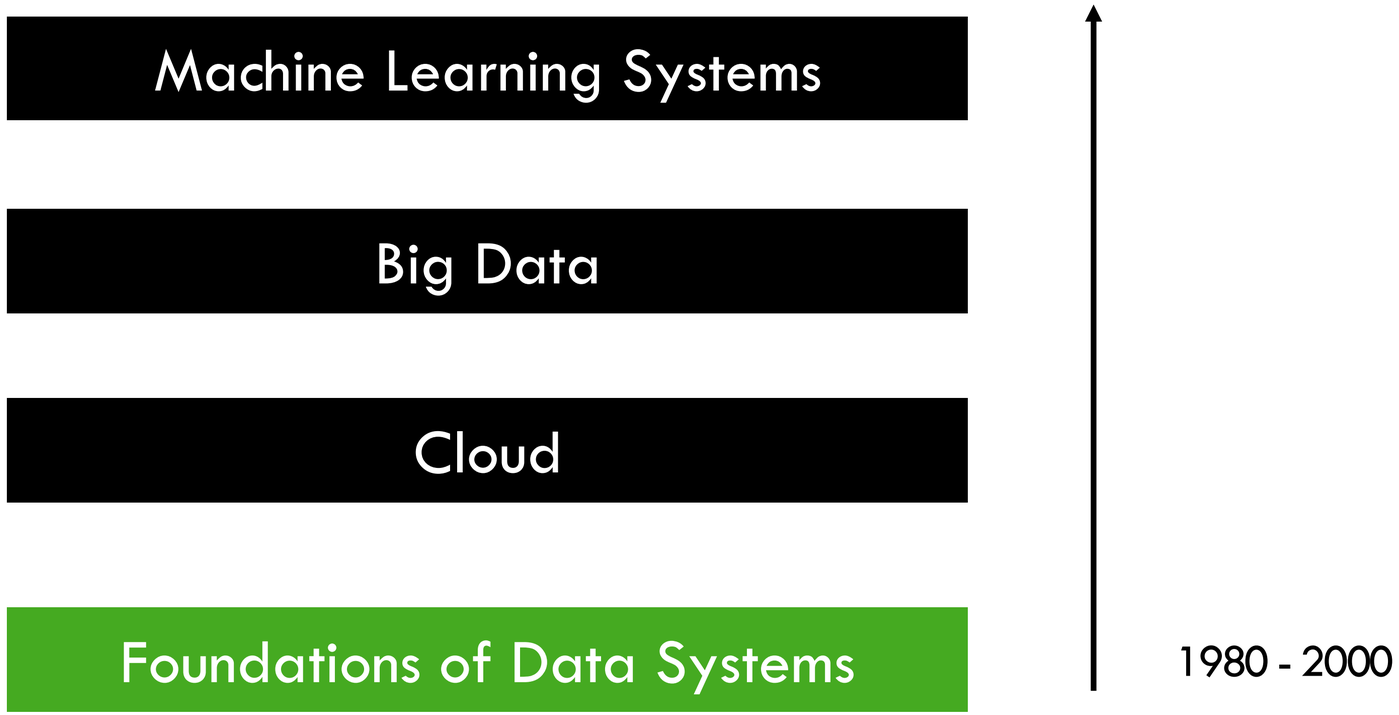
https://hao-ai-lab.github.io/dsc204a-f25/

DSC 204A: Scalable Data Systems
Fall 2025

Staff
Instructor: Hao Zhang
TAs: Mingjia Huo, Yuxuan Zhang

@haozhangml @haoailab
haozhang@ucsd.edu

Where We Are


Machine Learning Systems
Big Data
Cloud
Foundations of Data Systems
1980 - 2000

Logistics

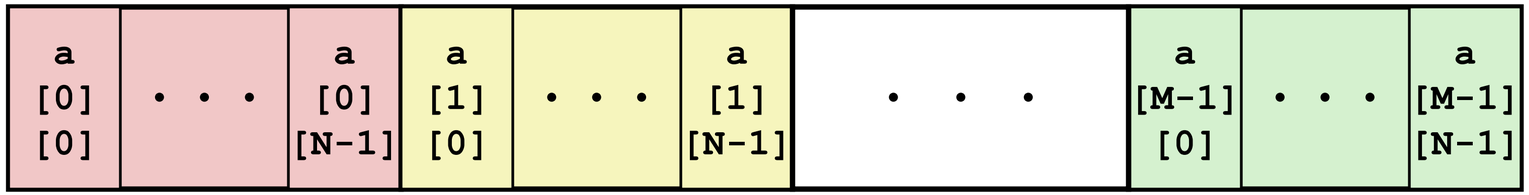
- Beginning of Quarter Survey: 77% completion
- Finish the 3% and you all get 1 point!

Qualitative Estimates of Locality

Assuming row-major array

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

Answer: yes



Question: Does this function have good locality with respect to array `a`?

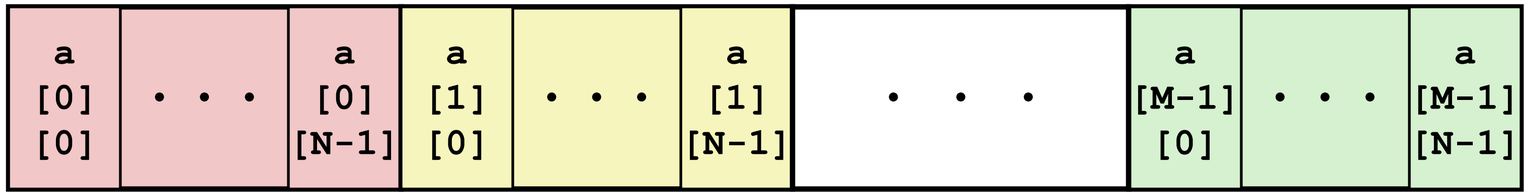
Locality Example

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Answer: no, unless…

M is very small

- Question: Does this function have good locality with respect to array `a`?

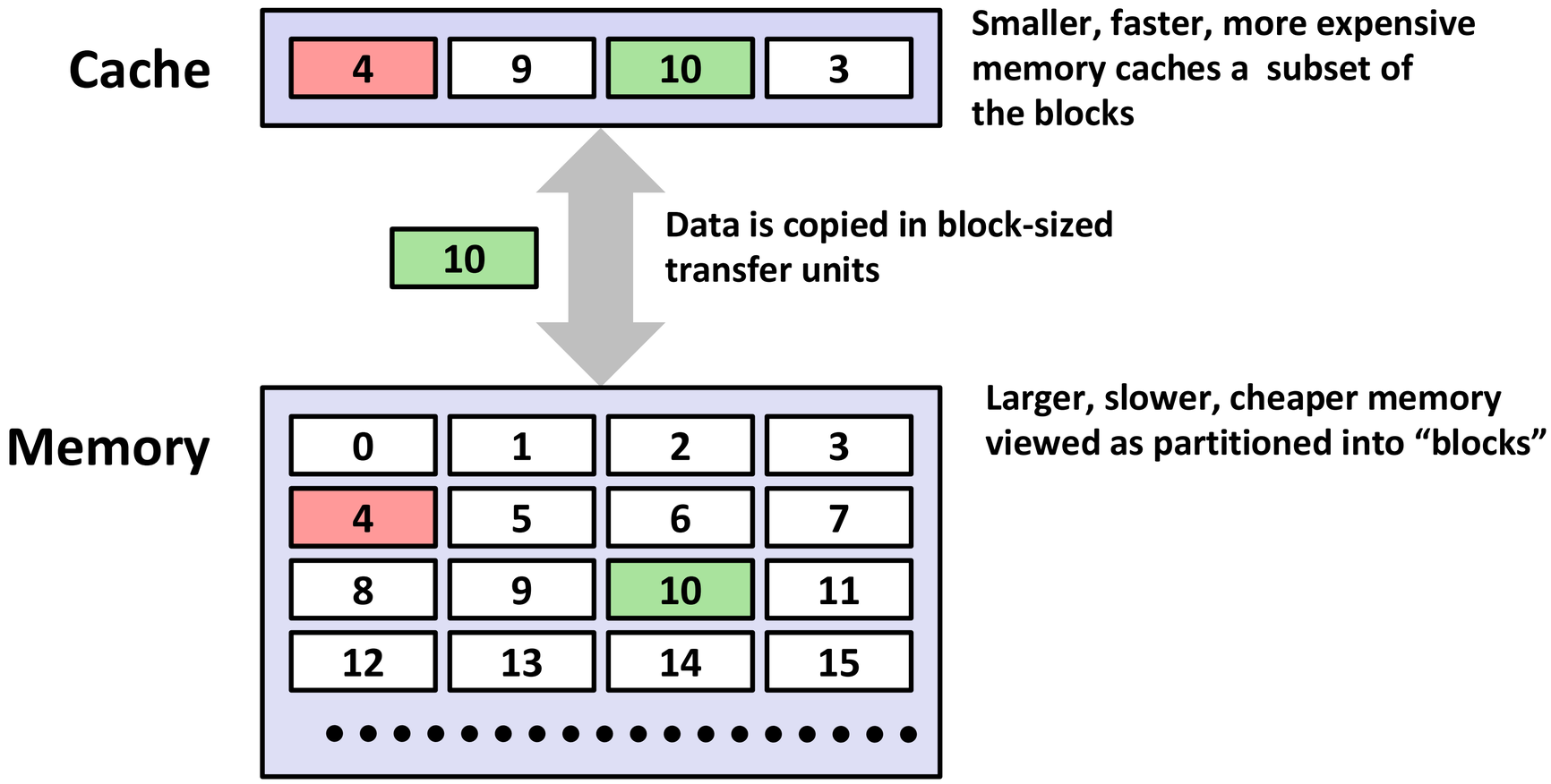


Example Exam Question

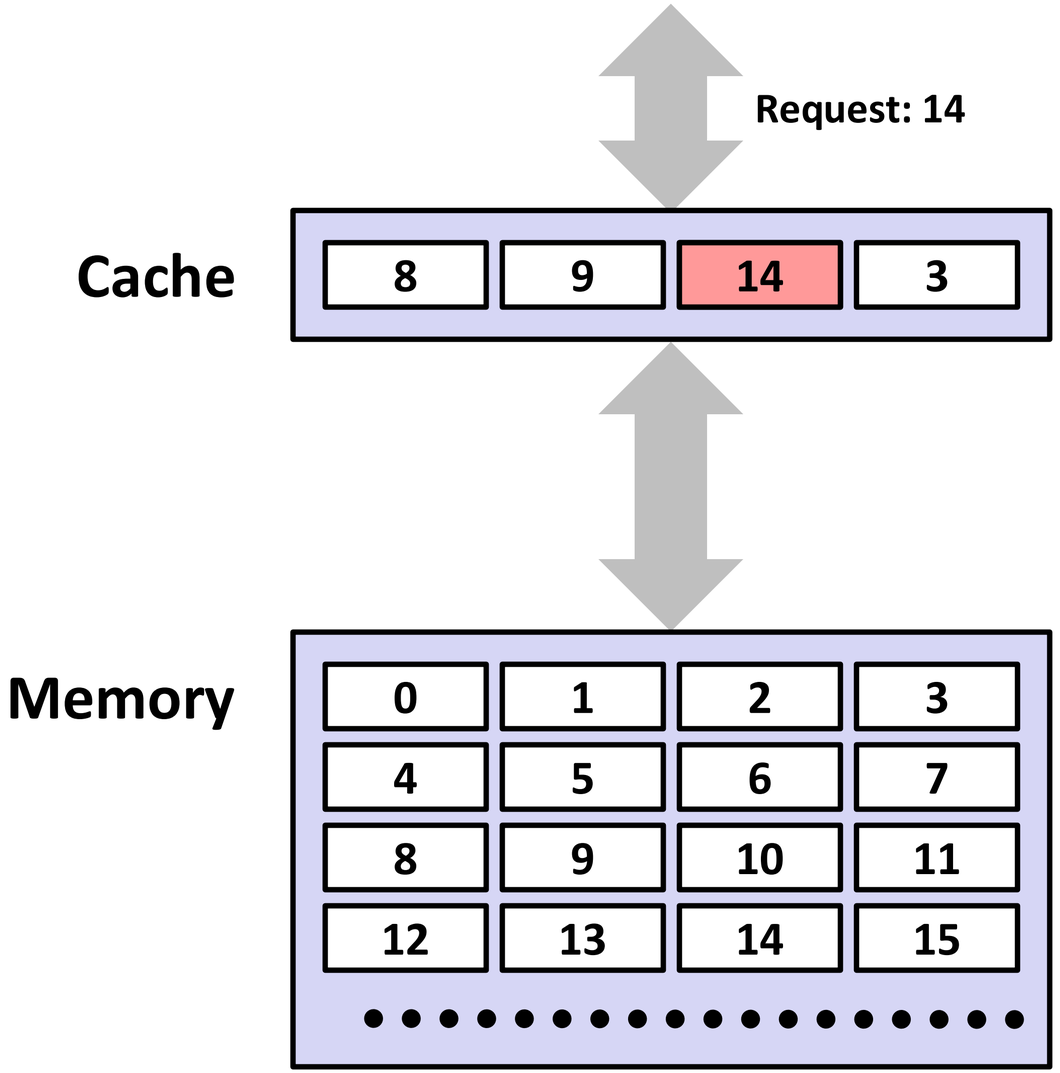
```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < M; k++)
                sum += a[k][i][j];
    return sum;
}
```

- Question: Can you permute the loops so that the function scans the 3-d array `a` with a stride-1 reference pattern (and thus has good spatial locality)?

Cache in action



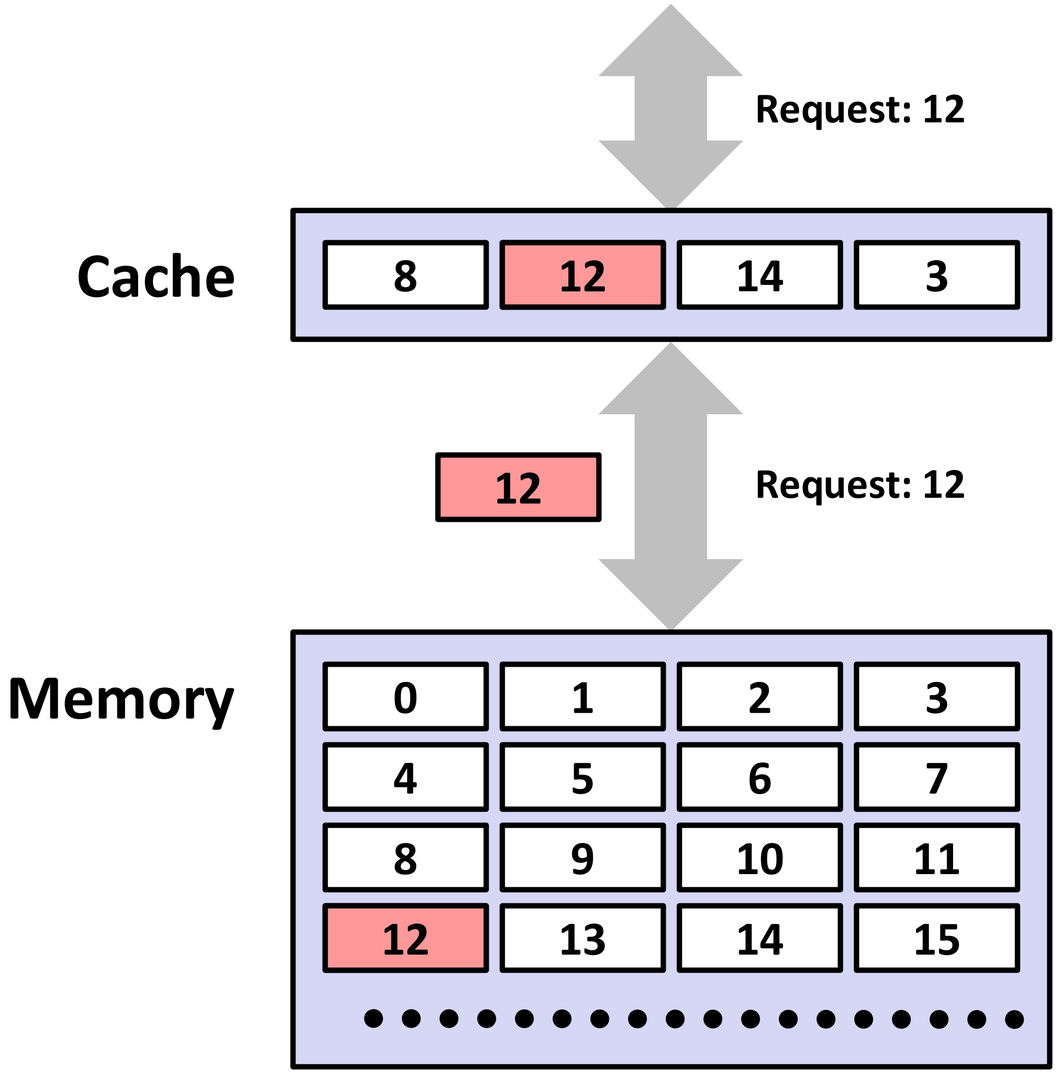
General Cache Concepts: Hit



Data in block 14 is needed

Block 14 is in cache:
Hit!

General Cache Concepts: Miss



Data in block 12 is needed

Block 12 is not in cache:
Miss!

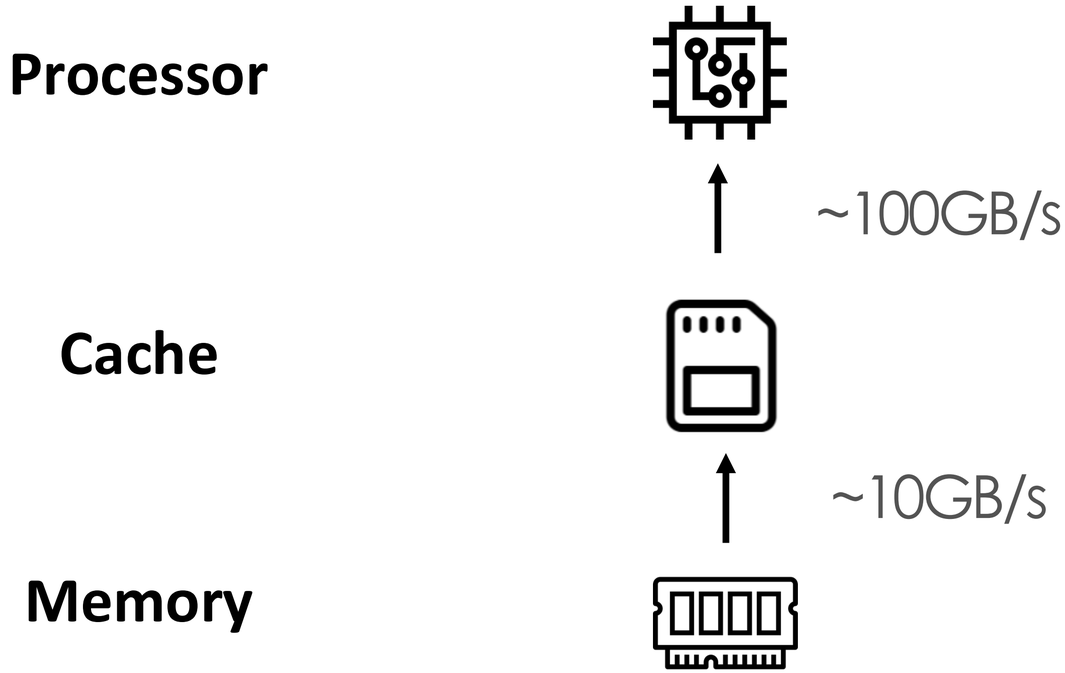
Block 12 is fetched from memory

Block 12 is stored in cache

- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

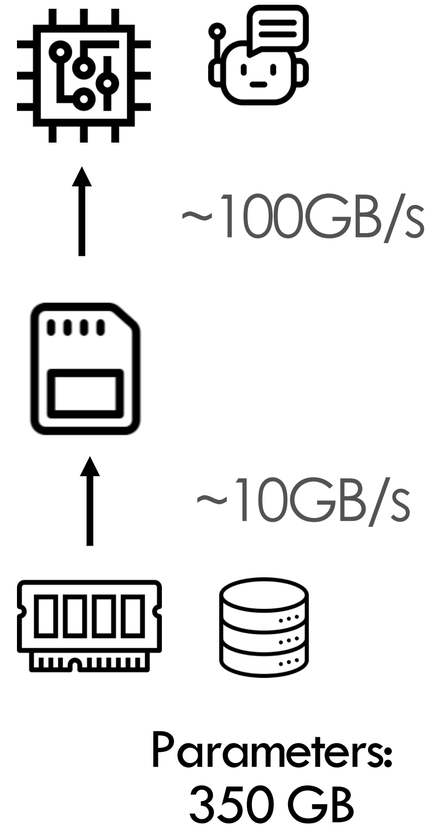
Cache in action

- If always cache hit, bandwidth?
- If always cache miss, bandwidth?



Open Question in Cache: ChatGPT

- ChatGPT: every time ChatGPT outputs token, it needs to see 350 GB parameters
- How to optimize this?



Foundation of Data Systems

- Computer Organization
  - Representation of data
  - processors, memory, storage
- **OS basics**
  - Process, scheduling
  - Memory

What is Operation System?

- Layers between applications and hardware

- OS makes computer hardware useful to programmers
  - Otherwise, users need to speak machine code to computer
- **[Usually]** Provides abstractions for applications
  - Manages and hides details of hardware
  - Accesses hardware through low/level interfaces unavailable to applications
- **[Often]** Provides protection
  - Prevents one app/user from clobbering another

A Primitive OS v1

- OS v1: just a library of standard services [no protection]

- Simplifying assumptions:
  - System runs one program at a time
  - No bad users or programs (?)
- Problem: poor utilization
  - poor utilization of hardware (e.g., CPU idle while waiting for disk)
  - poor utilization of human user (must wait for each program to finish)

OS v2: Multi-tasking

- Say: we extend the OS a bit to support many APPs
  - When one process blocks (waiting for disk, network, user input, etc.) run another process

- Problem: What can ill-behaved process do?
  - Go into infinite loop and never relinquish CPU
  - Scribble over other processes' memory to make them fail
- OS provides mechanisms **protection** to address these problems:
  - Preemption – take CPU away from looping process
  - Memory protection – protect one process' memory from one another

What is A Real OS?

- OS: manage and assign hardware resources to apps
- Goal: with N users/apps, system not N times slower
  - **Idea:** Giving resources to users who actually need them
- What can go wrong?
  - One app can interfere with other app (need **isolation**)
  - Users are gluttons, use too much CPU, etc. (need **scheduling**)
  - Total memory usage of all apps/users greater than machine's RAM (need **memory management**)
  - Disks are shared across apps / users and must be arranged properly (need **file systems**)

Summary of OS: a software between apps and hardware

- Goal 1: Provide convenience to users
- Goal 2: Efficiency -- Manage compute, memory, storage resources
  - Goal 2.1: Running N processes Not N times slower
    - As fast as possible
  - Goal 2.2: Running N apps
    - Even when their total memory >> physical memory cap
- Goal 3: Provide **protection**
  - One process won't mess up the entire computer
  - One process won't mess up with other processes

Process management

Memory management

System calls

Summary of OS: a software between apps and hardware

- Goal 1: Provide convenience to users
- Goal 2: Efficiency -- Manage compute, memory, storage resources
  - Goal 2.1: Running N processes Not N times slower
    - As fast as possible
  - Goal 2.2: Running N apps
    - Even when their total memory >> physical memory cap
- **Goal 3: Provide protection**
  - **One process won't mess up the entire computer**
  - **One process won't mess up with other processes**

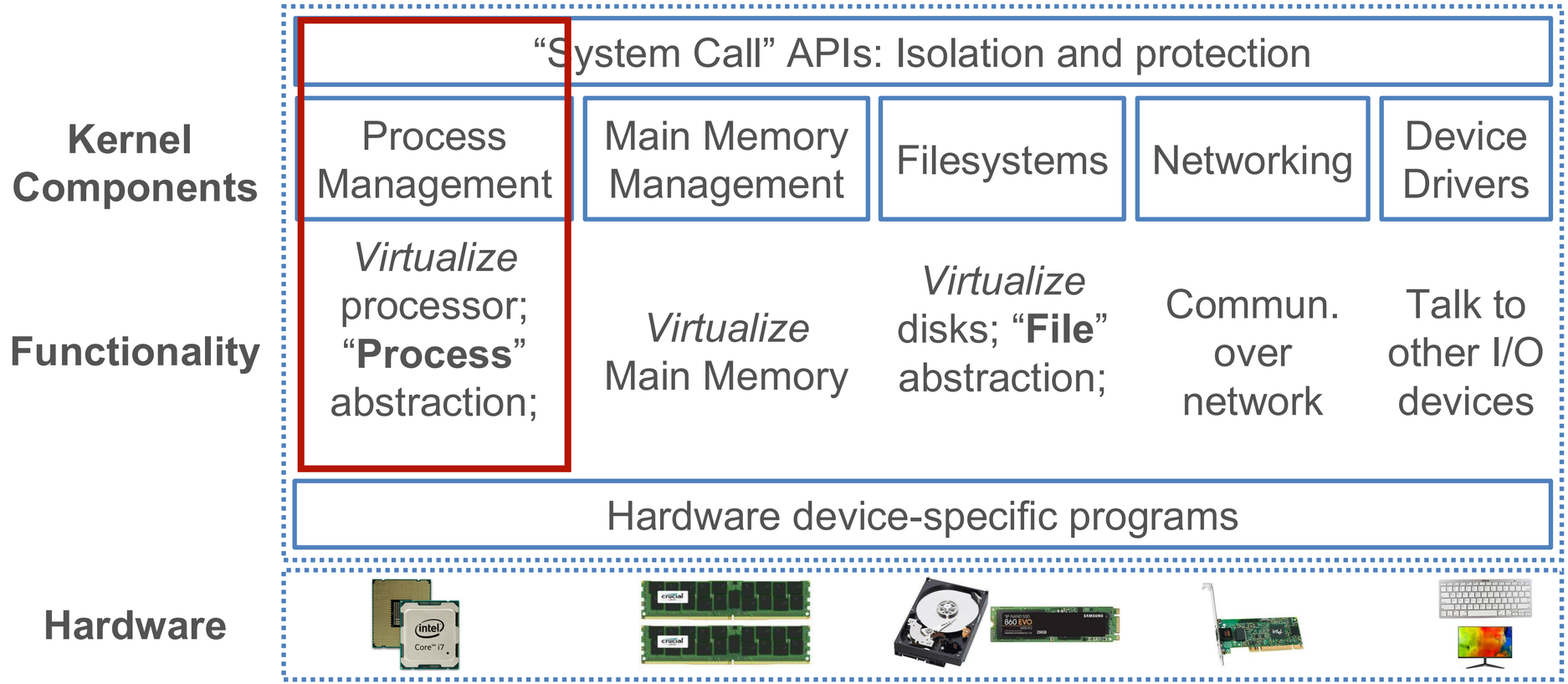
Process management

Memory management

System calls

OS provides Isolation using System Calls

- **System call:** The layer for isolation -- it abstracts the hardware and APIs for programs to use

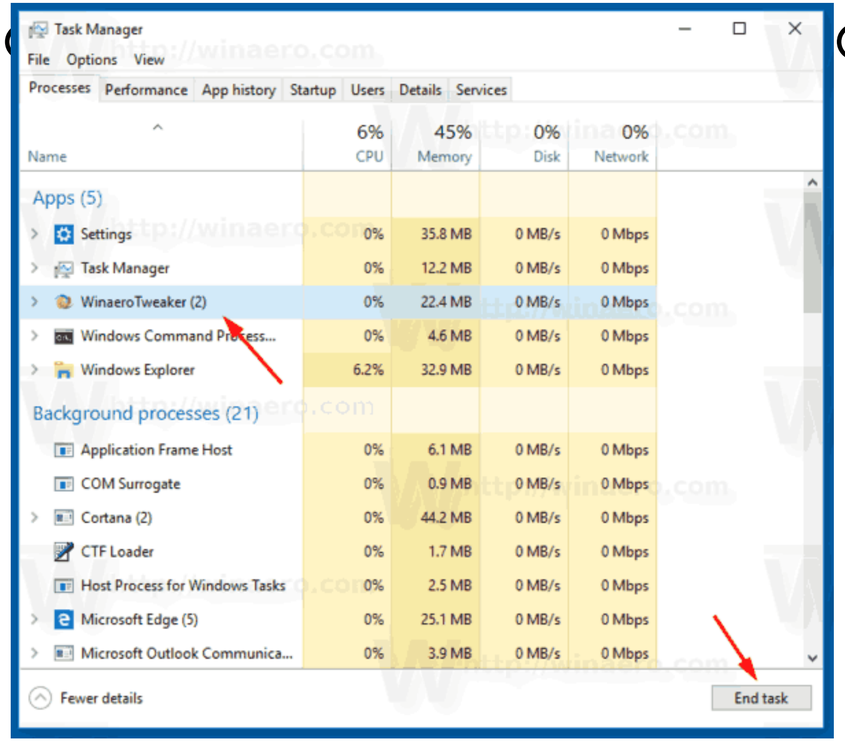


Foundation of Data Systems

- Computer Organization
  - Representation of data
  - processors, memory, storage
- OS basics
  - **Process, scheduling**
  - Memory

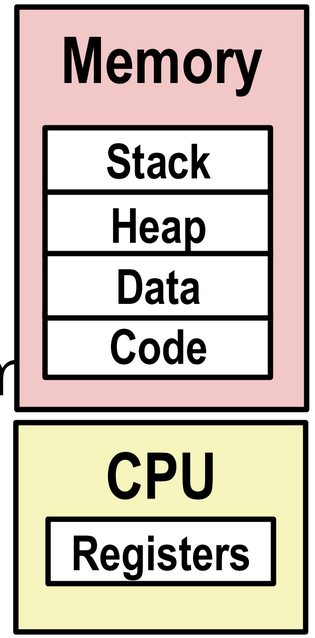
Processes - the central abstraction in OS

- Definition: A *process* is an instance of a running program.
  - One of the most profound ideas in computer science
  - N… …or

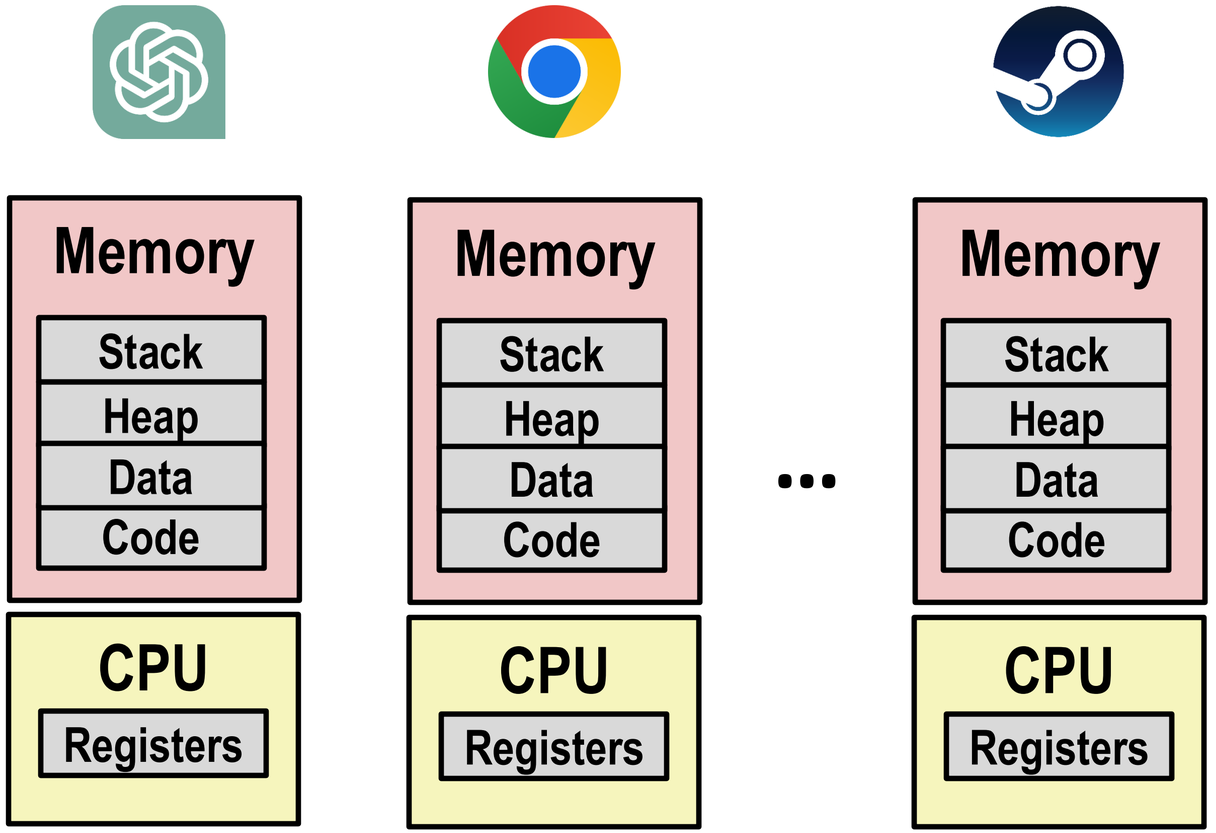


Processes - the central abstraction in OS

- Process provides each program with two key abstractions (for resources):
  - **Compute Resource**
    - Each program seems to have exclusive use of the CPU
    - Provided by kernel mechanism called *context switching*
  - **Memory Resource**
    - Each program seems to have exclusive use of main mem
    - Provided by kernel mechanism called virtual memory



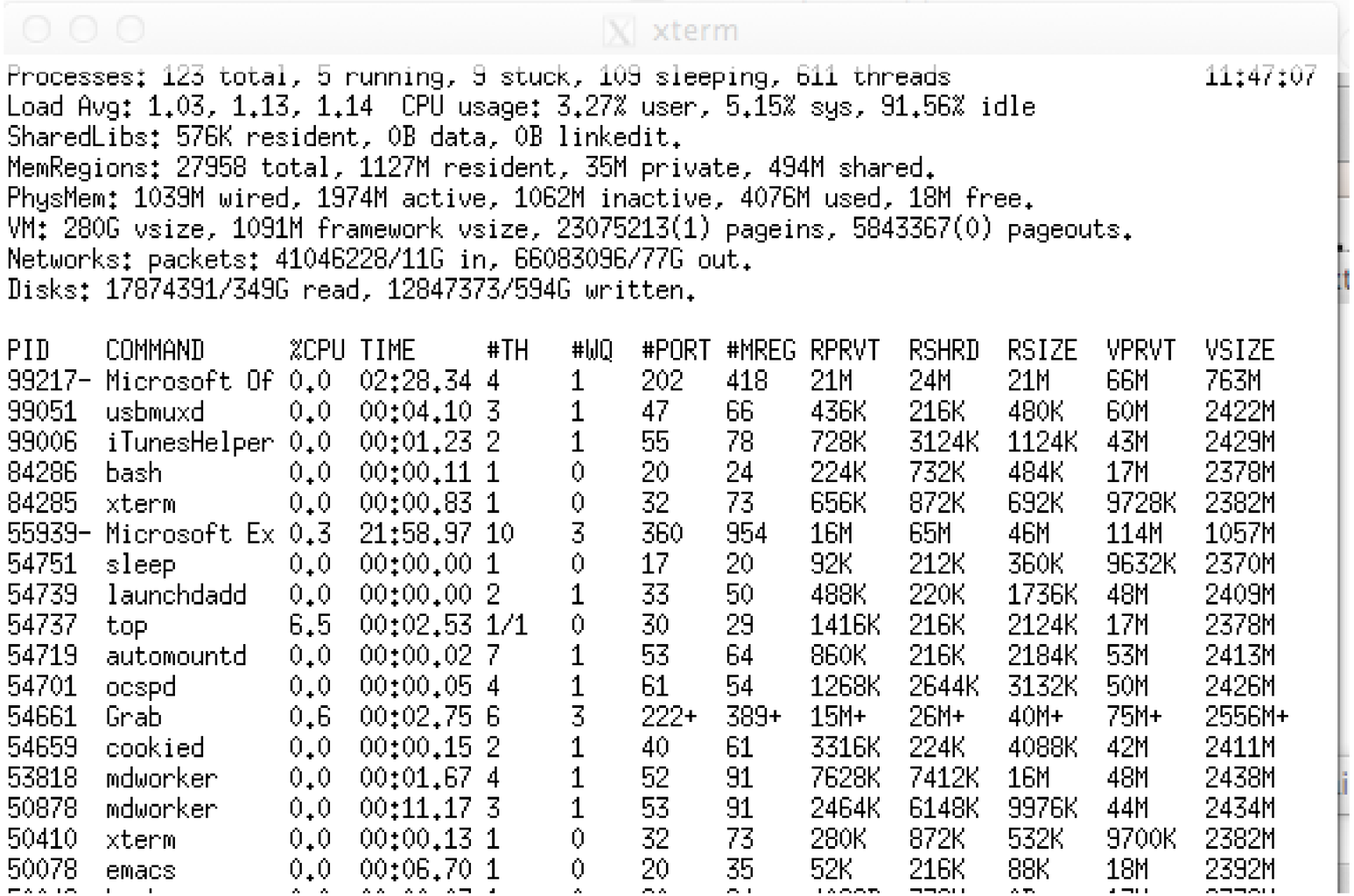
Multiprocessing in OS: The Illusion



- Computer runs many processes simultaneously

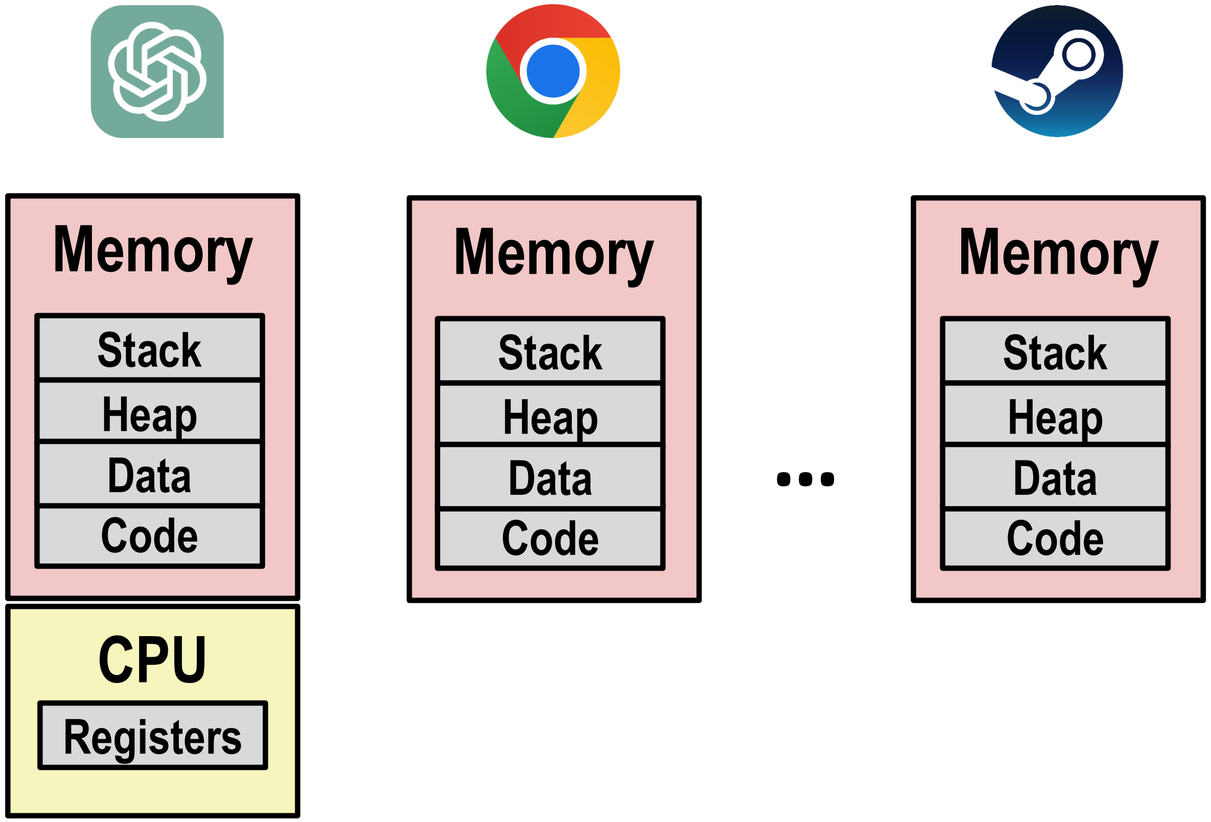
Multiprocessing Example

top command in terminal: many processes, Identified by Process ID (**PID**)



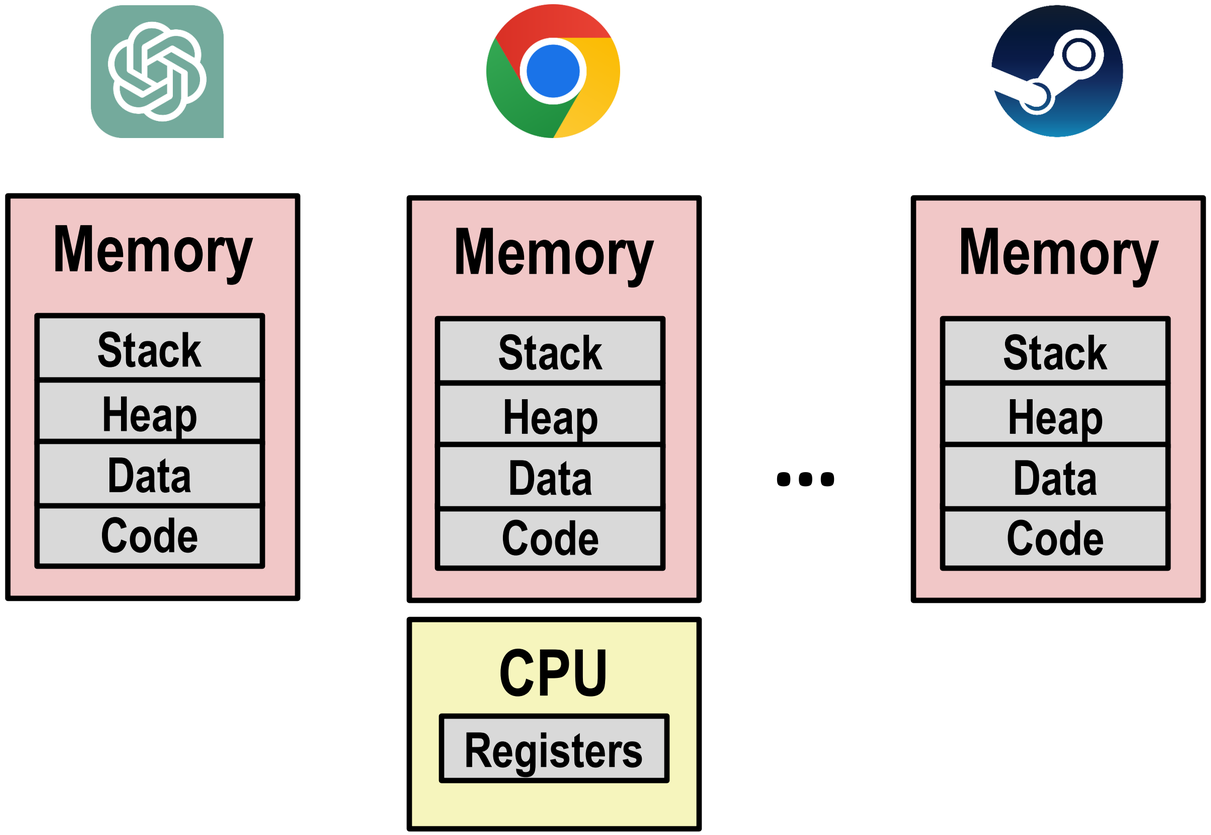
Multiprocessing: A strawman solution

- Assign individual memory (say 1/3) to each APP
- Assign CPU to work on an APP until completion -> then next



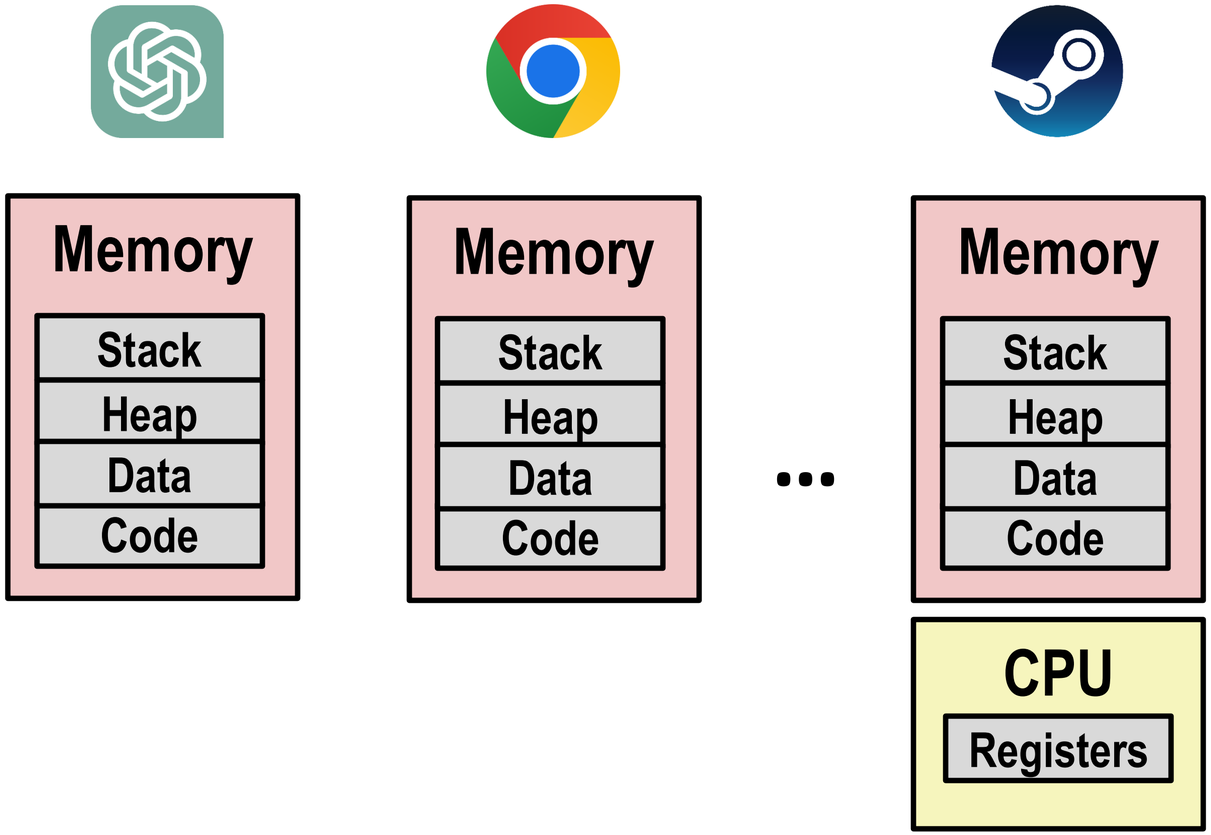
Multiprocessing: A strawman solution

- Assign individual memory (say 1/3) to each APP
- Assign CPU to work on an APP until completion -> then next



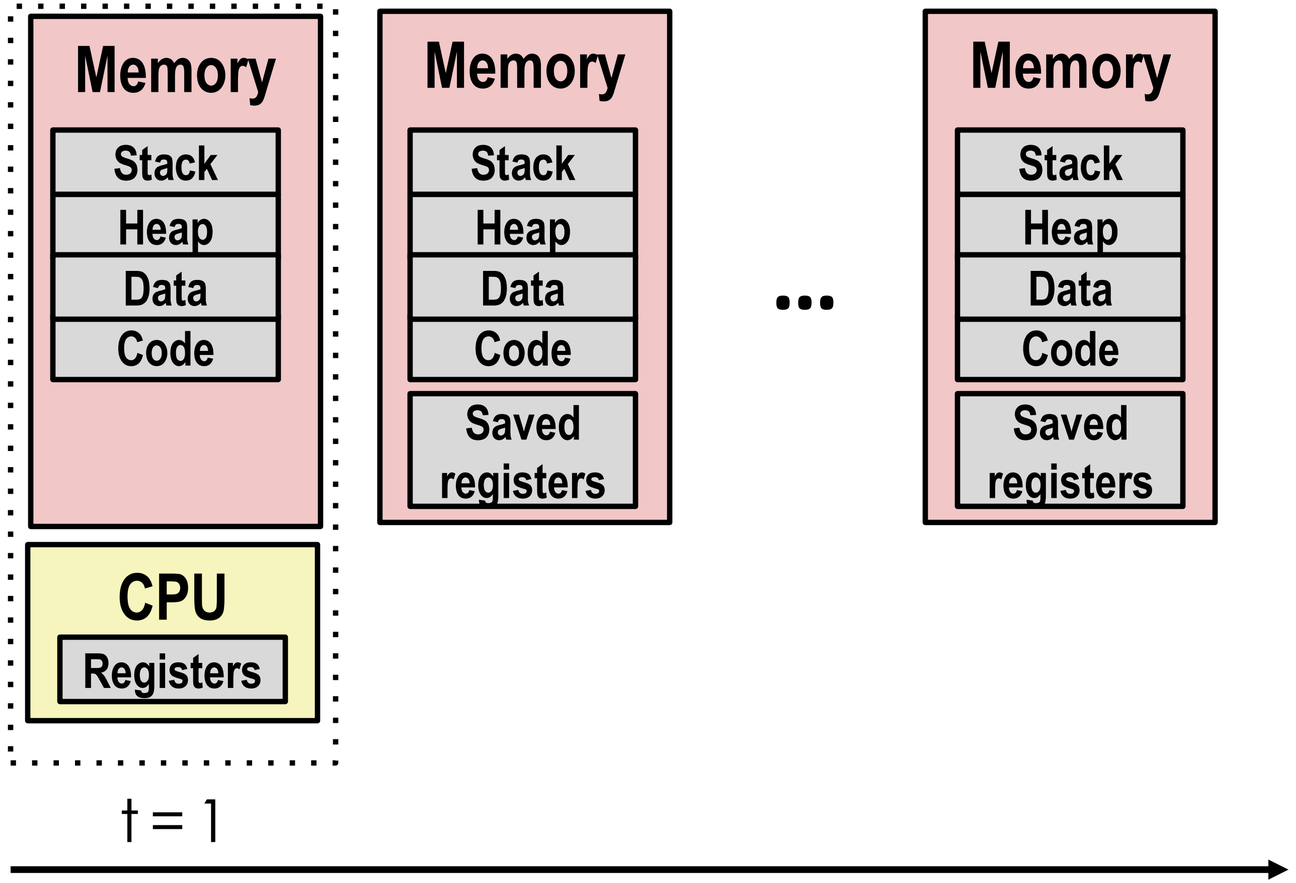
Multiprocessing: A strawman solution

- Assign individual memory (say 1/3) to each APP
- Assign CPU to work on an APP until completion -> then next



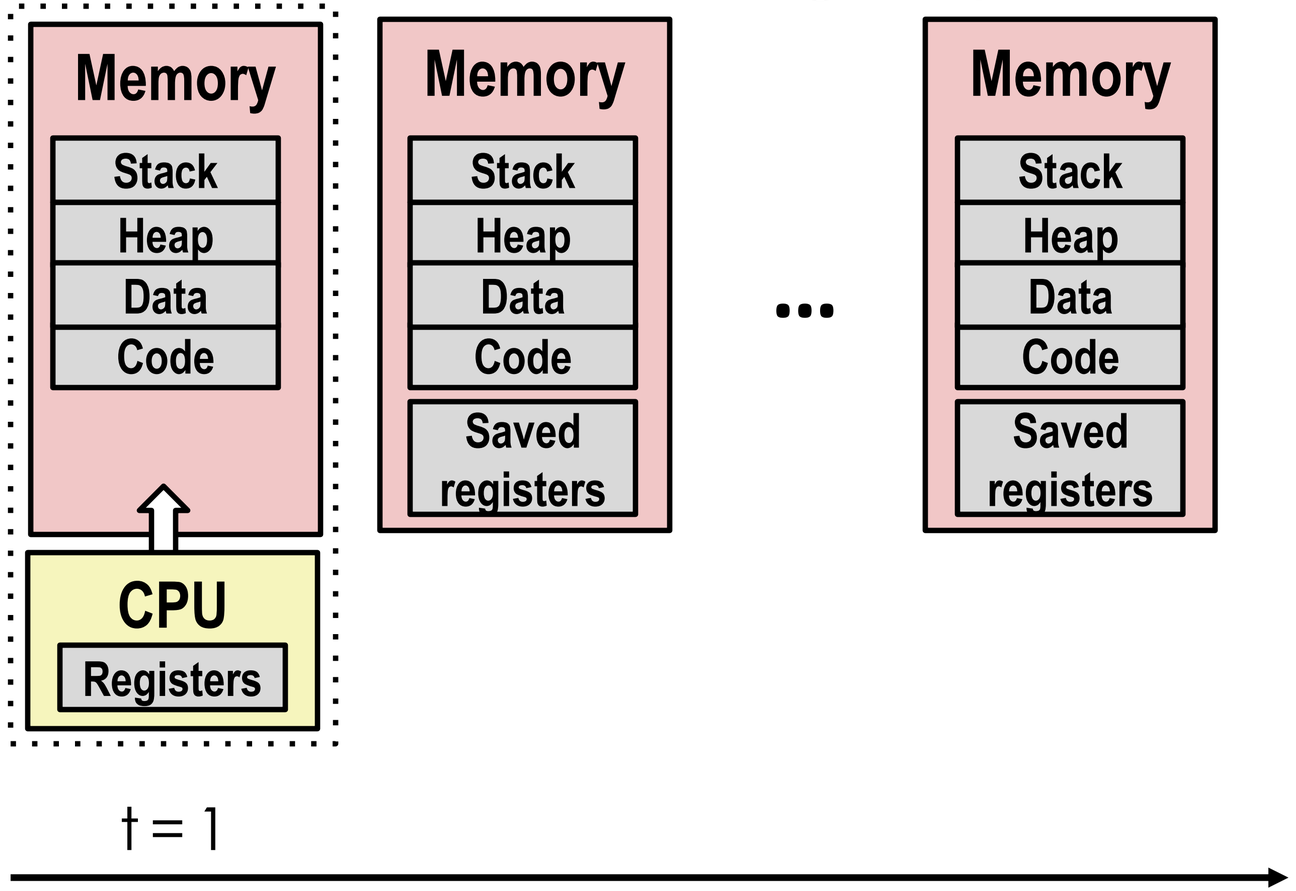
G1. Convenient?
G3: protection?
G2. Efficient?
!!!we are N times slower when running N processes

Multiprocessing: Time sharing of processors



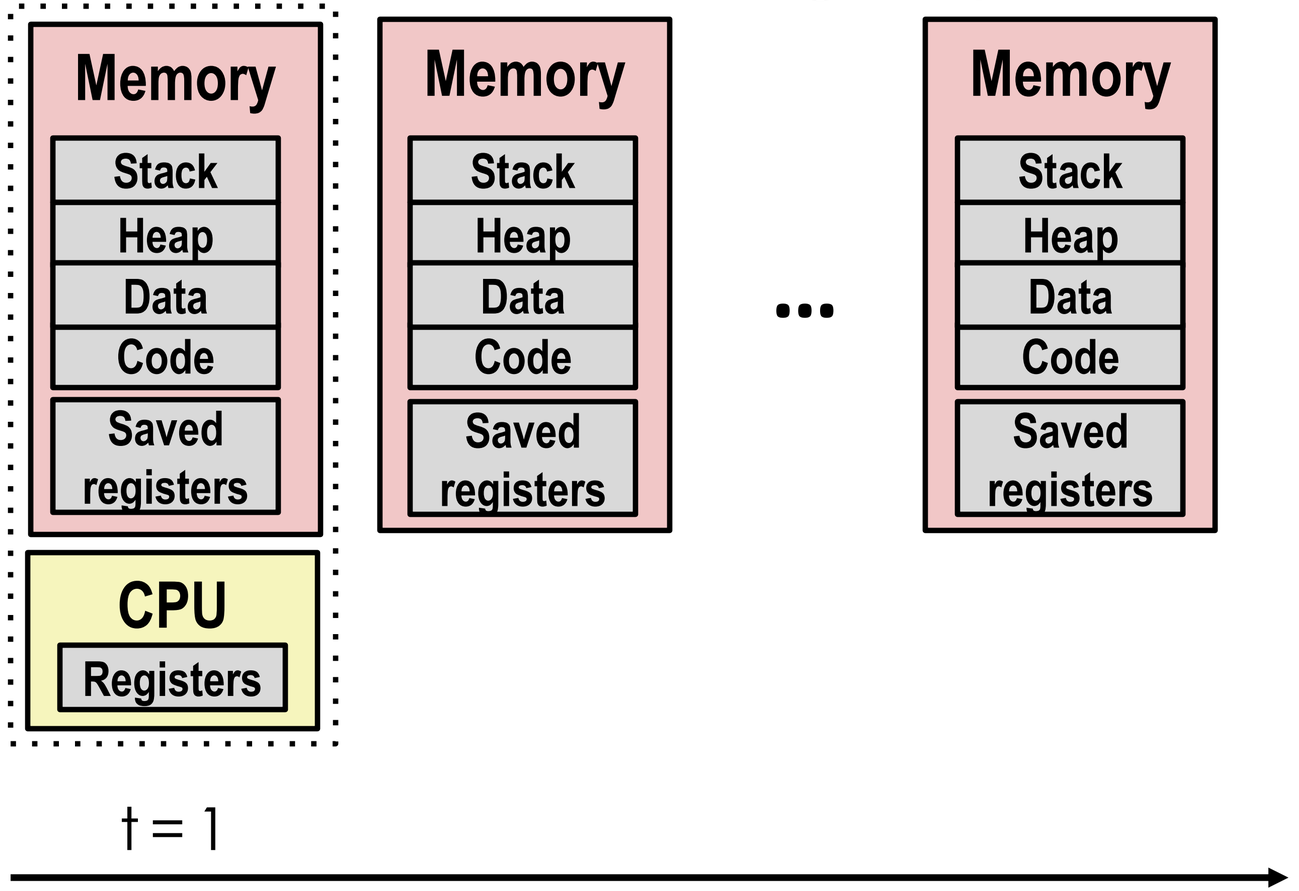
- Idea: Virtualize the CPU time as time slices
- Assign time slices to different processes

Multiprocessing: Time sharing of processors



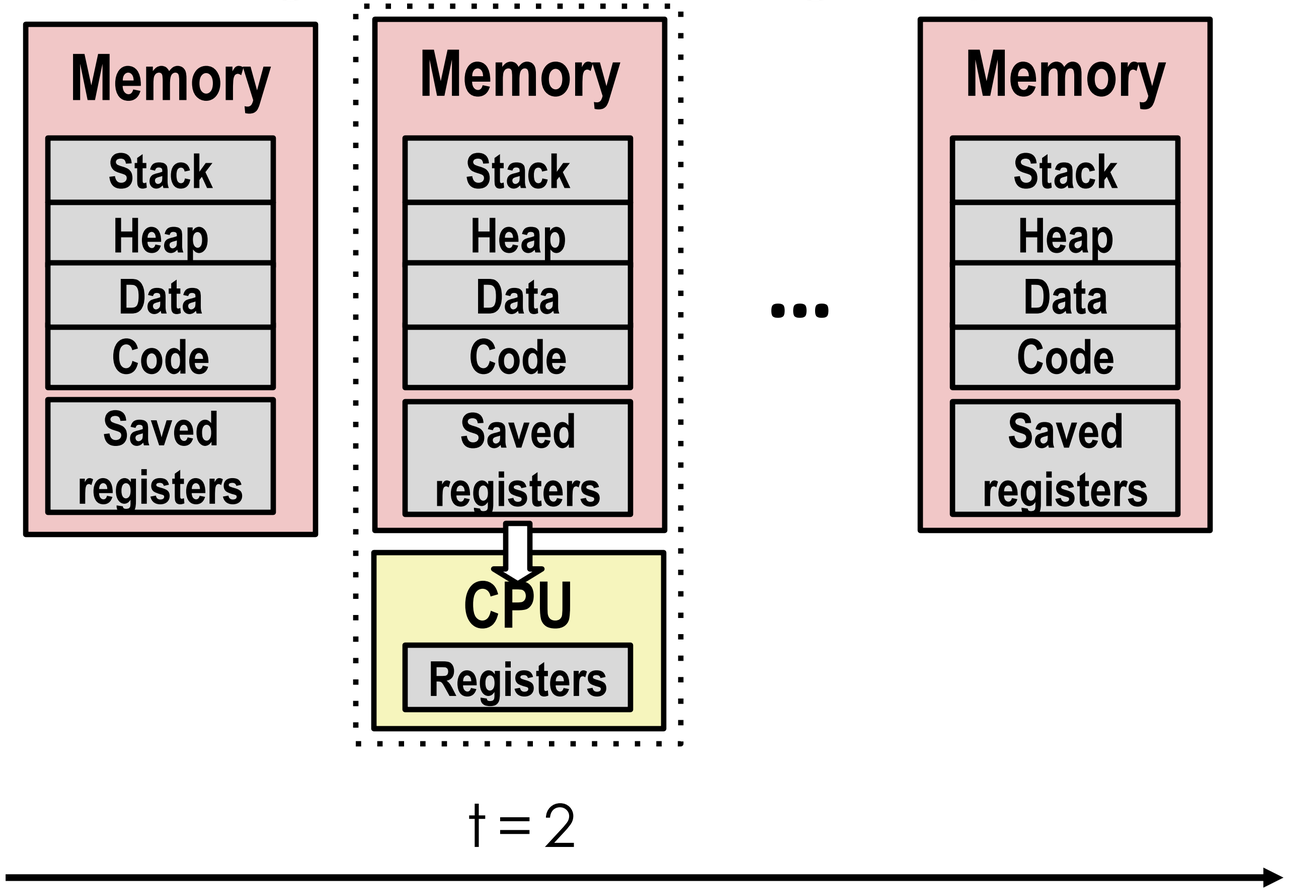
- Save current registers in memory

Multiprocessing: Time sharing of processors



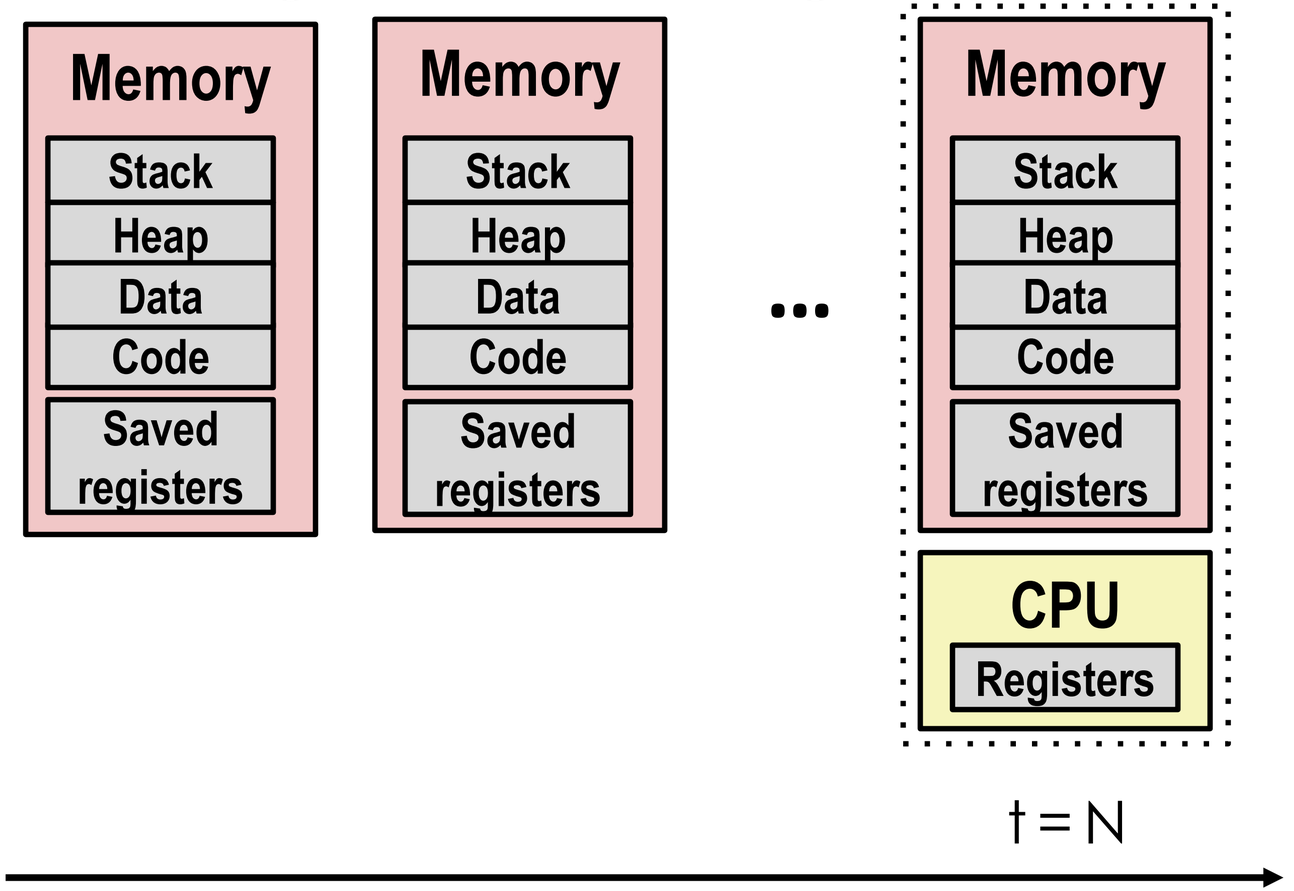
- Save current registers in memory

Multiprocessing: Time sharing of processors



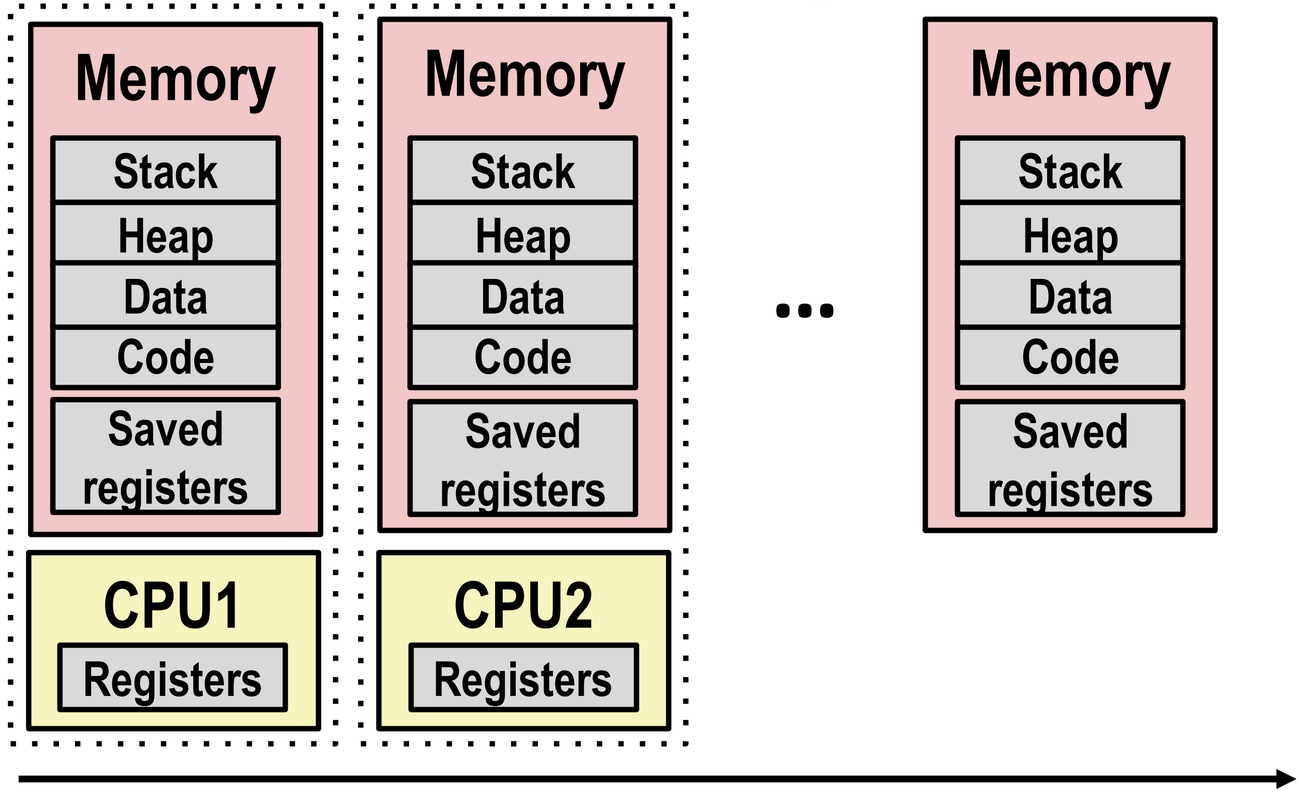
- Assign time slice t = 2 to the next process
- Resume progress: Move Saved registers from memory to CPU

Multiprocessing: Time sharing of processors



- Then we repeat.
- This is called **context switch**

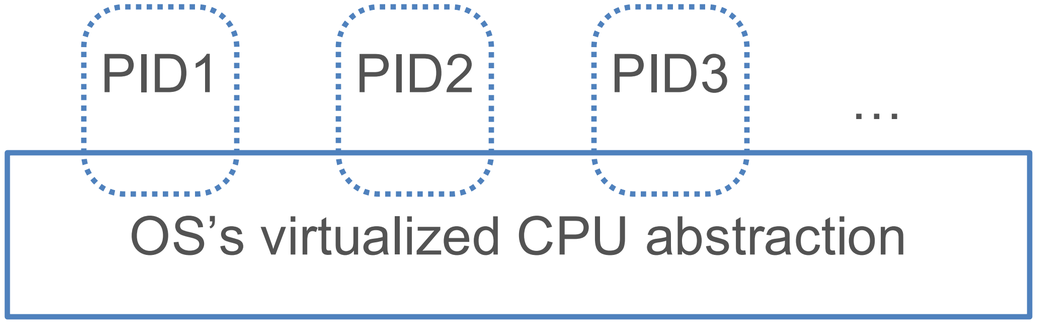
Multiprocessing: Time sharing of multiple processors



Multiple CPU cores?

1. All processors sweep from left (1st process) to right (last process)
2. Each process accounts for ½ of the processes

Let's Implement It!

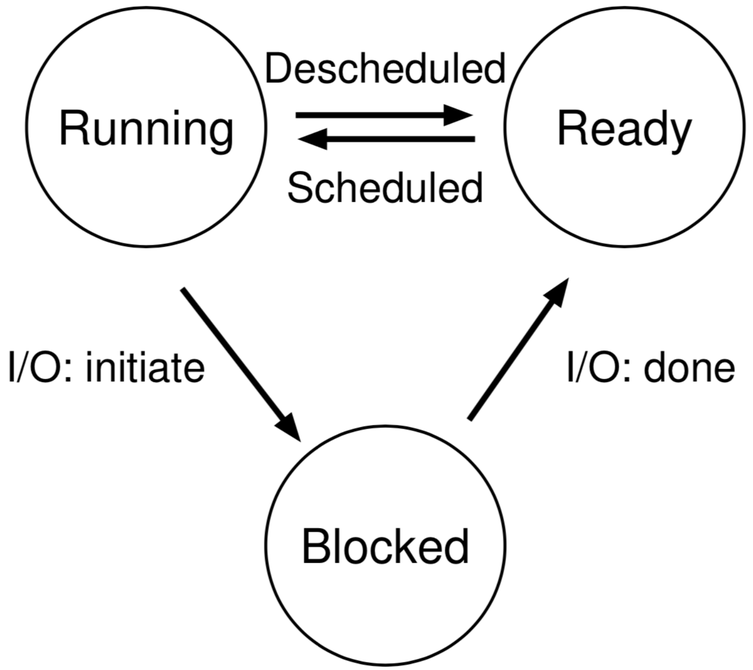


GAP1: How to virtualize CPU resources **temporally** and **spatially**?

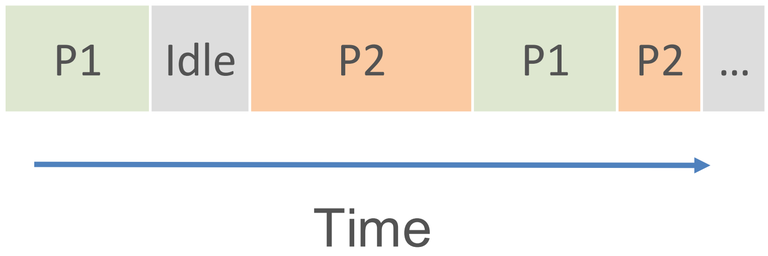
Physical Processor

Temporal Abstraction: Process State and CPU Time

- OS keeps moving processes between 3 states:



- Gantt Chart: A viz. to show what process runs when (on processor)



Scheduling question naturally emerges:
Q: how to schedule processes on time axis so **the objective** is optimal?

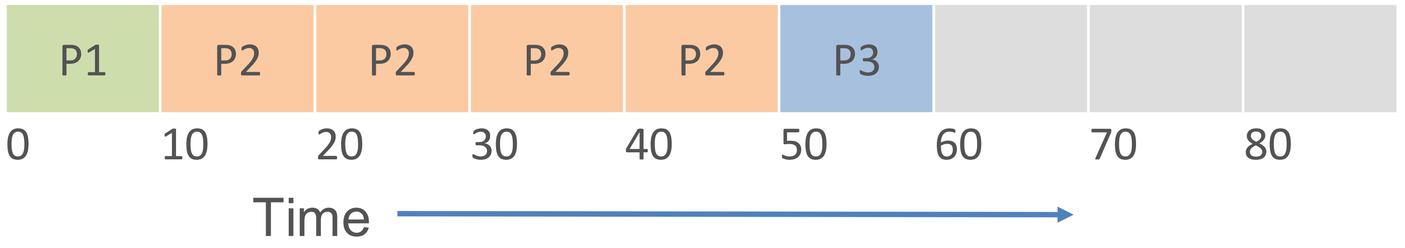
Scheduling Policies/Algorithms

- Schedule: Record of what process runs on each CPU when
- Policy controls how OS time-shares CPUs among processes
- Key terms for a process (aka job):
  - Arrival Time: Time when process gets created
  - Job Length: Duration of time needed for process
  - Start Time: Time when process first starts on processor
  - Completion Time: Time when process finishes/killed
  - Response Time = Start Time — Arrival Time
  - Turnaround Time = Completion Time — Arrival Time
- Workload: Set of processes, arrival times, and job lengths that OS Scheduler has to handle

Scheduling Policy: FIFO

- First-In-First-Out aka First-Come-First-Serve (FCFS)
- Ranking criterion: Arrival Time; no preemption allowed

Example: P1, P2, P3 of lengths 10,40,10 units arrive closely in that order



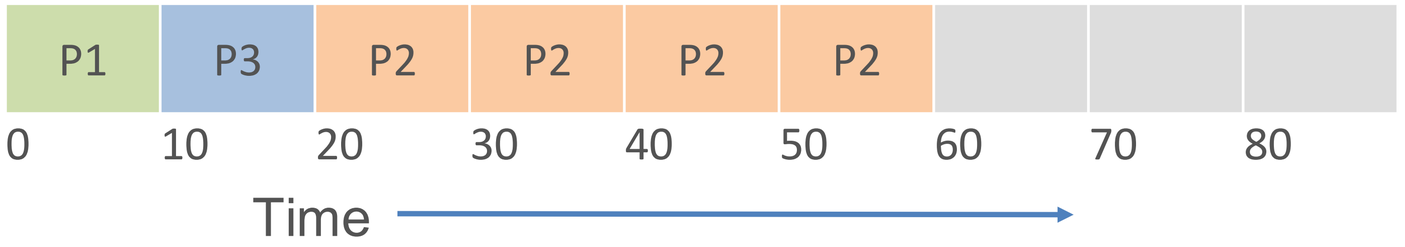
Process	Arrival Time	Start Time	Completion Time	Response Time	Turnaround Time
P1	0	0	10	0	10
P2	0	10	50	10	50
P3	0	50	60	50	60
			Avg:	20	40

- Main con: Short jobs may wait a lot, aka “Convoy Effect”

Scheduling Policy: SJF

- Shortest Job (next) First
- Ranking criterion: Job Length; no preemption allowed

Example: P1, P2, P3 of lengths 10,40,10 units arrive closely in that order



Process	Arrival Time	Start Time	Completion Time	Response Time	Turnaround Time
P1	0	0	10	0	10
P2	0	20	60	20	60
P3	0	10	20	10	20
			Avg:	10	30

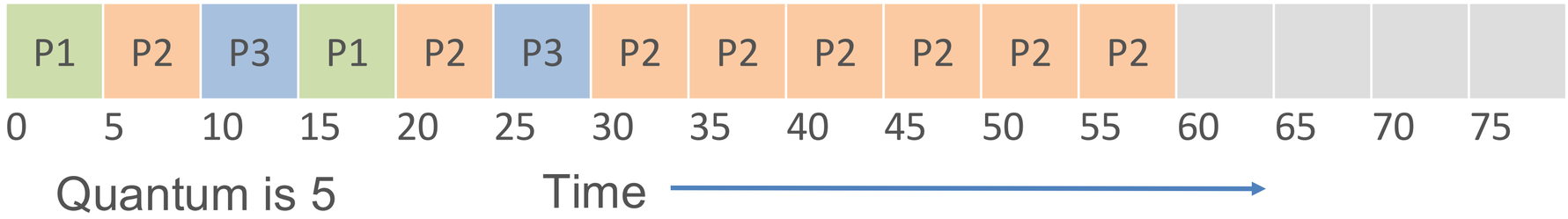
(FIFO Avg: 20 and 40)

- Main con: Not all Job Lengths might be known beforehand
- Long processes may be held off indefinitely

Example Exam Q1: Round Robin Schedule

- RR does not need to know job lengths
- Fixed time *quantum* given to each job; cycle through jobs

Example: P1, P2, P3 of lengths 10,40,10 units arrive closely in that order

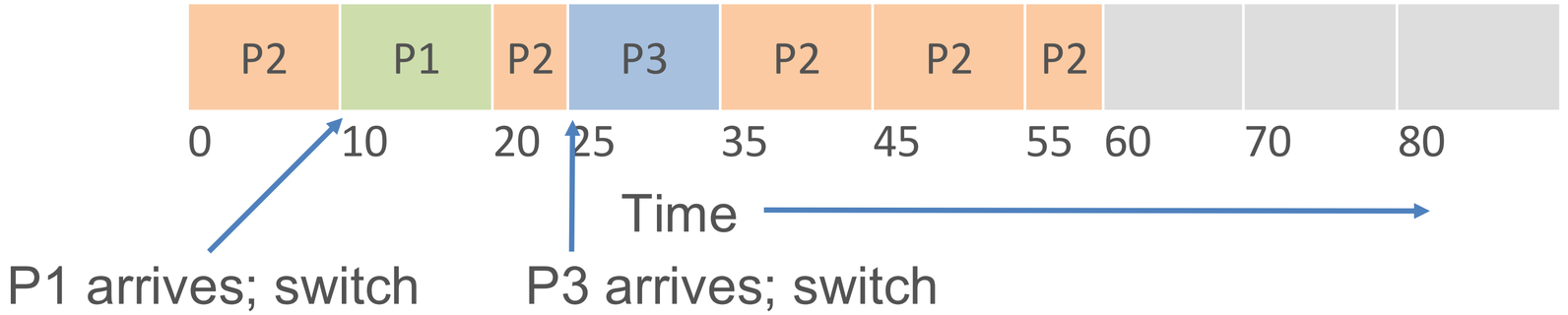


- RR is often very fair, but Avg Turnaround Time goes up!

Example Exam Q2: SCTF

- Shortest Completion Time First
- Jobs might not all arrive at same time; preemption possible

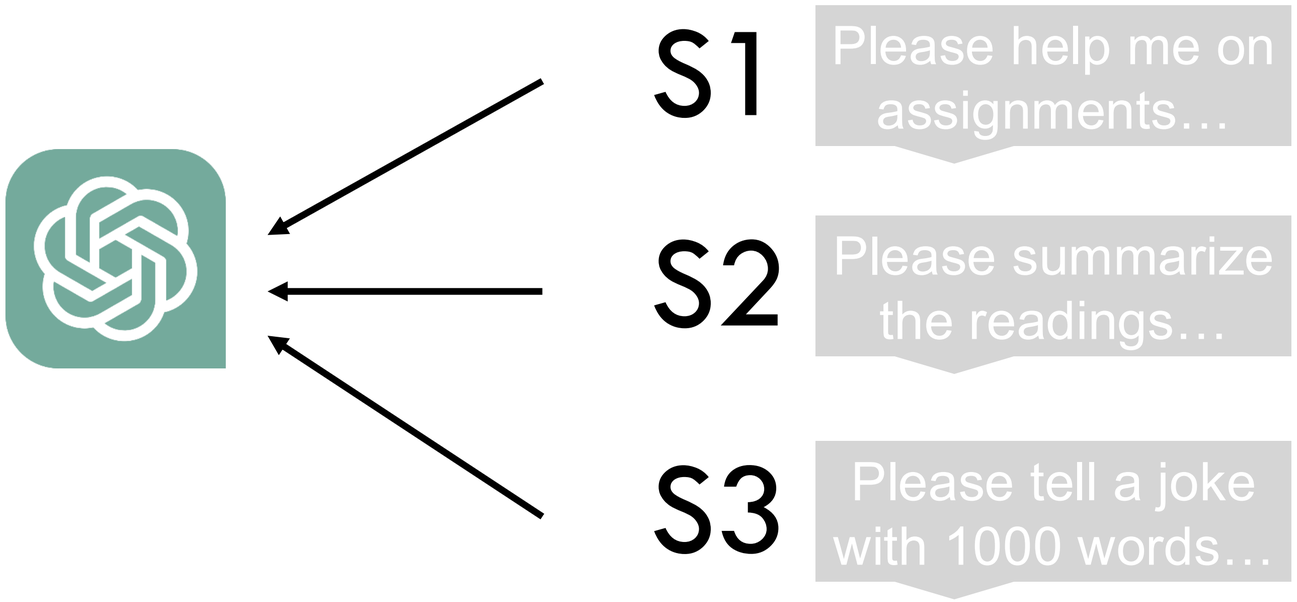
Example: P1, P2, P3 of lengths 10,40,10 units arrive at different times



Scheduling Policies/Algorithms

- In general, not all Arrival Times and Job Lengths will be known beforehand. But preemption is possible.
- Key Principle: Inherent tension in scheduling between overall workload *performance* and allocation *fairness*
  - Performance metric is usually *Average Turnaround Time*
  - Many fairness metrics exist, e.g., Jain's fairness index
- 100s of scheduling policies studied! Well-known ones: FIFO, SJF, STCF, Round Robin, Random, etc.
  - Different criteria for ranking; preemptive vs not
  - Complex "multi-level feedback queue" schedulers
  - ML-based schedulers are "hot" nowadays!

Scheduling in ChatGPT



- What is the response time
- What is the turnover time
- What is fairness?
- Do we know the job length?
- Can we run S1/S2/S3 together?
- How to schedule?