Operators

Math Operators +, −, *, /, %

% ← Modulus → Returns remainder of Division.

ex. 10 % 6 = 4

All can be hit by overflow

→ Operations occur usually in 32-bit integer mode

→ Data loss (overflow) occurs on storage into smaller types.

more significant bytes thrown away.

Ex. uint8_t b = #; # can be larger than 255 (max value)

→ b = # % 256

Dividing issues

uint16_t a, b, c; a = #; b = #;

c = a/b;

what if a = 7, b = 3

c = ? → 2

what is c = (a/100) * b;

=0

c = 0

solutions

c = (a*b)/100;

c = (a/100.0)*b;

float → use floating point math.

Bitwise Logic Operators

& , | , ^ , ~

- & : and
- | : or
- ^ : xor — NOT EXPONENT
- ~ : not, inverse.

A	B	A&B	A|B	A^B	~A
f	f	f	f	f	T
f	T	f	T	T	T
T	f	f	T	T	f
T	T	T	T	f	f

for bits: f = 0
T = 1

```
uint8_t a, b, c;
a = 0x5A; → 01011010
b = 0x78; → 01111000
```

c = a & b;

```
  a: 01011010
& b: 01111000
-------------
     01011000
```

c → 0x58

```
  a: 01011010
| b: 01111000
-------------
     01111010
```

c = a | b

c → 0x7A

c = ~a; c → 10100101 = 0xA5

Bitwise Shift Op.

<<, >>

left shift — right shift

a = 0x5A;

c = a << 3;

01011010

01011010000 ← fill with 0's

save into c, keep least 8 bits.

11010000

c => 0xD0.

---

c = a >> 3;

01011010

00001011~~010~~

c → 0x0B

Logical operators, value operators

==, !=, <, >, <=, >=, ← operate on full value

&&, ||, ! ← operate on True or false "value"

&& = and, || = or, ! = not, inverse.

value is "True" if # is ≠ 0

"false" if # is = 0

1 && 100.1 → T && T = T

0.0001 && 2 → T && T = T

0 && -2 → F && T = F

All of these output either:

True = 0x01 = 0b00000001 = 1

false = 0x00

Ex. !! 365 = 0x01 = !(!365) = !0 = 0x01