K22 – Operating Systems: Design Principles and Internals

Fall 2026 @dit (α.μ. mod 2 = 1)

Vaggelis Atlidakis et al.

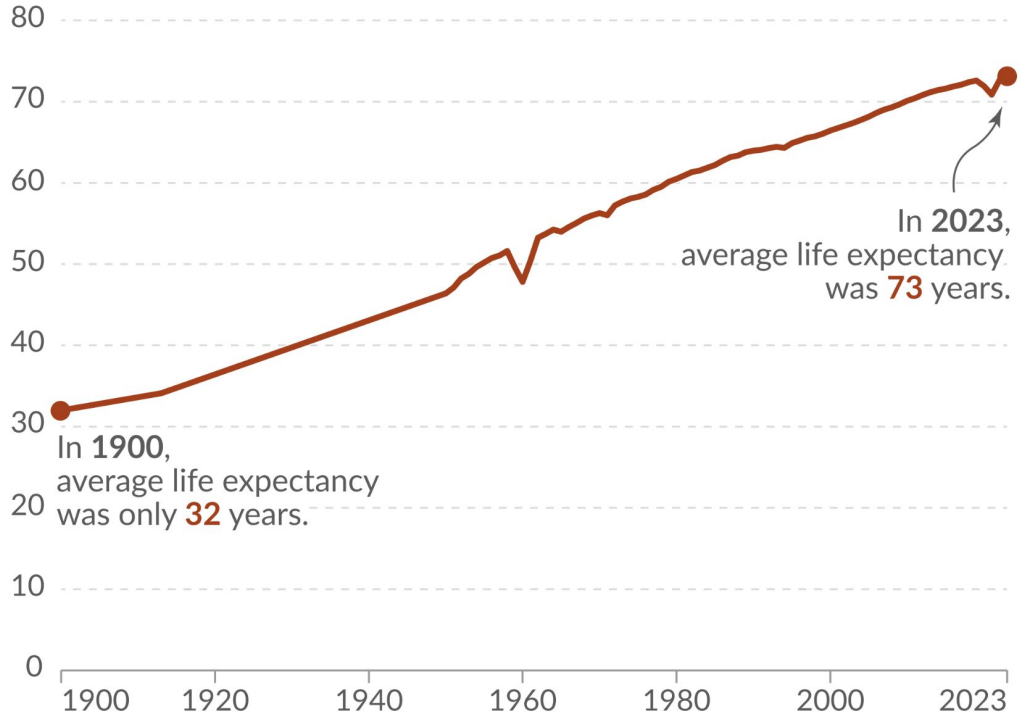
Lecture 02

References: Similar OS courses @Columbia, @Stanford, @UC San Diego, @Brown, @di (previous years); and textbooks: Operating Systems: Three Easy Pieces, Operating Systems: Principles and Practice, Operating System Concepts, Linux Kernel Development, Understanding the Linux Kernel

Global average life expectancy has more than doubled since 1900

Our World in Data

Period life expectancy (in years) at birth, in a given year.

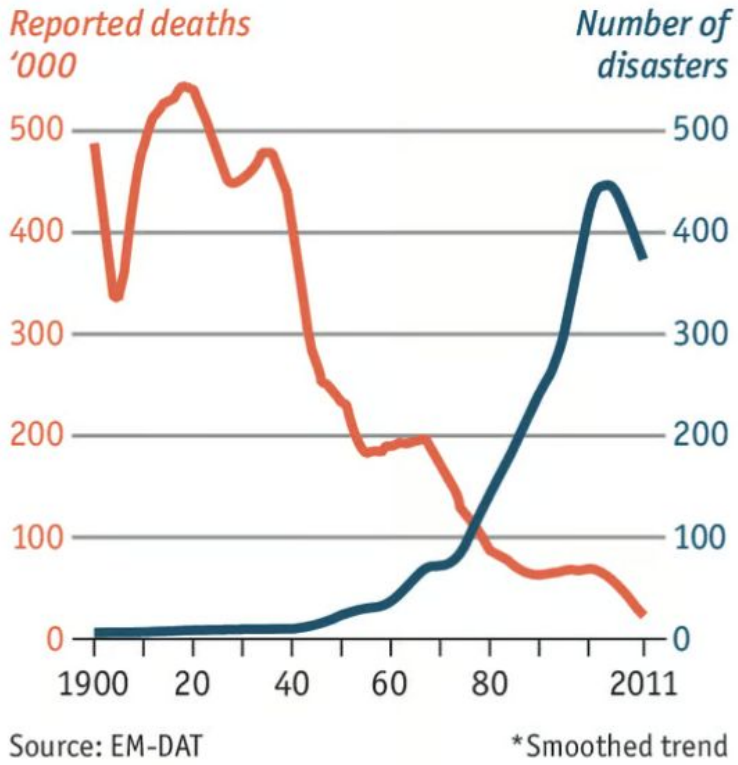


Data source: UN WPP (2024); HMD (2024); Zijdeman et al. (2015)

OurWorldinData.org/life-expectancy | CC BY

More, but less deadly

Global deaths from natural disasters*

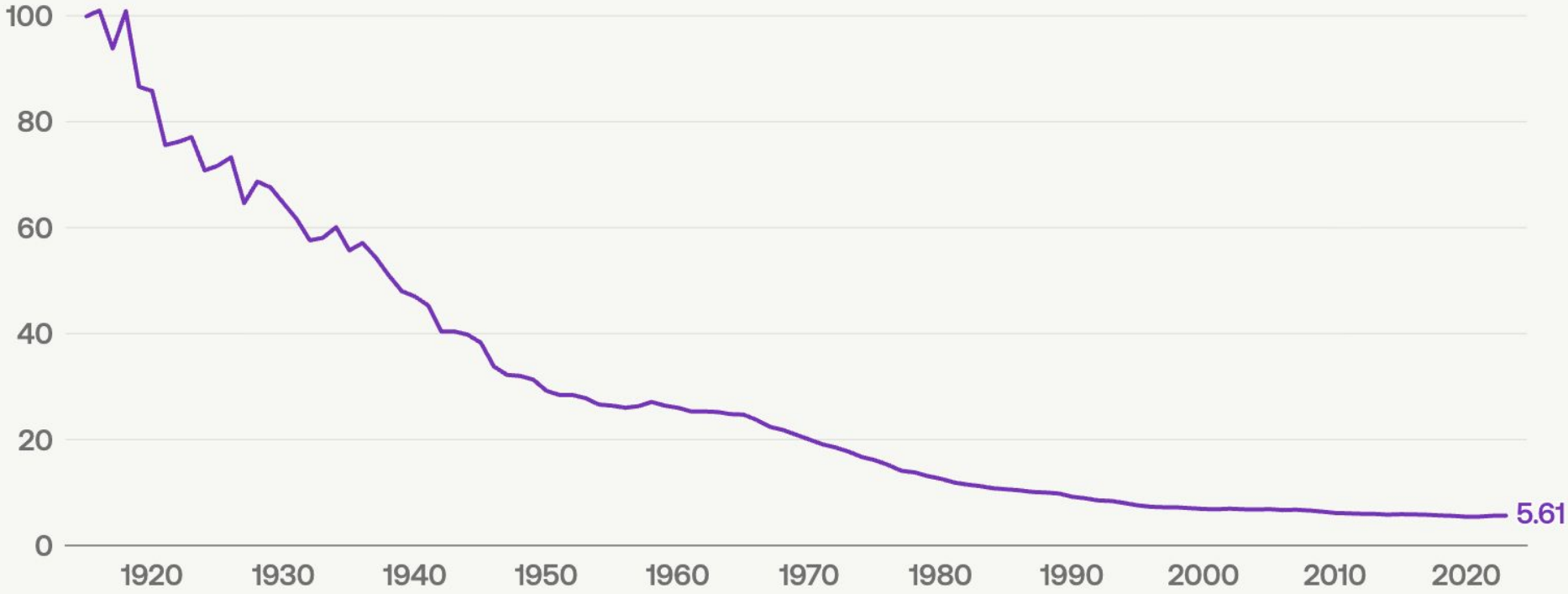


Source: EM-DAT

*Smoothed trend

Economist.com

Infant mortality rate per 1,000 live births, 1915–2023



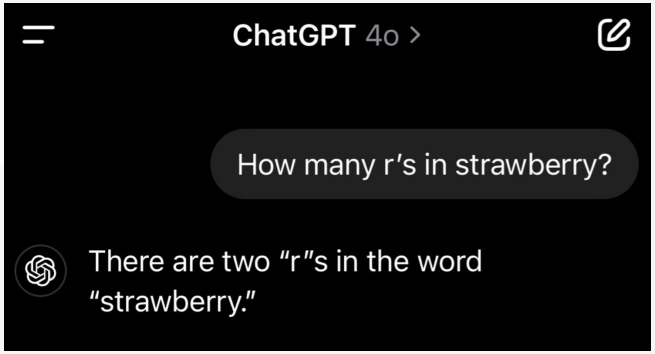
Source: Centers for Disease Control and Prevention and National Center for Health Statistics

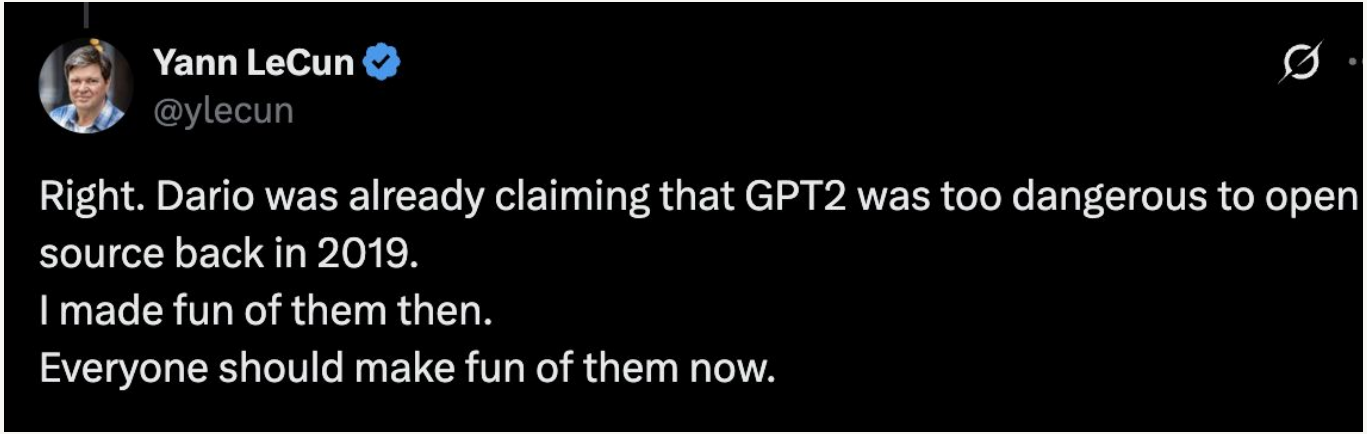
USAFacts


NASA Mars Rover, May 19, 2005

Dario Amodei Warned Against GPT-2 in 2019.

BLOG/DARIO-AMODEI-GPT2-OPENAI-OPEN-SOURCE-AI-CONTROVERSY-2026





Geoffrey Hinton

Professor emeritus, University of Toronto; Turing Award laureate

WHY THEY MATTER One of the 'godfathers of deep learning' and a Turing Award laureate

STATUS	PREDICTION MADE	PREDICTED ARRIVAL	INTERVAL
Closed	2016	2021	5 yrs

> *"People should stop training radiologists now. It's just completely obvious that within five years deep learning is going to do better than radiologists."*

ERAS applicants	2018	2019	2020	2021	2022	2023	Trendline	6y slope	3y slope	Trend category
Radiology-Diagnostic	2,033	1,893	1,839	1,991	2,232	2,409		4.3%	10.5%	rising
Dermatology	862	896	855	921	1,037	1,038		4.5%	6.4%	rising
Physical Medicine and Rehabilitation	966	901	877	968	1,049	1,092		3.4%	6.4%	rising
Child Neurology	336	312	329	352	345	391		3.4%	5.5%	rising
Orthopaedic Surgery	1,254	1,341	1,413	1,523	1,727	1,666		7.6%	4.7%	rising
Anesthesiology	2,721	2,991	3,015	3,287	3,348	3,599		6.0%	4.7%	rising
Urology	462	449	503	539	605	556		6.0%	1.6%	plateauing
Plastic Surgery - Integrated	304	301	358	416	420	405		8.6%	-1.3%	plateauing
Neurological Surgery	422	438	489	502	486	461		2.4%	-4.1%	pulling back
Otolaryngology - Head and Neck Surgery	398	532	569	631	644	549		8.3%	-6.5%	pulling back
Vascular Surgery - Integrated	148	178	222	265	237	227		11.9%	-7.2%	pulling back
Neurology	2,069	1,925	1,931	2,240	2,233	2,251		3.0%	0.2%	stable
Surgery	5,329	5,118	5,059	5,515	5,476	5,466		1.2%	-0.4%	stable
Internal Medicine	22,930	23,414	23,433	25,025	24,373	24,579		1.6%	-0.9%	stable
Thoracic Surgery - Integrated	185	209	175	210	189	205		1.2%	-1.2%	stable
Obstetrics and Gynecology	2,543	2,593	2,508	2,577	2,631	2,488		-0.1%	-1.7%	stable
Pathology-Anatomic and Clinical	1,556	1,507	1,472	1,580	1,573	1,497		0.0%	-2.6%	stable
Radiation Oncology	253	230	213	209	235	246		-0.3%	8.9%	rebounding
Interventional Radiology - Integrated	522	452	330	391	408	444		-2.5%	6.8%	rebounding
Psychiatry	4,914	4,559	4,246	4,583	4,162	4,052		-3.0%	-5.8%	falling
Pediatrics	5,707	5,403	5,330	5,698	5,202	4,959		-2.0%	-6.5%	falling
Internal Medicine/Pediatrics	961	935	967	961	842	824		-2.9%	-7.1%	falling
Family Medicine	12,648	12,383	12,217	12,538	11,606	10,681		-2.7%	-7.4%	falling
Emergency Medicine	3,576	3,584	3,788	4,391	3,632	3,282		-0.6%	-12.6%	falling

Data source: AAMC Electronic Residency Application Service Statistics, as of 2/23 of each year.

@francisdeng

Elon Musk

CEO, Tesla and SpaceX

WHY THEY MATTER CEO of Tesla, SpaceX and xAI

STATUS	PREDICTION MADE	PREDICTED ARRIVAL	INTERVAL
Closed	2020	2025	5 yrs

> *"We are headed toward a situation where AI is vastly smarter than humans. I think that time frame is less than five years from now."*



Elon Musk  

@elonmusk

Follow  ···

Our C/C++ training stack was written by humans. AI is not yet good enough for extremely high performance software (but it will be).

7:56 PM Sep 14, 2026 · **1.1M** Views

 540  750  10K  863 

Relevant  View quotes 

There was "Nothing left to discover" before

Dawn of the 20th century

> Scientific community was certain nature laws were fully understood
- Philipp von Jolly advised young Max Planck against studying physics
- "Almost everything in physics is already discovered." [1]

> Plank ignored this advice and went on to originate quantum theory
- One of the founders of modern physics
- Revolutionized understanding of atomic and subatomic processes

> Narrative: We reached the "end of tech/ innovation history"
- Obviously false then
- Most certainly false now

Motos of top-5 S&P-500 companies

Rarely about worshiping the technology itself

> Amazon: *"Obsess with customer experience."*
  - Proving the best customer service
  - See this (Jeff Bezos 1999)

> Apple: *"You've got to start with the customer experience"*
  - Work back toward the technology—not the other way around.
  - See this (Steve Jobs 1997)

> Google: *"Focus on the user and all else will follow."* [2]

> Microsoft: *"Empower every person…to achieve more."* [3]

Instructions on how to use GenAI

Don't be a meat proxy

> *A person who passes along AI-generated content w/o understanding*

- You bring zero value, if you do what everyone else does
- How did we ended up here? Systems Engineering -> Vibe Coding ->
  -> Prompt Engineering -> Context Engineering -> Harness Engineering

Don't be a gatekeeper

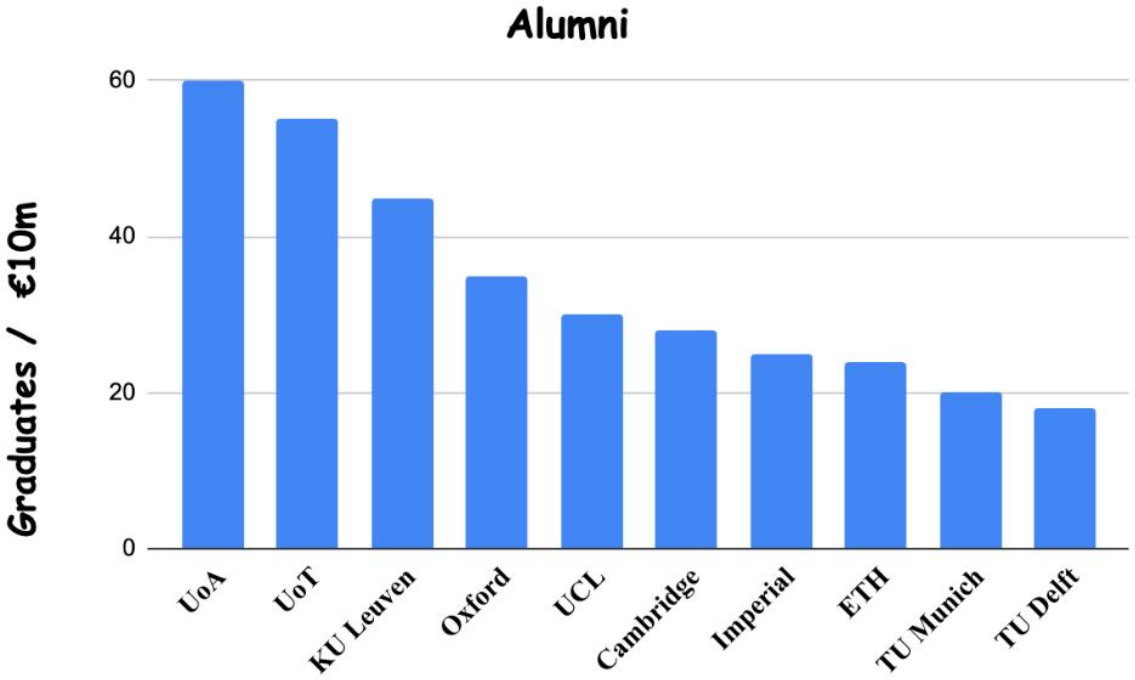
> *A person in denial of a transition happening around them*

- Leverage new tech to learn, and understand more
- Search for the new, **timely** problems

GenAI / Agents are tools: You **must** watch this end-to-end

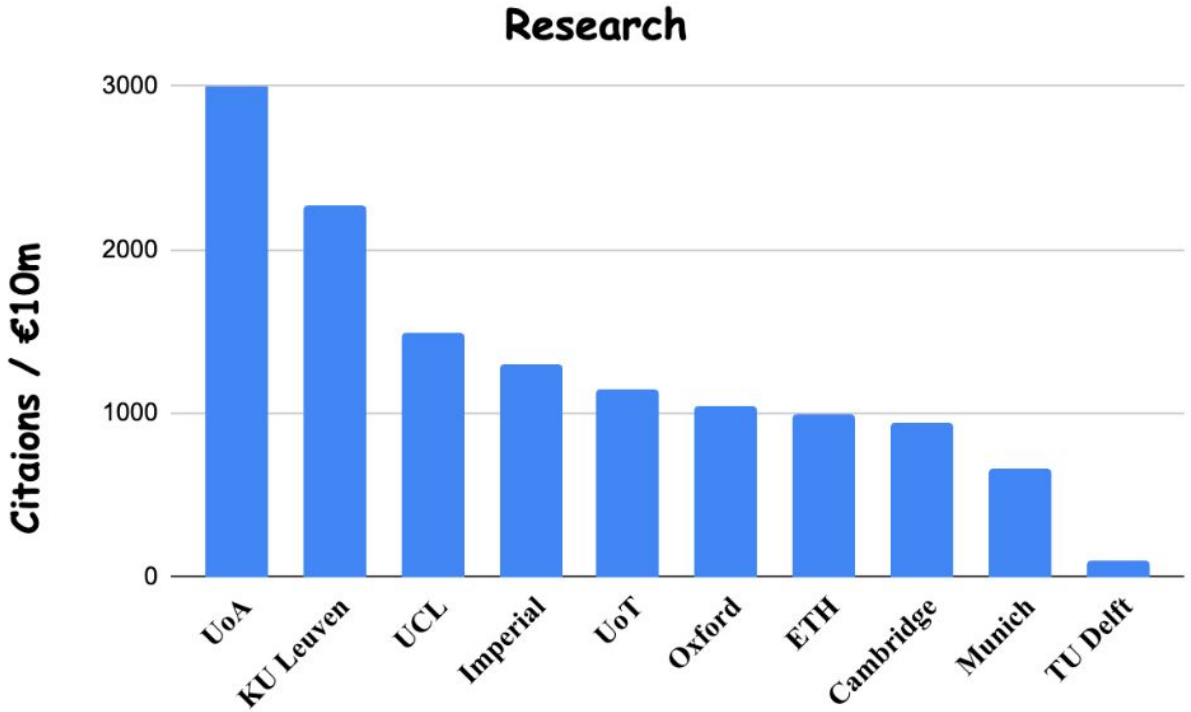
The role of the university

To teach you how to **learn**



The role of the university

To teach you how to **learn**



The role of the university

To teach you how to **socialize**

- The Cooperative Eye Hypothesis



The role of this class

No free lunch

> Code developed with superficial understanding

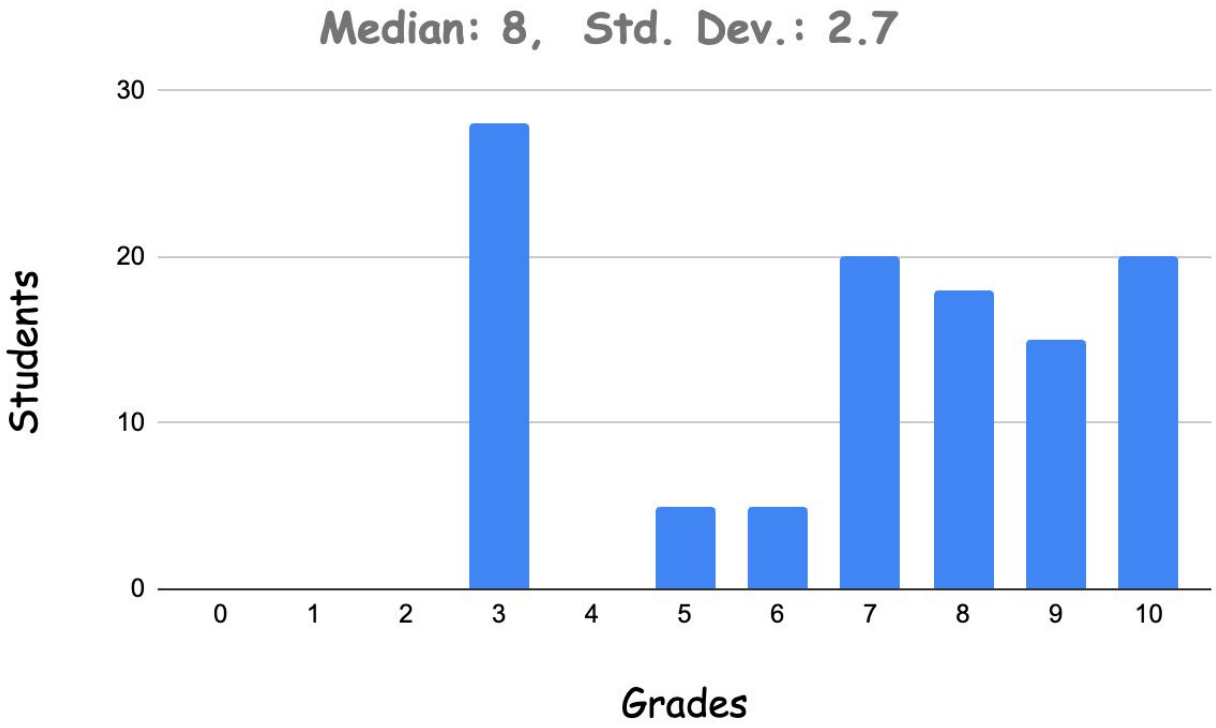
- Bugs won't fix themselves in production
- Requirement for software engineers with legit systems understanding will soon be at an all-time high

Understand things at a fundamental level

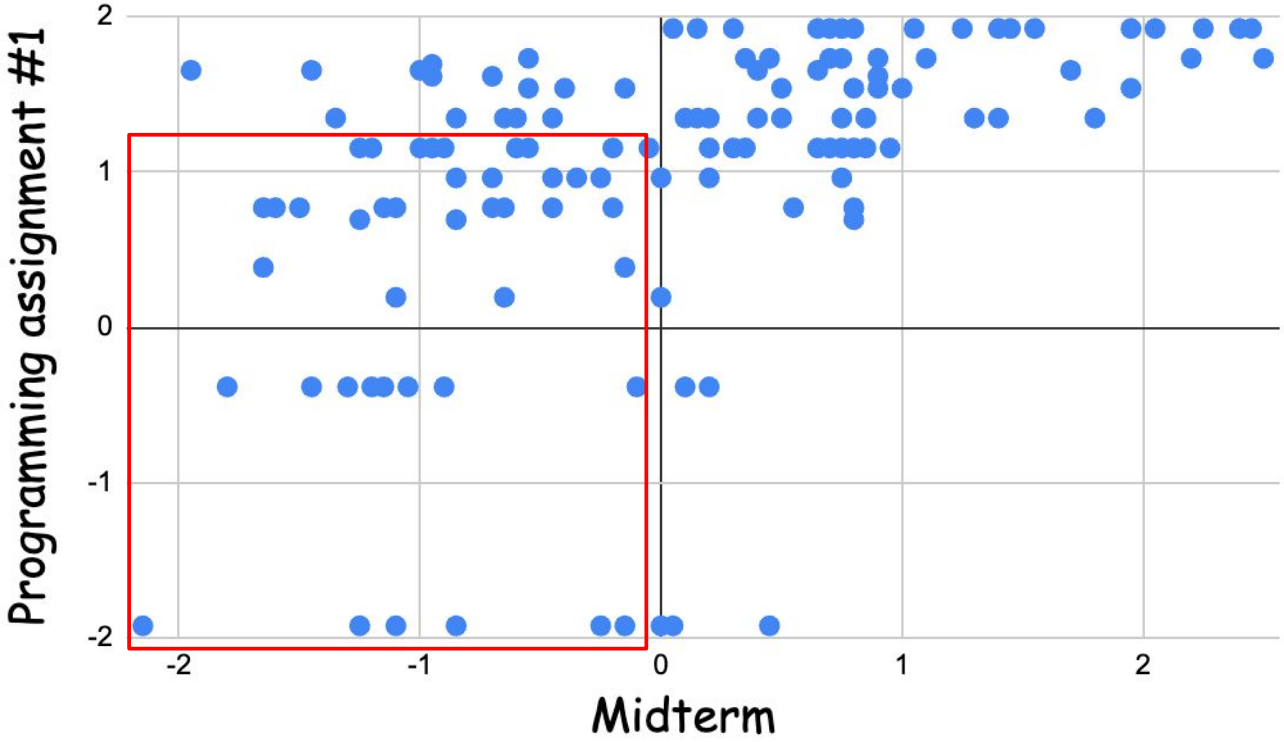
> A line of code is not just a line of code

- Learn how to navigate complex codebases
- Know your circle of competence (Warren Buffet 1996)

What happened last year



Midterm (median 51 / std.dev.: 21)



Administrivia

Programming assignments (50%)

- You'll add features to the Linux kernel and run a VM with it
- You'll need to understand lots of code; won't need to write much
- #1 individual, must get >= 50/100
- #2, #3, #4 group of three

Exams (50%)

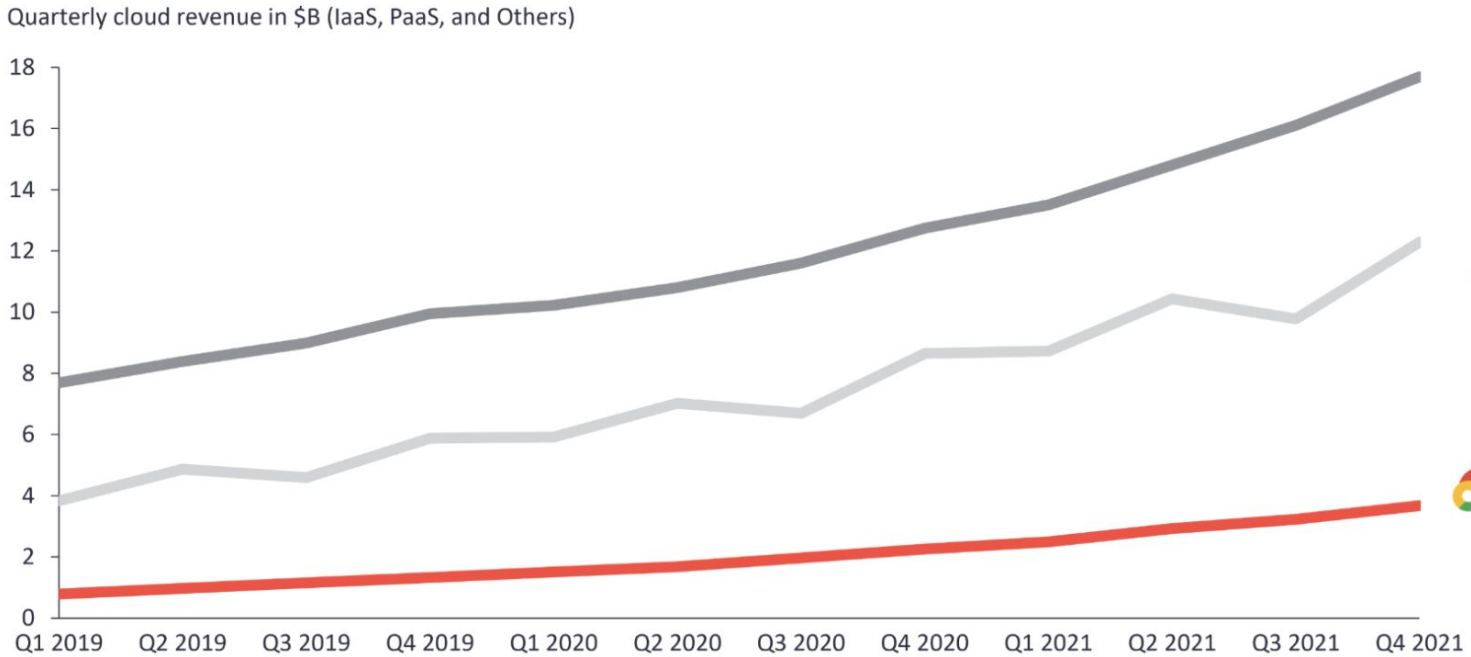
- Midterm (20%)
- Final exam (30%)
- Must get >= 50/100 in at least one exam

Blast from the past

- Drop those kernels boi
- Οι ωραίοι έχουν χρέη!
- Syscall Syndicate
- Printer Hate Club
- Συστουργικα Λειτηματα
- Panic! In the Kernel
- GadaffiOS
- We dont byte
- Onlyfun
- Too Legit to Buffer (TLB)
- The Deadlocks

Why study OSes?

Revenue of leading cloud vendors (2019-21)



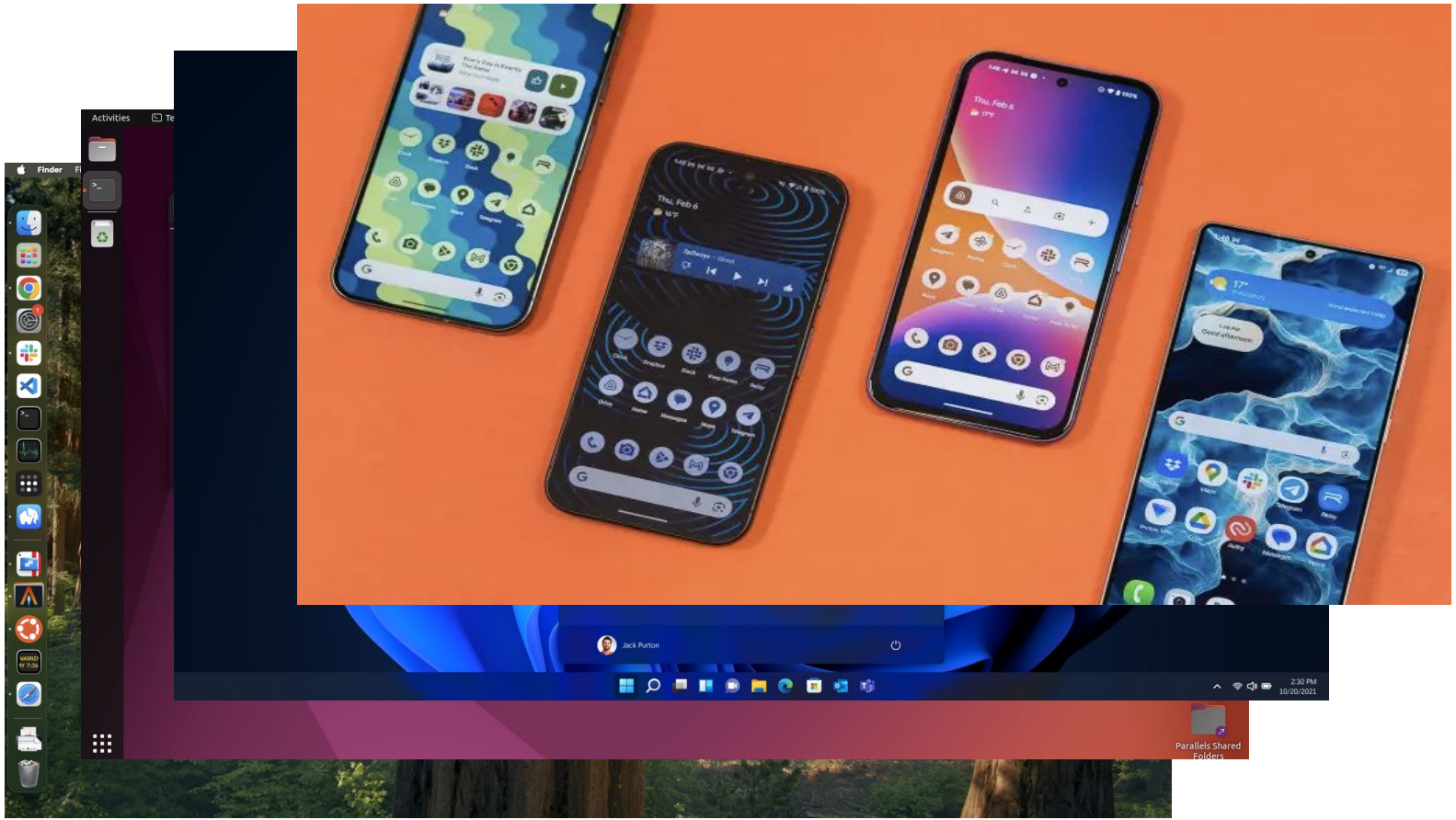
aws

Microsoft Azure

Google Cloud

[1] https://iot-analytics.com/cloud-market/

What is an OS?



What is an OS?

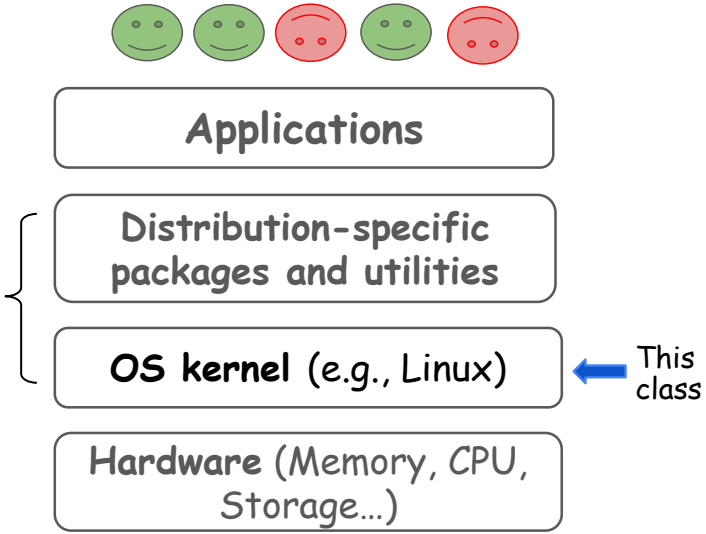
- A layer of abstraction that translates the hardware of a machine into (standardized) software concepts that applications use

Why need an OS?

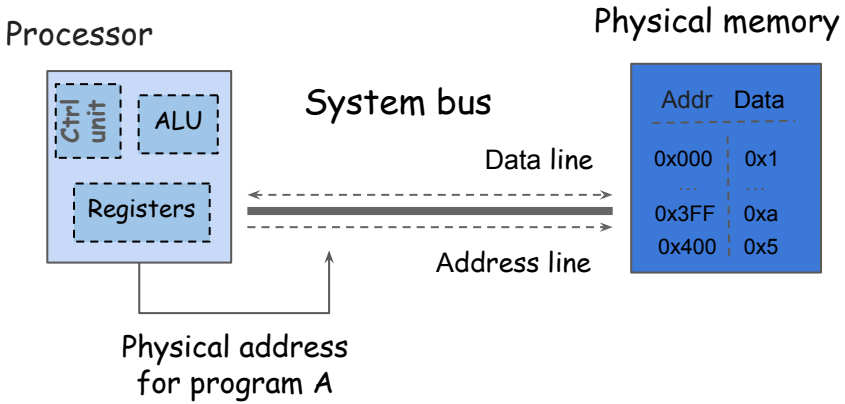
- Multiple users not necessarily sane, build and run different applications, at the same time on shared hardware (design assumptions)

Desirable properties?

- ***Security***: Does the system behave as expected in adversarial contexts?
- ***Reliability***: Does the system behave as expected given benign failures?
- ***Portability***: Is it easy for developers to build and maintain apps?
- ***Performance & Responsiveness***: Is the system "pleasant" to use?



What is the hardware we build OSes for?



Physical Memory (PM)

> Stores data addressable on byte granularity

Processor

> Reads data from memory to its registers

- Performs computations
- Writes the data from its registers to PM

System bus

> Memory and processor communication channel

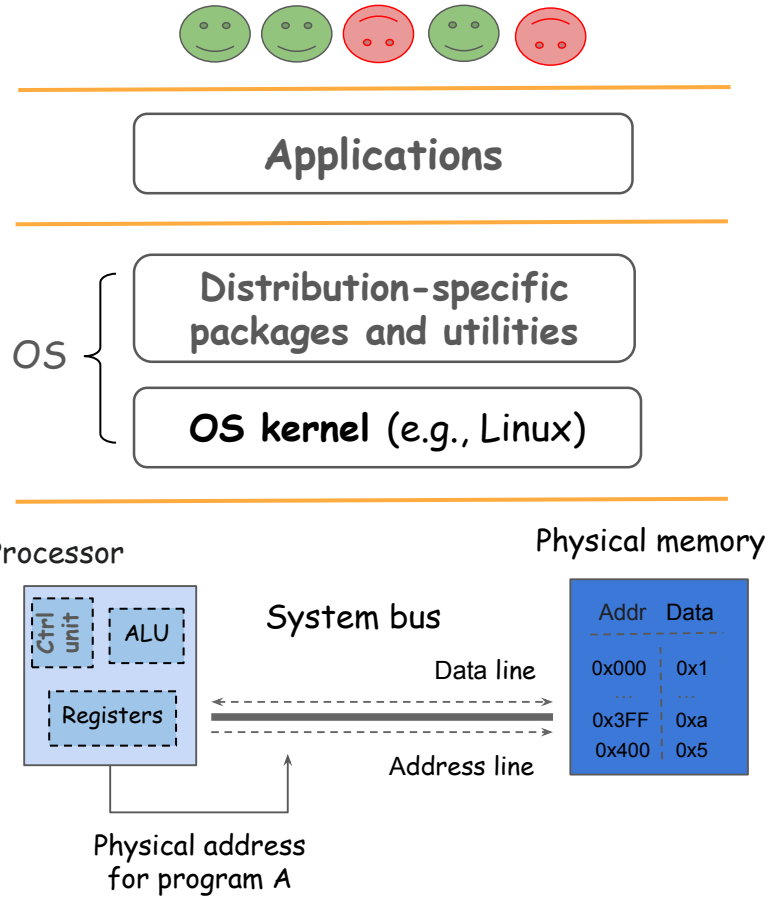
- Is this model satisfactory?
- ~~Reminds you of something?~~

Designing an OS kernel

Desirable properties

1) Reliability, 2) Security, 3) Fairness, 4) Portability

From desirable properties **to** **design principles**



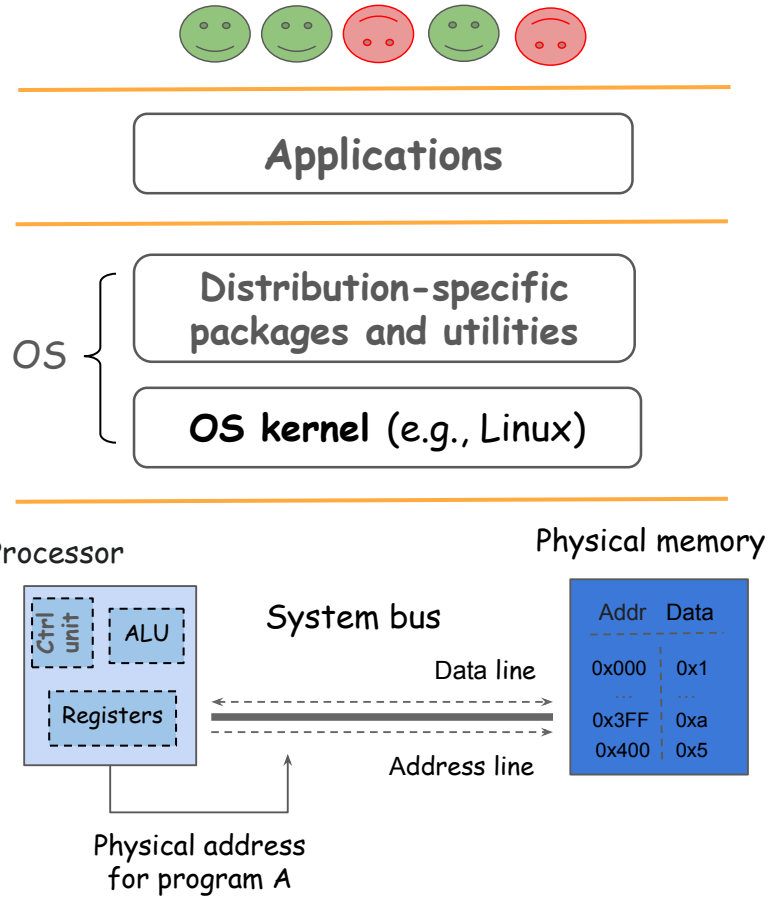
Designing an OS kernel

Desirable properties

1) Reliability, 2) Security, 3) Fairness, 4) Portability

From desirable properties **to** **design principles**

- ***Fault Isolation***: Errors in one running program do not affect any other program



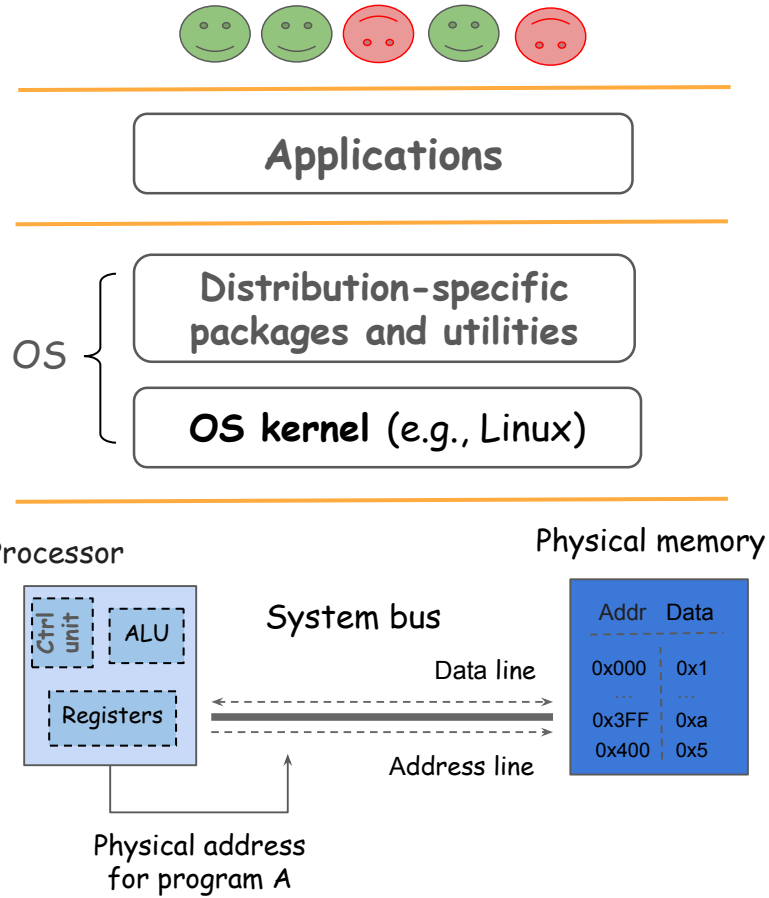
Designing an OS kernel

Desirable properties

1) Reliability, 2) Security, 3) Fairness, 4) Portability

From desirable properties **to** ***design principles***

- ***Fault Isolation***: Errors in one running program do not affect any other program
- ***Principle of Least Privilege (PoLP)***: Any program has the minimum privileges necessary to perform



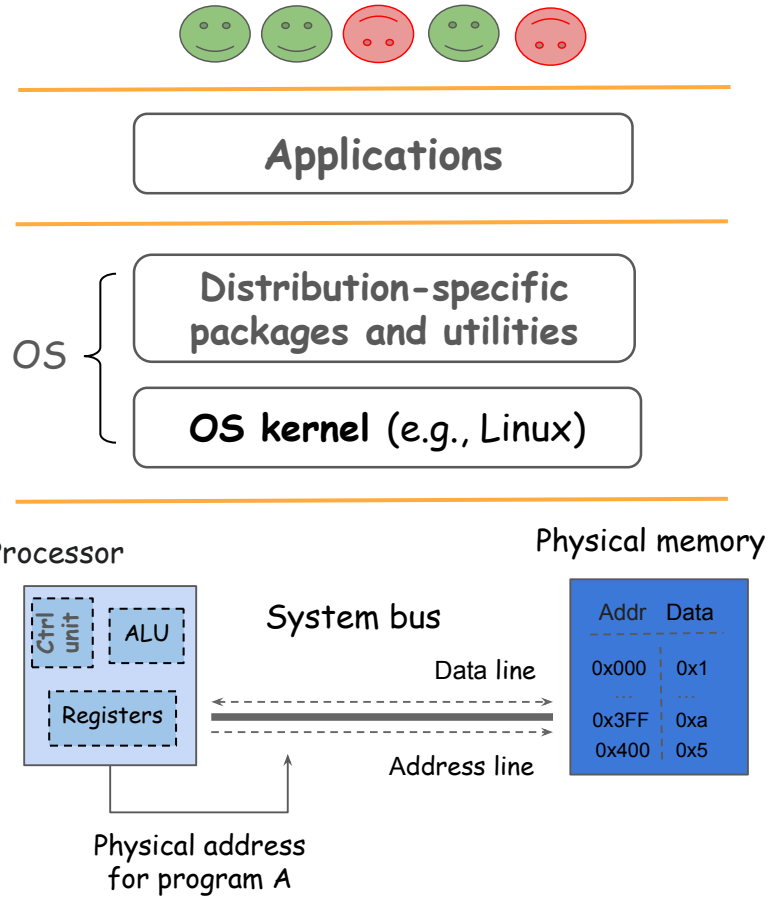
Designing an OS kernel

Desirable properties

1) Reliability, 2) Security, 3) Fairness, 4) Portability

From desirable properties **to** **design principles**

- ***Fault Isolation***: Errors in one running program do not affect any other program
- ***Principle of Least Privilege (PoLP)***: Any program has the minimum privileges necessary to perform
- ***Preemption***: The OS is always able to take control of the processor, regardless of what programs are running

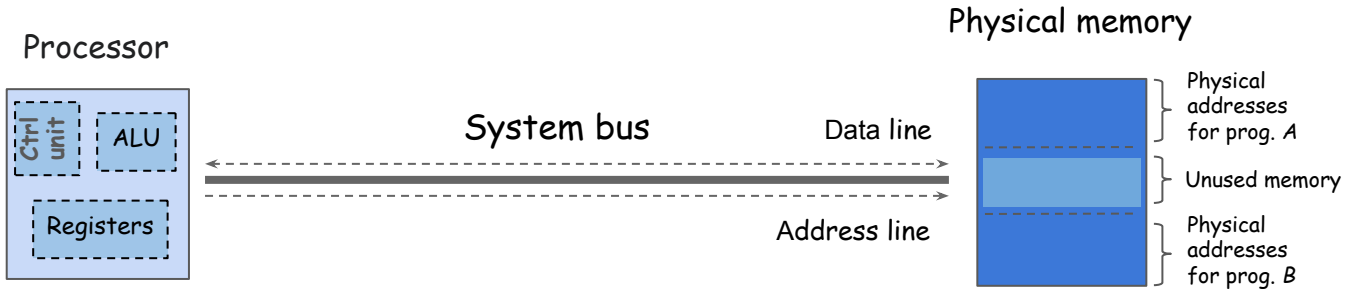


Implementing fault isolation

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory

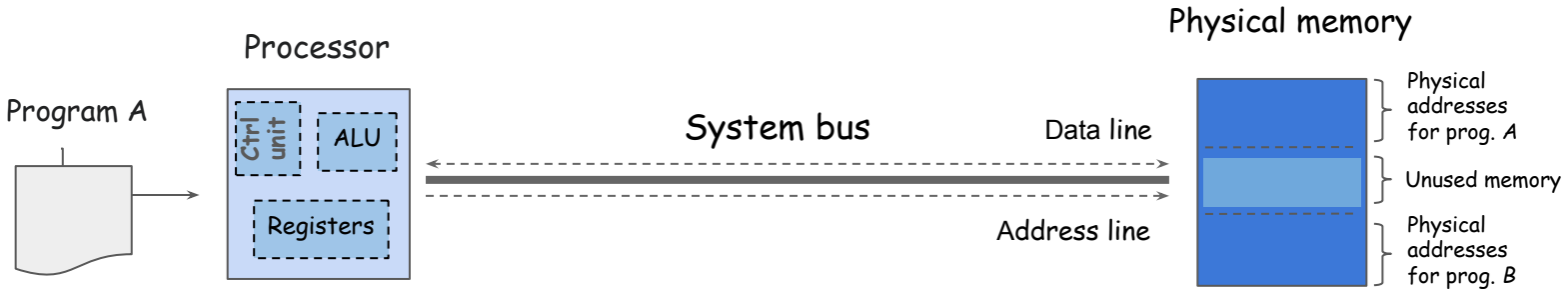
Implementing fault isolation

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory



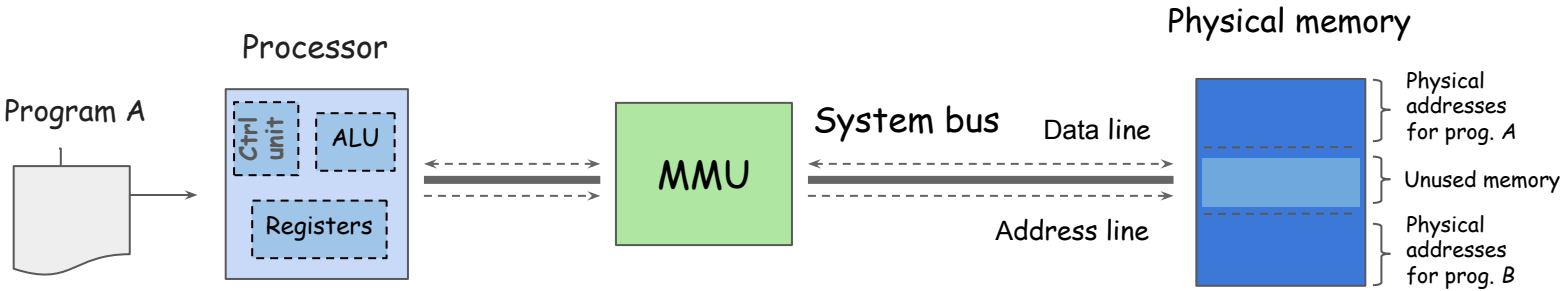
Implementing fault isolation (Virtual Memory)

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory



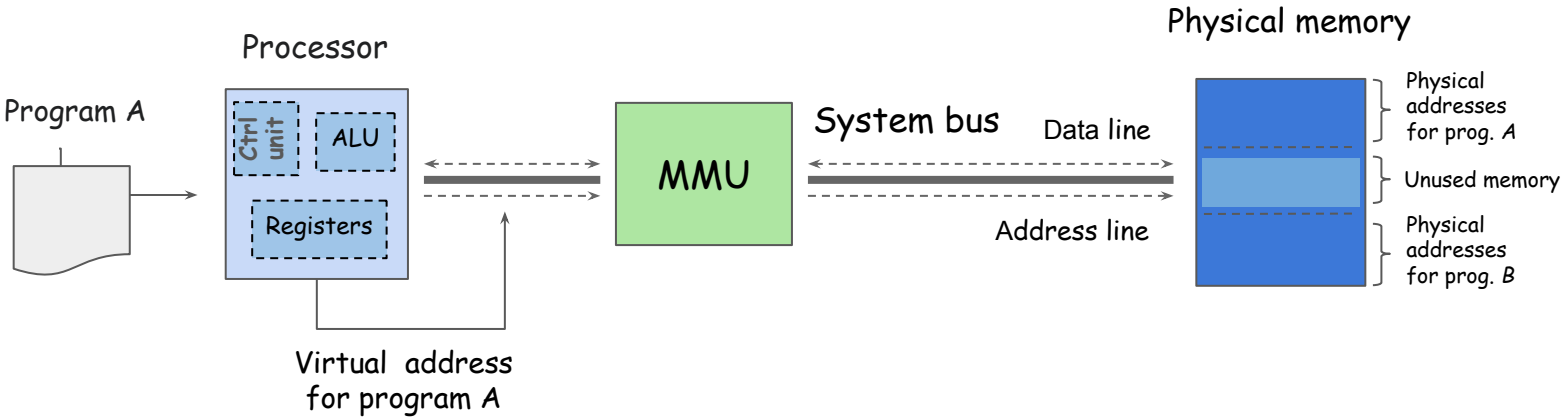
Implementing fault isolation (Virtual Memory)

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory



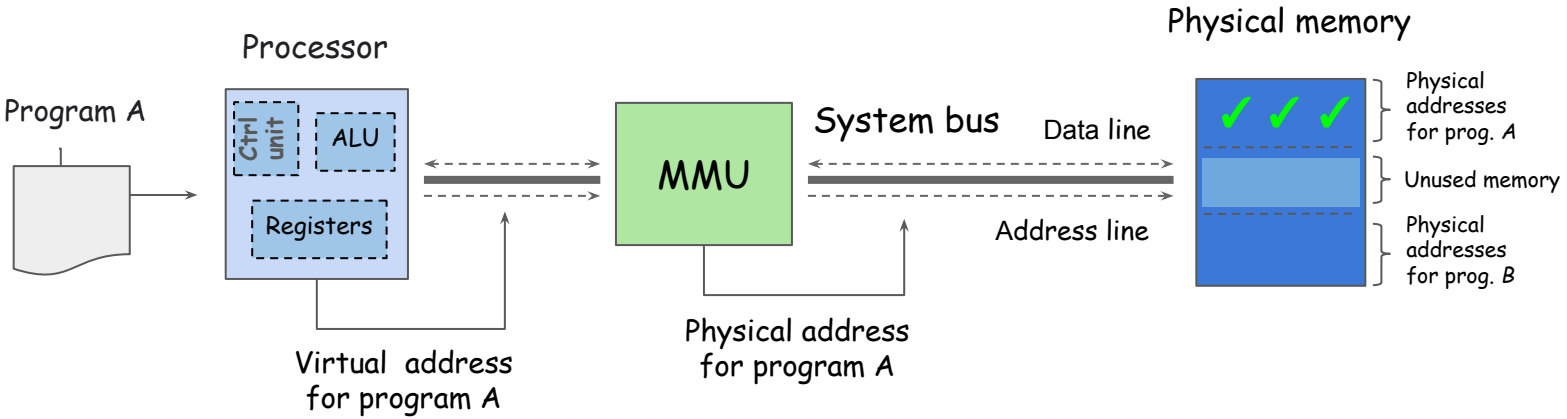
Implementing fault isolation (Virtual Memory)

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory



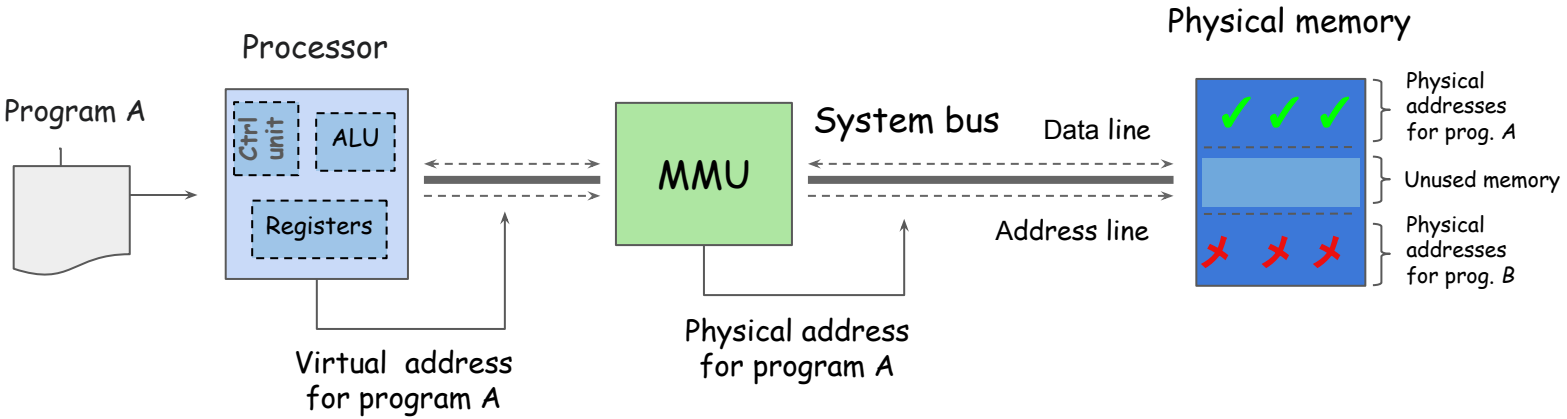
Implementing fault isolation (Virtual Memory)

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory



Implementing fault isolation (Virtual Memory)

- In simple words: load/ store/ jmp instructions of a program cannot read, write, or jump to another program's memory

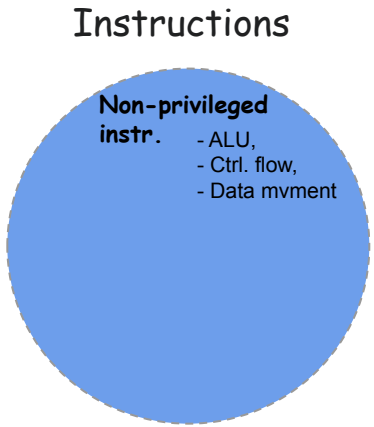


Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware

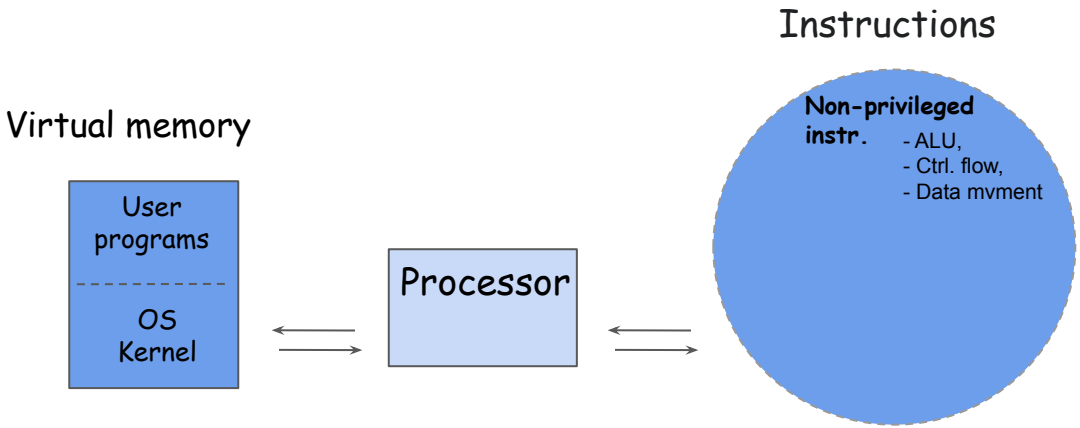
Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware



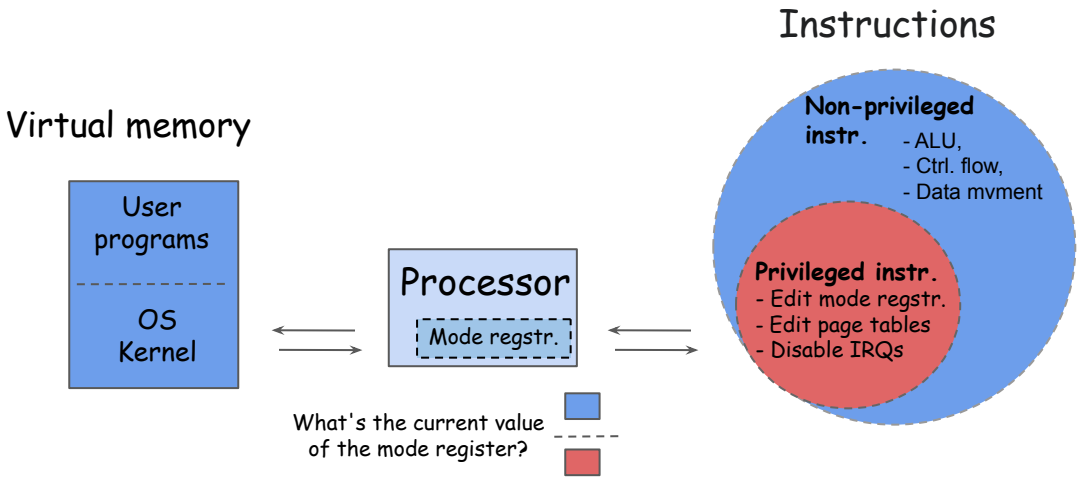
Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware



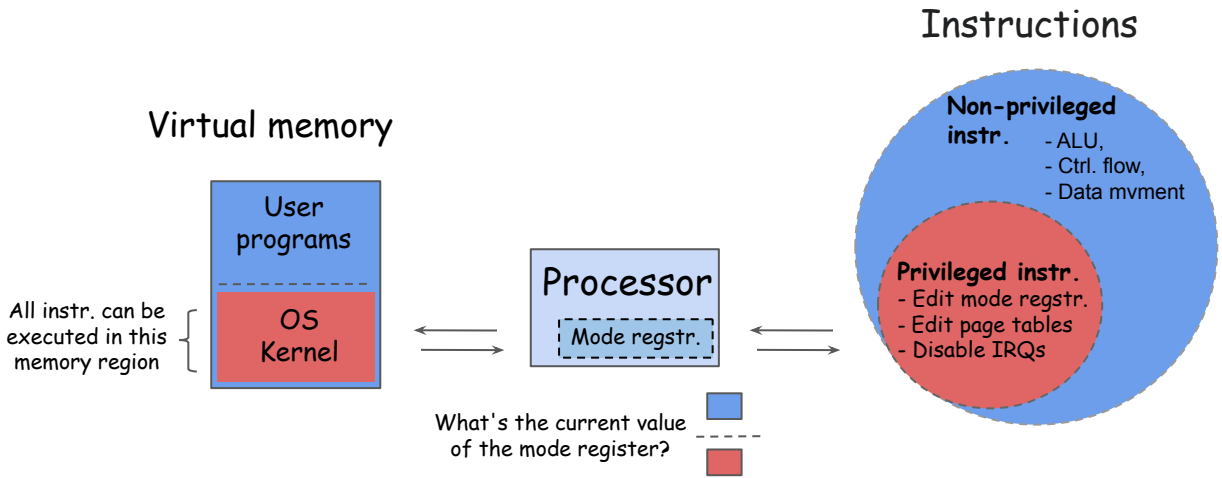
Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware



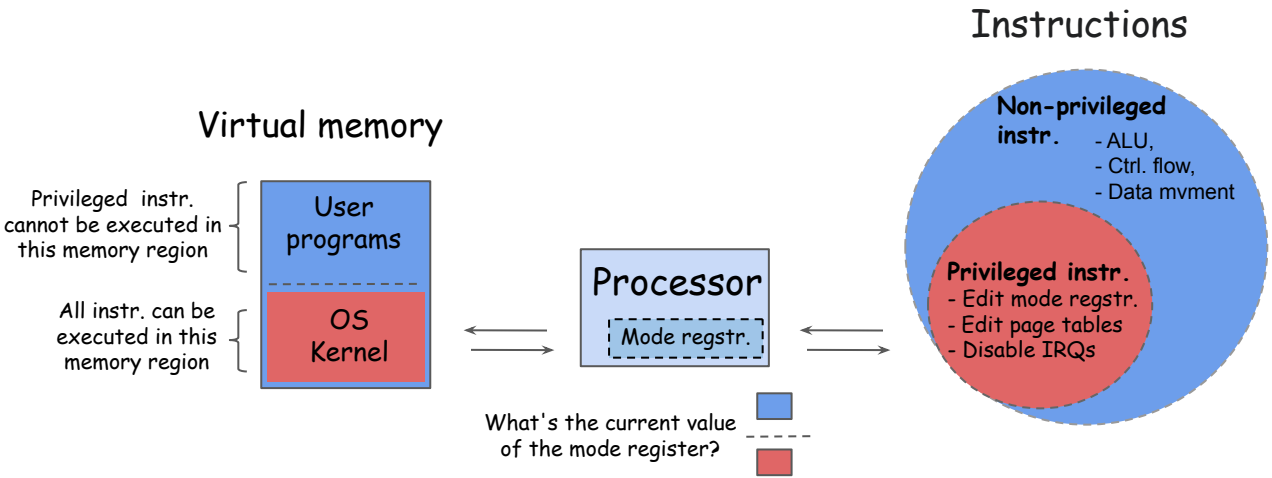
Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware



Implementing PoLP (Dual-mode execution)

- In simple words: A trusted portion of the code (the OS) must have full control of the hardware

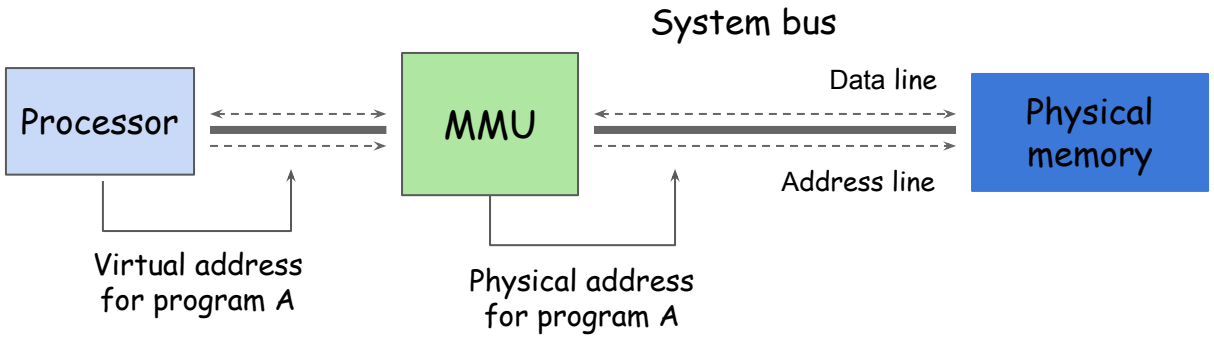


Implementing preemption

- In simple words: The OS should be able to periodically gain control of the processor regardless of what programs are executing

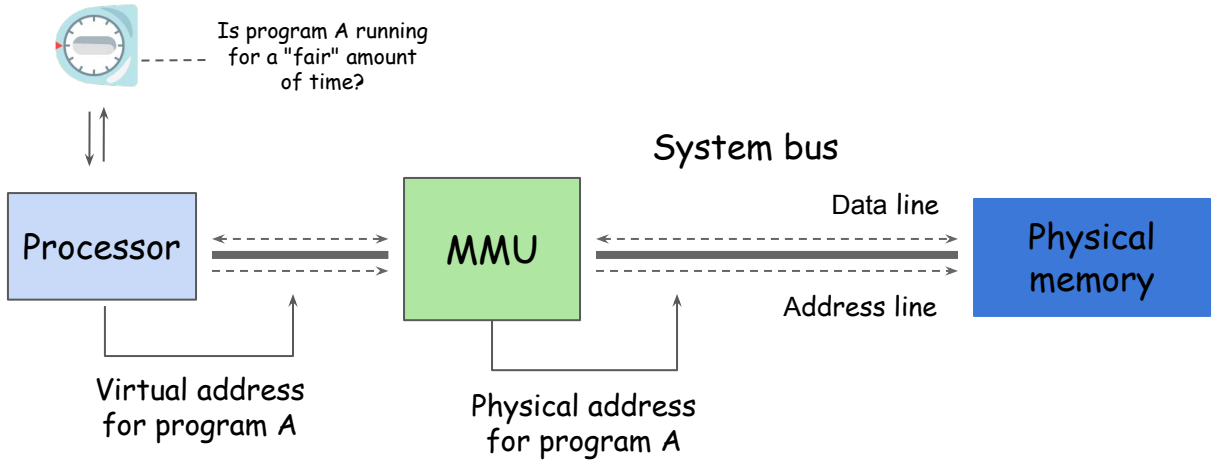
Implementing preemption

- In simple words: The OS should be able to periodically gain control of the processor regardless of what programs are executing

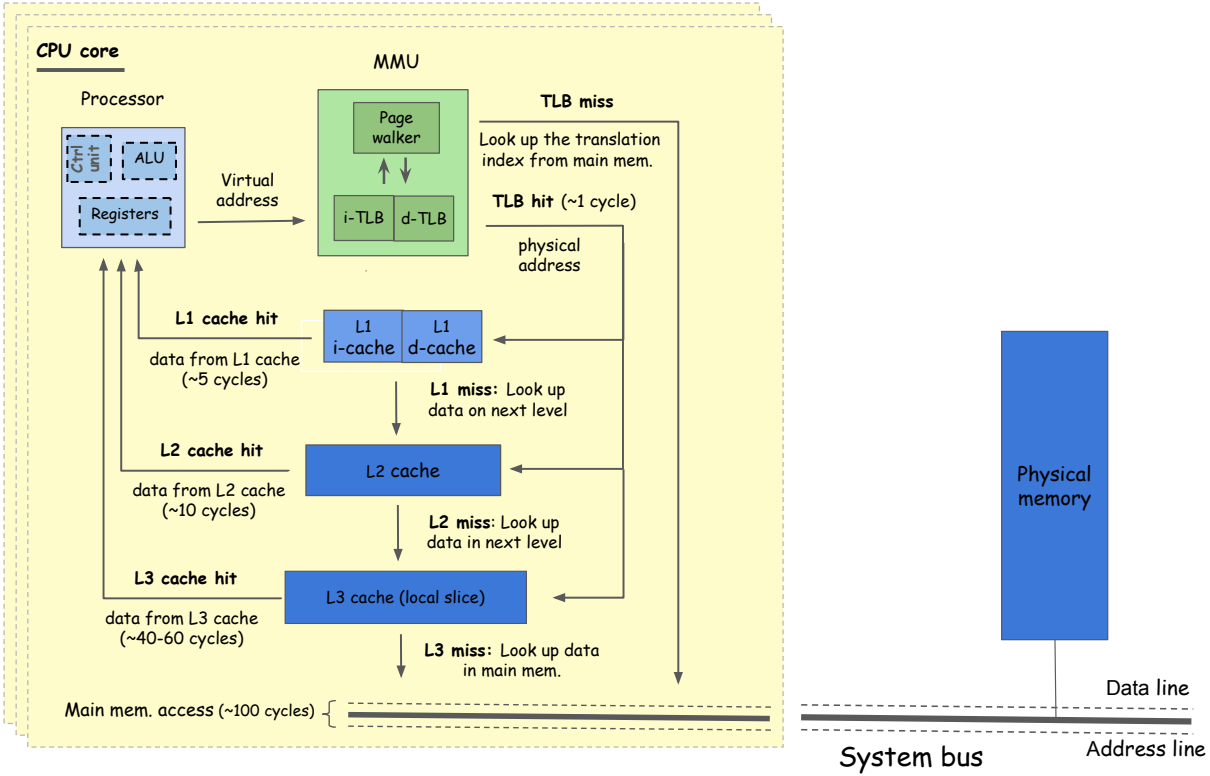


Implementing preemption (Timer interrupts)

- In simple words: The OS should be able to periodically gain control of the processor regardless of what programs are executing

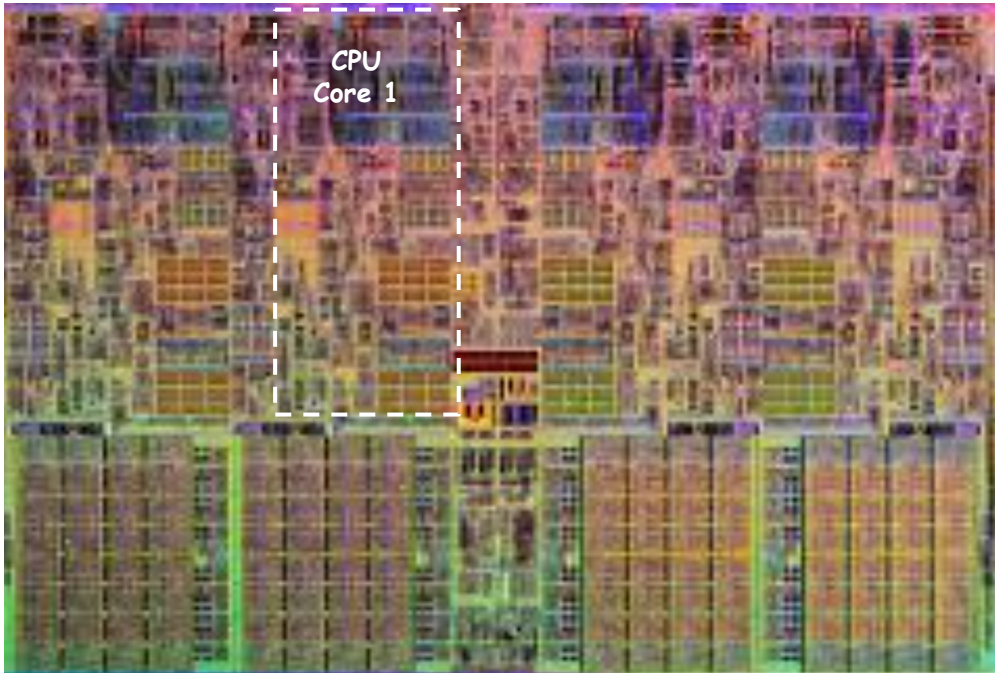


Hardware components leveraged by OS mechanisms



Reality check on hardware's scale

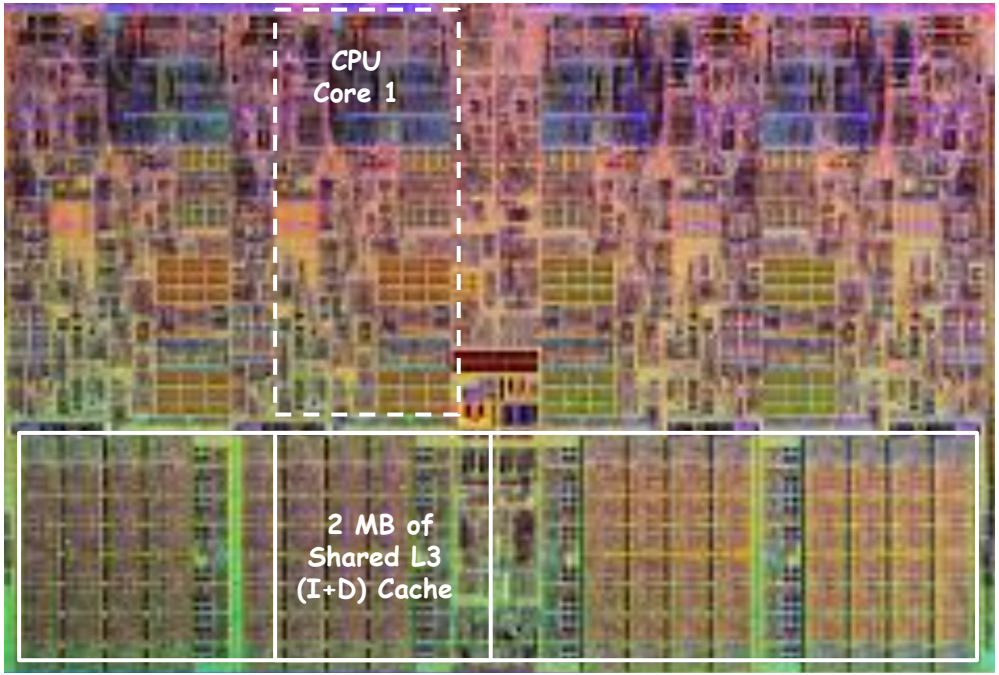
Nahalem Intel® core i7



1.8 cm

Reality check on hardware's scale

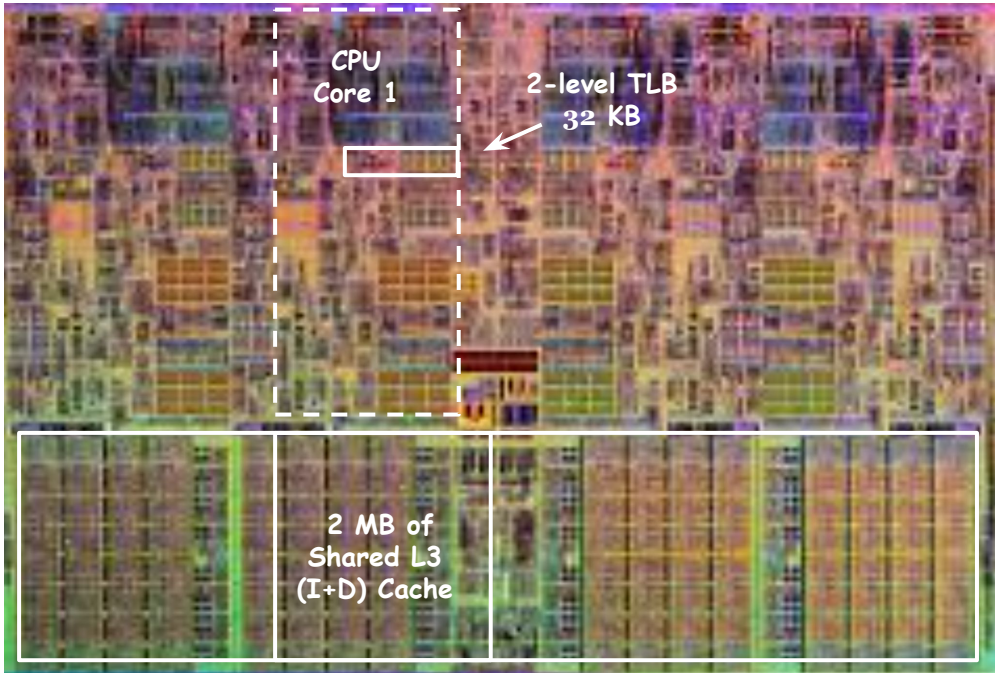
Nahalem Intel® core i7



1.8 cm

Reality check on hardware's scale

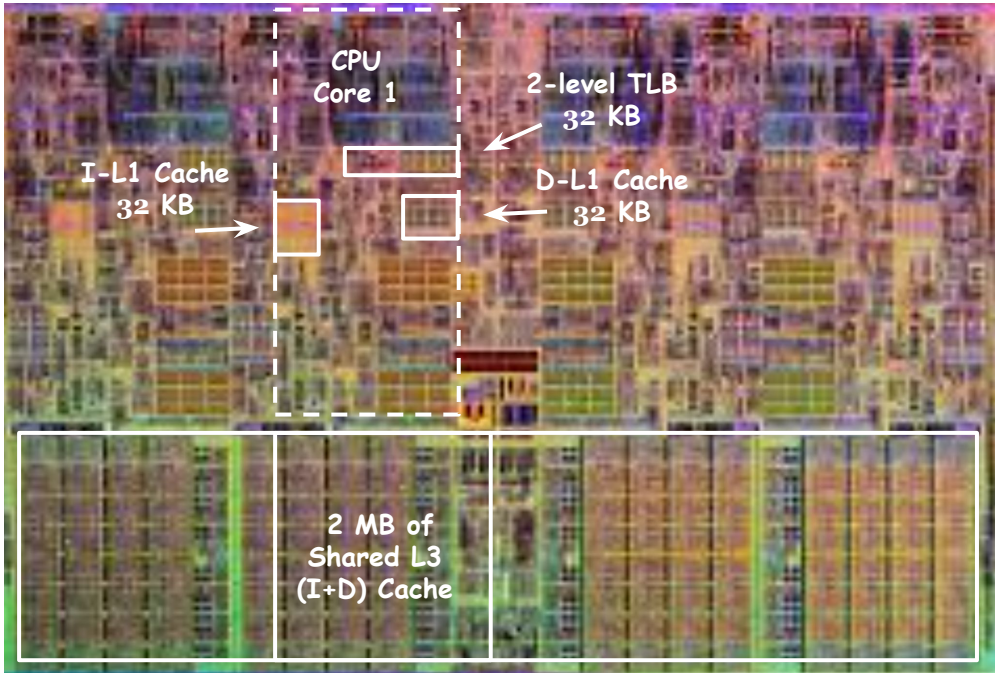
Nahalem Intel® core i7



1.8 cm

Reality check on hardware's scale

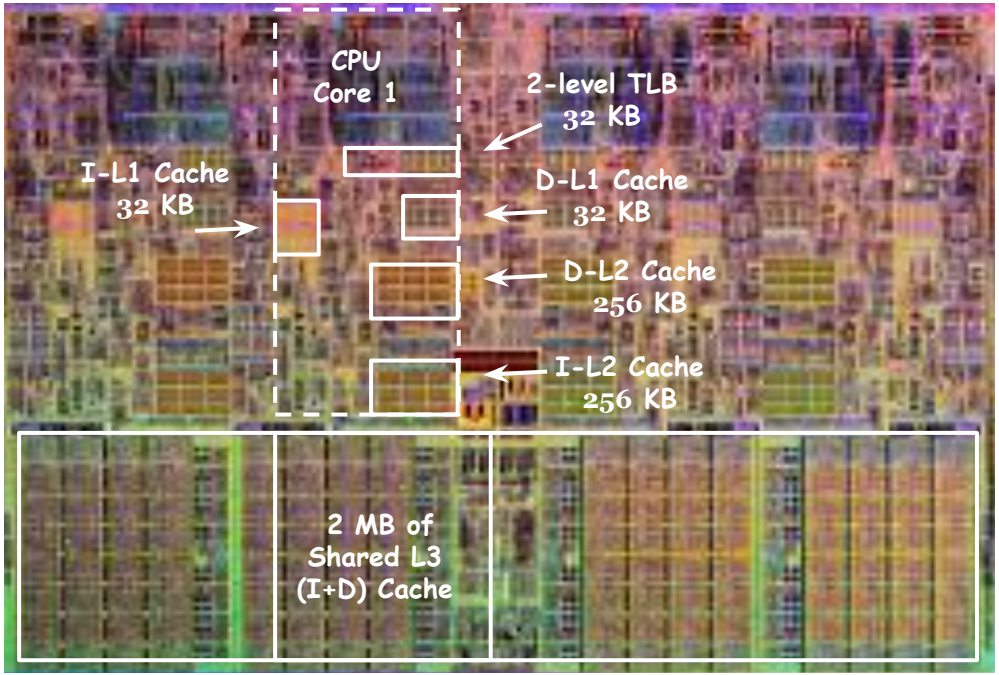
Nahalem Intel® core i7



1.8 cm

Reality check on hardware's scale

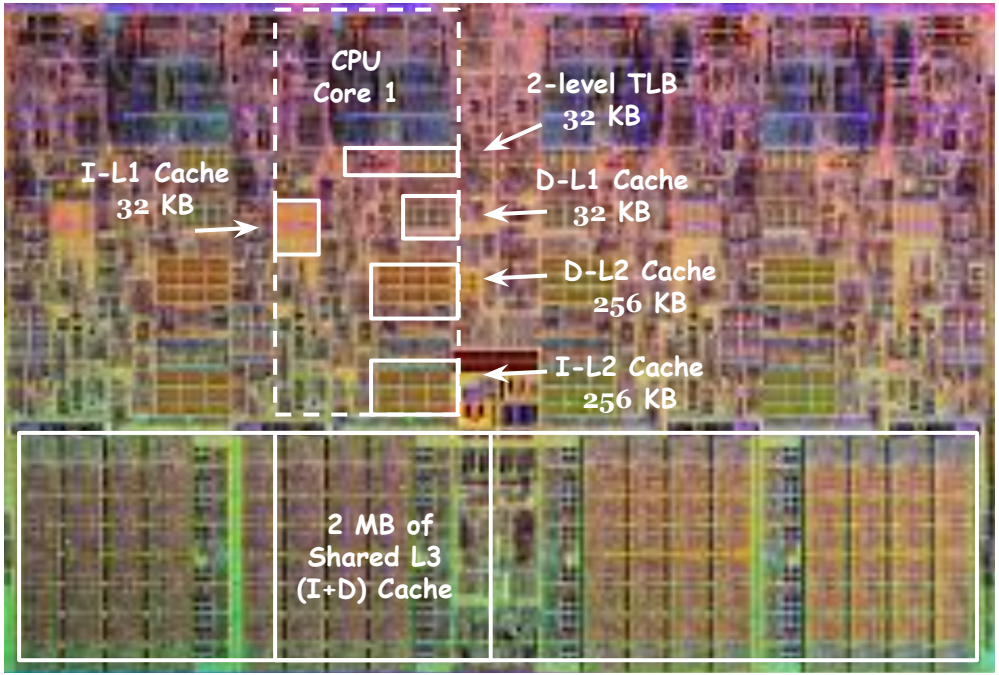
Nahalem Intel® core i7



1.8 cm

Reality check on hardware's scale

Nahalem Intel® core i7

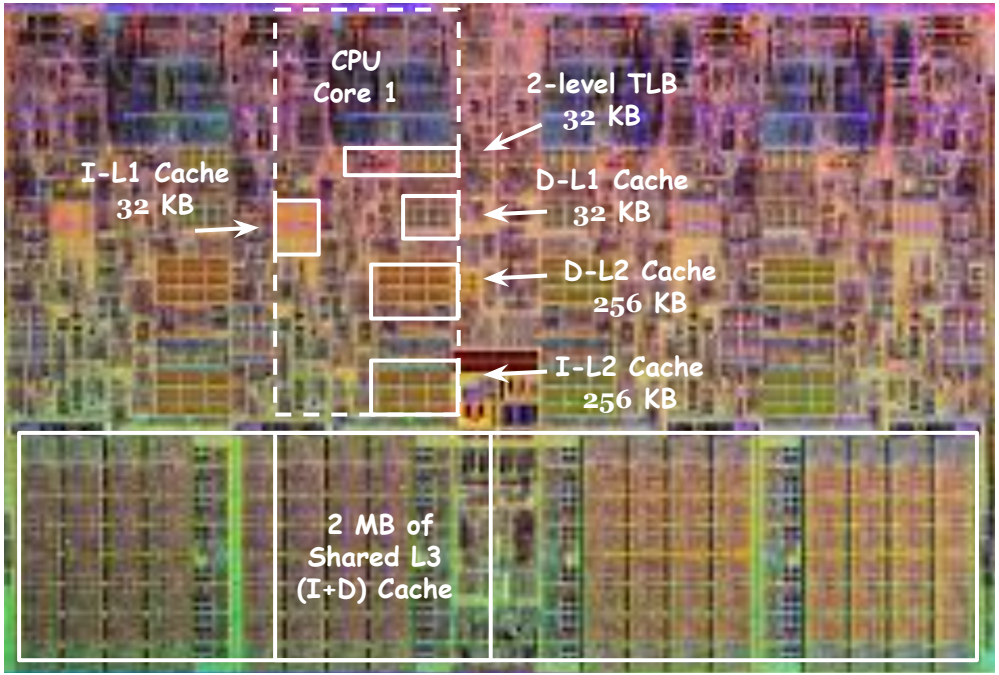


1.8 cm

Access times (1 clock cycle ~ 0.5ns)

Reality check on hardware's scale

Nahalem Intel® core i7

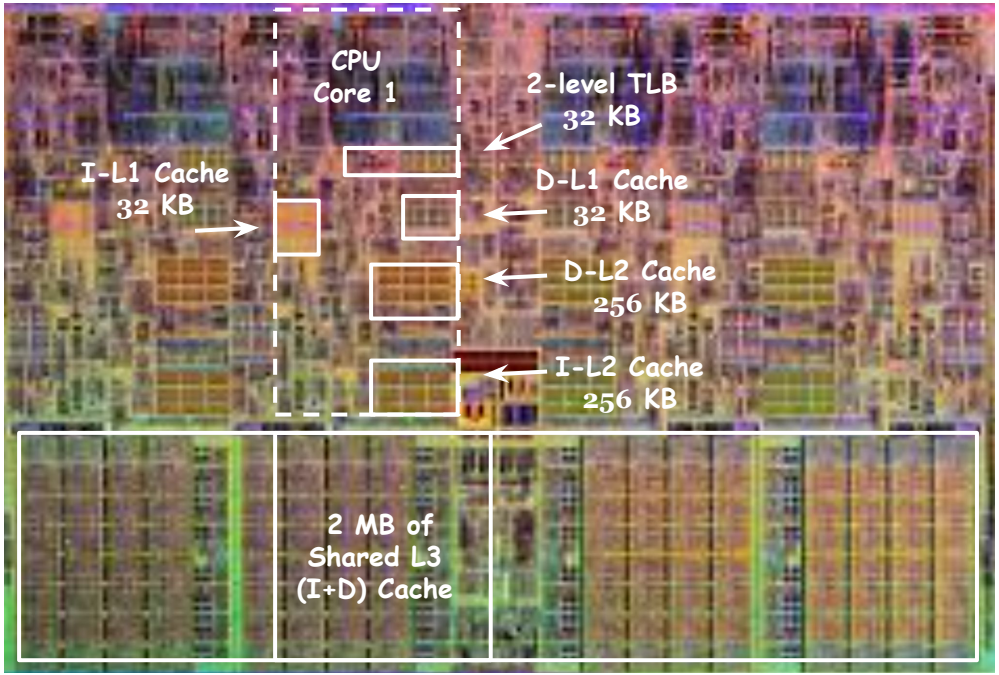


Access times (1 clock cycle ~ 0.5ns)

- L1 Cache (hit): ~ 5 clock cycles

Reality check on hardware's scale

Nahalem Intel® core i7

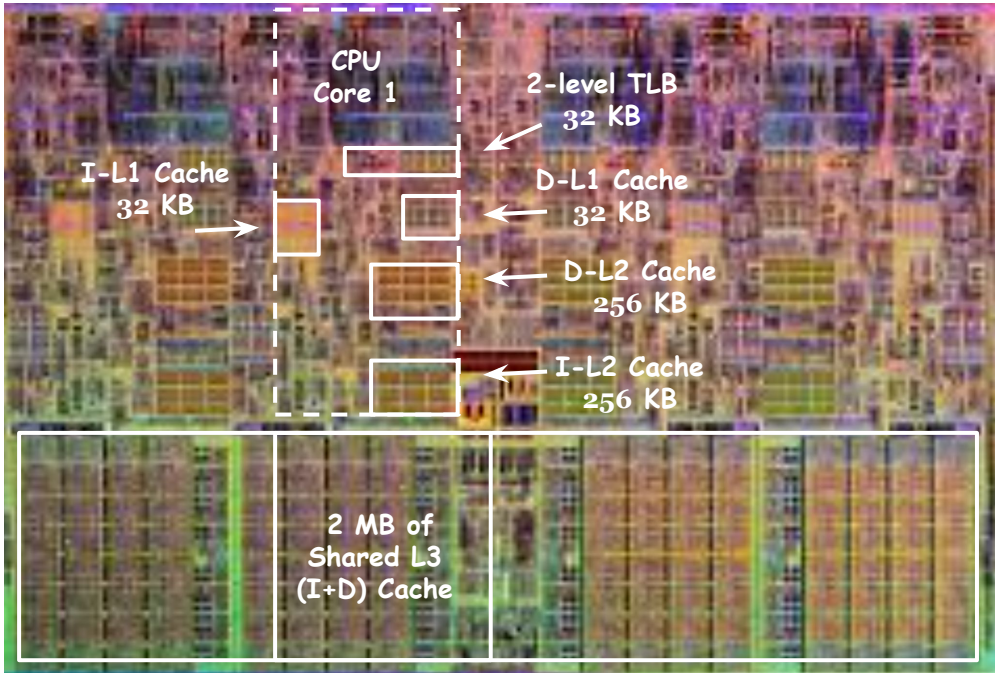


Access times (1 clock cycle ~ 0.5ns)

- L1 Cache (hit): ~ 5 clock cycles
- L2 Cache (hit): ~10 clock cycles

Reality check on hardware's scale

Nahalem Intel® core i7



1.8 cm

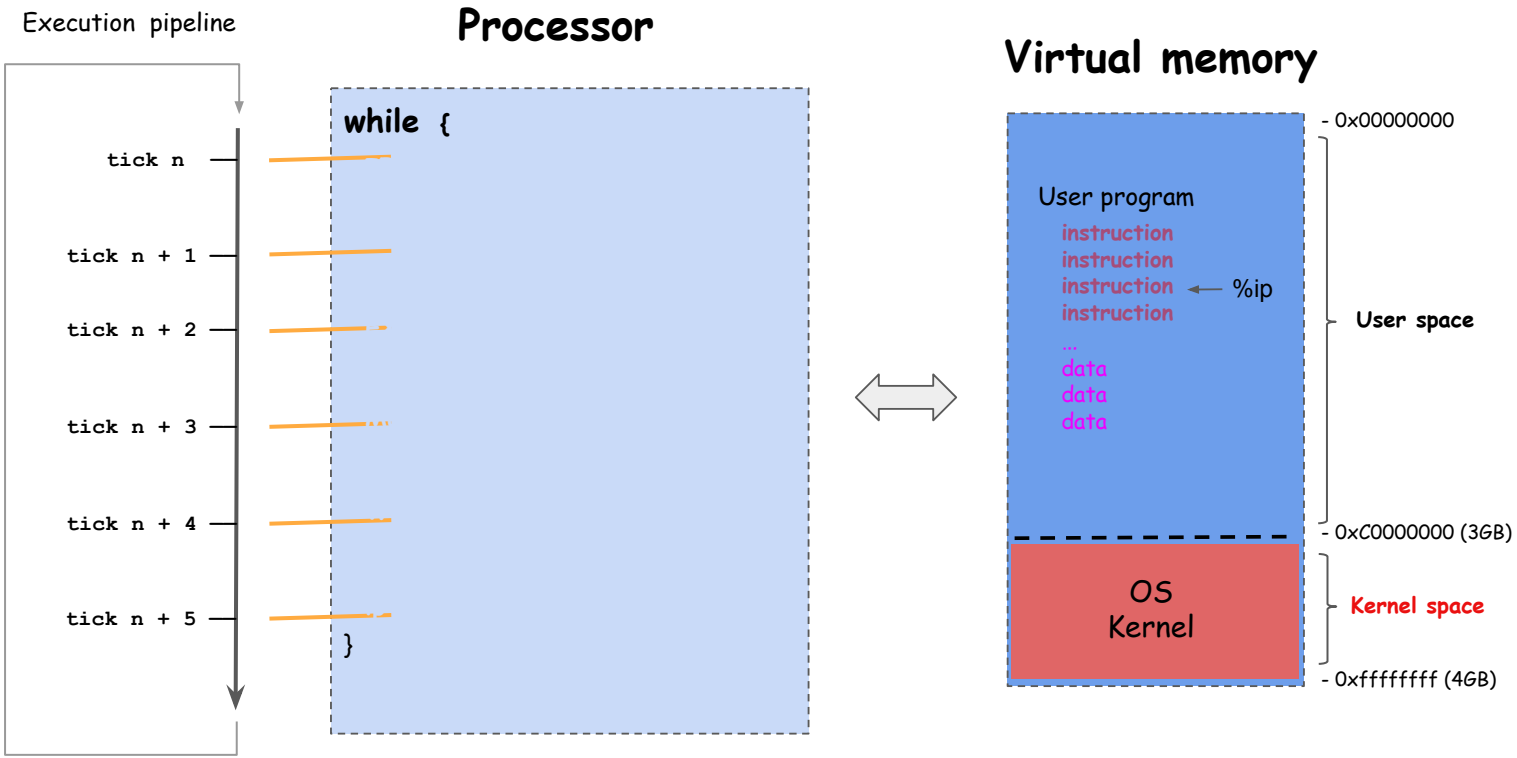
Access times (1 clock cycle ~ 0.5ns)

- L1 Cache (hit): ~ 5 clock cycles
- L2 Cache (hit): ~10 clock cycles
- L3 Cache (hit): ~40-60 clock cycles

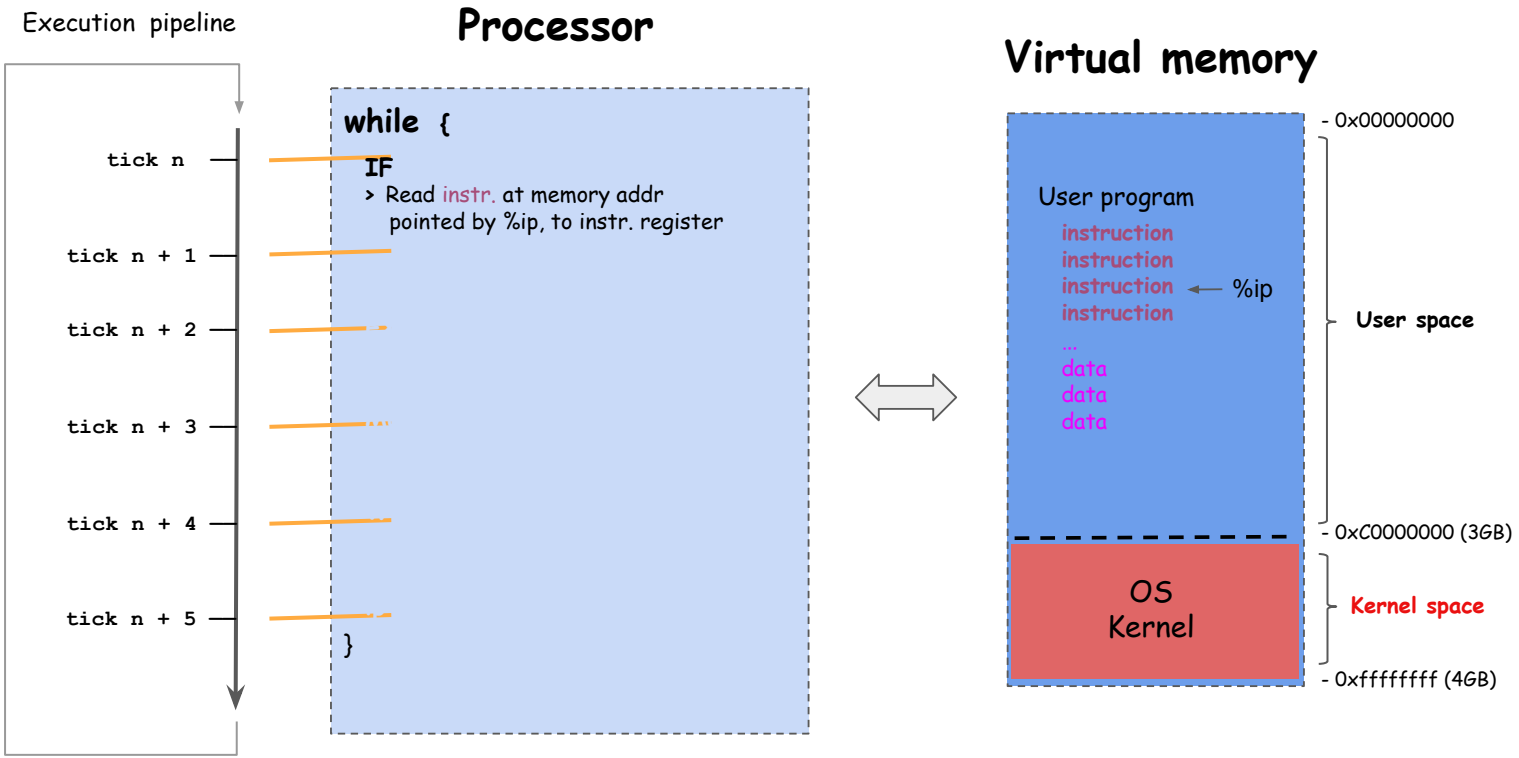
Summary so far

- Introduction (7/10)
  - Q1: What is a OS?
  - Q2: Why need an OS?
  - Q3: Desirable properties for an OS?
  - Q4: Abstract hardware model?
  - Q5: Design principles?
  - Q6: Hardware support to enforce design principles?
  - Q7: Basic hardware components so far?

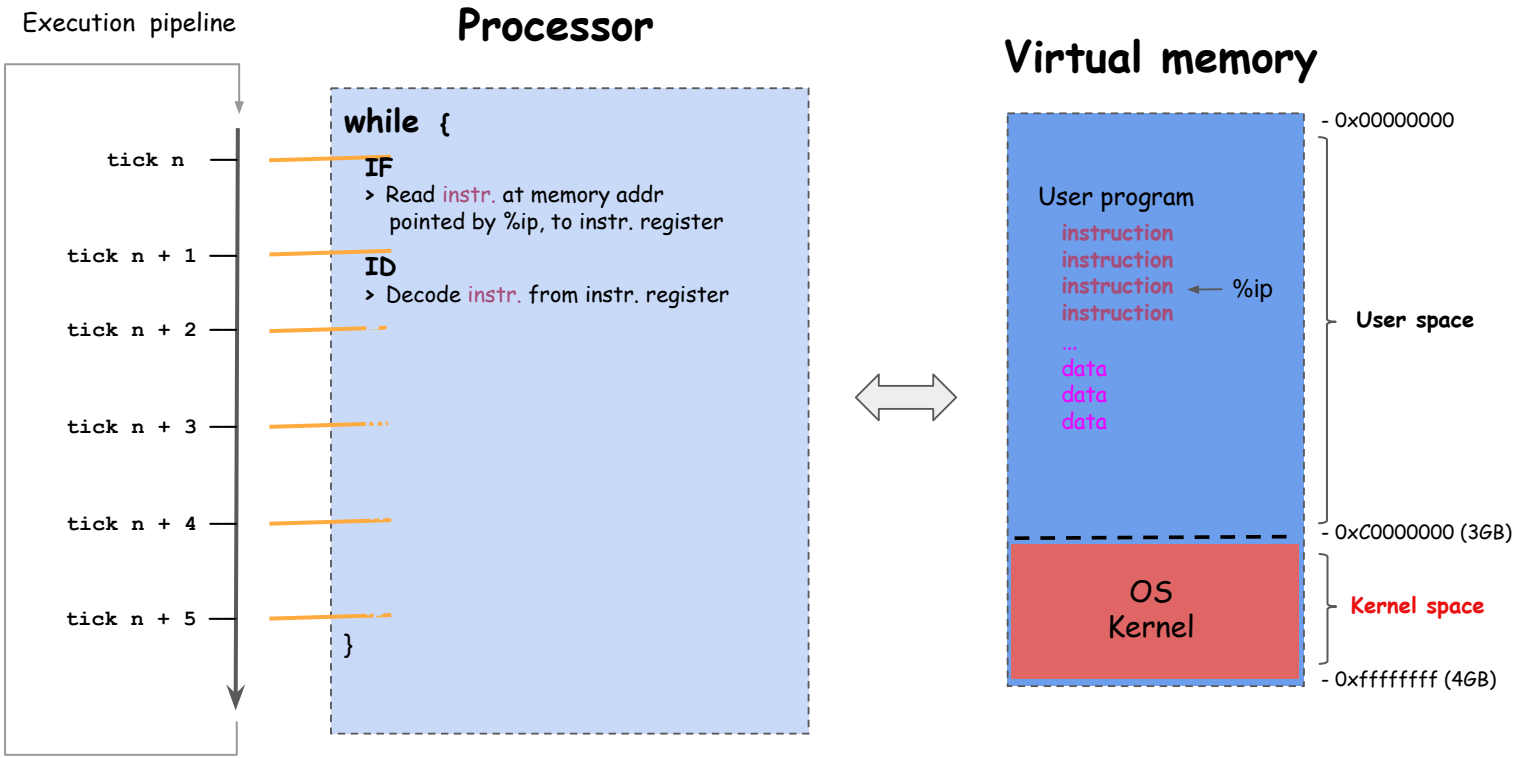
How does the processor execute programs?



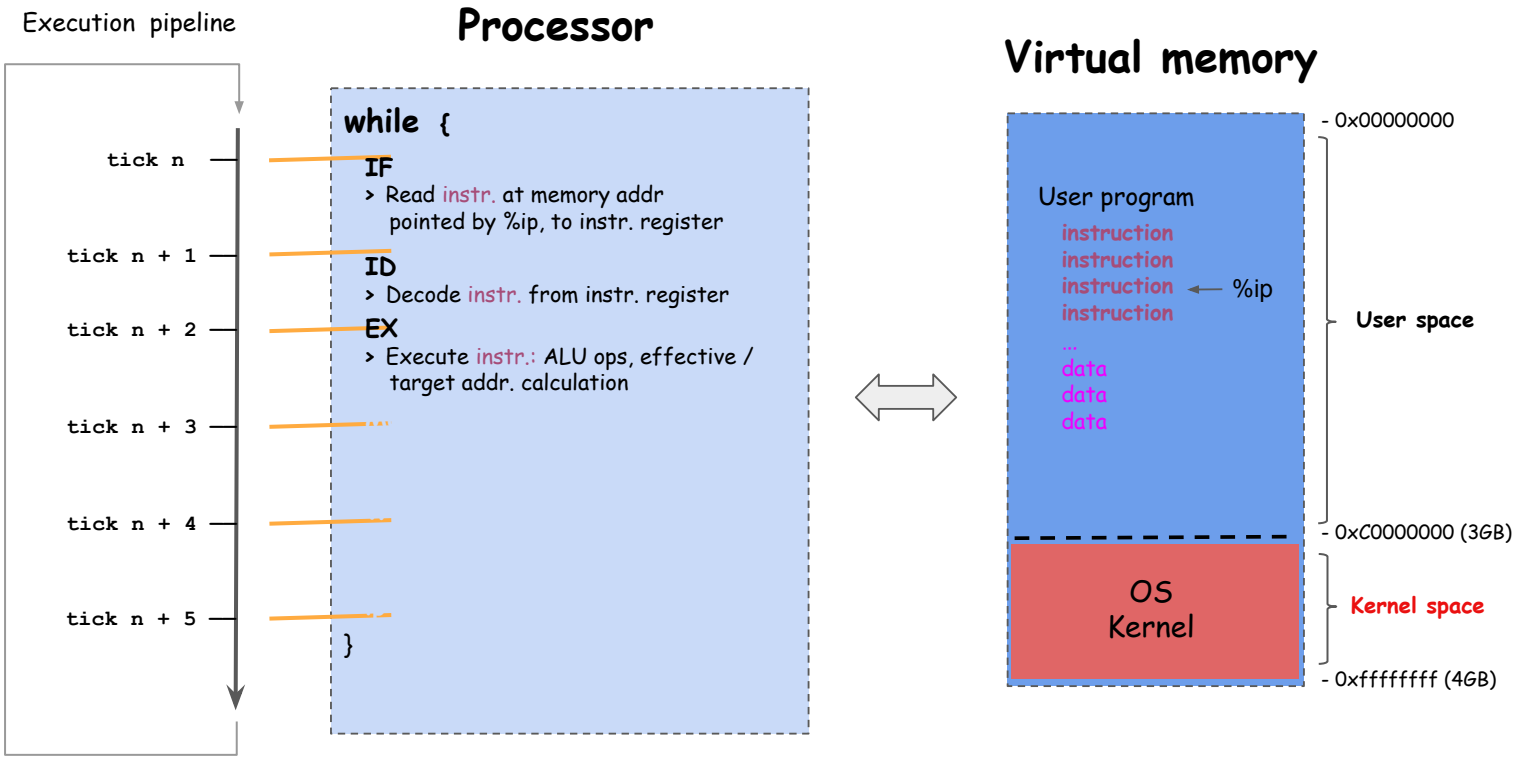
How does the processor execute programs?



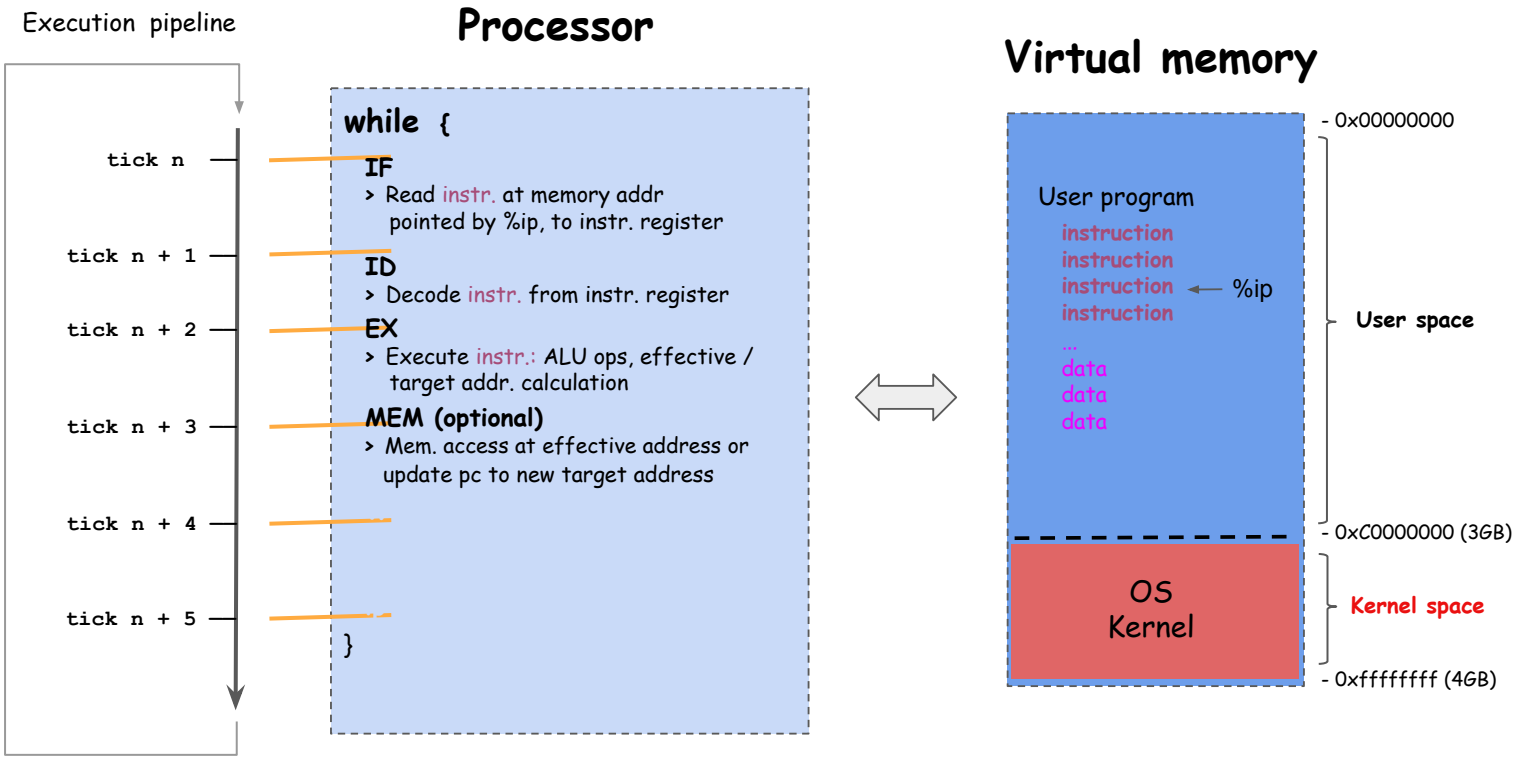
How does the processor execute programs?



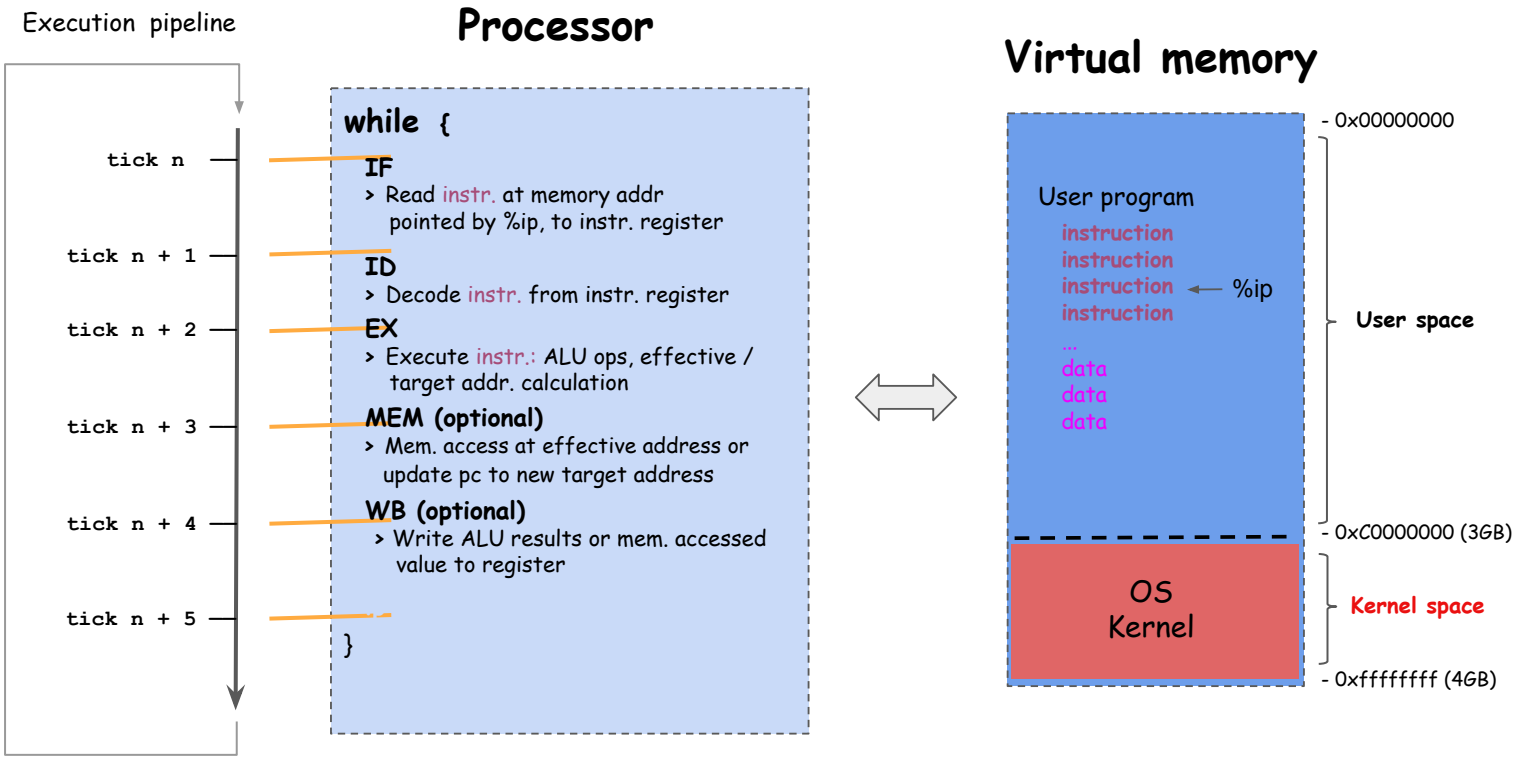
How does the processor execute programs?



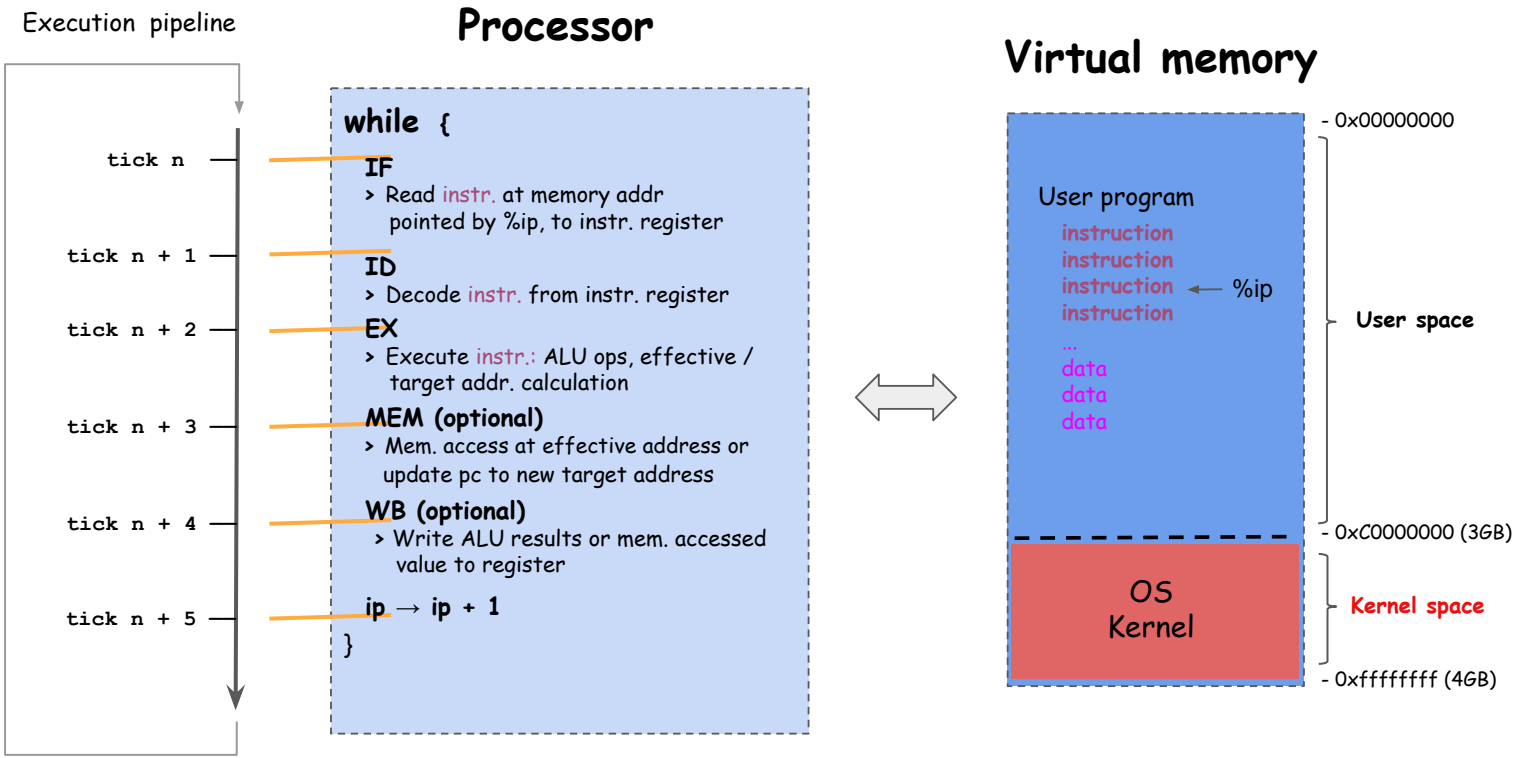
How does the processor execute programs?



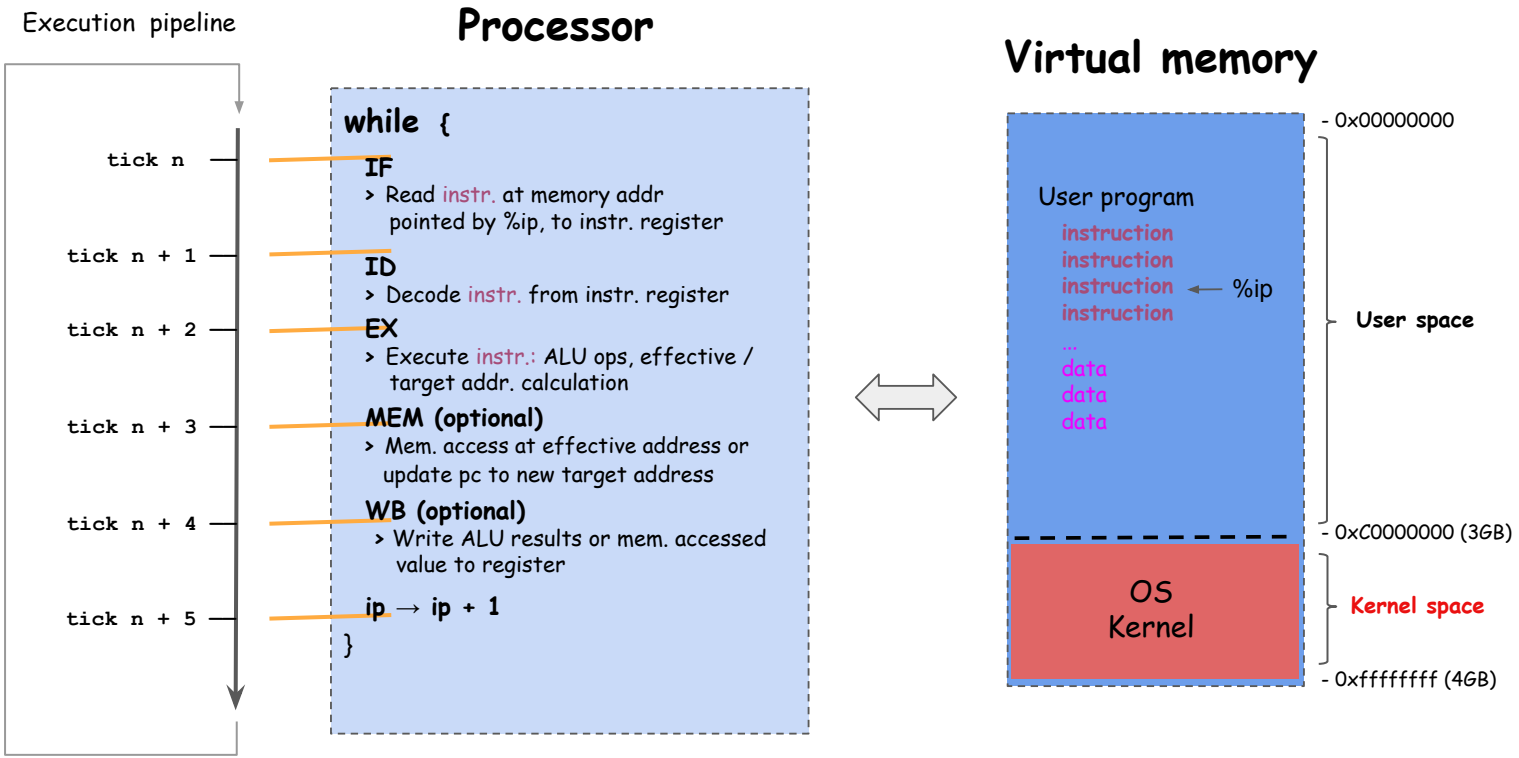
How does the processor execute programs?



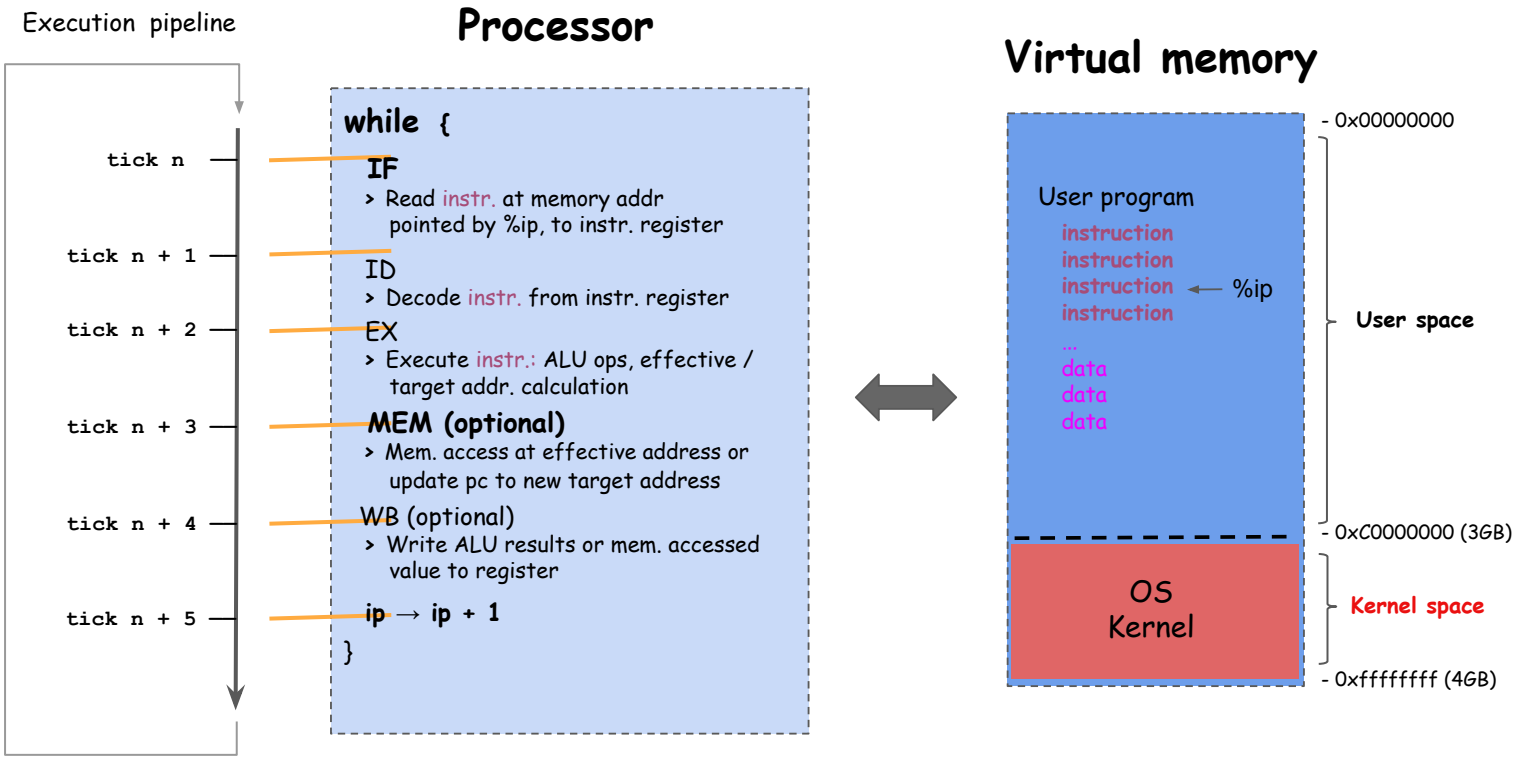
How does the processor execute programs?



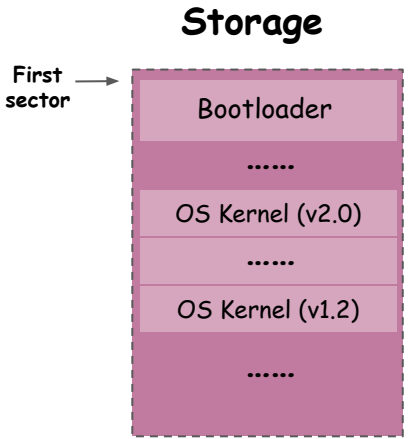
When may the processor access memory?



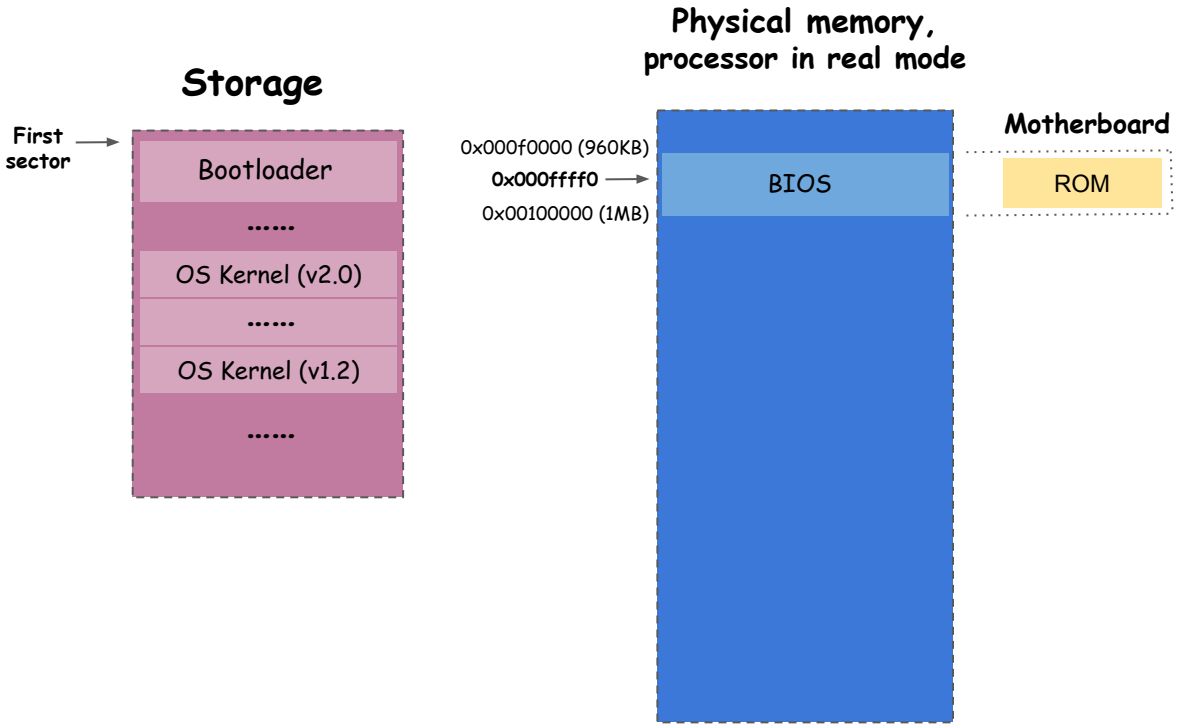
When may the processor access memory?



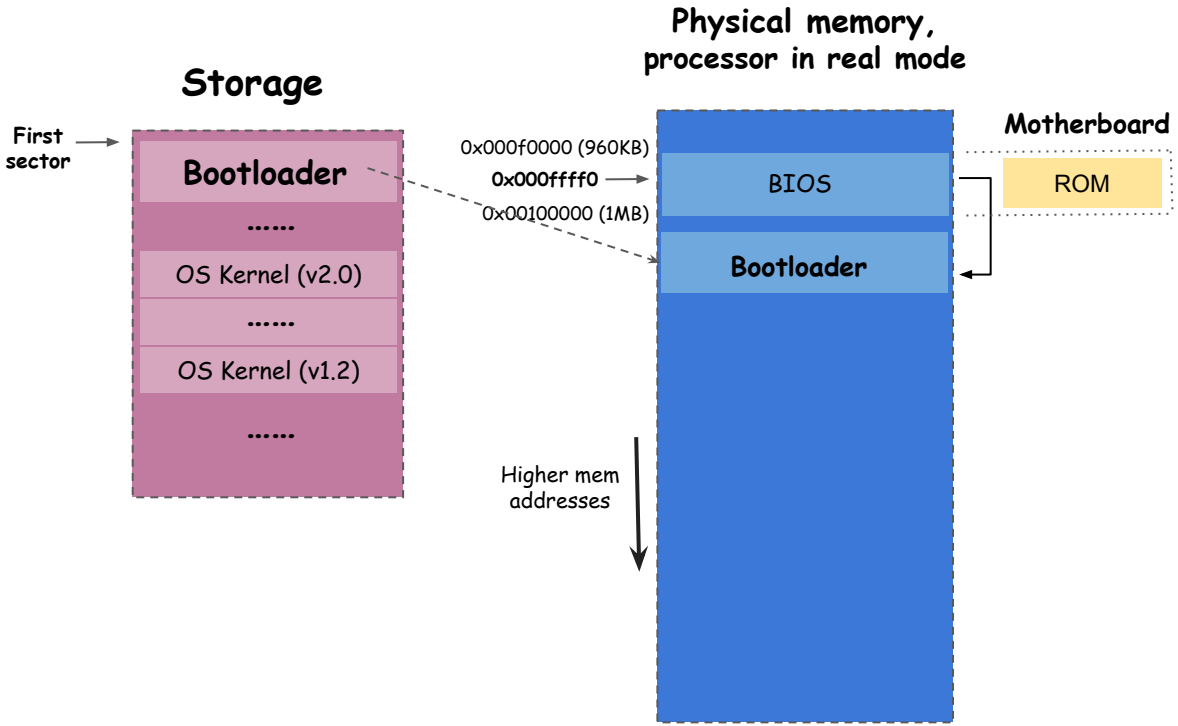
How does the system boot?



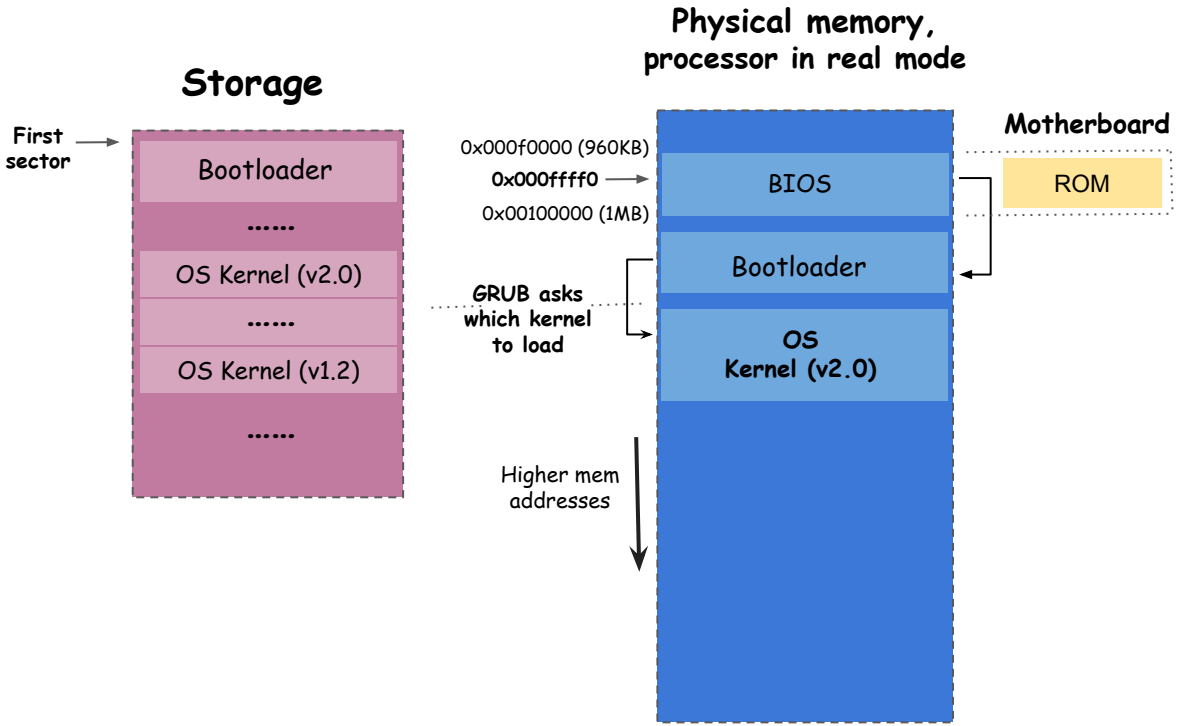
How does the system boot?



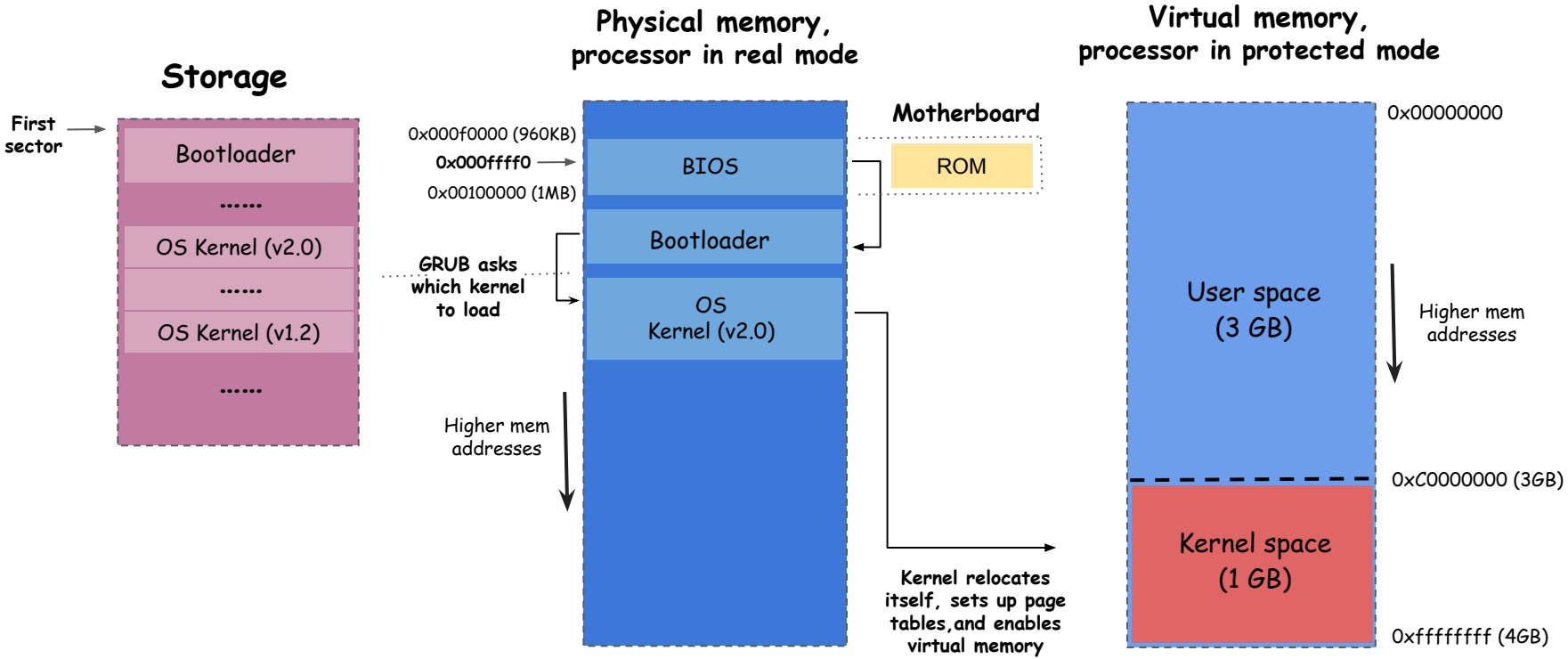
How does the system boot?



How does the system boot?



How does the system boot?



Summary so far

- **Introduction (10/10)**
  - Q1: What is a OS?
  - Q2: Why need an OS?
  - Q3: Desirable properties for an OS?
  - Q4: Abstract hardware model?
  - Q5: Design principles?
  - Q6: Hardware support to enforce design principles?
  - Q7: Basic hardware components so far?
  - Q8: Reality check on hardware's scale and access times?
  - Q9: How does the processor execute programs?
  - Q10: How does the system boot?

Overview

- We'll follow a question-oriented approach

Overview

- We'll follow a question-oriented approach
  - Intro [Q: What is an OS?]
  - Events [Q: When does the OS run?]
  - Runtime [Q: How does a program look like in memory?]

* Basic (H/W & S/W)

Overview

- We'll follow a question-oriented approach
  - Intro [Q: What is an OS?]
  - Events [Q: When does the OS run?]
  - Runtime [Q: How does a program look like in memory?]
  - Processes [Q: What is a process?]
  - Threads [Q: What is a thread?]
  - Files [Q: What is a file descriptor?]

* Basic (H/W & S/W)
* **Abstractions**

Overview

- We'll follow a question-oriented approach
  - Intro [Q: What is an OS?]
  - Events [Q: When does the OS run?]
  - Runtime [Q: How does a program look like in memory?]
  - Processes [Q: What is a process?]
  - IPC [Q: How do processes communicate?]
  - Threads [Q: What is a thread?]
  - Synchronization [Q: What goes wrong w/o synchronization?]
  - Files [Q: What is a file descriptor?]

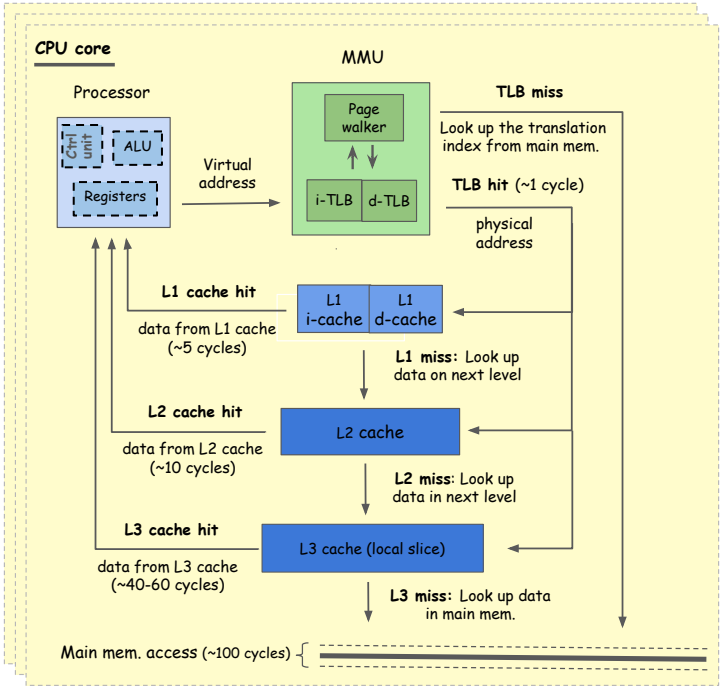
* Basic (H/W & S/W)
* **Abstractions**
* **Primitives**

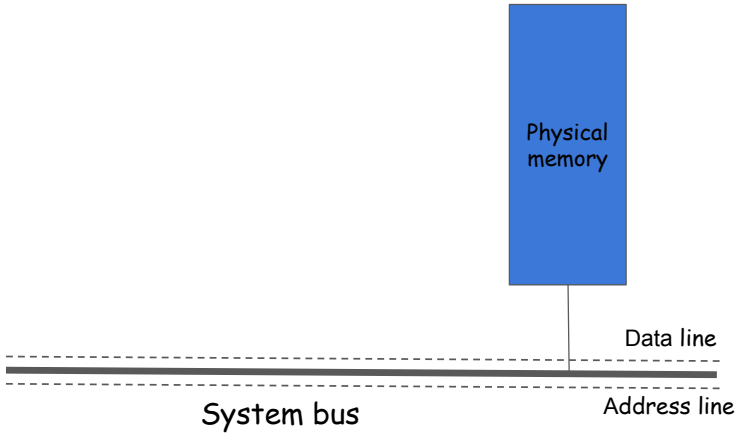
Overview

- We'll follow a question-oriented approach
  - Intro [Q: What is an OS?]
  - Events [Q: When does the OS run?]
  - Runtime [Q: How does a program look like in memory?]
  - Processes [Q: What is a process?]
  - IPC [Q: How do processes communicate?]
  - Threads [Q: What is a thread?]
  - Synchronization [Q: What goes wrong w/o synchronization?]
  - Time Management [Q: What is scheduling?]
  - Memory Management [Q: What is virtual memory?]
  - Files [Q: What is a file descriptor?]
  - Storage Management [Q: How do we allocate disk space to files?]

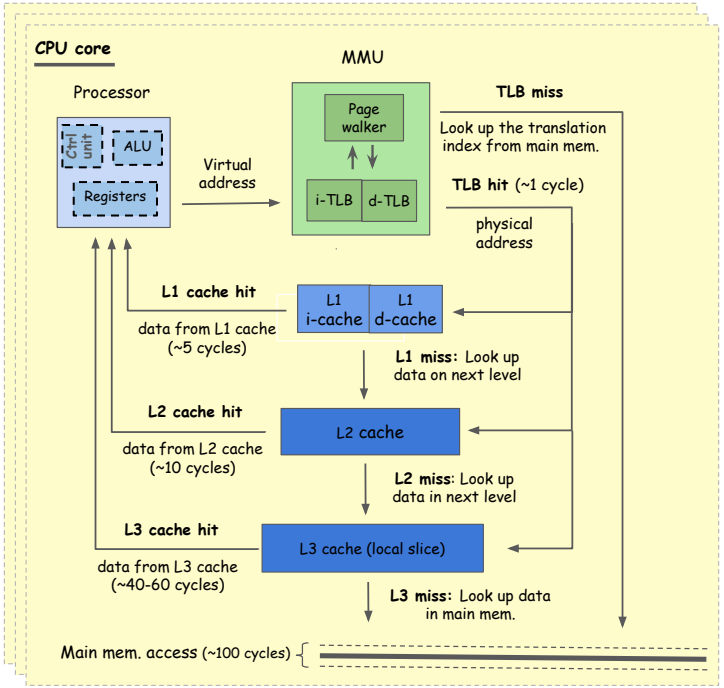
* Basic (H/W & S/W)
* **Abstractions**
* **Primitives**
* **Mechanisms**

What are we missing?



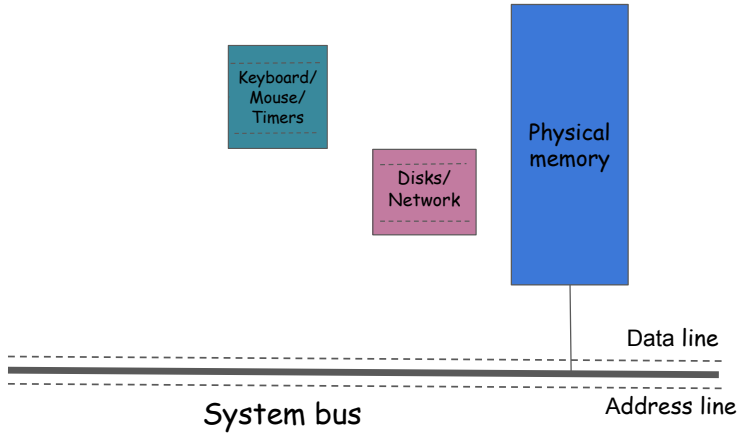


Handling I/O devices

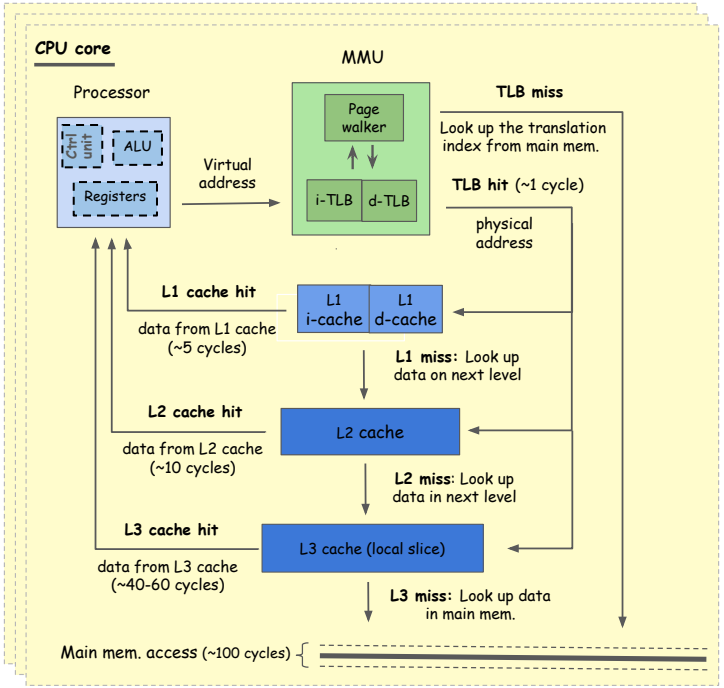


- Keyboard? Mouse?
- Disk? Network?

} I/O devices

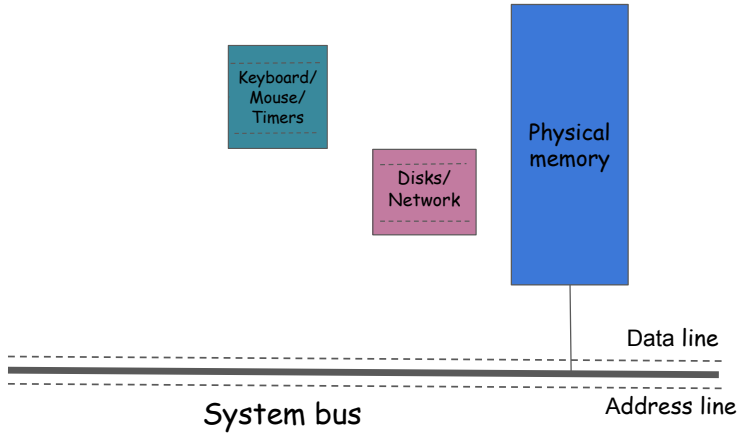


Handling I/O devices

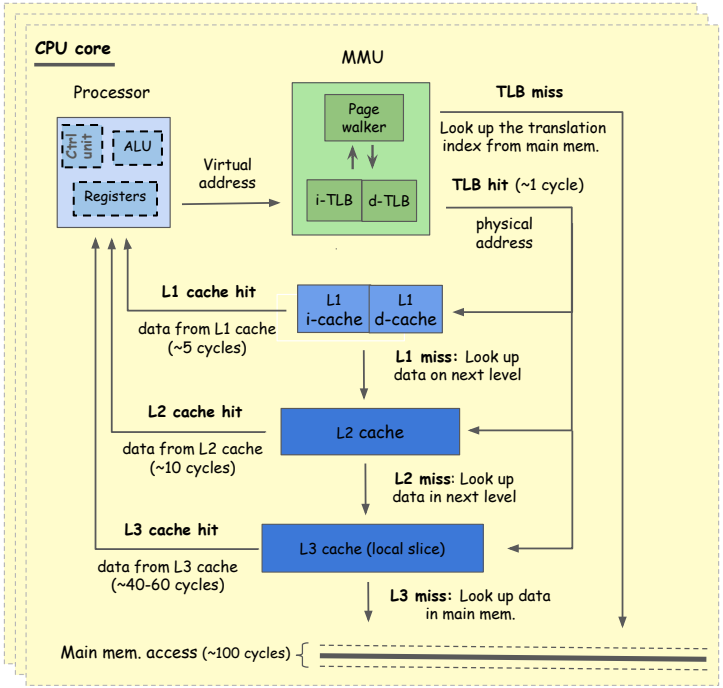


I/O devices

› Need a way to get processor's attention



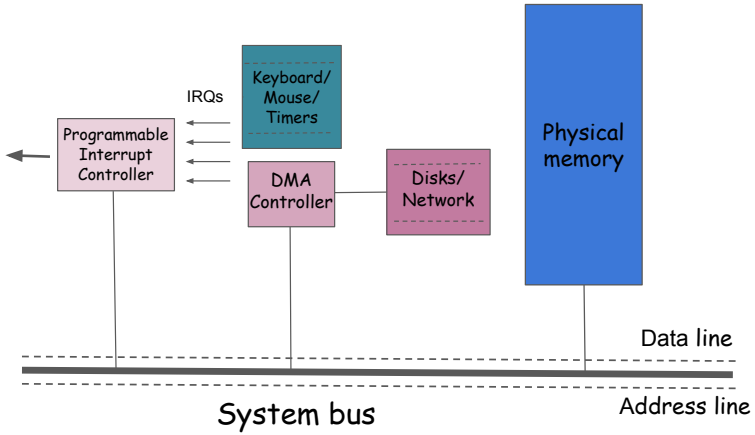
Handling I/O devices



I/O devices

› Need a way to get processor's attention

- Interrupts, Interrupt Requests (IRQs)
- Direct Memory, Access (DMA) controler…



Handling I/O devices

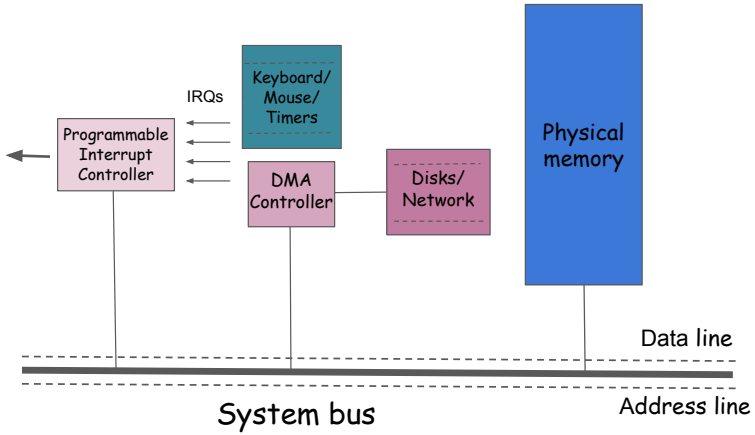
Execution pipeline

```
while {
    IF - Instruction Fetch
    ID - Instruction Decode
    EXE - Execute Instruction
    MEM - Memory Access
    WB - Write back
    ip → ip + 1                          Instruction
    if (Interrupt) {                     boundary
        What??
    }
}
```

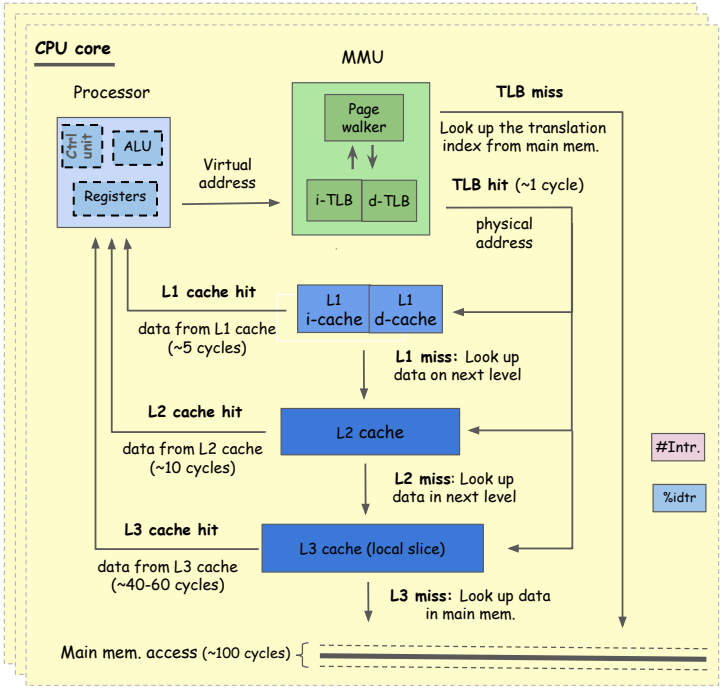
I/O devices

› Need a way to get processor's attention

- Interrupts, Interrupt Requests (IRQs)
- Direct Memory, Access (DMA) controler…



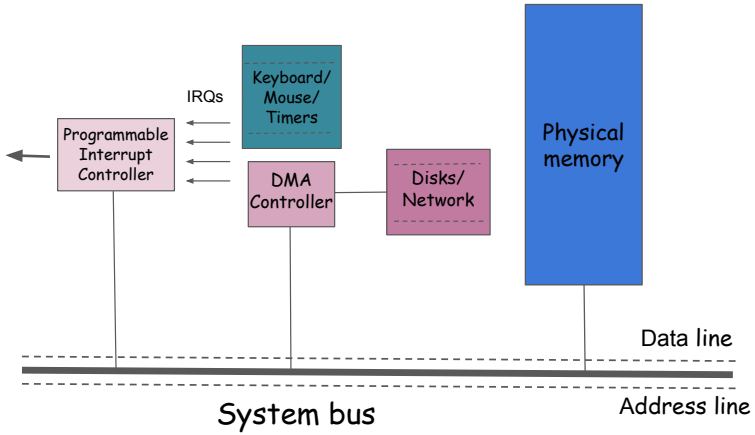
Handling I/O devices



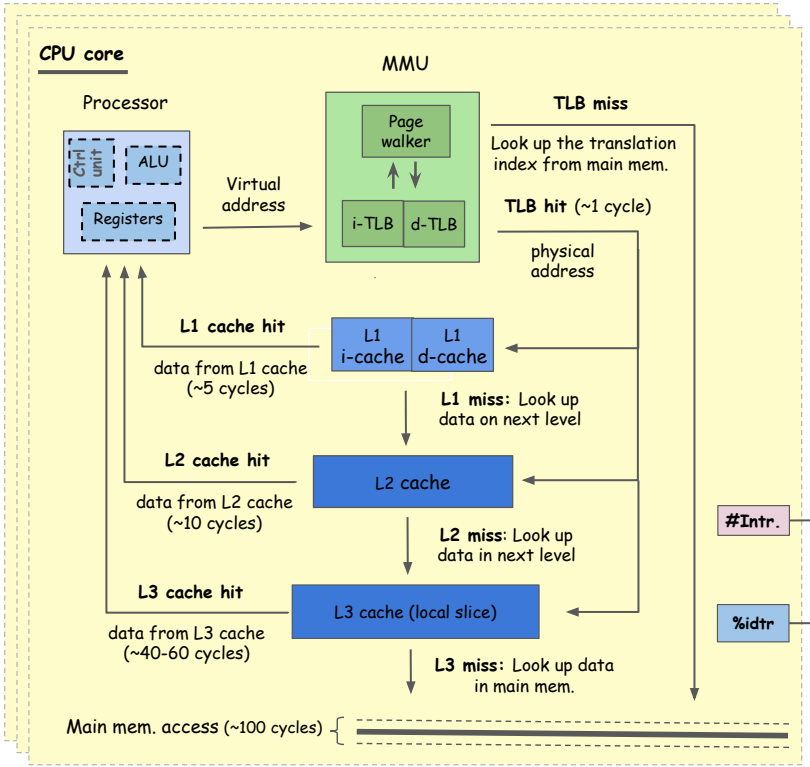
I/O devices

› Need a way to get processor's attention

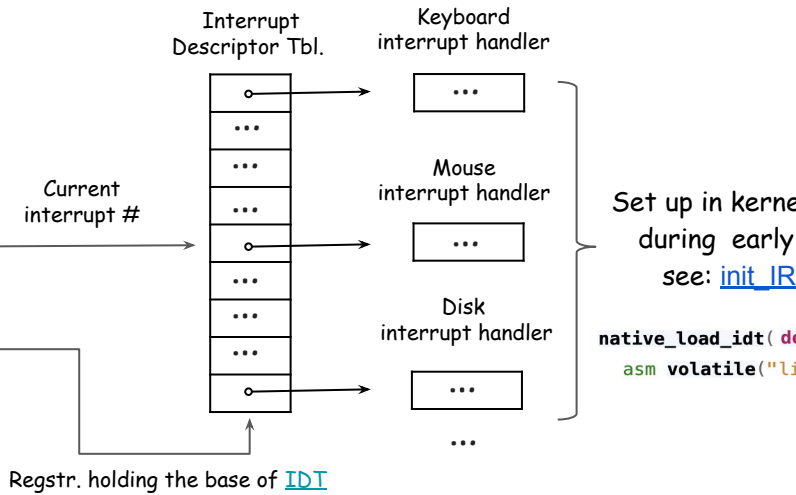
- Interrupts, Interrupt Requests (IRQs)
- Direct Memory, Access (DMA) controler…



Interrupt handling



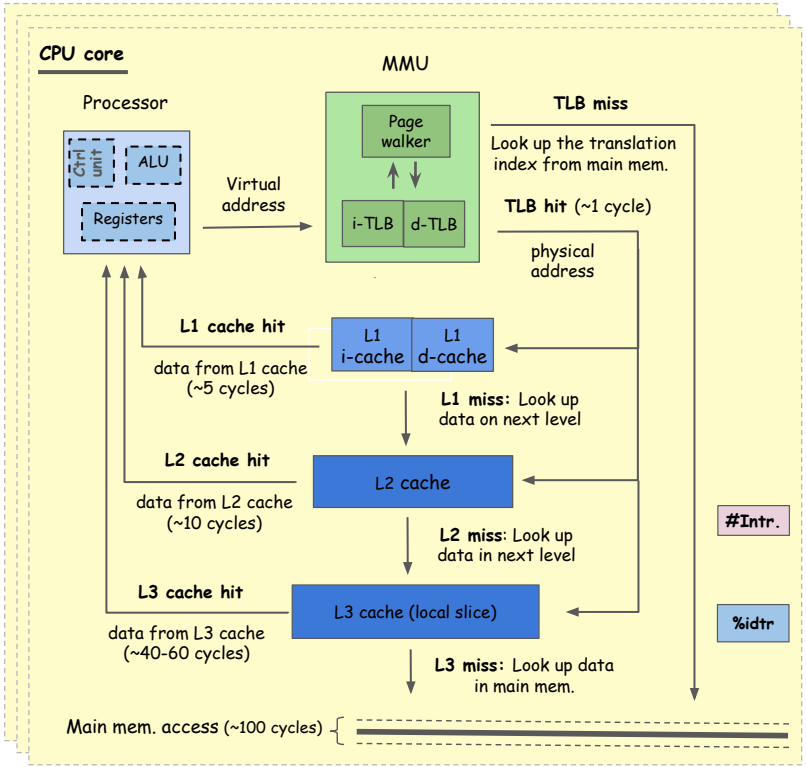
do_handle_interrupt()



Set up in kernel space during early boot
see: init_IRQ()

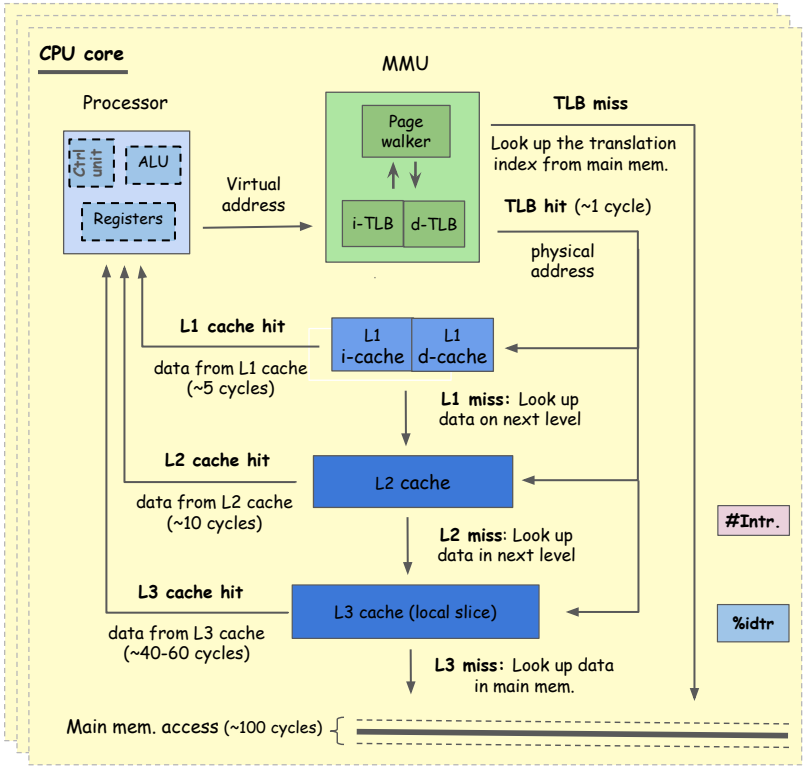
```
native_load_idt( desc_ptr *dtr)
  asm volatile("lidt %0"::"m" (*dtr));
```

Interrupt handling



do_handle_interrupt()

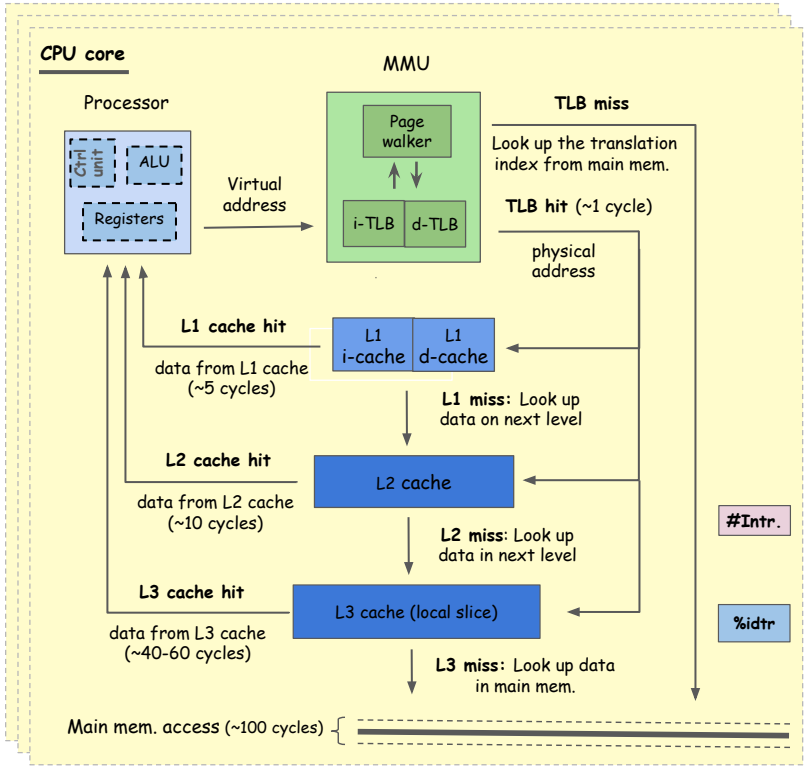
Interrupt handling



do_handle_interrupt()

- Switch stack
  - Kernel stack, if in user mode
  - Dedicated stack for critical exceptions

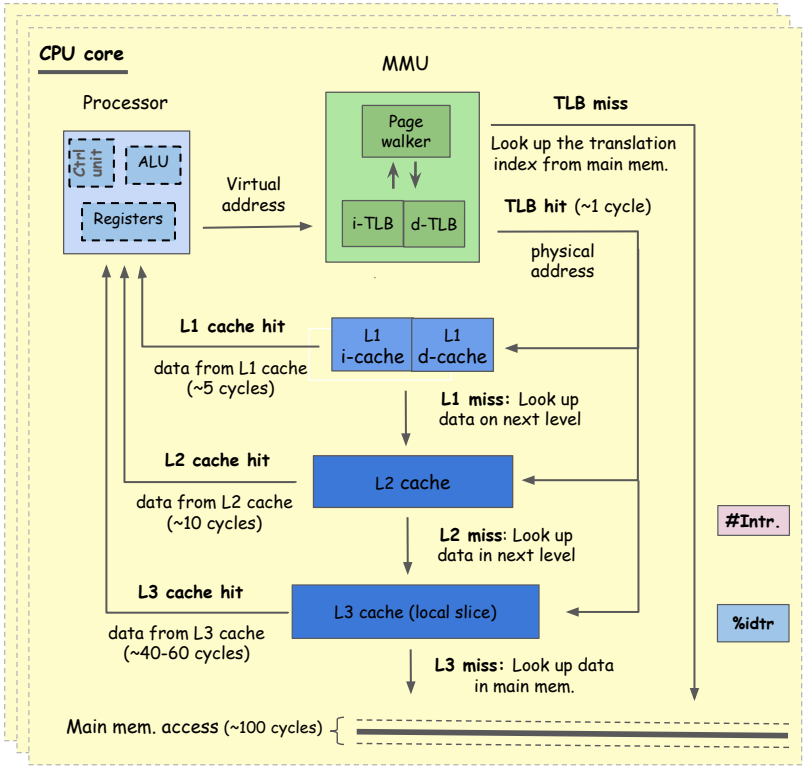
Interrupt handling



do_handle_interrupt()

- Switch stack
  - Kernel stack, if in user mode
  - Dedicated stack for critical exceptions
- Save ctrl.-state & interrupt-ctx. registers
- Save error code (if any)

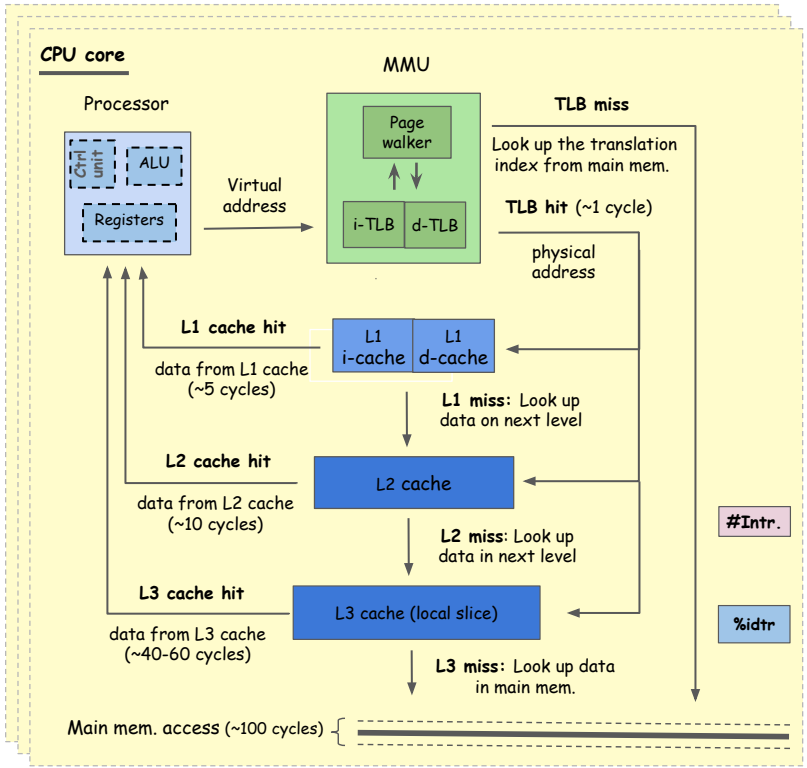
Interrupt handling



do_handle_interrupt()

- Switch stack
  - Kernel stack, if in user mode
  - Dedicated stack for critical exceptions
- Save ctrl.-state & interrupt-ctx. registers
- Save error code (if any)
- Save general-purpose registers

Interrupt handling



do_handle_interrupt()

- Switch stack
  - Kernel stack, if in user mode
  - Dedicated stack for critical exceptions
- Save ctrl.-state & interrupt-ctx. registers
- Save error code (if any)
- Save general-purpose registers
- Use #intr. and %idtr to execute the appropriate handler