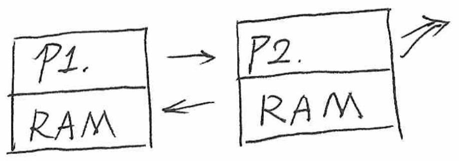
Sep. [illegible]

multiprocessing

↳ parallel programming.

{shared / distributed} > memory. → threads / → processes.



communication.

message-passing.

duals of each other.

(Laver & Needham, 1979)

Threads   Processes.

new Thread( )

Order of the executions of threads matters.

⇓

RACE. condition

↳ nondeterminism

determinism function: $f(x)=f(x)$

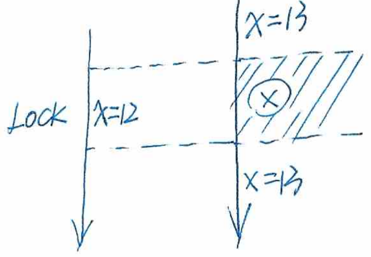
non   function: $f(x) \Rightarrow \begin{cases} 12 \\ 13 \end{cases}$

- local variables (stack) unshared.
- thread local value.

Control Ordering ⟨ wait for threads / synchronization op.

↳ { prevent concurrency / orders events

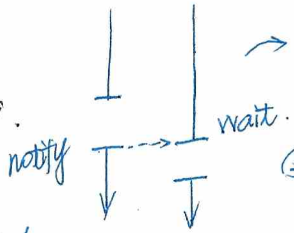
Lock mutual execusion.



condition variables.

In. C++. this→lock( ) could cause dead lock

In Java. wait( ) and notify( )



→ "happens-before".

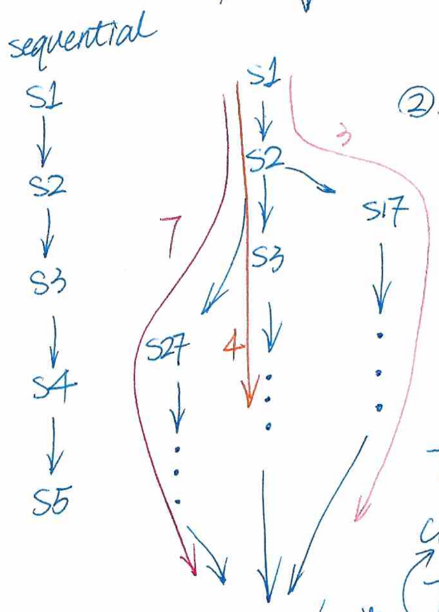
Multithreaded Challengs:

①. Correctness.   Heisenbug.

↳ because of ⟨ nondeter / # potential inter~~fearing~~ leavings.

②. (Speed) Performance.

serialization is the enemy.



Join / wait.

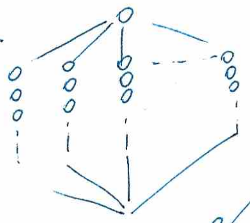
T_1 : 9 units.

Critical Path : ~~tag~~ longest sequence

$T_\infty = 7$ "span"

$T_p \le \frac{T_1}{p} + T_\infty$

embarrassingly.

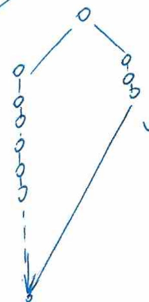


$\rightarrow \left. \begin{array}{l} T_1 = 4000,000 \\ T_p = 1,000,000 \end{array} \right\} \Rightarrow \frac{4,000,000}{1,000,000} = 4$

Let $T_\infty = 4$.

↓

$4+4=8$ ‼



→ load imbalance.