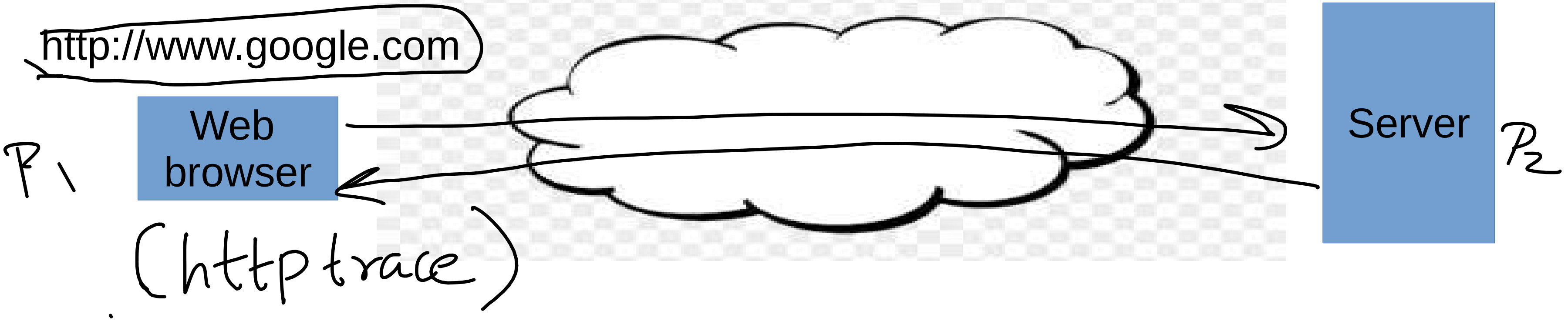
Introduction to Communicating Distributed Processes
Lecture 1

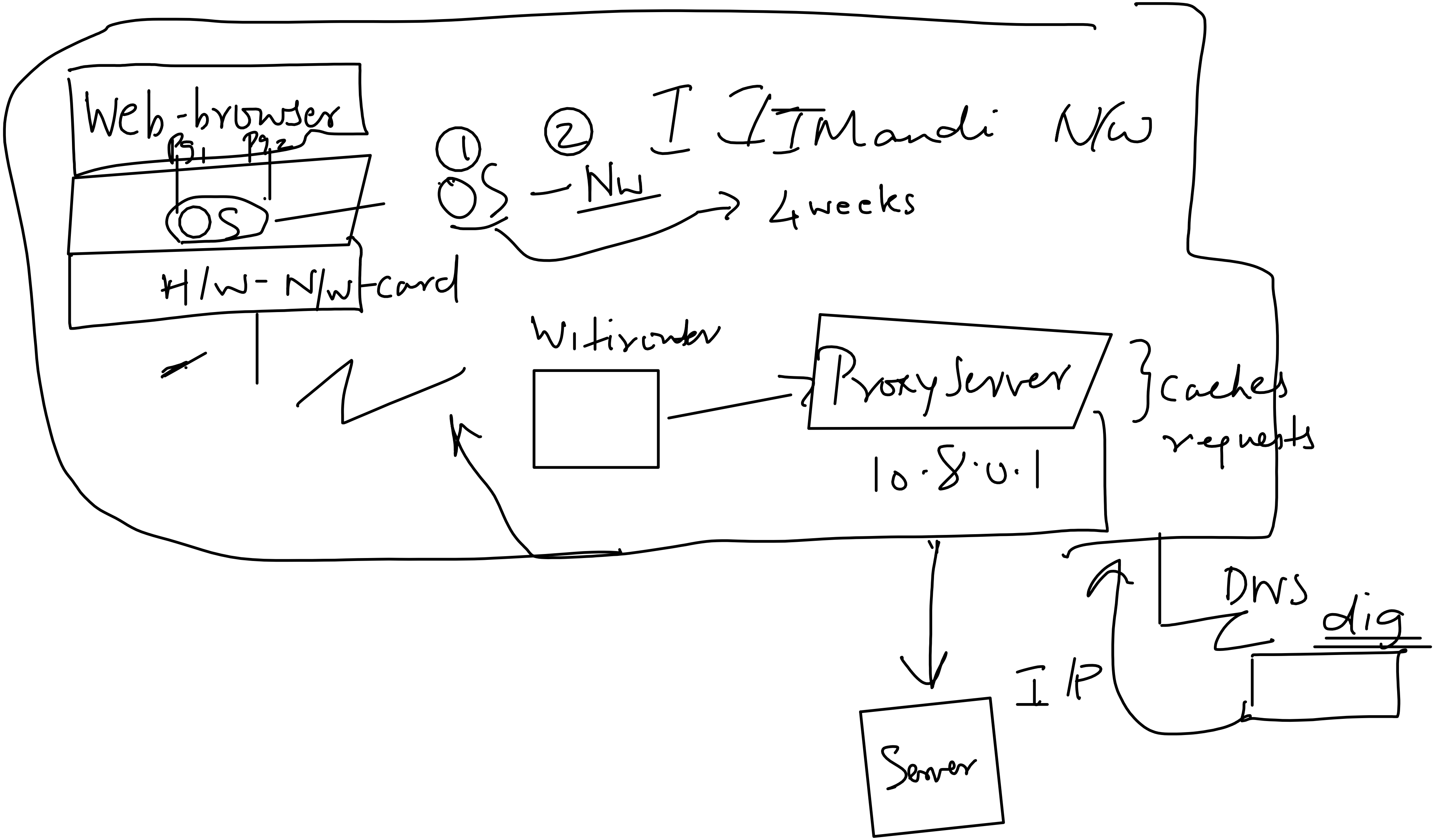
Motivating Example

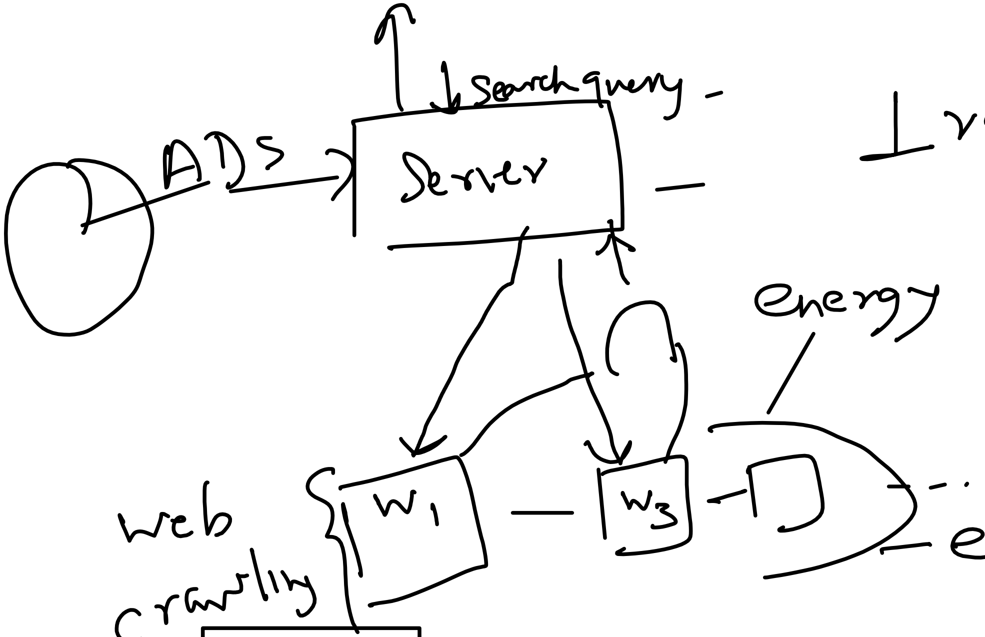


Get - App^n layer protocol

- 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
②
IIT Mandi N/W
OS — Nw
4 weeks
Wifirouter
Proxy Server
} caches requests
10.8.0.1
DNS
dig
IP
Server



1 req — 1000s req/s

Sequential

↓

multithreading

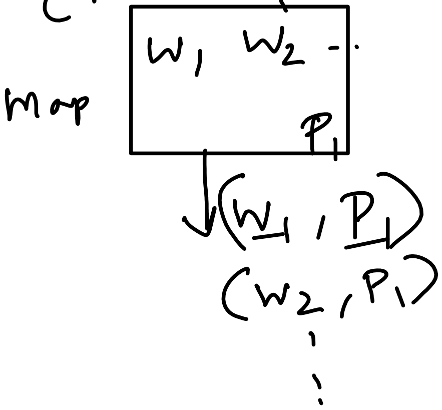
Concurrent

ACID

Threads

Google date

10000s -m/c

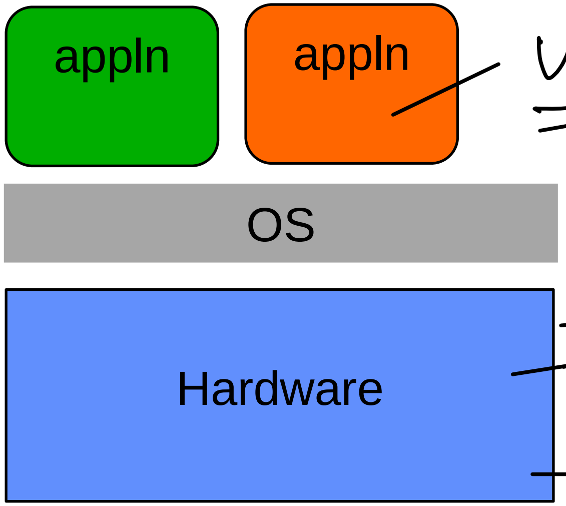


Index: word → $P_1, P_2, \dots P_n$

$W_1 \rightarrow P_1, P_2, P_- \; P_n$ — ∩ 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

4 core

1 Resource allocator → Gatekeeper

Referee

2 Illusionist - wifi - reliable

Overcomes limitation.

Time sharing

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’
```

OS

- Timer Interrupt
- Ctrl-C

segmentation fault

2^{32} - 4GB x 4GB

64-bit processor : 0 - $2^{64} - 1$

  - Communication -

- 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

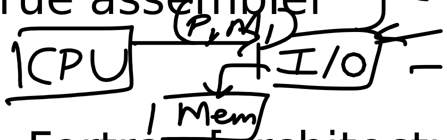
m/c bit-bit

CPU - idle

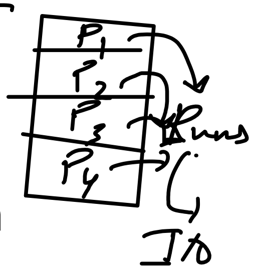
- 1945: ENIAC- loading programs / data was difficult and time-consuming

huge - expensive m/c | humans cheap

- 1949: EDVAC - Computers got memory
- 1949: BINAC - Programming language
- 1951: UNIVAC – Reusable code fn.
- 1952: Shared IO routines and True assembler } error-handling
- 1956: Interrupts



– DMA transfer



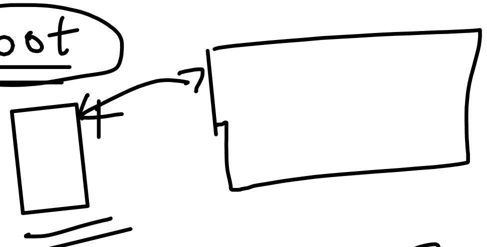
- 1954 – 1957: Batch processing, Fortran [architecture independent, high level language]

– Improve utilization

- 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 – Assembly lang
- 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

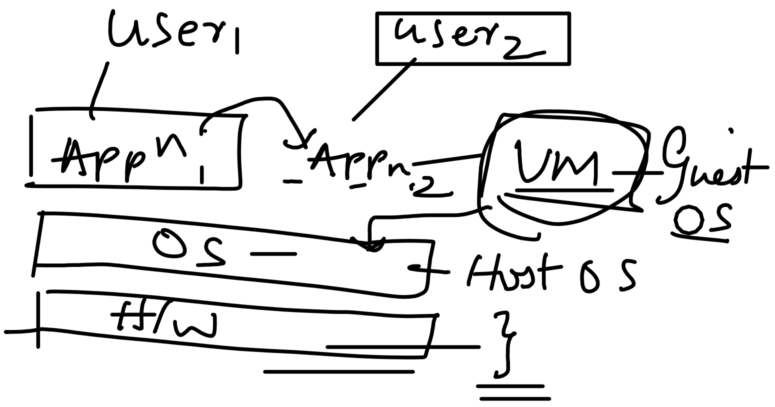
Where is it headed?

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

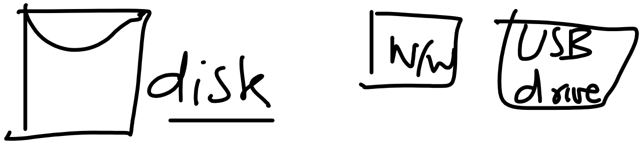
Recap: Sept. 17, 2020

OS plays 3 roles

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



byte-addressable



blocks —

Program to Process

helloworld.c

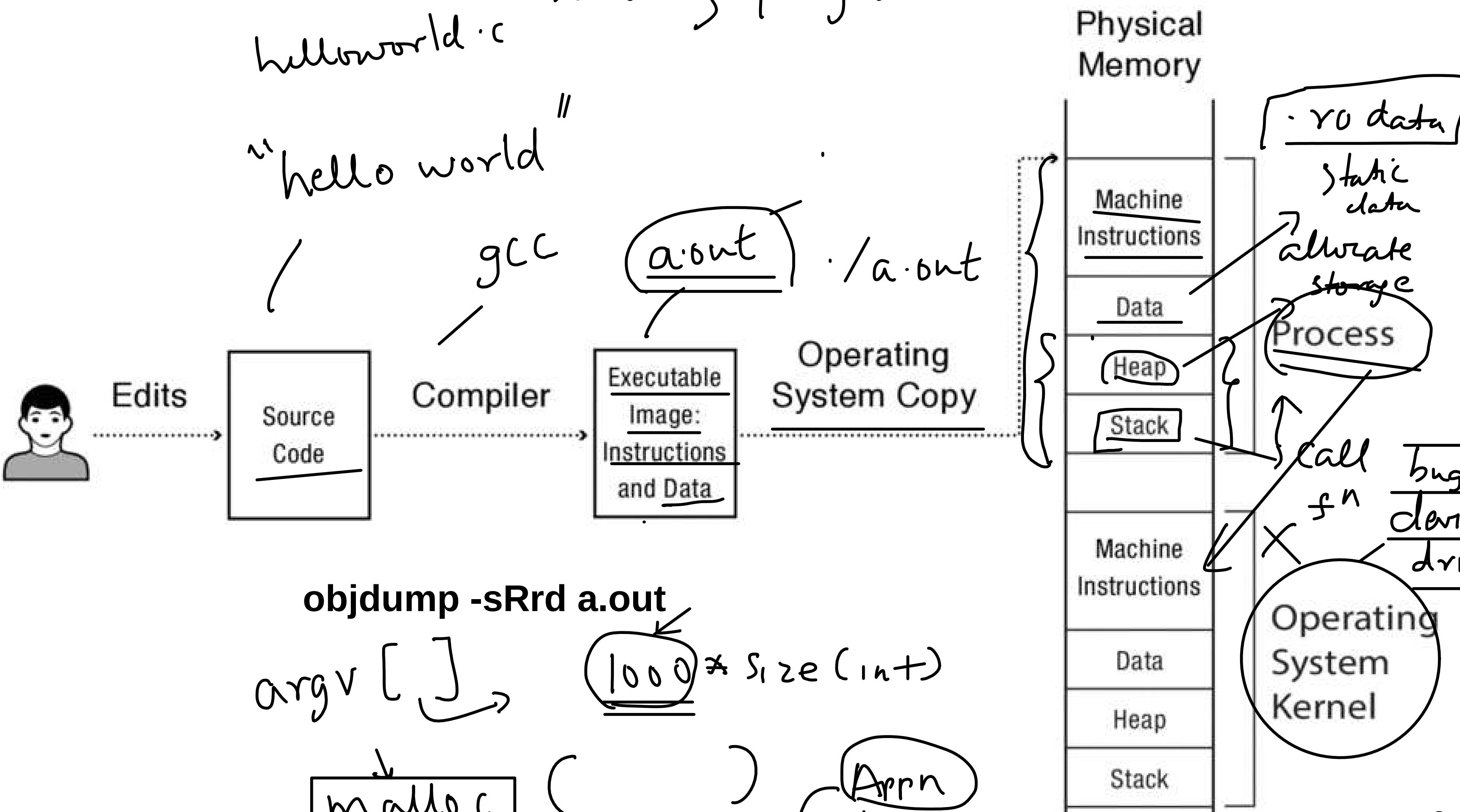
running program

"hello world"

gcc

a.out

./a.out



.ro data

static data

allocate storage

call fn

buggy device drivers

objdump -sRrd a.out

argv [ ]

1000 * size(int)

malloc ( )

Appn

OS

H/W

while(1){

}

C program

```
#include <stdio.h>
int main(int argc, char const *argv[])
{
    printf("Hello world\n");
    return 0;
}
```

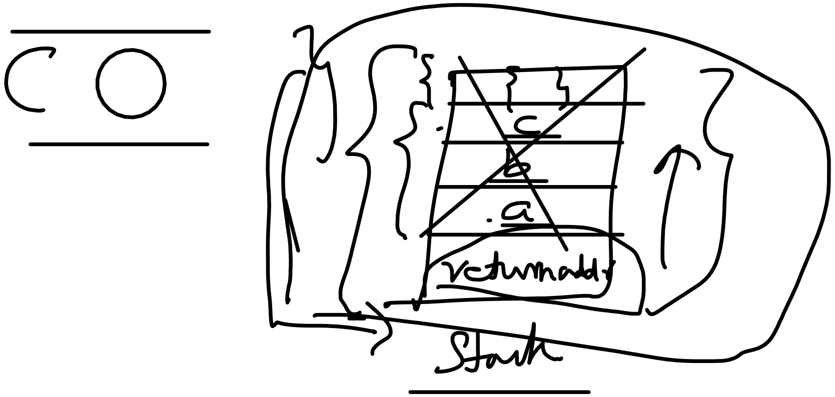
func(

addr.

fun(a b c)

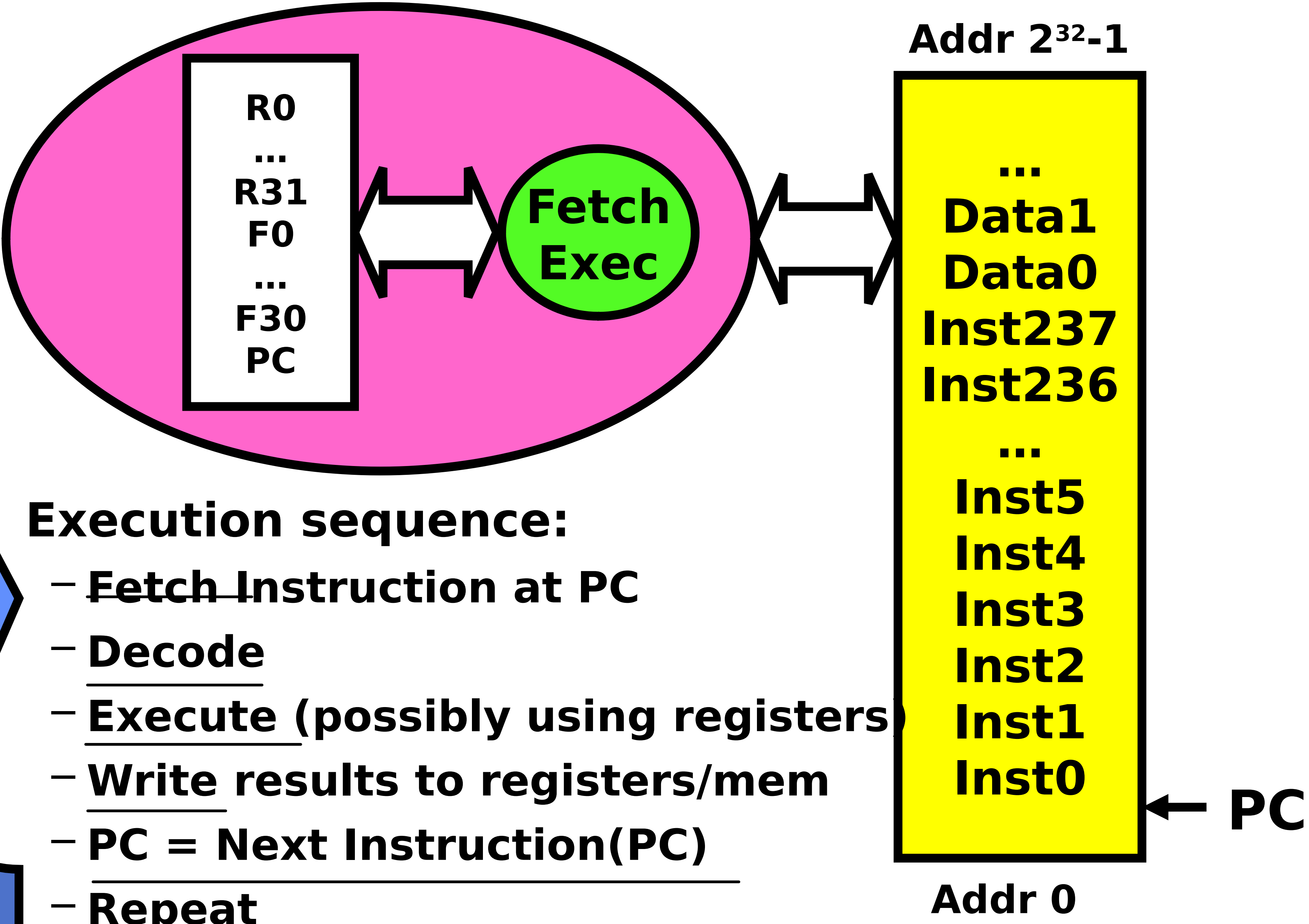
return addr

int d; local var.



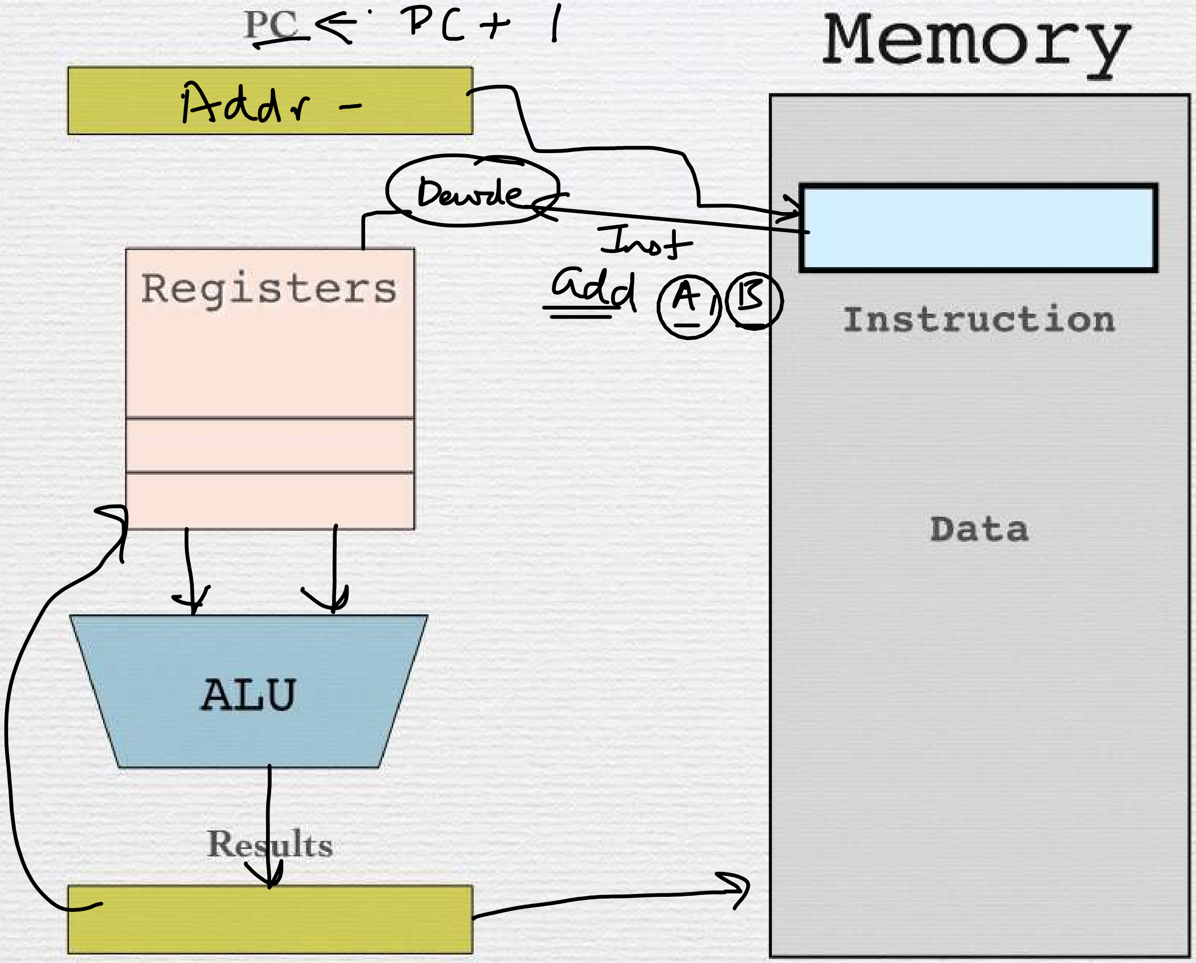
```
objdump -sRrd a.out
```

What happens during program execution?



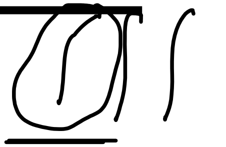
- Execution sequence:
  - Fetch Instruction at PC
  - Decode
  - Execute (possibly using registers)
  - Write results to registers/mem
  - PC = Next Instruction(PC)
  - Repeat

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


PC <- PC + 1
Addr -
Decode
Inst
add A, B
Memory
Registers
Instruction
Data
ALU
Results

First OS Concept: Thread of Control

Process

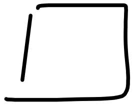


- **Thread: Single unique execution context**
  - **Program Counter, Registers, Execution Flags, Stack, State in memory for that thread**

VR1, VR2 . .

- **PC: holds the address of executing instruction in the thread.**
- **Certain registers hold the *context* of thread**
  - **Stack pointer,Heap Pointer, Data**

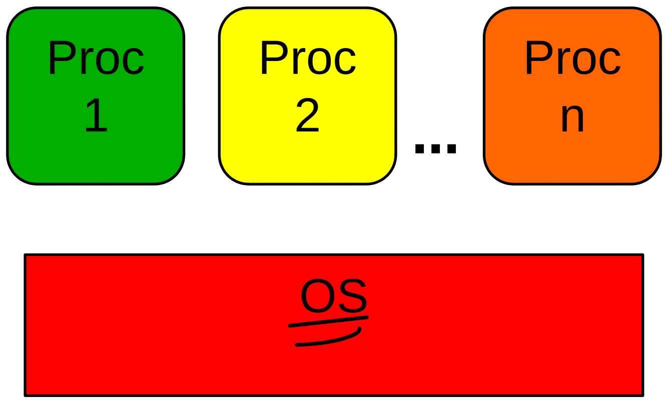
CS - .

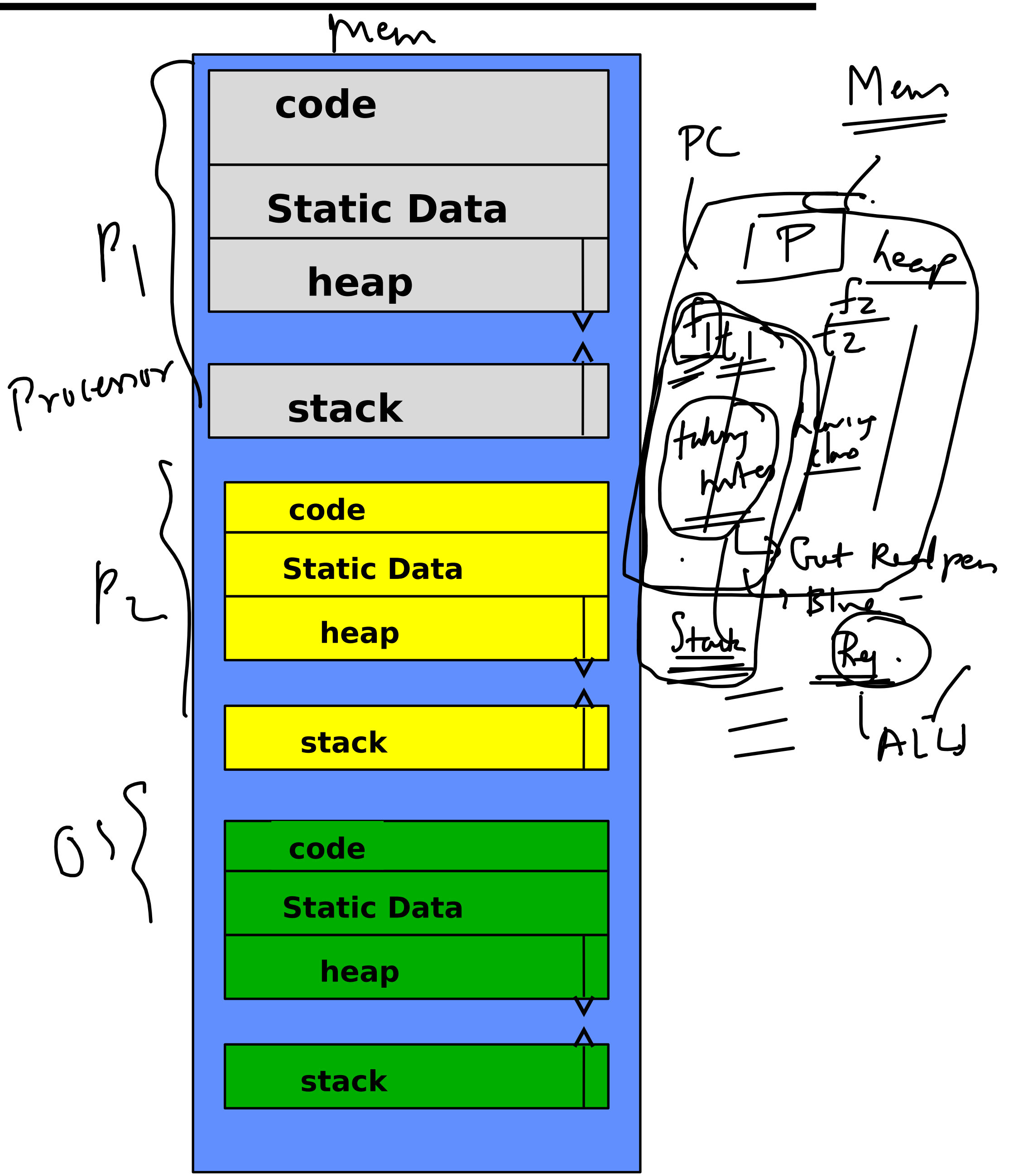


- **Registers hold the root state of the thread.**
  - **The rest is "in memory"**

Adapted from lecture slide 2.12 of CS162 by Prof. John Kubiatowicz, Copyright 2020 UCB.

Multiprogramming - Multiple Threads of Control



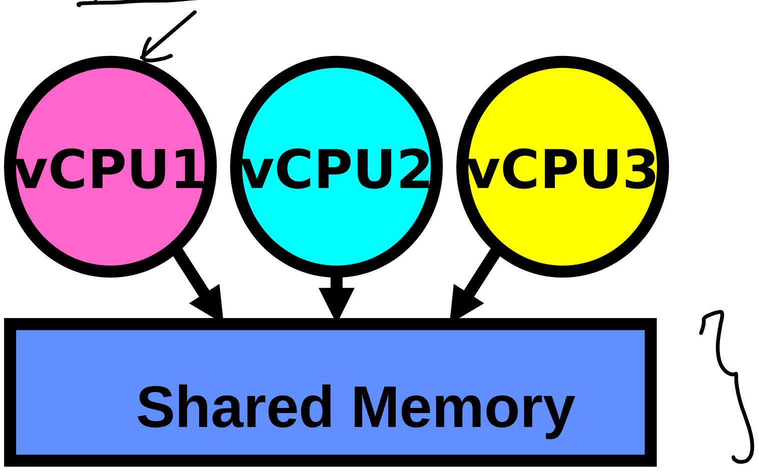


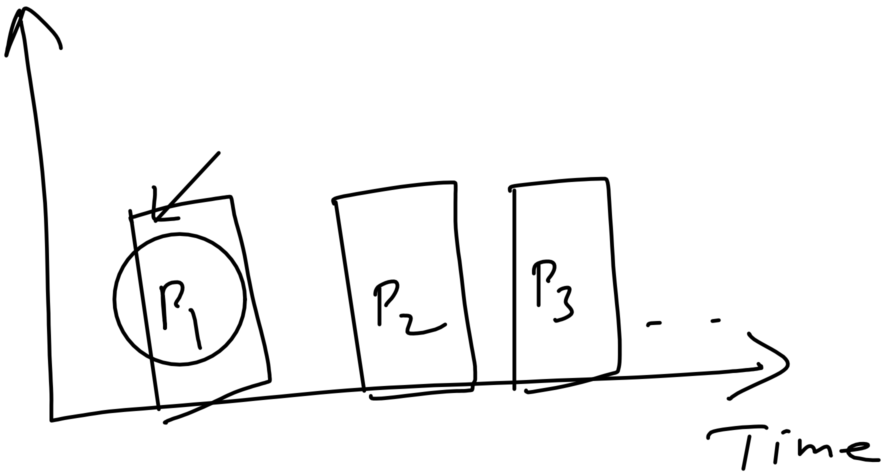
pstree -p

Adapted from lecture slide 2.17 of CS162 by Prof. John Kubiatowicz, Copyright 2020 UCB.

Virtualization of Resources

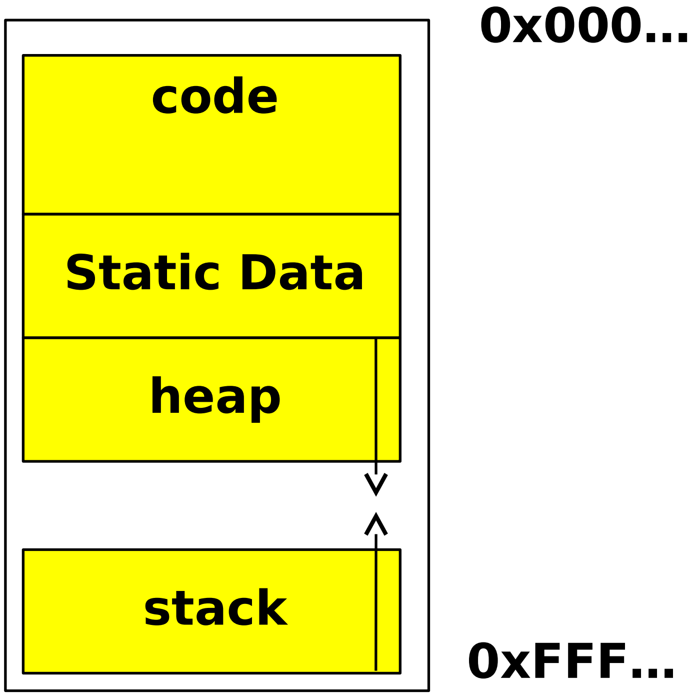
- **Virtual CPU**





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



Adapted from lecture slide 2.21 of CS162 by Prof. John Kubiatowicz, Copyright 2020 UCB.

Demos

- **Virtualization of CPU and memory**
- **Concurrency issues lead to non-reproducible and non-deterministic output**

How to evaluate an Operating System?

- **Reliability**
- **Availability**
- **Adoption**
- **Security and Privacy**
- **Portability**
- **Performance**

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