首先，做P7需要**确保的是自己的P6完全正确**，因为这一次对于P6的测试应该是强测了（之前的课上貌似都是中测，至于课下，除了前三个P之外，放出来的所有测试应该都是弱测），之所以这样说还有一个原因，那就是P8没有乘除法，想在这方面卡人的话P7是最后一次机会。

然后再说一些可能不一定得到认同的，但是笔者认为有一定用处的话。如果是下一次上机P6的话，笔者不建议再去研究P7，而应该去研究期末考试。如果下一次P7而且想要尝试P8的话，最好从这周末开始学习P8的内容，双线作战。毕竟计组课设只有3学分，拿到了通关也不能把分数提高一大截。

从CPU到微系统的转变

01

写在前面的话

首先是从高层次上看一下P7到底是个什么东西。笔者认为，**从测试的角度来说，P7包括4个部分：P6，异常，中断，IO（就是读写TIMER）**。其中，这一次P7的课上完全没有检测中断（有随机中断的检测不叫检测了中断，只能说测了一下异常处理），异常和IO也是弱测，P6会有另一个专门的提交点进行强测。**从搭建的角度来说，P7有3个部分：P6，异常与中断的检测与处理，外设与桥。**其中，外设与桥的部分相对容易，可以先搭建好但不与CPU连接。之后**推荐的顺序是：异常检测->中断处理->中断检测->异常处理。**之所以这样思考，后面会具体说明。当然，正如HDLdalao所说，1000个人有4000个CP0设计方案，而一种CP0方案对应着也可能不止一种CPU结构，所以本次推送介绍具体的代码没有意义，将**主要描述思路**。而笔者本人经历过课上debug后几乎不能信任自己的架构，事实上笔者的架构和绝大部分人都不一样，连CP0的部署方式这种大框架都不一样。考虑到人多力量大的原理，**建议大家选择将CP0视作DM一部分的M级架构。**

02

初步搭建

首先是**桥和TIMER**，桥就是一些assign，TIMER先大致了解一下设计规范里面的思路，然后按照状态转移图做代码，这一部分助教说只有合理的操作，课上的操作真的是特别合理。所以不用细究，差不多就行。至此桥和TIMER的介绍结束。个人偏好是detector型译码风格，就是随用随译，每次只针对一个信号进行译码（而不是来源），因此bridge内部进行译码与控制。TIMER和CPU两部分的地址规则，功能规则等等保持独立且不同。

然后是**异常检测与中断处理**。首先，大家要有这样一个思路，指令执行起来是一个连续不断的，确定的序列。异常处理程序也是一个类似的序列。由于所有的异常出现与否，何时出现由原指令序列唯一确定，因此没有中断的微系统任意时刻的状态是唯一确定的。中断，断如其名，就是一个中断出去，再从原处返回的过程。因此，大家应该可以看出两个问题。**1：中断受害指令不能生效，因此中断检测应该尽可能靠前，使尽可能少的指令类型生效。换句话说，没有异常的话，把中断检测放在F级应该也可以。2：中断处理应该在异常检测之后。因为中断检测是当前指令与后序的不能执行，而之前的全部执行，因此要确保这一部分放掉的指令没有异常。**因此，我们便得出了一个无特判微系统关系式：**异常早于中断早于生效**。在P7中，最先的生效是写入HI和LO，最晚的异常是M级存取异常。为了解决这个矛盾，有大概三个做法，大家可以自行选取。

第一种，是**最广为流传**的，**M级+回滚**。具体来说是CP0部署在M，异常中断都在M，特判，如果恰好遇上了生效比异常早的那一段序列，就回滚。据笔者所知，几乎所有dalao都采用的是这种结构。但课上听助教的意思，不回滚也行，毕竟符合see。。run。。的思想，纯M级也是可以的。

第二种，是**异常中断分开处理**，使用两套信号，将不等式拆开，对重合的特定序列进行特判。这种方式最好像，但容易出现bug，尤其是死锁，时序等等隐蔽bug。

第三种，是我采用的做法，**将M级的异常放在E级进行处理**。也就是说，有一个只运算加减法的运算器被部署在了E级，结果和ALU比较得出Ov异常，然后讨论E级指令是否为访存类型，并检测所有异常情况。笔者的这个方案在使用ipcore初步综合后比M级的架构大概慢了4%左右，可以忽略不计，并不会特别加大时序开销，然后CP0部署在E级。真实的微系统需要处理cache缺失，TLBmiss，缺页等异常，不可能这样做，不过用来通过P7课上中测是可以的。强测会不会出现系统bug（所谓系统bug，是说没有边界条件，连线全部正确，语义与预期完全一样的情况下仍然存在的bug。存在系统bug的架构一般可以通过特判解决，仍然无法解决的很可能根本无法使用）尚未可知。不是特别推荐大家使用。不过，E级的代码量少的可观。思路清楚的话，从P6开始做出一个P7大概只需要两个小时。隔壁M级可能会多一些。

然后是**CP0内部**，大体结构肯定是几乎一样的。这里面要完成中断的检测，直接抄**PPT**的表达式，不会错。

最后是**中断和异常的处理**。这里面E级架构可以再省掉几倍的代码量。M级的话，要做几件事情：PC转移，清楚流水线寄存器，关掉几个WE信号和start信号。

03

坑点和实现

**坑点**的话**讨论区**有集思广益的记录帖，建议每一条看一下，理解一下。主要出现在中断异常来临的时候传哪个EPC的问题，一定要搞清楚。然后是注意优先级，各种优先级。这几个问题一定要想清楚再动笔。即使是比较复杂的M架构，我可以确定从光秃秃的P6开始写也用不了三四个小时，周三下午动笔可能都来的及。之前我说设计文档没什么用，但在P7，一定要先写文档，先做测试，先构思。最后做代码的时候就像抄答案一样简单。

04

测试阶段

P7的课下两个都是弱测。建议的测试方式是**随机+定点爆破**。定点爆破所有可能的异常类型，一个类型一种（小类，比如SW没对齐算一类，用LH读TIMER算一类，大概加一起三四十类）。随机+定点爆破大概可以达到课上中测的程度。至于强测，最强的当然是遍历32位二进制数构造所有指令，然后再来个1000次方构造长度为1000的所有可能序列，但这肯定不现实，具体如何做到强测就有待各位斩下P8成为助教后再向笔者介绍了。功能的话，可以屏蔽掉PC异常去测P6，简单的异常可以去测高工P6，中断的话可以初始化设置好大概10周期一次中断（模式1），然后跑几百份几千行的代码，看会不会在某个意想不到的地方被中断搞崩。至于P7强测怎么测中断，笔者构造了一些思路，但都是定点爆破的中测，没有很大的实际意义。IO部分最简单，但**一定要测，一定要测！一定要测！一定要测！**（重要的事情说三遍）

~~除非你也想刺激一把！（划掉）~~

05

课上debug

接下来是比较刺激的P7课上debug。笔者在周四下午两点半左右是非常崩溃的，原因你懂的。课上debug最重要的是**思路和心态**。首先，如果**P6强测**被挂了，应该只能是活过了P6课上定点爆破然后被覆盖性挂掉了。**对策**：检查阻塞模块，有可能漏写。检测功能控制的译码器，有可能漏写。全扫一遍大概半个小时，debug改代码的过程大概10分钟，有机会抢救。如果**P6P7一起挂**，P6改对后有一丝可能连P7一起救了。因为之前P6课上被中测测试过，所以应该不会WA一大片，WA一大片的肯定是交错文件之类的低级错误。然后是**P7**，如果**全都WA**了，反而可能有救命的机会，因为可能是高层信号的失误。找一个最短的样例用MARS对比跑一下就行。如果**WA了5-6组**，其实是有些不幸的，debug成功的概率可能会渺茫，但是也要按照之前说的，用他提供的代码逐行比对一下。可以早到大概20分钟到半小时，现场写一个简单的，屏蔽handler内部PC的对拍器。**WA了两组左右**，也就是最常见的情况的话，就按照我之前的说法，考虑四个部分：P6，异常，中断，IO。P6不用管了，把irq赋值为恒0试一试，过了的话就下来查中断来临时存的EPC和BD。还没过的话先查IO，看看能不能写进去，写完了能不能读出来。他给的测试样例.data内部的内容在测试点里面都是以先写TIMER再读TIMER的形式做出的，形式不一样，但是不影响评测。

可能说到这里大家也猜到了，笔者正是在IO上面除了大问题。说的严重一些，就是所有通过LW读TIMER的全报废了，说的简单一些，就是看错了一根线。P7课上debug，找bug比修改困难很多。笔者改bug仅用了一分钟都不到，但是寻找它耗费了将近半个小时。

最后是上机的心态问题，一定**不要慌张**，尤其P7正考的时候一大片1A走人的dalao，肯定会给debugger带来一些负面影响，总之就是不要慌乱，当做高考就行了。笔试题很简单，个人推测是防止完全或半完全抄袭的，本质上没有什么用处。笔者笔试遇到的有意义的题目是无间中断相关的内容。至于TIMER设计状态图，照自己代码抄就是了。至于传EPC的问题，如果采用了我的架构，所有的答案都不一样。比如有一道题他问BEQ后面的nop在MEM时来中断传EPC传哪个，他是要考BD，传BEQ的，但笔者直接写了一个BEQ跳转目标，并说明了E级架构。其次，一个小技巧，尽可能晚的举手，拖时间不怕，被问答挂了才可怕。先看看他们都会问什么问题，准备一下再举手。如果课上debug的话当然就不用考虑这个问题了，肯定是要拖到很晚的。

**建议大家课下多测测，课上debug虽然刺激好玩，但真玩脱了也挺不幸的不是。（重点/捂脸）**