CS31: Introduction to Computer Systems

Week 14, Class 2
Deadlock
05/02/24

Dr. Sukrit Venkatagiri
Swarthmore College

“Deadly Embrace”

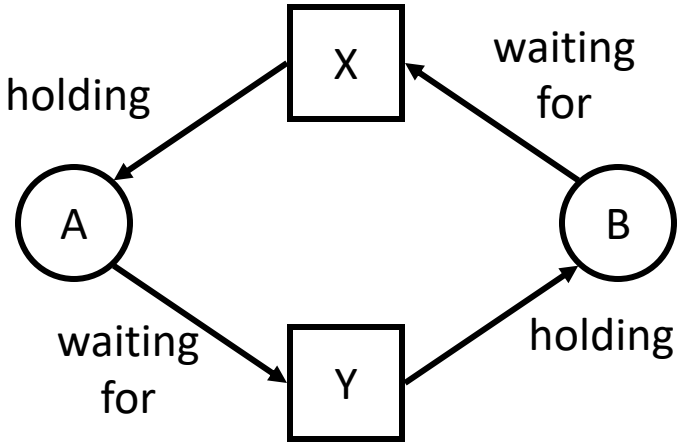
- *The Structure of the THE-Multiprogramming System* (Edsger Dijkstra, 1968)
- Also introduced semaphores, nucleus (kernel today), deadlock
- Deadlock is as old as synchronization

What is Deadlock?

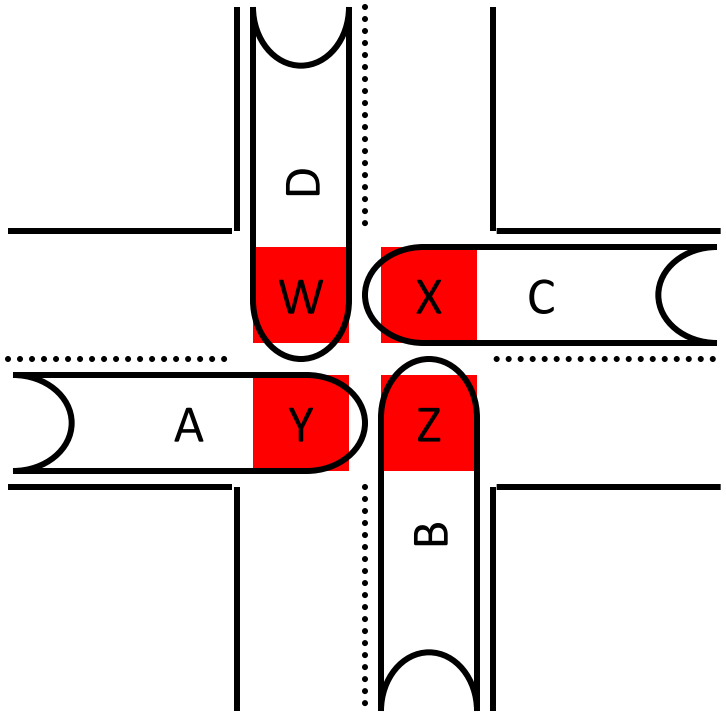
- Deadlock is a problem that can arise:
  - When processes compete for access to limited resources
  - When threads are incorrectly synchronized
- Definition:
  - Deadlock exists among a set of threads if every thread is waiting for an event that can be caused only by another thread in the set.

What is Deadlock?

- Set of threads are permanently blocked
  - Unblocking of one relies on progress of another
  - But none can make progress!
- Example
  - Threads A and B
  - Resources X and Y
  - A holding X, waiting for Y
  - B holding Y, waiting for X
  - Each is waiting for the other; will wait forever



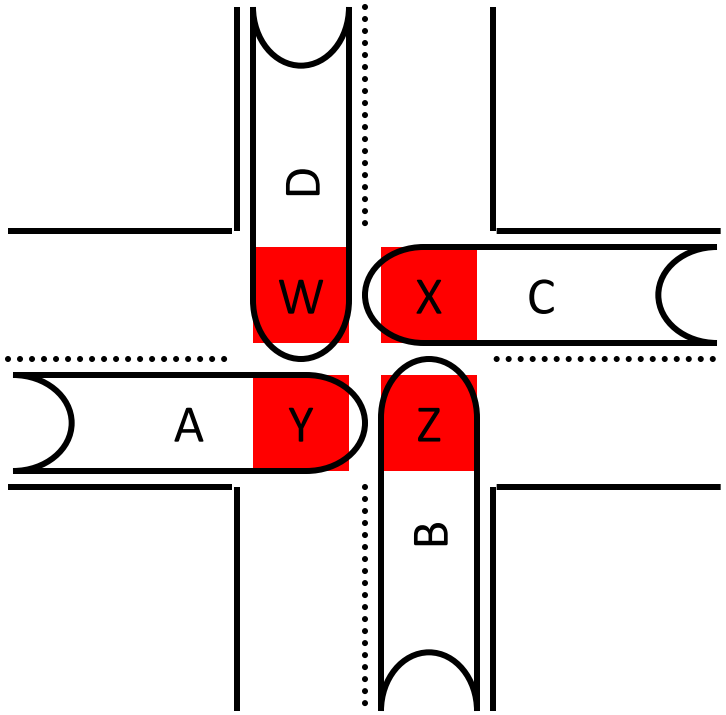
Traffic Jam as Example of Deadlock



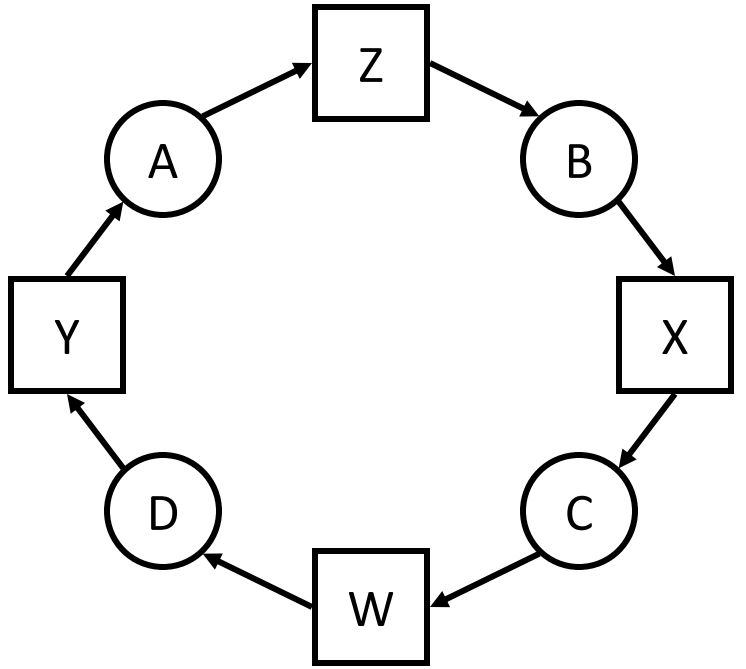
Cars deadlocked
in an intersection

- Cars A, B, C, D
- Road W, X, Y, Z
- Car A holds road space Y, waiting for space Z
- “Gridlock”

Traffic Jam as Example of Deadlock



Cars deadlocked
in an intersection



Resource Allocation
Graph

Four Conditions for Deadlock

1. Mutual Exclusion
   – Only one thread may use a resource at a time.
2. Hold-and-Wait
   – Thread holds resource while waiting for another.
3. No Preemption
   – Can’t take a resource away from a thread.
4. Circular Wait
   – The waiting threads form a cycle.

Four Conditions for Deadlock

1. Mutual Exclusion
   – Only one thread may use a resource at a time.
2. Hold-and-Wait
   – Thread holds resource while waiting for another.
3. No Preemption
   – Can’t take a resource away from a thread.
4. Circular Wait
   – The waiting threads form a cycle.

How to Attack the Deadlock Problem

- What should your OS do to help you?

- Deadlock Prevention
  - Make deadlock impossible by removing a condition
- Deadlock Avoidance (“Banker’s Algorithm”)
  - Avoid getting into situations that lead to deadlock
- Deadlock Detection
  - Don’t try to stop deadlocks
  - Rather, if they happen, detect and resolve

How to Attack the Deadlock Problem

- Deadlock Prevention
  - Make deadlock impossible by removing a condition
- Deadlock Avoidance
  - Avoid getting into situations that lead to deadlock
- Deadlock Detection
  - Don't try to stop deadlocks
  - Rather, if they happen, detect and resolve

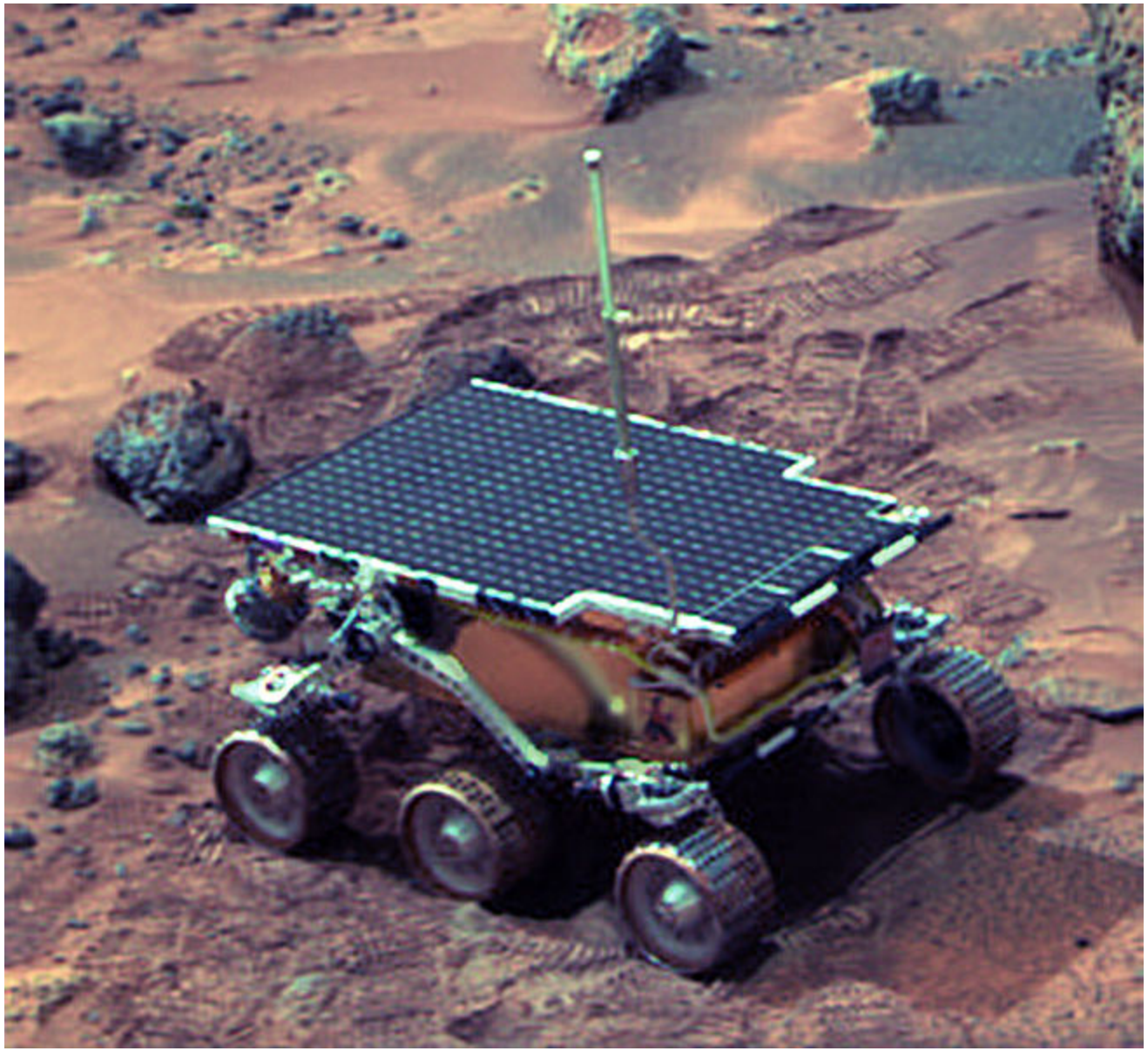
- <u>These all have major drawbacks…</u>
  "Ostrich algorithm"

Other Thread Complications

- Deadlock is not the only problem
- Performance: too much locking?
- Priority inversion
- …

Priority Inversion

- Problem: Low priority thread holds lock, high priority thread waiting for lock.
  - What needs to happen: boost low priority thread so that it can finish, release the lock
  - What sometimes happens in practice: low priority thread not scheduled, can't release lock
- Example: Mars Pathfinder (1997)



Sojourner Rover on Mars

Mars Rover

- Three periodic tasks:
  1. Low priority: collect meteorological data
  2. Medium priority: communicate with NASA
  3. High priority: data storage/movement

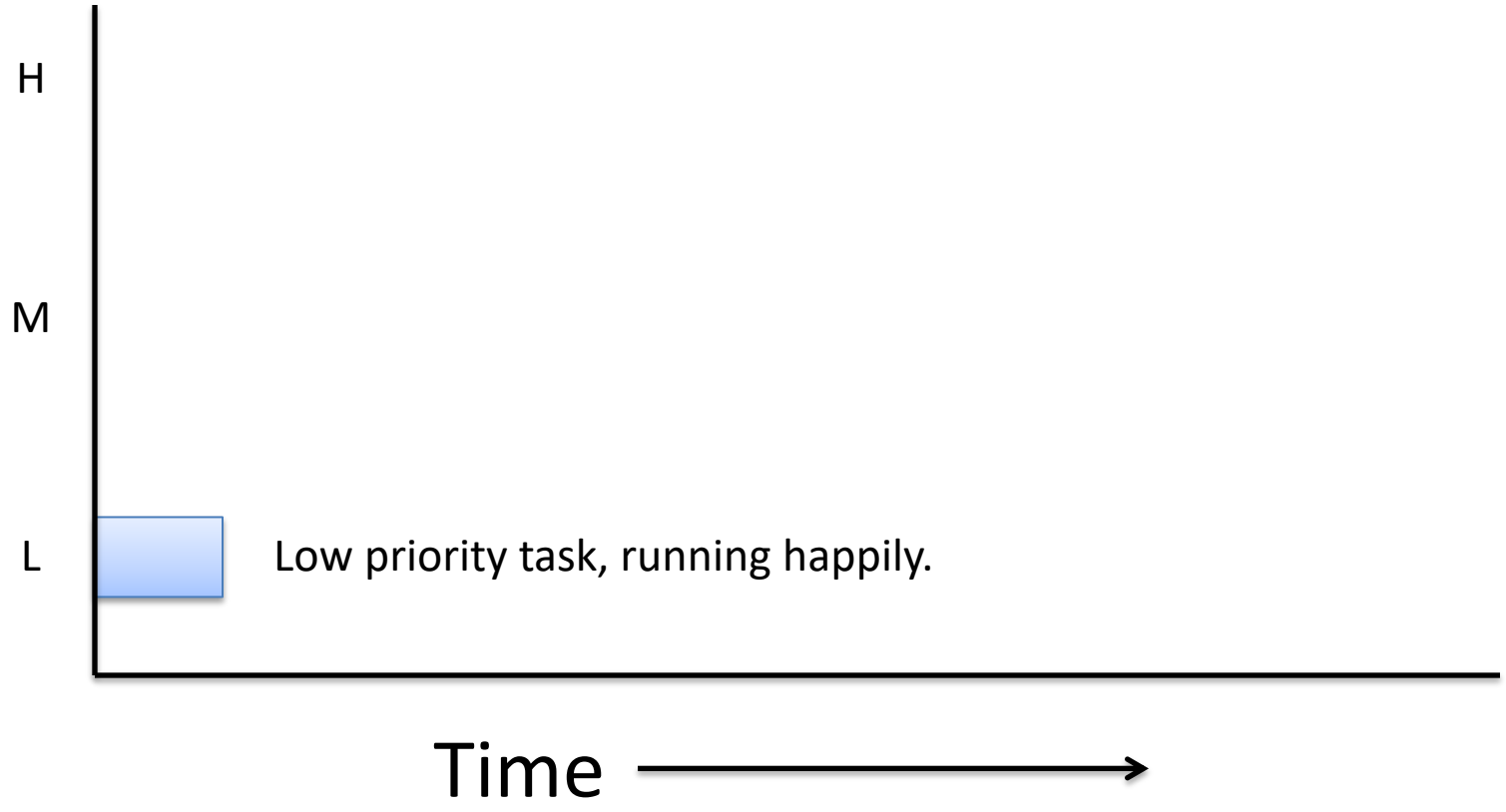
- Tasks 1 and 3 require exclusive access to a hardware bus to move data.
  - Bus protected by a mutex.

Mars Rover

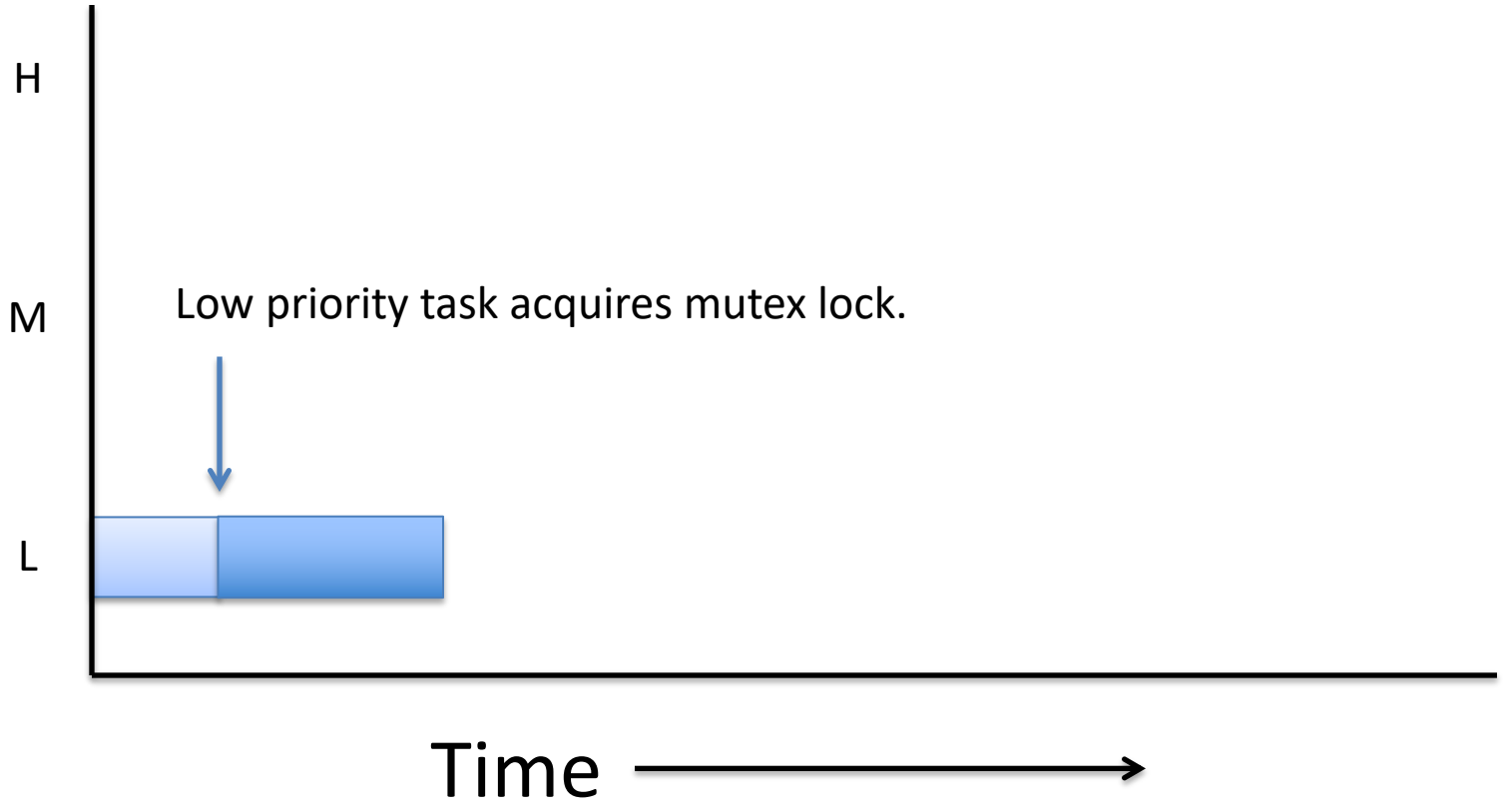
- Failsafe timer (watchdog): if high priority task doesn’t complete in time, reboot system
- Observation: uh-oh, this thing seems to be rebooting a lot, we’re losing data…

JPL engineers later confessed that one or two system resets had occurred in their months of pre-flight testing. They had never been reproducible or explainable, and so the engineers, in a very human-nature response of denial, decided that they probably weren't important, using the rationale "it was probably caused by a hardware glitch".

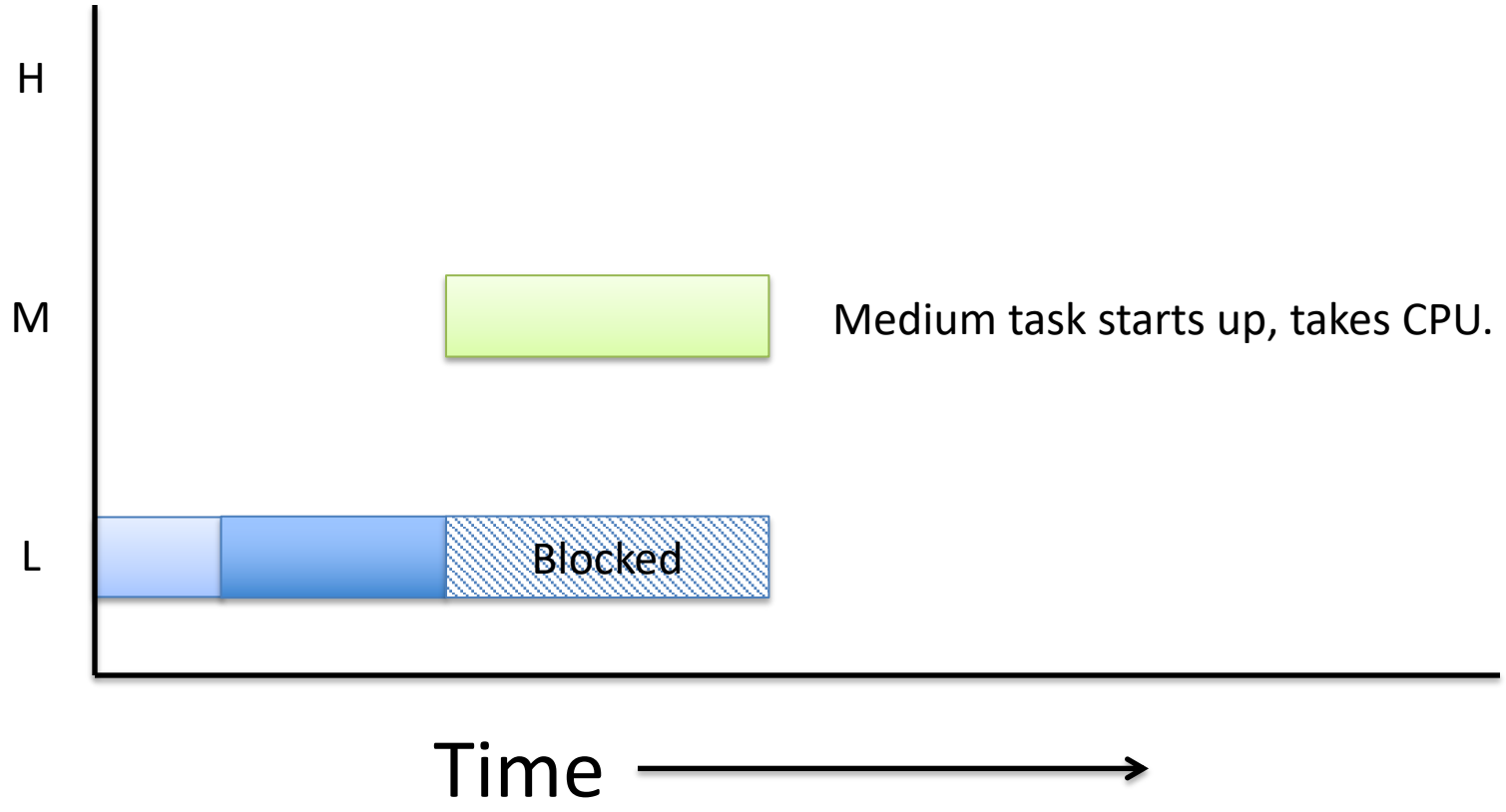
What Happened: Priority Inversion



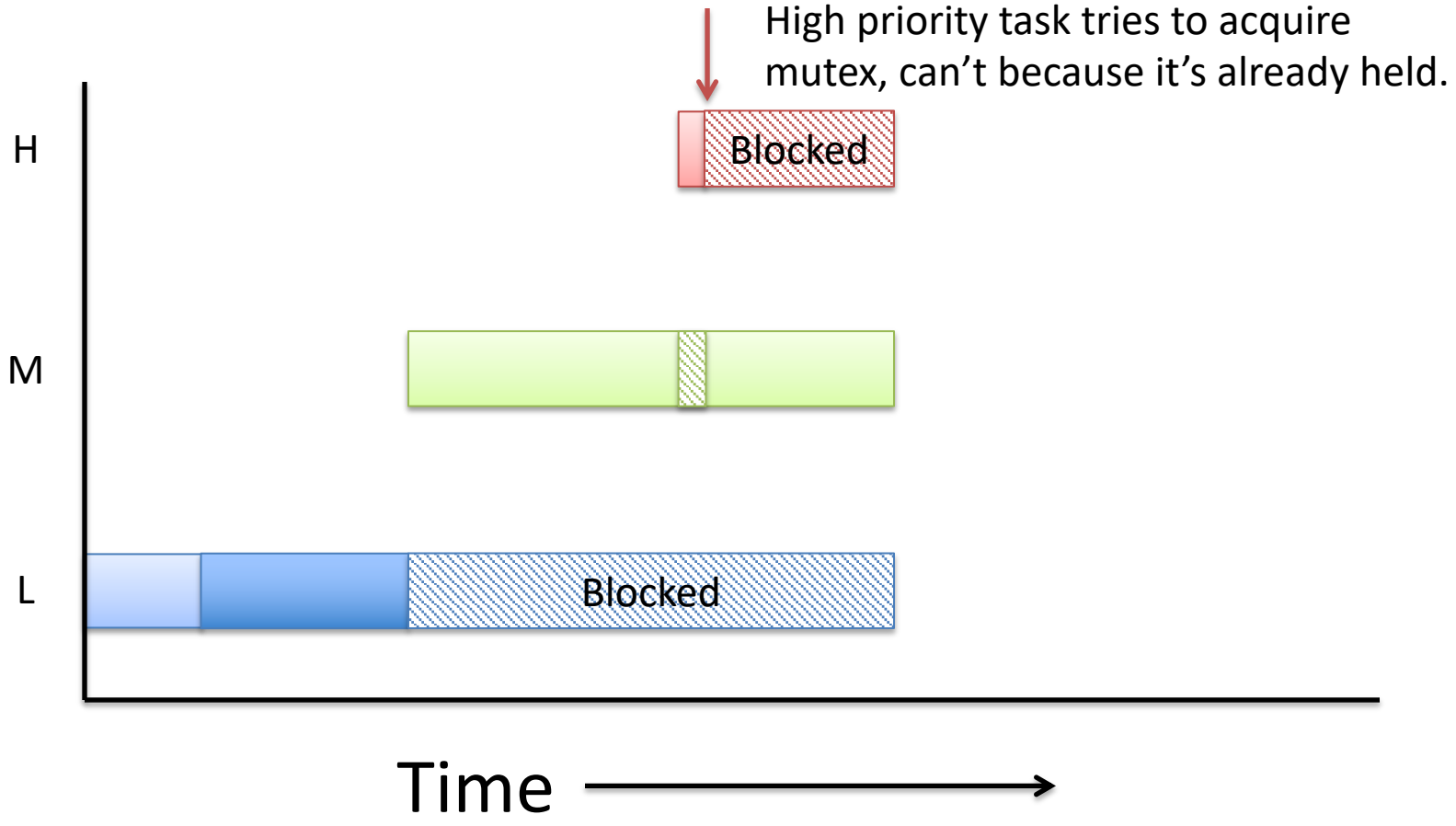
What Happened: Priority Inversion



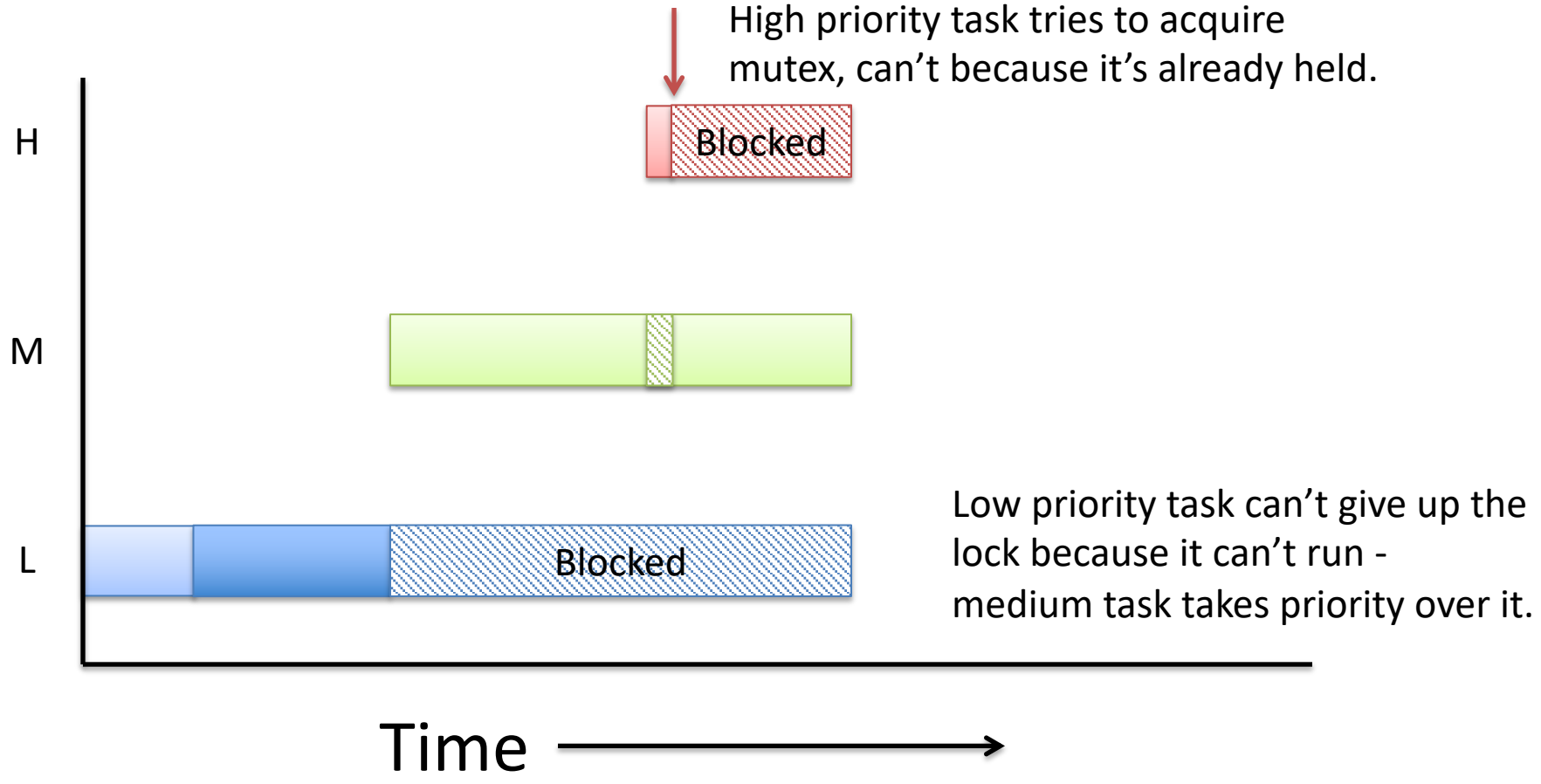
What Happened: Priority Inversion



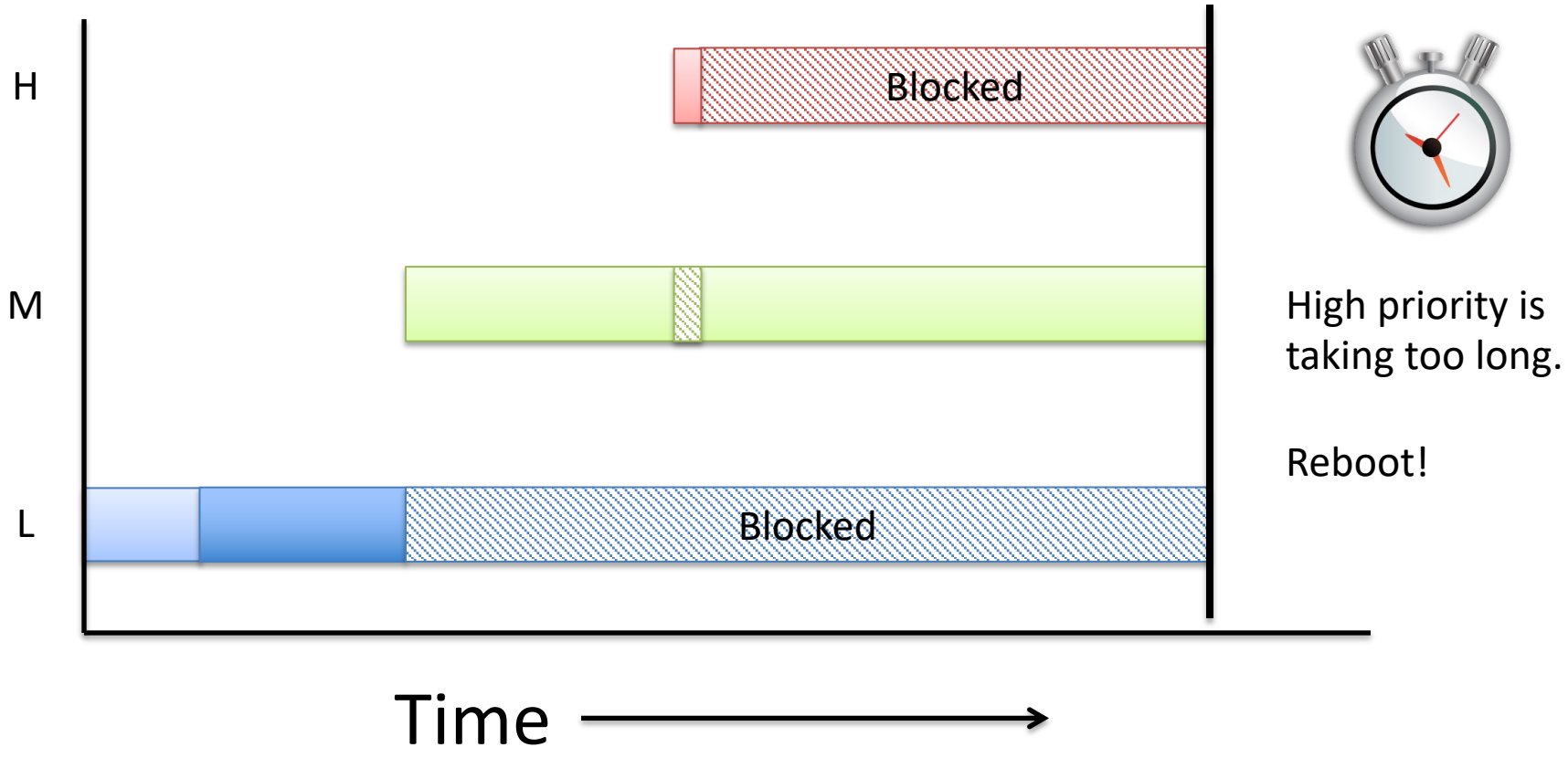
What Happened: Priority Inversion



What Happened: Priority Inversion



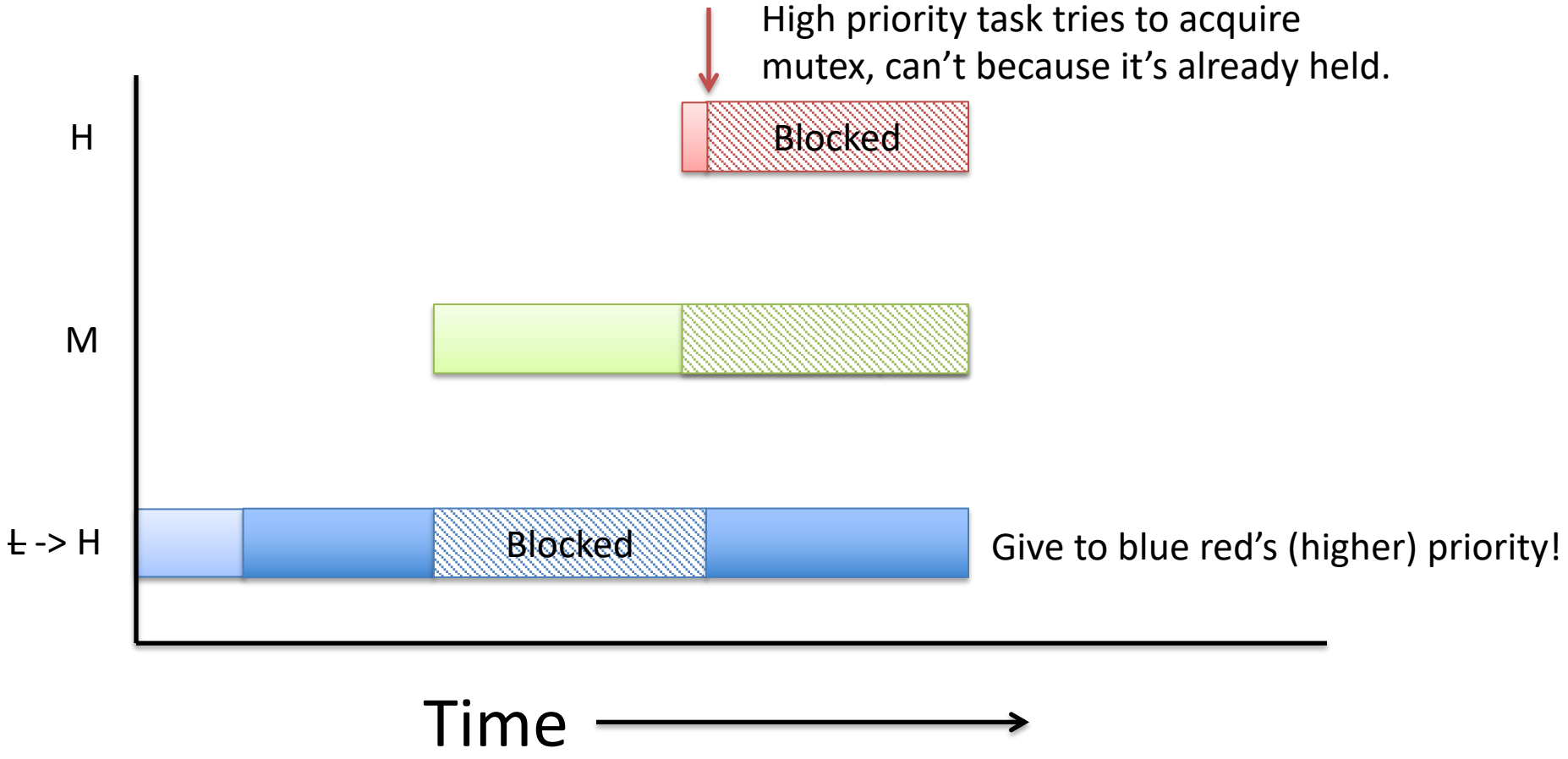
What Happened: Priority Inversion



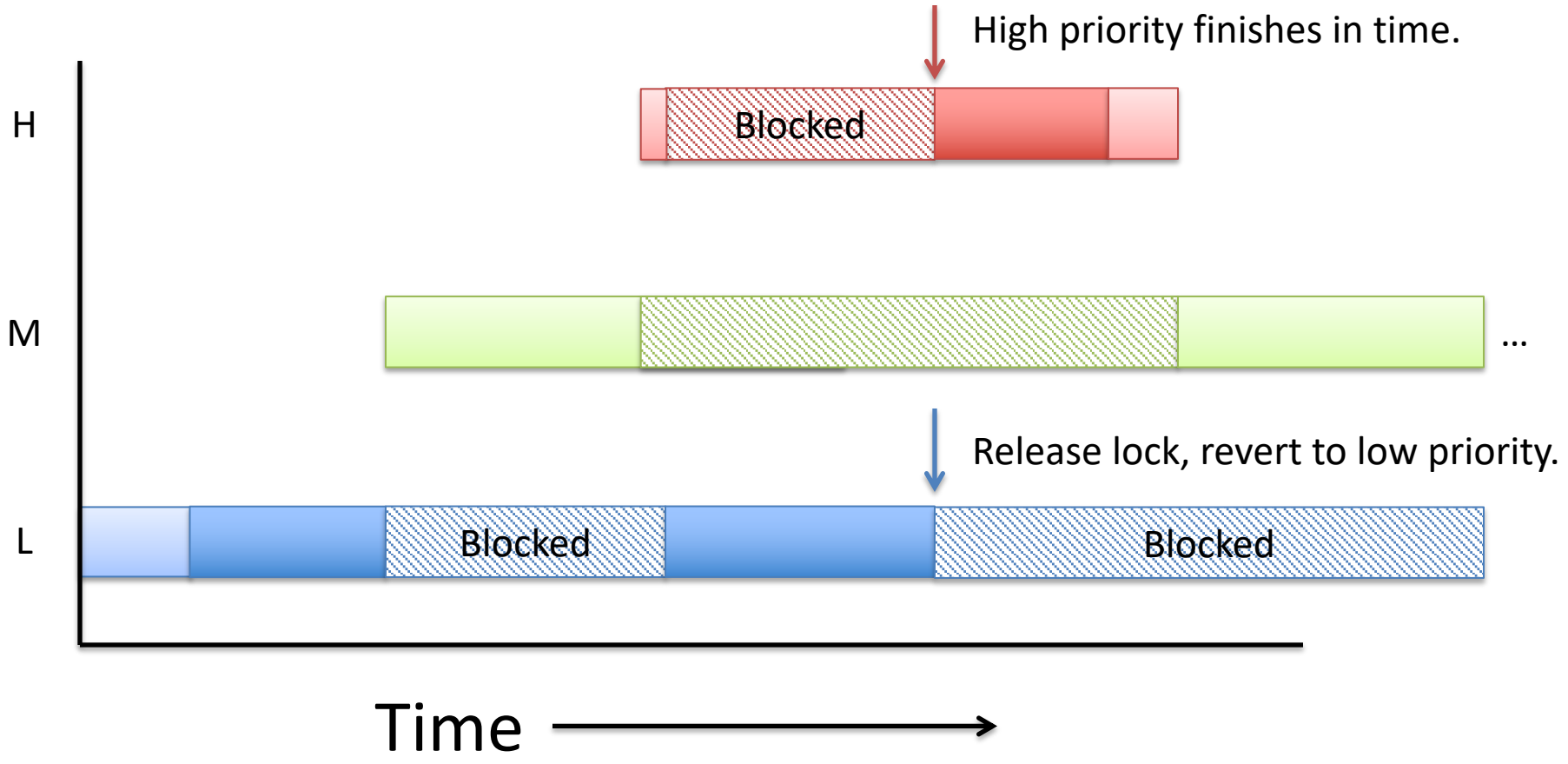
High priority is taking too long.

Reboot!

Solution: Priority Inheritance



Solution: Priority Inheritance

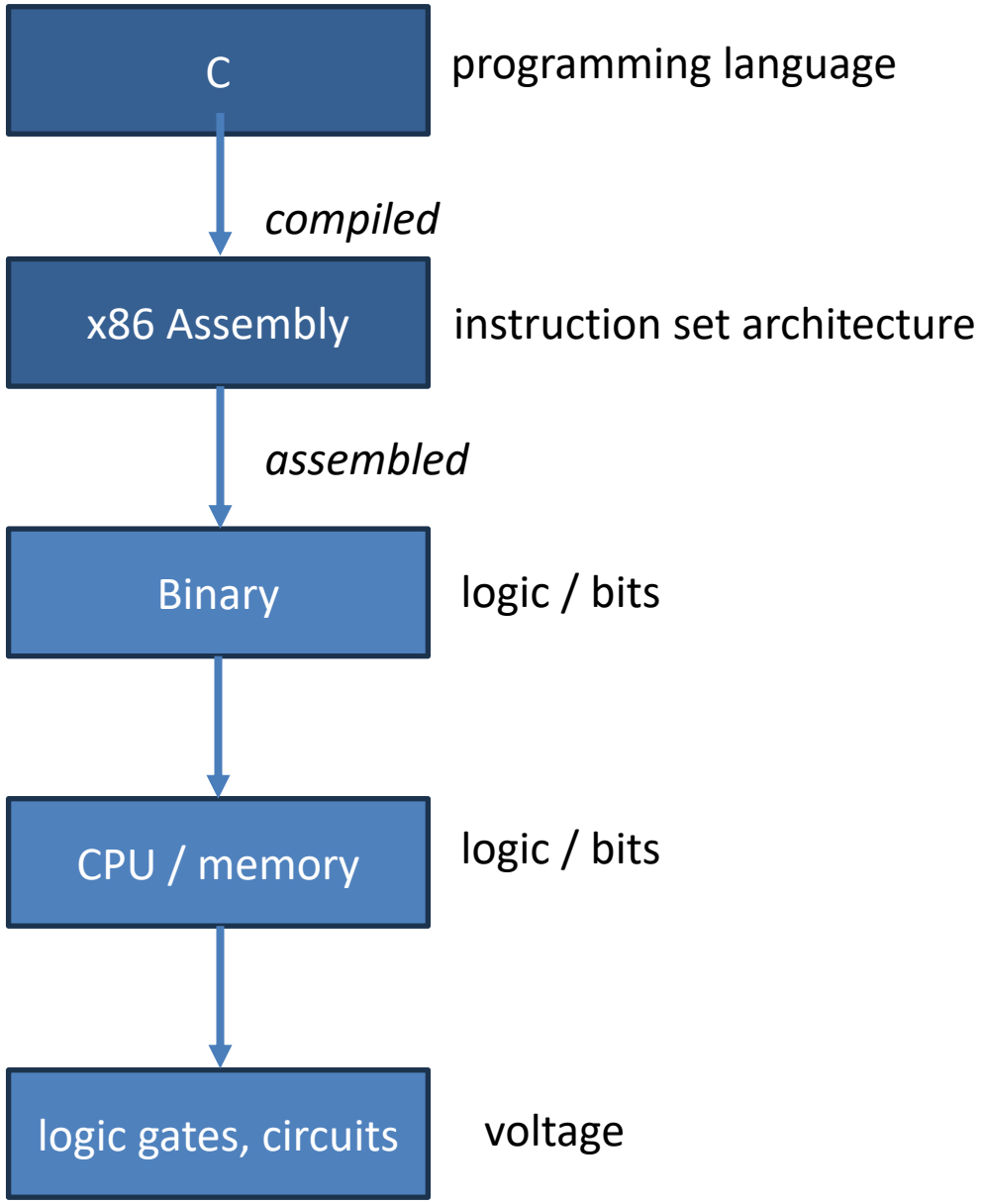


Deadlock Summary

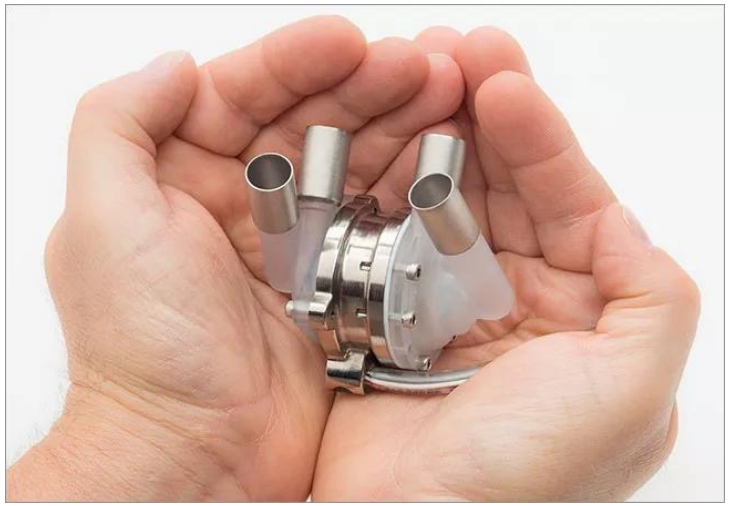
- Deadlock occurs when threads are waiting on each other and cannot make progress.
- Deadlock requires four conditions:
  - Mutual exclusion, hold and wait, no resource preemption, circular wait
- Approaches to dealing with deadlock:
  - Ignore it – Living life on the edge (most common!)
  - Prevention – Make one of the four conditions impossible
  - Avoidance – Banker's Algorithm (control allocation)
  - Detection and Recovery – Look for a cycle, preempt/abort

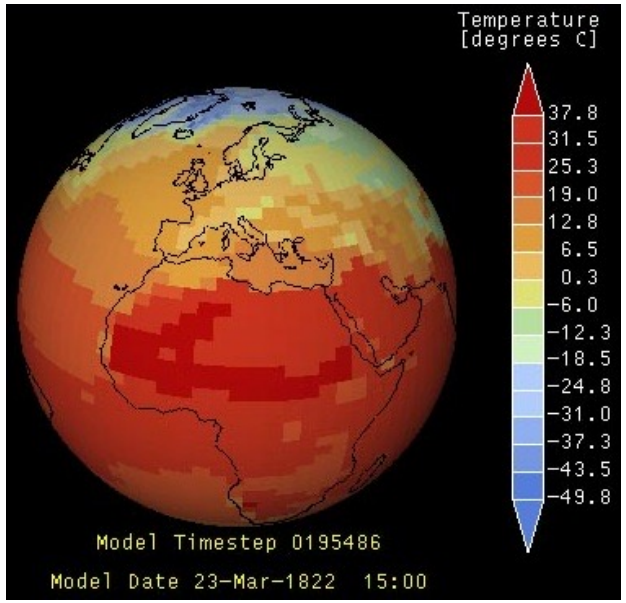
Wk	Lecture	Lab
1	Intro to C	C Arrays, Sorting
2	Binary Representation, Arithmetic	Data Rep. & Conversion
3	Digital Circuits	Circuit Design
4	ISAs & Assembly Language	”
5	Pointers and Memory	Pointers and Assembly
6	Functions and the Stack	Maze Lab
7	Arrays, Structures & Pointers	”
Spring Break		
8	Storage and Memory Hierarchy	Game of Life
9	Storage, Caching	”
10	Caching, Operating System	Strings
11	Processes, Virtual Memory	Unix Shell
12	Parallel Applications, Threading	”
13	Threading	pthreads Game of Life
14	Threading	”
Exam Week		

Where are we?



Your Responsibility as Computer Scientists





Stay tuned for Spring 2025: Social Computing

Today we interact with our friends and enemies, our team partners and romantic partners, our doctors and our teachers, and our organizations and societies, all through computational systems. Think: TikTok, Instagram, iMessage, Slack, Google Docs, Edstem, and so much more. We not only shape these systems, but they shape us, too, individually and collectively.

How can we better design these social computing systems to lead us towards our preferred social future? How do we know whether a system will be effective or even adopted in the first place?