Slide 1

Administrivia

- Reminder: Exam 1 Wednesday.
- Written problems for Homeworks 1, 2, and 3 graded and returned. I may be able to finish grading the one for Homework 4 later today.
- Sample solutions to all homework problems (written and programming) will be available via Google Drive.
- Sample solutions to quizzes available on course Web site (bottom of “lecture topics” etc.).

Slide 2

Minute Essay From Last Lecture

- (Review.)

Slide 3

Homework 3 Essays

- Many people found at least some parts challenging, but several also said they were starting to understand better. Good!
- Some students had trouble translating jump and branch instructions into machine language.
- One student wanted a review of how to read that “green card’.
- One student wanted a review of loops.
- One student said the problem “showed me how little I knew”.

Slide 4

Homework 3 Essays, Continued

- One student commented that the biggest challenge with assembly-language coding was keeping track of registers (what’s used for what). Agreed! I say source-code comments help.
- One student asked about how jumping to an address with a different high-order byte. I’m not entirely clear on that myself!

Exam Hints — Concepts and Terms

- Instruction set architecture — what it is, what does it mean for two machines to implement the same one.
- “From source to executable” — what compilers, assemblers, linkers, and loaders do.

Slide 5

Exam Hints — Performance

- Textbook’s measure of CPU execution time.
- (There might be calculations, but if so they’ll be simple ones.)

Slide 6

Slide 7

Exam Hints — MIPS Assembly and Machine Language

- "What does this MIPS code do?"
- C to MIPS assembly, and possibly vice versa.
- "Write some MIPS assembly code to . . . " If a procedure, must follow conventions for saving/restoring values, passing parameters, returning a value.
- MIPS assembly to machine language and vice versa.

Slide 8

Review — Addressing Modes

- `bne` and `beq` use "PC-relative addressing":

  16 bits available to represent address. Even dropping two low-order bits (always zero), clearly not enough for 32-bit address!

  So, value in instruction is offset from next instruction to target, divided by four.
- `j` and `jal` use "pseudo-direct addressing":

  26 bits available to represent address. Dropping two low-order bits, that leaves 28. Dropping four high-order bits sounds risky, but most of the time same as next instruction, so that's what's done.

Review — “Green Card” Information

Slide 9

- (Review.)
- (To me the only tricky thing is opcode versus “function code”. The latter is only used for the three-registers arithmetic instructions, and shows up in that last column of the table on the card.)

Exam Hints — Assembling and Linking

Slide 10

- How assembly and linking works. I won’t ask you to do as much as in the homework, but possibly a simpler similar problem.

Slide 11

Minute Essay

- Any questions? otherwise just “sign in”.