

CS3210: Operating Systems

Taeso Kim

What is an *operating system*?

- e.g. OSX, Windows, Linux, FreeBSD, etc.
- What does an OS do for you?
 - **Abstract** the hardware for convenience and portability
 - **Multiplex** the hardware among multiple applications
 - **Isolate** applications to contain bugs
 - Allow **sharing** among applications

View: layered organization

- **User:** applications (e.g., vi and gcc)
- **Kernel:** file system, process, etc.
- **Hardware:** CPU, mem, disk, etc.

→ Interface b/w layers

View: core services

- Processes
 - Memory
 - File contents
 - Directories and file names
 - Security
 - Many others: users, IPC, network, time, terminals, etc.
- Abstraction for applications

Example: system calls

- **Interface** : applications talk to an OS via system calls
- **Abstraction** : process and file descriptor

```
fd = open("out", 1);  
write(fd, "hello\n", 6);  
pid = fork();
```

Why is designing OS interesting?

- Conflicting design goals and trade-offs
 - Efficient yet portable
 - Powerful yet simple
 - Isolated yet interactable
 - General yet performant
- Open problems: security and multi-core

Who should take CS3210?

- Anyone wants to work on the above problems
- Anyone cares about what's going on under the hood
- Anyone has to build high-performance systems
- Anyone needs to diagnose bugs or security problems

Also, you have *lots of free time ...*

About this course

- CS3210: "Design" Operating Systems
- Goals
 - Understand operating systems in detail by designing and implementing a small O/S
 - Hands-on experience with building systems

Prerequisite

- C programming (strict)
- CS 2200: Systems and Networks (strict)
- CS 2110: Computer Organization and Programming (recommended)
- CS 3220: Processor Design (recommended)



I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. -- Linus Torvalds

General information

- Web: <https://tc.gtisc.gatech.edu/cs3210>
- Piazza: <https://piazza.com/gatech/spring2016/cs3210agr>
- Text: freely available online
 - xv6: a simple, Unix-like teaching operating system
 - (optional) Linux Kernel Development

General information

- TA: send us an email for an appointment (3210-staff@cc.gatech.edu)
 - Sudarsun Kannan (head TA)
 - Sanidhya Kashyap
 - Kyuhong Park

Grading policy

- Preparation (10%)
- Quiz (20%)
- Lab (40% + 10% bonus)
- Final project (30%)
 - Proposal presentation (5%)
 - Demo & presentation (15%)
 - Write-up (10%)

Class structure

- Tue: Lecture, in-class exercises
- Thr: Tutorial
 - Individual exercises
 - Group meeting

Bring your laptop!

Class structure

Monday	Tuesday	Wednesday	Thursday	Friday
Jan 11 First day of class (No class)	Jan 12 LEC 1: Operating systems (xv6, code and book) Assignment: Lab 1: Booting a PC	Jan 13	Jan 14 TUT 1: Tools Preparation: Read git (Question)	Jan 15 <i>ADD DATE</i>
Jan 18 M.L.K, Jr. National Holiday	Jan 19 LEC 2: Booting and x86 Preparation: Read Appendix A/B (Question)	Jan 20	Jan 21 TUT 2: C and gdb (pointers example) Preparation: Read 2.9 (Bitwise operators), 5.1 (Pointers and Addresses) through 5.5 (Character Pointers and Functions) and 6.4 (pointers to structures) in K&R Assignment: Lab 2: Memory management	Jan 22

- First week:
 - Lecture: about PC, Booting, and C
 - Tutorial: tools
- NOTE: preparation questions

About quiz, project

- Two quiz (in-class, about lec/tut/lab)
- Final project
 - <https://tc.gtisc.gatech.edu/cs3210/2016/proj.html>
 - Pre-proposal, Team proposal, and Demo day

About preparation questions

- Every lecture and tutorial (DUE: by 10pm the day before)

Question for Lecture 3 (lec3.txt)



What kind of UNIX's features do you see on today's operating systems? and what's not? Do you see any missing features that are prevalent in modern, commodity operating systems?

About labs

- A toy operating system, called JOS (exokernel)
 - Lab 1: Booting a PC
 - Lab 2: Memory management
 - Lab 3: User environments
 - Lab 4: Preemptive multitasking
 - Lab 5: File system and shell
 - Lab 6: Network driver

About labs

- "Lab 1: Booting a PC" is out (DUE: Jan 19th)
- Ask questions via inline comments

Lab 1: Booting a PC

- Handed out: Tuesday, January 12, 2016
- Due: Tuesday, January 19, 2016

Introduction

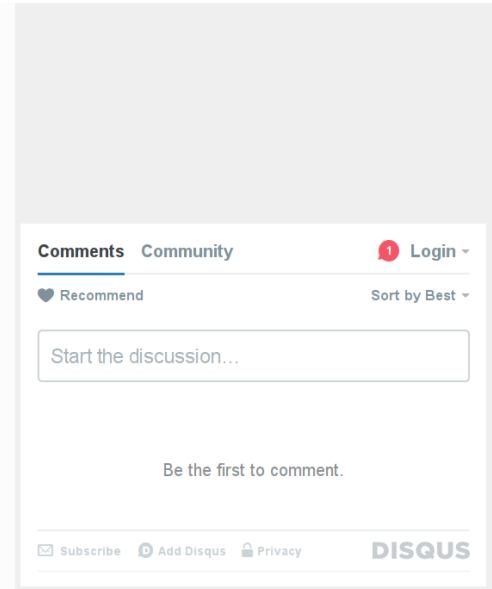
This lab is split into three parts. The first part concentrates on getting familiarized with x86 assembly language, the QEMU x86 emulator, and the PC's power-on bootstrap procedure. The second part examines the boot loader for our cs3210 kernel, which resides in the `boot/` directory of the `lab/` tree. Finally, the third part delves into the initial template for our cs3210 kernel itself, named JOS, which resides in the `kernel/` directory.

Software Setup

The files you will need for this and subsequent lab assignments in this course are distributed using the [Git](#) version control system. To learn more about Git, take a look at the [Git user's manual](#), or, if you are already familiar with other version control systems, you may find this [CS-oriented overview of Git](#) useful.

The URL for the course Git repository is `git://tc.gtisc.gatech.edu/cs3210-lab`

...



The screenshot shows a Disqus comment interface. At the top, there are tabs for "Comments" and "Community", with "Comments" being the active tab. To the right of the tabs is a "Login" button with a notification icon. Below the tabs, there is a "Recommend" button with a heart icon and a "Sort by Best" dropdown menu. A large text input field contains the placeholder text "Start the discussion...". Below the input field, the text "Be the first to comment." is displayed. At the bottom of the interface, there are links for "Subscribe" (with an envelope icon), "Add Disqus" (with a plus icon), and "Privacy" (with a lock icon), followed by the "DISQUS" logo.

Class policy

- Late day
 - Four days of grace period (entire labs)
 - Don't have to inform us (e.g., job interview, sick)
- No cheating
 - Cheating vs. collaboration
 - Write the name(s) of your sources

See, <https://tc.gtisc.gatech.edu/cs3210/2016/info.html>

Today's agenda

- Hardware trends
- What is an operating system?
 - Design
 - Goal
 - Role
 - Example: xv6 and JOS
- In-class exercise: *C quiz!* & submission site

Micro-processor trends: CPU

- by David A. Patterson and John L. Hennessy (2013)

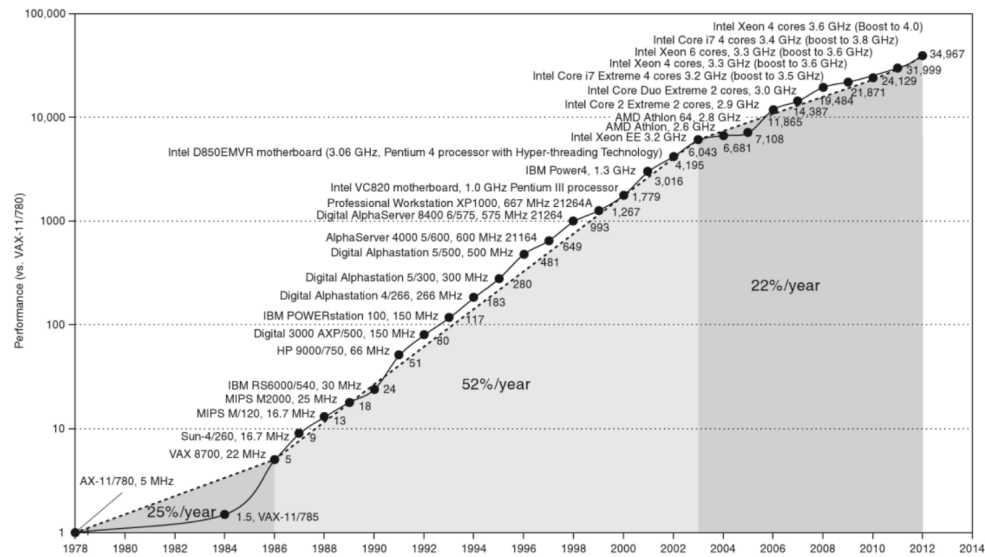
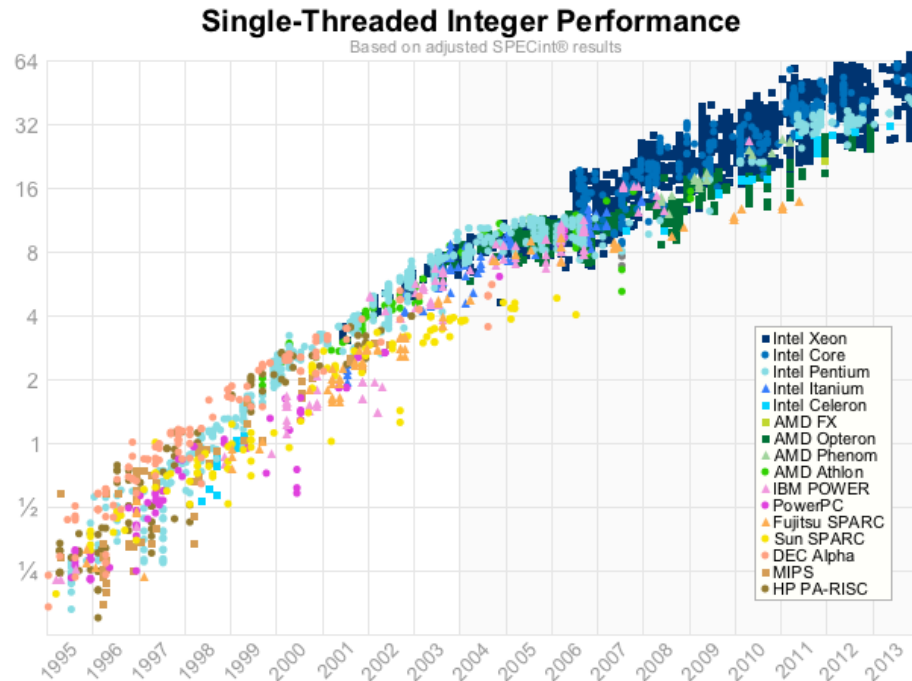


FIGURE 1.17 Growth in processor performance since the mid-1980s. This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks (see Section 1.10). Prior to the mid-1980s, processor performance growth was largely technology-driven and averaged about 25% per year. The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. The higher annual performance improvement of 52% since the mid-1980s meant performance was about a factor of seven higher in 2002 than it would have been had it stayed at 25%. Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 22% per year.

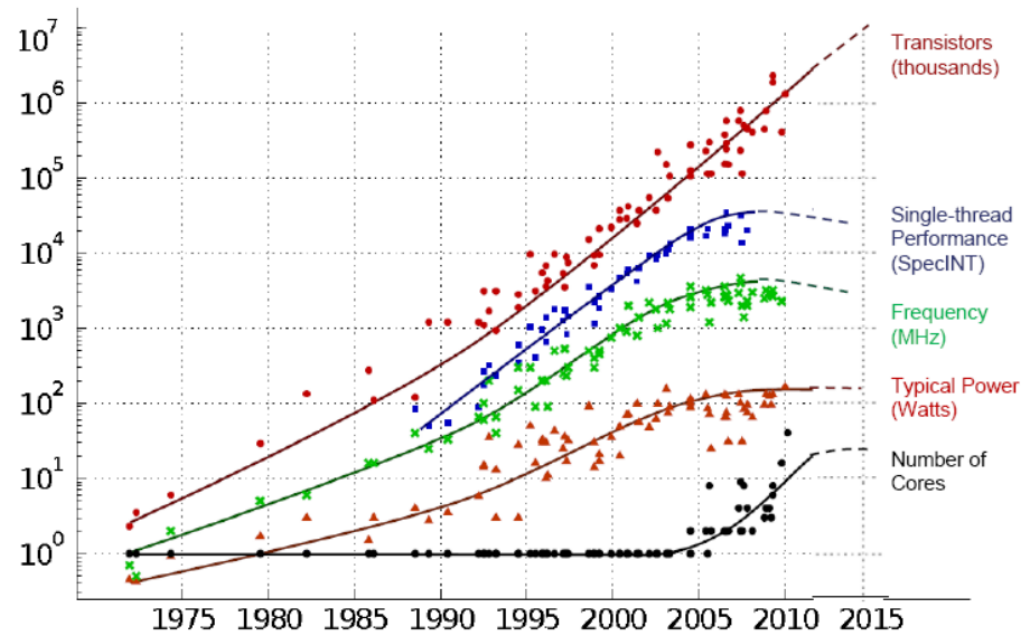
Micro-processor trends: SPECInt

- by Henk Poley (2014)



Projecting micro-processor Trends

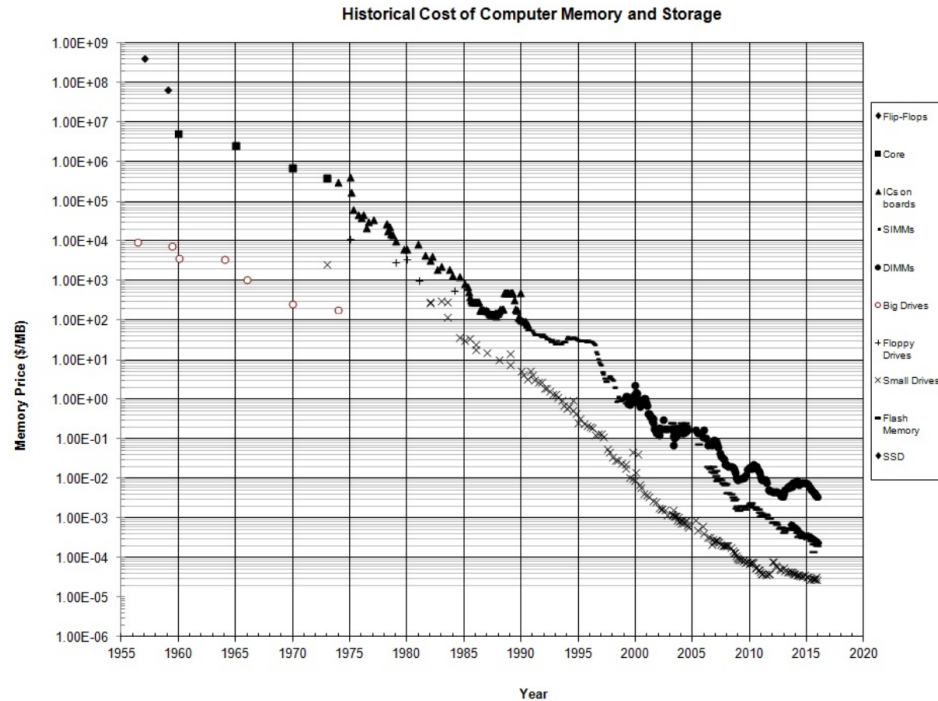
- by Chuck Moore (2012)



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

Memory trends

- by John C. McCallum (2015)



Computer performance trends

- by Thomas E. Anderson (2015)

	1981	1997	2014	Factor (2014/1981)
Uniprocessor speed (MIPS)	1	200	2500	2.5 K
CPUs per computer	1	1	10+	10+
Processor MIPS/\$	\$100K	\$25	\$0.20	500 K
DRAM Capacity (MiB)/\$	0.002	2	1K	500 K
Disk Capacity (GiB)/\$	0.003	7	25K	10 M
Home Internet	300 bps	256 Kbps	20 Mbps	100 K
Machine room network	10 Mbps (shared)	100 Mbps (switched)	10 Gbps (switched)	1000+
Ratio of users to computers	100:1	1:1	1:several	100+

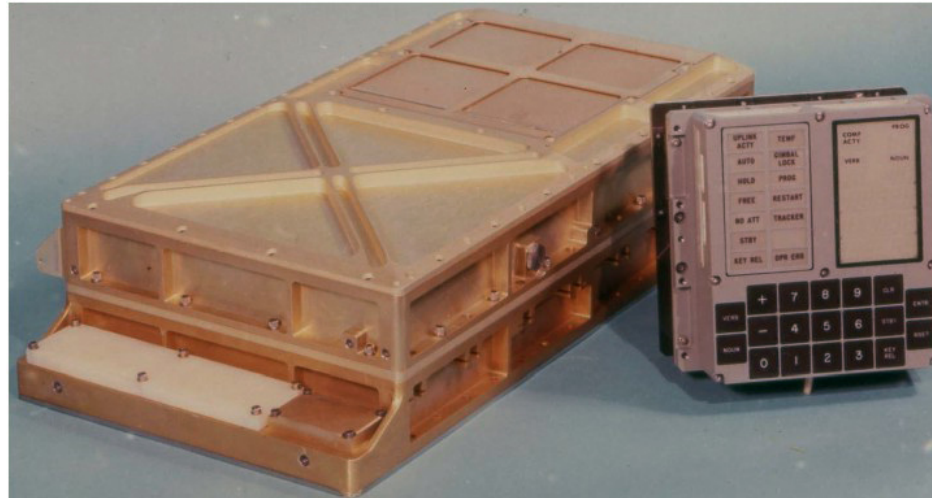
Figure 1.8: Approximate computer server performance over time, reflecting widely used servers of each era in 1981, a minicomputer; in 1997, a high-end workstation; in 2014, a rack-mounted multicore server. MIPS stands for “millions of instructions per second,” a rough measure of processor performance. The VAX 11/782 was introduced in 1982; it achieved 1 MIP. DRAM prices are from Hennessey and Patterson, *Computer Architecture: A Quantitative Approach*. Disk drive prices are from John McCallum. The Hayes smartmodem, introduced in 1981, ran at 300 bps. The 10 Mbps shared Ethernet standard was also introduced in 1981. One of the authors built his first operating system in 1982, used a VAX at his first job, and owned a Hayes to work from home.

Hardware: IBM System/360 30/75 (1964)

- Ref. [Computer History Museum](#)
 - CPU: 0.034/1 MIPS, 64K/8M (8-bit), \$133,000+/\$2.2m+
 - Tape Operating System (TOS), Basic Operating System (BOS/360)

Hardware: Apollo 11 (1969)

- Ref. [Wikipedia](#)
 - CPU: 1 MIPS, 64K (15-bit + 1-bit), \$???
 - Apollo Guidance Computer Software



Hardware: iPhone (2015)

- Ref. [Apple](#)
 - CPU: 25k MIPS (1.84 GHz, Dual-core), 2 GB (64-bit), \$649
 - iOS 9



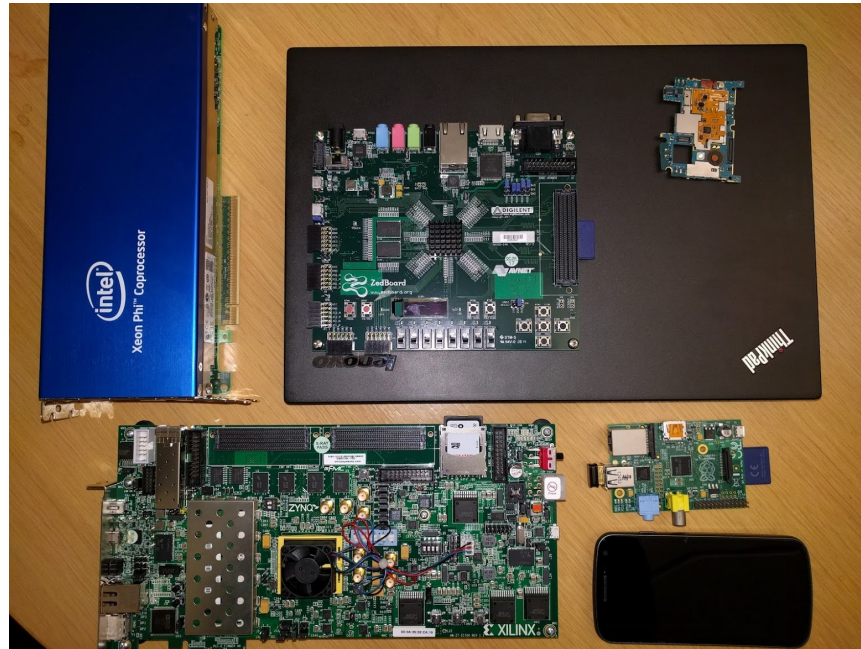
Hardware: Raspberry Pi Zero (2015)

- Ref. Raspberry.org
 - CPU: 1k MIPS (1 Gz, Single-core), 512 MB (32-bit), \$5
 - Linux



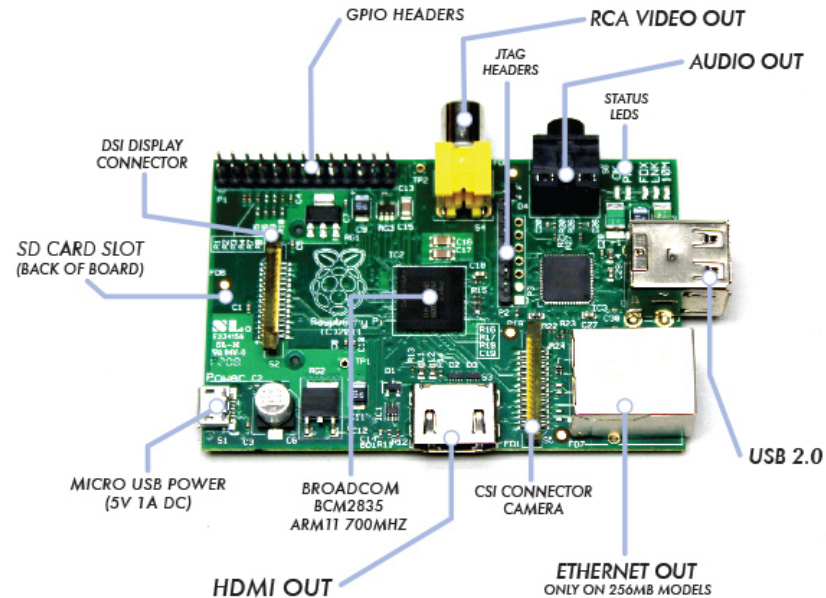
Modern hardware

- What's common among these devices?



Example: Raspberry Pi B

- Ref. [The Raspberry Pi Board](#)



Challenges in operating systems

- Portability
- Performance
- Reliability
- Security

Challenges in (practical) operating systems

e.g. Mac OSX, Windows, Linux

- Legacy (compatibility)
- Implementation
- Business

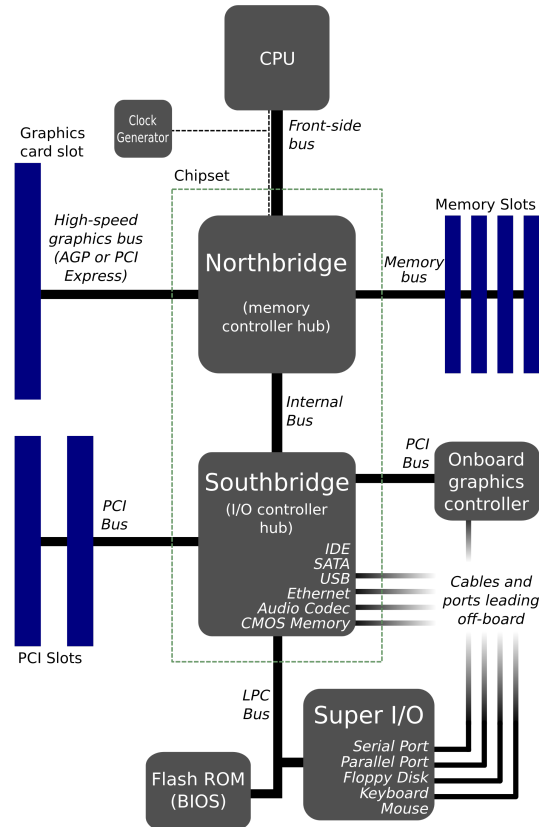
About "compatibility"

- <https://lkml.org/lkml/2012/3/8/495>

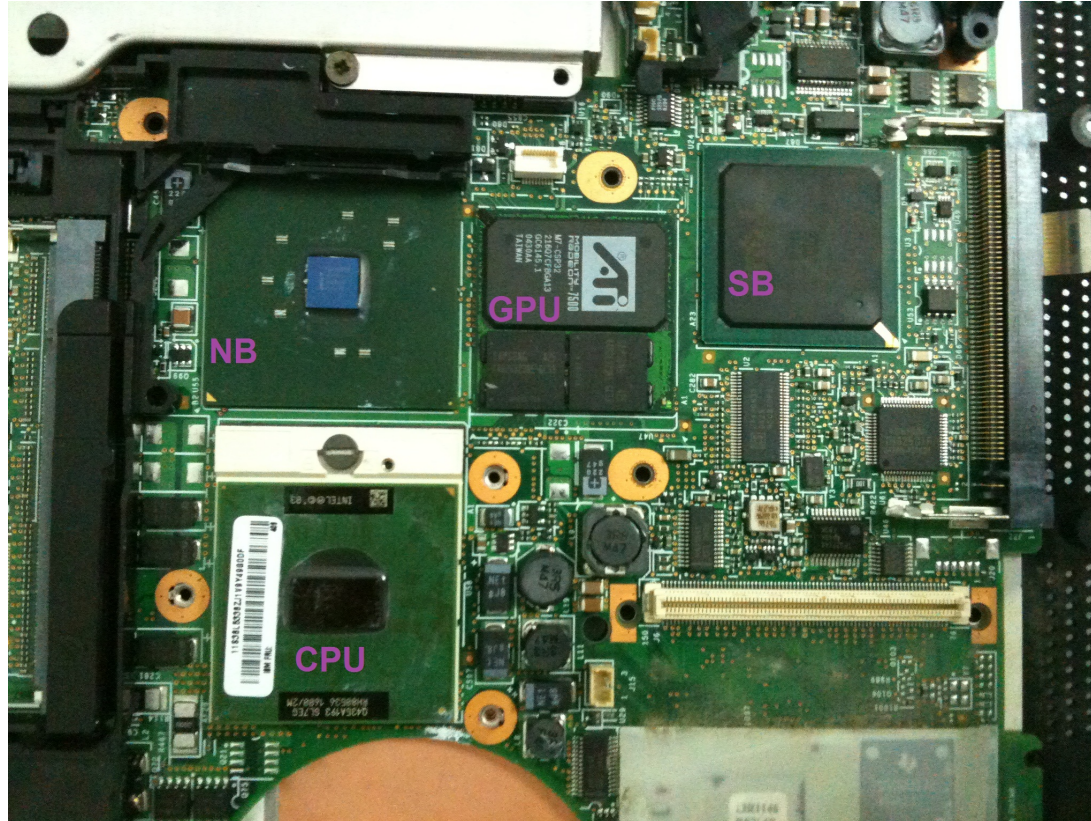


Seriously. Binary compatibility is so important that I do not want to have anything to do with kernel developers who don't understand that importance. If you continue to pooh-pooh the issue, you only show yourself to be unreliable. Don't do it. -- Linus

Conceptual hardware for CS3210



It's realistic (IBM T42)

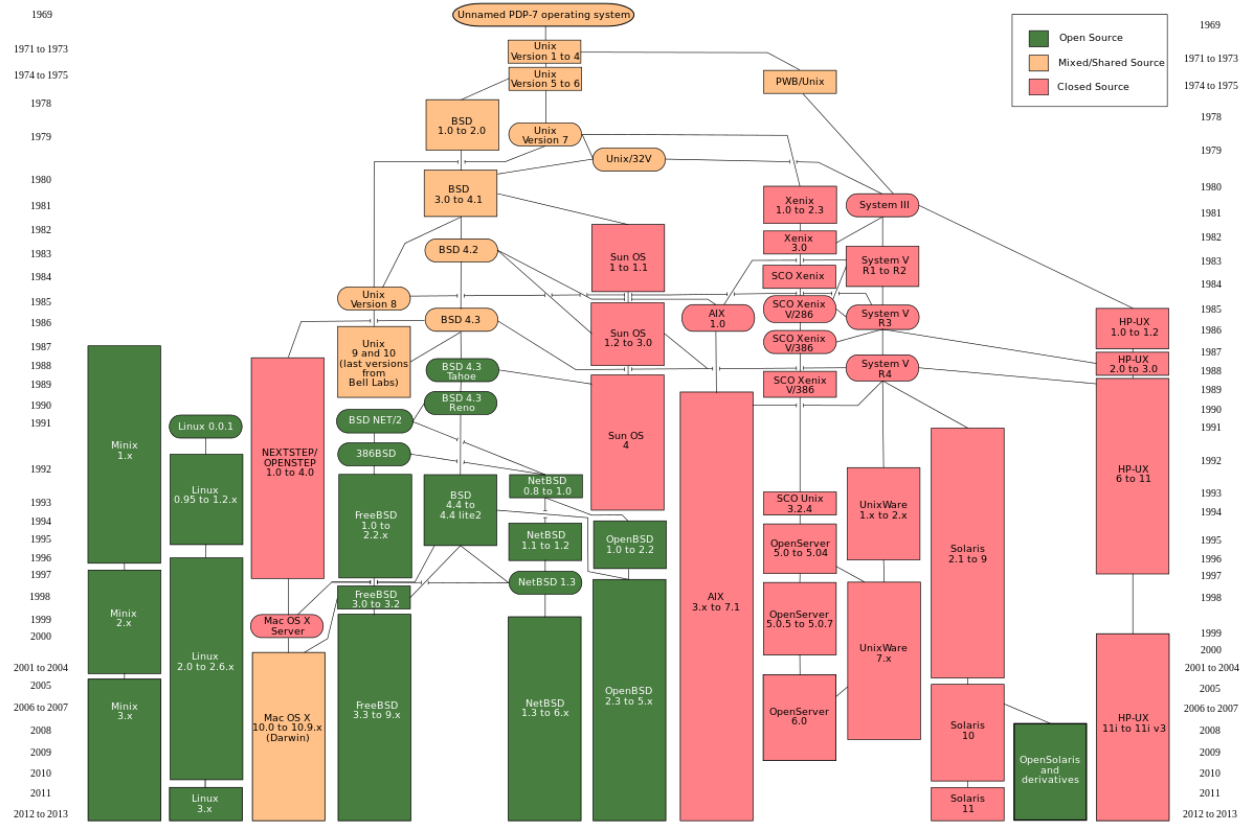


CS3210: JOS and xv6

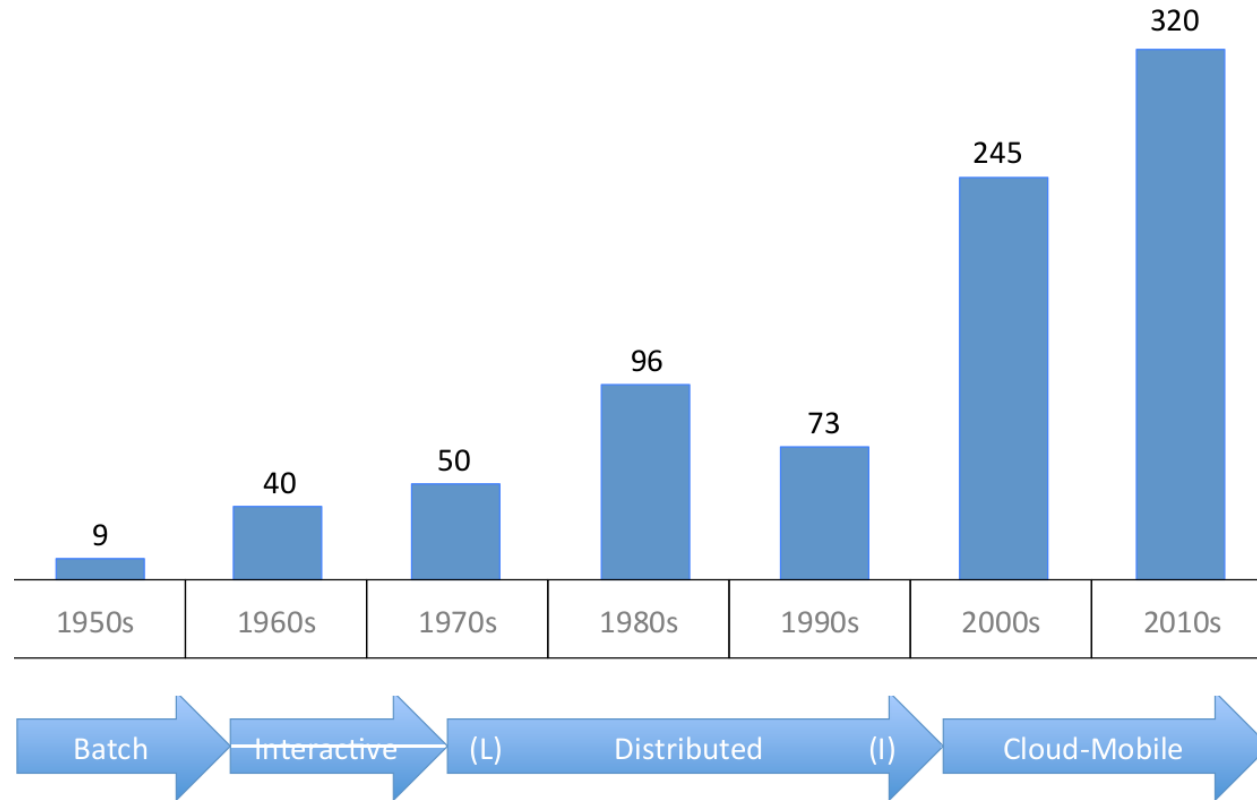
- Micro-kernel: JOS (exokernel)
- Monolithic: [xv6](#) (UNIX-like)
 - Book: [Xv6, a simple Unix-like teaching operating system](#)
 - Code: [commentary](#)

```
$ git clone git://github.com/mit-pdos/xv6-public.git
```

UNIX history (Wikipedia)

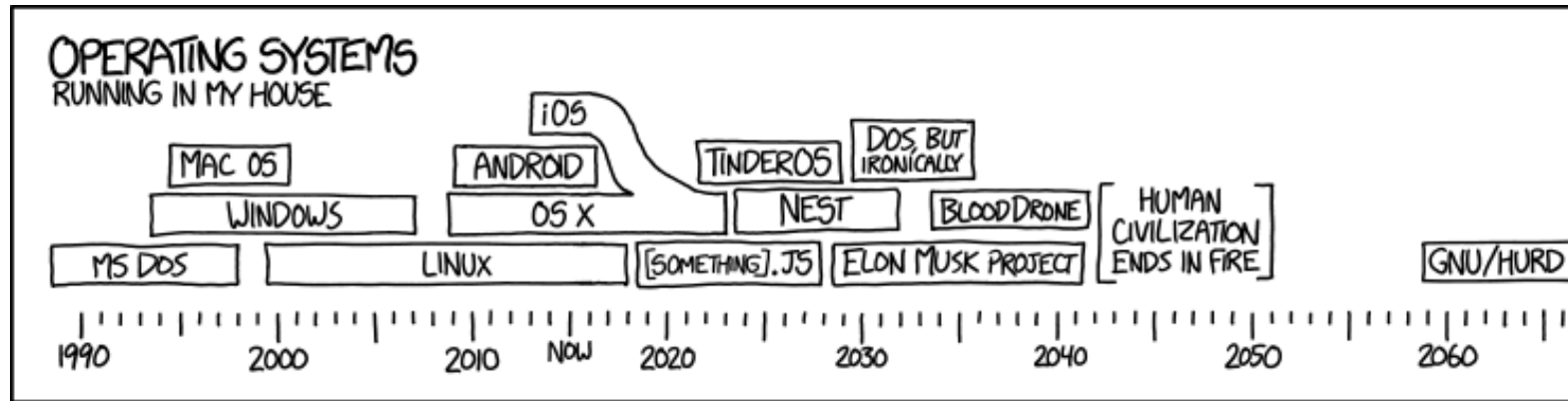


New operating systems (Peter J. Denning)



Future operating systems (xkcd)

- Large scale: lots of cores, bigger memory, huge storage
- Small scale: dust?
- Heterogeneity



Programming languages for OS development

- ASM, LISP, C, C++, Object C, Java, Smalltalk, C#, Rust, Haskell, etc
- Why not X, Y, Z?
 - e.g., Brainf***

What language is used for Linux, Windows, Mac OSX, FreeBSD, etc?

Why using C for OS development?

- Portability
- No runtime
- Direct hardware/memory access
- (decent) Usability

Prep quiz: the C programming language

- Open your laptop
- Visit [submission site](#)
- Download the quiz
- Submit your answer!

Self evaluation

- < 11/21 (50%): shall we meet next year in cs3210?
- < 15/21 (70%): do you have enough time to catch up?

First two tutorials (Jan 14/21) will cover in depth Tools and C/gdb!

Next lecture

- Tutorial: Group, Tools, Lab1
- Register [Piazza](#)
- [Lab 1: Booting a PC](#) is out (DUE: 10pm, Jan 19)
- Don't forget to submit "preparation question" (DUE: 10pm, Jan 13)

References

- [UW CSE 451](#)
- [OSPP](#)
- [MIT 6.828](#)
- Wikipedia
- The Internet