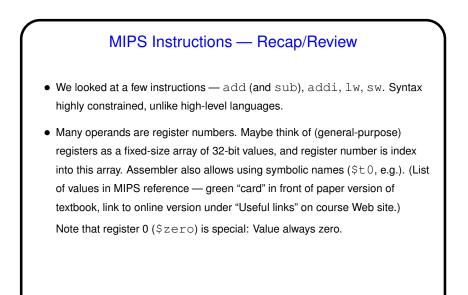
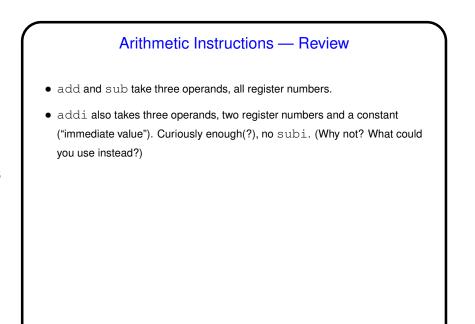


Minute Essay From Last Lecture • (Review.) Most people got it right; a few didn't.



## Registers and Variables (Recap/Review)

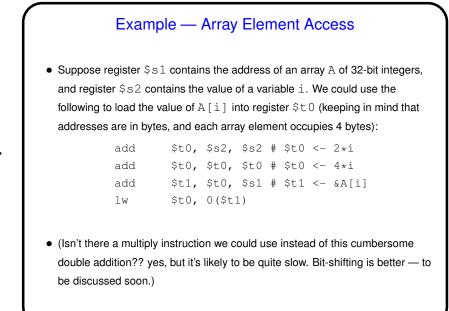
- Examples in textbook and in class talk about registers being associated with variables.
- The idea is more or less this: In MIPS, can only do arithmetic on values in registers. So if compiling from a high-level language, to do arithmetic on variables, have to first load values into registers, then do arithmetic, then store the results back.
- Repeated loads/stores can be inefficient, though, so "good" compilers typically try to associate a register with each variable and do loads/stores only when necessary. (If more variables than registers? then use registers for most-frequently-used variables, do more loads/stores.)



## Load/Store Instructions — Review

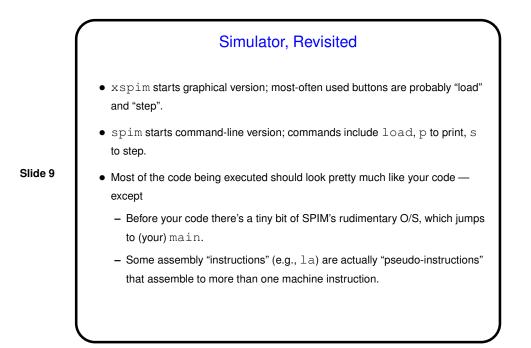
• Load and store instructions take two operands, one a register to load into / store from, and one specifying address in terms of register containing base address and displacement (constant).

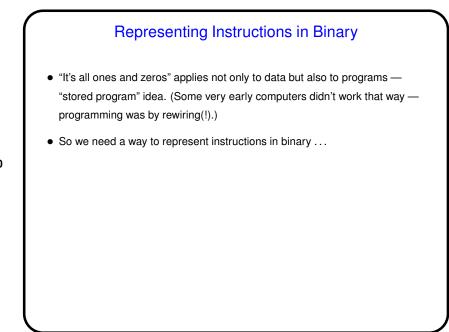
- Fixed displacement isn't maybe ideal for all situations (e.g., array element), but simple, and displacement useful for addressing element of, say, a C struct.
- (How then to address array element? compute address by computing displacement and adding to base address. Example on next slide.)

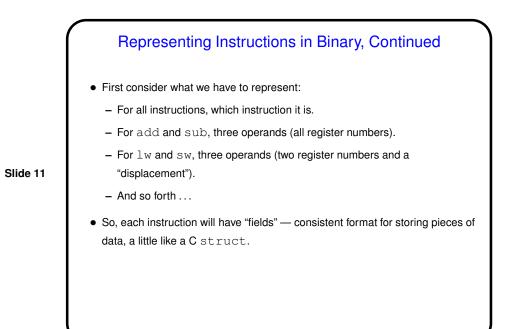


MIPS Assembly Language Program Structure
(Look again at starter.s under "sample programs" on course Web site.)
Overall structure mixes instructions and "directives" (things that start with .). Programs typically have two sections, one for code (starting with .text directive) and one for data (starting with .data).
For now, ignore "opening linkage" and "closing linkage". Most of the rest should seem at least sort of plausible?

Slide 7





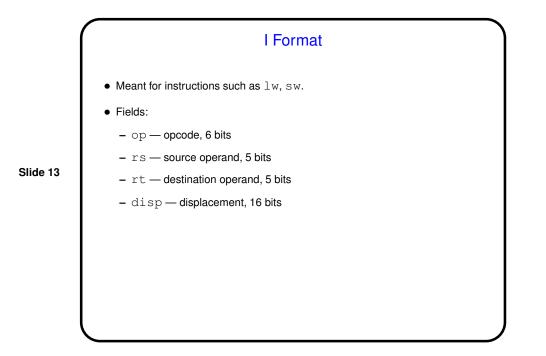


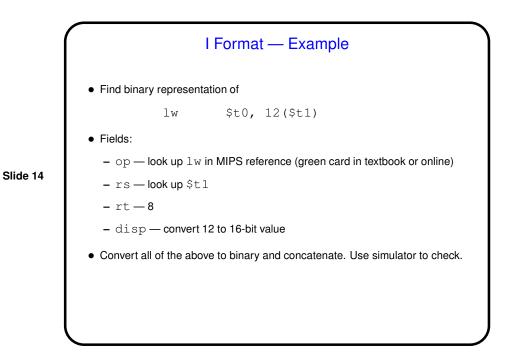
## Representing Instructions in Binary, Continued

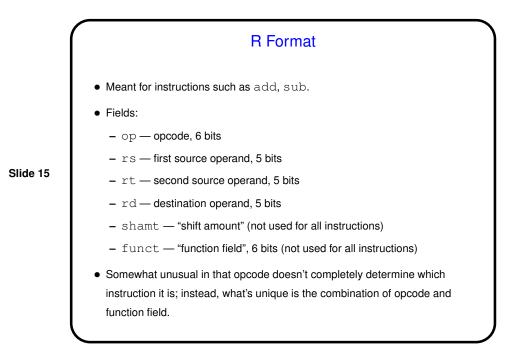
• So, can we use the same format for all instructions? Some data ("which instruction") is common to all, but operands may need to be different.

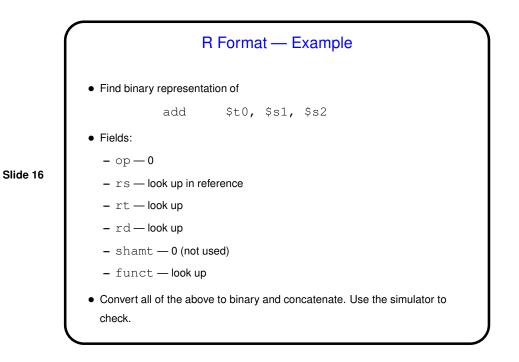
• Can we / should we make all instructions the same length? For MIPS, yes (other architectures differ), and then define different ways of dividing up the length — "formats".

(Another way to say it, maybe; In MIPS all machine-language instructions are 32 bits. Of those, 6 are always something identifying which instruction; the remaining bits are split up differently for different kinds of instructions.)

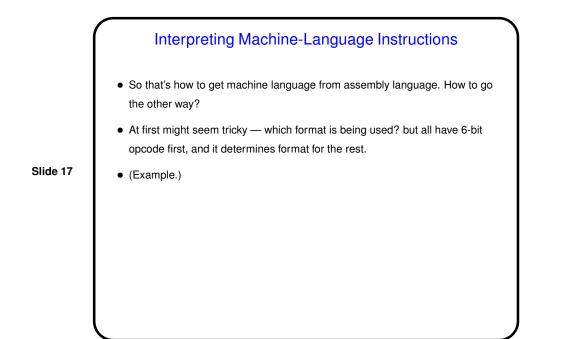


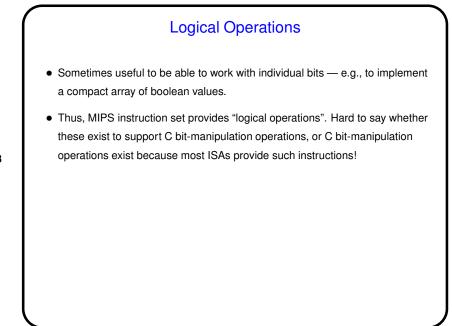


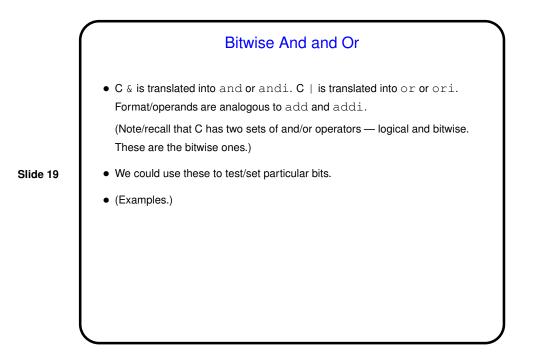


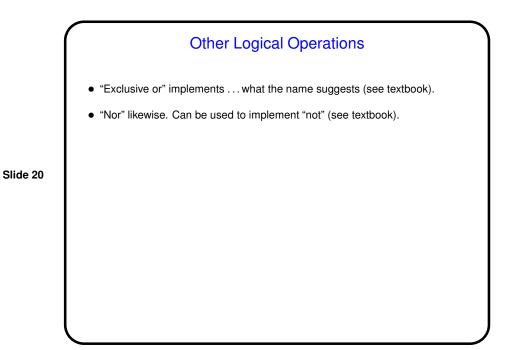


8

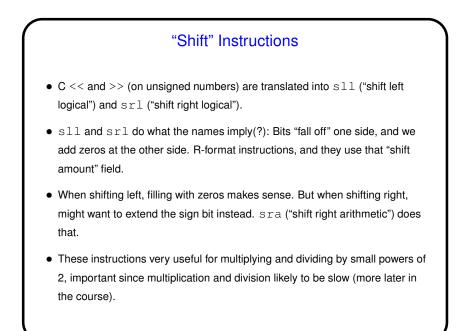


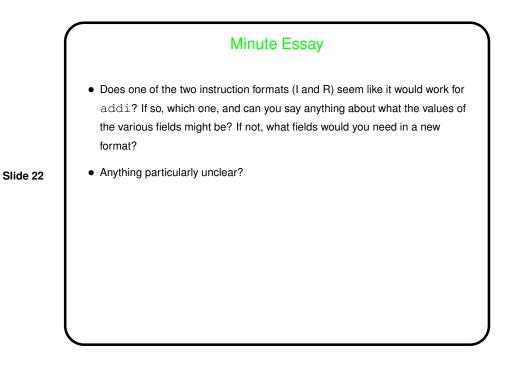






10





11

## Minute Essay Answer

• I format works — the operands of addi are two register numbers and a 16-bit constant value, same as lw and sw. Like those two instructions, it has "source" and "destination" registers, which can go in those two fields, and a 16-bit immediate value that can go in the field used for displacement in the load/store instructions.