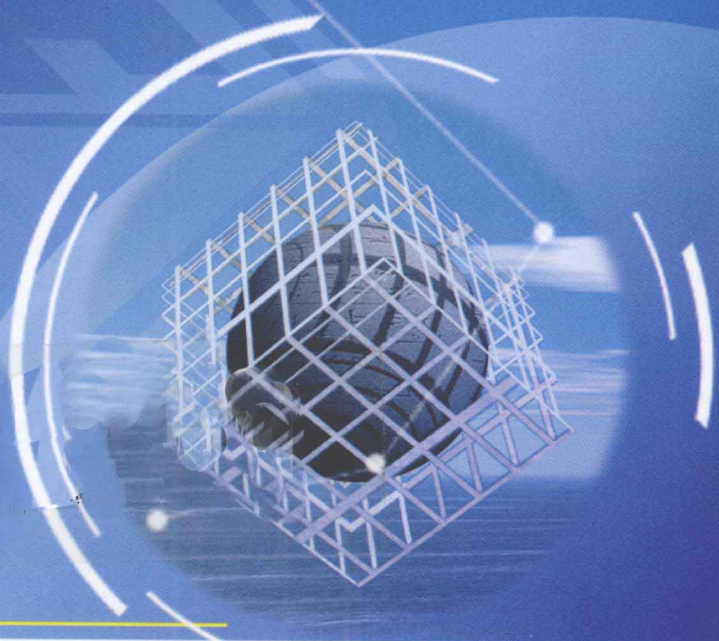
微机原理与接口技术实验
——基于 Proteus 仿真

Computer

胡建波　编著



机械工业出版社
CHINA MACHINE PRESS

微机原理与接口技术实验
——基于 Proteus 仿真

胡建波　编著

机械工业出版社

本书是基于 Proteus 软件平台的“微机原理与接口技术”仿真实验教材，书中提供了完整的实验参考程序和模拟仿真电路，特别适合学生学习使用，有助于提高学生的自主创新能力，从而使学生不再是被动地做实验，而是实验的设计者。本书的主要内容包括：emu8086 汇编指令应用、汇编语言程序设计实验、Proteus 操作基础、接口技术实验和高级 C 语言程序设计。

本书可作为职业院校计算机应用、工业自动化、电子与通信等相关专业的实训教材，也可作为工程技术人员的参考用书。

图书在版编目（CIP）数据

微机原理与接口技术实验：基于 Proteus 仿真/胡建波编著．—北京：机械工业出版社，2011.8

ISBN 978-7-111-35065-1

Ⅰ.①微… Ⅱ.①胡… Ⅲ.①微型计算机-理论-高等职业教育-教材 ②微型计算机-接口-高等职业教育-教材 Ⅳ.①TP36

中国版本图书馆 CIP 数据核字（2011）第 115684 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：陈玉芝 责任编辑：林运鑫

版式设计：霍永明 责任校对：闫玥红

封面设计：张 静 责任印制：杨 曦

北京中兴印刷有限公司印刷

2011 年 8 月第 1 版第 1 次印刷

184mm×260mm · 13.75 印张 · 334 千字

0 001—3 000 册

标准书号：ISBN 978-7-111-35065-1

定价：28.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

社服务中心：(010)88361066

销售一部：(010)68326294

销售二部：(010)88379649

读者购书热线：(010)88379203

网络服务

门户网：http://www.cmpbook.com

教材网：http://www.cmpedu.com

封面无防伪标均为盗版

前　言

“微机原理与接口技术”是一门实践性很强的课程，学习时必须理论联系实际，亲自动手做实验，才能达到预期效果。在 Proteus 仿真软件出现前，传统的实验教学一般都要在实验箱上完成，学生只有在上实验课时才能动手进行实验操作，不仅灵活性差，硬件电路不便改动，而且也不利于提高创新能力。Proteus 从 7.5 版开始提供了 VSM (Virtual System Modeling) for 8086 模块，为实验教学提供了理想的实验平台。随着计算机技术的发展，利用虚拟软件进行电路设计与仿真已经成为现代电子技术系统设计的必然趋势。Proteus 仿真软件丰富的元器件模型、多处理器的支持、多样的虚拟仪器、强大的图表分析功能和第三方集成开发环境，已成为电类实践教学与科研的巨大资源。

本书共分为 5 章。第 1 章介绍了 emu8086 的使用并重点练习 8086 指令系统。emu8086 具有直观的操作窗口，有利于理解 8086 CPU 的内部结构及操作数的寻址方式。其中的指令练习多是以任务驱动方式给出一些实例，使同学们在做中学，在学中做。当读得多了，写得多了以后，所用的指令就会逐渐地被深入理解。第 2 章介绍了汇编语言程序设计实验，目的是使学生掌握从汇编语言源程序到可执行程序的步骤。第 3 章介绍了 Proteus 操作基础，主要讲述了 Proteus 的使用及第三方程序开发编译工具。第 4 章介绍了接口技术实验，对本章中的每种接口都给出了多个实例，如 LED 控制循环彩灯、LED 数码管可调时电子钟、LED 点阵显示汉字、LCD 液晶显示器显示汉字等。在学习时先不要改动每个实例的硬件仿真电路，应把程序成功调试通过，对接口电路有了初步的理解后，再试着改变或设计电路及编制程序，进一步提高程序的编写及硬件接口电路的设计能力。第 5 章介绍了高级 C 语言程序设计，C 语言更接近于工程实际，具有丰富的函数接口，可极大地减少编程工作量，是现代机电类工程师进行嵌入式开发必备的编程语言工具。本章内容还可作为课程设计或毕业设计的参考文献。

因仿真软件中部分元器件符号与现行国家标准图形符号不一致，而为了便于学习，书中均采用仿真软件中的元器件符号。

由于编者水平有限，加之时间仓促，书中难免会有错误和不妥之处，恳请广大读者批评指正，可通过 E－mail 与作者联系：hjb372@sohu.com。

编　者

目　录

第 1 章　emu8086 汇编指令应用

1.1　emu8086 汇编软件

1. 实验任务

在屏幕第 10 行 20 列上显示“HELLO WORLD!”。

2. 实验目的

通过实例熟练掌握 emu8086 汇编软件实验环境，观察和分析在不同的寻址方式下存储单元逻辑地址的表示以及指令的执行结果。

3. 字符显示原理

在 25×80 彩色字符模式下，从第 0 行 0 列到第 24 行 79 列共有 2000 个字符位置。对于屏幕上的每个字符，内存的显示缓冲区中都有相应的存储单元与之相对应。在内存的 B8000H ~ BFFFFH 共 32KB 的区域是 25×80 彩色字符模式的显示缓冲区。如果向这个地址范围的单元中写入数据，写入的字符就可立即显示在屏幕上。屏幕上显示的每个字符用两个字节表示，第一个字节为字符的 ASCII 码，第二个字节为字符的属性。字符的属性代表了字符的显示特性，如颜色、闪烁、高亮反相显示等。

（1）彩色字符显示属性　彩色字符的背景色有 8 种颜色，前景色有 16 种颜色，其字符属性格式如图 1-1 所示。前景色由 4 位（D_0 ~ D_3位）组成，背景色由 3 位（D_4 ~ D_6位）组成；最高位 BL 表示闪烁，RGB 为红、绿、蓝，I 表示亮度。由表 1-1 可知，02H 表示黑底绿字；1EH 表示蓝底黄字，应避开前景色、背景色一致，否则字符看不出来。

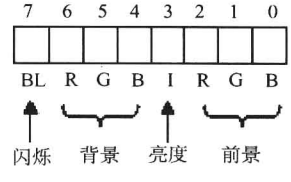


图 1-1　字符属性格式

表 1-1　字符显示属性

IRGB	颜色	IRGB	颜色
0000	黑	1000	灰
0001	蓝	1001	浅蓝
0010	绿	1010	浅绿
0011	青	1011	浅青
0100	红	1100	浅红
0101	品红	1101	浅品红
0110	棕	1110	黄
0111	灰白	1111	白

（2）字符显示位置　对于 25×80 彩色字符模式，一屏字符需占用 4000B，因此 32KB 的显示缓冲区分为 8 页，每页 4KB。B8000H ~ B8F9FH（4000B）为第 0 页的内容。由于一

行有 80 个字符，共占用 160B（A0H），因此显存单元和显示器中行的对应关系为：000H ~ 09FH 单元对应显示器上的第 0 行；0A0H ~ 13FH 单元对应显示器上的第 1 行；140H ~ 1DFH 单元对应显示器上的第 2 行。

内存中每两个字节对应一个字符，高字节为字符的属性，低字节为字符的 ASCII 码。偶地址单元存放字符的 ASCII 码，奇地址单元存放字符的属性。字符的位置计算公式为：位置 = 行号 ×160 + 列号 ×2。

为了简化程序设计，本例用顺序程序设计方法，根据上述任务要求写出相应的汇编指令。在上机时可先不看注释前面的指令，自己先试着编写一下再作对比。

4. 参考程序

```
MOV   AX, 0B800H          ;立即数 0B800H 送到寄存器 AX
MOV   DS, AX              ;寄存器 AX 内容送到寄存器 DS
MOV   ES, AX              ;寄存器 AX 内容送到寄存器 ES
MOV   AL, 'H'             ;字符'H'送到寄存器 AL
MOV   AH, 1EH             ;颜色属性值 1EH(蓝色背景黄色字体)送到寄存器 AH
                          ;第 10 行 20 列显存地址偏移量:10 ×160 +20 ×2 =1640 =
                           0668H
MOV   [0668H], AX         ;寄存器 AX 内容送到数据段直接地址 0668H

MOV   AL, 'E'             ;字符'E'送到寄存器 AL
MOV   AH, 02H             ;02H(黑底绿字)送到寄存器 AH
MOV   ES:[066AH], AX      ;寄存器 AX 内容送附加段直接地址 066AH

MOV   BX, 066CH           ;066CH 送到寄存器 BX
MOV   AL, 'L'             ;字符'L'送到寄存器 AL
MOV   AH, 02H             ;02H(黑底绿字)送到寄存器 AH
MOV   [BX], AX            ;AX 寄存器内容送到数据段 BX 间接寻址

MOV   AL, 'L'             ;字符'L'送到寄存器 AL
MOV   2[BX], AX           ;AX 寄存器内容送到数据段 BX +2 相对间接寻址

MOV   SI, 0004H           ;立即数 0004H 送到寄存器 SI
MOV   AL, 'O'             ;字符'O'送到寄存器 AL
MOV   ES:[BX + SI], AX    ;寄存器 AX 内容送到附加段 BX + SI 基址变址寻址

MOV   AL, 'W'             ;字符'W'送到寄存器 AL
MOV   ES:4[BX + SI], AX   ;寄存器 AX 内容送到附加段 4 + BX + SI 相对基址变址
                           寻址
ADD SI, 2                 ;SI +2 →SI
MOV   AL, 'O'             ;字符'O'送到寄存器 AL
```

```
MOV   ES:4[BX+SI], AX     ;寄存器 AX 内容送到附加段 4+BX+SI 相对基址变址寻址

MOV   BP, 0678H           ;立即数 0678H 送到寄存器 BP
MOV   AL, 'R'             ;字符'R'送到寄存器 AL
MOV   DS:[BP], AX         ;寄存器 AX 内容送到数据段 BP 间接寻址
MOV   AL, 'D'             ;字符'D'送到寄存器 AL
MOV   ES:2[BP], AX        ;寄存器 AX 内容送到附加段 BP 相对间接寻址
MOV   AL, '!'             ;字符'!'送到寄存器 AL
MOV   ES:4[BP], AX        ;寄存器 AX 内容送到附加段 BP 相对间接寻址
HLT
```

5. 实验步骤

1）直接双击桌面上的 emu8086 快捷图标进入 emu8086 汇编语言编辑窗口，图 1-2 所示为其编辑窗口菜单，选择“new”图标，在弹出的窗口中选择“empty workspace”，新建一个空的工作区。

2）输入上述源程序。特别强调的是可先输入两三条指令直接单击图 1-2 中“emulate”图标进入调试环境，单步执行，观察指令执行时 CPU 寄存器的变化情况。由于刚开始对汇编指令格式掌握还不太牢固，最好不要一次将全部指令输入完毕，否则不易发现错误。

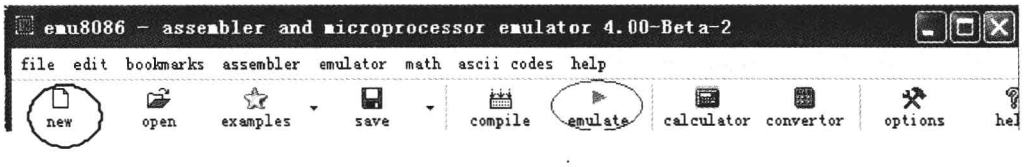


图 1-2　emu8086 编辑窗口菜单

3）待程序全部输入完，可存盘为“hello. asm”，单击图 1-2 中“compile”图标进行编译，若没有错误，编译通过弹出可执行文件保存对话框，存盘为“hello. exe”。在新弹出的图 1-3 所示对话框中单击“external”图标选择“run”全速执行。需要说明的是，步骤 2）中的 emulate 调试环境中也有 run 功能但速度较慢。

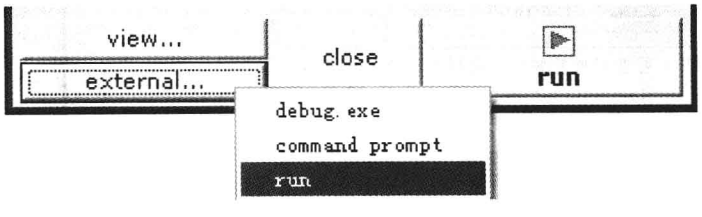


图 1-3　emu8086 外部命令窗口

4）调试。单击“emulate”图标（见图 1-2）进入调试窗口（见图 1-4），单步执行并观察 CPU 寄存器值、指令的物理地址、逻辑地址、机器码等变化；打开堆栈窗口（见图 1-5）、附加段窗口（见图 1-6）、debug 命令窗口（见图 1-7）、flags 标志位窗口（见图 1-8），单步执行观察其数据的变化，并分析每一条指令执行后的结果是否与要求一致。

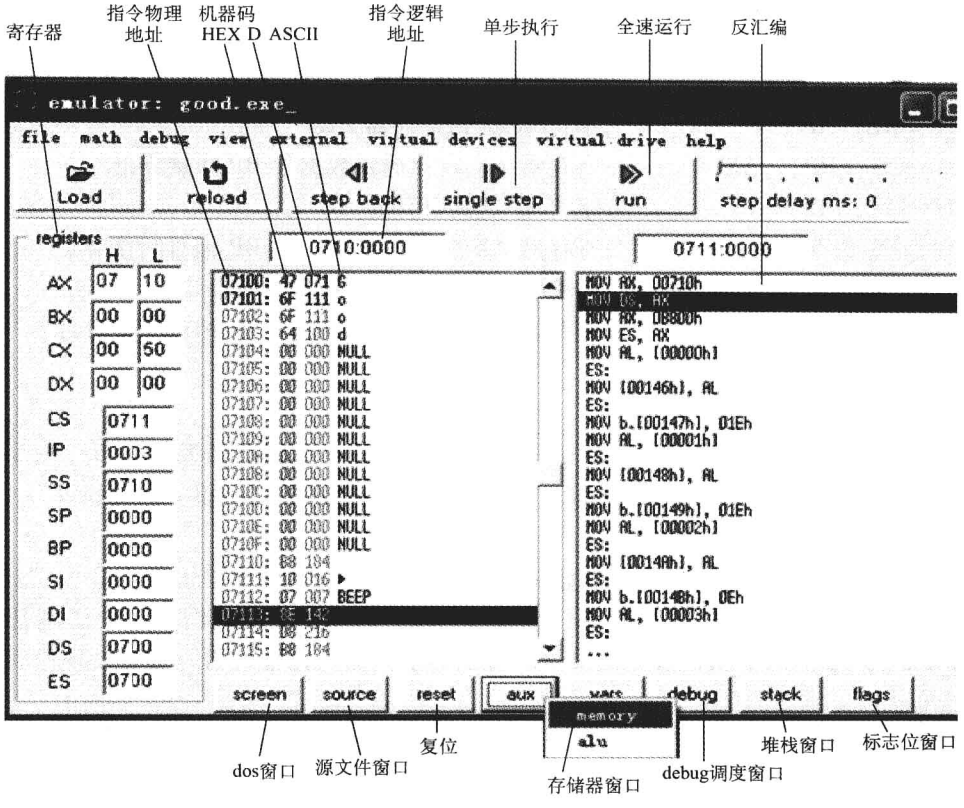


图 1-4 emu8086 调试窗口

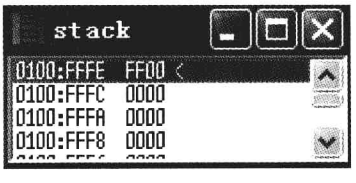


图 1-5 堆栈窗口

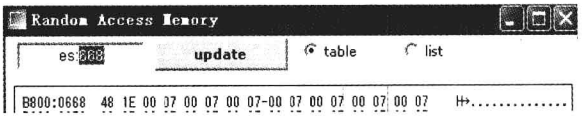


图 1-6 附加段窗口

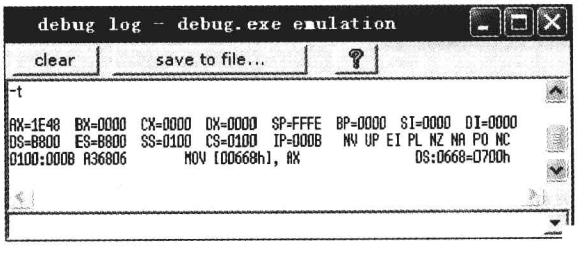


图 1-7 debug 命令窗口

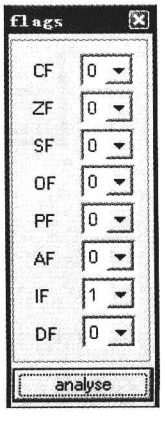


图 1-8 flags 标志位窗口

6. 练习

1）设当程序执行到最后一务指令“MOV　ES：4［BP］，AX”时，debug 状态如下：

0100：004C 26　　ES：

AX=0221　BX=066C　CX=0000　DX=0000　SP=FFFE　BP=0678　SI=0006　DI=0000

DS=B800　ES=B800　SS=0100　CS=0100　IP=004D　NV UP EI PL NZ NA PE NC

0100：004D 894604　　MOV［BP］+ 04H，AX　　ES：067C=0700H

附加段状态如下：

B800:0668　48 1E 45 02 4C 02 4C 02 -4F 02 00 07 57 02 4F 02

B800:0678　52 02 44 02 00 07 00 07 -00 07 00 07 00 07 00 07

从上述指令 debug 状态中可知当前指令使用了段超越，使用附加段寄存器 ES 作为目的操作数的段基址，附加段基址为_____；当前指令的 IP 值为_____，物理地址为_____；当前指令长度为_____B（不含 ES），其中机器码中的 04 表示目的操作数的_____，BP 寄存器的值为_____，两者的和为目的操作数的有效地址，目的操作数的逻辑地址为_____，物理地址为_____。

2）在屏幕的第 4 行 25 列以红色字体黄色背景显示：“ONE WORLD! ONE DREAM!”。

```
    MOV AX,0B800H              ;显示缓冲区首地址
    MOV ES,AX                  ;附加段基址
    MOV SI,OFFSET ASC          ;取 ASC 有效地址送 SI 寄存器
    MOV BX,4*160+25*2          ;第 4 行 25 列显存地址
    MOV AH,0ECH                ;0ECH 表示黄底红字
    MOV CX,21                  ;ASC 字符串长度
LP:MOV AL,[SI]                 ;取字符
    MOV ES:[BX],AX             ;送显示缓冲区
    INC SI                     ;字符串地址加 1
    ADD BX,2                   ;待显示字符串 ASC 的下一列显存地址
    LOOP LP                    ;循环
    HLT
    ASC DB 'ONE WORLD! ONE DREAM!'
```

1.2　emu8086 汇编指令应用

本节继续通过一些实例加深对 emu8086 汇编指令的理解与应用，初步掌握汇编语言程序的编写方法。

例 1-1　在屏幕的第 8 行 20 列开始显示如图 1-9 所示一个宽 40、高 10 行的红色㊀窗口。

图 1-9　红色窗口

1）参考程序。

㊀因图书为黑白印刷，在图中显示为深灰色，仿真软件中显示为红色。

```
WINWIDTH    = 40                          ;窗口宽度
HEIGHT      =10
WINTOP      = 8                           ;窗口左上角行号
WINLEFT     =20                           ;窗口左上角列号
WINBOTTOM   =WINTOP + HEIGHT - 1          ;窗口右下角行号
WINRIGHT    =WINLEFT + WINWIDTH - 1       ;窗口右下角列号
COLOR       = 47H                         ;47H 红屏白字
        MOV  AX,0B800H                    ;显示缓冲区首址
        MOV  DS,AX
        MOV  BX,WINTOP * 160              ;BX 用作行指针
LP0:    MOV  SI,WINLEFT * 2               ;SI 用作列指针
LP:     MOV  BYTE PTR 1[BX + SI],COLOR    ;颜色属性送到显示缓冲区
        CMP  SI,WINRIGHT * 2              ;是否到窗口最后一列
        JE   NEXTLINE
        INC  SI
        INC  SI
        JMP  LP
NEXTLINE:CMP  BX,WINBOTTOM * 160          ;是否到窗口最后一行
        JE   EXITBOTTOM
        ADD  BX,160
        JMP  LP0
EXITBOTTOM:HLT
```

2）练习。单步执行上述程序，查看 debug 状态，指令“MOV BX, WINTOP * 160”的逻辑地址、机器码、反汇编指令分别是_____、_____、_____，机器码中后两个字节是_____；LP 标号的逻辑地址是_____；当程序执行到 JE NEXTLINE 指令时，其指令逻辑地址、机器码、反汇编指令分别是_____、_____、_____，该指令转移的条件标志是_____；JMP LP 指令其机器码中的相对量是_____；JE EXITBOTTOM 指令其机器码中的相对量是_____；JMP LP0 指令其机器码中的相对量是_____。

例 1-2　在屏幕的第 8 行 20 列开始显示如图 1-10 所示一个高 10 行的红色直角三角形窗口。

```
HEIGHT      =10
WINTOP      = 8                           ;窗口左上角行号
WINLEFT     =20                           ;窗口左上角列号
WINBOTTOM   =WINTOP + HEIGHT - 1          ;窗口右下角行号
COLOR       = 47H                         ;47H 红屏白字

        MOV  AX,0B800H                    ;显示缓冲区首址
        MOV  DS,AX
```

图 1-10　红色直角三角形窗口（一）

```
        MOV BX,WINTOP*160              ;BX 用作行指针
        MOV CX,WINLEFT*2               ;列尾
LP0:    MOV SI,WINLEFT*2               ;SI 用作列指针
LP:     MOV BYTE PTR 1[BX+SI],COLOR    ;颜色属性送到显示缓冲区
        CMP SI,CX                      ;是否到窗口最后一列
        JE  NEXTLINE
        INC SI
        INC SI
        JMP LP
NEXTLINE:CMP BX,WINBOTTOM*160          ;是否到窗口最后一行
        JE  EXITBOTTOM
        ADD BX,160
        INC CX
        INC CX
        JMP LP0
EXITBOTTOM:HLT
```

例 1-3　在屏幕的第 8 行 20 列开始显示如图 1-11 所示一个高 10 行的红色直角三角形窗口。

图 1-11　红色直角三角形窗口（二）

```
WINWIDTH   = 10                        ;窗口宽度
HEIGHT     =10
WINTOP     = 8                         ;窗口左上角行号
WINLEFT    =20                         ;窗口左上角列号
WINBOTTOM  = WINTOP + HEIGHT - 1       ;窗口右下角行号
WINRIGHT   = WINLEFT + WINWIDTH        ;窗口右下角列号
COLOR      = 47H                       ;47H 红屏白字
DATA       SEGMENT
ASC        DB 30H,31H,32H,33H,34H,35H,36H,37H,38H,39H
DATA       ENDS
CODE       SEGMENT
START:       MOV AX,DATA
        MOV DS,AX                      ;加载数据段基址
        MOV AX,0B800H                  ;显示缓冲区首址
        MOV ES,AX
        MOV BX,WINTOP*160              ;BX 用作行指针
        MOV CX,WINLEFT*2               ;列尾
LP0:     MOV DI,0                      ;字符指针
        MOV SI,WINLEFT*2               ;SI 用作列指针
LP:      MOV AL,ASC[DI]
        MOV AH,COLOR
```

```
        MOV  ES:[BX+SI],AX          ;ASC 码及颜色属性送显示缓冲区
        CMP  SI,CX                  ;是否到窗口最后一列
        JE   NEXTLINE
        INC  SI
        INC  SI
        INC  DI                     ;下一字符
        JMP  LP
NEXTLINE:CMP  BX,WINBOTTOM*160      ;是否到窗口最后一行
        JE   EXITBOTTOM
        ADD  BX,160
        INC  CX
        INC  CX
        JMP  LP0
EXITBOTTOM:HLT
CODE    ENDS
        END  START
```

例 1-4　在屏幕的第 8 行 20 列开始显示如图 1-12 所示一个高 10 行的红色倒立直角三角形窗口。

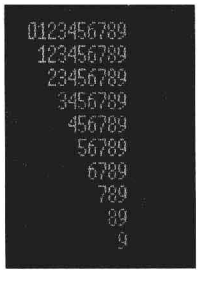


图 1-12　红色倒立直角三角形窗口

```
WINWIDTH    =  10                   ;窗口宽度
HEIGHT      =10
WINTOP      =  8                    ;窗口左上角行号
WINLEFT     =20                     ;窗口左上角列号
WINBOTTOM   =WINTOP+HEIGHT-1        ;窗口右下角行号
WINRIGHT    =WINLEFT+WINWIDTH-1     ;窗口右下角列号
COLOR       =  47H                  ;47H 红屏白字
DATA    SEGMENT
ASC     DB 30H,31H,32H,33H,34H,35H,36H,37H,38H,39H
DATA    ENDS
CODE    SEGMENT
ST:     MOV  AX,DATA
        MOV  DS,AX                  ;加载数据段基址
        MOV  AX,0B800H              ;显示缓冲区首址
        MOV  ES,AX
        MOV  BX,WINTOP*160          ;BX 用作行指针
        MOV  CX,WINLEFT*2           ;列头指针
LP0:    MOV  SI,WINLEFT*2           ;SI 用作列指针
        MOV  DI,0
LP:     CMP  SI,CX
        JB   NEXT_COL               ;小于列头,转 NEXT_COL
```

```
            MOV  AL,ASC[DI]
            MOV  AH,COLOR
            MOV  ES:[BX+SI],AX              ;颜色属性送到显示缓冲区
            CMP  SI,WINRIGHT*2              ;是否到窗口最后一列
            JE   NEXTLINE
NEXT_COL:INC  SI
            INC   SI
            INC   DI
            JMP   LP
NEXTLINE:CMP  BX,WINBOTTOM*160              ;是否到窗口最后一行
            JE   EXITBOTTOM
            ADD  BX,160
            INC  CX
            INC  CX
            JMP  LP0
EXITBOTTOM:HLT
CODE  ENDS
    END  ST
```

例 1-5　在屏幕的第 3 行 20 列（160×3+20×2=520）循环显示 0～9。

1）参考程序。

```
CODE   SEGMENT
       ASSUME   CS:CODE,DS:DATA
START:MOV  AX,DATA
       MOV  DS,AX
       MOV  AX,0B800H                       ;数据段基址
       MOV  ES,AX                           ;显示缓冲区首地址
LP0:   MOV  SI,0                            ;0～9
       MOV  CX,10
LP:    MOV  AL, VAR[SI]                     ;ASCII 码
       MOV  ES:[520],AL                     ;ASCII 码送到显示缓冲区
       CALL DEL                             ;延时
       INC SI
       MOV   AH,0BH                         ;Ctrl+C 退出
       INT   21H
       LOOP LP
       JMP   LP0
 DEL   PROC                                 ;延时
       PUSH   BX
```

```
        PUSH   CX
        MOV    BX,0400H
LP2:    MOV    CX,4000H
LP1:    PUSHF
        POPF
        LOOP   LP1
        DEC    BX
        JNZ    LP2
        POP    CX
        POP    BX
        RET
DEL     ENDP
CODE    ENDS
DATA    SEGMENT
VAR     DB '0123456789'
DATA    ENDS
        END   START
```

2）调试说明。在 emu8086 中输入上述程序，存盘为“num. asm”，调试时，子程序中的“MOV BX, 0400H”和“LP2：MOV CX, 4000H”两条指令中的立即数可以先赋值 1，调试通过后再改为原值，或者先用一个空的（仅有 RET 指令）延时子程序。上述程序调试通过后编译生成 num. exe，在新弹出的对话框中单击“external”图标选择“run”，观察程序执行效果。

3）练习。单步执行上述程序，查看 debug 和 stack 状态，指令 CALL　DEL 的逻辑地址、机器码、反汇编指令分别是_____、_____、_____，指令执行前 SP 的值是_____，指令 CALL DEL 执行后 SP 的值是_____，栈顶内容是_____，子程序 DEL 的逻辑地址是_____。

RET 指令的 IP 地址是_____，RET 指令执行前 SP 的值是_____，栈顶内容是_____，RET 指令执行后将栈顶内容送入_____并将 SP 值加_____，程序返回主程序继续运行。

INT 21H 指令的逻辑地址、机器码、反汇编指令分别是_____、_____、_____，指令执行前 SP 的值是_____，执行指令 INT 21H 进入软中断服务程序后 SP 的值是_____，栈顶的 3 个字内容是（由高到低）_____、_____、_____，分别是进入中断前的_____标志、程序返回指令的_____、_____。

例 1-6　如图 1-13 所示，设外部有一个 16 位输出端口，地址为 4，用其低 12 位分别控制东西南北四个方向的交通灯。

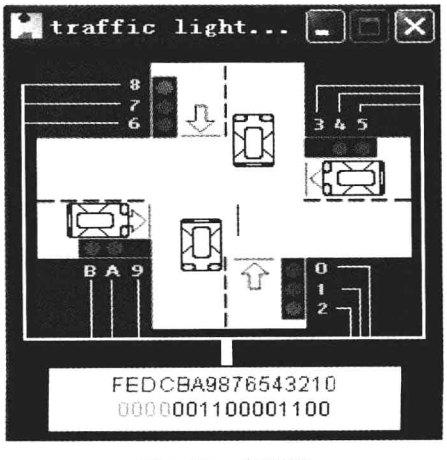


图 1-13　交通灯

1）任务分析。由图 1-13 可知，0、1、2 位分

别控制路口南边的红、黄、绿 3 个交通灯；3、4、5 位分别控制路口东边的红、黄、绿 3 个交通灯；6、7、8 位分别控制路口北边的红、黄、绿 3 个交通灯；9、A、B 位分别控制路口西边的红、黄、绿 3 个交通灯；当某位为 1 时，对应于该位交通灯点亮，为 0 时熄灭。设交通信号有如下 4 个状态：

① 状态 1 为南北绿、东西红；延时 20s；控制字为：0000_ 001_ 100　_ 001_ 100B。

② 状态 2 为南北黄、东西红黄；延时 5s；控制字为：0000_ 011_ 010　_ 011_ 010B。

③ 状态 3 为南北红、东西绿；延时 20s；控制字为：0000_ 100_ 001　_ 100_ 001B。

④ 状态 4 为南北红黄、东西绿；延时 5s；控制字为：0000_ 010_ 011　_ 010_ 011B。

2）参考程序。

```
#START = TRAFFIC_LIGHTS. EXE# ;模拟交通灯
NAME "TRAFFIC"
AGAIN:MOV SI, OFFSET  S1                    ;状态 1
      MOV AX, [SI]
      OUT 4, AX
      CALL DEL20s                           ;延时 20s
      ADD SI, 2                             ;状态 2
      MOV AX, [SI]
      OUT 4, AX
      CALL DEL05s                           ;延时 5s
      ADD SI, 2                             ;状态 3
      MOV AX, [SI]
      OUT 4, AX
      CALL DEL20s                           ;延时 20s
      ADD SI, 2                             ;状态 4
      MOV AX, [SI]
      OUT 4, AX
      CALL DEL05s                           ;延时 5s
      JMP  AGAIN                            ;返回状态 1
;                        FEDC_BA98_7654_3210
S1              DW       0000_0011_0000_1100B
S2              DW       0000_0110_1001_1010B
S3              DW       0000_1000_0110_0001B
S4              DW       0000_0100_1101_0011B
DEL20s PROC                                 ;延时 20s 子程序
       MOV      CX, 0131H    ;01312D00H = 20,000,000
       MOV      DX, 2D00H
       MOV      AH, 86H
       INT      15H
```