

---

# 《汇编语言与接口技术》

## 实验指导

河南大学计算机与信息工程学院

2021.9

# 汇编语言实验环境和要求

《汇编语言》是计算机及计算机类专业学生的专业基础课，是培养学生直接使用计算机硬件资源能力的一门课程。它不仅能帮助学生进一步理解计算机组成原理课程中的各种概念，而且还为其他课程：操作系统、接口与通信技术和计算机控制技术等课程提供必要的预备知识。该课程在计算机学科设置中起着承上启下的作用。

## 一、实验环境

汇编实验是在一般 PC 机上完成的，采用 MASM6.11 宏汇编进行实验。MASM6.11 版本对硬件环境无特殊要求，对软件的要求如下：

- (1) 操作系统：Windows2000 的命令提示符状态。

在 Windows 下进入 DOS 命令行的方法：“开始”→“运行” 键入 “cmd” 即可进入 DOS 命令行；

Windows7 的操作系统下进入 Dosbox，在模拟的 16 位环境下键入命令。

- (2) 编辑环境：EDIT.COM、记事本等文本编辑器（随操作系统提供），建议使用 notepad++.exe 文本编辑。

- (3) 汇编和连接程序文件：

汇编程序 MASM.EXE    连接程序 LINK.EXE

汇编连接程序 ML.EXE    汇编链接错误提示 ML.ERR

- (4) 调试程序：DEBUG.EXE（随操作系统提供）。

## 二、PC 机 DEBUG 调试工具的使用

DEBUG.EXE 程序是专门为分析和开发汇编语言程序而设计的一种调试工具，具有跟踪程序执行、观察中间运行结果、显示和修改寄存器或存储单元内容等多种功能。是学习汇编语言必须掌握的调试工具。

### 1、DEBUG 程序使用

在命令提示符下键入命令：

DEBUG [盘符:][路径][文件名.EXE][参数 1][参数 2]

这时屏幕上出现 **DEBUG 的提示符“-”**，表示系统在 DEBUG 状态下，此时可以用 DEBUG 命令进行程序调试。**在 DEBUG 环境下，默认采用十六进制数制**，所有数值不需要带数制后缀。

若进入 DEBUG 的命令中将所有的参数都省略，则仅进入 DEBUG 环境，内存中不包

含特定的程序和数据。此时，可写入指令和数据进行验证调试，也可以使用 N 或者 L 命令从指定盘上装入要调试的程序。如果进入 DEBUG 的命令中包含文件名，则进入 DEBUG 环境的同时，将指定程序调入内存，当前程序的代码段作为默认的 CS 段，从 0 单元保存。

## 2、DEBUG 的常用命令

DEBUG 命令都是单字母命令，按照不同命令的格式其后可加上一个或多个参数，若包含多个命令参数，其间用空格作为分隔符。DEBUG 命令不区分大小写。

### (1) 汇编命令 A

**格式：**A[起始地址]

**功能：**以汇编指令的形式输入代码，系统自动将汇编指令翻译成机器指令代码，并从默认或指定地址单元开始存放。

若缺省起始地址，则从当前 CS: 100 地址开始存放。A 命令按行汇编，主要是用于小段程序的汇编或对目标程序的修改。

**举例：**

- 命令：A                    含义：从默认地址输入汇编指令；
- 命令：A 1000:20        含义：从地址为 1000H:20H 的单元输入汇编指令；
- 命令：A CS:1000        含义：从 CS 段的 1000H 单元输入汇编指令；

### (2) 反汇编命令 U

**格式 1：**U[起始地址]

**格式 2：**U[起始地址][结束地址|字节数]

**功能：**格式 1 从指定起始地址处开始将 32 个字节的**目标代码**(非 32 条指令)转换成汇编指令形式，如果省略起始地址，则从当前 CS:IP 指向地址开始反汇编、或接着上次 U 命令继续反汇编。

格式 2 将指定范围的内存单元中的目标代码转换成汇编指令。注意，这里的反汇编是以指令为单位进行显示。

**界面说明：**

-u0			
13C8:0000	CD20	INT	20
13C8:0002	FF9F009A	CALL	FAR [BX+9A00]
13C8:0006	EE	OUT	DX,AL
13C8:0007	FE1D	CALL	FAR [DI]
13C8:0009	F0	LOCK	
13C8:000A	4F	DEC	DI
13C8:000B	032C	ADD	BP,[SI]
13C8:000D	0E	PUSH	CS
13C8:000E	8A03	MOV	AL,[BP+DI]
13C8:0010	2C0E	SUB	AL,0E
13C8:0012	17	POP	SS
13C8:0013	032C	ADD	BP,[SI]
13C8:0015	0E	PUSH	CS
13C8:0016	250401	AND	AX,0104
13C8:0019	0101	ADD	[BX+DI],AX
13C8:001B	0002	ADD	[BP+SI],AL
13C8:001D	FFFF	???	DI
13C8:001F	FFFF	???	DI

- 界面左边：以逻辑地址形式显示每条汇编指令所在存储单元的首单元地址；
- 界面中间：每条汇编指令对应的机器指令代码；
- 界面右边：处理了符号之后的汇编指令；

举例：

- 命令：U                    含义：查看默认地址的汇编指令，默认条数；
- 命令：U 0                含义：从 0 地址查看汇编指令，默认 CS 段；
- 命令：U 0 50            含义：查看地址 0 到 50H 的汇编指令，默认 CS 段；  
地址范围要确认该范围中所有指令都是完整的。
- 命令：U DS:0            含义：将 DS:0 地址中的内容当作指令，查看默认条数；

### (3) 显示、修改寄存器命令 R

格式：R[寄存器名]

功能：如果给出寄存器名，则显示该寄存器的内容并可进行修改。如果不指定寄存器名，则显示所有寄存器的内容及当前值（不能修改）。

R 命令只能显示、修改 16 位寄存器，对于标志寄存器只能通过执行指令的方式修改。

界面说明：

```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=13C8 ES=13C8 SS=13C8 CS=13C8 IP=0100 NV UP EI PL NZ NA PO NC
13C8:0100 89D8          MOV     AX,BX
```

- 界面中间两行：显示各寄存器内容；先显示 13 个寄存器的值；后面的 8 个符号分别表示 OF、DF、IF、SF、ZF、AF、PF、CF 标志位的状态，符号含义如下表：

标志位	OF	DF	IF	SF	ZF	AF	PF	CF
置位符号（1）	OV	DN	EI	NG	ZR	AC	PE	CY
复位符号（0）	NV	UP	DI	PL	NZ	NA	PO	NC

- 界面最后一行：显示下一条将要执行的指令；若指令中有存储单元寻址方式

的操作数，则指令后显示该操作数的地址和**数值**；

**举例：**

- 命令：R                    含义：查看所有寄存器内容；
- 命令：R ax                含义：查看寄存器 AX 中的内容，并可修改；

#### (4) 显示存储单元命令 D

**格式 1：**D[起始地址]

**格式 2：**D[起始地址][结束地址][L 字节数]

**功能：**格式 1 从起始地址开始按十六进制显示 128 个单元的内容，每行 16 个单元，共 8 行。格式 2 显示指定范围内存储单元的内容，其他显示方式与格式 1 一样。如果省略起始地址或地址范围，则从当前的地址开始按格式 1 显示。

**界面说明：**

```
-d100
13c8:0100  30 31 32 33 34 35 36 37-38 39 00 00 00 00 00 00  0123456789.....
13c8:0110  41 42 43 44 45 46 47 48-49 50 00 00 00 00 00 00  ABCDEFGHIP.....
13c8:0120  61 62 63 64 65 66 67 68-69 70 00 00 00 00 00 00  abcdefghip.....
13c8:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
13c8:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
13c8:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
13c8:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
13c8:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
```

- 界面左边：以逻辑地址形式显示指定的或默认的存储单元地址；
- 界面中间：存储单元中的实际存放内容，每一行显示 16 个单元，每个单元存放一个字节数据；
- 界面右边：存储单元中值的对应的 ASCII 码字符；若存储单元中存放的是不可显示的 ASCII 码值，则显示“.”；

例如：-D 200                    ;表示从 DS:0200H 开始显示 128 个单元内容

-D 100 120                    ;表示显示 DS:0100-DS:0120 单元的内容

说明：在 DEBUG 中，地址表示方式有如下形式：

段寄存器名：相对地址，如：DS:100

段基值：偏移地址（相对地址），如：23A0:1500

**举例：**

- 命令：D                    含义：默认查看；
- 命令：D 1000:0            含义：从指定单元查看数据，默认长度；
- 命令：D DS:0              含义：从指定单元查看 DS 段中的数据，默认长度；
- 命令：D 0 5                含义：查看 DS 段中 0~5 单元中的 6 个数据；
- 命令：D 10 L 5            含义：查看 DS 段中从 10H 开始的连续 5 个单元内容；

#### (5) 修改存储单元命令 E

**格式 1：**E[起始地址] [内容表]

**格式 2：**E[地址]

**功能：** 格式 1 按内容表的内容修改从起始地址开始的多个存储单元内容，即用内容表指定的内容来代替存储单元当前内容。

例如：

—E DS: 0100 'VAR' 12 34

表示从 DS:0100 为起始单元的连续五个字节单元内容依次被修改为 'V'、'A'、'R'、12H、34H。

格式 2 是逐个修改指定地址单元的当前内容。

如：—E DS: 0010

156F: 0010 41. 5F

其中 156F:0010 单元原来的值是 41H，5FH 为输入的修改值。若只修改一个单元的内容，这时按回车键即可；若还想继续修改下一个单元内容，此时应按空格键，就显示下一个单元的内容，需修改就键入新的内容，不修改再按空格跳过，如此重复直到修改完毕，按回车键返回 DEBUG “-” 提示符。如果在修改过程中，将空格键换成按 “-” 键，则表示可以修改前一个单元的内容。

**举例：**

➤ 命令：E 100

含义：查看 DS 段中 100H 单元起始的存储单元内容，并可随时修改；

➤ 命令：E ES:10 1 31 20

含义：将 ES 段中地址为 10H~12H 的 3 个单元中的值修改为 1、31H、20H；

➤ 命令：E 10 'ABCD'

含义：将 DS 段中地址 10H~13H 的连续 4 个单元内容修改为 41H~44H；

## (6) 运行命令 G

**格式：** G[=起始地址][断点地址]

**功能：** CPU 从指定起始地址开始执行，在断点地址处停止，即断点地址中的指令不执行。若省略起始地址，则从当前 CS:IP 指示地址开始执行一条指令。

**注意，断点地址必须是某一条指令的起始地址，否则查看到的程序有可能会出错。**

**举例：**

➤ 命令：G 含义：从当前位置连续执行程序，到结束；

➤ 命令：G=0 含义：从 0 地址连续执行程序，到结束；

➤ 命令：G 14 含义：从当前位置执行程序到 14H，中断；

➤ 命令：G=3 20 含义：从地址 3 执行程序到地址 20H，中断；

## (7) 跟踪命令 T

**格式：** T[=起始地址][指令条数]

**功能：** 从指定地址开始执行指定条数的指令，若省略指令条数，则默认执行一条指令，若省略起始地址，则从当前 CS:IP 指示地址开始执行。

**举例：**

- 命令：T                   含义：从当前位置单步执行 1 条指令；
- 命令：T 3                含义：从当前位置单步执行 3 条指令；
- 命令：T=0               含义：从 0 地址单步执行 1 条指令；
- 命令：T=4 2            含义：从 4 地址单步执行 2 条指令；

## (8) 退出命令 Q

**格式：**Q

**功能：**退出 DEBUG，返回到操作系统。

# 三、PC 机汇编语言程序设计实验步骤

## (1) 编辑源程序文件

使用 EDIT 文本编辑器编辑源文件，键入

**EDIT <源文件名.ASM>**

或 **EDIT**

但使用后者时应注意将文件保存为.ASM 文件。

## (2) 汇编连接源程序文件

**ML <源文件名.ASM>**

如果源程序没有错误，则自动生成.OBJ 文件和.EXE 可执行文件。

*注意：若源程序有语法错误时，会出现错误信息提示，需回到编辑状态下修改源程序后重新汇编。*

## (3) 运行程序

经过汇编、连接后生成的.EXE 文件，可直接运行，只要键入相应的文件名即可。

## (4) 调试程序：

使用 DEBUG 调试前面生成的可执行的.EXE 文件。各种 DEBUG 命令如前所述。

# 四、实验报告书写要求

(1) 使用专用的实验报告纸，每个实验一份，字迹工整，内容清晰，注意填写必要的信息（如：姓名、学号、班级、辅导教师、同实验者）；

(2) 填写实验题目、实验目的等；实验步骤中要求列出当次实验的过程及各种数据输入输出的情况；汇编语言程序设计部分还要写出完整的源程序，以及上机调试过程中遇到的问题和解决方法。

# 预备实验一 DEBUG 命令练习

## 实验目的

- 1、熟悉汇编语言程序的编写、汇编、运行的一般过程；
- 2、掌握常用的 DEBUG 命令，并可应用于汇编语言程序调试过程中。

## 实验内容

- 1、使用给定程序练习汇编语言程序的编辑、汇编、执行、调试的过程；
- 2、在 DEBUG 状态下，练习常用的 DEBUG 命令调试给定程序。

## 实验学时

本实验内容共用 4 学时，分两次实验完成。

## 实验步骤

### 1、汇编语言程序的编辑、汇编、执行练习

#### (1) 建立汇编语言程序源文件

使用 DOS 或 Windows 下的任何文本编辑器来建立汇编语言程序源文件，建议使用记事本，文件后缀名应保存为 “.asm”（请注意不要隐藏已知后缀名）。

将如下程序代码保存为 “test.asm” 的源文件。

#### 程序功能：

在屏幕上显示 CHAR 变量所定义的字符。

#### 程序代码：

```
DATA SEGMENT
    CHAR DB 'F'
;请注意源代码中的标点符号均为英文状态
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE , DS:DATA
START: MOV AX , DATA
        MOV DS , AX
```



```

MOV DL , CHAR
MOV AH , 2
INT 21H
MOV AX , 4C00H
;请注意以上语句中, 4C00H 是十六进制数据, 而非字母
INT 21H

CODE ENDS
END START

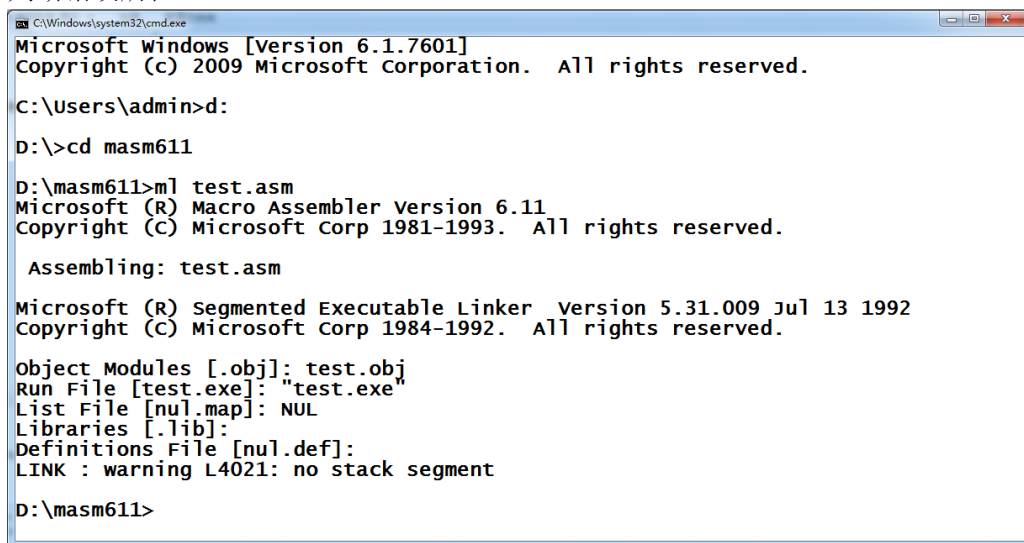
```

## (2) 汇编源程序, 生成可执行文件

在保证工作目录下存在 ml.exe 文件和 link.exe 文件时, 可直接对汇编语言源程序进行汇编连接, 使用的命令如下 (以 test.asm 文件名为例):

```
ml test.asm
```

若源程序无语法和逻辑错误, 即可汇编连接成功, 同时生成目标文件 test.obj 和可执行文件 test.exe。当源程序中存在语法错误时, 汇编连接不会进行。若工作目录下存在 ml.err 文件, 则会显示包含错误存在行的提示信息, 程序员可以依次来判断错误所在。



```

C:\Windows\system32\cmd.exe
Microsoft windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\admin>d:
D:\>>cd masm611
D:\masm611>ml test.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

Assembling: test.asm

Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.

Object Modules [.obj]: test.obj
Run File [test.exe]: "test.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:
LINK : warning L4021: no stack segment

D:\masm611>

```

图1 test.asm文件的汇编过程

**注意:** 该命令中源文件的后缀名一定不可省去, 否则会提示无法找到源文件。

### 错误提示

若源程序中存在语法错误, 则汇编程序会停止对源程序的汇编过程, 并显示错误原因, 错误提示的格式为如下, 图 2 为显示示例。

**汇编语言程序文件名 (错误行号): 错误编号: 错误原因**

```
D:\masm611>ml test.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

Assembling: test.asm
test.asm(2): error A2044: invalid character in file
test.asm(14): error A2048: nondigit in number
test.asm(10): error A2006: undefined symbol : CHAR
```

图2 错误信息

### 本程序中常出现的错误:

- ① 第 2 行出错: 源程序第 2 行中的“CHAR DB ‘F’”单引号改为英文状态, 如图 2 中的第一行错误“invalid character in file”; 该错误会导致第三行的错误“undefined symbol”, 即 CHAR 定义语句写错, 导致程序中使用 CHAR 变量时无法找到。
- ② 第 11 行出错: 源程序第 11 行中的“MOV AX, 4C00H”, 将数字“00”写成了字母“oo”; 该错误会导致如图 2 中的第二行错误“nondigit in number”, 即该使用数值中没有找到数字。
- ③ 找不到源文件: 源文件不在当前操作的目录下; 或者文件名用错了。

### (3) 执行程序, 查看结果

在 DOS 命令行下, 运行生成的可执行文件, 命令为:

**test.exe**

该命令中, 文件的后缀名可省去。

若程序有输入输出内容, 则会在 DOS 命令行下看到结果。若程序没有输入输出内容, 则只能进入 DEBUG 状态下查看程序的运行结果。

## 2、DEBUG命令练习

将上面生成的可执行文件 test.exe 文件调入 DEBUG 进行调试, 使用的命令是:

**debug test.exe**

注意, 这里调入的是可执行文件, 后缀名必须加。

进入 DEBUG 下, 命令提示符是“-”, 可以使用各种 DEBUG 命令调试。这里建议使用以下命令查看, **注意以下命令是进入 DEBUG 后连续执行的结果**, 单独使用可能无法得到预期效果。

#### (1) 反汇编命令 U

当调入程序文件后, 直接使用无参数的 U 命令可从第一条指令开始查看当前的程序代码。屏幕显示的内容是: 默认从第一行代码开始显示, 共反汇编 32 个单元的指令 (指令条数依据指令长度而不同, 以整条指令为单位)。

请注意查看 DEBUG 状态下的指令与源文件中的指令有何不同。

#### (2) 寄存器查看命令 R

直接使用无参的 R 命令可查看当前各寄存器状态, 由于程序还未执行, 可以看到

寄存器的状态为初始的随机值。

请注意当前 CS:IP 寄存器的内容与上一步 U 命令看到的指令地址是一致的，但 DS、ES 等段寄存器则为初始的随机值。

### (3) 查看存储单元命令 D

直接使用无参数的 D 命令，显示存储单元内容。

应注意区分的是：当前查看到的是从 DS:0 地址开始的连续 128 个单位的数据，并不是当前程序的数据段数据。

### (4) 单步执行命令 T

从程序开始单步执行程序的前两条指令，使用的命令格式为：

**T 2**

执行完前两条指令即返回，显示当前的寄存器状态，请注意查看当前的 CS、IP 寄存器状态与执行指令前的差别，请试着解释这两个寄存器的含义。

**注意：INT 指令不可单步执行，需使用 G 命令连续执行。**

### (5) 查看存储单元命令 D

使用带参数的 D 命令，显示从有效地址为 0 的存储单元开始的连续 128 个单元中的数据。在此查看存储单元地址、单元内容、以及对应的字符显示。命令为：

**D 0**

请注意该 D 命令显示的内容与上一个 D 命令显示的单元地址是否相同，为什么？

### (6) 反汇编命令 U

直接使用反汇编命令 U，会接着上次 T 命令执行的位置 05 处开始反汇编显示指令。

### (7) 断点/连续执行命令 G

使用断点执行的功能，从当前位置执行到显示指令结束，命令为：

**G 0D**

程序执行，可在屏幕上看到“F”字符的显示。

注意，该命令中的“0D”是断点地址，即指令“MOV AX, 4C00H”的保存位置，该命令会将该位置之前的所有指令执行完毕。断点地址必须是 U 命令可见的地址。

### (8) 修改存储单元内容命令 E

使用 E 命令将 CHAR 变量中的字符修改为“M”，命令为：

**E 0 'M'**

或者使用命令

**E 0**

在提示下输入 M 的 ASCII 码 4DH。

### (9) 查看存储单元内容命令 D

使用 D 命令查看所修改的内容，命令为：

**D 0 L1**

### (10) 连续执行命令 G

1 1

从第一条指令开始重新执行程序，使用 G 命令：

**G=0 10**

指定程序执行的区间，在屏幕上可以看到“M”字符的显示。

(11) 汇编命令 A

在 DEBUG 下修改汇编语言程序，使用 A 命令：

**A 5**

在给定的单元中输入指令“MOV DL, 39”，两次回车，返回到 DEBUG 提示符下。

(12) 连续执行命令 G

重新执行程序，使用 G 命令：

**G=0**

指定程序执行的区间，在屏幕上可以看到“9”字符的显示，同时显示“Program terminated normally”的提示信息，表示程序执行结束。

(13) 退出命令 Q

执行 Q 命令，即可从 DEBUG 状态下退出，返回 DOS 系统。

## 思考问题

1. DEBUG 状态下如何观察源程序？和文本编辑方式下的源程序有哪些区别？
2. 试解释以上第 2 步中各 DEBUG 命令的含义。
3. 若要将程序中的第三条汇编指令换成“MOV DL, 61H”，则更换该指令的 DEBUG 命令应为什么？更换后程序的执行结果是什么？

## 实验报告要求

本次实验不写实验报告。

## 练习

尝试按上面的方法调试该程序。

### 程序功能：

从键盘上输入一个字符串，保存于数据段中，分别显示该字符串的长度和字符串中第二个字符。

### 程序代码：

```
DATA SEGMENT
    INMESS DB 'PLEASE INPUT A STRING(LENGTH<9):$'
    LENMESS DB 10,13,'THE LENGTH OF THE STRING IS:$'
    CHARMESS DB 10,13,'THE SECOND CHAR OF THE STRING IS:$'
    INSTRING DB 10,?,10 DUP(?)
```

1 2

```

DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE, DS:DATA
START:  MOV  AX , DATA
        MOV  DS , AX
        LEA  DX , INMESS    ;① 显示提示信息
        MOV  AH , 09H
        INT  21H
        LEA  DX , INSTRING  ;② 输入字符串, 保存于 INSTRING
        MOV  AH , 0AH
        INT  21H
        LEA  DX , LENMESS   ;③ 显示提示信息
        MOV  AH , 09H
        INT  21H
        LEA  BX , INSTRING  ;④ 获取输入字符串的长度值, 并显示
        INC  BX
        MOV  DL , [BX]
        ADD  DL , 30H
        MOV  AH , 02H
        INT  21H
        LEA  DX , CHARMESS  ;⑤ 显示提示信息
        MOV  AH , 09H
        INT  21H
        MOV  DL , [BX+2]    ;⑥ 显示字符串中的第二个字符
        MOV  AH , 02H
        INT  21H
        MOV  AX , 4C00H     ;⑦ 程序返回
        INT  21H
CODE    ENDS
        END    START

```

### 调试内容:

- (1) 将生成的可执行文件调入 DEBUG, 观察各寄存器的内容;
- (2) 断点执行到指令①的位置, 观察各寄存器的变化情况、数据段中定义的各字符串 INMESS、LENMESS、CHARMESS、INSTRING 的位置;
- (3) 断点执行到指令②的位置, 观察程序执行的结果;
- (4) 断点执行到指令③的位置, 从键盘输入字符串, 返回后观察输入字符串的保存位置;

- (5) 断点执行到指令④的位置，观察程序执行的结果；
- (6) 分别单步执行指令④开始的 4 条指令，试观察每条指令执行的结果；
- (7) 断点执行到指令⑤的位置，观察程序执行的结果；
- (8) 断点执行到指令⑥的位置，观察程序执行的结果；
- (9) 断点执行到指令⑦的位置，观察程序执行的结果；

# 预备实验二 预备知识练习

## 实验目的

- 1、熟悉各种操作数的寻址方式，并能在 DEBUG 环境中查看各操作数；
- 2、熟悉 8086 指令系统中的各类指令的用法和功能，为以后的程序编写打下基础。

## 实验内容

- 1、使用给定程序练习汇编语言程序的编辑、汇编、执行、调试的过程；
- 2、在 DEBUG 状态下，练习常用的 DEBUG 命令调试给定程序。

## 实验学时

本实验内容共用 6 学时，分三次实验完成。

## 实验练习一：寻址方式练习

### 1、在DEBUG命令行下，按以下步骤完成操作。

- (1) 使用 e 命令修改存储地址以 150、200 为首的 6 个单元的内容：

```
-e150
****: 0150  **,00  **,01
-e200
****: 0200  **,10  **,20  **,30  **,40
```

注意：带下划线的部分是需要输入的内容。

- (2) 使用查看命令 d，查看上步所修改的存储单元的内容（注意对应的段地址）：

```
-d150  L2
-d200  L4
```

- (3) 使用汇编命令 a，输入以下指令序列：

```
-a100
****: 0100  mov ax , cs
****: 0102  mov ds , ax
****: 0104  mov bx , 100
****: 0107  mov ax , [150]
****: 010A  add bx , ax
```

```

****: 010C mov al , [bx]
****: 010E add al , [bx+1]
****: 0111 mov si , 2
****: 0114 add al , [bx+si]
****: 0116 mov ah , 0
****: 0118 mov ch , 0
****: 011A mov cl , [bx+si+1]
****: 011D add ax , cx

```

(4) 使用反汇编命令 u，查看上步输入的汇编语言程序（注意对应的段地址）：

-u100

(5) 使用单步执行命令 t 执行该指令序列，查看每一步执行的结果，并记录。

## 2、使用适当的DEBUG命令，按以下步骤完成操作。

(1) 修改以下寄存器的值：

(DS)=1000H , (ES)=2000H , (SI)=1FEH

(2) 将从存储单元 1000H:01FEH 开始的连续四个单元内容修改为：11H、22H、33H、44H；将从存储单元 2000H:01FEH 开始的连续四个单元内容修改为：12H、34H、56H、78H；

(3) 输入汇编指令：

```

MOV AX , DS
MOV BX , [SI+2]
MOV CX , [BX]
ES:
MOV DX , [SI]

```

(4) 指出以上各指令中源操作数的寻址方式，逻辑地址，并使用相关的 DEBUG 查看源操作数的值。

## 3、在Debug命令行下，完成以下操作。

(1) 将字数据 2345H 存放到地址为 1200H:100H 单元中；

(2) 使用适当的 DEBUG 命令，采用三种不同的逻辑地址形式查看这个字数据；

(3) 选择不同的寻址方式，使用 MOV 指令将这个字数据送入 AX 寄存器中；要求至少采用 3 种不同的寻址方式。



## 实验练习二：基本指令练习

### 1、数据传送指令的练习

在 DEBUG 命令行下，利用 A 命令输入以下指令序列，然后使用 T 命令单步跟踪执行，并使用相应的 DEBUG 命令查看每条指令的执行结果。

指令序列如下：

```
MOV  SP , B0
MOV  AX , 1234
MOV  BX , 5678
PUSH AX
PUSH BX
POP  CX
POP  DX
MOV  SI , 2000
LEA  BX , [SI]
MOV  WORD PTR [SI] , 5566
MOV  WORD PTR [BX+2] , 7788
LDS  AX , [SI]
MOV  DI , [SI]
```

### 2、算术运算指令的练习

在 DEBUG 命令行下，利用 A 命令输入以下指令序列，然后使用 T 命令单步跟踪执行，并使用相应的 DEBUG 命令查看每条指令的执行结果和标志位变化情况。

指令序列如下：

```
MOV  AX , 1234
MOV  BX , 5678
ADD  AX , BX
MOV  CX , 9ABC
SUB  CL , CH
MOV  BYTE PTR [2000] , 55
INC  BYTE PTR [2000]
MOV  AX , 20
MOV  BL , 8
DIV  BL
MOV  AL , 2
```

```

MOV CL , 10
MUL CL
MOV AL , 6
CBW
MOV AX , 1050
CWD

```

### 3、逻辑运算指令的练习

在 DEBUG 命令行下，利用 A 命令输入以下指令序列，然后使用 T 命令单步跟踪执行，并使用相应的 DEBUG 命令查看每条指令的执行结果和标志位变化情况。

指令序列如下：

```

MOV AX , FE
MOV BX , 5678
AND AX , BX
MOV CX , 9ABC
MOV DX , FF00
OR DX , CX
MOV AL , 55
XOR AL , 0F
MOV BL , 1
SHL BL , 1
SHL BL , 1
MOV CL , 80
SAR CL , 1
SHR CL , 1

```

### 实验练习三：操作符与表达式练习

1、已知数据段定义如下，请验证以下各指令的正误，并指出错误原因，或给出指令执行结果。

```

DATA SEGMENT
    WordVar dw 2 dup(?)
    ByteVar db ?
DATA ENDS

```

验证指令：

- ① MOV byte ptr ES:WordVar[BX] , 100
- ② MOV AX , offset WordVar[SI]
- ③ LEA AX , WordVar[SI]
- ④ CMP WordVar , ByteVar
- ⑤ MOV AL , ByteVar + WordVar
- ⑥ ADD WordVar , AL
- ⑦ MOV ByteVar , ByteVar - WordVar

2、已知数据段定义如下，请画出数据段中各数据的存放形式，并指出下列指令的执行结果。

```

DSEG    SEGMENT
        ORG    10H
        MyAddr    DW    $
        Bvar       DB    1, 2, 3
                   DB    '123'
        Buf        DB    5 DUP(?)
        EVEN
        Len1       =    $-Bvar
        Wvar       DW    1, 2
        ALIGN      4
        Dvar       DD    1, 2, 3
        Len2       EQU    $-Dvar
        Len3       EQU    Buf-Bvar
DSEG     ENDS

```

指令如下：

- ① MOV AX , offset Dvar
- ② MOV AX , Len1
- ③ MOV AX , Len2
- ④ MOV AX , Len3
- ⑤ MOV AX , MyAddr
- ⑥ MOV AX , word ptr BVar + 2
- ⑦ MOV AX , lengthof WVar + lengthof BVar
- ⑧ MOV AX , type DVar + sizeof BVar
- ⑨ MOV AX , word ptr DVar + 1

3、已知数据段定义如下，试完成以下要求的操作。

```
DSEG    SEGMENT
        ARRAY    DW    34H , 56H , 12H , 78H
        OTHER    DW    ?
        DA1       DB    20H DUP(?)
        DA2       DW    10H
DSEG     ENDS
```

- ① 试用 MOV 指令将数组 ARRAY 中的最后一个字数据传送到 BX 寄存器；
- ② 试用 MOV 指令将数组长度存入 CX 寄存器中；
- ③ 试使用一条指令将变量 DA1 中的数据个数送入 CX 寄存器中。

## 实验报告要求

本次实验不写实验报告。

# 实验一 基本编程方法练习

## 实验目的

- 1、通过编制简单的程序，练习汇编语言基本编程方法。
- 2、练习在 Debug 状态下调试程序的方法。

## 实验内容

根据以下题目要求，编写汇编语言源程序，并完成调试。

- 1、试编写程序，完成下面公式的计算。

$$A \leftarrow (X-Y+24) / Z \text{ 的商}, B \leftarrow (X-Y+24) / Z \text{ 的余数}$$

其中，变量 X 和 Y 是 32 位有符号数，变量 A、B、Z 是 16 位有符号数。

- 2、试将字节数据 B1 拆分成两个半字节数据，分别存放于其后两个单元 X 和 Y 中。

编程提示：

数据段定义应包含 B1、X、Y 三个变量，其中 B1 变量自定义具体的数值，X、Y 变量的数值由程序赋值，分别是 B1 的高低半个字节。该程序中需要使用逻辑操作完成半字节数据的拆分。

- 3、从键盘接收两个不大于 5 的十进制数字，并以十进制数据形式显示其和。

编程提示：

该程序中要求的均为一位数据的输入输出，暂时不考虑多位数据，请输入 0~5 之间的数据，显示的结果为 0~9。请不要输入两个 5，以免出现不能直接输出的情况。

- 4、从键盘接收一个字符串（假定输入字符串长度大于 3），试换行输出该字符串中第二个字符开始的连续 2 个字符。

编程提示：

该程序应先使用 DOS 功能调用接收一个字符串。

显示其中的子串，要求从第二个字符开始，连续显示 2 个字符，该功能可使用字符串显示的 DOS 功能调用，也可以使用单字符显示的 DOS 功能调用，重复显示两次。

## 实验学时

本实验内容共用 4 学时，分两次实验完成。

## 实验报告要求

本次实验报告中要求写实验内容 3 的编写、调试过程，其中包括程序设计思路、程序流程图、程序代码、和调试过程中遇到的典型问题以及解决方法。

## 实验二 分支循环结构程序练习

### 实验目的

- (1) 通过编制包含分支、循环结构的程序，练习汇编语言综合编程方法。
- (2) 练习在 Debug 状态下调试程序的方法。

### 实验内容

1、试编写程序，完成以下功能。

(1) 程序执行时，显示提示信息“Please input a string(length<9):”，由用户输入一个长度小于 9 的字符串；

(2) 然后显示提示信息“Please input the index of the char to display:”，请用户指定该串中的某个字符的位置号，程序控制用户输入的位置号必须是合法的，例如实际输入字符串长度为 5 个字符，位置号只能是 0-4 之间的数值；若位置号不合法，则程序退出。

(3) 程序将用户指定位置的字符显示出来。

#### 程序输出样例 1:

```
Please input a string(length<9): ABCDEFG
Please input the index of the char to display: 2
The char is: C
```

#### 程序输出样例 2:

```
Please input a string(length<9): ABCD
Please input the index of the char to display: 4
The index is invalid!
```

2、试使用分支结构和循环结构程序完善上面的程序功能。

(1) 采用 **01 号 DOS 功能调用**，接收用户输入的字符串，并在程序中对字符串进行长度检查，若长度大于 9，则要求用户重新输入；

(2) 用户在指定位置号时，进行判断，若输入非法（位置号大于实际字符串长度），则提示用户重新输入；若输入位置号合法，则显示字符串中该位置号对应的字符。

(3) 统计用户输入的字符串中特定字符的数目并显示。可先指定特定字符进行统计，然后再由用户指定字符进行统计。

### 程序输出样例 1:

```
Please input a string(length<9): ABCD
Please input the index of the char to display: 1
The char is: B
Please input a char: A
The count of 'A' is : 1
```

### 程序输出样例 2:

```
Please input a string(length<9): ABCDEFGHIJ
The string is too long!
Please input a string(length<9):ABCDEF
Please input the index of the char to display: 9
The index is invalid!
Please input the index of the char to display: 3
The char is: D
Please input a char: 1
The count of '1' is : 0
```

## 实验学时

本实验内容共用 6 学时，分三次实验完成。

## 实验报告要求

本次实验报告中，可只写分支结构和循环结构的程序段，报告中要包含分支、循环结构程序段的设计思路、程序流程图、程序代码、和调试过程中遇到的典型问题以及解决方法。



## 实验三 综合程序练习

### 实验目的

- 1、练习编写包含分支、循环、子程序结构的程序。
- 2、练习在 Debug 状态下跟踪调试程序的方法。

### 实验内容

根据以下题目要求，编写汇编语言源程序，并完成调试。

1、已知无符号数组 LIST 的第一个字数据为其元素个数，其后数据按大小关系排序保存。试将某无符号字数据 X 插入数组 LIST 的正确位置，并修改元素个数。

编程提示：

该程序中要将 X 插入递增数组，需要和数组中每一个数据值进行大小比较，直至找到大于 X 的元素。

数据比较的方式：（1）从前向后比较：找到合适位置后，再移动其后元素；

（2）从后向前比较：比较到不合适，直接移动该元素，直到找到合适位置，刚好腾出 X 的保存空间；

2、试用子程序实现以下要求。

题目：从键盘接收若干个一位十进制数值（0~9），并以十进制数据形式显示其和。

要求：（1）用子程序实现一位十进制数值的输入；

（2）当用户未输入数值，直接回车时，结束输入；

（3）输出的数据为多位十进制数据，而机器内部计算的和是十六进制形式，需要进行

数制转换，然后以字符串的形式输出结果；

（4）程序中要求有必要的提示信息。

例如：用户在提示信息下输入三个数值：

Please input a number: 5

Please input a number: 3

Please input a number: 4

程序显示运算结果

The sum is: 12

3、试用子程序改进以上题目。

题目：从键盘接收若干个 N 位的十进制数值(0~65535)，并以二进制、十进制、十六进制三种数制形式显示其和。

要求：(1) 用子程序实现一个 N 位十进制数值的输入，在主程序的循环结构中调用该子程序；

(2) 当用户未输入数值，直接回车时，结束输入；

(3) 输出的数据为多位十进制数据，而机器内部计算的和是十六进制形式，需要进行数制转换，然后以十进制字符串的形式输出结果；

(4) 程序中要求有必要的提示信息。

例如：用户在提示信息下输入三个数值：

Please input a number: 15

Please input a number: 30

Please input a number: 45

程序显示运算结果

The sum is: 0101 1010B

90D

5AH

## 实验学时

本实验内容共用 6 学时，分三次实验完成。

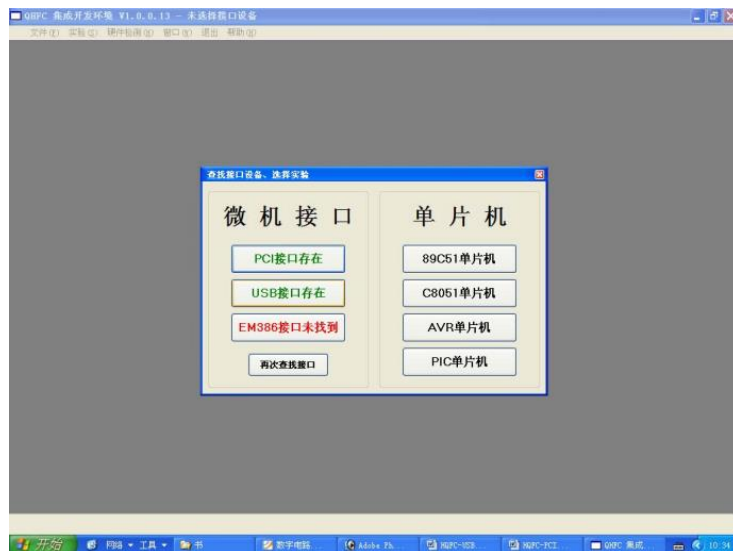
## 实验报告要求

本次实验报告中要求写实验内容 1 的编写、调试过程，其中包括程序设计思路、程序流程图、程序代码、和调试过程中遇到的典型问题以及解决方法。

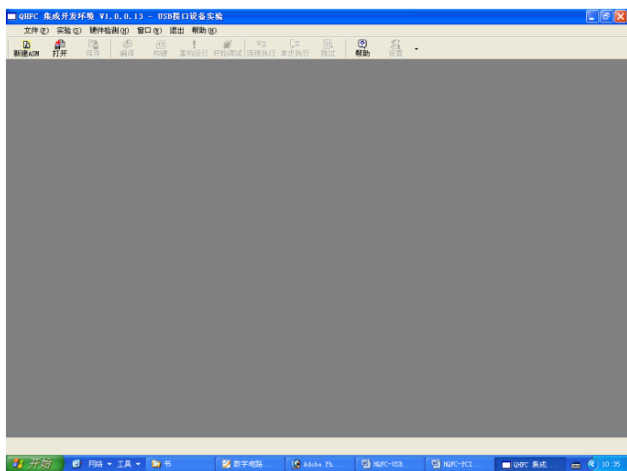
# 接口实验教材

## HQFC 集成开发环境的使用说明

- 1、运行程序/“HQFC 集成开发环境.EXE”，如下图所示

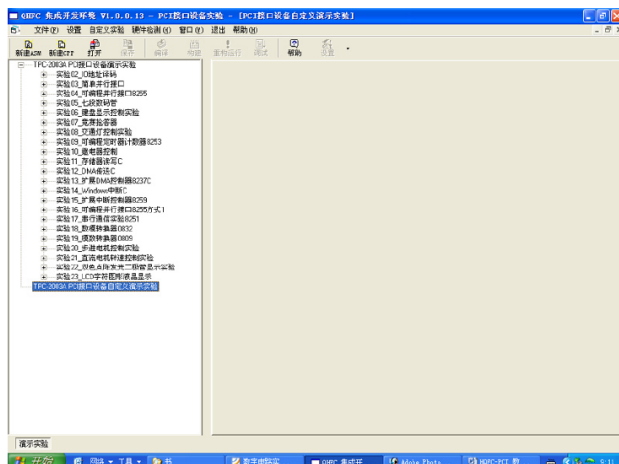


- 2、自动检测接口  
软件自动检测所安装的所有接口，如果检测到硬件显示为绿色，否则为红色。
- 3、选择接口类型  
选择 USB 接口，进入 USB 微机接口开发环境。如下图所示



#### 4、实验

点击实验/演示实验，HQPC 集成开发环境提供了部分参考实验，也可以自定义实验。



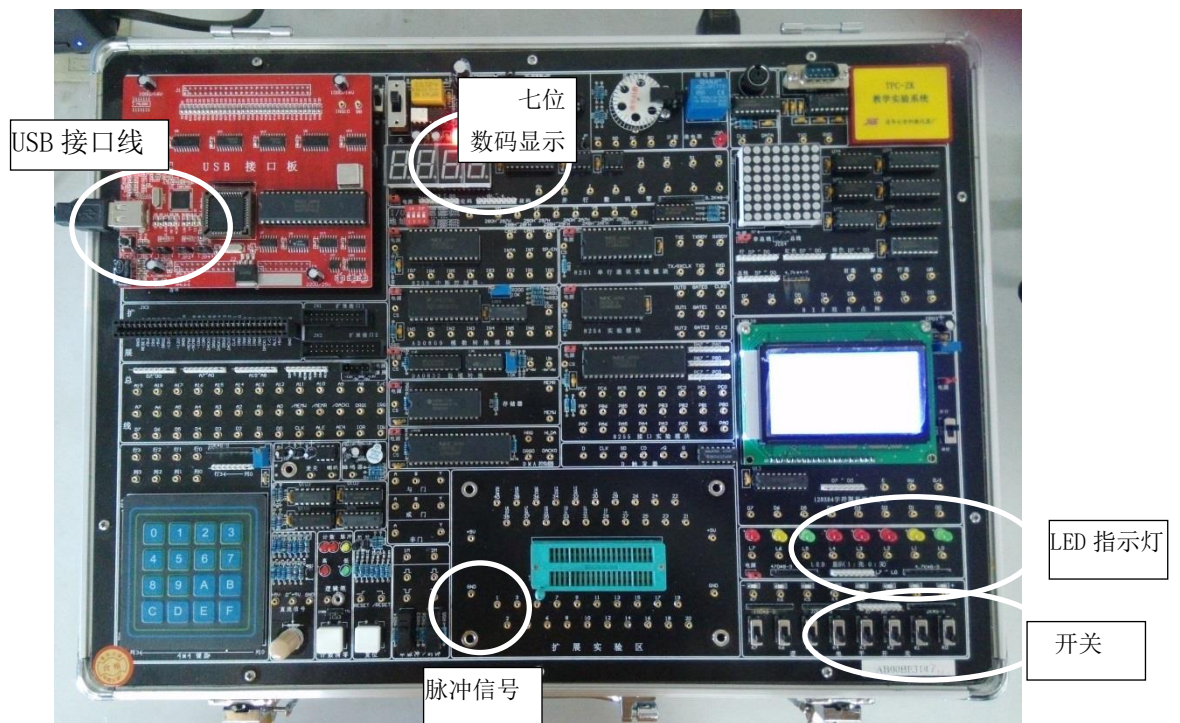
#### 5、编写 ASM 文件

选择新建 ASM 文件，编写程序或者打开 ASM 文件。

#### 6、运行

点击编译→构建→重构运行，即可在实验箱上看到实验运行的结果。

#### 7、实验箱



## 实验四 并行接口芯片 8255A 实验

### 实验目的

掌握可编程并行接口芯片 8255A 的原理和与微机的接口方法，熟悉对 8255A 的初始化编程方法，及其简单的工作方式和编程原理。

### 实验内容

#### 实验内容一：8255A 数据传送实验

利用 8255A 将逻辑电平开关  $K_1 \sim K_8$  的开关状态反映在实验仪的发光二极管上。

#### 实验内容二：七段数码管的数字显示

将 8255A 的 A 口作为数据输出端口连接七段数码管的输入端，编程使数码管上循环显示“0-9”。

### 实验学时

本实验内容共用 2 学时，一次实验完成。

### 实验步骤

#### 1、8255A 数据传送实验

##### (1) 实验原理

本实验是对 8255A 方式 0 的简单应用，即将逻辑电平开关  $K_1 \sim K_8$  的状态通过 8255A 的 C 口输入，并由 A 口输出至实验仪的发光二极管上。

➤ 实验仪上的逻辑电平开关  $K_1 \sim K_9$ ，其开关上拨，输出低电平“0”；开关下拨，输出高电平“1”。

➤ 实验仪上的发光二极管  $L_1 \sim L_{12}$  显示时，其输入端为低电平“0”时，发光二极管点亮；其输入端为高电平“1”时，发光二极管熄灭。

##### (2) 硬件连接

➤ 8255 片选信号  $CS/\overline{CS}$  接 138 译码器的 Y1/输出端（译码地址：

288—28FH)

- 8255 的 PC7—PC0      接    逻辑电平开关 K7—K0
- 8255 的 PA7—PA0      接    发光二极管 L7—L0

## 2、七段数码管的数字显示

### (1) 实验原理

七段数码管是由 7 段发光二极管和 1 个小数点组成的，这些段分别由字母 a,b,c,d,e,f,g,dp (小数点) 来表示。当数码管特定的段加上电压后，形成不同的组合，显示数字、符号和字母。

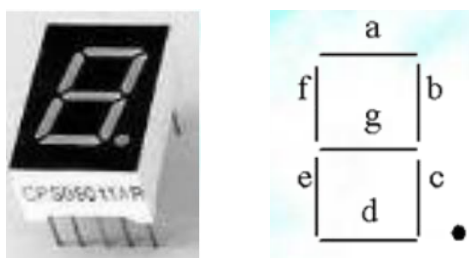


图4.2 七段数码管的外形结构

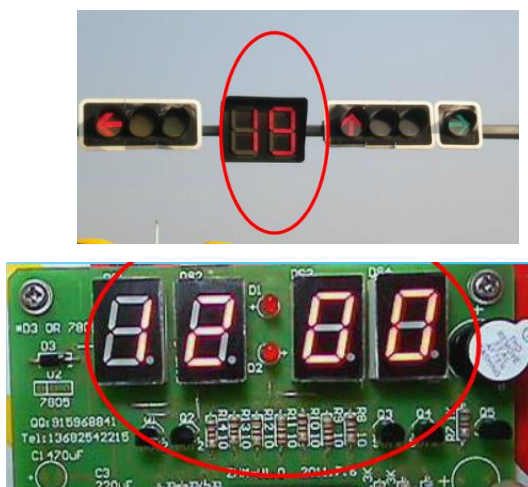


图4.3 七段数码管的应用

七段数码管分为共阴极和共阳极两种，本实验箱中的是共阴极数码管，即输入低电平点亮对应的数段，输入高电平对应的数段熄灭。



图4.4 七段数码管数字显示形式

表 4-1 七段数码管的字型代码表

显 示 字形	h	g	f	e	d	c	b	a	输 入 数据
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH
6	0	1	1	1	1	1	0	1	7DH
7	0	0	0	0	0	1	1	1	07H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	0	1	1	1	1	6FH

## (2) 硬件连接

本实验中要显示两位的数字，应先选中高位数码管显示数据，再选中低位数码管显示数据，位码由 8255 的 PC1 和 PC0 两位输出 02（高位数码管）、01（低位数码管）控制，要显示的数据段码由 8255 的 A 口输出。

➤ 8255 片选信号 CS/ 接 138 译码器的 Y1/输出端（译码地址：288—28FH）

➤ 8255 的 PA7—PA0 接 数码管的数目输入端 dp—a

➤ 8255 的 PC1—PC0 接 数码管的 S1—S0

➤ 地线 GND 接 数码管的 S3—S2

## (3) 参考实验程序流程图



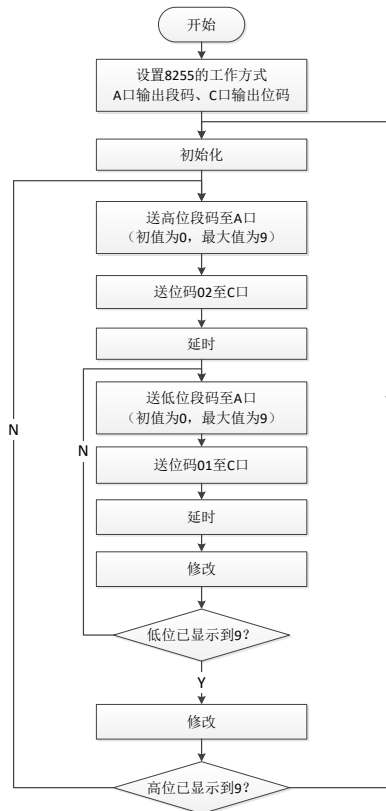


图4.5 七段数码管数字显示流程图

## 实验思考问题

1. 试考虑“竞赛抢答器”如何设计？

竞赛抢答器：逻辑开关 K 作为竞赛抢答按钮，当某组的抢答按钮按下时，在七段数码管上显示其相应的组号。

## 实验五 可编程定时器/计数器（8254）

### 实验目的

掌握 8254 的基本要作原理和编程方法。

### 实验原理和内容

- 1、 按图 5-1 虚线连接电路，将计数器 0 设置为方式 0，计数器初值为 N ( $N \leq 0FH$ )，用手动逐个输入单脉冲，编程使计数值在屏幕上显示，并同时用逻辑笔观察 OUT0 电平变化（当输入 N+1 个脉冲后 OUT0 变高电平）。

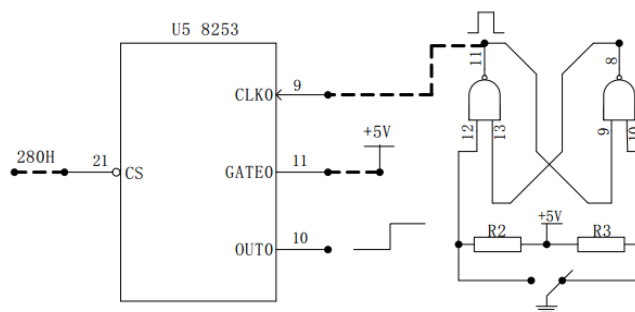


图 5-1

- 2、 按图 5-2 连接电路，将计数器 0，计数器 1 分别设置为方式 3，计数初值设为 1000，用逻辑笔观察 OUT1 输出电平的变化（频率 1Hz）。

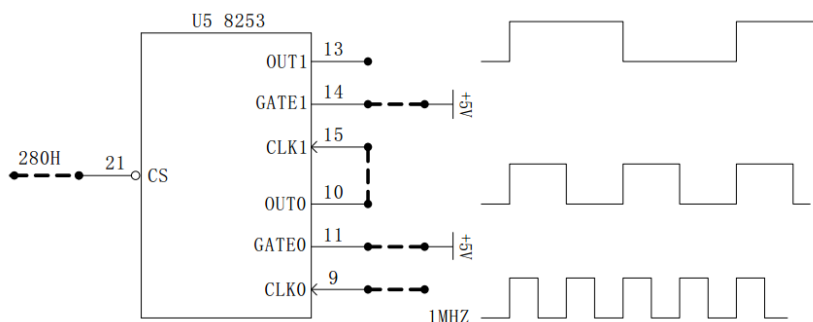


图 5-2

- 3、 接线：

- 1) CS/8254 接 Y0/I0 地址

GATE0/8254 接 +5V

CLK0/8254 接 单脉冲

2) CS/8254 接 Y0/I0 地址

GATE0/8254 接+5V

CLK0/8254 接 1M 时钟

OUT0/8254 接 CLK1/8254

GATE1/8254 接+5V

### 三、编程提示

1、8254 控制寄存器地址 283H

计数器 0 地址 280H

计数器 1 地址 281H

CLK0 连接时钟 1MHz

2、参考流程图

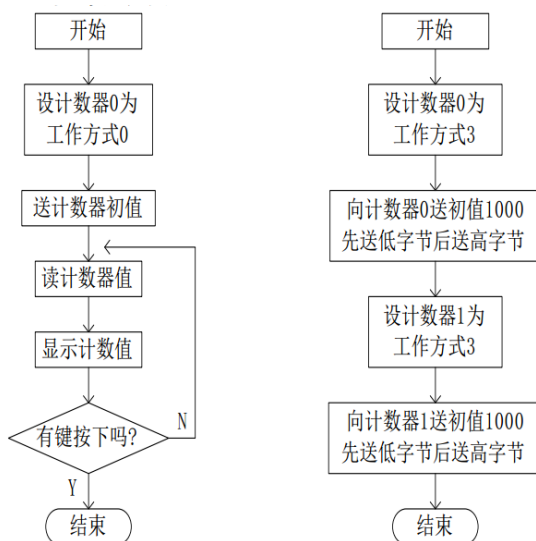


图 5-3

### 实验思考问题

由 8254 产生两组频率不同的方波，期中方波 1 为频率为方波 2 频率的两倍。

# 实验六 中断控制器 8259A 实验

(可本实验根据教学进度情况适当调整)

## 一. 实验目的

理解和掌握 8259A 中断控制器的单片使用，单级中断、多级中断嵌套时的工作原理和编程应用方法，并能综合应用于各种领域。

## 实验学时

本实验内容共用 2 学时，一次实验完成。

## 二. 实验原理和内容

### 8259A 单级中断

采用 8259A 的 IR<sub>3</sub> 中断，以边沿触发的方式，将单脉冲 2 所产生的上升沿作为中断源，每按一次脉冲按钮产生一次中断，在屏幕上显示“TPCA Interrupt! ”，当中断满 10 次时程序退出。

### 实验原理

#### 1、中断控制器 8259A 单级中断

PC 机用户可用的硬件中断只有可屏蔽中断，由 8259 中断控制器管理。中断控制器用于接收外部的中断请求信号，经过优先级判别等处理后向 CPU 发出可屏蔽中断请求。IBMPC、PC/XT 机内有一片 8259 中断控制器对外可以提供 8 个中断源：

中断源	中断类型号	中断功能
IRQ0	08H	时钟
IRQ1	09H	键盘
IRQ2	0AH	保留
IRQ3	0BH	串行口 2
IRQ4	0CH	串行口 1
IRQ5	0DH	硬盘
IRQ6	0EH	软盘
IRQ7	0FH	并行打印机

8 个中断源的中断请求信号线 IRQ0-IRQ7 在主机 62 线 ISA 总线插座中可以引

出，系统已设定中断请求信号为“边沿触发”，普通结束方式。对于 PC/AT 及 286 以上微机内又扩展了一片 8259 中断控制，IRQ2 用于两片 8259 之间级连，对外可以提供 16 个中断源：

中断源	中断类型号	中断功能
IRQ8	070H	实时时钟
IRQ9	071H	用户中断
IRQ10	072H	保留
IRQ11	073H	保留
IRQ12	074H	保留
IRQ13	075H	协处理器
IRQ14	076H	硬盘
IRQ15	077H	保留

TPC-ZK-USB 实验系统上，固定的接到了 3 号中断 IRQ3 上，即进行中断实验时，所用中断类型号为 OBH。

## 2、实验提示：

1)、中断 10 的优先级要高于 3，因为中断 10 是扩展 8259 上的中断，扩展 8259 使用主 8259 的中断 2 向主 8259 申请中断，中断 2 优先级高于中断 3，所以中断 10 的优先级要高于 3。

2)、由上所述，中断 0-15 的优先级顺序从高到低为：中断 0、中断 1、中断 8、中断 9、中断 10、中断 11、中断 12、中断 13、中断 14、中断 15、中断 3、中断 4、中断 5、中断 6、中断 7。

3)、上述中断优先级顺序中，不包含中断 2，因为中断 2 已被扩展 8259 使用从而扩展中断 8-15，所以不再出现中断 2。

## 3、接线：

单脉冲 2 接 IRQ /总线

单脉冲 1 接 IRQ10/USB 核心板

## 实验步骤

1)、中断 IRQ3 实验，直接用手动产生单脉冲 2 作为中断请求信号(只需连接一根导线)。要求每按一次开关产生一次中断，在屏幕上显示一次“TPCA Interrupt!”，中断 10 次后程序退出。

2)、中断 IRQ10 实验，用手动产生单脉冲 1 作为中断请求信号，每按一次开关产生一次中断，在屏幕上显示一次“TPCA Interrupt!”，中断 10 次后退出。（选做）

3)、中断嵌套实验，实验电路如图 4-12-1,分别用手动产生单脉冲作为中断 IRQ3 和 IRQ10 的请求信号，申请中断 IRQ3 后，进入中断 3 程序，再申请高级中级 IRQ10。（选做）

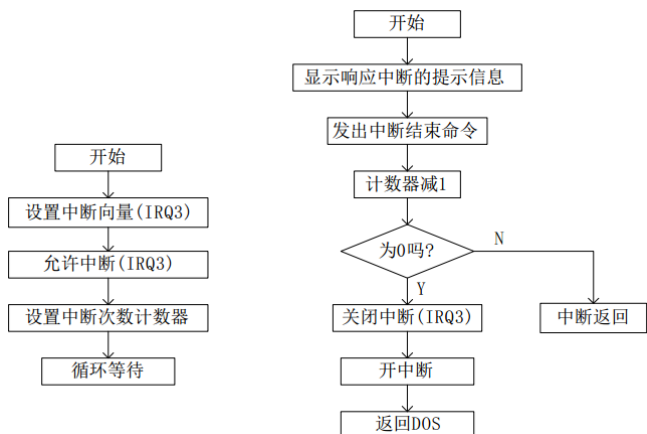


图 6-1

## 实验思考问题

- 1、若将 8259A 的片选 CS#端连接在 138 译码器 Y<sub>0</sub>#输出端, 则应如何更改程序?
- 2、若将中断源改为 IR5, 则如何更改连接和程序才能实现同样的功能?

## 实验思考问题

- 1、在实验内容一中,
  - (1)若将 8259A 的片选 CS#端连接在 138 译码器 Y<sub>0</sub>#输出端, 则应如何更改程序?
  - (2)若将中断源改为 IR5, 则如何更改连接和程序才能实现同样的功能?
- 2、在实验内容二中,
  - (1)若采用两个中断源, 则应如何修改该实验的硬件连接和软件编程? 并观察中断嵌套现象。
  - (2)试使用 8255A 方式 1 下的 INTR 请求作为中断源完成该实验, 观察是否能够达到同样的实验效果?
- 3、在实验内容三中, 试用 8255A 增加对计时启停的控制功能。
 

提示: 假定采用 PA0 连接逻辑开关, PB0 连接 8253 的 GATE 门控端, 在实验程序中增加对 8255A 的控制部分。在实验程序运行时, 当逻辑电平开关下拨, 为高电平时, LED 数码管显示器上开始循环显示 1~6 的计时结果; 当逻辑电平开关上拨, 为低电平时, LED 数码管显示器上的时间显示不再改变, 停止计时。