上海交通大学《微型计算机》(53)

微机通用汇编语言程序设计

上海交通大学科技交流室

一九八七年十二月

微机通用汇编语言程序设计

白英彩　章仁龙

上海交通大学科技交流室

内 容 简 介

本书介绍如何利用通用汇编语言编写适用于Z-80、8080/8085、M6800和6502等八位微机的应用程序。该通用汇编语言是由上述各机的通用助记符和伪指令组成的。借助通用汇编程序和通用反汇编程序，可以实现上述几种微机间程序的互相转换。通用汇编语言易学、易记、易用和编程简便。执行由通用汇编语言所产生的目标程序，具有执行速度快和节省存贮空间等。且解决了微机软件的兼容问题。

本书编辑：罗宗信、蔡锡泉
徐建明、杨福兴

前　言

一种适用于Z-80，8080/8085. M6800，6502等8位微处理机的通用汇编系统，经过几年的潜心研究，终于成功了。通用汇编系统中包括通用汇编语言(由通用助记符系统GMS和伪指令组成)、通用汇编程序GA和通用反汇编程序GDA等三部分。

本书旨在介绍如何利用这种通用汇编语言编写上述各种微处理机的应用程序。对于编程人员而言，利用通用汇编语言编程，犹如利用高级语言编程一样简便；对于具体微机而言，执行通用汇编语言所产生的目标程序，具有执行速度快、节省存贮空间，即程序效率高的特点。

几年来试用效果表明，这种以通用助记符为基础的通用汇编语言，对于那些英语知识薄弱的非计算机人员来说，是十分乐于接受的。

此外，由通用汇编程序和通用反汇编程序(其软盘盘片将在不久大量提供)，可以实现各种微处理机间程序的互相转换，将大大地节省程序人员的时间和精力，它将为推广普及微机应用打开新的局面。

当然，关于使用通用汇编系统时还可能遇到一些问题，例如，当出现硬件中断指令或系统调用时，要实现程序的转换，还需要人的干预，才能最终解决。深信，通过本书的介绍，将会获得“抛砖引玉”的结果。

书中援引了大量编程实例，说明怎样运用通用汇编语言来编写复杂程序。

为了说明某种微处理机的目标程序能容易地转换为其他微处理机的目标程序，为此，将第四章中的各个示例，除给出其通用汇编语言源程序外，还在各个示例中给出某种微处理机的目标程序，(事实上，可以同时地给出上述各种微处理机的目标程序)。对第四章示例的校对和汇编出目标码的工作，是由刘晓平、郭晓萍和陈继龙等同志协助完成的，杭诚方同志、黄德坤同志贾维生同志审读过该书手稿、并提出有益的建议和意见，在此，一并表示谢意。

书中定有不少欠妥之处和错误之处，请指正。

编　者

1987年11月

目　录

第一章 概 述

第一节 引 言

目前，电子计算机，特别是微型计算机的飞速发展和广泛应用，几乎已渗透到所有的科学研究、工农业生产、文化教育、社会生活的各个领域，并且正在引起一场新的技术革命，它将把人类推向一个全新的信息化和智能化的社会。因此，为了迎接这场工业革命的到来，适应电脑化、自动化、信息化社会的需要，必需大力普及计算机教育，推广计算机应用。

就计算机应用来说，软件的重要性是不言而喻的。任何一台计算机，一旦失去正确程序的指引，必然立即瘫痪。而一个程序设计得好，不仅使得计算机正确无误地完成所指定的任务，而且工作周期短，所占内存空间少，运算精度高。在计算机科学中，程序设计技术发展很快，日益显示其重要性，并已形成一门独立的学科。

随着程序设计学科的发展，编写程序所用的语言从简单到复杂，从初级语言到高级语言，门类齐全，种类繁多，所谓初级语言是指机器语言和汇编语言，它们是与具体所用的计算机指令系统密切相关的。所谓高级语言是指ALGOL,FORTRAN,BASIC,COBOL,PASCAL和PL/1等程序设计语言，它们的用法与具体的机器指令系统无关。本书在进行程序设计时，采用汇编语言。一是因为高级语言更适合于数值问题，而本书则主要讨论非数值问题。所谓非数值问题，就是逻辑判断处于主要地位，而算术运算相对地处于次要地位的问题。二是因为汇编语言无论是在执行速度或占用空间上都更有效，更接近实际。三是汇编语言本身也是需要的，有时甚至是重要的。例如，要求响应速度的实时控制程序。

既然采用了汇编语言，那么采用哪一种汇编语言呢？采用Z80或PDP-11，以及其它型号的微机的汇编语言是不行的，因为这样不但会把本身局限于某种型号的微机，而且还带来以下所讲的问题。所以我们采用一种“通用的”汇编语言。

第二节 微机教学必须改革

一、微机教学存在问题

在第三届世界教育应用计算机会议上指出：“程序设计是人类的第二文化”（第一文化系指阅读和写作能力）。不久，计算机将会普及到这种地步：不会阅读和编写计算机程序，就相当于现在的文盲。因此，世界各国在高等教育，甚至在中等教育中都把普及计算机教育作为重要内容。

然而，微型机的教学无论从质量上还是从发展速度上看，还远远不能适应微型机技术研究和普及与提高的需要，显而易见，如何提高微型机教学水平和效率，加速培养成批的研究和应用微机的人才已是推动微机进一步向前发展的一个重要课题，因而也是微机教学的重大课题。

微机教学水平提不高或进展迟缓的一个重要因素是，微机汇编语言、指令系统的教学质量不高，效果不理想。然而，凡从事微机工作的人，又必须通过汇编语言的学习和训练。因为汇编语言及其汇编程序特别适用于微型机，主要有以下五点理由：

（1）用汇编语言编程再经过汇编程序“汇编”为目标程序，其效率最高，而以高级语言编程再经过编译程序或解释程序“翻译”为目标程序，其效率很低，即占用的存贮空间大，机时多。由于微型机的存贮空间较小，速度也慢。因而要求其执行的目标程序的效率越高越好。

（2）微型机有许多多字节指令，只有汇编程序能够方便地选择适当的指令字节数，并对程序计数器保持跟踪，也能正确地划分多字节的操作数和地址中的各个字节。

（3）使用汇编语言编程可允许有多个起始地址，绝大多数微处理机的程序都要求有多个起始地址，以便适当地安排复位和中断服务程序，并能方便的避开为存贮数据的保留区和为输入/输出设备设置的保留区。

（4）使用汇编语言便于将固定的数据和变量分开存贮。常数保存在ROM，变量保存在RAM。

（5）单板机和单片机主要使用汇编语言编程。

正因为汇编语言的这些优点，所以各代微型机都配有汇编语言，这就发生了与此有关的以下三种情况：

① 微机汇编语言，指令系统的教学在整个微机教学中占极重要的地位，所谓微机教学中应遵循的“软硬结合”的原则，其软件主要是指汇编语言编写程序，在微机应用中，为了降低软件的开销，充分利用微机的特性以及为了适应某些场合（如输入/输出过程）的特殊需要，通常都要求使用汇编语言编制各种应用程序，所以对掌握汇编语言编程技术的要求也就特别高。因而把汇编语言视为微机与人的主要“软接口”，它贯穿在微机软件甚至是整个微机教学的始终。

由于汇编语言在微机教学中占有如此重要的地位，所以汇编语言本身的问题或它的教学方法问题就自然左右着整个微机教学的局面。

② 汇编语言的根本问题在于，它是面向机器的语言，而不是面向过程的语言。这就是说，汇编语言过分地依赖于微型机的硬件结构，它与计算机结构的关系，比之它与工程任务的关系显得更为密切。由于各种微机的系统结构，指令系统和寻址方式互不相同，因而每种机器都有它自己专用的汇编语言，针对某一机器（如Z—80或8080型微处理机）所编写的程序，很难直接地搬到另一种微型处理机（如M6800）上去运行。这在当今软件价格日益高涨（据称，每字节价值10～30美元）情况下是十分不利的，它势必浪费大量软件工时。

这就是说，使用汇编语言要求程序人员十分熟悉某特定微机指令系统，否则无法编程，即使勉强地编出程序，也是十分欠佳。以致程序员使用寄存器和编制指令序列所花费的时间比他解决实际问题的时间多得多。

③ 各种微机汇编语言，均以其指令名称的英文缩写来代表指令语言，这对英语基础薄弱的计算机使用者来说，要记忆数以百计的指令名称是件困难的事。

据报道，当前8位微处理机已有上百种，它们很少互相兼容或个别机种只有单向兼容性，这就意味着，同时存在上百种微处理机的汇编语言。诚然，微机市场竞争的结果，使有一些消失了，现在最流行的只有五、六种（如Z—80，8080/8085，M6800和6502等），但也难免会不

断地出现新的机种。它们的指令系统往往差异很大，而且即使有着功能相同的指令，也是采用各自的助记符和相应的汇编语言规则。这就使微机的教学很难求得对汇编语言教学内容和方式的一个较长时期的相对稳定性和一致性；而一直处于穷于应付的局面，十分被动，一本教科书中常常要介绍多种微型机的汇编语言(这只在十分有限的场合，对某些专门训练来说，才是必要的)。这些数目大量的以单词缩写方式来形成的“助记符”，实际上令人望而生畏，难以记忆和学习。而且，还会因注意了各种细枝末节而忽略了微机汇编指令系统的根本特征和共同部分，以致被束缚在一、二种教学中讲授的机型上，同时为记忆和学习这些指令系统要耗费许多精力，使整个微机教学的效率很低。

综上所述，微型机教学的改革确实是势在必行，既要继续沿用汇编语言，又必须改革逐一罗列每种微机汇编语言的老的教学方法。

二、微机教学改革的设想

微机的汇编语言及其指令系统应具有一种标准的形式和灵活可变的通用性，它应当既能集中各种微机指令系统的共同点，又能反映不同机器的一些具体特点。既便于记忆和学习，又有助于编出各种类型的程序，特别是能方便地把某一机种已有的程序变换为适用于另一机种的程序，从而减轻使用者的负担，而使汇编语言仍不失其作为认识和使用微机技术的重要手段和地位。

国外，在这方面已有众多学者对此提出各种见解和方案。英国Bristel大学的F.Duncan就是一个范例，他用某种通用表示法在微机教学上进行了试验，取得了积极的效果。

在国内，也有许多计算机专家在呼吁微机教学改革。一些专家指出：“目前，(微机)品种繁多，型号杂乱。这种情况既不便于经验交流，又占有大量软件人员。同样一个题目，由于使用机型不同，不得不重新编制或修改应用程序”。“应该通过各种手段，使各个系列的机器相互在软件上兼容，这对我国微机的发展是很有意义的”。

因此，我们设想采用通用汇编语言进行微机教学，并在微机领域逐步推广通用汇编系统。

第二章　通用汇编指令系统

第一节　指令系统基本概念

微型计算机虽说神通广大，可是一旦离开了指令的控制，却变得一无所能，这一节先介绍一些与指令有关的基本知识。

一、程序和指令系统

众所周知，计算机所以能脱离人的直接干予，自动地进行计算或控制，这是由于人把实现这个计算或控制的一步步操作用指令的形式预先输入到存贮器中，在执行时，计算机把这些指令一条条地取出来，加以翻译和执行。

我们把要求计算机执行的各种操作用命令的形式写下来，这就是指令（Instruction）。每条指令都使计算机执行一个指定的操作。由于计算机只能直接接受二进制码，所以操作指令和数据都要用二进制码表示，只有用二进制码表示的指令码才能被计算机直接识别，人们称它为指令的机器码（Machine Code）。

计算机能识别的各种指令的集合称为指令系统（Instruction Set）。计算机能执行什么样的操作，能做多少种操作，是由设计计算机时所规定的指令系统决定的。指令的种类越多越全面，一条指令完成的操作越复杂，则指令系统的功能越强，一台计算机的指令系统反映了该计算机的功能，功能不同的计算机其指令系统自然也就大有差异。

在我们使用计算机时，必须要把我们要解决的问题编成一条条指令，把一系列指令按一定次序组合在一起就构成了程序。每段程序都能让计算机执行一个完整的任务。用户为解决自己的问题所编的程序，称为源程序（Source Program）。

二、指令的组成和格式

为了命令计算机执行某种操作，一条指令一般应包含如下内容：

（1）操作码

指明计算机完成的操作性质（如加、减、逻辑与、逻辑或、移位、输入，输出等等）。

（2）源地址

指明被操作的数来自何方。有的指令有两个数据参加其操作（如被加数、加数），便有两个源地址，大多数指令只有一个数据参加操作（如移位，求补，传送等等）。

（3）目的地址

指明操作结果的存放地址。

（4）下一条指令的地址。

指明下一条指令到哪里去取，以此保证程序连续执行。

这就是四地址的指令：（示意图见下页）

由于程序通常是顺序执行的，所以程序中的指令在存贮器中也是一条条顺序存放的，计算机在执行时要把这些指令一条条取出来加以执行，所以计算机中设置了一个程序计数器

操作码	第一操作数地址	第二操作数地址	操作结果地址	下一条指令的地址

PC（Program Counter），来自动追踪指令所在的地址。即把给出下一条指令的地址任务交给PC来完成，于是指令简化成三地址格式。

一般八位微型机的内存空间为65536单元（64k），则地址就需16位长（$2^{16}=65536$），如采用三地址格式指令，那么指令占用位数就很长，这不适合于字长较短的微型机。

为了适应微型机的特点，就必须减少地址个数，设法缩短指令，同时又要包含以上各种内容，于是，采取以下措施：

（1）减少地址个数

首先是指定操作结果的地址为两个操作数地址中的一个，形成二地址格式指令，这样虽然破坏了第二操作数（已有操作结果代替），但分析表明，原始数据一般并无保存价值。

操作码	第一操作数地址	第二操作数地址兼放操作结果

其次是将一个地址由操作码隐含指出（如微机中的累加器A），于是指令中只给出一个地址就行了。这就是单地址指令。

操作码	操作数地址

例如把累加器的数和另一数进行操作，结果放到累加器中。累加器由操作码隐含指出，指令中只给出源地址就行了。

（2）间接给出地址

指令中以很少的位数（2至3位）间接给出16位内存地址或通过计算得到有效地址。

例如要把某内存单元中的数送累加器，单是地址就占16位，但当我们事先把16位地址存入DE寄存器对时，则指令只要规定把DE寄存器对指定的单元中的数送累加器就行了。所以一般在微型机中这样的指令全长只有八位。

八位微型机的数据宽度和寄存器都是八位的，即字长为八位，称为一个字节（Byte）。但指令却不可能都缩短成八位。为解决这个矛盾，人们把指令码分段存放和处理，每段与主机字长相同。如八位微型机指令以八位为一段，而十六位微型机指令则以十六位为一段。

例如八位微处理机要往内存单元送数需用三字节指令。第一字节指明操作，第二字节和第三字节是十六位内存地址。

用一个字节作操作码，则可能有$2^8=256$种不同的操作。这些操作由微处理机的指令译码器（Instruction Decode）进行区分。实际上，M6800只有197种操作，6502只有151种操作；8080有244种操作；8085有246种操作，故对这些微处理机而言，其操作码只占一个字节。但Z80微处理机共有693种操作，所以Z80除了占有全部256种操作码外，其中有4种表示还有其它的操作码字节，即有双字节操作码。

不少指令，在操作码后面还有一个或两个字节给出操作数，故一般来说，微处理机指令是不定长的，从一字节长至四字节长。

三、指令的表示方式

因为计算机只认得二进制码，所以计算机中的指令无论是在存贮器中还是在微处理机的寄存器中，都只是依次排列的一组二进制数字。例如在Z80微处理机中，“累加器A的内容减1”这条指令的二进制编码为00111101。为阅读和书写方便常将二进制数码写成十六进制或八进制的形式。上述指令就可写成十六进制3DH（其中H表示十六进制）或八进制075Q（其中Q表示八进制）。这就是指令的机器码（Machine Code）。用机器码形式表示指令又叫机器语言（Machine Language）。由于微处理机的字长往往是八位或八的整数倍，所以微处理机中更多地使用十六进制形式。

在本书的其余部分里除非作特殊规定，否则在书写指令机器码时都采用十六进制表示法。

虽然用十六进制表示二进制的机器码带来了很大的方便，并且与二进制表示法相比还有不易写错的优点，但仍然有一个很大的缺点：即无法以写下来的数字看出一条指令的含义。

例如指令机器码53H送入M6800的指令寄存器，就会使累加器B所有各位均取反。而这同一机器码若被送入8080、8085或Z80的指令寄存器，则会使寄存器D中的内容被另一寄存器E中的内容所取代。从机器码“53H”中是看不出这些操作的含义的。

针对机器码不好记忆，难以理解，容易出错的问题，人们就用一些助记符(Mnemonic)——通常是指令功能的英文词的缩写来表示指令。

例对于数的传送操作，如将寄存器B中的内容传送至累加器A，在M6800中用助记符TBA（Transfer B To A）；在8080和8085中用助记符MOV A,B(Move B TO A)，而在Z80中助记符则为LD A,B(Load B TO A)。这样，每条指令有了明显的特征，易于理解和记忆，也不易出错。

这种用助记符取代机器码进行编程的程序设计语言即是符号语言（Symbolic Language)。但是，如前所述，计算机只能识别、理解和执行机器语言，而对人们所易于理解和掌握的符号语言反倒一无所知。所以，用符号语言编制的源程序还必须通过一个汇编程序翻译成机器语言的目标程序，故符号语言又叫汇编语言（Assembly Language）。

第二节　通用汇编指令系统

从二进制码的机器语言发展到采用助记符的汇编语言，这无疑是一大进步。随着微处理机的迅猛发展和广泛应用，人们对助记符也提出了新的要求。

一、按机型而异的助记符

汇编语言是一种基本上还保留机器语言某些特点的中间语言，这种语言在编写程序时所使用的语句个数和该微处理器的指令大致相等，因此非常有利于尽力发挥该微处理器的功能。故在微型机的应用中，由于考虑到充分利用机器的特点，尽量降低程序翻译的软、硬件开销，尽量提高目标程序的质量等因素，一般都选择汇编语言进行编程。然而，这种汇编语言通常只是为某一种具体的计算机设计的，所以编写的程序只能在对应的一种机器上运行，不同的机器使用的助记符形式也往往因生产厂家不同而大相径庭。

如将寄存器B的内容加入累加器A这一操作，不同机型的助记符及机器码分别是：

机　型	助记符	机器码
M6800	ABA	1BH
8080，8085	ADD　B	80H

Z80　　ADD A,B　　80H

显而易见，即使是功能相同的操作，但因助记符指令按机型而异，故不能互相兼容，给软件的移植带来很大的困难。

此外，这些助记符不适于直观地、清晰地辩识和记忆。在各种先进机型微处理器不断涌现的今天，我们不得不面临机种繁杂，助记符指令系统各异的困难局面，人们非但很难迅速理解掌握，且极易混淆出错。这对学习汇编语言程序设计，理解和利用在别种机器上已形成的编程经验，以及设计不同机种混合的多微机系统等带来了很大困难。

二、通用助记符

针对按机型而异的助记符的弊病，“微处理机的通用助记符系统”[注]提出了一种更方便地实现微机软件共享，更有利于汇编语言程序设计教学以及混合多微机系统的开发和应用的通用汇编系统（General Assembly System，简称GAS）及其实现方法。

GAS包含四个方面的内容，即通用助记符（GM）的概念和定义，通用助记符指令系统（General Mnemonic Instruction Set，简称GMIS）；通用汇编语言（General Assembly Language，简称GAL）；通用汇编程序（General Assembler，简称GA）和通用反汇编程序（General Disassembler，简称GDA）。

通用助记符指令系统实际上就是若干不同微机的指令系统的功能性并集。因此，每一具体机种的指令系统以子集形式存在于其中（原理上，通用助记符系统可任意扩充以至包含全部市售的微机指令系统）。用此系统的助记符编写的程序可经过通用汇编程序转移成若干微处理器的机器码，从而在这些机器上运行。

这里介绍的GAS包含M6800，6502，8080，8085和Z80五种机器的指令系统，但由于这几种机器相当普遍，几乎占据了整个八位微处理器的应用领域，所以这样的GAS实际上是“通用的”。当然，随着有更强生命力的新机器的出现，GAS本身还应当进一步扩展。

1.通用汇编系统的特点

通用汇编系统具有下述几个重要特点：

（1）通用助记符采用一般ASCII码键盘所具备的标准字符，它以简洁而直观的形式代表各种指令，使指令的操作对象，寻址方式及其功能一目了然，易于掌握，便于记忆。

（2）统一了多种机器共有操作的助记符形式，而各机的一些独特的指令操作则丰富了系统的总的功能。

（3）构造了一种通用汇编程序和通用反汇编程序，能实现程序语言形式的各种灵活转换，从而便于研究、比较不同机器的性能。

不难看出，利用通用汇编系统的这些优点，我们就能很方便地提高系统性能：

① 提高一般程序员经验和编程技术的利用率。按机型而异的助记符，使我们无法方便地把针对某一机种的已有程序改为另一机种的汇编语言程序，任何一种改变都得重编、重调程序。这在软件费用相对于硬件费用直线上升的今天，简直是无法容忍的。使用了GAS，则就可借助于GDA把某一机器上现有的目的程序转化为GAL编写的源程序，再通过GA汇编为

[注] 白英彩，程　杰　“Microprocessor Oriented General Mnemonics System”〔Proceedings of TENCON，IEEE Dec 6-8，1982 Hong knog〕

能直接在另一种机器上运行的目的程序。这不但可省去对程序的改写、重调等步骤，而且能使一个只懂一种汇编语言的用户，却可利用好多种机器上的汇编语言程序，并使其转化到他所能理解的那种汇编语言形式。

② 提高微机软件教学功效。迄今为止的微机教科书通常都向学生介绍几种典型微机的指令系统。由于指令条目繁多，其助记符各自不同，很易使初学者望而生畏，或在注意了指令系统的细节特征时却忽视了它们的根本特性和原理。有了GAS，则就大大方便了形象化教学，而且在学习指令系统时，还可很容易地实现个别的分析比较和总体的概括综合。在学习汇编语言程序设计时，则可彻底摆脱对某一类机种的依附，不受特定指令系统的羁绊，有利于对程序设计理论的学习和编程技巧的掌握。

③ 提高程序的可读性和降低编程时的出错率。通用助记符编写的程序虽不及高级语言，但却比较一般助记符更接近于人表述问题的方式，所以编制程序文本时可减轻注释工作量，且易于阅读理解，还可避免许多因使用指令量大而引起的技术性细小错误。

④ 提高不同机种的混合多微机系统的功能和降低开销。多微机系统对实时响应速度、信息传输控制、内部机间耦合都有较高的要求，面向过程的高级算法语言是力所不及的。然而若系统内各机只能接受自己那一种汇编语言，则系统就不能很好地作为整体来发挥作用，而且还得为每个机器配上一套汇编程序，浪费了系统资源，也增加了开销，在系统中首先要掌握和利用的应当是各机最重要的和最普遍有效的特征。GAS则恰到好处地解决了这个问题，只需一套通用汇编模块，当接受到系统给予的不同汇编数据基后，其就能把输入的程序汇编成相应的机器代码段，送到这一机器上运行。显然，系统将因而大大增强功能和提高效率。

2. 通用助记符（GM）的基本符号

GAS为达到其通用地“助记”功能而定义下列基本符号：

（1）寄存器。r或s表示任一八位寄存器或累加器。rr或ss表示任一十六位寄存器或八位寄存器对。

（2）赋值号。“:”表示将其右端指定的或计算得到的结果送入其左端指出的存贮位置。“:”可与各种运算符号组合使用。“:=”表示简单赋值；而“:=:”则表示符号左端和右端规定的寄存器或存贮单元的内容相互交换。

（3）数据和地址。用数字“12”表示单字节十六进制数据，“1234”则表示双字节十六进制数据。在数字前写上“M”则表示此数据为地址值。对某字节存贮单元中的位序，则在单元（地址）号后加上“.S”（表示第S位），S可取0,1,……7中的任一数值。

（4）状态标志。共有五种状态标志符号，即N（负数），Z（零），P（奇/偶），K（进位），V（溢出）。

（5）操作助记符。

操作名称	助记符
加	+
减	-
带进位加	++
带借位减	--
与	&
或	U