**Fix a malfunctioning kernel**

**Problem symptom**

Build and run the given code on **the Rpi3 hardware**, it works as expected:

kernel boots ...  
Kernel process started. EL 1  
User process  
123451234512345abcdeabcdeabcdeabcd12345123451234eabcdeabcdeabcdeab512345123451234cdeabcdeabcdeabcde51234512345123abcdeabcdeabcdeabc451234512345123deabcdeabcdeabcdea45123451234512bcdeabcdeabcdeabcd345123451234512eabcdeabcdeabcdeab34512345123451cdeabcdeabcdeabcde234512345123451abcdeabcdeabcdeabc23451234512345deabcdea

Now build and run it **on QEMU**. What have you observed?

VNC server running on 127.0.0.1:5900  
kernel boots ...  
Kernel process started. EL 1  
User process  
12345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123451234512345123

**Problem cause**

Why is our kernel not preempting user tasks on QEMU?

The kernel's preemptive scheduling is driven by timer interrupts. Turns out that the given kernel source code only includes a driver (timer.c) for Rpi3's **"system timer"** (which we briefly mentioned in experiment 3 when introducing interrupts). However, QEMU does NOT emulate the system timer; it emulates **the ARM generic timer**, which we have been using since experiment 3.

The ARM generic timer is present on both the Rpi3 hardware and QEMU.

**Fix it!**

We will make the kernel (which implements user-level process support) work with the ARM generic timer. If successful, our kernel should be able to preempt user tasks on QEMU; it will behave as usual as running on the Rpi3 hardware.

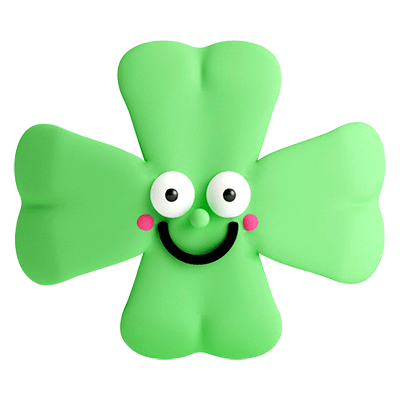
**Roadmap & challenges**

The first step is to port the driver for the generic timer from our previous kernel versions to this kernel version. This is easy -- mostly copy & paste, update some macros, etc.

The challenge is the memory mapping for registers of the generic timer. In previous experiments, this was not a problem because MMU was off and our kernel uses physical addresses. In this experiment, we turned on MMU so the kernel must establish memory mapping for IO registers in order to access them.

Check out the table "Other timers on Rpi3" in experiment 3: the registers for the generic timers are above address 0x40000040. Unfortunately, our current kernel only maps up to 1GB (0x40000000) physical memory. Additional mapping is needed for these registers.

Now you will have to figure out how to allocate and populate the needed additional pgtables and eventually get everything work.

Good luck! 

1. In your own words, describe the cause of the problem – why isn’t the given kernel doing preemptive scheduling? (10%)
2. In your own words, write a list of items to fix (10%)
3. Describe 1-2 most challenging & unexpected issues you have run into. (10%)
4. Code (70%)

[upload a standalone tarball named as p1-6.tar.gz]