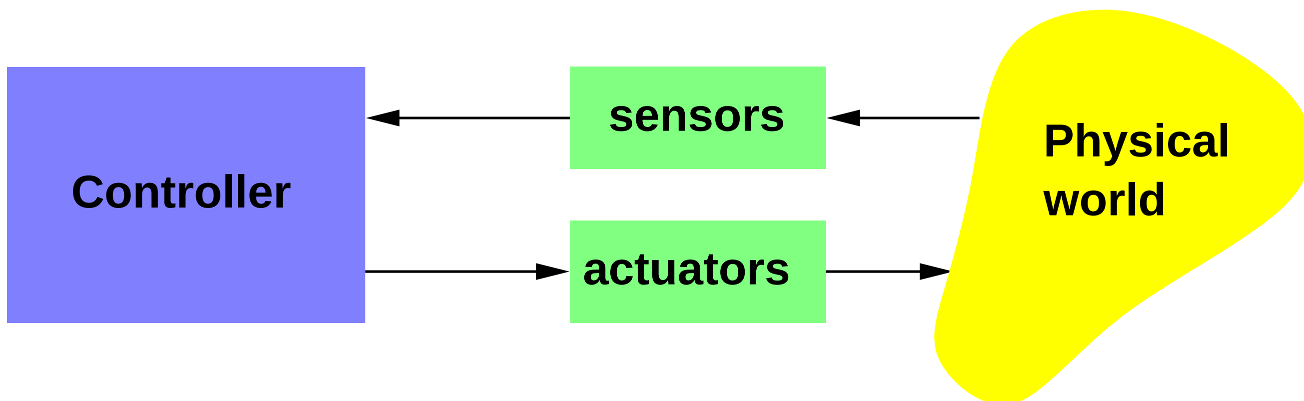
ECE 3140 / CS 3420
EMBEDDED SYSTEMS
LECTURE 14

Prof. José F. Martínez

TR 1:25-2:40pm in 150 Olin

Introduction to Real Time

Time-critical systems are the norm.



Tight interaction between sensing and actuation
$\Rightarrow$ need predictable timing of operations

We won't cover the details of *how* a system is controlled.

Introduction to Real Time

A computing system that is able to respond to events within *precise* timing constraints is a real-time system.

- Correct operation depends on
  - Usual properties (producing the correct output, etc)
  - Also on the *time* at which the output is produced

Some interesting observations:

- Time between different entities must be *synchronized*
  Note: time synchronization is not a simple problem
- Real time is *not* the same as fast!

Performance v/s Real Time

- Real time: have to *guarantee* timing properties
- Performance: minimize *average* response time

These are not the same!

Source of unpredictability:

- Architecture: cache, pipelining, ...
- Run-time system: scheduling, other tasks, ...
- Environment: Bursty information flow, extreme conditions, ...
- Input: no explicit notion of time in most languages

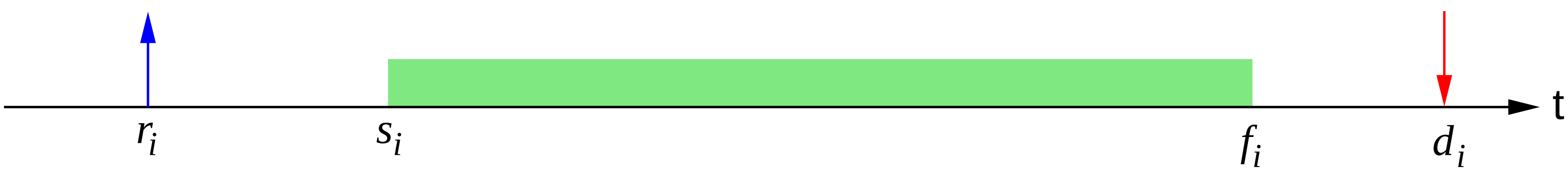
Real Time Systems

Terminology:

- A *job* is a sequence of operations that, in the absence of any other activities, is executed by the processor
- A *task* is a sequence (possibly infinite) of jobs
- Jobs have:
  - A request time r_i (arrival time)
  - A start time s_i
  - A finishing time f_i
  - An absolute deadline d_i

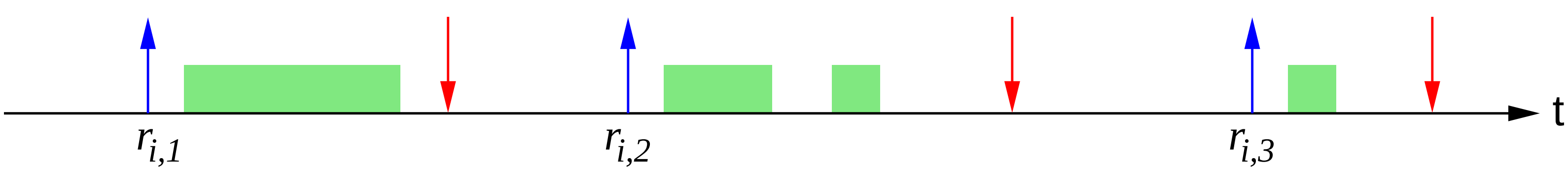
Tasks and Jobs

A single job:



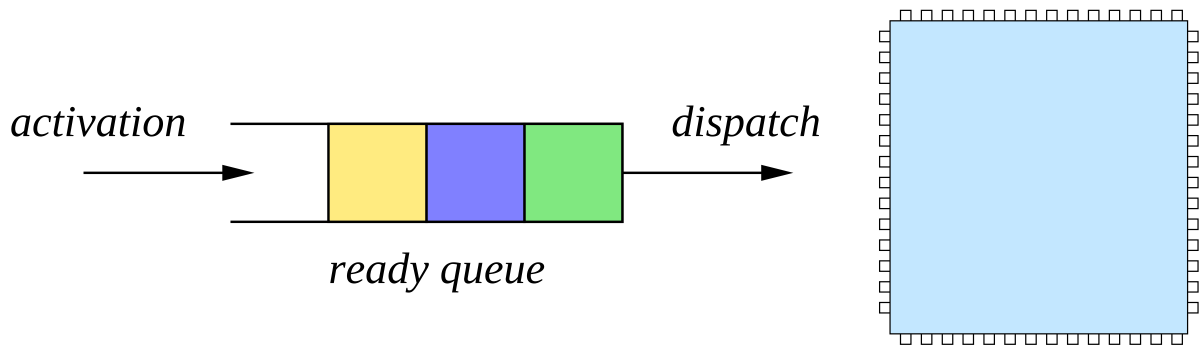
$f_i - s_i$ is the *worst case execution time* (wcet)

A task:



SCHEDULING

Scheduling algorithm: the strategy used to pick a ready task for execution.



Two categories:

- Preemptive: The running task can be temporarily suspended to execute another task
- Non-preemptive: The running task cannot be suspended until completion or until it is blocked

Scheduling

A schedule is a particular assignment of tasks to the processor.

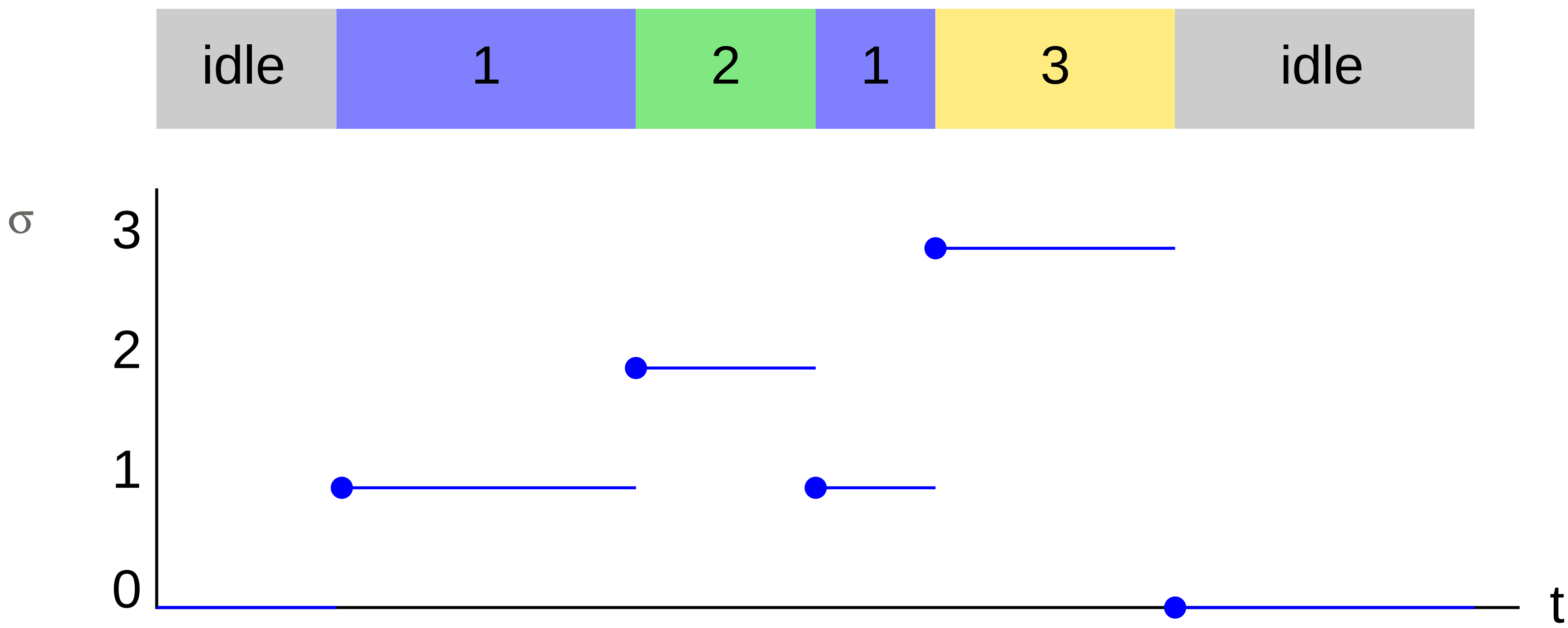
Given a set of tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$, a schedule is a mapping $\sigma : \mathbb{R}_{>0} \rightarrow \{0, 1, \ldots, n\}$ such that:

$$\sigma(t) = \begin{cases} k > 0 & \text{if } \tau_k \text{ is running} \\ 0 & \text{if the processor is idle} \end{cases}$$

and in any interval $[t_1, t_2) \in \mathbb{R}_{>0}$ $\sigma(t)$ can only change value a finite number of times.

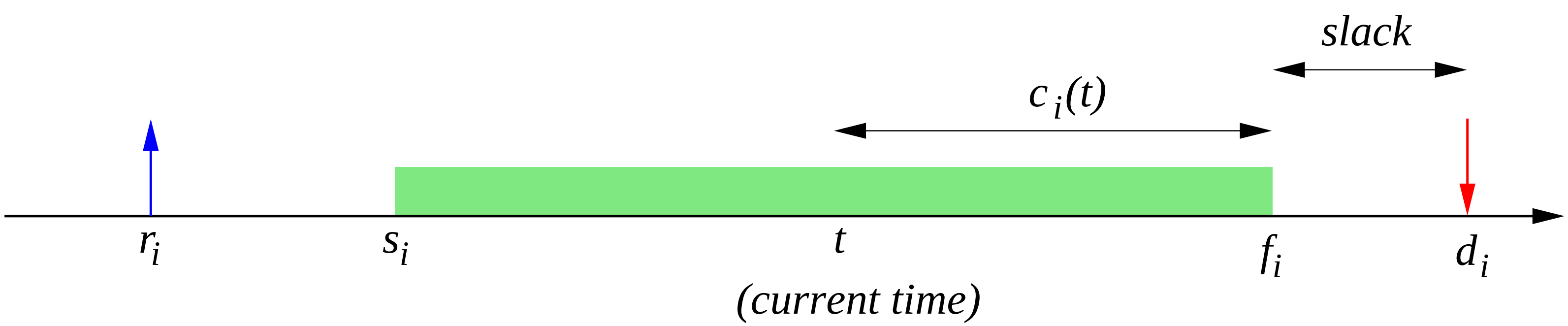
Scheduling

Example:



The points at which σ changes value is where a context switch occurs. Each interval $[t_i, t_{i+1})$ is a time slice.

Derived Parameters

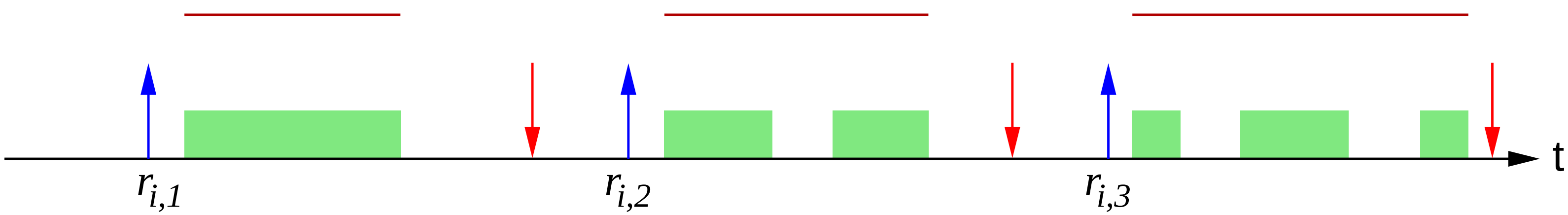


- Lateness: $L_i = f_i - d_i$
- Tardiness: $\max(0, L_i)$
- Residual wcet: $c_i(t)$
- Slack: $d_i - t - c_i(t)$

JITTER

Jitter is the time variation of a periodic event.

Example: completion-time jitter



$$\max_k(f_{i,k} - s_{i,k}) - \min_k(f_{i,k} - s_{i,k})$$

IMPORTANCE OF TASKS

- Hard tasks: All jobs must meet their deadlines. Missing a deadline has a catastrophic effect.
  - low-level control
  - sensor-actuator interactions for critical functions
- Soft tasks: Missing deadlines is undesirable, but only causes performance degradation
  - reading keyboard input
  - displaying a message
  - updating graphics

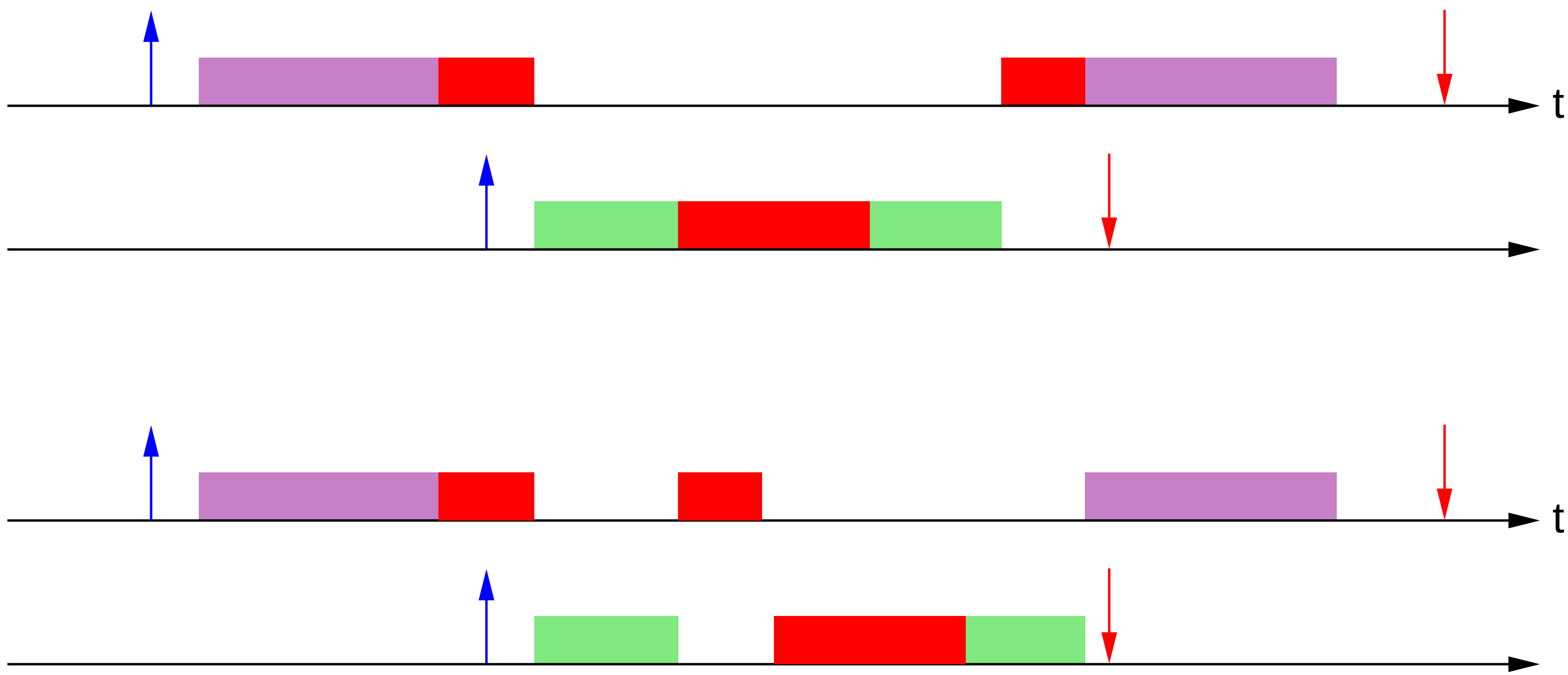
Tasks can be assigned *priorities*.

Tasks can be time-driven (periodic) or event-driven (aperiodic).

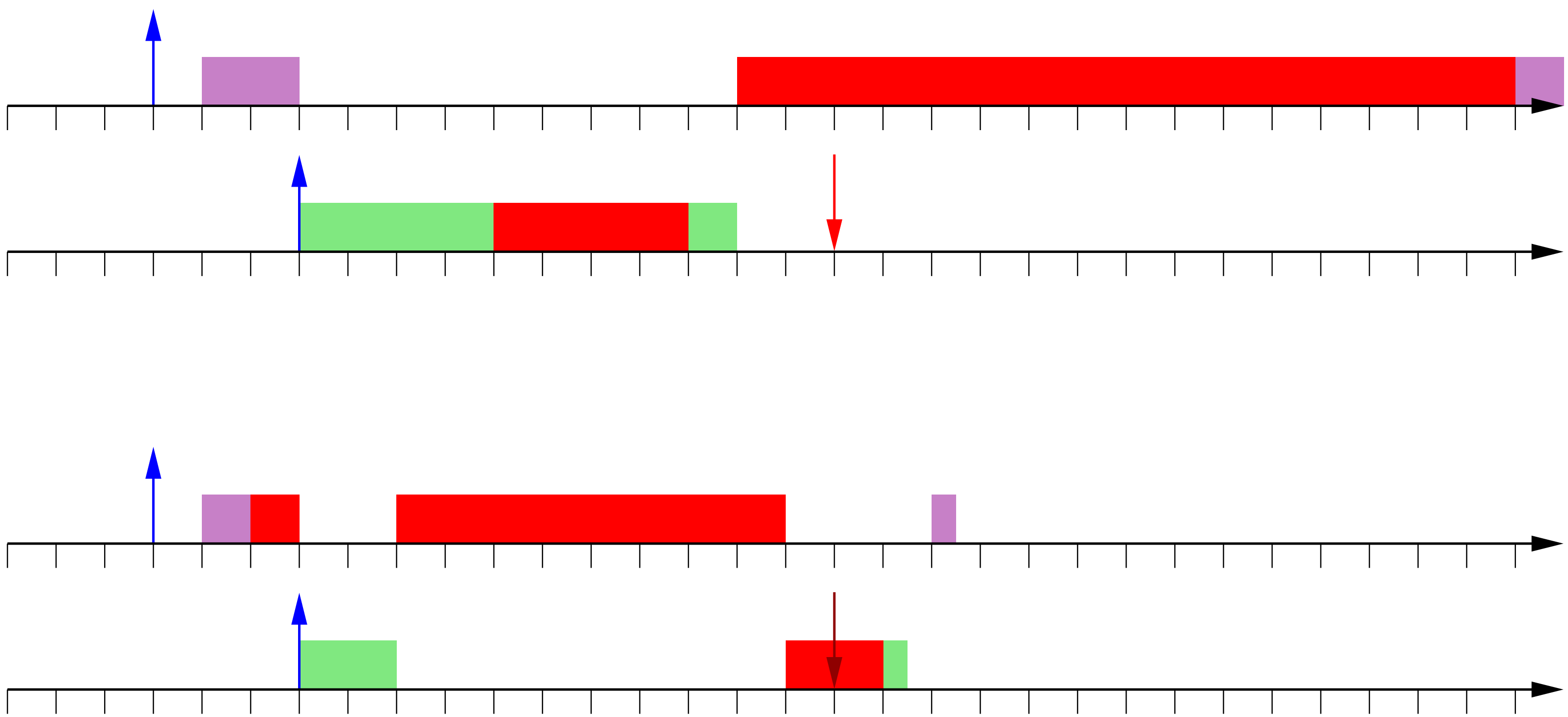
Resource Constraints

- Resources may be limited or even unavailable
- Shared resources may require mutual exclusion

$\Rightarrow$ all these introduce delays



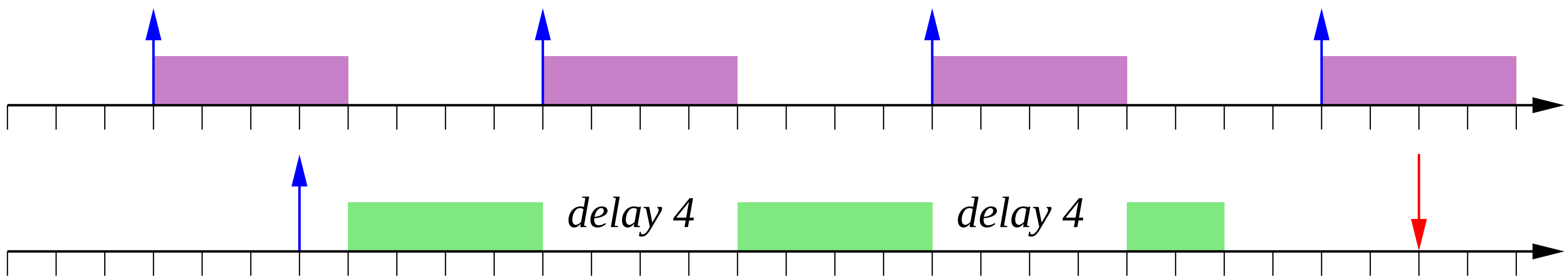
FASTER PROCESSOR

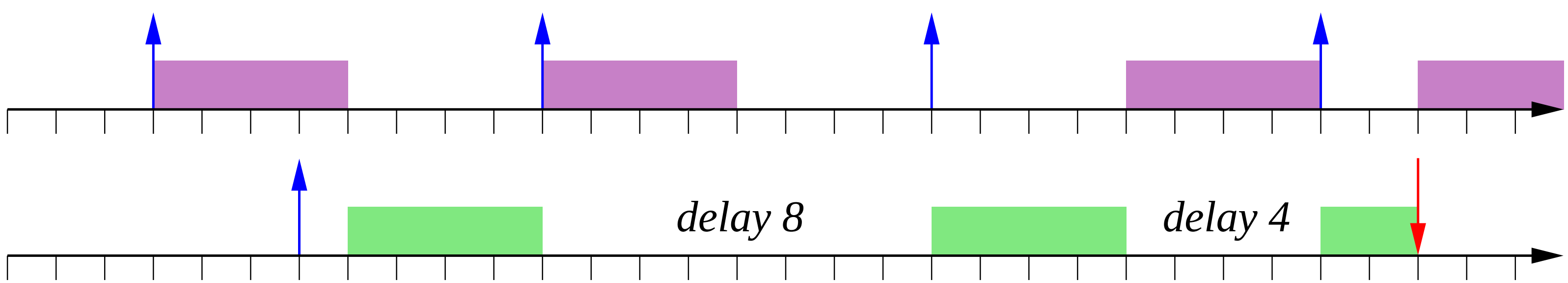


Having a faster processor doesn't automatically mean it is easier to meet deadlines.

DELAYS

Delays can cause other tasks to take longer!





Predictability

- Run-time system is responbile for ensuring predictable behavior
- Scheduling algorithm matters
- Concurrency control and resource sharing matters
- Interrupt handling must also be predictable

All sources of uncertainty must be minimized.

Scheduling

A schedule σ is *feasible* if all tasks are able to complete with their set of constraints.

A set of tasks Γ is set to be *schedulable* if a feasible schedule exists.

General problem: given Γ, a set of processors P, and a set of resources R, find an assignment of P and R that produces a feasible schedule.

Scheduling Algorithms

- Preemptive or nonpreemptive
- Static or dynamic: are the scheduling decisions based on parameters that change with time?
- Online or offline: are the decisions made apriori with knowledge of task activations, or are they taken at run time based on the set of active tasks?
- Optimal or heuristic: can you prove that the algorithm optimizes a certain criteria or not?