Unit 1: Logical Operations

Term	Definition
Propositional logic	• A proposition is a simply a statement. • Propositional statements when evaluated will result in either true or false. • Propositional logic considers the way statements interact with each other. • Propositional logic follows mathematical rules.

Logic statements

Most rules to simplify a logic statement are not dependent on the contents of the statement but on the structure of the statement.

Propositional logic uses symbols to represent logical links between propositions. A logic statement includes propositions linked connected by logical links.

Term	Definition
Propositional logic symbols	Propositional logic uses symbols to represent logical links

Symbol	Formal term	Informal term
.	Connection	AND
+	Separation	OR
$\bar{A}$	Negation	NOT
$\oplus$	Exclusive separation	XOR

Term	Definition
Truth table	A truth table is a mathematical table used to analyse a set of local statements.

Connection (AND)

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

A AND B can be written using a symbol as A.B

Separation (OR)

A	B	A AND B
1	1	1
1	0	1
0	1	1
0	0	0

A AND B can be written using a symbol as A + B

Negation (NOT)

A	$\bar{A}$
1	0
0	1

A AND B can be written using a symbol as $\bar{A}$

Exclusive separation (XOR)

A	B	A AND B
1	1	0
1	0	1
0	1	1
0	0	0

A AND B can be written with a symbol as A $\oplus$ B

Bitwise manipulation and masking

Bitwise operations are similar to Boolean operations except they work on individual bits in a byte. A mask or bitmask is data that is used to carry out bitwise operations.

Examples:

Masking bits to 1, using 1111 0000 with an `OR` operation:

1	0	0	1	0	1	0	1
1	1	1	1	0	0	0	0
1	1	1	1	0	1	0	1

This operation masks the four most significant bits leaving the four remaining bits unchanged.

Masking bits to 0, using 0000 1111 with an `AND` operation:

1	0	0	1	0	1	0	1
0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	1

This operation masks the four most significant bits to zero leaving the remaining four bits unchanged.

An `XOR` operation can be used to toggle bits:

1	0	0	1	1	1	0	1
0	0	0	0	1	1	1	1
1	0	0	1	0	0	1	0

1	0	0	1	0	1	0	1
1	1	1	1	1	1	1	1
0	1	1	0	1	0	1	0