

第1章 微型机的基本知识

1. 微型机：由 CPU、存储器、输入/输出接口电路和系统总线构成。
2. 系统总线：地址总线 AB，控制总线 CB 和双向数据总线 DB。

数据总线 (Data Bus—DB) 双向、三态：数据总线的根数决定了一次可以传递二进制数的位数。

地址总线 (Address Bus—AB) 单向、三态：地址总线的位数决定了可以直接访问的存储单元(或 I/O 口)的最大可能数量 (即容量)。

控制总线 (Control Bus—CB)：控制总线用来传输控制信号，数据总线和每个元件的数据线相连，为了使 CPU 能够和其中一个元件正确通信，必须使用三态逻辑元件 (特别针对输入元件)。

3. 运算器：运算器由算术逻辑单元 ALU、累加器 A、标志寄存器 F 和寄存器组，相互之间通过内部总线连接而成。
4. 控制器：控制器 CU 由程序计数器 PC 、指令寄存器 IR 、指令译码器 ID 、操作控制部件或称为组合逻辑阵列 PLA 和时序发生器等电路组成，是发布操作命令的“决策机构”。
5. 特殊功能寄存器(PSW)：

CY(PSW.7)进位/借位标志位；

AC(PSW.6)半进位/借位标志位；

F0 (PSW.5)可由用户定义的标志位；

RS1(PSW.4)、RS0(PSW.3)工作寄存器组选择位；

OV (PSW.2)溢出标志位； P (PSW.0)奇偶标志位。

6. 堆栈与堆栈指示器 SP (Stack Pointer)：堆栈是按照“先进后出”或“后进先出”原则组织的一个存储区域。
7. 字长：字长就是计算机的运算器一次可处理（运算、存取）二进制数的位数。字长越长，一个字能表示数值的有效位就越多，计算精度也就越高，速度就越块。
 - 8 位二进制数称为 1 个字节，以 B (Byte) 表示；
 - 2 个字节定义为 1 个字，以 W (Word) 表示；
 - 32 位二进制数就定义为双字，以 DW (Double word) 表示。
8. 存储容量：存储器存储二进制信息多少的一个技术指标
 - $1KB = 1024B$ (即 1024×8 位)
 - $1024KB = 1MB$ (兆字节)
 - $1024MB = 1GB$ (千兆字节)

第2章 单片机硬件结构

1. 单片机主要功能特点：

8 位的 CPU，片内有振荡器和时钟电路，工作频率为 $1\sim12MHz$

片内有 128B 数据存储器 RAM

片内有 4KB 程序存储器 ROM

可寻址片外 64K 字节数据存储器 RAM

可寻址片外 64K 字节程序存储器 ROM

片内 21/26 个特殊功能寄存器 (SFR)

4 个 8 位的并行 I/O 口 (PIO)

1 个全双工串行口 (SIO/UART)

2 个 16 位定时器/计数器 (TIMER/COUNTER)

2 个优先级可处理 5 个中断源，两级中断优先级

1 个布尔处理器和 1 个布尔累加器 (Cy)

MCS-51 指令集含 111 条指令

2. 时钟电路：XTAL1, XTAL2：片内振荡电路输入/输出端；通常外接一个晶振两个电容。
3. 控制信号：RST/V_{PD} (9 脚) 复位信号，复位使单片机进入某种确定的初始状态。MCS-51 通常采用上电自动复位 (正脉冲保持约 10ms 以上) 和开关复位 (输出高电平) 两种形式。
4. ALE/PROG (30 脚) 地址锁存信号，ALE 高电平期间，P0 口上一般出现地址信息，在 ALE 下降沿时，将 P0 口上的地址信息锁存到片外地址锁存器，在 ALE 低电平期间 P0 口上一般出现指令和数据信息。
5. PSEN (29 脚) 片外程序存储器读选通

6. /Vpp (31脚) 当访问内部程序存储器时, 保持高电平; 当接低电平时, 只访问片外程序存储器
7. P1.0—P1.7: 准双向 I/O 口 (内置了上拉电阻); 准双向: 在作输入口用时要先对其写“1”。
8. P0.0—P0.7: 双向 I/O (内置场效应管上拉); 寻址外部程序存储器时分时作为双向 8 位数据口和输出低 8 位地址复用口; 不接外部程序存储器时可作为 8 位准双向 I/O 口使用。
9. P2.0—P2.7: 双向 I/O (内置了上拉电阻); 寻址外部程序存储器时输出高 8 位地址; 不接外部程序存储器时可作为 8 位准双向 I/O 口使用。
10. P3.0—P3.7: 双功能口 (内置了上拉电阻); 它具有特定的第二功能。在不使用它的第二功能时它就是普通的通用准双向 I/O 口。
11. MCS-51 的存储器组织分为 4 个存储空间: 片内程序存储器和片外扩展的程序存储器, 片内数据存储器和片外扩展的数据存储器。
 - 片内 RAM 128 字节 (00H—7FH);
 - 片内 RAM 前 32 个单元是工作寄存器区(00H—1FH)
 - 片内 RAM 有 128 个可按位寻址的位, 占 16 个单元。位地址编号为: 00H—7FH, 分布在 20H—2FH 单元
 - 片内 21 个特殊功能寄存器(SFR)中: 地址号能被 8 整除的 SFR 中的各位也可按位寻址
 - 可寻址片外 RAM 64K 字节 (0000H—FFFFH)
 - 可寻址片外 ROM 64K 字节 (0000H—FFFFH)
 - 片内 Flash ROM 4K 字节 (000H—FFFH)

12. **MCS-51 的内部 RAM** 可以分成三个物理空间, 分别是工作寄存器区、位寻址区和数据缓冲区。

13. 专用寄存器: MCS-51 共有 23 个特殊功能寄存器 (3 个属于 8032/8052), 其中 5 个是双字节寄存器, 程序计数器 PC 在物理上是独立的, 其余 22 个寄存器都属于片内数据存储器 SFR 块, 共占 26 个字节。

程序计数器 PC (16bit): 用于存放下一条要执行指令的地址

累加器 A: 最常用的专用寄存器

寄存器 B: 是一个 8 位寄存器

程序状态字 PSW: 是一个 8 位寄存器, 用来存放程序的状态信息, 表征指令的执行状态, 供程序查询和判别之用。

堆栈指针 SP: 是一个 8 位寄存器, 用来存放栈顶地址

数据指针 DPTR: 是一个 16 位专用寄存器, 高字节寄存器用 DPH 表示, 低字节寄存器用 DPL 表示

I/O 端口 P0~P3: I/O 端口 P0~P3 的锁存器

定时器/计数器

14. MCS-51 外部数据存储器寻址空间为 64KB。

15. 1 个机器周期=6 个状态=12 个振荡周期

第 3 章 MCS-51 指令系统

1. 指令中所用符号说明

A——累加器

B——专用寄存器

C——进位或借位标志, 或布尔处理机中的累加器

#data —— 8 位立即数

#data16 —— 16 位立即数

direct —— 8 位直接地址

@Ri—— R0 或 R1, 可以间接寻址

@DPTR —— 可按 DPTR 中地址对外部存储器寻址

bit —— 8 位位地址

addr11 —— 11 位目标地址

addr16 —— 16 位目标地址

rel—— 8 位地址偏移量

\$—— 当前指令的地址

2. 七种寻址方式:

立即寻址: 指令直接给出一个操作数, 它紧跟在操作码后, 通常称它为立即数 (8 位或 16 位)

特点: 指令码中含有操作数本身

直接寻址：指令直接给出操作数所在存储单元的地址，它紧跟在操作码后。访问专用寄存器的唯一方式

特点：指令码中含有操作数地址，机器根据该地址寻找操作数

寄存器寻址：指令选定的寄存器内容就是实际操作数

特点：指令码中含有操作数地址所在的寄存器号，根据该寄存器号可以找到操作数

寄存器间接寻址：指令所选中的寄存器内容是实际操作数地址（而不是操作数）

特点：地址的地址

变址寻址(基址寄存器+变址寄存器间接寻址)：此种寻址方式以 DPTR 或 PC 作基址寄存器，A 作变址寄存器（存放 8 位无符号数），两者相加形成 16 位程序存储器地址作操作数地址

特点：

操作数=基地址+地址偏移量

指令码内隐含存放基地址的寄存器（DPTR/PC）号

DPTR/PC 中的基地址常常是表格的起始地址，累加器中存放偏移量 rel

相对寻址：此种寻址方式以程序计数器 PC 的内容为基地址，加上指令中给出的偏移量 rel，所得结果为转移目标地址，用于转移指令。偏移量 rel 是一 8 位有符号补码数，范围-128～+127。所以转移范围应在当前 PC 指针的-128～+127 之间某一程序存储器地址中

特点：

操作码中含有相对地址偏移量 rel

目标地址=源地址+转移指令字节数 rel

源地址=相对转移指令的始址

位寻址：以访问触发器的方式对内部 RAM、特殊功能寄存器 SFR 中的位地址空间进行访问称之为位寻址

特点：

操作数是 8 位二进制数中的某位

指令码中含有位地址

位地址用 bit 表示

3. 数据传送指令：

内部存储器间传送：MOV

特点：源操作数在内部，目的操作数也在内部。

分四类介绍如下

以累加器 A 为目的字节的传送指令（4 条）

(1) 立即数送累加器

`MOV A, #data ;A ← #data`

(2) 寄存器内容送累加器

`MOV A, Rn ;A ← (Rn) (n=0~7)`

(3) 内部 RAM 或 SFR 内容送累加器

`MOV A, direct ;A ← (direct)`

(4) 内部 RAM 内容送累加器

`MOV A, @Ri ;A ← ((Ri)) (i=0, 1)`

以 Rn 为目的字节的传送指令（3 条）

(1) 立即数送寄存器

`MOV Rn, #data ;(Rn) ← #data (n=0~7)`

(2) 累加器内容送寄存器

`MOV Rn, A ;Rn ← (A) (n=0~7)`

(3) 内部 RAM 或 SFR 内容送寄存器

`MOV Rn, direct ;Rn ← (direct) (n=0~7)`

以直接地址为目的字节的传送指令（5 条）

(1) 立即数送内部 RAM 或 SFR

`MOV direct, #data ;direct ← #data`

(2) 累加器内容送内部 RAM 或 SFR

`MOV direct, A ;direct ← (A)`

(3) 寄存器内容送内部 RAM 或 SFR

MOV direct , Rn ; direct ← (Rn) (n=0~7)

(4) 内部 RAM 或 SFR 之间直接传送

MOV direct1, direct2 ; direct1 ← (direct2)

(5) 内部 RAM 内容送内部 RAM 或 SFR

MOV direct, @Ri ; direct ← ((Ri)) (i=0, 1)

以间接地址为目的字节的传送指令 (4 条)

(1) 立即数送内部 RAM

MOV @Ri , #data ; (Ri) ← #data (i=0, 1)

(2) 累加器内容送内部 RAM

MOV @Ri , A ; (Ri) ← (A) (i=0, 1)

(3) 内部 RAM 或 SFR 内容送内部 RAM

MOV @Ri , direct ; (Ri) ← (direct) (i=0, 1)

(4) 16 位立即数传送指令

MOV DPTR , #data16 ; DPTR ← #data16

; DPH ← #dataH

; DPL ← #dataL

外部数据存储器与累加器间传送: MOVX

格式:

MOVX A, @DPTR ; A ← ((DPTR))

MOVX @DPTR, A ; (DPTR) ← (A)

MOVX A, @Ri ; A ← ((Ri)) (i=0, 1)

MOVX @Ri , A ; A ← ((Ri)) (i=0, 1)

特点:

单字节指令

第 1、2 条指令可以在外部 RAM 64K 范围寻址;

第 3、4 条指令可以对外部 RAM 的第 0 页寻址;

寻址范围是 0000H~0FFFH。

程序存储器向累加器传送: MOVC

格式:

MOVC A, @A+DPTR ; A ← ((A)+(DPTR))

MOVC A, @A+PC ; PC ← (PC)+1

; A ← ((A)+(PC))

特点:

这两条指令为单字节查表指令

DPTR、PC 中存放被查表的始址

功能: 根据 A 中项数, 查得表中对应值送入累加器 A 中。

数据交换: XCH, XCHD, SWAP

格式:

XCH A, Rn ; (A)↔(Rn) (n=0~7)

XCH A, direct ; (A)↔(direct)

XCH A, @Ri ; (A)↔((Ri)) (i=0, 1)

XCHD A, @Ri ; (A3~0)↔((Ri)3~0) (i=0, 1)

SWAP A ; (A3~0)↔(A7~4)

特点: 前三条指令执行后会影响 PSW 中的 P 标志 (奇偶标志)。XCHD 使用该指令前应预先给 Ri 中置地址。

堆栈操作: PUSH, POP

格式:

PUSH direct ; SP ← (SP)+1

; (SP) ← (direct)

```
POP direct ; direct ← ((SP))
; SP ← (SP)-1
```

特点：Rn 和 A 不能直接用在本指令中，A 必须用 ACC（或 EOH），Rn 也要用它的物理地址。

4. 算术运算指令

不带进位加：ADD

格式：

```
ADD A, #data ; A ← (A)+#data
ADD A, Rn ; A ← (A)+(Rn) (n=0~7)
ADD A, @Ri ; A ← (A)+((Ri)) (i=0, 1)
ADD A, direct ; A ← (A)+(direct)
```

特点：两个操作数均为 8 位，其中之一是累加器 A；加法操作在 ALU 中完成，结果送回累加器，运算时产生的标志位在 PSW 中；不论两操作数是否为带符号数，机器均按带符号数运算。

带进位加：ADDC

格式：

```
ADDC A, #data ; A ← (A)+#data+Cy
ADDC A, Rn ; A ← (A)+(Rn)+Cy (n=0~7)
ADDC A, @Ri ; A ← (A)+((Ri))+Cy (i=0, 1)
ADDC A, direct ; A ← (A)+(direct)+Cy
```

特点：注释中的 Cy 中内容为指令执行前 Cy 内容，指令执行后形成新的 Cy 保留在 PSW 中，其余和不带 Cy 的加法指令相同；本类指令用于多字节加法程序中。

增量指令：INC

格式：

```
INC A ; A ← (A)+1
INC Rn ; Rn ← (Rn)+1 (n=0~7)
INC direct ; direct ← (direct)+1
INC @Ri ; (Rn) ← ((Rn))+1 (n=0, 1)
INC DPTR ; DPTR ← (DPTR)+1
```

特点：INC 指令仅对 PSW 中 P 标志有影响。

十进制调整指令：DA

格式：

```
DA A
```

操作：

若累加器低 4 位大于 9 或 BCD 码加时有半进位 AC=1，则 A ← (A)+06H

若累加器高 4 位大于 9 或 BCD 码加时有全进位 CY=1，则 A ← (A)+60H。

特点：DA 紧跟加法指令后，对加法结果调整；用于 BCD 加、BCD 减运算。

减法指令：SUBB

格式：

```
SUBB A, #data ; A ← (A)-#data-Cy
SUBB A, Rn ; A ← (A)-(Rn)-Cy (n=0~7)
SUBB A, direct ; A ← (A) - (direct) - Cy
SUBB A, @Ri ; A ← (A)-((Ri))-Cy (n=0, 1)
```

特点：SUBB 是 Subtraction Borrow 的缩写；本指令用于单字节、多字节减法程序；机器按带符号数运算，并产生 PSW 中标志；形成 OV 标志的规则为：

正数-正数 不会溢出，故 OV=0；

负数-负数 不会溢出，故 OV=0；

正数-负数 若差为负，则 OV=1；

负数-正数 若差为正，则 OV=1。

减量指令：DEC

格式：

```

DEC A ; A ← (A)-1
DEC Rn ; Rn ← (Rn)-1 (n=0~7)
DEC direct ; direct ← (direct) - 1
DEC @Ri ; (Ri) ← ((Ri))-1 (n=0, 1)

```

特点：DEC是Decrease的缩写；DEC A指对PSW中P标志有影响。

功能：使源地址所对应的RAM单元中内容减1。

乘法指令：MUL

除法指令：DIV

格式：

```

MUL AB ; B A ← (A)×(B)
          ; Cy←0
DIV A B ; A ← (A)/(B)的商;
          ; B ← (A)/(B)的余数;
          ; Cy ← 0, OV ← 0

```

特点：MUL是Multiply的缩写，DIV是Divide的缩写；单字节指令，执行时间为48T（4机器周期）；操作数均为8位无符号数。

5. 逻辑运算指令

逻辑与指令：ANL

格式：

```

ANL A, #data ; A ← (A)∧#data
ANL A, Rn ; A ← (A)∧(Rn) (n=0~7)
ANL A, direct ; A ← (A)∧(direct)
ANL A, @Ri ; A←(A)∧((Ri)) (n=0, 1)
ANL direct, A ; direct ← (direct)∧(A)
ANL direct, #data ; direct ← (direct)∧#data

```

特点：ANL是AND Logical的缩写；前四条指令执行后会对PSW中P标志产生影响。

用途：可从某字节中取出某几位。

逻辑或指令：ORL

格式：

```

ORL A, #data ; A ← (A) ∨ #data
ORL A, Rn ; A ← (A) ∨ (Rn) (n=0~7)
ORL A, direct ; A ← (A) ∨ (direct)
ORL A, @Ri ; A←(A) ∨ ((Ri)) (n=0, 1)
ORL direct, A ; direct ← (direct) ∨ (A)
ORL direct, #data ; direct ← (direct) ∨ #data

```

特点：ORL是OR Logical的缩写；前四条指令执行后会对PSW中P标志产生影响。

用途：给某字节中某几位置1。

逻辑异或指令：XRL

格式：

```

XRL A, #data ; A ← (A)⊕#data
XRL A, Rn ; A ← (A)⊕(Rn) (n=0~7)
XRL A, direct ; A ← (A)⊕(direct)
XRL A, @Ri ; A ← (A)⊕((Ri)) (n=0, 1)
XRL direct, A ; direct ← (direct)⊕(A)
XRL direct, #data ; direct ← (direct)⊕#data

```

特点：XRL是XOR Logical的缩写；前四条指令执行后会对PSW中P标志产生影响。

用途：令某字节中某几位置取反，其余位不变。

累加器清除与求反指令：CLR、CPL

格式：

```
CLR A      ; A ← 0
CPL A      ; A ←  $\neg$ 
```

特点：CLR是Clearing的缩写；CPL是Complement of one's的缩写；CLR A指令执行后，Cy=0。

用途：CLR用于累加器清零；CPL可用于求某数的补码。

移位指令：RL、RLC、RR、RRC

格式：

```
RL A
RLC A
RR A
RRC A
```

特点：RL是Rotate Left的缩写，RR是Rotate Right的缩写；第2、4条指令对PSW中的Cy和P标志产生影响。

用途：对某数扩大或缩小1倍。

6. 控制转移指令

无条件转移指令：LJMP(长转移)、AJMP(绝对转移)、SJMP(短转移)、JMP(间接长转移)；程序计数器PC

格式：

```
LJMP addr16      ; PC ← addr16
AJMP addr11      ; PC ← (PC)+2
                  ; PC10~0 ← 指令中的A10~0
SJMP rel         ; PC ← (PC)+2
                  ; PC ← (PC)+rel
JMP @A+DPTR     ; PC ← (A)+(DPTR)
```

特点：这类指令执行时不会影响PSW中各标志位；指令的操作对象是PC中地址；第2、3条指令是相对转移指令，汇编时能产生浮动代码，在子程序中很有用。

条件转移指令：JZ、JNZ、CJNE、DJNZ

累加器A判零转移指令

格式：

```
JZ rel        ; 若(A)≠0, 则 PC ← (PC)+2
              ; 若(A)=0, 则 PC ← (PC)+2+rel
JNZ rel       ; 若(A)=0, 则 PC ← (PC)+2
              ; 若(A)≠0, 则 PC ← (PC)+2+rel
```

特点：双字节指令，第一字节是操作码，第二字节rel是一个带符号数；rel在编程时采用符号地址，汇编时翻译成内存地址。

比较条件转移指令

格式：不等则转移

```
CJNE A, #data, rel
CJNE Rn, #data, rel    (n=0~7)
CJNE @Ri, #data, rel    (i=0, 1)
CJNE A, direct, rel
```

特点：三字节指令，第3字节是rel，转移范围为-125~+130；若A、B为带符号数，则Cy的形成原则是：

若A>=B，则Cy=0；

若A<B，则Cy=1。

减1条件转移指令

格式：内容减1不等于零则转移

```
DJNZ Rn, rel    ; Rn ← (Rn)-1    (n=0~7)
                  ; 若(Rn)=0, 则 PC ← (PC)+2
                  ; 若(Rn)≠0, 则 PC ← (PC)+2+rel
DJNZ direct, rel ; direct ← (direct)-1
                  ; 若(direct)=0, 则 PC ← (PC)+3
                  ; 若(direct)≠0, 则 PC ← (PC)+3+rel
```

特点：在 DJNZ 中，D 是 Decrease，J 是 Jump，N 是 Not，Z 是 Zero；本类指令不影响 PSW 标志。

用途：判断循环是否结束。

子程序调用及返回指令：LCALL(长调用)、ACALL(绝对调用)、RET(子程序返回)、RETI(中断返回)

格式：

```

LCALL  addr16      ; PC ← (PC)+3
                    ; SP ← (SP)+1, (SP) ← PC7~0
                    ; SP ← (SP)+1, (SP) ← PC15~8
                    ; PC ← 指令中 addr16

ACALL  addr11      ; PC ← (PC)+2
                    ; SP ← (SP)+1, (SP) ← PC7~0
                    ; SP ← (SP)+1, (SP) ← PC15~8
                    ; PC10~0 ← 指令中 A10~0

RET                 ; PC15~8 ← ((SP)), SP ← (SP)-1
                    ; PC7~0 ← ((SP)), SP ← (SP)-1

RETI                ; PC15~8 ← ((SP)), SP ← (SP)-1
                    ; PC7~0 ← ((SP)), SP ← (SP)-1

```

空操作指令：NOP

格式：

```
NOP      ; PC ← (PC)+1
```

代码：00000000 00H

说明：该指令为单字节指令，其操作使程序计数器 PC 加“1”，在时间上消耗 12 个时钟周期，可用于延时，等待或用于修改程序保留空间等情况。

7. 布尔处理类指令

布尔传送指令：MOV

格式：

```

MOV C, bit      ; Cy ← (bit)
MOV bit, C      ; bit ← (Cy)

```

功能：被传送的不是字节，而是字节中的某位。

说明：

bit 是位地址（二进制 8 位）；

bit 和 bit 之间无直接传送指令。

布尔状态控制指令：CLR(清除)、SETB(置位)、CPL(取反)

格式：

```

CLR C      ; Cy ← 0
CLR bit    ; bit ← 0
SETB C     ; Cy ← 1
SETB bit   ; bit ← 1
CPL C      ; Cy ←  $\overline{Cy}$ 
CPL bit    ; bit ←  $\overline{bit}$ 

```

功能：CLR 是 Clear 缩写，SETB 是 Set Bit 的缩写。

布尔逻辑运算指令：ANL、ORL

格式：

```

ANL C, bit  ; Cy ← (Cy)  $\wedge$  (bit)
ANL C,  $\overline{bit}$  ; Cy ← (Cy)  $\wedge$   $\overline{bit}$ 
ORL C, bit  ; Cy ← (Cy)  $\vee$  (bit)
ORL C,  $\overline{bit}$  ; Cy ← (Cy)  $\vee$   $\overline{bit}$ 

```

布尔条件转移指令：JC、JNC、JB、JNB、JBC

格式：

```
JC rel      ; 若(Cy)=1，则 PC ← (PC)+2+rel
```

```

;若(Cy)=0, 则 PC ← (PC)+2
JNC rel ;若(Cy)=0, 则 PC ← (PC)+2+rel
;若(Cy)=1, 则 PC ← (PC)+2
JB bit, rel ;若(bit)=1, 则 PC ← (PC)+3+rel
;若(bit)=0, 则 PC ← (PC)+3
JB bit, rel ;若(bit)=0, 则 PC ← (PC)+3+rel
;若(bit)=1, 则 PC ← (PC)+3
JBC bit, rel ;若(bit)=0, 则 PC ← (PC)+3
;若(bit)=1, 则 PC ← (PC)+3+rel, 且 bit←0

```

第4章 汇编语言程序设计

1. 假设两个双字节无符号数, 分别存放在 R1R0 和 R3R2 中, 高字节在前, 低字节在后。编程使两数相加, 和数存放回 R2R1R0 中。

```

ORG 1000H
CLR C
MOV A, R0 ; 取被加数低字节至 A
ADD A, R2 ; 与加数低字节相加
MOV R0, A ; 存和数低字节
MOV A, R1 ; 取被加数高字节至 A
ADDC A, R3 ; 与加数高字节相加
MOV R1, A ; 存和数高字节
MOV A, #0
ADDC A, #0 ; 加进位位
MOV R2, A ; 存和数进位位
*SJMP $ ; 原地踏步
END

```

2. 将一个字节内的两个 BCD 码拆开并转换成 ASCII 码, 存入两个 RAM 单元。设两个 BCD 码已存放在内部 RAM 的 20H 单元, 将转换后的高半字节存放到 21H 中, 低半字节存放到 22H 中。

法一编程: ORG 1000H

```

MOV R0, #22H ; R0←22H
MOV @R0, #0 ; 22H←0
MOV A, 20H ; 两个 BCD 数送 A
XCHD A, @R0 ; BCDL 送 22H 单元
ORL 22H, #30H ; 完成转换
SWAP A ; BCDH 至 A 的低四位
ORL A, #30H ; 完成转换
MOV 21H, A ; 存数
SJMP $
END

```

法二编程: ORG 1000H

```

MOV A, 20H ; 取 BCD 码至 A
MOV B, #10H
DIV AB ; 除 10H 取余, 使 BCDH→A 、 BCDL→B
ORL B, #30H ; 完成转换
MOV 22H, B ; 存 ASCII 码
ORL A, #30H ; 完成转换
MOV 21H, A ; 存 ASCII 码
SJMP $

```

END

```

法三编程: ORG 1000H
    MOV A, 20H      ; 取 BCD 码
    ANL A, #0FH     ; 屏蔽高四位
    ORL A, #30H     ; 完成转换
    MOV 22H, A      ; 存 ASCII 码
    MOV A, 20H      ; 取 BCD 码
    ANL A, #0FOH    ; 屏蔽低四位
    SWAP A          ; 交换至低四位
    ORL A, #30H     ; 完成转换
    MOV 21H, A      ; 存 ASCII 码
    SJMP $          ; 跳转到起始地址
END

```

3. 双字节数求补, 设两个字节原码数存在 R1R0 中, 求补后结果存在 R3R2 中。

解: 求补采用“模-原码”的方法, 因为补码是原码相对于模而言的, 对于双字节数来说其模为 10000H。

编程: ORG 1000H

```

CLR C          ; 0→CY
CLR A          ; 0→A
SUBB A, R0     ; 低字节求补
MOV R2, A      ; 送 R2
CLR A          ; 0→A
SUBB A, R1     ; 高字节求补
MOV R3, A      ; 送 R3
SJMP $          ; 跳转到起始地址
END

```

4. 将内部 RAM 的 20H 单元中的 8 位无符号二进制数转换为三位 BCD 码, 并将结果存放在 FIRST(百位)和 SECOND(十位、个位)两单元中。

解: 可将被转换数除以 100, 得百位数; 余数再除以 10 得十位数; 最后余数即为个位数。

```

FIRST DATA 22H
SECOND DATA 21H
ORG 1000H
HBCD:  MOV A, 20H      ; 取数
        MOV B, #64H     ; 除数 100→B
        DIV AB         ; 除 100
        MOV FIRST, A   ; 百位 BCD
        MOV A, B
        MOV B, #0AH     ; 除数 10→B
        DIV AB         ; 除 10
        SWAP A          ; 十位数送高位
        ORL A, B        ; A 为(十位、个位)BCD
        MOV SECOND, A   ; 存十位、个位数
        SJMP $          ; 跳转到起始地址
END

```

5. 设内部 RAM 30H, 31H 单元中存放两个无符号数, 试比较它们的大小。将较小的数存放在 30H 单元, 较大的数存放在 31H 单元中。

```

ORG 1000H
START: CLR C          ; 0→CY
       MOV A, 30H

```

```

SUBB A, 31H      ; 做减法比较两数
JC    NEXT       ; 若(30H)小，则转移
MOV   A, 30H
XCH   A, 31H
MOV   30H, A     ; 交换两数
NEXT: NOP
SJMP $           ; 原地踏步
END

```

6. 空调机在制冷时，若排出空气比吸入空气温度低 8°C ，则认为工作正常，否则认为工作故障，并设置故障标志。设内存单元 40H 存放吸入空气温度值， 41H 存放排出空气温度值。若 $(40\text{H})-(41\text{H}) \geq 8^{\circ}\text{C}$ ，则空调机制冷正常，在 42H 单元中存放“0”，否则在 42H 单元中存放“FFH”，以示故障（在此 42H 单元被设定为故障标志）。

```

编程: ORG 1000H
START: MOV A, 40H      ; 吸入温度值送 A
        CLR C          ; 0→CY
        SUBB A, 41H     ; (40H)-(41H)→A
        JC  ERROR       ; CY=1，则故障
        SUBB A, #8       ; 温差小于  $8^{\circ}\text{C}$ ?
        JC  ERROR       ; 是则故障
        MOV 42H, #0      ; 工作正常
        SJMP EXIT       ; 转出口
ERROR: MOV 42H, #0FFH   ; 否则置故障标志
EXIT: SJMP $           ; 原地踏步
END

```

实验程序：

P1 口循环亮灯。
P1 口全亮，延时，全灭，循环。

```

ORG 8000H
LJMP Main
ORG 80F0H

```

Main:

```

MOV R7, #0
Loop:
MOV R6, #0
DJNZ R6, $
DJNZ R6, $
DJNZ R6, $
DJNZ R6, $
DJNZ R7, Loop
CPL P1.0      ; P1.0 取反
CPL P1.1      ; P1.1 取反
CPL P1.2      ; P1.2 取反
CPL P1.3      ; P1.3 取反
CPL P1.4      ; P1.4 取反
CPL P1.5      ; P1.5 取反
CPL P1.6      ; P1.6 取反
CPL P1.7      ; P1.7 取反
SJMP Main
END

```

P1 口走马灯形式亮灯。

ORG 8000H

LJMP Main

ORG 8100H

Main:

MOV A,#0FFH

CLR C

MainLoop:

CALL Delay

RLC A

MOV P1,A

SJMP MainLoop

Delay:

MOV R7, #0

Loop:

MOV R6, #0

DJNZ R6, \$

DJNZ R6, \$

DJNZ R6, \$

DJNZ R7, Loop

RET

END

P1 口由 P1.7 开关控制灯亮灭。

ORG 8000H

LJMP Main

ORG 8100H

Main:

JB P1.7,SETLED

CLRLED:

CLR P1.0

CLR P1.1

CLR P1.2

CLR P1.3

CLR P1.4

CLR P1.5

CLR P1.6

SJMP Main

SETLED:

SETB P1.0

SETB P1.1

SETB P1.2

SETB P1.3

SETB P1.4

SETB P1.5

SETB P1.6

SJMP Main

END

双字节数加法

内部存储器在 30H-----3FH 中有 8 个双字节二进制数, 30H(高), 31H(低)为第一个, 32H(高), 33H(低)为第二个……3EH(高), 3FH (低) 为第八个。

求其平均值放 40H (高), 41H (低), 并将其转换成十进制, 以压缩 BCD 码形式放 50H (高), 51H, 52H。

ORG 8000H

MAIN: MOV R0, #3FH
MOV R7, #08H
MOV 40H,#00H
MOV 41H,#00H
MOV 42H,#00H

LOOP: MOV A,@R0
ADD A,42H
MOV 42H,A
DEC R0
MOV A,@R0
ADDC A,41H
MOV 41H,A
MOV A,40H
ADDC A,#00H
MOV 40H,A

DEC R0

DJNZ R7, LOOP

MOV R7,#08H

LOOP1: CLR C
MOV A,40H
RRC A
MOV 40,A
MOV A,41H
RRC A
MOV 41H,A
MOV A,42H
RRC A
MOV 42H,A
DJNZ R7,LOOP1
MOV 40H,41H
MOV 41H,42H

CLR A

MOV 50H,A
MOV 51H,A
MOV 52H,A
MOV R7,#10H

IBTL2: CLR C
MOV A,41H
RLC A
MOV 41H,A
MOV A,40H
RLC A
MOV 40H,A
MOV A,52H
ADDC A,52H
DA A
MOV 52H,A

```

MOV A,51H
ADDC A,51H
DA A
MOV 51H,A
MOV A,50H
ADDC A,50H
DA A
MOV 50H,A
DJNZ R7,IBTL2
SJMP $
END

```

寻找最大最小数

在 30H-----3FH 共 16 个二进制数中，寻找一个最大数放 R7，寻找一个最小数放 R6. (编在同一个程序中)。

```

ORG 0000H
MAIN: MOV R5, #10H
      MOV R0, #30H
      MOV R7, #00H
LOOP:  MOV A, @R0
      CJNE A, 07H, LOOP1
LOOP3: INC R0
      DJNZ R5,LOOP
      SJMP MA2
LOOP1: JC LOOP3
      MOV R7,A
      SJMP LOOP3
MA2:  MOV R5, #10H
      MOV R0, #30H
      MOV R6, #0FFH
LOP:   MOV A, @R0
      CJNE A, 06H, LOP1
LOP3:  INC R0
      DJNZ R5,LOP
      SJMP $
LOP1:  JNC LOP3
      MOV R6,A
      SJMP LOP3
END

```

蜂鸣器驱动

编制一段程序，用 P1.3 口控制，使蜂鸣器发出“生日快乐”的音乐。

```

ORG 8000H
JMP MAIN
ORG 800BH
JMP INTTO
ORG 8100H
MAIN: MOV SP,#60H
      MOV TMOD,#01H
      SETB ET0
      SETB EA
      SETB TR0

```

```

START0: SETB    P1.3
        MOV     30H,#00H
NEXT:   MOV     A,30H
        MOV     DPTR,#TABLE
        MOVC   A,@A+DPTR
        MOV     R2,A
        JZ    ENDD
        ANL     A,#0FH
        MOV     R5,A
        MOV     A,R2
        SWAP   A
        ANL     A,#0FH
        JNZ    SING
        CLR     TR0
        JMP    D1
SING:  DEC     A
        MOV     22H,A
        RL    A
        MOV     DPTR,#TABLE1
        MOVC   A,@A+DPTR
        MOV     TH0,A
        MOV     21H,A
        MOV     A,22H
        RL    A
        INC     A
        MOVC   A,@A+DPTR
        MOV     TL0,A
        MOV     20H,A
        SETB   TR0
D1:    CALL   DELAY
        INC     30H
        JMP    NEXT
ENDD:  CLR     TR0
        JMP    START0
INTT0:
        PUSH   PSW
        PUSH   ACC
        MOV    TL0,20H
        MOV    TH0,21H
        CPL    P1.3
        POP    ACC
        POP    PSW
        RETI
DELAY: MOV    R7,#02
DELAY0: MOV   R4,#187
DELAY1: MOV   R3,#248
        DJNZ   R3,$
        DJNZ   R4,DELAY1
        DJNZ   R7,DELAY0

```

```

DJNZ    R5,DELAY
RET
TABLE: DB 82H,01H,81H,94H,84H,0B4H,0A4H,04H
        DB 82H,01H,81H,94H,84H,0C4H,0B4H,04H
        DB 82H,01H,81H,0F4H,0D4H,0B4H,0A4H,94H
        DB 0E2H,01H,0E1H,0D4H,0B4H,0C4H,0B4H,04H
        DB 82H,01H,81H,94H,84H,0B4H,0A4H,04H
        DB 82H,01H,81H,94H,84H,0C4H,0B4H,04H
        DB 82H,01H,81H,0F4H,0D4H,0B4H,0A4H,94H
        DB 0E2H,01H,0E1H,0D4H,0B4H,0C4H,0B4H,04H,00H

```

```

TABLE1: DW 64260,64400,64524,64580,64684,64777,64820,64898
        DW 64968,65030,65058,65110,65157,65178,65217

```

END

HC164 串并转换

编写一段程序，通过单片机的 P1 口控制 74HC164 的串行输入端口，实现串并转换。通过修改串行数据，使输出 LED 数码管显示 1,2,3,, 8, 9, 0.

```

CLK EQU P1.0
DINA EQU P1.1
DINB EQU P1.2
CLR164 EQU P1.3
ORG 0000H
LJMP MAIN
ORG 0100H

```

MAIN:

```

MOV SP,#60H      ;设置堆栈向量
NOP            ;设置以下端口初始化
CLR CLK        ;CLK=0
SETB DINB       ;DINB=1
CLR CLR164      ;CLR=0 输出端口清零
SETB CLR164      ;CLR=1
MOV A,#0AAH      ;用户输出数据初始化
MOV R4,#08H

```

SLCHG: RLC A

```

MOV DINA,C      ;串行输出一位数据
SETB CLK        ;移位时钟
NOP
CLR CLK
NOP
DJNZ R4,SLCHG
SJMP $          ;程序结束,完成一次串并转换

```

END

步进电机

编写一段程序，通过单片机的 P1 口控制步进电机的控制端，使其按一定的控制方式进行转动。分别可以用双四拍（AB,BC,CD,DA,AB……）方式，单四拍方式（A,B,C,D,A……）方式等。

```

BA EQU P1.0
BB EQU P1.1
BC EQU P1.2
BD EQU P1.3
ORG 8000H

```

```

LJMP    MAIN
ORG 8100H
MAIN:
    MOVSP,#60H
    ACALL  DELAY
SMRUN:           ;电机控制方式为单双八拍
    MOVP1,#08H      ;A
    ACALL  DELAY
    MOVP1,#0CH      ;AB
    ACALL  DELAY
    MOVP1,#04H      ;B
    ACALL  DELAY
    MOVP1,#06H      ;BC
    ACALL  DELAY
    MOVP1,#02H      ;C
    ACALL  DELAY
    MOVP1,#03H      ;CD
    ACALL  DELAY
    MOVP1,#01H      ;D
    ACALL  DELAY
    SJMP   SMRUN      ;循环转动
DELAY:           ;单步延时程序
    MOVR4,#10
DELAY1:  MOVR5,#250
    DJNZR5,$
    DJNZR4,DELAY1
    RET
    END

```

乘法实验

编制一段程序，用 30H, 31H(高)2 个二进制数，乘以 40H 1 个二进制数，结果放 50H,51H,52H(高)中。

```

        ORG    0000H
MAIN:   MOV    R2,31H
        MOV    R3,30H
        MOV    R6, #00H
        MOV    R7,40H
LCALL   QMUL
MOV     52H,R5
        MOV    51H,R6
        MOV    50H,R7
        SJMP   $
QMUL:   MOV    R4,#00H      ;无符号双字节乘法子程序。
MOV     R5,#00H
        MOV    R0,#10H
        CLR    C
NMLP:   MOV    A,R4
        RRC    A
        MOV    R4,A

```

```

MOV    A,R5
RRC    A
MOV    R5,A
MOV    A,R6
RRC    A
MOV    R6,A
MOV    A,R7
RRC    A
MOV    R7,A
JNC    NMLN
MOV    A,R5
ADD    A,R3
MOV    R5,A
MOV    A,R4
ADDC   A,R2
MOV    R4,A
NMLN: DJNZ   R0,NMLP
      MOV    A,R4
      RRC    A
      MOV    R4,A
      MOV    A,R5
      RRC    A
      MOV    R5,A
      MOV    A,R6
      RRC    A
      MOV    R6,A
      MOV    A,R7
      RRC    A
      MOV    R7,A
      RET
END

```

第5章 MCS-51 定时/计数器、串行口及中断系统

1. 两个 16 位定时计数器
2. 定时器控制寄存器 TCON (88H); C/T 方式寄存器 TMOD(89H), 不能位寻址
3. 定时器的工作方式:
 - 方式 0: 13 位定时计数器, 注意: TL0 的低 5 位和 TH0 共同组成
 - 方式 1: 16 位定时计数器
 - 方式 2: 自动重装入的 8 位定时计数器, 溢出后 ($TF0=0$) 由 $TH0 \rightarrow TL0$
 - 方式 3: T0 成为两个独立的 8 位计数器
 - $TL0$ 作为定时计数器; $TH0$ 仅作定时器用
 - $TL0$ 的控制用原 $T0$ 的, $TH0$ 用原 $T1$ 的控制位
 - $T1$ 工作在方式 0~2, 溢出时送串行口, 经常作为串行口波特率发生器
4. 计数器初值:
 - 设计数模值为 M, 计数初值设定为 TC, 计数器计满为零所需的计数值为 C, 则: $TC=M-C$ ($M=213, 216, 28$)
5. 定时器初值:

$$T = (M-TC) T_{\text{机器}}$$
6. $F_{\text{osc}}=12\text{MHz}$, 试计算定时时间 2ms 所需的定时器的初值
 - 方式 2, 方式 3: $T_{\text{MAX}}=0.256 \mu\text{s}$, 所以必须将工作方式设在方式 0 或方式 1

方式 0: $TC=2^{13}-2_{MS}/1_{US}=6192=1830H$

$TL0=10H, TH0=0C1H$

方式 1: $TC=2^{16}-2_{MS}/1_{US}=63536=0F830H$

$TL0=30H, TH0=0F8H$

7. 串行通信: 数据一位接一位顺序传送, 可只用一根数据线传送多位信息
8. 串行通信有两种基本方式: 同步通信和异步通信
9. 串行接口有单工、半双工和全双工 3 种
10. MCS-51 单片机片内有一个串行接口, 可提供同步或全双工异步串行通信方式.
11. 与串行口有关的特殊功能寄存器有:

SCON:串行口控制寄存器

SBUF:缓冲寄存器

PCON:功耗控制寄存器(D7:SMOD 为波特率系数选择位)

12. 波特率的选择

串行口方式 0 的波特率是固定的, 为系统时钟的 12 分频($fosc/12$), 即每个机器周期传送一位数据位。

串行口用方式 2 工作时, 波特率为 $(2SMOD/64) \times fosc$ 。

串行口方式 1 和方式 3 用定时器 T1 作为波特率发生器, 其波特率有多种选择, 与 T1 的溢出率有关

串行口方式 1、3 的波特率 = $(2SMOD / 32) \times T1$ 溢出率

T1 的溢出率即 T1 溢出时间的倒数, 它与 T1 选择的功能、工作方式和预置初值等有关

若定时器 T1 设定为自动重装方式, T1 的溢出率及串行口波特率算式如下:

当单片机与 PC 机通讯, 假定波特率为 9600 波特, 当单片机的 FOSC=11.0592MHZ, SMOD=1 时, 可计算 $X=250=0FAH$

将 X 写入 TH1 和 TL1 时, 波特率发生器产生的实际传输率为波特率 = 9599.84 波特

波特率相对误差 = $(9600 - 9599.84)/9600 = 0.00177\%$

13. MCS-51 单片机中断系统: 在执行程序的过程中, 由于某种外界的原因, 必须尽快终止当前的程序执行, 而去执行相应的处理程序, 待处理结束后, 在回来继续执行被终止的程序。这个过程叫中断。
14. 中断技术的优点: 提高 CPU 的效率; 提高实时数据的处理时效; 故障处理
15. 中断系统的功能: 中断优先权排队; 实现中断嵌套; 自动响应中断; 实现中断返回。
16. 5 个中断源, 具有两个中断优先级, 可实现二级中断服务程序的嵌套。每个中断源均可软件编程为高优先级或低优先级中断, 允许或禁止向 CPU 请求中断。
17. 有关的特殊功能寄存器 (SFR) 有:

中断允许寄存器 IE

中断优先级控制寄存器 IP

中断源寄存器 (TCON、SCON 中的有关位)

18. 外部中断源 INT0、INT1: 中断标志和触发方式控制位锁存在 TCON 的低四位。

IE0 (IE1) =1 表示正在向 CPU 申请中断, 响应后自动清零。

IT0 (IT1) =0: 低电平触发; IT0 (IT1) =1: 边沿触发

19. 内部中断源

T0: TF0 定时器 T0 的溢出中断请求

T1 : TF1 定时器 T1 的溢出中断请求

串行口中断: 发送中断 TI 和接收中断 RI 逻辑或后做为内部的一个中断源。

20. 中断使能控制 IE (A8H)EA — ES ET1 EX1 ETO EX0, 实现两级控制, 注意: 复位时, 禁止所有中断

21. 中断优先级控制 IP(B8H) PS PT1 PX1 PTO PX0, 每一中断源可编程为高优先级或低优先级中断, 以实现二级嵌套。

22. 默认的优先次序为: INT0、C/T0、INT1、C/T1、串行口中断 (依次从高到低)

第 6 章 单片机系统扩展设计

1. MCS-51 单片机有四个并行 I/O 口。当用 MCS-51 单片机组成的应用系统需外扩程序存储器和数据存储器时, 真正可用的并行口, 就只有一个 P1 口了。
2. 8255A 工作方式
 - 方式 0: 基本式输入输出
 - 方式 1: 选通式输入输出
 - 方式 2: 双向传送方式

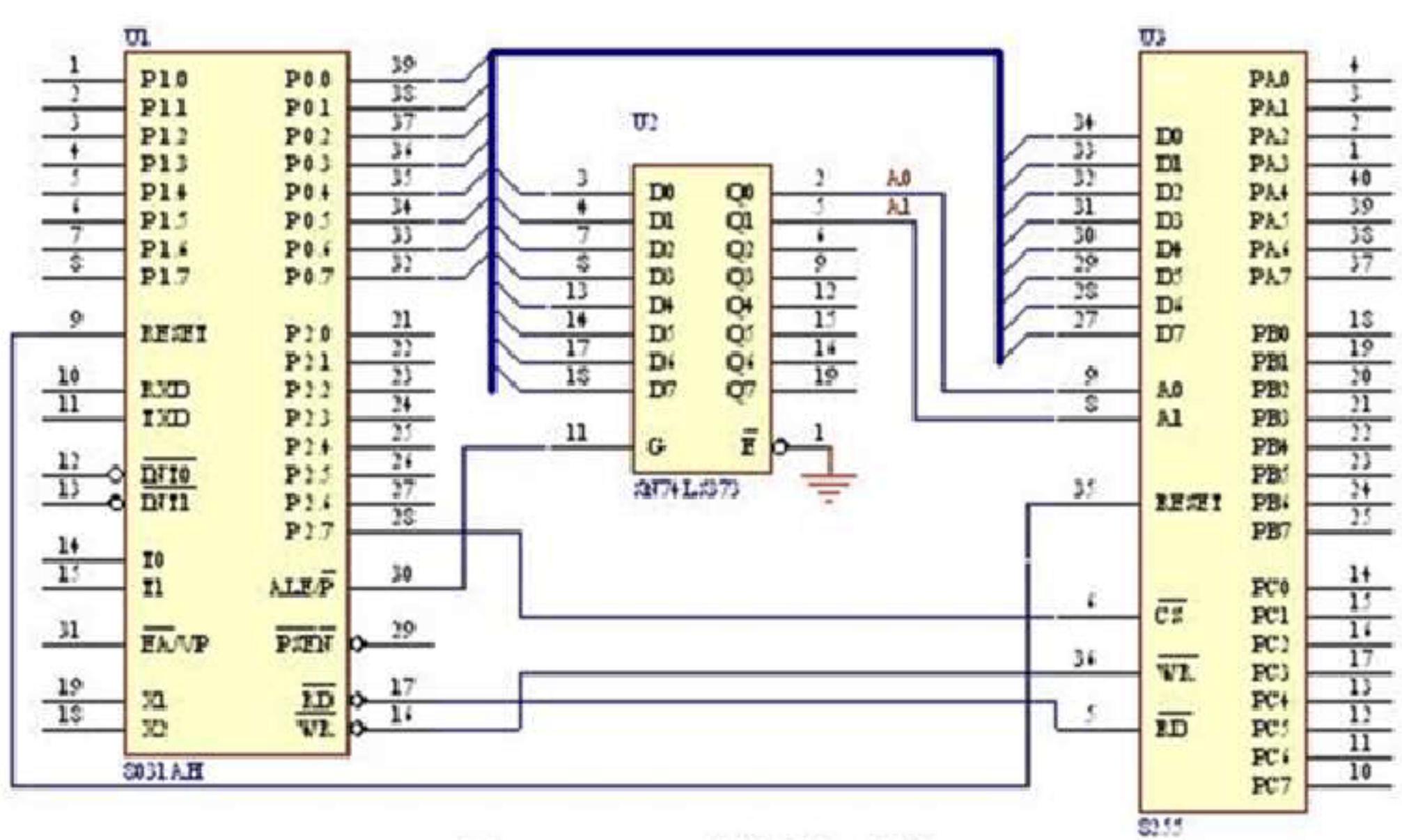


图 7.3 8255A 的扩展接口电路

- 3.
4. MOV A, #98H ; 方式控制字→A
MOV DPTR, #7FFFH; 选通控制寄存器
MOVX @DPTR, A ; 方式控制字送入 8255A
MOV DPTR, #7FFCH;
MOVX A, @DPTR ; 读 PA 口数据
MOV DPTR, #7FFDH;
MOVX @DPTR, A ; 送 PB 口输出
5. 8155 扩展 I/O 组成的行列式键盘

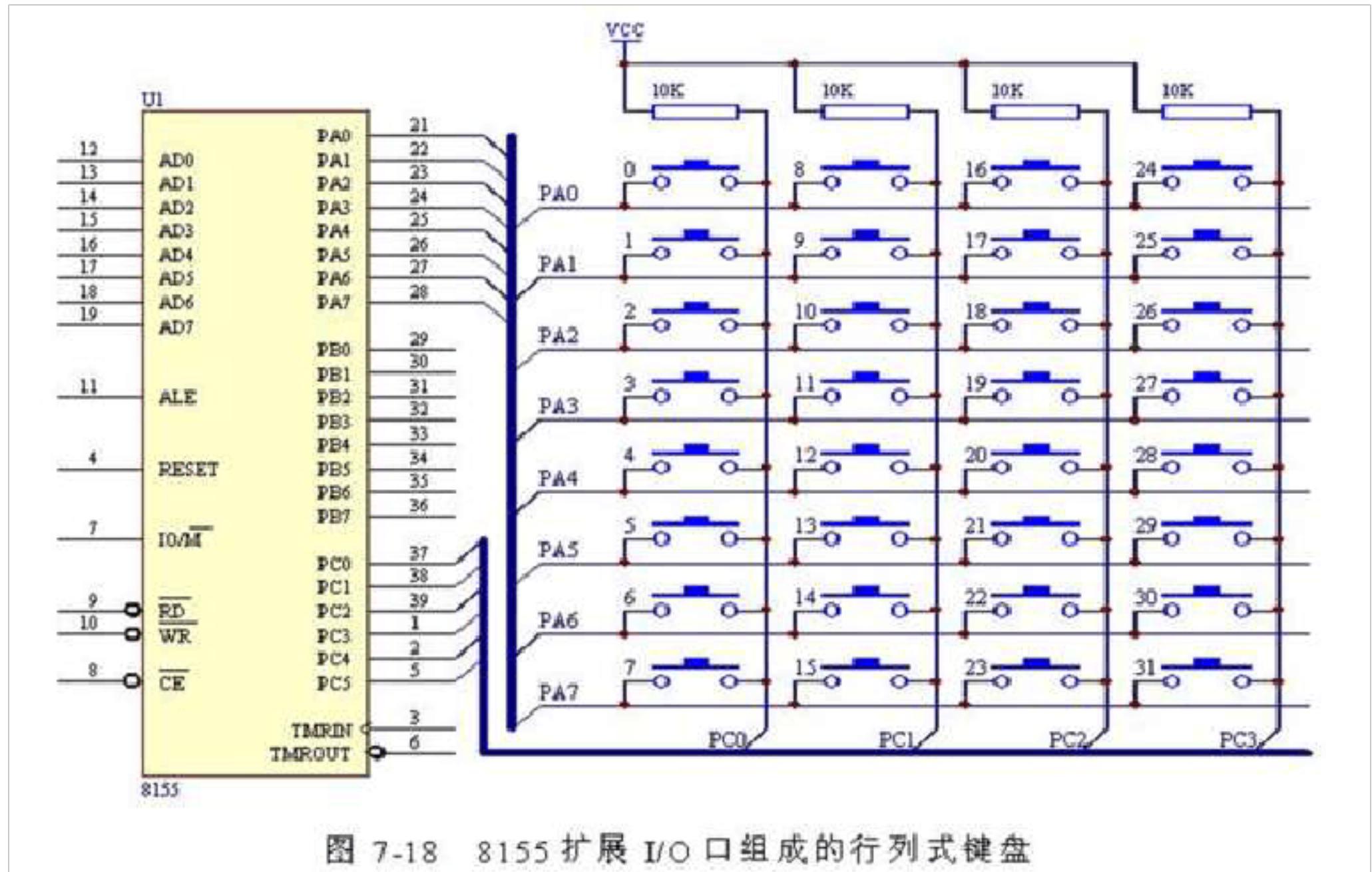


图 7-18 8155 扩展 I/O 口组成的行列式键盘

KEY: CLR A

```

ACALL KS      ; 有键按下吗?
JZ     NK      ; 无键按下返回
ACALL DLAY    ; 调用延时程序, 消除抖动
CLR    A       ;
ACALL KS      ; 再次判断是否有键按下
JZ     NK      ; 无键按下返回
MOV    A, #0FEH; 行扫描信号, 从最低位开始
MOV    R4, #0   ; 行计数器
K1:   MOV    R2, A
      ACALL KS      ; 扫描键盘
      JNZ    FIND    ; 找到键转移
      INC    R4      ; 行计数器加 1, 指向下一行
      MOV    A, R2
      RL     A       ; 行扫描信号左移一位
      CJNE   A, #0FE, K1 ; 8 行扫描完?
      MOV    A, #0      ; 没找到键
      SJMP   NK
FIND: SWAP A
      ADD    A, R4
NK:   RET
KS:   MOV    DPTR, #PA  ; A 口地址送 DPTR
      MOVX   @DPTR, A  ; 送行扫描信号
      MOV    DPTR, #PC  ; C 口地址送 DPTR
      MOVX   A, @DPTR  ; 读列回扫信号
      CPL    A       ; 求反
      ANL    A, #0FH   ; 屏蔽高四位
  
```

RET ; A=0, 无键按下

第7章 数模及模数转换器接口

1. 能够把模拟量变成数字量的器件称为模数转换器(A/D); 能够把数字量变成模拟量的器件称为数模转换器(D/A)
2. D/A转换器原理——R-2R T型解码网络 D/A转换器

N位二进制数码输入时, 输出模拟电压: $V_0 = -\frac{V_{REF}}{2^N} \sum_{i=0}^{N-1} A_i \times 2^i$

8位并行D/A转换器, 输出模拟电压 $V_0 = -\frac{V_{REF}}{2^8} (2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0)$