计算机组成原理
CH7_异常与特权

Computer Organization and Design

Chapter 7: Exception and Privilege

马子睿

目录

PPT翻页那一刻

处理器都做了什么

例外与中断

Exception and Interrupt

例外？中断？异常？

- **例外**
  - **定义**：在LA32R中，例外（Exception）是在处理器内部产生的同步信号，用以打断当前指令流，从而使处理器转移去执行其他实现特定功能的程序。
  - **特点**：伴随某条指令的执行而产生，是同步于处理器时钟的信号。
  - **例子**：
    - 地址非对齐例外：ld.w $r2, 0x1c000002
    - 系统调用例外：syscall
    - 指令不存在例外：Instruction Code = 0x00000000

LA32R非浮点架构中的例外

- 0x1~0x7：页表相关例外
- 0x8-0：PC未能4字节对齐
- 0x8-1：在用户模式下load/store最高位为1的地址
- 0x9：字访存未4字节对齐，或半字访存未2字节对齐
- 0xb：执行syscall指令
- 0xc：执行break指令
- 0xd：执行未在La32R架构中的指令
- 0xe：在用户模式下执行特权指令
- 0x3f：TLB缺失

例外不能脱离指令的存在而存在！

ECode	ESubCode	类型
0x1	0	load操作页无效
0x2	0	store操作页无效
0x3	0	取指操作页无效
0x4	0	页修改例外
0x7	0	页特权等级不合规
0x8	0	取指地址错
0x8	1	访存指令地址错
0x9	0	地址非对齐
0xb	0	系统调用
0xc	0	断点
0xd	0	指令不存在
0xe	0	指令特权级错
0x3f	0	TLB重填

例外？中断？异常？

- **中断**
  - **定义**：中断（Interrupt）是在处理器外部或内部产生的异步或同步信号，用以打断当前指令流，从而使处理器转移去执行其他实现特定功能的程序。
  - **特点**：可能由特权指令同步触发，但主要由外围设备异步触发。
  - **例子**：
    - 定时器中断
    - 硬中断：按下键盘、鼠标后传递给处理器的电信号
    - 软中断：特权模式下，特权指令写特定寄存器触发

LA32R中的中断

- 中断也有自己的异常编码：0x0
- 中断的具体类型使用独热码编码，称为“中断向量”：
  - 第0~1位：软件可写入1来触发中断
  - 第2~9位：对来自核外的8个中断源的采样结果
  - 第11位：对定时器的中断信号的采样结果
  - 第12位：来自其他处理器核的中断信号

ECode	ESubCode	类型
0x0	0	中断

IS（One Hot）	中断类型
0x0001~0x0002	软件中断（2个）
0x0004~0x0200	硬中断（8个）
0x0800	定时器中断
0x1000	核间中断

即便不存在指令流，中断依然可以产生！

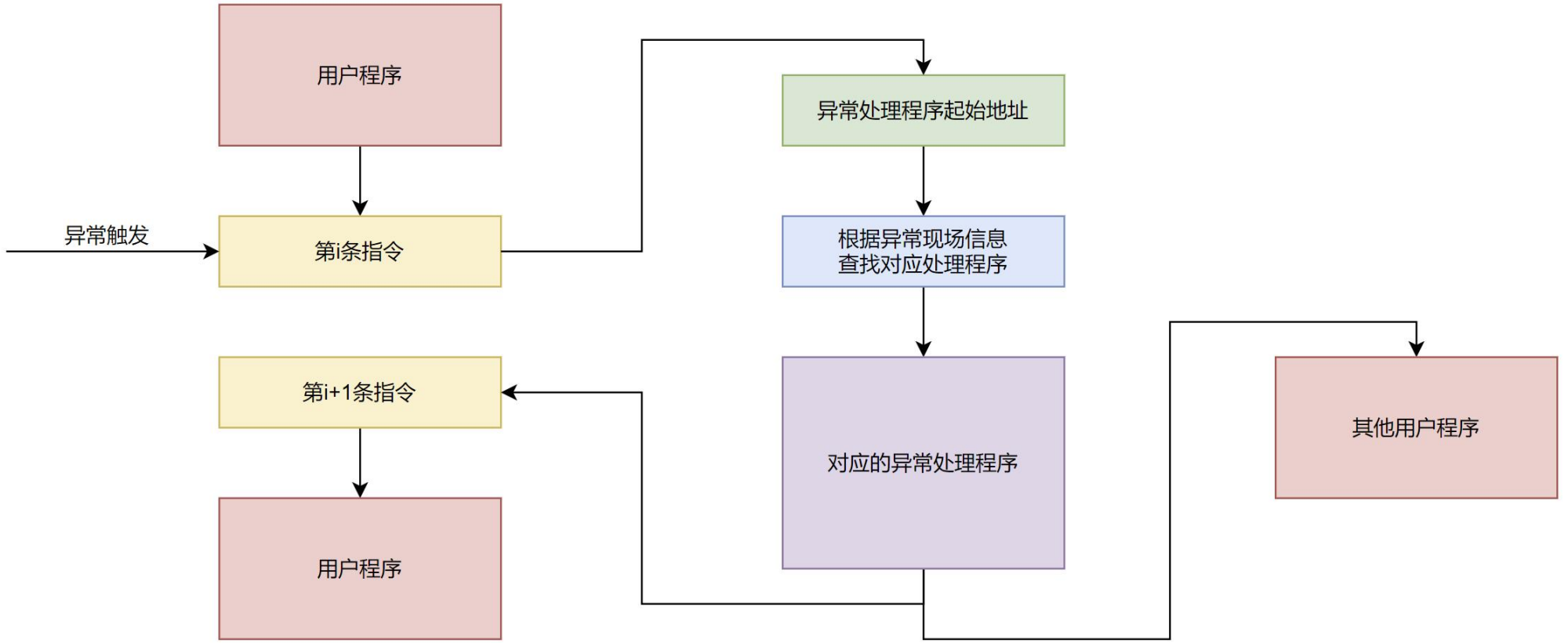
例外？中断？异常？

- **异常**
  - **定义**：在LA32R中，例外和中断统称为异常。

- 注意
  - 在不同的架构中，例外、中断、异常经常被混用，例如在RISCV中，异常和中断时同一级别的，而不存在“例外”这个名词。
  - “中断”和“例外”的划分界线并不是一成不变的，有些架构认为由外围设备触发的才算“中断”，而“例外”是由处理器内部触发的。

异常处理行为

- 当用户程序执行时，异常可能随时触发，这时处理器会直接跳转到异常处理程序的起始地址，这段程序会根据处理器保存下来的信息选择执行对应的处理代码。
- 处理完成后，指令流可能会返回用户程序，也可能会进入其他应用程序。



多个异常同时触发的优先级

例外优先级遵循两个基本原则：其一，中断的优先级高于例外；其二，对于例外，取指阶段检测出的优先级最高，译码阶段检测出的优先级次之，执行阶段检测出的优先级再次之。

对于取指阶段检测出的例外：取指地址错例外优先级最高，取指 TLB 相关例外优先级次之。

译码阶段可检测出的例外彼此互斥，故无需考虑其间的优先级。

执行阶段仅访存指令或同时触发多种例外，其优先级从高到低依次为：要求地址对齐的访存指令因地址不对齐而产生的地址对齐错例外(ALE) > TLB 相关的例外。

——摘录自《龙芯架构32位精简版参考手册》v1.0.3

异常是坏事吗？

- **对计算机发展来说**：
  - 例外让计算资源的跨级访问成为了可能：用户程序可以通过例外进入操作系统，从而获取到它所需要的资源（例如printf）。
  - 中断为操作系统的并发操作提供了保障：操作系统可以通过定时器中断、外围设备中断来进行进程调度、外围设备访问等操作。

- **对计算机设计者来说**：
  - 至少算不上什么好事，因为异常对于流水线设计的冲击是很大的。

异常处理需要解决的问题

- 处理器如何对异步中断进行采样和记录?
- 处理器如何记录例外和中断的编码?
- 异常处理程序的起始地址保存在哪里?
- 异常发生时的现场如何保护？如何返回异常触发地?
- 在执行异常处理程序的过程中再次发生中断怎么办?

处理器特权架构

Processor Privilege Architecture

什么是“特权”？

- **不是谁都能访问的**
  - 在我们之前接触的架构中，通用寄存器、内存等资源是在任何情况下可以被任何指令访问的，但处理器中还有许多重要信息，我们不希望它们被轻易访问、修改：
    - 当前的机器模式（用户态 or 核心态）
    - 异常处理程序的起始地址
    - 内存映射方式（直接翻译 or 映射翻译）
  - “特权”信息依然需要能够被操作系统访问、修改，我们需要定义一种特殊的方式，让操作系统能够操作这些特权信息。

控制状态寄存器（CSR）

- **CSR vs GPR**
  - 和通用寄存器（GPR）相同，每个CSR也有自己的编号，可以使用编号进行访问。
  - 和GPR不同，CSR只能在处理器核心态使用CSR访问指令进行访问、修改，如果程序试图在用户态执行一条CSR访问指令，则处理器会触发“指令特权级错”例外。
  - 每个CSR的每一个位都在架构层面有自己独特的意义，且并非只有CSR访问指令可以修改其值。
- **CSR记录的信息**
  - CSR记录了当前处理器的一些重要状态，包括但不限于：当前特权等级、上一特权等级，异常处理程序入口地址，当前例外编号、当前中断向量，中断总使能、各中断使能，内存地址映射方式，触发例外的指令地址。

LA32R CSR访问指令

- **csrrd　　rd，csr_num**
  - CSR读取指令
  - 将编号为csr_num的CSR中的值读取到rd通用寄存器中
- **csrwr　　rd，csr_num**
  - CSR读写指令
  - 将编号为csr_num的CSR中值与编号为rd的通用寄存器中的值进行交换
- **csrxchg　　rd，rj，csr_num**
  - CSR掩码交换指令
  - 将编号为csr_num的CSR中值更新到rd中，并以rj中值为掩码，使用rd值更新CSR

LA32R中与异常相关的部分CSR

- **CRMD：当前模式信息**
  - 第0~1位：当前特权等级
  - 第2位：全局中断使能
  - 第3位：直接地址翻译模式使能
  - 第4位：映射地址翻译模式使能
  - 第5~6位：直接地址翻译模式下，取指操作的存储访问类型
  - 第7~8位：直接地址翻译模式下，访存指令的存储访问类型
  - 第9~31位：始终为0，软件对其修改无效

位	名字	读写	描述
1:0	PLV	RW	当前特权等级。其合法的取值范围为 0 和 3。其中 0 表示最高特权等级，3 表示最低特权等级。 当触发例外时，硬件将该域的值置为 0，以确保陷入后处于最高特权等级。 当执行 ERTN 指令从例外处理程序返回时，硬件将 CSR.PRMD 的 PPLV 域的值恢复到这里。
2	IE	RW	当前全局中断使能，高有效。 当触发例外时，硬件将该域的值置为 0，以确保陷入后屏蔽中断。例外处理程序决定重新开启中断响应时，需显式地将该位置 1。 当执行 ERTN 指令从例外处理程序返回时，硬件将 CSR.PRMD 的 PIE 域的值恢复到这里。
3	DA	RW	直接地址翻译模式的使能，高有效。 当触发 TLB 重填例外时，硬件将该域置为 1。 当执行 ERTN 指令从例外处理程序返回时，如果 CSR.ESTAT.Ecode=0x3F，则硬件将该域置为 0。 DA 位和 PG 位的合法组合情况为 0、1 或 1、0，当软件配置成其它组合情况时结果不确定。
4	PG	RW	映射地址翻译模式的使能，高有效。 当触发 TLB 重填例外时，硬件将该域置为 0。 当执行 ERTN 指令从例外处理程序返回时，如果 CSR.ESTAT.Ecode=0x3F，则硬件将该域置为 1。 PG 位和 DA 位的合法组合情况为 0、1 或 1、0，当软件配置成其它组合情况时结果不确定。
6:5	DATF	RW	直接地址翻译模式时，取指操作的存储访问类型。 当软件将 PG 置为 1 时，推荐同时将 DATF 域置为 0b01，即一致可缓存类型。
8:7	DATM	RW	直接地址翻译模式时，load 和 store 操作的存储访问类型。 当软件将 PG 置为 1 时，推荐同时将 DATM 置为 0b01，即一致可缓存类型。
31:9	0	R0	保留域。读返回 0，且软件不允许改变其值。

LA32R中与异常相关的部分CSR

- **PRMD：例外前模式信息**
  - 记录了例外触发那一刻的CRMD
- **ECFG：例外控制**
  - 每个中断源的中断使能
- **ESTAT：例外状态**
  - 对每个中断源的中断信号进行周期性采样，并记录当前例外的编码
- **ERA：例外返回地址**
  - 记录了触发例外的指令地址（PC）
- **BADV：出错虚地址**
  - 记录了触发例外的虚拟地址
- **EENTRY：例外入口地址**
  - 记录了异常处理程序的入口地址

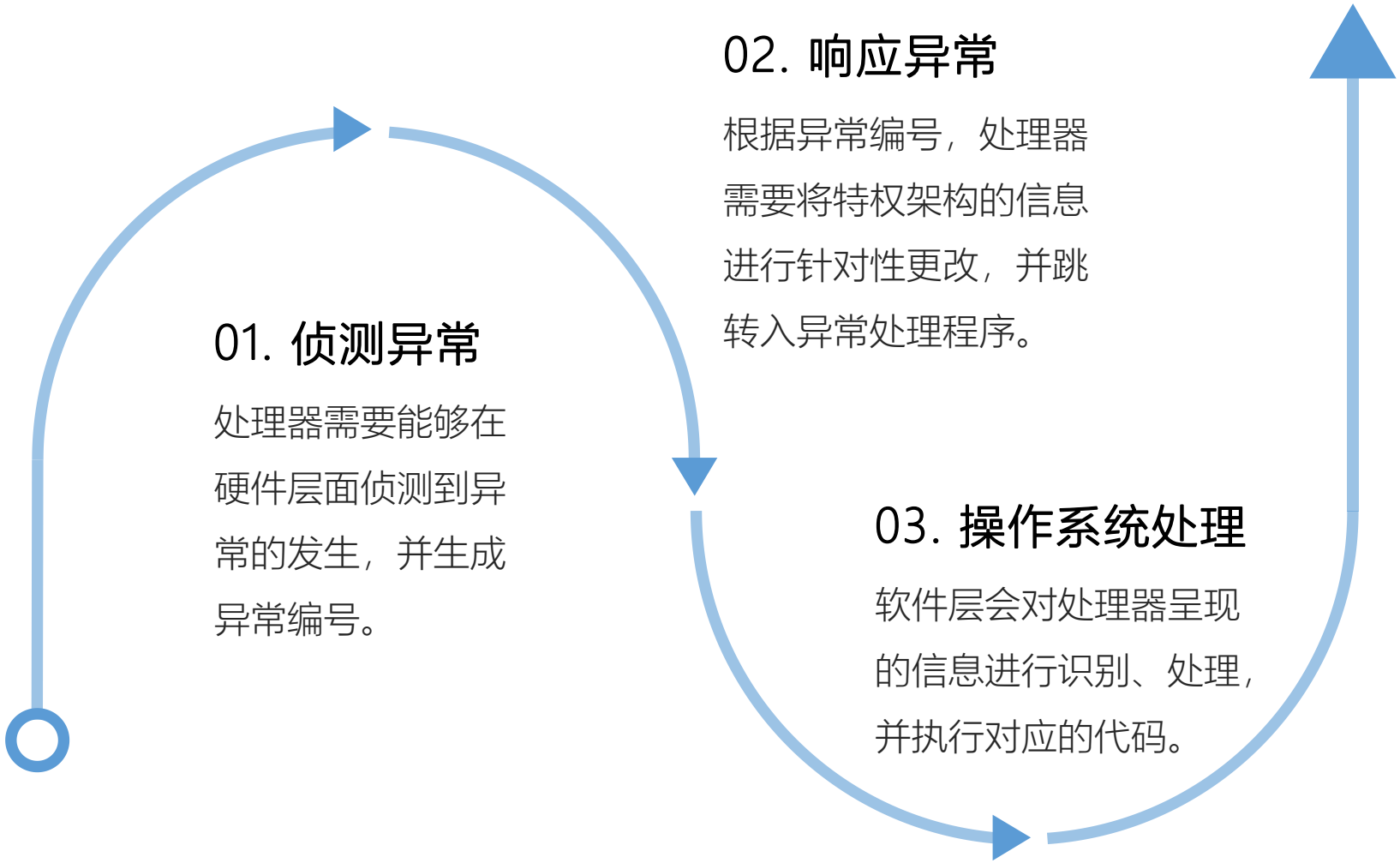
LA32R异常响应流程

Exception Response Process of LA32R

异常处理对流水线的影响

- **CSR的加入与CSR指令的实现**
  - CSR写指令无法流水化，这是因为CSR记录了流水线的特征信息（例如内存地址映射方式），这种更改可能会导致已经进入流水线的后续指令从取指开始就是错误的。
  - 流水线的每个阶段都会产生异常，每个阶段都需要能侦测异常。
- **异步中断同步化问题**
  - 中断虽然不依赖指令，但处理器需要找到一条指令，使其与中断一起同步提交，否则我们将无法记录中断的现场。
- **异常指令流重定向问题**
  - 在经典的五级流水线中，如果在WB端提交异常，那么可能下一条store指令已经执行，这会给存储器带来不可挽回的后果。

异常处理流程



01. 侦测异常

处理器需要能够在硬件层面侦测到异常的发生，并生成异常编号。

02. 响应异常

根据异常编号，处理器需要将特权架构的信息进行针对性更改，并跳转入异常处理程序。

03. 操作系统处理

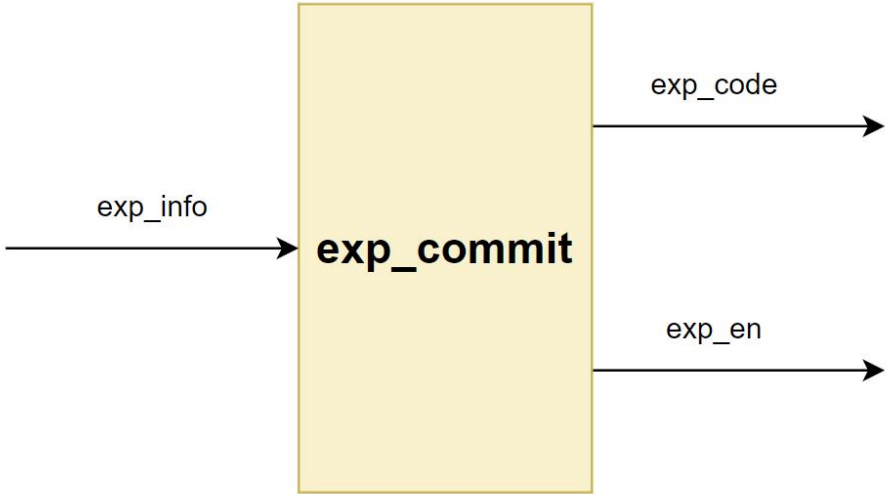
软件层会对处理器呈现的信息进行识别、处理，并执行对应的代码。

04. 异常返回

返回异常现场重新执行，或跳转到其他位置。

侦测异常

- 每个流水级都会根据当前流水级的执行结果，判断是否出现异常
- 随着异常在流水线中流动，每个流水级会根据上一流水级报出的异常、本流水级报出的异常来进行优先级比较。
- 当指令到达WB段时，一个硬件编码模块会对该指令可能的异常进行权衡，并生成最终的异常码和异常使能。



响应异常

01 保存处理器状态

将CRMD的特权级、中断使能存入PRMD，然后将CRMD的特权级置为内核态，中断使能置为无效。

02 保存案发现场

将触发异常的指令的PC值保存到ERA中，从而记录异常返回地址

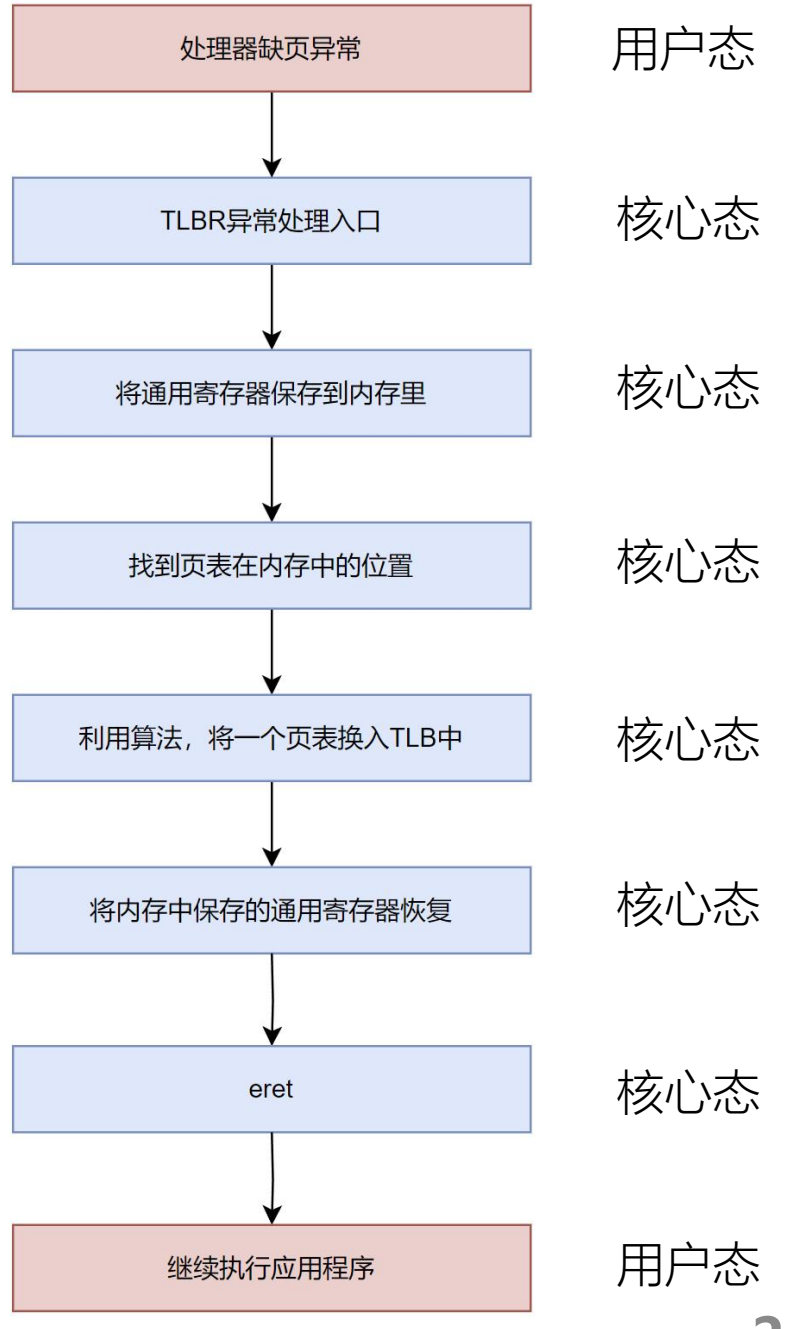
03 进入异常处理程序

将PC重定向到EENTRY存储的异常处理程序入口地址上。

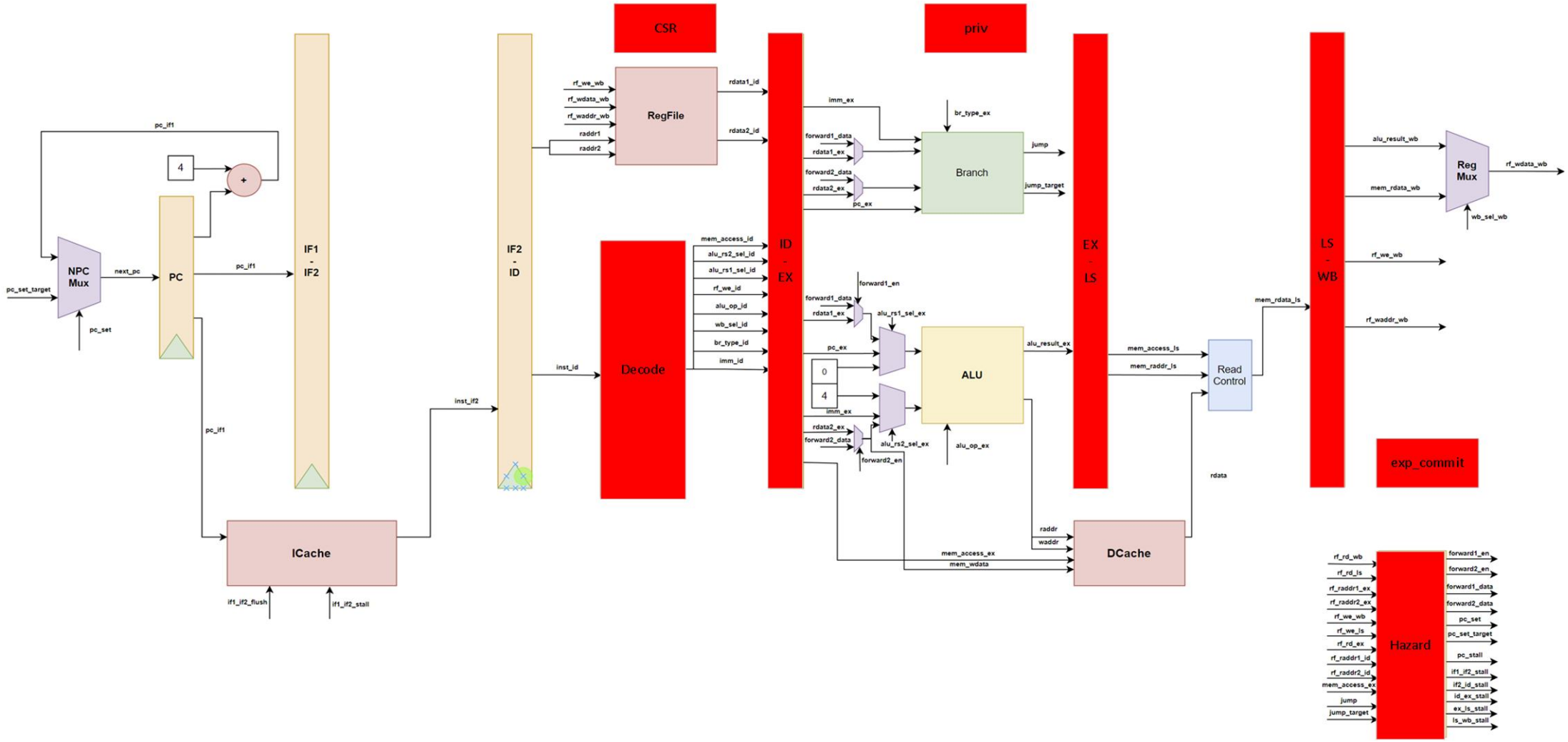
一旦触发异常，LA32R架构会立刻进入内核态（将控制权交还操作系统），并无效掉中断使能，使其不能嵌套。

操作系统处理 & 异常返回

- **异常处理程序的行为**
  - 异常处理程序会从ESTAT寄存器中读出当前异常的类型，并在一个记录了所有类型异常的个性化处理程序入口的表中查找其对应入口
  - 保存异常触发时的通用寄存器
  - 执行过对应的个性化处理程序后
  - 恢复异常触发时的通用寄存器
  - 操作系统会使用eret指令，将era的值复原回pc中，并恢复特权级和中断使能



包含特权指令的流水线

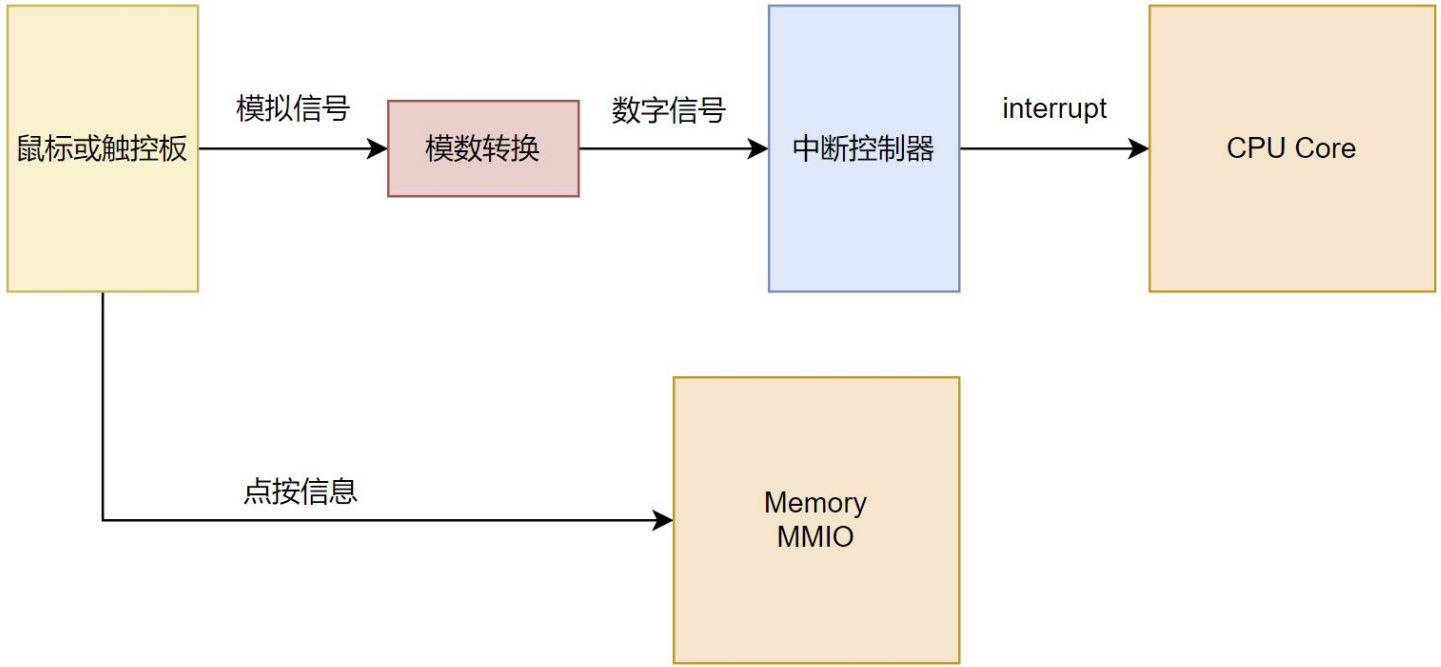

CSR
priv
RegFile
Branch
Decode
ALU
ICache
DCache
Read Control
Reg Mux
Hazard
exp_commit
NPC Mux
PC
IF1 - IF2
IF2 - ID
ID - EX
EX - LS
LS - WB
pc_if1
pc_set_target
pc_set
next_pc
inst_if2
inst_id
if1_if2_flush
if1_if2_stall
rf_we_wb
rf_wdata_wb
rf_waddr_wb
raddr1
raddr2
rdata1_id
rdata2_id
imm_ex
br_type_ex
forward1_data
rdata1_ex
forward2_data
rdata2_ex
pc_ex
jump
jump_target
mem_access_id
alu_rs2_sel_id
alu_rs1_sel_id
rf_we_id
alu_op_id
wb_sel_id
br_type_id
imm_id
forward1_en
alu_rs1_sel_ex
alu_result_ex
alu_op_ex
alu_rs2_sel_ex
forward2_en
mem_access_ex
mem_wdata
raddr
waddr
mem_access_ls
mem_raddr_ls
mem_rdata_ls
rdata
alu_result_wb
mem_rdata_wb
wb_sel_wb
rf_wdata_wb
rf_we_wb
rf_waddr_wb
rf_rd_wb
rf_rd_ls
rf_raddr1_ex
rf_raddr2_ex
rf_we_ls
rf_rd_ex
rf_raddr1_id
rf_raddr2_id
mem_access_ex
forward1_en
forward2_en
forward1_data
forward2_data
pc_set
pc_set_target
pc_stall
if1_if2_stall
if2_id_stall
id_ex_stall
ex_ls_stall
ls_wb_stall

PPT翻页那一刻……

At the moment of flipping the PPT slide...

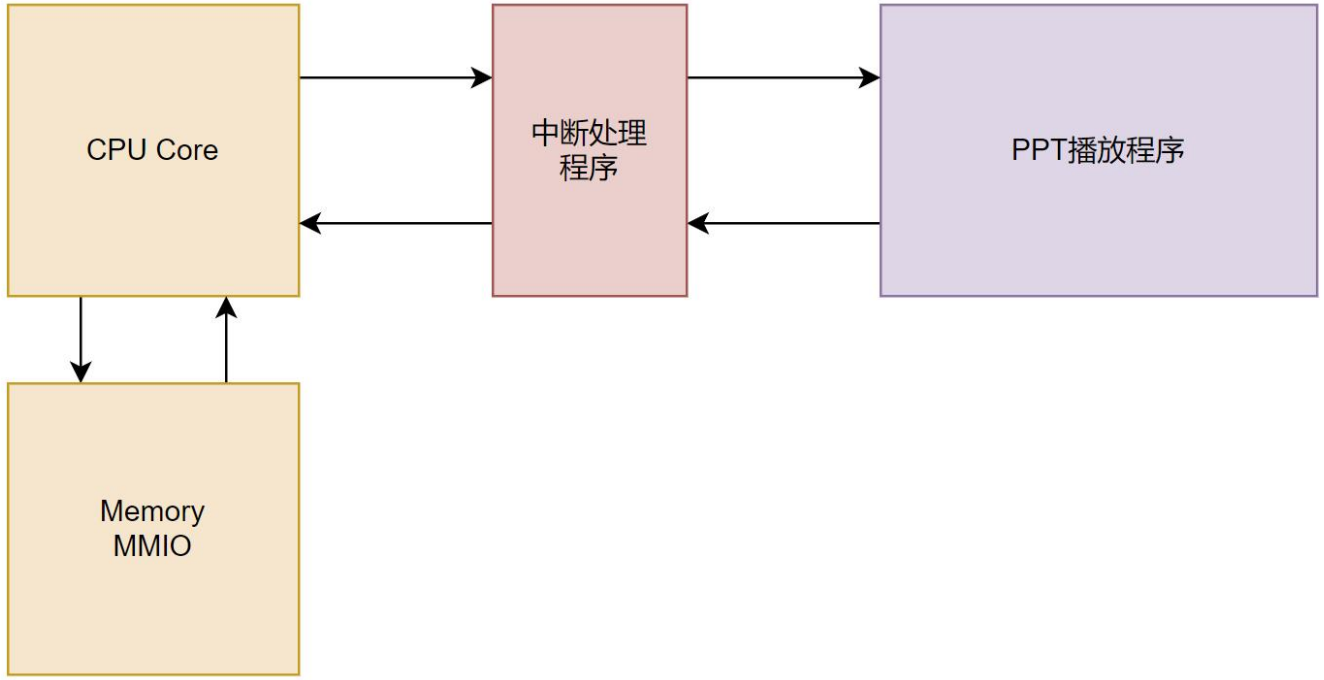
鼠标/触控板中断感知

- 在按下鼠标或触控板的那一刻，其内部会通过压敏电阻等设备产生模拟电信号
- 模拟电信号通过模数转换装置，生成对应的数字信号，并通过数据线传递给片上系统
- 数字信号会经过片上系统的中断控制器，通过CPU核的硬中断线，向CPU传递中断信息
- 与此同时，鼠标或触控板在内存中的映射地址，会记录点按的按键、坐标、力度等信息



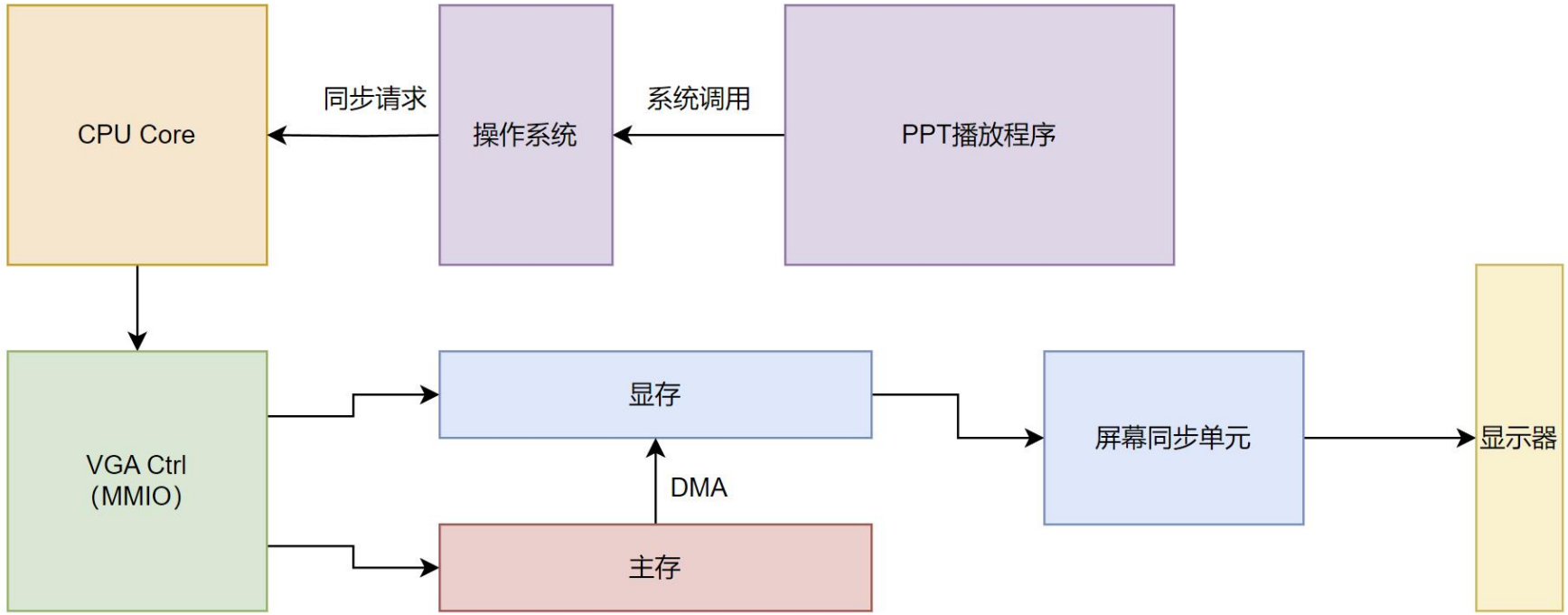
处理器 & 应用程序响应

- 处理器捕捉到中断信号，进入核心态，执行中断处理程序
- 中断处理程序会通过内存地址映射，访问点按的信息，将其反馈给PPT播放程序
- 程序返回用户态，PPT播放程序识别出这是“点击下一页”，从硬盘中指定一个新PPT



屏幕刷新与显示

- PPT播放程序通过系统调用，告知操作系统对显示器的访问请求，并进入内核态
- 操作系统将请求发送给VGA控制模块，将主存中存储的PPT画面同步到显存中
- 屏幕同步单元将画面同步到显示器上，同时程序返回用户态，回到PPT播放程序



总结

异常与特权是处理器适配操作系统等应用功能的重要手段，而它的实现需要软硬件的配合：

- 硬件主要负责保存异常现场，并在发生异常时切换模式
- 软件主要负责根据异常现场的信息，执行对应的处理程序，并在执行结束后返回用户程序和用户态

感谢！ Thank you!

汇报人

2024-6-7