ECE3411 – Fall 2015
Lecture 6b.

Real Time Operating System: Scheduling Policies

Marten van Dijk, Syed Kamran Haider
Department of Electrical & Computer Engineering
University of Connecticut
Email: {vandijk, syed.haider}@engr.uconn.edu

UCONN

With the help of:
www.wikipedia.org
www.freertos.org

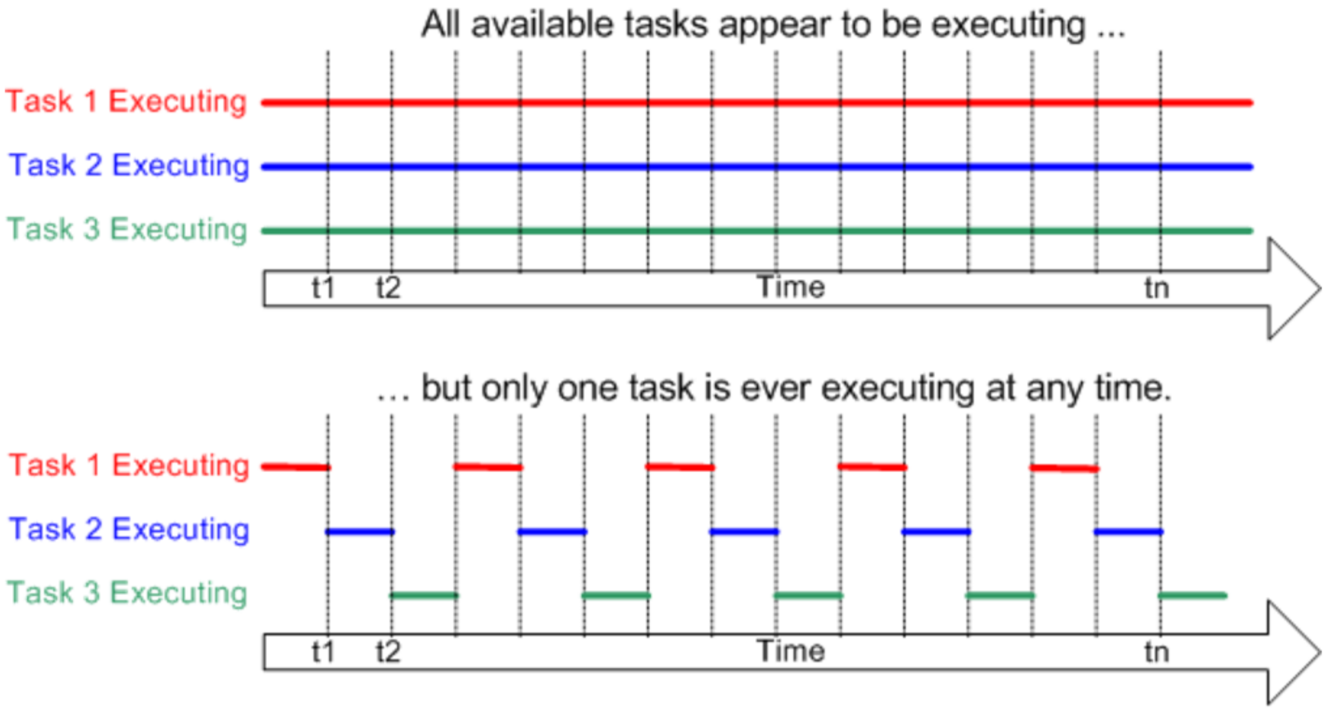
Operating System Fundamentals

- Multitasking
- Scheduling
- Context Switching
- Preemption

Multitasking

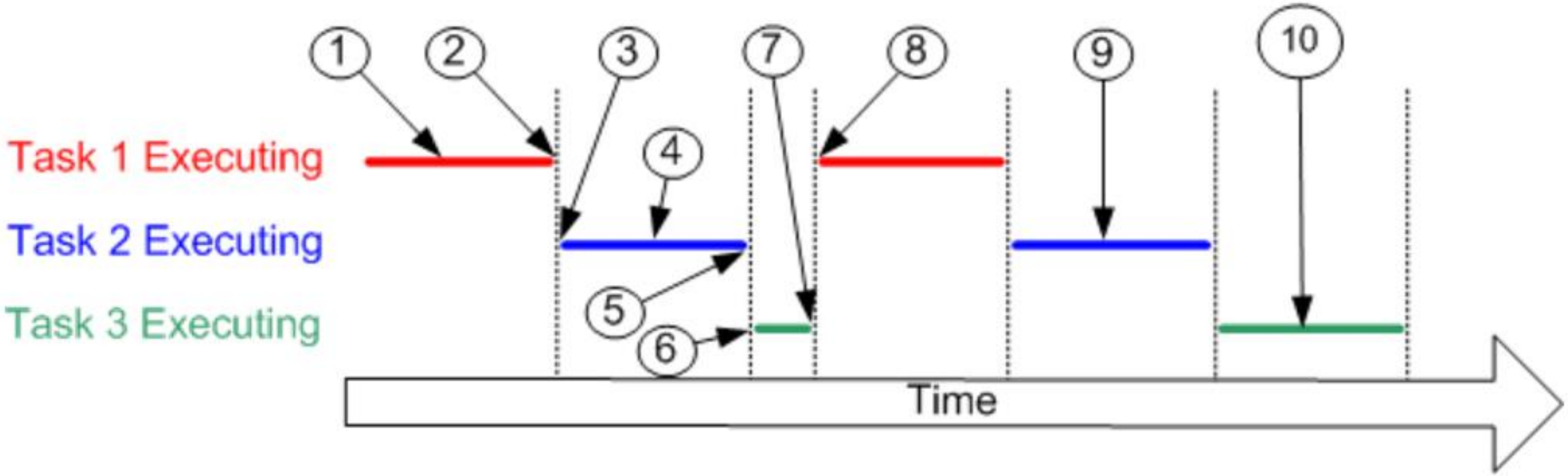
If an operating system can execute multiple tasks in *seemingly concurrent* manner, it is said to be **multitasking**.

- A conventional processor can only execute a single task at a time.
- However, rapidly switching between tasks can make it **appear** as if each task is executing concurrently.



Scheduling (1)

- The **scheduler** is the part of the OS kernel responsible for deciding which task should be executing at any particular time.
- The **scheduling policy** is the algorithm used by the scheduler to decide which task to execute at any point in time.



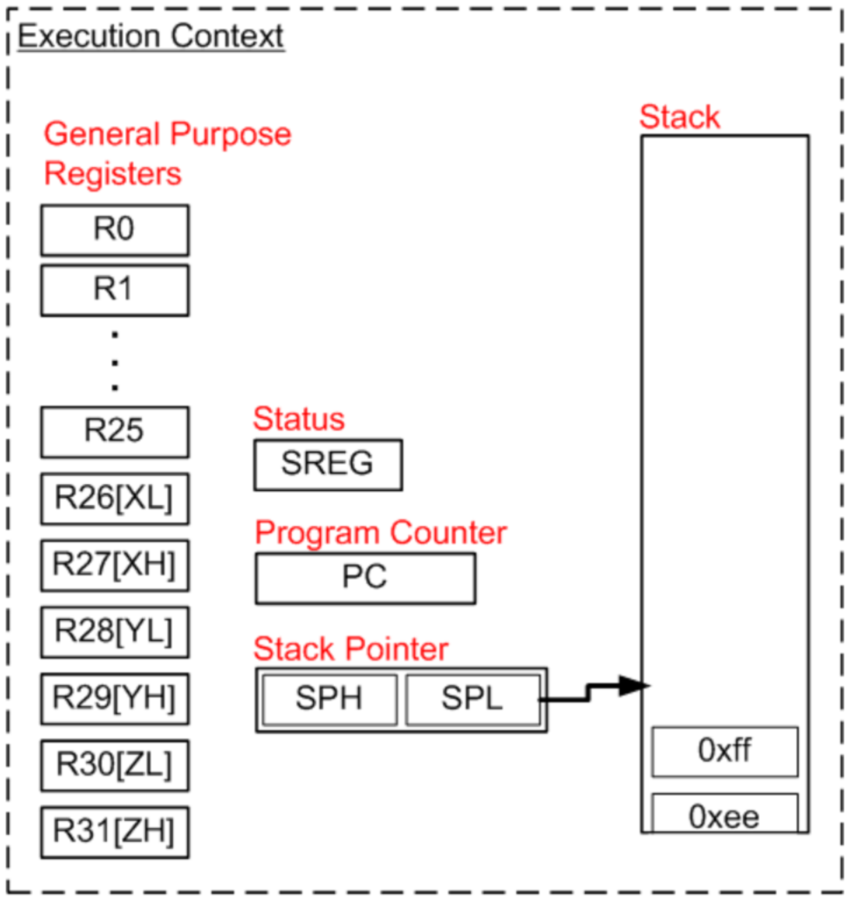
Scheduling (2)

Referring to the figure on the last slide:

1. Task 1 is executing.
2. The kernel suspends (swaps out) task 1 …
3. and resumes task 2.
4. While task 2 is executing, it locks a processor peripheral for its own exclusive access.
5. The kernel suspends task 2 …
6. … and resumes task 3.
7. Task 3 tries to access the same processor peripheral, finding it locked. Task 3 cannot continue so suspends itself at (7).
8. At (8) the kernel resumes task 1.
9. The next time task 2 is executing (9) it finishes with the processor peripheral and unlocks it.
10. The next time task 3 is executing (10) it finds it can now access the processor peripheral and this time executes until suspended by the kernel.

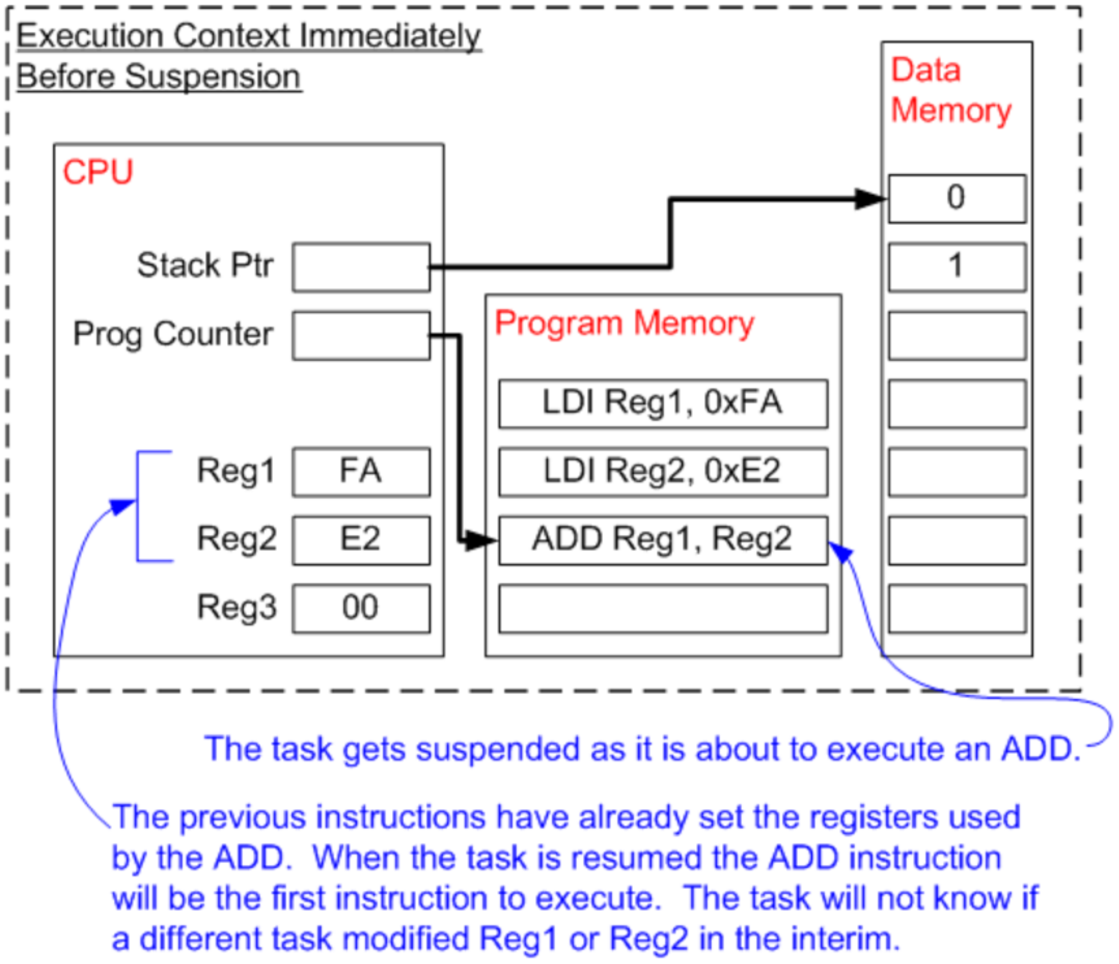
Execution Context

- As a task executes it utilizes the processor registers and accesses RAM.
- These resources together comprise the task execution **context**. In particular;
  - The Program Counter (PC)
  - The Status Register (SREG)
  - Processor's general purpose registers (R0 - R31)
  - The Stack Pointer



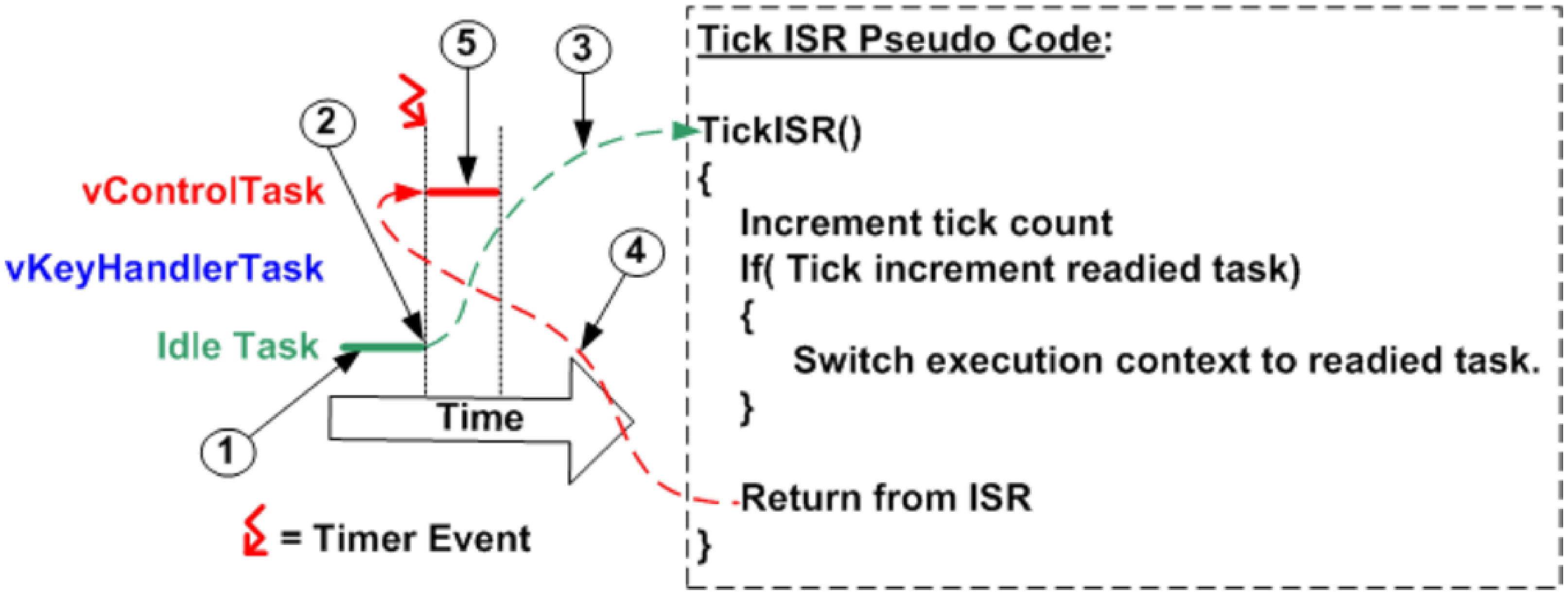
Context Switching

- It is essential that upon resumption from a suspended state, a task has a context identical to that immediately prior to its suspension.
- The operating system kernel saves the context of a task as it is suspended.
- The process of saving the context of a task being suspended and restoring the context of a task being resumed is called **context switching**.



Preemption

- A context switch when the interrupted task is suspended without the task suspending itself voluntarily is called **Preemptive** context switch.
- FreeRTOS implements context switching in Timer1 ISR.
- Upon a context switch, the ISR effectively interrupts one task but returns to another.



Scheduling Policies

Static Scheduling Schemes

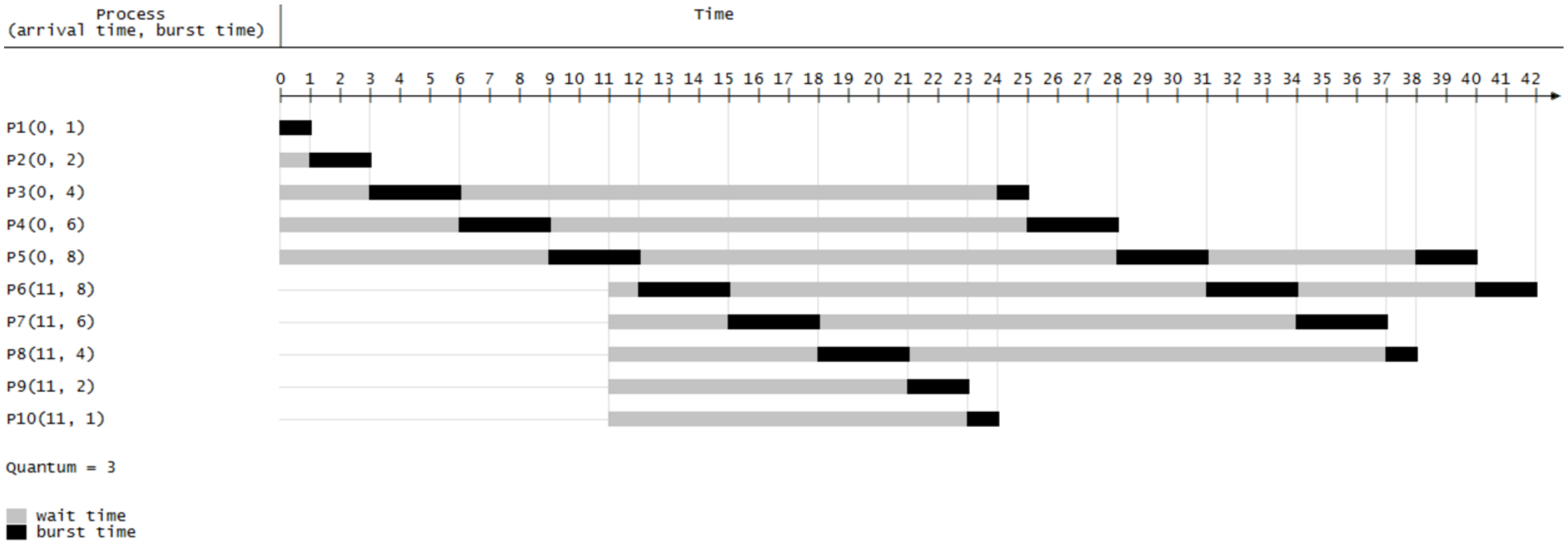
- Round-robin scheduling
- Rate-monotonic scheduling
- Deadline-monotonic scheduling
- Shortest Remaining Time First

Dynamic Scheduling Schemes

- Earliest deadline first scheduling
- Least slack time scheduling

Round-robin scheduling

- Round-robin (RR) is scheduling that assigns time slices to each process in equal portions and in circular order.
- All processes are handled without priority.



Rate-monotonic scheduling

- **Rate-monotonic scheduling** (**RMS**) assigns static priorities to the processes on the basis of the cycle duration of the job.
- The shorter the cycle duration is, the higher is the job's priority.
- RMS scheduling is generally preemptive.
- Preemption takes place when a higher priority task needs to be run while a lower priority task is running.

Rate-monotonic scheduling

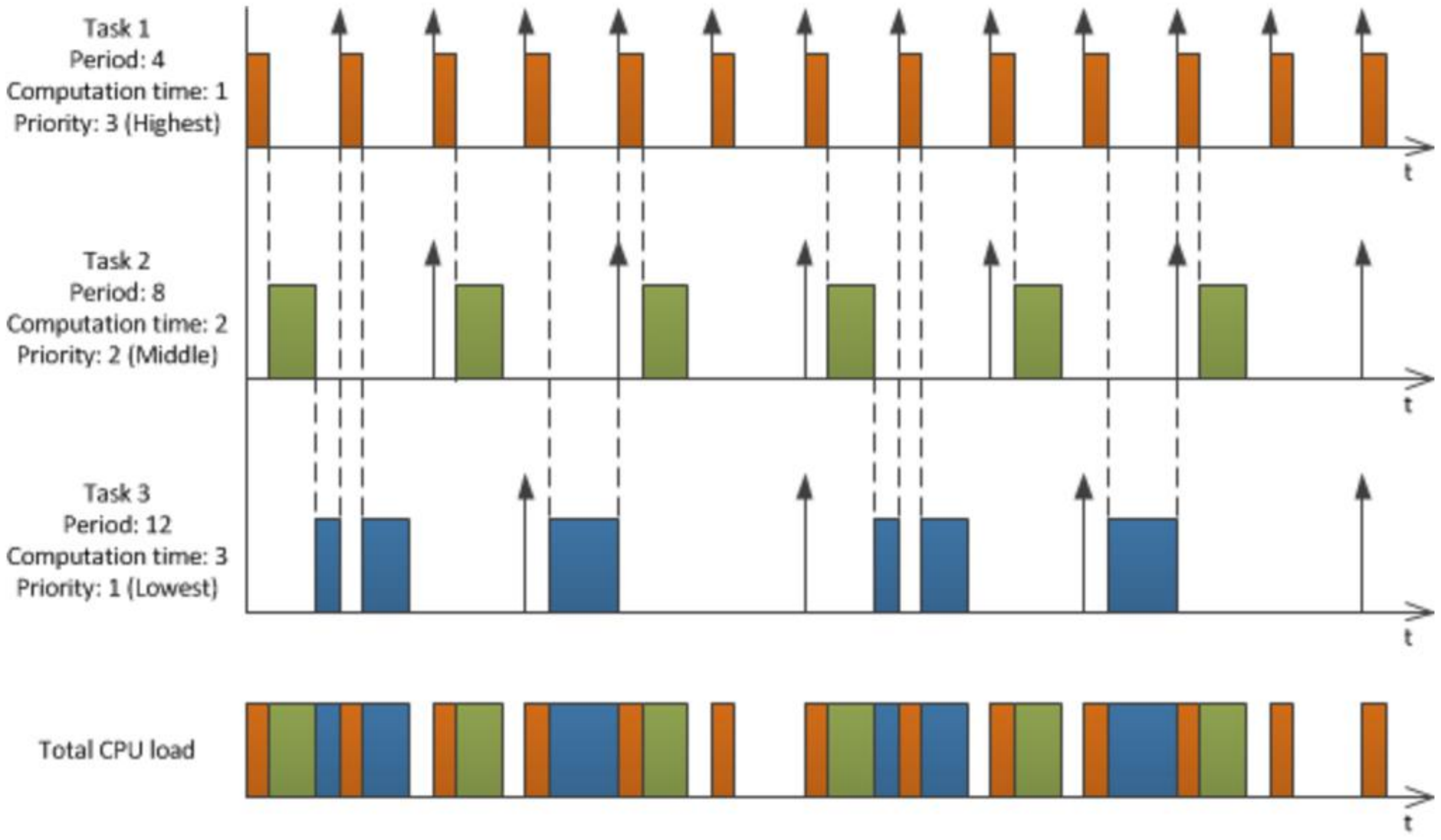


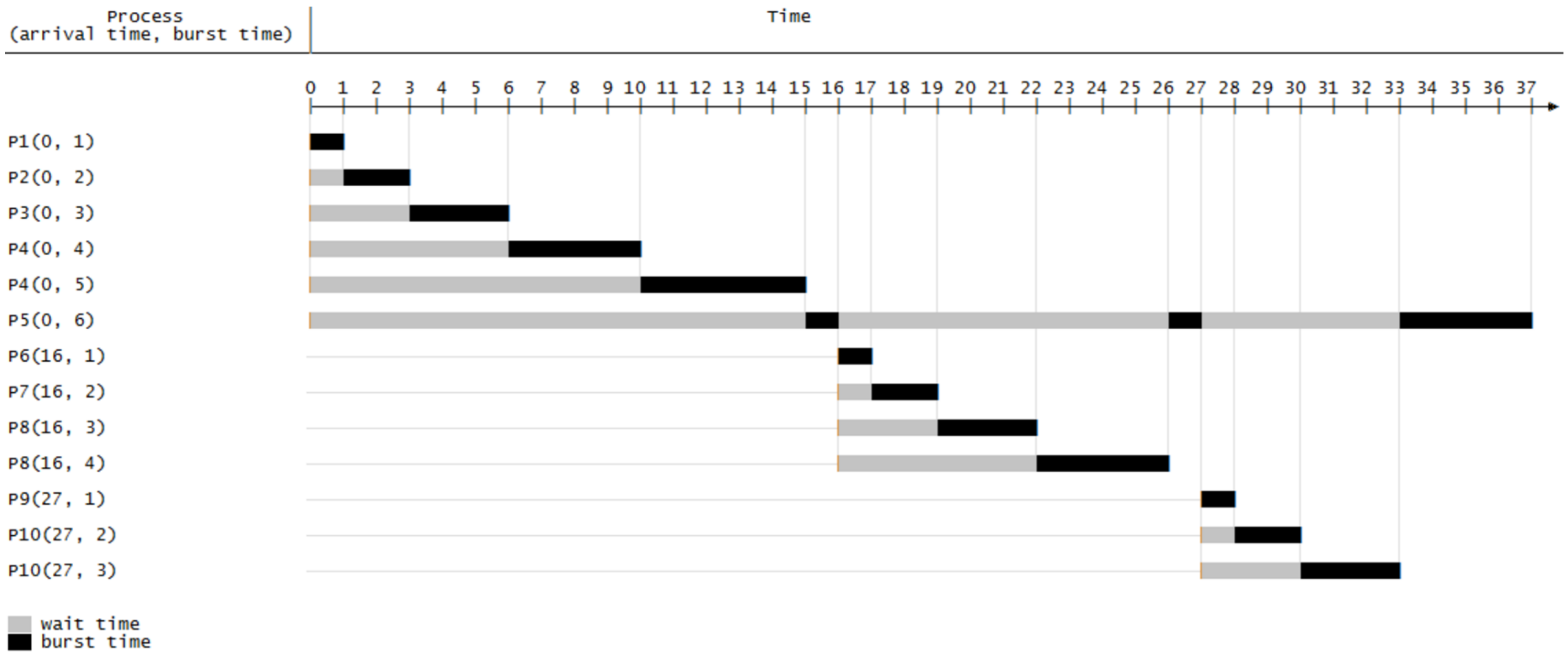
Image Reference: Generating multithread code from Simulink model for embedded target, by Petr Alexeev

Deadline-monotonic scheduling

- Deadline-monotonic priority assignment is a priority assignment policy used with fixed priority pre-emptive scheduling.
- Tasks are assigned priorities according to their deadlines.
- The task with the shortest deadline is assigned the highest priority.

Shortest Remaining Time First

- In **shortest remaining time first (SRTF)** scheduling method is preemptive.
- The process with the smallest amount of time remaining until completion is selected to execute.

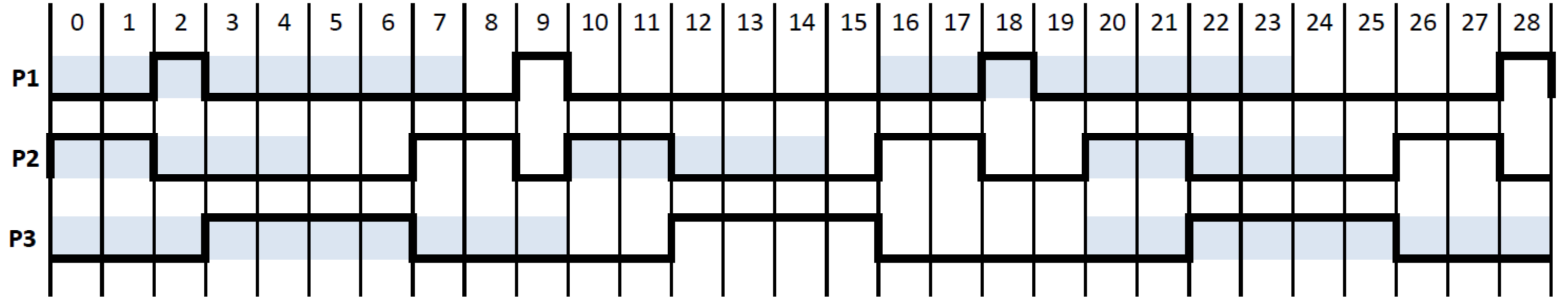


Earliest deadline first scheduling

- **Earliest deadline first (EDF)** is a dynamic scheduling algorithm to place processes in a priority queue.
- Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline.
- This process is the next to be scheduled for execution.
- EDF is an *optimal* scheduling algorithm on preemptive uniprocessors.
- In the example below, deadlines are shaded as background in the timing diagram.

Process Timing Data

Process	Execution Time	Period
P1	1	8
P2	2	5
P3	4	10



Least slack time scheduling

- **Least slack time (LST)** is a dynamic scheduling algorithm that assigns priority based on the slack time of a process.
- **Slack time** is the amount of time left after a job if the job was started now.
- More formally, the *slack time* for a process at any time t is defined as:

$$Slack = (d - t) - c'$$

where d is process deadline, t is the real time, and c' is the remaining computation time.