Bitwise Logic

Last updated 6/16/23

These slides introduce bitwise logic concepts used in programming

Bitwise Logic

- Terminology
  - Consider an 8 bit value

    abcd efgh where the values are unknown to us, but are either 0 or 1

    e.g. abcd efgh where a,d,f,g are 1, the others are 0 → 1001 0110

  - Bitwise
    - Match bits between two values and perform the desired operation bit by bit – resulting in a new binary number

      abcd efgh bitwise-AND ijkl mnop → (a and i) (b and j) (c and k) …

      1011 1010 bitwise-AND 1100 1001 →

1	0	1	1	1	0	1	0
1	1	0	0	1	0	0	1
and'd	1	0	0	0	1	0	0	0

    - Bitwise operators: AND, OR, NOT, XOR

Bitwise Logic

- Bitwise Logic in C
  - Logic Expression
    - Operation Operand(bits) → new binary word
    - Operand(bits) Operation Operand(bits) → new binary word
  - Operations
    - NOT – flips the evaluation of the operand bits
    - OR – evaluates as True if either operand bit is true (including both)
    - AND – evaluates as True if both operands bits are true
    - XOR – evaluates as True if only 1 operands bit is true (but not both)

Bitwise Logic

- Bitwise NOT – flips the evaluation of the operand's bits
  - ~ operand

A = 0000 1001
B = 1111 1101

~ A → 1111 0110
~ B → 0000 0010

Bitwise Logic

- Bitwise OR – evaluates to T if either operand's bit is T
  - operand | operand

A = 0000 1001
B = 1111 1101
C = 0100 1010

A | C →

```
  0000 1001
OR 0100 1010
   0100 1011
```

A | (~B) →

```
  0000 1001
OR 0000 0010
   0000 1011
```

Bitwise Logic

- Bitwise AND – evaluates to T if both operand's bits are T
  - operand & operand

A = 0000 1001
B = 1111 1101
C = 0100 1010

A & C →

```
  0000 1001
AND 0100 1010
    0000 1000
```

B & C →

```
  1111 1101
AND 0100 1010
    0100 1000
```

Bitwise Logic

- Bitwise XOR – evaluates to T if only one operand's bit is T
  - operand ^ operand

A = 0000 1001
B = 1111 1101
C = 0100 1010

A ^ C →

```
  0000 1001
XOR 0100 1010
-----------
  0100 0011
```

B ^ C →

```
  1111 1101
XOR 0100 1010
-----------
  1011 0111
```