【博客藏经阁丛书】

ARM 程序分析与设计

王宇行 编著

北京航空航天大学出版社

博客藏经阁丛书

ARM®程序分析与设计

王宇行 编著

北京航空航天大学出版社

内 容 简 介

以实例阐述知识点，从易到难，系统阐述 ARM 嵌入式开发的知识和技能。内容包括：ARM 开发工具，ARM 映像文件分析，ARM 汇编语言，ARM C 语言，标准 C 库的应用，ARM 汇编语言和 C 语言混合编程，引导代码分析，ARM 中断处理原理和实现方法，调试的基本原理，scatter 文件分析，位置无关代码和数据。

本书可作为学习 ARM 嵌入式技术的培训教材或 ARM 嵌入式系统开发人员的参考书，也可作为高等院校电子工程、自动控制和计算机等专业的参考书。

图书在版编目(CIP)数据

ARM 程序分析与设计/王宇行编著. —北京：北京航空航天大学出版社，2008.3

ISBN 978-7-81124-252-2

Ⅰ.A… Ⅱ.王… Ⅲ.微处理器，ARM—系统设计 Ⅳ.TP332

中国版本图书馆 CIP 数据核字(2008)第 010089 号

ARM® 程序分析与设计

王宇行 编著

责任编辑 张冀青

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话：010-82317024 传真：010-82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本：787 mm×960 mm 1/16 印张：21 字数：470千字

2008 年 3 月第 1 版 2008 年 3 月第 1 次印刷 印数：5 000 册

ISBN 978-7-81124-252-2 定价：32.00 元

前言

嵌入式技术与 ARM

在现实生活中，嵌入式设备随处可见，成为人们生活的一部分，例如手机、数码相机、个人数字助理(PDA)等。中国作为嵌入式设备消费和制造大国，对嵌入式技术人才的需求是巨大的。ARM 处理器(它的全称是 Advanced RISC Machines)是目前世界上最为流行的和应用最为广泛的 32 位处理器之一，它被嵌入到日常生活使用的各种产品中，从手机到汽车里的定位仪器。它体积小，质量轻，成本低，可靠性高，而且高性能、低功耗，是嵌入式设备的核心。社会对基于 ARM 嵌入式技术的软硬件设计人才、应用开发人才的需求越来越大，同时基于 ARM 系统的开发人员也越来越庞大，有初学者，也有身经百战的资深工程师。这就是本书诞生的根本缘由。本书的目的，就是为读者提供学习和开发 ARM 系统所需要的技能和经过消化的知识点；使读者掌握基础知识的同时，系统地掌握基于 ARM 的嵌入式系统的开发方法，以最快的速度和最短的时间掌握 ARM 系统下的开发技术。

本书特点

(1) 基础与实践的结合。本书详细地以实例阐述了开发基于 ARM 的嵌入式系统所需要的基础知识，而采用的实例都来源于实际的工程开发，所以可以在实际的工程项目中直接采用。对于入门者，可以学习基础知识；对于资深工程师，可以强化基础知识，也可以在实际的开发中参考借鉴。

(2) 分散与系统的结合。本书独立地阐述了基于 ARM 的嵌入式系统基础知识，同时也用实例系统地阐述了嵌入式系统的开发流程，采用点面结合的方法，帮助读者系统地掌握 ARM 嵌入式系统的开发技术。

主要内容

全书共分 11 章，用实例体现知识点，从易到难，系统地阐述了基于 ARM 的嵌入式开发知识和技能。

第 1 章：描述 ARM 开发工具及其开发的整个过程，包括编译、链接以及最终可执行映像文件的产生。对常用的编译参数和 makefile 作了简单的描述。目的是，给出一个总体开发轮

廓和概念，为理解以后的章节做准备。

第 2 章：系统地介绍 ARM 编译器 armcc、armasm 输出的目标文件，以及 ARM 链接器输出的可执行映像文件的物理结构和逻辑结构，详细描述与之相关的一些技术和实现实例：装载域和运行域的存储映射，分散装载技术。

第 3 章：详细阐述 ARM 处理器的基本概念、ARM 汇编语言编程及大量的 ARM 汇编实例。

第 4 章：介绍用 C 语言开发 ARM 系统常用的一些概念，以及 C 语言编程的优化技术。

第 5 章：介绍标准 C 库的使用方法、使用流程以及对标准 C 库的裁剪方法，以满足实际的目标系统。

第 6 章：以实例阐述 ARM 汇编语言和 C 语言混合编程的技术，包括内联汇编、嵌入式汇编以及 ARM 汇编语言和 C 语言函数相互调用的编程技术。

第 7 章：详细阐述系统初始化原理和流程。对系统初始化过程中相关的硬件作了详细描述。

第 8 章：详细阐述中断处理原理和实现方法。以一个外部中断为例来介绍编写中断服务程序的流程和一些基本概念。

第 9 章：介绍调试的基本原理和技术。重点讨论基于 JTAG 调试的基本理论和方法。

第 10 章：阐述分散装载技术所需要的配置文件和分散装载技术在实际开发中的使用技术和方法。

第 11 章：用实例阐述位置无关代码和数据的原理和开发方法。

主要读者

本书对 ARM 处理器从基本概念到整个系统开发做了详细的阐述，涉及每个概念和知识点，不是简单的罗列，而是以通俗易懂的语言和详细的实例来阐述它，使读者易于消化和吸收。通过阅读本书，读者可以系统地掌握 ARM 开发的各方面知识，所以，本书可以作为学习 ARM 技术的培训教材或 ARM 开发人员的参考用书，也可以作为高等院校电子工程、自动控制和计算机等专业的参考书。

致 谢

感谢我的很多同事提供了技术上的建议，感谢我的家人对我的支持和关心。由于时间仓促和水平所限，本书难免有疏漏和不足之处，请读者批评指正并给出宝贵意见。联系方式 E-mail：micro9229@gmail.com，或登陆我的博客 http://micro9229.bokee.com/index.html，与我进一步交流。

王宇行

2008 年 1 月

目 录

第 5 章 标准 C 库的应用

第 6 章 ARM 汇编语言和 C 语言混合编程

第1章 ARM开发工具

工欲善其事，必先利其器。做嵌入式系统的开发，必须了解和熟知开发流程和开发工具，这是开发人员必须具备的基本能力。本章简要描述这两个专题。

1.1 开发流程

本节介绍从源程序到运行在硬件设备上的二进制代码的整个演变流程。整个流程包括下面5个步骤：

① 用C语言或汇编语言编写源代码。

② 用编译器把源代码编译成ELF(Executable and Linking Format，对于更为详细的解释，请参见http://www.arm.com/pdfs/DUI0101A_Elf.pdf)的目标文件。

③ 用链接器把目标文件链接成ELF格式的可执行映像文件。

④ 用fromelf工具把可执行的映像文件转换成二进制映像文件。

⑤ 把二进制映像文件烧写到目标系统二进制映像文件。

二进制映像文件就是最终在RAM(Random Access Memory)运行的、完成特定功能的二进制代码。俗话说：一张图胜过千言万语。下面用图1.1来展示这个流程。

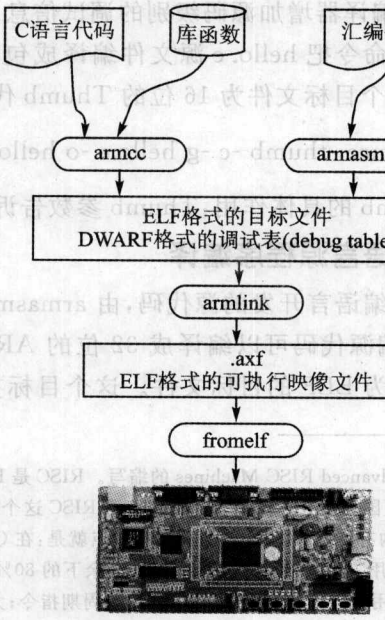

图1.1 开发流程图

1.2 开发工具

1.2.1 编译器简介

1. C语言源程序编译

ARM[①]编译器armcc负责编译用C语言或C++语言编写的源代码。根据编译参数的选择，可以编译成32位的ARM代码，也可以编译成16位的Thumb[②]代码。最终输出格式为ELF的目标文件。这个目标文件可以包括DWARF[③]格式的调试表，也可以不包括，完全由编译参数控制。armcc的使用方法如下：

编译 armcc -c -g hello.c -o hello.o。

这条命令把hello.c源文件编译成包括DWARF格式调试表的ELF格式的目标文件(hello.o)。这个目标文件为32位的ARM代码。参数c和g的具体作用如下：

-c：告诉编译器只编译，不链接。

-g：告诉编译器增加源码级别的调试信息。

下面这条命令把hello.c源文件编译成包括DWARF格式调试表的ELF格式的目标文件：hello.o，这个目标文件为16位的Thumb代码。

编译 armcc -thumb -c -g hello.c -o hello.o。

参数thumb的具体作用：Thumb参数告诉编译器生成16位的Thumb代码。

2. 汇编语言源程序编译

对于用汇编语言开发的源代码，由armasm编译器负责编译。与armcc一样，根据编译参数的选择，汇编源代码可以编译成32位的ARM代码，也可以编译成16位的Thumb代码。最终输出格式为ELF的目标文件。这个目标文件可以包括DWARF格式的调试表，也可以

---

① ARM是Advanced RISC Machines的缩写。RISC是Reduced Instruction Set Computer的缩写，意思为精简指令集计算机。1979年美国加州大学伯克利分校提出了RISC这个概念。传统的CISC(Complex Instruction Set Computer，复杂指令集计算机)结构有其固有的缺点，最明显的缺点就是：在CISC指令集的各种指令中，其使用频率相差悬殊，大约有20%的指令会被反复使用，占整个程序代码的80%；而余下的80%的指令却不经常使用，在程序设计中只占20%。RISC体系结构具有的特点：采用固定长度的指令格式；使用单周期指令；大量使用寄存器。

② Thumb指令长度为16位。Thumb指令集为ARM指令集的功能子集，但与等价的ARM代码相比较，可节省30%～40%的存储空间，同时具备32位代码的所有优点。

③ 调试器调试程序，需要调试信息。如果在编译程序的时候加上参数g，那么编译输出的目标文件就会包含调试信息。调试信息的表示格式有很多种，在ARM系统里使用DWARF格式。DWARF(Debugging With Attributed Record Formats)现在已经有DWARF 1、DWARF 2和DWARF 3三个版本。

不包括，完全由编译参数控制。

下面两个实例演示了 armasm 的使用方法。

① 编译成带调试信息的 32 位 ARM 代码：

编译　armasm -32 -g hello.s -o hello.o。

② 编译成带调试信息的 16 位 Thumb 代码：

编译　armasm -16 -g hello.s -o hello.o。

3. 目标文件

源代码经过 armcc 和 armasm 的编译，输出了 ELF 格式的目标文件。这个目标文件由编译参数 g 来决定是否包含 DWARF* 格式的调试表。调试表里的信息就是调试器调试程序时需要的信息，简称调试信息。如果在编译的时候，使用了参数 g，则在编译的时候，编译器会从源文件中收集变量名、变量类型、变量所在行号、函数名、函数参数、函数的地址范围、行号和地址的对应关系等调试器需要的信息；然后按照一种特定的格式写入到编译后的目标文件中。调试的时候，调试器便从文件中读取并解析这些信息，从而达到调试程序的目的。调试信息一般都是按照什么样的格式存放的呢？在 ARM 开发系统中使用 DWARF 格式来存放调试信息。为什么调试信息要用其他格式来表示呢？这是因为 ELF 标准没有定义调试信息的表示格式，所以才选择其他的标准来表示调试信息。当 ELF 文件里包含 DWARF 格式的调试信息时，常常称这个文件为 ELF/DWARF 1 或 ELF/DWARF 2 格式的文件。

dwarf 英文意思为有魔法的小矮人；elf 为小精灵。

编译器生成的目标文件由有魔法的小矮人和小精灵构成，这两个小人究竟是哪路神仙，在第 2 章会详细介绍。

1.2.2 链接器简介

在 ARM 系统开发中使用的链接工具为 armlink，armlink 涉及了几乎 ARM 开发的全部内容。它的每一个参数几乎都可以独立成章。如果用户掌握了它的全部，那么用户对 ARM 的掌握就达到了一个很高的层次。本节丢开细节，从总体上对它予以介绍。

1. 主要功能

简单地说，armlink 的工作就是合并一个或多个 ELF 格式的目标文件的内容，生成一个 ELF 格式的可执行文件(executable images)，可以细分为下面几个任务：

① 解析目标文件之间的符号引用关系。

② 如果目标文件包含 C/C++库函数的调用，从 C/C++运行时库(run-time library)中

---

* 通过访问网站 http://dwarf.freestandards.org 可以了解 DWARF 标准的详细信息。

提取相应的功能模块。

③ 将输入段按规则组成相应的输出段。

④ 如果目标文件里包含调试信息,则删除重复的调试信息。

⑤ 根据用户指定的分组和定位信息,建立映像文件的地址映射关系。

⑥ 重新定位需要重定位的变量的值。

⑦ 生成可执行的映像文件。

下面这条命令演示了armlink的基本用法,它把两个目标文件test_1.o和test_2.o链接成一个可执行的映像文件test.axf。

链接 armlink -ro-base 0x8000 test_1.o test_2.o -o test.axf。

其中,参数ro-base和o的具体作用如下:

-ro-base 表示代码段(RO)在装载域(load view)和运行域(execution view)里的起始地址;

-o 表示输出文件为test.axf。

对于装载域和运行域的具体含义,在第2章有详细的阐述。

1.2.3 fromelf工具

ELF格式的可执行的映像文件如test.axf,在调试通过后,如果需要下载到正式的产品上,必须经过fromelf工具处理,掐头去尾,留下真正有用的执行代码。它把armlink生成的ELF格式的映像文件转换成适合在ROM和RAM里运行的各种格式的二进制代码。在ARM系统里用的二进制格式是:Plain binary format。如果在编译源代码时,使用g参数,那么在编译器输出的目标文件和armlink输出的映像文件里将包含调试信息,这些信息可以帮助开发人员来调试代码、修改程序错误。但对于最终产品来说,它就成了占用资源的垃圾了。因此,在调试成功之后,在下载到最后的产品之前,必须去掉这些调试信息。fromelf就是用来完成这个功能的。

下面这个命令演示了fromelf的基本用法,它把ELF格式的映像文件test.axf转换成bin(Plain binary format)格式的二进制代码。

```
fromelf test.axf --bin -o test.bin
```

其中,--bin表示输出格式为Plain binary。

二进制映像文件test.bin诞生了,下一步要做的,就是把它下载到最终产品。

fromelf有一个参数:no_debug,可以去掉一个包含调试信息的可执行映像文件的调试信息。下面这条命令演示了fromelf的用法:去掉包含调试信息的可执行映像文件里的调试信息。

```
C:\arm\fromelf --no_debug --elf -o nodebug.axf debug.axf
```

当然，在默认情况下，fromelf在把可执行映像文件转换成二进制映像文件时，去掉了调试信息。

1.2.4 常用的编译和链接参数

1. 编译参数

(1) armcc*

用户在Windows的命令行执行armcc或者TCC编译器命令C:\arm>armcc(TCC)，就会列出主要的编译参数，如下所示：

ARM/Thumb C/C++Compiler,RVCT2.2 [Build 349] 编译器的版本号

Usage:armcc [option] file1 file2…file n 编译器的用法，其中的options就是各种编译参数。

主要[参数]的含义如下：

- --arm 编译器输出的为ARM指令集。
- --thumb 编译器输出的为Thumb指令集。
- --c90 编译的为C语言源程序。
- --cpp 编译的为C++语言源程序。
- -O0 最小的优化级别，关闭了除代码变换外的所有优化功能。
- -O1 关闭了那些影响调试效果的优化功能。
- -O2 高的优化级别，提供所有的优化功能。
- -Ospace 从空间上优化代码，目的是追求紧凑的目标代码。
- -Otime 从时间(执行效率)上优化代码，目的是使代码的执行效率最好。
- --cpu <cpu> 处理器的型号或者CPU的体系版本号，这样可以让编译器根据处理器的特点编译出最优的代码。
- --cpu list 输出可供选择的处理器列表。
- -o <file> 编译器输出的文件名字，它和下面三个参数结合使用，输出不同类型的文件。
- -c 对源程序只编译，不进行链接，与-o参数结合，编译输出目标文件，目标文件的名字就是-o参数指定的名字。如果没有使用-o参数，那么输出的目标文件名和源文件名相同。比如，源文件为hello.c，则编译输出的目标文件为hello.o。
- --asm 输出目标文件的同时，也输出相应的汇编代码。
- -S 只输出汇编代码，不输出目标代码。
- --interleave 和-asm或-S组合使用，同时输出汇编代码和源代码，如C:\mix>armcc

---

* 对于armcc更为详细的用法，可以参考文档：http://www.arm.com/pdfs/DUI0205G_rvct_compiler_and_libraries_guide.pdf。