微型計算机实用教材

MCS-48
单片微型計算机

北京工业大学

微型计算机研究开发应用中心

再 版 前 言

电子计算机的发展在近三十年的时间内，经历了从电子管到大规模集成电路这样四代产品。在前二十年中，电子计算机向着规模大、速度高的方向不断前进。而在后十年中，微型计算机开始迅速发展和普及。单晶片微型计算机的产生进一步推动了这一发展趋势，使计算机的应用达到前所未有的普遍程度。产生了由微电子技术为先导，带动了包括新材料技术，生物工程，……等多种学科的新技术革命。在整个计算机产品应用“树”中，其庞大的根系尖端，就是单晶片微型计算机化的产品。而目前我国在这方面所做的工作是远远不够的。为了推动这项工作的发展，我们研制开发了TP801MCS 开发系统，为MCS系列单片机的应用提供了一种廉价的，然而基本功能齐全的单片机化产品的开发手段。

本书是为设计研制单片机化产品的工程技术人员编写的。因考虑到技术人员水平不一，故在第一章的第二节中，用了相当大的篇幅，由浅入深地讲述了计算机的基本原理及有关的基础知识。并逐步引出有关计算机软、硬件方面的各种常用术语和概念，作为后续章节的基础部分。

由于 TP801MCS 单片机开发系统是专门用于 Intel8048／8049（包括 8035、8039、8748、8749）单片机的产品开发而设计的，因此，根据这几种型号产品的技术手册、有关资料及笔者应用设计的经验，在本书的第二章至第五章中结合 MCS 单片机的结构原理、指令系统和硬件及软件的应用设计，对 MCS 上述的六种产品作了较详细的介绍。

为了便于读者编写应用程序，我们还整理和编写了 MCS-48 单片机编程设计袖珍手册。作为本书单开本附录资料。

本书的附录中，给出了有关单片机产品的英文简介。使用英文原文形式可确保其技术指标含意的准确性，供读者使用时参考。

原版由北京工业大学自动化系研究室侯伯文编写。

为了满足广大用户及读者的要求，本书进行了再版。

本书新版由北京工业大学自动化系徐宁寿、鹿树理对原书第一版作了较多的修改和补充。

第一、二、三章和袖珍手册由徐宁寿执笔，第四章、第五章由鹿树理执笔。梁瑞恒、朱世宁校阅。程玮组织、编辑出版。

限于笔者水平，书中难免仍有错误和不足之处，恳请广大读者指正。

北京工业大学
微型计算机研究开发应用中心
1987年5月

目　录

第一章　单片计算机基础

1.1 概　述

一个典型的数字计算机系统，包括以下几个基本部分：

·中央处理器（Central Processing Unit），简称CPU（包括了算术／逻辑运算单元和控制器）

·程序存储器

·数据存储器

·输入/输出接口（Input/Output Port），简称I/O接口

就TP801单板微型计算机而言，其结构包括如下几个部分：

·Z80-CPU

·2 KB只读存储器（Read-Only Memory），简称ROM（存有监控程序）

·4 KB随机存取（读/写）存储器（Random Access Memory），简称RAM

·数个I/O接口（包括了并行接口Z80-PIO，定时/计数器Z80-CTC，以及键盘输入接口，发光二极管显示器输出接口，……）。

TP801微型计算机是由多片大规模集成电路IC芯片组成的多片系统，因这些芯片都布置在一块印刷电路板上，故称之为单印刷电路板微型计算机，简称单板机。

随着大规模集成电路生产技术的发展及应用的需要，生产了这样一种集成电路元件，即在一片晶体芯片上（自然是一个IC封装之中），就可包括上述一个数字计算机的四个基本组成：CPU，ROM（或EPROM），RAM和I/O接口。这种元件被称为单芯片微型电脑或单片微型计算机（Single Chip Microcomputer）简称为单片机。

例如，Intel公司研制出产的8048H，单片微型计算机，在一个40脚双列直插式封装（DIP）集成电路元件中就包括了：

·8位CPU

·1 K×8 ROM——程序存储器

·64×8 RAM——数据存储器

·27根I/O接口引线

·一个8位的定时/计数器

因单片机的资源（存储器容量，I/O接口引线），可满足很多场合的应用，加之在体积，功耗，价格和操作性能等方面的特点，所以单片机的应用日益广泛，品种也日趋增多。这些产品按照其基本操作处理的数据位数分类，可分为1位（位片机）、4位、8位和16位单片机。就8位单片机而言，自Intel公司于1976年推出MCS-48系列产品后，已涌现出不少型号的8位单片机，其中产量较大的有以下品种：

Intel公司的MCS-48和MCS-51系列(8035,8039,8048,8049,8748,8749和8020,8021,8022, 8043, 8031, 8051, 8751等)；

Motorola公司的M6801和M6805系列 (6801, 68701, 6803,6805等)；

General Instrument公司的PIC1650系列 (PIC1650A, 1654, 1655A, 1656, 1670等)；

Mostek公司的3870系列 (3870, 38E70, 3871, 3873, 3875等)；

Rockwell公司的6500/1系列 (6500/1, 6500/11, 6500/12, 6500/13, 6500/14, 6500/41, 6500/42, 6500/43)；

NEC公司的μCOM-87系列 (μPD7800, 7801, 7802, 78C06, 7811等)；

Zilog公司的Z8系列 (Z8601, Z8603, Z8611, Z8612, Z8613。Z8671, Z8681/82)；

Texas Instrument公司的7000系列；

National公司的8073。

尽管单片机品种日益增多，但是Intel公司的MCS-48系列还是单片机的主导产品，该系列的产品被誉为单片机的行业标准。目前，该系列产品的供应厂家以NEC和Nationla两家最为积极。其它的厂家按目前产量排列：富士通，东芝,Signetic/Philips,AMD,西门子，OKI，SSS，Huges，三菱，Harris……等。有的次供应厂家还发展了一些超出原厂家的新产品，例如带有LED驱动的MCS-48产品，或者使用“猪背” (Piggyback)式结构的IC封装，把ROM或EPROM集成电路重叠在单片机封装之上。

MCS-48系列产品在设计上的最大特点是适合于作为控制处理器使用。例如通过指令使I/O的某一根线复位为0或置位为1，对累加器的某一位测试，为数众多的条件分枝指令，……等，加上片内定时/计数器和某些产品带有A/D转换器，使MCS-48系列提供了有效的控制功能。难怪Intel公司自称MCS-48系列为“微型控制器” (Microcontoller)。

近年来，CMOS技术大量用于单片机产品之中，使8位单片机的标准功耗降到25 mW，待机方式为5μW。加上CMOS的温度稳定性和抗干扰能力，使单片机应用范围更加扩大，特别是研制了各种智能化的、便携式的产品，扩大了单片机的应用领域。

对于应用单片机产品的设计者来说，第一个面临的问题是开发工具问题。通常的单片机开发系统的价格都在数千元美元以上。然而，没有开发系统，要进行单片机的应用设计是十分困难的。为了适合我国的国情，北京工业大学微型计算机研制开发中心的二系研究室，设计了TP801 (或TP805) MCS单片计算机开发系统。该系统的基本组成是：

- TP801单板微型计算机 (或TP805灵巧计算机)
- TP801I接口板或TPZCPU驱动板 (或TP805MCS I/M接口板)
- TP80MCS单片机开发模板
- TP MCSU用户样机板

这样一个开发系统所具有的功能是：

- 对使用8035/8048/8748/8039/8049/8749等型号MCS-48系列单片微型计算机产品进行研制开发。

· 用户可以十分方便地在TP MCSU“用户样机板”上组成样机电路板。

· 可以通过单片机仿真电路及仿真插头实现用户样机单片机的实时仿真调试。

· 使用硬件比较器电路实现实时仿真运行断点。

· 使用静态RAM进行单片机程序存储器仿真，便于进行程序的诊断和调试。

· 可以对单片机片内全部RAM存储单元，以及PC，Acc，PSW，SP和寄存器进行检查和更换。

· 可以单步跟踪用户程序的流程，了解每条指令的执行效果。

· 仿真/调试诊断中，完全不占用单片机的用户可用资源：程序和数据存储器的任何单元，全部I/O引线等。

· 对8748/8749片内EPROM的编程写入/校验。

· 对8048/8049/8748/8749片内ROM存储器程序的读出校验。

TP801（或TP805）MCS单片机开发系统所能开发的单片机型号是：

8035　外部程序存储器版本

8048　内部掩膜ROM程序存储器版本（1K）

8748　内部EPROM程序存储器版本（1K）

8039　同8035，但内部RAM加倍

8049　同8048，但内部RAM及ROM加倍

8749　同8748，但内部RAM及ROM加倍

以上六个产品全部是40脚DIP封装，管脚信号完全相同。本教材的全部内容，是为了使用户通过使用TP801（或TP805）MCS开发系统，开发应用上述六种型号产品而编写的。因此，没有特殊说明时，凡用到“MCS单片计算机”或“单片机”时，均指上述六种产品，而“MCS-48”则指前三种，“MCS-49”则指后三种。实际上，Intel公司的“MCS-48系列”除包括上述六种外，还包括了8021等几种单片机产品，这些产品所用指令系统基本相同，只有少数例外。

1.2　微型计算机基础

单片机的出现，推动了仪器仪表智能化的进程，并广泛应用于生产和社会的各个领域。因此，各种产品的设计者都开始研究如何将单片机应用于自己所在领域的产品设计中。考虑到部份设计人员没有接触过计算机的原理和设计知识，本节将就计算机的基础知识作一些概述，以便为后续章节提供背景知识（已经熟悉本节内容的读者可直接续读下一节内容）。本节介绍的内容包括：

典型的计算机结构

CPU的结构和功能

计算机的操作和定时

计算机程序设计

1.2.1 典型的计算机结构

数字计算机，是对数据进行处理（计算）的设置。它包括数据的传送和处理这样两大部份。

数据的来源有两个：其一，来自输入/输出（I/O）设备，例如键盘，光电输入机，模拟量/数字量（A/D）转换器，……一些处理结果也要输出到外部设备。例如发光二极管（LED）显示器，CRT显示屏幕，打印机，绘图机，数字量/模拟量（D/A）转换器，……。外部设备通过计算机的I/O接口电路来传送数据。其二，数据来自称为存储器的设备。存储器中存有被计算机所处理的数据（程序，常数和运算结果等）。程序是由指令组成，即为了指定计算机实施某种操作而由一组指令按照正确逻辑的组合。计算机每读一条指令，就执行一种基本操作。

有了数据（包括指令），还必须有其处理逻辑（电路）。如果所处理的是指令，还必须有一个根据指令实施控制操作的逻辑。计算机的处理（控制）逻辑就在中央处理器之中。不论大、中、小、微型计算机都包括图1.1所示基本结构：

- 存储器
- 输入/输出（I/O）接口
- 中央处理器（CPU）

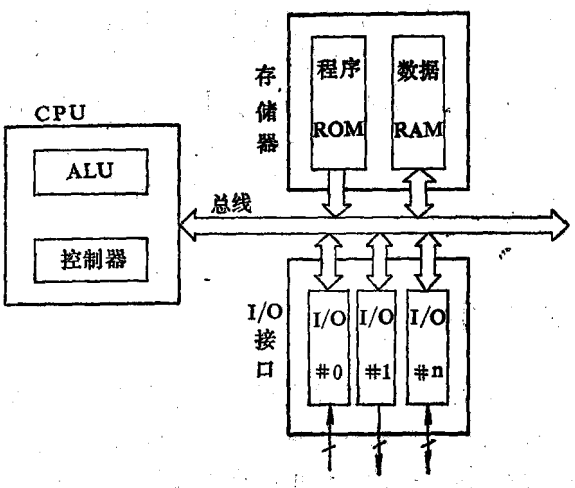


图 1.1 计算机结构

既然计算机是数据处理设备，那么电子计算机中的数据是一个什么形式呢？

电子计算机的基本电路结构是逻辑电路，逻辑电路的输入/输出是逻辑1或0，逻辑1即高电平，逻辑0即低电平。以TTL电路而言，大于2.4伏电压为逻辑1，低于0.8伏电压为逻辑0，而0.8～2.4伏间的电压是无效的，可能产生误码。当某逻辑电路输入（输出）高电平时，说“输入（输出）1”。当某导线上出现高（低）电平时，说“正传送1（0）”。既然计算机的结构是逻辑电路，计算机处理的数据也就只能用1/0这样两个数字符号来计数，而不能像十进制那样用0～9这样十个符号来记数。因此当数值

为多位数时，用这样两个符号来计数就只能是逢二进一。如果是一个2位的二进制数，可以表示00，01，10，11这样四个相当于十进制0，1，2，3的数值。显然，与十进制一样，二进制也有“位、进位、低位、高位”等概念，仅进制不同。通常每一位记作“b”（bit），最低有效位记作LSB，最高有效位记作MSB，进位记作C（或CY）。

实际上，一个n位二进制数代表了2^n个数。因此一个8位二进制数为0～255（$2^8=256$）。MCS—48系列单片机是8位计算机，从数据的传送来看，一个8位数据是通过8根导线（管脚）同时（并行）传送的。这样一个8位数据称为一个字节（Byte），由b0～b7共8位组成，其中低四位又叫低半字节（Low Nib）高四位又叫高半字节（High Nib），如图1.2所示。

一个8位计算机，在数据传送、数据寄存、处理时都以8位，即字节为单元。

数据如此，地址也是一样的。CPU从存储器中读取一个字节时，必须要先向存储器输入所要读取的数据存储单元的地址，这个地址往往是大于8位，但也是二进制的。如果通过10根导线／管脚输入地址数，那么此存储器的存储单元有2^{10}，即1024个地址单元。每个地址单元存储着一个字节。此存储器称为1 KB（一千字节）存储量，计算机中的“1 K”实际是1024的简称。于是，11位地址的8位数据存储器为2 KB存储量，而12位地址则为4 KB。

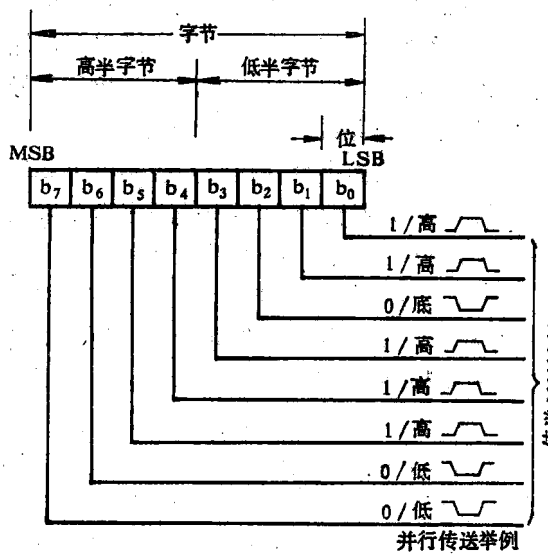


图 1.2 数据及传送示意

欲从存储器中读出或写入（存入）一个数据，必须首先向存储器输入要访问（读/写）单元的二进制地址，并发出控制信号脉冲（存储器读/写）。这都需要由存储器的控制逻辑电路进行控制，把数据线上的二进制数据送入指定地址单元中存储（写时），或者把指定地址单元中所存的数据输送到数据线上（读时）。在计算机中，存储器的读/写操作是在CPU控制下进行的。因此，CPU和存储器之间，必定有传送二进制地址和数据的通道，以及传送控制信号的控制线。

CPU不仅从存储器，还要从I/O接口交换数据。由于计算机的I/O接口不止一个，因此每个接口都有该接口的二进制“口地址”。所以，CPU和每个接口之间，也必定有传送地址和数据通道，以及传送控制信号的控制线。由于CPU只有一个，不会同时访问存储器和接口。因此，CPU与存储器及接口间地址和数据传送通道可以而且实际上也正是共用的。CPU在执行某指令时访问存储器，而在执行另一条指令时才访问某接口，因此是分时使用传送通道的。所以，CPU、存储器和接口间的通道称为总线（BUS）。当地址与数据通道分开时，分别称为地址总线和数据总线。由于地址总是由CPU输出的，所以地址总线是单向的，这种单向性显示了CPU的控制中心性。数据总线却是双向的，即

某一刻是输入到CPU，而另一刻可以是由CPU输出。

存储器中存储的数据虽然都是二进制代码，但是所代表的信息含意却是不同的，有一些是计算机所要处理的数据，而另一些是代表了某种要求CPU执行的操作——操作码。

例如，同是00010111。对MCS—48而言，可能代表了一个数：（十进制）23；也可能是一条指令："将CPU内一个叫做A 的寄存器中所存的数加 1 "。那么，存储器中所存数据字节是一个数还是一个操作命令代码呢？这要看CPU读到这个字节后如何处理。

实际上，任何CPU中都包括了两个最基本的数据处理逻辑单元：

·算术/逻辑运算单元（Arithmetical and Logical Unit），简称ALU

·指令译码器

凡是输入CPU后，被CPU控制送入指令寄存器，并通过指令译码器进行处理的数据，都是指令的操作码字节。指令译码器的输出送入"定时/控制器"，简称控制器。控制器根据指令译码的结果发出各种控制信号脉冲，这些脉冲或送到CPU以外，或CPU以内，控制整个系统的运行。因此，CPU是中央处理单元，也是"中央控制"单元。由此可见CPU是根据指令（程序）控制整个系统工作的。

虽然CPU是按照存储器中的程序指令来工作的，但是计算机的设计者又考虑了一种打断计算机(正运行中的)程序的办法，即中断。

中断是CPU与接口间一种十分重要的联络方式。通过接口传送数据有两个特点，一个是随机性：某些外设（例如操作者使用键盘输入，或者外设的超温超压报警信号……等）数据输入是随机发生的。对此，CPU不能一直监视接口而不去处理其它数据。另一个是速度上的差异：CPU对接口访问（读/写）只要几微秒就可以完成一次，而外设的速度则相对太慢，例如打印机打印速度每秒钟160字符，即每秒钟接收160次接口输出字符代码，平均为6千多微秒。当外设要求和CPU交换数据时，就向CPU发出一个信号脉冲——中断申请。CPU在收到此申请后，可以停止正在执行的程序（主程序）而转去执行一个特定的程序——该接口的服务程序（中断服务程序）。在此之前，CPU可自动将被打断的程序地址保存在一个称为堆栈的存储单元中。这一过程就叫做CPU的中断响应。在执行完中断服务程序后，当CPU执行了该程序的最后一条指令——中断返回指令后，就从中断返回，即从堆栈中取回原程序被打断处的地址，然后继续执行被打断的主程序。

1.2.2 CPU的结构和功能

一个典型的CPU包括了以下单元（图1.3）

·寄存器（Registers）

·算术/逻辑运算单元（ALU）

·控制器

CPU内有很多存储数据的暂存存储单元，这些单元都是可以读，可以改写的存储单元，叫做寄存器。CPU内部寄存器除了少数有初值外（例如下文讲到的PC寄存器在上电时为零），大都在上电时为随机数。因此，寄存器中的数据都是在程序执行的过程中送入的，而且数据处理的最后结果，不是送到数据存储器中，就是通过接口输出。可

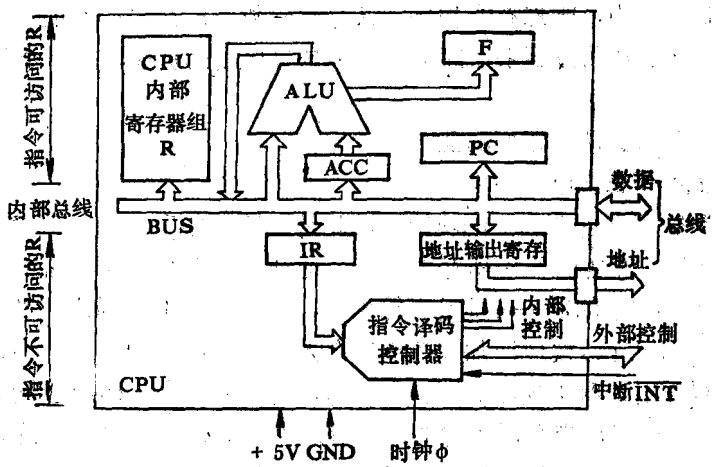


图 1.3 CPU内部结构示意

R寄存器 Acc累加器 F标志寄存器 PC程序寄存器

ALU算术/逻辑运算单元 IR指令寄存器

见，寄存器中所存的只是暂存的数据，这也是“寄存”的含意。

CPU寄存器又分为两大类：通用寄存器和专用寄存器。下文介绍的累加器就属于通用寄存器，而程序计数器和指令寄存器则属专用寄存器。

累加器（Accumulator）

累加器简称Acc（或A寄存器），是CPU中使用最频繁的一个寄存器。也是ALU单元不可缺少的一个输入寄存器，而ALU在CPU中又是必不可少的数据处理单元。Acc中的数据，不仅送入ALU单元中进行算术或逻辑运算，而且经ALU运算后的结果还要送回Acc中，再通过使用其它指令才将Acc中数据送到其它单元（其它寄存器，某个I/O接口，某个存储单元）中去。

程序计数器（Program Counter）

程序计数器，简称PC，是各种CPU都不可少的另一个寄存器。它是专门用于存储所要读取指令的地址的专用寄存器。

计算机中的程序存储器，一般采用只读存储器ROM。它的特点是：

- 断电后，所存数据不会丢失，因此上电时（一通电）所存数据即可读取使用。
- 所存数据在计算机运行中，只能读取而不能改写，因此称为“只读”。

显然，这种存储器适合于存储计算机上电就必须执行的程序，也适合于存储计算机所要使用的一些常数。

至于计算机运行中的变量c就要存入另一种存储器——随机存取存储器RAM中了。这种存储器一经断电，所存数据就丢失了。因此在上电时，各单元内容为随机数。在程序运行中，计算机将经过计算后准备输出，或者有待进一步处理的数据，或者接口输入准备处理的数据，暂时储存在RAM之中。

程序存储器ROM中，既然存有程序，又存有常数，那么怎么保证CPU读的是指令而不会是常数呢。这是从两方面来保证的，一个是程序员的责任，另一个是CPU的PC功能来保证的。

CPU的PC寄存器有两个重要功能：

·程序计数器PC在上电时清零。

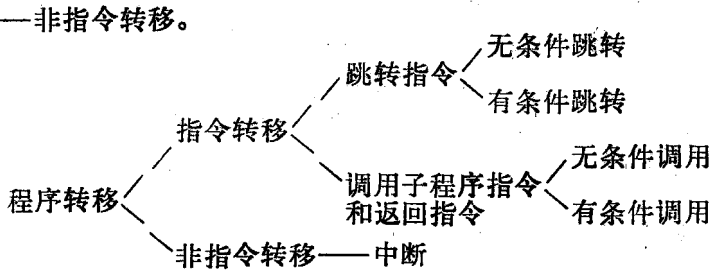
·程序计数器PC有自动加1的功能，因此是“计数器”。

CPU读取指令，就是把PC中所存的数——地址，输送给程序存储器，以便从该地址单元中读取指令。PC中的数——地址在输出后，就自动加1。因此，当程序员把程序的各条指令，从0地址开始，按地址加1的次序存入程序存储器中。那么，计算机一上电，把PC中的地址0输出，取来程序的第一个字节，即进入程序。而当CPU经过指令译码和控制器控制系统完成了该指令所规定的操作后，PC已经自动加过一。这时CPU再把PC输出，取来的便是下一个地址处的指令字节。……如此循环下去，CPU读一条指令，执行一条指令。只要程序员编写和存入程序时，按照地址加一的顺序存入存储器，就能保证读入指令寄存器IR的都是指令了。

但是，PC值也可能在输出前被更换掉。这样就打破了按“地址加一”执行程序的次序。换入的新地址，可能属于存储器的另一个区域。当CPU把PC新值输出时，就转入新地址区读取指令了，这时我们说程序执行发生了转移。读者以后将会发现，这种转移可以大大提高程序存储器的占用效率，并且也正是计算机智能的体现。PC值发生“跳变”而实现程序转移的情况，发生在下面两种情况下：

·执行了一条能改变PC的指令——指令转移。

·中断响应——非指令转移。



从分枝图中可知，能改变PC值的指令有两类，一类为跳转（JMP）指令，另一类为调用（CALL）子程序指令和调用返回（RET）指令。这两类指令所实现的转移是不同的，但指令中都包括了所要转移的目的地址——“新”程序块的入口地址。

跳转（JMP）与标志（Flags）寄存器

跳转指令是各种计算机都不可少的指令，一条跳转指令通常是2～3个字节，第一个字节代表了跳转操作命令；“按后面下随的1（或2）个字节所提供的转移目标地址实现转移”。因此，当CPU把第一个字节读入IR并译码后，就按此字节所指定的操作，将指令中提供的地址数据送入PC中。从而当CPU再输出PC去读取指令时，就不是跳转指令下面PC＋1地址处的指令，而是跳转指令所指定的地址处的指令了。这就是程序转移。图1.4（a）是跳转指令引起程序转移的示意图。

跳转指令有两类，一类叫无条件转移，一类叫有条件转移。无条件转移指令执行后，一定会实现程序转移，而有条件则不然。只有当条件“是（符合）”时转移，而条件“否(不符合)”时，指令中所提供的地址并不送入PC。这样PC依然是自动加一后的PC，程序继续“直行”而不发生跳转转移。由于条件转移指令执行后可产生两种结果：直行或跳转，因此在程序的流程中产生了分枝。故条件转移指令又可称为分枝转移指令。条件转移指令操作见图1.4（b）所示。

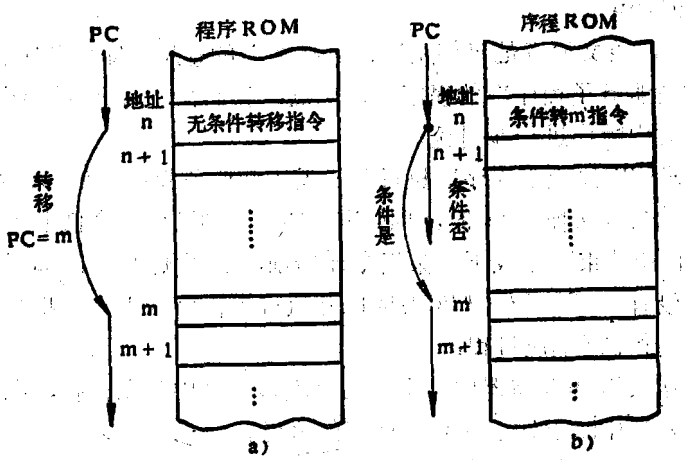


图 1.4 分枝转移指令的执行

条件转移指令根据什么来判定转移/直行呢？就象交叉路口的“路标”一样，是通过被称为“标志”的逻辑测试来判定的。在CPU中，有很多标志触发器。这些触发器的输出1/0，代表了CPU的某种状态。通常，有两类标志：

- ALU的运算结果标志和内部状态标志；
- 外部输入信号标志（有的CPU没有）。

从图1.3可知，ALU运算处理后的数据结果，是通过内部总线送回Acc中。而ALU处理后，反映数据结果的状况，送入F寄存器——即标志（Flags）寄存器中。F寄存器的每一位都有一个标志名称，代表了数据的某个状况。例如标志Z，叫累加器零标志，当处理后送入Acc的数据为全零时，Z = 1，反之Z = 0。又如，标志C，叫进位标志。当处理的数据最高位有进位溢出时，C = 1，反之C = 0。各种CPU标志位的多少是有差异的。

当程序中某条指令为加法指令，后随的指令为“根据C标志跳转指令”，那么当程序执行到后一条指令时，如果前条加法后有进位，则按后一条转移指令中的指定地址跳转，反之就不能转移。由于通过测试标志可以改变程序的进程——状态，有时将这些标志集中起来寄存于一个称做“程序状态字”（Program State Word，简称PSW）的寄存器中。

由于F寄存器ALU的输出之一，因此，只有通过ALU进行处理的指令才影响标志的变化。CPU指令中，有很多单纯传送数据的指令，这种指令一般不会影响标志。这样，每位标志位将一直保持到下次变更时为止。

在某些CPU中，条件分枝指令不仅通过对标志测试来实现判定，而且还有一些跳转指令是通过“外部输入信号”来判定分枝与否的。例如MCS-48系列单片机有两个信号线T0和T1（管脚），相应地有两条跳转指令JT0和JT1。当执行JT0指令时，CPU控制逻辑对T0信号输入进行测试（Test），如果T0为1，则跳转，否则不跳转。T0、T1和$\overline{INT}$（中断申请输入）线都是CPU的输入信号线，但是互相有所区别。下面让我们看看，中断是如何改变程序执行流程的。

中断（Interruption）

中断，简写为INT，英文原意是“停止”、“打断”的意思。即中断主程序的执行而转入一个称为“中断服务子程序”的程序。这一点，我们在1.2.1中已经简介过。下面我们从程序转移角度对中断作进一步讨论。

当某外部设备要求CPU停止主程序的执行而转入中断服务程序（例如，在服务程序中进行一次与某接口的数据交换I/O）时，向CPU的$\overline{INT}$管脚发出一个低电平信号——申请中断。CPU在每条指令结束时，或下一条指令开始前，总要测试一下$\overline{INT}$信号，看是否有中断申请。如果没有，CPU将输出PC去读取并执行主程序的下一条指令。如果有中断申请，CPU就开始一个中断响应过程：

• 把PC中的地址送入RAM存储器中一个称为“堆栈”区中加以保护，以使在中断服务程序结束后返回主程序中断点处，继续主程序的执行；

• 把中断服务程序的首地址——中断矢量，送入PC之中。

此后，CPU就按PC中的中断矢量地址进入中断服务程序。

当中断服务程序结束时，最后一条指令是中断返回指令。执行这条指令时，CPU的操作是：把中断点处主程序的地址，从堆栈中取出并送入PC，这样CPU就返回到主程序中断处，继续执行主程序。中断时，程序的转移过程如图1.5所示。

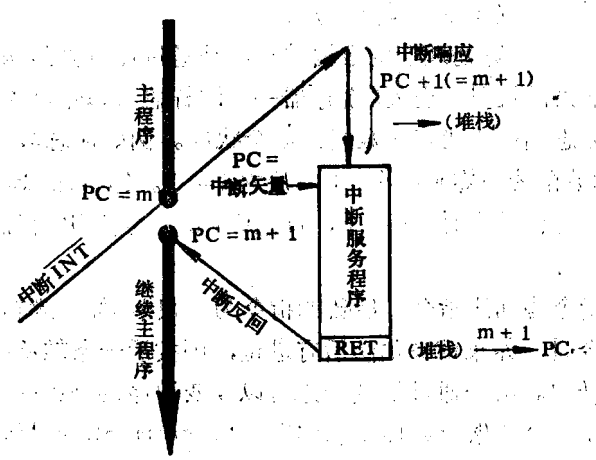


图 1.5 中断时程序的转移

表1.1为外部中断转移和输入测试信号指令转移的比较。

表中所指“中断矢量”提供的来源是随不同CPU而异的。一种是中断响应过程中由

表 1.1

转移	测试条件	转移地址	发生时机	有无程序返回
测试信号指令转移	CPU外输入	指令之中	执行指令时	无
中断转移	CPU外输入	中断矢量	随机申请	中断服务后返回主程序

申请中断的外设提供，通过数据总线输入CPU的PC中。另一种是固定的中断矢量地址，即中断时，CPU向PC中送入一个固定不变的地址常数，这个地址就是中断服务程序的首地址。在这种情况下，用户设计中断服务程序时，该程序的第一条指令一定要存于中断矢量所指的存储单元处。

子程序（Subroutine）调用

除了跳转指令外，能改变PC的另一类指令为子程序调用(CALL)指令。与跳转指令相似之处是，这类指令的字节中，也包括了所要转去执行的程序块首地址。这个地址在调用指令执行中，也会送到PC中去，从而发生程序的转移。这个所要转去执行的程序块称为子程序。子程序的出口为一条子程序返回（RET）指令，执行此指令时，再次发生程序转移——返回到CALL指令下面一条指令。这个过程与中断的返回十分相似。为了能从子程序返回主程序，在执行CALL指令时，要将返回地址由CPU自动送入堆栈中保存起来。此返回地址从PC送入堆栈的过程称为入栈，从堆栈中取回并送入PC的过程称为出栈，子程序的调用及返回过程见图1.6。

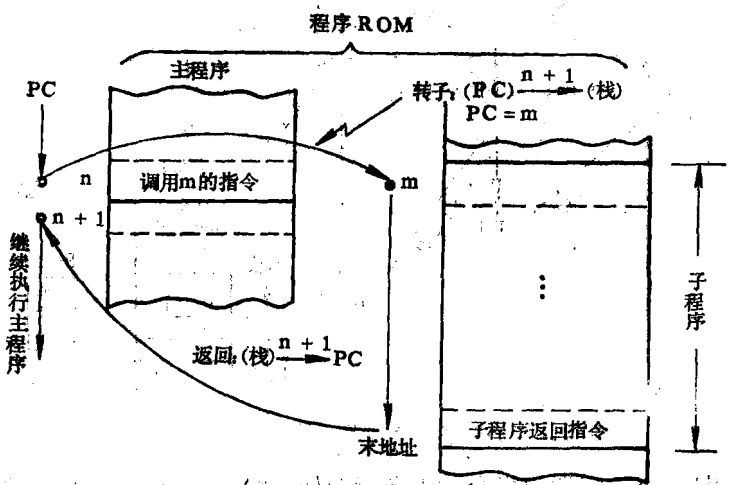


图 1.6 子程序调用时程序的转移

如果在执行一个子程序时，又执行了另一条CALL指令，就会转入另一个——下一级子程序。依此类推，还可以再转入第三个、第四个子程序……。这就叫做子程序的嵌

表（见图1.7）。即调用新的一级子程序，都必须把该级子程序的返回地址加以保存。这样将有三个问题产生：

一、必须在RAM存储器中开辟一个足够大的区域专门用以保存各级返回地址，这个区域叫栈区（Stack）。

二、返回地址的存入和取出必须符合这样的次序：先存入的后取出，后存入的先取出。以保证各级子程序按先后嵌套次序逐级返回。

三、保存返回地址的是RAM单元，这个存储单元的地址由谁提供呢？——指令中只有子程序入口地址，而PC中为返回地址。实际上，在CPU中专门设置了一个寄存器，用以指向栈区。这个寄存器叫堆栈指示器（Stack Pointer），简称SP。

堆栈指示器是这样一个地址寄存器，其初值指向RAM存储器的一个单元，此单元称为栈区的栈底，由于初始时栈中未存任何返回地址，因此这时的栈底即栈顶。在执行子程序调用指令时，CPU首先将SP的地址输出，而后将PC中的返回地址通过数据总线送入SP所指定的RAM单元中去。每向栈区送入一个返回地址字节，SP会自动加一*。这样，当子程序嵌套时，各级返回地址按SP＋1的次序逐级送入栈区中“堆放”。这就是返回地址的入栈过程。最后一次嵌套的子程序返回地址总是被SP现行值－1所指，或说在堆栈的栈顶。因此，每次子程序返回时SP先会自动减一，再从SP所指的单元，即栈顶取回返回地址送入PC中，而SP的新现行值－1后又指向下次返回的返回地址存储单元。这样就实现了返回地址的先进后出——后进先出的次序，保证了各级子程序的正确返回。子程序的调用及堆栈操作示意于图1.7中。

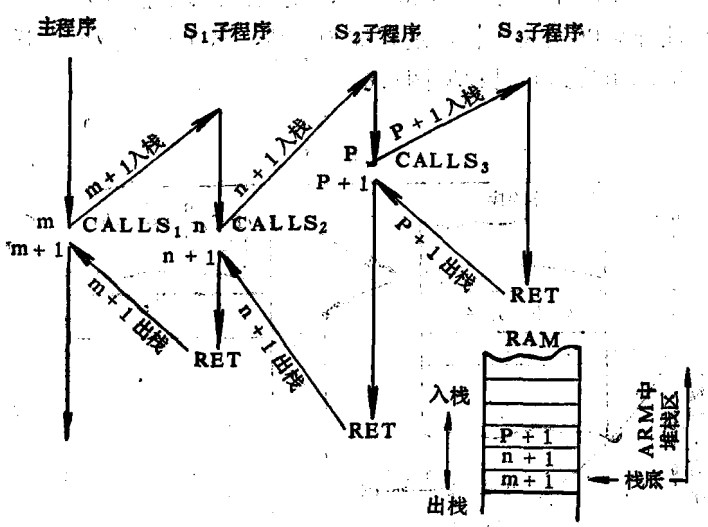


图 1.7 子程序嵌套与返回地址入栈出栈

子程序调用后，会返回调用指令的下条指令处，继续主程序的执行。这一特点，就可以在一个主程序中，通过多处执行CALL指令调用同一个子程序。可见，子程序是一个可

*某些CPU的SP操作正好相反，进栈自动减一，出栈加一。

以公共使用的程序块。因此，在程序设计中，子程序通常用以实现某种通用算法：除法子程序，乘法子程序，开平方子程序，从某接口输入/输出数据的子程序，……。一个子程序可以被主程序多次调用，又只占一块存储区，这就大大提高了存储器的使用效率。调用一次子程序，在主程序中只出现一条调用指令，子程序的功能又是已知的，因此方便了主程序的编写和阅读。

指令寄存器（Instruction Register）和指令译码器

CPU不可缺少的另一个寄存器叫指令寄存器，简称IR。指令寄存器不能通过指令来访问（读出或写入），它是CPU内部专门用以存储现行指令操作码的寄存器。程序指令是存在程序存储器中的，CPU采用读一条指令，执行一条指令的方法来执行一个由很多指令按一定逻辑所组成的程序。我们知道CPU取指令时，要将程序计数器PC中的指令地址输出，而后从该地址读入一个指令字节。那么读入指令字节送入到CPU的哪个寄存器中，又如何处理呢？下面我们先讨论一下指令的结构。

指令在存储器中的形式也是一个二进制的数据。和一般数值的区别在于这个二进制数要代表一种命令，一种CPU要去实现的操作，因此又称之为指令代码。一个8位的指令代码可代表256种操作命令。当操作命令超过256种时，一个字节就不够了，就会用两个字节来代表操作命令。无论是几个字节，在一条指令中代表操作命令的字节叫指令的操作码字节。任何指令的第一个字节一定是操作码字节，而第二个字节就不然了。因为在一条指令中，不仅要指定CPU所要进行的操作，而且有时还要提供参与操作的数据。这些数据可能是一个字节，也可能是多字节的，这就因不同型号CPU而异了。这些字节叫做指令操作数字节。

总之，一条指令可以是单字节或多字节的。而且有的字节是操作码，有的字节是操作数。例如MCS-48单片机有这样一条指令：

0001　0111

是个单字节指令，此操作码指定CPU：将A中的数加1。

另一条指令是：

1111　0110　0000　0010

这是个双字节指令，指令为：“当进位位C＝1时，转入第二个字节所指定的地址去”，即条件C＝1符合时，将第二个字节0000　0010的数据送入PC的低8位中。可见这条指令的第一个字节为操作码，第二个字节为操作数。我们就以这条指令为例来看一看CPU是如何读取一条指令并执行之。

CPU的PC指向这条指令时，按PC地址将操作码读入CPU中的指令寄存器IR中，而后PC自动加一，指向此指令的第二字节。操作码字节存入IR后，经过与IR输出相联接的指令译码器（图1.3）的译码，译码结果在控制器产生了控制脉冲输出。由于指令的操作码已读入IR并译码，CPU“得知”此操作码的命令意图是：“测试标志位C，如果C＝1，则将读入的下个指令字节送入PC中，以实现程序转移，否则读入的下个字节无用，不于置理。”因此，CPU控制将PC（已自动加一，指向第二字节）输出并由控制器发出控制信号读取第二字节。由于第二字节不是操作码，没必要进行译码，当然不必存入