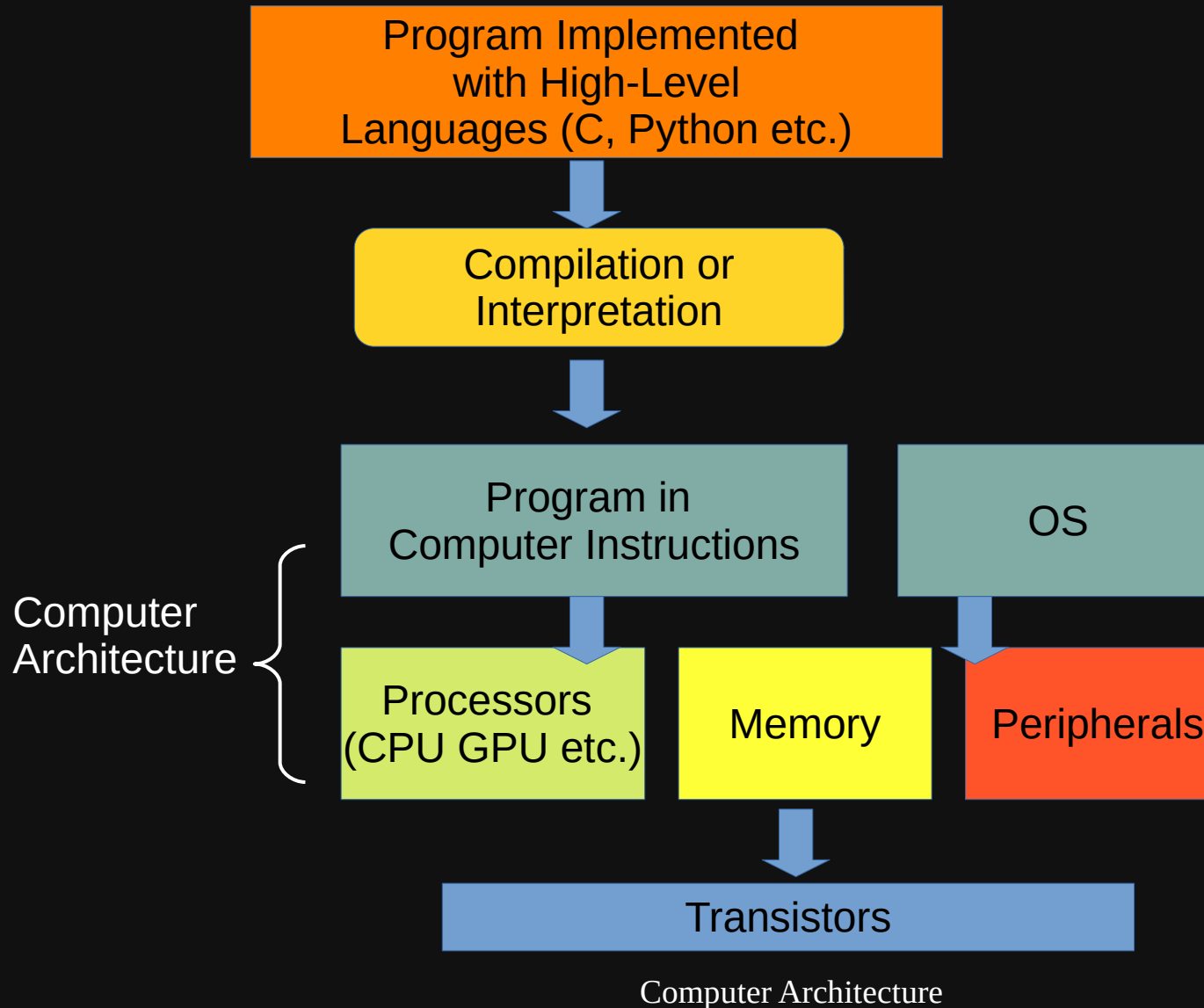


Introduction to Computer Architecture

Wei Wang

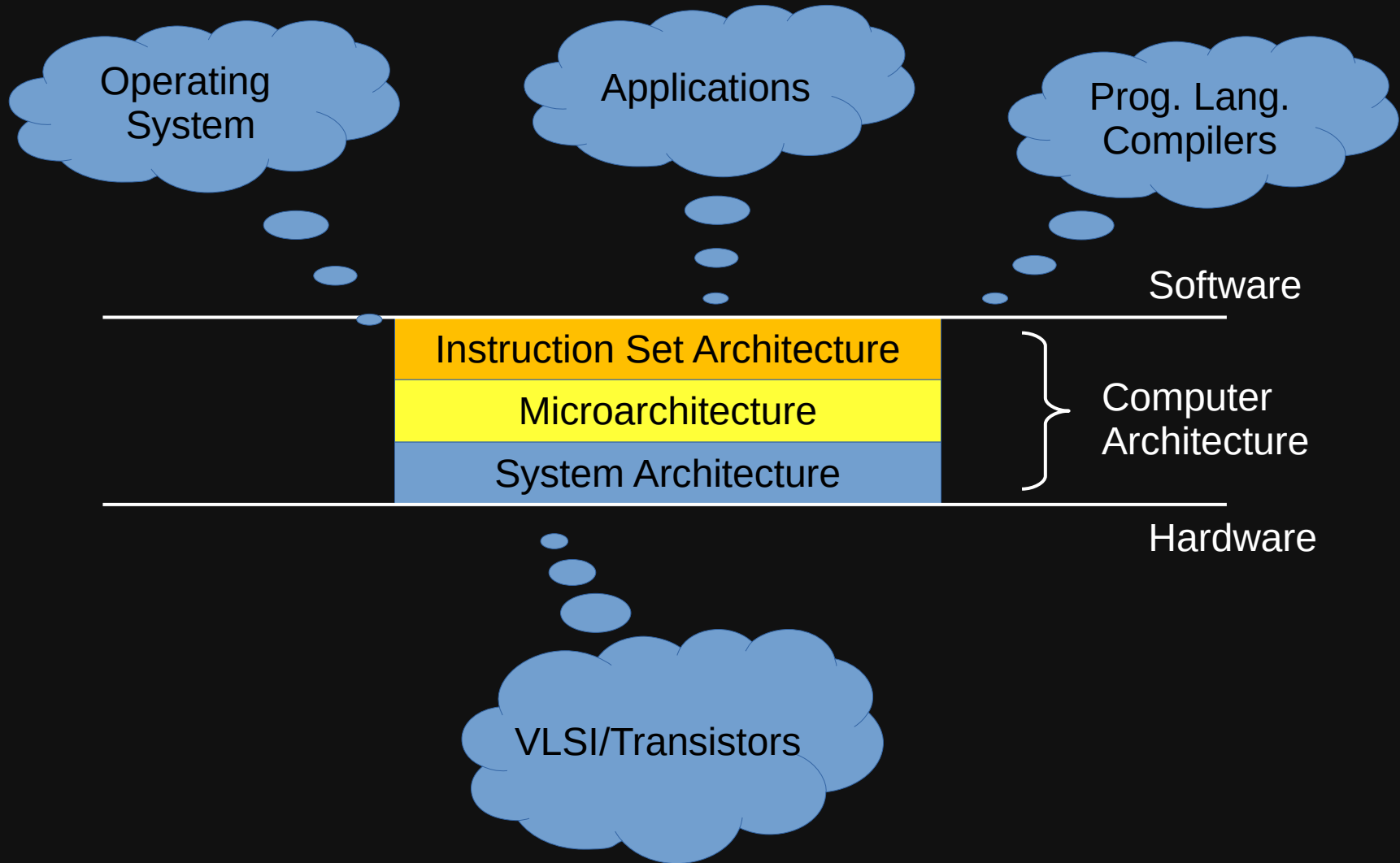
What is Computer Architecture?

What is Computer Architecture?



What is Computer Architecture?

Cont.



What is Computer Architecture?

Cont.

- Instruction Set Architecture (ISA)
 - ISA is set of computer instructions provided to programmers to implement software
 - An abstract model of a computer
 - ISA is implemented in Microarchitecture.
- Microarchitecture (uarch)
 - The actual implementation of ISA in processors, e.g., adders and multipliers.
 - Uarch is about ensuring instructions run correctly and quickly.
 - Intel and AMD have two implementations of x86 ISA, thus two uarchs.
 - Uarch is implemented with transistors
- System architecture (sys-arch)
 - Any hardware implementation beyond processors, such as memory and I/O devices
 - Sys-arch is about the connecting processors and other devices

Topics in This Course

- Instruct Set Architecture (ISA)
- Computer Arithmetic
- Instruction Level Parallelism (pipelining and super-scalar, CMP, and SMT)
- Speculative Executions (branch prediction and memory disambiguation)
- Memory hierarchy (TLB, caches and DRAM)
- Modern parallel processors (NUMA, GPUs and ASICs).

Goals of Computer Architecture Design

- Provide easy to use Instruction Sets
- Design hardware to execute instructions correctly
- Optimized hardware management to execute instructions with best possible performance.
 - Mostly, computer architecture is all about performance.

Why Learn Computer Architecture

It is All About Performance

- Writing efficient programs requires understanding of Computer Architecture. E.g.,
 - Does the code has too much dependencies between instructions?
 - Does the code using the cache efficiently?
 - Why does this code run so slow?
- You need to understand how hardware components are utilized by your program to optimize your code.

It is Also About Correctness and Security

- Parallel Programming is now an essential part of modern software.
 - Writing correct parallel program requires understanding of parallel processor designs.
- Uarch designs have builtin security features.
 - E.g., NX bit (non-executable memory)
- Some Uarch features are also causing security vulnerabilities
 - E.g., Spectre and Meltdown.

Concerns for Computer Architecture Designs

Application Areas

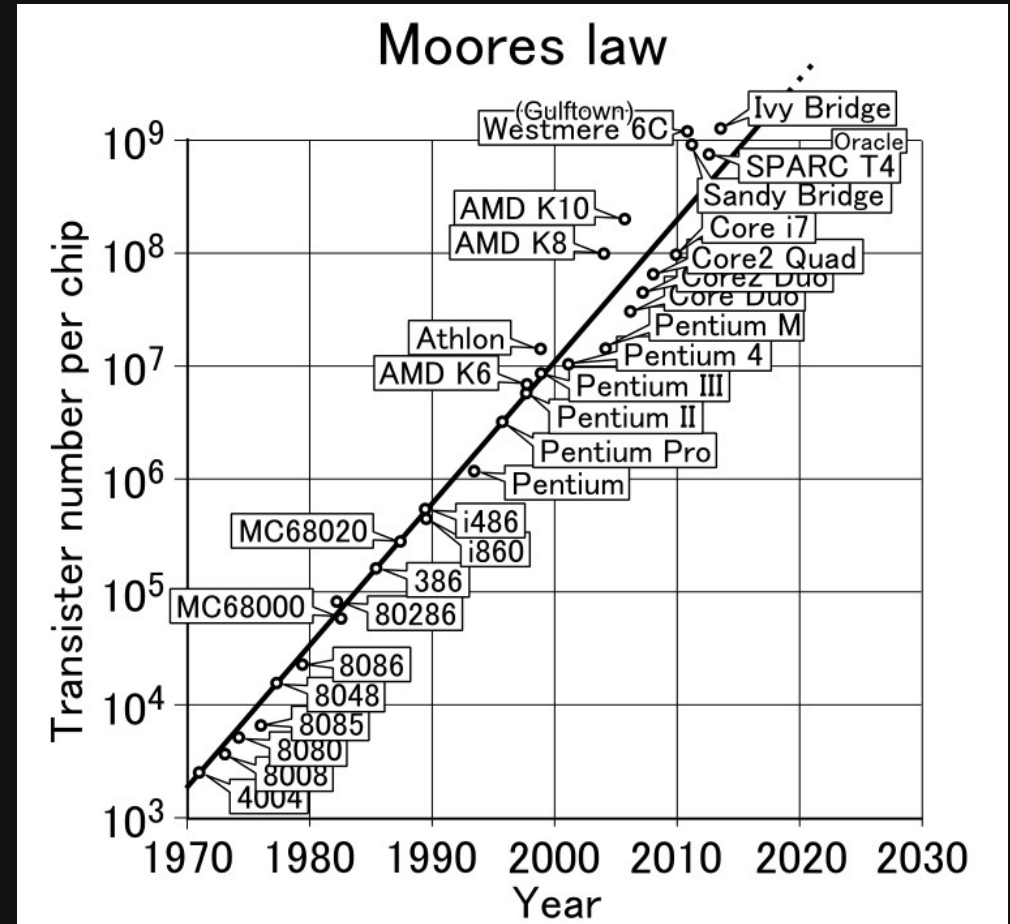
- General-Purpose Laptop/Desktop
 - Productivity, interactive graphics, video, audio
 - Optimize price-performance
 - Examples: Intel Core, AMD Ryzen
- Embedded Computers
 - PDAs, cell-phones, sensors => Price, Energy efficiency
 - Examples: Arm Processors
 - Game Machines, Network devices => Price-Performance
 - Examples: Sony Emotion Engine, IBM 750FX

Application Areas cont.

- Commercial Servers
 - Database, transaction processing, search engines and machine learning
 - Performance, Availability, Scalability
 - Server downtime could cost a brokerage company more than \$6M/hour
 - Examples: Google datacenters
- Scientific Applications
 - Protein Folding, Weather Modeling, CompBio, Defense
 - Floating-point arithmetic, Huge Memory
 - Examples: IBM DeepBlue, Cray Titan, etc.

Moore's Law

- Gordon Moore, Founder of Intel
- The number of transistors in a dense integrated circuit doubles approximately every two years
- Moore's law is expected to end around 2025 when reaching 1nm.



The Survival of Moore's Law

- Leakage power was a “killer” of Moore's Law
 - FinFET reduced leakage power
- Multi-gate transistors on a single node
 - i.e., multiple fins on one node
 - Smaller transistors in theory

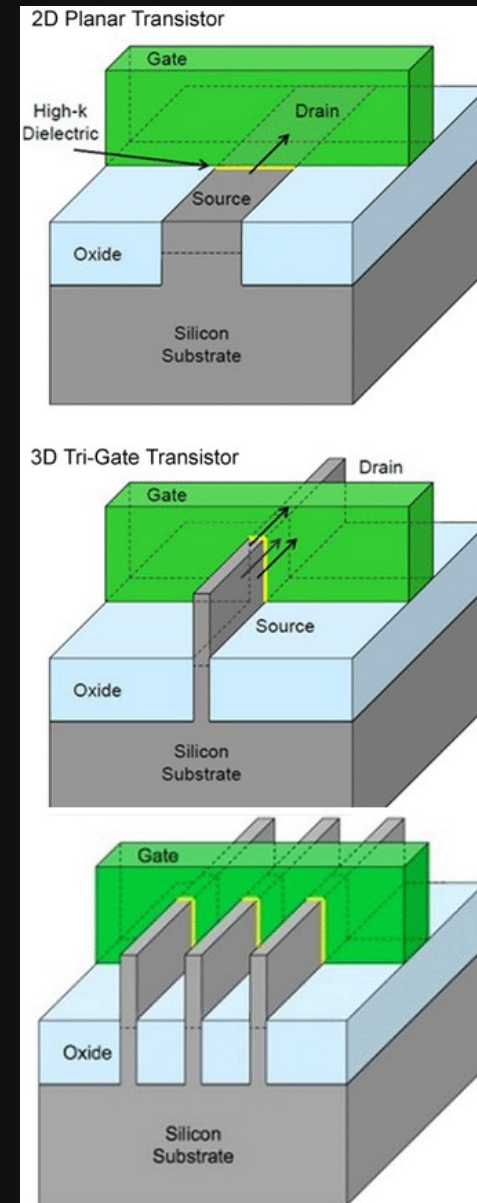
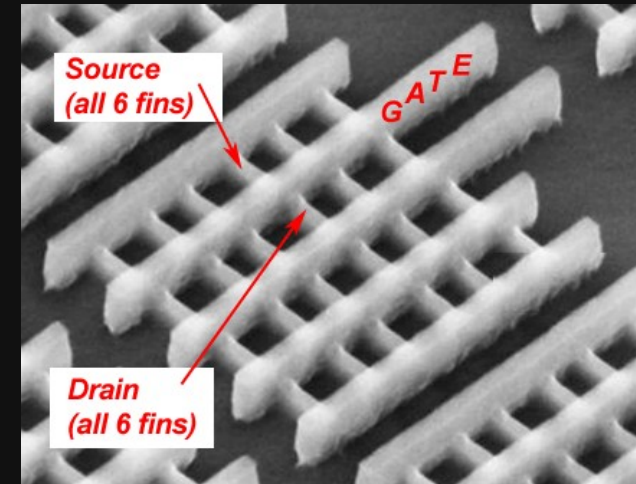


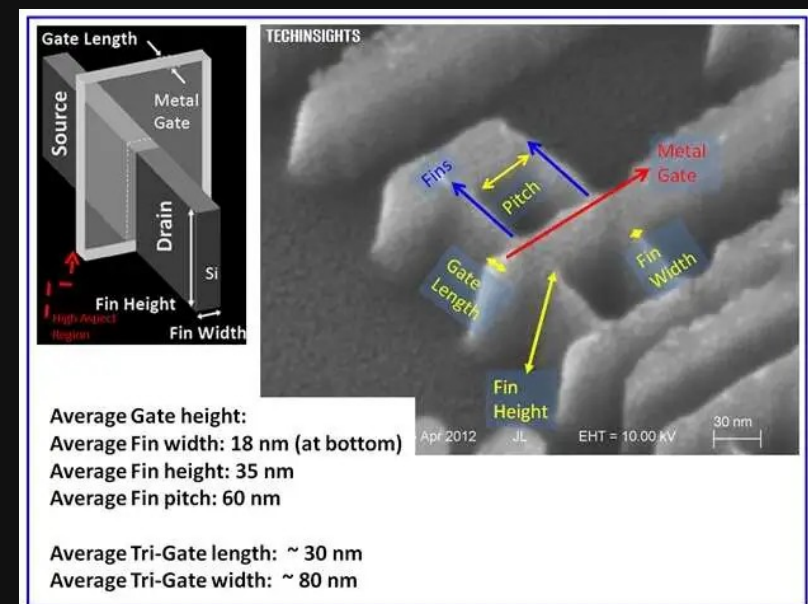
Image by Intel

The Survival of Moore's Law cont.

- Multi-gate transistors on a single node
 - i.e., multiple fins on one node
 - Smaller transistors in theory

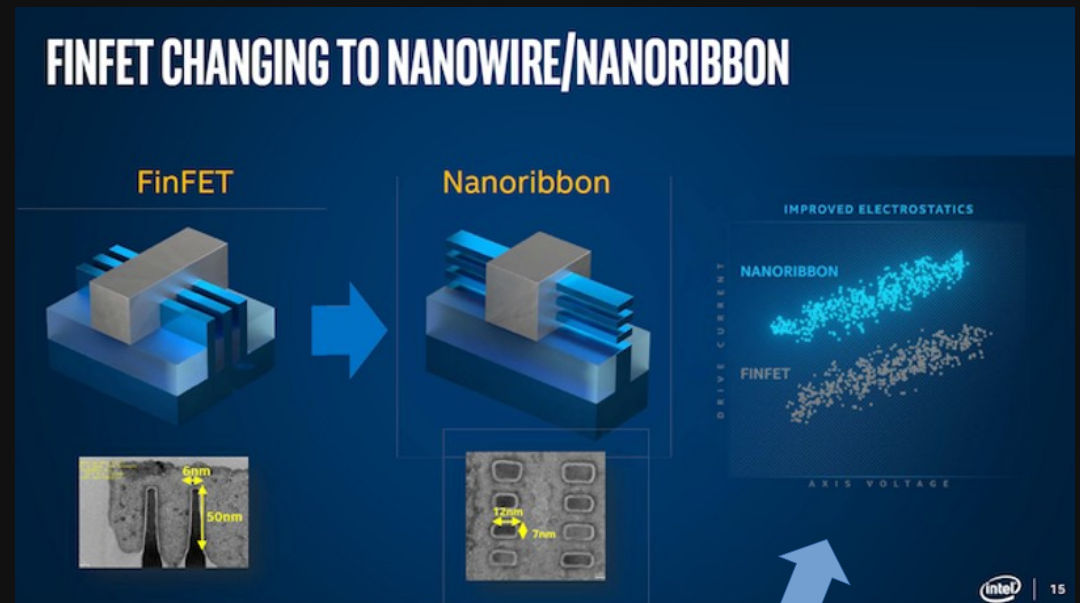


Intel 22nm 3D transistors:
Three-gate on a 80nm node
Images by Intel



The Survival of Moore's Law cont.

- Beyond FinFET => GAA-FET (?)
- GAA for Gate-All-Around



GAAFET has larger current than FinFET at the same supply voltage

The Benefits of Moore's Law

- More transistors!
- More transistors => more functional units (ALUs, FPUs etc.)
 - Can execute multiple instructions simultaneously.
- More transistors => more cores?
 - Can execute multiple programs simultaneously.
- More transistors => big cache
 - Can hold more data in CPU instead of DRAM.
- More transistors => More complex management units.
 - Branch predictors, data prefetchers, out-of-order executions.

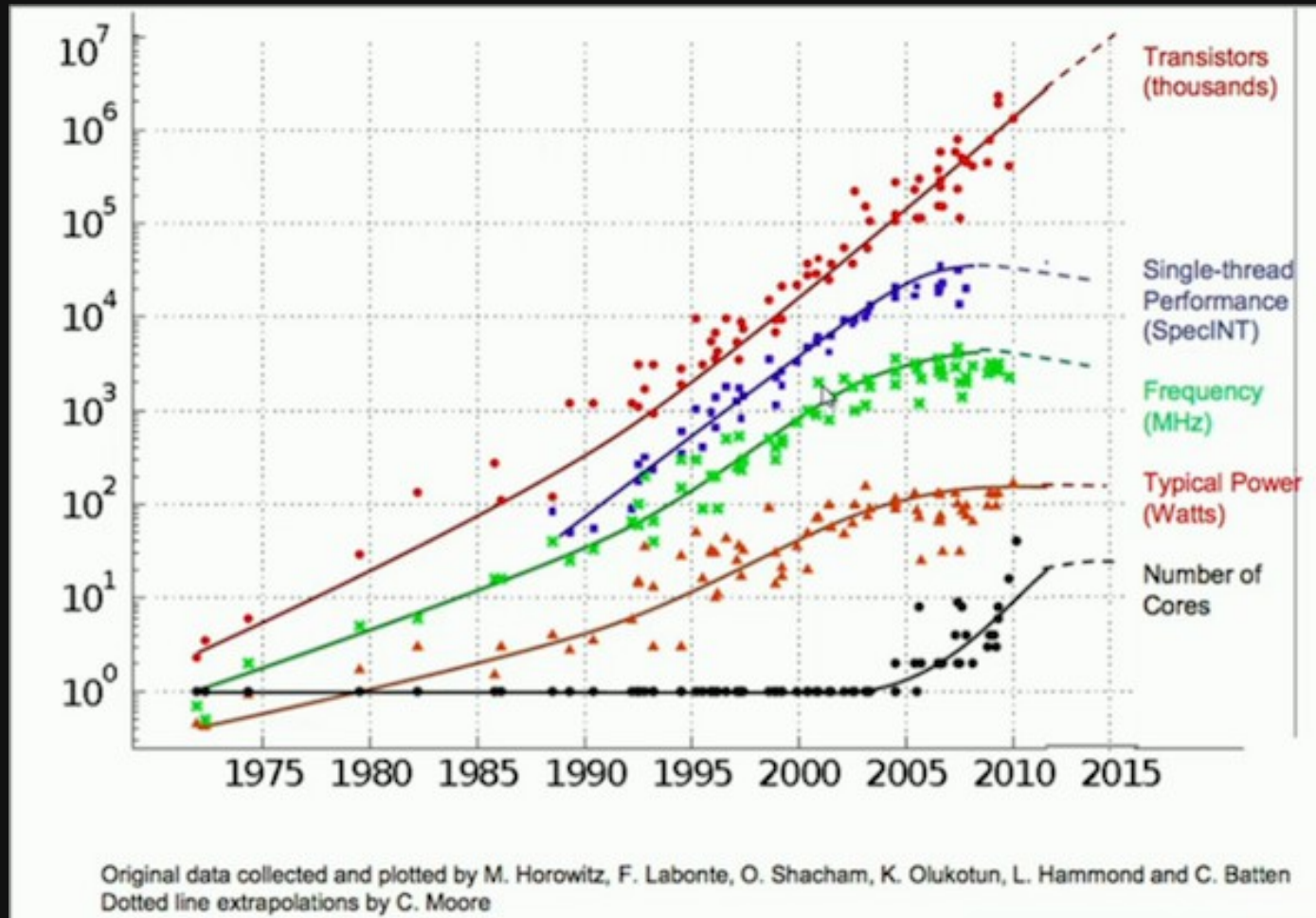
Dennard Scaling

- Robert Dennard, IBM fellow
- Dennard Scaling: As transistors get smaller their power density stays constant, so that the power use stays in proportion with area => both voltage and current scale (downward) with length

Dennard Scaling cont.

- What Dennard Scaling mean:
 - For each new generation of processors:
 - We can double the transistor count
 - We can double the frequency without extra power usage
 - Therefore, we can more than double the performance for each new generation of processors
 - This is why computer industry was so successful in 80s and 90s. Performance gain is guaranteed and is free (no extra power).
- Unfortunately, Dennard Scaling started to fail around 2004.
 - The frequency growth stopped, and growth of single core performance has significantly slowed down.

The Stall of CPU Frequency



Computer Architecture in Post Dennard Scaling Era

- Parallel processors:
 - Instead of doubling frequencies, we double the number of cores
 - Can help, but not a long term solution
- Specialize processors:
 - Graphic Processing Units (GPUs) for general purpose data processing, i.e., scientific and machine learning
 - Application-specific Integrated Circuits (ASIC). E.g., machine learning processor.
- Hopefully, Graphene or Quantum computing can save us in time.

Dealing with Complexity: Abstractions

- As an architect, the main job is to deal with tradeoffs
 - Performance, Power, Die Size, Complexity, Applications Support, Functionality, Compatibility, Reliability, etc.
- Technology trends, applications... How do we deal with all of this to make real tradeoffs?
 - Abstractions allow this to happen
- Layer-ed design allows us to focus on the design and optimization of one layer at a time
 - E.g., ISA, uarch and Sys-arch are three abstraction layers
 - CPU and memory are two abstraction layers

Design Metrics: Performance

- Reduce Response/Execution Time
 - Time between start and completion of an event
- Increase Throughput
 - Total amount of work in a given amount of time
- Execution Time is reciprocal of performance
- “X is N times faster than Y”
- $N = \text{Execution Time Y} / \text{Execution Time X}$
- Wall-Clock Time, CPU time (no I/O)

Design Metrics: Cost

- The goal is to reduce manufacture cost
- Moore's Law also helps here, as doubling transistors does not require more raw materials.
 - However, smaller transistors make it harder for quality control.

Design Metrics: Availability

- Availability: Fraction of Time a system is available
- For servers, may be as important as performance
- Mean Time Between Failures (MTBF)
 - Period that a system is usable
 - Typically 500,000 hours for a PC Hard drive
 - Typical life time for a CPU is 10 years.
- Mean Time to Repair (MTRR)
 - Recovery time from a failure
 - Should approach 0 for a big server (redundancy)

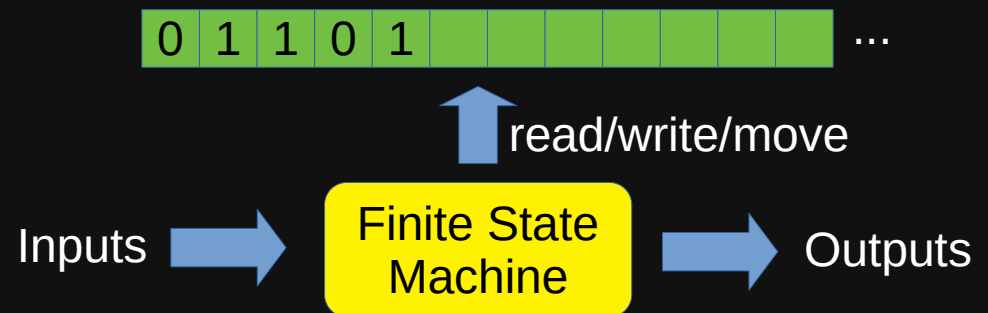
Metrics: Power Dissipation

- We all understand why energy is important for embedded CPUs...
- High-end Server: ~130-170W
- High-end Desktop: ~70-100W
- High-end Laptop: ~30W
- Battery-Optimized Laptop: ~3-10W
- Embedded CPUs: ~.5-1W
- DSP: ~100mW
- Microcontrollers: ~10mW

Theoretical Computer Architectures

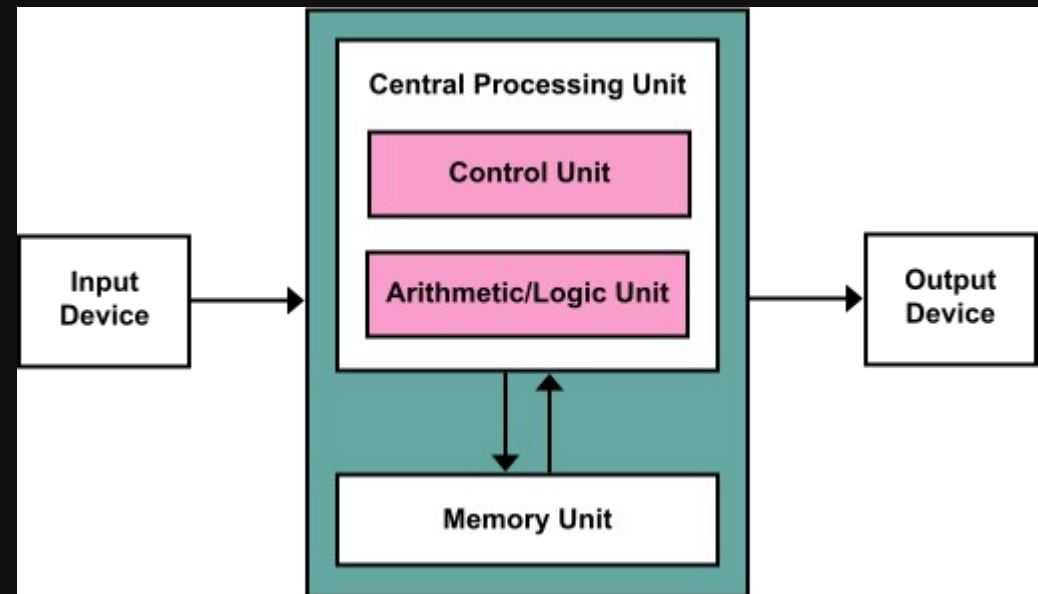
Turing Machine

- Turing machine is the mathematical model of modern computers.
 - Invented in 1936
- Turing machines are composed of:
 - A state machine specifies a sequence of operations
 - Equivalent to processors
 - A tape for storing data
 - Equivalent to memory



Von Neumann architecture

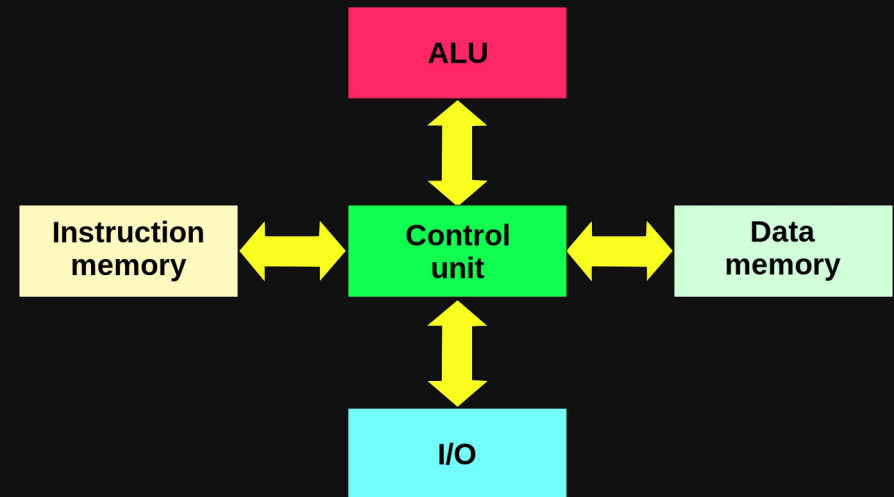
- A theoretical computer architecture that is very close to modern computers
 - Proposed around 1945
- Four components:
 - A CPU, with control and arithmetic units
 - A memory for data and instructions
 - An input device
 - An output device



* shamelessly taken from Wikipedia, figure made by Kapoht

Harvard Architecture

- A theoretical computer architecture based on Harvard Mark I.
- Similar to von Neumann architecture, except that instructions and data are stored separately.



* also shamelessly taken from Wikipedia, figure made by Nessa Ios

Dataflow Architecture

- An architecture where functional units are triggered by the arrival of data instead of clock signal.
- Every few commercially available products.
 - Mostly academia research processors
 - Some DSP, Network router, GPU use dataflow architecture.
- Inherently hard to scale-up.
- There is a recent revival of dataflow architecture due to smaller transistors cannot all run at lowest frequency.

Class Information

Prerequisites

- CS 3423, System Programming
- CS 3843, Computer Organization
- You should be able to write and read assembly, C and Python programs.

Grading

- Two in class midterm exams (25%)
- One final exam (25%)
- Assignments, including written assignments and paper reading assignments (30%)
- Project (15%)
- Class participation and in class quizzes (5%)
- extra credit events (extra 3%)
- The percentages may be adjusted depending on the number of assignments and quizzes

Exams

- Midterm exams are in class
 - Feb 21, Wednesday
 - Mar 27, Wednesday
- Final Exam
 - April 29th, in class
- Midterm exam days are fixed, plan your travel accordingly
- Exams are online
- No make-up exam unless university allowed excuses.

Class Information

- Textbooks
 - Computer Architecture: A Quantitative Approach, 5th ed, John L. Hennessy and David A. Patterson
 - Computer Organization and Design MIPS Edition: The Hardware/Software Interface, 5th ed, by David A. Patterson and John L. Hennessy
 - Both books are optional.
- Late homework is docked 10%
 - If it is more than one week late, the assignment will not be accepted

Class Information cont'd

- Class site
 - Canvas,
 - I will post assignment and project handouts on Canvas
 - Syllabus, Schedule and Slides:
 - https://wwang.github.io/teaching/Spring2024/Comp_Arch/syllabus/general_info.html
 - https://wwang.github.io/teaching/Spring2024/Comp_Arch/syllabus/syllabus.html
- Read Emails!!!

Instructor

- Instructor: Wei Wang (wei.wang@utsa.edu)
- Office hours
 - By Zoom, link is posted in Blackboard
 - Mon 3 pm to 5pm, Tuesday 3pm to 4pm.
 - And by appointment
- For email contact, always include “CS3853” in the subject line.
- Research areas:
 - Performance Engineering and Computer Systems
 - Computer Architecture, Compilers, Cloud Computing, Software Engineering, Cloud 3D and Applied AI.
 - Mainly focusing on performance analysis and system design for data centers and clouds

Teaching Assistants

- TAs:
 - TBD
- TA office hours:
 - None
- For grading questions, please ask TA first.

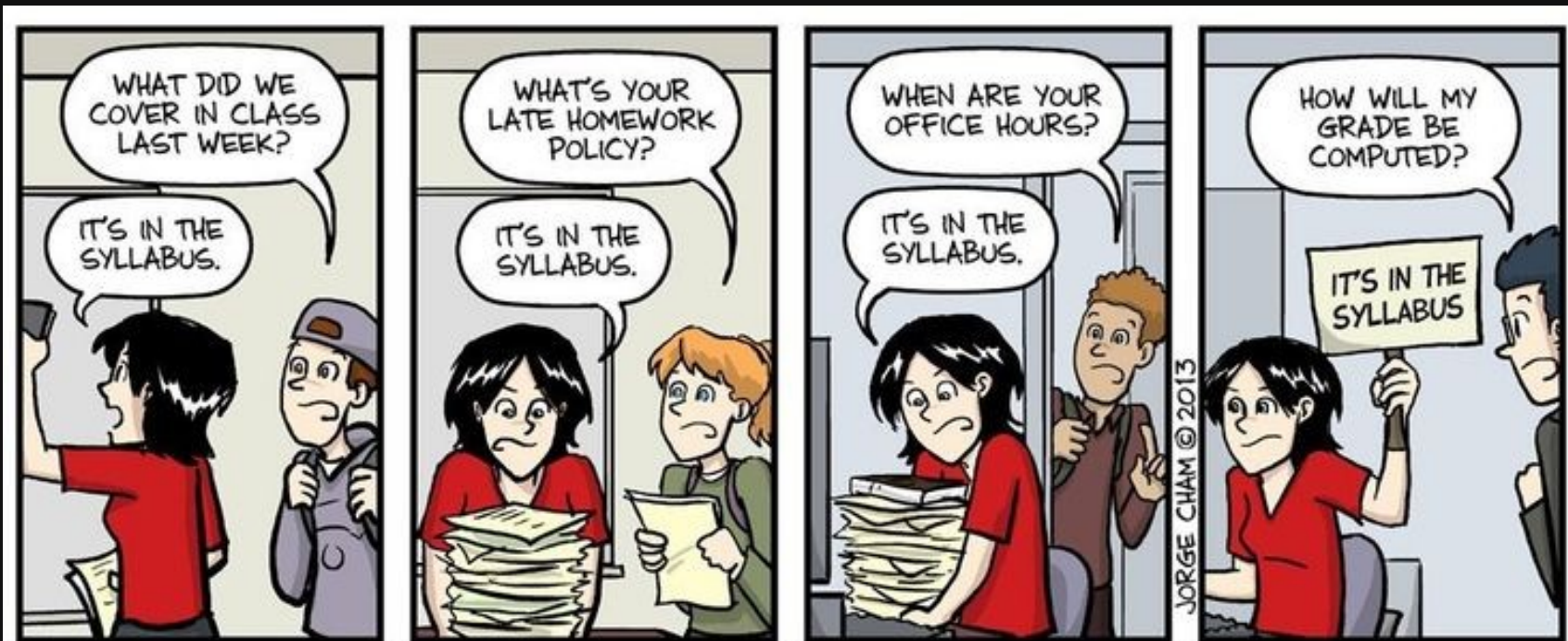
Project Information

- Topic:
 - Experiments with a CPU simulator and report the performance results of different CPU configurations
 - A report summarizes the results of the experiments.

Online Teaching

- Lectures
 - Will be synchronous-asynchronous hybrid.
 - Lectures will be mainly delivered through pre-recorded videos posted in YouTube.
 - Class hours will be used for important/difficult topics, homework/exams review, and QA
 - Announcement will be made on Canvas for each week's schedule.
 - Generally, I will not record regular synchronous lectures.
 - Reviewing lectures will be recorded.
 - For privacy reasons, unless approved by SDS, students should not record the lectures.
- Exams
 - Will be given through Canvas
 - For the sake of privacy, I cannot proctor the exam as I should be. Therefore, please be honest and do not cheat.

Syllabus – It is your friend



IT'S IN THE SYLLABUS

This message brought to you by every instructor that ever lived.

WWW.PHDCOMICS.COM

Student Participation

- Attendance
- In-class discussion
- Ask questions
- Let me know what do you think about this course
- Any feedback is welcome

Acknowledgment

- This lecture is partially based on the slides from the Computer Architecture course by Dr. David Brooks.