CPE 746-Embedded Real-Time Systems- Fall 06

Types of RTS

Done By :
Mohammed Al Afifi
Mohammed Ismail

Supervised By: Dr. Lo'ai Tawalbeh

Computer Engineering Department
Jordan University of Science and Technology

Introduction

- The world has thousands of applications running on millions of micro-controllers that utilize simple executives or “bare metal” applications. These applications typically are built around one of three configurations.
- First, the super-loop foreground/background processing system is the smallest and simplest design.
- Second, the co-operative multi-tasking system, requiring tricky synchronization planning.
- Third, the pre-emptive multi-tasking system is the most complex.

Multi-Tasking

- Multi-tasking refers to the ability of a system to execute more than one task at the same time.

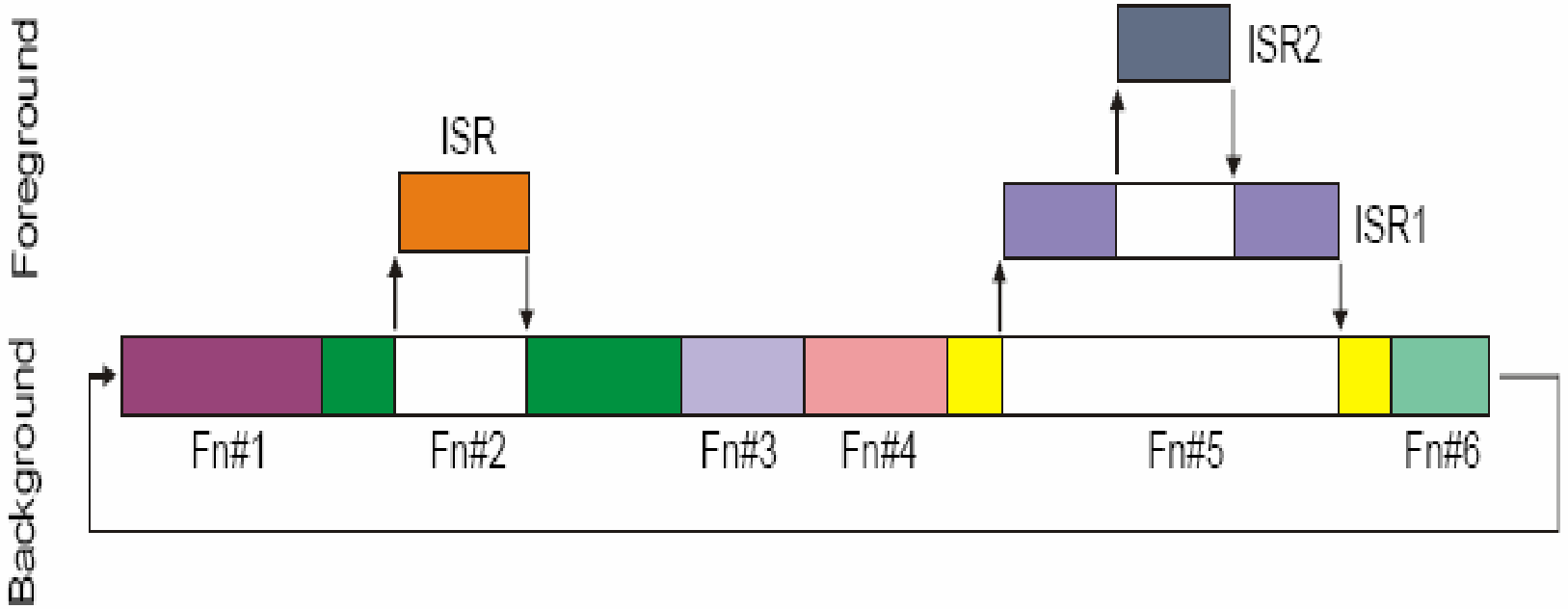
Preemption

- Preemption is defined as the act of a higher-priority process taking control of the processor from a lower-priority task.

Foreground/Background Systems

- Small, simple systems usually don't have an OS
- Instead, an application consists of an infinite loop that calls modules (functions) to perform various actions in the "**Background**".
- Interrupt Service Routines (ISRs) handle asynchronous events in the "**Foreground**"
- Foreground = **interrupt level**
- Background = **task level**

Foreground/Background System

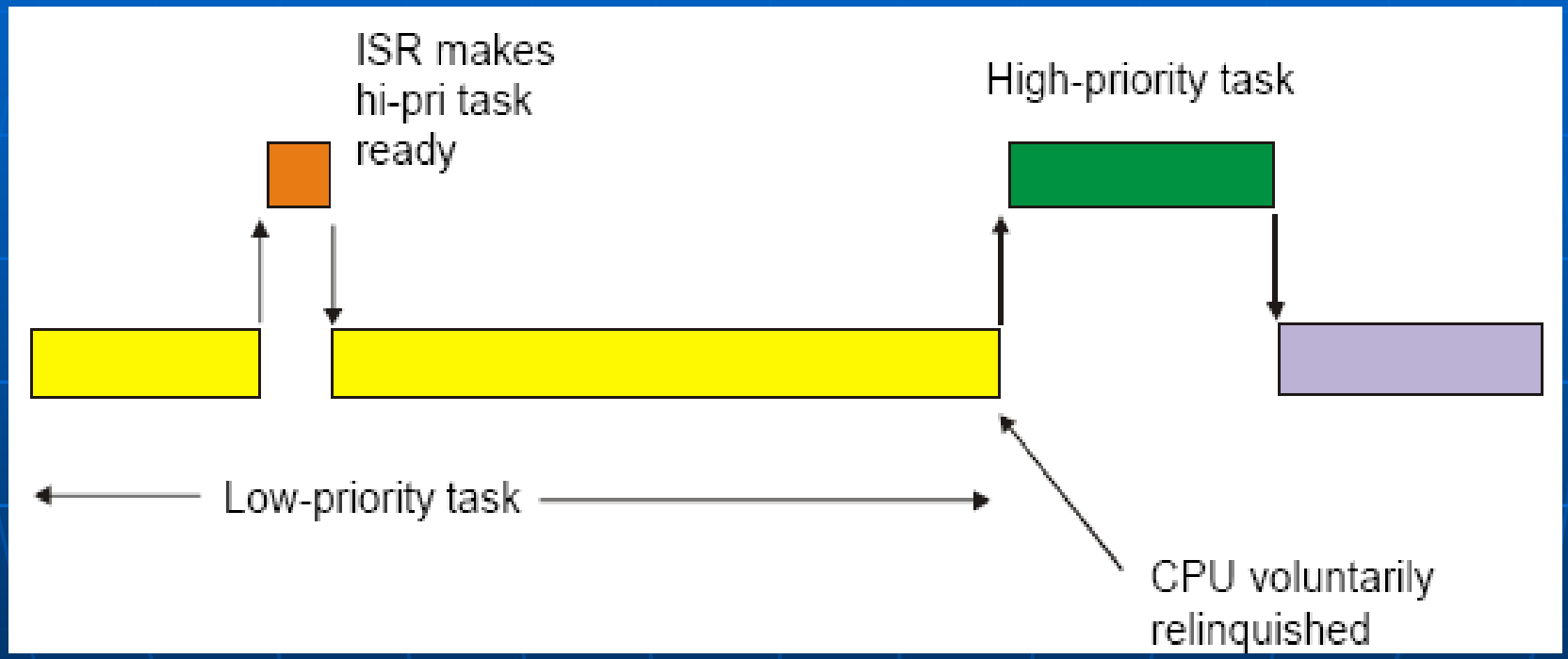


- Foreground - relies on hardware based scheduling
- Priority scheduling for interrupts is often provided by micro-controllers.
- Background: "main loop"

Non-Preemptive “Cooperative” Multitasking

- Each task can control the CPU for as long as it needs it.
- The task currently controlling the CPU must offer control to other tasks.
- Called cooperative because all tasks must cooperate for it to work. If one task acts like a selfish and self-centric person and does not cooperate, it can hog the CPU.
- *Cooperative multitasking* has the advantage of making the operating system design much simpler, but it also makes it less stable because a poorly designed application may not cooperate well, and this often causes system freezes

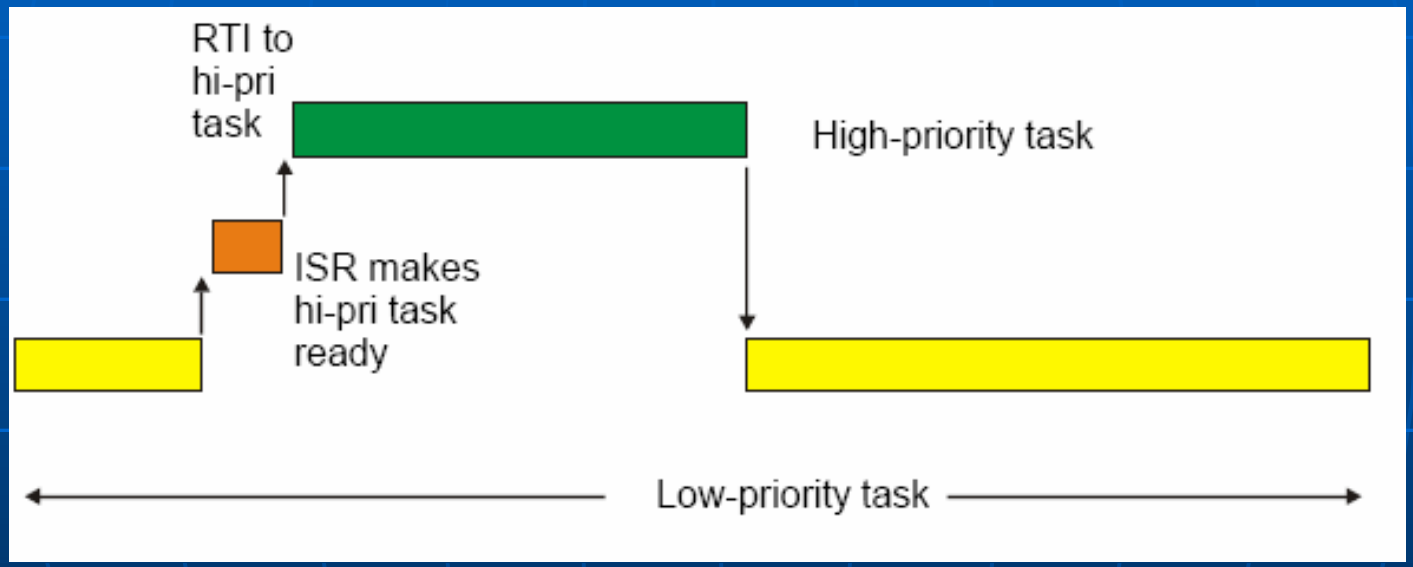
non-preemptive Kernel



Preemptive Multitasking

- The operating system allocates the CPU time slices to each task.
- Preemptive multitasking forces tasks to share the CPU whether they want to or not.

Pre-emptive kernel

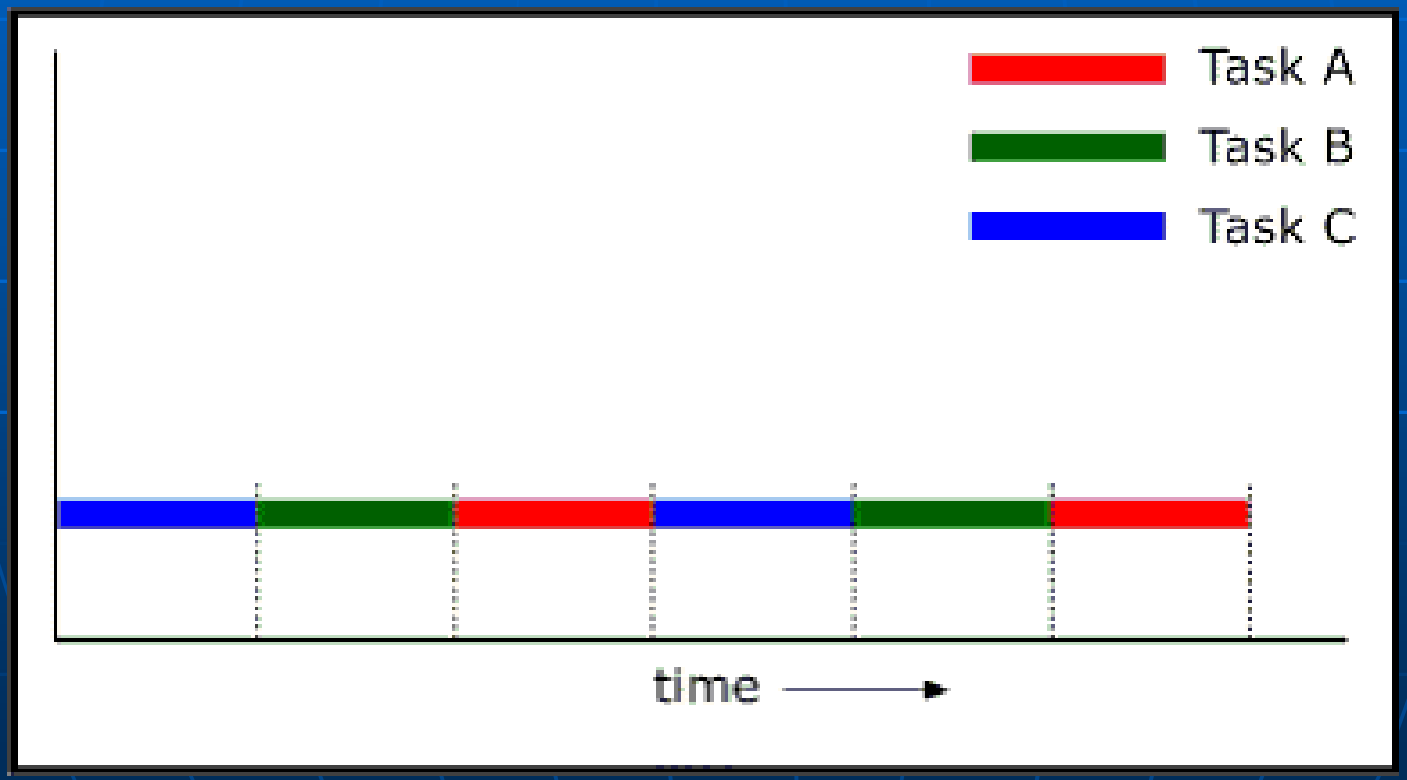


Scheduling

- It is the process of determining which task runs when in a multi-tasking system is referred to as CPU scheduling or plain scheduling.
- **Scheduling algorithm** : The algorithm followed to decide who gets next turn on CPU.
- The program that does this is called the **Scheduler**.

Scheduling Algorithms

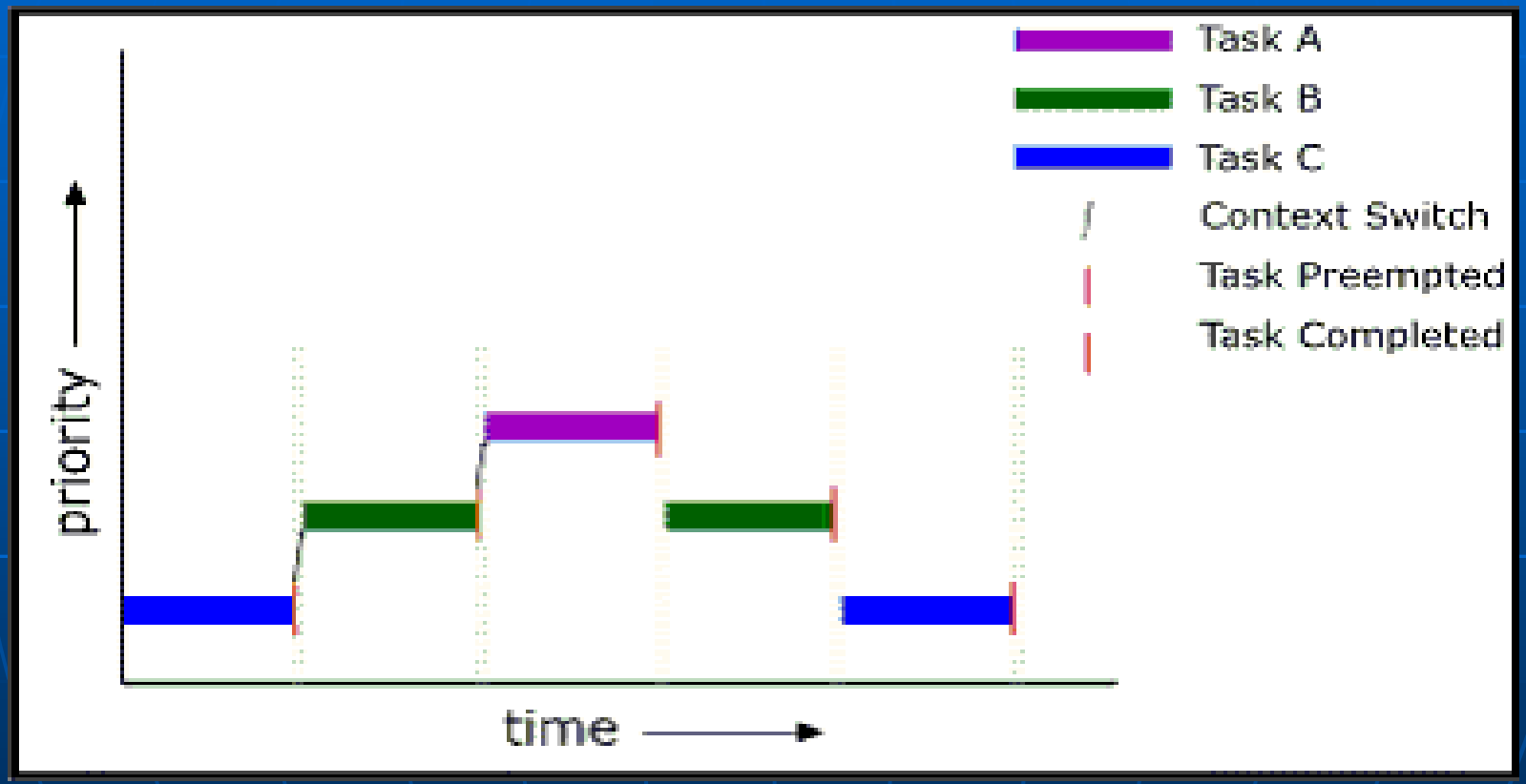
- Round-Robin Scheduling



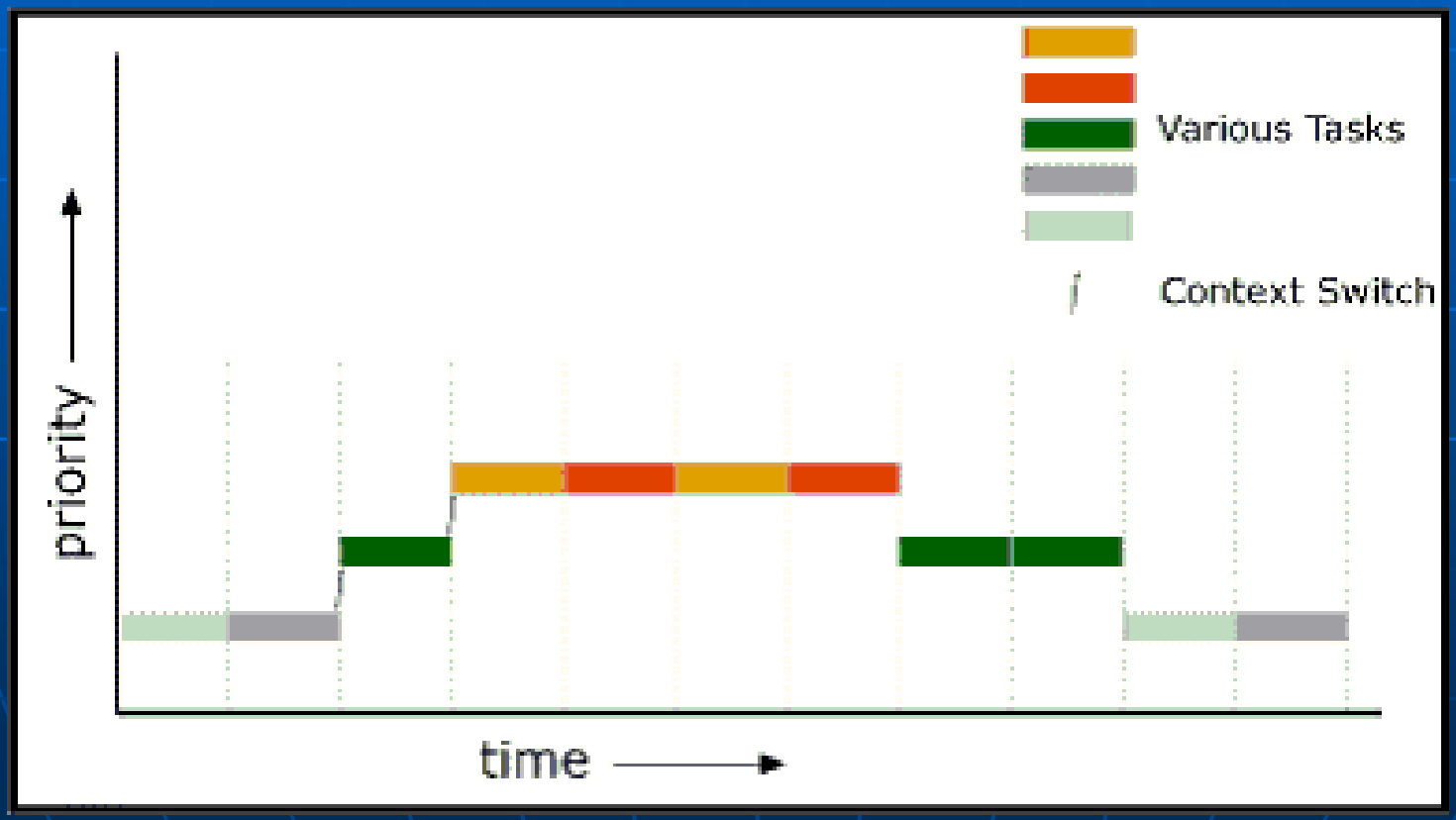
Priority Based Pre-emptive Scheduling

- Most RTOSs today control the execution of application software tasks by using priority based pre-emptive scheduling.
- In this approach, software developers assign a numeric “priority” value to each task in their application software.
- The RTOS’s task scheduler will allow tasks to run, and will switch among the tasks.
- the highest priority task that is ready to run, should always be the task that is actually running.
- if a relatively low priority task is running and a higher priority task becomes ready – the scheduler must immediately stop the low priority task (even in mid-execution) and allow the higher priority task to begin to run immediately
- When the higher priority task is done, the low priority task is allowed to continue running – from the point at which it was stopped.

- Priority Scheduling



- **Fixed-Priority Preemptive Round-Robin Scheduling**



Priority Based Pre-emptive Scheduling Drawbacks

- low priority tasks may suffer “starvation”.
- there is no way to tell a priority-based preemptive scheduler about software deadlines.

Alternatives to priority-based pre-emptive scheduling

- **Deadline Scheduling:**
  - In this approach, the RTOS kernel's task scheduler is provided with information about task deadlines.
  - temporarily raises the priorities of tasks as they approach their deadlines if they have not yet run
  - In this way, the deadline scheduler gets the tasks to run before they miss their deadlines, by preemptively “borrowing” running time from tasks that normally have higher priorities.

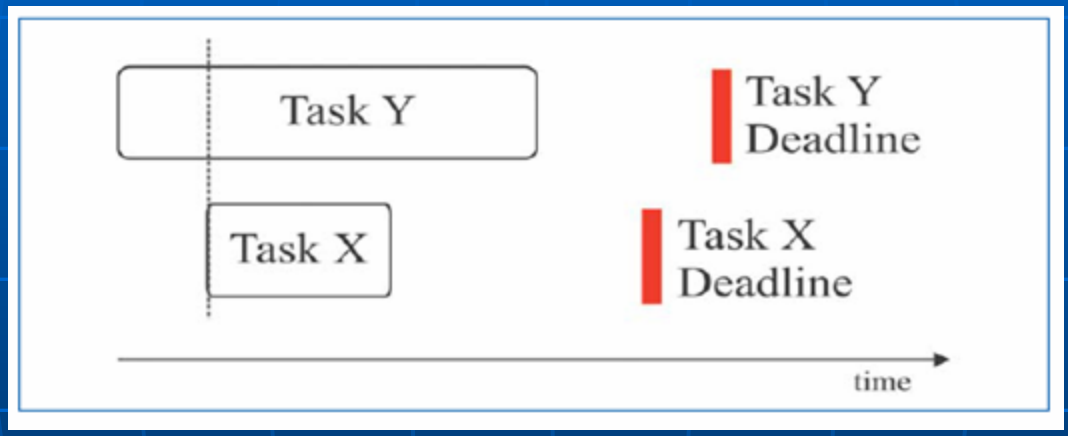
- There are a number of ways for an RTOS to do deadline scheduling:
  - EDF – earliest deadline first scheduling:
  - LL – least laxity scheduling
  - MUF – maximum urgency first scheduling.

EDF –Earliest Deadline First Scheduling:

- The task that is nearest to its deadline (and has not yet run) will be allowed to run first.
- Thus, an EDF scheduler views task deadlines as more important than task priorities

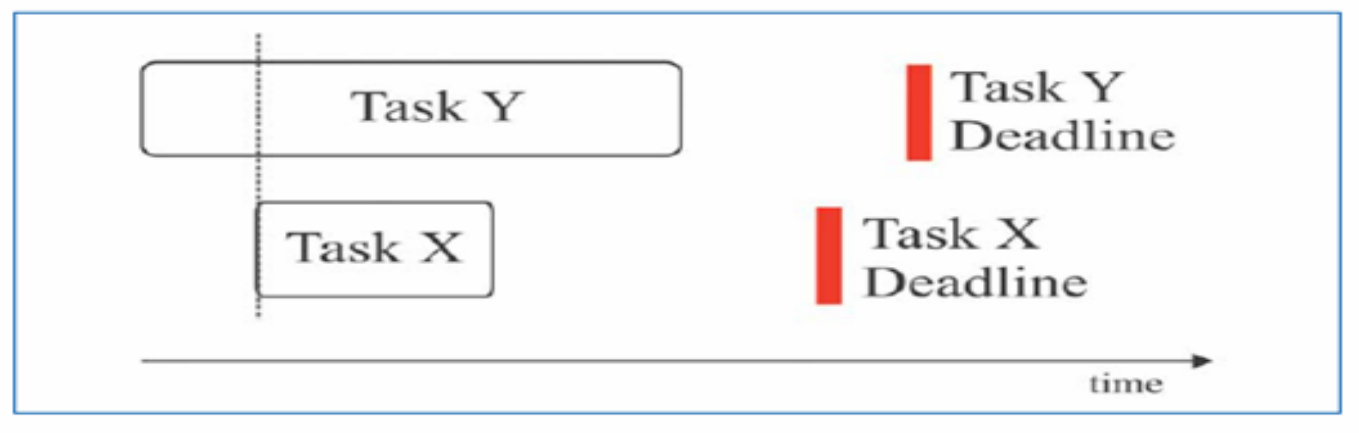
LL – Least Laxity Scheduling

- It takes into account both a task's deadline and its processing load.



• At above figure, an EDF deadline scheduler would allow Task X to run before Task Y, even if Task Y normally has higher priority

LL – Least Laxity Scheduling



- An LL scheduler evaluates the urgency of tasks using a value called "laxity", where laxity = (task deadline - task execution time).

- Laxity is the amount of time that the scheduler can "play with" before causing the task to fail to meet its deadline.

- When the LL scheduler has evaluated the laxity for all tasks, it finds the task with the smallest current value of laxity – and that is the task that needs to be scheduled to run next.

MUF – Maximum Urgency First Scheduling.

- It is a mixture of some LL deadline scheduling, with some traditional priority-based pre-emptive scheduling.
- high-priority time-critical tasks are scheduled with LL deadline scheduling.
- while within the same scheduler other (lower-priority) tasks are scheduled by priority-based pre-emption.

Summary

- Three configurations of RTS are studied:
  - **Foreground/Background Systems**
  - **Non Preemptive "Cooperative" Multitasking**
  - **Preemptive Multitasking**

- **Some of Scheduling algorithms are studied:**
  - Priority Based Pre-emptive
  - EDF – earliest deadline first scheduling
  - LL – least laxity scheduling
  - MUF – maximum urgency first scheduling.