

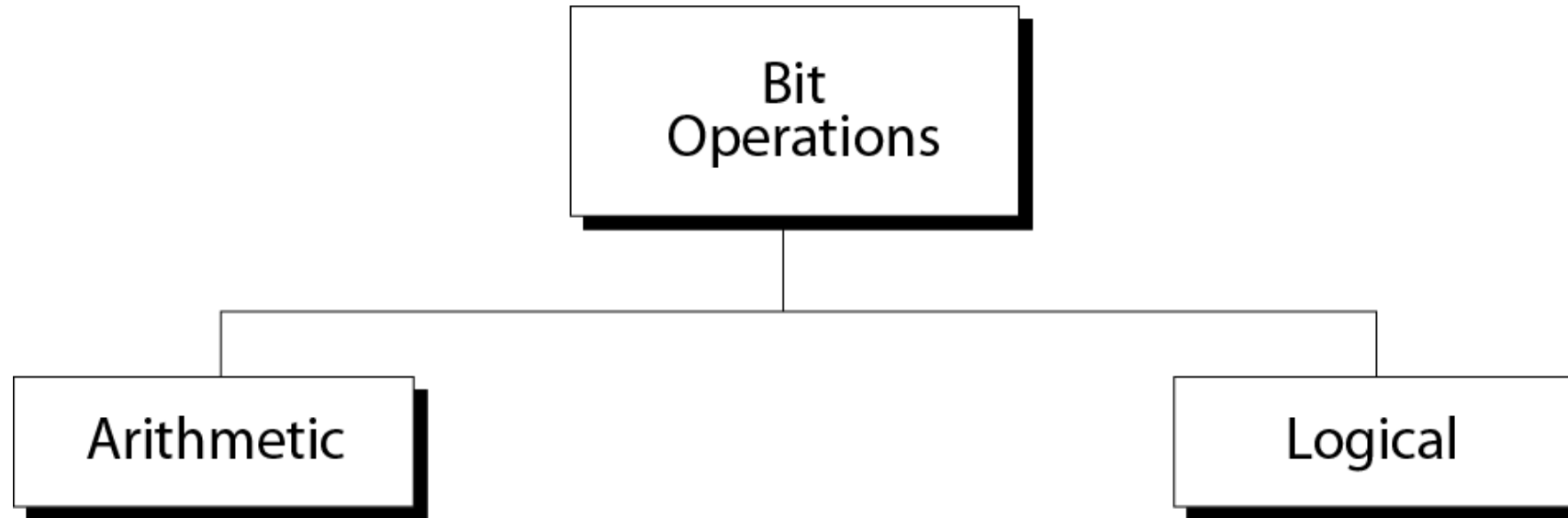
IT111 - Introduction to Information Technology Bachelor of Science in Information Technology

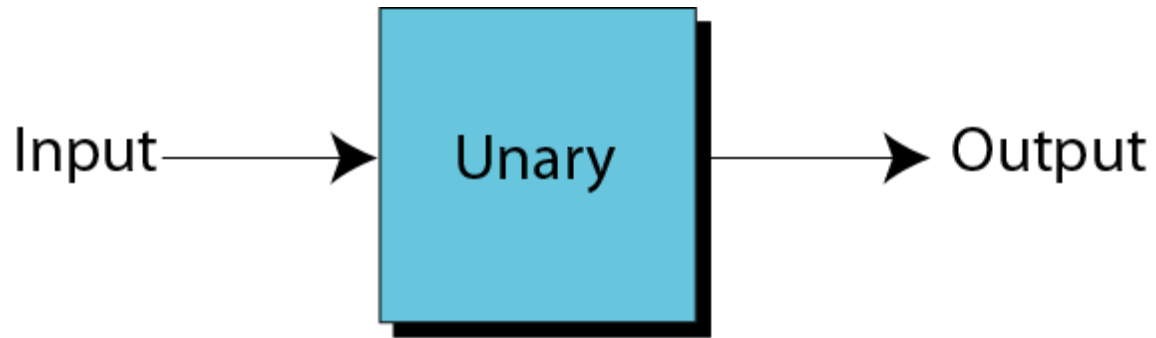
Eng. Eman Hamdy

Lecture 5 - Operation on Data

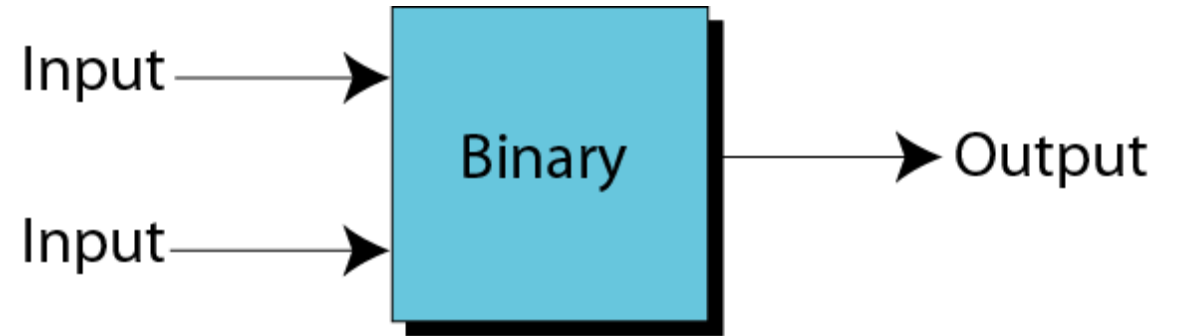
Objectives

1. List the three categories of operations performed on data.
2. Perform unary and binary logic operations on bit patterns.
3. Perform logic shift operations on bit patterns.
4. Understand some applications of logic and shift operations such as setting, unsetting, and flipping bits.

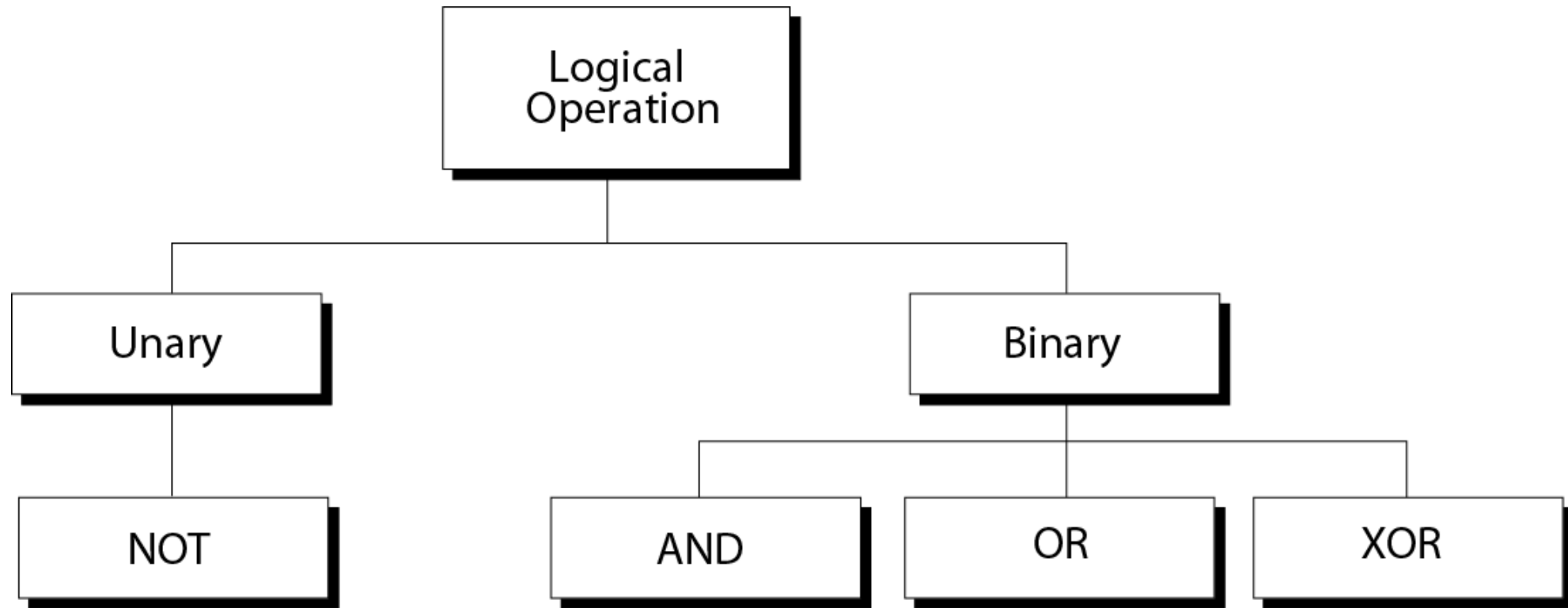




a. Unary operator



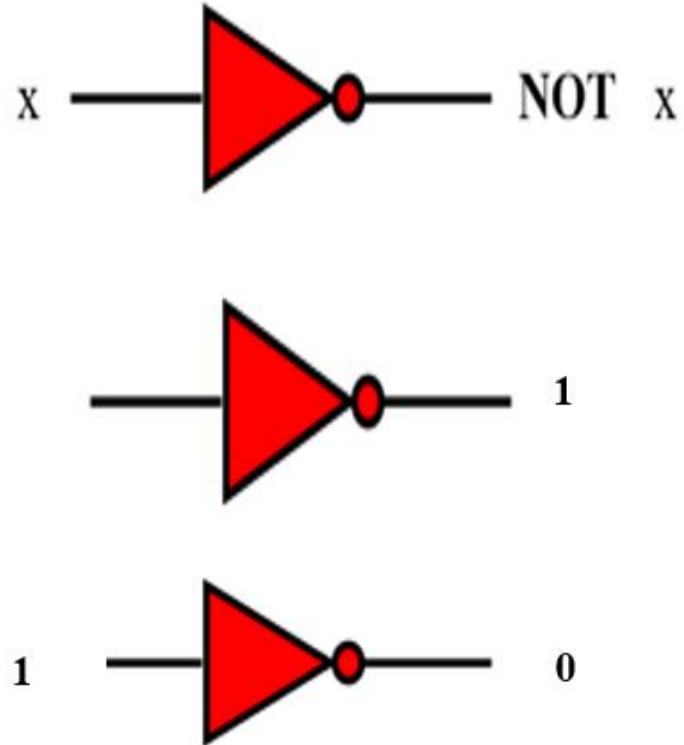
b. Binary operator

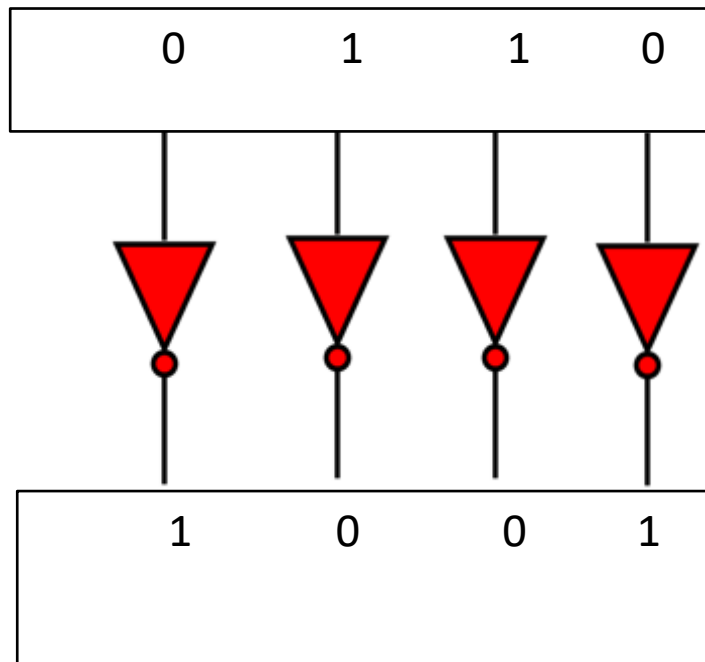


Logic operations refer to those operations that apply the same basic operation on individual bits of a pattern, or on two corresponding bits in two patterns.

This means that we can define logic operations:

- at the **bit level**: A bit can take one of the two values: 0 or 1.
- at the **pattern level**



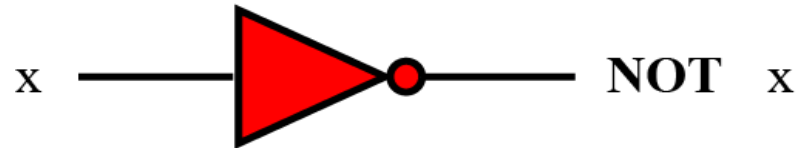


4 Logical operation(Not Gate/Not Operator)

a logic operation at the pattern level is n logic operations, of the same type, at the bit level where n is the number of bits in the pattern.

NOT

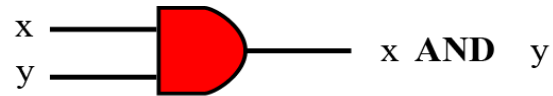
The NOT operator is a unary operator: it takes only one input. The output bit is the complement of the input.

**NOT**

x	NOT x
0	1
1	0

AND

The AND operator is a binary operator: it takes two inputs. The output bit is 1 if both inputs are 1s and the output is 0 in the other three cases.

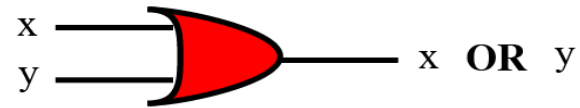


AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

OR

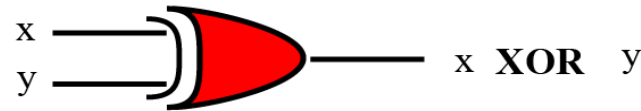
The OR operator is a binary operator: it takes two inputs. The output bit is 0 if both inputs are 0s and the output is 1 in the other three cases.

**OR**

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

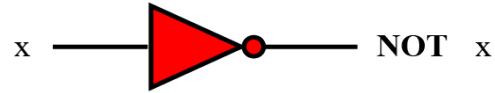
XOR

The XOR operator is a binary operator like the OR operator, with only one difference: the output is 0 if both inputs are 1 or 0.



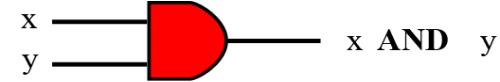
XOR

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0



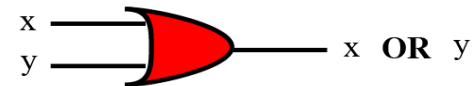
NOT

x	NOT x
0	1
1	0



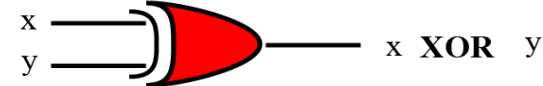
AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



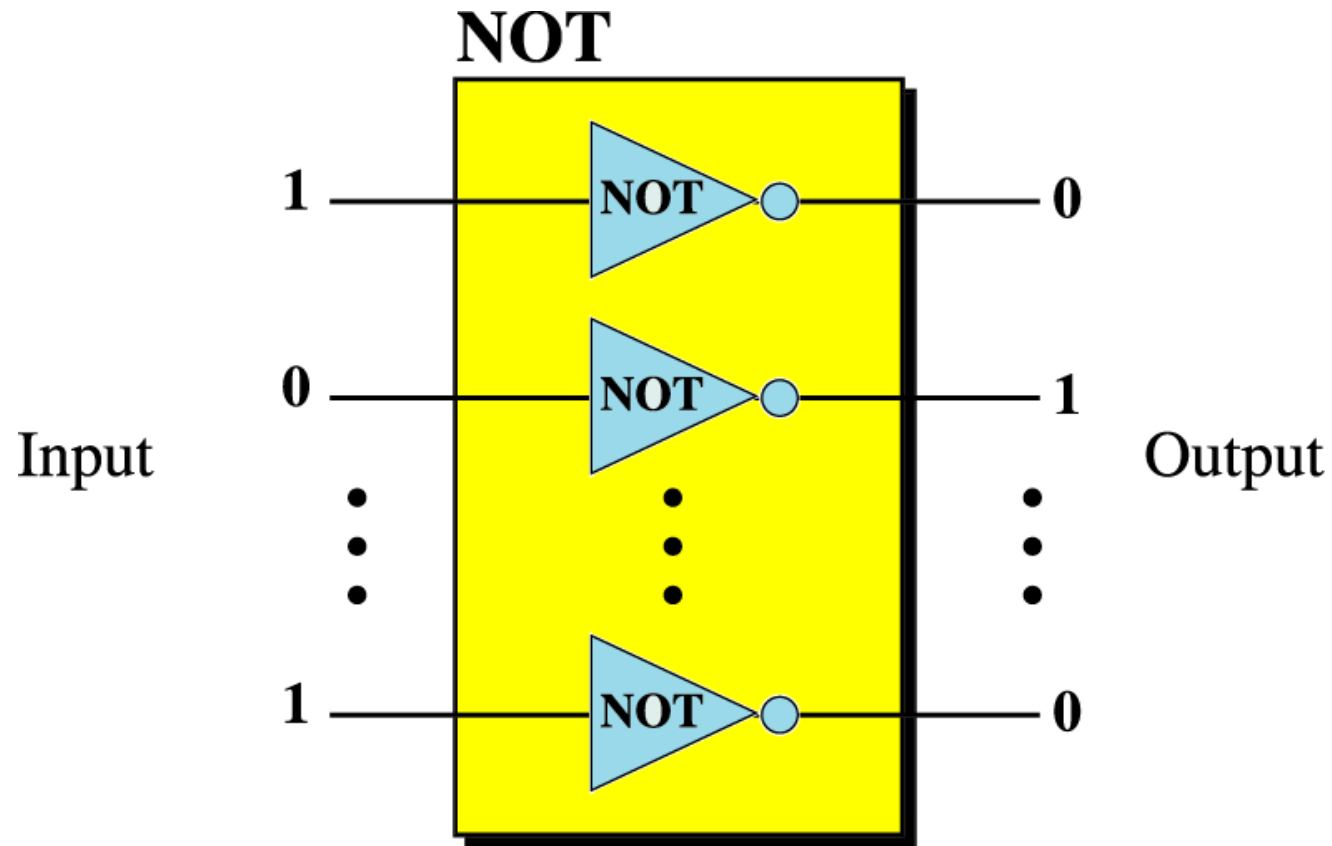
OR

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1



XOR

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

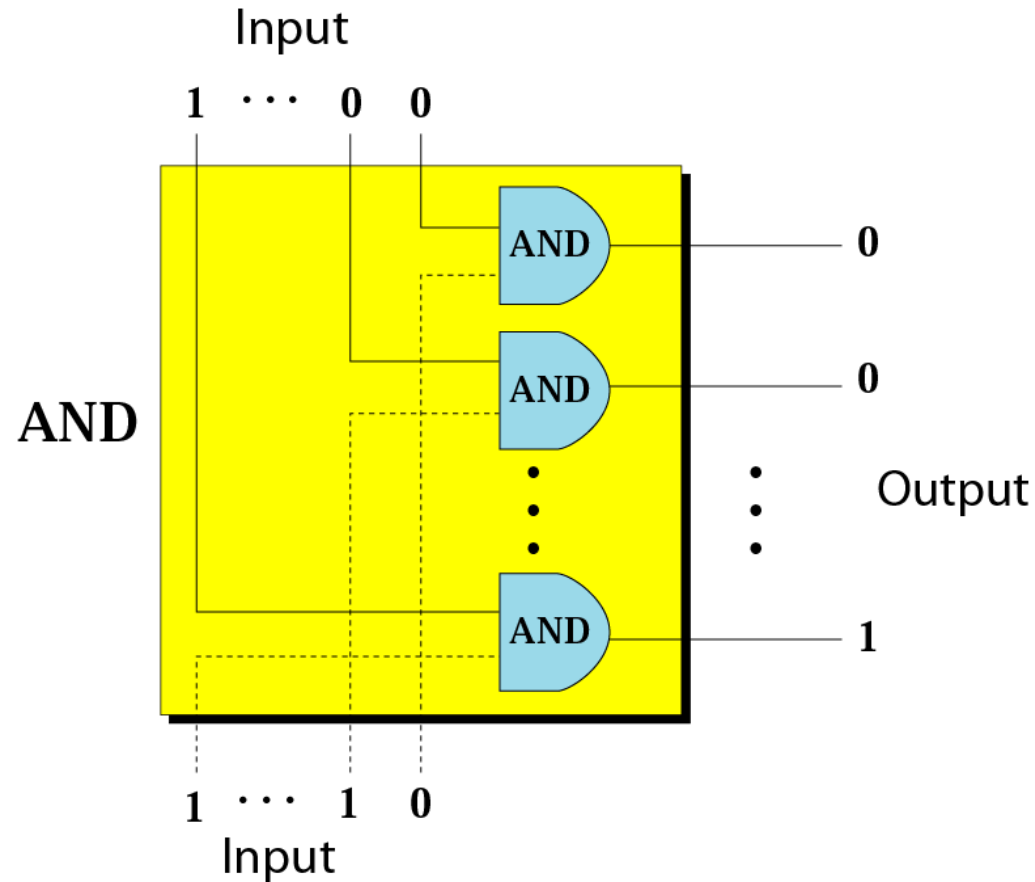


Use the NOT operator on the bit pattern 10011000.

Solution

The solution is shown below. Note that the NOT operator changes every 0 to 1 and every 1 to 0.

NOT	1	0	0	1	1	0	0	0	Input
	0	1	1	0	0	1	1	1	Output

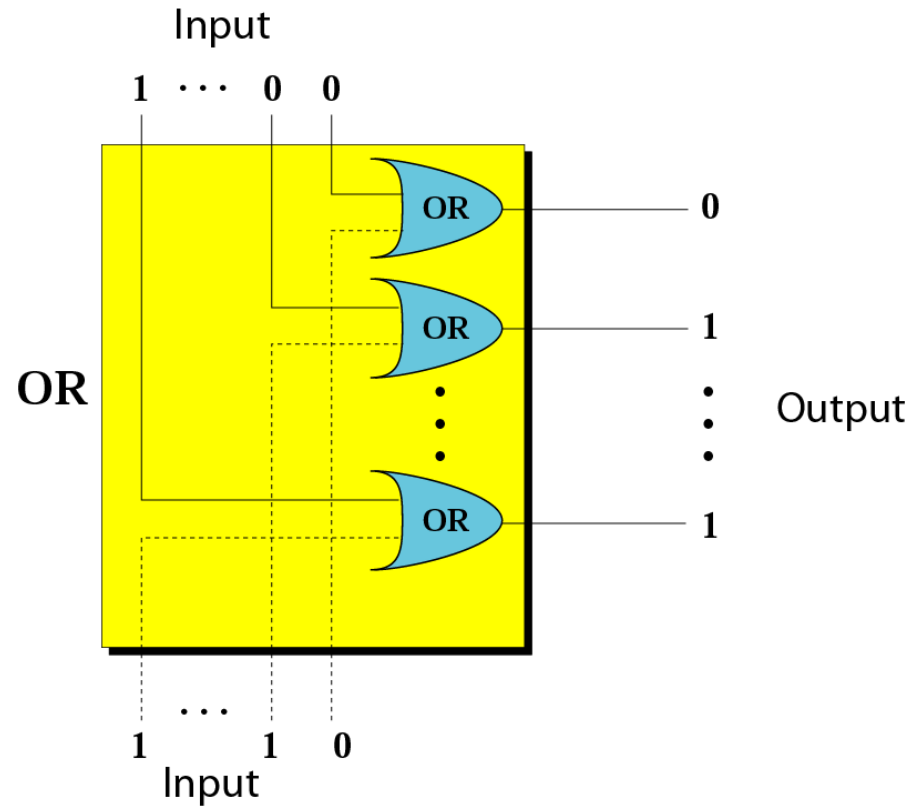


Use the AND operator on the bit patterns 10011000 and 00101010.

Solution

The solution is shown below. Note that only one bit in the output is 1, where both corresponding inputs are 1s.

	1	0	0	1	1	0	0	0	Input 1
AND	0	0	1	0	1	0	1	0	Input 2
	0	0	0	0	1	0	0	0	Output

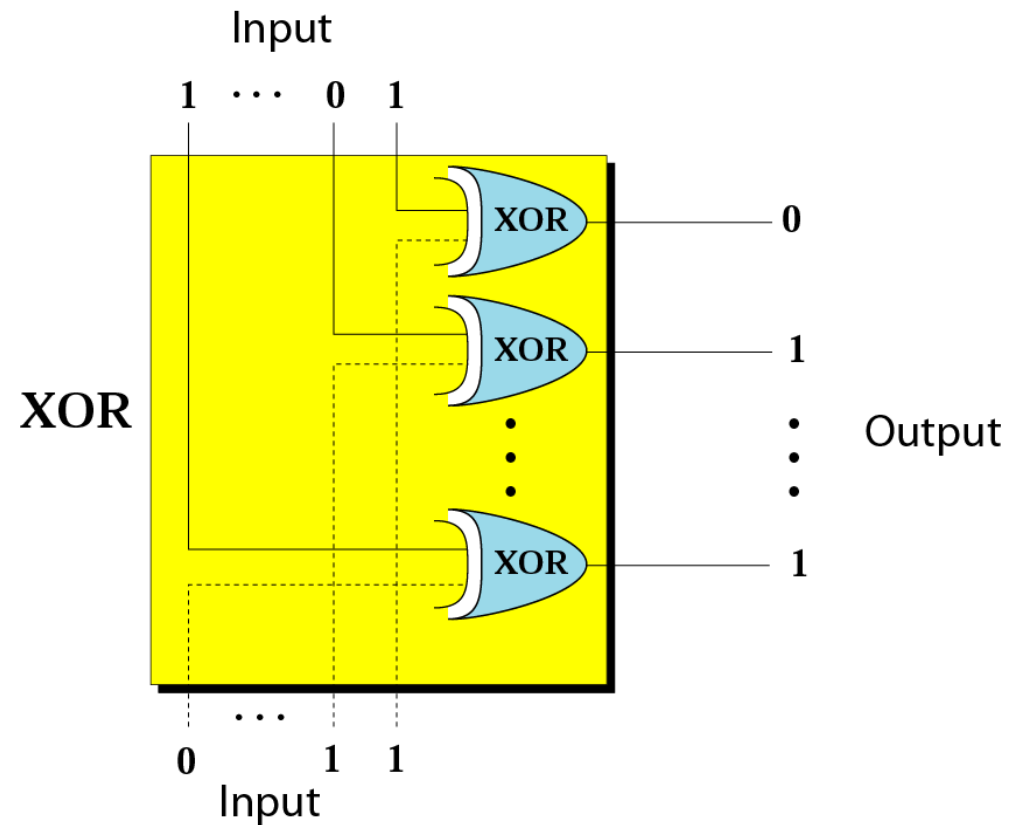


Use the OR operator on the bit patterns 10011001 and 00101110.

Solution

The solution is shown below. Note that only one bit in the output is 0, where both corresponding inputs are 0s.

	1	0	0	1	1	0	0	1	Input 1
OR	0	0	1	0	1	1	1	0	Input 2
	1	0	1	1	1	1	1	1	Output

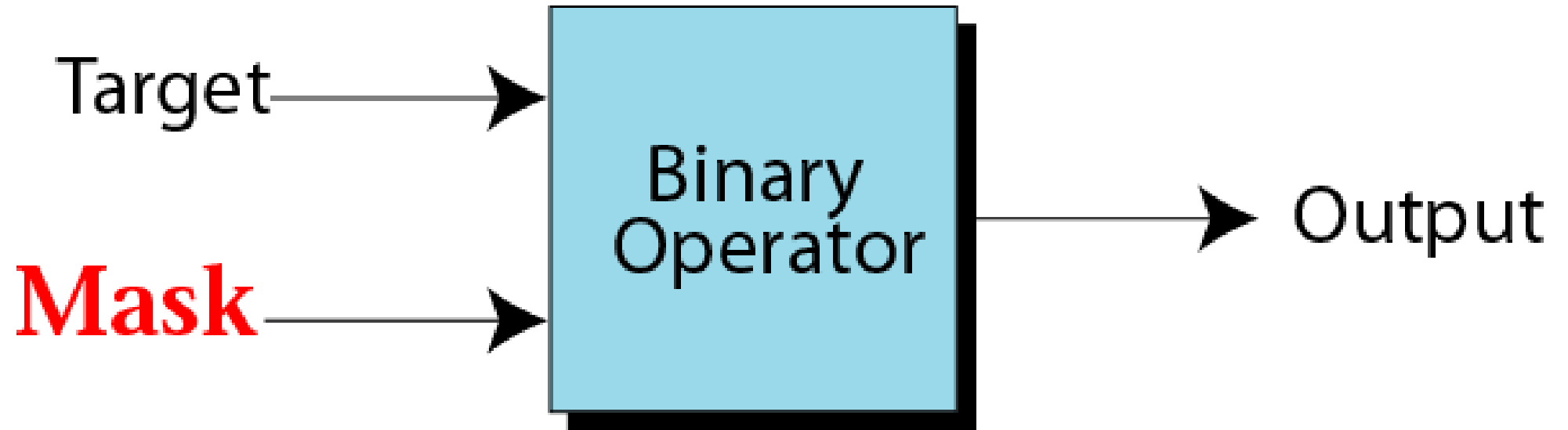


Example

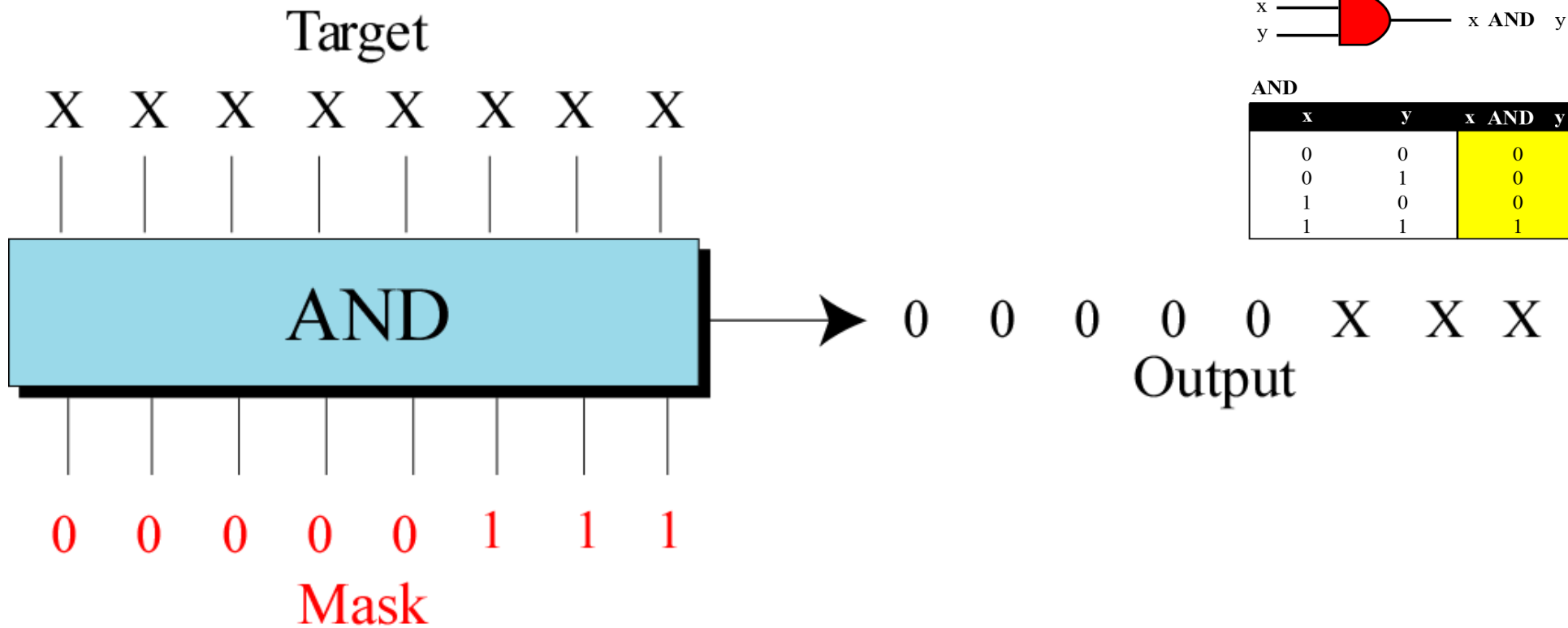
Use the XOR operator on the bit patterns 10011001 and 00101110.

	1	0	0	1	1	0	0	1	Input 1
XOR	0	0	1	0	1	1	1	0	Input 2
	1	0	1	1	0	1	1	1	Output

Name	NOT	AND	OR	XOR																																																			
Symbol																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0
A	X																																																						
0	1																																																						
1	0																																																						
B	A	X																																																					
0	0	0																																																					
0	1	0																																																					
1	0	0																																																					
1	1	1																																																					
B	A	X																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	1																																																					
B	A	X																																																					
0	0	0																																																					
0	1	1																																																					
1	0	1																																																					
1	1	0																																																					



Example of unsetting specific bits



Test the mask with the pattern 10100110.

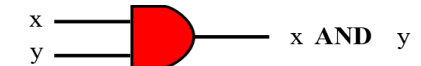
The mask is 00000111. The result of applying the mask is:

	1	0	1	0	0	1	1	0	Input
AND	0	0	0	0	0	1	1	1	Mask
	0	0	0	0	0	1	1	0	Output

Use the mask 10111111 to AND with the target pattern. The only 0 bit (bit 7) in the mask turns off

Target	1 1 0 0 0 1 1 1	AND
Mask	1 0 1 1 1 1 1 1	

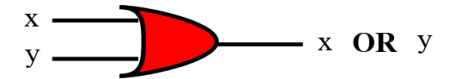
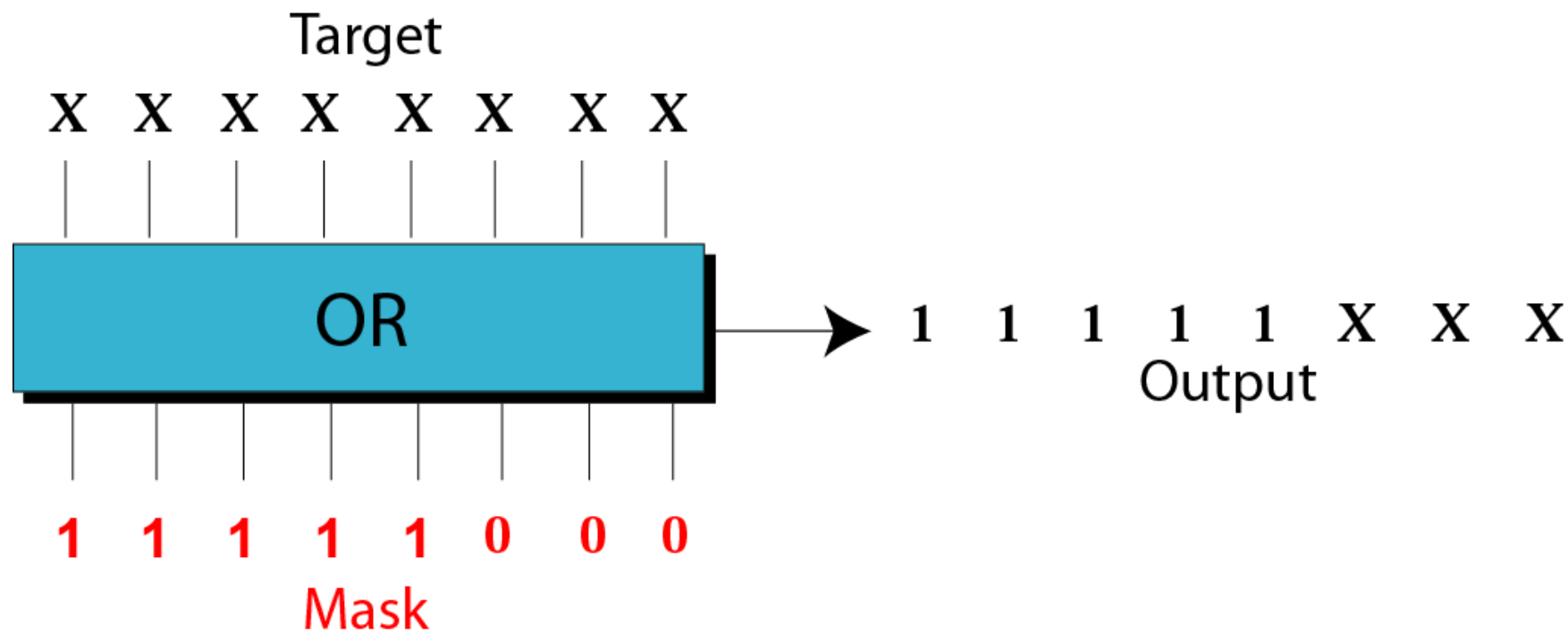
Result	1 0 0 0 0 1 1 1	



AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

Example of unsetting specific bits



OR

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

Use a mask to set the five leftmost bits of a pattern. Test the mask with the pattern 10100110.

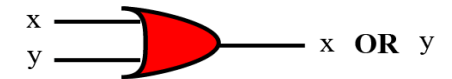
The mask is 11111000. The result of applying the mask is:

	1	0	1	0	0	1	1	0
OR	1	1	1	1	1	0	0	0
	1	1	1	1	1	1	1	0

Input

Mask

Output



OR

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

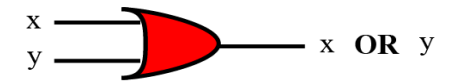
Using the power plant example, how can you use a mask to show that pump 6 is now turned on?

*Use the mask **00100000**.*

Target 1 0 0 0 0 1 1 1 *OR*

Mask 0 0 1 0 0 0 0 0

Result 1 0 1 0 0 1 1 1



OR

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

Shift Operations

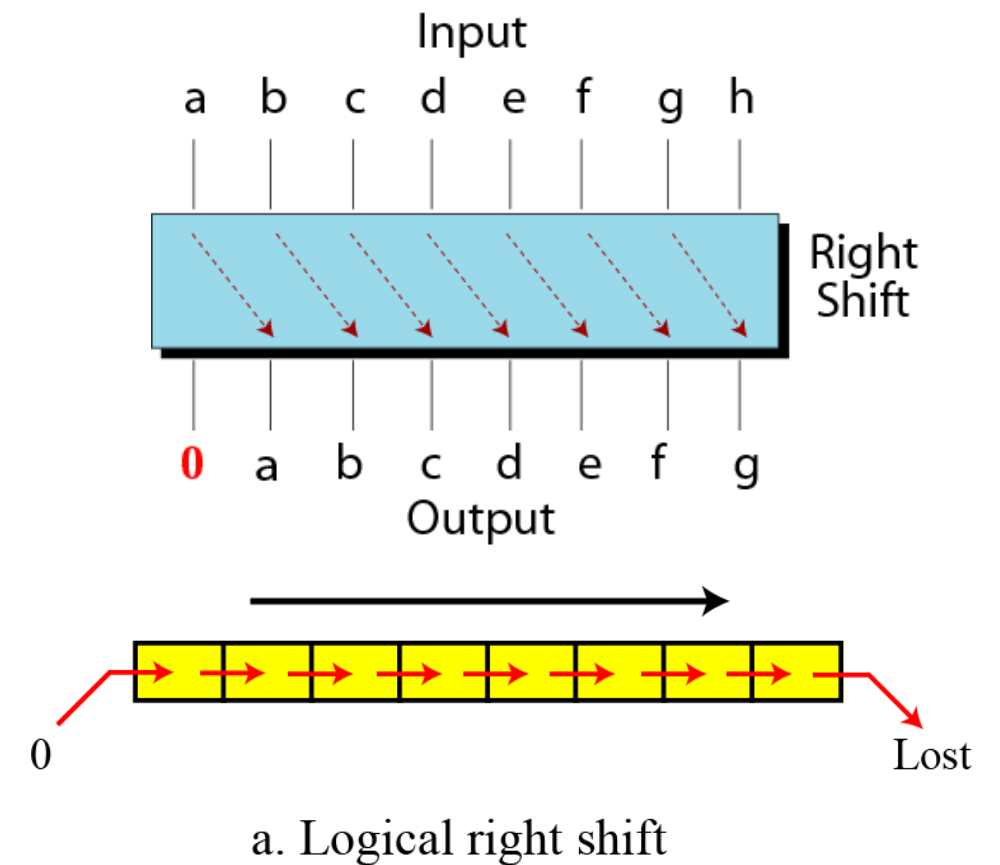
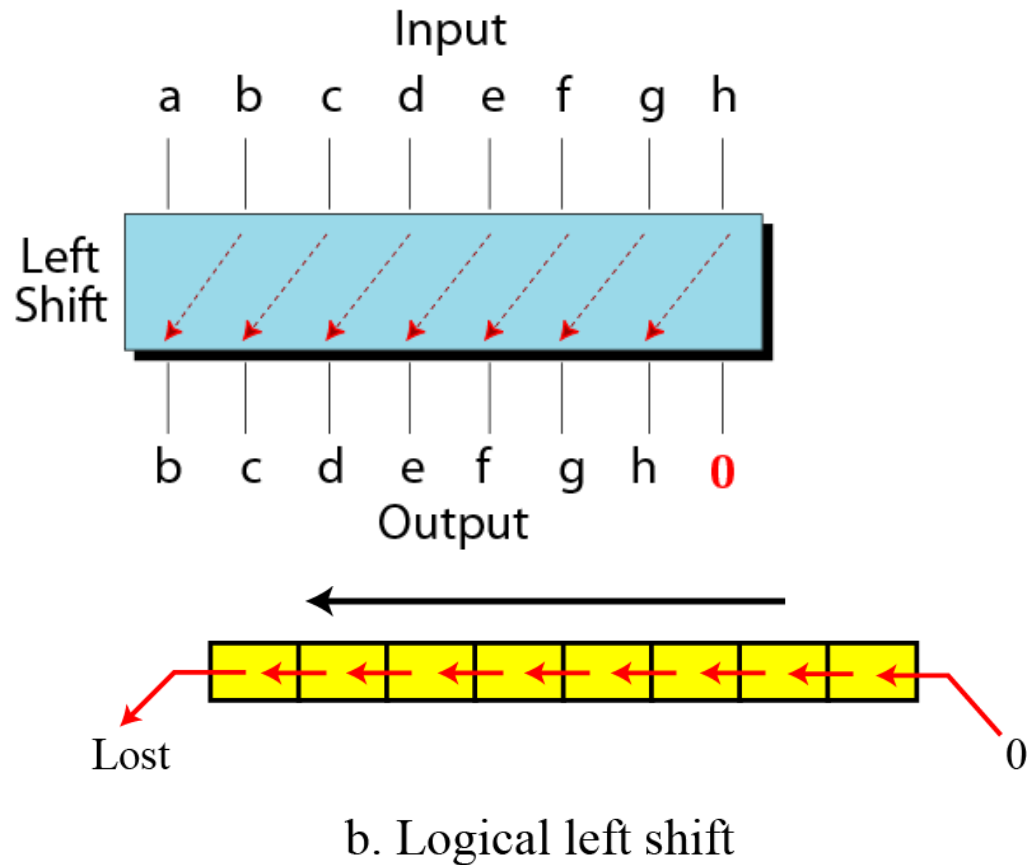


Shift operations move the bits in a pattern, changing the positions of the bits. They can move bits to the left or to the right. We can divide shift operations into two categories:

1. logical shift operations

- ✓ Logical shift
- ✓ Logical circular shift (Rotate)

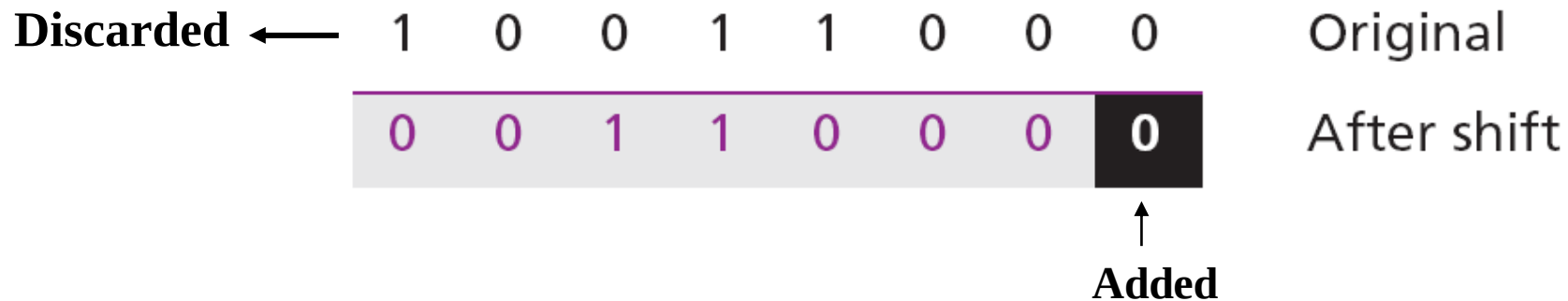
2. Arithmetic shift operations.

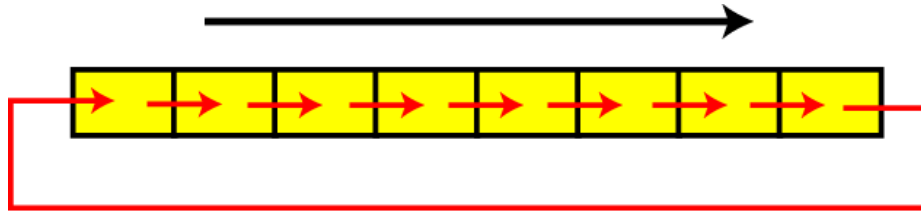


Use a logical left shift operation on the bit pattern 10011000.

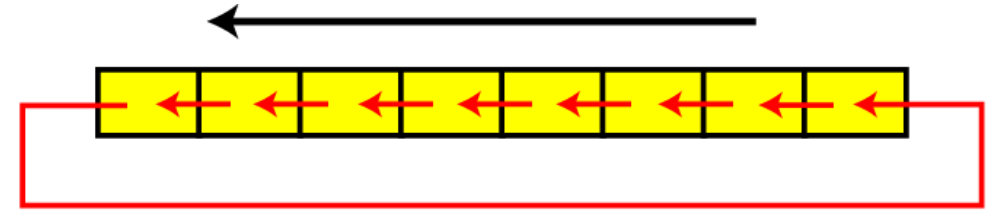
Solution

The solution is shown below. The leftmost bit is lost and a 0 is inserted as the rightmost bit.





a. Circular right shift

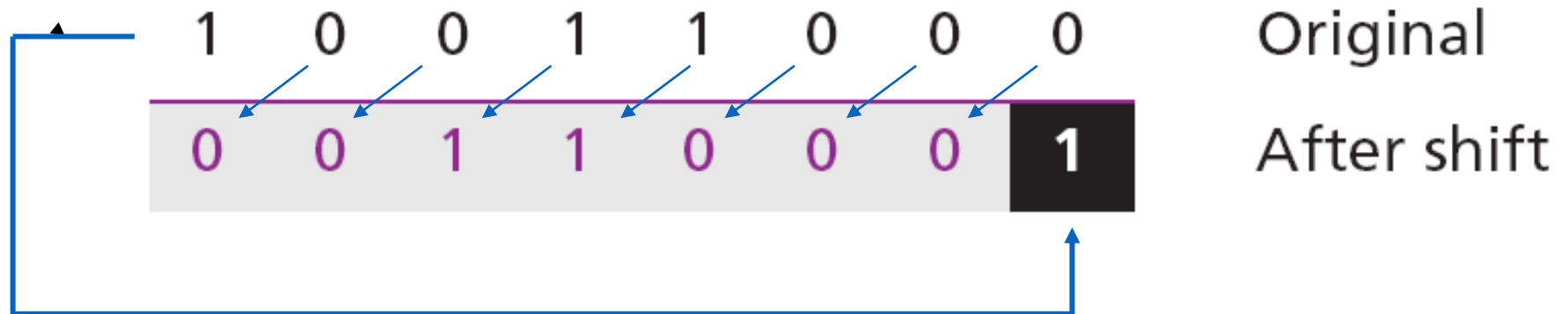


b. Circular left shift

Use a circular left shift operation on the bit pattern 10011000.

Solution

The solution is shown below. The leftmost bit is circulated and becomes the rightmost bit.



Binary operations



Binary Addition

Case	Sum	Carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1
1+1+1	1	1

Add the binary numbers and show the equivalent decimal addition.

111+001

$$\begin{array}{r} 111 \\ + \\ 001 \\ \hline \end{array}$$

1000

Case	Sum	Carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1
1+1+1	1	1

$$\begin{array}{r}
 0111 \\
 00111 \quad 7 \\
 10101 \quad 21 \\
 \hline
 11100 = 28
 \end{array}$$

$$\begin{array}{r}
 1100 \\
 01111 \quad 15 \\
 11100 \quad 28 \\
 \hline
 101011 = 43
 \end{array}$$

Case	Sum	Carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1
1+1+1	1	1

$$1010 + 1010 \\ = \mathbf{010100}$$

$$\begin{array}{r} 1010 \\ + \\ 1010 \\ \hline \end{array}$$

$$= \mathbf{10100}$$

$$1010 + 101 \\ = \mathbf{1111}$$

$$\begin{array}{r} 1010 \\ + \\ 101 \\ \hline \end{array}$$

$$= \mathbf{1111}$$

Case	Sum	Carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1
1+1+1	1	1



Thank you