Operating Systems: Overview and Introduction

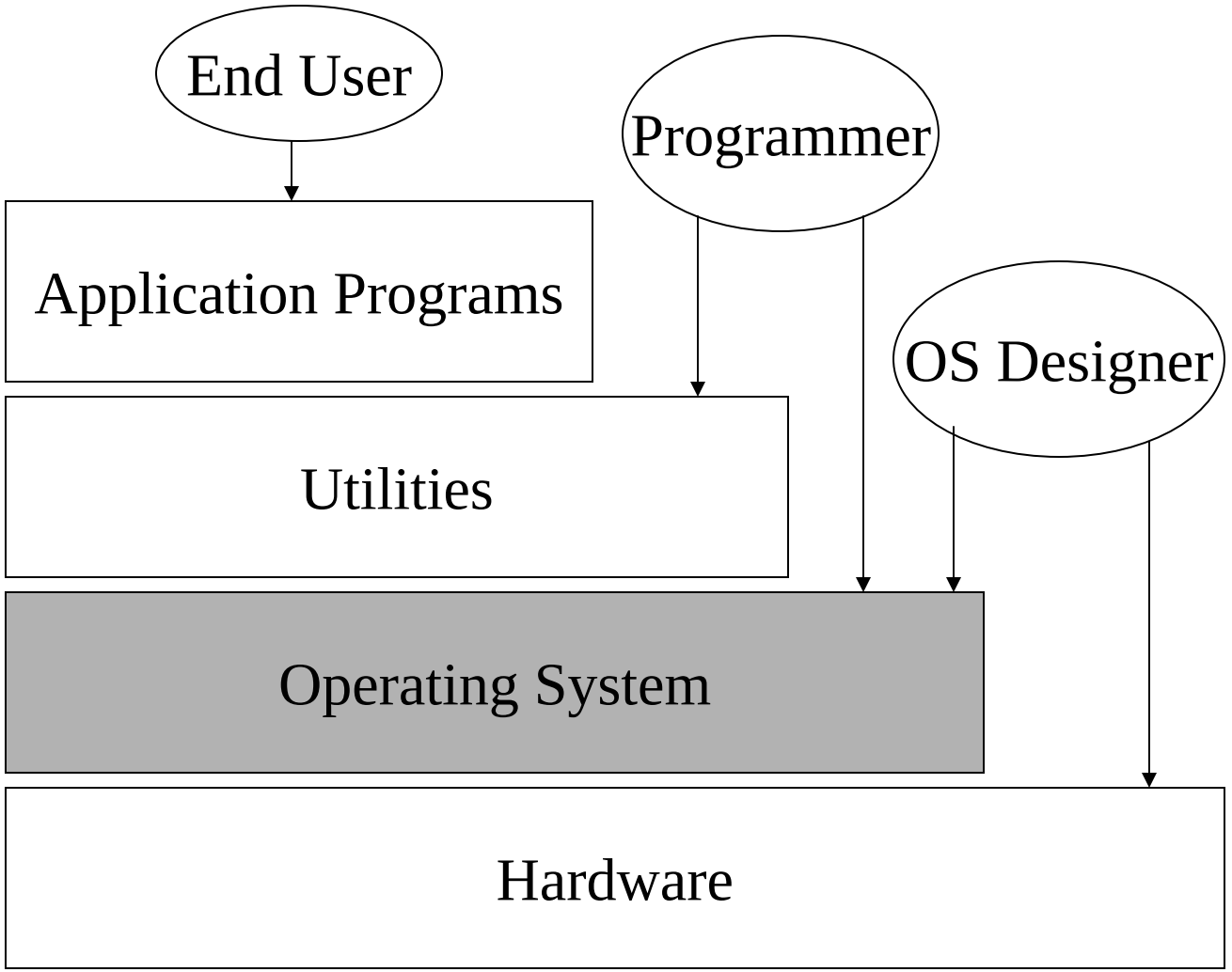
Announcements:

- Assignment is assigned
  - Easy, steep learning curve, fun!!
- Need to register for CCnet
- Midterm & final are CLOSED BOOK

What are the Objectives of an Operating System

- **Convenience** & **abstraction**
  - OS should facilitate the tasks of application and system programmer
  - Hardware details should be hidden
  - Uniform interface for system resources (e.g., I/O, files)
- **Efficiency**
  - Take up few resources, make good use of resources, and be fast
- **Protection**
  - Fairness, security, safety, resources

The Layers of Computer Systems



Services Provided By the OS I

- Program execution
  - Load instructions and data into memory
  - Initialize I/O devices
- Access to I/O devices
  - Provides a uniform interface to I/O devices (e.g., a file-like read/write interface)
- Controlled access to files
  - Provides file access control mechanisms

```
$ ls -l
total 102
-rwxr-xr-x  1 Arno Profs 104448 Jan 13 08:10 w2.ppt
```

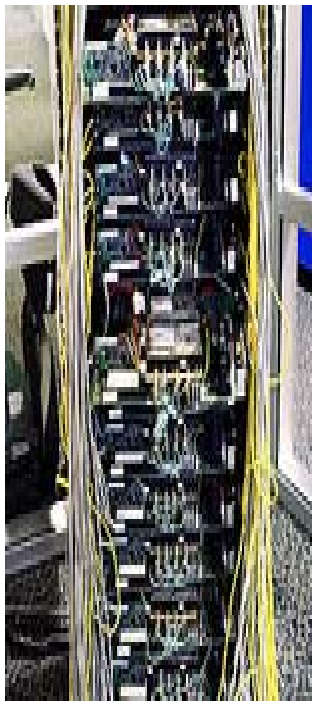
Services Provided By the OS II

- System access and protection
  - Data, processes
  - System resources
  - Resolves access conflicts
- Program development
  - Provides editors, debuggers, loader, linker (not part of core, but usually supplied together with OS and often intimately tied to OS.)

Services Provided By the OS II

- Error detection and response
  - Internal and external hardware errors (e.g., memory error, device failure)
  - Software errors (e.g., division by zero, arithmetic overflow, access of forbidden memory location)
  - OS cannot grant request of application
- The OS has to
  - Clear error conditions
  - Minimize effects on other applications

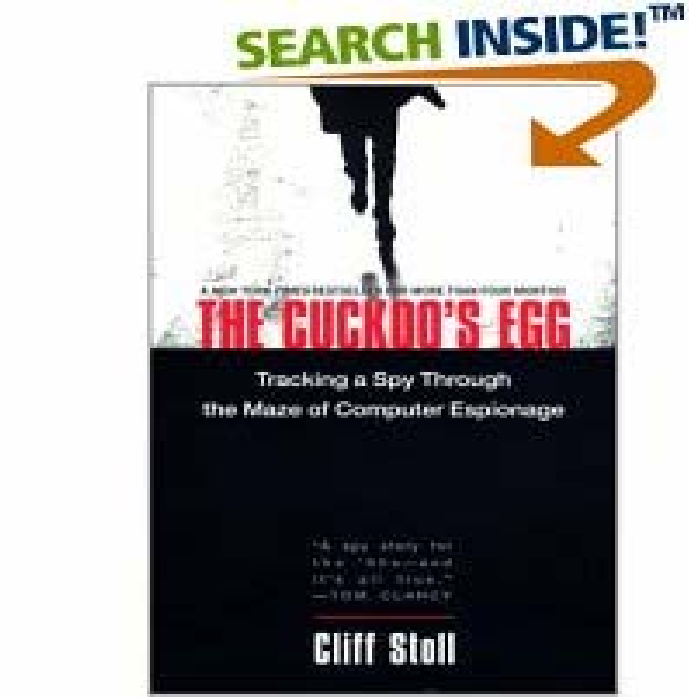
Designing for Errors



Services Provided By the OS

- Accounting
  - Collect statistics
  - Monitor performance
  - Anticipate future developments and enhancements
  - Billing of users

Due to a $US 0.75 accounting discrepancy



Operating System

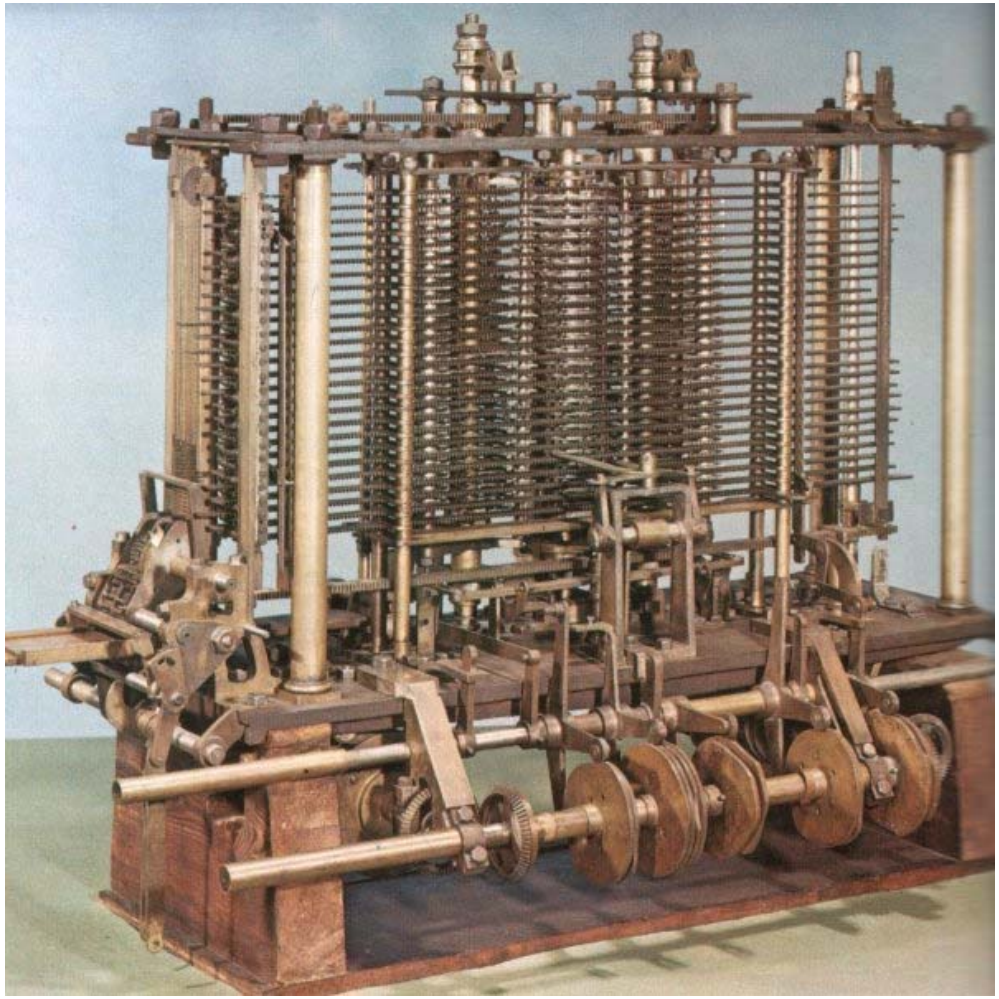
- The OS controls the
  - Movement, storage, and processing of data
- But it is not always “in control”
- Functions in same way as ordinary computer program
  - It is just a program (set of programs) in execution
  - Relinquishes control of processor to execute other programs
  - Must depend on the processor to regain control
- The OS is a resource manager

Kernel

- Portion of OS that is running in **privileged** (or “kernel”, “supervisory” mode)
- Usually resides in main memory (its pages are “pinned” to memory)
- Implements protection mechanisms
- Contains fundamental functionality to implement other services and functions
- A,k.a. the nucleus, supervisor, core, (resident) monitor

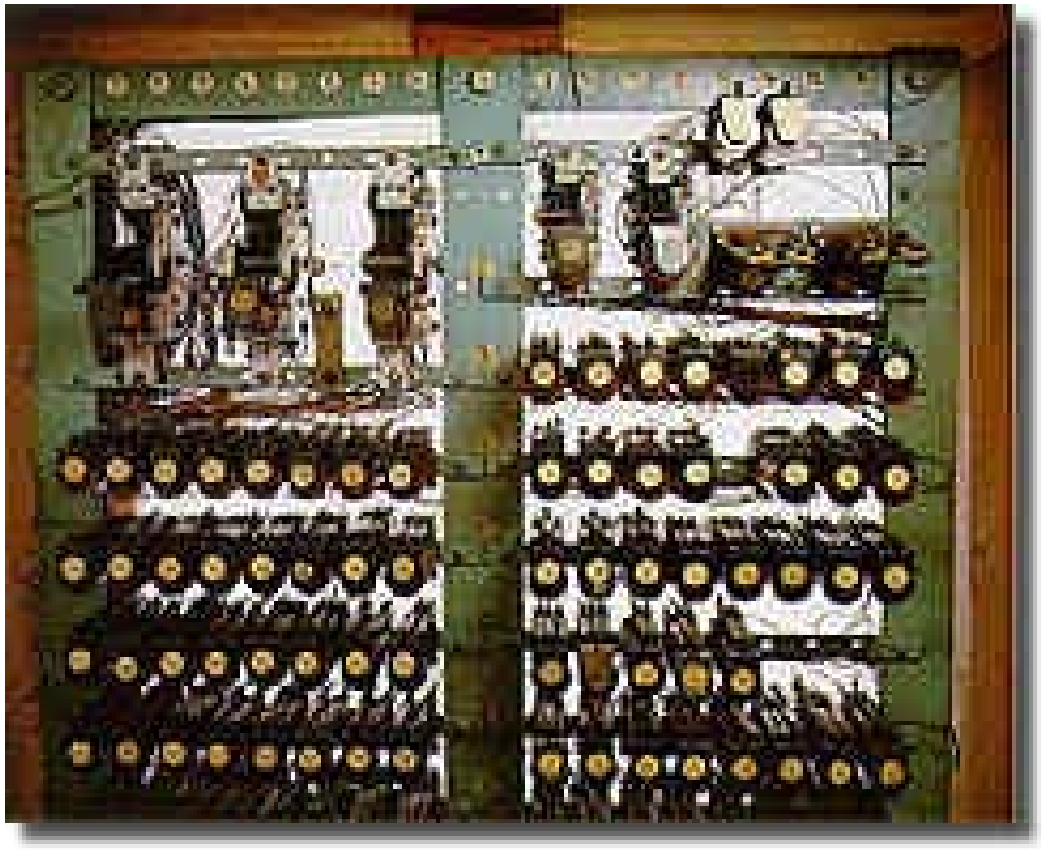
A Crude History of Operating Systems

Charles Babbage 1792-1871

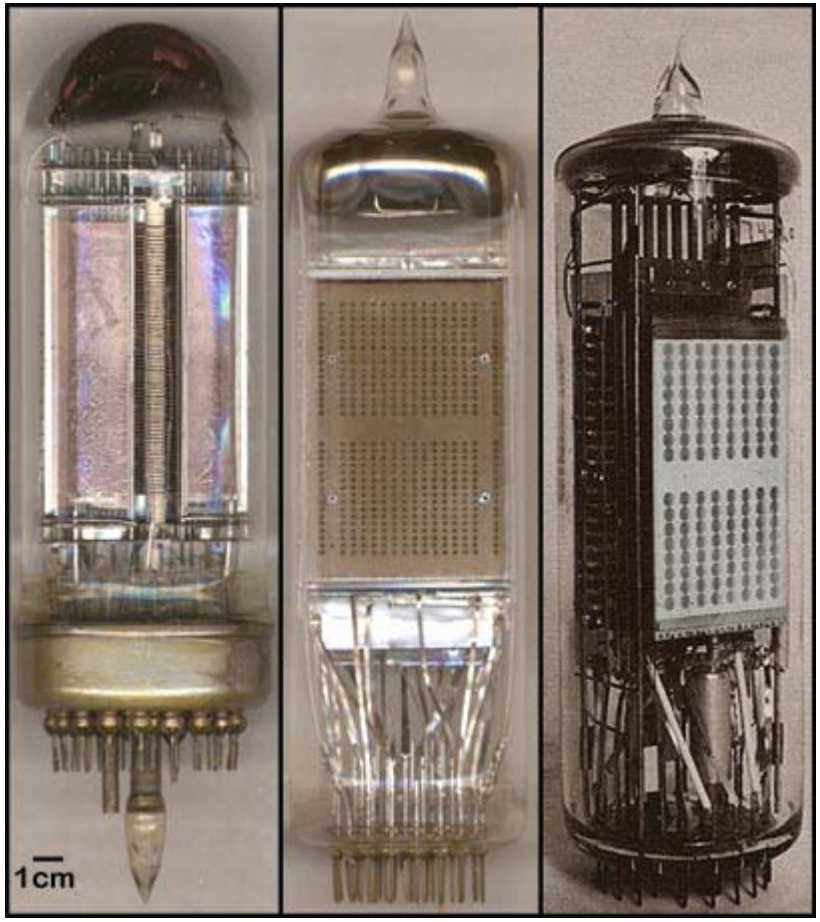


- Analytical engine
- No operating system
- Ada Lovelace hired as first programmer

First Generation 1945-1955
Zuse, Germany

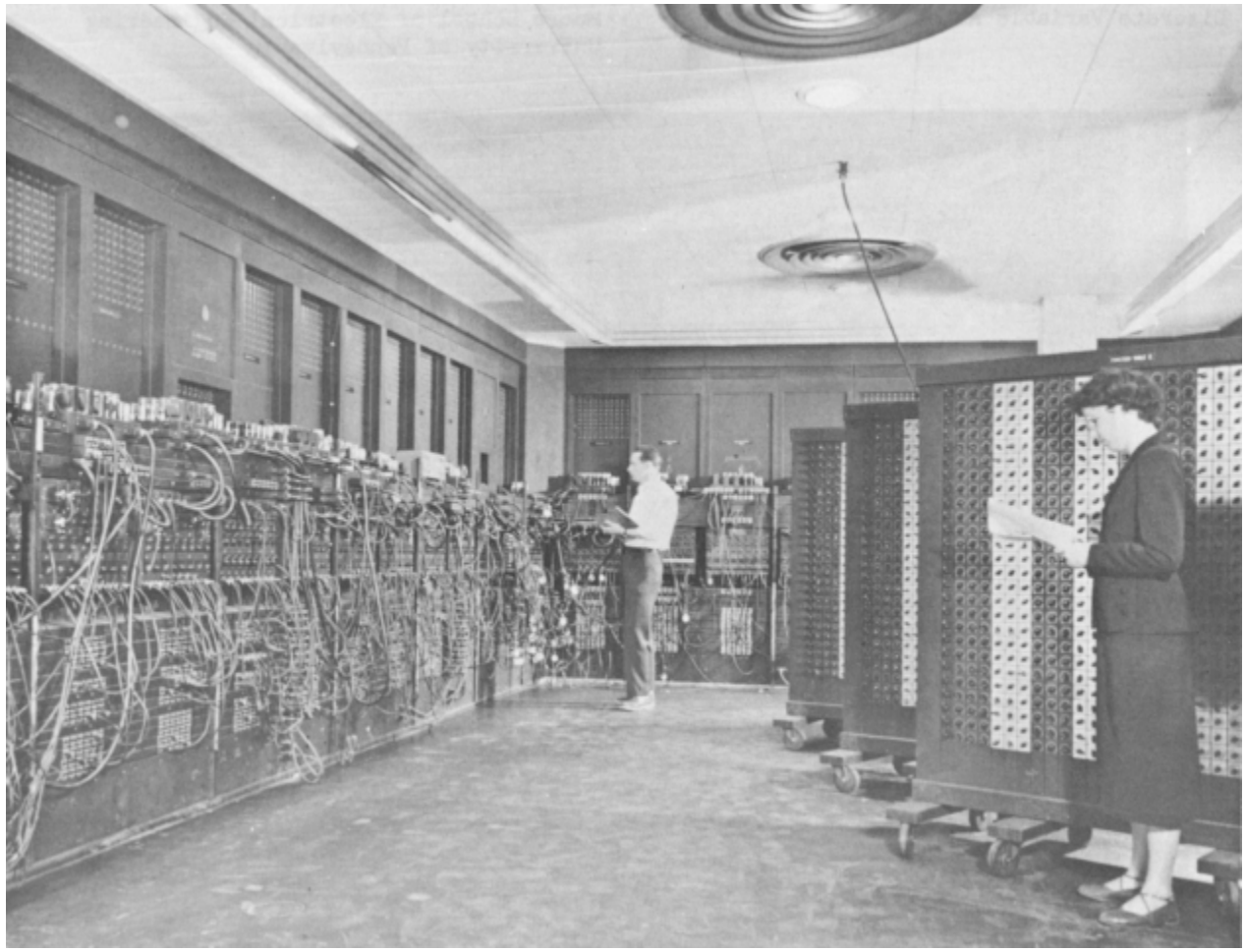


Vacuum Tube Memory 1940

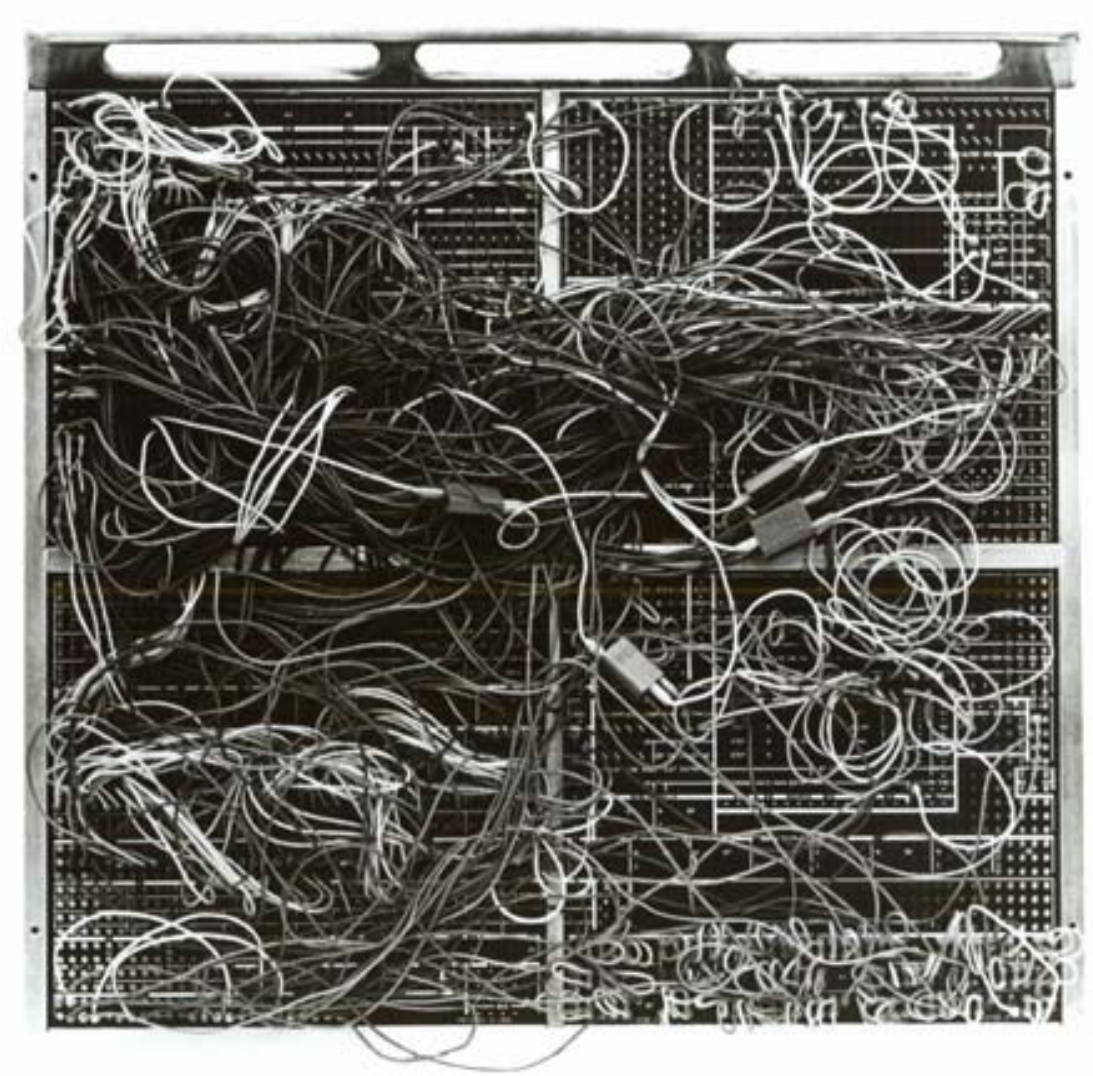


This electron tube developed at RCA Laboratories in Princeton, New Jersey, could **store 4096-bits** in one glass envelope. The Selectron was designed at the behest of **John von Neumann** of the Institute for Advanced Study and intended for use in their digital computer being designed during the 1940s.

First Generation 1945-1955
ENIAC-1, U.S.



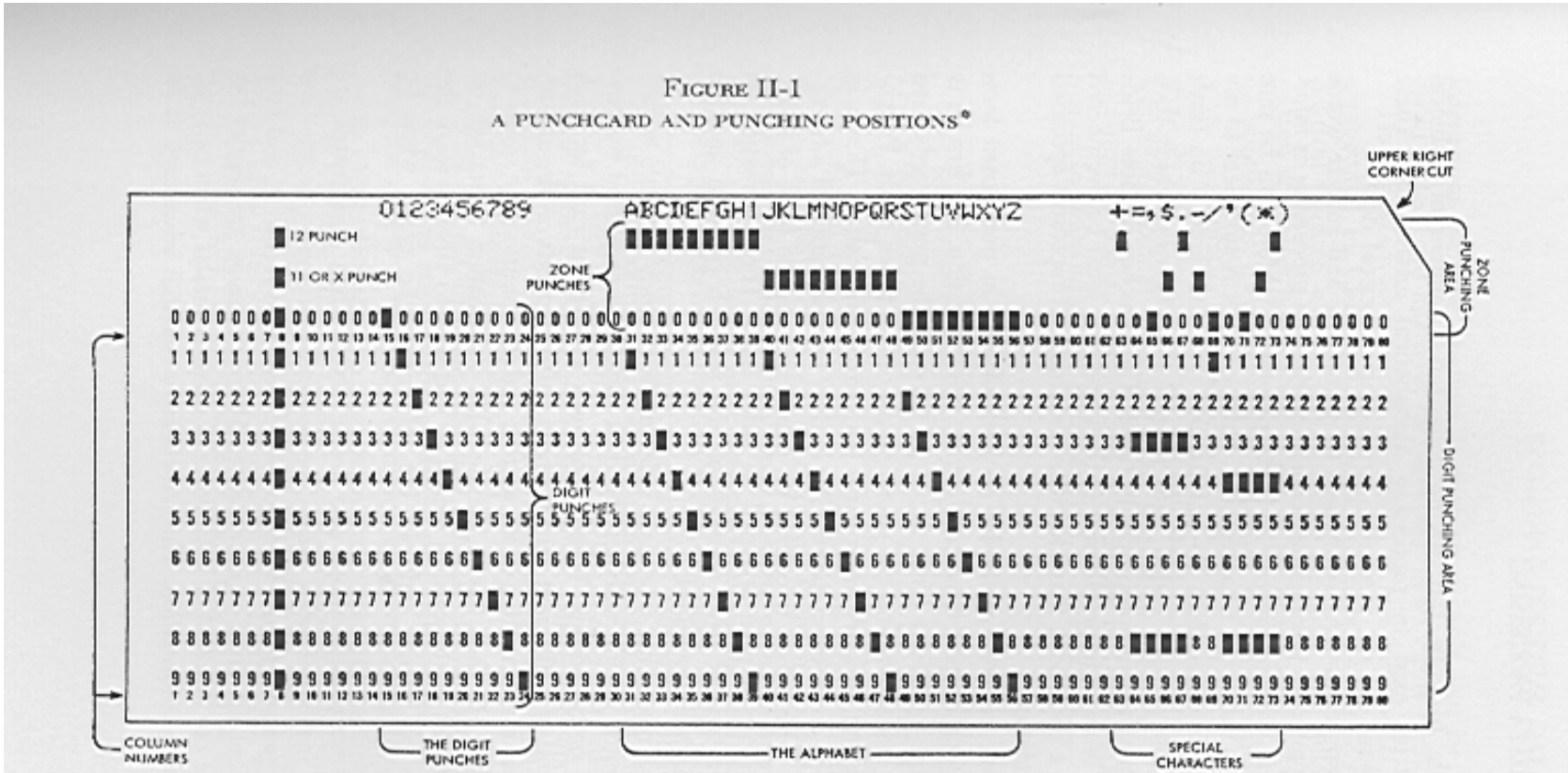
Plugboard Example (Age Unknown)



First Generation

- Computers build from vacuum tubes and plug boards
- No operating system
- Operator, administrator, programmer were all the same
- Machines about a million times slower than today's PCs (very crude estimate)
- Programmed in machine code (not assembly)
- Used to calculate tables for sinus, cosines, logarithms et al.

A Punchcard (introduced around 1950s)



http://www.murraystate.edu/polcrjlst/p660punchcard.htm

Second Generation 1955-1965
Transistors and Batch Systems

- Reliable manufacturing of computer due to **transistor technology**
- **Clear separation** between computer operators, programmers, maintenance personnel
- Introduction of mainframes
- Programmed through Fortran and **initially based on punchcards**
- Operating system was the FMS (**Fortran Monitoring System**)

ENIAC-10

Evolution of OS

- Computing setup for each user included
  - Loading compiler, source program
  - Saving compiled program
  - Loading and linking
  - Executing

Improvements: Libraries of common functions, loader, linker, compiler, debugger available to all users and **batching** of jobs.

Evolution of OS about 1950s-1960s

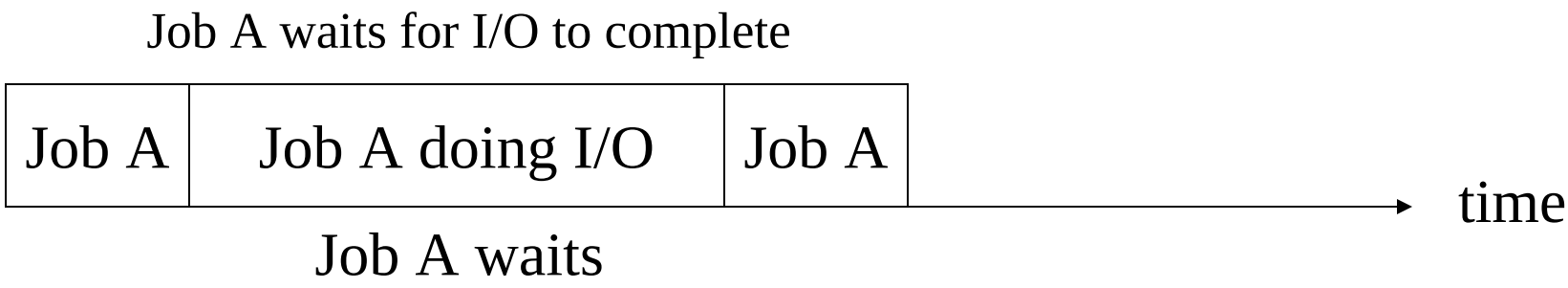
Simple Batch Systems: (e.g., by GM for IBM 701)

- The **monitor** controls the execution of all program
  - **Batches** jobs together
  - Program branches back to monitor when finished
  - **Resident monitor** is in main memory and is ready for execution
- Instructions to monitor via **Job Control Language**
  - Monitor contains a JCL interpreter
  - Each Job includes instructions in JCL that tell the monitor what compiler to use and what data to use
  - Predecessor of today's shell
- Monitor takes up compute time and memory but **improves overall utilization of computer**

Predominant Model at the time: Uniprogramming

Problem:

- CPU must wait for I/O instructions to complete before proceeding
- I/O instructions are very slow as compared to compute instructions

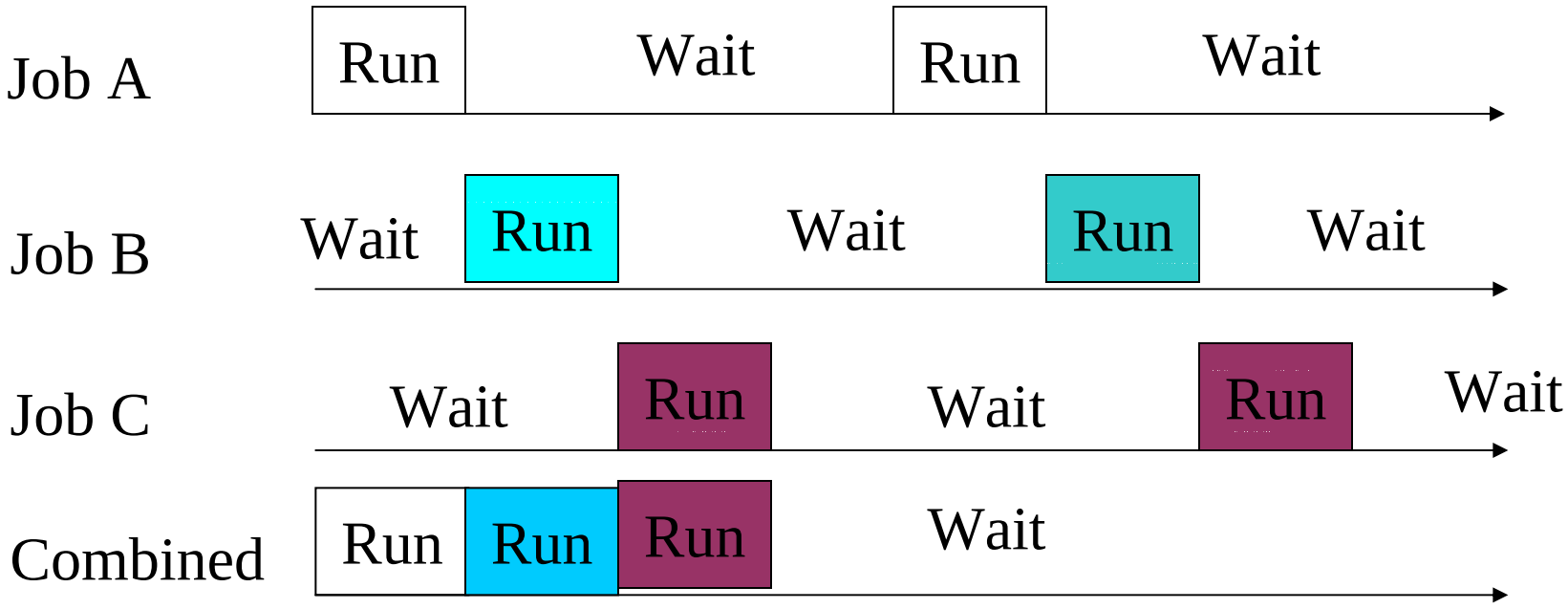


Terminology: processes = jobs = tasks

Solution: Multiprogramming

When one job waits for I/O, the processor can switch to another job (interleave I/O and computing.)

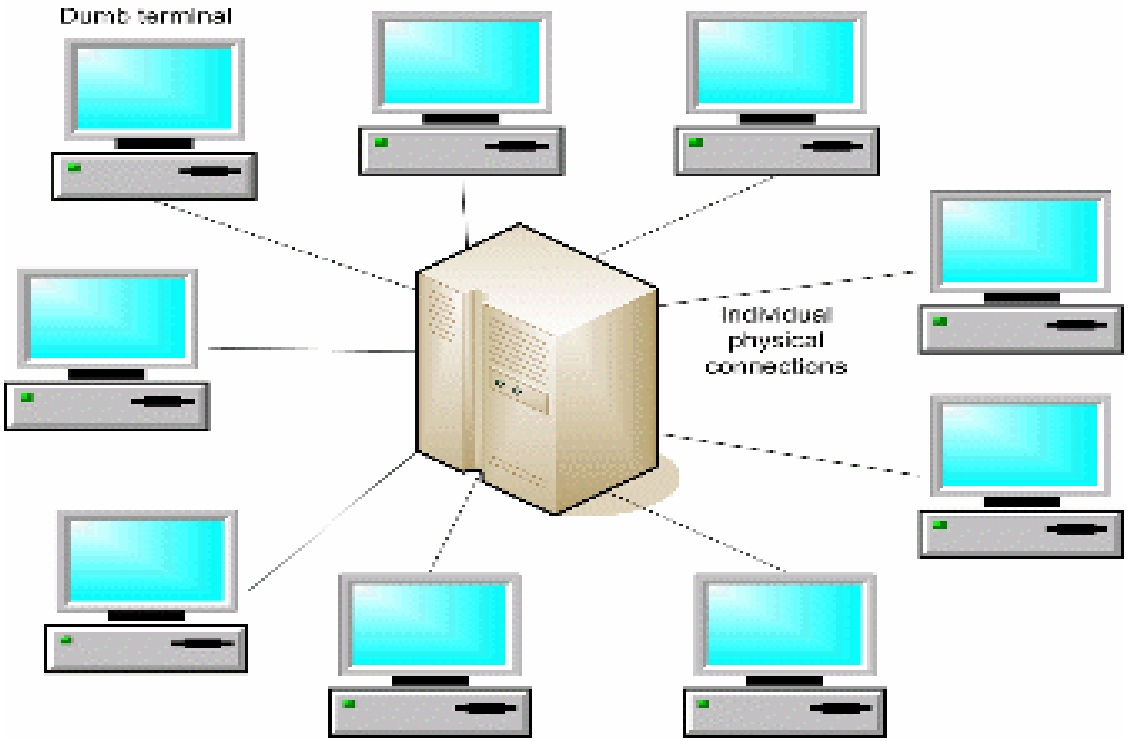
- Increased throughput
- Increased system utilization



Concurrency vs. parallelism

- Multiprogramming leads to the **interleaved execution** of multiple processes
- True **parallelism** is only possible if we have multiple CPU to execute processes in parallel
- Otherwise (single CPU) we say processes execute **concurrently**
  - Apparent, simultaneous execution of multiple different processes in an interleaved fashion

Time Sharing



Time Sharing

- Batch multiprogramming improves the utilization of the CPU through **batch** jobs, but what about **interactive** jobs?
- Batch jobs and interactive jobs have different characteristics
- CPU's time is shared among multiple users
- Multiple users simultaneously access the system through multiple terminals
- Referred to as **time shared system**, i.e., each terminal / users **obtains a share of the compute time**

Third Generation 1965-1980
ICs and Multiprogramming

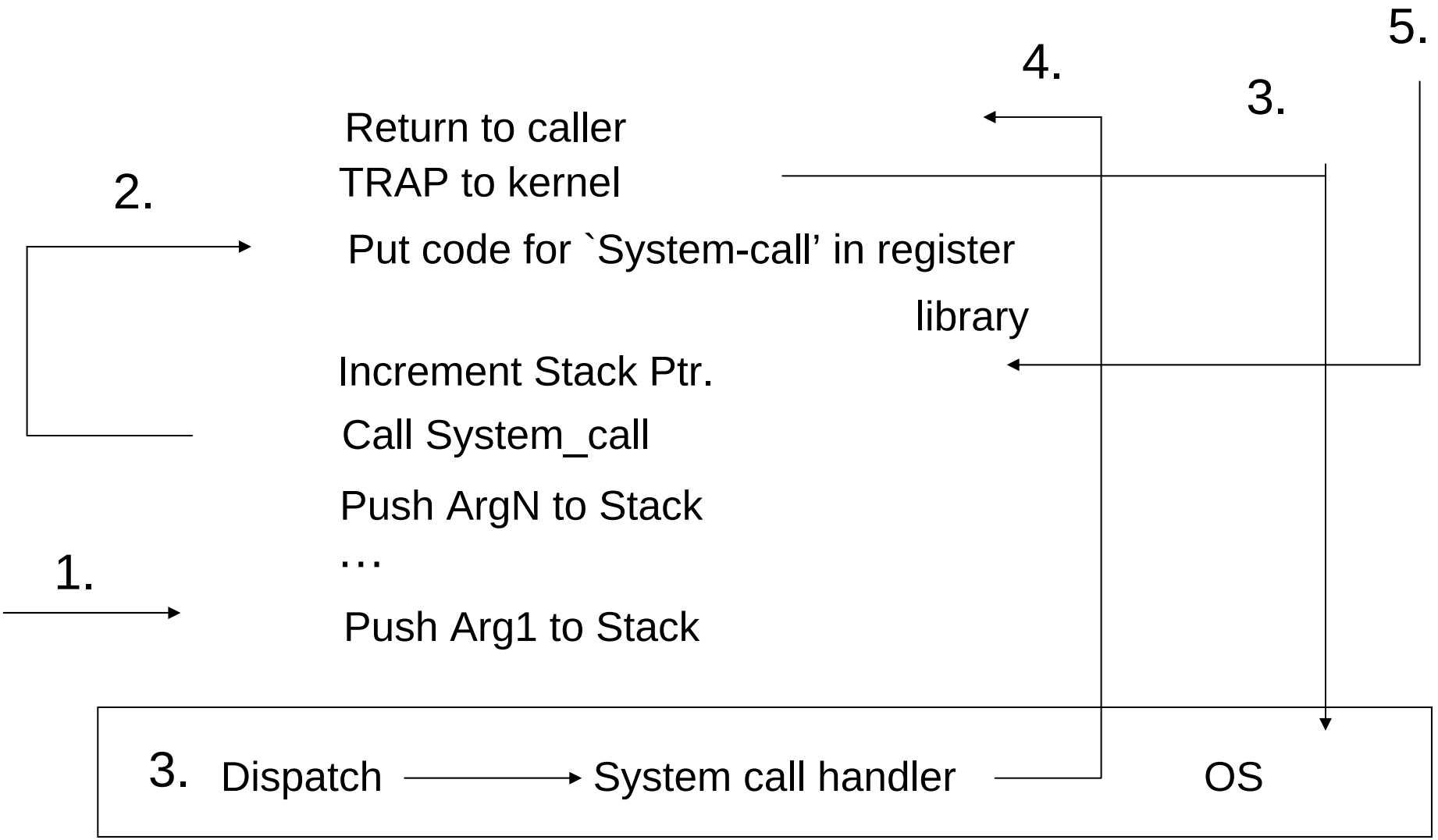
- Multiprogramming and time-sharing are part of the third generation
- MULTICS as first time-shared OS developed at M.I.T.

System Calls

- System calls provide the interface between a running program and the operating system.
  - In the past implemented in assembly-language
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)
- System calls are handled like interrupts

A System Call

System_call(*Arg1, …, ArgN*)



Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

Summary

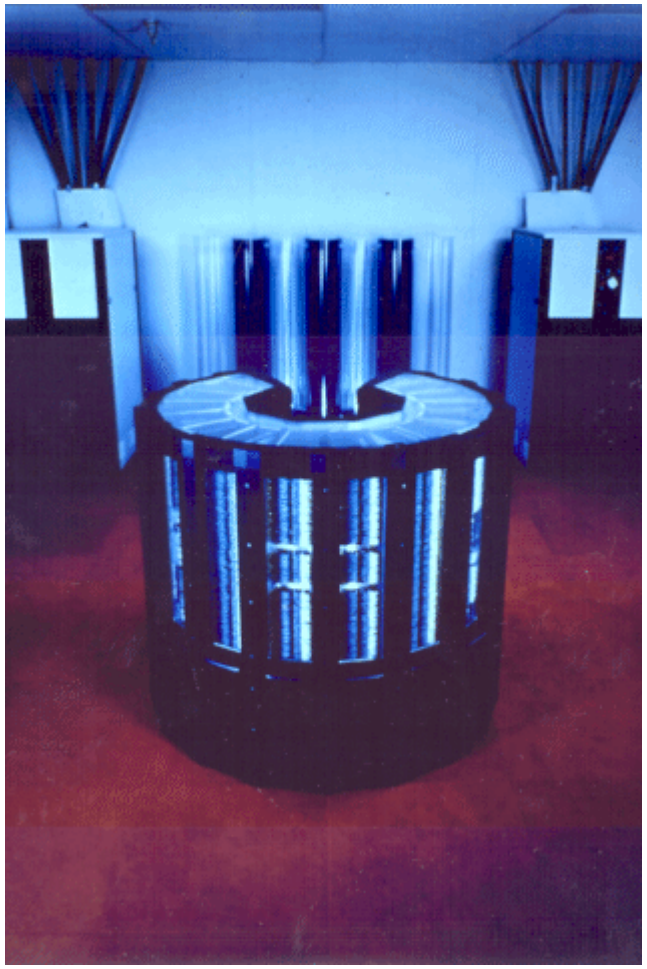
- Services of OS
- Protection mechanisms
- Batch processing
- Uniprogramming vs. multiprogramming
- Time sharing
- System Calls

Hardware Features

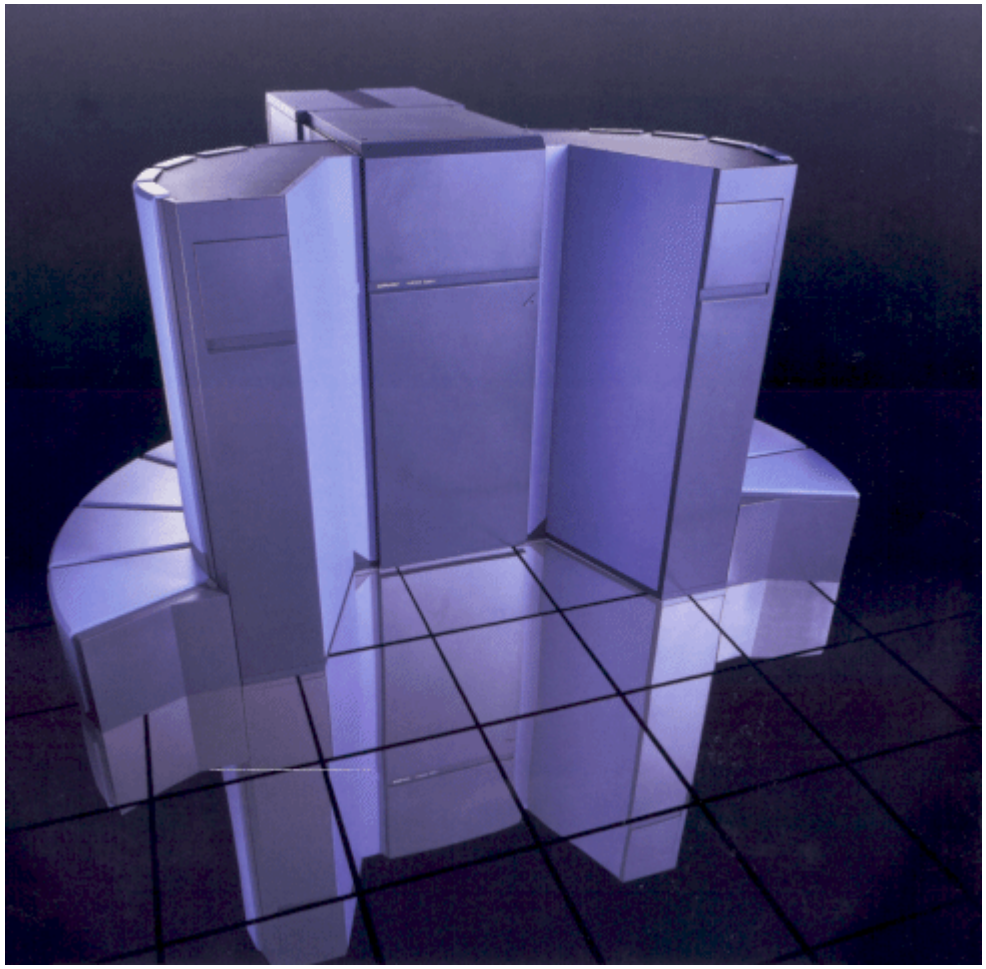
New hardware features support development of OS features:

- Memory protection
  - Do not allow memory area containing monitor to be altered. Protect individual jobs.
- Timer
  - Prevent jobs from monopolizing the system.
- Privileged instructions
  - For example, I/O instructions
- Interrupts
  - Relinquish control and gain control over user programs.

Cray



More like Today



Batch Multiprogramming vs. Time Sharing

	Batch Multiprogramming	**Time Sharing**
Objective	maximize CPU utilization	minimize response time
Control	JCL with Jobs	interactive commands

One of the first systems: Compatible Time Sharing System (CTSS), 1961,

IBM 709 and IBM 7094.

Multics at MIT

- System clock emits interrupts in regular intervals
- System switches to new user
- Old user's program and data saved to disk