Tutorial: Floating Point Numbers

Imperial College London

Department of Computing

C113 Architecture

Lecturer:
Dr. Jana Giceva
j.giceva@imperial.ac.uk

Head Teaching Assistant:
Izaak Coleman
ic711@imperial.ac.uk

Date: March 6, 2018

1 Multiple Choice

Which of the following 8-bit floating point numbers (1 sign, 3 exponent, 4 fraction) represent NaN?

a) 1 000 1111

b) 0 111 1111

c) 0 100 0000

d) 1 111 0000

Consider a tiny floating-point representation with a sign bit, 2 exponent bits, 2 fractional bits, and otherwise following the conventions of IEEE floating point formats. Which of the following bit patterns corresponds to the numerical value of $+1_{10}$?

a) 0 00 00

b) 0 00 01

c) 0 01 00

d) 0 01 10

e) 1 01 00

Assuming that they represent single-precision floating point numbers (with an 8-bit exponent), which of the following has the greatest value?

a) 0x00000000

b) 0x00000001

c) 0x50000000

d) 0xc0000000

2 Conversion Calculations

1. Convert the following decimal numbers to binary.

   (a) $5.5 =$ `101.1`

   (b) $8.25 =$ `1000.01`

   (c) $9.3 =$ `1001.010011001[1001]`

2. Convert `0xC0200000` from 32-bit IEEE floating point to decimal number (remember: 1 bit sign, 8 bits exp, 23 bits significand).

   Solution:
   First convert to binary format:
   `0xC0200000 = 1 1000 0000 0100 0000 0000 0000 0000 000.`
   Which corresponds to:

Sign	Exponent	Significand
1	1000 0000	0100 0000 0000 0000 0000 000

   The exponent E:
   Exponent field is `1000 0000` $= 128$. Hence, $E = 128 - 127 = 1$.
   The mantissa M:
   Significand field is: `01`. Adding the hidden bit $M = 1.01$
   Therefore, the number is:
   $(-1)^1 \times 1.01 \times 2^1 = -10.1 = -2.5_{10}$

3. Convert -31.3 to 32-bit IEEE Floating Point number.

   Solution:
   First convert to a binary format:
   $-31.3 = -11111.010011001[1001]$.
   Next normalize:
   `1.11110 1001 1001 1001 1001 1001` $\times 2^4$.
   Significand field is:
   `1111 0100 1100 1100 1100 110` (23 bits with 1. omitted).
   Exponent field is:
   $4 + 127 = 131 =$ `1000 0011`
   Number is negative, hence the sign field is:
   `1`
   The floating point number is:

Sign	Exponent	Significand
1	1000 0011	1111 0100 1100 1100 1100 110

4. Calculate $31.3 + 13.25$ using IEEE Single Precision arithmetic. *Solution:*

Number	Sign	Exponent	Significand
31.3	0	1000 0011	1111 0100 1100 1100 1100 110
13.25	0	1000 0010	1010 1000 0000 0000 0000 000

Significand of Larger Number = 1.1111 0100 1100 1100 1100 110.
Significand of Smaller Number = 1.1010 1000 0000 0000 0000 000.

Exponents differ by 1. Therefore shit binary point of smaller number for 1 place.

Significand of Larger Number = 1.1111 0100 1100 1100 1100 1100.
Significand of Smaller Number = 0.1101 0100 0000 0000 0000 0000.

Significand of sum = 10.1100 1000 1100 1100 1100 1100

Sum = 10.1100 1000 1100 1100 1100 1100 $\times 2^4$

Which needs to be normalized: 1.0110 0100 0110 0110 0110 0110 $\times 2^5$

Number	Sign	Exponent	Significand
44.55	0	1000 0100	0110 0100 0110 0110 0110 011

3 Tiny Float

Consider a six-bit floating point representation based on the IEEE floating point format, with one sign bit, two exponent bits ($k = 2$) and three fraction bits ($n = 3$).

The table shows a few possible six-bit numbers. Fill in the blank table entries using the following directories:

- `exp`: The value of the exponent field.
- E: The value of the exponent after biasing.
- `frac`: The value of the fraction.
- M: The value of the significand.
- V: The numeric value represented.

Express the value of V as a fraction of the form x/P with suitably chosen P.

Bits	`exp`	E	`frac`	M	V
`0 00 000`	`00`	0	`000`	`0.000`	0
`0 00 110`	`00`	0	`110`	`0.110`	3/4
`0 01 110`	`01`	0	`110`	`1.110`	7/4
`0 10 000`	`10`	1	`000`	`1.000`	2
`0 10 001`	`10`	1	`001`	`1.001`	9/4
`0 10 111`	`10`	1	`111`	`1.111`	15/4