

# ARM异常处理

→ (因未见过)

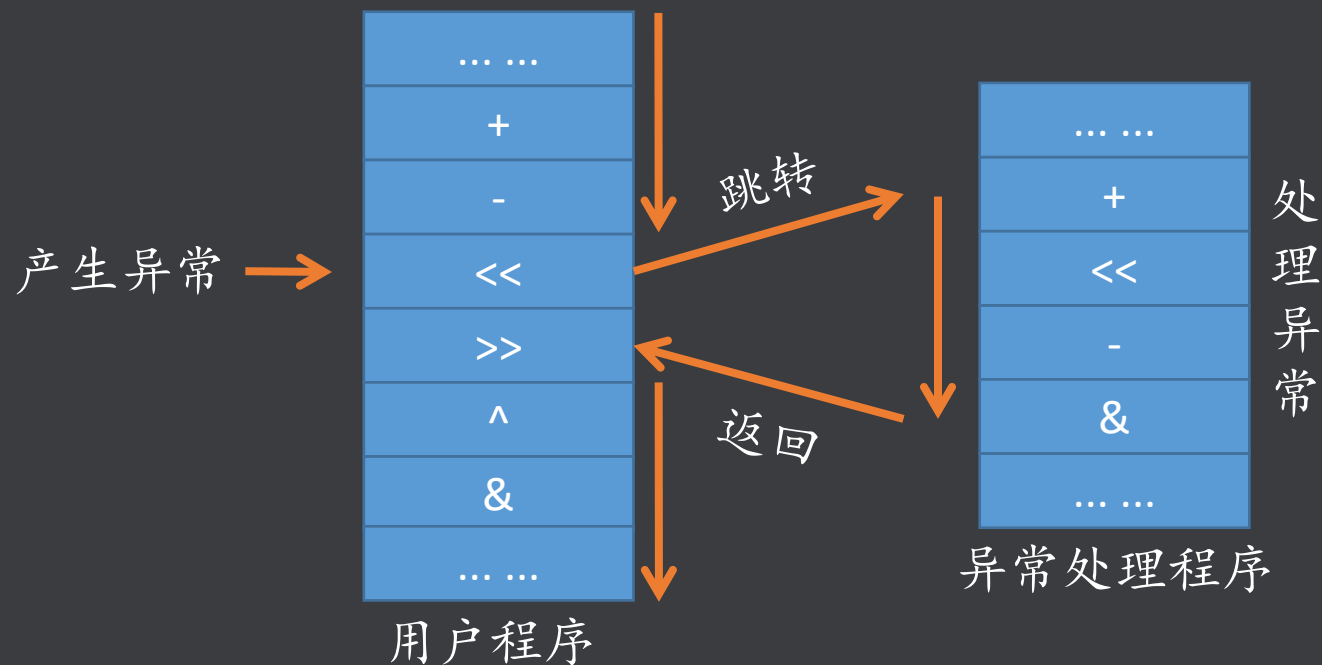
笔记处理)

创客学院 武老师

# 异常

## ■ 概念

处理器在正常执行程序的过程中可能会遇到一些不正常的事件发生  
这时处理器就要将当前的程序暂停下来转而去处理这个异常的事件  
异常事件处理完成之后再返回到被异常打断的点继续执行程序



# 异常处理机制

## ■ 异常处理机制

不同的处理器对异常的处理的流程大体相似，但是不同的处理器在具体实现的机制上有所不同；比如处理器遇到哪些事件认为是异常事件、遇到异常事件之后处理器有哪些动作、处理器如何跳转到异常处理程序、如何处理异常、处理完异常之后又如何返回到被打断的程序继续执行等。我们将这些细节的实现称为处理器的异常处理机制。

# ARM异常源

## ■ 概念

导致异常产生的事件称为异常源

## ■ ARM异常源

FIQ	快速中断请求引脚有效
IRQ	外部中断请求引脚有效
Reset	复位电平有效
Software Interrupt	执行swi指令
Data Abort	数据终止
Prefetch Abort	指令预取终止
Undefined Instruction	遇到不能处理的指令

# ARM异常模式

## ■ 异常模式

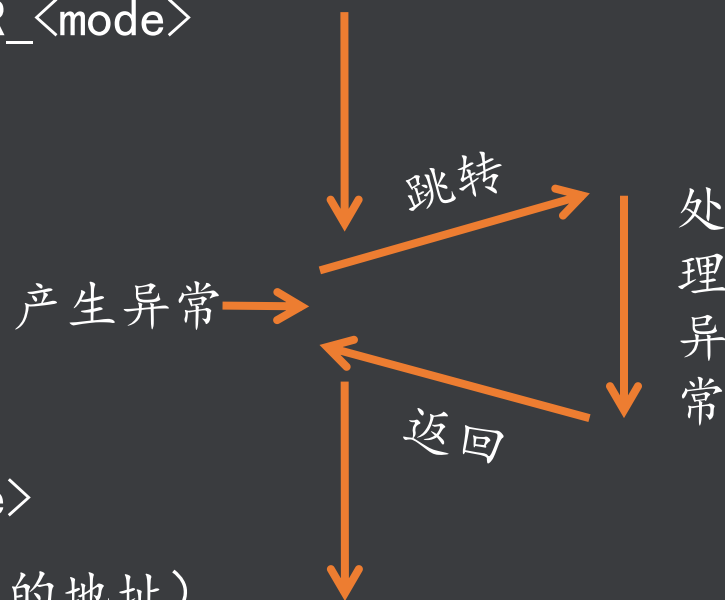
在ARM的基本工作模式中有5个属于异常模式，即ARM遇到异常后会切换到对应的异常模式

异常源	FIQ	IRQ	Reset SWI	Data Abort Prefetch Abort	Undef Instruction
异常模式	FIQ	IRQ	SVC	Abort	Undef

# ARM异常响应

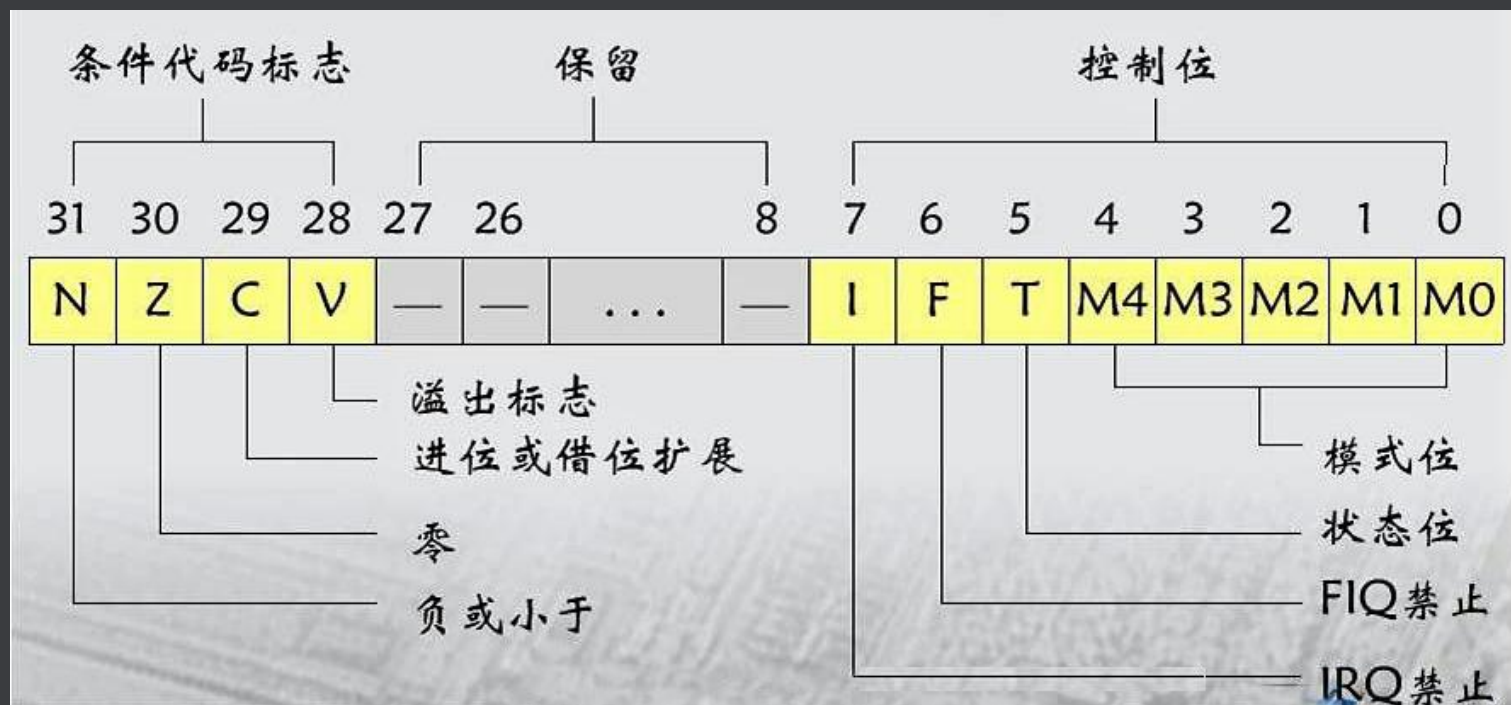
## ■ ARM产生异常后的动作（自动完成）

1. 拷贝CPSR中的内容到对应异常模式下的SPSR\_<mode>
2. 修改CPSR的值
  - 2.1. 修改中断禁止位禁止相应的中断
  - 2.2. 修改模式位进入相应的异常模式
  - 2.3. 修改状态位进入ARM状态
3. 保存返回地址到对应异常模式下的LR\_<mode>
4. 设置PC为相应的异常向量（异常向量表对应的地址）




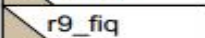



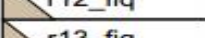
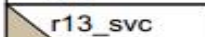
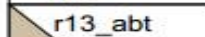

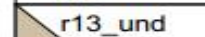

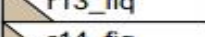
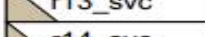
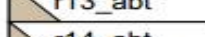
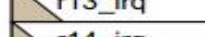
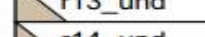

# CPSR寄存器

- CPSR (Current Program Status Register), 当前程序状态寄存器









# ARM寄存器

ARM state general registers and program counter

System and User	FIQ	Supervisor	Abort	IRQ	Undefined	Secure monitor
r0	r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7	r7
r8	 r8_fiq	r8	r8	r8	r8	r8
r9	 r9_fiq	r9	r9	r9	r9	r9
r10	 r10_fiq	r10	r10	r10	r10	r10
r11	 r11_fiq	r11	r11	r11	r11	r11
r12	 r12_fiq	r12	r12	r12	r12	r12
r13	 r13_fiq	 r13_svc	 r13_abt	 r13_irq	 r13_und	 r13_mon
r14	 r14_fiq	 r14_svc	 r14_abt	 r14_irq	 r14_und	 r14_mon
r15	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

ARM state program status registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	 SPSR_fiq	 SPSR_svc	 SPSR_abt	 SPSR_irq	 SPSR_und	 SPSR_mon

 = banked register

## ■ 注

在某个特定模式下只能使用当前模式下的寄存器，一个模式下特有的寄存器其他模式下不可使用



# 异常向量表

## ■ 异常向量表

- 异常向量表的本质是内存中的一段代码
- 表中为每个异常源分配了四个字节的存储空间
- 遇到异常后处理器自动将PC修改为对应的地址
- 因为异常向量表空间有限一般我们不会再这里写异常处理程序，而是在对应的位置写一条跳转指令使其跳转到指定的异常处理程序的入口

注：ARM的异常向量表的基地址默认在0x00地址

但可以通过配置协处理器来修改其地址

偏移地址

...	...
0x1C	FIQ
0x18	IRQ
0x14	Reserved
0x10	DataAbort
0x0C	PrefetchAbort
0x08	SWI
0x04	Undef
0x00	Reset

异常向量表

# 异常返回

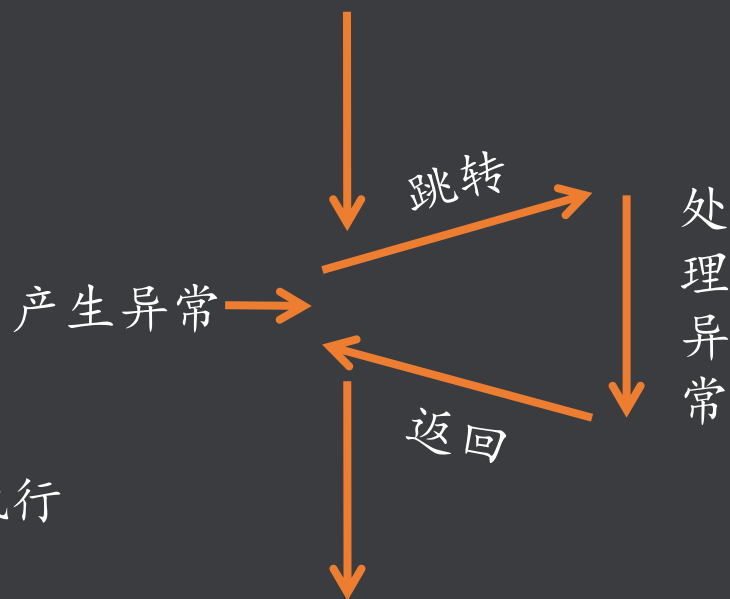
## ■ ARM异常返回的动作（自己编写）

1. 将SPSR\_<mode>的值复制给CPSR

使处理器恢复之前的状态

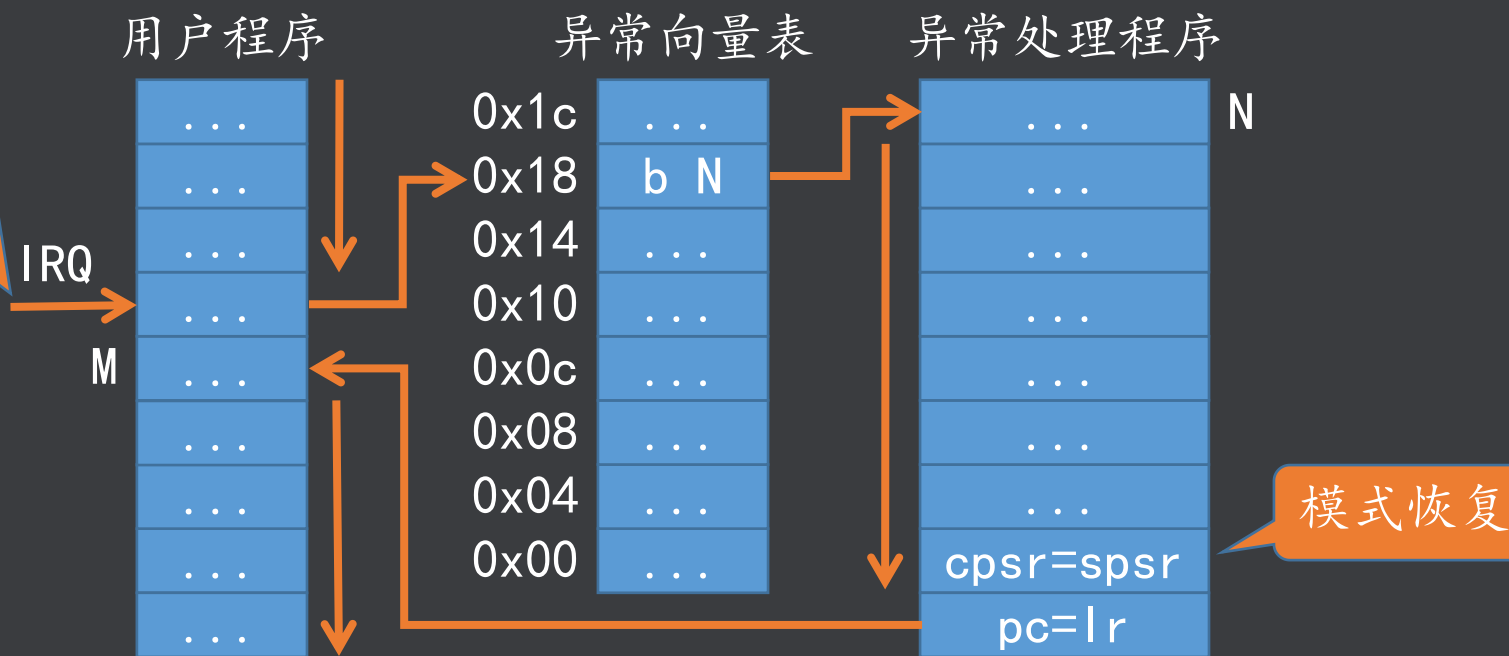
2. 将LR\_<mode>的值复制给PC

使程序跳转回被打断的地址继续执行



# IRQ异常举例

1. `spsr_<irq> = cpsr`  
2. `cpsr =`  
    2.1. irq模式  
    2.2. irq禁止  
    2.3. arm状态  
3. `lr_irq = M`  
4. `pc = 0x18`



注：整个过程CPSR保存的永远是当前程序运行状态

SPSR只是异常时对原来的CPSR进行备份

# 异常优先级

## ■ 多个异常同时产生时的服务顺序

Reset

Data Abort

FIQ

IRQ

Prefetch Abort

Software Interrupt

Undefined instruction



# FIQ和IRQ

## ■ FIQ的响应速度比IRQ快

### 1. FIQ在异常向量表位于最末

可直接把异常处理写在异常向量表之后，省去跳转

### 2. FIQ模式有5个私有寄存器(R8-R12)

执行中断处理程序前无需压栈保存寄存器，可直接处理中断

### 3. FIQ的优先级高于IRQ

#### 3.1 两个中断同时发生时先响应FIQ

#### 3.2 FIQ可以打断RIQ，但RIQ不能打断FIQ

扫一扫，获取更多信息



THANK YOU