实验 4 报告

第 32 小组
施璠、袁峥

一、实验任务（10%）

编译一段汇编程序， 运行在 SoC_Lite 上， 调用 Confreg 模块的数码管和按钮开关等外设， 实现一个 12 /24 小时进制的电子表，并在实验板上予以演示。该电子表的显示包含时、分、秒，采用实验箱开发板上的 4 组数码管显示， 并通过板上的矩阵键盘完成电子表的设置功能。具体要求是：

(1) 电子表具有一个暂停/启动键， 具有时、分、秒设置键。

(2) 电子表复位结束后从 0 时 0 分 0 秒开始计时，按下暂停/启动键一次则计时暂停进入设置模式，此时可以通过时、分、秒的设置键修改时、分、秒的值，再次按下暂停/启动键则推出设置模式并从设置好的时间开始继续计时。

(3) 时、分、秒设置键的设置方式是每按下一次，对应的时、分、秒值循环加 1，按住不放则按照一定频率不停地循环加 1 直至按键松开。

(4) 时、分、秒设置键仅在设置模式下操作才有效果。

(5) 矩阵键盘上非设置键被按下，应当不影响电子表的精确计时。

(6) 采用硬件中断+时钟中断完成本次实验。

二、实验设计（30%）

此时实验全部采用中断处理来进行，一共设置了 5 个中断，分别为时钟中断和四个按键中断、对应暂停启动、时设置、分设置、秒设置四个按键。代码中使用了几个寄存器来固定保存一些数据，a0 保存数码管的内存地址，a1 保存当前态（0 为运行态，1 为暂停态），a2 保存从运行态进入暂停态时 CP0_COUNT 的值，a3 保存当前时间。

在初始化阶段设置当前时间为 0，设置当前模式在运行态，设置 CP0_STATUS 为打开时钟中断和暂停、启动键中断、关闭时分秒设置键中断，设置 CP0_COMPARE 为 25000000（对应一秒）、设置 CP0_COUNT 为 0。

结束初始化以后，进入 loop 死循环，等待中断进入。

在中断处理程序部分，首先判断中断类型。

（1）时钟中断

将 a3 寄存器即当前时间加 1，然后更新数码管显示，并重置 CP0_COMPARE 和 CP0_COUNT，同时完成清中断，然后退出。

（2）暂停启动键中断

国科大UCAS2019H计算机体系结构研讨课17-18秋季

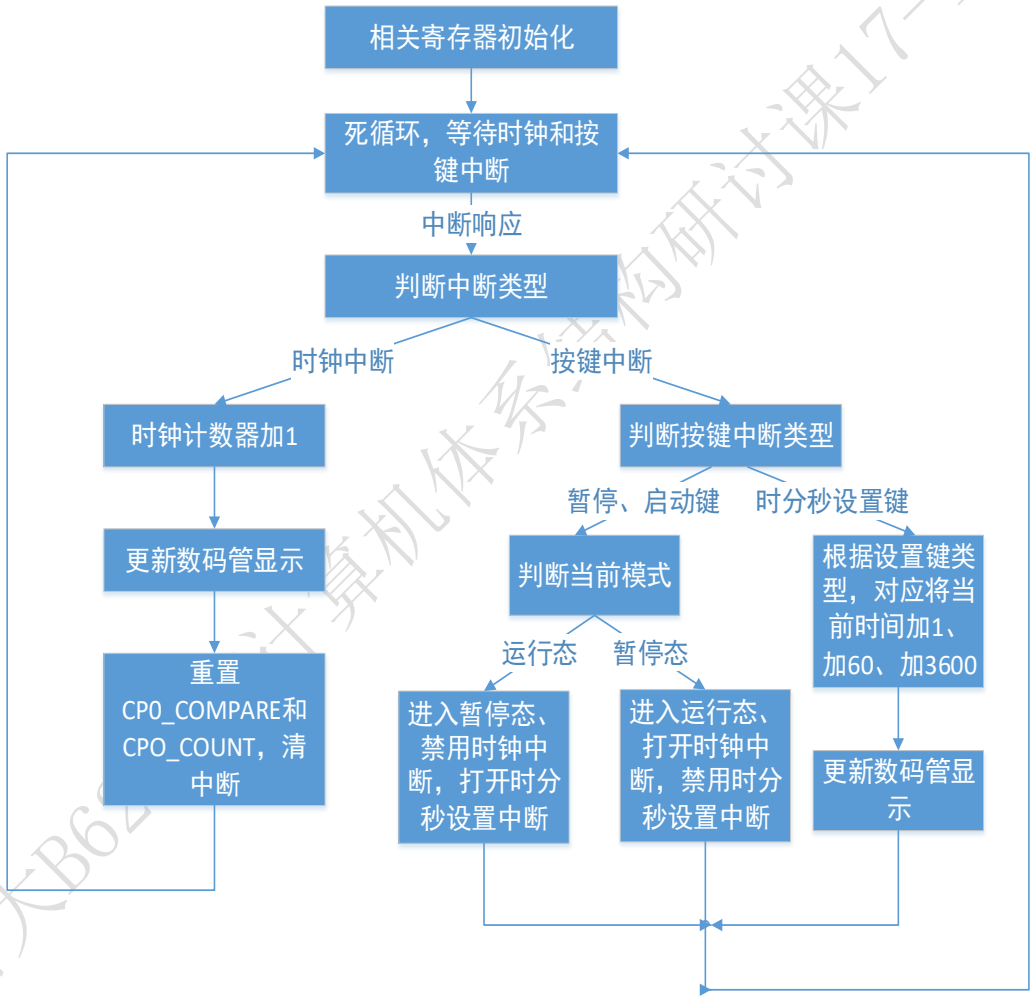
如果当前在运行态（即 a1 为 0），则切换至暂停态，同时禁用时钟中断，启用时分秒设置键中断，并保存此时的 CP0_COUNT 至 a2。

如果当前在暂停态（即 a1 为 1），则切换至运行态，同时启动时钟中断，禁用时分秒设置键中断，并恢复 a2 的值至 CP0_COUNT。

（3）时分秒设置键中断

检测具体是时、分、秒哪一个按键的中断，然后对应分别将 a3 寄存器加 3600、60、1，并更新数码管显示。

本次实验整体流程图如下：



此外，在上板检验时发现，此时的设计会导致按一次秒设置键，时间会增加几千甚至几万秒，这是由于每次按键的时间可能有几十毫秒，在这段时间中按键中断一直有效，因此每次处理完中断后又检测到按键中断，于是继续处理，在几十毫秒内可能会将这一操作进行几千次。因此后来在所有的按键中断处理中，增加了大约 0.1 秒的空跑，这样增加了每次按键中断处理的时间，能够准确区分出每次按键，同时如果一直按着按键，也可以实现时间连续增加的效果。

在顶层连线方面，从 confreg.v 模块将 btn_key_r 寄存器引出至 soc_lite_top.v，并根据 btn_key_r 的值

（是否为 1、2、4、8）将 int_n_i 的 0 到 3 位的相应位置 0，作为按键的检测。同时将 int_n_i[5]常置 1，用来处理时钟中断。

三、实验过程（60%）

（一）实验流水账

2017 年 11 月 2 日晚上，阅读任务书，查阅相关资料，编写代码。（大约 4 小时）

2017 年 11 月 3 日下午，调程序，最终实现相关功能。（大约 4 小时）

（二）错误记录

1、错误 1

（1）错误现象

仿真时检测到中断后，PC 跳转至 0x80000180 而不是 0xbfc00380。

（2）分析定位过程

根据波形看到 debug_wb_pc 跳转不正确。

（3）错误原因

查找相关资料后了解到，如果 CP0_STATUS 的 BEV 位为 0，则异常入口基地址（EBASE[CPU]）为 0x80000000，如果 CP0_STATUS 的 BEV 位为 1，则异常入口基地址（EBASE[CPU]）为 0xbfc00200，而在程序开始初始化寄存器时恰好将这一位置了 0。

（4）修正效果

在初始化时将 CP0_STATUS 的 BEV 位置 1。

（5）归纳总结

在对不了解的寄存器进行操作时，应该先要仔细查找资料，了解每一位对应的功能。

2、错误 2

（1）错误现象

仿真时除法运算出错，并意外更改了 a1 寄存器，导致后序也出现问题。

（2）分析定位过程

首先发现 a1 寄存器的值和自己预想的不一样，然后根据波形查找到了 a1 寄存器被意外更改的地方，发现是在除法指令中。

（3）错误原因

首先我相信了 CPU_gs132 处理器中真正运行除法的过程肯定是对的，然后我查看了交叉编译结果反汇编后的代码，发现编译器对 divu t3,t4 此类指令进行了重新调整，编译后的结果变成了 divu zero,t3,t3 mflo a1。这样就意外

国防科大B62009H计算机体系结构研讨课17-18秋季

的使用了 a1 寄存器。一开始我对这一现象很费解，然后我查看了在上一次实验中的 divu 测试程序，发现其反汇编后的结果没有出现这类情况，再发现测试程序中的除法是直接使用了 divu zero,t3,t4 的写法。后来查看资料后发现，两者的除法效果是等价的，但是编译器在编译时可能会有所区别。

（4）修正效果

将所有除法指令进行了修改，并重新查看编译后的反汇编结果发现问题解决。

3、错误 3

（1）错误现象

上板时发现在暂停态按秒设置键时，每次时间会增加几千秒。

（2）分析定位过程

分析实际按键情况，知道了问题的原因。

（3）错误原因

每次按键尽管很快，但也可能会要花费几十毫秒，这段时间内中断处理程序可以运行几千次，也就检测到了几千次中断，时间也就相应增加。

（4）修正效果

在所有的按键中断处理中，每次增加了 0.1 秒左右的空跑。重新上板检查后发现问题解决。

（5）归纳总结

实验设计时的考虑不全面，没有想到最后使用时出现的问题。

4、错误 4

（1）错误现象

上板时一开始运行在运行态时，按下秒设置键时间也会增加。

（2）分析定位过程

根据错误现象想到可能是初始化时的寄存器值没有设对。

（3）错误原因

查看了 start.S 代码，发现程序一开始设置在运行态，但在初始化是没有禁用时分秒键设置中断。

（4）修正效果

在寄存器初始化时将 CP0_STATUS 的时分秒键设置中断关闭。

（5）归纳总结

代码编写时疏忽了细节问题，寄存器初始化时没有考虑全面。

国科大B62009I计算机体系结构研讨课17-18秋季

四、实验总结

（一）组员：袁峥

啊啊啊，这周心情超级差，强忍着悲痛做完了这次的实验。发现比起操作系统的实验，这次的任务真是良心。

在编写代码前先仔细阅读了任务书和相关代码，特别是 confreg.v 模块的代码，对其中的接线上产生了一些疑惑，然后咨询老师得到了解决。在全部过程捋顺后，代码编写过程比较顺利。调试过程中主要的困难一个是每次修改完代码都需要重新交叉编译、然后替换存储器数据，比较麻烦，另一个是对 CPU_gs132 的代码不太熟悉，信号跟踪和查找上比较费劲。总体来说还算顺利，仿真测试大约进行了 5、6 次，上板测试了 2 次后所有功能得到了正确实现。

（二）组员：施璠

周六收到了队友发来的实验报告，还没开始的我又被带飞了。看看任务书觉得本次实验的要求还是挺浅显易懂的，等到要动手写代码的时候才发现有点无从下手。因为一些事情惹队友不开心了，影响了他的情绪，也没法向他请教问题，下次实验我会好好做的，也希望得到他的原谅。最后对于本次实验还有一些地方没有弄清楚，比如 CONFREG 模块里的数码管、矩阵键盘寄存器的访问方式以及 MIPS 架构的硬件中断机制。

国科大B6209H计算机体系结构研讨课17-18秋季