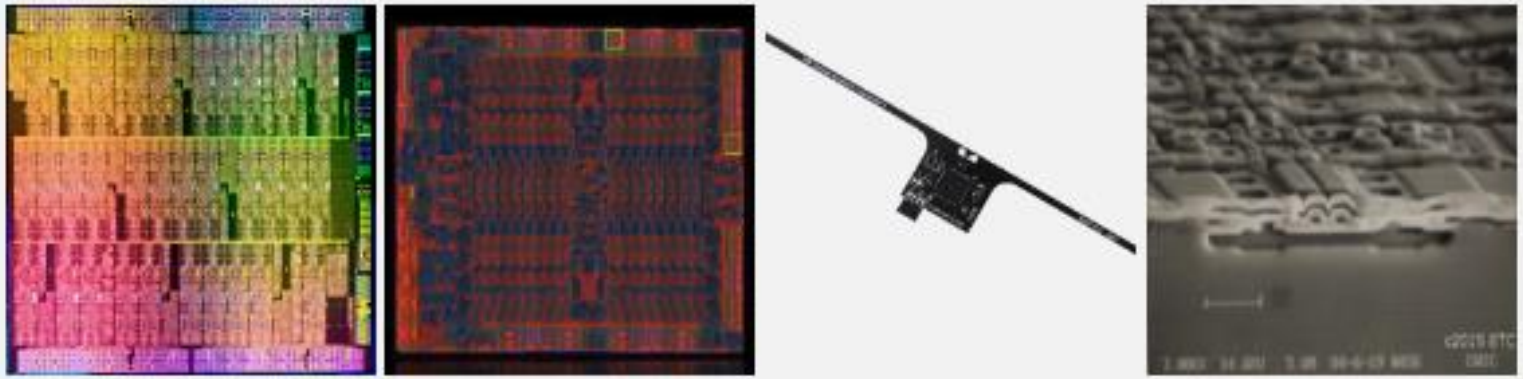
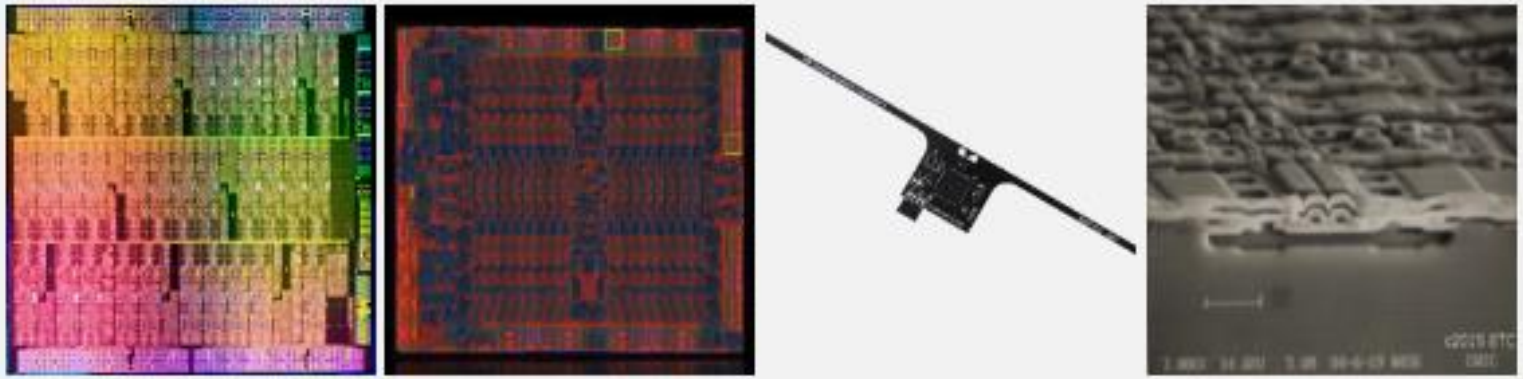
18-344: Computer Systems and the Hardware-Software Interface



Course Description

This course covers the design and implementation of computer systems from the perspective of the hardware software interface. The purpose of this course is for students to understand the relationship between the operating system, software, and computer architecture. Students that complete the course will have learned operating system fundamentals, computer architecture fundamentals, compilation to hardware abstractions, and how software actually executes from the perspective of the hardware software/boundary. The course will focus especially on understanding the relationships between software and hardware, and how those relationships influence the design of a computer system's software and hardware. The course will convey these topics through a series of practical, implementation-oriented lab assignments.

18-344: Computer Systems and the Hardware-Software Interface



Course Description

This course covers the design and implementation of computer systems from the perspective of the hardware software interface. The purpose of this course is for students to understand the relationship between the operating system, software, and computer architecture. Students that complete the course will have learned operating system fundamentals, computer architecture fundamentals, compilation to hardware abstractions, and how software actually executes from the perspective of the hardware software/boundary. The course will focus especially on understanding the relationships between software and hardware, and how those relationships influence the design of a computer system's software and hardware. The course will convey these topics through a series of practical, implementation-oriented lab assignments.

Akshitha Sriraman
(Call me “Akshitha”)

Pronouns: She/her
Assistant Professor
Research: Data center HW/SW systems

Course Staff & Logistics

- Prof. Akshitha Sriraman & Brandon Lucia
  - akshitha@cmu.edu, blucia@andrew.cmu.edu
  - CIC 4114, CIC 4107
  - http://akshithasriraman.com, https://brandonlucia.com/
- Teaching Assistants (18-344 Veterans):
  - Matthew Ngaw
  - Nathan Serafin
  - Liam Merino
  - Yufei Shi

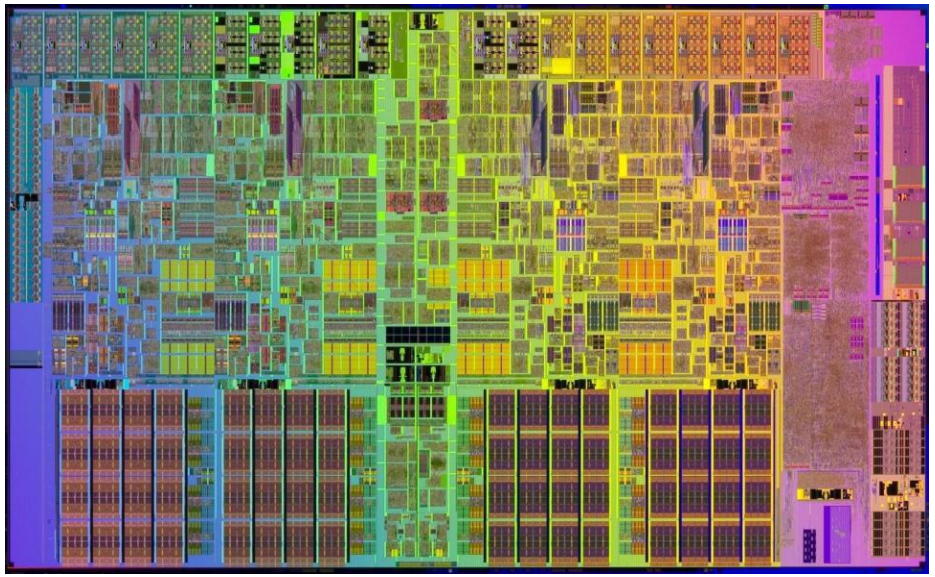
- Lecture: Monday & Wednesday, 3:30pm – 4:50pm in SH 234
  - Some lectures are designed to run short, some to run long. We may leave early, we may spill into next week.
- Recitation: Friday, 10-10:50am in MM A14
  - Project focused + Reinforcement

- Webpage: https://course.ece.cmu.edu/~ece344
- Office Hours (per website)
- 5 Labs (more later), 10 Homeworks
- Slack (for continuous Q&A)
- Late policy: -10% for each day late w/ 15 minute grace period for 11th hour submission problems. i.e., if assignment is due at 11:59:59pm ET Thursday, then at 12:15:00am ET on Friday your orig. score is multiplied by 0.9, at 12:00:00am ET on Saturday your orig. score is multiplied by 0.8, etc. Do your best to not get behind. There are times we have back-to-back labs being assigned. Having 2 labs at the same time will be very difficult to manage.

What are your expectations from this course?

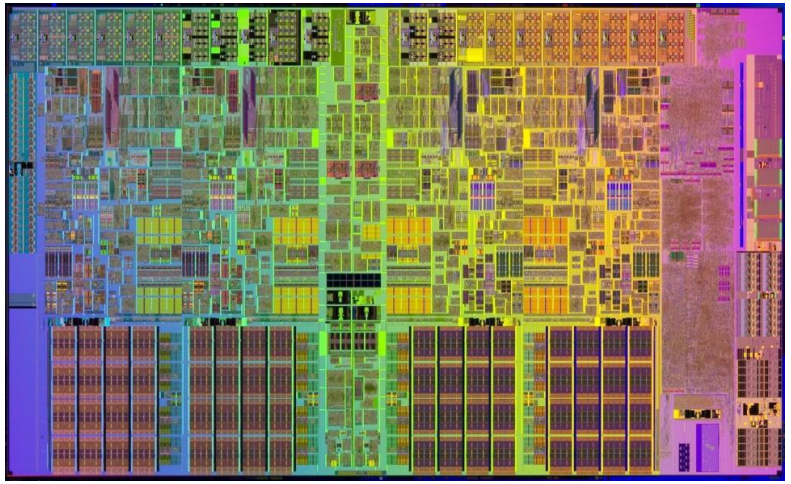
- Why did you decide to take this course?
- What are you excited about learning in this course?
- Is there anything that you're nervous about regarding this course?

What is this course about?



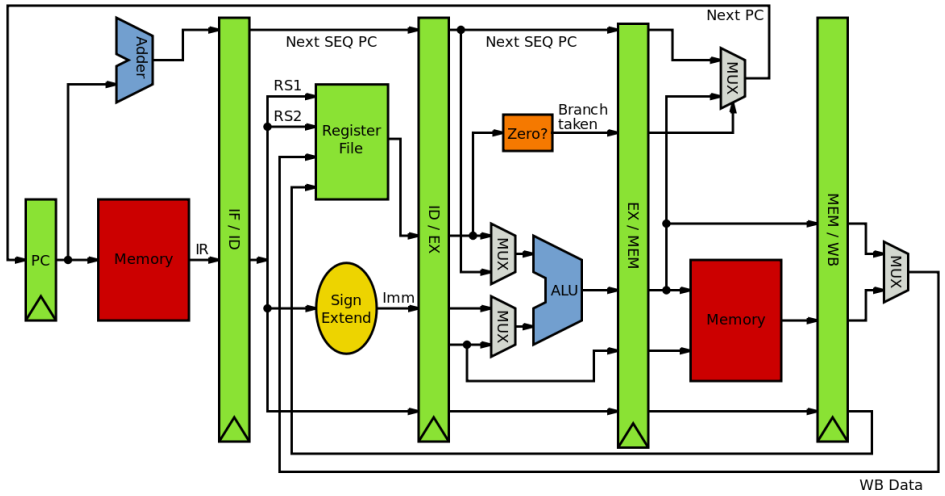
```
379  int walk_page_range(struct mm_struct *mm, unsigned long start,
380                  unsigned long end, const struct mm_walk_ops *ops,
381                  void *private)
382  {
383          int err = 0;
384          unsigned long next;
385          struct vm_area_struct *vma;
386          struct mm_walk walk = {
387                  .ops            = ops,
388                  .mm             = mm,
389                  .private        = private,
390          };
391
392          if (start >= end)
393                  return -EINVAL;
394
395          if (!walk.mm)
396                  return -EINVAL;
397
398          mmap_assert_locked(walk.mm);
399
400          vma = find_vma(walk.mm, start);
401          do {
402                  if (!vma) { /* after the last vma */
403                          walk.vma = NULL;
404                          next = end;
```

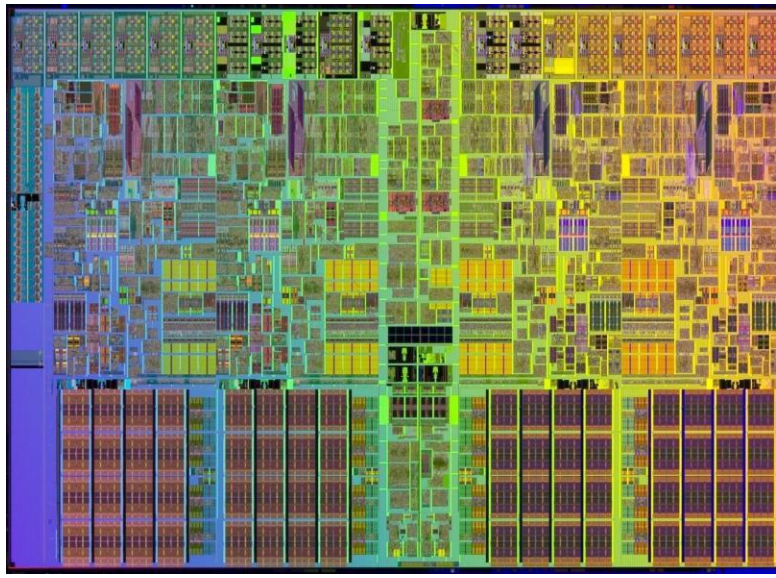
What is the hardware software boundary?



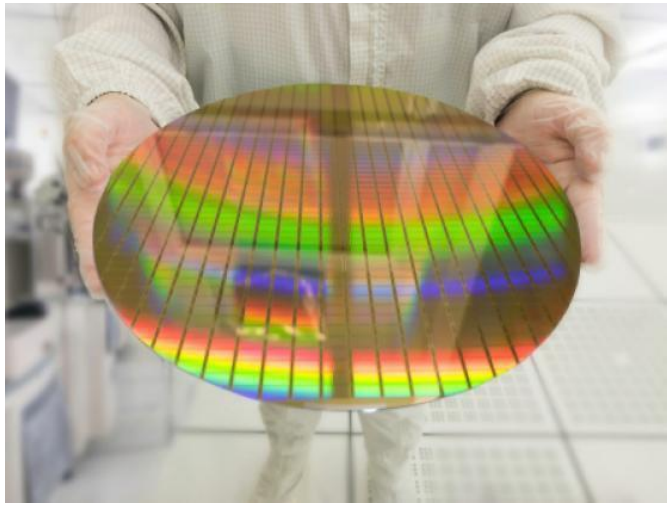
What is computer (μ)architecture?

Inst	Name	Opcode	funct3	funct7	Description (C)
add	ADD	0110011	0x0	0x00	rd = rs1 + rs2
sub	SUB	0110011	0x0	0x20	rd = rs1 - rs2
xor	XOR	0110011	0x4	0x00	rd = rs1 ^ rs2
or	OR	0110011	0x6	0x00	rd = rs1 | rs2
and	AND	0110011	0x7	0x00	rd = rs1 & rs2

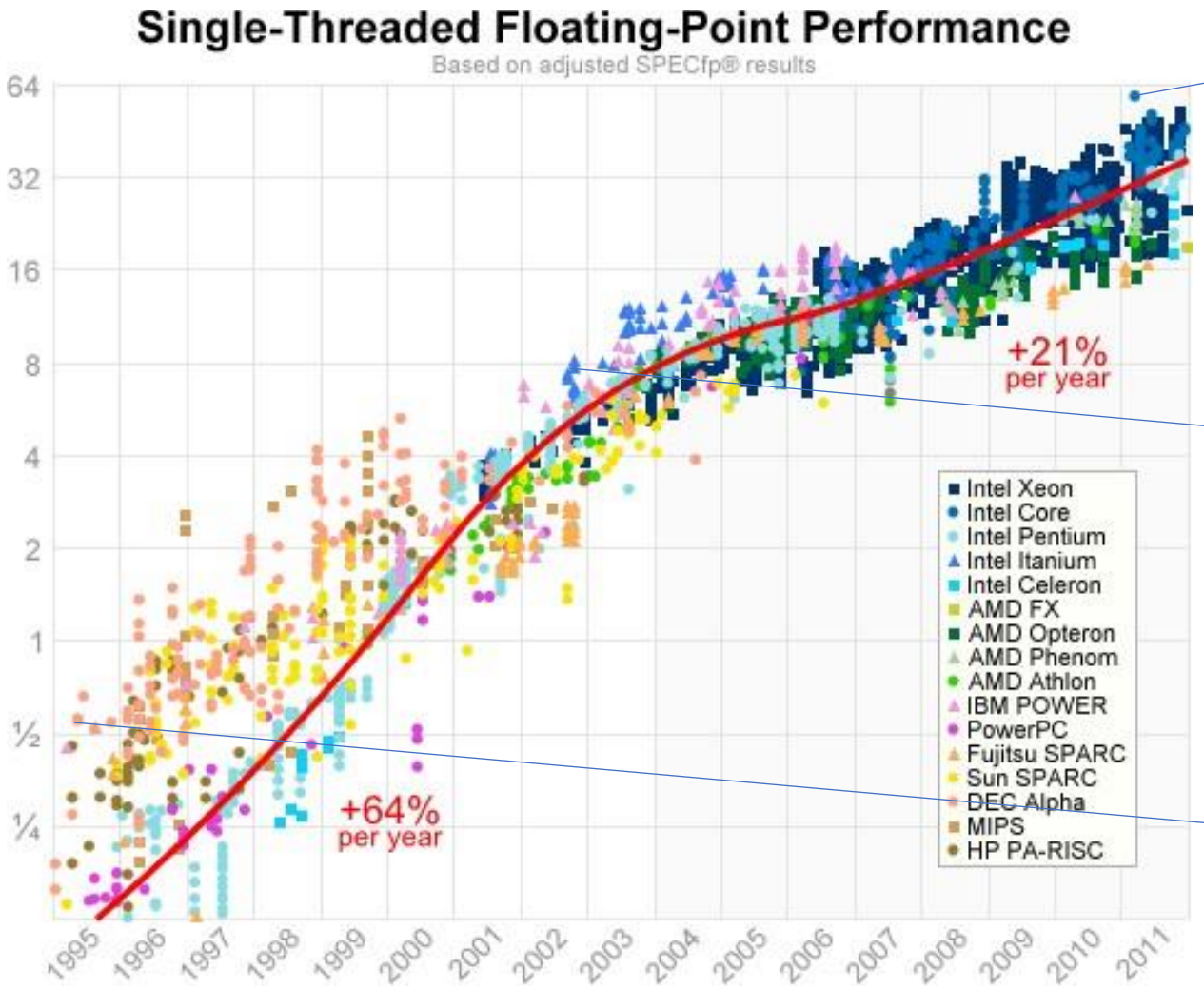


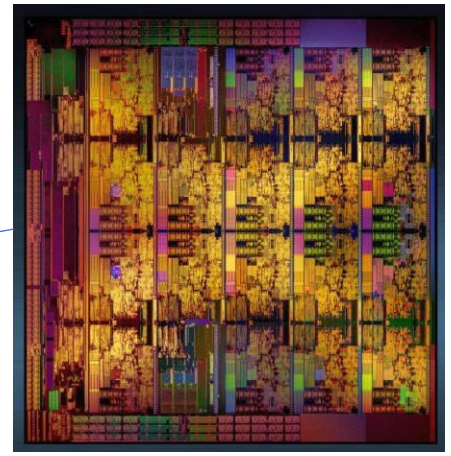


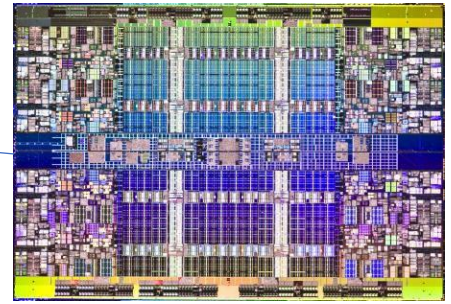
What constrains a computer system?

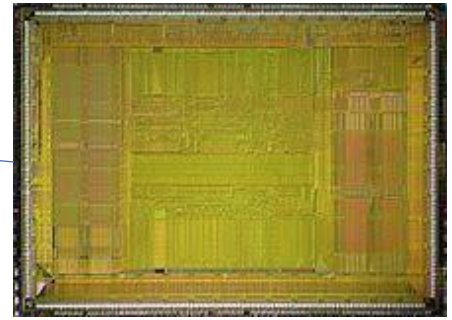


Why are these processors different?

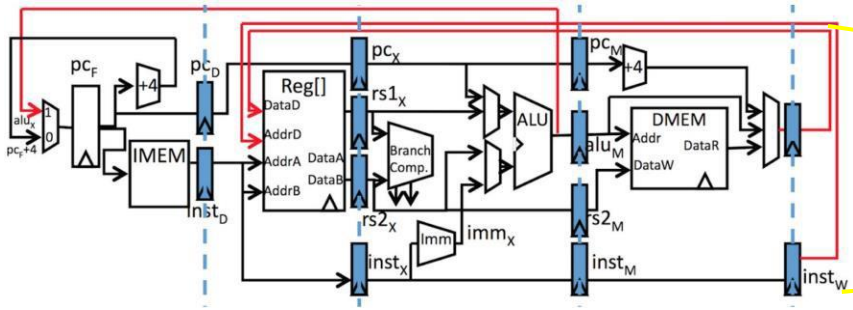


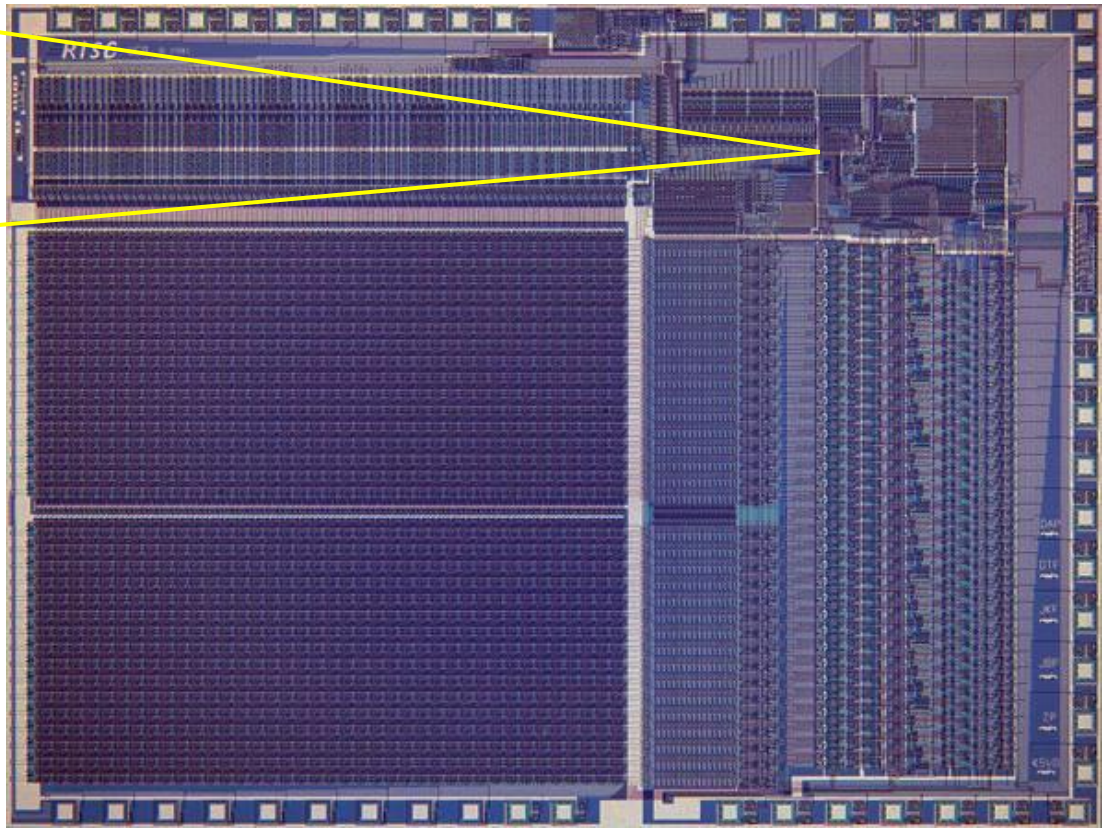


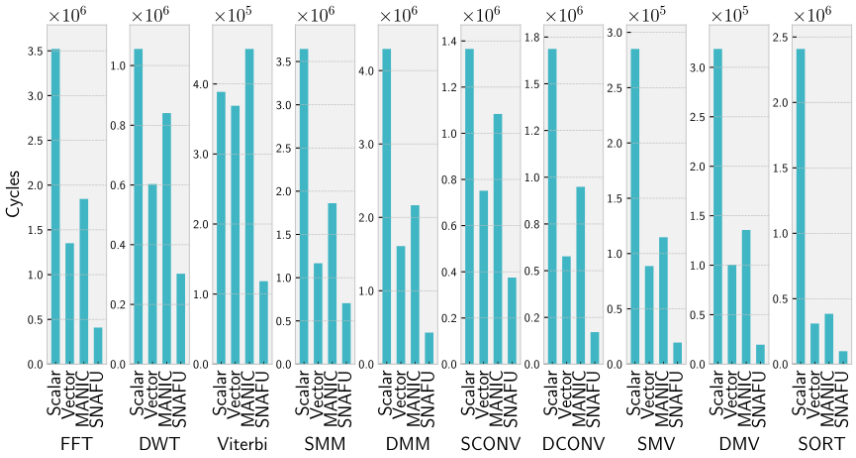




How do you measure a computer's performance?

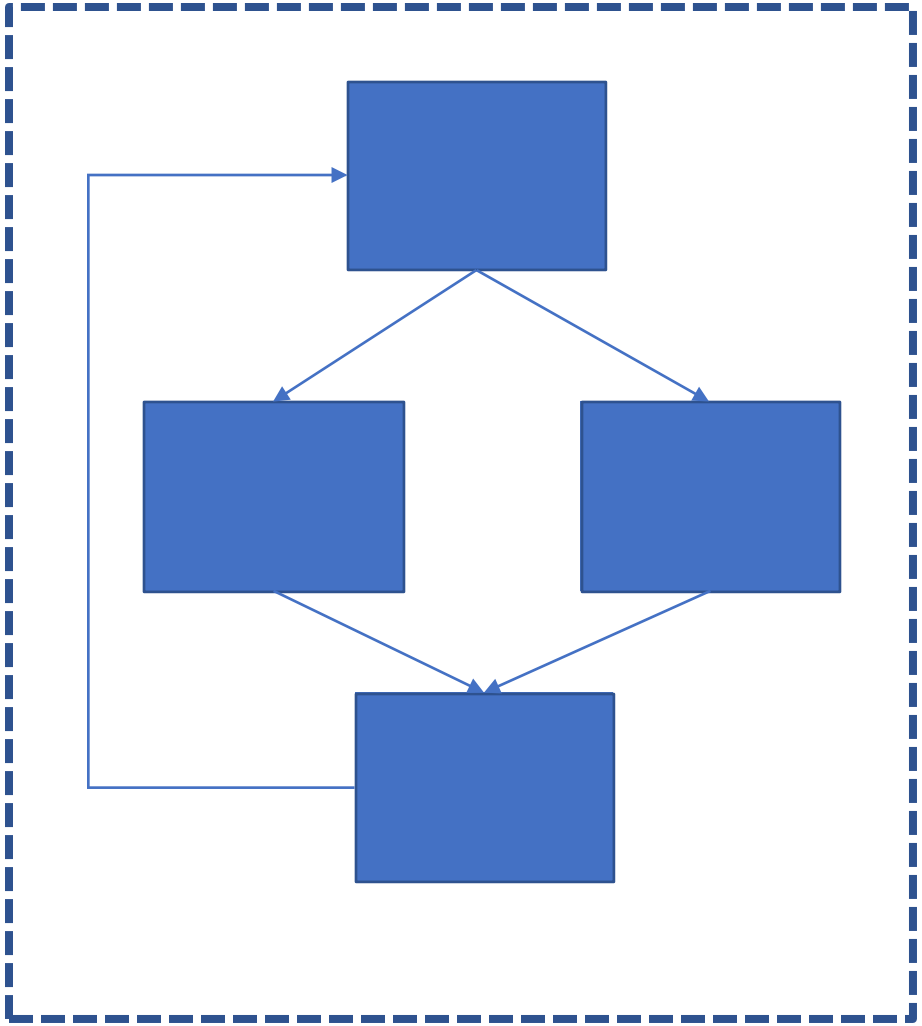






What makes software runnable?

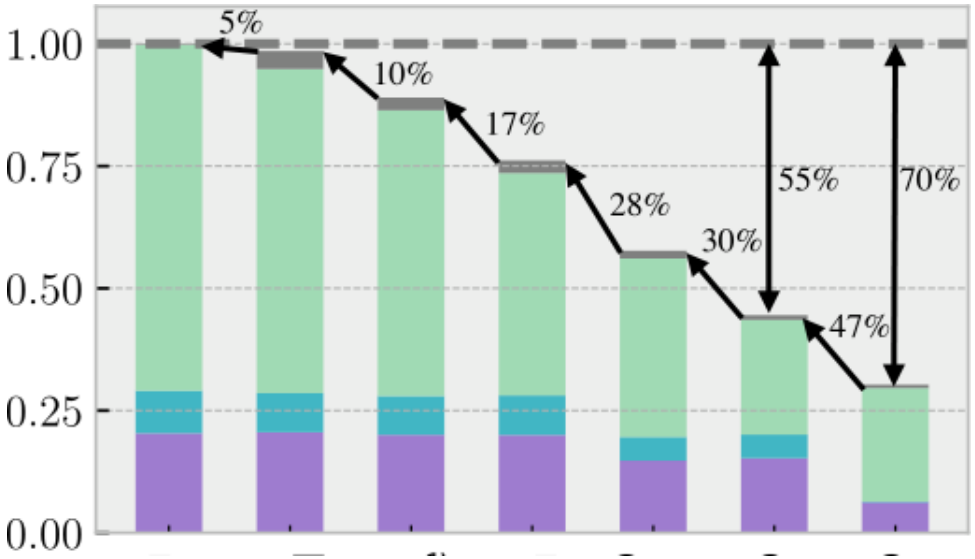
```
379  int walk_page_range(struct mm_struct *mm, unsigned long start,
380                  unsigned long end, const struct mm_walk_ops *ops,
381                  void *private)
382  {
383          int err = 0;
384          unsigned long next;
385          struct vm_area_struct *vma;
386          struct mm_walk walk = {
387                  .ops            = ops,
388                  .mm             = mm,
389                  .private        = private,
390          };
391
392          if (start >= end)
393                  return -EINVAL;
394
395          if (!walk.mm)
396                  return -EINVAL;
397
398          mmap_assert_locked(walk.mm);
399
400          vma = find_vma(walk.mm, start);
401          do {
402                  if (!vma) { /* after the last vma */
403                          walk.vma = NULL;
404                          next = end;
```

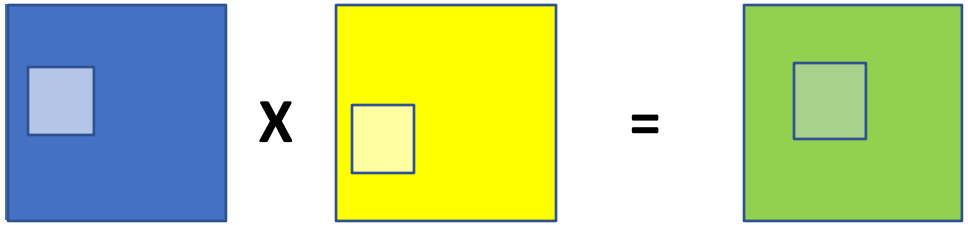


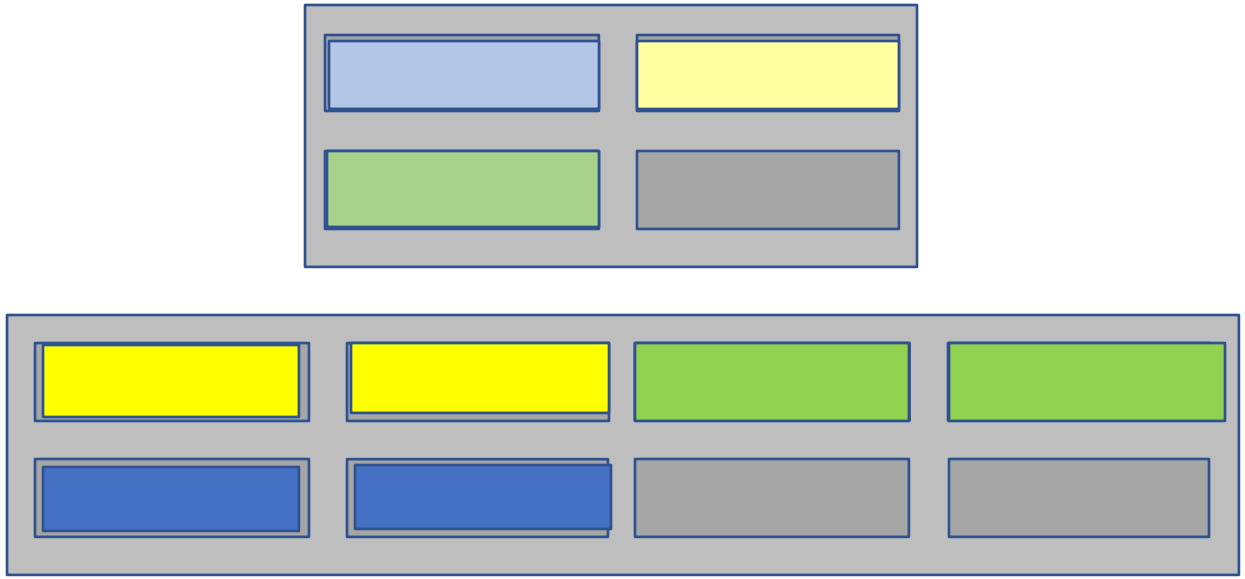
```
mov     0x18(%rdi),%r14
sub     0x10(%rdi),%r14
mov     0x8(%rdi),%rbp
add     %r14,%rsi
mov     %r14,%rdx
setb    %al
add     0x30(%rdi),%rsi
setb    %cl
shr     $0x3,%rdx
cmp     %rsi,(%rdi)
lea     0x64(%rsi,%rdx,1),%r12
cmovae  (%rdi),%rsi
cmp     %r12,%rsi
cmovae  %rsi,%r12
test    %rax,%rax
```

```
16aea:      48 c1 ea 03
16aee:      48 39 37
16af1:      4c 8d 64 16 64
16af6:      48 0f 43 37
16afa:      4c 39 e6
16afd:      4c 0f 43 e6
16b01:      48 85 c0
```

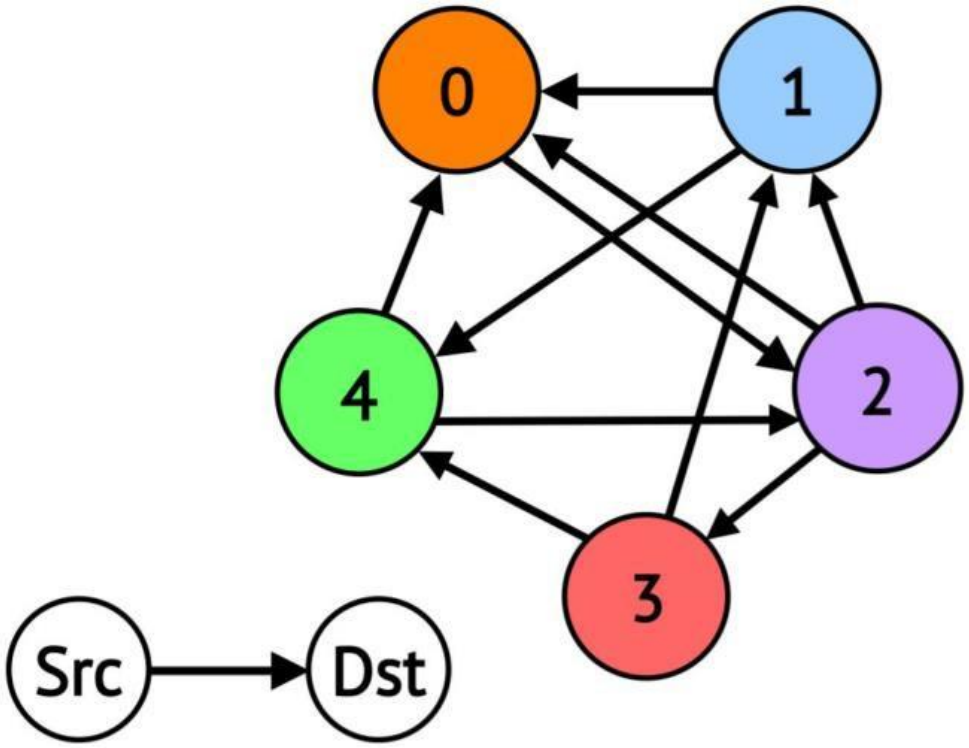
How do you *improve* software's performance?

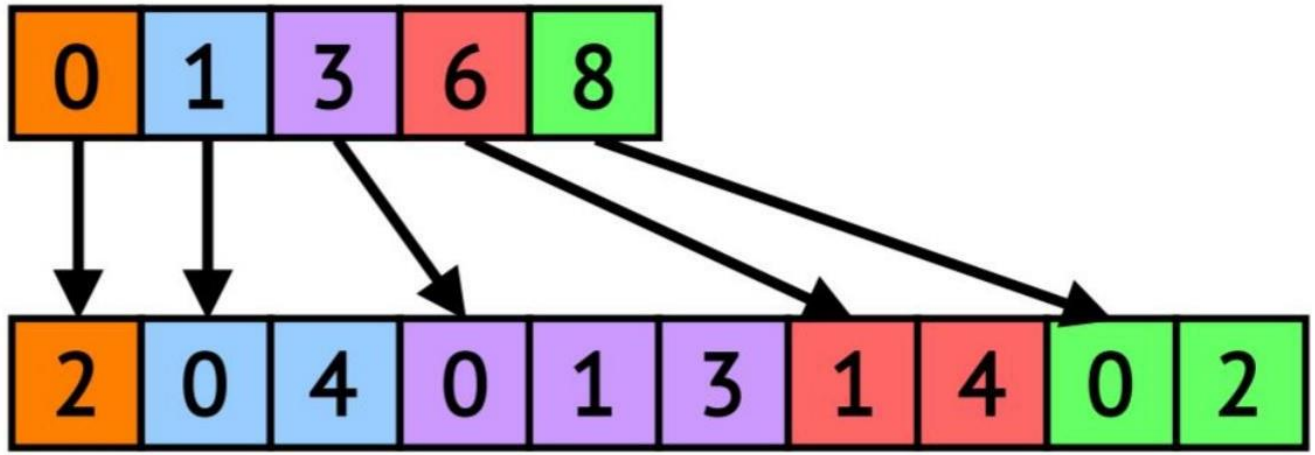




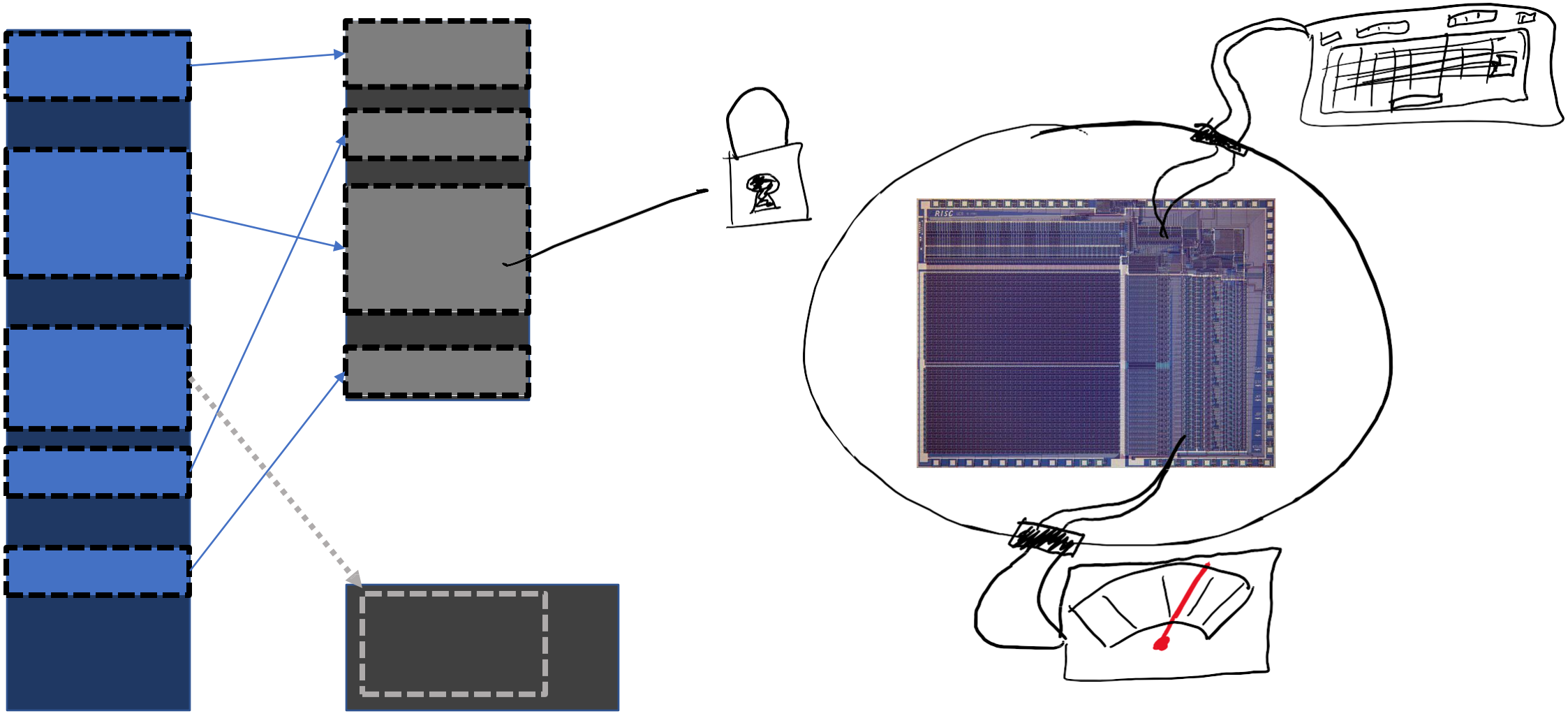


Are some programs intrinsically slow?

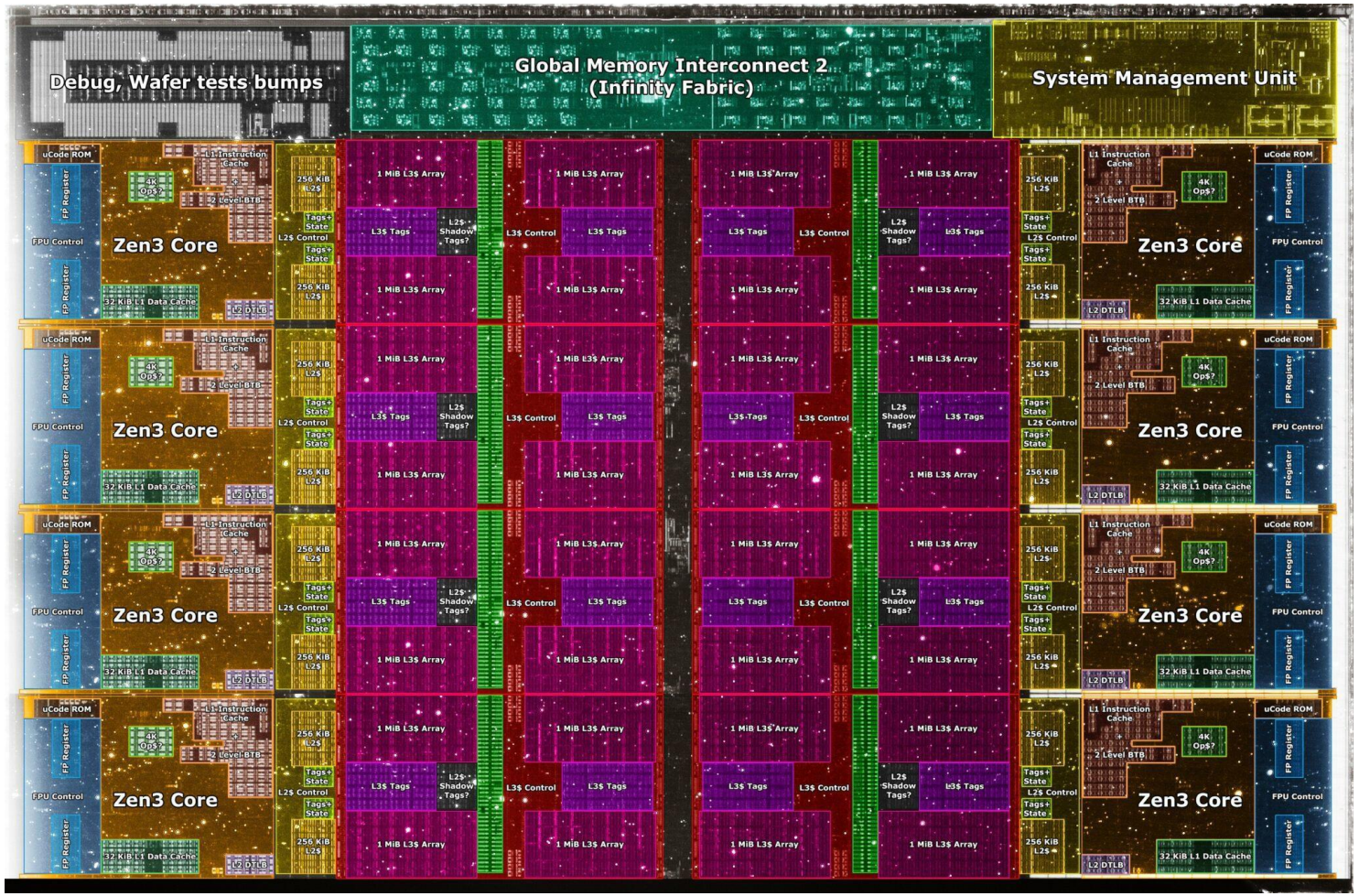




What are the lower-extremities of software?

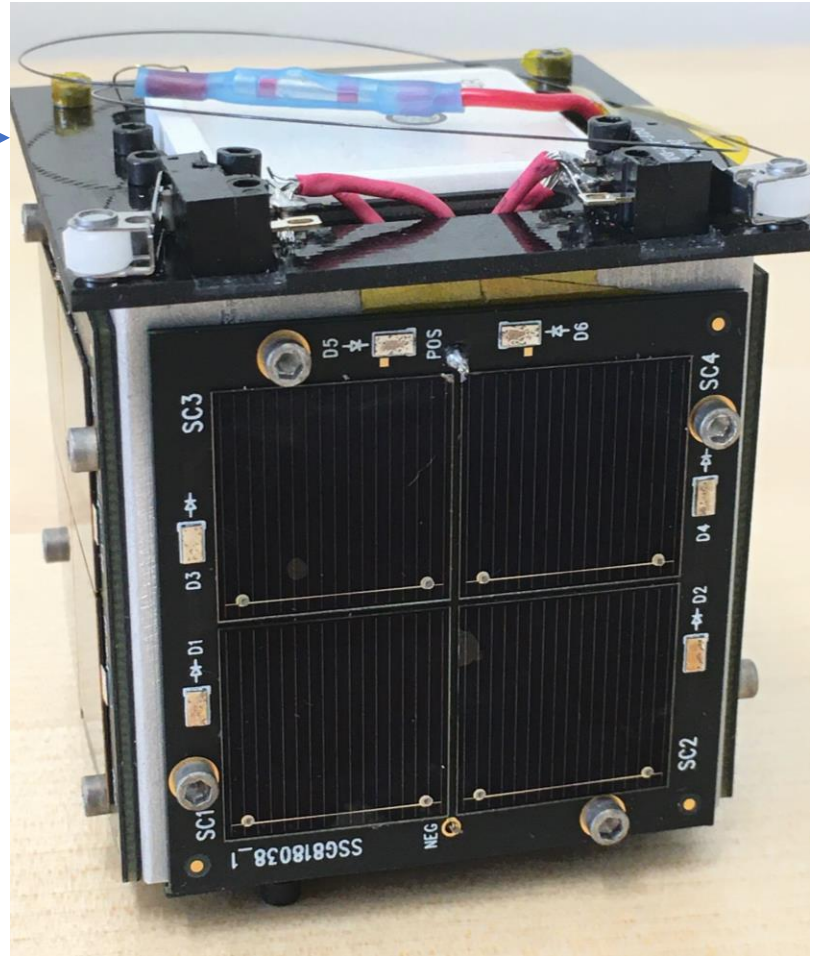


How (& why?) to do two things at the same time?



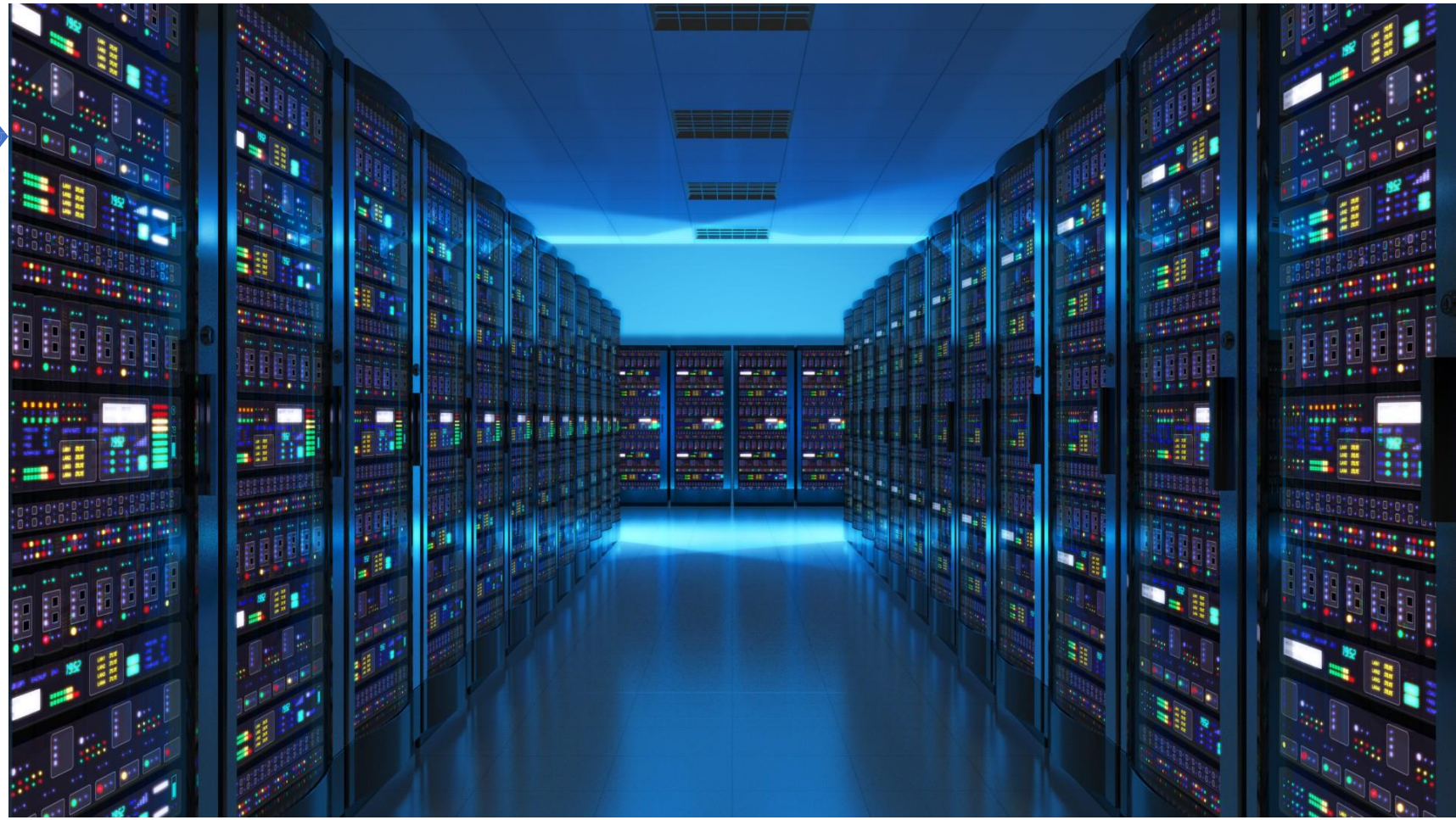
How does it all fit together in real systems?

Tartan-Artibeus-1 Batteryless Energy-harvesting Nanosatellite System



How does it all fit together in real systems?

Hyperscale computing in data centers



Setting expectations upfront

- Series of labs (Lab 0 – "bootstrapping" goes out soon, check slack)
  - Do all labs except lab 0 in pairs. The point is not to make you code or experiment for the rest of your life. **We want you to learn by doing**.
- Labs released via AFS: /afs/ece.cmu.edu/class/ece344/assign
  - For each lab, *handout.txt* is your main reference & documentation. **READ IT CAREFULLY** because it will tell you what you need to know to do the lab
- Labs are **not** just "code & submit". Instead, you'll be building, studying, measuring, and evaluating systems. Coding it up is step 1.

Course Calendar

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures
10/9/2024	Virtual Memory
10/14/2024	**Fall Break**
10/16/2024	**Fall Break**
10/21/2024	The Compiler Is Here to Help (And, wrapping up VM)
10/23/2024	Sparse Problems Introduction
10/28/2024	Sparse Problems Optimization (Propagation Blocking)
10/30/2024	Parallelism, Coherency, and Concurrency Basics
11/4/2024	Synchronization and Transactional Memory
11/6/2024	Consistency, Coherency, and Understanding the Model
11/11/2024	Special Topic: Introduction to Data Center Computing
11/13/2024	Special Topic: Introduction to Building Ethical Computer Systems

Diving into Computer Architecture

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures

Broad introduction to computer architecture, understanding the architecture / microarchitecture distinction, what is an ISA?, what limits computer performance?, how to think about hardware using abstractions, Amdahl's Law

ILP & Dealing with Hazards

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures

Introduction to ILP, pipelining and what gets in its way, how control flow happens at execution time, branch prediction

Caches & Memory Hierarchies

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures

Memory is the real problem!
Caches, memory hierarchies: an architect's view, cache replacement and other cache optimizations

Principled Performance Analysis

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures

Measuring a system in a meaningful way, understanding performance measurement pitfalls, Pareto analysis, design space iteration & exploration, Amdahl's Law (again)

Microarchitectural Optimizations

Date	Topic
08/26/2024	Introduction to the Hardware/Software Boundary
08/28/2024	von Neumann Architectures
09/04/2024	Computer Architecture Basics
09/09/2024	ISAs: The RISC-V ISA
09/11/2024	Pipelines and Hazards [pptx] / [pdf]
09/16/2024	Control hazards and Branch Prediction
09/18/2024	Caches and Memory Hierarchy
09/23/2024	Cache Replacement Policies and Enhancements
09/25/2024	Introduction to Performance Evaluation
09/30/2024	Design Space Exploration
10/02/2024	Advanced Architecture: Superscalar and Out of Order
10/07/2024	Advanced Dataflow Architectures

Going beyond IPC=1, advanced ILP techniques, superscalar & out-of-order execution, vector processors, Very Large Instruction Word processors, other more exotic architectures like dataflow

Virtual Memory

10/9/2024	Virtual Memory
10/14/2024	**Fall Break**
10/16/2024	**Fall Break**
10/21/2024	The Compiler Is Here to Help (And, wrapping up VM)
10/23/2024	Sparse Problems Introduction
10/28/2024	Sparse Problems Optimization (Propagation Blocking)
10/30/2024	Parallelism, Coherency, and Concurrency Basics
11/4/2024	Synchronization and Transactional Memory
11/6/2024	Consistency, Coherency, and Understanding the Model
11/11/2024	Special Topic: Introduction to Data Center Computing
11/13/2024	Special Topic: Introduction to Building Ethical Computer Systems

Virtual Memory, virtualization basics, bad ways to do VM, hardware/software co-design for virtualization, VM advanced topics, huge pages, TLB design

Sparse Computation

10/9/2024	Virtual Memory
10/14/2024	**Fall Break**
10/16/2024	**Fall Break**
10/21/2024	The Compiler Is Here to Help (And, wrapping up VM)
10/23/2024	Sparse Problems Introduction
10/28/2024	Sparse Problems Optimization (Propagation Blocking)
10/30/2024	Parallelism, Coherency, and Concurrency Basics
11/4/2024	Synchronization and Transactional Memory
11/6/2024	Consistency, Coherency, and Understanding the Model
11/11/2024	Special Topic: Introduction to Data Center Computing
11/13/2024	Special Topic: Introduction to Building Ethical Computer Systems

Introduction to sparsity, what is a sparse problem and why is one difficult?, understanding the performance limiters, sparse problem optimization strategies, open problems

Parallelism & Concurrency

10/9/2024	Virtual Memory
10/14/2024	**Fall Break**
10/16/2024	**Fall Break**
10/21/2024	The Compiler Is Here to Help (And, wrapping up VM)
10/23/2024	Sparse Problems Introduction
10/28/2024	Sparse Problems Optimization (Propagation Blocking)
10/30/2024	Parallelism, Coherency, and Concurrency Basics
11/4/2024	Synchronization and Transactional Memory
11/6/2024	Consistency, Coherency, and Understanding the Model
11/11/2024	Special Topic: Introduction to Data Center Computing
11/13/2024	Special Topic: Introduction to Building Ethical Computer Systems

Parallel computation, parallel architectures concurrency, memory consistency models, synchronization, atomics, transactional memory networks on chip

End of semester: emerging architecture topics

10/9/2024	Virtual Memory
10/14/2024	**Fall Break**
10/16/2024	**Fall Break**
10/21/2024	The Compiler Is Here to Help (And, wrapping up VM)
10/23/2024	Sparse Problems Introduction
10/28/2024	Sparse Problems Optimization (Propagation Blocking)
10/30/2024	Parallelism, Coherency, and Concurrency Basics
11/4/2024	Synchronization and Transactional Memory
11/6/2024	Consistency, Coherency, and Understanding the Model
11/11/2024	Special Topic: Introduction to Data Center Computing
11/13/2024	Special Topic: Introduction to Building Ethical Computer Systems

(Subject to change) What is happening in the field of computer architecture and systems today? What are the exciting new ideas?
Hyperscale data center architectures
Q/A

Lab 0: Bootstrapping

- 18-344 has lots of moving parts. Lab 0 is about figuring them all out
  - You'll be using each of them again in subsequent labs.
- SPEC2017: collection of benchmark programs designed for evaluating computer architectures
  - Needlessly complex and very difficult to change infrastructure. SPEC will be a pain, but will give you the Real Computer Systems Experience.
- Pin: binary instrumentation tool used to insert code into program binaries to implement computer architecture simulators
- Destiny: memory modeling tool useful for evaluating the time & energy to access different cache/memory designs

Lab 1: Branch Prediction

- As you will learn, (and as you may recall from 213) microarchitectures predict the outcome of their branch instructions
- Write a branch predictor simulator
- Evaluate different implementations
  - Cost, accuracy, implementation feasibility
- Write-up in English prose characterizing and explaining your design, and with quantitative evaluation of your design's performance.

Lab 2: Memory Hierarchy Design Space

- Cache hierarchies are big complex microarchitectural components
- Which is best for a given set of programs?
- You will start by writing a cache, and then a whole memory hierarchy.
- Then you will run a design space exploration process to optimize your memory hierarchy implementation, subject to physical constraints and performance & efficiency goals
- You will explore different replacement policies
- Write-up in English prose summarizing your design space exploration and conclusions, including quantitative evaluation.

Lab 3: Virtual Memory

- Virtual memory provides process isolation and access control using paging and some hardware support
- You will implement an emulation of a page table and the basic functions that you will use to manipulate the page table
- You will implement simulated hardware to accelerate translation of virtual to physical addresses and evaluate its impact on system performance
- You will study your implementation and quantitatively analyze your page table and hardware support
- Write-up describing your design, including quantitative evaluation of your system and its performance

Lab 4: Sparse Workload Optimization

- Sparse workloads are programs that work on datasets that have sparse structure, like graphs that have lots of vertices and far fewer edges (sparse adjacency matrix). Sparse workloads are hard to cache.
- Given a simple unoptimized sparse workload implementation, you will use a highly specialized optimization called Propagation Blocking to optimize, yielding a much higher performance implementation
- You will study your optimized version, and quantitatively analyze your design choices.
- Write-up including description of your implementation and summary of quantitative results

Lab 5: Synchronization for Parallel Code

- Parallel programs require synchronizing to avoid confusing interactions between threads.
- There are many ways to implement synchronization
- You will implement different synchronization mechanisms (from spin locks to transactional memory) for two performance-sensitive test programs
- You will quantitatively study your implementations and their performance on the two test programs
- Write-up including description of your synchronization alternatives and a quantitative analysis of performance