Computer Arithmetic

Dr. Ahmed H. Zahran

WGB 182

a.zahran@cs.ucc.ie

Objectives

- Understand the binary representation of floating point
- Understand the HW implementation of basic floating point operation

Floating Point

- Representation for non-integral numbers
  - Including very small and very large numbers
- Like scientific notation
  - -2.34×10^{56}
  - $+0.002 \times 10^{-4}$
  - $+987.02 \times 10^{9}$
- In binary
  - $\pm 1.xxxxxxx_2 * 2^{yyyy}$
  - Called binary points
- Types float and double in C

Floating Point Standard

- Defined by IEEE Std 754-1985
- Developed in response to divergence of representations
  - Portability issues for scientific code
- Now almost universally adopted
- Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)

IEEE Floating-Point Format

	single: 8 bits double: 11 bits	single: 23 bits double: 52 bits
S	Exponent	Fraction

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: sign bit (0 ⇒ non-negative, 1 ⇒ negative)
- ***Normalize significand***: 1.0 ≤ |***significand***| < 2.0
  - Always has a leading pre-binary-point 1 bit, so no need to represent it explicitly (***hidden bit***)
  - Significand is Fraction with the “1.” restored
- Exponent: excess representation: actual exponent + **Bias**
  - Ensures exponent is unsigned
  - **Single: Bias = 127; Double: Bias = 1203**
  - Exponents 00000000 and 11111111 reserved

Floating-Point Example

- Represent –0.75
  - $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
  - S = 1
  - Fraction = $1000\ldots00_2$
  - Exponent = –1 + Bias
    - Single: –1 + 127 = 126 = 01111110_2
    - Double: –1 + 1023 = 1022 = 01111111110_2
- Single: 1011111101000…00
- Double: 1011111111101000…00

Floating-Point Example

- What number is represented by the single-precision float

  11000000101000…00

  - S = 1
  - Fraction = $01000\ldots00_2$
  - Fxponent = $10000001_2 = 129$

- $x = (-1)^1 \times (1 + 01_2) \times 2^{(129-127)}$

  $= (-1) \times 1.25 \times 2^2$

  $= -5.0$

Single-Precision Range

- Smallest value
  - Exponent: 00000001
    ⇒ actual exponent = 1 – 127 = –126
  - Fraction: 000…00 ⇒ ***significand*** = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2_{10} \times 10^{-38}$
- Largest value
  - exponent: 11111110
    ⇒ actual exponent = 254 – 127 = +127
  - Fraction: 111…11 ⇒ ***significand*** ≈ 2.0
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4_{10} \times 10^{+38}$

Double-Precision Range

- Smallest value
  - Exponent: 00000000001
    ⇒ actual exponent = 1 – 1023 = –1022
  - Fraction: 000…00 ⇒ significand = 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2_{10} \times 10^{-308}$

- Largest value
  - Exponent: 11111111110
    ⇒ actual exponent = 2046 – 1023 = +1023
  - Fraction: 111…11 ⇒ significand ≈ 2.0
  - $\pm 2.0 \times 2^{+1023} \approx \pm 1.8_{10} \times 10^{+308}$

Accurate Arithmetic

- Different between computer number and number in real world
  - Computer numbers have limited size → ***limited precision***
  - ***Programmers must remember these limits and write programs accordingly***
- IEEE Std 754 specifies five rounding control
- Not all FP units implement all options
  - Most programming languages and FP libraries just use defaults

Floating-Point Precision

- Relative precision
  - Single: approx 2^{-23}
    - Equivalent to $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 7$ decimal digits of precision
  - Double: approx 2^{-52}
    - Equivalent to $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$ decimal digits of precision

FP Instructions in MIPS

- Separate FP **registers**
  - ***32 single-precision***: \$f0, \$f1, … \$f31
  - ***Paired for double-precision***: \$f0/\$f1, \$f2/\$f3, …
    - Release 2 of MIPs ISA supports 32 × 64-bit FP reg’s
- FP instructions operate only on FP registers
  - Programs generally don’t do integer ops on FP data, or vice versa
  - More registers with minimal code-size impact
- FP load and store instructions
  - lwc1, ldc1, swc1, sdc1
    - e.g., ldc1 \$f8, 32(\$sp)

FP Instructions in MIPS

- Single-precision arithmetic
  - add.s, sub.s, mul.s, div.s
    - e.g., add.s $f0, $f1, $f6
- Double-precision arithmetic
  - add.d, sub.d, mul.d, div.d
    - e.g., mul.d $f4, $f4, $f6
- Single- and double-precision comparison
  - c.*xx*.s, c.*xx*.d (*xx* is eq, lt, le, …)
  - Sets or clears ***FP condition-code bit***
    - e.g. c.lt.s $f3, $f4
- Branch on FP condition code true or false
  - bc1t, bc1f
    - e.g., bc1t TargetLabel

Reading

- Section 3.5