ECE 271 – Microcomputer Architecture and Applications Lecture 20

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

9 April 2019

Announcements

- Read Chapter 12

Floating Point / Fixed Point

- We have been working with integers, signed and unsigned.
- How can you represent fractional numbers?
- How does it work in base 10?
  1234.56 $= 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1} + 6 \times 10^{-2}$
- You can do the same thing in binary (base2)
  1010.10 $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}$
  This is 10.5 in decimal

- You can do this for arbitrary bases.
  You have to keep track of the decimal or “radix” point handled

Fixed Point

- Fix the decimal point somewhere inside the number
- In decimal, note that $123.45 + 12.51$ is the same as $12345+1251$, just you move the decimal point.
- So we can have fractional parts of integers by just moving the decimal point.

Fixed Point – Notation

- UQm.n = Unsigned fixed point, m bits to left of point, n bits to right

- Qm.n = Signed fixed point, m bits to left (one is sign bit) n bits to right

Fixed Point – Size

- Tradeoff in m vs n values
- Accuracy – how close it is to the number you are trying to represent
- Resolution – the smallest change that will give you another value

Fixed Point – Q16.16

- Q16.16 – 16 bits of integer, 16 bits of fraction
- Use regular integer register and regular math
- Limited range, you now have smallest max value you can have
- Also need to track the radix point yourself
- Binary example

- $101.111 = 5 + 0.25 + 0.125 = 5.375$

Addition

- Easy. Make sure Q for both is the same and just add
- $0101.1 + 0101.1 = 1011.0$

Subtraction

- Just like addition

Multiplication

- Think about decimal. 10.1 * 2.0 = 20.2
  but how do you do it

```
   10.1
    2.0
  =====
    000
   202
=========
   2020
```

```
Then you shift the point left by the number to
20.2
```

- What you are doing is 101×10^{-1} times 20×10^{-1} so you can do the first, then do the second

- Regular multiply

- Need to adjust radix point back

- 0010.1 * 0010.1 = 0000 0000 0000 0010 1000 0000 0000 0000

- 0x28000*0x28000 = x6 4000 0000 Q16.16*Q16.16 = Q32.32
- ¿¿ 16 = 0x6 4000 = 6.25
- ARM SMULL instruction 32x32 = high/low 64-bit values

Division

- Similar to multiply
- 0x28000 / 0x28000 = 1 Q16.16 / Q16.16 = Q1. ¡¡ 16
  What happens to fraction part?
  Shift one by ¡¡ 16 first before divide to not lose all fraction

Converting to int

- Just shift right by Q.

- Rounding

Overflow

- can be a problem

Why ARM is good at it

- barrel-shift instructions

Can you exactly represent all numbers?

- In decimal, 1/3? No
- In binary, only combinations of powers of 2. So even things like 1/5 (0.2) you can't represent exactly.
- Irrational numbers like Pi?

Arbitrary Precision Number Libraries

- If you need *exact* values
- Tend to be slow and use lots of RAM, but give exact results

Fixed Point Limited Range

- What if you want to operate on numbers with different Q values
- What if you want to add very large or very small numbers