Right Shift**

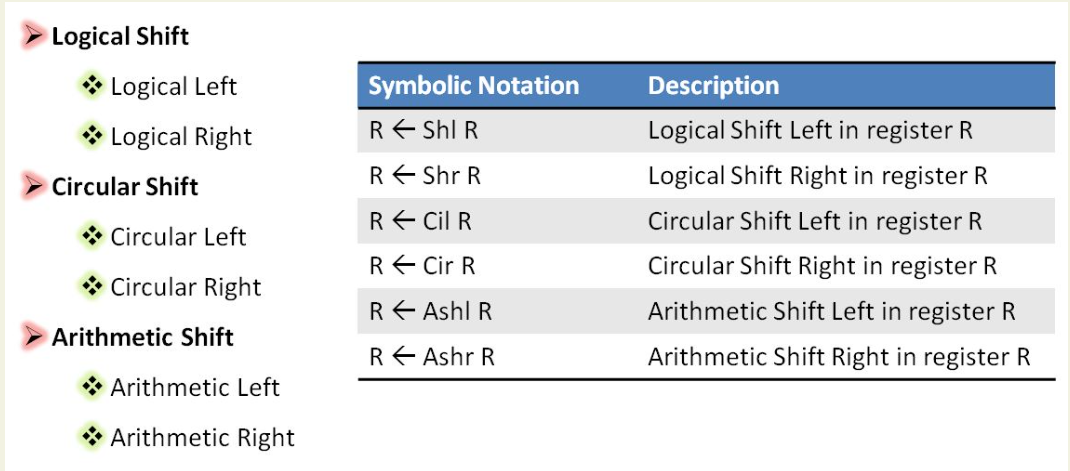
In this micro shift operation each bit in the register is shifted to the right one by one. After shifting, the MSB becomes empty, so the value of the LSB is filled in there.

Circular Right Shift

**Usage:**Used in situations where rotation of bits is required, such as in encryption algorithms.

**Example:**Let's take the same 8-bit binary number 11010011. If we perform a circular shift right:

* **Before Shift:**11010011
* **After Circular Shift Right:** 11101001



HARDWARE IMPLEMENTATION

