

第十一讲 Cortex-M3架构和指令系统

§ 1 ARM公司简介

§ 2 Cortex-M3内核

§ 3 指令系统



§ 1.1 ARM公司概况

- 成立于1990年，总部在英国剑桥，目前拥有员工2000多名，分布在全球的32个分支机构。全球有700多家ARM Connected Community成员企业一起支持ARM的技术，成为业界最大的一个联盟。
- ARM自2001年起在中国市场发展，大力支持中国企业、科研机构、大学教育的创新。国内已有400多所大学开设了ARM相关的课程和实验室，出版了120多本中文的ARM相关教科书。

§ 1.1 ARM公司概况

- ARM中国在上海、北京及深圳设有办事处,和国内70多家ARM Connected Community 成员企业一起支持ARM技术推广
- ARM 技术已在 95% 的智能手机、80% 的数码相机以及 35% 的所有电子设备中得到应用
- 向 250 多家公司出售了 800 个处理器许可证
- 目前为止已销售超过 200 亿个基于 ARM 的芯片

§ 1.1 ARM公司概况

- ARM中国在上海、北京及深圳设有办事处,和国内70多家ARM Connected Community 成员企业一起支持ARM技术推广。
- ARM 成为MCU 制作方法(IP)最大的提供商;
- 2016年7月, 软银宣布以320亿美元收购英国芯片设计公司ARM, 好像全世界都不太开心。

§ 1.2 ARM公司的主要产品

IP Products

Processors

Cortex-A

Cortex-R

Cortex-M

Classic Processors

Machine Learning

Graphics and Multimedia

Processors

Mali GPUs

Mali Video Processors

Mali Display Processors

Mali Camera

Assertive Display

Physical IP

Logic IP

Memory Compilers

Interface IP

POPIP

System Design Tools

System Design Kits

System Guidance

Subsystems

Socrates System Builder

Fast Models

Cycle Models

Development Boards

Fixed Virtual Platforms

System IP

CoreLink Interconnect

CoreLink System Controllers

CoreSight Debug and Trace

Memory Controllers

TrustZone Security IP

Free System IP Whitepapers

Software Tools

Graphics Development Tools

Solutions

Product Enquiries

Compilers

HPC

DS-5 Development Studio

Keil MDK

Debug Probes and Adapters



浙江大学

ZheJiang University


§ 1.2 ARM公司的主要产品

➤2017年3月，推出全新芯片架构DynamIQ，通过这项技术，AI的性能可提升50倍。

- 1) 在V8.3指令集中增加面向**人工智能的专用指令**；
- 2) 在单个集群（Cluster）中支持大小核或异构核；
- 3) 实现每个核独立的电压频率调节；
- 4) 可用于汽车无人驾驶系统。

§ 1.2 ARM公司的主要产品

➤ AI的3大部件



The image is a screenshot of the Arm Project Trillium webpage. At the top, the Arm logo is on the left, and navigation links for PRODUCTS, MARKETS, COMPANY, DEVELOP, NEWS, a search icon, and a user icon are on the right. The main heading is "Arm's Project Trillium". Below it, a paragraph describes the project: "Designed for unmatched versatility and scalability, Project Trillium is enabling a new era of ultra-efficient machine learning (ML) inference. Providing a massive efficiency uplift from CPUs, GPUs, DSPs and accelerators, Project Trillium completes the Arm Heterogenous ML compute platform with the Arm ML processor, the second-generation Arm object detection (OD) processor and open-source Arm NN software." Below this text are three blue rectangular boxes. The first box contains the text "arm ML PROCESSOR" and is labeled "Arm ML Processor" below it. The second box contains the text "arm OD PROCESSOR" and is labeled "Arm OD Processor" below it. The third box contains the text "arm NN" and is labeled "Arm NN" below it.

arm

PRODUCTS MARKETS COMPANY DEVELOP NEWS Q

Arm's Project Trillium

Designed for unmatched versatility and scalability, Project Trillium is enabling a new era of ultra-efficient machine learning (ML) inference. Providing a massive efficiency uplift from CPUs, GPUs, DSPs and accelerators, Project Trillium completes the Arm Heterogenous ML compute platform with the Arm ML processor, the second-generation Arm object detection (OD) processor and open-source Arm NN software.

arm
ML PROCESSOR

Arm ML Processor

arm
OD PROCESSOR

Arm OD Processor

arm
NN

Arm NN

§ 1.2 ARM公司的主要产品

1991年 推出第一款嵌入式 RISC 核，即 ARM6

1993年 推出 ARM7 核

1997年 发布 ARM9TDMI 系列

2004年 ARM 发布作为新型 Cortex 处理器内核系列中首款的 Cortex-M3 处理器

2003年 ARM 宣布已销售 10 亿多颗微处理器

2008年 ARM 宣布已销售 100 亿台处理器

2010年 ARM推出了高性能数字信号控制的Cortex-M4

2016年 Cortex-R8: 4个32位实时内核，针对5G手机市场

➤Cortex-A73: 64位，多核，性能更强的移动SoC



浙江大学

Zhejiang University

§ 1.3 ARM公司的商业模式

ARM公司的商业模式是**提供技术许可**的知识产权，而不是制造和销售实际的半导体芯片。

ARM的合作伙伴包括世界领先的半导体和系统公司。全球目前有200多家（包括最大的前20家）半导体公司已从ARM公司获得技术授权，成为ARM的客户。

模式好：避开与INTEL等直接竞争，化敌为友。

时机好：半导体厂商推MCU需要设计自己CPU核，开发成本、推广成本很高。



浙江大学

ZheJiang University

§ 1.4 ARM微处理器的特点

处理器分类：

CISC: Intel CPU

RISC: PowerPC、MIPS、ARM

ARM具有经典RISC的特点：

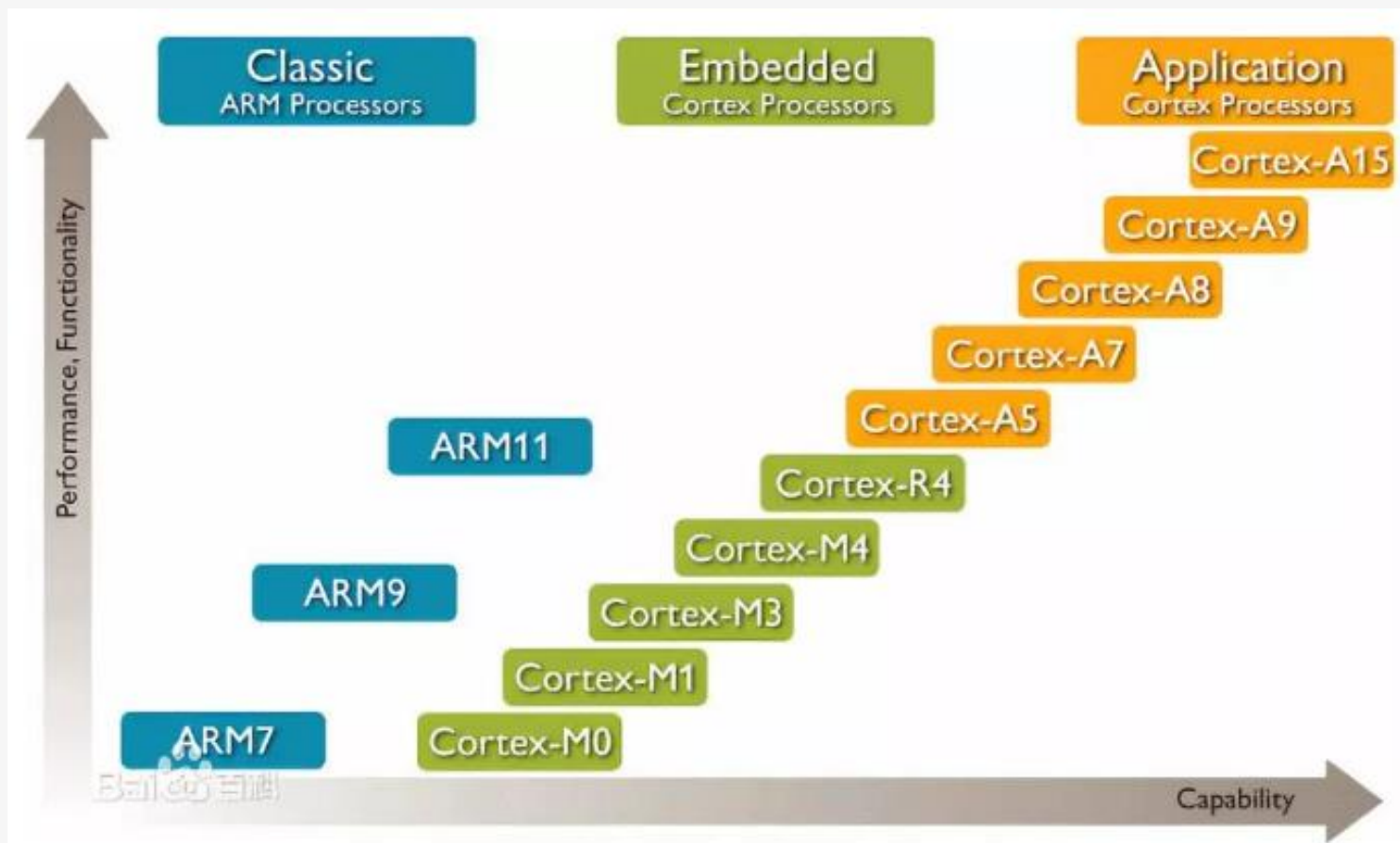
- 1) 大的、统一的寄存器文件
- 2) 装载/保存结构，数据处理操作只针对寄存器的内容，而不直接对存储器进行操作；
- 3) 简单的寻址模式；
- 4) 统一和固定长度的指令域，简化了指令的译码。



浙江大学

Zhejiang University

§ 1.5 ARM的常见种类



§ 1.5 ARM的常见种类

1) 经典处理器：ARM系列

ARM7: 70MHz, 成本低, 应用最广;

ARM9: 200-500MHz, 运算能力较强、应用多

ARM11: ARM9的性能提升, 应用较少

推出时间已超过15年, ARM7TDMI 仍是市场上销量最高的32位处理器。每个季度的设备销量超过10亿台, 或者每秒超过90台, 并且颁发的许可证超过500个。此类范围广泛的处理器占据目前市场上销售的所有电子产品的四分之一, 仍处于核心地位

2) Cortex-A系列：高性能



ARMv7-A
High performance
32-bit CPU with
enterprise class
feature set



ARMv7-A
High performance
32-bit CPU with
lower power and
smaller area



ARMv8-A
Highest performance
64/32-bit CPU

High
Performance



ARMv7-A
Smallest and
lowest power CPU



ARMv7-A
High efficiency
32-bit CPU
big.LITTLE™ compatible



ARMv8-A
High efficiency
64/32-bit CPU
big.LITTLE compatible

High
Efficiency

ARM®

§ 1.5 ARM的常见种类

2) Cortex-A系列：高性能

- Cortex-A15：移动应用，性能最高的解决方案
- Cortex-A7：800 MHz - 1.2 GHz 的典型频率
- Cortex-A9：800 MHz - 2 GHz 的标准频率，每个内核可提供 5000 DMIPS 的性能
- Cortex-A8：单核解决方案，可提供经济有效的高性能，在 600 MHz - 1 GHz 的频率下，性能超过 2000 DMIPS
- Cortex-A5：低成本实现，在 400- 800 MHz 的频率下，提供的性能超过 1200 DMIPS。

上述处理器都支持 ARM 的第二代多核技术

§ 1.5 ARM的常见种类

3) Cortex-R系列：实时处理器

Cortex-R4、Cortex-R5 和 Cortex-R7 处理器用于嵌入式实时产品（如汽车安全或无线基带）等

- 硬件除法器、浮点单元 (FPU) 选项
- 具有 Thumb-2 指令的 ARM v7-R 架构，高代码密度
- 指令集增强，包括 SIMD、DSP 和媒体处理
- 具有指令和数据高速缓存控制器的哈佛架构
- 用于获得快速响应代码和数据的处理器本地的紧密耦合内存 (TCM)
- 高性能 64 位 AMBA 3 AXI 总线接口
- 奇偶校验检测和 ECC，用于 1 级内存系统和总线的软错误和硬错误检测/更正等。

§ 1.5 ARM的常见种类

3) Cortex-M系列：全球微控制器标准

- Cortex-M4: 168MHz, 高性能, 硬件浮点
- Cortex-M3: 普及型, 高性价比, 取代ARM7
- Cortex-M0: 高性价比, 取代8、16位mcu
- Cortex-M0+: 超低功耗, 取代低功耗8、16位mcu
- 已许可给40个以上的ARM合作伙伴, 包括NXP、STMicroelectronics(M3, 性价比)、Freescale(M4, 模拟特性)、Texas Instruments等领先供应商。

Cortex-M3现状

- NXP:LPC17000
- TI->Luminary Micro:LM3S
- ST:STM32
- Atmel:SAM3U
- 此四大知名厂商采用Cortex-M3处理器的设计出来的芯片，主要在于片上存储器容量、集成外设、功能模块的区别。

第十讲 Cortex-M3架构和指令系统

§ 1 ARM公司简介

§ 2 Cortex-M3内核

§ 3 指令系统

§ 4 嵌入式编程



§ 2 Cortex-M3 内核简介

一、CM3 的内核组成

二、三级流水线

三、Thumb2 指令集

四、工作模式和访问特权

五、内部寄存器

六、中断机制

七、复位功能

八、存储器及其映射

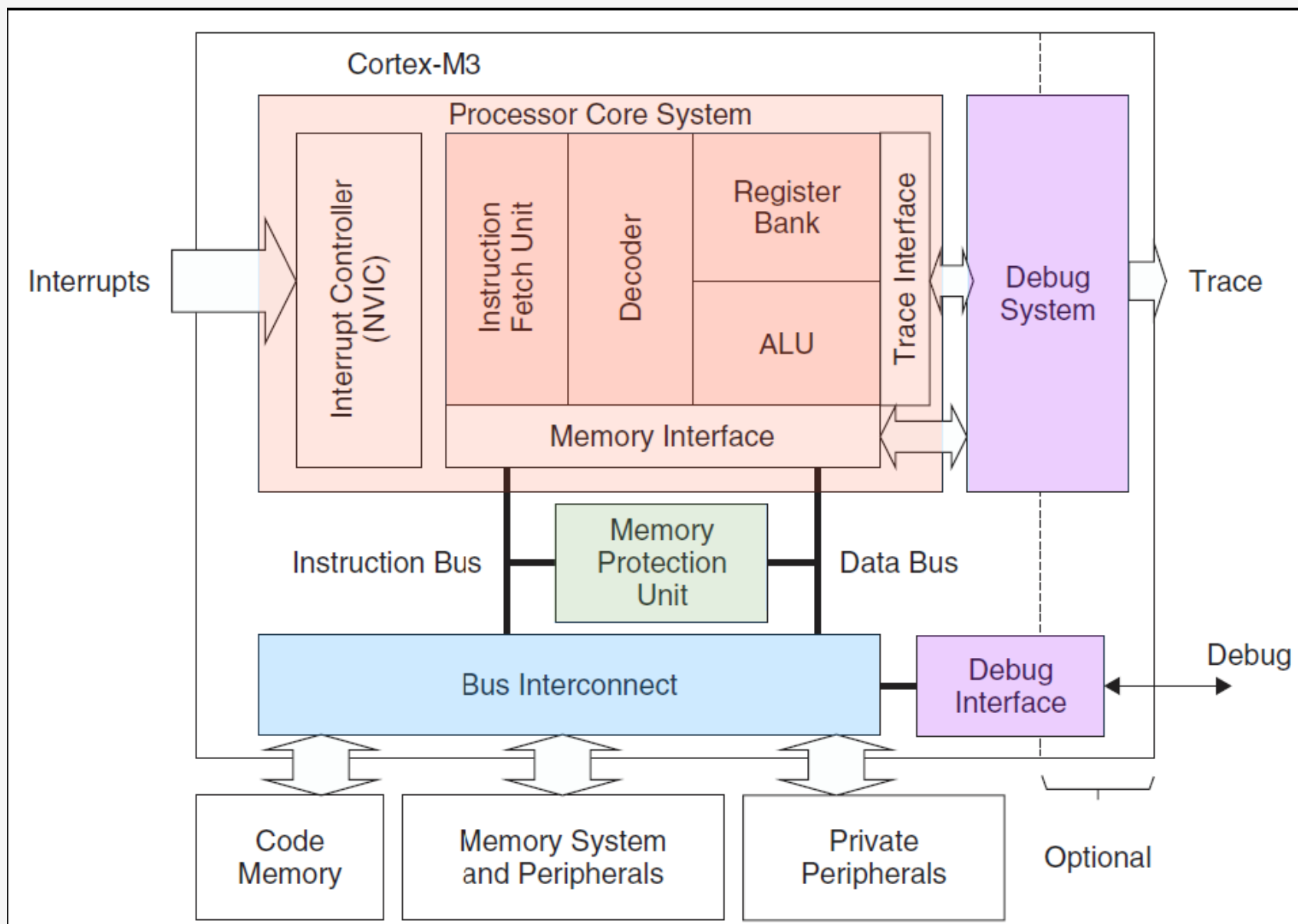
九、调试接口



§ 2 Cortex-M3内核简介

2.1 Cortex-M3组成 (ARM7的简化、改进版)

比较项目	ARM7	Cortex-M3
架构	ARMv4T (普林斯顿) 指令和数据总线共用, 会出现瓶颈	ARMv7-M (哈佛) 指令和数据总线分开, 无瓶颈
指令集	32位ARM指令+16位Thumb指令 两套指令之间需要进行状态切换	Thumb/Thumb-2指令集 16位和32位 指令可直接混写, 无需状态切换
流水线	3级流水线 若出现转移则需要刷新流水线, 损失惨重	3级流水线+分支预测 出现转移时流水线无需刷新, 几乎无损失
性能	0.95DMIPS/MHz (ARM模式)	1.25DMIPS/MHz
功耗	0.28mW/MHz	0.19mW/MHz
低功耗模式	无	内置睡眠模式
面积	0.62mm ² (仅内核)	0.86mm ² (内核+外设)
中断	普通中断IRQ和快速中断FIQ太少, 大量外设不得不复用中断	不可屏蔽中断NMI+1-240个物理中断 每个外设都可以独占一个中断, 效率高
中断延迟	24-42个时钟周期, 缓慢	12个时钟周期, 最快只需6个
中断压栈	软件手工压栈, 代码长且效率低	硬件自动压栈, 无需代码且效率高
存储器保护	无	8段存储器保护单元 (MPU)
内核寄存器	寄存器分为多组、结构复杂、占核面积多	寄存器不分组 (SP除外), 结构简单
工作模式	7种工作模式, 比较复杂	只有线程模式和处理模式两种, 简单
乘除法指令	多周期乘法指令, 无除法指令	单周期乘法指令, 2-12周期除法指令
位操作	无 访问外设寄存器需分“读-改-写”3步走	先进的Bit-band位操作技术, 可直接访问外设寄存器的 某个值
系统节拍定时	无	内置系统节拍定时器, 有利于操作系统移植



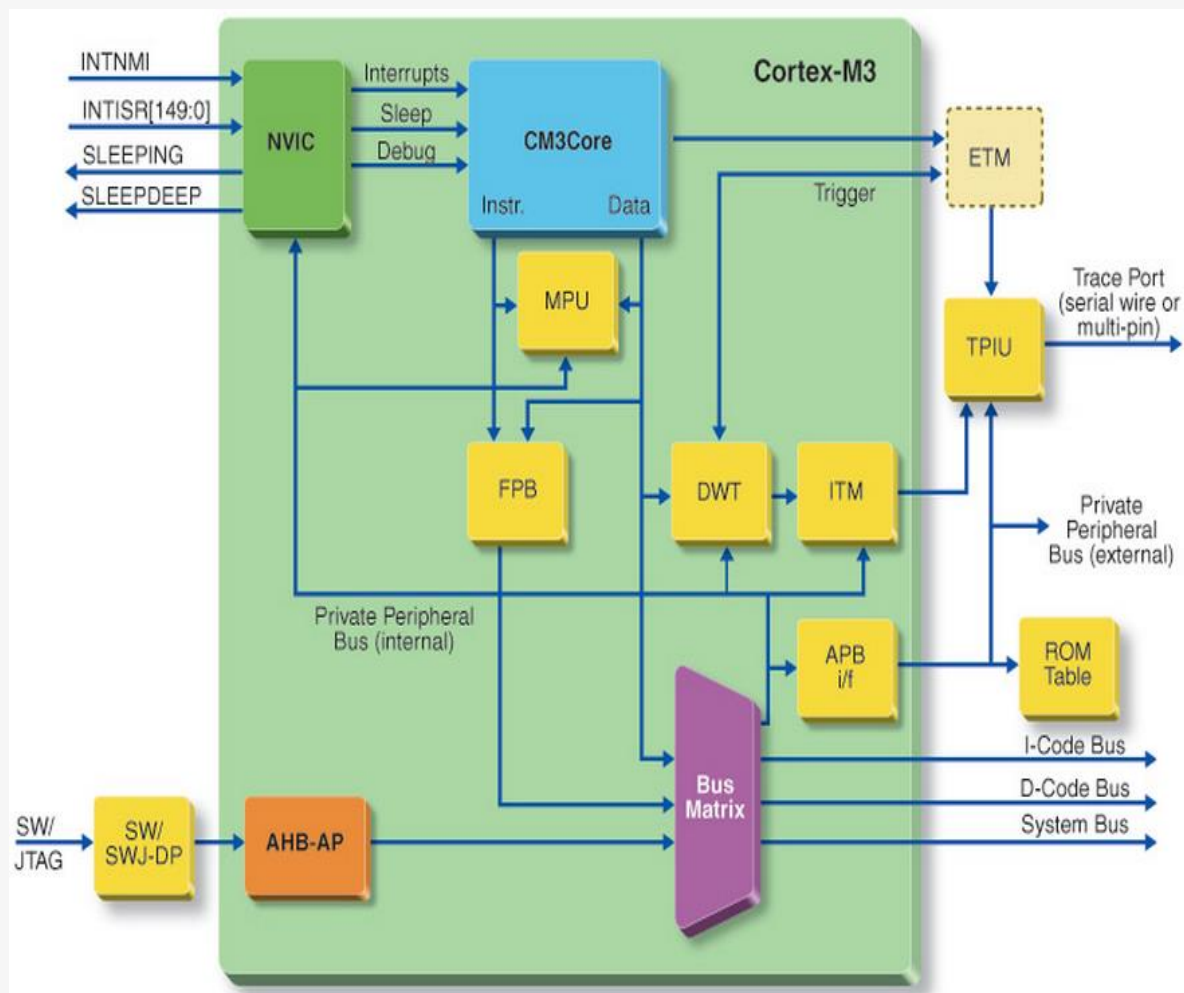
Cortex - M3 的一个简化视图

来源: STM32权威指南.pdf

Cortex—M3内核方框图

●Cortex-M3处理器主要包括：

1. 处理器内核
2. 与处理器内核紧密结合的嵌套向量中断控制器 (NVIC) 以实现低延迟的中断处理
3. 存储器保护单元 (MPU)，可选部件MPU实现存储器保护
4. 总线接口
5. 调试接口



§ 2 Cortex-M3 内核简介

一、CM3 的内核组成

二、三级流水线

三、Thumb2 指令集

四、工作模式和访问特权

五、内部寄存器

六、中断机制

七、复位功能

八、存储器及其映射

九、调试接口



§ 2.2 三级流水线

CM3处理器使用流水线+分支预测的操作模式，使几个操作同时进行，并使处理和存储器系统连续操作，当出现转移时流水线无需刷新，几乎无损失，增加了处理器指令流的速度。

CM3的流水线分3级，分别为：

取指→译码→执行

来源：zlgmcpu ppt

§ 2.2 三级流水线

正常操作过程中，在执行一条指令的同时对下一条指令进行译码，并将第三条指令从存储器中取出。这三条指令之间的位置关系如下表所示：

流水线上各指令的地址		流水线工位	描述
ARM指令集	Thumb指令集		
PC	PC	取指	指令从存储器中取出
PC-4	PC-2	译码	对指令使用的寄存器进行译码
PC-8	PC-4	执行	从寄存器组中读出寄存器，执行移位和ALU操作，寄存器被写回到寄存器组中

来源：zlgmcu ppt

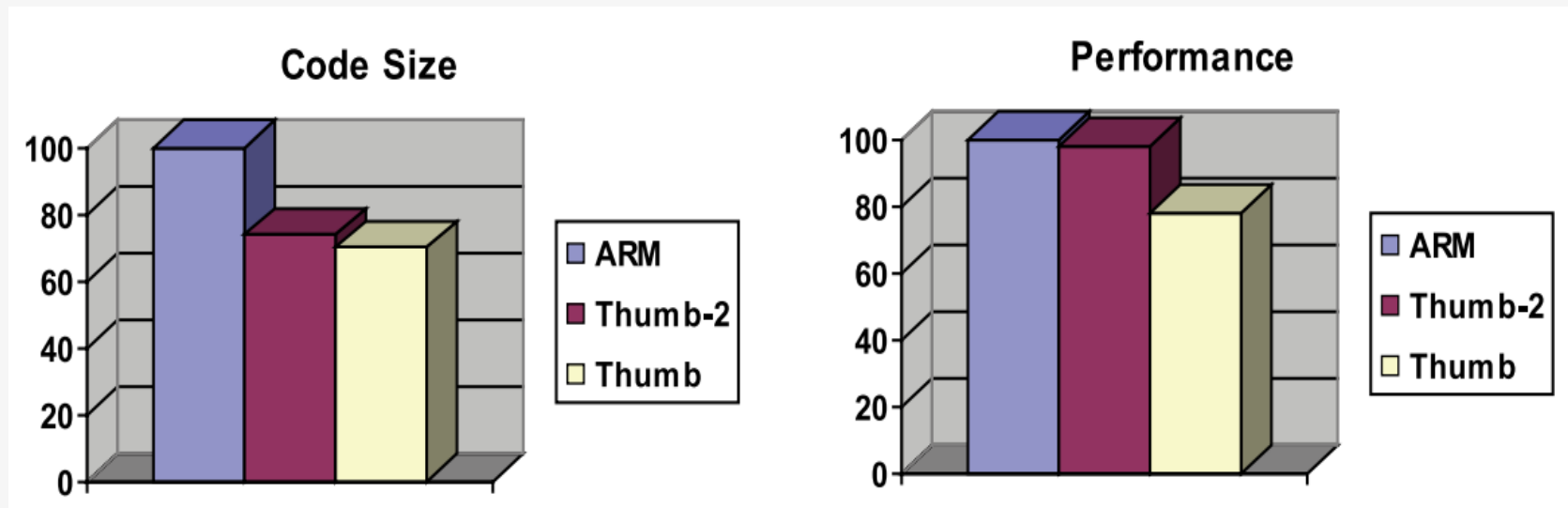
§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



Cortex-M3处理器指令集Thumb-2

- ARM Thumb-2技术的代码密度和代码性能



- 引自：ARM公司Richard Phelan撰写的《如何使用Thumb-2
- 改善代码密度和性能》

Cortex-M3处理器指令集Thumb-2

- Thumb-2技术可以带来很多好处：
 - 可以实现ARM指令的所有功能。
 - 增加了12条新指令，可以改进代码性能和代码密度之间的平衡。
 - 代码性能达到了纯ARM代码性能的98%。
 - 相对ARM代码，Thumb-2代码的大小仅有其74%
 - 代码密度比现有的Thumb指令集更高。
 - 代码大小平均降低5%。
 - 代码速度平均提高2-3%。

注：Cortex-M3不支持ARM指令集。

§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



§ 2.4 两种工作模式

1) 线程模式和处理模式

目的：引入处理模式 (handle mode) 和线程模式 (thead mode) 的本意，是用于区别普通应用程序（线程模式）和中断服务程序（处理模式）；

效果：有效的简化了 ARM7 体系结构支持 7 种处理器模式（用户模式、快中断模式、中断模式、管理模式、中止模式、未定义模式和系统模式）。

§ 2.4 两种工作模式

2) 特权级和用户级

目的：用于存储器访问的保护机制。使得普通的用户程序代码不能意外地，甚至是恶意地执行涉及到要害的操作；

特权级：该级别的程序可以访问所有范围的存储器（如果有MPU，还要在MPU规定的禁地之外），并且可以执行所有指令；

用户级：用户级程序不能直接改写CONTROL寄存器，需执行一条系统调用指令(SVC)，由异常服务例程修改CONTROL寄存器，才能在用户级的线程模式下重新进入特权级。



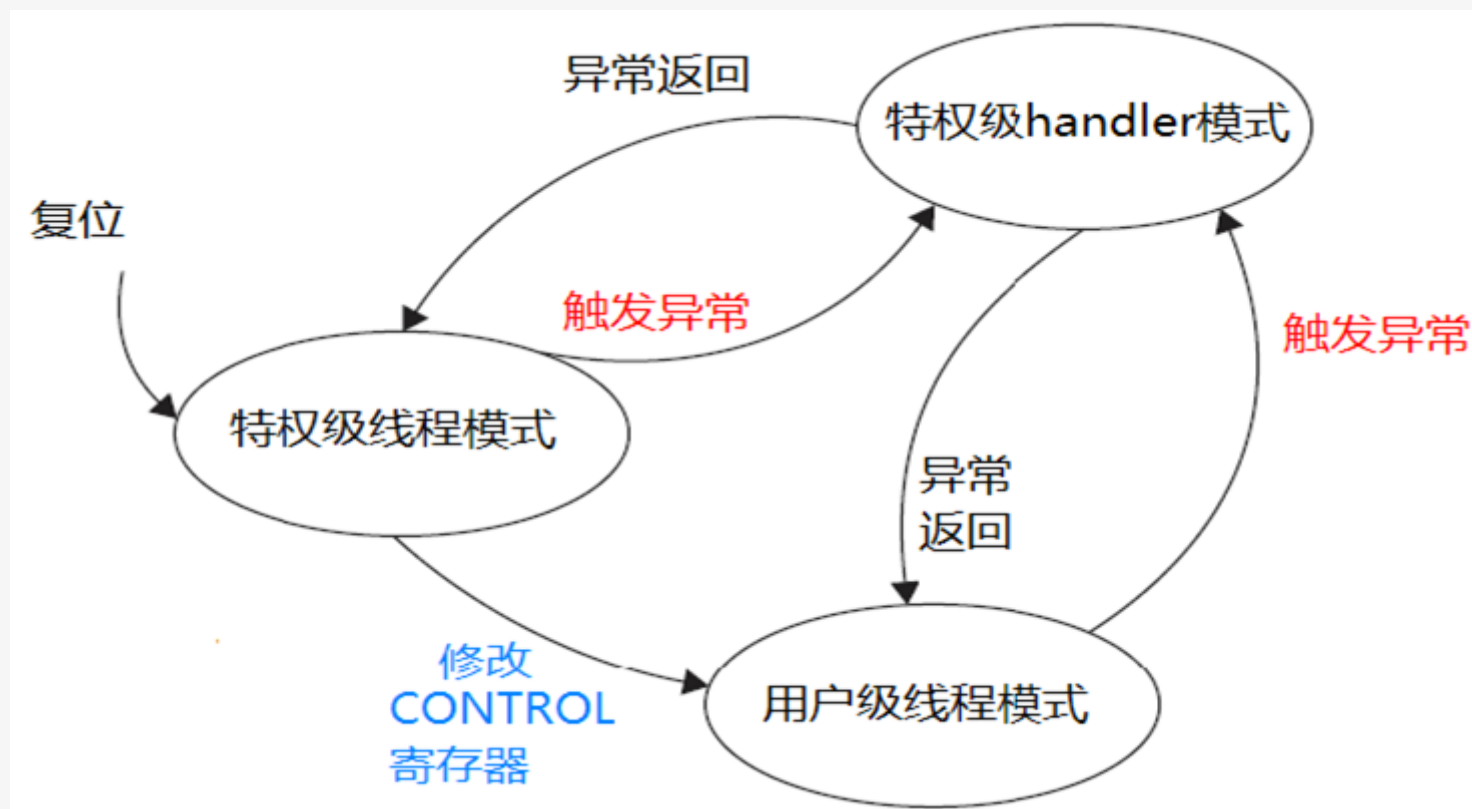
§ 2.4 两种工作模式

3) 默认状态

- 在CM3 运行主程序（后台程序）时是线程模式，既可以使用特权级，也可以使用用户级；
- 中断(异常)服务程序必须在处理模式（始终特权级）下执行；
- 复位后，mcu默认进入线程模式，特权极访问。

§ 2.4 两种工作模式

4) 模式转换



存储器保护单元MPU

- MPU是保护内存的一个组件。 支持标准的 ARMv7 (PMSA) “Protected Memory System Architecture ” 模型。
- **MPU**为以下操作提供完整的支持：
 - 保护区
 - 重叠保护区区域
 - 访问权限
 - 将存储器属性输出到系统
- **MPU**可以用于：
 - 强制执行特权原则
 - 分离程序
 - 强制执行访问原则

§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



§ 2.5 内部寄存器

1) 组成: CM3 处理器拥有R0 - R15 的寄存器组。其中R13 作为堆栈指针SP, SP 有两个, 但在同一时刻只能有一个可以使用。

R0	通用寄存器	Low Registers
R1	通用寄存器	
R2	通用寄存器	
R3	通用寄存器	
R4	通用寄存器	
R5	通用寄存器	
R6	通用寄存器	
R7	通用寄存器	
R8	通用寄存器	High Registers
R9	通用寄存器	
R10	通用寄存器	
R11	通用寄存器	
R12	通用寄存器	
R13 (MSP)	R13 (PSP)	主堆栈指针(MSP) , 进程堆栈指针 (PSP)
R14		连接寄存器(LR)
R15		程序计数器(PC)

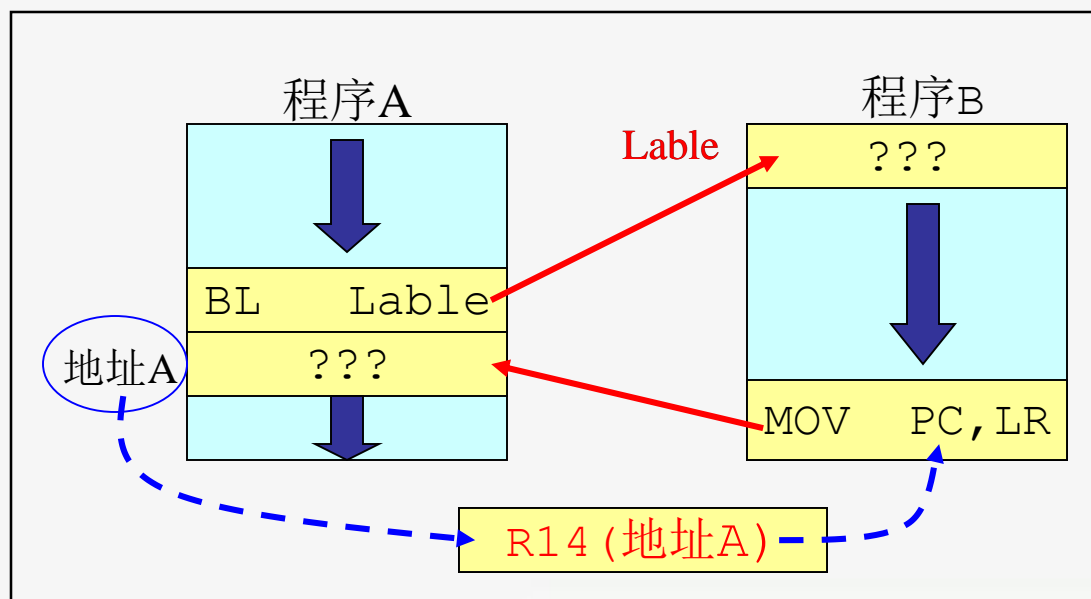
§ 2.5 内部寄存器

2) R14连接寄存器：用于存放子程序的返回地址调用

S1.程序A执行过程中调用程序B；

S2.程序跳转至标号Lable，执行程序B。同时硬件将“BL Lable”指令的下一条指令所在地址存入R14；

S3.程序B执行最后，将R14寄存器的内容放入PC，返回程序A；



来源：zlgmcu ppt

§ 2.5 内部寄存器

3) R15 程序指针：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

来源：zlgmcu ppt

§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



§ 2.6 中断机制

1) 重要性

嵌入式系统的实时性通过MCU的中断功能来实时实现，因此中断机制和中断部件使用非常重要

2) 中断流程

当预定义的事件发生时，正常程序被暂停，处理器进入异常模式。自动将处理器状态保存到堆栈中，执行中断服务程序（ISR），结束时自动从堆栈中恢复，继续执行被暂停的正常程序

3) 高效率

嵌套向量中断控制器（NVIC）支持末尾连锁（tail-chaining）中断技术，执行背对背中断（back-to-back interrupt），能够省略连续中断之间的状态保存和恢复指令

§ 2.6 中断机制

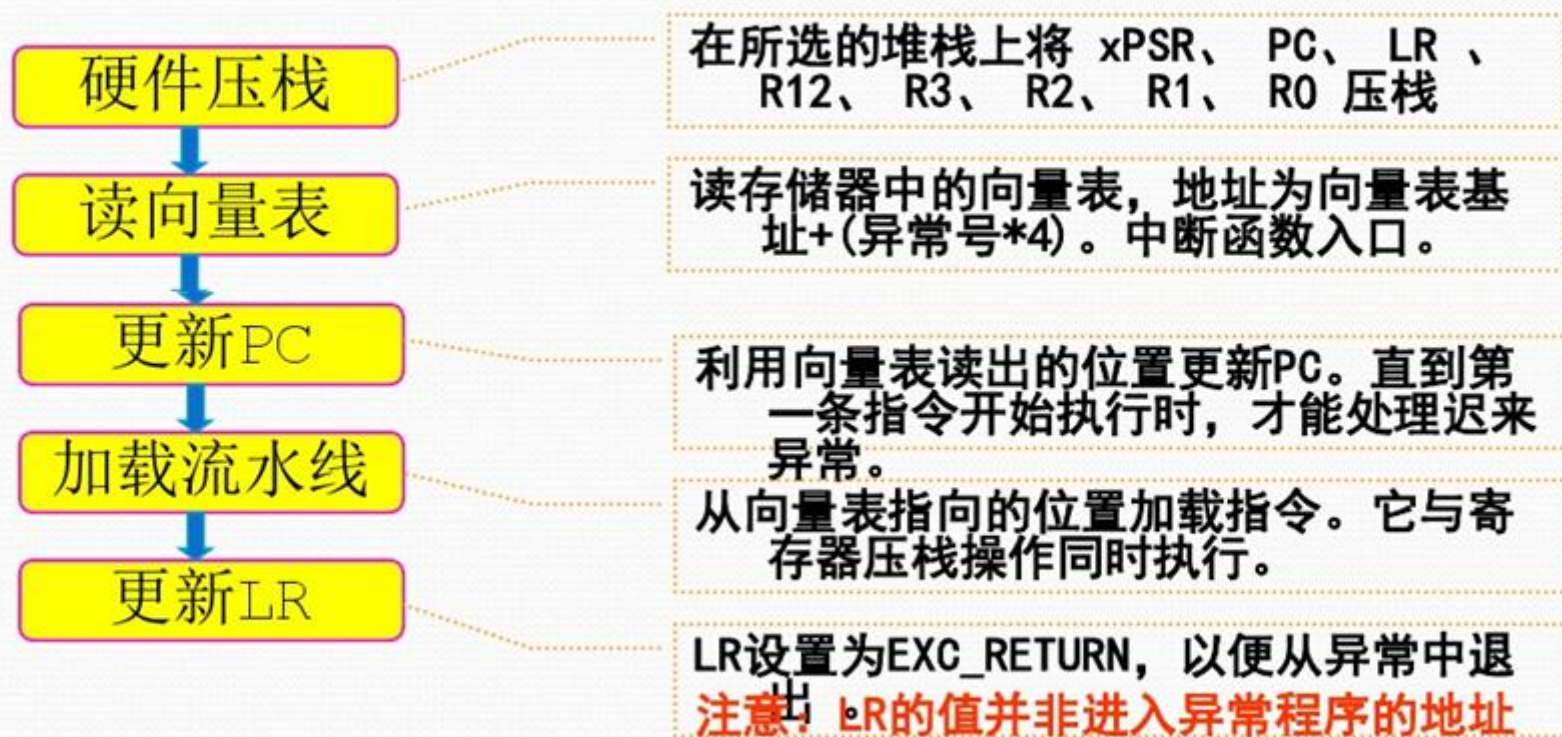
4) 其它特点

- 中断优先级可动态重新设置
- 中断数目可配置为1~240
- 中断优先级的数目可配置为1~8 位 (1~256 级)
- 处理模式和线程模式具有独立的堆栈和特权等级
- 使用C/C++标准的调用规范: *ARM 架构的过程调用标准 (PCSAA)* 执行ISR 控制传输。

异常类型	位置	优先级	描述
-	0	-	在复位时栈顶从向量表的第一个入口加载。
复位	1	-3 (最高)	在上电和热复位 (warm reset) 时调用。在第一条指令上优先级下降到最低 (线程模式)。异步的。
不可屏蔽的中断	2	-2	不能被除复位之外的任何异常停止或占先。异步的
硬故障	3	-1	由于优先级的原因或可配置的故障处理被禁止而导致不能将故障激活时的所有类型故障。同步的
存储器管理	4	可调整 ^a	MPU 不匹配, 包括违反访问规范以及不匹配。是同步的。即使 MPU 被禁止或不存在, 也可以用它来支持默认的存储器映射的 XN 区域。
总线故障	5	可调整 ^b	预取故障, 存储器访问故障, 以及其它相关的地址/存储故障。精确时是同步, 不精确时是异步。
使用故障	6	可调整	使用故障。例如, 执行未定义的指令或尝试不合法的状态转换。是同步的。
-	7-10	-	保留
SVCall	11	可调整	利用 SVC 指令调用系统服务。是同步的。
调试监控	12	可调整	调试监控, 在处理器没有停止时出现。是同步的, 但只有在使能时是有效的。如果它的优先级比当前有效的异常的优先级要低, 则不能被激活。
-	13	-	保留
PendSV	14	可调整	可挂起的系统服务请求。是异步的, 只能由软件来实现挂起。
SysTick	15	可调整	系统节拍定时器 (tick timer) 已启动。是异步的。
外部中断	16 及以上	可调整	在内核的外部产生。INTISR[239:0], 通过 NVIC (设置优先级) 输入, 都是异步的。

^a 该异常的优先级可修改, 见系统处理器优先级寄存器的位分配。可调整的范围为 NVIC 的 0~N 优先

3 中断、异常过程



注：以上步骤由硬件自动完成，仅需12个时钟周期 

§ 2.6 中断机制

● 优先级

通过对中断优先级寄存器的8位PRI_N区执行写操作，将中断的优先级指定为0~255，（0优先级最高，255优先级最低）

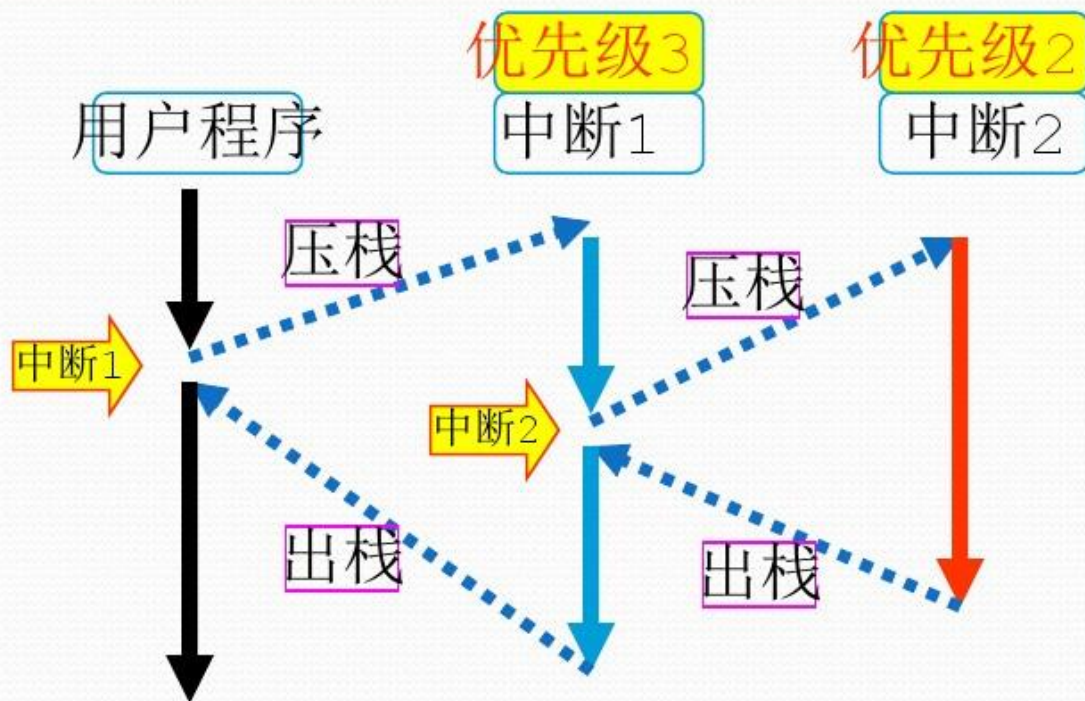
软件优先级的设置对复位，NMI，和硬故障无效。
它们的优先级始终比外部中断要高。

如果两个或更多的中断指定了相同的优先级，
则由它们的硬件优先级来决定处理器对它们进行处理时的顺序。

来源：zlgmcu ppt

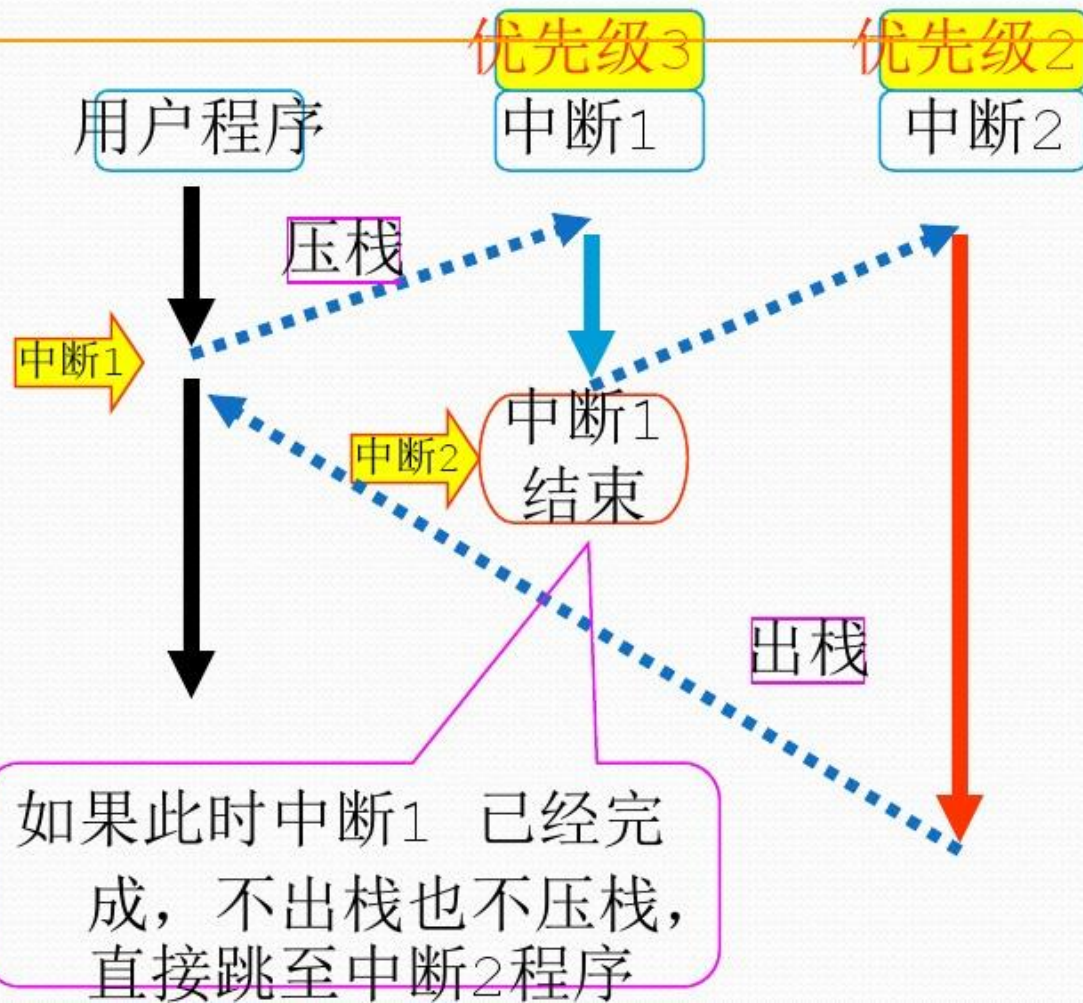
4 占先

在异常处理程序中，一个新的异常比当前的异常优先级更高，处理器打断当前的流程，响应优先级更高的异常，此时产生中断嵌套。

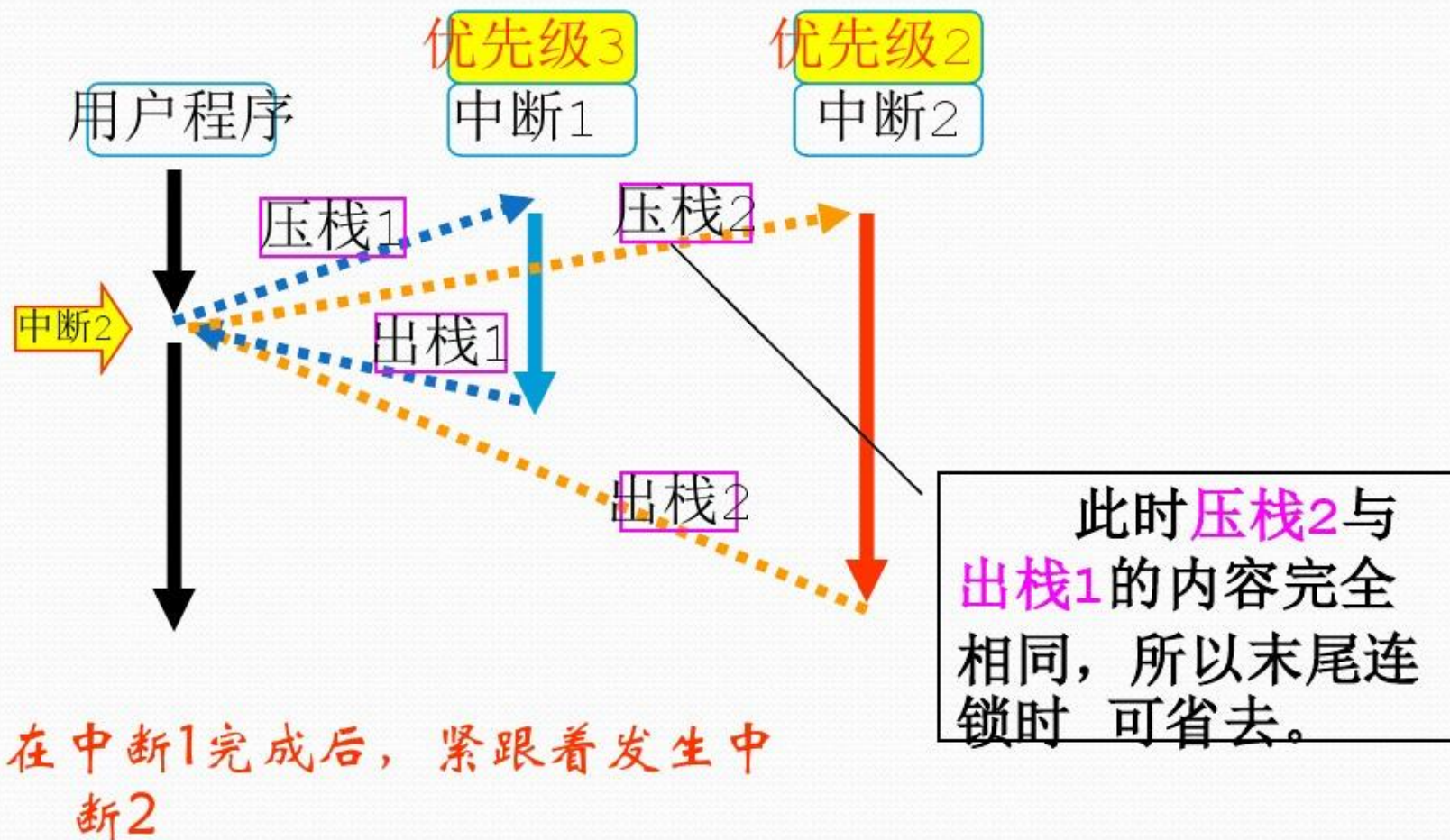


5 末尾连锁

末尾连锁是处理器用来加速中断响应的一种机制。在结束ISR时，如果存在一个挂起中断，其优先级高于正在返回的ISR或线程，那么就会跳过出栈操作，转而将控制权让给新的ISR。



不用末尾连锁的情况



§ 2.6 中断机制

● 2个独立的堆栈

进程堆栈（process stack），SP_process 为进程堆栈的SP 寄存器。线程模式在复位后使用主堆栈主堆栈，可以配置为使用进程堆栈。

主堆栈（main stack），SP_main 为主堆栈的SP 寄存器。处理模式使用主堆栈，在将8 个寄存器压栈之后，ISR 使用主堆栈，并且后面所有的抢占中断都使用主堆栈。

来源：zlgmcu ppt

§ 2.6 中断机制

● 压栈操作

当MCU进入中断时，它自动将下面的8个寄存器按以下顺序压栈：

PC

xPSR

r0~r3

r12

LR

● 出栈操作：顺序相反的操作



浙江大学

Zhejiang University

§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



§ 2.7 复位功能

CM3 处理器含3个复位输入:

系统复位、电源复位和后备域复位。

系统复位:

1. NRST引脚上的低电平(外部复位)
2. 窗口看门狗计数终止(WWDG复位)
3. 独立看门狗计数终止(IWDG复位)
4. 软件复位(SW复位)
5. 低功耗管理复位

电源复位:

1. 上电/掉电复位(POR/PDR复位)
2. 从待机模式中返回

后备域复位:

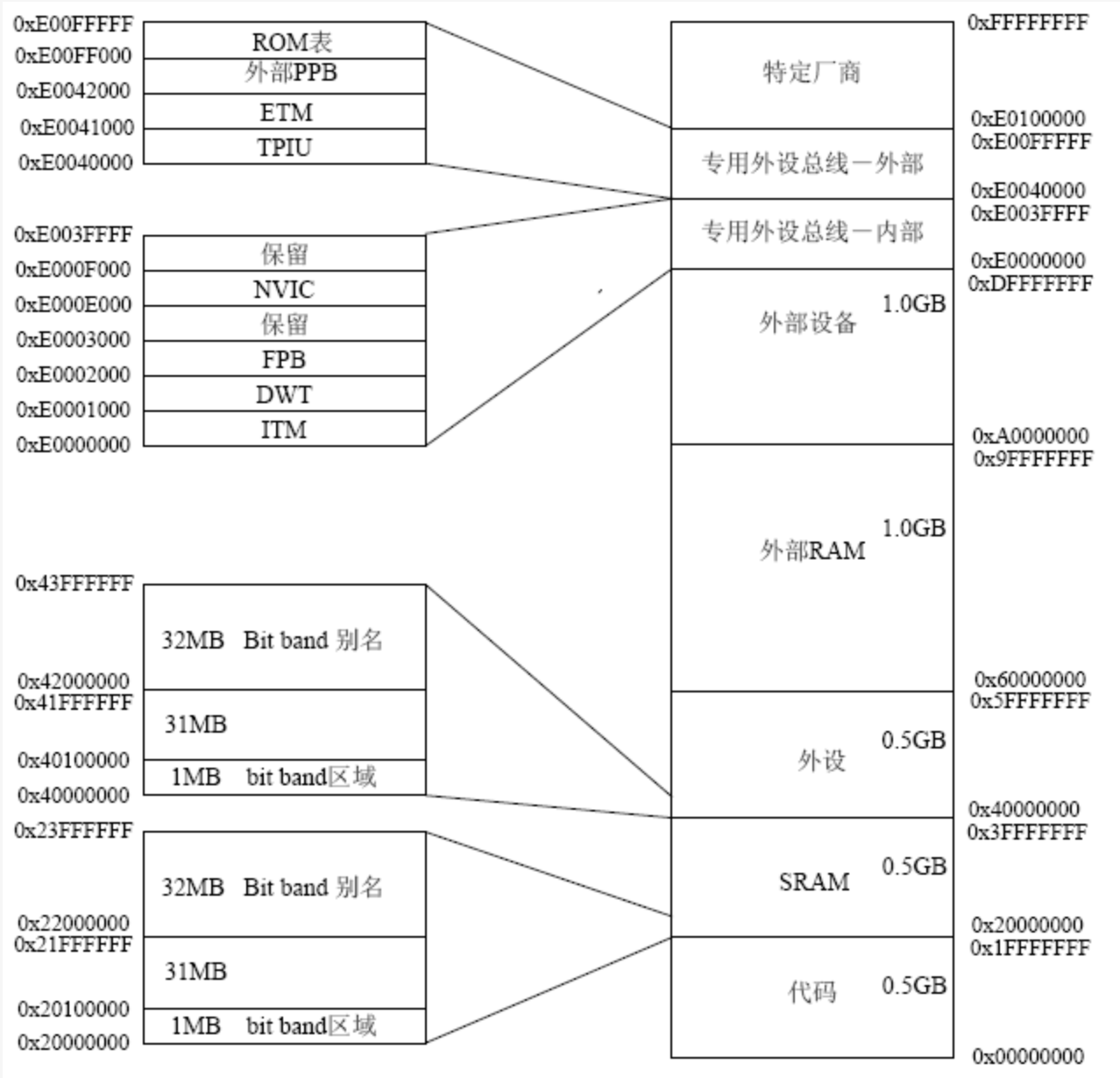
1. 软件复位, 备份区域复位可由设置备份域控制寄存器(RCC_BDCR)中的BDRST位产生。



§ 2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口





§ 2.8 存储器及其映射

1) 硬件直接支持的数据类型

- 字节：8位
- 半字：16位（必须分配为占用两个字节）
- 字：32位（必须分配为占用4各字节）

1			
1	2		
1	2	3	4

来源：zlgmcu ppt

§ 2.8 存储器及其映射

2) 存储器格式

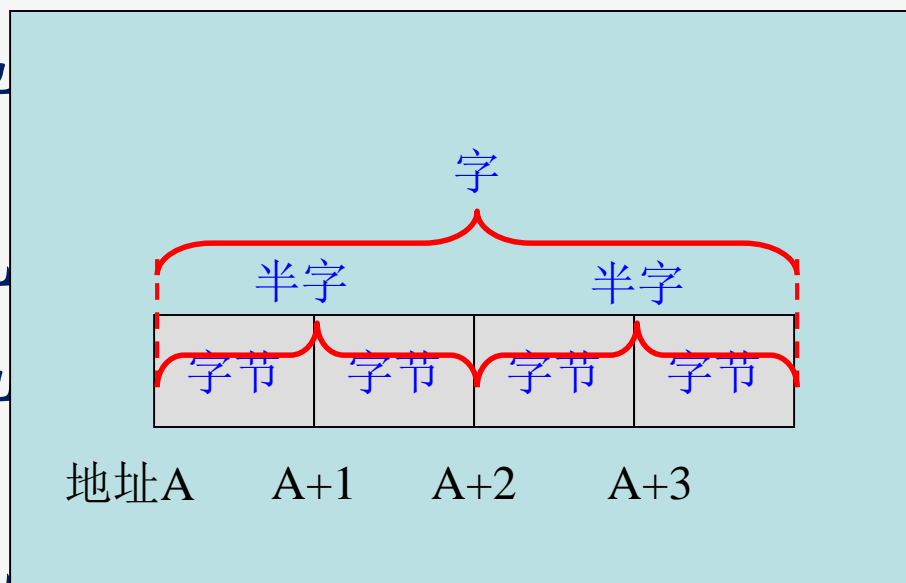
- 地址空间的规则：

- 位于地址A和A+3;

- 位于地址A+1和A+2;

- 位于地址A+2和A+3;

- 位于地址A+3和A+4;



A+1,A+2

和A+1;

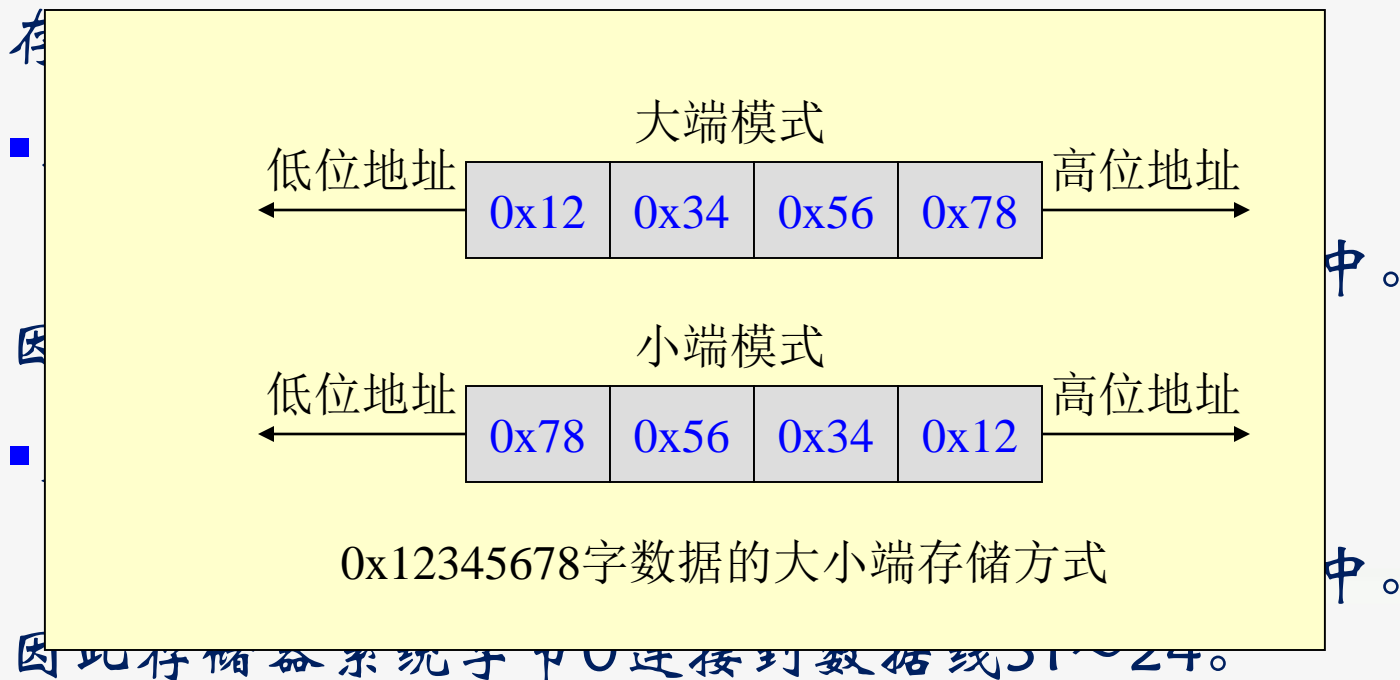
址A+2和

A+2;

来源: zlgmcu ppt

§ 2.8 存储器及其映射

3) 小端模式和大端模式



来源: zlgmcu ppt

§ 2.8 存储器及其映射

3) 小端模式和大端模式

一个基于ARM内核的芯片可以只支持大端模式或小端模式，也可以两者都支持。

在ARM指令集中不包含任何直接选择大小端的指令，但是一个同时支持大小端模式的ARM芯片可以通过硬件配置（一般使用芯片的引脚来配置）来匹配存储器系统所使用的规则。

注意：如果实际的存储器格式与芯片的存储器格式不符时，只有以字为单位的数据存取才正确，否则将出现不可预期的结果。

来源：zlgmcu ppt

§ 2.8 存储器及其映射

4) 未对齐的存储器访问

ARM结构通常希望所有的存储器访问都合理的对齐。具体来说就是字访问的地址通常是字对齐的，而半字访问使用的地址是半字对齐的。不按这种方式对齐的存储器访问称为非对齐的存储器访问。

- 将一个非字（半字）对齐的地址写入ARM (Thumb) 状态的R15寄存器，将引起非对齐的指令取指。
- 在一个非字（半字）对齐的地址读写一个字（半字），将引起非对齐的数据访问。

来源：zlgmcu ppt

§ 2.2 Cortex-M3 内核简介

- 一、CM3 的内核组成
- 二、三级流水线
- 三、Thumb2 指令集
- 四、工作模式和访问特权
- 五、内部寄存器
- 六、中断机制
- 七、复位功能
- 八、存储器及其映射
- 九、调试接口



§ 2.9 调试接口

CM3处理器的调试接口有2种：

1) JTAG: 6线制接口

2) SWD: 2线制接口

§ 2 Cortex-M3 内核简介

CM3 内核资料参见:

- Cortex-M3 Technical Reference Manual.pdf (英文, ARM)
- Cortex-M3 技术参考手册.pdf(中文, zlgmcu 译)



第十讲 Cortex-M3架构和指令系统

§ 1 ARM公司简介

§ 2 Cortex-M3内核

§ 3 指令系统

§ 4 嵌入式编程



§ 3 指令系统

一、Thumb2指令集

二、9种寻址方式

三、6大类指令

四、4种伪指令



§ 3.1 Thumb-2指令集

- 1) CM3处理器指令的特点：是加载/存储型的，指令集只能处理寄存器的数据，而且处理结果都要放回寄存器，而对数据的访问要通过专门的加载/存储指令来完成；
- 2) CM3 处理器采用ARM v7-M 架构。它包括所有的16位thumb指令集和基本的32位thumb-2指令集架构，不能执行ARM指令；
- 3) Thumb-2在thumb指令集架构（ISA）上进行了大量的改进，它与thumb相比，代码密度更高，并且通过使用16/32位指令，提供更高的性能。

§ 3 指令系统

一、Thumb2指令集

二、9种寻址方式

三、6大类指令

四、4种伪指令



§ 3.2 9 种寻址方式

一、寻址方式：

根据指令中地址码字段来寻找实际操作数地址的方式；

二、ARM处理器的9种寻址方式：

- ① 寄存器寻址
- ② 立刻寻址
- ③ 寄存器偏移寻址
- ④ 寄存器间接寻址
- ⑤ 基址寻址
- ⑥ 多寄存器寻址
- ⑦ 堆栈寻址
- ⑧ 块拷贝寻址
- ⑨ 相对寻址



§ 3.2 9 种寻址方式

① 寄存器寻址

操作数在寄存器中，指令中的地址码字段指出的是寄存器编号，指令执行时直接取出寄存器中的数值进行操作，执行速度快。

例如：

MOV R1,R2 ;R1=R2

SUB R0,R1,R2 ;R0=R1-R2

§ 3.2 9 种寻址方式

② 立刻寻址

指令中的地址码字段是操作数，即数据包含在指令中，取出指令时也就取出可以立刻使用的操作数。例如：

MOV R1,#0xff000 ;R1=0xff000

SUBS R0,R0,#1 ;R0=R0-1

§ 3.2 9 种寻址方式

③ 寄存器偏移寻址

ARM指令集特有的寻址方式。

第2个寄存器操作数在与第1个操作数结合之前，先进行移位操作。例如：

MOV R0,R2, LSL,#3 ;R2左移3位
;R0=R2

ANDS R1,R1,R2,LSL R3 ;R2左移R位
;R1=R1 AND R2

§ 3.2 9 种寻址方式

④ 寄存器间接寻址

寄存器间接指令中的地址码字段指出的是寄存器编号，目标操作数在寄存器指定地址的存储单元中，即寄存器为操作数的地址指针。例如：

LDR R1,[R2] ;R1=[R2]

SUB R1,R1,[R2] ;R1=R1-[R2]

§ 3.2 9 种寻址方式

⑤ 基址寻址

寄存器的内容与指令中给出的偏移量相加，形成操作数的有效地址。基址寻址用于访问基址附近的存储单元，常用于查表、数组操作、功能部件的寄存器访问等。例如：

LDR R2,[R3,#0x0c]

STR R1,[R0,#-4] ;R0=R0-4,[R0]=R1

§ 3.2 9 种寻址方式

⑥ 多寄存器寻址

多寄存器寻址允许一条指令中16寄存器中的部分或全部寄存器。例如：

LDMIA R1!,{R2-R7,R12}

;将R1指向的单元中的数据读出到R2~R7, R12中, R1自动加1。

STMIA R0!,{R2-R7,R12}

;将R2~R7, R12中的数据保存到R1所指的单元中, R1自动加1。

§ 3.2 9 种寻址方式

⑦ 堆栈寻址

堆栈寻址采用SP寄存器作为指向存储器的指针，有几种形式：

LDMFA, STMFA ;满递增

LDMEA, STMEA ;空递增

LDMFD, STMFD ;满递减

LDMED, SP!, {R1-R7, LR} ;空递减

;数据出栈，放入R1-R7, LR中

STMED SP!, {R1-R7, LR}

;将R1-R7, LR的值压入堆栈

§ 3.2 9种寻址方式

⑧ 块拷贝寻址

多寄存器数据传送指令，将一块数据在寄存器组和数据存储器之间传送。下列指令的功能时将R1-R7的数据保存到存储器中，但指针变化方式不同。存储器指针的变化

STMIA R0!,{R1-R7} ;在数据保存后增加

STMIB R0!,{R1-R7} ;在数据保存前增加

STMDA R0!,{R1-R7} ;在数据保存后减小

STMDB R0!,{R1-R7} ;在数据保存前减小

§ 3.2 9 种寻址方式

⑨ 相对寻址

由程序计数器PC提供基地址,指令中的地址码段为偏移量,形成操作数的有效地址。

BL SUBR1 ;调用SUBR1子程序

BEQ LOOP ;相等则跳转到LOOP

LOOP MOV R6,#1

...

SUBR1 ...

RET

§ 3 指令系统

一、Thumb2指令集

二、9种寻址方式

三、6大类指令

四、4种伪指令



§ 3.3 6大类指令

- ① 跳转指令
- ② 数据处理指令
- ③ 加载/存储指令
- ④ 协处理指令
- ⑤ 程序状态寄存器访问指令
- ⑥ 异常产生指令

§ 3.3 6大类指令

① 跳转指令

直接向程序计数器PC写入跳转地址值

MOV LR,PC ;PC=LR

以下指令的前后的跳转范围为32MB:

B Lable

BNE Lable

BL Lable ; 跳转之前把当前地址保存到
; LR(R14), 用于子程序调用

BLX Lable ; 从ARM指令跳转到Thumb
; 指令, 保存PC到LR

BX Lable ; 跳转到ARM或Thumb指令

§ 3.3 6大类指令

② 数据处理指令

➤ 数据传送指令

MOV R1,R0 ; R1=R0

MOVS R1,R0 ; S表示更新CPSR寄存器中的
; 条件标志位

➤ 算术逻辑运算指令

ADD、ADC、ADCS、SUB、SBC、SUBS
RSB、ORR、EOR、MUL、MLA、SMULL、
SMLAL、UMULL、UMLAL

➤ 比较指令

CMP 比较、CMN 取反比较、TST、TEQ

§ 3.3 6大类指令

③ 加载/存储指令

用于寄存器与存储器之间的数据传送。

LDR/LDRB/LDRH,STR/STRB/STRH

B:字节操作指令 H:半字操作指令

LDM/STM(批量)

IA:每次传送后地址加1

IB:每次传送前地址加1

DA:每次传送后地址减1

DB:每次传送前地址减1

FD:满递减堆栈

ED:空递减堆栈

FA:满递增堆栈

EA:空递增堆栈

§ 3.3 6大类指令

④ 协处理器指令

ARM处理器可支持最多16个协处理器，ARM协处理器指令有5条：

CDP: 通知ARM协处理器执行特定的操作，若协处理器未能完成，则产生未定义指令异常

LDC: 协处理器数据加载指令

STC: 协处理器数据存储指令

MCR: ARM寄存器到协处理器寄存器的数据传送指令

MRC: 协处理器寄存器到ARM处理器数据传送指令

§ 3.3 6大类指令

⑤ 程序状态寄存器访问指令

程序状态寄存器是指CPSR或SPSR。

MRS: 程序状态寄存器到通用寄存器的
数据传送指令。

MRS R0,CPSR ; R0=CPSR

MRS R0,SPSR ; R0=SPSR

MSR: 通用寄存器到程序状态寄存器的
数据传送指令。

MSR CPSR,R0 ; CPSR=R0

MSR SPSR,R0 ; SPSR=R0

§ 3.3 6大类指令

⑥ 异常产生指令

SWI: 软件中断指令

BKPT: 软件断点中断指令,用于程序调试

BKPT 16位立即数

§ 3 指令系统

一、Thumb2指令集

二、9种寻址方式

三、6大类指令

四、4种伪指令



§ 3.4 四种伪指令

一、伪指令

是编译器约定的指令，仅在汇编过程中起作用；不是CPU的指令，没有对应的操作码；

二、ARM的伪指令集共有4类

- ① 符号定义伪指令
- ② 数据定义伪指令
- ③ 汇编控制伪指令
- ④ 其它常用伪指令

§ 3.4 四种伪指令

① 符号伪指令

➤ **GBLA/GBLL/GBLS** ;定义全局变量

A:数字变量,默认值0; L:逻辑变量,
默认值F; S:字符串变量,默认值""

➤ **LCLA/LCLL/LCLS** ;定义局部变量

➤ **SETA/SETL/SETS** ;变量赋值

LCLA TEST3 ;默认值0

TEST3 SETA 0X55;TEST3=55H

➤ **RLIST**: 定义寄存器的列表名称

Reglist RLIST {R0-R5,R8,R10}

将列表名称定义为Reglist,在ARM指令

LDM/STM中通过Reglist访问寄存器列表

§ 3.4 四种伪指令

② 数据定义伪指令

为特定的数据分配存储单元，完成存储单元初始化

➤ DC*: 定义数据

DCB/DCW/DCWU/DCD/DCDU

DCFS/DCFSU/DCQ/DCQU U: 非严格对齐

➤ SPACE: 分配一个连续的内存单元

DS SPACE 100 ; 分配100个字节的连续单元

➤ MAP: 定义一个结构化的内存表首地址

➤ FIELD: 定义一个结构化的内存表的数据域

MAP 0X100 ; 定义结构化内存表首地址的值100H

A FIELD 16 ; A的长度16字节，位置0X100

B FIELD 32 ; B的长度为32字节，位置0X110

§ 3.4 四种伪指令

③ 汇编控制伪指令

用于控制汇编程序的执行流程。

➤ IF、ELSE、ENDIF; 可以嵌套使

➤ WHILE 逻辑表达式
(指令序列)

WEND

➤ MACRO、MEND: 宏指令, 可以嵌套使用

\$ MACRO Var1,...VarN

(指令序列)

MEND ; 用标号\$代替指令序列

MEXIT ; 用于从宏定义中跳转出去

§ 3.4 四种伪指令

④ 其它常用的伪指令

- AREA : 定义数据段、代码段
- ALIGN : 对齐方式
- ENTRY : 指定汇编程序入口
- END : 汇编程序结束
- EQU : 定义一个标号的值
- INPORT: 通知编译器后面的标号在其它源文件中
- EXPORT: 定义全局标号,供其它程序使用
- EXTERN: 标号在其它源文件中,不用就不引用
- INCLUDE(GET): 源文件要包含的文件
- INCBIN: 源文件要包含的数据文件,目标文件

致谢

谢谢！

