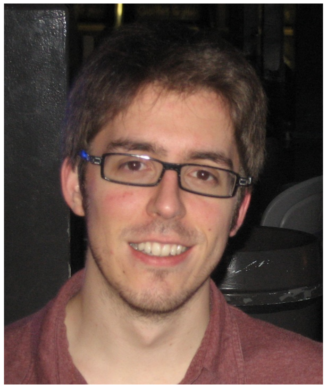
ECSE 425 Lecture 1: Course Introduction

Staff

- Instructor:
  - Brett H. Meyer, Professor of ECE
  - Email: brett dot meyer at mcgill.ca
  - Phone: 514-398-4210
  - Office: McConnell 525
  - OHs: M 14h00-15h00; R 11h00-12h00; or, by appointment

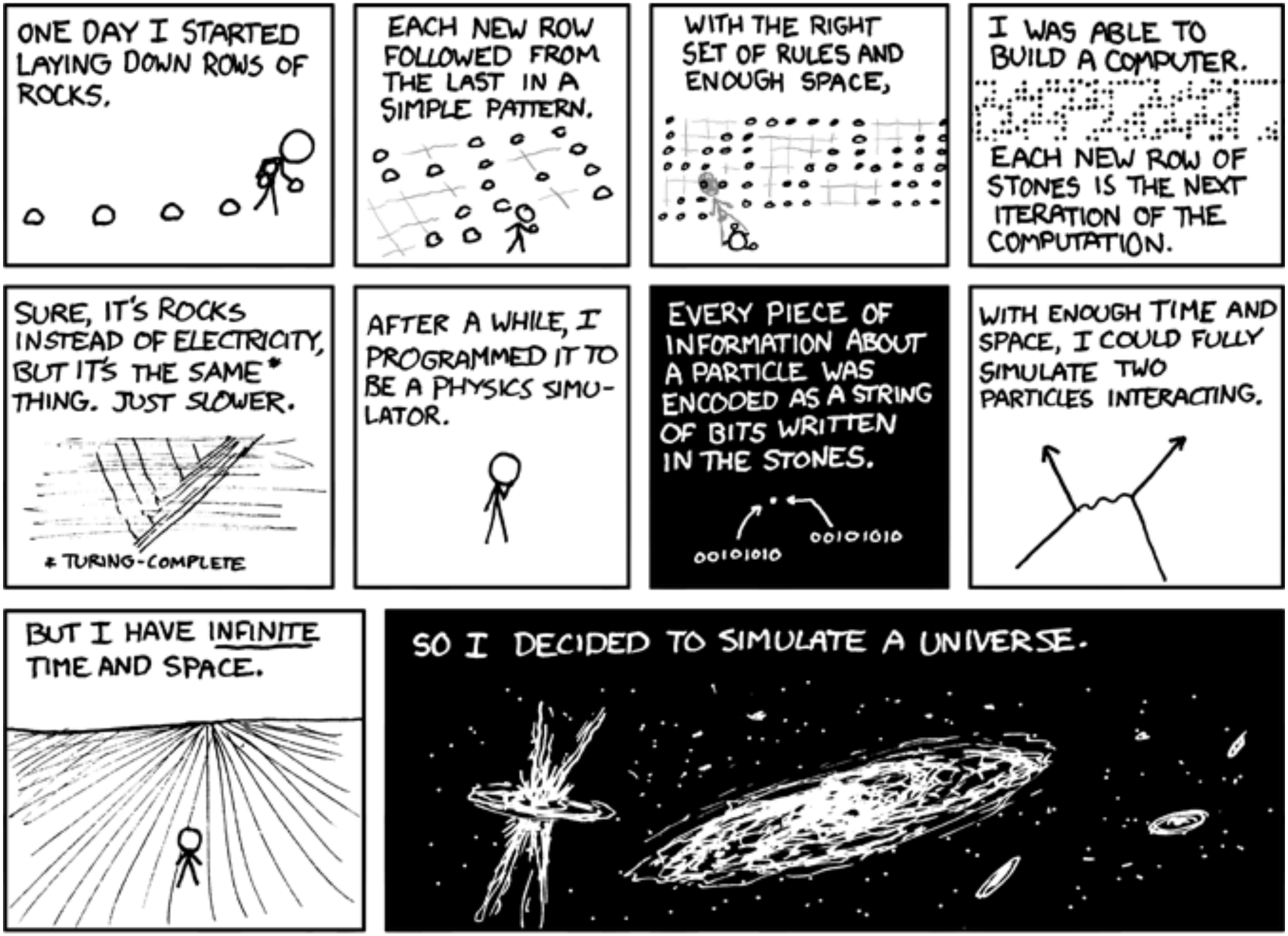
- Teaching Assistant:
  - Alexandre Raymond
  - Email: alexandre dot raymond at mail.mcgill.ca



What ECSE 425 is

- A course on the architecture of modern, high-performance computers
- What is computer architecture?
  - The art and science of selecting and interconnecting hardware components to create a computer that satisfies application requirements
- What we'll learn
  - How to organize a processor for **performance**
  - How other constraints (e.g., **cost**, **power**, etc.) influence computer design
  - How to judge the resulting trade-offs *quantitatively*

What ECSE 425 isn't

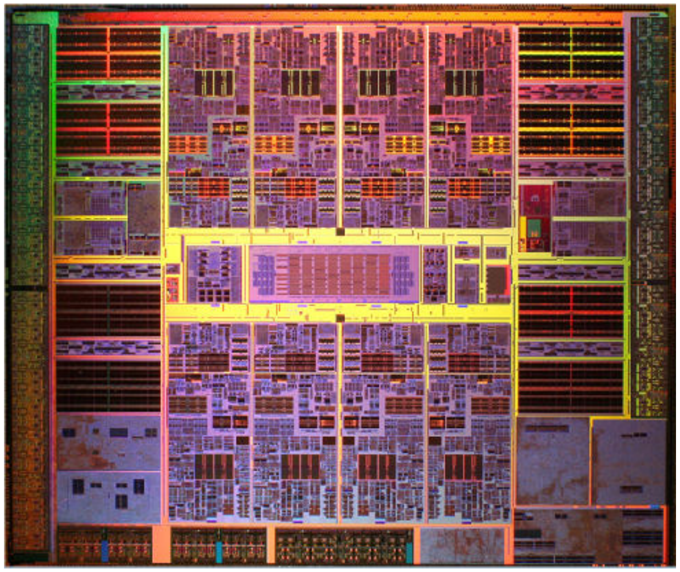


[Credit: xkcd.com]

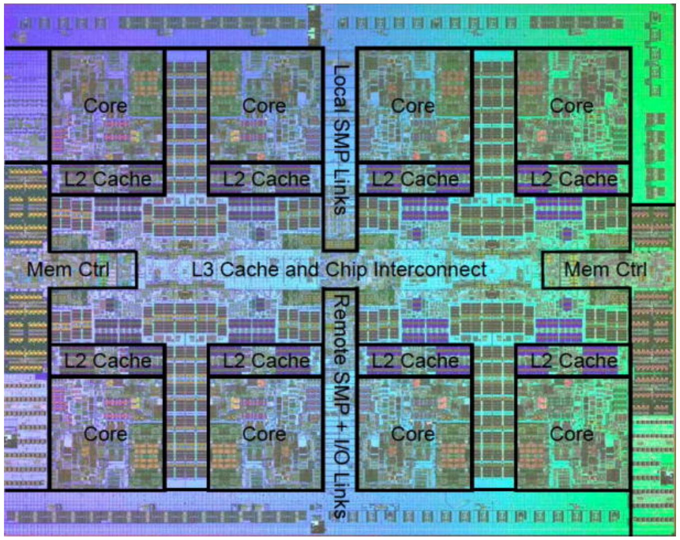
Why is Architecture Interesting?

- Comp. architecture is the heart of comp. engineering
- Bigger, stronger, faster!
  - Semiconductor industry relies on constant improvement
  - Architects must take growing resources and deliver!
- Clever design and organization
- Remarkable insights and algorithms

[Source: Sun; IBM]



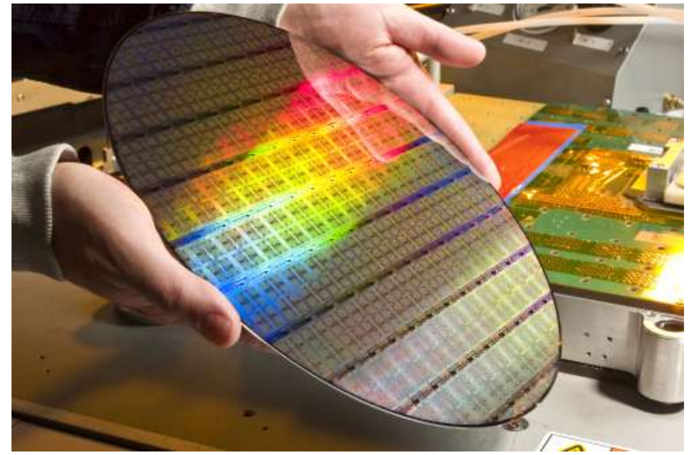
UltraSPARC T2, 64 threads, 342 mm^2



Power7, 32 threads, 576 mm^2

Why is Architecture Challenging?

- Constraints conspire to make continual improvement difficult
  - Cost: can't just make chips bigger
  - Power: can't just make chips do more in the same time
  - Reliability: must ensure chips do things right
- The effects of new architectural features are complex
- Evaluation is complex, too!
- A good idea isn't enough:
  - Implementable?
  - Used enough to change metrics?



[Source: IBM; AnandTech]

In This Course

- Fundamentals of computer design
- Pipelining
- Instruction-level parallelism (ILP)
- Memory hierarchy
- Multiprocessor architecture and thread-level parallelism (TLP)

Material based on H&P, 4th edition

Grading

- 6 homework assignments (10%)
  - Pencil and paper, some programming
- 1 project (30%)
  - Evaluate architectural effects using SimpleScalar
  - Significant C/C++ programming
- 2 midterm exams (30%)
  - In class, closed book, 1 page of notes allowed
- 1 final exam (30%)
  - Closed book, 2 pages of notes allowed

Homework

- Distributed on the course website
  - http://www.info425.ece.mcgill.ca
- Due at the beginning of class
- No credit for late work!
  - Without prior permission, of course
  - And …

Project

- Simulate the effect of architectural changes to a superscalar, out-of-order processor
- Simulation framework
  - SimpleScalar
  - Written in C
  - Evaluation using real benchmarks
- Work in pairs; consider early who to work with!
- Proposal ⇒ Implementation ⇒ Evaluation ⇒ Presentation ⇒ Report

Historical perspectives on processors

- Late 1970s: *"Birth of microprocessor"*
  - Single-chip processor, programmable controllers
- The decade of 1980s: *"Instruction set architecture"*
  - RISC (Reduced Instruction Set Computer)
  - Instruction pipelining, cache memories, compliers
  - Workstations
- The decade of 1990s: *"Instruction level parallelism"*
  - Superscalar, speculative micro-architectures
  - Low-cost desktop (super)computers
- The decade of 2000s: *"Multi-core era"*
  - Multi-core architectures, power constrained designs
  - Mobile/portable computing, large servers / data centers

Source: Stanford EE382a course slides by Prof. Christos Kozyrakis

Fundamentals of Computer Design

- Three principles
  - Make the common case fast (Amdahl's law)
  - Exploit parallelism (at all levels)
  - Exploit locality (caches / memory hierarchy)
- How should performance be measured?
- What other factors affect an architecture?
  - Latency
  - Cost
  - Power
  - Reliability

Pipelining

- Just like an assembly line
- Ideally: one instruction per clock cycle

	Clock number								
Instruction number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Instruction i	IF	ID	EX	MEM	WB				
Instruction $i + 1$		IF	ID	EX	MEM	WB			
Instruction $i + 2$			IF	ID	EX	MEM	WB		
Instruction $i + 3$				IF	ID	EX	MEM	WB	
Instruction $i + 4$					IF	ID	EX	MEM	WB

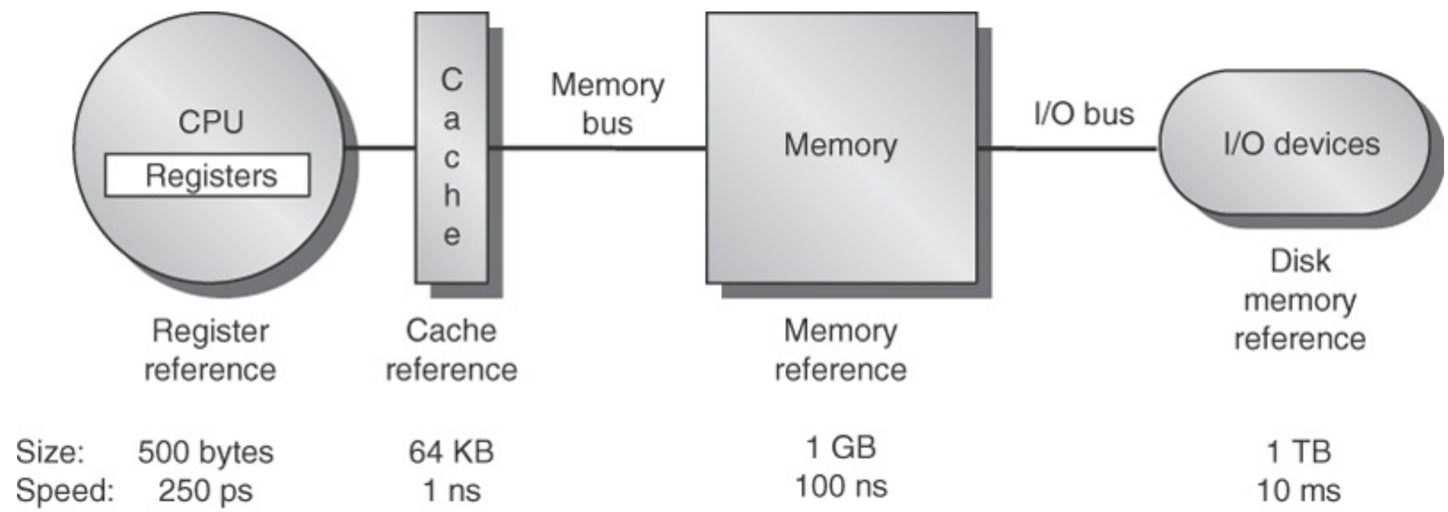
Reality: overheads; hazards, dependencies ⇒ stalls!

Instruction-Level Parallelism (ILP)

- *Goal*: exploit instruction independence to improve performance
- *Challenge*: hazards, dependencies ⇒ stalls
- Branch prediction: guess which execution path
- Instruction scheduling: reorder instructions
- Speculative instruction execution
- Superscalar instruction execution: allow more than one instruction to complete at a time

Memory Hierarchy

- Maintaining the illusion of fast, infinite memory
- Multiple-level memory system
- Cache performance and optimization
  - Size, internal organization, behavior, etc.
- Virtual memory organization



Multiprocessor Architecture

- ILP has limits; boost performance with thread-level parallelism! (TLP)
- Multiple-instruction, multiple-data machines (MIMD)
  – Concurrently execute multiple threads in parallel
  – New overheads: communication, synchronization
- Two classes (physical memory structure)
  – centralized memory multiprocessors
  – physically distributed-memory multiprocessors
- Two models (memory architecture and communications)
  – shared memory multiprocessors
  – message-passing multiprocessors

Next Time

- Fundamentals of Computer Design
  - Read Chapter 1!
- No class Monday
- First homework out on Wednesday
- First tutorial the end of next week