Operating Systems
CS 415

Suyash Gupta

Assistant Professor

Distopia Labs and ONRG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Welcome!

Course Webpage:

https://gupta-suyash.github.io/cs415-spring26.html

Course Canvas:

https://canvas.uoregon.edu/courses/282775

Instructors

- **Suyash Gupta**
  - Office Hours: Thursday, 10-11am, Deschutes 334
- **Nihal Balivada (TA)**
  - Office Hours: Wednesday/ Friday, 11-12pm, Deschutes 335
- **Ranjitha Rani (TA)**
  - Office Hours: Monday /Tuesday, 1-2pm, Deschutes 229

TextBooks

- **Main Course Book:**
  - Operating Systems Concepts, A. Silberschatz, P. Baer Galvin, G. Gagne, Tenth Edition, ISBN: 978-1-118-06333-0.

- **Optional Reading:**
  - Operating Systems: Principles & Practice, Thomas Anderson and Michael Dahlin, Second Edition, ISBN: 978-0-9856735-2-9.

Grading Scheme

- **25% - Projects**
- **20% - Midterm**
- **20% - Final**
- **15% - Class Participation + Attendance**
- **20% - Laboratory**

Grading Rubric

- **Rubric**
  - **97% - A+**
  - **92% - A**
  - **87% - A-**
  - **82% - B+**
  - **77% - B**
  - **72% - B-**
  - **67% - C+**
  - **62% - C**

Projects

- **Each Project will be completed individually.**
- **No groups!**

Assignments

- Total **three assignments**:
  - Assignment 1 - 5%
  - Assignment 2 – 10%
  - Assignment 3 – 10%
  - Total Assignment weight: **25**%.

- **Auto-grading**:
  - We will give you scripts to test your code.
  - We will have hidden test cases to evaluate your code.

Assignments

- **You have total 3 late days**. You can use them all in one assignment, or one late day per assignment.
  - No additional late days will be given.

- **Deadline is strict. It will be always at 11:59pm**.

Assignment Schedule

- **Assignment 1:**
  - Release Date: April 2nd 2026.
  - Due Date: April 21st 2026.

- **Assignment 2:**
  - Release Date: April 22nd 2026.
  - Due Date: May 12th 2026.

- **Assignment 3:**
  - Release Date: May 13th 2026.
  - Due Date: June 2nd 2026.

Laboratory

- Each student is assigned to a lab.
- All the labs are on **Friday**.
- **Ranjitha** will conduct each lab.
- Labs carry total **20% of the grade**.
- There are a total of **10 labs**.
  - **Lab 0 is this Friday**.

Plagiarism

- **Please don’t plagiarize.**
- **Just don’t!**
- **We will apply the harshest UO policy available to us to deal with plagiarism!**
- **More details on the webpage.**

Missed Attendance Policy

- You can miss **at most 2 classes (including labs)** without getting penalized.

Operating Systems
CS 415

Lecture 1:

Overview

UNIVERSITY OF OREGON

Suyash Gupta

Assistant Professor

Distopia Labs and ONRG

Dept. of Computer Science

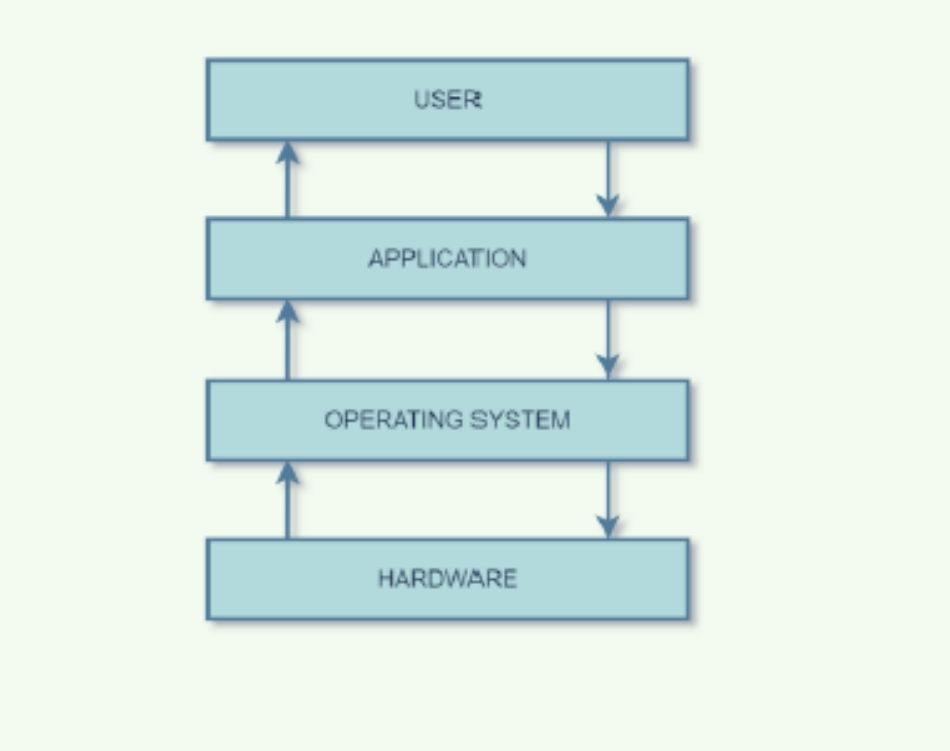
(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

What is an Operating System?

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.



Goals of an Operating System?

Goals of an Operating System?

- Execute user programs
- Make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

Do we need an OS to print?

```
main()
{
printf(“This is not a printout ...\n”);
}
```

Do we need an OS to print?

```
main()
{
printf(“This is not a printout ...\n”);
}
```

Without OS!

- Get printer manual.
  - Figure out how to send messages to it.

- Write the program.
  - Put the character string, “This is not a printout …” in a memory buffer.
  - Do the stuff printer requires to send buffer to it.

- Get hold of a computer
  - It has to be all yours!

Without OS!

- Translate your program into machine code
  - Has to be specific to the computer you are using.

- Figure out how to get program into memory.
  - Font panel switches.
  - Burn a ROM.
  - Punch cards.

- Somehow start program.

- Turn off computer when done.

With OS!

- Put program in file
  - Put character string “This is not a printout …” in a memory buffer.
  - Issue a **system call** to send buffer to printer.

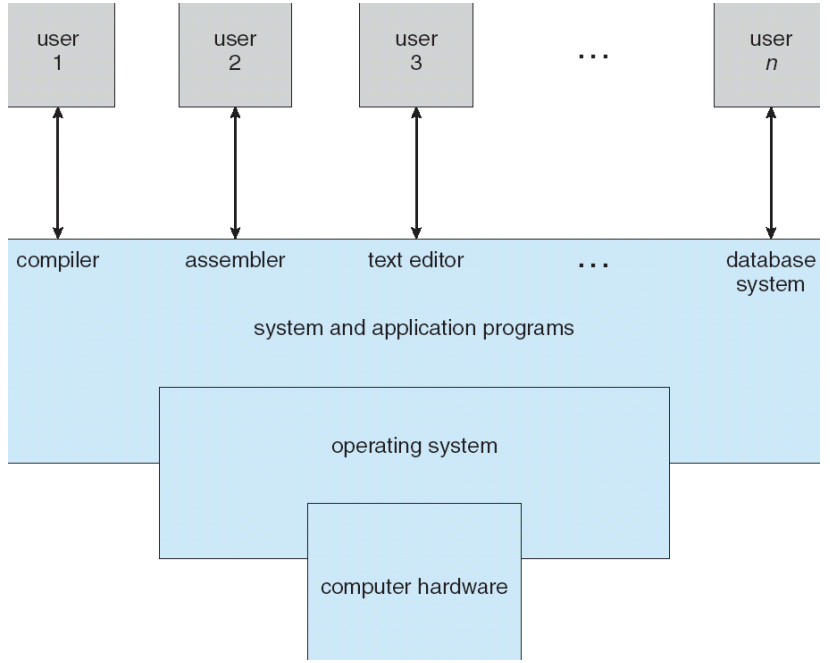
- Compile the program.

- Tell OS to run the executable file.
  - Make sure to turn on the printer first.

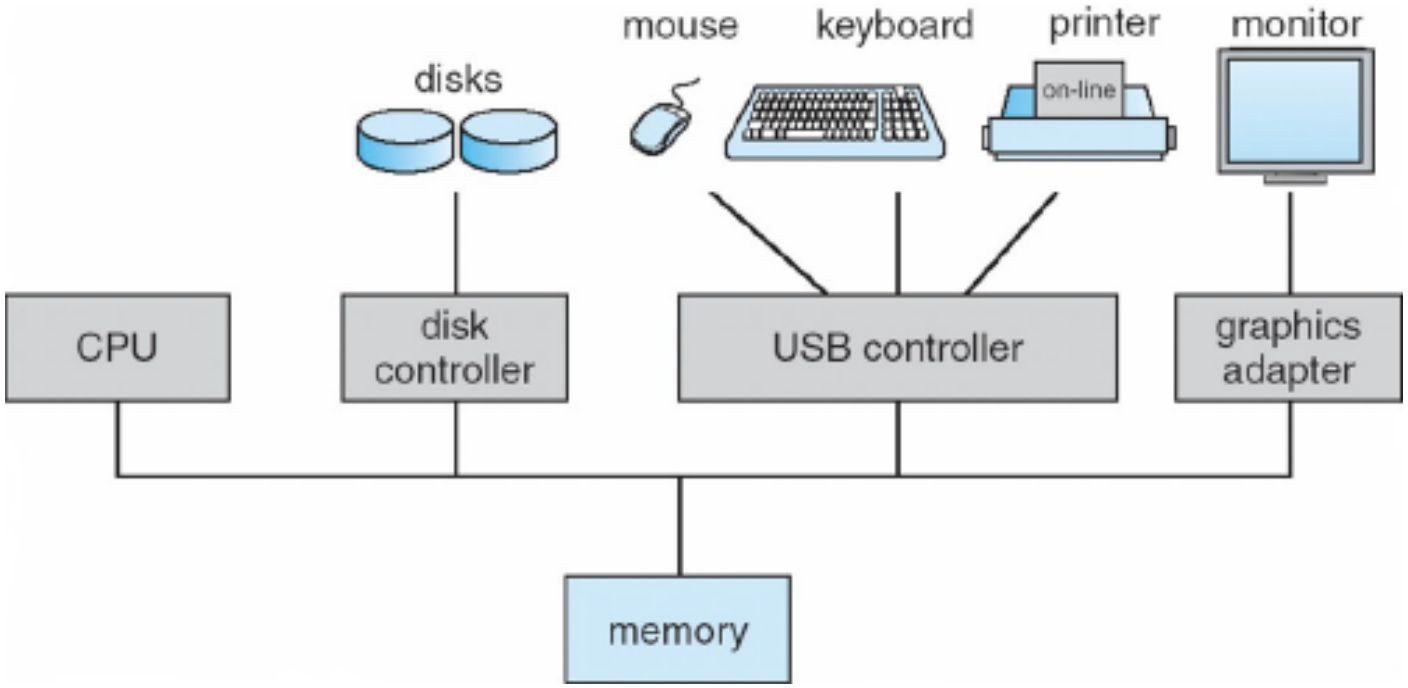
- That’s everything!

Computer System Structure

- Four key components:
- **Hardware:** provides basic computing resources.
  - CPU, memory, I/O devices
- **Operating system:** controls and coordinates use of hardware among various applications and users.
- **Application programs:** use system resources to solve computing problems.
  - word processors, compilers, web browsers, database systems, video games
- **Users:**
  - People, machines, other computers, robots.



Computer System Organization



- One or more CPUs, device controllers connect through common bus providing access to shared memory.

- Concurrent execution of CPUs and devices competing for memory cycles.

What Operating Systems Do?

- Depends on the point of view.
- Users perspective:
  - Convenience, ease of use, good performance.
  - Dedicated resources.
- System perspective:
  - Keep all users happy.
  - Share resources.
- Both perspectives are valid.
- Different OS requirements for different environments (hardware accelerators, embedded systems).

What Operating Systems Do?

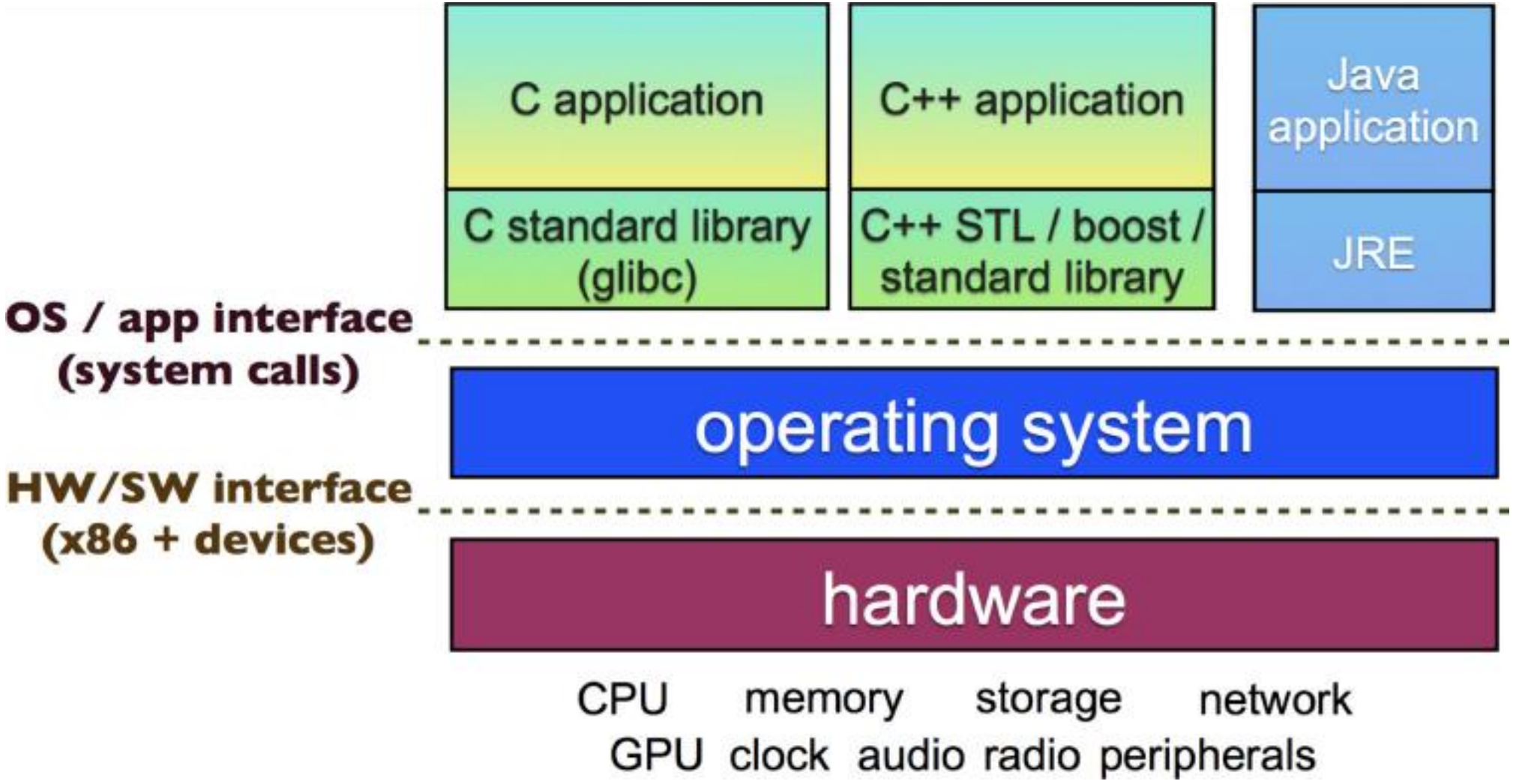
- OS is a resource allocator
  - Manages all resources.
  - Decides between conflicting requests → efficient and fair resource use.

- OS is a control program
  - Controls execution of programs to prevent errors and improper use.

- An OS is the one program running at all times (kernel), and everything else is a system program or application.

- An OS is everything a vendor ships you when you order an operating system

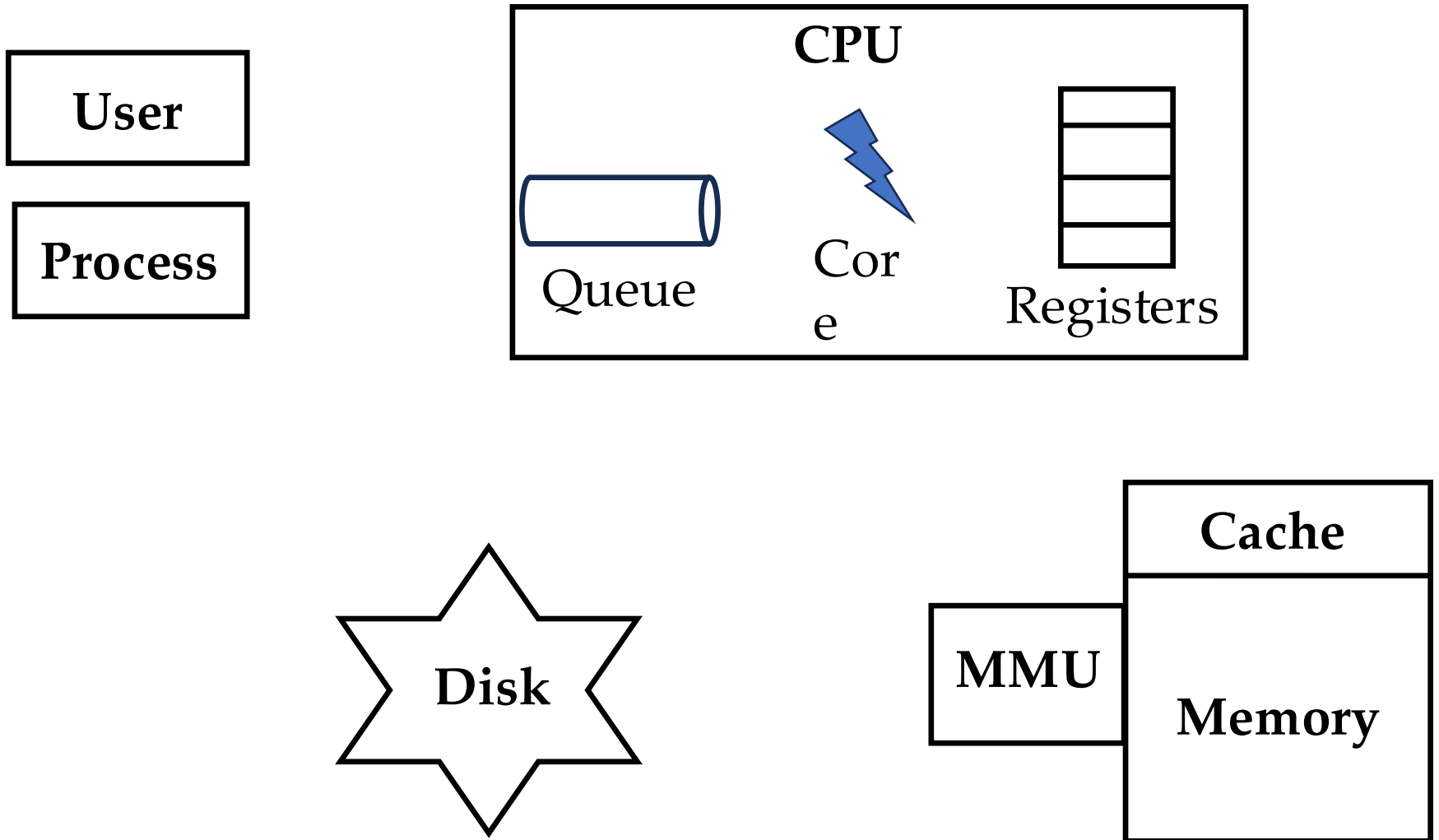
Where does the OS fit in?



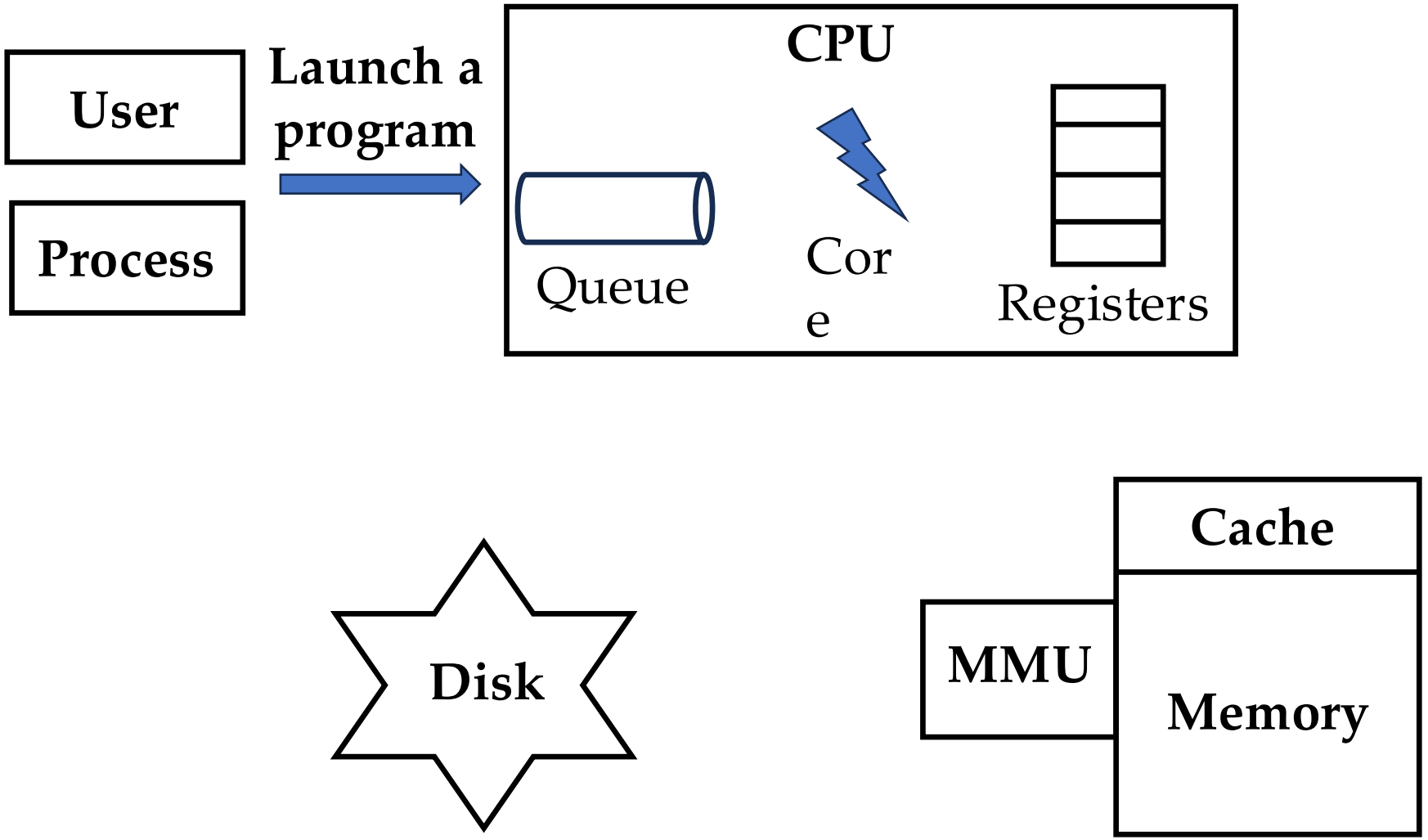
What does an OS do again?

- Performs resource and device management.
- Processes → Hides programs from one another.
- Traffic cop → Resource management → Who gets to run, when?
- Memory management → Protection from other programs' mistakes.
- Security → Protection from other programs' malice.
- System call interface → Like a function library, but communicates with the OS.
- Portability → Programs don't have to worry about details of their environment.
- Communication → Between processes; To devices & networks.

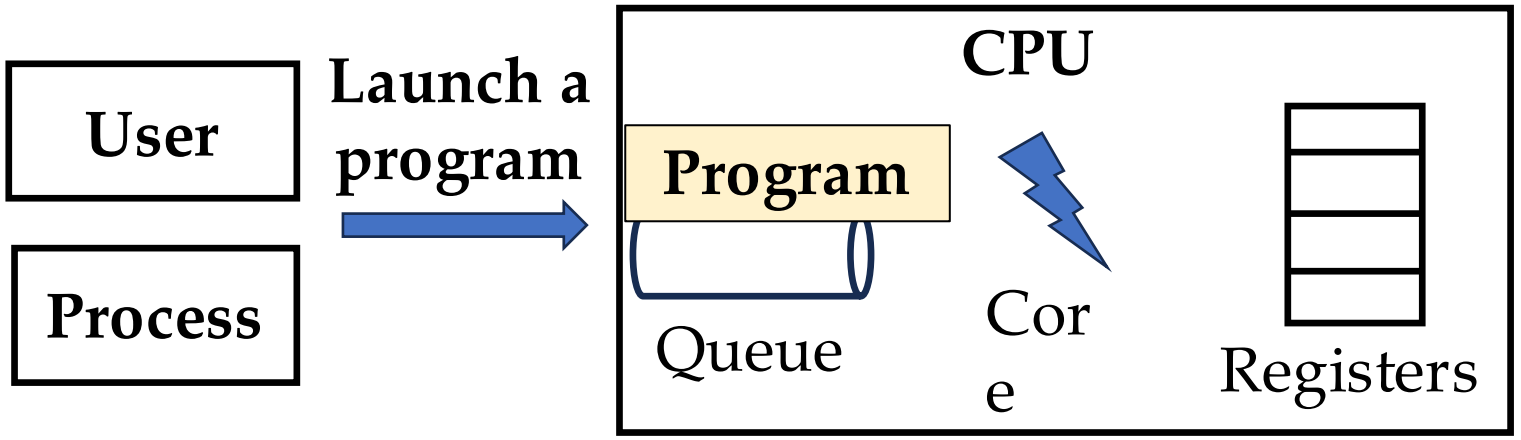
Modern Computer:
A Von Neumann Architecture



Modern Computer: A Von Neumann Architecture



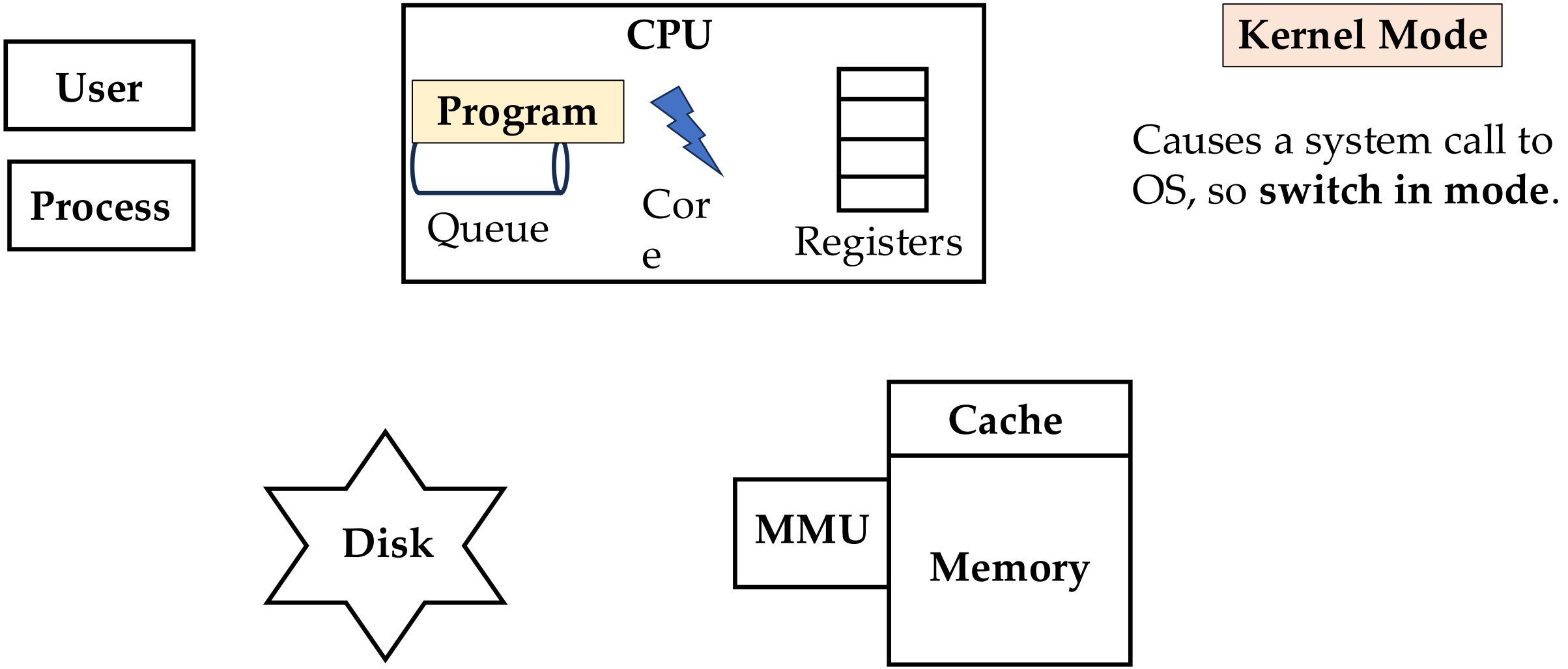
Modern Computer: A Von Neumann Architecture



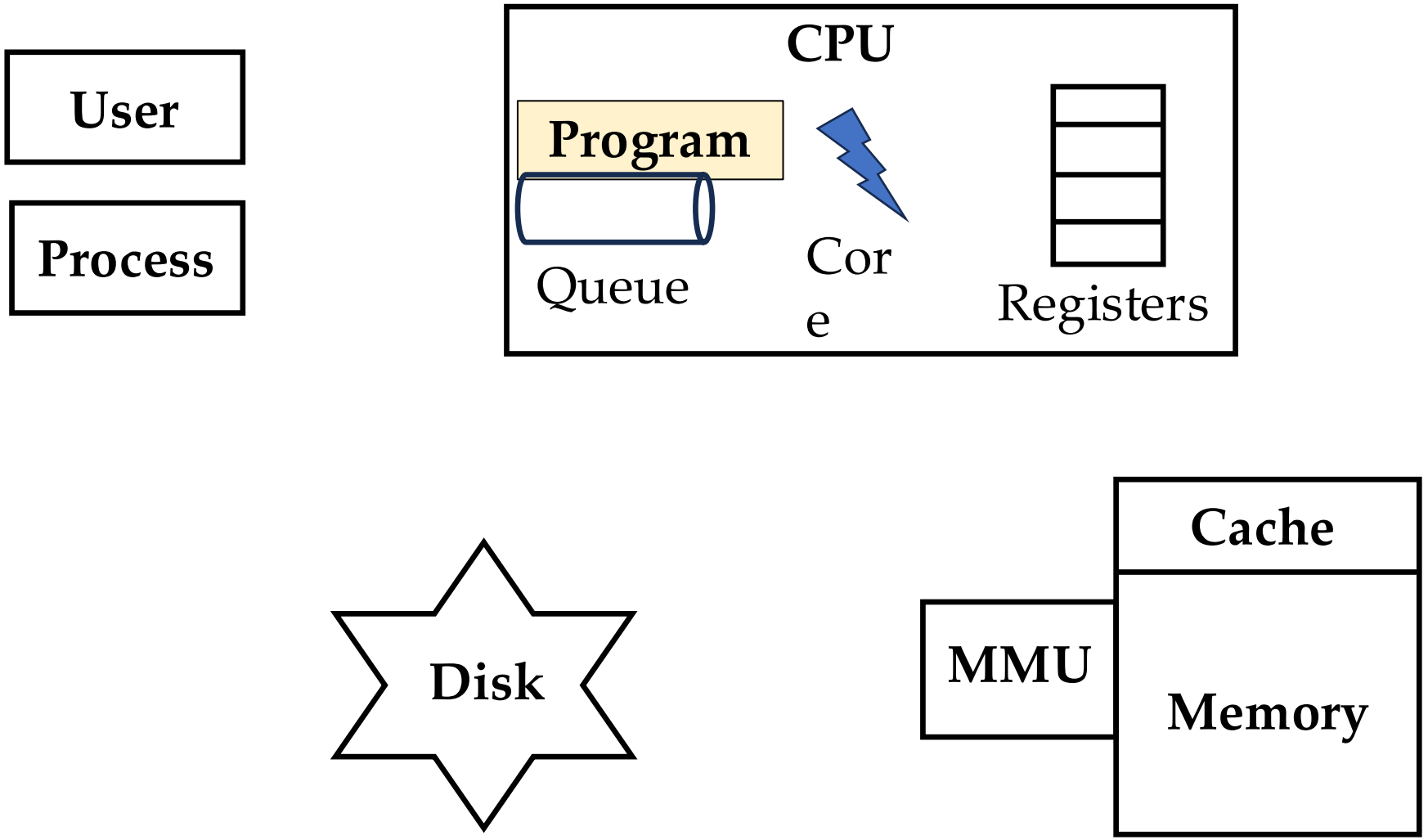
User Mode

User or any existing process can launch a program or sub-process.

Modern Computer: A Von Neumann Architecture



Modern Computer: A Von Neumann Architecture

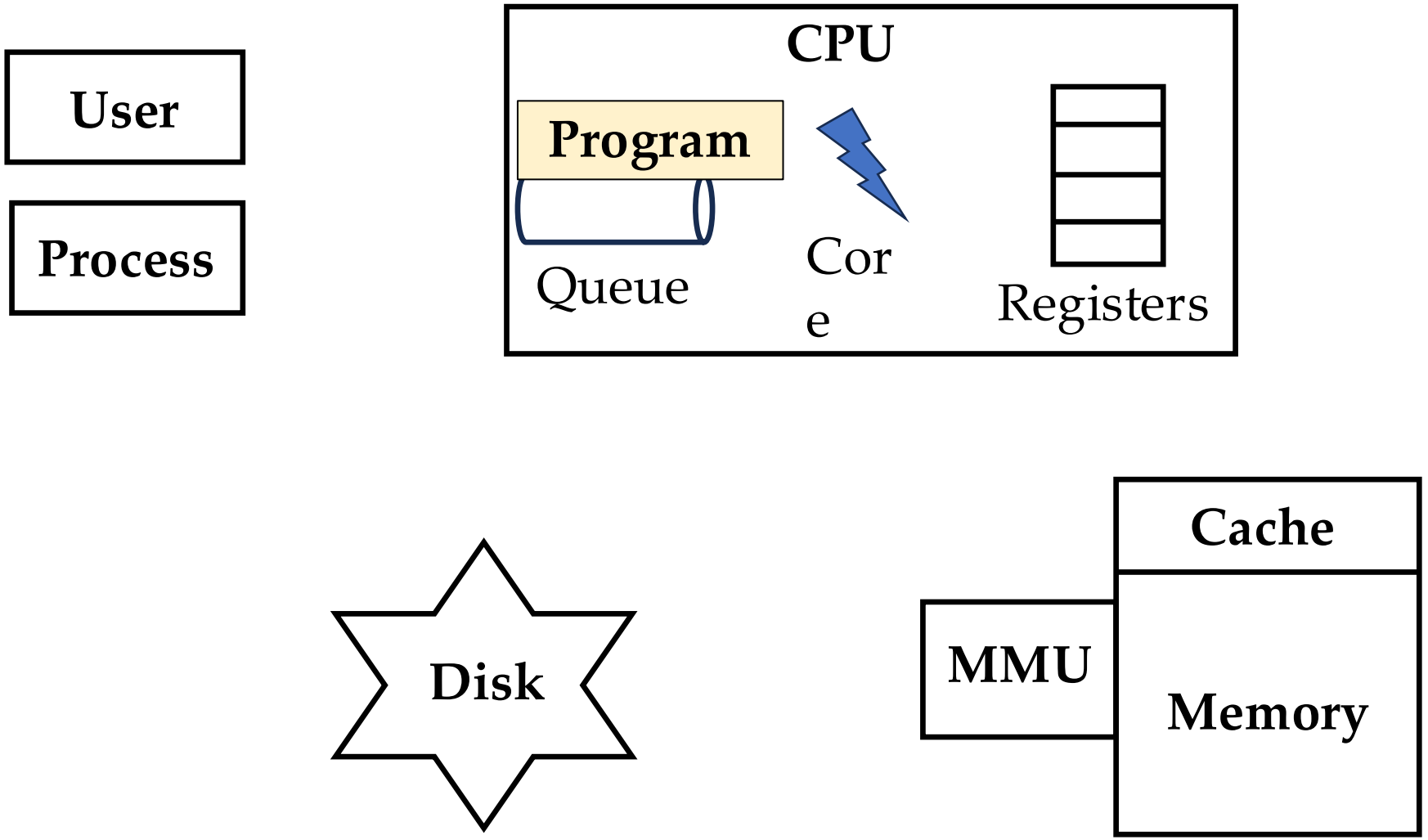


Kernel Mode

OS performs **process management**:

- Sets up region for code, heap.
- Process record
- Page structure
- Places process in scheduler's queue.

Modern Computer: A Von Neumann Architecture

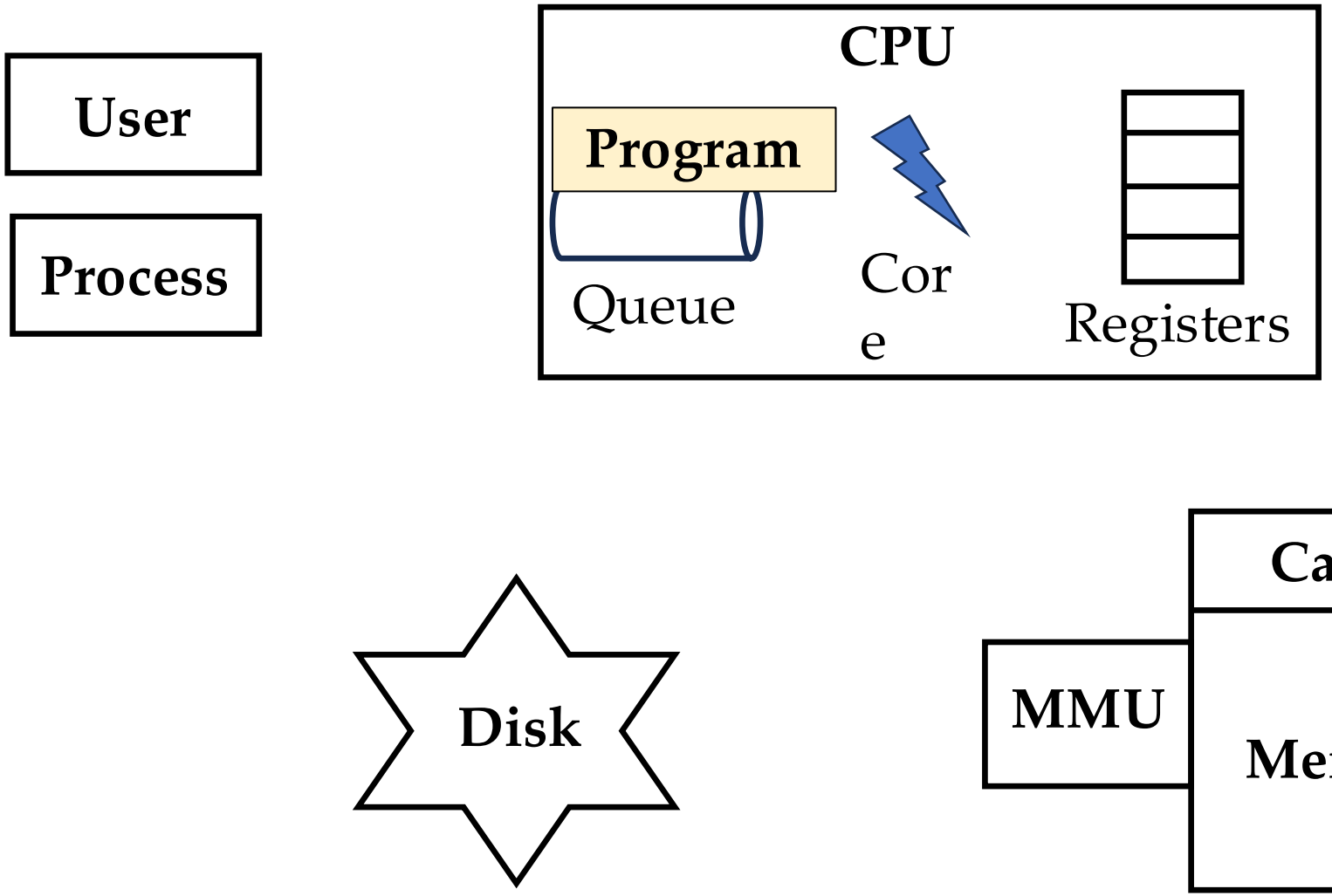


Kernel Mode

OS performs **process scheduling**:

- Selects the process.
- Loads its state, registers, page table and structure.
- Switch to **user mode**.

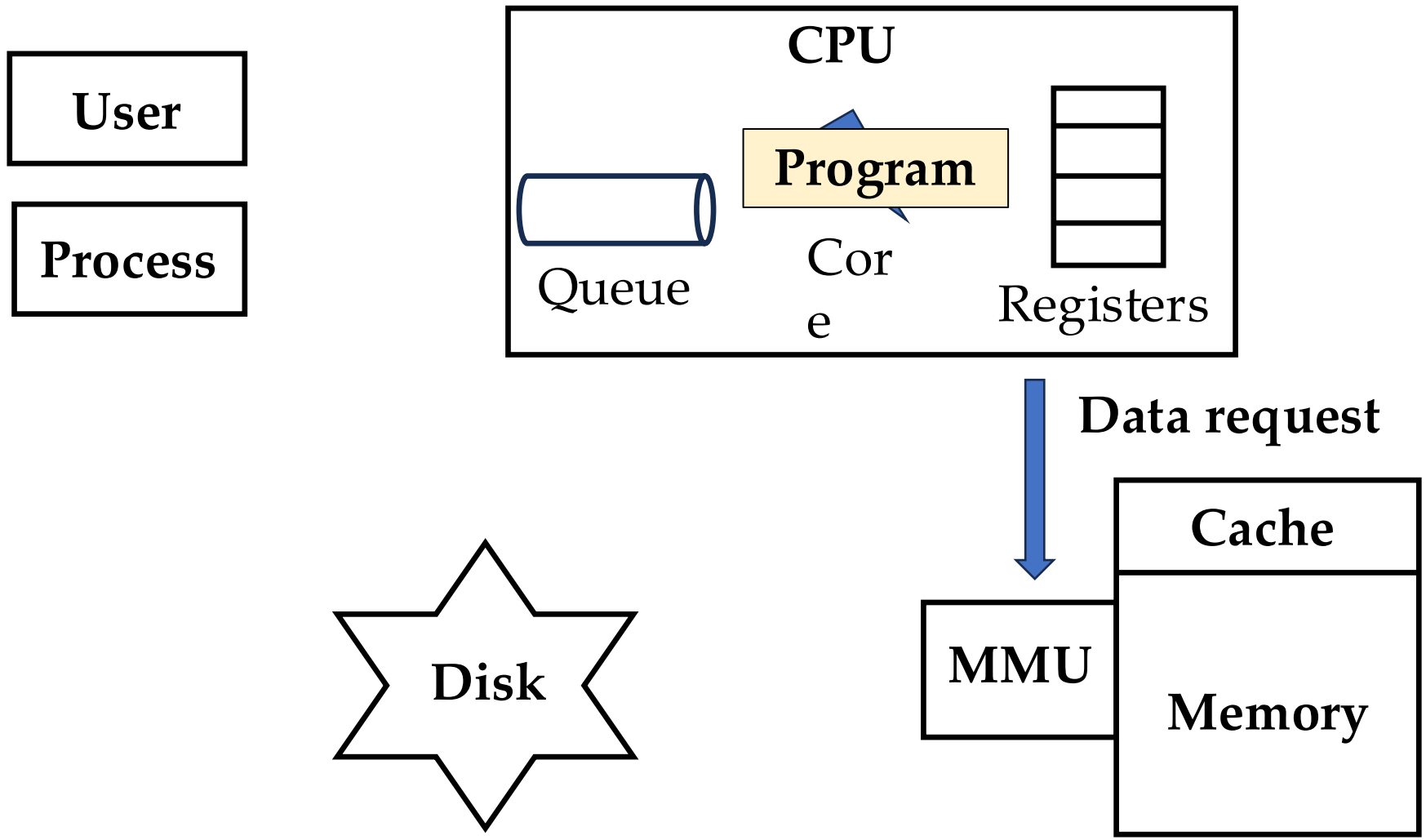
Modern Computer: A Von Neumann Architecture



User Mode

Now the process starts executing.

Modern Computer: A Von Neumann Architecture

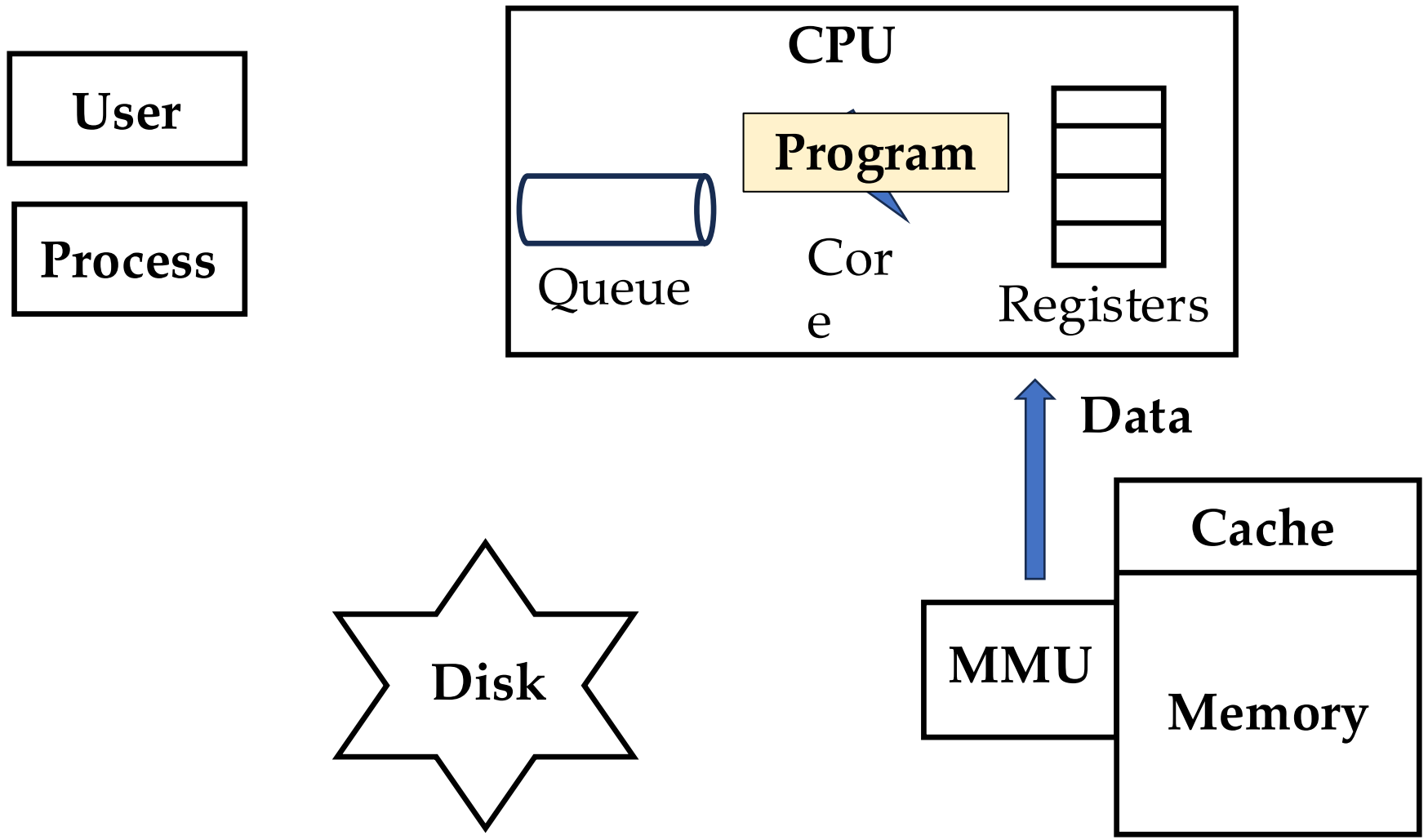


User Mode

Now the process starts executing.

If any data is needed, the CPU contacts MMU (Memory management unit.)

Modern Computer:
A Von Neumann Architecture



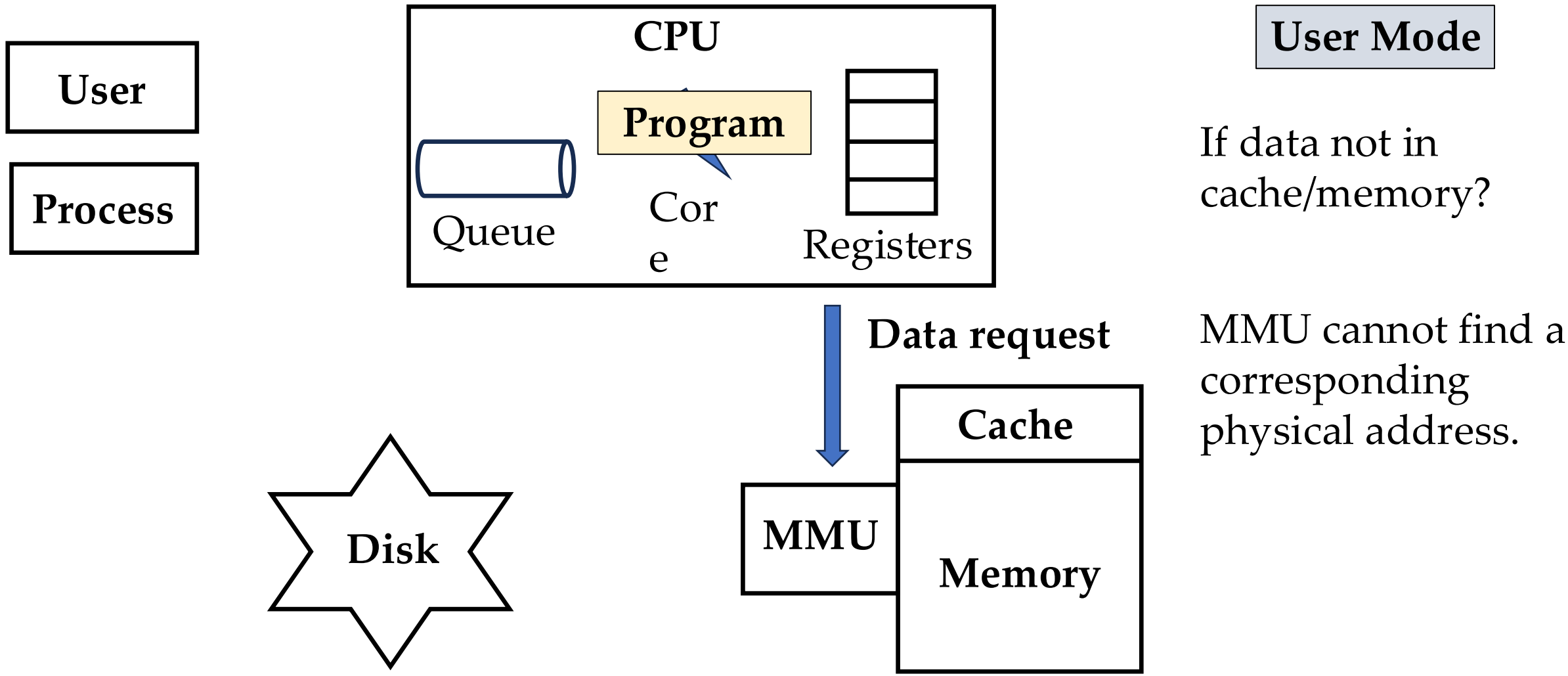
User Mode

MMU converts virtual address to physical address.

Cache or memory subsystem fetches the data.

Data is returned back.

Modern Computer:
A Von Neumann Architecture

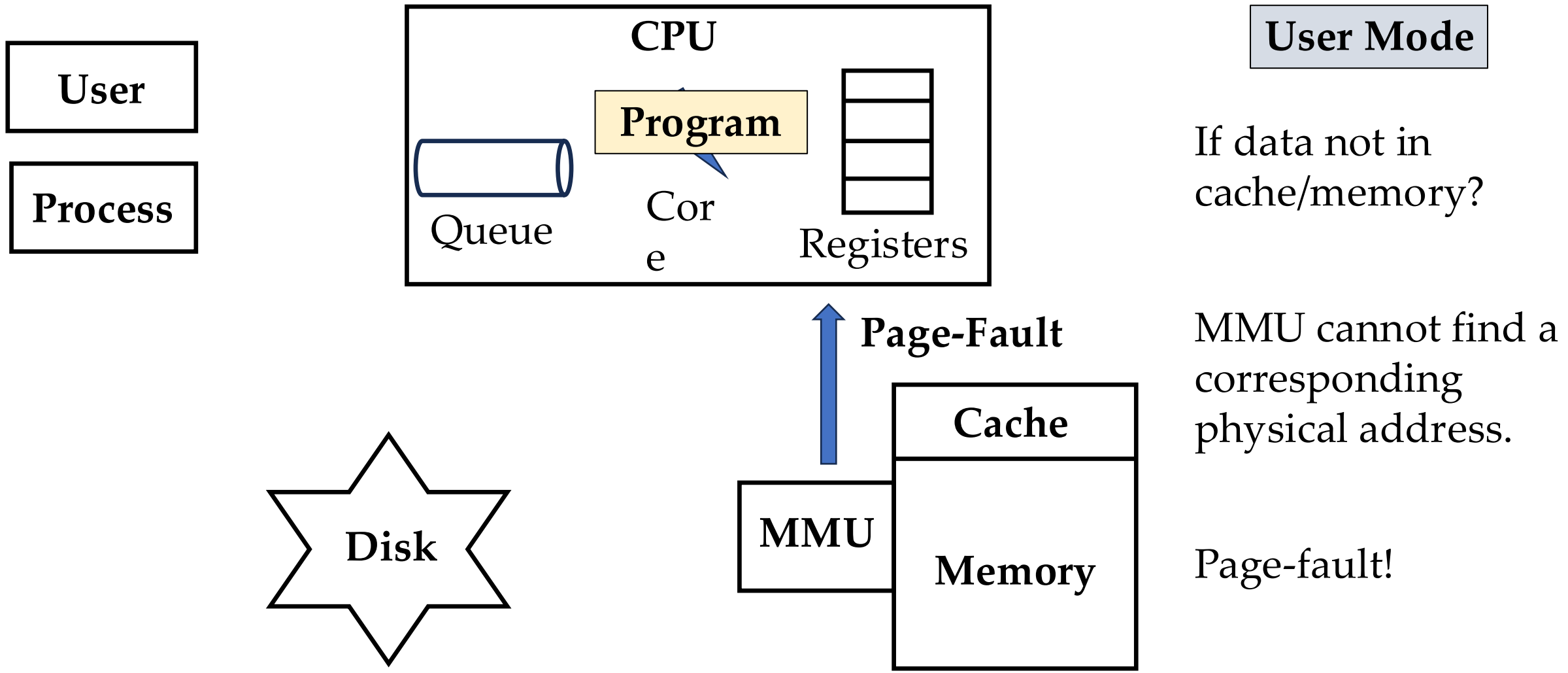


User Mode

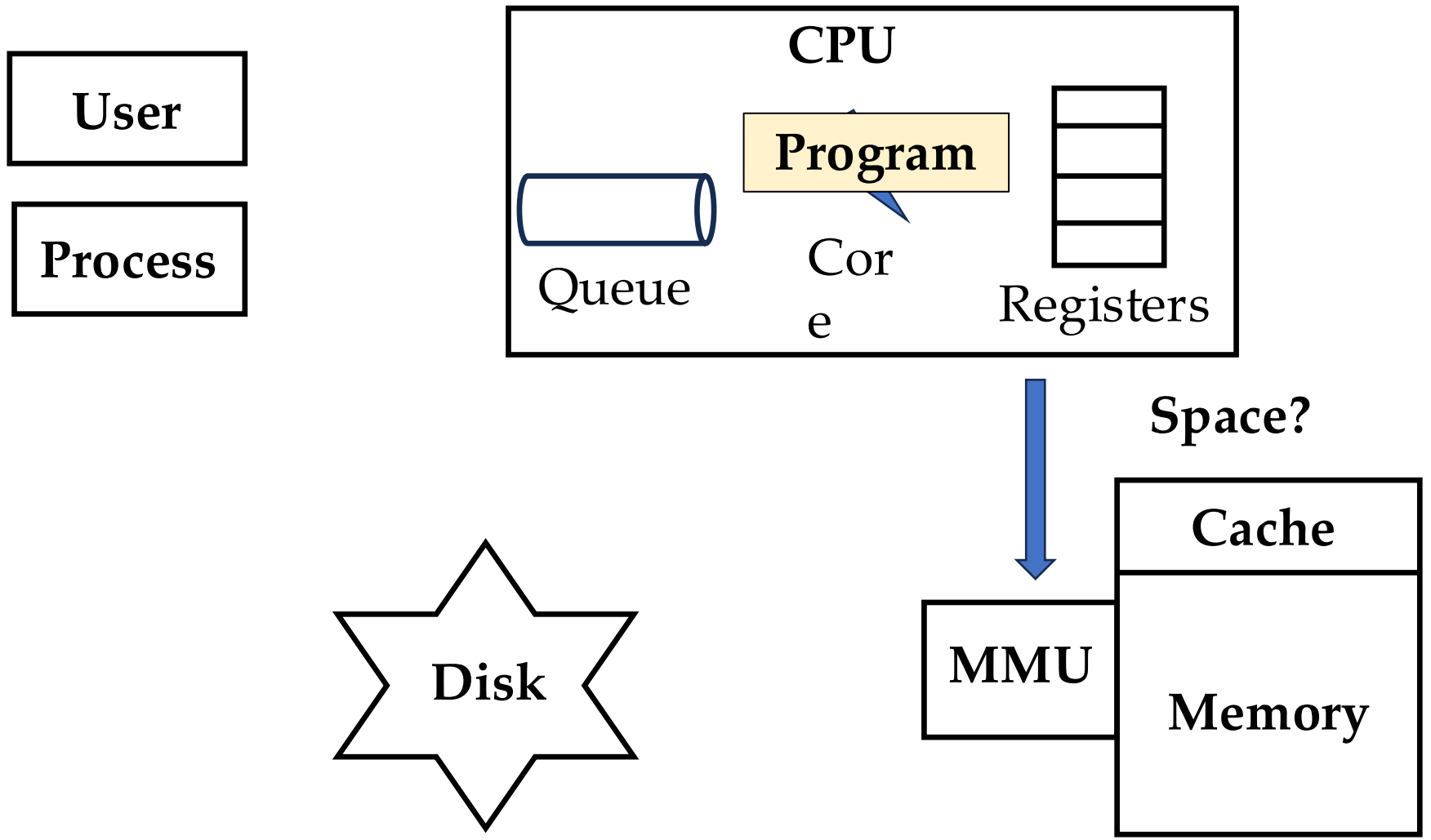
If data not in cache/memory?

MMU cannot find a corresponding physical address.

Modern Computer:
A Von Neumann Architecture



Modern Computer: A Von Neumann Architecture

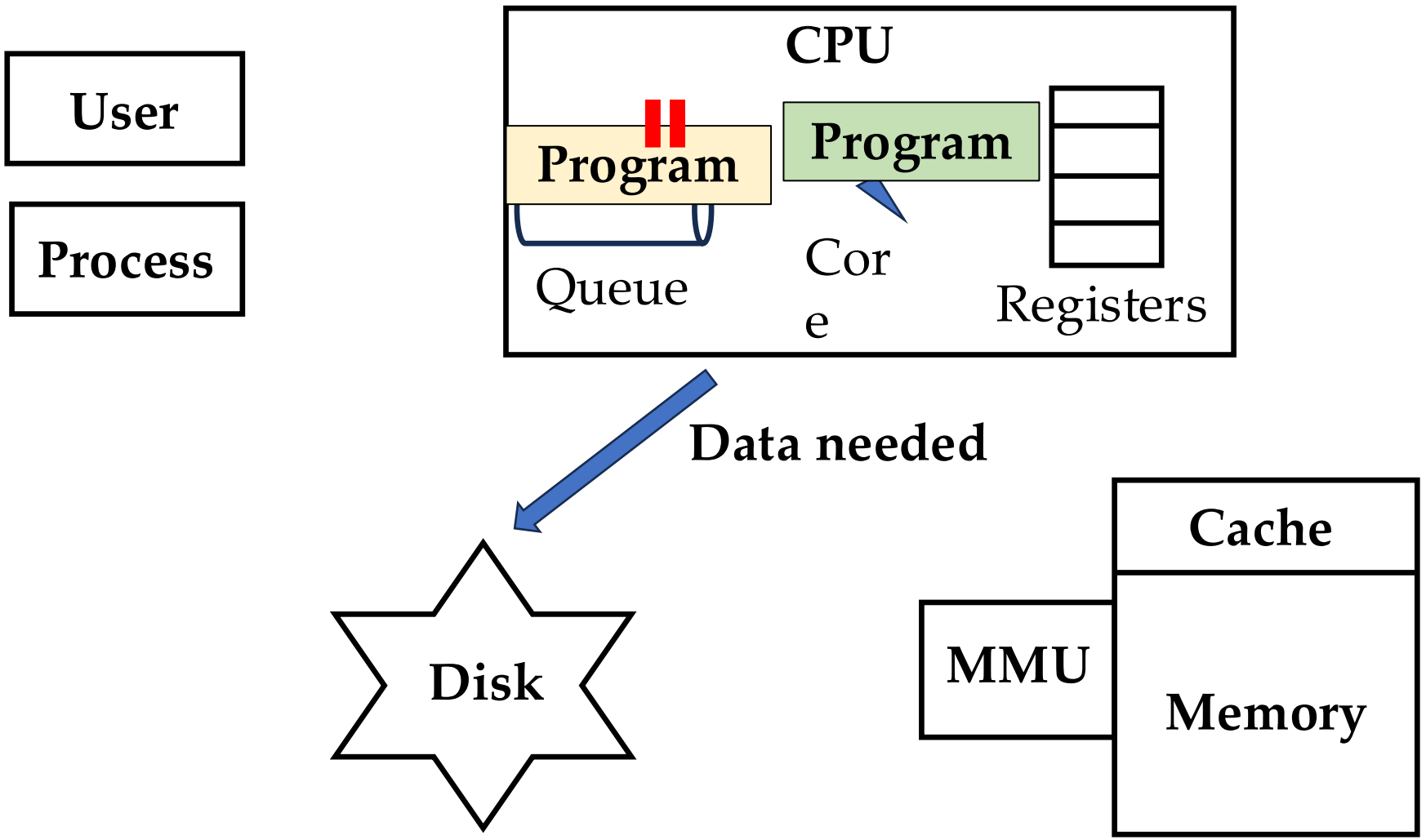


Kernel Mode

OS performs **page-fault handling**.

Makes space in memory if no space available!

Modern Computer:
A Von Neumann Architecture



Kernel Mode

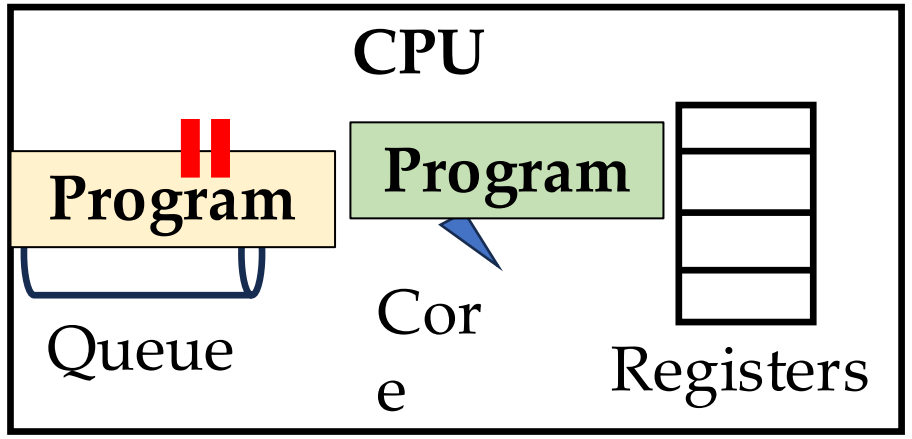
OS requests disk for data.

Pauses current program ands starts the next.

Modern Computer: A Von Neumann Architecture

User

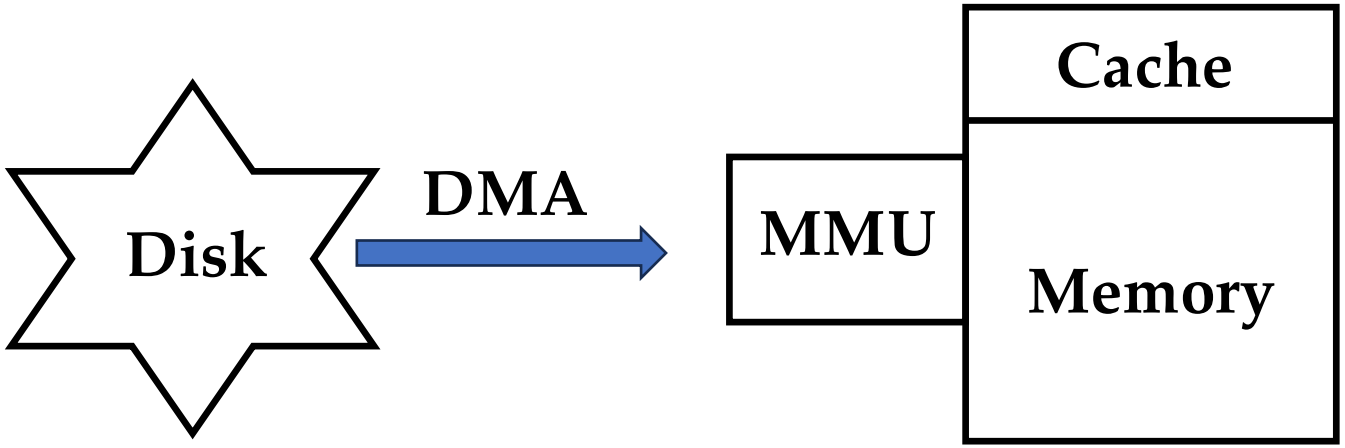
Process



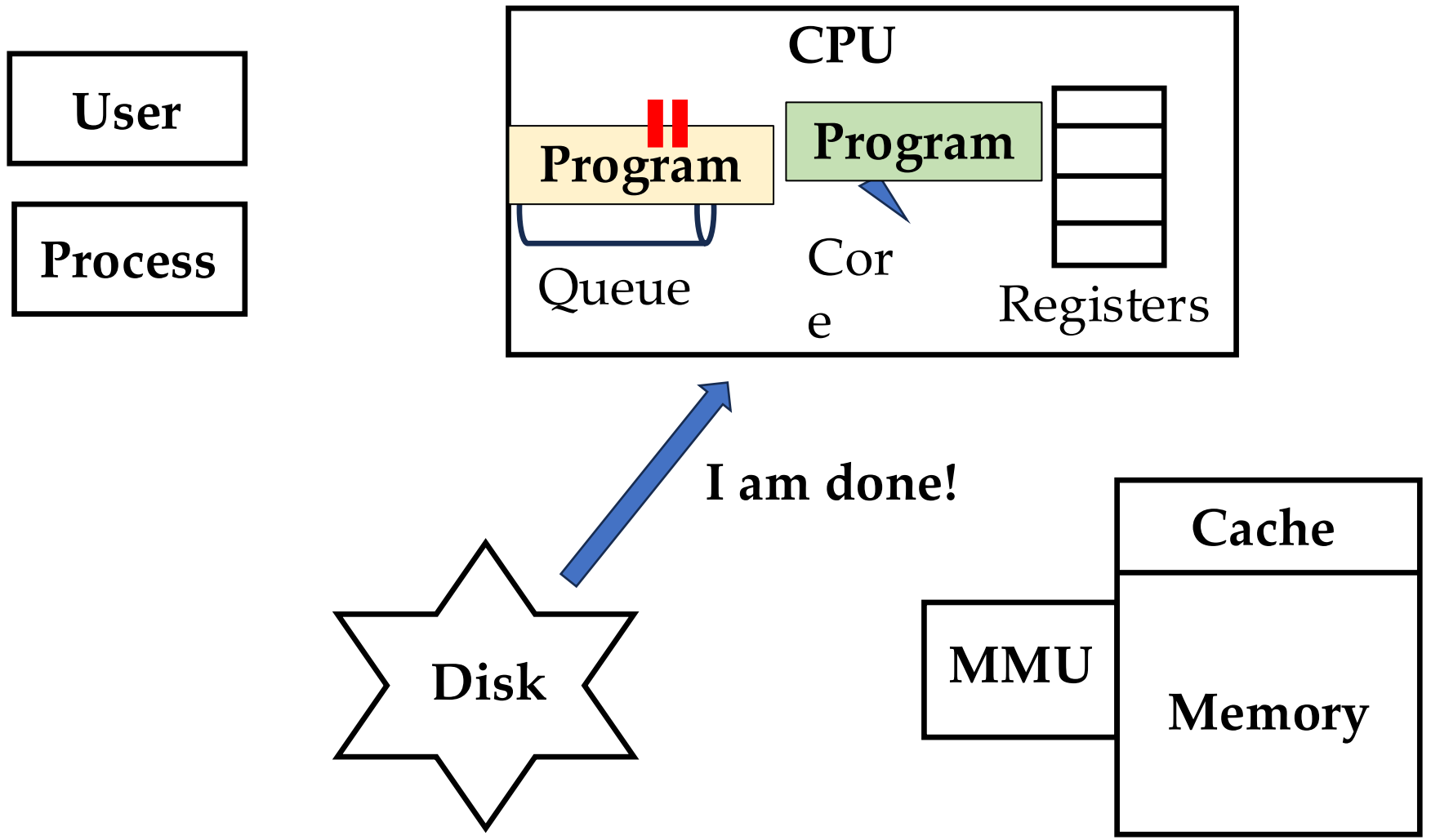
User Mode

We are back to user mode.

In parallel, Disk is performing Direct Memory Access (DMA) to send data to memory.



Modern Computer:
A Von Neumann Architecture

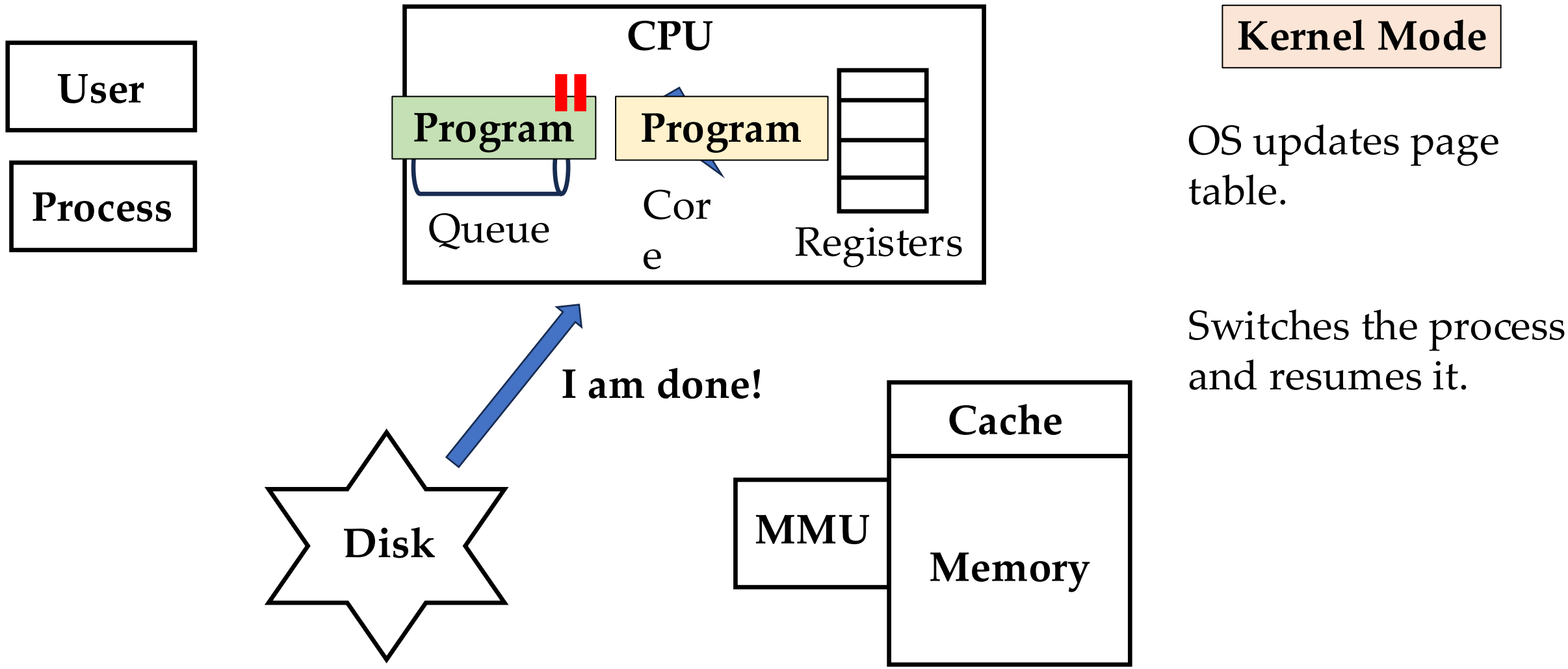


User Mode

When DMA finishes, Disk interrupts OS.

Switch modes.

Modern Computer: A Von Neumann Architecture



OS Modes: Multiprogramming vs. Timesharing

- **Multiprogramming**
  - **Goal:** Keep the CPU busy by having several programs in memory so when one waits for I/O, another can run.
  - **Trigger for switching:** mainly I/O wait or blocking (non-preemptive).
  - **Use:** batch systems, throughput-focused environments.

- **Time-sharing**
  - **Goal:** Give many users/programs quick, interactive response and the illusion that each has its own CPU.
  - **Trigger for switching:** a timer (preemption after a fixed time), plus I/O or blocking.
  - **Use:** interactive, multi-user systems (shells, terminals, desktops).

OS Modes: Multiprogramming vs. Timesharing

- **Similarities**
  - **Resource Utilization:** Both aim to maximize CPU utilization by keeping it busy.
  - **Concurrent Execution:** Multiple programs/processes are in memory at the same time.
  - **Scheduling:** Both rely on scheduling algorithms to decide which process gets CPU time.
  - **Context Switching:** The CPU switches between processes, so it looks like they are running simultaneously.

OS Modes: Multiprogramming vs. Timesharing

Aspect	Multiprogramming	Timesharing
Goal	Maximize CPU utilization/throughput.	Minimize response time, enable interactivity.
Trigger	When a job blocks on I/O.	On timer tick (time slice expiry) or I/O/block.
Preemption	Non-preemptive.	Preemptive (round-robin)
Users	Batch jobs	Many interactive users at terminals

History of Operating Systems

- 1950s: Simplify operators' job.
- 1960s: Structure, concepts, concurrency, theory.
- 1970s: Small and flexible (UNIX).
- 1980s: Individual user systems (PCs).
- 1990s: Internet, distributed systems.
- 2000s: Security, Linux/Windows/Mac OS.
- 2010s: Embedded, distributed, mobile, parallel, virtual.
- 2020s: IoT.

https://en.wikipedia.org/wiki/Timeline_of_operating_systems

History of Operating Systems

- Usage shared of operating systems

  https://en.wikipedia.org/wiki/Usage_share_of_operating_systems

- Most popular operating systems by market share (1978-2025)

  https://www.youtube.com/watch?v=cTKhqtll5cQ

History of Operating Systems (1950s)

- Primitive systems
  - Little memory
  - Programs stored on tape
- Single user
  - Batch processing
  - Computer executes one function at a time.
- No overlap of I/O and computation
- OS for IBM 704
  - Atlas Supervisor

  (University of Manchester)
  - University of Michigan Executive System (UMES)

History of Operating Systems (1960s)

- Multiprogramming
  - Timesharing
  - Multiple programs run concurrently
- Many operating systems concepts invented.
  - Virtual memory, hierarchical file systems, synchronization, security and many more.
- End up with slow, complex systems on limited hardware (Multics).
- Multics was an MIT project which partnered with Bell Labs and GE to try to support multiple users.

Edsger W. Dijkstra

Peter J. Denning

History of Operating Systems (1970s)

- Becoming more available
  - UNIX (Bell Labs)
  - written in a high-level language.
- Becoming more flexible
  - Extensible system.
  - Community forms beyond developers (precursor to open source movement).
- Performance focus
  - Optimization of algorithms from 1960s.

Brian W. Kernighan (L)
Dennis M. Ritchie (R)

History of Operating Systems (1980s)

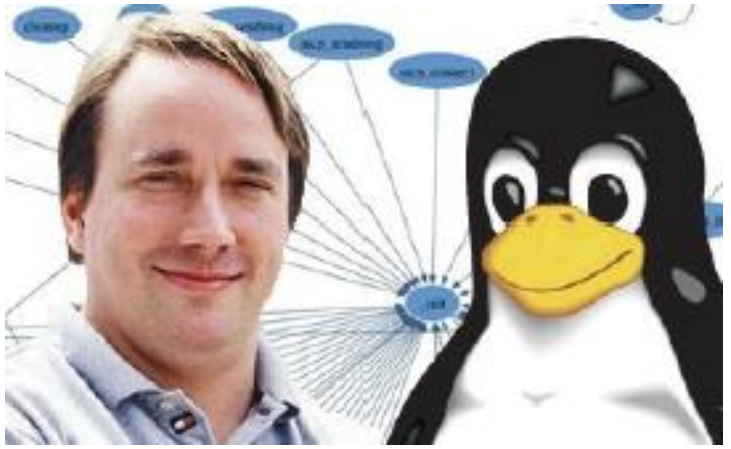
- Critical mass reached
  - A variety of well-known systems, concepts
  - UNIX fragments (BSD, SysV, …)
  - Open source begins to emerge
- PC Emerges
  - Simple, single user
  - No network
  - Simple OSes: DOS
- Graphical User Interfaces
  - Xerox Alto
  - Apple Macintosh

Bill Gates

Steve Jobs

History of Operating Systems (1990s)

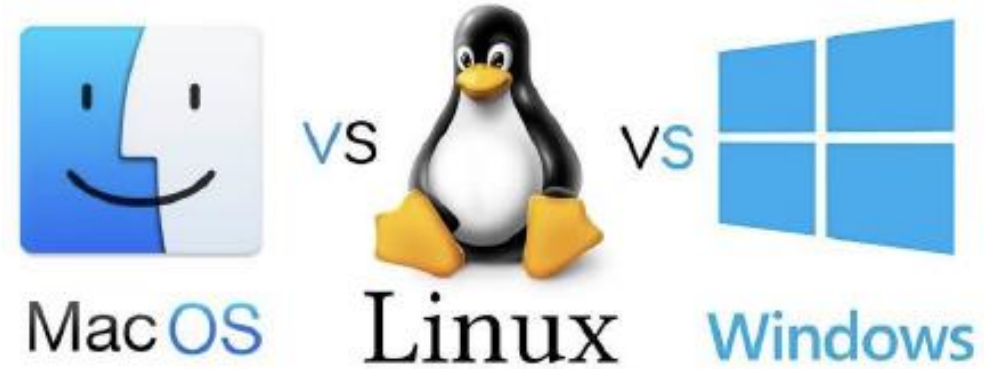
- Connect to Internet
  - "Real OSes" for PCs: NT/2000+, Linux, Mac OS X
  - Web emergence
- Server systems galore
  - Mainframes even re-emerge
- Complex systems and requirements
  - Multiprocessors, memory hierarchy, interconnects.
  - Parallel
  - Real-time
  - Distributed



Linux B. Torvalds

History of Operating Systems (2000s)

- More dimensions (problems) and range of scales
  - Distributed systems
  - Multicore
  - Security
  - Ubiquitous
  - Virtual Machines
  - Embedded
  - Mobile



- More dominance of Oses
  - Linux, MacOS, Windows