

Slide 1

One More Thing About Performance — Amdahl's Law

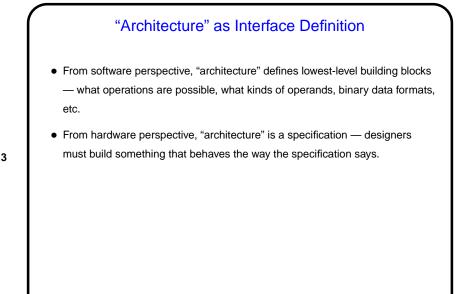
- Parallel-computing version: Can define "speedup" gained by using *P* processors as ratio of execution time using 1 processor to execution time using *P* processors. (So, in a perfect world it would be *P*).
- But most "real programs" have some parts that have to be done sequentially. Gene Amdahl (principal architect of early IBM mainframe(s)) argued that this limits speedup — "Amdahl's Law":

If γ is the "serial fraction", speedup on P processors is (at best — this ignores overhead)

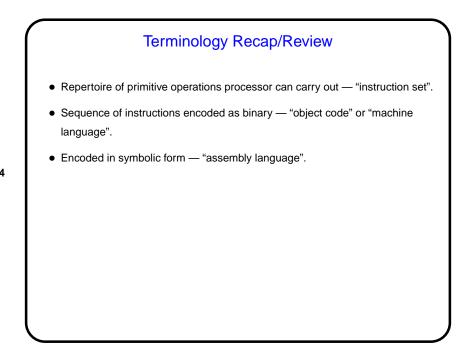
$$S(P) = \frac{1}{\gamma + \frac{1-\gamma}{P}}$$

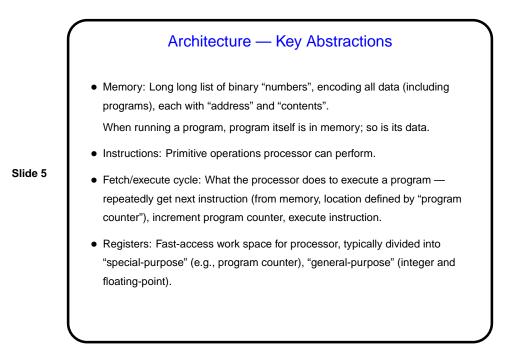
and as P increase, this approaches $\frac{1}{\gamma}$ — upper bound on speedup.

• Textbook points out that this is more broadly applicable!



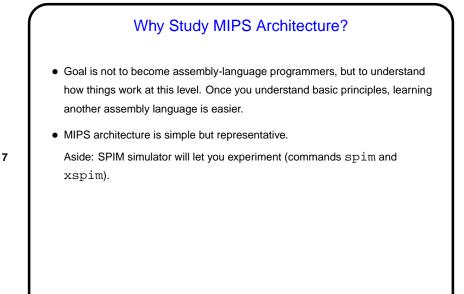
Slide 3





Design Goals for Instruction Set

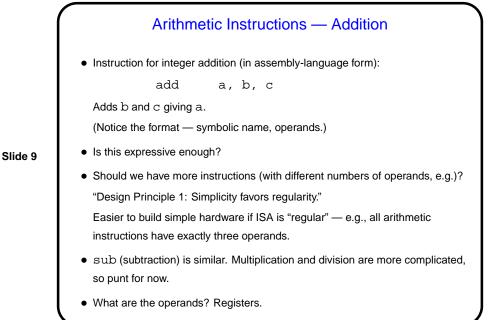
- From software perspective expressivity.
- From hardware perspective good performance, low cost.

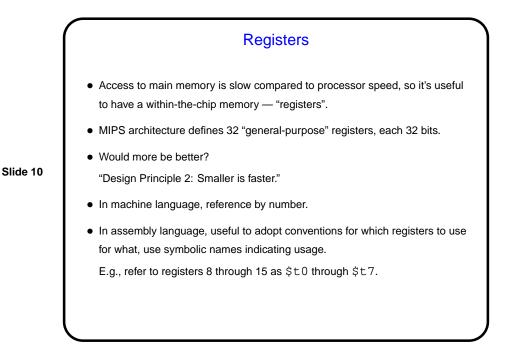


Slide 7

A Bit About Assembly Language Syntax Syntax for high-level languages can be complex. Allows for good expressivity, but translation into processor instructions is complicated.

• Syntax for assembly language, in contrast, is very simple. Less expressivity but much easier to translate into (binary form of) instructions.



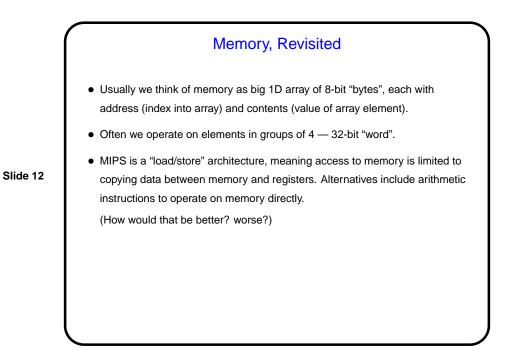


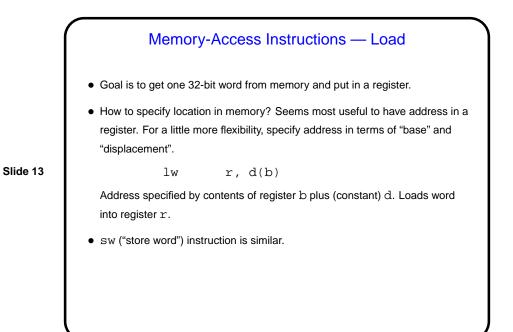
Example

• Suppose we have this in C

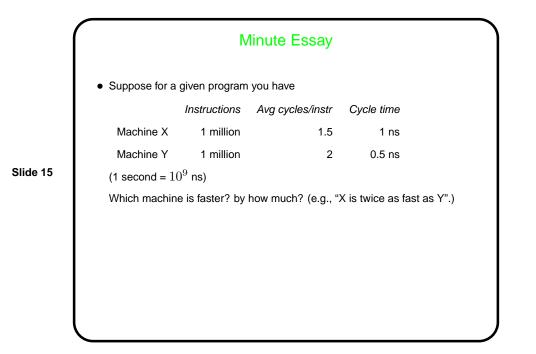
f = (g + h) - (i + j)

• What instructions should compiler produce? Assume we're using \$s0 for f, \$s1 for g, \$s2 for h, \$s3 for i, \$s4 for j.





Example • Suppose we have this in C g = h + a[8]; • What instructions should compiler produce? Assume we're using \$\$3 for starting ("base") address of a, \$\$2 for h, \$\$1 for g.



 Minute Essay Answer

 • time for X = $10^6 \times 1.5 \times 10^{-9} = 1.5 \times 10^{-3}$

 time for Y = $10^6 \times 2 \times 0.5 \times 10^{-9} = 10^{-3}$

 so Y is 1.5 times as fast as X

 Slide 16