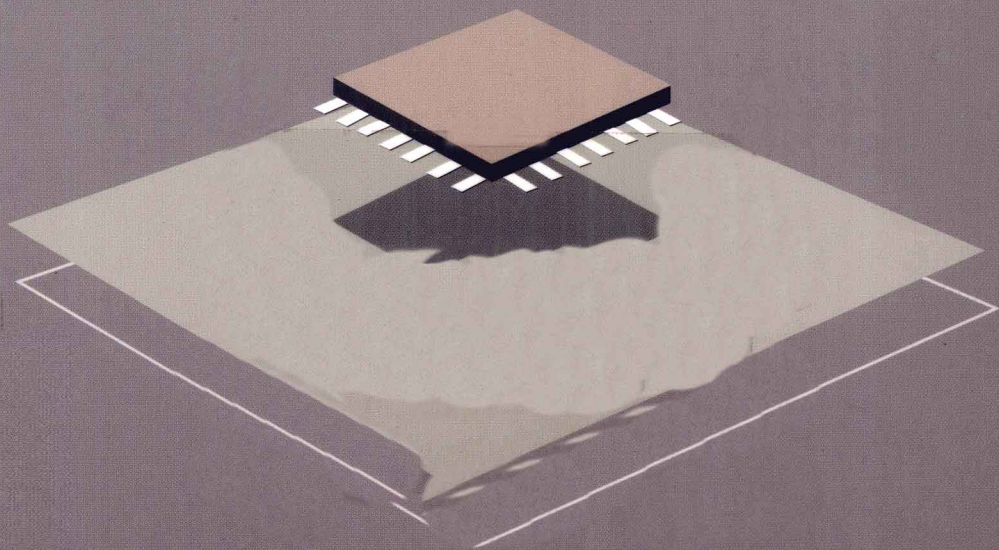


嵌入式系统基础

—ARM 与 Realview MDK
(Keil for ARM)

任 哲 张永忠 编著



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

嵌入式系统基础

——ARM 与 Realview MDK(Keil for ARM)

任 哲 张永忠 编著



北京航空航天大学出版社

内 容 简 介

本书在介绍微型计算机一般原理的基础上,重点介绍目前在应用中流行的 ARM RISC 体系结构及其嵌入式处理器,同时介绍当前流行的 ARM 工程开发工具 Realview MDK,并以该开发工具为基础介绍 ARM 汇编语言程序设计、工程开发及 RTX 操作系统的简单应用。

为了拓展学生的知识和培养学生的自学能力,本书还在附录中简要地介绍另一个常用的基于 ARM 体系结构的嵌入式处理器 S3C44B0X 的构成及其主要接口以及 ARM 体系结构的高级存储管理部分,以便为读者学习高档 ARM 处理器核建立必要的基础。

本书适合高等院校电气自动化、仪器仪表、电子技术等专业微机原理课程使用,也可作为其他以计算机嵌入式应用为教学目标的专业教学用书,当然也可供对 ARM 体系结构及其嵌入式处理器感兴趣的专业人士阅读、参考。

图书在版编目(CIP)数据

嵌入式系统基础:ARM 与 Realview MDK(Keil for ARM) / 任哲,张永忠编著. -- 北京:北京航空航天大学出版社,2012.2

ISBN 978-7-5124-0692-6

I. ①嵌… II. ①任… ②张… III. ①实时操作系统
IV. ①TP316.2

中国版本图书馆 CIP 数据核字(2011)第 278300 号

版权所有,侵权必究。

嵌入式系统基础

——ARM 与 Realview MDK(Keil for ARM)

任 哲 张永忠 编著

责任编辑 王慕冰 龚荣桂 王平豪

*

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(邮编 100191) <http://www.buaapress.com.cn>

发行部电话:(010)82317024 传真:(010)82328026

读者信箱:bhpress@263.net 邮购电话:(010)82316936

北京时代华都印刷有限公司印装 各地书店经销

*

开本:787×960 1/16 印张:27.75 字数:622 千字

2012 年 2 月第 1 版 2012 年 2 月第 1 次印刷 印数:4 000 册

ISBN 978-7-5124-0692-6 定价:56.00 元

若本书有倒页、脱页、缺页等印装质量问题,请与本社发行部联系调换。联系电话:(010)82317024

前 言

近年来,由于计算机技术的发展,嵌入式系统已在汽车、航空、航天、造船、仪器仪表、家用电器等领域得到了极其广泛的应用,进而也激发了社会对嵌入式系统人才的巨大需求。所以,近几年来我国各高等院校相继开设了嵌入式系统的相关课程。本书就是一本专为高等院校教学使用的、专门介绍 ARM 体系结构及其嵌入式处理器的教科书。

本书可以说是作者原作《ARM 体系结构及其嵌入式处理器》(北京航空航天大学出版社 2008 年出版)一书的改写版。本书除了保留原作起点低、通俗易懂、信息量大的特点之外,还借助书中指令系统及编程例题介绍了 ARM 工程开发方法和开发工具 Realview MDK(Keil for ARM)的使用;更具特色的是,本书借助开发工具的介绍以通俗的语言详细地介绍了 ARM 程序结构、映像文件、分散加载等一些其他书籍涉及较少,但又需要嵌入式系统开发者了解并掌握的计算机基本理论和知识。

全书共分 11 章。第 1、2 章为基础部分,重点介绍微型计算机系统的基本构成及基本工作原理;第 3~5 章介绍 ARM 体系结构、指令系统及汇编语言程序设计的基本知识;第 6 章介绍 ARM 工程及其开发工具;第 7 章在介绍微型机中断技术的基本概念基础上介绍了 ARM 的中断系统及其特点;第 8、9 章介绍基于 ARM 体系结构的嵌入式处理器 LPC2000 的构成、特点及其外围接口;第 10 章在简要介绍计算机固件的基础上介绍了 LPC2000 的固件;第 11 章介绍嵌入式操作系统基础及 MDK 提供的嵌入式操作系统 RTX 的简单应用。

为了拓展学生的知识和培养学生的自学能力,本书还在附录 C 中简要地介绍了另一个常用的基于 ARM 体系结构的嵌入式处理器 S3C44B0X 的构成及其主要接口,以使读者对 ARM 体系结构嵌入式处理器有一个更全面的认识;为保证 ARM 知识体系的完整性,在附录 D 中介绍 ARM 体系结构的高级存储管理,从而为读者进一步学习高档 ARM 处理器核建立了必要的基础。

本书编者为任哲、张永忠、章小卫和房红征,其中任哲和张永忠为主编。

本书在编写过程中参考和借鉴了大量的相关资料(见参考文献)及网络资源,并引用了其中一些文字和代码,在此谨对这些作者表示衷心的感谢。

由于作者水平有限,书中难免出现错误,希望广大读者能给予批评指正。作者的电子邮箱为 renzhe71@sina.com。

任 哲
2011 年 7 月

目 录

第 1 章 微型计算机基础知识	1	2.1.1 总线的基本概念	26
1.1 微型计算机的发展历程	1	2.1.2 系统总线	27
1.2 数字电路与计算机	2	2.1.3 系统总线结构	28
1.2.1 数据在计算机中的表示 ——二进制	2	2.1.4 片内总线	30
1.2.2 运算器及二进制数运算	2	2.2 存储器	30
1.2.3 数据的存储	8	2.2.1 半导体存储器的一般结构	31
1.2.4 指令及指令译码	10	2.2.2 随机读/写存储器	33
1.2.5 程序及程序计数器 PC	11	2.2.3 只读存储器	35
1.2.6 计算机执行程序的过程	12	2.2.4 存储器的逻辑表示	36
1.3 微型计算机系统的基本组成	13	2.3 外部设备及接口	37
1.3.1 冯·诺依曼计算机结构	14	2.3.1 外部设备及其特点	37
1.3.2 计算机的硬件系统	14	2.3.2 I/O 接口电路的功能	38
1.3.3 计算机的软件系统	17	2.3.3 I/O 设备接口电路的基本结构	42
1.4 计算机体系结构的发展	17	2.3.4 外部设备与处理器的联络和 数据传输	43
1.4.1 RISC	18	2.4 常用接口电路	45
1.4.2 指令流水线	20	2.4.1 并行接口电路	45
1.4.3 高速缓存	21	2.4.2 串行接口电路	47
1.4.4 协处理器	21	2.4.3 定时器/计数器	51
1.4.5 片上系统	21	习 题	53
1.5 微型计算机的两种主要应用方向	21	第 3 章 ARM 体系结构	55
1.5.1 桌面系统	21	3.1 ARM 及其嵌入式处理器的研发与 生产方式	55
1.5.2 嵌入式系统	22	3.1.1 SoC 与嵌入式处理器	55
1.6 计算机程序设计语言	22	3.1.2 嵌入式处理器的研发和生产 方式	56
1.6.1 低级语言	22	3.2 ARM 处理器核的结构	59
1.6.2 高级语言	23	3.3 冯·诺依曼结构及哈佛结构在 ARM 中的应用	60
1.7 计算机常用标准编码	23	3.4 ARM 处理器的运行模式	61
1.7.1 ASCII 码	23	3.5 ARM 的两种工作状态	62
1.7.2 BCD 码	24	3.6 ARM 处理器的寄存器	62
习 题	24		
第 2 章 总线、存储器和接口	26		
2.1 总 线	26		



3.6.1	ARM 状态下寄存器的组织方式	62	5.2.4	数据区定义伪指令	119
3.6.2	Thumb 状态下寄存器的组织方式	66	5.3	宏与宏指令	122
3.7	存储器的组织	67	5.3.1	宏	122
3.8	ARM 体系结构的其他特点	69	5.3.2	宏指令	126
3.8.1	灵活方便的协处理器接口	69	5.4	其他伪指令	131
3.8.2	嵌入式的在线仿真调试	69	5.4.1	有关程序结构的一些伪指令	131
3.8.3	低电压低功耗的设计	69	5.4.2	有关数据空间定义的一些伪指令	135
3.9	ARM 体系结构的版本及处理器系列	69	5.4.3	汇编控制伪指令	137
3.9.1	ARM 体系结构的版本	70	5.4.4	其他常用伪指令	138
3.9.2	ARM 处理器系列	71	5.5	汇编语言规范	139
习 题		72	5.5.1	汇编语句格式	139
第 4 章	ARM 指令系统	73	5.5.2	汇编语言的表达式和运算符	139
4.1	ARM 指令集的常用指令	73	5.6	ARM 汇编语言程序设计	143
4.1.1	数据传送指令	74	5.6.1	段	143
4.1.2	ARM 指令的附加操作	80	5.6.2	分支程序设计	143
4.1.3	跳转(转移)指令	81	5.6.3	循环程序设计	149
4.1.4	指令的条件码及条件指令	85	5.6.4	子程序及其调用	150
4.1.5	算术运算指令	88	习 题		151
4.1.6	逻辑运算指令	93	第 6 章	ARM 工程开发及 MDK	153
4.1.7	程序状态寄存器访问指令	96	6.1	ARM 工程及其开发工具	153
4.1.8	加载/存储指令	97	6.2	映像文件及程序	158
4.1.9	批量数据加载/存储指令	100	6.2.1	程序的内存布局	158
4.1.10	数据交换指令	104	6.2.2	程序的加载域和执行域	160
4.1.11	协处理器指令	106	6.2.3	映像文件的结构	166
4.1.12	异常产生指令	108	6.3	程序的分散加载	170
4.2	Thumb 指令简介	108	6.3.1	基本概念	170
4.2.1	Thumb 寄存器的使用	109	6.3.2	分散加载文件	171
4.2.2	ARM-Thumb 的交互	109	6.3.3	MDK 生成的分散加载文件	177
习 题		110	6.4	ARM 工程框架	185
第 5 章	ARM 汇编语言基础	111	6.4.1	初始化程序部分	185
5.1	汇编器与汇编语言	111	6.4.2	初始化部分与主应用程序部分的衔接	186
5.2	ARM 常用伪指令	112	6.5	C 程序与汇编程序之间的函数调用	189
5.2.1	段定义伪指令	112	6.5.1	ATPCS 简介	190
5.2.2	符号定义伪指令	114	6.5.2	汇编程序调用 C 函数实例	197
5.2.3	程序中的标号	117			

6.5.3 汇编程序访问全局 C 变量	199	8.6.2 锁相环及处理器时钟 cclk 参数的设置	247
6.6 C/C++语言和汇编语言的混合编程	200	8.6.3 pclk 时钟参数的设置	250
6.6.1 内联汇编	200	8.7 定时器	250
6.6.2 嵌入式汇编	201	8.7.1 定时器的原理和功能	251
6.6.3 内联汇编代码与嵌入式汇编代码之间的差异	202	8.7.2 定时器的寄存器	252
习 题	203	8.7.3 定时器的初始化	254
第 7 章 中断和异常	205	8.8 LPC2000 的脉宽调制器 PWM	256
7.1 中断和异常的基本概念	205	8.8.1 LPC2000 脉宽调制器原理	256
7.1.1 中断和异常	205	8.8.2 脉宽调制器的寄存器	257
7.1.2 中断请求信号的屏蔽	206	8.8.3 LPC2000 脉宽调制器的使用示例	259
7.1.3 中断优先级及中断嵌套	207	8.9 看门狗	260
7.1.4 中断服务程序	208	8.9.1 看门狗的结构及原理	260
7.1.5 中断向量和中断向量表	209	8.9.2 看门狗寄存器	261
7.1.6 中断的处理过程	211	8.9.3 看门狗的使用	261
7.2 ARM 的中断(异常)	211	8.10 功率控制模块	261
7.2.1 ARM 中断(异常)的种类	211	8.11 LPC2000 的 UART 接口	263
7.2.2 ARM 中断(异常)的优先级	213	8.11.1 UART 接口的结构	263
7.2.3 ARM 的中断(异常)向量表	214	8.11.2 UART 接口的寄存器	264
7.2.4 中断(异常)的响应过程及返回	217	8.11.3 UART1 接口与 Modem	267
7.3 软中断 SWI 的应用	222	8.11.4 UART 接口的初始化及其应用	268
7.3.1 软中断 SWI 的一般应用	222	8.12 LPC2000 的 SPI 接口	270
7.3.2 SWI 函数(系统调用)	228	8.12.1 SPI 简介及 LPC2000 的 SPI 接口逻辑	270
习 题	231	8.12.2 LPC2000 SPI 接口的寄存器	271
第 8 章 LPC2000 系列嵌入式处理器	232	8.12.3 LPC2000 SPI 接口应用实例	273
8.1 LPC2000 嵌入式处理器概貌	232	8.13 LPC2000 的 I ² C 接口	276
8.2 引脚连接模块	233	8.13.1 I ² C 总线简介	276
8.3 通用可编程并行数据接口 GPIO	234	8.13.2 LPC2000 的 I ² C 总线接口	278
8.4 LPC2000 的存储器	239	8.13.3 I ² C 接口的工作过程	280
8.5 外部存储器的连接	242	8.14 A/D 转换器	282
8.5.1 EMC 的结构	242	8.15 LPC2000 的中断管理	286
8.5.2 外部存储器的连接	244	8.15.1 外部中断通道	287
8.6 LPC2000 的时钟	246	8.15.2 中断控制器 VIC	289
8.6.1 LPC2000 时钟模块	246		



习 题	296	寄存器	360
第 9 章 LPC2000 外部电路	298	C.3 时钟和功耗管理	363
9.1 复位电路	298	C.3.1 时钟设置	363
9.2 人机交互设备	299	C.3.2 功耗管理控制	364
9.2.1 键盘与触摸屏	299	C.3.3 功耗管理控制寄存器	365
9.2.2 显示器	303	C.4 中断管理	366
习 题	308	C.4.1 中断源与中断系统结构	366
第 10 章 LPC2000 的固件	309	C.4.2 中断控制寄存器	366
10.1 LPC2000 的重映射存储区	309	C.4.3 中断源优先排队模块	368
10.2 LPC2000 的固件及其作用	310	C.5 DMA 控制器	370
10.3 LPC2000 引导块	313	C.5.1 ZDMA 和 BDMA	370
10.3.1 LPC2000 引导块的地址映射	313	C.5.2 与 ZDMA 相关的寄存器	372
10.3.2 向量表的重映射	314	C.5.3 与 BDMA 相关的寄存器	373
10.3.3 LPC2114/2124 的 Boot Block 的工作流程	314	C.6 LCD 控制器	375
10.3.4 有效用户程序的识别	316	C.6.1 概 述	375
10.4 RAM 空间的重映射	317	C.6.2 LCD 控制器的控制寄存器	378
习 题	317	C.6.3 帧缓冲区起始地址寄存器	379
第 11 章 嵌入式操作系统基础	319	C.7 IIS 总线接口	381
11.1 基本概念	319	C.7.1 IIS 总线接口框图	381
11.1.1 什么是操作系统	319	C.7.2 IIS 接口工作模式	383
11.1.2 嵌入式操作系统	324	C.7.3 IIS 接口的特殊功能寄存器	383
11.2 RTX 构成及其应用基础	325	C.8 S3C44B0X 的引脚与 I/O 端口	386
11.2.1 RTX 中的任务	325	C.8.1 端口配置寄存器 PCON	386
11.2.2 时间片、优先级及调度	329	C.8.2 端口数据寄存器 PDATA~ PDATG	387
11.2.3 任务的状态	336	C.8.3 上拉寄存器 PUPC~PUPG	387
11.2.4 RTX 的任务间通信	339	习 题	387
11.2.5 RTX 的数据类型	350	附录 D ARM 高级存储管理	389
习 题	351	D.1 高速缓冲存储器	389
附录 A LPC2148 引脚说明	352	D.1.1 Cache 的基本概念和工作原理	389
附录 B MDK 中的文件	356	D.1.2 Cache 与主存之间的关系	392
附录 C S3C44B0X 处理器简介	358	D.1.3 Cache 的写缓冲器	393
C.1 S3C44B0X 的结构	358	D.2 协处理器	394
C.2 S3C44B0X 的存储器	359	D.2.1 ARM 的协处理器	394
C.2.1 S3C44B0X 存储映射	359	D.2.2 协处理器操作指令	395
C.2.2 S3C44B0X 用于存储管理的		D.3 ARM 存储器保护单元 MPU	396

D. 3.1 ARM 的 MPU	397	D. 4.5 采用虚拟存储技术的优点	419
D. 3.2 协处理器 CP15 及保护区域的 定义	397	D. 5 ARM 的虚拟存储管理	420
D. 3.3 保护区域的重叠应用	403	D. 5.1 ARM 的页表结构	420
D. 3.4 MPU 应用示例	404	D. 5.2 访问控制	424
D. 4 虚拟存储与 MMU	411	D. 5.3 MMU 的配置	426
D. 4.1 虚拟存储空间与物理内存空间 ...	411	D. 5.4 页 Cache 和写缓冲器的设置 ...	426
D. 4.2 地址映射机构	412	D. 5.5 快表	427
D. 4.3 页表的缓存——快表	416	D. 5.6 MMU 异常	428
D. 4.4 页表的存储及二级页表	417	习 题	429
		参 考 文 献	431

第1章 微型计算机基础知识

自1946年世界上出现第一台数字电子计算机ENIAC之后,它就以强大的运算能力使世界感到震惊。其后,经过半个多世纪的飞速发展,计算机技术应用已经深入到人们生产和生活的各个方面,以至于不管你是否情愿,都要与之打交道。

本章的主要内容有:

- 数字电子电路是计算机硬件的基础;
- 二进制数制及机器数;
- 有符号数的表示及溢出的概念;
- 存储器的基本电路及其作用,存储器存储单元的地址;
- 运算器、控制器的基本概念;
- 指令及指令系统,计算机程序的执行过程;
- 计算机的硬件组成;
- SISC和RISC架构;
- 微型计算机的桌面系统应用与嵌入式应用。

1.1 微型计算机的发展历程

根据计算机所使用的电子器件,计算机的发展经历了电子管、晶体管、集成电路、大规模集成电路和超大规模集成电路等几个阶段。同时,计算机在体系结构、组成方式及其体积、性能、价格、应用领域等方面也发生了重大变化。目前,计算机可分为巨型机、大型机、中型机、小型机和微型机这5类。其中,人们接触最多的是叫做“微机”的微型计算机。

微型计算机出现在20世纪70年代。1971年,以Intel公司设计的世界上第一个微处理器芯片Intel 4004为标志,电子数字计算机的发展便进入了微型计算机时代。微型计算机一出现,就表现出了强大的生命力,在短短的几十年间就经过了从第一代到第五代的发展历程:第一代是以4004、4040和8008微处理器为代表的字长为4位和8位的微型计算机;第二代是以微处理器Z80、I8085、M6800和Apple-II等为代表的中高挡8位机;第三代是以8086、8088、80286微处理器为代表的16位机;第四代是以80386、80486、Pentium、Pentium II、



Pentium III、Pentium IV 等微处理器为代表的 32 位机；第五代则是以 Itanium、MIPS 为代表的具有精简指令架构(RISC)的 64 位机。

1.2 数字电路与计算机

电子数字计算机,是一种用电子装置构成,能够按照机器操作者输入的命令进行数字信息处理的机器。从功能上看,它具有两个基本功能:一是表示和存储数字数据的功能;二是对数字数据进行运算的功能。从物理构成的角度来看,它是一种由数字电子器件和电路构成的运算装置。

1.2.1 数据在计算机中的表示——二进制

作为计算机的硬件基础——数字电路,只能用电路信号端子的电信号(电平)来表示数字信息,即规定,低电平代表数码 0(低电平也叫做电路的“0”状态),高电平代表数码 1(高电平也叫做电路的“1”状态),所以它的一个信号端也就只能表示两个数码——0 和 1。但是,如果电路有两个信号端子,那么这两个端子就可以有 4 个状态——00、01、10 和 11,如果用 00 表示十进制数 0,用 01 表示十进制数 1,用 10 表示十进制数 2,用 11 表示十进制数 3,那么这个电路就能表达 4 个十进制数了。这就是说,一个数字电路的信号端子越多,其状态就越多,所能表达的数也就越多。稍微分析一下就会知道,一个具有 n 个信号端子的数字电路所具有的状态数为 2^n ,那么它所能表达的数的个数就为 2^n 个。

例如,有 8 个信号端子的数字电路就可以有如下的 256 个状态,并可像下面那样表示 256 个十进制数 0~255:

0000 0000 表示十进制数 0;

0000 0001 表示十进制数 1;

0000 0010 表示十进制数 2;

0000 0011 表示十进制数 3;

0000 0100 表示十进制数 4;

⋮

1111 1111 表示十进制数 255。

从上面左边的表示中可以看到,电路端子的表示是以“逢 2 进 1”为加法运算规则,以“借 1 当 2”为减法运算规则的二进制数制。

1.2.2 运算器及二进制数运算

计算机的核心任务是进行数的运算,该功能是由一个叫做算术逻辑运算单元(ALU)的电路来实现的。

1. 算术逻辑运算单元 ALU 的概念

最基本的算术运算是加法运算,其他各种运算都可以通过加法运算来完成,因此二进制的加法运算就是计算机的核心运算功能。当然,这个二进制运算要由数字电路来完成。按照逢 2 进 1 的二进制加法运算规则,两个 1 位二进制数 A 和 B 相加时,其加法运算电路的真值表如表 1-1 所列。

这种加法运算产生了两个信息,即两数和 S 及进位 C。

显然,作为能进行完整加法运算的加法器,应该能进行被加数、加数、低位进位三个数的相加运算,即一个能进行完整加法运算的加法器的真值表应该如表 1-2 所列。

表 1-1 一位二进制加法的真值表

输入		输出	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

表 1-2 带有进位值的一位二进制数加法真值表

输入			输出		输入			输出	
A_i	B_i	C_{i-1}	S_i	C_i	A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0	0	0	1	0	1
0	1	0	1	0	0	1	1	0	1
1	0	0	1	0	1	0	1	0	1
1	1	0	0	1	1	1	1	1	1

为了区别,把具有表 1-1 运算功能的加法器叫做半加器,而具有表 1-2 运算功能的加法器叫做全加器。

根据表 1-2 的真值表,可以用数字电路构成一个如图 1-1(a)所示的全加器。而用多个全加器就可以构成如图 1-1(b)所示的多位二进制数加法器。

图 1-1(b)所示的多位二进制数加法器就是计算机的核心部件,如果在此基础上配置一些其他辅助电路就构成计算机 CPU 中的算术逻辑运算单元 ALU。

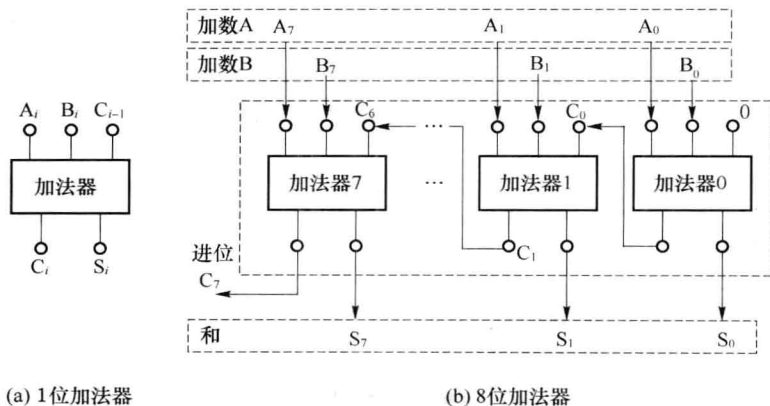


图 1-1 数字加法器示意图

由于进行数的运算时,还会产生一些与运算过程和结果相关的其他信息,例如运算中是否发生了进位、运算结果是否为零,等等。为了记录这些信息,ALU 中还设有一个专门保存上述信息的存储装置,这个存储装置叫做程序状态寄存器。

ALU 和程序状态寄存器的示意图如图 1-2 所示。

2. 机器数

显然,是由于计算机硬件电路的物理限制,才使得计算机不得不用这种人们既不熟悉也不方便的二进制数,所以这种二进制数也常常叫做机器数。

凡使用机器来计数的装置都有一个共同的特点:它们的字长是有限的,或者说它们的计数范围是有限的,即当计数计到最大值再进行加法计数时,其和会返回(回卷)到 0 重新计数。

例如图 1-3 所示的钟表的表盘,当顺时针计数(加法计数)计到最大值 11 后,如果再加 1,则从表盘得到的值是 0 而不是 12(当然,真实的表盘写的还是 12,而不是人们不喜欢的 0,但实质是 0)。计算机的计数器和运算器也有同样的特点,例如,8 位二进制计数装置在进行加法和减法计数时的变化情况如图 1-4 所示。

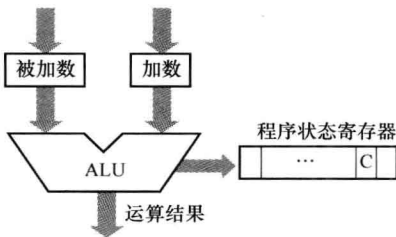
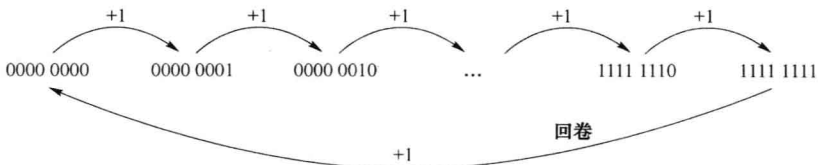


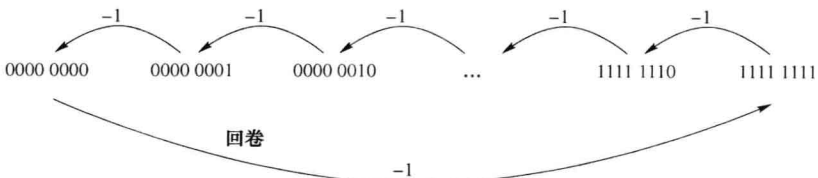
图 1-2 ALU 及程序状态寄存器的示意图



图 1-3 表盘



(a) 8位二进制加法计数时数的变化情况



(b) 8位二进制减法计数时数的变化情况

图 1-4 8 位二进制计数时数的变化情况

3. 有符号数的机器数表示及其运算

前面谈到的都是无符号数的二进制形式,那么机器数的有符号数又如何表示呢?下面以8位二进制为例来进行说明。前面已经知道,8位二进制无符号数为:

0000 0000 表示十进制数 0;
 0000 0001 表示十进制数 1;
 0000 0010 表示十进制数 2;
 0000 0011 表示十进制数 3;
 0000 0100 表示十进制数 4;
 ⋮
 1111 1111 表示十进制数 255。

众所周知,在十进制数制中,0 减去某个正数得到的就是与这个正数对应的负数,例如 0 减去 1 就是与 1 对应的负数-1,0 减去 2 就是与 2 对应的负数-2。那么,从图 1-4 中可知,8 位二进制的 0000 0000 减 0000 0001 的结果 1111 1111 就应该是对应十进制的-1,如果从 1111 1111 再减 0000 0001,即减 0000 0010 得到的 1111 1110 就应该十进制的-2。以此类推,可以一直减到 1000 0000(对应十进制的-128)。这就是说,0000 0000~1111 1111 这 256 个数既可以看成正数也可以看成负数;也就是说,0000 0000~1111 1111 只是一种表示而已,至于它们究竟是正数还是负数,则是由使用数的人来解释的。但是,作为一种通用运算装置,对于计算机中的有符号数最好还是有一个通用的规则。由于通常希望在一个计数规则中的正数和负数的数目最好基本相等,于是在表达有符号数时,人们就把 0000 0000~1111 1111 这 256 个数里那些最高位为 1 的二进制数看成负数,而把最高位为 0 的那些二进制数看成正数(0000 0000 除外),这样这 256 个数就可以 0000 0000 为界分为如下两部分:

0111 1111 表示十进制数 127	}	正数
⋮		
0000 0011 表示十进制数 3		
0000 0010 表示十进制数 2		
0000 0001 表示十进制数 1		
0000 0000 表示十进制数 0	}	负数
1111 1111 表示十进制数-1		
1111 1110 表示十进制数-2		
1111 1101 表示十进制数-3		
⋮		
1000 0000 表示十进制数-128		



由于所有正数的最高位都为 0,所有负数的最高位都为 1,因此可以用数的最高位来判别有符号数的正负,于是就可以把有符号数的最高位叫做符号位,而把剩下的所有位叫做数值位。同时,为了讨论问题方便,把有符号数所对应的十进制值叫做机器数的真值。

仔细观察就会发现,在 8 位有符号数中,所有负数与其对应的正数相加,其结果都为 10000 0000(对应十进制的 256),于是就把这个值叫做 8 位二进制数的模。这也意味着 8 位有符号数的负数与其对应的正数是以 10000 0000 为模互补的,所以上述这种有符号数的表示形式叫做有符号数的补码形式。

虽然上面有符号数是用 8 位二进制数来介绍的,但其规律同样可以推广到其他字长的计数规则中。

在十进制计数中,人们熟悉的只是用真值来表示负数的方法,而不熟悉上述这种补码表示方式,为了方便从真值求取负数的补码,又引入了反码的概念。所谓反码,就是把一个二进制数按位取反所得到的数码。例如,0001 0101 的反码为 1110 1010。有了这个反码的概念后,再求取负数的补码形式就方便得多。具体方法为:先把一个负数的真值的绝对值用二进制形式表示出来,然后把这个二进制数按位取反,最后再加 1,得到的就是该负数的补码形式。例如,一个有符号数的真值为 -33,为了求得它的 8 位二进制补码,就先写出其绝对值 33 的二进制形式 0011 0011,然后按位取反得 1100 1100,最后再加 1,得到的就是 -33 的二进制补码形式 1100 1101。

采用上述补码的最大优点是,计算机的运算器在做二进制的加法运算时,无须对两个加数是否为有符号数进行区分,只要简单地把这两个数相加就可以了,至于是有符号数运算还是无符号数运算则由用户来解释。所以,要注意,有符号数的最高位(符号位)也是参与运算的。

计算机之所以采用二进制的补码形式,就是为了把有符号数与无符号数的运算规则统一起来。因为减法运算就是一个加负数的运算,这样,运算器只要具有加法运算功能就可以了,而没有必要再另外设计减法器。

例如,1001 0011(如果看成无符号数,则该数为 147;如果看成有符号数,则该数为 -109)和 0011 1111(如果看成无符号数,则该数为 63;如果看成有符号数,则该数仍为 63)相加的结果为 1101 0010(如果看成无符号数,则该数为 210;如果看成有符号数,则该数为 -46)。可以看到,所有结果都是正确的。

从上面的介绍中已经知道,在有符号数中,二进制形式中的最高位为符号位,也就是说,有符号数的数值位比数据的整个位数少了 1 位。例如在 8 位有符号数中,它们的数值位只有 7 位,这样就使得 8 位有符号正数的数值范围为 $0 \sim 127$,负数的数值范围为 $-128 \sim -1$,因此,在 8 位有符号数的运算过程中,其运算结果的数值不得超过上述范围,否则得到的是一个错误结果,这种现象叫做溢出。为了在出现溢出时能为用户提供相应的提示信息,运算器应该具有判断溢出的能力。图 1-5 给出了计算机运算器用双进位法来判断溢出的示意图。图中 C_2 是运算结果最高位 D_{n-1} 向进位位产生的进位信号, C_1 是运算结果数值位的最高位 D_{n-2} 向 D_{n-1}

产生的进位信号,如果这两个信号相同,则无溢出,否则有溢出。即:

$$OF = C_1 \oplus C_2$$

运算器用上述方法来判断运算是否有溢出,其溢出信息被存入程序状态寄存器。

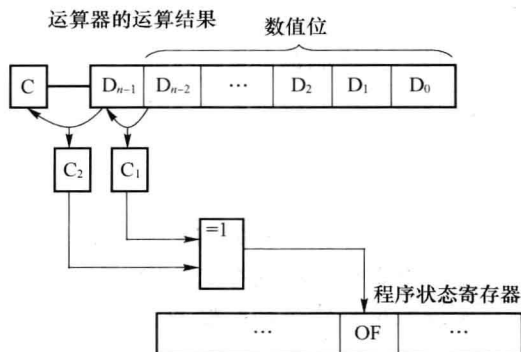


图 1-5 双进位位法判断溢出的示意图

4. 二进制数据的十六进制表示

显然,数的二进制表示方式有一个缺点:一个很小的数的二进制表示不仅很长,而且不便于阅读、书写和记忆。为此,常常把一个较长的二进制数分成若干组,每组 4 位二进制数码,然后把每组所表示的数用一位数码来表示。由于 4 位二进制数 0000~1111 对应的十进制数为 0~15,是两位数,所以又引入了 6 个大写英文字符 A、B、C、D、E、F 作为数码来分别表示 10、11、12、13、14 和 15,从而形成了二进制数的十六进制表示方式。4 位二进制数对应的十六进制数码见表 1-3。

表 1-3 4 位二进制数对应的十六进制数和十进制数

二进制	十六进制	十进制	二进制	十六进制	十进制
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

例如,上面 8 位二进制的 256 个数,用十六进制方式表达出来就是:



0x00 表示二进制数 0000 0000,十进制数 0;
 0x01 表示二进制数 0000 0001,十进制数 1;
 0x02 表示二进制数 0000 0010,十进制数 2;
 0x03 表示二进制数 0000 0011,十进制数 3;
 ⋮
 0xFF 表示二进制数 1111 1111,十进制数 255。

为了区别,在十六进制数的前面要加上前缀 0x。

1.2.3 数据的存储

人们在进行数的运算时,为了记录运算所需的初始数据、中间运算结果和最终结果,一般需要一张纸,这张纸实质上就是一个存储数据的存储装置。同样,计算机在进行运算时也需要相应的数据存储装置,这种装置就叫做计算机的存储器。

1. 存储器的基本电路

原则上说,任何能接收并且能保存数字信号的装置都可以作为计算机的存储装置。而数字电路中的双稳态电路就是这样一种可以存储二进制信息的装置。

例如图 1-6 所示的一个双稳态电路,在选择信号使开关 K1 和 K2 接通时,就会使 D 端的输入传送到 A 端。即当 D 端输入信号为高电平“1”(或低电平“0”),则会使双稳态电路的 A 端为“1”(或“0”),B 端为“0”(或“1”)。通过数字电子电路中所学的知识知道,由于 Q1 和 Q2 的交叉耦合作用,即使以后开关 K1 和 K2 断开,电路 A 端和 B 端的电平也不会发生变化。也就是说,电路可以把输入的信息保存下来,并且在其后的某个时刻如果再度接通开关 K1 和 K2,则会在 D 端测得 A 端的“1”(或“0”)信号。这就是说,这种电路可以输入(写)和保存(存储)电平信号,而且在需要时还可以输出(读)该信号。因此,图 1-6 所示的电路就是用来存储一位二进制数字的基本存储电路。

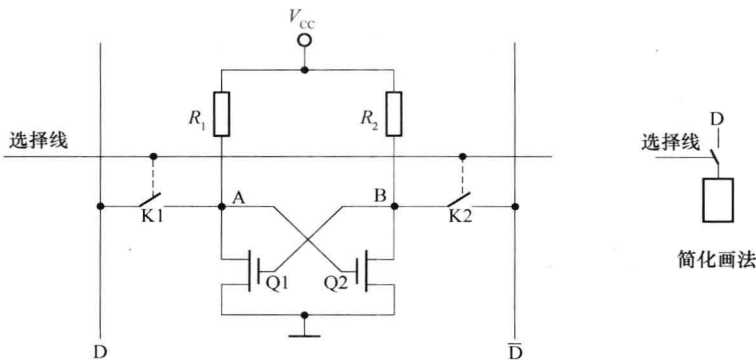


图 1-6 双稳态电路