מספרים רציונליים
Rational Number Representation

ייצוג

ארגון המחשב ושפת סף

מרצה: ***רועי אש***

אפקה
המכללה האקדמית
להנדסה בתל אביב

Fixed Point vs. Floating Point

- We've already seen two ways to represent an integer in computer hardware:
  - signed
  - unsigned
- Both ways were with a ***fixed point*** representation - the location of the binary point was fixed:

0	1	0	1	1	1	0	1

אפקה
המכללה האקדמית להנדסה בתל אביב

Floating Point

- Going back to decimal…
- Sometimes it is more comfortable to represent a value using a floating point.
  - For instance: $1{,}200{,}000{,}000 = 1.2 \cdot 10^{9}$
    - 1.2E9

Exponent

Mantissa

Base

- Generally, we use the form $d_0.d_1d_2\ldots d_{p-1} \cdot B^{e} \quad (d_0 \neq 0)$

$$d_0.d_1d_2\ldots d_{p-1} \cdot B^{e} = \left(d_0 + \sum_{j=1}^{p-1} d_j B^{(-j)} \right) \cdot B^{e}$$

Floating Point

- If we look at 123:
  - $$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

$$10^2 \quad 10^1 \quad 10^0$$

- The same goes for (fixed point) 123.456:
  - $$123.456 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2} + 6 \cdot 10^{-3}$$

$$10^2 \quad 10^1 \quad 10^0 \quad 10^{-1} \quad 10^{-2} \quad 10^{-3}$$

Floating Point

- Using the form: $d_0.d_1d_2...d_{p-1} \cdot B^e$
  - $123 = 1\cdot10^2 + 2\cdot10^1 + 3\cdot10^0$

    $10^2 \cdot (1\cdot10^0 + 2\cdot10^{-1} + 3\cdot10^{-2}) =$

    $1.23\cdot10^2$ = 1.23E2

  - $123.456 = 1\cdot10^2 + 2\cdot10^1 + 3\cdot10^0 + 4\cdot10^{-1} + 5\cdot10^{-2} + 6\cdot10^{-3}$

    $= 10^2 \cdot (1\cdot10^0 + 2\cdot10^{-1} + 3\cdot10^{-2} + 4\cdot10^{-3} + 5\cdot10^{-4} + 6\cdot10^{-5})$

    $= 1.23456\cdot10^2$ =1.23456E2

Binary Rational

- Converting from any radix to decimal is done as before.
  - So, for instance, the binary number 100.101:

$$100.101 = 1\cdot 2^{2} + 0\cdot 2^{1} + 0\cdot 2^{0} + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3}$$

$$= 1\cdot 4 + 0\cdot 2 + 0\cdot 1 + 1\cdot 0.5 + 0\cdot 0.25 + 1\cdot 0.125$$

$$= 4 + 0.5 + 0.125 = 4.625$$

IEEE 754 Floating Point Standard

IEEE 754 standard, released in 1985 after many years of development

- Floating Point numbers approximate values that we want to use.
- IEEE 754 Floating Point Standard is most widely accepted attempt to standardize interpretation of such numbers
  - Every desktop or server computer sold since ~1997 follows these conventions

IEEE - ***I***nstitute of ***E***lectrical and ***E***lectronics ***E***ngineers

Float vs. Double

- So how many digits do we really have?
- Depends on the representation. We have two possible representations for floating point values:

  4-byte **float** and 8-byte **double**.
- It all depends on the amount of **accuracy** we need.

Hidden Bit

- In IEEE 754, we use the form:

$$d_0 \bullet d_1 d_2 \ldots d_{p-1} * B^e$$

- Where: $B = 2$, $d_i \in \{0,1\}$
- In decimal, every digit would have values in the range 0..9 besides d_0 which have values in range 1..9.
- Likewise, in binary, d_0 could only have the value of 1.
- So why should we save it?
- Since we won't save it, we'll refer to it as the "hidden bit"

Float

31	30 … 23	22 … 0
sign	exponent+127	mantissa
1 bit	8 bits	23 bits

- 32-bit (4-byte) representation.
- 1 bit for sign: 1 for negative, 0 for positive.
- 23 bits for mantissa.
- 8 bits for the exponent.
  Important: The **true value** of a exponent is
  unsigned exponent representation - 127.

Float Limitations

31	30 ... 23	22 ... 0
sign	exponent+127	mantissa
1 bit	8 bits	23 bits

- 0 is represented with mantissa=0 and “computer” exponent=0.
- Max absolute value (all 1’s in mantissa and 11111110 exponent):

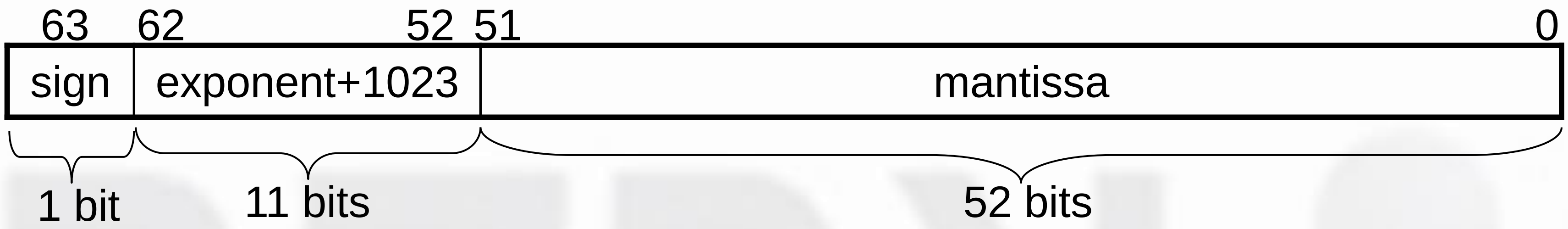
  $$1.1111....1 * 2^{127} = 2^{128} - \varepsilon$$

- Min absolute value (0 in mantissa and 1 as “computer” exponent):

  $$1.0000....0 * 2^{-127} = 2^{-126}$$

- “Computer” exponent=0 and mantissa different from 0 represent sub-normal numbers $-2^{-126} < X < 2^{128}$
- “Computer” exponent=255 represent $\pm\infty$ and Nan.

Double



- 64-bit (8-byte) representation.
- 1 bit for sign: 1 for negative, 0 for positive.
- 52 bits for mantissa.
- 11 bits for the exponent.
  Important: The **true value** of a exponent is
  unsigned exponent representation - 1023

Double Limitations

63	62 … 52	51 … 0
sign	exponent+1023	mantissa
1 bit	11 bits	52 bits

- 0 is represented with mantissa=0 and “computer” exponent=0.
- Max absolute value (all 1’s in mantissa and 11111111110 exponent): $1.1111.....1 * 2^{1023} = 2^{1024} - \varepsilon$
- Min absolute value (0 in mantissa and 1 as “computer” exponent): $1.0000....0 * 2^{-1023} = 2^{-1022}$
- “Computer” exponent=0 and mantissa different from 0 represent sub-normal numbers $-2^{-1022} < X < 2^{1023}$
- “Computer” exponent=2047 represent $\pm\infty$ and Nan.

Examples (1)

31	30 ... 23	22 ... 0
1	10000110	10100101000000000000000
1 bit	8 bits	23 bits

Convert the following float to decimal:

- 0xc3528000 = 1100 0011 0101 0010 1000 0000 0000 0000
- As float parts: 1 10000110 10100101000000000000000
- with the hidden bit: 1   10000110   (1.)10100101
- As decimal: -   134 $1 + 2^{-1} + 2^{-3} + 2^{-6} + 2^{-8}$
- Real exp. (less 127): -   7 $1 + 2^{-1} + 2^{-3} + 2^{-6} + 2^{-8}$
- Sum up: $(-1) * 2^7 * (1 + 2^{-1} + 2^{-3} + 2^{-6} + 2^{-8})$

$= (-1) * (2^7 + 2^6 + 2^4 + 2^1 + 2^{-1})$

$= (-1) * 210.5 = -210.5$

Examples (2)

31	30 … 23	22 … 0
0	10000010	10000100000000000000000
1 bit	8 bits	23 bits

Convert 12.125 to float:

• As polynomial:	+	$12.125 = + 2^{3} + 2^{2} + 2^{-3}$	
• Factor out:	+	$2^{3} * (1 + 2^{-1} + 2^{-6})$	
• As parts:	+	3	(1.)100001
• Represented as:	0	130 *	1000010…0
• Binary exp.:	0	10000010	1000010…0

* Note: we have to add 127 to the real exponent

Binary Rational

- For converting from decimal to binary, we'll use a polynomial of base 2:
  - So, for instance 20.75 to binary:

$$
\begin{aligned}
20.75 &= 2^4 + 2^2 + 2^{-1} + 2^{-2} \\
&= 1*2^4 + 1*2^2 + 1*2^{-1} + 1*2^{-2} \\
&= 10100.11
\end{aligned}
$$

- What about converting to floating point?

Binary Rational

- Here, of course $B = 2, di \in \{0,1\}$
- In order to transform the result to floating point, we'll continue from here:

$$20.75 = 2^4 + 2^2 + 2^{-1} + 2^{-2}$$
$$= 2^4 * (1 + 2^{-2} + 2^{-5} + 2^{-6})$$
$$= 2^4 * (1*2^0 + 1*2^{-2} + 1*2^{-5} + 1*2^{-6})$$
$$= 1.010011 * 2^4$$

Binary Rationals

- Problem: how can we convert simple fractions to binary? Binary representation might require infinite number of digits.
  - For example:

$$\frac{1}{3} = 0.01010101\ldots$$

  - We have an algorithm.

Algorithm for Simple Fractions

- write “0.”
- while (true) do:
  - If $x = 0$
    - Break
  - else
    - $x \leftarrow x \cdot 2$
    - If $x \geq 1$
      - write “1”
      - $x \leftarrow x - 1$
    - else write “0”

Algorithm for Simple Fractions

For instance: $x = \frac{1}{3}$

- 0.

  $\frac{1}{3} \cdot 2 = \frac{2}{3} < 1$

- 0.0

  $\frac{2}{3} \cdot 2 = \frac{4}{3} \geq 1$

- 0.01

  $\frac{4}{3} - 1 = \frac{1}{3}$

  $\frac{1}{3} \cdot 2 = \frac{2}{3} < 1$

- 0.010
  - And so on…
- 0.01010101…

- write "0."
- while (true) do:
  - If $x = 0$
    - Break
  - else
    - $x \leftarrow x \cdot 2$
    - If $x \geq 1$
      - write "1"
      - $x \leftarrow x - 1$
    - else write "0"

Algorithm for Simple Fractions

- From here, converting to floating point is easy:

$$
\begin{aligned}
0.01010101\ldots &= \\
&= 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + \ldots \\
&= 2^{-2} * (1 + 2^{-2} + 2^{-4} + 2^{-6} + \ldots) \\
&= 1.01010101\ldots * 2^{-2}
\end{aligned}
$$

Binary Rationals

- Problems:
  - This algorithms can run to infinity.
  - Furthermore, we do not have an endless supply of digits.
- Solution:
  - Run the main loop the number of times as the number of digits you have for the fraction part.

Fixed Algorithm for Simple Fractions

- write “0.”
- For each available digit to fraction part do:
  - If $x = 0$
    - Break
  - else
    - $x \leftarrow x \cdot 2$
    - If $x \geq 1$
      - write “1”
      - $x \leftarrow x - 1$
    - else write “0”

Mixed Part Numbers

- For mixed *whole* and *simple fraction* parts numbers, like $5\frac{1}{3}$:
  - Convert the integer part to binary as we learned on integers.
  - Convert the fraction part as learned now.
  - Add the results.
  - Only now, if desired, convert to floating point.

סיימנו...

שאלות?

אפקה
המכללה האקדמית
להנדסה בתל אביב