University of Pittsburgh

Introduction to Operating Systems
CS 1550

5 YEARS Pitt SCI

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Some slides are from **Silberschatz, Galvin and Gagne ©2013)**

Course Goal and Objectives

Continue to demystify a good portion of the **magic** about how computers work so that you can write more **efficient** programs

The specific objectives are:

1. Modify and compile the Linux kernel to **add system calls**
2. Write **multi-process/multi-thread** programs free from race conditions and deadlocks
3. Simulate page-replacement algorithms for **virtual memory** management
4. Implement a user-land **file system**

Contact Info

- **Course website:** http://www.cs.pitt.edu/~skhattab/cs1550/
- **Instructor:** Sherif Khattab ksm73@pitt.edu

Student Support Hours: https://khattab.youcanbook.me

MW: 11:00-12:00

Th: 9:00-10:00 and 11:00-12:00

F: by appointment

  - 6307 Sennott Square, Virtual Office: https://pitt.zoom.us/my/khattab

- **Teaching Team:**

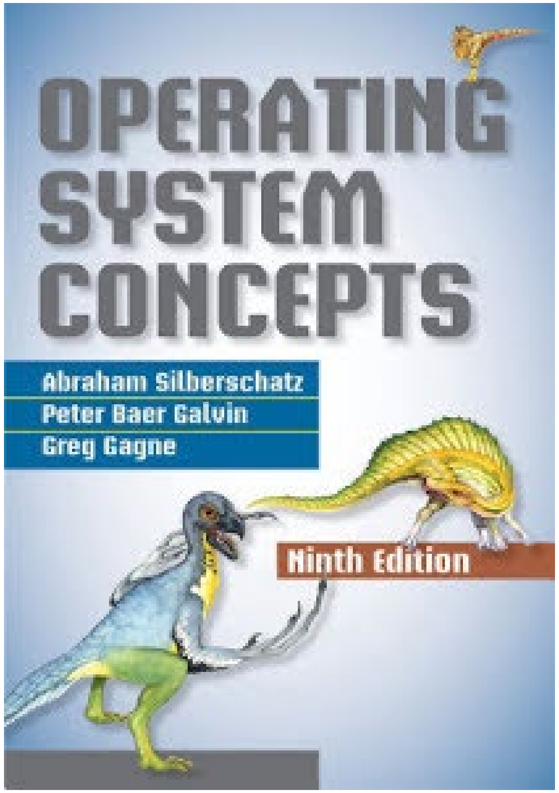
TBD

- Recitations start next week!
- **Communication**

Piazza (**Please expect a response within 72 hours)**

Email not recommended!

Textbook



Operating System Concepts (9th Edition)

Silberschatz, Galvin, and Gagne

Grading

- 40% on four programming projects
  - System programming using C
  - posted on Canvas, distributed using Github, and submitted on **Gradescope** from Github
  - partially auto-graded with unlimited submission attempts
- 30% on exams: 18% on higher grade and 12% on lower
- 10% on homework assignments on Canvas
  - auto-graded with three attempts
- 10% on lab exercises and 4 quizzes on the projects; auto-graded
  - Labs use MIT's Xv6 operating system
- 10% on in-class Top Hat questions
  - participation only

Canvas Walkthrough

- Lecture PPTs posted on Tophat after class
  - Draft slides in PDF available on Github usually before class
- Lecture and recitation recordings
  - under Panopto Video
- Pre- and Post-course test
  - Pre-test must be taken to unlock the course modules
- Piazza for discussion and communication
- Gradescope and autograding policies
- Academic Integrity
- NameCoach

Expectations

- Your continuous feedback is important!
  - Anonymous Qualtrics survey (link on Syllabus page)
  - Midterm and Final OMET
- Your engagement is valued and expected with
  - classmates
  - teaching team
  - course material

Lecture structure (mostly)

Time	Description
~5 min before and after class	Informal chat
~25 min	Announcements, review of muddiest points on previous lecture, and QA on assignments/labs/homework problems
~45 min	Lecturing with Tophat questions and/or activities
~5 minutes	QA and muddiest points/reflections

Why is this class (notoriously) hard?

- Lots of concepts
  - Attend lectures and recitations (if you absolutely cannot attend in-person, watch the video recordings)
  - Study often!
  - Put effort into the weekly homework assignments
- Projects are relatively hard
  - Refresh your C programming and GDB debugging skills (CS 0449!)
  - Start early and show up to student support hours!

Announcements

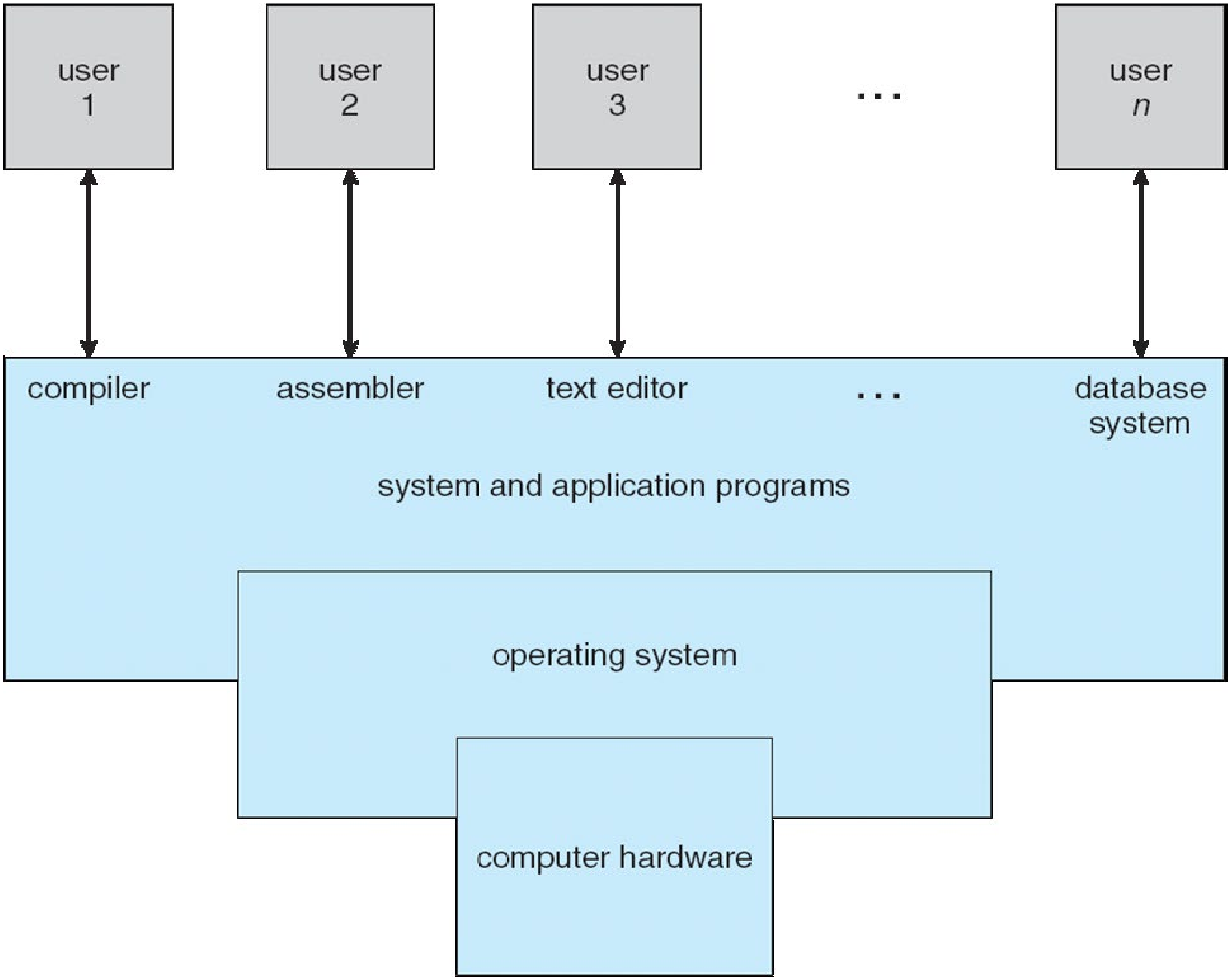
- Lab 0 due next Friday (soft deadline; not graded)
- Homework 1 will be posted this Friday
- Recitations start next week
- VS Code setup tutorial on Piazza (also linked from Canvas)
- Draft Slides linked from Canvas

Agenda

- Main tasks of an operating system
- System Calls
  - What an interrupt is
  - What happens when an interrupt occurs
  - What a system call is
  - How system calls implemented
  - Effect of OS structure on system calls

What is an Operating System?

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



What does an OS do?

- Manages (controls and arbitrates) resources
  - Processors, Memory, Input/output devices, Communication devices, Storage, Software applications
  - Conflicting goals:
    - e.g., performance *vs.* utilization
    - Separation of policy and mechansim
- Provides abstractions to application programs
  - Ease of use
  - Virtualization
- Protects resources

Interrupts

Hardware or software:

- Hardware interrupt by one of the devices
- Software interrupt (**exception** or **trap):**
  - Software error (e.g., division by zero)
  - Other process problems include processes trying to modify each other’s or the operating system’s memory (e.g., segmentation fault)
  - Request for operating system service (i.e., system call)

Interrupt Descriptor Table

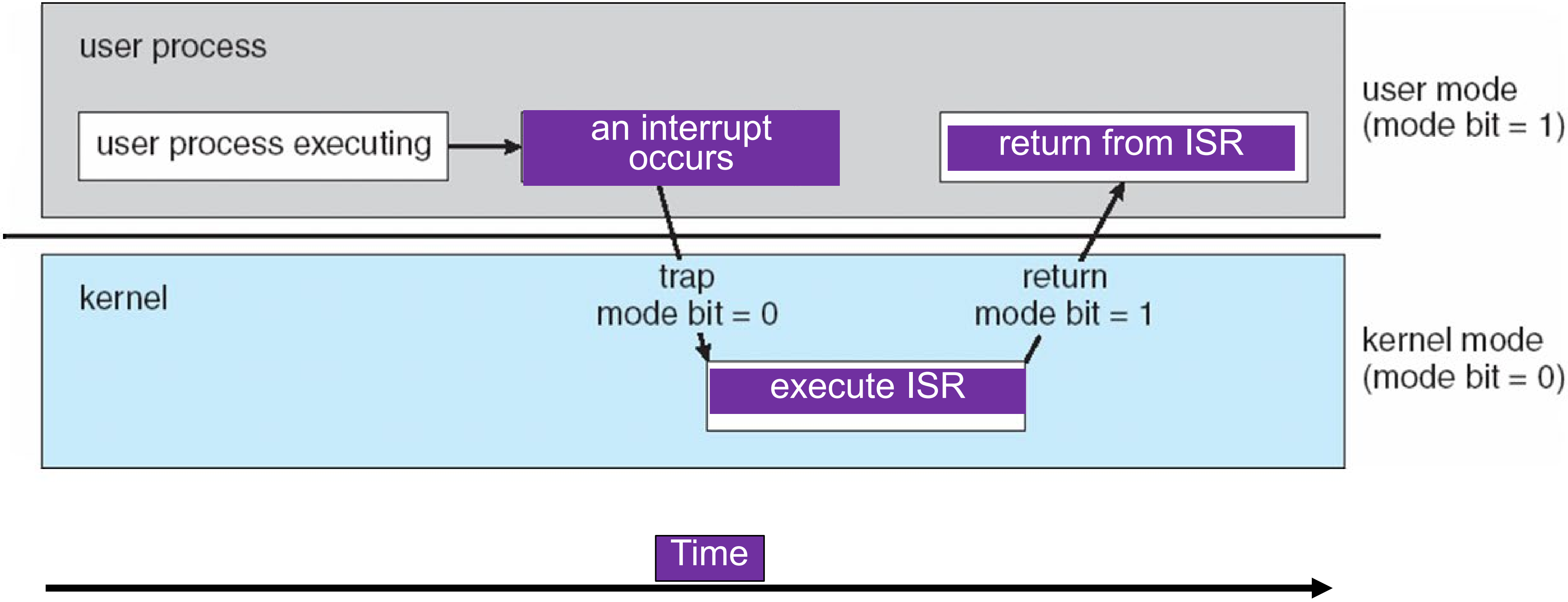
- Interrupt transfers control to the interrupt service routine (ISR)
- ISRs are segments of code that determine what action should be taken for each type of interrupt
  - part of the OS kernel
- An **interrupt vector** contains the address of the ISR for one interrupt
- An **interrupt vector table** is an array of interrupt vectors
  - **also known as interrupt descriptor table (IDT)**

Dual-mode Operation

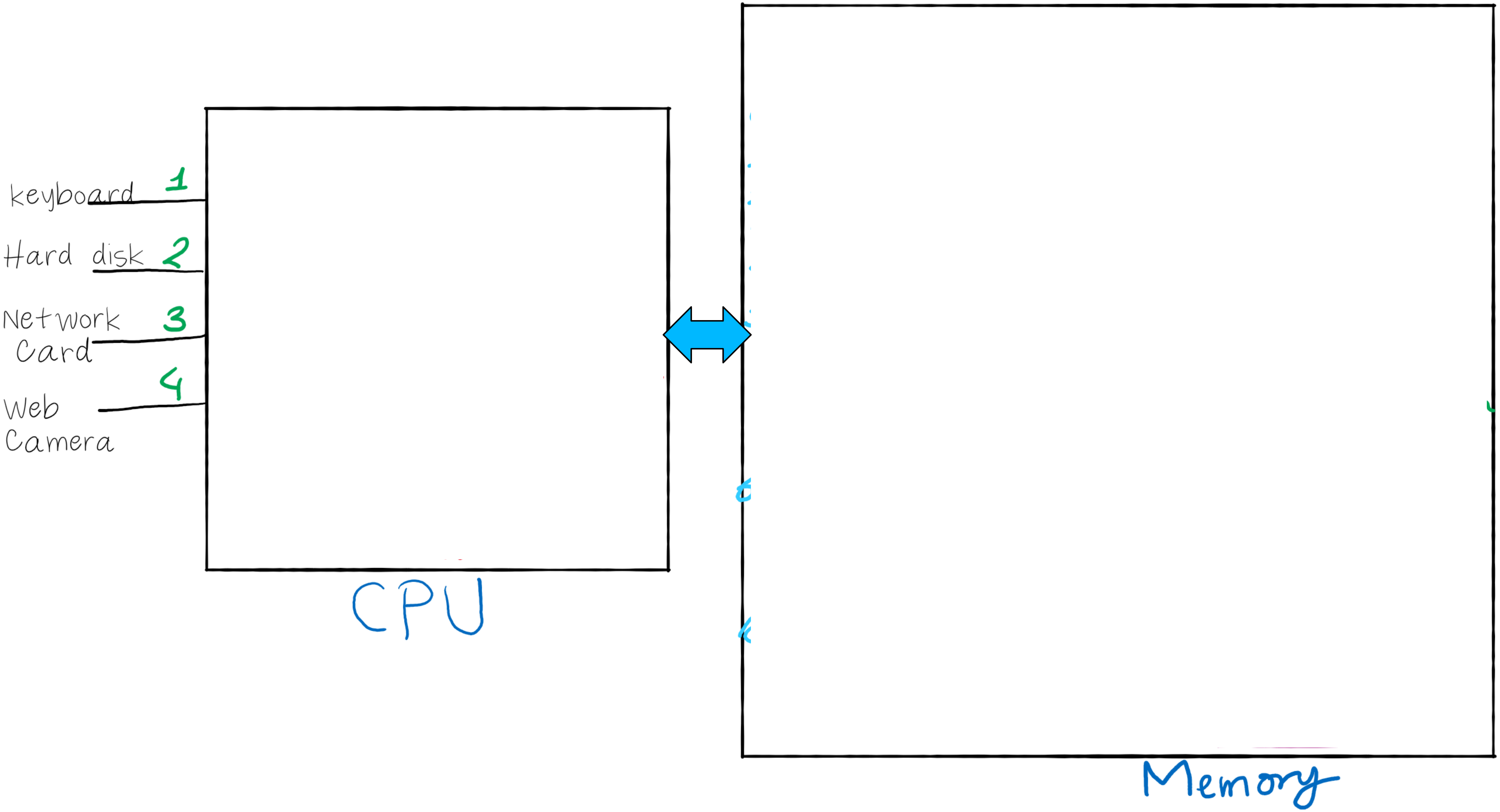
- **Dual-mode** operation allows OS to protect itself and other system components
  - **At least two modes: user mode** and **kernel mode**
  - **Mode bit(s)** provided by hardware (inside CPU registers)
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - Some memory addresses designated as **privileged**, only accessible in kernel mode
      - Therefore, we get segmentation fault on null (i.e., 0) pointer dereference
    - Interrupts change mode to kernel
      - return from interrupt resets mode back to user
- Increasingly CPUs support multi-mode operations
  - **virtual machine manager** (**VMM**) mode for guest **VMs**

What happens when an interrupt occurs?

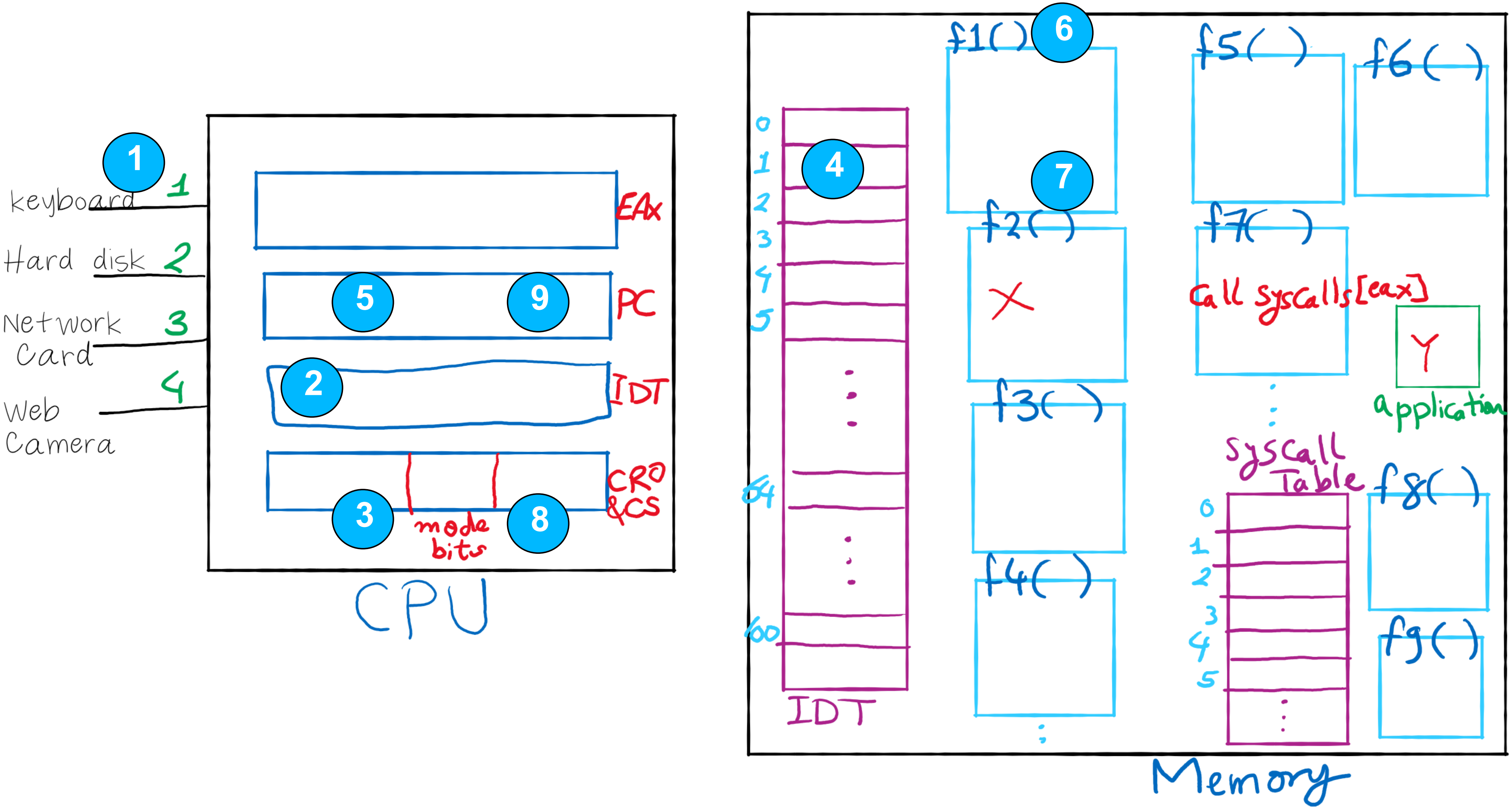
The CPU transitions from User Mode to Kernel Mode



How does an OS (roughly) work?



What happens on a hardware interrupt?


1
keyboard
1
Hard disk
2
Network Card
3
Web Camera
4
EAx
5
9
PC
2
IDT
3
8
CR0 &CS
mode bits
CPU
0
1
2
3
4
5
64
60
4
IDT
f1()
6
7
f2( )
X
f3( )
f4( )
f5( )
f6( )
f7( )
call syscalls[eax]
Y
application
syscall Table
0
1
2
3
4
5
f8( )
f9( )
Memory