Assignments week 4 – Pre-emptive Scheduler

An alternative to sequentially running tasks is to pre-empt a task to run a new (more important) task. To be able to pre-empt a task, a context switch has to be implemented. Gladly, this has already been done by the instructor! Your goal with this assignment is actually implementing a specific scheduler, i.e. the algorithm that determines the next task to be run. A scheduler that chooses the next task based on its properties can be implemented in many ways.

4.1 Clone or pull *the last version of* the project repository from `https://bitbucket.org/HR_ELEKTRO/ros01_ccs_projecten` and import the project `VersdOS` into your workspace. Rename it to `Assignment4.1` and make sure you're able to run it. The leds should blink.

- Look at the currently implemented scheduler and be amazed (take hint of the sarcasm). An alternative to the currently implemented delay function has to be implemented. Create a function `delayTask(int ticks)` that will delay a task for a given amount of ticks. "Delay" in this case means preventing the scheduler from executing this task for the given amount of ticks.
- Now implement a priority-based scheduler. If necessary modify things to your need. Currently the task list is an array, perhaps several linked lists are better suitable to your cause? When tasks with equal priority are to be executed, a back-up algorithm should be used. The backup algorithm you are going to implement is the Shortest Job First (SJF) algorithm. You are allowed to specify the estimated run-time at task creation or to use a counter to determine its run-time.
- Verify the correct behavior of your priority-based scheduler by using a logic analyzer.