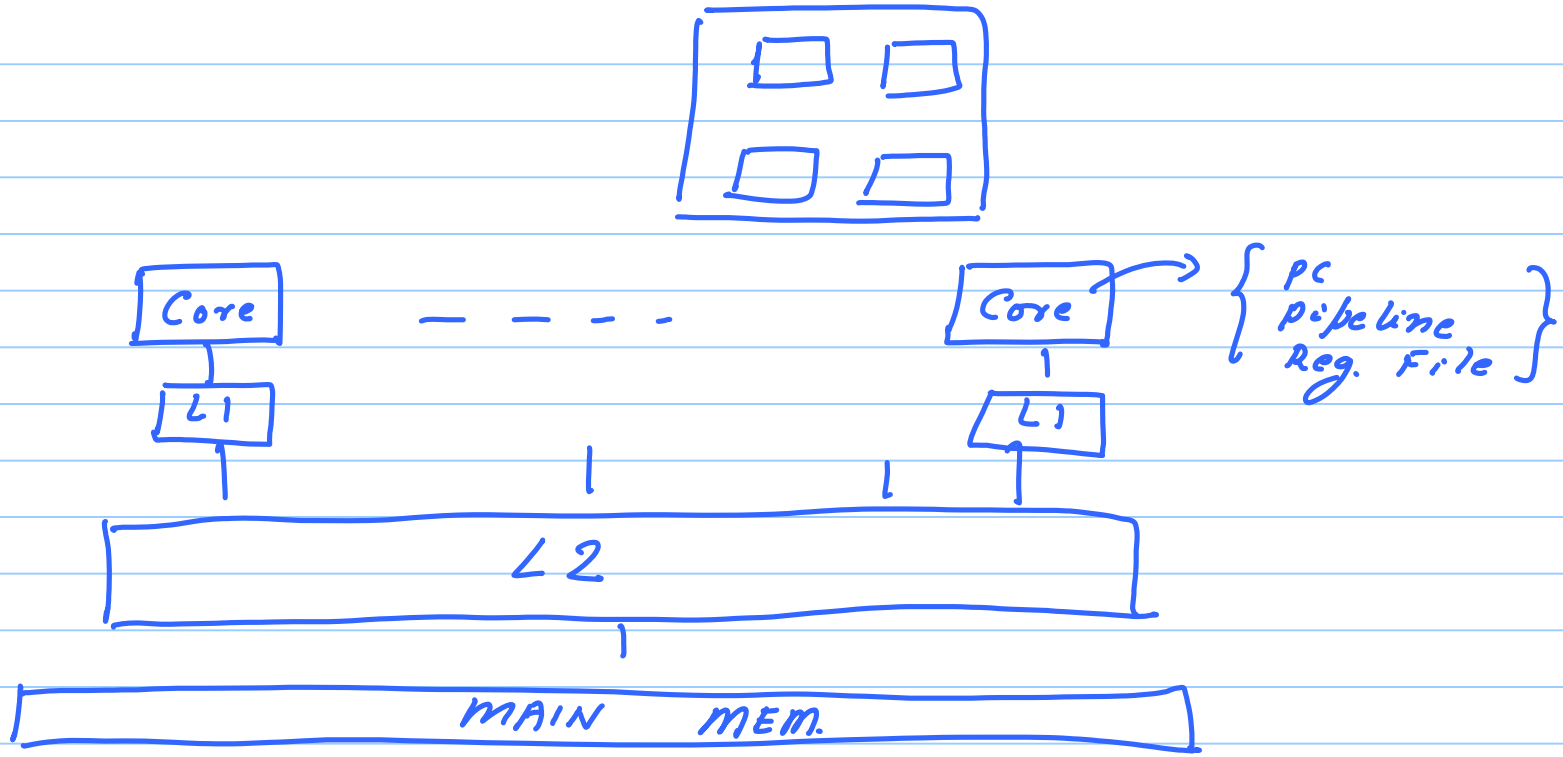
Nov - 8

Note Title 08-11-2012

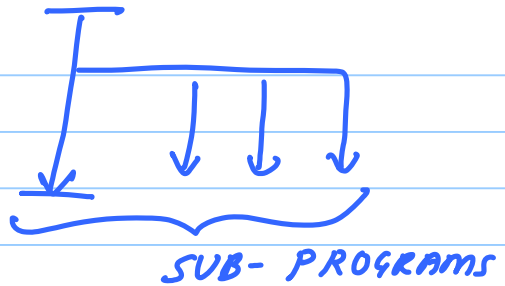
Multiprocessors

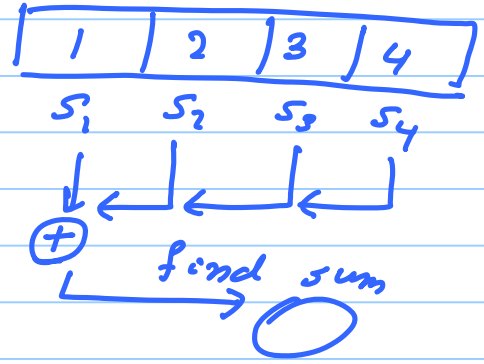
- Shared Memory
  - Multicore
  - Multithreaded.
- Distributed Memory

Multicore.



Parallel Program



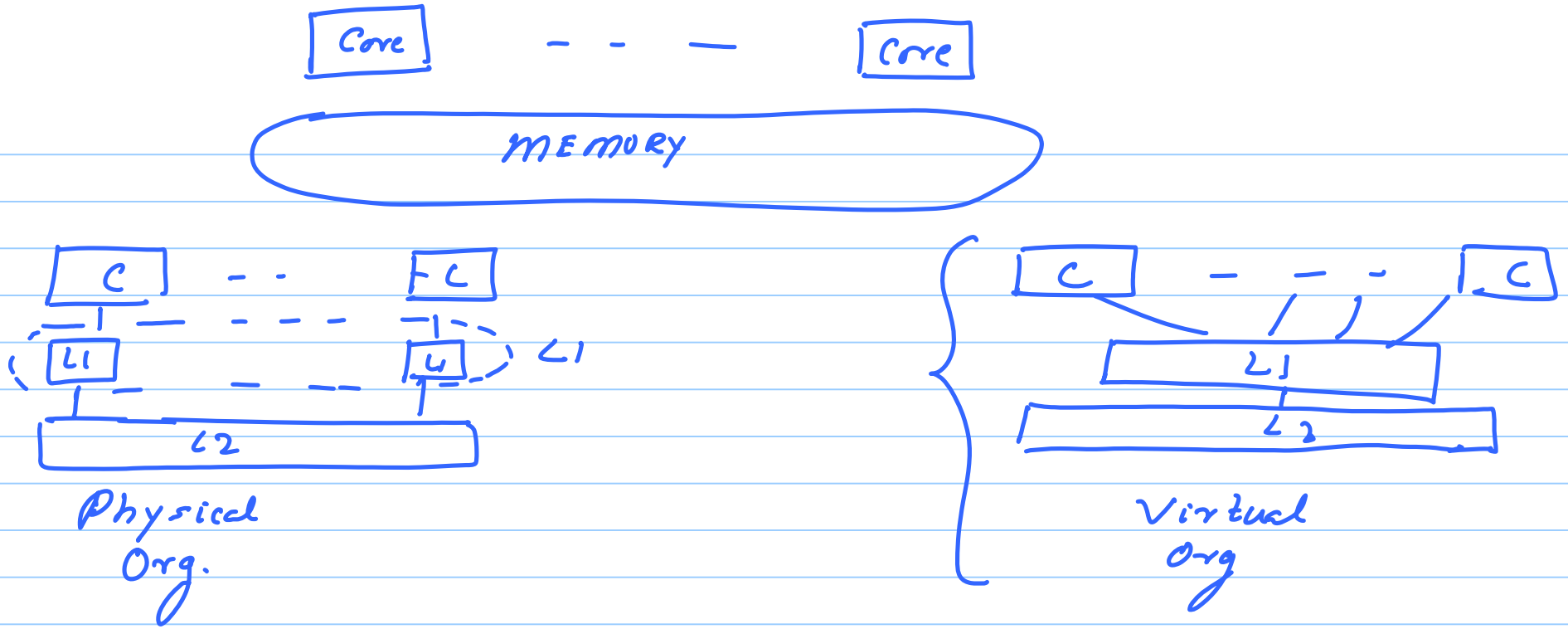


Ex: Add : 1 to 10^6

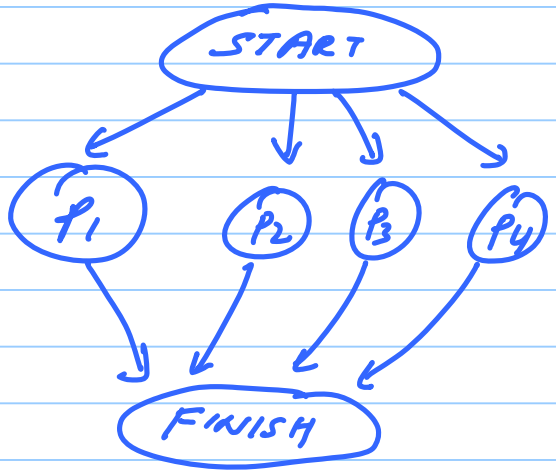
1) Divide the set of numbers into four parts

2) Each processor computes its partial sum.

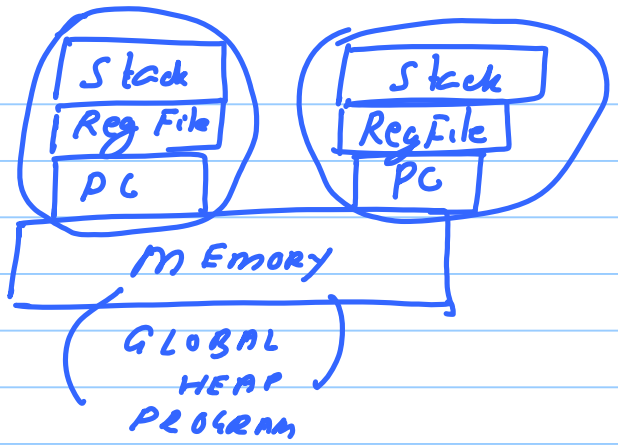
3) Final sum = Sum of partial sums


Core
Core
MEMORY
C
C
L1
L1
L2
Physical
Org.
C
C
L1
L2
Virtual
Org

Cache Coherence: Make a distributed set of caches behave as one unified single cache.



SUB - PROGRAMS.



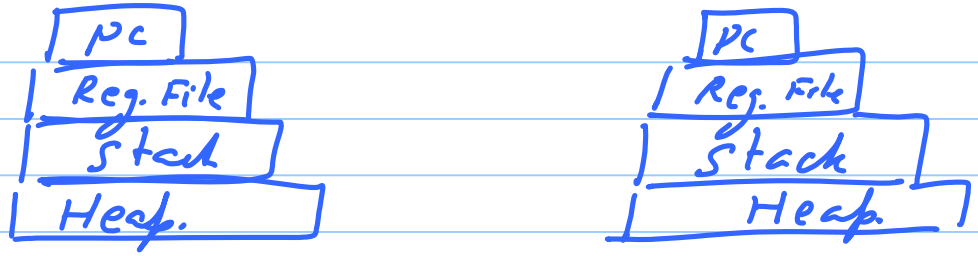
Software Thread:

Private: Registers, PC, stack

Shared: global, heap, program (text)

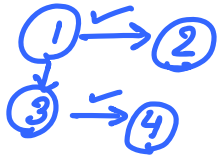
C → google for pthreads

PROCESS → Instance of a running program



Two processes do not share anything between each other.

Example



$x = 0,\ y = 0$

T_1:	T_2:
① $x = 1$	③ $y = 1$
② $r_1 = y$	④ $r_2 = x$

Q : $(r_1 = 0, r_2 = 0)$ possible ?

Set of possible outcomes:

$x = 1$	r_1	r_2
$r_1 = y$	0	1
$y = 1$		
$r_2 = x$		

	r_1	r_2
$x = 1$		
$y = 1$		
$r_1 = y$	1	1
$r_2 = x$		

---

	r_1	r_2
$y = 1$		
$r_2 = x$	1	0
$x = 1$		
$r_1 = y$		

When instructions execute in program order.

$(r_1 = 0,\ r_2 = 0)$ Not possible

Sequential Consistency

On Intel Multicores:

$(r_1 = 0,\ r_2 = 0)$ is possible

Seq. Consistency for different addresses

- $R \to W$
- $W \to W$
- $R \to R$
- $W \to R$.

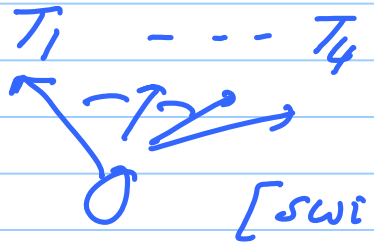
INTEL TSO consistency for diff. addresses.

- $R \to W$
- $W \to W$
- $R \to R$
- $W \not\to R$

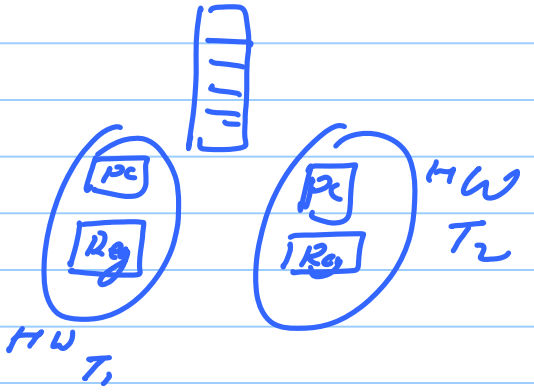
Memory access rules for different addresses $\longrightarrow$ memory consistency

Multi-Threaded Processor.

Fine-grained multi-threading:



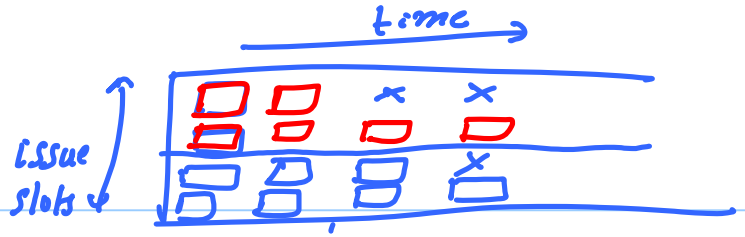
[switches every cycle]

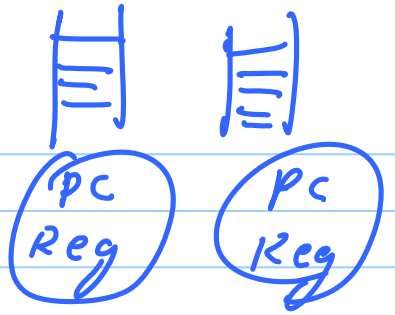


Coarse grained multi-threading.

switches on an event
→ L2 miss

hyper-threading





Simultaneous Multi-Threading (SMT)