DBUG

—8080解释调试程序—

郁振宇　陈　克　译

上海市计算技术研究所

DBUG: 8080解释调试程序
—对汇编语言程序进行输入、调试、存贮的程序

克里斯托弗 A·泰特斯
乔纳森 A·泰特斯 编著

E & L 仪器公司出版

目　　录

序　言

本书是 Blacksburg 专业进修丛书中第一本有关微型计算机程序的教科书。于期，你将会看到 DBUG 程序对于 8080 微型计算机软件的研制，与丛书中其它几本书中的程序对于数字硬件和微型计算机接口技术的研制是同样十分有用的。我们编写的 DBUG 为用户的程序设计提供了很大的方便。DBUG 能装入数据和修改数据，也能改变读/写存贮器中的程序步。在把程序装入存贮器后，DBUG 就可以控制步进执行程序，并记录一条特定指令执行后所产生的各种结果，这些结果是：8080 所有的内部寄存器，由 HL 寄存器对寻址的存贮单元、栈指针及最后进栈的二个字节。同硬件单步不一样——8080 硬件的单步通常只步进一个指令周期，而 DBUG 则步进一条完整的指令，不管需要多少个指令周期。DBUG 也可以使用电传打字机来读纸带和穿纸带。于是，对于能正确运行的程序，就可以把它们以穿孔纸带的形式保留下来。当然也可以把存贮器中的数值保留在纸带上。对 DBUG 来说，被穿孔的内容是数据还是程序它是不管的。在今后需要的时候，可以使用 DBUG 能从阅读器上读纸带的特性，把这些纸带内容装入 8080 微型计算机。

虽然 DBUG 最初是为使用电传打字机的微型计算机系统编制的，然而只要对其输入/输出(I/O)程序作些小的改动，就能与音频盒带或可见终端相匹配。这两种设备通常被用于微型计算机系统中。定位包括在 DBUG 内的 I/O 程序照理是不会有困难的。事实上，在 Tychon 的系统中，我们在大多数的时间里都在使用由 DBUG 控制的 CRT，因为 CRT 有很高的数据传输速度。只有在需要的时候，才打开电传打字机来穿纸带或读纸带。

我们起初编写的 DBUG，主要的要求是把该程序留驻在读/写存贮器、EPROM、PROM 或 ROM 器件的 1024 (1K) 个存贮单元内。当时，1702 A 这个 EPROM 片子是十分昂贵的，所以我们认为选用 4 片 EPROM 对一个初级（基本的）系统来说是合理的。现在，已能方便地买到 1702 A 这种片子，而新的 2708(1K×8)EPROM 片子也能从零售商那里买到，且它们的价格已大大下降。因此，我们正在设计 DBUG Ⅱ，它是一个功能更广泛的调试工具。由于把 DBUG 限制在 1K 的存贮单元内，所以用户将会发现 DBUG 缺少在其他调试软件包中所具有的一些功能。尽管如此，我们对 DBUG 的性能还是十分满意的。

我们在附录中列出了两份 DBUG 的程序清单。一份为 8 进制代码形式，并含有 8 进制的 I/O 子程序。另一份为 16 进制代码形式，并做了一些必要的改动，以便能将 I/O 子程序按照 16 进制定位。如果用户是使用 8 进制代码的程序员，那你就得把 8 进制版本的 DBUG 送入读/写存贮器或用 8 进制的 I/O 子程序来请求 DBUG 的 2 进制纸带文本。如果用户是使用 16 进制代码的程序员，那你就要把 16 进制版本的 DBUG 送入读/写存贮器或用 16 进制的 I/O 子程序来请求 DBUG 的 2 进制纸带文本。

一个涉及到微型计算机程序空间分配的最大问题是，有些用户的计算机系统同其他系统有些不一样。有些用户有许多外部设备，例如磁盘，高速纸带阅读机和穿孔机，CRT 和可编程的 PROM，而许多其他用户则只有电传打字机。另一个可能不同的是：存贮器的字节数，存贮器的类型 (R/W 或 PROM) 以及存贮器地址。DBUG 是按照留驻在存贮器的第

5 块上编址的。DBUG 留驻的1K存贮器具有 020000 到 023377(hex 1000 到 13FF)的地址。同时还需要一个小容量的读/写存贮器，它的地址为 003300 到 003376(hex 03CO 到 03FF)和 000070 到 000072 (hex 0038 到 003A)。在这些地址上用户可能已经有了读/写存贮器或 PROM。但不幸的是，没有比较容易的办法来解决这个"存贮分配"问题。然而，用户可以把 DBUG 装入存贮器的其它块中(即：100000 到 103377 或 4000 到 43FF)，然后逐条检查 DBUG 并改变所有 JMP，CALL，LXI，SHLD，LHLD，STA 和 LDA 指令的高地址字节。在把 DBUG 装入读/写存贮器并对需要修改的地址进行修改后，用户可能还得改变 I/O 子程序，来与目前用户系统中的 I/O 设备相匹配。

另一个要发生的问题是如何把纸带初次装入存贮器。我们已经提供了一个引导装入程序(该程序在本书中——译注)，用它可以把 DBUG 纸带装入存贮器。像 DBUG 本身一样，该引导装入程序也可能不得不"传送"到存贮器的其它块中。用户此时就必须对该引导装入程序中的JMP 和 CALL 指令的高地址和低地址进行修改，否则就无法输入 DBUG。用户可能还得改变引导程序中的输入程序(称为 READ)，这取决于用来读纸带的外部设备。注意，READ 子程序是在引导装入程序的末尾，所以可以十分方便地按照用户系统的要求来改变或扩展该READ 子程序。

尽管 DBUG 程序和文件编制是 Blacksburg 专业进修丛书中第一本介绍软件的书，但我们已经计划在今后再增加有用的子程序，数值计算包和编辑/汇编软件。我们的目的不仅是提供好的软件——它能被微型计算机使用，而且提供操作命令和资料——它将使可以自学的程序清单读起来更容易。

我们目前的兴趣是在 8080 系统的汇编语言程序设计上。如果用户有特殊兴趣，多半是有一个用这种语言编制的有趣的通用程序，我们对这种程序很感兴趣。如果用户对 DBUG 有什么问题或改进的建议，请把你的想法告诉我们。

Tychon 公司致力于在数字电子，微型计算机硬件和软件领域中培养科学家、工程师和电子/计算机的业余爱好者。我们的月刊出现在 American Laboratory (International Scientific Communications, Inc, 8080 Kings Highway, Fairfield, CT 06430)，泰文出版的 Semiconductor Electronics Journal(Science, Engineering and Education Co., Ltd., Bangkok, Thailand) 和德文出版的 Elektroniker (Burgdorf, Switzerland)。我们的月刊也出现在Electronic News(IPC Business Press Pty Ltd., Australia)，Ham Radio (Greenville, NH)，Computer Design (Concord, MA)，和 Radio-Electronics(New York City, NY)，我们的丛书现在正在被翻译成意大利文、德文、法文、日文和泰文。如果用户对学习基本的数字电子学感兴趣的话，Logic and Memory Experiments Using TTL Integrated Circuits(Books 1 and 2)这两本书将向你介绍这个课题，并给予你试验环境(学习验证实验 (Student-proven Experiments))。Interfacing and Scientific Data Communications Experiments 这本书是特地为对使用异步通信技术感兴趣的工程师、科学家和业余爱好者而写的。它所包含的实验将允许你使用异步通信技术把仪器接到计算机或其他 UART 基本设备。现在已有三本有关 8080 微处理器/微型计算机的书。The 8080A Bugbook：Microcomputer Interfacing and Programming and a 2-volume set of Introductory Experiments Interfacing。这些自学课本描述了基本的接口硬件和软件的设计，以及它们是怎样解决这些问题的；用软件代替硬件；及

许多其他有兴趣的题目。

我们也有一套应用丛书。目前这套丛书共有5本书。555 Timer Sourcebook, With Experiments; Design of Actire Filters, With Experiments; Design of Op-Amp Circuits, With Experiments; Design of Phase-Locked Loop Circuits, With Experiments 和本书。

在 Tychon 的学术活动中，讨论班起着很大的作用。我们的讨论班已被介绍到各种组织，包括Analog Devices, Lnc. Xerox Corporation, Defense and Readiness Command, Illinois Central College and the Pratt Institute。讨论班也已经被传到英国，荷兰，澳大利亚和瑞士。如果用户对参加我们的讨论班感兴趣的话，请同我们联系。

Tychon 也印制 8080 8进制代码卡片和 16 进制代码卡片。代码卡片是计算尺，它就像编制和调试 8080 软件的工具一样。代码卡片包含所有 8080 助忆符和助忆符所对应的 8 进制或16进制操作码。所有的指令被染色编码，以指示出某条指令执行后将受到影响的标志位。袖珍代码卡片也有一张包括 128 个字符的 ASCII 代码表以及 8080 状态字和寄存器对的图样。

克里斯托弗 A. 泰特斯
乔纳森 A. 泰特斯

程序设计简介

随着微型计算机数量的日益增多，许多用户正在认识到，要有效地运行和使用微型计算机，就需要了解硬件和软件。硬件包括微型计算机和所有连接到微型计算机系统的外部电气设备。微型计算机是以微处理器集成电路片子（典型的片子为8080，6800，6502或F8）、微型计算机的读/写存贮器以及只读存贮器为基础而构成的。外部设备可以是D/A转换器和A/D转换器，实时时钟，软盘，CRT，电传打字机或高速纸带阅读/穿孔机。用户如果想要得到一个微型计算机系统的话，不难找到某些制造商，他们能供应组装一个微型计算机系统所需要的所有片子。而微型计算机的软件或程序，则是另外一回事了。因为用户对微型计算机的应用很可能是专用的，没有现成的软件能解决它。所以，用户将不得不自己编写程序或软件，以便在不同的深度解决实际问题。

用户可以用机器语言编制微型计算机的程序。机器语言是由所有存贮在微型计算机存贮器中的逻辑1和逻辑0所组成的。这种程序设计的方法通常只需要少量的外部设备，不必使用软盘或高速行式打印机。然而，用机器语言进行程序设计有许多缺点。因为记住2进制数是非常困难的。你宁愿说"我喜欢三打桔子"还是宁愿说"我喜欢100100（2进制）个桔子呢"？如果用户有一个一千步的机器语言程序，那就不得不将一个个2进制数存贮到微型计算机的存贮器中去。这不仅在输入所有这些数据时很容易出错，而且如果程序中有任何错误的话，这些错误也将很难找到。当然用户可以使用8进制数、10进制数或16进制数的机器语言来进行程序设计。这会使得记住许多微型计算机的指令编码变得容易一些，但它还是一种困难的程序设计方法。用机器语言进行程序设计仅有的优点是，它只需要很小的存贮器。如果程序有二百条指令，就只需要二百个存贮单元（这是对单字节的指令而言，对双字节和3字节的指令则另当别论——译注）。对于短的程序，由手工产生机器语言程序比使用符号编辑程序和汇编程序产生机器语言程序要快。

从使用机器语言进行程序设计前进一步是使用汇编语言进行程序设计。汇编语言程序设计是使用助忆符来完成的。助忆符由简单的3个到7个字母符号组成，例如LXIH，STA，IN和MOVAB。程序员记住这些助忆符是很容易的，因为助忆符同能被微型计算机执行的操作码有直接的关系。一个助忆符的子集或一张助忆符的表就称为源程序。当用户在纸上编写程序时，就会使用助忆符来生成一个源程序。用户也将充当编辑程序的角色，因为你要添加、删除和改变助忆符，直到程序正确为止。程序设计技巧成熟的用户最终将会感到，在纸上修改源程序是非常吃力的。现在，用户就可以使用能在8080微型计算机上执行的符号编辑程序来代替这些工作。这个程序使得用户可以对助忆符组进行校正操作，然后存贮到微型计算机的存贮器中。这同用户在纸上移动、修改、扦入和删除助忆符是一样的。然而，微型计算机是不可能执行一个符号程序的。符号必须被转换为它们的二进制等价值，并要在程序被执行以前，存贮到微型计算机的存贮器中。

存贮在微型计算机存贮器中的一个00100001代码同汇编语言程序中的LXIH（助忆符）

是等价的。那么，从助忆符 LXIH 到 00100001 的转换是怎样进行的呢？如果用户有一个程序在纸上，就能在表格(附录 C)中查到 LXIH 指令，并找到它的 2 进制等价值。用户也可以使用像 Tychon8080 的 8 进制代码卡片或 16 进制代码卡片这样的程序设计辅助工具来确定 LXIH 指令的 2 进制 (8 进制或 16 进制) 代码。这个过程是非常枯燥无味和耗费时间的，而且容易出错。正如我们在生成一个源程序时可以利用符号编辑程序一样，在将助忆符转换为 2 进制值、8 进制值或 16 进制值的过程中，我们可以利用汇编程序。汇编程序的工作是对LXIH 指令和所有其他 8080 指令确定其相应的 2 进制值、8 进制值或 16 进制值。当汇编程序确定了与 LXIH 指令相应的数值后，该数值必须被保留在某个地方。汇编程序将决定数值 00100001 是保留在纸带上，音频盒带上，还是在软盘上。此时，这个汇编后的程序文本就可以称为目标程序或 2 进制程序(一系列的"1"和"0")，它可以装入到微型计算机的存贮器中去执行。十分有趣的是，汇编程序汇编源程序的方法同用户将源程序写在纸上时所应做的一切是一样的。手工汇编源程序时，用户是在一张表格中查找某个助忆符，再找到相应的 2 进制值、8 进制值或 16 进制值，然后把它们写在纸上(一种存贮方法)。当源程序通过微型计算机进行汇编时，微型计算机为了在汇编程序代码表格和源程序二者之间的一次符合而不断搜索表格中的值。如果符合产生，微型计算机就会从表格中取出对应的 2 进制值、8 进制值或 16 进制值。如果符合没有产生，汇编程序通常就打印出错误的某种类型。

使用汇编程序有二个基本优点。第一个是速度。汇编程序对一个特定的助忆符确定其对应的 2 进制、8 进制或 16 进制值仅花费 1 至 2 个毫秒。而对用笔和纸的程序员来说，这个过程最少也得用 2 至 3 秒钟。汇编程序可以完成成千上万的这些"翻译"而不会出一个差错。这种方法当然比用笔和纸的方法要好得多。

用户在进行较大的汇编语言程序设计时，将会出现重复。电传打字机子程序可能同其他一些由程序转向程序的子程序一样，这些子程序可以是完成数据块传送的子程序或者是搜索数据块的子程序。如果用户有一个助忆符或一条命令代表着 1 百到 2 百条汇编语言程序的指令，那将是很好的。而这恰好是 BASIC，FORTRAN 和 PL/1 允许这样做的。

在 BASIC 语言的程序中，用户可能会有一个像 PRINT"THIS IS A TEST"这样的程序声明(statemen)。当程序被最后执行时，PRINT 这个字将引起执行一段汇编语言子程序，该子程序在CRT上或在电传打字机上打印出 THIS IS A TEST 这个信息。程序员不必关心这是怎样发生的，也不必关心在 8080 内部哪一个寄存器对使用是有效的。

一般地，大多数程序(语言或命令——译注)可以被分为解释程序和编译程序。解释型的程序一次解释执行程序中的一行。大多数对 8080 微型计算机可用的 BASIC 程序是解释程序。编译程序编译或翻译输入的程序并产生该程序的目标格式或 2 进制格式。FORTRAN 编译程序编译 FORTRAN 程序清单并把新的程序文本保留在磁带或一叠卡片上。以后，这叠卡片或磁带就可以装入计算机中去执行，而不需要 FORTRAN 编译程序出现。对 8080 微型计算机来说，在运行一个 BASIC 程序时，BASIC 解释程序必须同时在存贮器中。

我们想要说明的这点是：DBUG 是一个机器语言解释程序。在打入一行数字和命令后，它们就能被解释执行。假如 DBUG 是一个编译程序，用户就得打完所有的命令，才能执行一条命令(DBUG 的命令——译注)，然后，把已打入的数据和命令统统穿孔输出在新的纸带上。大多数的 8080 微型计算机的符号编辑程序都是解释程序，这是因为用户每次只打

入一个命令或一行文本。但是，汇编程序都是编释程序，这是因为汇编后的新的目标程序或二进制程序是由汇编程序产生的。最后，我们想说明的是，BASIC 和 FORTRAN 都可以是解释程序或编译程序。然而，BASIC 程序通常都是解释程序，而 FORTRAN 程序通常都是编译程序。记住，这些程序都是在微型计算机中实际执行的程序，而不是用户输入的 BASIC 或 FORTRAN 的源程序。

DBUG 引言

无论对初学的还是老练的程序设计人员来说，DBUG 程序都不失为一个强有力的程序设计和调试工具。有了 DBUG，短小的程序可以通过电传打字机键盘或纸带阅读机，方便地装入到微型计算机中。也可以查看以 8080 为基础的微型计算机中所有的 65536 个存贮单元的内容，也可以打入一个新值，并存贮到读/写存贮器中去。

一旦一个程序或几个程序和几个子程序装入到微型计算机中后，用户就可以对某一程序设置一个启动地址。这就不再受 RESET 的限制了，因为 RESET 键总是强制微型计算机从 000 000 号地址开始执行程序。用户也可以对留驻在读/写存贮器中的程序设置一个断点，这样程序就能全速执行，直到碰到断点。一旦碰到断点，寄存器A，B，C，D，E，H 和 L 中的内容、标志寄存器中的内容、由 HL 寄贮器对寻址的存贮单元中的内容、栈指针及栈中最后 2 个字节的内容全被打印出来。从这点可以看出：用户不但可以全速连续执行程序，单步执行程序，也可以全速、单步交替执行程序。

在单步操作方式中，用户可以一次执行一条单字节，双字节或 3 字节的指令，并记录下该条指令执行后所有寄存器中的结果，由 HL 寄存器对寻址的存贮单元的内容，栈指针和栈中最后 2 个字节。用单步方式执行了程序中的许多步后，用户可以再次以全速继续执行程序，或者把存贮在读/写存贮器中的程序加以修改①。

用户也可以把存放在存贮器任意区域中的程序穿在纸带上。穿孔子程序并不规定信息长度，所以可以一次穿一个任意长度的连续文件。使用 DBUG 可以在任何时刻把穿孔纸带读回到微型计算机中。

至此完成了对 DBUG 的简短介绍，现在让我们来讨论实际的 DBUG 命令以及一些程序设计的例子。

DBUG 命令

在最初启动 DBUG 时，DBUG 总是用一个问号“?”及回车换行来作为应答，使你能以命令方式来使用 DBUG。下列字符为合法的 DBUG 命令：

P　穿纸带(注意：必须指定好地址信息)

R　读纸带

K　删除断点

S　单步执行，执行一条指令并打印出寄存器内容和其他有关内容

①注意：并不是所有的指令都可以用单步方式来执行的。这些不能用单步执行的指令是：转移、调用、返回和重启动指令。然而，这并不会削弱 DBUG 的能力，在下面的一些程序例子中将说明这些指令以及它们是怎样被执行的。

X　继续执行程序

020 000　接受一个 8 进制或 16 进制的(取决于用户所用的 DBUG 版本)存贮地址。这
或　里给出的例子是 002 000(8 进制)或 02 00(16 进制)

0200

如果想把一条指令或一个数据字节送入存贮器，必须用 2 个 8 进制字节或 2 个 16 进制字节来指定一个 16 位的地址。因此，在命令方式中，也可以打入地址信息。例如，可以打入 002 000 或 02 00，这里地址高字节是 002(16 进制为 02)，地址低字节是 000(16 进制为 00)。在每一个 8 进制或 16 进制字节后面，微型计算机自动打印出一个空格。

要看一下 002 000 单元中的内容，就应打入：

002 000/

用户的输入(作为 DBUG 的一个用户)是以粗体字来印出的，以后所有的例子都将这样做。记住，微型计算机是用空格来分隔 8 进制字节或 16 进制字节的，为了响应打入的斜线符/，微型计算机立即打印出用户已设好地址的存贮单元中的内容：

002 000/125

为了修改这个单元中的内容，应再打入一个 8 进制或 16 进制字节：

002 000/125　**231**

现在可以打入一个回车，换行或另外一个数字。假如你把数字打错了——真正想打入的是 232，而不是 231，那就继续打入 8 进制或 16 进制数值，直到认为正确为止。因此，用 8 进制代码格式时，有：

002 000/125　**231　241　232**

用 16 进制代码格式时，有：

02 00/55　**99　A1　9A**

现在应打入一个回车或换行符。一个回车符将会引起微型计算机输出回车换行，并引起程序返回到命令方式，以致用户可以指定一个新的地址，或执行其他操作。在正常情况下，返回到命令方式时，微型计算机是不打印问号“?”的。只有当用户企图打入非法命令或非法的 8 进制或 16 进制数时，才会打印出一个问号“?”来。例如用户打入了一个 Q，12P 或 AG，微型计算机就会打印出一个问号“?”来。

现在，假定用户有一个 11 步的程序要打入到存贮器中去。就必须为这个程序指定第一个存贮地址，但是否必须总是先打一个地址然后打一个数据呢？回答是否定的。一旦微型计算机有了第一个地址后，总是递增 1，并由打入的换行符来打印下一个地址值。因此，为了装入这个 11 步程序，应该打入下列信息(我们仍给出 8 进制和 16 进制两种清单)：

程序示例 1

	8 进制	助忆符	16 进制
002000/125	**076　LF**	MVIA	**0200**/55　**3E　LF**
002001/025	**001　LF**	001	0201/15　**01　LF**
002002/321	**006　LF**	MVIB	0202/D1　**06　LF**
002003/056	**002　LF**	002	0203/2E　**02　LF**
002004/077	**200　LF**	ADDB	0204/3F　**80　LF**
002005/123	**200　LF**	ADDB	0205/53　**80　LF**

	8 进制		助忆符	16 进制		
002006/276	**117**	LF	MOVCA	0206/BE	**4F**	LF
002007/301	**127**	LF	MOVDA	0207/C1	**57**	LF
002010/247	**025**	LF	DCRD	0208/A7	**15**	LF
002011/105	**025**	LF	DCRD	0209/45	**15**	LF
002012/201	**166**	CR	HLT	020A/81	**76**	CR

在166(HLT)指令后面打入回车符时，要注意到DBUG将返回到命令方式。

为了检查刚才打入的程序是否正确，可以指定一个开始地址，然后打入一个斜线符"/"。于是这个地址单元中的内容就被打印输出。如果打入一个换行符，则下一个存贮地址以及该地址中的内容将被打印输出。打换行符这个步骤可以连续进行，直到整个程序被列出为止。假定该程序已被打入并已存贮在读/写存贮器中，用户将得到下列清单：

8 进制		16 进制	
002000/076	LF	**0200**/3E	LF
002001/001	LF	0201/01	LF
002002/006	LF	0202/06	LF
002003/002	LF	0203/02	LF
002004/200	LF	0204/80	LF
002005/200	LF	0205/80	LF
002006/117	LF	0206/4F	LF
002007/127	LF	0207/57	LF
002010/025	LF	0208/15	LF
002011/025	LF	0209/15	LF
002012/166	CR	020A/76	CR

无需说明，一直打换行符肯定是吃力的，所以我们已经在DBUG中设置了列表的性能。用户可以简单地打一个开始地址再打入一个字母L来使用这个性能：

002000 L 或 **0200** L

然后，微型计算机开始列出从指定地址开始的存贮器地址和它们的内容。由于没有指定一个终止地址，微型计算机不知道什么时候应该停止列表。这时，用户所要做的是敲一下电传打字机上任何一个键，但不能是SHIFT或CTRL。此后，微型计算机将结束打印并返回到命令方式。

用户已经知道了如何通过电传打字机或适当的ASCII终端来输入一个程序，如何改变一个存贮单元中的内容以及如何列出8进制或16进制代码的程序清单。DBUG也有把数据送到纸带上或从纸带接受数据的能力。在命令方式中，P命令和R命令就是派这个用处的。

纸带的阅读和穿孔

为了读纸带，纸带的格式必须是DBUG规定的格式(见附录B)。把纸带头(纸带头是由一列穿在纸带一侧的孔所组成的)放在电传打字机的纸带阅读器上，把选择开关置在RUN

的位置，然后在电传打字机上打一个 R。当读完纸带，且将数据存贮到存贮器后，DBUG 就返回到命令方式。

为了穿纸带，用户必须知道要被穿孔的存贮区域的开始地址和结束地址。地址信息作为 4 个 8 进制或 16 进制字节被输入，前2个字节表示开始地址，后 2 个字节表示结束地址。在这个地址区域中的数据都将被穿孔输出，包括 2 个指定地址单元中的内容。

示意如下：

```
P                    P
002000 （开始地址） 0200
002013 （结束地址） 020B
```

16 进制代码在右边，8 进制代码在左边。对应于被穿孔的存贮字节的前面和后面，在纸带上按照 8 进制字节 200 或 16 进制字节 80 穿好带头和带尾。在读纸带时，使用 DBUG R 命令的装入程序是忽略带头和带尾的。

执行一个程序

一旦一个程序装入读/写存贮器，要执行它是很容易的。由于同时在存贮器中可能有许多程序和子程序，所以必须有一个方法来指定某个程序的启动地址。为了做到这点，用户必须指定一个 2 个字节的地址，这就像在穿纸带和在存贮器单元中扦入数据时一样，然后打入 G(GO)命令。为了从存贮地址 002000 开始执行程序，应该打入：

```
002000 G 或 0200 G
```

只要一打入 G 命令，DBUG 就将寄存器 A，B，C，D，E，H 和 L 清为全 0，然后微型计算机从 002000 地址开始执行程序。

假如有一个示例 2 的程序在存贮器中：

程序示例 2

```
003000/041    LXIH    0300/21
003001/303    303     0301/C3
003002/001    001     0302/01
003003/174    MOVAH   0303/7C
003004/323    OUT     0304/D3
003005/000    000     0305/00
003006/175    MOVAL   0306/7D
003007/323    OUT     0307/D3
003010/002    002     0308/02
003011/043    INXH    0309/23
003012/303    JMP     030A/C3
003013/003    003     030B/03
003014/003    003     030C/03
```

用户应打入

003000 G 或 0300 G

来开始程序的执行。当程序正在运行时，H 和 L 寄存器中的内容将被输出到 000 输出口和 002 输出口，现在，如果用户忘了这个程序的开始地址并错误地打入了：

003001 G 或 0301 G

自然地，DBUG 将从存贮地址 003001 开始执行指令。然而，DBUG 程序不知道这是一个错误的启动地址，也不知道在这个地址中的内容是 303(C3)；303(C3)对 LXIH 指令来说是一个最小有效(the least significant)的 8 位数据字节。于是 DBUG 程序就把这个数据字节解释为一条指令！因为 8 进制代码 303 是一条无条件转移指令的操作吗，所以在 303 后面的 2 个字节就被 DBUG 错误地解释为转移指令的地址高低字节。8080 把 003002 存贮单元中的内容解释为低地址字节，把 003003 存贮单元中的内容解释为高地址字节。于是，8080 就执行了一条无条件转移到 174001 的指令。显然，这并不是用户原来想要 8080 做的事。所以，记住所有程序的启动地址是很重要的。如果把程序穿在纸带上，用户将会发现把启动地址直接穿在纸带上是很有益处的。

断点的使用

断点是 DBUG 最有用的一个特性。在调试一个程序时，通常要求把一个程序运行到某一个预先指定的存贮地址。然后检查 8080 的内部寄存器，检查程序中可能要修改的存贮单元——各种指令和数据，并根据 8080 内部寄存器的状态或特定的存贮单元的内容进行修改。为了完成修改步骤，DBUG 充当了系统监控程序的角色。

用户可以决定停止或“打断”正常程序流程的地址，DBUG 在该断点地址单元中扦入一条指令。当这条指令在程序中得到执行时，控制就被自动地转到了 DBUG。当这种情况发生时，DBUG 立即保护所有 8080 内部寄存器中的内容，方法是把它们存贮在 DBUG 自己的栈中。然后打印出被打断的地址，标志字节的内容，所有寄存器的内容，由 HL 寄存器对寻址的存贮单元的内容，栈指针以及用户栈中最后进入的 2 个字节的内容。此时，用户可以改变断点值，请求 DBUG 继续执行程序。这时，DBUG 将所有8080内部寄存器的内容恢复到执行打断指令以前的值，然后继续执行程序，直到再次碰到一个断点或直到程序正常结束。用户也可以使用断点来步进执行程序中的一条或多条指令，来看看各种指令执行后，8080 内部寄存器中的结果。

断点可以设置在程序中的任何一点，但是必须设置在多字节指令的第一个字节，断点不可以设置在转移指令或调用指令的第一个字节，以及所有其他不能在单步方式中执行的指令的第一个字节。那么，怎样在程序中设置断点呢？这已经在 DBUG 命令这一节中说明过了，应打入：

002005 B 或 0205 B (B 为扦入断点的命令—译注)

这就把断点(我们从这里打断正常程序的执行)设置在第二条 ADDB 指令的位置。如果接着打入

002000 G 或 0200 G

电传打字机将打印出下列信息(假定是 8 进制代码输出)：

002005 B

```
002000  G
002005
SZ 1 P 2  A    B   C   D   E   H   L   M    SP      CS
000 0 011 0  005  002 000 000 000 000 000  ××× ××××××  ××××××
```

注意，原来在断点地址中的指令已经被执行了(该程序在第 7 页上一译注)。微型计算机二次把 B 寄存器的内容(002)加到 A 寄存器的内容(001)上去，结果为 005。B 寄存器仍保持着 002，其他寄存器为 000。数字 1 和数字 2 在标志字中表示 2 个进位标志位。其中数字 2 表示从 D_7 位溢出的进位标志，它是很有意义的一位；数字 1 表示 4 位状态或 D_3 和 D_4 之间的辅助进位标志。在单步执行程序时或每次建立断点时，寄存器的标记(the register "labels")并不是每次都打印出来的，而只是每五次才打印一次。这里没有讲到 M，SP 和 CS 的内容，等一会儿，我们将在后面讨论。

单步执行

用户通过打入 S (STEP) 命令来请求 DBUG 执行下一条指令——这条指令是 MOVCA，然后打印出寄存器，标志等来显示这条指令执行后的结果。注意，在 MOVCA 指令执行后，只有 C 寄存器的内容改变过，以前是 000，现在是 005。如果再次打入 S 命令，微型计算机将执行下一条指令——一条 MOVDA 指令。这时只有 D 寄存器中的内容会被改变。现在，可以打入 X 命令来继续执行程序。打入 X 命令后，DBUG 连续执行程序，直到执行到 HLT 指令。现在用户大概明白了，我们没有办法步进执行程序中的一条 HLT 指令。如果不相信的话，可以用程序示例 1 的程序来试一下。为了节省时间，试着通过打入

```
002010  B   或   0208  B
002000  G        0200  G
```

来启动程序而不用打入

```
002000  B   或   0200  B
002000  G        0200  G
```

来启动程序，这将快速到达 002010 地址或 0208 地址 (16 进制)。再由单步继续执行程序。

假定示例 3 的程序已存贮在读/写存贮器中。

程序示例 3

```
002000/076    MVIA    0200/3E
002001/002    002     0201/02
002002/006    MVIB    0202/06
002003/003    003     0203/03
002004/007    RLC     0204/07
002005/200    ADDB    0205/80
002006/200    ADDB    0206/80
002007/303    JMP     0207/C3
002010/004    004     0208/04
```

020011/002　　002　　0209/02

用户可以把断点放在程序的开头，然后单步执行它。但是假如用户已经打入了下列信息：

002015　B　或　020D　B

002000　G　　0200　G

这时，用户还能在电传打字机键盘上打入任何东西吗？微型计算机还在运行吗？

回答是：微型计算机正在执行用户在程序中建立的"LOOP"，这样断点就永远不会到达。因为断点设置在循环的外边。然而，断点却还是存在的。

前面我们提到过，可以用 DBUG 的断点特性来调试存贮在读/写存贮器中的程序，但不能调试存贮在只读存贮器中的程序。这是由于 DBUG 要在所设置的断点地址上将程序（该程序存贮在读/写存贮器中）中的一条指令操作码删去，而代之于一条单字节指令（RST7—译注）。假如一直没有到达先前设置好的断点，那就请检查一下 002015 或 020D (hex) 单元中的内容，来看看这条单字节指令究竟是什么。为了检查这个单元的内容，重新启动 DBUG，并打入下列信息：

002015/　　或　020D/

用户将看到，DBUG 所使用的打断指令是 377(hex FF)，它在 8080 指令集中是一条重启动 7(RST7)指令。

当 RST7 指令执行时，它将把程序运行引向存贮地址 000070(hex 00 38)。在这个单元中，有一条转移指令，它将程序控制转回到 DBUG 的断点软件中去。因此，在使用 DBUG 的时候，请用户务必不要在自己的程序中执行 RST7 指令，也不要产生一个使用 377(hex FF)的中断指令的中断。

为了说明 K 命令的功能，请做下面的事。把 002015 (hex 02 0D)单元中的内容改为除 377(hex FF)外的任何一个数，它由下列操作来完成：

002015/377　×××　CR　或　020D/FF　××　CR

这里×××(hex ××)是用户自己选择的字节。现在请打入：

002015　B　　020D　B

002015　/　CR　或　020D　/　CR

若问现在把什么打入到 002015 单元中去了？那当然是断点指令 377。现在再打入：

K　　K

002015/×××　　或　020D/××

用户应该看到自己所选择的字节 ×××(hex ××)。通过揿一下 K 键，就删除了原来设置在 002015 (hex 02 0D)单元中的断点，并扦入(恢复——译注)原先在该地址中的字节。

如果这一节给出的实例程序还不在存贮器中，请重新打入，然后打入下列信息：

002015　B　或　020D　B

002000　G　　0200　G

由于这时 8080 已不再执行 DBUG 程序了，所以用户不可能在键盘上用 K 键来删除断点。然而，重新启动 DBUG 后，用户仍旧可以用 K 键来删除断点。

DBUG 所不能做的

我们已经提到过，用户不应当企图单步执行一条暂停(halt)指令。其他途径也会使用户的程序或 DBUG 程序“无影无踪”(“bomb out”)，故 DBUG 对多字节指令有所规定，即断点应该放在任何多字节指令的第一个存贮地址。

研究一下示例 4 的程序。

程序示例 4

```
002000/076    MVIA     0200/3E
002001/250    250      0201/A8
002002/041    LXIH     0202/21
002003/003    003      0203/03
002004/200    200      0204/80
002005/167    MOVMA    0205/77
002006/166    HLT      0206/76
```

用户不应当企图打入

```
002001  B  或  0201  B
002000  G      0200  G
```

来单步执行这个程序，如果这样做，就永远不会碰到断点。为什么会这样呢？因为你已经用 377(hex FF)取代了 002001 单元中的“指令”，而这条“指令”是 MVIA 指令的数据。于是 8080 把 377 这个值(hex FF) 装入到 A 寄存器中去，而不是将 250 这个值 (hex A8)装入。然后程序继续进行，直到执行到 HLT 指令。正是由于这一点，用户必须小心所有立即型的算术/逻辑指令，LXT，MVI，STA，LPA，LHLD，SHLD，JMP 和 CALL 等所有多字节指令。用户可以单步执行除转移、调用、返回和重启动指令外的所有指令，而且 DBUG 知道正在执行的指令是一条单字节，双字节指令还是一条 3 字节的指令。

用上述的示例程序将 002001 单元中的内容改回到 250，然后单步执行它，

```
002000  B  或  0200  B
002000  G      0200  G
```

如果打入 S 命令，就能看到 8080 仅仅执行了在

```
002000      或  0200
002002          0202
002005          0205
```

存贮单元中的指令。

转移指令、调用指令、返回指令和重启动指令

根据程序示例 1 和程序示例 4 中 HLT 指令的使用，用户可能理解了为什么用 DBUG 不能单步执行任何转移、调用、返回和重启动指令。如果单步执行这些指令，DBUG 将丢失对程序执行的控制。让我们来看一下当用户企图这样做时会发生些什么。

我们使用程序示例5来作为例子。

程序示例5

```
002000/061    LXISP    0200/31
002001/300    300      0201/C0
002002/003    003      0202/03
002003/303    JMP      0203/C3
002004/200    200      0204/80
002005/002    002      0205/02
```

用

```
002000 B 或 0200 B
002000 G    0200 G
```

来执行这段程序。

```
002000
SZ  1  P  2  A  B  C  D  E  H  L  M  SP     CS
010 0  011 0  000 000 000 000 000 000 000 303 003300 001100
S 002003?
```

注意一下LXISP指令的执行。下一条指令就是一条无条件转移指令。若试图单步执行该条指令，用户将会看到一个问号"?"，这是DBUG告诉用户转移指令不能被步进的信号。如果用户试图单步执行任何其它的条件转移指令(3X2，X=0～7)、条件调用指令(3X4，X=0～7)、条件返回指令(3X0，X=0～7)、调用指令(315)、返回指令(311)或重启动指令(3X7，X=0～7)，都将会看到同样的一个问号"?"。注意，所有的重启动指令都被包括在内，甚至377，因为这是DBUG用来设置断点的重启动指令。

那么究竟怎样来调试含有转移、调用和返回指令的程序呢？研究下面的程序。

程序示例6

```
002000/021  LXID   0200/11
002001/012  012    0201/0A
002002/123  123    0202/53
002003/303  JMP    0203/C3
002004/020  020    0204/10
002005/002  002    0205/02

002020/041  LXIH   0210/21
002021/252  252    0211/AA
002022/125  125    0212/55
002023/166  HLT    0213/76
```

用

```
002000 B 或 0200 B
002000 G    0200 G
```

来开始执行这个程序，

```
002000
010  0  011  0  000  000  000  123  012  000  000  303  003170  000106
```

注意每个寄存器的内容。用户应观察到，只有 D 和 E 寄存器不是 0。现在打入

```
002020  B  或  0210  B
X              X
```

```
002020
010  0  011  0  000  000  000  123  012  125  252  377  003170  000106
```

通过打入这些信息，就把断点移到了 002020 (hex 02 10) 这个地址，然后 DBUG 从以前的断点——002000(hex 02 00)继续执行程序。注意，执行了 002020 (hex 02 10)单元中的 LXIH 指令后，仅仅对 H 和 L 寄存器赋了新值。尽管用户执行了转移指令，但 DBUG 仍控制着程序的执行！列出从 002000 开始，直到 HLT 指令为止的程序。

```
002000  L  或  0200  L
```

377 指令在哪里？回答是，DBUG 自动地删去了断点(RST7 指令)，并在每次断点指令执行后把原来的指令送回用户的程序。记住，DBUG 已经执行了在断点地址内原来的指令。我们对转移指令使用的这种技巧同样也能用于调用指令和返回指令。断点被传送到软 件 步，8080也就将控制传送到软件步。

栈 操 作

在执行断点指令时，DBUG 暂时使用了用户栈的 2 级(4 个字节)。DBUG 然后保护用户的栈指针并为自己建立一个栈。所以用户可以在程序中对自己的栈进行任何操作，而不必担心 DBUG 对它的影响。记住。如果用户想在程序中使用调用、返回、进栈、退 栈 等指令，就必须在这些指令的执行前用一条 LXISP 指令来建立栈指针。而 把 一 条 LXISP 指令作为任何程序的第一条指令是个好的经验。然而，如果用户不建立栈指针，则 DBUG 将为自己建立一个栈指针。因此，用户现在之所以能看到“SP”和“CS” 的内容，完全是由于使用了 DBUG 栈的缘故。但用户决不能以此假设可以用这个栈来进行自己的栈操作。用户想要使用栈，就必须建立自己的栈指针。

现在让我们来看一下栈操作的例子。输入下列程序(到目前为止，用户在做这些 事 情时不应该感到有困难)：

程序示例 7

```
003000/061    LXISP    0300/31
003001/100    100      0301/40
003002/003    003      0302/03
003003/041    LXIH     0303/21
003004/200    200      0304/80
003005/100    100      0305/40
003006/341    POPH     0306/E1
003007/321    POPD     0307/D1
003010/325    PUSHD    0308/D5
```

```
003011/345    PUSHH    0309/E5
003012/063    INXSP    030A/33
003013/063    INXSP    030B/33
003014/073    DCXSP    030C/3B
003015/041    LXIH     030D/21
003016/000    000      030E/00
003017/000    000      030F/00
003020/071    DADSP    0310/39

003021/000    NOP      0311/00
003022/000    NOP      0312/00
003023/000    NOP      0313/00

003100/001             0340/01
003101/002             0341/02
003102/003             0342/03
003103/004             0343/04
003104/005             0344/05
003105/006             0345/06
003106/007             0346/07
```

从003000地址开始，单步执行这个程序，并显示一些栈指针和栈内容的有趣变化，它恰好是该程序所期望的：

```
003  000  B
003  000  G
003  000
SZ  1  P  2  A   B   C   D   E   H   L   M    SP    CS
010  0  011  0  000  000  000  000  000  000  000  303  003  100  002  001

S  003  003
010  0  011  0  000  000  000  000  000  100  200  115  003  100  002  001

S  003  006
010  0  011  0  000  000  000  000  000  002  001  015  003  102  004  003

S  003  007
010  0  011  0  000  000  000  004  003  002  001  015  003  104  006  005

S  003  010
010  0  011  0  000  000  000  004  003  002  001  015  003  102  004  003

S  003  011
```

```
SZ  1  P  2  A  B  C  D  E  H  L  M    SP    CS
010 0  011 0  000 000 000 004 003 002 001 015 003 100 002 001

S 003 012
010 0  011 0  000 000 000 004 003 002 001 015 003 101 003 002

S 003 013
010 0  011 0  000 000 000 004 003 002 001 015 003 102 004 003

S 003 014
010 0  011 0  000 000 000 004 003 002 001 015 003 101 003 000

S 003 015
010 0  011 0  000 000 000 004 003 000 000 303 003 101 003 000

S 003 020
SZ  1  P  2  A  B  C  D  E  H  L  M    SP    CS
010 0  011 0  000 000 000 004 003 003 101 000 003 101 003 000

S 003 021
010 0  011 0  000 000 000 004 003 003 101 000 003 101 003 000

S 003 022
010 0  011 0  000 000 000 004 003 003 101 000 003 101 003 000

S 003 023
010 0  011 0  000 000 000 004 003 003 101 000 003 101 003 000
```

对于调用指令，如果用户用单步执行转移指令的技巧来处理，就能在栈中看到返回地址。研究下列程序：

程序示例 8

```
003000/061    LXISP    0300/31
003001/100    100      0301/40
003002/003    003      0302/03
003003/076    MVIA     0303/3E
003004/240    240      0304/A0
003005/315    CALL     0305/CD
003006/020    TEST     0306/10
003007/003    0        0307/03
003010/000    NOP      0308/00
003011/000    NOP      0309/00
```

```
003020/006  TEST,   MVIB      0310/06
003021/001          001       0311/01
003022/200          ADDB      0312/80
003023/200          ADOB      0313/80
003024/117          MOVCA     0314/4F
003025/311          RET       0315/C9
```

单步执行这个程序并观察栈指针和栈内容。

```
003  000  B
003  000  G
003  000
SZ   1   P   2   A   B   C   D   E   H   L   M     SP    CS
010  0   011 0   000 000 000 000 000 000 000 303   003   100 157 374

S  003  003
010  0   011 0   240 000 000 000 000 000 000 303   003   100 157 374

003  020  B
X
003  020
010  0   011 0   240 001 000 000 000 000 000 303   003   076 003 010

S  003  022
100  0   001 0   241 001 000 000 000 000 000 303   003   076 003 010

S  003  023
100  0   001 0   242 001 000 000 000 000 000 303   003   076 003 010
S  003  024

SZ   1   P   2   A   B   C   D   E   H   L   M     SP    CS
100  0   001 0   242 001 242 000 000 000 000 303   003   076 003 010

003  010  B
X
003  010
100  0   001 0   242 001 242 000 000 000 000 303   003   100 157 374

S  003  011
100  0   001 0   242 001 242 000 000 000 000 303   003   100 157 374
```

现在，用户已经看到了所有 DBUG 所能执行的命令和操作。同时也可能开始知道：为什么为了完成似乎是很简单的事，要这样长的程序。前面没有提到过的最后一个性质是，DBUG 将允许用户单步执行输入/输出(I/O)指令，这将使得接口电路的调试变得简

单一些。

怎样启动读/写存贮器中的DBUG

DBUG 使用的存贮单元：

020	000—023	377	DBUG 程序(读/写存贮器或 PROM)
000	070—000	072	RST7 转回到 DBUG
003	300—003	377	栈和暂存器
		或	
10	00—13	FF	DBUG 程序(读/写存贮器或 PROM)
00	38—00	3A	RST 7 转回到 DBUG
03	C0—03	FF	栈和暂存器

引导 DBUG

为了把 DBUG 从纸带装入读/写存贮器，用户在存贮器中就必须有一个引导装入程序来将 DBUG 程序装入存贮器。下面 2 个程序就是引导装入程序的清单。这 2 个程序在功能上是完全一样的，只不过一个是 8 进制，另一个是 16 进制(我们翻译了 16 进制版本程序中的注解部分，没有翻译 8 进制版本程序，因为它们是完全一样的——译注)。把下列 2 个程序中任何一个装入存贮器：

```
TYCHON EDITOR-ASSEMBLER V-2 (8 进制版本—译注)
    /TYCHON BOOTSTRAP LOADER WITH CHECKSUM
    /V3, 21APR77, TYCHON, INC, BLACKSBURG, VA
    /IF THIS BOOTSTRAP IS TO BE MOVED TO A DIFFERENT SECTION
    /OF MEMORY, MAKE SURE THE ADDRESS FOR THE STACK
    /REFLECTS WHERE YOU HAVE R-W MEMORY.
    /ALSO, YOU WILL HAVE TO CHANGE THE HI ADDRESS BYTES
     OF
    /ALL THE JUMP AND CALL INSTRUCTIONS. ASSUMING YOU KEEP
    /THE LO ADDRESSES THE SAME, YOU HAVE TO CHANGE THE HI
    /ADDRESSES AT LO ADDRESSES OF 007, 014, 021, 024, 030,
    /034, 037, 044, 047, 057, 064, 067, 072, 075, 105, 113,
    /AND 126, YOU MAY ALSO HAVE TO CHANGE THE I/O INSTRUC
    /TIONS TO REFLECT THE DEVICE ADDRESSES USED WITH
    /YOUR SYSTEM.
          *000  000
000  000  061  BOOT, LXISP  /SET THE SP
000  001  350         350
000  002  000         000
000  003  026         MVID   /INITIALIZE THE CHECKSUM
```

```
000  004  000          000
000  005  315 LDRIN, CALL   /READ A CHARACTER FROM THE TAPE
000  006  116          READ
000  007  000          0
000  010  376          CPI     /IS IT THE LEADER(200O)?
000  611  200          200
000  012  312          JZ      /YES,KEEP READING UNTIL WE
000  013  005          LDRIN   /PASS ALL THE LEADER
000  014  000          0
000  015  376 CHKFRM,CPI   /NO,NOT LEADER IS IT THE ADDRESS
                             FLAG?
000  016  100          100
000  017  302          JNZ     /NO,SEE IF IT WAS THE CHECKSUM
                             FLAG (300)
000  020  040          NOTA
000  021  000          0
000  022  315          CALL    /YES,IT WAS THE ADDRESS FLAG
                             (A 100)
000  023  073          ADDCHK  /GET THE NEXT 2 FRAMES, WHICH
                             ARE
000  024  000          0       /THE HIGH ADDRESS
000  025  147          MOVHA
000  026  315          CALL    /THEN GET THE 2 FRAMES WHICH ARE
000  027  073          ADDCHK  /THE LO ADDRESS
000  030  000          0
000  031  157          MOVLA
000  032  315 NXTIN, CALL    /NOW GET ANOTHER FRAME
000  033  116          READ    /AND PROCESS IT
000  034  000          0
000  035  303          JMP
000  036  015          CHKFRM
000  037  000          0
000  040  376  NOTA, CPI      /WAS NOT A 100 FOR AN ADDRESS
000  041  300          300     /IS IT THE CHECKSUM FLAG(A 300)?
000  042  302          JNZ
000  043  055          NOTSUM  /NO,THEN TREAT IT AS DATA
000  044  000          0
000  045  315          CALL    /IT WAS THE FLAG,GET THE
000  046  103          BYTE    /2 FOLLOWING FRAMES,BUT DON'T
```

```
000  047  000          0       /ADD THE CHECKSUM TO THEM.
000  050  272          CMPD    /COMPARE THE CALCULATED VERSUS
                               /ACTUAL
000  051  312  READOK, JZ      /IF THEY ARE EQUAL WE JUMP
000  052  000          DBUG
000  053  020          0
000  054  166          HLT     /THE CHECKSUMS WERE NOT EQUAL,
                               HALT ! ! !
000  055  315  NOTSUM,CALL
000  056  065          ADD1    /TREAT THE CURRENT FRAME AND
000  057  000          0       /NEXT FRAME AS DATA
000  060  167          MOVMA   /THEN SAVE IT IN MEMORY
000  061  043          INXH    /INCREMENT THE STORAGE POINTER
000  062  303          JMP     /AND GET ANOTHER FRAME
000  063  032          NXTIN
000  064  000          0
000  065  315  ADD1, CALL     /GET 2 FRAMES, 1 BYTE
000  066  106          BYTE1
000  067  000          0
000  070  303          JMP
000  071  076          ADDCHK+3
000  072  000          0
000  073  315  ADDCHK,CALL
000  074  103          BYTE
000  075  000          0
000  076  365          PUSHPSW  /SAVE "A"
000  077  202          ADDD     /ADD THE CHECKSUM
000  100  127          MOVDA    /AND SAVE IT BACK IN "D"
000  101  361          POPPSW   /GET THE DATA MORD BACK
000  102  311          RET
000  103  315  BYTE, CALL     /READ THE 2 MSB'S FROM THE TAPE
000  104  116          READ
000  105  000          0
000  106  017  BYTE1, RRC     /ROTATE THE BITS TO THE MSB'S
000  107  017          RRC
000  110  117          MOVCA   /SAVE THE TEMPORARY VALUE IN "C"
000  111  315          CALL
000  112  116          READ    /THEN READ THE 6 LSB'S
000  113  000          0
```

```
000  114  201          ADDC  /ADD THE MSB'S
000  115  311          RET   /AND RETURN WITH THE VALUE IN "A"
000  116  323  READ,   OUT   /PULSE THE READER CONTROL RELAY
000  117  021          021   /IF THE TTY IS SO EQUIPPED
000  120  333          IN    /GET THE RECEIVER'S STATUS
000  121  021          021
000  122  346          ANI   /MASK OUT ALL THE BITS EXCEPT
                             THE RECEIVERS
000  123  001          001
000  124  312          JZ
000  125  120          READ+2   /BIT IS STILL A 0,KEEP MAITING
000  126  000          0
000  127  333          IN       /DATA IS AVAILABLE, INPUT IT
000  130  020          020
000  131  311          RET      /RETURN WITH THE CHARACTER
                                 READ IN "A"
```

TYCHON EDITOR-ASSEMBLER V-2 （16 进制版本——译注）

/TYCHON 带有校验和的引导装入程序第 3 卷，1977.4.21.TYCHON 公司，BLACKSBURG，VA。

/如果该引导装入程序被传送到存贮器中其他块时，用户就必须重新确定栈地址，以便同用户的读/写存贮器相匹配。同时，用户必须改变所有转移和调用指令的地址高字节。如果用户要想保持低地址不变，就必须改变低地址为 07，0C，11，14，18，1C，1F，24，27，2F，34，37，3A，3D，45，48，56 这些地址的高地址。以便同用户系统所使用的设备地址相匹配。

```
                    *00H  00H
00  00  31  BOOT,  LXISP    /建立栈指针
00  01  E8          E8H
00  02  00          00H
00  03  16          MVID     /校验和初始化
00  04  00          00H
00  05  CD LDRIN,   CALL     /从纸带上读一个字符
00  06  4E          READ
00  07  00          0
00  08  FE          CPI      /该字符是带头(80H)吗?
00  09  80          80H
00  0A  CA          JZ       /是，继续读，直到读完所有带头
00  0B  05          LDRIN
00  0C  00          0
```

```
00  0D  FE  CHKFRM, CPI    /否，不是带头，那是地址标志吗？
00  0E  40          40H
00  0F  C2          JNZ     /否，看看是否是校验和标志(C0H)
00  10  20          NOTA
00  11  00          0
00  12  CD          CALL    /是，是地址标志(40H)，取接下来的2个记录*，这
00  13  3B          ADDCHK   2个记录是地址的高字节
00  14  00          0
00  15  67          MOVHA
00  16  CD          CALL    /再取下面2个记录，这2个记录是地址的低字节
00  17  3B          ADDCHK
00  18  00          0
00  19  6F          MOVLA
00  1A  CD NXTIN, CALL      /现在取另外的记录并处理之
00  1B  4E          READ
00  1C  00          0
00  1D  C3          JMP
00  1E  0D          CHKFRM  (重复上述过程—译注)
00  1F  00          0
00  20  FE    NOTA, CPI     /不是地址标志40H，它是不是校验和标志(C0H)？
00  21  C0          C0H
00  22  C2          JNZ
00  23  2D          NOTSUM  /否，把它作为数据来看待
00  24  00          0
00  25  CD          CALL    /它是校验和标志，取接下来的2个记录，但不把
00  26  43          BYTE     它们加入校验和
00  27  00          0
00  28  BA          CMPD    /将累加器同实际值比较(累加器中为校验和)
02  29  CA READOK, JZ       /如果两者相等，转到DBUG
00  2A  00          DBUG
00  2B  10          0
00  2C  76          HLT     /如果两者不相等，停机！！！
00  2D  CD NOTSUM, CALL     /把本次记录和下一次记录作为数据看待
00  2E  35          ADD1
00  2F  00          0
00  30  77          MOVMA   /然后把它们存入存贮器
00  31  23          INXH    /存贮器指针增1
00  32  C3          JMP     /取另外一个记录
00  33  1A          NXTIN
```

```
00 34 00          0
00 35 CD ADD1, CALL     /取2个记录；1个字节
00 36 46          BYTE1
00 37 00          0
00 38 C3          JMP
00 39 3E          ADDCHK+3
00 3A 00          0
00 3B CD ADDCHK, CALL
00 3C 43          BYTE
00 3D 00          0
00 3E F5          PUSHPSW /保护A寄存器内容
00 3F 82          ADDD    /将A寄存器内容加入校验和
00 40 57          MOVDA   /把相加后的校验和送回D寄存器
00 41 F1          POPPSW  /取回原A寄存器的内容
00 42 C9          RET
00 43 CD BYTE, CALL     /从纸带上读字符的高2位(MSB'S——为字符的
00 44 4E          READ    最大有效位—译注)
00 45 00          0
00 46 0F BYTE1, RRC     /对高2位进行循环右移(移到真正的高2位上去
00 47 0F          RRC     ——译注)
00 48 4F          MOVCA   /把暂存器中的值保留在C寄存器中
00 49 CD          CALL    /然后读字符的低6位(LSB'S——为字符的最小
00 4A 4E          READ    有效位——译注)
00 4B 00          0
00 4C 81          ADDC    /将高2位同低6位相拼(形成一个完整的8位字
                          符——译注)
00 4D C9          RET     /带着A寄存器中的字符返回
00 4E D3 READ,OUT       /如果TTY准备好的话，就启动阅读控制继电器
00 4F 11          021
00 50 DB          IN      /取接收器的状态
00 51 11          021
00 52 E6          ANI     /除接收器状态这一位外，屏蔽所有其他位
00 53 01          001
00 54 CA          JZ
00 55 50          READ+2  /状态位为0，继续等待
00 56 00          0
00 57 DB          IN      /数据有效，输入该数据
00 58 10          020
00 59 C9          RET     /带着A寄存器中读入的字符返回
```

(Frame 为纸带上的一个记录，8 位字符由 2 个记录组成。前一个记录为字符的高 2 位，后一个记录为字符的低 6 位——译注。)

在引导装入程序装入存贮器以后，用户就可能要修改纸带阅读子程序，该子程序从符号地址READ开始。000116 单元中的一条输出指令将用来启动电传打字机的纸带阅读控制继电器。如果用户的电传打字机没有阅读控制继电器，就应该在 000116 和 000117 (hex 00 4E 和 00 4F) 单元内放 2 条 NOP 指令(操作码为 000 或 00)。

8 进制			16 进制	
存贮地址	指令	助忆符	存贮地址	指令
000116	323	0UT	004E	D3
000117	021	〈 ADDR 〉	004F	11

000120(hex 00 50) 单元中的输入指令用来输入 UART 的数据有效标志，该标志作为 8 位状态字中的一部分。如果数据有效标志为逻辑 1，则表示已经从电传打字机接收了一个字。用户可能要修改这条指令的设备地址码，这取决于用户系统对这个状态位输入口所指定的地址。这个设备地址在 000121 (hex 00 51) 单元中。

根据输入的状态位置，可以修改 ANI 指令中的立即数字节。这取决于用户系统的状态输入口中的哪一位被指定为 UART 的数据有效(DA)标志。可以使用的立即数字节有以下这些：

立即数字节		数据有效标志使用的位	
8 进制	16 进制		
200	80	D7	(最大有效位)
100	40	D6	
040	20	D5	
020	10	D4	
010	08	D3	
004	04	D2	
002	02	D1	
001	01	D0	(最小有效位)

000127(hex 00 57) 单元中的输入指令将 UART 的 8 位数据字输入到 8080A 寄存器中。000130(hex 00 58) 单元中的设备地址必须是 UART 的 8 位数据输入口的地址。

如果用户不准备使用电传打字机的纸带阅读器来读 DBUG 程序的纸带，那就得写一个新的 READ 子程序。不管这个子程序写得怎样，总得让 8080 带着从纸带读到 A 寄存器的 8 位字符退出这个子程序。而且在这个子程序中只能使用 A 和 B 寄存器。在这些修改完成后(如果需要修改的话)，就把纸带带头放入纸带阅读器。纸带头是一段纸带，在这一段上，纸带被连续沿着带的一侧穿孔。带尾是用同样的方法穿孔的，但带尾在纸带的末端。带头和带尾(LDR, TRLR)分别是通过将 8 进制数 200 或 16 进制数 80 穿在纸带上来生成的。在把纸带放入纸带阅读器并把阅读器选择开关置在 START 位置以后，就揿一下微型计算机的 RESET 开关。如果 DBUG 纸带已经正确装入，引导装入程序将自动启动 DBUG。然后，DBUG 将在电传打字机上打印出一个问号"?"，等待用户的命令。如果 DBUG 没有被正确装入，计算机将停机。如果这种情况发生，而用户确信引导装入程序是正确装入的，那

就再次把 DBUG 装入存贮器。

怎样启动 PROM 中的 DBUG

在 DBUG PROM 已经加到用户系统中，且接上电源以后，用户必须将下列 3 个指令字节放入微型计算机的读/写存贮器中。

8 进制			16 进制	
存贮地址	指令	助忆符	存贮地址	指令
000 000	303	JMP	00 00	C3
000 001	000	DBUG	00 01	00
000 002	020	0	00 02	10

送入这些指令后，撤一下 RESET 开关。这些指令就使微型计算机转到启动 DBUG（当然，系统中 DBUG PROM 的首地址必须是 020000—译注），此时 DBUG 将在电传打字机上打出一个问号“?”作为响应，然后等待用户命令。

DBUG 的修改

DBUG 使用2个输入口和 1 个输出口来同真实环境通讯。通常这些 I/O 口子连接着 UART 或 USART，使得 DBUG 能同 CRT 或电传打字机通讯。一个输入口用来输入数据，这个数据是由 UART(USART) 从 CRT 或电传打字机那里接收数据。另一个输入口被用来测试 UART 的(USART 的)接收器和发送器的状态标志。输出口被 8080 用来将信息传送到 UART (USART)，使得 CRT 或电传打字机能发送和接收信息。现将所有这些功能概述如下：

输入口 020 (10)　输入由 UART(USART) 接收的数据

输入口 021 (11)　输入 UART(USART) 的状态标志，这个状态标志通常称为状态字。

输出口 020 (10)　输出数据到 UART (USART)

8 位状态字输入口中有 2 位用来指示出：UART (USART) 什么时候可以发送一个字，UART (USART) 什么时候可以接收一个字。状态输入口中的最小有效位 (D0) 被用来查明 UART (USART) 什么时候已经接收了一个字符。如果 D0 位为逻辑 1，那么说明已经接收了一个字符。如果 D2 位为逻辑 1，就可以把另一个字符输出到 UART(USART)，然后由 UART (USART) 自动地将该字符传送到 CRT 或电传打字机。这些标志位的意义如下：

D0 位 = 逻辑 1　UART (USART)能发送一个字符

D2 位 = 逻辑 1　UART (USART) 已经接收了一个字符

在 DBUG 中，ANI 指令被用来测试状态位中的某一位(D0或D2)是逻辑1还是逻辑0。

下列存贮单元含有输入口的地址，或者含有用来查明一个状态位的 ANI 指令中的立即数字节。

存贮地址				内容
8 进制		16 进制		
021	214	11	8C	状态输入口地址
021	216	11	8E	查明一个字符是否被接收的状态字
021	223	11	93	用来输入接收字符的输入口地址
021	275	11	BD	阅读控制继电器启动输出口地址
021	320	11	D0	状态输入口地址
021	322	11	D2	查明一个字符是否被接收的状态字
021	327	11	D7	用来输入接收字符的输入口地址
021	340	11	E0	状态输入口地址
021	342	11	E2	查明字符是否已被发送的状态字
021	350	11	E8	用来发送一个字符的输出口地址

用户应该保证所有这些单元中包含真正的设备地址和状态码。如果用户系统需要使用状态字中的其他位，可以对用来测试标志位的 ANI 指令中的立即数字节加以修改。下列数据字节是可供使用的：

立即数字节		使用的状态位
8 进制	16 进制	
200	80	D7 （最大有效位）
100	40	D6
040	20	D5
020	10	D4
010	08	D3
004	04	D2
002	02	D1
001	01	D0 （最小有效位）

MITS/IMSAI 8080 用户（S—100 总线微型计算机）

如果用户有这样一个读/写存贮器，即该存贮器的地址从 000 000 到 003 377（hex 00 00 到 03 FF）和从 020 000 到 023 377（hex 10 00 到 13FF），那么对于装入和使用 DBUG 将是没有困难的。仅需修改输入/输出子程序。上一节已经描述了这些修改。如果用户的微型计算机接有一个 USART，则可以使用 DBUG 最初的 8 个单元，来存放对 USART 进行初始化的指令。我们将在下一节推荐一个打印程序。

英特尔 SBC 80/10，80/20 和 SDK 80 用户

我们已经在 DBUG 最初的 8 个存贮单元留下了空间，使用户能够加上对 USART 进行初始化的指令。这段程序将按照以下的形式出现*：

```
MVIA
```

* 对于有关状态位、标志等更为详细的信息，请参考 SBC—80/10，80/20 或 SDK—80 的用户手册，或英特尔 8251 可编程通讯接口等资料。

```
        MODE
        OUT
        CNCTL
        MVIA
        CMD
        OUT
        CNCTL
```

MODE = 8 位模式指令

CNCTL = USART 的控制寄存器地址

CMD = 8 位命令指令

用户必须根据自己的 8080 系统来决定真正的 MODE、CMD、和 CNCTL 的值。我们已经对电传打字机的纸带阅读子程序加过 NOP。为了启动电传打字机的阅读控制继电器，可以试着：

```
READ,   MVIA
        TTYADV
        OUT
        CNCTL
        PUSHB
        LXIB
        <LO>
        <HI>
LOOP,   DCXB
        MOVAB
        ORAC
        JNC
        LOOP
        O
        POPB
        MVIA
        CMD
        OUT
        CNCTL
        ……
```

TTYADV = 引带命令

CNCTL = USART 的控制寄存器地址

CMD = 停止阅读器引带命令

用户必须再次根据自己的 8080 系统决定 TTYADV，CNCTL 和 CMP 的值。用户也必须确定 LXIB 指令后面的 2 个数据字节的值，这个值用来向软件提供时间延迟。用户的 8080 微型计算机可能要比其它用户系统快些或慢些，所以延迟时间必须随着不同的

速度而调整(trimmed)。还得修改输入/输出子程序。这些修改已经在“DBUG 的修改”一节中描述过了。

E &L 仪器公司 MMD—1 用户

在 MMD—1/MI 存贮器接口模块上，I/O 子程序同 UART 的设备地址是相一致的。然而，用户必须对 DBUG 做 4 点修改，这是因为用户在最初的 256(512)个存贮单元中有一个 EPROM。在 DBUG 的开头，有一个称为 VECT 的符号地址，它指向 000070(hex 00 38)这个地址。由于这个地址已被定位在 KEX EPROM 内，所以用户对 VECT 只能使用003 060到003 063(hex 03 3 0 到 03 33)的地址，而不能使用 000 070 到 000 073(hex 00 38 到 00 3B)的地址。

首先修改020 014(10 0C)单元中的内容，将 070 改为 060(38 改为 30)。然后修改 020 015(10 0D)单元中的内容，将 000 改为 003(00 改为 03)。用户还必须修改供断点使用的重启动指令。即修改 020 314(10 CC)单元中的内容，将 377 改为 367 (FF 改为 F7)。再修改 022 016 (12 0E)单元中的内容，将 377 改为 367(FF 改为 F7)。

从 E &L 仪器公司可以得到一套含有 DBUG 的 4 片 1702A PROM。在配有 MMD-1/MI 存贮器接口模块的 MMD-1 微型计算机中，DBUG 就可以直接工作了。

对断点使用不同的重启动指令

如果用户有用于 Restart 7(RST7)的硬件，就必须在 3 个地方修改 DBUG 的向量(oct 377，hex FF)指令。首先修改 020 014(hex 10 0C)单元中的 LXID 指令的数据字节。这个数据值应该被改为 010，020，030，040，050 或 000(hex 08，10，18，20，28 或 30)，究竟改为哪一个，则根据用户想要使用的那一条重启动指令而定。

被使用的重启动指令	数据值改为(在 020 014(10 0C)地址中)
RST 1 = 317 (CF)	010 (08)
RST 2 = 327 (D7)	020 (10)
RST 3 = 337 (DF)	030 (18)
RST 4 = 347 (E7)	040 (20)
RST 5 = 357 (EF)	050 (28)
RST 6 = 367 (F7)	060 (30)

后 2 个修改包括修改用于断点的实际指令。把 020 314 (10 CC) 单元中的内容从 377 改为上面所示的相应的重启动指令。

例　题

在本节中，我们将举 3 个例子。在这些例子中，DBUG 被用来单步执行一个示例程序。要想在这些例子中使用所有 8080 的 244 条指令将是不可能的。但我们使用一些指令来演示一个特定的操作或 8080 能执行的一组操作。

如果用户已将 DBUG 装入了自己的 8080 微型计算机系统，就可以使用这些例子来使自己熟悉 DBUG 的命令和操作。如果用户不准备使用 DBUG，这些例子将向你提供一个评

价 DBUG 能力的机会。

第一个例子演示了 2 条 8080 移位指令执行后对进位标志和 A 寄存器内容的影响。第二个例子使用了数据传送指令，将数据在 8080 内部寄存器和读/写存贮器之间进行传送。第三个例子演示了 DBUG 怎样被用来单步执行一条调用指令。在这个特定的例子中观察一下栈指针和栈内容将是很有趣的。这些例子可以概括如下：

例 1， 演示了 2 条移位指令执行后对进位标志和 A 寄存器内容的影响。

例 2， 演示了几条数据传送指令。

例 3， 演示了如何单步执行一条调用指令。

例 题 一

演示 2 条 8080 移位指令

目的

这个例子的目的是演示 2 条8080移位指令执行后对A寄存器内容和进位标志的影响。

程序

8 进制			16 进制	
存贮地址	指令	助忆符	存贮地址	指令
003 000	076	MVIA	03 00	3E
003 001	004	<DATA>	03 01	04
003 002	017	RRC	03 02	0F
003 003	303	JMP	03 03	C3
003 004	002	<LO>	03 04	02
003 005	003	<HI>	03 05	03

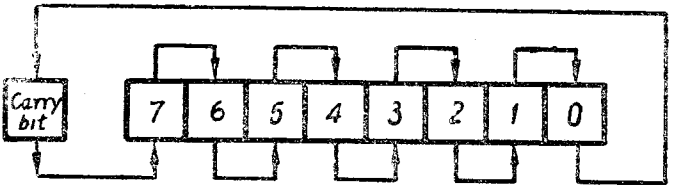
讨论

8080 可以执行 4 条不同的移位指令。2 条指令将 A 寄存器的内容向右移，另外 2 条指令将 A 寄存器的内容向左移。移位指令的数据流程图概述了这 4 条指令以及进位标志和 A 寄存器内容是怎样受到每条指令影响的。

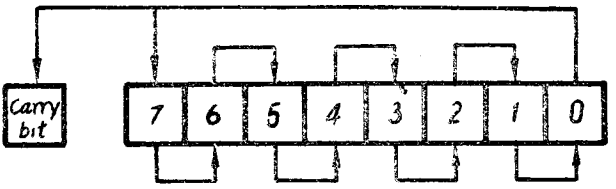
根据我们步进这个程序时所获得的电传打字机上的输出信息，就可以容易地理解 RRC 指令是如何影响 A 寄存器和进位标志的了。下面就是打印出来的信息（观察一下 A 寄存器内容和进位在位 2 的状态(一个逻辑 1 或一个逻辑 0)）：

```
003  000  B
003  000  G
003  000
SZ 1 P 2   A    B    C    D    E    H    L    M      SP      CS
01000110  004  000  000  000  000  000  000  303  004  000  200  000

S 003  002
01000110  002  000  000  000  000  000  000  303  004  000  200  000

003  002  B
```

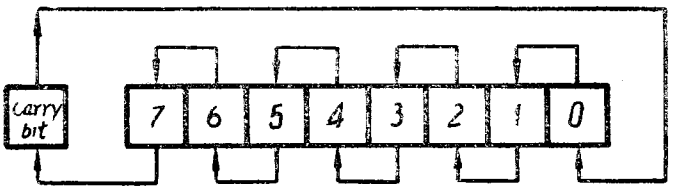
RAR:



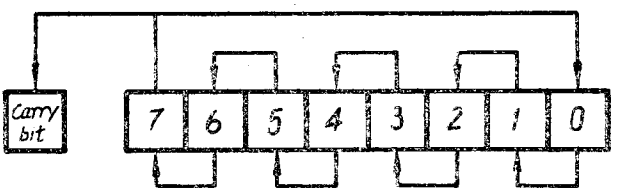
RRC:



RAL:



RLC:




```
×
003  002
01000110  001  000  000  000  000  000  000  303  004  000  200  000

003  002  B
×
003  002
01000111  200  000  000  000  000  000  000  303  004  000  200  000

003  002  B
×
003  002
01000110  100  000  000  000  000  000  000  303  004  000  200  000
```

003 000(hex 03 00)单元中的 MVIA 指令将 004(hex 04)这个数据值送入 A 寄存器。接着，003 002(hex 03 02)单元中 RRC 指令将 A 寄存器的内容向右移 1 位。第一次执行了 RRC 指令后，A 寄存器中的值是 002(hex 02)。由于 RRC 后面的一条指令是 JMP，所以断点被再次设置在 003 002(hex 03 02)地址上。为了测试其他信息，断点被反复设

置在那个地址上。

第二次执行了 RRC 指令后，A 寄存器中的值从 002 被移成了 001(hex从 02 移成 01)。第三次执行了 RRC 指令后，A 寄存器中 D0 位上的逻辑 1 被同时移入进位标志（进位在位 2)和 A 寄存器的最大有效位 D7。这就是根据上述 RRC 指令的数据流程图所做的。再次通过程序循环后，(即第四次执行了 RRC 指令后——译注)，D7 位上的逻辑 1 被移入 D6 位，进位被清为 0。由于 D6 位上为逻辑 1，A 寄存器中的 8 进制值就是 100(hex 40)。

现在让我们将 RRC 指令改为 RLC 指令，8 进制 007(hex 07)。

003 002 / 017 **007 CR** 或 **03 02** / 0F **07 CR**

除了 A 寄存器的逻辑 1 应移到左边外，我们希望看到同样的结果。正如从打印输出的信息中所能看到的，第一次执行了 RLC 指令后，A 寄存器的内容从 100 变为 200 (hex 从 40 变为 80)。第二次执行了 RLC 指令后，D7 位同时被左移入进位标志(进位在位 2)和 A 寄存器中的最小有效位 D0。第三次执行了 RLC 指令后，A 寄存器的内容从 001 变为 002(hex 从 01 变为 02)。

```
003  002  B
×
003  002
SZ 1 P 2   A    B    C    D    E    H    L    M      SP       CS
01000110  200  000  000  000  000  000  000  303  004  000  200  000

003  002  B
×
003  002
01000111  001  000  000  000  000  000  000  303  004  000  200  000

003  002  B
×
003  002
01000110  002  000  000  000  000  000  000  303  004  000  200  000
```

例 题 二

演示数据传送指令

目的

这个例子的目的是演示数据在 8080 内部寄存器和读/写存贮器之间的传送。

程序

8 进制			16 进制	
存贮地址	指令	助忆符	存贮地址	指令
003 020	041	LXIH	03 10	21
003 021	200	<LO>	03 11	80

003	022	003	<HI>	03	12	03
003	023	066	MVIA	03	13	36
003	024	123	<DATA>	03	14	53
003	025	043	INXH	03	15	23
003	026	064	INRM	03	16	34
003	027	176	MOVAM	03	17	7E
003	030	053	DCXH	03	18	2B
003	031	167	MOVMA	03	19	77
003	032	116	MOVCM	03	1A	4E

讨论

我们使用 DBUG 将上述程序输入到读/写存贮器。为了观察来往于 8080 寄存器和 存贮器之间的数据流，我们步进该程序来获得下列打印输出：

```
003  020  B
003  020  G
003  020
SZ 1 P 2   A    B    C    D    E    H    L    M      SP     CS
01000110  000  000  000  000  000  003  200  001  004  000  200  000

S  003  023
01000110  000  000  000  000  000  003  200  123  004  000  200  000

S  003  025
01000110  000  000  000  000  000  003  201  001  004  000  200  000

S  003  026
00000010  000  000  000  000  000  003  201  002  004  000  200  000

S  003  027
00000010  002  000  000  000  000  003  201  002  004  000  200  000

S  003  030
SZ 1 P 2   A    B    C    D    E    H    L    M      SP     CS
00000010  002  000  000  000  000  003  200  123  004  000  200  000

S  003  031
00000010  002  000  000  000  000  003  200  002  004  000  200  000

S  003  032
```

00000010 002 000 002 000 000 003 200 002 004 000 200 000

当我们把断点设置在003 020(hex 03 10)地址上，并从该地址启动程序后，数据值就被装入到HL寄存器对——H寄存器中的值为003(hex 03),L寄存器中的值为200(hex 80)。下一条指令是将立即数123(hex 53)传送到由HL寄存器对寻址的存贮单元。注意存贮器内容的变化(打印信息中M这一列)。尔后，003 025(hex 03 15)单元中的INXH指令使得L寄存器中的内容增1(如果L寄存器的内容增1后从377变为000(hex FF变为00)，产生了进位条件，那么，H寄存器的内容也将增1)。我们也观察到，当HL寄存器对中的存贮地址增1时存贮器内容的变化。

INXH指令执行后，INRM指令接着就被执行了。这条指令使得由HL寄存器对寻址的存贮单元的内容增1。在执行这条指令时，由HL寄存器对寻址的存贮单元是003 201(hex 03 81)。然后MOVAM指令将HL寄存器对寻址的存贮单元的内容送入A寄存器。DCXH指令使得L寄存器的内容减1(如果L寄存器从000减到377(hex 00减到FF)，产生了借位条件，H寄存器的内容也将减1)。保留在A寄存器中的数据值然后被存贮到HL寄存器对寻址的存贮单元。所以，A寄存器中的数据值被存放在003 200(hex 03 80)单元中。最后，MOVCM指令将存贮在由HL寄存器对寻址的存贮单元中的数据送到C寄存器中去。

例 题 三

用单步性质来执行调用指令

目的

这个例子的目的是演示怎样使用DBUG的单步性质来步进执行一条CALL指令。

程序

8进制				16进制	
存贮地址		指令	助忆符	存贮地址	指令
003	000	061	LXISP	03 00	31
003	001	100	<LO>	03 01	40
003	002	003	<HI>	03 02	03
003	003	076	MVIA	03 03	3E
003	004	240	<DATA>	03 04	A0
003	005	315	CALL	03 05	CD
003	006	020	<LO>	03 06	10
003	007	003	<HI>	03 07	03
003	010	000	NOP	03 08	00
003	011	000	NOP	03 09	00
003	020	006	MVIB	03 10	06
003	021	001	<DATA>	03 11	01
003	022	200	ADDB	03 12	**80**

```
003  023    200        ADDB       03  13      80
003  024    117        MOVCA      03  14      4F
003  025    311        RET        03  15      C9
```

讨论

当我们步进执行这个程序时，我们将获得下列打印信息：

```
003  000  B
003  000  G
003  000
SZ 1 P 2   A    B    C    D    E    H    L    M      SP       CS
01000110  000  000  000  000  000  000  000  303  003  100  157  374
S  003  003
01000110  240  000  000  000  000  000  000  303  003  100  157  374
S  003  005  ?
003  005 / 315
003  006 / 020
003  007 / 003
003  020  B
×
003  020
01000110  240  001  000  000  000  000  000  303  003  076  003  010

S  003  022
10000010  241  001  000  000  000  000  000  303  003  076  003  010

S  003  023
10000010  242  001  000  000  000  000  000  303  003  076  003  010

S  003  024
SZ 1 P 2   A    B    C    D    E    H    L    M      SP       CS
10000010  242  001  242  000  000  000  000  303  003  076  003  010

S  003  025  ?
003  025 / 311
003  010  B
×
003  010
10000010  242  001  242  000  000  000  000  303  003  100  157  374

S  003  011
```

10000010 242 001 242 000 000 000 000 303 003 100 157 374

注意，8080 执行了 003 000(hex 03 00)单元中的 LXISP 指令后，接着又执行了 003 003(hex 03 03)单元中的 MVIA 指令。当我们企图步进执行 003 005(hex 03 05)单元中的 CALL 指令时，DBUG 打印出一个问号"?"来告诉我们刚才执行了一条不可单步执行的指令。为了找出这条指令，我们检查一下 003 005(hex 03 05)单元的内容。在这个单元中存放的是一条 CALL(315, CD)指令，我们再检查接下来的二个存贮单位，来找出将被调用的子程序的低地址和高地址。低地址是 020(hex 10)，存贮在 003 006(hex 03 06)单元中，高地址是 003(hex 03)，存贮在 003 007(hex 03 07)单元中。在把断点设置在 003 020(hex 03 10)这个新地址以后，就可以用继续(X)命令来继续执行程序。

然后执行 CALL 指令，将计算机转到 003 020(hex 03 10)的断点。注意，在执行了 CALL 指令和到达断点后，由于执行了 003 020(hex 02 10)单元中的 MVIB 指令而将 001(hex 01)送入了 B 寄存器。

用户也将观察到，栈指针已经递增了 2 次，而变为 003 076(hex 03 3E)。正如我们所期望的，返回地址(003 010, hex 03 08)被存放在栈中，显示在 CS 这一列。下二条指令都是 ADDB。所以当我们步进执行了 003 023(hex 03 13)单元中的最后一条 ADDB 指令时，我们将发现 A 寄存器中的内容是 242(hex A2)。当我们步进执行 003 024(hex 03 14)单元中的指令时，A 寄存器中的内容就被传送到了 C 寄存器。当我们企图步进执行 003 025(hex 03 15)单元中的指令时，DBUG 却没有执行这条指令。而当我们检查 003 025(hex 03 15)单元中的内容时，我们发现这是一条无条件返回(RET = 311, C9)指令。

知道了 RET 指令会从栈中弹出返回地址这一点，我们检查栈的内容 (CS)，并用存放在栈中的 003 010(hex 03 08)作为下一个断点地址。在把断点设置在 003 010(hex 03 08)地址上以后，就使用继续命令(X)来继续执行程序。在 RET 这条指令的执行期间，8080 将返回地址从栈中弹出，并使栈指针递增 2 次。

当碰到 003 010(hex 03 08)这个断点地址后，这个地址中的指令也就被执行了，且打印输出寄存器的内容。被执行的这条指令是一条 NOP (无操作)指令。所以存放在寄存器中的值不会改变。当我们步进执行 003 011(hex 03 09)单元中的指令时，存放在寄存器中的值也不会改变。这是因为 8080 执行了另一条 NOP 指令。

附　录　A

可供用户调用的通用子程序

名称	使用的寄存器	地址
BYTE	A,B,C	020130；1058H

本子程序从电传打字机的纸带阅读器中读 2 个记录。第一个纸带记录含有 2 个最大有效位，第二个记录含有 6 个最小有效位，带着 A 寄存器中的 8 位字节退出该子程序。更详细的信息请见附录 B。

LDDR	A,B,C	020235；10 9DH

本子程序在电传打字机的纸带穿孔机上穿出 10 寸长度的 8 进制 200 (hex 80) 的纸带(即带头和带尾—译注)。

BYTOUT	A,B,C	020251；10 A9H

本子程序在电传打字机的纸带穿孔机上把 A 寄存器的内容穿出。2 个最大有效位被移入A 寄存器的 2 个最小有效位，然后作为一个记录穿孔。剩下 6个最小有效位被穿在下一个纸带记录上。

CRLT	A,B	021075；11 3DH

本子程序在电传打字机设备上打出一个回车和换行。

OCTIN(HEXIN)	A,B,C,E	021 122；11 52H

本子程序用来将 8 进制数转换为 2 进制数。对 8 进制数打印 3 个数字，对 16 进制数打印 2 个数字。当 8080 返回时，该数值的 2 进制等价值将在 E 寄存器中。无效数值将使程序控制返回到 DBUG。

HLOUT	A,B,C	021 227；11 97H

本子程序以空格分离的 8 进制或 16 进制格式打印输出 HL 寄存器对的内容。

OCTOUT(HEXOUT)	A,B,C	021 234；11 9CH

本子程序以 8 进制或 16 进制格式打印输出 A 寄存器中的内容。

READ	A	021 247；11 BCH

本子程序启动电传打字机的阅读控制继电器，并从纸带上输入 8 位字符，然后带着 A 寄存器的内容退出该子程序。

TTY1	A	021 317；11 CFH

当电传打字机键盘上的键被揿下时，本子程序将该键盘字符输入。然后带着 A 寄存器中的 8 位字符退出。这个键盘字符并不打印在电传打字机上。

TTYIN　　A,B　　021 331；11 D9H

当电传打字机键盘上的键被揿下时，本子程序将该键盘字符输入，并打印在电传打字机上。然后带着 A 寄存器和 B 寄存器中的 7 位数值退回。最大有效位(D7)将置为 0。

TTYOUT　　A,B　　021 336；11 DEH

本子程序将打入的字符送入A寄存器。当8080返回时，这个字符将在 A 寄存器和 B 寄存器中。

SPC　　A,B,C　　021 362；11 F2H

本子程序在电传打字机上打印一个空格。

MORSPC　　A,B,C　　021 364；11 F4H

本子程序把将在电传打字机上打印输出的空格数(oct 或 hex)送入 C 寄存器。当 C 寄存器的内容递减到 0 时，8080 将从这个子程序退出。

BIT　　A,B,C,E　　023 207；13 87H

本子程序以 2 进制的格式打印输出 E 寄存器中的内容。最大有效位将在左边，最小有效位将在右边。在这串二进制数后面打印输出一个空格。

NXTLET　　A,B,C,H,L　　023 267；13 B7H

本子程序被用来在电传打字机上打印输出 ASCII 码信息。带着 HL 寄存器对所指向的第一个字符进入该子程序，这个字符后面跟着一串要被打印的字符。为了从这个子程序退出，代码 000(hex 00)必须被放在串末。当 8080 从这个子程序返回时，HL 寄存器将指向串末的 000(hex 00)。该子程序有输出多个空格的特殊功能。任何一个小于 010(hex 08)而不是 000(hex 00)的字符，其值就表示打印输出空格的个数。

存贮在存贮器中的下列字符

101, 102, 103, 003, 061, 063, 215, 212, 002, 130, 131, 132, 000

将导致打印输出下列信息：

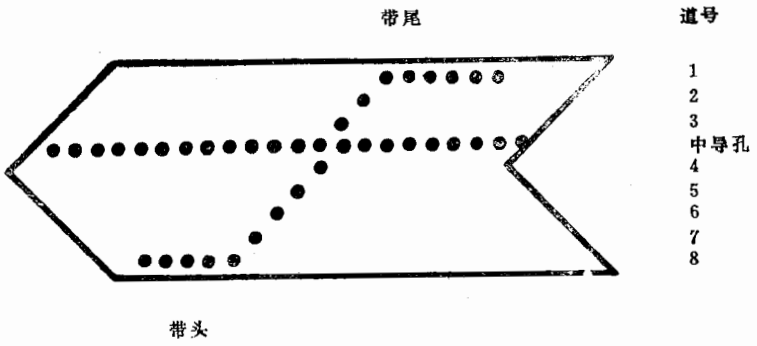
A B C 1 2 3

XYZ

附　录 B

DBUG 对纸带格式的要求

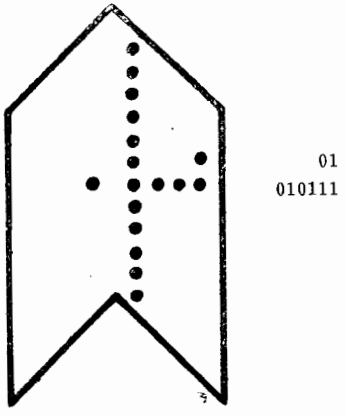
纸带格式如下：



第 8 道(在它上面有孔的话)表示带头/带尾(LDR/TRLR；8 进制数为 200，16 进制数为 80)。第7道(在它上面有孔的话)表示接下来的 4 个记录(字)是地址信息。数据信息和地址信息均被分为一个 2 位字节和一个 6 位字节。所以，一个127(hex 57)的代码将以下面的形式出现：

01 010 111 = 01 010111

0101 0111 = 01 010111



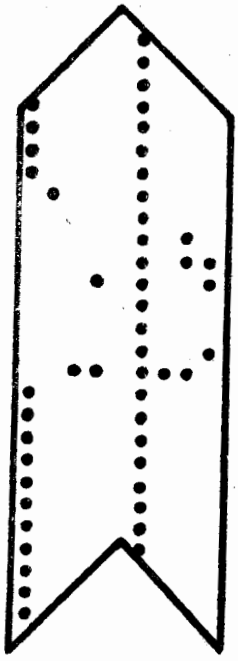
下列纸带已有地址和数据信息。地址 002 321(hex 0 2 D1)将是：

002 = 00 000 010 = 00 000010

321 = 11 010 001 = 11 010001

02 = 0000 0010 = 00 000010

D1 = 1101 0001 = 11 010001



带头(LDR)

地址标志

00 000010 高地址

11 010001 低地址

00 000000 数据

01 110110 数据

带尾(TRLR)

附　录 C

8080 微机指令集

助忆符	说明	指令代码 D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	时钟周期数
MOVr_1,r_2	Move register to register	0	1	D	D	D	S	S	S	5
MOVM,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOVr,M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVIr	Move immediate register	0	0	D	D	D	1	1	0	7
MVIM	Move immediate memory	0	0	1	1	0	1	1	0	10
INRr	Increment register	0	0	D	D	D	1	0	0	5
DCRr	Decrement register	0	0	D	D	D	1	0	1	5
INRM	Increment memory	0	0	1	1	0	1	0	0	10
DCRM	Decrement memory	0	0	1	1	0	1	0	1	10
ADDr	Add register to A	1	0	0	0	0	S	S	S	4
ADCr	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUBr	Subtract register from A	1	0	0	1	0	S	S	S	4
SBBr	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANAr	And register with A	1	0	1	0	0	S	S	S	4
XRAr	Exclusive or registerwithA	1	0	1	0	1	S	S	S	4
ORAr	or register with A	1	0	1	1	0	S	S	S	4
CMPr	Compare register with A	1	0	1	1	1	S	S	S	4
ADDM	Add memory to A	1	0	0	0	0	1	1	0	7
ADCM	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUBM	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBBM	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANAM	And memory with A	1	0	1	0	0	1	1	0	7
XRAM	Exclusive Or memory withA	1	0	1	0	1	1	1	0	7
ORAM	Or memory with A	1	0	1	1	0	1	1	0	7
CMPM	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immedia to A with carry	1	1	0	0	1	1	1	0	7

SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
ARL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	1	0	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11

RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXIB	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXID	Load immediate register pair D & E	0	0	0	1	0	0	0	0	10
LXIH	Load immediate register pair H & L	0	0	1	0	0	0	0	1	10
LXISP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSHB	Push register pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSHD	Push register pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSHH	Push registter H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POPB	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POPO	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POPH	Pop register pair H & L off stcak	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct Into Memory	0	0	1	1	0	0	1	0	13
LDA	Load A direct From Memory	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E,H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DADB	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DADD	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DADH	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DADSP	Add stack Pointer to H & L	0	0	1	1	1	0	0	1	10
STAXB	Store A indirect to Memory	0	0	0	0	0	0	1	0	7
STAXD	Store A indirect to Memory	0	0	0	1	0	0	1	0	7

LDAXB	Load A indirect From Memory	0	0	0	0	1	0	1	0	7
LDAXD	Load A indirect From Memory	0	0	0	1	1	0	1	0	7
INXB	Increment B & C registers	0	0	0	0	0	0	1	1	5
INXD	Increment D & E registers	0	0	0	1	0	0	1	1	5
INXH	Increment H & L registers	0	0	1	0	0	0	1	1	5
INXSP	Ihcrement stack pointer	0	0	1	1	0	0	1	1	5
DCXB	Decrement B & C	0	0	0	0	1	0	1	1	5
DCXD	Decrement D & E	0	0	0	1	1	0	1	1	5
DCXH	Decrement H & L	0	0	1	0	1	0	1	1	5
DCXSP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carrv	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

附　录 D

DBUG 命令集

在命令方式时：

P　穿纸带(必须指定好地址信息)

R　读纸带

S　单步，执行一条指令并打印结果

X　继续执行程序

K　删除断点

如果要打入一个数，DBUG 要求有一个地址，高地址和低地址必须是 2 个 3 位 8 进制数或 2 个 2 位 16 进制数的格式。

在打入一个地址(XXXYYY 或 XXYY)后，用户可以打入一个 B，G，L，/，CR 或 LF：

XXXYYY　B　在指定的地址上扦入一个断点

G　从指定的地址开始执行程序

L　从指定的存贮地址开始，以 8 进制数或 16 进制数列出存贮地址和内容

/　输出指定的存贮地址中的内容

CR　退出命令方式

LF　存贮新的数据字，并输出下一个存贮地址及内容

附　录 E

DBUG 程序清单(8 进制)

```
/  ***     ***   *****   ***   *
/ *   *  *   *    *    *   *  *
/ *   *  *        *    *   *  *
/ *   *  *   *    *    ******  *
/  ***     ***     *    *   *  *****

/THIS IS A 1K VERSION OF TYCHON DBUG.
/VERSION 3, 1MAY77.

                *000 070
000 070 303 VECT,   JMP     /WHEN THE RST7 INSTRUCTION IS EXE
                            UTED
000 071 032         ENTRY   /WE VECTOR TO HERE AND THEN JUMP
                            BACK
000 072 022         0       /TO DBUG.

                *003 300
003 300 000 BRKADD, 000     /THIS IS THE ADDRESS OF THE USER
003 301 000         000     /INSERTED BREAKPOINT.
003 302 000 NEWADD, 000     /THIS IS THE ADDRESS OF THE NEXT
                            EXECUT.
003 303 000         000     /ABLE INSTRUCTION AFTER THE
                            BREAKPOINT.
003 304 000 TMP,    000     /THREE BYTES OF THE USER'S PROGRAM
                            ARE
003 305 000         000     /DUPLICATED HERE IN RAM STARTING
003 306 000         000     /FROM THE BREAKPOINT ADDRESS.
003 307 303         JMP
003 310 323       EXECUT    /AFTER EXECUTING 1 INSTRUCTION,WE
003 311 002         0       /TYPE OUT THE REGISTERS ETC.
003 312 000 LAB,    000     /COUNTER FOR REGISTER "LABELS".
003 313 000 USERSK, 000     /THE USER'S STACK POINTER
003 314 000         000     /IS STORED HERE.
003 315 000 TEMPO,  000     /2 TEMPORARY STORAGE LOCATIONS
003 316 000         000
```

```
003 317 000 FLAG,   000      /THE A REGISTER AND FLAGS GO HERE.
003 320 000         000

                    *00 376
003 376 000 STACK,  000      /THE STACK POINTER IS SET FOR HERE.
                /THIS IS THE STARTING ADDRESS OF DBUG !

                    *020 000
020 000 000 START,  NOP      /THESE 8 MEMORY LOCATIONS
020 001 000         NOP      /CAN BE USED TO INITIALIZE
020 002 000         NOP      /A USART
020 003 000         NOP
020 004 000         NOP
020 005 000         NOP
020 006 000         NOP
020 007 000         NOP
020 010 061         LXISP
020 011 376         STACK
020 012 003         0
020 013 021         LXID     /SET D &E = 000 070
020 014 070         VECT
020 015 000         0
020 016 041         LXIH     /SET H&L = STORAGE OF "JMP VECT"
020 017 361         BUFF2
020 020 023         0
020 021 315         CALL     /WRITE THE "JMP VECT" INTO RAM
020 022 247         SETUP    /STARTING FROM 000 070.
020 023 023         0
020 024 021         LXID     /D &E = 000 130
020 025 307         TMP+3
020 026 003         000
020 027 016         MVIC     /SET THE TRANSFER COUNTER TO 6
020 030 006         006      /FOR JMP(3),LAB(1) & USP(2)
020 031 315         CALL     /TRANSFER 6 BYTES FROM H &L TO D&E
020 032 251         SETUP+2  /SKIP THE MVIC 003 !
020 033 023         0
020 034 341 QUEST,  POPH     /DESTROY RETURN ADDRESS ON STACK
020 035 315         CALL
020 036 075         CRLF     /TYPE A CARRIAGE RETURN-LINE FEED
```

```
020 037 021          0
020 040 076          MVIA
020 041 277          277
020 042 315          CALL
020 043 336          TTYOUT /THEN TYPE OUT A ?.
020 044 021          0
020 045 315 QUEST2   CALL   /TYPE A CARRIAGE RETURN-LINE FEED
020 046 075          CRLF
020 047 021          0
020 050 315 QUEST3,  CALL
020 051 331          TTYIN  /GET A KEYBOARD CHARACTER(USER
                            COMMAND)
020 052 021          0
020 053 376          CPI
020 054 122          "R"    /R FOR READ A PAPER TAPE
020 055 302          JNZ
020 056 143          NOREAD /NOT AN R,TRY A P
020 057 020          0
020 060 315 RDR,     CALL   /READ A CHARACTER
020 061 274          READ
020 062 021          0
020 063 376          CPI    /WAS THE CHARACTER READ IN
020 064 200          200    /LEADER(200)?
020 065 312          JZ     /YES,GET ANOTHER CHARACTER
020 066 060          RDR
020 067 020          0
020 070 376 CHKFRM,CPI      /NO,WAS IT AN ADDRESS FLAG?
020 071 100          100    /(SEE THE PAPER TAPE FORMAT)
020 072 312          JZ     /YES,IT WAS AN ADDRESS FLAG
020 073 115          AIN
020 074 020          0
020 075 376          CPI    /WAS IT TRAILER(200)?
020 076 200          200
020 077 312          JZ     /YES,RETURN TO THE COMMAND DECODER
020 100 045          QUEST2
020 101 020          0
020 102 315 NOEND,   CALL   /NO,IT MUST BE DATA TO BE SAVED
020 103 133          BYTE1  /COMBINE THE 2 BIT AND 6 BIT BYTE
020 104 020          0      /INTO AN 8 BIN BYTE
```

```
020 105 167          MOVMA /SAVE IT IN MEMORY
020 106 043          INXH
020 107 315 NXTIN,   CALL  /AND GET ANOTHER PAPER TAPE FRAME.
020 110 274          READ
020 111 021          0
020 112 303          JMP
020 113 070          CHKFRM
020 114 020          0

020 115 315 AIN,     CALL  /AN ADDRESS FLAG WAS FOUND
020 116 130          BYTE  /THE NEXT 4 FRAMES REPRESENT
020 117 020          0     /THE H&L ADDRESS
020 120 147          MOVHA
020 121 315          CALL
020 122 130          BYTE
020 123 020          0
020 124 157          MOVLA /SAVE THE ADDRESS IN H&L
020 125 303          JMP
020 126 107          NXTIN
020 127 020          0

                     /THE FORMAT OF THE PAPER TAPE IS,THE 2 MSB'S ARE
                     PUNCHED OUT FIRST, THEN THE REMAINING 6 BITS
                     /ARE PUNCHED OUT.

020 130 315 BYTE,    CALL  /GET THE 2 MSB'S
020 131 274          READ
020 132 021          0
020 133 017 BYTE1,   RRC   /ROTATE THEM INTO THE MSB
020 134 017          RRC
020 135 117          MOVCA /AND SAVE THEM IN C
020 136 315          CALL  /GET THE NEXT PAPER TAPE FRAME
020 137 274          READ
020 140 021          0
020 141 201          ADDC  /ADD THE MSB'S TO THE LSB'S
020 142 311          RET   /AND RETURN WITH THE 8 BIT # IN A.

020 143 376 NOREAD,CPI
020 144 120          "P"   /P FOR PUNCH A PAPER TAPE
```

```
020 145 302          JNZ     /NOT A P,SEE IF IT WAS
020 146 267          NOPUN   /A VALID OCTAL(0-7)NUMBER BEING
020 147 020          0       /USED TO SPECIFY AN ADDRESS.
020 150 315 PUNCH,   CALL    /AFTER THE P WAS TYPED IN,PRINT
020 151 075          CRLF    /A CR & LF.
020 152 021          0
020 153 315          CALL    /GET THE BEGINNING OF FILE ADDRESS
020 154 116          TWOOCT
020 155 021          0
020 156 353          XCHG    /PUT IT IN H&L
020 157 315          CALL
020 160 075          CRLF
020 161 021          0
020 162 315          CALL    /GET THE END OF FILE ADDRESS
020 163 116          TWOOCT
020 164 021          0
020 165 173          MOVAE   /CALCULATE THE NUMBER OF BYTES
020 166 225          SUBL    /TO BE PUNCHED OUT.
020 167 137          MOVEA
020 170 172          MOVAD
020 171 234          SBBH
020 172 127          MOVDA
020 173 023          INXD    /AND SET THE RESULT TO 1MORE.
020 174 315          CALL    /PUNCH OUT 10" OF LEADER
020 175 235          LDDR
020 176 020          0
020 177 075          MVIA    /PUNCH OUT AN ADDRESS FLAG
020 200 100          100
020 201 315          CALL
020 202 336          TTYOUT
020 203 021          0
020 204 174          MOVAH
020 205 315          CALL    /PUNCH OUT THE BEGINNING HI ADDRESS
020 206 251          BYTOUT  /AS A 2 & 6 BIT BYTE.
020 207 020          0
020 210 175          MOVAL
020 211 315          CALL    /PUNCH OUT THE BEGINNING LO ADDRESS
020 212 251          BYTOUT  /AS A 2 & 6 BIT BYTE.
020 213 020          0
```

```
020 214 176 NXTOUT,MOVAM /THEN START PUNCHING OUT THE
020 215 315          CALL   /CONTENTS OF MEMORY ADDRESSED BY
                            H&L
020 216 251          BYTOUT
020 217 020          0
020 220 043          INXH
020 221 033          DCXD   /ANY MORE DATA TO BE PUNCHED OUT?
020 222 172          MOVAD
020 223 263          ORAE
020 224 302          JNZ    /YES,GET THE NEXT MEMORY LOCATION
020 225 214          NXTOUT/NO,THEN PUNCH OUT TRAILER AND
020 226 020          0      /RETURN TO THE COMMAND DECODER
020 227 315          CALL
020 230 235          LDDR
020 231 020          0
020 232 303          JMP
020 233 045          QUEST2
020 234 020          0
020 235 016 LDDR,    MVIC
020 236 144          144    /C=144=100 DECIMAL=10" OF LEADER
020 237 076          MVIA
020 240 200          200    /CODE FOR LEADER AND TRAILER
020 241 315          CALL
020 242 336          TTYOUT/PUNCH THE 200 CODE
020 243 021          0
020 244 015          DCRC   /10" OF LEADER OR TRAILER YET ?
020 245 302          JNZ    /NO,PUNCH SOME MORE
020 246 237          LDDR+2
020 247 020          0
020 250 311          RET    /YES,RETURN

020 251 117 BYTOUT,MOVCA /DIVIDE AN 8 BIT BYTE INTO
020 252 346          ANI    /A 2 BIT AND A 6 BIT BYTE
020 253 600          300
020 254 007          RLC
020 255 007          RLC
020 256 315          CALL   /PUNCH OUT THE 2 MSB'S
020 257 336          TTYOUT
020 260 021          0
```

```
020 261 171          MOVAC
020 262 346          ANI      /THEN PUNCH OUT THE 6 LSB,S
020 263 077          077
020 264 303          JMP
020 265 336          TTYOUT
020 266 021          0

                     /A"K"TYPED IN MEANS TO REMOVE THE BREAKPOINT.
                     /A"X"TYPED IN MEANS TO CONTINUE EXECUTION
                     /A"S"TYPED IN MEANS TO SINGLE STEP

                     /SINCE THESE ARE COMMANDS WHICH REQUIRE THE
                     /USE OF THE BREAKPOINT,DBUG CHECKS TO SEE
                     /WHETHER THE USER HAS SET A BREAKPOINT
                     /SOMEPLACE. IF ONE HAS NOT BEEN SET,THEN
                     /WE CANNOT EXECUTE THESE COMMANDS.

020 267 137 NOPUN,   MOVEA    /SAVE THE CHARACTER IN E
020 270 376 BCS,     CPI      /S IS FOR SINGLE STEP
020 271 123          "S"
020 272 312          JZ       /IT WAS A S,EXECUTE 1 INSTRUCTION !
020 273 134          STEP
020 274 023          0
020 275 376          CPI      /X IS FOR CONTINUE(FULL SPEED)
020 276 130          "X"      /PROGRAM EXECUTION
020 277 312          JZ       /IT WAS A X,CONTINUE EXECUTION
020 300 074          CONTIN
020 301 023          0
020 302 376          CPI      /K IS FOR REMOVE THE LAST BREAKPOINT
020 303 113          "K"      /(THE USER HAS FOUND A BETTER PLACE)
020 304 302          JNZ      /NOT A K
020 305 331          NOBCS    /NOT S,K OR X,MAYBE ITS AN OCTAL
                              NUMBER
020 306 020          0
020 307 052          LHLD     /A"K"WAS TYPED TO REMOVE A BREAKPO
                              INT
020 310 300          BRKADD/NOW SEE IF THERE IS ONE TO REMOVE !
020 311 003          0
020 312 176          MOVAM    /IS THERE A BREAKPOINT IN MEMORY
```

```
020 313 376          CPI     /TO BE REMOVED ?
020 314 377          377
020 315 302          JNZ     /NO,GO BACK TO THE COMMAND DECODER
020 316 035          QUEST + 1/SO IT'S NOT A VALID COMMAND.
020 317 020          0
020 320 315          CALL    /IT WAS A K,TAKE THE CONTENTS OF
                             "TMP"
020 321 177          RSTRE   /AND PUT IT BACK IN THE USER'S
                             PROGRAM
020 322 023          0       /USING"BRKADD"AS THE ADDRESS.
020 323 315          CALL    /SINCE THE BREAKOINT HAS BEEN
                             REMOVED
020 324 240          ZEROIT  /WE HAVE TO RESET BRKADD TO 377 377
020 325 023          0       /TO PREVENT THE USE OF K,X OR S.
020 326 303          JMP
020 327 045          QUERST2/NOW GET ANOTHER COMMAND.
020 330 020          0
020 331 315 NOBCS,   CALL    /IT WAS A VALID OCTAL NUMBER,SO
020 332 107          SPCOCT  /CONVERT THE ASCII TO OCTAL
020 333 021          0       /(THERE MUST BE 3 DIGITS TYPED IN)
020 334 123          MOVDE   /SAVE THE HI ADDRESS IN D
020 335 315          CALL    /NOW GET THE 3 DIGIT LO ADDRESS
020 336 122          OCTIN   /ON RETURNING,IT IS IN E
020 337 021          0
020 340 353          XCHG    /THE ADDRESS IS NOW IN H&L
020 341 315          CALL    /NOW THAT AN ADDRESS HAS BEEN
                             SPECIFIED,
020 342 331          TTYIN   /SEE WHAT THE USER WANTS TO DO AT
020 343 021          0       /THAT ADDRESS!
020 344 376          CPI
020 345 107          "G"     /G=GO,BEGINNING AT THE USER'S
                             ADDRESS
020 346 302          JNZ     /A G WAS NOT TYPED IN,TRY A B
020 347 365          NOGO
020 350 020          0
020 351 315          CALL    /PRINT OUT A CR,LF AFTER THE G
020 352 075          CRLF    /WAS TYPED IN
020 353 021          0
020 354 257          XRAA    /SET A-E TO 000
```

```
020 355 107             MOVBA
020 356 117             MOVCA
020 357 127             MOVDA
020 360 137             MOVEA
020 361 345             PUSHH  /PUSH THE ADDRESS ON THE STACK
020 362 147             MOVHA  /THEN SET H&L TO 000
020 363 157             MOVLA
020 364 311             RET    /POP THE USER'S ADDRESS INTO THE
                               /PROGRAM COUNTER,AND OFF YOU GO.

020 365 376 NOGO,       CPI    /B AFTER AN ADDRESS=SET A
                               BREAKPOINT.
020 366 102             "B"
020 367 312             JZ     /IT WAS A B,SO SET UP TH E BREAKPOINT.
020 370 376             BREAK
020 371 021             0
020 372 376             CPI    /L=LIST THE CONTENTS OF MEMORY
020 373 114             "L"    /BEGINNING AT THE USER
020 374 312             JZ     /SPECIFIED ADDRESS.
020 375 200             LIST
020 376 021             0
020 377 376             CPI    /A"/"MEANS TYPE OUT THE
021 000 057             "/"    /CONTENTS OF MEMORY.
021 001 302             JNZ
021 002 035             QUEST+1/NOT A G,B,L OR / SO TYPE OUT
021 003 020             0      /A? AND GET ANOTHER COMMAND.
021 004 315             CALL   /WE TYPED A SLASH SO PUT OUT A SPACE
021 005 362             SPC
021 006 021             0
021 007 176 NXTLOC,MOVAM /GET THE CONTENTS OF MEMORY
021 010 315             CALL
021 011 234             OCTOUT /PRINT IT OUT AS A 3
021 012 021             0      /DIGIT OCTAL NUMBER
021 013 315             CALL   /DID THE USER TEAN TYPE IN A
021 014 317             TTYI   /CR,LF OR AN OCTAL#?
021 015 021             0
021 016 346             ANI    /MASK OUT ANY PARITY BIT
021 017 177             177
021 020 376             CPI
```

```
021 021 015          015       /CR = RETURN TO THE COMMAND DECODER
021 022 312          JZ
021 023 045          QUEST2
021 024 020          0
021 025 376          CPI
021 026 012          012       /LF = CONTENTS OF THE NEXT MEMORY
                                LOCATION
021 027 302          JNZ       /ANOTHER OCTAL NUMBER WAS TYPED IN
021 030 044          CNT       /(NEW CONTENTS OF THE SAME LOCATION)
021 031 021          0
021 032 315 NXTLN,   CALL      /IT WAS A LE,SO PRINT OUT THE
021 033 075          CRLF
021 034 021          0
021 035 043          INXH
021 036 315          CALL      /ADDRESS AND CONTENTS OF THE NEXT
021 037 352          HLSLSH    /CONSECUTIVE MEMORY LOCATION
021 040 021          0         /HLSLSH = H &L THEN A /AND SPACE.
021 041 303          JMP
021 042 007          NXTLOC    /NOW TYPE OUT THE CONTENTS OF
021 043 021          0         /THE NEXT LOCATION

021 044 315 CNT,     CALL      /OUTPUT THE FIRST CHARACTER(NUMBER)
021 045 336          TTYOUT
021 046 021          0
021 047 315          CALL      GO TO THE OCTAL INPUT ROUTINE
021 050 107          SPCOCT
021 051 021          0
021 052 163          MOVME     /SAVE THE NUMBER INPUT
021 053 315          CALL      /SEE IF A CR,LF OR OCTAL
021 054 317          TTYI      /NUMBER WAS THEN TYPED IN
021 055 021          0
021 056 346          ANI       /MASK OUT ANY PARITY BIT
021 057 177          177
021 060 376          CPI
021 061 015          015       /CR = RETURN TO THE COMMAND DECODER
021 062 312          JZ
021 063 045          QUEST2
021 064 020          0
021 065 376          CPI
```

```
021 066 012            012
021 067 302            JNZ      /IN WASN'T A LF,IT MUST BE
012 070 044            CNT      /NEW CONTENTS AGAIN !
021 071 021            0
021 072 303            JMP
021 073 032            NXTLN    / A LF WAS TYED IN,TYPE OUT THE
021 074 021            0        /NEXT MEM.LOCATION AND ITS CONTENTS

021 075 076 CRLF,      MVIA     /TYPE OUT A CR & LF
021 076 215            215
021 077 315            CALL
021 100 336            TTYOUT
021 101 021            0
021 102 076            MVIA
021 103 212            212
021 104 303            JMP
021 105 336            TTYOUT
021 106 021            0

                    /THIS IS THE OCTAL INPUT ROUTINE

021 107 036 SPCOCT,    MVIE     /E IS A TEMPORARY STORAGE REGISTER
021 110 000            000
021 111 016            MVIC     /C IS A DIGIT INPUT COUNTER
021 112 003            003
021 113 303            JMP
021 114 131            NXTOCT+3
021 115 021            0

021 116 315 TWOOCT,    CALL     /GET ONE THREE DIGIT OCTAL NUMBER
021 117 122            OCTIN
0211 200 21            0
021 121 123            MOVDE    /SAVE THE VALUE IN"D"
021 122 257 OCTIN,     XRAA
021 123 137            MOVEA
021 124 016            MVIC
021 125 003            003
021 126 315 NXTOCT,    CALL     /GET A TTY CHARACTER
021 127 331            TTYIN
```

```
021 130 021          0
021 131 376          CPI      /AND SEE IF IT IS A VALID OCTAL #
021 132 060          060      /IF IT ISN'T,TYPE A ?
021 133 332          JC
021 134 034          QUEST
021 135 020          0
021 136 376          CPI
021 137 070          070
021 140 322          JNC
021 141 034          QUEST
021 142 020          0
021 143 346          ANI      /MASK OUT ALL BUT THE 3 LSB'S
021 144 007          007
021 145 107          MOVBA    /SAVE THE # IN B
021 146 173          MOVAE    /GET THE PREVIOUS # (INITIALLY = 0)
021 147 007          RLC      /ROTATE IT INTO THE 3 MIDDLE BITS.
021 150 007          RLC
021 151 007          RLC
021 152 200          ADDB     /ADD THE # JUST INPUT
021 153 137          MOVEA    /AND SAVE IT IN E
021 154 015          DCRC     /3 DIGITS YET ?
021 155 302          JNZ      /NO,GET ANOTHER 1
021 156 126          NXTOCT
021 157 021          0
021 160 303          JMP      /YES,THEN TYPE OUT A SPACE.
021 161 362          SPC
021 162 021          0
021 163 000          0
021 164 000          0
021 165 000          0
021 166 000          0
021 167 000          0
021 170 000          0
021 171 000          0
021 172 000          0
021 173 000          0
021 174 000          0
021 175 000          0
021 176 000          0
```

```
021 177 000             0

                        /L = LIST THE CONSECUTIVE MEMORY LOCATIONS
                        /AND THEIR CONTENTS UNTIL THE USER PRESSES
                        /A PRINTING TELETYPE KEY.

021 200 315 LIST,       CALL    /TYPE A CR & LF
021 201 075             CRLF
021 202 021             0
021 203 315             CALL    /THEN THE H&L ADDRESS, A SLASH
021 204 352             HLSLSH  /AND A SPACE
021 205 021             0
021 206 176             MOVAM   /GET THE CONTENTS OF MEMORY
021 207 315             CALL    /AND PRINT IT OUT.
021 210 234             OCTOUT
020 211 021             0
021 212 043             INXH    /INCREMENT THE MEMORY POINTER
021 213 333             IN      /ANY TELETYPE KEY PRESSED YET ?
021 214 021             021     /TTY SENSE REGISTER
021 215 346             ANI
021 216 001             001
021 217 312             JZ      /NO, LIST OUT THE NEXT LOCATION
021 220 200             LIST
021 221 021             0
021 222 333             IN      /YES, INPUT THE TERMINATING
021 223 020             020     /CHARACTER
021 224 303             JMP
021 225 045             QUEST2  /AND GO TO THE COMMAND DECODER
021 226 020             0

                        /TYPE OUT THE CONTENTS OF H&L AS 2
                        /3 DIGIT OCTAL NUMBERS, WITH A SPACE AFTER
                        /THE HI AND LO ADDRESS.

021 227 174 HLOUT,      MOVAH
021 230 315             CALL
021 231 234             OCTOUT
021 232 021             0
021 233 175             MOVAL
```

```
021 234 117 OCTOUT,MOVCA
021 235 346         ANI
021 236 300         300
021 237 007         RLC
021 240 007         RLC
021 241 306         ADI     /YOU ADD 260 TO MAKE IT AN ASCII #
021 242 260         260
021 243 315         CALL
021 244 336         TTYOUT
021 245 021         0
021 246 171         MOVAC
021 247 346         ANI
021 250 070         070
021 251 017         RRC
021 252 017         RRC
021 253 017         RRC
021 254 306         ADI
021 255 260         260
021 256 315         CALL
021 257 336         TTYOUT
021 260 021         0
021 261 171         MOVAC
021 262 346         ANI
021 263 007         007
021 264 306         ADI
021 265 260         260
021 266 315         CALL
021 267 336         TTYOUT
021 270 021         0
021 271 303         JMP
021 272 362         SPC
021 273 021         0

021 274 323 READ,   OUT     /PULSE THE READER CONTROL RELAY
021 275 021         021
021 276 000         NOP
021 277 000         NOP
021 300 000         NOP
021 301 000         NOP
```

```
021 302 000             NOP
021 303 000             NOP
021 304 000             NOP
021 305 000             NOP
021 306 000             NOP
021 307 000             NOP
021 310 000             NOP
021 311 000             NOP
021 312 000             NOP
021 313 000             NOP
021 314 000             NOP
021 315 000             NOP
021 316 000             NOP
021 317 333 TTYI,       IN      /HAS A CHARACTER BEEN RECEIVED YET?
021 320 021             021
021 321 346             ANI
021 322 001             001
021 323 312             JZ      /NO,KEEP WAITING FOR THE FLAG
021 324 317             TTYI
021 325 021             0
021 326 333             IN      /YES,THEN INPUT IT
021 327 020             020
021 330 311             RET

021 331 315 TTYIN,      CALL    /GET A TELETYPE CHARACTER
021 332 317             TTYI
021 333 021             0
021 334 346             ANI     /AND THEN ECHO IT
021 335 177             177
021 336 107 TTYOUT,MOVBA /SAVE THE CHARACTER TO BE PRINTED
                                IN B
021 337 333             IN      /IS THE TRANSMITTER READY YET ?
021 340 021             021
021 341 346             ANI
021 342 004             004
021 343 312             JZ      /NO,KEEP WAITING
021 344 337             TTYOUT+1
021 345 021             0
021 346 170             MOVAB   /YES,GET THE CHARACTER
```

```
021 347 323          OUT     /AND TRANSMIT IT.
021 350 020          020
021 315 311          RET

021 352 315 HLSLSH,CALL
021 353 227          HLOUT   /TYPE OUT H & L,THEN A SPACE
021 354 021          0
021 355 076          MVIA    /THEN TYPE A"/"
021 356 257          257
021 357 315          CALL
021 360 336          TTYOUT
021 361 021          0

                     /PRINT OUT SPACES,THE NUMBER OF WHICH
                     /IS STORED IN C.

021 362 016 SPC,     MVIC
021 363 001          001
021 364 076 MORSPC,MVIA
021 365 240          240     /ASCII SPACE
021 366 315          CALL
021 367 336          TTYOUT
021 370 021          0
021 371 015          DCRC    /ANYMORE TO PRINT ?
021 372 302          JNZ     /YES,PRINT ANOTHER 1
021 373 364          MORSPC
021 374 021          0
021 375 311          RET     /NO,THEN RETURN

021 376 315 BREAK,   CALL    /THE USER WANTS TO PUT A BREAKPOINT
021 377 004          BRK     /IN A PROGRAM
022 000 022          0
022 001 303          JMP
022 002 045          QUEST2
022 003 020          0

022 004 042 BRK,     SHLD    /SAVE THE ADDRESS OF THE BREAK-
022 005 300          BRKADD/POINT ADDRESS IN BRKADD.
022 006 003          0
```

```
022 007 353           XCHG    /NOW WE WANT TO DUPLICATE PART OF
022 010 041           LXIH    /USER'S PROGRAM, STARING A THE
022 011 304           TMP     /BREAKPOINT ADDRESS, DOWN AT "TMP"
022 012 003           0
022 013 032 BREAK1,   LDAXD
022 014 107           MOVBA
022 015 076           MVIA    /WRITE A RST7 INTO THE USER'S PROGRAM
022 016 377           377     /THIS IS A RST7 INSTRUCTION
022 017 022           STAXD
022 020 160           MOVMB
022 021 043           INXH
022 022 023           INXD
022 023 032           LDAXD
022 024 167           MOVMA
022 025 043           INXH
022 026 023           INXD
022 027 032           LDAXD
022 030 167           MOVMA
022 031 311           RET     /WE HAVE DONE 3 MEMORY LOCATIONS

                      /WE "HIT" THE RST7 IN THE USER'S PROGRAM AND
                      /WE GET TO ENTRY BY THE JUMP INSTRUCTION
                      STORED
                      /AT LOC.000 070. THE RST7 MAY BE CHANGED
                      /AT THIS POINT, ALL OF THE REGISTERS
                      /AND THE STACK HAVE USER VALUES IN THEM
                      /SO DBUG HAS TO PRESERVE TEEM.

022 032 042 ENTRY,    SHLD    /SAVE H&L IN "TEMPO"
022 033 315           TEMPO
022 034 003           0
022 035 365           PUSHPSW
022 036 341           POPH    /GET THE FLAGS AND A IN H&L
022 037 042           SHLD    /NOW SAVE THEM IN "FLAG"
022 040 317           FLAG
022 041 003           0
022 042 041           LXIH    /NOW DESTROY THE RETURN ON THE
                      STACK
022 043 002           002     /DUE TO THE RST INSTRUCTION AND
```

```
022 044 000             000     /GET THE STACK POINTER IN H&L
022 045 071 SENTRY,DADSP
022 046 042             SHLD    /SAVE THE USER'S STACK IN"USERSK"
022 047 313             USERSK
022 050 003             0
022 051 061             LXISP   /NOW SET THE SP SO DBUG CAN USE IT
022 052 376             STACK
022 053 003             0
022 054 052             LHLD    /GET H&L BACK INTO H&L FROM"TEMPO"
022 055 315             TEMPO
022 056 003             0
022 057 345             PUSHH   /PUSH B-H ONTO DBUG'S STACK
022 060 325             PUSHD
022 061 305             PUSHB
022 062 052             LHLD    /NOW GET THE FLAGS AND A INTO H&L
022 063 317             FLAG
022 064 003             0
022 065 345             PUSHH   /AND PUSH THEM ON THE STACK
022 066 052             LHLD    /GET THE USER SELECTED BREAKPOINT
022 067 300             BRKADD/ADDRESS AND TYPE IT OUT SO WE DON'T
022 070 003             0       /FORGET WHAT IT WAS.
022 071 315             CALL
022 072 227             HLOUT
022 073 021             0
022 074 072             LDA     /GET THE USER INSTRUCTION AT"TMP"
022 075 304             TMP
022 076 003             0
022 077 107             MOVBA   /AND SEE IF IT IS EXECUTABLE.
022 100 376             CPI
022 101 303             303     /CAN'T DO A JUMP
022 102 312             JZ
022 103 247             NOGOOD
022 104 022             0
022 105 376             CPI
022 106 315             315     /CAN'T CALL ANY SUBRUTINES
022 107 312             JZ
022 110 247             NOGOOD
022 111 022             0
022 112 376             CPI
```

```
022 113 311             311     /RETURN
022 114 312             JZ
022 115 247             NOGOOD
022 116 022             0
022 117 376             CPI
022 120 351             351     /PCHL
022 121 312             JZ
022 122 247             NOGOOD
022 123 022             0
022 124 376             CPI
022 125 333             333     /OUTPUT INSTRUCTION IS A 2 BYTE INSTR.
022 126 312             JZ
022 127 235             IMMED
022 130 022             0
022 131 376             CPI
022 132 323             323     /AND SO ARE INPUT INSTRUCIONS.
022 133 312             JZ
022 134 235             IMMED
022 135 022             0
022 136 346 NO300,      ANI     /IT WASN'T 1 OF THOSE, TRY THESE.
022 137 307             307
022 140 376             CPI
022 141 306             306
022 142 312             JZ
022 143 235             IMMED
022 144 022             0
022 145 376             CPI
022 146 302             302     /CONDITIONAL JUMPS
022 147 312             JZ
022 150 247             NOGOOD
022 151 022             0
022 152 376             CPI
022 153 304             304     /CONDITIONAL CALLS
022 154 312             JZ
022 155 247             NOGOOD
022 156 022             0
022 157 376             CPI
022 160 300             300     /CONDITIONAL RETURES
022 161 312             JZ
```

```
022 162 247          NOGOOD
002 163 022          0
022 164 376          CPI
022 165 307          307       /RESTARTS
022 166 312          JZ
022 167 247          NOGOOD
022 170 022          0
022 171 376          CPI
022 172 006          006       /IMMEDIATE MOVES
022 173 312          JZ
022 174 235          IMMED
022 175 022          0
022 176 376          CPI
022 177 001          001       /LOAD IMMEDIATE REGISTER PAIR
022 200 312          JZ
022 201 224          IMMSL
022 202 022          0
022 203 376          CPI
022 204 002          002       /STA,LDA,SHLD OR LHLD
022 205 312          JZ
022 206 263          IMMSL2
022 207 022          0
022 210 041 SING,    LXIH      /IT WAS SINGLE BYTE INSTRUCTION
022 211 000          000       /SO WE WRITE 2 NOPS IMMEDIATELY
022 212 000          000       /AFTER THE INSTRUCTION.
022 213 042          SHLD
022 214 305          TMP+1
022 215 003          0
022 216 052          LHLD      /SET H&L=TO THE CONTENTS OF
                               "BRKADD"
022 217 300          BRKADD
022 220 003          0
022 221 303          JMP
022 222 276          ONEMOR
022 223 022          0
022 224 170 IMMSL1,  MOVAB     /IS THE INSTRUCTIONS REALLY A
022 225 346          ANI       /SINGLE BYTE OR 3 BYTE?
022 226 010          010
022 227 302          JNZ       /IF A=010,ITS A SINGLE BYTE
```

```
022 230 210            SING
022 231 022            0
022 232 303            JMP      /IF A=000,ITS A 3 BYTE
022 233 271            THREBY
022 234 022            0
022 235 257 IMMED,     XRAA     /IMMEDIATE AND I-O INSTRUCTIONS
022 236 062            STA      /ARE TWO BYTE INSTRUCTIONS
022 237 306            TMP+2    /SO WE SAVE 1 NOP AFTER THE
022 240 003            0        /INSTRUCTION.
022 241 052            LHLD
022 242 300            BRKADD
022 243 003            0
022 244 303            JMP
022 245 275            TWOMOR
022 246 022            0

022 247 052 NOGOOD,LHLD   /IF THE BREAPOINT WAS PLACED ON A
022 250 300            BRKADD/NONEXECUTABLE INSTRUCTION,WE RE-
022 251 003            0        /MOVE THE RST7 FROM THE USER'S
022 252 042            SHLD     /PROGRAM AND GO TO THE COMMAND
022 253 302            NEWADD/DECODER
022 254 003            0
022 255 315            CALL
022 256 177            RSTRE
022 257 023            0
022 260 303            JMP
022 261 040            QUEST+4
022 262 020            0
022 263 170 IMMSL2,    MOVAB    /IS IT REALLY A STA,LDA,SHLD OR LHLD?
022 264 376            CPI
022 265 042            042
022 266 332            JC       /NO,THEN IT MUST BE A SINGLE BYTE.
022 267 210            SING
022 270 022            0
022 271 052 THREBY,LHLD
022 272 300            BRKADD
022 273 003            0
022 274 043            INXH
022 275 043 TWOMOR,INXH
```

```
022 276 043 ONEMOR,INXH
022 277 042         SHLD    /STORE THE ADDRESS OF THE NEXT
022 300 302         NEWADD/INSTRUCTION
022 301 003         0
022 302 361         POPPSW  /GET BACK ALL THE USER'S REGISTERS
022 303 301         POPB
022 304 321         POPD
022 305 341         POPH
022 306 042         SHLD    /SAVE H&L IN"TEMPO"
022 307 315         TEMPO
022 310 003         0
022 311 052         LHLD    /SET H&L WITH THE USER'S STACK
                            POINTER
022 312 313         USERSK
022 313 003         0
022 314 371         SPHL    /NOW SET THE SP WITH THAT VALUE
022 315 052         LHLD    /NOW GHT H&L BACK TO WHAT THEY
                            WERE
022 316 315         TEMPO
022 317 003         0
022 320 303         JMP     /NOW DO THE 1,2 OR 3 BYTE INSTRUCTION
022 321 304         TMP     /STORED AT"TMP"(WE MAY ALSO DO
022 322 003         0       /1 OR 2 NOP'S).

                    /AFTER EXECUTING THE INSTRUCTION AT "TMP"
                    /WE JUMP BACK TO HERE. THE CONTENTS OF
                    /THE REGISTERS MUST BE SAVED AGAIN,SP'S
                    /SWITCHED ETC. BEFORE WE CAN TYPE OUT THE
                    /CONTENTS OF THE REGISTERS.

022 323 024 EXECUT,SHLD     /FROM "TMP" WE JUMP TO HERE.
022 324 315         TEMPO   /SAVE H&L IN "TEMPO"
022 325 003         0
022 326 365         PUSHPSW
022 327 341         POPH
022 330 042         SHLD    /SAVE THE FLAGS AND A IN "FLAG"
022 331 317         FLAG
022 332 003         0
022 333 041         LXIH    /SET H&L = TO 000
```

```
022 334 000          0
022 335 000          0
022 336 071          DADSP  /PUT THE USER'S SP INTO H&L
022 337 042          SHLD   /AND SAVE IT IN "USERSK"
022 340 313          USERSK
022 341 003          0
022 342 061          LXISP  /SET UT A SP FOR DBUG TO USE
022 343 376          STACK
022 344 003          0
022 345 052          LHLD   /SET H&L TO WHAT THEY WHERE
022 346 315          TEMPO
022 347 003          0
022 350 345          PUSHH  /SAVE B-H ON THE STACK
022 351 325          PUSHD
022 352 305          PUSHB
022 353 052          LHLD   /GET THE FLAGS AND A INTO H&L
022 354 317          FLAG
022 355 003          0
022 356 345          PUSHH  /AND PUT THEM ON THE STACK ALSO.
022 357 315          CALL   /TYPE OUT A CR & LF
022 360 075          CRLF
022 361 021          0
022 362 041          LXIH   /DECREMENT THE REGISTER LABEL
                            COUNTER
022 363 312          LAB
022 364 003          0
022 365 065          DCRM
022 366 314          CZ     /IF 0,TYPE OUT THE REGISTER LABELS
022 367 262          LABEL
022 370 023          0
022 371 041          LXIH
022 372 000          000
022 373 000          000
022 374 071          DADSP  /SET H&L = TO THE SP
022 375 136          MOVEM  /GET THE FLAG WORD
022 376 315          CALL   /AND DISSASSEMBLE IT INTO 1'S & 0'S
022 377 207          BIT
023 000 023          0
023 001 043          INXH
```

```
023 002 176          MOVAM /NOW GET THE A REGISTER
023 003 315          CALL   /AND TYPE IT OUT AS A 3 DIGIT OCTAL #
023 004 234          OCTOUT
023 005 021          0
023 006 026          MVID   /NOW WE GET THE OTHER REGISTERS
023 007 003          003    /AND TYPE OUT THEIR OCTAL VALUES
023 010 036 TWOTIM,MVIE
023 011 002          002
023 012 043          INXH
023 013 043          INXH
023 014 176 MOR1,    MOVAM
023 015 315          CALL
023 016 234          OCTOUT
023 017 021          0
023 020 053          DCXH
023 021 035          DCRE
023 022 302          JNZ
023 023 014          MOR1
023 024 023          0
023 025 043          INXH
023 026 043          INXH
023 027 025          DCRD   /ANYMORE REGISTER PAIRS ?
023 030 302          JNZ    /YES,DO ANOTHER
023 031 010          TWOTIM
023 032 023          0
023 033 126          MOVDM /NO,GET THE VALUES OF H&L INTO H&L
023 034 053          DCXH
023 035 136          MOVEM
023 036 353          XCHG
023 037 176          MOVAM /GET CNTS OF MEMORY ADDRESSED BY
                           H & L
023 040 315          CALL   /AND TYPE OUT ITS VALUE
023 041 234          OCTOUT
023 042 021          0
023 043 052          LHLD   /SET H&L = TO THE USER'S SP
023 044 313          USERSK
022 045 003          0
023 046 315          CALL   /AND TYPE OUT THE 2 OCTAL WORDS
023 047 227          HLOUT
```

```
023 050 021          0
023 051 043          INXH    /GET THE HI OFF THE STACK
023 052 176          MOVAM
023 053 315          CALL    /AND PRINT IT OUT
023 054 234          OCTOUT
023 055 021          0
023 056 053          DCXH
023 057 176          MOVAM /THEN GET THE LO OFF THE STACK
023 060 315          CALL    /AND PRINT IT OUT
023 061 234          OCTOUT
023 062 021          0
023 063 315          CALL    /PRINT A CR & LE AFTER ALL OF THIS
023 064 075          CRLF
023 065 021          0
023 066 315          CALL    /REMOVE THE BREAKPOINT(RST7)
023 067 177          RSTRE   /FROM THE USER'S PROGRAM
023 070 023          0
023 071 303          JMP     /AND GO BACK TO THE COMMAND DECODER
023 072 045          QUESTZ
023 073 020          0

                /IF WE TYPE A X TO CONTINUE,WE HAVE TO GET ALL
                /OF THE REGISTERS BACK THE WAY THEY WERE,WITH
                /THE VALUES IN THEM THAT THE USER ESTABLISHED,
                AL-
                /ONG WITH THE USERS'S STACK POINTER,AND
                THEN WE CAN USE THE CONTENTS OF"NEWADD"
                /AS THE ADDRESS FROM WHERE WE SHOULD
                CONTINUE THE
                /PROGRAMS EXECUTION.

023 074 052 CONTIN, LHLD
023 075 300          BRKADD/IS THERE A BREAKPOINT ADDRESS?
023 076 003          0
023 077 043          INXH
023 100 174          MOVAH
023 101 265          ORAL
023 102 312          JZ      /NO,BRKADD WAS SET TO 377 377
023 103 035          QUEST+1
```

```
023 104 020          0
023 105 315          CALL    /THERE IS AN ADDRESS,CONTINUE
023 106 075          CRLF
023 107 021          0
023 110 361 CNTIN1,  POOPPSW/GET BACK A-H
023 111 301          POPB
023 112 321          POPD
023 113 341          POPH
023 114 042          SHLD    /SAVE H&L IN"TEMPO"
023 115 315          TEMPO
023 116 003          0
023 117 052          LHLD    /GET THE USER'S SP
023 120 313          USERSK
023 121 003          0
023 122 371          SPHL    /AND PUD IT INTO THE SP
023 123 052          LHLD    /GET H&L BACK THE WAY THEY SHOULD
                             BE
023 124 315          TEMPO
023 125 003          0
023 126 345          PUSHH   /PUT H&L ON THE STACK
023 127 052          LHLD    /GET THE ADDRESS WHERE WE SHOULD
023 130 302          NEWADD/CONTINUE PROGRAM EXECUTION
023 131 003          0
023 132 343          XTHL    /NEW ADDRESS ON STACK, H&L THE
023 133 311          RET     /WAY THEY WERE. HERE WE GO....

                /IF WE TYPE S FOR SINGLE STEP,WE WANT TO
                EXECUTE
                /1 INSTRUCTION AND THEN SEE WHAT EFFECT IT HAD
                /ON THE REGISTERS, STACK POINTER, STACK OR
                MEMORY.

023 134 052 STEP,    LHLD    /SEE IF THERE IS A BREAKPOINT
023 135 300          BRKADD /ALREADY SET.
023 136 003          0
023 137 043          INXH
023 140 174          MOVAH
023 141 256          ORAL
023 142 312          JZ      /IF NOT BREAKPOINT,H=L=377
```

```
023 143 035          QUEST+1/AFTER INXH,H=L=000 AND WE ERROR
023 144 020          0
023 145 315          CALL    /TYPE A SPACE AFTER THE S
023 146 362          SPC
023 147 021          0
023 150 052          LHLD    /GET THE ADDRESS OF THE NEXT EX-
023 151 302          NEWADD  /ECUTABLE INSTRUCTION.
023 152 003          0
023 153 315          CALL    /SET A BREAKPOINT AT THIS NEW
                             ADDRESS
023 154 004          BRK
023 155 022          0
023 156 361          POPPSW  /GET THE REGISTERS AND SP BACK
023 157 301          POPB    /TO WHAT THEY SHOULD BE
023 160 321          POPD
023 161 341          POPH
023 162 042          SHLD
023 163 315          TEMPO
023 164 003          0
023 165 052          LHLD
023 166 313          USERSK
023 167 003          0
023 170 371          SPHL
023 171 041          LXIH
023 172 000          0
023 173 000          0
023 174 303          JMP     /THEN JUMP TO THE BEGINNING OF
023 175 045          SENTRY  /BREAKPOINT ROUTINE
023 176 022          0

023 177 072 RSTRE,   LDA     /GET THE FIRST BYTE OF THE INSTRUCTI
                             ON
023 200 304          TMP
023 201 003          0
023 202 052          LHLD    /GET THE ADDRESS OF WHERE IT CAME
                             FROM
023 203 300          BRKADD
023 204 003          0
023 205 167          MOVMA   /PUT IT BACK IN THE USER'S PROGRAM
023 206 311          RET     /WRITING OVER THE RST7 INSTRUCTION !
```

```
                        /THIS SUBROUTINE IS USED TO UNPACK THE FLAG
                        /REGISTER INTO A STRING OF 8 1'S AND 0'S.
                        /ENTER WITH THE WORD TO BE UNPACKED IN E

023 207 016 BIT,        MVIC    /C = THE ROTATE OR BIT COUNTER
023 210 010             010
023 211 173 NXTBIT,     MOVAE   /GET THE WORD
023 212 007             RLC     /ROTATE 1 BIT LEFT INTO THE CARRY
023 213 137             MOVEA   /AND SAVE THE WORD AGAIN
023 214 332             JC      /IF THE CARRY IS SET,WE ROTATED
023 215 233             ONE     /A 1 INTO IT,SO WE PRINT A 1.
023 216 023             0
023 217 076             MVIA    /OTHERWISE,WE TYPE A 0
023 220 260             260
023 221 315 ONEOUT,     CALL    /PRINT THE CHARACTER
023 222 336             TTYOUT
023 223 021             0
023 224 015             DCRC    /8 BITS YET ?
023 225 302             JNZ     /NO,DO ANOTHR ONE
023 226 211             NXTBIT
023 227 023             0
023 230 303             JMP     /YES,NOW PRINT A SPACE
023 231 362             SPC
023 232 021             0
023 233 076 ONE,        MVIA
023 234 261             261
023 235 303             JMP
023 236 221             ONEOUT
023 237 023             0

                        /THIS CLEARS THE CONTENTS OF "BRKADD"

023 240 041 ZEROIT,     LXIH
023 241 377             377     /SET BRKADD TO 377 377
023 242 377             377
023 243 042             SHLD
023 244 300             BRKADD
023 245 003             0
023 246 311             RET
```

```
                /THIS DUPLICATES PORTIONS OF "BUFF2"
                /INTO R-W MEMORY

023 247 016 SETUP,  MVIC
023 250 003         003
023 251 176         MOVAM
023 252 022         STAXD
023 253 043         INXH
023 254 023         INXD
023 255 015         DCRC
023 256 302         JNZ
023 257 251         SETUP+2
023 260 023         0
023 261 311         RET

                /IF WE HAD DECREMEMTED "LAB" TO 0, WE COME HERE.
                /WE USE THE CONTENTS OF "REGLST"
                /TO PRODUCE THE REGISTER LABELS. NOTICE THAT ANY
                /NUMBER IN "REGLST" LESS THAN 010 (OCTAL), PRODUCES
                /THAT NUMBER OF SPACES, EXCEPT 0, WHICH
                TERMINATES
                /THE OUTPUT.

023 262 066 LABEL,  MVIM    /SET THE LABEL COUNTER TO 5 AGAIN
023 263 005         005
023 264 041         LXIH
023 265 316         REGLST /H&L POINT TO "REGLST"
023 266 023         0
023 267 176 NXTLET, MOVAM /GET A "REGLST" TABLE CHARACTER
023 270 376         CPI     /IS IT 000 ? (END OF THE TABLE ?)
023 271 000         0
023 272 310         RZ      /YES. THEN RETURN
023 273 376         CPI     /IS IT A NUMBER BELOW 010 ?
023 274 010         010     /IF SO, USE IT AS A SPACE COUNT
023 275 332         JC      /YES, SO TYPE SOME SPACES
023 276 307         SPCIT
023 277 023         0
023 300 315         CALL    /NONE OF THE ABOVE, JUST
023 301 336         TTYOUTIPRINT THE CHARACTER.
023 302 021         0
```

```
023 303 043 NXTLAB,INXH    /INCREMENT REGISTER PAIR H&L
023 304 303         JMP     /THEN GET AND INTERPRETE THE NEXT
023 305 267         NXTLET  /CHARACTER
023 306 023         0
023 307 117 SPCIT,  MOVCA
023 310 315         CALL
023 311 364         MORSPC
023 312 021         0
023 313 303         JMP
023 314 303         NXTLAB
023 315 023         0
023 316 323 REGLST, 323     /S
023 317 332         332     /Z
023 320 001         001     /1 SPACE
023 321 261         261     /1
023 322 001         001     /1 SPACE
023 323 320         320     /P
023 324 001         001     /1 SPACE
023 325 262         262     /2 (4 BIT CARRY)
023 326 002         002     /2 SPACES
023 327 301         301     /A
023 330 003         003     /3 SPACES
023 331 302         302     /B
023 332 003         003     /3 SPACES
023 333 303         303     /C
023 334 003         003     /3 SPACES
023 335 304         304     /D
023 336 003         000     /3 SPACES
023 337 305         305     /E
023 340 003         003     /3 SPACES
023 341 310         310     /H
023 342 003         003     /3 SPACES
023 343 314         314     /L
023 344 003         003     /3 SPACES
023 345 315         315     /M
023 346 004         004     /4 SPACES
023 347 323         323     /S
023 350 001         001     /1 SPACE
023 351 320         320     /P
```

```
023 352 005             005     /5 SPACES
023 353 303             303     /C
023 354 001             001     /1 SPACE
023 355 323             323     /S
023 356 215             215     /CR
023 357 212             212     /LF
023 360 000             000     /MESSAGE TERMIATOR
023 361 303 BUFF2,      JMP     /THIS IS WHAT GETS DUPLICATED
023 362 032             ENTRY   /IN RAM. THIS STARTS AT 000 070
023 363 022             0
023 364 303             JMP     /THIS GOES TO 003 307
023 365 323             EXECUT
023 366 022             0
023 367 001             001     /THIS IS THE VALU EOF"LAB"
023 370 200             200
023 371 003             003
```

AIN	= 020 115	BRKADD	= 003 300	BYTE	= 020 130	BYTE1	= 020 133
BYTOUT	= 020 251	BCS	= 020 270	BREAK	= 021 376	BRK	= 022 004
BREAK1	= 022 013	BIT	= 023 207	BUFF2	= 023 361	CHKFRM	= 020 070
CNT	= 021 044	CRLF	= 021 075	CONTIN	= 023 074	CNTIN1	= 023 110
ENTRY	= 022 032	EXECUT	= 022 323	FLAG	= 003 317	HLOUT	= 021 227
HLSLSH	= 021 352	IMMSL1	= 022 224	IMMED	= 022 235	IMMSL2	= 022 262
LAB	= 003 312	LDDR	= 020 235	LIST	= 021 200	LABEL	= 023 264
MORSPC	= 021 364	MOR1	= 023 014	NEWADD	= 003 302	NOEND	= 020 107
NXTIN	= 020 107	NOREAD	= 020 143	NXTOUT	= 020 214	NOPUN	= 020 260
NOBCS	= 020 331	NOGO	= 020 365	NXTLOC	= 021 007	NXTLN	= 021 032
NXTOCT	= 021 126	NO300	= 022 136	NOGOOD	= 022 247	NXTBIT	= 023 212
NXTLET	= 023 267	NXTLAB	= 023 303	OCTIN	= 021 122	OCTOUT	= 021 231
ONEMOR	= 022 276	ONEOUT	= 023 221	ONE	= 023 233	PUNCH	= 020 153
QUEST	= 020 034	QUEST2	= 020 045	QUEST3	= 020 050	RDR	= 020 060
READ	= 021 274	RSTRE	= 023 177	REGLST	= 023 316	STACK	= 003 376
START	= 020 000	SPCOCT	= 021 107	SPC	= 021 362	SENTRY	= 022 045
SING	= 022 210	STEP	= 023 134	SETUP	= 023 247	SPCIT	= 023 307
TMP	= 003 304	TEMPO	= 003 315	TWOOCT	= 021 116	TTYI	= 021 317
TTYIN	= 021 331	TTYOUT	= 021 336	THREBY	= 022 271	TWOMOR	= 022 275
TWOTIM	= 023 010	USERSK	= 003 313	VECT	= 000 070	ZEROIT	= 023 240

ERRORS DETECTED = 000

附　录 F

DBUG 程序清单(16 进制)

```
/ *    *  *****  *    *
/ *    *  *       *  *
/ ******  ***      **
/ *    *  *       *  *
/ *    *  *****  *    *

/ 这是 TYCHON DBUG 的 1K 版本
/ 版本 3, 1977, 5, 1

          *00 H   38 H
00 38  C3     VECT,     JMP          /当执行到 RST7 指令时，8080 就指向这里，
00 39  1A               ENTRY         并转回到 DBUG。
00 3A  12               0
          *03 H   C0H
03 C0  00     BRKADD,   000          /这里存放用户插入的断点地址。
03 C1  00               000
03 C2  00     NEWADD,   000          /这里存放断点处理后接着要被执行的指令
03 C3  00               000           地址。
03 C4  00     TMP,      000          /用户程序中从断点地址开始的 3 个字节将
03 C5  00               000           被复制在这里。
03 C6  00               000
03 C7  C3               JMP
03 C8  D3               EXECUT       /执行了一条指令后，打印输出寄存器内容等。
03 C9  12               0
03 CA  00     LAB,      000          /寄存器标记计数器单元
03 CB  00     USERSK,   000          /用户栈指针保留单元
03 CC  00               000
03 CD  00     TEMPO,    000          /2 个暂存单元
03 CE  00               000
03 CF  00     FLAG,     000          /保留A寄存器和标志位的存贮单元
03 D0  00               000

          *03 H   FEH
03 FE  00     STACK,    000          /设置栈指针为该地址值

/这是 DBUG 的启动地址!
*10 H   00 H
```

```
10 00  00   START,   NOP        /这8个单元可以用来对USART进行初始
10 01  00            NOP         化。
10 02  00            NOP
10 03  00            NOP
10 04  00            NOP
10 05  00            NOP
10 06  00            NOP
10 07  00            NOP
10 08  31            LXISP
10 09  FE            STACK       (设置栈指针——译注)
10 0A  03            0
10 0B  11            LXID        /置DE为00 38
10 0C  38            VECT
10 0D  00            0
10 0E  21            LXIH        /将转至转移向量的地址值(标号为
10 0F  F1            BUFF2        BUFF2)置入HL
10 10  13            0
10 11  CD            CALL        /把"JMP  VECT"(13F1开始——译注)
10 12  A7            SETUR        写入从00 38开始的RAM中
10 13  13            0
10 14  11            LXID        /置DE为03 C7
10 15  C7            TMP+3
10 16  03            000
10 17  0E            MVIC        /将发送计数器(Transfer Counter)置为
                                  6，供传送JMP(3)，LAB(1)和USP(2)
                                  之用

10 18  06            006
10 19  CD            CALL        /从HL指示的地址开始(13 F4—译注)传
                                  送6个字节到DE指示的地址空间中(03
                                  C7)，跳过MVIC
10 1A  A9            SETUP+2      003!(子程序SETUP中第一条指令—译
10 1B  13            0            注)
10 1C  E1   QUEST,   POPH        /废除栈中的返回地址
10 1D  CD            CALL
10 1E  3D            CRLF        /打印一个回车换行
10 1F  11            0
10 2O  3E            MVIA
10 21  BF            277         (是八位(带校验位)的ASCII码"?"—译注)
```

```
10 22  CD            CALL
10 23  DE            TTYOUT  /然后打印一个问号？
10 24  11            0
10 25  CD  QUEST2,    CALL    /打印一个回车换行
10 26  3D            CRLF
10 27  11            0
10 28  CD  QUEST3,    CALL
10 29  D9            TTYIN   /取一个键盘字符(用户命令)
10 2A  11            0       (是一直处在等待打入中—译注)
10 2B  FE            CPI
10 2C  52            "R"     /是用于读纸带的 R 命令？
10 2D  C2            JNZ
10 2E  63            NOREAD  /不是 R，试试是 P 否
10 2F  10            0
10 30  CD   RDR,     CALL    /是 R,则从 TTY 阅读器读一个字符
10 31  BC            READ
10 32  11            0
10 33  FE            CPI     /读入的字符是带头(200)吗？
10 34  80            200
10 35  CA            JZ      /是，取下一个字符
10 36  30            RDR
10 37  10            0
10 38  FE  CHKFRM,   CPI     /否，是地址标志吗？(看看纸带记录)
10 39  40            100
10 3A  CA            JZ      /是，它是地址标志
10 3B  4D            AIN
10 3C  10            0
10 3D  FE            CPI     /是带尾吗？
10 3E  80            200
10 3F  CA            JZ      /是，返回到命令译码
10 40  25            QUEST2
10 41  10            0
10 42  CD  NOEND,    CALL    /否，那它一定是要保留的数据。把 2 位字节和 6
10 43  5B            BYTE1     位字节拼成一个 8 位字节。
10 44  10            0
10 45  77            MOVMA   /把该字节保留在存贮器中
10 46  23            INXH    /修改地址准备下一个记录
10 47  CD  NXTIN,    CALL    /取另一个纸带记录
10 48  BC            READ
```

```
10 49  11              0
10 4A  C3              JMP
10 4B  38              CHKFRM   /重复读纸带
10 4C  10              0

10 4D  CD  AIN,        CALL     /找到地址标志，那么下面4个记录表示地址高低位
10 4E  58              BYTE       注：  地址高——十六位地址的高八位，
10 4F  10              0                地址低——十六位地址的低八位。
10 50  67              MOVHA    /地址高存于H中
10 51  CD              CALL     /从纸带读二个记录
10 52  58              BYTE
10 53  10              0
10 54  6F              MOVLA    /把地址保留在HL中(地址低存于L中)
10 55  C3              JMP
10 56  47              NXTIN
10 57  10              0

                       /纸带记录是这样的：先穿出2个最大有效位，然后再穿出剩下
                       的6位

10 58  CD  BYTE,       CALL     /取2个最大有效位
10 59  BC              READ
10 5A  11              0
10 5B  0F  BYTE1,      RRC      /把它们移入最大有效位(D7,D6—译注)
10 5C  0F              RRC
10 5D  4F              MOVCA    /把它们保留在C中
10 5E  CD              CALL     /取下一个纸带记录
10 5F  BC              READ
10 60  11              0
10 61  81              ADDC     /将最大有效位和最小有效位相拼
10 62  C9              RET      /带着A中的8位数返回

10 63  FE NOREAD, CPI
10 64  50              "P"      /P命令，用于穿纸带
10 65  C2              JNZ      /不是P，看看是否是正被用来指定地址的有效的
10 66  B7              NOPUN      16进制数
10 67  10              0
10 68  CD PUNCH, CALL           /在P命令打入后，打印一个回车换行
10 69  3D              CRLF
```

```
10 6A 11          0
10 6B CD          CALL      /取文件开始地址
10 6C 4E          TWOHEX
10 6D 11          0
10 6E EB          XCHG      /把该地址送入 HL
10 6F CD          CALL
10 70 3D          CRLF
10 71 11          0
10 72 CD          CALL      /取文件结束地址
10 73 4E          TWOHEX
10 74 11          0
10 75 7B          MOVAE     /计算将被穿孔输出的字节数
10 76 95          SUBL
10 77 5F          MOVEA
10 78 7A          MOVAD
10 79 9C          SBBH
10 7A 57          MOVDA
10 7B 13          INXD      /并将 DE 增 1
10 7C CD          CALL      /穿出 10 寸带头
10 7D 9D          LDDR
10 7E 10          0
10 7F 3E          MVIA      /穿出一个地址标志
10 80 40          100
10 81 CD          CALL
10 82 DE          TTYOUT
10 83 11          0
10 84 7C          MOVAH
10 85 CD          CALL      /以一个 2 位字节和一个 6 位字节的形式穿出 地 址
10 86 A9          BYTOUT     高位
10 87 10          0
10 88 7D          MOVAL
10 89 CD          CALL      /以一个 2 位字节和一个 6 位字节的形式穿出地 址
10 8A A9          BYTOUT     低位
10 8B 10          0
10 8C 7E NXTOUT,  MOVAM     /然后开始穿出由 HL 寻址的存贮单元的内容
10 8D CD          CALL
10 8E A9          BYTOUT
10 8F 10          0
10 90 23          INXH
```

```
10 91  1B            DCXD      /是否还有数据要穿孔输出?
10 92  7A            MOVAD
10 93  B3            ORAE
10 94  C2            JNZ       /是，取下一个存贮单元的内容
10 95  8C            NXTOUT    /否，穿出带尾并返回到命令译码
10 96  10            0
10 97  CD            CALL
10 98  9D            LDDR
10 99  10            0
10 9A  C3            JMP
10 9B  25            QUEST2    (再等打入命令—译注)
10 9C  10            0
10 9D  0E  LDDR,     MVIC
10 9E  64            144       /C = 144 = 100(10进制)即为 10 寸带头
10 9F  3E            MVIA
10 A0  80            200       /带头和带尾的代码
10 A1  CD            CALL
10 A2  DE            TTYOUT    /穿200这个代码
10 A3  11            0
10 A4  0D            DCRC      /10寸带头或带尾到否?
10 A5  C2            JNZ       /否，继续穿
10 A6  9F            LDDR+2
10 A7  10            0
10 A8  C9            RET       /是，返回

10 A9  4F  BYTOUT,   MOVCA     /把 8 位字节分解为 2 位字节和 6 位字节
10 AA  E6            ANI
10 AB  C0            300
10 AC  07            RLC
10 AD  07            RLC
10 AE  CD            CALL      /穿出 2 个最大有效位
10 AE  DE            TTYOUT
10 B0  11            0
10 B1  79            MOVAC
10 B2  E6            ANI       /然后穿出 6 个最小有效位
10 B3  3F            077
10 B4  C3            JMP
10 B5  DE            TTYOUT
10 B6  11            0
```

```
                /删去断点的方法是打入"K"命令
                /连续执行的方法是打入"X"命令
                /单步执行的方法是打入"S"命令

                /由于这些命令都是要求使用断点的命令，所以 DBUG 要对用户
                 是否已经建立了断点进行检查。如果断点还没有建立,那么就不
                 能执行这些命令。

10 B7 5F NOPUN, MOVEA    /把字符保留在 E 中
10 B8 FE BCS,   CPI      /S 命令，用来步进程序
10 B9 53        "S"
10 BA CA        JZ        /是 S 命令，执行一条指令!
10 BB 5C        STEP
10 BC 13        0
10 BD FE        CPI       /X 命令，用来连续(全速)执行程序
10 BE 58        "X"
10 BF CA        JZ        /是 X 命令，连续执行程序
10 C0 3C        CONTIN
10 C1 13        0
10 C2 FE        CPI       /K 命令，用来删除最后一个断点(用户已经找到更
10 C3 4B        "K"        适当的地方来设置断点)
10 C4 C2        JNZ       /不是 K 命令
10 C5 D9        NOBCS     /不是 S，K，X 命令，也许是一个 16 进制数
10 C6 10        0
10 C7 2A        LHLD      /打入 K 命令来删除一个断点，现在来看看断点 是
10 C8 C0        BRKADD     否被删除了!
10 C9 03        0
10 CA 7E        MOVAM
10 CB FE        CPI       /在读/写存贮器中还有断点吗?
10 CC FE        377
10 CD C2        JNZ       /没有断点，所以不能使用 S，K，X 命令
10 CE 1D        QUEST+1
10 CF 10        0
10 D0 CD        CALL      /是K命令，把"TMP"中的内容取出送回到用 户 程
10 D1 7F        RSTRE      序，地址值在"BRKADD"中。
10 D2 13        0
10 D3 CD        CALL      /由于断点已被删除，我们就必须 把 BRKADD 中
10 D4 A0        ZEROIT     的内容复置为 377377,来防止 K、X、S 命令的使
10 D5 13        0          用。
```

```
10 D6 C3            JMP
10 D7 25            QUEST2      /现在取另一个命令
10 D8 10            0
10 D9 CD NOBCS,     CALL        /它是一个有效的16进制数,因此把A中的ASCII
10 DA 47            SPCHEX       码转换为16进制数(必须打入3个数字)。(纸带上
10 DB 11            0            是以八进制数格式其中一个为空格符，后二个十
                                 六进制数是有效的一译注)
10 DC 53            MOVDE       /把地址高保留在D中
10 DD CD            CALL        /现在取地址低(3个数字)
10 DE 52            HEXIN
10 DF 11            0
10 E0 EB            XCHG        /地址现在在HL中
10 E1 CD            CALL        /现在，地址已经指定，看看用户想在那个地址干
10 E2 D9            TTYIN        些什么?
10 E3 11            0
10 E4 FE            CPI
10 E5 47            "G"         /G=GO，从用户指定的地址开始执行程序
10 E6 C2            JNZ         /打入的不是G，试试B
10 E7 F5            NOGO
10 E8 10            0
10 E9 CD            CALL        /在G命令打入后，打印输出回车换行
10 EA 3D            CRLF
10 EB 11            0
10 EC AF            XRAA        /将A到E的所有寄存器清为0
10 ED 47            MOVBA
10 EE 4F            MOVCA
10 EF 57            MOVDA
10 F0 5F            MOVEA
10 F1 E5            PUSHH       /把用户的地址压入栈中
10 F2 67            MOVHA       /然后将HL清为0
10 F3 6F            MOVLA
10 F4 C9            RET         /将用户的运行地址从栈中弹出而送入程序计数器，
                                 并从这个地址开始继续执行用户程序。
10 F5 FE NOGO,      CPI         /B命令，用来将断点设在某个地址上
10 F6 42            "B"
10 F7 CA            JZ          /是B命令，那么建立断点
10 F8 FE            BREAK
10 F9 11            0
10 FA FE            CPI         /L命令，用来列出从用户指定的地址开始的存贮器
```

```
10 FB 4C          "L"          内容
10 FC CA          JZ
10 FD 80          LIST
10 FE 11          0
10 FF FE          CPI          /"/"命令，用来打印输出存贮器的内容
11 00 2F          "/"
11 01 C2          JNZ
11 02 1D          QUEST+1      /不是G、B、L、/命令，那么打印一个问号？并
11 03 10          0             取另外一个命令
11 04 CD          CALL         /打入一个斜线符/,输出一个空格
11 05 F2          SPC
11 06 11          0
11 07 7E NXTLOC,  MOVAM        /取存贮器内容(HL中为打入的地址值—译注)
11 08 CD          CALL
11 09 9C          HEXOUT       /把该内容作为一个2位的16进制数打印输出
11 0A 11          0
11 0B CD          CALL         /用户打入的是回车、换行、还是一个16进制数#？
11 0C CF          TTYI
11 0D 11          0
11 0E E6          ANI
11 0F 7F          177
11 10 FE          CPI
11 11 0D          015          /CR，返回到命令译码
11 12 CA          JZ
11 13 25          QUEST2
11 14 10          0
11 15 FE          CPI
11 16 0A          012          /LF，列出下一个存贮单元的地址和内容
11 17 C2          JNZ          /另一个16进制数被打入(相同单元中的新内容)
11 18 24          CNT
11 19 11          0
11 1A CD NXTLN,   CALL         /是换行符LF，打印输出下一个存贮单元的地址
11 1B 3D          CRLF          和内容
11 1C 11          0
11 1D 23          INXH
11 1E CD          CALL         /打印输出下一个存贮单元的地址。HLSLSH=HL
11 1F EA          HLSLSH        寄存器对的内容，再加上斜线符和空格符
11 20 11          0
11 21 C3          JMP
```

```
11 22  07          NXTLOC   /现在打印输出下一个单元的内容
11 23  11          0

11 24  CD  CNT,     CALL     /输出第一个字符(数)
11 25  DE          TTYOUT
11 26  11          0
11 27  CD          CALL     /进入 16 进制输入子程序
11 28  47          SPCHEX
11 29  11          0
11 2A  73          MOVME    /保留输入的数
11 2B  CD          CALL     /看看打入的字符是否是一个回车、换行 或 16 进 制
11 2C  CF          TTYI     数。如果是的，输入该字符
11 2D  11          0
11 2E  E6          ANI
11 2F  7F          177
11 30  FE          CPI
11 31  0D          015      /CR，返回到命令译码
11 32  CA          JZ
11 33  25          QUEST2
11 34  10          0
11 35  FE          CPI
11 36  0A          012
11 37  C2          JNZ      /它不是换行符 LF，那它一定是又一个新内容!
11 38  24          CNT
11 39  11          0
11 3A  C3          JMP
11 3B  1A          NXTLN    /换行符LF被打入，打印输出下一个存贮单元的地
11 3C  11          0        址和内容

11 3D  3E  CRLF,    MVIA     /打印输出回车换行
11 3E  8D          215
11 3F  CD          CALL
11 40  DE          TTYOUT
11 41  11          0
11 42  3E          MVIA
11 43  8A          212
11 44  C3          JMP
11 45  DE          TTYOUT
11 46  11          0
```

```
                    /这是一个 16 进制数输入程序

11 47  1E SPCHEX,  MVIE      /E 中是暂存器指针
11 48  00          000
11 49  0E          MVIC      /C 中是位数输入计数器
11 4A  02          002
11 4B  C3          JMP
11 4C  59          NXTHEX+3
11 4D  11          0

11 4E  CD TWOHEX,CALL        /取一个 2 位 16 进制数
11 4F  52          HEXIN
11 50  11          0
11 51  53          MOVDE     /把该数值保留在 D 中
11 52  AF HEXIN    XRAA
11 53  5F          MOVEA
11 54  0E          MVIC
11 55  02          002
11 56  CD NXTHEX   CALL      /取一个电传打字机字符
11 57  D9          TTYIN
11 58  11          0
11 59  FE          CPI       /并看看它是否是一个有效的 16 进制 数#，如果
11 5A  30          060        不是，打印一个问号?
11 5B  DA          JC
11 5C  1C          QUEST
11 5D  10          0
11 5E  FE          CPI
11 5F  47          "G"
11 60  D2          JNC
11 61  1C          QUEST     (打入的数超越十六进数范围就打一个问号"?",等
11 62  10          0          打新命令——译注)
11 63  FE          CPI
11 64  3A          ":"       是":"
11 65  DA          JC
11 66  6F          HEX09
11 67  11          0
11 68  FE          CPI
11 69  41          "A"       是ASCII 码?
11 6A  DA          JC
```

```
11 6B  1C              QUEST
11 6C  10              0
11 6D  C6              ADI
11 6E  09              011          /(大于9的十六进制ASCII码。变换成真十六进数
11 6F  E6 HEX09,       ANI           —译注)
11 70  0F              017          /(截取低四位,即2位十六进制数中的低位—译注)
11 71  47              MOVBA
11 72  7B              MOVAE        /(E 中为上次打入的一十六进制数—译注)
11 73  07              RLC
11 74  07              RLC
11 75  07              RLC
11 76  07              RLC          /(把十六进制数移位，即移入高位—译注)
11 77  80              ADDB         /(把 2 个十六进制数装在一个字节中，存E中
11 78  5F              MOVEA         —译注)
11 79  0D              DCRC
11 7A  C2              JNZ
11 7B  56              NXTHEX
11 7C  11              0
11 7D  C3              JMP
11 7E  F2              SPC          (印空格—译注)
11 7F  11              0

                       /L = 列出连续的存贮地址及内容，直到用户揿了一个 TTY 键

11 80  CD LIST,        CALL         /打印回车换行
11 81  3D              CRLF
11 82  11              0
11 83  CD              CALL         /然后打印 HL 中的地址，一个斜线符和一个空 格
11 84  EA              HLSLSH
11 85  11              0
11 86  7E              MOVAM        /取该地址中的内容
11 87  CD              CALL         /打印输出该内容
11 88  9C              HEXOUT
11 89  11              0
11 8A  23              INXH         /存贮器指针增 1
11 8B  DB              IN           /揿过 TTY 键否?
11 8C  11              021          /TTY 读出寄存器的口子号
11 8D  E6              ANI
11 8E  01              001
```

```
11 8F  CA            JZ          /否，列出下一单元的地址和内容
11 90  80            LIST
11 91  11            0
11 92  DB            IN          /是，输入结束字符
11 93  10            020
11 94  C3            JMP
11 95  25            QUEST2      /并进入命令译码
11 96  10            0

                     /按照2个2位的16进制数打印输出HL的内容，在地址高和
                      地址低后面跟一个空格

11 97  7C  HLOUT,    MOVAH
11 98  CD            CALL
11 99  9C            HEXOUT
11 9A  11            0
11 9B  7D            MOVAL
11 9C  CD  HEXOUT,   CALL
11 9D  A6            HEX
11 9E  11            0
11 9F  79            MOVAC
11 A0  CD            CALL
11 A1  A6            HEX
11 A2  11            0
11 A3  C3            JMP
11 A4  F2            SPC
11 A5  11            0

11 A6  0F  HEX,      RRC         /(把A中十六进数变换成ASCII码并输出——译注)
11 A7  0F            RRC
11 A8  0F            RRC
11 A9  0F            RRC
11 AA  4F            MOVCA
11 AB  E6            ANI
11 AC  0F            017
11 AD  C6            ADI
11 AE  B0            260
11 AF  FE            CPI         /是否大于9
11 B0  BA            272
```

```
11 B1  DA            JC
11 B2  B6            NMB09
11 B3  11            0
11 B4  C6            ADI        /(修改成 ASCII 码 A—F—译注)
11 B5  07            007
11 B6  C3  NMB09,    JMP
11 B7  DE            TTYOUT
11 B8  11            0
11 B9  00            0
11 BA  00            0
11 BB  00            0

11 BC  D3  READ,     OUT        /启动阅读控制继电器
11 BD  11            021
11 BE  00            NOP
11 BF  00            NOP
11 C0  00            NOP
11 C1  00            NOP
11 C2  00            NOP
11 C3  00            NOP
11 C4  00            NOP
11 C5  00            NOP
11 C6  00            NOP
11 C7  00            NOP
11 C8  00            NOP
11 C9  00            NOP
11 CA  00            NOP
11 CB  00            NOP
11 CC  00            NOP
11 CD  00            NOP
11 CE  00            NOP
11 CF  DB  TTYI,     IN         /已经接收了一个字符吗?
11 D0  11            021
11 D1  E6            ANI
11 D2  01            001
11 D3  CA            JZ         /否，继续等待标志
11 D4  CF            TTYI
11 D5  11            0
11 D6  DB            IN         /是，输入这个字符
```

```
11 D7 10              020
11 D8 C9              RET

11 D9 CD  TTYIN,    CALL        /取一个 TTY 字符
11 DA CF            TTYI
11 DB 11            0
11 DC E6            ANI         /在将该字符同代码 177 进行"与"操作后，回送该
11 DD 7F            177          字符
11 DE 47  TTYOUT,   MOVBA       /把该字符保留在 B 中
11 DF DB            IN          /发送器准备好了没有?
11 E0 11            021
11 E1 E6            ANI
11 E2 04            004
11 E3 CA            JZ          /否，继续等待
11 E4 DF            TTYOUT+1
11 E5 11            0
11 E6 78            MOVAB       /是，取这个字符
11 E7 D3            OUT         /并发送该字符
11 E8 10            020
11 E9 C9            RET

11 EA CD  HLSLSH,   CALL
11 EB 97            HLOUT       /打印输出 HL 的内容和一个空格
11 EC 11            0
11 ED 3E            MVIA        /然后打印输出一个斜线符/
11 EE AF            257
11 EF CD            CALL
11 F0 DE            TTYOUT
11 F1 11            0

                    /打印输出一串空格，其个数在 C 中

11 F2 0E  SPC,      MVIC
11 F3 01            001
11 F4 3E  MORSPC,   MVIA
11 F5 A0            240         /代表空格的 ASCII 代码(带校验位—译注)
11 F6 CD            CALL
11 F7 DE            TTYOUT
11 F8 11            0
```

```
11 F9 0D            DCRC      /还有空格要打印吗?
11 FA C2            JNZ       /是，打印另一个空格
11 FB F4            MORSPC
11 FC 11            0
11 FD C9            RET       /否，返回

11 FE CD BREAK,     CALL      /用户想把一个断点置入程序
11 FF 04            BRK
12 00 12            0
12 01 C3            JMP
12 02 25            QUEST2
12 03 10            0

12 04 22 BRK,       SHLD      /把断点地址保留在 BRKADD 中
12 05 C0            BRKADD
12 06 03            0
12 07 EB            XCHG      /现在我们要复制部分用户程序，从断点地址开始，
12 08 21            LXIH       把用户程序中的 3 个字节送到"TMP"中
12 09 C4            TMP
12 0A 03            0
12 0B 1A BREAK1,    LDAXD
12 0C 47            MOVBA
12 0D 3E            MVIA      /将 RST7 指令写入用户程序
12 0E FF            377       /这是一条 RST7 指令
12 0F 12            STAXD
12 10 70            MOVMB
12 11 23            INXH
12 12 13            INXD
12 13 1A            LDAXD
12 14 17            MOVMA
12 15 23            INXH
12 16 13            INXD
12 17 1A            LDAXD
12 18 77            MOVMA
12 19 C9            RET       /我们已将上述 3 个存贮单元加工复制好了
```

/当碰到用户程序中的 RST7 指令时，DBUG 就由存贮在 00 38 单元中的 JMP 指令而进入某一入口（用户自定义——译注）。在这点上，可以修改 RST7 指令。由于所有的寄存器和栈

中都有用户的值，所以 DBUG 必须保护好它们。

```
12 1A 22  ENTRY,  SHLD      /把 HL 保留在"TEMPO"中
12 1B CD          TEMPO
12 1C 03          0
12 1D F5          PUSHPSW
12 1E E1          POPH      /把标志和 A 寄存器送入 HL
12 1F 22          SHLD      /现在把它们保留在"FLAG"中
12 20 CF          FLAG
12 21 03          0
12 22 21          LXIH      /现在由于 RST 指令破坏了栈中的返回 地 址，故
12 23 02          002        我们将栈指针保存到 HL 中
12 24 00          000
12 25 39  SENTRY, DADSP
12 26 22          SHLD      /把用户栈指针保留在"USERSK"
12 27 CB          USERSK
12 28 03          0
12 29 31          LXISP     /现在建立 DBUG 的栈指针，以使 DBUG 能工作
12 2A FE          STACK
12 2B 03          0
12 2C 2A          LHLD      /把"TEMPO"中的内容(为原 HL)送回 HL(即恢
12 2D CD          TEMPO      复 HL——译注)
12 2E 03          0
12 2F E5          PUSHH     /把 B 到 H 的所有寄存器都压入栈 (DBUG的 栈)
12 30 D5          PUSHD      中
12 31 C5          PUSHB
12 32 2A          LHLD      /现在，把标志和A寄存器送入 HL
12 33 CF          FLAG
12 34 03          0
12 35 E5          PUSHH     /把它们压入栈中
12 36 2A          LHLD      /取用户选择的断点地址并把它打印输出，使我们
12 37 C0          BRKADD     能够记住它
12 38 03          0
12 39 CD          CALL
12 3A 97          HLOUT
12 3B 11          0
12 3C 3A          LDA       /取"TMP"中的用户指令
12 3D C4          TMP
12 3E 03          0
12 3F 47          MOVBA     /看看它是否是能够被执行的指令
```

```
12 40  FE            CPI
12 41  C3            303          /不能执行的转移指令
12 42  CA            JZ
12 43  A7            NOGOOD
12 44  12            0
12 45  FE            CPI
12 46  CD            315          /不能执行的调用指令
12 47  CA            JZ
12 48  A7            NOGOOD
12 49  12            0
12 4A  FE            CPI
12 4B  C9            311          /返回指令
12 4C  CA            JZ
12 4D  A7            NOGOOD
12 4E  12            0
12 4F  FE            CPI
12 50  E9            351          /PCHL 指令
12 51  CA            JZ
12 52  A7            NOGOOD
12 53  12            0
12 54  FE            CPI
12 55  DB            333          /输出指令是 2 个字节的指令
12 56  CA            JZ
12 57  9D            IMMED
12 58  12            0
12 59  FE            CPI
12 5A  D3            323          /输入指令
12 5B  CA            JZ
12 5C  9D            IMMED
12 5D  12            0
12 5E  E6  NO300,    ANI          /如果这些都不是，继续测试
12 5F  C7            307
12 60  FE            CPI
12 61  C6            306
12 62  CA            JZ
12 63  9D            IMMED
12 64  12            0
12 65  FE            CPI
12 66  C2            302          /条件转移指令
```

```
12 67  CA          JZ
12 68  A7          NOGOOD
12 69  12          0
12 6A  FE          CPI
12 6B  C4          304          /条件调用指令
12 6C  CA          JZ
12 6D  A7          NOGOOD
12 6E  12          0
12 6F  FE          CPI
12 70  C0          300          /条件返回指令
12 71  CA          JZ
12 72  A7          NOGOOD
12 73  12          0
12 74  FE          CPI
12 75  C7          307          /重启动指令
12 76  CA          JZ
12 77  A7          NOGOOD
12 78  12          0
12 79  FE          CPI
12 7A  06          006          /立即数传送指令
12 7B  CD          JZ
12 7C  9D          IMMED
12 7D  12          0
12 7E  FE          CPI
12 7F  01          001          /立即数送寄存器对指令
12 80  CA          JZ
12 81  94          IMMSL1
12 82  12          0
12 83  FE          CPI
12 84  02          002          /STA，LDA，SHLD 或 LHLD 指令
12 85  CA          JZ
12 86  B3          IMMSL2
12 87  12          0
12 88  21  SING,    LXIH         /它是一种单字节的指令，所以我们在这种指令后
12 89  00          000           面立即写上二条 NOP 指令
12 8A  00          000
12 8B  22          SHLD
12 8C  C5          TMP+1
12 8D  03          0
```

```
12 8E 2A            LHLD     /将"BRKADD"中的内容送入 HL
12 8F C0            BRKADD
12 90 03            0
12 91 C3            JMP
12 92 BE            ONEMOR   /(准备单字节指令—译注)
12 93 12            0
12 94 78 IMMSL1,    MOVAB    /是真正可执行的指令
12 95 E6            ANI      /单字节还是3字节?
12 96 08            010
12 97 C2            JNZ      /如果 A = 010，则是单字节指令
12 98 88            SING
12 99 12            0
12 9A C3            JMP      /如果 A = 000，则是3字节指令
12 9B B9            THREBY
12 9C 12            0
12 9D AF IMMED,     XRAA     /立即型指令和 I/O 指令都是双字节指令，所以我
12 9E 32            STA       们在这种指令后面保留一条 NOP 指令
12 9F C6            TMP+2
12 A0 03            0
12 A1 2A            LHLD
12 A2 C0            BRKADD
12 A3 03            0
12 A4 C3            JMP
12 A5 BD            TWOMOR
12 A6 12            0
12 A7 2A NOGOOD,    LHLD     /如果断点放在一条不能被执行的指令上，我们就
12 A8 C0            BRKADD    从用户程序中删去 RST7 指令并进入命令译码
12 A9 03            0
12 AA 22            SHLD
12 AB C2            NEWADD
12 AC 03            0
12 AD CD            CALL
12 AE 7F            RSTRE
12 AF 13            0
12 B0 C3            JMP
12 B1 20            QUEST+4
12 B2 10            0
12 B3 78 IMMSL2,    MOVAB    /它真是一条 STA，LDA，SHLD 或 LHLD 指
12 B4 FE            CPI       令吗?
```

```
12 B5  22              042
12 B6  DA              JC          /否，那它一定是一条单字节指令
12 B7  88              SING
12 B8  12              0
12 B9  2A THREBY,      LHLD
12 BA  C0              BRKADD
12 BB  03              0
12 BC  23              INXH
12 BD  23 TWOMOR,      INXH
12 BE  23 ONEMOR,      INXH
12 BF  22              SHLD        /存贮下条指令的地址
12 C0  C2              NEWADD
12 C1  03              0
12 C2  F1              POPPSW      /送回所有的用户寄存器
12 C3  C1              POPB
12 C4  D1              POPD
12 C5  E1              POPE
12 C6  22              SHLD        /把 HL 保留在"TEMPO"中
12 C7  CD              TEMPO
12 C8  03              0
12 C9  2A              LHLD        /把用户的栈指针置入 HL
12 CA  CB              USERSK
12 CB  03              0
12 CC  F9              SPHL        /现在就用 USERSK 中的值来建立栈指针
12 CD  2A              LHLD        /现在把原 HL 的内容送回 HL（恢复 HL）
12 CE  CD              TEMPO
12 CF  03              0
12 D0  C3              JMP
12 D1  C4              TMP
12 D2  03              0

                       /在执行了"TMP"中的指令后，我们就转回到这里。在我们能够
                        打印输出寄存器内容之前，必须再次保护寄存器并要关闭栈
                        指针。

12 D3  22 EXECUT,      SHLD        /从"TMP"转到这里。把 HL 保留在"TEMPO"
12 D4  CD              TEMPO        中
12 D5  03              0
12 D6  F5              PUSHPSW
```

```
12 D7 E1        POPH
12 D8 22        SHLD    /把标志和 A 寄存器保留在"FLAG"中
12 D9 CF        FLAG
12 DA 03        0
12 DB 21        LXIH    /置 HL 为全 0
12 DC 00        0
12 DD 00        0
12 DE 39        DADSP   /把用户的栈指针置入 HL
12 DF 22        SHLD    /并把它保留在"USERSK"中
12 E0 CB        USERSK
12 E1 03        0
12 E2 31        LXISP   /设置栈指针，供 DBUG 使用
12 E3 FE        STACK
12 E4 03        0
12 E5 2A        LHLD    /恢复 HL
12 E6 CD        TEMPO
12 E7 03        0
12 E8 E5        PUSHH   /把 B 到 H 的所有寄存器保留在栈中
12 E9 D5        PUSHD
12 EA C5        PUSHB
12 EB 2A        LHLD    /把标志和 A 寄存器送入 HL
12 EC CF        FLAG
12 ED 03        0
12 EE E5        PUSHH   /同时把它们送入栈中
12 EF CD        CALL    /打印输出回车换行
12 F0 3D        CRLF
12 F1 11        0
12 F2 21        LXIH    /寄存器标记计数器递减 1
12 F3 CA        LAB
12 F4 03        0
12 F5 35        DCRM
12 F6 CC        CZ      /如果该计数器为 0，就打印输出寄存器标志
12 F7 B2        LABEL
12 F8 13        0
12 F9 21        LXIH
12 FA 00        000
12 FB 00        000
12 FC 39        DADSP   /将SP置入HL（标志寄存器和A寄存器——译注）
12 FD 5E        MOVEM   /取标志字
```

```
12 FE CD            CALL     /并把该标志字分解为2进制格式(原文DISSA-
12 FF 87            BIT       SSEMBLE错,应为DISASSEMBLE——译注)
13 00 13            0
13 01 23            INXH
13 02 7E            MOVAM    /现在取A寄存器(老的——译注)(到A累加器
                              中——译注)
13 03 CD            CALL     /并把它作为2位16进制数#打印输出
13 04 9C            HEXOUT
13 05 11            0
13 06 16            MVID     /现在我们取其他寄存器并以16进制格式打印输
13 07 03            003       出它们的数值
13 08 1E TWOTIM,    MVIE
13 09 02            002
13 0A 23            INXH
13 0B 23            INXH
13 0C 7E MOR1,      MOVAM
13 0D CD            CALL
13 0E 9C            HEXOUT
13 0F 11            0
13 10 2B            DCXH
13 11 1D            DCRE
13 12 C2            JNZ
13 13 0C            MOR1
13 14 13            0
13 15 23            INXH
13 16 23            INXH
13 17 15            DCRD     /还有其他寄存器对吗?
13 18 C2            JNZ      /是,取其内容并打印输出
13 19 08            TWOTIM
13 1A 13            0
13 1B 56            MOVDM    /否,恢复HL
13 1C 2B            DCXH
13 1D 5E            MOVEM
13 1E EB            XCHG
13 1F 7E            MOVAM    /取由HL寄存器对寻址的存贮单元的内容
13 20 CD            CALL     /并打印输出它的值
13 21 9C            HEXOUT
13 22 11            0
13 23 2A            LHLD     /将用户的SP值置入HL
```

```
13 24  CB          USERSK
13 25  03          0
13 26  CD          CALL      /并打印输出 2 个 16 进制字
13 27  97          HLOUT
13 28  11          0
13 29  23          INXH      /从栈中取出高地址
13 2A  7E          MOVAM
13 2B  CD          CALL      /并把它打印输出
13 2C  9C          HEXOUT
13 2D  11          0
13 2E  2B          DCXH
13 2F  7E          MOVAM     /从栈中取出低地址
13 30  CD          CALL      /并把它打印输出
13 31  9C          HEXOUT
13 32  11          0
13 33  CD          CALL      /在这些地址后面打印回车换行
13 34  3D          CRLF
13 35  11          0
13 36  CD          CALL      /删除用户程序中的断点(RST7)
13 37  7F          RSTRE
13 38  13          0
13 39  C3          JMP       /并返回到命令译码
13 3A  25          QUEST2
13 3B  10          0
```

/如果我们要打入 X 命令来连续执行程序的话，就必须恢复所有的寄存器和用户的栈指针，因为这些寄存器中的值都是用户建立的，在此之后我们才能用“NEWADD”中的内容作为我们连续执行程序的开始地址来开始执行程序。

```
13 3C  2A CONTIN,  LHLD
13 3D  C0          BRKADD    /有断点地址吗?
13 3E  03          0
13 3F  23          INXH
13 40  7C          MOVAH
13 41  B5          ORAL
13 42  CA          JZ        /否，将 BRKADD 置为 377 377
13 43  1D          QUEST+1
13 44  10          0
```

```
13 45  CD            CALL      /有一个断点地址，继续
13 46  3D            CRLF
13 47  11            0
13 48  F1 CNTIN1,    POPPSW    /恢复 A 到 H 的所有寄存器的内容
13 49  C1            POPB
13 4A  D1            POPD
13 4B  E1            POPH
13 4C  22            SHLD      /把 HL 保留在"TEMPO"中
13 4D  CD            TEMPO
13 4E  03            0
13 4F  2A            LHLD      /取用户的 SP 值
13 50  CB            USERSK
13 51  03            0
13 52  F9            SPHL      /并把它置入 SP
13 53  2A            LHLD      /恢复 HL
13 54  CD            TEMPO
13 55  03            0
13 56  E5            PUSHH     /把 HL 压入栈中
13 57  2A            LHLD      /取得我们继续执行程序的地址
13 58  C2            NEWADD
13 59  03            0
13 5A  E3            XTHL      /把 HL 置入栈顶，作为新的返回地址，于是我们
13 5B  C9            RET        就……运行…(用户程序了——译注)

                     /我们要求执行一条指令，然后看看该条指令对寄存器、栈指针、
                      栈或存贮器的影响。我们就打个单步命令 S

13 5C  2A STEP,      LHLD      /看看是否已经建立了断点
13 5D  C0            BRKADD
13 5E  03            0
13 5F  23            INXH
13 60  7C            MOVAH
13 61  B5            ORAL
13 62  CA            JZ        /没有断点时,H = L = 377。如果在 INXH 指令 执
13 63  1D            QUEST+1    行后，H = L = 000，那我们就出错了(说明 用 户
13 64  10            0          没有设置断点——译注)
13 65  CD            CALL      /在 S 后面打印输出一个空格
13 66  F2            SPC
13 67  11            0
```

```
13 68  2A            LHLD      /取下一条能够执行的指令地址
13 69  C2            NEMADD
13 6A  03            0
13 6B  CD            CALL      /在这个新的地址上设置一个断点
13 6C  04            BRK
13 6D  12            0
13 6E  F1            POPPSW    /恢复所有寄存器和栈指针
13 6F  C1            POPB
13 70  D1            POPD
13 71  E1            POPH
13 72  22            SHLD
13 73  CD            TEMPO
13 74  03            0
13 75  2A            LHLD
13 76  CB            USERSK
13 77  03            0
13 78  F9            SPHL
13 79  21            LXIH
13 7A  00            0
13 7B  00            0
13 7C  C3            JMP       /然后转移到断点程序的头上
13 7D  25            SENTRY
13 7E  12            0

13 7F  3A RSTRE,     LDA       /取指令的第一个字节
13 80  C4            TMP
13 81  03            0
13 82  2A            LHLD      /将"BRKADD"中的断点地址送入 HL
13 83  C0            BRKADD
13 84  03            0
13 85  77            MOVMA     /把指令送回用户程序中写着 RST 7 指令的地方!
13 86  C9            RET

                     /这个子程序用来将标志寄存器中的值分解为 8 位的 2 进制 格 式
                      的数
                     /带着将被分解在 E 寄存器中的标志进入该子程序

13 87  0E BIT,       MVIC      /C = 循环或位计数器
13 88  08            010
```

```
13 89  7B NXTBIT,   MOVAE     /取该字
13 8A  07           RLC       /将该字左移1位送入进位位
13 8B  5F           MOVEA     /再次保留该字
13 8C  DA           JC        /如果有进位，我们就把这个进位打印输出（打印
13 8D  9B           ONE        一个“1”——译注）
13 8E  13           0
13 8F  3E           MVIA      /否则，我们打印一个“0”
13 90  B0           260
13 91  CD ONEOUT,   CALL      /打印该字符
13 92  DE           TTYOUT
13 93  11           0
13 94  0D           DCRC      /8位都打完了吗?
13 95  C2           JNZ       /否，打印另一位
13 96  89           NXTBIT
13 97  13           0
13 98  C3           JMP       /是，打印一个空格
13 99  F2           SPC
13 9A  11           0
13 9B  3E ONE,      MVIA
13 9C  B1           261
13 9D  C3           JMP
13 9E  91           ONEOUT
13 9F  13           0

                    /清除“BRKADD”中的内容

13 A0  21 ZEROIT,   LXIH
13 A1  FF           377       /将BRKADD置为377 377
13 A2  FF           377
13 A3  22           SHLD
13 A4  C0           BRKADD
13 A5  03           0
13 A6  C9           RET

                    /把“BUFF2”中的一部分内容复制到存贮器中去

13 A7  0E SETUP,    MVIC
13 A8  03           003
13 A9  7E           MOVAM
```

```
13 AA 12              STAXD
13 AD 23              INXH
13 AC 13              INXD
13 AD 0D              DCRC
13 AE C2              JNZ
13 AF A9              SETUP+2
13 B0 13              0
13 B1 C9              RET
```

/如果“LAB”已被递减到0，我们就来到这里。我们使用“REGLST”的内容来产生寄存器的标记。注意，如果“REGLST”中的数小于010(8进制)，将会产生一定的空格，(除了0以外)并停止输出

```
13 B2 36 LABEL,       MVIM        /再次将标记计数器置为5
13 B3 05              005
13 B4 21              LXIH
13 B5 CE              REGLST      /HL 指向“REGLST”
13 B6 13              0
13 B7 7E NXTLET,      MOVAM       /取“REGLST”字符表
13 B8 FE              CPI         /该字符是000吗?(字符表结束了吗?)
13 B9 00              0
13 BA C8              RZ          /是，那么返回
13 BB FE              CPI         /该字符是一个小于010的数吗？如果是的，把该
13 BC 08              010          字符算作一个空格
13 BD DA              JC          /是，打印一些空格
13 BE C7              SPCIT
13 BF 13              0
13 C0 CD              CALL        /如果都不是，就打印这个字符
13 C1 DE              TTYOUT
13 C2 11              0
13 C3 23 NXTLAB,      INXH        /HL 寄存器对递增1
13 C4 C3              JMP         /然后取出并解释下一个字符
13 C5 B7              NXTLET
13 C6 13              0
13 C7 4F SPCIT,       MOVCA
13 C8 CD              CALL
13 C9 F4              MORSPC
13 CA 11              0
```

```
13 CB C3              JMP
13 CC C3              NXTLAB
13 CD 13              0
13 CE D3 REGLST,      323          /S
13 CF DA              332          /Z
13 D0 01              001          /1 个空格
13 D1 B1              261          /1
13 D2 01              001          /1 个空格
13 D3 D0              320          /P
13 D4 01              001          /1 个空格
13 D5 B2              262          /2(D3 向 D4 的进位位)
13 D6 01              001          /1 个空格
13 D7 C1              301          /A
13 D8 02              002          /2 个空格
13 D9 C2              302          /B
13 DA 02              002          /2 个空格
13 DB C3              303          /C
13 DC 02              002          /2 个空格
13 DD C4              304          /D
13 DE 02              002          /2 个空格
13 DF C5              305          /E
13 E0 02              002          /2 个空格
13 E1 C8              310          /H
13 E2 02              002          /2 个空格
13 E3 CC              314          /L
13 E4 02              002          /2 个空格
13 E5 CD              315          /M
13 E6 03              003          /3 个空格
13 E7 D3              323          /S
13 E8 01              001          /1 个空格
13 E9 D0              320          /P
13 EA 03              003          /3 个空格
13 EB C3              303          /C
13 EC 01              001          /1 个空格
13 ED D3              323          /S
13 EE 8D              215          /CR
13 EF 8A              212          /LF
13 F0 00              000          /信息终止
13 F1 C3 BUFF2,       JMP          /这就是被复制到 RAM 中的内容，RAM 的开始
```

```
13 F2 1A          ENTRY      地址为 000 070(hex 00 38)
13 F3 12          0
13 F4 C3          JMP        /就从这里转到 03 C7
13 E5 D3          EXECUT
13 F6 12          0
13 F7 01          001        /这就是"LAB"的值
13 F8 80          200
13 F9 03          003
```

AIN	=10 4D	BRKADD	=03 C0	BYTE	=10 58	BYTE1	=10 5B
BYTOUT	=10 A9	BCS	=10 B8	BREAK	=11 FE	BRK	=12 04
BREAK1	=12 0B	BIT	=13 87	BUFF2	=13 F1	CHKFRM	=10 38
CNT	=11 24	CRLF	=11 3D	CONTIN	=13 3C	CNTIN1	=13 48
ENTRY	=12 1A	EXECUT	=12 D3	FLAG	=03 CF	HEXIN	=11 52
HEX09	=11 6F	HLOUT	=11 97	HEXOUT	=11 9C	HEX	=11 A6
HLSLSH	=11 EA	IMMSL1	=12 94	IMMED	=12 9D	IMMSL2	=12 B3
LAB	=03 CA	LDDR	=10 9D	LIST	=11 80	LABEL	=13 B2
MORSPC	=11 F4	MOR1	=13 0C	NEWADD	=03 C2	NOEND	=10 42
NXTIN	=10 47	NOREAD	=10 63	NXTOUT	=10 8C	NOPUN	=10 B7
NOBCS	=10 D9	NOGO	=10 F5	NXTLOC	=11 07	NXTLN	=11 1A
NXTHEX	=11 56	NMB09	=11 B6	NO300	=12 5E	NOGOOD	=12 A7
NXTBIT	=13 89	NXTLET	=13 B7	NXTLAB	=13 C3	ONEMOR	=12 BE
ONEOUT	=13 91	ONE	=13 9B	PUNCH	=10 68	QUEST	=10 1C
QUEST2	=10 25	QUEST3	=10 28	RDR	=10 30	READ	=11 BC
RSTRE	=13 7F	REGLST	=13 CE	STACK	=03 FE	START	=10 00
SPCHEX	=11 47	SPC	=11 F2	SENTRY	=12 25	SING	=12 88
STEP	=13 5C	SETUP	=13 A7	SPCIT	=13 C7	TMP	=03 C4
TEMPO	=03 CD	TWOHEX	=11 4E	TTY1	=11 CF	TTYIN	=11 D9
TTYOUT	=11 DE	THREBY	=12 B9	TWOMOR	=12 BD	TWOTIM	=13 08
USERSK	=03 CB	VECT	=00 38	ZEROIT	=13 A0		

ERRORS DETECTED = 000