Slide 1

Administrivia

- (In e-mail.)

Slide 2

Floating Point in MIPS Architecture

- Architecture supports IEEE 754 “single” (32 bits) and “double” (64 bits).
- Architecture defines 32 floating-point registers (`$f0` through `$f31`), used singly for single-precision, in pairs for double-precision.

Slide 3

MIPS Floating-Point Instructions

- Arithmetic instructions (single-precision):

  Basics: `add.s`, `sub.s`, `mul.s`, `div.s`.

  Interesting extras: `abs.s`, `neg.s`, `sqrt.s`.

  All have double-precision counterparts (replace `.s` in name with `.d`).

- Load/store instructions:

  Single-precision `lwc1`, `swc1`.

  Double-precision `ldc1`, `sdc1` (pseudoinstructions).

  Pseudoinstructions `li.s`, `li.d`.

Slide 4

MIPS Floating-Point Instructions, Continued

- Comparisons:

  `c.eq.s`, `c.lt.s`, etc., plus double-precision counterparts.

  These set a bit true/false, which can be used by `bc1t`, `bc1f`.

- Data copying:

  `mov.s`, `mov.d` to copy from one (pair of) register(s) to another.

  `mtc1`, `mfc1` to copy from general-purpose register to floating-point register and vice versa. *NOTE* that this just copies bits!

- Conversion between integer and floating point:

  `cvt.w.s`, `cvt.s.w`, and double-precision counterparts.

Slide 5

Floating Point in MIPS, Continued

- Some instruction names include `c1`. Short for "coprocessor 1". What's that? well, as textbook mentions, once upon a time chips for PC-class machines didn't have enough transistors to implement floating-point arithmetic, so if it was included in the hardware at all, it was as a separate chip ("coprocessor"). This may also explain why there are distinct floating-point registers. Now a thing of the past, but the name stuck.
- "If at all"? was it not possible on machines without floating-point hardware to do floating-point arithmetic? Well . . . (Minute-essay question.)

Slide 6

Example Programs

- Simple example: Sample program `echo-float.s`.
- Example of using floating-point instructions: program to compute square root using Newton's method. Sample program `newton.s`.
- Also use SPIM to examine what its assembler does with `li.s`, `li.d`: Loads bit patterns into `$at`, moves to floating-point register. (Assembler computes bit patterns.)

Slide 7

Minute Essay

- Can you still do floating-point arithmetic without hardware support? If so, how?

Slide 8

Minute Essay Answer

- You can — in software. (Eek! slow but if packaged in libraries better than nothing?)