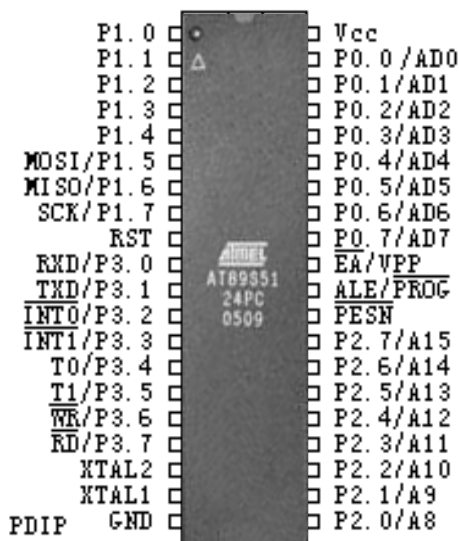


《 精通MCS-51单片机 绝世秘笈 》



★★★

本资料纯属虚构，如有雷同，此乃巧合也！

★★★

前言

本秘笈共有三部：

- 《六脉神剑》（指令篇）—— 以指为剑 以气为剑
- 《九阴真经》（资源篇）—— 内家密功 积厚薄发
- 《葵花宝典》（中断篇）—— 葵花点穴 叫停就停

打印成手册效果最佳。

谨以此书献给所有热爱单片机的朋友。

E-Mail: aki_studio@163.com

技术问题，欢迎登录交流 [HTTP://www.8951.com/bbs](http://www.8951.com/bbs)

《精通MCS-51单片机之——六脉神剑》

指令手册篇

§ 汇编语言与指令系统简介

一、汇编语言程序设计的意义

MCS-51指令格式:

标号: 操作码助记符 第一操作数, 第二操作数; 注释

汇编语言程序的每一条语句都与计算机的某一条指令对应, 所以必需熟悉指令系统。

指令 = 操作码 + 操作数

操作码——表示了该指令所能执行的操作功能。

操作数——表示参加操作的数的本身或操作数所在的地址。

二、MCS-51系列单片机的指令系统

111条指令, 共分五大类:

数据传送类; (29条) 算术运算类; (24条) 逻辑运算类; (24条)

控制转移类; (17条) 位操作类。(17条)

指令中操作数的描述符号:

Rn —— 工作寄存器R0~ R7

Ri —— 间接寻址寄存器R0、R1

Direct —— 直接地址, 包括内部128B RAM单元地址、26个SFR地址。

#data —— 8位常数

#data 16 —— 16位常数

addr 16 —— 16位目的地址

addr 11 —— 11位目的地址

rel —— 8位带符号的偏移地址

DPTR —— 16位外部数据指针寄存器

bit —— 可直接位寻址的位

A —— 累加器
B —— 寄存器B
C —— 进、借位标志位，或位累加器
@ —— 间接寄存器或基址寄存器的前缀
/ —— 指定位求反
(x) —— x中的内容
((x)) —— x中的地址中的内容
\$ —— 当前指令存放的地址

三、寻址方式

寻找操作数存放单元的地址的方式，共6种方式。

1、立即数寻址

所要找的操作数是一二进制数或十进制数，出现在指令中，用“#”作前缀

```
MOV A, #20H
```

2、寄存器寻址

操作数存放在工作寄存器R0~R7中，或寄存器B中。

```
MOV A, R2
```

3、直接寻址

指令中直接给出操作数的地址。

```
MOV A, 30H
```

```
MOV 30H, DPH
```

4、寄存器间接寻址

指令中寄存器的内容作为操作数存放的地址，指令中间接寻址寄存器前用“@”表示前缀。

举“两个抽屉，两把钥匙”的例子。

```
MOV R0, #30H
```

```
MOV A, @R0
```

```
MOV A, #20H
```

```
MOV R1, #40H
```

```
MOV @R1, A
```

5、变址寻址

操作数地址 = 变地址 + 基地址

基址寄存器 DPTR 或 PC

变址寄存器 @A

该寻址方式常用于访问程序存储器，查表。

```
MOV A, @A + DPTR
```

6、相对寻址

把指令中给定的地址偏移量与本指令所在单元地址（PC内容）相加得到真正有效的操作数所存放的地址。

举“李同学20岁，张同学比李同学大3岁”的例子。

JC 60H ; 设 (PC) = 2000H,

则当C = 1时,

转移的目的地址 = (PC) + 2 + 60H

★数据传送类指令

一、内部RAM数据传送类

1、一般数据传送指令

```
MOV A, Rn
MOV A, direct
MOV A, @Ri
MOV A, #data
MOV Rn, A
MOV Rn, direct
MOV Rn, #data
MOV direct, A
MOV direct, Rn
MOV direct, direct
MOV direct, @Ri
MOV direct, #data
MOV @Ri, A
MOV @Ri, direct
MOV @Ri, #data
MOV DPTR, #data16
```

上述指令不影响任何标志位，但PSW的P位除外。

注意: MOV Rn, Rn

MOV @Ri, @Ri

MOV Rn, @Ri

MOV #data, A

等指令是非法指令。

2、栈操作指令

PUSH direct

POP direct

不影响任何标志位。

3、字节交换指令

XCH A, Rn

XCH A, direct

XCH A, @Ri

XCHD A, @Ri

SWAP A

不影响任何标志位。

二、外部RAM数据传送

MOVX A, @Ri

MOVX A, @DPTR

MOVX @Ri, A

MOVX @DPTR, A

执行过程中会使/WR、/RD有效。

[例]试编写一程序段，实现将外RAM 0FAH单元中的内容传送到外RAM 04FFH单元中。

解：MOV DPTR, #04FFH

MOV R0, #0FAH

MOVX A, @R0

MOVX @DPTR, A

三、查表指令

与ROM之间的数据传送。

MOVC A, @A + DPTR

MOVC A, @A + PC

执行后会使/PSEN有效。

MOVC 含义是传送常数。

- 以DPTR 为基址的指令，可在ROM 的64KB范围内查表；
- 以PC为基址的指令只在 (PC) + 1为中心上、下256B范围内查表。

[例] 设 (A) = 一个BCD码常数, 试用查表法获得其相应的ASCII码。

解法I: MOV DPTR, #TAB
 MOVC A, @.A+DPTR

TAB: DB 30H
 DB 31H
 DB 32H, 33H, 34H, 35H

解法II: MOVC A, @A+PC
TAB: DB 30H, 31H, 32H, 33H
 DB 34H, 35H, 36H, 37H

★算术运算类指令

一、加法指令

包括: 加、减、乘、除; 加一、减一。

ADD A, Rn
ADD A, direct
ADD A, @Ri
ADD A, #data

无符号数相加时: 若C = 1, 则有溢出 (值 > 255)。

带符号数相加时: 若OV = D7c ⊕ D6c = 1, 则有溢出。

ADDC A, Rn
ADDC A, direct
ADDC A, @Ri
ADDC A, #data

上述四条指令多用于多字节数相加。

INC A
INC Rn
INC direct
INC @Ri
INC DPTR

[例] 设 (R0) = 7FH; (7EH) = 40H 执行:

INC @R0

INC R0

INC @R0 后,

(R0) = 7FH; (7EH) = 00H; (7FH) = 41H.

DA A ; 二~十进制调整

执行过程中, CPU能根据加法运算后, 累加器中的值和PSW中的AC及C标志位的状况自动选择一个修正值(00H、06H、60H、66H)与原运算结果相加, 进行二~十进制调整。

选择修正值的规则:

(A3 ~ 0) > 9 或 (AC) = 1 时, (A3 ~ 0) ← (A3 ~ 0) + 6

(A7 ~ 4) > 9 或 (C) = 1 时, (A7 ~ 4) ← (A7 ~ 4) + 6

[例] 设 (A) = 56H 为 56 的压缩的BCD码数, (R3) = 67H, (CY) = 1

执行 ADDC A, R3

DA A

结果为: 124

注意: 1) **DA**指令只能跟在加法指令后面使用;

2) 调整前参与运算的两数是BCD码数;

3) **DA**指令不能与减法指令配对使用, 但可以实现对A中压缩BCD减一操作。

[例] 设 (A) = 30H (压缩BCD码数), 执行: ADD A, #99H

DA A

便实现了 $30 - 1 = 29$ 的操作

[例] 两个4位BCD码相加, 一个存放在 (31H) (30H); 另一个存放在 (33H) (32H); 和数拟回存在 (31H) (30H) 中, 试编程实现。

解: MOV R0, #30H

MOV R1, #32H

MOV A, @R0

ADD A, @R1

DA A

MOV @R0, A

INC R0

INC R1

MOV A, @R0

ADDC A, @R1

二、减法指令

SUBB A, Rn

SUBB A, direct

SUBB A, @Ri

SUBB A, #data

注意：减法之前先清零C。

DEC A

DEC Rn

DEC @Ri

DEC direct

[例] 设 (R0) = 7FH, 在内RAM中, (7EH) = 00H, (7FH) = 40H

执行: **DEC** @R0

DEC R0

DEC @R0

结果为: (R0) = 7EH, (7EH) = 0FFH, (7FH) = 3FH。

三、乘法和除法指令

乘法:

MUL AB; (A) × (B), 积的低8位在A中, 积的高8位在B中;
C总为0。

除法:

DIV AB; (A) ÷ (B), 商在A中, 余数在B中。
若 (B) = 0, 则结果不定, (0V) = 1, (C) = 0。

[例] 试将A中的二进制数转换为3位BCD码, 其中百位数存放于31H单元, 十位数和个位数压缩后存于30H单元中。

解: **MOV** B, #100

DIV AB

MOV 31H, A

MOV A, #10

XCH A, B

DIV AB

SWAP A

ADD A, B

MOV 30H, A

★逻辑操作类指令(共24条)

共分两大类：单字节逻辑操作，双字节逻辑操作。

一、单字节逻辑操作指令

CLR A

CPL A ; A中8位按位求反

循环左移、右移指令：

RL A

RLC A

RR A

RRC A

注：左移一位相当于乘2；右移一位相当于除2。

二、双字节逻辑操作指令

“与操作”：

ANL A, Rn

ANL A, direct

ANL A, @Ri

ANL A, #data

ANL direct, A

ANL direct, #data

[例] (P1) = 35H, 使其高4位输出0, 低4位不变。

解: **ANL** P1, #0FH

此做法称为“屏蔽”位。

“或操作”：

ORL A, Rn

ORL A, direct

ORL A, @Ri

ORL A, #data
ORL direct, A
ORL direct, #data

[例]将A中的低3位送入P1中，并且保持P1中高5位不变。

ANL A, #07H
ANL P1, #0F8H
ORL P1, A ; (P1) = P17P16P15P14P13A2A1A0
这称为“数位组合”。

“异或操作”：

XRL A, Rn
XRL A, direct
XRL A, @Ri
XRL A, #data
XRL direct, A
XRL direct, #data

[例]设 (P1) = 0B4H = 10110100B，执行：

XRL P1, #00110001B

结果按# 0 0 1 1 0 0 0 1 取反，即：

(P1) = 1 0 0 0 0 1 0 1 B = 85H

这称为“指定位取反”。

在上述ANL、ORL、XRL操作中，用于端口操作时，无论P0 ~ P3是第一，还是第二操作数，都遵循“读—修改—写”端口锁存器的操作。

★控制转移类指令

作用：改变程序计数器PC的值，从而改变程序执行方向。

分为四大类：无条件转移指令；条件转移指令；调用指令；返回指令。

一、无条件转移指令

LJMP addr16; 长跳转
AJMP addr11; 绝对转移
SJMP rel ; 短转移
JMP @A + DPTR; 间接转移

[例] 设A中为键值，试编写按键值处理相应事件的程序段。

解： **MOV DPTR, #KYG**

MOV B, #03H

MUL AB

JMP @A + DPTR

.....

KYG: LJMP KYG0

LJMP KYG1

.....

画图比较LJMP、AJMP、SJMP、JMP转移的起点和范围。

●调用指令

LCALL addr16; 长调用

[例] 设 (SP) = 07H, (PC) = 2100H, 子程序首地址为3456H,

执行: **LCALL 3456H**

MOV A, 20H

.....

画出执行过程示意图。

执行结果: (SP) = 09H, (09H) = 21H, (08H) = 03H, (PC) = 3456H

ACALL addr11; 绝对调用

●返回指令

从子程序返回主程序。

RET ; 调用子程序返回

RETI ; 中断子程序返回

画图比较两种返回指令含义上的异同点,

结论: **RET**返回地址事先已知, 而**RETI**的返回地址在程序执行中产生的, 不固定。

不影响标志位, 但PSW不能恢复到中断前的状态。

●空操作指令

NOP ; 空操作

不执行任何操作, 仅仅使 (PC) + 1, 继续执行下条指令, 不影响标志位, 在ROM中占一个字节。用于延时调整。

二、条件转移指令

实现按照一定条件决定转移的方向。分三类。

1、判零转移

JZ rel ; 若 $(A) = 0$, 则转移, 否则顺序执行。

JNZ rel; 若 $(A) \neq 0$, 则转移, 否则顺序执行。转移目的地址 = $(PC) + 2 + rel$
不影响任何标志位。

[例]将外RAM的一个数据块(首地址为DATA1)传送到内部数据RAM(首地址为DATA2), 遇到传送的数据为零时停止传送, 试编程

解: **MOV** R0, #DATA2

MOV DPTR, #DATA1

LOOP1: MOVX A, @DPTR

JZ LOOP2

MOV @R0, A

INC R0

INC DPTR

SJMP LOOP1

LOOP2: SJMP LOOP2

2、比较转移指令

功能: 比较二个字节中的值, 若不等, 则转移。

CINE A, #data, rel

CJNE A, direct, rel

CJNE @Ri, #data, rel

CJNE Rn, #data, rel

该类指令具有比较和判断双重功能, 比较的本质是做减法运算, 用第一操作数内容减去第二操作数内容, 但差值不回存。

转移目的地址 = $(PC) + 3 + rel$

若第一操作数内容小于第二操作数内容, 则 $(C) = 1$, 否则 $(C) = 0$ 。

该类指令可产生三分支程序:

即, 相等分支; 大于分支; 小于分支。

[例] 设P1口的P1.0 ~ P1.3为准备就绪信号输入端，当该四位为全1时，说明各项工作已准备好，单片机可顺序执行，否则，循环等待。

解: `MOV A, P1`
`ANL A, #0FH`
`CJNE A, #0FH, WAIT; P1.0 ~P1.3不为全1时，返回WAIT`
`MOV A, R2`
`.....`

3、循环转移指令

DJNZ Rn, rel; (二字节指令)

DINZ direct, rel; (三字节指令)

本指令也为双功能指令，即减1操作和判断转移操作。

第一操作数内容减1后，若差值不为零，则转移；否则顺序执行。

转移目的地址 = (PC) + 2或3 + rel

[例] 将8031内部RAM的40H ~ 4FH单元置初值 #A0H ~ #AFH.

解: `MOV R0, #40H`
`MOV R2, #10H`
`MOV A, #0A0H`
`LOOP: MOV @R0, A`
`INC R0`
`INC A`
`DJNZ R2, LOOP`
`.....`

★位操作指令

包括：位传送指令、条件转移指令、位运算指令。

位操作由单片机内布尔处理器来完成。

位地址的四种表示：

- 1) 使用直接位地址表示：如20H、30H、33H等；
- 2) 使用位寄存器名来表示；如C、OV、F0等；
- 3) 用字节寄存器名后加位数来表示：如PSW.4、P0.5、ACC.3等；
- 4) 字节地址加位数来表示：如20.0、30.4、50.7等。

一、位传送指令

MOV C, bit

MOV bit, C

功能: $(C) \longleftrightarrow (\text{bit})$;

二、位状态控制指令

CLR bit ; $(\text{bit}) \leftarrow 0$

SETB bit ; $(\text{bit}) \leftarrow 1$

CPL bit ; $(\text{bit}) \leftarrow \neg(\text{bit})$

[例]编程通过P10线连续输出256个宽度为5个机器周期长的方波。

解: **MOV** R0, #00H
 CLR P10
LOOP: **CPL** P10
 NOP
 NOP
 DJNZ R0, **LOOP**

三、位逻辑操作指令

ANL C, bit; $C \leftarrow C \wedge (\text{bit})$

ANL C, /bit; $C \leftarrow C \wedge (\neg \text{bit})$

ORL C, bit; $C \leftarrow C \vee (\text{bit})$

ORL C, /bit; $C \leftarrow C \vee (\neg \text{bit})$

四、布尔条件转移指令

有5条，分别对C和直接位地址进行测试，并根据其状态执行转移。

1、判布尔累加器转移

JC rel ; $(C) = 1$, 转移, 否则顺序执行。

JNC rel ; $(C) = 0$, 转移, 否则顺序执行。

不影响标志。转移地址 : $(PC) \leftarrow (PC) + \text{rel}$

[例]较内部RAM的30H和40H单元中的二个无符号数的大小，将大数存入20H单元，小数存入21H单元，若二数相等，则使内RAM的第127位置1。

解: MOV A, 30H
 CJNE A, 40H, LOOP
 SETB 7FH
 SJMP \$
LOOP1: JC LOOP2
 MOV 20H, A
 MOV 21H, 40H
 SJMP \$
LOOP2: MOV 20H, 40H
 MOV 21H, A
 SJMP \$

2、判位变量转移

JB bit, rel; (bit) = 1，则转移，否则顺序执行。

JBC bit, rel; (bit) = 1，则转移，否则顺序执行，且无论 (bit) 是否等于1，均使该位清零。

JNB bit, rel; (bit) = 0，则转移，否则顺序执行
不影响标志。

[例]试判断A中的正负，若为正数，存入20H单元；若为负数则存入21H单元。

解: JB ACC7, LOOP
 MOV 20H, A
 SJMP \$
LOOP: MOV 21H, A
 SJMP \$

★伪指令

伪指令既不控制机器的操作，也不能被汇编成机器代码，故称**伪指令**

1、起始地址伪指令

ORG addr16

规定目标程序段或数据块的起始地址，设置在程序开始处。

2、汇编结束伪指令

END

告诉汇编程序，对源程序的汇编到此结束。一个程序中只出现一次，在末尾。

3、赋值伪指令

将汇编语句操作数的值赋予本语句的标号。

标号名称 **EQU** 数值或汇编符号

“标号名称”在源程序中可以作数值使用，也可以作数据地址、位地址使用。

先定义后使用，放在程序开头。

4、定义字节伪指令

从指定地址单元开始，定义若干字节存储单元并赋初值

[标号:] **DB** 字节数据或字节数据表

5、定义字伪指令

DW 从指定地址开始，定义若干个16个位数据，高8位存入低地址；低8位存入高地址。

[例] **ORG 1000H**

PI0I: DW 7654H, 40H, 12, 'AB'

6、数据地址赋值伪指令

字符名 **DATA** 表达式

将表达式指定的数据地址赋予规定的字符名称

注：该指令与**EQU**指令相似，只是，可先使用后定义，放于程序开头、结尾均可。

7、定义空间伪指令

[标号:] **DS** 表达式

从指定地址开始，保留由表达式指定的若干字节空间作为备用空间。

[例] **ORG 1000H**

DS 0AH

DB 71H, 11H, 11H 从**100BH**开始存放**71H、11H、11H**。

注：DB、DW、DS 只能用于程序存储器；而不能用于数据存储器。

8、位地址赋值伪指令

字符名称 **BIT** 位地址

将位地址赋予规定的字符名称。

[例] **X1 BIT P12**

相当于 **X1 EQU 92H**

凌晨5: 38 <待续…>

Q33NY

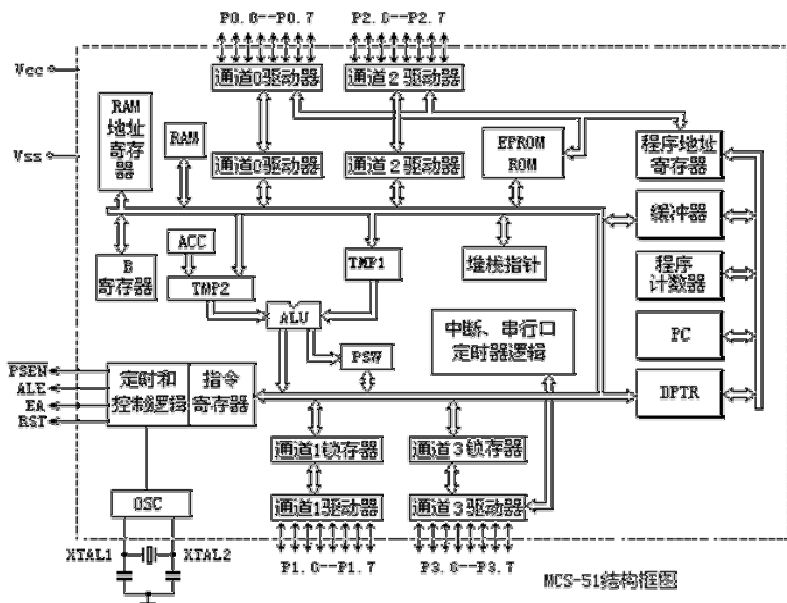
《精通MCS-51单片机之——九阴真经》

内部资源篇

一、MCS-51 系列单片机是因特公司 1980 推出的高档 8 位机，分为二个子系列、三个版本。

存储器类			掩膜 MOS	EPROM
型	单片机系列			
MCS-51	51子系列	8031	/	/
		8051	4KB	/
		8751	/	4KB
	52子系列	8032	/	/
		8052	8KB	/

二、MCS-51 单片机内部结构工作框图：

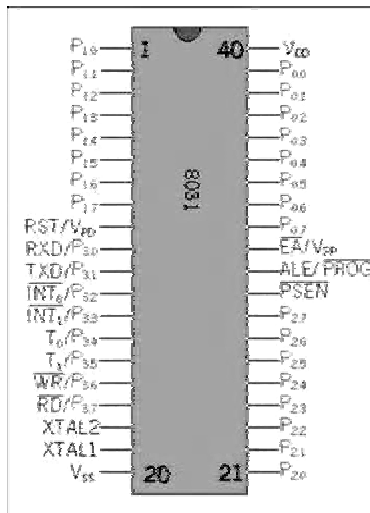
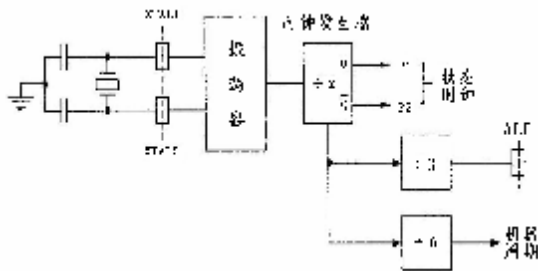


① 中央处理单元 CPU (8 位)

用于数据处理、位操作 (位测试、置位、复位)

- ② 只读存储器 **ROM** (4KB 或 8KB)
用于永久性存储应用程序, 掩膜 **ROM** EPROM EEPROM
- ③ 随机存取存储器 **RAM** (256B)
用于程序运行中存储工作变量和数据
- ④ 并行输入/输出口 **I / O** (32 线)
用作系统总线、扩展外存、**I / O** 接口芯片
- ⑤ 串行输入/输出口 **UART** (二线)
串行通信、扩展 **I / O** 接口芯片
- ⑥ 定时/计数器 **T** (16 位增量可编程)
它与 **CPU** 之间各自独立工作, 当它计数满时向 **CPU** 中断
- ⑦ 时钟电路 **fosc**
分为内部振荡器、外接振荡电路
- ⑧ 中断系统
五源中断、两级优先, 可编程进行控制。

引脚分类:

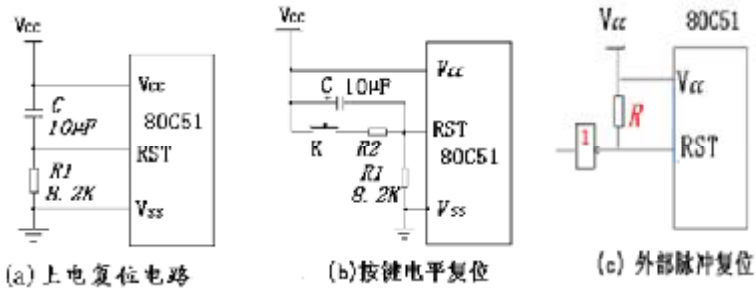


机器周期: 完成一个基本操作所需要的时间。
一个机器周期由 12 个时钟周期组成。

以机器周期为单位：单周期、双周期和四周期指令。

RST / VPD —— 当出现两个机器周期高电平时，单片机复位。

参考复位电路如下：



- ALE / PROG —— 地址锁存控制端
提供 $1/6 f_{osc}$ 振荡频率；为其内的 EPROM 输入编程脉冲
- PSEN —— 外部程序存储器的读选信号端
- EA/ Vpp —— 内\外 ROM 选择端
- EA = 1 时，访问内部程序存储器，即内 ROM
- EA = 0 时，只访问外部程序存储器，即外 ROM
- 对于 8751 单片机编程时，该端施加编程电压

④ 输入/输出引脚

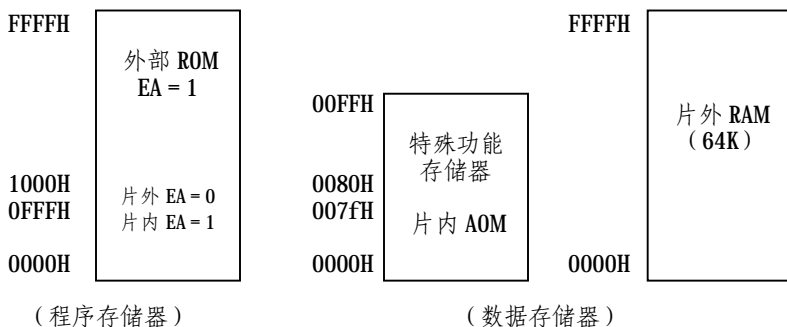
P0.0 ~ P0.7; P1.0 ~ P1.7; P2.0 ~ P2.7; P3.0 ~ P3.7

四个 I / O 口，每口八条线；还兼作地址/数据线。

四、MCS-51 寄存器配置

◆1、内存结构

- 物理上分为 4 个空间：
- 片内 ROM 片外 ROM 片内 RAM 片外 RAM
- 逻辑上分为 3 个空间：
- 程序内存（片内、外 64KB）统一编址 MOVc
- 数据存储器（片内 256B） MOV
- 数据存储器（片外 64KB） MOVX



① 程序存储器

寻址范围: 0000H ~ FFFFH 容量 64KB, 即地址长度: 16 位

EA = 1, 寻址内部 ROM (当 PC 值超过片内 ROM 容量时会自动转向 外部存储器空间。)

EA = 0, 寻址外部 ROM (8031 单片机 EA 接低电平)

作用: 存放程序及程序运行时所需的常数。

● 七个具有特殊含义的单元是:

0000H —— 系统复位, PC 指向此处;

001BH —— T1 溢出中断入口

0003H —— 外部中断 0 入口

0023H —— 串口中断入口

000BH —— T0 溢出中断入口

002BH —— T2 溢出中断入口

0013H —— 外中断 1 入口

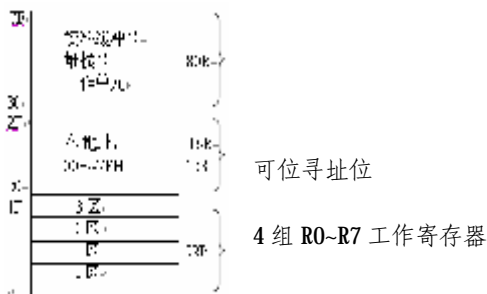
● 程序和数据存储器逻辑空间

普林斯顿结构: 程序和数据共用一个存储器逻辑空间, 统一编址。

哈佛结构: 程序与数据分为两个独立存储器逻辑空间, 分开编址。

② 内部数据存储器

物理上分为两大区: 00H~7FH 即 128B 内 RAM 区和 SFR 区。



(内部数据存储器) 作用: 作数据缓冲器用。

◆2、**特殊功能寄存器 SFR**（22 个，包括 PC 及 SFR）

寻址空间：80H ~ FFH ， 其中有 6 个双字节寄存器。

注意 PC 不在此范围内。

- PC 为程序计数器。它是一个双字节寄存器, 寻址范围为：0000H~ FFFFH, 即 ~ 64KB。
 - SFR 为特殊功能寄存器。其寻址空间：80H~ FFH ；
- 其中， 51 子系列有 18 个寄存器，占有 21 个字节；
52 子系列有 21 个寄存器，占有 26 个字节。

① 算术运算寄存器

- (1)累加器 A（E0H）
- (2)B 寄存器：乘、除法运算用
- (3)程序状态字 PSW 寄存器：包含程序运行状态信息。

PSW	CY	AC	FO	RS1	RS0	OV	—	P
-----	----	----	----	-----	-----	----	---	---

- CY —— 进位/借位标志；位累加器。
- AC —— 辅助进/借位标志；用于十进制调整。
- FO —— 用户定义标志位；软件置位/清零。
- OV —— 溢出标志； 硬件置位/清零。
- P —— 奇偶标志；A 中 1 的个数为奇数 P = 1；
- RS1、RS0 —— 寄存器区选择控制位。

0	0	: 0 区	R0 ~ R7	1	0	: 2 区	R0 ~ R7
0	1	: 1 区	R0 ~ R7	1	1	: 3 区	R0 ~ R7

② 指针寄存器

- (1)程序计数器**PC**:
指明即将执行的下一条指令的地址，16位，寻址64KB范围，复位时PC = 0000H
- (2)堆栈指针**SP**:
指明栈顶元素的地址，8位，可软件设置初值，复位时SP = 07H
- (3)数据指针**DPTR**
@R0、@R1、@DPTR; 指明访问的数据存储器的单元地址，16 位，寻址范围 64KB。
DPTR = DPH + DPL，也可单独使用。

③ 并行输入/输出端口

寄存器**P0、P1、P2、P3**实为相应端口锁存器。

④ 串行输入/输出端口

- (1)串行数据缓冲器 **SBUF**

是物理上独立的两个寄存器，共同使用一个地址。

(2) 串行控制/状态寄存器 **SCON**

控制监视串行口的工作状态

(3) 电源控制寄存器 **PCON**

控制单片机的低功耗工作方式及波特率选择。

⑤ 中断系统

(1) 中断优先级寄存器 **IP**：2 级优先，可软件设定

(2) 中断允许寄存器 **IE**

⑥ 定时/计数器

(1) 定时器方式寄存器：**TMOD**

(2) 定时器控制寄存器：**TCON**

(3) 计数寄存器：**TH0、TL0；TH1、TL1**。可设定计数初值。

⑦ 8052/8032 增设专用寄存器

(1) 定时器 2 控制寄存器 **T2CON**；控制、设置工作方式。

(2) 计数寄存器：**TH2、TL2**

(3) 定时器 2 捕获/重装载寄存器：**RCAP2H、RCAP2L**

存放自动重装载到 **TH2、TL2** 的数据。

◆ 3、位地址空间

凡地址能被 8 整除的寄存器都是可位寻址的寄存器。

(1) 内部 RAM 20H ~ 2FH 共 16 个单元可按位寻址 128 位

(2) SFR 80H ~ FFH 51 子系列，有 21 个寄存器，83 位；
52 子系列，有 26 个寄存器，93 位。

◆ 4、外部数据存储器

(1) 容量最大扩展到 64KB

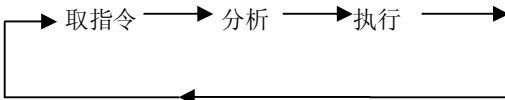
(2) 寻址范围：0000H ~ FFFFH

(3) 寻址方式：间接寻址可用 R0，R1 及 DPTR

* 五、CPI 时序

时钟的基本概念

启动单片机后，指令执行顺序



时序的定义：

单片机内的各种操作都是在一系列脉冲控制下进行的，而各脉冲在时间上是有先后顺序的，这种顺序就称为时序。

· 指令周期：即从取指到执行完，所需时间。

不同机器指令周期不一样；即使相同机器，不同的指令其指令周期也不一样。

· 机器周期：机器的基本操作周期。

一个指令周期含若干机器周期（单、双、四周期）

· 状态周期：一个机器周期分 6 个状态周期 S_i 每个状态周期含两个振荡周期。

即相位 P1、P2。

· 振荡周期：由振荡时钟产生。

振荡周期 $T_{osc} = 1/f_{osc}$

一个机器周期 = 12 个振荡周期 = $12 \times 1/f_{osc}$ 。

例如，若 $f_{osc} = 12\text{MHz}$ ，则一个机器周期 = $1\mu\text{s}$ 。

(1) 单字节单周期指令：INC A

只需进行一次读指令操作（指令只有一个字节），当第二个 ALE 有效时，由于 PC 没有加 1，读出的还是原指令。属于一次无效操作。

(2) 双字节单周期指令：ADD A, #data

ALE 两次读操作都有效，第一次读操作码（指令第一字节），第二次读立即数（指令第二字节）。

(3) 单字节双周期指令：INC DPTR

两个机器周期共进行四次读指令操作，但其后三次的读操作都是无效的。

时钟的产生：XTAL1（19）、XTAL（18）。

1、内部方式

与作为反馈元件的片外晶体或陶瓷谐振器一起组成一个自激振荡器。

2、外部方式

CMOS 工艺的 8031，其 XTAL1 接外信号；XTAL2 可悬空。HMOS 工艺的 8031，其 XTAL2 接外信号；XTAL1 接地。

六、并行 I/O 端口

四个端口、双向、每个口包含一个锁存器、一个输出驱动器和二个输入缓冲器。

【小结】

1、P0 口：地址低 8 位与数据线分时使用端口，

2、P1 口：按位可编址的输入输出端口，

3、P2 口：地址高 8 位输出

4、P3 口：双功能口。若不用第二功能，也可作通用 I / O 口。

5、按三总线划分：

地址线：P0 低八位地址，P2 高八地址；

数据线：P0 输入输出 8 位数据；

控制线：P3 口的 8 位加上 /PSEN、ALE 共同完成控制总线。

《精通MCS-51单片机之——葵花宝典》

一、中断系统

中断源的中断请求通过中断请求标志位来通知 CPU

◆1、五源中断

五个中断源:	入口地址
外部中断 0 (/INT0)	0003H
T0 溢出中断	000BH
外部中断 1 (/INT1)	0013H
T1 溢出中断	001BH
串口中断	0023H

●外部中断源、定时/计数器的中断请求标志位分布在 **TCON** ；串口中断标志位分布在 **SCON** 寄存器中。

TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-------------	------------	------------	------------	------------	------------	------------	------------	------------

- TF1—— T1的溢出中断标志。
硬件置1，硬件清0（也可软件清0）。
- TF0—— T0的溢出中断标志。（同TF1，只是针对T0的）
- IE1 —— 外部中断1（/INT1）请求标志。
外部有中断请求时，硬件使IE1置1，硬件清0。
- IE0 —— 外部中断0（/INT0）请求标志。
- IT1 —— 外部中断1（/INT1）触发类型控制位。
IT1 = 0 ， 低电平触发。
IT1 = 1 ， 下降沿触发。
- IT0 —— 外中断0（/INT0）触发类型控制位，用法同IT1

SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-------------	------------	------------	------------	------------	------------	------------	-----------	-----------

- TI —— 串口发送中断标志位。发送完数据，硬件使TI置1，软件清0（CLR TI）
- RI —— 串行口接收中断标志位。硬件置1，软件清0。

◆2、中断控制(两级管理)

1、中断屏蔽

在中断源与CPU之间有一级控制，类似开关，其中第一级为一个总开关，第二级为五个分开关，由IE控制。

IE

EA			ES	ET1	EX1	ET0	EX0
----	--	--	----	-----	-----	-----	-----

- EA —— 总控制位
- ES —— 串口控制位 若为“1”，允许（开关接通）
- ET1—— T1 中断控制位 若为“0”，不允许（开关断开）
- EX1—— /INT1 控制位 例如，SETB EA
- ET0—— T0 中断控制位 CLR IE.7
- EX0—— /INT0 控制位

2、中断优先级

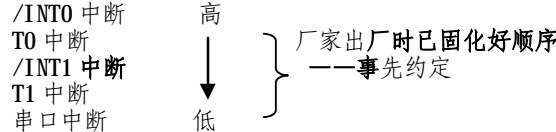
CPU 同一时间只能响应一个中断请求。为此将 5 个中断源分成高级、低级两个级别，高级优先，由 IP 控制。

IP

			PS	PT1	PX1	PT0	PX0
--	--	--	----	-----	-----	-----	-----

以上各位与 IE 的低五位相对应，为“1”时为高级。初始化编程时，由软件确定。
例如，SETB PT0 或 SETB IP1
CLR PX0 等。

同一级中的 5 个中断源的优先顺序是：



○中断优先原则：低级不中断高级；高级不睬低级；同级不能打断；同级、同时中断，事先约定。

◆3*、MCS-51 中断的响应过程 （略）

能否响应，还要看下述情况是否存在：

- (1) CPU正处理相同级别或更高级别的中断；
- (2) 正在执行指令，还未到最后一个机器周期；
- (3) 正在执行的指令是RETI或访问IP、IE指令，则执行完上述指令后，再执行一条指令后，才会响应新中断。

3) 计算时间常数 X(计算初值)

计数功能: $X = 2^n - \text{计数值}$ $n: 8/13/16$

定时功能: $X = 2^n - t/T$

t : 定时时间 (s) T : 机器周期 = $12/\text{晶振频率}$

如: 晶振为 12MHz 时, $T = 12/12 \text{ MHz} = 12 \div (12 \times 10^{-6})$

(秒) = $1 \times 10^{-6} = 1\mu\text{s}$

4) 定时器初始化编程: ①写 TMOD; ②确定 IE、IP; ③写计数初值; ④启动计数 (TRi)

MOV TMOD, # 方式字 ; 选择方式

MOV THx, #XH ; 装入 Tx 时间常数

MOV TLx, #XL

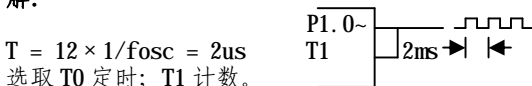
SETB EA ; 开 Tx 中断

SETB ETx

SETB TRx ; 启动Tx定时器。

[例] 设 $f_{\text{osc}} = 6\text{MHz}$, 利用单片机内定时/计数器及 P10 口线输出 1000 个脉冲, 脉冲周期为 2ms, 试编程。

解:



设 T0 采用中断方式产生周期为 2ms 方波, T1 对该方波计数, 当输出至第 1000 个脉冲时, 使 TF1 置 1。

在主程序中用查询方法, 检测到 TF1 变 1 时, 关掉 T0, 停止输出方波。

T0、T1 参数的确定:

T0 模式 0、定时: 脉宽为脉冲周期的一半。所以,

$$X = 213 - 1\text{ms} / 2\mu\text{s} = 0001\ 1110\ 0000\ 1100\text{B}$$

TH0 = 0F0H

TL0 = 0CH

T1 模式 1、计数: $N = 1000$

$$\text{则 } X = 65536 - 1000 = 64536 = 0\text{FC}18\text{H}$$

(若选模式 0 也可以, 此时 $X = 7192 = 1\text{C}18\text{H}$)

程序:

```
ORG 0000H
LJMP MAIN
ORG 000BH
LJMP TOS
ORG 1000H
MAIN: MOV TMD, #50H; T0 定时, 模式 0; T1 计数, 模式 1
      MOV TL0, #0CH
      MOV TH0, #0F0H
```

```

MOV TL1, #18H
MOV TH1, #0FCH
SETB TR1
SETB TR0
SETB ET0
SETB EA
WAIT: JNB TF1, WAIT ; 查询 1000 个脉冲计够没有? 没有 CLR EA
CLR ET0 ; 等待。
ANL TCON, #0FH ; 停 T0、T1
SJMP $
TOS: MOV TLO, #0CH
MOV THO, #0F0H
CPL P10
RETI
END

```

[例] 例 P1.7 驱动 LED 亮 1 秒灭 1 秒地闪烁，设时钟频率为 12MHz。

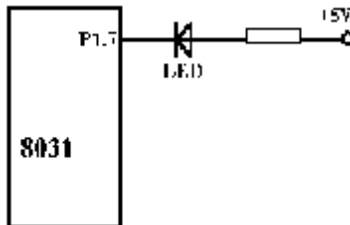
长定时方法：增加一个软件计数器（如 R7），记录中断次数，计满 n 个中断为 1 秒。

解：

```

ORG 0000H
AJMP MAIN
ORG 001BH
AJMP PT1INT
ORG 0030H
START: MOV R7, #00H
MOV TMD, #10H
MOV TL1, #0F0H
MOV TH1, #0D8H
SETB EA
SETB ET1
SETB TR1
HERE: SJMP HERE
PT1INT: MOV TL1, #0F0H
MOV TH1, #0D8H
INC R7
CJNE R7, #10, PEND
MOV R7, #00H
CPL P1.7
PEND: RETI

```



实战 电子钟设计

<实验内容>

1. 程序结构采用主程序和中断服务子程序结构.

2. 定时中断程序

定时器 1 于方式 1, 16 位, 10ms 中断一次

40H 10ms 计数单元

42H 分计数单元

41H 秒计数单元

43H 时计数单元

定时方式 1 为 10MS: THX=0ECH, TLX=78H

方式 0 为 10MS: THX=63H, TLX=18H

3. 参考程序;

DISP EQU 0DE00H ; 显示子程序入口

ORG 0000H

START: AJMP MAIN

ORG 001BH ; 定时器 1 中断服

AJMP TINT1 ; 务程序入口

ORG 0030H

MAIN: MOV R0, #39H ; 显示缓冲区初始化

MAIN_0: MOV @R0, #0

INC R0

CJNE R0, #44H, #0

MOV TMOD, #10H

MOV IE, #10001000B

MOV TH1, #0ECH; 10ms: EC78H

MOV TL1, #78H

SETB TR1

MAIN_1: LCALL DISP ; 主程序循环

SJMP MAIN_1

; 定时器 1 中断服务程序

TINT1: MOV TH1, #0ECH

MOV TL1, #78H

PUSH ACC

PUSH PSW

INC 40H

MOV A, 40H

CJNE A, #100, TINT1R

MOV 40H, #0

MOV A, 41H ; 秒单元加 1

INC A

DA A

MOV 41H, A

CJNE A, #60H, TINT10

MOV 41H, #0

MOV A, 42H ; 分单元加 1

INC A

```

DA A
MOV 42H, A
CJNE A, #60H, TINT10
MOV 42H, #0
MOV A, 43H ;小时单元加 1
INC A
DA A
MOV 43H, A
CJNE A, #24H, TINT10
MOV 43H, #0
TINT10: MOV A, 41H ;秒拆字
ANL A, #0FH
MOV 39H, A
MOV A, 41H
ANL A, #0FOH
SWAP A
MOV 3AH, A
MOV A, 42H ;分拆字
ANL A, #0FH
MOV 3BH, A
MOV A, 42H
ANL A, #0FOH
SWAP A
MOV 3CH, A
MOV A, 43H ;小时拆字
ANL A, #0FH
MOV 3DH, A
MOV A, 43H
ANL A, #0FOH
SWAP A
MOV 3EH, A
TINT1R: POP PSW
POP ACC
RETI
End

```

结束语：——尽信书不如无书！

哈哈，本资料约有5处错误，自己慢慢找吧，找出来的时候就真正入门了开始进军大虾吧。 ^_^

aki_studio 收集整理
2005年7月16 始 自学单片机 纪念