Process Synchronization

1. Mutual Exclusion Fundamentals

   □ Goal: understand mechanisms for controlling parallelism
   □ Classical problem: Producer ⇒ Consumer
     ○ ASM: counter++

     LOAD reg₁, M[counter]
     INC reg₁
     STORE reg₁, M[counter]

2) The Critical Section (CS) Problem

   • There are \( n \) processes competing to use some shared data
   • Within each process is a code segment (CS) that accesses the shared data
   • A solution must meet certain requirements:
     ○ Mutual Exclusion: no collisions
     ○ Progress: no deadlock, no halting
     ○ Bounded Waiting: no starvation

   • Software Solutions (e.g. Peterson's algorithm)
     ○ Common variables used for process synchronization
     ○ Entry code: barrier/synch
     ○ CS: critical section
     ○ Exit code: done, inform/enable others

   • Hardware Solutions (e.g. test-and-set instruction)
     ○ Idea: test and modify memory contents in one memory cycle (atomic)

   • Semaphore (generalized synchronization tool)
     ○ Defn: shared integer variable accessed only through 2 atomic operations
     ○ Operations: wait(S) and signal(S)
     ○ Implementation: integer value with associated linked-list of processes

\[
\text{block}(P_i, S): \text{suspends invoking process on queue associated with } S
\]
\[
\text{wakeup}(S): \text{resumes exactly 1 process blocked on } S
\]