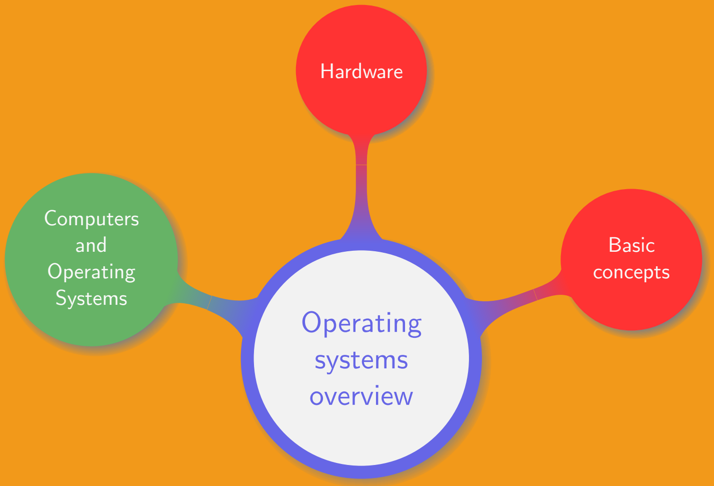
Introduction to Operating Systems

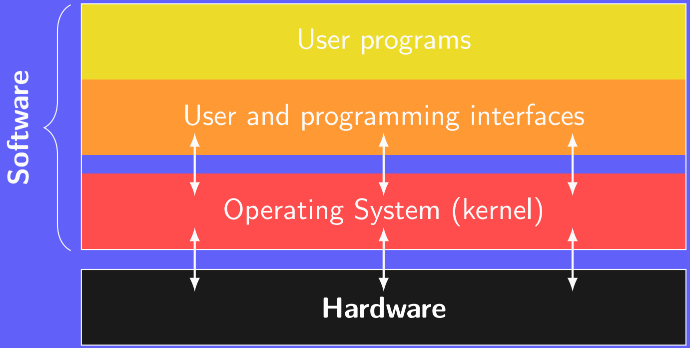
1. Operating systems overview

Manuel – Fall 2020

Chapter organisation



Software and hardware layout



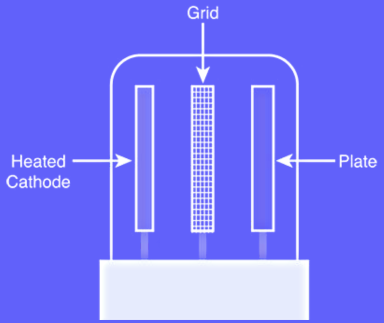
Job of an Operating System (OS):

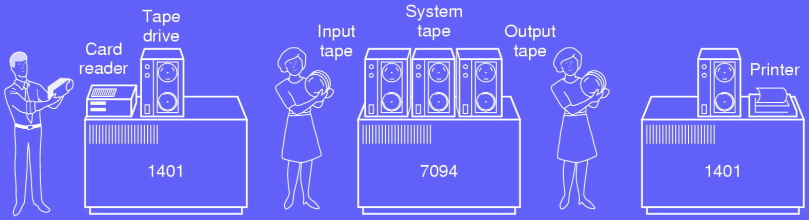
- Manage and assign the hardware resources
- Hide complicated details to the end user
- Provide abstractions to ease interactions with the hardware

A bit of history

The early days:

- Birth of modern computing: 19th century (Babbage)
- Vacuum tube: 1945–1955 (1st generation)
- Transistor: 1955–1965 (2nd generation)





Remmington Rand 409

Using the device:

- Program at most 40 steps
- Wire them on a plugboard
- Read input from cardboards
- Punch output on cardboards

A bit of history

Multiprogramming: 1965–1980 (3rd generation)

- Multiple jobs kept in memory at the same time
- CPU multiplexed among them

A bit of history

Multiprogramming: 1965–1980 (3rd generation)

- Multiple jobs kept in memory at the same time
- CPU multiplexed among them

Multiprogramming requires:

- Memory management: allocate memory to several jobs
- CPU scheduling: choose a job to be run
- Simultaneous Peripheral Operation On Line (SPOOL): load a new job from disk, run it, output it on disk

Modern OS

Most famous OS:

- Disk Operating System (DOS)
- DOS/Basic package sold to computer companies
- MS-DOS, including many features from UNIX
- GUI invented in the 1960s, then copied by Apple
- Microsoft copied Apple (Windows working on top of MS-DOS)
- Many OS derived from UNIX (MINIX, LINUX, BSD...)

Common OS

Device and task oriented OS types:

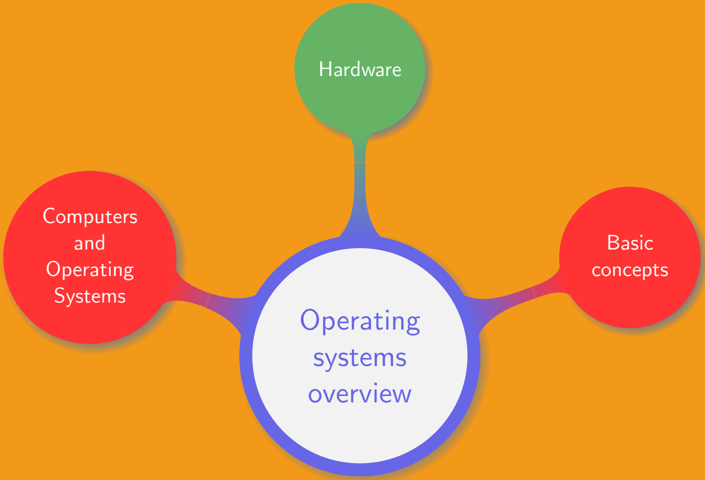
- Personal Computers (PC)
- Servers: serve users over a network (print, web, IM...) $\rightarrow$ Solaris, FreeBSD, Linux, Windows Server
- Multi processors: multiple CPU in a single system $\rightarrow$ Linux, Windows, OS X...
- Handheld computers: PDA, smartphone
- Embedded devices: TV, microwave, DVD player, mp3 player, old cell phones $\rightarrow$ everything stored in ROM, much more simple OS

Less common OS

More device and task oriented OS types:

- Real-Time: time is key parameter (e.g. assembly line, army, avionics...) $\rightarrow$ overlap with embedded/handheld systems
- Mainframe: room-sized computers, data centers $\rightarrow$ OS oriented toward processing many jobs at once and efficient I/O
- Sensor node: tiny computers communicating between each other and a base station (guard border, intrusion/fire detection etc...). Composed of CPU RAM ROM (+other sensors), small battery $\rightarrow$ simple OS design TinyOS
- Smart card: credit card size with a CPU chip, severe memory/processing constraints $\rightarrow$ smallest/primitive OS

Chapter organisation



Basic PC hardware

A computer is often composed of:

- CPU
- Memory
- Monitor + video controller
- Keyboard + keyboard controller
- Hard Disk Drive (HDD) + hard disk controller
- Bus

What are the controllers, and the bus?

CPU

Basics:

- CPU is the “computer’s brain”
- CPU can only execute a specific set of instructions
- CPU fetches instructions from the memory and executes them

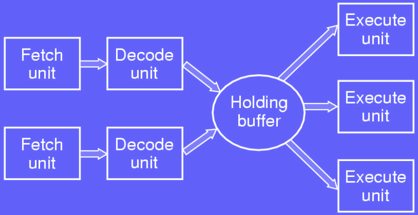
Registers:

- General register: hold variables/temporary results
  e.g. program counter: address of next instruction to fetch
- Stack pointer: parameters/variables not kept in registers
- Program Status Word (PSW): control bits

Pipeline vs. superscalar

Pipeline:

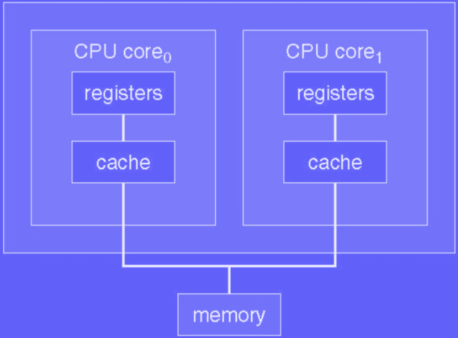
- Execute instruction n, decode $n+1$ and fetch $n+2$
- Any fetched instruction must be executed
- Issue: conditional statements



Superscalar:

- Multiple execution units
  e.g. one for float, int, boolean
- Multiple instructions fetched and decoded at a time
- Instructions held in buffer to be executed
- Issue: no specific order to execute buffered instructions

Multi-threading



CPUs and cores:

- A CPU core can hold the state of more than one thread
- A core can switch between threads in a nanosecond time scale
- The OS sees several CPUs instead of one core
- Issue: what happens if there are more than two such cores?

Clarification

Modern terminology[1]:

- CPU: computing component (the physical entity)
- Number of cores: number of independent CPUs in a computing component
- Number of threads: maximum number of instructions that can be passed through or processed by a single core
- Number of logical cores: number of cores times number of threads

[1]source: ARK

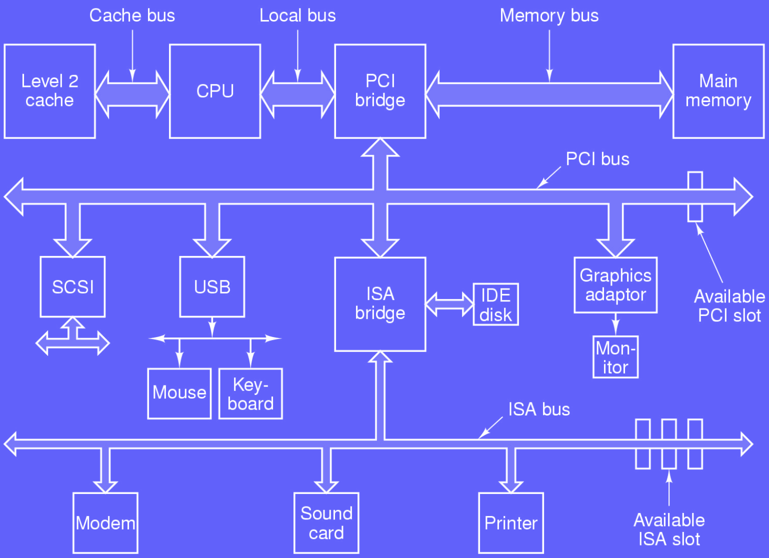
Memory

Access time		Capacity
1 ns	Registers	$<$ 1 KB
2 ns	Cache	4 MB
10 ns	Main memory	1–8 GB
10 ms	Magnetic disk	200–1000 GB
10 s	Magnetic tape	400–800 GB

Memory types:

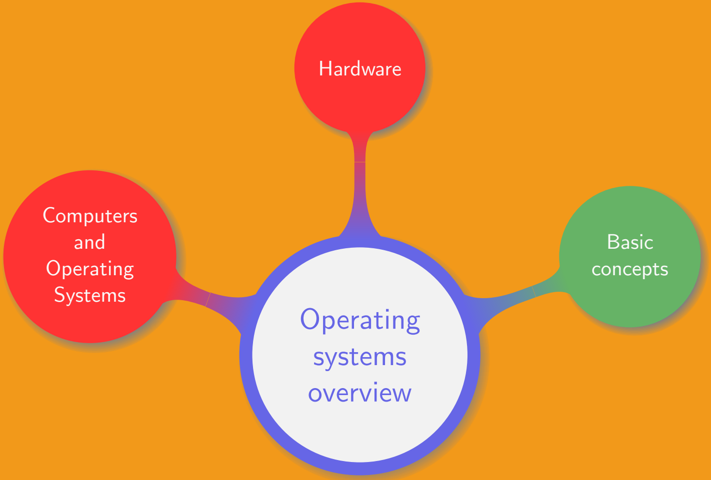
- Random Access Memory (RAM): volatile
- Read Only Memory (ROM)
- Electrically Erasable PROM (EEPROM) and flash memory: slower than RAM, non volatile.
- CMOS: save time and date , BIOS parameters
- HDD: divided into cylinder, track and sector

Bus



Cache bus
Local bus
Memory bus
Level 2 cache
CPU
PCI bridge
Main memory
PCI bus
SCSI
USB
ISA bridge
IDE disk
Graphics adaptor
Mon-itor
Available PCI slot
Mouse
Key-board
ISA bus
Modem
Sound card
Printer
Available ISA slot

Chapter organisation

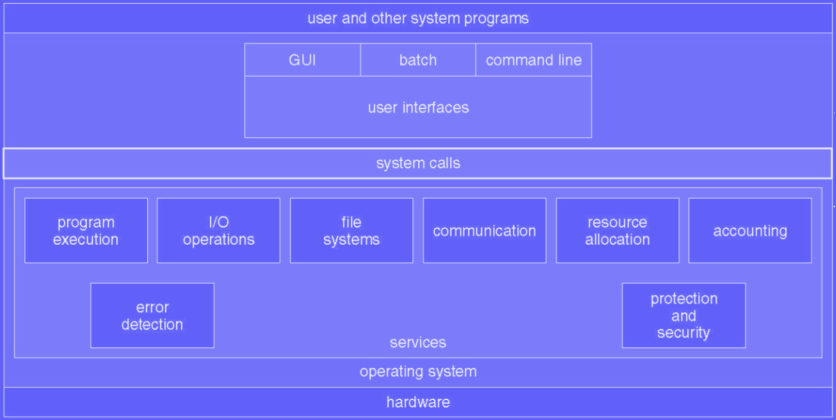


Main categories

Five major components of an OS:

- System calls: allows to interface with user-space
- Processes: defines everything needed to run programs
- File system: store persistent data
- Input-Output (IO): allows to interface with hardware
- Protection and Security: keep the system safe

System calls



Common system calls

Partial list of common Unix system calls:

- Processes: `pid=fork(); pid=waitpid(pid, &statloc, options); s=execve(name, argv, environp); exit(status);`
- Files: `fd=open(file,how,…); s=close(fd); s=stat(name,*buf); n=read(fd,buffer,nbytes); n=write(fd,buffer,nbytes); position=lseek(fd,offset,whence);`
- Directory and file system: `s=mkdir(name,mode); s=rmdir(name); mount(special,name,flags,types,args); umount(name); s=unlink(name); s=link(name1,name2);`
- Misc: `s=chdir(dirname); s=chmod(name,mode); sec=time(*t); s=kill(pid,signal);`

Processes

A process holds all the necessary information to run a program:

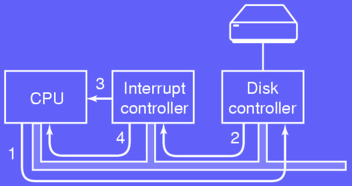
- Address space belonging to the process and containing:
  - Executable program
  - Program's data
  - Program's stack
- Set of resources:
  - Registers
  - List of open files
  - Alarms
  - List of related processes
  - Any other information required by the program

File system

The OS hides peculiarities of the disk and other IO devices

- Data stored in files grouped into directories
- The top directory is called *root* directory
- Any file can be specified using its path name
- Each process has a working directory
- Removable devices can be mounted onto the main tree
- Block files: for storage devices such as disks
- Character files: for devices accepting or outputting character streams
- Pipe: pseudo file used to connect two processes

Interrupts



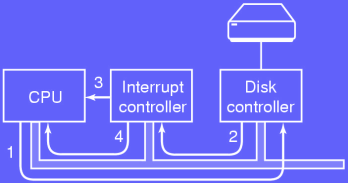
Hardware interrupt:

1. Send instructions to the controller
2. The controller signals the end
3. Assert a pin to interrupt the CPU
4. Send extra information

Software interrupt:

- A call coming from userspace
- A software interrupt handler is invoked
- System call: switch to kernel mode to run privileged instruction

Interrupts



Hardware interrupt:

1. Send instructions to the controller
2. The controller signals the end
3. Assert a pin to interrupt the CPU
4. Send extra information

Software interrupt:

- A call coming from userspace
- A software interrupt handler is invoked
- System call: switch to kernel mode to run privileged instruction

Operating systems are almost always interrupt driven

Busy waiting

Simplest method:

1. Call the driver
2. Start the input-output
3. Wait in a tight loop
4. Continuously poll the device to know its state

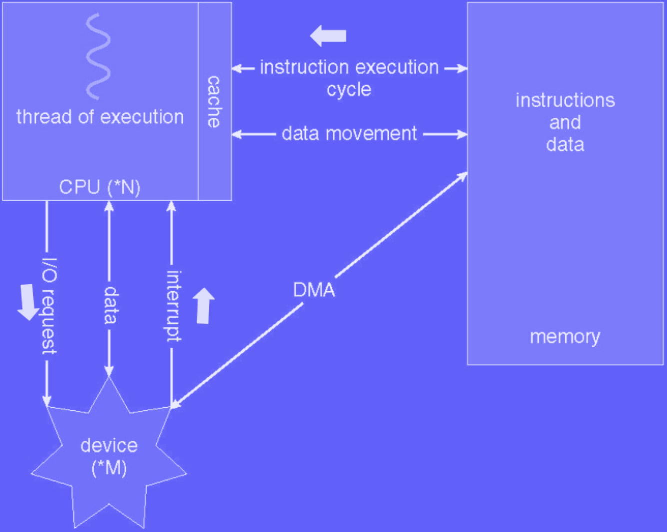
What is the drawback of this strategy?

Direct memory access

Direct Memory Access (DMA):

- Can transmit information close to memory speeds
- Directly transfer blocks of data from the controller to the RAM
- Only little needed from the CPU
- Issue a single interrupt per block, instead of one per byte

The three communication startegies



Protection and Security

CPU:

- Kernel Mode:
  - Set using a bit in the PSW
  - Any CPU instruction and hardware feature are available
- User mode:
  - Only a subset of instructions/features is available
  - Setting PSW kernel mode bit forbidden

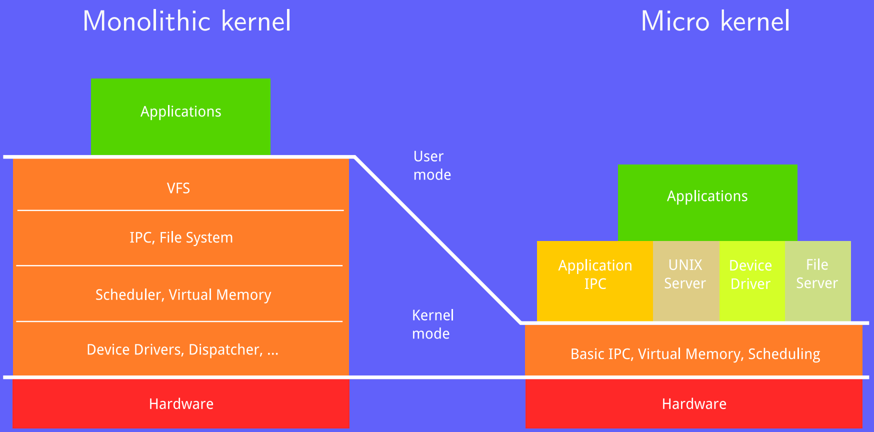
Memory:

- Base and limit registers: holds the smallest legal physical memory address and the size of the range, respectively
- Memory outside the address space is protected

Input and Output:

- They are all privilege instructions
- The OS processes them to ensure their correctness and legality

Common operating system structures



Key points

- What is the main job of an OS?
- Why are there so many types of OS?
- Why is hardware important when writing an OS?
- What are the main components of an OS?
- What are system calls?

Thank you!