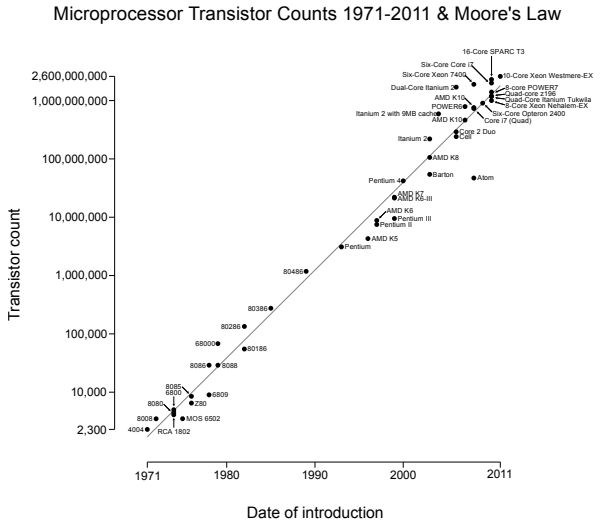
CS 6534: Tech Trends / Intro

Charles Reiss

24 August 2016

Moore's Law



Wikimedia Commons / Wgsimon

Good Ol' Days: Frequency Scaling

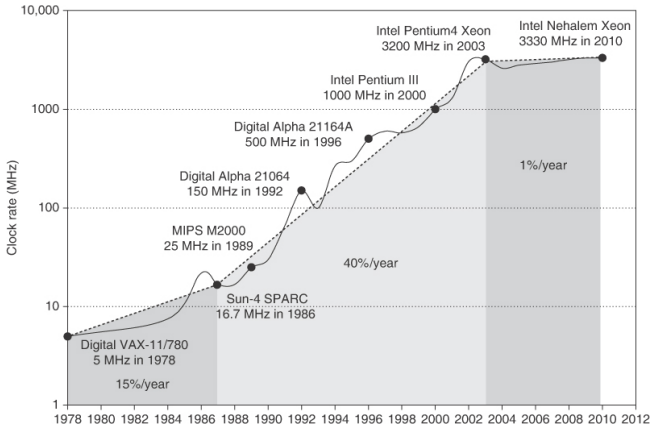


Figure 1.11 Growth in clock rate of microprocessors in Figure 1.1. Between 1978 and 1986, the clock rate improved less than 15% per year while performance improved by 25% per year. During the "renaissance period" of 52% performance improvement per year between 1986 and 2003, clock rates shot up almost 40% per year. Since then, the clock rate has been nearly flat, growing at less than 1% per year, while single processor performance improved at less than 22% per year.

H&P

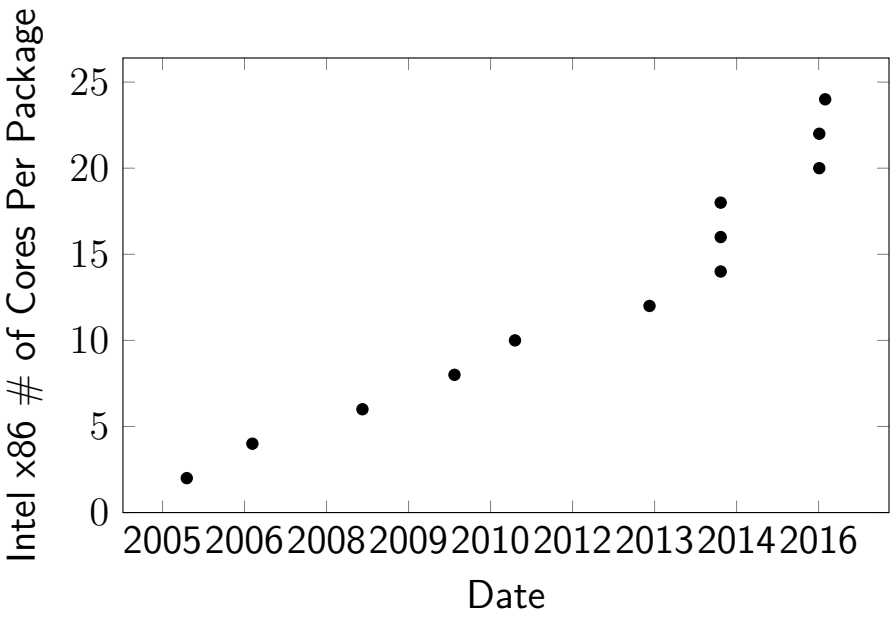
Copyright © 2011, Elsevier Inc. All rights Reserved.

The Power Wall

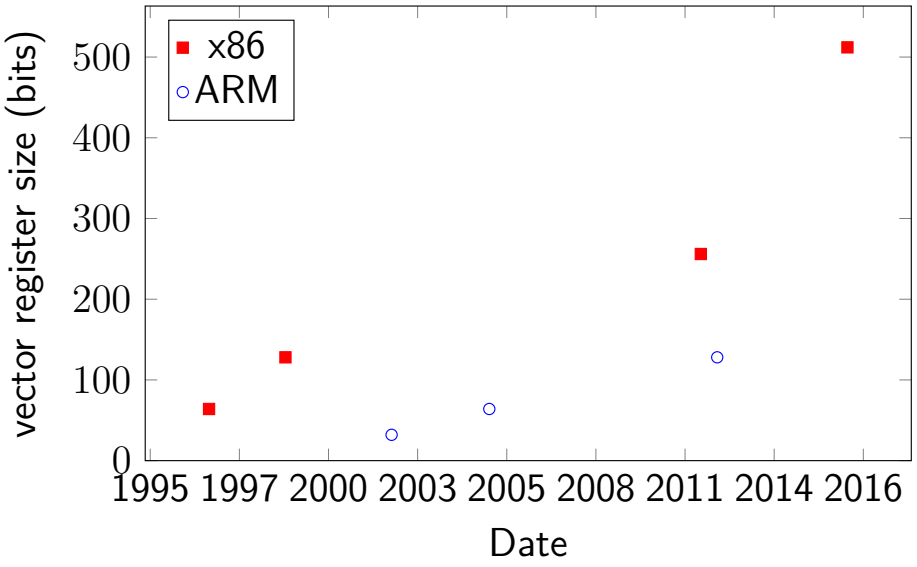
$$\text{Power} \sim \text{Switching Power} + \text{Leakage Power}$$

$$\text{Switching Power} \sim \text{Capacitance} \times \text{Voltage}^2 \times \text{Frequency}$$

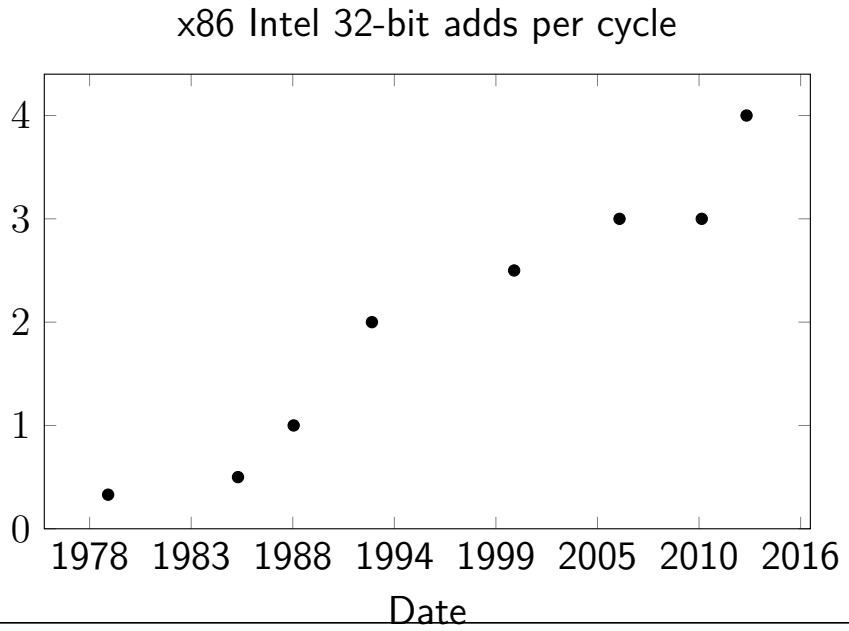
Increasing Parallelism: Cores



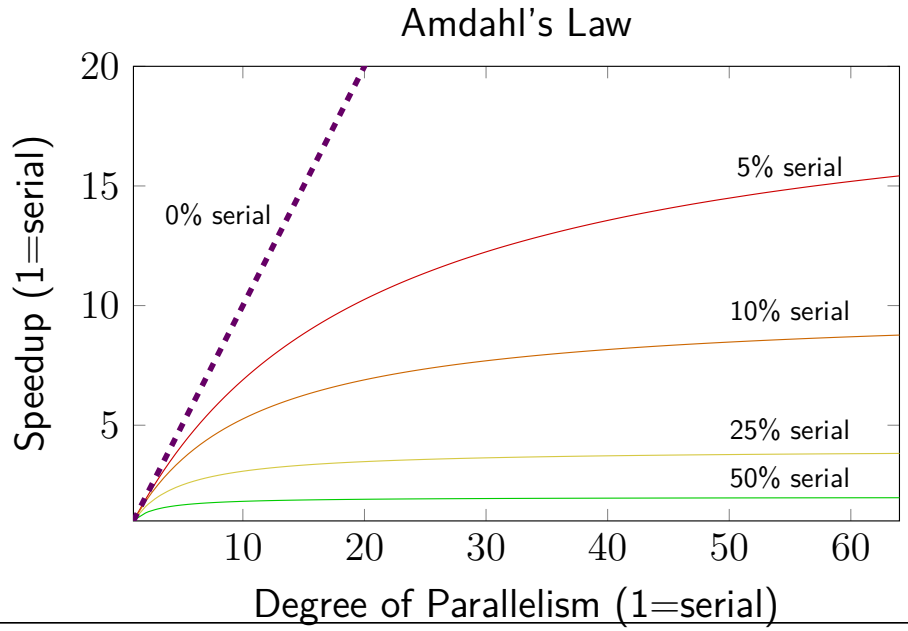
Increasing Parallelism: Vector width



Increasing Parallelism: ILP



Limits: Parallelism



Limits: Communication

[Balfour et al, "Operand Registers and Explicit Operand Forwarding", 2009.]

TABLE I
ENERGY CONSUMED BY COMMON OPERATIONS

Arithmetic Operations		Relative Energy
32-bit addition	520 fJ	1×
16-bit multiply	2,200 fJ	4.2×
General-Purpose Register File — 32 words (4R+2W)		
32-bit read	830 fJ	1.6×
32-bit write back	1,450fJ	2.8×
General-Purpose Register File — 32 words (2R+2W)		
32-bit read	800 fJ	1.5×
32-bit write back	1,380fJ	2.7×
Operand Register File — 4 words (2R+2W)		
32-bit read	120 fJ	0.2×
32-bit write back	540 fJ	1.0×
Forwarding Register — 1 word (1R+1W)		
32-bit forward	530 fJ	1.0×
Data Cache — 2K words (1R+1W) [8-way set-associative with CAM tags]		
32-bit load	10,100 fJ	19.4×
32-bit store	14,900 fJ	28.6×

[Malladi et al, "Towards Energy-Proportional Datacenter Memory with Mobile DRAM", 2012.]

DDR3 DRAM (32-bit read/write)

full utilization	2 300 000 fJ	4300×
low utilization	7 700 000 fJ	15000×

Increasing Efficiency: Specialization

Task/workload-specific coprocessors or instructions

Maybe reconfigurable?

Heterogeneous systems

different parts for different types of computation

Interlude: Logistics

Paper reviews — approx 2/class

Homeworks — programming assignments

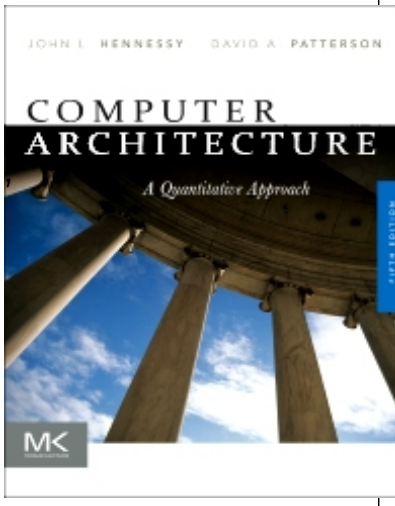
Exam — end of semester

Textbook?

Primarily paper readings

Classic + some newish papers

Reference: Hennessy and Patterson,
Computer Architecture:
A Quantitative Approach



Paper Reviews

What was your most significant insight from the paper?

What evidence does the paper have to support this insight?

What is the weakest part of the paper or how could it be approved?

What topic from the paper would you like to see discussed in class (if any)?

Paper Discussions

and not paper lectures.

Requires your cooperation.

Homeworks

Individual programming + writing assignments

First — on memory hierarchy — available now.

Second — to be announced — likely on superscalar

Third — to be announced — likely GPU programming

Homework 1

Description on course website (linked off Collab)

Memory system parameters by benchmarking

Example: 32K cache means accessing 32K repeatedly is faster than 128K repeatedly.

Homework 1: Disclaimer

This is probably hard

Modern memory hierarchies are complicated

Documentation is incomplete

Mainly looking for: measurement technique that 'should' work

If it doesn't, try to come up with good reasons why

Exam

There will be in final, probably in-class.

Cover material from papers, homeworks, discussions in class

Exceptions / etc.

Need accommodations — please ask

Disability accommodations — Student Disability Access Center

Asking Questions

Piazza (linked of Collab)

Office Hours:

Instructor	Lecturer Charles Reiss	TA Luowan Wang
Loation	Soda 205	TBA
Times	Monday 1PM–3PM Friday 10AM–noon	Tuesday 1PM–2PM

Email: `creiss@virginia.edu`

Survey

linked off Collab

anonymous

please do it

Preview of coming topics

Memory hierarchy

caching — review(?) and advanced techniques

homework 1

Pipelining

different parts of multiple instructions at the same time

more advanced topics: handling exceptions

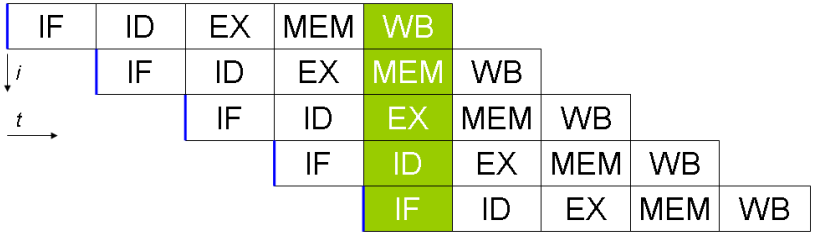
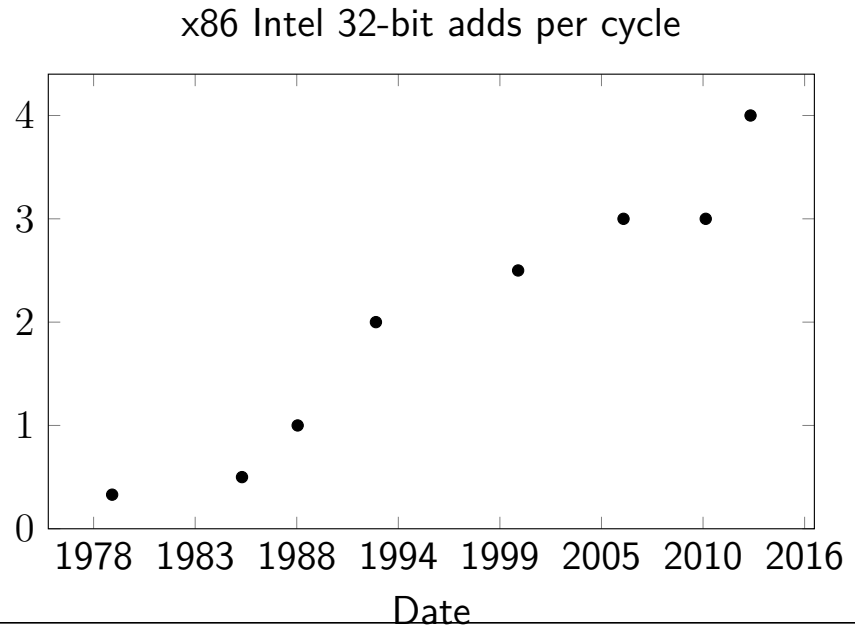


Image: Wikimedia commons / Poil

Increasing Parallelism: ILP



Beyond pipelining: Multiple issue

starting multiple instructions at the same time

allows cycles per instruction < 1

Beyond pipelining: Out-of-order

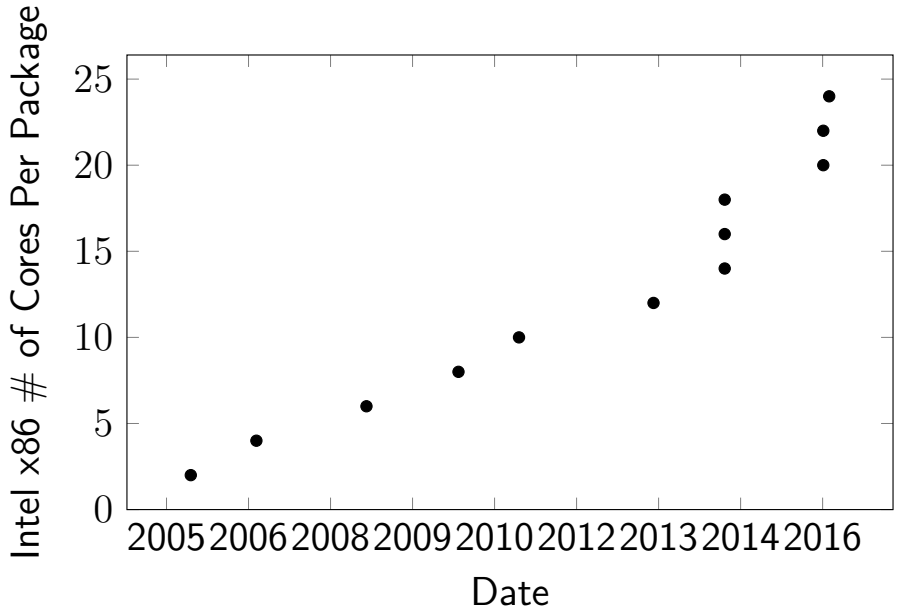
run next instruction despite stall of prior one

- slow cache
- read-after-write hazard
- ...

speculation — guess outcome of branch/load/etc.

- fix later if wrong

Increasing Parallelism: Cores



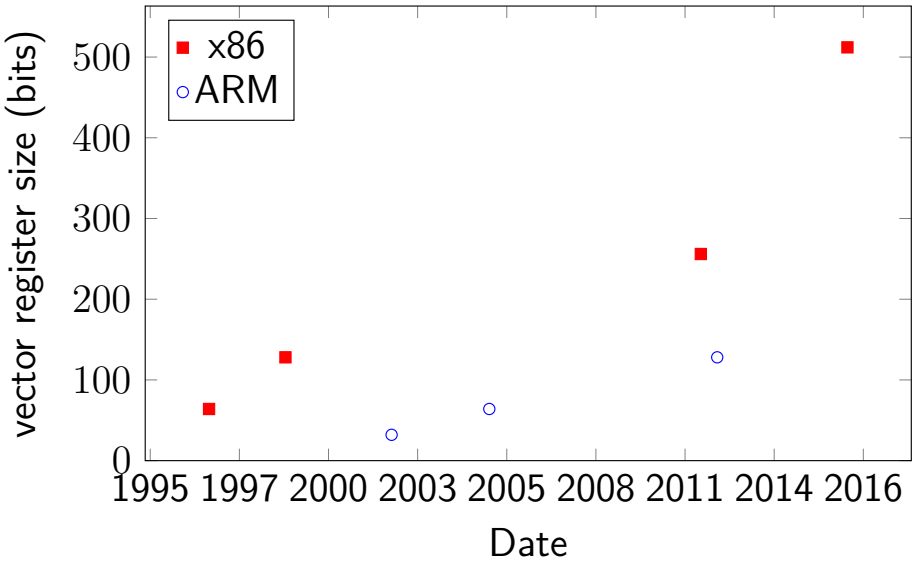
Multiprocessor/multicore

connecting processors together

shared memory — multiple threads

synchronization

Increasing Parallelism: Vector width



Vector/SIMD/GPUs

single instruction/multiple data

started with early supercomputers

basis of GPU programming model

Specialization

using custom chips (or circuits within chips)

reconfigurable processors (e.g. FPGAs)

Miscellaneous topics

hardware security

warehouse-scale computers

. . . depends on time

Suggestions?

Papers for Next Class

Alan Smith's review of caching in 1982

D. J. Bernstein's timing attack and suggestions to computer architects in 2005

Note: We're not reading this to learn about AES