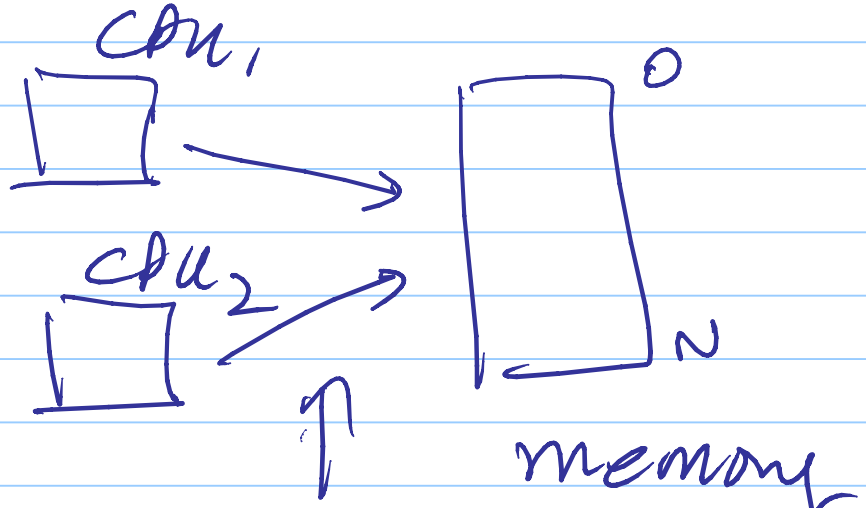
UMA – uniform mem access

↳ SMP – sym multi proc

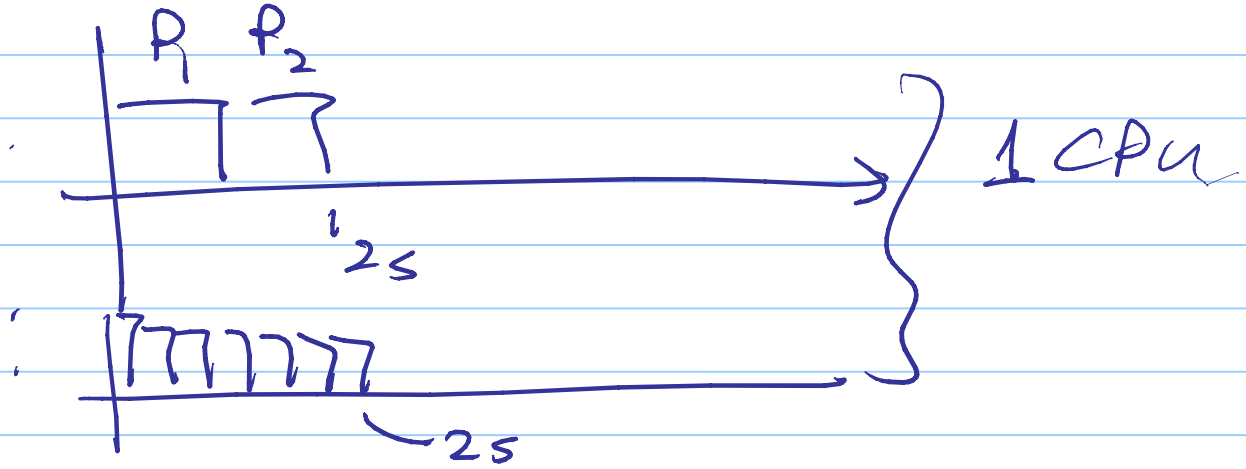
SMMP – shared mem multi proc

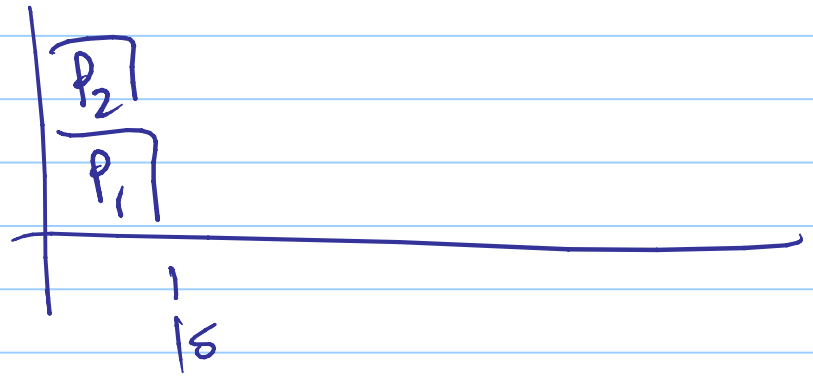
2 processors



2 processes

2 processes → each takes 1s of CPU



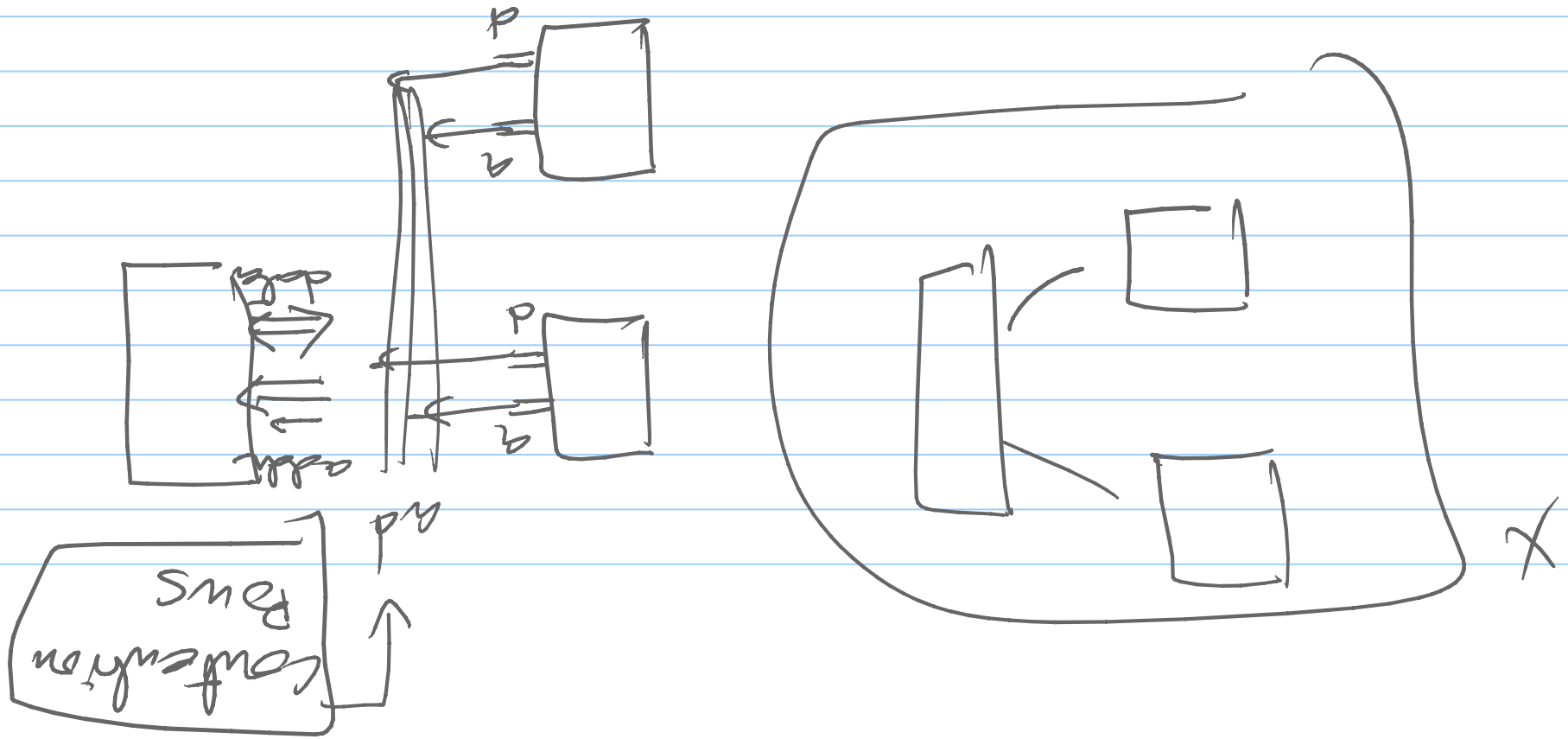


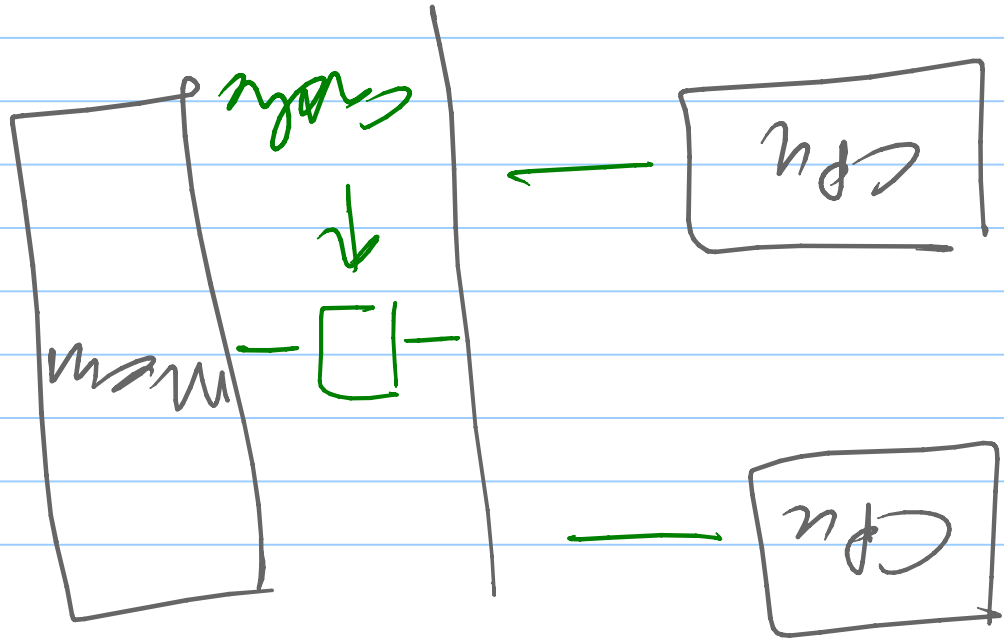
P1
P2
1s
con
1s
1s
1s
SS
1 phase
P1
P2
P1
P2
P1
P2
SS
SS
2 phase
P1
P2
P1
P2
P1
P2
1
2
1
2
1
Same

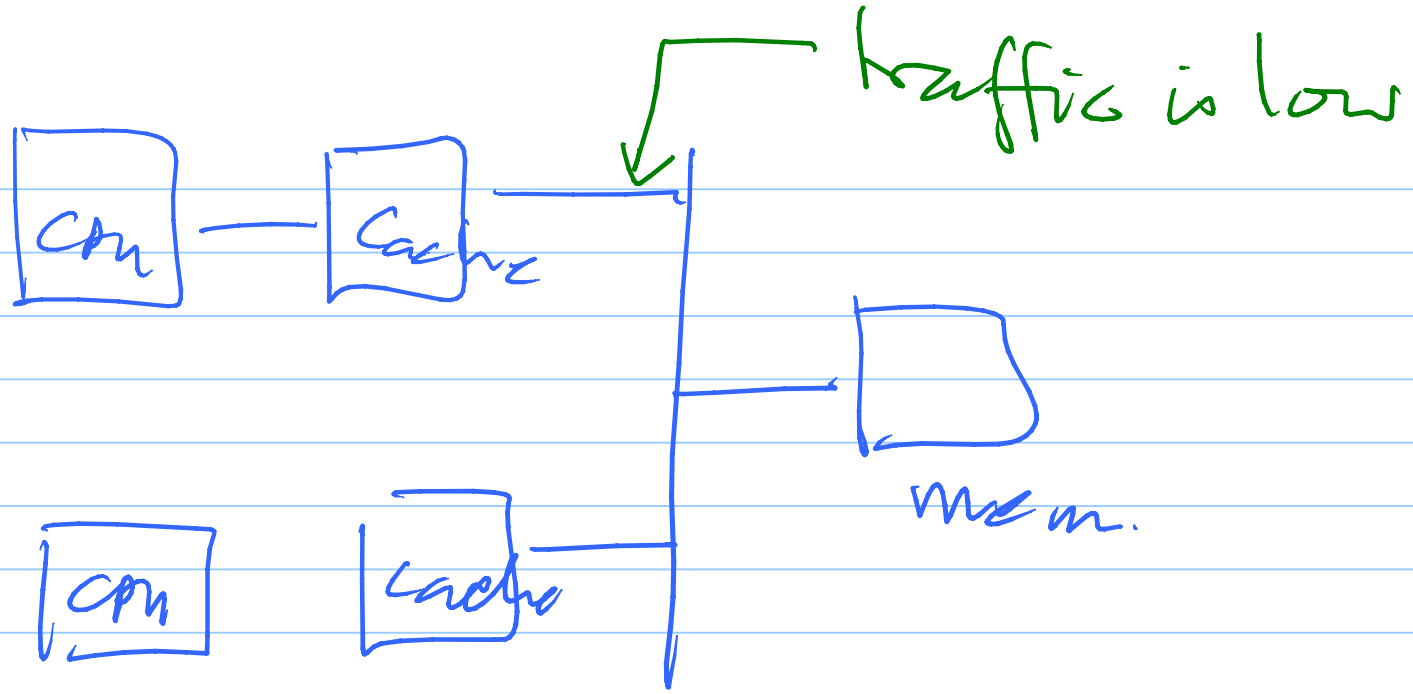
Single proc — 2 GHz

Multiple proc — 2 x 1 GHz

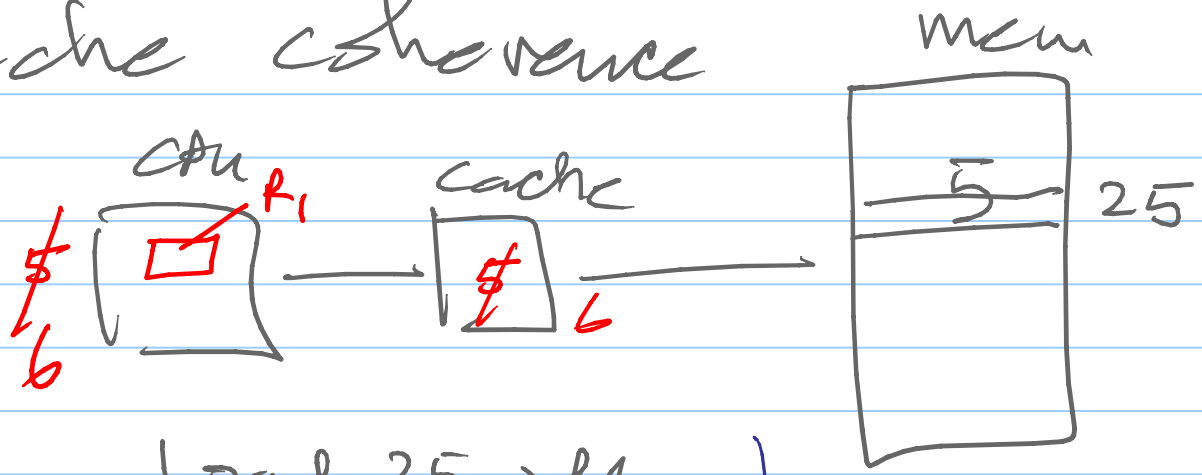
↳ slow on a single process.


P
b
P
b
Bus
Controller
X




traffic is low
CPU
Cache
CPU
Cache
mem.

Cache coherence



i = 5

i = 6

print i

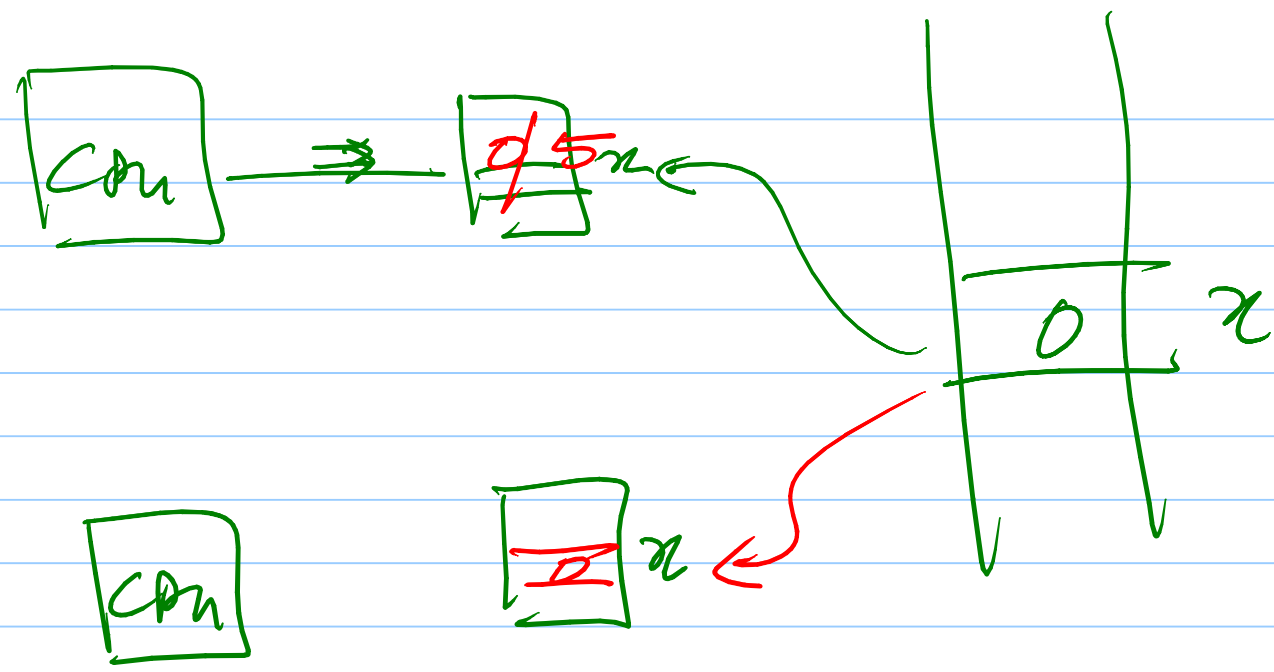
Load 25 → R1

add 1 to R1

sto R1 → 25

print @25 → 6

a read from mem returns the most recent write



Cpu
0 5 x
Cpu
0 x
0
x

Single copy semantics

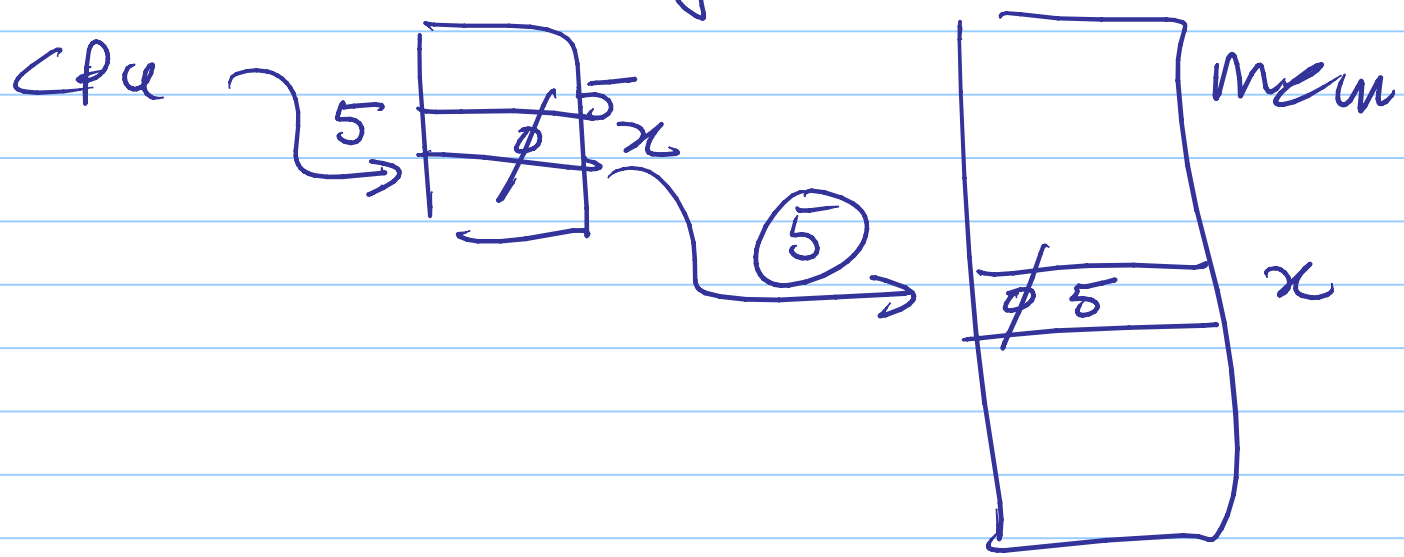
a.k.a coherence

"a read returns the most recent write"

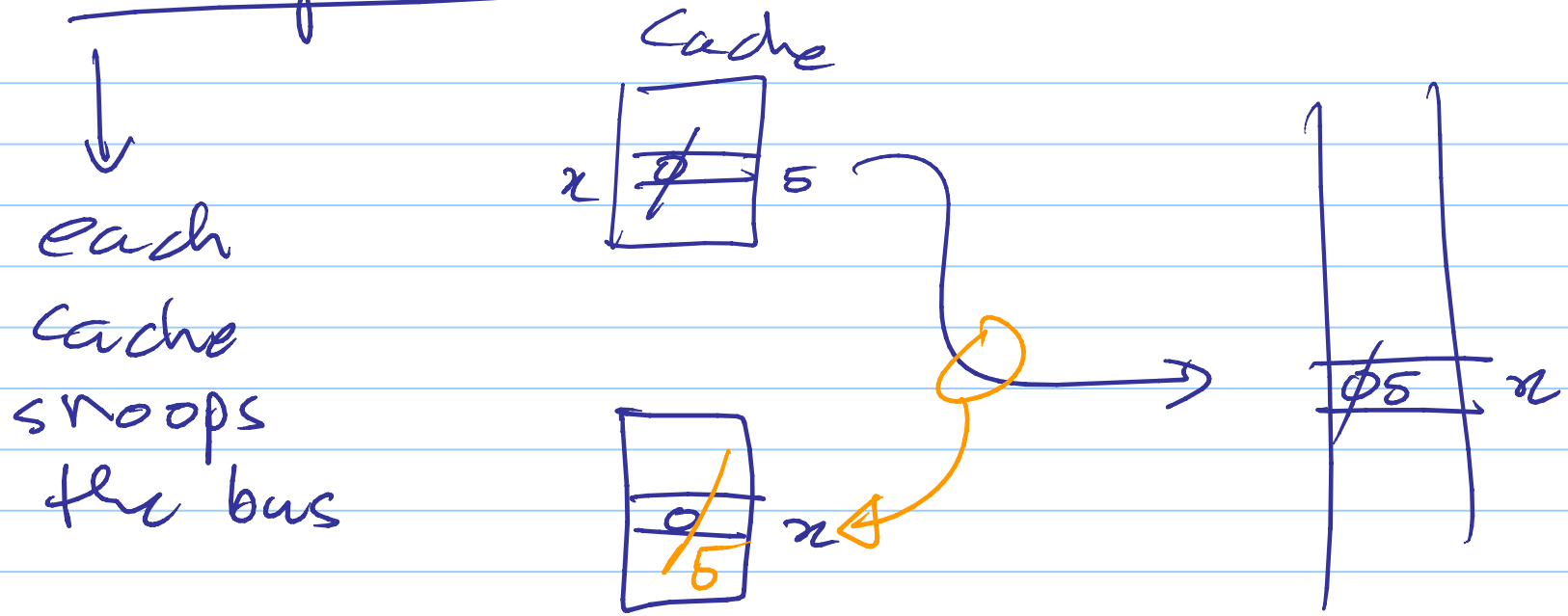
⇒ needed

Cache coherence

① use write through caches



Snoopy caches



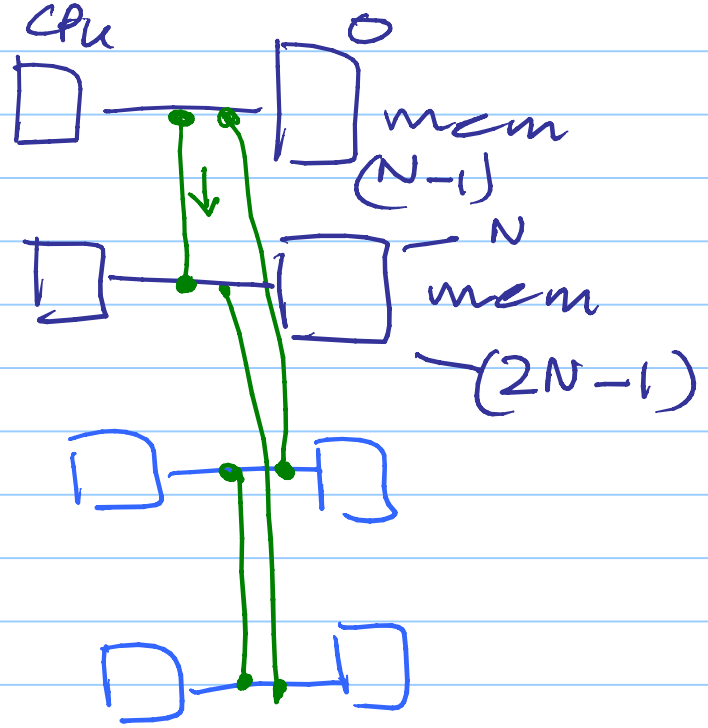
Shared mem multicores (UMA)

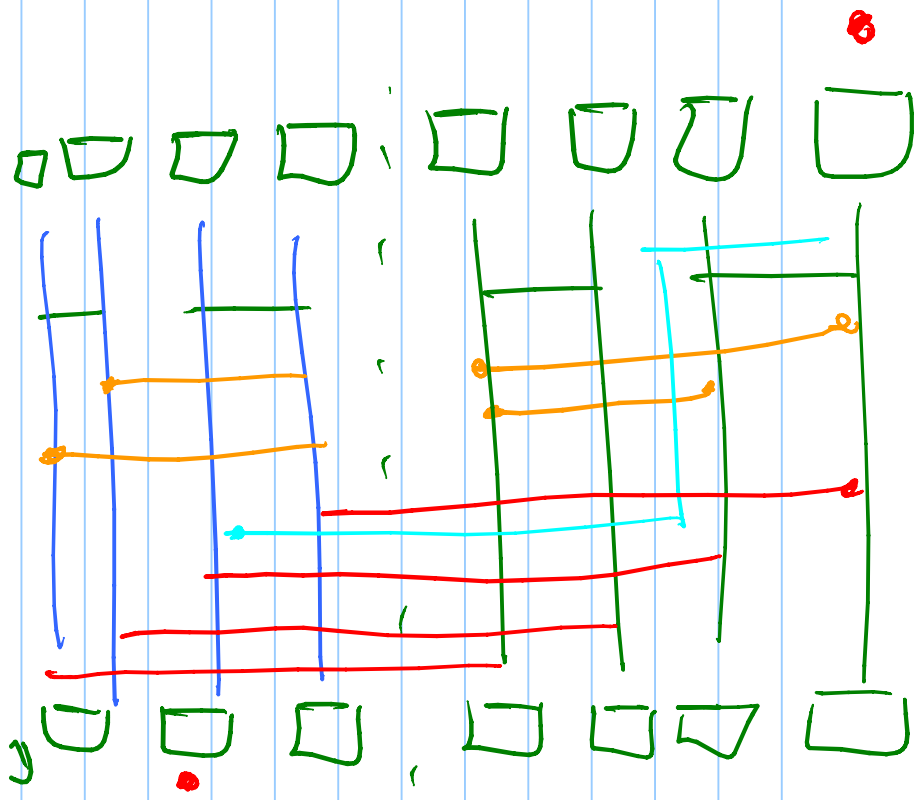
- has bottlenecks → addr bus traffic
- " hardware overhead
  - ↳ coherence

⇒ Does not scale!

How to scale?

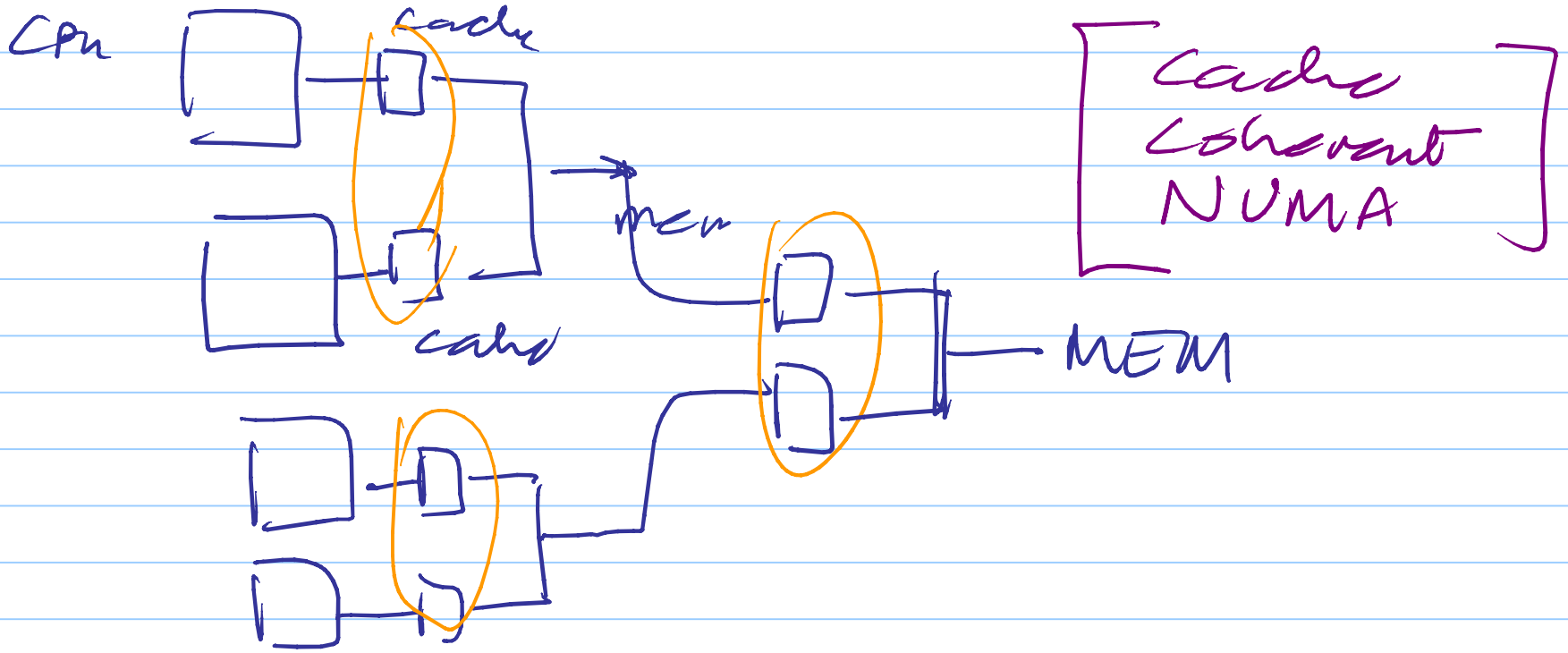
① NUMA





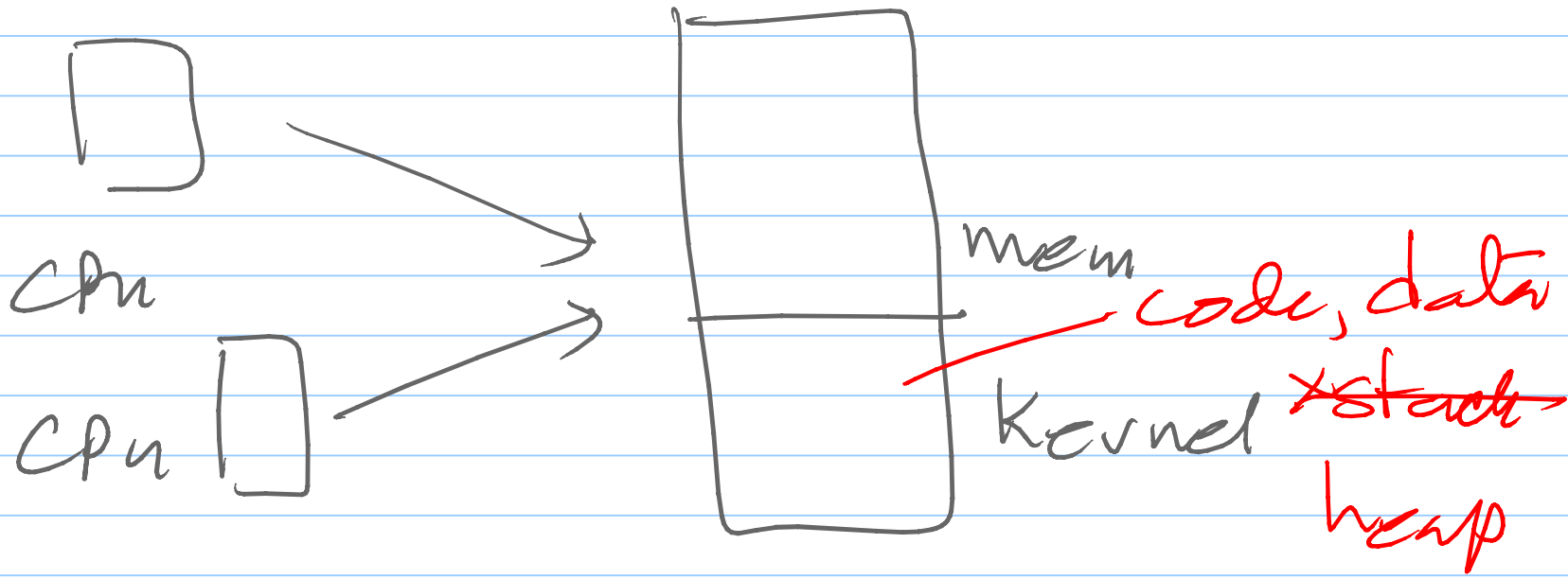
hierarchical caches

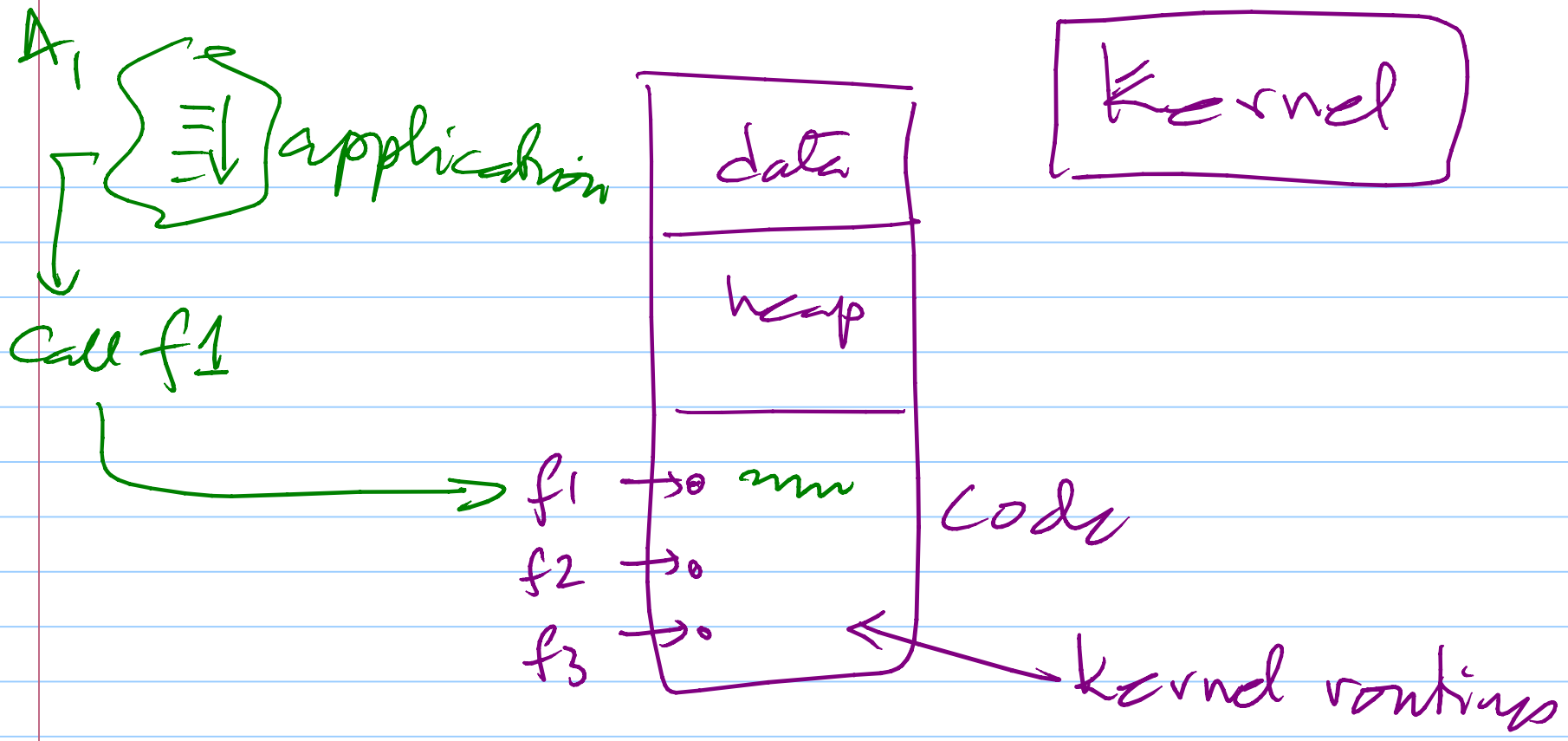
CC-NUMA

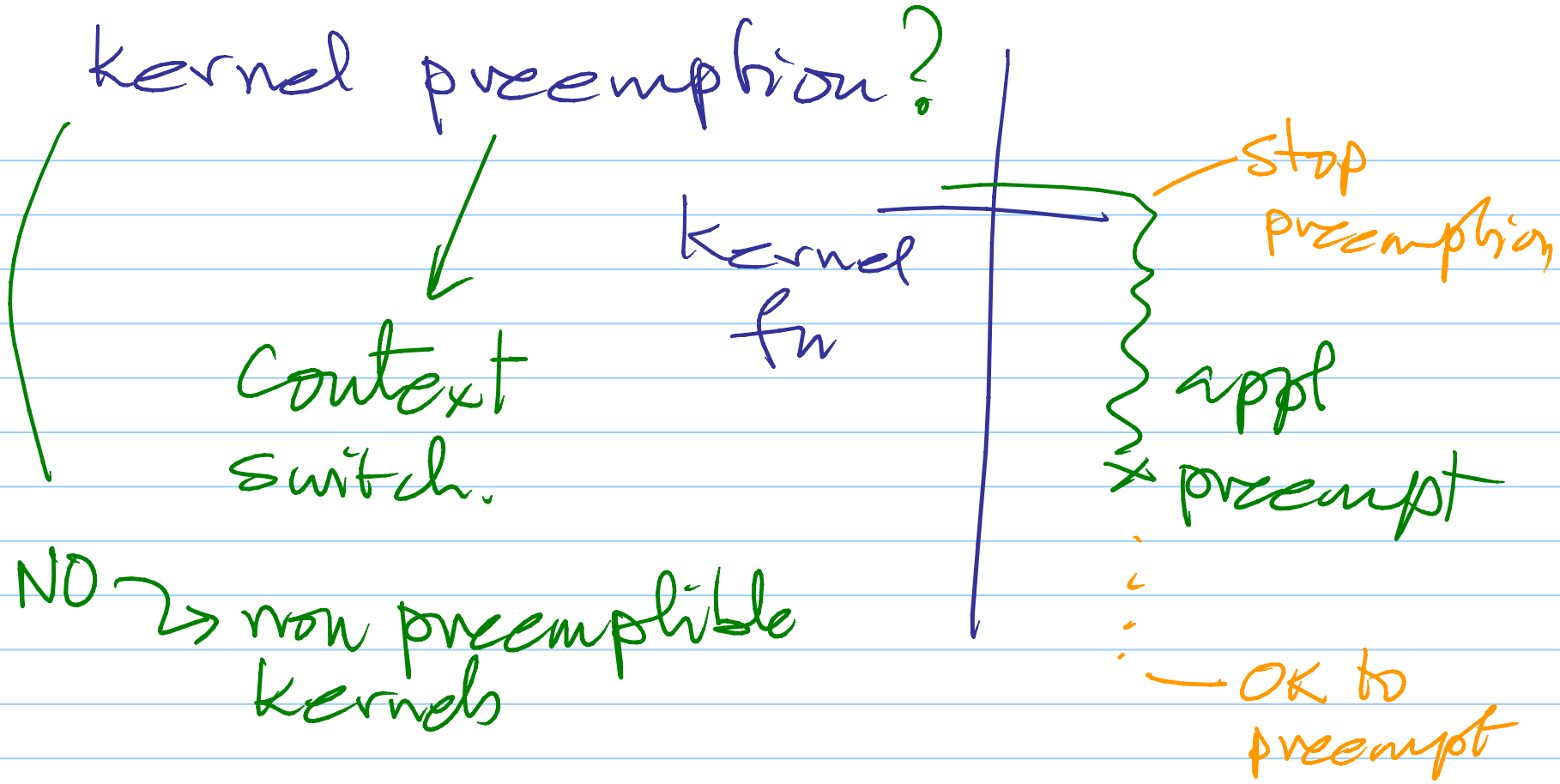


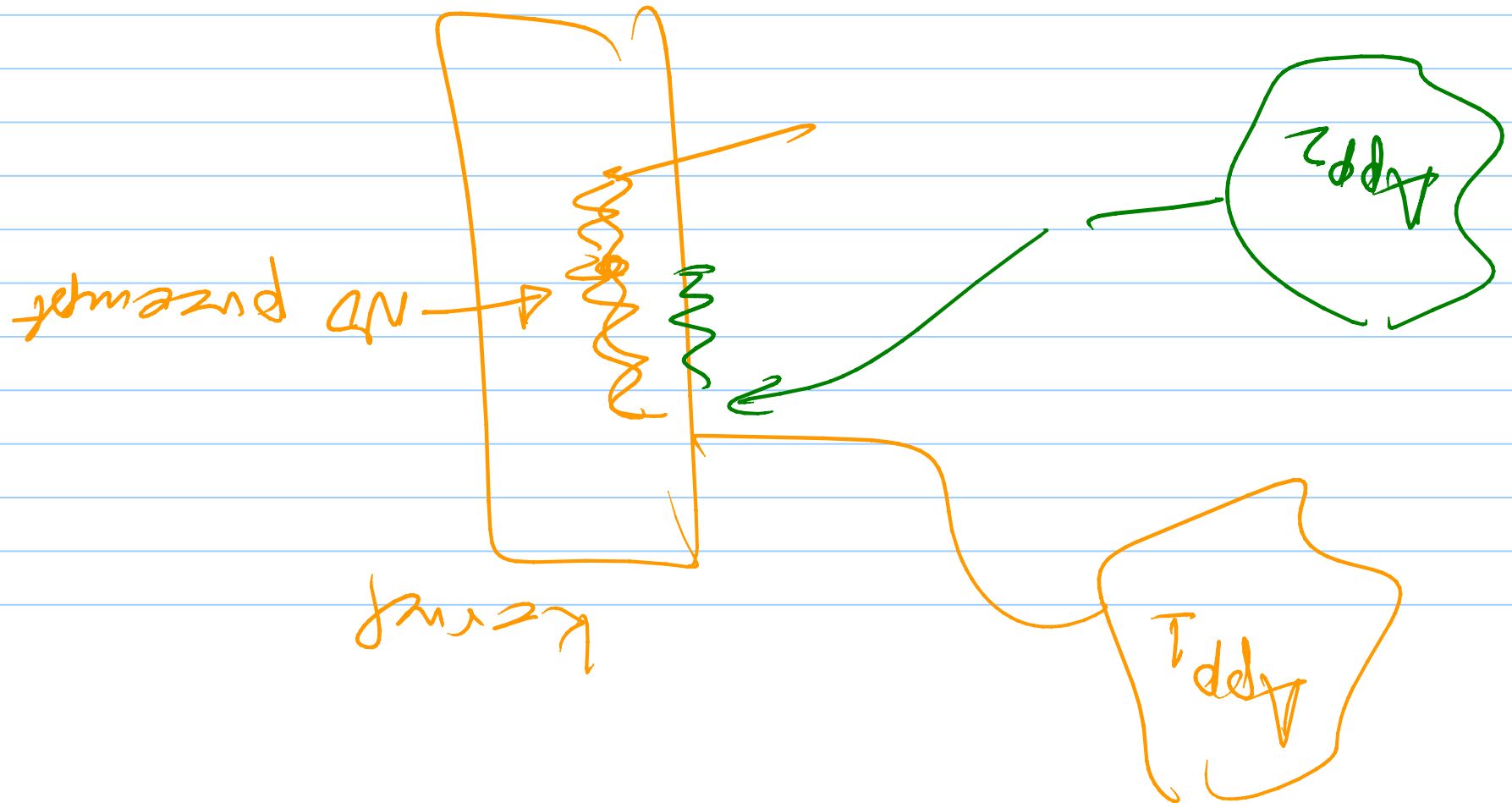
[cache coherent NUMA]

The shared memory problem




A1
application
call f1
data
heap
f1
f2
f3
code
kernel
kernel routines


kernel preemption?
context switch.
NO → non preemptible kernels
kernel fn
stop preemption
appl preempt
OK to preempt



Kernels

UNIPROC

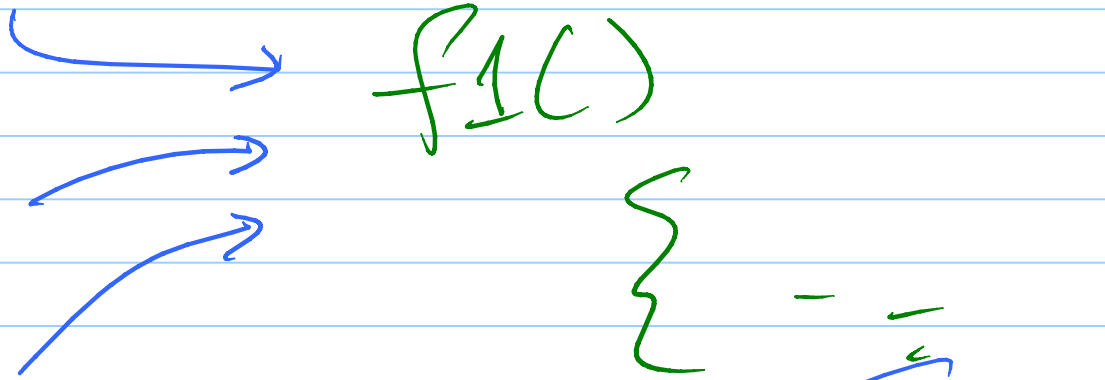
① not preemptible (V1)

mP → ② uses "big lock" (V2)

③ No biglock (V3)

Single / multiple CPU

int x | global



- application processes
- threads

Single proc