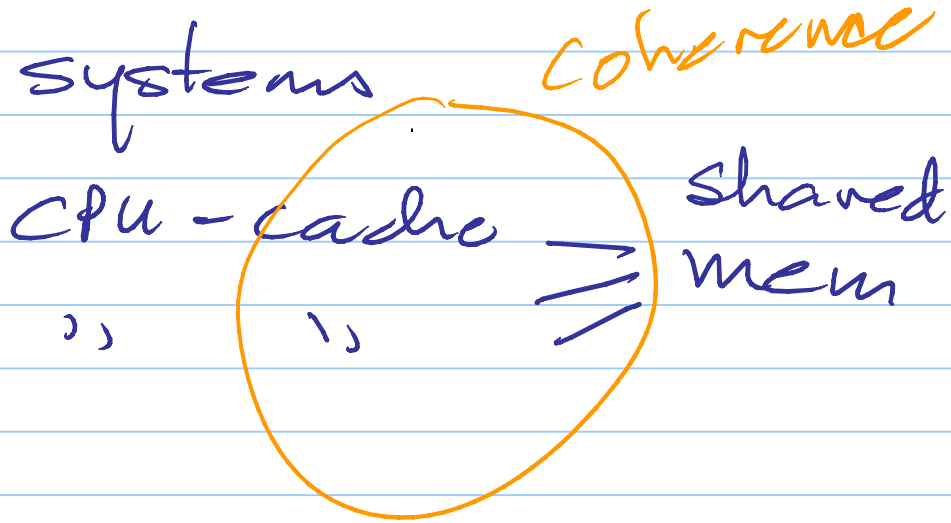
CSE 531: Distributed and Multiprocessor Operating Systems

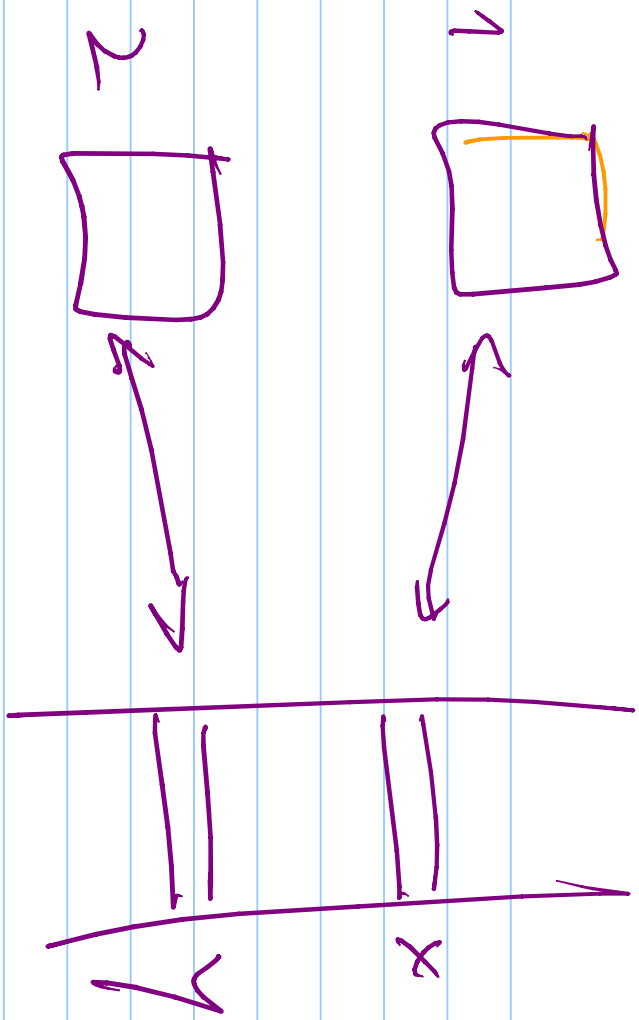
Class: 04 Date: 8/29

Fall 2017

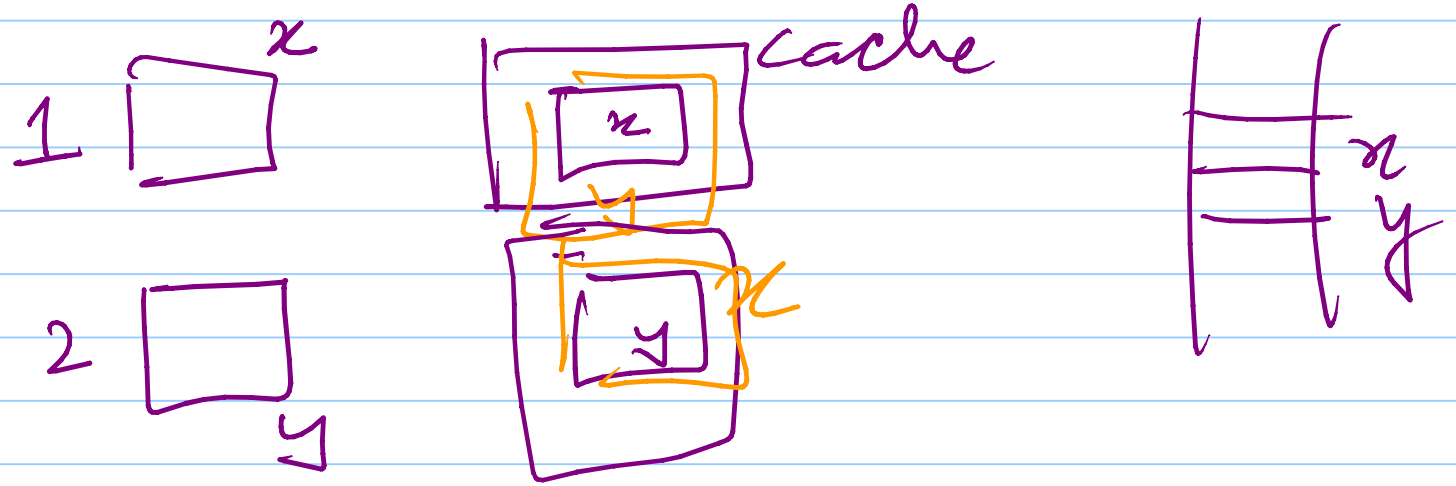
Note Title

UMA systems





Caches can spoil performance due to false sharing



Multiple processors — why?

1. Power / compute capacity
2. Parallelism ...
   - multiple application / services
   - parallel application

Parallel applications

memory is shared

1 app → multiple threads

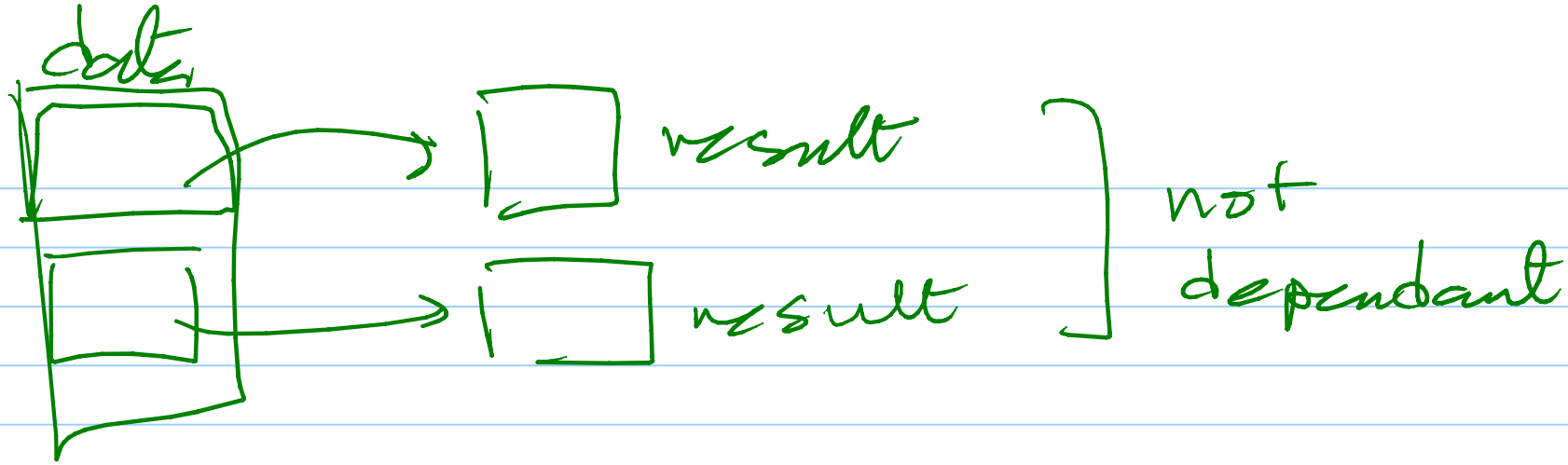
or

processes

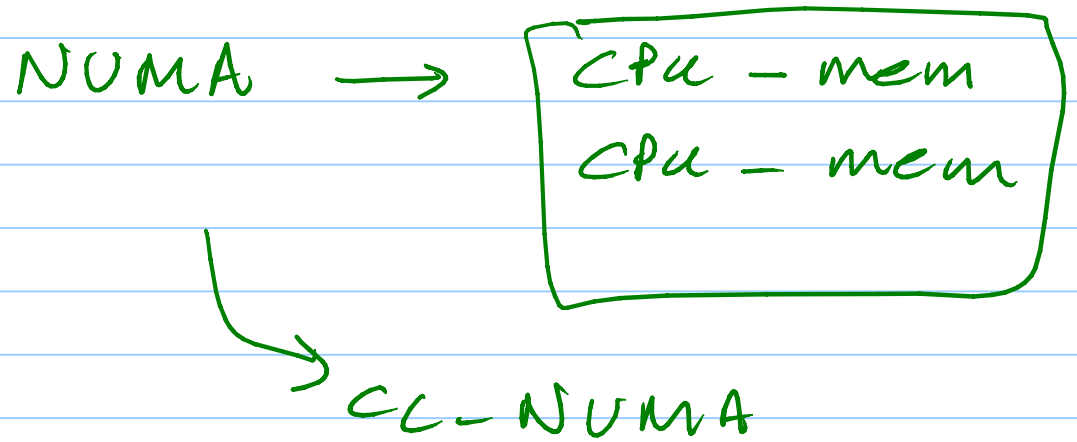
memory is private

→ cooperative

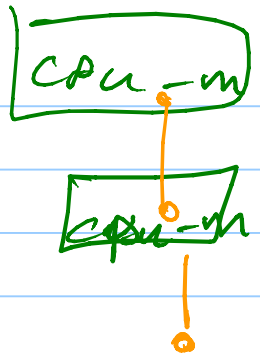
→ parallel - "embarrasingly" →

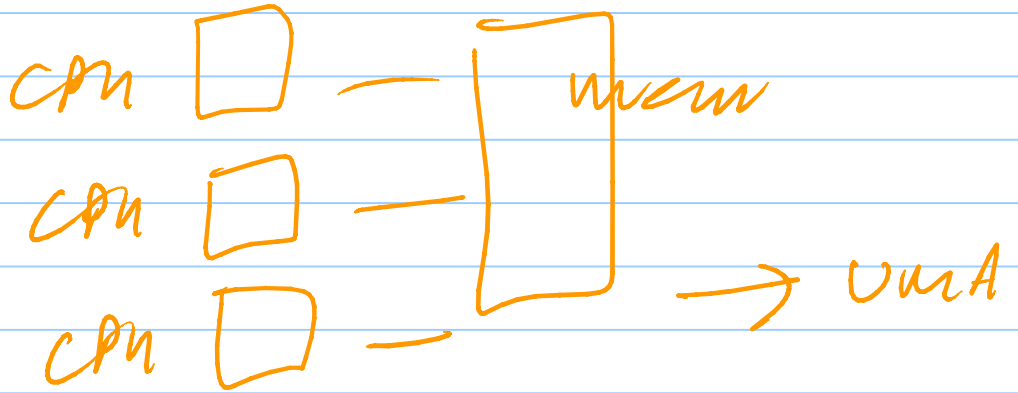

data
result
result
not
dependant

UMA ✓

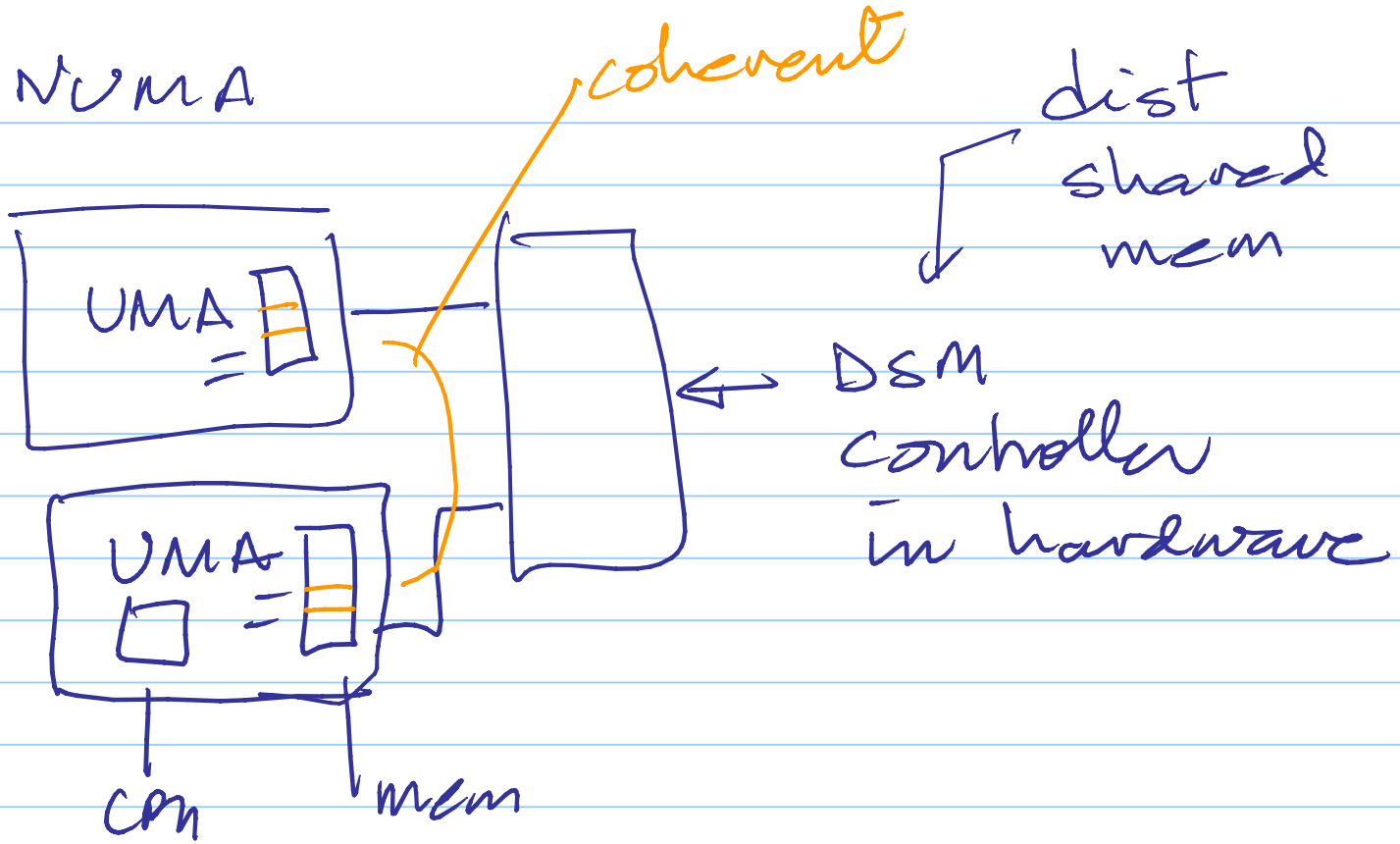



cpu mem


cpu-m
cpu-n

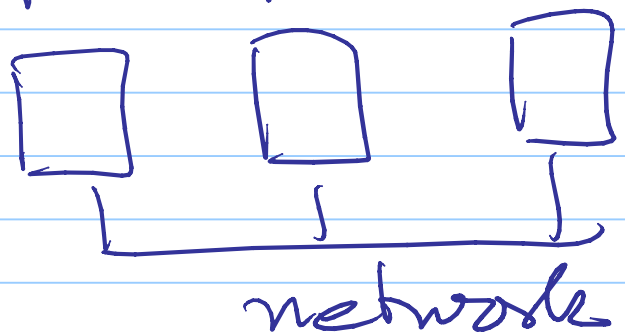

cpu
cpu
cpu
mem
UMA

CC-NUMA

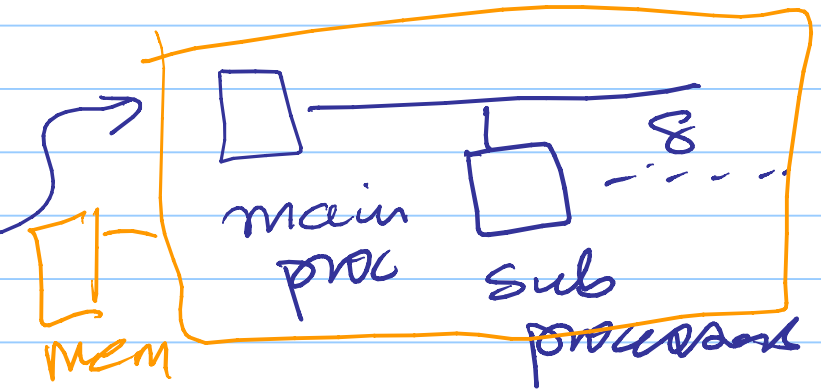

coherent
dist
shared
mem
UMA
UMA
DSM
controller
in hardware
CPU
mem

NORMA or distributed

① cluster of computers



② Single computer with NORMA-like features

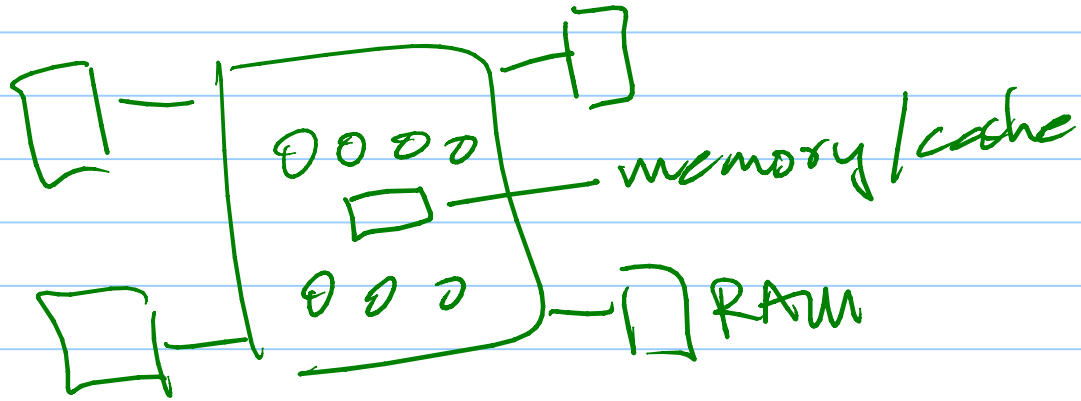


- IBM Cell
- Intel SCC
- Intel Xeon Phi

x Intel SCC

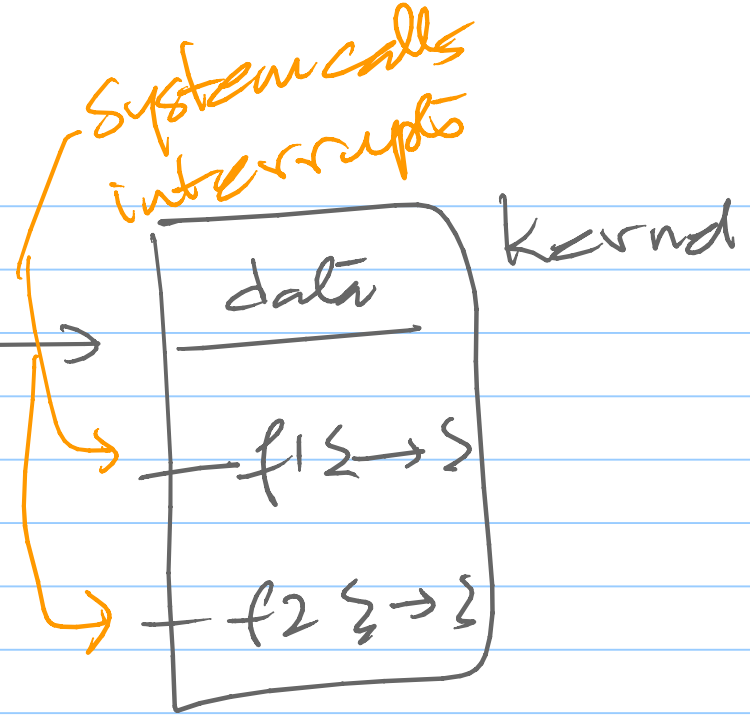
✓ Intel Xeon Phi

→ 40 - 100 CPU's @ > 1 GHz

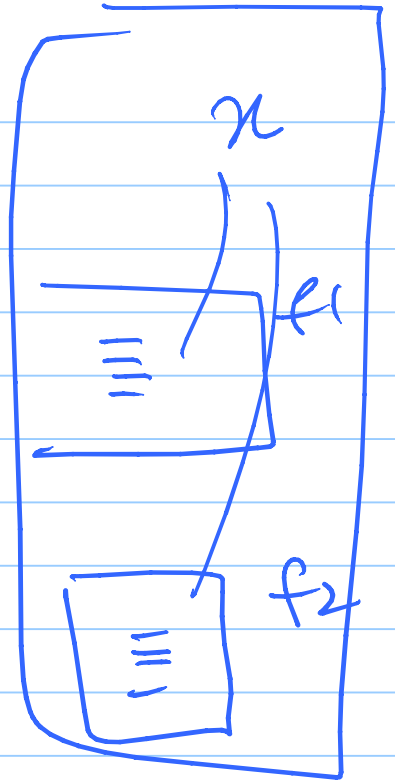


shared memory

- kernels
- threads



threads



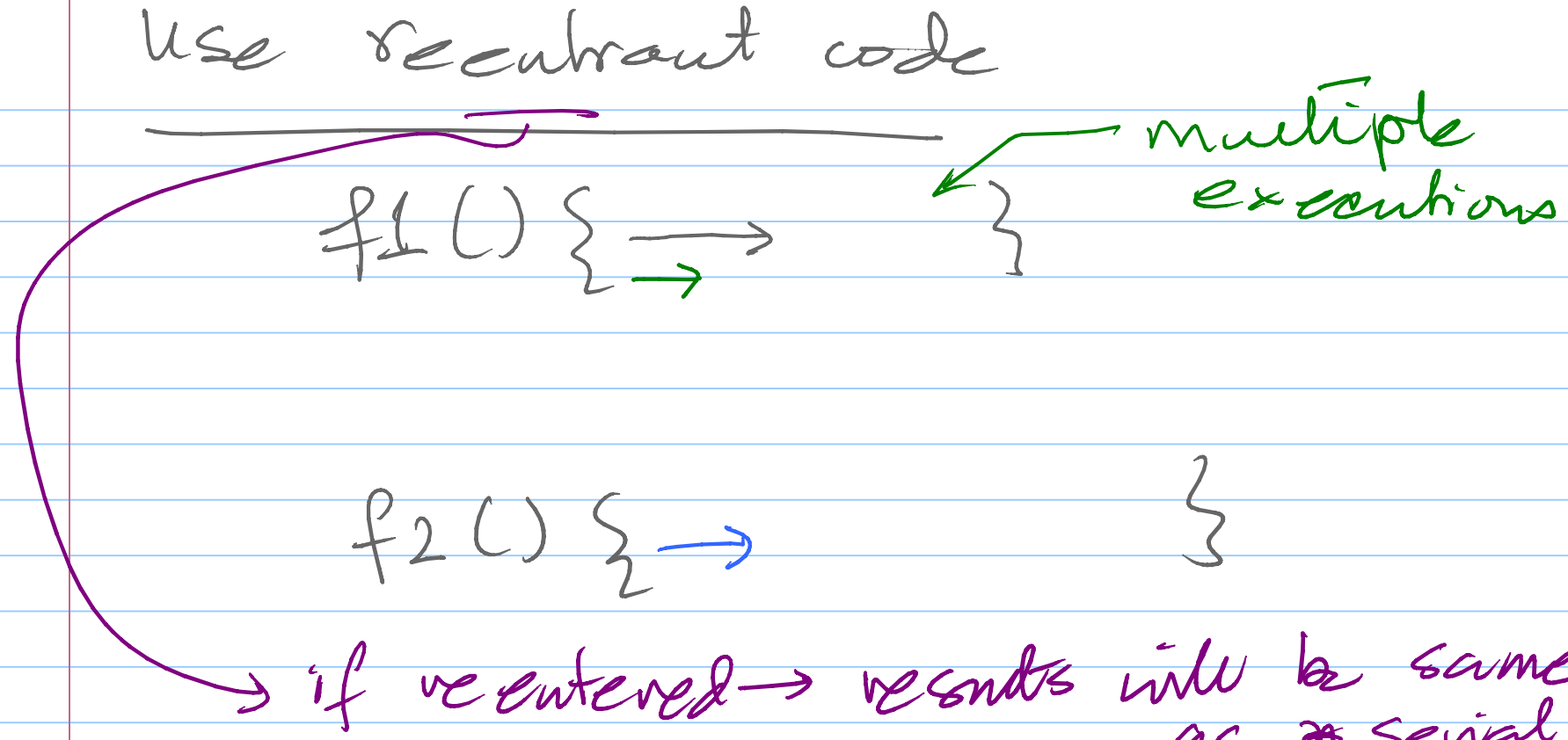
application memory

Concurrent threads (or kernel calls) → Do not share memory

⇓

Do not use globals

Use reentrant code

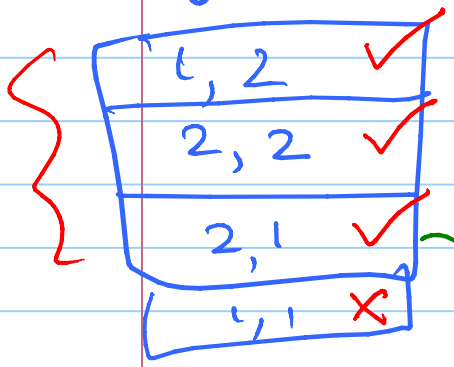


RACE Condition

global $x = 0$

	Thread 1	Thread 2
	$x = x+1$	$x = x+1$
	:	
	print x	print x

output



→ final value = 1 or 2

1 1

$x = x + 1$

$x = x + 1$

1. load $x \rightarrow R1$
3. add 1 to R1
5. store R1 $\rightarrow x$

2. load $x \rightarrow R1$
4. add 1 to R1
6. store R1 $\rightarrow x$

final $\rightarrow$ $x = 1$

How to get reentrant code?

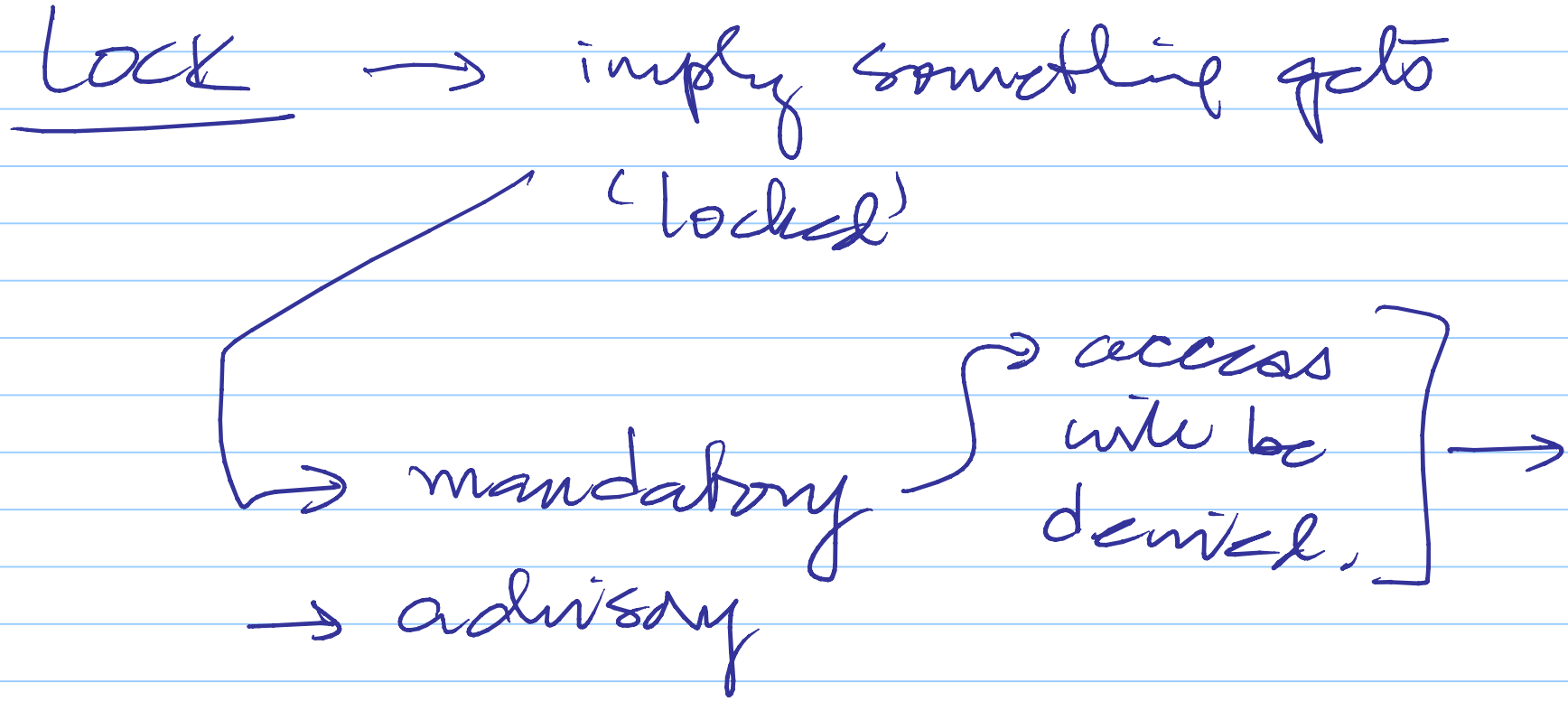
1. Lock ⟶ gives critical sections
2. synchronize

↓

critical section

↓

one execution @ any point in time


Lock → imply something gets
'locked'
→ mandatory
→ advisory
→ access will be denied.

lock(x)

x = x+1

unlock(x)

print(x)

Advisory locks

lock(x)

x = x + 1

unlock(x)

cannot lock an integer data variable

use a lock variable

T_1	T_2
lock(L_x)	
$x = x + 1$	SAME
unlock(L_x)	