High Performance Computing

Lecture 2

Matthew Jacob

Indian Institute of Science

How is Data Represented?

- Character data: ASCII code
- Integer data

Integer Data

- Whole numbers, i.e., numbers without fractional part
- In computer systems, you usually find support for both “signed integers” and “unsigned integers”
  - e.g., C programming

    int x;   Can take +ve or -ve whole number values

    unsigned int y;   Can take on +ve whole number values

Representing Signed Integer Data

- Sign-magnitude representation

Sign bit

least significant bit (lsb)

$$x_{n-1}x_{n-2}\cdots x_2 x_1 x_o$$

represents the value

$$(-1)^{x_{n-1}} \times \sum_{i=0}^{n-2} x_i 2^i$$

Example: In 8 bits

13 is represented as 00001101

-13 is represented as 10001101

Alternative: 2s Complement Representation

The *n* bit quantity

least significant bit

$$x_{n-1}x_{n-2}\cdots x_0$$

represents the signed integer value

$$-x_{n-1}2^{n-1}+\sum_{i=0}^{n-2}x_i 2^i$$

Example: In 8 bits

-128 + 64 + 32 + 16 + 2 + 1

13 is represented as 00001101

-13 is represented as 11110011

Example: Signed integer

16bit 2s complement value 0xED7E

Tells you that the binary value is being shown in Hexadecimal notation

Hexadecimal: Base 16

1110110101111110

1110 1101 0111 1110   from `right', groups of 4 bits

The base 16 digits are 0..9,A,B,C,D,E,F

E   D   7   E

Which Representation is Better?

- Considerations
  - Speed of arithmetic (addition, multiplication)
  - Speed of comparison
  - Range of values that can be represented
- The 2s complement representation is widely used

How is Data Represented?

- Character data: ASCII code
- Signed Integer data: 2s complement
- Real data

Real data

- Real numbers: points on the infinitely long real number line
  - There are an infinitely many points between any two points on the real number line

Real Data: Floating Point Representation

IEEE Floating Point Standard (IEEE 754)

32 bit value with 3 components (*s*, *e*, *f*)

1. *s* (1 bit sign)
2. *e* (8 bit exponent)
3. *f* (23 bit fraction)

represents the value

$$(-1)^s \times 1.f \times 2^{e-127}$$

Example: IEEE Single Float

Consider the decimal value 0.5

- Equal to 0.1 in binary 1.0×2^{-1}

  $(-1)^{s} \times 1.f \times 2^{e-127}$
- s: 0, e: 126, f: 000…000
- In 32 bits,

  0 01111110 0000000000000000000000

Example: IEEE Single Float.

32bit IEEE single float 0xBDCCCCCC

1011 1101 1100 1100 1100 1100 1100 1100

1 01111011 100 1100 1100 1100 1100 1100

Sign bit: 1 Negative value

Exponent field: 123 Exponent value: 123 -127 = -4

- 1.100 1100 1100 1100 1100 1100 x 2^{-4}

$$2^{-3} \times 0.110011001100110011001100$$

Answer: -0.1 decimal

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

More on IEEE Floating Point

- Why is the exponent represented in this way? (excess-127 representation for signed integers)
- Normalized representation
- Special forms

Denormalized values	(exp = 0; f = non-zero)
Zero	(exp = 0; f = 0)
Infinity	(exp = 255; f = 0)
NaN	(exp = 255; f = non-zero)

How is Data Represented?

- Character data: ASCII code
- Signed Integer data: 2s complement
- Real data: IEEE floating point