Bit Manipulation

Common - Last updated 2/23/19

Bit Manipulation

- Terminology
  - Consider an 8 bit value

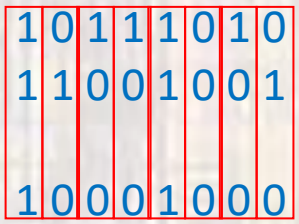
    abcd efgh where the values are unknown to us, but are either 0 or 1

    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

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

      1011 1010 AND 1100 1001 →

    - AND, OR, NOT, XOR, …

Bit Manipulation

- Bit testing
  - How can we determine the value of just 1 bit out of the 8?
    - If we want to know the value of bit 3 (e) we can bitwise-AND the value with another 8 bit value with just bit 3 set to 1

(and)

abcd efgh & 0000 1000 → 0000 e000

      - If e is 1 then the result will be 8
      - If e is 0 then the result will be 0

    - We can test the result to determine what value e has
      - Result = 0 → e must be 0
      - Result = 8 → e must be 1
      - Result > 0 → e must be 1
      - Result < 1 → e must be 0
      - Result = TRUE → e must be 1
      - Result = FALSE → e must be 0

Bit Manipulation

- Bit setting
  - How can we set the value of a bit to 1 (set)?
    - We can bitwise-OR the value with another 8 bit value with just the desired bit(s) set to 1

(or)
abcd efgh | 0000 1000 → abcd 1fgh

(or)
abcd efgh | 0101 1000 → a1c1 1fgh

Bit Manipulation

- Bit clearing
  - How can we set the value of a bit to 0 (clear)?
    - We can bitwise-AND the value with another 8 bit value with just the desired bit(s) set to 0, all others set to 1

abcd efgh (and) & 1111 0111 → abcd 0fgh

abcd efgh (and) & 1010 0111 → a0c0 0fgh

    - If we prefer to indicate the bits to clear with a 1 we can:

abcd efgh (and) & (not) ~(0000 1000) → abcd 0fgh

⇩

abcd efgh & 1111 0111 → abcd 0fgh

abcd efgh (and) & (not) ~(0101 1000) → a0c0 0fgh

⇩

abcd efgh & 1010 0111 → a0c0 0fgh