CS 3650

Spring 2024 lecture 1

1/9/24

Prof. Peter Desnoyers

Me room 334 WVH (background)

what the course is about

computer systems: opening the hood & voiding warranty

mental model of computer system
- CPU
- memory
- storage

~ fiddly bits

Outline of class

- Programs, processes, operating systems ← abstractions
- C programming, debugging & testing
- basic networking
- concurrency - threads, synchronization, locking
- virtual memory & memory management
- I/O devices
- block storage, file systems

Course logistics

office hours - will be posted

midterm - week 2/26-3/1

homework, labs

communication:

→ Piazza

→ github classroom

→ gradescope

→ canvas

canvas

→ piazza

→ web page

40% exams

20% homeworks

40% labs

Academic integrity

→ do your own work

Computing

→ local VMs

the `ls` demo

high level: ls -l /etc

one level down:

lower:

keyboard driver

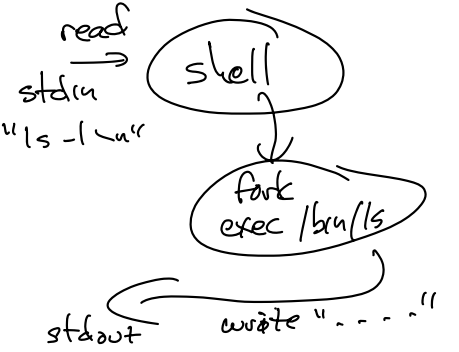
write → frame buffer

virtualized

hypervisor, emulated real devices

on top OS

on real real devices



how does fork work?

" " exec work?

what's a process?

what's in the /bin/ls program

⋮

how did we get /bin/ls off the ~~disk~~ drive?

→ file system

→ block device

→ virtual machine

→ real versions

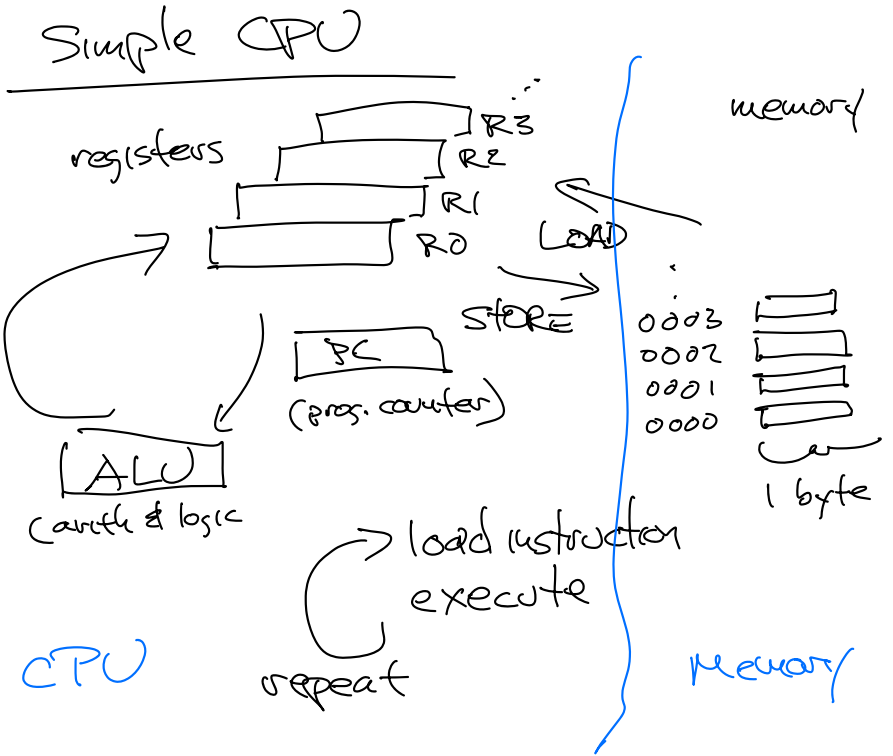
Applications / areas of programming

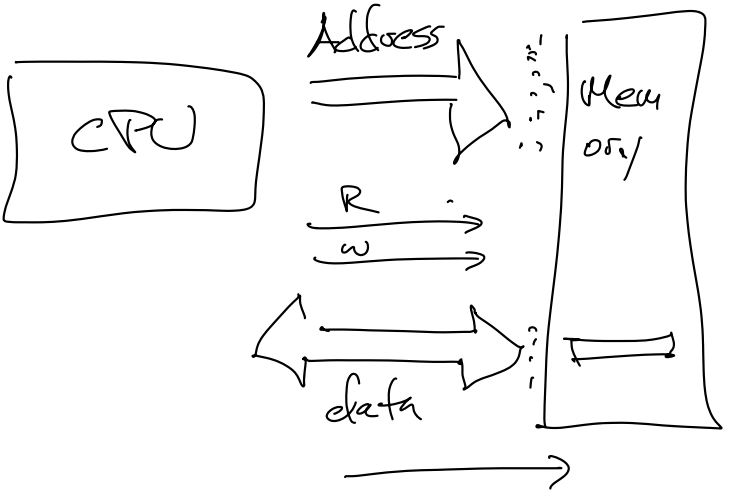
"classic" - UI, screen, keyboard / mouse / ...

"embedded" - computers we don't see

OS / system - underneath the classic app
runtime

"cloud" - virt. machines
containers
deployment / orchestration


Simple CPU
registers
R3
R2
R1
R0
memory
LOAD
STORE
PC
(prog. counter)
ALU
(arith & logic
0003
0002
0001
0000
1 byte
load instruction
execute
repeat
CPU
Memory


CPU
Address
Mem
ory
R
W
data

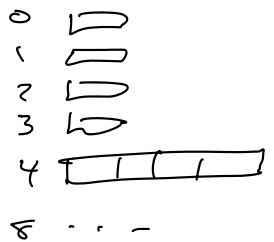
Complications:

- byte isn't big enough
- 2, 4, 8-byte integers

little-endian ←

big-endian

2049 = 0100000000001



Instructions:

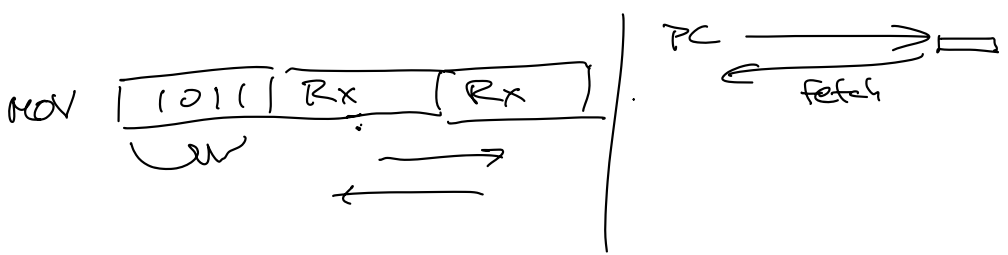
LOAD <addr> → RegX (read from memory)

STORE RegX → <addr> (write . . .

<op> RegX, RegY → RegZ

ADD, SUB, MUL, AND, OR, ...

→ set PC to next instruction



Control flow instructions

JMP <addr> → set PC to <addr>

<cond> JMP <addr>

004 JMP 1234
008 MOV R1->R2

1234 :