CSC180: Lecture 35

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Low-level Programming

Usage

- Operating Systems
- Embedded Systems
- Binary Content
  - Images, videos, audio
- Data Compression

Binary representation of data

- All data in a the computer's memory is represented as a sequence of bits

- **Bit** : unit of storage, represents the level of an electrical charge.

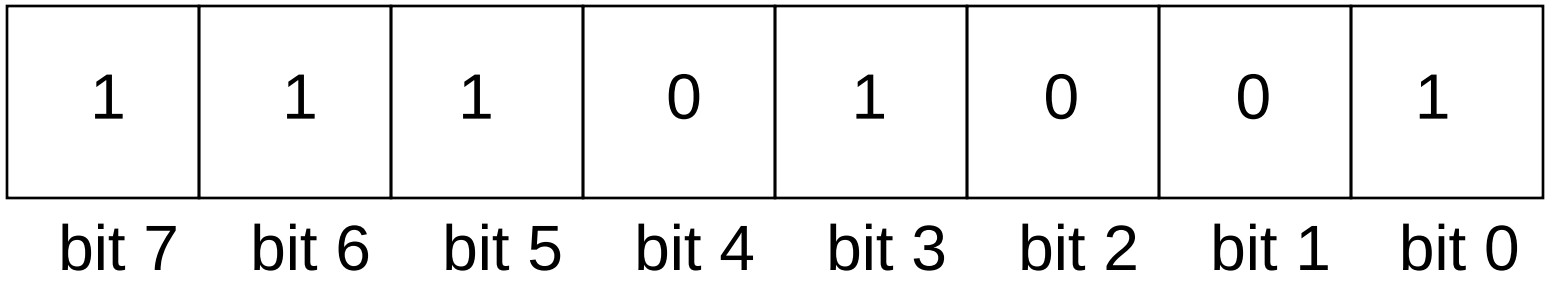
  Can be either 0 or 1.

  0 no voltage

  1 voltage (really ~1.5v)

Binary representation of data

A single byte of memory



Binary representation of data

0 ➔ 0 0 0 0 0 0 0 0 //8-bit binary representation of positive integers

1 ➔ 0 0 0 0 0 0 0 1

2 ➔ 0 0 0 0 0 0 1 0

3 ➔ 0 0 0 0 0 0 1 1

...

255 ➔ 1 1 1 1 1 1 1 1 (Hence, the largest number we can store in a byte is 255)

.....

10505 ➔ 1010 010 000 1001

……

65535 ➔ 1111111111111111 (Hence, the largest number we can store in 2 adjacent bytes is 65,535)

Representation of Data – base2 to base10

- Representation: a number A = $\sum_{i=0}^{n-1} d_i \beta^i$

1	1	1	0	1	0	0	1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

In decimal representation, this number is:

$1*2^0 + 0*2^1 + 0*2^2 + 1*2^3 + 0*2^4 + 1*2^5 + 1*2^6 + 1*2^7$
$= 233$

Bit operators: AND , OR , NOT

AND

A	B	C = A & B
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	C = A | B
0	0	0
0	1	1
1	0	1
1	1	1

NOT

A	C = $\overline{A}$
0	1
1	0

Bit Operators in C

Operator	Meaning
~	complement (not)
&	And
|	Or
^	Exclusive or
<<	Left shift
>>	Right shift

Examples

Expression	Result
a = 21;	00000000 00000000 00000000 00010101
b = 33;	00000000 00000000 00000000 00100001
~a	11111111 11111111 11111111 11101010
a&b	00000000 00000000 00000000 00000001
a|b	00000000 00000000 00000000 00110101
a^b	00000000 00000000 00000000 00110100

Example

Expression	Result
c = 67;	00000000 00000000 00000000 01000011
c<<5	00000000 00000000 00001000 01100000

Expression	Result
c = 67;	00000000 00000000 00000000 01000011
c>>3	00000000 00000000 00000000 00001000

Setting a bit to one (e.g. 2nd index from right)

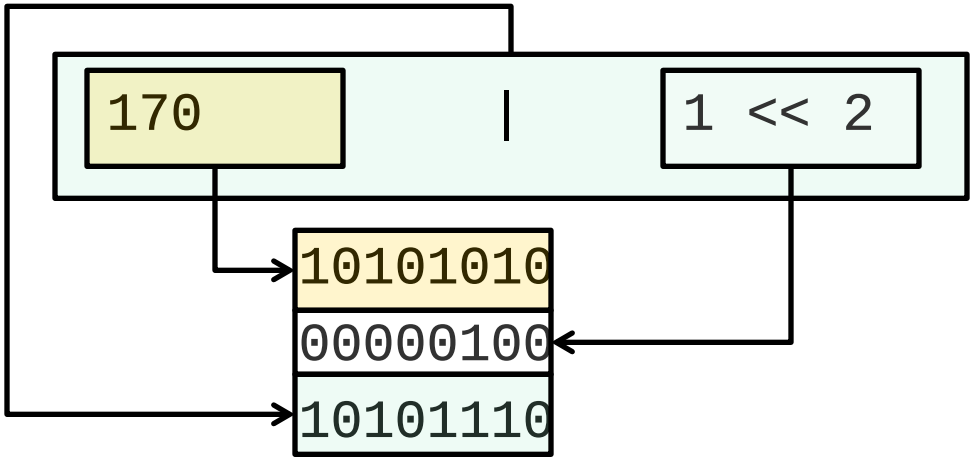
```
int X;
int value;

X = 1;                  // 0000000000000001
X = X << 2;             // 0000000000000100

value = 170;            //0000000010101010
value = value | x ;     //0000000010101110
```

Setting a bit to one (e.g. 2nd index from right)

```
value =     value | 1 << bit_number;
```



Setting a bit to zero (e.g. 3rd index from right)

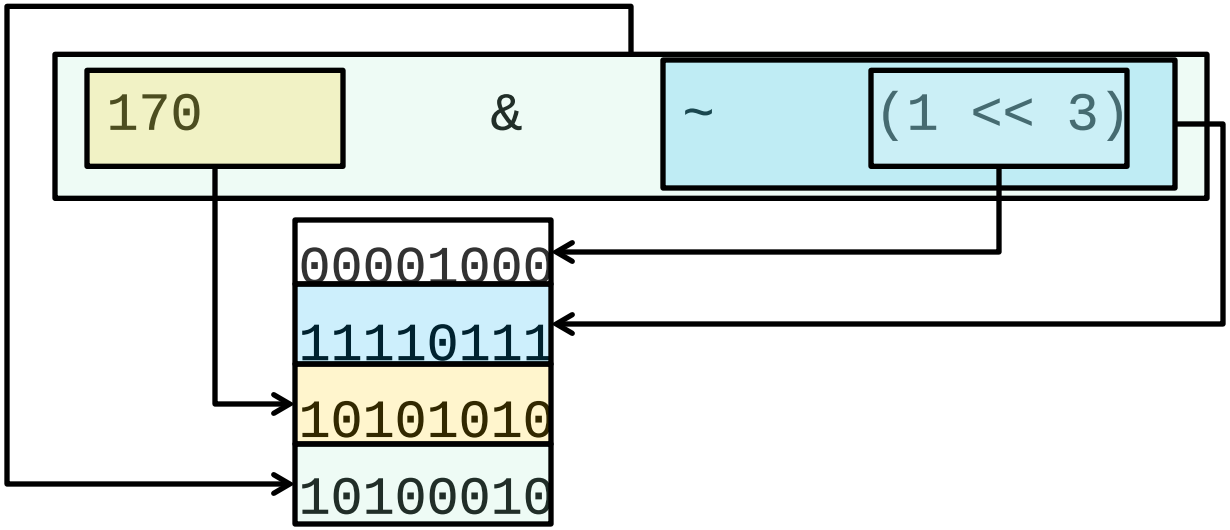
```
int X;
int value;

X = 1;                  // 0000000000000001
X = X << 3;             // 0000000000001000
X = ~X;                 // 1111111111110111

value = 170;            //0000000010101010
value = value & x ;     //0000000010100010
```

Setting a bit to zero (e.g. 3rd index from right)

```
value =     value & ~ ( 1 << bit_number );
```



Data Compression – pack

- Example:

```
int pack4char (char char1, char char2,
                        char char3, char char4)
{
        int a=char1;
        a =  (a<<8)|char2;
        a  = (a<<8)|char3;
        a  = (a<<8)|char4;
        return a;
   }
```