Slide 1

Administrivia

- Reminder: Quiz 3 Wednesday. Topics from chapter 2. (Some possibility this won't happen and the next quiz will be after the first exam. I will send e-mail tomorrow one way or another.)
- If you're watching the video lectures, don't forget to send me an answer to the minute essay. Only half the students have done that for 2/15 (and they're mostly saying it was good review).
- If you didn't do well on Quiz 2 (and several people didn't), don't panic: I drop the lowest quiz score. But it should be a warning that you might not be understanding?

Slide 2

Video Lecture Follow-Up

- Most people who've viewed the one for 2/22 said they found it helpful. (Good! I hoped it would be.)

Slide 3

Linking — Offsets, Addresses, Etc.

- One person asked about which instructions need to be included in "relocation information". Criterion to use is "can I translate this instruction completely and correctly without knowing where in memory something (code or data) resides?"

  So, `jal`? `jr`? `la`? `lw` (the real instruction)?

- One person asked about offset versus address. Offset is relative to start of the segment (text/code or data); address is memory address and depends on where in memory the segment is loaded.

Slide 4

MIPS Assembler Directives for Data

- One person asked about `.word`. Examples may help:

```
A:  .word 100
B:  .word 1, 2
```

  C equivalent (assuming 32-bit `int`s):

```
int A = 100;
int B[] = {1, 2};
```

Slide 5

Arithmetic Overflow

- One person asked about the factorial example and why it just quietly gives wrong results for larger inputs (and they don't even have to be very large!).
- Compare to what happens if you write an equivalent program in a high-level language ...
- When result-in-process gets too big to fit into available space (32-bit register here), two options: Hardware can signal exception, or it can just drop high-order bits. Result can look negative, or it can just be wrong.

Slide 6

Arithmetic Overflow, Continued

- "Signal exception"? Yes. We'll talk more about this later, but possible to build hardware that detects overflow and does something. (Apparently SPIM doesn't do this.)
- But since many programming languages ignore overflow, often instructions have signed form that checks and unsigned form that doesn't (e.g., `addu` versus `add`).
- Really careful programmers put in their own checks for overflow. May actually be *easier* in assembly language: `mult` instruction generates 64-bit result in special-purpose registers `lo` and `hi`.

Slide 7

Compiling Revisited

- As previously mentioned, compilers are big and complicated partly because they try to generate efficient code (while, one hopes, preserving the program's meaning!).
- As an example: Textbook goes into some detail about compiling C code to loop through an array, showing a version that uses indices and one that uses pointers. A "good" compiler will likely generate the same code for both.

  Can test this with `gcc`: `-S` generates (x86) assembler, so we could write it both ways, compile to assembler, and compare. Same *if* compiled with `-O`.
- Note in passing that compiler optimizations can play havoc with attempts to time things: C compilers are allowed to just skip any code that doesn't have an observable effect (i.e., result isn't printed or otherwise used). (In practice they may or may not.)

Slide 8

SPIM Tips

- Debugging MIPS assembly programs with SPIM can be tedious. I find the command-line version easier to work with. Some tips:

  Can set a breakpoint with `breakpoint`. Only works with labels defined as global symbols.

  Can print values of variables, but again only if defined as global symbols.

  (Example / demo?)
- SPIM also useful as a way to check work, e.g., translating assembly to machine language, or assembling / linking.

  (Example / demo?)

Slide 9

MIPS Programming — Another Example

- Try revising the factorial example (under "sample programs" on the course Web site) to use a loop rather than recursion.
- Try this individually or in pairs. When/if you have something, copy to my directory:

```
chmod go+r pgmname.s
cp -p pgmname.s
/users/bmassint/TEMP2321/your_username
```

Slide 10

Minute Essay

- Was it helpful to do a practice problem in class like this?