第2章 ARM Cortex-M体系结构

2.1 ARM Cortex体系概述

2.2 Cortex-M4内核基础

2.3 存储器系统

2.4异常和中断

电子与通信工程系

华东理工大学信息学院

2.1 ARM Cortex体系概述

2.1 ARM Cortex体系概述

ARM公司在经典处理器ARM11以后的产品都改用Cortex命名，主要分成A、R和M三类

A　尖端的基于虚拟内存的操作系统和用户应用；

R　针对实时系统；

M　针对微控制器。

2.1 ARM Cortex体系概述

2.1.1 CISC和RISC

从现阶段的主流体系结构来讲，指令集可分为复杂指令集（CISC）和精简指令集（RISC）两部分。

表2-1　RISC和CISC特点对比

项目	RISC	CISC
指令系统	简单、精简	复杂、丰富
指令数目	一般小于100条	一般大于200条
指令格式	少	多
寻址方式	少	多
指令字长	基本等长	不固定
可访存指令	主要是Load/Store	不加限制
各种指令使用频率	相差不大	相差很大
各种指令执行时间	大部分单周期	相差很大
优化编译实现	较容易	难

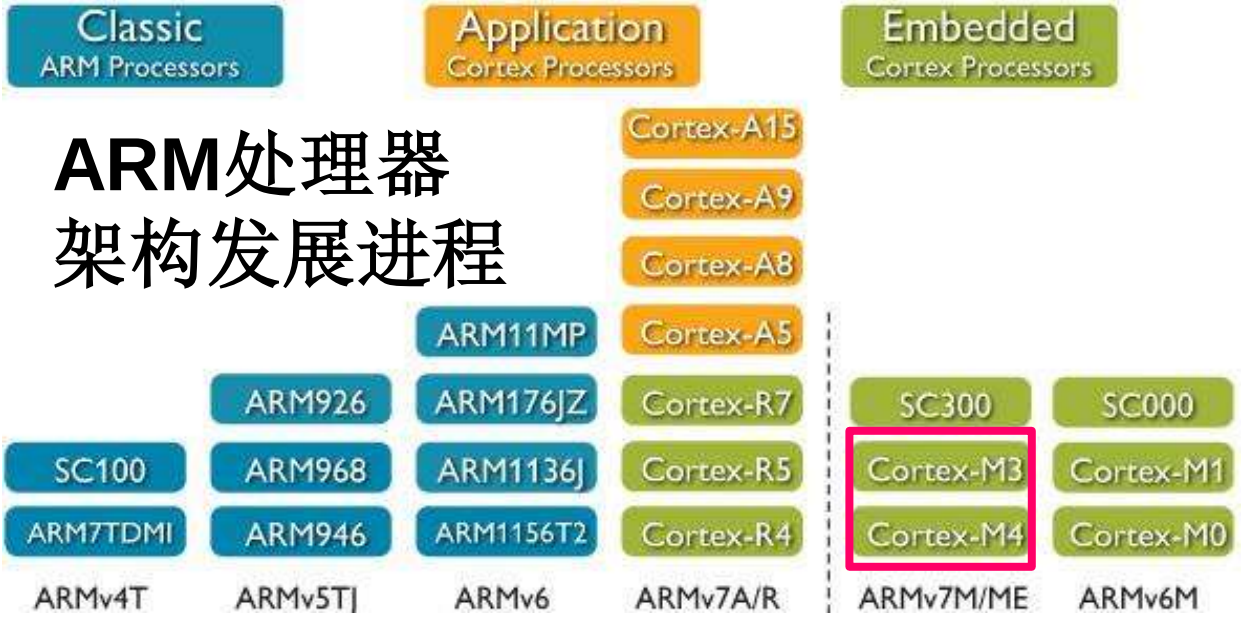
2.1 ARM Cortex体系概述

2.1.2 ARM架构发展史

- 1985年，Roger Wilson和Steve Furber设计了第一代32位、6MHz的处理器，做出了一台RISC指令集的计算机，简称ARM（Acorn RISC Machine）——ARM的由来。
- 1990年，Acorn公司正式改组为ARM计算机公司。
- ARM公司生产的芯片，称为ARM芯片。
- ARM公司的业务一直是出售IP核。
- ARM控股的ARM处理器家庭的突出例子包括ARM7, ARM9, ARM11 and Cortex-M，Cortex-A系列 。

2.1 ARM系统概述

2.1.2 ARM架构发展史



ARM推出新款ARMv8架构。包括Cortex-A53、Cortex-A57、 Cortex-A73、 Cortex-A75 、 Cortex-A76等处理器。

2.1 ARM系统概述

2.1.3 ARM体系结构类型

ARM处理器分为6类：Cortex-A系列处理器、Cortex-R系列处理器、Cortex-M系列处理器、Machine Learning系列处理器、SecurCore系列处理器、Neoverse系列处理器。

表2-2　Cortex系列处理器的主要特征

项目	Cortex-A系列处理器	Cortex-R系列处理器	Cortex-M系列处理器
设计特点	高时钟频率，长流水线，高性能，对媒体处理支持（NEON指令集扩展）	高时钟频率，较长的流水线，高确定性（中断延迟低）	较短的流水线，超低功耗
系统特性	内存管理单元（MMU），cache memory，ARM TrustZone®安全扩展	内存保护单元（MPU），cache memory，紧耦合内存（TCM）	内存保护单元（MPU），嵌套向量中断控制器（NVIC），唤醒中断控制器（WIC），最新ARM TrustZone®安全扩展
目标市场	移动计算、智能手机、高能效服务器、高端微处理器	工业微控制器、汽车电子、硬盘控制器	微控制器、深度嵌入系统（如传感器、MEMS、混合信号IC、IoT）

2.1 ARM系统概述

2.1.3 ARM体系结构类型

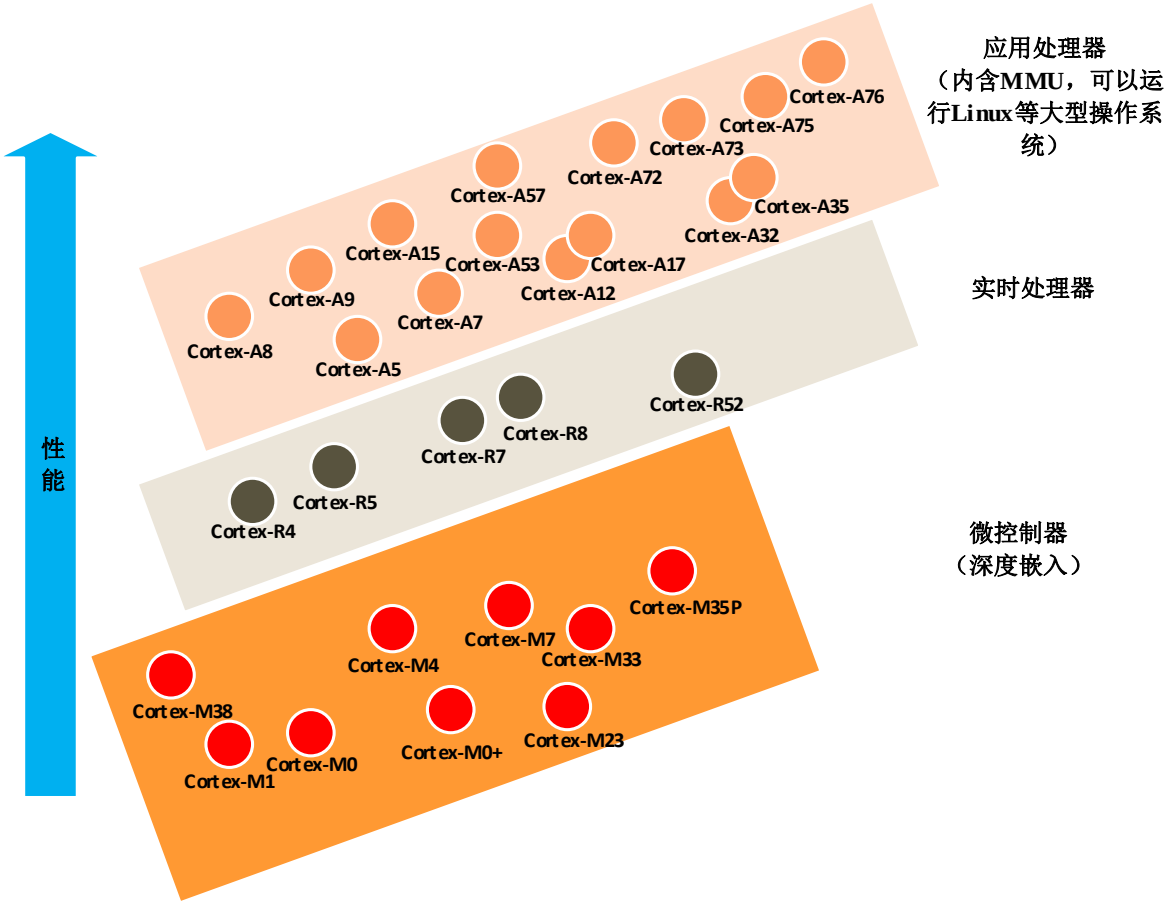


图2-1　Cortex系列处理器发展历程简图

2.1 ARM系统概述

2.1.4 Cortex-M系列处理器

表2-2　Cortex系列处理器的主要特征

处理器	主要特性
Cortex-M0处理器	
Cortex-M0+处理器	
Cortex-M1处理器	
Cortex-M3处理器	针对低功耗微控制器设计的处理器，面积小但是性能强劲，支持可以处理器快速处理复杂任务的丰富指令集，具有硬件除法器和乘加指令（MAC）。并且，由于Cortex-M3处理器支持全面的调试和跟踪功能，软件开发者可以快速地开发他们的应用

2.1 ARM系统概述

2.1.4 Cortex-M系列处理器

表2-2　Cortex系列处理器的主要特征-续

处理器	主要特性
Cortex-M4处理器	不但具备Cortex-M3处理器的所有功能，而且扩展了面向数字信号处理（DSP）的指令集，如单指令多数据指令（SMID）和更快的单周期MAC操作。此外，它还有一个可选的支持IEEE754浮点标准的单精度浮点运算单元
Cortex-M7处理器	
Cortex-M23处理器	
Cortex-M33处理器	

2.2 Cortex-M4内核基础

2.2 Cortex-M4内核基础

Cortex-M4处理器已设计具有适用于数字信号控制市场的多种高效信号处理功能。Cortex-M4处理器采用扩展的单周期MAC指令、优化的SIMD指令、饱和运算指令和一个可选的单精度浮点单元（FPU）。

表2-4　Cortex-M4系列处理器数字信号处理功能

硬件体系结构	单周期16位、32位MAC
用于指令提取的32位AHB-Lite接口 用于数据和调试访问的32位AHB-Lite接口	大范围的MAC 32位或64位累加选择 指令在单个周期中执行
单周期SIMD	单周期双16位MAC
4路并行8位加法或减法 2路并行16位加法或减法 指令在单个周期中执行	2路并行16位MAC 32位或64位累加选择 指令在单个周期中执行
浮点单元	其他
符合IEEE 754标准 单精度浮点单元 用于获得更高精度的融合MAC	饱和数学 桶形移位器

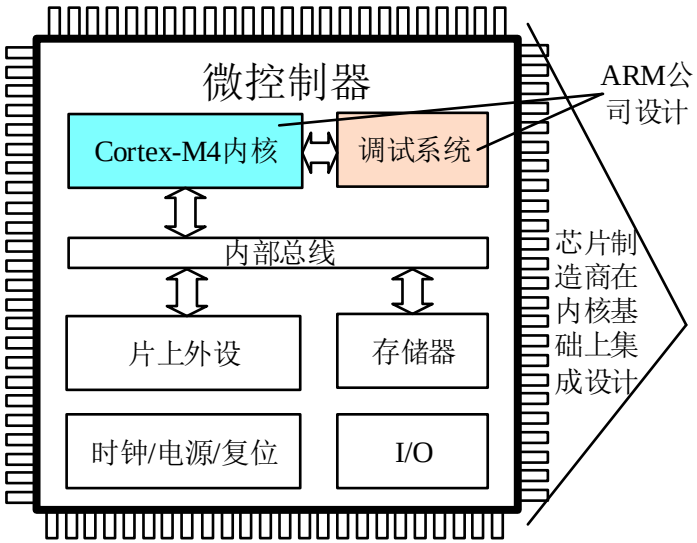


图2-2　Cortex-M4系列微控制器内部构造

2.2 Cortex-M4内核基础

基于Cortex-M3/M4的微控制器芯片

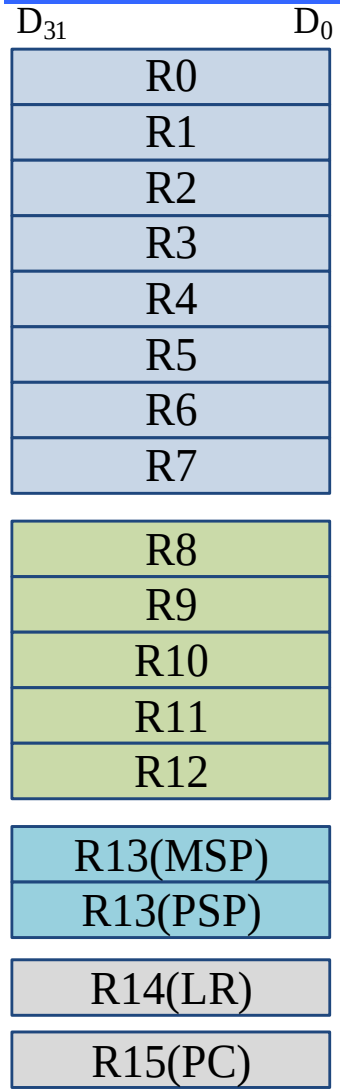
2.2 Cortex-M4内核基础

1. 32-bit微控制器： 32-bit 寄存器组、存储器接口。
2. 哈佛架构：独立的指令总线和数据总线。
3. 存储空间：4GB。
4. 寄存器：寄存器 (R0 到 R15) 和 特殊功能寄存器。
5. 运行模式：线程模式和处理器模式;特权级和用户级。
6. 中断和异常：内置嵌套向量中断控制器；支持11 种系统异常外加240种外部 IRQ。
7. 总线接口：若干总线接口允许 Cortex-M4 同时取指令和取数据。
8. MPU：一个可选的存储器保护单元允许对特权访问和用户程序访问制定访问规则。
9. 指令集：Thumb-2 指令集；允许 32位指令和16位指令被同时使用。
10. 内部调试组件：提供在线调试功能，例如：断点、单步、变量查看。

2.2 Cortex-M4内核基础

2.2.1 寄存器组

1、R0-R12



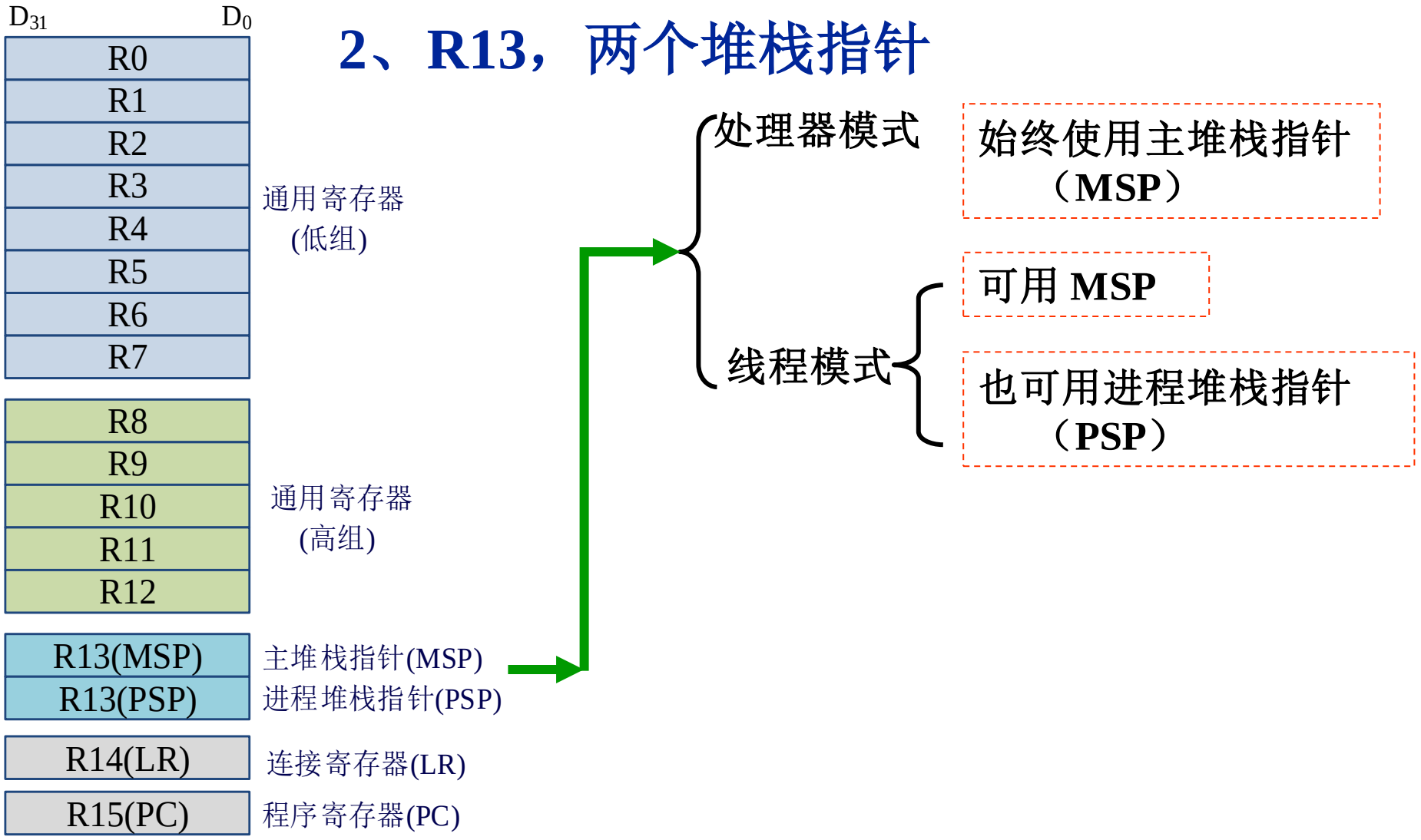
r0～r12 ,为13个通用目的寄存器

低组寄存器, r0-r7可以被指定通用寄存器的所有指令访问

高组寄存器, r8-r12可以被指定通用寄存器的所有32位指令访问，16位Thumb指令不能访问它们，32位的Thumb－2指令则不受限制。

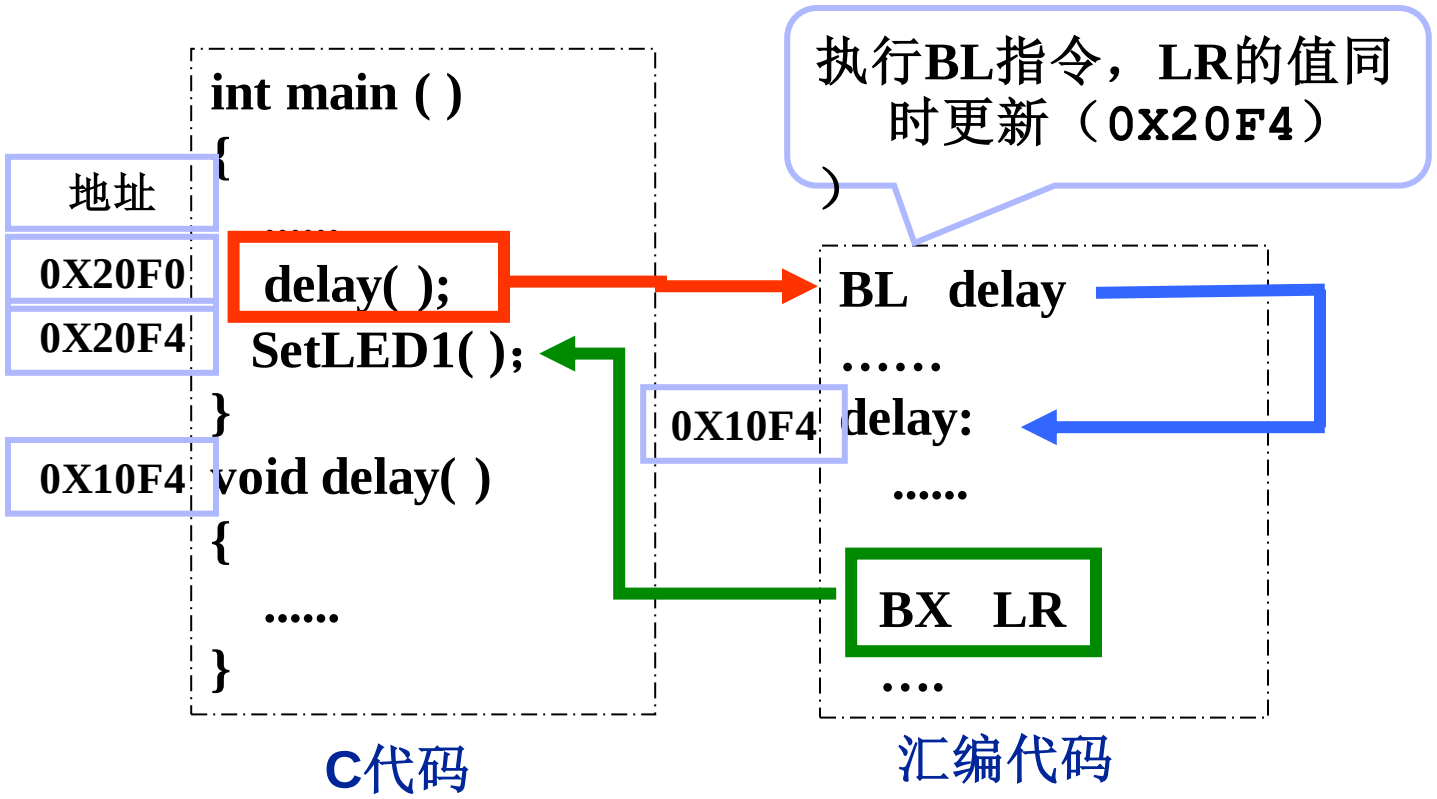
2.2 Cortex-M4内核基础

2、R13，两个堆栈指针



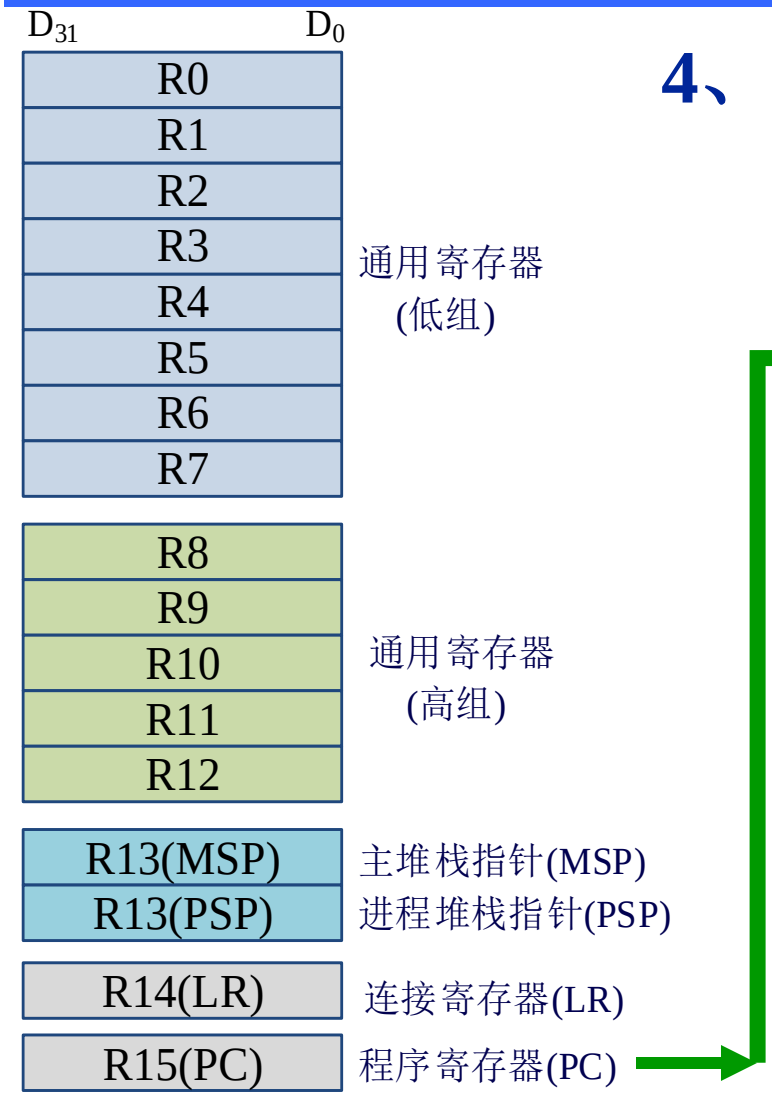
2.2 Cortex-M4内核基础

3、链接寄存器（LR）



2.2 Cortex-M4内核基础

4、 程序寄存器（PC）



程序计数器（PC）

程序计数器 总是指向正在取指的指令。该寄存器的位0始终为0，因此，指令始终与字或半字边界对齐。

R15 **是程序计数器。可以在汇编语言中通过R15或PC访问。**

2.2 Cortex-M4内核基础

2.2.2 堆栈操作

1. 栈的作用

（1）用于在正在执行的函数需要使用寄存器（寄存器组）进行数据处理时，临时存储数据的初始值，这些数据在函数结束时可以被恢复出来，以免调用函数的程序丢失数据。

（2）用于函数或子程序中的信息传递。

（3）用于存储局部变量。

（4）用于在中断等异常产生时保存处理器状态和寄存器数值。

四种类型：

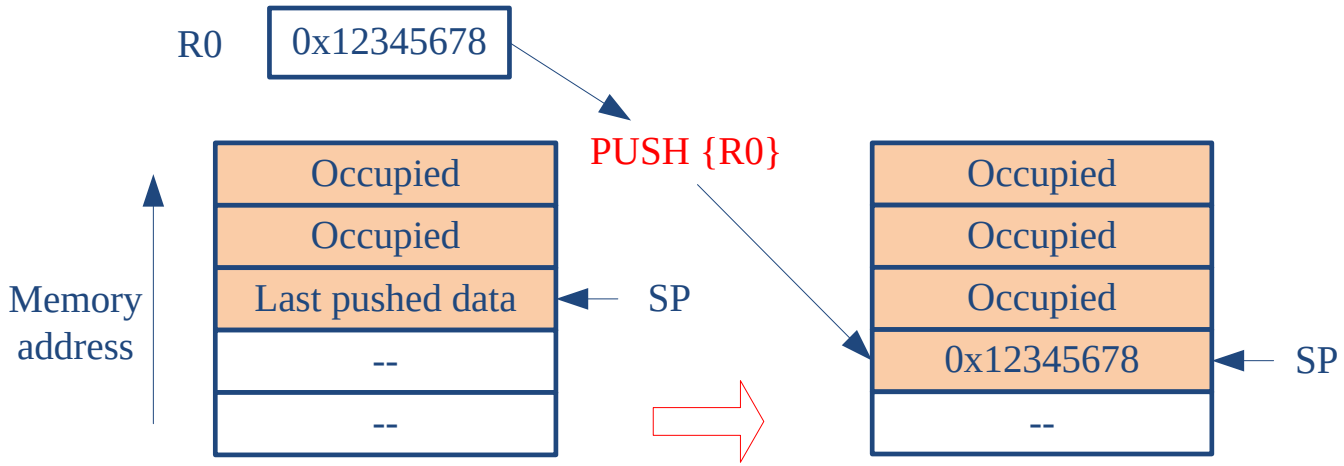
① 满递减堆栈。
② 满递增堆栈。
③ 空递增堆栈。
④ 空递减堆栈。

2.2 Cortex-M4内核基础

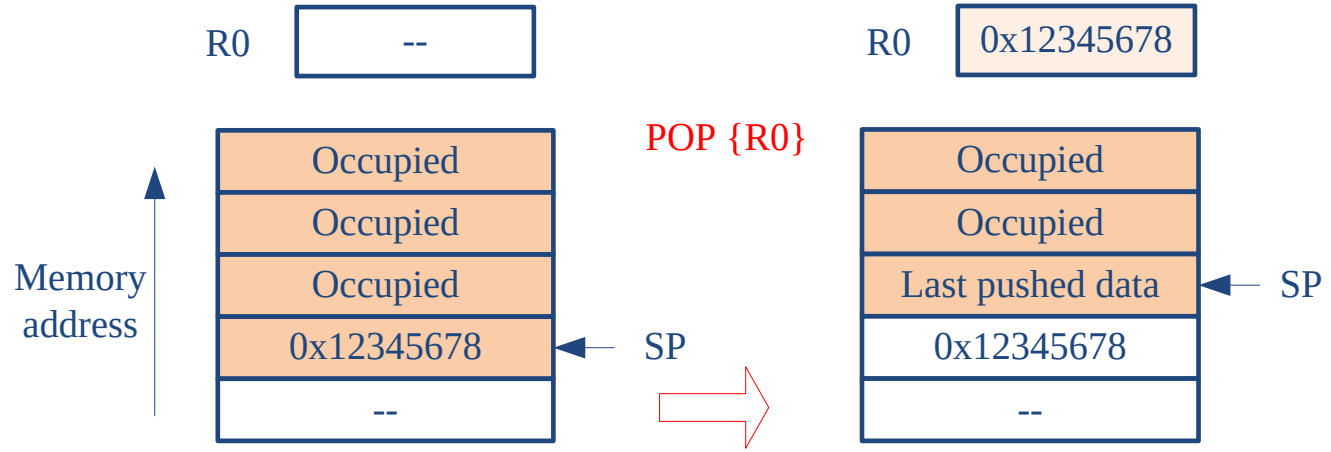
2.2.2 堆栈操作

默认：满递减堆栈

2、入栈

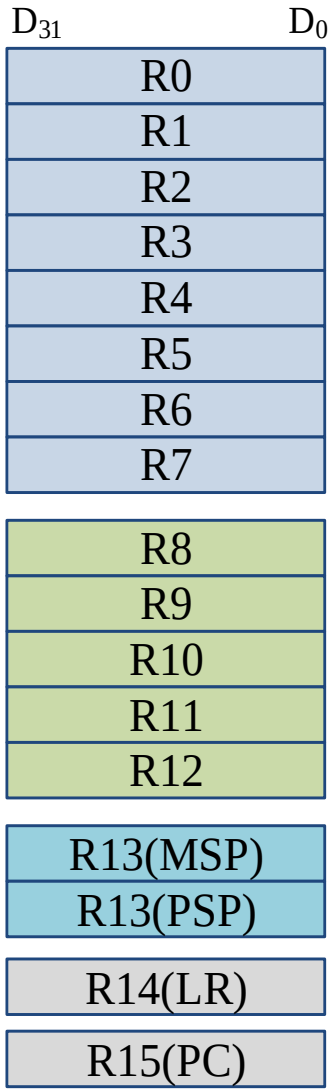


3、出栈



2.2 Cortex-M4内核基础

3、 链接寄存器（LR）



链接寄存器（LR）

在执行分支(branch)和链接(BL)指令或带有交换的分支和链接指令(BLX)时，LR用于保存PC的返回地址。

主要用于保存子程序的 返回地址。

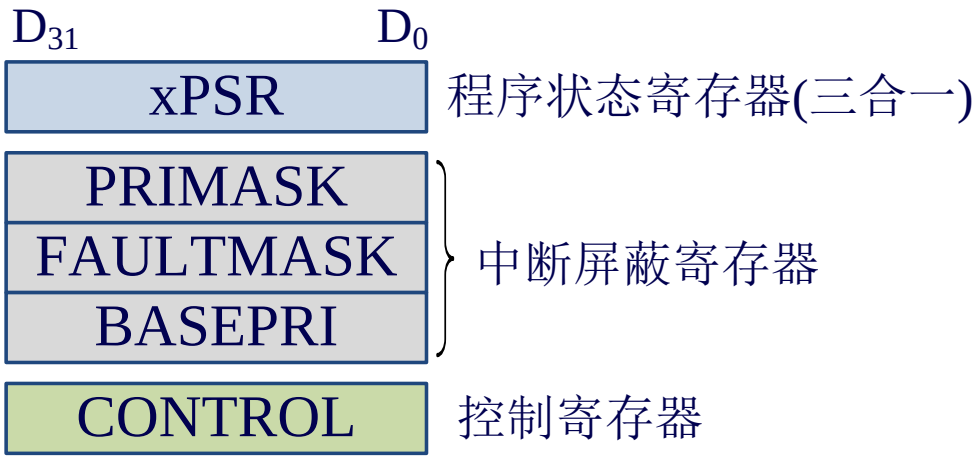
LR也用于异常返回。

2.2 Cortex-M4内核基础

2.2.3 特殊功能寄存器

在Cortex-M3/M4处理器中的特殊寄存器包括:

1. 程序状态寄存器 (PSRs)
2. 中断屏蔽寄存器(PRIMASK, FAULTMASK, and BASEPRI)
3. 控制寄存器 (CONTROL)



2.2 Cortex-M4内核基础

1 程序状态寄存器 (PSRs)

程序状态寄存器可以分为三个状态寄存器：

1. 应用 PSR (APSR) [负、零、进/借、溢、饱和(用于饱和运算)]
2. 中断 PSR (IPSR)
3. 执行 PSR (EPSR) [ICI/IT: ICI指令/IT指令状态位；T总为1]

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q											
IPSR												Exception No.				
EPSR						ICI/IT	T			ICI/IT						

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
xPSR	N	Z	C	V	Q	ICI/IT	T			ICI/IT		Exception No.				

2.2 Cortex-M4内核基础

2 中断屏蔽寄存器PRIMASK、FAULTMASK 和 BASEPRI

用来开放/禁用异常

Cortex-M3/M4 中断屏蔽寄存器

寄存器名	描 述
PRIMASK	一个1-bit 寄存器。1：仅允许NMI 和硬件默认异常，所有其他的中断和异常将被屏蔽；0：开放中断
FAULTMASK	一个1-bit 寄存器。1：仅允许NMI，所有中断和默认异常处理包括硬件异常被忽略。
BASEPRI	一个9位寄存器。它定义了屏蔽优先级。 当设置为某值时，所有大于或等于该值的中断被屏蔽（值越大，优先级越低）。全为0（默认值）：不屏蔽任何中断。

2.3 存储器系统

2.3 存储器系统

Cortex-M4内核的存储器系统的主要特性如下:

（1）可寻址4GB线性地址物理空间。

（2）支持小端和大端的存储器系统。Cortex-M4处理器可以选择使用小端或者大端的存储器系统。

（3）位带访问。

（4）写缓冲。对可缓冲存储器区域写操作需要花费几个周期时间，Cortex-M4处理器的写缓冲可以把写操作缓存起来，因此处理器可以继续执行下一条指令，从而提高了程序的执行速度。

（5）存储器保护单元（MPU）。MPU定义了各存储器区域的访问权限，且为可编程。Cortex-M4处理器中的MPU支持8个可编程区域，可在嵌入式操作系统中提高系统的健壮性。Cortex-M4处理器中的MPU是可选的。多数应用不会用到MPU，可以忽略。

（6）非对齐传输支持。ARMv7-M架构的所有处理器（包括Cortex-M4处理器）支持非对齐传输。

2.3 存储器系统

2.3.1 数据类型

Cortex-M4处理器支持以下数据类型：32位字、16位半字、8位字节。

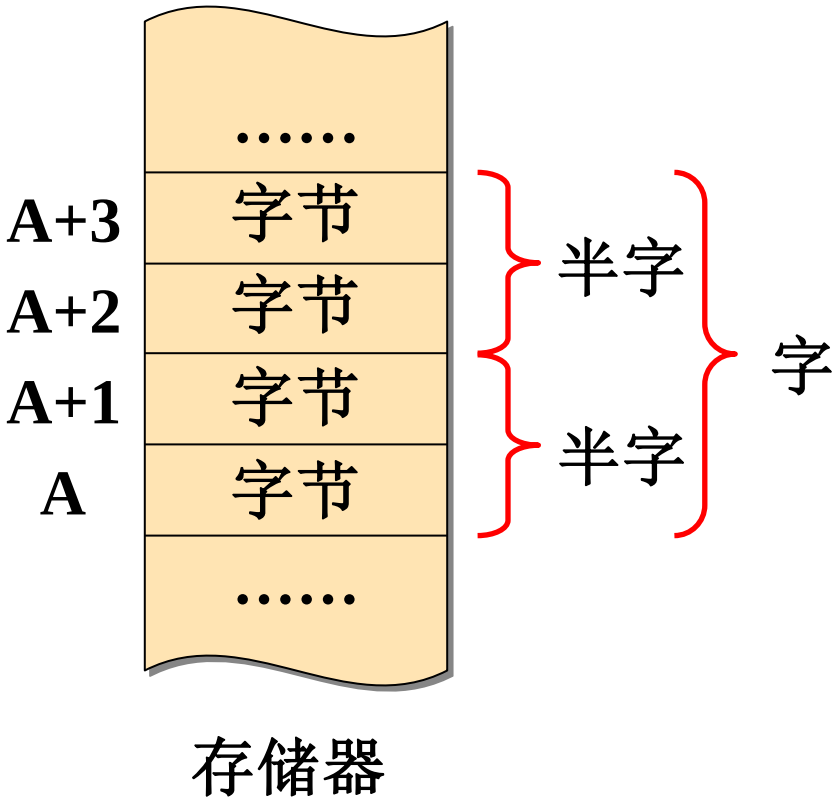
存储器介质：

RAM：SRAM和DRAM

ROM：ROM、PROM、EPROM、EEPROM

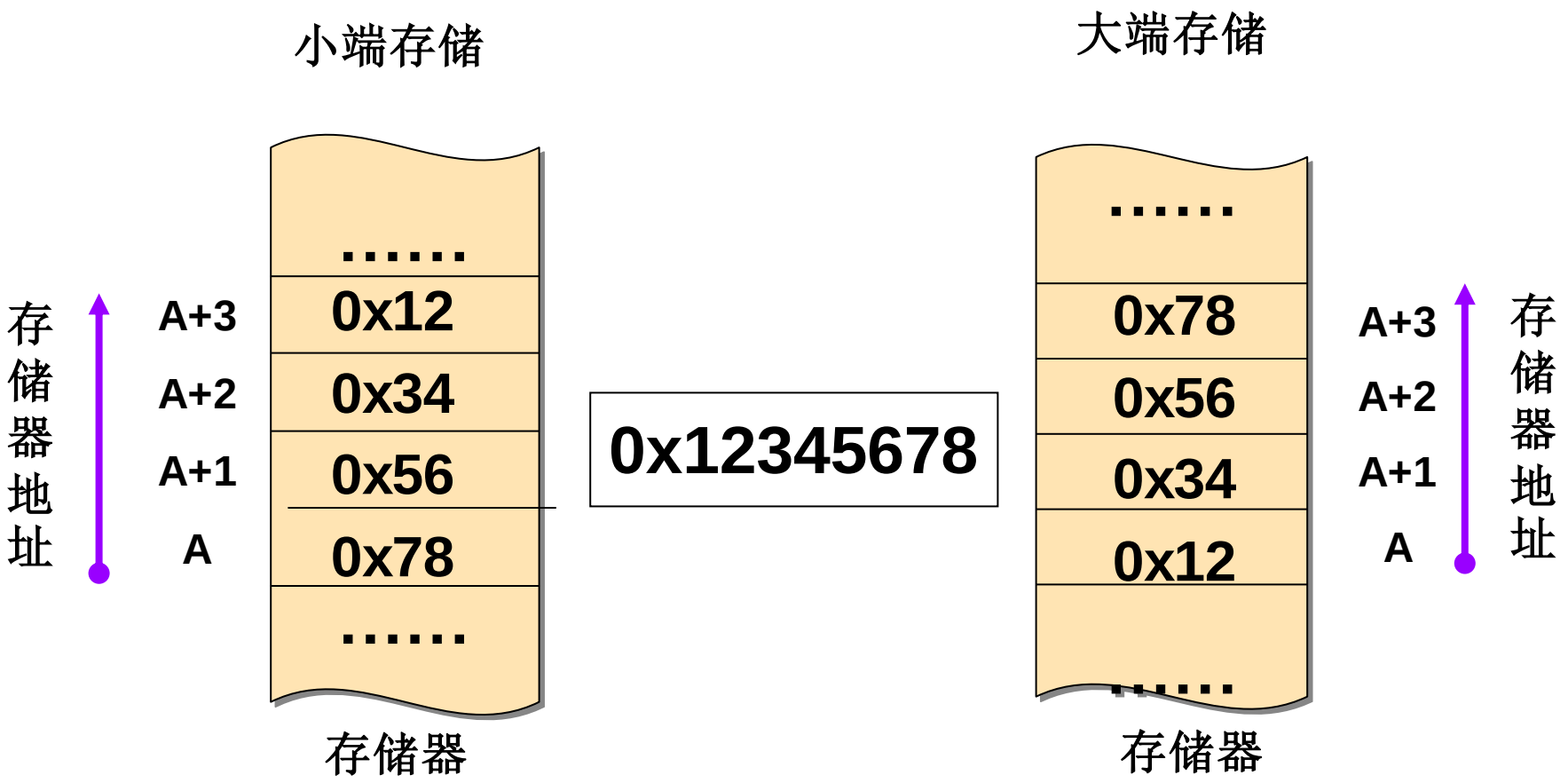
FLASH：NOR FLASH、NAND FLASH

2 存储器格式



2.3 存储器系统

2.3.2 存储形式

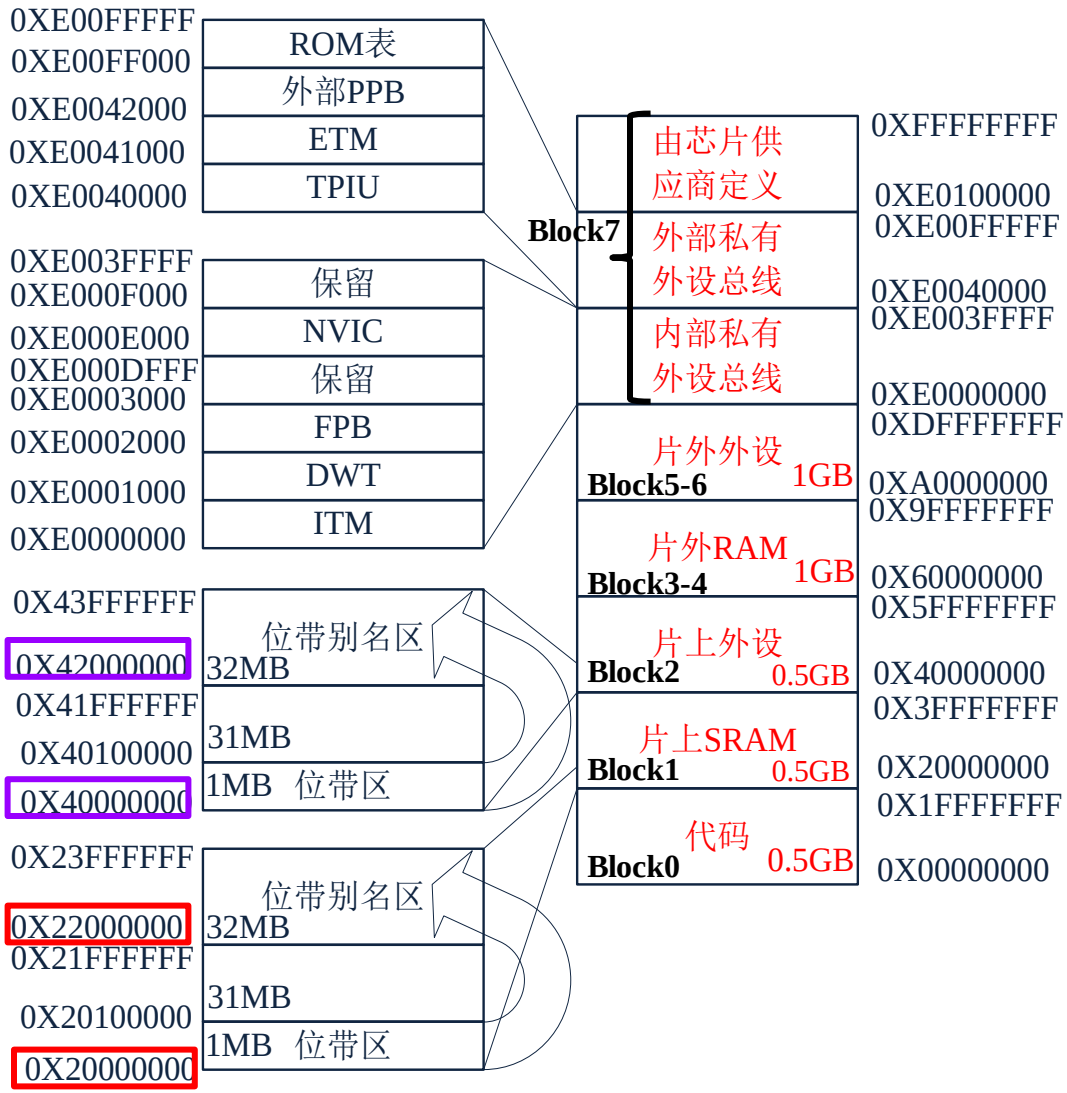


2.3 存储器系统

2.3.3 存储器映射

Cortex-M4 处理器有一个固定的存储映射。这一点极大地方便了软件在各种M4 微控制器间的移植。

例：各款M4微控制器的NVIC 和MPU都在相同的位置布设寄存器，使得它们变得通用。



2.3 存储器系统

2.3.4 位带区

●Cortex-M4支持位带操作，可以使用普通的加载/存储指令来对单一的比特进行读写。

●在Cortex-M4中，有两个区中实现了位带：

（1）SRAM区的最低1MB范围（0x20000000～0x20100000）。

（2）片内外设区的最低1MB范围（0x40000000～0x40100000）。

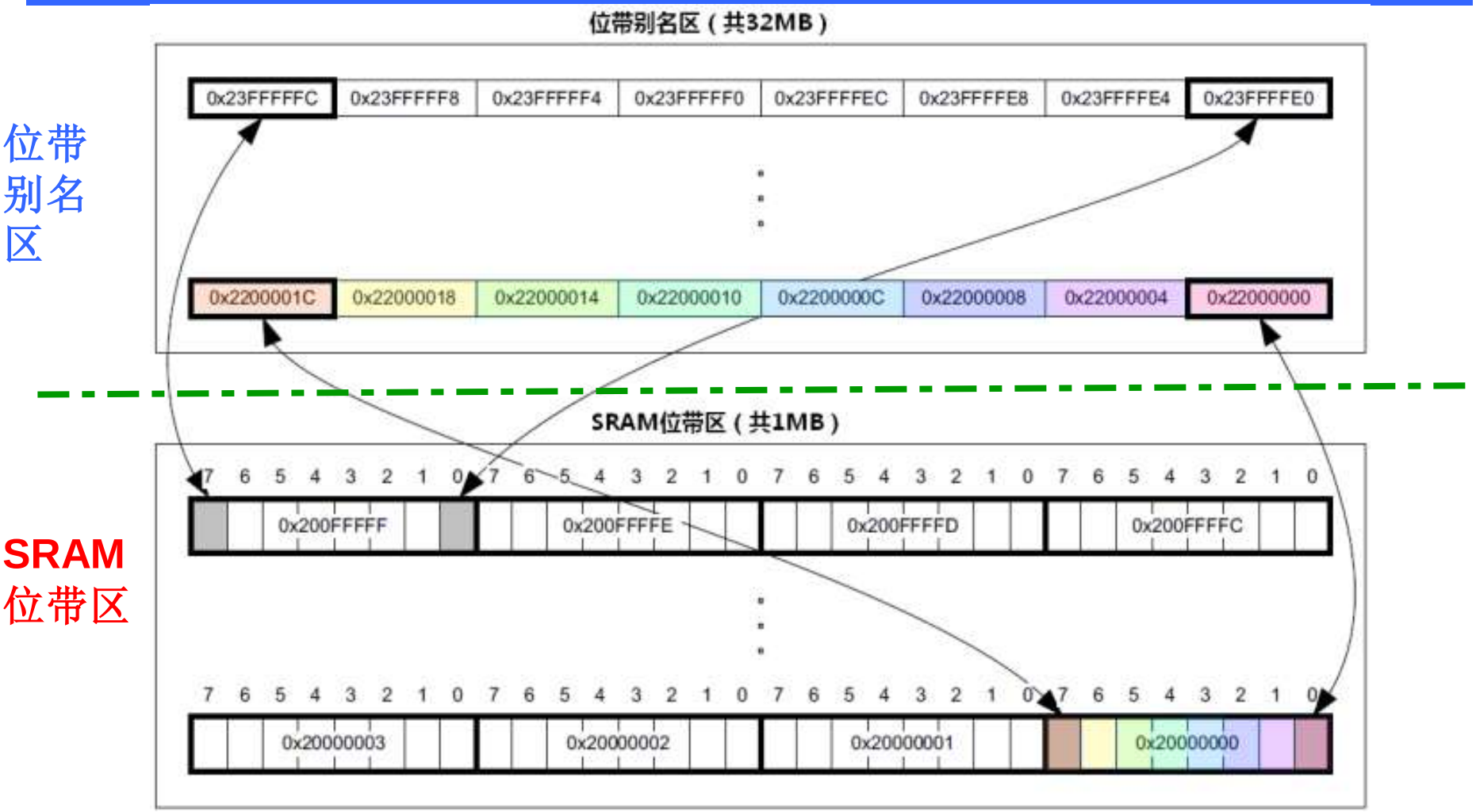
●这两个区中的地址除了可以像普通的RAM一样使用外，它们还都有自己的“位带别名区”，位带别名区把每个比特膨胀成一个32位的字。

（1）32MB SRAM位带别名区（0x22000000～0x23FFFFFF）。

（2）32MB外设位带别名区（0x42000000～0x43FFFFFF）。

●它们可以通过一个单独的被称为bit-band alias的存储区域被访问。

2.3 存储器系统



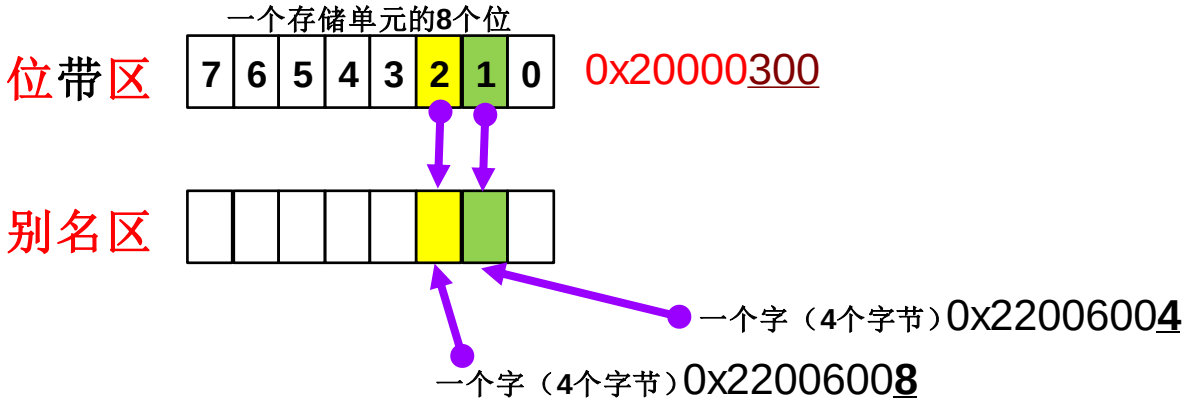
位通过Bit-Band Alias访问Bit-Band Region

2.3 存储器系统

位带别名区中的字与位带区的位映射公式

```
bit_word_addr = bit_band_base +
(byte_offset x 32) + bit_number × 4
```

例如：SRAM 位带区中地址为 0x20000300 的字节中的位 2 被映射到别名区中的地址为：0x22006008（= 0x22000000 + (0x300*32) + (2*4)）的字。



2.3 存储器系统

SRAM 区域的Bit-Band 地址重映射

Bit-Band 区域	别名等效
0x20000000 bit[0]	0x22000000 bit[0]
0x20000000 bit[1]	0x22000004 bit[0]
0x20000000 bit[2]	0x22000008 bit[0]
…	…
0x20000000 bit[31]	0x2200007C bit[0]
0x20000004 bit[0]	0x22000080 bit[0]
…	…
0x20000004 bit[31]	0x220000FC bit[0]
…	…
0x200FFFFC bit[31]	0x23FFFFFC bit[0]

外设存储区 Bit-Band 地址重映射

Bit-Band 区域	别名等效
0x40000000 bit[0]	0x42000000 bit[0]
0x40000000 bit[1]	0x42000004 bit[0]
0x40000000 bit[2]	0x42000008 bit[0]
…	…
0x40000000 bit[31]	0x4200007C bit[0]
0x40000004 bit[0]	0x42000080 bit[0]
…	…
0x40000004 bit[31]	0x420000FC bit[0]
…	…
0x400FFFFC bit[31]	0x43FFFFFC bit[0]

2.3 存储器系统

C语言中宏定义位带操作

位带操作定义：

```
#define BITBAND(addr, bitnum)   ((addr &0xF0000000)+0x2000000+((addr
&0xFFFFF)<<5)+(bitnum<<2))

#define MEM_ADDR(addr)          *((volatile unsigned int*)(adr))

#define DEVICE_REG0             0x40000000
```

位带操作定义：　（将单元0x40000000的1位置位）

```
MEM_ADDR(BITBAND(DEVICE_REG0,1)) = 0x1;
```

普通操作（读修改写）：

```
*((volatile unsigned int*) (0x40000000)) |= (0x00000001<<1);
```

2.4 异常和中断

2.4 异常和中断

异常和中断的作用是指示系统中的某个地方发生一些事件，需要引起处理器（包括正在执行的程序和任务）的注意。

当中断和异常发生时，典型的结果是迫使处理器将控制从当前正在执行的程序或任务转移到另一个例程或任务中，该例程叫作中断服务程序（ISR），或者异常处理程序。

中断：请求信号来自于CM4内核外面，来自片上外设或外扩的外设；

异常：是由于CM4内核的活动而产生的 。

只要正常的程序被暂时中止，微控制器就进入处理器模式。包括复位、系统故障、中断等事件。

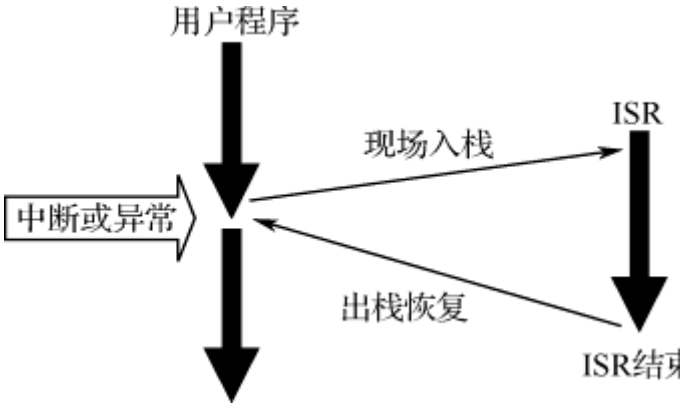


图2-15　异常和中断程序状态切换图

2.4 异常和中断

2.4.2 优先级

1. 优先级概述

所有异常都有其相关的优先级，具有以下特点。

（1）一个较低的优先级值，表示较高的优先级；

（2）可配置的优先事项为除复位、硬件复位和NMI处的所有异常。

在处理器的异常模型中，优先级决定了处理器何时及怎样处理异常。优先级分两组，抢占优先级和响应优先级，程序能够指定中断的优先级。

NVIC支持由软件指定的优先级，可以将中断的优先级指定为0～255。硬件优先级随着中断号的增加而降低，0优先级高，255优先级低。指定软件优先级后，硬件优先级无效。

2.4 异常和中断

2.4.2 优先级

2. 优先级分组

为了对具有大量中断的系统加强优先级控制，NVIC支持优先级分组机制，分为抢占优先级区和响应优先区。

表2-13 抢占优先级和亚优先级的表达位数与分组位置的关系

分组位置	表示抢占优先级的位段	表示亚优先级的位段
0	[7:1]	[0:0]
1	[7:2]	[1:0]
2	[7:3]	[2:0]
3	[7:4]	[3:0]
4	[7:5]	[4:0]
5	[7:6]	[5:0]
6	[7:7]	[6:0]
7	无	[7:0]（所有位）

2.4 异常和中断

2.4.2 优先级

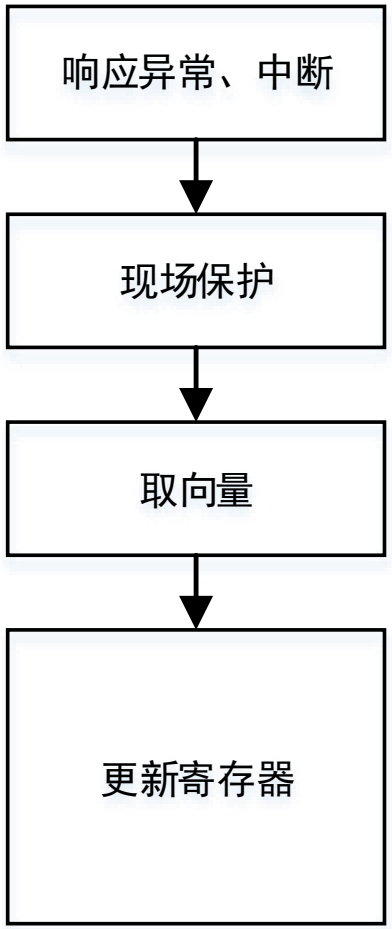
在大多数实际微控制器中只用到了部分优先级分组。例如，在STM32F429微控制器中对应于中断的8位优先级寄存器（IP）只用到高4位，而低4位保留。这样优先级分组的值只能设置为3～7。下面通过一个例子来加深对优先级分组的应用。

假设将STM32F429微控制器优先级分组的值设置为4（从bit 4处分组，5～7位是占先优先级，0～4位是次优先级，但是0～3位保留不用），则可得到8级抢占优先级，且在每个抢占优先级的内部有1个次优先级。STM32F429微控制器抢占优先级和次优先级在优先级寄存器中的位数关系如图2-16所示。

图2-16 STM32F429微控制器抢占优先级和次优先级在优先级寄存器中的位数关系

2.4 异常和中断

2.4.3 响应过程



1、 响应流程

在所选的堆栈上将 xPSR、 PC、 LR 、R12、 R3、 R2、 R1、 R0 压栈

流水线在向量地址处（向量表基址+(异常号*4)）读取中断函数入口（I—Code总线）。

SP(执行ISR时使用MSP)；PSR（更新IPSR位段为新的异常编号）；PC(ISR入口地址)；LR设置为EXC_RETURN，以便从异常中退出 。 ISR返回时执行BX LR即可返回原程序断点处。

注意：

LR的值被从新定义并非进入异常程序的地址

2.4 异常和中断

2.4.3 响应过程

2. 退出异常步骤

（1）根据EXC_RETURN指示的堆栈，弹出进入中断时被压栈的八个寄存器。
（2）从刚出栈的IPSR寄存器[8:0]位检测恢复到哪个异常（此时为嵌套中断中），若为0则恢复到线程模式。
（3）根据EXC_RETURN，选择使用相应堆栈指针。

2.4 异常和中断

3．抢占

在中断处理程序中，一个新的中断比当前的中断优先级更高，处理器打断当前的中断处理程序流程，去响应优先级更高的中断，此时产生抢占或中断嵌套。

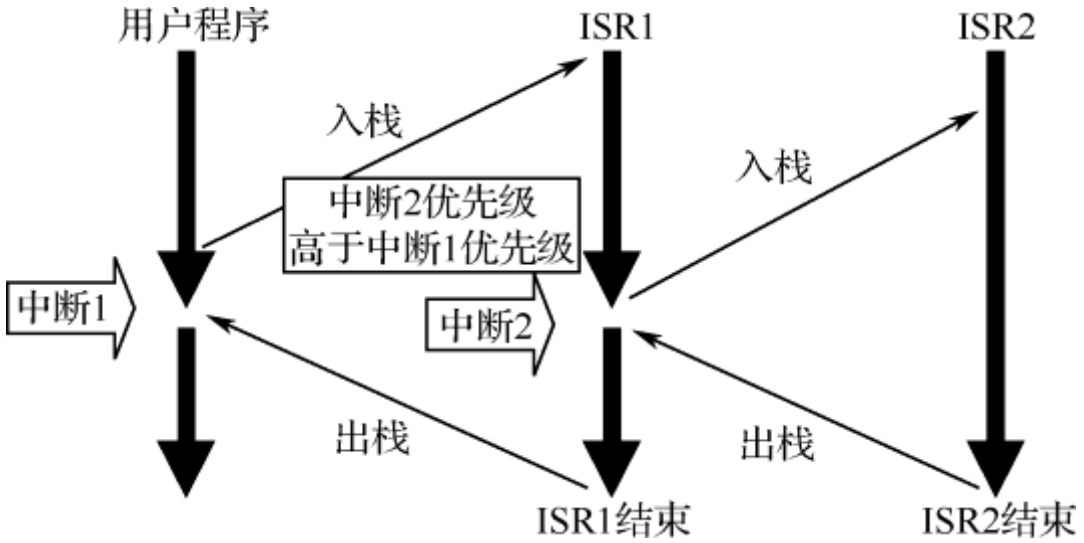


图2-17　Cortex-M4处理器中断抢占状态转换

2.4 异常和中断

4. 末尾连锁

末尾连锁是处理器用来加速中断响应的一种机制。在一个ISR正在执行时，如果有另一个中断请求，其优先级低于正在执行的ISR，那么这个新的中断会被挂起。当前执行的ISR结束后，会跳过出栈操作，直接将控制权让给新的ISR。

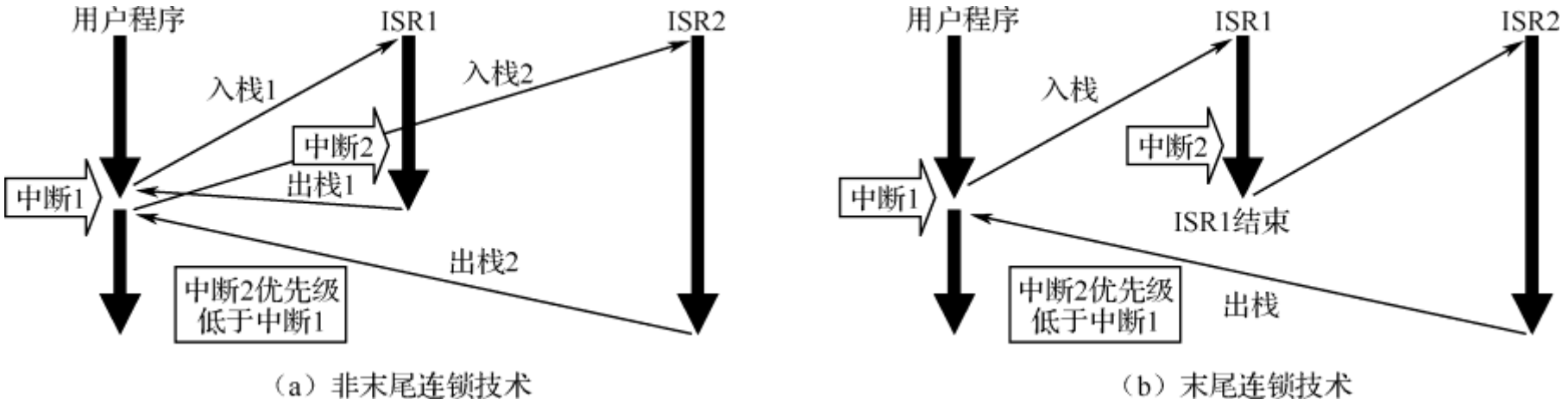


图2-18　非末尾连锁和末尾连锁技术之间的区别

2.4 异常和中断

5. 返回

在没有挂起异常或没有比被压栈的ISR优先级更高的挂起异常时，处理器执行出栈操作，并返回到被压栈的ISR或线程模式。

在响应ISR之后，处理器通过出栈操作自动将处理器状态恢复为进入ISR之前的状态。如果在状态恢复过程中出现一个新的中断，并且该中断的优先级比正在返回的ISR或线程更高，则处理器放弃状态恢复操作并将新的中断作为末尾连锁来处理。

2.4 异常和中断

2.4.4 复位

处理器复位后，它会从存储器中读取两个字:

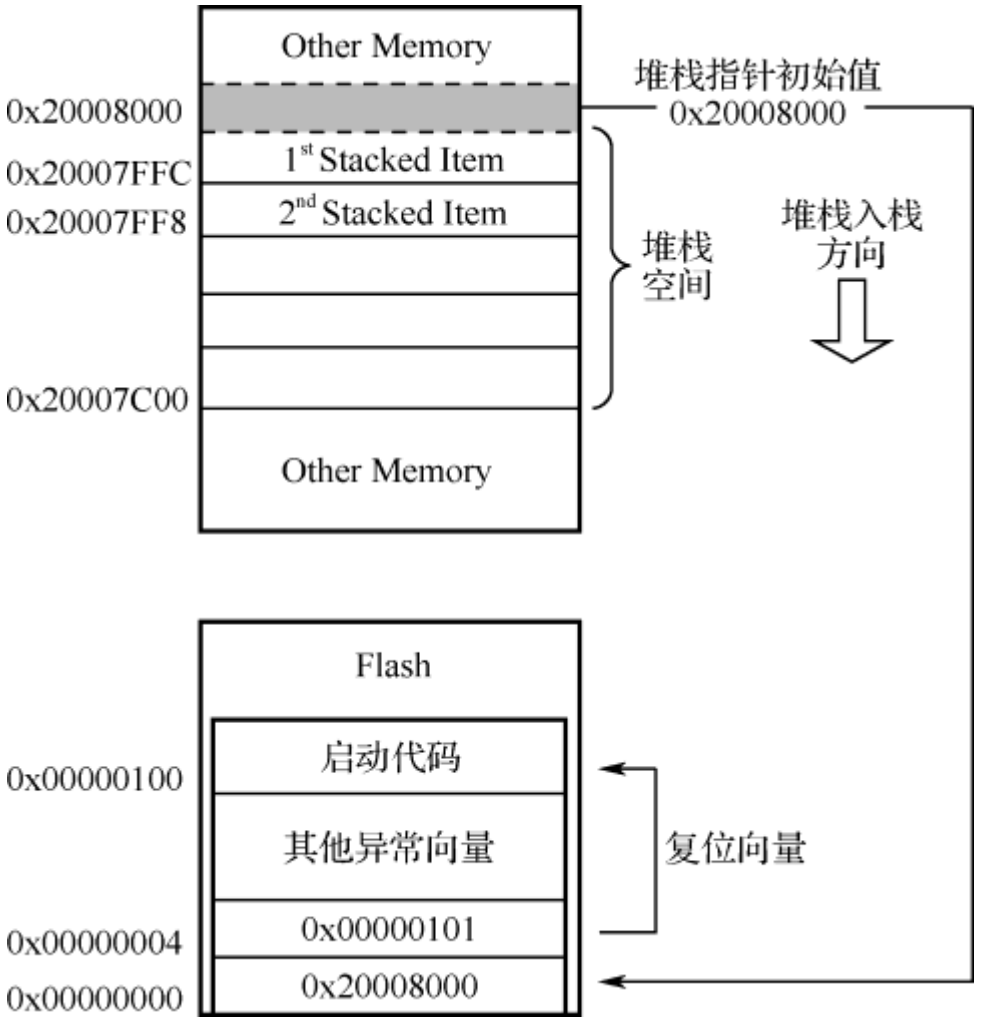
1.从地址 0x0000 0000 处取出MSP 的初始值。

2.从地址 0x0000 0004 处取出PC 的初始值——这个值是复位向量。然后从这个值所对应的地址处取指。

- **MSP 的初始值必须是堆栈内存的末地址加1。例：如果堆栈区域在0x20007C00-0x20007FFF之间，则MSP 的初始值就必须是0x20008000。**

2.4 异常和中断

初始MSP 及PC 初始化的一个范例



- MSP =堆栈内存末地址+1
- PC 初始化=启动代码的首地址+1（CM3在Thumb状态执行）