

## Lab Three: Multitasking on MPC875

In this lab we continue exploring the PowerPC instruction set and the architectural features of MPC875. We will develop a process scheduler that allows multiple processes to share a processor. We will use the decremter exception to interrupt a process and switch to another in the exception handler. A template file is available on the course website. You can download it and start from there.

You do not have to implement all the features of a process, for example, creating individual address space for each process. However, your scheduler must work with the test codes provided on the course website and other codes we may provide.

### 1 Process scheduler

The main goal of this lab is to implement an exception handler that switch processes. It uses the decremter exception to interrupt the execution of a process and performs the context switching inside the exception handler. All the processes are in infinite loops and never exit.

#### 1.1 Writing two simple processes

You may want to start with two processes. You can simply use your `blink_led` function in lab 2 with different parameters, so they will blink two different LEDs. You can either put the processes in the same file as the exception handler or put them in a different file and include the file with C directive `#include`.

In the test codes that will be provided, the first process is a function called `process0`, the second process is `process1`, and so on.

#### 1.2 Enabling timebase

The timebase and decremter may be disabled by default. You need to enable them by setting proper bits in TBSCR. Note that this register is a keyed register, as well as the decremter register. For more information about TBSCR, please refer to section 10.9.3 of MPC855 PowerQUICC Family Reference Manual. You can also find detailed description for decremter register (DEC) in section 10.8.1 of the manual.

#### 1.3 Enabling the decremter interrupt

Before enabling the decremter interrupt, you need to make sure a working handler is already placed at the location `0x0000_0900`. A simple one is given in the template file, which just returns to the interrupted process. Note that the exception handler is copied to `0x0000_0900` from other locations in the main function.

Now, you may enable the decremter interrupt by setting DEC to a proper value and setting proper bits in machine status register (MSR) to allow external and decremter interrupt. For more information about MSR, please refer to section 4.1.2.3.1 of the reference manual.

## 1.4 Handling the decremter exception

Now you can work on your exception handler, or process scheduler. When switching processes, make sure you save all the user registers and proper supervisor registers for the old process. To save those registers, you need to define a data structure called Process Control Block (PCB) for each process. In addition to the content of registers, you may also need to keep the information about the starting address of process, pointer to next PCB, etc.

You can do all the initialization work in a function, say, `init_multi_tasking()`, in which you first initialize the PCB for each process, set proper registers for exception handler, enable the timebase, set the decremter register, and enable the decremter exception.

The decremter exception handler is located at `0x00000900`. You can only place a small piece of code there. However, you can call another large function to do more complicated tasks. To call another function in the exception handler, you need to use the absolute address (not the relative address) of the function. You load the entry point of the function in the link register first and make the function call with the `blr1` instruction. An example is given below, in which `SC_Handler` is called. Of course, you need to save the value of `r3` and `LR` first.

```
lis      r3,SC_Handler@ha
addi    r3,r3,SC_Handler@l
mtlr    r3
blr1
```

When the processes blink two LEDs simultaneously, observe the blinking rates of LEDs. Are they the same as when only one process is running? Change the time slice assigned to processes and observe how the changes affect the blinking rates.

## 1.5 Testing your code thoroughly

In addition to running two processes that blink LEDs, your scheduler should be able to handle more than two processes; the maximum number of processes is decided by a constant in your code. Test codes will be posted on the course website, and will be used to test your exception handler. TA may also test your scheduler with the codes that are not available on the course website.

## 2 Deliverables

You will write a report adhering to the lab report requirements. You will be asked to demonstrate your programs, and TA may test your scheduler with other codes.