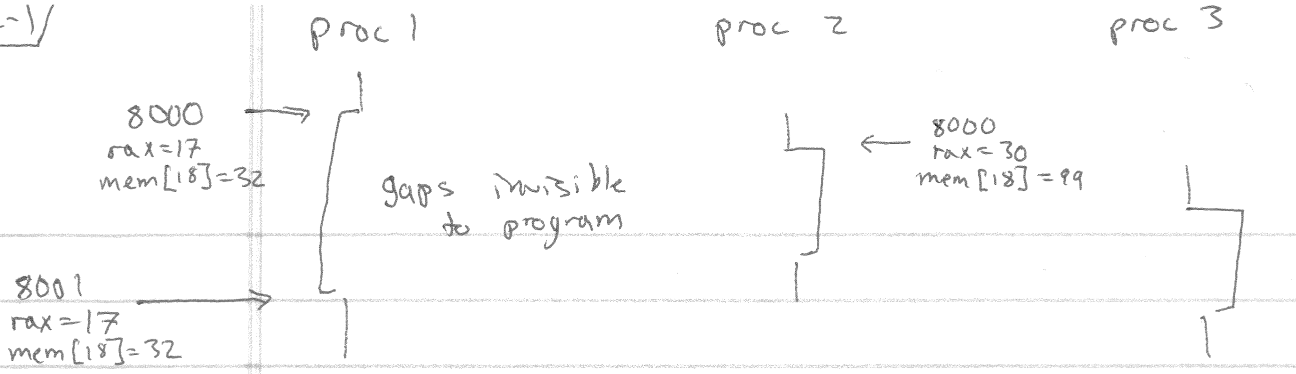
22-1/



time sharing

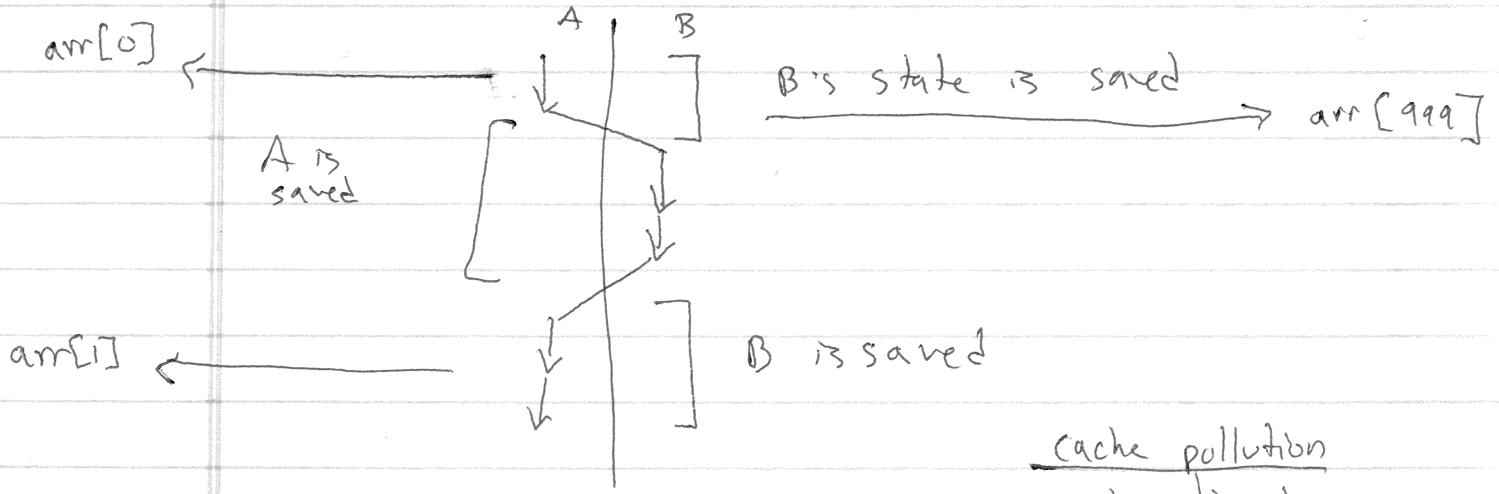
A program is some source (instructions)
A program instance runs
An instance runs "in the context" of a process
A process holds the state $(\underbrace{\text{PC, registers}}_{\text{cpu}}, \underbrace{\text{open files}}_{\text{os}})$

Preemption – can be interrupted

mulit-processing

interaction w/ outside exposes
↓ looking at time ↓ reading files ↓ use the network
if the gaps are visible — then this is a concurrent system
if not — parallel system

every time we switche processes (incld. w/ kernel)
the state saved and then restored
— a context switch —



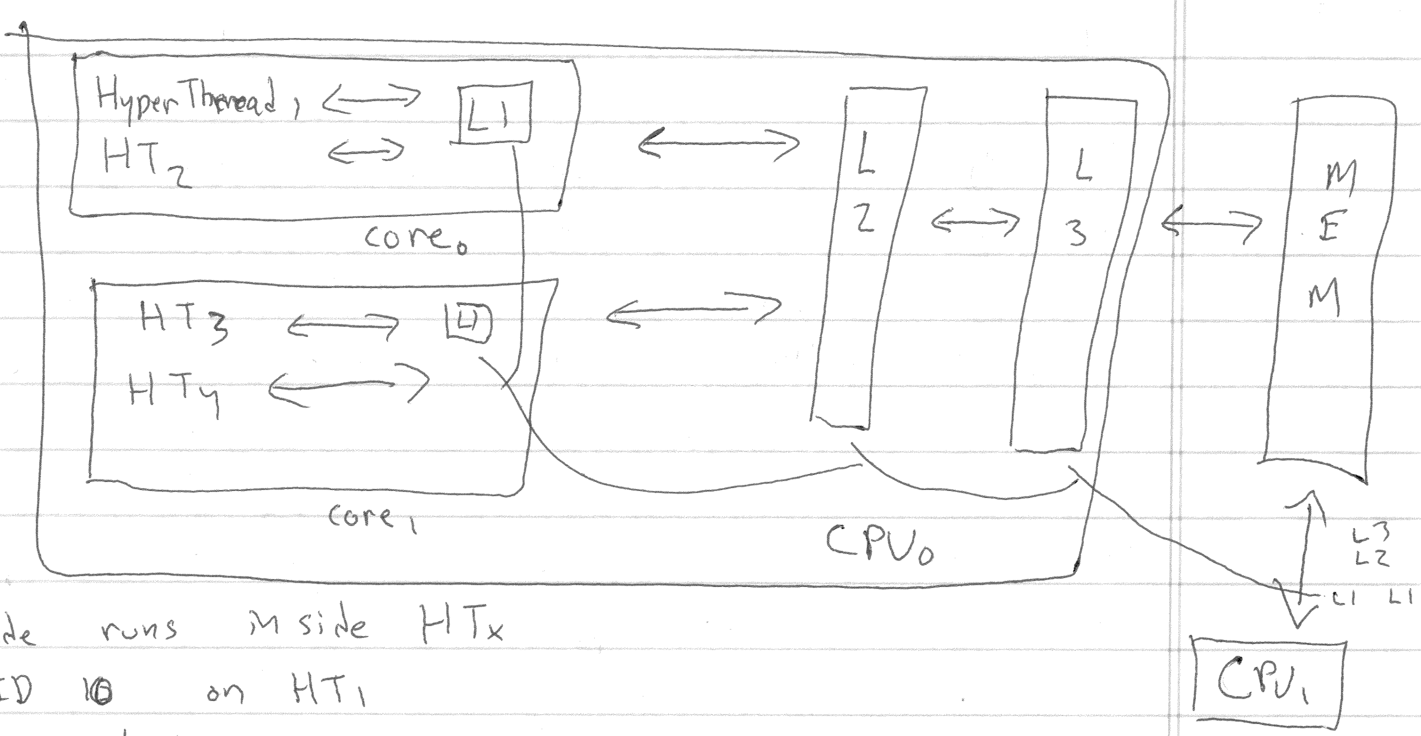
cache pollution
when time sharing
messes with caches

22-6

① $$\begin{tikzcd} \boxed{\text{CPU}} \arrow[r, leftrightarrow] & \boxed{\text{mem}} \end{tikzcd}$$

② $$\begin{tikzcd} \text{CPU} \arrow[r, leftrightarrow] & \text{L1} \arrow[r, leftrightarrow] & \text{L2} \arrow[r, leftrightarrow] & \text{L3} \arrow[r, leftrightarrow] & \text{MEM} \end{tikzcd}$$

③



Code runs inside HT_x

PID 10 on HT_1

pause pid 10

restart pid 10 on whatever HT has pid 10's data

$HT_{1 or 2}$ > $HT_{3 or 4}$ > $HT_{5,6,7,8}$

---

reads mem 1000 — writes mem 1000

HT_1 — HT_8

22-3

$$\begin{tikzcd} \text{Kernel} \arrow[d] & & \text{Bios} \arrow[d] \\ \text{first user program (init)} & & \text{Boot loader} \arrow[llu] \end{tikzcd}$$

start process (path to the program)

⋮

pid = startproc ("/bin/ls");

start (path , args)

pid = start ("/bin/ls" , & args);

"/home/jay"

start (path, args , env)

PATH = /bin : /usr/bin

$$\begin{tikzcd} \text{parent} \arrow[d] \arrow[dr] & \\ \text{keep going} & \text{child} \end{tikzcd}$$

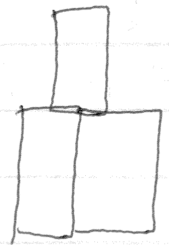
parent → childs is via args/env (byte strings)

child → parent is NONE except a pid

$$\begin{tikzcd} & \text{parent} \arrow[r, bend left] \arrow[dl] \arrow[d] & \text{wrote to 2000} \\ \text{parent A} \arrow[d] & \text{parent B} \arrow[d] & \\ \text{read 1000} \arrow[r, leftrightarrow, "\times"] & \text{write 1000} & \end{tikzcd}$$

parent A, parent B } — share everything (incl'd open files) (& pipes) except in different address spaces

both see 2000



−1 = fork() if parent is pid is Y
or a pid

if parent A gets pid X , then parent A's pid is Y
if parent B gets −1 , then it's pid is X

$$\begin{tikzcd} \arrow[d] \arrow[r] & X \text{ doesn't know } Y \\ Y \text{ knows } X & \end{tikzcd}$$

---

exec (path , args , env) — replaces the program & all memory EXCEPT open files

22-4/ Signals are interrepts at the process level

OS:
```
    while (1)
        select a proc to run
        if proc has a signal
            push PC
            jump proc mem [~~ean~~ sig table + signal no]
        else
            run ~~mormally~~
```
Program controls sigtable (sigaction)
send signals (pid, signo)

SIG - 1 - HUP (hang up) (Ctrl-D)
default: die
SIG - 9 - KILL (kill)
default: die (can't be replaced)

2 - INT (interrupt) ((Ctrl-C)
default: die (CAN be replaced)
17 - CHLD (one of your children died)