微机原理课程作业及答案

一、第二章作业

第二章作业答案

2.1 微处理器主要由哪几部分构成?

答案:
微处理器主要由以下部分构成:

- **运算器（ALU)**:完成算术运算和逻辑运算。
- **控制器**:分析指令并生成控制信号，协调各部件工作。
- **寄存器组**:包括通用寄存器（如AX、BX)、段寄存器（如CS、DS)、指令指针（IP）和标志寄存器（FLAGS）等，用于暂存数据和地址。

2.3 说明8086 CPU中EU和BIU两个单元的主要功能。在执行指令时，EU能直接访问寄存器吗?

答案:

- **EU（执行单元）** 的主要功能:
  - 执行指令中的算术逻辑运算;
  - 管理通用寄存器和标志寄存器;
  - 向BIU发送数据和地址请求。
- **BIU（总线接口单元）** 的主要功能:
  - 负责与存储器、I/O端口进行数据传输;
  - 取指令并放入指令队列;
  - 计算物理地址（由段地址和偏移地址合成)。
- **EU执行指令时可直接访问寄存器**，因为通用寄存器和标志寄存器属于EU的内部部件，无需通过BIU。

2.6 由8088工作在单CPU下，在表中填入不同操作时各控制信号状态（表格略，常见控制信号状态如下)。

答案:

操作类型	M/IO#	RD#	WR#
存储器读	1	0	1
存储器写	1	1	0
I/O端口读	0	0	1
I/O端口写	0	1	0

- **说明**:

- M/IO#：1表示访问存储器，0表示访问I/O端口；
- RD#：0表示读操作有效；
- WR#：0表示写操作有效。

2.9 8086/8088系统中，存储器为什么要分段？一个段最大为多少字节？最小为多少字节？

答案：

- **分段原因**：
  8086/8088的地址线为20位（可寻址1MB），但内部寄存器为16位。分段可将1MB空间划分为多个64KB的段，通过段寄存器和偏移地址组合生成20位物理地址，解决寄存器位数不足的问题，同时便于程序模块化设计。
- **段的大小**：
  - 最大段：64KB（2^{16}字节，因偏移地址为16位，范围0~65535）；
  - 最小段：16字节（当段内偏移地址范围为0~15时，实际使用的存储空间为16字节）。

2.10 在8086/8088 CPU中，物理地址和逻辑地址是指什么？已知逻辑地址为1F00H:38A0H，如何计算其对应的物理地址？若已知物理地址，其逻辑地址唯一吗？

答案：

- **逻辑地址**：程序中使用的地址，由段地址和偏移地址组成，格式为“段地址:偏移地址”。
- **物理地址**：存储器的实际地址，是20位二进制数，由逻辑地址计算得到。
- **物理地址计算**：
  物理地址 = 段地址 × 16 + 偏移地址（即段地址左移4位 + 偏移地址）。
  对逻辑地址1F00H:38A0H：
  物理地址 = 1F00H × 16 + 38A0H = 1F000H + 38A0H = **228A0H**。
- **逻辑地址不唯一**：不同的段地址和偏移地址组合可能生成相同的物理地址。例如，物理地址228A0H可对应逻辑地址2000H:28A0H、2100H:18A0H等。

2.11 若CS=8000H，则当前代码段可寻址的存储空间范围是？

答案：

- CS为代码段寄存器，段地址为8000H，偏移地址范围为0000H~FFFFH。
- 代码段寻址范围：
  起始地址 = 8000H × 16 + 0000H = **80000H**，
  结束地址 = 8000H × 16 + FFFFH = **8FFFFH**。
- 范围：**80000H ~ 8FFFFH**（共64KB）。

2.12 8086/8088 CPU在最小模式下构成计算机系统至少应包括哪几个基本部分（器件）？

答案：
最小模式下的基本器件包括：

- **8086/8088 CPU**：核心处理单元；
- **时钟发生器8284A**：提供时钟信号；
- **地址锁存器（如8282或74LS373）**：锁存地址信号；
- **数据缓冲器（如8286/8287）**：缓冲数据总线信号，增强驱动能力；

- **电源和复位电路**：提供电源和系统复位功能。

二、第三章作业

第三章作业答案

一、简答题

1. **什么叫寻址方式？8086/8088 CPU共有哪几种寻址方式?**
   答案：
   - **寻址方式**是指CPU在执行指令时，寻找操作数或操作数地址的方法。
   - 8086/8088 CPU共有7种寻址方式：
     a. **立即寻址**：操作数直接包含在指令中（如 `MOV AX, 100H`）。
     b. **寄存器寻址**：操作数存放在寄存器中（如 `MOV AX, BX`）。
     c. **直接寻址**：操作数地址直接在指令中（如 `MOV AX, [100H]`）。
     d. **寄存器间接寻址**：操作数地址存放在寄存器中（如 `MOV AX, [BX]`）。
     e. **寄存器相对寻址**：操作数地址为寄存器值与偏移量之和（如 `MOV AX, [BX+10H]`）。
     f. **基址变址寻址**：操作数地址为基址寄存器与变址寄存器值之和（如 `MOV AX, [BX+SI]`）。
     g. **相对基址变址寻址**：操作数地址为基址寄存器、变址寄存器值与偏移量之和（如 `MOV AX, [BX+SI+10H]`）。
     h. **隐含寻址**：操作数地址隐含在指令中（如 `MUL BL`）。

二、指令寻址方式与物理地址计算

2. **设(DS)=6000H，(ES)=2000H，(SS)=1500H，(SI)=00A0H，(BX)=0800H，(BP)=1200H，VAR=0050H。指出下列指令源操作数的寻址方式及物理地址。**

指令	寻址方式	物理地址计算	物理地址
(1) MOV AX, BX	寄存器寻址	无（操作数在寄存器中）	-
(2) MOV DL, 80H	立即寻址	无（操作数为立即数）	-
(3) MOV AX, VAR	直接寻址	`DS×16 + VAR = 6000H×16 + 0050H`	**60050H**
(4) MOV AX, VAR[BX][SI]	相对基址变址寻址	`DS×16 + VAR + BX + SI = 6000H×16 + 0050H + 0800H + 00A0H`	**608F0H**
(5) MOV AL, 'B'	立即寻址	无（操作数为字符'B'的ASCII码42H）	-

指令	寻址方式	物理地址计算	物理地址
(6) MOV DI, ES:[BX]	寄存器间接寻址（ES段）	`ES×16 + BX = 2000H×16 + 0800H`	**20800H**
(7) MOV DX, [BP]	寄存器间接寻址（SS段）	`SS×16 + BP = 1500H×16 + 1200H`	**16200H**
(8) MOV BX, 20H[BX]	寄存器相对寻址	`DS×16 + 20H + BX = 6000H×16 + 20H + 0800H`	**60820H**

三、转移指令地址计算

3. **假设(DS)=212AH，(CS)=0200H，(IP)=1200H，(BX)=0500H，DATA=40H，(217A0H)=2300H，(217E0H)=0400H，(217E2H)=9000H。确定转移指令的目标地址。**
   - (1) **JMP 2300H**
     - 类型：段内直接转移（IP修改为2300H）。
     - 转移地址： `CS:IP = 0200H:2300H` （物理地址=0200H×16+2300H=04300H）。
   - (2) **JMP WORD PTR [BX]**
     - 类型：段内间接转移（[BX]指向的内存单元存偏移地址）。
     - 地址计算： `DS×16 + BX = 212AH×16 + 0500H = 217A0H` ，该单元内容为2300H（IP新值）。
     - 转移地址： `0200H:2300H` （物理地址04300H）。
   - (3) **JMP DWORD PTR [BX+DATA]**
     - 类型：段间间接转移（[BX+DATA]指向的内存单元存偏移地址和段地址）。
     - 地址计算： `BX+DATA=0500H+40H=0540H` ， `DS×16+0540H=212AH×16+0540H=217E0H` 。
     - `217E0H` 单元存0400H（IP新值）， `217E2H` 单元存9000H（CS新值）。
     - 转移地址： `9000H:0400H` （物理地址9000H×16+0400H=90400H）。

四、指令执行后标志位状态

4. **(AL)=7BH，(BL)=38H，执行ADD AL, BL后，AF、CF、OF、PF、SF、ZF的值各是多少?**
   - **二进制运算**：

     `AL=7BH=01111011B` ， `BL=38H=00111000B` ，相加结果：

     ```
     01111011
     +00111000
     ---------
     10110011 = B3H
     ```

   - **标志位解析**：
     - **AF（辅助进位）**：低4位相加（1011+1000=10011），向高位进位，AF=1。
     - **CF（进位）**：最高位无进位（0+0=1，无进位），CF=0。
     - **OF（溢出）**：两正数相加结果为负数（最高位1），发生溢出，OF=1。
     - **PF（奇偶）**：结果B3H=10110011B，1的个数为5（奇数），PF=0。

- **SF（符号）**：结果最高位为1，视为负数，SF=1。
- **ZF（零）**：结果非0，ZF=0。

- **答案**：AF=1，CF=0，OF=1，PF=1，SF=1，ZF=0。
- **解释** OF 公式（更通用的方法）
  OF = 最高位进位（C7） ⊕ 次高位进位（C6）
  （⊕ 表示异或，即不同时为 1 或 0 时 OF=1）

五、程序段编写

5. **完成以下功能的程序段：**
   - (1) 从DS:0012H取56H到AL；
   - (2) AL左移两位；
   - (3) AL与DS:0013H相乘；
   - (4) 乘积存DS:0014H。

   程序段：

```
MOV AL, [0012H]   ; (1) 取DS:0012H的数据到AL
SHL AL, 2         ; (2) AL左移两位（乘以4）
MUL BYTE PTR [0013H]  ; (3) AL与DS:0013H相乘，结果在AX
MOV [0014H], AX   ; (4) 乘积存DS:0014H
```

六、堆栈操作与SP变化

6. **SP初值2300H，AX=50ABH，BX=1234H。执行PUSH AX、PUSH BX、POP AX后，SP、AX、BX的值是多少?**
   - **操作解析**：
     a. **PUSH AX**：SP=2300H-2=22FEH，堆栈存入AX=50ABH。
     b. **PUSH BX**：SP=22FEH-2=22FCH，堆栈存入BX=1234H。
     c. **POP AX**：AX=堆栈顶部值=1234H，SP=22FCH+2=22FEH。
   - **答案**：
     - SP=**22FEH**，
     - AX=**1234H**，
     - BX=**1234H**（值不变）。
     -

第四章作业

第四章作业答案

1. 数据串操作程序段

题目要求：使用串操作指令完成DATA1与DATA2的字符串传送、比较及数据存取。
答案：

```
; （1）字符串从DATA1传送到DATA2
LEA SI, DATA1          ; SI指向源串首地址
LEA DI, DATA2          ; DI指向目标串首地址
MOV CX, 19             ; 字符串长度（'HELLO! GOOD MORNING!'共19字节）
CLD                    ; 清除方向标志，从左到右传送
REP MOVSB              ; 重复传送字节，直到CX=0

; （2）比较DATA1和DATA2的内容
LEA SI, DATA1          ; 重置SI到DATA1首地址
LEA DI, DATA2          ; 重置DI到DATA2首地址
MOV CX, 19             ; 长度不变
CLD                    ; 方向标志不变
REPE CMPSB             ; 相等时继续比较，不等则停止
JZ SAME                ; 若相等，跳转至SAME标签
; 若不相等，此处可添加错误处理代码
SAME:
; 相等时的处理代码

; （3）将DATA1的第3、4字节装入AX（索引从0开始）
LEA SI, DATA1
ADD SI, 2              ; 指向第3字节（偏移量2）
MOV AL, [SI]           ; 第3字节存入AL
INC SI                 ; 指向第4字节
MOV AH, [SI]           ; 第4字节存入AH

; （4）将AX内容存入DATA2+5的位置
LEA DI, DATA2
ADD DI, 5              ; 指向DATA2+5
MOV [DI], AL           ; 存AL到DATA2+5
INC DI                 ; 指向DATA2+6
MOV [DI], AH           ; 存AH到DATA2+6
```

2. 字符串尾端移动程序段

题目要求：将STRING1的最后20个字符移至STRING2，顺序不变。
答案：

```
LEA SI, STRING1        ; SI指向STRING1首地址
MOV CX, LENGTHOF STRING1    ; CX = STRING1 的长度
ADD SI, CX                  ; SI = STRING1 的末尾地址 +1
SUB SI, 20                  ; SI = STRING1 的最后 20 字符的首地址
; 设置目标地址（STRING2）
LEA DI, STRING2        ; DI指向STRING2首地址
CLD                    ; 正向传送
MOV CX, 20                  ; 复制 20 次
REP MOVSB              ; 复制20字节到STRING2
```

3. 若接口 03F8H 的第1位（bit1）和第3位（bit3）同时为1，表示接口 O3FBH 有准备好的8位数据；当CPU 将数据取走后，bitl 和bit3 就不再同时为1了，而仅当又有数据准备好时才再同时为

1。

编写程序，从上述接口读人 200字节的数据，并按顺序放在从 DATA 开始的单元中

题目要求：从03F8H检测状态，03FBH读取数据，存入DATA开始的单元。

答案：

```
DATA SEGMENT
    DATA DB 200 DUP (?)  ; 存储数据的缓冲区
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    LEA SI, DATA         ; SI指向数据缓冲区首地址
    MOV CX, 200          ; 读取200字节

WAIT_DATA:
    MOV DX, 03F8H        ; DX = 状态端口（03F8H）
    IN AL, DX     ; 读取状态端口03F8H
    TEST AL, (1<<1) | (1<<3)  ; 检测bit1和bit3是否同时为1
    JZ WAIT_DATA         ; 未准备好则等待
    MOV DX, 03FBH        ; DX = 状态端口（03F8H）
    IN AL, DX
    MOV [SI], AL         ; 存入缓冲区
    INC SI               ; 缓冲区指针后移
    LOOP WAIT_DATA       ; 循环直到CX=0

    MOV AH, 4CH          ; 程序结束
    INT 21H
CODE ENDS
END START
```

4. 用子程序结构编写如下程序：从键盘输入一个二位十进制的月份数（01~12），然后显示出相应月份的英文缩写名。

题目要求：输入两位十进制月份（01~12），显示英文缩写。

答案：

```
; 月份显示程序 - 使用子程序结构
; 功能：输入两位月份(01-12)，显示对应英文缩写
; 作者：AI助手
; 日期：2023-11-15

STACK_SEG SEGMENT STACK
    DB 100H DUP(?)      ; 定义256字节堆栈空间
STACK_SEG ENDS

DATA_SEG SEGMENT
    ; 月份缩写表（每个月份固定3字节）
    MONTH_TABLE DB 'JAN','FEB','MAR','APR','MAY','JUN'
                DB 'JUL','AUG','SEP','OCT','NOV','DEC'

    ; 提示信息
    MSG_PROMPT  DB 0DH,0AH,'Enter month (01-12): $'
    MSG_ERROR   DB 0DH,0AH,'Invalid input!$'
    MSG_RESULT  DB 0DH,0AH,'Month: $'

    ; 输入缓冲区
    INPUT_BUF   DB 3            ; 最大长度
                DB ?            ; 实际长度
                DB 3 DUP(?)     ; 存储空间

    MONTH_NUM   DB ?            ; 存储转换后的月份数值
DATA_SEG ENDS

CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG, DS:DATA_SEG, SS:STACK_SEG

; ============================================
; 主程序
; ============================================
MAIN PROC FAR
    MOV AX, DATA_SEG
    MOV DS, AX
    MOV ES, AX      ; 额外段寄存器也指向数据段

    CALL INPUT_MONTH    ; 调用输入子程序
    CALL DISPLAY_MONTH  ; 调用显示子程序

    MOV AH, 4CH     ; 程序结束
    INT 21H
MAIN ENDP

; ============================================
; 子程序：输入月份
; 功能：从键盘读取两位月份并验证
; 输出：MONTH_NUM = 二进制月份值(1-12)
; ============================================
```

```
INPUT_MONTH PROC NEAR
    PUSH AX
    PUSH BX
    PUSH DX

    ; 显示输入提示
    MOV AH, 09H
    LEA DX, MSG_PROMPT
    INT 21H

    ; 读取键盘输入
    MOV AH, 0AH
    LEA DX, INPUT_BUF
    INT 21H

    ; 验证输入长度是否为2
    CMP INPUT_BUF+1, 2
    JNE INVALID_INPUT

    ; 验证十位是否为数字'0'-'1'
    MOV AL, INPUT_BUF+2
    CMP AL, '0'
    JB INVALID_INPUT
    CMP AL, '1'
    JA INVALID_INPUT

    ; 验证个位是否为数字'0'-'9'
    MOV AL, INPUT_BUF+3
    CMP AL, '0'
    JB INVALID_INPUT
    CMP AL, '9'
    JA INVALID_INPUT

    ; 转换为二进制数值
    ; 十位数字处理
    MOV AL, INPUT_BUF+2
    SUB AL, '0'     ; ASCII转数字
    MOV BL, 10
    MUL BL          ; AL = 十位×10

    ; 个位数字处理
    MOV BL, INPUT_BUF+3
    SUB BL, '0'     ; ASCII转数字
    ADD AL, BL      ; AL = 十位×10 + 个位

    ; 检查范围(1-12)
    CMP AL, 1
    JB INVALID_INPUT
    CMP AL, 12
    JA INVALID_INPUT
```

```
    ; 保存有效月份
    MOV MONTH_NUM, AL
    JMP INPUT_DONE

INVALID_INPUT:
    MOV AH, 09H
    LEA DX, MSG_ERROR
    INT 21H
    MOV MONTH_NUM, 0    ; 标记为无效输入

INPUT_DONE:
    POP DX
    POP BX
    POP AX
    RET
INPUT_MONTH ENDP

; ==========================================
; 子程序：显示月份缩写
; 输入：MONTH_NUM = 二进制月份值(1-12)
; ==========================================
DISPLAY_MONTH PROC NEAR
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    ; 检查月份是否有效
    CMP MONTH_NUM, 1
    JB DISPLAY_DONE
    CMP MONTH_NUM, 12
    JA DISPLAY_DONE

    ; 显示结果前缀
    MOV AH, 09H
    LEA DX, MSG_RESULT
    INT 21H

    ; 计算月份缩写地址
    MOV AL, MONTH_NUM
    DEC AL          ; 调整为0-based索引
    MOV AH, 3       ; 每个缩写3字节
    MUL AH          ; AX = 月份索引×3
    LEA SI, MONTH_TABLE
    ADD SI, AX      ; SI指向目标缩写

    ; 显示3个字符
    MOV CX, 3
DISPLAY_LOOP:
    MOV AH, 02H     ; 单字符显示功能
```

```
    MOV DL, [SI]
    INT 21H
    INC SI
    LOOP DISPLAY_LOOP

DISPLAY_DONE:
    POP SI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
DISPLAY_MONTH ENDP

CODE_SEG ENDS
    END MAIN
```

5. 斐波那契数列计算程序

题目要求：计算前20项斐波那契数列（F(0)=0, F(1)=1, F(n)=F(n-1)+F(n-2)）。
答案：

```
DATA SEGMENT
    FIB DB 20 DUP (0)     ; 存储前20项的数组
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

MAIN:
    MOV AX, DATA
    MOV DS, AX

    ; 初始化前两项
    MOV FIB[0], 0         ; F(0)=0
    MOV FIB[1], 1         ; F(1)=1

    ; 计算F(2)到F(19)共18项
    MOV CX, 18            ; 循环次数=20-2=18
    MOV SI, 2             ; 从索引2开始存储

CALC_LOOP:
    ; F(n) = F(n-1) + F(n-2)
    MOV AL, FIB[SI-1]     ; 取F(n-1)
    ADD AL, FIB[SI-2]     ; 加F(n-2)
    MOV FIB[SI], AL       ; 存F(n)
    INC SI                ; 索引后移
    LOOP CALC_LOOP        ; 循环直到CX=0

    ; 此处可添加显示结果的代码（略）

    MOV AH, 4CH
    INT 21H
CODE ENDS
END MAIN
```

第五章作业

第五章作业答案

一、简答题

1. **说明Flash芯片的特点。**

   答案：

   Flash芯片（闪存）的特点包括：

   - **非易失性**：断电后数据不丢失，类似ROM。
   - **可擦除改写**：支持电擦除（区别于EPROM的紫外线擦除），可重复编程（擦写次数通常10万~100万次）。
   - **存储密度高**：单位面积存储容量大，成本低，适合大容量数据存储。
   - **读取速度快**：接近SRAM的读取速度，但写入速度较慢（需先擦除再写入）。

- **接口简单**：通常支持地址/数据总线复用，兼容标准总线协议（如SPI、并行总线）。
- **低功耗**：待机功耗极低，适合便携设备（如U盘、SSD）。

2. **用全地址译码将6264（8KB SRAM）连接到8088系统，地址范围38000H~39FFFH，画连接图。**

   答案：

   - **地址分析：**

     38000H~39FFFH的地址范围为8KB（2000H字节），6264容量8KB，正好匹配。

     地址线A12 ~ A0用于片内寻址（8KB=2^13，A0~A12），A19 ~ A13用于全地址译码。
   - **译码逻辑：**
     - 地址范围二进制：38000H=0011 1000 0000 0000 0000 0000B，39FFFH=0011 1001 1111 1111 1111 1111B。
     - 高位地址A19~A13需满足：0011100B（A19=0, A18=0, A17=1, A16=1, A15=1, A14=0, A13=0），通过74LS138译码器生成片选信号。

3. **用2764（8KB EPROM）和6264（8KB SRAM）组成16KB内存，ROM地址FE000H ~ FFFFFH，RAM地址F0000H ~ F1FFFH，用74LS138设计译码电路。**

   答案：

   - **地址分配：**
     - ROM（2764）：FE000H ~ FFFFFH（8KB），高位地址A19 ~ A13=1111111B（全1）。
     - RAM（6264）：F0000H ~ F1FFFH（8KB），高位地址A19 ~ A13=1111000B。
   - **74LS138译码：**
     - 输入A2=A15, A1=A14, A0=A13，其余高位地址（A19 ~ A16）需固定为1111。
     - ROM片选：当A19~A13=1111111时，74LS138输出Y7#有效。
     - RAM片选：当A19~A13=1111000时，输出Y0#有效。

4. **两个6116（2KB SRAM）地址61000H~61FFFH，连接8088并编写测试程序，输入数据后比较，出错显示“Wrong！”，正确显示“OK！”。**

   答案：

   - **硬件连接：**

     6116为2KB×8，两个芯片并联组成2KB×16（16位数据），地址线A0~A10（2KB=2^11），片选接地址译码器输出（61000H ~ 61FFFH对应A19 ~ A11=0110001）。
   - **测试程序：**

```
TEST_PROC PROC
    MOV AX, 61000H
    MOV DS, AX
    MOV CX, 2048        ; 2KB=2048字
    MOV AX, 55AAH       ; 测试数据
WRITE_LOOP:
    MOV [0], AX
    INC SI
    LOOP WRITE_LOOP
    MOV CX, 2048
    MOV SI, 0
CHECK_LOOP:
    MOV DX, [SI]
    CMP DX, 55AAH
    JNE ERROR
    INC SI
    LOOP CHECK_LOOP
    JMP DISPLAY_OK
ERROR:
    LEA DX, WRONG_MSG
    MOV AH, 09H
    INT 21H
    JMP EXIT
DISPLAY_OK:
    LEA DX, OK_MSG
    MOV AH, 09H
    INT 21H
EXIT:
    RET
TEST_PROC ENDP
```

5. **16位地址总线，用4K×8b SRAM构成32KB内存，地址0000H~7FFFH，求芯片数并设计电路。**
   问题分析与解答

（1）需要多少个 SRAM 芯片?

- **目标内存容量**：32KB（32 × 1024 字节）
- **单个 SRAM 芯片容量**：4K × 8b = 4KB
- **所需芯片数量**：

$$\frac{32\,\mathrm{KB}}{4\,\mathrm{KB}} = 8\,\text{片}$$

答案：需要 **8 片** 4K×8b 的 SRAM 芯片。

（2）内存电路设计（地址范围 0000H～7FFFH）

设计步骤：

1. **地址空间分配**：
   - 32KB 内存的地址范围为 **0000H～7FFFH**（共 32K 字节）。
   - 16 位地址总线（A0～A15），其中：
     - **A0～A11**：用于片内寻址（4KB = 2^{12}，需 12 位地址）。
     - **A12～A15**：用于片选（选择 8 个芯片中的一个）。
2. **片选逻辑设计**：
   - 8 个芯片需要 **3 位片选信号**（$\log_2 8 = 3$）。
   - 使用 **A12～A14** 作为片选输入（A15 固定为 0，因为地址范围是 0000H～7FFFH）。
   - 片选信号通过 **3-8 译码器（如 74LS138）** 生成。
3. **芯片连接方式**：
   - **数据总线**（D0～D7）：所有芯片的 D0～D7 直接并联到系统数据总线。
   - **地址总线**（A0～A11）：连接到所有芯片的 A0～A11（片内寻址）。
   - **控制信号**：
     - **#CE（片选）**：由 3-8 译码器输出控制。
     - **#RD（读）**、**#WR（写）**：直接连接到系统控制总线。
4. **地址译码逻辑**：
   - 3-8 译码器的输入：
     - **A12～A14** 作为选择信号。
     - **A15 = 0** 使能译码器（确保地址范围在 0000H～7FFFH）。
   - 译码器输出 **Y0～Y7** 分别连接 8 个芯片的 **#CE**。
5. **地址分配表**：

芯片编号	片选信号（#CE）	地址范围
0	Y0	0000H～0FFFH
1	Y1	1000H～1FFFH
2	Y2	2000H～2FFFH
3	Y3	3000H～3FFFH
4	Y4	4000H～4FFFH
5	Y5	5000H～5FFFH
6	Y6	6000H～6FFFH
7	Y7	7000H～7FFFH

电路图关键部分：

1. **地址译码器（74LS138）**：
   - 输入：A12、A13、A14（A15 接地或接低电平）。

- 输出：Y0～Y7 分别连接 8 个 SRAM 的 #CE。

2. **SRAM 芯片连接**：
   - 所有芯片的 A0～A11 并联到系统地址总线 A0～A11。
   - 所有芯片的 D0～D7 并联到系统数据总线 D0～D7。
   - #RD 和 #WR 直接连接到系统控制总线。

验证地址范围：

- **总地址空间**：16 位地址总线可寻址 $2^{16} = 64\,\mathrm{KB}$。
- **设计范围**：0000H～7FFFH（32KB），符合要求。
- **片选逻辑**：
  - 当 A15=0 时，译码器工作。
  - A12～A14 的组合选择 8 个芯片之一。

最终答案

1. **需要 8 片 4K×8b 的 SRAM 芯片**。
2. **电路设计**：
   - 使用 3-8 译码器（74LS138）生成片选信号（A12～A14 输入）。
   - 8 个芯片的地址范围分别为 0000H～0FFFH、1000H～1FFFH、…、7000H～7FFFH。
   - 数据总线（D0～D7）和控制信号（#RD、#WR）直接并联。

此设计完全满足 **32KB 内存（0000H～7FFFH）** 的需求，且硬件连接简洁可靠。

第六章作业

第六章作业答案

1. **试比较4种基本输入输出方式的特点。**

   答案：

   4种基本输入输出方式包括程序查询方式、中断方式、DMA方式和通道方式，特点如下：
   - **程序查询方式**：CPU通过循环查询外设状态，状态就绪时执行I/O操作。优点是编程简单，硬件开销小；缺点是CPU利用率低，实时性差。
   - **中断方式**：外设就绪时向CPU发送中断请求，CPU响应后执行I/O操作。优点是CPU无需持续查询，利用率提高；缺点是处理中断需保存/恢复现场，适合中低速设备。
   - **DMA方式**：通过DMA控制器（DMAC）直接控制数据在内存与外设间传输，无需CPU干预。优点是传输速度快，适合高速数据块传输；缺点是硬件复杂，需DMAC支持。
   - **无条件传送方式**：CPU无需查询外设状态，直接执行I/O操作。优点是实时性强；缺点是编程复杂，CPU占用率高。

2. **8086/8088系统如何确定硬件中断服务程序的入口地址?**

   答案：

   8086/8088通过中断向量表确定中断服务程序入口地址，步骤如下：

- 硬件中断（可屏蔽中断INTR或非屏蔽中断NMI）产生时，CPU获取中断类型号n（0~255）。
- 中断向量表位于内存0地址开始处，每个中断向量占4字节，前2字节为偏移地址IP，后2字节为段地址CS。
- 入口地址计算：IP = (n×4)单元内容，CS = (n×4+2)单元内容，组合为CS:IP即中断服务程序入口。

3. **单片8259A能够管理多少级可屏蔽中断？若用3片8259A级联，能管理多少级可屏蔽中断？**

   答案：

   - 单片8259A可管理**8级**可屏蔽中断（IR0~IR7）。
   - 3片级联时，1片为主片，2片为从片：
     - 主片IR0~IR7中，2个引脚（如IR2、IR3）连接从片，每个从片提供8级中断。
     - 总中断级数 = 主片剩余6级 + 从片各8级 → **6 + 8×2 = 22级**。

4. **设计题：设输入接口地址0E54H，输出接口地址01FBH，用74LS244（输入）和74LS273（输出）连接8088系统总线，编写程序：当输入接口bit1、bit4、bit7同时为1时，输出DATA为首址的20字节数据，否则等待。**

   答案：

   - **接口连接：**
     - 74LS244（三态缓冲器）：数据端接数据总线D0~D7，片选端接地址译码器输出（对应0E54H），读信号接CPU的RD#和IO/M#。
     - 74LS273（D触发器）：数据端接数据总线，时钟端接地址译码器输出（对应01FBH），写信号接CPU的WR#和IO/M#。
   - **程序段：**

```
DATA SEGMENT
    DATA DB 20 DUP(?)  ; 数据区
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV CX, 20         ; 输出数据个数
    LEA SI, DATA       ; SI指向数据区
CHECK:
    IN AL, 0E54H       ; 读取输入接口
    TEST AL, (1<<1)+(1<<4)+(1<<7)  ; 检测bit1、4、7
    JNZ TRANSFER       ; 全为1则转移
    JMP CHECK          ; 否则等待
TRANSFER:
    MOV AL, [SI]       ; 取数据
    OUT 01FBH, AL      ; 输出数据
    INC SI             ; 指向下一数据
    LOOP TRANSFER      ; 循环20次
    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

5. **编写8259A初始化程序：单芯片，8个中断源边沿触发，非缓冲，一般全嵌套，中断向量40H。**

   答案：

```
; 8259A端口地址假设为20H（偶地址）和21H（奇地址）
MOV AL, 00010001B  ; ICW1：边沿触发，单片，需ICW4
OUT 20H, AL
MOV AL, 40H        ; ICW2：中断向量基址40H
OUT 21H, AL
MOV AL, 00000001B  ; ICW4：一般全嵌套，非缓冲，8086模式
OUT 21H, AL
MOV AL, 00000000B  ; OCW1：开放所有中断（IR0~IR7）
OUT 21H, AL
```

第七章作业答案

一、简答题

1. **接口芯片的读写信号应与系统的哪些信号相连?**
   答案：
   接口芯片的读写信号通常连接系统的以下信号：
   - **读信号（RD#）**：与CPU的RD#信号相连，低电平有效时允许接口输出数据到数据总线。
   - **写信号（WR#）**：与CPU的WR#信号相连，低电平有效时允许CPU向接口写入数据。
   - **地址选择信号（如片选CS#、端口选择A0/A1等）**：通过地址译码器生成，确保仅当接口地址有效时读写信号生效。
   - **总线控制信号（如IO/M#）**：区分I/O操作（IO/M#=1）或内存操作（IO/M#=0），与RD#/WR#组合控制读写。

2. **8255各端口可以工作在几种方式下？当端口A工作在方式2时，端口B和C工作于什么方式?**
   答案：
   - **8255端口工作方式：**
     - 端口A：可工作于方式0（基本输入/输出）、方式1（选通输入/输出）、方式2（双向传输）。
     - 端口B：可工作于方式0、方式1。
     - 端口C：可工作于方式0，或作为端口A/B的控制信号（方式1/2时）。
   - **端口A工作在方式2时：**
     - 端口B可工作于方式0或方式1（输入/输出）。
     - 端口C的PC3$_{PC7作为端口A方式2的控制信号（如STB#、ACK#等）}$，$_{PC0}$PC2可作为端口B的控制信号（若端口B工作于方式1）或基本输入/输出（方式0）。

3. **8253芯片地址D0D0H~D0D3H，时钟2MHz，用计数器0、1、2产生周期10μs方波、1ms和1s负脉冲，画连接图并编写程序。**
   答案：

我们需要使用 8253 定时器/计数器 的三个计数器（Counter 0、1、2）分别产生以下信号：

1. **Counter 0**：周期 **10μs** 的方波（频率 = 100kHz）。
2. **Counter 1**：周期 **1ms** 的负脉冲（低电平有效）。

3. **Counter 2**：周期 **1s** 的负脉冲（低电平有效）。

给定的 **8253 地址范围** 是 `D0D0H ~ D0D3H`，对应：

- `D0D0H` → Counter 0
- `D0D1H` → Counter 1
- `D0D2H` → Counter 2
- `D0D3H` → 控制字寄存器

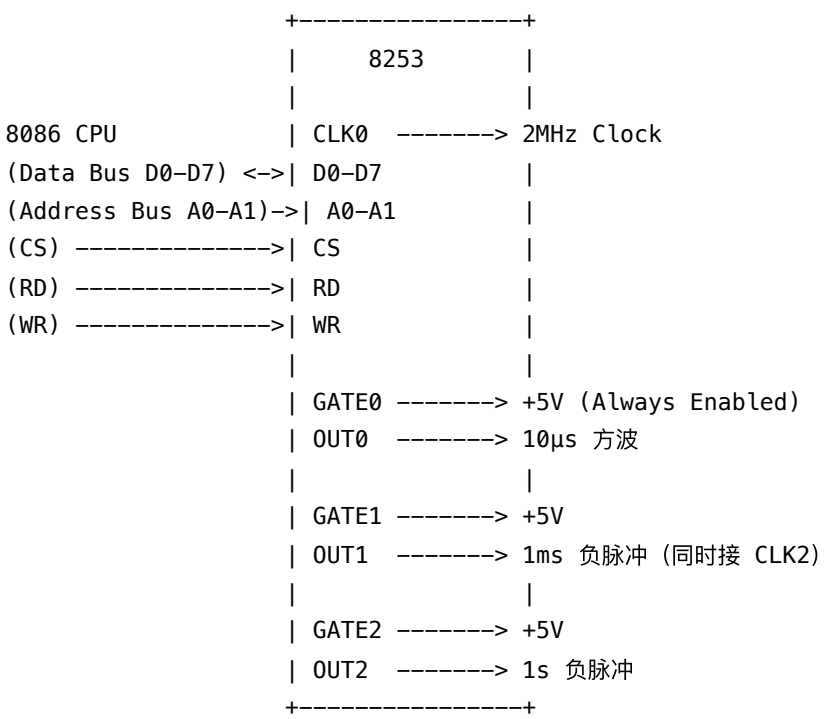
时钟频率 为 **2MHz**（即时钟周期 = 0.5μs）。

为了产生 **1s 的负脉冲**，我们需要 **级联 Counter1 和 Counter2**：

- **Counter1** 输出 **1ms** 的脉冲，作为 **Counter2 的时钟输入**。
- **Counter2** 计数 **1000 次**，得到 **1s** 的负脉冲。

1）. 硬件连接图

8253 需要连接 **8086 CPU** 的数据总线、地址总线、控制信号（`RD`、`WR`、`CS`）以及 **CLK** 输入（2MHz）。
以下是 **8253 的连接示意图**：



说明：

- **CLK0**、**CLK1** 接 **2MHz** 时钟。
- **CLK2** 接 **Counter1 的 OUT1**（1ms 时钟）。
- **GATE0**、**GATE1**、**GATE2** 接 **+5V**（始终允许计数）。
- **OUT0** 输出 **10μs 方波**。
- **OUT1** 输出 **1ms 负脉冲**（同时作为 Counter2 的时钟）。
- **OUT2** 输出 **1s 负脉冲**。

2）. 8253 初始化计算

8253 的每个计数器需要设置 **工作模式** 和 **计数初值**。

(1) Counter 0（10μs 方波）

- **模式 3（方波发生器）**
- 时钟周期 = 0.5μs（2MHz）
- 输出周期 = 10μs → **计数值 = 10μs / 0.5μs = 20**
- **控制字**： `00110110` （36H）→ 计数器 0，低+高字节写入，模式 3，二进制计数

(2) Counter 1（1ms 负脉冲）

- **模式 1（硬件可重触发单稳）**
- 时钟周期 = 0.5μs
- 输出脉宽 = 1ms → **计数值 = 1ms / 0.5μs = 2000**
- **控制字**： `01110010` （72H）→ 计数器 1，低+高字节写入，模式 1，二进制计数

(3) Counter 2（1s 负脉冲）

- **模式 1（硬件可重触发单稳）**
- 时钟输入来自 **Counter1（1ms 脉冲）**
- 输出脉宽 = 1s → **计数值 = 1s / 1ms = 1000**
- **控制字**： `10110010` （B2H）→ 计数器 2，低+高字节写入，模式 1，二进制计数

3）. 8086 汇编代码

```
; 8253 端口地址定义
PORT_COUNTER0 EQU 0D0D0H
PORT_COUNTER1 EQU 0D0D1H
PORT_COUNTER2 EQU 0D0D2H
PORT_CTRL     EQU 0D0D3H

; 初始化 8253
MOV AL, 36H       ; 计数器0，模式3（方波），二进制
OUT PORT_CTRL, AL ; 写入控制字
MOV AX, 20        ; 计数值 = 20（10μs）
OUT PORT_COUNTER0, AL ; 先写低字节
MOV AL, AH
OUT PORT_COUNTER0, AL ; 再写高字节

MOV AL, 72H       ; 计数器1，模式1（单稳），二进制
OUT PORT_CTRL, AL
MOV AX, 2000      ; 计数值 = 2000（1ms）
OUT PORT_COUNTER1, AL ; 低字节
MOV AL, AH
OUT PORT_COUNTER1, AL ; 高字节

MOV AL, 0B2H      ; 计数器2，模式1（单稳），二进制
OUT PORT_CTRL, AL
MOV AX, 1000      ; 计数值 = 1000（1s）
OUT PORT_COUNTER2, AL ; 低字节
MOV AL, AH
OUT PORT_COUNTER2, AL ; 高字节

; 主程序（循环等待）
MAIN_LOOP:
JMP MAIN_LOOP
```

4）. 说明

1. **Counter 0**：
   - 工作在 **模式 3（方波）**，输出 **10μs 周期**（50% 占空比）。
2. **Counter 1**：
   - 工作在 **模式 1（单稳）**，输出 **1ms 负脉冲**（低电平有效）。
   - 同时 **OUT1 接 CLK2**，作为 Counter2 的时钟输入。
3. **Counter 2**：
   - 工作在 **模式 1（单稳）**，计数 **1000 次**（1ms × 1000 = 1s）。
   - 输出 **1s 负脉冲**。

4. **8255地址A380H~A383H，工作于方式0，A/B口输出，将PC4置0、PC7置1，画连接图并编写初始化程序。**
   答案：

1. 8255 基本连接图

- **芯片寻址**：地址范围 A380H~A383H，对应 8255 的 4 个端口：
  - A380H：A 口
  - A381H：B 口
  - A382H：C 口
  - A383H：控制寄存器
- **地址译码逻辑**：
  - 高位地址线 A15~A2 译码得到 A380H~A383H（假设 A15~A2=10100011100000b）。
  - 低位地址线 A1~A0 选择内部端口：
    - 00：A 口
    - 01：B 口
    - 10：C 口
    - 11：控制寄存器
- **控制信号连接**：
  - #CE（片选）：由地址译码器输出（如 A15~A2 译码）。
  - #RD、#WR：直接连接系统控制总线。
  - RESET：接系统复位信号。

2. 初始化程序（8086汇编）

```
.MODEL SMALL
.STACK 100H
.CODE
START:
    MOV DX, 0A383H      ; 控制寄存器地址
    MOV AL, 10000001B   ; 控制字：方式0，A/B口输出，C口高4位输出，低4位输出
    OUT DX, AL          ; 写入控制字

    ; 设置PC0=0, PC1=1
    MOV DX, 0A382H      ; C口地址
    MOV AL, 10000000B   ; PC4=0, PC7=1（其他位保持原状）
    OUT DX, AL

    ; 程序结束
    MOV AH, 4CH
    INT 21H
END START
```

3. 控制字解析

- **控制字格式**：

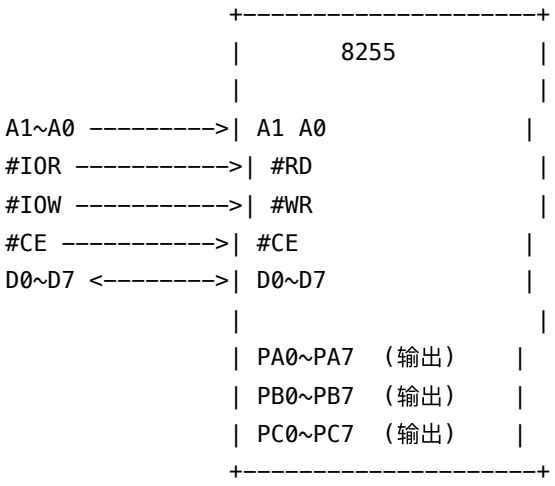
```
1 D6 D5 D4 D3 D2 D1 D0
| |  |  |  |  |  |  └─ C口低4位（0=输出，1=输入）
| |  |  |  |  |  └──── B口（0=输出，1=输入）
| |  |  |  |  └─────── B组方式（0=方式0，1=方式1）
| |  |  |  └────────── C口高4位（0=输出，1=输入）
| |  |  └───────────── A口（0=输出，1=输入）
| |  └──────────────── A组方式（00=方式0，01=方式1，1X=方式2）
| └─────────────────── 模式标志（1=有效）
└───────────────────── 必须为1
```

- **本例控制字**： `10000001B`
  - 方式0，A/B口输出，C口高4位输出，低4位输出。

4. PC4 和 PC7 设置

- 直接向 C 口写入 `10000000B` （PC4=0, PC7=1）。
- 注意：C 口是按位操作的，写入时会同时更新所有位的状态。

5. 电路连接示意图



此设计满足题目所有要求，代码简洁，硬件连接清晰。

5. **教材第七章设计题：根据图7-51接线，写8255、8253端口地址，初始化程序，及8255的I/O控制子程序。**

7.12　已知某 8088 微机系统的 I/O 接口电路框图如图 7-47 所示。试完成以下几点。

（1）根据图中接线，写出 8255、8253 各端口的地址。

（2）编写 8255 和 8253 的初始化程序。其中，8253 的 OUT_1 端输出 100Hz 方波，8255 的 A 口为输出，B 口和 C 口为输入。

（3）为 8255 编写一个 I/O 控制子程序，其功能为：每调用一次，先检测 PC_0 的状态，若 $PC_0=0$，则循环等待；若 $PC_0=1$，可从 PB 口读取当前开关 K 的位置（0～7），经转换计算从 A 口的 PA_0～PA_7 输出该位置的二进制编码，供 LED 显示。

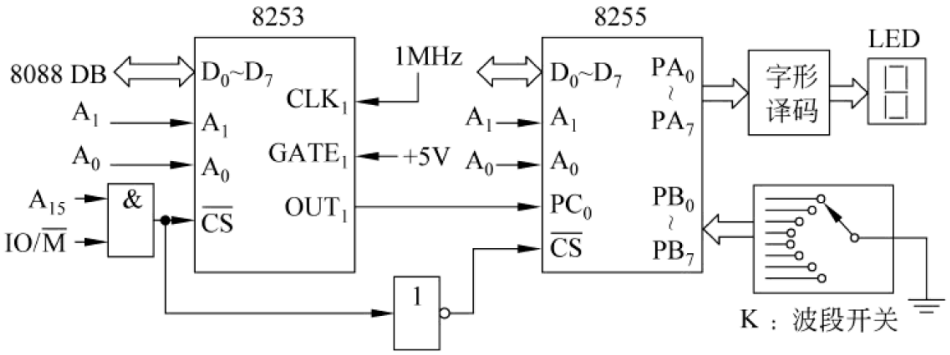


图 7-47　题 7.12 接口电路框图

答案：

(1) 根据图中接线，写出 8255、8253 各端口的地址。

地址译码分析:

1. **I/O操作选择**: 图中 `IO/M` 信号上有一横杠，表示这是8088的 `M/IO` 信号，在进行I/O操作时，该信号为低电平（L）。
2. **芯片选择 (CS)**:
   - 一个与门（AND gate）的输入为地址线 `A15` 和 `IO/M` 。我们通常假设这是一个I/O地址映射，因此 `M/IO` 信号为低电平。但是，AND门的输出直接连接到 `CS` (低电平有效) 是不合逻辑的。
   - 更合理的解释是， `IO/M` 信号（带反转条）表示 "进行I/O操作"，此时该信号为高电平（H）。这与8086/8088系统中I/O读写控制信号（如 `IOWC` ）的产生逻辑一致。
   - 在此假设下：当进行I/O操作时， `IO/M` =1。
     - **8253 的片选**: 与门的输出连接到 8253 的 `CS` 端（低电平有效）。要选中8253，与门的输出必须为0。由于 `IO/M` =1，则必须 `A15` =0。所以，**当 `A15=0` 时，8253被选中**。
     - **8255 的片选**: 与门的输出经过一个反相器连接到 8255 的 `CS` 端（低电平有效）。要选中8255，反相器的输出必须为0，这意味着与门的输出必须为1。由于 `IO/M` =1，则必须 `A15` =1。所以，**当 `A15=1` 时，8255被选中**。
3. **端口地址计算**:
   - 地址线 `A1` 和 `A0` 用于选择芯片内部的端口或寄存器。
   - 其他地址线（A2-A14）没有参与译码，为“无关位”（Don't Care）。为了简化，我们假设它们都为0。

地址计算结果:

- **8253 地址 (当 A15=0):**

    - 计数器0 (A1=0, A0=0): 地址为 `0000H`
    - 计数器1 (A1=0, A0=1): 地址为 `0001H`
    - 计数器2 (A1=1, A0=0): 地址为 `0002H`
    - 控制字寄存器 (A1=1, A0=1): 地址为 `0003H`
- **8255 地址 (当 A15=1):**
    - A口 (A1=0, A0=0): 地址为 `8000H`
    - B口 (A1=0, A0=1): 地址为 `8001H`
    - C口 (A1=1, A0=0): 地址为 `8002H`
    - 控制字寄存器 (A1=1, A0=1): 地址为 `8003H`

(2) 编写 8255 和 8253 的初始化程序。

初始化要求:

- **8253**: `OUT1` 端输出 100Hz 方波。 `CLK1` 输入为 1MHz。
- **8255**: A口为输出，B口和C口为输入。

8253 初始化编程:

1. **工作方式**: 输出方波，应选择方式3（Square Wave Generator）。
2. **计数器**: 使用计数器1（因为连接到 `OUT1` ）。
3. **计数初值**: `N = f_CLK / f_OUT = 1,000,000 Hz / 100 Hz = 10000` 。
    - 转换为十六进制： `10000 (D) = 2710H` 。
4. **控制字**:
    - `SC1 SC0`： `01` (选择计数器1)
    - `RL1 RL0`： `11` (先读/写低8位，后读/写高8位)
    - `M2 M1 M0`： `011` (方式3)
    - `BCD`： `0` (16位二进制计数)
    - 控制字为 `01110110B = 76H` 。

8255 初始化编程:

1. **工作方式**: 基本输入/输出，选择方式0。
2. **端口方向**: A口输出，B口输入，C口输入。
3. **控制字**:
    - D7: `1` (方式选择控制字)
    - D6 D5: `00` (A组，方式0)
    - D4: `0` (A口输出)
    - D3: `0` (C口高四位输入)
    - D2: `0` (B组，方式0)
    - D1: `1` (B口输入)
    - D0: `1` (C口低四位输入)
    - 控制字为 `10000011B = 83H` 。

汇编语言初始化程序:

```
; 假设段寄存器已正确设置
; --- 初始化 8253 ---
MOV AL, 76H          ; 8253 控制字: 选计数器1, 方式3, 16位二进制, 先低后高
OUT 0003H, AL        ; 写入8253控制字寄存器

MOV AX, 10000        ; 计数初值 10000 (或 MOV AX, 2710H)
OUT 0001H, AL        ; 向计数器1写入低8位
MOV AL, AH           ; 准备高8位
OUT 0001H, AL        ; 向计数器1写入高8位

; --- 初始化 8255 ---
MOV AL, 83H          ; 8255 控制字: A口输出, B/C口输入, 方式0
MOV DX, 8003H        ; 8255 控制字寄存器地址
OUT DX, AL           ; 写入8255控制字寄存器
```

(3) 为 8255 编写一个 I/O 控制子程序。

子程序功能:

1. 每调用一次，先检测 `PC0` 的状态。
2. 若 `PC0=0` ，则循环等待。
3. 若 `PC0=1` ，则从 PB 口读取开关 K 的位置（0~7）。
4. 将该位置（0-7的数值）转换为二进制码，从 A 口输出，供 LED 显示。

逻辑分析:

- `PC0` 由 8253 的 `OUT1` 驱动，是一个100Hz的方波。检测 `PC0=1` 相当于在方波的高电平期间执行操作。
- 开关K是一个波段开关，当它拨到某个位置（如位置 `i` ）， `PBi` 引脚被接地，变为低电平0，其他引脚为高电平1（假设有上拉）。
- 例如，位置0对应PB0=0，PB口读数为 `11111110B` (FEH)。
- 我们需要找到值为0的位，其位置索引（0-7）就是开关的位置。
- 将找到的位置索引（如 0, 1, 2, ..., 7）直接输出到A口。

I/O 控制子程序 (IO_CONTROL):

```
; 子程序名: IO_CONTROL
; 功能: 等待 PC0=1, 然后读取开关K位置, 并将位置号(0-7)输出到A口
IO_CONTROL PROC NEAR
WAIT_PC0:
    MOV  DX, 8002H      ; C口地址
    IN   AL, DX         ; 读C口状态
    TEST AL, 01H        ; 测试PC0位 (bit 0)
    JZ   WAIT_PC0       ; 如果PC0=0 (结果为0), 则跳转继续等待

; PC0=1, 开始读取开关位置
    MOV  DX, 8001H      ; B口地址
    IN   AL, DX         ; 从B口读取开关状态
    MOV  CX, 8          ; 设置循环次数为8
    MOV  BL, 0          ; 用BL寄存器存放开关位置号, 初始为0

FIND_POS_LOOP:
    SHR  AL, 1          ; 将AL右移一位, 最低位移入进位标志CF
    JNC  FOUND_POS      ; 如果CF=0, 说明该位是0, 找到了位置, 跳转
    INC  BL             ; 如果CF=1, 说明该位是1, 位置号加1
    LOOP FIND_POS_LOOP  ; 循环继续查找
    ; 如果8次循环后仍未找到0 (例如开关悬空), 可在此添加错误处理
    ; 这里假设开关总有一个位置被选中

FOUND_POS:
    MOV  AL, BL         ; 将找到的位置号(0-7)放入AL
    MOV  DX, 8000H      ; A口地址
    OUT  DX, AL      ; 将位置号从A口输出到LED

    RET                 ; 子程序返回
IO_CONTROL ENDP
```