

**宣城校区**

**课 程 设 计 报 告**

**课 程 设 计 名 称 微机原理与接口技术**

**课 程 设 计 题 目 简易电子琴、LED 16×16点阵**

**院 系**

**专 业 班 级**

**学 生 姓 名**

**学 号**

**指 导 教 师**

**实 验 地 点 计算中心103**

**完 成 日 期 2019 年 12 月 28 日**

目 录

[设计1：简易电子琴 1](#_Toc28803370)

[1. 设计任务 1](#_Toc28803371)

[2. 设计目标 1](#_Toc28803372)

[3. 设计原理 1](#_Toc28803373)

[4. 设计实现 2](#_Toc28803374)

[4.1. 硬件实现 2](#_Toc28803375)

[4.2. 软件实现 3](#_Toc28803376)

[5. 设计结果 6](#_Toc28803377)

[6. 设计心得与改进方法 6](#_Toc28803378)

[设计2：LED 16×16点阵 7](#_Toc28803379)

[1 设计任务 8](#_Toc28803380)

[2 设计目标 8](#_Toc28803381)

[3 设计原理 8](#_Toc28803382)

[3.1 A2区 9](#_Toc28803383)

[3.2 A3区 9](#_Toc28803384)

[3.2.1 CPU总线 10](#_Toc28803385)

[3.2.2 片选区 10](#_Toc28803386)

[4 设计实现 10](#_Toc28803387)

[4.1 硬件实现 10](#_Toc28803388)

[4.2 软件实现 11](#_Toc28803389)

[4.2.1 示例代码分析 13](#_Toc28803390)

[4.2.2 重构代码分析 14](#_Toc28803391)

[5 设计结果 15](#_Toc28803392)

[6 设计心得与改进方法 17](#_Toc28803393)

[附录A 19](#_Toc28803394)

[附录B 28](#_Toc28803395)

设计1：简易电子琴

# 设计任务

掌握蜂鸣器的使用方法；掌握蜂鸣器的不同发音的方法。

# 设计目标

借助可编程并行接口芯片8255，通过PC机编程在SUN ES86PCIU+实验仪平台上实现了一个简易的电子琴。

# 设计原理

1. 对蜂鸣器输入不同频率的方波，会发出各个音阶的声音；
2. 通过编程设定或按键，由8255芯片控制发出不同频率的方波，即产生不同音阶的声音。

我们主要采用了可编程并行接口芯片8255，通过8255的PAD,使实验仪F5区的1〜7号键由低到高发出1-7的音阶。

设计有关的原理图如图3.1所示。

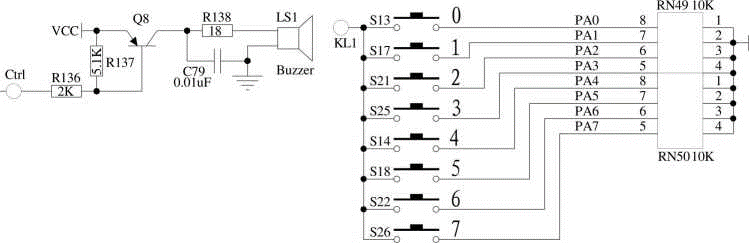


图 3.１简易电子琴实验原理图

主机连线说明如表3.1所示，连线的目的是编程或按键控制F5区的按键使F8区的蜂鸣器发出不同音阶的声音。

表 3.１简易电子琴连线说明

|  |  |
| --- | --- |
| D3 区：CS、AO、Al | A3 区：CS1、AO、A1 |
| D3 区：PC7 | F8 区：Ctrl |
| D3 区：JP23 (PA 口) | F5 区：JP37 (A) |
| Cl 区：GND | F5 区：KL1 |

# 设计实现

## 硬件实现

我们按照原理图与实验连线说明在实验仪上连线，具体如图4.1所示。

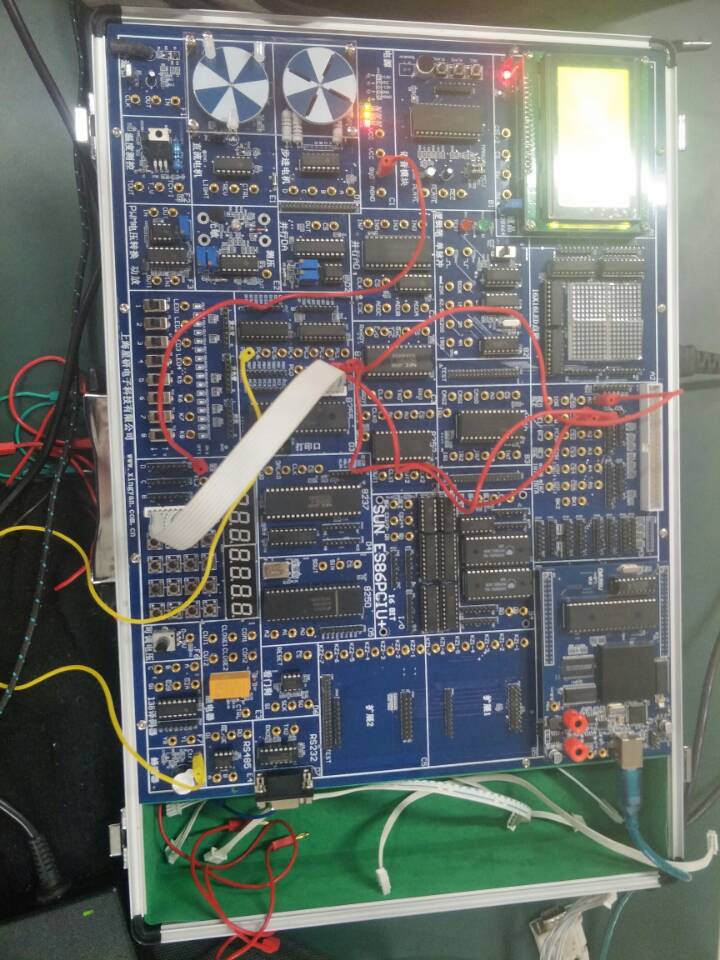


图 4.１实验仪连线

## 软件实现

本次设计的系统流程图如图4.2所示。



图 4.２ 简易电子琴系统执行流程

本次设计所用代码放在附录A中以供查阅，接下来对代码做一些必要的说明。

程序首先定义了最终可执行文件的格式，由于本次实验的代码行数较少、访存空间较小，因此采用了tiny格式。接着设置控制口地址、输入口地址与输出口地址，需要注意的是，不同类型的实验仪的口地址不一定相同，在编程时应予以注意。程序所用的栈空间为100（dw），数据段定义了由各函数组成的《两只老虎》的乐谱。程序的功能部分主要由以下七个部分组成：

1. 调用初始化8255芯片以及播放默认乐曲的函数。显然，为了8255芯片可以正常工作，我们需要根据实验的实际情况，编程决定8255芯片的方式选择控制字与工作方式。本部分主要由两个函数调用组成：

**call** init8255 ;8255 初始化

**call** demo ;播放一段音乐

1. 按键查询有关的一系列操作。我们的设计是先由CPU发送一段默认的乐谱給8255芯片，使之控制蜂鸣器演奏乐曲，再允许操作者做其他演奏。本部分主要由start1~start7等7个分支组成，其中start1完成有无按键按下的查询并检测所按键是否为1，其余6个语句块完成的功能基本相同：查询对应的按键是否按下。
2. 自动播放乐曲的主函数。为了实现自动播放的功能，我们实现了一个函数demo，主要定义了歌曲的节拍数并顺序发出各个音阶。值得一提的是，我们使用寄存器CX来保存节拍数，为了避免在函数调用时其值被覆盖，因此需要提前压入堆栈中保护。
3. 响应按键的一系列函数。这个部分主要由music1~music7等7个函数组成，它们的功能基本相似，区别仅在于为了能够区别不同的音阶而设置的延时，即蜂鸣器在一定时间内震荡的频率。为了保证音阶可以方便地被识别，我们应当取各震荡周期基本一致。
4. 响应自动播放的一系列函数。本部分代码结构与上一部分在形式上具有很大的相似性，主要由m1~m7等7个不同函数组成，所不同的是此部分代码的功能是响应自动播放。不得不说的是，各个函数为了使其对应的音阶更明显，通过寄存器CX设置LOOP循环的次数，即延时时间。另外，上部分与本部分的延时如何调整才能使音阶达到更好的效果需要一定的音乐知识，并加以调试修改。
5. 初始化8255芯片与读写控制有关的函数。这个部分的代码比较简单，由w\_l、w\_h与init8255等三个函数构成，其中w\_l定义向PC口的第7位写0时则蜂鸣器响，w\_h的功能相反，而init8255定义了8255芯片的方式选择控制字与具体采用的芯片口，PC口第7位用于输出，PA口用于输入。
6. 为了满足拍子的要求而构建的一些列延时函数。最后这部分的代码具体实现了各个延时函数，本部分的关键在于函数复用，即使用已定义的函数完成新的功能。

至此，本次实验所用代码的总体架构介绍完毕。

# 设计结果

本实验的演示分为两个部分，首先是对8255芯片编程使之发布不同频率的方波，控制蜂鸣器发出不同的音阶，演奏一首完整的简易乐曲；其次是通过按控制板上的键盘，手动演奏乐曲。

我们经过不断地调试，使电子琴最终完整地演奏了乐曲《两只老虎》，音阶明晰、音色较好，整体仿真效果较好。

# 设计心得与改进方法

在这次课程设计中，我们通过编程控制8255芯片，对电子琴主体部分的电路进行仿真设计，实现了一个具备自动播放与自主演奏功能的简易电子琴。在实验过程中，我们遇到了许多疑难，尤其是代码调试阶段，譬如：未正确设置端口地址导致程序无法载入8255芯片；如何调整曲谱的节拍，使之更加接近标准的演奏效果；如何在不采用外部中断源的情况下，利用软件产生长短适宜的间隔；等等。在经历了一系列的失败与调试后，我们最终达取得了想要的效果。

必须承认的是，我们的实验方法还存在值得改进之处。为了产生更为精确的时间间隔，使乐曲拍子的仿真效果更好，可以考虑使用可编程计数器/定时器8253芯片，通过编程来产生不同频率的方波，将方波输入蜂鸣器来产生不同频率、不同声强的音阶；蜂鸣器同时受8255芯片控制，在控制口允许的情况下才发声。

设计2：LED 16×16点阵

# 设计任务

1. 熟悉8255的功能，了解点阵显示的原理及控制方法；
2. 学会使用LED点阵，通过编程显示不同字符。

# 设计目标

借助可编程并行接口芯片8255，通过PC机编程在SUN ES86PCIU+实验仪的16×16 LED点阵上自下而上循环显示“欢迎使用星研实验仪”字样。本设计的扩展实验为：修改程序与实验线路，在LED点阵上从左至右显示指定的字符串。

# 设计原理

本设计的实验原理图如图3.1所示。

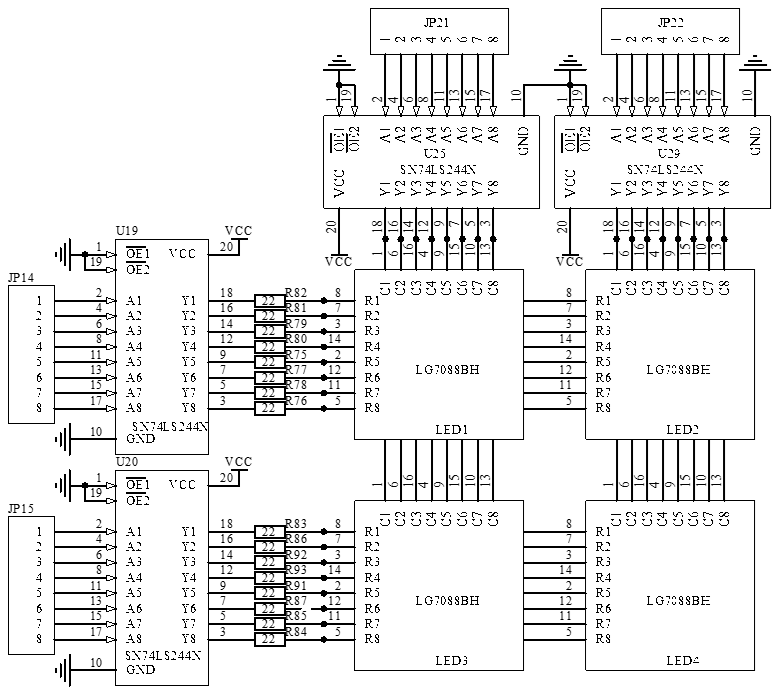


图 3.１LED 16×16点阵实验原理图

下面对图3.1所示的各个区域与接口进行简单介绍。

## A2区

A2区为16×16 LED实验电路。

JP14、JP15组成16根行扫描线；JP21、JP22组成16根列扫描线。

## A3区

A3区是CPU总线与片选区。

### CPU总线

JP28：地址线A0..A7；

JP32：地址线A8..A15；

JP29：地址线A0..A9；

JP33：地址线A8..A17； 根据数据总线宽度，选择合适的地址线。

JP41、JP42：数据总线D0..D7；

JP39、JP40：数据总线D8..D15；

JP47、JP48：数据总线D16..D23；

JP45、JP46：数据总线D24..D31；

控制线：IOR、IOW、MEMR、MEMW、HOLD、HLDA、BLE、BHE、INTR、INTA；

BK1、BK2：备用片选区。

### 片选区

片选情况如表3.1所示。

表 3.１A3区片选情况

|  |  |  |
| --- | --- | --- |
| 片选 | 地址范围 | 说明 |
| mCS0 | 80000H～BFFFFH | 存贮器芯片的片选，16位数据总线 |
| CS1 | 0270H～027FH | I/O芯片的片选，8位数据总线 |
| CS2 | 0260H～026FH |
| CS3 | 0250H～025FH |
| CS4 | 0240H～024FH |
| CS5 | 0230H～023FH | I/O芯片的片选，16位数据总线 |

# 设计实现

## 硬件实现

在实验仪上按照表4.1所示进行连线。

表 4.１LED 16×16点阵主机连线

|  |  |  |
| --- | --- | --- |
| D3区：CS（8255）、A0、A1 | —— | A3区：CS1、A0、A1 |
| D3区：JP23(PA)、JP20(PB) | —— | A2区：JP21、JP22（列输出线） |
| B4区：JP57(D0..D7) | —— | A3区：JP42(D0..D7) |
| B4区：JP56(D8..D15) | —— | A3区：JP40(D8..D15) |
| B4(I/O)区：CS273、BLE、BHE | —— | A3区：CS5、BLE、BHE |
| B4(I/O)区：RD、WR | —— | A3区：IOR、IOW |
| B4(I/O)区：JP51、JP55 | —— | A2区：JP14、JP15（行输出线） |

连线的目的是为了保证控制口命令与地址能够正确地载入8255芯片中。

## 软件实现

本次设计的系统流程图如图4.1所示。



图 4.１ LED 16×16点阵系统执行流程图

本次设计所用代码放在附录B中以供查阅，若需运行出样例结果，只需正确连线并执行程序即可。本节重点是分析如何重构程序，使之可以完成扩展实验，即：在LED点阵上从左至右显示指定的字符串。为了使分析更加自然与透彻，首先介绍与显示有关的器件与基本概念。

本设计所用实验仪的16×16 LED屏幕由二极管组成，当AB端电压差（ROW=1,LINE=0）大于一定值时二极管导通，LED发亮。

所谓的字形码，每两个字节构成一行，输入列控制线ROW1和ROW2，共16行。值得一提是，由于字形码自下而上滚动，需要把最后一个字流动完，所以需要一个32字节的全零数据，此即NONE为全零的作用。

下面开始正式分析整个示例程序。

### 示例代码分析

程序先定义各个IO口和控制口地址。

.MODEL **TINY**

ADDR\_8255\_PA EQU 0270H ;8255 PA口;列线控制端口1

ADDR\_8255\_PB EQU 0271H ;8255 PB口;列线控制端口2

ADDR\_8255\_C EQU 273H ;8255控制口

ADDR\_273 EQU 230H ;8155 PA口;行线控制端口

LINE EQU ADDR\_273 ;行控制线

ROW1 EQU ADDR\_8255\_PA ;列控制线1

ROW2 EQU ADDR\_8255\_PB ;列控制线2

其次，INIT\_IO函数往8255输入控制字，设置8255的PA、PB口为输出口；

再用测试函数TEST\_LED测试LED是否可以全亮；

最后用CLEAR清屏，使LED全灭。

为了使分析不失细节性，接下来介绍程序的一些关键变量。

寄存器CX1：将需要滚动的字符计数。

寄存器SI：将SI原地址指针指向字形码数据首地址。

寄存器CX2：将一个完整字符需要滚动的次数（16屏）计数。

在本设计中，编程控制LED的刷新进而增减点阵的滚动速度。

现在分析已经到了显示一个完整字符的子程序。

寄存器CX3：一屏刷新次数（控制滚动频率）。

寄存器CX4：将显示一屏的全部行数（16）给计数器。

需要注意的是，将IO扩展口输入的最低位输入0、其他位置1通过语句

LINE = FFFEH

实现之。

示例代码分析的最后，介绍实现的几个关键点。

1. 把一行的字形码送入ROW1和ROW2：用LODSB读入当前字节码的一个字节到PA数据口（ROW1），再读入下一个字节到PB数据口（ROW2），若LED的列控制线引脚与PA、PB口的位号相反，则需要调用ADJUST函数将二进制数旋转180°；
2. 将LINE数据循环左移（RCL）。

依次执行1）与2）共16次，每次CX3计数减一，显示一个完整的字形码。

当显示完一屏点阵后SI+2，使得下一次显示的点阵为当前点阵向上滚动一行，并将CX1减一，若CX1=0则已将所有字形码滚动完毕，否则继续循环。当所有字符滚动完毕后用JMP指令跳转到显示第一个字符的代码，循环执行字符滚动。

### 重构代码分析

重构代码完成了设计的扩展，实现了字符串自上而下滚动到自左向右滚动的功能转变，同时修改显示字符串为“我爱♥微机”。

相较4.2.1节分析的示例代码，重构部分主要改变了显示一个完整字符的子程序代码，具体改动为：

1. 将ROW2的最低位输入1，ROW2的其他位以及ROW1所有位清0，通过如下语句实现：

ROW2=01H,ROW1=00H

1. 把一行的字形码送入LINE：用LODSB连续读入当前字节码的两个个字节到LINE，若LED的行控制线引脚与I/O扩展口的位号相反，则需要调用ADJUST函数将二进制数逆置。
2. 将LINE数据循环左移（RCL）。

函数依次执行2）与3）共16次，每次CX3计数减一，显示一个完整的字形码。

# 设计结果

按照图3.1所示原理图与表4.1所示主机接线示意，我们成功运行程序得到了示例结果。经过审慎的分析与大胆的实验，我们完成了本设计的扩展功能。具体结果如图5.1至图5.4所示。



图 5.１示例代码运行结果

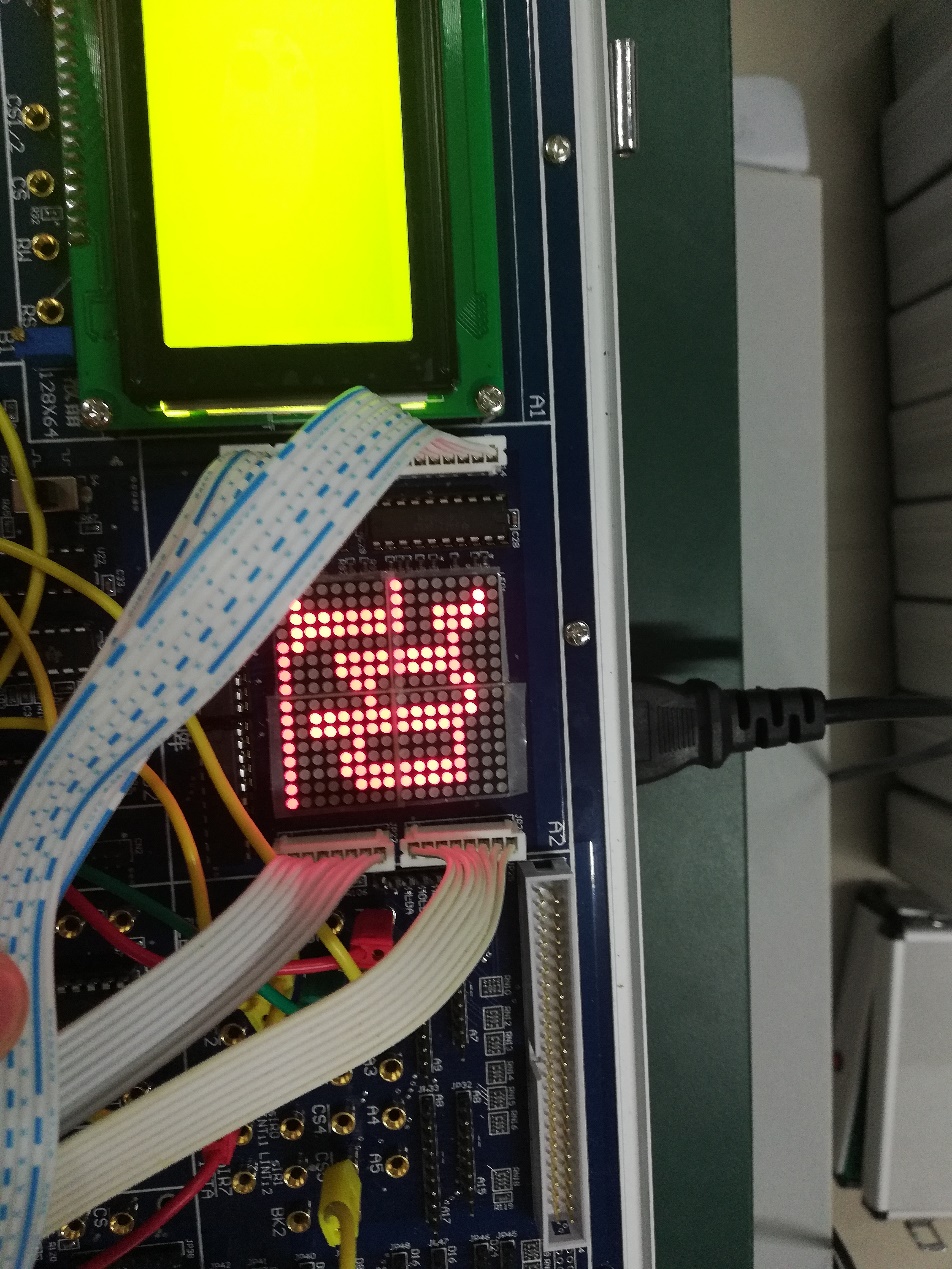


图 5.２示例代码运行结果2

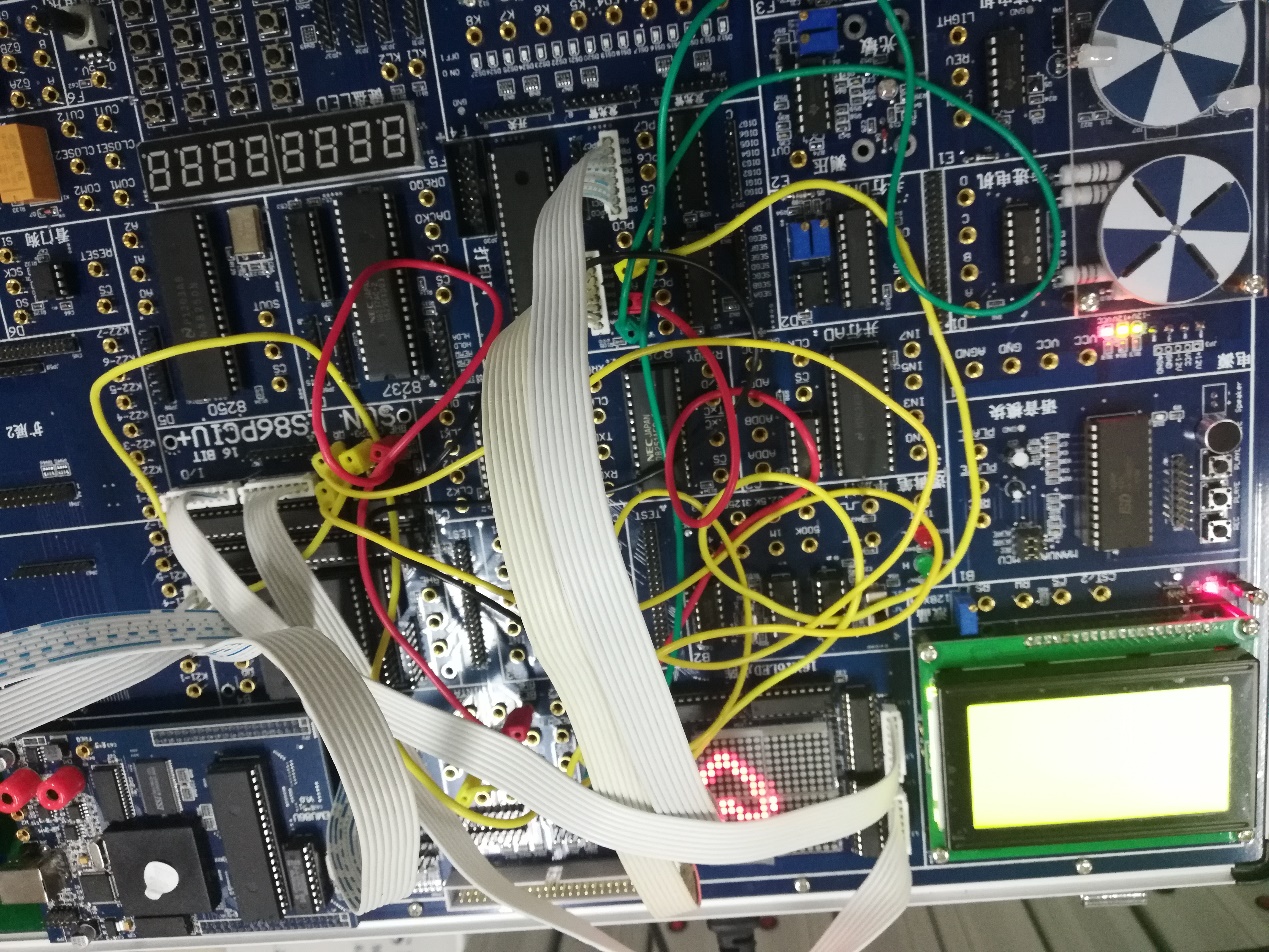


图 5.３ 设计扩展结果1

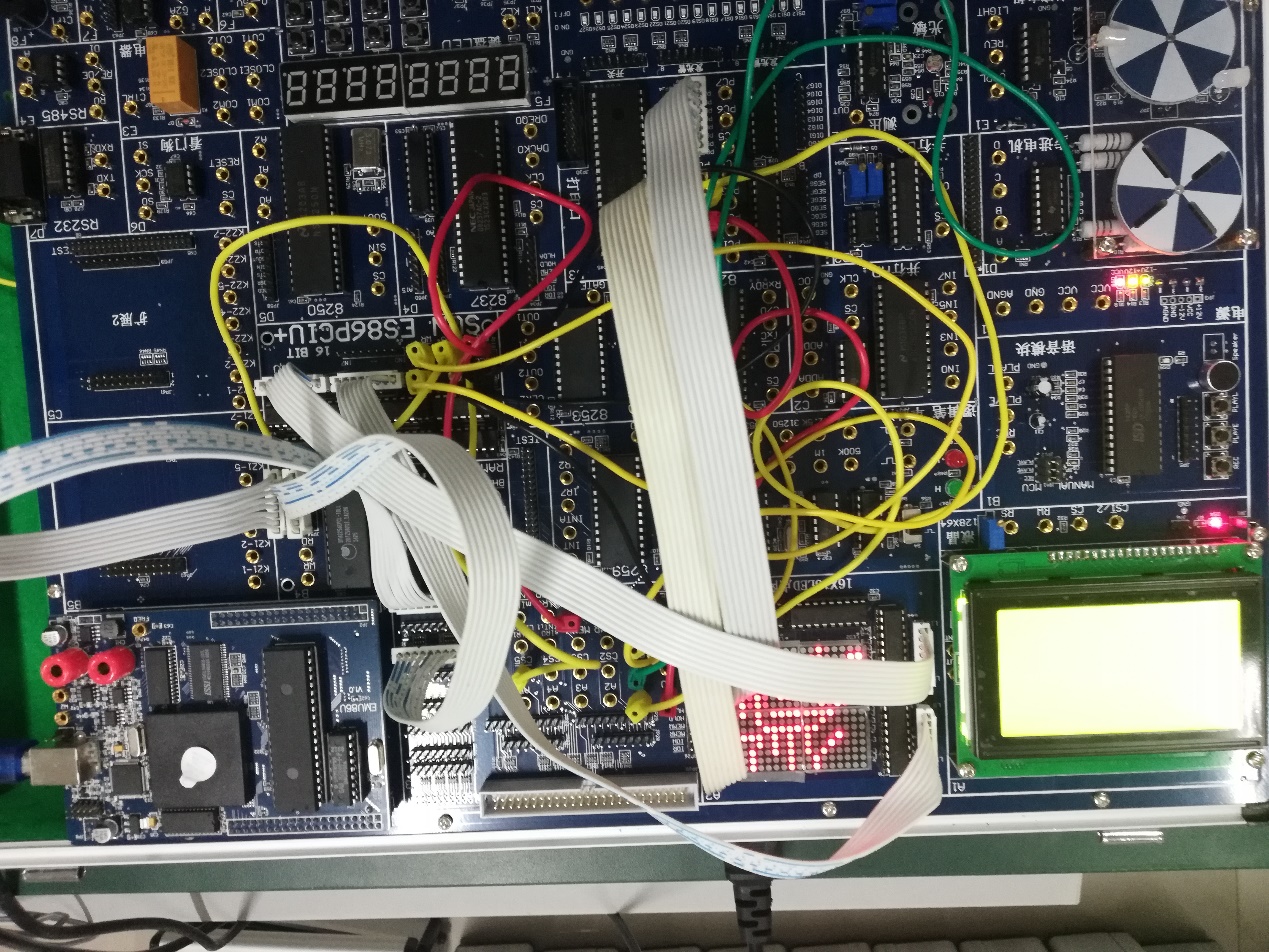


图 5.４设计扩展结果2

# 设计心得与改进方法

在本次课程设计中，我们经过反复的调试与大量的失败，最终成功得到了预期的实验结果。很大程度上本次设计的疑难都是在实现实验扩展时遇到的，由于汇编代码并非十分直观，而我们编写汇编程序的能力又较为稚嫩，调试工作初期进行得异常艰难。总之，得益于我们不畏失败、愈战愈勇的优良品质，以及参考网络上大量的参考资料，我们完满地实现了本次课程设计地既定目标。

本设计地改进方法可以从以下几方面考虑。

1. 精简代码、优化架构。由于我们编写汇编代码能力不够成熟，导致写了许多冗余的代码。
2. 使用可编程计数器/定时器8253芯片。通过编程来控制8253芯片产生不同频率的方波，可以更精确地增减字符串滚动的速度。
3. 高效地生成字形码。本设计所有的待显示字符串都是通过使用常量来固定字形码，进而在程序中调用。可以改进算法，使得用户只需输入给定的字符串，由程序自动将其转换为字形码，增强用户的使用体验。

附录A

简易电子琴实现代码：

.model **tiny**

c8255 equ 0273h

pa8255 equ 0270h

pc8255 equ 0272h

.stack 100

.data

music dw m1**,**q3**,**m2**,**q3**,**m3**,**q3**,**m1**,**q3**,**m1**,**q3**,**m2**,**q3**,**m3**,**q3**,**m1**,**q3

dw m3**,**q2**,**m4**,**q2**,**m5**,**q3**,**m3**,**q2**,**m4**,**q2**,**m5**,**q3

dw m5**,**q2**,**m6**,**q2**,**m5**,**q2**,**m4**,**q2**,**m3**,**q2**,**m1**,**q2**,**m5**,**q2**,**m6**,**q2**,**m5**,**q2

dw m4**,**q2**,**m3**,**q2**,**m1**,**q2**,**m1**,**q2**,**m5**,**q2**,**m1**,**q2**,**m1**,**q2**,**m5**,**q2**,**m1

.code

start**:**

**call** init8255 ;8255 初始化

**call** demo ;播放一段音乐

start1**:**

**mov** **dx,** pa8255 ;按键查询

**in** **al,** **dx** ;读键值

**cmp** **al,** 0ffh ;judge if al equals 11111111b

**jz** start1 ;无键

**xor** **al,** 0ffh ;有键

**test** **al,** 1

**jz** start2

**call** music1 ;1 号键 , 调 1 号键输出

**jmp** start1

start2**:**

**test** **al,** 2 ;2h = 00000010b

**jz** start3

**call** music2 ;2 号键

**jmp** start1

start3**:**

**test** **al,** 4 ;4h = 00000100b

**jz** start4

**call** music3 ;3 号键

**jmp** start1

start4**:**

**test** **al,** 8 ;8h = 00001000b

**jz** start5

**call** music4 ;4 号键

**jmp** start1

start5**:**

**test** **al,** 10h ;10h = 00010000b

**jz** start6

**call** music5 ;5 号键

**jmp** start1

start6**:**

**test** **al,** 20h ;20h = 00100000b

**jz** start7

**call** music6 ;6 号键

**jmp** start1

start7**:**

**test** **al,** 40h ;40h = 01000000b

**jz** start1

**call** music7 ;7 号键

**jmp** start1

demo proc **near**

**mov** **cx,** 80 ;生日歌共 25 拍

**lea** **bx,** music

demo10**:**

**push** **cx** ;保护 cx 的值

**call** **[bx]** ;播放 bx 指向的音阶

;call [bx+50]

;call [si]

;call [bx]

**inc** **bx**

**inc** **bx** ;bx <- bx + 2，指向预定的下一个音阶

;inc si

;inc si

**pop** **cx** ;恢复 cx 的值

**loop** demo10 ;继续播放

**ret**

demo endp

music1 proc **near** ;节拍 1(手动按键时用 )

**call** w\_l ;写 0, 蜂鸣器响

**call** t10 ;延时 100us

**call** t5 ;延时 50us

**call** t5 ;延时 50us

**call** w\_h ;写 1, 蜂鸣器不响

**call** t10 ;延时

**call** t5 ;

**call** t5 ;

**ret**

music1 endp

music2 proc **near** ;节拍 2, 同上

**call** w\_l ;写 0, 蜂鸣器响

**call** t10

**call** t5

**call** t2

**call** t1

**call** w\_h ;写 1, 蜂鸣器不响

**call** t10

**call** t5

**call** t2

**call** t2

**ret**

music2 endp

music3 proc **near** ;节拍 3, 同上

**call** w\_l

**call** t10

**call** t5

**call** t2

**call** t1

**call** w\_h

**call** t10

**call** t5

**call** t2

**ret**

music3 endp

music4 proc **near** ;节拍 4, 同上

**call** w\_l

**call** t10

**call** t5

**call** t1

**call** w\_h

**call** t10

**call** t5

**call** t1

**ret**

music4 endp

music5 proc **near** ;节拍 5, 同上

**call** w\_l

**call** t10

**call** t5

**call** w\_h

**call** t10

**call** t5

**ret**

music5 endp

music6 proc **near** ;节拍 6, 同上

**call** w\_l

**call** t10

**call** t2

**call** t2

**call** w\_h

**call** t10

**call** t2

**call** t2

**ret**

music6 endp

music7 proc **near** ;节拍 7, 同上

**call** w\_l

**call** t10

**call** t2

**call** t1

**call** w\_h

**call** t10

**call** t2

**call** t1

**ret**

music7 endp

;m1~m7每个节拍放音的时间基本相同，约为0.2s，通过修改函数体所call的函数体来改变每秒蜂鸣器震

; 荡的次数，以此来体现放音时不同的音阶。

m1 proc **near** ;节拍 1(自动放音时用 , 时间约 0.2s)

**mov** **cx,** 1100 ;cx 控制的是下面两条loop指令执行的次

;数，1100 共执行loop指令 550 次

m10**:**

**call** w\_l

**call** t10

**call** t10

**call** t2

**loop** m11

m11**:**

**call** w\_h

**call** t10

**call** t10

**call** t1

**loop** m10

**ret**

m1 endp

m2 proc **near** ;节拍 2, 同上

**mov** **cx,** 1150

m20**:**

**call** w\_l

**call** t10

**call** t5

**call** t2

**call** t2

**loop** m21

m21**:**

**call** w\_h

**call** t10

**call** t5

**call** t2

**call** t2

**loop** m20

**ret**

m2 endp

;节拍 3, 同上

m3 proc **near**

**mov** **cx,** 1200

m30**:**

**call** w\_l

**call** t10

**call** t5

**call** t2

**call** t1

**push** **ax**

**pop** **ax**

**nop**

**nop**

**loop** m31

m31**:**

**call** w\_h

**call** t10

**call** t5

**call** t2

**call** t1

**loop** m30

**ret**

m3 endp

;节拍 4, 同上

m4 proc **near**

**mov** **cx,** 1250

m40**:**

**call** w\_l

**call** t10

**call** t5

**call** t2

**call** t1

**loop** m41

m41**:**

**call** w\_h

**call** t10

**call** t5

**call** t2

**call** t1

**loop** m40

**ret**

m4 endp

;节拍 5, 同上

m5 proc **near**

**mov** **cx,** 1300

m50**:**

**call** w\_l

**call** t10

**call** t5

**call** t2

**loop** m51

m51**:**

**call** w\_h

**call** t10

**call** t5

**call** t2

**loop** m50

**ret**

m5 endp

;节拍 6, 同上

m6 proc **near**

**mov** **cx,** 1350

m60**:**

**call** w\_l

**call** t10

**call** t5

**call** t1

**loop** m61

m61**:**

**call** w\_h

**call** t10

**call** t5

**call** t1

**loop** m60

**ret**

m6 endp

;节拍 7, 同上

m7 proc **near**

**mov** **cx,** 1420

m70**:**

**call** w\_l

**call** t10

**call** t5

**loop** m71

m71**:**

**call** w\_h

**call** t10

**call** t5

**loop** m70

**ret**

m7 endp

;写 0(8255.pc.7=0)，蜂鸣器响

w\_l proc **near**

**mov** **dx,** c8255

**mov** **al,** 0eh

**out** **dx,** **al**

**ret**

w\_l endp

;写 1(8255.pc.7=1)

w\_h proc **near**

**mov** **dx,** c8255

**mov** **al,** 0fh

**out** **dx,** **al**

**ret**

w\_h endp

;8255 初始化

init8255 proc **near**

**mov** **dx,** c8255

**mov** **al,** 90h ;pc.7输出 , pa输入

**out** **dx,** **al**

**mov** **dx,** c8255

**mov** **al,** 0fh

**out** **dx,** **al**

**ret**

init8255 endp

;延时 10us

t1 proc **near**

**ret**

t1 endp

;延时 20us

t2 proc **near**

**call** t1

**ret**

t2 endp

;延时 50us

t5 proc **near**

**call** t2

**call** t2

**ret**

t5 endp

;延时 100us

t10 proc **near**

**call** t2

**call** t2

**call** t5

**ret**

t10 endp

q1 proc **near**

**mov** **cx,**300

loopq1**:** **call** t10

**loop** loopq1

**ret**

q1 endp

q2 proc **near**

**call** q1

**call** q1

**ret**

q2 endp

q3 proc **near**

**call** q2

**call** q2

**ret**

q3 endp

end start

附录B

LED 16×16点阵（自下而上滚动）：

.model **tiny**

addr\_8255\_pa equ 0270h

addr\_8255\_pb equ 0271h

addr\_8255\_c equ 273h

addr\_273 equ 230h

line equ addr\_273

row1 equ addr\_8255\_pa

row2 equ addr\_8255\_pb

.stack 100

.data

huan db 00h**,**0c0h**,**00h**,**0c0h**,**0feh**,**0c0h**,**07h**,**0ffh**,**0c7h**,**86h**,**6fh**,**6ch**,**3ch**,**60h**,**18h**,**60h

db 1ch**,**60h**,**1ch**,**70h**,**36h**,**0f0h**,**36h**,**0d8h**,**61h**,**9ch**,**0c7h**,**0fh**,**3ch**,**06h**,**00h**,**00h

ying db 60h**,**00h**,**31h**,**0c0h**,**3fh**,**7eh**,**36h**,**66h**,**06h**,**66h**,**06h**,**66h**,**0f6h**,**66h**,**36h**,**66h

db 37h**,**0e6h**,**37h**,**7eh**,**36h**,**6ch**,**30h**,**60h**,**30h**,**60h**,**78h**,**00h**,**0cfh**,**0ffh**,**00h**,**00h

shi db 00h**,**00h**,**06h**,**30h**,**07h**,**30h**,**0fh**,**0ffh**,**0ch**,**30h**,**1fh**,**0ffh**,**3bh**,**33h**,**7bh**,**33h

db 1bh**,**0ffh**,**1bh**,**33h**,**19h**,**0b0h**,**18h**,**0e0h**,**18h**,**60h**,**18h**,**0fch**,**19h**,**8fh**,**1fh**,**03h

yong db 00**,**0**,**1fh**,**0feh**,**18h**,**0c6h**,**18h**,**0c6h**,**18h**,**0c6h**,**1fh**,**0feh**,**018h**,**0c6h**,**18h**,**0c6h

db 18h**,**0c6h**,**1fh**,**0feh**,**18h**,**0c6h**,**18h**,**0c6h**,**30h**,**0c6h**,**30h**,**0c6h**,**60h**,**0deh**,**0c0h**,**0cch

xing db 00h**,**00h**,**1fh**,**0fch**,**18h**,**0ch**,**1fh**,**0fch**,**18h**,**0ch**,**1fh**,**0fch**,**01h**,**80h**,**19h**,**80h

db 1fh**,**0feh**,**31h**,**80h**,**31h**,**80h**,**6fh**,**0fch**,**01h**,**80h**,**01h**,**80h**,**7fh**,**0ffh**,**00h**,**00h

yan db 0**,**0**,**0ffh**,**0ffh**,**18h**,**0cch**,**18h**,**0cch**,**30h**,**0cch**,**30h**,**0cch**,**7fh**,**0ffh**,**7ch**,**0cch

db 0fch**,**0cch**,**3ch**,**0cch**,**3ch**,**0cch**,**3dh**,**8ch**,**3dh**,**8ch**,**33h**,**0ch**,**06h**,**0ch**,**0ch**,**0ch

shi0 db 01h**,**80h**,**00h**,**0c0h**,**3fh**,**0ffh**,**3ch**,**06h**,**67h**,**0cch**,**06h**,**0c0h**,**0ch**,**0c0h**,**07h**,**0c0h

db 06h**,**0c0h**,**7fh**,**0ffh**,**00h**,**0c0h**,**01h**,**0e0h**,**03h**,**30h**,**06h**,**18h**,**1ch**,**1ch**,**70h**,**18h

yan0 db 00h**,**00h**,**0fch**,**60h**,**0ch**,**60h**,**6ch**,**0f0h**,**6ch**,**0d8h**,**6dh**,**8fh**,**6fh**,**0f8h**,**7eh**,**00h

db 06h**,**0c6h**,**07h**,**66h**,**3fh**,**0ech**,**0e7h**,**0ech**,**06h**,**18h**,**1fh**,**0ffh**,**0ch**,**00h**,**00h**,**00h

yi db 0ch**,**0c0h**,**0ch**,**60h**,**18h**,**7ch**,**1bh**,**6ch**,**33h**,**0ch**,**73h**,**18h**,**0f1h**,**98h**,**31h**,**98h

db 30h**,**0f0h**,**30h**,**0f0h**,**30h**,**60h**,**30h**,**0f0h**,**31h**,**98h**,**33h**,**0fh**,**3eh**,**06h**,**30h**,**00h

**none** db 00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h

db 00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h

.code

start**:** **mov** **ax,**@data

**mov** **ds,ax**

**mov** **es,ax**

**nop**

**call** init\_io

**call** test\_led

**call** clear

chs\_show**:** **mov** **cx,**9

**lea** **si,**huan

chs\_1**:** **push** **cx**

**mov** **cx,**16

chs\_2**:** **call** disp\_ch

**inc** **si**

**inc** **si**

**loop** chs\_2

**pop** **cx**

**loop** chs\_1

**jmp** chs\_show

disp\_ch proc **near**

**push** **cx**

**mov** **cx,**8

disp\_ch\_1**:** **call** disp1

**loop** disp\_ch\_1

**pop** **cx**

**ret**

disp\_ch endp

disp1 proc **near**

**push** **si**

**push** **cx**

**mov** **cx,**16

**mov** **bl,**0feh

**mov** **bh,**0ffh

repeat**:** **mov** **dx,**line

**mov** **ax,bx**

**out** **dx,ax**

**lodsb**

**mov** **dx,**row1

**out** **dx,al**

**lodsb**

**mov** **dx,**row2

**out** **dx,al**

**call** dl10ms

**call** clear

**stc**

**rcl** **bl,**1

**rcl** **bh,**1

**loop** repeat

**pop** **cx**

**pop** **si**

**ret**

disp1 endp

init\_io proc **near**

**mov** **dx,**addr\_8255\_c

**mov** **al,**80h

**out** **dx,al**

**ret**

init\_io endp

clear proc **near**

**mov** **al,**0ffffh

**mov** **dx,**line

**out** **dx,ax**

**mov** **al,**0

**mov** **dx,**row1

**out** **dx,al**

**mov** **dx,**row2

**out** **dx,al**

**ret**

clear endp

test\_led proc **near**

**mov** **dx,**line

**xor** **ax,ax**

**out** **dx,ax**

**mov** **al,**0ffh

**mov** **dx,**row1

**out** **dx,al**

**mov** **dx,**row2

**out** **dx,al**

**call** dl500ms

**call** dl500ms

**ret**

test\_led endp

dl10ms proc **near**

**push** **cx**

**mov** **cx,**133

**loop** **$**

**pop** **cx**

**ret**

dl10ms endp

dl500ms proc **near**

**push** **cx**

**mov** **cx,**0ffffh

**loop** **$**

**pop** **cx**

**ret**

dl500ms endp

end start

LED 16×16点阵（自左向右滚动）：

.model **tiny**

addr\_8255\_pa equ 0270h

addr\_8255\_pb equ 0271h

addr\_8255\_c equ 273h

addr\_273 equ 230h

line equ addr\_273

row1 equ addr\_8255\_pa

row2 equ addr\_8255\_pb

.stack 100

.data

wo db 00h**,**00h**,**04h**,**00h**,**0ch**,**07h**,**04h**,**01h**,**35h**,**86h**,**44h**,**58h**,**04h**,**60h**,**0ffh**,**90h**,**04h**,**08h**,**44h**,**84h**,**0c4h**,**40h**,**7fh**,**0feh**,**24h**,**21h**,**24h**,**12h**,**24h**,**10h**,**04h**,**00h

ai db 00h**,**00h**,**0ch**,**00h**,**0ah**,**02h**,**89h**,**03h**,**0a9h**,**42h**,**99h**,**64h**,**89h**,**54h**,**49h**,**48h**,**79h**,**54h**,**4fh**,**62h**,**49h**,**0c1h**,**59h**,**31h**,**69h**,**0ch**,**49h**,**02h**,**0dh**,**00h**,**02h**,**00h

wei db 00h**,**00h**,**08h**,**02h**,**1eh**,**03h**,**09h**,**0cch**,**0e8h**,**30h**,**1bh**,**0cch**,**04h**,**12h**,**3dh**,**09h**,**05h**,**7ch**,**0fdh**,**40h**,**05h**,**40h**,**3dh**,**7ch**,**0c5h**,**02h**,**23h**,**0ffh**,**11h**,**00h**,**08h**,**80h

ji db 00h**,**00h**,**00h**,**0eh**,**00h**,**02h**,**20h**,**02h**,**7fh**,**0fch**,**20h**,**00h**,**20h**,**00h**,**20h**,**00h**,**3fh**,**0f8h**,**00h**,**04h**,**08h**,**82h**,**09h**,**01h**,**0ffh**,**0ffh**,**0bh**,**00h**,**08h**,**0c0h**,**08h**,**20h

xin db 00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**03h**,**00h**,**07h**,**0c0h**,**0ch**,**0e0h**,**0ch**,**30h**,**06h**,**18h**,**03h**,**0ch**,**03h**,**0ch**,**06h**,**18h**,**0ch**,**30h**,**0ch**,**0e0h**,**07h**,**0c0h**,**03h**,**00h**,**00h**,**00h

**none** db 00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h

db 00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h**,**00h

.code

start**:** **mov** **ax,**@data

**mov** **ds,ax**

**mov** **es,ax**

**nop**

**call** init\_io

**call** test\_led

**call** clear

chs\_show**:**

**mov** **cx,**5

**lea** **si,**wo

chs\_1**:** **push** **cx**

**mov** **cx,**16

chs\_2**:** **call** disp\_ch

**inc** **si**

**inc** **si**

**loop** chs\_2

**pop** **cx**

**loop** chs\_1

**jmp** chs\_show

disp\_ch proc **near**

**push** **cx**

**mov** **cx,**20

disp\_ch\_1**:**

**call** disp1

**loop** disp\_ch\_1

**pop** **cx**

**ret**

disp\_ch endp

disp1 proc **near**

**push** **si**

**push** **cx**

**mov** **cx,**16

**mov** **bl,**01h

**mov** **bh,**00h

repeat**:** **mov** **dx,**row2

**mov** **al,bl**

**out** **dx,al**

**mov** **dx,**row1

**mov** **al,bh**

**out** **dx,al**

**lodsb**

**call** adjust

**mov** **dl,al**

**lodsb**

**call** adjust

**mov** **ah,dl**

**xchg** **al,ah**

**xor** **ax,**0ffffh

**mov** **dx,**line

**out** **dx,ax**

**call** dl10ms

**call** clear

**clc**

**rcl** **bl,**1

**rcl** **bh,**1

**loop** repeat

**pop** **cx**

**pop** **si**

**ret**

disp1 endp

init\_io proc **near**

**mov** **dx,**addr\_8255\_c

**mov** **al,**80h

**out** **dx,al** ;

**ret**

init\_io endp

clear proc **near**

**mov** **al,**0ffffh

**mov** **dx,**line

**out** **dx,ax**

**mov** **al,**0

**mov** **dx,**row1

**out** **dx,al**

**mov** **dx,**row2

**out** **dx,al**

**ret**

clear endp

test\_led proc **near**

**mov** **dx,**line

**xor** **ax,ax**

**out** **dx,ax**

**mov** **al,**0ffh

**mov** **dx,**row1

**out** **dx,al**

**mov** **dx,**row2

**out** **dx,al**

**call** dl500ms

**call** dl500ms

**ret**

test\_led endp

adjust proc **near**

**push** **cx**

**mov** **cx,**8

adjust1**:** **rcl** **al,**1

**xchg** **al,ah**

**rcr** **al,**1

**xchg** **al,ah**

**loop** adjust1

**mov** **al,ah**

**pop** **cx**

**ret**

adjust endp

dl10ms proc **near**

**push** **cx**

**mov** **cx,**133

**loop** **$**

**pop** **cx**

**ret**

dl10ms endp

dl500ms proc **near**

**push** **cx**

**mov** **cx,**0ffffh

**loop** **$**

**pop** **cx**

**ret**

dl500ms endp

end start