Computer Architecture and Operating Systems

CS 6013 – Systems 1
Lecture 1: Introduction

Master of Software Development (MSD) Program
Kahlert school of computing, university of Utah
J. Davison de St. Germain
Spring 2024

Lecture 1 – Topics

- Course Information
- A Brief History of Operating Systems
- Overview of Course Topics

Accreditation

- These slides have been adapted from John Regehr’s CS 5460 class, and Ben and Varun’s past CS 6013 classes.

Course Resources

- TAs: Gloria, Avishek
  - TA Office Hours: TBA
- Textbook: Operating Systems: 3 Easy Pieces
  - OSTEP

- See also: Operating Systems Concepts, Silberschatz, Galvin, and Gagne (aka the Dinosaur Book)

Course Management

- All course content will be posted on Canvas
- All assignments will be submitted through Canvas
- Grades will be displayed on Canvas, though we will use Gradescope for Exams

Collaboration vs Cheating

- It’s fine to discuss solution strategies with your classmates
  - To avoid duplication of strategies or code, we recommend you wait about 30 minutes after discussion to start coding / writing.
- Do not…
  - Copy code from another student
  - Even look at code from another student
  - Copy code from the web
  - Ask for answers on StackOverflow or a similar web site
- See Syllabus

Overview

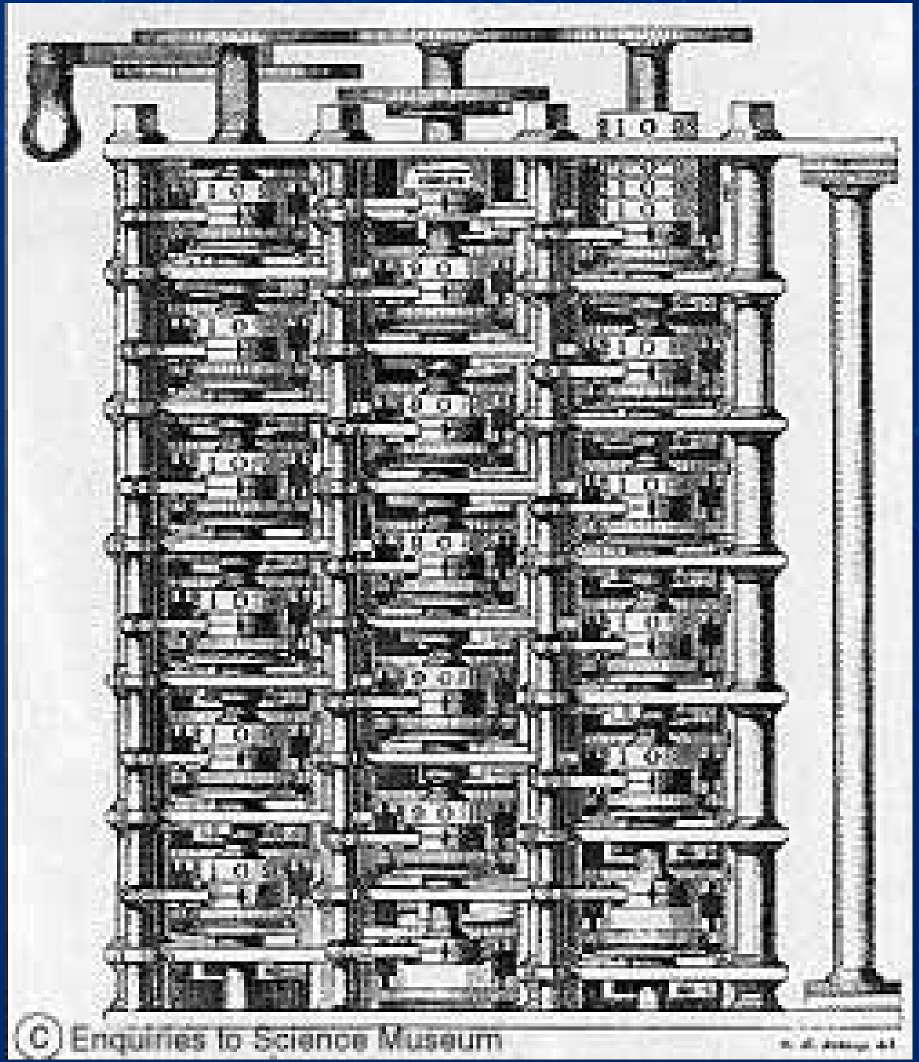
- Course will present a high-level view of
  - computer architecture (a little)
  - operation systems (a lot)
- This course helps set the MSD program apart from boot camps
  - Specifically, the knowledge of some of the underlying details of OSes and how they interact with hardware
- We will look at the pieces of a computer
  - Memory, motherboard, CPU, graphics cards, etc.
- We will learn a little assembly language. Why?
  - To get an idea of how high-level code gets converted into assembly
  - See how assembled code can be combined with compiled code
  - You won't be an expert in assembly – but you don't need to be for most jobs
- Course Time Management
  - Different from last semester. Don't leave things until the day they are due.

Operating Systems

- The operating system (OS) is *just another program!*
- What is an Operating System?
  - Hard to define…
  - Discipline arose historically from a set of problems. Definition has changed over the years.
    - Set of programs…
      - Including the “Kernel”
  - Best to start at the beginning.

A Brief History of Operating Systems

- Pre-history (pre-1945)
  - Charles Babbage (1792-1871) & Ada, Countess of Lovelace (1815-1852)
    - Babbage: 1st computer architect
    - First digital computer – wheels and gears with different teeth. Powered by cranking a handle.
    - “Analytical engine”
    - Never actually got it to work (although others subsequently have)
    - No operating system: programmer programmed to raw hardware
    - Ada: 1st computer programmer
  - Trivia Question: Who was Ada’s father?
  - Non-trivia question: difference between digital and analog?



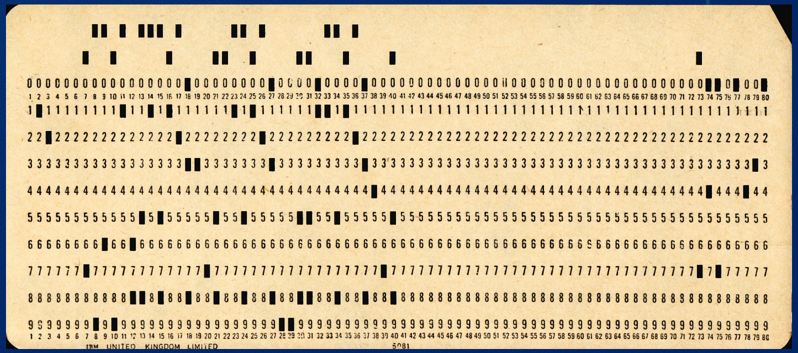
OS History – Phase 1 (~1945-1965)

- Hardware is very expensive, humans are cheap!
- One user at the console
  - One function at a time (no overlap between computation and I/O)
  - User sitting at console to debug
  - First OSes: Just common library routines

IBM Model 701 (Early 1950's)

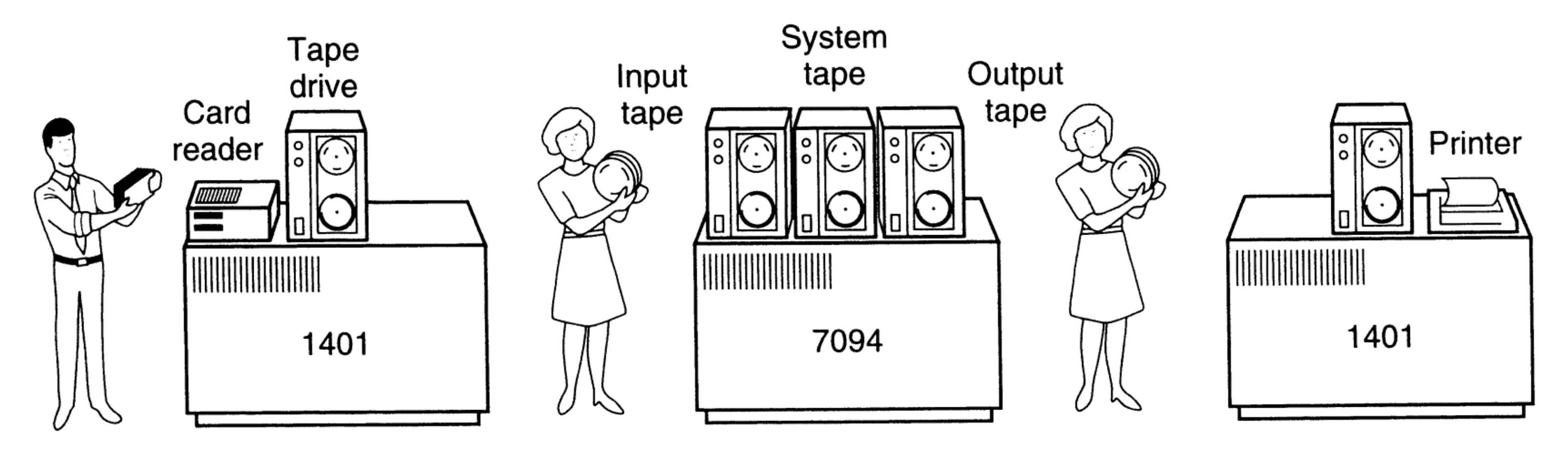
OS History – Phase 1 (~1945-1965)

- Batch processing: load, run, print, dump, repeat
  - Put up glass walls around computer
  - Users give program (punch-cards or tape) to human who schedules jobs
  - OS loads, runs, and dumps user jobs
  - Non-interactive batch processing
  - Efficient use of HW (and in some ways programmer time.)
  - Debugging was hard, core dumps
  - Short jobs starve behind large ones



OS History – Phase 1 (~1945-1965)

- Data channels and interrupts
  - Buffering and interrupt handling in OS ("batch monitor")
  - Spooling (SPOOL: Simultaneous Peripheral Operation On-Line)
  - No protection - one job running at a time!
  - Improves performance by running computation and IO in parallel



IBM 7094

- Early 1960s
- 50K Transistors
- Much smaller and uses less power than vacuum tubes
- Modern CPU has several billion transistors

OS History – End of Phase 1 (Mid-1960s)

- Multiprogramming
  - More and more memory available – can load several jobs at once
  - OS (*monitor*) always resident to coordinate activities
  - OS manages interactions between concurrent jobs:
    - Runs programs until they block due to I/O
    - Decides which blocked jobs to resume when CPU freed
    - Protects each job’s memory from other jobs
      - (Virtual Memory)
  - Example: IBM/360 (c1964) combined jobs of IBM 1401’s and IBM 7094
    - First machine to use ICs (integrated circuit) instead of individual transistors

OS History – Phase 2 (1965-1980) Hardware is cheap(er), humans are expensive!

- Timesharing
  - First time share system: CTSS from MIT (1962)
  - Timer interrupts: enable OS to take control (preemptive multitasking)
  - MIT/Bell Labs/GE collaboration led to MULTICS
    - Envisioned one huge machine for all of Boston (!!!)
    - Started in 1963, "done" in 1969, dead shortly thereafter
    - Bell Labs bailed on project, GE bailed on computers!
  - DEC PDP minicomputers: start of bottom feeding frenzy
    - PDP 1 in 1961 (4K 18-bit words, $120,000)
    - Kernighan dubbed OS "UNICS" to poke fun at Ken Thompson
    - "C" language developed for Unix (ancestor was "B")
    - Guiding principle of UNI(X): Keep it simple so it can be built

OS History – Phase 2 (1965-1980)

- Timesharing (continued)
  - Terminals are cheap
    - Let all users interact with the system at once
    - Debugging gets a lot easier
    - Process switching occurs much more frequently
  - New OS services:
    - Shell to accept interactive commands
    - File system to store data persistently
    - Virtual memory to allow multiple programs to be resident
  - New problems: response time and thrashing
    - Need to limit number of simultaneous processes or you can fall off performance cliff

OS History – Phase 3 (1980-2000s)

- Personal computing: every "terminal" has computer
  - One user per machine, since machines are no longer scarce
  - Initial PC OSes similar to old batch systems (w/ TSR hacks)
  - Advanced OS features creep back in!

Original IBM PC

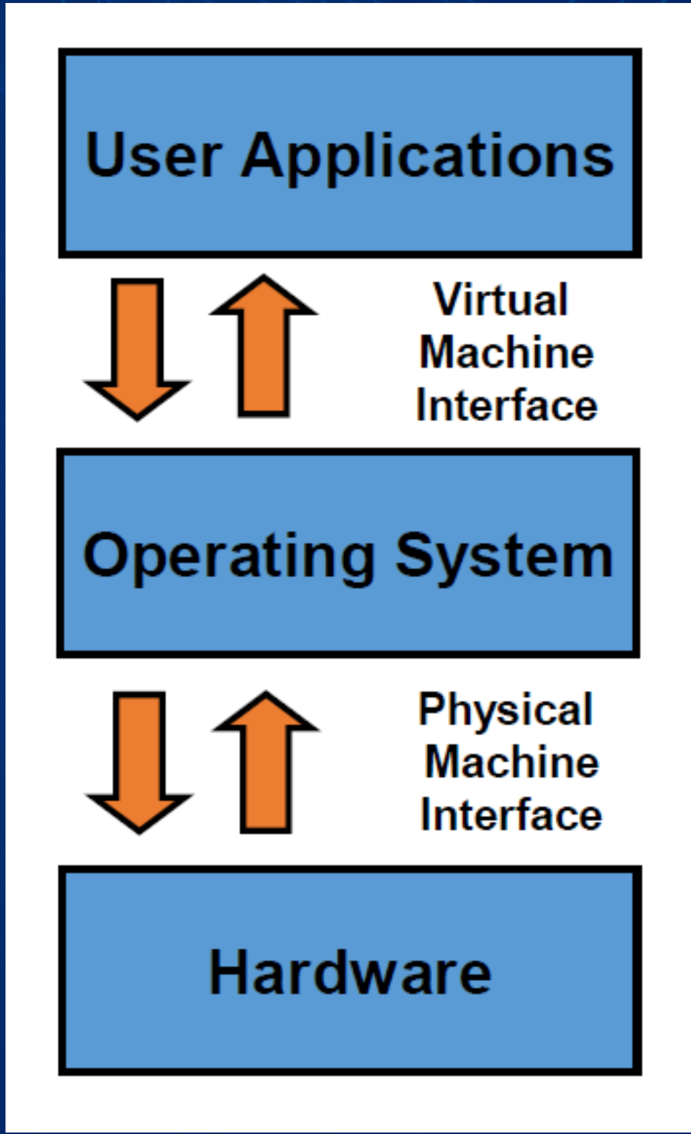
A young Bill Gates

OS History – Phase 4 (2000s - Now)

- Lots and lots of computers per person
  - Embedded systems
    - Cars commonly have 50+ processors
      - Airplanes, factories, power plants run a huge amount of software
      - Internet of (Insecure, Compromised) Things
      - “Why Software is Eating the World”
      - Some embedded systems run a “real OS,” others run an RTOS, still others run a program on the bare metal
  - Mobile computing
    - Smart phones and tablets
    - PCs and laptops are mostly power tools for content producers
  - Cloud computing
    - Buying and administering hardware sucks, maybe most of us should avoid doing that
    - Virtualized compute resources are flexibly allocated on demand
    - Computing as a service rather than a product

What is an Operating System?

- Interface between user and hardware
  - Exports simpler "virtual machine" interface
  - Hides ugly details of real hardware
- Manages shared resources
  - CPU, memory, I/O devices, …
  - Ensures fairness
  - Protects processes from one another
- Provides common services
  - File system, virtual memory, network, CPU scheduling, etc…
- Goals:
  - Convenient to use
  - Efficient
- OS versus kernel
  - Device drivers, process management, network and file management, memory, system calls



What will you learn about OSes?

- Concurrency* and concurrent programs**
  - *Threads within a single program (managed by the programmer)
  - ** Multiple different programs running at the same time (managed by the OS)
  - Very important these days
  - Very hard to get right
- Resource management
  - All big programs are resource managers
- Performance engineering
  - OS is performance critical and a lot of its complexity stems from this
- Security policy design and enforcement
  - Both at the hardware and software level
  - Very important
  - We all need to be good at it
  - Almost nobody is good at it

What will you learn about OSes?

- Enduring ideas
  - OS concepts are decades old, and they keep being relevant across incredible changes in hardware and applications
  - My job is to teach you how these ideas relate to current OSes and ongoing trends
- The OS is not magic
  - In fact, most of its parts are really simple
  - But there are a lot of parts
- Interface design
  - The OS hides the complicated low-level interfaces exported by the bare metal
- Tradeoffs
  - Many policy decisions strike a balance between different requirements
  - You can’t make everyone happy all the time

Common Theme: Coordination

- Concurrency – Processes and Threads
  - Allow several different tasks to be underway at the same time
  - The OS provides abstractions to make this work
  - Hard to keep processes from interacting in bad ways
- I/O Devices
  - Don’t want CPU (or user) to sit idle while an I/O device is working
  - OS starts operations in parallel on many devices
  - Interrupt notify on completion; lots of juggling
- Memory
  - How can a single memory be shared among several processes (safely, efficiently)?
  - How can we give the illusion that there’s more memory than we actually have?

The "3 Easy Pieces" (Textbook)

- Virtualization
  - Each program thinks it gets its own computer
- Concurrency
  - Each program is actually running on the same hardware…
- Persistence
  - Some programs want to be remembered after they're gone

To Do

- Lab
  - C Programming.
  - A good warm up.
  - Due at the end of the week.
- Readings – OSTEP
  - Chapters 1, 2, 3
  - Due next Monday

~ Fin ~