

计算机接口实验 指导书

（在线实验版）

郑榕 主编

北京科技大学
计算机与通信工程学院
计算机专业实验室
2022 年 11 月

目 录

目 录	II
第 1 章 接口实验平台系统（仿真）	- 1 -
第 1.1 节 仿真环境	- 1 -
第 2 章 微机接口验证设计实验	- 2 -
第 2.1 节 8259 中断控制器应用实验	- 2 -
一、实验目的	- 2 -
二、实验内容	- 2 -
三、实验原理与步骤	- 2 -
四、实验报告内容及要求	- 7 -
第 2.2 节 8254 定时/计数器应用实验	- 8 -
一、实验目的	- 8 -
二、实验内容	- 8 -
三、实验原理与步骤	- 8 -
四、实验报告内容及要求	- 10 -
第 2.3 节 8255 并口控制器应用实验	- 11 -
一、实验目的	- 11 -
二、实验内容	- 11 -
三、实验原理与步骤	- 11 -
四、实验报告内容及要求	- 14 -
第 3 章 微机接口综合设计实验	- 15 -
第 3.1 节 创新实验	- 17 -
一、实验目的	- 17 -
二、实验内容	- 17 -
三、实验报告内容及要求	- 18 -
第 4 章 微机接口仿真软件实验平台系统	- 19 -
第 4.1 节 PROTEUS 操作界面	- 19 -
第 4.2 节 PROTEUS 实验示例	- 22 -
一、实验内容	- 22 -
二、实验环境	- 22 -
三、操作步骤	- 22 -
四、实验运行结果	- 36 -
五、参考资料	- 37 -
第 4.3 节 PROTEUS 设计实验	错误!未定义书签。
一、实验目的	错误!未定义书签。
二、实验内容	错误!未定义书签。
三、实验报告内容及要求	错误!未定义书签。
附 录	- 19 -
附录一、INT 21H 指令说明及使用方法	- 19 -
附录二、I386 汇编指令集	- 45 -
一、数据传输指令	- 45 -
二、算术运算指令	- 48 -
三、逻辑运算指令	- 49 -
四、串指令	- 49 -
五、程序转移指令	- 51 -
六、伪指令	- 52 -
七、位测试指令	- 53 -
附录三、74 系列芯片功能列表	- 55 -

第1章 接口实验平台系统（仿真）

第1.1节 仿真环境

汇编编译器：EMU8086

仿真实验编译器：本实验选用 Proteus8.8 版，详细操作参看实验指导书第 4 章及所给演示视频。由于 Demo 版软件只能用于查看接线图、元件连线，无法保存与仿真运行，所以实验验证请参照“课程要求”上的说明进行。

第2章 微机接口验证设计实验

第2.1节 8259 中断控制器应用实验

一、实验目的

1. 掌握 PC 机中断处理系统的基本原理。

二、实验内容

1. PC 机内中断嵌套实验。使用单次脉冲模拟两个中断源的中断产生，填写中断处理程序，体会中断嵌套的过程。

三、实验原理与步骤

1. 实验原理

(1) PC 微机系统中的 8259 介绍

中断控制器 8259 是 Intel 公司专为控制优先级中断而设计开发的芯片。它将中断源优先级排队、辨别中断源以及提供中断矢量的电路集于一片中，因此无需附加任何电路，只需对 8259 进行编程，就可以管理 8 级中断，并选择优先模式和中断请求方式，即中断结构可以由用户编程来设定。同时，在不需增加其他电路的情况下，通过多片 8259 的级连，能构成多达 64 级的矢量中断系统。它的管理功能包括：1) 记录各级中断源请求，2) 判别优先级，确定是否响应和响应哪一级中断，3) 响应中断时，向 CPU 传送中断类型号。

8259A 的命令字共有 7 个，可分为两类。一类是初始化命令字，另一类是操作命令字。8259 的编程就是根据应用需要将初始化命令字 ICW1-ICW4 和操作命令字 OCW1- OCW3 分别写入初始化命令寄存器组和操作命令寄存器组。ICW1-ICW4 各命令字格式如表 1-1 所示，OCW1-OCW3 各命令字格式如表 1-2 所示，其中 OCW1 用于设置中断屏蔽操作字，OCW2 用于设置优先级循环方式和结束方式的操作命令字，OCW3 用于设置和撤消特殊屏蔽方式，设置中断查询方式以及设置对 8259A 内部寄存器的读出命令。

表 2-1 初始化命令字

	D7	D6	D5	D4	D3	D2	D1	D0
ICW1	X	X	X	1	1: 电平触发 0: 边沿触发	X	1: 单片 8259 0: 多片 8259	1: 需要 ICW4 0: 不需要 ICW4
ICW2	中断向量的高 5 位					X	X	X
ICW3 (主)	分别对应 8 个 IRQ, Di=1, I 有 RQi 上级连芯片, =0, 无级连芯片							
ICW3 (从)	0	0	0	0	0	指明从片连接在主片的哪个 IRQ 上		
ICW4	0	0	0	1: 特殊全嵌套 0: 非特殊	1: 缓冲 0: 非缓冲	1: 主片	1: 自动 EOI 0: 非自	1: 8086 系统

				全嵌套		0: 从	动 EOI	0 : 8080 系统
--	--	--	--	-----	--	------	-------	----------------

表 2-2 操作命令字

	D7	D6	D5	D4	D3	D2	D1	D0
OCW1	Di=0: 开放 IRi 中断请求 Di=1: 禁止 IRi 中断请求							
OCW2	优先级循环控制位	L2~L0 有效控制位	中断结束命令位	0	0	中断级编码		
OCW3	X	0X: 无意义 10: 撤销特殊屏蔽 11: 设置特殊屏蔽		0	1	中断查询控制位	0X: 无意义 10: 读 IRR 寄存器 11: 读 ISR 寄存器	

(2) PC 微机系统中的 8259A

PC 机用户可使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。

在 80x86 系列 PC 微机系统中，系统中包含了两片 8259 中断控制器，通过级连可以管理 15 级硬件中断，但其中部分中断号已经被系统硬件占用，具体情况如表 2-3 所示。两片 8259 的端口地址为：主片 8259 使用 020H 和 021H 两个端口；从片使用 0A0H 和 0A1H 两个端口。系统初始化两片 8259 的中断请求信号均采用上升沿触发，采用全嵌套方式，优先级的排列次序为：0 级最高，依次为 1 级、8 级~15 级，然后是 3 级~7 级。

表 2-3 PC 微机系统中的硬件中断

中断号	功能	中断向量号	中断向量地址
主 8259 IRQ0	时钟/计数器 0	08H	0020H~0023H
主 8259 IRQ1	键盘	09H	0024H~0027H
主 8259 IRQ2	接从片 8259	0AH	0028H~002BH
主 8259 IRQ3	串行口 2	0BH	002CH~002FH
主 8259 IRQ4	串行口 1	0CH	0030H~0033H
主 8259 IRQ5	并行口 2 (硬盘)	0DH	0034H~0037H
主 8259 IRQ6	软盘	0EH	0038H~003BH
主 8259 IRQ7	并行口 1 (并行打印机)	0FH	003CH~003FH
从 8259 IRQ8	实时钟	70H	01C0H~01C3H
从 8259 IRQ9	用户中断	71H	01C4H~01C7H
从 8259 IRQ10	保留	72H	01C8H~01CBH
从 8259 IRQ11	保留	73H	01CCH~01CFH

从 8259 IRQ12	保留	74H	01D0H~01D3H
从 8259 IRQ13	协处理器中断	75H	01D4H~01D7H
从 8259 IRQ14	硬盘控制器	76H	01D8H~01DBH
从 8259 IRQ15	保留	77H	01DCH~01DFH

注意：

1、CZ-CIUS-USB 实验系统总线区的 IRQ 接到了 3 号中断 IRQ3 上，即进行中断实验时，所用中断类型为 0BH。USB 核心板上的 IR10 接到了 10 号中断 IRQ10 上，所用中断类型为 072H。

2. 实验步骤

(1) 实验 1-1：PC 机内中断嵌套实验

实验要求：

- ① 请对下面给出的程序参考代码，进行注释，注明每句代码起到了的作用。
- ② 请用所学理论知识解释什么是嵌套中断。

接线图：

说明：本实验接线图如图 2-1 所示，从单脉冲单元引出两根导线，一根接到系统总线单元区的 IRQ 引脚，一根接到 USB 核心卡的 IRQ10 端。

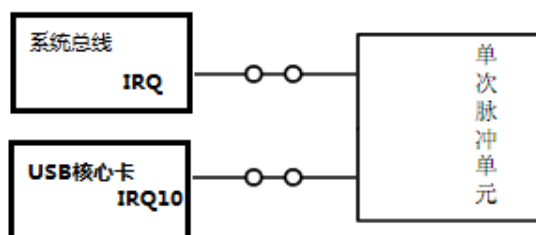


图 2-1 8259 实验 2 接线图

参考流程图（如图 2-2）：

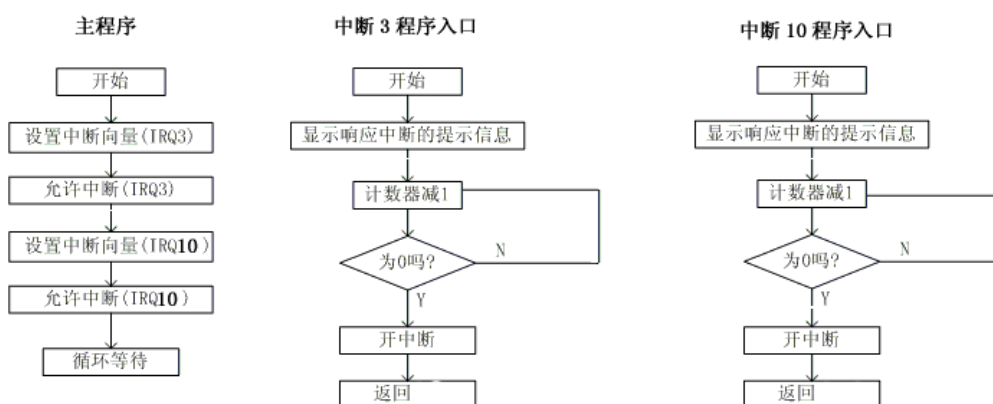


图 2-2 8259 实验 2 程序流程图

程序源代码：

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

.386

CLI

MOV AX,CS

MOV DS,AX

MOV DX,OFFSET INT10

MOV AX,2572H

INT 21H

MOV DX,OFFSET INT3

MOV AX,250BH

INT 21H

IN AL,21H

AND AL,0F3H

OUT 21H,AL

IN AL,0A1H

AND AL,0FBH

OUT 0A1H,AL

MOV CX,10

STI

WAIT: JMP WAIT

INT10: CLI

PUSHAD

PUSHFD

MOV CX,10

NEXT10_1:

MOV DX,59H

MOV AH,02H

INT 21H

MOV DX,20H

MOV AH,02H

INT 21H

```
CALL DELAY1
LOOP NEXT10_1

MOV DX,0DH
MOV AH,02H
INT 21H
MOV DX,0AH
MOV AH,02H
INT 21H

MOV AL,20H
OUT 0A0H,AL
OUT 20H,AL
POPF
POPAD
STI
IRET

INT3:  CLI
        PUSHAD
        PUSHFD
        MOV CX,10
NEXT3_1:
        MOV DX,4EH
        MOV AH,02H
        INT 21H
        MOV DX,20H
        MOV AH,02H
        INT 21H

        CALL DELAY1
        LOOP NEXT3_1

        MOV DX,0DH
        MOV AH,02H
        INT 21H
        MOV DX,0AH
```

```
        MOV AH,02H
        INT 21H

        MOV AL,20H
        OUT 20H,AL
        OUT 0A0H,AL
        POPFD
        POPAD
        STI
        IRET
DELAY1  PROC
        PUSHAD
        PUSHFD
        MOV CX,0FH
DELAY_LOOP1:
        MOV BX,0FFFFH
DELAY_LOOP2:
        DEC BX
        NOP
        JNZ DELAY_LOOP2
        LOOP DELAY_LOOP1
        POPFD
        POPAD
        RET
DELAY1  ENDP

CODE ENDS
END START
```

四、实验报告内容及要求

1. 根据实验 1-1 的实验要求中的加黑字体要求，填写实验报告。
 - 1) 请**对实验 1 所给出的实验代码，进行注释**，标注各语句的主要功能。
 - 2) 请用所学原理**详细解释中断嵌套是如何实现的**。

第2.2节 8254 定时/计数器应用实验

一、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。

二、实验内容

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。
2. 自设计实验。参考实验一的程序，编写程序，以 1MHz 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。

三、实验原理与步骤

1. 8254 简介

8254 是 Intel 公司生产的可编程定时器。是 8253 的改进型，它的操作方式以及引脚和 8253 完全相同，比 8253 具有更优良的性能。8254 具有以下基本功能：

- (1)有 3 个独立的 16 位计数器；
- (2)每个计数器可按二进制或十进制(BCD)计数；
- (3)每个计数器可编程工作于 6 种不同工作方式；
- (4)8254 每个计数器允许的最高计数频率为 10MHz(8253 为 2MHz)；
- (5)8254 有读回命令(8253 没有)，除了可以读出当前计数单元的内容外，还可以读出状态寄存器的内容。
- (6)计数脉冲可以是有规律的时钟信号，也可以是随机信号。计数初值公式为 $n = f_{clk} \div f_{out}$ 。

其中 f_{clk} 是输入时钟脉冲的频率， f_{out} 是输出波形的频率。

8254 的工作方式如下述：

- (1)方式 0：计数到 0 结束输出正跃变信号方式。
- (2)方式 1：硬件可重触发单稳方式。
- (3)方式 2：频率发生器方式。
- (4)方式 3：方波发生器。
- (5)方式 4：软件触发选通方式。
- (6)方式 5：硬件触发选通方式。

表 2-4 8254 的方式控制字格式

D7	D6	D5	D4	D3	D2	D1	D0
计数器选择		读 / 写格式选择		工作方式选择		计数码制选择	
00 一计数器 0		00 一锁存计数值		000 一方式 0		0 一二进制数	
01 一计数器 1		01 一读 / 写低 8 位		001 一方式 1		1 一十进制数	

10 一计数器 2	10 一读 / 写高 8 位	010 一方式 2	
11 一读出控制字标志	11 一先读 / 写低 8 位 再读 / 写高 8 位	011 一方式 3 100 一方式 4 101 一方式 5	

8254 的控制字有两个：一个用来设置计数器的工作方式，称为方式控制字；另一个用来设置读回命令，称为读回控制字。这两个控制字共用一个地址，由标识位来区分。控制字格式如表 2-1 所示。读回控制字格式如表 2-2 所示。当读回控制字的 D4 位为 0 时，由该读回控制字 D1-D2 位指定的计数器的状态寄存器内容将被锁存到状态寄存器中。状态字格式如表 2-3。

表 2-5 8254 读出控制字格式

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0 一锁存计数 值	0 一锁存状态信息	计数器选择(同方式控制字)			0

表 2-6 8254 状态字格式

D7	D6	D5	D4	D3	D2	D1	D0
OUT 引脚现行状态 1 一高电子 0 一低电 子	计数初值是否装入 1 一无效计数 0 一计数有效	计数器方式(同方式控制字)					

2. 实验步骤

(1) 实验 2-1：计数器应用实验

实验要求：

①将计数器 0 设置为方式 0，设定计数器初值为 5，将代码中划横线的部分填上相应的代码，并在实验报告中所填代码做原理分析。

说明：运行源程序 8254.asm，按接线图接线，实现计数器功能。

接线图：

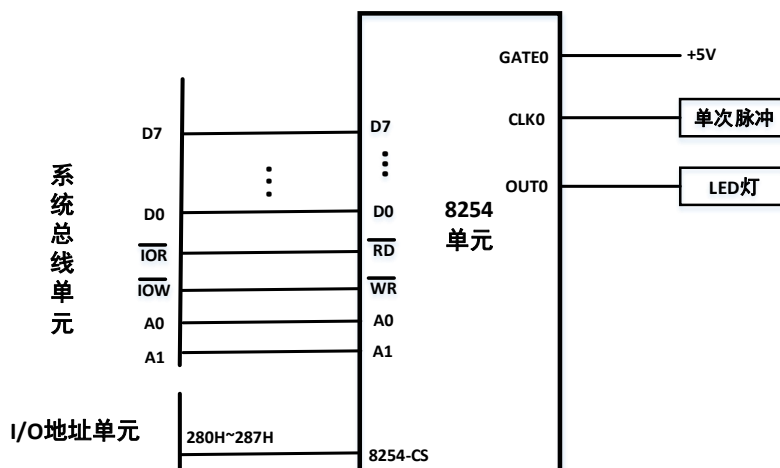


图 2-5 8254 实验 1 接线图

程序源代码：

```
IO8254_MODE EQU 283H ;8254 控制寄存器端口地址
IO8254_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, IO8254_MODE      ;初始化 8254 工作方式
        MOV AL,____ (1) _____ ;计数器 0, 方式 0
        OUT DX, AL

        MOV DX, ____ (2) _____ ;装入计数初值
        MOV AL,____ (3) _____
        OUT DX, AL

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

CODE ENDS
END START
    
```

(2) 实验 2-2: 自设计实验

实验要求:

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 **1MHz** 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。请根据所学理论知识, 在作业中画出接线图, 并给出源代码。

四、实验报告内容及要求

1. 根据实验 2-1 实验要求中的加黑字体要求, 填写实验作业。

①将代码中划横线的部分 (1)– (3) 填上相应的代码, 并在实验作业中对所填代码做原理分析。

2. 按实验要求完成自设计实验, 在实验报告中画出接线图, 给出实现设计要求功能的程序源码, 如果需要验证, 请参看“课程要求”。

第2.3节 8255 并口控制器应用实验

一、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。

二、实验内容

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的数据传输。要求只要开关拨动，数据灯的显示就改变。

2. 自设计实验。利用 8255 芯片，设计一个流水灯，实现 8 个 LED 灯中的最左边的两个灯从左向右循环显示。

三、实验原理与步骤

1. 8255 简介

并行接口是以字节或字为单位与 I/O 设备或被控制对象之间传递信息。CPU 和接口之间的数据传送总是并行的，即可以同时传递 8 位、16 位、32 位等。Intel 公司的可编程通用并行接口芯片 8255 具有 A、B、C 三个并行接口。提供以下三种工作方式：

- 方式 0--基本输入/出方式
- 方式 1--选通输入/出方式
- 方式 2--双向选通方式。

8255 的引脚如图 2-8 所示，8255 的工作方式控制字和 C 口按位置位/复位控制字格式如表 2-7 (a) 和 (b) 所示。

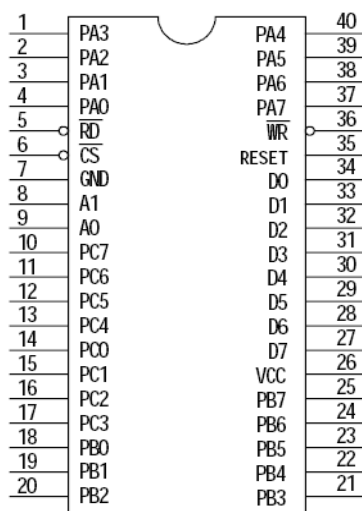


图 2-6 8255 的引脚图

表 2-7 (a) 工作方式控制字

D7	D6	D5	D4	D3	D2	D1	D0
1	00: 端口 A 方式 0		0: 端口	0: 端口 C 高	0: 端口 B	0: 端口	0: 端口 C 低四
	01: 端口 A 方式 1		A 输出	四位输出	方式 0	B 输出	位输出
	1X: 端口 A 方式 2		1: 端口	1: 端口 C 高	1: 端口 B	1: 端口	1: 端口 C 低四
			A 输入	四位输入	方式 1	B 输入	位输入

表 2-7 (b) C 口按位置位/复位控制字

D7	D6	D5	D4	D3	D2	D1	D0
0	X	X	X	PC0: 000 PC3: 011 PC6: 110	PC1: 001 PC4: 100 PC7: 111	PC2: 010 PC5: 101	0: 复位 1: 置位

2. 实验步骤

(1) 实验 3-1: 可编程并行接口 8255 方式 0 应用实验

实验要求:

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 8255 接线时, 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 8255 接线时, 选择用 C 口输入、A 口输出, 实现基本输入输出实验。

根据接线图, 填写程序中缺失的代码, 完成拨动开关到 LED 灯显示的数据传输。要求只要开关拨动, 对应的 LED 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作业中。

说明：运行源程序 8255.asm，按接线图接线，实现并行数据传输功能。

接线图：

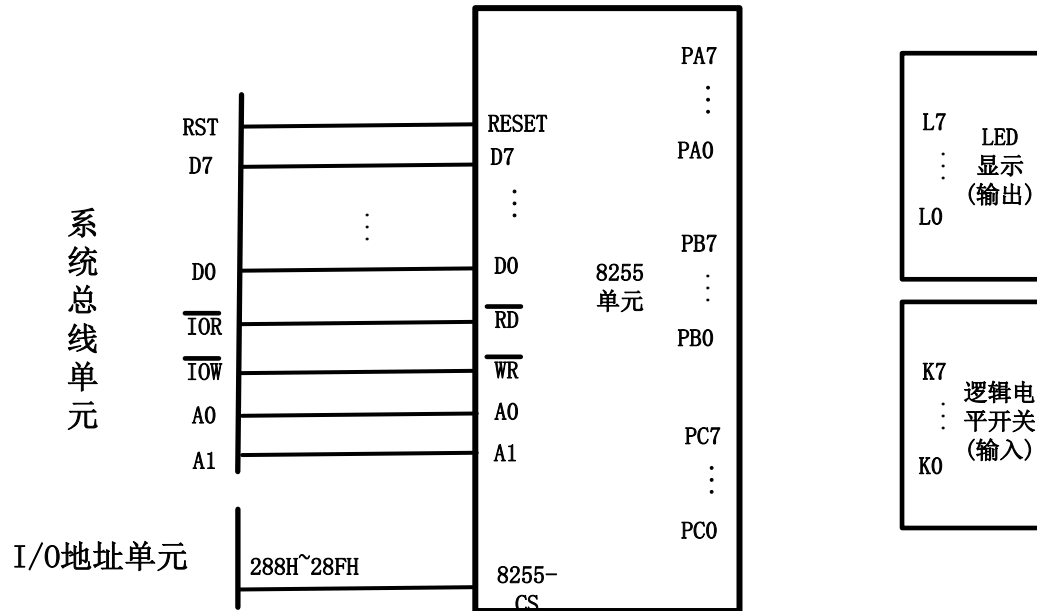


图 2-7 8255 实验 1 接线图

程序源代码：

```

IO8255_MODE      EQU    28BH
IO8255_A          EQU    288H
IO8255_B          EQU    289H
IO8255_C          EQU    28AH

CODE SEGMENT
    ASSUME CS: CODE
START: MOV DX, IO8255_MODE          ;8255 初始化
      MOV AL, ____ (1) ____
      OUT DX, AL
INOUT: MOV DX, ____ (2) ____        ;读入数据
      ____ (3) ____
      MOV DX, ____ (4) ____        ;输出数据
      ____ (5) ____
      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) 实验 3-2: 自设计试验

实验要求:

自行设计完成本实验,利用 8255 芯片,设计一个流水灯,实现 8 个 LED 灯中的最左边的一个灯从左向右循环显示。

四、实验报告作业及要求

1. 根据本节实验 1 实验要求中的加黑字体要求,补全代码中缺失的部分(1)–(5),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作业中。
2. 完成自设计实验,并在实验报告中给出所编程序和接线图,如果需要验证,请参看“课程要求”。

第3章 汇编综合设计实验

一、实验目的

1. 掌握汇编语言的编程。

二、实验内容

请在以下三道题目中选择一道，编程完成。

- 1、编写汇编程序实现以下完整功能：

- 1) 程序运行后，在屏幕上显示提示信息：“Please input 10 numbers:”
- 2) 根据提示，用键盘输入 10 个数，数的范围为 0-99;
- 3) 将输入的 10 个数从大到小排序，显示在屏幕上;
- 4) 统计 0-36、37-59、60-74、75-99 的数各是所少个;
- 5) 最后屏幕的显示格式如下：(XX 代表刚才输入的 10 个数，每个数之间以空格隔开，YY 代表统计出来的这个区间内的个数)

Sorted numbers: XX XX XX

0-36: YY

37-59: YY

60-74: YY

75-99: YY

- 2、编写汇编程序，实现以下完整功能：

程序运行后，在屏幕上显示当时的日期和时间。

- 1) 显示程序运行当天是今年的第多少天。
- 2) 最后屏幕的显示格式如下：(XX 代表是第多少天的具体数字，假设程序运行时为 2021.11.17 下午 14:30)

Now is 2:30P.M, Wednesday, Nov 17, 2021

Today is the XX day。

- 3、编写汇编程序，实现以下功能：

- 1) 程序运行后，在屏幕上显示提示信息：“Please input the number:”
- 2) 在屏幕上输入一个数（0-20 之间），然后显示斐波那契数列中前这么多个数的。
- 3) 计算这些数的总和。
- 4) 最后屏幕的显示格式如下：(XX 代表斐波那契数列中的值，YY 表示所有数相加的总和)

The sequence is: XX XX XX.....

The sum is: YY

第4章 微机接口综合设计实验

第4.1节 创新实验

一、实验目的

学习各接口芯片功能及应用特性，能够组合多个接口芯片，完成设计实验。

二、实验内容

根据已学习各接口芯片的应用特性，参考本讲义前述实验接线图及代码，自己设计完成综合设计实验。要求所选多个芯片或模块的加权值总和 ≥ 2 ，权值列表见表 3-1 所示。并且，所选芯片或模块中必须包含 8259 或 8254 芯片的其中一个作为功能模块部分。

表 3-1 各芯片模块加权值

模块类型	芯片/模块	权值
功能模块	8259	1
	8254	1
输入	PC 机键盘输入	0.5
	4*4 小键盘	1
输出	PC 机屏幕显示	0.5
	数码管、点阵等显示	1
其他	LCD 液晶屏	1
	步进电机	1
	AD0809	1
	实验箱其他功能芯片	0.5-1
复杂代码逻辑		1

特别要求：

- 1、2022 年创新实验不得设计交通灯、电子琴/八音盒、简单流水灯/跑马灯/流水式霓虹灯、抢答器、投票器。
- 2、用仿真软件 Proteus 实现多模块创新设计。
- 3、不能是验证实验中代码的简单叠加，需要包含一定量自己编写的汇编代码，在验收时需说明自己所编写的代码量大概是多少。
- 4、代码逻辑具备一定的复杂度，可以折抵 1 分加权分，但在验收时，需要提供程序流程图。
- 5、请参看附录中关于 Proteus 的操作，学习 Proteus 的使用，然后在 Proteus Demo 完成接线图的设计，注意，由于 Demo 版软件不能保存、不能仿真，所以请按照“课程要求”文档中的说明完成仿真设计。
- 6、按照你的设计编写代码，完成代码设计工作。
- 7、实验设计的验证调试请参看“课程要求”
- 8、通过视频验收后，请针对你的设计录制一段视频，详细解释你的设计，解释内容中要包括你的设计的应用场景、实现的主要功能、所选的芯片模块、每个芯片的工作方式、最

后在仿真软件中进行运行及操作的展示。

三、实验报告内容及要求

设计并完成本实验，提交实验报告。实验报告中至少应包含以下内容：

设计题目、实现功能、应用场景、设计思路、设计连线图、程序流程图、仿真设计与实现的实验源代码、实验现象描述以及对本门课程的学习心得或思考。

注：流程图的标准画法请参看课程资料中的《GB 1526-89 流程图规范》。

附录

附录一、微机接口仿真软件实验平台系统

附 1.1 Proteus 操作界面

运行 Proteus 8 Professional，显示如图 4-1、4-2 的 Proteus 8.8 操作界面。

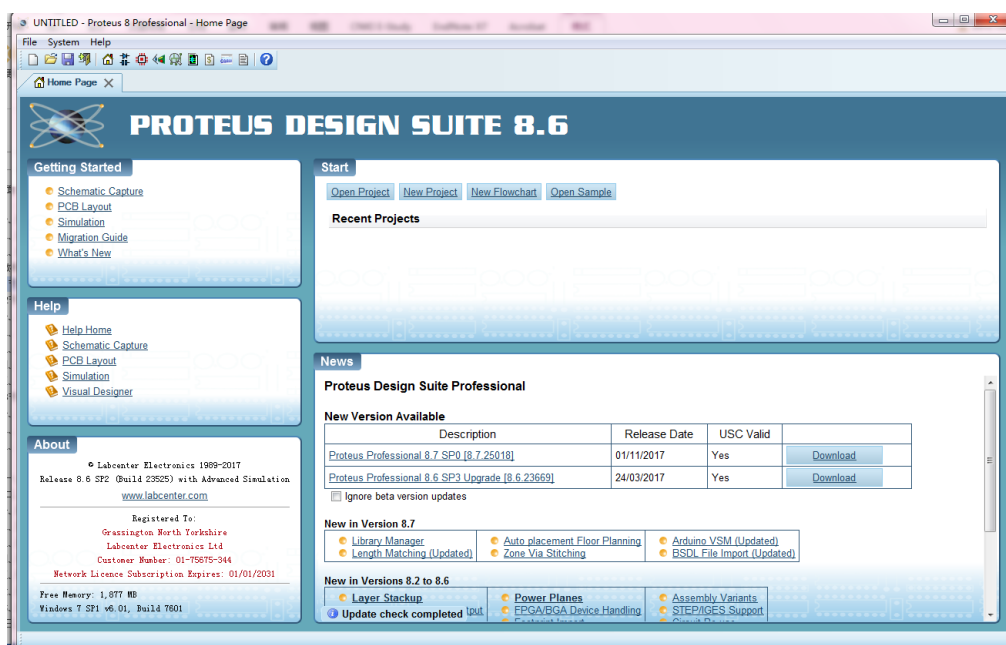
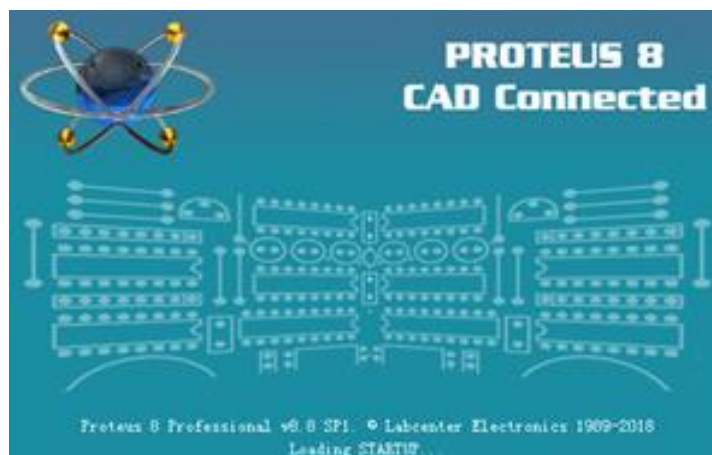


图 4-1 Proteus8.6 启动界面

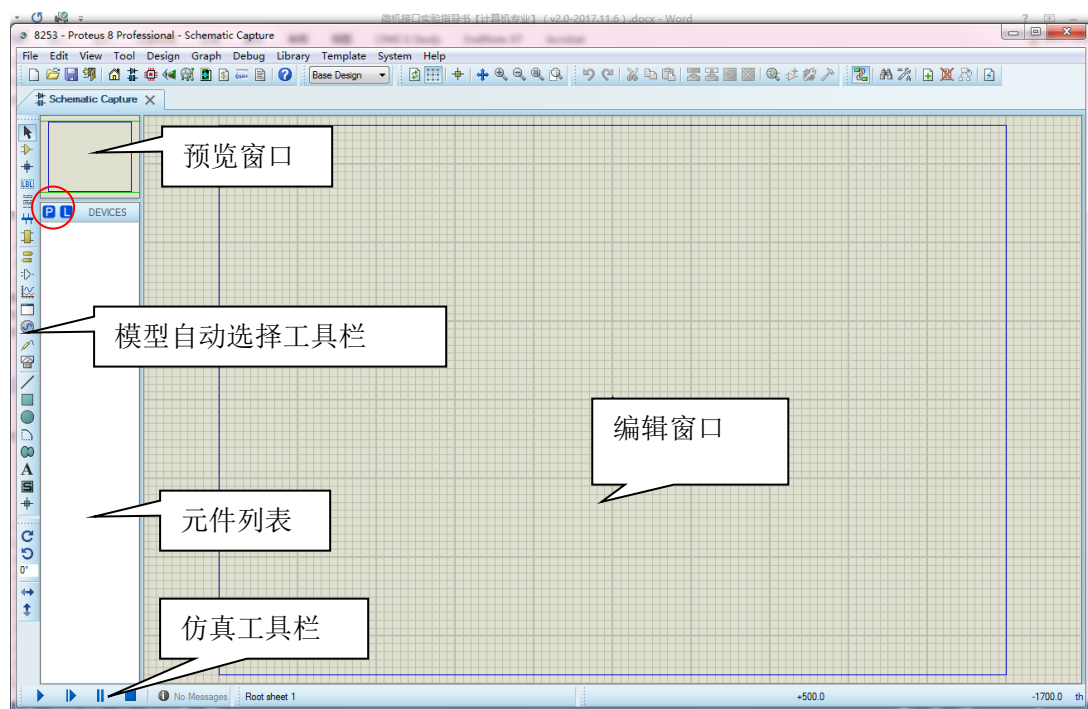


图 4-2 Proteus 8 操作界面

分为以下几个部分：

编辑窗口 (The Editing Window): 用于绘制设计原理图。蓝色方框内为可编辑区，所有设计元件需放置在蓝色编辑区内。可以通过预览窗口来改变原理图的可视范围。

预览窗口 (The Overview Window): 该窗口可以显示两种内容，第一，在元件列表选中某个元件时，可以显示该元件的预览图；第二，当鼠标焦点落在编辑窗口时，可以显示编辑窗口的缩略图，调整缩略图中的绿色方框，可以改变编辑窗口的可视范围，可以通过鼠标滚轮调整绿色方框的大小，放大/缩小编辑窗口的可视范围。



模型选择工具栏 (Mode Selector Toolbar): 罗列所有可选元件模型或工具模型。自上至下分别为：

- 选择元件 (components) (默认选择的)
- 放置连接点
- 放置标签 (用总线时会用到)
- 放置文本
- 用于绘制总线 (Buses Mode)
- 用于放置子电路
- 用于即时编辑元件参数 (先单击该图标再单击要修改的元件)
- 终端接口 (terminals): 有 VCC、地、输出、输入等接口
- 器件引脚: 用于绘制各种引脚
- 仿真图表 (graph): 用于各种分析, 如 Noise Analysis
- 录音机

- 信号发生器 (generators)
- 电压探针：使用仿真图表时要用到
- 电流探针：使用仿真图表时要用到
- 虚拟仪表：有示波器
- 画各种直线
- 画各种方框
- 画各种圆
- 画各种圆弧
- 画各种多边形
- 画各种文本
- 画符号
- 画原点

元件列表 (The Object Selector)：用于挑选元件 (components)、终端接口 (terminals)、信号发生器 (generators)、仿真图表 (graph) 等。举例，当你选择“元件 (components)”，单击“P”按钮会打开挑选元件对话框，选择了一个元件后 (单击了“OK”后)，该元件会在元件列表中显示，以后要用到该元件时，只需在元件列表中选择即可。

仿真工具栏：用于设计的仿真运行。

图 4-2 上红色圆圈圈定的两个按钮， 用于从库中挑选元件， 用于管理元件库。

附 1.2 Proteus 实验示例

本节以 8254 计数实验为例，帮助读者了解 Proteus 仿真软件实现对 8086 微机接口程序设计的基本步骤。

由于在 Proteus 元件库中没有 8254 芯片的元件模型，因此选用与之功能相近的 8253 元件模型完成芯片设计。

一、实验内容

使用 8253A 可编程芯片元件模型，应用 Proteus 仿真设计软件，实现计数功能。

要求：设置计数器 0，工作在方式 0，使用二进制计数，计数初值为 3。

二、实验环境

操作系统：Win7 32 位

仿真软件：Proteus 8.8

三、操作步骤

1、创建工程

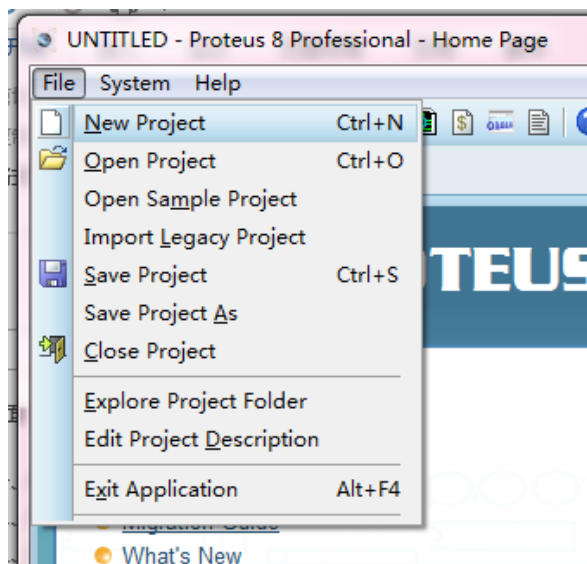


图 4-3 创建工程菜单栏

选择“New Project”，通过创建向导，创建一个项目工程。

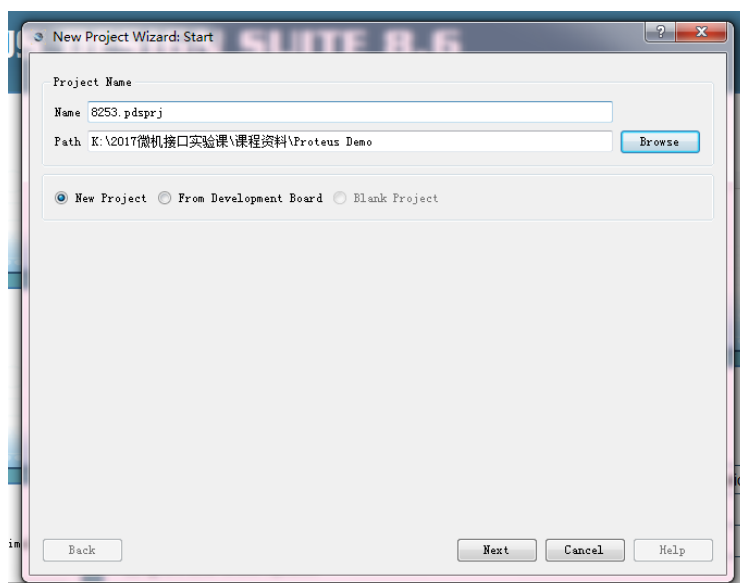


图 4-4 创建项目向导

在向导中可以修改新建项目的名称及保存路径，项目文件的后缀是*.pdsprj，且 Proteus 7 以前的版本软件无法打开该类项目文件。

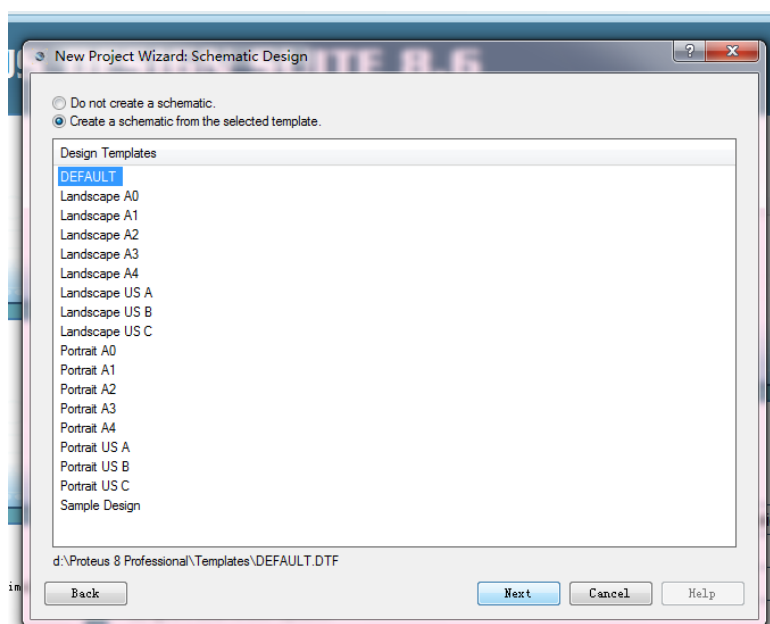


图 4-5 创建项目向导-1

单击 Next，进入原理图设置图纸尺寸界面。

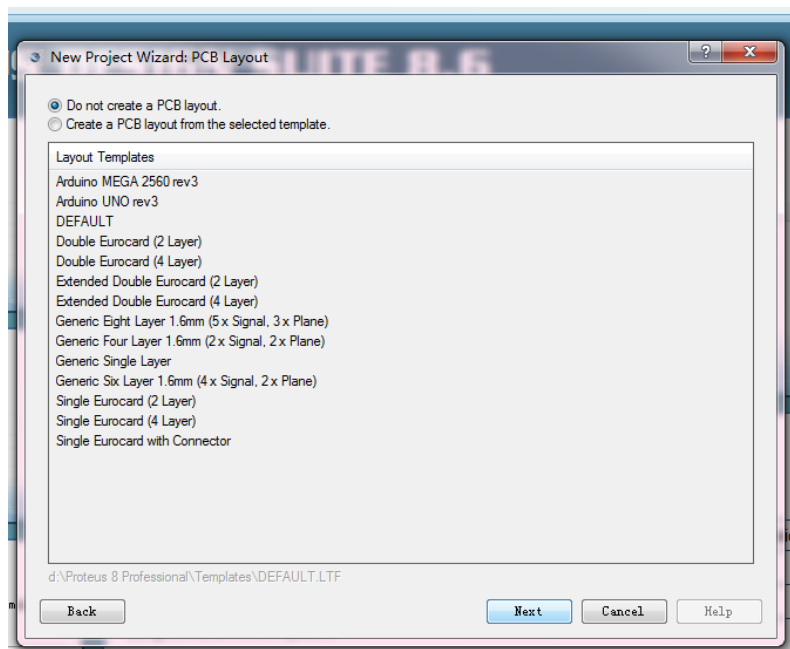


图 4-6 创建项目向导-2

单击 Next，进入 PCB 参数设置界面。

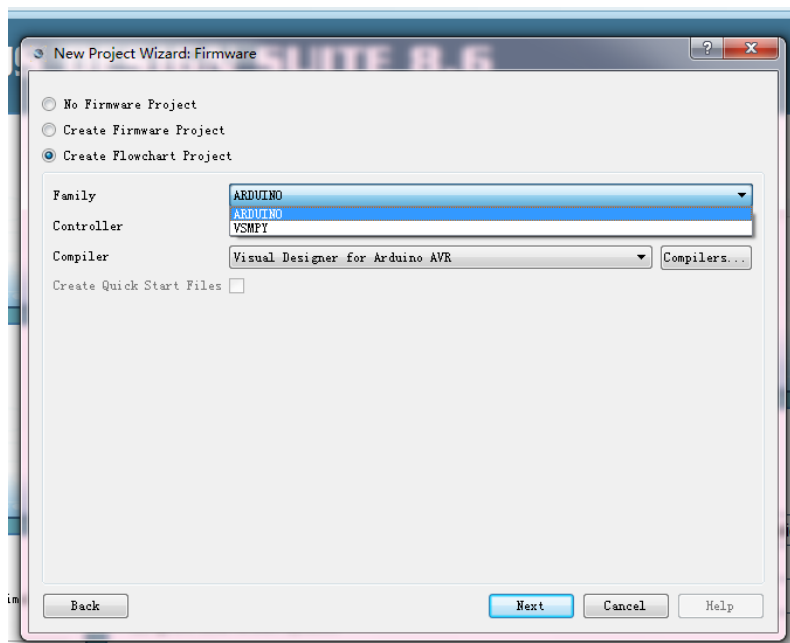


图 4-7 创建项目向导-3

选择 No Firmware Project，不创建 PCB 板，或根据设计的微处理器系列选择相应的 PCB 板，均可。如果选定了微处理器的类型，还需要选择相对应的编译器，如果没有安装对应的编译器，则需要下载安装后才能使用。本实验不选择 PCB 板。单击进入下一步。

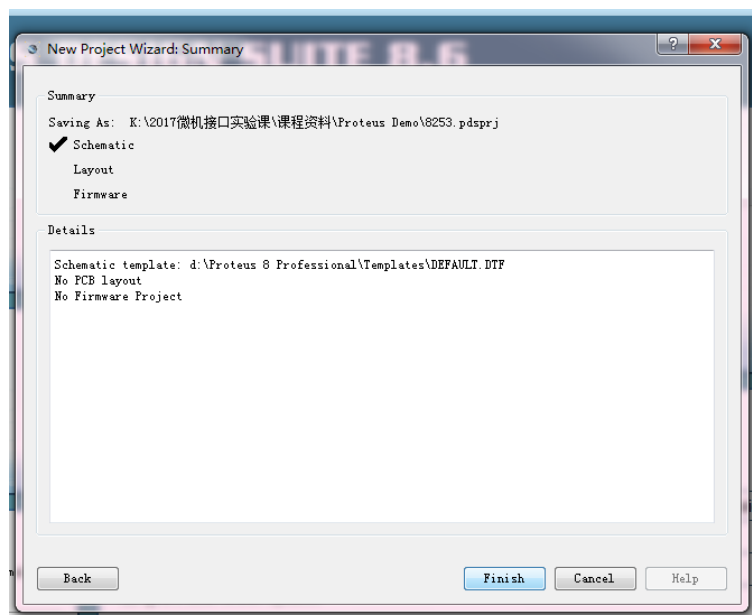



图 4-8 创建项目向导-4

项目初始化到这一步结束。

2、根据功能，选取元件

本实验应用到的元件包括：

元件	所在元件库	功能
8086	Microprocessor ICs	8086 仿真元件，作为整个设计的核心芯片
74273	TTL 74 series	8 位的数据锁存器
74LS138	TTL 74 series	3 线-8 线译码器
7404	CMOS 4000 series	反相器
8253A	Microprocessor ICs	定时/计数芯片
BUTTON	Active	开关
RES	Device	电阻
LED-RED	Optoelectronics	LED 灯（红色）
GROUND	Terminals Mode	接地
POWER	Terminals Mode	电源

点击图 4-1 中的“模型自动选择工具栏”中的“选择元件”（Component mode），在预览图下方的元件列表中，点击 ，弹出元件选取窗口。如图 4-9 所示。

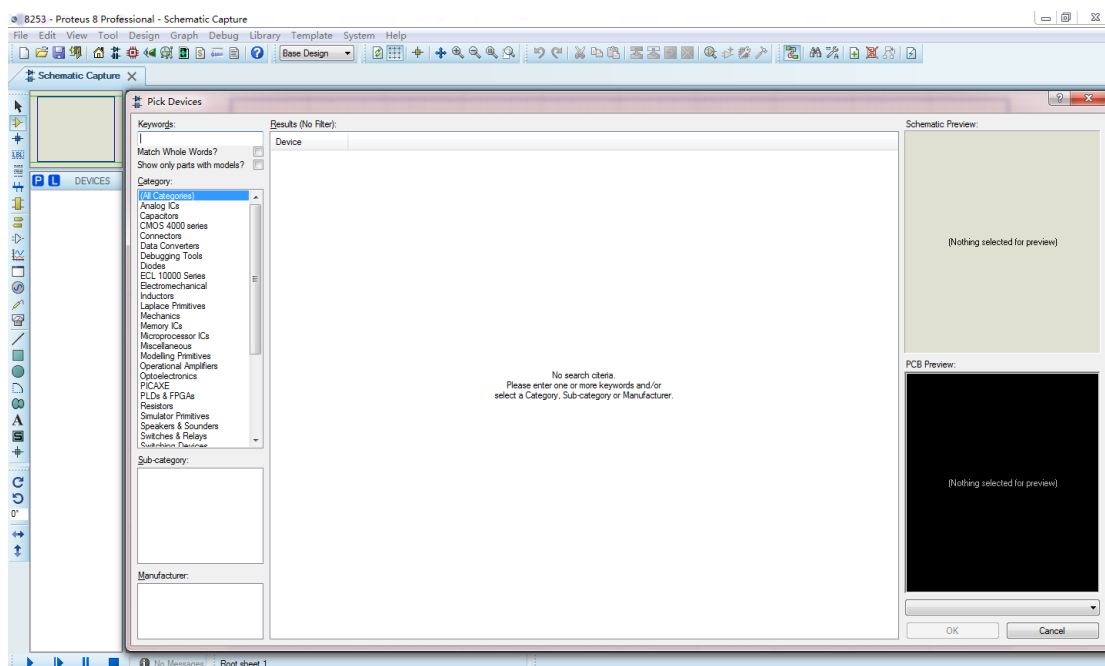
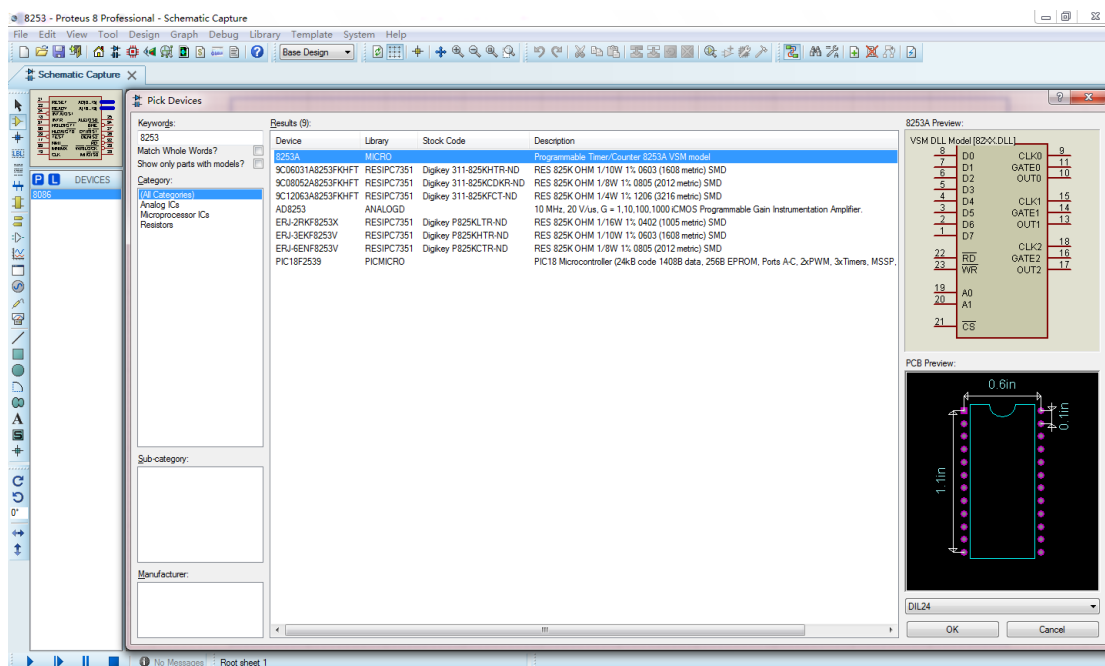


图 4-9 元件选取窗口

在关键字中输入元器件的名称，例如 RES，可以在搜索结果中找到对应的元器件，例如 RES 对应的元器件为“电阻”，如图 4-9 所示。选中搜索结果，双击选中的元件，在左侧的元件列表中可看到点选的元件。点击右下角的 OK 键，返回主界面。



选择元件结束后，开始建立连线图。

3、连接所有元件，完成连线图

单击元件列表中的元件，将鼠标移动到右侧的编辑窗口，鼠标左键点击在编辑窗口，放置元件，可以看到一个紫红色的元件示意图，如图 4-10 所示：

元件会有相应的标识（例如，R1 为元件的名称、10K 为当前电阻 R1 对应的阻值等）。

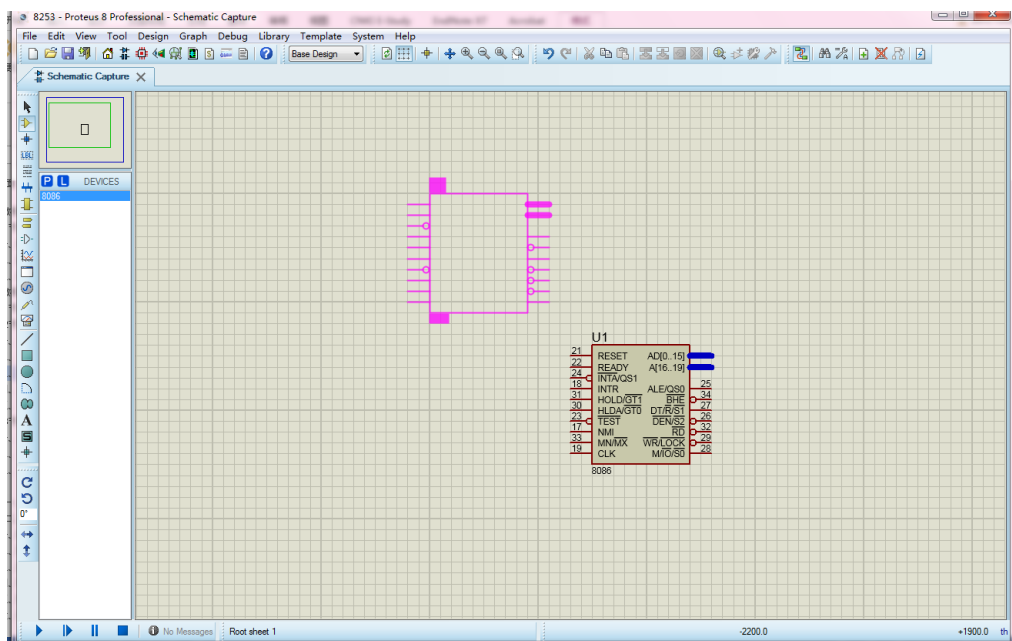


图 4-10 元件拖拽示意图

根据需要，在元件上单击鼠标右键，“Edit Properties”可修改元件的属性，如图 4-11 所示。继续拖拽元件到编辑窗口，已经列在元件列表框内的元件可重复放置，元件的名称会自动生成，也可根据自己的需要修改调整。

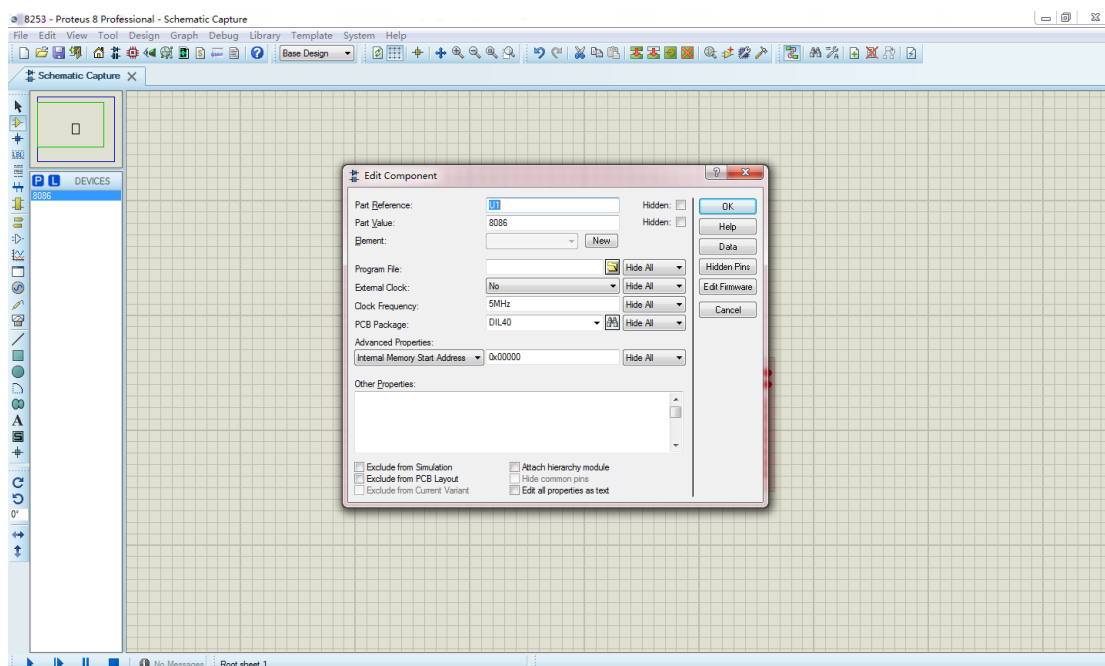


图 4-11 元器件属性修改界面

在 8086 系统中，地址线和数据线是分时复用的，也就是说 8086 的 16 根地址/数据复用线 AD15-AD0 以及 4 根地址/状态复用线 A19-A16 在 1 个总线周期的不同时刻分别传送地址信息或数据，所以必须要用锁存器将地址信息先保存起来，否则地址信息将会丢失。

在 8086 最小系统中，是利用 8282 作为地址锁存器的，如图 4-12 所示。但在 proteus 中没有 8282 和 8286 芯片，因此，需要使用 2-3 片地址锁存器，实现地址所存功能。其中 8086

采用的是 16 位的地址，而地址锁存器是 8 位的，所以地址锁存器使用 2 片即可，如果要实现 20 位地址的话，需要 3 片地址锁存器。

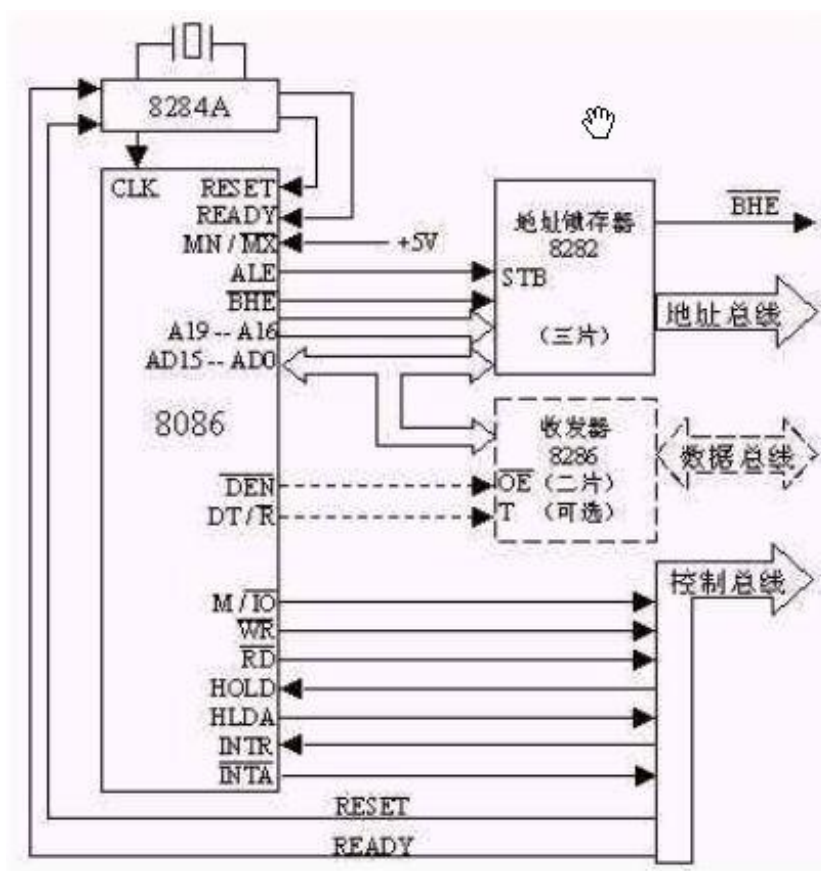


图 4-12 8086 最小系统示意图

可以选择 74273 或 74LS373 作为地址锁存器，其中，74273 是带公共时钟复位八 D 触发器，74LS373 是三态同相八 D 锁存器。二者唯一的区别是第 1、11 引脚功能不同。

对于 74273，它的 1 脚是复位 CLR，低电平有效。当 1 脚是低电平时，输出脚 2(Q0)、5(Q1)、6(Q2)、9(Q3)、12(Q4)、15(Q5)、16(Q6)、19(Q7)全部输出 0，即全部复位；当 1 脚为高电平时，11(CLK)脚是锁存控制端，并且是上升沿触发锁存，当 11 脚有一个上升沿，立即锁存输入脚 3、4、7、8、13、14、17、18 的电平状态，并且立即呈现在输出脚 2(Q0)、5(Q1)、6(Q2)、9(Q3)、12(Q4)、15(Q5)、16(Q6)、19(Q7)上。

对 74LS373，它的 1 脚是输出使能(OE)，是低电平有效。当 1 脚是低电平时，只要 11 脚(锁存控制端，G)上出现一个下降沿，输出 2(Q0)、5(Q1)、6(Q2)、9(Q3)、12(Q4)、15(Q5)、16(Q6)、19(Q7)立即呈现输入脚 3、4、7、8、13、14、17、18 的状态；当 1 脚是高电平时，不管输入 3、4、7、8、13、14、17、18 如何，也不管 11 脚(锁存控制端，G)如何，输出 2(Q0)、5(Q1)、6(Q2)、9(Q3)、12(Q4)、15(Q5)、16(Q6)、19(Q7)全部呈现高阻状态(或者叫浮空状态)。

所以，如果分别用 74273 和 74LS373 来作为地址锁存器的话，对 74273 来说，1(CLR)脚必须接高电平，ALE 信号经过反相后接 11 脚(因为 8086 的 ALE 信号是以下降沿方式出现)；对 74LS373 来说，1 脚接低电平，保证使能，11 脚直接接 8086 的 ALE 信号。

如图 4-13 所示。

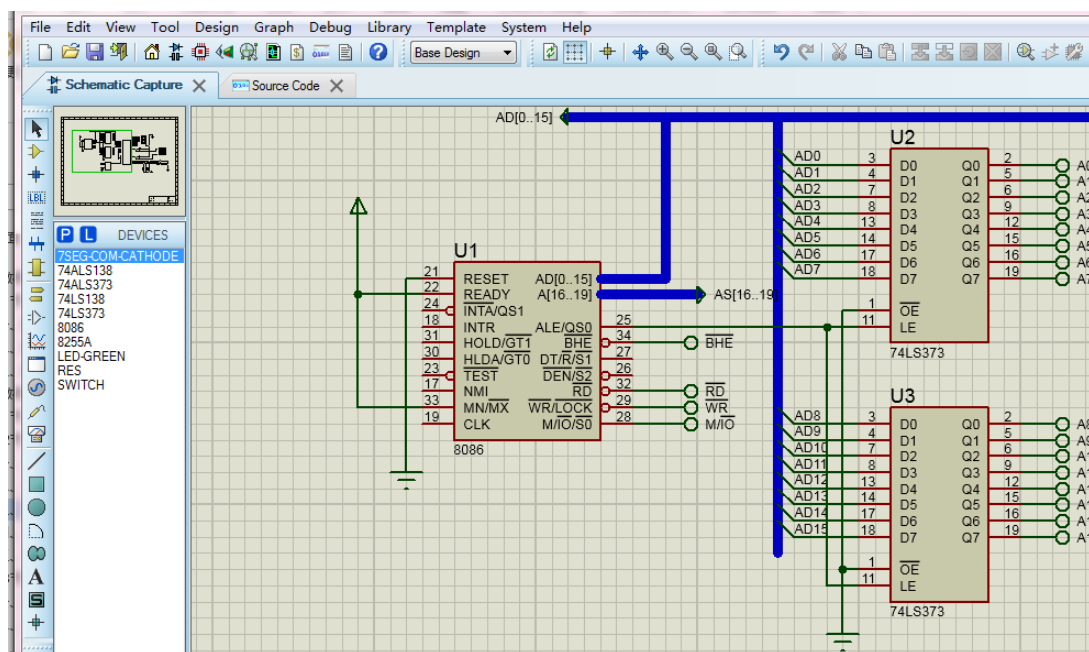


图 4-13 8086 接线与标识方案一

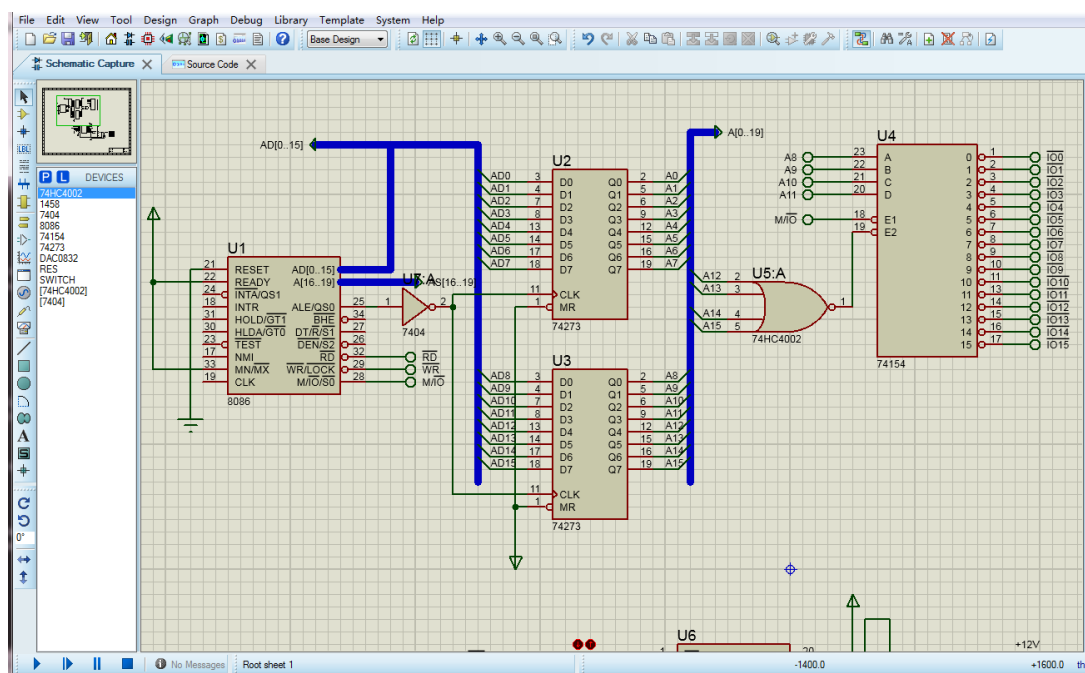


图 4-14 8086 接线与标识方案二

此外，需要为芯片提供一个片选信号，可以选用 74LS138 实现，或者使用 74154 实现。其中 74LS138 可以提供 8 个片选信号，74154 可以提供 16 个片选信号，如图 4-15 所示。

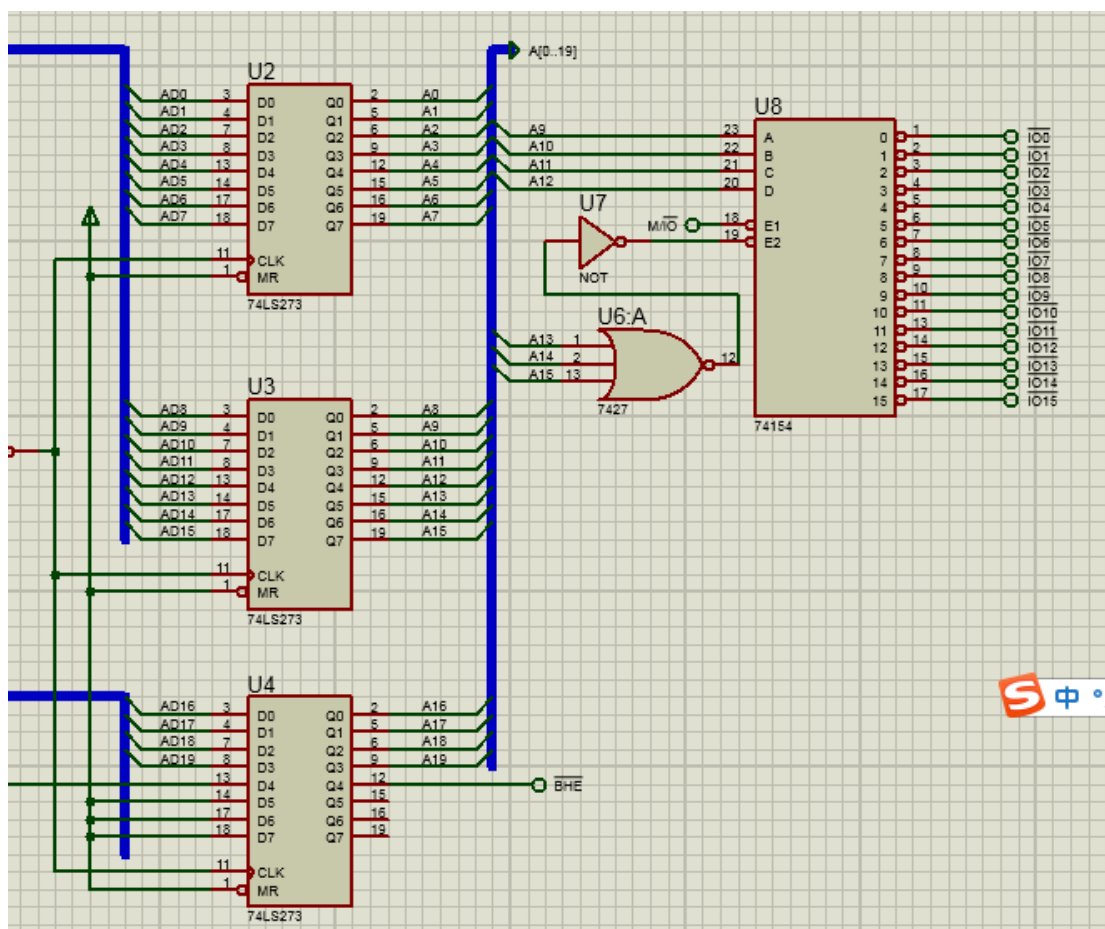
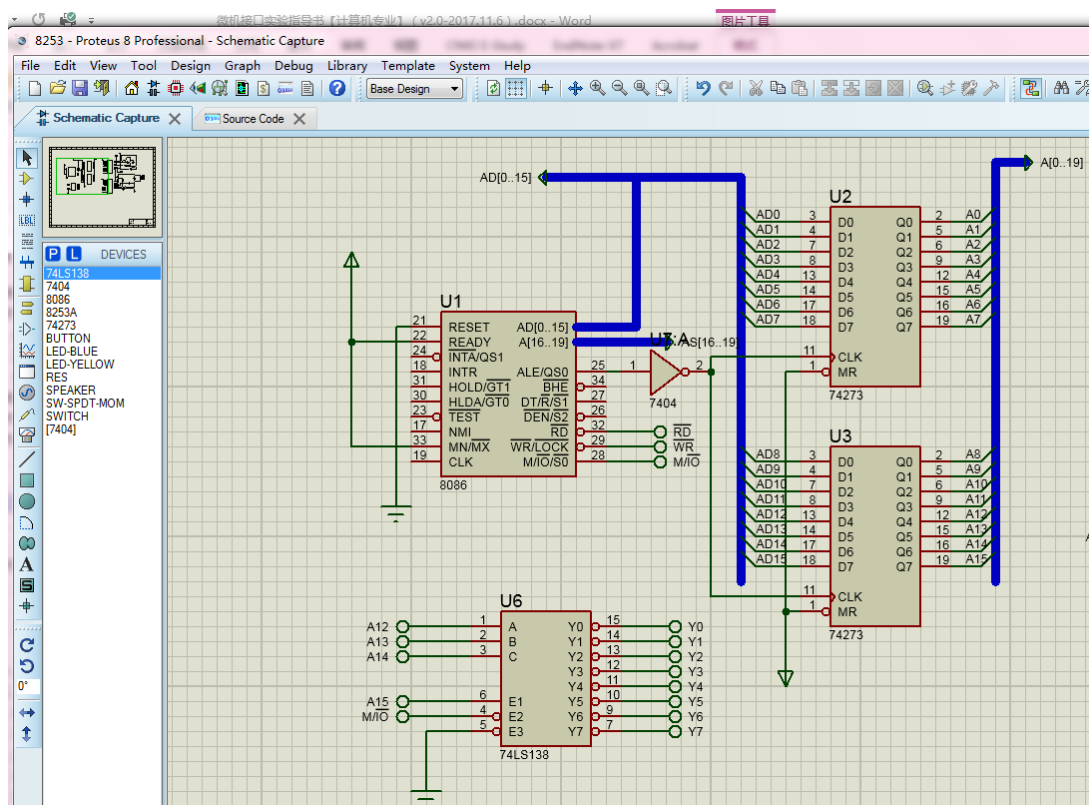


图 4-15 片选信号

从图 4-16 中可以看出，Proteus 中的连接线有两种，一种是单导线，为墨绿色，鼠标焦点置于元件某个管脚处，显示图标变换成笔状，拖动鼠标，可以将两个元件的管脚用连线连接在一起。

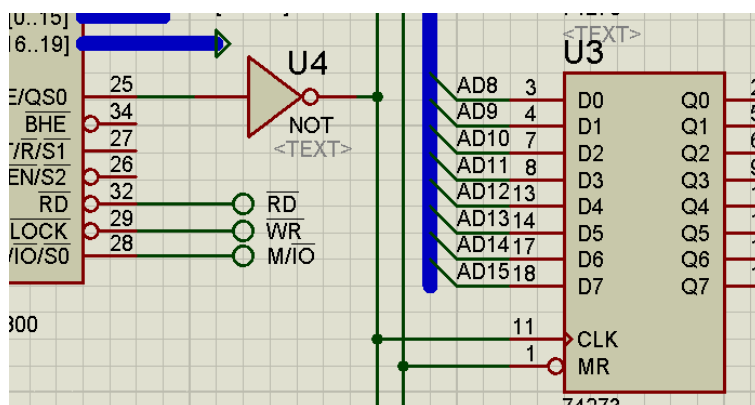


图 4-16 连线示意图

若多根导线连接在一起，会以一个墨绿色圆点标识，如图 4-16 所示。

另外一种连接线是总线模式，如图 4-16 中蓝色粗线所示。使用总线连接时，可将多个不同信号的通过总线逻辑连通在一起。

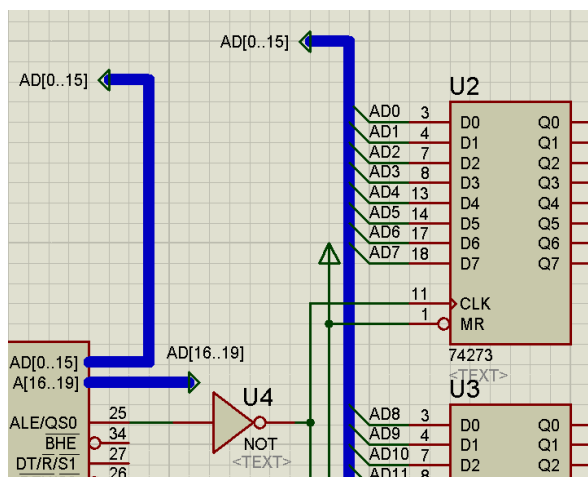


图 4-17 总线连接示意图

图 4-17 中显示的蓝色粗线，为总线连接方式，需要使用总线连接时，点选左侧菜单栏中的“Buses Mode”选项，单击鼠标一次，为总线的起始，拖动鼠标，到预期的位置，双击鼠标，结束总线线体的绘制，然后将元件管脚连接到已绘制的总线线体上，在总线的末端，连接一个总线终端标识（在“Terminals Mode”中可找到），双击总线终端标识，编辑其 string 属性，如图 4-18 所示。例如 AD[0..15]，即代表该总线输入/出的信号为 AD0-AD15。总线连接方式可以简化原理设计图中的连线数量，提高设计图的清晰度。

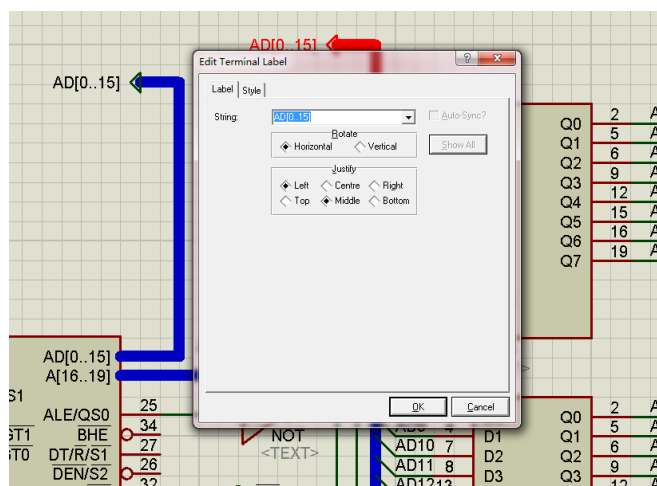


图 4-18 总线标识示意图

依次，完成所有连线图，如图 4-19 所示。

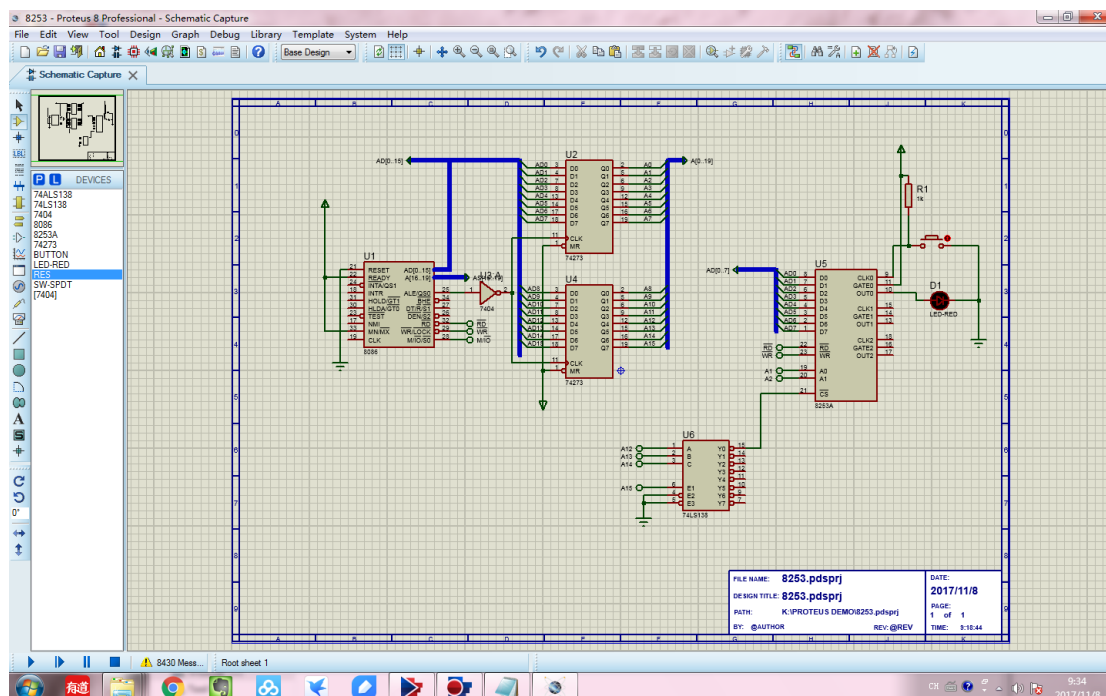


图 4-19 8253A 设计连线图

详细设计图可参见给出的实验 Demo。

4、根据功能，设计源代码

所有元件连接完成后，需要为该设计图添加对应的功能仿真源码文件，使设计图能够按照源码中设定的功能仿真运行。

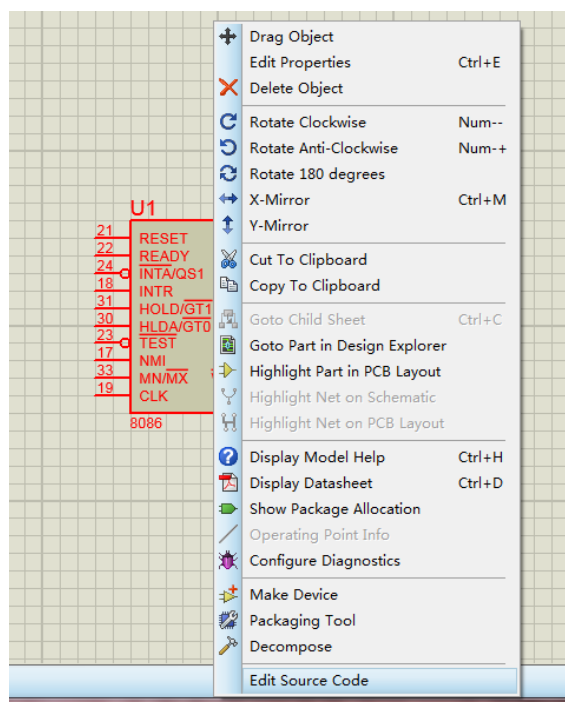


图 4-20 源代码编辑调用菜单项

选中 8086 元件，右键选择“Edit Source Code”，打开代码编辑页面，如图 4-20 所示。如果在创建项目的时候没有选择微处理器的类型，此时会出现微处理类型选择窗口，本项目选择微处理器类型为“8086”，选择编译器为“MASM32”，这里如果系统中没有安装过该编译器，则需要手动添加编译器。

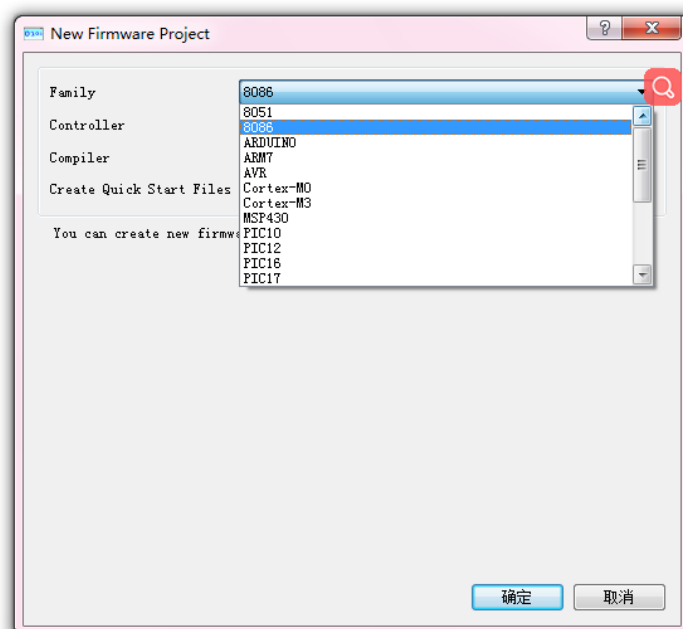


图 4-21 微处理器编译器设定界面-1

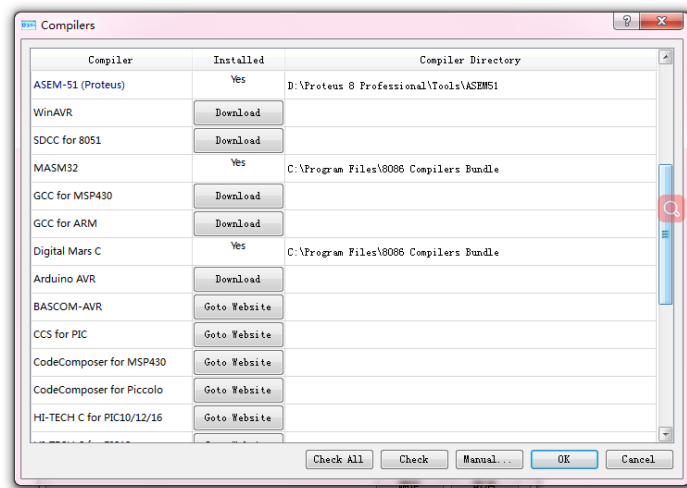


图 4-22 微处理器编译器设定界面-2

选择好微处理器后，软件可自动生成编码框架，如图 4-23 所示。在代码框架中，完成相应编码工作。

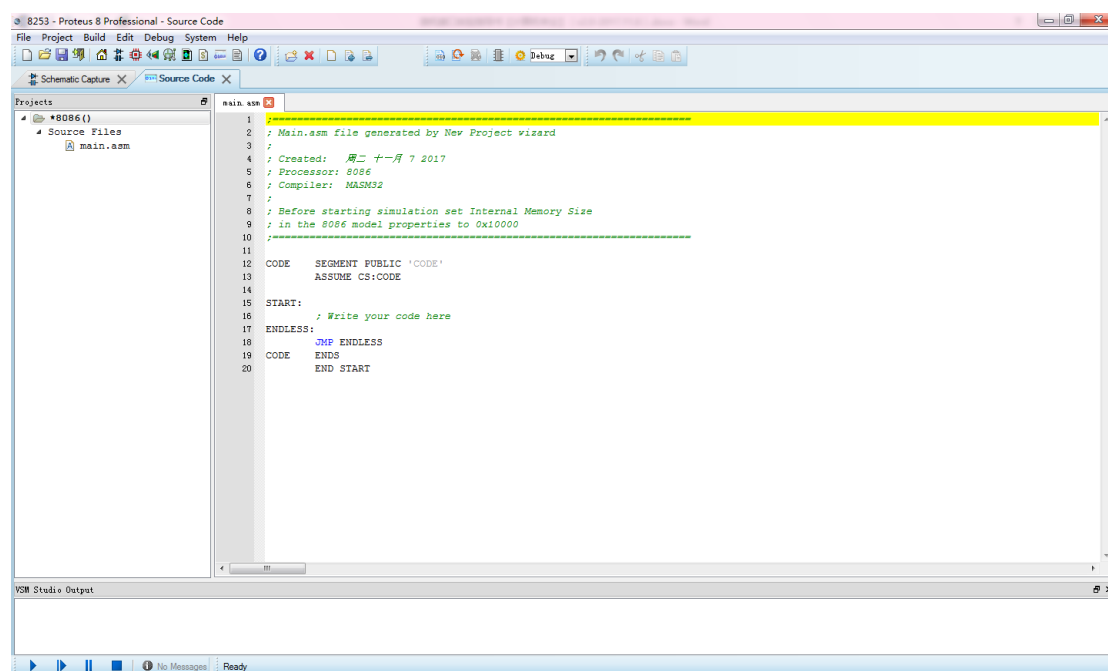


图 4-23 源代码编辑界面-1

根据本实验实现的功能，其汇编源代码如下：

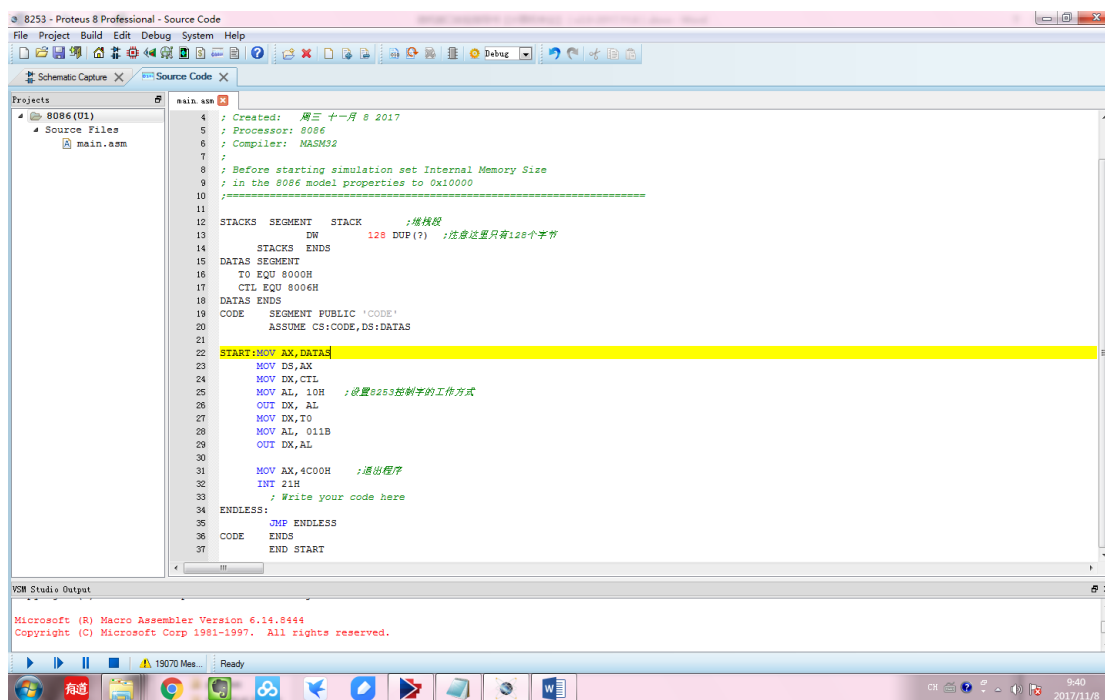


图 4-24 源代码编辑界面-2

汇编代码编写完成后，选择菜单栏里的“Build-Build Project”编译源程序。

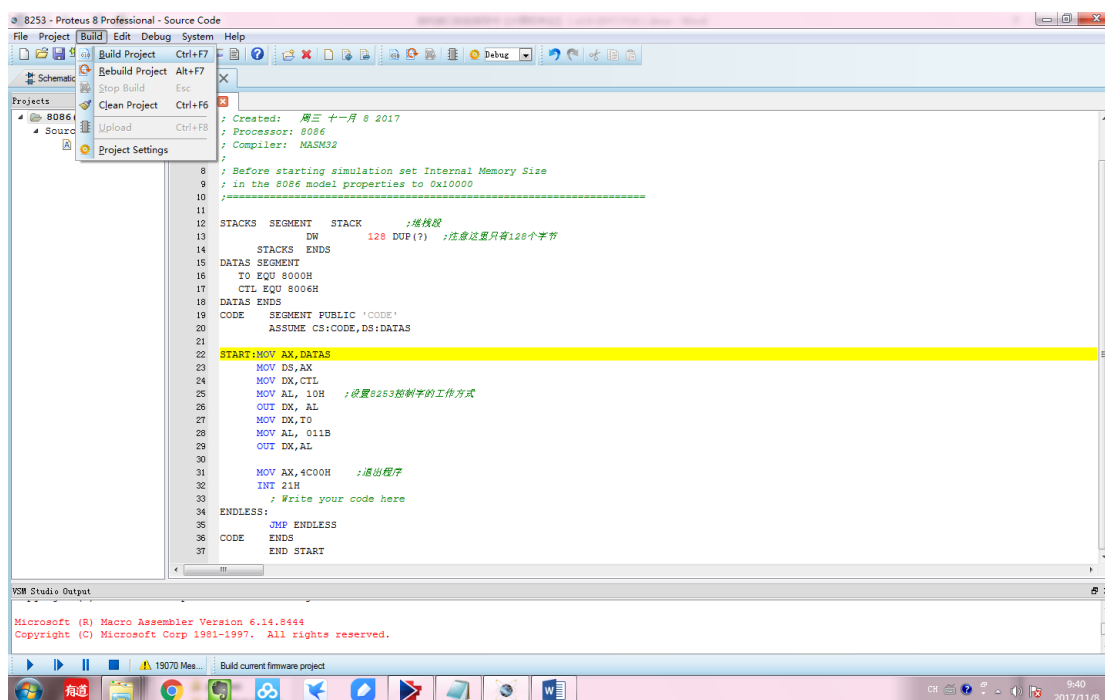


图 4-25 源代码编辑界面-3

5、加载文件，运行调试

由于 Proteus 提供的 8086 仿真元件内没有内置存储器，因此在选择对 8086 相关设计进行仿真时，需要修改 8086 元件的属性参数，例如内存起始地址、内存的大小以及外部程序加载到内存的地址段起始等。

双击 8086 元件，弹出窗体如图 4-26 所示。

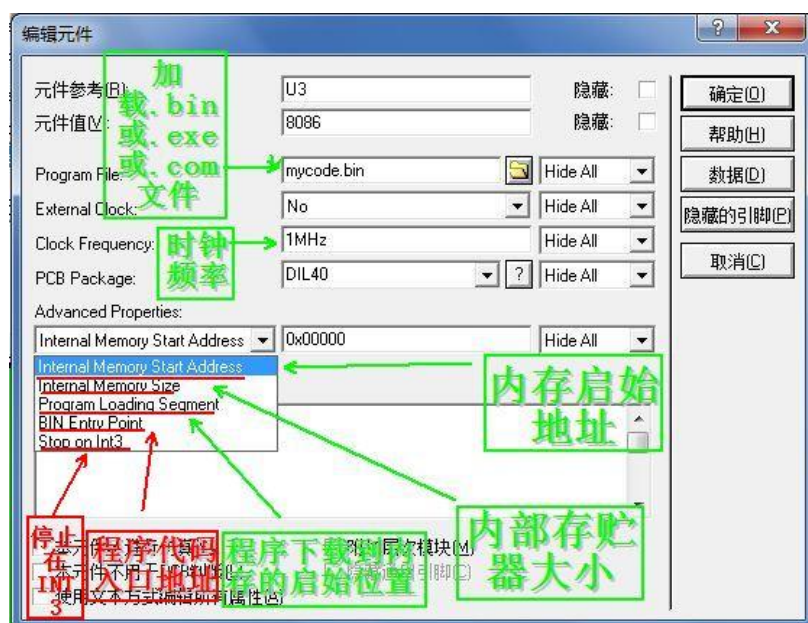


图 4-26 8086 元件属性修改界面

其中，时钟频率可以设为 1MHz，在“Advanced Properties”的下拉选项里，包含了对内置存储器的参数设定。

8086 设置		修改值
Internal memory start address	内部存储器起始地址	0x00000
Internal memory size	内部存储器容量	0x10000
Program loading segment	程序载入段	0x0200
BIN entry Point	程序运行入口地址	0x02000
Stop on int 3	是否在 INT3 处停止	No

Proteus 中的 8086 芯片模型不含操作系统，因此汇编程序不支持 BIOS 和 DOS 的调用，8086 模型可加载的程序类型包括汇编软件生成的文件.EXE 和.bin 或.com 文件，而不需要 PC BIOS 或 DOS 操作系统的支持。

完成设计后，点击仿真运行工具栏的运行按钮，对设计进行仿真，根据仿真的运行结果来判断设计是否正确，如果存在偏差，需调整设计图或仿真文件，直至验证正确。

四、实验运行结果

本实验根据程序源码的设计，仿真时显示效果为，点击开关四次后，LED 灯点亮。如图 4-25 所示。

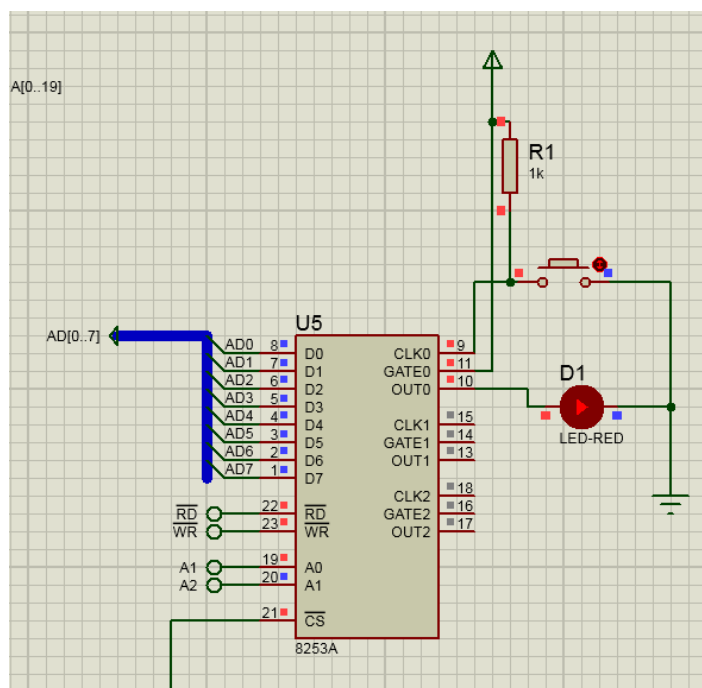


图 4-27 实验运行效果示意图

五、参考资料

本节所写的所有设计内容，仅作为设计的其中一种参考，可根据其它书籍学习其它的设计思路与参数设置。

由于篇幅有限，对于 Proteus 的设计介绍仅摘取了重点步骤，一些应用细节，可参考以下书籍：

《微机原理与接口技术——基于 8086 和 Proteus 仿真》

《微机原理与接口技术:基于 Proteus 仿真的 8086 微机系统设计及应用》

《微机原理与接口技术实验:基于 Proteus 仿真》

《Proteus 从入门到精通 100 例》

网上也可找到相关的参考资料或问题解答。

附录二、INT 21H 指令说明及使用方法

表附 1 DOS 系统功能调 INT 21H

AH	功能	调用参数	返回参数
0	程序终止(同 INT 20H)	CS=程序段前缀	
1	键盘输入并回显		AL=输入字符
2	显示输出	DL=输出字符	
3	异步通讯输入		AL=输入数据
4	异步通讯输出	DL=输出数据	
5	打印机输出	DL=输出字符	
6	直接控制台 I/O	DL=FF(输入) DL=字符(输出)	AL=输入字符
7	键盘输入(无回显)		AL=输入字符
8	键盘输入(无回显) 检测 Ctrl-Break		AL=输入字符
9	显示字符串	DS:DX=串地址 '\$'结束字符串	
0A	键盘输入到缓冲区	DS:DX=缓冲区首地址 (DS:DX)=缓冲区最大字符数	(DS:DX+1)=实际输入的字符数
0B	检验键盘状态		AL=00 有输入 AL=FF 无输入
0C	清除输入缓冲区并 请求指定的输入功能	AL=输入功能号 (1,6,7,8,A)	
0D	磁盘复位		清除文件缓冲区
0E	指定当前缺省的磁盘驱动器	DL=驱动器号 0=A,1=B,...	AL=驱动器数
0F	打开文件	DS:DX=FCB 首地址	AL=00 文件找到 AL=FF 文件未找到
10	关闭文件	DS:DX=FCB 首地址	AL=00 目录修改成功 AL=FF 目录中未找到文件

AH	功能	调用参数	返回参数
11	查找第一个目录项	DS:DX=FCB 首地址	AL=00 找到
			AL=FF 未找到
12	查找下一个目录项	DS:DX=FCB 首地址	AL=00 找到
		(文件中带有*或?)	AL=FF 未找到
13	删除文件	DS:DX=FCB 首地址	AL=00 删除成功
			AL=FF 未找到
14	顺序读	DS:DX=FCB 首地址	AL=00 读成功
			=01 文件结束,记录中无数据
			=02 DTA 空间不够
			=03 文件结束,记录不完整
15	顺序写	DS:DX=FCB 首地址	AL=00 写成功
			=01 盘满
			=02 DTA 空间不够
16	建文件	DS:DX=FCB 首地址	AL=00 建立成功
			=FF 无磁盘空间
17	文件改名	DS:DX=FCB 首地址	AL=00 成功
		(DS:DX+1)=旧文件名	AL=FF 未成功
		(DS:DX+17)=新文件名	
19	取当前缺省磁盘驱动器		AL=缺省的驱动器号 0=A,1=B,2=C,...
1A	置 DTA 地址	DS:DX=DTA 地址	
1B	取缺省驱动器 FAT 信息		AL=每簇的扇区数
			DS:BX=FAT 标识字节
			CX=物理扇区大小
			DX=缺省驱动器的簇数
1C	取任一驱动器 FAT 信息	DL=驱动器号	同上
21	随机读	DS:DX=FCB 首地址	AL=00 读成功
			=01 文件结束

AH	功能	调用参数	返回参数
			=02 缓冲区溢出
			=03 缓冲区不满
22	随机写	DS:DX=FCB 首地址	AL=00 写成功
			=01 盘满
			=02 缓冲区溢出
23	测定文件大小	DS:DX=FCB 首地址	AL=00 成功(文件长度填入FCB)
			AL=FF 未找到
24	设置随机记录号	DS:DX=FCB 首地址	
25	设置中断向量	DS:DX=中断向量	
		AL=中断类型号	
26	建立程序段前缀	DX=新的程序段前缀	
27	随机分块读	DS:DX=FCB 首地址	AL=00 读成功
		CX=记录数	=01 文件结束
			=02 缓冲区太小,传输结束
			=03 缓冲区不满
28	随机分块写	DS:DX=FCB 首地址	AL=00 写成功
		CX=记录数	=01 盘满
			=02 缓冲区溢出
29	分析文件名	ES:DI=FCB 首地址	AL=00 标准文件
		DS:SI=ASCIIZ 串	=01 多义文件
		AL=控制分析标志	=02 非法盘符
2A	取日期		CX=年
			DH:DL=月:日(二进制)
2B	设置日期	CX:DH:DL=年:月:日	AL=00 成功
			=FF 无效
2C	取时间		CH:CL=时:分
			DH:DL=秒:1/100 秒
2D	设置时间	CH:CL=时:分	AL=00 成功
		DH:DL=秒:1/100 秒	=FF 无效

AH	功能	调用参数	返回参数
2E	置磁盘自动读写标志	AL=00 关闭标志	
		AL=01 打开标志	
2F	取磁盘缓冲区的首址		ES:BX=缓冲区首址
30	取 DOS 版本号		AH=发行号,AL=版本
31	结束并驻留	AL=返回码	
		DX=驻留区大小	
33	Ctrl-Break 检测	AL=00 取状态	DL=00 关闭 Ctrl-Break 检测
		=01 置状态(DL)	=01 打开 Ctrl-Break 检测
		DL=00 关闭检测	
		=01 打开检测	
35	取中断向量	AL=中断类型	ES:BX=中断向量
36	取空闲磁盘空间	DL=驱动器号	成功:AX=每簇扇区数
		0=缺省,1=A,2=B,...	BX=有效簇数
			CX=每扇区字节数
			DX=总簇数
			失败:AX=FFFF
38	置/取国家信息	DS:DX=信息区首地址	BX=国家码(国际电话前缀码)
			AX=错误码
39	建立子目录(MKDIR)	DS:DX=ASCIIZ 串地址	AX=错误码
3A	删除子目录(RMDIR)	DS:DX=ASCIIZ 串地址	AX=错误码
3B	改变当前目录(CHDIR)	DS:DX=ASCIIZ 串地址	AX=错误码
3C	建立文件	DS:DX=ASCIIZ 串地址	成功:AX=文件代号
		CX=文件属性	错误:AX=错误码
3D	打开文件	DS:DX=ASCIIZ 串地址	成功:AX=文件代号
		AL=0 读	错误:AX=错误码
		=1 写	
		=3 读/写	

AH	功能	调用参数	返回参数
3E	关闭文件	BX=文件代号	失败:AX=错误码
3F	读文件或设备	DS:DX=数据缓冲区地址	读成功:
		BX=文件代号	AX=实际读入的字节数
		CX=读取的字节数	AX=0 已到文件尾
			读出错:AX=错误码
40	写文件或设备	DS:DX=数据缓冲区地址	写成功:
		BX=文件代号	AX=实际写入的字节数
		CX=写入的字节数	写出错:AX=错误码
41	删除文件	DS:DX=ASCIIZ 串地址	成功:AX=00
			出错:AX=错误码(2,5)
42	移动文件指针	BX=文件代号	成功:DX:AX=新文件指针位置
		CX:DX=位移量	出错:AX=错误码
		AL=移动方式(0:从文件头绝对位移,1:从当前位置相对移动,2:从文件尾绝对位移)	
43	置/取文件属性	DS:DX=ASCIIZ 串地址	成功:CX=文件属性
		AL=0 取文件属性	失败:CX=错误码
		AL=1 置文件属性	
		CX=文件属性	
44	设备文件 I/O 控制	BX=文件代号	DX=设备信息
		AL=0 取状态	
		=1 置状态 DX	
		=2 读数据	
		=3 写数据	
		=6 取输入状态	
		=7 取输出状态	

AH	功能	调用参数	返回参数
45	复制文件代号	BX=文件代号 1	成功:AX=文件代号 2
			失败:AX=错误码
46	人工复制文件代号	BX=文件代号 1	失败:AX=错误码
		CX=文件代号 2	
47	取当前目录路径名	DL=驱动器号	(DS:SI)=ASCIIZ 串
		DS:SI=ASCIIZ 串地址	失败:AX=出错码
48	分配内存空间	BX=申请内存容量	成功:AX=分配内存首地
			失败:BX=最大可用内存
49	释放内容空间	ES=内存起始段地址	失败:AX=错误码
4A	调整已分配的存储块	ES=原内存起始地址	失败:BX=最大可用空间
		BX=再申请的容量	AX=错误码
4B	装配/执行程序	DS:DX=ASCIIZ 串地址	失败:AX=错误码
		ES:BX=参数区首地址	
		AL=0 装入执行	
		AL=3 装入不执行	
4C	带返回码结束	AL=返回码	
4D	取返回代码		AX=返回代码
4E	查找第一个匹配文件	DS:DX=ASCIIZ 串地址	AX=出错代码(02,18)
		CX=属性	
4F	查找下一个匹配文件	DS:DX=ASCIIZ 串地址	AX=出错代码(18)
		(文件名中带有?或*)	
54	取盘自动读写标志		AL=当前标志值
56	文件改名	DS:DX=ASCIIZ 串(旧)	AX=出错码(03,05,17)
		ES:DI=ASCIIZ 串(新)	
57	置/取文件日期和时间	BX=文件代号	DX:CX=日期和时间
		AL=0 读取	失败:AX=错误码
		AL=1 设置(DX:CX)	
58	取/置分配策略码	AL=0 取码	成功:AX=策略码

AH	功能	调用参数	返回参数
		AL=1 置码(BX)	失败:AX=错误码
59	取扩充错误码		AX=扩充错误码
			BH=错误类型
			BL=建议的操作
			CH=错误场所
5A	建立临时文件	CX=文件属性	成功:AX=文件代号
		DS:DX=ASCIIZ 串地址	失败:AX=错误码
5B	建立新文件	CX=文件属性	成功:AX=文件代号
		DS:DX=ASCIIZ 串地址	失败:AX=错误码
5C	控制文件存取	AL=00 封锁	失败:AX=错误码
		=01 开启	
		BX=文件代号	
		CX:DX=文件位移	
		SI:DI=文件长度	
62	取程序段前缀		BX=PSP 地址

附录三、I386 汇编指令集

一、数据传输指令

它们在存储器、寄存器、寄存器和输入输出端口之间传送数据。

1. 通用数据传送指令。

1) MOV DST, SRC

其中：DST：目的操作数可以是存储器、通用寄存器、段寄存器（CS 除外）操作数，不能是操作数

SRC：源操作数可以是立即数、存储器、通用寄存器、段寄存器（包括 CS）操作数

执行操作：DST ← SRC

指令功能：把一个字节或字或双字从源操作数送目的操作数

2) MOVSX 先符号扩展,再传送。

3) MOVZX 先零扩展,再传送。

4) PUSH SRC

其中：SRC：源操作数可以是 16 位或 32 位的寄存器、存储器、段寄存器、立即数

功能：PUSH 指令可以将通用寄存器，段寄存器、存储器、立即数中的一个字或双字压进栈顶

执行操作：

16 位指令：SP ← SP - 2

[SP+1, SP] ← SRC

32 位指令：ESP ← ESP - 4

[ESP+3, ESP+2, ESP+1, ESP] ← SRC

5) POP DST

其中：DST：目的操作数必须是 16 位或 32 位寄存器、存储器、段寄存器（除 CS 以外）

功能：将现行的 SP/ESP 所指栈顶的一个字/双字传到寄存器、存储器、段寄存器（除 CS 以外）

执行操作：

16 位指令：DST ← [SP+1, SP]

SP ← SP + 2

32 位指令：DST ← [ESP+3, ESP+2, ESP+1, ESP]

ESP ← ESP + 4

6) PUSHAD 把 AX, CX, DX, BX, SP, BP, SI, DI 依次压入堆栈。

7) POPAD 把 DI, SI, BP, SP, BX, DX, CX, AX 依次弹出堆栈。

8) PUSHAD 把 EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI 依次压入堆栈。

9) POPAD 把 EDI, ESI, EBP, ESP, EBX, EDX, ECX, EAX 依次弹出堆栈。

10) BSWAP 交换 32 位寄存器里字节的顺序

11) XCHG DST,SCR

功能：把一个字或一个字节的源操作数与目的操作数进行交换

执行操作：DST<-->SCR

注意问题：1、只能通用寄存器与通用寄存器互换

2、只能通用寄存器与存储器互换

3、段寄存器不能作为 XCHG 的操作数

4、该指令不影响标志位

5、交换指令的约束与 MOV 相同（除立即数寻址方式）

6、386 以上机型允许双字操作。

12) CMPXCHG 比较并交换操作数.(第二个操作数必须为累加器 AL/AX/EAX)

13) XADD 先交换再累加.(结果在第一个操作数里)

14) XLAT SCR

其中：源操作数为被查找表的首地址或省略（功能相同）

在执行 XLAT 之前，首先要将字节表的首地址（一般是符号地址）送入 BX/EBX 并将表的相对值（要读出来的字节序号）送入 AL

功能：在换码表中找到一个值（字节），送 AL；这个值的地址为 BX+AL。BX 中是换码表的首地址，AL 中是所找到的值相对于码表首地址的相对值

执行操作：16 位指令：AL<--[BX+AL]

32 位指令：AL<--[EBX+AL]

需要注意的问题：

(1) 16 位指令在执行 XLAT 之前，要将字节表的首地址送入 BX

(2) 32 位指令在执行 XLAT 之前，要将字节表的首地址送入 EBX

(3) 该指令不影响标志位

(4) 由于 AL 寄存器是 8 位寄存器，所示表的长度不能超过 256 个字节

2. 输入输出端口传送指令.

1) IN AL/AX/EAX/,addr

其中：目的操作数只能是累加器 AL 或 AX 或 EAX 操作数，addr 为源操作数的地址，其寻址方式包括直接端口寻址方式和间接端口寻址方式。

指令功能：将指定的 I/O 端口中的内容输入到累加器 AL/AX/EAX 中

I/O 端口输入.(语法: IN 累加器, {端口号 | DX})

输入输出端口由立即方式指定时，其范围是 0-255; 由寄存器 DX 指定时，其范围是 0-65535.

2) OUT addr, AL/AX/EAX

其中：源操作数只能是累加器 AL 或 AX 或 EAX 操作数，Addr 为目的操作数的端口地址

指令功能：将累加器 AL 或 AX 或 EAX 中的内容输出到指定的 I/O 端口中

I/O 端口输出。(语法: OUT {端口号 | DX},累加器)

输入输出端口由立即方式指定时，其范围是 0-255; 由寄存器 DX 指定时，其范围是 0-65535.

3. 目的地址传送指令.

1) LEA reg, mem

其中：源操作数是存储器（mem）操作数，目的操作数（reg）必须是 16 位/32 位通用寄存器。

指令功能：将源操作数的有效地址送到目的通用寄存器（reg）中

执行操作：mem-->reg

注意问题：

- (1) 源操作数必须是存储器操作数，目的操作数必须是通用寄存器
- (2) 不影响状态标志位
- (3) 当时 32 位操作数时，目的操作数使用的寄存器可以是 32 位通用寄存器

例: LEA DX,string ;把偏移地址存到 DX.

2) LDS (/LES/LFS/LGS/LSS) reg, mem

其中：源操作数为存储器 mem 操作数，具有双字（32 位）特性的存储地址。目的操作数 reg 为 16/32 位通用寄存器

指令功能：将存储器 mem 中的双字内容依次送入通用寄存器 reg 和段寄存器 DS（或 ES 或 FS 或 GS 或 SS）中。

执行操作：mem 底字-->reg,mem 高字-->DS(或 ES 或 FS 或 GS 或 SS)

需要注意问题：

- (1) 操作数必须是存储器操作数，目的操作数必须是通用寄存器操作数
- (2) 若寄存器 reg 是 16 位通用寄存器，则存储器 mem 必须是 32 位地址属性；若寄存器 reg 是 32 位通用寄存器，则存储器 mem 必须是 48 位地址属性
- (3) 不影响状态标志位

LDS 传送目标指针,把指针内容装入 DS.

例: LDS SI,string ;把段地址:偏移地址存到 DS:SI.

LES 传送目标指针,把指针内容装入 ES.

例: LES DI,string ;把段地址:偏移地址存到 ES:DI.

LFS 传送目标指针,把指针内容装入 FS.

例: LFS DI,string ;把段地址:偏移地址存到 FS:DI.

LGS 传送目标指针,把指针内容装入 GS.

例: LGS DI,string ;把段地址:偏移地址存到 GS:DI.

LSS 传送目标指针,把指针内容装入 SS.

例: LSS DI,string ;把段地址:偏移地址存到 SS:DI.

4. 标志传送指令.

LAHF 标志寄存器传送,把标志装入 AH.

SAHF 标志寄存器传送,把 AH 内容装入标志寄存器.

PUSHF 标志入栈.

POPF 标志出栈.

PUSHD 32 位标志入栈.

POPD 32 位标志出栈.

二、算术运算指令

ADD 加法.

ADC 带进位加法.

INC 加 1.

AAA 加法的 ASCII 码调整.

DAA 加法的十进制调整.

SUB 减法.

SBB 带借位减法.

DEC 减 1.

NEG 求反(以 0 减之).

CMP 比较.(两操作数作减法,仅修改标志位,不回送结果).

AAS 减法的 ASCII 码调整.

DAS 减法的十进制调整.

MUL 无符号乘法.

IMUL 整数乘法.

以上两条,结果回送 AH 和 AL(字节运算),或 DX 和 AX(字运算),

AAM 乘法的 ASCII 码调整.

DIV 无符号除法.

IDIV 整数除法.

以上两条,结果回送:

商回送 AL,余数回送 AH, (字节运算);

或 商回送 AX,余数回送 DX, (字运算).

AAD 除法的 ASCII 码调整.

CBW 字节转换为字.(把 AL 中字节的符号扩展到 AH 中去)

CWD 字转换为双字.(把 AX 中的字的符号扩展到 DX 中去)

CWDE 字转换为双字.(把 AX 中的字符符号扩展到 EAX 中去)

CDQ 双字扩展.(把 EAX 中的字的符号扩展到 EDX 中去)

三、逻辑运算指令

AND 与运算.

OR 或运算.

XOR 异或运算.

NOT 取反.

TEST 测试.(两操作数作与运算,仅修改标志位,不回送结果).

SHL 逻辑左移.

SAL 算术左移.(=SHL)

SHR 逻辑右移.

SAR 算术右移.(=SHR)

ROL 循环左移.

ROR 循环右移.

RCL 通过进位的循环左移.

RCR 通过进位的循环右移.

以上八种移位指令,其移位次数可达 255 次.

移位一次时,可直接用操作码.如 SHL AX,1.

移位>1 次时,则由寄存器 CL 给出移位次数.

如 MOV CL,04

SHL AX,CL

四、串指令

串操作指令是无操作数指令,其隐含的操作数是内存操作数,源操作数在 DS:[SI]中;目的操作数在 ES:[DI]中

(当地址长度为 32 位时操作数地址应在 ESI 和 EDI 中),SI 和 DI 是变址寄存器间接寻址。重复前缀只能用于串操作指令之前。串操作指令使用前要求:

(1) 源操作数设置在 DS:[SI]中,目的操作数设置在 ES:[DI]中。SI 和 DI 存放的是源串和目的串的偏移地址

(2) 设置方向标志 DF,用指令 CLD 是 DF=0,为地址自动增量,或用指令 STD 是 DF=1,为地址自动减量

(3) 当使用重复前缀时,重复次数送 CX。

1) 串传送

MOVSB 字节操作: ES:[DI]<--DS:[SI]; SI+/-1-->SI, DI+/-1-->DI REP

MOVSW 字操作: ES:[DI]<--DS:[SI]; SI+/-2-->SI, DI+/-2-->DI

MOVSD 双字操作: ES:[DI]<--DS:[SI]; SI+/-4-->SI, DI+/-4-->DI

2) 串存储

STOSB 字节操作: ES:[DI]<--AL; DI+/-1-->DI REP

STOSW 字操作: ES:[DI]<--AX; DI+/-2-->DI

STOSD 双字操作: ES:[DI]<--EAX; DI+/-4-->DI

3) 串装入

LODSB 字节操作: AL<--DS:[SI], SI+/-1-->SI REP

LODSW 字操作: AL<--DS:[SI], SI+/-2-->SI

LODSD 双字操作: AL<--DS:[SI], SI+/-4-->SI

4) 串比较

CMPSB 字节操作: DS:[SI]-ES:[DI], SI+/-1-->SI, DI+/-1-->DI REPE/REPZ

CMPSW 字操作: DS:[SI]-ES:[DI], SI+/-2-->SI, DI+/-2-->DI

CMPSD 双字操作: DS:[SI]-ES:[DI], SI+/-4-->SI, DI+/-4-->DI

5) 串搜索

SCASB 字节操作: AL(或 AX 或 EAX)-ES:[DI]; DI+/-1-->DI REPE/REPZ

SCASW 字操作: AX-ES:[DI]; DI+/-2-->DI

SCASD 双字操作: EAX-ES:[DI]; DI+/-4-->DI

6) 串输入

INSB 字节操作: [DX]<--DST; DI+/-1-->SI REP

INSW 字操作: [DX]<--DST; DI+/-2-->SI

INSD 双字操作: [DX]<--DST; DI+/-4-->SI

7) 串输出

OUTSB 字节操作: SRC-->[DX]; SI+/-1-->SI REP

OUTSW 字操作: SRC-->[DX]; SI+/-2-->SI

OUTSD 串输出操作: SRC-->[DX]; SI+/-4-->SI

8) 重复

REP CX!=0 则执行串操作指令 CX-1-->CX

9)相等/为零则重复

REPE/REPZ CX!=0 且 ZF=1 则执行串操作指令 CX-1-->CX

10)不相等/不为零则重复

REPNE/REPNZ CX!=0 且 ZF=0 则执行串操作指令 CX-1-->CX

五、程序转移指令

1) 无条件转移指令 (长转移)

JMP 无条件转移指令

CALL 过程调用

RET/RETF 过程返回.

2) 条件转移指令 (短转移,-128 到+127 的距离内)

(当且仅当(SF XOR OF)=1 时,OP1<OP2)

JA/JNBE 不小于或不等于时转移.

JAE/JNB 大于或等于转移.

JB/JNAE 小于转移.

JBE/JNA 小于或等于转移.

以上四条,测试无符号整数运算的结果(标志 C 和 Z).

JG/JNLE 大于转移.

JGE/JNL 大于或等于转移.

JL/JNGE 小于转移.

JLE/JNG 小于或等于转移.

以上四条,测试带符号整数运算的结果(标志 S,O 和 Z).

JE/JZ 等于转移.

JNE/JNZ 不等于时转移.

JC 有进位时转移.

JNC 无进位时转移.

JNO 不溢出时转移.

JNP/JPO 奇偶性为奇数时转移.

JNS 符号位为 "0" 时转移.

JO 溢出转移.

JP/JPE 奇偶性为偶数时转移.

JS 符号位为 "1" 时转移.

3) 循环控制指令(短转移)

LOOP CX 不为零时循环.

LOOPE/LOOPZ CX 不为零且标志 Z=1 时循环.

LOOPNE/LOOPNZ CX 不为零且标志 Z=0 时循环.

JCXZ CX 为零时转移.

JECXZ ECX 为零时转移.

4) 中断指令

INT 中断指令

INTO 溢出中断

IRET 中断返回

5) 处理器控制指令

HLT 处理器暂停, 直到出现中断或复位信号才继续.

WAIT 当芯片引线 TEST 为高电平时使 CPU 进入等待状态.

ESC 转换到外处理器.

LOCK 封锁总线.

NOP 空操作.

STC 置进位标志位.

CLC 清进位标志位.

CMC 进位标志取反.

STD 置方向标志位.

CLD 清方向标志位.

STI 置中断允许位.

CLI 清中断允许位.

六、伪指令

DW 定义字(2 字节).

PROC 定义过程.

ENDP 过程结束.

完整段定义伪指令(SEGMENT、ENDS、ASSUME)

段定义伪指令(SEGMENT、ENDS)

格式:

段名 SEGMENT [对齐类型] [组合类型] [使用类型] ['类别名']

.....

段名 ENDS

功能：定义逻辑段，SEGMENT 表示某个逻辑段开始，ENDS 表示逻辑段结束

其中：4 个选项是可选，有可选项时各项顺序不能错，可选项之间用空格隔开。

(1)对齐类型

表示当前段的起始边界要求。即定义了当前段在内存中起始边界的设定，说明了段与段之间的空隙。对齐类型可以是：PAGE(页)、PARA(节)、WORD(字)、BYTE(字节)、DWORD(双字)

(2)组合类型

共有六种组合类型：

①缺省

没有说明，汇编程序就认为本段不和别的段连接。

②PUBLIC

本段与同名段顺序连接。组成一个大的逻辑段，它们共用同一个段起始地址。

③COMMON

本段与同名段同一地址开始重叠连接。段长是同名段中最长的段的长度

④STACK

表示该段是堆栈段的一部分。把所有相同‘类型名’的具有 STACK 组合类型的段连接成一个连续段。将连续段首地址送 SS，段内最大偏移地址送 SP。

在主程序中可省略对 SS 和 SP 的初始化。

⑤MEMORY

表示在 N 个互相连接的段中本段的定位地址为最高地址，如果有多个 MEMORY 的段，则把第一个遇到的段当作 MEMORY 处理，其他均当作 COMMON(重叠处理)

⑥AT

本段定位在表达式所表示的位置并且节对齐

如：S1 SEGMENT PARA AT 0A800H

表示本段段地址为 0A800H 并且节对齐

(3)使用类型

使用类型包括：USE16 和 USE32。用来说明使用 16 位寻址方式还是 32 位寻址方式。

(4)'类别名'

在引号中给出相连接的段名，完成把具有相同类名的段连接在一起。

END 程序结束.

七、位测试指令

1、位测试指令（BT）

BT DST, SRC

指令功能：把目的操作数 DST 中由源操作数 SRC 所指定的值送标志寄存器 CF 位

执行操作：CF←DST 的 SRC 位

2、测试并置 1 指令（BTS）

BTS DST, SRC

指令功能：把目的操作数 DST 中由源操作数 SRC 所指定位的值送标志寄存器 CF 位，并将目的操作数 DST 中的该位置 1

执行操作：CF←DST 的 SRC 位

DST 的 SRC 位←1

3、位测试并置 0 指令（BTR）

BTR DST, SRC

指令功能：把目的操作数 DST 中由源操作数 SRC 所指定位的值送标志寄存器 CF 位，并将目的操作数中的该位置 0

执行操作：CF←DST 的 SRC 位

DST 的 SRC 位←0

4、位测试并变反指令（BTC）

BTC DST, SRC

指令功能：把目的操作数 DST 中由源操作数 SRC 所指定位的值送标志寄存器 CF 位，并将目的操作数 DST 中的该位取反

执行操作：CF←DST 的 SRC 位

DST 的 SRC 位←取反

需要注意的问题：

（1）指令中的目的操作数 DST 为 16/32 位寄存器/存储器操作数，源操作数 SRC 可以使 16/32 位寄存器/立即数操作数（0~31）

（2）指令中目的操作数 DST 和源操作数 SRC 类型相同

（3）本组指令只影响标志寄存器的 CF 位，其他标志位则无定义

附录三、74 系列芯片功能列表

型号	功能
7400	2 输入端四与非门
7401	集电极开路 2 输入端四与非门
7402	2 输入端四或非门
7403	集电极开路 2 输入端四与非门
7404	六反相器
7405	集电极开路六反相器
7406	集电极开路六反相高压驱动器
7407	集电极开路六正相高压驱动器
7408	2 输入端四与门
7409	集电极开路 2 输入端四与门
7410	3 输入端 3 与非门
74107	带清除主从双 J-K 触发器
74109	带预置清除正触发双 J-K 触发器
7411	3 输入端 3 与门
74112	带预置清除负触发双 J-K 触发器
7412	开路输出 3 输入端三与非门
74121	单稳态多谐振荡器
74122	可再触发单稳态多谐振荡器
74123	双可再触发单稳态多谐振荡器
74125	三态输出高有效四总线缓冲门
74126	三态输出低有效四总线缓冲门
7413	4 输入端双与非施密特触发器
74132	2 输入端四与非施密特触发器
74133	13 输入端与非门
74136	四异或门
74138	3-8 线译码器/复工器
74139	双 2-4 线译码器/复工器
7414	六反相施密特触发器
74145	BCD—十进制译码/驱动器
7415	开路输出 3 输入端三与门
74150	16 选 1 数据选择/多路开关
74151	8 选 1 数据选择器
74153	双 4 选 1 数据选择器
74154	4 线—16 线译码器
74155	图腾柱输出译码器/分配器
74156	开路输出译码器/分配器
74157	同相输出四 2 选 1 数据选择器
74158	反相输出四 2 选 1 数据选择器
7416	开路输出六反相缓冲/驱动器

型号	功能
74160	可预置 BCD 异步清除计数器
74161	可预置四位二进制异步清除计数器
74162	可预置 BCD 同步清除计数器
74163	可预置四位二进制同步清除计数器
74164	八位串行入/并行输出移位寄存器
74165	八位并行入/串行输出移位寄存器
74166	八位并入/串出移位寄存器
74169	二进制四位加/减同步计数器
7417	开路输出六同相缓冲/驱动器
74170	开路输出 4×4 寄存器堆
74173	三态输出四位 D 型寄存器
74174	带公共时钟和复位六 D 触发器
74175	带公共时钟和复位四 D 触发器
74180	9 位奇数/偶数发生器/校验器
74181	算术逻辑单元/函数发生器
74185	二进制—BCD 代码转换器
74190	BCD 同步加/减计数器
74191	二进制同步可逆计数器
74192	可预置 BCD 双时钟可逆计数器
74193	可预置四位二进制双时钟可逆计数器
74194	四位双向通用移位寄存器
74195	四位并行通道移位寄存器
74196	十进制/二—十进制可预置计数锁存器
74197	二进制可预置锁存器/计数器
7420	4 输入端双与非门
7421	4 输入端双与门
7422	开路输出 4 输入端双与非门
74221	双/单稳态多谐振荡器
74240	八反相三态缓冲器/线驱动器
74241	八同相三态缓冲器/线驱动器
74243	四同相三态总线收发器
74244	八同相三态缓冲器/线驱动器
74245	八同相三态总线收发器
74247	BCD—7 段 15V 输出译码/驱动器
74248	BCD—7 段译码/升压输出驱动器
74249	BCD—7 段译码/开路输出驱动器
74251	三态输出 8 选 1 数据选择器/复工器
74253	三态输出双 4 选 1 数据选择器/复工器
74256	双四位可寻址锁存器
74257	三态原码四 2 选 1 数据选择器/复工器
74258	三态反码四 2 选 1 数据选择器/复工器

型号	功能
74259	八位可寻址锁存器/3-8 线译码器
7426	2 输入端高压接口四与非门
74260	5 输入端双或非门
74266	2 输入端四异或非门
7427	3 输入端三或非门
74273	带公共时钟复位八 D 驱动器
74279	四图腾柱输出 S-R 锁存器
7428	2 输入端四或非门缓冲器
74283	4 位二进制全加器
74290	二/五分频十进制计数器
74293	二/八分频四位二进制计数器
74295	四位双向通用移位寄存器
74298	四 2 输入多路带存贮开关
74299	三态输出八位通用移位寄存器
7430	8 输入端与非门
7432	2 输入端四或门
74322	带符号扩展端八位移位寄存器
74323	三态输出八位双向移位/存贮寄存器
7433	开路输出 2 输入端四或非缓冲器
74347	BCD—7 段译码器/驱动器
74352	双 4 选 1 数据选择器/复工器
74353	三态输出双 4 选 1 数据选择器/复工器
74365	门使能输入三态输出六同相线驱动器
74365	门使能输入三态输出六同相线驱动器
74366	门使能输入三态输出六反相线驱动器
74367	4/2 线使能输入三态六同相线驱动器
74368	4/2 线使能输入三态六反相线驱动器
7437	开路输出 2 输入端四与非缓冲器
74373	三态同相八 D 锁存器
74374	三态反相八 D 锁存器
74375	4 位双稳态锁存器
74377	单边输出公共使能八 D 锁存器
74378	单边输出公共使能六 D 锁存器
74379	双边输出公共使能四 D 锁存器
7438	开路输出 2 输入端四与非缓冲器
74380	多功能八进制寄存器
7439	开路输出 2 输入端四与非缓冲器
74390	双十进制计数器
74393	双四位二进制计数器
7440	4 输入端双与非缓冲器
7442	BCD—十进制代码转换器

型号	功能
74352	双 4 选 1 数据选择器/复工器
74353	三态输出双 4 选 1 数据选择器/复工器
74365	门使能输入三态输出六同相线驱动器
74366	门使能输入三态输出六反相线驱动器
74367	4/2 线使能输入三态六同相线驱动器
74368	4/2 线使能输入三态六反相线驱动器
7437	开路输出 2 输入端四与非缓冲器
74373	三态同相八 D 锁存器
74374	三态反相八 D 锁存器
74375	4 位双稳态锁存器
74377	单边输出公共使能八 D 锁存器
74378	单边输出公共使能六 D 锁存器
74379	双边输出公共使能四 D 锁存器
7438	开路输出 2 输入端四与非缓冲器
74380	多功能八进制寄存器
7439	开路输出 2 输入端四与非缓冲器
74390	双十进制计数器
74393	双四位二进制计数器
7440	4 输入端双与非缓冲器
7442	BCD—十进制代码转换器
74447	BCD—7 段译码器/驱动器
7445	BCD—十进制代码转换/驱动器
74450	16:1 多路转接复用器多工器
74451	双 8:1 多路转接复用器多工器
74453	四 4:1 多路转接复用器多工器
7446	BCD—7 段低有效译码/驱动器
74460	十位比较器
74461	八进制计数器
74465	三态同相 2 与使能端八总线缓冲器
74466	三态反相 2 与使能端八总线缓冲器
74467	三态同相 2 使能端八总线缓冲器
74468	三态反相 2 使能端八总线缓冲器
74469	八位双向计数器
7447	BCD—7 段高有效译码/驱动器
7448	BCD—7 段译码器/内部上拉输出驱动
74490	双十进制计数器
74491	十位计数器
74498	八进制移位寄存器
7450	2-3/2-2 输入端双与或非门
74502	八位逐次逼近寄存器
74503	八位逐次逼近寄存器

型号	功能
7451	2-3/2-2 输入端双与或非门
74533	三态反相八 D 锁存器
74534	三态反相八 D 锁存器
7454	四路输入与或非门
74540	八位三态反相输出总线缓冲器
7455	4 输入端二路输入与或非门
74563	八位三态反相输出触发器
74564	八位三态反相输出 D 触发器
74573	八位三态输出触发器
74574	八位三态输出 D 触发器
74645	三态输出八同相总线传送接收器
74670	三态输出 4×4 寄存器堆
7473	带清除负触发双 J-K 触发器
7474	带置位复位正触发双 D 触发器
7476	带预置清除双 J-K 触发器
7483	四位二进制快速进位全加器
7485	四位数字比较器
7486	2 输入端四异或门
7490	可二/五分频十进制计数器
7493	可二/八分频二进制计数器
7495	四位并行输入\输出移位寄存器
7497	6 位同步二进制乘法器
74LS00	四 2 输入与非门
74LS02	四 2 输入与非门
74LS04	六反相器
74LS06	六反相缓冲器/驱动器
74LS08	四 2 输入与非门
74LS10	三 3 输入与非门
74LS12	三 3 输入与非门
74LS14	六反相器. 斯密特触发
74LS16	六反相缓冲器/触发器
74LS20	双 4 输入与门
74LS22	双 4 输入与门
74LS26	四 2 输入与非门
74LS28	四输入端或非缓冲器
74LS32	四 2 输入或门
74LS37	四输入端与非缓冲器
74LS40	四输入端与非缓冲器
74LS47	BCD—七段译码驱动器
74LS49	BCD—七段译码驱动器
74LS54	四输入与或非门

型号	功能
74LS63	六电流读出接口门
74LS74	双 D 触发器
74LS76	双 J—K 触发器
74LS83	双 J—K 触发器
74LS86	四 2 输入异或门
74LS90	4 位十进制波动计数器
74LS92	12 分频计数器
74LS96	5 位移位寄存器
74LS109	正沿触发双 J—K 触发器
74LS113	双 J—K 负沿触发器
74LS121	单稳态多谐振荡器
74LS123	双稳态多谐振荡器
74LS125	三态缓冲器
74LS131	3—8 线译码器
74LS133	13 输入与非门
74LS137	地址锁存 3—8 线译码器
74LS139	双 2—4 线译码—转换器
74LS147	10—4 线优先编码器
74LS153	双 4 选 1 数据选择器
74LS155	双 2—4 线多路分配器
74LS157	四 2 选 1 数据选择器
74LS160	同步 BDC 十进制计数器
74LS162	同步 BDC 十进制计数器
74LS164	8 位串入并出移位寄存
74LS166	8 位移位寄存器
74LS169	4 位可逆同步计数器
74LS172	16 位多通道寄存器堆
74LS174	6D 型触发器
74LS176	可预置十进制计数器
74LS182	超前进位发生器
74LS189	64 位随机存储器
74LS191	二进制同步可逆计数器
74LS193	二进制可逆计数器
74LS195	并行存取移位寄存器
74LS197	可预置二进制计数器
74LS238	3—8 线译码/多路转换器
74LS241	八缓冲/驱动/接收器
74LS243	四总线收发器
74LS245	八总线收发器
74LS248	BCD—七段译码驱动器
74LS251	三态 8—1 数据选择器

型号	功能
74LS256	双四位选址锁存器
74LS258	四 2 选 1 数据选择器
74LS260	双 5 输入或非门
74LS266	四 2 输入异或非门
74LS275	七位树型乘法器
74LS279	四 R—S 触发器
74LS283	4 位二进制全加器
74LS293	4 位二进制计数器
74LS365	六缓冲器带公用启动器
74LS367	六总线三态输出缓冲器
74LS373	8D 锁存器
74LS375	4 位双稳锁存器
74LS386	四 2 输入异或门
74LS393	双 4 位二进制计数器
74LS574	8 位 D 型触发器
74LS684	8 位数字比较器
74LS01	四 2 输入与非门
74LS03	四 2 输入与非门
74LS05	六反相器
74LS07	六缓冲器/驱动器
74LS09	四 2 输入与非门
74LS11	三 3 输入与非门
74LS13	三 3 输入与非门
74LS15	三 3 输入与非门
74LS17	六反相缓冲器/驱动器
74LS21	双 4 输入与门
74LS25	双 4 输入与门
74LS27	三 3 输入与非门
74LS30	八输入端与非门
74LS33	四 2 输入或门
74LS38	双 2 输入与非缓冲器
74LS42	BCD—十进制译码器
74LS48	BCD—七段译码驱动器
74LS51	三 3 输入双与或非门
74LS55	四 4 输入与或非门
74LS73	双 J—K 触发器
74LS75	4 位双稳锁存器
74LS78	双 J—K 触发器
74LS85	4 位幅度比较器
74LS88	4 位全加器
74LS91	8 位移位寄存器

型号	功能
74LS93	二进制计数器
74LS95	4 位并入并出寄存器
74LS107	双 J—K 触发器
74LS112	双 J—K 负沿触发器
74LS114	双 J—K 负沿触发器
74LS122	单稳态多谐振荡器
74LS124	双压控振荡器
74LS126	四 3 态总线缓冲器
74LS132	二输入与非触发器
74LS136	四异或门
74LS138	3—8 线译码/转换器
74LS145	BCD 十进制译码/驱动器
74LS148	8—3 线优先编码器
74LS151	8 选 1 数据选择器
74LS154	4—16 线多路分配器
74LS156	双 2—4 线多路分配器
74LS158	四 2 选 1 数据选择器
74LS161	4 位二进制计数器
74LS163	4 位二进制计数器
74LS165	8 位移位寄存器
74LS168	4 位可逆同步计数器
74LS170	4x4 位寄存器堆
74LS173	4D 型寄存器
74LS175	4D 烯触发器
74LS181	运算器/函数发生器
74LS183	双进位保存全价器
74LS190	同步 BCD 十进制计数器
74LS192	BCD—同步可逆计数器
74LS194	双向通用移位寄存器
74LS196	可预置十进制计数器
74LS221	双单稳态多谐振荡器
74LS240	八缓冲/驱动/接收器
74LS242	四总线收发器
74LS244	八缓冲/驱动/接收器
74LS247	BCD—七段译码驱动器
74LS249	BCD—七段译码驱动器
74LS253	双三态 4—1 数据选择器
74LS257	四 3 态 2—1 数据选择器
74LS259	8 位可寻址锁存器
74LS261	2x4 位二进制乘发器
74LS273	八进制 D 型触发器

型号	功能
74LS276	四 J—K 触发器
74LS280	9 位奇偶数发生校检器
74LS290	十进制计数器
74LS295	4 位双向通用移位寄存器
74LS366	六缓冲器带公用启动器
74LS368	六总线三态输出反相器
74LS374	8D 触发器
74LS377	8 位单输出 D 型触发器
74LS390	双十进制计数器
74LS573	8 位三态输出 D 型锁存器
74LS670	8 位数字比较器
74HC00	四 2 输入与非门
74HC02	四 2 输入或非门
74HC03	四 2 输入或非门
74HC04	六反相器
74HC05	六反相器
74HC08	四 2 输入与门
74HC10	三 3 输入与非门
74HC11	三 3 输入与门
74HC14	六反相器/斯密特触发器
74HC20	双四输入与门
74HC21	双四输入与非门
74HC27	三 3 输入与非门
74HC30	八输入端与非门
74HC32	四 2 输入或门
74HC42	BCD 十进制译码器
74HC73	双 J—K 触发器
74HC74	双 D 型触发器
74HC76	双 J—K 触发器
74HC86	四 2 输入异或门
74HC107	双 J—K 触发器
74HC137	二输入与非缓冲器
74HC123	双稳态多谐振荡器
74HC125	三态缓冲器
74HC126	四三态总线缓冲器
74HC132	二输入与非缓冲器
74HC137	二输入与非缓冲器
74HC138	3—8 线译码/解调器
74HC139	双 2—4 线译码/解调器
74HC148	8 选 1 数据选择器
74HC151	双 4 选 1 数据选择器

型号	功能
74HC154	4—16 线多路分配器
74HC157	四 2 选 1 数据选择器
74HC161	4 位二进制计数器
74HC163	4 位二进制计数器
74HC164	8 位串入并出移位寄存器
74HC165	8 位移位寄存器
74HC173	4D 型触发器
74HC174	6D 触发器
74HC175	4D 型触发器
74HC191	二进制同步可逆计数器
74HC221	双单稳态多谐振荡器
74HC238	3—8 线译码器
74HC240	八缓冲器
74HC244	八总线 3 态输出缓冲器
74HC245	八总线收发器
74HC251	三态 8—1 数据选择器
74HC259	8 位可寻址锁存器
74HC266	四 2 输入异或非门
74HC273	8D 型触发器
74HC367	六缓冲器/总线驱动器
74HC368	六缓冲器/总线驱动器
74HC373	8D 锁存器
74HC374	8D 触发器
74HC393	双 4 位二进制计数器
74HC541	8 位三态输出缓冲器
74HC573	8 位三态输出 D 型锁存器
74HC574	8D 型触发器
74HC595	8 位移位寄存器/锁存器
74HC4028	7 级二进制串行加数器
74HC4046	锁相环
74HC4050	六同相缓冲器
74HC4051	8 选 1 模拟开关
74HC4053	三 2 选 1 模拟开关
74HC4060	14 位计数/分频/振荡器
74HC4066	四双相模拟开关
74HC4078	3 输入端三或门
74HC4511	7 段锁存/译码驱动器
74HC4520	双二进制加法计数器
74F00	高速四 2 输入与非门
74F02	高速四 2 输入或非门
74F04	高速六反相器

型号	功能
74F08	高速四 2 输入与门
74F10	高速三 3 输入与门
74F14	高速六反相斯密特触发
74F32	高速四 2 输入或门
74F38	高速四 2 输入或门
74F74	高速双 D 型触发器
74F86	高速四 2 输入异或门
74F139	高速双 2—4 线译码/驱动器
74F151	高速双 2—4 线译码/驱动器
74F153	高速双 4 选 1 数据选择器
74F157	高速双 4 选 1 数据选择器
74F161	高速 6D 型触发器
74F174	高速 6D 型触发器
74F175	高速 4D 型触发器
74F244	高速八总线 3 态缓冲器
74F245	高速八总线收发器
74F373	高速 8D 锁存器