21st CENTURY
规划教材

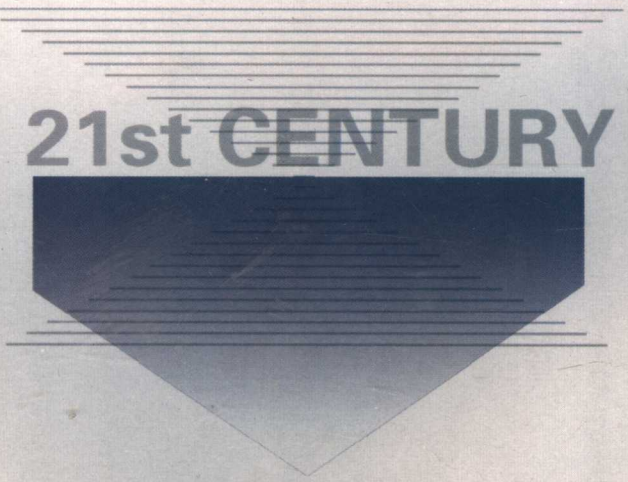
高等院校信息与电子技术类规划教材
University Textbooks of Information Technology

嵌入式微型计算机系统实例教程

——ARM与Linux

郑慕德　编著

★ ★ ★ ★ ★



科学出版社
www.sciencep.com

高等院校信息与电子技术类规划教材

嵌入式微型计算机系统实例教程
——ARM与Linux

郑慕德 编著

科学出版社
北京

图字 01-2005-2288 号

内 容 简 介

本书以 Linux 环境作为嵌入式系统的应用平台，以 ARM 芯片作为嵌入式系统的核心。因为本书的实例程序是用 C 语言编写的，所以读者应对 Linux 及 C 语言有基本的了解。本书主要介绍 ARNUX 嵌入式系统开发平台、ARNUX 操作系统环境及其指令和语法，以及 13 个实例，每个实例包括原理介绍、功能说明、电路图和程序流程图、程序及说明。本书中的许多实例程序都浅显易懂，初学者能快速地掌握嵌入式系统的基本知识及应用。

本书是有关嵌入式系统的入门书籍，适合一般初学者了解嵌入式系统的基本程序设计及应用，也适合大专院校电子电工、计算机等专业“嵌入式系统设计”课程使用。

图书在版编目（CIP）数据

嵌入式微型计算机系统实例教程：ARM 与 Linux /郑慕德编著. —北京：科学出版社，2006

（高等院校信息与电子技术类规划教材）

ISBN 7-03-017612-X

Ⅰ.嵌… Ⅱ.郑… Ⅲ.微型计算机—系统设计 Ⅳ.TP360.21

中国版本图书馆 CIP 数据核字（2006）第 076315 号

责任编辑：赵卫江/责任校对：刘彦妮

责任印制：吕春珉/封面设计：飞天创意

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

[illegible] 印刷

科学出版社发行 各地新华书店经销

*

2006 年 7 月第 一 版 开本：787×1092 1/16

2006 年 7 月第一次印刷 印张：18 1/4

印数：1-3 000 字数：418 000

定价：26.00 元

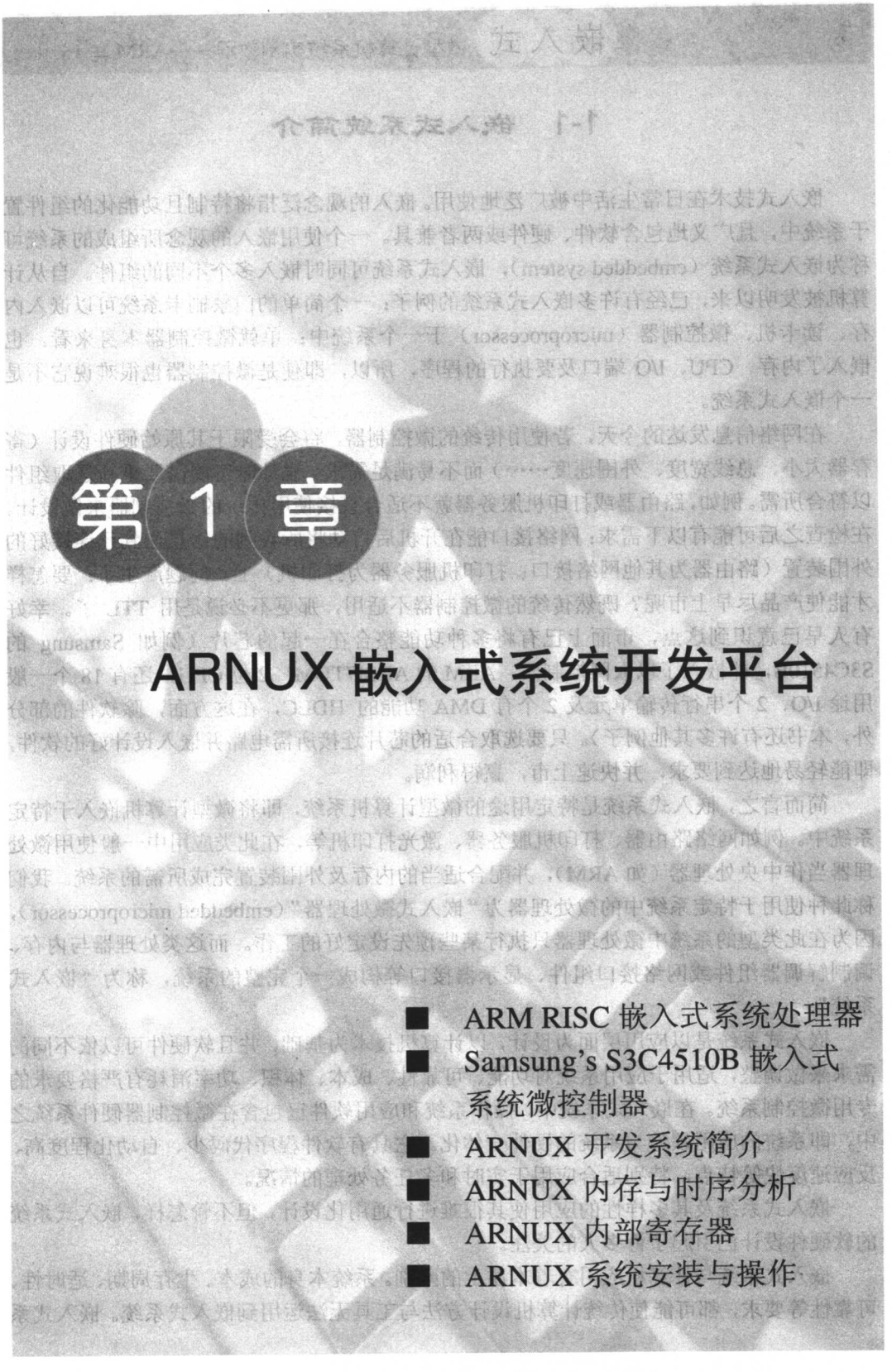
（如有印装质量问题，我社负责调换<环伟>）

销售部电话 010-62136131 编辑部电话 010-62138017（HI01）

目　录

目录

目录



第1章

ARNUX 嵌入式系统开发平台

- ARM RISC 嵌入式系统处理器
- Samsung's S3C4510B 嵌入式系统微控制器
- ARNUX 开发系统简介
- ARNUX 内存与时序分析
- ARNUX 内部寄存器
- ARNUX 系统安装与操作

1-1 嵌入式系统简介

嵌入式技术在日常生活中被广泛地使用。嵌入的观念泛指将特制且功能化的组件置于系统中，且广义地包含软件、硬件或两者兼具。一个使用嵌入的观念所组成的系统可称为嵌入式系统（embedded system），嵌入式系统可同时嵌入多个不同的组件。自从计算机被发明以来，已经有许多嵌入式系统的例子：一个简单的门禁刷卡系统可以嵌入内存、读卡机、微控制器（microprocessor）于一个系统中；单就微控制器本身来看，也嵌入了内存、CPU、I/O 端口及要执行的程序，所以，即使是微控制器也很难说它不是一个嵌入式系统。

在网络信息发达的今天，若使用传统的微控制器，将会受限于其原始硬件设计（寄存器大小、总线宽度、外围速度……）而不易满足需求。设计时可能需要重新选取组件以符合所需。例如，路由器或打印机服务器就不适合直接使用传统的微控制器来做设计。在检查之后可能有以下需求：网络接口能在开机后自动将所收到的信息送到已连接好的外围装置（路由器为其他网络接口；打印机服务器为打印机）上。问题产生了：要怎样才能使产品尽早上市呢？既然传统的微控制器不适用，那更不必说是用 TTL 了。幸好有人早已意识到这点，市面上已有将多种功能整合在一起的芯片（例如 Samsung 的 S3C4510B，即嵌入了以太网控制器、ARM 的 ARM7TDMI 32 位 CPU，还有 18 个一般用途 I/O、2 个串行传输单元及 2 个有 DMA 功能的 HDLC，在这方面，除软件的部分外，本书还有许多其他例子）。只要选取合适的芯片连接所需电路并嵌入设计好的软件，即能轻易地达到要求，并快速上市，赢得利润。

简而言之，嵌入式系统是特定用途的微型计算机系统，即将微型计算机嵌入于特定系统中。例如网络路由器、打印机服务器、激光打印机等，在此类应用中一般使用微处理器当作中央处理器（如 ARM），并配合适当的内存及外围装置完成所需的系统。我们称此种使用于特定系统中的微处理器为“嵌入式微处理器”（embedded microprocessor），因为在此类型的系统中微处理器只执行某些预先设定好的工作。而这类处理器与内存、调制解调器组件或网络接口组件、显示器接口等构成一个完整的系统，称为“嵌入式系统”。

嵌入式系统是以应用层面为设计，以计算机技术为基础，并且软硬件可以依不同的需求来做调整，适用于应用系统对功能、可靠性、成本、体积、功率消耗有严格要求的专用微控制系统。在嵌入式系统中，操作系统和应用软件已包含在微控制器硬件系统之中，即系统的应用软件与系统的硬件一体化。它具有软件程序代码少、自动化程度高、反应速度快等特点，特别适合应用于实时和多任务处理的情况。

嵌入式系统及其多样性的应用使其很难进行通用化设计，但不管怎样，嵌入式系统的软硬件设计已引起了许多人的关注。

嵌入式系统与计算机应用有着本质上的区别。系统本身的成本、生存周期、适时性、可靠性等要求，都可能使传统计算机设计方法与工具无法运用到嵌入式系统。嵌入式系

统在很多情况下，需要考虑的是其产品性能、生命周期和商业趋势要达到最优化，而不是努力提升其最大计算量与计算速度。

通用计算机系统的设计方法最关心的是处理的效能与速度，但是这种方法对于完成一个具有市场占有率的嵌入式系统来说是不够的。因为在很多情况下产品的成功与否不在于一个复杂的系统是否可运作，而在于是否有性价比上的优势。传统计算机设计总是致力于提供大量且快速的计算能力，而在嵌入式系统中，外部界面、控制和计算等可能更重要；CPU 仅仅是实现这些功能的手段，嵌入式系统有着多样性的需求。

嵌入式系统非常受限于功能和具体的应用环境，如对外部事件必须保证在规定时间内进行反应，有体积、重量的限制，功率消耗、散热必须符合环境要求，并需有令人满意的安全性、可靠性，以及系统本身的成本需求等。

1-2 ARM RISC 嵌入式系统处理器

本节将概述 ARM 微处理器内部细节。如果读者已具备微处理机汇编语言基础，且不想深入了解 ARM 微处理器，可跳至第 1-4 节阅读。

1-2-1 ARM RISC 处理器简介

ARM 公司是英国一家专门生产 RISC 架构嵌入式系统处理器的公司，成立于 1990 年，主要生产开发嵌入式系统用的 16 位或 32 位的处理器，并将此技术授权给半导体公司进行 ARM 处理器的制造。

ARM 公司生产的 ARM RISC 处理器近年来已广受市场喜爱。由于 ARM RISC 架构处理器拥有低消耗功率、高运算效能及高度整合性等特点，许多芯片设计公司纷纷向 ARM 公司购买 ARM RISC 架构微处理器的核心，并用来当作自己公司发展微处理器芯片中的运算核心。

ARM RISC 架构处理器系列目前主要分为几个家族产品，分别为 ARM 7 Thumb 家族、ARM 9 Thumb 家族、ARM 10 Thumb 家族以及 StrongARM 家族系列，其中 StrongARM 家族已被 Intel 公司取得专用授权。

1-2-2 ARM 处理器寄存器

ARM 处理器共有 37 个 32 位的寄存器，这些寄存器并不同时使用。寄存器的使用会依照 ARM 处理器所执行的指令集状态与工作模式来决定。ARM 的寄存器可大致分为两类：一般用途寄存器与特殊用途寄存器。

1. 一般用途寄存器

寄存器 R0～R12 为可任意使用的寄存器。除了在 FIQ 工作模式时 R8～R12 为特殊用途外，其余模式时 R8～R12 皆为共享寄存器。

寄存器 R13 通常作为堆栈指针寄存器（stack pointer，SP），虽然程序设计者可用其他的寄存器来当堆栈指针寄存器，但大多数的 ARM 汇编语言编译器会将 R13 寄存器指定为堆栈指针寄存器。所以使用者最好还是依惯例将 R13 用来当作堆栈指针寄存器，才不会发生不必要的错误。

寄存器 R14 为链接寄存器（link register，LR），用于储存调用子程序或例外事件处理程序后所要返回的地址。

寄存器 R15 为程序计数器（program counter，PC），此寄存器用来储存下一个要执行指令的地址。但 ARM 处理器采用管线架构来提升执行效率，所以会先将要执行的指令预先撷取、译码、执行。所以 R15 所储存的地址将会比执行中指令所在地址多 4 个字节或 2 个字节，会随所执行的指令集不同而有所不同。

2. 特殊用途寄存器

在 ARM 处理器中有一个程序现况状态寄存器（current program status register，CPSR）与程序状态保存寄存器（saved program status register，SPSR）。程序现况状态寄存器记录了处理器目前的各种状态，包含负号、进位、零值等，最重要的是记录了处理器目前的工作模式。

程序状况保存寄存器用来保存进入例外事件程序前的程序状况，总共有 5 个例外状况，分别为 FIQ 状态、Supervisor 状态、Abort 状态、IRQ 状态与 Undefined 状态。寄存器分别命名为 SPSR_FIQ、SPSR_SVC、SPSR_ABT、SPSR_IRQ 与 SPSR_UND。System 状态与 User 状态因为不属于例外事件处理程序中，所以并没有专用的程序状态保存寄存器。

CPSR 与 SPSR 寄存器中的内容值所代表的意义如图 1.1 所示。

31	30	29	28	…	7	6	…	4	3	2	1	0
N	Z	C	V		I	F		M4	M3	M2	M1	M0

图 1.1　PSR 寄存器

标记所代表的意义，如表 1.1 所示。

表 1.1　CPSR 与 SPSR 寄存器中各标记的意义

标志	含义	说明
N	negative	若产生负数，则此位设为 1
Z	zero	若产生零，则此位设为 1
C	carry	若产生进位，则此位设为 1
V	overflow	若产生溢位，则此位设为 1
I	IRQ	中断禁用
F	FIQ	快速中断禁用

工作模式设定，如表 1.2 所示。

表 1.2 CPSR 与 SPSR 寄存器中的工作模式设定

M4	M3	M2	M1	M0	工作模式
1	0	0	0	0	User 模式
1	0	0	0	1	FIQ 模式
1	0	0	1	0	IRQ 模式
1	0	0	1	1	Supervisor 模式
1	0	1	1	1	Abort 模式
1	1	0	1	1	Undefined 模式
1	1	1	1	1	System 模式

3. ARM 指令集与 THUMB 指令集下寄存器状态

ARM7TDMI 微处理器有两种指令集可供选用。关于这两个指令集的介绍，请参阅 1-2-3 节。下面分别介绍这两个指令集下使用的寄存器。

（1）ARM 指令集状态下的寄存器

当 ARM 处理器在 ARM 指令集状态时，若程序操作在 User 状态，可使用的寄存器为 R0～R15。其中 R15 为程序计数器（PC）及程序现况寄存器（CPSR）。当切换到 System 状态时，可使用的寄存器与 User 状态时完全相同。但变换到其他模式时所使用的寄存器就不完全相同了。ARM 处理器在 ARM 指令集状态下各种工作模式所能使用的寄存器如图 1.2 所示。

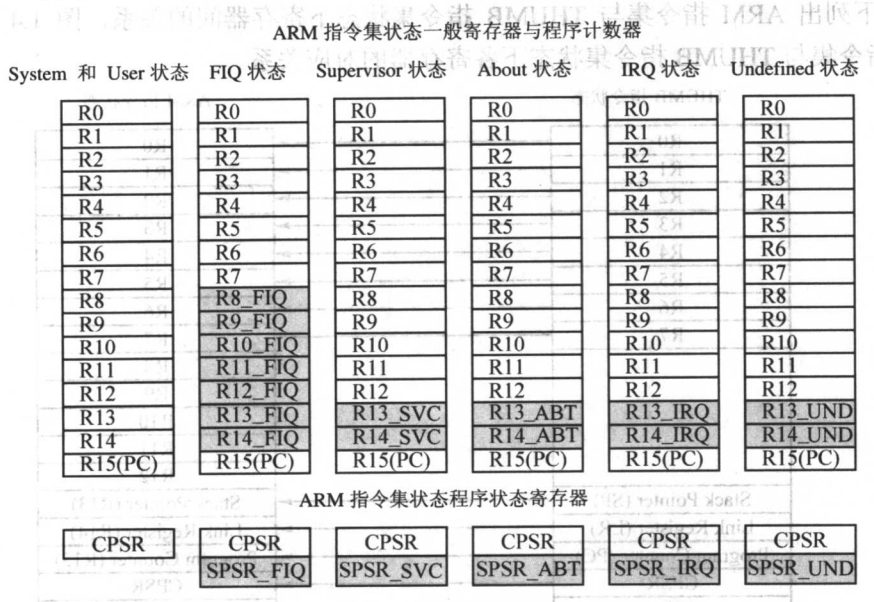


图 1.2 ARM 指令集下寄存器状态

（2）THUMB 指令集状态下的寄存器

ARM 处理器在 THUMB 指令集时所能使用的寄存器如图 1.3 所示。由图可知，在 THUMB 指令集中所使用到的寄存器其实只是在 ARM 指令集中所使用到寄存器的一部分。其中包含了 R0～R7 寄存器、堆栈指针寄存器（SP）、链接寄存器（LR）与程序计数器（PC）、程序现况状态寄存器（CPSR）及程序状态保存寄存器（SPSR）。

图 1.3　THUMB 指令集下寄存器状态

（3）ARM 指令集与 THUMB 指令集状态下的比较

以下列出 ARM 指令集与 THUMB 指令集状态下寄存器间的关系，图 1.4 说明了 ARM 指令集与 THUMB 指令集状态下各寄存器的对应关系。

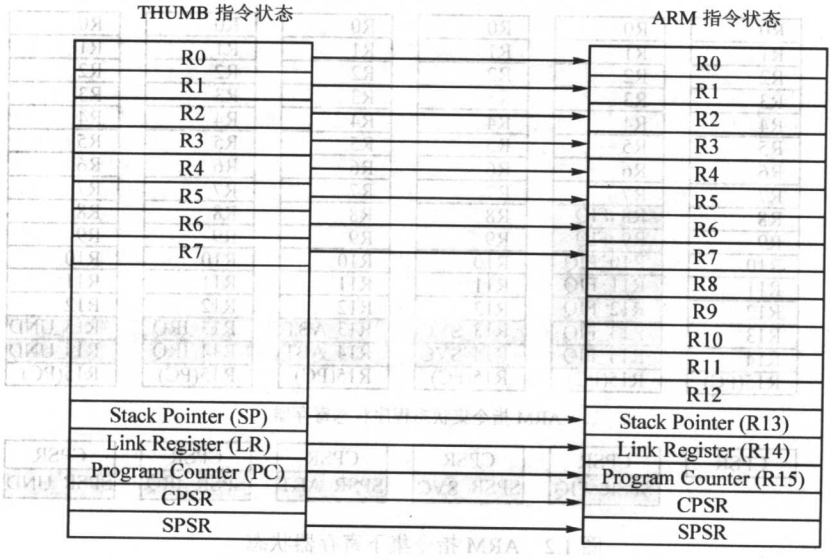


图 1.4　ARM 指令集与 THUMB 指令集寄存器的对应关系

1）THUMB 指令集状态 R0～R7 与 ARM 指令集状态 R0～R7 是相同的寄存器。

2）THUMB 指令集状态 CPSR 和 SPSR 与 ARM 指令集状态 CPSR 和 SPSR 是相同的寄存器。

3）THUMB 指令集状态 SP 寄存器对应 ARM 指令集状态的 R13 寄存器。

4）THUMB 指令集状态 LR 寄存器对应 ARM 指令集状态的 R14 寄存器。

5）THUMB 指令集状态 PC 寄存器对应 ARM 指令集状态的 R15 寄存器。

1-2-3 ARM 处理器指令集介绍

本节介绍 ARM 处理器的寄存器与各种运算的指令集。ARM 处理器有两种指令集可供选用——ARM 指令集状态与 THUMB 指令集状态。当 ARM 处理器在 ARM 指令集状态时，每一个指令码的长度为 32 位，也就是说一次存取数据大小为 32 位也就是 4 字节（byte）；当 ARM 处理器处于 THUMB 指令状态时，每个指令码的长度为 16 位。以下所要介绍的指令集为 ARM 指令集状态，也就是 32 位的 ARM 指令码。

在 ARM 指令集状态中，所有指令都会根据 CPSR 寄存器的状态与条件（condition field）做有条件的执行。格式为

```
<opcode> {条件} {S}
```

{条件}：{条件}为加在指令字尾的两字符符号。当 CPSR 寄存器的状态符合{条件}所设定的条件，该指令才被执行；否则，该指令被忽略。在指令之后总共可附加 15 种条件，如表 1.3 所示。

表 1.3 ARM 指令集状态下指令之后可附加的条件

代码	后缀	标志	含义
0000	EQ	Z=1	等于
0001	NE	Z=0	不等于
0010	CS	C=1	大于等于（无符号数）
0011	CC	C=0	小于（无符号数）
0100	MI	N=1	负数
0101	PL	N=0	正数或零
0110	VS	V=1	溢位
0111	VC	V=0	不溢位
1000	HI	C=1 and Z=0	大于（无符号数）
1001	LS	C=0 or Z=1	小于等于（无符号数）
1010	GE	N=V	大于等于（有符号数）
1011	LT	N!=V	小于（有符号数）
1100	GT	Z=0 and N=V	大于（有符号数）
1101	LE	Z=1 or N!=V	小于等于（有符号数）
1110	AL	none	空位

以加法指令（ADD）来说明。ADD 以 ADD EQ 表示：当 Z=1（Equal）时，ADD 指令才被执行。如果指令字尾未加 condition field，则预设为 AL，表示不管 CPSR 的状态为何，皆会执行该指令，所以 ADD = ADD AL。

{S}：在指令字尾加 S。该指令会按照执行状况去更改 CPSR 寄存器里的 ALU 位 Z、C、N、V。下面是程序为字尾{条件}{S}的实例：

```
ADD  r0, r1, r2; r0 = r1 + r2，不更新 CPSR 的 ALU 标记
ADD  S  r0, r1, r2; r0 = r1 + r2，更新 CPSR 的 ALU 标记
ADD  EQ  S   r0, r1, r2; If Z=1 then  r0 = r1 + r2，并更新 CPSR 的 ALU 标记
CMP  r0, r1; 根据 r0-r1 的结果，更新 CPSR 的 ALU 标记
```

其中，CMP、CMN、TST、TEQ 这些指令不要使用 S 字尾，因为其功能原本就是更改 CPSR 的 ALU 标记，也就是说已经隐含了 S 字尾。

1. ARM 寄存器控制指令

操作 32 位 PSR 的指令，用来设定状态寄存器的指令有以下两个：MRS 和 MSR。

值得注意的是，我们不能在 32 位模式中使用 MOVS PC，R14 或 LDMFD R13!，{registers,PC}^，也不能使用 ORRS PC,R14,#1<<28 来设置 V 标记，而是需要使用 MRS 和 MSR。关于 MOVS、LDMFD 与 ORRS 指令，请参阅本节后续内容。

（1）复制一个寄存器到 PSR 中

```
MSR CPSR, R0                ; 复制 R0 到 CPSR 中
MSR SPSR, R0                ; 复制 R0 到 SPSR 中
MSR CPSR_flg, R0            ; 复制 R0 的标记到 CPSR 中
MSR CPSR_flg, #1<<28        ; 复制（立即值）标记到 CPSR 中
```

（2）复制 PSR 到一个寄存器中

```
MRS R0, CPSR                ; 复制 CPSR 到 R0 中
MRS R0, SPSR                ; 复制 SPSR 到 R0 中
```

ARM 有两个 PSR，其中一个为 CPSR，另一个为 SPSR。CPSR 是当前的程序状态寄存器，而 SPSR 是保存的程序状态寄存器。每个有特权的模式都有自己的 SPSR，可获得的 PSR 如下：

CPSR_ALL——目前的状态寄存器。

SPSR_SVC——保存的状态寄存器，SVC（32）模式。

SPSR_IRQ——保存的状态寄存器，IRQ（32）模式。

SPSR_ABT——保存的状态寄存器，ABT（32）模式。

SPSR_UND——保存的状态寄存器，UND（32）模式。

SPSR_FIQ——保存的状态寄存器，FIQ（32）模式。

2. 寄存器加载与储存指令

寄存器加载与寄存器储存指令是非常有用的指令。在其他指令中大部分只能对寄存

器做控制而无法存取内存，而若要对内存内容做存取，必须先把数据放入寄存器中，做法是用寄存器来做间接寻址，先将要存取的地址放入寄存器中，再使用寄存器加载或储存指令对内存做存取动作。以下是几个常用的内存存取指令：

- LDR（见图 1.5）。
- STR。
- LDM。
- STM。
- SWP。

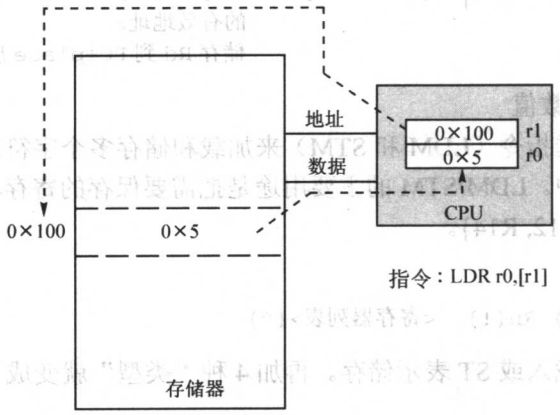


图 1.5 LDR 指令实例说明

（1）传送单一数值

使用单数值传送指令（STR 和 LDR）来加载和储存单一字节数值，可以从寄存器到内存或由内存中将值取出放到寄存器内。

首先来看一下指令的格式：

```
LDR{条件}        Rd,  <地址>
STR{条件}        Rd,  <地址>
LDR{条件}B       Rd,  <地址>
STR{条件}B       Rd,  <地址>
```

这些指令加载和储存 Rd 的值到指定的地址或从所指定地址读到 Rd 中。如果在指令后面指定了“B”，则只加载或储存一个单一的字节；而对于加载，寄存器中高位的 3 个字节被设为零（因为一个寄存器有 32 位，也就是有 4 字节，所以在加载单一字节时，需将较高的 3 个字节设为 0）。

地址可以是一个简单的值，或一个偏移量，或者是一个被移位的偏移量。还可以把合成的有效地址写回到基底寄存器（除去了对加/减操作的需要）。

各种寻址模式的实例如下：

```
STR Rd, [Rbase]                储存 Rd 到 Rbase 所包含的有效地址。
STR Rd, [Rbase, Rindex]        储存 Rd 到 Rbase + Rindex 所合成的有效地址。
```