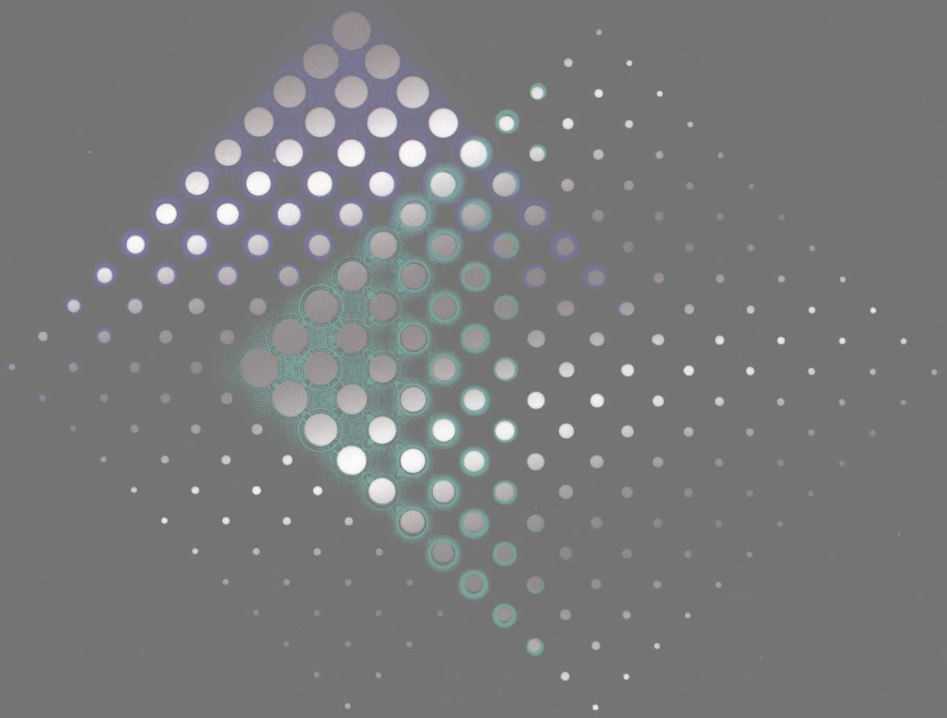
新编计算机类本科规划教材

嵌入式系统开发基础
——基于ARM微处理器和Linux操作系统

滕英岩 主编　窦 乔 孙建梅 副主编



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

新编计算机类本科规划教材

嵌入式系统开发基础

——基于ARM微处理器和Linux操作系统

滕英岩　主编

窦　乔　孙建梅　副主编

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

内 容 简 介

本书主要介绍嵌入式系统的软件开发技术，内容基于 ARM 微处理器和 Linux 操作系统。本书主要内容包括：嵌入式系统基础知识、嵌入式交叉编译环境、嵌入式开发环境的搭建、MiniGUI 应用程序设计、嵌入式数据库、Qt 图形界面应用程序开发。

本书涉及嵌入式系统从底层驱动到顶层应用的各个部分，配合实验操作循序渐进地帮助读者完成各个章节内容的学习，引领初学者顺利进入嵌入式世界。

本书适合作为高等院校计算机、电子信息等专业嵌入式方向的教材，也可作为嵌入式领域科研人员的参考书。

图书在版编目（CIP）数据

嵌入式系统开发基础：基于 ARM 微处理器和 Linux 操作系统 / 滕英岩主编.

北京：电子工业出版社，2008.10

（新编计算机类本科规划教材）

ISBN 978-7-121-07425-7

Ⅰ. 嵌… Ⅱ. 滕… Ⅲ. 微型计算机－系统开发－高等学校－教材 Ⅳ. TP360.21

中国版本图书馆 CIP 数据核字（2008）第 147356 号

策划编辑：张 濮

责任编辑：侯丽平

印 刷：北京市海淀区四季青印刷厂

装 订：涿州市桃园装订有限公司

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：14.5 字数：371.2 千字

印 次：2008 年 10 月第 1 次印刷

印 数：4000 册 定价：25.00 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。

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。

服务热线：(010)88258888。

前言

近年来，我国嵌入式软件发展迅猛，已成为中国软件产业新的市场增长点。据赛迪顾问2007年9月的数据，2007年嵌入式软件市场规模达到1938.1亿元人民币，增长率为32.6%，到2008年将达到2496.2亿元人民币，增长率为28.8%。嵌入式软件已经成为数字化产品设计创新和软件增值的关键因素，是未来市场竞争力的重要体现。与巨大的市场潜力和产业需求相比，嵌入式软件专业人才匮乏，严重制约着中国未来嵌入式软件产业的发展。

目前，嵌入式人才培养的渠道主要有两个：一个是嵌入式培训机构，另一个是高等院校。前者的培养对象大多数有一定的专业基础或者其他软件从业经验，而后者的培养对象大多数都是零起点。但无论哪种培养方式，都将面临一个问题：如何循序渐进地引导初学者踏入嵌入式系统开发的神秘殿堂？在这一过程中，作为体现教学内容和教学方式的载体——教材，其重要性不言而喻。

基于对上述背景的认识，我们在进行嵌入式人才培养的过程中，不断总结和积累嵌入式方向的教学内容和手段，把几年来的教学经验总结为一本嵌入式软件开发的基础教材。本着为初学者服务的宗旨，本书在撰写过程中力求通俗易懂，从基础知识入手，介绍嵌入式软件的基本理论及其搭建过程，并配有相应的实验环节，力争结合具体的实验设备，达到理论和实践的结合。

本书共分6章，各章主要内容如下。

第1章介绍嵌入式系统的软硬件结构，重点介绍了嵌入式微处理器ARM和嵌入式操作系统的基础知识。

第2章介绍如何在Linux环境下搭建嵌入式交叉编译环境。

第3章以ARM微处理器为硬件平台，介绍基于Linux的嵌入式软件开发流程，包括建立开发软件环境、移植BootLoader程序、裁减和编译内核、构建根文件系统、开发简单驱动程序等。

第4章以MiniGUI为例，介绍嵌入式GUI的程序设计方法。

第5章介绍嵌入式数据库SQLite3的特点及应用。

第6章介绍了嵌入式图形用户界面Qt4及Qtopia-Core的特点及应用。

本书从基础知识开始，循序渐进地引领读者进入嵌入式世界，同时结合大量实例使读者掌握嵌入式软件的基本开发技术。每章配有针对性很强的习题，通过这些习题可以帮助读者巩固每章的知识点。

为了便于教师授课，本书提供了每章的电子课件，这些电子课件也融合了我们嵌入式人才培养的方法和手段，读者可以访问电子工业出版社华信教育资源网（www.hxedu.com.cn）注册下载。本书还提供了习题答案、辅助案例等资源，授课教师可与作者联系获取，电子邮件地址为tengyingyan@neusoft.edu.cn。

全书由滕英岩、窦乔、孙建梅共同负责编写。其中，滕英岩编写了第 1、2 章，窦乔编写了第 3、4 章，孙建梅编写了第 5、6 章。另外，张福艳参加了第 1 章的编写，滕英岩参加了第 3、6 章的编写。全书由滕英岩统稿。

作者希望通过本书能为初学者开启通往嵌入式世界的大门，但因水平有限，错误或不妥之处在所难免，敬请读者提出宝贵意见。

作 者

2008 年 8 月

目 录

第1章 嵌入式系统基础知识

随着信息技术的高速发展，嵌入式技术的应用已经渗透到人们的工作、生活中，如家用电器、手持通信设备、信息终端、仪器仪表、汽车电子、航天航空、军事装备、制造工业等，各种形式的嵌入式产品由于其软硬件可裁减的特点使它们已经成为信息化时代市场中的主流。嵌入式技术具有广阔的应用前景，嵌入式产品无处不在，它将为人类生产带来革命性的发展，实现“PCs Everywhere”的生活梦想。

嵌入式系统是以应用为中心，以计算机技术为基础，其软硬件可裁减配置，对功能、可靠性、成本、体积、功耗有严格约束的一种专用计算机系统。嵌入式系统一般指非 PC 系统，包括硬件和软件。硬件是整个系统的物理基础，它提供软件运行的平台和通信接口。硬件包括微处理器、存储器、外围器件、I/O 端口和图形控制器等。软件由操作系统和其上运行的应用程序构成，控制系统的运行。嵌入式系统的操作系统和应用程序是紧密结合的，所以有时将其组合在一起不做区分。

本章主要内容包括：

- 嵌入式系统概述
- 嵌入式微处理器 ARM
- 嵌入式操作系统
- 典型的嵌入式开发平台

1.1 嵌入式系统的特点及分类

1.1.1 嵌入式系统的特点

根据嵌入式系统的定义，可以看出嵌入式系统具有以下特点：

（1）由于嵌入式系统采用的是微处理器，独立的操作系统，实现相对单一的功能，所以往往不需要大量的外围器件，因而在体积、功耗上有其自身的优势。

（2）嵌入式系统由于空间和各种资源相对不足，硬件和软件都必须高效率地设计，力争在同样的硅片面积上实现更高的性能。

（3）嵌入式系统为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存储于磁盘等载体中。

（4）为适应嵌入式分布处理结构和应用上网需求，嵌入式系统要求配备一种或多种标准的网络通信接口。

嵌入式系统和普通计算机系统（PC）相比有显著的区别，通过表 1.1 可以进一步理解嵌入式系统的特点。

表 1.1 嵌入式系统和普通计算机系统的区别

比 较 项 目	嵌入式系统	普通计算机系统
引导代码	BootLoader 引导，针对不同电路进行移植	主板的 BIOS 引导
OS	Windows CE、VxWorks、Linux 等，需要移植	Windows、Linux，不移植
驱动程序	每个设备都必须针对电路板进行开发	OS 中含有大多数，直接下载
协议栈	移植	OS 或第三方供应商提供
开发环境	借助服务器进行交叉编译	在本机可开发调试
仿真器	需要	不需要

1.1.2 嵌入式系统的分类

嵌入式系统的分类方式有很多种，下面按照嵌入式系统的发展历史将其分为两类。

（1）简单嵌入式系统

主要以单片机、DSP 为微处理器的系统，系统软硬件的复杂度很低。这种类型的嵌入式系统，在上、下位机通信和小的智能家电方面用得比较多。严格意义上讲，还不是“系统”的概念，只是使用 8 位的芯片执行了一些单线程的程序。

（2）复杂嵌入式系统

以嵌入式微处理器和 SoC 为核心，32 位以上（速度快，外围接口能力强），可以移植嵌入式操作系统，可以基于嵌入式操作系统编写嵌入式应用软件，从而缩短开发周期，降低开发成本并提高开发效率。

1.2 嵌入式系统的软硬件结构

嵌入式系统包括硬件和软件。在硬件方面，嵌入式系统的核心处理器称为嵌入式微处理器，负责控制整个嵌入式系统的执行。在软件方面，一般由嵌入式操作系统和应用软件组成，其中嵌入式操作系统可保证计算机硬件的使用，并能高效组织和正确地使用计算机资源。

嵌入式系统的硬件可分为 3 部分：微处理器、外围电路和外设，如图 1.1 所示。微处理器是嵌入式系统的控制核心，外围电路包括嵌入式系统的内存、I/O 端口、复位电路和电源等。外设包括通用串行总线（USB）、液晶显示（LCD）、键盘等。

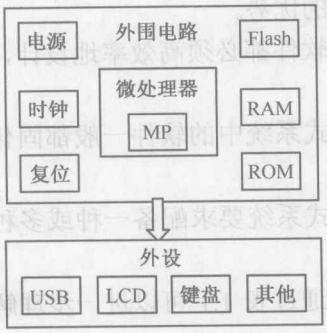


图 1.1 嵌入式系统的硬件

嵌入式系统的软件可分为设备驱动接口（Device Driver Interface, DDI）、实时操作系统（Real Time Operation System, RTOS）、可编程应用接口（Application Programmable Interface, API）和应用软件 4 个层次。其中 DDI 负责嵌入式系统与外部设备的信息交互；RTOS 通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等；API 为编制应用程序提供各种编程接口库。

1.3　嵌入式微处理器 ARM

嵌入式微处理器（Embedded Micro Processor Unit, EMPU）是与通用计算机中的 CPU 相对应的微处理器，32 位以上，具有较高的性能。在嵌入式应用中，通常将微处理器装配在专门设计的电路板上，只保留和嵌入式应用有关的母板功能，这样可以大幅度减小系统的体积和功耗。为了满足嵌入式应用的特殊要求，嵌入式微处理器虽然在功能上和标准微处理器基本上是一样的，但在工作温度、抗电磁干扰、可靠性等方面一般都做了各种增强。和工业控制计算机相比，嵌入式微处理器具有体积小、重量轻、成本低、可靠性高的优点，但是在电路板上必须包括 ROM、RAM、总线接口、各种外设等器件，从而降低了系统的可靠性，技术保密性也较差。

嵌入式微处理器对实时多任务有很强的支持能力。能完成多任务并且有较短的中断响应时间，从而使内部的代码和实时内核的执行时间减少到最低限度。

嵌入式微处理器具有功能很强的存储区保护功能。这是由于嵌入式系统的软件结构已模块化，而为了避免在软件模块之间出现错误的交叉作用，需要设计强大的存储区保护功能，同时也有利于软件诊断。

可扩展的处理器结构，可迅速地开发出满足应用的高性能的嵌入式微处理器。

嵌入式微处理器必须功耗很低，尤其是用于便携式的无线及移动的计算和通信设备中靠电池供电的嵌入式系统更是如此，如需要功耗达到毫瓦级，甚至微瓦级。

目前主要的嵌入式微处理器类型有 Am186/88、386EX、SC-400、PowerPC、68000、MIPS、ARM/StrongARM 系列等。

本书主要以 ARM 微处理器构建嵌入式系统，下面简单介绍一下嵌入式微处理器 ARM 的相关情况。

1.3.1　ARM 简述

ARM（Advanced RISC Machines）公司既不生产芯片，也不销售芯片，它只是出售芯片技术授权。它有众多的全球合作伙伴，包括大部分的半导体公司，这可以使其获得更多的第三方工具、制造和软件上的支持，降低成本，提高了市场竞争优势。ARM 微处理器以它卓越的性能和功效已被广泛地应用于工业控制、成像、网络、汽车、消费电子等各个领域，占据市场份额的 75%。

ARM 微处理器目前包括 ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、SecurCore 系列、Intel 的 StrongARM 和 XScale 等多个系列。每个系列的 ARM 微处理器都有各自的特点和功能，以适应不同领域的需求。

1.3.2　ARM 编程模型

所谓编程模型，是指对一款处理器进行编程控制时，所需要掌握和了解的与处理器相关的基本知识。

1. ARM 处理器模式

ARM 处理器共支持 7 种工作模式。在软件控制、外部中断或者异常处理下，都可以引发处理器工作模式的改变。ARM 处理器模式如表 1.2 所示。

表 1.2 ARM 处理器模式

处理器模式	用 途 描 述
用户模式	程序正常执行模式
系统模式	运行具有特权的操作系统任务
快速中断模式	处理快速中断，支持高速数据传送或者通道处理
外部中断模式	处理普通中断
管理模式	操作系统保护模式，处理软件中断（SWI）
中止模式	用于虚拟存储及存储保护
未定义指令中止模式	用于支持通过软件仿真硬件协处理器

2. ARM 寄存器

ARM 处理器共有 37 个寄存器。ARM 寄存器如表 1.3 所示。

表 1.3 ARM 寄存器

用 户 模 式	系 统 模 式	管 理 模 式	中 止 模 式	未定义指令模式	外部中断模式	快速中断模式
R0						
R1						
R2						
R3						
R4						
R5						
R6						
R7						
R8						R8_fiq
R9						R9_fiq
R10						R10_fiq
R11						R11_fiq
R12						R12_fiq
R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC						
CPSR						
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

有的寄存器在各个模式下只对应同一个物理寄存器，有些寄存器在各个模式下对应不同的物理寄存器。每种模式只能使用自己模式下的寄存器。

按照寄存器的功能，ARM 寄存器分为通用寄存器和状态寄存器两大类。

（1）通用寄存器 R0～R15

① 未备份寄存器 R0～R7。没有被系统用于特殊的用途，而且在各模式下它们对应同一个物理寄存器。

② 备份寄存器 R8～R14。寄存器 R8～R12 各对应两组物理寄存器：一组是 FIQ 模式下的，一组是除 FIQ 模式以外的所有模式下的。寄存器 R13 和 R14，除系统模式和用户模式共用一组物理寄存器外，其余的模式都有各自的物理寄存器。而且 R13 和 R14 有特殊的用途。R13 通常用于堆栈指针 SP，在 ARM 指令集中没有强制地使用它为堆栈指针，但是在 Thumb 指令集中则强制它作为堆栈指针。R14 又叫做程序连接寄存器 LR（Link Register），当发生子程序调用或者异常中断处理时，保存程序的返回地址。

③ 程序计数器 R15，也记做 PC。它用于标识下一条将要执行指令的地址。

（2）状态寄存器 CPSR 和 SPSR

CPSR 在任何模式下都可以被访问，它包含了条件标识位、中断标识位、当前处理器模式，以及其他的一些状态和控制位。

SPSR 在每种异常模式下都对应一个物理寄存器，当有异常中断发生时，它用于保存 CPSR 的内容，以便异常返回后恢复异常中断发生前的工作状态。

3. ARM 存储器格式

在 ARM 体系中，可以按照字地址、半字地址或者字节地址等方式寻址存储器。每个字单元包括 4 个字节单元或者两个半字单元。在构成一个字单元的 4 个字节中，哪一字节是高字节，哪一个字节是低字节，设计了大端和小端两种不同的存储格式。具体如图1.2 和图 1.3 所示。

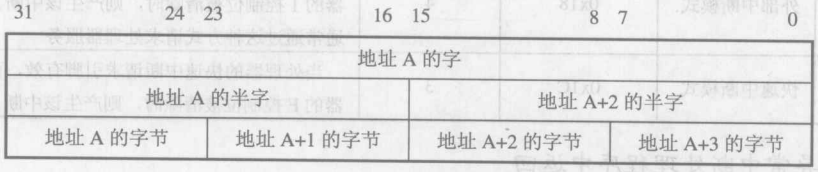


图 1.2　大端存储格式

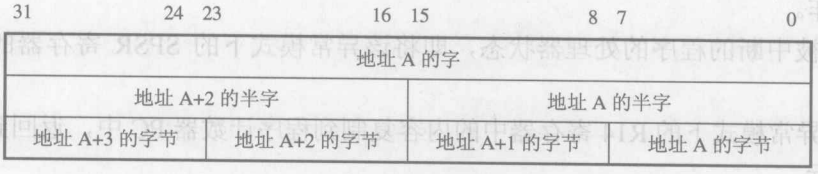


图 1.3　小端存储格式

4. 异常中断

在 ARM 体系结构中，把由内部或者外部源引发的事件称为异常，处理器由此而做的处理称为异常中断处理。ARM 体系中的异常中断如表 1.4 所示。

（1）ARM 处理器对异常中断的响应过程

① 保存处理器当前状态、中断屏蔽位及各种条件标志位。这是通过将程序状态寄存器 CPSR 保存到将要执行的异常中断所对应的寄存器 SPSR 中实现的。

② 设置程序状态寄存器 CPSR 中相应的位。包括设置 CPSR 中的位，使处理器进入相应的异常模式；设置 CPSR 中的位，禁止 IRQ 中断，当进入 FIQ 模式时，禁止 FIQ 中断。

③ 将异常中断的返回地址保存到相应异常模式下的 R14 中。

④ 将程序计数器 PC 的值设置成该中断的中断向量地址，从而跳转到相应的异常中断处理程序处执行。

表 1.4 ARM 体系中的异常中断

异常中断类型	异常中断模式	中断向量地址	优 先 级	异常中断含义
复位	管理模式	0x0	1	当处理器的复位引脚有效时，系统产生复位异常中断，程序跳转到复位异常中断处理程序处执行。产生复位异常中断的情况包括： （1）系统上电时； （2）系统复位时； （3）跳转到复位异常中断向量处执行，即软件复位
未定义指令	未定义指令中止模式	0x4	6	当 ARM 处理器或系统中的协处理器认为当前指令未定义时，产生该中断。可利用该中断机制模仿浮点向量运算
软件中断	管理模式	0x8	6	这是一个由用户定义的中断指令。可用于用户模式下的程序调用特权操作
指令预取中止	中止模式	0xC	5	如果处理器预取的指令地址不存在，或者该地址不允许当前指令访问，则产生该中断
数据访问中止	中止模式	0x10	2	如果数据访问的目标地址不存在，或者该地址不允许当前指令访问，则产生该中断
外部中断请求	外部中断模式	0x18	4	当处理器的外部中断请求引脚有效，而且 CPSR 寄存器的 I 控制位被清除时，则产生该中断。系统中各外设通常通过这种方式请求处理器服务
快速中断请求	快速中断模式	0x1C	3	当处理器的快速中断请求引脚有效，而且 CPSR 寄存器的 F 控制位被清除时，则产生该中断

（2）从异常中断处理程序中返回

除复位异常中断之外，其余类型的中断都要从中断处理程序返回。返回的过程包括以下两个基本操作。

① 恢复被中断的程序的处理器状态，即将该异常模式下的 SPSR 寄存器的内容复制到 CPSR 中。

② 将该异常模式下的 R14 寄存器中的内容复制到程序计数器 PC 中，返回到被中断的程序处继续执行。

1.3.3 ARM 指令集

ARM 处理器支持双指令集。一种是 32 位的 ARM 指令集，一种是 16 位的 Thumb 指令集。两种指令集之间可以自由切换，以优化软件设计。这里只对 ARM 指令集做简单介绍。

1. ARM 寻址方式

由于寻址方式是指令中提供操作数或操作数地址的方法，即它是针对操作数而做的规定。此外，操作数又分为立即数、寄存器类型、存储器类型这三类，不同种类的操作数肯定对应不同的读取操作方式，所以对应于各指令也会有不同的寻址方式。

（1）立即寻址方式：操作码字段后面的部分就是操作数本身，即要操作的数据直接就包含在指令中。

（2）寄存器直接寻址：要操作的数据在寄存器中存放。

（3）寄存器间接寻址：指令中要操作的数据为存储器操作数，该操作数的地址由一个寄存器存放（该寄存器充当地址指针的作用）。

（4）基址寻址：指令中要操作的数据为存储器操作数，该操作数的地址由两部分组成，分别是基址部分和偏移量部分，即[基址，偏移量]。基址必须由寄存器存放，该寄存器也相应地称做基址寄存器；偏移量则可以有多种形式，可以是立即数、寄存器或寄存器移位形式。将基址寄存器的内容与指令中给出的偏移量相加，则形成操作数的有效地址。

基址寻址又可以分为不带自动索引的基址寻址、前索引基址寻址和后索引基址寻址三类。

（5）寄存器移位寻址方式：这种寻址方式为 ARM 指令集特有，第 2 个寄存器操作数在与第一个操作数结合之前，选择进行移位操作。可选择的移位操作包括逻辑左移、逻辑右移、算术右移、循环右移和扩展为 1 的循环右移 5 种。

（6）多寄存器寻址：在寄存器和存储器之间进行数据传送时，一次可以传送多个寄存器值，允许一条指令传送 16 个寄存器的任何子集或所有寄存器的值。

（7）堆栈寻址：对堆栈空间进行存取操作，一次可以操作多个存储器单元。

2. ARM 指令格式

一条典型的 ARM 指令的编写格式如下：

```
<opcode>{<cond>}{s}  <Rd>,<Rn>{,operand2}
```

说明：

（1）<>内的项是必需的，{}内的项是可选的。

（2）opcode：指令助记符，标识指令的功能，如 LDR, STR, ADD 等。

（3）cond: 指令的执行条件，如 EQ, NE, LE 等等。ARM 指令可以都是有条件执行指令，只要引用恰当的条件助记符就能达到事半功倍的效果。

（4）s：是否影响 CPSR 寄存器的值，如果要使指令执行后影响 CPSR 的条件标志位，除四条逻辑比较指令之外，绝大部分的指令在编写时都需要加上“s”，否则不影响。

（5）Rd: 目标寄存器，用来存放运算的结果。

（6）Rn: 第一个操作数的寄存器。

（7）operand2: 第二个操作数，这是一条指令中最后一个操作数，只有它可以有多种形式，其他的必须是寄存器类型的操作数。

3. ARM 指令分类

ARM 指令集可以分为数据处理指令、跳转指令、存储器操作指令、程序状态寄存器传输指令、异常中断指令和协处理器传输指令 6 类。

（1）数据处理指令

ARM 数据处理指令包括数据传送指令、算术运算指令、逻辑运算指令、比较指令和乘法指令，如表 1.5 所示。

（2）跳转指令

通过跳转指令，可以实现程序的转移、子程序的调用，还可以实现在转移的同时在两种指令集间进行切换。跳转指令如表 1.6 所示。

表 1.5 ARM 数据处理指令

分类	指令助记符	指令名称	指令格式	操作
数据传送指令	MOV	数据传送指令	MOV {cond}{s} Rd, operand2	Rd←operand2
	MVN	数据非传送指令	MVN {cond}{s} Rd, operand2	Rd←(NOT operand2)
算术运算指令	ADD	加法指令	ADD {cond}{s} Rd, Rn, operand2	Rd←(operand2+Rn)
	ADC	带进位加法指令	ADC {cond}{s} Rd, Rn, operand2	Rd←(operand2+Rn+C)
	SUB	减法运算指令	SUB {cond}{s} Rd, Rn, operand2	Rd←(Rn–operand2)
	RSB	逆向减法运算	RSB {cond}{s} Rd, Rn, operand2	Rd←(operand2–Rn)
	SBC	带进位减法指令	SBC {cond}{s} Rd, Rn, operand2	Rd←(Rn–operand2+C–1)
	RSC	带进位逆向减法指令	RSC {cond}{s} Rd, Rn, operand2	Rd←(operand2–Rn+C–1)
逻辑运算指令	AND	与指令	AND {cond}{s} Rd, Rn, operand2	Rd←(Rn & operand2)
	ORR	或指令	ORR {cond}{s} Rd, Rn, operand2	Rd←(Rn | operand2)
	EOR	异或指令	EOR {cond}{s} Rd, Rn, operand2	Rd←(Rn ^ operand2)
	BIC	位清除指令	BIC {cond}{s} Rd, Rn, operand2	Rd←(Rn & (NOT operand2))
比较指令	CMP	比较指令	CMP{cond} Rn, operand2	根据 Rn–operand2 设置条件标志位
	CMN	负数比较指令	CMN{cond} Rn, operand2	根据 Rn+operand2 设置条件标志位
	TST	位测试指令	TST{<cond>} <Rn>, <operand2>	根据 Rn & operand 设置条件标志位
	TEQ	测试等价指令	TEQ{<cond>} <Rn>, < operand2>	根据 Rn ^ operand 设置条件标志位
乘法指令	MUL	32 位乘法指令	MUL{<cond>}{s} <Rd>, <Rm>, <Rs>	Rd←(Rm x Rs)[31:0]
	MLA	32 位带加数的乘法指令	MLA{<cond>}{s} <Rd>, <Rm>, <Rs>,<Rn>	Rd←(Rm x Rs+Rn)[31:0]
	SMULL	64 位有符号数乘法指令	SMULL{<cond>}{s} <RdLo>, <RdHi>, <Rs>,<Rn>	RdHi: RdLo←Rs×Rn
	SMLAL	64 位带加数的有符号数乘法指令	SMLAL{<cond>}{s} <RdLo>, <RdHi>, <Rs>,<Rn>	RdHi: RdLo←RdHi: RdLo+Rs×Rn
	UMULL	64 位无符号数乘法指令	UMULL{<cond>}{s} <RdLo>, <RdHi>, <Rs>, <Rn>	RdHi: RdLo←Rs×Rn
	UMLAL	64 位带加数的无符号数乘法指令	UMLAL{<cond>}{s} <RdLo>, <RdHi>, <Rs>, <Rn>	RdHi: RdLo←RdHi: RdLo+Rs×Rn

表 1.6 ARM 跳转指令

指令助记符	指令名称	指令格式	操作
B	无链接跳转指令	B{cond} label	PC←lable
BL	带链接跳转指令	BL{cond} label	LR←PC–4 ; PC←lable
BX	带状态切换的跳转指令	BX{cond} Rm	PC←Rm; 根据 Rm[0]修改寄存器 CPSR 中的 T 标志位，并切换处理状态
BLX	带链接和状态切换的跳转指令	BLX{cond} lable	LR←PC–4; PC←lable; 根据 Rm[0]修改寄存器 CPSR 中的 T 标志位，并切换处理状态

（3）存储器操作指令

ARM 是 RISC 结构的处理器芯片，为了提高系统的运行效能，不允许所有的指令都可以对存储器进行操作，因为对存储器操作意味着执行时间的延长，不利于流水线作业。所以，

对存储器的访问采用了 LDR/STR 结构，只有存储器操作指令才可以对存储器进行访问。对存储器进行读操作时，可以按照字、半字、字节或多字进行，在按照半字或字节操作时，可以将数据区分成带符号数据和无符号数据。根据操作性质的不同，ARM 中提供了如表1.7 所示的多种类型存储器操作指令。

表 1.7　ARM 存储器操作指令

指令助记符	指 令 名 称	指 令 格 式	操　作
LDR	字数据加载指令	LDR{cond}　Rd, ADDR	将内存中字地址为 ADDR 的字数据读取到 Rd 中
LDRB	字节数据加载指令	LDR{cond}B　Rd, ADDR	将内存中字节地址为 ADDR 的字节数据读取到 Rd 中的低字节处，Rd 的高 24 位部分清零
LDRH	半字数据加载指令	LDR{cond}H　Rd, ADDR	将内存中半字地址为 ADDR 的半字数据读取到 Rd 中的低 16 位处，Rd 的高 16 位部分清零
LDRSB	带符号字节数据加载指令	LDR{cond}SB　Rd, ADDR	将内存中字节地址为 ADDR 的带符号字节数据读取到 Rd 中的低字节处，Rd 的高 24 位部分设置成该字节数据的符号位的值
LDRSH	带符号半字数据加载指令	LDR{cond}SH　Rd, ADDR	将内存中半字地址为 ADDR 的带符号半字数据读取到 Rd 中的低 16 位处，Rd 的高 16 位部分设置成该字节数据的符号位的值
LDRT	以用户模式操作的字数据加载指令	LDR{cond}T　Rd, ADDR	以用户模式操作，将内存中字地址为 ADDR 的字数据读取到 Rd 中
LDRBT	以用户模式操作的字节数据加载指令	LDR{cond}BT　Rd, ADDR	以用户模式操作，将内存中字节地址为 ADDR 的字节数据读取到 Rd 中的低字节处，Rd 的高 24 位部分清零
STR	字数据存储指令	STR{cond}　Rd, ADDR	将 Rd 内容存储到以 ADDR 为字地址的字单元中
STRB	字节数据存储指令	STR{cond}B　Rd, ADDR	将 Rd 内容存储到以 ADDR 为字节地址的字节单元中
STRH	半字数据存储指令	STR{cond}H　Rd, ADDR	将 Rd 内容存储到以 ADDR 为半字地址的半字单元中
STRT	以用户模式操作的字数据存储指令	STR{cond}T　Rd, ADDR	以用户模式操作，将 Rd 内容存储到以 ADDR 为字地址的字单元中
STRBT	以用户模式操作的字节数据存储指令	STR{cond}BT　Rd, ADDR	以用户模式操作，将 Rd 内容存储到以 ADDR 为字节地址的字节单元中
LDM{mode}	批量寄存器加载指令	LDM{mode}　Rn{!}, Reglist	把以 Rn 为地址的附近的存储单元中的内容加载到一批寄存器中
STM{mode}	批量寄存器存储指令	STM{mode}　Rn{!}, Reglist	把一批寄存器中的内容存储到以 Rn 为地址的附近的存储单元中

（4）程序状态寄存器传输指令

如表 1.8 所示，ARM 中有两条专门的指令用于在状态寄存器和通用寄存器之间传送数据，其他的指令不允许对程序状态寄存器进行访问。这两条指令分别是 MRS（读状态寄存器指令）和 MSR（写状态寄存器指令）。修改状态寄存器是 MRS 和 MSR 这两条指令通过“读取—修改—写回”的操作序列来实现的。两条指令配合使用，可用来进行处理器模式切换、允许/禁止 IRQ/FIQ 中断设置。程序不能通过直接修改 CPSR 中的 T 控制位实现程序 ARM/Thumb 状态的切换，这必须通过 BX 等指令完成。

表 1.8　ARM 程序状态寄存器传输指令

指令助记符	指 令 名 称	指 令 格 式	操　作
MRS	读状态寄存器指令	MRS{cond}　Rd, psr	将状态寄存器的内容传送到通用寄存器中
MSR	写状态寄存器指令	MSR{cond}　psr_fields, operand2	将通用寄存器的内容或立即数传送到状态寄存器中