**体系结构 第三章**

**陈彦帆 2018K8009918002**

1.

（1）异常处理准备：MIPS把例外的pc存到EPC寄存器，X86则用栈存放CS:EIP。

（2）确定异常来源：MIPS将异常来源存于cause寄存器，由异常处理程序进行进一步查询和处理（并对TLB\_refill这样的频繁的异常设置了专用的异常处理入口地址）。而X86由硬件进行异常和中断号的查询，根据预设好的中断描述符表得到不同异常处理的入口地址，并将CS/EIP等压栈。

（3）保存执行状态，处理异常，恢复执行状态并返回：保存的寄存器名称不同；X86的返回指令为iret，而MIPS的返回指令为eret。

2. 一般来说，发生异常后，会在eret时清除ll bit。因为可能在上下文切换时发生未知的嵌套ll指令，而mips不支持ll/sc嵌套。

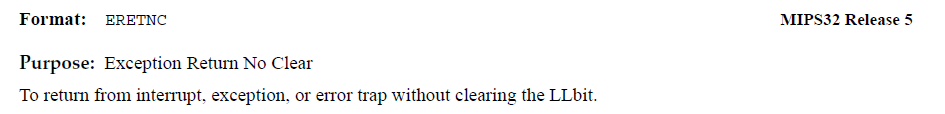
但在一些情况下，若内核能确保例外处理不会引发同步问题，则不会清除，改用eretnc指令。[[1]](#footnote-1)

图1

3. 精确异常：按照指令的编程顺序报异常，且异常产生时，被异常打断的指令前的指令都已经执行完，异常指令和异常指令之后的指令都未执行。

非精确异常： 由于乱序执行，异常指令的上报顺序和编程顺序可能不同。[[2]](#footnote-2)

alpha是一种采用非精确异常的处理器架构。

4. 假设不发生TLB项替换。且a相应地址已存在于tlb中。

地址空间为512KB，128页。共拷贝3次。第一次发生64次TLB Refill和128次TLB Invalid。第二次和第三次不发生TLB例外。共发生192次异常。

将b的值从内存拷贝到寄存器，一开始， TLB查找失败，发生TLB Refill例外，从相应页表位置取出页表内容填入TLB。重新访问，TLB表项找到，但是b尚未初始化，所以无效，发生TLB Invalid例外，为其分配物理页面，并将物理地址填入页表，更新tlb。

设a相应地址已存在于tlb中且tlb足够大，所以把值从寄存器拷贝到a不会发生新的tlb例外。

5.

paddr\_t SearchTLB(vaddr\_t va, int reftype)

{

    int found = 0;

    unsigned pfn;

    char v,c,d;

    for(int i=0;i<TLBMAX;i++)

    {

        if(TLB[i].VPN2 == va.VPN2 && TLB[i].ASID == EntryHi.ASID){

            pfn = va.select ? TLB[i].PFN1 : TLB[i].PFN0;

            v = va.select ? TLB[i].V1 : TLB[i].V0;

            c = va.select ? TLB[i].C1 : TLB[i].C0;

            d = va.select ? TLB[i].D1 : TLB[i].D0;

            if(v==0)

                SignalException(TLBInvalid,reftype);

            if(d==0 && reftype == STORE)

                SignalException(TLBModified);

            paddr\_t pa = pfn << 12 | va.offset;

            found = 1;

            break;

        }

    }

    if(found == 0)

        SignalException(TLBRefill,reftype);

    return pa;

}

1. The MIPS64® Instruction Set Reference Manual, Revision 6.06, Volume II-A.: p250 [↑](#footnote-ref-1)
2. *Simon Peyton Jones et al.* A semantics for imprecise exceptions

   <https://www.microsoft.com/en-us/research/wp-content/uploads/1999/05/except.pdf> [↑](#footnote-ref-2)