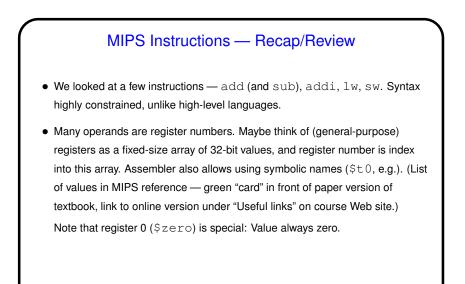


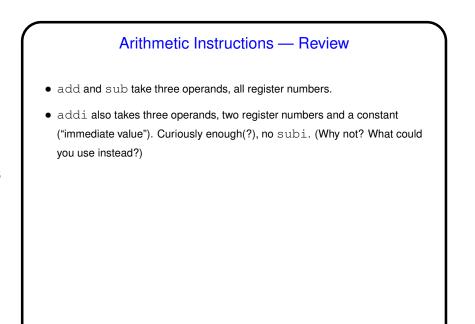
Slide 2

Minute Essay From Last Lecture
Pretty much everyone got it right.
(Review answers?)



Slide 4

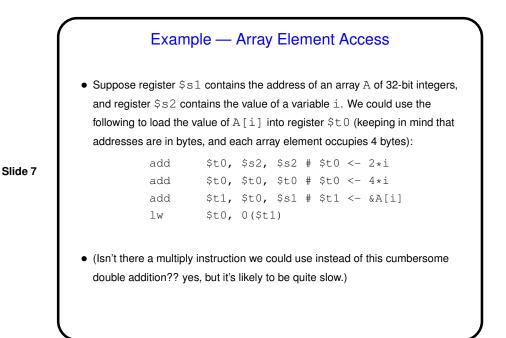
Registers and Variables Examples in textbook and in class talk about registers being associated with variables. The idea is more or less this: In MIPS, can only do arithmetic on values in registers. So if compiling from a high-level language, to do arithmetic, then store the results back. Repeated loads/stores 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.)

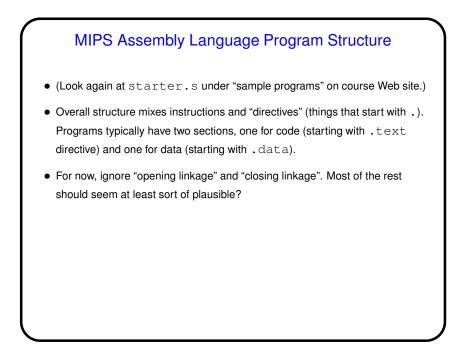


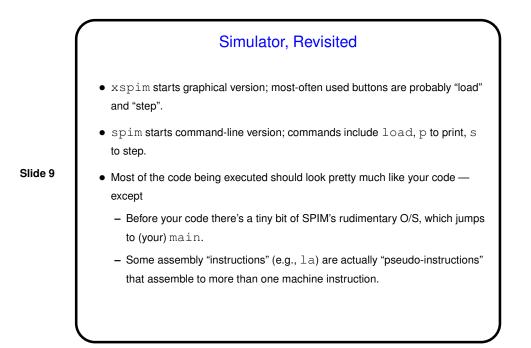
Load/Store Instructions — Review

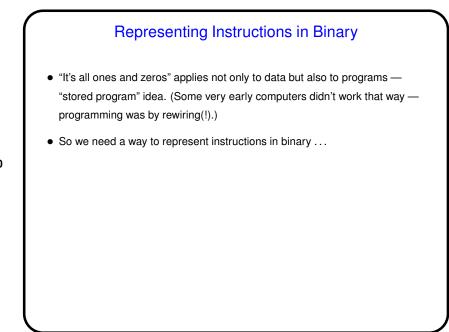
• Load and store instructions take two operands, one a register to load into / store from, and one specifying address in terms of register containing base address and displacement (constant).

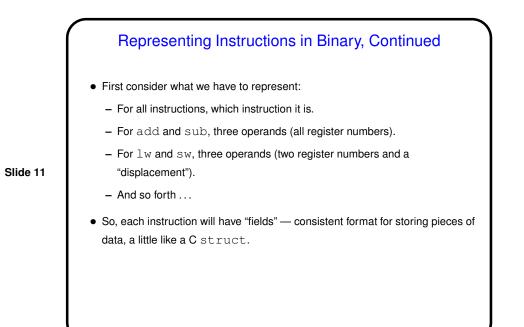
- Fixed displacement isn't maybe ideal for all situations (e.g., array element), but simple, and displacement useful for addressing element of, say, a C struct.
- (How then to address array element? compute address by computing displacement and adding to base address. Example on next slide.)

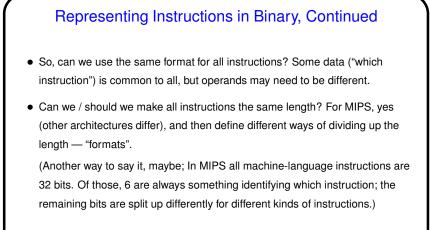


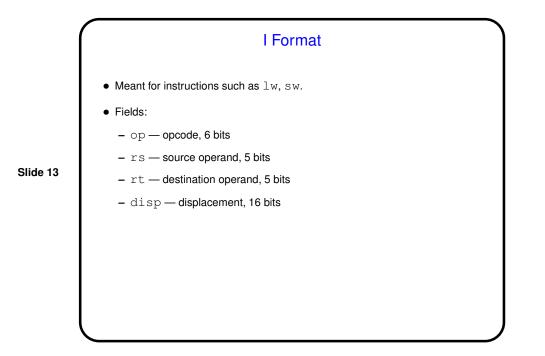


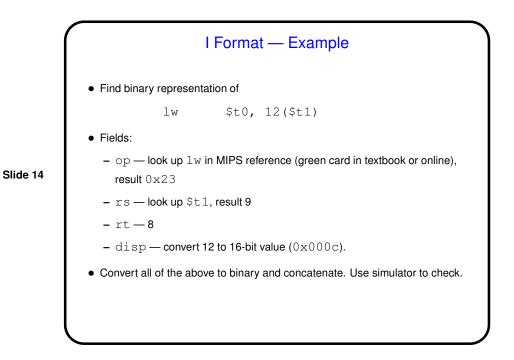


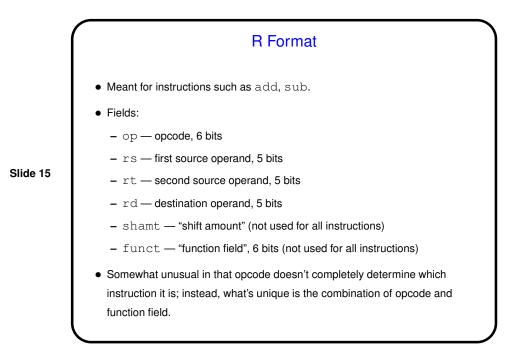


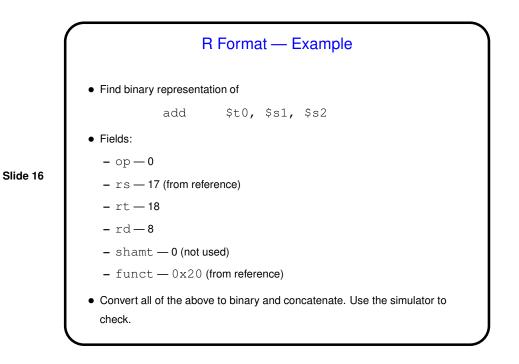


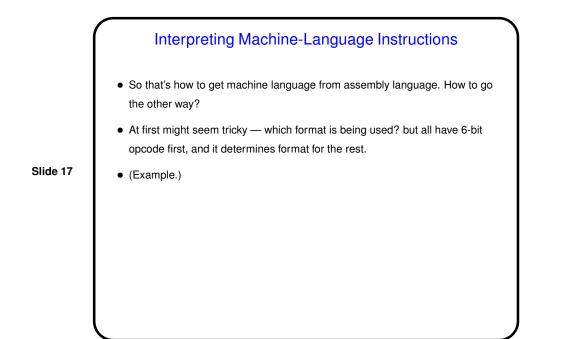


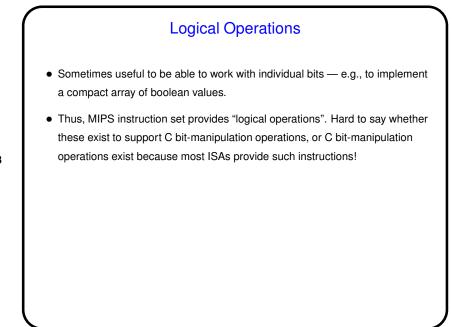


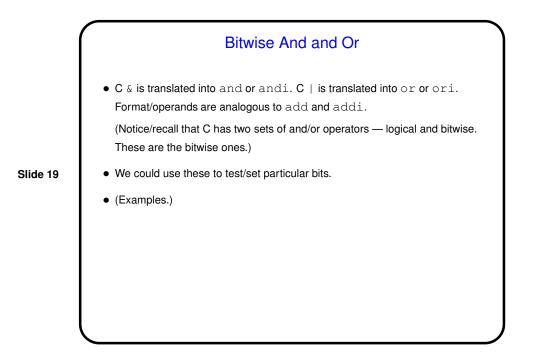


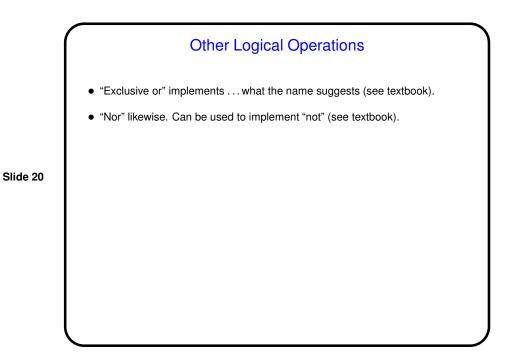


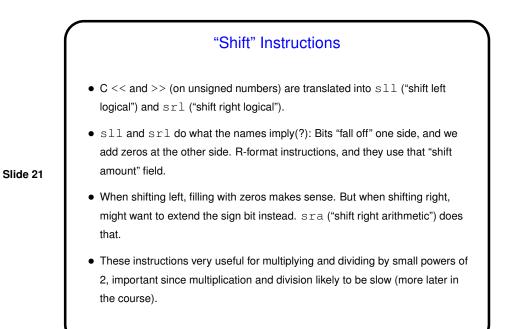








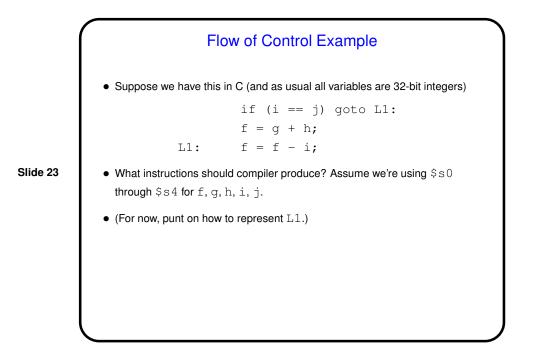


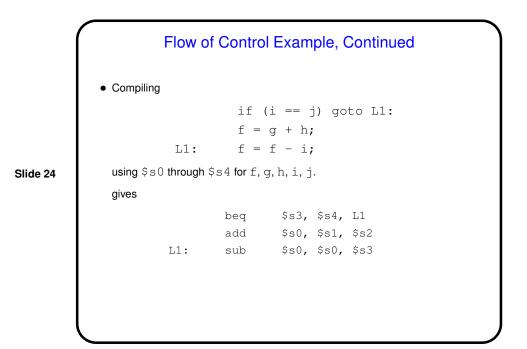


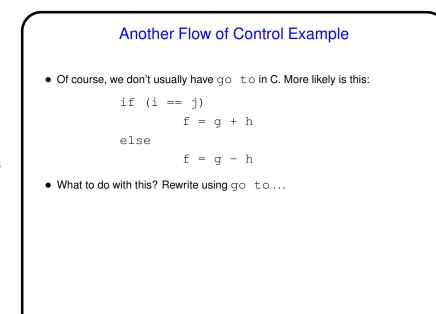
Flow of Control
So far we know how to do (some) arithmetic, move data into and out of memory. What about if/then/else, loops? (See sidebar on p. 90 for early

commentary on conditional execution.)We need instructions that allow us to "make a decision". Perhaps surprisingly,

- MIPS provides only two: beq ("branch if equal"), bne ("branch if not equal").
- Illustrate with an example ...



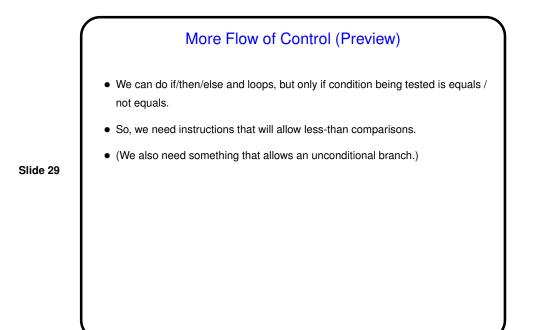


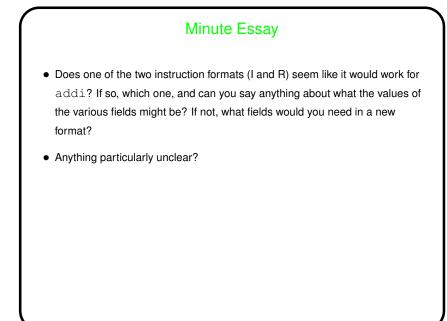


Another Flow of Control Example • Rewriting if (i == j) f = g + helse f = g - hSlide 26 gives if (i != j) goto Else: f = g + hgoto End: Else: f = g - hEnd: and then we can continue as before (punt for now on how to do unconditional goto).

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.
(This time we'll use sll rather than two adds to multiply i by 4.)

	Loops — Example Continued						
	 Result 						
Slide 28	Loop:	sll	\$t1,	\$s3,	2	#	\$t1 <- 4*i
		add	\$t1,	\$t1,	\$s5	#	\$t1 <- & of A[i]
		lw	\$t0,	0(\$t1)		#	\$t0 <- A[i]
		add	\$s1,	\$s1,	\$t0	#	g = h + A[i]
		add	\$s3,	\$s3,	\$s4	#	i = i + j
		bne	\$s3,	\$s2,	Loop	#	if (i!=j) goto Loop
	l						





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.