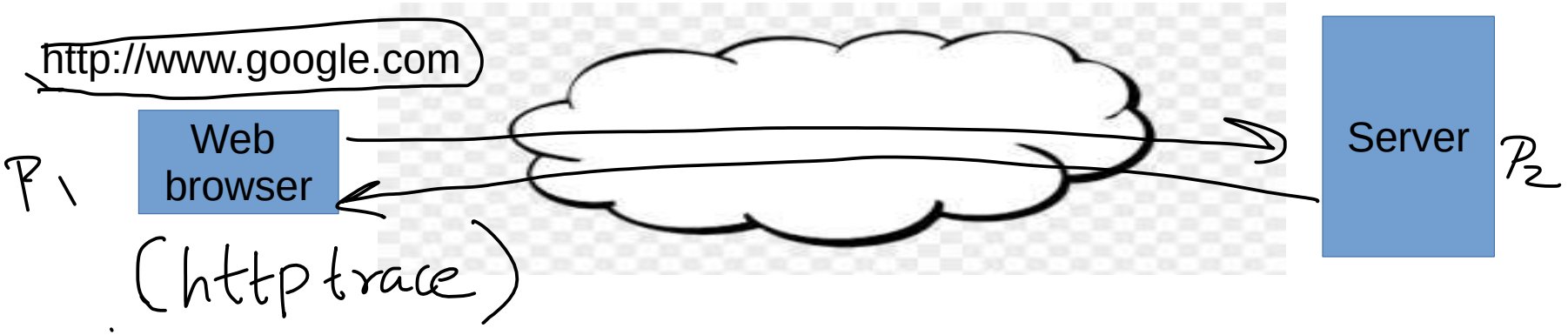
Introduction to Communicating Distributed Processes
Lecture 1

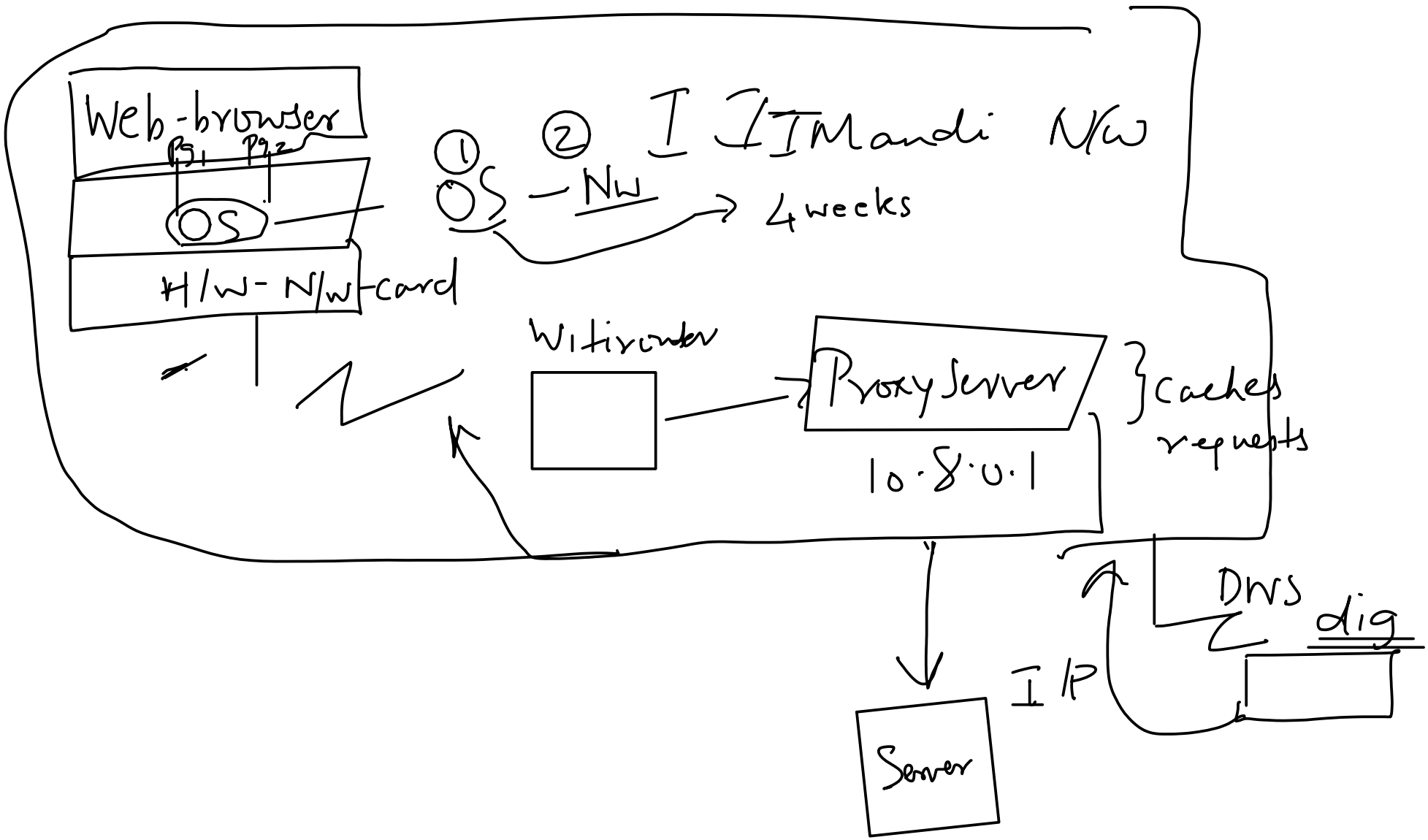
Motivating Example

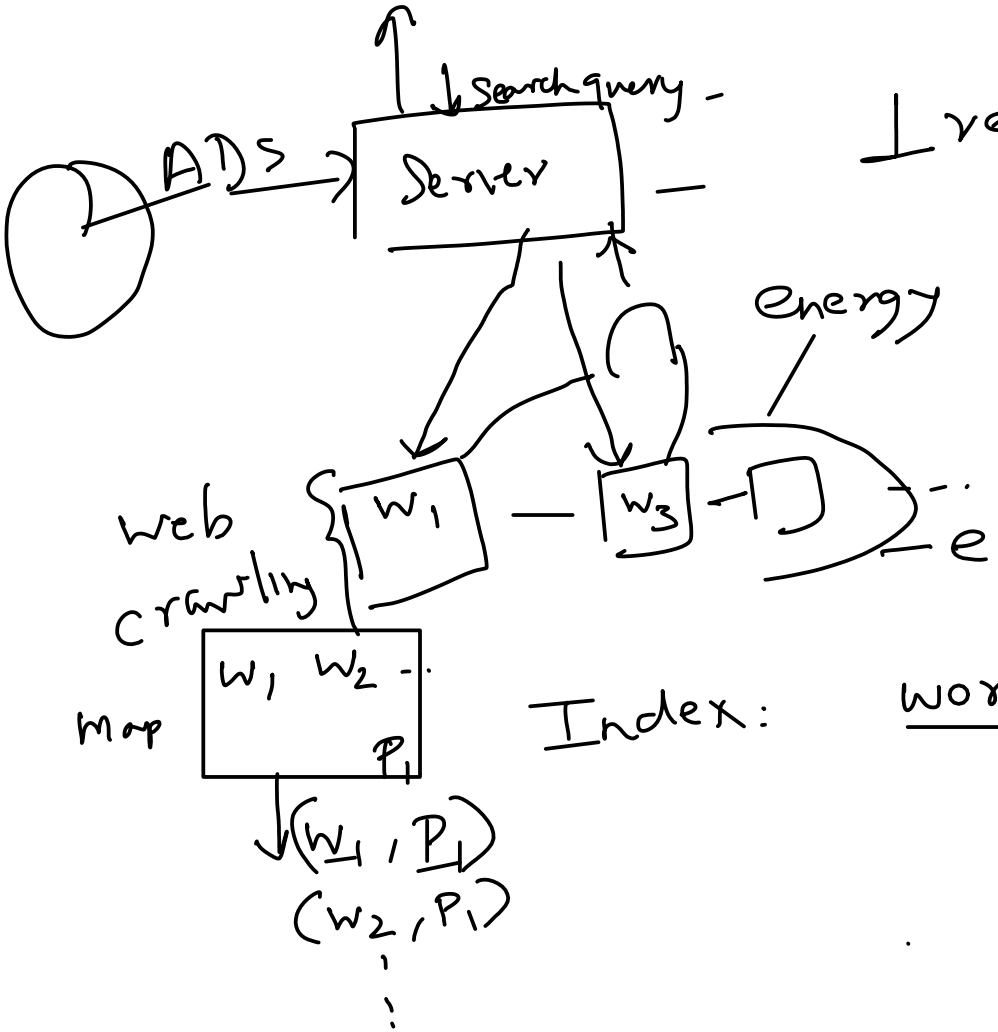


Get - Appⁿ layer protocol

- Flood the msg to all m/c (Broadcast) - infeasible in Internet-scale

Content-Delivery Network - latencies ↓


Web-browser
P1 P2
OS
H/W - N/W card
① OS - NW
② I I I Mandi N/W
4 weeks
Wireless
Proxy Server
10.8.0.1
} caches requests
DNS
dig
IP
Server



1 req — 1000s req/s

Sequential

↓

multithreading

Google data

10000s m/c

Concurrent

ACID

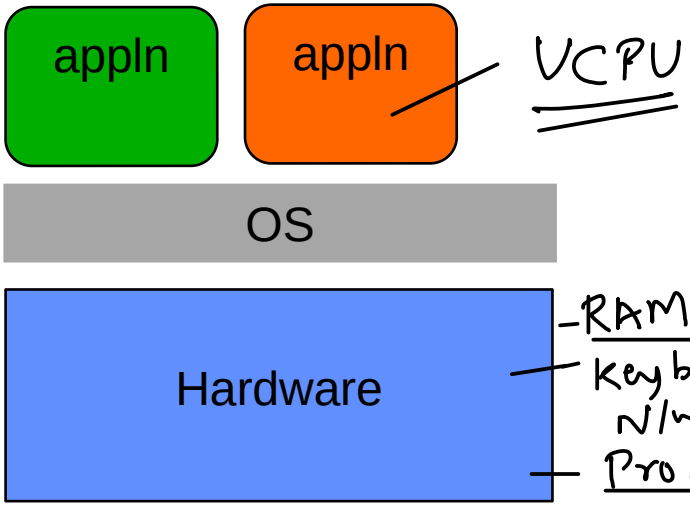
Threads

Index: word → $P_1, P_2, \ldots P_n$

$w_1 \rightarrow P_1, P_2, P_3 \ldots P_n$ — ∩ w_3 -

What is an operating system?

- **Special layer of software that manages a computer's resources for its users and applications**



VCPU

RAM
Keyboard
N/w, disk
Processor

I/O

1. Resource allocator → Gatekeeper, Referee
2. Illusionist – wifi – reliable
   overcomes limitations
   Time sharing
   4 core 270 — 4 cores
   Virtualizes resources – CPU, Mem
3. Common services – cut-paste

OS

Roles played by OS

- Referee
  - Resource allocation
  - Isolation

```
def test():
    while True:
        print 'x=1'
```

  - Communication

Timer Interrupt

OS

Ctrl-C

segmentation fault

2^{32} - 4 GB x 4 GB

64-bit processor : $2^{64} - 1$

- Illusionist
  - Virtualize resources – illusion of reliable service using Wifi, infinite memory, ability to deal with evolving hardware

- Common services
  - Cut, copy and paste across different applications

Evolution of OS: a brief history

- 1945: ENIAC- loading programs / data was difficult and time-consuming
- 1949: EDVAC - Computers got memory
- 1949: BINAC - Programming language
- 1951: UNIVAC – Reusable code
- 1952: Shared IO routines and True assembler
- 1956: Interrupts
- 1954 – 1957: Batch processing, Fortran [architecture independent, high level language]
- 1960s – Multi-programming, multi-processing, virtual memory to make OS portable
- 1960s – disks became mainstream
- 1966 – mini-computers became cheaper [time-sharing system] – Interactive use
- 1969 – Unix Operating System
- 1972 – Virtual machine operating system
- 1973 – Unix written in C [portable]
- Graphical user Interface and then ubiquitous devices

Modern Operating Systems

- Desktop — sudo ~~su~~ — root
- Smartphone — portability, samsung
- Cloud Operating Systems
- Embedded operating systems — app — app, app — OS; motes; Real-time OS

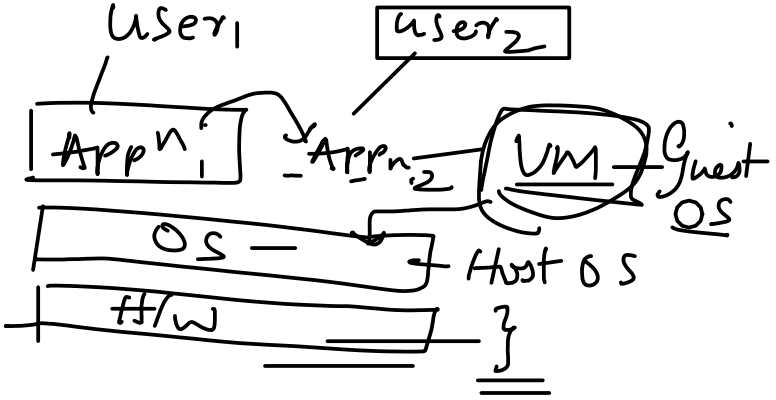
Where is it headed?

- Very large-scale data-centers,very heterogeneous hardware, multi-core machines, large storage

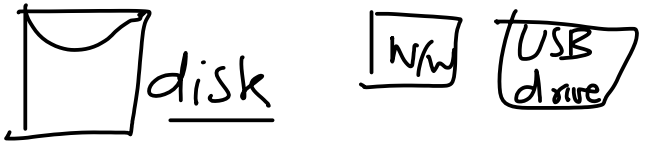
Recap: Sept. 17, 2020

OS plays 3 roles

- Referee – protection
- Illusionist – virtualizes res.
  - convenient abstr.
  - infinite mem.
- Common service
  - VMs
    - ↳ Windows OS on VM
    - Linux OS

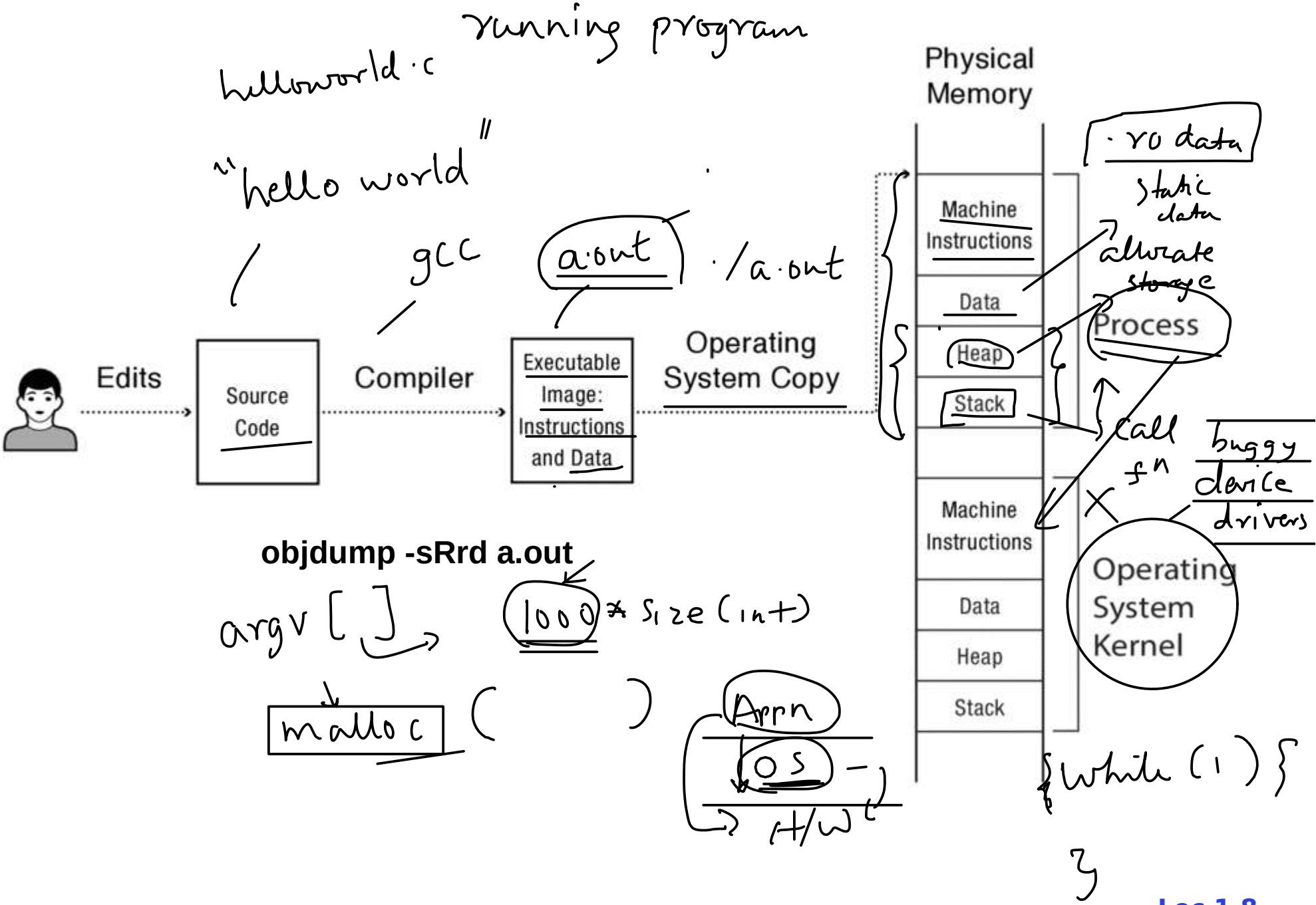


byte-addressable



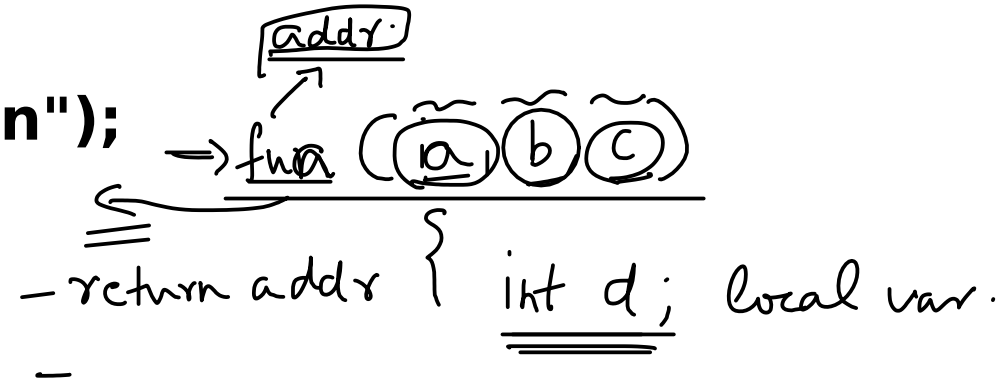
blocks –

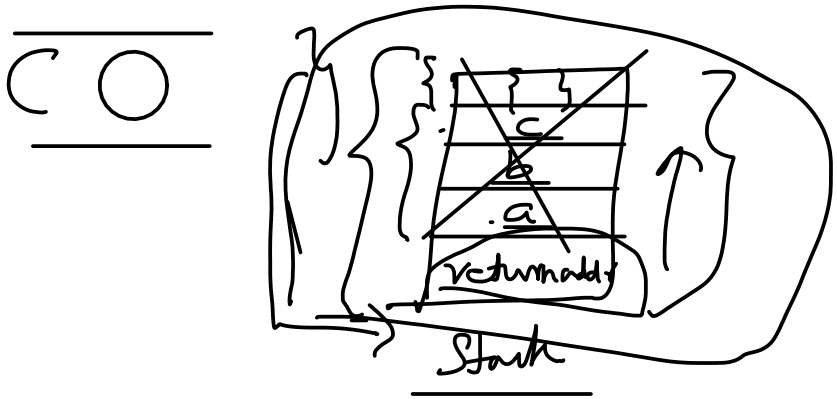
Program to Process



C program

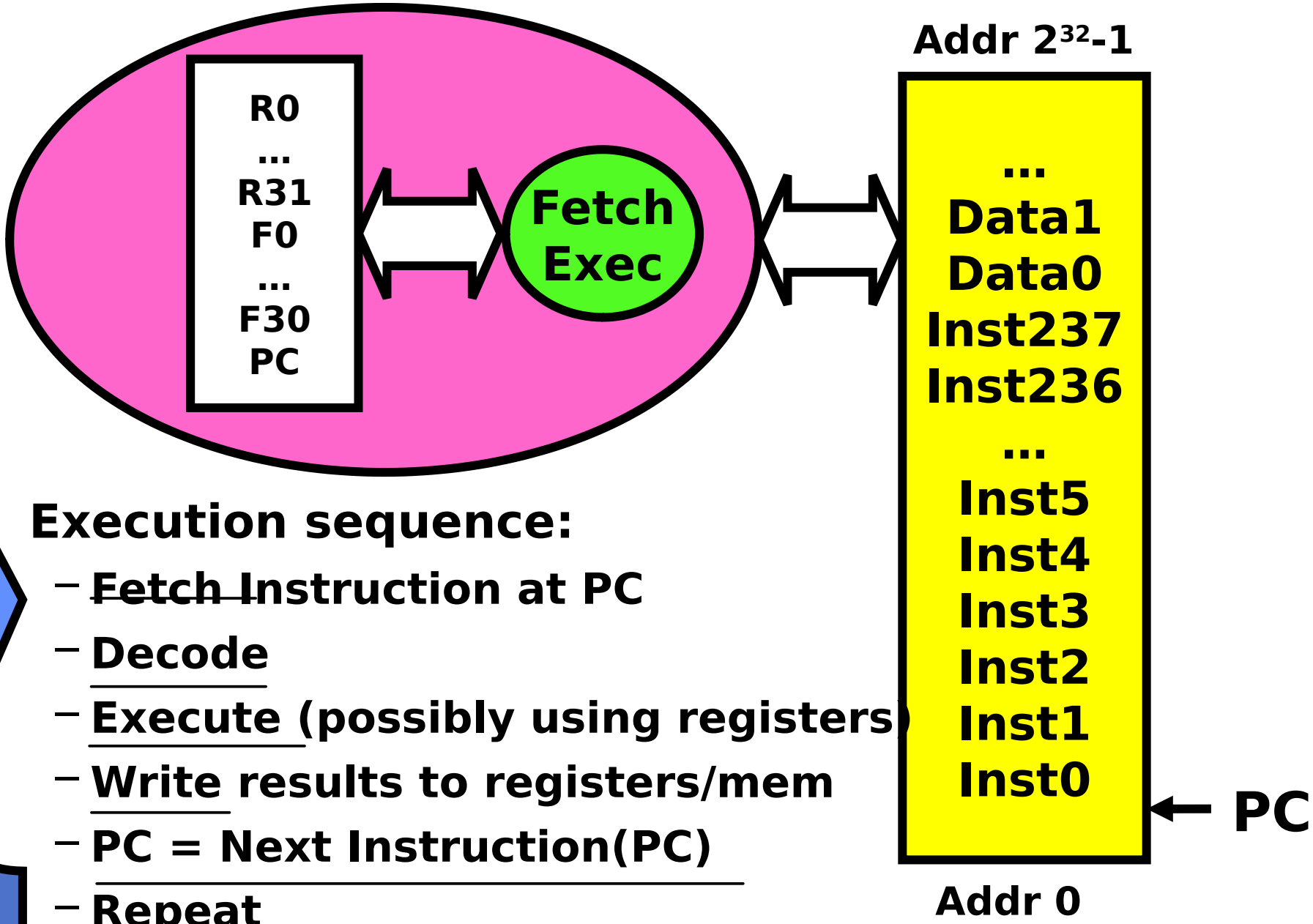
```
#include <stdio.h>

int main(int argc, char const *argv[])

{

    printf("Hello world\n");

    return 0;

}
```





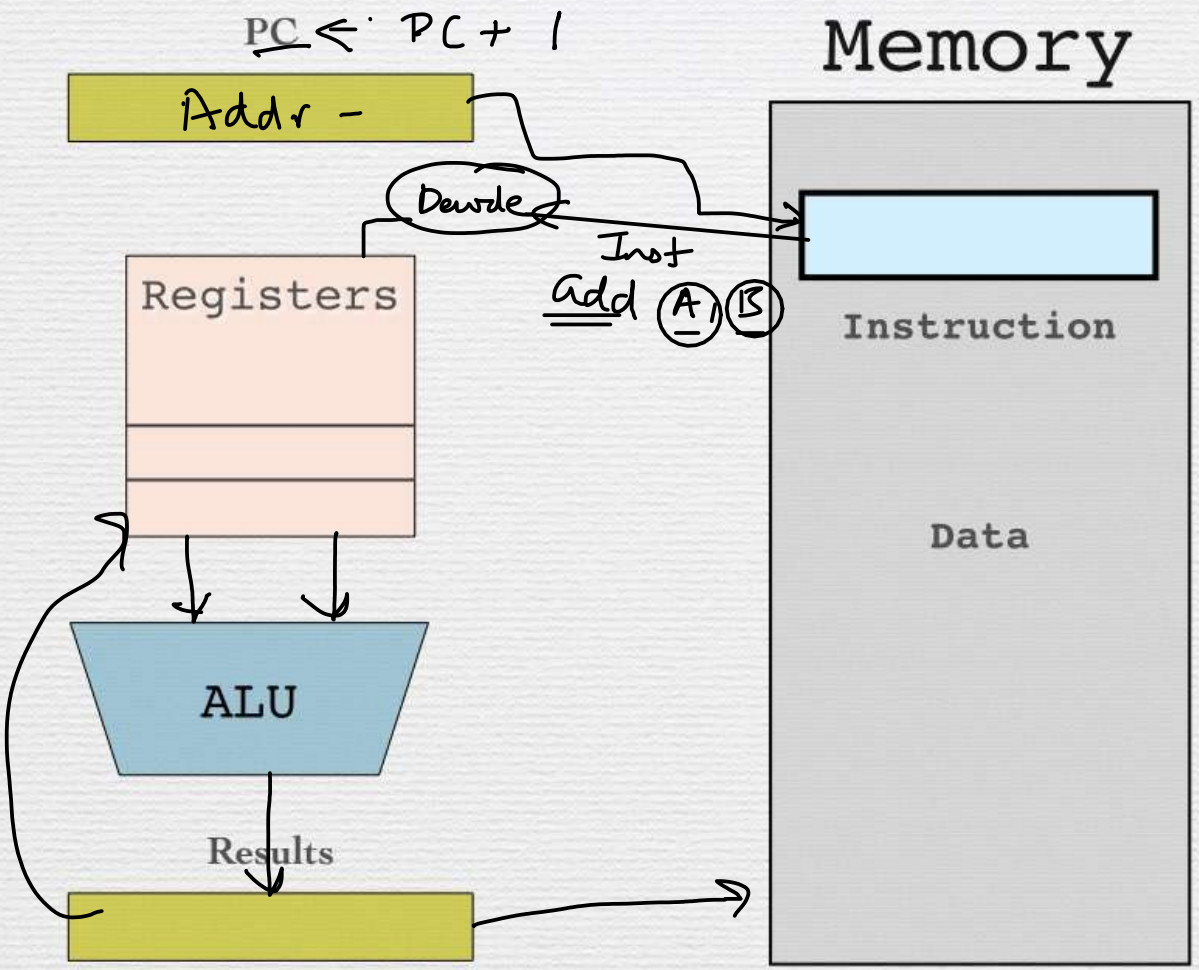
objdump -sRrd a.out

What happens during program execution?



- **Execution sequence:**
  - ~~Fetch~~ Instruction at PC
  - Decode
  - Execute (possibly using registers)
  - Write results to registers/mem
  - PC = Next Instruction(PC)
  - Repeat

Adapted from lecture slide 2.13 of CS162 by Prof. John Kubiatowicz, Copyright 2020 UCB.


PC ← PC + 1
Memory
Addr
Decode
Inst
Add A, B
Registers
Instruction
Data
ALU
Results

First OS Concept: Thread of Control

Process

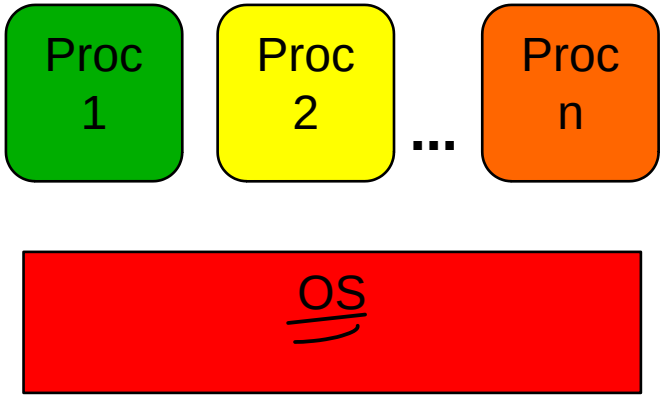
- **Thread: Single unique execution context**
  - **Program Counter, Registers, Execution Flags, Stack, State in memory for that thread**

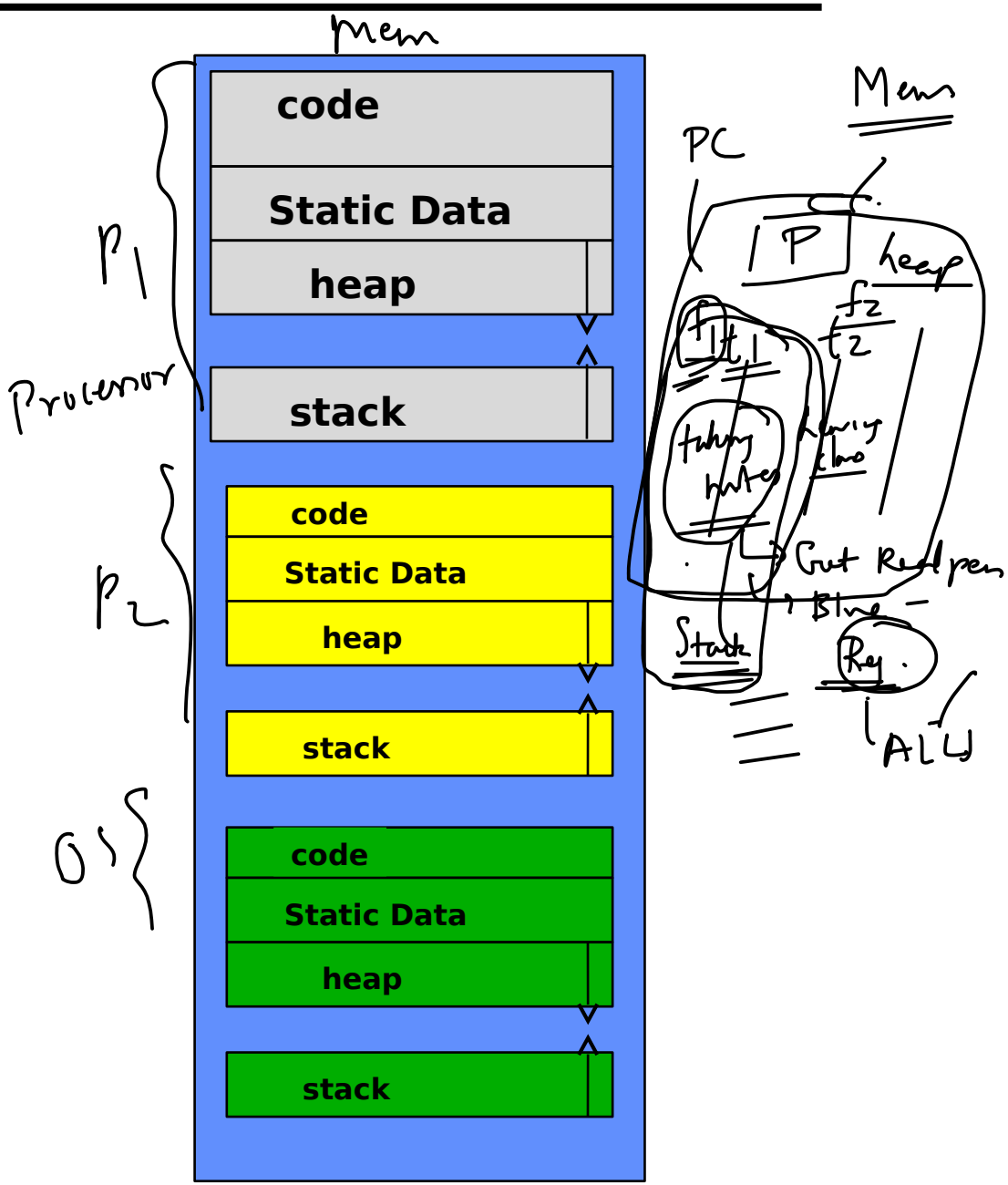
- **PC: holds the address of executing instruction in the thread.**

- **Certain registers hold the *context* of thread**
  - **Stack pointer, Heap Pointer, Data**

- **Registers hold the root state of the thread.**
  - **The rest is "in memory"**

Multiprogramming - Multiple Threads of Control

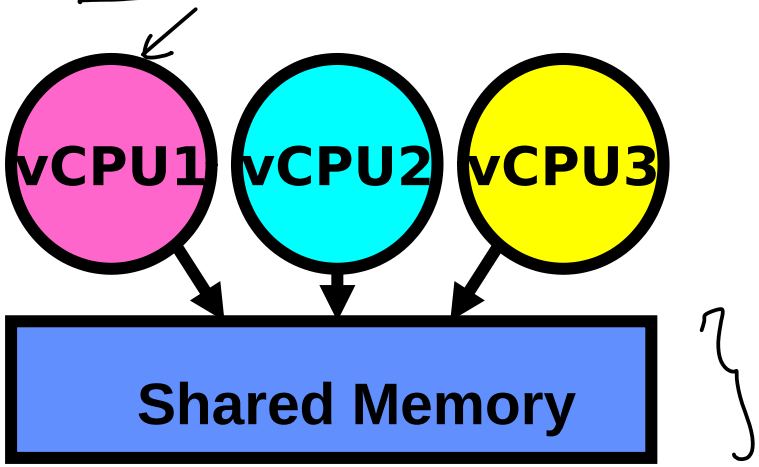


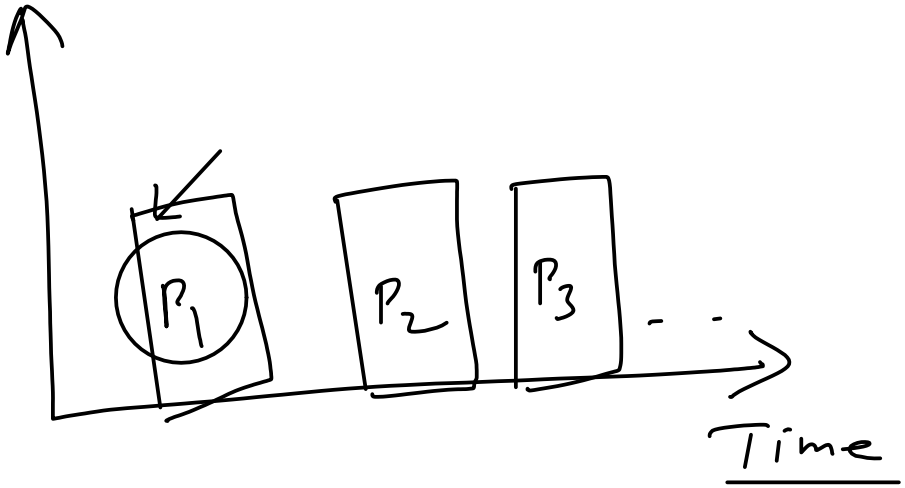


pstree -p

Virtualization of Resources

- **Virtual CPU**



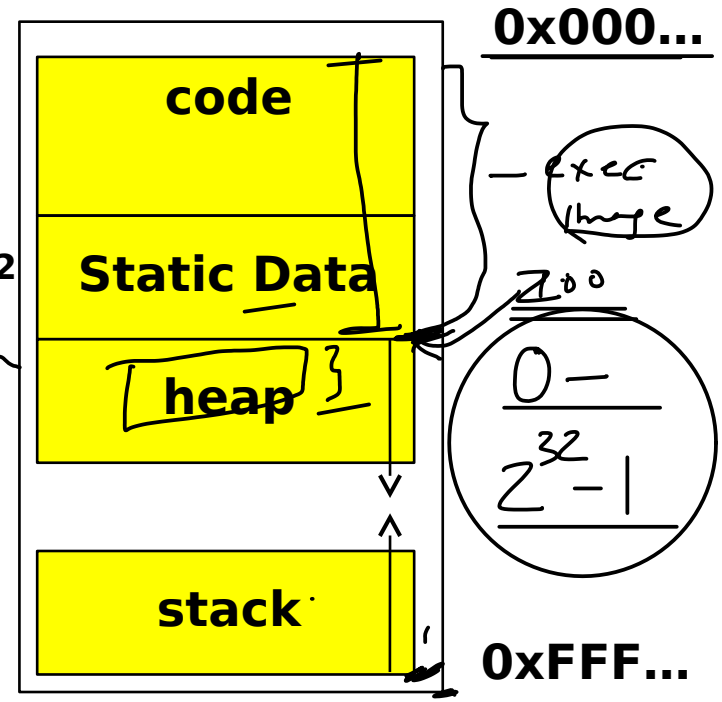


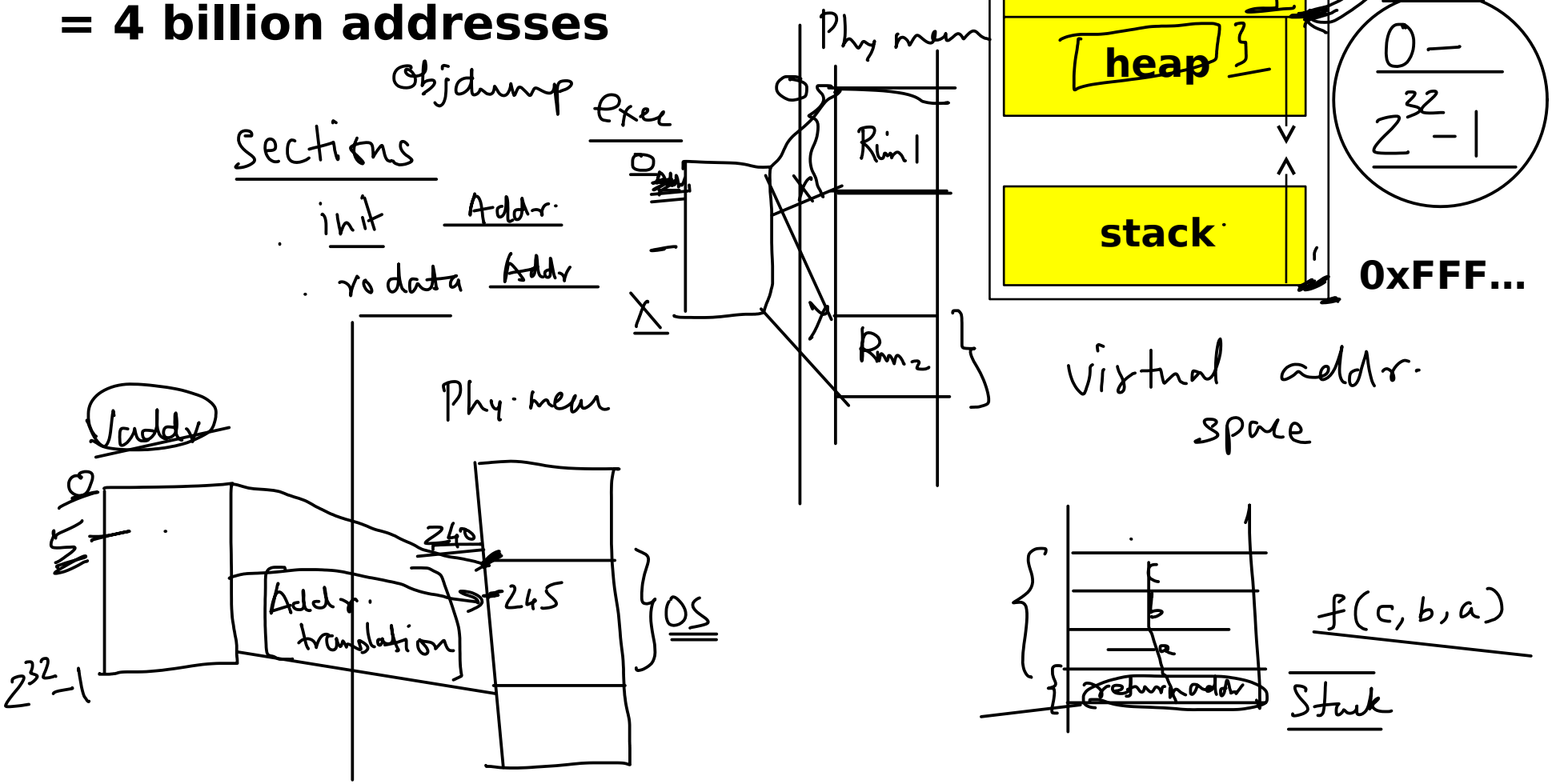
Recap Sept 18

- Program → Process
- Thread → unique single execution context
  PC, Registers, Exec. flags.

Second OS Concept: Program's Address Space

- **Address space ⇒ the set of accessible addresses + state associated with them:**
  - **For a 32-bit processor there are 2^{32} = 4 billion addresses**





Demos

- **Virtualization of CPU and memory**
- **Concurrency issues lead to <u>non-reproducible</u> and <u>non-deterministic output</u>**

T_1

Virtualization

Concurrency

OS: 3 easy pieces

Sys

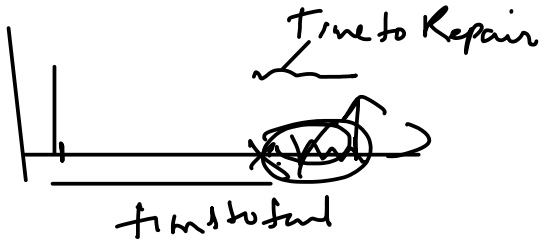
T_1	T_2
Read (x) 100	
	Read (x)
	200
X = X + 100	
200	
	X = X + 10
	210
Write (x)	
200	
	Write (x)
	210

How to evaluate an Operating System?

- **Reliability** — correct — write 3 — 2 to 50 million lines —
- **Availability** — crashing few sec. MTTR↓ MTTF↑
- **Adoption** —
- **Security and Privacy** · unauthorized
- **Portability** — Newer Apps } — API
- **Performance**

H/w Abstraction layer

fast efficiency



Summary

- This course covers concepts from OS and Networks.
- OS is a layer of software that manages computer resources for its users and applications.
- Evolution of OS: IO routines, Batch processing, Multi-programming, Interactive processing …
- Roles played by OS: referee, illusionist and common services
- Two concepts: threads and address space
- Metrics for evaluating the OS