实验 3 报告

学号：2016K8009908007
姓名：薛峰

一、实验任务

实验目的：

编写一段汇编程序，运行在 ls132 的 SoC_Lite 上，实现一个 24 小时进制的电子表。其功能如下：

(1)电子表具有一个暂停/启动键以及时、分、秒设置键；

(2)电子表复位后从 23 时 59 分 55 秒开始计时。按下暂停/启动键则进入设置模式，此时可以通过时、分、秒的设置键修改时间。再次按下暂停/启动键则退出设置模式并从设置好的时间开始继续计时。并且时、分、秒设置键在计时模式下无效；

(3) 每按一次时、分、秒设置键，对应的位置的值加 1，按住不放则按照一定频率不停地加 1 直至按键松开；

(5) 按下矩阵键盘上非设置键，不影响电表的计时。

检验方法：

(1) 仿真检查：每过一定的时间，数码管寄存器显示的时钟秒钟上加 1。在 testbench 里增加按钮开关的激励，以模拟电子表设置的功能；

(2) 上板检查：需要准确看到每过 1s，板上数码管秒钟加 1，并且可以通过按键进行电子表设置。

二、实验设计

本次实验采用时钟中断和硬件中断的方式完成。

硬件方面：

需要将按键信号从 confreg 模块中取出，并传到 cpu 模块中的 interface 模块中。

软件方面：

用寄存器 t0~t5 分别表示秒钟个位、秒钟十位、分钟个位、分钟十位、时钟个位、时钟十位；用寄存器 s0 表示电子表状态，0 为计时状态，1 为设置状态。

主要有以下几个函数：

（1）locate 函数：主要工作为初始化，初始化 t0~t5 寄存器、s0 寄存器、cp0_count 和 cp0_compare 寄存器、cp0_status 寄存器；

（2）display_time 函数：将当前时间写入 num_data 中。需要注意的是，因为 num_data 以 16 进制数展示，因此需要做进制转换；

（3）中断入口函数：根据 IM 和 IP 域判断中断的类型，从而跳转到相应的中断处理函数；

（4）int_timer 函数：时钟中断处理函数，将当前的时间加 1 秒，并显示出来。需要注意的是各个位置的进制。

（5）set_time 函数：硬件中断处理函数，用于处理模式转换按钮。对 IM[7]和 s0 取反，从而达到屏蔽/打开时钟中断、切换模式的效果。

（6）set_hour，set_minute，set_second 函数：硬件中断处理函数，分别用于处理设置时、分、秒按钮。需要注意的是如果当下状态不是状态，需要跳过，这样可以保证时、分、秒设置键在计时模式下无效。并且各个位置的进制也不尽相同。

三、实验过程

（一）实验流水账

时间	记录
11 月 2 日 14：00~16：30	一、阅读讲义，并思考总体的实现流程。 二、修改硬件源码，为实现硬件中断做准备。
11 月 2 日 17：50~22：10	一、写 start.s； 二、 写 testbench; 三、进行仿真验证，失败。
11 月 4 日 13：30~18：10	一、根据仿真波形，并对照 test.s 进行 debug，最终波形正确； 二、上板运行通过。

（二）错误记录

1、错误 1

（1）错误现象

上板验证时发现设置时钟和分钟的时候，没按下一次设置时钟或设置分钟键，秒钟总会加 1。

（2）分析定位过程

进行仿真验证发现，按过设置按钮后，虽然 IM[7]是 0，并且寄存器 s0（保存当前状态）指示的是设置状态，但是每当硬件中断到来时(假设此时 count 寄存器的值大于 compare 寄存器的值)，总会进入到时钟中断处理函数中去。

（3）错误原因

我之前以为硬件会保障只要 IM[7]为 0，IP[7]的值就不会为 1。因此我之前在中断处理函数的入口处，只是根据 cause 寄存器 IP 域的值来判断中断的类型。但是查看源码后发现，IP 的值只与 count 和 compare 寄存器有关，与 IM 无关。因此，如果此时如果 count 寄存器的值大于 compare 寄存器的值，IP[7]置 1，则会跳转到时钟中断处

理函数。因此在按下设置时、分、秒的按键的时候，会进入时钟中断，所以秒会加 1。

（4）修正效果

在中断入口处将 IP 与 IM 按位与，即可屏蔽时钟中断。

（5）归纳总结

这个错误是我对硬件处理中断的流程不太清楚造成的。

2、错误 2

（1）错误现象

在 testbench 中有对按键的赋值，但是并不产生硬件中断。

（2）分析定位过程

调出 confreg 模块里的 conf_int_n_i 信号，发现这个信号并没有变化。于是判断是对 conf_int_n_i 的赋值出现了问题。

（3）错误原因

之前 conf_int_n_i 信号是由 btn_key_r 信号决定，即

```
assign conf_int_n_i = {1'b1, ~btn_key_r[3:0], 1'b1};
```

但因为在 testbench 中无法 force btn_key_r，于是改为 force btn_key_data。因此 conf_int_n_i 信号没有变化。

（4）修正效果

将对 conf_int_n_i 赋值的语句改为：

```
assign conf_int_n_i = {1'b1, ~btn_key_data[3:0], 1'b1};
```

后，仿真结果正确。

四、实验总结

本次实验主要通过汇编完成，虽然与其他实验关联不大，但是通过本次实验，我进一步熟悉了 MIPS 汇编；并且这次实验让我更加清楚了时钟中断的处理流程，修改了我之前对于 IM 和 IP 的误解。