实验一 Debug.exe使用

## 实验目的

1．学习Debug软件的基本使用方法。

2．掌握8086/8088的寻址方式。

3. 掌握数据传送、算术运算逻辑运算等指令的基本操作。

## 实验原理

debug是Dos下自带的程序，是调试计算机语言程序的工具，它不仅能跟踪程序的运行，还可显示修改内存与寄存器内容，以便在调试程序中检查程序，找错，改错；它还支持小汇编，直接对小段程序进行汇编；也可对盘进行读写操作。但debug不宜汇编较长的程序，不便于分块程序的设计，不能使用绝大多数ASM和MASM提供的伪指令。

debug的进入是DOS状态下键入debug文件名，便进入debug状态，在debug的提示符“—”下键入debug命令。

C>debug ↙

debug命令以单个字母的命令符开头，然后是命令操作参数，操作参数与操作参数之间，用空格或逗号分开，操作参数与命令符之间用空格隔开，命令的结束符是回车键↙。命令及参数的输入可以是大小写的结合，CtrL +break键中止命令的执行。debug下所用的数均为十六进制数，且不必写H，指令也无必要一定大写。

debug的常用命令有：

1．汇编命令A

格式：(1) A<段寄存器名>：<位移>

(2) A<段地址>：<位移>

(3) A<位移>

(4) A

功能：键入该命令后显示段地址和位移并等待用户从键盘逐条键入汇编命令，逐条汇编成代码指令，直到显示下一地址时用户直接键入回车键返回到提示符“—”。其中(1)用指定寄存器内容作段地址，(3)用CS内容作段地址，(4)用CS:0100作地址。

以后其它命令中提及的各种“地址”形式，均为(1)、(2)、(3)中A后的地址形式。

2．反汇编命令U

格式1：U <地址>

格式2：U <地址范围>

功能：将指定范围内的代码以汇编语句形式显示，同时显示地址及代码。注意，反汇编时一定确认指令的起始地址后再作，否则将得不到正确结果。地址及范围的缺省值是上次U指令后下一地址的值。这样可以连续反汇编。

3．显示内存命令

格式：(1) D<地址>；以CS为段寄存器

(2) D<地址范围>

(3) D;显示CS:0100起始的一片内容。

功能：以两种形式显示指定内存范围内容。一种形式为十六进制内容，一种形式为相应字节的ASCII码字符，对不可见字符以“·”代替。

4．修改内存命令E

格式：(1)　Ｅ<地址><单元内容>

(2) E<地址><单元内容表>

功能：(1)将指定内容写入指定单元后显示下一地址，可连续键入修改内容，直到新地址出现后键入↙为止。

(2)将<单元内容表>逐一写入由<地址>开始的一片单元。

5．填充内存命令F

格式：F <地址范围> <单元内容表>

功能：将单元内容表中的值逐个填入指定范围，单元内容表中内容用完后重复使用。

例如：—F 12FF：0000 0100 A↙

—D 12FF：0000 0100 ↙

6．显示并修改寄存器命令R

格式1:R

格式2:R<寄存器名>

功能：格式1显示当前所有寄存器内容，状态标志及将要执行的下一指令的地址，代码及汇编语句格式。其中对状态标志FLAG以每位的形式显示。格式2显示指定寄存器内容。

例如：R IP ↙

7．单步执行并显示系统环境T

格式：T[=<地址>][<条数>]

功能：<地址>的缺省值是当前IP值，<条数>的缺省值是一条。执行由指定地址起始的<条数>指定的若干条指令。

例如：—T；执行当前指令并显示状态

—T10；从当前指令始执行10H条指令。

8．连续执行命令Ｇ

格式：(1)G=<起始地址> <断点地址>

(2)G<结束地址>；从IP指向的地址开始执行程序到结束地址。

9．退出debug命令Q

—Q ↙ ；进入DOS状态下

C>

例如：利用串操作传送指令将0FFH送入附加段的0180H开始的128个字节存储单元中，编写一段程序。

1．进入DEBUG状态

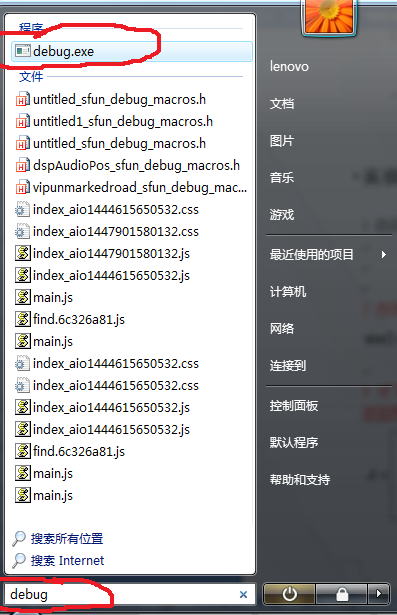
C>DEBUG ↙

—

## 实验内容

1 启动debug.exe，可如下图，点击debug.exe即可启动。如上述方法不能出现debug，到系统盘中直接搜索debug文件，仍然不行，参照

<http://jingyan.baidu.com/album/86f4a73e58413137d7526970.html?picindex=6>



2 用A命令编辑源程序，连续按回车键可退出指令输入

一A ↙

××××：0100 MOV DI，0180 ↙

0103 MOV CX, 80 ↙

0106 MOV AL, FF ↙

0108 REP STOSB ↙

010A HLT ↙

010B ↙

3.用P命令查看各寄存器

4.用U命令反汇编以上源程序

一U CS:0100 ↙

××××：0100 BF8001 MOV DI,0180

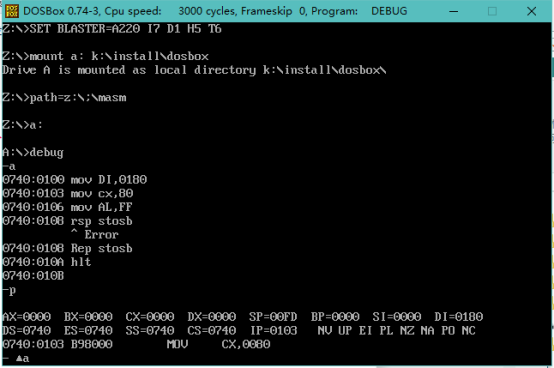
××××：0103 B98000 MOV CX,0080

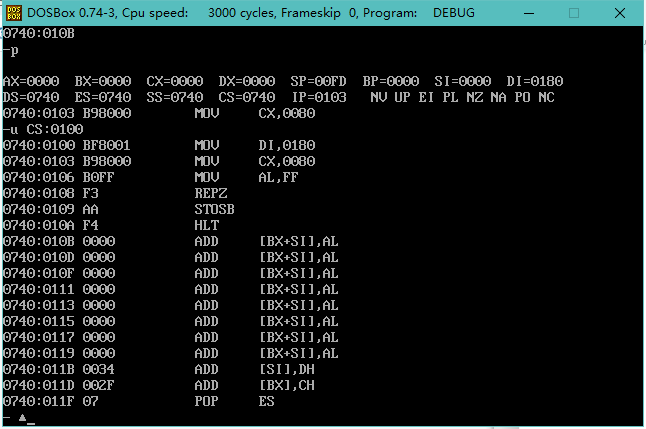
××××：0106 BOFF MOV AL,FF

××××：0108 F3 REPE

××××：0109 AA STOSB

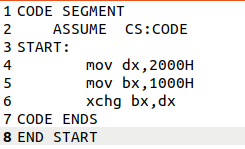
××××：010A F4 HLT





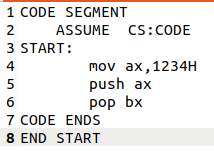
1. 完成书后第三章7

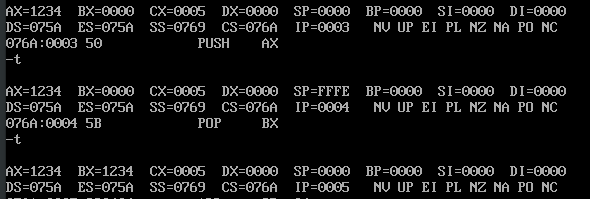
（1）

xchg是exchange，交换



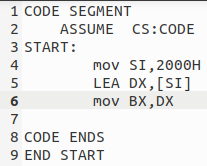
（2）

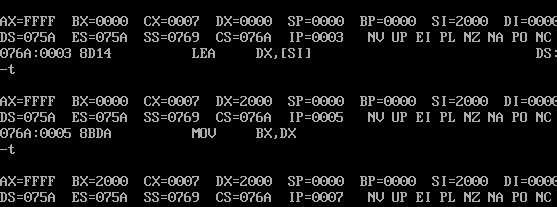
ax值入栈，弹出给bx



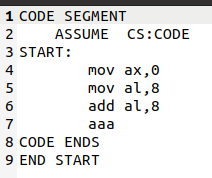
（3）

LEA 是 Load Effective Address 的缩写,这里似乎不能使用直接寻址，因此我使用了SI来进行间接寻址

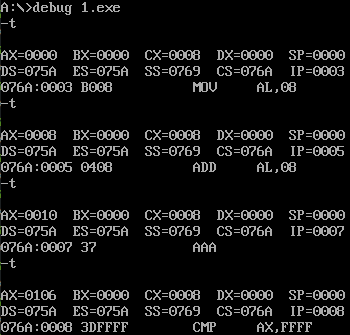




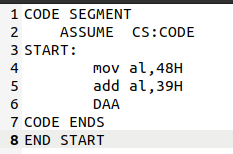
（4）



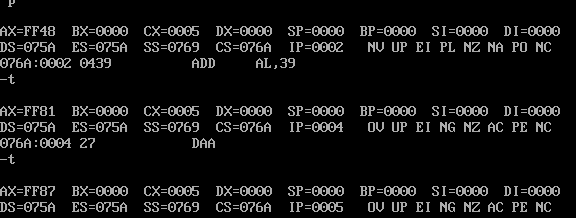
将10进制数8+8赋给AL，因此为16进制数10。AAA，ASCII Adjust for Addition，一般在加法后使用，将相加结果转化成非压缩BCD格式（AH存十位，AL存个位）。比如16将记录成0106。但是，AH的产生与进位符号CF有关，最多只会置位成01。同时，AL只会保留后四位。因此，如果计算的数值已经超过010F，也就是超过25的话，那么计算的结果会使得依然保持AH有01进位，但是AL的结果实际取余16（保留后四位）



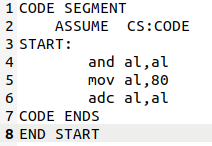
（5）



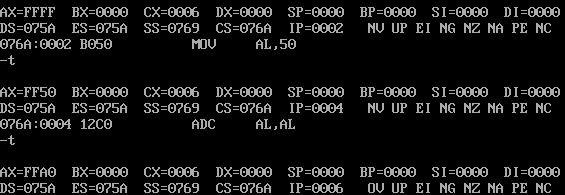
16进1，因此是81H。DAA将这个原来的数视为十进制计算，即48+39=87，然后由于AL只有2位16进制位，因此DAA后只会保留后两位即十位个位



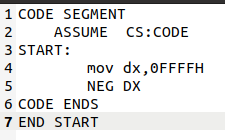
（6）

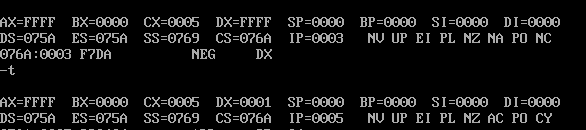


ADC带进位加法，如果CF位有置1，那么这里将加1，这里没有，因此是直接相加得A0

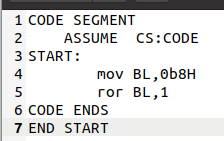


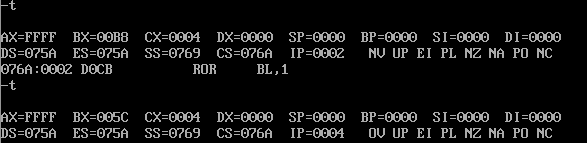
（7）

补码，等于反码+1

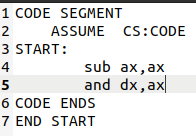


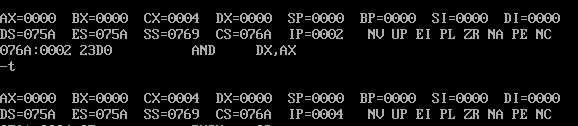
（8）

右移1位，除2

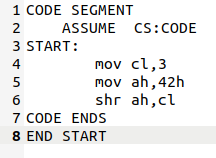


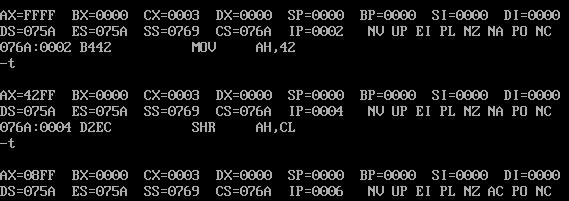
（9）



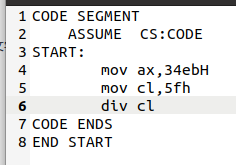


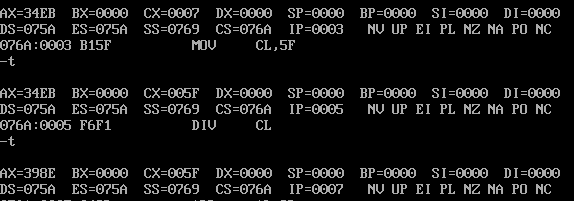
（10）

42右移3位，42是01000010，右移剩1000

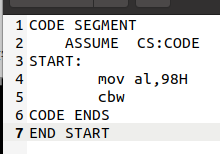


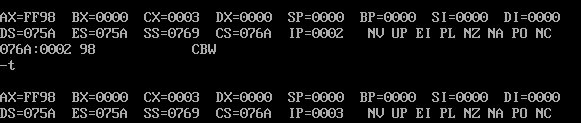
（11）

除数在AL里，余数在AH

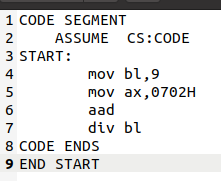


（12）

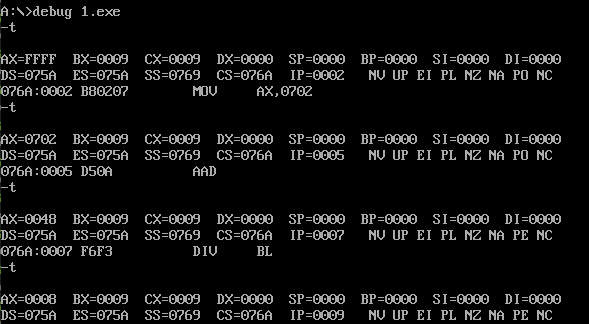
CBW，如果al的最高位是负数，那么AX赋值FF，否则赋值00



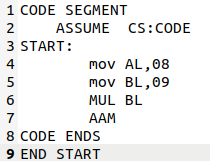
（13）



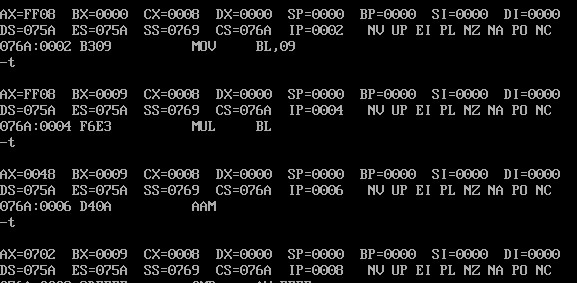
aad将非压缩BCD码十进制72，转化回去十六进制数48



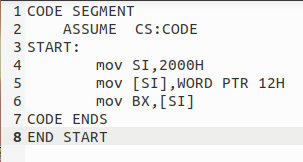
（14）



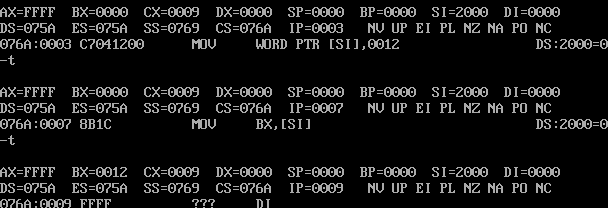
aam与aaa相似，一般在乘法后面使用



（15）

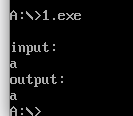


这里使用WORD PTR将12H当做16位的数字，因此可以直接赋值给BX

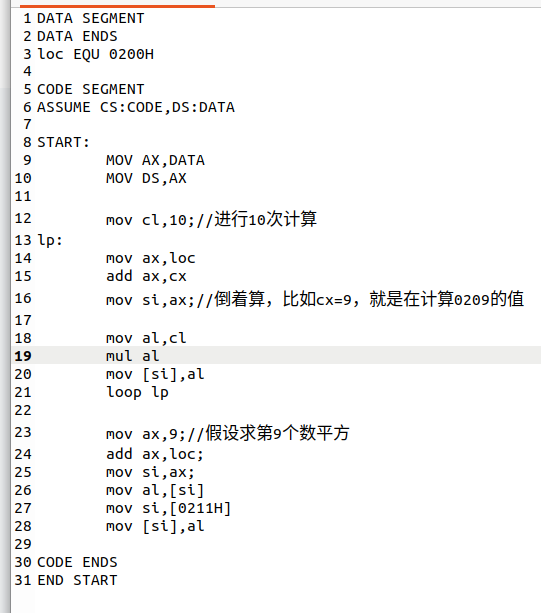


6 显示一个字符‘Ａ’；从键盘上读取一个字符并显示





7. 在debug中编写一段代码，在内存中自0200Ｈ地址开始的连续十个单元中存入0～9的平方值。要求利用计算法查表求0210Ｈ单元中指定数0～9中的任意一数的平方值，并将所求平方值存入0211Ｈ单元中。



对应第23行，al=51H=81，是9的平方

