ECS 150 - OS Introduction

Prof. Joël Porquet-Lupine

UC Davis - 2020/2021

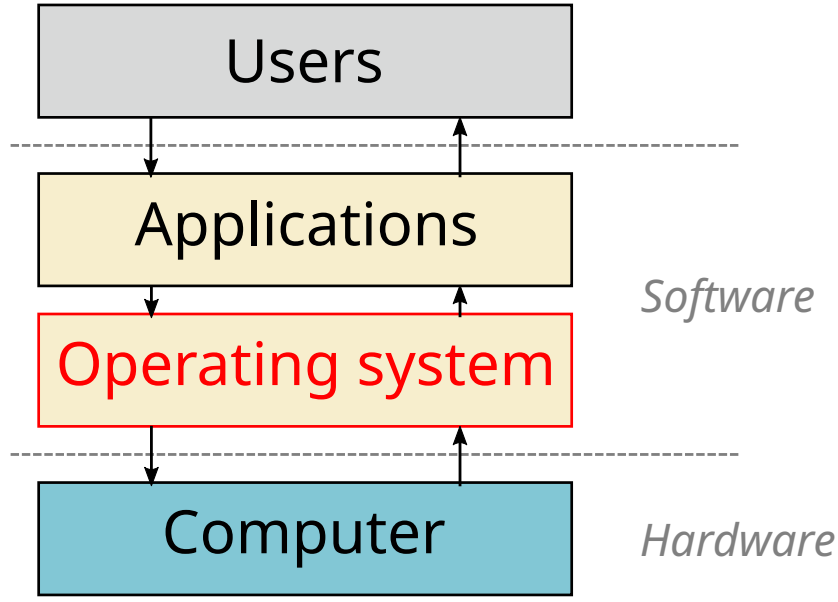
Operating system definition

Used to be a bit blurry

- Everything that was shipped by the operating system vendor
- But prone to abuse (e.g., US vs Microsoft, 98)

(Somewhat) clearer definition

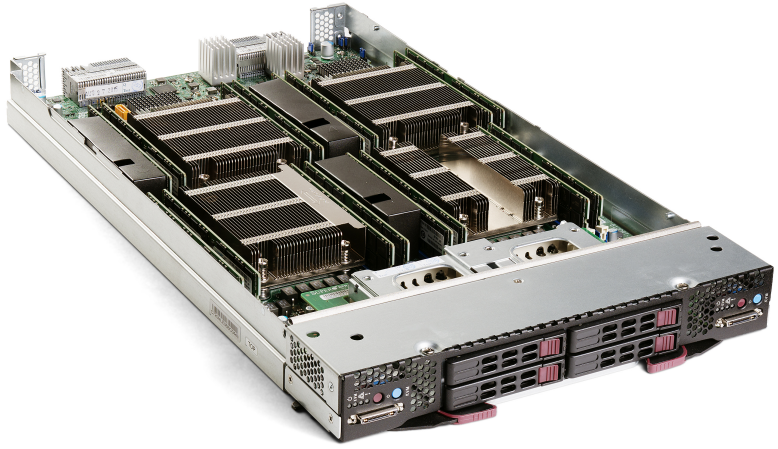
An operating system (OS) is the layer of software that manages a computer's resources for its users and their applications.



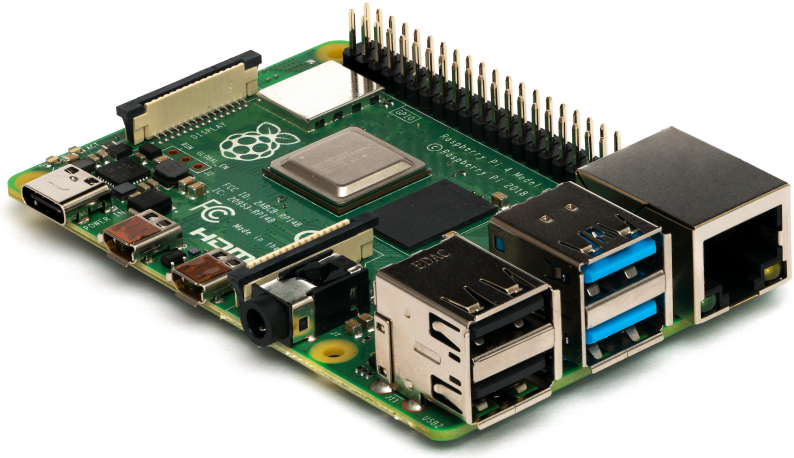
Computer organization

Types of computers

Desktop computer



Blade server (by Dmitry Nosachev - CC BY-SA 4.0)

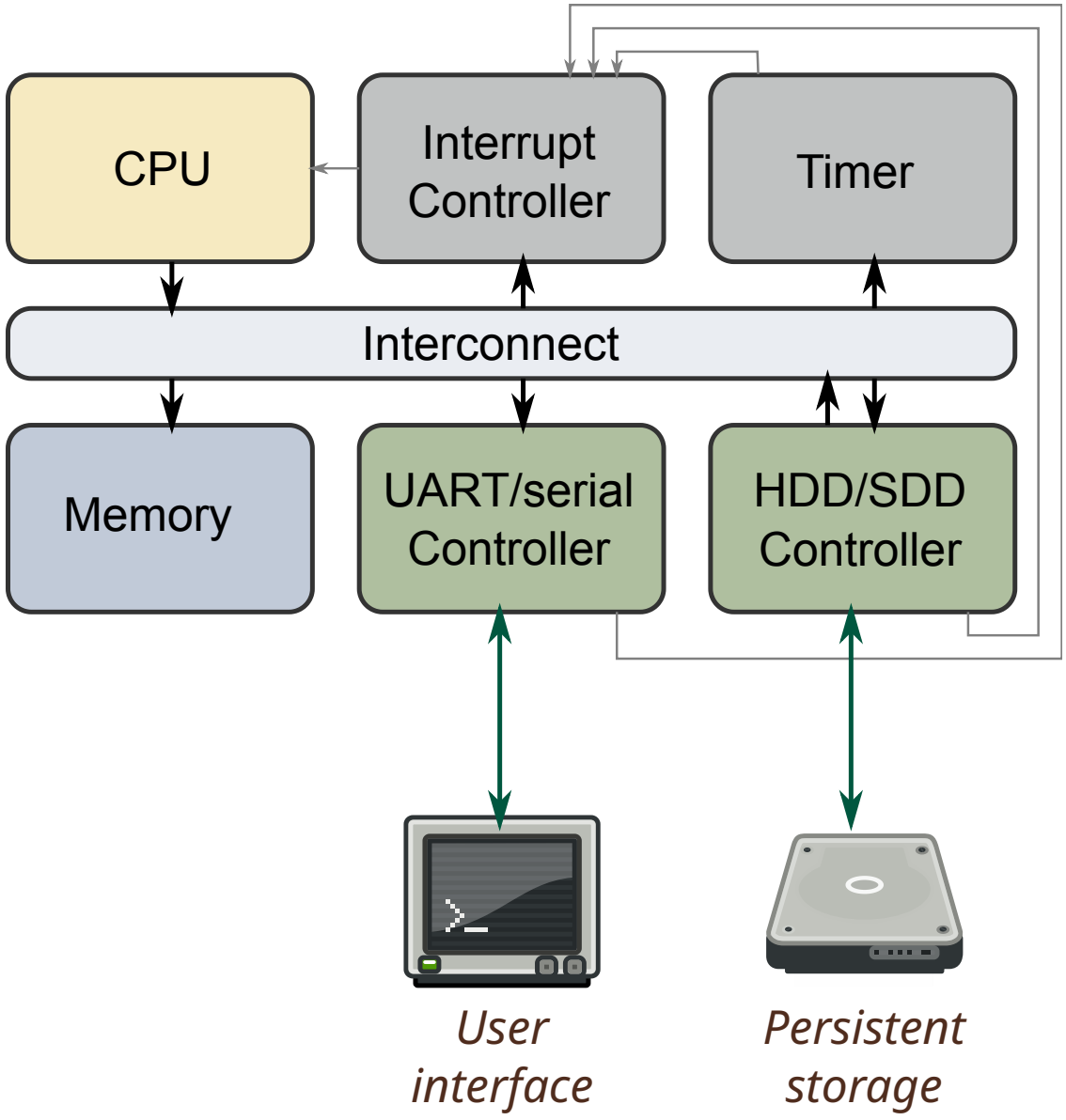


Internet of Things (by Miiicihiaieil Hieinizilieir, CC BY-SA 4.0)

Car computer

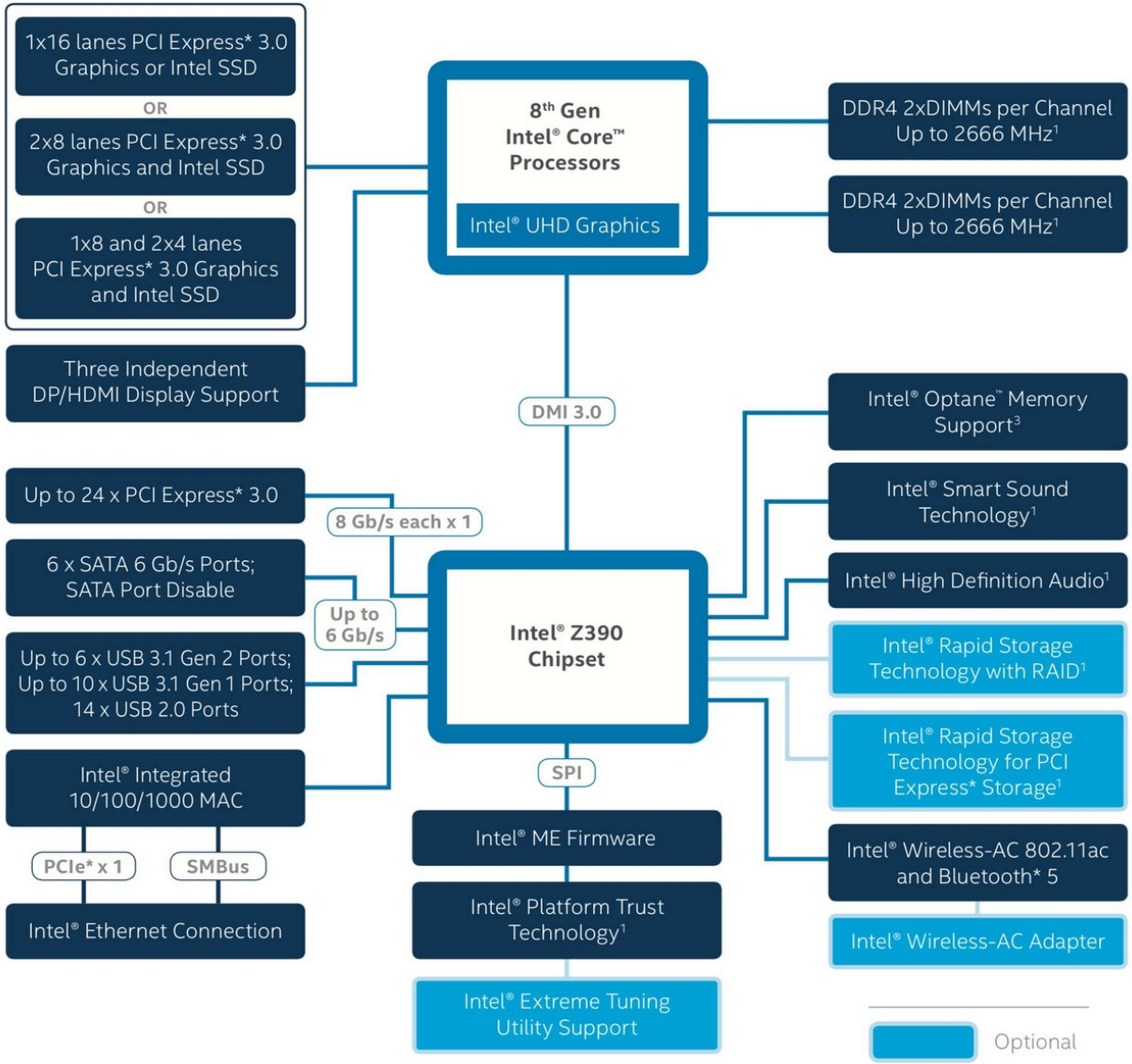
Computer organization

The bare minimum



Computer organization

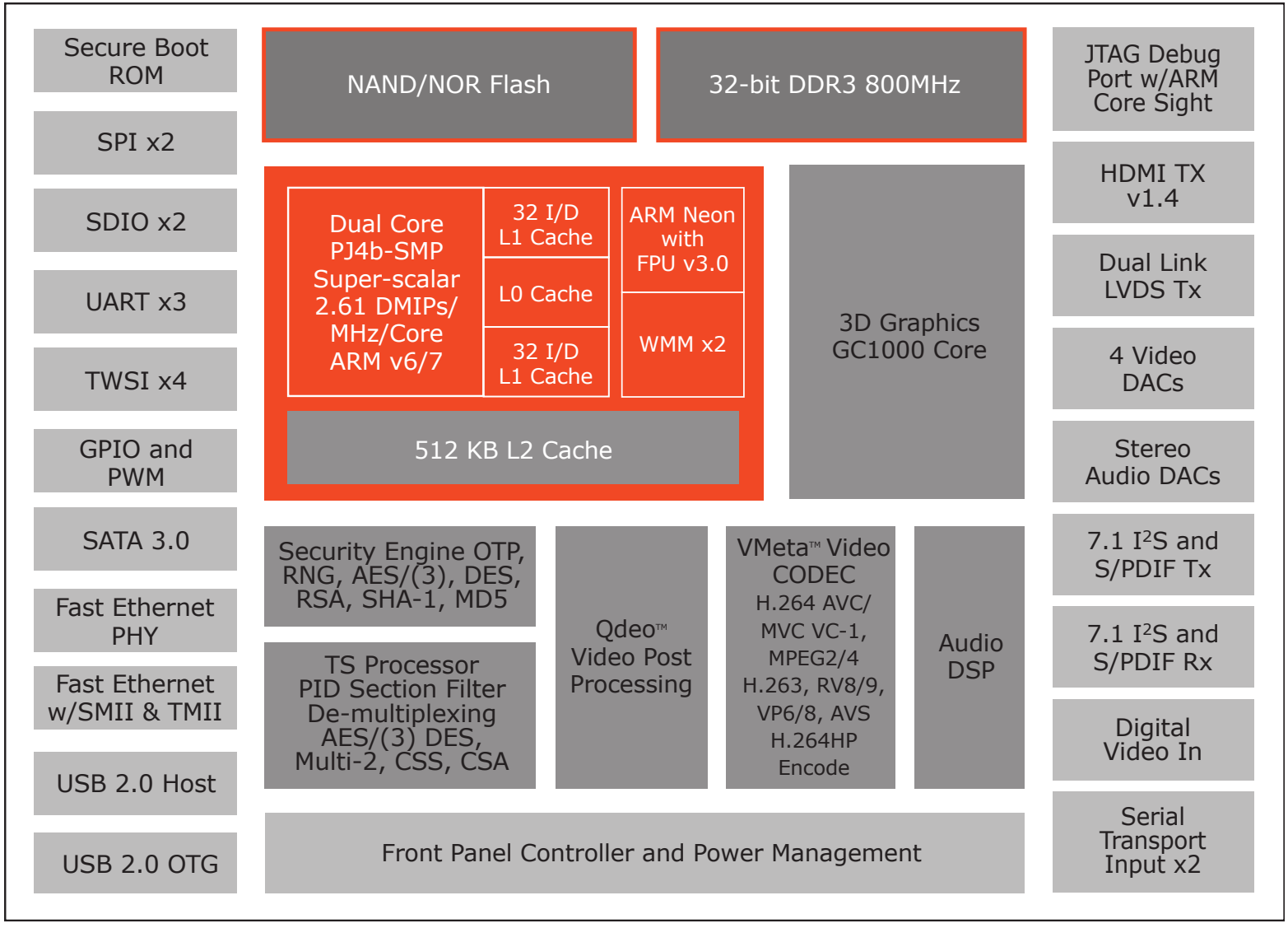
General-purpose computing



Intel Chipset Z390 (2018)

Computer organization

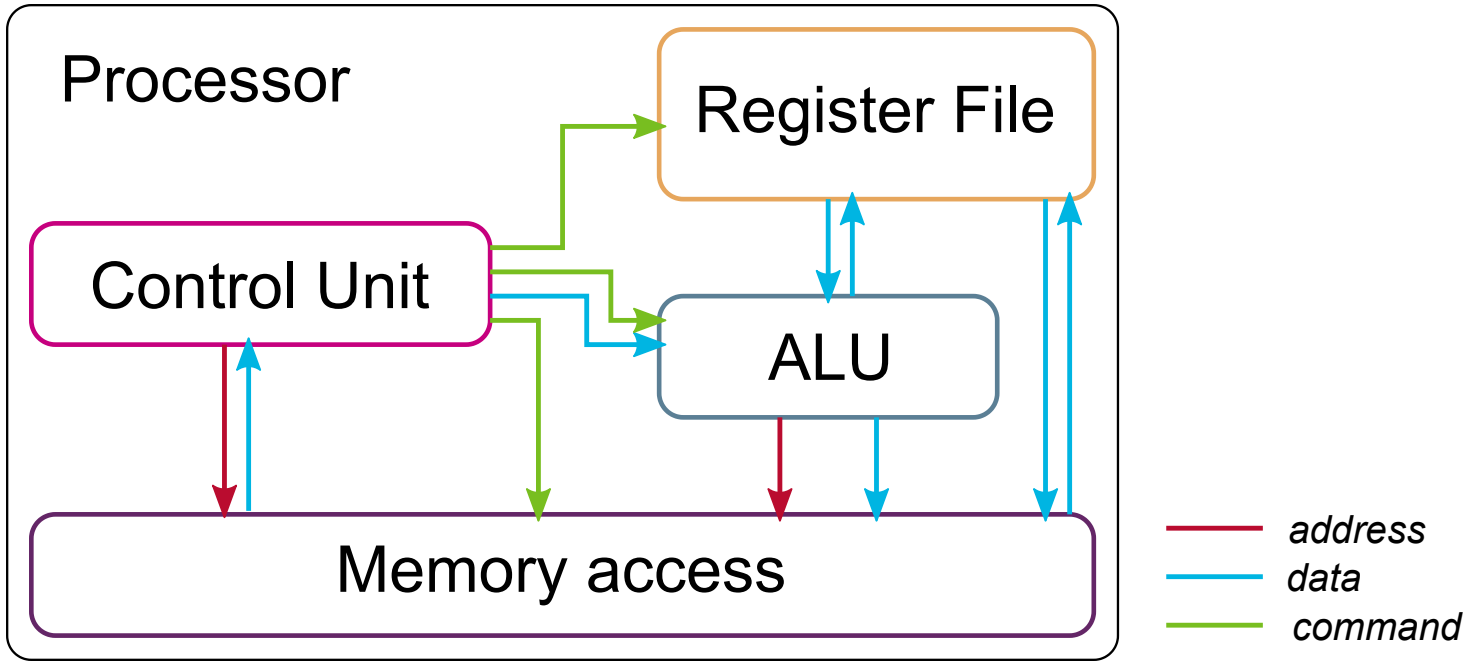
Embedded computing



Marvell Armada 1500 (similar to SoC found in Google Chromecast)

Hardware components

Processor

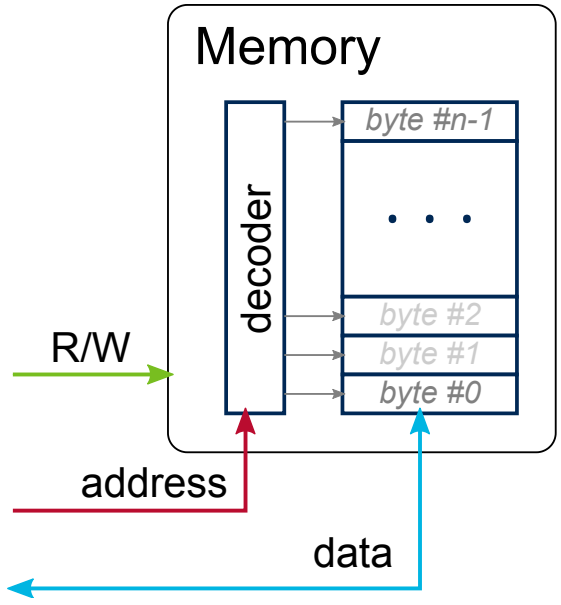


- Fetches instructions from memory, decodes them, and executes them
- Characterized by an *Instruction Set* (e.g., `i686`, `x86_64`, `AArch64`)
  - Access to memory, arithmetic/logical operations, control flow
- Contains a set of *registers*
  - General-purpose registers, program counter, status register

Hardware components

Memory

- Set of **addressable** bytes that can hold numerical values



- Organized in a hierarchy of layers (with various access times)

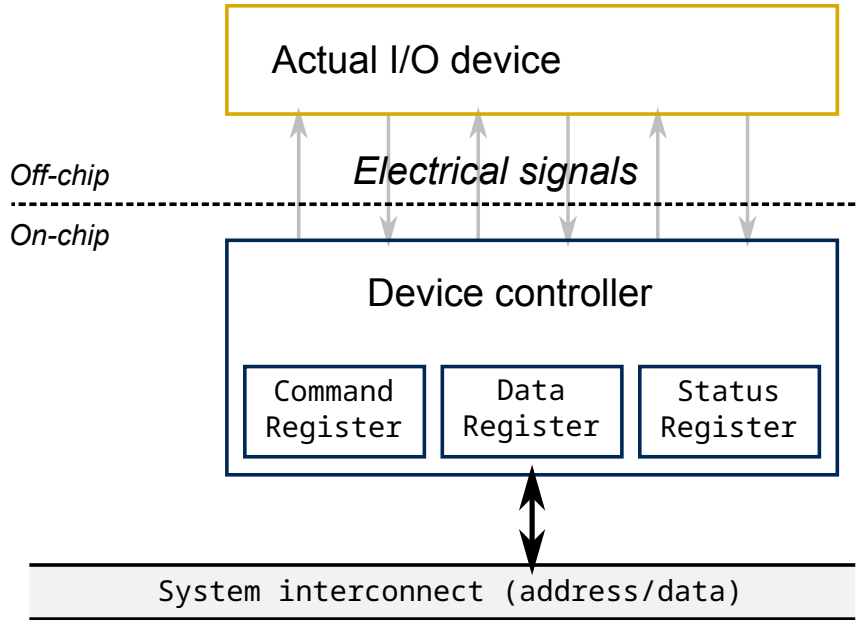
Memory layer	Access time	Accessibility
CPU registers	Immediate	Via CPU instructions
CPU cache	Few cycles	Not addressable (transparent)
RAM/Main memory	Few hundred cycles	Addressable from CPU
Second-level storage	Few thousand cycles	Indirectly addressable (see later in course)

Hardware components

I/O Devices

- Controllers connected to system interconnect
- Devices connected to controllers

- The controller provides an interface for accessing the device resources/functionalities
  - Via device registers



- Memory-mapped access
  - Device registers mapped into memory address space
  - Accessible through regular memory instructions

- Port-mapped access
  - Device registers mapped into special I/O space
  - Accessible through special I/O instructions

Hardware components

Interconnects (buses, networks)

- Transfer data between components
- Characterized by various features, speed, bandwidth, etc.

CPU to memory buses

- Cache bus (aka Back side bus in Intel lingo)
- Memory bus (aka Front side bus in Intel lingo)
  - Now implemented by AMD HyperTransport, or Intel QuickPath Interconnect

CPU to device buses

Bus	Created	Bandwidth	Type
ISA	1981	~8 MiB/s	Expansion
IDE	1986	~8 MiB/s	Mass-storage
PCI	1992	~133 MiB/s	Expansion
AGP	1997	~266 MiB/s	Video card
SATA 1.0	2000	~150 MiB/s	Mass-storage
PCI-e 1.x	2003	~250 MiB/s per lane	Expansion/Video card
PCI-e 5.x	**2019**	**~8 GiB/s per lane**	**Expansion/Video card**

OS definition, part II

Roles

In order to manage a computer's resources, an OS needs to play various roles:

- Referee
- Illusionist
- Glue

Design principles

A well-constructed OS needs to achieve various design goals:

- Reliability
- Security
- Portability
- Performance

OS roles

Referee

Manage the **resource sharing** between applications

- Resource allocation (e.g., CPU, memory, I/O devices)
- Isolation (e.g., fault isolation)
- Communication (e.g., safe communication)

Illusionist

Abstraction of hardware via **resource virtualization**

- Mask scarcity of physical resources
- Mask potential hardware failure

Glue

Set of **common services** to applications

- Hardware abstraction
- Filesystem, message passing, memory sharing

OS design principles

Reliability

- OS does exactly what it is designed to do
- Related to *availability*: OS is always usable

Security

- OS is secure if cannot be compromised by a malicious attacker
- Related to *privacy*: data is only accessible by authorized users
- Enforcement mechanisms vs security policies

Portability

- OS provides the same abstractions regardless of the underlying hardware
  - Applications
  - System libraries
  - Kernel

> *Binary compatibility is **so** important that I do not want to have anything to do with kernel developers who don't understand that importance. [...] The **only** reason for an OS kernel existing in the first place is to serve user-space.*
>
> *Linus Torvalds, LKML, 2012*

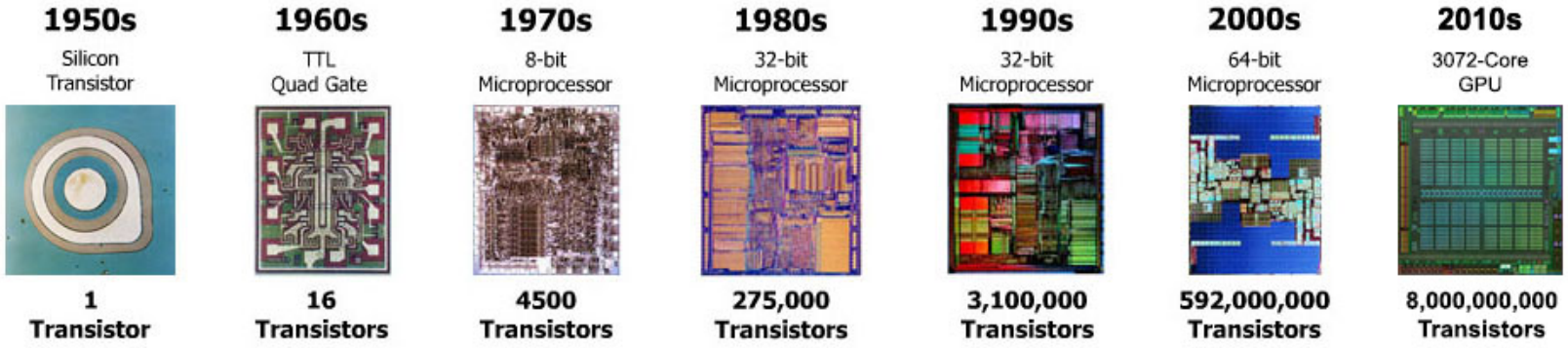
Performance

- Overhead
- Fairness
- Latency
- Throughput
- Predictability

OS history

Three major phases

- Hardware is very expensive (1955-65)
- Hardware slowly becomes affordable (1965-80)
- Hardware becomes dirt cheap (1980-present)



OS history

Hardware is very expensive (1955-65)

- Introduction of the transistor in the mid-50s
- Expensive mainframes, operated by humans
- Read from punch card, run job, print result
- Batch systems in order to better serialize jobs

IBM 7090, circa 1960

Software

- OS is simply a runtime *library* (common I/O functions)
- Applications have full access to hardware

OS history

Hardware slowly becomes affordable (1965-80)

- Use of integrated circuits
- Mainframe computers
  - E.g., IBM System/360

- Personal computers
  - E.g., DEC PDP-11

Software

- IBM OS/360
  - Multiprogramming, memory protection
- Multics
  - Timesharing, dynamic linking, security, hierarchical file-system
- UNIX, BSD/SystemV variants
  - C language
- POSIX

OS history

Hardware becomes dirt cheap (1980-present)

- Continuation of Moore's law
- Personal computers
  - Command line interfaces
  - GUI
- Pervasive computers
  - Desktops, laptops
  - Smartphones, tablets
  - Embedded systems
  - Data centers

OS history

Progression over time

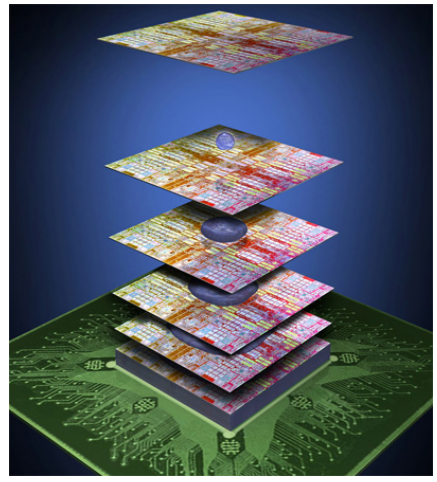
Metric	1981	1997	2014	Factor(2014/1981)
CPU performance (MIPS)	1	200	2500	2.5 K
CPU/computer	1	1	10+	10+
MIPS cost	$100K	$25	$0.20	500 K
DRAM (MiB)/$	$500	$0.5	$0.001	500 K
Disk (GiB)/$	$333	$0.14	$0.00004	8 M
WAN (bps)	300	256K	20M	60 K
LAN (bps)	10M	100M	10G	1000+
Ratio users to computers	100:1	1:1	1:several	100+

Numbers taken from OSPP textbook

Future of OSes

End of Moore's law

- Make better use of the same area
  - Deep redesign of processors?
- 3-D stacking
  - Multiprocessors
  - Multi- and many-cores
- Quantum computing



From the very small... to the super big

- Power-efficient IoT devices
  - Smart home, smart city, smart [blank]
- Giant data centers
  - Very large-scale storage

Heterogeneity

- Different processors on same chip
  - E.g., ARM big.LITTLE, Intel Hybrid Technology
- Specialized computing accelerators
  - GP-GPUs, FPGAs, AI accelerators