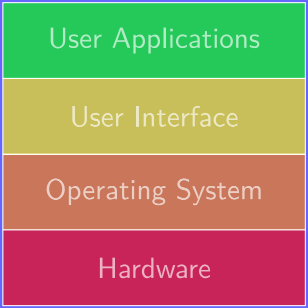
Introduction to Operating Systems
Chapter 1: Operating systems overview

Manuel

Fall 2017

Outline

Hardware and software



A computer consists of:

- Hardware
- Software
  - Kernel mode
  - User mode

Operating System

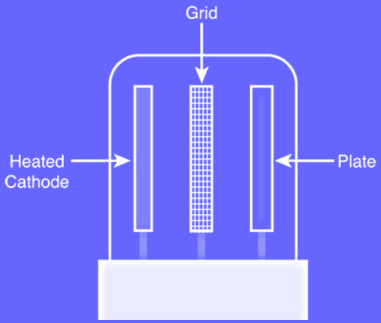
Operating System (OS)

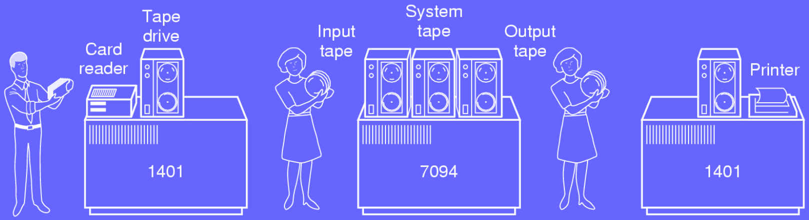
- Hardware is complicated to handle
- OS hides all the messy details
- OS manages resources for each program (time and space)
- Renders computer much easier to use

A bit of history

The first 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
- Program wired on the 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:

- 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

Outline

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

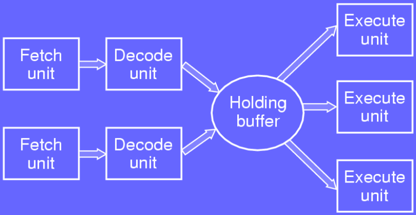
Pipeline vs. superscalar

Pipeline:

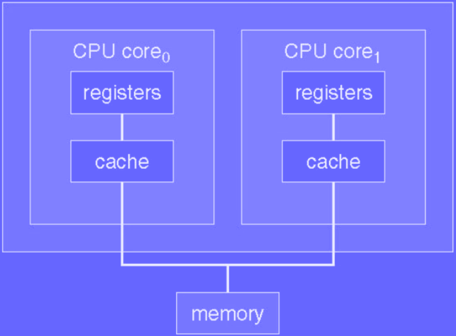
- Execute instruction n, decode $n+1$ and fetch $n+2$
- Any fetched instruction must be executed
- Issue: conditional statements

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



- 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

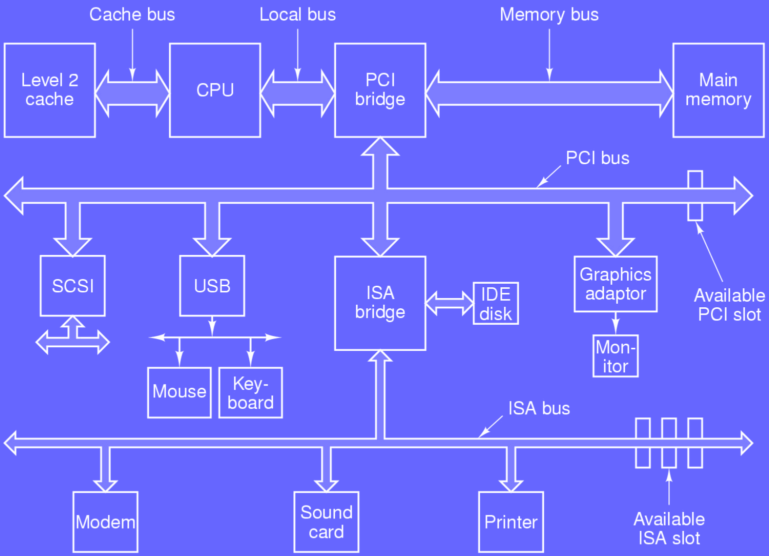
[1]source: ARK

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

- 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
Available PCI slot
Mon-itor
Mouse
Key-board
ISA bus
Modem
Sound card
Printer
Available ISA slot

Outline

Main categories

Five major components of an OS:

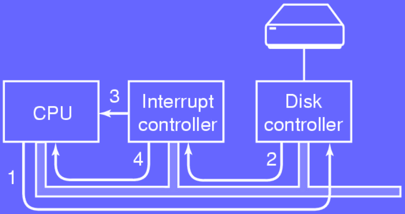
- Input/Output
- Protection/Security
- Processes
- File system
- System calls

First communication strategy

Using the device controller:

- I/O devices and CPU can execute concurrently
- Device controller in charge of a particular device
- Drive controllers have a buffer
- Buffer used to move data to/from buffer to/from memory
- Device controllers interacts with the CPU using interrupts

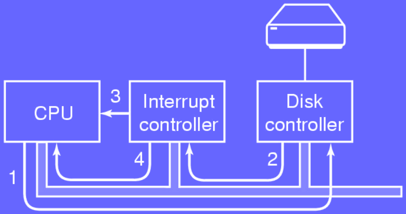
Interrupt



Starting an I/O device and getting an interrupt:

1. Send instruction to controller
2. Controller signals the end to interrupt controller
3. Assert pin to interrupt the CPU
4. Send extra information

Interrupt



Starting an I/O device and getting an interrupt:

1. Send instruction to controller
2. Controller signals the end to interrupt controller
3. Assert pin to interrupt the CPU
4. Send extra information

Processing an interrupt:

1. Push user program counter and PSW onto the stack
2. Switch to kernel mode
3. Find the device interrupt handler in the interrupt vector
4. Query the device for its status
5. Return to user program (1st instruction not yet executed)

Interrupts

Remarks of interrupts:

- Incoming interrupts are disabled during the process
- Softwares can generate interrupts: trap
  e.g. java exception, division by 0...
- An OS is almost always interrupt driven

Second communication strategy

Simplest method:

1. Call the driver
2. Start the I/O
3. Wait in a tight loop
4. Continuously poll the device to know its state

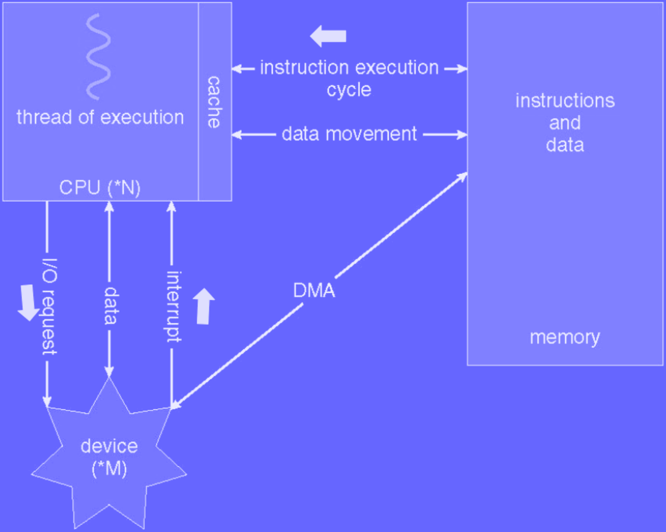
Disadvantage: CPU **busy waiting** until device is finished

Third communication strategy

Direct Memory Access (DMA):

- High speed I/O devices
- Transmit information close to memory speeds
- Device controller directly transfers the blocks of data from the buffer to the main memory
- No help from the CPU
- One interrupt per block, instead of one per byte

Summary


thread of execution
CPU (*N)
cache
instruction execution
cycle
data movement
instructions
and
data
memory
I/O request
data
interrupt
DMA
device
(*M)

Protection/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 smallest legal physical memory address and size of range, respectively
- Memory outside the address space is protected

Input/Output:

- All I/O instructions are privilege instructions
- OS treats I/O operations to ensure correctness and legality

Processes

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

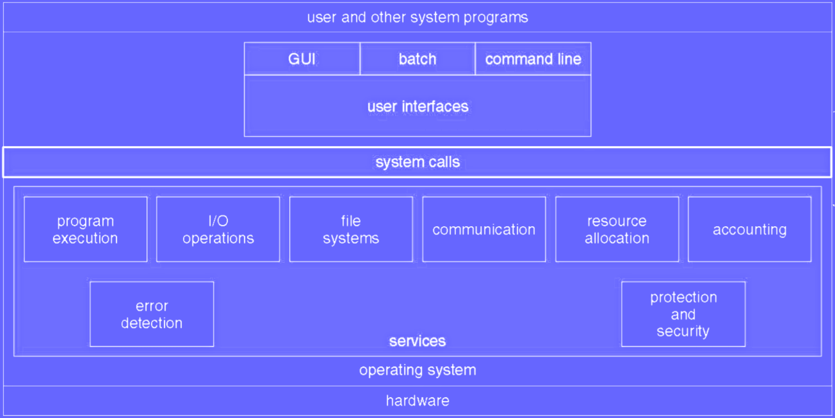
- Address space belonging to the process and containing:
  - Executable program
  - Program's data
  - Program's stack
- Set of resources:
  - Registers
  - List of open files
  - Alarms
  - List of related processes
  - Any other information required by the program

File system

OS hides peculiarities of the disk and other I/O devices

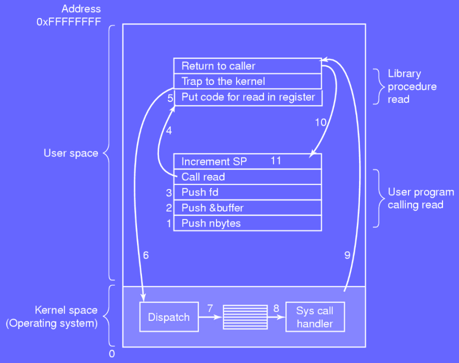
- Data stored in files
- Files are grouped inside directories
- Top directory is called root directory
- Any file can be specified using its path name
- Each process has a working directory
- Before reading/writing in a file permissions are checked
- If access is granted a file descriptor is returned
- Removable devices can be mounted on the main tree
- Block special files: represent devices such as disks
- Character special files: represent devices that accept or output character stream
- Pipe: pseudo file used to connect two processes

System calls



Simple example

```
ssize_t read(int fd, void *buf, size_t count);
```

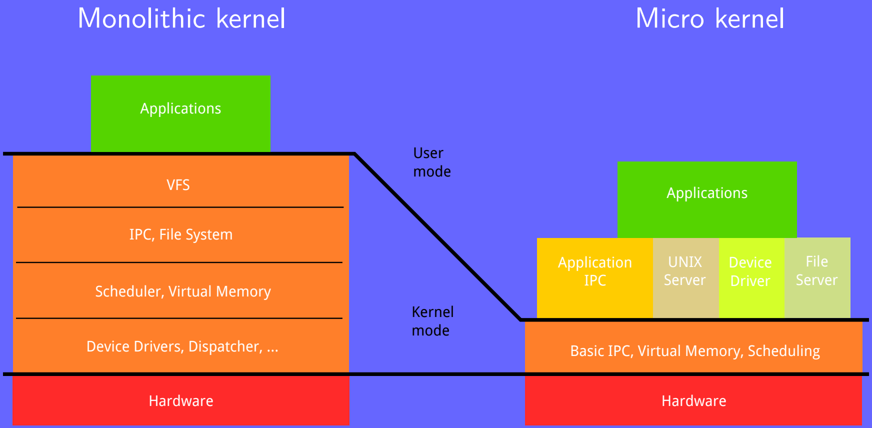


- 1-3: push parameters on the stack
- 5-6: switch to kernel mode
- 7: use a table of pointers to system calls handler to dispatch to the correct one
- 11: increment stack pointer to remove parameters pushed (1-3)

Common 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); umount(name);`
  `s=unlink(name); mount(special,name,flags,types,args);`
  `s=link(name1,name2);`
- Misc:
  `s=chdir(dirname); s=chmod(name,mode); sec=time(*t);`
  `s=kill(pid,signal);`

OS structure



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!