Learning Objectives

- To understand how to use a binary shift
- Binary numbers can be shifted to multiply or divide them

Binary Shifts

- **Binary shifts** are used to **move all the bits** in a binary pattern left or right
- Logical shifts treat all the bits of binary pattern in the same way
- To multiply binary numbers, the bits are shifted to the left
- To divide binary numbers, the bits are shifted to the right

A logical shift left (LSL) moves each bit left by n positions

Worked example

Perform a logical shift left of two positions on the binary pattern 0001 0100.

Move each bit two positions to the left. Discard the two leftmost bits. Fill up the empty spaces on the right with 0s.

Before shift	0	0	0	1	0	1	0	0
After shift	0	1	0	1	0	0	0	0

The result is 0101 0000.

- Left shift is used to multiply

A logical shift right(LSR) moves each bit right by n positions

Worked example

Perform a logical shift right of three positions on the binary pattern 0001 1000.

Move all the bits three positions to the right. Discard the three rightmost bits. Fill up the empty spaces on the left with 0s.

Before shift	0	0	0	1	1	0	0	0
After shift	0	0	0	0	0	0	1	1

The result is 0000 0011.

- right shift is used to divide

Loss of precision

Sometimes a binary shift produces an imprecise result. This can happen when the pattern is interpreted as numbers. For example, if we shift 0010 0001 (+33) by one position right (a division by 2) the rightmost bit is lost, producing a result of +16, which is imprecise.

Place values	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Before shift	0	0	1	0	0	0	0	1
After shift	0	0	0	1	0	0	0	0

Figure 2.1.3 In this case shifting right produces an imprecise result

Loss of precision occurs because one or more of the right-most bits are discarded during the shift.

Rules for binary shift

Multiply or divide by	Shifts
$2 = 2^1$	1
$4 = 2^2$	2
$8 = 2^3$	3
$16 = 2^4$	4
$32 = 2^5$	5
$64 = 2^6$	6
$128 = 2^7$	7

Arithmetic shift – Right Shift

- An arithmetic shift is a binary shift used with 2's complement
- An arithmetic shift right (ASR) moves each bit right by n positions.
- **Fill rule:**
  - The **most significant bit (MSB)** — the *sign bit* — is copied into the empty positions on the left.
  - If MSB = 0 → fill with 0s (number is positive).
  - If MSB = 1 → fill with 1s (number is negative).

- This preserves the **sign** of the number.

Worked example

Perform an arithmetic shift right of one position on the binary pattern 1011 0000. Move all the bits one position to the right. Discard the rightmost bit. Put a 1 in the empty space on the left.

Before shift	1	0	1	1	0	0	0	0
After shift	1	1	0	1	1	0	0	0

The result is 1101 1000.

Arithmetic shifts – Left

Before shift	1	1	0	1	0	1	0	1
After shift	0	1	0	1	0	1	0	0

Same as left logical shift – fill with 0's

However a left shift on a 2's complement will lose the MSB (negative sign)