Slide 1

Administrivia

- Reminder: Midterm Tuesday after break. “Review sheet” on Web talks about format, topics.
- Sample solutions for homeworks on Google Drive. Will be updated soon (two problems incomplete).
- Graded work — I will let you know by e-mail if I make progress.
- There’s a strong possibility that I’ll have office hours part of the Monday before the exam (probably 1:30pm to 3:30pm). I’ll let you know for sure by e-mail.

Slide 2

Hints and Tips

- In computing performance-related things, do check that your answers are plausible w.r.t. order of magnitude.
- If asked to write MIPS code, avoid “pseudo-instructions” unless told you can use them. And pay attention to syntax — a stray comma here and there is not a problem, but ignoring rules about number and types of operands is.

Slide 3

Linking, Revisited

- (Unlikely that you'll be asked to do anything as complete as the homework problem, but there might be *some* question(s) about this process, so a short review here.)
- Easy to get lost in details, but keeping the "big picture" in mind may help:

  The goal of the process is to be able to combine separately-generated "object files", usually produced by an assembler or compiler, into one "executable" that can be loaded into memory and executed. So what does that imply . . .

Slide 4

Linker Input/Output

- Output of the linker is meant to be something the operating system can load into memory and execute. Notice that this might not be true for output of assembler or compiler — e.g., if there are calls to procedures not included in the source code (e.g., to library functions). Also probably need information on how much memory is needed for combined code (machine instructions) and combined "fixed data" (as opposed to data for which we get memory at runtime). For SPIM (which incorporates a very primitive o/s) we assume that we always load the code and data into the same place in memory.
- Input to linker is "object files". What do they need to include to make the above possible? for one thing, sizes (of code and fixed data). for another, a list of labels that could be referenced from other object code. finally, a list of places where the assembler/compiler wasn't able to completely generate code because there was a reference to a label (possibly in another object file).

Slide 5

Linker Processing

- In principle anyway, what a linker does is all kind of common sense based on inputs and desired outputs . . .
- Start by figuring out where in memory the code and data pieces (one set from each object file) are to be place, and combined sizes.
- Also build a list of labels and their locations.
- Finally, go through the information about instructions that couldn't be completely assembled earlier and "fix them up". (Notice that some are absolute addresses and some are relative addresses.)

Slide 6

Review

- (Item by item through review sheet, questions.)

Slide 7

Minute Essay

- Anything noteworthy about homework 2?
- (And best wishes for a good spring break!)