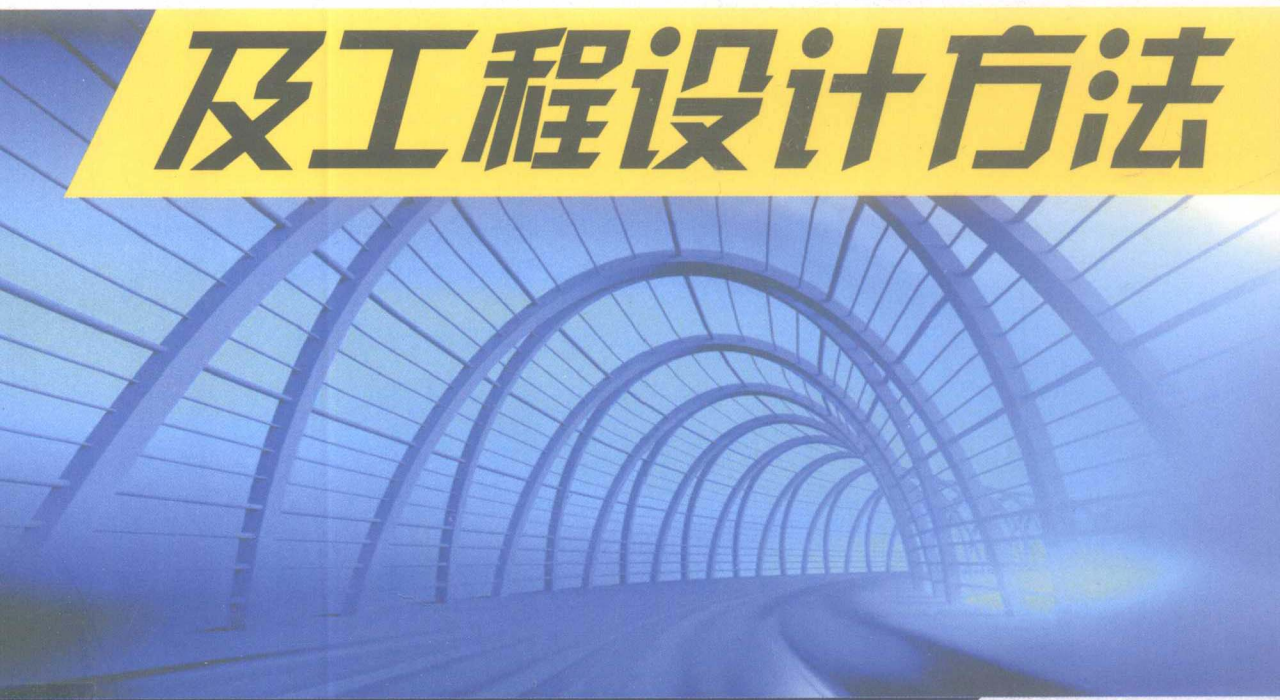
DANPIANJI
YUANLI JI GONGCHENG
SHEJI FANGFA

单片机原理及工程设计方法



鲜浩 任爱芝 史建华 编著

国防工业出版社
National Defense Industry Press

单片机原理及工程设计方法

鲜 浩　任爱芝　史建华　编著

国防工业出版社

·北京·

内容简介

本书从单片机原理和工程设计方法两个方面进行了阐述。

首先阐述了CPU与存储器的组织原理，使读者能够深刻理解嵌入式微处理器硬件系统设计的原理、思想与方法；然后以8051单片机为例，阐述了8051CPU与存储器的组织及访问、汇编语言程序的设计与仿真、C51程序的设计与仿真（并特别深入地讲解了C51语言）、Keil uVision集成开发环境的使用、程序调试技术的一般思想与方法、单片机硬件接口的设计思想与方法、并行端口的工程应用、8051单片机的中断系统及其工程应用、8051单片机定时器/计数器的工程应用、8051单片机串行口的工程应用等内容。目的就是为了使读者能够切实地掌握单片机原理及工程设计方法。

本书可作为高等学校通信工程、电子信息工程、测控技术与仪器、自动化、电气工程、机械工程及计算机科学与技术等专业学生的自学用书或参考书，也可作为从事单片机系统开发应用的工程技术人员及单片机爱好者的自学用书或参考书。

图书在版编目（CIP）数据

单片机原理及工程设计方法/鲜浩，任爱芝，史建华编著. —北京：国防工业出版社，2012.7

ISBN 978-7-118-08204-3

Ⅰ.①单… Ⅱ.①鲜…②任…③史… Ⅲ.①单片微型计算机－基本理论②单片微型计算机－系统设计 Ⅳ.①TP368.1

中国版本图书馆CIP数据核字（2012）第143456号

※

国防工业出版社出版发行

（北京市海淀区紫竹院南路23号 邮政编码100048）

北京奥鑫印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 **印张** 18¾ **字数** 428千字

2012年7月第1版第1次印刷 **印数** 1—4000册 **定价** 35.00元

（本书如有印装错误，我社负责调换）

国防书店：（010）88540777 发行邮购：（010）88540776

发行传真：（010）88540755 发行业务：（010）88540717

前　言

嵌入式技术具有非常广泛的应用范围，很多产品都可以定义为嵌入式系统，如手机、电子字典、可视电话、数字相机、数字摄像机、机顶盒、游戏机、智能玩具、数控设备或仪表、汽车电子、家电控制系统、医疗仪器、航天航空设备等。

不管是电类专业还是非电类专业（尤其是电类专业），嵌入式技术对于就业的意义是不言而喻的。

8051 单片机技术是嵌入式技术的入门技术，其原因是 8051 单片机应用最早、影响最深远、品种最多、相对简单，最合适的初学者。

单片机技术的工程实践性非常强，读者学习单片机技术的目的也是为了工程设计，解决实际问题。

编写此书的目的就是为了使读者能够切实掌握单片机原理及工程设计方法，除了引用必要的相关手册的资料（如指令、开发工具、器件参数等）外，书中全是作者多年教学和工程实践的总结。本书侧重于单片机原理以及工程设计方法，因此，在阅读本书时，需要读者深刻理解相关原理与思想方法；涉及到硬件方面的知识时，必须认真阅读相关器件的数据手册，理解其电气参数的物理含义以及工程应用；涉及到软件设计时，必须动手熟悉使用 Keil uVision 集成开发环境，必须认真阅读开发工具链的用户指南或手册（Keil uVision 集成开发环境中的“Complete User's Guide Selection”包含了所有工具链的用户手册、开发环境的使用指南），尤其是调试技术，必须动手进行大量的实践。另外，需要特别强调的是，在实际工程中，绝大多数的资料都是英文的，读者必须面对这一现实，不能回避。

本书没有追求大而全，也不可能做到大而全，主要侧重于单片机原理和工程设计的思想与方法。读者通过应用这些原理和思想方法，就能够学习和使用所有的单片机，这是编者著作本书最主要的目的。因为，嵌入式处理器非常多，不同的工程应用使用不同的嵌入式处理器，读者只要能够做到工程应用需要使用什么嵌入式微处理器，就能很快掌握，解决实际的应用问题，这样才能胜任实际工作。

本书共分为 7 章。

第 1 章，CPU 与存储器的组织原理。本章是理解所有嵌入式微处理器硬件系统的组织原理，与具体的单片机无关。主要是为了让读者知道应该从哪些角度来理解所有的嵌入式微处理器的硬件架构。掌握了本章内容，就能够理解嵌入式微处理器硬件系统设计的原理、思想与方法。

第2章,8051CPU与存储器的组织及访问。阐述了CPU与存储器的组织原理在8051单片机上的具体体现。其他单片机的学习方法与此类似。

第3章,8051单片机的程序设计与仿真。在实际工程中,单片机的软件开发基本上都是基于Keil uVision集成开发环境的。本章分别讲述了汇编语言与C51语言在该环境下的程序设计与仿真,特别阐述了工具链的重要作用。其他单片机的程序设计方法类似,即都必须理解和熟悉程序开发工具链的使用,充分应用开发系统提供的各种资源,特别是库函数、仿真工具。另外,提出了在工程设计中程序调试技术的一般思想与方法。

第4章,单片机硬件接口的设计思想与方法以及并行端口的工程应用。阐述了单片机硬件接口的设计思想与方法,在工程设计具有共性,以便读者能够使用其他单片机。侧重于并行端口如何以I/O方式(而不是总线方式)与外部设备进行输入、输出,并行端口如何模拟时序(虽然只给出了LCD的时序模拟,其他器件的时序模拟方法完全一样)。体会程序设计是描述客观事物工作原理与工作过程这一核心思想。

第5章,8051单片机的中断系统。首先阐述了中断系统的一般结构与工作原理,其目的是为了读者能够理解所有嵌入式微处理器的中断系统;然后以8051单片机的中断系统为例进行具体说明。通过中断扫描触点开关说明外部中断的工程应用,从而体会程序设计是描述客观事物工作原理与工作过程这一核心思想。

第6章,8051单片机定时器/计数器的工程应用。首先阐述定时器/计数器的一般结构与工作原理,以便于读者能够应用其他单片机类似的片上外围。通过定时器/计数器的工程应用来说明在工程实际中,程序应该如何准确描述客观事物的工作原理与工作过程。

第7章,8051单片机串行口的工程应用。详细阐述了串行通信的重要原理以及串行通信在工程应用中的一般方法,并举了两个实例来说明串行口在工程如何应用。

每章后面的思考题与习题,主要用来引导读者如何阅读每章的内容,同时强调读者必须动手实践。

中北大学的鲜浩老师编著了本书的第1章、第2章;中北大学的任爱芝老师编著了本书的第3章、第4章的4.1节、4.2节;大同大学的史建华老师编著了本书的第4章的4.3节、第5章~第7章。由于作者水平有限,书中难免有疏漏之处,敬请广大读者提出意见和建议,可发至电子信箱:xianhao@nuc.edu.cn。

编者

2012年4月

目　录

第 1 章　CPU 与存储器的组织原理

计算机硬件系统的组织原理，其实质是 CPU 与存储器的组织原理：将所有的硬件设备一律视为存储器，通过总线连接到 CPU 上。因此，计算机硬件系统组织的核心是 CPU。

1.1　CPU 的组成与工作原理

CPU 是一种高速数字器件，具有运算功能和控制功能。因此，从功能上看，CPU 由运算器、控制器以及为运算器和控制器服务的一系列寄存器组成。

为了让 CPU 的使用者能够根据实际应用的各种需要灵活应用 CPU 的各种功能，CPU 的设计者将 CPU 的绝大部分功能按照某种格式编成了指令集，每条指令对应了 CPU 的一种功能，向 CPU 输入一条指令，CPU 就执行该指令规定的功能。用户可以根据实际应用的各种需要，使用多条指令形成程序，让 CPU 执行某种特定的功能。

让 CPU 实现某种特定功能的具体方式：用户首先将程序存储在外部存储器中，然后 CPU 从外部存储器中将指令逐条读入并执行指令规定的功能。

存储器由多个存储单元组成，存储单元是信息存储的实体。CPU 传输与处理的信息分为指令与数据两类，存储指令的存储单元称为指令存储单元；存储数据的存储单元称为数据存储单元。

1.1.1　CPU 的控制器

CPU 的控制器的控制功能有控制指令的读取、控制指令的译码与执行、控制数据的传输。

1.1.1.1　控制指令的读取

CPU 要执行的程序存储在外部存储器中，存储器由多个存储单元组成。由于存储器的存储单元数量很大，CPU 必须确定所要读取的指令存储在存储器的哪个存储单元中。采用的方法是：CPU 为外部存储单元提供了一组编号，CPU 每输出一个编号，通过一个专门电路，CPU 都能唯一选中一个外部存储单元进行访问。在计算机中，这种编号称为地址，将地址与外部存储单元进行对应的专门电路称为地址译码电路。

这样，CPU 要访问外部的哪个存储单元，CPU 就可以输出该存储单元的地址，通过专门设计的地址译码电路选中这个存储单元，然后进行信息传输。

为了读取指令，CPU 的控制器专门使用一个寄存器用于存储外部指令存储单元的地址，这个寄存器称为指令地址寄存器(Instruction Address Register，IAR)。另外，CPU 的控制器还专门使用一组寄存器队列来存储从外部存储器中读回来的指令，这组寄存器队列称为 CPU 的指令队列。指令队列是一个先入先出的队列，即先读入的指令先被控制器

译码和执行。

CPU 输出指令地址寄存器的值作为外部存储单元的地址，通过地址译码电路选中相应的存储单元，并将该存储单元所存储的信息读入 CPU 的指令队列，这种信息传输操作称为 CPU 指令读取操作。

对于 CPU 而言，只有指令读取操作才是指令传输，其余所有的信息传输操作，CPU 一律视为数据传输。因此，计算机系统中信息传输形成了指令流与数据流两类。如 PC，要运行存储在硬盘上的程序时，该程序先以数据流的形式从硬盘拷入内存中，然后再以指令流的形式读入 CPU 执行。

CPU 是通过输出地址来选择外部存储单元进行信息传输，指令地址寄存器的值就是所要读取的外部指令存储单元的地址。指令地址寄存器的值不同，CPU 就从外部不同的指令存储单元中读取指令，因此，控制器控制指令读取的实质是控制指令地址寄存器值的修改。

控制器控制指令地址寄存器值的修改有三种方式：硬件复位方式、指令控制方式、硬件中断方式。

1. 硬件复位方式

硬件复位是指由硬件电路触发，使 CPU 进入规定的初始工作状态(称为复位状态)。硬件复位的具体方式：给 CPU 上电，或给 CPU 复位引脚施加规定的复位信号。

CPU 复位以后，CPU 的控制器赋给指令地址寄存器一个固定的值，这个值称为指令地址寄存器的复位值。CPU 复位以后，CPU 始终从地址为指令地址寄存器复位值的存储单元中开始读取指令并执行。因此，指令地址寄存器的复位值是整个软件系统的入口。

需要强调的是，不同的 CPU，指令地址寄存器具体的名称可能不同，CPU 复位以后，指令地址寄存器的复位值也不同。

1) 工程应用

(1) 设计软件时，必须将整个软件系统第一条指令的存储地址指定为指令地址寄存器的复位值，只能使用汇编语言，例如：

```
ORG   指令地址寄存器的复位值
整个软件系统的第一条指令
```

ORG 是汇编器(将汇编语言变换为 CPU 的机器代码的工具程序)的命令，表示下一条指令存储在 ORG 指定的地址。

(2) 设计硬件时，将含有整个软件系统第一条指令的程序存储在一个非易失性的存储器中；地址译码电路的设计必须保证存储第一条指令的存储单元的地址为指令地址寄存器的复位值。这样，CPU 复位以后，就能够从该存储单元中开始读取指令执行。

存储单元的信息存储分为易失性和非易失性两种。易失性是指存储单元供电时所存储的信息在掉电后就会丢失。非易失性是指存储单元供电时所存储的信息在掉电后不会丢失，即不管存储单元是否供电，所存储的信息都不会丢失。计算机通电后，具有易失性的存储器中没有信息，而非易失性的存储器则始终保持着用户预先写入的信息(如预先写入的程序或数据)，因此，CPU 上电复位时，只能从非易失性的存储器中读取程序执行。

2) 举例说明

PC 的 CPU 总是从内存中将程序读入指令队列并执行。PC 的内存分为两部分，一部

分为内存条，存储容量很大，是易失性的；另一部分是主板上的 Flash 存储器，存储容量比较小，是非易失性的。

PC 复位以后，其指令地址寄存器的值为 FFFF0H，主板设计时，将含有整个软件系统的第一条指令的程序(BIOS 程序)预先存储在主板上的 Flash 存储器中，并且，硬件设计保证存储整个软件系统第一条指令的存储单元的地址为指令地址寄存器的复位值，因此，CPU 复位后执行的是主板上非易失性的 Flash 存储器中的 BIOS 程序。BIOS 程序的最后一个功能是将操作系统从硬盘上以数据流的形式复制到易失性的内存条中，然后 CPU 从内存条中再以指令流的形式读取操作系统并执行。

3) 重要结论

(1) 在一个计算机系统中，CPU 复位后要执行的第一个程序必须预先存储在一个非易失性的存储器中。

(2) 存储程序的存储器一般有以下两种组织方法：

① 存储程序的所有存储器都为非易失性的存储器，CPU 要执行的所有程序都预先存储在这些非易失性的存储器(如 Flash 存储器)。

② 存储程序的存储器分为非易失性和易失性两类，CPU 复位以后先从非易失性的存储器中读取预先存储好的程序执行，通过这些程序的执行，再将其他存储介质(如硬盘等)中的程序以数据流的形式拷入易失性的存储器(如内存条)中，然后再以指令流的形式读入 CPU 执行。

2. 指令控制方式

要理解程序执行的具体流程，必须理解 CPU 执行指令时具体是如何修改指令地址寄存器 IAR 的值的。一般来说，一个程序的多条指令是按顺序存储在存储器中的。某个程序在存储器中的存储见表 1.1 所列。

表 1.1　某个程序在存储器中的存储

指令的首地址	指　令	说　　明
1000H	程序的第 1 条指令	该程序的入口地址为 1000H，第 1 条指令共占 2 个存储单元(1002H～1000H)
1002H	程序的第 2 条指令	第 2 条指令占 3 个(1005H～1002H)存储单元
1005H	程序的第 3 条指令	第 3 条指令占 1 个(1006H～1005H)存储单元
1006H	程序的第 4 条指令	第 4 条指令占 1 个(1007H～1006H)存储单元
1007H	……	……

CPU 要执行一条指令，其具体过程分为三个阶段：

第一阶段，控制器从外部指令存储单元中读取指令，称为指令读取；

第二阶段，控制器分析指令所要执行的操作，称为指令译码；

第三阶段，控制器执行指令规定的操作，称为指令执行。

一般来说，只要 CPU 完成了第一阶段指令读取操作，CPU 的控制器就会将指令地址寄存器的值自动加上所读指令的长度(占多少个存储单元)，这时，指令地址寄存器就指向了相邻存储单元中指令，即下一条指令。

CPU 执行表 1.1 中的程序，指令地址寄存器 IAR 等于 1000H，这时，CPU 开始读取程序的第 1 条指令并执行，CPU 一旦将程序的第 1 条指令读入指令队列后，CPU 的控制

器就会将指令地址寄存器的值自动加上 2，即等于 1002H，此时，指令地址寄存器就指向了相邻存储单元中的指令，即第 2 条指令。

对于有些指令，在第三阶段指令执行时，CPU 还会根据指令规定的方法再次修改指令地址寄存器的值，这类指令统称为控制转移指令(或程序分支指令)。如果在指令执行阶段，CPU 不再修改指令地址寄存器的值，这类指令统称为顺序执行指令。因此，根据在指令执行阶段是否再次修改指令地址寄存器的值，可以将 CPU 的指令集分为控制转移指令和顺序执行指令两大类。

设表 1.1 中程序的指令都为顺序执行指令，则 CPU 在执行该程序时，总是从第 1 条指令开始按顺序逐条读取指令执行。

控制转移指令总是在指令执行阶段，根据指令规定的方法再次修改指令地址寄存器的值，使 CPU 转移到指令指定的地址去读取程序执行。

CPU 提供了多种控制转移指令，即具有多种再次修改指令地址寄存器的值的方式，以便于用户能够灵活地控制程序执行的流程，实现复杂的程序执行流程。

一般来说，CPU 控制转移指令的分类如图 1.1 所示。

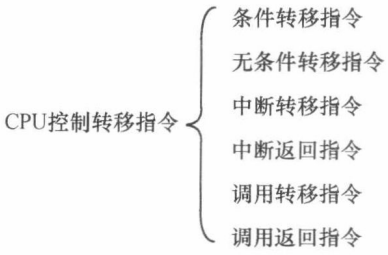


图 1.1　CPU 控制转移指令的分类

1) 条件转移指令

条件转移指令规定 CPU 执行如下操作：

(1) 在指令执行阶段，规定了修改指令地址寄存器的条件。如果满足规定的条件，则在指令执行阶段根据指令规定的方法修改指令地址寄存器的值，程序则发生条件转移；如果不满足规定的条件，则在指令执行阶段不修改指令地址寄存器的值，即程序将顺序执行。

(2) 在指令执行阶段，规定了修改指令地址寄存器的方法。

以上两条是学习及应用 CPU 条件转移指令的具体方法。

条件转移指令的工程应用：在软件设计上，使用条件转移指令实现程序可以根据不同的情况(条件)，执行不同的程序段，完成不同的功能。

需要说明的是，不同的 CPU，在指令执行阶段，当条件满足时，修改指令地址寄存器的方法不同。一般有以下两种方法：

(1) 在指令中存储着“转移目的地址”与“读取该条件转移指令后指令地址寄存器的值”之差，在指令执行阶段，通过将指令地址寄存器的值与这个差值相加得到转移的目的地址，然后再赋给指令地址寄存器，从而实现条件转移。某些 CPU 规定了这个差值的范围，在程序编译阶段，如果这个差值超过了该条件转移指令规定的范围，则编译器会报错。

(2) 在指令中直接存储着“转移目的地址”，在指令执行阶段，通过将“转移目的地址”直接赋给指令地址寄存器，实现条件转移。

表 1.2 说明了程序中条件转移指令执行的具体过程。由表 1.2 可知，当指令地址寄存器 IAR 等于 1000H 时，CPU 读取该条件转移指令以及执行的具体过程为(设条件满足时转移到 1008H)：

CPU 首先从地址为 1000H 的存储单元中将该条件转移指令读入指令队列，这时，CPU 的控制器自动将指令地址寄存器 IAR 的值修改为 1002H。在该条件转移指令的执行阶段，CPU 判断该指令所规定的条件是否满足，如果条件满足，CPU 则根据该指令规定的方法将指令地址寄存器的值再修改为 1008H，然后再从地址为 1008H 的存储单元中读取指令并执行；如果条件不满足，CPU 则不再修改指令地址寄存器 IAR 的值，CPU 将顺序读取指令并执行，即 CPU 将从地址为 1002H 的存储单元中读取指令并执行。

表 1.2　程序中条件转移指令的执行过程

指令的首地址	指 令	说　明
……	……	
1000H	条件转移指令	执行该条指令时，如果条件满足，设 CPU 将转移到地址为 1008H 的存储单元中读取指令并执行；如果条件不满足，CPU 将顺序读取指令并执行，即，从地址为 1002H 的存储单元中读取指令并执行
1002H	……	
……	……	
1008H	……	

2) 无条件转移指令

无条件转移指令在指令执行阶段，总是根据规定的方法修改指令地址寄存器的值。表 1.3 说明了程序中无条件转移指令执行的具体过程。

表 1.3　程序中无条件转移指令的执行过程

指令的首地址	指 令	说　明
……	……	
1000H	无条件转移指令	执行该条指令时，设 CPU 将转移到地址为 1008H 的存储单元中读取指令并执行
1002H	……	
……	……	
1008H	……	

由表 1.3 可知，当指令地址寄存器 IAR 等于 1000H 时，CPU 读取该无条件转移指令以及执行的具体过程：CPU 首先从地址为 1000H 的存储单元中将该无条件转移指令读入指令队列，这时，CPU 的控制器自动将指令地址寄存器 IAR 的值修改为 1002H。在该无条件转移指令的执行阶段，CPU 则根据该指令规定的方法将指令地址寄存器的值再修改为 1008H，然后再从地址为 1008H 的存储单元中读取指令并执行。

3) 中断转移指令与中断返回指令

在中断转移指令的执行阶段，CPU 执行如下操作。

(1) 根据某种规定的方法，CPU 将指令地址寄存器的值以及 CPU 其他重要寄存器的值保存起来。

(2) 根据某种规定的方法，CPU 获得一个地址，再将这个地址赋给指令地址寄存器 IAR。这个地址称为中断向量，中断向量所指向的程序称为中断服务程序，中断向量是中断服务程序第 1 条指令的地址，即中断服务程序的入口地址。

在第一步中，CPU 所保存的指令地址寄存器的值(中断调用指令下一条指令的地址)称为断点。CPU 保存断点的原因是：当 CPU 通过中断转移指令去执行中断服务程序结束后，还要返回到原来程序的断点位置继续执行原来的程序。

需要说明的是，不同的 CPU，保存 CPU 指令地址寄存器以及其他重要寄存器的值的方法不同，获得中断向量的方法也不同，在工程实际中，应参考 CPU 的用户手册。

在中断返回指令的执行阶段，CPU 执行如下操作：根据某种规定的方法，CPU 恢复中断转移操作所保存的寄存器的值——恢复指令地址寄存器的值，恢复 CPU 其他重要寄存器的值。

中断返回指令位于中断服务程序的出口，一个中断服务程序只有一个入口(这个入口就是中断向量)，但可以有多个出口，每个出口就是一条中断返回指令。中断服务程序可以根据不同的情况，从中断服务程序不同的出口出去，返回到原来被中断的程序。

中断返回指令位于中断服务程序中，中断转移指令位于被中断的程序中。表 1.4 说明了中断转移指令执行的具体过程。

表 1.4　中断转移指令的执行过程

指令的首地址	指 令	说　明	程 序
……	……		被中断的程序
1000H	中断转移指令	执行该条指令时，设 CPU 将转移到地址为 1008H 的存储单元中读取指令并执行	
1002H(断点)	……	中断服务程序执行结束后，CPU 将返回到这里读取指令继续执行原来的程序	
……	……		
1008H(中断向量)	……	中断服务程序的入口地址	中断服务程序
……	……	中断服务程序	
1100H	中断返回指令	中断服务程序的出口地址	
1102H	……		

由表 1.4 可知，当指令地址寄存器 IAR 等于 1000H 时，CPU 读取该中断转移指令以及执行的具体过程为：CPU 首先从地址为 1000H 的存储单元中将该中断转移指令读入指令队列，这时，CPU 的控制器自动将指令地址寄存器 IAR 的值修改为 1002H，这个值称为断点。在该中断转移指令的执行阶段，CPU 则根据该指令规定的方法将指令地址寄存器的值以及 CPU 其他重要的寄存器的值保存起来；CPU 再根据指令规定的方法获得中断向量 1008H，然后 CPU 再从地址为 1008H 的存储单元中开始读取中断服务程序并执行。

当指令地址寄存器 IAR 等于 1100H 时，CPU 读取中断返回指令以及执行的具体过程为：CPU 首先从地址为 1100H 的存储单元中将中断返回指令读入指令队列，这时，CPU 的控制器自动将指令地址寄存器 IAR 的值修改为 1102H。在指令执行阶段，CPU 根据某种规定的方法，恢复中断转移操作所保存的寄存器的值——恢复指令地址寄存器的值为断点 1002H，恢复 CPU 其他重要寄存器的值，从而 CPU 从中断服务程序中返回到程序原来被中断的位置继续读取并执行原来的程序。

通过中断转移指令所执行的中断服务程序，称为软件中断服务程序。

4) 调用转移指令与调用返回指令

在调用转移指令执行阶段，CPU 执行如下操作：

(1) 根据某种规定的方法，CPU 将指令地址寄存器的值保存起来；

(2) 目的地址存储在指令中，CPU 将这个地址直接赋给指令地址寄存器 IAR。

在第一步中，CPU 所保存的指令地址寄存器的值称为断点，断点就是调用转移指令下一条指令的地址。CPU 保存断点的原因是：当 CPU 通过调用转移指令去执行被调用的程序结束以后，还要返回到原来程序的断点位置继续执行原来的程序。

需要说明的是：不同的 CPU，保存指令地址寄存器的值的方法不同，在工程应用中，可参考 CPU 的用户手册。

调用返回指令规定 CPU 执行的操作：根据某种规定的方法，CPU 恢复调用转移操作所保存的指令地址寄存器的值。

调用转移指令位于调用程序中，调用返回指令位于被调用的程序中。调用返回指令位于被调用程序的出口，一个被调用程序只有一个入口，但可以有多个出口，每个出口就是一条调用返回指令。

在程序设计中，调用转移指令与调用返回指令是成对使用的，中断转移指令与中断返回指令是成对使用的。中断服务程序的出口只能使用中断返回指令，不能使用调用返回指令；被调用程序的出口只能使用调用返回指令，不能使用中断返回指令。

3. 硬件中断方式

CPU 的控制器有一个中断控制电路，与外部的中断控制器一起构成计算机系统的中断控制系统。中断控制系统的组成如图 1.2 所示。

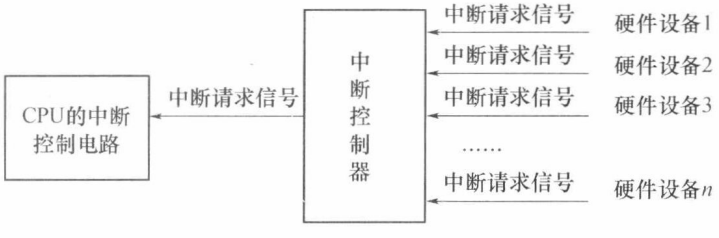


图 1.2　中断控制系统的组成示意图

中断控制器同时监视多个硬件设备的中断请求信号。当有多个硬件设备同时发出中断请求信号时，中断控制器按照预先规定的优先顺序，只输出优先级别最高的硬件设备的中断请求信号到 CPU 的中断控制电路。当 CPU 的中断控制电路监测到这个硬件设备的中断请求信号并且响应该中断请求时，则 CPU 在执行完当前指令之后，立即执行与中断转移指令相同的操作。

(1) 根据某种规定的方法，CPU 将指令地址寄存器的值(CPU 执行完当前指令之后，IAR 的值)以及 CPU 其他重要寄存器的值保存起来；

(2) 根据某种规定的方法，CPU 获得一个地址，再将这个地址赋给指令地址寄存器 IAR。这个地址也称为中断向量，中断向量所指向的程序也称为中断服务程序。然后，CPU 执行专门服务于该硬件设备的程序，这种程序称为硬件中断服务程序。

触发 CPU 产生中断操作的硬件设备，称为中断源。硬件中断服务程序是由相关硬件设备触发的，触发时刻 CPU 正在执行的程序被 CPU 中断执行(暂停执行)，然后 CPU 以某种规定的方法获得中断向量，CPU 再将该中断向量赋给指令地址寄存器，最后 CPU 开始读取相应的中断服务程序并执行。

CPU 执行完硬件中断服务程序以后，还要返回到原来被中断的程序中继续执行。具体方法是：在硬件中断服务程序的出口处，使用一条中断返回指令。

在发生硬件中断转移之前，指令地址寄存器的值就是断点，这个值是当前指令执行完以后所确定的指令地址寄存器的值。如果当前指令是顺序指令，则断点就是该指令的下一条指令；如果当前指令是控制转移指令，则断点就该控制转移指令所确定的目的地址。另外，在执行被中断的程序的某一条指令时，硬件设备产生中断请求是不确定的，因此，断点不是固定的。

硬件中断服务程序的执行与软件中断服务程序、被调用程序的执行区别：硬件中断服务程序的执行是非常特殊的，它的执行是通过硬件设备触发的，而不是通过指令触发的。因此，硬件设备产生中断请求时，程序正在执行哪条指令是不确定的，所以，断点不是固定的。硬件中断服务程序何时执行是不确定的，这取决于外部的硬件设备，这种程序代码称为异步代码(Asynchronous Code)。需要说明的是，硬件中断服务程序中所有调用的程序都属于异步代码。

对于软件中断服务程序以及被调用程序，由于其执行是由中断转移指令或调用指令触发的，因此，被中断的程序或调用程序的断点是确定的(中断转移指令或调用指令的下一条指令)，所以，整个程序流程是确定的。当程序代码的执行流程是确定的时，这种程序代码称为同步代码(Synchronous Code)。

硬件中断服务程序执行的具体情况如图 1.3 所示。

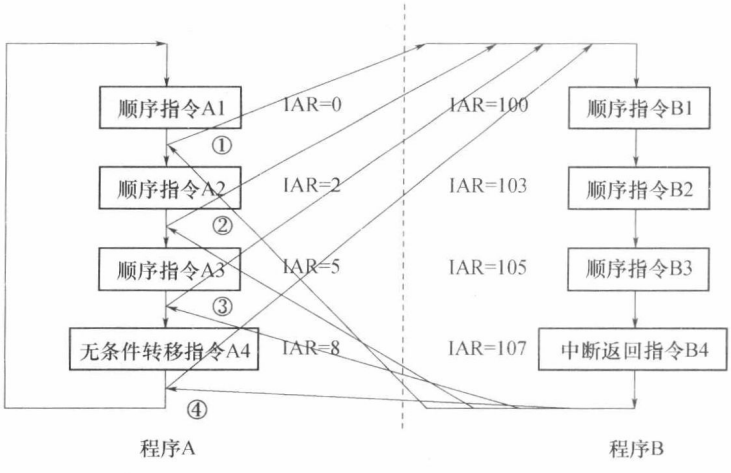


图 1.3　硬件中断转移与中断返回

设指令地址寄存器 IAR 的复位值为 0，因此，CPU 复位以后，将以死循环的方式执行程序 A。设程序 B 为某个硬件设备的中断服务程序，当该硬件设备没有发生中断请求时，CPU 将以死循环的方式始终执行程序 A，程序 B 将始终不会被执行。当该硬件设备向 CPU 发送中断请求并且 CPU 响应该设备的中断请求时，CPU 将自动执行某种规定的操作：

(1) 根据某种规定的方法，CPU 将指令地址寄存器的值(断点)以及 CPU 其他重要寄存器的值保存起来。在图 1.3 中，断点的值有 4 种可能：0、2、5、8。

(2) 根据某种规定的方法，CPU 获得该硬件设备的中断向量，再将这个中断向量赋给指令地址寄存器 IAR。在图 1.3 中，这个中断向量为 100，这时，CPU 将从地址为 100 的存储单元中开始读取指令并执行。

当执行到中断返回指令时，CPU 将返回到被中断的程序的断点位置，继续读取原来的指令并执行。

以下是 5 种可能的程序执行流程：

(1) 当硬件设备不发生中断请求时，程序流程为程序 A 所规定的程序流程，因此，程序 A 中的代码是同步代码。

(2) 当 CPU 在执行程序 A 中的顺序指令 A1 时，CPU 响应硬件设备产生的中断请求：这时，断点为 2，指令地址寄存器 IAR 被 CPU 的硬件中断操作修改为 100，CPU 执行中断服务程序 B，通过中断返回指令，CPU 再返回程序 A，执行顺序指令 A2。这时，程序流程如图 1.3 中的①所示。

(3) 当 CPU 在执行程序 A 中的顺序指令 A2 时，CPU 响应硬件设备产生的中断请求：这时，断点为 5，指令地址寄存器 IAR 被 CPU 的硬件中断操作修改为 100，CPU 执行中断服务程序 B，通过中断返回指令，CPU 再返回程序 A，执行顺序指令 A3。这时，程序流程如图 1.3 中的②所示。

(4) 当 CPU 在执行程序 A 中的顺序指令 A3 时，CPU 响应硬件设备产生的中断请求：这时，断点为 8，指令地址寄存器 IAR 被 CPU 的硬件中断操作修改为 100，CPU 执行中断服务程序 B，通过中断返回指令，CPU 再返回程序 A，执行无条件转移指令 A4。这时，程序流程如图 1.3 中的③所示。

(5) 当 CPU 在执行程序 A 中的无条件转移指令 A4 时，CPU 响应硬件设备产生的中断请求：这时，断点为 0(由转移指令确定的目的地址)，指令地址寄存器 IAR 被 CPU 的硬件中断操作修改为 100，CPU 执行中断服务程序 B，通过中断返回指令，CPU 再返回程序 A，执行顺序指令 A1。这时，程序流程如图 1.3 中的④所示。

1.1.1.2 控制指令的译码与执行

CPU 的指令集是 CPU 的设计者将 CPU 的绝大部分功能按照某种格式进行编码形成的，每一条指令规定了 CPU 要执行的硬件操作。CPU 的控制器有一个专门的电路来分析指令编码，从而知道所要求执行的硬件操作，这个过程称为指令译码，这个专门电路称为指令译码电路。

CPU 处理指令的操作包含了读取指令、对指令译码和执行指令三部分。CPU 处理指令的速度不仅仅与 CPU 的工作频率有关，也与这三种操作的流水线的级数有关。