普通高等教育“十三五”规划教材
电子信息科学与工程类专业规划教材

Cortex-A8
原理、实践及应用

◎ 姜余祥　杨　萍　邹　莹　编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

电子信息科学与工程类专业规划教材

Cortex-A8 原理、实践及应用

姜余祥　杨　萍　邹　莹　编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

作为一款 32 位高性能、低成本的嵌入式 RISC 微处理器，Cortex-A8 目前已经成为应用广泛的嵌入式处理器。本书在全面介绍 Cortex-A8 处理器的体系结构、编程模型、指令系统及开发环境的同时，基于 Cortex-A8 应用处理器——S5PV210 为核心应用板，详细阐述了其外围接口技术、U-Boot 启动流程及其移植技术、Linux 裁剪和移植技术、驱动程序的编程技术和 Qt 的应用编程技术，并提供了在物联网中的应用工程案例。书中所涉及的技术领域均提供实验工程源代码，便于读者了解和学习。

本书可作为高等院校电子类、通信类、自动化类和计算机类等各专业“嵌入式应用系统”课程的教材，也可供从事嵌入式应用系统开发的工程技术人员参考。

图书在版编目（CIP）数据

Cortex-A8 原理、实践及应用 / 姜余祥，杨萍，邹莹编著. —北京：电子工业出版社，2018.1
电子信息科学与工程类专业规划教材

ISBN 978-7-121-33306-4

Ⅰ. ①C… Ⅱ. ①姜… ②杨… ③邹… Ⅲ. ①微处理器—系统设计—高等学校—教材 Ⅳ. ①TP332

中国版本图书馆 CIP 数据核字（2017）第 311546 号

责任编辑：凌　毅
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：18.5　字数：498 千字
版　　次：2018 年 1 月第 1 版
印　　次：2018 年 1 月第 1 次印刷
定　　价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254528，lingyi@phei.com.cn。

前　言

随着手持类设备的普及，嵌入式应用技术得到了快速发展。嵌入式应用系统由三层结构组成，分别为硬件层、系统层和应用层。其中，硬件层主要涉及 CPU 的选型及板级电路的设计；系统层主要涉及操作系统的移植及驱动程序的设计，通过抽象过程完成硬件层与应用层的隔离；应用层建立在系统层之上，主要完成用户应用程序的编写和调试。

针对三层结构的特点，本书以嵌入式应用系统设计过程为主干线，按照系统设计流程组织教材的框架结构。主要包含嵌入式 CPU 的组成结构和接口电路设计，BootLoader 的定制，Linux 操作系统的裁剪和移植，Yaffs 文件系统的定制，驱动案例的设计，物联网应用系统工程案例的设计，全书共分 8 章。

第 1 章 Cortex-A8 处理器。作为嵌入式应用系统的关键组成部分，Cortex-A8 处理器已经被广泛应用于移动终端、掌上电脑以及其他消费电子设备。本章介绍了 Cortex-A8 处理器的内部结构和各组成部分功能。

第 2 章 汇编语言。本章侧重于 Linux 环境下的应用，介绍了 ARM 汇编语言指令集、GNU ARM 汇编器汇编命令以及汇编语言程序设计基础。

第 3 章 S5PV210 概述。本章主要讲述 S5PV210 芯片的存储结构、寄存器结构和 GPIO 结构。以 UART 为例介绍了 S5PV210 内部接口控制器的使用方法，并介绍了该芯片上电复位后的启动流程，在案例一节中介绍了基于 S5PV210 裸机应用程序的开发过程。

第 4 章 U-Boot。基于 Cortex-A8 硬件平台运行的嵌入式 Linux 系统软件平台可以分为 4 个部分：①引导加载程序（BootLoader），依赖于所运行的硬件平台；②Linux 内核，依据应用需求，需要通过裁剪和移植完成内核的定制；③文件系统，包括根文件系统和 Yaffs 文件系统；④嵌入式 GUI 和用户应用程序。本章在基于 S5PV210 微处理器的硬件平台上，分析了 U-Boot 启动流程。在使用 U-Boot 引导嵌入式 Linux 操作系统的过程中，通过工程案例详细介绍了在指定硬件和软件平台条件下，完成 U-Boot 的定制过程。

第 5 章 Linux 内核移植。本章主要介绍了嵌入式 Linux 系统的构建过程。通过学习本章内容，能够对嵌入式 Linux 系统的结构有一个清晰认识，并掌握基于 Tiny210 硬件平台的嵌入式 Linux 操作系统搭建过程。

第 6 章 嵌入式 Linux 程序设计。嵌入式硬件设备需要专用的驱动程序，驱动程序需要通过特定的方法和步骤添加到嵌入式操作系统中，应用层需要编写程序调用驱动程序才能完成对系统硬件的操作。本章介绍了基于 ARM-Linux 驱动程序的开发、驱动程序的加载方法和基于驱动程序的应用程序开发。

第 7 章 图形用户接口 Qt。Qt 是一个基于 C++图形用户界面的应用程序开发框架，本章首先介绍了 Qt 应用程序的开发环境，随后以案例形式介绍基于嵌入式硬件平台的 Qt 应用程序编写方法。

第 8 章 嵌入式物联网应用系统设计。本章通过实际案例介绍基于 Cortex-A8 微处理器的嵌入式应用系统设计，主要涉及智能家居、物联网应用云平台搭建和访问等领域。

书中涉及 Windows 和 Linux 两个操作系统，在描述两个系统中的文件路径时，使用“\”符号表示 Windows 环境下的文件路径，使用“/”符号表示 Linux 环境下的文件路径。

本书提供配套的电子课件及相关配套资源，主要包括：教学课件 PPT 和实验指导书，嵌入式系统开发过程中常用到的软件工具包，各章案例的程序源代码，本书所使用的硬件平台软件系统文件以及 Cortex-A8 系统更新和系统文件烧写说明。

书中各章节提供了大量工程案例，其中实践部分内容依托于北京赛佰特科技有限公司的 CBT-Super IOT 型全功能物联网教学科研实验平台，唐冬冬为本书提供了丰富的软硬件资料及技术支持。

本书应用例程和教学参考讲义，请读者到华信教育资源网注册后免费下载（www.hxedu.com.cn）。

本书可作为高等院校电子类、通信类、自动化类和计算机类等各专业“嵌入式应用系统”课程的教材，也可供从事嵌入式应用系统开发工程技术人员参考。

本书主要由姜余祥、杨萍和邹莹编写。其中，第 1、2、3、4 章由姜余祥编写，第 5、6、7 章由杨萍编写，第 8 章由邹莹编写。胡字滢、李晓峰参与了本书的校对以及配套电子课件和实验指导书的编写。李强和赵永永同学对本书所提供的工程案例中的程序进行了调试和整理工作。

本书在编写过程中，感谢电子工业出版社工作人员的大力支持，尤其要感谢我的家人，是她们多年来的理解、帮助和支持，才能够完成本书的撰写工作。

在此向所有关心和支持本书编写工作的人士表示衷心的感谢。

由于目前嵌入式应用领域的迅速发展，且作者的实际工作经验及水平有限，书中会有许多不足之处，望读者不吝指正。

姜余祥

2017 年 12 月

目　录

第 1 章　Cortex-A8 处理器

Cortex-A8 是一款成功的 ARM 内核。作为嵌入式应用系统的关键组成部分，目前已被广泛应用于移动终端、掌上电脑和许多其他日常便携式消费电子设备。

本章主要内容：

（1）Cortex-A8 处理器内部组成结构；

（2）Cortex-A8 处理器的编程模型；

（3）Cortex-A8 处理器的时钟、复位和电源控制等功能单元的管理机制。

通过本章的介绍，对 Cortex-A8 处理器内部结构和各部分功能可以有一个初步了解，为后续章节的学习打下基础。

1.1　概　　述

Cortex-A8 处理器是一款高性能、低功耗、完整虚拟内存管理能力并具有高速缓存的应用处理器。

1．处理器特性

（1）支持 ARM 体系结构的 v7-A 指令集。

（2）使用内部 AXI（Advanced Extensible Interface）接口，可将主存配置为 64 位或 128 位高速 AMBA（Advanced Microprocessor Bus Architecture）模式，来支持多种数据传输行为。

（3）一条用于执行整数指令的流水线。

（4）一条用于执行 SIMD 和 VFP 指令集的 NEON 流水线，为多媒体应用提供硬件加速。

（5）使用分支目标地址缓存和全局历史缓冲区实现动态分支预测。

（6）MMU（Memory Management Unit）。用于内存单元管理。

（7）L1 级指令缓存和数据缓存可配置为 16KB 或 32KB。

（8）L2 缓存大小可配置为 0KB 或 128KB～1MB。

（9）L2 缓存可配置奇偶校验模式和纠错码（Error Correction Code，ECC）。

（10）嵌入式跟踪宏单元（Embedded Trace Macrocell，ETM）支持在线调试。

（11）动态和静态的电源管理方案（Intelligent Energy Management，IEM）。

（12）用于调试的探针和断点寄存器，支持片上调试。

2．ARMv7 架构指令集

（1）采用 ARM Thumb-2 指令集，兼容 Thumb 和 ARM 指令。

（2）使用 Thumb-2（Thumb-2EE）技术，提高运行速度。

（3）增强型的安全功能（NEON），有利于安全应用程序的开发。

（4）高级 SIMD（Single Instruction Multiple Data）架构扩展，用于多媒体三维图形和图像处理的加速技术。

（5）用于浮点计算的向量浮点 v3（Vector Floating Point v3，VFPv3）架构，兼容 IEEE 754 标准。

1.2 处理器组成结构

Cortex-A8 处理器的主要组成部分包含：指令取指单元、指令解码单元、指令执行单元、Load/Store 单元、L2 缓存（上述 5 部分统称为内核）、NEON 和 ETM 单元。其内部结构如图 1.1 所示。

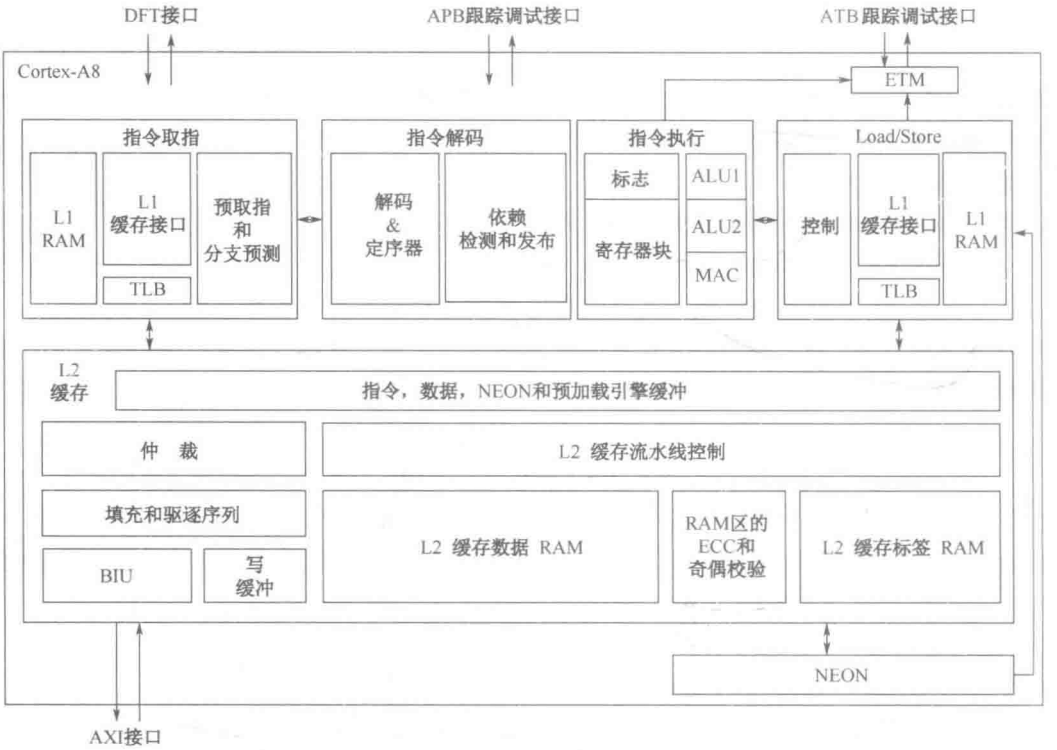

图 1.1 Cortex-A8 处理器结构图

1.2.1 内部功能单元

1. 指令取指单元

指令取指单元负责完成指令流预测，从 L1 指令缓存中取出指令代码，并将所获取指令送到解码流水线进行解码。指令取指单元还包括 L1 指令缓存。

2. 指令解码单元

指令解码单元依次对 ARM 和 Thumb-2 指令完成解码工作。这些指令可以来自调试控制协处理器（CP14）、指令和系统控制协处理器（CP15）。

当存在多条指令需要解码时，指令解码单元的执行顺序是：异常、调试事件、复位初始化、内存内置自测试（MBIST）、等待中断处理和其他异常事件。

3. 指令执行单元

指令执行单元包含两条对称的算术逻辑单元（ALU）流水线、一个用于加载和存储指令的地址发生器和多条专用功能流水线，其中执行流水线同时负责寄存器回写。

指令执行单元的作用：

（1）执行所有整数 ALU 和乘法运算指令，包括产生标志位；

（2）为 Load/Store 指令生成虚拟地址和基地址；

（3）为 Load/Store 指令提供格式化数据；

（4）处理分支和其他变化的指令流，评估指令条件码。

4. Load/Store 单元

Load/Store 单元含有完整的 L1 数据存储系统和整数 Load/Store 指令流水线，包括 L1 数据缓存、数据 TLB、整数存储缓冲区、NEON 存储缓冲区、整数数据加载地址边沿对齐和格式化以及整数数据存储数据地址边沿对齐和格式化。

5. L2 缓存

L2 高速缓存单元包括 L2 缓存和缓冲接口单元（BIU）。它服务于被 L1 缓存错过的预取指和 Load/Store 指令部分。

6. NEON

NEON 单元包括完整 10 条流水线，用于解码和执行 SIMD 媒体指令集。NEON 单元包括 NEON 指令队列、NEON 加载数据队列、两条 NEON 解码逻辑流水线、三条用于 SIMD 整数指令的执行流水线、两条用于 SIMD 浮点指令的执行流水线、一条用于 SIMD 和 VFP 的 Load/Store 指令执行流水线和完全执行 VFPv3 的数据处理指令集 VFP 引擎。

7. 非侵入式跟踪宏单元（ETM）

ETM 过滤和压缩那些用于系统调试和系统分析而需要跟踪的指令和数据。ETM 单元在处理器外部有一个专用 ATB 接口。

1.2.2 处理器外部接口

处理器含有以下扩展接口与外部相连：与 AMBA 总线相连的 AXI 接口、APB 跟踪调试接口、ATB 跟踪调试接口和 DFT 接口。

（1）AXI 接口是系统总线主要接口，为指令和数据完成 L2 缓存的填充和非缓存类访问。AXI 接口支持 64 位或 128 位的输入和输出数据总线宽度。AXI 总线上支持多个未处理请求。AXI 信号与时钟输入同步，使用时钟允许信号 ACLKEN 可以获得一个较大总线占用比。

（2）APB 跟踪调试接口可以访问 ETM、CTI 和用于寄存器的调试。

（3）ATB 跟踪调试接口可以输出跟踪信息用于调试。

（4）DFT 接口为制造商提供内存内置自测试和自动测试方案。

1.2.3 可配置的操作

对于内部单元，Cortex-A8 处理器提供了灵活的配置方案。处理器内核可配置选项参见表 1.1。

表 1.1 内核可配置选项

状 态	可 配 置 项
AXI 总线宽度	64bit/128bit
L1 RAM 容量	16KB/32KB
L2 RAM 容量	0KB/128KB/256KB/512KB/1MB
L2 Parity/ECC 校验	Yes/No
ETM	Yes/No
IEM	支持内核所有组成部分的电源域管理
NEON	Yes/No。当处理器配置为 No 模式时，所有 SIMD 和 VFP 的指令将尝试进入未定义模式

1.3 编 程 模 型

Cortex-A8 处理器通过设置可以工作在 ARM 或 Thumb 状态。工作在 ARM 状态时，支持 32 位 ARM 指令集；工作在 Thumb 状态时，支持 16/32 位的 Thumb 指令集。通过实现 SIMD 架构，

增加了主要针对音频、视频、3D 图形、图像和语音处理的指令。在介绍指令集前，需要了解处理器的编程模型（Programmer's Model）。

1.3.1 内核数据流模型

Cortex-A8 内核采用冯·诺依曼结构，这是一种将程序指令存储器和数据存储器合并在一起的存储器结构。程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置，由于公用数据总线，因此程序指令总线和数据总线的宽度相同，总线宽度为 32bit。

Cortex-A8 内核支持的 ARM 指令集为 RISC 集。其特点是所有数据都需要在内核的寄存器中处理，不支持在内存中直接操作数据。通过 Load/Store 结构，使用 load 指令将数据从内存中复制到寄存器中，使用 store 指令将寄存器中的数据复制到内存中。

从编程角度来看，ARM 内核由指令解码器（Instruction Decoder）、桶形移位器（Barrel Shifter）、符号扩展单元（Sign Extend）、R 寄存器组（Register File）、ALU（算术逻辑单元）、MAC（乘累加单元）、地址寄存器（Address Register）、地址加法器（Incrementer）等部分组成。各部分通过内部数据总线相连，其连接结构如图1.2(参考来自 ARM System Developer's Guide)所示。

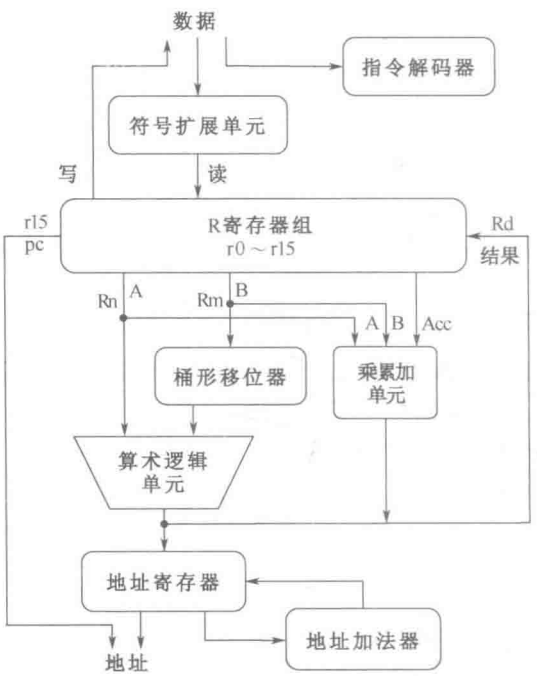

图 1.2 ARM 内核数据流模型

符号扩展单元：存储单元中定义的有符号 8/16 位数据类型的数据在这里被扩展到 32 位后送入寄存器。

R 寄存器组：由 16 个 32 位寄存器组成。可以提供 Rn 和 Rm 两个寄存器，用于存储 ALU 或 MAC 的源操作数；一个 Rd 寄存器，用于记录操作结果（目的操作数）。其中，Rm 支持算术逻辑运算前的移位操作，可用于位操作运算或灵活的寻址方案。指定 r15 寄存器作为 PC（程序计数器寄存器），它为地址总线提供指令代码存放位置的 32 位地址信息。

算术逻辑单元（Arithmetic Logic Unit）：针对来自 Rn 和 Rm（或 N）两个寄存器的源操作数，实现指令规定的运算。运算结果若是数据则送到 Rd 寄存器中，若是地址则送到地址寄存器中，并广播到地址总线中。

乘累加单元（Multiply-Accumulate Unit）：实现硬件乘法或乘-加运算。

地址加法器：当处理器使用 Load/Store 指令访问连续的存储空间时，可以通过地址加法器得到下一个存储单元地址。

1.3.2 工作模式

Cortex-A8 处理器有 8 种工作模式。

（1）系统模式（System）。操作系统的特权用户模式。

（2）用户模式（User）。ARM 程序的正常工作模式，并用于执行大多数应用程序。该模式下，限制内存的直接访问和通过物理地址对硬件设备的读/写操作。

（3）快速中断模式（FIQ）。用于处理快速中断。

（4）中断模式（IRQ）。用于通用中断处理。

（5）超级用户模式（SVC）。ARM 内核上电时处于 SVC 模式，主要用于 SWI（软件中断）和受保护的操作系统模式。

（6）中止模式（Abort）。数据中止或预取中止的异常事件发生后进入中止模式。

（7）未定义模式（Undefined）。当执行一个未定义指令的异常事件发生时，进入未定义模式。

（8）监视模式（Monitor）。是一种安全模式，用于执行安全监视代码。

除用户模式以外的 7 种模式称为特权模式。特权模式用于服务中断或异常，或访问受保护资源，具有对 CPSR 寄存器的读/写控制权。

1.3.3 寄存器结构

Cortex-A8 处理器有 40 个 32 位寄存器，分为 33 个通用寄存器和 7 个程序状态寄存器。ARM 正常状态下有 16 个数据寄存器和 1～2 个状态寄存器可随时被访问，可被访问的寄存器依赖于处理器当前的工作模式。每种工作模式下处理器可访问的寄存器名称见表 1.2。

表 1.2　基于 ARM 工作模式的寄存器分组

系　统	用　户	快速中断	中　断	超级用户	中　止	未定义	监　视
r0	r0	r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7	r7	r7
r8	r8	r8_fiq	r8	r8	r8	r8	r8
r9	r9	r9_fiq	r9	r9	r9	r9	r9
r10	r10	r10_fiq	r10	r10	r10	r10	r10
r11	r11	r11_fiq	r11	r11	r11	r11	r11
r12	r12	r12_fiq	r12	r12	r12	r12	r12
r13(SP)	r13(SP)	r13_fiq	r13_irq	r13_svc	r13_abt	r13_und	r13_mon
r14(LR)	r14(LR)	r14_fiq	r14_irq	r14_svc	r14_abt	r14_und	r14_mon
r15(PC)	PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_fiq	SPSR_irq	SPSR_svc	SPSR_abt	SPSR_und	SPSR_mon

下面依照寄存器的功能进行划分。

r0～r12 是通用寄存器，可用来保存数据项或表示地址信息的数值。其中，快速中断工作模式具有私有的 r8～r12 寄存器。

r13 是堆栈指针寄存器（Stack Pointer，SP），用于指向堆栈区的栈顶。每种工作模式具有各自私有的堆栈区和堆栈指针寄存器。

r14 是链接寄存器（Link Register，LR），用于存储子程序返回主程序的链接地址。当处理器执行一条调用指令（BL 或 BLX）时，r14 用于存储主程序的断点地址，供子程序返回主程序；其他时间，r14 可以作为一个通用寄存器使用。每种工作模式具有各自私有的链接寄存器。

r15 是程序计数器（Program Counter，PC），用于存放下一条指令所在存储单元的地址。由于 ARM 指令集中的一条指令代码为 4 字节，因此在取指时指令代码的存储地址应满足字对齐，即 r15[1:0]=b00。

CPSR 是当前程序状态寄存器（Current Program Status Register，CPSR）。寄存器中包含条件码标志、状态位和当前工作模式位。除了系统模式外，其他每种特权模式额外拥有一个备份程序状态寄存器（Saved Program Status Register，SPSR），用来保存当异常发生需要进入其他模式时所产生的程序断点处状态信息。通常是将断点的 CPSR 内容复制到 SPSR 中，以便在异常处理程序结束并返回时恢复断点处的程序状态。

8 种工作模式都有各自的 16 个数据寄存器和状态寄存器。依照每种模式的特点，这些寄存器又分为公有和私有两种情况。

公有寄存器：如程序计数器寄存器，8 种工作模式公用一个 r15（PC）。

私有寄存器：8 种工作模式拥有各自独立的寄存器，如堆栈指针 r13（SP）。在寄存器命名过程中，使用添加表示工作模式下标的方法加以区分，如 r13_fiq。此时虽然是表示堆栈指针寄存器（各种模式的此功能寄存器，在编译环境下同为 r13），但是快速中断模式与其他模式有不同的物理空间。在使用过程中，需要通过设置 CPSR 中的模式位来表明处理器当前的工作模式，以确认当前模式下寄存器组的物理位置。8 种工作模式的私有寄存器可参见表 1.2 中加阴影单元中的命名。

1.3.4 程序状态寄存器

Cortex-A8 处理器的程序状态寄存器包含 1 个主程序使用的 CPSR 寄存器和 6 个异常处理程序使用的 SPSR 寄存器。

程序状态寄存器（CPSR 和 SPSR）的主要用途：保存所执行的最后一条逻辑或算术运算指令运行结果的相关信息，控制开启/禁用中断，设置处理器工作模式。程序状态寄存器位定义见表 1.3。

表 1.3 程序状态寄存器（CPSR 和 SPSR）位定义

位	31	30	29	28	27	26～25	24	23～20	19～16	15～10	9	8	7	6	5	4～0
定义	N	Z	C	V	Q	IT[1:0]	J	DNM	GE[3:0]	IT[7:2]	E	A	I	F	T	M[4:0]

表 1.3 中，“位”一行表示了组成程序状态寄存器的 32 位二进制数排序位置。最右边的规定为第 0 位，依次向左排序，最左边的为程序状态寄存器的最高位第 31 位。“定义”一行表示了程序状态寄存器中一位二进制数所代表的含义。理解位定义的概念要注意两方面：位在寄存器中的排序位置，位的内容（1/0）所代表意义。

在描述位定义过程中，其位置通常使用其所代表意义的英文缩写字母来表示。例如，程序状态寄存器中第 30 位的内容用来表示当前算术运算指令运行结果是否为零，为了便于描述，将该位定义为“Z”标志位。

1．条件代码标志位

N（CPSR[31]）：负数或小于标志位。N=1 表示当前指令运行结果为负数（非正数或零）。

Z（CPSR[30]）：零标志位。Z=1 表示当前指令运行结果为零，常表示两个操作数相等。

C（CPSR[29]）：进位/借位/延伸标志位。C=1 表示两个无符号数相加结果溢出。

V（CPSR[28]）：溢出标志位。V=1 表示两个有符号数相加结果溢出。

Q（CPSR[27]）：若执行乘法和分数算术运算指令（QADD、QDADD、QSUB、QDSUB、SMLAD、SMLAxy、SMLAWy、SMLSD、SMUAD、SSAT、SSAT16、USAT、USAT16）后发生溢出，则 Q 标志位被置 1。此时需要执行 MSR 指令才可清除 Q 标志位，否则后续乘法和小数运算指令不