CS 211: Computer Architecture

Lecture 1

Sept. 4, 2019

Logistics

Staff

Instructor: Jeff Ames (jca105@cs.rutgers.edu)

TAs:

- Priya Parikh (pp649@scarletmail.rutgers.edu)
- Wenjia Zhang (wz225@rutgers.edu)
- Haozhe Su (hz.su@rutgers.edu)
- Bo Zhang (bz186@rutgers.edu)

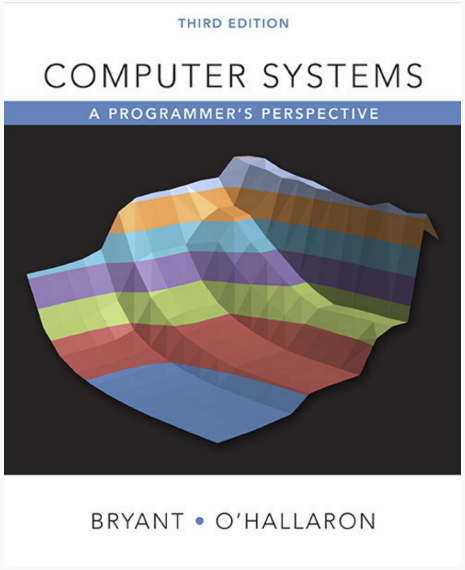
Office hours TBA

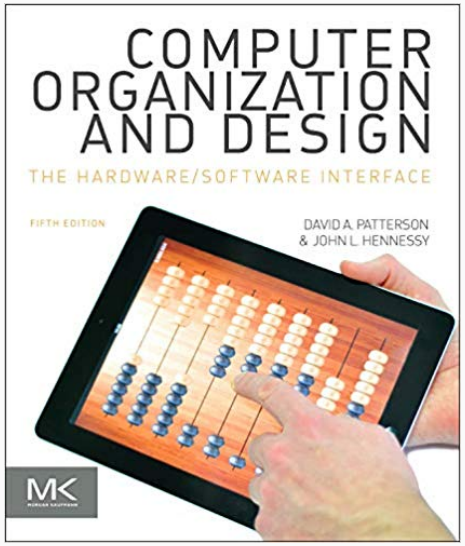
Web sites

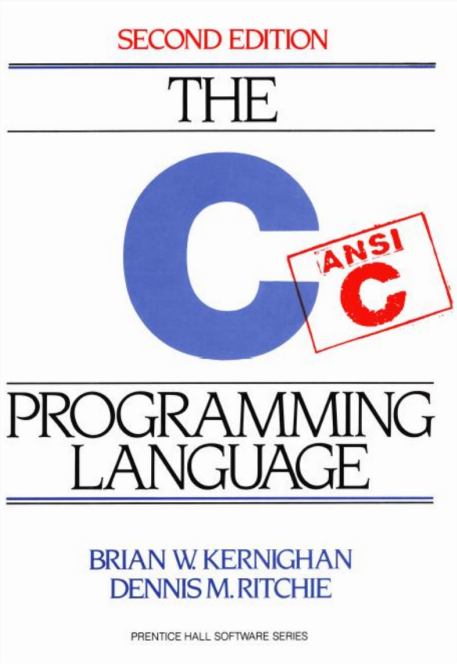
Canvas will be the primary source of course info.

We'll also use Piazza for questions.

Textbooks







Another option for C programming:

http://icube-icps.unistra.fr/img_auth.php/d/db/ModernC.pdf

Prerequisites

- You know some math
- You know a bit about algorithms and data structures
- You know at least one programming language (Java)
- You know something about how to write, run, and test programs

What you'll learn

- How to program in two more programming languages (C and Assembly)
- The major hardware components in computer systems
- Trends in technology and computer architecture
- How hardware components are built from digital logic
- How programs written in a high-level language (e.g., C) is actually executed by the hardware
- How to understand and improve the performance of programs

Expectations

- 5 programming assignments
- 2 midterm exams
- 1 final exam
- All exams are comprehensive

How to succeed in this class

- Attend lectures and recitations
- Actively participate in class
- Read the assigned readings before lecture
- Start the programming assignments early
- Read and think about the programming and homework assignments
- Ask questions
- Don't copy or cheat

Late assignments

- Late assignments will not be accepted
- Programming assignments to be handed in on Canvas
- Deadline will be enforced by Canvas
- Can hand in assignments multiple times so if you are working at the last minute, hand in a version early

Collaboration

Collaboration is encouraged. You learn by discussing with others. But must not cheat...

Follow the CS Department's academic integrity policy.

If you are having trouble with the course for any reason, come talk to us.

Tentative list of topics

- Introduction
- Hardware trends
- C programming
- Information representation
- Assembly (x86) programming
- Digital logic
- Processor architecture
- Memory hierarchy

Grading

Grading:

- 20%: programming assignments
- 25%: midterm 1
- 25%: midterm 2
- 30%: final exam
- All exams are cumulative

No make-up exams except for university sanctioned reasons.
You must get approval from the professor before the exam.

Programming assignments

- 5 programming assignments
- Program in C and/or Assembly
  - Don't wait until the last minute
  - Learn how to use tools
  - Don't program/debug "by accident" or "by blind search"
- Will be done using the Instructional Lab
  - https://resources.cs.rutgers.edu
  - If you don't already have an account, get one asap
  - You will be programming in a Linux environment

Assignments

If you don't already have an account on the Instructional Laboratory cluster create one at:

https://resources.cs.rutgers.edu/docs/new-users/getting-started/

In addition, there are a number of references to help you understand how to use the iLab systems. Please look at:

https://resources.cs.rutgers.edu/docs/new-users/beginners-info/

Other useful resources

Some other manuals that will be useful:

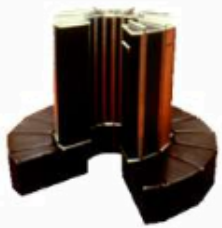
- (GCC) https://gcc.gnu.org/onlinedocs/
- (Emacs) https://www.gnu.org/software/emacs/manual
- (Make) https://www.gnu.org/software/make/manual/
- (Gdb) https://www.gnu.org/software/gdb/documentation

Why architecture?

- Security
- Performance

Computers

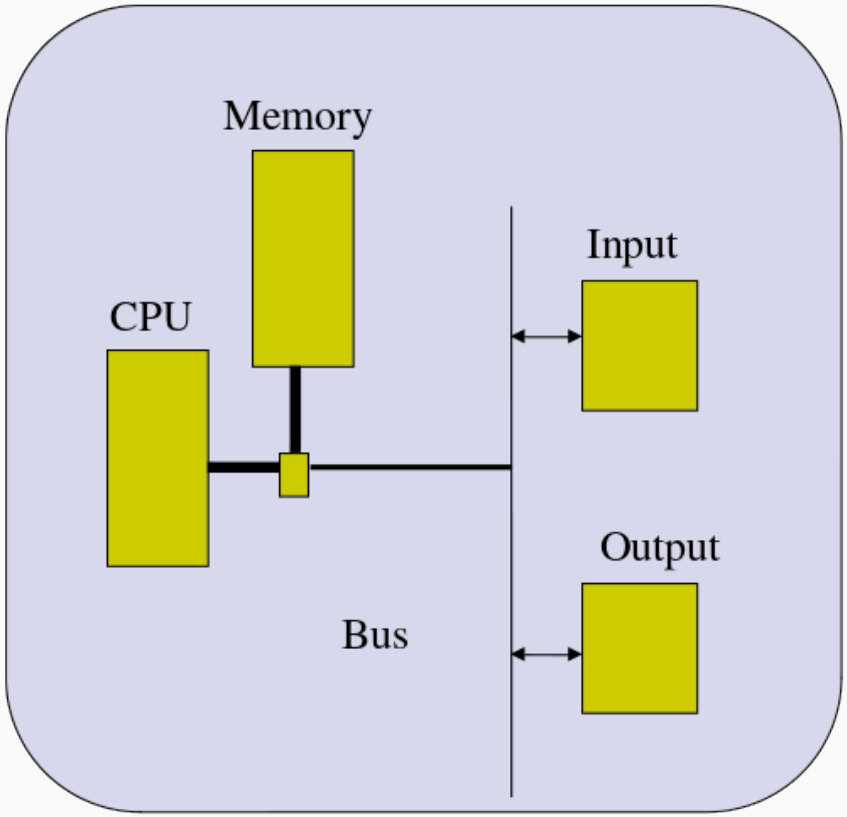
Architecture



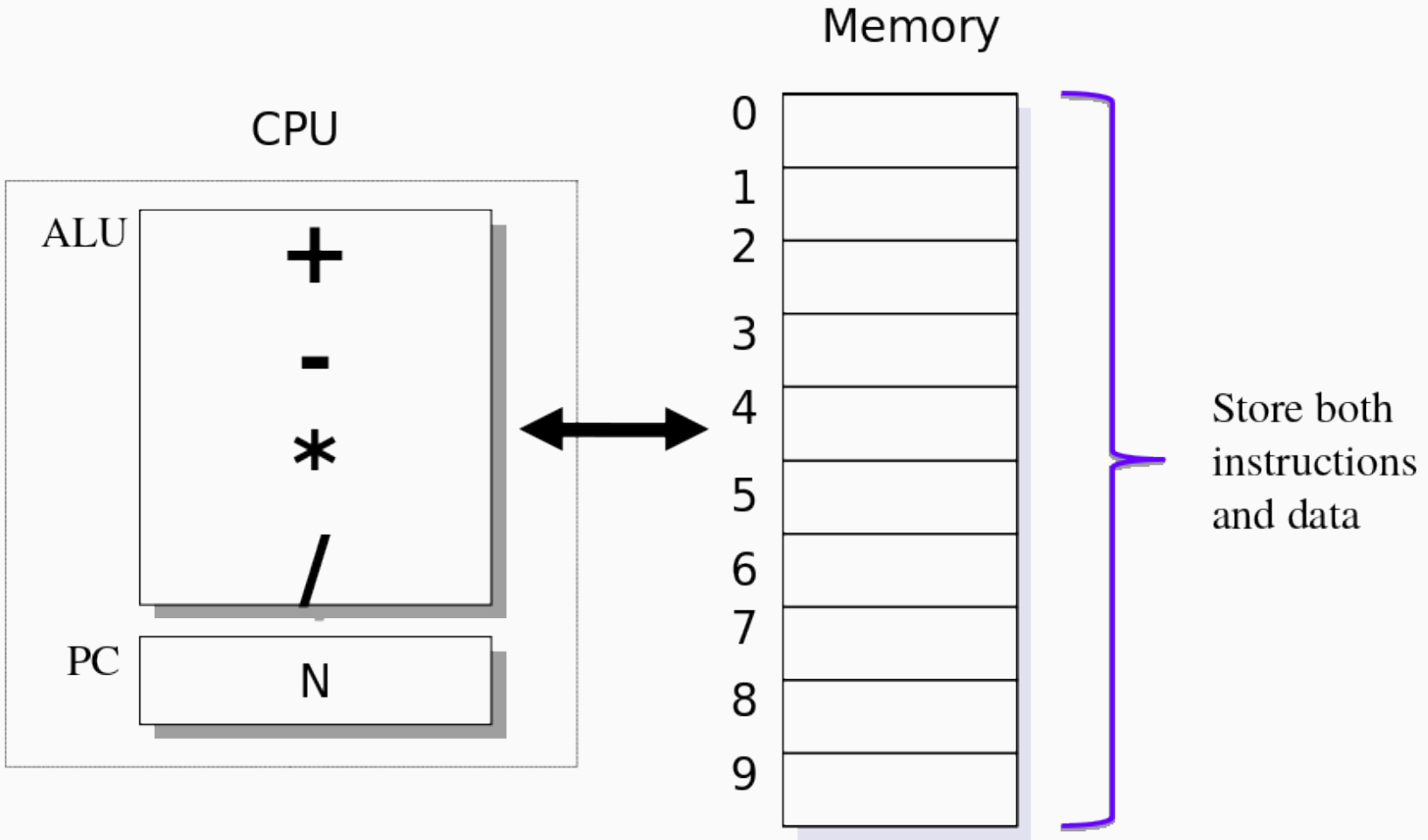
?

Main components

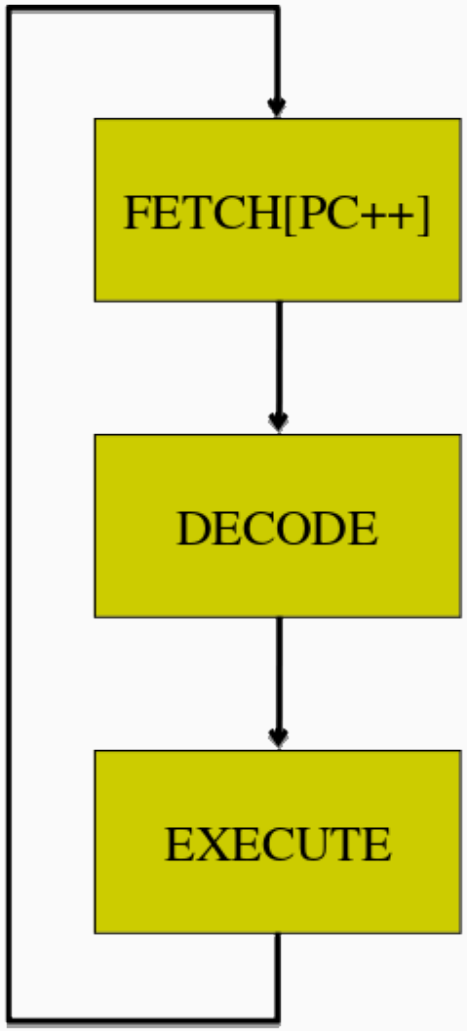
- CPU
- Memory
- Bus
- I/O devices
  - Mouse, keyboard, screen
  - Storage
  - Network
  - Graphics



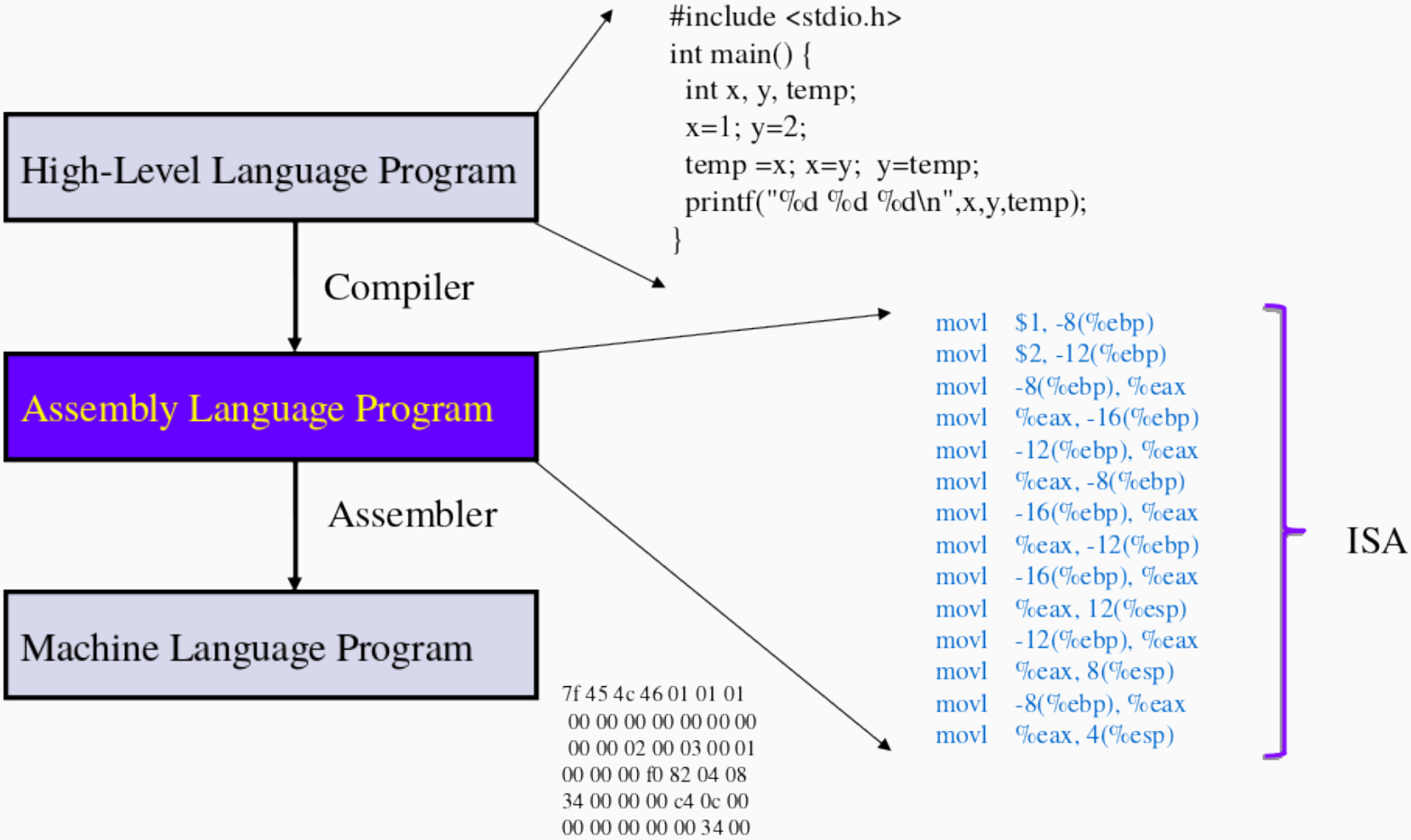
Von Neumann model



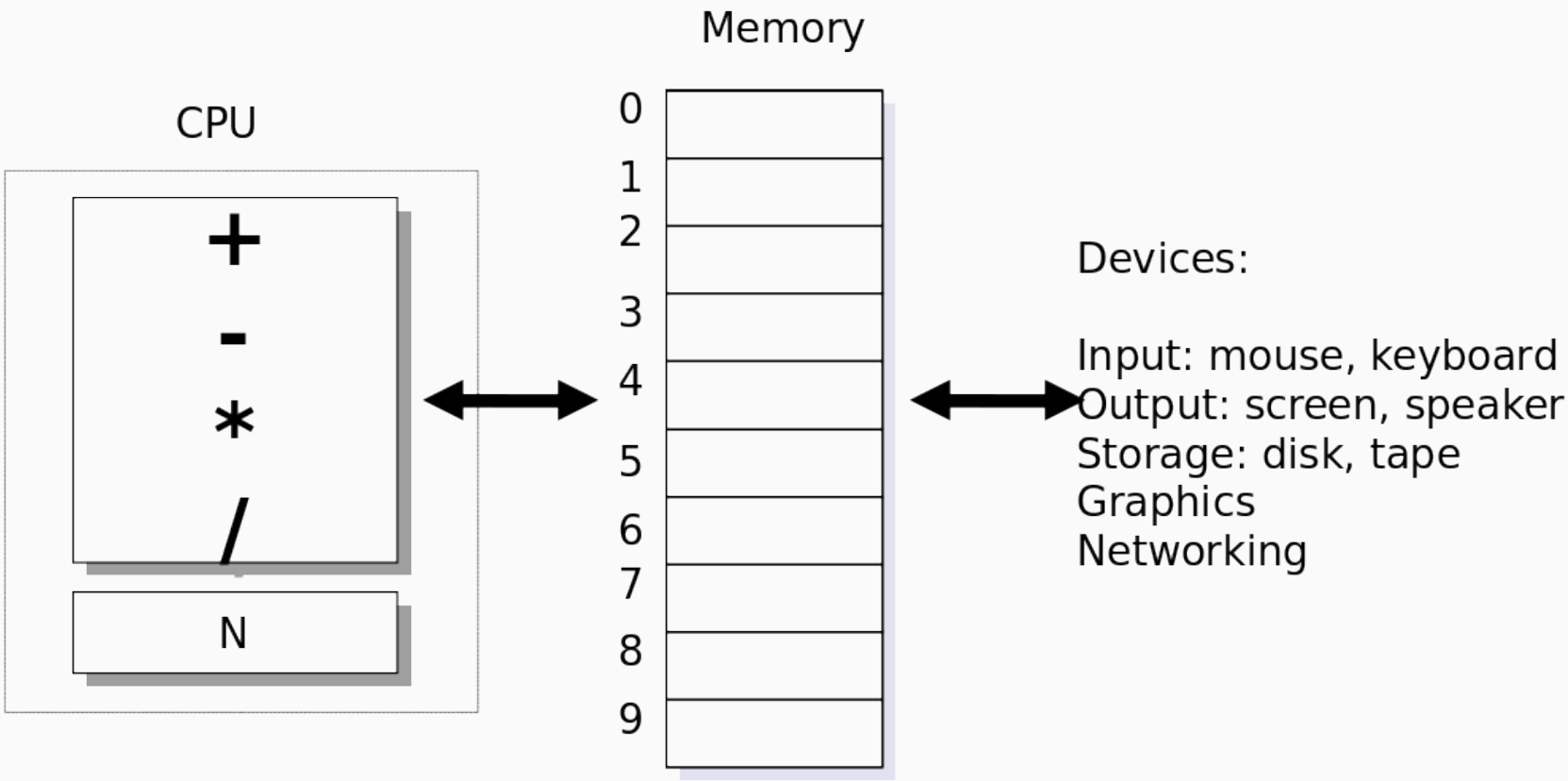
CPU function



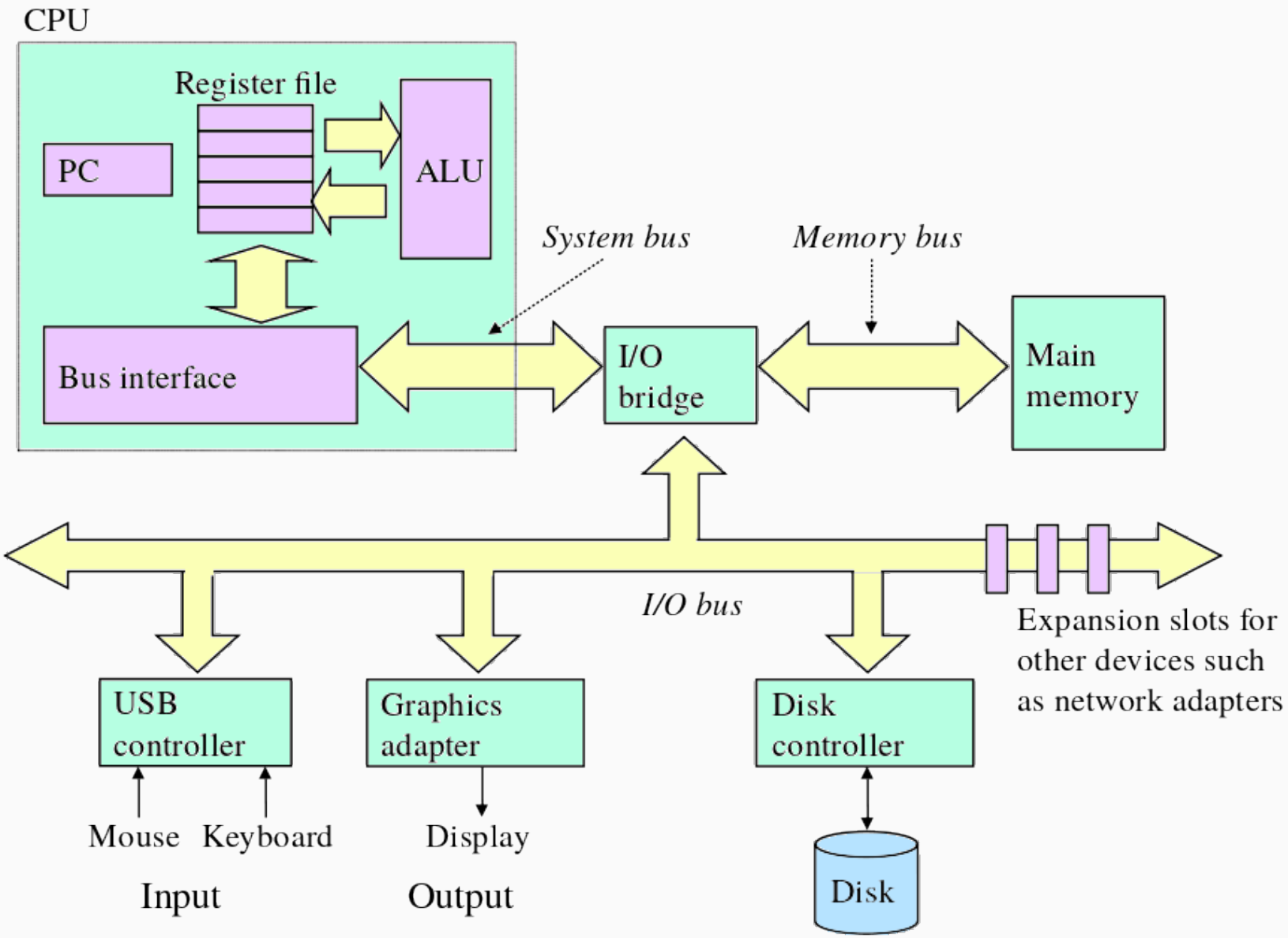
Compiler, assembler



I/O


Memory
CPU
+
-
*
/
N
0
1
2
3
4
5
6
7
8
9
Devices:
Input: mouse, keyboard
Output: screen, speaker
Storage: disk, tape
Graphics
Networking

Von Neumann details



Architecture

- How to design and build the components
- How to design and build systems from the components
- In this class:
  - Understand basics of current components and systems
  - Understand how programs run on current systems
  - Understand how current architecture affect my high-level language programs
  - How can I make my program run faster?
  - How can I make my program more power efficient?
  - Run longer when on battery

Moore's law

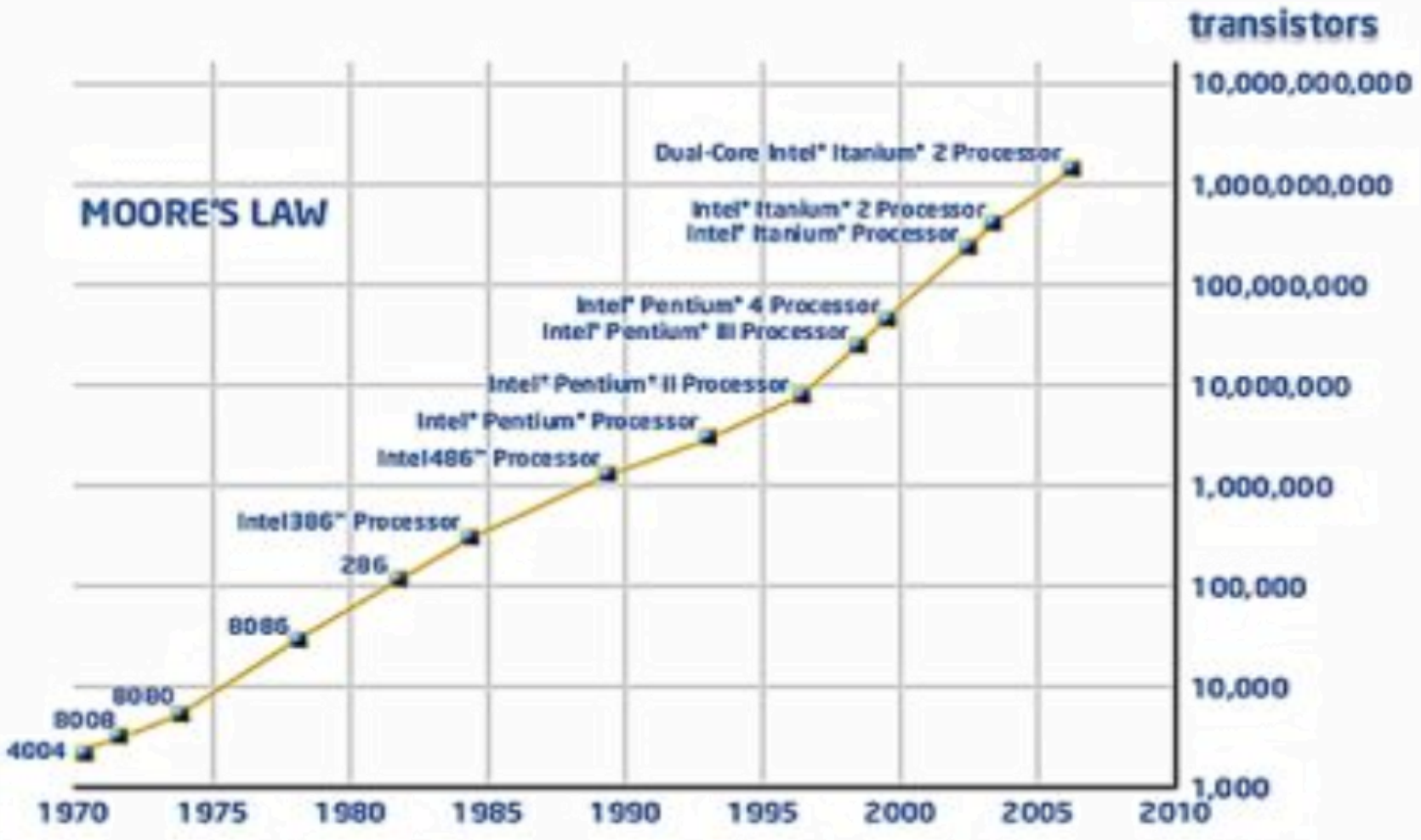
- Gordon Moore was an Intel Engineer
- An observation about improvements in hardware
- No of transistors on a chip double every 18 months
- Exponential growth seen in other hardware dimensions
  - Processor speed: 2x every 18 months
  - Memory capacity: 2x every 2 years
  - Disk capacity: 2x every year

Transistors

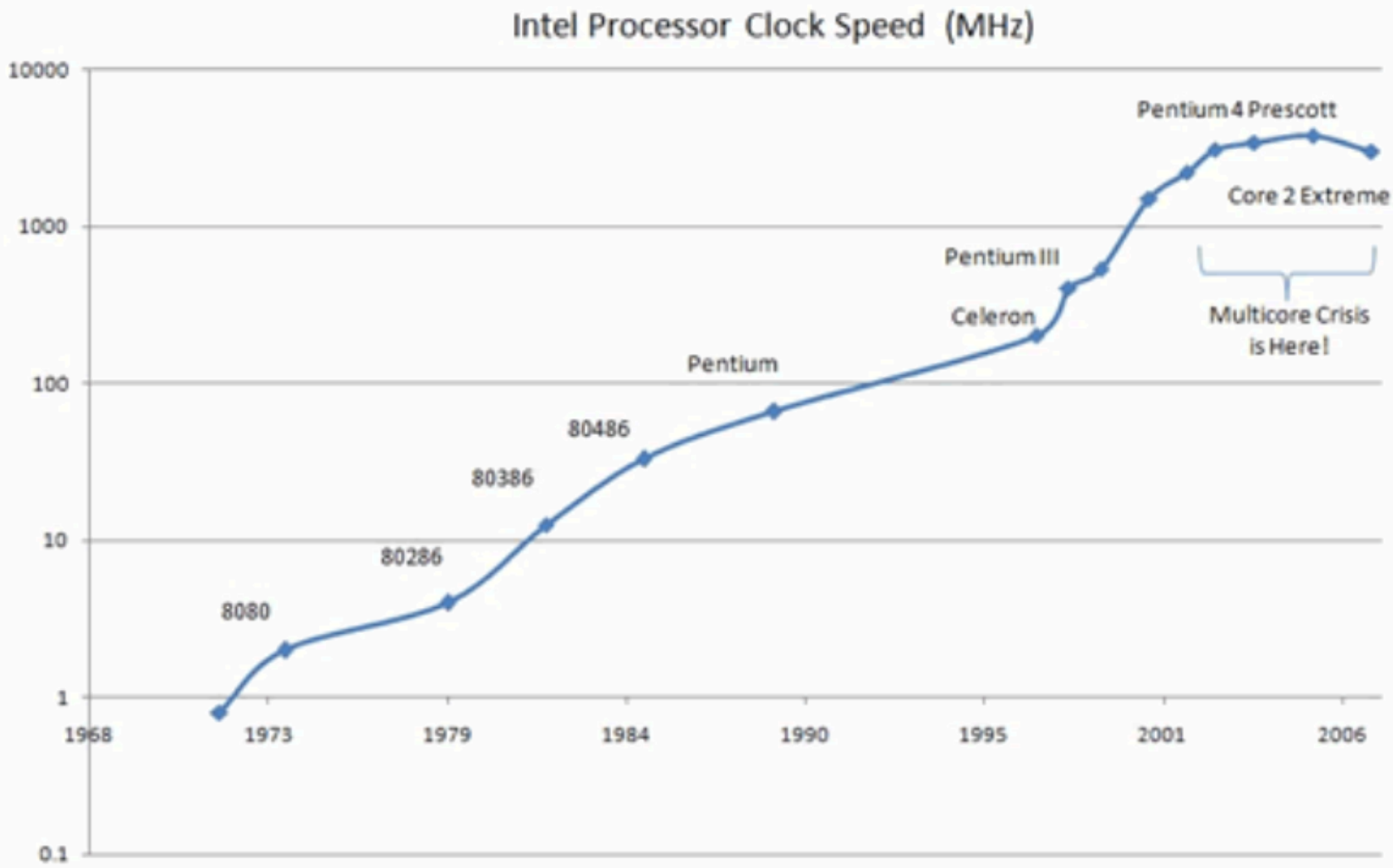
How many transistors in the latest Intel chip in your laptops?

Intel Broadwell Xeon chip has 7.2 billion transistors!

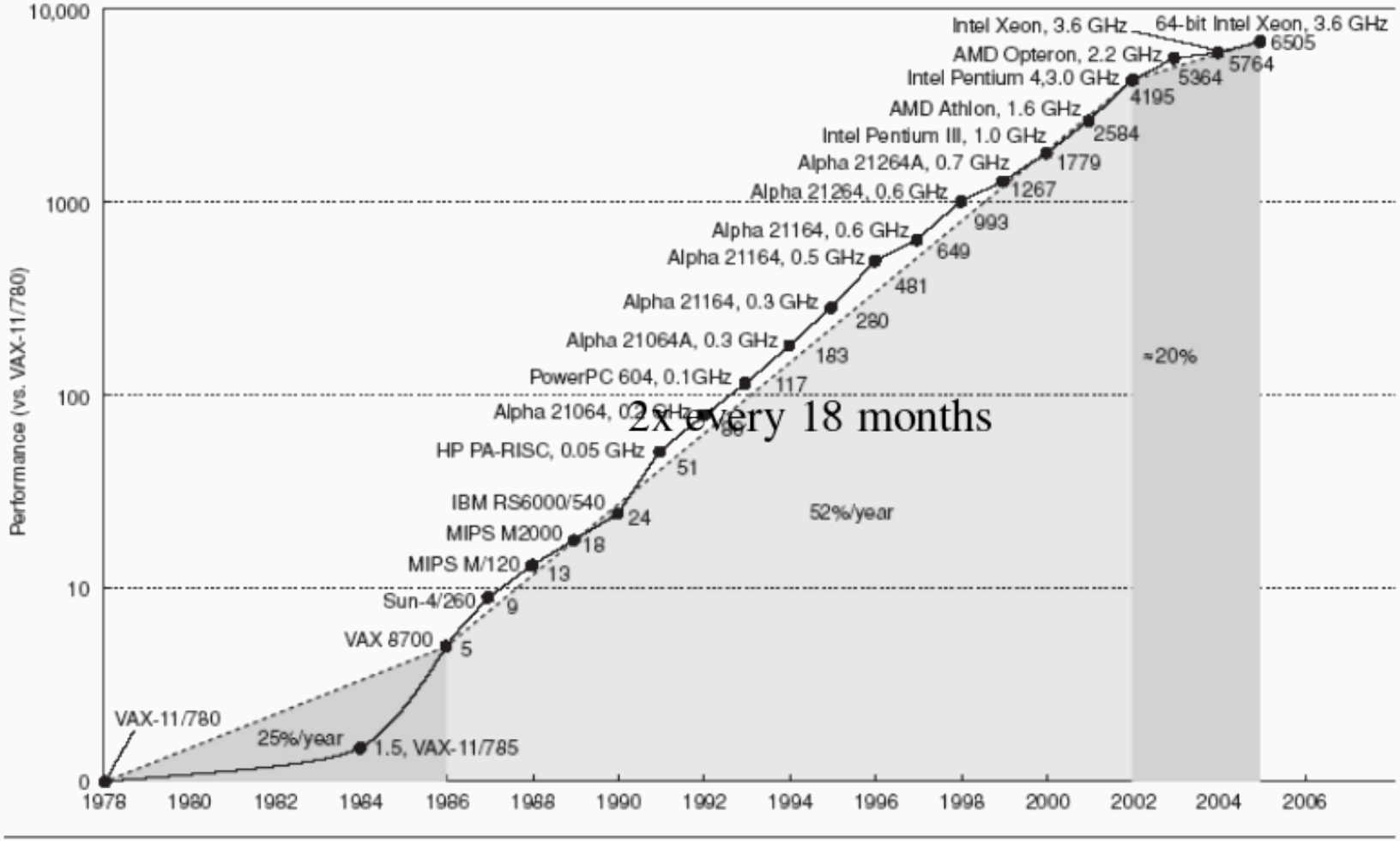
Transistors



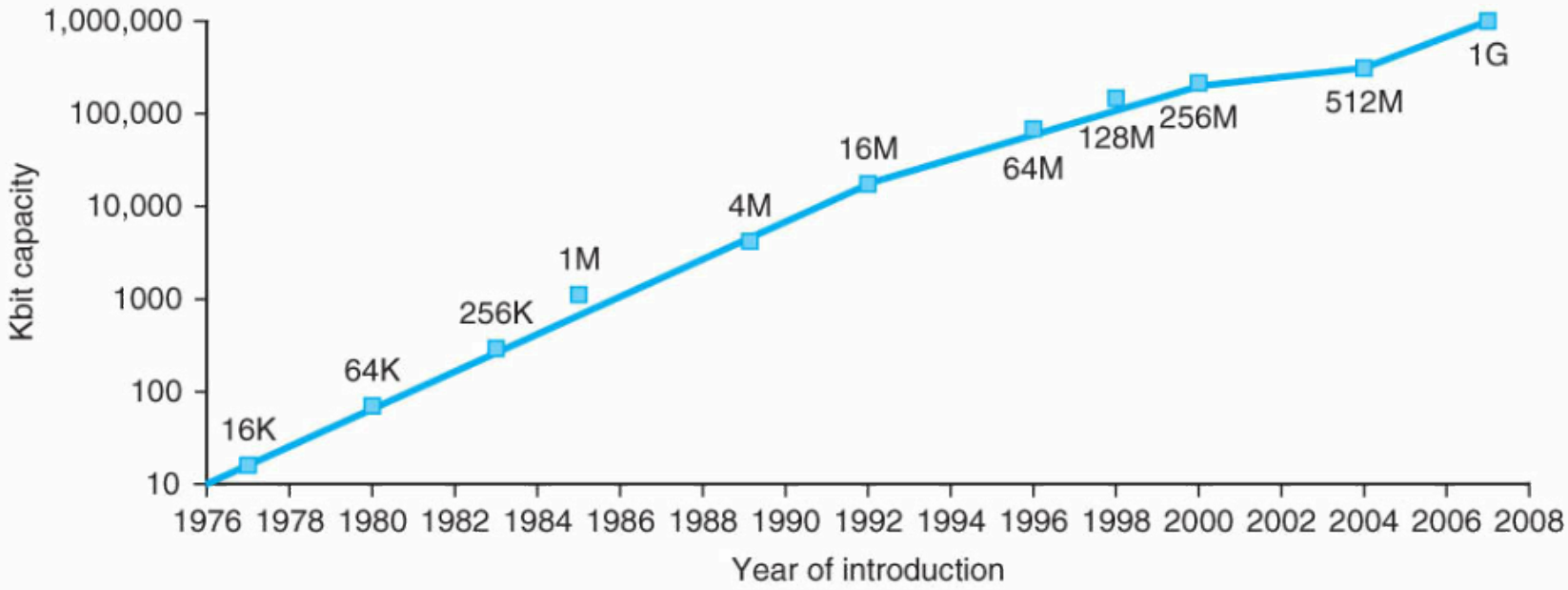
Clock speed


Intel Processor Clock Speed (MHz)
10000
1000
100
10
1
0.1
1968
1973
1979
1984
1990
1995
2001
2006
8080
80286
80386
80486
Pentium
Celeron
Pentium III
Pentium 4 Prescott
Core 2 Extreme
Multicore Crisis is Here!

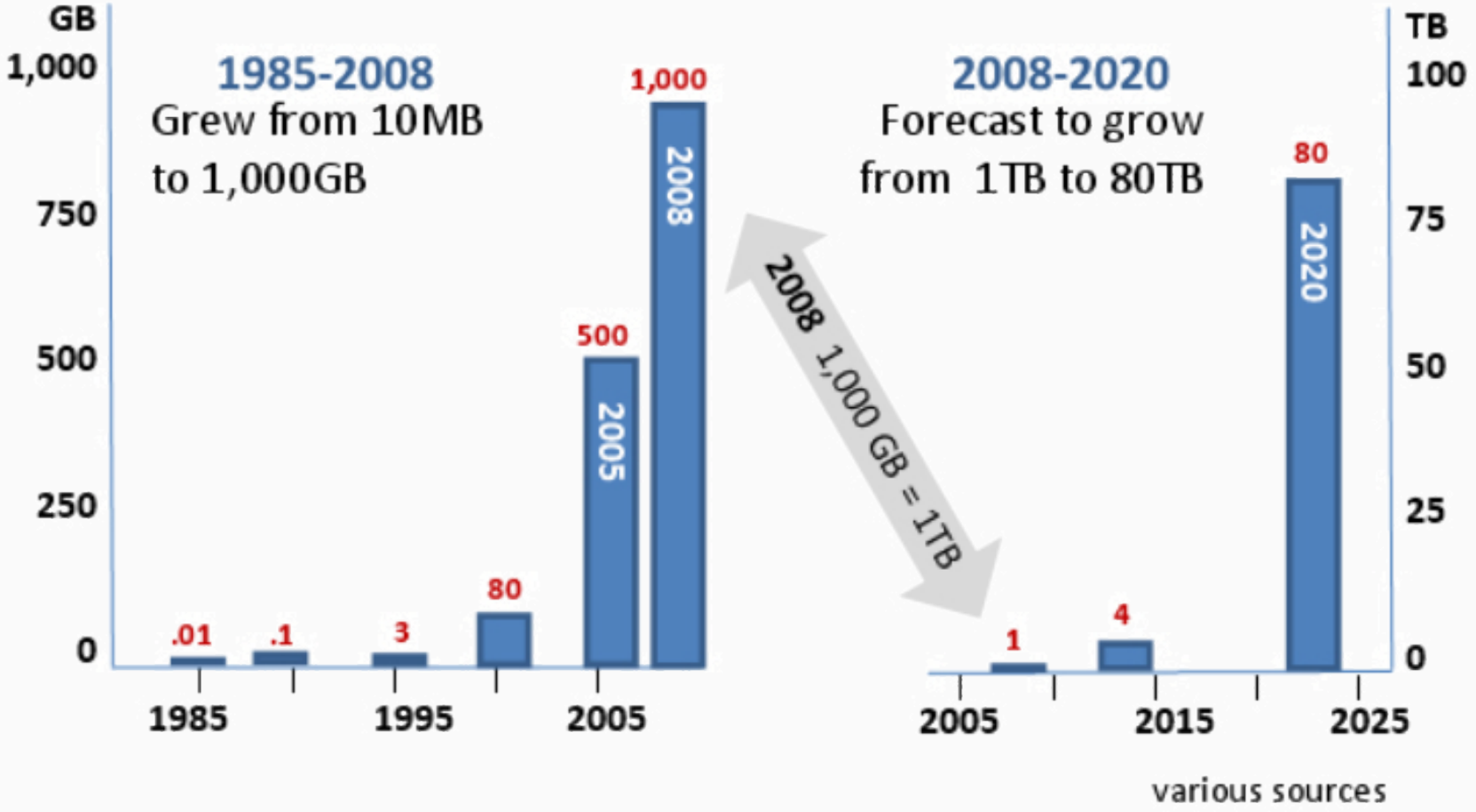
Performance



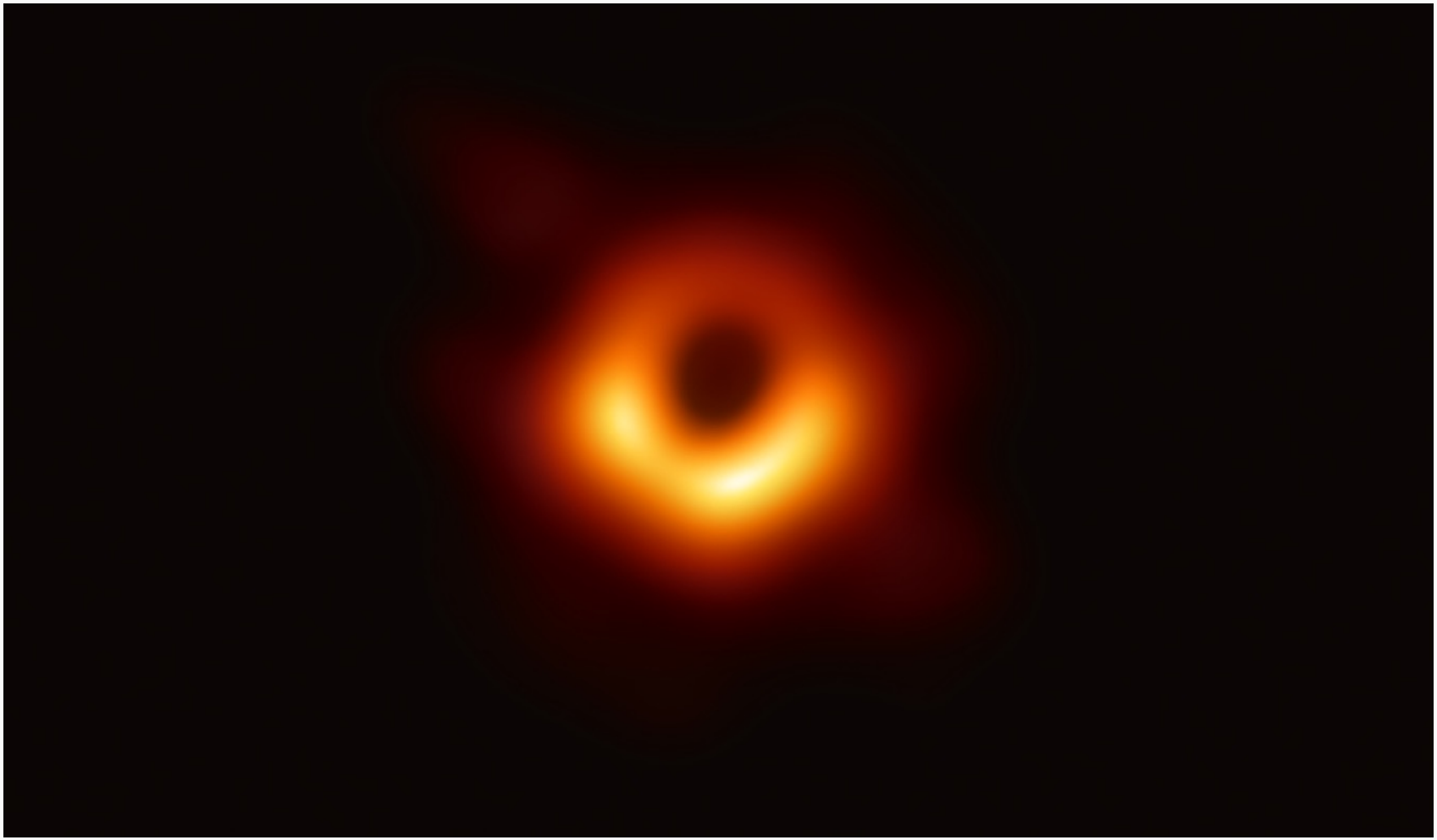
Memory capacity



Disk capacity

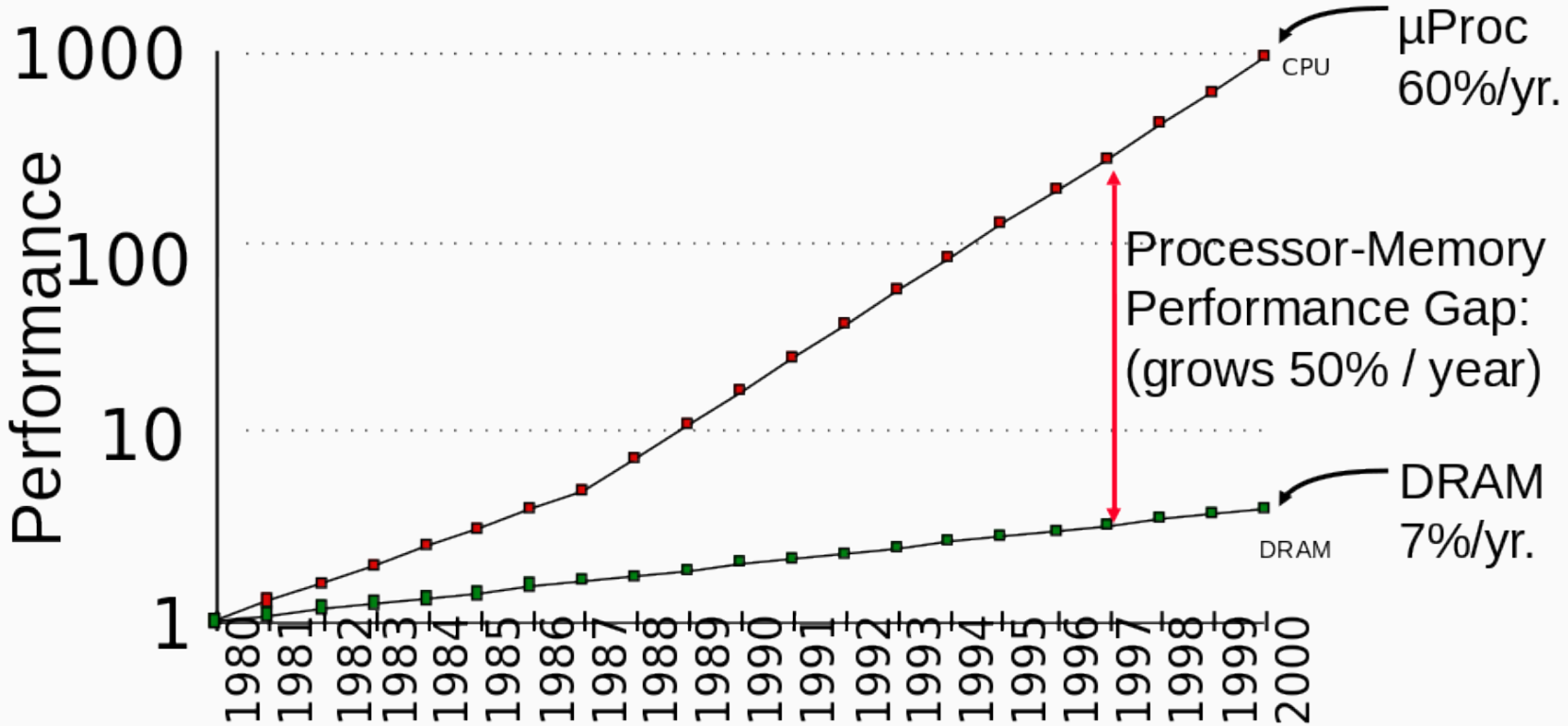


A black hole

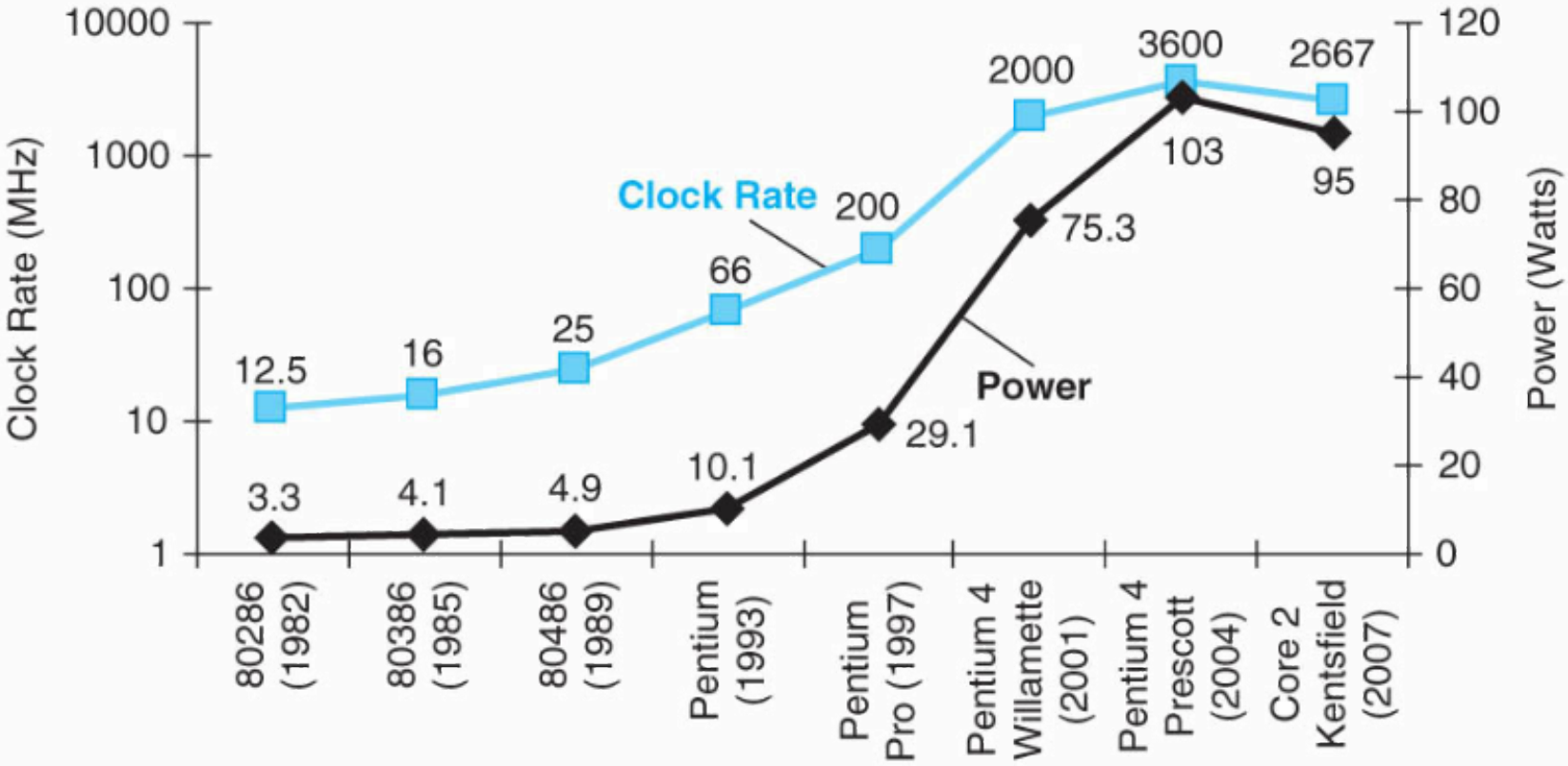


5 PB to image a black hole

CPU/memory performance gap



Power wall


Clock Rate (MHz)
10000
1000
100
10
1
Power (Watts)
120
100
80
60
40
20
0
Clock Rate
Power
12.5
16
25
66
200
2000
3600
2667
3.3
4.1
4.9
10.1
29.1
75.3
103
95
80286 (1982)
80386 (1985)
80486 (1989)
Pentium (1993)
Pentium Pro (1997)
Pentium 4 Willamette (2001)
Pentium 4 Prescott (2004)
Core 2 Kentsfield (2007)

- Today's systems are complex with various components
- Faster processors and systems
  - Enables new domains of applications
  - Imagine the running your favorite game on a Pentium machine.
- Understanding the systems crucial
  - To program these systems
  - To make pragmatic performance/power/energy tradeoffs
  - Address various walls — memory wall, power wall, etc

C programming

Hello world

```
1 public class MyApp
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello, world!");
6     }
7 }
```

Hello world

- Save file as MyApp.java
- Compile with “javac MyApp.java”
- Run as “java MyApp”

Hello world in C

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("Hello, world!\n");
}
```

Hello world in C

- Save file as something.c
- Compile with “gcc -o something something.c”
- Run as “./something”

C vs. Java

- Older (1972 vs. 1995)
- Compiles to machine code
- Faster?
- No objects
- Pointers
- Manual memory management
- Separate struct/function declaration

C control flow

Many parts of C are very similar to Java:

- if, if-else, if-else if-else
- for
- while
- do
- switch

C data types

- int
- char
- float
- double
- bool (in a later revision of C called C99)

C input/output

- printf
- scanf

C input/output

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("What's your name?\n");
    char name[100];
    scanf("%s", name);
    printf("Welcome to CS 211, %s!\n", name);
}
```