Lecture 09: Floating Points

(CPEG323: Intro. to Computer System Engineering)

Goals for Floating Point

- Support a wide range of values
  - Both the very small (.000000003842) and the very large (6.022×10^{23}).
- Keep as much precision as possible
- Keep encoding that is somewhat compatible with two's complement
  - E.g., 0 in Floating point is 0 in two's complement
  - Can compare two floating point numbers in the same as as comparing two integers.

Scientific Notation (e.g., Base 10)

- Normalized *scientific notation* (aka *standard form* or *exponential notation*):
  - Normalized => No leading 0s
  - 61 is 6.10×10^{2}, 0.000061 is 6.10×10^{-5}
- $r \times E^i$, E where is exponent, i is a positive or negative integer, r is a real number $\geq 1.0, < 10$

Translating To and From Scientific Notation

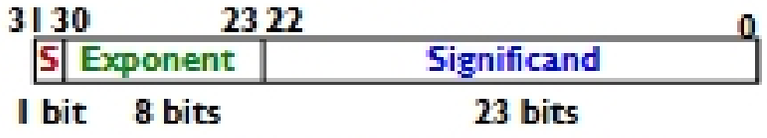
- Consider the number 8.12×10^{4}
- To convert to ordinary number, shift the decimal to the right by 4.
  - Result: 81200
- For negative exponents, shift decimal to the left
  - 8.12×10^{-4} => .0000812
- Can reverse this process to go from ordinary number to scientific notation
  - .00812 => 8.12×10^{-3}

What About *Real* Numbers in Base 2?

- $r \times 2^i$, where i is a positive or negative integer, r is a real number $\geq 1.0, < 2$
- Computers version of normalized scientific notation called *Floating Point* notation

Floating Point Encoding

- Use normalized, Base 2 scientific notation:
  $+1.xxx\ldots x_{two} * 2^{yyy\ldots y_{two}}$
- Split a 32 bit word into 3 fields:



- S represents Sign (1 is negative, 0 positive)
- Exponent represents y's
- Significand represents x's
- Represent numbers as small as ~2.0×10^{-38} to as large as ~2.0×10^{38}

The Exponent Field

- Definitely want a signed exponent.
- Use two's compliment?
  - Smallest exponent looks like 1000000, largest exponent looks like 01111111.
  - Can reuse integer comparison hardware if exponents range from 00000000 for smallest to 11111111 for largest.
- Use biased two's compliment. Instead of -128 through 127 being 10000000 through 01111111, define -127 through 128 to be 00000000 through 11111111.
  - 1 => -128 => 10000000
  - 127 => -254 => 11111110

Floating Point Encoding

- Note the implicit 1 in front of the Significand.

Example: Converting Decimal to FP

-2.340625×10^1

- Denormalize: -23.40625
- Convert integer part:
  - $23 = 16 + (7 = 4 + (3 = 2 + (1))) = 10111_2$
- Convert fractional part:
  - $.40625 = .25 + (.15625 = .125 + (.03125)) = .01101_2$
- Put parts together and normalize:
  - $10111.01101 = 1.011101101 \times 2^4$
- Convert exponent: $127 + 4 = 10000011_2$

1	1000 0011	011 1011 0100 0000 0000 0000

No, questions? It is your turn!

- What is the single-precision representation of 347.625?
  1. First convert the number to binary: $347.625 = 101011011.101_2$.
  2. Normalize the number by shifting the binary point until there is a single 1 to the left:

     $101011011.101 \times 2^0 = 1.01011011101 * 2^8$

  3. The bits to the right of the binary point comprise the fractional field f.
  4. The number of times you shifted gives the exponent. The field e should contain: exponent + 127.
  5. Sign bit: 0 if positive, 1 if negative.

  0 10000111 01011011101000...0

  s e f

Example: Converting Binary FP to Decimal

0	0110 1000	101 0101 0100 0011 0100 0010

- Sign: 0 => positive
- Exponent:
  - $0110\ 1000_{two} = 104_{ten}$
  - Bias adjustment: 104 - 127 = -23
- Significand:

  $1 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5} + \ldots$
  $= 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-9} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-22}$
  $= 1.0 + 0.666115$

- Represents: $1.666115_{ten} * 2^{-23} \sim 1.986 * 10^{-7}$

Your turn, again!

- Let' s find the decimal value of the following IEEE number.

  1 01111100 11000000000000000000000

- First convert each individual field to decimal.
  - The sign bit s is 1.
  - The e field contains $01111100 = 124_{10}$
  - The significand is $0.11000\ldots = 0.75_{10}$
- Then just plug these decimal values of s, e and f into our formula.

$$(1 - 2s) * (1 + f) * 2^{e-Bias}$$

- This gives us $(1 - 2) * (1 + 0.75) * 2^{124-127} = (-1.75 * 2^{-3}) = -0.21875$.

Floating-point representation

– We can represent floating-point numbers with three binary fields: a sign bit s, an exponent field e, and a fraction field f.

– The IEEE 754 standard defines several different precisions.

  – Single precision numbers include an 8-bit exponent field and a 23-bit fraction, for a total of 32 bits.

  – Double precision numbers have an 11-bit exponent field and a 52-bit fraction, for a total of 64 bits.

MIPS Floating Point Instructions

- C has single precision (float) and double , precision (double) types
- MIPS instructions: .s for single, .d for double
  - Fl. Pt. Addition single precision: add.s
    Fl. Pt. Addition double precision: add.d
  - Fl. Pt. Subtraction single precision: sub.s
    Fl. Pt. Subtraction double precision: sub.d
  - Fl. Pt. Multiplication single precision: mul.s
    Fl. Pt. Multiplication double precision: mul.d
  - Fl. Pt. Divide single precision: div.s
    Fl. Pt. Divide double precision: div.d

MIPS Floating Point Instructions

- C has single precision (float) and double precision (double) types
- MIPS instructions for comparison: .s for single, .d for double
- Since rarely mix integers and Floating Point, MIPS has separate registers for floating-point operations: $f0, $f1, …, $f31
- Need data transfer, comparison and branch instructions for these new registers

Question

What is the decimal equivalent of the floating pt # above?

```
1: -1.75
2: -3.5
3: -3.75
4: -7
5: -7.5
6: -15
7: -7 * 2^129
8: -129 * 2^7
```

Answer

What is the decimal equivalent of:

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

$(-1)^1 \times (1 + .111) \times 2^{(129-127)}$

$-1 \times (1.111) \times 2^{(2)}$

-111.1

-7.5

```
1: -1.75
2: -3.5
3: -3.75
4: -7
5: -7.5
6: -15
7: -7 * 2^129
8: -129 * 2^7
```

Last question

Order the following (single precision) floating point numbers in increasing value from 1 to 4, where 1 is the smallest number (i.e., the most negative) and 4 is the largest number (i.e., the most positive).

$(1 - 2s) * (1 + f) * 2^{e-bias}$