

嵌入式系统设计导论

——基于32位微处理器与实时操作系统

第三讲 ARM嵌入式微处理器体系结构

北京航空航天大学
机器人研究所

魏洪兴

本节提要

1

嵌入式微处理器概述

2

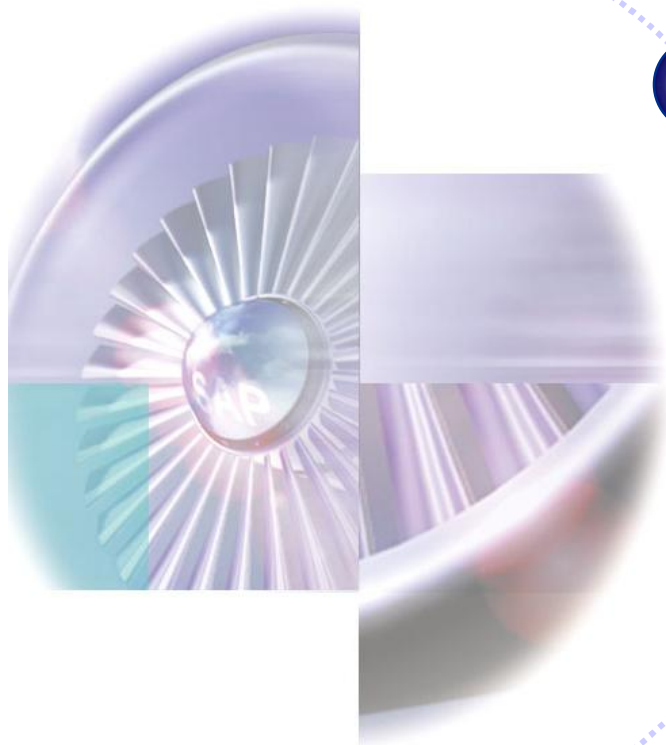
ARM体系结构概览

3

ARM编程模型

4

ARM 异常处理



嵌入式处理器概述

- 1 嵌入式微处理器是嵌入式系统的核心。目前**32位嵌入式微处理器**是市场的主流。
- 1 **32位嵌入式微处理器市场**，我们可以发现超过**100家**的芯片供应商和近**30种**指令体系结构。
- 1 在**1996年**以前，最成功的嵌入式微处理器是**Motorola公司**的**68000**系列。此外嵌入式微处理器市场还包括其它体系结构，如**Intel公司**的**I960**，**Motorola公司**的**Coldfire**，**Sun公司**的**Sparc**，以及嵌入式**X86**系列平台。
- 1 当然，最引人注目的还是**ARM公司**的**ARM**系列、**MIPS公司**的**MIPS**系列，以及**Hitachi公司**的**SuperH**系列（其中**ARM**和**MIPS**都知识产权公司，把他们的微处理器IP技术授权给半导体厂商，由他们生产形态各异的微处理器芯片）。

嵌入式处理器评价指标（1）

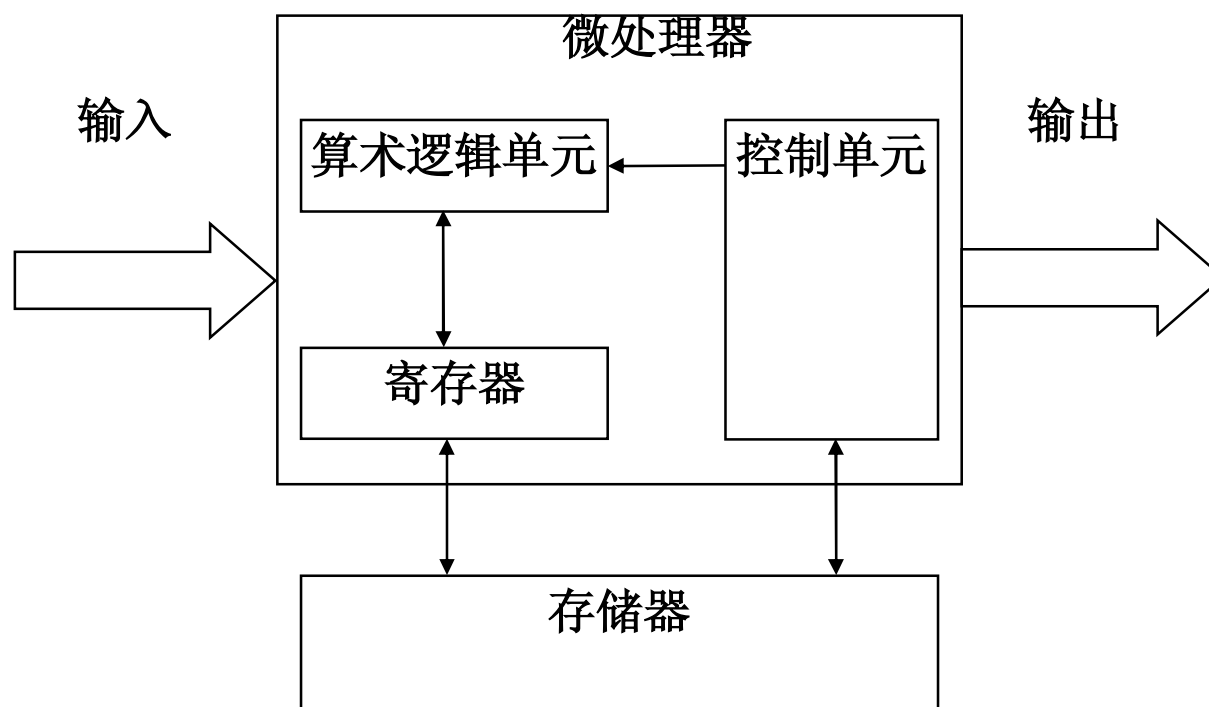
- I 功耗。一般的嵌入式微处理器都有三种运行模式：运行模式（**operational**）；待机模式（**standby or power down**）；停机模式（**and clock-off**）。功耗的评测指标是**MIPS/W**；
- I 代码存储密度。传统的**CISC**指令集计算机具有较好的代码存储密度。而**RICS**指令集计算机由于要求指令编码长度固定，虽然可以简化和加速指令译码过程，但为了实现与**CISC**指令集计算机相同的作业，往往需要更多的指令来完成，从而增加了代码长度。如**Hitachi**的**SuperH**体系结构采用了定长的**16**位指令，对每条指令按**16**位的格式存储。**ARM**则采用**16**位扩展的**Thumb**指令集，片内的逻辑译码器将其等价为**32**位的**ARM**指令而实时解码。而**MIPS**则采用**MIPS16**方法来解决这个问题。（影响代码密度的另外一个主要因素是所采用的**C**编译器。**ANSI C**是当前嵌入式领域的标准编程语言，随着嵌入式微处理器性能的提高，面向对象的语言也将被采用并会逐渐成为主流，一些编译器供应商已经开始着手解决代码密度问题。）

嵌入式处理器评价指标（2）

- I 集成度。嵌入式微处理器一般都为专用市场设计的，需要较高的集成度。但把所有的外围设备都集成到一个芯片上也不是一种好的解决方案。这是因为高集成度使芯片变得复杂，芯片引脚变密，增加了系统设计和测试的复杂性。因此，集成外围设备时必须要考虑简化系统设计，并缩短整个系统的开发周期。
- I 多媒体加速。为实现多媒体加速功能，嵌入式微处理器的设计者在传统的微处理器指令集的基础上增加**JPEG**和**MPEG**解压缩的离散余弦变换指令。还有一些半导体厂商针对智能手机和移动通讯市场的需求，将**RISC**微处理器和**DSP**集成在一个芯片上，如**TI**的**OMAP**。

嵌入式处理器的基本结构（1）

- 微处理器是整个系统的核心，通常由3大部分组成：控制单元、算术逻辑单元和寄存器。



嵌入式处理器的基本结构（2）

- I 控制单元：主要负责取指、译码和取操作数等基本动作，并发送主要的控制指令。控制单元中包括两个重要的寄存器：程序计数器（PC）和指令寄存器（IR）。程序计数器用于记录下一条程序指令在内存中的位置，以便控制单元能到正确的内存位置取指；指令寄存器负责存放被控制单元所取的指令，通过译码，产生必要的控制信号送到算术逻辑单元进行相关的数据处理工作。
- I 算术逻辑单元：算术逻辑单元分为两部分，一部分是算术运算单元，主要处理数值型的数据，进行数学运算，如加、减、乘、除或数值的比较；另一部分是逻辑运算单元，主要处理逻辑运算工作，如AND、OR、XOR或NOT等运算。
- I 寄存器：用于存储暂时性的数据。主要是从存储器中所得到的数据（这些数据被送到算术逻辑单元中进行处理）和算术逻辑单元中处理好的数据（再进行算术逻辑运行或存入到存储器中）。

评估嵌入式系统处理器的主要指标

要先明确预期最终应用程序在待选平台上的运行情况 and 测试目的，然后再挑选符合要求的特定测试向量。

- | MIPS测试基准。测试方法是计算在单位时间内各类指令的平均执行条数，单位：MIPS。
- | Dhrystone。测试基准是一个简单的C语言程序。EEMBC验证实验室研究指出，Dhrystone不适于作为嵌入式系统的测试向量。虽然它是市面上最普遍适用的测试向量，但它有许多漏洞。
- | EEMBC。基于每秒钟算法执行的次数和编译代码大小的统计结果。
- | 一次详尽的分析需要仔细衡量的因素包括：性能分析、功耗和效率分析、开发工具支持以及价格

本节提要

- 
- 1 嵌入式微处理器概述
 - 2 **ARM体系结构概览**
 - 3 ARM编程模型
 - 4 ARM 异常处理

ARM简介

ARM——Advanced RISC Machines

ARM——32位RISC结构IP核提供商



ARM Ltd

- 丨 成立于1990年11月
 - 丨 前身为 Acorn计算机公司
 - 丨 Advance RISC Machine(ARM)
- 丨 主要设计ARM系列RISC处理器内核
- 丨 授权ARM内核给生产和销售半导体的合作伙伴
 - 丨 ARM 公司不生产芯片
 - 丨 IP(Intelligence Property)
- 丨 另外也提供基于ARM架构的开发设计技术
 - 丨 软件工具, 评估板, 调试工具, 应用软件,
 - 丨 总线架构, 外围设备单元, 等等



ARM微处理器的应用领域

- | 工业控制领域
- | 无线通讯领域
- | 网络应用
- | 消费电子产品
- | 成像和安全产品

ARM微处理器系列

- | **ARM7系列**
- | **ARM9系列**
- | **ARM9E系列**
- | **ARM10E系列**
- | **ARM11系列**
- | **SecurCore系列**
- | **Cortex系列**
- | **Inter的StrongARM和Xscale系列**

ARM的发展历程-1

- I ARM公司成立于1981年，最初与英国广播公司合作为英国教育界设计小型机，当时采用的是美国的6502芯片。取得成功后，他们开始设计自己的芯片，受当时美国加州大学伯克利分校提出的RISC思想的影响，他们设计的芯片也采用RISC体系结构，并命名为“Acorn RISC Machine”。
- I ARM公司的第一款芯片ARM1在1985年被设计出来，次年又设计了真正实用的ARM2。ARM2具有32位数据总线和24位地址总线，带有16个寄存器。ARM2可能是当时最简化的32位微处理器，上面仅有30000个晶体管（4年前Motorola公司的68000则有68000个晶体管）。这种精简的结构使ARM2具有优异的低功耗特性，而性能则超过了同期Intel公司的286（134K个晶体管）。
- I 1990年ARM公司另外组建了一个名为“Advanced RISC Machines”的公司，专门从事ARM系列微处理器的开发。1998年ARM公司在伦敦证券交易所和NASDAQ上市。

ARM的发展历程-2

- I ARM7TDMI 是ARM公司最成功的微处理器IP之一，至今在蜂窝电话领域已销售了数亿个微处理器。
- I DEC公司获得ARM公司授权设计并生产了StrongARM系列微处理器，这款CPU的主频达到了233MHz，而功率不到1瓦。后来DEC公司StrongARM部门被Intel公司并购，Intel公司用StrongARM取代了他们境况不佳的*i860*和 *i960*体系，并在此基础上开发了新的体系结构XScale系列。
- I 世界各大半导体生产商从ARM公司购买其设计的ARM微处理器核，根据各自不同的应用领域，加入适当的外围电路，从而形成自己的ARM微处理器芯片进入市场。目前，Motorola、IBM、TI、Philips、VLSI、Atmel和Samsung等几十家大的半导体公司都获得了ARM公司的授权，生产形态各异的ARM芯片

ARM处理器的技术优势

- 1 低能耗：当初刚刚起步的嵌入式应用对运算性能并不苛求，但对芯片的功耗却相当敏感。而相对同时期的其他解决方案，ARM架构的能效比优势非常明显。
- 1 应用方案非常灵活：由于ARM公司只是提供了一个高效精简的核心，各半导体厂商可根据自身需求进行应用设计，架构灵活简便、扩展力很强。如厂商可为多媒体信号处理加入相关的指令集，或为Java相关的应用加入高效执行单元，或增加3D图形协处理器等等。
- 1 得到大量的软件支持：包括Windows CE、Symbian和Palm OS在内的手持设备三种主要操作系统系统都是基于ARM架构所设计。目前，ARM已经牢牢占领手机、PDA以及其他的掌上电子产品市场，这些领域都非常注重软件兼容和设计延续性，ARM在这些领域会继续保持优势。

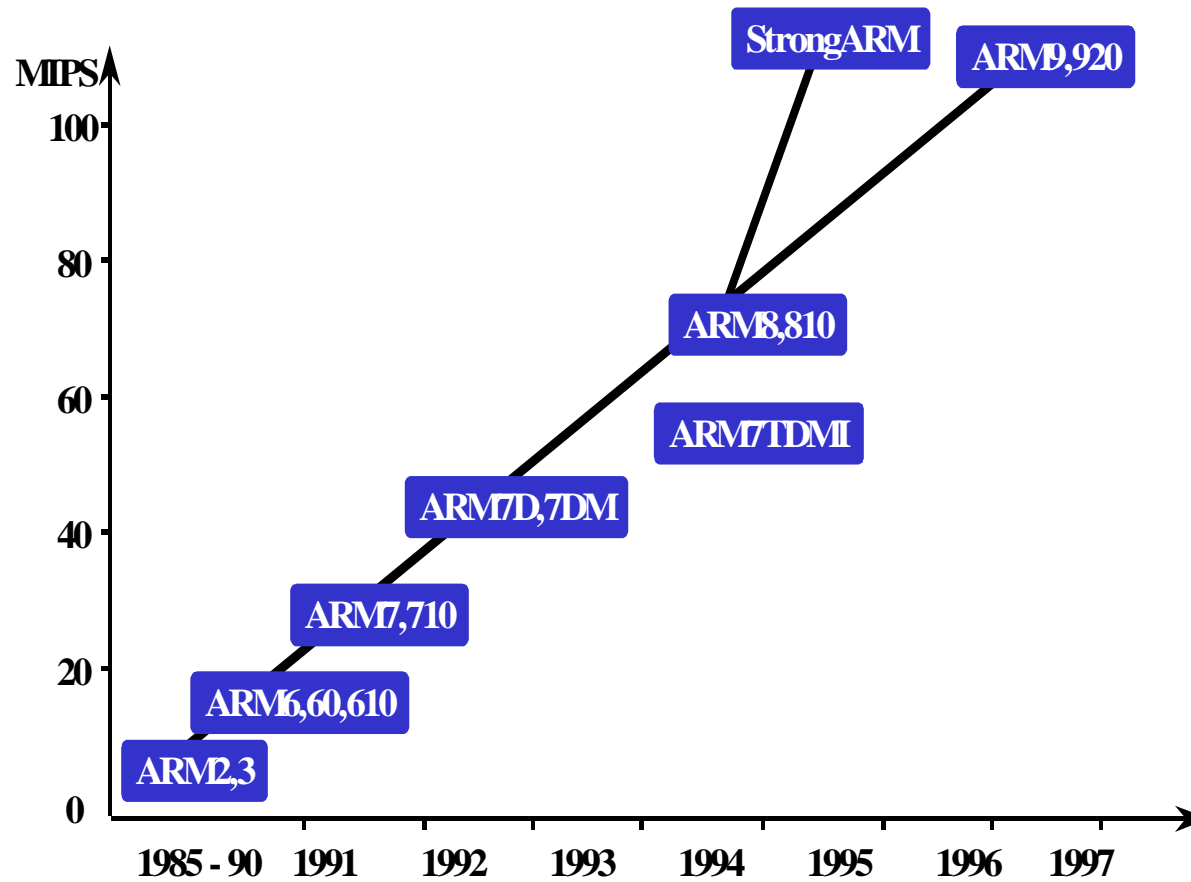
ARM处理器的应用

- | 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的发展历程-2

- I 1991 - ARM 推出第一款RISC嵌入式微处理器核 ARM6
- I 1993 - ARM 推出 ARM7 核
- I 1995 - ARM的 Thumb扩展指令集结构为16位系统增加了32位的性能,提供业界领先的代码密度

ARM的发展历程-3



ARM处理器的特点

ARM处理器的3大特点如下:

- | 小体积、低功耗、成本低、高性能;
- | 16位/32位双指令集;
- | 全球众多的合作伙伴。

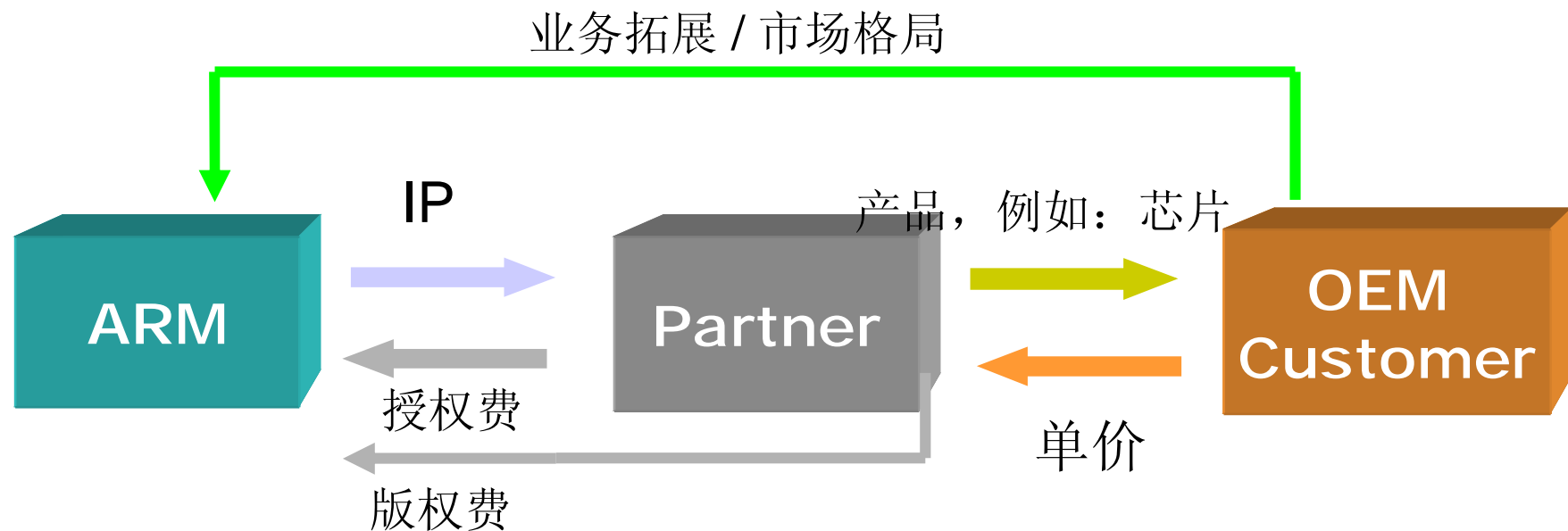
当前ARM体系结构的扩充包括:

- | Thumb: 16位指令集, 用以改善代码密度;
- | DSP: 用于DSP应用的算术运算指令集;
- | Jazeller: 允许直接执行Java代码的扩充。

ARM处理器系列提供的解决方案包括:

- | 在无线、消费电子和图像应用方面的开放平台;
- | 存储、自动化、工业和网络应用的嵌入式实时系统;
- | 智能卡和SIM卡的安全应用。

ARM的业务模型



ARM 创造和
设计IP

Partner把ARM IP
和其他 IP 集成进
产品

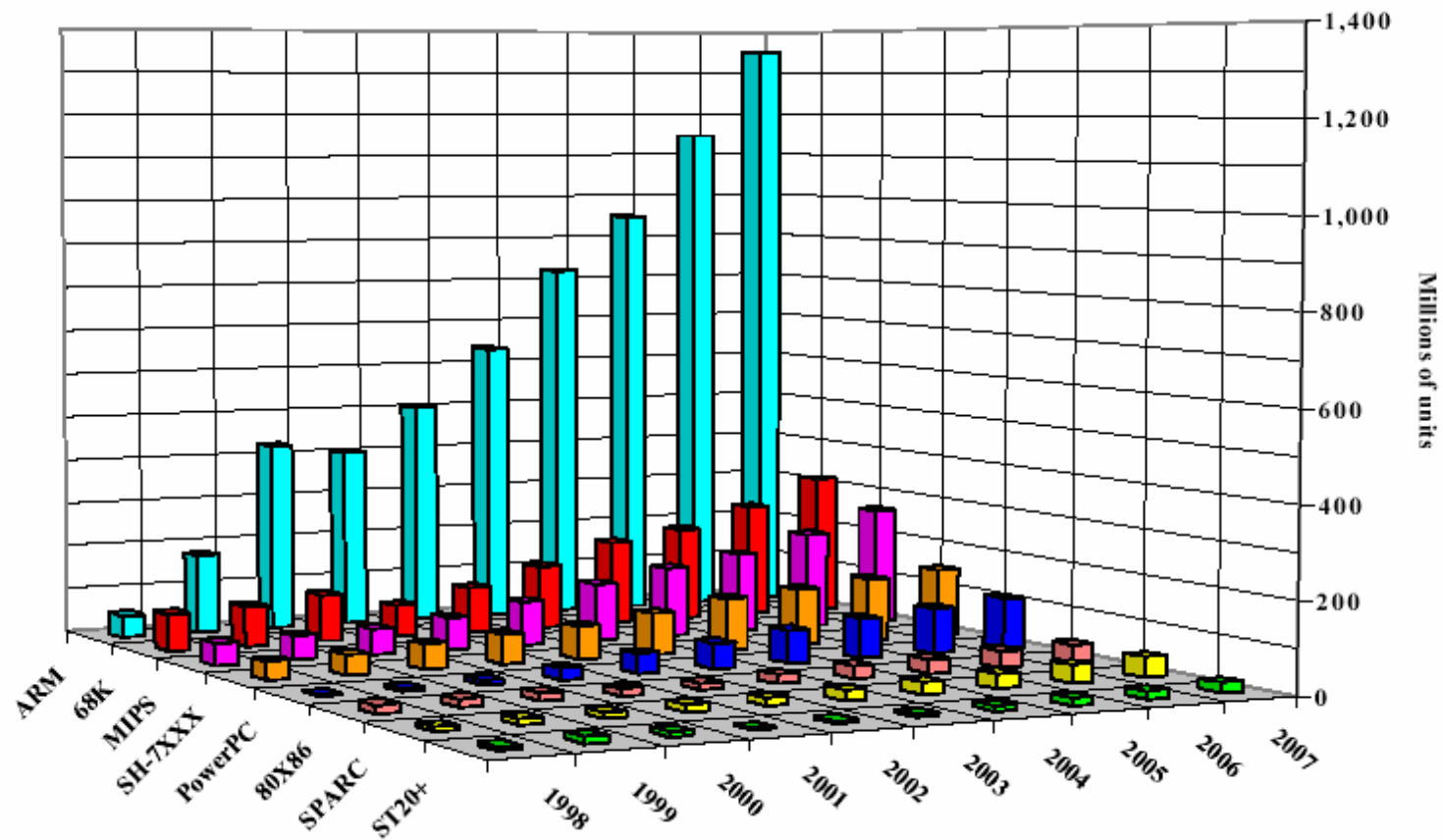
OEM 用来自
ARM Partner的
芯片设计制造最
终用户产品

ARM 微处理器

- 微处理器核: ARM6, ARM7, ARM9, ARM10, ARM11
- 扩展: Thumb, DSP, SIMD, Jazelle etc.
- 其它IP核: UART, GPIO, memory controllers, etc

CPU	Description	ISA	Process	Voltage	Area mm ²	Power mW	Clock / MHz	Mips / MHz
ARM7TDMI	Core	V4T	0.18u	1.8V	0.53	<0.25	60-110	0.9
ARM7TDMI-S	Synthesizable core	V4T	0.18u	1.8V	<0.8	<0.4	>50	0.9
ARM9TDMI	Core	V4T	0.18u	1.8V	1.1	0.3	167-220	1.1
ARM920T	Macrocell 16+16kB cache	V4T	0.18u	1.8V	11.8	0.9	140-200	1.06
ARM940T	Macrocell 8+8kB cache	V4T	0.18u	1.8V	4.2	0.85	140-170	1.06
ARM9E-S	Synthesizable core	V5TE	0.18u	1.8V	?	~1	133-200	1.1
ARM1020E	Macrocell 32+32kB cache	V5TE	0.15u	1.8V	~10	~0.85	200-400	1.25

ARM处理器的使用量



ARM 2003

ARM体系结构版本 - 1

I Version 1 (obsolete)

- I 基本数据处理
- I 字节, 字以及多字 load/store
- I 软件中断
- I 26 bit 地址总线

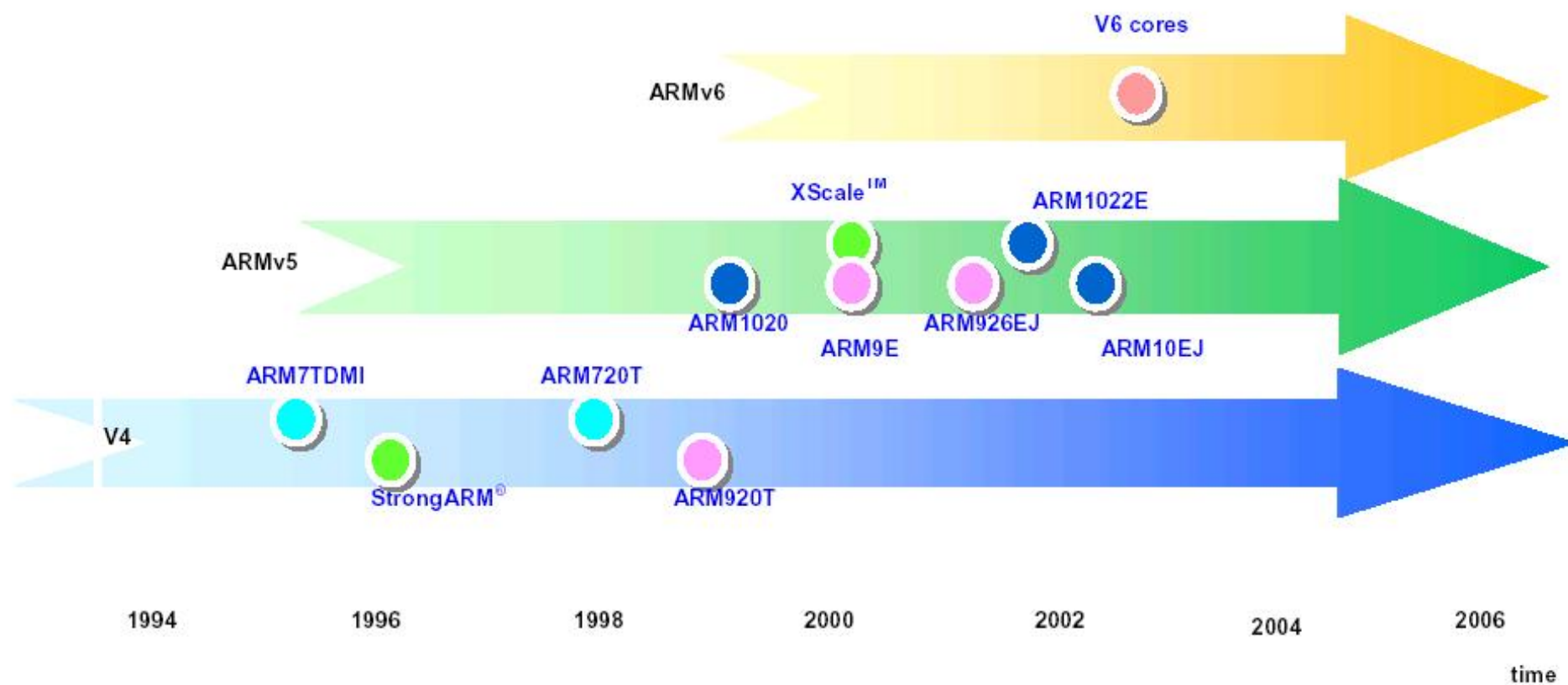
I Version 2 (obsolete)

- I Multiply & Multiply-accumulate
- I 支持协处理器
- I 支持线程同步
- I 26 bit 地址总线

ARM体系结构版本 - 2

- | V3版本推出32位寻址能力, 结构扩展变化为
 - T—16位压缩指令集
 - M—增强型乘法器, 产生全64位结果 ($32 \times 32 \rightarrow 64$ or $32 \times 32 + 64 \rightarrow 64$)
- | V4版本增加了半字load和store指令
- | V5版本改进了ARM和Thumb之间的交互, 结构扩展变化为:
 - 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

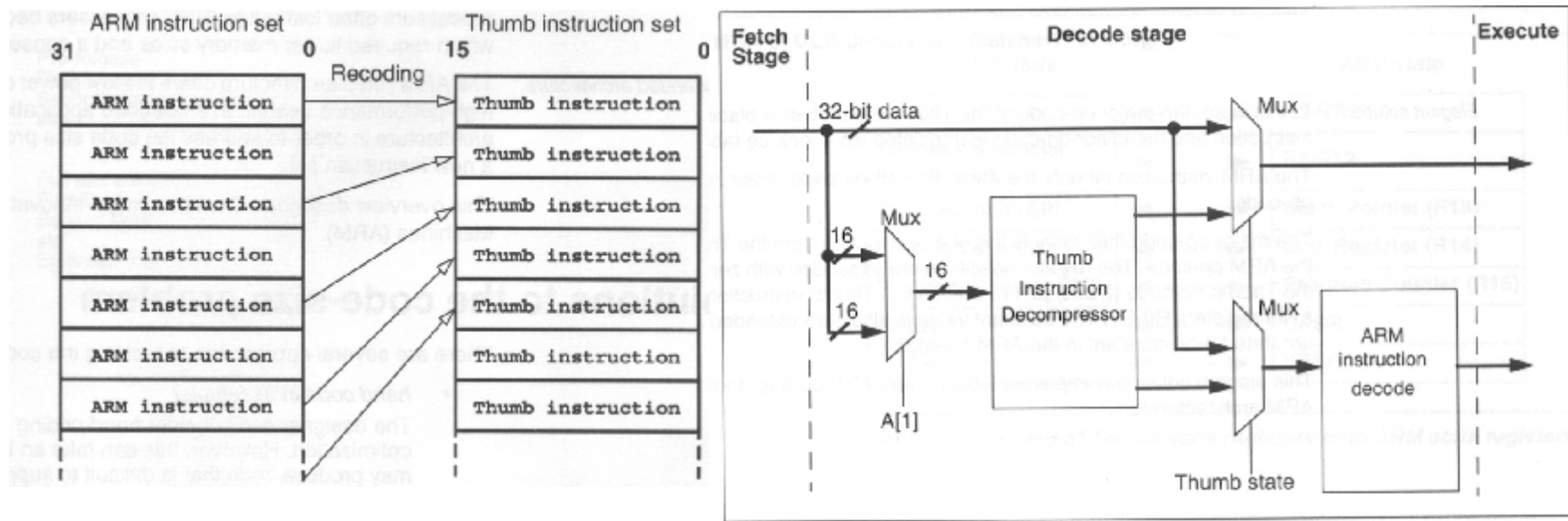
ARM 体系结构更新



体系结构变化 – 1*

I THUMB指令集 (‘T’)

▣ THUMB 指令集: 32位ARM指令集的子集, 按16位指令重新编码



▣ 代码尺寸小 (up to 40 % compression)

▣ 简化设计

体系结构变化 - 2

I 长乘法指令 ('M')

- u $32 \times 32 = 64$ bit. 提供全64位结果

I 增强DSP 指令集 ('E')

- u 可附加在ARM中的DSP指令
- u 64 bit 转换
- u 在 v5版本中第一次推出

I 处理器内核的变化

- u D: 在片调试. 处理器可响应调试暂停请求
- u I: Embedded ICE. 支持片上断点调试

体系结构变化 - 3

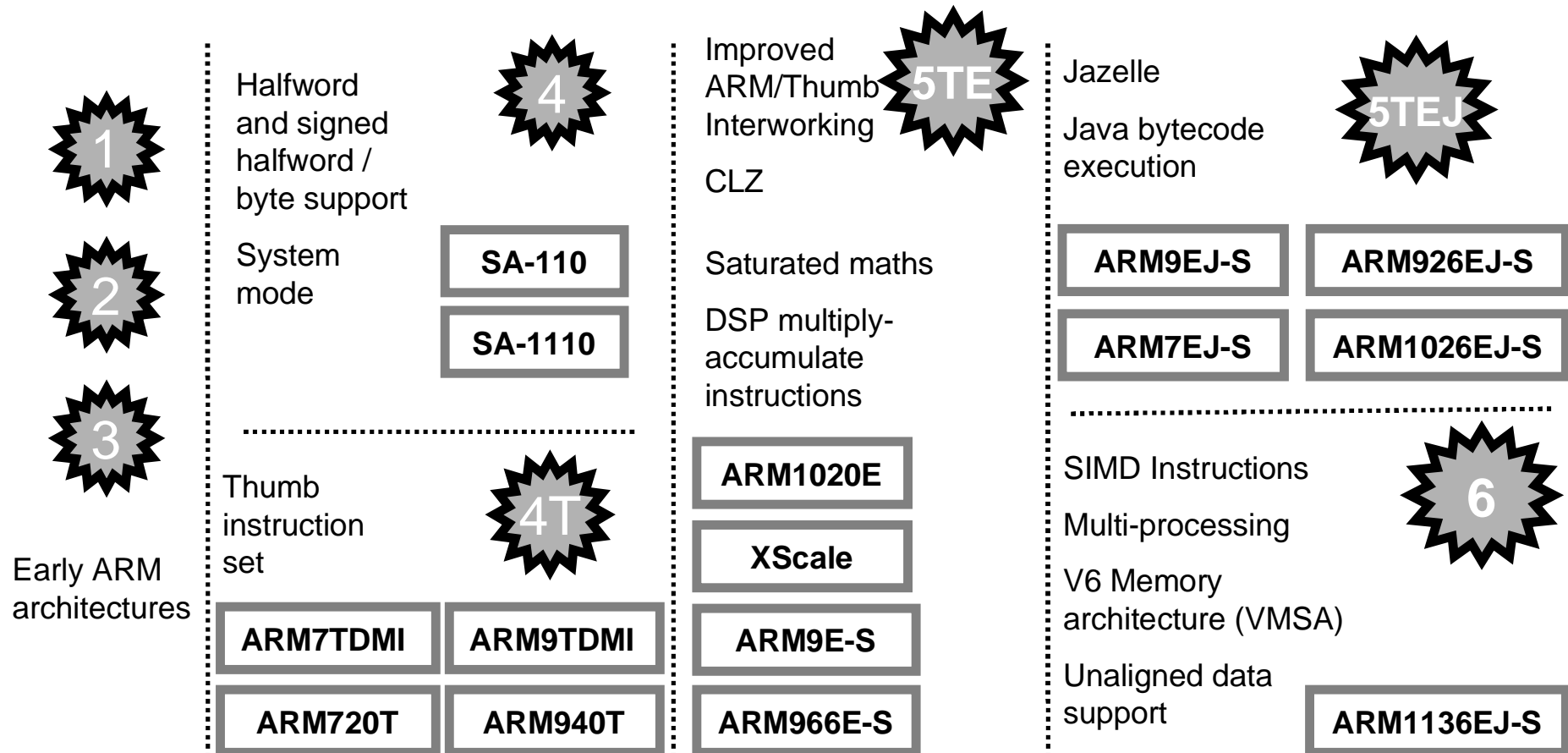
I ARM DSP 指令集

- 对于音频DSP应用提供高达70%的处理速度

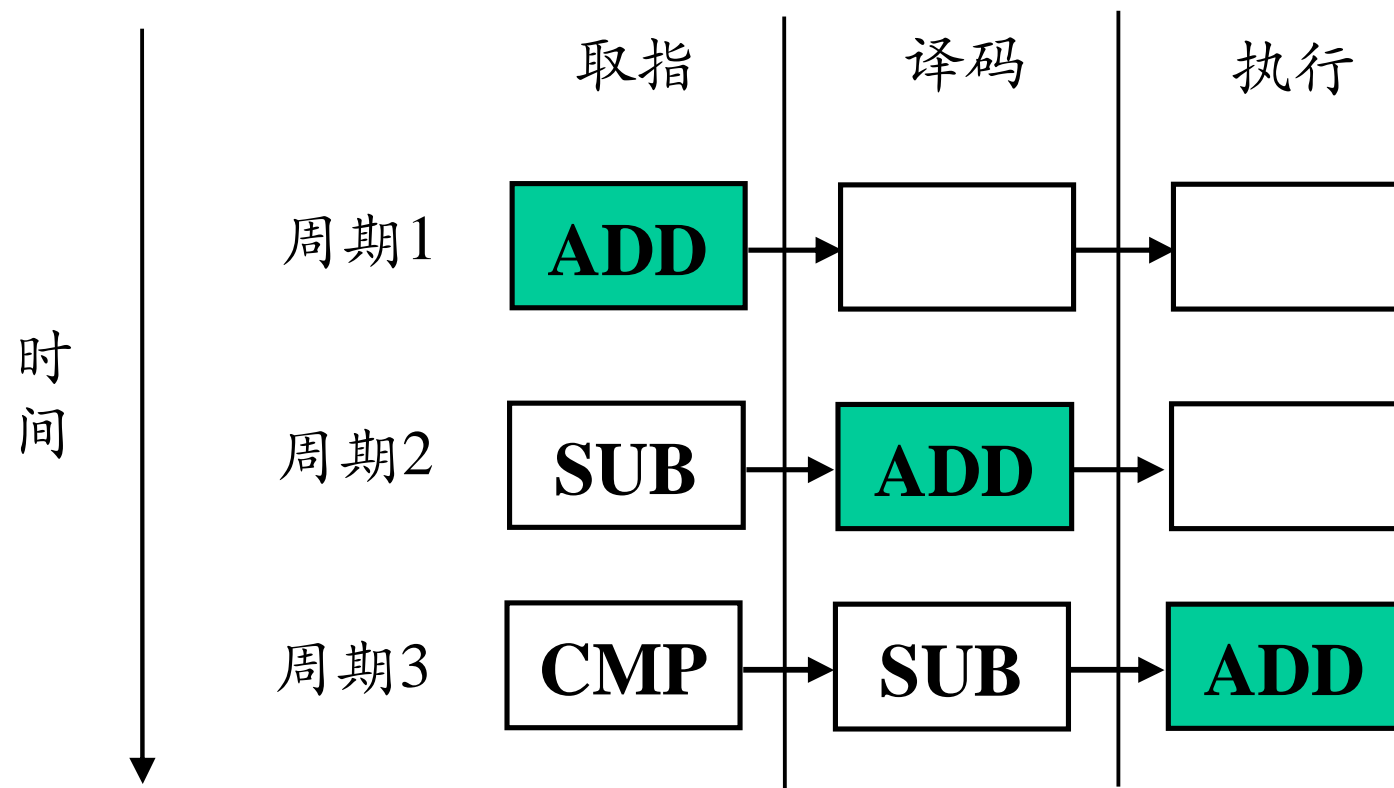
I Jazelle

- 提供比基于软件的JAVA虚拟机（JVM）更高的性能
- 与非JAVA加速核相比，提供8倍JAVA加速性能和降低80%的功耗
- 139 字节码直接在硬件上执行，88个字节码在软件上执行

ARM体系结构的发展



ARM流水线



ARM流水线的级数

- | ARM9流水线增加到5级，增加了存储器访问段和回写段，使ARM9处理能力平均可达到1.1 Dhrystone，指令吞吐量增加了约13%。
- | 随着流水线深度（级数）的增加，每一段的工作量被削减了，这使得处理器可以工作在更高的频率，同时改进了性能；
- | 负面作用是增加了系统的延时，即内核在执行一条指令前，需要更多的周期来填充流水线；
- | 流水线级数的增加也意味着在某些段之间会产生数据相关；

ARM流水线结构的发展

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 (Snnny)	执行 (Execute)	访存 (Memory)	写入 (Write)
-------	---------------	---------------	---------------	----------------	---------------	-----------------	----------------	---------------

ARM微处理器特性

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)
性能 MIPS**/MHz	0.97	1.1	1.3	1.2
架构	冯·诺伊曼	哈佛	哈佛	哈佛
乘法器	8×32	8×32	16×32	16×32

AMBA总线

- I 高级微控制器总线协议（AMBA）是1996年提出的，被ARM处理器做为片上总线结构；
- I 最初的AMBA总线包含ARM系统总线（ASB）和ARM外设总线（APB）；
- I ARM高性能总线（AHB）是新的标准，可以支持64位和128位宽度的ARM总线；

Cache和紧耦合器

- 丨 冯诺伊曼结构数据和指令共用一个缓存；哈佛体系结构有独立的指令和数据缓存；
- 丨 Cache改善了系统的整体性能，但也使程序的执行时间变得不可预测，对实时系统而言，代码执行的确定性——装载和存储指令或数据的时间必须是可预测的；
- 丨 ARM采用紧耦合器TCM实现可预测，TCM紧靠内核，保证取指或数据操作的时钟周期数。TCM位于存储器的地址映射中，可作为快速存储器访问；
- 丨 结合Cache和TCM，ARM即能改善性能，又能够获得可预测的实时响应；

存储器管理

- 1 无保护模式：没有存储器的硬件保护，只能提供非常有限的灵活性。通常用于小的、简单的嵌入式系统，不要求存储器保护；
- 1 提供有限保护的存储器保护单元（MPU）：MPU使用一个只用到少量存储区域的简单系统，这些区域由一组特殊的协处理器寄存器控制，每一个区域定义了专门的访问权限。适用于要求有存储器保护但没有复杂存储器系统映射的系统；
- 1 提供全面保护的存储器管理单元（MMU）：MMU使用一组转化表，以提供精细的存储器控制。这些表保存在主存里，并且提供虚拟地址与物理地址的映射和访问权限。MMU适用于支持多任务的复杂操作系统平台。

协处理器

- 丨 协处理器可以附属于ARM处理器，一个协处理器通过扩展指令或提供配置寄存器来扩展内核处理功能；
- 丨 协处理器可以通过一组专门的、提供load-store类型接口的ARM指令来访问。如协处理器15（CP15），用于控制Cache、TCM和存储器管理；
- 丨 协处理器也能通过提供一组专门的新指令来扩展指令集，如，处理向量浮点运算的指令集；
- 丨 这些指令在ARM流水线的译码阶段被处理，如果在译码阶段发现是一条协处理器指令，则把它送给相应的协处理器。如果该协处理器不存在，或不认识该指令，则ARM认为发生未定义指令异常；

ARM体系结构的命名规则

ARM{X}{y}{z}{T}{D}{M}{I}{E}{J}{F}{-S}

x——系列

y——存储管理/保护单元

z——Cache

T——Thumb16位译码器

D——JTAG调试器

M——快速乘法器

I——嵌入式跟踪宏单元

E——增强DSP指令

J——Jazelle

F——向量浮点单元

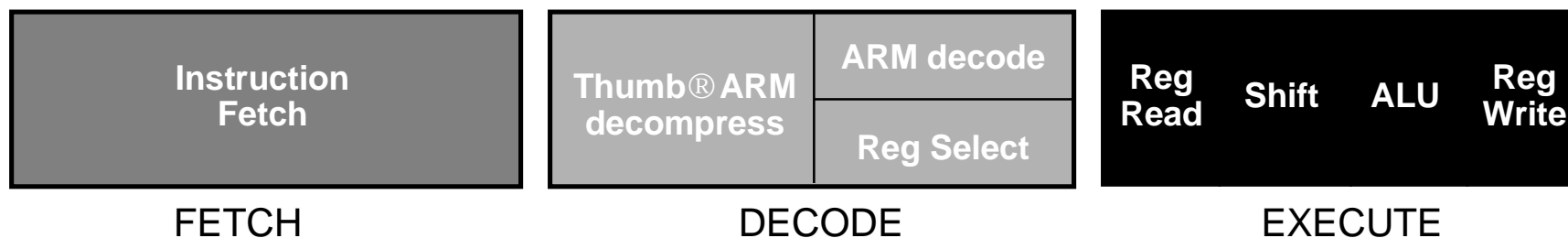
S——可综合版本，以源代码形式提供的ARM核

ARM9TDMI

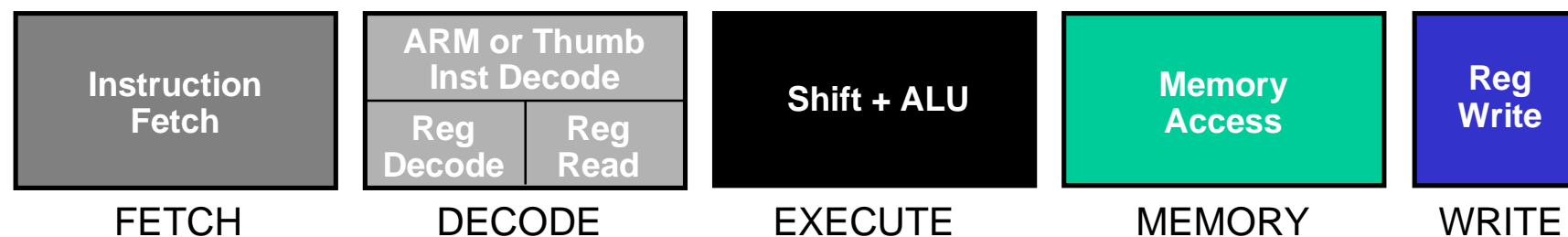
- | Harvard架构
 - | 增加了可用的存储器宽度
 - | 指令存储器接口
 - | 数据存储器接口
 - | 可以实现对指令和数据存储器的同时访问
- | 5 级流水线
- | 实现了以下改进：
 - | 改进 **CPI** 到 **~1.5**
 - | 提高了最大时钟频率

ARM9TDMI 流水线的变化

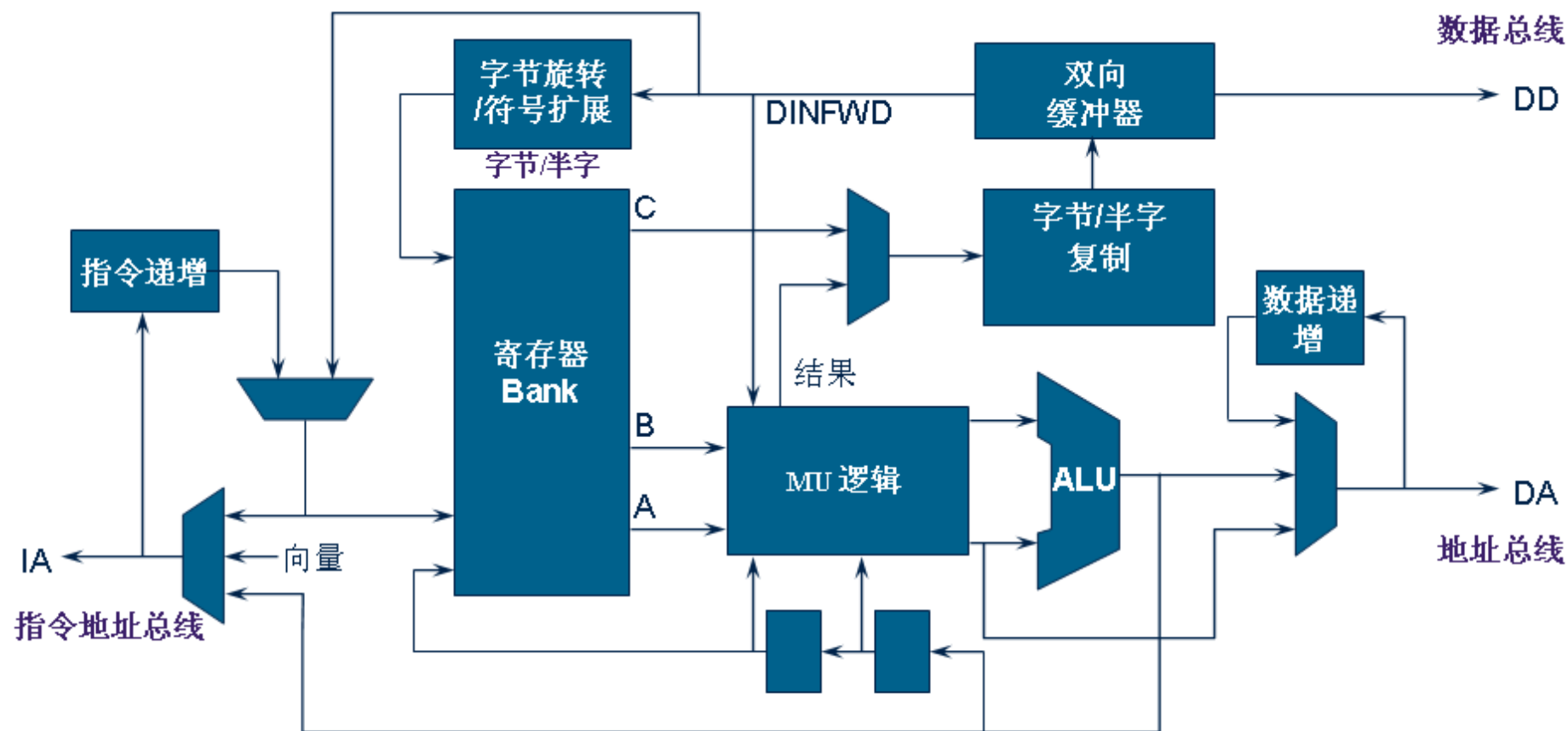
ARM7TDMI



ARM9TDMI

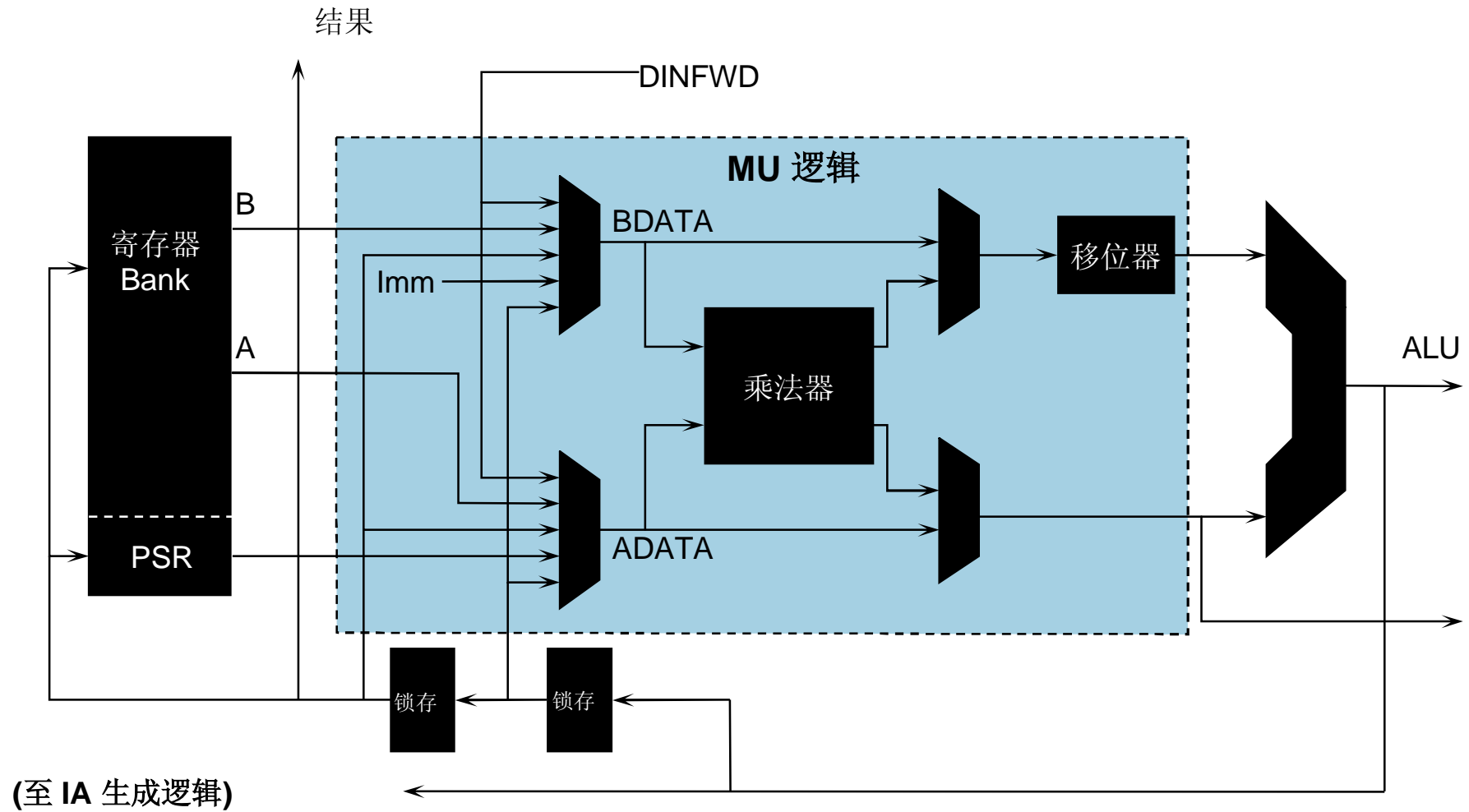


ARM9TDMI 数据通道 (1)



MU逻辑单元包含有：多路复用器,乘法器和桶形移位器

ARM9TDMI 数据通道 (2)



LDR互锁

周期			1	2	3	4	5	6	7	8	9	
操作												
ADD	R1, R1, R2	F	D	E		W						
SUB	R3, R4, R1		F	D	E		W					
LDR	<u>R4</u> , [R7]			F	D	E	M	W				
ORR	R8, R3, <u>R4</u>				F	D	I	E		W		
AND	R6, R3, R1					F	I	D	E		W	
EOR	R3, R1, R2							F	D	E		W

F – 取指 (Fetch)

D – 解码 (Decode)

E – 执行 (Execute)

I – 互锁 (Interlock)

M – 存储器 (Memory)

W – 写回 (Writeback)

- 本例中，用了7个时钟周期执行6条指令， **CPI = 1.2** 。
- LDR**指令之后立即跟一条数据操作指令，由于使用了相同的寄存器，将会导致互锁。

最佳流水线

周期		1	2	3	4	5	6	7	8	9
操作										
ADD	R1, R1, R2	F	D	E	W					
SUB	R3, R4, R1		F	D	E	W				
LDR	<u>R4</u> , [R7]			F	D	E	M	W		
AND	R6, R3, R1				F	D	E	W		
ORR	R8, R3, <u>R4</u>					F	D	E	W	
EOR	R3, R1, R2						F	D	E	W

F – 取指 (Fetch) D – 解码 (Decode) E – 执行 (Execute)
 I – 互锁 (Interlock) M – 存储器 (Memory) W – 写回 (Writeback)

- 本例中，用了6个时钟周期执行6条指令， **CPI = 1**。
- LDR**指令没有引起流水线互锁

LDM互锁 (1)

周期		1	2	3	4	5	6	7	8	9
操作										
LDMIA	R13!, {R0-R3}	F	D	E	M	MW	MW	MW	W	
SUB	R9, R7, R8		F	D	I	I	I	E		W
STR	R4, [R9]			F	I	I	I	D	E	M
ORR	R8, R4, R3						F	D	E	
AND	R6, R3, R1							F	D	E

F – 取指 (Fetch) **D** – 解码 (Decode) **E** – 执行 (Execute) **MW**-存储器 and 回写同时执行
I – 互锁 (Interlock) **M** – 存储器 (Memory) **W** – 写回 (Writeback)

- 本例中，用了8个时钟周期执行5条指令， **CPI = 1.6**
- 在**LDM**期间，有并行的存储器访问和回写周期

LDM 互锁 (2)

周期			1	2	3	4	5	6	7	8	9		
操作													
LDMIA	R13!, {R0-R3}	F	D	E	M	MW	MW	MW	W				
SUB	R9, R7, R3		F	D	I	I	I	I	E		W		
STR	R4, [R9]			F	I	I	I	I	D	E	M	W	
ORR	R8, R4, R3								F	D	E		W
AND	R6, R3, R1									F	D	E	

F – 取指（Fetch）

D – 解码（Decode）

E – 执行（Execute）

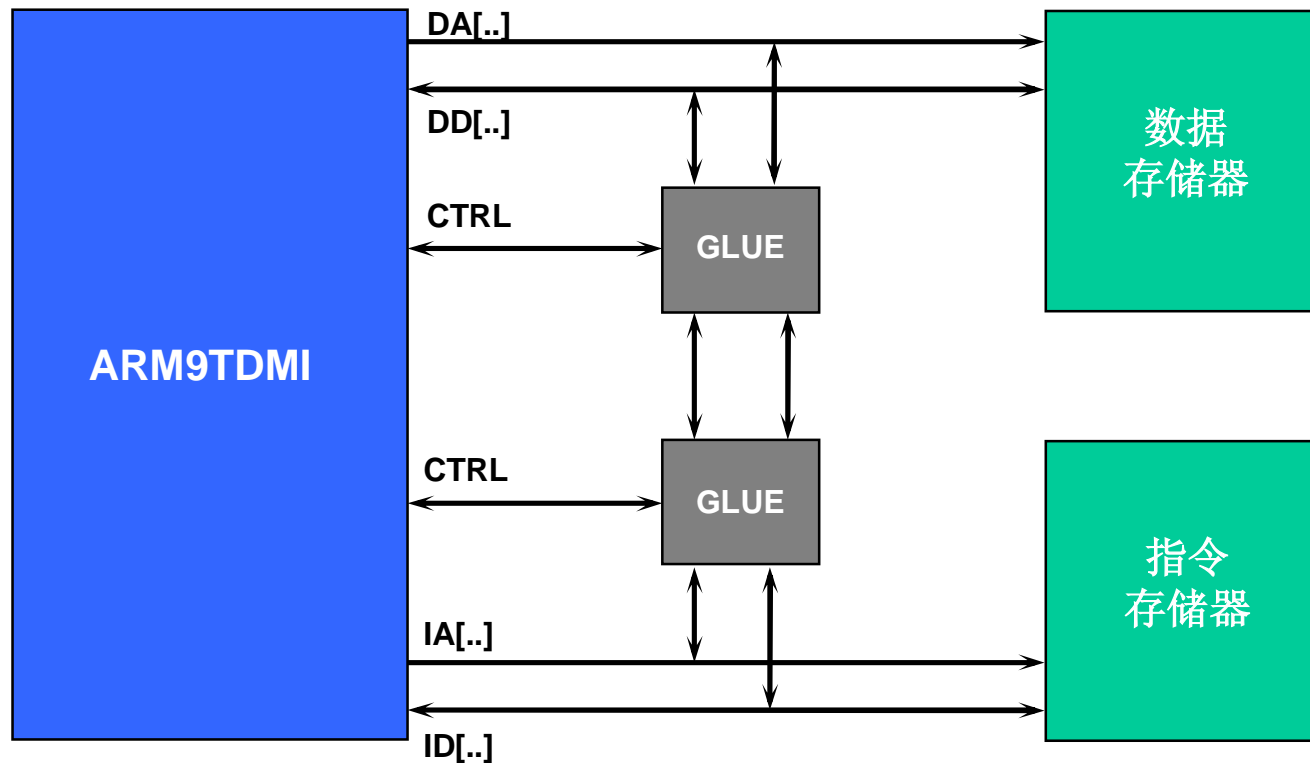
I – 互锁（Interlock）

M – 存储器（Memory）

W – 写回（Writeback）

- 本例中，用了**9**个时钟周期执行**5**条指令， **CPI = 1.8**
- 此处**SUB** 使用了 **R3**，增加了一个额外的互锁周期来完成该寄存器数据的获取
 - 这种情况对任何**LDM** 指令，像带**IA**, **DB**, **FD**,等，都会发生。

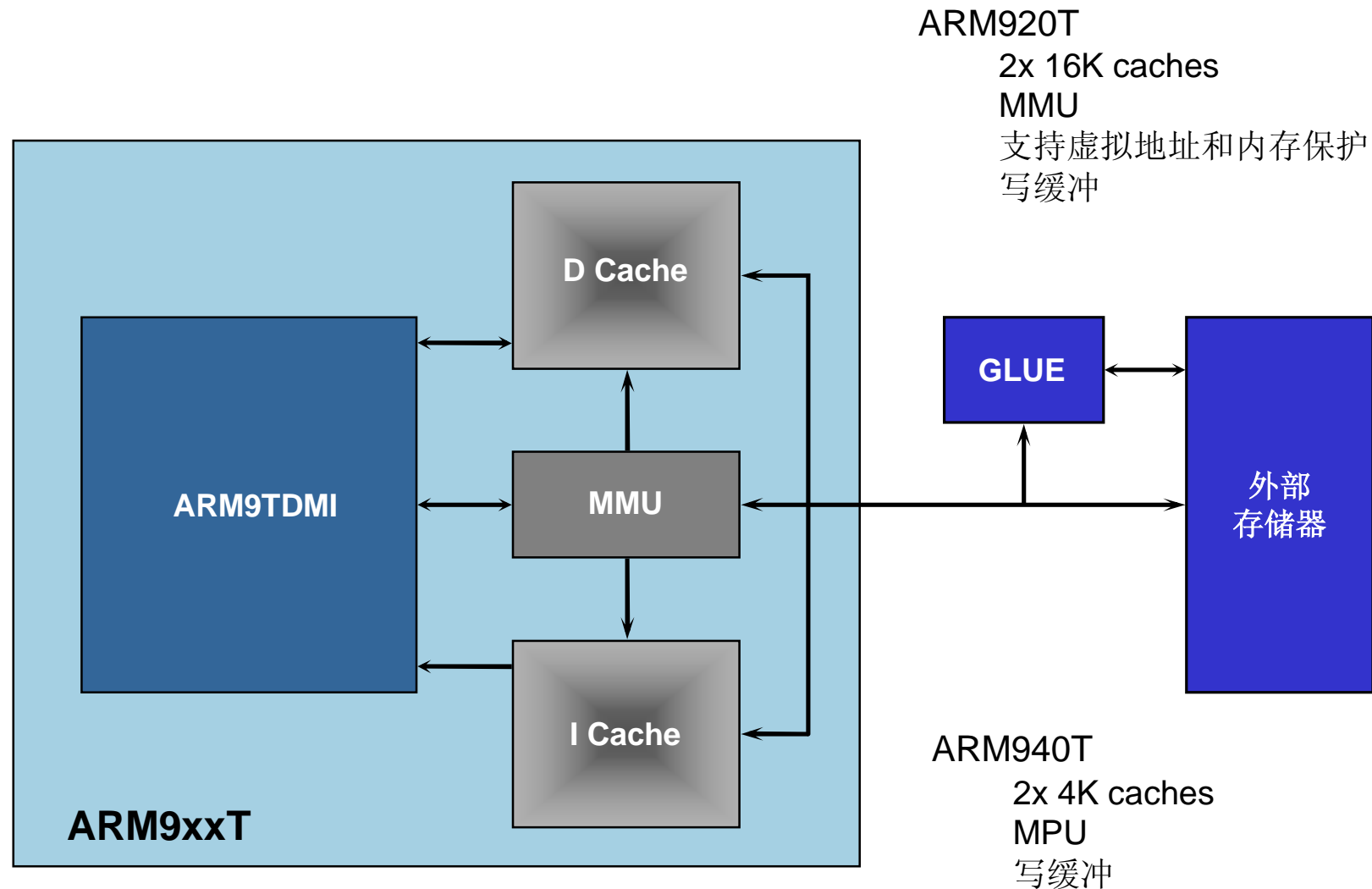
ARM9TDMI 系统举例



注意:

数据接口必须能够读取指令存储器中的数据。
为调试方便, 建议数据接口能够读写指令存储器。

带Cache的ARM9TDMI



ARM处理器结构综述（一）

ARM系列	微处理器核	特点
ARM7	<u>ARM7TDMI</u> : 整数处理核 ARM7TDMI 处理器的可综合版本; <u>ARM720T</u> : 带MMU的处理器核心, 支持操作系统; <u>ARM7EJ-S</u> : 带有DSP和Jazelle™ 技术, 能够实现Java加速功能	<ul style="list-style-type: none">丨 冯·诺伊曼体系结构;丨 ARMTDMI是目前应用最广的微处理器核丨 ARM720T带有MMU和8KB的指令数据混合cache;丨 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	<u>ARM920T</u> : 带有独立的16KB 数据和指令Cache; <u>ARM22T</u> : 带有独立的8位KB 数据和指令Cache; <u>ARM940T</u> -包括更小数据和指令Cache和一个MPU	<ul style="list-style-type: none">丨 基于ARM9TDMI , 带16位的Thumb指令集, 增强代码密度最多到35%;丨 在0.13μm工艺下最高性能可达到300MIPS (Dhrystone 2.1测试标准);丨 集成了数据和指令Chche;丨 32位AMBA总线接口的MMU支持;丨 可在0.18μm、 0.15μm和0.13μm工艺的硅芯片上实现。

ARM处理器结构综述（二）

ARM9E	<p>ARM926EJ-S: Jazelle 技术, 有MMU, 可配置的数据和指令Cache, TCM接口;</p> <p>ARM946E-S: 可配置的数据和指令Cache及TCM;</p> <p>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</p> <p>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;</p>	<p>! ARM9E是针对微控制器、DSP和Java的单处理器解决方案;</p> <p>! ARM Jazelle 技术提供 8倍的 Java 加速性能 (ARM926EJ-S) ;</p> <p>! 5-级整数流水线;</p> <p>! 在0.13μm工艺下最高性能可达到300MIPS (Dhrystone 2.1测试标准) ;</p> <p>! 可选择的 向量浮点单元VFP9 协处理器指令优秀浮点性能, 对于3D图形加速和实时控制可达到 215MFLOPS。</p> <p>! 高性能的AHB总线, 带MMU</p> <p>! 可在0.18μm, 0.15μm, 0.13μm工艺的硅芯片上实现。</p>
ARM10E	<p>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;</p> <p>ARM1022E: 与ARM1020E相同, 只是独立的数据和指令Cache变为16KB;</p> <p>ARM1026EJ-S: 同时具有MPU和MMU, 可综合版本;</p>	<p>! 带分支预测的6级整数流水线;</p> <p>! 在0.13μm工艺下最高性能可达到430MIPS (Dhrystone 2.1测试标准) ;</p> <p>! 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;</p> <p>! 双64位AMBA总线接口和64位内部总路线接口 ;</p> <p>! 优化的缓存结构提高了处理器访问低速存储器的性能;</p> <p>! 可在0.18μm, 0.15μm, 0.13μm工艺的硅芯片上实现</p>

ARM处理器结构综述（三）

ARM11	<p><u>ARM11 MPCore</u>: 可综合的多处理器核, 1至4个处理器可配置;</p> <p><u>ARM1136J(F)-S</u>: 可配置的数据和指令Cache, 可提供1.9位的MPEG4编码加速功能;</p> <p><u>ARM1156T2(F)-S</u>: 带集成浮点协处理器, 带内存保护单元MPU;</p> <p><u>ARM1176JZ(F)-S</u>: 带针对CPU和系统安全架构扩展的TrustZone技术。</p>	<ul style="list-style-type: none"> 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	<p><u>SC100</u>: 第一个32位安全处理器; 、<u>SC110</u>: 在SC100上增加密钥协处理器;</p> <p><u>SC200</u>: 带Jazelle技术的高级安全处理器;</p> <p><u>SC210</u>: 在SC200上增加密钥协处理器</p>	<ul style="list-style-type: none"> 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处理器结构综述（四）

Cortex	<p><u>Cortex-A</u>: 面向应用的微处理器, 针对复杂操作系统和应用程序设计;</p> <p><u>Cortex-R</u>: 针对实时系统的嵌入式处理器;</p> <p><u>Cortex-M</u>: 针对成本敏感应用优化的深度嵌入式处理器;</p>	<p>! 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;</p> <p>! 支持ARM、Thumb、Thumb-2指令集;</p> <p>! Thumb-2指令集提供了更高的代码存储密度, 进一步降低成本;</p>
Intel系列	<p><u>StrongARM</u>: ARMv4体系</p> <p><u>XScale</u>: ARMv5TE体系, 增加MMX指令</p>	<p>! 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;</p> <p>! 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。</p>

PXA架构的应用



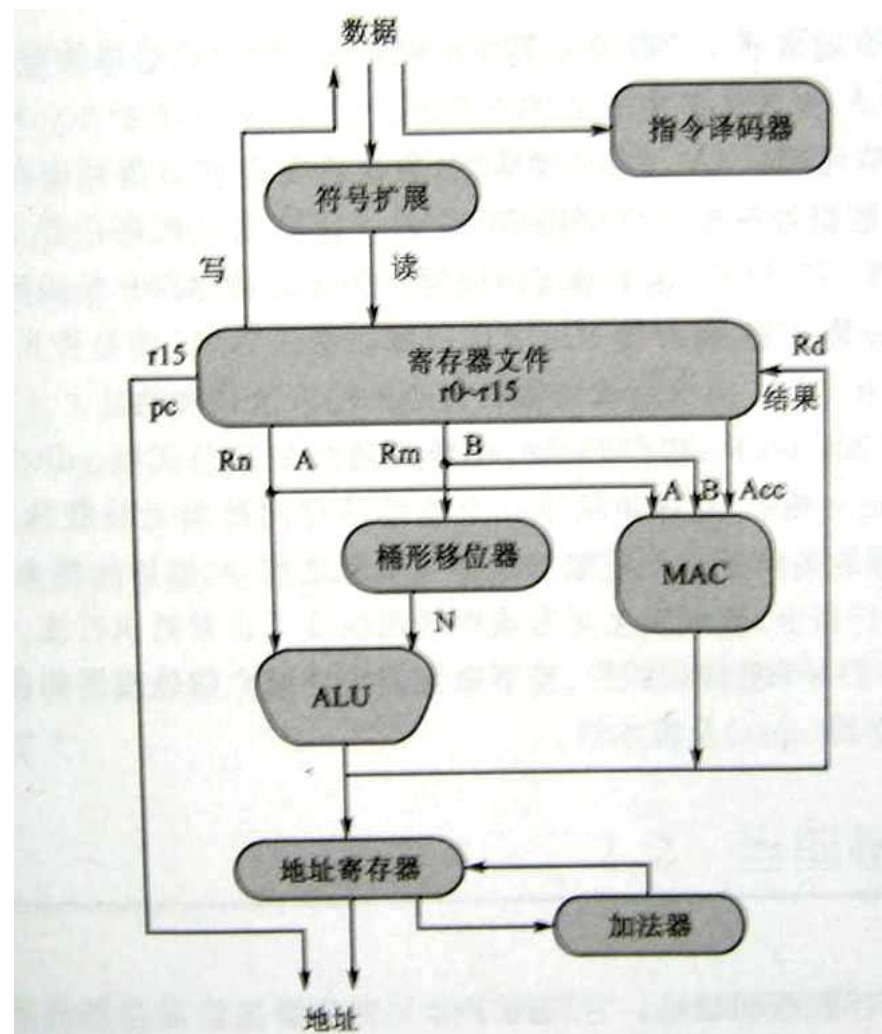
ARM微处理器的选型

- | ARM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

本节提要

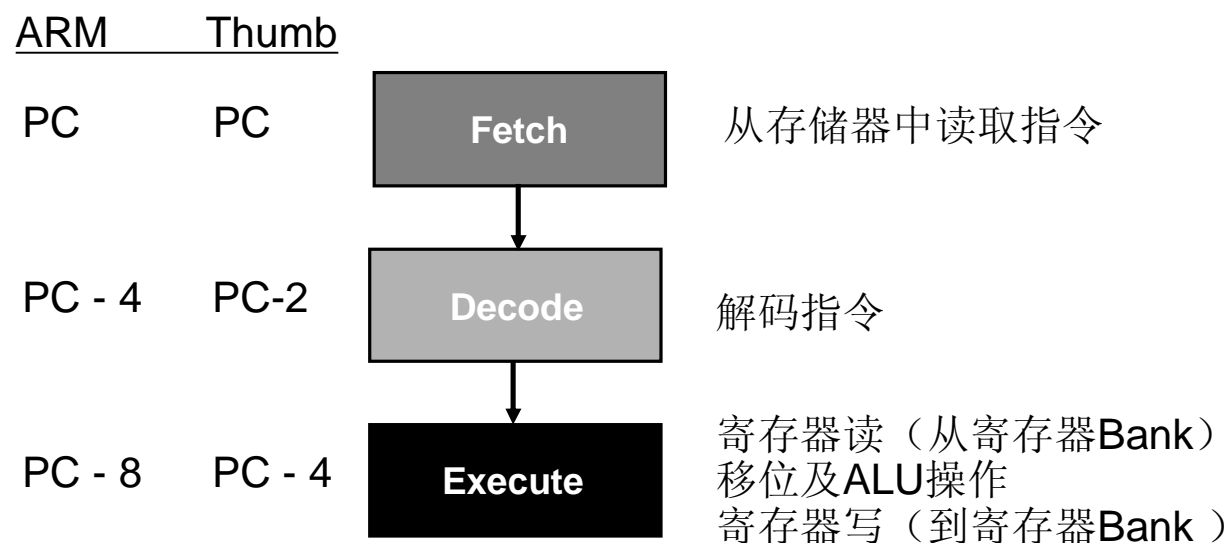
- 
- 1 嵌入式微处理器概述
 - 2 ARM体系结构概览
 - 3 **ARM编程模型**
 - 4 ARM 异常处理

ARM内核的数据流模型



指令流水线

- 为增加处理器指令流的速度，ARM7 系列使用3级流水线。
- 允许多个操作同时处理，比逐条指令执行要快。



- PC指向正被取指的指令，而非正在执行的指令

处理器的工作状态

- I ARM/TDMI 处理器有两种工作状态：
 - ARM - 32-bit, 按字排列的ARM指令集
 - Thumb - 16-bit, 按半字排列的Thumb指令集
- I ARM/TDMI 核的操作状态可能通过BX指令(分支和交换指令)在ARM状态和Thumb状态之间切换

例:

从ARM状态切换到Thumb状态:

```
LDR R0,=Label+1
```

```
BX R0
```

从Thumb状态切换到ARM状态:

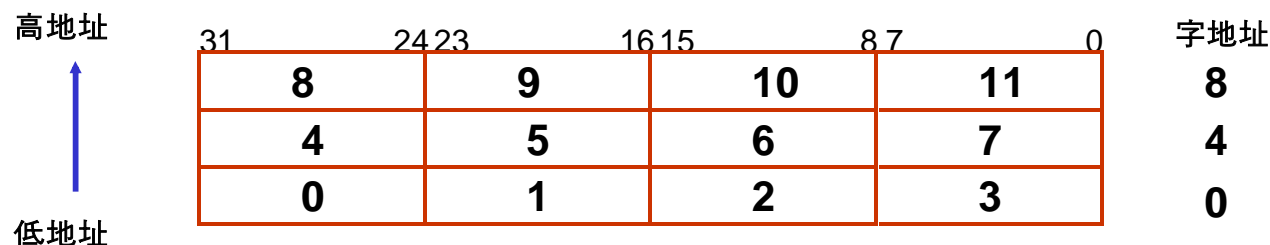
```
LDR R0,=Label
```

```
BX R0
```

存储器模式*

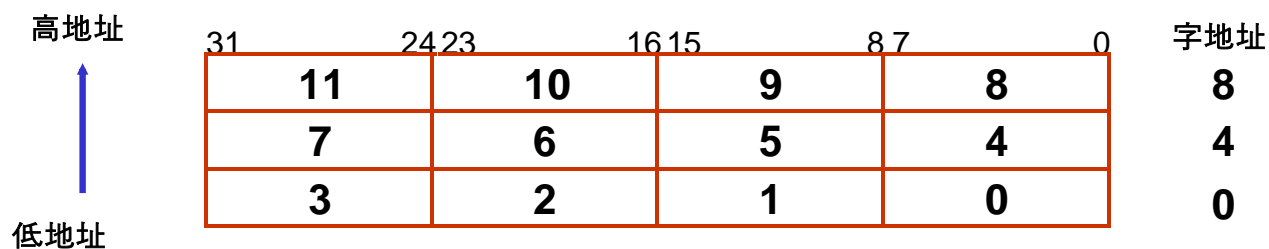
I 大端模式

- 字数据的高位字节存储在低地址中
- 字数据的低字节则存放在高地址中



I 小端模式

- 低地址中存放字数据的低字节
- 高地址中存放字数据的高字节



示例

A. 小端存储法(0x01234567)

地址	0x8000	0x8001	0x8002	0x8003
数据（16进制表示）	0x67	0x45	0x23	0x01
数据（二进制表示）	01100111	01000101	00100011	00000001

B. 大端存储法

地址	0x8000	0x8001	0x8002	0x8003
数据（16进制表示）	0x01	0x23	0x45	0x67
数据（二进制表示）	00000001	00100011	01000101	01100111

数据和指令类型

- | ARM 采用的是32位架构.
- | ARM 约定:
 - | Byte : 8 bits
 - | Halfword : 16 bits (2 byte)
 - | Word : 32 bits (4 byte)
- | 大部分ARM core 提供:
 - | ARM 指令集 (32-bit)
 - | Thumb 指令集(T变种) (16-bit)
- | Jazelle cores 支持 Java bytecode(J变种, 4TEJ)

处理器工作模式（1）

ARM 有7个基本工作模式:

- 1、User: 非特权模式, 大部分任务执行在这种模式
 - | 正常程序执行的模式
- 2、FIQ: 当一个高优先级(fast)中断产生时将会进入这种模式
 - | 高速数据传输和通道处理
- 3、IRQ: 当一个低优先级(normal)中断产生时将会进入这种模式
 - | 通常的中断处理
- 4、Supervisor: 当复位或软中断指令执行时将会进入这种模式
 - | 供操作系统使用的一种保护模式
- 5、Abort: 当存取异常时将会进入这种模式
 - | 虚拟存储及存储保护
- 6、Undef: 当执行未定义指令时会进入这种模式
 - | 软件仿真硬件协处理器
- 7、System: 使用和User模式相同寄存器集的特权模式
 - | 特权级的操作系统任务

寄存器组织 – 1*

User32		Fiq32		Supervisor32		Abort32		IRQ32		Undefined32
R0		R0		R0		R0		R0		R0
R1		R1		R1		R1		R1		R1
R2		R2		R2		R2		R2		R2
R3		R3		R3		R3		R3		R3
R4		R4		R4		R4		R4		R4
R5		R5		R5		R5		R5		R5
R6		R6		R6		R6		R6		R6
R7		R7		R7		R7		R7		R7
R8		R8_fiq		R8		R8		R8		R8
R9		R9_fiq		R9		R9		R9		R9
R10		R10_fiq		R10		R10		R10		R10
R11		R11_fiq		R11		R11		R11		R11
R12		R12_fiq		R12		R12		R12		R12
R13(SP)		R13_fiq		R13_svc		R13_abt		R13_irq		R13_und
R14(LR)		R14_fiq		R14_svc		R14_abt		R14_irq		R14_und
R15(PC)		R15(PC)		R15(PC)		R15(PC)		R15(PC)		R15(PC)

CPSR		CPSR		CPSR		CPSR		CPSR		CPSR
		SPSR_fiq		SPSR_svc		SPSR_abt		SPSR_irq		SPSR_und

ARM 寄存器

当前可见寄存器

Abort Mode

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
r15 (pc)
cpsr
spsr

备用寄存器

User

FIQ

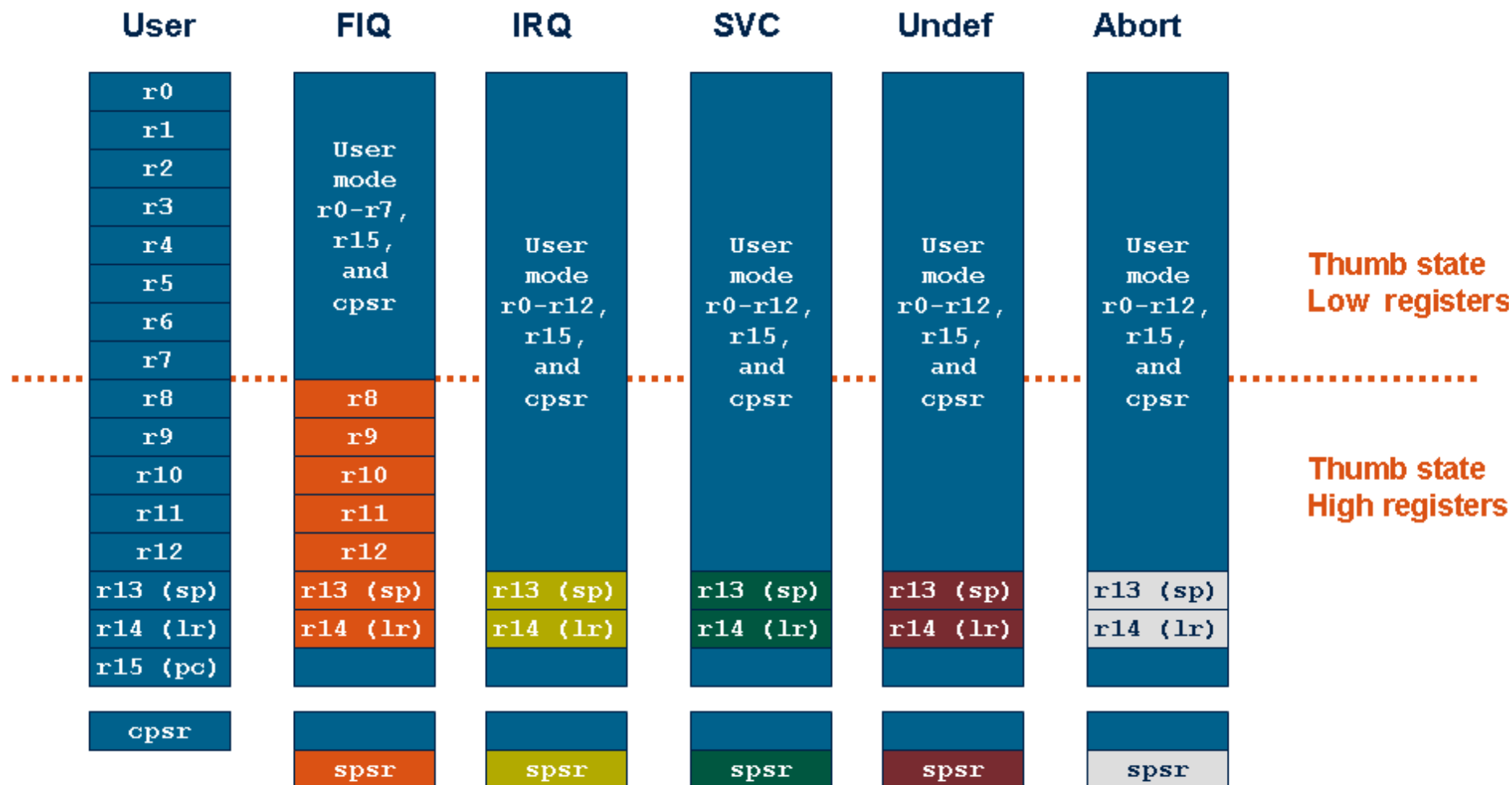
IRQ

SVC

Undef

	r8			
	r9			
	r10			
	r11			
	r12			
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
	spsr	spsr	spsr	spsr

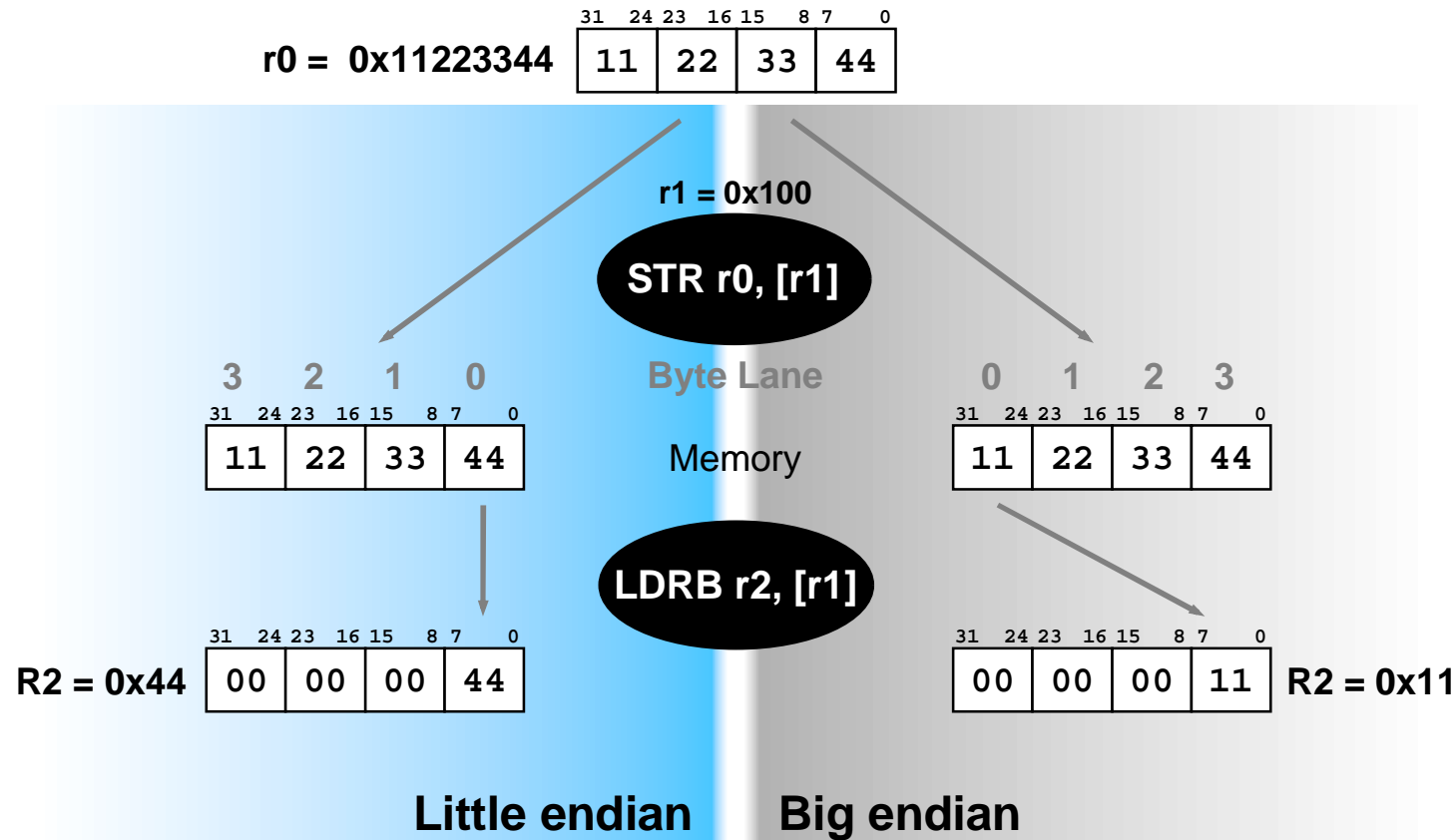
寄存器组织概要



Note: System模式使用user模式寄存器集

字节顺序

- The ARM 可以用 little/big endian 格式存取数据.



- For more information, see:
“Application Note 61: Big and Little Endian Byte Addressing”

寄存器 - 2

I 37 寄存器

- u 31 个通用32位寄存器，包括程序计数器PC
 - o 未分组寄存器R0-R7
 - o 分组寄存器R8-R14
 - o 程序计数器PC (R15)
- u 6 个状态寄存器
- u 15 通用寄存器 (R0 to R14)，以及1或者2个状态寄存器和程序计数器在任何时候都是可以访问的

I 可访问的寄存器取决于处理器的模式

- I 其它寄存器 (the banked registers) 的状态在支持IRQ, FIQ, 管理员, 中止和未定义模式处理时被切换

寄存器 - 3

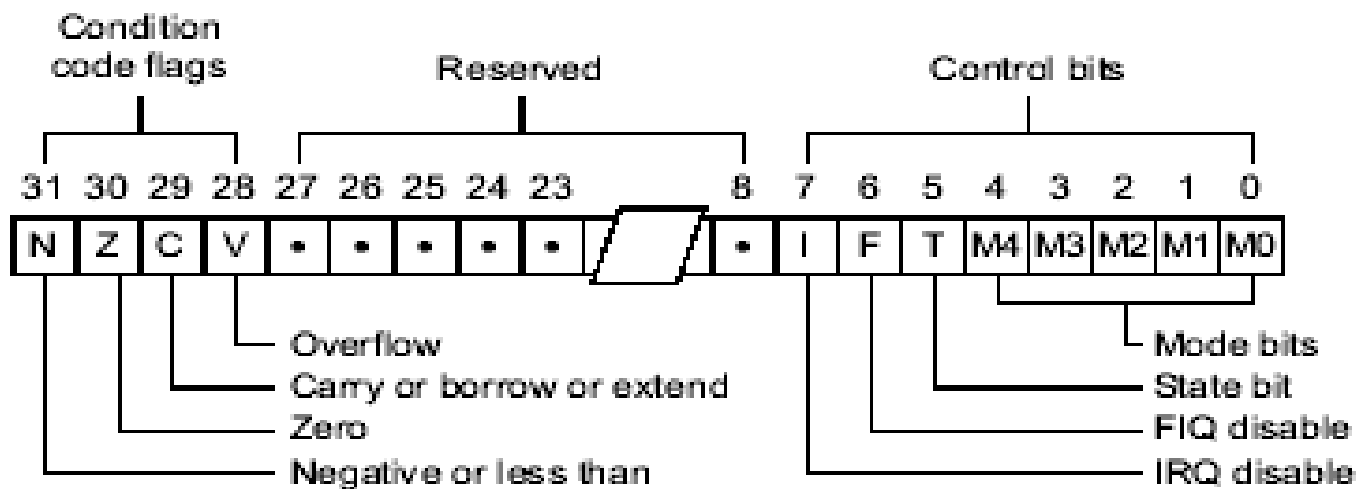
- | R0 到 R15 可以直接访问
- | R0 到 R14 是通用寄存器
- | R13: 堆栈指针 (sp) (通常)
 - └ 每种处理器模式都有单独的堆栈
- | R14: 链接寄存器 (lr)
- | R15 包含程序计数器 (PC)
- | CPSR — 当前程序状态寄存器，包括代码标志状态和当前模式位
- | 5个SPSRs-- (程序状态保存寄存器) 当异常发生时保存CPSR状态

程序状态寄存器 - 1

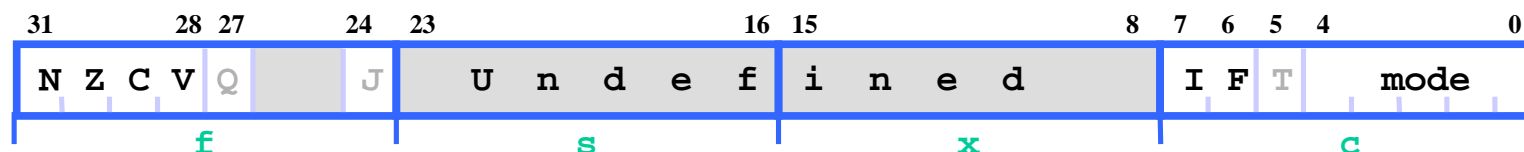
- | ARM/TDMI 包含当前程序状态寄存器 (CPSR)，加上5个程序状态保存寄存器SPSR，当异常发生时，用于保存CPSR的状态
- | 这些寄存器的功能是：
 - ▣ 保存ALU当前操作信息
 - ▣ 控制允许和禁止中断
 - ▣ 设置处理器操作模式

程序状态寄存器 - 2

- N, Z, C and V 条件码标志
 - 可以在处理器中作为数学和逻辑操作改变
 - 可以被所有的指令测试，以决定指令是否被执行
 - N : Negative. Z : Zero. C : Carry. V : oVerflow
- I and F 位是中断禁止位
- MD, M1, M2, M3 and M4 位是模式位



程序状态寄存器



条件位:

- | N = 1-结果为负, 0-结果为正或0
- | Z = 1-结果为0, 0-结果不为0
- | C = 1-进位, 0-借位
- | V = 1-结果溢出, 0结果没溢出

Q 位:

- | 仅ARM 5TE/J架构支持
- | 指示增强型DSP指令是否溢出

J 位

- | 仅ARM 5TE/J架构支持
- | J = 1: 处理器处于Jazelle状态

中断禁止位:

- | I = 1: 禁止 IRQ.
- | F = 1: 禁止 FIQ.

T Bit

- | 仅ARM xT架构支持
- | T = 0: 处理器处于 ARM 状态
- | T = 1: 处理器处于 Thumb 状态

Mbde位(处理器模式位):

- | 0b10000 User
- | 0b10001 FIQ
- | 0b10010 IRQ
- | 0b10011 Supervisor
- | 0b10111 Abort
- | 0b11011 Undefined
- | 0b11111 System

处理器工作模式（2）

- 丨 处理器模式主要决定了哪些寄存器是活动的及对CPSR的访问权；
- 丨 除USER模式之外，其余都是特权模式，特权模式允许对CPSR的完全读/写访问；
- 丨 非特权模式中允许对CPSR的控制域进行读访问，但允许对标志位的读写访问；
- 丨 系统模式是一种特殊的用户模式，允许对CPSR的完全读写访问；

程序状态寄存器PSR的模式位

M[4:0]	Mode	Visible Thumb-state registers	Visible ARM-state registers
10000	User	r0-r7, SP, LR, PC, CPSR	r0-r14, PC, CPSR
10001	FIQ	r0-r7, SP_fiq, LR_fiq, PC, CPSR, SPSR_fiq	r0-r7, r8_fiq-r14_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	r0-r7, SP_irq, LR_irq, PC, CPSR, SPSR_irq	r0-r12, r13_irq, r14_irq, PC, CPSR, SPSR_irq
10011	Supervisor	r0-r7, SP_svc, LR_svc, PC, CPSR, SPSR_svc	r0-r12, r13_svc, r14_svc, PC, CPSR, SPSR_svc
10111	Abort	r0-r7, SP_abt, LR_abt, PC, CPSR, SPSR_abt	r0-r12, r13_abt, r14_abt, PC, CPSR, SPSR_abt
11011	Undefined	r0-r7, SP_und, LR_und, PC, CPSR, SPSR_und	r0-r12, r13_und, r14_und, PC, CPSR, SPSR_und
11111	System	r0-r7, SP, LR, PC, CPSR	r0-r14, PC, CPSR

程序指针PC (r15)

┆ 当处理器执行在ARM状态:

- ┆ 所有指令 32 bits 宽
- ┆ 所有指令必须 word 对齐
- ┆ 所以 pc值由bits [31:2]决定, bits [1:0] 未定义 (所以指令不能halfword / byte对齐).

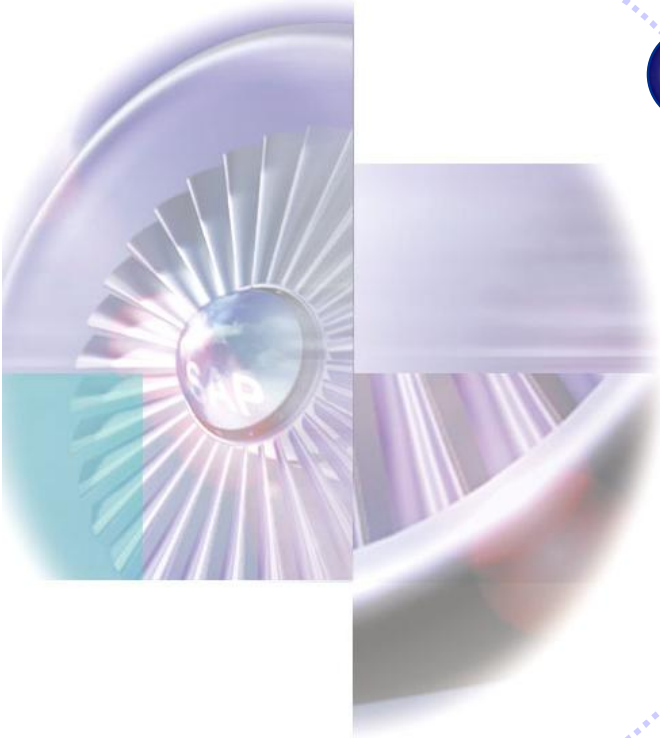
┆ 当处理器执行在Thumb状态:

- ┆ 所有指令 16 bits 宽
- ┆ 所有指令必须 halfword 对齐
- ┆ 所以 pc值由bits [31:1]决定, bits [0] 未定义 (所以指令不能 byte对齐).

┆ 当处理器执行在Jazelle状态:

- ┆ 所有指令 8 bits 宽
- ┆ 处理器执行 word 存取一次取4条指令

本节提要

- 
- 1 嵌入式微处理器概述
 - 2 ARM体系结构概览
 - 3 ARM编程模型
 - 4 **ARM 异常处理**

异常和中断

- 1 异常 (Exception)是指任何打断处理器正常执行, 并且迫使处理器进入一个由有特权的特殊指令执行的事件。
- 1 异常可分为两类: 同步异常 (synchronous exceptions)和异步异常 (asynchronous exceptions);

同步异常

- | 同步异常是由内部事件（如处理器指令运行产生的事件）引起的异常称为同步异常，包括：
 - | 在某些处理器体系结构中，对于确定的数据尺寸必须从内存的偶数地址进行读和写操作。从一个奇数内存地址的读或写操作将引起存储器存取一个错误事件并且引起一个异常；
 - | 造成被零除的算术运算引发一个异常

异步异常

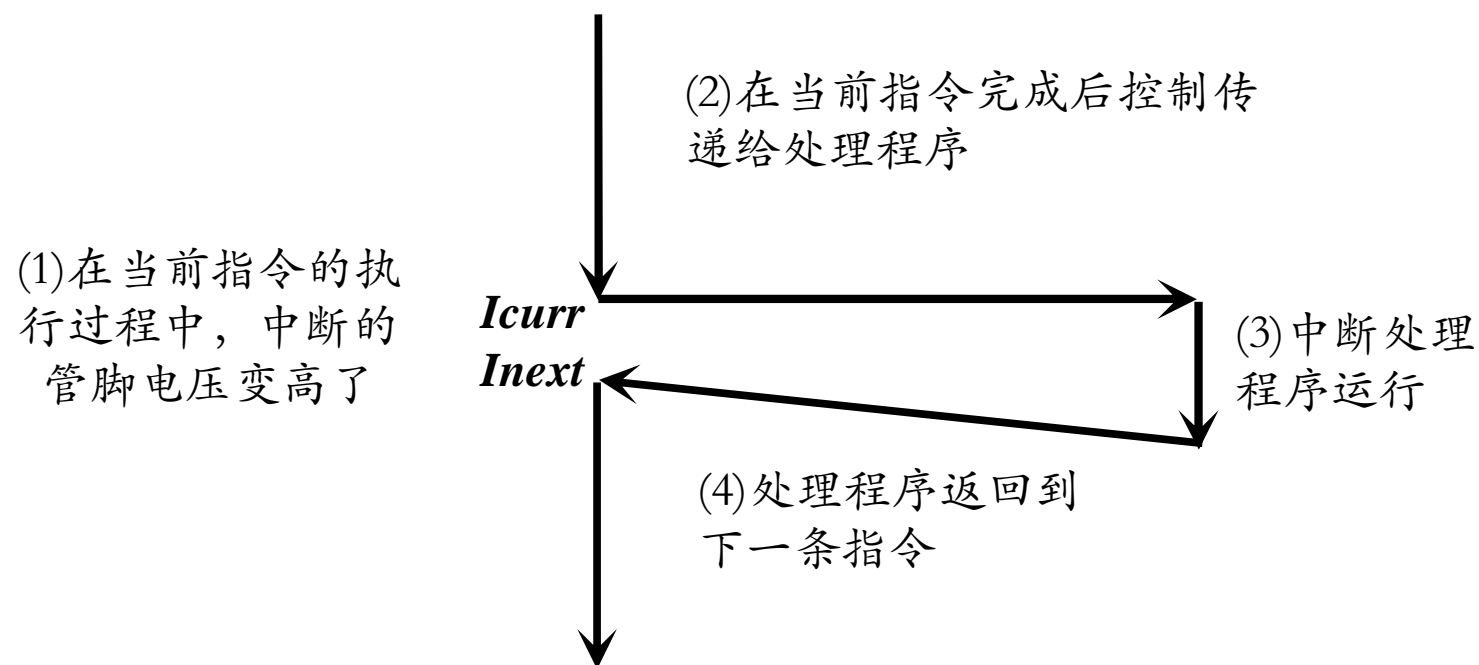
- | 异步异常是由外部事件（如处理器指令执行不相关的事件）引起的异常称为异步异常，一般这些外部事件与硬件信号相关，又称为中断，包括：
 - | 复位异常，按下嵌入式板上的复位按钮，触发一个异步的异常；
 - | 如串口、网口等通讯模块，接收数据包产生异常；

异常分类

- I 异常可分为4类：中断（interrupt）、陷井（trap）、故障（fault）和终止（abort）

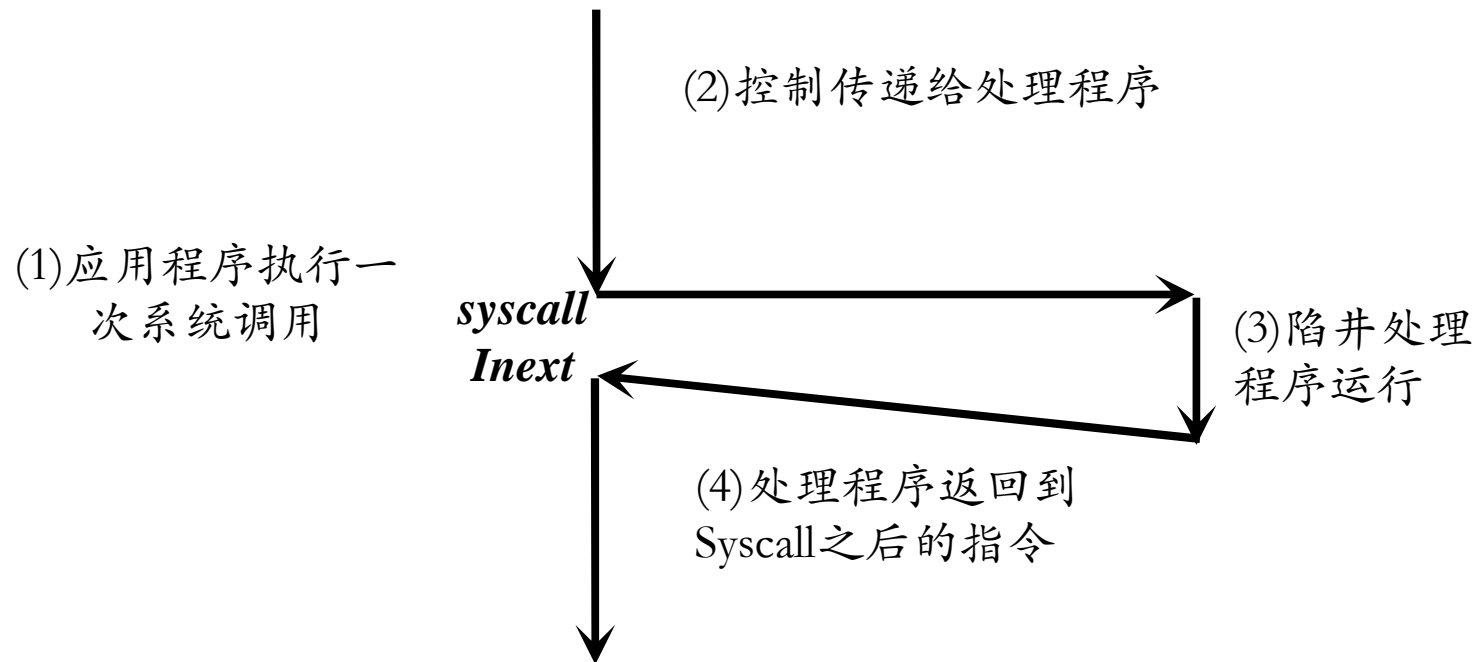
类别	原因	异步/同步	返回行为
中断	来自I/O设备的信号	异步	总是返回到下一条指令
陷井	有意的异常	同步	总是返回到下一条指令
故障	潜在可恢复的错误	同步	可能返回到当前指令
终止	不可恢复的错误	同步	不会返回

中断



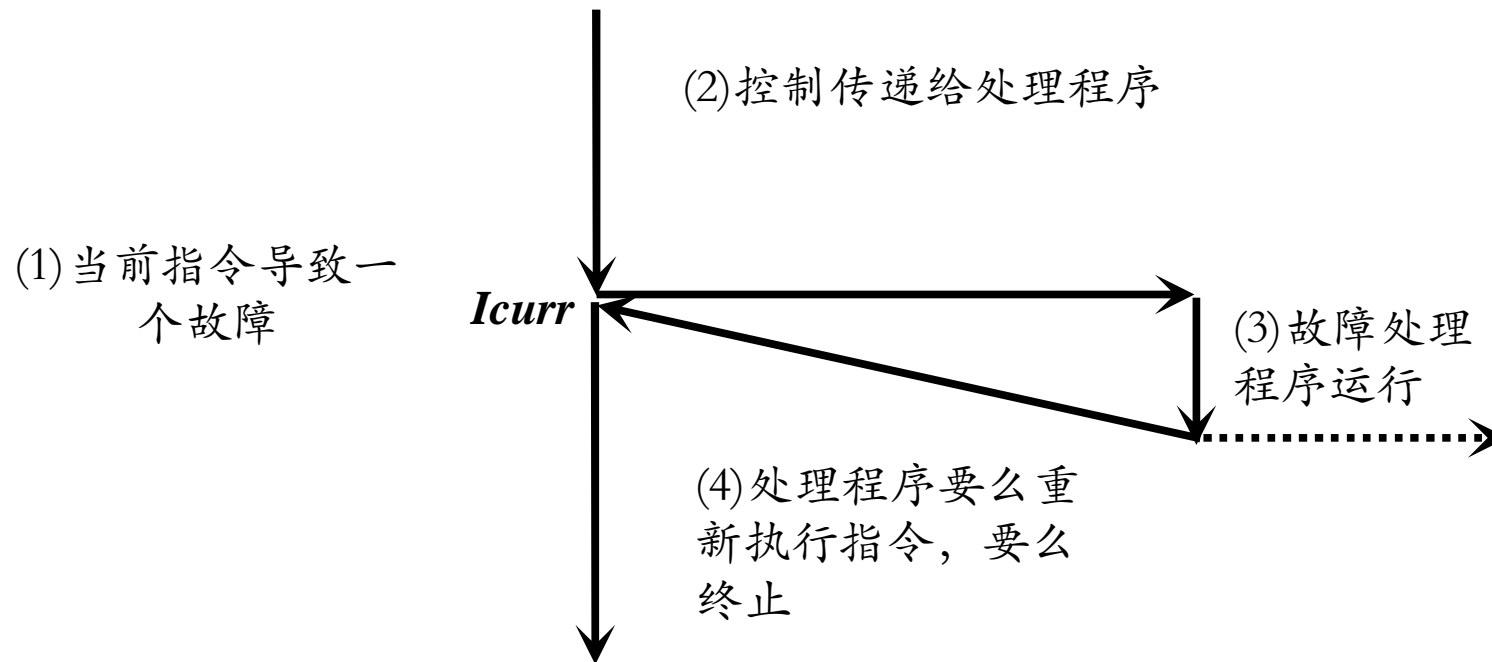
陷井

- I 陷井是有意的异常，通常在用户程序和内核之间提供系统调用。



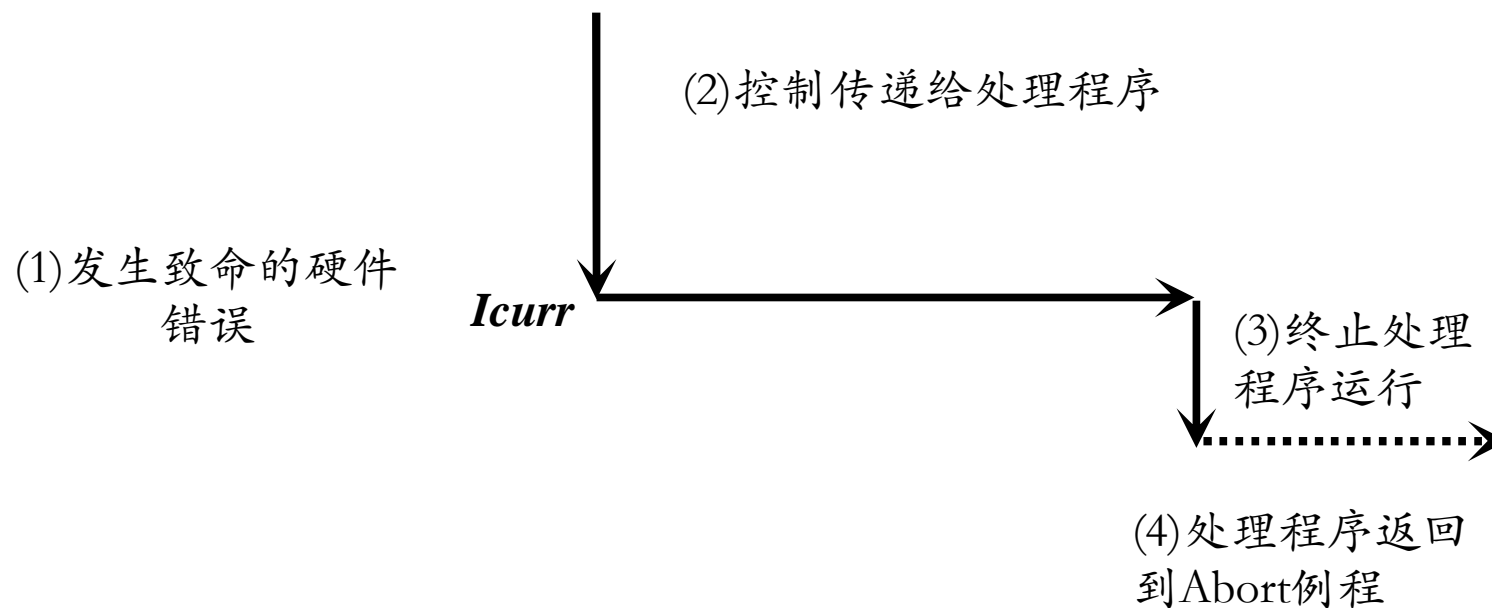
故障

- I 故障是由错误情况引起的，它可能被故障处理程序修正，如缺页异常



终止

- I 终止是不可恢复的致命错误造成的结果，如硬件错误，DRAM或SRAM位损坏时发生的奇偶错误。



ARM异常 - 1

- I 异常——内部或外部中断源产生并引起处理器处理一个事件，如外部中断或试图执行未定义指令都会引起异常。
 - u 处理异常之前必须保留处理器的状态
- I 异常类型
 - u FIQ
 - u IRQ(Interrupt ReQuest)
 - u 未定义指令
 - u 预取中止
 - u 数据中止
 - u 复位
 - u 软件中断Software interrupt
 - u 通过软件中断产生
 - u 进行管理员模式中获得
 - u 通常要求特殊的管理功能，如操作系统支持

异常 - 2

I 异常类型

u 未定义的指令陷阱

- u 当ARM接受到一条不能处理的指令, ARM把这条指令提供给任何一个协处理器执行
- u 如果协处理器可以执行这条指令但此时协处理器忙, ARM将等待直到协处理器准备好或中断发生
- u 如果没有协处理器处理这条指令, 那么ARM将处理未定义的指令陷阱

I 异常优先级

- (1) Reset (highest priority)
- (2) Data abort
- (3) FIQ
- (4) IRQ
- (5) Prefetch abort
- (6) 未定义指令, Software interrupt (最低优先级)

异常 - 3

- I 只要产生异常就会导致正常和程序流程被临时停止, 例如外围中断服务程序
- I 在异常被处理前, 当前的处理器状态必须被保存, 以便处理程序完成后, 最后的程序可以被恢复.

异常向量

Address	Exception	Mode on entry
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	<i>Reserved</i>	<i>Reserved</i>
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

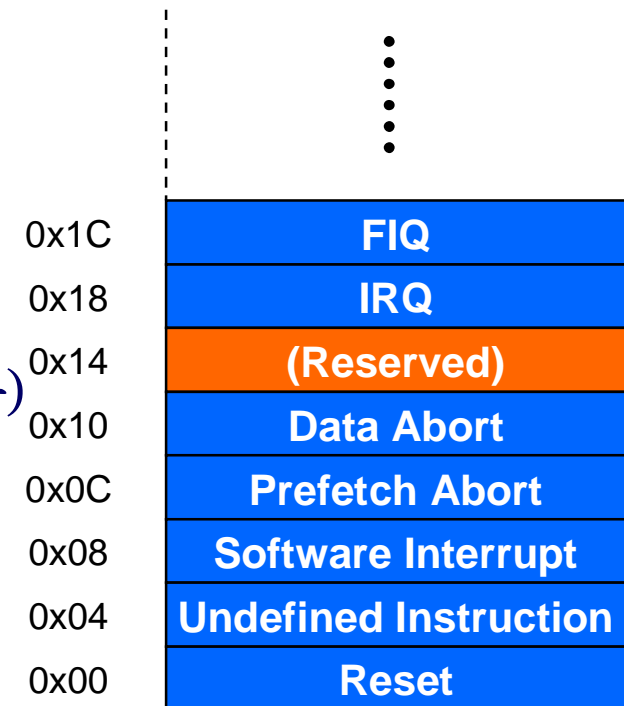
异常处理

当异常产生时，ARM core:

- 拷贝 CPSR 到 SPSR_<mode>
- 设置适当的 CPSR 位:
 - 改变处理器状态进入 ARM 态
 - 改变处理器模式进入相应的异常模式
 - 设置中断禁止位禁止相应中断 (如需要)
- 保存返回地址到 LR_<mode>
- 设置 PC 为相应的异常向量

返回时，异常处理需要:

- 从 SPSR_<mode>恢复CPSR
- 从LR_<mode>恢复PC
- Note: 这些操作只能在 ARM 态执行.



Vector Table

Vector table can be at
0xFFFF0000 on ARM720T
and on ARM9/10 family devices

进入异常的操作-2

R14_<Exception_Mode>=Return Link

SPSR_<Exception_Mode>=CPSR

CPSR[4:0]=Exception Mode Number

CPSR[5]=0 ;当运行于**ARM**状态时

IF<Exception_Mode>==Reset or FIQ then

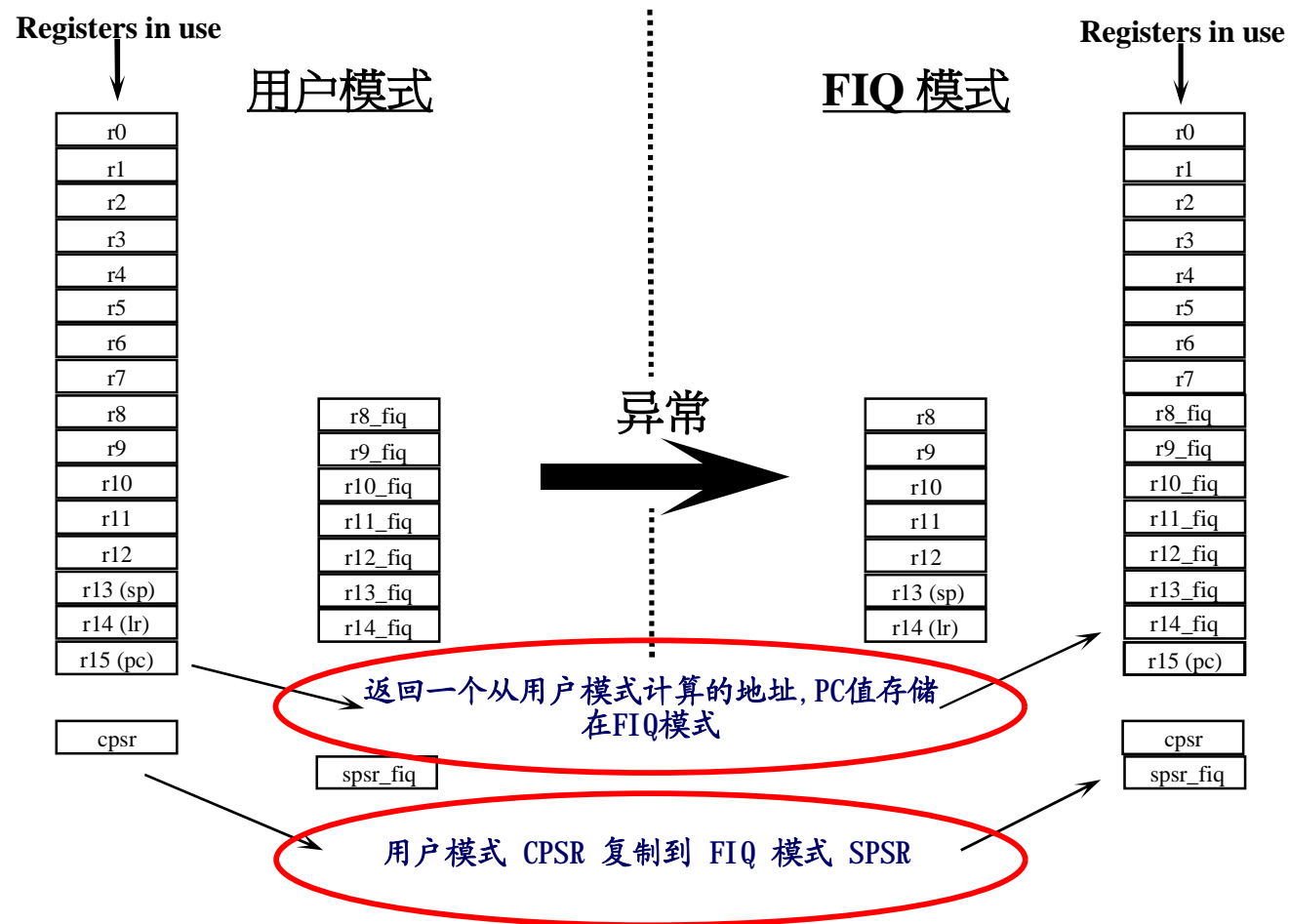
;当响应**FIQ**异常时，禁止新的**FIQ**异常

CPSR[6]=1;

CPSR[7]=1;

PC=Exception Vector Address

例子：用户模式到 FIQ 模式*



进入/退出异常概述

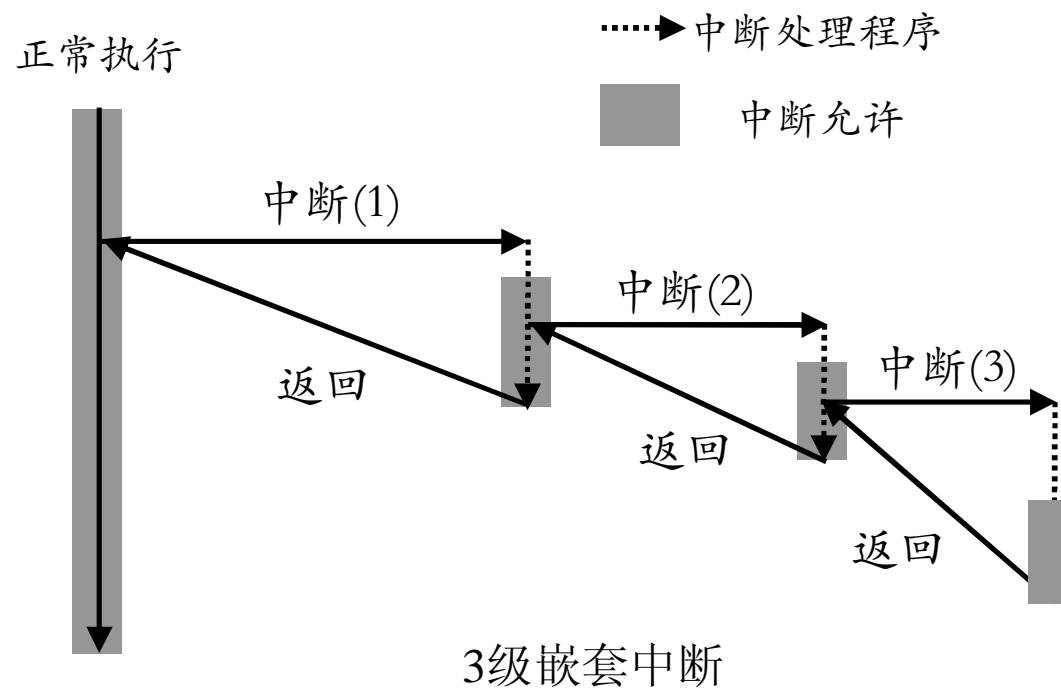
	Return Instruction	Previous State	
		ARM R14_x	THUMB R14_x
BL	MOV PC, R14	PC + 4	PC + 2
SWI	MOVS PC, R14_svc	PC + 4	PC + 2
UDEF	MOVS PC, R14_und	PC + 4	PC + 2
FIQ	SUBS PC, R14_fiq, #4	PC + 4	PC + 4
IRQ	SUBS PC, R14_irq, #4	PC + 4	PC + 4
PABT	SUBS PC, R14_abt, #4	PC + 4	PC + 4
DABT	SUBS PC, R14_abt, #8	PC + 8	PC + 8
RESET	NA	-	-

ARM的中断

- | ARM微处理器有两种中断类型：硬件中断（IRQ或FIQ）和软件中断（SWI）；
- | 软件中断SWI用于用户模式下的程序切换到特权模式；
- | IRQ中断用于通用中断，优先级低，中断延迟长；
- | FIQ中断通常为要求快速响应的个别中断源保留，如DMA传输等；

中断延迟

- 中断延迟是指从外部中断请求信号发出到取出对应的中断服务程序（ISR）的第一条指令，期间的间隔时间；
- 软件可采用两种方法减小中断延迟：嵌套中断和中断优先级。



中断优先级

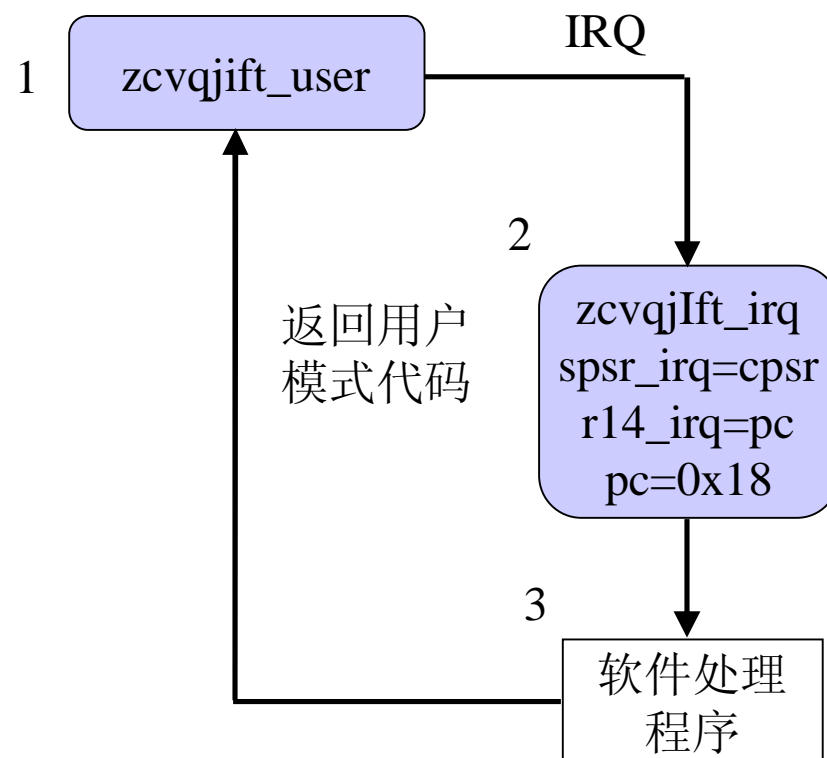
- 丨 中断优先级是对中断进行分级，高优先级的中断可以打断正在执行的中断服务；
- 丨 这种工作方式要求每个中断处理程序中重新允许中断（由进入每个中断处理程序硬件自动关中断），以保证高级别的中断处理程序能够执行；
- 丨 与低优先级中断相比，高优先级中断平均中断延迟时间少；

IRQ和FIQ异常

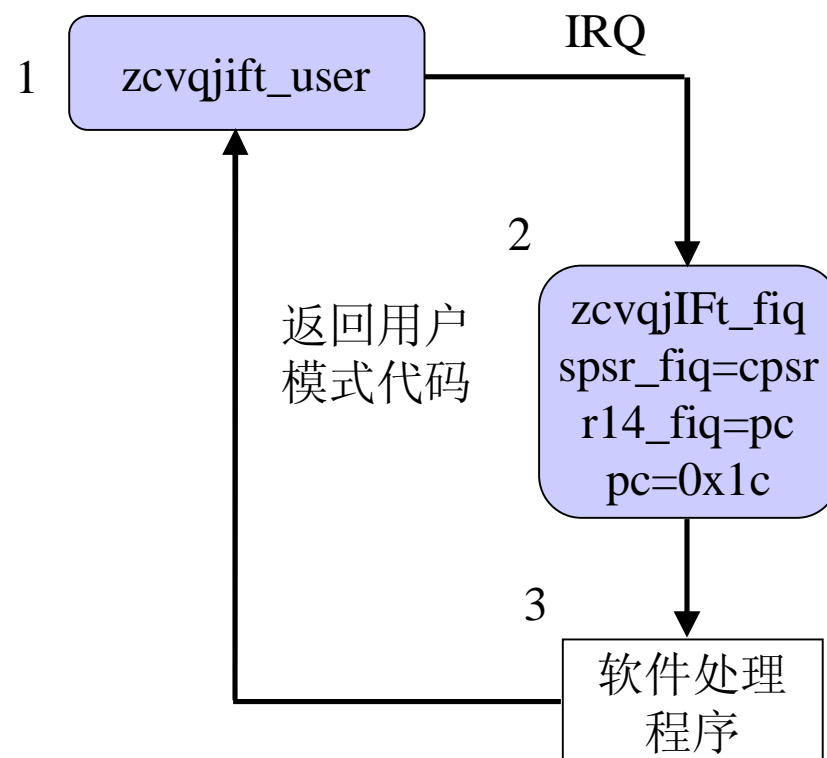
I IRQ和FIQ异常会使处理器硬件执行以下操作：

- 1) 处理器切换到一个特定的中断请求模式，表明产生了中断；
- 2) 前一个模式的CPSR被保存到新的SPSR；
- 3) PC被保存到新的中断请示模式的LR；
- 4) 关中断——在CPSR中禁止IRQ，或IRQ与FIQ都被禁止；
- 5) 处理器跳转到微量表中的一个特定的入口；

IRQ中断



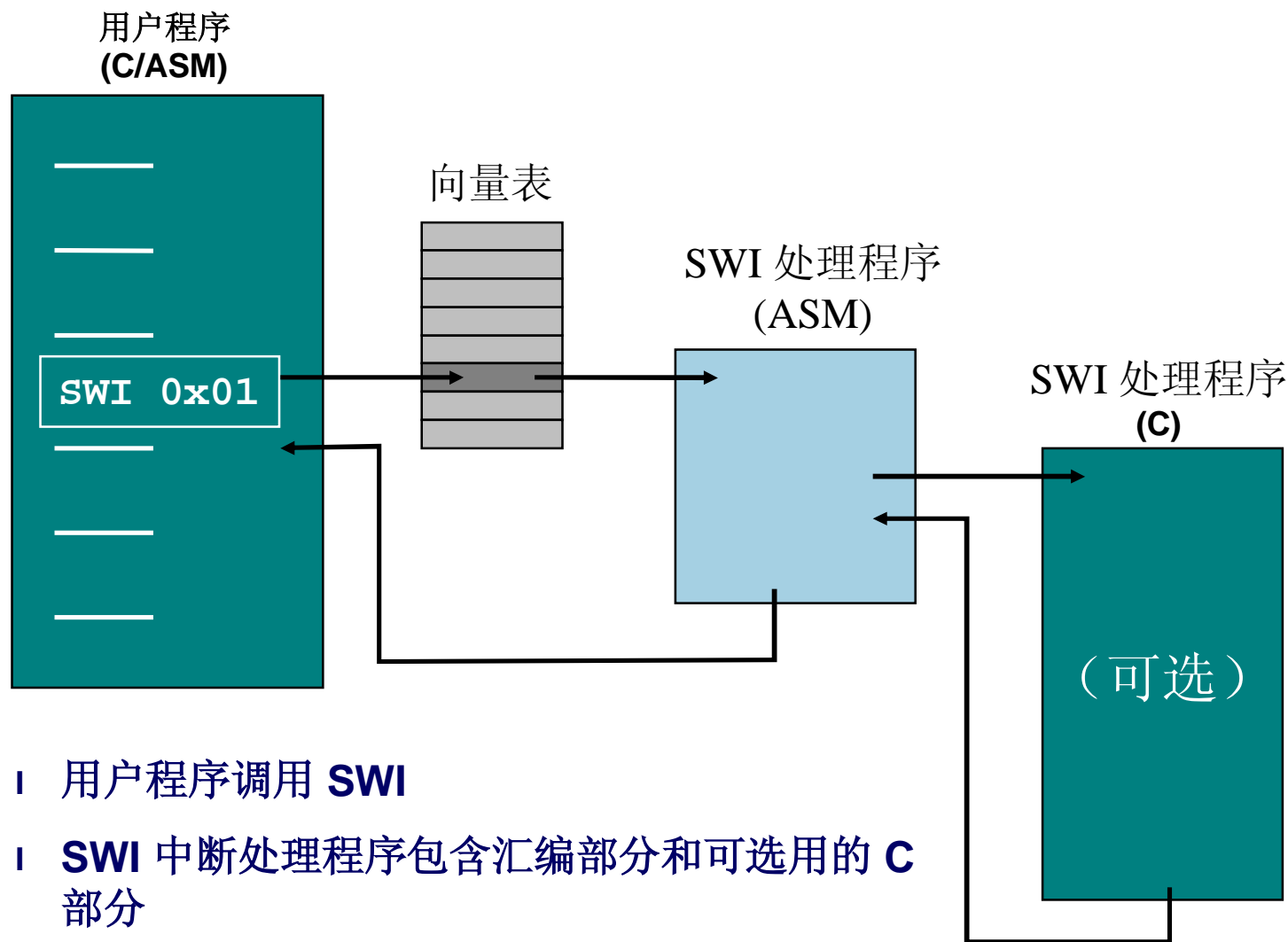
FIQ中断



FIQ vs IRQ

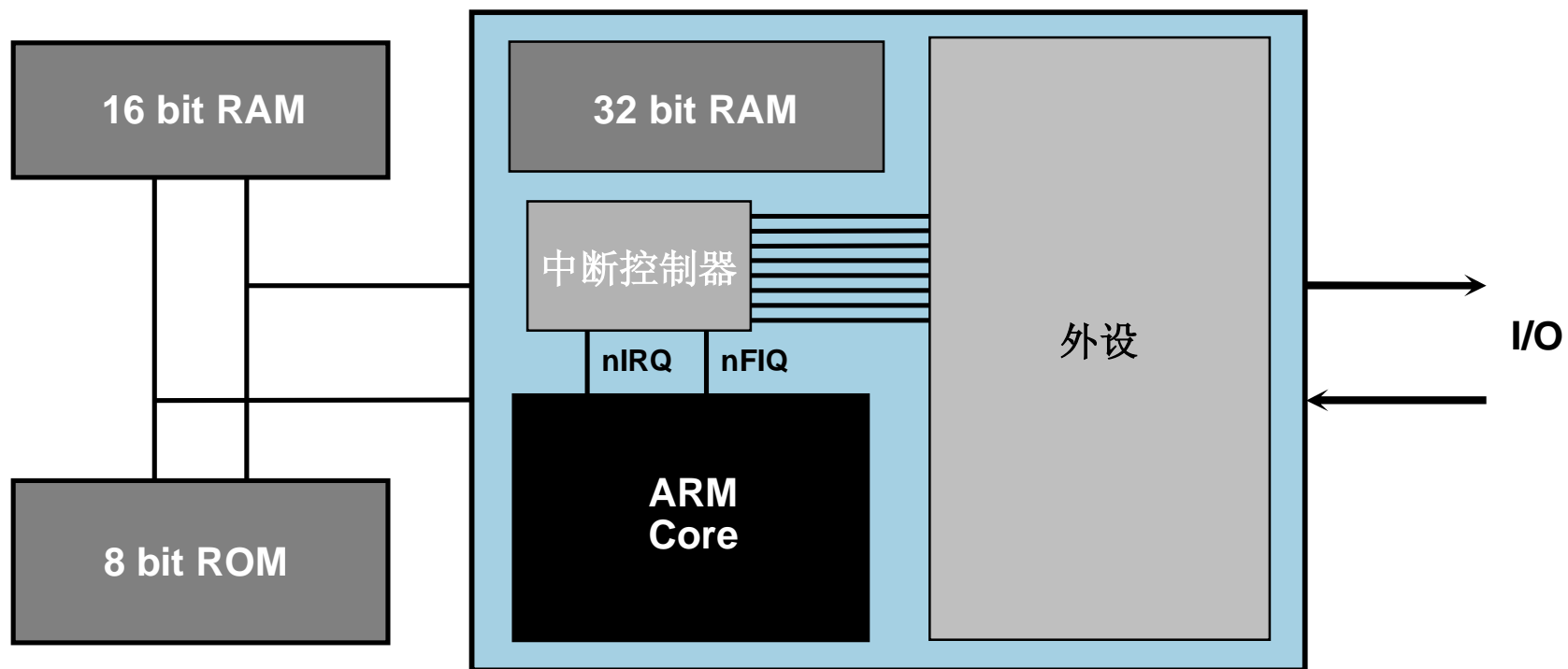
- | FIQ 和 IRQ 提供了非常基本的优先级级别。
- | 在下边两种情况下，FIQs有高于IRQs的优先级:
 - | 当多个中断产生时，FIQ高于IRQ.
 - | 处理 FIQ时禁止 IRQs.
 - | IRQs 将不会被响应直到 FIQ处理完成.
- | FIQs 的设计使中断处理尽可能的快.
 - | FIQ 向量位于中断向量表的最末.
 - | 为了使中断处理程序可从中断向量处连续执行
 - | FIQ 模式有5个额外的私有寄存器 (r8-r12)
 - | 中断处理必须保护其使用的非私有寄存器
 - | 可以有多个FIQ中断源,但是考虑到系统性能应避免嵌套。

软中断（SWI）

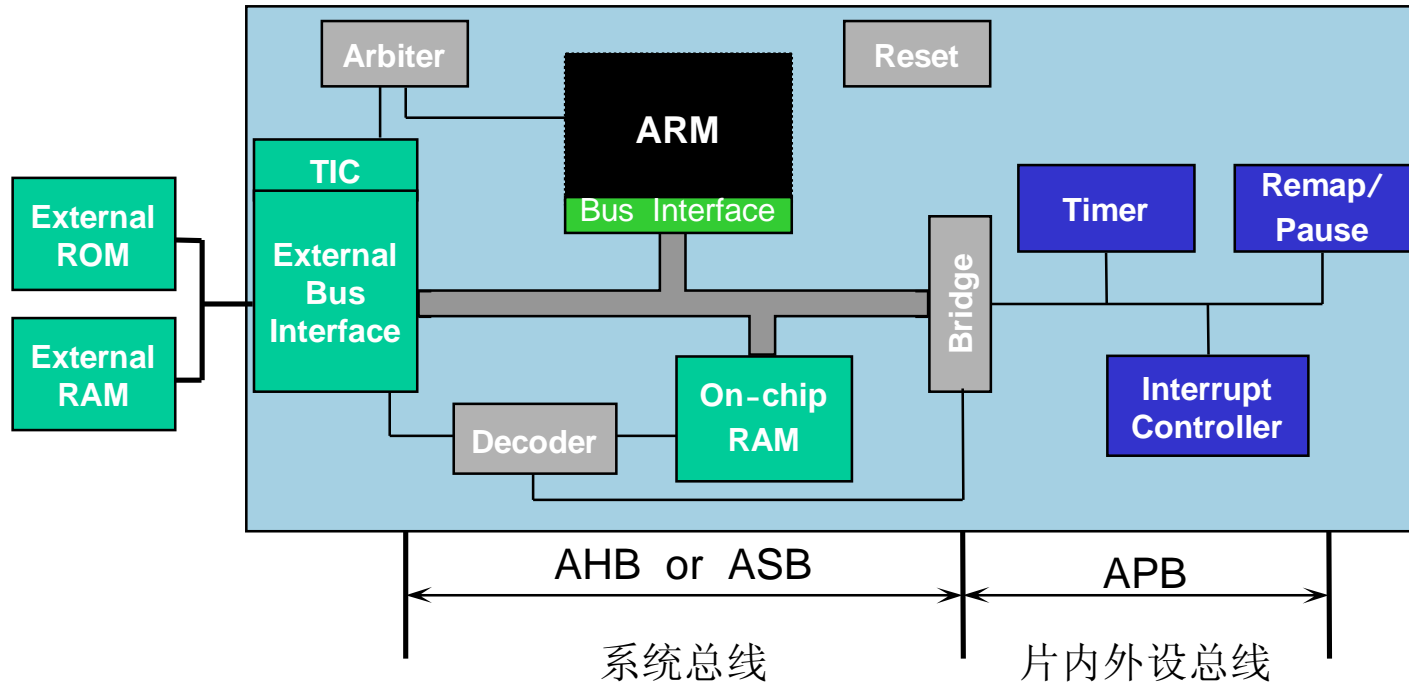


- 用户程序调用 **SWI**
- SWI** 中断处理程序包含汇编部分和可选用的 **C** 部分

基于ARM的系统示例



AMBA总线



I AMBA

I Advanced Microcontroller Bus Architecture

I ADK

I Complete AMBA Design Kit

I ACT

I AMBA Compliance Testbench

I PrimeCell

I ARM's AMBA compliant peripherals

谢谢各位