Floating Point Arithmetic

Murat Manguoğlu

Sections 1-2

Patterson Hennessy, Computer Organization and Design: The Hardware/Software Interface, 3rd Edition

Floating-Point Addition

- **Consider a 4-digit decimal example**
  - $9.999 \times 10^{1} + 1.610 \times 10^{-1}$
- **1. Align decimal points**
  - Shift number with smaller exponent
  - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
- **2. Add significands**
  - $9.999 \times 10^{1} + 0.016 \times 10^{1} = 10.015 \times 10^{1}$
- **3. Normalize result & check for over/underflow**
  - 1.0015×10^{2}
- **4. Round and renormalize if necessary**
  - 1.002×10^{2}

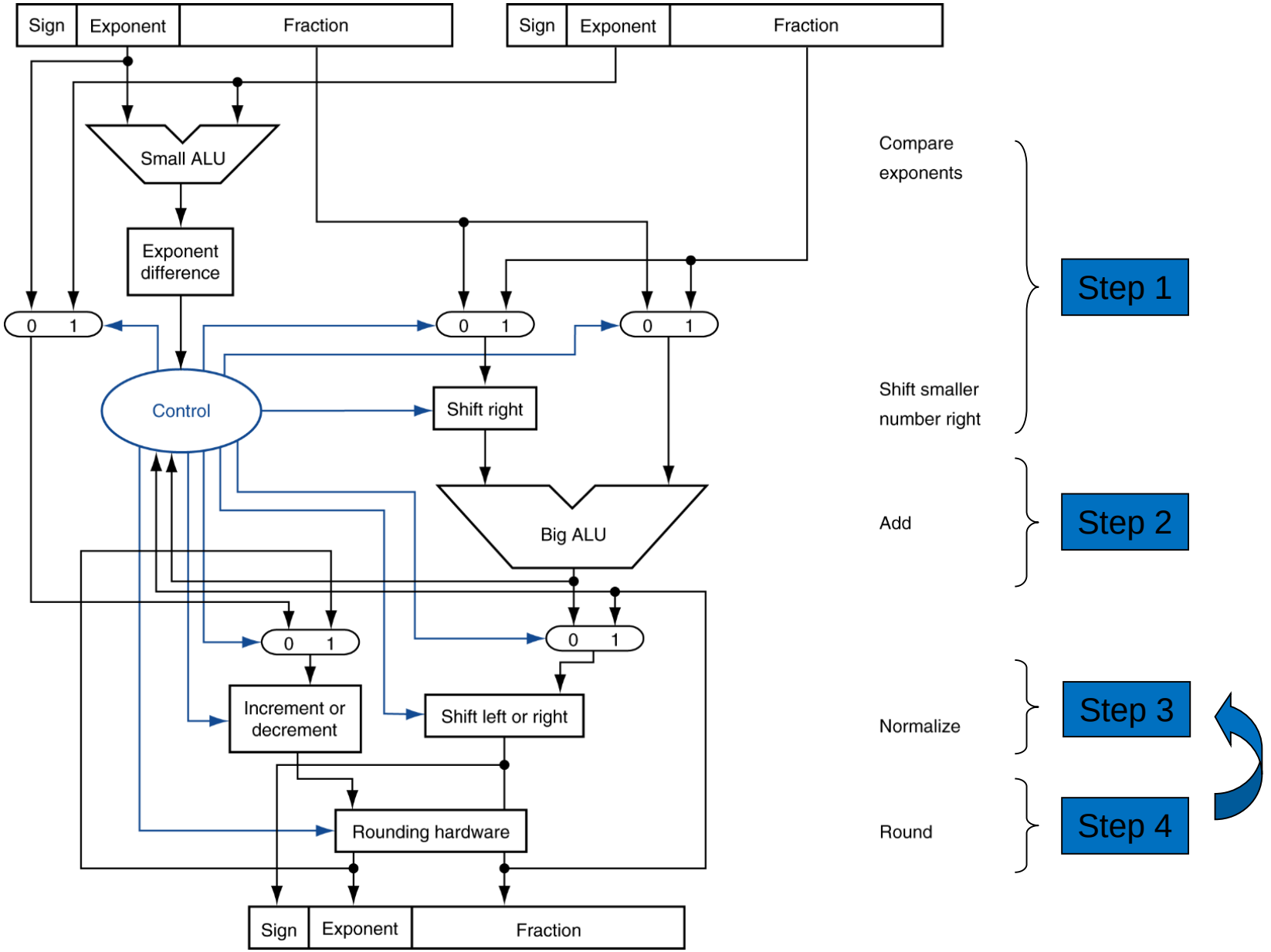
Floating-Point Addition

- **Now consider a 4-digit binary example**
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + –0.4375)
- **1. Align binary points**
  - Shift number with smaller exponent
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- **2. Add significands**
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-}1 = 0.001_2 \times 2^{-1}$
- **3. Normalize result & check for over/underflow**
  - $1.000_2 \times 2^{-4}$, with no over/underflow
- **4. Round and renormalize if necessary**
  - $1.000_2 \times 2^{-4}$ (no change) = 0.0625

FP Adder Hardware

- **Much more complex than integer adder**
- **Doing it in one clock cycle would take too long**
  - Much longer than integer operations
  - Slower clock would penalize all instructions
- **FP adder usually takes several cycles**
  - Can be pipelined

FP Adder Hardware



Floating-Point Multiplication

- **Consider a 4-digit decimal example**
  - $1.110 \times 10^{10} * 9.200 \times 10^{-5}$
- **1. Add exponents**
  - For biased exponents, subtract bias from sum
  - New exponent = 10 + –5 = 5
- **2. Multiply significands**
  - $1.110 \times 9.200 = 10.212 \Rightarrow 10.212 \times 10^{5}$
- **3. Normalize result & check for over/underflow**
  - 1.0212×10^{6}
- **4. Round and renormalize if necessary**
  - 1.021×10^{6}
- **5. Determine sign of result from signs of operands**
  - $+1.021 \times 10^{6}$

Floating-Point Multiplication

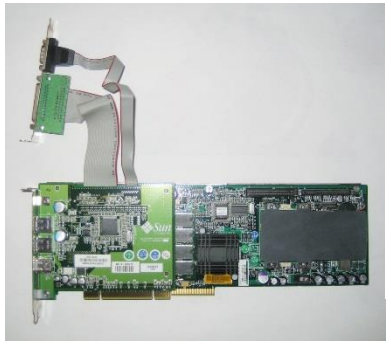
- **Now consider a 4-digit binary example**
  - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$ (0.5 × –0.4375)
- **1. Add exponents**
  - Unbiased: –1 + –2 = –3
  - Biased: (–1 + 127) + (–2 + 127) = –3 + 254 – 127 = –3 + 127
- **2. Multiply significands**
  - $1.000_2 \times 1.110_2 = 1.1102 \Rightarrow 1.110_2 \times 2^{-3}$
- **3. Normalize result & check for over/underflow**
  - $1.110_2 \times 2^{-3}$ (no change) with no over/underflow
- **4. Round and renormalize if necessary**
  - $1.110_2 \times 2^{-3}$ (no change)
- **5. Determine sign: +ve × –ve ⇒ –ve**
  - $-1.110_2 \times 2^{-3} = -0.21875$

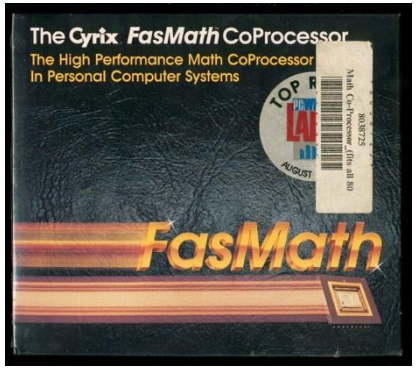
FP Arithmetic Hardware

- **FP multiplier is of similar complexity to FP adder**
  - But uses a multiplier for significands instead of an adder
- **FP arithmetic hardware usually does**
  - Addition, subtraction, multiplication, division, reciprocal, square-root
  - FP $\leftrightarrow$ integer conversion
- **Operations usually takes several cycles**
  - Can be pipelined

x86 FP Architecture

- **Originally based on 8087 FP coprocessor (x87)**
  - 8 × 80-bit extended-precision registers
  - Used as a push-down stack
  - Registers indexed from TOS: ST(0), ST(1), …





x86 FP Architecture

- **FP values are 32-bit or 64 in memory**
  - Converted on load/store of memory operand
  - Integer operands can also be converted on load/store
- **Not easy to generate and optimize code**
  - Result: poor FP performance
  - With XMM registers in X86-64 this is no longer the case

Streaming SIMD Extension 2 (SSE2)

- **Adds 4 × 128-bit registers**
  - Extended to 8 registers in AMD64/EM64T
- **Can be used for multiple FP operands**
  - 2 × 64-bit double precision
  - 4 × 32-bit double precision
  - Instructions operate on them simultaneously
    - Single-Instruction Multiple-Data

Thank you!