单片机应用丛书之一

MCS-48™

微计算机用户手册

江苏省射阳工业电脑厂资料室

MCS——48™

微计算机用户手册

江苏省射阳工业电脑厂资料室

再　版　前　言

值此单片机应用丛书重印之际，编辑要我为其写一篇前言。我想，关于单片机的原理，书中已讲得很多，此处不必赘述。关于应用，也不是一篇前言所能容纳的。因此，就谈谈所谓“单片机热”吧。

其实，资料室的同志拿出他们的资料销售记录就可以作出一条单片机热的“升温曲线”。一九八三年，国内知道8035的人还廖廖无几；八四年开始推出一些简陋的开发装置，出了一些油印资料；八五年开始，学习班在各地陆续举办，资料开始零星销售；八六年的下半年开始，温度骤升，到了今年可以用“如火如荼，方兴未艾”之类的词汇形容。当初印的几千册资料，担心销不掉，一次讲座便销售一空，每天资料室的索求信“象雪片一样飞来”。更重要的是采用单片机开发的应用项目，在各种杂志、报刊上越来越多。

单片机技术的优点已经被广大工程技术人员所认识，它完全有可能取代以往几乎所有的控制手段，而跃为自动控制和智能仪表领域中的主角。从发展的眼光看，尽管计算机技术发展神速，但单片机技术在技术量的应用寿命大约至少有15至20年。如果没有出人意外的特殊手段出现，按现在计算机技术的发展轨迹，只有等到集成度再提高若干个数量级，並且寻找出一种更巧妙、通用的编程语言，目前的单片机才有可能被彻底淘汰。近几年的发展只仅仅会使位数的增加，速度的提高，内存的增加或者仅仅增添几句强有力的指令而已。

因此我们可以预料，单片机技术会成为所有工程技术人员，工科学生的必修课，就象电工学、电子学一样，一定要拿到60分的成绩报告书才能结业。

编辑告诉我，今年将重印已经出版的三本丛书，除此之外，还将新出一批单片机应用丛书：

丛书之一：《ＭＣＳ—４８微计算机用户手册》	已出
丛书之二：《８０３５汇编语言程序设计手册》	已出
丛书之三：《８０３５单片微机使用参考资料》	已出
丛书之四：《ＭＣＳ—５１微计算机用户手册》	即出
丛书之五：《８０３１汇编语言程序设计手册》	即出
丛书之六：《８０３１单片微机使用参考资料》	待出
丛书之七：《ＭＣＳ—９６微计算机用户手册》	即出
丛书之八：《Ｚ８系列单片微型计算机》	即出

还要出版一些应用专集和两种杂志。资料室同志们的辛勤劳动将会化成大量的应用成果，这是必定无疑的。

茅　洪　生

１９８７年４月

目　录

第一章 引 言

1·0 MCS—48简介：

INTEL公司继1973年制成8080多片微计算机后，于1976年又制成了一系列真正的单片微计算机，它们在一个芯片上包含了一个数字处理系统所需的全部功能。这一系列单片微计算机及它们的各种外围芯片叫做MCS—48微计算机系列。

MCS—48的最早和典型的产品为8048微计算机，它在一个40脚的封装内，包含如下功能：

8位CPU；1K×8 ROM程序存贮器；64×8 RAM数据存贮器；27根I/O线；8位定时/计数器。

单片8048周期时间为2.5μS—5μS，有90条以上单周期或双周期指令，这使它的性能类似于大多数现有的8位多片NMOS微处理机。8048是一个真正的且价格低廉的单片微计算机，而且MCS—48的所有器件都只需单一+5V电源，这也降低了系统的电源成本。

由8048开始的MCS—48微计算机系列，现已发展了许多新品种，比原有品种，功能更强，价格更低。为了扩大单片微计算机的应用范围，这些新微计算机具有与8048相容的指令系统，并能共享8048的开发支持手段。

8049是一个与8048可完全互换的单片微计算机，但它的程序存贮器和数据存贮器容量均两倍于8048。8039与8035是可完全互换的处理机，与8049、8048相比只是没有内部程序存贮器，8039的数据存贮器容量两倍于8035。

8021是一个新的价格非常低的MCS—48系列品种，它的指令系统包含了8048的一个子集和其它一些适于廉价应用的指令。

然而，即使元件成本非常低，由于使用掩模制造ROM使产品设计非常困难，并需很高的开发成本，这将严重影响单片机的使用。Intel公司制造了两种引脚完全相同的8048微计算机，从而解决了这个问题：

带有掩模可编程序ROM程序存贮器的8048，它的价格低廉；

带有用户可编程序和可擦的EPROM程序存贮器的8748，它适于样机开发。

8748基本上是一个单片微计算机实验器，在开发和试造过程中，它能一次次地修改，然后用价廉的8021、8048或8049来进行大规模生产。8748既方便了设计开发，又使开发到生产的转变非常容易。

MCS—48特点：

- 单—5V电源
- 40脚或28脚封装
- ROM或EPROM引脚相同
- 周期时间为2.5μS
- 所有指令需1或2个周期

- 具有单拍功能
- 有8级堆栈
- 有2个工作寄存器区
- 可使用RC、LC、晶振或外部频率源
- 可有时钟输出
- 有掉电维持方式

为了用MCS—48解决各种问题和便于以后的扩展，所有8048、8049的功能能使用特殊的扩展接口或标准存贮器和外围接口从外部来加以扩展。一个价廉有效的I/O扩展手段是使用8243I/O扩展器或标准TTL或CMOS电路。8243在一个24脚的芯片上提供有16根I/O线。对于需大量I/O的系统，可使用多片8243。

对于诸如键盘、显示、串行通讯等应用场合，可使用标准MCS—80/85外围电路。程序和数据存贮器能用标准存贮器片或8355、8155存贮器来加以扩展，其中后两者还包括了可编程序I/O口及定时功能。

对于专门用于接口的场合，可使用8041或8741通用外围接口器件（UPI—41）。UPI—41器件有ROM和EPROM两种形式。它们基本上是8048/8748的派生形式，设计成能与可扩展的MCS—48处理机直接接口，并具有智能I/O能力。8041/8741与MCS—48系列具有相同的指令系统。

8035和8039分别是8048或8049没有内部程序存贮器的形式，它们允许用户用各种外部存贮器来作为程序存贮器。不管用户需多大的程序存贮器，使用8035和8039使用户能选择价格最低的系统。

MCS—48处理机设计成既可作为运算处理机，又可作有效的控制处理机。它们的指令系统允许用户直接置位和复位I/O口的任一位、也能测试累加器的任一位。它们有大量转移、查表指令，使这些处理机能有效地实现标准的逻辑功能。它们的指令编码也很有效，70%以上的指令为单字节长，其它也仅为双字节。这使得在其它计算机需1.5K到2K字节的程序可装入8048的1K ROM中，而一般3K到4K的程序则可装入8049ROM中。(表1见下页)

图1显示了使用8355/8755程序存贮器和I/O扩展器及8155数据存贮器和I/O扩展器来扩展8048/8049的各种可能的组合形式。通过加入8155能把数据存贮器扩展256字节，加入8355/8755能把内部的1K或2K程序存贮器扩展2K字节。如果使用外部存贮器，能用8035/8039来取代8048或8049。

1.1 计算机的功能

本节介绍了计算机的一些基本概念。它为本手册以后几章提供了必要的准备知识。对计算机已经熟悉的人可以跳过这一部分。

1.1.1 典型的计算机系统

一个典型的数字计算机包括：

一个中央处理机（CPU）；程序存贮器；数据存贮器；输入/输出（I/O）口。

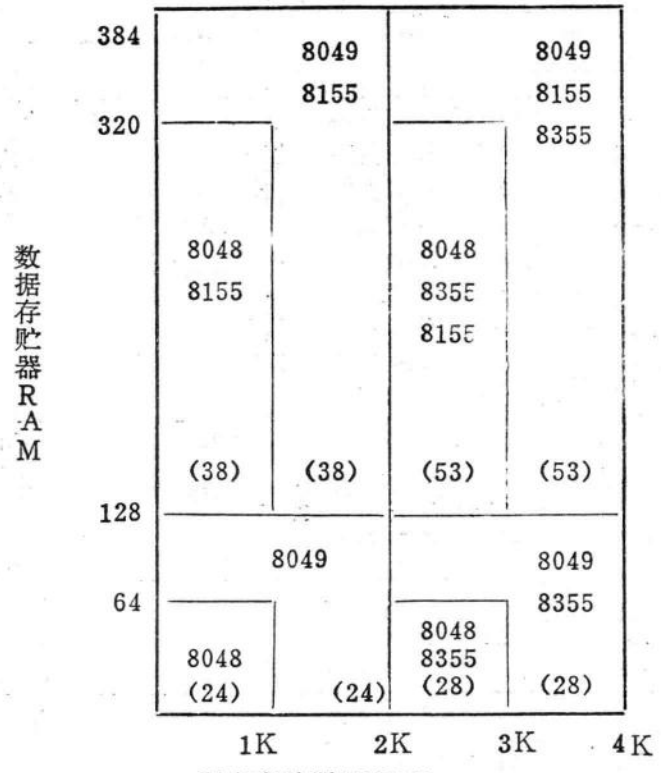
计算机的存贮器用于存放指令。指令是一些控制CPU活动的编码信息。数据存贮器用于存放数据，它们是一些需CPU处理的编码信息。程序为一组存放在存贮器中的逻辑上互相有关的指令。CPU按一定的逻辑顺序从存贮器中读入每一条指令，并用它来控制处理器操作。如

表1.1　　MCS—48微计算机器件

功能		型号	描述	备注
MCS—48	微计算机	8021	1K ROM程序存贮器—10μS周期	计算机，具有掩模制造或光可改写程序存贮器，或无程序存贮器。
		8048	1K ROM程序存贮器	
		8049	2K ROM程序存贮器	
		8035	无程序存贮器，64×8RAM	
		8035L	有掉电保持的8035	
		8039	无程序存贮器，128×8RAM	
		8048—8	1K ROM程序存贮器	
		8748—8	1K EPROM程序存贮器	
		8035—8	无程序存贮器	
	存贮器和I/O扩展器	8355	2K×8 ROM，16 I/O线	与MCS—48兼容，不需其他芯片就能直接扩展MCS—48的功能
		8755A	2K×8 EPROM，16 I/O线	
		8155/56	256×8 RAM，22 I/O线，定时器	
	I/O扩展器	8243	16线 I/O扩展器	价廉I/O扩展器
相容的MCS—80/85器件	标准 ROM	8308	1K×8 450nS	可廉价扩展程序存贮器8308、2316E分别与8708、2716互换。
		2316E	2K×8 450nS	
	标准EPROM	2708	1K×8 450nS 光可改写	用户可编程序和可擦除
		2716	2K×8 450nS 光可改写	
		2732	4K×8 450nS 光可改写	
	标准 RAM	8111A-4	256×4 450nS I/O共线	用标准 NMOS RAM可容易地扩展数据存贮器。5101 CMOS功耗仅为75nw/bit。
		8101A-4	256×4 450nS I/O分开	
		5101	256×4 650nS CMOS	
		2114—4	1K×4 450nS I/O共线	
	标准I/O	8212	8位I/O口	用作地址锁存或I/O口
		8255A	可编程序外围接口	3个8位可编程序I/O口
		8251A	可编程序通讯接口	串行通讯接收／发送器
	标准外围芯片	8205	三一八线译码器	这些是MCS—80外围器件，它们与MCS—48相容，能为它提供特别的接口功能，如8279键盘/显示接口。将来的MCS—8085器件也将是相容的。
		8214	优先级中断控制器	
		8216	双向总线驱动器	
		8226	双向总线驱动器(反向)	
		8253	可编程序定时器	
		8259	可编程序中断控制器	
		8279	可编程序键盘/显示接口(64键)	
		8278	可编程序键盘/显示接口(128键)	
	通用外围接口	8041	ROM程序存贮器	用户可编程序以执行任何I/O和控制功能。
		8741	EPROM程序存贮器	

译者注1：1979年后生产的8039的周期为1.39μS。

注2：1979年又生产了一种新的单片机8022，它比8021的ROM增加了一倍，I/O口也增加了，同时片上还有两路8位A/D变换器。



程序存贮器(ROM)

( )可得到的I/O线数

图 1 .1扩展的MCS—48系统

果程序序列是正确的，则处理这程序将产生有用的结果。程序必须编排得使当CPU读指令时不会读入一个非指令字。

CPU能迅速地访问存放在存贮器中的任何数据，但经常存贮器并不大得足以存放某个特定应用场合中所需的全部数据。给计算机附加一个或更多的输入口，可解决这个问题。CPU能给这些口编地址，并输入这些口中的数据。附加输入口还能使计算机能从外部设备（如纸带读入器或软磁盘）中高速、大量地接受信息。

计算机同样也需要一个或更多个输出口，它们允许CPU把处理结果输送到计算机外部去。输出可以到显示器供操作员使用，也可以到产生“硬抄件”的外围设备，如行式打印机，也可以到一个外部存贮设备如软磁盘，或者可以作为控制信号来控制另一个系统的运行，如自动装配流水线。象输入口一样，输出口也可以被编址使用。输入和输出口一起能允许处理机与外界通讯。

CPU是整个系统的核心。它控制其它部件执行的功能。CPU能从存贮器中取出指令，译出其二进制内容并执行它们。在执行指令时，CPU也能访问数据存贮器和I/O口。此外，CPU应能识别和响应某些外部控制信号，如中断请求。下面介绍执行这些功能的CPU内的各个功能部件。

1.1.2 CPU的结构

典型的中央处理机（CPU）由如下的内部相联的功能部件所组成：

寄存器；算术/逻辑部件（ALU）；控制电路。

寄存器是CPU内的暂时存贮器件。一些寄存器，如程序计数器和指令寄存器，具有特定的用处。其它寄存器如累加器等为通用寄存器。

累加器：

累加器通常用来存放ALU处理的一个操作数。一条典型的指令可能控制ALU把别的一个寄存器中的内容与累加器中的内容相加，并把结果仍放在累加器中。一般情况下，累加器既是源（操作数）寄存器，又是目的（结果）寄存器。一般CPU内有许多通用寄存器，用于存放操作数或中间数据。有了通用寄存器能减少把中间结果在存贮器与累加器中传来传去，从而能提高处理速度和效率。

程序计数器（转移，子程序和堆栈）：

组成一个程序的指令存放在系统的存贮器中。中央处理机按照存贮器的内容来决定应执行什么操作。这意味着处理机必须知道下一条指令在哪一个单元中。

存贮器的每一个单元用数字来标记，以使它与存贮器中所有其他单元相区别。标记一个存贮器单元的数字称为它的地址。处理机中有一个计数器。它包含了下一条程序指令的地址。这个计数器叫做程序计数器。每次CPU取出一条指令，它把程序计数器的内容加1，从而更新了它的内容，因此程序计数器经常是现行的——指向下一条指令。

程序员把他的指令按数字顺序地址存放，所以较低的地址中为前面的指令，而较高的地址存放后继指令。只有当一个存贮区中的一条指令是一条转向另一个存贮区的转移指令时，程序员才能违背这个顺序规则。

一条转移指令包含它所转向的指令的地址。只要程序转移指明了正确的地址，下一条指令可以存放在存贮器的任何单元中。在执行转移指令时，处理机用转移指令中的地址来替换程序计数器的内容。这样保持了程序的逻辑上的连续性。

当存放的程序“调用”一个子程序时，发生一种特别的程序转移。在这种转移中，处理机需要记住转移发生时的程序计数器内容。这使处理机在完成了程序的最后一条指令后能重新开始主程序的执行。

一个子程序是包含在一个程序里的程序。通常它是一套在执行主程序的过程中需重复执行的通用指令。被写成子程序形式的功能的典型例子为计算程序变量的平方、正弦或者对数的功能。设计用来从一个特定的外部设备输入数据的程序为子程序的另一个例子。

为了确保有次序地返回主程序，处理机有一种处理子程序的特别方法。当处理机收到一条调用子程序指令，它把程序计数器内容加1，并把计数器的内容存放在一个叫做堆栈的存贮器的保留区域中。这样堆栈中保存了子程序完成后应执行的指令地址。然后处理机把调用子程序指令所指明的地址装入程序计数器中。因而下一条取出的指令将是子程序的第一条指令。

任何子程序的最后一条指令是返回指令，它不需指明地址。当处理机取出一条返回指令时，它简单地把堆栈顶部的地址来替换程序计数器的现行内容。这使处理机重新开始执行原来调用指令后的主程序。

子程序经常为嵌套的。也就是说一个子程序有时会调用第二个子程序，第二个可能调用第三个，如此等等。只要处理机有足够的能力来存放必要的返回地址并且逻辑上允许这样做，嵌套是完全允许的。换句话说，嵌套的最大层次取决于堆栈的层次。如果堆栈可存放三个返回地址，则可实现三级子程序。

指令寄存器和译码器：

字长是计算机的特征之一。一台计算机的字长通常由它的内部存贮器件和内部连接通路（称为总线(BUS)）所决定，例如寄存器和总线能存贮和传输8位数据的计算机的字长为8位，并且称为8位并行处理机。一台8位并行处理机能有效地处理8位二进制数，因而与它相连的存贮器的每一个可寻址单元也组织成可存贮8位。数据和指令按8位二进制数或8的整数倍如16位、24位等等形式存放在存贮器中。这样的8位数经常被称为字节（Byte）。然而，如果需要有效地处理4位或甚至1位数据，可使用特殊的处理机指令。

处理机能执行的每一个操作对应于一个唯一的数据字节——指令码或操作码。一个8位字用于指令码能标识256种不同的操作，这超过大多数处理机的需要。

处理机分二步取出一条指令。首先处理机把程序计数器中的地址传输到程序存贮器，然后程序存贮器把该地址单元中的字节回送到处理机中。CPU把这个指令字节存放在指令寄存器中，并用它来控制余下的指令执行期间的活动。

存放在指令寄存器中的8位字节能被译码，并用于有选择地激活许多输出线，每根输出线代表与一个特定的指令码的执行有关的一组活动。使能的线能与某些定时脉冲组合，来产生用于初始化特定活动的电信号。这个把编码变为活动的翻译过程由指令译码器和相连的控制电路所执行。

一个8位指令一般足以定义一个特定的处理活动。然而有时执行一条指令需要的信息可能超过8位。

需标记一个存贮器单元的指令就是一个这样的例子。基本的指令码标识了执行的操作，但不能指出目标地址。在象这样的情况下需双字节指令。相继的指令字节存放在顺序相连的存贮器单元中，处理机连续执行两次取指令操作从而获得完整的指令。从存贮器中取出的第一个字节放于指令寄存器中，后继字节存放在暂存寄存器中，然后处理机可执行后面的操作步。

地址寄存器：

CPU可能用一个寄存器来保存要访问的存贮器单元的地址。如果地址寄存器是可编程序的（即有指令使程序员可改变这寄存器的内容），程序可在执行存贮器访问指令（即从存贮器中读出数据、把数据写入存贮器或对存贮器中的数据执行其它操作的指令）以前在地址寄存器中“建立”一个地址。

算术/逻辑部件（ALU）：

所有处理机都有一个算术/逻辑部件，它通常被简称为ALU。正如它的名字所表示的，ALU是对二进制数据执行算术和逻辑运算的CPU硬件部件。

ALU必须包含一个加法器，它能按二进制算术逻辑的规定合并二个寄存器的内容。它允许处理机对从存贮器和其它输入源得到的数据进行算术运算。

只用基本的加法器，有经验的程序员能写出执行减法、乘法和除法的子程序，这使计算

机具有各种算术运算能力。然而实际上大多数ALU提供有其它功能，包括布尔逻辑代数运算和移位能力。

ALU中有标志位（Elag Bit），它标志算术逻辑运算中产生的各种状态。可以以一个或更多的标志状态来决定程序是否要进行转移。这样可编写含有条件转移的程序，例在一条加法指令后，如进位为“1”，则移向一段特定的程序。

控制电路：

控制电路是CPU中的基本功能部件。利用时钟脉冲输入，控制电路产生各种处理活动所需的适当的信号序列。取入和译出一条指令后，控制电路发出适当的信号（到CPU内部的和外部的部件）来进行正确的处理活动。控制电路还能响应外部信号如中断。中断请求能使控制电路暂时中断主程序的执行，转向一个特别的子程序去为请求中断的设备服务，然后自动返回主程序。

1．1.3 计算机操作

有一些几乎所有计算机都有的基本操作，在学习一台特定的计算机的操作前，必须深入理解这些操作。

定时：

中央处理机的活动是周期性的。处理机取一条指令，执行所需的操作，取下一条指令，如此等等。这种有次序的活动序列，要求准确的定时，这样CPU需要一个自激振荡器时钟来供所有处理机活动作参考。取指令和执行该指令的过程叫一个指令周期。周期的各段由具有明确定义的活动——状态所标记。振荡器的定时脉冲之间的间隔时间叫做一个时钟周期。一般来说，一个状态包括一个或更多个时钟周期，而一个指令周期有几个状态。

取指令：

每一个指令周期的第一个状态必须是取下一条指令。CPU发出一个读信号，程序计数器的内容发送到程序存贮器，而存贮器则发回下一条指令字。指令的第一字节放于指令寄存器中。如果指令包含不止一个字节，则需额外的状态来取指令的第二字节。当全部指令取到CPU中时，程序计数器加1（准备取下一条指令），同时对指令进行译码。指令所定义的操作在指令周期的其余状态中执行。指令可能需进行读、写数据存贮器、输入或输出、或进行CPU内部操作，如寄存器到寄存器的传输或加法操作。

读存贮器：

取指令是一个特别的程序存贮器读操作，它把指令取到CPU的指令寄存器中。以后，取来的指令可能需从数据存贮器中把数据读入CPU。CPU再次发出一个读信号和适当的存贮器地址，存贮器进行响应发回所需的字。接收到的数据放于累加器或其它一个通用寄存器中（非指令寄存器）。

写存贮器：

写存贮器操作与读操作差不多，只是数据流向不同。CPU发出一个写脉冲和适当的存贮器地址，然后送出数据写入寻址的数据存贮器单元中。

输入/输出：

输入和输出操作与存贮器读和写操作差不多，只是寻址的是I/O口，而不是存贮器单元。CPU发出适当的输入或输出控制信号和地址，然后或者输入数据或者输出数据。

数据能以并行或串行的形式输入/输出，数字计算机中所有数据均为二进制码形式。一个二进制数据字包括一组字位（bit），每一位或是1或是0，并行I/O同时传输一个字的所有位，一根线一位。串行I/O一次在一根单线上传输一位。显然串行I/O要慢得多，但比并行I/O所需的硬件要少得多。

中断：

许多中央处理机有中断功能，它能提高处理机的效率。我们来考虑一台计算机对大量数据进行处理，而这些数据中的一部分需输出到一台打印机去的情况。CPU能在一个机器周期内输出一个数据字，但打印机却需相当于许多个机器周期时间才能实际打印出这个数据字所表示的字符。这样在打印机能接受下一个数据字之前，CPU只能处于等待状态。如果计算机有中断功能，则CPU可以先输出一个数据字，然后去处理数据。当打印机可以接受下一个数据字时，它发出中断请求。CPU响应中断后，暂停主程序的执行，自动转向输出下一个数据字的子程序。在输出一个数据字后，CPU继续执行主程序。注意这与子程序调用在原则上是相同的，只是转移由外部引起而不是由程序引起。

还有更复杂的中断结构；n个中断设备共享同一处理机，但有不同的优先级。中断处理是一个重要的特性，在系统吞吐量要求较高时它能有效地实现处理机的处理能力。

1.2 微计算机程序设计

1.2.1 机器语言程序设计

通过编写由一系列指令组成的程序，并把它们存放在程序存贮器中，可控制计算机的操作。处理机一次取出一条指令，并执行由它指定的操作。这些指令必须按处理机能理解的形式贮存。这种形式叫做机器语言。对于大多数微计算机来说，指令是一组8位二进制码（“1”和“0”），叫做一个字（Word）（如果一个字为8位，也叫做1个字节(byte)。有些指令需要一个以上程序存贮器单元。执行多字节指令时，处理机必须先对程序存贮器执行多次取操作。因为多字节指令比单字节指令需占用更多的程序存贮器单元并需较长的执行时间，所以通常尽量少用多字节指令。

表1.2

程序步号	机器码	解释
0	10111000	把十进制数32装入
1	00100000	寄存器R0中
2	10111010	把十进制数5装入
3	00000101	寄存器R2中
4	00001001	从口1中输入到累加器中
5	10100000	把累加器的内容传送到由寄存器R0寻址的寄存器中
6	00011000	把R0内容加1
7	11101010	把R2内容减1，如果结果为0则继续执行步9，如果不为0，则转向步4。
8	00000100	
9	—	

通过用二进制码（“1”和“0”）写一系列机器能直接译码的指令，能为处理机编写程序，这就是机器语言程序设计。在需写的程序较短时或在需设计者熟悉微处理机的应用场合中，机器语言程序设计十分有用。由于用户对机器比较熟悉，机器语言程序设计允许用户使用各种程序设计技巧编写出最有效的可能编码。

上面是机器语言程序设计的一个例子：这个程序从一个I/O口顺序读入5个8位字并把它们顺序存放在数据存贮器中。程序一开始，先初始化二个寄存器，一个决定把数据存放在哪儿，另一个计数存贮的字数。结束后处理机继续执行后面的指令。

你们可以看出，用“1”和“0”的形式写机器指令非常吃力并容易产生错误。如果机器指令用一种缩写形式——十六进制数来编写，可更有效地表示每一个8位字节。由于这种方法使用十六进制记数法所用的字符集，故它叫做十六进制数。十六进制数只是通过加入英文字母表的前六个字母对一般的十进制数进行扩展而得到的。这样它有16个不同的字符，每一个十六进制“数字”能代表16个值或等价于4位二进制数的16个量，因而每个8位机器语言字能用2个16进制数（记为hex）来表示。十进制数、二进制数和十六进制数系统的对照如下：

表1.3

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

这样前面的机器语言程序可写为：（见下表）

现在的这种编码方法读、写都非常容易，并且编码的错误也容易查找。十六进制码对于短的程序（几百行）通常很有效，然而对于更大的程序，它有两个主要缺陷：

1.十六进制码不是自含说明的，也就是说编码本身没有用人的术语来指出所执行的操

表1.4

程序步	十六进制码
0	B 8
1	2 0
2	BA
3	0 5
4	0 9
5	A 0
6	1 8
7	E A
8	0 4

作。用户必须学习每一种编码或不断用程序对照卡片来进行变换。

2.十六进制码是绝对的，即程序仅仅贮存在程序存贮器的特定单元中才能工作，这是由于程序中的分支或转移指令需转向程序中其它确定的地址。在前面的例子中，步7和8转向步（或地址）4。如果要移动这段程序，步8必须变成转向步4的新地址。

1.2.2 汇编语言程序设计

汇编语言使用字母——数字符号来表示机器操作码、分支地址和其它操作数，克服了机器语言的缺点。例如，把寄存器R0加1的指令变为INC R0而不是16进制数18，使用户一眼就能看出这指令的意义。（译者注：INC为英文Increment（加1）的缩写）我们的程序用汇编语言可写为如下形式：

表1.5

步骤号	十六进制码		汇编代码
0	B8		MOV R0, #32
1	20		
2	BA		MOV R2, #05
3	05		
4	09	INP:	IN A, P1
5	A0		MOV @R0, A
6	18		INC R0
7	EA		DJNZ R2, INP
8	04		

第一条语句能描述如下：把十进制数32传送到寄存器R0中，传送指令通常安排成目的地为第一位，而源为第二位。记号“#”指出源为“立即”数（数据包含在程序存贮器的下一个字节中）。在本程序中，数据指定为十进制数32，然而数据可写为十六进制20H或二进制00100000B，因为汇编程序能接收任一种形式的数据。注意，在这里二行十六进制码用一

行汇编代码来表示。

输入指令IN A，P1具有与MOV指令相同的形式，它指出应把口1的内容传送到累加器中。在输入指令前有一个地址标号，它用一个冒号结尾。这个标号允许写的程序独立于它在程序存贮器中的实际地址，因为在程序结尾的分支指令能按这个标号而不是一个特定的地址转移。这是汇编语言程序的一个非常重要的优点，因为它允许在调试时在程序中加入或删去指令，而不必要改变任何转移地址。

下一条指令MOV @R0，A描述如下：把累加器的内容传送到由R0所指出的数据存贮器单元中。记号@表示间接操作，这操作把寄存器R0或寄存器R1的内容作为指向需对其执行操作的数据存贮器单元的指针。

最后一条指令为减1和不为0转移指令，它把特定的一个寄存器作为循环计数器。在这种情况下，寄存器R2原先装入5，然后每次执行循环时减1。如减1的结果不为0，程序转向INP，并执行下一个输入操作。第五次循环后，结果为0，顺序执行DJNZ指令后的其它程序。

除了一般汇编程序的功能外，MCS—48提供有更先进的汇编程序，如在汇编时进行表达式的赋值，条件汇编和宏汇编。

1.表达式的赋值——某些汇编程序允许在指令的操作数场中使用算术表达式和各种符号，如MCS—48汇编程序能接受如下的指令：ADD A，#ALFA *BETA／2。

ALFA和BETA是二个前面已定义的符号，在汇编时，表达式ALFA *BETA／2将被赋值，而结果数（它是ALFA和BETA的平均值）将被作为立即数和ADD立即数指令的第二个字节。这个表达式允许这指令的立即数作为一个单一语句，省去了使用等于ALFA*BETA／2的第三个符号。

2.条件汇编——条件汇编允许程序员在汇编时只把他的一部分汇编语言程序（源程序）变换成机器码（目标代码）。例如，这允许在开发时，在程序中写入各种调试程序。使用条件汇编，在最后进行汇编时能删去这些调试程序。

条件汇编也允许在汇编时，选择一个大程序的各种各样部分，以产生一个基本程序的各种版本。

3.宏汇编——宏指令实际上是汇编程序能识别的一个符号。它代表由某些标准指令组成的一个序列。使用宏汇编可在程序的几个地方产生相同的指令序列，而不必每次使用这序列时重新写出它。例如，一个典型的宏指令可能为执行减法操作的指令串。8048没有减法指令，但这操作能容易地用下面三条指来实现：

```
CPL    A
ADD    A，REG
CPL    A
```

这程序从累加器减去一个寄存器的内容，并把结果留在累加器中，这个序列能定义为宏指令，它的名字为SUB，操作数为R0到R7。这样从累加器减去R7，程序员只需写：SUBR7。汇编程序自动插入以上三条指令，并用R7来代替REG。

写好了汇编语言源程序，可以通过汇编程序把它变换成可执行的目标代码。MCS—48汇编程序是在以8080为处理机的Intellec MDS系统上运行的一个程序，这将在下一节介绍。