Slide 1

Administrivia

- Due date change: Homework 1 due *Friday* at 5pm. I will send e-mail soon with information about coping with different editions of textbook(!).
- Reminder: Quiz 1 Friday. Ten minutes, open book/notes. Topics from chapter 1.
- Slide added to 2/04 notes for additional logical operations in textbook (exclusive or, nor).

Slide 2

Flow of Control — Review/Recap

- So far we know how to do (some) arithmetic, move data into and out of memory. What about if/then/else, loops?
- We need instructions that allow us to "make a decision" — `beq` ("branch if equal"), `bne` ("branch if not equal") and also ones that let us unconditionally "jump" (`j`):

```
beq     r1, r2, label
bne     r1, r2, label
j       label
```

  where `a` and `b` are registers, and `label` is a "label" (punt for now on how to turn that into ones and zeros).
- Simple example last time, using `goto`.

Slide 3

Another Flow of Control Example

- Of course, we don't usually have `go to` in C. More likely is this:

```
if (i == j)
        f = g + h
else
        f = g - h
```

- What to do with this? Rewrite using `go to` . . .

Slide 4

Loops

- Do we have enough to do (some kinds of) loops? Yes — example:

```
Loop:   g = g + A[i];
        i = i + j;
        if (i != h) goto Loop:
```

assuming we're using `$s1` through `$s4` for `g`, `h`, `i`, `j`, and `$s5` for the address of A.

- Or how about something that looks more like normal C?

```
while (A[i] == k) {
        i = i + j;
```

Slide 5

More Flow of Control

- We can do if/then/else and loops, but only if condition being tested is equals / not equals.
- So, we need instructions such as `blt`, `ble`, right?
- But those are difficult to implement well, so instead MIPS has "set on less than":

  ```
  slt   r1, r2, r3
  ```

  which compares the contents of registers `r2` and `r3` and sets `r1` — 1 if `r2` is smaller, else 0.
- Also define that register 0 (`$zero`) always contains 0.
- Example — compile the following C:

  ```
  if (a < b) go to Less:
  ```

  assuming we're using `$s0`, `$s1` for `a`, `b`

Slide 6

More Flow of Control, Continued

- Do we have enough now? for all six possible C comparisons of integers? Yes . . .
- One more C flow-of-control construct we could talk about — `switch` — but defer for now.
- But we do want to talk about one more HLL feature . . .

Slide 7

Minute Essay

- TBA