CSCI-564 Advanced Computer Architecture

Lecture 1: Introduction

Ismet Dagli

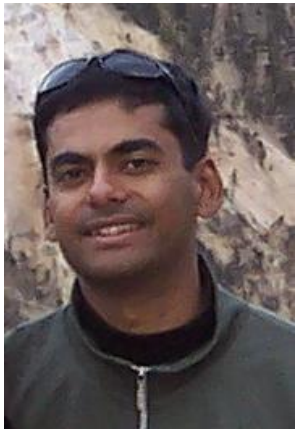
Colorado School of Mines

Course Advisor: Bo Wu

Disclaimer: most of the slides in this course are adapted based on material from four top-notch computer architecture researchers:

Onur Mutlu (EPFL)

Steven Swanson (UCSD)



Rajeev Balasubramonian (Utah)

David Wentzlaff (Princeton)

Today's Agenda

- What is architecture?
- Why is it important?
- What's in this class?
- Class logistics
- A bit history of computing

What is architecture?

- How do you build a machine that computes?
  - Quickly, safely, cheaply, efficiently, etc.

Why is architecture important?

- For the world
  - Computer architecture provides the engines that power all of computing

Civilization advances by extending the number of important operations which we can perform without thinking about them.

-- *Alfred North Whitehead*

Why is architecture important?

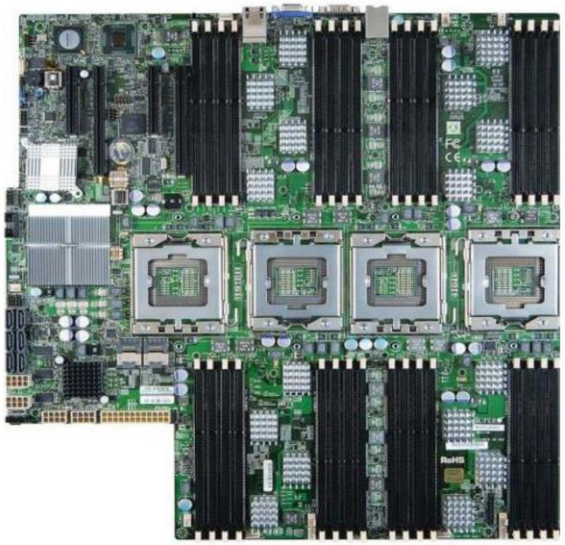
- For the world
  - Computer architecture provides the engines that power all of computing

    Civilization advances by extending the number of important operations which we can perform without thinking about them.

    -- *Alfred North Whitehead*
- For you
  - As computer scientists, software engineers, and sophisticated users, understanding how computers work is essential
  - The processor is the most important piece of this story
  - Many performance problems have their roots in architecture

Different scales

High-end Server

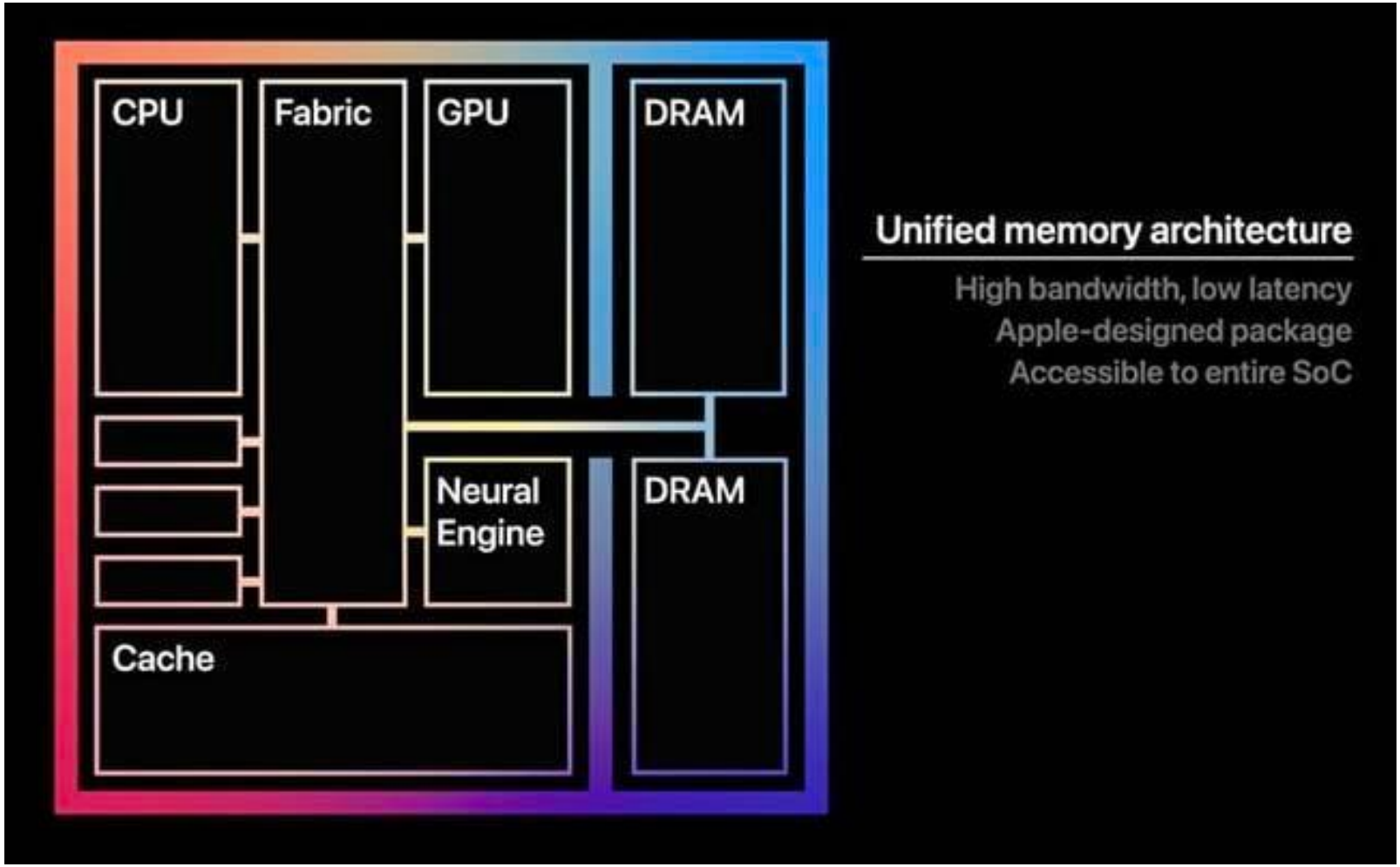


Ultra Portable

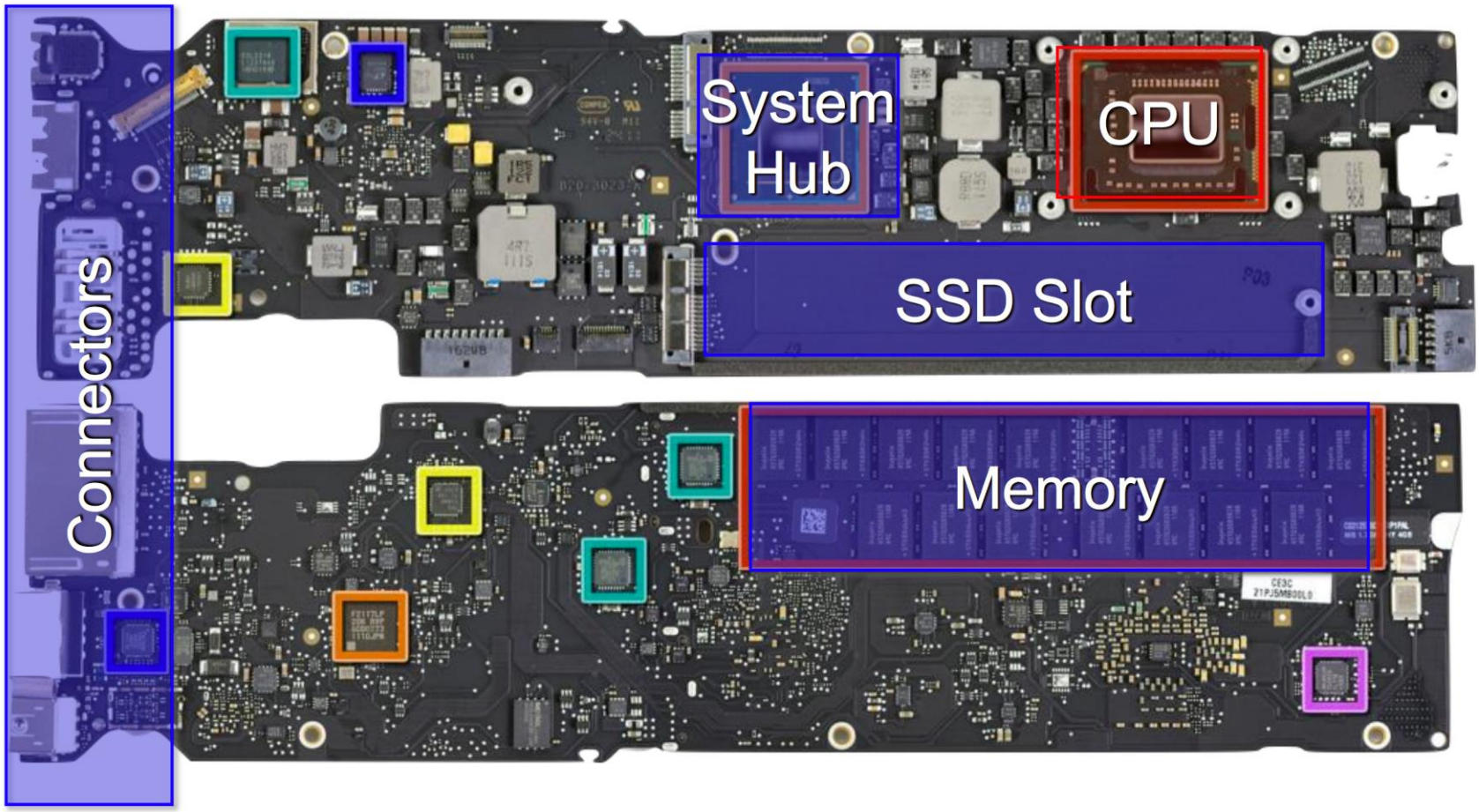
Handheld

- Architecturally, these machines are more similar than different
  - Same parts
  - Different Scale
  - Different Constraints

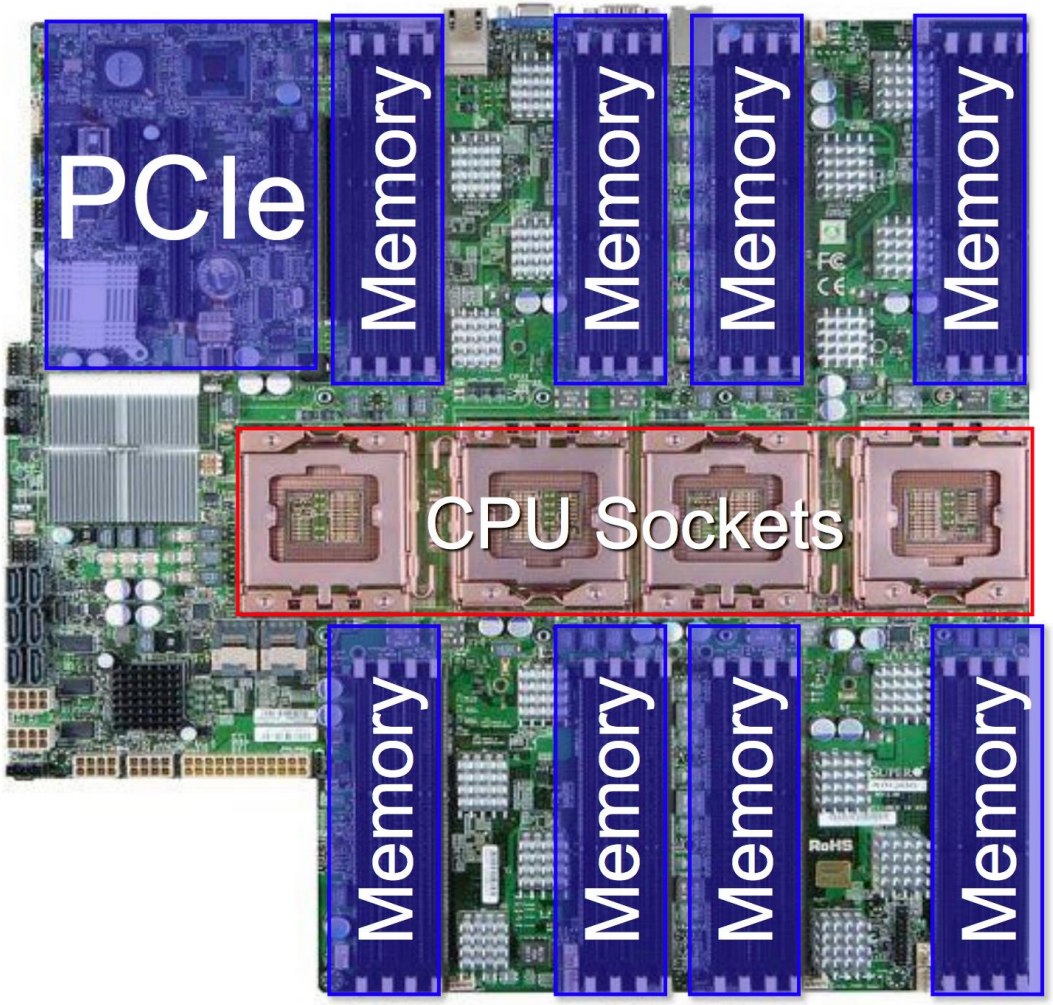
Macbook Air System Diagram (Demonstration)



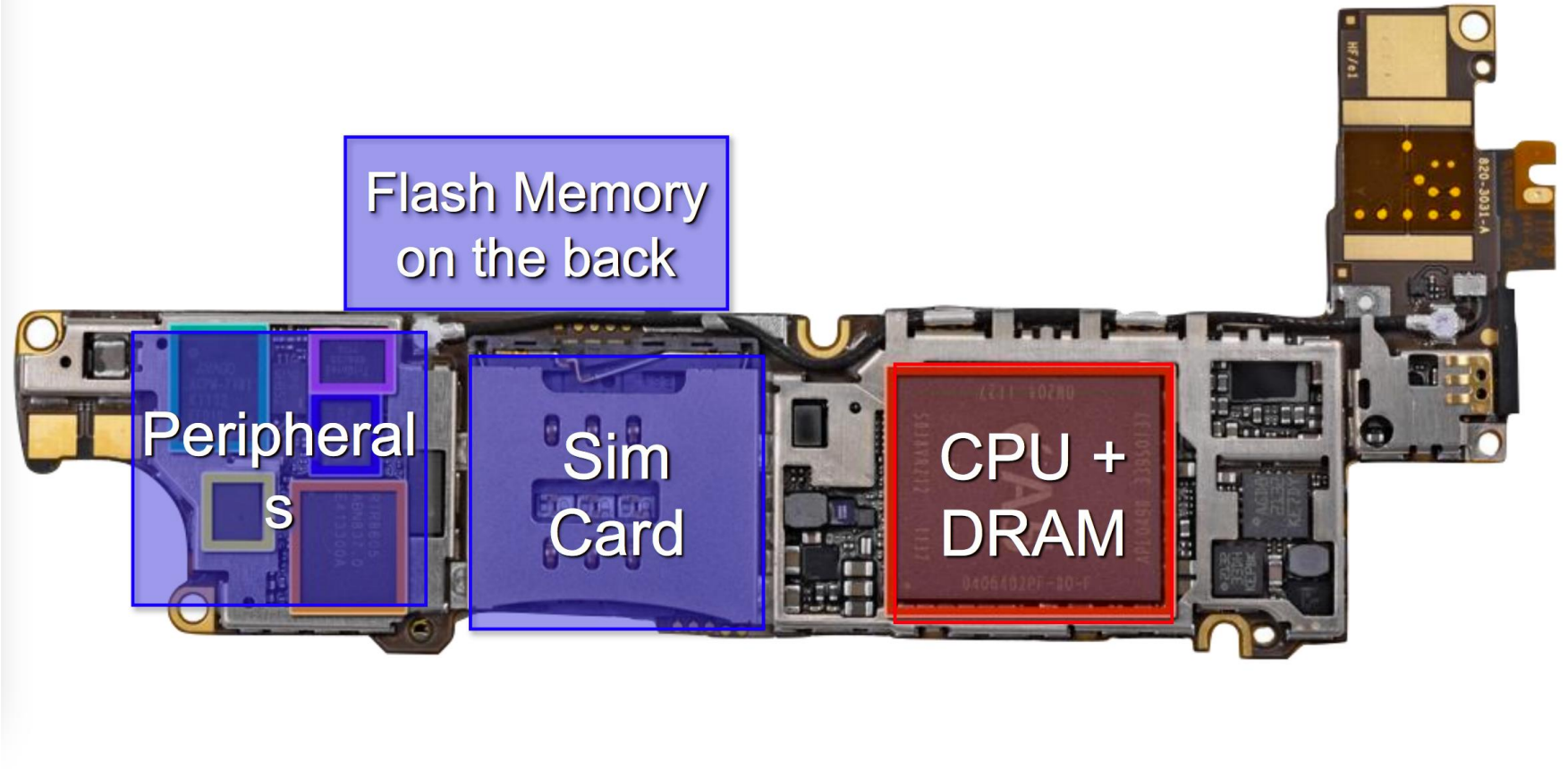
MacBook Air (In reality)



A server


PCIe
Memory
Memory
Memory
Memory
CPU Sockets
Memory
Memory
Memory
Memory

iPhone 4S

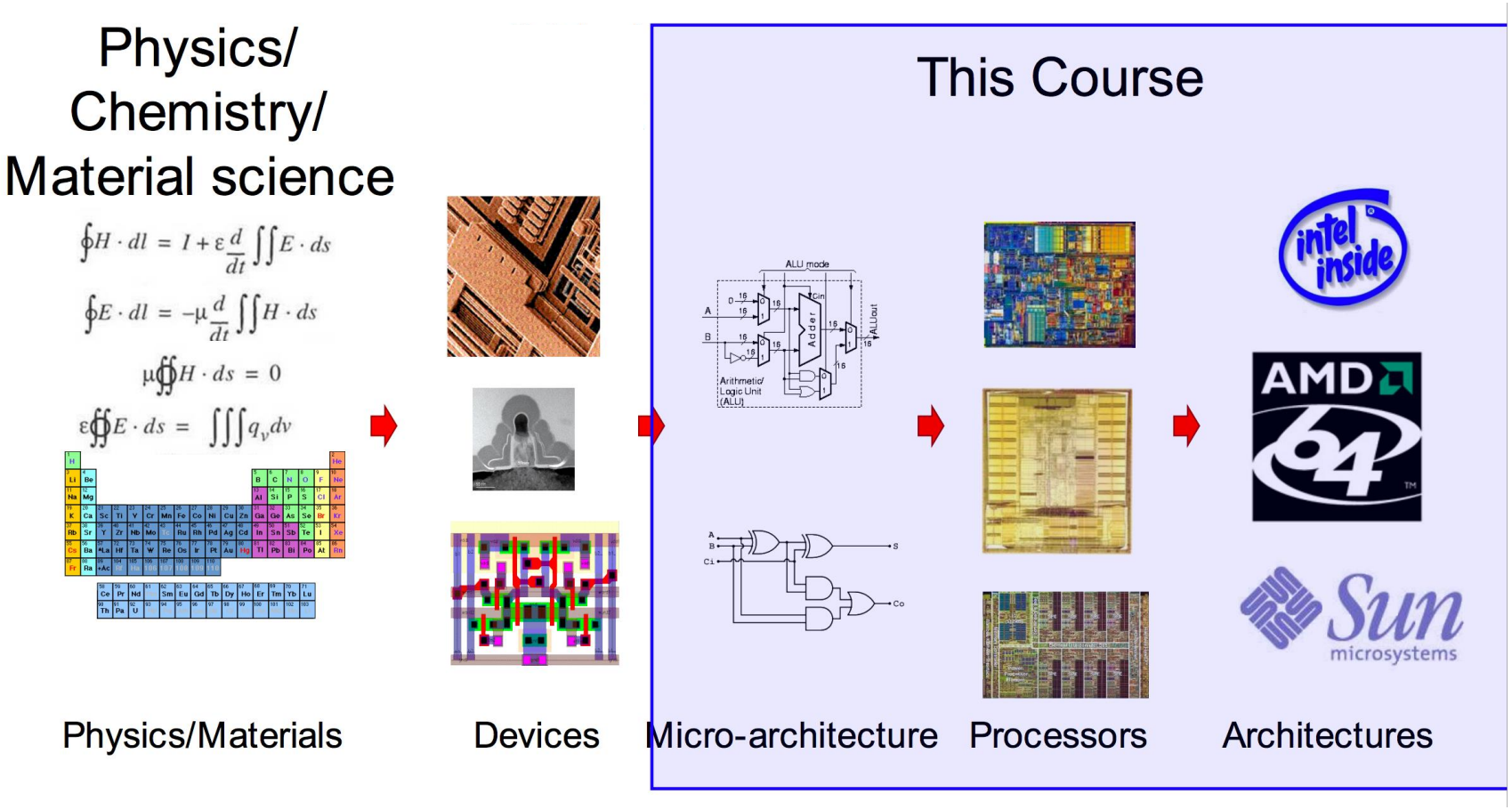


Processors are everywhere

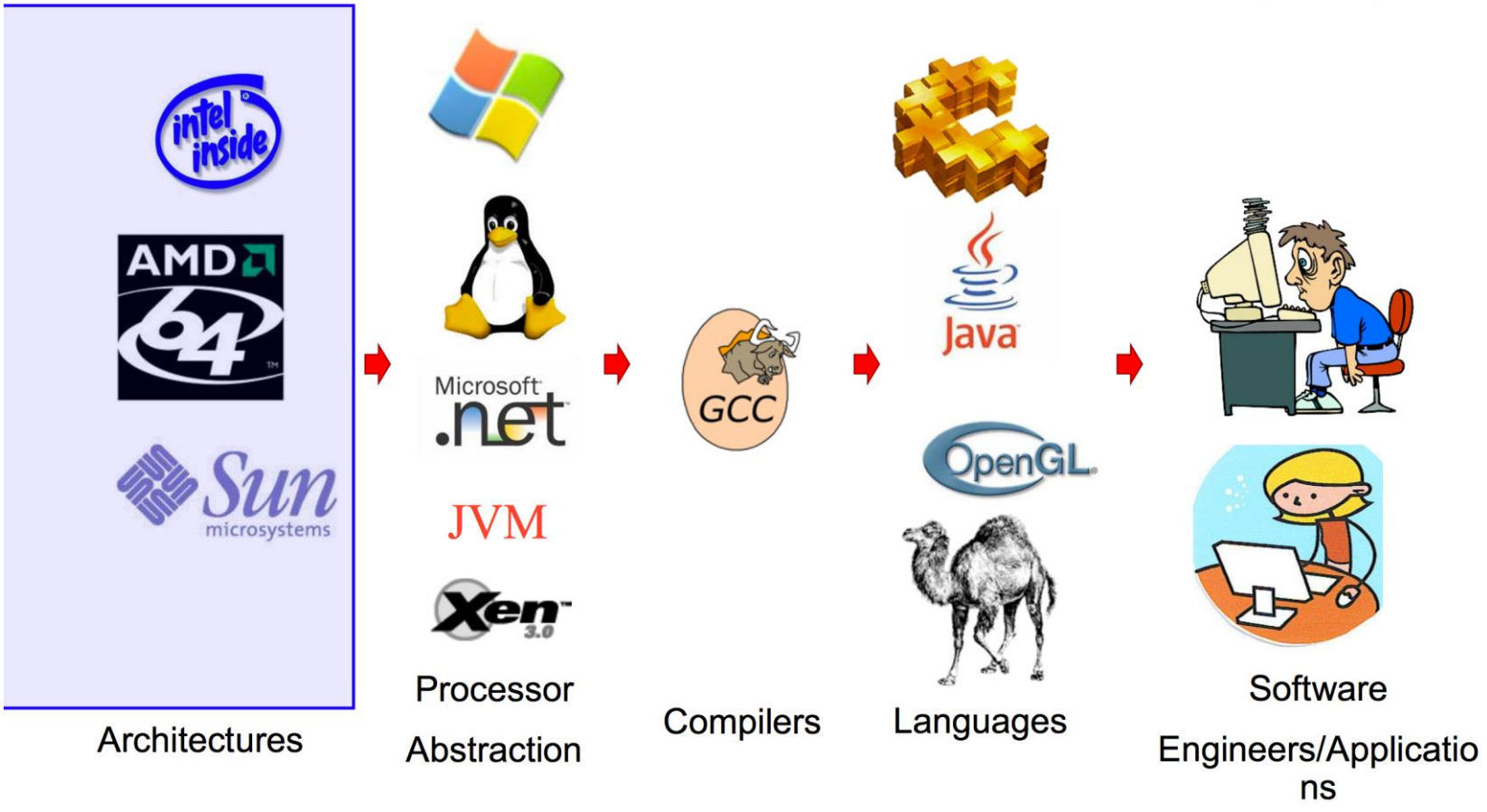
From sand to applications

Cool things happen here…

Abstractions of the physical world…



… for the rest of the system

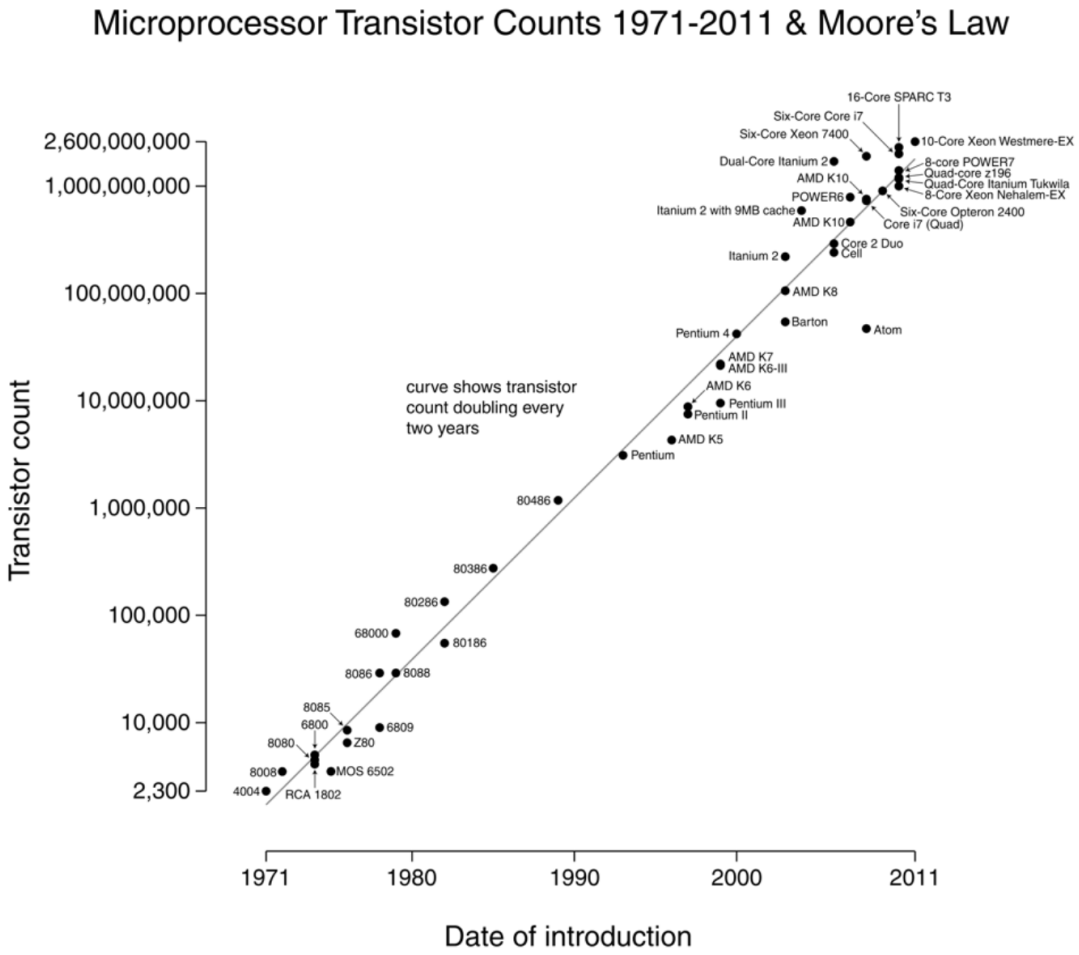


Current state of architecture
&
How did we reach this level?

Moore’s Law

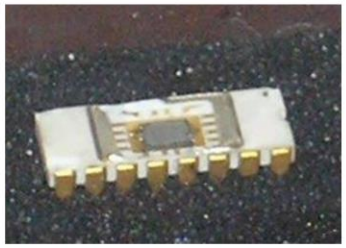
Moore's Law

- # of transistors we can build in a fixed area of silicon doubles every two years

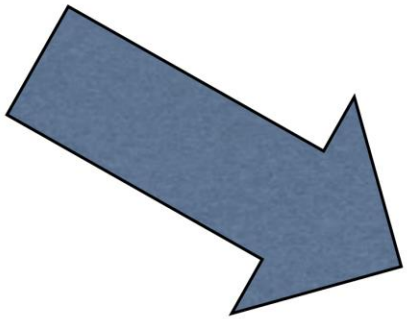


Moore's Law is the most important driver for historic CPU performance gains

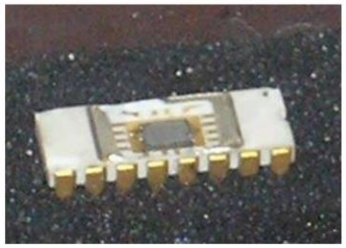
52 years of development



2300 Transistors
Intel 4004



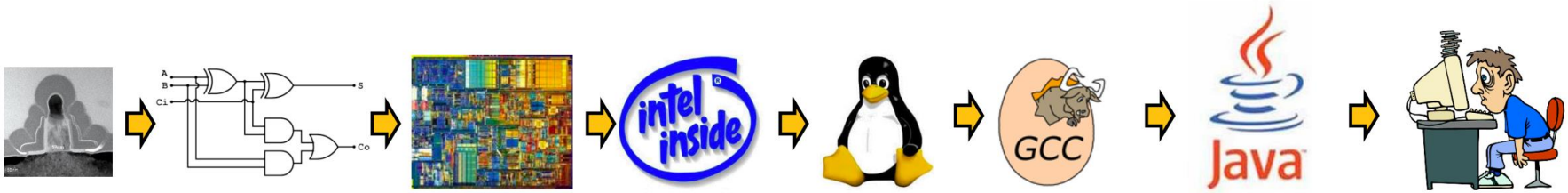
52 years of development



2300 Transistors
Intel 4004

16 billion transistors
6-core CPU, 5-core GPU
16-core neural engine

Since 1940



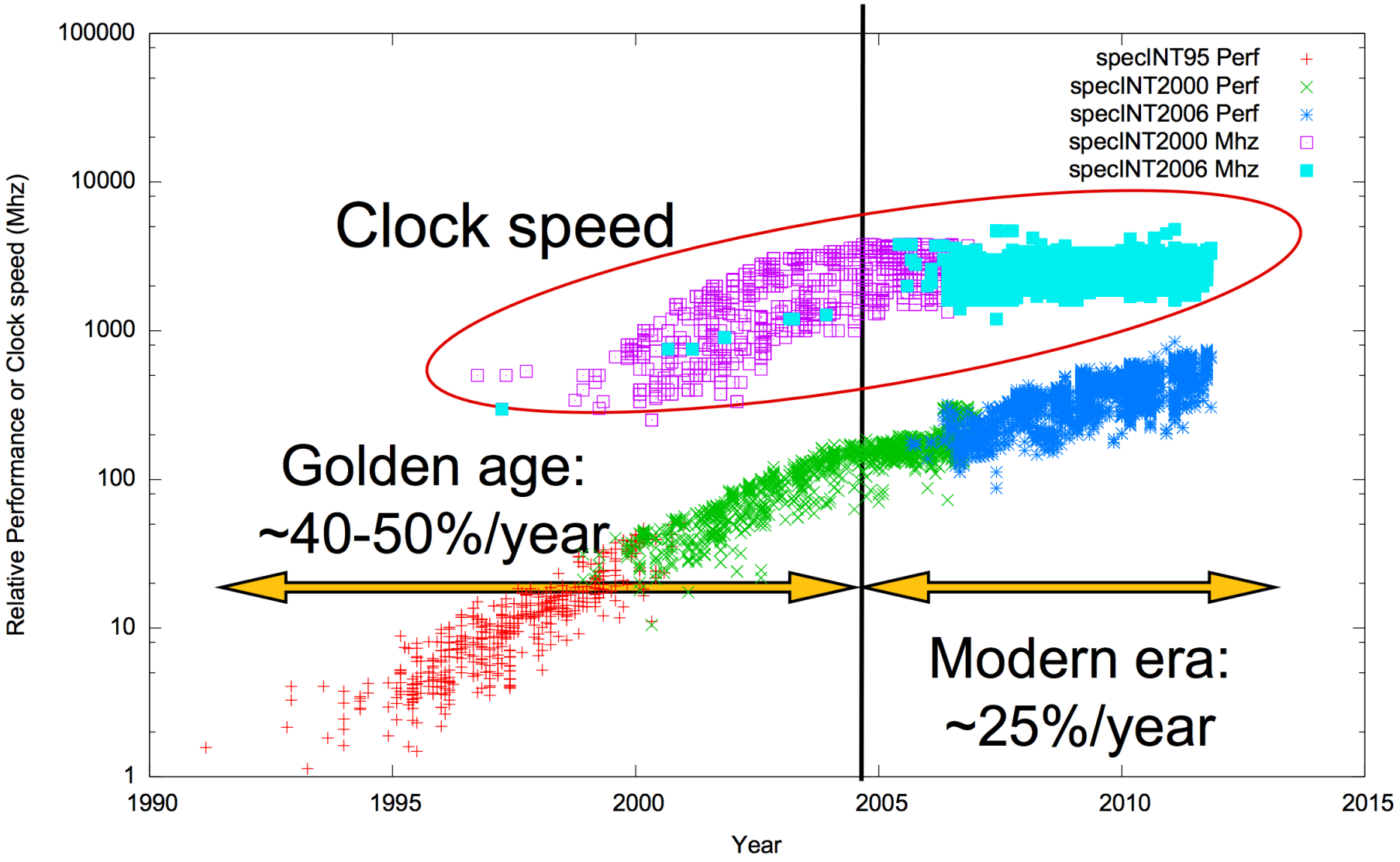
>100,000,000 x speedup

>1,000,000,000 x density
(Moore's Law)

Plug boards -> Java
Hand assembling -> GCC
No OS -> Windows 7

We have used this performance to make computers easier to use, easier to program, and to solve ever-more complicated problems.

Evidence


specINT95 Perf
specINT2000 Perf
specINT2006 Perf
specINT2000 Mhz
specINT2006 Mhz
Clock speed
Golden age:
~40-50%/year
Modern era:
~25%/year
Relative Performance or Clock speed (Mhz)
Year
100000
10000
1000
100
10
1
1990
1995
2000
2005
2010
2015

The end of clock speed scaling

- Clock speed is the biggest contributor to power
  - Doubling the clock speed increases power by 4-8X
  - Clock speed scaling is essentially finished

- Most future performance improvements will be due to architectural and process technology improvements
  - Indicates that computer architecture research is more important than ever

Power and heat

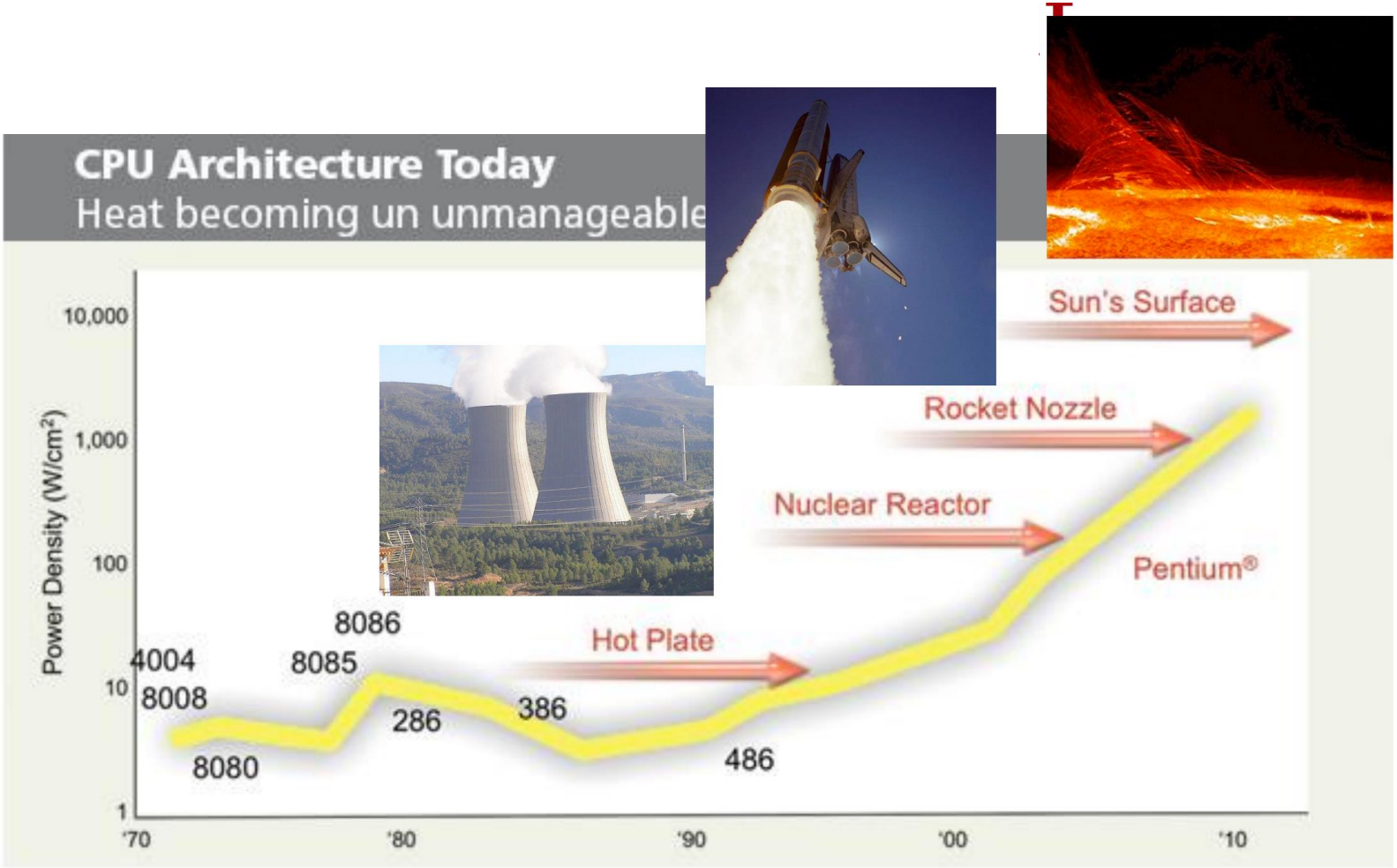
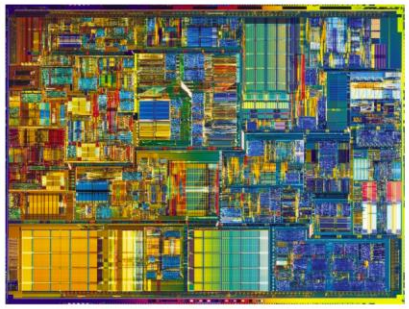


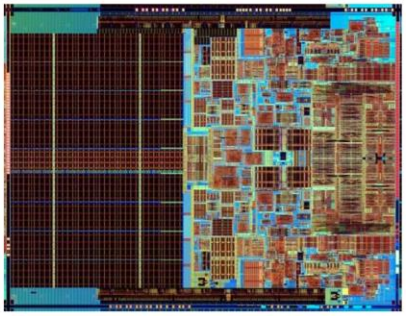
Figure 1. In CPU architecture today, heat is becoming an unmanageable problem. (Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)

The rise of parallelism

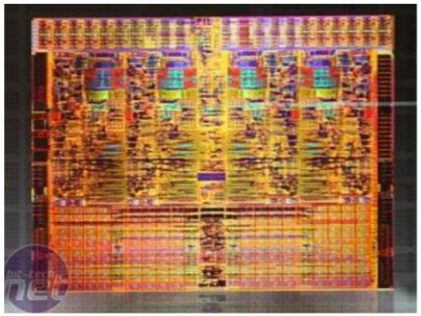
- Multi-processors
  - If one CPU is fast, two must be faster!
  - They allow you to (in theory) double performance without changing the clock speed.
- Seems simple, so why is it becoming so important now
  - Speeding up a single CPU makes everything faster!
    - An application's performance double every 18 months with no effort on the programmer's part
  - Getting performance out of multiprocessors requires work
    - Parallelizing code is difficult, it takes (lots of) work



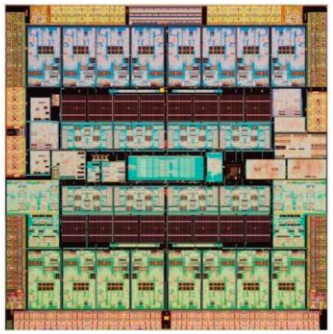
Intel P4
(2000)
1 core



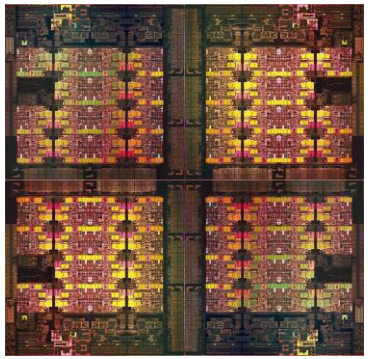
Intel Core 2 Duo
(2006)
2 cores



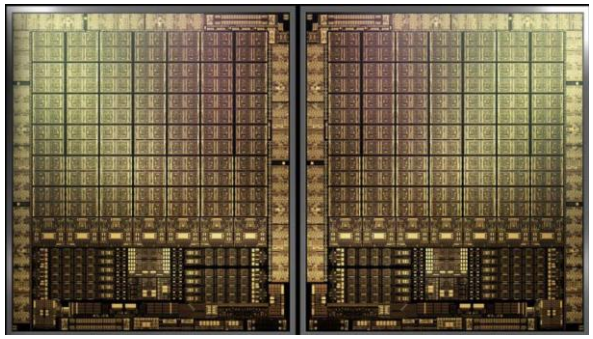
Intel Nahalem
(2010)
4 cores



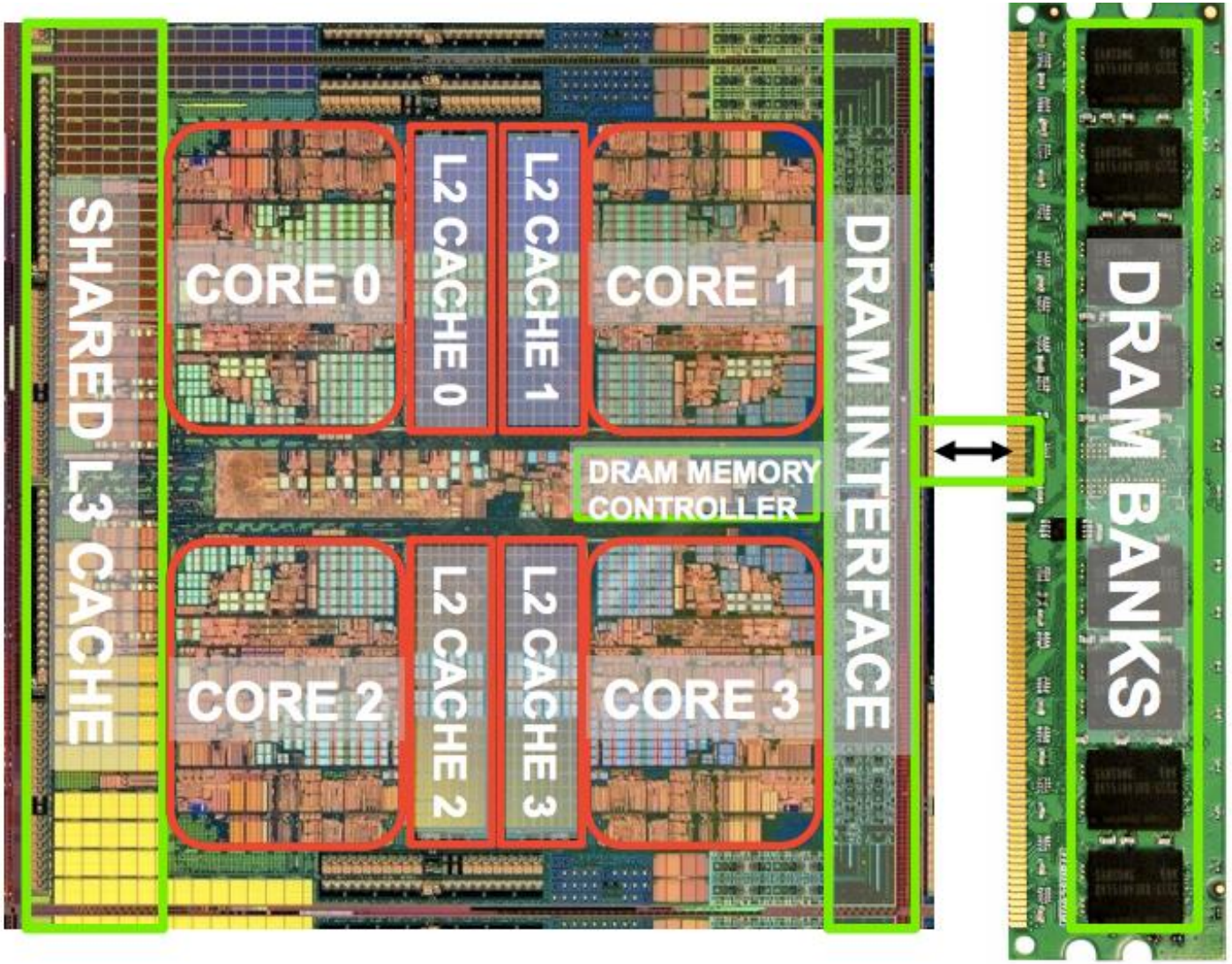
SPARC T3
(2010)
16 cores



Intel Sapphire Rapids
(2022)
32 cores



Nvidia Hopper
(2022)
43008 cores



AMD Barcelona

Power consumption trends

- Dynamic power is proportional to

  activity x capacitance x voltage^2 x frequency

- Energy = power x time = (dynamic_power + leakage_power) x time

Power consumption trends

- Dynamic power is proportional to

  activity x capacitance x voltage^2 x frequency
- Energy = power x time = (dynamic_power + leakage_power) x time
- Question (true or false): low power design is always good?

Power saving techniques

- Dynamic voltage scaling
  - Good: voltage is a big contributor to power
  - Bad: it hurts performance
  - Application scenarios?

- Shut down the power supply
  - Commonly used by cellphones
  - Good: obvious
  - Bad: May hurt performance significantly

A research problem

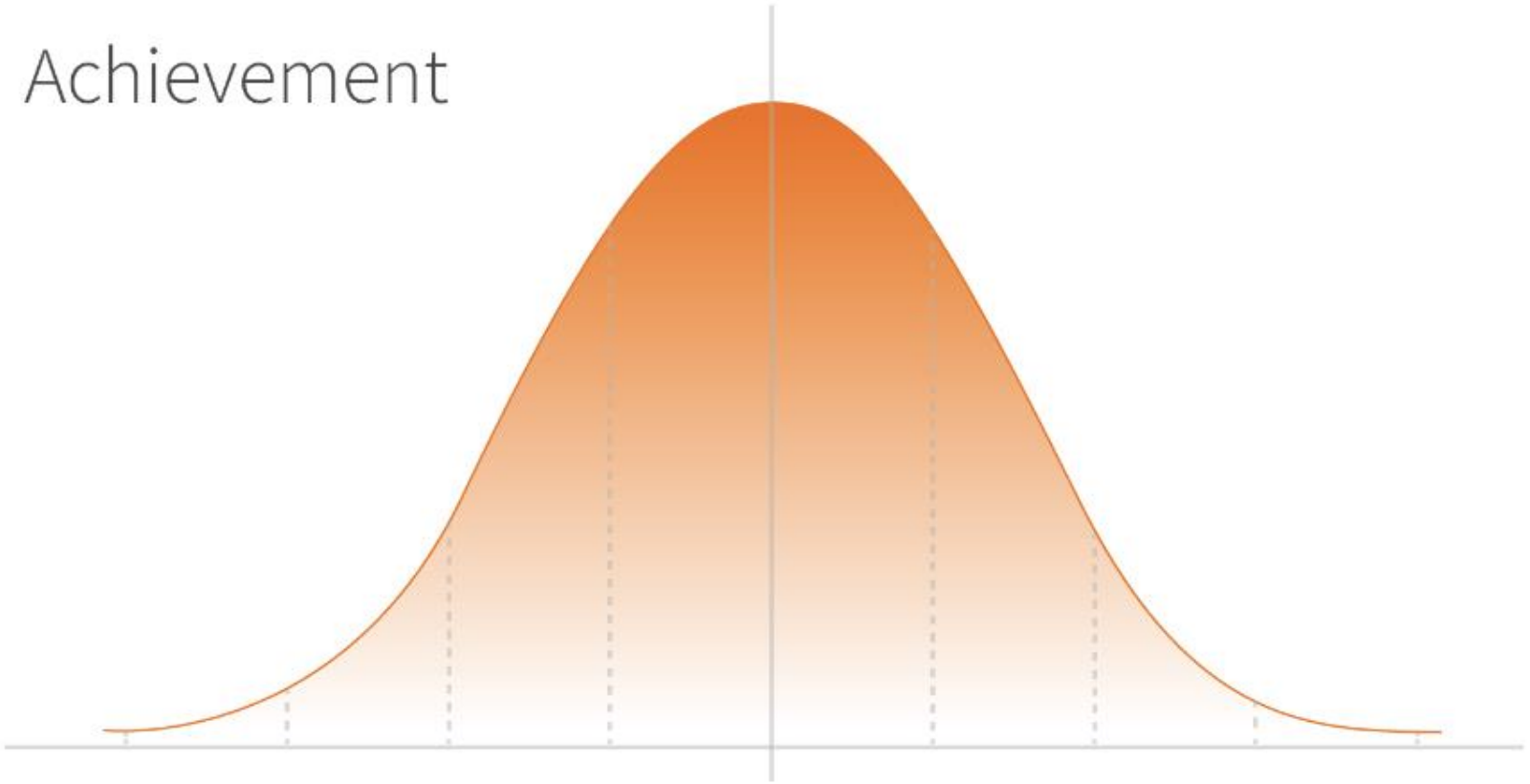
You have a server of 10 cores. Every 0.1s you receive a request, which can be served by one core using 0.1s when the core works at maximum voltage. You can optimize power consumption using the techniques we discussed. What kind of things you can do to reduce power consumption?

Logistics

Goals for this class

- Understand the trends shaping architecture today
  - Measuring Performance
  - Power issues
  - ISAs
  - Memory Hierachies
  - Multi-processors
  - Processor pipelines
  - Out-of-order execution
  - Multithreading
  - Storage systems
  - Data centers
  - GPUs
  - Virtual machines
  - Maybe more, maybe less…

Flow/Break for this class



Course Staff

- Instructor: Ismet Dagli (ismetdagli@mines.edu)
  - Lectures Mondays + Wednesdays (4.30pm to 5.45pm)
  - Office hours Monday 6.00-7.00pm at BB247 (Brown Hall) or by appointment at BB249 .

- TA: Chang Liu (liuchang@mines.edu)
  - Office hours: Friday 10.15am to 11.15am
  - For **homework & projects**

- Course Pages:
  - Canvas
  - Ed discussion

What you need to do

- Class participation 10%
  - Not showing up will impact your grade
  - Read the textbook!
  - I randomly pick students to ask basic questions about required reading and what I've taught
- Homework 20%
  - Four assignments
- Two exams:
  - Midterm 15%
  - Final 25%

Projects

- Cache simulation (30% total)
  - Two phases (each 15%)
  - Code quality matters

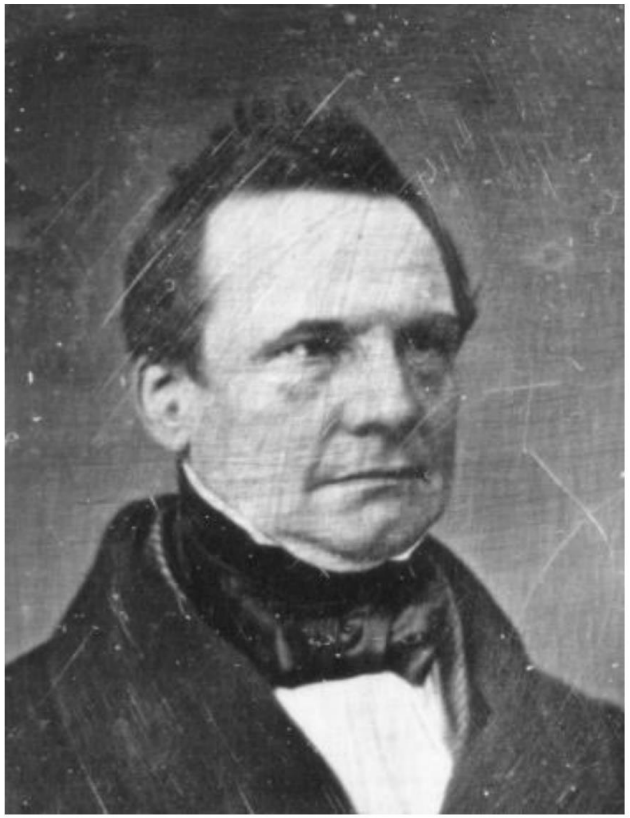
Academic honesty

- Do not cheat
  - Do not copy other people's code
  - Do not copy solutions during the exams
  - Do not copy sentences from other papers or web resources
  - Seriously, cheating leads to unhappy consequences

An incomplete history of computation

Charles Babbage 1791-1871

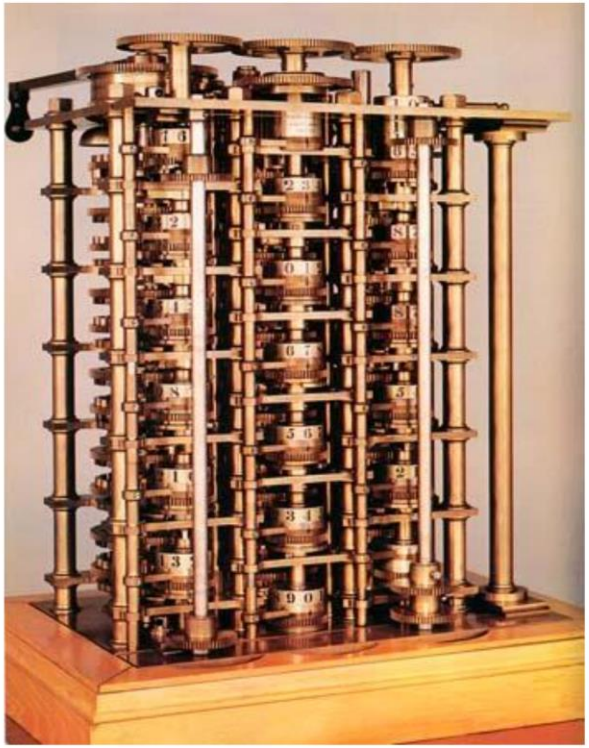
Lucasian Professor of Mathematics,
Cambridge University, 1827-1839
First computer designer



Ada Lovelace 1815-1852

First computer programmer

Difference Engine



- Can compute any 6th degree polynomial
- *Speed:* 33 to 44 32-digit numbers per minute!

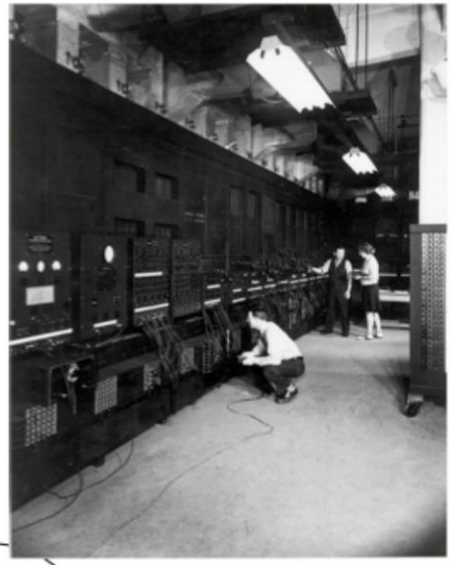
Now the machine is at the Smithsonian

Harvard Mark I

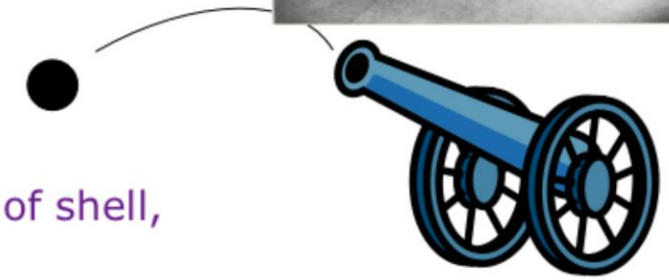
- Built in 1944 in IBM Endicott laboratories
  - Howard Aiken – Professor of Physics at Harvard
  - Essentially mechanical but had some electro-magnetically controlled relays and gears
  - Weighed *5 tons* and had *750,000* components
  - A synchronizing clock that beat every *0.015* seconds

Electronic Numerical Integrator and Computer (ENIAC)

- Inspired by Atanasoff and Berry, Eckert and Mauchly designed and built ENIAC (1943-45) at the University of Pennsylvania
- The first, completely electronic, operational, general-purpose analytical calculator!
  - 30 tons, 72 square meters, 200KW
- Performance
  - Read in 120 cards per minute
  - Addition took 200 μs, Division 6 ms
  - 1000 times faster than Mark I
- Not very reliable!



Application: Ballistic calculations



angle = f (location, tail wind, cross wind,
air density, temperature, weight of shell,
propellant charge, ...)

Electronic Discrete Variable Automatic Computer (EDVAC)

- ENIAC's programming system was external
  - –Sequences of instructions were executed independently of the results of the calculation
  - –Human intervention required to take instructions "out of order"
- Eckert, Mauchly, John von Neumann and others designed EDVAC (1944) to solve this problem
  - –Solution was the *stored program computer*
    - ®"*program can be manipulated as data*"

And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- more than 120 were sold in 1954
and there were orders for 750 more!
- eventually sold about 2000 of them

Users stopped building their own machines.

Why was IBM late getting into computer technology?

Into the 60's…:

Compatibility Problem at IBM

By early 60's, *IBM had 4 incompatible lines of computers!*

701	□	7094
650	□	7074
702	□	7080
1401	□	7010

Each system had its own

- Instruction set
- Peripherals: magnetic tapes, drums and disks
- Programming tools: assemblers, compilers, libraries,...
- market niche: business, scientific, etc....

IBM 360: Implementation

	Model 30	*Model 70*
Main Storage	8K - 64 KB	256K - 512 KB
Datapath	8-bit	64-bit
Circuit Delay	30 nsec/level	5 nsec/level
Local Store	Main Store	Transistor Registers

IBM 360 instruction set architecture completely hid the underlying technological differences between various models.

With “minor” modifications this is the approach we use today.


IBM
SYSTEM 360