

01:198:211

Computer Architecture

David Menendez
davemm@cs.rutgers.edu
Fall 2021

Instructors

- David Menendez
 - `davemm@cs.rutgers.edu`
 - <https://rutgers.webex.com/meet/davemm>
 - Office Hours Tuesday/Thursday 2:00 PM – 4:00 PM (ET)
- Recitations
 - TAs TBA — recitations should begin next week

Lectures

- Lectures will be offered via Webex
 - Listed in the Calendar tool
- Links to recordings will be posted to Canvas
- Attendance is encouraged!

Textbooks

Required

- Bryant and O'Hallaron. *Computer Systems: A Programmer's Perspective*. Prentice Hall.

Recommended

- Patterson and Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufman
- Jens Gustedt, *Modern C*. Manning Publications.
 - <http://freecomputerbooks.com/Modern-C-by-Jens-Gustedt.html>
 - Any good reference for C should be sufficient
 - There are many on-line resources, e.g. https://en.wikibooks.org/wiki/C_Programming

What You Should Know

- Prerequisites: 198:112
 - You should know some math
 - You should know some basic algorithms and data structures
 - You should know at least one programming language (e.g., Java)
 - You should know how to write, run, and test programs
- You will need to learn
 - Read the textbook
 - Participate in class
 - Start assignments early

**How can we write programs
to get good performance on
modern architectures?**

What You Will Learn

- A high-level language suitable for systems programming (C)
- A low-level language close to the machine (Assembly)
- Major hardware components in computer systems
- How hardware components are built from digital logic
- How programs are actually executed by the hardware
- How to understand and improve the performance of programs

Course Structure

- Assignments
 - 5 programming assignments
- Tests (on-line via Canvas)
 - Short, primarily multiple choice
 - Intended as a check on understanding
- Comprehensive final exam (on-line via Canvas)
- Expectations
 - Attend lectures and recitation
 - Read assigned material before lecture
 - Read carefully and think about assignments
 - Ask questions
 - Start assignments early

**Seriously, start the
assignments early**

No Late Assignments

- We will not accept late assignments
- Programming assignments will be submitted through Canvas
 - Deadlines will be enforced by Canvas
 - You can submit multiple times, if you are working close to the deadline and worried you will miss it
 - I recommend submitting a day in advance
- Do not submit via e-mail

Collaboration vs Cheating

- Your submissions must be your work
 - No code copied from other students
 - No code copied from on-line sources
- Discussing problems with other students is encouraged
 - Do not look at anyone else's code
- Do not post code on-line!
- We will be checking submissions for cheating
 - If we find evidence of copying, everyone involved will be reported
 - If you are having trouble, talk to me or a TA

Rough Schedule

- Recent trends in hardware
- C Programming
- Information Representation
- Assembly programming (Intel X86)
- Digital Logic
- Processor Architecture
- Memory Hierarchy
- Compiling and Linking
- Measuring and Optimizing Performance



Time Permitting!

Programming Assignments

- Languages will be C and Assembly in a Linux environment
 - Start early
 - Learn how to use your tools
 - Don't rely on luck or trying every possible solution
- We will use the Instructional Lab
 - <https://resources.cs.rutgers.edu/docs/instructional-lab/>
 - Sign up for an account if you do not have one
 - Includes directions for setting up remote access
 - You can write your program anywhere, but we will test in the lab

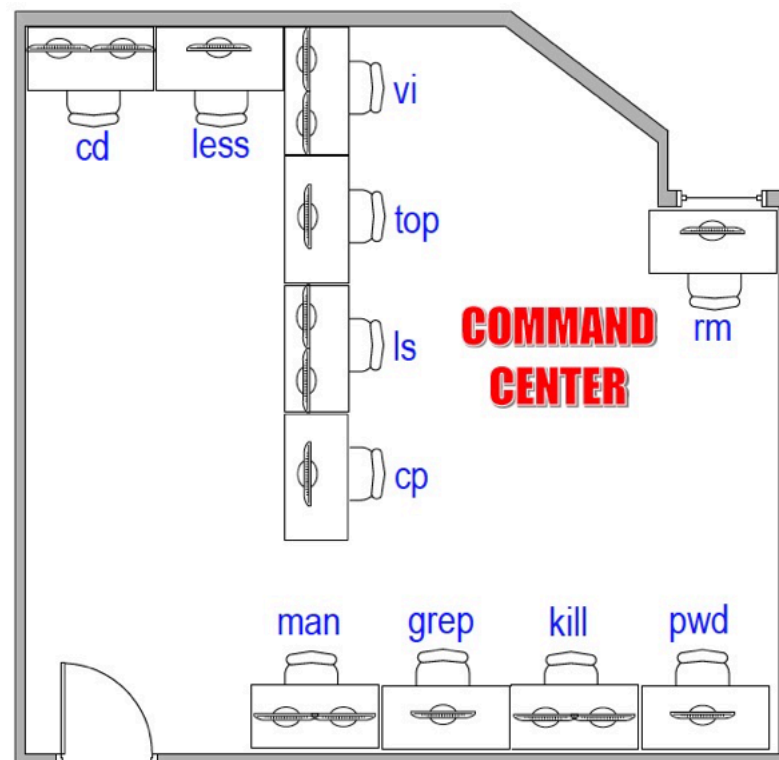
Instructional Lab

- For this class, we will use the machines in the Command Center, Cave, and Meltdown clusters as references

- Obtain the current list of active machines at

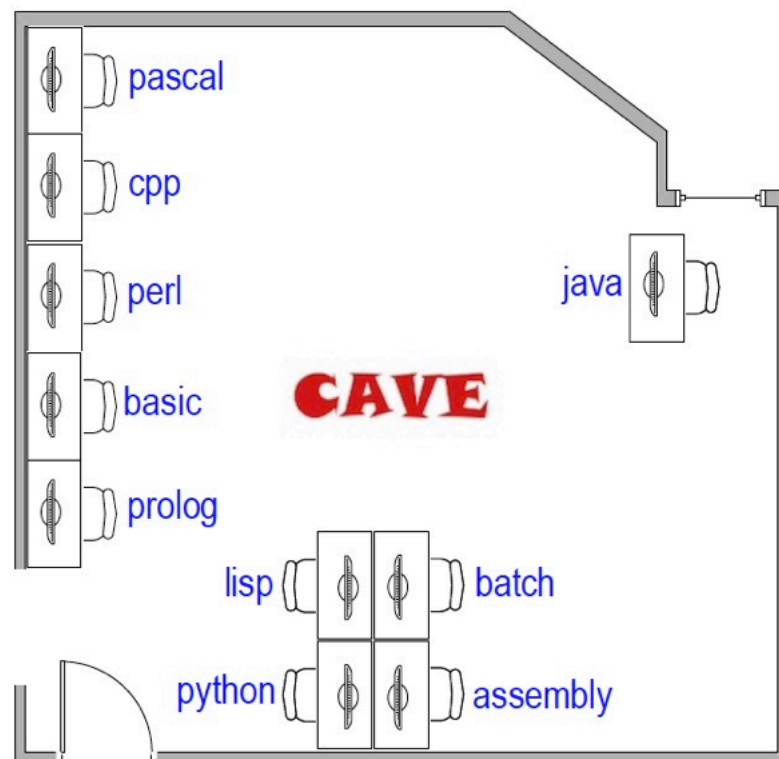
<https://report.cs.rutgers.edu/nagiosnotes/iLab-machines.html>

- These are actual machines located in the Hill Center Annex, rooms 248, 252, and 254
- Your files are stored in a network file system, so you may log into any machine
- You may use the machines in person, or log in remotely using ssh, X2Go, or Microsoft Remote Desktop
- You can transfer files between your machine and the iLab using scp or sftp
- *Do not use ilab.cs.rutgers.edu* — it is a different cluster with different software

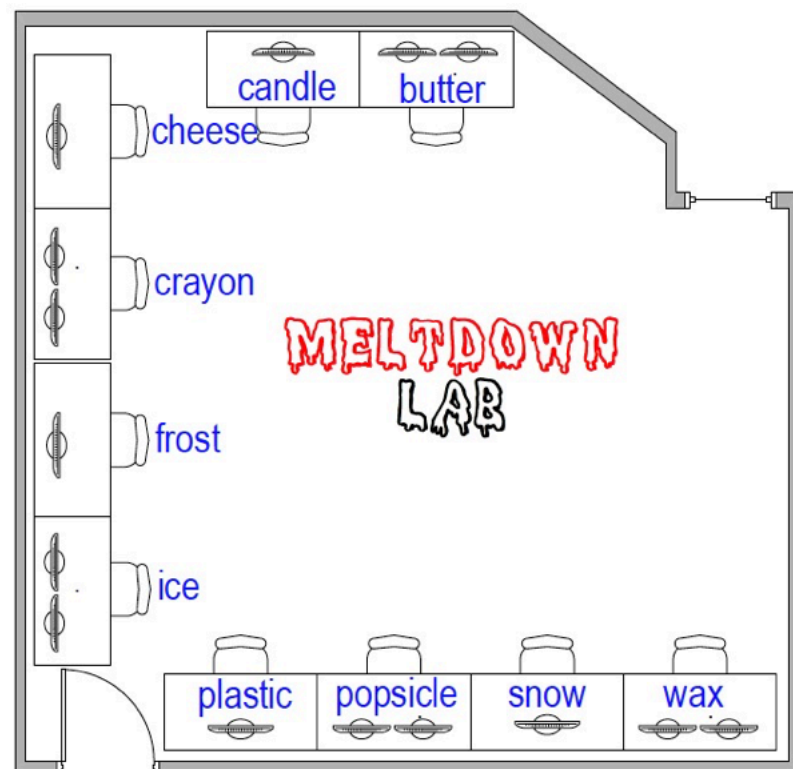


H248 Machines Status (11 units)					
Hostname	GPU	Logins	Load	Status	Misc
cd.cs.rutgers.edu	idle	1	4.09	up	↕ 🔥
cp.cs.rutgers.edu	idle	1	0.05	up	↕ 🔥
grep.cs.rutgers.edu	idle	0	0.02	up	↕ 🔥
kill.cs.rutgers.edu	idle	5	0.07	up	↕ 🔥
less.cs.rutgers.edu	idle	0	0.05	up	↕ 🔥
ls.cs.rutgers.edu	idle	0	0.08	up	↕ 🔥
man.cs.rutgers.edu	idle	0	0.07	up	↕ 🔥
pwd.cs.rutgers.edu	idle	0	0.02	up	↕ 🔥
rm.cs.rutgers.edu	idle	0	0.04	up	↕ 🔥
top.cs.rutgers.edu	idle	3	0.31	up	↕ 🔥
vi.cs.rutgers.edu	idle	1	0.52	up	↕ 🔥

OS: CentOS 7.6 | **CPU:** Intel® Core™ i7-7700 @ 3.60GHz - 8 vCores | **Memory:** 32GB
Make/Model: HP Prodesk 600 G3 Desktop | Aug 2017 | **CPUMark:** 10727
GPU Specs: GT 730 | **Memory:** 2 GB | **CUDA Cores:** 384 | **CUDA capability :** 3.0 | **Version:** 9.0



H252 Machines Status (10 units)					
Hostname	GPU	Logins	Load	Status	Misc
assembly.cs.rutgers.edu	-	0	0.05	up	↕ 🔥
basic.cs.rutgers.edu	-	0	0.04	up	↕ 🔥
batch.cs.rutgers.edu	-	0	0.06	up	↕ 🔥
cpp.cs.rutgers.edu	-	0	0.05	up	↕ 🔥
java.cs.rutgers.edu	-	0	0.08	up	↕ 🔥
lisp.cs.rutgers.edu	-	0	0.06	up	↕ 🔥
pascal.cs.rutgers.edu	-	0	0.09	up	↕ 🔥
perl.cs.rutgers.edu	-	0	0.16	up	↕ 🔥
prolog.cs.rutgers.edu	-	0	0.12	up	↕ 🔥
python.cs.rutgers.edu	-	0	0.08	up	↕ 🔥



H254 Machines Status (10 units)					
Hostname	GPU	Logins	Load	Status	Misc
ice.cs.rutgers.edu	idle	2	0.03	up	↕ 🔥
snow.cs.rutgers.edu	idle	0	0.02	up	↕ 🔥
butter.cs.rutgers.edu	idle	0	0.04	up	↕ 🔥
cheese.cs.rutgers.edu	idle	1	0.04	up	↕ 🔥
candle.cs.rutgers.edu	idle	0	0.03	up	↕ 🔥
frost.cs.rutgers.edu	idle	0	0.04	up	↕ 🔥
popsicle.cs.rutgers.edu	idle	1	0.04	up	↕ 🔥
plastic.cs.rutgers.edu	idle	0	0.05	up	↕ 🔥
crayon.cs.rutgers.edu	idle	0	0.04	up	↕ 🔥
wax.cs.rutgers.edu	idle	0	0.05	up	↕ 🔥

Grading Assignments

- Most programming assignments will be tested by an automated system
- You will be provided a copy of the auto-grader and a subset of the tests
- Your program **must** work with the auto-grader on the iLab machines
- It is your responsibility to make sure auto-grader can compile and execute your program
 - Test early and often
 - If you are having problems with the auto-grader, ask me or a TA
- It is your responsibility to make sure your submission is formatted correctly

Grading

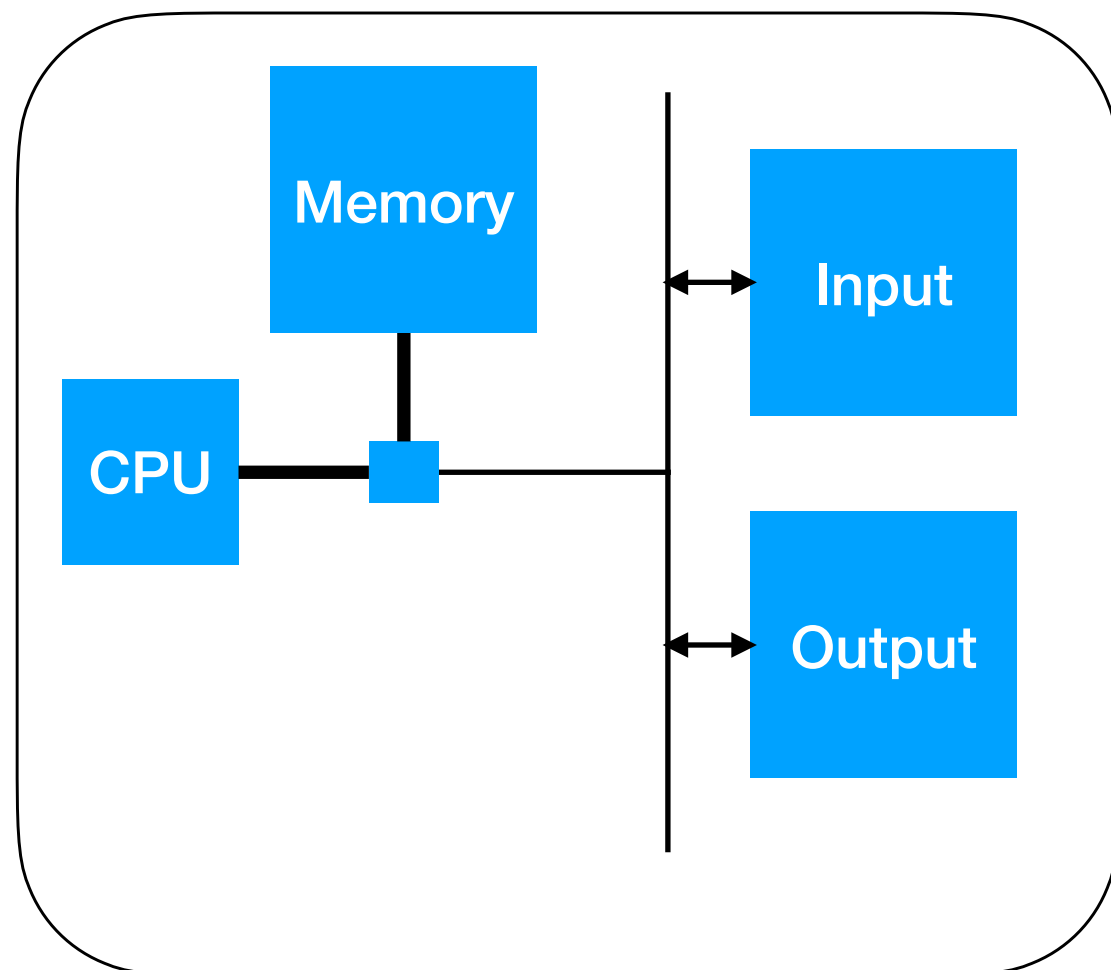
- Breakdown:
 - 25% tests
 - 25% final exam
 - 50% programming assignments
- No make-up exams, except for university-sanctioned reasons
 - You must inform me before the exam
 - Informing me does not guarantee that I will agree



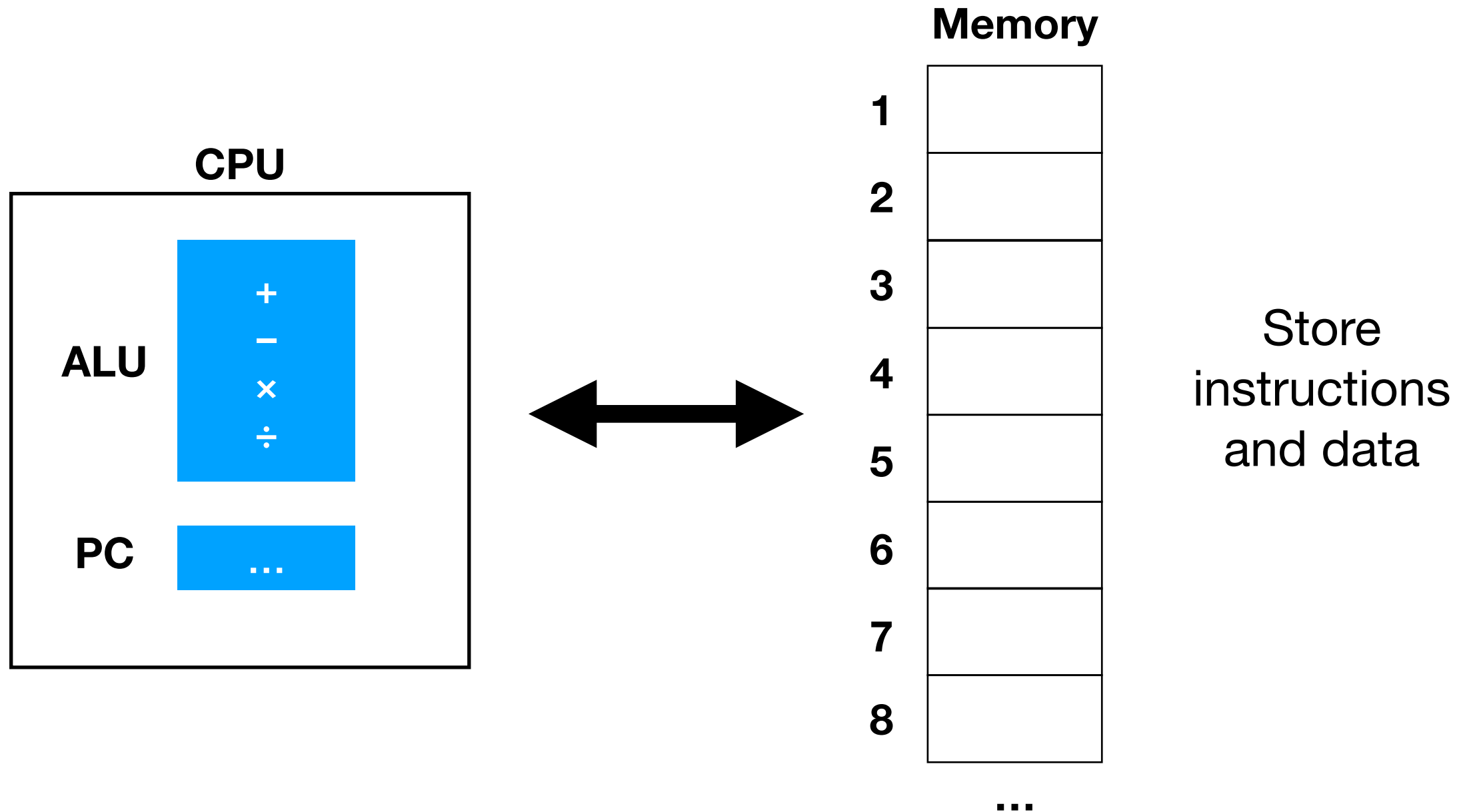
What is a computer?

Main Components

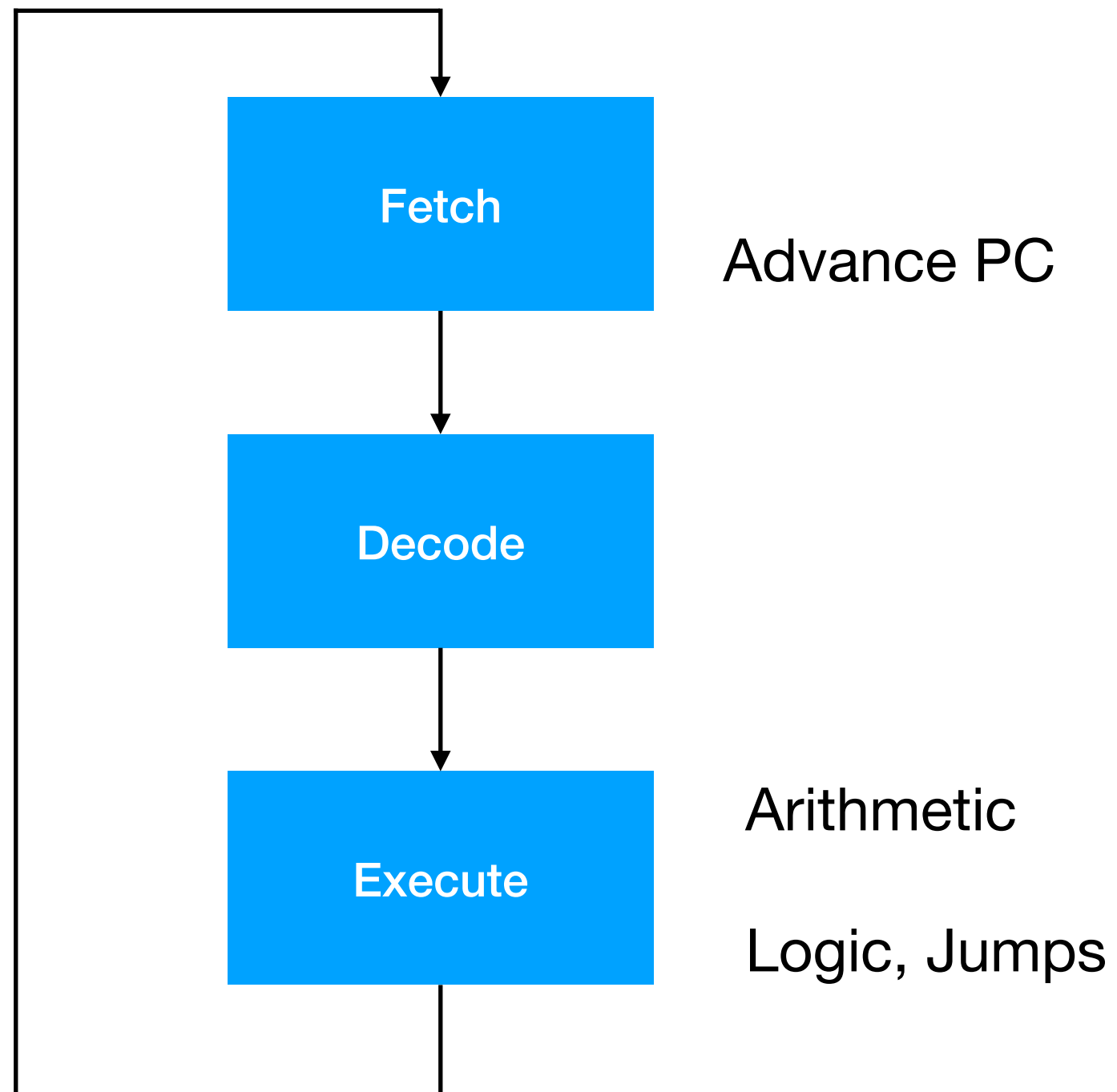
- CPU
- Memory
- Bus
- I/O devices
 - Human interface
 - Storage
 - Networking
 - Graphics



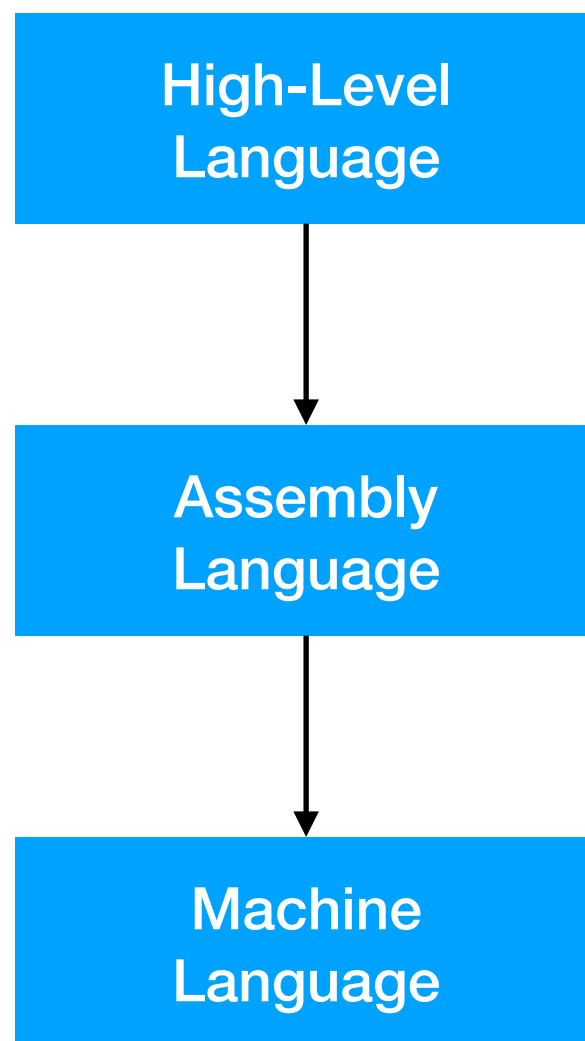
Von Neumann Model



Basic CPU Function



Running on Hardware

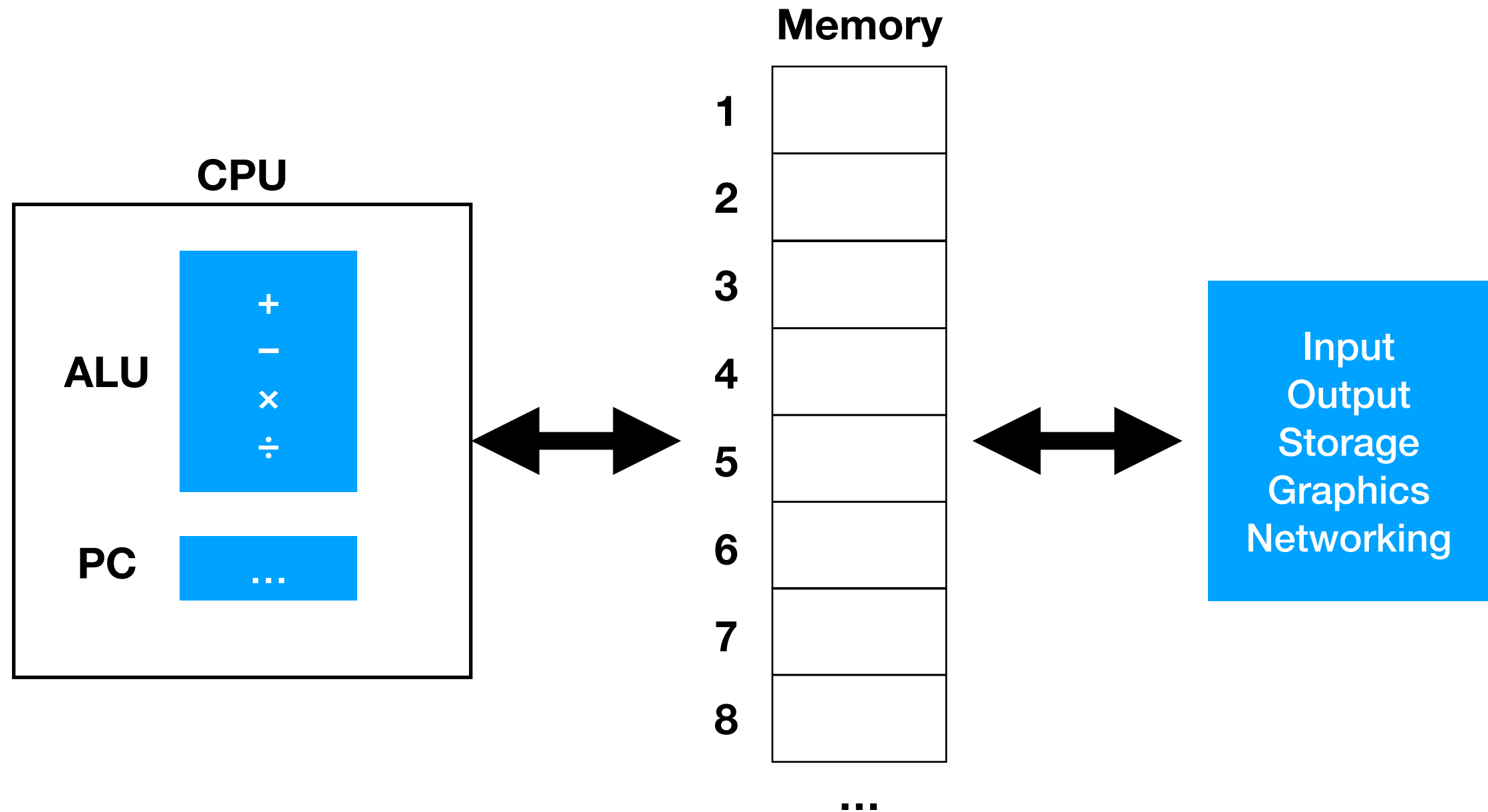


`x = x + y;`

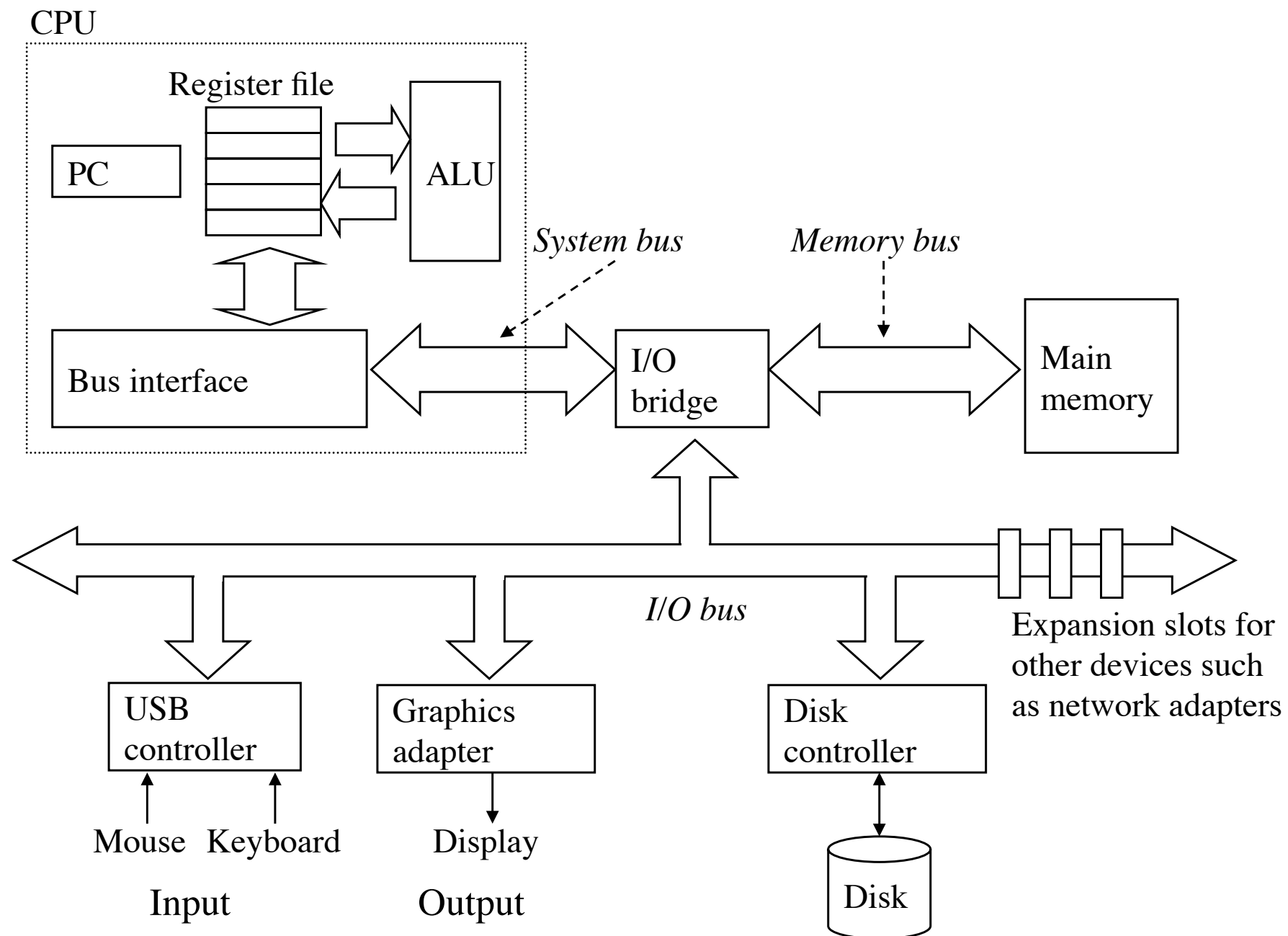
`mov -0x8(%esp), %ebx
add %ebx, %eax`

`7F 45 4C 46 01 01 01...`

Input/Output



Von Neumann in Practice



The Operating System

- Programs run on the hardware, calling into OS for some services
- On modern systems, OS and hardware collaborate for some features
 - e.g., virtual memory, multitasking

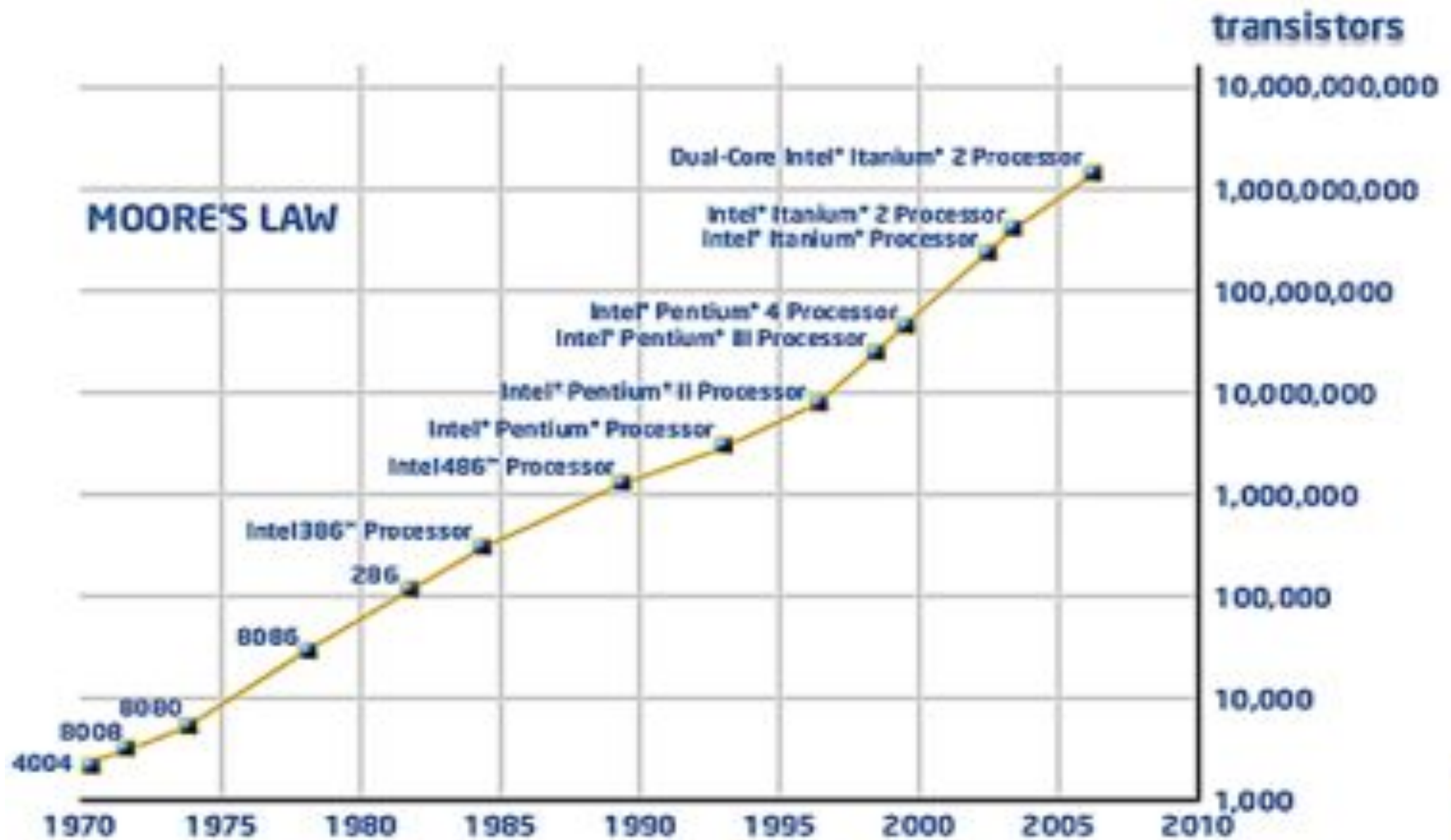
Computer Architecture

- How to build components
- How to design and build systems using components
- We will cover:
 - Basics of current components and systems
 - How programs run on current systems
 - How current architectures affect programs written in high-level languages

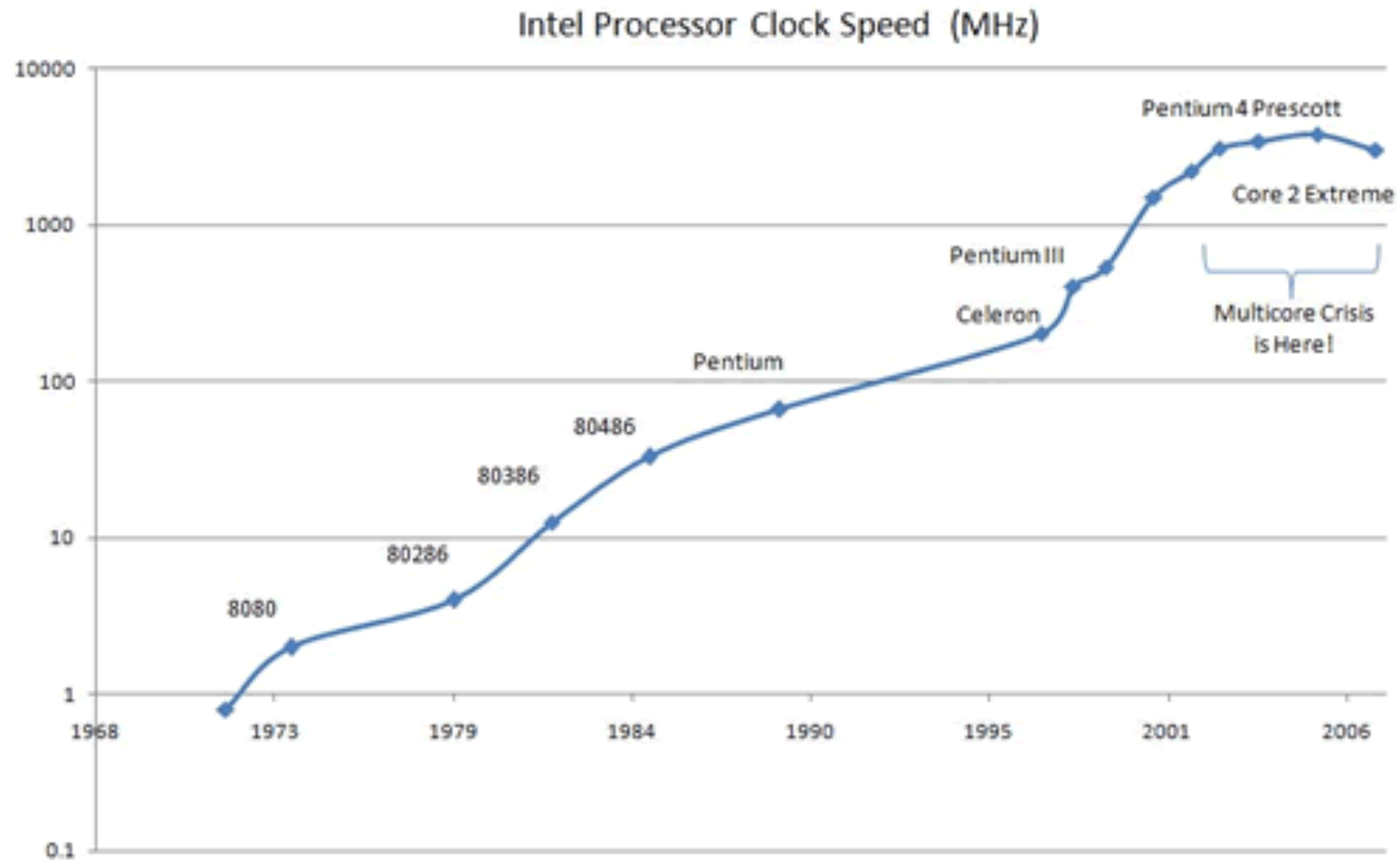
Moore's Law

- Gordon Moore was an Intel Engineer
- Observed that the number of transistors on a chip doubles every 18 months
- Exponential growth also seen elsewhere:
 - Processor speed doubles every 18 months
 - Memory capacity doubles every two years
 - Disk capacity doubles every year

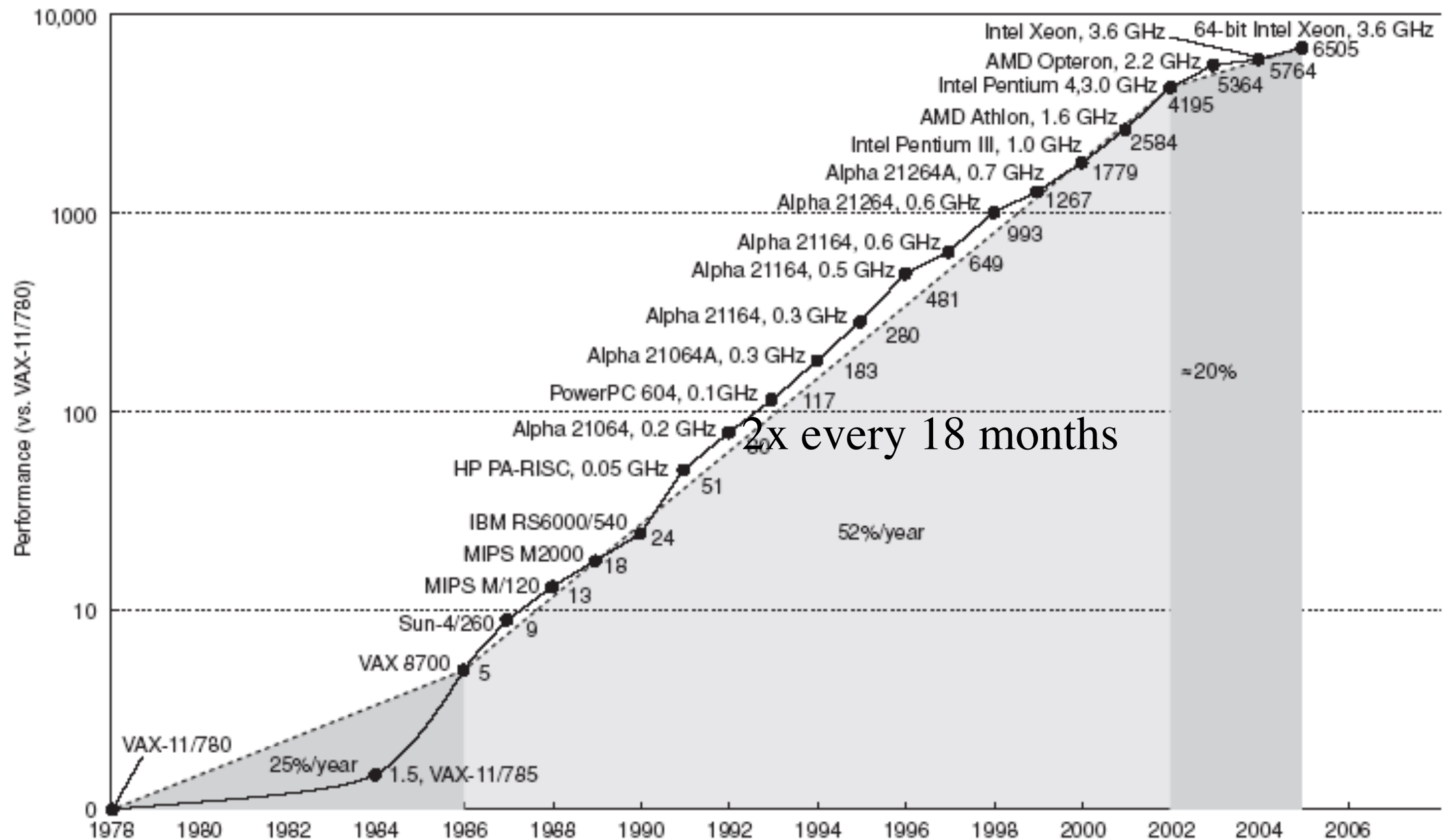
Transistors on a Chip



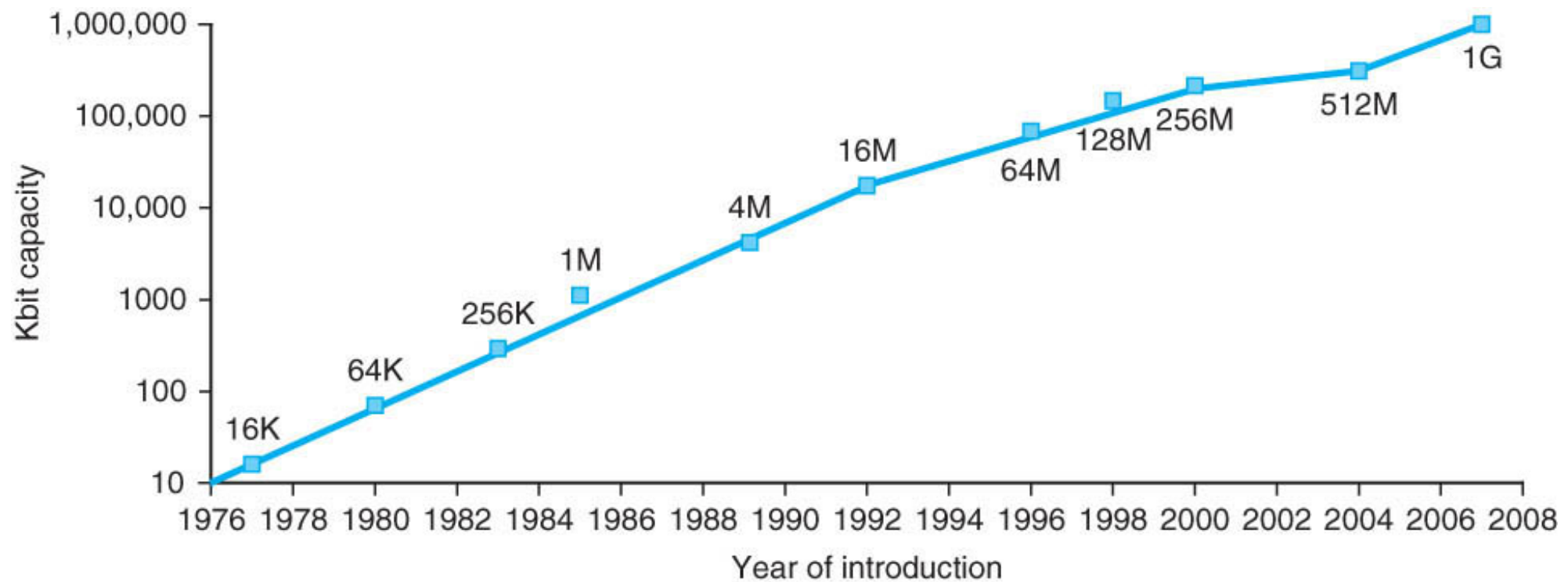
Clock Speed



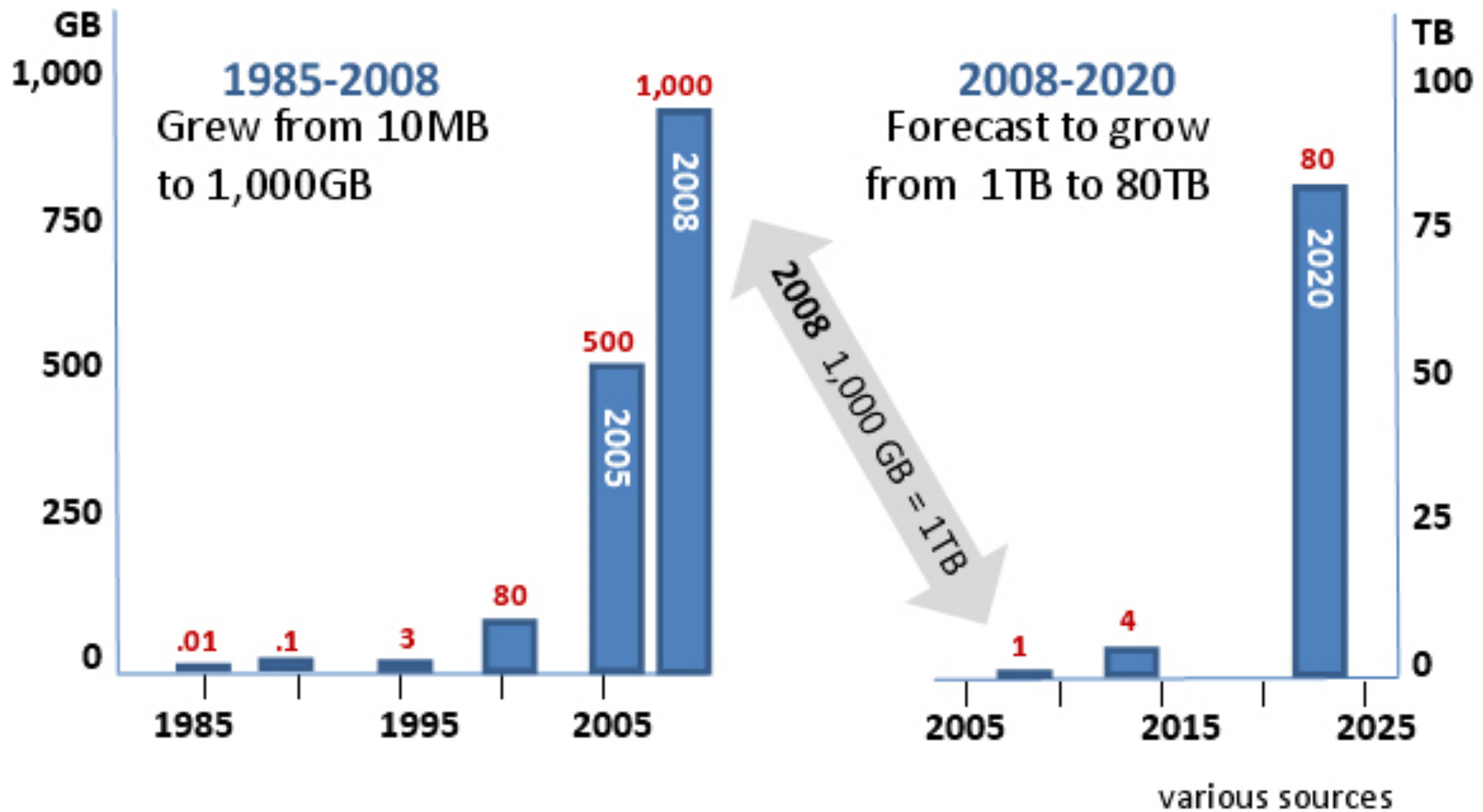
Processor Performance



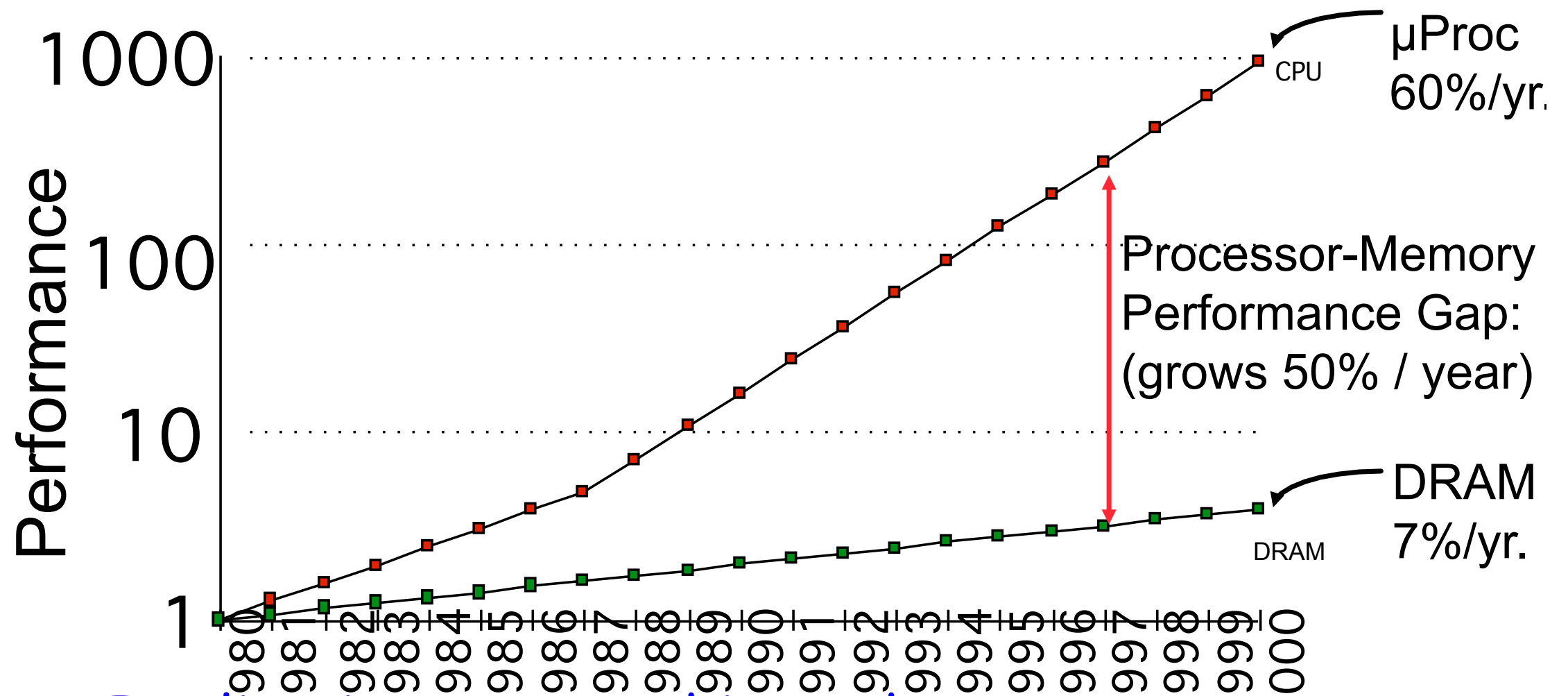
Memory Capacity



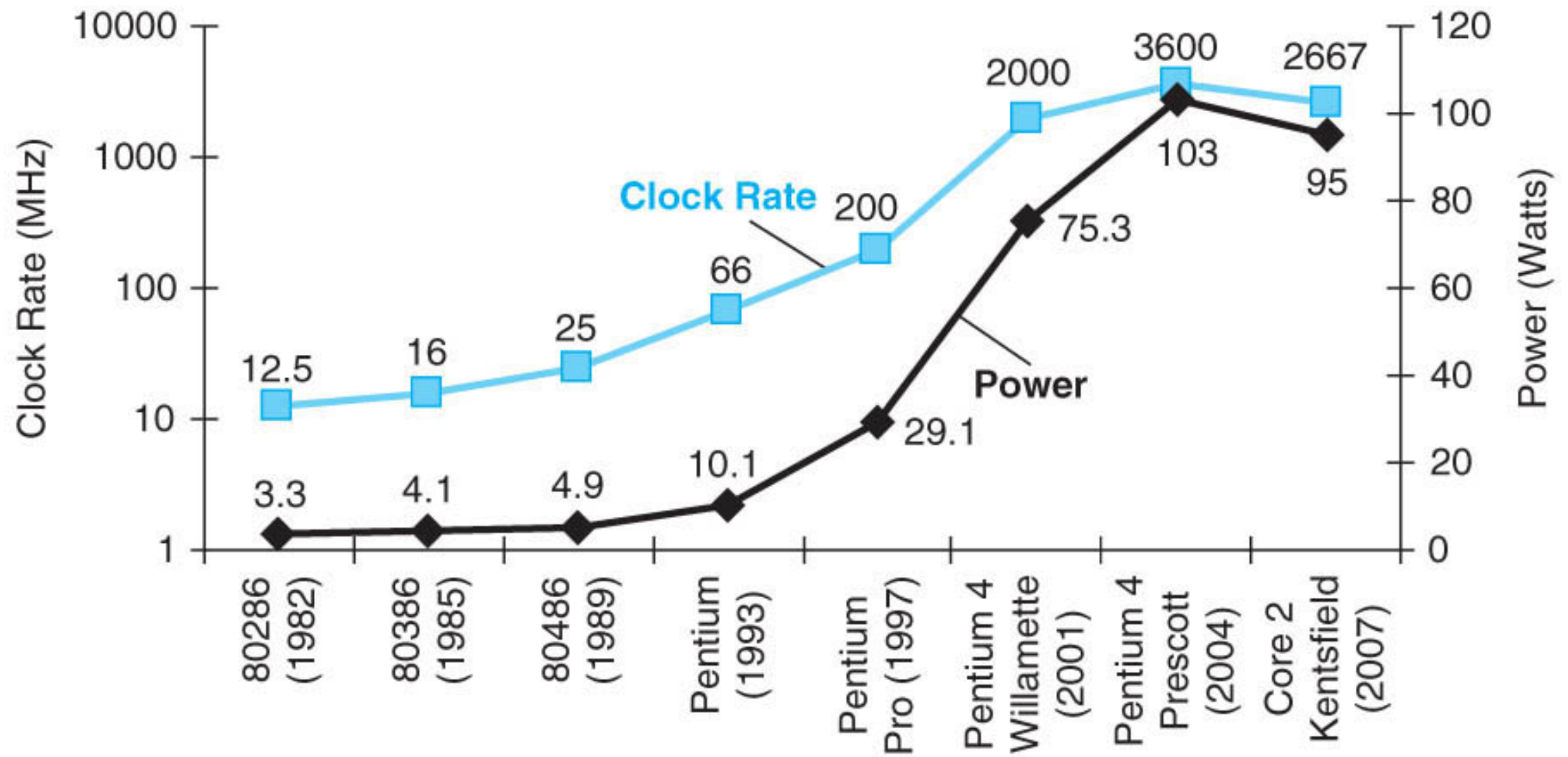
Disk Capacity



CPU–Memory Performance Gap



Power Wall



Summary

- Modern systems are complex
 - Don't expect much speed-up for single-threaded programs
- Faster processors and systems enable new applications
- Understanding these systems is crucial
 - What are the trade-offs? E.g., power vs performance
 - Addressing CPU-memory gap, power wall, etc.

Reminders

- Sign up for Piazza
- Activate your Instructional Lab account