8086 实验 1 设备使用及简单示例程序验证

一．实验目的

1 熟悉软件编程环境和硬件设备资源
2 熟悉工程创建的步骤，其中各个文件作用
3 掌握软件菜单的使用，掌握程序的调试基本步骤。

二．实验内容

以课本例 4-1 为源文件创建工程
进行工程的调试，查看程序运行过程和结果

三．实验仪器

微机、WAVE6000 编程环境，实验箱

注意：

文件不要用中文名称，保存时不要用中文路径（目录），不要放在“桌面”上，源文件和工程要放在同一个文件夹下，文件名称和路径名称不要太长。

调试过程：

2 查看存储器菜单使用：窗口---数据窗口---MEMORY，观察或者修改 MEMORY 内容。
3 查看 CPU 寄存器:窗口---CPU 窗口，CPU 寄存器
4 单步执行：执行---单步执行(F8)，每执行一步，查看每条语句涉及到的寄存器和存储器内容的变化结果，是否是指令所要得到的结果，如不是，检查错误原因，修改。

ASSUME CS:CSEG , DS:DSEG , ES:ESEG

SUM DW 2 DUP(?) DUP (duplicate) 含义如下：
定义变量（数组）SUM，类型为字（DW），大小为2，内容不定（?）

编译器设置：编译器将用户定义的数据段分配在 0400H（0040H×16）开始的存储器中。
数据段段基址寄存器 DS(0040H)。
查看数据的方法：

窗口---数据窗口--- MEMORY ---0400 的位置就是你定义的数据段的数据：

物理地址=段基址（DS）×16+段内偏移

参考程序及流程图：
见课本例 4-1

8086 实验 2　简单程序设计--存储块移动

一．实验目的

1 熟悉 86 汇编语言程序结构

2 熟悉循环结构程序的编写，进一步熟悉指令系统寻址方式

3 熟悉编程环境和程序的调试

二．实验内容

将指定源地址和长度（100 字节）的存储块移动到目的地址。

三．实验仪器：微机、WAVE6000 软件，实验箱

注意:1 源数据块和目的数据块自己在数据段中定义。

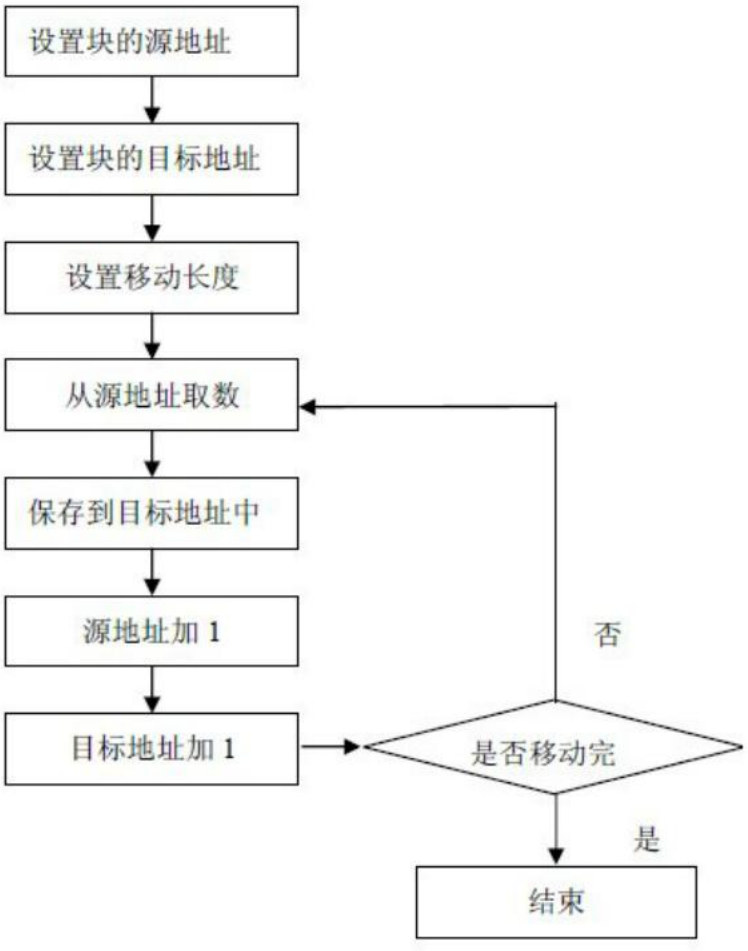
2 要查看移动后结果，定义时对源和目的块赋不同的值

S db 25 dup(11h)　含义?　dup (duplicate)

定义数组 S,大小为 25，类型为字节（DB），全部赋值为 11h

3 同实验一，数据段分配在 0400H 开始的存储器中

流程图如右：



两种移动的方式：

1 利用循环结构编程移动方式

2 利用重复指令（movsb）编程方式

参考程序

利用重复指令（movsb）编程方式

移动 256 字节

```
data    segment
Source db 256 dup(055h)
Target db 256 dup(0aah)
data    ends

code    segment
assume cs:code, ds:data, es:data
start:
        mov    ax, data
        mov    ds, ax
        mov    es, ax
        mov    si, offset Source
        mov    di, offset Target
        mov    cx, 256
        rep  movsb
        jmp  $
code    ends
        end start
```

8086 实验 3　存储块清零

一、实验要求

指定存储器中某块的起始地址和长度，要求能将其内容清零。

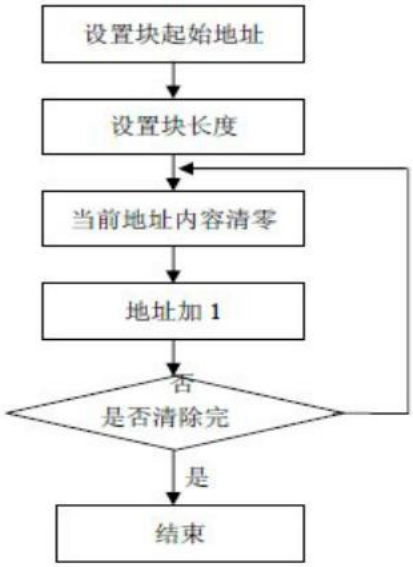
二、实验目的

1. 掌握存储器读写方法
2. 了解存储器的块操作方法

三、实验说明

通过本实验，学生可以了解单片机读写存储器的读写方法，同时也可以了解单片机编程，调试方法。如何将存储器块的内容置成某固定值(例全填充为 0FFH)？请学生修改程序，完成此操作。

四、程序框图




```
data    segment
Block   db 256 dup(55h)
data    ends

code    segment
        assume cs:code, ds:data

start   proc  near

        mov   ax, data
        mov   ds, ax

        mov   bx, offset Block  ; 起始地址
        mov   cx, 256           ; 清 256 字节
Again:
        mov   [bx], byte ptr 0
        inc   bx
        Loop  Again             ; 记数减一

        jmp  $

code    ends
        end  start
```

8086 实验 4　二进制到 BCD 码转换

一、实验要求

将给定的一个二进制数，转换成二十进制（BCD）码

二、实验目的

1. 掌握简单的数值转换算法
2. 基本了解数值的各种表达方法

三、实验说明

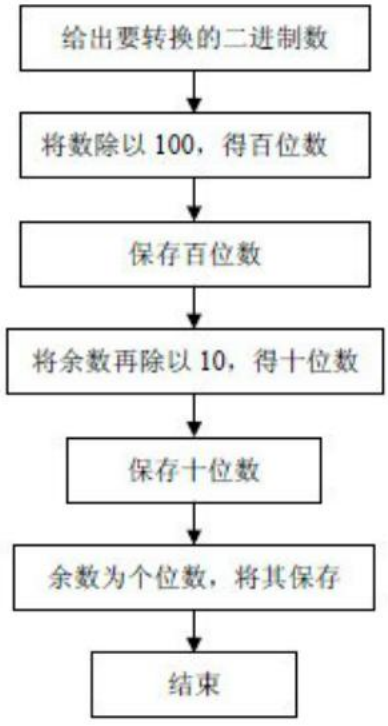
计算机中的数值有各种表达方式，这是计算机的基础。掌握各种数制之间的转换是一种基本功。有兴趣的同学可以试试将 BCD 转换成二进制码。

自己定义存放结果的数据空间（段）　百位　十位　个位

参考程序和框图如下：

; 将 AX 拆为三个 BCD 码，并存入 Result 开始的叁个单元　参考程序如下：

```
data    segment
Result  db  3  dup(?)          定义长度为 3 个字节的 Result    内容待定
data    ends

code    segment
        assume cs:code, ds:data
start:
    mov    ax, data
    mov    ds, ax          ;  初始化 DS

    mov    ax, 123
    mov    cl, 100
    div    cl
    mov    Result, al      ; 除以 100，得百位数

    mov    al, ah
    mov    ah, 0
    mov    cl, 10
    div    cl
    mov    Result+1, al    ; 余数除以 10，得十位数
    mov    Result+2, ah    ; 余数为个位数

    jmp    $

code    ends
  end   start
```



8086 实验 5　数据排序

一．实验目的

1 了解数据排序的常用算法，掌握冒泡算法。
2 进一步熟悉 86 编程环境和调试方法。
3 熟悉汇编程序设计，指令系统，伪指令的使用。

二．实验内容

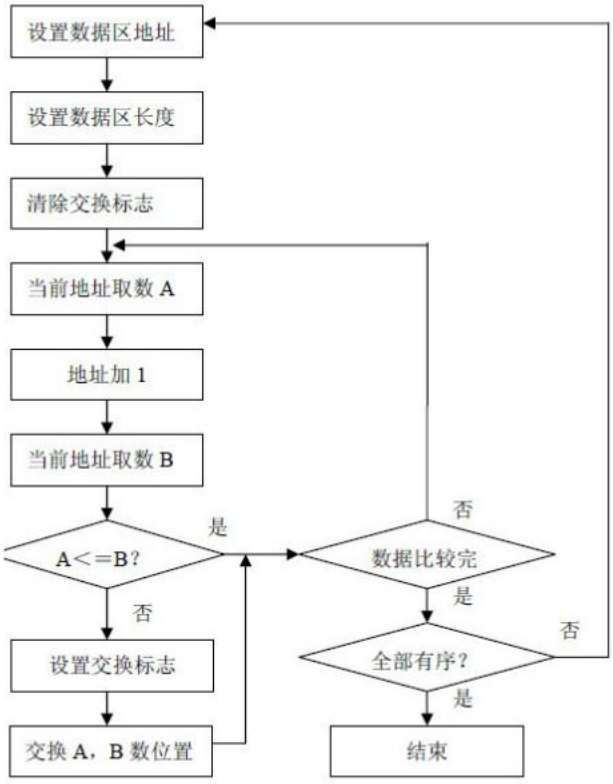
给出一组随机数（10 个，类型：字节），将此组数据数按从小到大（升序）排列（采用冒泡算法）。

三．实验仪器　微机、WAVE6000 编程环境

注意：在数据段中自定义一组随机数，对自定义的数据排序。
如自定义：　ARRAY　DB　6,2,4,9,3,7,1,5,8,0
冒泡算法（两层循环，以下是内层循环开始……）。
6，2，9，4，3，7，1，5，8，0
前大后小交换位置如下：
2，6，9，4，3，7，1，5，8，0
前小后大不交换位置如下;
2，6，9，4，3，7，1，5，8，0
前大后小交换位置
2，6，4，9，3，7，1，5，8，0　……
第一次内层循环结束顺序如下：最大的数 9 先冒出来
2，6，4，3，7，1，5，8，0，9 接着开始第二次内层循环……

参考程序：

```
Len      equ 10
data     segment
Array    db 5,2,1,0,2,3,8,6,5,9
Change db 0
data     ends
code     segment
         assume cs:code, ds:data
start    proc   near
         mov    ax, data
         mov    ds, ax
Sort:
         mov    bx, offset Array
         mov    cx, Len-1
         mov    Change, 0
Goon:
         mov    al, byte ptr [bx]
         inc    bx
         cmp    al, byte ptr [bx]
         jng    Next          ; 前小后大, 不交换
         mov    Change, 1     ; 前大后小, 置交换标志
         mov    ah, [bx]
         mov    [bx] ,al      ; 交换
         mov    [bx-1], ah
Next:
         loop   Goon
         cmp    Change, 0
         jne    Sort
         jmp    $
code     ends
         end start
```



8086 实验 6　A/D 转换实验

一　实验目的

1 掌握 ADC 与 8086 连接方法

2 了解 ADC0809 转换性能及编程

3 通过实验了解数据采集过程

二　实验内容

利用 0809 做 A/D 转换器，实验箱上电位器为其提供电压模拟量输入，编程将模拟量转换成数字量（8 位二进制），用 8255 的 PA 口输出到八个发光二级管显示。

注意：采用查询方式读入 A/D 转换结果。

ADC0809 的转换时间约 100uS（100 微秒）,所以每次转换完需要延时 100uS（采用软件延时程序）。

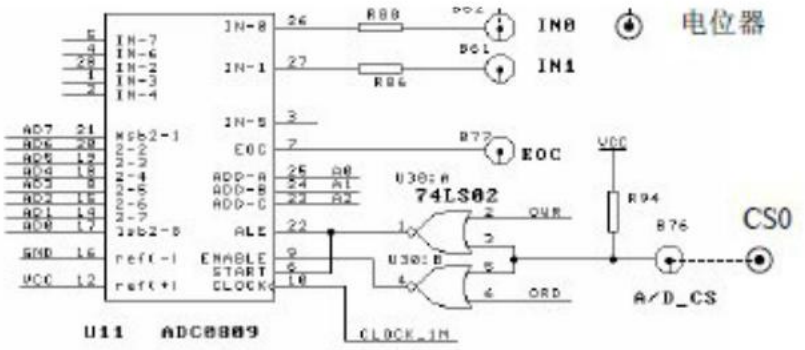
地址：8255，ADC0809 地址根据片选信号连线查表可得。

三　实验连线如下图表中所示：由下图实验连线可知：

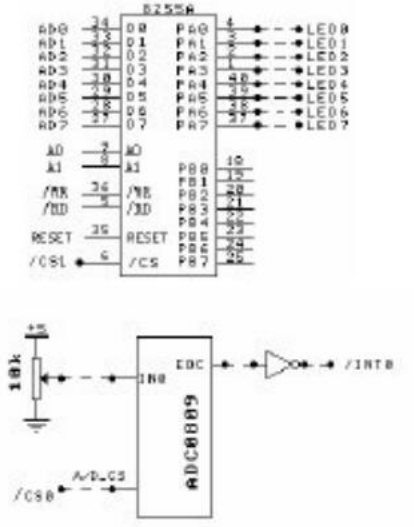
8255 地址为：**PA** 口地址**:9000H**　**PB** 口地址**:9001H**　**PC** 口地址**:9002H**

8255 控制寄存器（**8** 位）地址　**9003H**

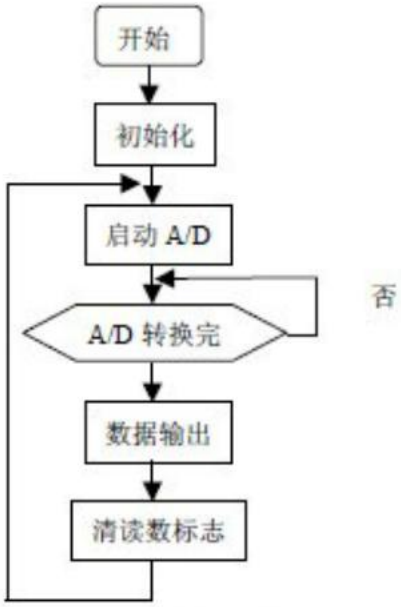
ADC0809 地址（读取的转换结果存放地址）：**8000H**



连线	连接孔 1	连接孔 2
1	IN0	电位器输出
2	AD_CS	CS0
3	EOC	INT0
4	8255_CS	CS1
5	PA0	L0
6	PA1	L1
7	PA2	L2
8	PA3	L3
9	PA4	L4
10	PA5	L5
11	PA6	L6
12	PA7	L7



实验参考程序及框图：



主程序框图

```
mode    equ 082h
PA      equ 09000h
CTL     equ 09003h
CS0809 equ 08000h
code    segment
assume cs:code
start:
mov    al, mode
mov    dx, CTL
out    dx, al
again:
mov    al, 0
mov    dx, CS0809
out    dx, al          ;起动 A/D
mov    cx, 40h
loop   $               ; 延时 > 100us
in     al, dx          ; 读入结果
mov    dx, PA
out    dx, al
jmp    again
code    ends
end start
```

8086 实验 7　8255 输入输出实验

一　实验目的

1　熟悉 8086 片外设备，硬件资源及其连接

2　熟悉 CPU 访问片外设备方式

3　了解 8255 芯片结构及编程使用方法

二　实验内容

利用 8255 可编程并行口芯片，实现输入输出实验。

1 实验用 8255PA 口（八位）作输出连接 LED0-LED7，编程实现八个 LED 的循环点亮。在每点亮一个 LED 后需要一定的延时才能够观察到灯的变化，延时程序编写与使用。

2 实验用 8255PA 口（八位）作输出，接 LED0-LED7；8255PB 口（8 位）做输入接 K0-K7(开关量 0 或 1)。用 PB 口读入开关状态，通过 PA 口用 LED 显示各个开关状态（1 亮，0 灭）。

三．实验仪器　微机、WAVE6000 编程环境，实验箱

说明：工作方式选择：工作方式 0

CPU 以片外映射存储器（IOMAP）的形式访问外设。采用不同的指令区分内外。

8255 片选信号 **CS** 连接 CPU 片选 **CS0** 时对应 8255 地址为（CPU 分配）：

PA 口地址**:8000H**

PB 口地址**: 8001H**

PC 口地址**:8002H**

8255 控制寄存器（**8** 位）地址　**8003H**

CPU 要使用外设（8255），首先设定 8255 的工作过方式，即写控制字寄存器(8003H)

注意：片外设备的访问指令形式（参考教材）

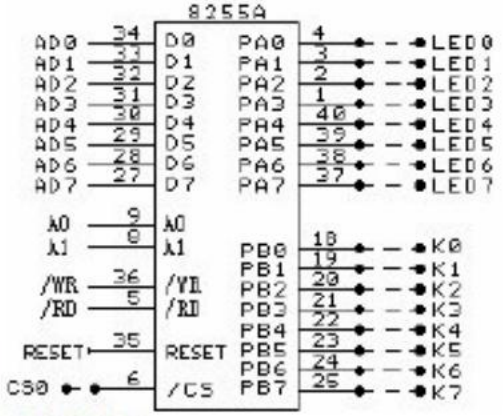
IN　　OUT　　DX 寄存器的寻址方式

参考程序（伪指令）：

```
mode    equ    082h     ; mode 8255 方式 0     ，PA，PC 输出，PB 输入
PortA   equ    8000h    ; Port A
PortB   equ    8001h    ; Port B
PortC   equ    8002h    ; Port C
CAddr   equ    8003h    ; 控制字地址
------
```

三、实验电路及连线

8255 的 CS/接地址译码/CS0，则命令字地址为 8003H，PA 口地址为 8000H，PB 口地址为 8001H，PC 口地址为 8002H。PA0-PA7（PA 口）接 LED0-LED7（LED）PB0-PB7（PB 口）接 K0-K7（开关量）。数据线、读/写控制、地址线、复位信号板上已接好。



连线	连接孔 1	连接孔 2
1	CS0	8255CS
2	L0	8255-PA0
3	L1	8255-PA1
4	L2	8255-PA2
5	L3	8255-PA3
6	L4	8255-PA4
7	L5	8255-PA5
8	L6	8255-PA6
9	L7	8255-PA7
10	K0	8255-PB0
11	K1	8255-PB1
12	K2	8255-PB2
13	K3	8255-PB3
14	K4	8255-PB4
15	K5	8255-PB5
16	K6	8255-PB6
17	K7	8255-PB7

四、实验说明

可编程通用接口芯片 8255A 有三个八位的并行 I/O 口，它有三种工作方式。本实验采用的是方式 0：PA，PC 口输出，PB 口输入。很多 I/O 实验都可以通过 8255 来实现。

五、实验框图

实验 8　8253 定时器实验

一　实验目的

1　熟悉 8086 控制片外设备 8253 定时器方法

2　熟悉 8253 硬件资源，工作方式，大时间常数的计数方法

3　了解 8253 芯片结构及编程使用方法

二　实验内容：用 8253 对标准脉冲信号进行计数，实现定时功能。用板上 1MHz 做标准脉冲，将 8253 的时间常数设为 1000000，在定时器的管教上输出高/1 秒低/1 秒的脉冲信号。因为 8253 每个计数器只有 16 位，（最大数值 65535），所以要用两个计数器实现 1000000 次的计数。

两级计数初值选取：　1000×1000 或者 20×50000=1000000

注意：各个计数器工作方式选择：要求输出方波信号，所以末级计数器采用工作方式 3（方波发生器）；第一级计数器的输出作为第二级计数器的时钟信号（标准脉冲）也要求是方波信号，所以也选工作方式 3。

两级计数都采用工作方式 3

8253CS 连接 CPU　CS4 时 8253 各部分的地址:(地址复用)

计数器 0 地址　c000H

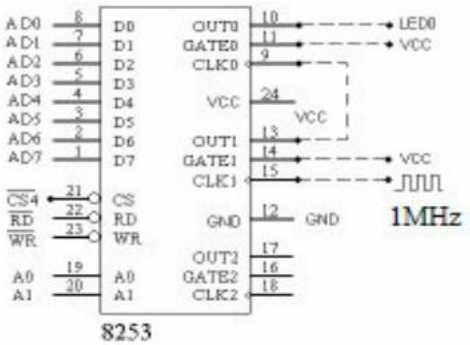
计数器 1 地址　0c001H

计数器 2 地址　0c002H

8253 的控制字寄存器地址 c003H

实验连线：

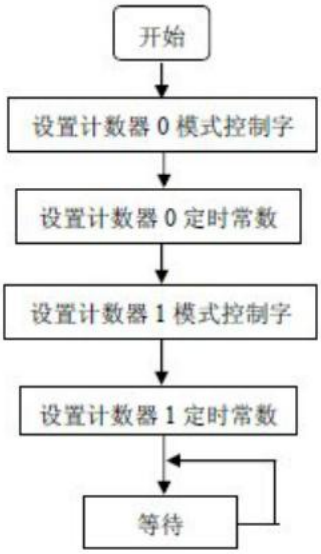
连线	连接孔 1	连接孔 2
1	8253_CS	CS4
2	8253_OUT0	L0
3	8253_GATE0	VCC
4	8253_CLK0	8253_OUT1
5	8253_GATE1	VCC
6	8253_CLK1	F/4(1M)
7	4MHz	Fin



说明：采用两级计数，计数器 0 和计数器 1；计数器 1(T1)是第一级数器见表中第 6 行（8253_CLK1 连接 F/4(1M)）;第一级输出 8253_OUT1 连接第二级计数器（T0）的计数脉冲输入端 8253_CLK0;第二级输出 8253_OUT0 到 LED（L0）。

8253 计数器实验参考程序：

框图如右：



主程序框图

```
CONTROL equ 0c003h
COUNT0  equ 0c000h
COUNT1  equ 0c001h
COUNT2  equ 0c002h
code    segment
        assume cs:code
start:
        mov   al, 36h; 00110110B  ; 计数器 0,16 位,方式 3,二进制
        mov   dx, CONTROL
        out   dx, al
        mov   ax, 1000
        mov   dx, COUNT0
        out   dx, al              ; 计数器低字节
        mov   al, ah
        out   dx, al              ; 计数器高字节
        mov   al, 76h; 01110110B  ; 计数器 1,16 位,方式 3,二进制
        mov   dx, CONTROL
        out   dx, al
        mov   ax, 1000
        mov   dx, COUNT1
        out   dx, al              ; 计数器低字节
        mov   al, ah
        out   dx, al              ; 计数器高字节
        jmp   $
code    ends
        end start
```

实验 9　8253 计数器实验

一、实验要求

利用 8088/86 外接 8253 可编程定时器/计数器，可以实现对外部事件进行计数。设置断点读回计数器的值。

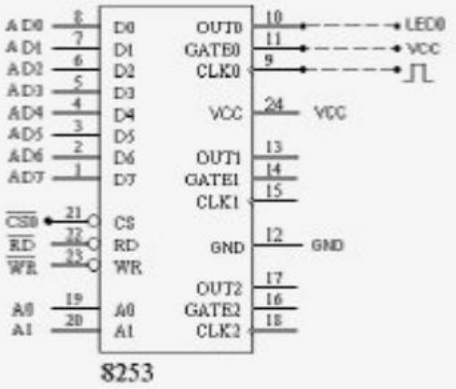
二、实验目的

1、学习 8088/86 与 8253 的连接方法。

2、学习 8088/86 对 8253 的控制方法。

三、实验电路及连线

连线	连接孔 1	连接孔 2
1	8253_CS	CS0
2	8253_OUT0	L0
3	8253_GATE0	VCC
4	8253 CLK0	单脉冲



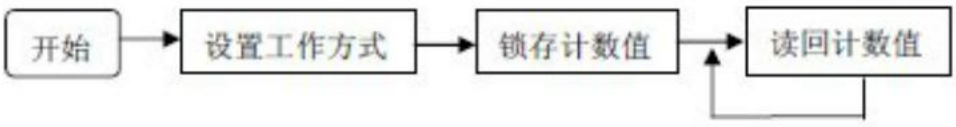
四、实验说明

3. 本实验中计数器按方式 0 工作。即十六位二进制计数器。当计数设置好后，计数器就开始计数。如果要读入计数器的值，要先锁存计数值，才能读到计数值。本实验

计数器计数执行单元在外部脉冲的激励下从技术初值 n 开始执行减 1 操作，直到执行单元中为 0 时，（n+1）个时钟，OUT 输出高电平，并保持到重新设计定计数初值或工作方式。将 OUT0 连接到 LED 上，以看其变化，看计数器是否工作。

实验中设定计数初值为 5。

实验框图如下：



由连线图可知：

8253 地址：

计数器 0：08000h

计数器 1：08001h

计数器 2：08002h

控制字地址：08003h

8253 计数器参考程序：

```
CONTROL equ 08003h
COUNT0  equ 08000h
COUNT1  equ 08001h
COUNT2  equ 08002h

code    segment
        assume cs:code

start   proc  near
        mov   al, 30h          ; 通道 0，方式 0
        mov   dx, CONTROL
        out   dx, al

        mov   al, 5            ; 计数器初始值。
        mov   dx, COUNT0
        out   dx, al           ; 低八位
        mov   al, 0
        out   dx, al           ; 高八位
Again:
        mov   al, 00000000B    ; 锁存计数器值
        mov   dx, CONTROL
        out   dx, al

        mov   dx, COUNT0
        in    al, dx           ; 读入计数值低八位
        mov   bl, al
        in    al, dx           ; 读入计数值高八位
        mov   ah, al
        mov   al, bl
        jmp   Again

start   endp
code    ends
        end start
```

8086 实验 10　步进电机控制实验

一、　实验要求

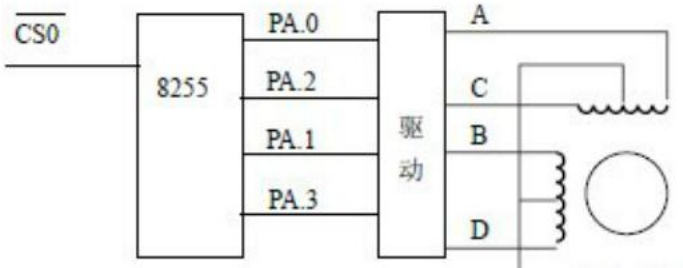
用 8255 扩展端口控制步进电机，编写程序输出脉冲序列到 8255 的 PA 口，控制步进电机正转、反转，加速，减速。

二、实验目的

1. 了解步进电机控制的基本原理。
2. 掌握控制步进电机转动的编程方法。
3. 了解单片机控制外部设备的常用电路。

三、实验电路连线框图

8255 控制的原理图参见 8255 实验。

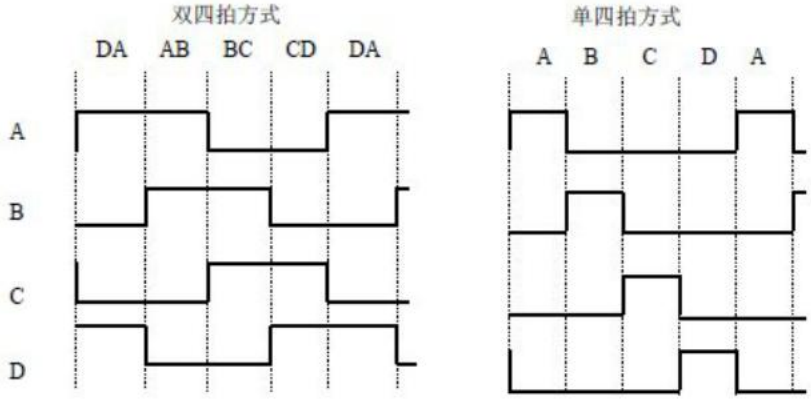


连线	连接孔 1	连接孔 2
1	8255_CS	CS0
2	PA0	步进电机 A 项
3	PA1	步进电机 B 项
4	PA2	步进电机 C 项
5	PA3	步进电机 D 项

四、实验说明

步进电机驱动原理是通过对每相线圈中的电流的顺序切换来使电机作步进式旋转。切换是通过单片机输出脉冲信号来实现的。所以调节脉冲信号的频率便可以改变步进电机的转速，改变各相脉冲的先后顺序，可以改变电机的旋转方向。步进电机的转速应由慢到快逐步加速。

电机驱动方式可以采用双四拍(AB→BC→CD→DA→AB)方式，也可以采用单四拍(A→B→C→D→A)方式，或单、双八拍(A→AB→B→BC→C→CD→D→DA→A)方式。各种工作方式的时序图如下：（高电平有效）



单四拍工作方式参考程序如下：

```
mode     equ 082h
ctl      equ 08000h
contrl equ 08003h
Astep    equ 01h
Bstep    equ 02h
Cstep    equ 04h
Dstep    equ 08h
data     segment
dly_c    dw   0
data     ends
code     segment
          assume cs:code, ds:data
start:
         mov    ax, data
         mov    ds, ax
         mov    dx, contrl
         mov    al, mode
         out    dx, al
         mov    dx, ctl
         mov    al, 0
         out    dx, al
         mov    dly_c, 1000h
step41:
          mov    dx,ctl
         mov    al,Dstep
         out    dx,al
         call   delay
         mov    al,Cstep
         out    dx,al
         call   delay
         mov    al,Bstep
         out    dx,al
         call   delay
         mov    al,Astep
         out    dx,al
         call   delay
         jmp    step41
delay: mov    cx, dly_c
dd1:   loop   dd1
         ret
code   ends
         end    start
```