**实验7报告2**

第32小组

袁峥

一、实验任务（10%）

1. CPU 增加 TLBR、 TLBWI、 TLBP 指令。
2. CPU 增加 Index、 EntryHi、 EntryLo0、 EntryLo1 、 PageMask CP0 寄存器。
3. CPU 增加 32 项 TLB 结构， 支持的页大小位 4KB。
4. CPU 增加 TLB 相关例外： Refill、 Invalid、 Modified。
5. 运行专用功能测试 tlb\_func， 要求全部通过， 共 10 项测试。

二、实验设计（30%）

（一）对于支持TLB的整体设计

本次实验需要在CPU的设计中增加一个32项的TLB，同时支持相关的指令和CP0寄存器。TLB模块的设计图如下：



对于原来的CPU设计，首先需要增加一个TLB模块来进行取指和访存地址的虚实转换，并对TLB表项进行读写。

同时在mycpu\_top中需要增加5个CP0寄存器，分别为CP0\_Index、 CP0\_EntryHi、 CP0\_EntryLo0、 CP0\_EntryLo1 和CP0\_PageMask。其中CP0\_Index在TLBP指令时需要自动更新，CP0\_EntryHi、 CP0\_EntryLo0、 CP0\_EntryLo1 和CP0\_PageMask在TLBR指令时需要自动更新。所有的CP0寄存器的修改统一放在执行级。

在中断和例外处理上，延续原来的想法，所有异常在执行级统一处理。inst\_refill和inst\_invalid在取指级触发，data\_refill、data\_invalid和data\_modified在执行级触发。同时区分不同的异常处理入口地址，对于TLB\_refill异常处理入口地址为0xbfc00200，其余异常处理入口地址仍为0xbfc00380。

在流水级的阻塞和清空上，如果在取指级或执行级为TLBWI、TBLR或者为修改CP0\_ENTRYHI的MTC0指令，则此时取指级经过虚实转换后取来的指令需要重做，因为上述指令会修改TLB和虚实转换时需要使用到的寄存器，可能会导致后续的取指转换的地址不正确。

（二）TLB模块

（1）基本概述

该模块主要用来存储TLB表的内容，同时支持取指虚地址和访存虚地址同时转换为对应物理地址。此外，需要支持TLBWI、TLBR和TLBP三条指令。

对于TLBWI指令，在时钟上升沿根据Index\_in的索引将对应的TLB表项更新为输入值。

对于TLBP指令，首先在输入时对于真正要查找TLB的data\_vaddr通过TLBP信号进行二选一，如果是TLBP指令，则查找EntryHi\_in对应的虚地址，否则查到data\_vaddr\_in对应的虚地址。对于查找结果，如果找到了对应项，需要将查找信号进行编码，如果没有找到，则把Index\_out的最高位置1。

对于TLBR指令，根据Index\_in将TLB表中的对应项输出。

对于虚实地址转换，首先将虚地址的高19位与TLB表中每项的EntryHi域的高19位结合PageMask域进行比较，同时如果EntryHi域的第12位为0，还需将CP0\_EntryHi寄存器的ASID域与TLB中每项的ASID域进行比较，如果全部满足则说明命中。如果32项没有一项命中，则found信号置0。对于命中的情况，根据虚地址的第12位分别选取EntryLo0域或EntryLo1域的物理页框号，结合虚地址的低12位组成物理地址，同时将V\_flag和D\_flag置为对应EntryLo域中的V位和D位。

（2）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| TLBWI | IN | 1 | TLBWI指令信号 |
| TLBR | IN | 1 | TLBR指令信号 |
| TLBP | IN | 1 | TLBP指令信号 |
| inst\_vaddr | IN | 32 | 取指虚地址 |
| data\_vaddr\_in | IN | 32 | 访存虚地址 |
| EntryHi\_in | IN | 32 | CP0\_EntryHi寄存器的输入信号 |
| PageMask\_in | IN | 32 | CP0\_PageMask寄存器的输入信号 |
| EntryLo0\_in | IN | 32 | CP0\_EntryLo0寄存器的输入信号 |
| EntryLo1\_in | IN | 32 | CP0\_EntryLo1寄存器的输入信号 |
| Index\_in | IN | 32 | CP0\_Index寄存器的输入信号 |
| EntryHi\_out | OUT | 32 | 需要修改的CP0\_EntryHi寄存器的值 |
| PageMask\_out | OUT | 32 | 需要修改的CP0\_PageMask寄存器的值 |
| EntryLo0\_out | OUT | 32 | 需要修改的CP0\_EntryLo0寄存器的值 |
| EntryLo1\_out | OUT | 32 | 需要修改的CP0\_EntryLo1寄存器的值 |
| Index\_out | OUT | 32 | 需要修改的CP0\_Index寄存器的值 |
| inst\_V\_flag | OUT | 1 | 取指地址虚页号在TLB查找到对应物理页是否有效 |
| data\_V\_flag | OUT | 1 | 访存地址虚页号在TLB查找到对应物理页是否有效 |
| data\_D\_flag | OUT | 1 | 访存地址虚页号在TLB查找到对应物理页是否可写 |
| inst\_paddr | OUT | 32 | 取指虚地址经过TLB映射后的物理地址 |
| data\_paddr | OUT | 32 | 访存虚地址经过TLB映射后的物理地址 |
| inst\_found | OUT | 1 | 取指地址虚页号是否在TLB查找到对应物理页 |
| data\_found | OUT | 1 | 访存地址虚页号是否在TLB查找到对应物理页 |

三、实验过程（60%）

（一）实验流水账

12月14日晚上8点至12点，阅读文档Lab07-2\_3\_CPU+TLB+MMU支持及MIPS手册，并编写代码。

12月15日下午3点至10点，调试程序并上板通过测试。

12月15日下午11点至1点，完成实验报告。

（二）错误记录

1、错误1

（1）错误现象

从CP0\_INDEX寄存器读出的数据不对。

（2）分析定位过程

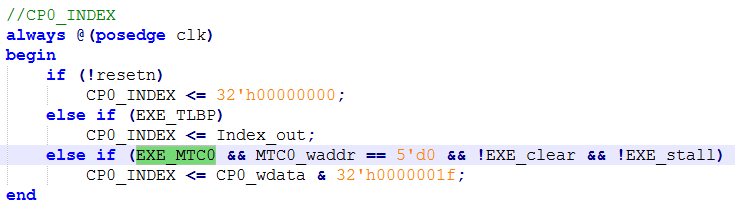
由于此次实验没有golden trace进行跟踪，因此在仿真时将每一条指令的写回级情况进行输出，这样便于观察。逐条指令跟踪，首先发现第一个测试点里的MFC0指令结果不对。

（3）错误原因

原来的设计中对于MTC0指令的写CP0寄存器的编号放在MTC0\_waddr信号中，默认0表示不是MTC0语句。在执行级看到MTC0\_waddr为对应CP0寄存器编号时就进行写CP0寄存器。但是增加CP0\_INDEX后，其编号恰好为0，因此CP0\_INDEX会被不断更新。

（4）修正效果

将CP0\_INDEX的写控制进行如下修改，增加判断是否为MTC0指令信号。



修改后通过第一个测试点。

（5）归纳总结

这个错误主要是由于在前面的实验时为了偷懒，由于没有用到CP0的0号寄存器就将其用作默认位，但此次实验中恰好增加了CP0的0号寄存器，因此出现了错误。这也提醒我以后写代码的时候不能偷懒而用小技巧，所有判断条件要写的更加完整严谨。

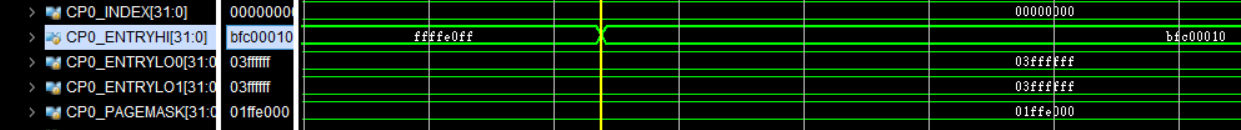
2、错误2

（1）错误现象

在TLBWI后，TLBR同一个TLB表项的内容结果不对。

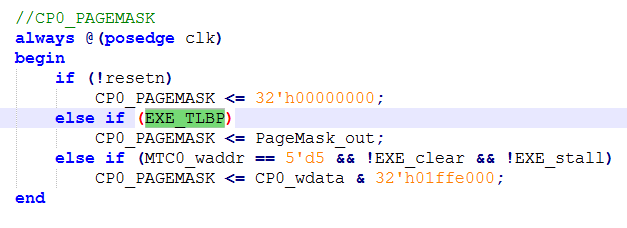
（2）分析定位过程

根据输出的运行信息，发现在第6个测试点中，提前跳到了inst\_error，然后找到出错的指令为TLBR，查看波形发现在TLBR时应该更改的四个CP0寄存器中，只有CP0\_ENTRYHI进行了更新，其余三个均没有更新。波形如下：



（3）错误原因

查看控制CP0\_ENTRYLO0、CP0\_ENTRYLO1和CP0\_PAGEMASK寄存器读写的代码发现，在写控制条件中将TLBR写成了TLBP，所有导致这三个寄存器没有正常更新。



（4）修正效果

将字母P改为R，重新运行通过该测试点。

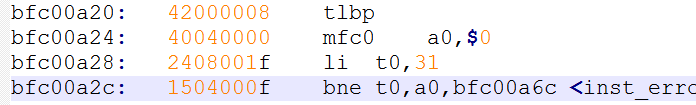
（5）归纳总结

写代码太不仔细了，本身指令名称就相近，应该更加细心一点。

下面是三处在调试过程中发现测试程序有问题的地方。

3、错误3

（1）错误现象



测试点7中指令bfc00a2c执行后t0和a0不相等导致跳转至inst\_error。

（2）分析定位过程

a0是TLBP执行的结果。我的程序得出的a0是29，但是标准的应该是31。而TLB上次更新是在测试点6中，于是查看了测试点6的代码，发现在更改TLB中的表项时，前29项是用循环更新的，下标会自动更新，但是最后3项是单独更新的，同时更新的时候没有更改下标，导致第29项TLB被连续更改三次。结果如下：



而测试程序本意应该是后三次更改第29、30、31项。所以导致最后应该出现在第31项中的值出现在了第29项，从而导致了测试点7中的错误。

（3）错误原因

测试点6中更新最后3项TLB时，下标没有每次加一。

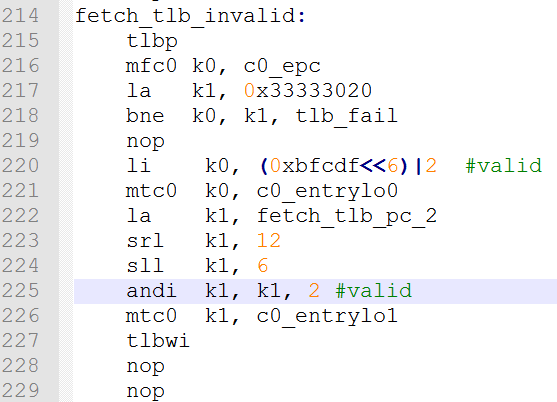
4、错误4

（1）错误现象

测试点10中，经过fetch\_tlb\_invalid处理后，取指指令仍然触发TLB\_invalid，从而后续程序执行出错。

（2）分析定位过程

查看了fetch\_tlb\_invalid的异常处理函数，如下：



导致取指异常的虚地址为0x33333020，第13位为1，因此对应的TLB中应该为EntryLo1项，但是按照异常处理函数中处理后，k1为0，即写入TLB的对应EntryLo1的项为0，从而下次取指仍然触发TLB\_invalid。

（3）错误原因

对比写入CP0\_ENTRYLO0寄存器的内容，发现第225行应该是或操作而不是与。

5、错误5

（1）错误现象

测试点10中，经过fetch\_tlb\_invalid处理后，下次取指时转换后的物理地址为bfc00020，该地址的指令与该测试点毫无联系，而没能继续后续测试点中的指令。

（2）分析定位过程

导致TLB\_invalid的虚地址为0x33333020，在经过异常处理后，在TLB中增加了对应项，物理页框号为0xbfc00，经过映射后对应的物理地址为0xbfc00020,从而取到了0xbfc00020处的指令执行。

（3）错误原因

如果要在异常处理后跳转到测试点10中后面的指令继续执行，应该修改虚地址的偏移，使其等于跳转到的指令的偏移。

四、实验总结

（一）组员：袁峥

这次实验在编写代码前花了两个多小时先进行了仔细思考，争取克服以前的毛病，在设计的时候不仔细从而导致后续调试时问题百出。最终发现好像多考虑了一些情况，比如设计中在完成一条TLBWI、TLBR或者MTC0修改了CP0\_ENTRYHI后，会让后续指令重新取指。因为上述指令会修改TLB和CP0\_ENTRYHI等内容，如果在完成前让后续指令取指，可能导致查找的TLB内容或者CP0\_ENTRYHI的ASID域不正确，从而出现取指错误。但这些情况在最后的测试程序没有出现。

这次实验代码的修改量还是有些大的，前四级流水、阻塞控制模块和mycpu\_top模块都进行了配套修改，但可能由于在编写代码前考虑的比较仔细，所以在调试过程中发现的错误不多，而且全都是笔误类的小错误，一看到就能够知道错误的原因。倒是发现测试程序的错误比较费时，需要同时判断是测试程序的问题还是自己的CPU的问题，而且由于没有标准的golden trace进行比较，许多内容需要逐条手算，找到错误的地方比较慢。

整体来说还算顺利，花了差不多一整天把这次的实验全部完成了。