Bit Operator

- Leading zeros `0` (i.e., to the left) are meaningless

~ NOT

- Invert a bit
- Examples:
  - `~0 = 1`
  - `~1 = 0`
  - `~0111 = 1000`
  - `~100 = 011`
- In Java, `~` inverts an `int` and not single bits, so:

```
int b = 0b10;
~b == 11111111111111111111111111111101
```

& AND

- Results in `1` in each position if the corresponding first bit ***and*** second bit are `1`, otherwise `0`.
- Enables to find if a certain bit in a number contains `1` or `0`. Can be considered like multiplying all bits.
- Examples:
  - `10 & 11 = 10`
  - `0011 & 0010 = 0010`

| OR

- Results in `1` in each position if the corresponding first bit ***or*** second bit are `1`, otherwise `0`.
- Enables to set a specific bit to `1`.
- Examples:
  - `10 | 11 = 11`
  - `0011 | 0010 = 0011`

^ XOR

- Exclusive OR – results in `1` in each position if the corresponding first bit ***or*** second bit are `1`, but ***not*** both, otherwise `0`.
- Enables to compare two bits – `1` means they are different, `0` means they are the same.
- Can be used to invert selected bits in a register. Any bit can be toggled by XOR-ing it with `1`.
- XOR-ing a value against itself yields zero.
- Examples:
  - `0101 ^ 0011 = 0110`
  - `0010 ^ 1010 = 1000`
- XOR can be used for "backup":
  - Calculate `a` and `b`'s XOR: `x = a^b`

    - If needed, recover `a`: `a = b^x`
    - If needed, recover `b`: `b = a^x`

<< Shift Left

- Add `n 0` bits to the right.
- A left arithmetic shift by `n` is equivalent to multiplying the number by `2^n`.
- For example: `10111 << 1 = 101110`

>> Shift Right

- Remove `n` bits from the right (`0` or `1`).
- A right arithmetic shift by `n` is equivalent to dividing by `2^n`.
- For example: `10010111 >> 1 = 1001011`

Binary Numbers

```
0 = 0
1 = 1
10 = 2
11 = 3
100 = 4
101 = 5
110 = 6
111 = 7
1000 = 8
1001 = 9
1010 = 10
1011 = 11
1100 = 12
1101 = 13
1110 = 14
1111 = 15
10000 = 16
...
```

```
10010110
^      ^
|      |------- bit 0
|
|-------------- bit 7
```

Having a `1` in the `k`-th bit, means that the decimal number is comprised of `2^k`. For example, for the above number:

```
2^7 + 2^4 + 2^2 + 2^1 = 150
```