Slide 1

Administrivia

- (By e-mail.)

Slide 2

Another Programming Example

- Programming examples looked at previously are pretty simple.
- Try a definitely-more-complicated example: In Low-Level I ask students to implement Euclid's algorithm for finding greatest common divisor (GCD). Can be done with recursion or iteration. Try implementing both . . .

Slide 3

Overall Strategy

- I like to start by writing a C program, but keeping in mind that some things that work in C (checking whether I/O succeeds, for example) can't be done in SPIM.
- Why? figure out program logic in a language easier to debug than assembly.
- Look at pre-written code (nothing deep here) . . .
- Then translate to MIPS assembler. Some parts (input and output, e.g.) you can write directly (doing a lot of copy-paste). For other parts, I like to translate the C to C with `goto` and from that to assembly language.

Slide 4

Sidebar: Pseudoinstructions

- I've discouraged use of pseudoinstructions: Better to understand what the hardware can actually do. Value in that and I'll ask you to do it with written problems.
- But in programming problems, insisting on real instructions only (if at all feasible) means spending a lot of time solving the same problem (e.g., how to implement the relational operators for which there's no single branch instruction), and better to use the shortcuts and solve slightly-more-interesting problems!

  (One thing that's still true though: However strange it may seem to use bit shifting to multiply by a power of 2, it's almost surely faster, enough to use it.)

Slide 5

Example, Continued

- First look at previously-written C programs (nothing deep here?).
- Next start translating. Some parts mostly copy-paste from other programs. Trickiest part is the two `gcd()` functions, so write them "live" . . .

Slide 6

Minute Essay

- Have you tried out SPIM? using the virtual desktop, or on your own machine (what?).
- Questions about this example, or programming for SPIM in general?