

---


















# 中断处理框架搭建

---







创客学院 武老师


# 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

# LR寄存器

## ■ R14(LR, Link Register)

链接寄存器，一般有以下两种用途：

- 执行跳转指令(BL/BLX)时，LR会自动保存跳转指令下一条指令的地址  
程序需要返回时将LR的值复制到PC即可实现
- 产生异常时，对应异常模式下的LR会自动保存被异常打断的指令的下一条指令的地址，异常处理结束后将LR的值复制到PC可实现程序返回

## ■ 原理

当执行跳转指令或产生异常时，LR寄存器中不会凭空产生一个返回地址  
其原理是当执行跳转指令或产生异常时，处理器内部会将PC寄存器中的值拷贝到LR寄存器中，然后再将LR寄存器中的值自减4

# BL

## ■ BL

当执行BL指令时，指令执行过程中处理器内部就会将PC寄存器的值拷贝到LR寄存器，然后再将LR寄存器中的值自减4，所以LR寄存器中保存的就是BL指令下一条指令的地址

## ■ 该时刻 $PC=N+8$ $LR=N+4$

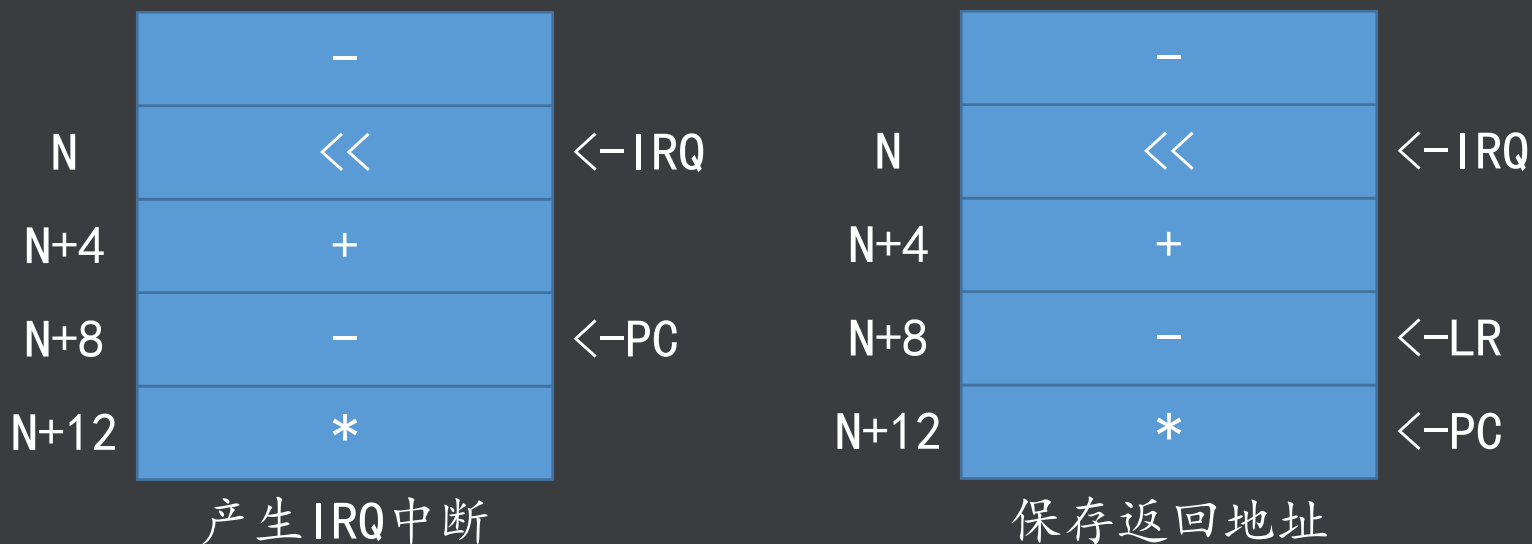
	—	
N	BL ADDR	←-执行
N+4	+	←-译码 (LR)
N+8	—	←-取指 (PC)
N+12	*	

# IRQ 中断

## ■ IRQ 中断

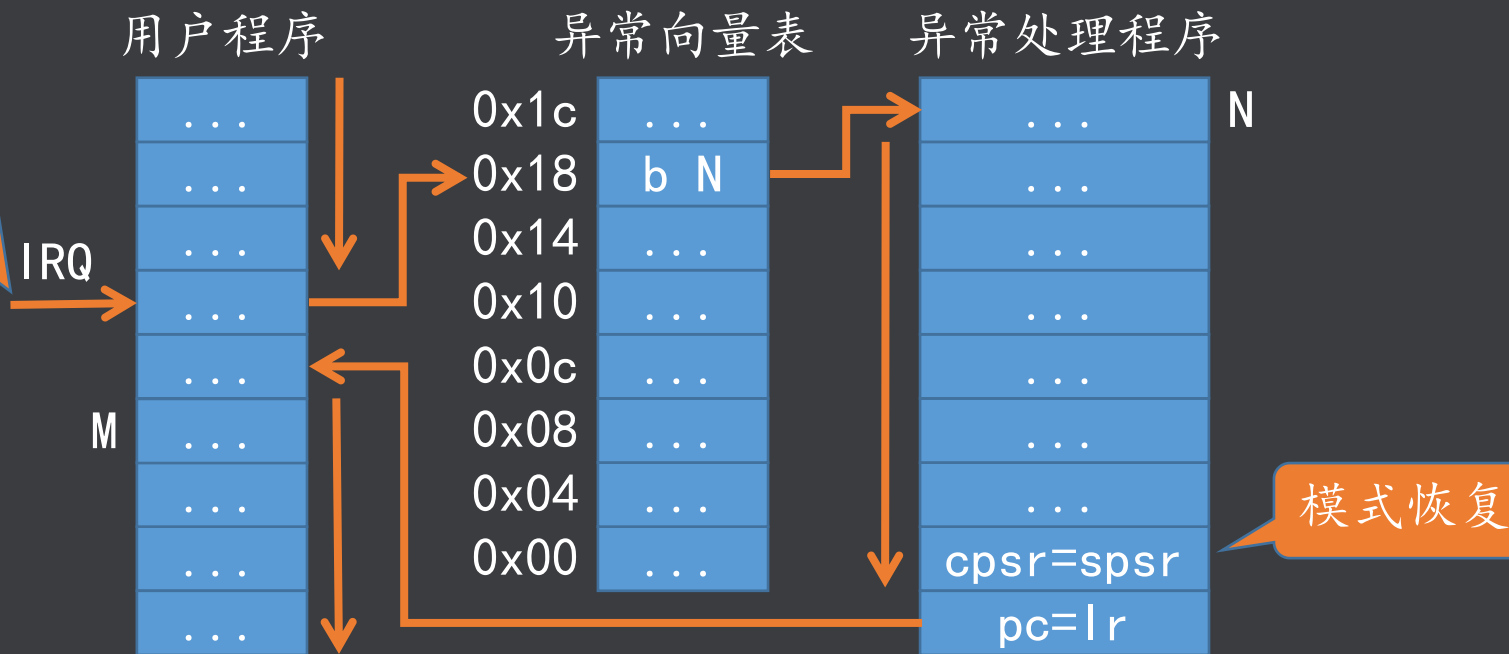
当执行一条指令时产生了一个IRQ中断，执行这条指令过程中处理器不会保存返回地址，而是执行完成后才会保存，但执行完成后PC的值又会自动增4，所以对于IRQ来说LR中保存的是被中断打断的指令的下一条指令的地址

## ■ 该时刻 $PC=N+12$ $LR=N+8$



# 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`



扫一扫，获取更多信息



THANK YOU