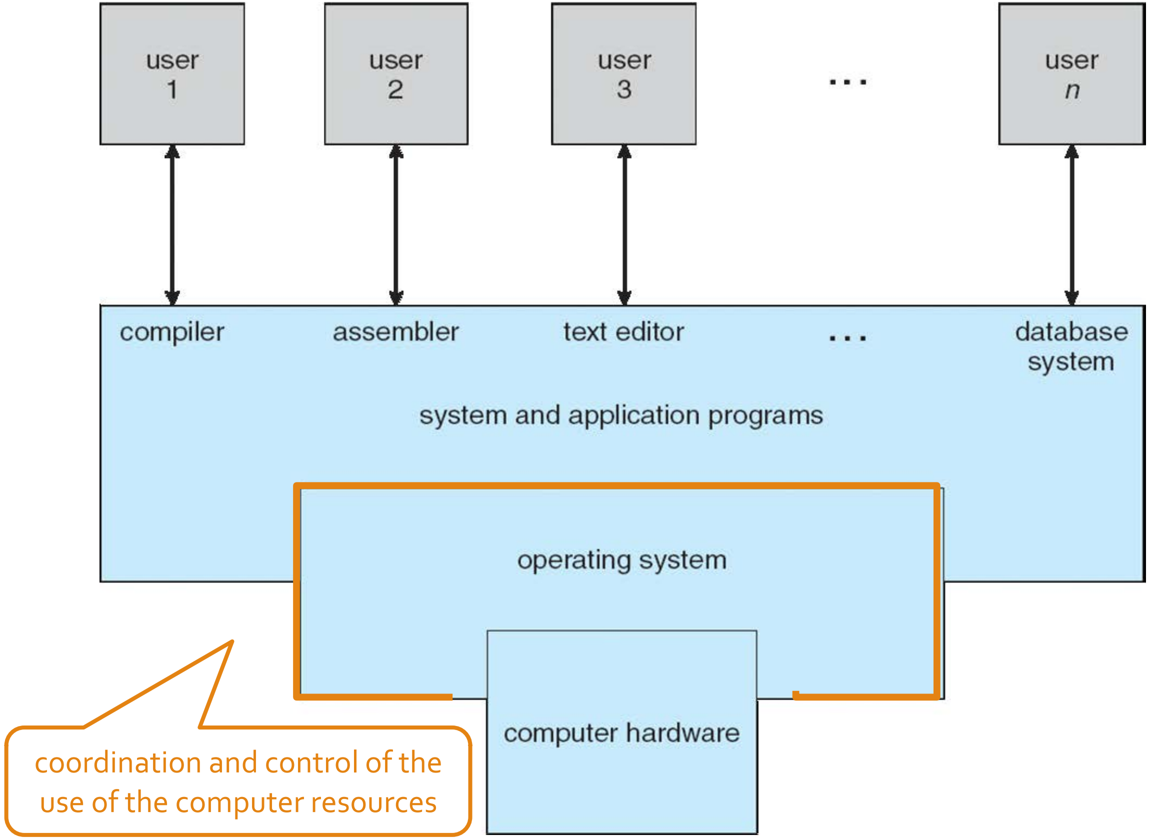
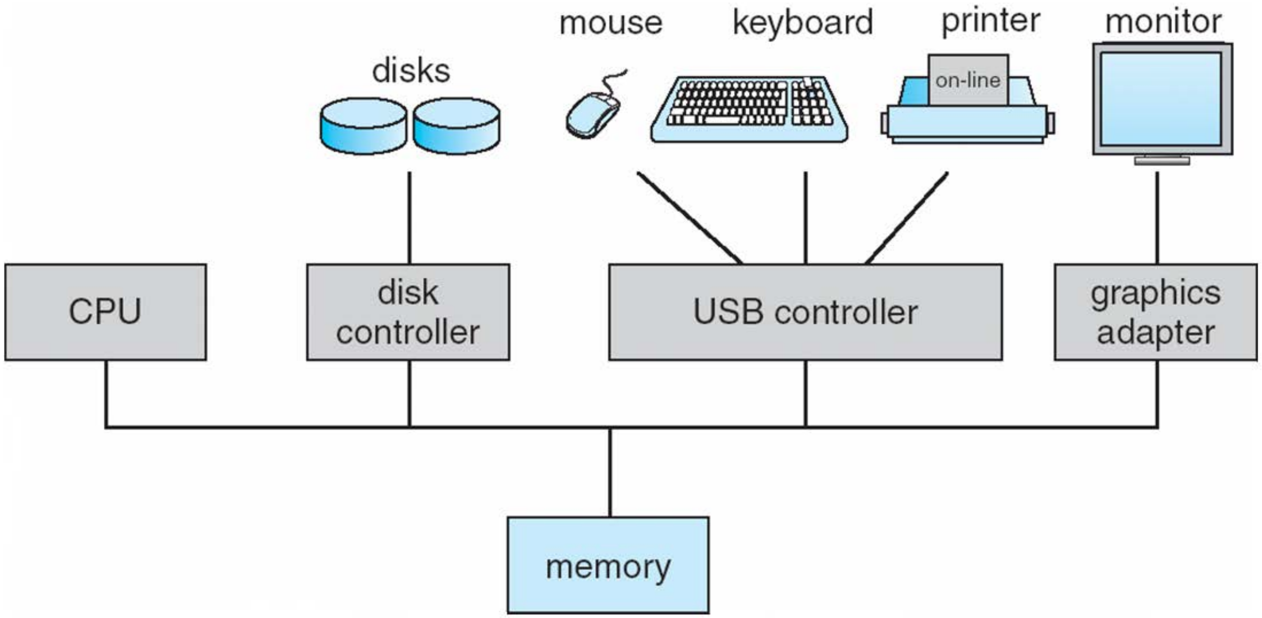
OPERATING SYSTEMS

COMPUTER ARCHITECTURES

Components of a computing system



Computer System Organization



Microprocessor

- Invention that brought about desktop and handheld computing
- Contains a processor on a single chip
- Fastest general purpose processors
- Multiprocessors
  - Each chip (socket) contains multiple processors (cores)

Graphical Processing Units (GPU's)

- Efficient computation on arrays of data using Single-Instruction Multiple Data (SIMD) techniques pioneered in supercomputers
- No longer used just for rendering advanced graphics Also used for general numerical processing
  - Physics simulations for games
  - Computations for complex machine learning models

Digital Signal Processors (DSPs)

- Streaming signals such as audio or video
- Used to be embedded in I/O devices like modems
  Are now becoming first-class computational devices, especially in handhelds
- Encoding/decoding speech and video (codecs)
- Provide support for encryption and security

System on a Chip (SoC)

- To satisfy the requirements of handheld devices, the classic microprocessor is giving way to the SoC
- On the same chip
  - not only the CPUs and caches but also
  - DSPs
  - GPUs
  - I/O devices (such as codecs and radios)

Main Memory

- This is the **core** component of a computer system
  - Each component `read`s from and `write`s to the main memory
- The CPU can only execute instructions that are already stored in the main memory.

Main Memory Organization

- Direct access, location independent
  - RAM and DRAM (Dynamic Random Access Memory)
- ROM (Read-Only Memory) to store instructions and data do not have to be (frequently) modified
- Memory organization
  - An array of cells with equal fixed size called *word*
  - Each memory cell has an associated address, i.e., the position in the array

Memory Access

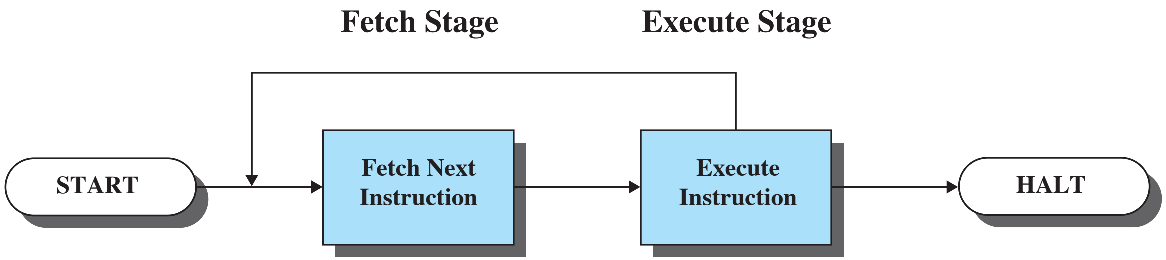
`load <address>`

- Transfer the content of the memory cell identified by `address` into one of the CPU registers

`store <address>`

- Transfer the content of one of the CPU registers into the memory cell identified by `address`

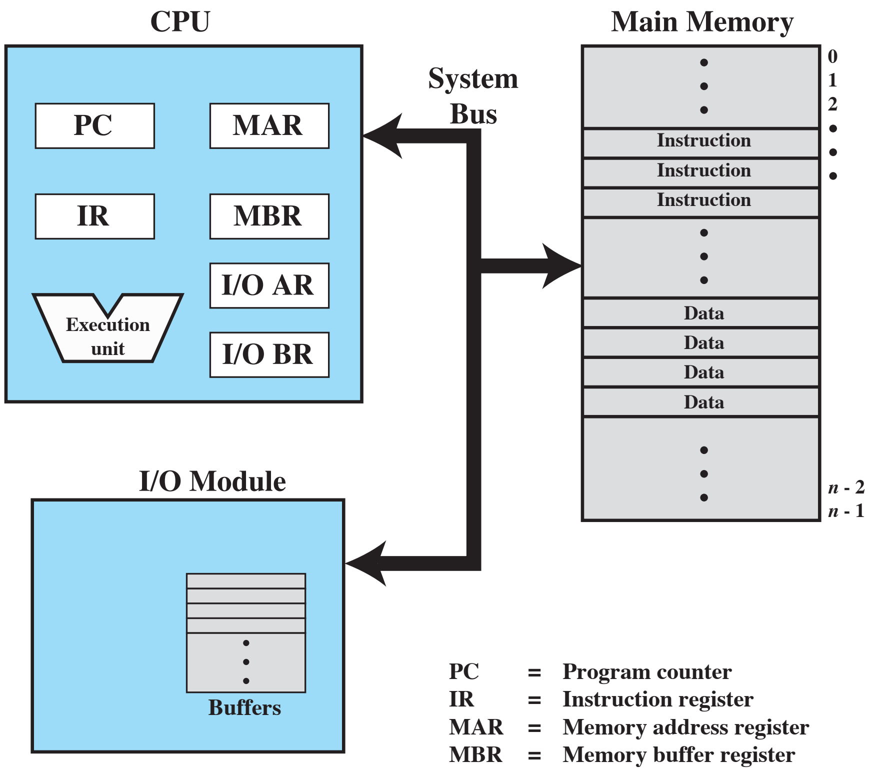
Instruction Execution Cycle



Von Neumann Architecture

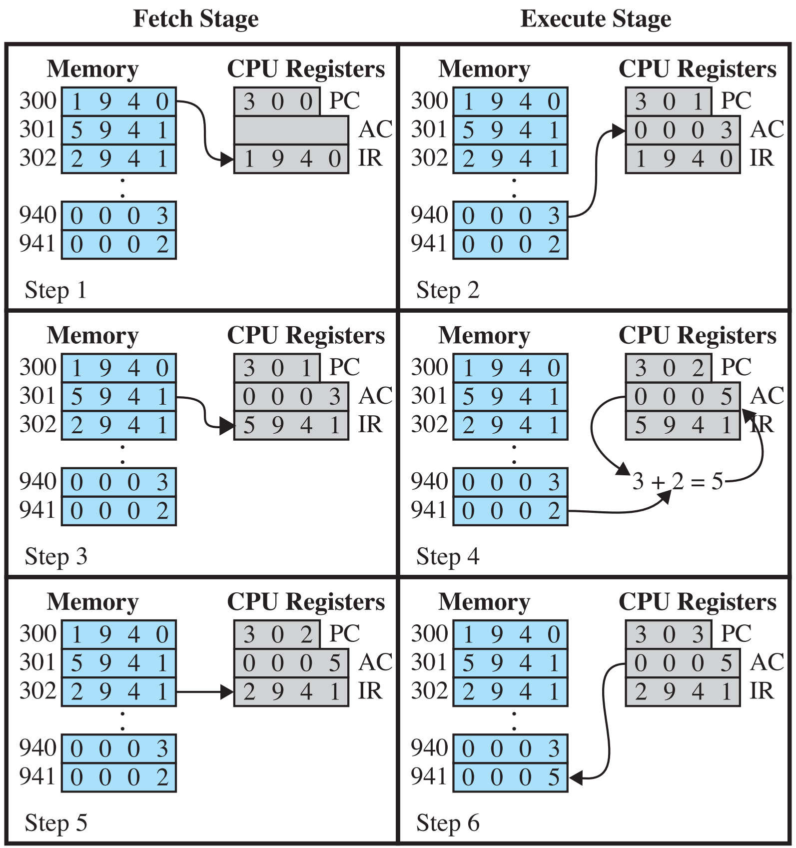
- The processor *fetches* the next instruction from the memory address stored in Program Counter (PC) register
- After the fetch phase, the Program Counter is incremented by 1 unit

CPU Registers



- PC = Program counter
- IR = Instruction register
- MAR = Memory address register
- MBR = Memory buffer register
- I/O AR = Input/output address register
- I/O BR = Input/output buffer register

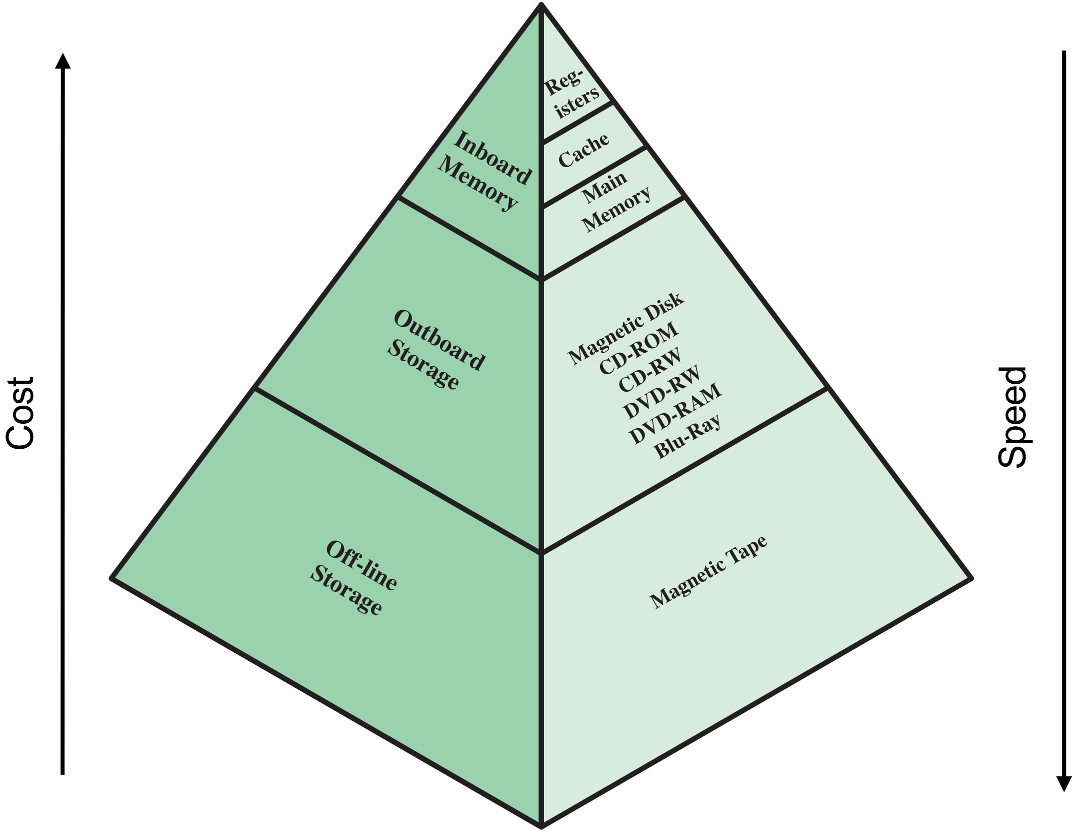
An Example of the execution of the sum of two numbers



Which size for the main memory?

- *In principle* all the instructions and data of the programs in execution should be stored in the main memory.
  However…
  - The memory size is usually limited by cost constraints
  - The technology used to fabricate the main memory does not support permanent storage.

The Memory Hierarchy



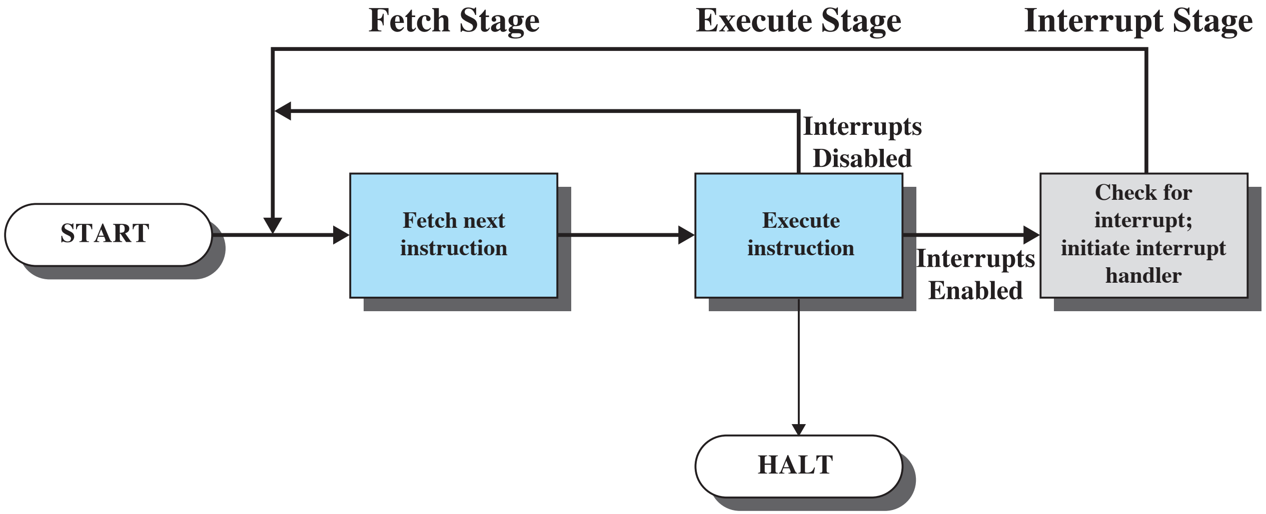
System Boot

- System Bootstrap
  When the system is switched on, a small program stored in a ROM (a.k.a. *firmware*) is executed
  - all components are tested and initialised
  - the operating system is loaded into main memory
    - typically just a small portion, the kernel
  - the operating system starts running
  - the operating system keeps waiting for some event

Events

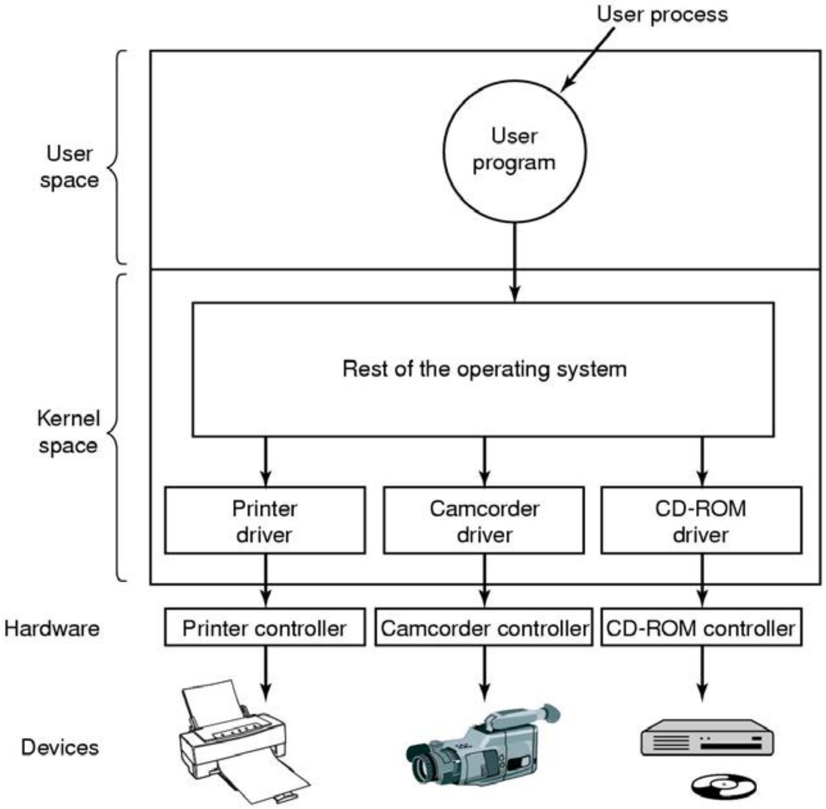
- The execution of a program can be seen as the execution of a sequence of instructions.
- Some events can break the continuous flow of execution of instructions
  - Events coming from a physical device
    - interrupts
  - Events generated by the program itself
    - exceptions
      hardware (i.e., division by zero)
      system calls, i.e., a call for a service from the OS

Interrupts



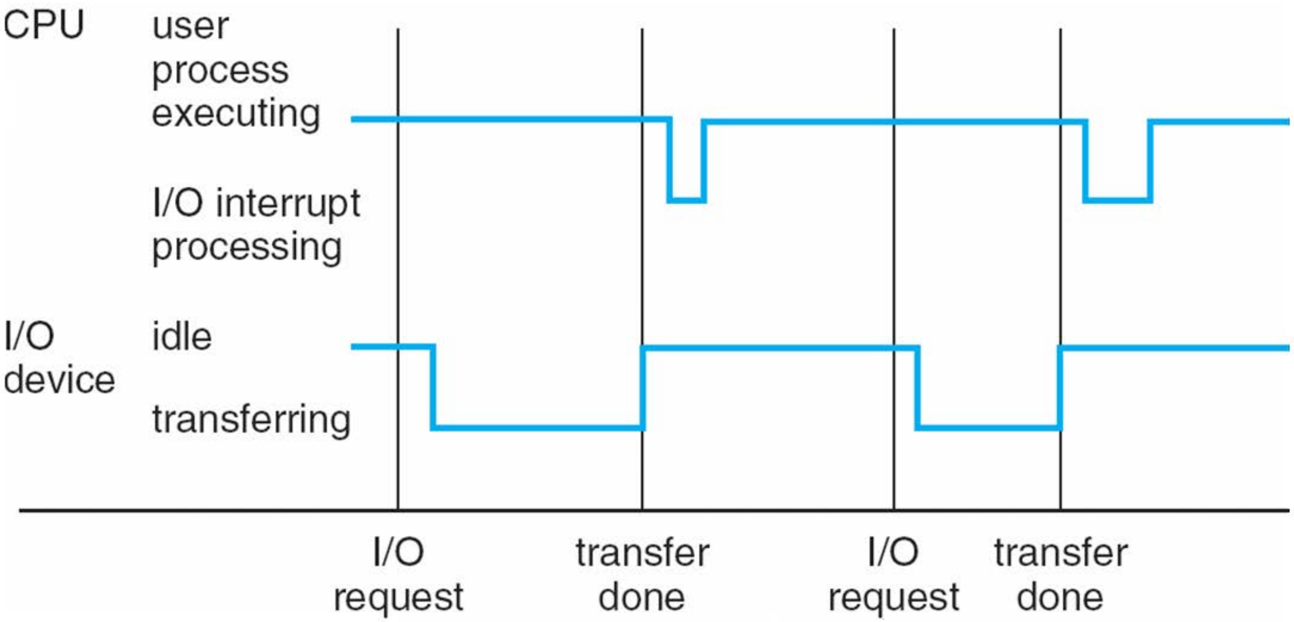
Fetch Stage
Execute Stage
Interrupt Stage
Interrupts Disabled
START
Fetch next instruction
Execute instruction
Interrupts Enabled
Check for interrupt; initiate interrupt handler
HALT

Interrupts and Device Drivers



1. The user program request the use of one device
2. The operating system sends the request to the device driver
3. The device driver sets the values of the registers of the device controller
4. The device controller starts the data transfer
5. After the data transfer is completed, the controller sends a signal to the driver, the driver notifies the operating system, and the OS notifies the user program

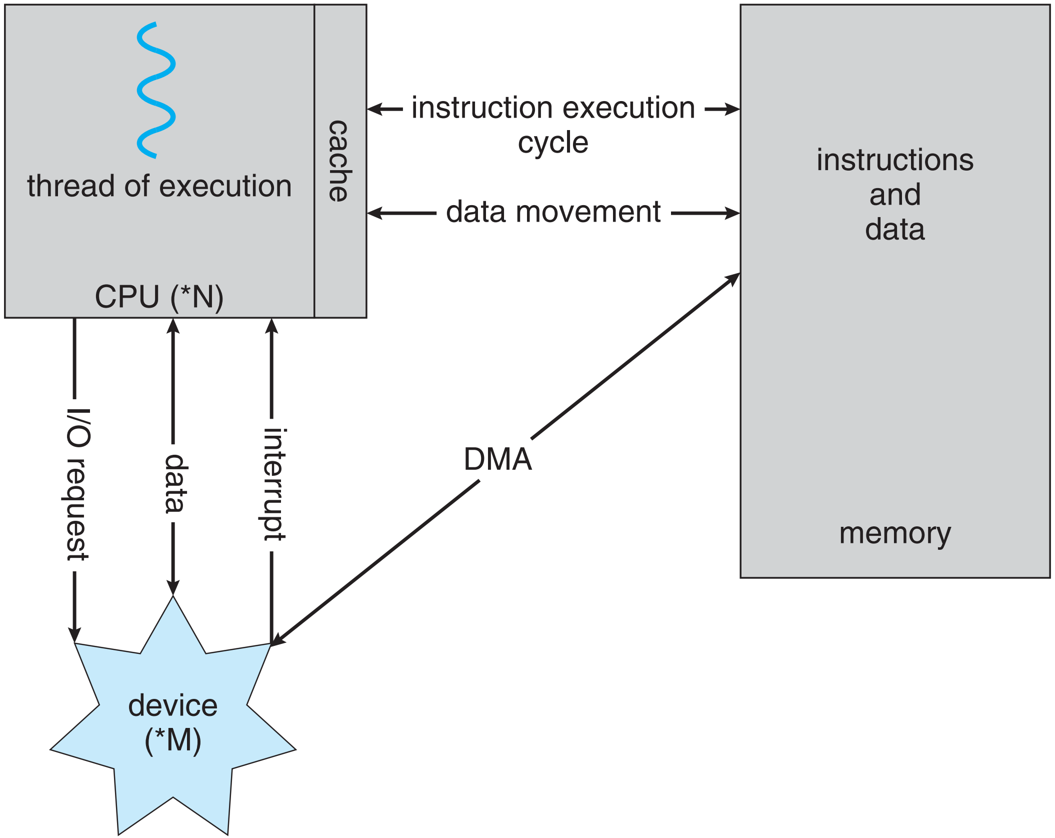
Interrupt timeline



Direct Memory Access (DMA)

- Data transfers from one device to main memory can be managed
  - by the CPU
  - by a specific hardware module called DMA

How data transfer works



CPU Organization

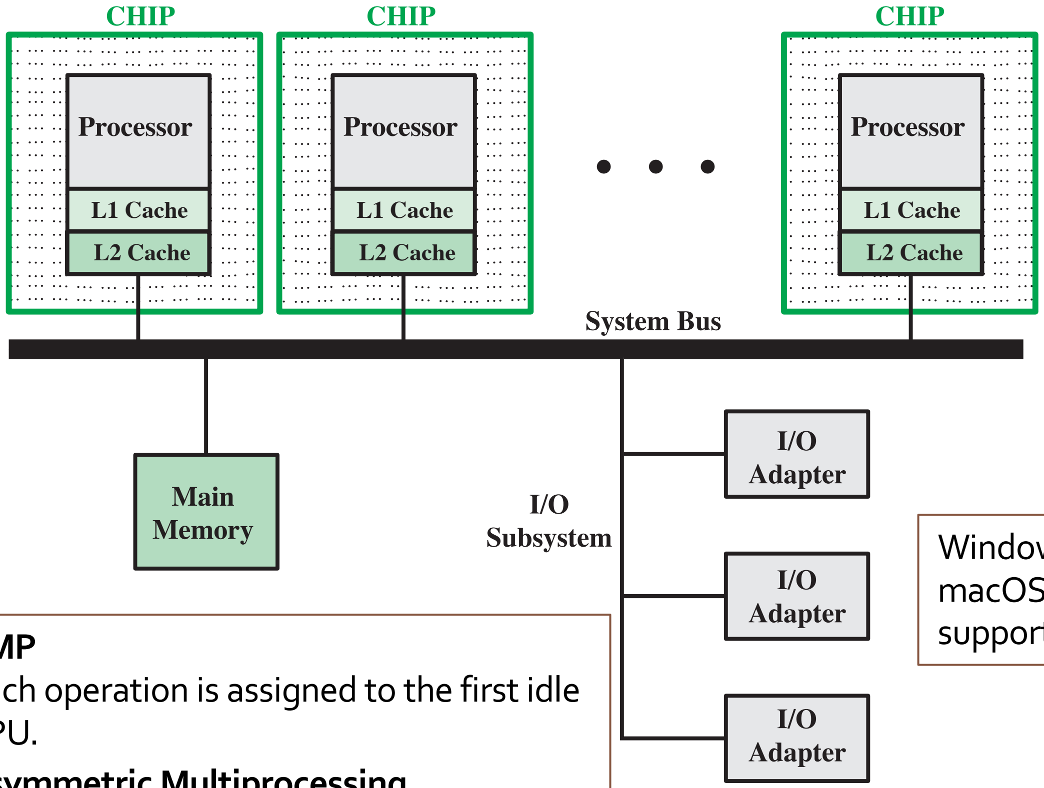
Uniprocessing and Multiprocessing Architectures

- Uniprocessor architectures
  - A single CPU that is responsible for performing *all* tasks,
- Multiprocessor Architectures
  - More than one CPU is available
    - CPUs with different ISAs, e.g., GPU
    - CPUs with the same ISA
  - Current multicore processors are an evolution of multiprocessor architectures

Advantages of Multiprocessing Architectures

- Productivity
  - increased *throughput*, i.e., more tasks can be completed in a time unit.
- Economy of scale
  - One multiprocessor system could be economically preferable to a cluster of uniprocessor systems
- Reliability
  - fault tolerance
  - graceful degradation

SMP
Symmetric Multiprocessing

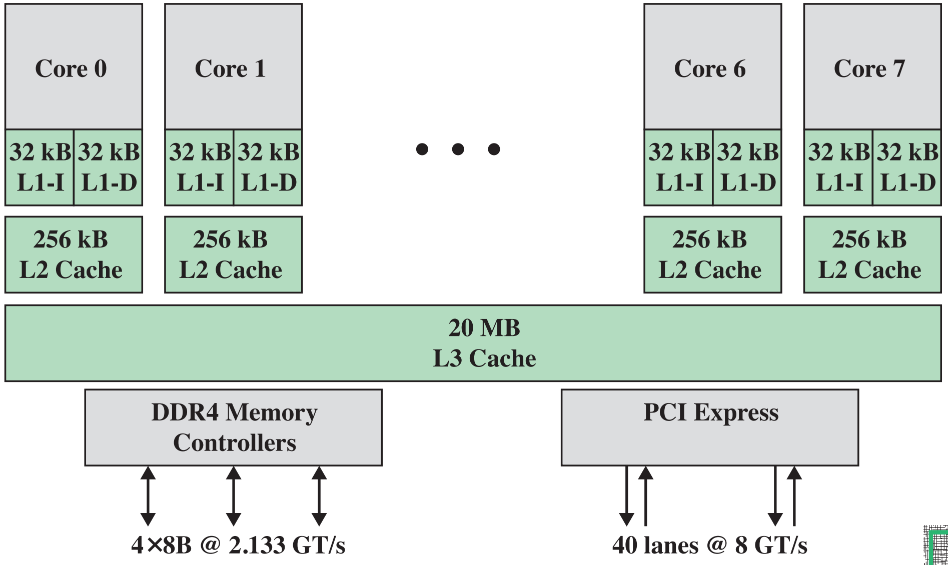


SMP
Each operation is assigned to the first idle CPU.

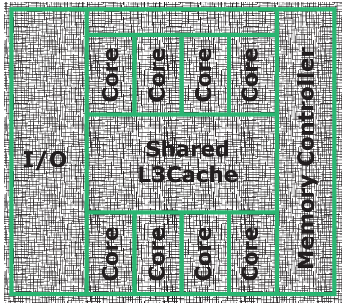
Asymmetric Multiprocessing
each CPU executes a specific task

Windows, macOS, Linux support SMP

Multicore systems



(a) Block diagram

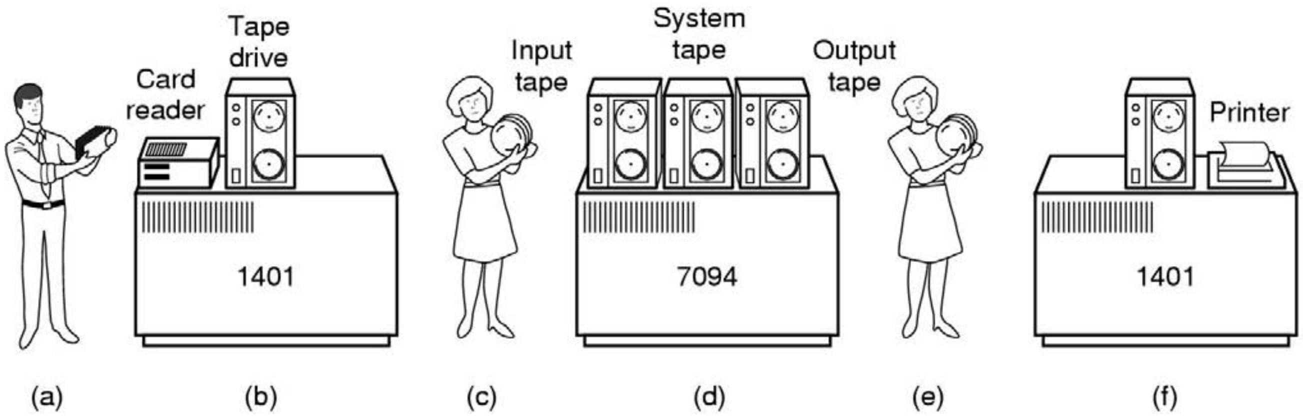


(b) Physical layout on chip

Intel Core i7-5960X

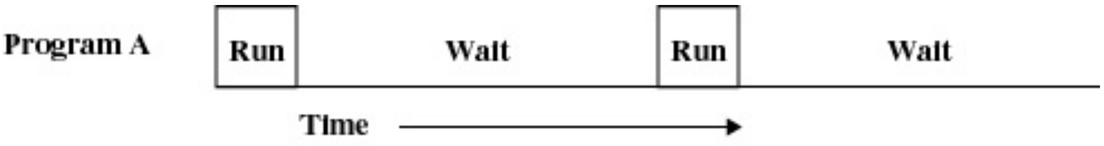
Structure of the Operating System

Before the OS



Uni programming

If the CPU is much faster than the I/O system, then the execution of one job at a time might result in the CPU idle most of the time.



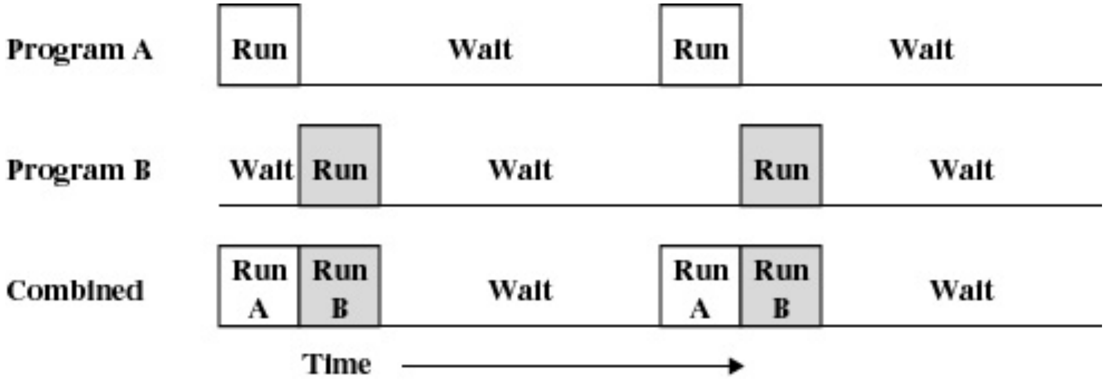
(a) Uniprogramming

Read one record from file	0.0015 seconds
Execute 100 instructions	0.0001 seconds
Write one record to file	0.0015 seconds
TOTAL	0.0031 seconds

$$\text{Percent CPU Utilization} = \frac{0.0001}{0.0031} = 0.032 = 3.2\%$$

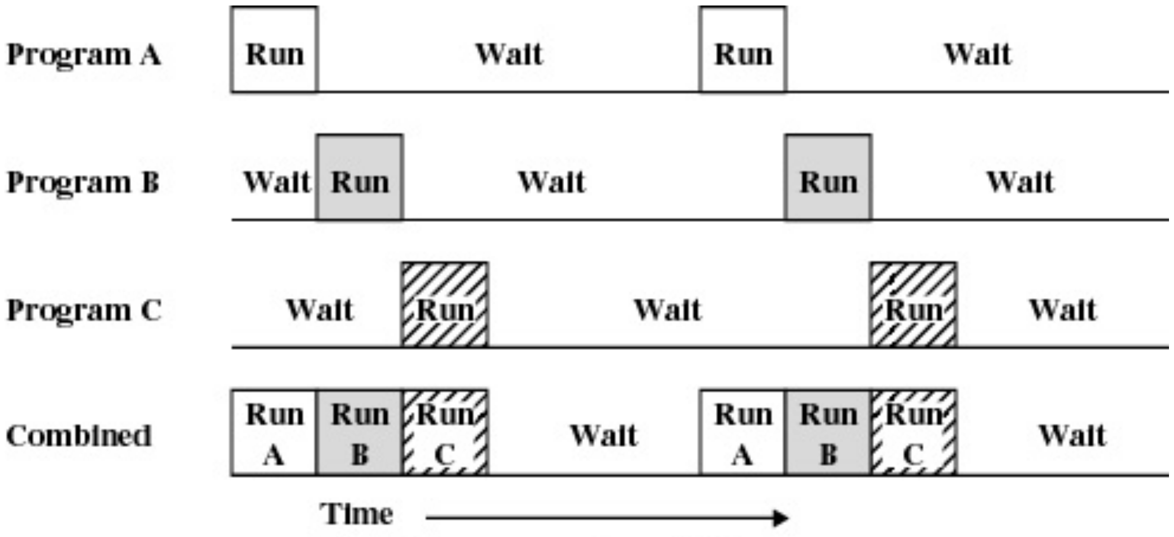
Multi programming

If we allow more jobs to run concurrently, the usage of the CPU increases, as well as the throughput of the system.



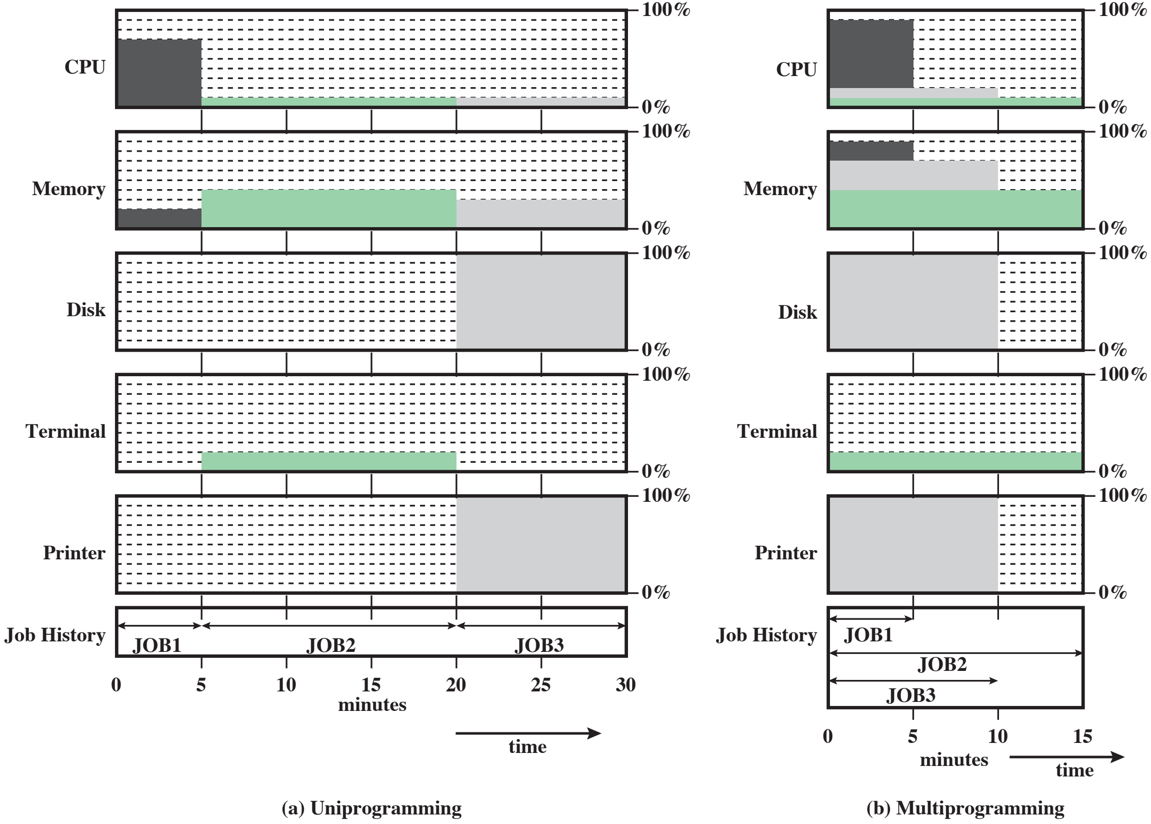
(b) Multiprogramming with two programs

Multi programming



(c) Multiprogramming with three programs

Multi programming

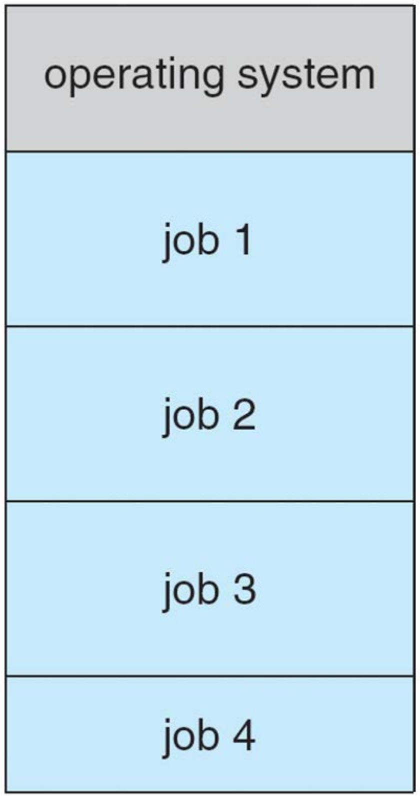


Uni Programming Vs. Multi Programming

	Uniprogramming	Multiprogramming
CPU usage	20%	40%
Memory usage	33%	67%
Disk usage	33%	67%
Printer usage	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/h	12 jobs/h
Average response time	18 min	10 min

Memory Layout for Multi programmed Systems

- The main memory should always be filled with jobs to keep the CPU working
- The operating system is in charge of
  - selecting the jobs to be resident in memory
  - scheduling the execution of the jobs
  - Managing interactive usersi



Timesharing systems

- Systems that allows interactions from multiple users
- The goals of the system are quite different from those of a multiprogrammed system
  - Multiprogramming seeks the maximization of the throughput
  - Time-sharing aims at reducing the response time to each user

Operating System components

- Multiprogramming and time-sharing were the the two driving forces that motivated the development of the components of an operating system
  - Concurrency management
  - Job and CPU scheduling
  - Virtual memory
  - File System Management
  - Disk and Storage Management
  - Data and Software Privacy and Security

Tasks of the Operating System

Interrupts

- If no program is executing…

  …the operating system should keep waiting

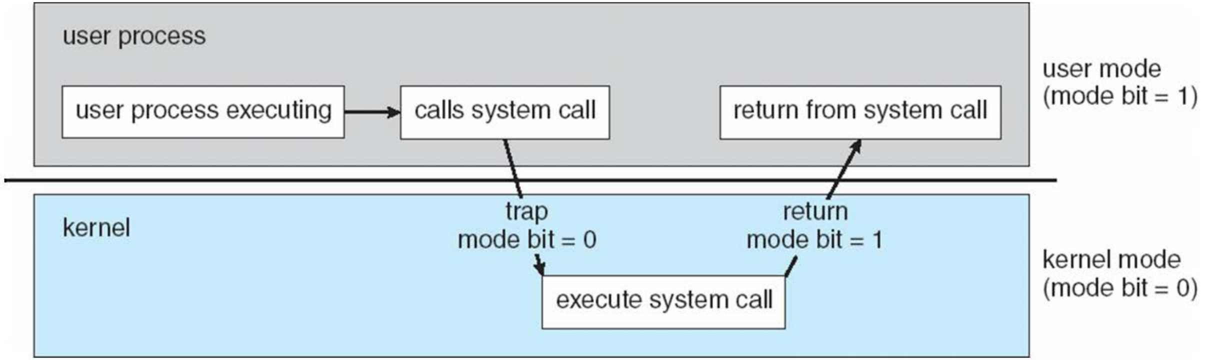
- The task of the operating system is to react to interrupts that are generated by users' programs

…However, the operating system is a program…

How this ambiguity has been addressed?

Two operating modes

- Each CPU has two modes of operation
  - user mode
  - kernel mode
- Privileged instructions can be executed only when the CPU is in kernel mode (a.k.a. supervisor mode)



Mode Switch

- Any CPU has an instruction that cause the mode switch
  - Only the operating system can execute this instruction
- Each time an activity is needed by the operating system
  - interrupt
  - system call

then the CPU switches from user to kernel mode

At the end of the activity the CPU switches back to user mode

It turns out the the operating system is strictly coupled with the underlying hardware platform.

Virtualization

- Virtualization can be seen as an extension of the modes of operation of the CPU
- Modern CPUs support different levels of privileged instructions called *rings*
  - sets of privileged Instructions with more privileges that user-level instructions and less privileges than kernel-level instructions
- Intel 64
  - 4 privilege levels, even if none of them is explicitly related to virtualization

Timer

- The operating system must control that no process
  - enters an infinite loop
  - owns some resources without releasing them at some time (such as the CPU, main memory, I/O channels, etc.)
- To this end, a timer is included in the computer hardware.

Process Management

- A coarse definition of process is the following:
  a process is a program in execution on a computer
- The operations that the operating system performs are the following:
  - process creation and termination
    - resource management (memory, CPU, I/O)
  - suspend and resume process execution
  - process synchronization and communication (IPC – interprocess communication mechanisms)
  - deadlock prevention and management

Memory management

- The main memory is the place where both instructions and data are stored in the Von Neumann architecture
  - Both the CPU and I/O devices need to get access to the main memory
- The operating system and the underlying hardware components implement specific mechanisms to properly manage the memory hierarchy
  - Note that the cache is not managed by the operating system

Permanent Storage

- File system
  - A structure to organise documents and program permanently stored on disks, solid state media, etc.
    - source code, binary code, user documents, images, music, video, etc.
- Operations on files
  - create / delete
  - binding files with the device they are stored on
  - backup and recovery

Disk management

- Disks represent the typical permanent storage device
  - data on a SSDs are organised in the same way as a mechanical disk
- Management tasks of the operating system
  - free space
  - file allocation
  - scheduling
    - to minimise the average response time in mechanical disks

Memory Hierarchy

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

I/O management

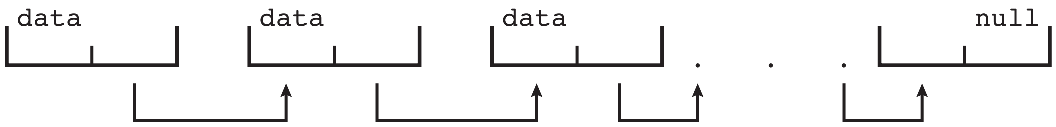
- The operating system is in charge of
  - buffer management
  - spooling - ***s**imultaneous **p**eripheral **o**peration **o**n**l**ine*
  - providing standard interfaces for device drivers
  - managing device drivers

Privacy & Security

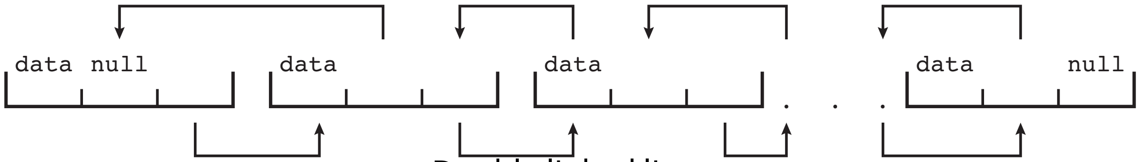
- The OS is in charge of executing different processes from different users
  - Protection of
    - processes
    - users
    - the OS itself
  - from errors caused by other processes or users
    - casual errors as well as intentional errors!

Data structures used by the kernel

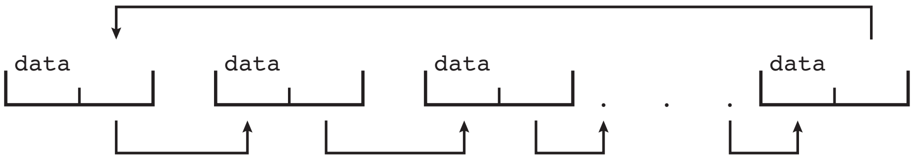
Lists



Singly linked list



Doubly linked list

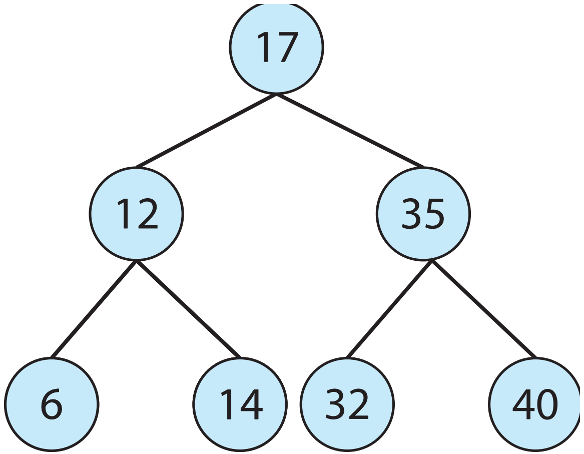


Circularly linked list

Stacks and Queues

- **Stack**
  - An ordered sequence of data where data can be inserted or extracted from one side only, i.e., the top of the stack
    LIFO (Last-In-First-Out)
    - `push` inserts a new item onto the stack
    - `pop` extracts the item, on top of the stack
  - This structure is used by the OS to store local variables, and return address to be used after the last instruction of a function
- **Queue**
  - An ordered sequence of data where insertion is performed on one side, and extraction from the other side, so that items are extracted in the same order as they join the queue
    LIFO (Last-In-First-Out)

Trees



- Suited to represent hierarchies
- Binary search trees
  - each node with two children
  - the left node has a value less than the parent node
  - the right node has a value greater than the parent node

Hash functions and Bitmaps

- **Hash functions**
  - Any function that maps data of arbitrary size to fixed-size values
    - e.g., a mapping between all students' names and the set of integers from 0 to 49
  - Collisions
    Depending on the function, and on the input size, two ore more inputs could be mapped to the same output value
- **Bitmaps**
  - A bit array of size N can be used to store the binary status of a set of N items
    - e.g., the list of free disk blocks

Computing Environments

Traditional computing

- The meaning of traditional has evolved in the past years
- Nowadays a traditional computer is a networked computer
  - Scientific and technical computation
  - Graphic design and engineering projects
  - Desktop publishing and Multimedia processing
- User's programs and data may be stored on
  - the local computer
  - some remote servers

Mobile Devices

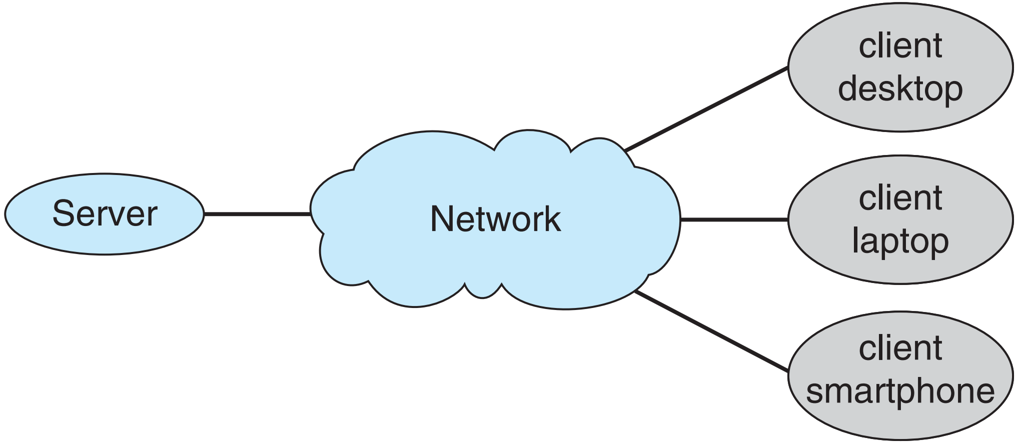
- Current mobile OSs exhibit the same functionalities as traditional computers
  - power management constraints
  - memory size constraints
- Main features
  - interaction by gestures and touch screens
  - always connected
  - localization services
  - movement sensors
  - other environmental or health related sensors

Distributed Systems

- All computers support networking
- Clusters of computers can be set-up to act as a single server.
- The OS can be executed on multiple hardware devices whose resources are considered as being part of one large distributed computer
  - distributed file system
  - distributed process management

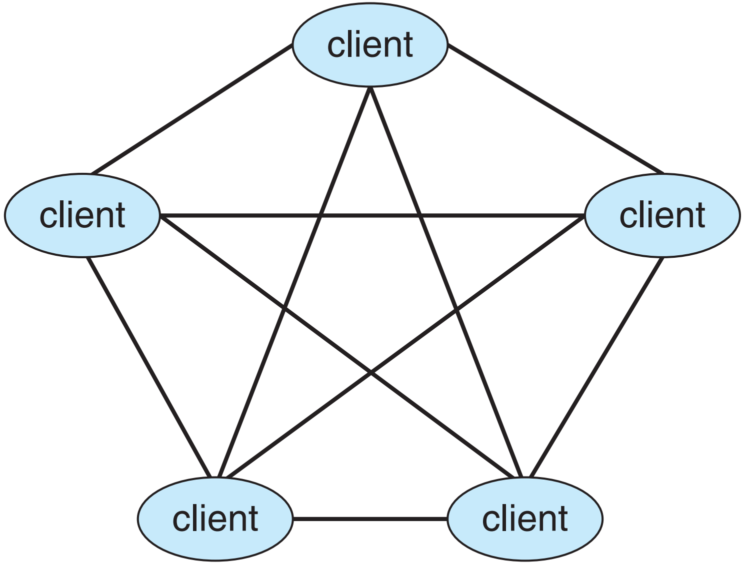
Client-Server

- Computational intensive tasks are executed by powerful server computers



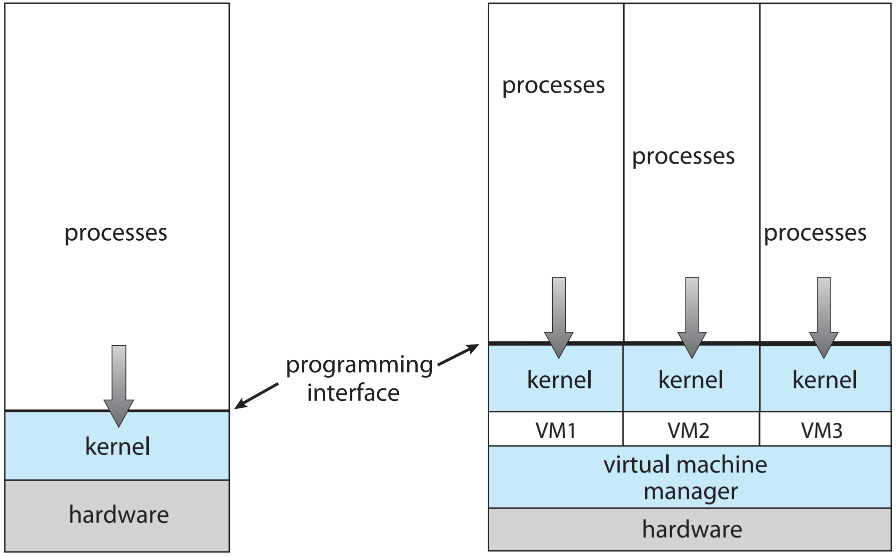
Peer-to-Peer

- Each node acts as a server for the other nodes
- A centralised registry can support the cooperation between nodes



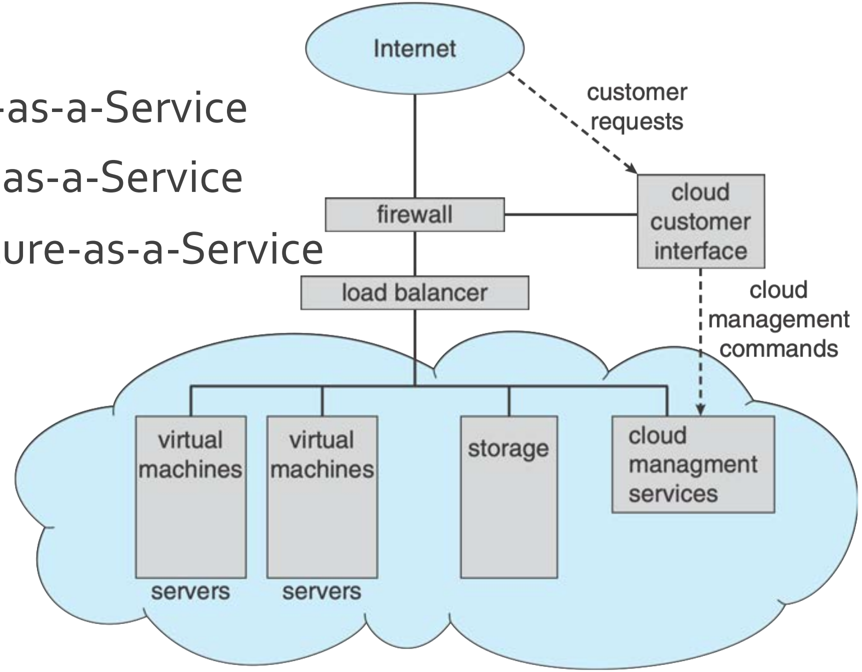
Virtualization

- Emulation or virtualization of the CPU
  - Concurrent execution of more than one operating system (guests) on one piece of hardware
    - Virtualisation software executed either directly on hardware or as a program executed by a host OS



Cloud Computing

- Public cloud
- Private cloud
- Hybrid Cloud
- SaaS – Software-as-a-Service
- PaaS – Platform-as-a-Service
- IaaS – Infrastructure-as-a-Service



Embedded systems

- These systems are more and more part of our daily lives as well as part of many production farms
  - automotive
  - aeronautics
  - household appliances
  - smartphones, Smart TV, game consoles, etc.
  - robotics
- The operating system and the underlying hardware are tightly coupled