# Logical operations

&&  ||

← priority  L-R

→ i = -3, j = 2, k = 0, m

m = ++i && ++j && ++k ;

-2  3  1

m = 0  0 X

i = -2  X -2

j = 2  3

k = 0  1

m = -2 && 3 && 1

→ 1

a = 2 + (3/4);

= 2 + 0

= 2

m = ++i || (++j && ++k);

m = 1  i = -2  j = 2  k = 1

X

= -2 || (  )

↓

1  X

= 1

m = 1, i = -2, j = 2, k = 1

i = 3,

sequence point

K = ++i && i++ ; ←

k = ? 1

i = ? 5

1 && 4 → 1

i = 4

k = i++ > 0 || ++i ;

k = ? 1

i = ? 5

$x = 12, \quad Y = 7$

&&  ||

$Z = (x != Y) \&\& (Y == 2);$

$Z = ?$ 1

1

1

OR

| | | |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

$i = -3, \quad j = 2, \quad k = 0$

$m = (++i \ \&\& \ ++j) \ || \ ++k;$

-2 — 3

1

$m = 1 \quad i = -2 \quad j = 3 \quad k = 0$

$i = -1, \quad j = 2, \quad k = 0$

$m = (++i \ \&\& \ ++j) \ || \ ++k;$

0

0

$m = 1 \quad i = 0 \quad j = 2 \quad k = 1$

## Not logical operator (!)

$!a \longrightarrow$

toggle

$a = 5;$

$!a \rightarrow !(5) \rightarrow 0$

$!a \rightarrow !(0) \rightarrow 1$

# Assignment operator

identifier = expression

variable

Assignment operation

collection of constants, variables and operators.

a : 3

b : c+d

If the left hand side data type is different then the right hand side then the value of the expression will be converted to the type of the identifier on the left hand side.

```
int a;
a = 2+3.2;
```

float

float = 2+3;

5

5.000000

5.2

int float

id1 = id2 = expr;

float

L R

5.4

5

5.0

## Additional

-128 to 127

→ += -= *= /= %=

a = b;

a += b;  a -= b;  a *= b;

↳ a = a+b  a = a-b  a = a*b;

# BITWISE OPERATOR

| & | \| | ^ | << | >> | ~ |
|---|---|---|---|---|---|
| (AND) | (OR) | (XOR) | (L shift) | (R shift) | Invert |

## & (AND)

```
int a = 11;
int b = 19;
```

a & b → 3

→ 001011

→ 010011

(000011)$_2$ → 3

AND

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Bitwise operations

## & (AND)

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

A → 11

B → 19

&

$(001011)_2 \rightarrow (11)_{10}$

$(010011)_2 \rightarrow (19)_{10}$

$(000011)_2 \rightarrow (3)_{10}$

electronic

IOT - related

number even/odd

17

10001

& 00001

Non-zero

odd

## 17 & 1

16 → 10000

1 → 00001

$(00000)_2$

$(0)_{10}$ → even

odd

10001 → $(17)_{10}$

0001 → $(1)_{10}$

$(00001)_2$

Non zero

$(1)_{10}$

## | (OR)

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## ^ (XOR)

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## ~ (Invert)

| | |
|---|---|
| 0 | 1 |
| 1 | 0 |

~a

a = 5

(000101)

111010

Unsigned

000001

111110

# Left shift & Right shift

Multiplication division

A = 11

<< >>

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 1 |

A << 2;

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 |

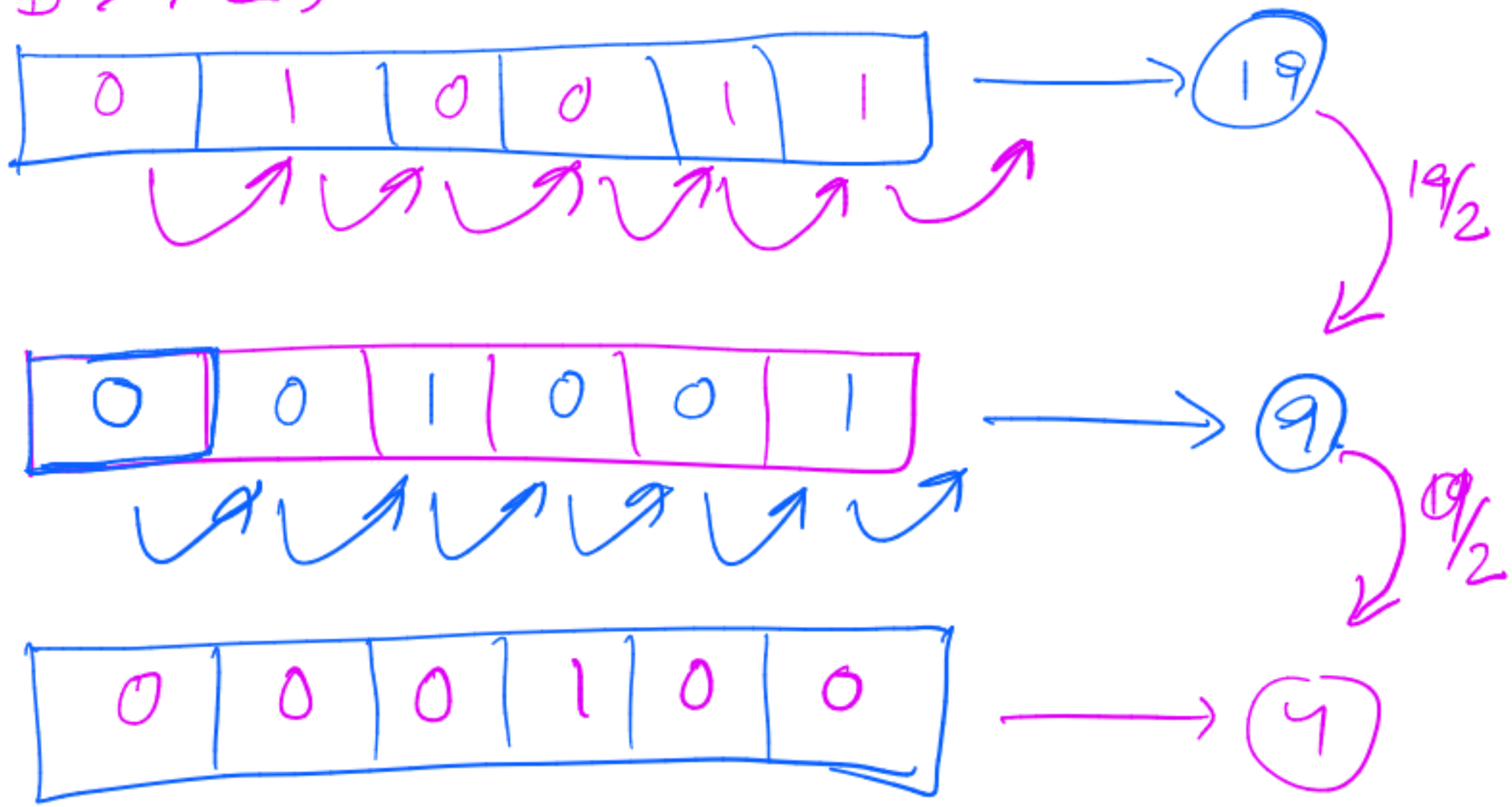
| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 32 | | 8 | 4 | | |

11 → ×2 → 22 → ×2 → 44 → 5 ×2 → 1

11×4

B = 19

B >> 2;



$B = -8$

$-8 \rightarrow$ 6 bits

2's complement

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |

$\rightarrow -8$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 |

$\rightarrow -4$

$\rightarrow 28$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 |

$\rightarrow -2$

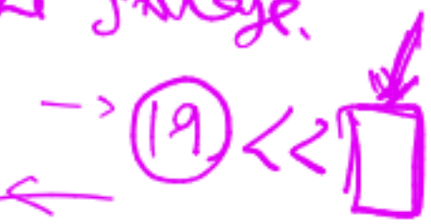
| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |

$\rightarrow -1$

$-1/2 \rightarrow$

## Rules to remember

→ Lshift and Rshift are undefined for -ve numbers and more than the size of the storage.




a<<2; ←

a<<-2; ✗

undefined

→ Bitwise operators should not be used in place of logical operator.

→ Lshift & Rshift are equivalent to multiplication and division by 2, respectively.

→ & is used to check whether the number is even / odd.

## Chapter-21 (363)

### Operating on bits