Operating System

Chapter 2. OS Overview

Lynn Choi

School of Electrical Engineering

Class Information

- *Lecturer*
  - Prof. Lynn Choi, School of Electrical Eng.
  - Phone: 3290-3249, 공학관 411, *lchoi@korea.ac.kr*,
  - TA: 김정현, 3290-3896, *kilingki@korea.ac.kr*
- *Time*
  - Mon/Wed 2pm – 3:15am
  - Office Hour: Wed 1:00pm – 2:00pm
- *Place*
  - 공학관 250
- *Textbook*
  - "*Operating Systems: Internals and Design Principles*", William Stallings, Pearson, 8th Edition, 2015.
- *References*
  - "*Computer Systems: A Programmer's Perspective*", Randal E. Bryant and David O'Hallaron, Prentice Hall, 3rd Edition, 2016.
- *Class homepage*
  - http://it.korea.ac.kr : slides, announcements

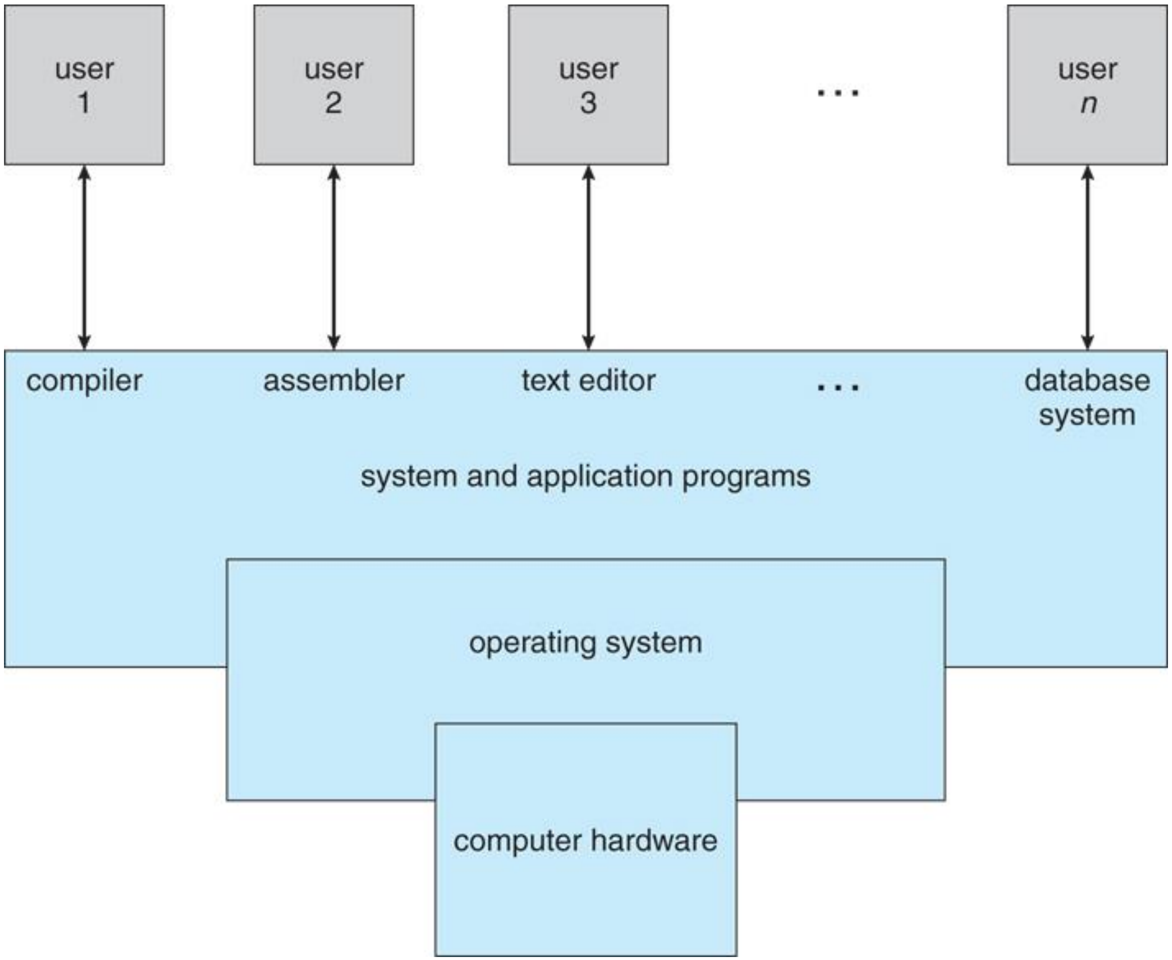
Class Information

- *Course overview*
  - 1. OS Overview
  - 2. Process
  - 3. Thread
  - 4. Mutual Exclusion and Synchronization
  - 5. Deadlock and Starvation
  - 6. Memory Management
  - 7. Virtual Memory
  - 8. Uniprocessor Scheduling
  - 9. Multiprocessor and Realtime Scheduling
  - 10. IO
  - 11. File Management
  - 12. Virtual Machine

Class Information

- *Evaluation*
  - Midterm : 35%
  - Final: 35%
  - Homework and Projects: 30%
  - Class participation: extra 5%
    - Attendance: no shows of more than 2 will get −5%
    - Bonus points

Abstract view of a computer system

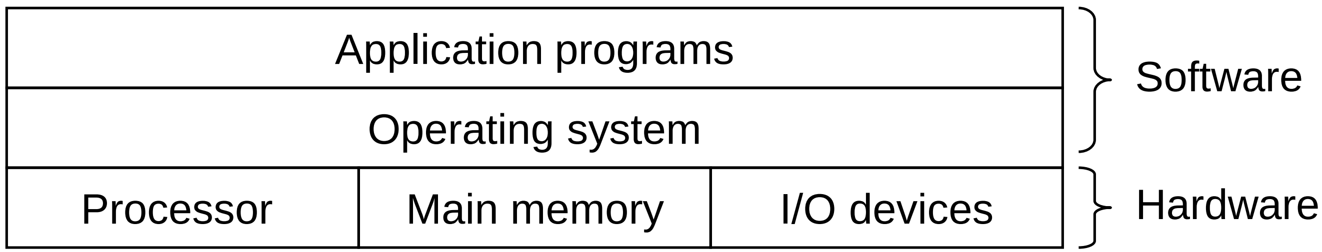


Operating system controls the hardware and coordinates its use among various application programs for various users

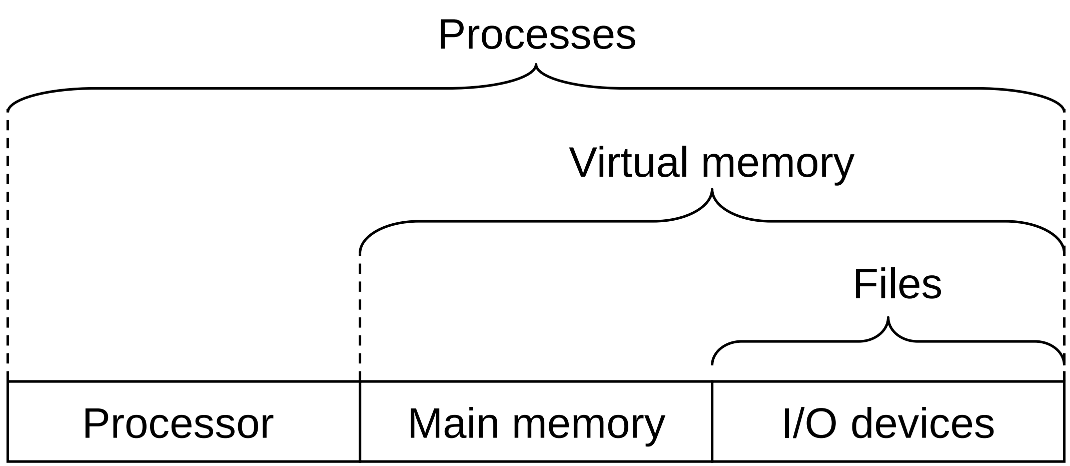
OS

- *Operating system*
  - A layer of software between application programs and the hardware
  - Three purposes
    - **Easy of use**: Provide applications with simple and uniform interface to manipulate complicated and often widely different low-level hardware devices
    - **Sharing**: Share the hardware resource from multiple processes/users and increase the **resource utilization**
    - **Protection**: Protect the hardware from misuse by runaway applications
      - Incorrect or malicious programs should not cause other programs to execute incorrectly
      - Smartphone and PC operating systems emphasize easy of use while server and mainframe OSs emphasize sharing, protection, and fairness
  - Commercial operating systems usually provide not only **kernels** (*programs running at all times on the computer*) but also system **middleware**, **system programs**, and application programs.
  - Use abstractions such as **processes**, **virtual memory**, and **files**

OS



Layered view of a computer system



Abstractions provided by an OS

Operating System

- *An OS is a program that provides convenient services for application programs by managing hardware resources and acts as an interface between applications and the computer hardware*

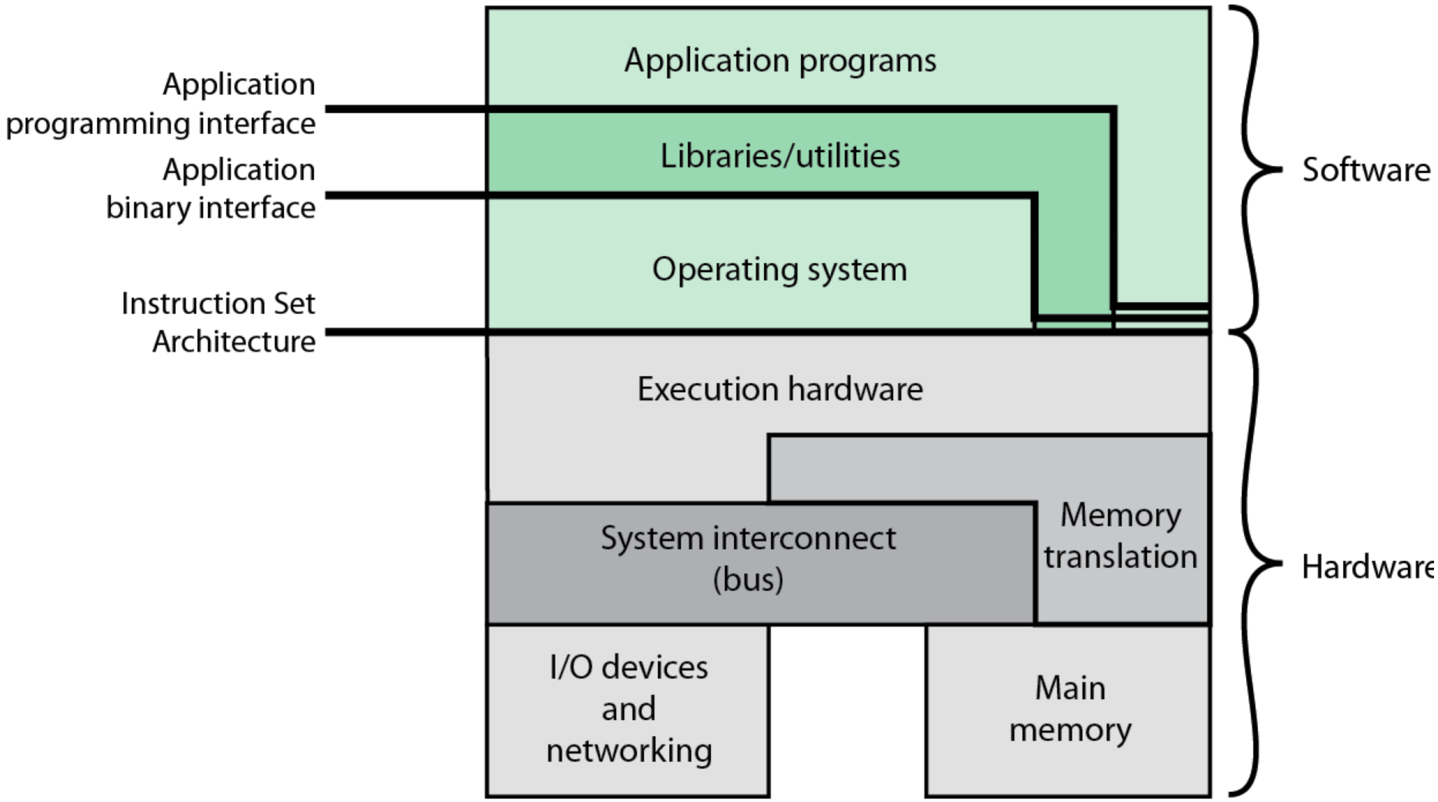


Figure 2.1 Computer Hardware and Software Infrastructure

Source: Pearson

Key Interfaces

- *Instruction set architecture (ISA)*
  - Define the interface between SW and HW
    - A set of machine language instructions
    - Both application programs and utilities can access the ISA directly
- *Application binary interface (ABI)*
  - Define the system call interface to OS
- *Application programming interface (API)*
  - Define the program call interface to system services. System calls are performed through library calls.
    - Enables application programs to be ported easily, through recompilation, to other systems that support the same API

Terminology

- **Microprocessor: a single chip processor**
  - Intel i7, Pentium IV, AMD Ryzen, SPARC T5, ..
- ***ISA (Instruction Set Architecture)***
  - Defines machine instructions and visible machine states (registers + memory)
  - Examples
    - IA32(x86), IA64, ARM, MIPS, SPARC, PowerPC
      - x86 family: Intel Pentium, Pentium Pro, Pentium 4, AMD Athlon
      - x86-64 family, Intel i3, i5, i7, i9
      - IA64 family: Itanium, Itanium2, Itanium 9300/9500/9700
      - ARM family: ARM7, ARM11, ARM Cortex-A53
- ***Microarchitecture***
  - Implementation: HW design and implementation according to the ISA
    - Pipelining, caches, branch prediction, buffers, out-of-order execution
    - 80386, 80486, Pentium, Pentium Pro, Pentium 4 are the 1st, 2nd, 3rd, 4th, 5th implementation (microarchitecture) of x86 ISA
  - Invisible to programmers
    - From the programmer's perspective, both 80486 and Pentium 4 support the same ISA and show the same behavior for a machine instruction

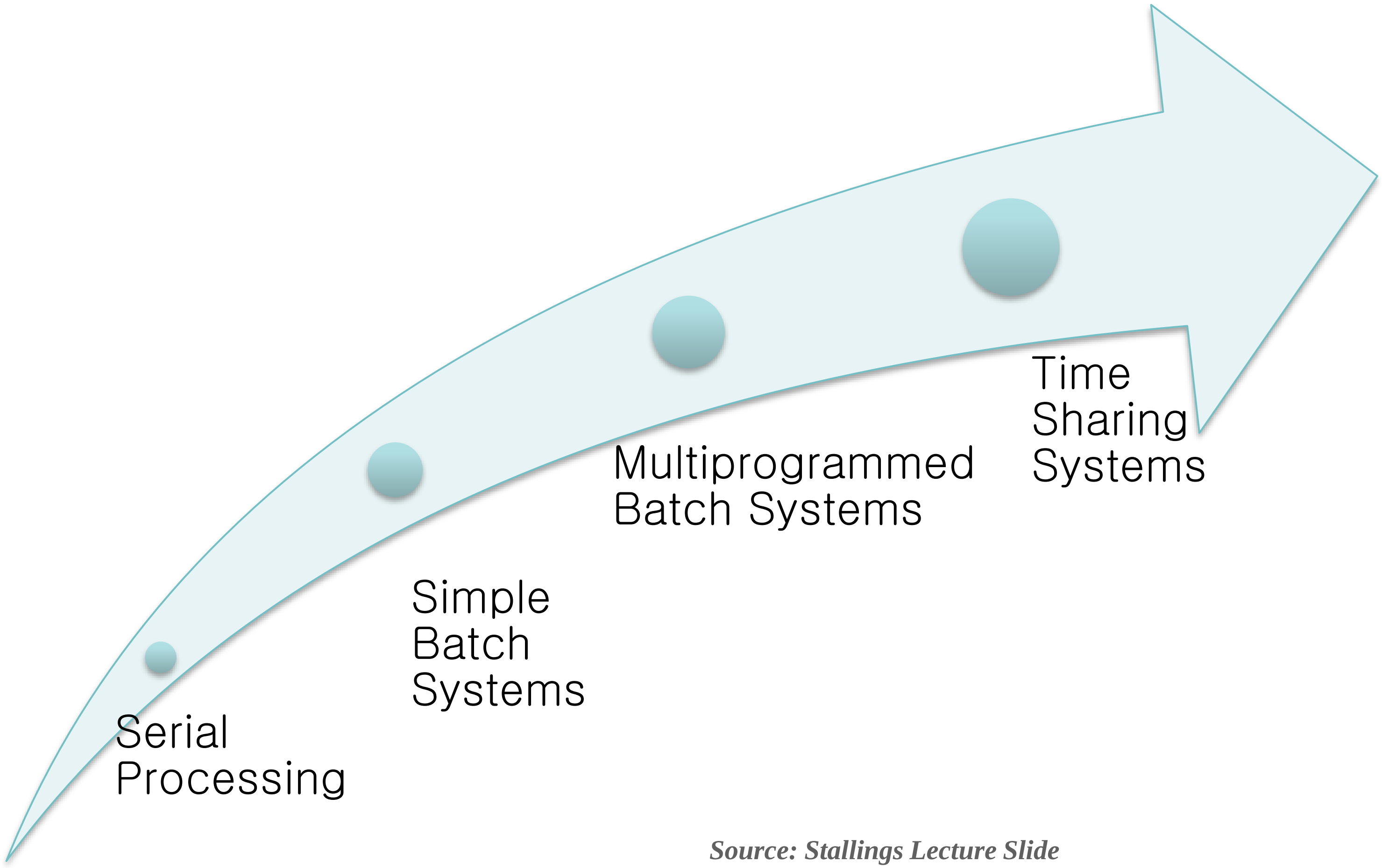
Terminology

- ***CISC (Complex Instruction Set Computer)***
  - Each instruction is complex
    - Instructions of different sizes, many instruction formats, allow computations on memory data, …
  - A large number of instructions in ISA
  - Architectures until mid 80's
    - Examples: x86, VAX
- ***RISC (Reduced Instruction Set Computer)***
  - Each instruction is simple
    - Fixed size instructions, only a few instruction formats
  - A small number of instructions in ISA
  - Load-store architectures
    - Computations are allowed only on registers
      - Data must be transferred to registers before computation
  - Most architectures built since 80's
    - Examples: MIPS, ARM, PowerPC, Alpha, SPARC, IA64, PA-RISC, etc.

Terminology

- ***Word***
  - Default data size for computation
    - Size of a GPR & ALU data path depends on the word size
      - GPR stands for general purpose (integer) registers
      - ALU stands for arithmetic and logic unit
  - The word size determines if a processor is a 8b, 16b, 32b, or 64b processor
- ***Address (or pointer)***
  - Points to a location in memory
  - Each address points to a byte (*byte addressable*)
    - If you have a 32b address, you can address 2^{32} bytes = 4GB
    - If you have a 256MB memory, you need at least 28 bit address since 2^{28} = 256MB
- ***Caches***
  - Faster but smaller memory close to processor
    - Fast since they are built using SRAMs
    - Smaller since they are expensive
    - Usually inside CPU chip (located between processor core and main memory)

Evolution of Operating Systems

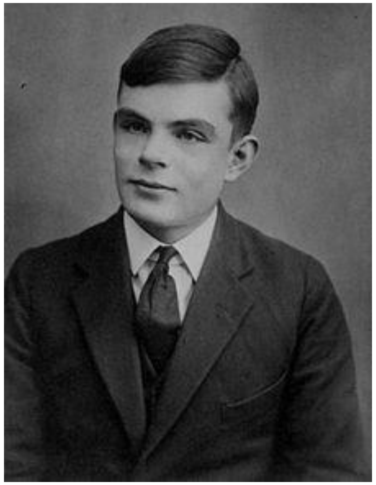


Source: Stallings Lecture Slide

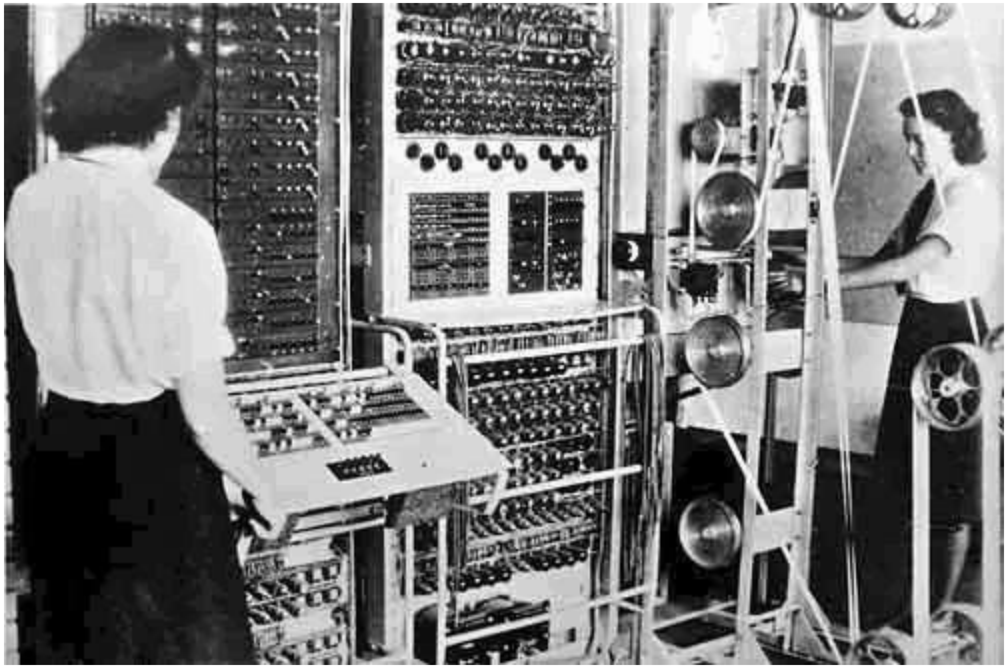
Serial Processing

- *Earliest computers*
  - No operating system until mid 1950s
    - Programmers interacted directly with the computer hardware
  - Computers ran from a console with display lights, toggle switches, jumper cables, some form of input device, and a printer
- *Problems*
  - Scheduling
    - Most installations used a *hardcopy sign-up sheet to reserve computer time*. However, time allocations could *run short or long*, resulting in *wasted time*
  - Setup time
    - A considerable amount of time was spent just on setting up the program to run. Compile/link/load require mounting tapes, setting up card decks, etc.
  - Early computers were very expensive
    - Important to maximize processor utilization

Alan Turing, Bombe and Colossus



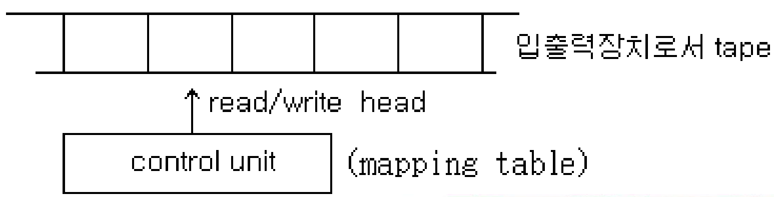
Bletchley Park

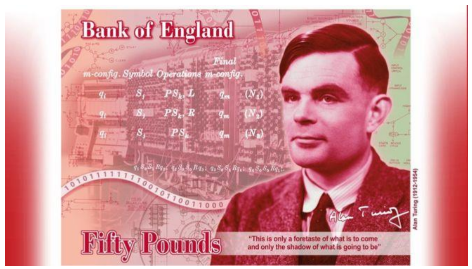


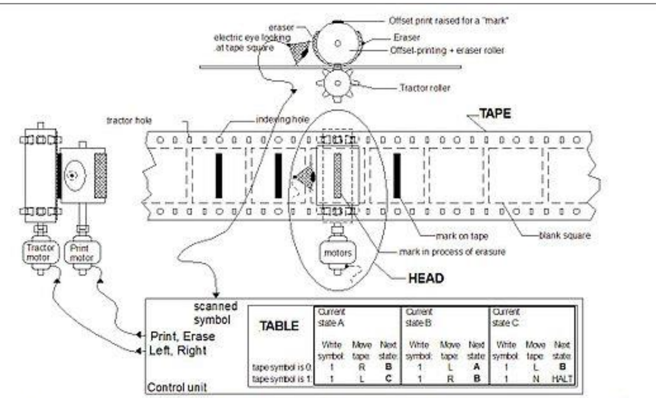
Source: Wikipedia

Alan Turing (1912 ~ 1954)

- ## *Turing Machine (1936)*
  - A new mathematical computing model
  - A symbol manipulating device that can simulate the logic of any computer
    - Consists of infinite linear tape, read/write head, and control unit
  - Theoretical background to modern computers





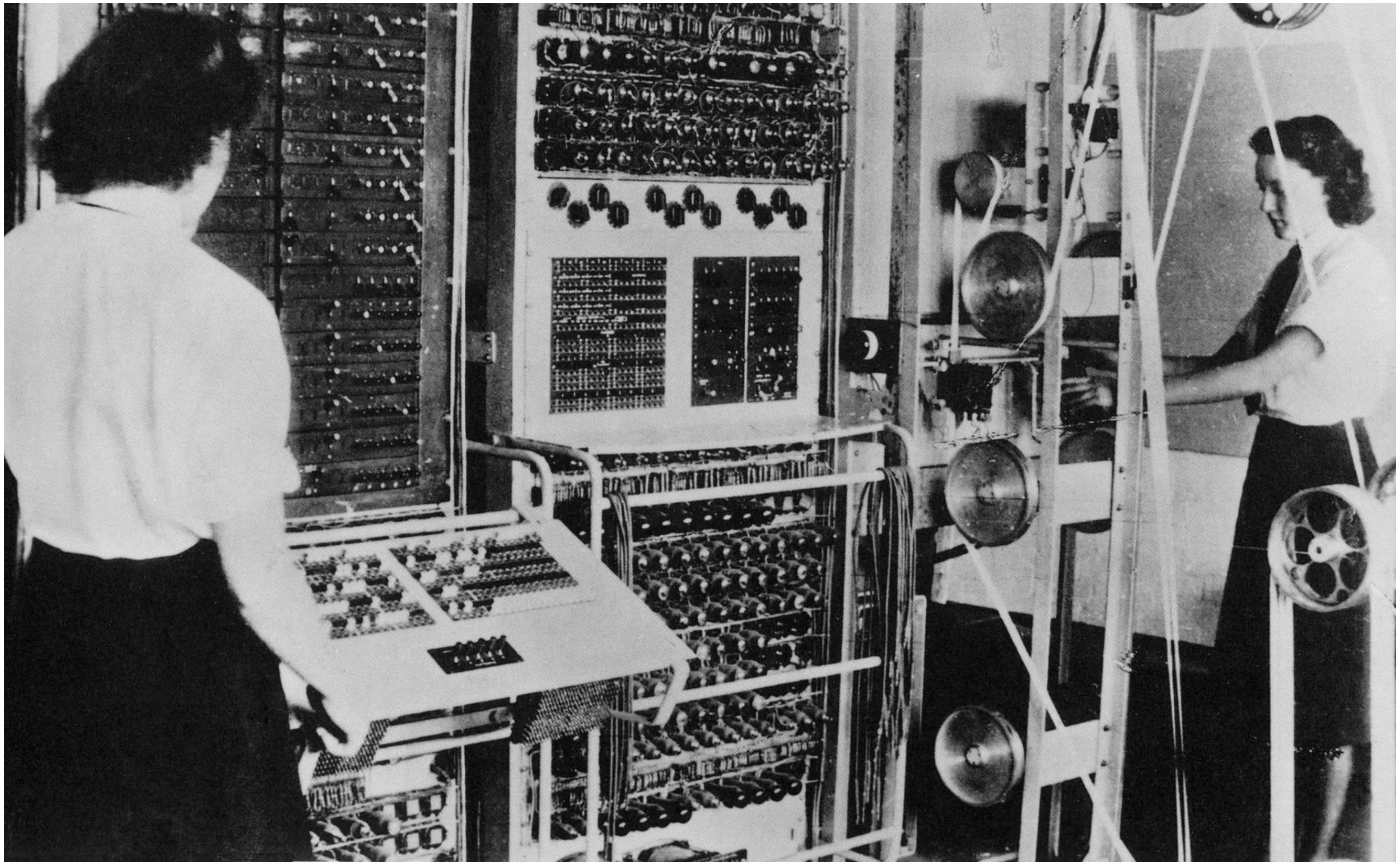


A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

- ## *Bombe (1940)*
  - A electromechanical machine to decipher German's Enigma
- ## *Turing Test (1950)*
  - A computer could be said to "think" if a human interrogator could not tell it apart, through conversation, from a human being
- ## *ACM Turing Award – the Novel prize in Computer Science*

Colossus

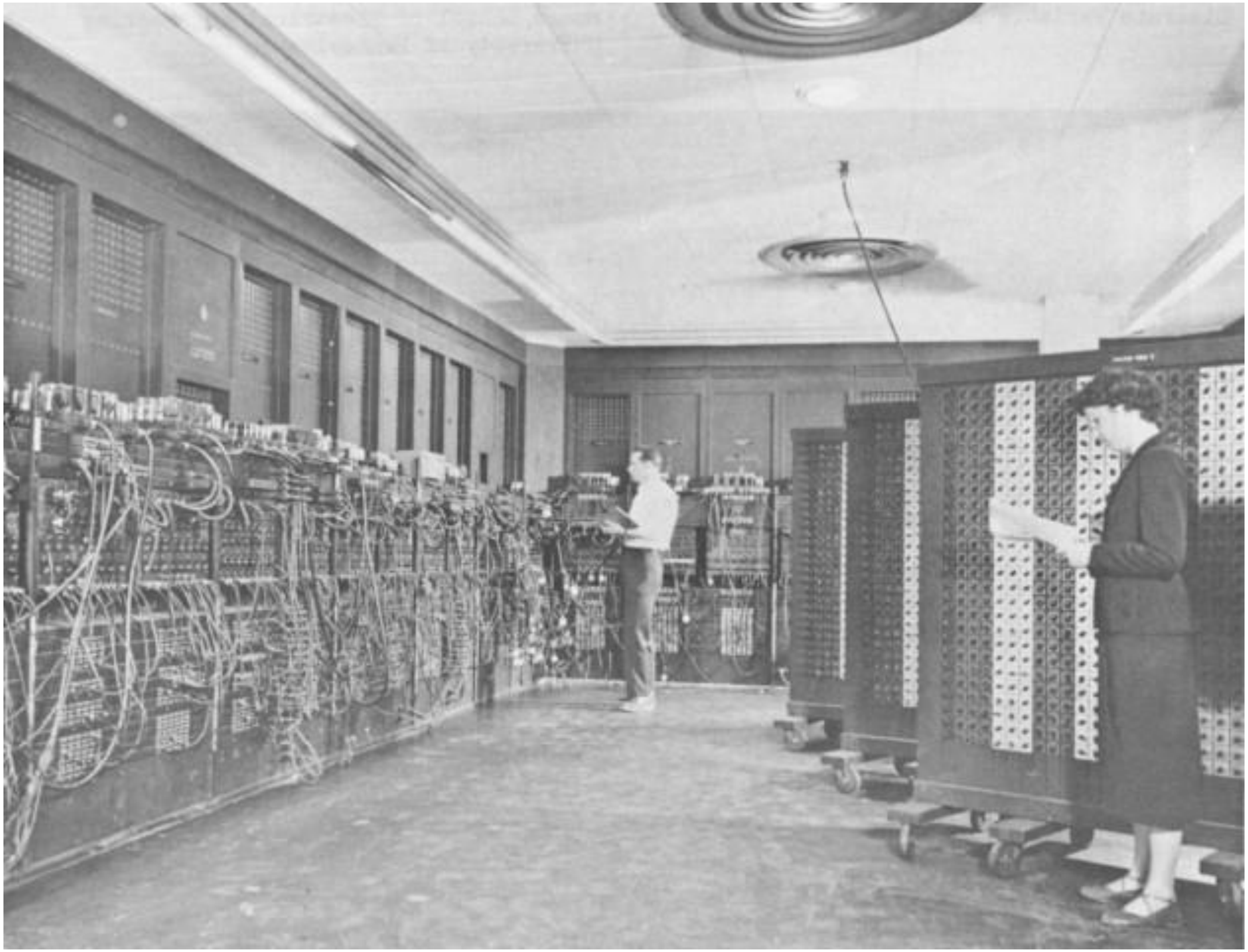
- *Colossus (1943)*
  - The 1st programmable digital (electronic) computer built by Thomas Flowers in London



ENIAC

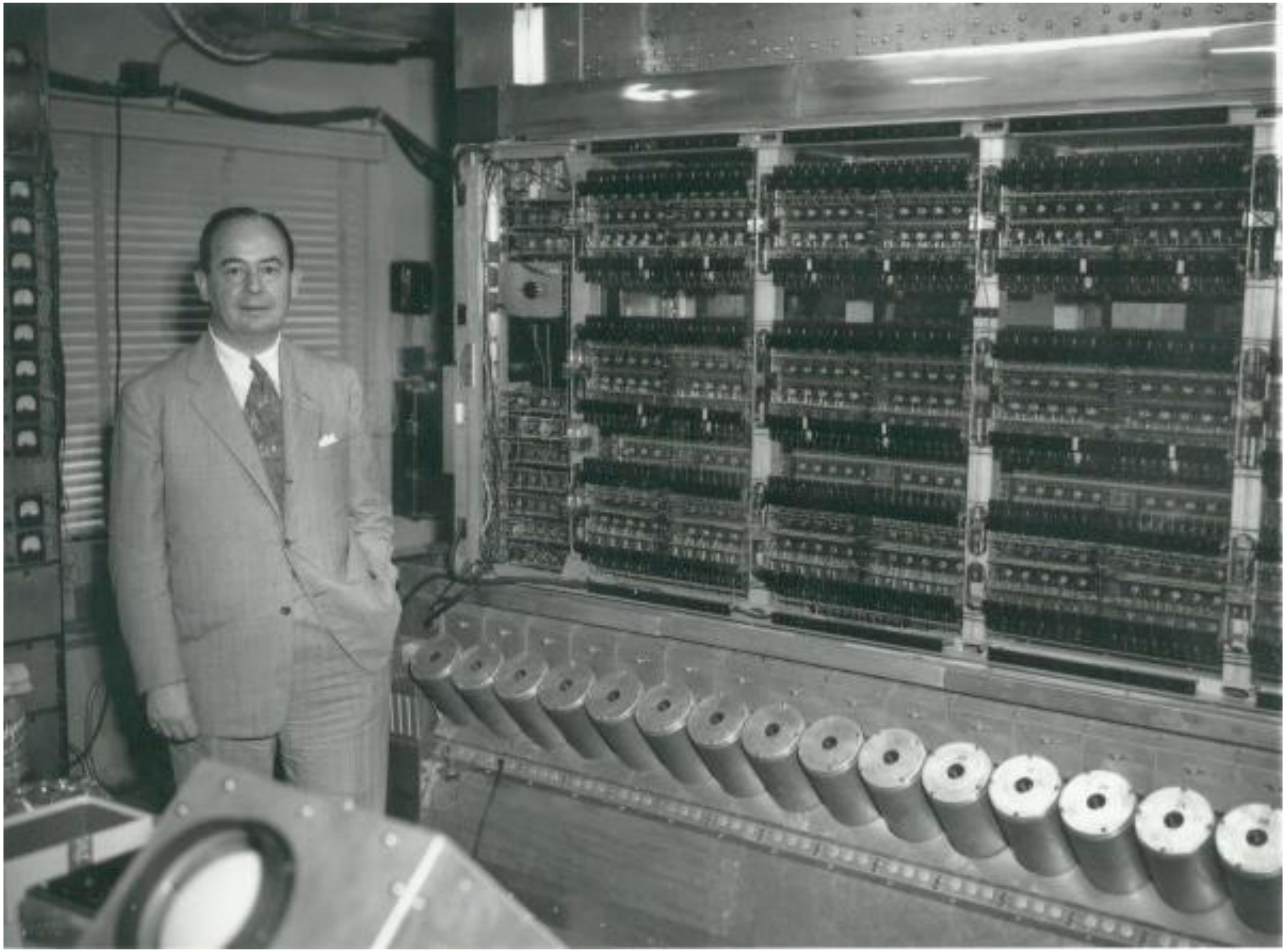
- *ENIAC (Electronic Numerical Integrator And Computer)*
  - Designed by John Mauchly and John Presper Eckert at University of Pennsylvania
  - Funded by US BRL (Ballistic Research Lab) to develop range and trajectory tables for new weapons
    - Until then, BRL employee more than 200 people with desktop calculators to solve the necessary ballistics equations
    - The proposal accepted in 1943, the machine completed in 1946, and dismantled in 1955
  - Used for H-bomb research
  - Characteristics
    - 30 tons, 15000 square feet, 18000 vacuum tubes, 140 KW power dissipation
    - Decimal machine
      - 20 accumulators each holding 10-digit decimal number
      - Each digit is represented by a ring of 10 vacuum tubes
    - *Manually programmed by setting switches and plugging/unplugging cables*
    - 5,000 additions per second

ENIAC



Source: Wikipedia

The Von Neumann Machine & IAS



Source: IAS

Simple Batch Systems

- ❑ *Monitor*
  - User submits the job on cards or tape to a computer operator, who batches them together sequentially and places them on an input device
  - Monitor is a resident software in main memory
  - Monitor reads in jobs one at a time from the input device.
  - The current job is placed in the user program area
  - The control is passed to the job.
  - When the job is completed, it returns control to the monitor
    - User no longer has direct access to the processor
- ❑ ***History***
  - The first batch OS was developed by GM in the mid-1950s for use on IBM 701
  - By the early 1960s, a number of vendors developed batch OS for their computer systems

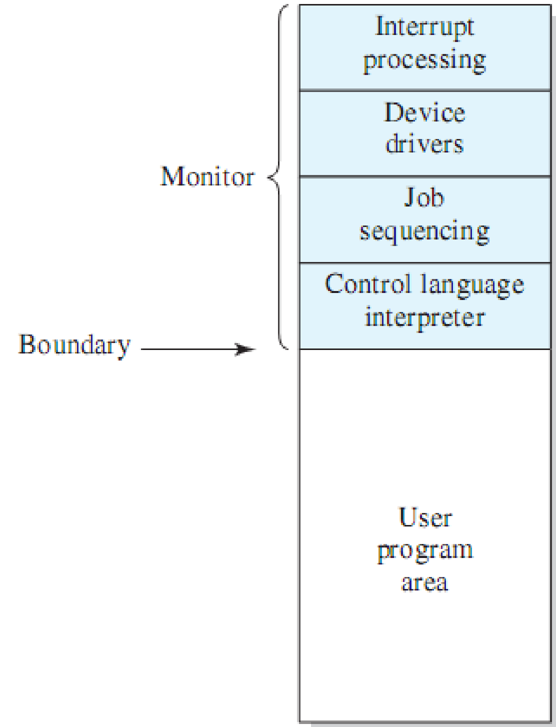


Figure 2.3 Memory Layout for a Resident Monitor

Source: Pearson

Batch Systems: Problems

- *Processor is often idle*
  - Even with automatic job sequencing
  - I/O devices are slow compared to processor

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

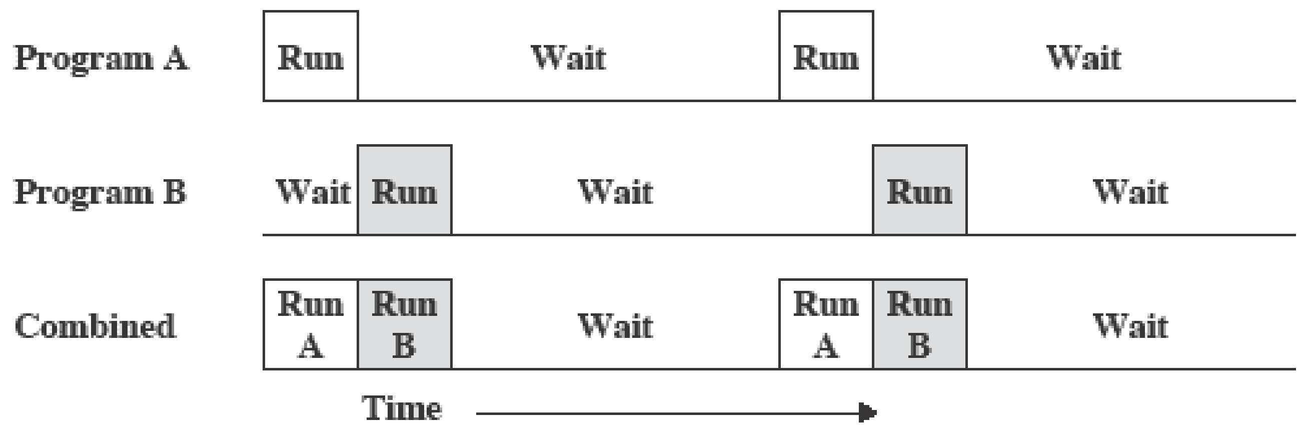
$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Source: Pearson

Figure 2.4 System Utilization Example

(a) Uniprogramming

Multiprogrammed Batch System



(b) Multiprogramming with two programs

Source: Pearson

- *When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O*
  - Also known as multitasking
  - Memory can be expanded to hold three, four, or more programs

Multiprogramming Example

Table 2.1 Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Source: Pearson

Utilization Histogram

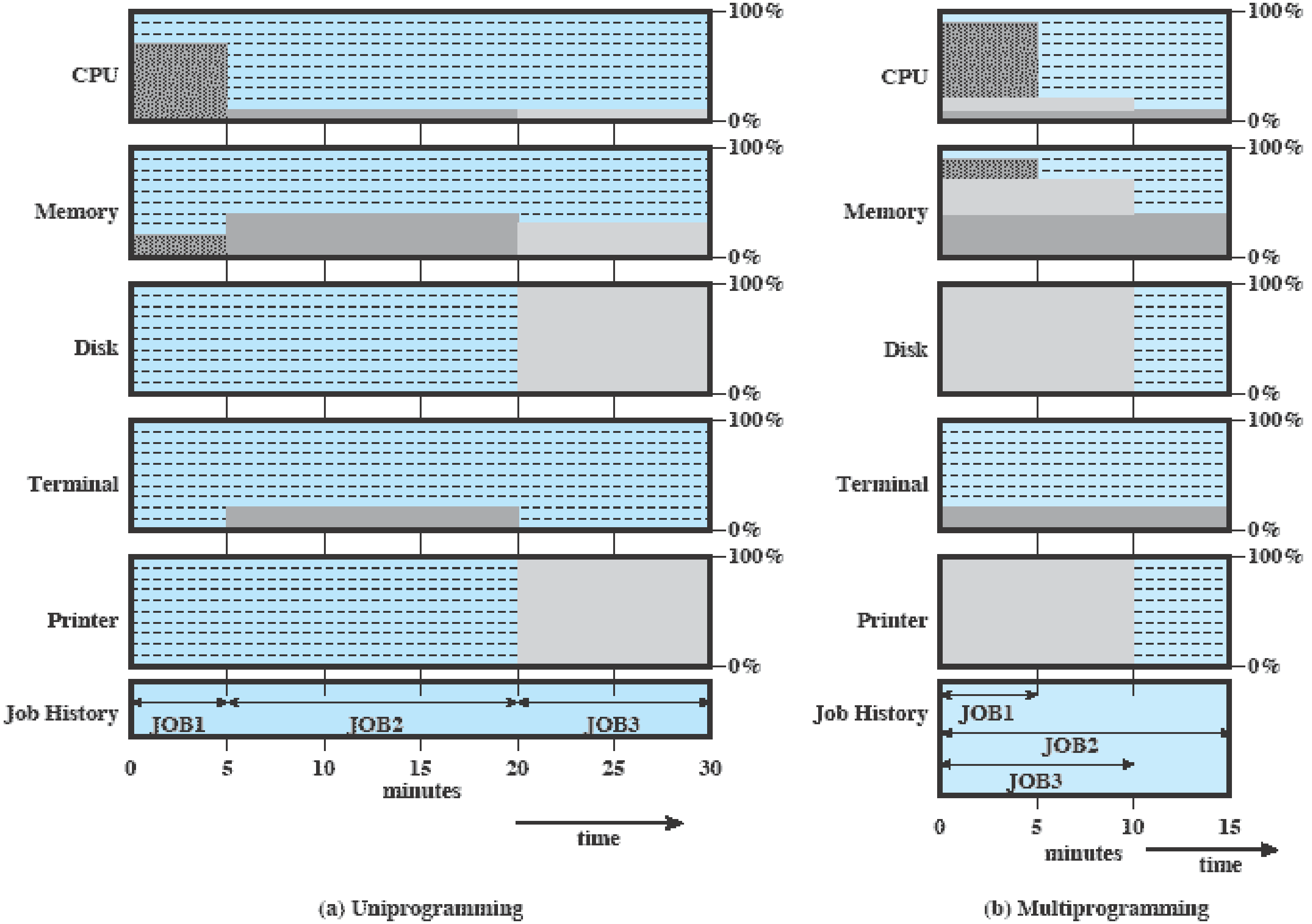


Figure 2.6 Utilization Histograms

Source: Pearson

Effects on Resource Utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Table 2.2 Effects of Multiprogramming on Resource Utilization

Source: Pearson

Time-Sharing Systems

- *Can be used to handle multiple interactive jobs*
  - In a time-sharing system, several users share a computer system through terminals at the same time.
  - In this system, *minimizing response time is more important* than maximizing throughput (processor utilization)
  - OS interleaves the execution of each user program in *time slice.*
    - Context switching occurs every time slice. The currently running job is suspended and a previously suspended job (or a new job) is resumed (is selected for execution).
  - Short jobs or I/O intensive jobs do not have to wait for long compute-intensive jobs

Compatible Time-Sharing Systems

- *CTSS: One of the first time-sharing OS*
  - Developed at MIT by a group known as Project MAC
  - Ran on a computer with 32,000 36-bit words of main memory, with the resident monitor consuming 5000 words
  - To simplify both the monitor and memory management a program was always loaded to start at the location of the 5000th word
    - First developed for IBM 709 in 1961 and later transferred to IBM 7094
    - Running on 7094, CTSS supported 32 users
- *Time Slicing*
  - System clock generates interrupts at a rate of approximately one every 0.2 seconds
  - At each interrupt OS regained control and could assign processor to another user
  - Old user programs and data were written out to disk
  - Old user program code and data were restored in main memory when that program was next given a turn

CTSS Operation

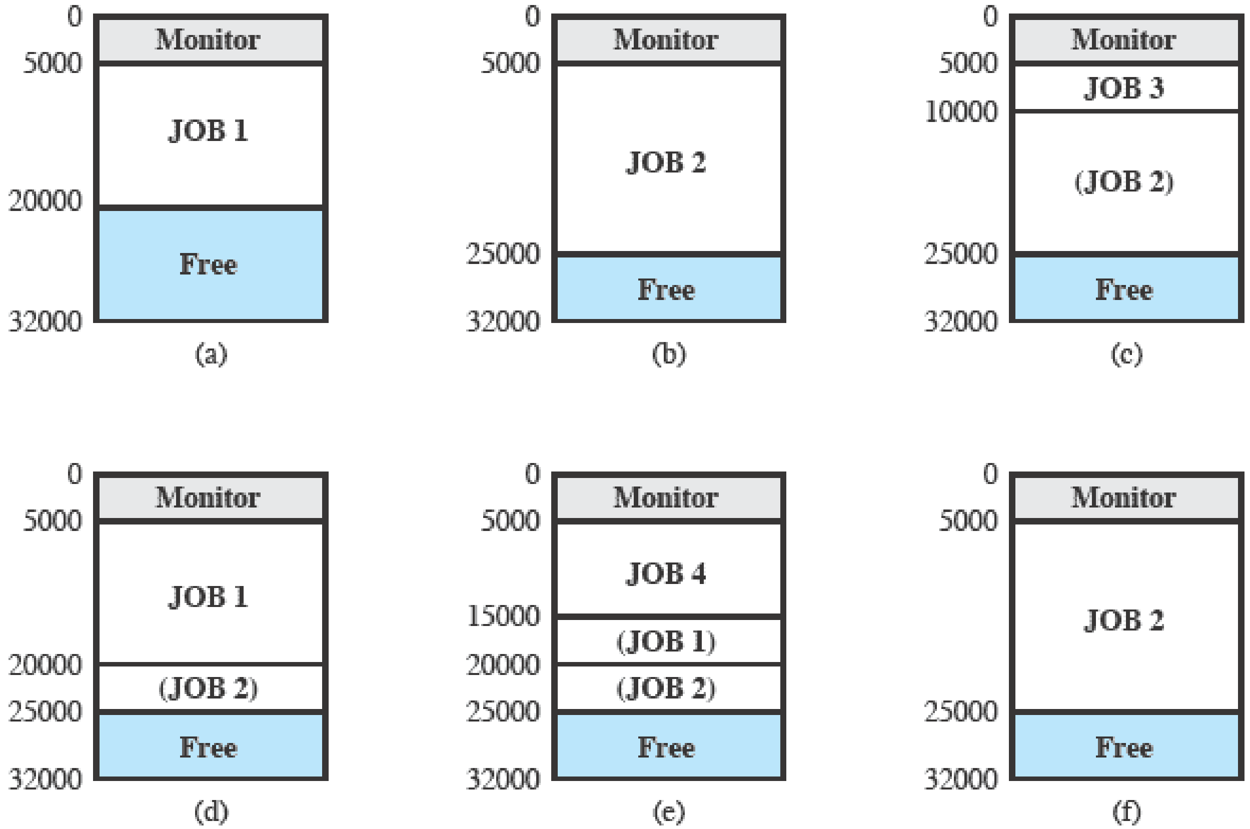


Figure 2.7 CTSS Operation

Source: Pearson

Homework 1

- *Read Chapter 1*
- *Read Chapter 2*
- *Exercise 2.1*
- *Exercise 2.2*
- *Exercise 2.4*
- *Read Chapter 3*