Floating Point Representation

Tom Kelliher, CS 220

Nov. 6, 2009

1 Administrivia

Announcements

Assignment

Read 4.1–4.3 for Wednesday.

From Last Time

Binary addition, subtraction, multiplication.

Outline

1. Floating point numbers.
2. Floating point representation.
3. Floating point addition.

Coming Up

Project day, then introduction to building a simple MIPS datapath.

2 Floating Point Numbers

1. Do we *really* need a way to represent floating point numbers?

2. Recall:
$$-1^{\text{sign}} \times \text{mantissa} \times 2^{\text{exponent}},$$
where mantissa is normalized: $1 \leq \text{mantissa} < 2$.

   Note: mantissa is sign-magnitude notation, while exponent is 2's complement!

3. This has to fit into one word. What are the tradeoffs?

   Is there a way to increase the range, while keeping the same precision? (Hint: consider changing the base of the exponent.)

4. Finally, we would like to compare floating point numbers using integer compare hardware. How should we order the fields of the floating point representation in order to achieve this?

   What problem will we run into, and how do we solve it?

5. Underflow and overflow.

6. Formats:

   (a) Single precision: 23 bit mantissa, eight bit exponent.

   (b) Double precision: 52 bit mantissa, 11 bit exponent.

7. Some real problems:

   (a) Accumulated errors in extended computations.

   Rounding techniques, in conjunction with guard, round, and sticky bits help with this.

   (b) Hardware that doesn't do floating point divide.

   (c) Getting floating point right is hard. (Ask Intel.)

   Doing it right and fast is even harder.

3 Floating Point Representation

Fields of a single precision floating point number:

1. Mantissa sign.
2. Exponent: Bias 127, 8 bits.

   How to use the bias:

   (a) Value → Representation: Add the bias.

   (b) Representation → Value: Subtract the bias.
3. Mantissa magnitude, 23 bits. Normalized.

Additional Feature: The hidden bit.

Values (f is the fractional part of the mantissa):

1. V = NaN, if $e = 255$ and $f \neq 0$.
2. V = $-1^s \times \infty$, if $e = 255$ and $f = 0$.
3. V = $-1^s \times 1.f \times 2^{e-127}$, if $0 < e < 255$.
4. V = $-1^s \times 0.f \times 2^{-126}$, if $e = 0$ and $f \neq 0$. (Denormalized.)
5. V = $-1^s \times 0$, if $e = 0$ and $f = 0$.

Examples:

1. Represent the value 3.14159E8 in IEEE FP. (In integer hex, it's `0x12B9AF98`.)

   FYI, here's a C program to confirm the result:

```
#include <stdio.h>

int main()
{
   float x = 3.14159E8;
   int *ptr = (int *) &x;

   printf("%E %X\n", x, *ptr);

   return 0;
}
```

2. What is the value of the FP representation `0XC1320000`?

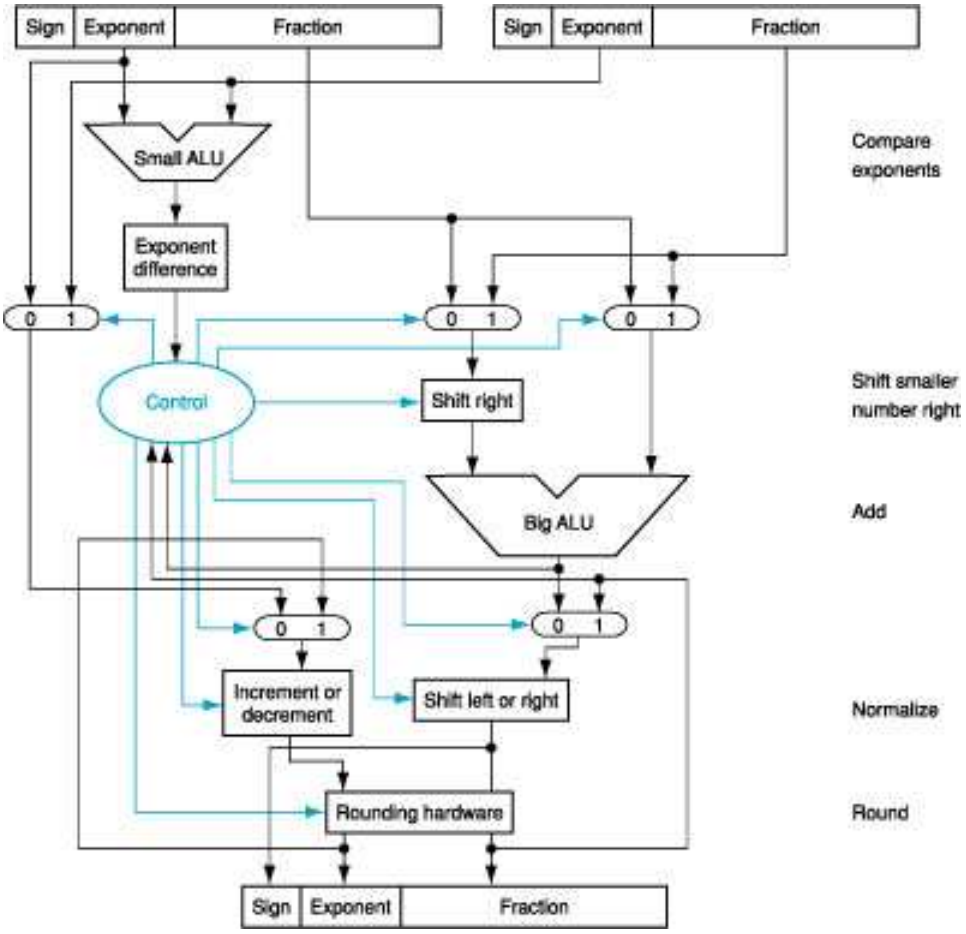
4 Floating Point Addition

1. How do we do this with pencil and paper?

   Add decimal 9.567E3 and 5.678E2.

2. What is the algorithm?

3. FP hardware:



Explain all the blocks, particularly the muxes!