Z80

汇编语言程序设计

科学技术文献出版社重庆分社

Z 80

汇编语言程序设计

科学技术文献出版社重庆分社

Z80汇编语言程序设计

中国科学技术情报研究所重庆分所　编辑
科学技术文献出版社重庆分社　出版
重庆市市中区胜利路91号
四川省新华书店重庆发行所　发行
科学技术文献出版社重庆分社印刷厂　印刷

开本：787×1092毫米1/32　印张：26.50　字数：60万
1981年8月第一版　1981年8月第一次印刷
科技新书目：3—260　印数：11000册

书号：15176·499　定价：2.80元

说　明

本书译自美国Osborne & Associates出版公司1979年出版的«z80 Assembly language Programming»一书。作者是Lance A.Leventhal。全书共分十六章。第一、二章对汇编语言的格式和编写汇编程序的规则作了一般说明；第三章逐条解释z80汇编语言指令的功能，为正确使用和编写优良的汇编语言程序打下基础；第四章至十三章，以大量微型机的典型任务为例子，说明了编写汇编语言程序的方法；最后三章，叙述了如何把具体的微型机任务（如控制、数据处理、数字计算）变成为可执行的程序，从问题的定义开始，包括程序设计的一般方法，查错和测试，最后编成文件。

本书的特点是逻辑性强，语言简洁，层次分明，通俗易懂。列举的大量例子，都是编程序中常见的问题。熟读此书，基本上可以掌握z80机的汇编语言程序设计，同时对其它机种也有参考价值。许多章节的后面附有习题，有助于理解和巩固所学得的知识。

由于我们水平有限，译文中难免有这样那样的缺点和错误，敬请读者批评指正。

参加本书翻译的有：许茂元（第一章）、薛家政（第二章、第三章一部分）、董秉枢（第三章）、林云梯（第四、五、六、七章）、黄藻华（第八、九、十二章）、杨世乐（第十一章）、沈志同(第十、十四章)、李天复（第十三章）、饶生忠（第十五、十六章)。全书由薛家政校对，由黄藻华、林云梯编辑。

1981.3.20.

目　录

第 一 章
汇编语言程序设计引言

本书讲述汇编语言的程序设计。作者假定读者曾阅读过《微型计算机入门：第1卷——基本概念》一书，特别是其中的第6和7章（译者注：此书的中译本即将出版）。本书并不讨论计算机、微型计算机、寻址方法或指令系统的一般特点，欲了解这些方面的知识，请参阅《微型计算机入门：第1卷》一书。

指令的意义

微型计算机的指令系统乃是一组二进制的输入，这些输入在一个指令周期内产生人们规定的动作。指令系统对微处理机的作用与函数表对逻辑器件（例如门、加法器或移位寄存器等等）的作用很相似。当然，微处理机按指令输入而执行的动作要比组合逻辑器件按它们的输入而执行的动作要复杂得多。

指令仅仅是一些二进制位的组合，它必须以数据的形式在适当的时候输入到微处理机中，以便将其解释为一条指令。例如，当z80微处理机在取指令的操作期间接收到8位二进制形式的输入10000000时，则表示：

"将寄存器B的内容加到累加器内容上"。同样，形式00111110的意思是：

"将程序存储器下一个字的内容装入累加器。"

就像任何其他计算机一样，微处理机只认识二进制形式的指令或数据，它并不认识字或八进制、十进制和十六进制的数。

计算机程序

程序是使计算机执行特定任务的一系列指令。

事实上，计算机程序不仅包括指令，它还包含微处理机完成指令规定任务所需的数据和存储器地址。显然，如果微处理机要作加法运算，就必须有要相加的二个数和存放结果的地方。计算机程序必须确定数据源和结果的存放地以及规定要进行的运算。

除非某条指令改变了执行次序或使计算机停机，否则所有计算机都按顺序执行各指令。例如，只要现行指令并未特别指示处理机要执行其他操作，处理机就按顺序从下一个存储器地址中取下一条指令。

每个程序最后都要翻译成为一组二进制的数。举例而言之，要使存储单元60_{16}和61_{16}的内容相加并把结果置入存储单元62_{16}，则z80的程序为：

```
0 0 1 1 1 0 1 0
0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 1 1 1
0 0 1 1 1 0 1 0
0 1 1 0 0 0 0 1
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 1 1 0 0 1 0
0 1 1 0 0 0 1 0
0 0 0 0 0 0 0 0
```

这是一个机器语言程序或目标程序。若将此程序送入以z80为基本组成部分的微型计算机的存储器中，则微型计算

机就能直接执行它。

程序设计的问题

要制作目标程序或二进制机器语言程序，会碰到许多困难。下面所列举的，乃是问题的一部分：

1)这种程序难于看懂和调试。你会把各个二进制数看成都是相同的，看了几小时后，尤为如此。

2)由于必须将每一位单个地送入计算机，所以程序的输入速度太慢。

3)程序并不用人们可读的格式来描述要计算机执行的任务。

4)程序太长，写起来很费力。

5)程序员经常会由于粗心而出错，这些错误很难发现。

例如，下面的加法目标程序有一位是错的，试检查之：

```
0 0 1 1 1 0 1 0
0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 1 1 1
0 1 1 1 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 1 1 0 0 1 0
0 1 1 0 0 0 1 0
0 0 0 0 0 0 0 0
```

虽然计算机容易处理二进制的数，但是人们却不易处理。对人来说，二进制程序太长、太费劲、容易混淆和没有意义。时间久了，程序员也许记住了一些二进制码，但却是事倍功半的事。

采用八进制或十六进制

用八进制或十六进制数而不用二进制数来写指令，则情

况会有所改善。在本书中作者将采用十六进制数，原因是它们较短，而且又是微处理机工业的标准。表1-1定义了十六进制数字以及与它们相对应的二进制数。现在二数相加的z80的程序变为：

```
3A
60
00
47
3A
61
00
80
32
62
00
```

至少十六进制数字写起来较短，检查起来也不会那么麻烦。

较易发现一列十六进制数字中的错误。此时有错误的十六进制形式的加法程序为：

```
3A
60
00
47
72
61
00
80
32
62
00
```

很容易看出其中的错误。

微处理机只懂得二进制的指令码。那么我们如何处理这种十六进制的程序呢?回答是,必须把十六进制数转换为二进

制数。这种变换是重复的、费力的工作。作这种变换的人会犯各种各样的小错误，例如看错了行、漏掉1位或者把位或数字的位置弄颠倒了。

但是，这种重复的、低级的工作用计算机来完成则可尽善尽美。计算机永不会疲劳或厌烦，也永不会因糊涂而出错。因此，人们希望写这样一种程序，它接受十六进制数并把它们转换成二进制数。人们为许多微处理机提供这种标准程序，并名之曰"十六进制装入程序。"

是否值得制作十六进制装入程序呢？如果你乐意用二进制数来写程序，并且准备把二进制形式的程序送入计算机中，那么你就不需要十六进制装入程序。

如果你选用十六进制装入程序，你就要为此付出代价。十六进制装入程序本身就是一种必须装入存储器中的程序。而且十六进制装入程序要占用存储器，而你却可能要给存储器派其他用场。

因此，一方面十六进制装入程序会增加费用和占用存储器，另方面又可节省程序员的时间，两者必须折衷。

十六进制装入程序是值得制作的，因为它的费用不高。

当然，十六进制装入程序并不能解决程序设计的每一个问题。用十六进制写的程序仍然难读或难解。例如，我们既不能将指令与数据或地址区分开来，也不能从程序清单中看出程序要做些什么。32、47或3A是什么意思呢？熟记写满代码的卡片，这种办法也不怎么好。而且不同的微处理机有完全不同的代码，并且程序还要求编制大量的文件。

指令码的助记符

显然可对程序设计作些改进，这就是对每条指令都给予一个名称，人们把指令码的名称谓之为"助记符"或记忆符。

表1-1　16 进 制 数 的 变 换 表

16进制数字	相应的 2 进制数	相应的10进制数
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
A	1 0 1 0	10
B	1 0 1 1	11
C	1 1 0 0	12
D	1 1 0 1	13
E	1 1 1 0	14
F	1 1 1 1	15

指令助记符能以某种方式描述指令的含义。

事实上，微处理机厂家自已也记不住16 进 制 代 码，但是，每个厂家都给微处理机指令系统提供一组助记符。你不一定要用厂家提供的助记符。这些助记符也并不是一点都不能更改的。但是，对于一定的微处理机来说，它们是标准的助记符，所以所有用户都是懂得的。你在手册、卡片、书籍、论文和程序中会碰到这些指令名称。选择指令的助记符时的问题是，并非所有的指令都有"一看就懂"的名称。某些指令的确有显见的名称（例如ADD、AND、OR）。另一些指令名称一看就知道是缩写（例如SUB是减法的缩写，XOR是"异"

的缩写)。还有一些指令名称则不能一看就知道其含义。例如，名为WMP、PCHL以及SOB的指令就属于这种情况。请读者猜猜这三条指令的含义究竟是什么。绝大多数厂家提供的名称大多是合理的，只有少数是不恰切的。但是，若用户自己对指令起名称，也不见得会比厂家起的名称好多少。

厂家除给指令命名外，常常也给CPU寄存器命名。正象指令名称一样，有的寄存器名称是一看就懂的，例如累加器的名称为A；而另一些名称则只是习惯叫法。作者也将只用厂家起的名称，以期促进命名标准化。

如果采用Zilog公司定义的标准Z80指令和寄存器名称，则Z80的加法程序为：

```
LD      A, (60H)
LD      B, A
LD      A, (61H)
ADD     A, B
LD      (62H), A
```

此程序远非一看就懂，但至少某些部分是容易了解的。ADD A，B比80要容易懂得多；LD也一看就知是，将数据装入寄存器或存储单元。这种程序就是汇编语言程序。

汇编程序

我们如何将汇编语言程序放入计算机中呢？必须将其翻译为十六进制或二进制的数。你可以用手工方法翻译汇编语言程序，此时将指令逐条地进行翻译。这就是所谓的手工汇编。加法程序的指令码的手工汇编如下表(见第8页)所示。

就象将十进制数转换成2进制数一样，手工汇编也是一件机械的作业，既乏味、重复，而且会犯大量的小错误。取错了行、颠倒了数字、遗漏了指令以及读错了代码等等，这

指 令 名 称		相应的16进制数
LD	A,(NN)	3A
LD	B,A	47
ADD	A,B	80
LD	(NN),A	32

些是可能犯的错误的一部分。由于绝大多数微处理机都有各种字长的指令，这一工作更复杂了。有些指令长度为1个字，而另一些指令长度却为2个或3个字。有些指令需要第2和第3个字的数据，而另一些指令却需要存储器地址、寄存器号码或其他什么。

我们也可以叫微型计算机来完成汇编这一机械的工作。微型计算机在翻译代码时从不会犯任何错误，它总是知道每条指令需要多少个字和要求怎样的格式。人们把完成这一作业的程序谓之为"汇编程序"。汇编程序是将用助记符书写的用户程序或"源"程序翻译为机器语言程序或"目标"程序，而后一程序微型计算机是可以执行的。汇编程序的输入是源程序，它的输出是目标程序。

我们在讨论16进制装入程序时所提到的折衷，在汇编程序的情况下更要加于考虑。与16进制装入程序相比，汇编程序的成本更高，占据的存储空间更大，需要更多的外围设备和更长的执行时间。用户可以（而且常常）写自已的装入程序，但几乎不考虑编写自已的汇编程序。

汇编程序有它们自已的规则，读者必须弄懂、遵守它。这些规则包括在适当的位置使用一定的记号（例如空格、逗号、分号或冒号），正确的拼写，适当的控制信息，也许还包括正确安放名称和数。这些规则一般并不很难，很快就可

掌握。

汇编程序的其他特性

最初，汇编程序只不过是把指令和寄存器的助记符名称翻译为与它们相应的二进制码。然而，现在绝大多数汇编程序都有下列特性：

1）允许用户给存储单元、输入和输出设备，甚至给指令序列起名。

2）把数据或地址由各种进制的数（例如十进制或十六进制数）变换为二进制数，并且把字符变换为它们的ASCII（美国信息交换标准码）或EBCDIC（扩充的二进制编码的十进制交换码）的二进制码。

3）作为汇编过程的一部分而进行某些运算。

4）告诉装入程序应把各程序部分或数据装入存储器的何处。

5）允许用户指定某存储区来暂存某些数据，并把固定数据放在程序存储区。

6）提供信息，指明在现行程序中包含程序库的标准程序和其它时间写的程序。

7）允许用户控制程序打印格式和指定输入输出设备。

当然，要具有上面这些特性，就要增加费用和占用存储器。一般的说微型计算机的汇编程序比大型计算机的汇编程序要简单得多。但是，汇编程序的规模一直有增加的趋势。你常常要对各汇编程序进行选择。重要的判据不是汇编程序有多少次要的特性，而是它是否便于用常规操作进行处理。

汇编语言的缺点

就象十六进制装入程序那样，汇编程序并不能解决程序设计的所有问题。问题之一是，微型计算机的指令系统与微

型计算机要执行的任务有很大的差别。计算机指令要做这样的事情：把二个寄存器的内容相加，把累加器的内容移1位，或把新的数值放进程序计数器中，等等。而另一方面，用户通常要计算机做那样的事情：检查模拟读数是否超过了阈值，寻找电传打字机发出的特殊命令并作出相应的反应，或者在适应的时间启动继电器，等等。汇编语言的程序员必须把这些任务翻译为一系列简单的计算机指令。这种翻译工作可能较难，并且很费时间。

其次，如果你要用汇编语言进行程序设计，你就必须详细了解你所用的那台微型计算机的性能。你必须了解微型计算机有什么寄存器和指令，确切了解指令是如何对各种寄存器作用的，计算机采用何种寻址法以及许多其他知识。这些知识都与微型计算机最后必须执行的任务无关。

此外，汇编语言程序是不能移植的。每台微型计算机都有它自已的汇编语言，它反映了其自身的体系结构。为Z80写的汇编语言程序就不能移植到Motorola公司的6800机、Fairchild公司的F8机或National Semiconductor公司的PACE机上。例如，为M6800写的加法程序将是：

```
LDAA    $60
ADDA    $61
STAA    $62
```

不能进行移植这一点不仅意味着，你不能把你的汇编语言程序用于其他微型计算机，而且也意味着，除了专门为你使用的微型计算机写的程序外，你不能使用任何其他程序。这对微型计算机来说是特别不方便的，因为这些装置是新产品，而且提供给它们的汇编语言程序不多。因此你经常要自力更生。如果你需要一个程序来完成特殊的任务，你不太可