清華大學
Tsinghua University

第三单元　层次存储系统

第六讲 MIPS异常及中断

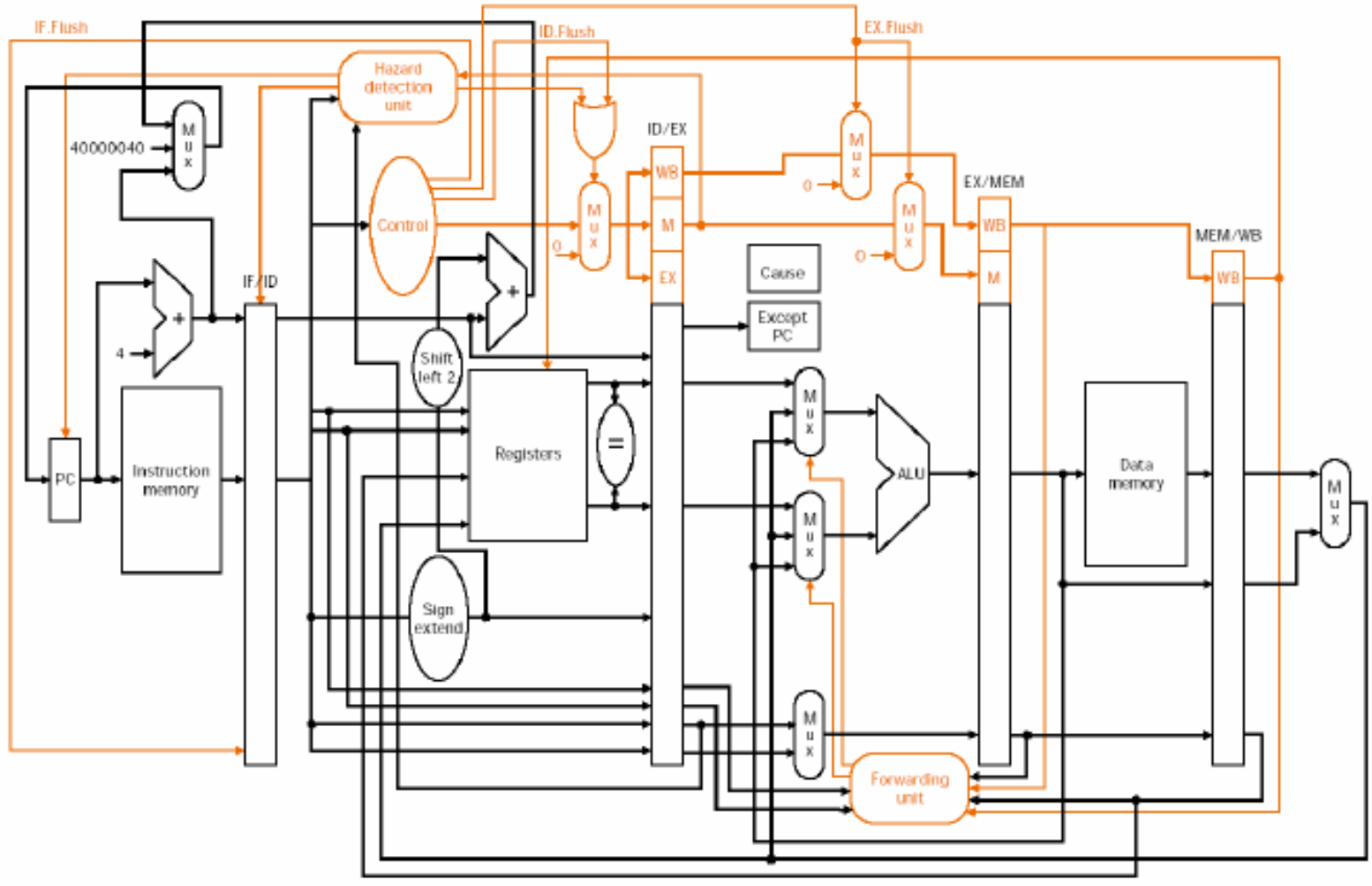
刘卫东

计算机科学与技术系

内容提要

- 异常及中断一般流程
- CP0协处理器的组成
- MIPS异常和中断处理机制
- MIPS虚拟存储管理机制
- TLB设计及维护

流水CPU的异常处理


IF.Flush
ID.Flush
EX.Flush
Hazard detection unit
40000040
M u x
Control
ID/EX
WB
M
EX
EX/MEM
MEM/WB
Cause
Except PC
IF/ID
4
Shift left 2
PC
Instruction memory
Registers
=
ALU
Data memory
Sign extend
Forwarding unit

MIPS异常和中断处理流程

清華大學 Tsinghua University

- 异常和中断检测
  - 异常：来自CPU内部，设置Cause寄存器
  - 中断：来自外部设备
- 异常和中断响应
  - 保留断点：EPC，程序状态字
- 异常和中断处理
  - 保存现场：通用寄存器等
  - 服务程序
- 异常和中断返回
  - 返回到断点处继续执行

寄存器

寄存器

MIPS协处理器CP0

- 用于处理难以用常规指令解决的问题
  - 配置
  - Cache控制
  - 异常/中断控制
  - 存储管理控制
  - 其他事项
- 使用寄存器实现

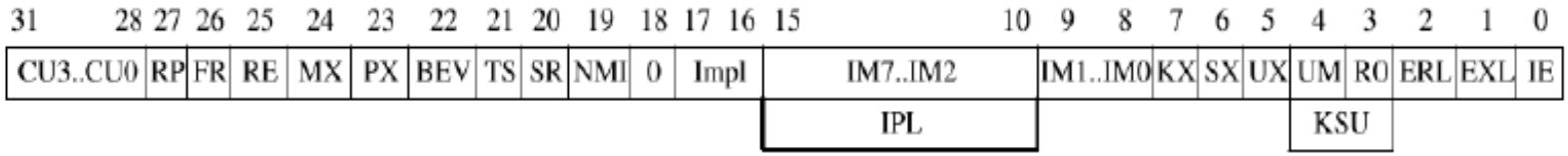
MIPS协处理器CP0

寄存器助记符	CP0寄存器编号	描述
SR	12	状态寄存器(Status Register) 包括确定 CPU 特权等级、哪些中断引脚使能和其它的CPU 模式等位域。
Cause	13	什么原因导致异常或者中断？
EPC	14	异常程序计数器（Exception Program Counter）：异常/中断结束后从哪里重新开始执行。
Count Compare	9 11	这两个寄存器一起形成了一个简单而有用的高分辨率定时器，频率（通常）为 CPU 流水线频率的一半。
BadVaddr	8	导致最近的地址相关异常的程序地址。即使没有 MMU，各种地址错误也都会设置它。
Conext EntryHi EntryLo0–1 Index PageMask Random Wired	4 10 2–3 0 5 1 6	对存储器管理和地址转换硬件(TLB)编程的寄存器

MIPS协处理器CP0

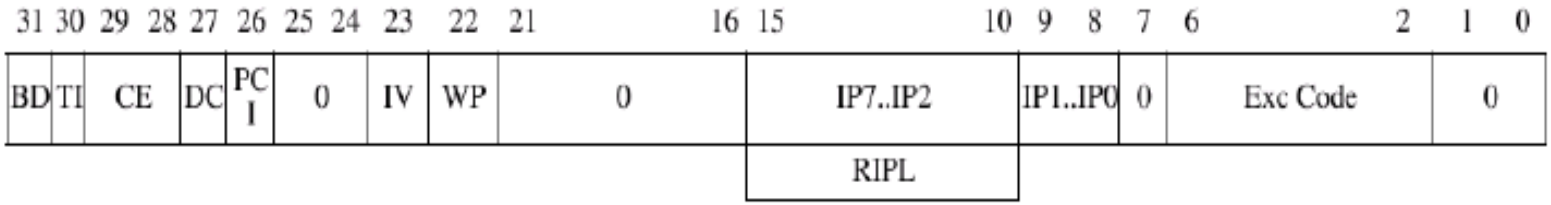
PRId	15	CPU 类型和版本号。类型号由 MIPS 公司管理,(至少)当 CP0 寄存器集变动时应当不同。
Config Config1-3	16 16.1–3	CPU 参数设置,通常由系统决定;一些可写,另一些只读。有些 CPU 有高编号的专用寄存器。
Ebase	15.1	异常入口点基址和—多 CPU 系统的—CPU ID。
IntCtl	12.1	设置中断向量和中断优先级

Status Register



- 状态寄存器
  - BEV:Boot Exception
  - KSU: (0:内核 1:管理 2:用户)
  - IM7..0: 中断屏蔽
  - IE:中断允许
  - EXL: 异常级别,进入内核态,禁止中断
  - ERL: 错误级别,禁止中断

CAUSE Register



- BD：是否在延迟槽中
- IP：中断源
- ExcCode：异常代码

EPC Register

- 异常指令地址
- 中断处理后返回点
- 如何获取?
  - 多周期CPU
    - 中断：指令执行完成后检查中断源，EPC=PC
    - 异常：指令执行错误后，EPC=上一条指令的地址
  - 指令流水CPU
    - 精确异常

BadVaddr Register

- 访存错误发生时虚拟地址
  - 当内存访问出错时保存
  - 可用于后续对TLB的维护

Ebase Register

- 服务程序的统一入口
  - Ebase + Offset

访问CP0寄存器

- 通过两条指令进行CP0寄存器的设置
  - MFC0
  - MTC0

```
mfc0 t0, SR
and  t0, <complement of bits to clear>
or   t0, <bits to set>
mtc0 SR, t0
```

MIPS异常及其分类

- 硬件中断、系统调用以及其他打断程序正常运行流程的事件统称为异常
- 例如：
  - 外部中断
  - 存储地址相关的异常
  - 系统调用（SYSCALL）
  - 计算异常
  - 程序或者硬件检测到的错误

精确异常

- 在处理异常时，产生异常的位置之前的指令都应已经执行完毕，该指令之后的则都不处理
- 需要精确记录异常的位置（指令）
  - 将地址写入到EPC中
  - 如果是延迟槽中的指令呢?
- 需要取消后续指令

MIPS32的异常种类

- 按优先级排列

Reset	由SI_ColdReset信号引起
Soft Reset	由SI_Reset信号引起
DSS	EJTAG调试单步异常
DINT	EJTAG调试中断异常，由外部的EJ_DINT输入引起，或由设置ECR寄存器中的EjtagBrk位引起
NMI	由SI_NMI信号引起
Machine Check	TLB写与一个存在的表项冲突
Interrupt	由未被屏蔽的硬件或软件中断信号引起
Deferred Watch	延迟的观察
DIB	EJTAG调试硬件指令断点匹配
WATCH	访问地址与观察寄存器中的地址匹配（取指时）
AdEL	指令地址对齐错误，或用户模式下访问核心地址空间
IBE	取指时总线错误
DBp	EJTAG断点（执行SDBBP指令）
Sys	执行SYSCALL指令
Bp	执行BREAK指令
CpU	对一个未使能的协处理器执行协处理器指令

MIPS32异常种类（续）

异常	描述
RI	执行保留指令
Ov	算术指令溢出
Tr	执行陷阱指令（陷阱条件为真时）
DDBL / DDBS	EJTAG数据地址断点（只对地址有效），或Store指令的EJTAG数据值断点（对地址和值有效）
WATCH	访问地址与观察寄存器中的地址匹配（访问数据时）
AdEL	读数据地址对齐错误，或用户模式下读核心地址空间数据
AdES	写数据地址对齐错误，或用户模式下写核心地址空间数据
TLBL	读数据/指令时TLB缺失，或TLB无效（有效位为0）
TLBS	写数据时TLB缺失，或TLB无效（有效位为0）
TLB Mod	写TLB错误（写使能位为0）
DBE	读/写数据时总线错误
DDBL	EJTAG数据硬件断点与读指令读出的数据匹配

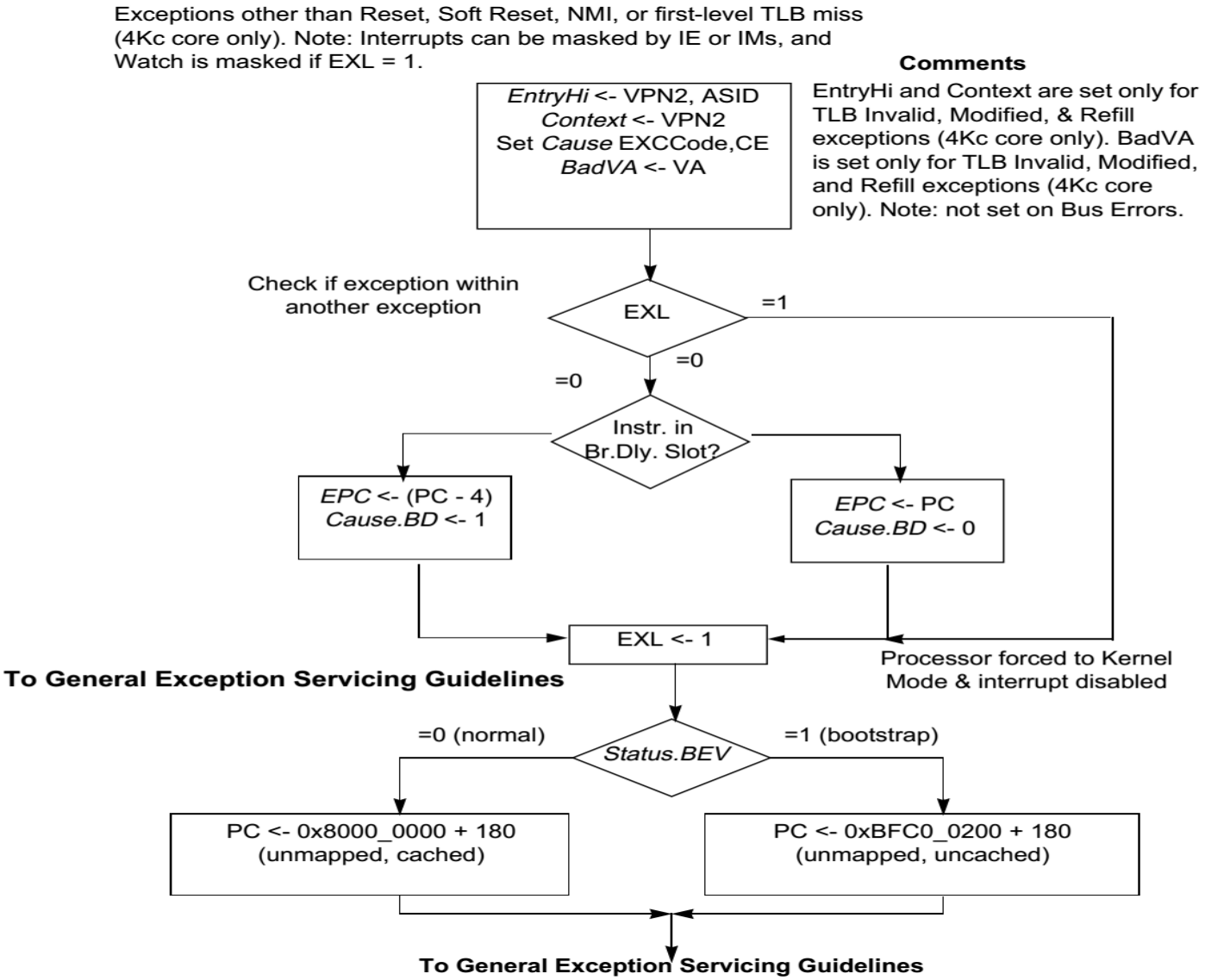
异常处理的基本流程

- 保存现场
  - 在异常程序入口，硬件只记录了被打断程序的很少量的信息，需要保留相关的控制寄存器等值使得异常处理程序能够执行（k0\k1寄存器保留给异常处理代码使用）
- 判断不同的异常
  - 查询Cause寄存器，根据其不同的定义来进行不同的处理流程
- 构造异常处理内存空间
  - 异常处理程序可能由高级语言编写，需要保留通用寄存器，构造堆\栈存储区
- 处理异常
- 准备返回
  - 恢复通用寄存器，将SR寄存器改回原值并返回

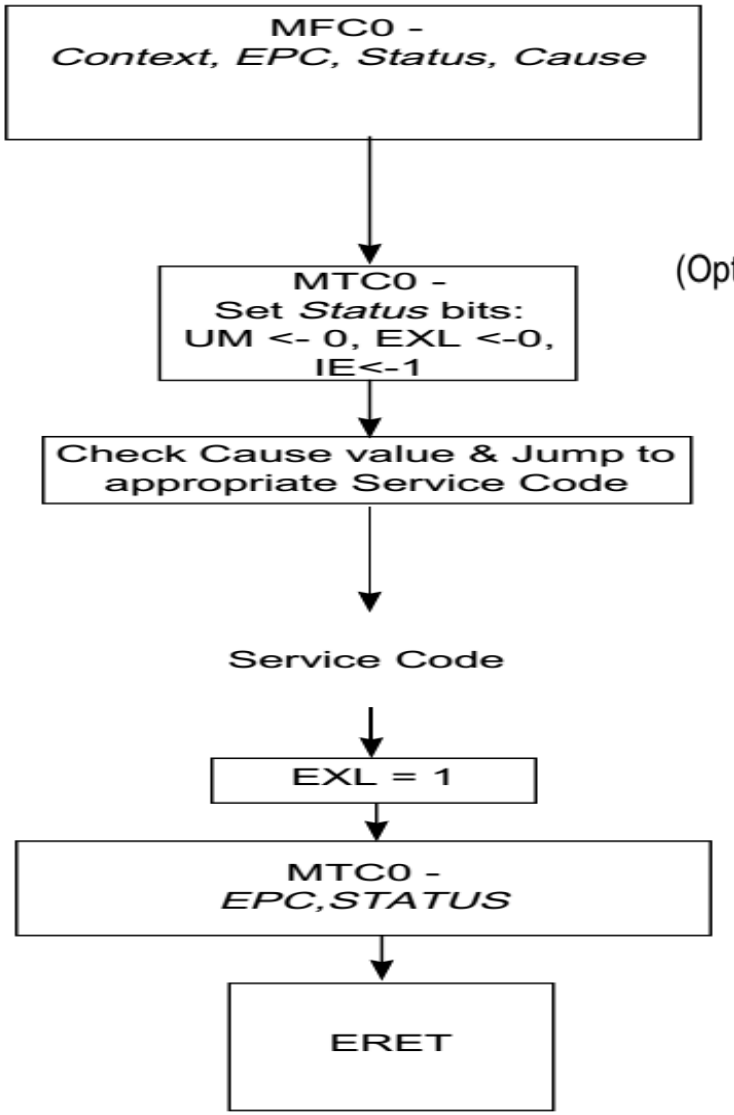
异常返回

- 一般而言，异常处理代码工作在核心态，而被中断的程序是在用户态，所以异常返回意味着状态转换
- 这个转换与指令返回必须“同时”完成，即用一条指令完成
- ERET
  - 返回EPC指向的地址
  - SR寄存器修改

处理流程（硬件部分）


Exceptions other than Reset, Soft Reset, NMI, or first-level TLB miss (4Kc core only). Note: Interrupts can be masked by IE or IMs, and Watch is masked if EXL = 1.
Comments
EntryHi and Context are set only for TLB Invalid, Modified, & Refill exceptions (4Kc core only). BadVA is set only for TLB Invalid, Modified, and Refill exceptions (4Kc core only). Note: not set on Bus Errors.
EntryHi <- VPN2, ASID
Context <- VPN2
Set Cause EXCCode,CE
BadVA <- VA
Check if exception within another exception
EXL
=1
=0
=0
Instr. in Br.Dly. Slot?
EPC <- (PC - 4)
Cause.BD <- 1
EPC <- PC
Cause.BD <- 0
EXL <- 1
Processor forced to Kernel Mode & interrupt disabled
To General Exception Servicing Guidelines
=0 (normal)
Status.BEV
=1 (bootstrap)
PC <- 0x8000_0000 + 180
(unmapped, cached)
PC <- 0xBFC0_0200 + 180
(unmapped, uncached)
To General Exception Servicing Guidelines

处理流程（软件部分）



(Optional - only to enable Interrupts while keeping Kernel Mode)

* After EXL=0, all exceptions allowed.
(except interrupt if masked by IE)

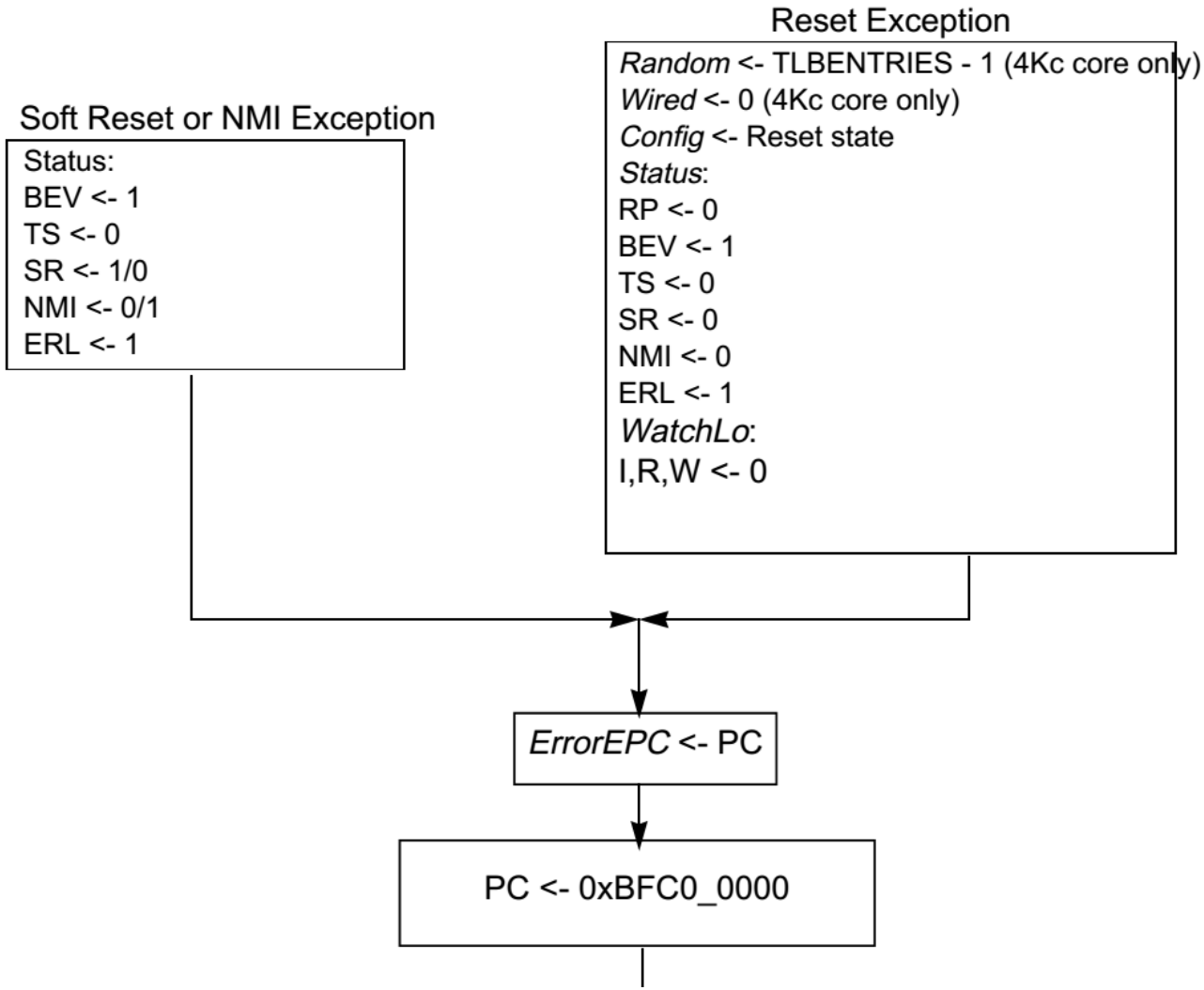
中断

- 中断是异步发生的，是来自处理器外部的I/O设备的信号的结果。
- 硬件中断不是由任何一条专门的指令造成的
  - I/O设备，比如网络适配器，磁盘控制器等通过处理器芯片上的一个引脚发送信号，并将异常号放到系统总线上，用来触发中断，这个异常号标识了引起中断的设备
- MIPS处理器的中断控制设计
  - 在中断发生时，如果该指令已经完成了MEM段的操作，则保证该指令执行完毕
  - 反之，则丢弃流水线上这条指令的工作

启动过程

- 当处理器收到*SI_ColdReset*信号:处理器执行完全的复位初始化，包括放弃当前状态机、建立临界状态
  - 使得处理器处于可以从非缓存、非映射地址空间执行指令的状态
- 在复位异常中，处理器的状态是未定义的，但以下寄存器需要初始化
  - Random寄存器被初始化为TLB表项条数减1;
  - Wired寄存器初始化为0;
  - Config寄存器初始化为引导时的状态;
  - Status寄存器的RP、BEV、TS、SR、NMI及ERL域初始化到特定的状态;
  - WatchLo寄存器的I、R、W域初始化为0;
  - ErrorEPC寄存器中装入PC;
  - PC中装入0xBFC0_0000

启动过程

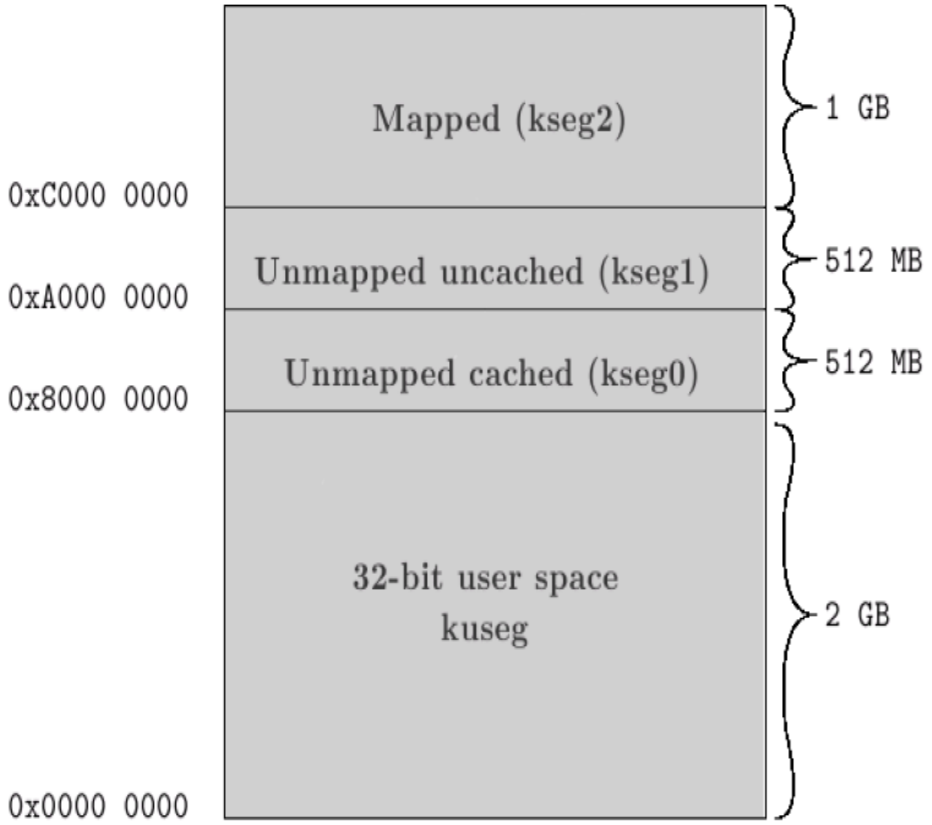

Reset Exception
Random <- TLBENTRIES - 1 (4Kc core only)
Wired <- 0 (4Kc core only)
Config <- Reset state
Status:
RP <- 0
BEV <- 1
TS <- 0
SR <- 0
NMI <- 0
ERL <- 1
WatchLo:
I,R,W <- 0
Soft Reset or NMI Exception
Status:
BEV <- 1
TS <- 0
SR <- 1/0
NMI <- 0/1
ERL <- 1
ErrorEPC <- PC
PC <- 0xBFC0_0000

异常嵌套

- 很多情况下，需要（或者是不可避免）允许在异常处理时发生另一个异常，这叫做异常嵌套
  - 如果只是简单的处理，会产生混乱，因为被中断的程序的关键状态存放在EPC、STATUS、CAUSE等寄存器中，并且另一个异常会立即修改这几个寄存器。所以在允许下一级嵌套异常之前，必须保存这些值。另外，一旦重新“使能”了异常，就不能再依赖k0和k1的值了。
- 嵌套异常的服务程序必须用一些主存空间来保存寄存器的值，使用的数据结构叫异常帧，多个嵌套异常的异常帧通常保存在栈中
  - 每一个异常都会消耗栈资源，所以任意深度的嵌套异常是不可容忍的
  - 大多数系统赋予每一类异常一个优先级，并且规定：当一个异常正在被服务时，只有高优先级的异常才被允许。这样的系统只需要和优先级数一样多的异常帧存储空间就够了

MIPS32存储管理

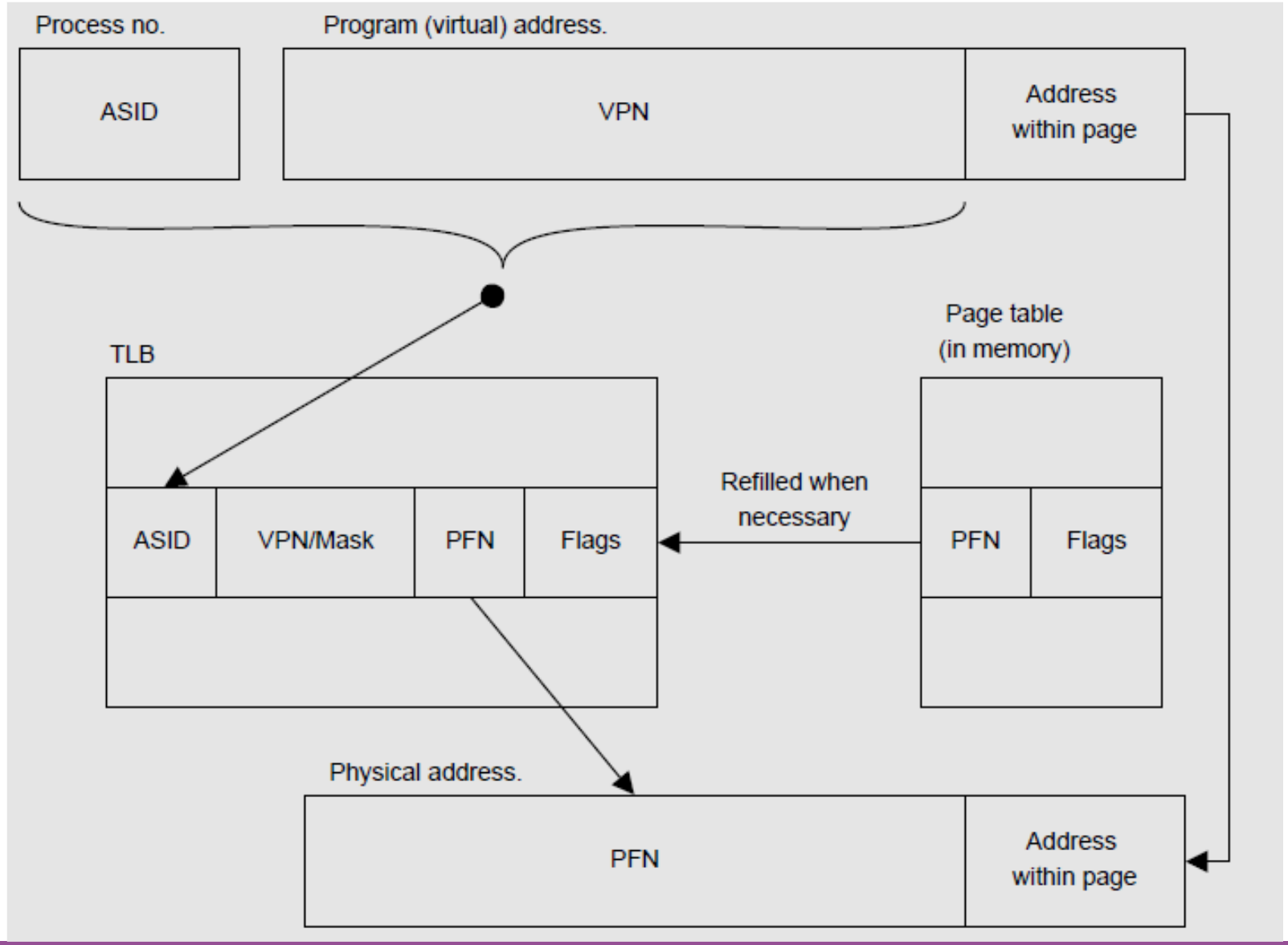
- 虚拟地址空间划分
  - Kuseg
    - 用户态可以使用的地址，需经过MMU转换
  - Kseg0
    - 最高位清零后就是物理地址
  - Kseg1
    - 最高三位清零后就是物理地址
  - Kseg2
    - 核心态使用，需经过MMU转换



虚实地址转换

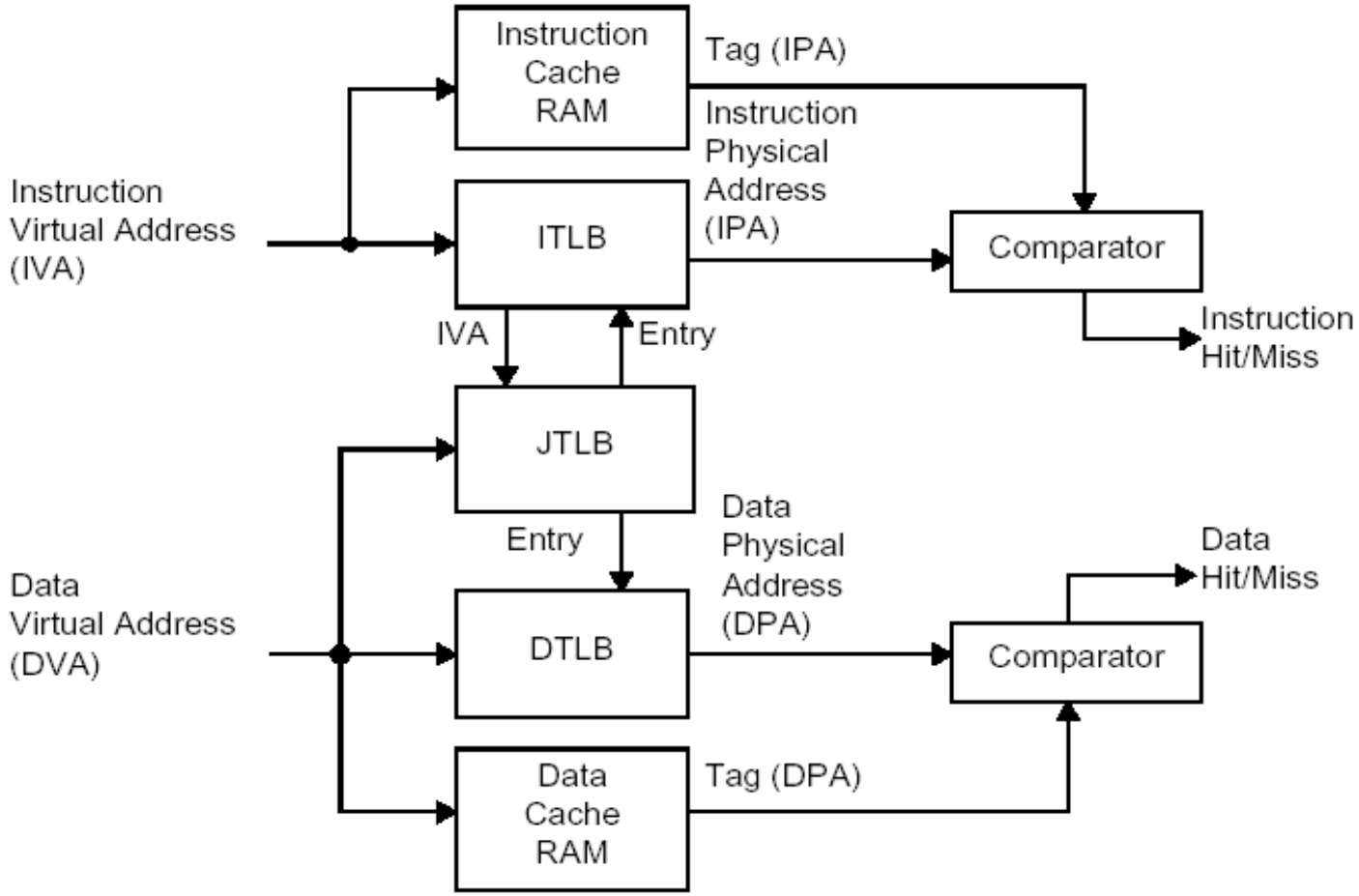
- 程序运行的级别
  - 用户态
  - 核心态
  - 调试态
- 存储管理需求
  - 程序装入的重定位
  - 动态申请空间
  - 隐藏和保护
  - 共享
  - 请求页
- 虚实地址转换
  - 任何的处理器地址的访问（包括指令与数据）都需要经过MMU的地址转换，即程序（虚）地址→物理地址

地址转换机制


Process no.
Program (virtual) address.
ASID
VPN
Address within page
TLB
Page table (in memory)
ASID
VPN/Mask
PFN
Flags
Refilled when necessary
PFN
Flags
Physical address.
PFN
Address within page

基于TLB的地址转换

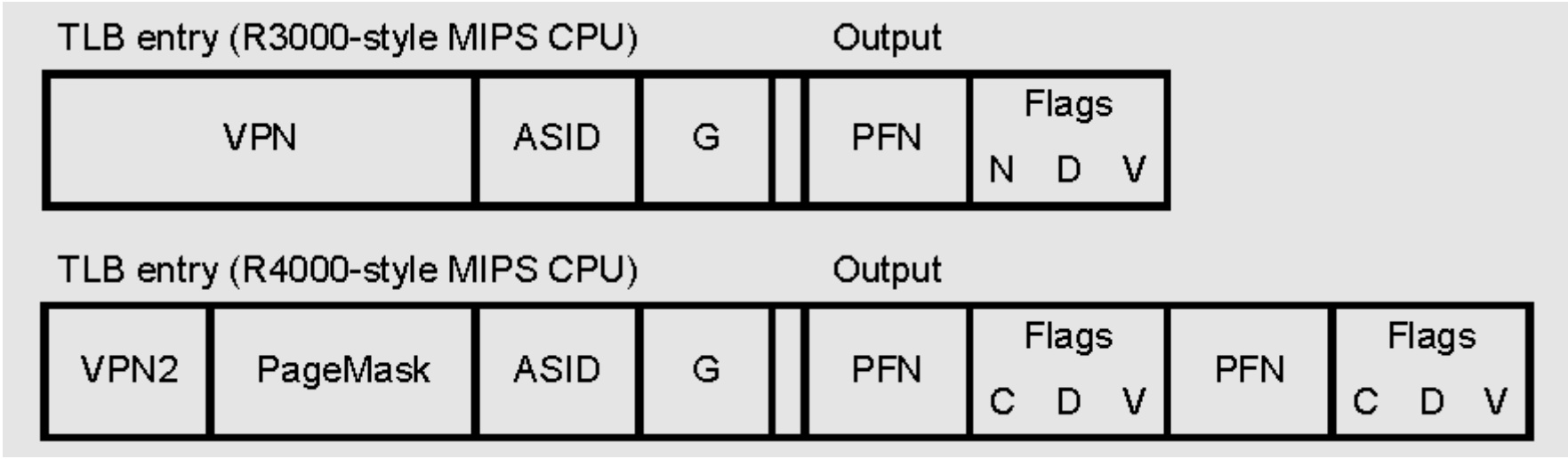
- 设置MMU，基于TLB来进行地址转换

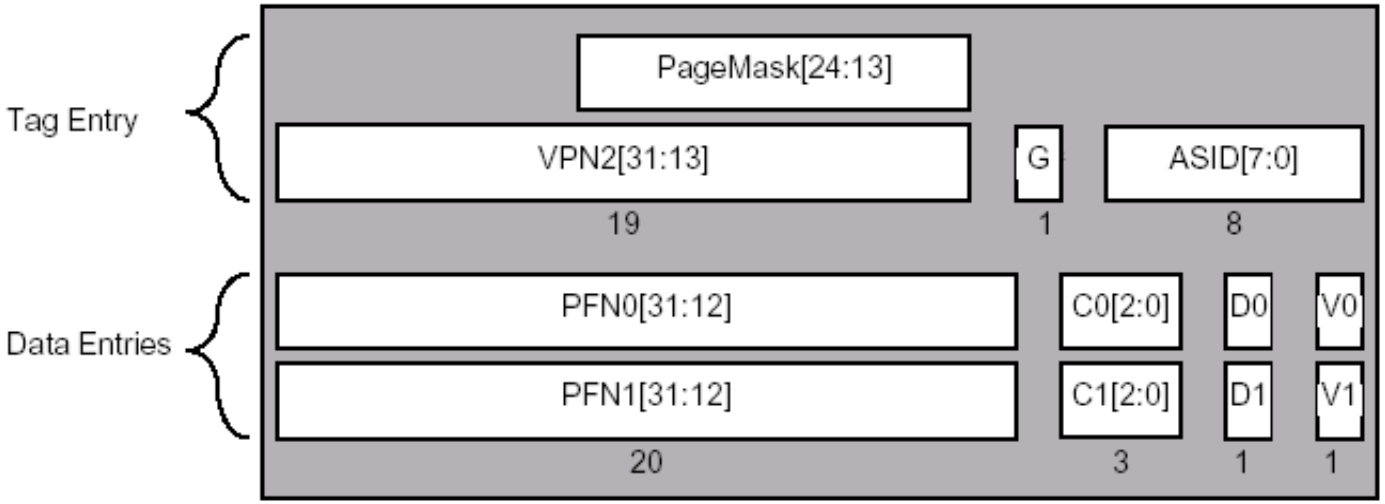


需要解决的问题

- TLB组成和结构设计
- TLB维护指令
- TLB Refill exception

TLB结构


TLB entry (R3000-style MIPS CPU)
Output
VPN
ASID
G
PFN
Flags
N D V
TLB entry (R4000-style MIPS CPU)
Output
VPN2
PageMask
ASID
G
PFN
Flags
C D V
PFN
Flags
C D V


Tag Entry
Data Entries
PageMask[24:13]
VPN2[31:13]
G
ASID[7:0]
19
1
8
PFN0[31:12]
C0[2:0]
D0
V0
PFN1[31:12]
C1[2:0]
D1
V1
20
3
1
1

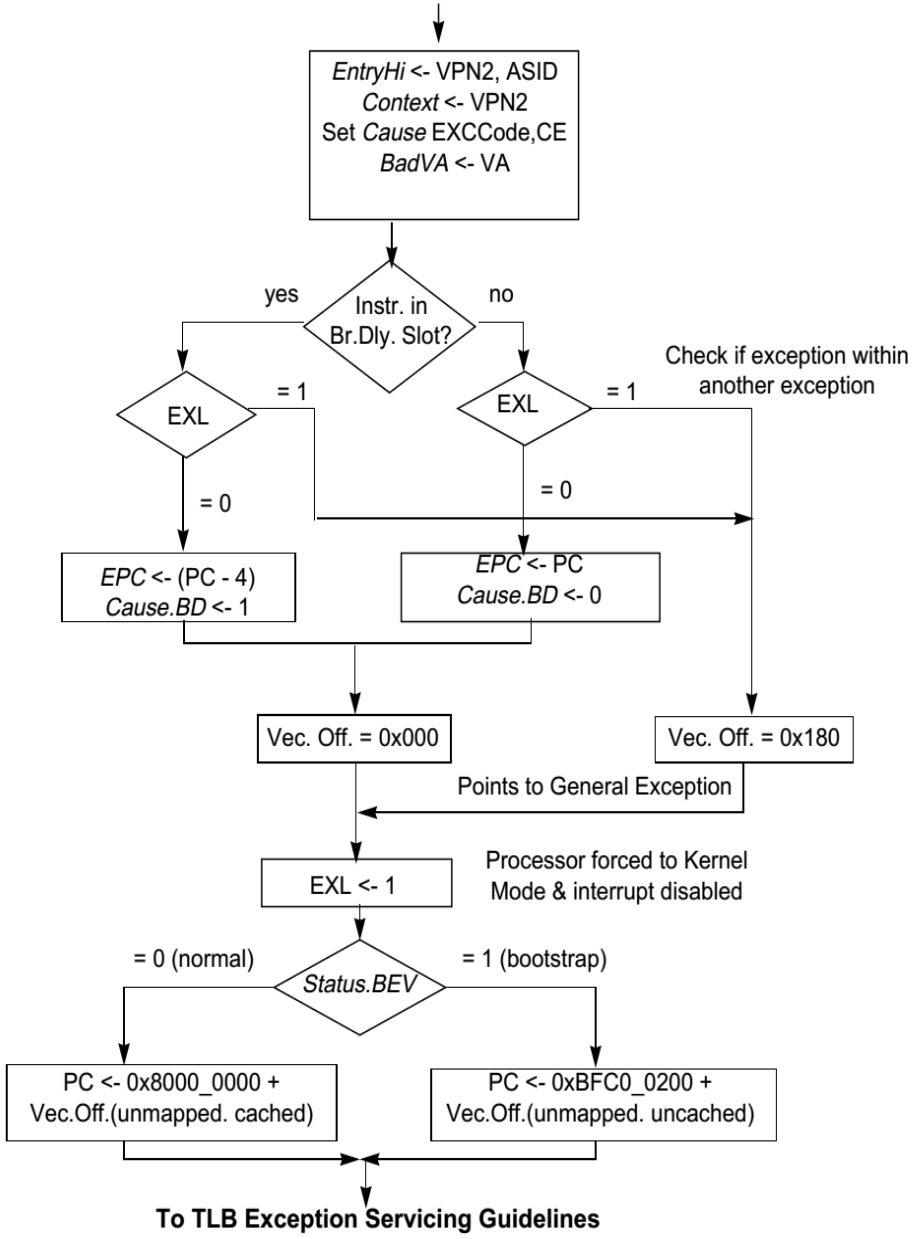
TLB设置

- TLB通过CP0的寄存器进行维护
  - Index
  - EntryHi
  - EntryLo0
  - EntryLo1
- 相关指令
  - MFC0 MTC0 TLBWI

TLB维护

- JTLB维护
  - 软件通过TLBWI和TLBWR指令填充JTLB的某个表项
  - 在填充TLB之前，首先用CP0指令更新和TLB相关的CP0寄存器，然后用相关CP0寄存器的内容填充TLB入口的不同字段
    - PageMask： PageMask register
    - VPN2 and ASID： EntryHi register
    - PFN0, C0, D0, V0 and G bit： EntryLo0 register
    - PFN1, C1, D1, V1 and G bit： EntryLo1 register
- ITLB及DTLB
  - 完全硬件管理，对软件透明
  - 相当于JTLB的“cache”

TLB异常的处理流程



- TLB异常——取指令和数据访问时，如果出现如下情况则会发生TLB异常
  - 在使用TLB的存储管理系统中没有TLB表项可以匹配，状态寄存器的EXL位被置为1
  - 在使用TLB的存储管理系统中有TLB表项可以匹配，但其合法位指示该表项无效
  - 虚存地址大于等于一个固定映射（fixed-mapping）存储管理器的上界

小结

- 协处理器CP0
- 中断及异常处理过程
- 存储管理实现方法
- 阅读
  - 《See MIPS Run》