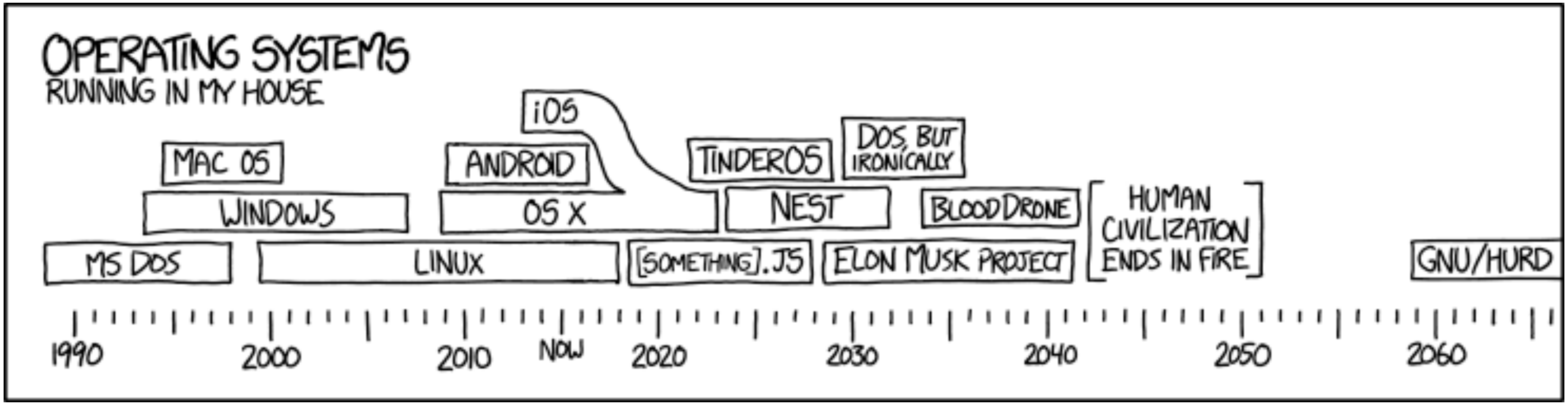
University of Glasgow | School of Computing Science

Networks & Operating Systems Essentials

Dr. Angelos Marnerides
<*angelos.marnerides@glasgow.ac.uk*>
School of Computing Science

Coming up next...



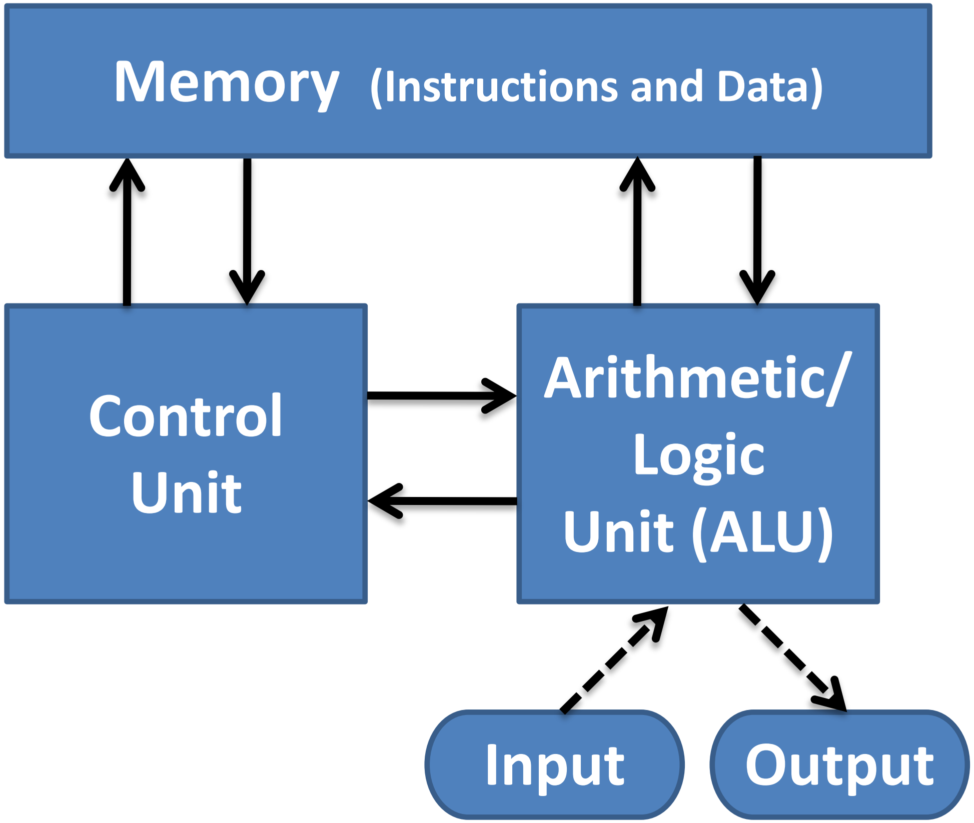
Source: https://xkcd.com/1508/

What is going to be covered?

- Processes
  - Process management
  - Scheduling
- Memory
  - Virtual memory
  - Page tables
- Storage
  - Block storage
  - File systems (plus case studies)

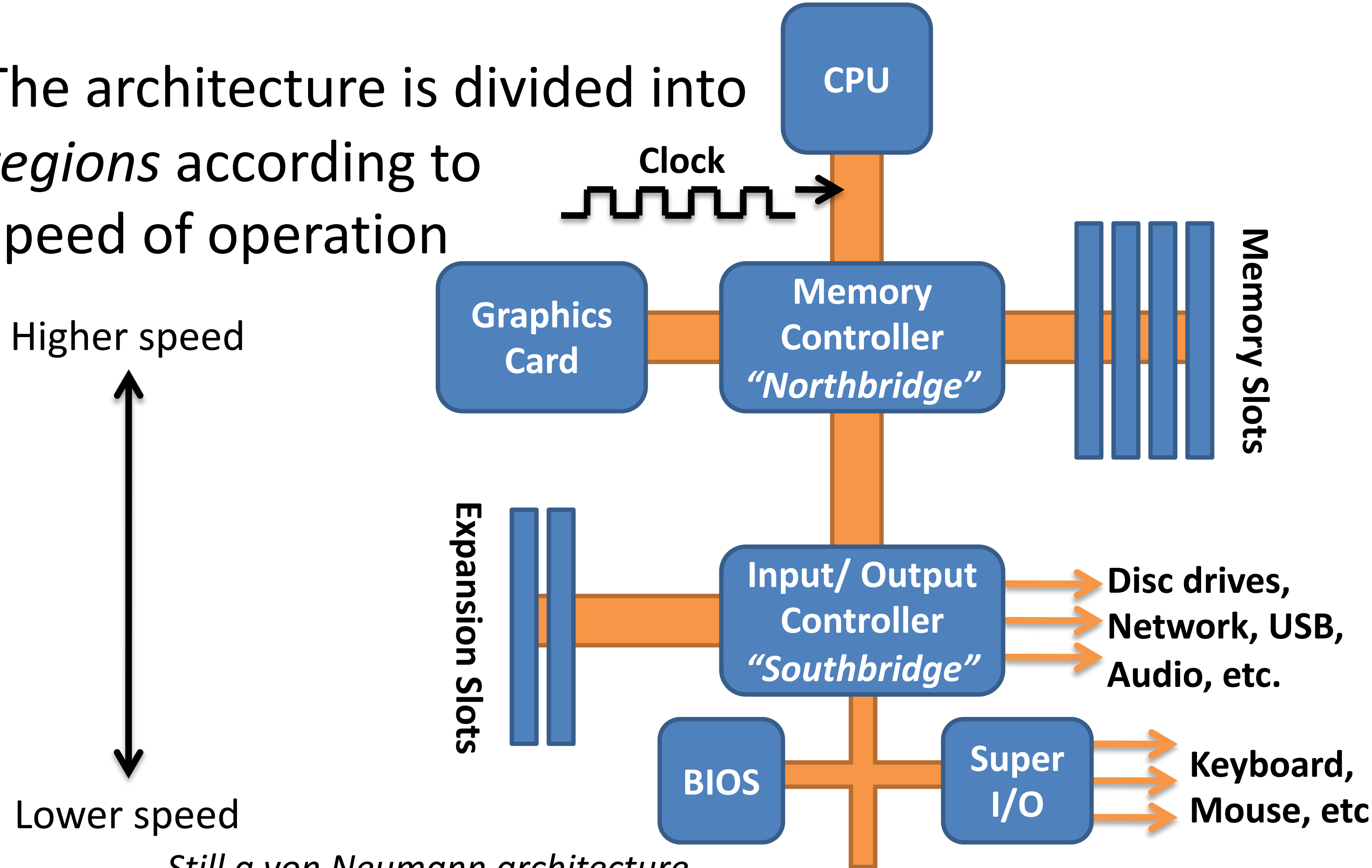
Elements of computer architecture

- Today, computer architecture is largely standardised, at a high level of abstraction, on the *von Neumann Architecture*



A (fairly) modern PC architecture

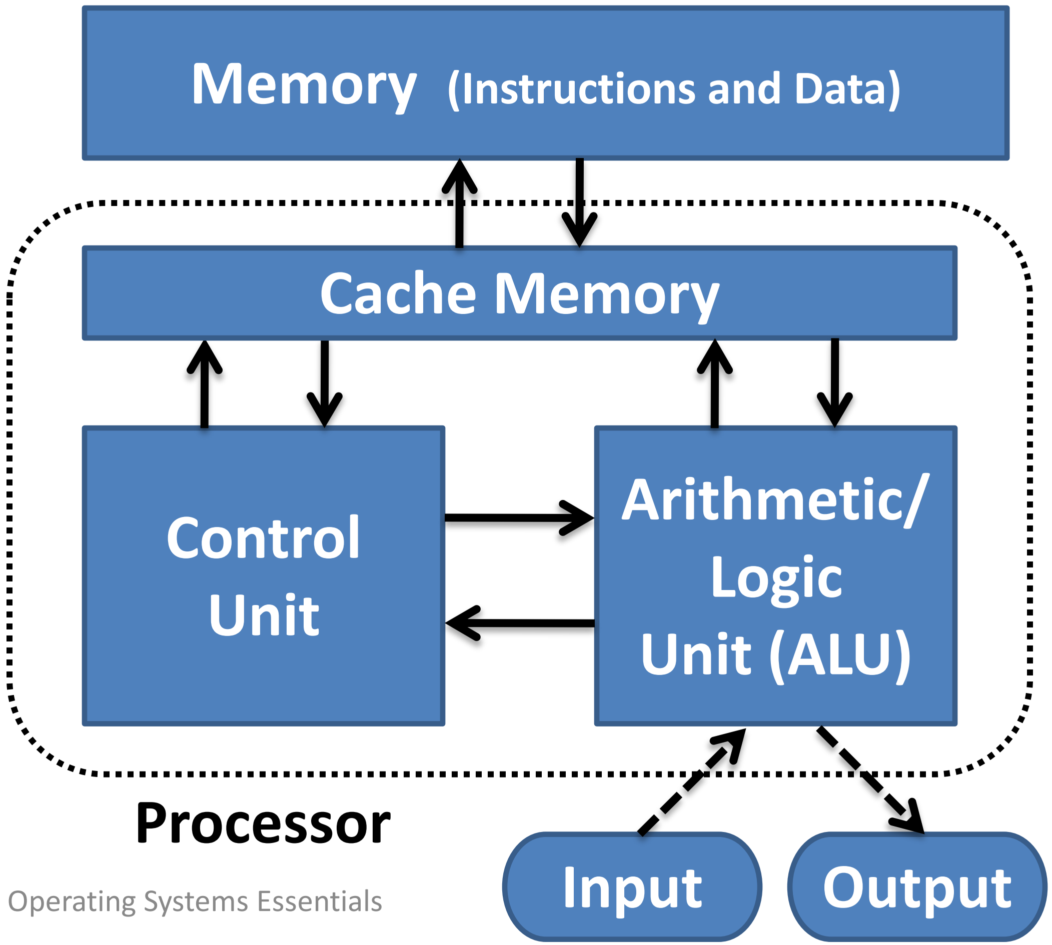
- The architecture is divided into *regions* according to speed of operation



Still a von Neumann architecture

The processor

- Often also referred to as the Central Processing Unit (CPU)
- ALU + Control Unit;
often also contains some internal high-speed *cache memory*
- Note, this is still logically the same picture as on the previous slide - still a von Neumann architecture



Instruction Set Architecture (ISA)

– The view of the processor that is seen by programs being executed by that processor

```
muli $2, $5, 4
add $2, $4, $2
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
jr $31
```

- What is the set of available instructions?
- What is the set of available registers?
- How many operands do instructions need?
- What are the sizes and types of the operands?
- How are operands accessed (e.g., stack-based ISAs don't support random access to memory – see next slide)?
- How many operands can be in registers (vs. in memory)?
- How many clock ticks does it take to execute an instruction?
- …

Registers

- We've looked at instructions, now let's look more at *registers*…

- Registers are *holding areas for data being worked on inside the CPU*
  - Often, arithmetic and logic instructions are designed to work *on registers only*, not on main memory
    - This is because registers are much *faster* than main memory
    - We use the data-transfer instructions for register ↔ main memory transfers

- General purpose and special registers
  - The registers used by the arithmetic and logic instructions are called **general purpose registers**
    - In computers like MIPS we have a largish set of these—a "file" of 32 x 32-bit registers
      - Because its registers are 32 bits wide, MIPS is said to have a *word size* of 32 bits
      - This is also the size of the basic unit of transfer between the registers and main memory (although main memory can also be accessed at the granularity of half-words and individual bytes)
  - Processors also have **special registers** such as the *program counter* (PC) and the *stack pointer* (SP)…

How your code is executed

Level 3 OS Level

Operating System

Level 2 ISA Level

Our Focus!

High-level language program (in C)

```
swap(int v[ ], int k)
{int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compiler

Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Assembly / Microprogam

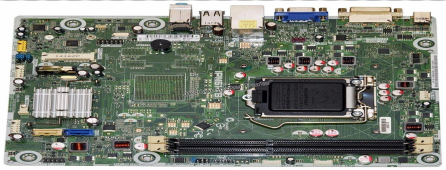
Assembler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Level 1 Microarchitecture Level

Hardware / Logic Circuits



Level 0 Physical Level

What is your favourite OS?

Mac OS X

solaris

What do you expect your OS to do for you?

- A program that acts as an intermediary between a user of a computer and the computer hardware (through the ISA)
- Goals:
  - Make the computer system usable
  - User to be able to execute programs as per needs
  - Use of computer hardware efficiently

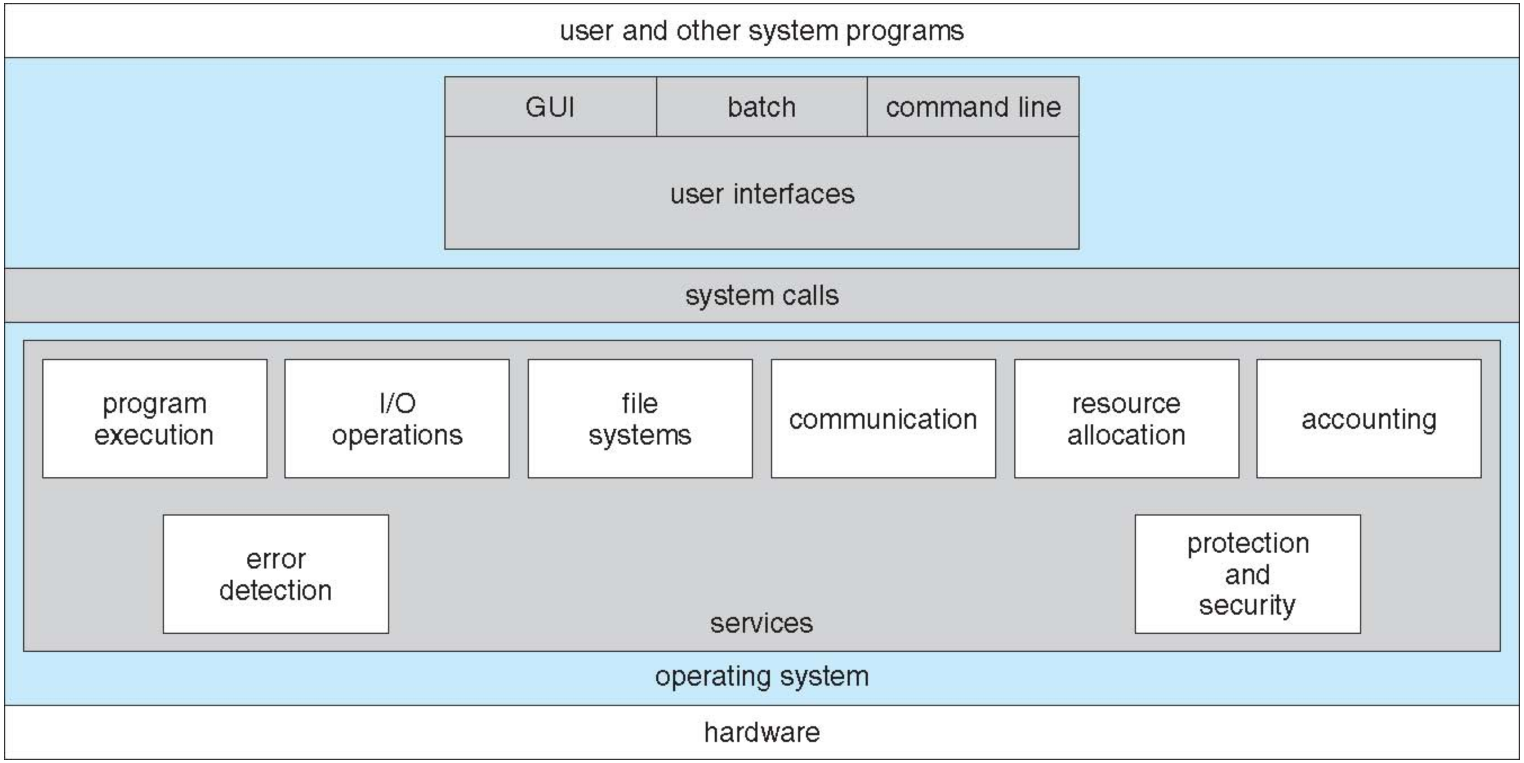
Operating System Roles

- OS acts as a resource allocator:
  - Manages all resources
  - In case of conflicting requests ensures efficient and fair resource use
- OS acts as a control program:
  - Try to prevent errors and improper use of the computer system

Core components

- Kernel
  - "The program running at all times on the computer; core part of the Operating System"
  - Everything else is either a system program or an application program
- Bootstrap program
  - Also known as firmware, BIOS, etc.
  - Loaded at power-up or reboot
  - Typically stored in ROM (Read Only Memory)
  - Initializes all aspects of system
  - Loads operating system and starts execution

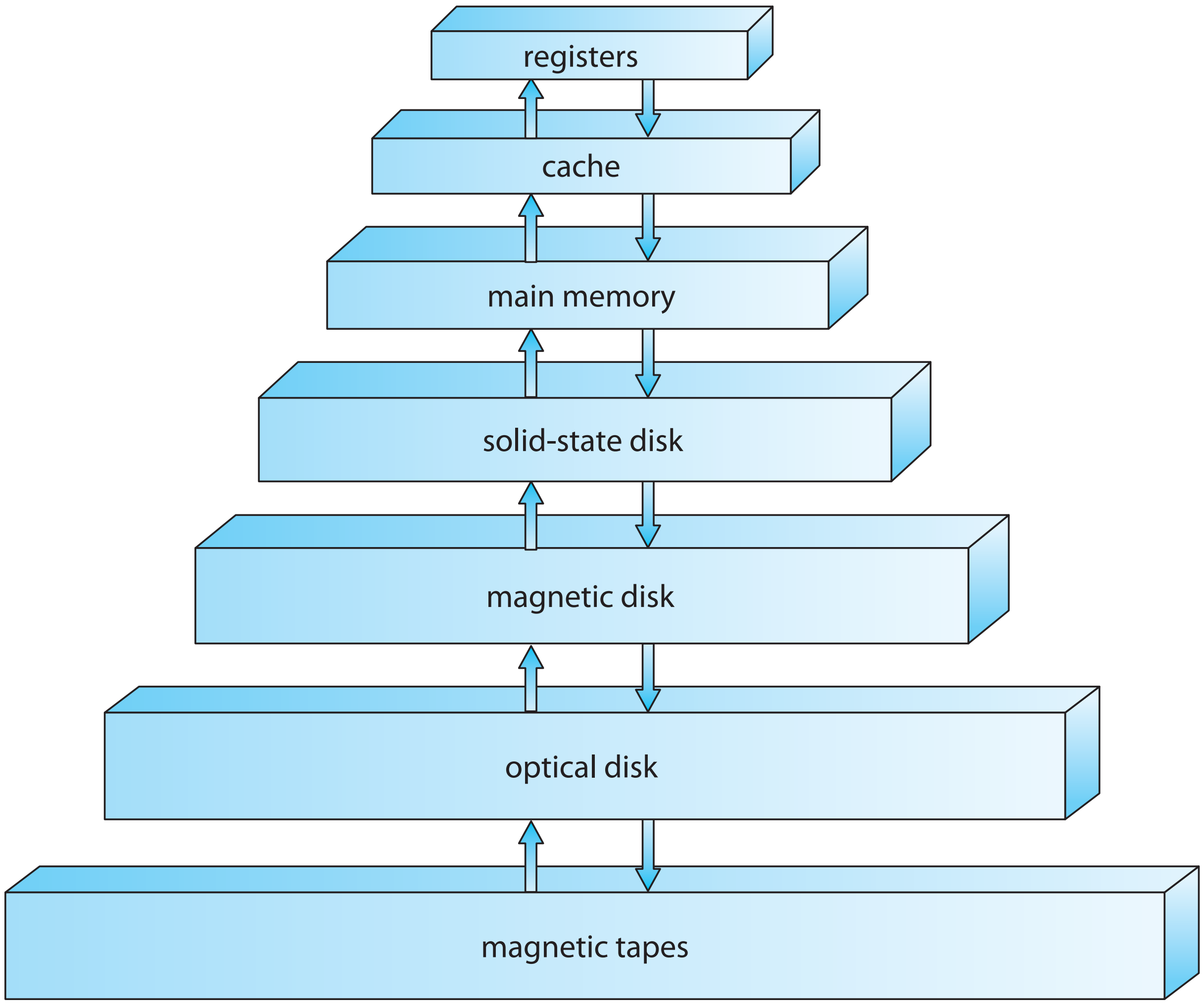
Operating System Services

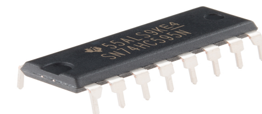


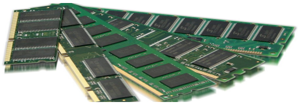
Memory/Storage Systems characteristics

- Storage systems organized in hierarchy
  - Speed – Cost – Availability & Ruggedness
- Caching
  - Copying information into faster storage system

Memory/Storage-Device Hierarchy







Aside: Levels of Storage

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

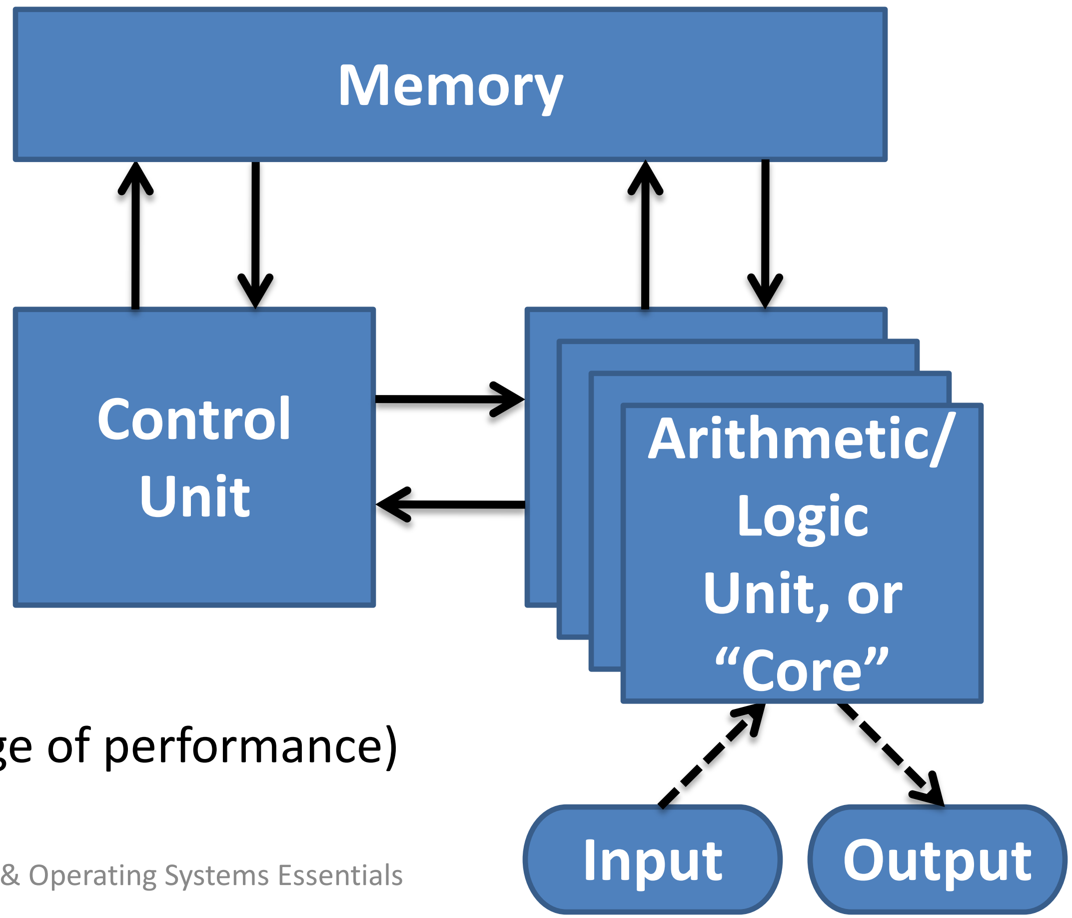
Aside: System Latencies

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 μs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

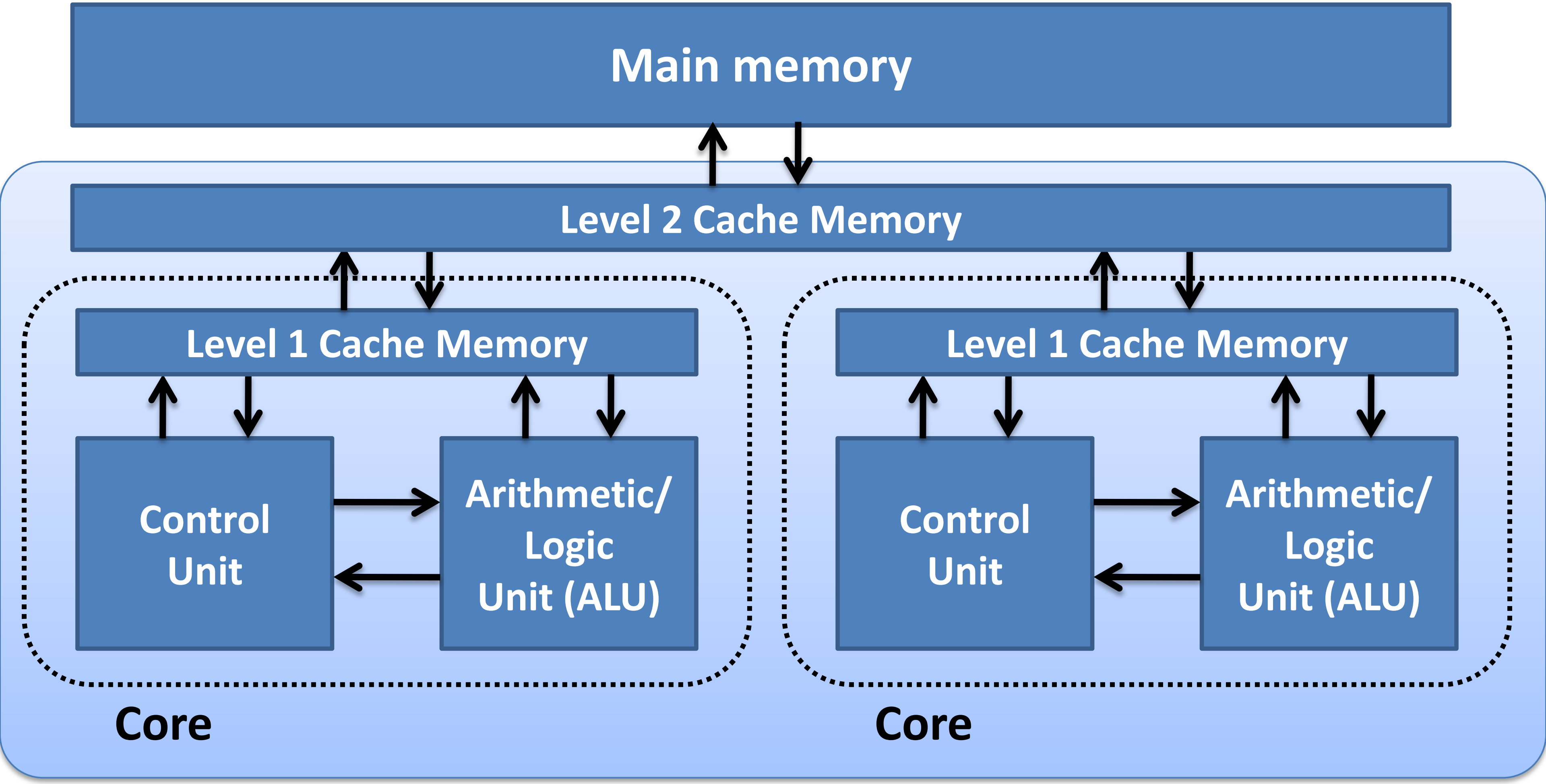
Source: B. Gregg, "Systems Performance: Enterprise and the Cloud", Prentice-Hall, 2013.

As performance demands increase…

- We see recent renewed interest in *parallel* architectures
  - – Faster, although
    - They complicate system software
    - We are not always able to hide the complexity from application software (esp. to take advantage of performance)



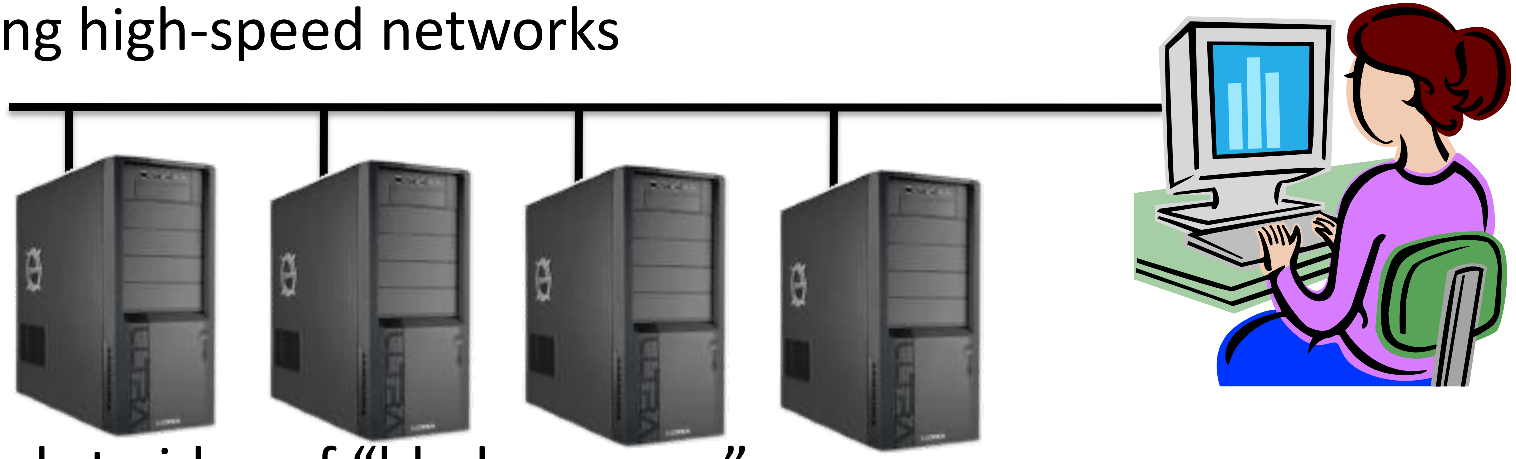
Another multi-core architecture



A dual-core processor

Coarser-grained parallelism: clustering

- We can increase performance by linking computers using high-speed networks



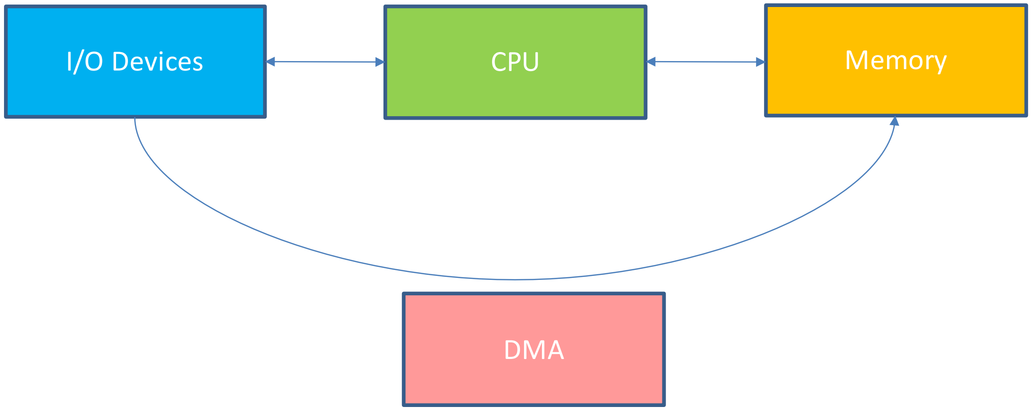
- Leads to idea of “blade servers”
  - Obviously they don’t all need screens, etc.
- Applications run across the cluster (ideally)
  - Although, as mentioned earlier, some applications can’t easily be decomposed in this way

Computer System Organisation

- One or more CPUs, device controllers → access to shared memory
- CPUs and devices competing for memory cycles
- Each device controller is in charge of a particular device type and has a local buffer
  - CPU moves data from/to main memory to/from local buffers
  - I/O communication from the device to local buffer of controller.
  - Device controller informs CPU that it has finished its operation by causing an interrupt

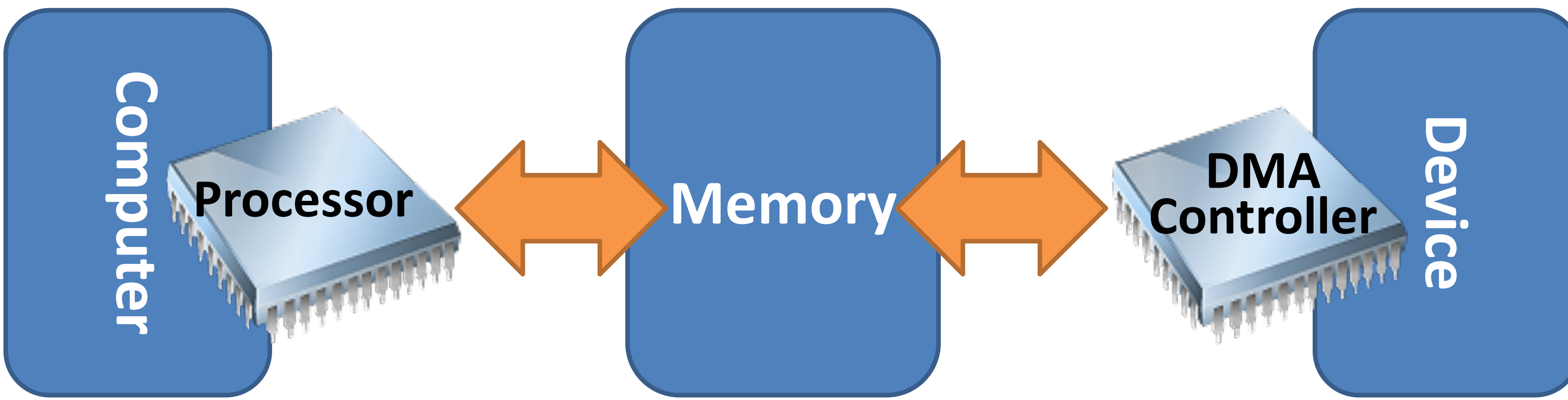
Direct Memory Access

- What happens then when an IO request needs to be processed?
  - Wait for it?
- Do something useful: DMA
  - Used by high-speed I/O devices
  - Direct transfer from device controller buffer’s storage to main memory region
    - No CPU intervention
  - Use of a single purpose processor
    - DMAC (DMA Controller)



DMA hardware requirements

- With DMA, we have *two* processors accessing main memory, potentially simultaneously
- This requires either **dual-port memory** or **arbitration circuitry**, both of which are quite complex/ expensive



Operating System terms

Multiprogramming, Multitasking, Multiprocessing & Multithreading

Process Management

- A process is a program in execution that needs resources
  - Program is a passive entity, process is an active entity

- Single-threaded process has one program counter specifying location of next instruction to execute

- Multi-threaded process has one program counter per thread

Process Management

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All instructions in memory
- Memory management determines what is in memory
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- File-System management:
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
- OS:
  - Creating/editing and deleting files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Recommended Reading

- Section 1.3, 1.8 : https://ict.iitk.ac.in/wp-content/uploads/CS422-Computer-Architecture-ComputerOrganizationAndDesign5thEdition2014.pdf
- Silberschatz et. al., chapter 1 up to section 1.8