《汇编语言程序设计》

实验指导书 V1.0

熊迎军 张俊艺 严一杏

信息科学技术学院

2016 年 7 月

目　录

第一部分 基本知识

第二部分 上机实验

第一部分 基本知识

这部分是关于汇编实验的一些准备性的知识，包含了实验要求与实验报告书写规范，汇编源程序的编译和连接方法，并详细介绍了汇编程序的调试方法。

另外，因为汇编程序主要是在 MS-DOS 方式下工作，对 MS-DOS 的一些基本命令也作了简要说明。

一　实验基本知识

1.实验要求

1）上机前，必须做好实验的预习工作；

2）遵守实验室的各项规章制度，爱护实验室设备；

3）以RAR压缩文件形式提交，其中内容由实验报告和实验代码以及其它相关文件组成，具体文件命名规则如下：

（1）压缩文件名称为：学号__N.rar（N代表第几次实验）。

（2）实验报告以及其它非代码文件命名：学号__姓名__文件名

（3）实验代码名称为：学号__N__M.asm（M代表本次实验的第几个代码）

4）实验代码必须有详细注释，代码开头必须以注释形式写名实验题目或者实验要求。

5）每节实验课结束之前完成实验代码调试，课后写出实验报告，不得抄袭。

6）实验报告参考格式：

（1）实验目的：描述实验所要解决的问题及要求。

（2）设计说明：说明程序的功能、结构、算法、设计思路以及模块功能与变量设计等。

（3）程序流程图：以流程图形式描述实验程序结构与流程。

（4）实验结果：实验程序的运行结果截图。

（5）心得体会：遇到的问题及其解决办法和实验收获。

2.实验方法

1）用文本编辑器（本课程推荐使用NotePad++）编辑程序，保存为汇编语言文件（.asm后缀文件）；

2）使用汇编工具（如MASM.EXE）将ASM文件翻译成目标代码文件（.obj后缀文件）；

3）使用链接工具（如LINK.EXE）连接目标代码程序与库函数代码生成可执行程序文件（.exe后缀文件）；

4）使用 DEBUG.EXE 调试或直接运行；

5）操作方法（首先要 CMD 界面进入代码所在路径）：

```
masm myfile.asm
link myfile.obj
debug myfile.exe
```

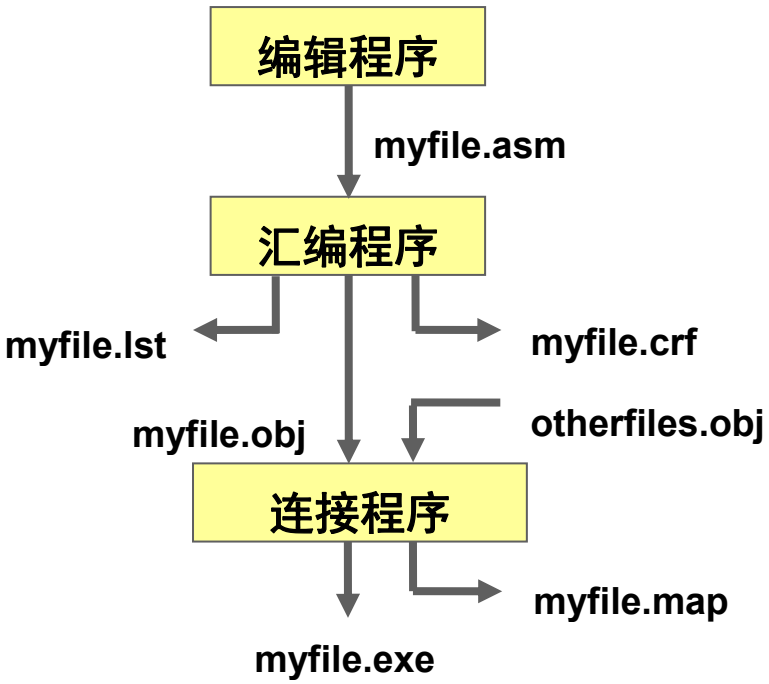
3. 汇编的功能

1）检查源程序，给出错误信息；

2）生成目标程序，并给出列表文件；

3）展开宏指令。

4. 汇编的上机过程



1）建立工作环境

为运行汇编程序，需要如下程序：

MASM.EXE

LINK.EXE

DEBUG.EXE 等

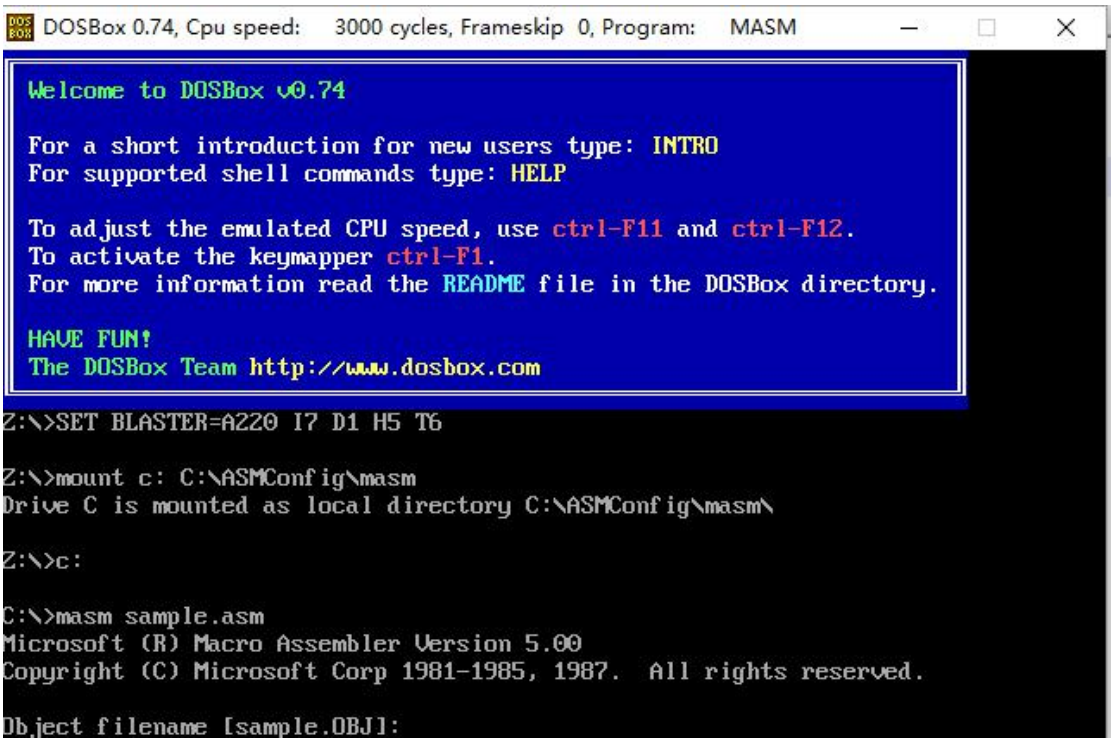
2）建立汇编文件

例：在屏幕上输出 This is a masm sample.

使用 NotePad++编辑如下程序代码，并保存为 sample.asm。

```
DATA SEGMENT
    STR DB  'This is a masm sample.', 0DH, 0AH,  '$'
DATA ENDS
CODE SEGMENT
    ASSUME CS: CODE, DS: DATA
START:
    MOV AX, DATA
    MOV DS, AX
    LEA DX, STR
    MOV AH, 09H
    INT 21H
    MOV AH, 4CH
    INT 21H
CODE ENDS
    END START
```

（1） 使用汇编工具生成 OBJ 文件



c:\>masm sample.asm

Microsoft (R) Macro Assenbler Version 5.00

Copyright (C) Microsoft Corp 1981-1985,1987……（软件提示）

Object filename [sample.OBJ]:（提示输入目标文件名，可省略）

Source listing [NUL.LST]:（提示输入列表文件名，可省略）

Cross-reference [NUL.CRF]:

Warning Severe

注意：若汇编的错误提示不为 0，则需重新调用 sample.asm 修改错误，然后再汇编，直到错误提示为 0。

（2） 使用链接工具生成 EXE 文件

c:\>link sample.obj

……

Run File [sample.EXE]:（提示输入可执行文件名，可省略）

List File [NUL.MAP]:（提示输入连接映像文件，可省略）

Libraries [.LIB]:（提示输入文件用到的库文件名，可省略）

……

（3） 执行 EXE 文件

c:\>sample.exe 或 debug sample.exe

二 DEBUG 简介

DEBUG.EXE 是 DOS 提供的用于调试可执行程序的工具软件，是汇编语言程序设计中常用的调试工具。在 DEBUG 环境下，不但可以调试经汇编、连接后生成的可执行程序，也可以编写简单的程序。

但是，DEBUG 只是调试工具，不是汇编工具，不能将 ASM 源文件汇编连接生成可执行程序。DEBUG 带有内置的汇编功能，但功能极弱，仅支持指令的汇编，不支持大多数常用伪指令的汇编。常用的汇编连接工具是 Microsoft 公司开发的 MASM。MASM 需要至少两个文件：汇编程序 MASM.EXE 和连接程序 LINK.EXE ，本课程推荐版本 MASM5.0。

1.DEBUG 调用

DEBUG 程序启动方法：

在 DOS 提示符下键入：DEBUG [路径\]文件名

DEBUG 执行，并将指定文件装入内存，供调试。此时屏幕显示提示符“-”，说明计算机当前处于 DEBUG 的管理之下，可接受、并执行 DEBUG 命令。

2.DEBUG 主要命令

DEBUG 命令是在 DEBUG 提示符“-”下由键盘键入的。每条命令以单个字母的命令符开头，然后是命令的操作参数。DEBUG 主要命令如下：

1.）汇编命令 A（Assemble）

- A [address]

功能：该命令允许键入汇编语言的语句，并能把它们汇编成机器代码，相继地存放在从指定地址开始的存储区中。

说明：

①输入程序时，以回车结束一行语句的输入，同时提示下一行语句的起始地址。如：

```
-a
0B27:0100 mov ax,ffff
0B27:0103
```

当程序输入结束时，在提示的地址后面键入回车结束汇编操作，返回 DEBUG 提示符。

②在 DEBUG 下键入的数字均看成十六进制数，不能键入十进制数。

③命令中提供地址的形式有三种：

◇段地址:偏移地址

◇段寄存器:偏移地址

◇偏地址

如果不给出段地址，是用 CS 的值作为段地址；如果不提供存储地址，第一次键 a 命令

是以 CS:0100 作为地址，以后则以前次汇编结束时的提示地址为新的起始地址。

2.）显示存储单元的命令 D（Dump）

- D[address] 或 - D[range]

功能：以两种形式显示指定内存范围的内容。一种为 16 进制形式的内容，一种形式为相应字节的 ASCII 码，对不可见字符以“.”代替。

例如，按指定范围显示存储单元内容的方法为：

-d100 120

0B25:0100 30 31 32 34 35 36 37 38-39 F1 47 FE C4 EB EC 4F 012456789.G....O

0B25:0110 A0 B7 96 C6 46 00 02 0A-E4 75 05 3A 34 00 14 0B ....F....u.:4...

0B25:0120 AA .

DEBUG 显示了 0100 到 0120 之间各单元的内容。 左边用十六进制表示每个字节，右边用 ASCII 字符表示每个字节，“.”表示不可显示的字符。

注意：如果没有指定段地址，D 命令自动显示 DS 段的内容。如果只指定首地址，则显示从首地址开始的 80 个字节的内容。如果完全没有指定地址，则显示上一个 D 命令的最后一个单元的内容。

3.）修改存储单元内容的命令 E（Enter）

输入命令 E，有两种格式如下：

第一种格式可以用给定的内容表替代指定范围的存储单元内容。命令格式：

- E address [list]

其中 list 为用空格作为分隔符的字节数据表。

功能：将[list]的内容写入 address 为起始地址的一片存储单元。

例如，-E DS:100 （段地址：偏移地址） F3'XYZ'8D

其中 F3，‘X’，‘Y’，‘Z’和 8D 各占一个字节，该命令可以用这五个字节来替代存储单元 DS：0100 到 0104 的原先的内容。

第二种格式则是采用逐个单元相继修改的方法。命令格式：

- E address

功能：显示 address 指定的存储单元内容，等待用户输入更新值，输入修改值后按空格后，又显示下一单元的内容，并等待用户输入更新值，……，可以连续修改多个连续存储单元的值，回车结束该命令的执行。

例如，-E CS:100

10

则可能显示为：

18E4：0100 89. -

如果需要把该单元的内容修改为 78，则可以直接键入 78，再按空格键可接着显示下一个单元的内容，这样可以不断修改相继单元的内容，直到 Enter 键结束该命令为止。（如果不修改当前值则直接按 Enter）

4.）填写内存单元命令 F（Fill）

- F range list

功能：将 list 中的内容逐字节填入指定的地址范围，list 中的内容使用完后会自动重复使用。

例如，-F 4BA:0100 5 F3'XYZ'8D

使 04BA:0100-0104 单元包含指定的五个字节的内容。如果 list 中的字节数超过指定的范围，则忽略超过的项；如果 list 的字节数小于指定范围，则重复使用 list 填入，直到填满指定的所有单元为止。

5.）执行程序命令 G（Go）

- G [=address1][address2[address3 ...]]

功能：从指定地址开始执行程序。 其中，address1 指定了运行的起始地址，如不指定则从当前的 CS:IP 开始运行。后面的地址均为断点地址，当指令执行到断点时，就停止执行并显示当前所有寄存器及标志位的内容，和下一条将要执行的指令及存放地址。

6.）逐行跟踪程序命令 P（Proceed）

- P[=address] [number]

功能：功能同下面介绍的命令 T。不同的是，当 P 命令执行的是 CALL 或 INT n 指令时，将一次执行完整个子程序或中断处理程序，通过寄存器返回执行的结果，并提示要执行的下一条指令。

7.）退出 DEBUG 命令 Q（Quit）

- Q

功能：退出 DEBUG，返回命令提示符。

8.）检查和和修改寄存器内容的命令 R（Register）

该命令有三种不同的格式：

（1）显示 CPU 内所有寄存器内容和标志位状态，其格式为：-R

例如，

-r

AX=0000 BX=0000 CX=010A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000

DS=18E4 ES=18E4 SS=18E4 CS=18E4 IP=0100 NV UP DI PL NZ NA PO NC

18E4：0100 C70604023801

MOV WORD PTR[0204], 0138

DS:0204=0000

（2）显示和修改某个寄存器内容，其格式为：-R register_name

例如，键入

-r ax

系统将响应如下：

AX 0000

:

即 AX 寄存器的当前内容为 0000，如不修改则按 Enter 键，否则，键入欲修改的内容如。

-r bx

BX 0000

：059F

则把 BX 寄存器的当前内容修改为 059F。

（3）显示和修改标志位状态，命令格式为：-RF

系统将响应，如：

OV DN EI NG ZR AC PE CY -

此时如不修改其内容可按 Enter 键，否则，建入欲修改的内容，如：OV DN EI NG ZR AC PE CY - PONZDINV

即可，键入的顺序是任意的。

9.）逐指令跟踪程序命令 T（Trace）

跟踪命令 T 有两种格式：

（1）逐条指令跟踪

-T[=address]

功能：从指定地址起执行一条指令后停下来，显示所有寄存器内空及标志位的值。如未指定则从当前的 CS:IP 开始执行。

（2）多条指令跟踪

-T[=address][value]

功能：从指定地址 address 起执行 value 条指令后停下来。若不给出地址 address，则执行 CS:IP 指定的指令；若没有提供要执行的指令条数，则只执行一条指令。

注意：

①若 T 命令执行的是 CALL 或 INT n 指令时，将跟踪到子程序或中断服务程序内部，逐条执行指令。

②在开始跟踪执行程序的第一条指令时，必须给出程序第一条指令的地址，若继续执行后继指令只需要简单的输入命令 T。

10.）反汇编命令 U（Unassemble）

反汇编是指对内存指定区域的内容，以汇编语句形式显示，同时显示地址和相应的机器码。反汇编命令 U 有两种格式。

（1）从指定地址开始，反汇编 32 个字节，其格式为：-U[address]

如果地址被省略则从上一个 U 命令的最后一条指令的下一个单元开始显示 32 个字节（是汇编指令实际占用的字节数）反汇编的结果。

（2）对指定范围内的存储单元进行反汇编，格式为：-U[range]

说明：在进行反汇编操作时，一定要确认指令的起始地址后，在操作，否则将得不到正确的结果。连续进行反汇编操作时，可以省略地址，DEBUG 自动以上一 U 命令操作结束后的下一地址为反汇编的起始地址。

三 MS-DOS 方式命令简介

MS-DOS 方式是 Windows 提供的一个字符界面的 shell 窗口，通过“开始>程序>MS-DOS 方式”进入 MS-DOS 方式后，可以按下 Alt+Enter 获得全屏幕的窗口，在该窗口下，主要通过键入命令和观察结果获得交互信息。

1.命令简介

1.）DIR 命令

显示当前目录下的文件和目录，格式为：dir

2.）CD 命令

改变当前工作目录。格式为：cd d:\student\myfile

3.)MD 命令

建立一个新的目录，格式为：md myfile

4.）DEL 命令

删除文件，格式为：del myfile.asm

5.）RD 命令

删除目录，要求需删除的目录下无任何目录或文件，格式为 rd myfile

6.）COPY 命令

复制命令，格式为：copy myfile.asm myfile2.org（将文件 myfile.asm 复制为 myfile2.org）

第二部分 上机实验

注意事项：

（1） 每次实验前，需要详细阅读实验目的、实验要求和实验提示，以便能够准确的理解实验要求达到实验目的。有测试数据要求的，需要给出测试结果，有要求回答问题的，需要给出问题的答案。

（2） 建议使用 MS-DOS 方式进入 DOS 平台。

（3） 如使用 DOSBOX 虚拟平台，并且 MASM.EXE 等汇编工具在 d:\masm5 目录下，则在 DOSBOX 平台输入如下命令：

```
Z:\>mount c d:\masm5
Z:\>c:
C:\>
```

（4） 调试程序的步骤方法：

设程序文件名为 myfile.exe，则：

```
path >debug myfile.exe
-T                    （单步执行）
```

实验一　寻址方式

1. 实验目的

（1）掌握操作数的不同寻址方式；

（2）熟练应用 DEBUG 调试汇编程序。

2. 实验原理

计算机中的指令由操作码字段和操作数字段两部分组成，8086 中与数据有关的寻址方式一共有 7 种，用来确定操作数地址从而找到操作数。

（1）立即寻址方式操作数直接存放在指令中，紧跟在操作码之后，它作为指令的一部分存放在代码段里；

（2）寄存器寻址方式使用寄存器来存放要处理的操作数；

（3）其它 5 种寻址方式：直接寻址方式、寄存器间接方式、寄存器相对寻址方式、基址变址寻址方式、相对基址变址寻址方式，操作数都在除代码段以外的存储区中， 在 8086 里，把操作数的偏移地址称为有效地址 EA，这五种寻址方式分别对应五种计算 EA 的方法。

有效地址可以由以下三种成分组成：

a. 位移量（Displacement）是存放在指令中的一个 8 位或 16 位数，但它不是立即数，而是一个地址。

b. 基址（Base）是存放在基址寄存器（BX 或 BP）中的内容。它是有效地址中的基址部分，通常用来指向数据段中数组或字符串的首地址。

c. 变址（Index）是存放在变址寄存器（SI 或 DI）中的内容。它通常用来访问数组中的某个元素或字符串中的某个字符。

这三种成分都可正可负，以保证指针移动的灵活性。它们任意组合使用，可得到不同的寻址方式。

（5） 伪操作 DB、DW、DD 分别用来定义字节、字和双字变量。

3. 实验内容

（1）立即数寻址

代码段：MOV AX,1000

```
-
-a 100
073F:0100 mov ax,1000
073F:0103
-t=0100

AX=1000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 0000          ADD     [BX+SI],AL                      DS:0000=CD
-r
AX=1000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 0000          ADD     [BX+SI],AL                      DS:0000=CD
```

（2）直接寻址

代码段：MOV AX,1200

MOV [1000],AX

MOV BX,[1000]

注意：默认段位数据段（即[1000] == DS:[1000]）

```
-a 100
073F:0100 mov ax,1200
073F:0103 mov [1000],ax
073F:0106 mov bx,[1000]
073F:010A
-t=0100

AX=1200  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 A30010        MOV     [1000],AX                       DS:1000=1200
-t

AX=1200  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 8B1E0010      MOV     BX,[1000]                       DS:1000=1200
-t

AX=1200  BX=1200  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI PL NZ NA PO NC
073F:010A B44C          MOV     AH,4C
```

（3）间接寻址

代码段：MOV AX,1200

MOV [1000],AX

MOV AX,0000

MOV BX,1000

MOV AX,[BX]

注意：

a.[R]中的 R 只能是通用寄存器 BX,DI,SI,BP,其余的寄存器不能使用寄存器的间接寻址法；

b.默认段仍然是数据段，但允许进行段超越。（例如：SS:[BX],即将 SS 段当中的数据进行转移）

```
-a 100
073F:0100 mov ax,1200
073F:0103 mov [1000],ax
073F:0106 mov ax,0000
073F:0109 mov bx,1000
073F:010C mov ax,[bx]
073F:010E
-t=0100

AX=1200  BX=1200  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 A30010        MOV     [1000],AX                       DS:1000=1200
-t

AX=1200  BX=1200  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 B80000        MOV     AX,0000
-t

AX=0000  BX=1200  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0109   NV UP EI PL NZ NA PO NC
073F:0109 BB0010        MOV     BX,1000
```

```
AX=0000  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010C   NV UP EI PL NZ NA PO NC
073F:010C 8B07          MOV     AX,[BX]                         DS:1000=1200
-t

AX=1200  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010E   NV UP EI PL NZ NA PO NC
073F:010E 0000          ADD     [BX+SI],AL                      DS:1000=00
```

（4）变址寻址

代码段:MOV AX,1200

MOV [1002],AX

MOV AX,0000

MOV BX,1000

MOV AX,2[BX]

```
-a 100
073F:0100 mov ax,1200
073F:0103 mov [1002],ax
073F:0106 mov ax,0000
073F:0109 mov bx,1000
073F:010C mov ax,2[bx]
073F:010F
-t=0100
```

```
AX=0000  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010C   NV UP EI PL NZ NA PO NC
073F:010C 8B4702        MOV     AX,[BX+02]                      DS:1002=1200
-t

AX=1200  BX=1000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010F   NV UP EI PL NZ NA PO NC
073F:010F 0000          ADD     [BX+SI],AL                      DS:1000=00
```

（5）基址加变址寻址

基址寄存器：BX、BP　变址寄存器：SI、DI

代码段：MOV AX,1200

MOV [1002],AX

MOV AX,0000

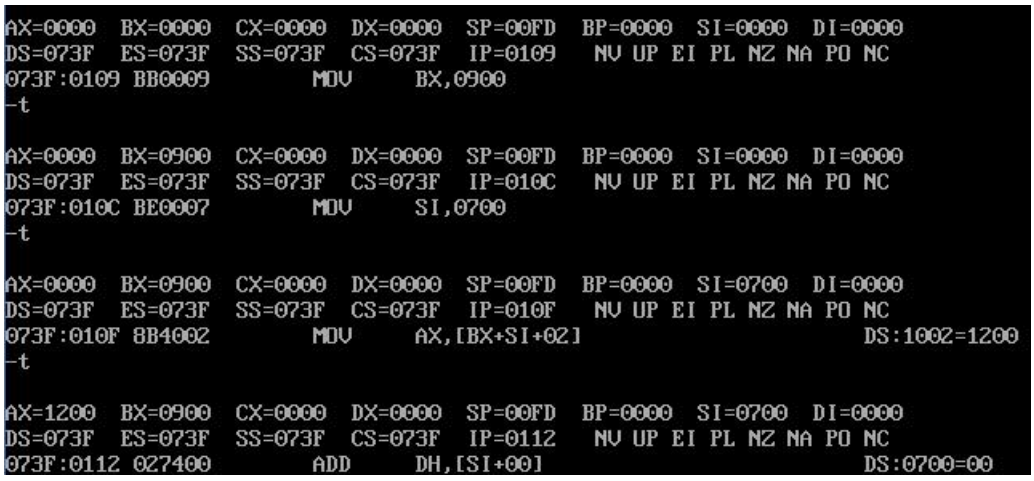
MOV BX,900

MOV SI,700

MOV AX,2[BX][SI]

注意：基址加变址寻址方式主要运用于对二维表格的操作

```
073F:0100 mov ax,1200
073F:0103 mov [1002],ax
073F:0106 mov ax,0000
073F:0109 mov bx,0900
073F:010C mov si,0700
073F:010F mov ax,2[bx][si]
073F:0112
```



4. 实验要求

（1）上机实验前，仔细复习课本有关知识；

（2）独立完成实验，并上交实验报告；

实验二　串操作指令的使用

1. 实验目的

（1）掌握串操作指令的使用；

（2）理解算数运算指令、BCD 码调整指令；

（3）熟练应用 DEBUG 调试汇编程序；

2. 实验内容

题目：使用串操作指令 MOVSB 对一段内存单元中的内容（1，2，3，……，100）进行转移，再使用串操作指令 CMPS 对转移的内容进行比较来判断传输是否正确，若不正确则进行重新传输；接着对已经正确传输的 100 个数据进行无符号型的累加，最后使用 BCD 调整码，最终将答案放入内存，并将其显示在屏幕上。

3. 实验要求

（1）上机实验前，仔细复习课本有关知识；

（2）独立完成实验，画出流程图并上交实验报告；

4. 实验步骤

1.）算法分析

从实验的内容分析可知，要完成如下实验，可分为以下步骤：

（1）将 1，2，3，……，100 存入数据段相应内存中；

（2）转移字符串并比较；

（3）数据累加并调整。

2.）算法设计

（1）数据的存入、转移与比较

已知：

a. MOVSB 指令的目标操作数与源操作数的逻辑地址由 ES:DI 和 DS:SI 指出；

b. 串传送指令常与无条件重复前缀连用；

c. 无条件重复 REP，仅仅判断 CX 是否为 0；

d. 串比较指令常与条件重复前缀连用，指令的执行不改变操作数，仅影响标志位。

注意：在使用串操作指令时需要修改 flag 寄存器当中的 DF 位（方向位），来确定串操作的进行方向，具体表现为：CLD 使 DF=0 增地址方向；STD 使 DF=1 减地址方向；

（2）数据的累加与调整

BCD 码调整指令 AAM 用来调整寄存器 AX 当中的值，将 AL/10 的商放在 AH 高位中，余数放在 AL 低位当中进行保存。

将结果答案显示到屏幕上时，需要的是数字的 ASCII 码，因此需要 ADD AX,3030H。

3.）完成实验报告

实验三　转移指令的使用

1.实验目的

（1）了解掌握汇编程序中指令跳转的实现；

（2）掌握算法的设计及表达方式；

（3）熟练应用 DEBUG 调试汇编程序。

2.实验原理

1.）顺序执行的指令地址是由指令指针寄存器 IP 自动增量形成的，而程序转移的地址必须由转移类指令和 CALL 指令指出，表示转向地址的寻址方式包括：段内直接寻址、段内间接寻址、段间直接寻址、段间间接寻址。

2.）与转移地址相关的有三个表示转移距离（称为位移量）的操作符：SHORT、NEAR、FAR。

（1）SHORT，短转移，表示位移量在-128~127 字节之间。

（2）NEAR，近转移，表示在同一段内转移，位移量在-32768~32767 字节范围内。

（3）FAR，远转移，表示转移距离超过±32K 字节，在不同段之间转移。

3.)CS:IP 寄存器总是指向下一条将要执行的指令的首地址(称为 IP 当前值)，当转移指令执行后，必须修改 IP 或 CS、IP 的值。

（1）SHORT 转移和 NEAR 转移，位移量分别用 8 位和 16 位表示，程序控制仍然在当前代码段，所以只修改 IP 的值，CS 的值不变。

（2）FAR 转移，程序控制超出了当前代码段，所以 CS 和 IP 都必须修改为新的值。

4.）与转移地址有关的 4 种寻址方式就是告诉 CPU 如何修改 CS 和 IP 的值，以达到控制程序转移的目的。

3.实验内容

题目：计算 X-|Y|+|Z|，并将计算结果输出显示。

4.实验要求

（1）上机实验前，仔细复习课本有关知识；

（2）独立完成实验，画出流程图并上交实验报告；

（3）尝试手动输入未知数 X、Y、Z。

4.实验步骤

1.）算法分析

从实验的内容分析可知，要完成如下实验，可分为以下步骤：

（1）接受键盘输入 X、Y、Z；

（2）计算|Y|、|Z|；

（3）计算结果并输出显示。

2.）算法设计

（1）输入未知数

a. 使用 MOV AH,01H INT 21H 实行对字符串的输入，而事实上此功能一次只能输入一个字符，因此使用循环，将每次输入的值，保存在内存当中，当检测到回车字符（ODH）时，即结束当前的循环；

b. 负数的输入情况，判断是否有‘-’符号的输入，若有的话，则在最后的数取其补码，一个 AX16 位长，每 4 位表示一个数，因此目前最多只能表示 4 位数以内的加减法运算。

（2）计算绝对值

首先使用 TEST AX,8000H，来判断 AX 的最高位符号位是 1 还是 0，若是 1 的话则需要 NEG AX,取 AX 的补码，进行运算，TEST 的实质是不保存结果的 AND 运算；

（3）计算结果并输出

a. 最后的计算结果使用 MOV AH,09H INT 21H 进行输出，使用 AAM 等 BCD 转换指令将 16 进制转换为 10 进制；

b. 判断最后结果的正负问题，若结果为负，则首先显示‘-’字符，再显示最后的结果的补码；

3.）完成实验报告

实验四　分支程序设计

1.实验目的

（1）掌握汇编语言的分支程序设计的原理与方法；

（2）掌握并能熟练应用汇编程序的输出显示设计方法；

（3）掌握汇编程序的数据输入设计方法；

（4）熟练应用 DEBUG 调试汇编程序；

（5）掌握算法的设计及表达方式。

2.实验内容

题目：分类统计字符个数

内容：程序接受用户从键盘输入的一行字符（字符个数不超过 80 个字符，该字符串以回车符结束），并按字母、数字及其他字符分类统计个数，然后将相应的结果存放于 letter、digit 和 other 中，并在显示器上显示如下信息：

The counted result of the program:

letter:××

digit:××

other:××

××表示一个十进制数

3.实验要求

（1）上机实验前，仔细复习课本有关知识；

（2）独立完成实验，画出流程图并上交实验报告；

（3）尝试将结果输出，如不输出，可通过单步执行，将内存单元 letter、digit 和 other 中的内容显示并截图。

4.实验步骤

1.）算法分析

从实验的内容分析可知，要完成如下实验，可分为以下步骤：

（1）接受键盘的输入，以回车键结束输入；

（2）对输入的字符分类统计；

（3）输出统计结果。

2.）算法设计

（1）输入与统计

已知：ASCII 码的 0DH 为回车符，DOS 的 1 号功能可接受键盘的输入，存放在（AL）中，可以选用如下算法（伪代码）：

```
        01h->AH;
NEXTCHAR:
        INT 21H;
```

```
if （AL）!= ODH
    if （AL）== letter
        letter++;
    else if（AL）== digit
        digit ++;
    else other++;
else JMP NEXTOPERATION
JMP NEXTCHAR;
```

（2）输出显示

请思考如何将内存中存放的十六进制数转换成 ASCII 码表示的十进制数并输出。（提示：参考将十进制数转换成二进制数的方法，必要时可借助堆栈）

3.）完成实验报告

实验五　循环程序设计

1.实验目的

（1）掌握汇编语言的循环程序设计的原理与方法；

（2）掌握并能熟练应用汇编程序的输出显示设计方法；

（3）掌握汇编程序的循环算法编写；

（4）熟练应用 DEBUG 调试汇编程序；

（5）掌握算法的设计及表达方式。

2.实验内容

题目：对随机输入的 11 名学生的成绩进行排序与分数段人数统计，输出最高分，最低分，中间值以及排序后的成绩。

内容：定义存储空间存放成绩，输出不及格人数，60~69 分数段人数，……，90~100 分数段人数，对成绩进行排序，输出排序后的结果，在显示器上显示如下信息：

The score between 90 and 100 ： ××

……

The score between 0 and 59 ： ××

Min is ××

Max is ××

The middle score is ××

Rank ordering ： ×× ×× ××……

××表示一个十进制数

3.实验要求

（1）上机实验前，仔细复习课本有关知识；

（2）独立完成实验，画出流程图并上交实验报告；

4.实验步骤

1.）算法分析

从实验的内容分析可知，要完成如下实验，可分为以下步骤：

（1）输入分数并转换；

（2）排序，得到中间值，最高分，最低分；

（3）统计各个分数段的人数；

（4）输出结果。

2.）算法设计

（1）输入与转换

定义一个缓冲区，存放操作者以数字和空格键入的字符串：

```
BUF db 80,?,80 dup(0)
```

提示：输入分数以空格（20H）隔开，通过判断是否为空格划分分数。对于不是空格的字符需要看它的后一个字符是不是空格，如果是则它是个位数，如果不是则该数字将是一个两位数，然后进行相应的操作；

在数据段定义 11 个存储空间以存放进行操作后的 11 个键入的分数。

（2）排序

可运用冒泡排序，选择排序等方法对 11 个分数进行排序。

（3）统计人数与输出显示

可参考实验四。

3.）完成实验报告

实验六 子程序设计

1.实验目的

（1）综合掌握比较复杂的汇编程序设计方法；

（2）熟练应用 DEBUG 调试汇编程序；

（3）掌握算法的设计及表达方式。

2.实验原理

利用本学期所学知识：汇编语言数据的存储方式，循环程序、分支程序以及子程序设计的方法，DOS 中断调用等，设计一个比较复杂的综合应用程序。

3.实验内容

1.）建立一个可存放50项的电话号码表，每项包括人名(20个字符)及电话号码(8个字符)两部分；

2.）程序可接收输入人名及相应的电话号码，并把它们加入电话号码表中；

3.）凡有新的输入后，程序应按人名对电话号码表重新排序；

4.）程序可接收需要查找电话号码的人名，并从电话号码表中查出电话号码，再在屏幕上显示出来。

name	tel..
×××	×××

4.实验要求

1.）上机实验前，仔细复习课本有关知识；

2.）独立完成实验，画出流程图并上交实验报告；

3.）主程序的主要部分如下：

(1) 显示提示符‘Input name:’；

(2) 调用子程序input_name接收人名；

(3) 调用子程序stor_name把人名存入电话号码表tel_tab中；

(4) 显示提示符‘Input a telephone number:’；

(5) 调用子程序inphone 接收电话号码，并把它存入电话号码表按人名排序；

(6) 显示提示符‘Do you want a telephone number? (Y/N)’；

(7) 回答N则退出程序；

(8) 回答Y则再显示提示符‘name?’；

(9) 调用子程序input_name 接收人名；

（10） 调用子程序 name_search 在电话号码表中查找所要的电话号码；

（11） 调用子程序 printline 按要求格式显示人名及电话号码；

（12） 重复查号提示符直至用户不再要求查号为止。

4.）完成实验报告。

附加程序：

```
    DATA SEGMENT
    INPUTNAME DB 'Please input the name: $'
    INPUTNUM DB 'Please input phone number: $'
    QUERY DB 'Please input the name you want to search: $'
    ANSWER DB ' phone number is :$'
    ERRORS DB 'ERROR$'
    LINE DB 0DH,0AH,'$'
    NUM DB 00H
    OVER DB '$'
    SEARCH DB 11 DUP(0)
    USER DB 110 DUP(0)
    PHONE DB 110 DUP(0)
    AI DB 'If you want to add,press 1',0DH,0AH,'If you want to search,press
2',0DH,0AH,'$'
DATA ENDS
STACK SEGMENT

STACK ENDS
CODE SEGMENT
    ASSUME CS:CODE,DS:DATA,SS:STACK
START:
    MOV AX,DATA
    MOV DS,AX

    CALL INPUT

    CALL PAIXU

    CALL PRINT

    AGAIN6:
    CALL SELECT
    JMP AGAIN6
```

```
MOV AH,4CH
INT 21H

SELECT:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV DX,OFFSET LINE
MOV AH,09H
INT 21H

MOV DX,OFFSET AI
MOV AH,09H
INT 21H

MOV AH,01H
INT 21H

CMP AL,31H
JZ SHURU
CMP AL,32H
JZ CHAXUN

JMP OUV

SHURU:
CALL PUT123

CALL PAIXU

CALL PRINT
JMP OUV
CHAXUN:
CALL INPUTQUERY
JMP OUV
OUV:
MOV DX,OFFSET LINE
MOV AH,09H
INT 21H
POP DX
POP CX
POP BX
```

```
POP AX
RET

PUT123:
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH SI

MOV BX,OFFSET NUM
MOV AL,BYTE PTR[BX]
MOV AH,00H
MOV CL,0BH
MUL CL
MOV SI,AX
MOV DX,OFFSET INPUTNAME
MOV AH,09H
INT 21H
PUSH SI

MOV BX,OFFSET USER

PUTNAME:

MOV AH,01H
INT 21H
MOV BYTE PTR[BX][SI],AL
INC SI

CMP AL,0DH
JNZ PUTNAME
DEC SI
MOV BYTE PTR[BX][SI],'$'

POP SI

MOV DX,OFFSET INPUTNUM
MOV AH,09H
INT 21H

MOV BX,OFFSET PHONE

PUTPHONE:
```

```
MOV AH,01H
INT 21H
MOV BYTE PTR[BX][SI],AL
INC SI

CMP AL,0DH
JNZ PUTPHONE

DEC SI
MOV BYTE PTR[BX][SI],'$'

MOV BX,OFFSET NUM
MOV AL,BYTE PTR[BX]
INC AL
MOV BYTE PTR[BX],AL

POP SI
POP DX
POP CX
POP BX
POP AX

RET

PAIXU:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV BX,OFFSET NUM
MOV CL,BYTE PTR[BX]
MOV CH,00H
MOV BX,OFFSET USER
AGAIN7:
    PUSH CX
    DEC CX
    CMP CL,00H
    JZ NOAGAIN
    MOV SI,0000H
    AGAIN8:
        MOV AH,BYTE PTR [BX][SI]
        MOV AL,BYTE PTR 0BH[BX][SI]
```

```
        CMP AH,AL
        JNC EXCHANGE
        NOEXCHANGE:
        ADD SI,0BH
    LOOP AGAIN8
    NOAGAIN:
    POP CX
LOOP AGAIN7

JMP ENDDD

EXCHANGE:
    PUSH BX
    PUSH CX
    PUSH SI
    PUSH AX
    PUSH DI
    ;;;;;;;交换SI 和 SI+0BH
    MOV CX,0000H
    MOV CL,0BH
    MOV DI,SI
    ADD DI,000BH
    AGAIN9:
        MOV AH,BYTE PTR[BX][SI]
        MOV AL,BYTE PTR[BX][DI]
        MOV BYTE PTR[BX][SI],AL
        MOV BYTE PTR[BX][DI],AH

        PUSH BX
        MOV BX,OFFSET PHONE
        MOV AH,BYTE PTR[BX][SI]
        MOV AL,BYTE PTR[BX][DI]
        MOV BYTE PTR[BX][SI],AL
        MOV BYTE PTR[BX][DI],AH
        POP BX

        INC SI
        INC DI
    LOOP AGAIN9
    POP DI
    POP AX
    POP SI
    POP CX
    POP BX
```

```
JMP NOEXCHANGE
ENDDD:
POP DX
POP CX
POP BX
POP AX
RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;
PRINT:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV BX,OFFSET NUM
MOV CL,BYTE PTR[BX]
MOV CH,00H
MOV SI,0000H

MOV DX,OFFSET LINE
MOV AH,09H
INT 21H

AGAIN3:
    MOV BX,OFFSET USER
    ADD BX,SI
    MOV DX,BX
    MOV AH,09H
    INT 21H
    MOV DL,' '
    MOV AH,02H
    INT 21H
    INT 21H
    INT 21H
    MOV BX,OFFSET PHONE
    ADD BX,SI
    MOV DX,BX
    MOV AH,09H
    INT 21H
    MOV DX,OFFSET LINE
    MOV AH,09H
```

```
    INT 21H
    ADD SI,000BH
LOOP AGAIN3

POP DX
POP CX
POP BX
POP AX

RET

INPUTQUERY:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV DX,OFFSET QUERY
MOV AH,09H
INT 21H
MOV BX,OFFSET SEARCH
MOV SI,0000H
PUT:
    MOV AH,01H
    INT 21H
    CMP AL,0DH
    JZ COM
    MOV BYTE PTR[BX][SI],AL
    INC SI
JMP PUT
COM:
MOV BYTE PTR[BX][SI],'$'
MOV DX,OFFSET LINE
MOV AH,09H
INT 21H;;查询的输入程序

;;目前为止 SI 当中保存的输入的字符的个数+1

SUB SI,01H
MOV CX,SI
MOV SI,0000H
MOV BX,OFFSET NUM
MOV DL,BYTE PTR[BX];电话本中的人的个数
MOV DH,00H
```

```
MOV CH,00H
MOV CL,00H
MOV DI,0000H
AGAIN4:
    MOV BX,OFFSET USER
    MOV AH,BYTE PTR [BX][SI];AH 当中保存的是用户
    MOV BX,OFFSET SEARCH
    MOV AL,BYTE PTR DS:[BX][DI];AL 当中保存的是所查询的人的姓名
    CMP AH,AL
    JZ EQUAL
    JMP UNEQUAL
EQUAL:
    INC SI
    INC DI
    CMP AH,'$'
    JNZ AGAIN4
    JMP PRINTPHONE
UNEQUAL:
MOV DI,0000H
DEC DL
INC CL
PUSH CX
MOV SI,0000H
AGAIN5:
    ADD SI,0BH
LOOP AGAIN5
POP CX

CMP DL,00H
JNZ AGAIN4

JMP FAIL

PRINTPHONE:
PUSH DX
MOV DX,OFFSET SEARCH
MOV AH,09H
INT 21H
MOV DX,OFFSET ANSWER
MOV AH,09H
INT 21H
POP DX
MOV BX,OFFSET NUM
MOV DH,BYTE PTR[BX]
```

```
SUB DH,DL
;;DH 是要输出的
MOV BX,OFFSET PHONE
MOV AX,0000H
MOV AL,DH
MOV CL,0BH
MUL CL
ADD BX,AX
MOV DX,BX
MOV AH,09H
INT 21H
JMP ENDD
FAIL:

MOV DX,OFFSET ERRORS
MOV AH,09H
INT 21H

ENDD:

MOV DX,OFFSET LINE
MOV AH,09H
INT 21H

POP DX
POP CX
POP BX
POP AX
RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;

INPUT:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV SI,0000H
MOV BX,OFFSET USER
MOV CX,0000H;CH 用来存人的个数 用来存目前字符的个数
AGAIN:
```

```
MOV DX,OFFSET INPUTNAME
MOV AH,09H
INT 21H
AGAIN1:
    MOV AH,01H
    INT 21H
    CMP AL,0DH
    JZ COMPLETE
    PUSH SI
    PUSH CX
    MOV CH,00H
    ADD SI,CX
    POP CX
    MOV BYTE PTR [BX][SI],AL
    POP SI
    INC CL
JMP AGAIN1
COMPLETE:
CMP CL,00H
JZ ENDD1

;输入电话号码:
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV DX,OFFSET LINE
MOV AH,09H
INT 21H
MOV DX,OFFSET INPUTNUM
INT 21H
MOV CL,00H
MOV BX,OFFSET PHONE
AGAIN2:
    MOV AH,01H
    INT 21H
    CMP AL,0DH
    JZ PHONEOVER
    PUSH SI
    PUSH CX
    MOV CH,00H
    ADD SI,CX
    POP CX
```

```
        MOV BYTE PTR [BX][SI],AL
        POP SI
        INC CL
    JMP AGAIN2

    PHONEOVER:

    PUSH SI
    PUSH CX
    MOV CH,00H
    ADD SI,CX
    MOV BYTE PTR [BX][SI],'$'
    POP CX
    POP SI

    POP DX
    POP CX
    POP BX
    POP AX

    INC CH
    PUSH SI
    PUSH CX
    MOV CH,00H
    ADD SI,CX
    MOV BYTE PTR [BX][SI],'$'
    POP CX
    POP SI
    ADD SI,0BH;最多存放的字符
    MOV DX,OFFSET LINE
    MOV AH,09H
    INT 21H
    MOV CL,00H
    JMP AGAIN

ENDD1:
MOV BX,OFFSET NUM
MOV BYTE PTR[BX],CH


POP DX
POP CX
POP BX
POP AX
```

```
    RET

CODE ENDS
END START
```