E2011: Theoretical fundamentals of computer science
Topic 4: Introduction to computer architectures

Vlad Popovici, Ph.D.

Fac. of Science - RECETOX

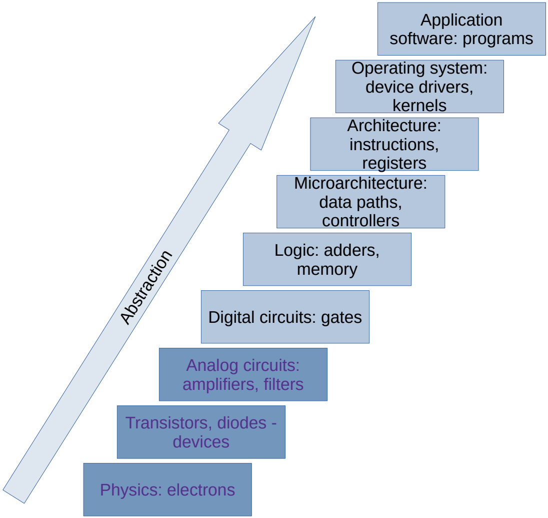
Outline

Motivation

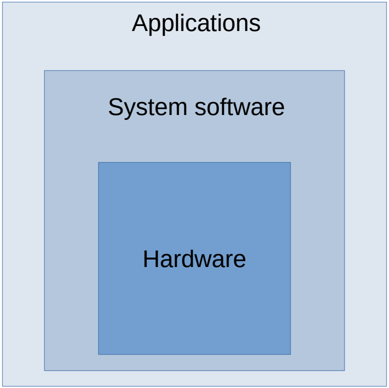
General Specifications

Platform:	Desktop
Product Family:	AMD Ryzen™ PRO Processors
Product Line:	AMD Ryzen™ Threadripper™ PRO 5000 WX-Series
AMD PRO Technologies:	Yes
Consumer Use:	No
Regional Availability:	Global, China, NA, EMEA, APJ, LATAM
Former Codename:	"Chagall PRO"
Architecture:	"Zen 3"
# of CPU Cores:	64
# of Threads:	128
Max. Boost Clock:	Up to 4.5GHz
Base Clock:	2.7GHz
L1 Cache:	4MB
L2 Cache:	32MB
L3 Cache:	256MB
Default TDP:	280W
Processor Technology for CPU Cores:	TSMC 7nm FinFET
Unlocked for Overclocking:	Yes
CPU Socket:	sWRX8
Socket Count:	1P
Max. Operating Temperature (Tjmax):	95°C

An abstract view of a computer system



Another view



Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast
4. performance via parallelism

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast
4. performance via parallelism
5. performance via pipelining

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast
4. performance via parallelism
5. performance via pipelining
6. performance via prediction

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast
4. performance via parallelism
5. performance via pipelining
6. performance via prediction
7. hierarchy of memories

Eight great ideas in computer design

(from Patterson and Hennessy's "Computer Organization and Design")

1. design for Moore's Law
2. use abstraction to simplify design
3. make the common case fast
4. performance via parallelism
5. performance via pipelining
6. performance via prediction
7. hierarchy of memories
8. dependability via redundancy

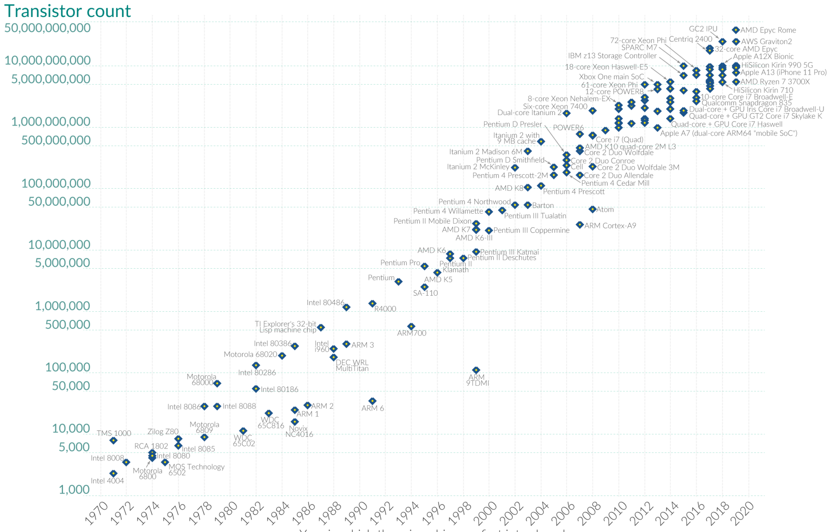
Moore's law

The number of transistors in cost-effective integrated circuit double every 18-24 months.

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

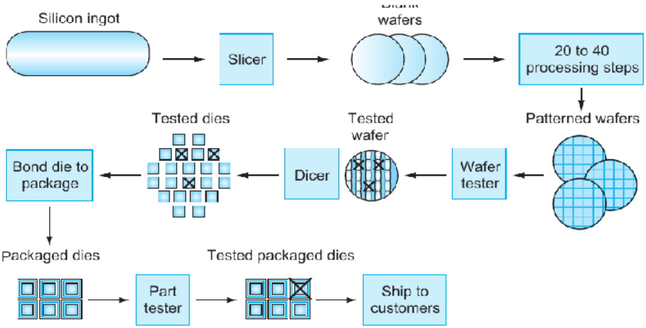
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Chip manufacturing process



(from Patterson and Hennessy's “Computer Organization and Design”)

Performance

- what is the *performance* of a computer?

Performance

- what is the *performance* of a computer?
- response time vs throughput

Performance

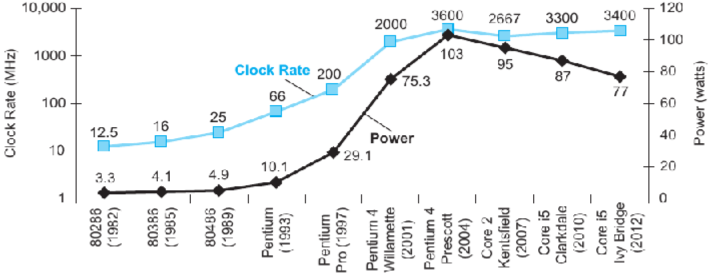
- what is the *performance* of a computer?
- response time vs throughput
- hardware vs software performance

Performance

- what is the *performance* of a computer?
- response time vs throughput
- hardware vs software performance
- energy per instruction

Performance

- what is the *performance* of a computer?
- response time vs throughput
- hardware vs software performance
- energy per instruction
- measuring performance

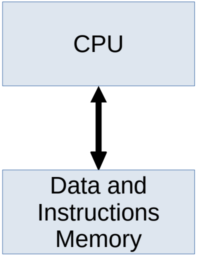


(from Patterson and Hennessy's "Computer Organization and Design")

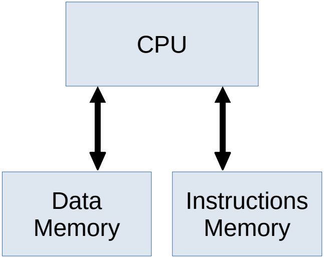
General architecture

In a very simplistic view,

$$\text{Computer} = \text{Central Processing Unit} + \text{Memory}$$

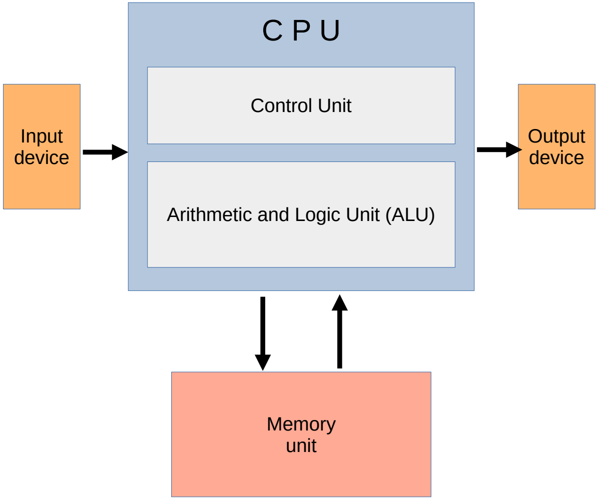


von Neumann architecture



Harvard architecture

Central Processing Unit (CPU)

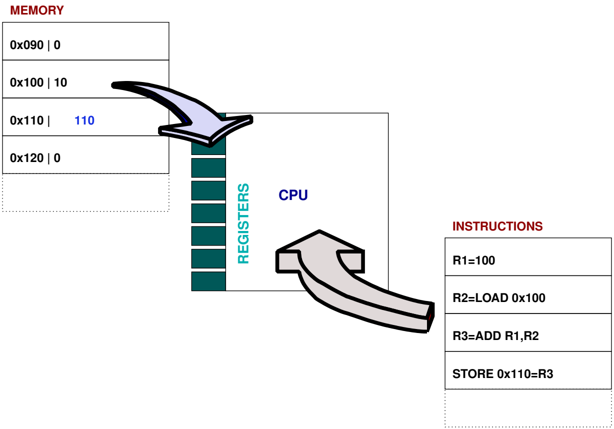


Central Processing Unit (CPU)

- CPU executes instructions read from memory
- instructions for *loading* and *storing* values
- instructions that operate on values from *registers*, e.g. additions, bitwise operations, math functions etc.
- *branching* instructions
- etc

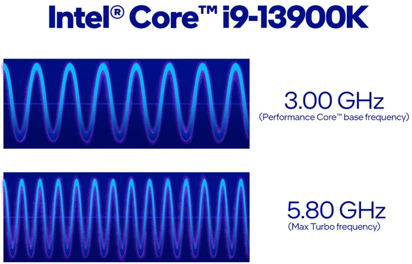
CPU

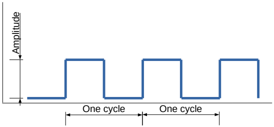
Registers: internal (to CPU) memory cells used



Speed, clock, cycles

- internal clock: used to maintain synchronicity of th operations
- the frequency of the clock (in MHz, or GHz nowadays) gives the speed of the CPU: one operation may start on each tick

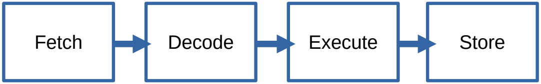




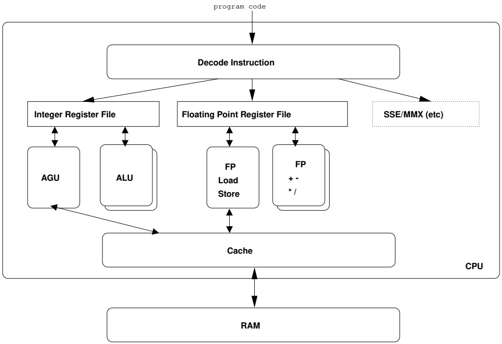
Instruction cycle

Main steps in executing an instruction

- fetch: read instruction from memory
- decode: figure out what to do
- execute: take values from register and execute instruction
- store: save the result in a register

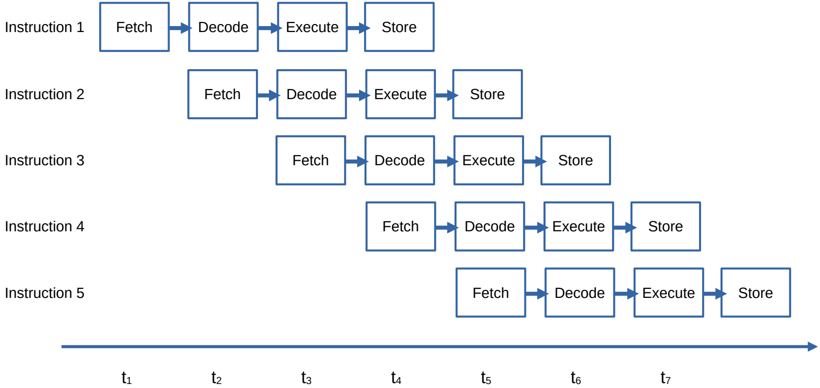


CPU: more details



- *register*: fast internal storage; small - several bytes per register
- *register file*: the set of similar registers within CPU
- register are specialized: storing integer, floating point, instructions, addresses etc
- AGU: *address generation unit* - handles data access

CPU: pipelines



CPU: CISC vs RISC

CISC: Complex Instruction Set Computer

- the original ISA
- one instruction may take several cycles
- emphasizes hardware over software
- complex instructions (e.g memory-to-memory LOAD/STORE)
- shorter programs
- high cycles per second

RISC: Reduced Instruction Set Computer

- improvement on CISC
- one clock-cycle per instruction
- emphasis on software
- register-to-register LOAD/STORE
- uses many internal registers
- low cycles per second

CPU: CISC vs RISC

Example: compute $A \times B$. Assume A is stored at memory location 1200, and B at 1201, respectively.

The following instruction(s) performs the multiplication and stores the result at the first memory location.

CISC

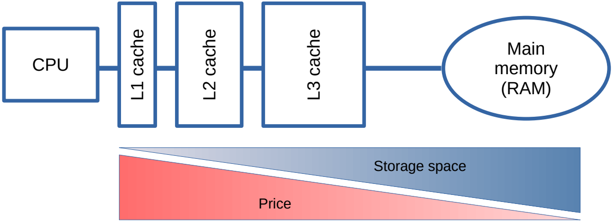
```
MUL 1200,1201
```

RISC

```
Load A, 1200
Load B, 1201
Mul A, B
Store 1200, A
```

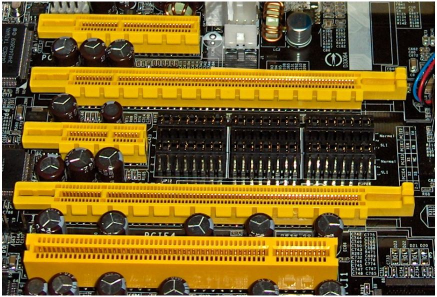
CPU: multilevel cache

- cache: fast memory closer to CPU
- improves data access speed by reducting emphmiss penalty



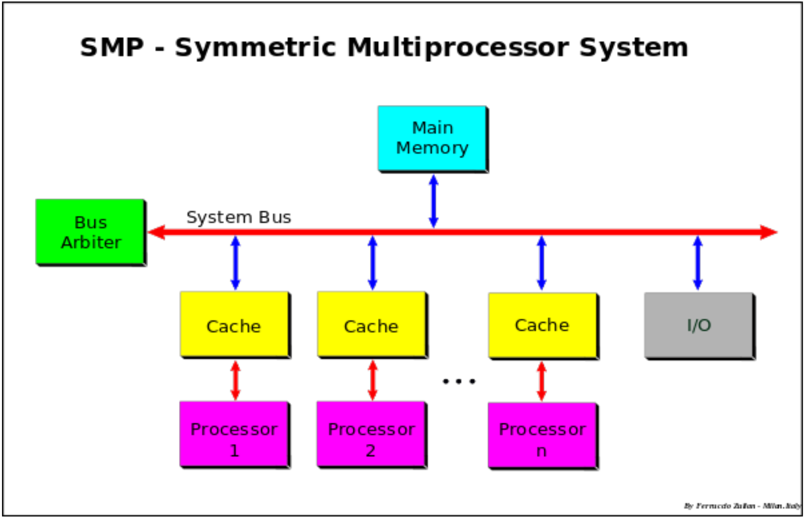
Moving bits and bytes - data buses

- a (computer) *bus* refers to hardware and protocols for transferring data
- internal buses: data (memory) bus, system bus, control bus, etc
- external (expansion) buses: connects devices to computer



Parallelism

SMP: symmetric multiprocessor systems



Parallelism

SMP: symmetric multiprocessor systems

Advantages:

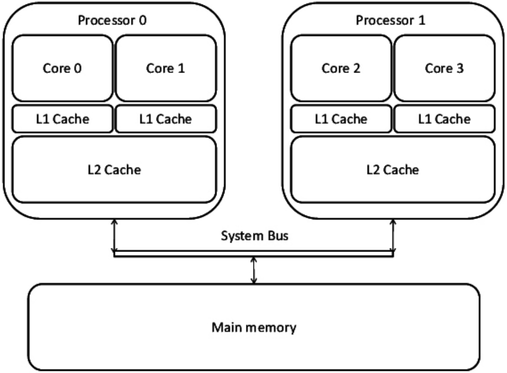
- increased throughput
- redundancy, hency reliability
- easy configuration.
- more *processes* executing at same time: MultiProcessing.

Drawbacks:

- increased traffic over bus, longer distances between two CPUs
- risk of bottlenecks on shared resources
- coordination becomes much more complex

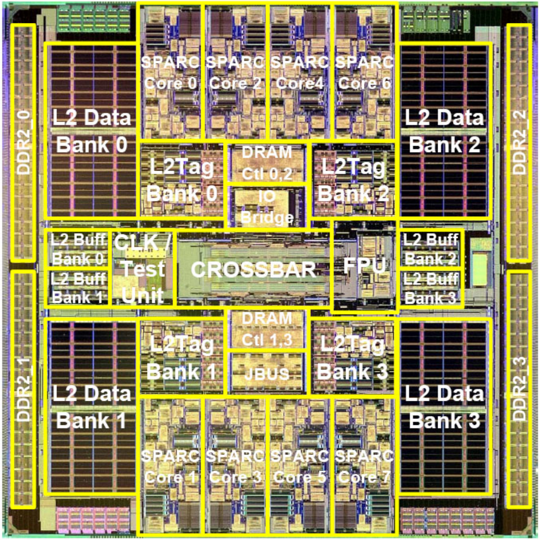
Parallelism

Multicore



Parallelism

Multicore - example OpenSPARC (Sun Microsystems)



Parallelism

Multicore

Advantages:

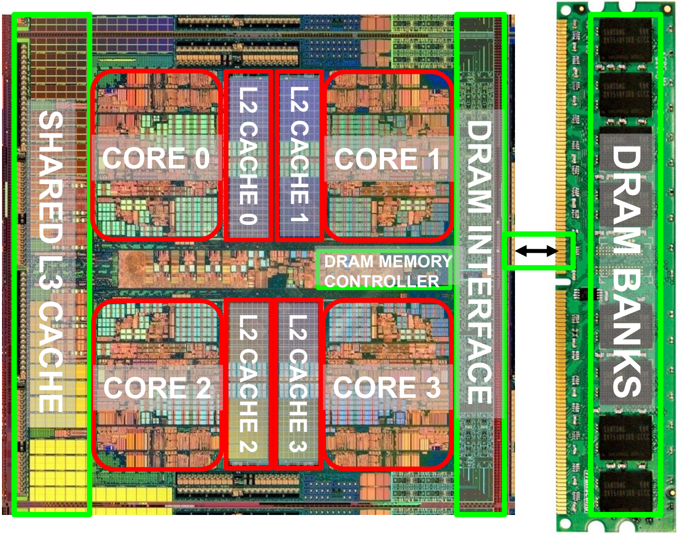
- run *instructions* in parallel on different cores
- usually use a single die, or onto multiple dies but in single chip package
- more energy efficient: higher performance at lower energy
- less traffic, shorter distances than SMP

Drawbacks:

- overhead in writing specific code
- dual-core processor does not work at $2\times$ speed of single processor, but $60\% - 80\%$ more speed
- some operating systems still not exploit the multicore

Memory organization

Computer = Central Processing Unit + Memory



Wishes:

- instantaneous access to any bit (0-latency)
- infinite capacity
- cheap (i.e. 0$)
- infinite bandwidth

Wishes:

- instantaneous access to any bit (0-latency)
- infinite capacity
- cheap (i.e. 0$)
- infinite bandwidth

Reality:

- larger memory is slower: more time to locate the desired position
- faster memory is more expensive (SRAM vs DRAM)
- larger bandwidth is more expensive

Memory technology

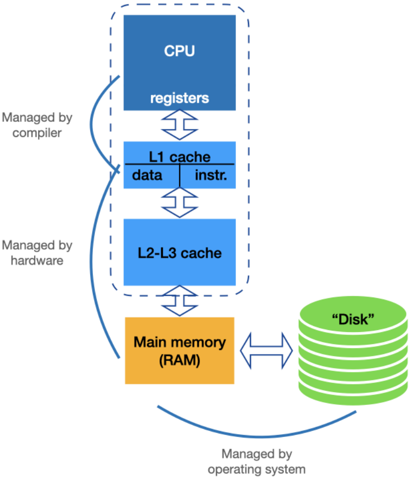
SRAM

- *Static Random Access Memory*
- per bit: 2 transistors for access, 4 transistors for storage
- it keeps state as long as the power is on

DRAM

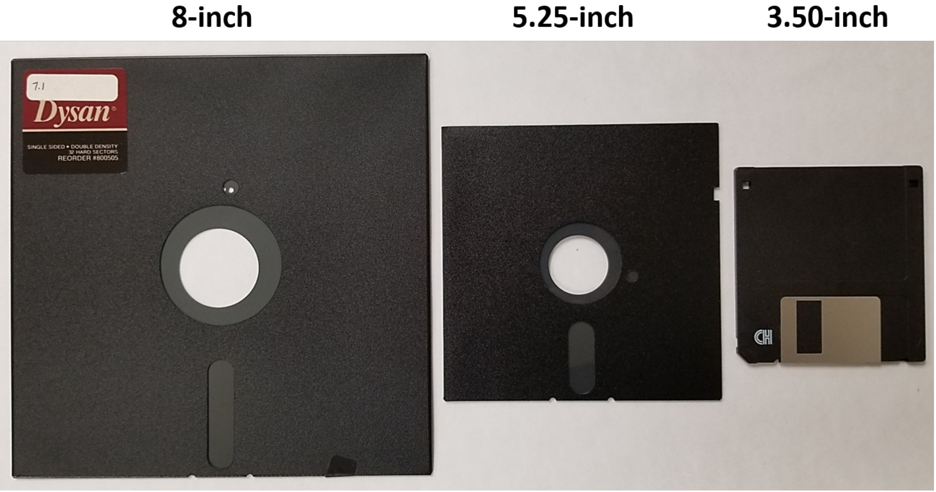
- *Dynamic Random Access Memory*
- per bit: 1 capacitor, 1 access transistor
- loses charge over time $\rightarrow$ needs refresh cycles

- level 0 (volatile): CPU registers: data for instructions, etc
- level 1 (volatile): L1 cache: SRAM, separate data and instruction space, KBs/core
- level 2 (volatile): L2/3 cache: SRAM, normally within the same chip as CPU, MBs/core
- level 3 (volatile): main memory: usually DRAM; tens GBs (less often hundreds GBs or 1TB); in embedded devices could be SRAM (KBs-MBs in size)
- level 4 (permanent): disks, SSD - TBs in size



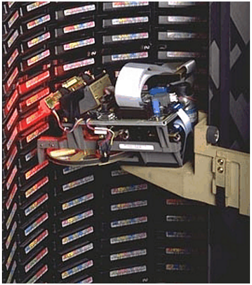
Memory - other storage media

Floppy disks - now mostly extinct



Magnetic tapes - still relevant since 50s...

Magnetic tapes - still relevant since 50s...



Flash memory

- non-volatile electronic memory that can be electrically reprogrammed
- based on NAND or NOR gates
- limited number of write/erase cycles
- data degradation over time

Questions?