CSc 360
Operating Systems
Introduction

Jianping Pan

Fall 2023

About the course

- *Introduction* to **Operating Systems**
  - (A01/2) **MR 8:30--9:50am**, **ELL168**
  - Bright: “Fall 2023 CSC 360 A01 - A02 **X**”
    - lectures, tutorials, additional resources, etc
    - assignments, gradebook, etc
    - **discussion channel hosted on UVic Teams**
  - prerequisites
    - Data structures (CSc **225** or 226)
    - Computer architecture (CSc **230** or CENG 255)
    - System programming (CSc/SENG **265**, CENG)

Message from Undergrad Advisor

- Email: cscadvisor@uvic.ca
- Do not have the prerequisite course(s)?
  - need to obtain a waiver
  - otherwise, prerequisite drop after the first week
- Taking the course more than twice?
  - need to have a letter from the Chair and the Dean
  - otherwise, being dropped from the class
- Make sure you can receive email from Bright!
  - important email announcement from time to time
  - use UVic Teams first

About the course instructor

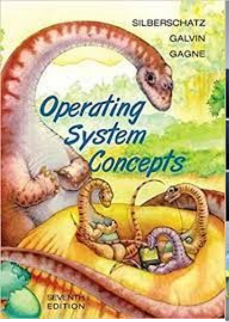
- Dr. Jianping Pan <pan@uvic.ca>
  - use UVic Teams first, as email not always reliable
    - to help, always include **[csc360]** in your email subject line
  - office hours: **M 1--3pm**
    - or by appointment
    - **on UVic Teams**
  - work experience
    - UVic, industry research labs, UWaterloo postdoc
  - research area
    - computer networks and distributed systems
    - http://web.uvic.ca/~pan

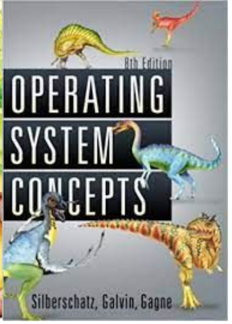
more about the instructor: http://tinyurl.com/panpcs

About the lab/tutorial instructors

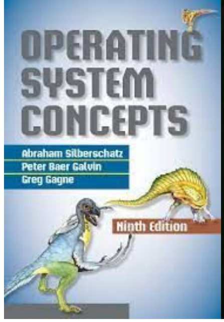
- Zhiming Huang
  - their email on Brightspace
- Tutorials: start this week!
  - Please attend your registered section only!
    - tutorial lectures
      - **C**, libc, sockets, pthreads, ...
    - assignment help
      - spec go-through, common problems, ...
    - practice problems

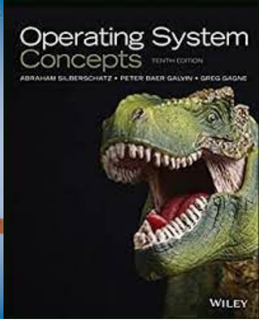
T01: ECS104 T 1:30pm
T02: ECS104 W 1:30pm
T03: ECS108 F 12:30pm





Course materials




Operating System Concepts
TENTH EDITION
ABRAHAM SILBERSCHATZ • PETER BAER GALVIN • GREG GAGNE
WILEY

- Required textbook
  - Operating system concepts, 7th or **newer** editions
    - 6th and other editions: different chapter schedules
  - online resources

    http://codex.cs.yale.edu/avi/os-book/
    or **http://os-book.com/**
    - errata, slides, practice exercises and solutions
- Explore further
  - web links @ course web site
  - Google!

FUN: why dinosaurs on the book cover?

What's operating system?

- OS is a *special* program (computer software)
  - run "directly" on computer hardware
    - **kernel**: CPU, memory, I/O, etc
  - support many other programs
    - **system** programs: shell, compiler, assembler, etc
    - **application** programs: editor, browser, game, etc
- Examples
  - **Linux**/Unix, MacOS, Windows, and many others

  -- **Android**, iPhone iOS, Symbian, WP/M, Maemo, etc

bare-metal virtualization? hypervisor?

What does OS offer?

- Computer systems
  - hardware, system programs (**OS+**), apps, users
- OS: between hardware and other software
  - present a virtual machine to other software
    - hide hardware details, extend hardware features
    - hardware ++
  - provide controlled access to hardware
    - restrict hardware access, manage hardware resources
    - hardware --

Why do we need OS?

- C&C: control and coordinate
  - allow a program to use computer properly
    - program execution, error detection, …
  - allow many programs to use computer properly
    - resource allocation, conflict arbitration, …
- S&S: share and separate (protect)
  - share btw devices, programs, computers, users
  - protect one from all the others

FUN: why “360”?

Why do we study OS?

- How to use OS
  - not as a computer user!
    - point-and-click or copy-and-paste
  - but as a system programmer!
    - programming!!!
- How to design OS
  - or design any *complex*, large-scale software
- How to implement OS
  - or write any *effective* and *efficient* code

* human OS: now we can learn something from computer OS ;-)

Course objectives

- “An introduction to the major concepts of modern operating systems and the relationship between the operating system and the architecture of computer systems.”
- Selected topics
  - **process**: process, thread, scheduling, synch
  - **memory**: memory management, virtual memory
  - **storage**: file systems, I/O systems

Your participation

- Lectures
  - essential for doing well in assignments/exams
- Assignments (55% total)
  - 3 programming assignments + 1 written assignment
- Tutorials
  - extra details and hints on assignments
- Exams
  - 3 midterms (15% each on Oct 5, Nov 2, Dec 4, 2023)
- See the course outline for detailed schedules

* each exam focuses on one major module

Suggested approach

- Before lectures
  - read textbook; preview video; find questions
- Attend lectures
  - take notes; ***ask*** questions; answer questions!
- After lectures
  - read textbook; explore further
  - write assignments (start early!)
  - get help and help others (discussion forum, etc)
- Do attend tutorials

Common *mistakes/complaints*

- "Slides are already online"
  - Lectures are much more than just browsing slides
  - Pay attention to in-class questions/discussion too!
- "**Slides are too brief**"
  - Slides are just guidelines to navigate/understand
  - Take notes in class and read the textbook!
- "Start to do assignments on the due date"
  - Simple fact: you cannot finish, or even start, them
  - Start early and let us know if you have questions!

read slides the night before the exam/assignment is NOT useful

More systems courses

- *Computer networks (CSc 361 ~ networkOS)*
  - suggestion: take it after CSC360
- *Advanced computer networks (CSc 466)*
- Advanced communication networks (CSc 467)
- Wireless and mobile networks (CSc 463)
- Network management and security (topics/DS)
- Embedded systems (CSc 460)
- ***Multimedia systems (CSc 461)***
- Distributed systems (CSc 462)

Your feedback

- Teaching/learning is interactive
  - two-way communications
- Let me know
  - what you think about lectures, assignments, tutorials, exams, topics, …
  - what you want to know more or probe further
- You can *always* reach me
  - in class, during office hours, by Teams/email

Course Reps

- Please volunteer yourself!
  - **A**nonymize
  - **A**ggregate
  - **A**mplify
- Feedback to the teaching team
  - lecture and lab/tutorial instructors, markers
  - meet once a month face-to-face or by Teams
  - communicate electronically when necessary

please indicate your willingness to volunteer in your A0. thanks!

Course policies

- See official course outline
  - – late assignments, mark appeals, etc
  - – academic integrity
    - zero tolerance on cheating!
  - – accommodation, etc
- No group assignment/project
  - – collaboration/participation is encouraged
  - – responsibility: your submitted work is yours
  - – obligation: give credits to references

whatever you can find elsewhere, we've probably already had it

Assignment 0

- Due on this Friday, Sept 15, 2023
- Through Brightspace
  - Assignment -> A0
    - your academic program
    - things you already known in OS
    - things you want to know in OS
    - issues with course logistics?
    - willing to be the course rep? are you in T01/2/3?
    - your Brightspace and Teams profile photo updated
      - let me know you! e.g., for reference letters in future

Need more challenge?

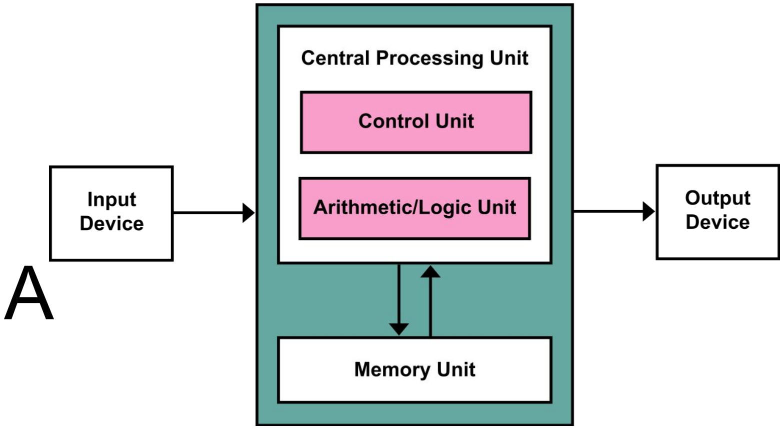
- ACM International Collegiate Programming Contest (ICPC)
  - UVic has been participating in the last few years
  - https://oac.uvic.ca/programmingclub/
- BCNET Broadband Innovation Challenge, or DMC
  - any applications running over broadband networks
  - previous winning projects (some from UVic)
    - http://www.cs.uvic.ca/~pan/bcnet
- Other student competitions and clubs
  - https://www.uvic.ca/ecs/computerscience/undergraduate/student-clubs/index.php

This lecture so far

- An introduction to the course
  - who, when, where, what, and why
  - course materials
  - course objectives
  - course topics
  - you and the course

A *quick* review and a quick overview

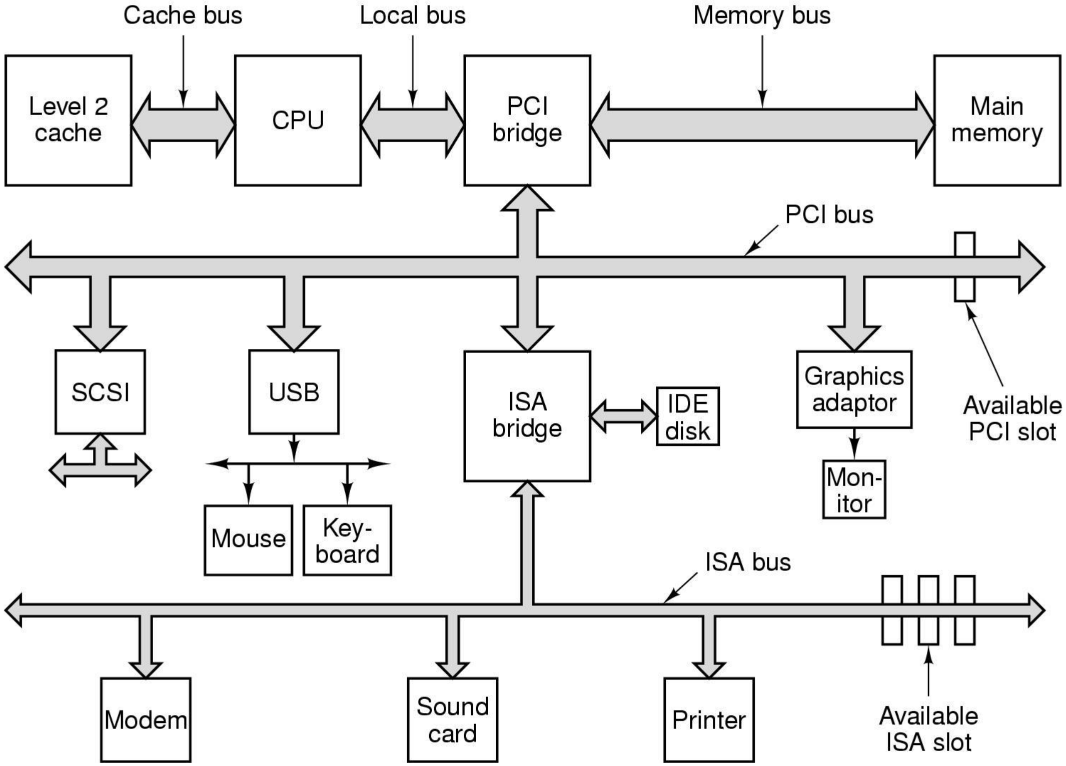
- Computer organization and architecture
  - CPU
  - Memory
  - I/O: polling, interrupt, DMA
- Operating system
  - The software that manages CPU, memory and I/O, as many more
  - so let's review CPU, memory and I/O first



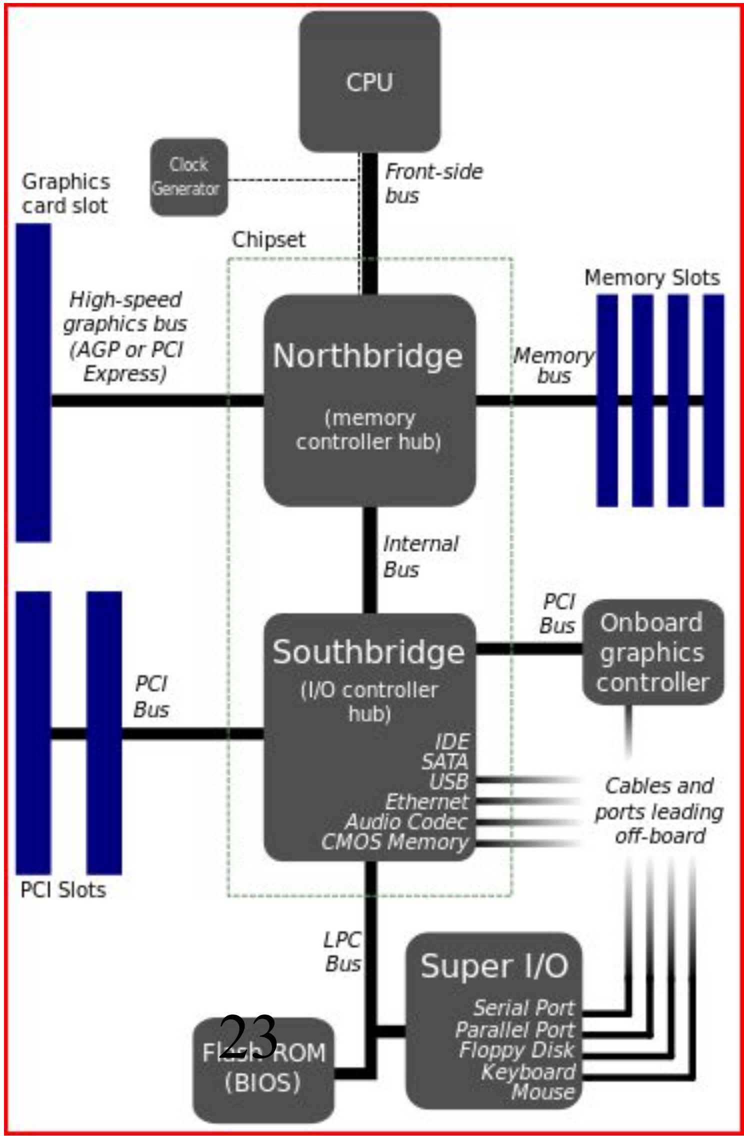
Computer Organization

- CPU
- Memory
- I/O

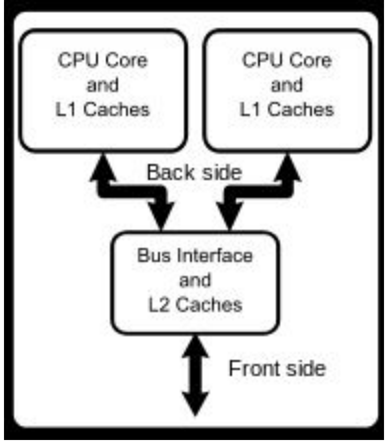
then



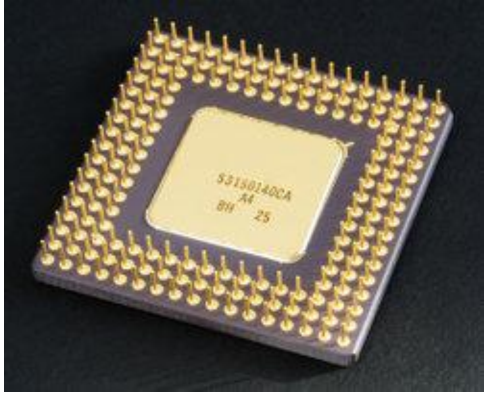
now

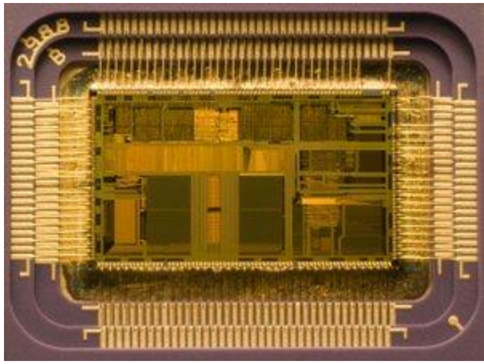


* north vs south bridge?

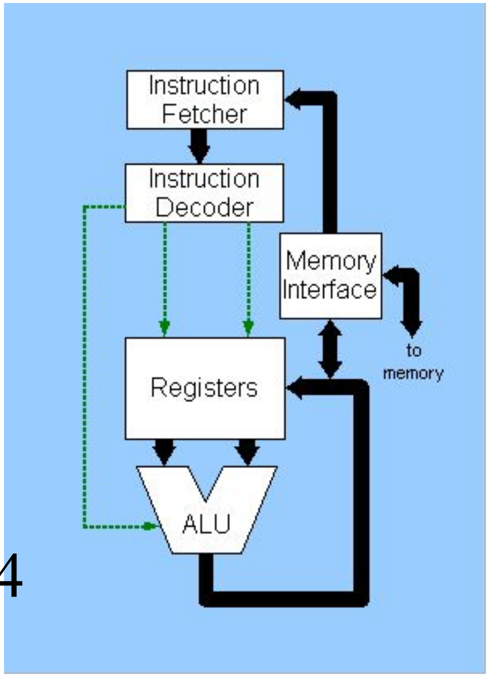


CPU





- Access
  - pins: address, data, control, status
- Internals
  - program counter (PC)
  - registers: address, data, control, flags
  - arithmetic logic unit (ALU), FPU, etc
- Benchmarks
  - clock (GHz), instructions/cycle, MIPS



* FLOPS * multi-core * GPU/GPGPU * top500.org

CPU operations

- Fetch
  - – retrieve instructions from memory (cache)
- Decode
  - – instruction: operator, operands; microcode
- Execute
  - – arithmetic/logic operation
  - – move data between register, memory, I/O
  - – change execution flow

* CISC vs RISC * x86 vs ARM, etc * what's in your (i)phone?

RAM

Memory

SSD

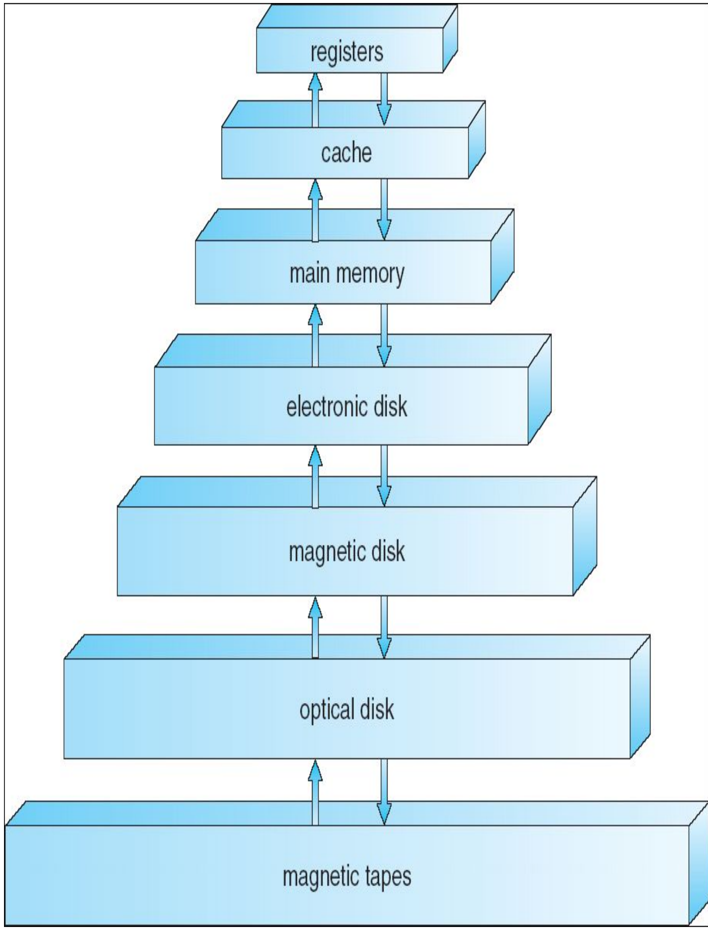
NVMe

- Access
  - linear address
  - segmented address: segment, index
  - physical address: cylinder, header, sector (disk)
- Benchmarks
  - clock (MHz); read/write cycles
  - width (bits)
  - throughput (Mbps)

* there are many different kinds of memory (devices)

Memory hierarchies

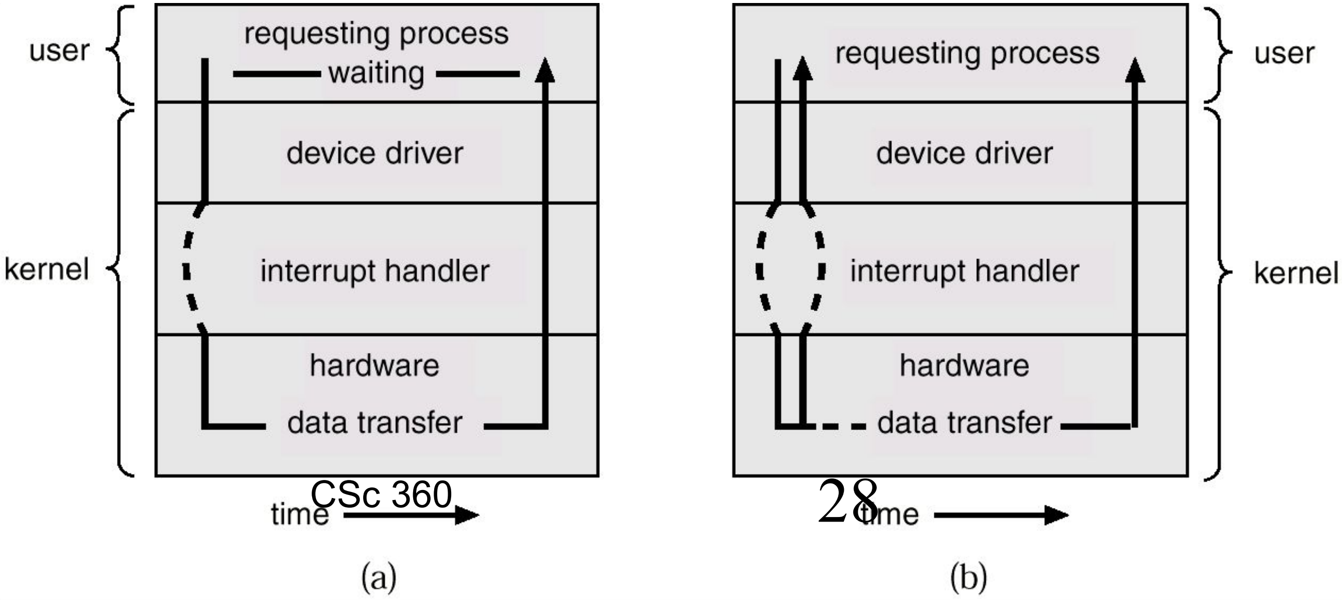
- Speed vs. size (vs. cost)
  - registers: inside CPU
  - cache: transparent to programs
  - memory: main storage
    - DRAM, DDR, SDRAM, SRAM, etc
  - disks: secondary storage
    - electronic, magnetic, optical, etc
  - tapes: backup storage
  - networked storage: NAS/SAN
- Caching



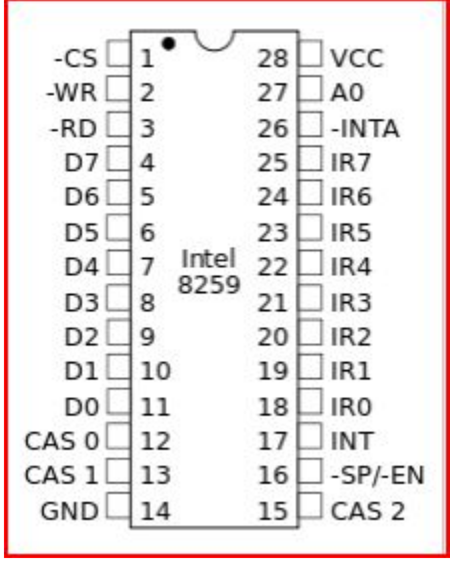
* who's the largest consumer of HDDs and tapes nowadays?

I/O

- A large variety of input/output devices
  - keyboard/mouse, video, audio, network, etc
- Access
  - Address
    - port numbers
    - I/O vs. memory space
  - Interrupt
  - Direct memory access (DMA)
- Synchronous vs asynchronous



Interrupts



- Asynchronous operation
- Non-maskable interrupts
  - e.g., hardware fault
- Hardware interrupts
  - hardware events: e.g., I/O completion
  - interrupt controller: priority & arbitration
- Software interrupts
  - trap, system call

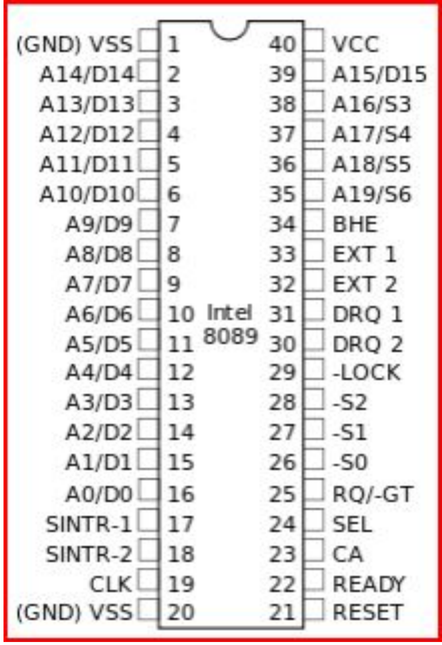
https://en.wikipedia.org/wiki/Interrupt

Interrupt handling

- Save current state
  - CPU counters, registers, flags at system stack
- Update program counter
  - interrupt controller; interrupt vectors
- Execute interrupt routine
- Restore previous state
- Multiple interrupts
  - priority, masking, reentry

e.g., fire bell rings (for a drill)

DMA



I/O controller

- High-speed I/O, bulk data transfe
- DMA controller
  - – source/destination address
  - – counter: the amount of data to be moved
- DMA handling
  - – program DMA controller
  - – execute DMA *concurrently*
  - – issue an interrupt on DMA completion

https://en.wikipedia.org/wiki/Direct_memory_access

Q: compare interrupt vs DMA

Computer architectures

- Single-processor systems
- Multi-processor systems
  - symmetric multiprocessing (SMP)
- Cluster systems
  - interconnected systems
- Distributed systems
  - networked systems
- Grid systems → Cloud → *Fog* systems

* the other end of the spectrum: embedded systems

OS: historical view

- Requirements evolve
- Was
  - computers were more expensive than users
  - goals: make computers more efficiently used
  - results: share computers
- Now
  - users become more “expensive” than computers
  - goals: make computers more effectively used
  - results: *share* users (among many computers)

* network OS; data-center OS

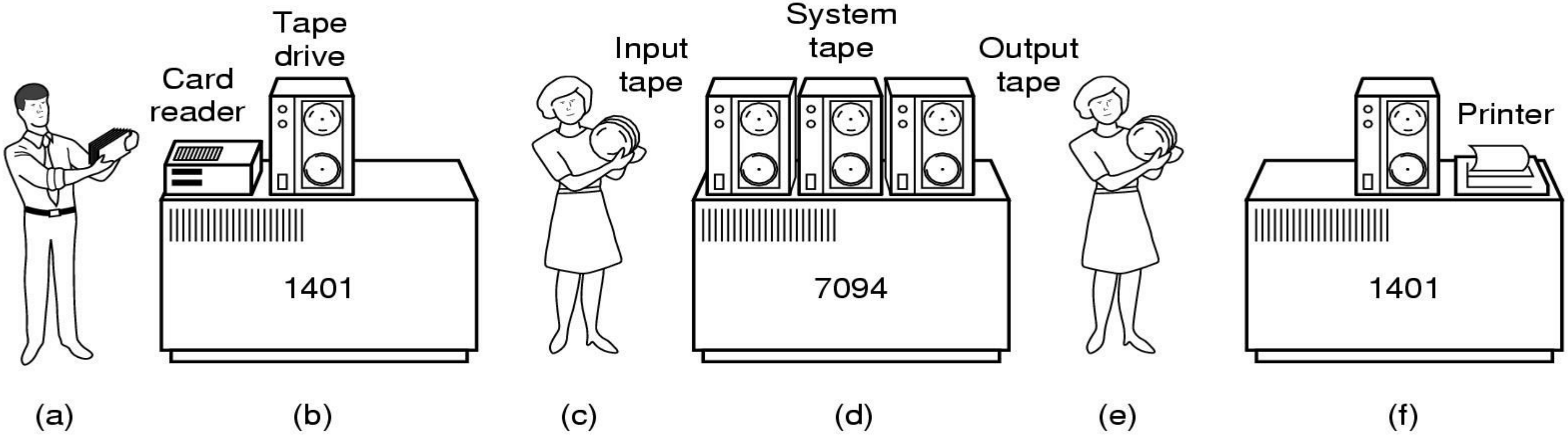
* “attention” war

OS: generations

- Uniprogramming
  - “One program at a time”
    - start, execute, {wait, execute}*, finish
    - wait for: input/output, other programs, etc
    - CPU may be idle most of the time
- Multiprogramming
  - “Many programs at a time”
    - try to keep CPU always busy
    - handle multiple programs at the same time
    - “share” (a) CPU

Batch processing

- Load a pool of jobs
- Execute one job *until* it is blocked
- Pick another one to execute



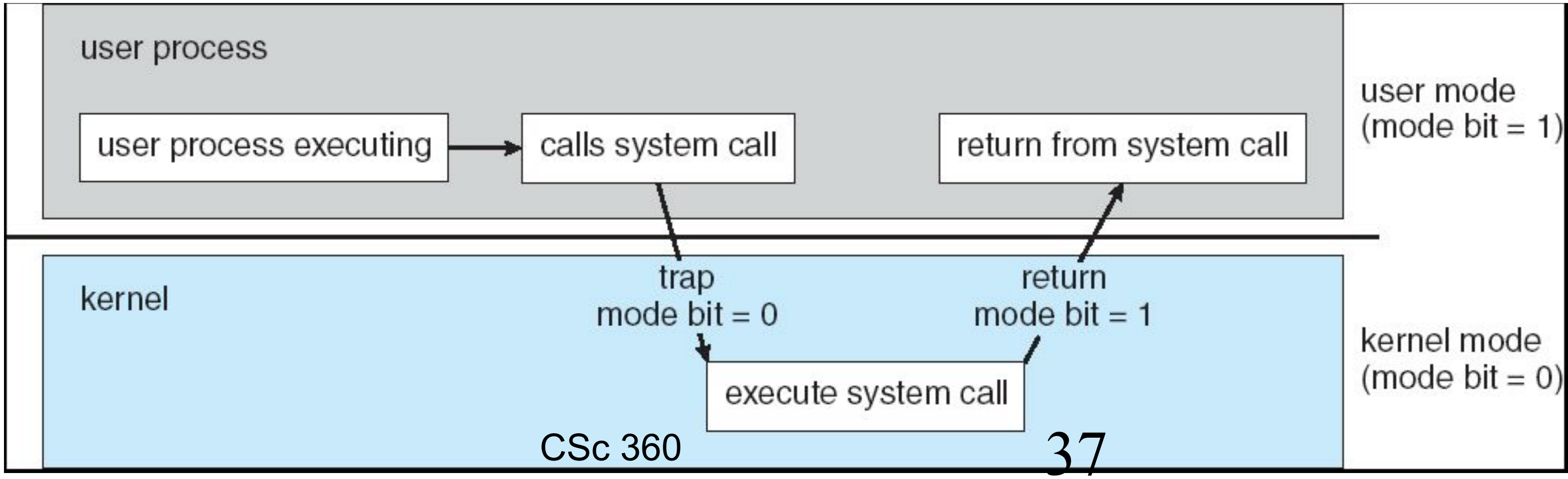
Time sharing

- Execute one job up to a certain time
  - e.g., hardware timer with counter
- Switch to another one to execute
  - job scheduling, memory swapping
- Seem to execute *many* jobs at the same time
- Batch processing vs time sharing
  - job responsiveness
  - switching overhead

Q: compare batch processing vs time sharing

OS: operations

- Interrupt the current job
  - *yield*: system call trap (e.g., I/O)
  - *yank*: hardware timer interrupt
  - how about an “abusive” job?
- Dual-mode operation
  - user mode for regular applications
  - kernel mode with privileged instructions
  - trap: user to kernel entry



Process management

- Process: a running program
  - vs thread
- Create, delete, suspend, resume process
  - resource allocation: CPU, memory, I/O, etc
- Schedule processes/threads
- Synchronize processes
- Communicate between processes
- Handle deadlocks

Memory management

- (Main) memory
  - store instructions for execution
  - store data for processing
- Keep track available memory
- Allocate and reclaim memory
  - provide protected access
  - trap invalid access
- Swap in/out (virtual) memory

* memory leak? garbage collection?

Storage and I/O management

- “In Unix, everything is a file”
  - a logical interface: open, read, write
- Create and delete files and directories
  - directory is a special file
  - file system hierarchy
- Manipulate files and directories
  - provide protected access
  - handle device-specific issues (disks, etc)

* main reason to see OS (installation disc) grow in size (drivers)

User management

- Authentication
  - who's who
  - user credentials (e.g., password, token)
- Authorization
  - what can do what
  - access control (e.g., read, write, execute)
- Accounting
  - what has been done (e.g., logging)

Specialized OS

- Different requirements and constraints
  - real-time systems
    - “hard” real-time OS in embedded systems
    - “soft” real-time OS in multimedia systems
  - handheld systems
    - almost a full-blown OS, with resource constraints
  - embedded systems
    - very severe resource constraints

The 2nd half of the lecture

- An overview on operating systems
  - Multiprogramming
    - Batch processing vs time sharing
  - Dual-mode operations
  - Issues in
    - process management
    - memory management
    - file management

Next lecture

- Interfaces to OS
  - – CLI, GUI, system calls, API
  - – read OSC7 Chapter 2 (or OSC6 Chapter 3)

* don’t forget your A0 by Friday, Sept 15 through Brightspace