COS 318: Operating Systems

Introduction

Jaswinder Pal Singh
Computer Science Department
Princeton University

(http://www.cs.princeton.edu/courses/cs318/)

Today

- Course Staff and Logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS 318?

Course Staff and Logistics

- Instructor
  - Jaswinder Pal Singh, 423 CS Building, jps@cs.princeton.edu
    Office hours: 3-5pm Thu
- Teaching Assistants
  - CJ Bell, cbell@princeton.edu
    Office hours: TBA (e.g. 2-4pm Mon and Fri)
  - Yun Zhang, 223 CS, yunzhang@cs.princeton.edu
    Office hours: TBA (e.g. 2-4pm Mon and Fri)
- Information
  - Website: http://www.cs.princeton.edu/courses/cos318
  - **Subscribe to cos318@lists.cs.princeton.edu**

Resolve "TBD"

- Precept
  - Time: ? 8:30-9:30pm Wed
  - Location will be announced on the website
- Design review
  - ? Monday 6-9pm

COS318 in Systems Course Sequence

- Prerequisites
  - COS 217: Introduction to Programming Systems
  - COS 226: Algorithms and Data Structures
- 300-400 courses in systems
  - **COS318: Operating Systems**
  - COS320: Compiler Techniques
  - COS333: Advanced Programming Techniques
  - COS432: Information Security
  - COS471: Computer Architecture
- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks

Course Materials

- Textbook – in U-Store
  - *Modern Operating Systems*, 3rd Edition, Andrew S. Tanenbaum
  - Note: 3rd edition came out in Dec 2007, older versions are dated
- Lecture notes
  - Handout in class and on website
- Precept notes
  - Handout in precepts and on website
- Other resources – on website

Exams, Participation and Grading

- Grading
  - First 5 assignments: 50% with extra points
  - Midterm: 20%
  - Final project 20%
  - Reading & participation 10%
- Midterm Exam
  - Tests lecture materials and projects
  - Tentatively scheduled on Thursday of midterm week
- Reading and participation
  - Signup for each class and hand in your reading notes
  - Grading (3: excellent, 2: good, 1: poor, 0: none)

The First 5 Assignments

- Assignments
  - Bootup (150-300 lines)
  - Non-preemptive kernel (200-250 lines)
  - Preemptive kernel (100-150 lines)
  - Interprocess communication and driver (300-350 lines)
  - Virtual memory (300-450 lines)
- How
  - Pair up with a partner, will change after 3 projects
  - Each project takes two weeks
  - Design review at the end of week one
  - All projects due Mondays 11:59pm
- The Lab
  - Linux cluster in 010 Friends Center, a good place to be
  - You can setup your own Linux PC to do projects

Project Grading

- Design Review
  - A signup sheet for making appointments
  - 10 minutes with the TA in charge
  - 0-5 points for each design review
  - 10% deduction if you miss the appointment
- Project completion
  - 10 points for each project
  - Extra points available
- Late policy of grading projects
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%

Final Project

- A simple file system
- Grading: 20 points
- Do it alone
- Due on Dean’s date (get approx. 3 weeks)

Things To Do

- **Do not to put your code or designs or thoughts on the Web**
  - Other schools are using similar projects
  - Not even on facebook or the like
- Follow Honor System: ask when unsure, cooperation OK but work is your own (or in pairs for programs)
- For today's material:
  - Read MOS 1.1-1.3
- For next time
  - Read MOS 1.4-1.5
- Email me the information on the next slide

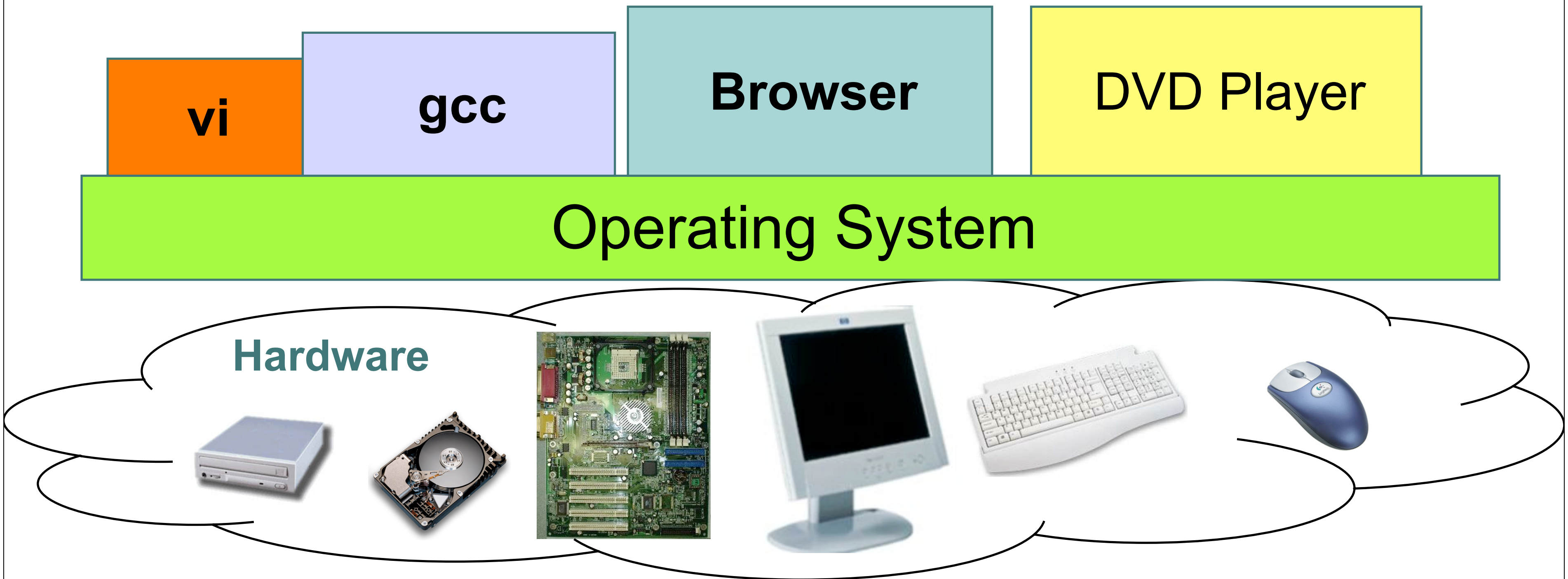
Email to jps@cs.princeton.edu

- Name
- Year
- Major
- E-mail address
- Phone #
- Picture via URL (optional, but helps me learn names)
- Why you're taking the class
- What you'd like/hope to learn

Today

- Course Staff and Logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS 318?

What Is an Operating System?



- Software that sits between applications and hardware
  - (Also between different applications, and between different users, see later)
- Provides services and interfaces to applications
- Has privileged access to hardware

User applications call OS routines for this access and for the services

What Does an Operating System Do?

- Provides a layer of abstraction for hardware resources
  - Allows user programs to deal with higher-level, simpler and more portable concepts than the vagaries of raw hardware
    - E.g. files rather than disk blocks
  - Makes finite resources seem infinite
- Manages these resources
  - Manages complex resources and their interactions for an application
  - Allows multiple applications to share resources without hurting one another
  - Allows multiple users to share resources without hurting one another

Abstraction

- Hiding underlying details, and providing cleaner, easier-to-use, more elegant concepts and interfaces instead
  - Also provides standardized interfaces despite diversity of implementation underneath
- A key principle in Computer Science
- A key to understanding Operating Systems

Example of Abstraction: Disk

- Disk hardware and operation are very complex
  - Multiple heads, cylinders, sectors, segments, each with different fininte sizes
  - Have to wait for physical movement before actually writing or reading data to/from disk
  - Data stored discontiguously for performance and reliability
  - To even read or write simple data would take a lot of coordination when dealing with hardware directly
  - Sizes and speeds are different on different computers
- OS provides simple read() and write() calls as the application programmer's interface (API)
  - Manages all the complexity transparently in conjunction with the disk controller hardware

Example of Abstraction: Networks

- Data communicated from one computer to another are:
  - Broken into fragments that are sent separately and arrive at different times and out of order
  - Waited for and assembled at the destination
  - Sometimes lost, so fragments have to be resent
  - An application programmer does not want to manage this
- OS provides a simple send() and recv() interface
  - Takes care of all the complexity, in conjunction with networking hardware

Resource Management

- Allocation
- Virtualization
- Reclamation
- Protection

Resource Allocation

- Computer has finite resources
- Different applications and users compete for them
- OS dynamically manages which applications get how many resources
- *Multiplex* resources in space and in time
  - Example of time multiplexing?
  - Example of space multiplexing?
- E.g. what if multiple applications run infinite loops?

```
while (1);
```

Resource Virtualization

- OS gives each program the illusion of effectively infinite, private resources
  - “infinite” memory (by backing up to disk)
  - CPU (by time-sharing)

Resource Reclamation

- The OS giveth, and the OS taketh away
  - Voluntary or involuntary at runtime
  - Implied at program termination
  - Cooperative

Protection

- You can’t hurt me, I can’t hurt you
- OS provides safety and security
- Protects programs and their data from one another, as well as users from one another
- E.g. what if I could modify your data, either on disk or while your program is running

Mechanisms and Policies

- Mechanisms are tools or vehicles to implement policies
- Examples of policies:
  - All users should be treated equally
  - Preferred users should be treated better

Today

- Course Staff and Logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS 318?

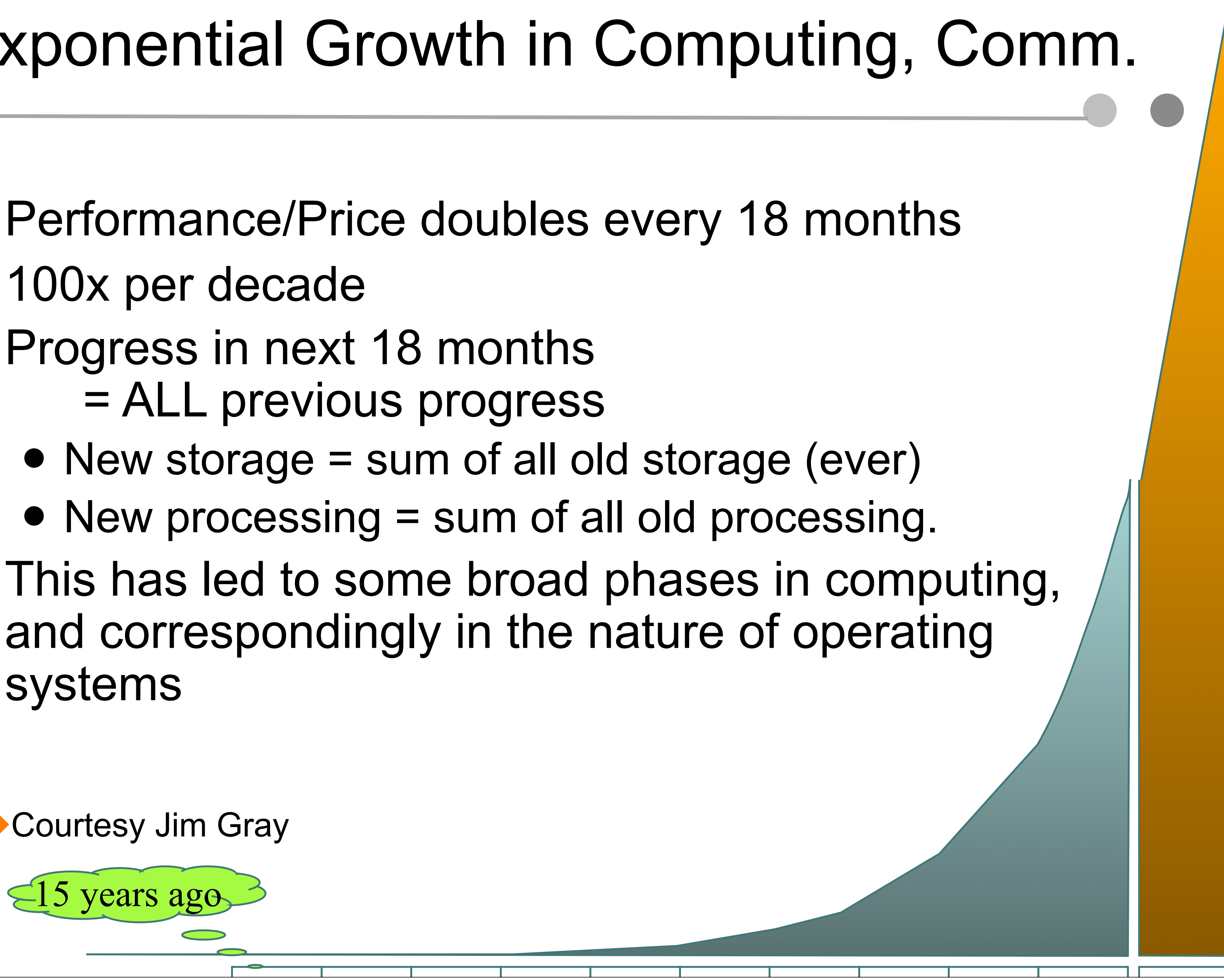
A Typical Academic Computer (1986 v. 2007)

	1986	2007	Ratio
CPU clock	4Mhz	4×3Ghz	3000x
$/machine	$60k	$600	1/100x
DRAM	1MB	2GB	2000x
Disk	50MB	0.5-1TB	10K-20Kx
Network BW	10Mbits/sec	1GBits/sec	100x
Address bits	32	64	2x
Users/machine	10s	< 1	>10x
$/Performance	$60k	$600/3000	1/200,000x

Exponential Growth in Computing, Comm.

- Performance/Price doubles every 18 months
- 100x per decade
- Progress in next 18 months
  = ALL previous progress
  - New storage = sum of all old storage (ever)
  - New processing = sum of all old processing.
- This has led to some broad phases in computing, and correspondingly in the nature of operating systems

- Courtesy Jim Gray



History of Computers and OSes

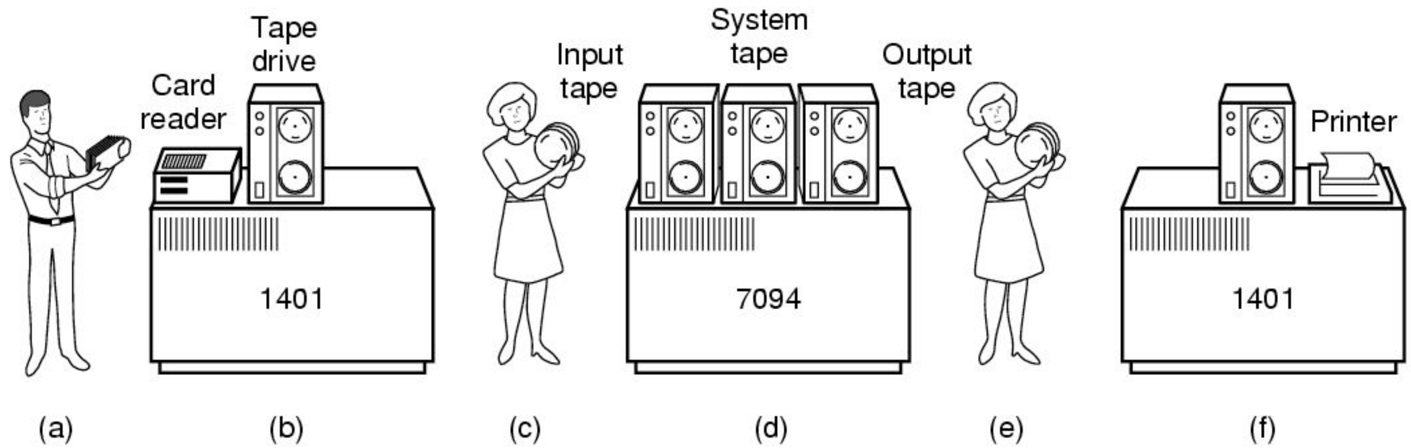
Generations:

- (1945–55) Vacuum Tubes
- (1955–65) Transistors and Batch Systems
- (1965–1980) ICs and Multiprogramming
- (1980–Present) Personal Computers

Phase 1: The Early Days

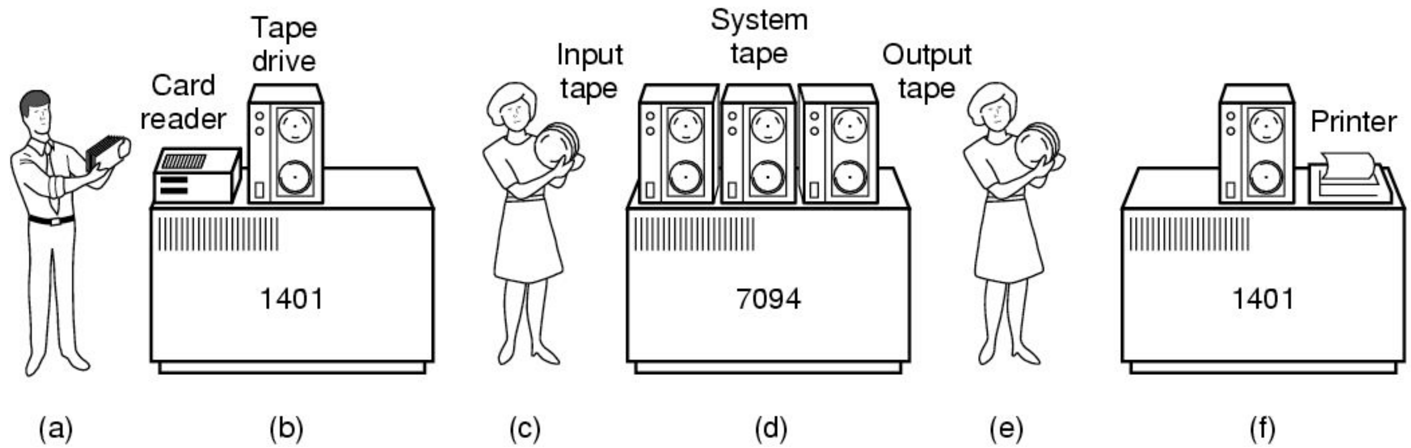
- Hardware very expensive, humans cheap
- When was the first functioning digital computer built?
- What was it built from?
- How was the machine programmed?
- What was the operating system?
- The big innovation: punch cards
- The really big one: the transistor
  - Made computers reliable enough to be sold to and operated by customers

Phase 2: Transistors and Batch Systems



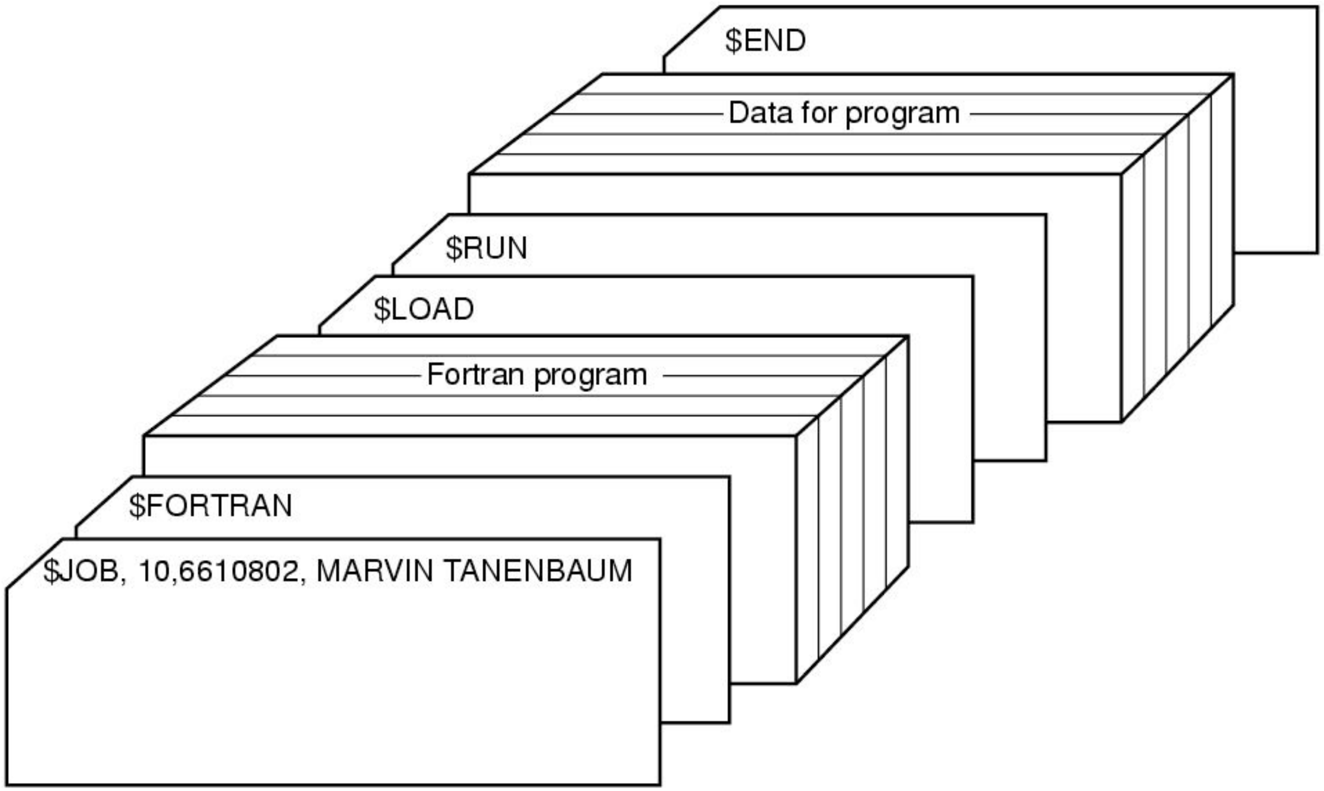
- Hardware still expensive, humans relatively cheap
- An early batch system
  - Programmers bring cards to reader system
  - Reader system puts jobs on tape

Phase 2: Transistors and Batch Systems



- An early batch system
  - Operator carries input tape to main computer
  - Main computer computes and puts output on tape
  - Operator carries output tape to printer system, which prints output

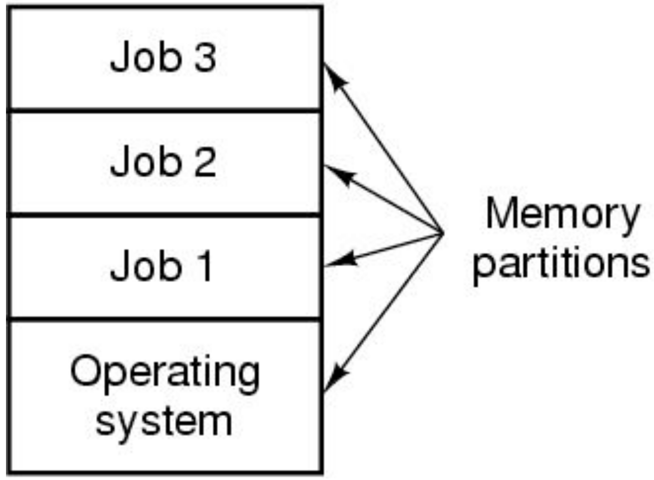
Punch cards and Computer Jobs



Phase 3: ICs and Multiprogramming

- Integrated circuits allowed families of computers to be built that were compatible
- Single OS to run on all (IBM OS/360): big and bloated
- Key innovation: multiprogramming
  - What happens when a job is waiting on I/O
  - What if jobs spend a lot of the time waiting on I/O?

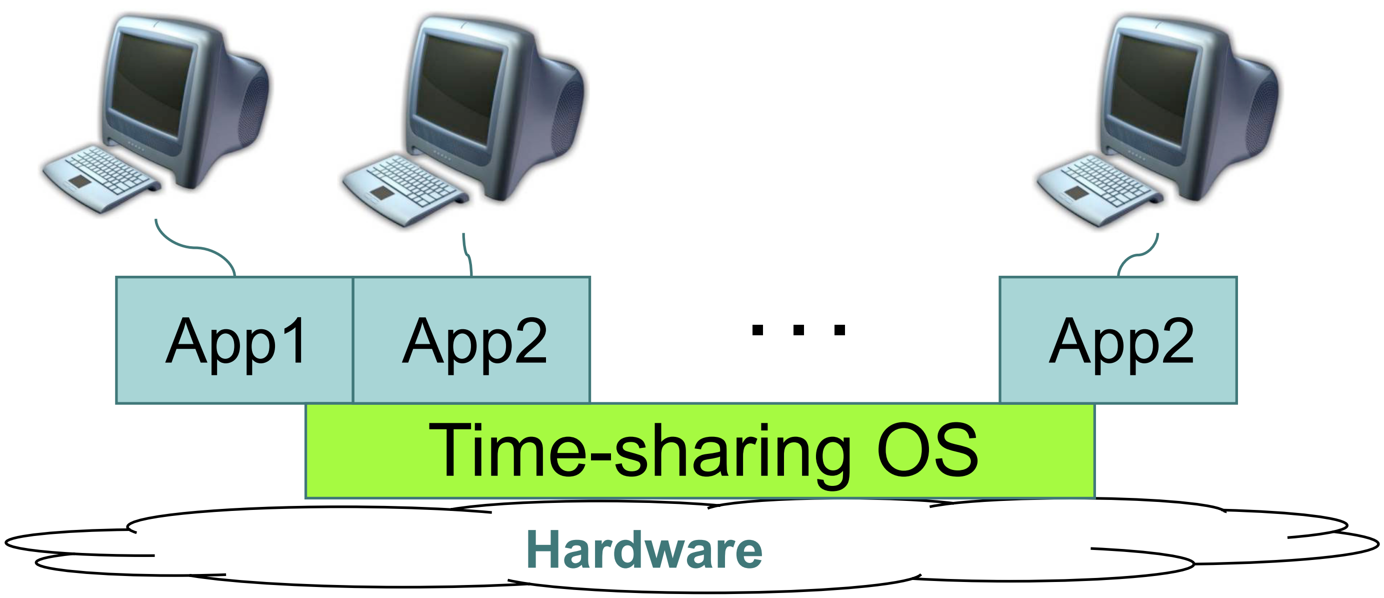
Phase 3: ICs and Multiprogramming



- Multiple jobs resident in computer's memory
- Hardware switches between them (interrupts)
- Hardware protects from one another (mem protection)
- Computer reads jobs from cards as jobs finish (spooling)
- Still batch systems: can't debug online
- Solution: time-sharing

Phase 3: ICs and Multiprogramming

- Time-sharing:
  - Users at terminals simultaneously
  - Computer switches among active ‘jobs’/sessions
  - Shorter, interactive commands serviced faster



Phase 3: ICs and Multiprogramming

- The extreme: computer as a utility: MULTICS (late 60s)
  - Problem: thrashing as no. of users increases
  - Didn't work then, but idea may be back
  - Let others administer and manage; I'll just use
- ICs led to mini-computers: cheap, small, powerful
  - Stripped down version of MULTICS, led to UNIX
  - Two branches (Sys V, BSD), standardized as POSIX
  - Free follow-ups: Minix (education), Linux (production)

Phase 4: HW Cheaper, Human More Costly

- Personal computer
  - Altos OS, Ethernet, Bitmap display, laser printer
  - Pop-menu window interface, email, publishing SW, spreadsheet, FTP, Telnet
  - Eventually >100M units per year
- PC operating system
  - Memory protection
  - Multiprogramming
  - Networking

Now: > 1 Machines per User

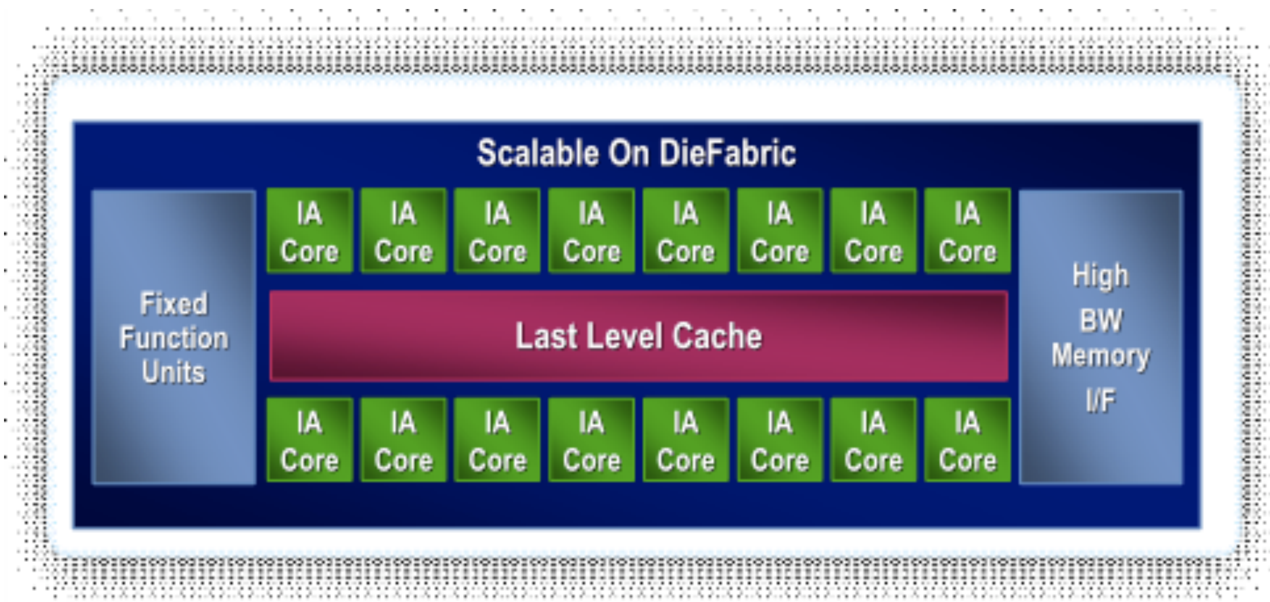
- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle
- OS are specialized
  - Embedded OS
  - Specially configured general-purpose OS

Now: Multiple Processors per Machine

- Multiprocessors
  - SMP: Symmetric MultiProcessor
  - ccNUMA: Cache-Coherent Non-Uniform Memory Access
  - General-purpose, single-image OS with multiproccesor support
- Multicomputers
  - Supercomputer with many CPUs and high-speed communication
  - Specialized OS with special message-passing support
- Clusters
  - A network of PCs
  - Commodity OS

Now: Multiple “Cores” per Processor

- Multicore or Manycore transition
  - Intel and AMD have released 4-core and soon 6-core CPUs
  - SUN’s Niagara processor has 8-cores
  - Azul Vega8 now packs 24 cores onto the same chip
  - Intel has a TFlop-chip with 80 cores
  - Ambric Am2045: 336-core Array (embedded, and accelerators)
- Accelerated need for software support
  - OS support for many cores; parallel programming of applications



Summary: Evolution of Computers

60’s-70’s - Mainframes

- Rise of IBM

70’s - 80’s – Minicomputers

- Rise of Digital Equipment Corporation

80’s - 90’s – PCs

- Rise of Intel, Microsoft

Now – Post-PC

- Distributed applications

Summary: Evolution and Implications for OS

	Mainframe	Mini	Micro
System $ / Worker $	10:1 – 100:1	10:1 – 1:1	1:10-1:100
Goal	System utilization	Overall cost	Productivity
Target	Capacity	Features	Ease of Use

Today

- Course Staff and Logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS 318?

Why Study OS?

- OS is a key part of a computer system
  - It makes our life better (or worse)
  - It is "magic" to realize what we want
  - It gives us "power"
- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware: all hw becoming parallel
  - OS is great way to learn concurrent programming
- Understand how a system works
  - How many procedures does a key stroke invoke?
  - What happens when program references 0 as a pointer?
  - Real OS is huge and impossible to read everything, but building a small OS will go a long way

Why Study OS?

- Important for studying further areas
  - Networking, distributed systems, …
- Full employment
  - New hardware capabilities and organizations
  - New features
  - New approaches
  - E.g. handheld computers, Java, WWW
  - Engineering tradeoffs, keep changing as the hardware changes from below and the needs of apps from above
- Lots of jargon: sound smart (or super-nerdy)

Today

- Course Staff and Logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
- What's in COS 318?

What Is in COS 318?

- Methodology
  - Lectures with discussions
  - Readings with topics
  - A lot of design and rationale, some theory, a fair bit of practice
  - Six projects to build key aspects of a basic OS
- Covered concepts
  - Operating system structure
    - Processes, threads, system calls and virtual machine monitor
  - Synchronization
    - Mutex, semaphores and monitors
  - I/O subsystems
    - Device drivers, IPC, and introduction to networking
  - Virtual memory
    - Address spaces and paging
  - Storage system
    - Disks and file system

What is COS 318 Like?

- Is tt theoretical or practical?
  - Focus on concepts, but also getting hands dirty in projects
  - More about engineering tradeoffs, constraints, optimization and imperfection than about optimal results and beautiful mathematics
  - High rate of change in the field yet lots of inertia in OSes
- Is it easy?
  - No. Fast paced, hard material, a lot of programming
- What will enable me to succeed?
  - Solid C background, pre-reqs, tradeoff thinking
  - NOT schedule overload