Otto-Friedrich-Universität Bamberg
Fakultät WIAI

Lehrstuhl für Praktische Informatik, insbes.
Systemnahe Programmierung – Prof. Dr. Michael Engel
https://www.uni-bamberg.de/sysnap
michael.engel@uni-bamberg.de

**Exercise 4**

# Preemptive multitasking

**Discussion on Monday, 4.7.2022**

*Since this will be a rather complex exercise, we will assign (almost) two weeks of time and discuss some solution approaches and remaining problems with your existing implementation next Monday (June 27th).*
There is no new skeleton code, so please finish your implementation of cooperative multitasking first, since the task switching implementation can also be used as the basis for preemptive multitasking.

## 4.1 Timer Interrupts

The timer interrupt is essential for our implementation of preemptive multitasking. So we need to write support for controlling the timer first.

- Initialize the CLINT timer in `setup` to interrupt the CPU after 100 ms. Enable the M-mode timer interrupts and the global interrupts. If your code works and you have added code to your `exception` function to print unknown exceptions, you should see exactly one of these printed (since you do not reload the timer yet).
- Add timer interrupt handling in `exception`. Whenever a timer interrupt occurs, set the `mtimecmp` register to a value that is 100 ms in the future, so the interrupt will trigger again. Add some debug statement (or check in gdb using a breakpoint) to confirm that the timer is in fact triggered periodically.

## 4.2 Preemptive Multitasking

With the timer interrupt enabled and working, we can now add code to the timer interrupt handling which enables *preemptive multitasking*.

- Switch to the next process (in round robin mode) when a timer interrupt occurs. Check your stack handling in `ex.S` to see if all important state is saved. If not, adjust it accordingly.

*Note:* You can leave your cooperative multitasking implementation in place, so a process could still give up the CPU by calling `yield()` before its assigned *time slice* is used up.

You might now notice problems with UART input or output, since our UART is still used in polled mode and can be accessed using system calls by several processes at the same time. We will add a *device driver* for the UART to properly handle the device in the next exercise.