

北京科技大学实验报告

学院：计通学院

专业：计算机科学与技术

班级：计 173

姓名：张宝丰

学号：41724081

实验日期：2019 年 12 月 12 日

实验名称：实验一 8259 中断控制器应用实验

实验目的：1) 掌握 PC 机中断处理系统的基本原理。

2) 掌握可编程中断控制器 8259 的应用编程方法。

实验要求：按要求完成实验内容，并撰写实验报告。

实验环境：软件环境：Win7 32 位操作系统

编程语言：汇编语言

硬件环境：CZ-CIUS 微机接口实验系统

实验内容：

实验 1-1：PC 机内中断嵌套实验

1) 按接线图连好接线，调用程序源代码 8259-2.asm，做如下操作，并将屏幕显示结果以截图的方式写在实验报告中，并分析产生该现象的原因：

按下连接 IRQ 的单次脉冲按键，屏幕上会显示 10 个 N，在屏幕上 10 次显示未结束之前，按下连接 IRQ10 的单次脉冲按键，观察现象；

按下连接 IRQ10 的单次脉冲按键，屏幕上会显示 10 个 Y，在屏幕上 10 次显示未结束之前，按下连接 IRQ 的单次脉冲按键，观察现象。

2) 请用所给出的实验代码，详细解释中断嵌套是如何实现的，哪些代码起到了哪些功能。（解释过程中需引用源代码）

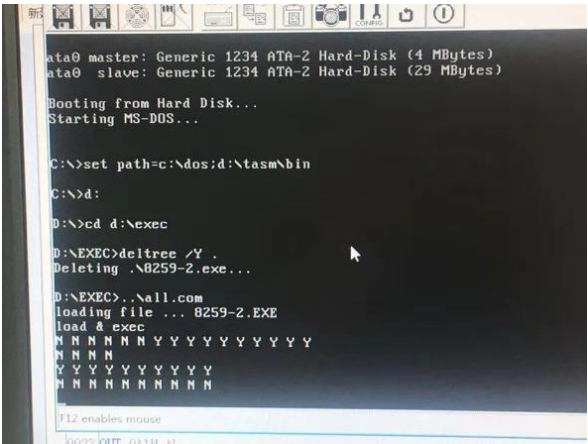
实验 1-2：PC 机内中断实验

本实验采用查询方式，应用实验箱提供的 8259 中断源，向 PC 机提交中断请求。拨动开关发起请求一次中断，屏幕上显示相应的中断请求号。调用程序源码文件 8259-3.asm，在程序源代码中划横线的位置，请按照所学 8259 工作原理填写并验证，然后将所填内容写在实验报告中，并分析所填数据的形成原理。

实验结果与分析：

实验 1-1：PC 机内中断嵌套实验

1) 实验结果



```
ata0 master: Generic 1234 ATA-2 Hard-Disk (4 MBytes)
ata0 slave: Generic 1234 ATA-2 Hard-Disk (29 MBytes)

Booting from Hard Disk...
Starting MS-DOS...

C:\>set path=c:\dos;d:\tasm\bin
C:\>d:
D:\>cd d:\exec
D:\EXEC>deltree /Y .
Deleting .\8259-2.exe...

D:\EXEC>.. \all.com
loading file ... 8259-2.EXE
load & exec
N N N N N Y Y Y Y Y Y Y Y Y Y
N N N N
Y Y Y Y Y Y Y Y Y Y
N N N N N N N N N N

FL/ enables mouse
```

2) 原因分析

触发 IRQ10 中断会输出 Y，触发 IRQ3 会输出 N。从上图的执行结果中，可以看到，IRQ3 会被 IRQ10 所打断，而反之则不然。这说明 IRQ10 的优先级比 IRQ3 的优先级高，其硬件原理如下：

默认情况下，单片 8259A 芯片的优先级从高到低依次为 IRQ0>IRQ1……>IRQ7。依据实验手册上的级联情况解释，从片的中断信号应该是连接到了主片 IRQ2 上，因此 IRQ10 的优先级高于 IRQ3，这就解释了上图中的两种不同现象。

3) 什么是中断嵌套

依照官方的解释，中断嵌套是指中断系统正在执行一个中断服务时，有另一个优先级更高的中断提出中断请求，这时会暂时终止当前正在执行的级别较低的中断源的服务程序，去处理级别更高的中断源，待处理完毕，再返回到被中断了的中断服务程序继续执行，整个过程称作中断嵌套。

在本次实验中，我们通过 25H 中断，为 IRQ3 和 IRQ10 分别设置了中断向量，之后分别读取两个 8259A 芯片的 IMR 并开启对应位的中断。

IN AL,21H	; 允许 3 号中断
AND AL,0F3H	; AL&=11110011 打开主片的 IRQ2 和 IRQ3
OUT 21H,AL	
IN AL,0A1H	; 允许 10 号中断
AND AL,0FBH	; AL&=11111011 打开从片的 IRQ2(即 IRQ10)
OUT 0A1H,AL	

此外，阅读示例代码时我发现，IRQ3 和 IRQ10 的中断服务程序都只是在开始有 CLI（关中断），在最后有 STI（开中断），中断服务程序运行时不相应外部中断，但是却会响应内中断。

实验 1-2：PC 机内中断实验

1) 实验结果

```
; 这里对端口地址的设置要求:
; ICW1 写入偶地址端口, ICW2-4 写入奇地址端口
; OCW2 和 OCW3 写入偶地址端口, OCW1 写入奇地址端口

I8259_1    EQU    2B4H        ; 8259 的 ICW1 端口地址
I8259_2    EQU    2B3H        ; 8259 的 ICW2 端口地址
I8259_3    EQU    2B6H        ; 8259 的 ICW3 端口地址
I8259_4    EQU    2B5H        ; 8259 的 ICW4 端口地址
O8259_1    EQU    2B1H        ; 8259 的 OCW1 端口地址
O8259_2    EQU    2B0H        ; 8259 的 OCW2 端口地址
O8259_3    EQU    2B2H        ; 8259 的 OCW3 端口地址

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACKS, ES:DATA
.386

    MOV DX, I8259_1            ;初始化 8259 的 ICW1
    MOV AL, 00010011b          ;边沿触发、单片 8259、需要 ICW4
    OUT DX,AL

    MOV DX,I8259_2            ;初始化 8259 的 ICW2
    MOV AL,0B0H
    OUT DX,AL
    MOV AL,03H
    OUT DX,AL

    MOV DX, O8259_1            ;初始化 8259 的中断屏蔽字 OCW1
    MOV AL, 00H                ;打开屏蔽位(填空)
    OUT DX,AL

QUERY:  MOV AH,1                ;判断是否有按键按下
        INT 16H
        JNZ QUIT                ;有按键则退出
        MOV DX, O8259_3          ;向 8259 发送查询命令(填空)
        MOV AL, 0CH              ;设置查询方式(填空)
```

```

        OUT DX,AL
        .....

        MOV DX,08259_2      ;向 8259 发送中断结束命令（填空）
        MOV AL,20H          ;普通的 EOI 命令（填空）
        OUT DX,AL           ;填空
        JMP QUERY
QUIT:    MOV AX,4C00H        ;结束程序退出
        INT 21H

CODE ENDS
END START

```

2) 运行结果的屏幕截图

The screenshot shows a Bochs for Windows - Display window. The command prompt shows the following sequence of commands and output:

```

C:\>set path=c:\dos;d:\tasm\bin
C:\>d:
D:\>cd d:\exec
D:\EXEC>deltree /Y .
Deleting .\8259-3.exe...
D:\EXEC>..\all.com
loading file ... 8259-3.EXE
load & exec
HELLO! THIS IS COMPUTER LAB      * 6 *!
HELLO! THIS IS COMPUTER LAB      * 6 *!
HELLO! THIS IS COMPUTER LAB      * 4 *!
HELLO! THIS IS COMPUTER LAB      * 4 *!
HELLO! THIS IS COMPUTER LAB      * 4 *!

```

At the bottom of the window, it says "F12 enables mouse".

实验结论（讨论）:

外接的 8259A 芯片在使用前需要进行初始化，ICW1-4 是初始化字，需要在初始化时设置；OCW1-3 是控制状态字，可以在程序运行过程中根据需要调整。ICW 和 OCW 会分别对应一共 7 个端口，根据要求查阅芯片手册，依次设置 ICW 和 OCW 即可。

在日常使用 PC 时，经常遇到的中断请求来源包括磁盘、内存、键盘、显示器、打印机等，这些都称作外中断，是外设产生的中断；此外还有内中断，由软件本身产生，比如操作系统利用时间片轮转算法调度进程，当为某个进程分配的时间片用尽时，操作系统就会产生内中断，用来调度程序；还有子程序返回时调用 `ret` 或 `iret` 也会产生软中断。

实验名称：实验二 8254 定时/计数器应用实验

实验目的：1) 掌握 8254 的工作方式及应用编程。

2) 掌握 8254 典型应用电路的接法。

实验要求：按要求完成实验内容，并撰写实验报告。

实验环境：

软件环境：Win7 32 位操作系统

编程语言：汇编语言

硬件环境：CZ-CIUS 微机接口实验系统

实验内容：

实验 3-1：计数应用实验

应用 8254 的计数功能，用开关模拟计数，使每当按照计数初值的次数按动单次脉冲后，观察 LED 的变化。

将计数器 1 设置为方式 0，计数器初值为 7，用手动逐个输入单脉冲，用 LED 灯观察 OUT0 电平变化。

- 1) 将代码中划横线的部分填上相应的代码，并在实验报告中对所填代码做原理分析。
- 2) 如果令计数器 1 工作在方式 3，其实验现象有什么不同，在报告中详细分析说明。

实验 3-2：自设计实验

参考实验一的程序和接线，自行设计接线图以及程序代码，实现，以 2MHz 为时钟源，应用 8254 的定时功能，将其分频为 1Hz。以 LED 灯作为输出显示。经过实际验证后，在报告中画出接线图，并给出源代码。

实验结果与分析：

实验 3-1：计数应用实验

1) 实验结果

```
I08254_MODE      EQU    283H      ;8254 控制寄存器端口地址
I08254_COUNT1     EQU    281H      ;8254 计数器 1 端口地址

STACK1 SEGMENT STACK
            DW 256 DUP(?)
STACK1 ENDS
```

```

CODE SEGMENT
    ASSUME CS:CODE
START: MOV DX, I08254_MODE      ;初始化 8254 工作方式
      MOV AL, 01010000b        ;计数器 1, 方式 0 (填空 1)
      ;MOV AL, 01010110b      ;计数器 1, 方式 3 (填空 1)
      OUT DX, AL

      MOV DX, 281H             ;装入计数初值 (填空 2)
      MOV AL, 7                ; (填空 3)
      OUT DX,AL

      MOV AX,4C00H             ;返回到 DOS
      INT 21H

CODE ENDS
      END START

```

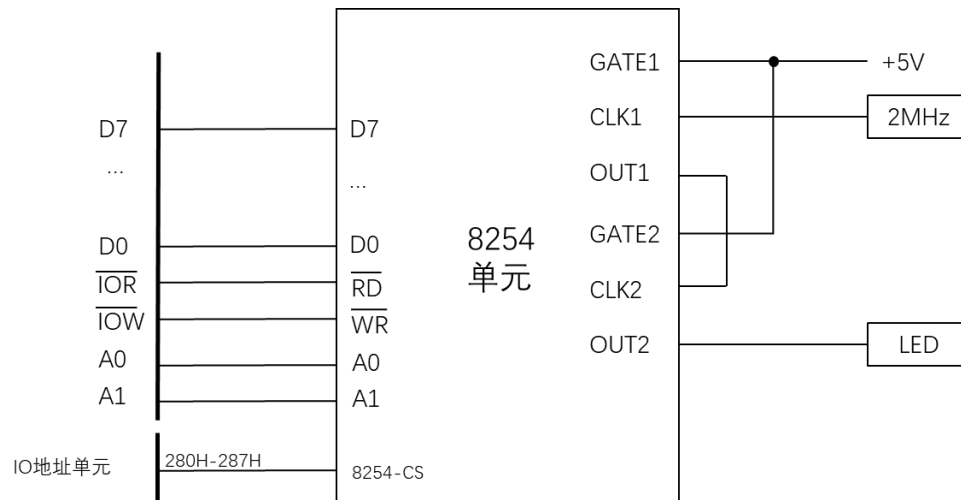
2) 如果令计数器 1 工作在方式 3, 其实验现象与计数器工作在方式 0 有什么不同, 在报告中进行分析说明。

- 方式 0 现象: 初始时 LED 灭, 按 7 下单次脉冲键, LED 亮。
- 方式 3 现象: 初始时 LED 亮, 按 4 下 LED 灭, 再按 3 下 LED 亮, 如此往复。

方式 0 计数到 0 结束输出正跃变信号, 故而按 7 下按键后, 输出信号由低电平跃至高电平, LED 亮起; 方式 3 是方波发生器, 设计数初值为 N , 初始时输出为高电平, $(N+1)/2$ 个时钟周期后, 输出变为低电平; 再过 $N/2$ 个时钟周期后, 输出再变回高电平, 如此循环。

实验 3-2：自设计实验

1) 实验接线图



2) 实验源码

```

I08254_MODE    EQU    283H        ;8254 控制寄存器端口地址
I08254_COUNT1  EQU    281H        ;8254 计数器 1 端口地址
I08254_COUNT2  EQU    282H        ;8254 计数器 2 端口地址

STACK1 SEGMENT STACK
    DW    256 DUP(?)
STACK1 ENDS

CODE SEGMENT
    ASSUME CS:CODE
START: MOV     DX, I08254_MODE      ;初始化 8254 工作方式
        MOV     AL, 01110110B      ;计数器 1, 方式 3
        OUT     DX, AL

        MOV     DX, I08254_COUNT1  ;装入计数初值 a1
        MOV     AL, 0E8H           ;
        OUT     DX, AL

        MOV     DX, I08254_COUNT1  ;装入计数初值 ah
        MOV     AL, 03H           ;
        OUT     DX, AL

        MOV     DX, I08254_MODE      ;初始化 8254 工作方式
        MOV     AL, 10110110B      ;计数器 2, 方式 3
        OUT     DX, AL

        MOV     DX, I08254_COUNT2  ;装入计数初值 a1
        MOV     AL, 0D0H           ;
    
```

```

                                OUT      DX,AL

                                MOV      DX, IO8254_COUNT2      ;装入计数初值 ah
                                MOV      AL, 07H      ;
                                OUT      DX,AL

                                MOV      AX,4C00H      ;返回到 DOS
                                INT      21H

                                CODE  ENDS
                                END      START

```

3) 原因分析

把 8254 的计数器 1 和计数器 2 级联(把计数器 1 的 OUT),把 2MHz 为时钟源分频为 1Hz,以 LED 灯作为输出显示。计数器 1 和计数器 2 都工作在方式 3,计数器 1 的初值为 1000 (03E8H),计数器 2 的初值为 2000 (07D0H)。

实验结论 (讨论):

按照如上方式接线并烧录程序之后,可以观察到 LED 灯以 1Hz 的频率闪烁,说明功能正确。

8254 芯片主要有两方面的功能:

- 1. 计数器: 设置好计数初值之后,便开始减 1 计数,减为 0 时,输出一个信号。计数器可以用于 CPU 产生中断信号。
- 2. 定时器: 设置好定时常数之后,进行减 1 计数,减为 0 时输出一个信号,再重新开始定时。定时器可以用于进行一些定时重复的工作,如定时检测、定时扫描、定时中断等。

在 8254 的六种工作模式中,工作方式 0145 是计数器,工作方式 23 是定时器,下表简单描述了六种工作方式各自的功能。

工作方式编号	描述
0	计数结束产生中断
1	可编程的单稳态触发器
2	分频器
3	方波发生器
4	软件触发的选通信号发生器
5	硬件触发的选通信号发生器

实验名称：实验二 8255 并口控制器应用实验

实验目的： 1) 掌握 8255 的工作方式及应用编程。

2) 掌握 8255 典型应用电路的接法。

实验要求：按要求完成实验内容，并撰写实验报告。

坚决杜绝抄袭！

实验环境：软件环境：Win7 32 位操作系统

编程语言：汇编语言

硬件环境：CZ-CIUS 微机接口实验系统

实验内容：

实验 2-1：基本输入输出实验

根据你的学号末位的奇偶来选择设计输入输出的接线。其中，

- 学号末位为奇数的，在设计接线以及填写代码时，选择用 A 口输入、B 口输出，实现基本输入输出实验；
- 学号末尾为偶数的，在设计界限以及填写代码时，选择用 C 口输入、A 口输出，实现基本输入输出实验。

编写程序，完成拨动开关到数据灯显示的数据传输。要求只要开关拨动，对应的数据灯的亮灭就改变。

根据原理补全代码中缺失的部分，根据自己的设计，补全接线图，并将代码和对应的接线图写在实验报告中。

实验 2-2：自设计实验

自行设计完成本实验，要求，利用数码管和 8255 芯片，设计并实现 0-9 数字在数码管上的循环显示。该实验完成后，需要经老师验收，并在实验报告中画出接线图，给出对应的程序代码。

实验结果与分析：

实验 2-1：基本输入输出实验

1) 实验源码及分析

```
ASSUME CS: CODE ,DS:DATA
```

```
DATA SEGMENT
```

```
    I08255_MODE      EQU    28BH
```

```
    I08255_A         EQU    288H
```

```
    I08255_B         EQU    289H
```

```
    I08255_C         EQU    28AH
```

```
DATA ENDS
```

; 我学号为 41724081, 用 A 口输入, B 口输出

```
CODE SEGMENT
```

```
    ASSUME CS: CODE
```

```
START:  MOV DX, I08255_MODE      ; 8255 初始化
```

```
        MOV AL,10010000B        ; (填空) 端口 A 方式 0 输入, 端口 B 方式 0
```

输出

```
        OUT DX, AL
```

```
INOUT:  MOV DX,I08255_A         ; (填空) A 口读入数据
```

```
        IN AL,DX
```

```
        MOV DX,I08255_B         ; (填空) B 口输出数据
```

```
        OUT DX,AL               ; (填空)
```

```
        MOV DL,0FFH             ; 判断是否有按键
```

```
        MOV AH,06H
```

```
        INT 21H
```

```
        JZ INOUT                ; 若无,则继续
```

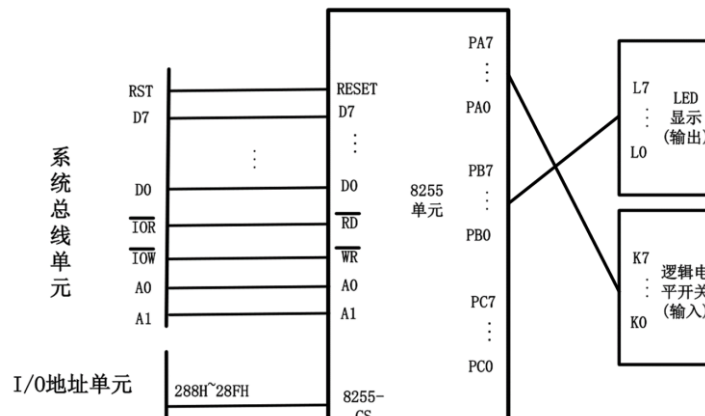
```
        MOV AH,4CH              ; 否则返回
```

```
        INT 21H
```

```
CODE ENDS
```

```
END START
```

2) 补全的接线图

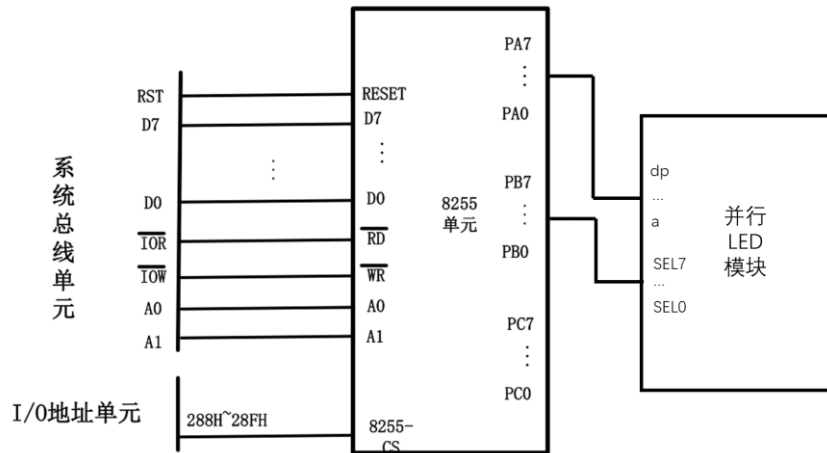


3) 运行结果

拨动实验箱右下区域的逻辑电平开关，对应的 LED 灯会亮起。

实验 1-2：自设计实验

1) 实验接线图



2) 实验源码

```

datasg segment
    shape          db      3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh
    I08255_MODE     equ     28BH
    I08255_A        equ     288H
    I08255_B        equ     289H
    I08255_C        equ     28AH
datasg ends

mystack segment
    dw 128 dup (0)
mystack ends

code segment
    assume ds:datasg,cs:code,ss:mystack
start:
    mov ax,datasg      ; 初始化
    mov ds,ax
    mov ax,mystack
    mov ss,ax
    mov cx,0
    mov si,0

    mov dx,I08255_MODE
    mov al,80H          ; 端口 ABC 输出
    
```

```

out dx,al

    mov dx,I08255_B
    mov al,0FFH
    out dx,al

    mov dx,I08255_A
    mov al,06h
    out dx,al

change_num_loop:
    mov dx,I08255_A
    mov al,shape[si]
    out dx,al
    inc si          ; si++

    push ax
    mov ax,si        ; si = si%10
    mov bx,10
    div bl
    mov al,ah
    mov ah,0
    mov si,ax
    pop ax

    mov cx,5H
delay_1:
    call waitf
    loop delay_1
    jmp change_num_loop

; ----- wait_loop start -----
waitf proc near          ; 延时 700ms
    push cx
    push ax

    mov cx,0FFFFH
waitf_loop:
    in al, 61h
    and al, 10h
    cmp al, ah
    je waitf_loop
    mov ah, al

```

```

loop waitf_loop

    pop ax
    pop cx
    ret
waitf endp
; ----- wait_loop end -----

code ends
end start

```

3) 原因分析

LED 模块是共阳极的 LED, 输入高电平时对应的 LED 管亮起。dp..a 是 8 位的数据信号, 用于控制 LED 灯管; SEL[7..0]是片选信号, 用于选择 LED 管, 高电平有效, 我的代码中直接选择了全部 8 个 LED。

汇编代码中, 第一行 shape 定义了 0-9 的控制信号, 程序中利用循环实现 0-9 的循环显示。考虑到 CPU 频率很高, 因此要使用延时子程序来让数字暂停一会, waitf 子程序通过多次循环来占据时钟周期, 从而实现延时。

实验结论 (讨论):

8255 是并行通信接口, 可以做 CPU 和外设连接的中间级, 使用时先通过控制端口写入控制字, 再从 ABC 端口读/写数据即可。

实验中有延时的要求, 有两种实现方式: 借助计数器或者利用循环占据总线周期。从理论上说, 应当使用计数器+中断的方式实现一段时间的定时, 这样可以避免总线周期的浪费。但是本次实验中, 出于简单的目的, 我使用的是多次循环的方式实现延时。

在我的实验过程中出现了字符显示不正确的现象, 后来发现是数据段没有正确识别, 需要在 code segment 的第一行加入对段寄存器的声明 assume, 否则编译器无法识别数据段的数据。还有, assume 只能写在 code segment 中, 而不能像王爽的《汇编语言》一书中那样把 assume 写在程序开头, 那种格式无法被实验中使用的软件正确识别。

实验名称：实验四 Proteus 仿真设计实验

实验目的： 1) 掌握 Proteus 仿真软件的基本操作。

2) 应用仿真软件实现设计任务。

实验要求： 按要求完成实验内容，并撰写实验报告。

坚决杜绝抄袭！

实验环境： 软件环境：Win7 32 位操作系统

Proteus 仿真软件

编程语言：汇编语言

实验内容：

实验 4-1：Proteus 基本练习

学习 Proteus 软件的操作方法，完成基本操作的学习。然后从以下三个题目中选取其中一个，完成设计，并将设计结果写在验证设计实验报告中。

1) 8255 仿真设计：使用 8255 芯片，实现八盏灯自右向左依次亮灭。

2) 8259 仿真设计：使用 8259 芯片，应用非屏蔽中断，使用开关或按钮控制灯的亮灭，每次按下开关，发送中断请求，灯在亮灭间切换。（注意中断向量的设置）

3) 8253 仿真设计：使用 8253 芯片，实现 Led 灯的定时亮灭，可自行设定灯亮灭的频率。

实验结果与分析：

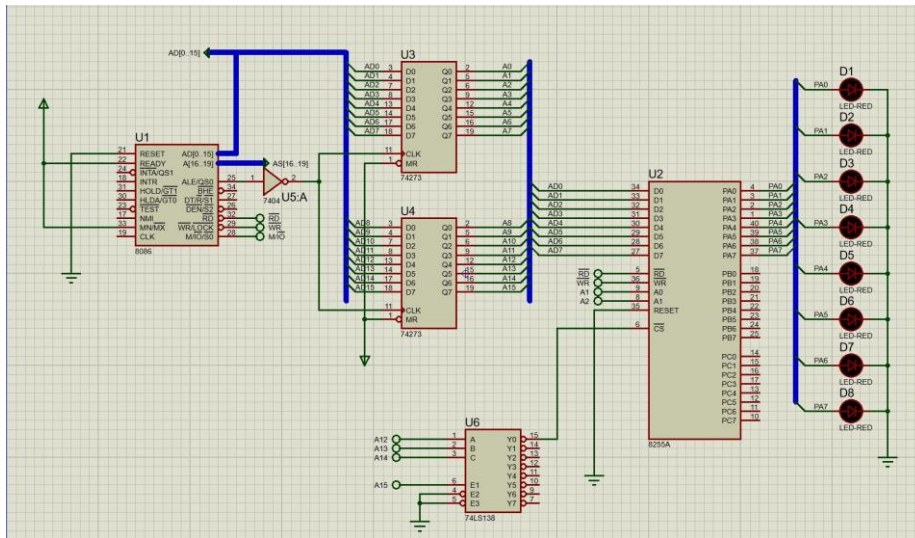
实验 4-1：Proteus 基本练习

1) 仿真选题

选择题目 1，使用 8086 和 8255 芯片，用 8 位并行输出控制 8 个 LED 的亮灭，数据存放在 shape 数组中。设计图中利用 8255 的端口 A 进行输出，工作在方式 0，控制字为 80H。程序中利用循环将 shape[i] 输出给 8255，i 在 0-7 中取值，这里取巧的利用 and 07H 取 i 的低 3 位（即 0-7）实现循环。

关于延时，我使用的仍然是多次循环占据总线的方式，修改 DELAY 子程序的参数 cx 可以控制延时长度。经查阅资料，执行一次 loop 需要大约 5-10 个时钟周期，可以结合设定的 CPU 频率设置 cx 的值，从而实现目标延时。但是如果要实现精确的延时，仍然需要借助定时器。

2) 实验连线图



3) 实验源代码

```
stacks segment stack
    dw 128 dup(?)
stacks ends

data segment
    shape db 01h,02h,04h,08h,10h,20h,40h,80h
    PORTA equ 8000h
    PORTB equ 8002h
    PORTC equ 8004h
    PORTM equ 8006h
data ends

CODE SEGMENT PUBLIC 'CODE'
    ASSUME CS:CODE,ds:data

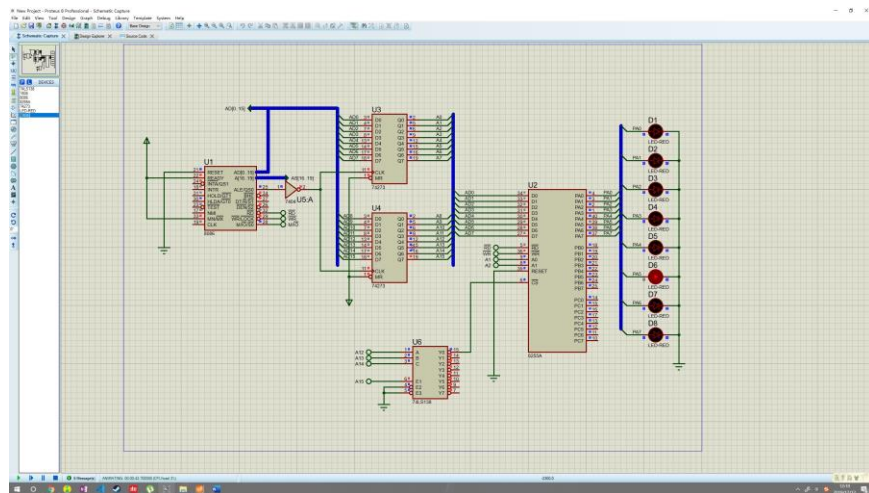
START:
    ; Write your code here
    ; init()
    mov ax,data
    mov ds,ax
    mov dx,PORTM
    mov al,80h ; 1000 0000 端口 ABC 都是输出
    out dx,al
    mov si,0
```

```

change_num_loop:
    mov dx,PORTA    ; 向 A 端口输出灯型
    mov al,shape[si]
    out dx,al
    inc si          ; si++
    and si,0007H    ; 只取低 3 位
    mov cx,0FFFFH
DELAY:
    LOOP DELAY
    jmp change_num_loop
ENDLESS:
    JMP ENDLESS
CODE    ENDS
        END START

```

4) 运行结果



D1-D8 从上至下依次亮起，图中显示的是 D6 亮起时的情形。

实验结论（讨论）：

我的设计中参照给出的样例设计了 8086 和 74273 的连接，并通过 74LS138 实现 8255A 的片选信号译码。需要注意的是，8255A 连接的信号不是 74273 缓存的信号，而是直接连接到 8086 的 AD 输出。

下面总结一些实验中用到的技巧：

1. Proteus 中利用标线的 Label 区分连线，选中连线，右键菜单的倒数第二个选项可以显示直连的线；
2. 按下 A 键可以打开快速属性分配；
3. 不必把所有的线都连接到一起，只要保证 Label 相同即可。