Numbers Representation

Operating Systems I

Victor Yacovlev (Viktor Iakovlev)

The Binary System

01010101

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

0 - Voltage is less then 0-thereshold (usually 0.7V for most chips)
1 - Voltage is greater then 1-thereshold (usually 2.0V for most chips)
undefined - Voltage between theresholds

Integer Types

- The Byte – is the smallest data amount able to address
- The Bit – 0 or 1. Individual bits are not accessible directly from memory
- The Word – several Bytes processor accesses at once

Bits Manipulation

- Shift Left, Right: << >>
- Bitwise operations: &, |, ^, ~
- Use & mask to extract bit at specific positions
- Use | operations to add some bits into ‘set’

Byte Order

`00000000 11111111`

- 255 in case of red byte is low order
- 65’280 in case of blue byte is low order

Which byte order is correct?

Both

Byte Order

- From High to Low (like people writes) – Big Endian
  - default mode for PowerPC, MIPS, SPARC
  - standard for network data exchange
- From Low to High (like arabic letters) – Little Endian
  - all x86 processors family and default mode for ARM

Signed v.s. UnSigned

Unsigned	Binary	Signed
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	-0
9	1001	-1
10	1010	-2
11	1011	-3
12	1100	-4
13	1101	-5
14	1110	-6
15	1111	-7

Signed v.s. UnSigned

Unsigned	Binary	Signed	Two's Complement
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	-0	-8
9	1001	-1	-7
10	1010	-2	-6
11	1011	-3	-5
12	1100	-4	-4
13	1101	-5	-3
14	1110	-6	-2
15	1111	-7	-1

Overflow

$$X + Y = Z + C$$

When X, Y, Z are fixed-size numbers

$C > 0$ – means **overflow**

$C=0 \leftrightarrow Z>=X$ and $Z>=Y$

But… only when $X>=0$ and $Y>=0$

Overflow

- There is no standard function in C and C++ to check overflows
- But most processors support overflow detection
- Use GCC builtins (non standard functions)

The Real Values

- Fixed-Point
  - several meaningful bytes for fractional part
  - often used for ‘Currency’ type in applications
- Floating-Point
  - arbitary fractional part
  - less precise
  - more complex to implement

The Real Types

- Float – 32 bit, values about +- 10^{37}
- Double – 64 bit, values about +- 10^{307}

IEEE 754 Representation

Single-Precision (type **float**); B = 127		
S (1 bit)	E (8 bits)	M (23 bits)

Double-Precision (type **double**); 1023		
S (1 bit)	E (11 bits)	M (52 bits)

$$\text{Value} = (-1)^{S} \cdot 2^{E-B} \cdot (1 + M / (2^{23 or 52} - 1))$$

Special Values

- Zeroes: M=0, E=0, S – any
- Infinity: M=0, E=maximum, S – any
- Not-a-Number: M<>0, E=maximum, S – any
- Denormalized (not all processors support):

  E=0, M – any, S – any

Denormalized Values

Single-Precision (type **float**); B = 127		
S (1 bit)	E (8 bits)	M (23 bits)

Double-Precision (type **double**); 1023		
S (1 bit)	E (11 bits)	M (52 bits)

Regular Values (when E>0):

$$\text{Value} = (-1)^{S} \cdot 2^{E-B} \cdot (1 + M / (2^{23\text{or}52} - 1))$$

Denormalized Values (when E=0):

$$\text{Value} = (-1)^{S} \cdot M / (2^{23\text{or}52} - 1)$$