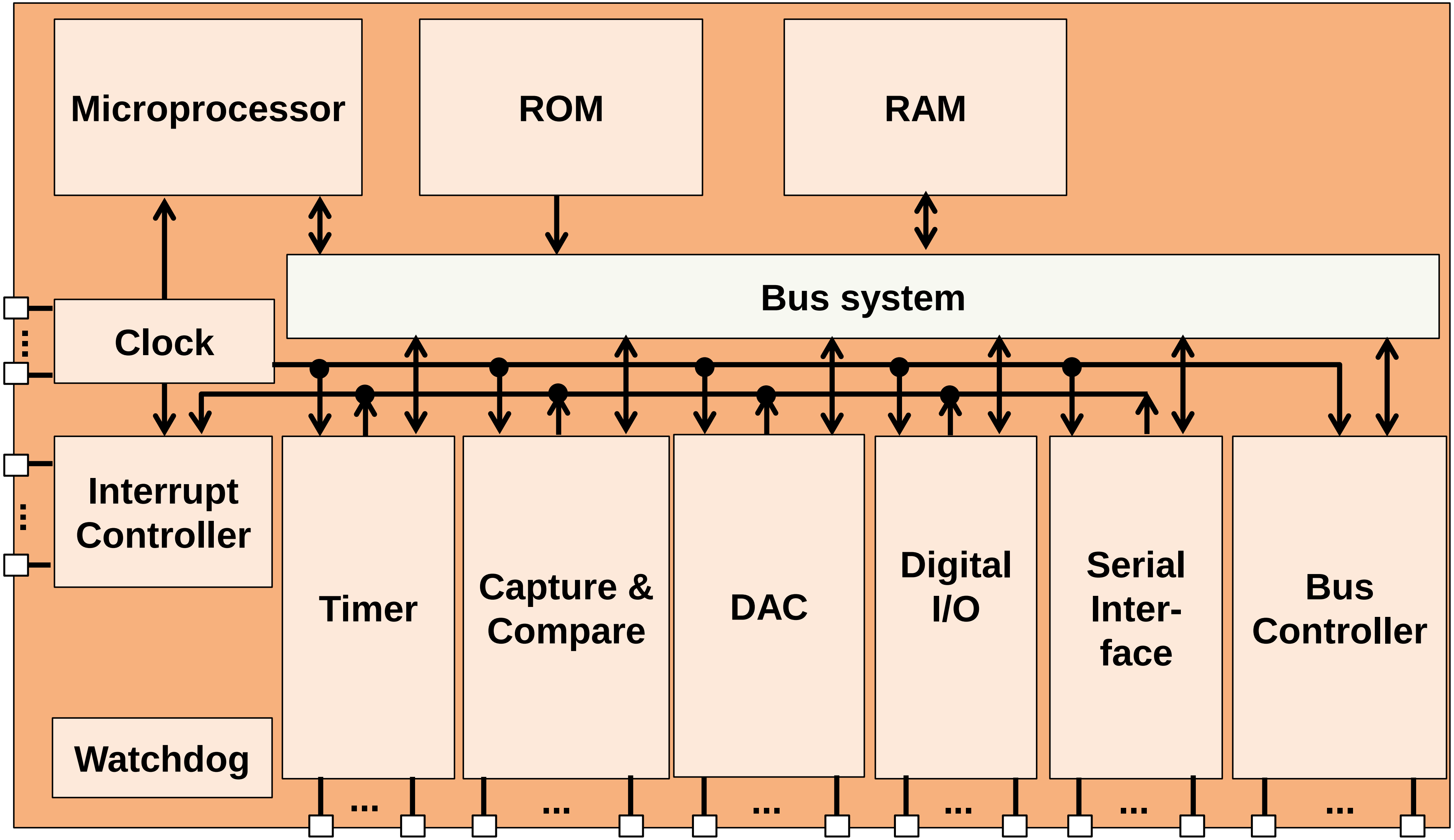
# Software Platforms for Automotive Systems

## Lecture 4:OS and Real-Time Behavior

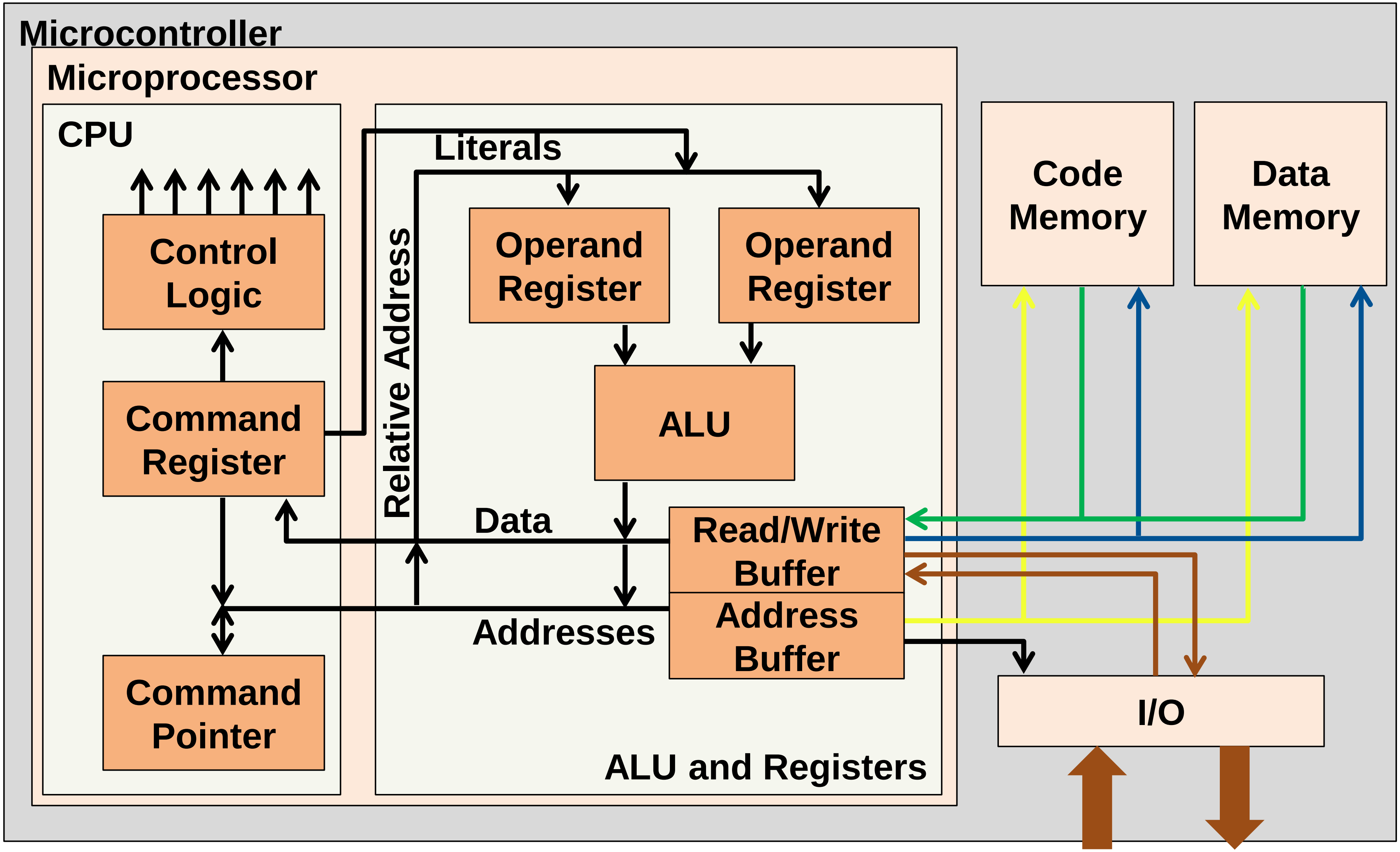
**Alejandro Masrur**

**5th November 2015, TU Chemnitz**

# Microcontroller: Typical Structure



# Microprocessor: Typical Structure




TECHNISCHE UNIVERSITÄT CHEMNITZ

# AUTOSAR/OSEK OS

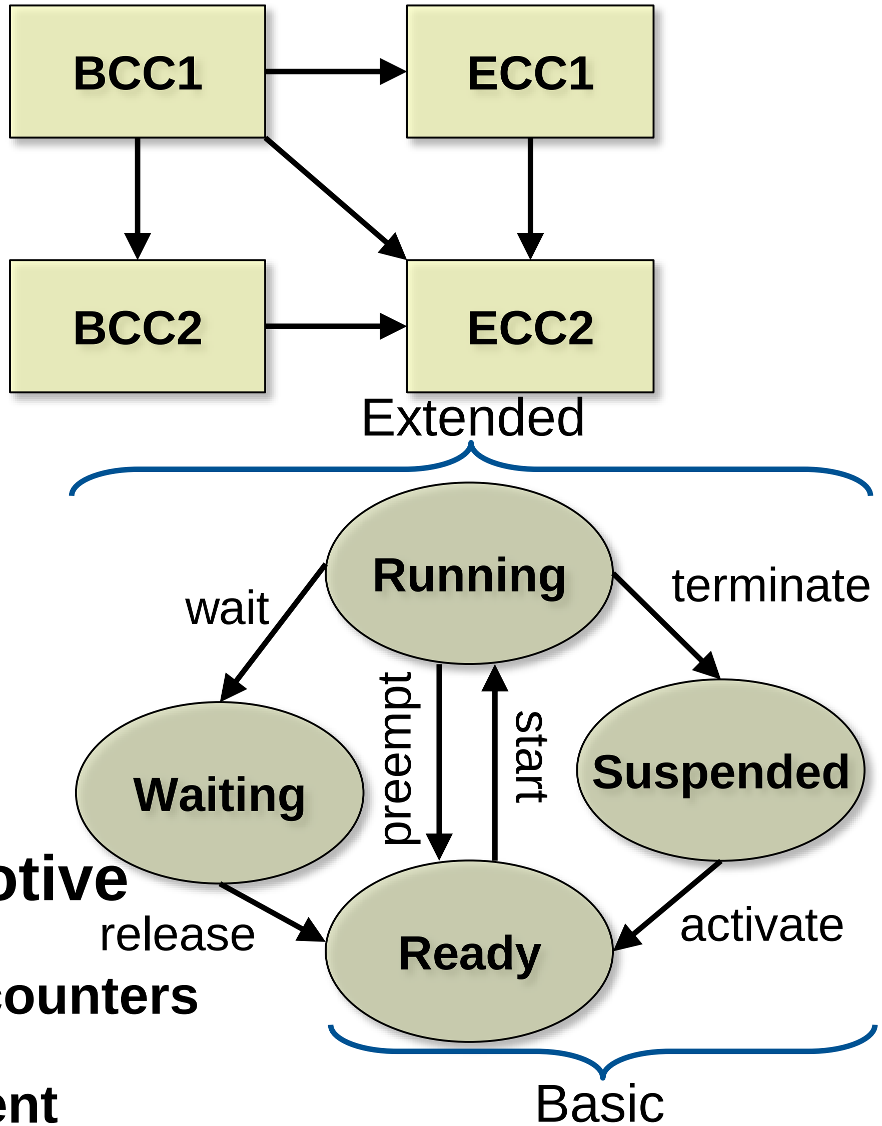
- **Configurable and scalable**
  - **Conformance classes**

| Property | BCC1 | BCC2 | ECC1 | ECC2 |
|---|---|---|---|---|
| Multiple activation | no | yes | BT:no ET:no | BT:yes ET:no |
| # tasks | 8 | | 16 | |
| Tasks/ priority | =1 | >1 | =1 | >1 |
| Events/ task | - | | 8 | |
| # priorities | 8 | | 16 | |

- **Predictable behavior**
  - **Real-time scheduling**
  - **Fixed priorities**
  - **Priority inheritance**
  - **Priority ceiling**
- **Special features for automotive**
  - **Single and cyclic alarms on timers**
    - **E.g. for engine management**

# AUTOSAR Operating System

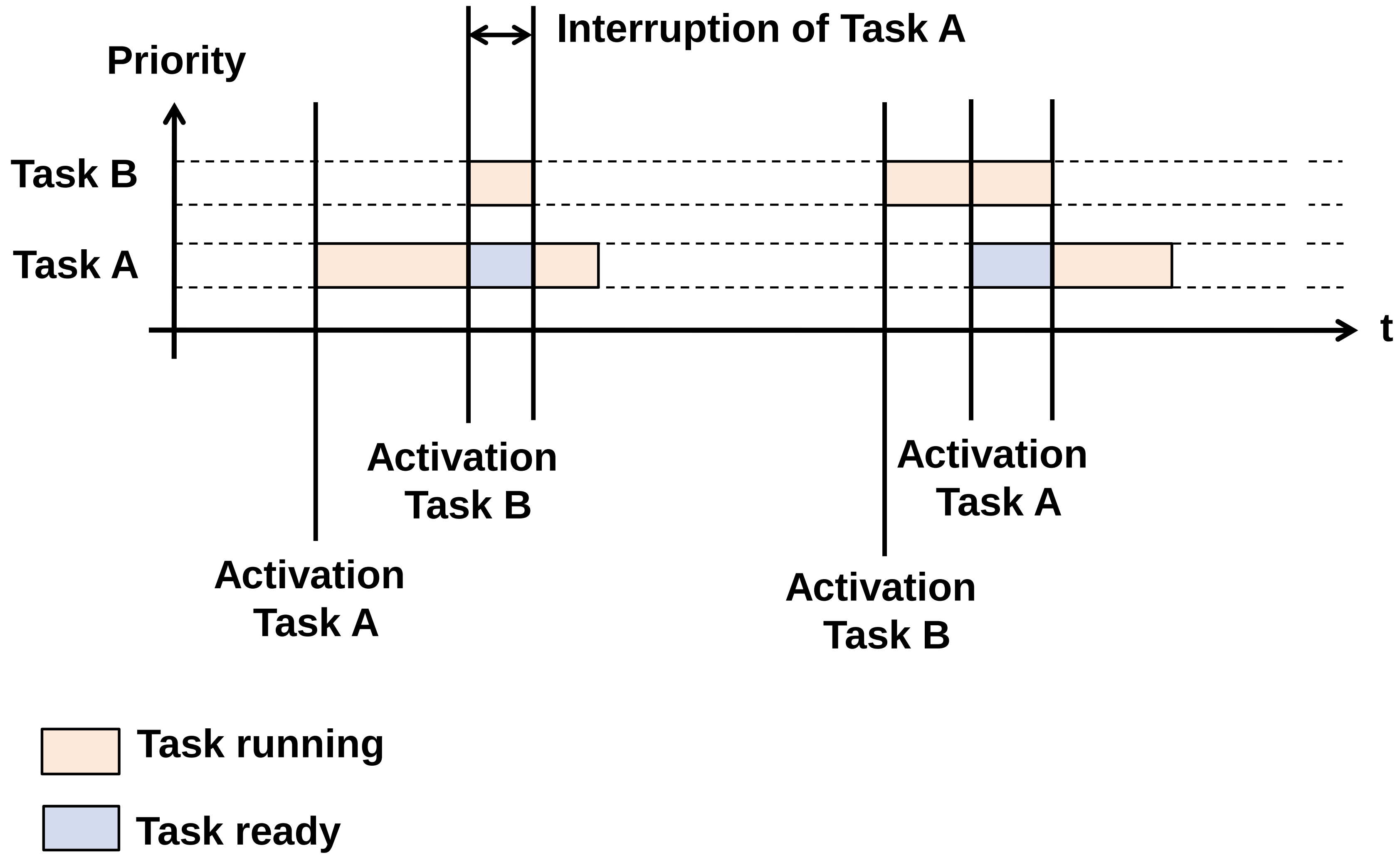
- **Configurable and scalable**
  - **Conformance classes**
- **Predictable behavior**
  - **Real-time scheduling**
  - **Fixed priorities**
  - **Priority inheritance**
  - **Priority ceiling**
- **Special features for automotive**
  - **Single and cyclic alarms on counters**
    - **E.g. for engine management**



# Real-Time Behavior

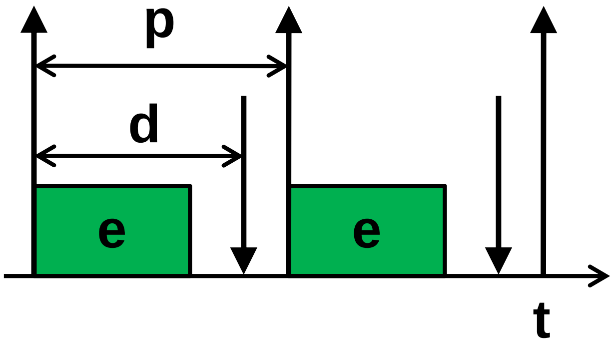
- Tasks need to execute on time
- Tasks are associated with deadlines
  - Need for computing Worst-Case Execution Time (WCET)
- Deadline misses may cause damage
  - Human lives in danger = hard real-time
    - Brakes in a car, steer by wire, etc.
  - Quality of service goes down = soft real-time
    - Multimedia systems, MMI, etc.
- Need for schedulability/feasibility analysis
  - Requires an appropriate task model

# Preemptive Scheduling



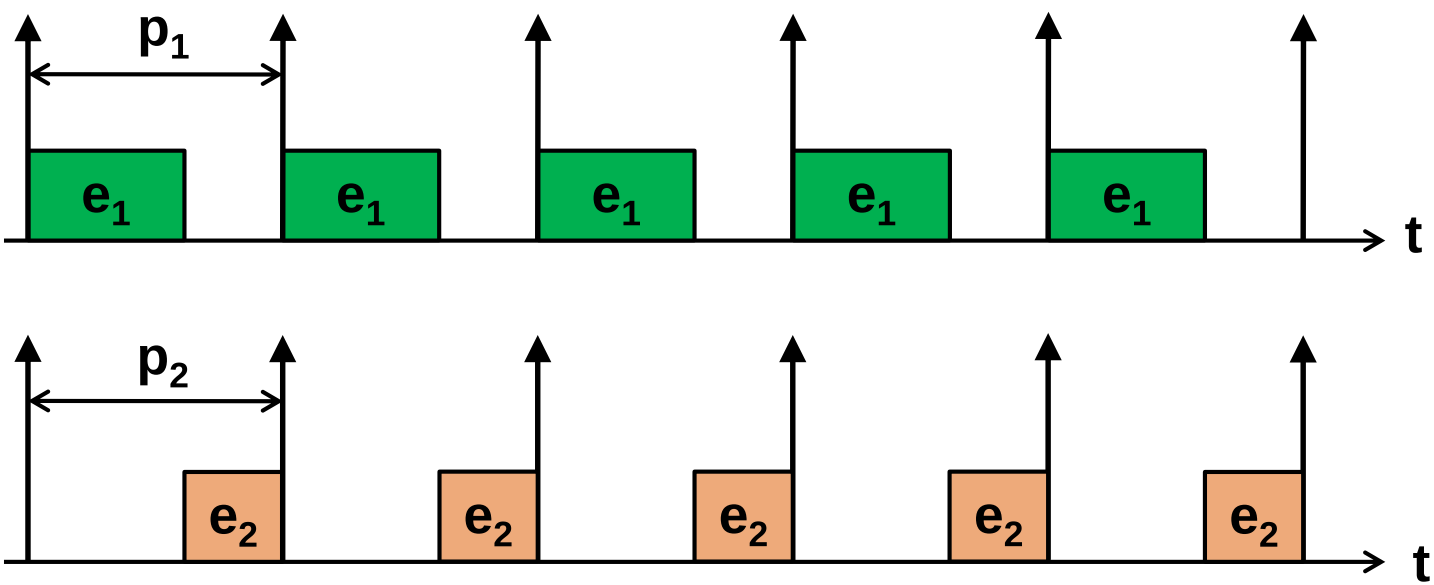
# Schedulability with Preemptions

- **Model for real-time tasks**
  - **Periodic/sporadic with inter-release time: p**
  - **Relative deadline: d**
  - **Worst-case execution time: e**
- **The case d =p is more usual**
- **The case of d ≤ p is harder to handle**
- **Scheduling algorithm**
  - **OS supports fixed priorities**
  - **Rate Monotonic (RM)**
  - **Deadline Monotonic (DM)**



# Schedulability with Preemptions

- **Utilization bound on one processor:** $U = \sum_{\forall T_i} \frac{e_i}{p_i} \leq 1$
  - **This is necessary but not sufficient test**




  - **$e_1+e_2=p_1=p_2 \rightarrow U=1$**
  - **$e_1+\varepsilon$ or $e_1+\varepsilon \rightarrow$ Deadline miss**

# Schedulability with Preemptions

- Worst-case response time (WCRT) analysis
  - Necessary and sufficient
  - "Critical instant": jobs of all tasks are released simultaneously

$$t^{(j+1)} = e_k + \sum_{\forall T_i \in HP(k)} \left\lceil \frac{t^{(j)}}{p_i} \right\rceil \cdot e_i$$

$HP(k)$ = Set of tasks with higher priority than the k-th task

  - Compute iteratively until: $t^{(j+1)} = t^{(j)}$
- This is the WCRT of the k-th task: $w_k$
- Task set is feasible, if for all possible k: $w_k \le d_k$

# Example (1/3)

- A two-task set should be tested under DM
  - $T_1$: $e_1$=1.5ms, $p_1$=3ms, $d_1$=2.5ms
  - $T_2$: $e_2$=1.2ms, $p_2$=5ms, $d_2$=2ms
- Utilization: $U = \frac{1.5}{3} + \frac{1.2}{5} = 0.74$
  - 0.74<1 → first check is OK
- WCRT for highest priority tasks (i.e., $T_2$)

$$t^{(0)} = e_2 = 1.2$$

$$t^{(1)} = 1.2 + \sum_{\substack{\forall\, T_i \in \\ HP(2)}} \left\lceil \frac{1.2}{p_i} \right\rceil \cdot e_i = 1.2 \;\text{ - HP(2) is empty}$$

# Example (2/3)

- $t^{(1)} = t^{(0)} \rightarrow$ stop $\rightarrow w_2 = t^{(1)} = 1.2\text{ms}$
- $1.2\text{ms} \leq 2\text{ms} \rightarrow$ deadline can be met

- WCRT for next highest priority (i.e.,$T_1$)

$$t^{(0)} = e_2 + e_1 = 2.7$$

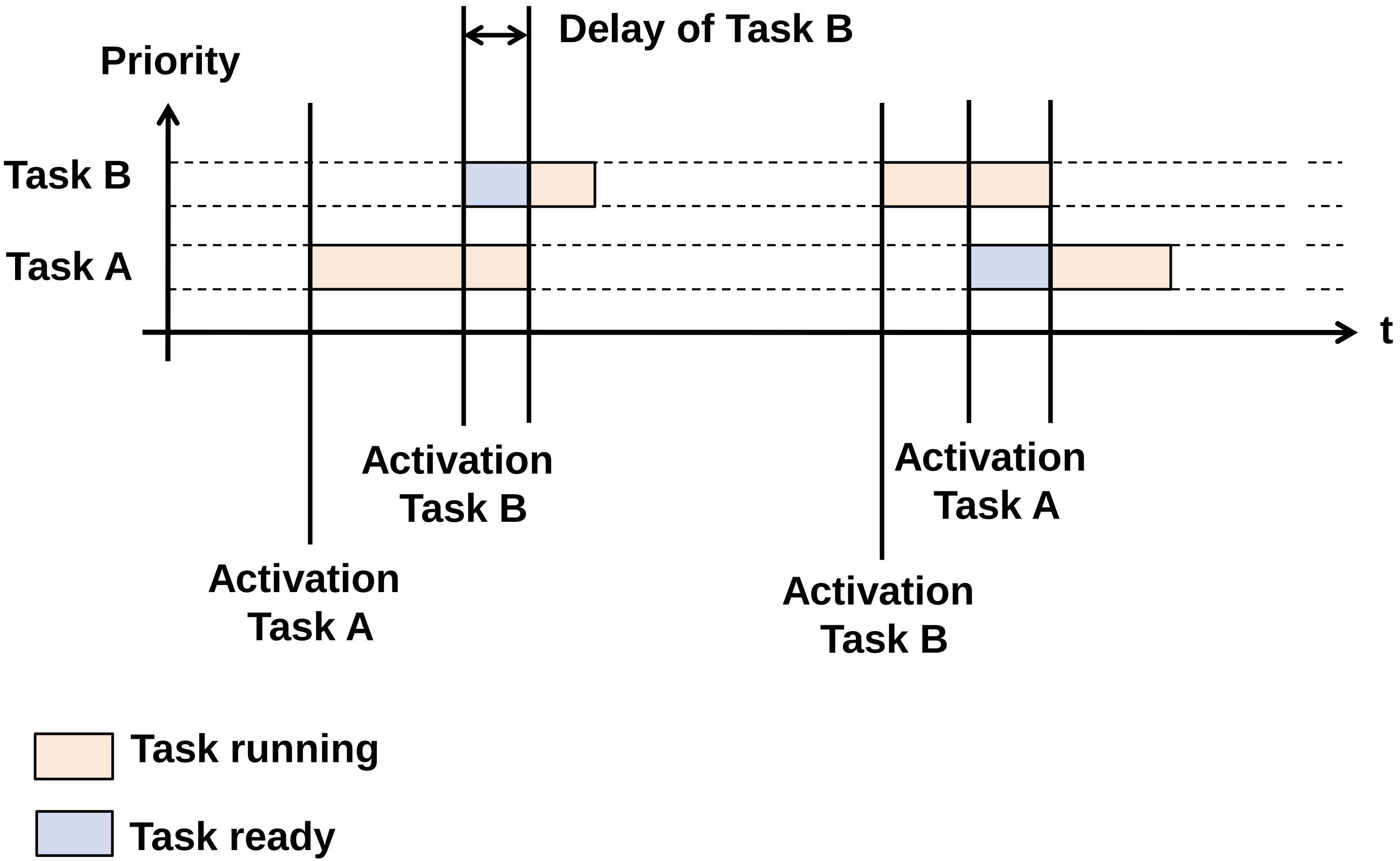
$$t^{(1)} = 1.5 + \sum_{\substack{\forall\ T_i \in \\ HP(1)}} \left\lceil \frac{2.7}{p_i} \right\rceil \cdot e_i = 1.5 + \left\lceil \frac{2.7}{5} \right\rceil \cdot 1.2 = 2.7$$

- HP(1) contains $T_2$
- $t^{(1)} = t^{(0)} \rightarrow$ stop $\rightarrow w_1 = t^{(1)} = 2.7\text{ms}$
- 2.7ms > 2.5ms → deadline cannot be met!

# Example (3/3)

- What can we do to make $T_1$ meet its deadline?
  - Increase its deadline from 2.5ms to 2.7ms
    - When can I do this in practice?
      - $T_1$ should allow it. If increasing $T_1$'s deadline leads to malfunctioning, then we need to find another solution.
  - Decrease the execution time of $T_1$ or $T_2$
    - Either optimizing algorithm or using a faster processor
  - Change $T_1$'s priority
    - This might lead to another task missing its deadline
      - In our example, if $T_1$ gets higher priority than $T_2$ then $T_2$ will miss its deadline.

# Non-Preemptive Scheduling



# Schedulability without Preemptions

- Almost the same task model as before
  - Periodic/sporadic with inter-release time: p
  - Relative deadline: d
  - Worst-case execution time: e
- In addition, we model the “blocking time”: b
  - This is the maximum time a given task can be “blocked” due to a lower-priority task running
- The schedulability analysis becomes more complex
  - With preemptions we compute the WCRT of the first job of each task in the critical instant; without preemptions we need to compute the WCRT of all jobs in the busy interval

# Schedulability without Preemptions

- Utilization bound is still necessary: $U = \sum_{\forall T_i} \frac{e_i}{p_i} \leq 1$
  - But not a sufficient test
- Worst-case response time (WCRT) analysis
  - Necessary and sufficient
  - “Busy interval”: no idle time starting from critical instant
    - This can be determined in the following manner:

$$t^{(j+1)} = \sum_{\forall T_i} \left\lceil \frac{t^{(j)}}{p_i} \right\rceil \cdot e_i$$

    - Until $t^{(j+1)} = t^{(j)}$ → busy interval

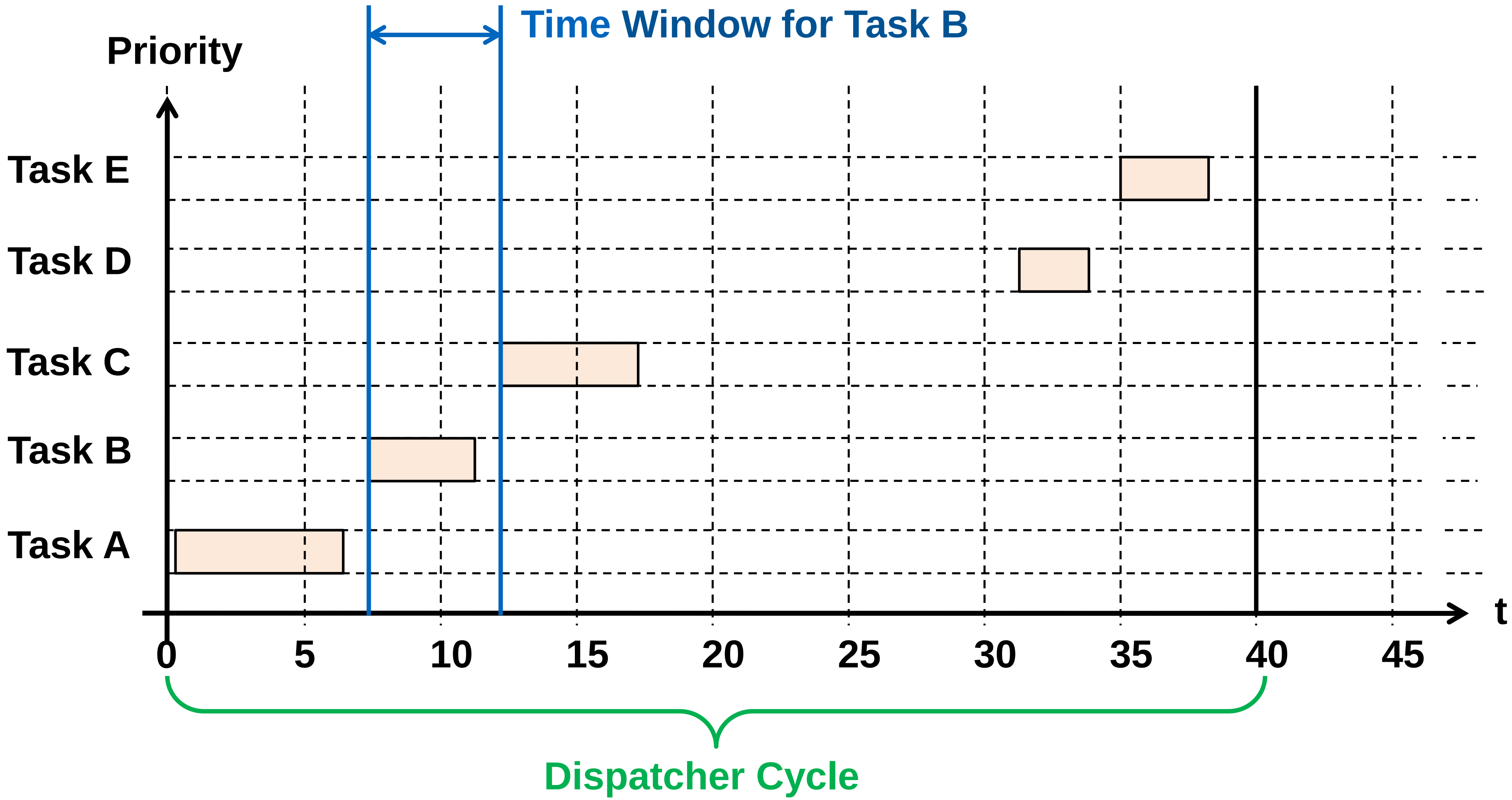
# Schedulability with Preemptions

- **For each job of the k-th task in the busy interval**
  - **We need to compute the WCRT**
  - **Indexing k-th task's jobs by 1 ≤ l ≤ α, we have**

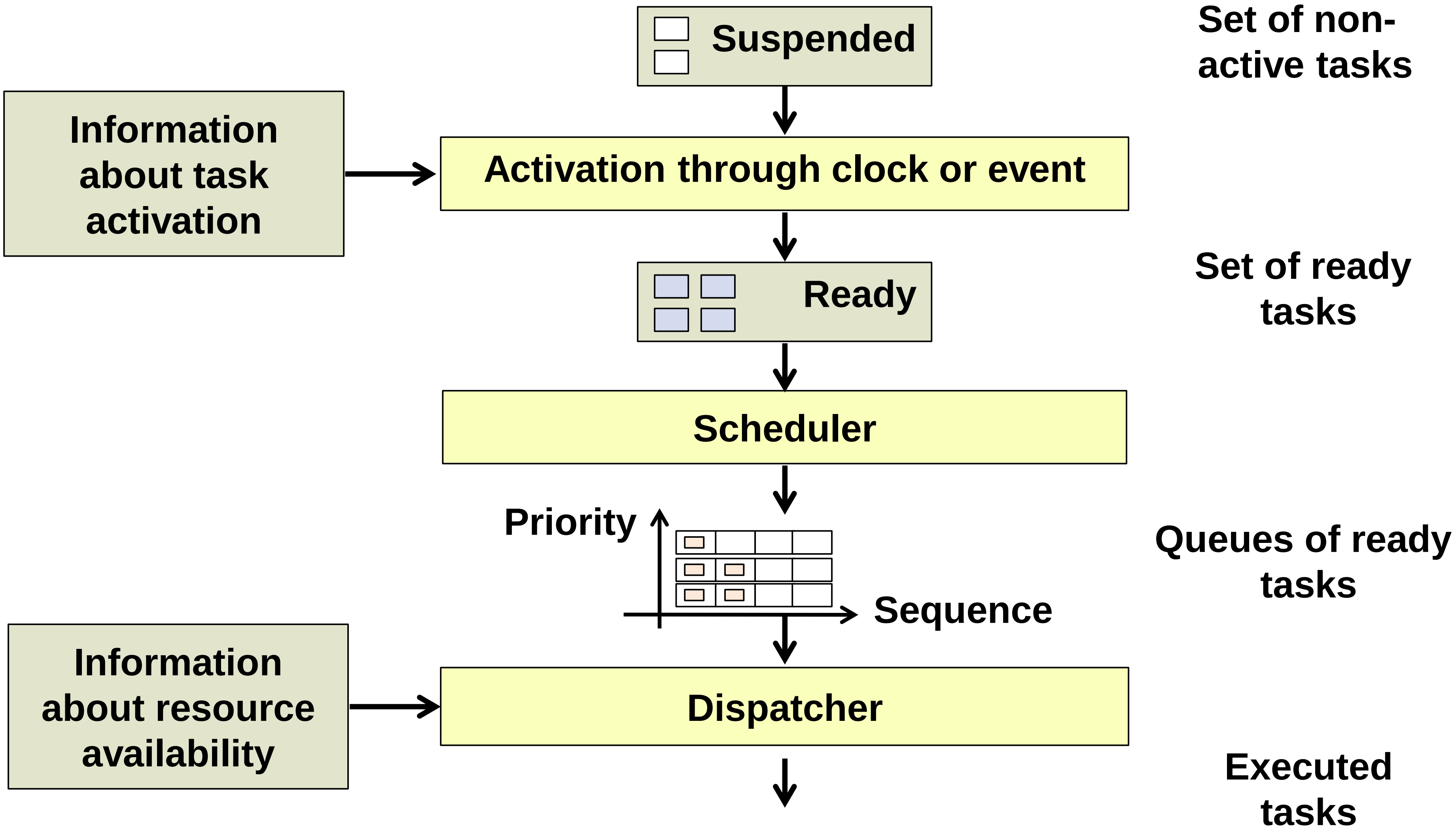
$$t^{(j+1)} = b_k + l \cdot e_k + \sum_{\forall T_i \in HP(k)} \left\lceil \frac{t^{(j)}}{p_i} \right\rceil \cdot e_i \quad \left\{ \begin{array}{l} b_k = \max\limits_{\forall T_i \in LP(k)} (e_i) \\ LP(k) = \text{Set of tasks with lower priority than the k-th task} \end{array} \right.$$

  - **Compute iteratively until:** $t^{(j+1)} = t^{(j)}$
- **This is the WCRT of the l-th job of k-th task:** $w_{k,l}$
- **Feasible, if for all possible k and l:** $w_{k,l} \leq (l-1) \cdot p_k + d_k$

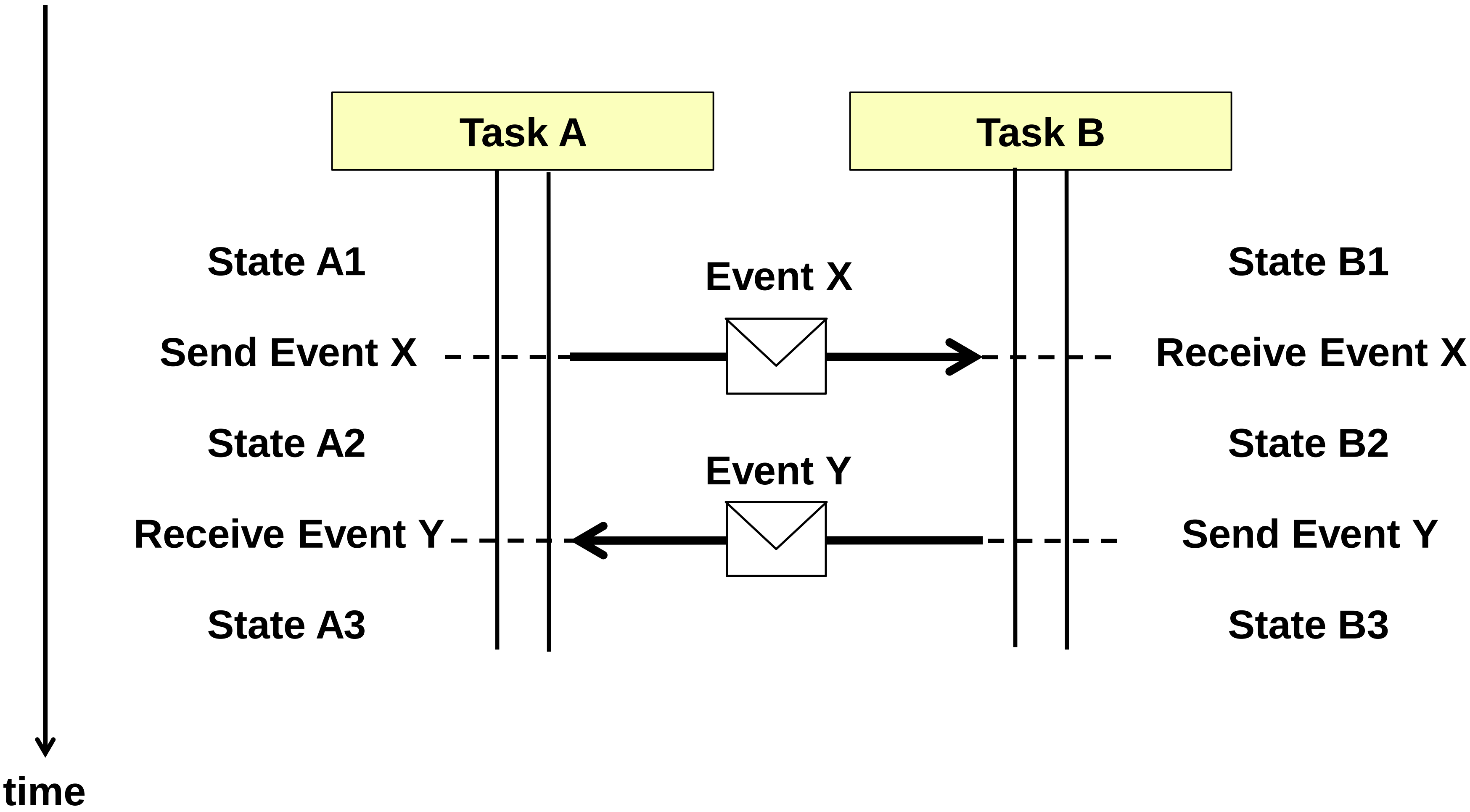
# Time-Triggered Scheduling



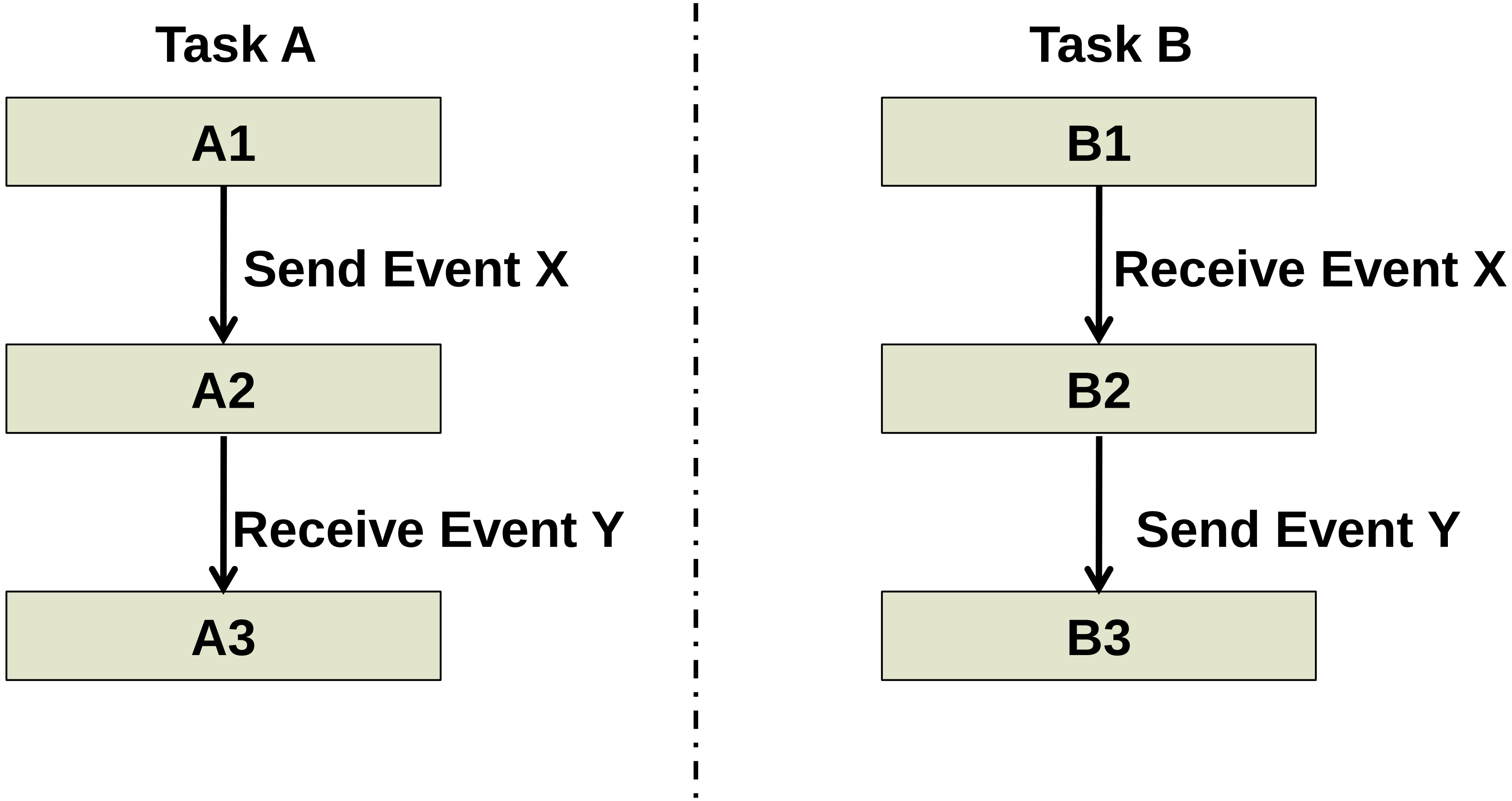
# Scheduling Process



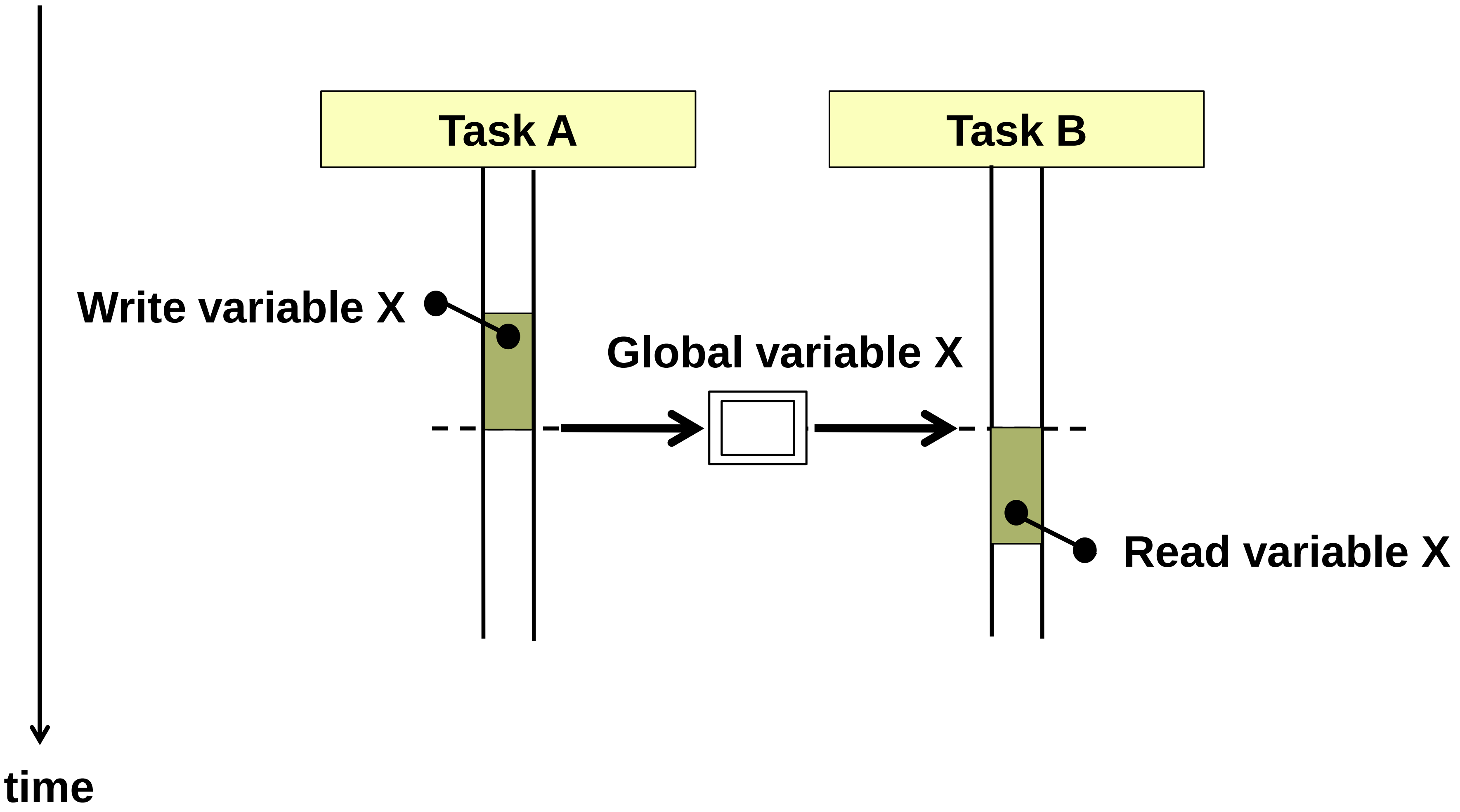
# Task Synchronization



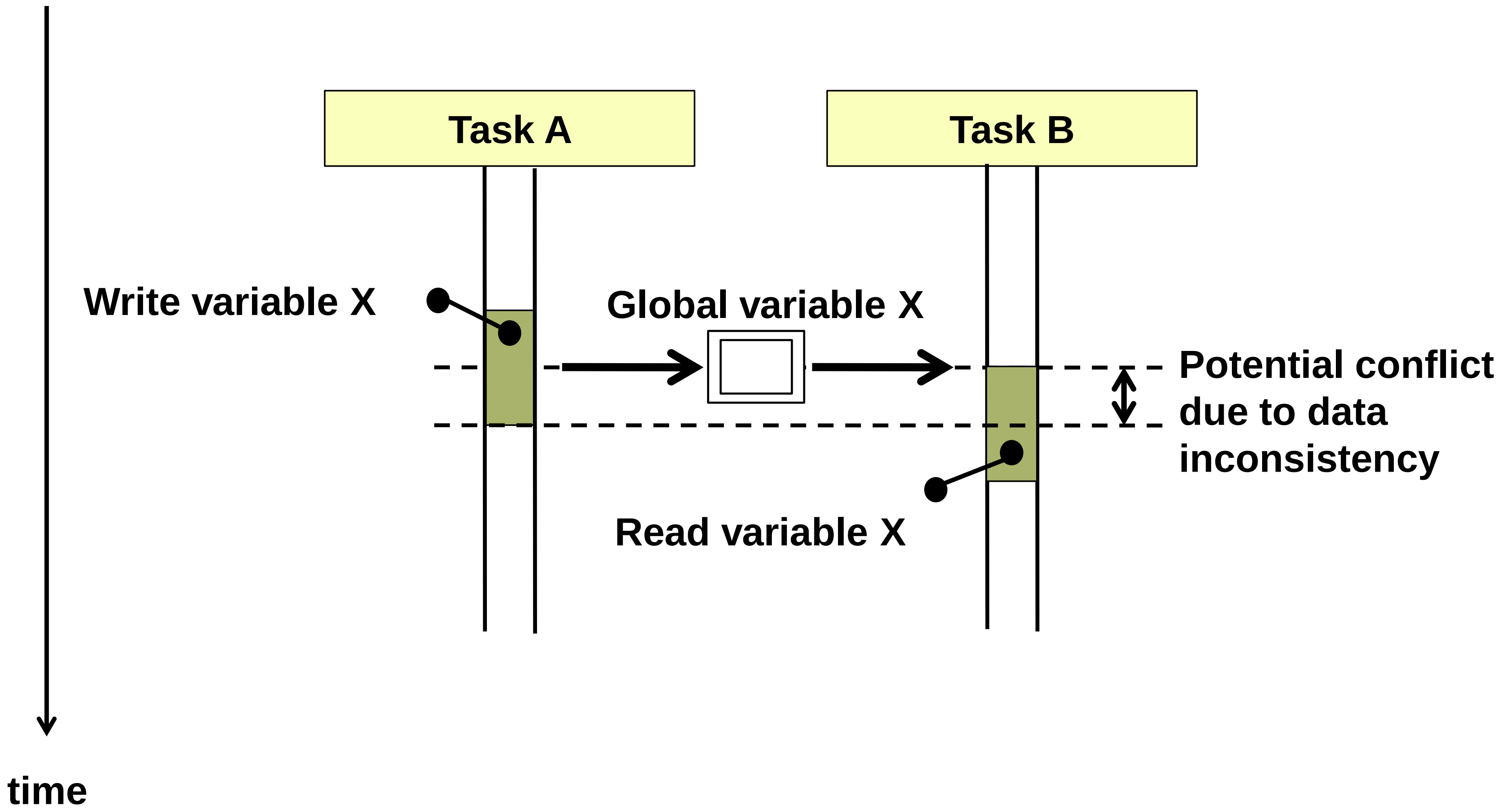
# Tasks‘ State Flow



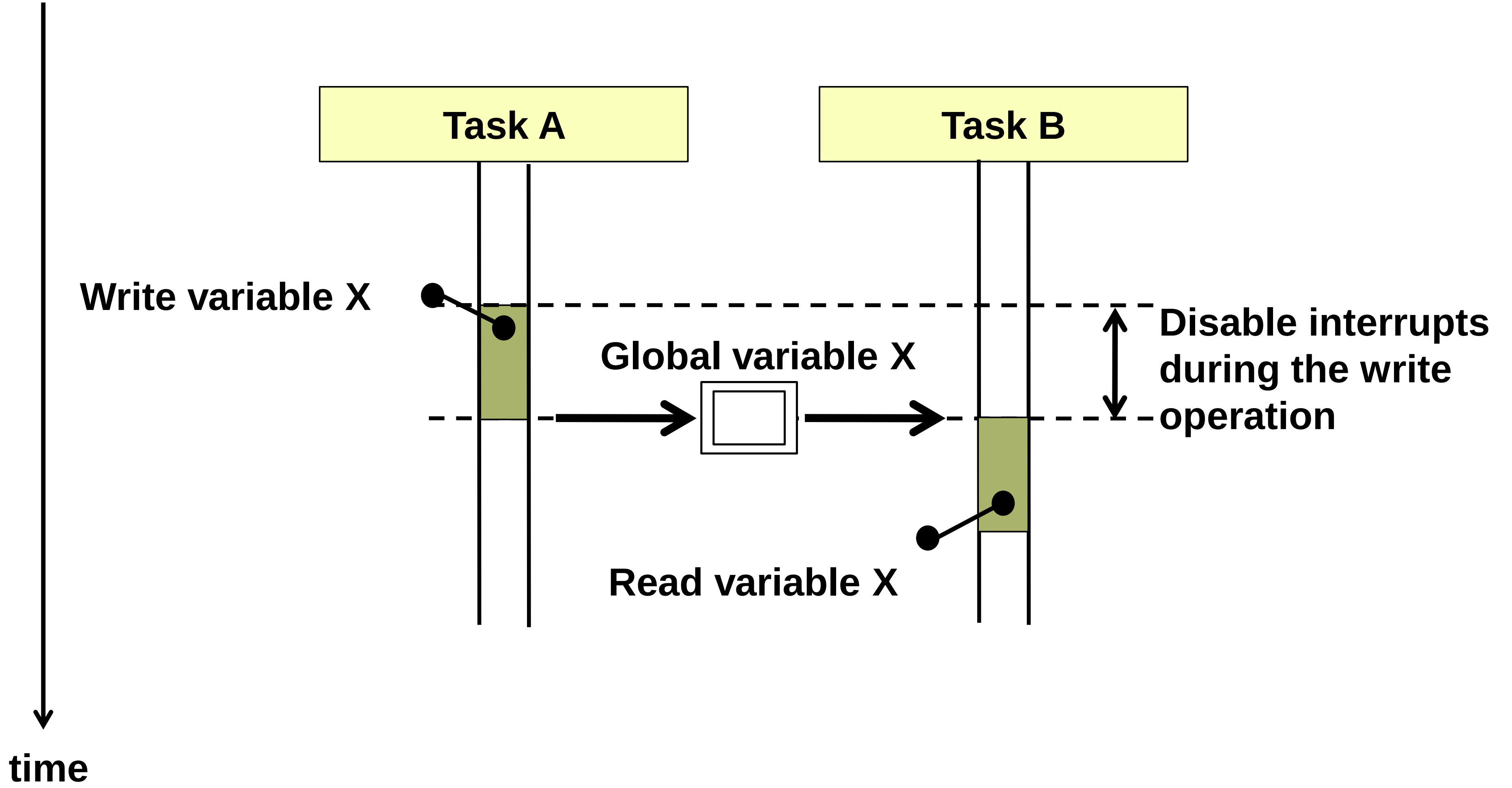
# Synchronization: Global Variable



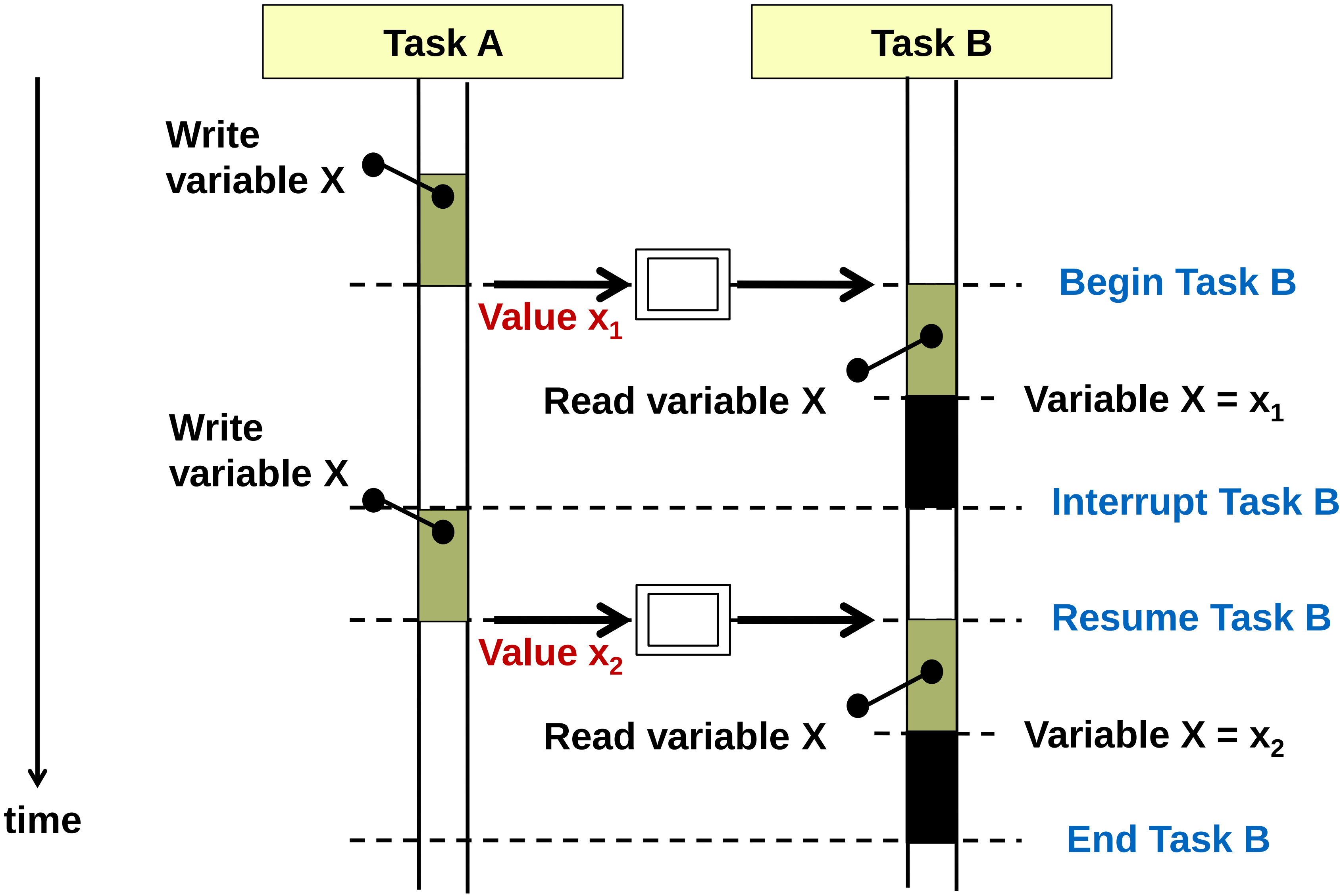
# Potential Problem



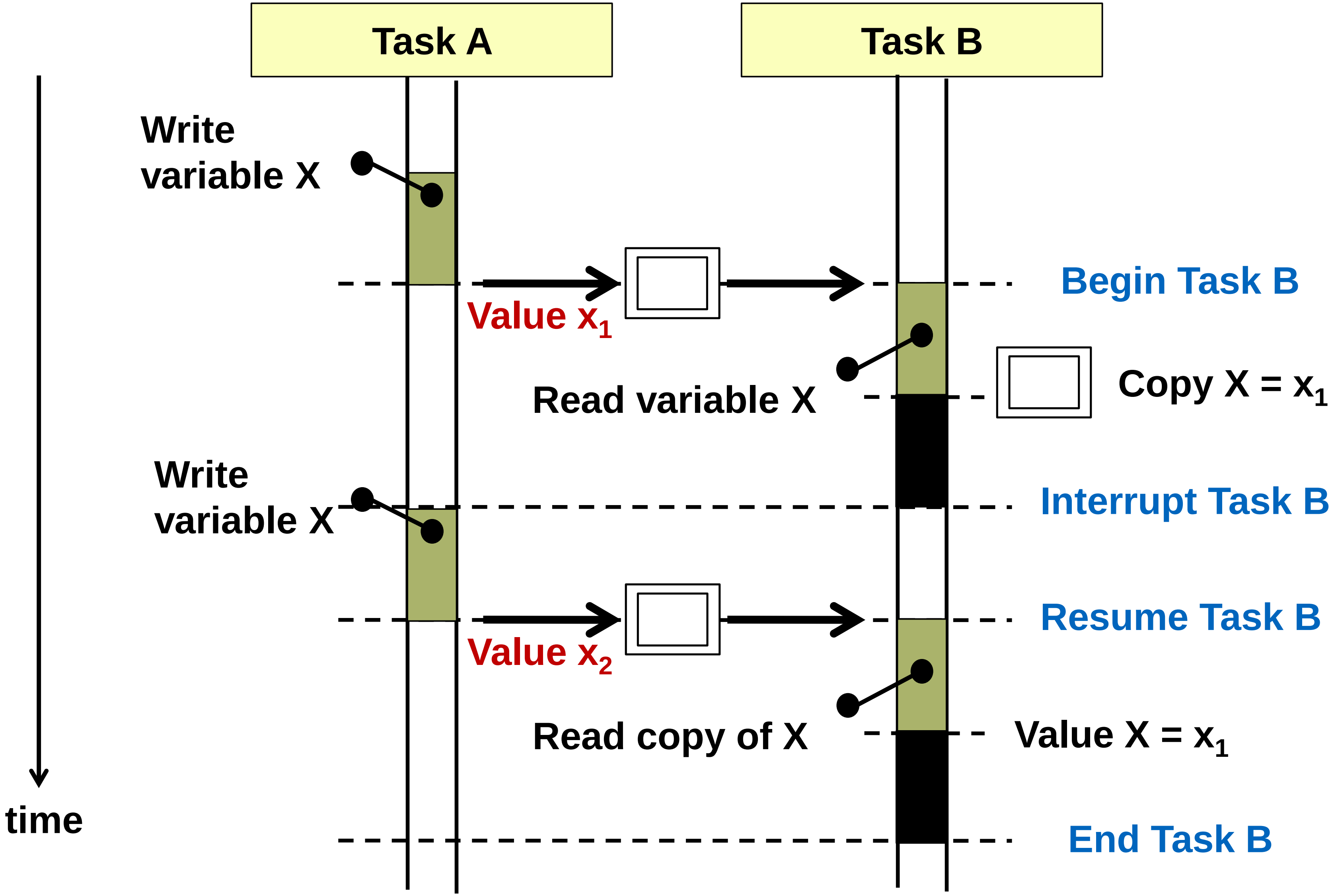
# Solution to that Problem



# Another Problem



# Solution: Local Copy



# Summary

- **Microcontroller and Microprocessor structure**
  - **Need for OS support to deal with hardware**
- **OS needs to provide real-time behavior**
  - **Fixed-priority scheduling algorithm**
    - **Need to perform a schedulability test**
  - **Time-triggered scheduling**
    - **Easier to analyze but less flexible**
  - **Mechanism for synchronizing tasks**
    - **Global variables are commonly used**
    - **There might be problems with data consistency**