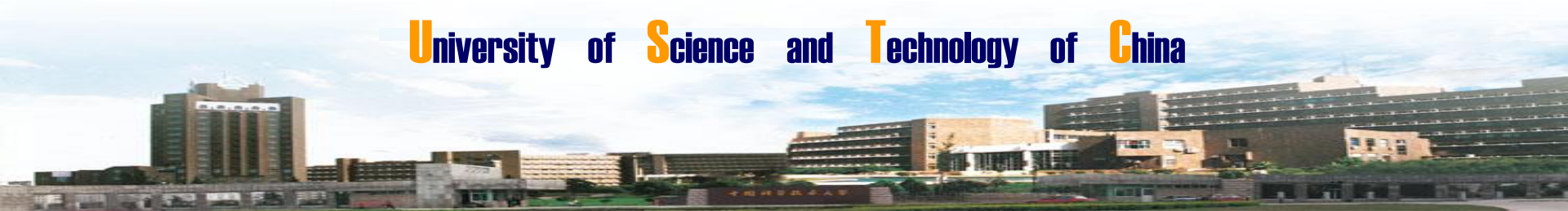
嵌入式系统安全与设计

庄连生

Email: { *lszhuang@ustc.edu.cn* }

***Fall 2018*，USTC**



第3讲 ARM9体系结构

内容提要：

- ARM简介
- ARM微处理器系列
- ARM9工作模式
- ARM9寄存器
- ARM9存储组织结构
- ARM9异常模式

第2讲 ARM处理器体系结构

内容提要：

- ARM简介
- ARM微处理器系列
- ARM9工作模式
- ARM9寄存器
- ARM9存储组织结构
- ARM9异常模式

ARM简介

ARM公司简介

ARM（Advanced RISC Machines）既可以认为是一个公司的名字，也可以认为是对一种微处理器体系结构的通称。1991年ARM公司成立于英国剑桥，设计了大量高性能、廉价、耗能低的RISC（精简指令集）处理器。

公司的特点是只设计芯片，而不生产。它将技术授权给世界上许多著名的半导体、软件和OEM厂商，并提供服务。

公司也提供基于ARM架构的开发设计技术：软件工具，评估板，调试工具，应用软件；总线架构，外围设备单元等。

ARM简介

ARM公司简介

ARM

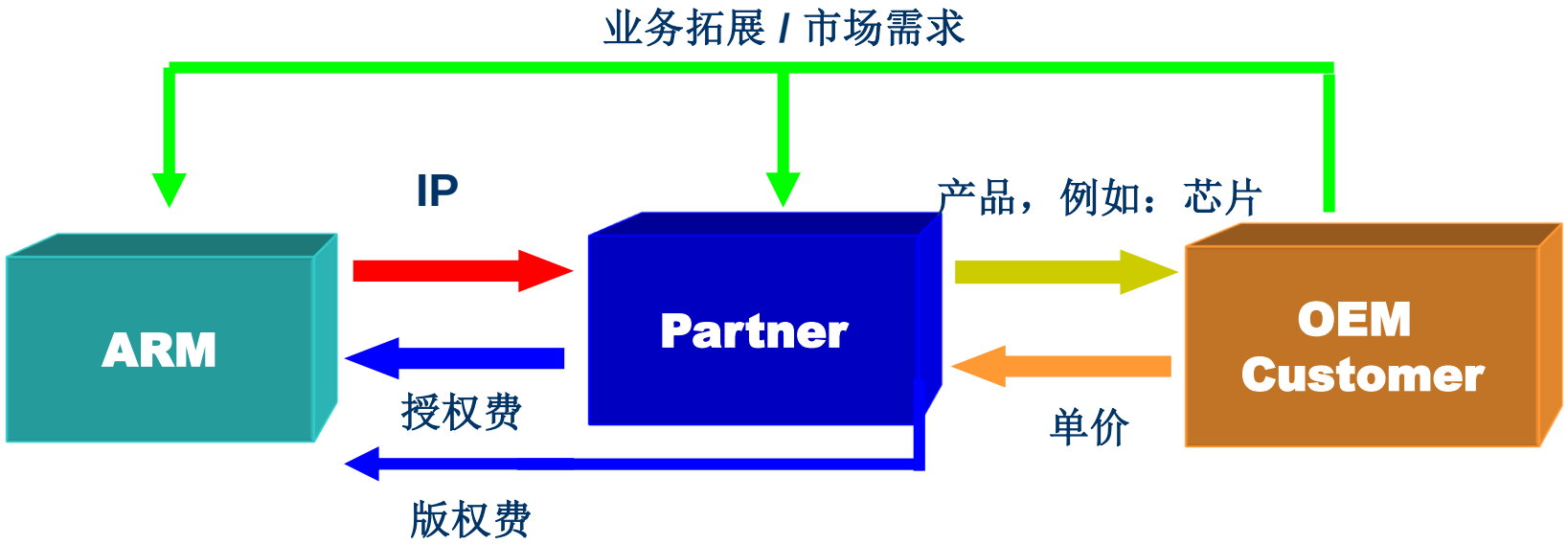
将技术授权给其它芯片厂商

PHILIPS　intel　SAMSUNG　SHARP　ATMEL　…

形成各具特色的ARM芯片

ARM简介

ARM公司商业模式



ARM 创造和设计IP

Partner把ARM IP 和其他 IP 集成进产品

OEM 用来自ARM Partner的芯片设计制造最终用户产品

ARM简介

ARM处理器的应用

- ✓当前主要应用于消费类电子领域；
- ✓到目前为止，基于ARM技术的微处理器应用约占据了32位嵌入式微处理器75%以上的市场份额
- ✓全球80%的GSM/3G手机、99%的CDMA手机以及绝大多数PDA产品均采用ARM体系的嵌入式处理器，
- ✓“掌上计算”相关的所有领域皆为其所主宰。
- ✓ARM技术正在逐步渗入到我们生活的各个方面。

ARM简介

- `所谓“体系结构”，也可以称为“系统结构”，是指程序员在为特定处理器编制程序时所“看到”从而可以在程序中使用的资源及其相互间的关系。体系结构最为重要的就是处理器所提供的指令系统和寄存器组。寄存器组与采用的指令系统是密切相关的，从这一点上考虑，体系结构中最为重要的应该就是指令系统了。

ARM简介

ARM体系结构

ARM处理器为RISC芯片，其简单的结构使ARM内核非常小，这使得器件的功耗也非常低。它具有经典RISC的特点：

- ✓ 大的、统一的寄存器文件；
- ✓ 装载/保存结构，数据处理操作只针对寄存器的内容，而不直接对存储器进行操作；
- ✓ 简单的寻址模式；
- ✓ 统一和固定长度的指令域，简化了指令的译码，便于指令流水线设计。

ARM简介

ARM体系结构的特点：

- ✓ 每条数据处理指令都对算术逻辑单元和移位器控制，实现了ALU和移位器的最大利用；
- ✓ 地址自动增加和减少寻址模式，优化程序循环；
- ✓ 多寄存器装载和存储指令实现最大数据吞吐量；
- ✓ 所有指令的条件执行实现最快速的代码执行。

ARM简介

各ARM体系结构版本

ARM体系结构从最初开发到现在有了很大的改进，并仍在完善和发展。为了清楚的表达每个ARM应用实例所使用的指令集，ARM公司定义了8种主要的ARM指令集体系结构版本，以版本号V1～V8表示。

ARM体系结构版本——V1

该版本ARM体系结构，只有26位的寻址空间，没有商业化，其特点为：

- ✓ 基本的数据处理指令（不包括乘法）；
- ✓ 字节、字和半字加载/存储指令；
- ✓ 具有分支指令，包括在子程序调用中使用的分支和链接指令；
- ✓ 在操作系统调用中使用的软件中断指令。

ARM简介

ARM体系结构版本——V2

同样为26位寻址空间，现在已经废弃不再使用，它相对V1版本有以下改进：

- ✓ 具有乘法和乘加指令；
- ✓ 支持协处理器；
- ✓ 快速中断模式中的两个以上的分组寄存器；
- ✓ 具有原子性加载/存储指令SWP和SWPB。

ARM简介

ARM体系结构版本——V3

寻址范围扩展到32位（目前已废弃），与以前的版本兼容：

- 具有乘法和乘加指令；
- 支持协处理器；
- 快速中断模式中具有的两个以上的分组寄存器；
- 具有原子性加载/存储指令SWP和SWPB。

ARM简介

- **ARM体系结构版本——V4**

不在为了与以前的版本兼容而支持26位体系结构，并明确了哪些指令会引起未定义指令异常发生，它相对V3版本作了以下改进：

- ✓ 半字加载/存储指令；
- ✓ 字节和半字的加载和符号扩展指令；
- ✓ 具有可以转换到Thumb状态的指令（BX）；
- ✓ 增加了用户模式寄存器的新的特权处理器模式。

ARM简介

ARM体系结构版本——V5

在V4版本的基础上，对现在指令的定义进行了必要的修正，对V4版本的体系结构进行了扩展并并增加了指令，具体如下：

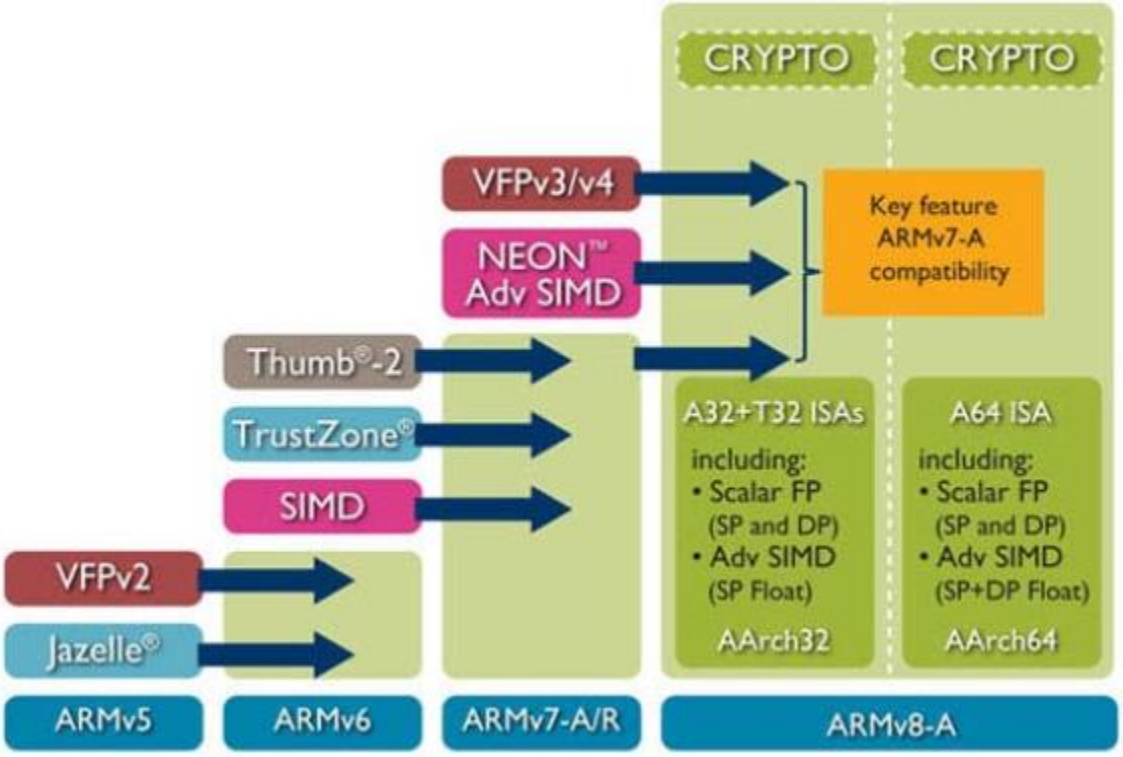
- ✓ 改进了ARM/Thumb状态之间的切换效率；
- ✓ E---增强型DSP指令集,包括全部算法操作和16位乘法操作；
- ✓ J----支持新的JAVA,提供字节代码执行的硬件和优化软件加速功能。

ARM简介

体系结构	内核名称
ARM V7	Cortex系列： Cortex-M3 / Cortex-R / Cortex A8/Cortex A9/Cortex A15 Scorpion：高通获得ARM授权后，在Cortex A8基础上设计的.
ARM V4T	ARM7 TDMI
ARM V5TE	ARM10 TDMI / ARM 10E / XScale / ARM 9 ARM9系列包含：ARM920T/ARM922T/ARM940T ARM10E系列包含：ARM 1020E / ARM 1022E / ARM 1026EJ-S
ARM V6	ARM 11

ARM简介

- ARM体系结构版本——V8



http://www.arm.com/zh/files/downloads/ARMv8_white_paper_v5.pdf

第2讲 ARM9体系结构

内容提要：

- ARM简介
- ARM微处理器系列
- ARM9工作模式
- ARM9寄存器
- ARM9存储组织结构
- ARM9异常模式

ARM处理器系列

- ARM处理器内核简介

ARM公司开发了很多系列的ARM处理器核，内核划分主要以数字来区分，包括ARM7, ARM9, ARM10, ARM11，ARM6核以及更早的系列已经很罕见了，在ARM11之后统一以Cortex命名。目前应用比较广泛的系列是：

ARM7	**ARM9**	**ARM9E**	**ARM10**	**ARM11**
SecurCore				
Cortex				
Xscale				

其中，后缀数字7,9,10,11表示不同的内核设计，数字的升序说明性能和复杂度的提高。

ARM处理器系列

ARM7	预取（Fetch）	译码（Decode）	执行（Execute）					
ARM9	预取（Fetch）	译码（Decode）	执行（Execute）	访存（Memory）	写入（Write）			
ARM10	预取（Fetch）	发送（Issue）	译码（Decode）	执行（Execute）	访存（Memory）	写入（Write）		
ARM11	预取（Fetch）	预取（Fetch）	发送（Issue）	译码（Decode）	转换（Snny）	执行（Execute）	访存（Memory）	写入（Write）

ARM处理器系列

项目	ARM7	ARM9	ARM10	ARM11
流水线	3	5	6	8
典型频率（MHz）	80	150	260	335
功耗（mW/MHz）	0.06	0.19（+cache）	0.5（+cache）	0.4（+cache）
版本	V4	V5	V5	V6
架构	冯·诺伊曼	哈佛	哈佛	哈佛

ARM处理器系列

ARM系列	微处理器核	特点
ARM7	ARM7TDMI：整数处理核 ARM7TDMI 处理器的可综合版本； ARM720T：带MMU的处理器核心，支持操作系统； ARM7EJ-S：带有DSP和Jazelle™ 技术，能够实现Java加速功能	● 冯·诺伊曼体系结构； ● ARMTDMI是目前应用最广的微处理器核 ● ARM720T带有MMU和8KB的指令数据混合cache； ● ARM7EJ-执行ARMv5TEJ指令，5级流水线，提供Java加速指令，没有存储器保护。
ARM9	ARM920T：带有独立的16KB 数据和指令Cache； ARM922T：带有独立的8KB 数据和指令Cache； ARM940T–包括更小数据和指令Cache和一个MPU(Memory Protection Unit)	● 基于ARM9TDMI，带16位的Thumb指令集，增强代码密度最多到35%； ● 在0.13μm工艺下最高性能可达到300MIPS（Dhrystone 2.1测试标准）； ● 集成了数据和指令Cache； ● 32位AMBA总线接口的MMU支持； ● 可在0.18μm、 0.15μm和0.13μm工艺的硅芯片上实现。

ARM处理器系列

ARM9E	ARM926EJ-S：Jazelle技术，有MMU，可配置的数据和指令Cache,TCM接口； ARM946E-S：可配置的数据和指令Cache及TCM; ARM966E-S：针对要求高性能和低功耗的可预测的指令执行时间的硬实时应用设计 ARM968E-S：最小、功耗最小的ARM9E系列处理器，针对嵌入式实时应用设计；	●ARM9E是针对微控制器、DSP和Java的单处理器解决方案； ●ARM Jazelle 技术提供8倍的 Java 加速性能(ARM926EJ-S)； ●5-级整数流水线； ●在0.13μm工艺下最高性能可达到300MIPS（Dhrystone 2.1测试标准）； ●可选择的 向量浮点单元VFP9 协处理器指令优秀海浮点性能，对于3D图形加速和实时控制可达到 215MFLOPS。 ●高性能的AHB总线，带MMU ●可在0.18μm, 0.15μm, 0.13μm工艺的硅芯片上实现。
ARM10E	ARM1020E：带DSP指令集，在片调试功能，独立的32KB数据和指令Cache，MMU支持； ARM1022E：与ARM1020E相同，只是独立的数据和指令Cache变为16KB； ARM1026EJ-S：同时具有MPU和MMU，可综合版本；	●带分支预测的6级整数流水线； ●在0.13μm工艺下最高性能可达到430MIPS（Dhrystone 2.1测试标准）； ●对于3D图形运算和实时控制采用VFP协处理器，浮点运算性能最高可达650MFLOPS； ●双64位AMBA总线接口和64位内部总路线接口； ●优化的缓存结构提高了处理器访问低速存储器的性能； ●可在0.18μm, 0.15μm, 0.13μm工艺的硅芯片上实现

ARM处理器系列

ARM11	ARM11 MPCore：可综合的多处理器核，1至4个处理器可配置； ARM1136J(F)-S：可配置的数据和指令Cache，可提供1.9位的MPEG4编码加速功能； ARM1156T2(F)-S：带集成浮点协处理器，带内存保护单元MPU； ARM1176JZ(F)-S：带针对CPU和系统安全架构扩展的TrustZone技术。	●增强的Thumb、Jazelle、DSP扩展支持； ●带片上和系统安全TrustZone 技术支持； ●在0.13μm工艺下最高可达到550MHz； ●MPCore在0.13μm工艺下最高性能可达到740MIPS（Dhrystone 2.1测试标准）； ●支持多媒体指令SIMD； ●采用三种电源模式：全速/待命/休眠 ●集成DMA的TCM ●低功耗、高性能。
SecurCore	SC100：第一个32位安全处理器； SC110：在SC100上增加密钥协处理器； SC200：带Jazelle技术的高级安全处理器； SC210：在SC200上增加密钥协处理器	●SecurCore是专门为智能卡、安全IC提供的32位安全处理器， 为电子商务、银行、网络、移动多媒体、公共交通提供安全解决方案； ●体积小、功耗低，代码压缩密度高； ●为快速增长的Java卡平台提供Java加速功能；

ARM处理器系列

Cortex	Cortex-A（ARMv7A）：面向应用的微处理器，针对复杂操作系统和应用程序设计； Cortex-R：针对实时系统的嵌入式处理器； Cortex-M：针对成本敏感应用优化的深度嵌入式处理器；	●2004年发布，提供增强的媒体和数字处理能力，增加了系统性能； ●支持ARM、Thumb、Thumb-2指令集； ●Thumb-2指令集提供了更高的代码存储密度，进一步降低成本；
Intel系列	StrongARM：ARMv4体系 XScale：ARMv5TE体系，增加MMX指令	●StrongARM主要应用于手持设备和PDA，5级流水线，具有独立的数据和指令Cache，不支持Thumb指令集，目前已停产； ●XScale是目前Intel公司主推的高性能嵌入式处理器，分通用处理器、网络处理器和I/O处理器三类。其中通用处理器有PXA25x、PXA26x、PXA27x三个系列，被广泛应用于智能手机、PDA领域。

ARM处理器系列

ARM处理器内核的选择

- ✓系统的工作频率
- ✓片内存储器的容量
- ✓片内外围电路的选择
  - MMU, USB接口/以太网接口, GPIO数量, 中断控制器, IIS音频接口, nWAIT信号, RTC, LCD控制器, ADC和DAC, PS2, CAN总线,IIC接口， UART和IrDA, 时钟计数器和看门狗计数器, 电源管理功能, DMA控制器

ARM处理器系列

- ARM9系列微处理器是低功耗的32位RISC结构，最适合要求低成本、低功耗的消费类应用产品，是目前应用最广泛的、高性价比的嵌入式处理器。
- 基于ARM9处理器核产生了多种微控制器芯片，如三星公司S3C2410/2440，Atmel公司的AT91RM9200、Intel公司的PXA255等。
- 不同微处理器芯片在功能及引脚等方面有所不同，但其核心部分是相同的，其软件可相互兼容。
- 本课程选择S3C2410/2440这两款ARM作为主要教学对象：
  - ✓ 推出时间比较长，开发板比较成熟和便宜，技术资料多，有利于学习

第2讲 ARM9体系结构

内容提要：

- ARM简介
- ARM微处理器系列
- ARM9工作模式
- ARM9寄存器
- ARM9存储组织结构
- ARM9异常模式

ARM9结构特点

- ARM9系列微处理器使用ARM9TDMI处理器内核，包含了16位的Thumb指令集，在高性能和低功耗特性方面有良好的表现。

- ARM9系列微处理器包含ARM920T、ARM922T和ARM940T三种类型，以适应不同的市场需求。
  - ✓ ARM920T微处理器在ARM9TDMI通用内核的基础上增加了存储器管理单元(MMU)和数据缓存(cache)。

ARM9结构特点

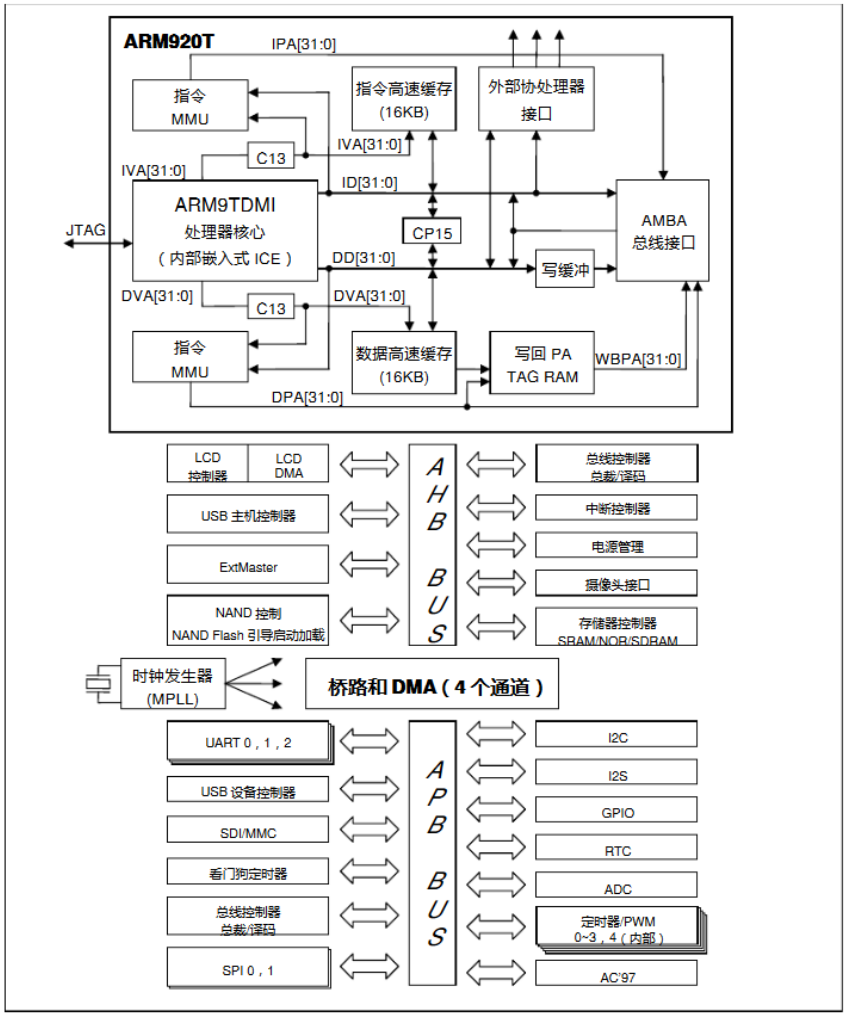


图 1-1. S3C2440A 方框图

ARM9结构特点

- ARM920T内核：
  - ✓ ARM9采用五级指令流水处理结构（取指、译码、执行、访存、写回），一条指令分配到5个时钟周期内完成；
  - ✓ 分离的cache结构（指令Cache和数据Cache）；
  - ✓ 程序计数器(PC)总是指向取指的指令而不是正在执行的指令，即正在执行的指令对应的地址总是当前程序计数器(PC)值对应地址之前2条指令的地址；
  - ✓ ARM920T核的内部具有指令缓存和数据缓存，允许处理器同时进行取指和读写数据操作。

ARM9指令集特点

- ARM920T的指令集是基于精简指令集计算机(RISC)原理的，RISC指令集计算机与复杂指令集计算机(CISC)相比较而言，译码机制更简单，从而使RISC指令集的处理器具有以下一些优点：
  - ✓ 具有较高的指令吞吐率；
  - ✓ 实时中端响应性能好；
  - ✓ 具有体积小、性价比高的处理器宏单元。
- ARM920T支持Thumb指令集，在32位体系结构上实现了16位指令集，提供比16位体系结构更高的性能和比32位体系结构更高的代码密度。

ARM指令集特点

Thumb指令集

- ✓ 32位体系结构上的16位指令集，以ARM指令集一半的存储空间，换取接近ARM指令集的性能（约有10%左右的性能损失）
- ✓ 16位指令集对应相同功能的32位指令，是ARM指令的子集，在32位的体系结构上实现16位指令集
- ✓ 实时解压缩，全32位操作
- ✓ ARM/THUMB代码快速切换(子程序调用的时间，3时钟周期)
- ✓ 使用于有存储器限制而性能要求较高的场合

■ **Thumb指令有一个优点：能够从Thumb代码状态切换回全ARM代码状态并全速执行，其时间开销仅与子程序入口相当。**

ARM9工作模式

处理器7种模式

处理器模式	说明	备注
用户 (usr)	正常程序执行模式	不能直接切换到其它模式
系统 (sys)	运行操作系统的特权任务	与用户模式类似，但具有可以直接切换到其它模式等特权
快中断 (fiq)	支持高速数据传输及通道处理	FIQ异常响应时进入此模式
中断 (irq)	用于通用中断处理	IRQ异常响应时进入此模式
管理 (svc)	操作系统保护模式	系统复位和软件中断响应时进入此模式
中止 (abt)	用于支持虚拟内存和/或存储器保护	在ARM7TDMI没有大用处
未定义 (und)	支持硬件协处理器的软件仿真	未定义指令异常响应时进入此模式

ARM9工作模式

特权模式

处理器模式	说明	备注
用户 (usr)	正常程序工作模式	不能直接切换到其它模式
系统 (sys)		
快中断 (fiq)		
中断 (irq)		
管理 (svc)		
中止 (abt)		
未定义 (und)		

除用户模式外，其它模式均为特权模式。ARM内部寄存器和一些片内资源在硬件设计上只允许（或者可选为只允许）特权模式下访问。此外，特权模式可以自由的切换处理器模式，而用户模式不能直接切换到别的模式。

ARM9工作模式

异常模式

处理器模式	说明	备注
用户 (usr)	正常程序工作模式	不能直接切换到其它模式
系统 (sys)		
快中断 (fiq)		
中断 (irq)		
管理 (svc)		
中止 (abt)		
未定义 (und)		

这五种模式称为异常模式。它们除了可以通过程序切换进入外，也可以由特定的异常进入。当特定的异常出现时，处理器进入相应的模式。每种异常模式都有一些独立的寄存器，以避免异常退出时用户模式的状态不可靠。

ARM9工作模式

- 用户和系统模式

处理器模式	说明	备注
用户 (usr)	正	
系统 (sys)	用权	
快中断 (fiq)	支道	
中断 (irq)	用	
管理 (svc)	操	
中止 (abt)	用存	
未定义 (und)	支件仿真	模式

这两种模式都不能由异常进入，而且它们使用完全相同的寄存器组。

系统模式是特权模式，不受用户模式的限制。操作系统在该模式下访问用户模式的寄存器就比较方便，而且操作系统的一些特权任务可以使用这个模式访问一些受控的资源。

ARM9工作状态

- ARM9TDMI处理器内核使用V4T版本的ARM结构，该结构包含32位ARM指令集和16位Thumb指令集。因此ARM9TDMI处理器有两种操作状态：
  - ✓ ARM状态：32位，这种状态下执行的是字方式的ARM指令（地址[1:0]为0）；
  - ✓ Thumb状态：16位，这种状态下执行半字方式的ARM指令（地址[0]为0）。
  - ✓ 如果处理器在Thumb状态进入异常(自动切换到ARM状态)，则当异常处理返回时，自动切换到Thumb状态。
- ARM指令集和Thumb指令集中均有切换处理器核工作状态的指令，使处理器核可在两种工作状态之间切换。
- 两个状态之间可以随时切换，并不影响工作模式和寄存器的内容；
- 微处理器在上电或复位而开始执行代码时，应该处于ARM状态。

ARM9工作状态

- 使用BX指令将ARM9TDMI内核的操作状态在ARM状态和Thumb状态之间进行切换，程序如下所示。

```
;从Arm状态切换到Thumb状态
      LDR     R0,=Lable+1
      BX      R0
;从Thumb状态切换到ARM状态
      LDR     R0,=Lable
      BX      R0
      ……
Label MOV     R1,#6
```

跳转地址标号

地址最低位为1，表示切换到Thumb状态

地址最低位为0，表示切换到ARM状态

Label标号地址

* BX 带状态切换的跳转指令；

ARM状态各模式下的寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
	R6(v3)	R6						
	R7(v4)	R7						
	R8(v5)	R8						R8_fiq *
	R9(SB,v6)	R9						R9_fiq *
	R10(SL,v7)	R10						R10_fiq *
	R11(FP,v8)	R11						R11_fiq *
	R12(IP)	R12						R12_fiq *
	R13(SP)	R13		R13_svc*	R13_abt *	R13_und *	R13_irq *	R13_fiq *
	R14(LR)	R14		R14_svc *	R14_abt *	R14_und *	R14_irq *	R14_fiq *
	R15(PC)	R15						
状态寄存器	R16(CPSR)	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

ARM状态各模式下的寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
		R6						
		R7						
		R8						R8_fiq
		R9						R9_fiq
		R10						R10_fiq
		R11						R11_fiq
	R12(IP)	R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

所有的37个寄存器，分成两大类：

- 31个通用32位寄存器；
- 6个状态寄存器。

ARM状态各模式下可以访问的寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
	R6(v3)	R6						
	R7(v4)	R7						
	R8(v5)	R8						R8_fiq
	R9(SB,v6)	R9						R9_fiq
	R10(SL,v7)	R10						R10_fiq
	R11(FP,v8)	R11						R11_fiq
	R12(IP)	R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

一般的通用寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用程序寄存器	R0(a1)	R0						
	R1(a2)	R1						
		R2						
		R3						
		R4						
		R5						
		R6						
		R7						
		R8						R8_fiq
		R9						R9_fiq
		R10						R10_fiq
		R11						R11_fiq
	R12(IP)	R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

在汇编语言中寄存器R0～R15为保存数据或地址值的**通用寄存器**。它们是完全通用的寄存器，不会被体系结构作为特殊用途，并且可用于任何使用通用寄存器的指令。

一般的通用寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器	R0	R0						
	R1	R1						
	R2	R2						
	R3	R3						
	R4	R4						
	R5	R5						
	R6	R6						
	R7	R7						
	R8(v5)	R8						R8_fiq
	R9(SB,v6)	R9						R9_fiq
	R10(SL,v7)	R10						R10_fiq
	R11(FP,v8)	R11						R11_fiq
	R12(IP)	R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

其中R0～R7为**未分组的寄存器**，也就是说对于任何处理器模式，这些寄存器都对应于相同的32位物理寄存器。

一般的通用寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
		R5						
		R6						
		R7						
		R8						R8_fiq
		R9						R9_fiq
		R10						R10_fiq
		R11						R11_fiq
		R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器R8～R14为**分组寄存器**。它们所对应的物理寄存器取决于当前的处理器模式，几乎所有允许使用通用寄存器的指令都允许使用分组寄存器

一般的通用寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)							
	R2(a3)							
	R3(a4)							
	R4(v1)							
	R5(v2)							
	R6(v3)							
	R7(v4)							
	R8(v5)	**R8**						**R8_fiq**
	R9(SB,v6)	**R9**						**R9_fiq**
	R10(SL,v7)	**R10**						**R10_fiq**
	R11(FP,v8)	**R11**						**R11_fiq**
	R12(IP)	**R12**						**R12_fiq**
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器R8～R12有两个分组的物理寄存器。一个用于除FIQ模式之外的所有寄存器模式，另一个用于FIQ模式。这样在发生FIQ中断后，可以加速FIQ的处理速度。

一般的通用寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
	R6(v3)	R6						
	R7(v4)							
	R8(v5)							R8_fiq
	R9(SB,v6)							R9_fiq
	R10(SL,v7)							R10_fiq
	R11(FP,v8)							R11_fiq
	R12(IP)				R12			R12_fiq
	R13(SP)	**R13**		**R13_svc**	**R13_abt**	**R13_und**	**R13_irq**	**R13_fiq**
	R14(LR)	**R14**		**R14_svc**	**R14_abt**	**R14_und**	**R14_irq**	**R14_fiq**
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器R13、R14分别有6个分组的物理寄存器。一个用于用户和系统模式，其余5个分别用于5种异常模式。

堆栈指针寄存器R13（SP）

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)							
	R6(v3)							
	R7(v4)							
	R8(v5)							R8_fiq
	R9(SB,v6)							R9_fiq
	R10(SL,v7)							R10_fiq
	R11(FP,v8)							R11_fiq
	R12(IP)							R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器R13常作为**堆栈指针**（SP）。在ARM指令集当中，没有以特殊方式使用R13的指令或其它功能，只是习惯上都这样使用。但是在Thumb指令集中存在使用R13的指令。

链接寄存器R14（LR）

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)							
	R4(v1)							
	R5(v2)							
	R6(v3)							
	R7(v4)							
	R8(v5)							
	R9(SB,v6)							
	R10(SL,v7)							
	R11(FP,v8)							
	R12(IP)							
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

R14为**链接寄存器**（LR），在结构上有两个特殊功能：

- 在每种模式下，模式自身的R14版本用于保存子程序返回地址；
- 当发生异常时，将R14对应的异常模式版本设置为异常返回地址（有些异常有一个小的固定偏移量）。

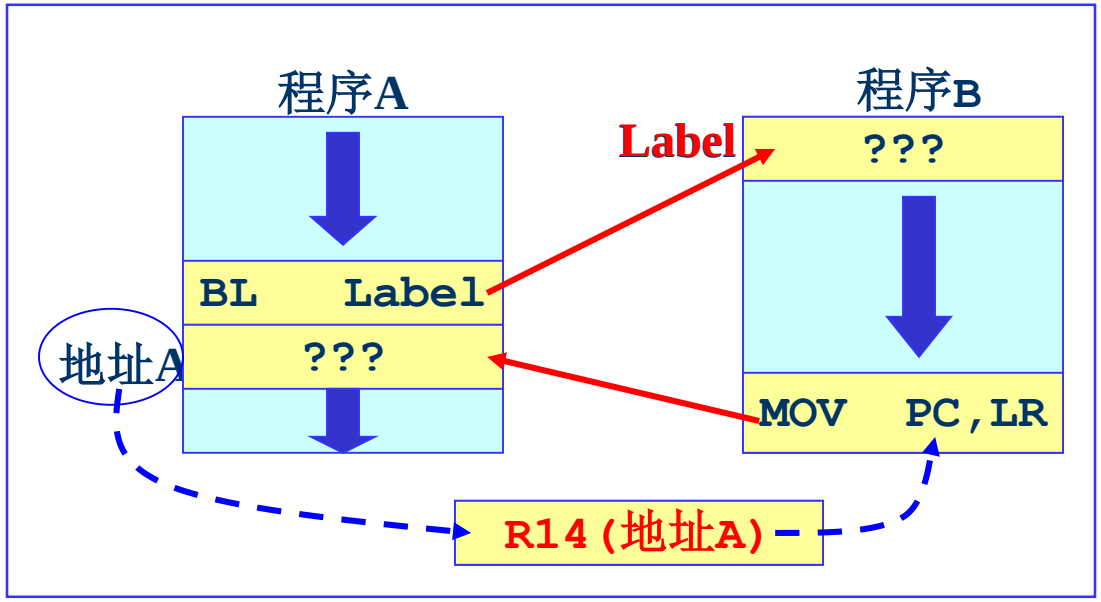
3.7 内部寄存器

R14（LR）寄存器与子程序调用

操作流程

1.程序A执行过程中调用程序B

2.程序跳转至标号Label，执行程序B。同时硬件将“BL Label”指令的下一条指令所在地址存入R14（LR）；



3.程序B执行最后，将R14寄存器的内容放入PC，返回程序A；

3.7 内部寄存器

- R14寄存器与异常发生

异常发生时，程序要跳转至异常服务程序，对返回地址的处理与子程序调用类似，都是由硬件完成的。区别在于有些异常有一个小常量的偏移。

3.7 内部寄存器

- R14寄存器注意要点

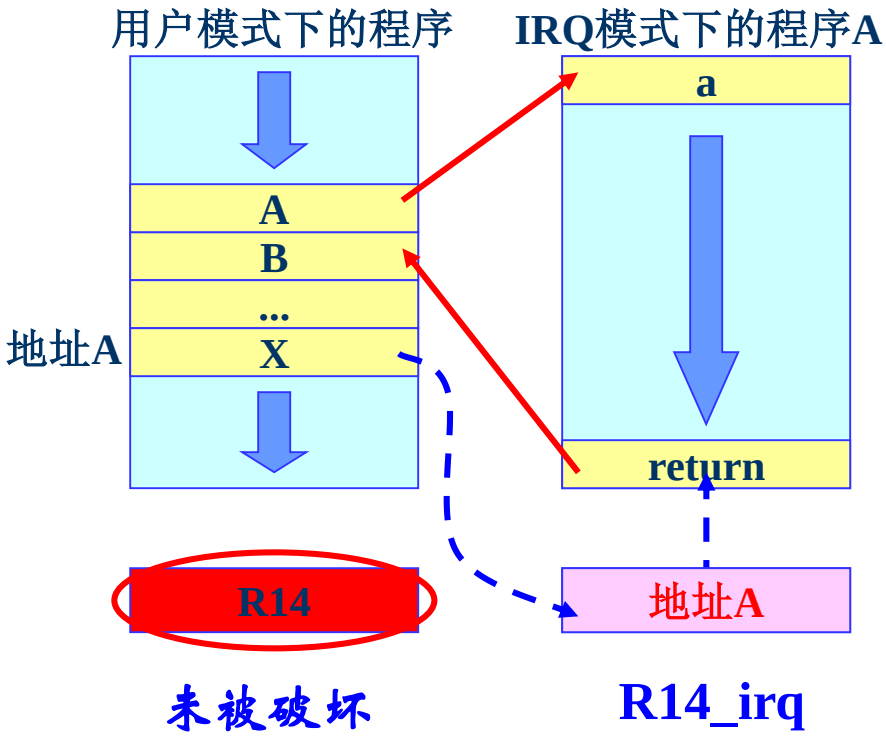
当发生异常嵌套时，这些异常之间可能会发生冲突。

例如：如果用户在用户模式下执行程序时发生了IRQ中断，用户模式寄存器不会被破坏。但是如果允许在IRQ模式下的中断处理程序重新使能IRQ中断，并且发生了嵌套的IRQ中断时，外部中断处理程序保存在R14_irq中的任何值都将被嵌套中断的返回地址所覆盖。

3.7 内部寄存器

- R14寄存器注意要点

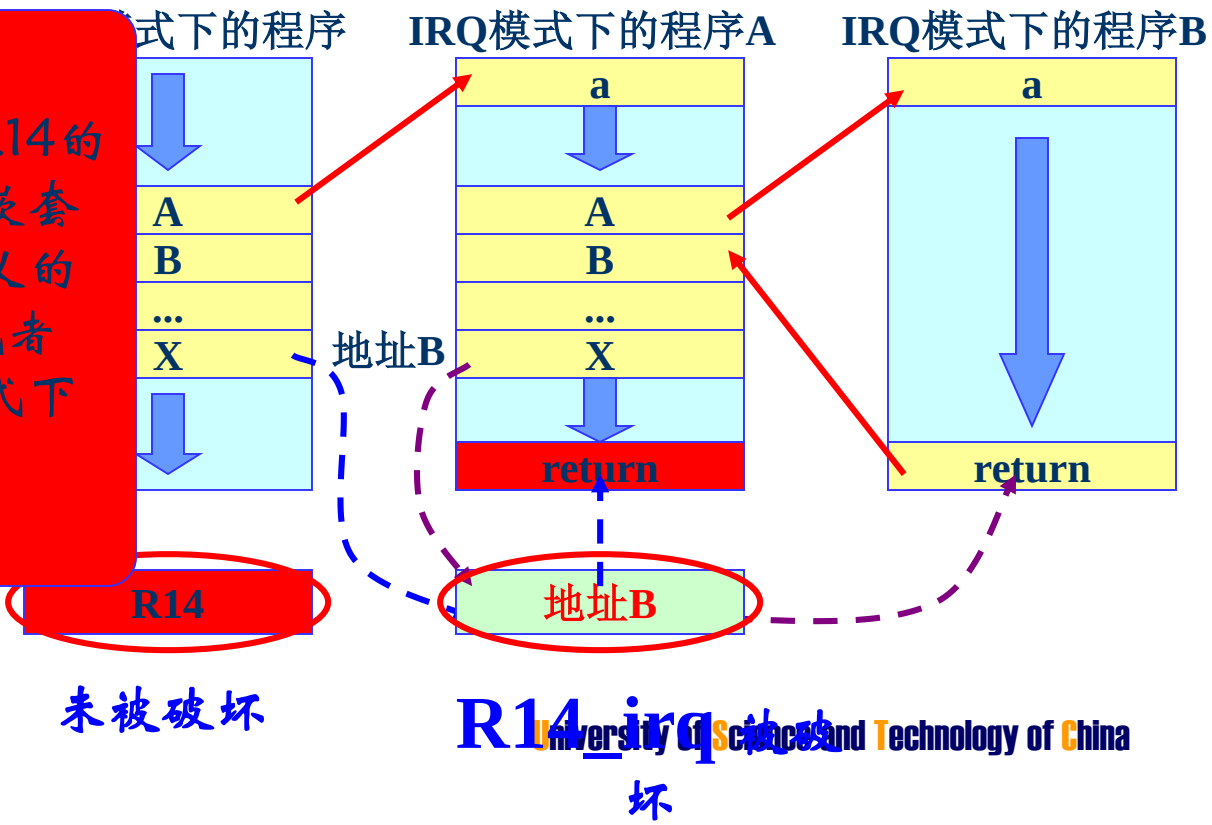
3. IRQ服务程序A执行完毕，将R14_irq寄存器的内容减去某个常量后存入PC，返回之前被中断的程序；



3.7 内部寄存器

- R14寄存器注意要点

解决办法是确保R14的对应版本在发生中断嵌套时不再保存任何有意义的值（将R14入栈），或者切换到其它处理器模式下。



程序计数器R15（PC）

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)							
	R5(v2)							
	R6(v3)							
	R7(v4)							
	R8(v5)							
	R9(SB,v6)							
	R10(SL,v7)							
	R11(FP,v8)							
	R12(IP)							
	R13(SP)							
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器R15为**程序计数器**（PC），它指向正在取指的地址。可以认为它是一个通用寄存器，但是对于它的使用有许多与指令相关的限制或特殊情况。如果R15使用的方式超出了这些限制，那么结果将是不可预测的。

3.7 内部寄存器

读R15的限制

正常操作时，从R15读取的值是处理器正在取指的地址，即当前正在执行指令的地址加上8个字节（两条ARM指令的长度）。由于ARM指令总是以字为单位，所以R15寄存器的最低两位总是为0。

地址	程序代码	流水线状态
PC-8	LDR R0,PC	正在执行
PC-4	???	正在译码
PC	???	正在取指

3.7 内部寄存器

- 读R15的限制

当使用STR或STM指令保存R15时，会有一个例外。这些指令可能将当前指令地址加8字节或加12字节保存（将来可能还有其它数字）。偏移量是8还是12取决于具体的ARM芯片，但是对于一个确定的芯片，这个值是一个常量。

所以最好避免使用STR和STM指令来保存R15，如果很难做到，那么应当在程序中计算出该芯片的偏移量。

3.7 内部寄存器

- 写R15的限制

正常操作时，写入R15的值被当作一个指令地址，程序从这个地址处继续执行（相当于执行一次无条件跳转）。

3.7 内部寄存器

- 写R15的限制

由于ARM指令以字节为边界，因此写入R15的值最低两位通常为0b00。具体的规则取决于内核结构的版本：

- 在ARM结构V3版及以下版本中，写入R15的值的最低两位被忽略，因此跳转地址由指令的实际目标地址（写入R15的值）和0xFFFFFFFC相与得到；
- 在ARM结构V4版及以上版本中，写入R15的值的最低两位为0，如果不是，结果将不可预测。

程序状态寄存器CPSR

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
	R6(v3)	R6						
	R7(v4)	R7						
	R8(v5)	R8						
	R9(SB,v6)	R9						
	R10(SL,v7)	R10						
	R11(FP,v8)	R11						
	R12(IP)	R12						
	R13(SP)	R13						
	R14(LR)	R14						
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_abt	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

寄存器CPSR为**程序状态寄存器**，在异常模式中，另外一个寄存器“程序状态保存寄存器（SPSR）”可以被访问。每种异常都有自己的SPSR，在进入异常时它保存CPSR的当前值，异常退出时可通过它恢复CPSR。详细描述参看3.8小节。

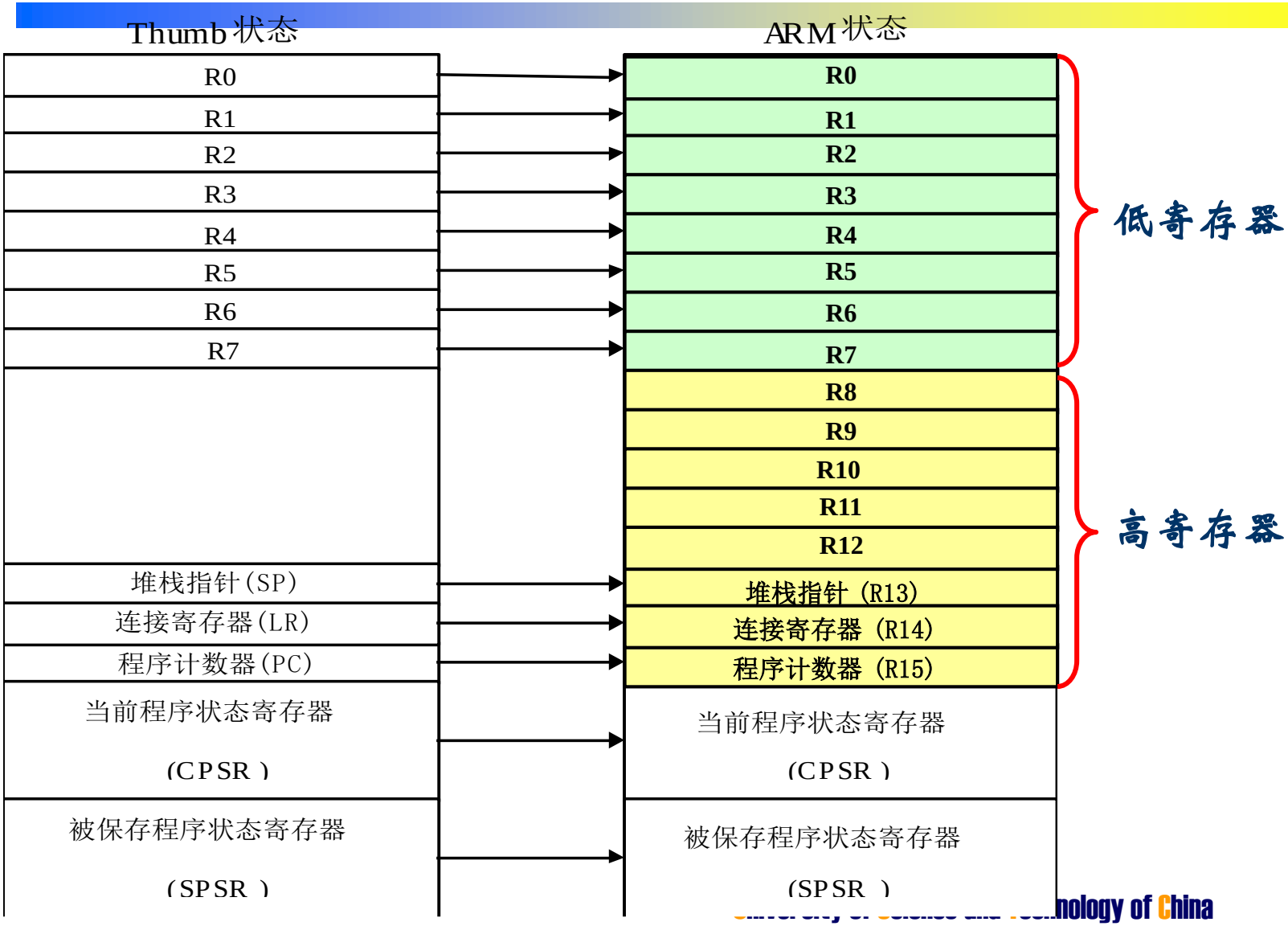
3.7 内部寄存器

- **ARM状态和Thumb状态之间寄存器的关系**

Thumb状态寄存器与ARM状态寄存器有如下的关系：

- Thumb状态R0～R7与ARM状态R0～R7相同；
- Thumb状态CPSR和SPSR与ARM状态CPSR和SPSR相同；
- Thumb状态SP映射到ARM状态R13；
- Thumb状态LR映射到ARM状态R14；
- Thumb状态PC映射到ARM状态PC（R15）。

Thumb状态寄存器在Arm状态寄存器上的映射



3.7 内部寄存器

- 在Thumb状态中访问高寄存器

在Thumb状态中，高寄存器（R8～R15）不是标准寄存器集的一部分。汇编语言程序员对它们的访问受到限制。

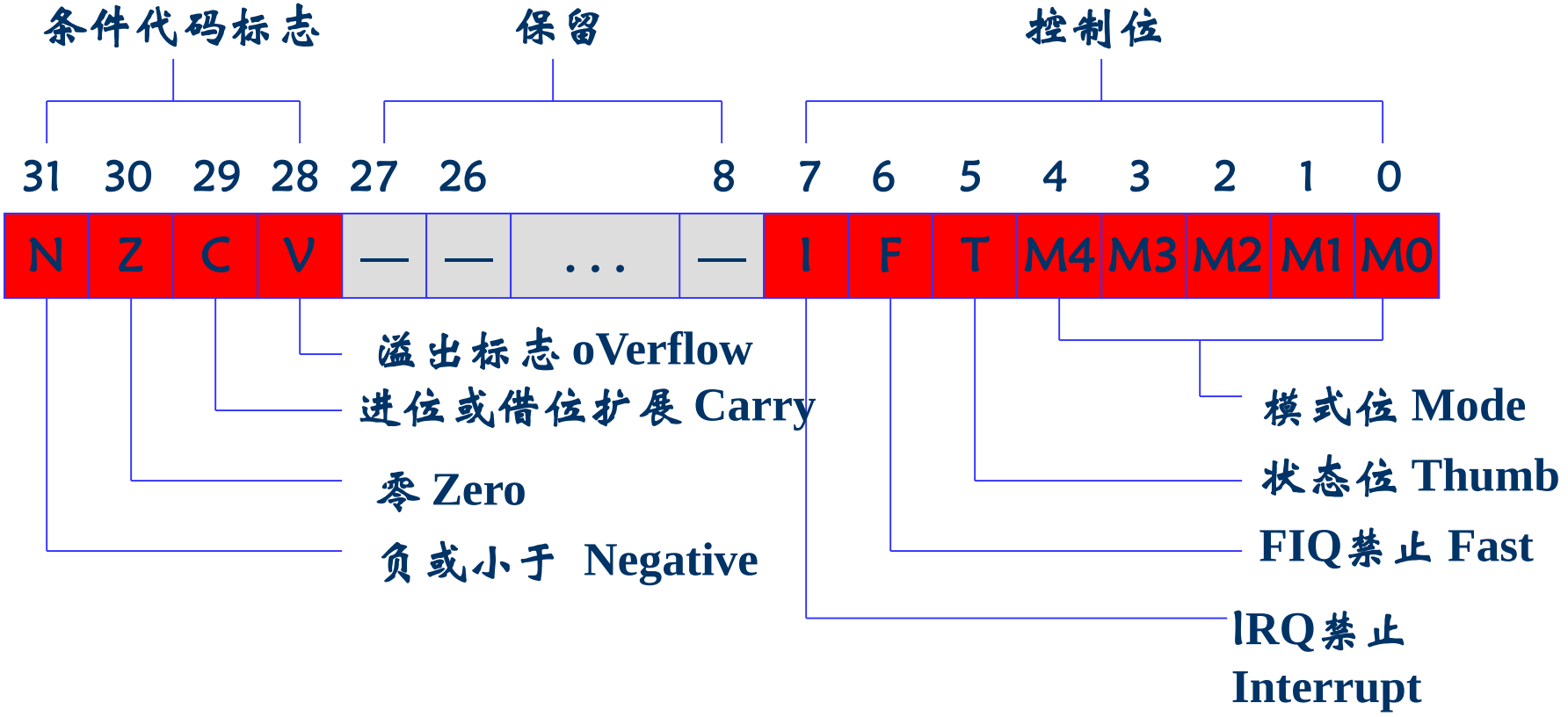
可以使用MOV、CMP和ADD指令对高寄存器操作，详见第4讲。

程序状态寄存器

- ARM9TDMI内核包含1个CPSR和5个供异常处理程序使用的SPSR。CPSR反映了当前处理器的状态，包含：
  - 4个条件代码标志（负(N)、零(Z)、进位(C)和溢出(V)）；
  - 2个中断禁止位，分别控制一种类型的中断；
  - 5个对当前处理器模式进行编码的位；
  - 1个用于指示当前执行指令(ARM还是Thumb)的位。

程序状态寄存器

CPSR寄存器的格式



程序状态寄存器

- 条件代码标志

大多数“数值处理指令”可以选择是否影响条件代码标志位。通常如果指令带S后缀，则该指令的执行会影响条件代码标志；但有一些指令的执行总是会影响条件代码标志。

N、Z、C和V位都是条件代码标志。算术操作、逻辑操作、MSR或者LDM指令可以对这些位进行设置。所有ARM指令都可按条件来执行，而Thumb指令中只有分支指令可按条件执行。

3.8 程序状态寄存器

- 条件代码标志

各标志位的含义如下：

- N 运算结果的最高位反映在该标志位。对于有符号二进制补码，结果为负数时N=1，结果为正数或零时N=0；
- Z 指令结果为0时Z=1（通常表示比较结果“相等”），否则Z=0；

3.8 程序状态寄存器

条件代码标志

各标志位的含义如下：

- C 当进行加法运算(包括CMN指令)，并且最高位产生进位时C=1，否则C=0。当进行减法运算(包括CMP 指令)，并且最高位产生借位时C=0，否则C=1。对于结合移位操作的非加法/减法指令，C为从最高位最后移出的值，其它指令C不变；
- V当进行加法/减法运算，并且发生有符号溢出时V=1，否则V=0，其它指令V通常不变。

3.8 程序状态寄存器

- 各种控制位

CPSR的最低8位为控制位，当发生异常时，这些位被硬件改变。当处理器处于一个特权模式时，可用软件操作这些位。

它们分别是：

- I和F是中断禁止位；
- T位是处理器状态位；
- M[4:0]模式位，决定处理器的工作模式

3.8 程序状态寄存器

控制位

- 中断禁止位包括I和F位：
  - 当I位置为1时，IRQ中断被禁止；
  - 当F位置为1时，FIQ中断被禁止。
- T位反映了正在操作的状态：
  - 当T位为1时，处理器运行在Thumb状态；
  - 当T位清零时，处理器运行在ARM状态。
  - CPTBIT输出信号反映当前状态（1：Thumb）
  - 不要使用MSR指令改变T位，否则处理器状态不可预测

3.8 程序状态寄存器

- 控制位

➢模式位包括M4、M3、M2、M1和M0，这些位决定处理器的操作模式。

注意：不是所有模式位的组合都定义了有效的处理器模式，如果使用了错误的设置，将引起一个无法恢复的错误。

CPSR模式位设置表

M[4:0]	模式	可见的Thumb状态寄存器	可见的ARM状态寄存器
10000	用户	R0～R7,SP,LR,PC,CPSR	R0～R14,PC, CPSR
10001	快中断	R0～R7,SP_fiq,LR_fiq,PC,CPSR, SPSR_fiq	R0～R7,R8_fiq～R14_fiq,PC, CPSR, SPSR_fiq
10010	中断	R0～R7,SP_irq,LR_irq,PC,CPSR, SPSR_fiq	R0～R12,R13_irq,R14_irq,PC, CPSR, SPSR_irq
10011	管理	R0～R7,SP_svc,LR_svc,PC,CPSR, SPSR_svc	R0～R12,R13_svc,R14_svc, PC,CPSR, SPSR_svc
10111	中止	R0～R7,SP_abt,LR_abt,PC,CPSR, SPSR_abt	R0～R12,R13_abt,R14_abt,PC, CPSR, SPSR_abt
11011	未定义	R0～R7,SP_und,LR_und,PC,CPSR, SPSR_und	R0～R12, R13_und, R14_und, PC, CPSR, SPSR_und
11111	系统	R0～R7,SP,LR,PC,CPSR	R0～R14,PC, CPSR

3.8 程序状态寄存器

- 保留位

CPSR中的保留位被保留将来使用。为了提高程序的可移植性，当改变CPSR标志和控制位时，请不要改变这些保留位。另外，请确保程序的运行不受保留位的值影响，因为将来的处理器可能会将这些位设置为1或者0。

3.9 异常

- 异常是指由内部或外部产生一个引起处理器处理的事件。也就是指，正常的程序执行流程被暂时中断而引发的过程。
- ARM920T总共定义了7种异常类型，这些异常可以由外部信号引发，也可以由软件中断指令引发。

异常名称	产生情况	进入的模式
复位	系统上电或按下复位按键	管理
未定义指令	执行未定的指令时	未定义
软件中断(SWI)	执行软件中断指令(SWI)	管理
指令预取中止	指令预取时，访问存储器失败	中止
数据中止	数据访问时，存储器访问失败	中止
IRQ(中断)	外部中断信号引起(通过nIRQ引脚)	IRQ
FIQ(快速中断)	外部中断信号引起(通过nFIQ引脚)	FIQ

3.9 异常向量

- 异常产生后，处理器的PC值将被强制赋予该异常所对应的存储器地址，处理器从此地址处开始执行程序。异常程序的入口地址通常称为**异常向量**。

异常名称	对应模式	正常向量	高地址向量
复位	管理	0x00000000	0xFFFF0000
未定义指令	未定义	0x00000004	0xFFFF0004
软件中断(SWI)	管理	0x00000008	0xFFFF0008
指令预取中止	中止	0x0000000C	0xFFFF000C
数据中止	中止	0x00000010	0xFFFF0010
IRQ(中断)	IRQ	0x00000018	0xFFFF0018
FIQ(快速中断)	FIQ	0x0000001C	0xFFFF001C

异常进入和退出

- 处理器在进入异常处理程序前，该异常模式下的R14保存断点处的PC值，SPSR保存断点处的CPSR值。处理结束之后，把SPSR的内容赋给CPSR，R14的内容赋给PC。
- 如果异常处理程序已经把返回地址拷贝到堆栈，那么可以使用一条多寄存器传送指令来恢复用户寄存器并实现返回。

中断处理代码的开始部分和退出部分

```
SUB     LR,LR,#4            ;计算返回地址
STMFD   SP!,{R0-R3,LR}      ;保存使用到的寄存器
. . .
LDMFD   SP!,{R0-R3,PC}^     ;中断返回
```

异常进入和退出

- 如果异常处理程序已经把返回地址拷贝到堆栈，那么可以使用一条多寄存器传送指令来恢复用户寄存器并实现返回。

中断处理代码的开始部分和退出部分

```
SUB     LR,LR,#4            ;计算返回地址
STMFD   SP!,{R0-R3,LR}      ;保存使用到的寄存器
. . .
LDMFD   SP!,{R0-R3,PC}^     ;中断返回
```

注意：中断返回指令的寄存器列表（其中必须包括PC）后的“^”符号表示这是一条特殊形式的指令。这条指令在从存储器中装载PC的同时（PC是最后恢复的），CPSR也得到恢复。这里使用的堆栈指针SP（R13）是属于异常模式的寄存器，每个异常模式有自己的堆栈指针。这个堆栈指针应必须在系统启动时初始化。

异常进入和退出

进入异常

在异常发生后，ARM9TDMI内核会作以下工作：

1. 在适当的LR中保存下一条指令的地址，当异常入口来自：

   - ARM状态，那么ARM9TDMI将当前指令地址加4或加8复制（取决于异常的类型）到LR中；
   - 为Thumb状态，那么ARM9TDMI将当前指令地址加2、4或加8（取决于异常的类型）复制到LR中；异常处理器程序不必确定状态。

异常进入和退出

- 进入异常

在异常发生后，ARM9TDMI内核会作以下工作：

2.将CPSR复制到适当的SPSR中；

3. 将CPSR模式位强制设置为与异常类型相对应的值；

4.强制PC从相关的异常向量处取指。

异常进入和退出

- 进入异常

ARM9TDMI内核在中断异常时置位中断禁止标志，这样可以防止不受控制的异常嵌套。

注：异常总是在ARM状态中进行处理。当处理器处于Thumb状态时发生了异常，在异常向量地址装入PC时，会自动切换到ARM状态。

异常进入和退出

退出异常

当异常结束时，异常处理程序必须：

1.将LR（R14）中的值减去偏移量后存入PC，偏移量根据异常的类型而有所不同；

2.将SPSR的值复制回CPSR；

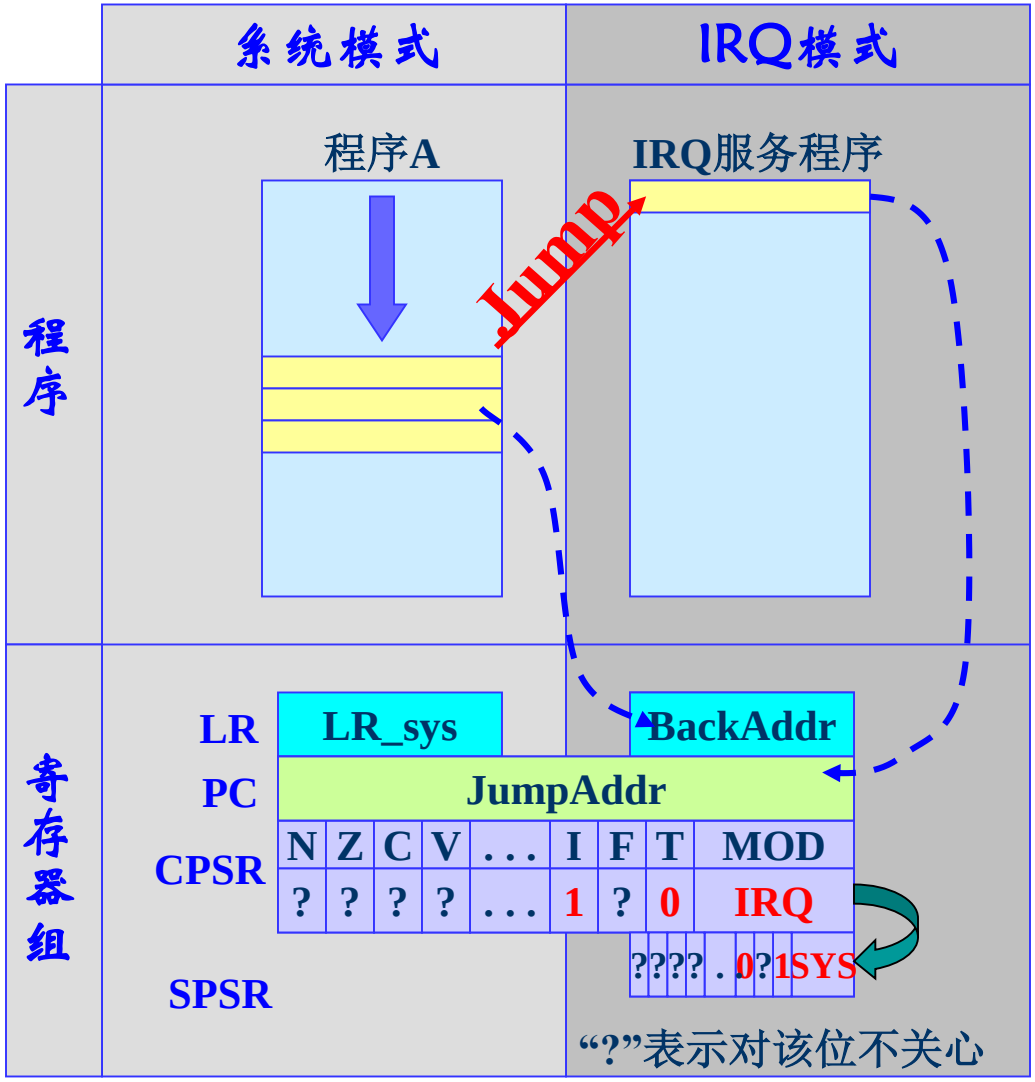
3.清零在入口置位的中断禁止标志。

注：恢复CPSR的动作会将T、F和I位自动恢复为异常发生前的值。

图示进入异常过程

2. 程序运行时发生IRQ中断，硬件寄存器系统模式下运行用户程序，假定当前处理器状态为Thumb状态、允许IRQ中断：

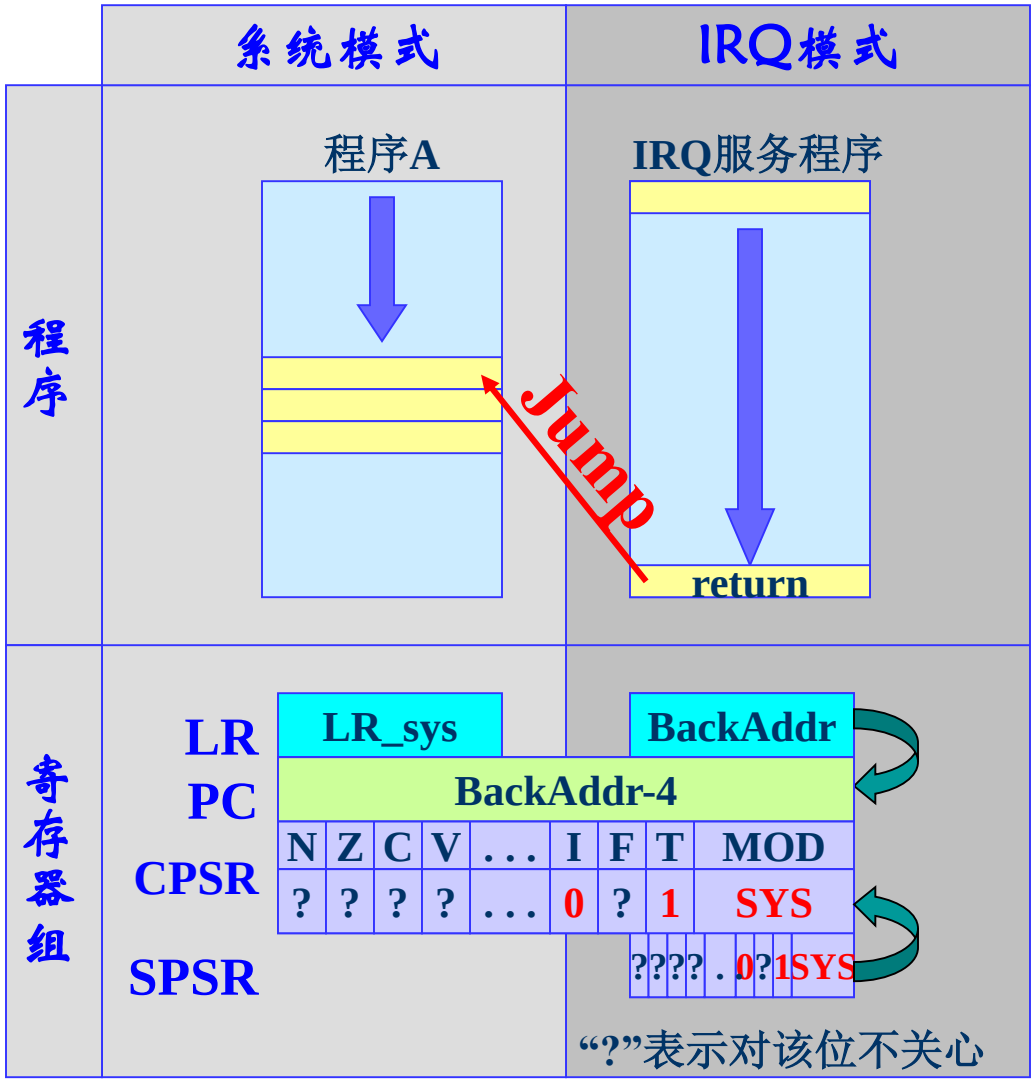
- 将CPSR寄存器内容存入IRQ模式的SPSR寄存器
- 置位I位（禁止IRQ中断）
- 清零T位（进入ARM状态）
- 设置MOD位，切换处理器模式至IRQ模式
- 将下一条指令的地址存入IRQ模式的LR寄存器
- 将跳转地址存入PC，实现跳转



图示退出异常过程

在异常处理结束后，异常处理程序完成以下动作：

- 将SPSR寄存器的值复制回CPSR寄存器；
- 将LR寄存的值减去一个常量后复制到PC寄存器，跳转到被中断的用户程序。



异常入口/出口汇总

异常类型	返回指令	R14当前状态		备注
		ARM R14_x	Thumb R14_x	
软件中断 SWI	MOVS PC,R14_svc	PC+4	PC+2	此处PC是SWI，未定义的指令取指或预取指中止指令的地址
未定义的指令	MOVS PC,R14_und	PC+4	PC+2	
预取指中止	SUBS PC,R14_abt,#4	PC+4	PC+4	
快中断	SUBS PC,R14_fiq,#4	PC+4	PC+4	此处PC为由于FIQ或IRQ占先而没有被执行的指令的地址
中断	SUBS PC,R14_irq,#4	PC+4	PC+4	
数据中止	SUBS PC,R14_abt,#8	PC+8	PC+8	此处PC为产生数据中止的装载或保存指令的地址。
复位	无	—	—	复位时保存在R14_svc中的值不可预知。

注意：“MOVS PC,R14_svc”是指在管理模式执行MOVS PC,R14指令。“MOVS PC,R14_und”、“SUBS PC,R14_abt,#4”等指令也是类似的。

异常优先级

- 多个异常可同时发生，处理器将按固定的优先级顺序进行处理。

异常类型	优先级
复位	1（最高优先级）
数据中止	2
FIQ	3
IRQ	4
预取指中止	5
未定义指令	6
SWI	7（最低优先级）

优先级降低

3.9 异常

快速中断请求

快速中断请求(FIQ)适用于对一个突发事件的快速响应（数据传输和通道处理）保护的需要（这可以加速上下文切换的速度）。通过处理器上的FIQ输入引脚，由外部产生FIQ异常。

不管异常入口是来自ARM状态还是Thumb状态，FIQ处理程序都会通过执行下面的指令从中断返回：

```
SUBS    PC,R14_fiq,#4
```

在一个特权模式中，可以通过置位CPSR中的F位来禁止FIQ异常。若F位清零，则ARM在执行完当前指令后时检查FIQ输入引脚。

3.9 异常

中断请求

中断请求（IRQ）异常是一个由nIRQ输入端的低电平所产生的正常中断（在具体的芯片中，nIRQ由片内外设拉低，nIRQ是内核的一个信号，对程序员不可见）。IRQ的优先级低于FIQ。对于FIQ序列它是被屏蔽的。任何时候在一个特权模式下，都可通过置位CPSR中的I位来禁止IRQ。

不管异常入口是来自ARM状态还是Thumb状态，IRQ处理程序都会通过执行下面的指令从中断返回：

```
SUBS    PC,R14_irq,#4
```

3.9 异常

- 中止

中止意味着对当前存储器的访问不能完成，由ABORT输入信号触发（高电平），在存储器访问周期末处理器检查该信号。中止包含两种类型：

- 预取中止　发生在指令预取过程中
- 数据中止　发生在对数据访问时

3.9 异常

- 中止——预取指中止

当发生预取中止时，ARM7TDMI内核将预取的指令标记为无效，但在指令到达流水线的执行阶段时才进入异常。如果指令在流水线中因为发生分支而没有被执行，中止将不会发生。

在处理中止的原因之后，不管处于哪种处理器操作状态，处理程序都会执行下面的指令恢复PC和CPSR并重试被中止的指令：

```
SUBS    PC,R14_abt,#4
```

3.9 异常

- 中止——数据中止

当发生数据中止后，根据产生数据中止的指令类型作出不同的处理：

- 数据转移指令（LDR、STR）回写到被修改的基址寄存器。中止处理程序必须处理这一细节。

LDR R0，[R1, #4]!　　;R1←R1+4　R0←[R1]

- 交换指令（SWP）中止好像没有被执行过一样（中止发生在SWP指令进行读访问时）

SWP Rd, Rm, Rn　　;Rd←[Rn]　[Rn]←Rm

3.9 异常

- 中止——数据中止

在修复产生中止的原因后，不管处于哪种处理器操作状态，处理程序都必须执行下面的返回指令，重试被中止的指令：

```
SUBS    PC,R14_abt,#8
```

3.9 异常

- 软件中断指令

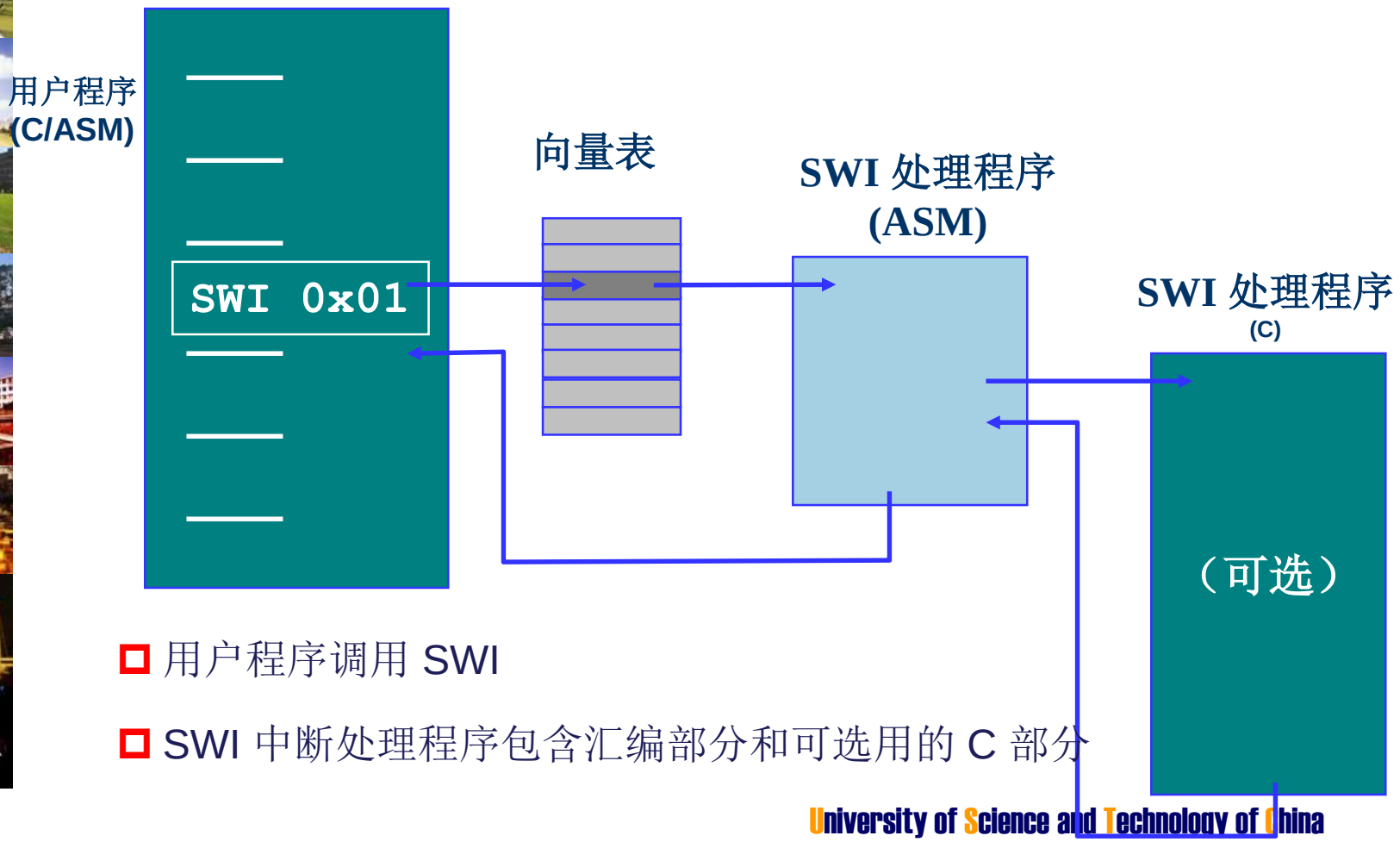
使用软件中断(SWI)指令可以进入管理模式，通常用于请求一个特定的管理函数。SWI处理程序通过执行下面的指令返回：

```
MOVS    PC,R14_svc
```

这个动作恢复了PC并返回到SWI之后的指令。SWI处理程序读取操作码以提取SWI函数编号。

3.9 异常

- 软件中断指令



- 用户程序调用 SWI
- SWI 中断处理程序包含汇编部分和可选用的 C 部分

3.9 异常

- 未定义的指令

当ARM9TDMI处理器遇到一条自己和系统内任何协处理器都无法处理的指令时，ARM9TDMI内核执行未定义指令陷阱。软件可使用这一机制通过模拟未定义的协处理器指令来扩展ARM指令集。

注：ARM9TDMI处理器完全遵循ARM结构v5T，可以捕获所有分类未被定义的指令位格式。

3.9 异常

- 未定义的指令

在模拟处理了失败的指令后，陷阱程序执行下面的指令：

```
MOVS    PC,R14_svc
```

这个动作恢复了PC、CPSR并返回到未定义指令之后的指令。

3.10 复位

复位

当nRESET信号再次变为高电平时，ARM处理器执行下列操作：

1.强制CPSR中的M[4:0]变为b10011（管理模式）；

2.置位CPSR中的I和F位；

3.清零CPSR中的T位；

4.强制PC从地址0x00开始对下一条指令进行取指；

5.返回到ARM状态并恢复执行。

谢谢！

Q & A