
ELEC2041

Microprocessors and Interfacing

Lectures 19: Floating Point Number Representation – I

<http://webct.edtec.unsw.edu.au/>

April 2006

Saeid Nooshabadi

saeid@unsw.edu.au

ELEC2041 lec19-fp-I.1

Saeid Nooshabadi

Overview

- Floating Point Numbers
- Motivation: Decimal Scientific Notation
 - Binary Scientific Notation
- Floating Point Representation inside computer (binary)
 - Greater range, precision
- IEEE-754 Standard

ELEC2041 lec19-fp-I.2

Saeid Nooshabadi

Review of Numbers

- Computers are made to deal with numbers
- What can we represent in N bits?
 - Unsigned integers:
0 to $2^N - 1$
 - Signed Integers (Two's Complement)
 $-2^{(N-1)}$ to $2^{(N-1)} - 1$

ELEC2041 lec19-fp-I.3

Saeid Nooshabadi

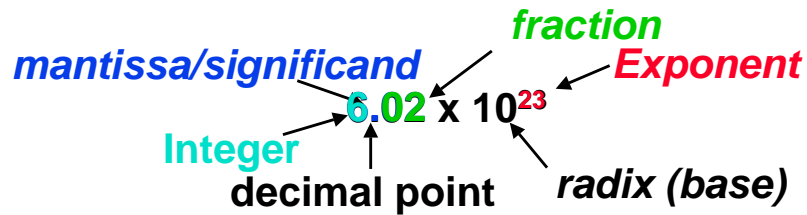
Other Numbers

- What about other numbers?
 - Very large numbers? (seconds/century)
 $3,155,760,000_{10}$ ($3.15576_{10} \times 10^9$)
 - Very small numbers? (atomic diameter)
 0.00000001_{10} ($1.0_{10} \times 10^{-8}$)
 - Rationals (repeating pattern)
 $\frac{2}{3}$ (0.66666666. . .)
 - Irrationals
 $2^{1/2}$ (1.414213562373. . .)
 - Transcendentals
e (2.718...), π (3.141...)
- All represented in scientific notation

ELEC2041 lec19-fp-I.4

Saeid Nooshabadi

Scientific Notation Review



- Normalized form: no leading 0s (exactly one digit to left of decimal point)
 - Alternatives to representing 1/1,000,000,000
 - Normalized: 1.0×10^{-9}
 - Not normalized: $0.1 \times 10^{-8}, 10.0 \times 10^{-10}$
- How to represent 0 in Normalized form?**

Interesting Properties

Finite precision

- i.e. the number of bits in which to represent the significand and exponent is limited.
- Thus not all non-integer values can be represented exactly.
- Example: represent 1.32 as $d_1 d_2 . f_1 \times 10^{d_3}$ (only one digit after the decimal point)

Floating Point Number Range vs Precision (#1/4)

- For simplicity, assume integer Representation for Significand.
- Represent N by $d_1 d_2 d_3$, where

Case 1) $N = d_1 d_2 \times 10^{d_3}$

Case 2) $N = d_1 \times 10^{d_2 d_3}$

Case 3) $N = d_1 d_2 \times 100^{d_3}$

Floating Point Number Range vs Precision (#2/4)

- Which representation can represent the largest value?

Case 1) $N = d_1 d_2 \times 10^{d_3}$ $N = 99 \times 10^9 = 9.9 \times 10^{10}$

Case 2) $N = d_1 \times 10^{d_2 d_3}$

Case 3) $N = d_1 d_2 \times 100^{d_3}$ $N = 99 \times 100^9 = 9.9 \times 10^{19}$

Floating Point Number Range vs Precision (#3/4)

◦ In which representation can the most different values be represented?

Case 1) $N = d_1 d_2 \times 10^{d_3}$

Case 2) $N = d_1 \times 10^{d_2 d_3}$

Case 3) $N = d_1 d_2 \times 100^{d_3}$

◦ All can represent the same number of different values.

◦ Two of the systems can represent an equal number of values, more than the third.

◦ #3 can represent more values than the other two.

◦ #2 can represent more values than the other two.

◦ #1 can represent more values than the other two

Floating Point Number Range vs Precision (#4/4)

◦ In which representation can the most different values be represented?

Case 1) $N = d_1 d_2 \times 10^{d_3}$
 $99 \times 10 + 1 - 9 \times 9 = 910$

– (9×9) because $0d_2 \times 10^{d_3}$
 $= d_2 0 \times 10^{d_3 - 1}$ when $d_3 > 0$
 and $d_2 > 0$

Case 2) $N = d_1 \times 10^{d_2 d_3}$
 $9 \times 100 + 1 = 901$

Case 3) $N = d_1 d_2 \times 100^{d_3}$
 $99 \times 10 + 1 - 9 \times 9 = 910$

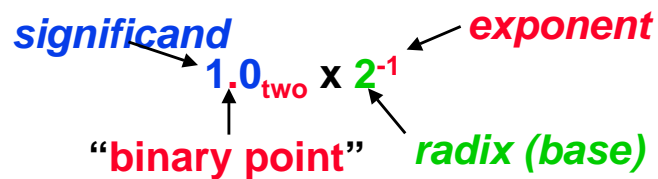
◦ All can represent the same number of different values.

◦ #3 can represent more values than the other two.

◦ #2 can represent more values than the other two.

◦ #1 can represent more values than the other two

Scientific Notation for Binary Numbers



◦ Computer arithmetic that supports it is called **floating point**, because it represents numbers where binary point is not fixed, as it is for integers

- Declare such variable in C as `float`

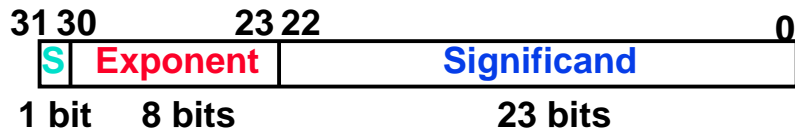
Properties of a good FP Number rep.

- Represents many useful numbers
 - most of the 2^N possible are useful
 - How many LARGE? How many small?
- Easy to do arithmetic (+, -, *, /)
- Easy to do comparison (==, <, >)
- Nice mathematical properties
 - $A \neq B \Rightarrow A - B \neq 0$

Floating Point Representation (#1/2)

◦ Normal format: $+1.xxxxxxxx_{two} * 2^{yyyy}_{two}$

◦ Multiple of Word Size (32 bits)



◦ S represents Sign

◦ Exponent represents y's

◦ Significand represents x's

◦ Leading 1 in Significand is implied

◦ Represent numbers as small as 2.0×10^{-38} to as large as 2.0×10^{38}

Floating Point Representation (#2/2)

◦ What if result too large? ($> 2.0 \times 10^{38}$)

• **Overflow!**

• Overflow \Rightarrow Exponent larger than represented in 8-bit Exponent field

◦ What if result too small? ($< 0 < 2.0 \times 10^{-38}$)

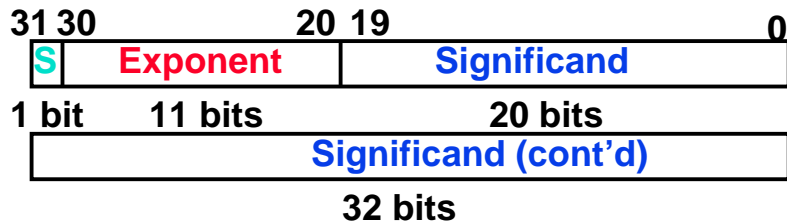
• **Underflow!**

• Underflow \Rightarrow Negative exponent larger than represented in 8-bit Exponent field

◦ How to reduce chances of overflow or underflow?

Double Precision Fl. Pt. Representation

◦ Next Multiple of Word Size (64 bits)



◦ **Double Precision** (vs. **Single Precision**)

• C variable declared as `double`

• Represent numbers almost as small as 2.0×10^{-308} to almost as large as 2.0×10^{308}

• But primary advantage is greater accuracy due to larger significand

Fl. Pt. Hardware

◦ Microprocessors do floating point computation using a special coprocessor.

- works under the processor supervision
- Has its own set of registers

◦ Most low end processors do not have Ft. Pt. Coprocessors

- ARM processor on DSLMU board does not have Ft. Pt. Coprocessor
- Ft. Pt. Computation by software emulation
- Ft. Pt. Emulator: **A student mini project on:** <http://dsl.ee.unsw.edu.au/unsw/projects/armvfp/README.html>. (Closely mimics hardware)
- Some high end ARM processors do have coprocessor hardware support.

IEEE 754 Floating Point Standard (#1/6)

- Single Precision, (DP similar)
 - Sign bit: 1 means negative
0 means positive
 - Significand:
 - To pack more bits, leading 1 implicit for normalized numbers. (Hidden Bit)
 - 1 + 23 bits single, 1 + 52 bits double
 - always true: $0 < \text{Significand} < 1$ (for normalized numbers)
 - What about Zero?
- Next Lecture**

IEEE 754 Floating Point Standard (#2/6)

- Kahan* wanted FP numbers to be used even if no FP hardware; e.g., sort records with FP numbers using integer compares
- Wanted Compare to be faster, by means of a single compare operation, used for integer numbers, especially if positive FP numbers
- How to order 3 parts (Sign, Significand and Exponent) to simplify compare?
- The Author of IEEE 754 FP Standard
<http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html>

IEEE 754 Floating Point Standard (#3/6)

- How to order 3 Fields in a Word?
 $+1.\text{XXXXXXXXXX}_{\text{two}} * 2^{\text{YYYY}_{\text{two}}}$
- “Natural”: Sign, Fraction, Exponent?
 - Problem: If want to sort using integer compare operations, won't work:
 - 1.0×2^{20} vs. 1.1×2^{10} ; latter looks bigger!
- Exponent, Sign, Fraction?
 - Need to get sign first, since negative < positive
- Therefore order is **Sign Exponent Fraction**
 $1.0 \times 10^{20} > 1.1 \times 10^{10}$

0	10000	10100	0	11000	01010
---	-------	-------	---	-------	-------

S	Exponent	Significand
---	----------	-------------

IEEE 754 Floating Point Standard (#4/6)

- Want compare Fl.Pt. numbers as if integers, to help in sort
 - Sign first part of number
 - Exponent next, so big exponent => bigger No.
 - Negative Exponent?
 - 2's comp? 1.0×2^{-1} vs $1.0 \times 2^{+1}$ (1/2 vs 2)
- | | | | |
|-----|---|-----------|------------------------------|
| 1/2 | 0 | 1111 1111 | 000 0000 0000 0000 0000 0000 |
| 2 | 0 | 0000 0001 | 000 0000 0000 0000 0000 0000 |
- This notation using integer compare of 1/2 vs 2 makes $1/2 > 2!$

IEEE 754 Floating Point Standard (#5/6)

◦ Instead, pick notation:

0000 0000 is most negative,
1111 1111 is most positive

- Called **Biased Notation**;
bias subtracted to get number
- 127 in Single Prec. (1023 D.P.)

-127	0000	0000
-126	0000	0001
-1	0111	1110
0	0111	1111
+1	1000	0000
+127	1111	1110
+128	1111	1111

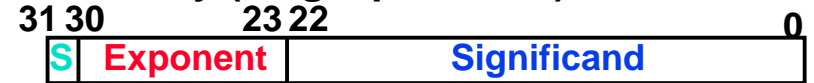
◦ 2's comp? 1.0×2^{-1} vs $1.0 \times 2^{+1}$ (1/2 vs 2)

1/2	0	0111 1110	000 0000 0000 0000 0000 0000
2	0	1000 0000	000 0000 0000 0000 0000 0000

- This notation using integer compare of
1/2 vs 2 makes 2 look greater than 1/2

IEEE 754 Floating Point Standard (#6/6)

◦ Summary (single precision):



1 bit 8 bits

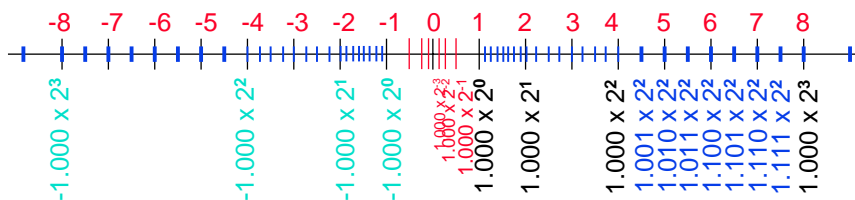
23 bits

◦ $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$

◦ Double precision identical, except with
exponent bias of 1023

Floating Point Number Distribution

◦ Which numbers can be represented?



Using mantissa of 1.000 and positive exponents

and Sign bit

and negative exponents

in each of the intervals of exponentially increasing size
can represent 2^s ($s=3$ here) numbers of uniform difference

But how do we represent 0? Next Lecture!

“And in Conclusion..”

◦ Number of digits allocated to significand and exponent, and choice of base, can affect both the number of different representable values and the range of values.

◦ Finite precision means we have to cope with roundoff error (arithmetic with inexact values) and truncation error (large values overwhelming small ones).

◦ IEEE 754 Standard allows Single Precision (1 word) and Double Precision (2-word) representation of FP. Nos.