

## 1.1 基本概念

---

### 什么是单片机

---

ROM - Flash

RAM - SRAM

### MCU

---

MCU - Micro Controller Unit

## 1.2 发展概况

---

### 系列介绍

---

#### 8位

Intel MCS-51兼容单片机

Atmel AVR单片机 Arduino-UNO

#### 16位

Ti MSP430

#### 32位

ST STM32

Atmel AVR32 Arduino-MEGA2560

乐鑫 ESP32-C3 RISC-V

## 1.3 特点

---

## 1.4 趋势

---

低电压：3.3V - 1.8V **STM32L4**

高速：32位 高主频 **STM32H7/F7** OpenMV

集成性：A/D D/A **MSP430**

## 2.1 MCS-51单片机的基本组成

---

### 2.1.1 MCS-51单片机的基本组成

---

#### 1.中央处理器

#### 2.内部程序存储器

内部ROM

存放程序、原始数据、表格内容

#### 3.内部数据存储器

RAM的低128B

存放随机数据及运算结果

#### 4.定时/计数器

2个16位

#### 5.并行IO口

4个8位并行IO

P0: 低8位地址/8位数据, GPIO

P1: GPIO

P2: 高8位地址, GPIO

P3: 第二功能输入、输出, GPIO

#### 6.串行口

全双工异步通信收发

#### 7.中断控制系统

2个外部中断源

2个定时/计数器中断源

1个串行中断源

#### 8.时钟电路

外接石英晶体和微调电容

8051: 25MHz

89C51: 24MHz

89S51: 33MHz

9.位处理器

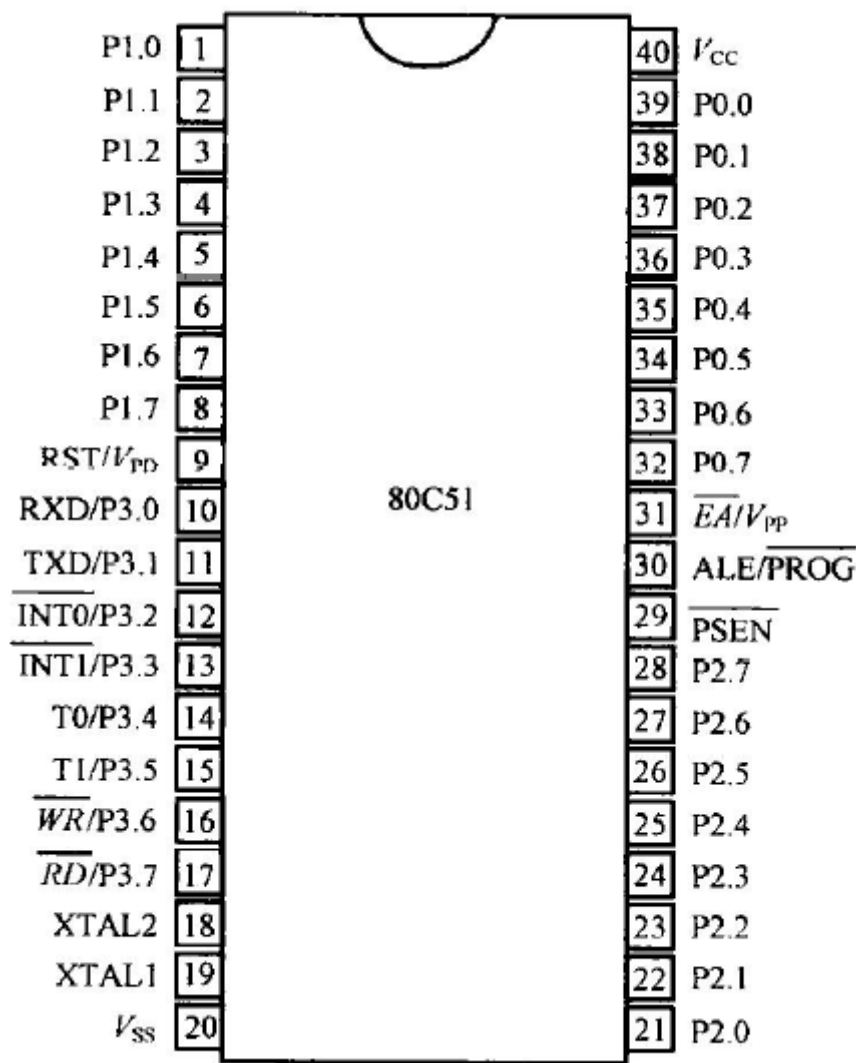
10.内部总线

2.1.2 封装及引脚

一、封装

DIP-40 双列直插

二、引脚



1.输入、输出

P0-P3

2.电源及时钟

$V_{CC}$ : +5V

$V_{SS}$ : GND

XTAL: 晶振

3.控制线和复位

ALE：访问外部存储器时锁存地址；不访问时输出振荡器频率1/6的正脉冲信号

$\overline{EA}$ ：EA为低时，只访问外部程序存储器；EA为高时，先访问内部，超出容量后访问外部

$\overline{PSEN}$ ：访问外部程序存储器时每周期两次有效

RST：两个周期高电平触发复位

三、第二功能

1.P3

引脚	第二功能	第二功能信号名称
P3. 0	RXD	串行口输入端
P3. 1	TXD	串行口输出端
P3. 2	$\overline{INT0}$	外部中断 0 请求输入端,低电平有效
P3. 3	$\overline{INT1}$	外部中断 1 请求输入端,低电平有效
P3. 4	T0	定时/计数器 0 的计数脉冲输入端
P3. 5	T1	定时/计数器 1 的计数脉冲输入端
P3. 6	$\overline{WR}$	外部 RAM 写选通信号输出端,低电平有效
P3. 7	$\overline{RD}$	外部 RAM 读选通信号输出端,低电平有效

2.烧录

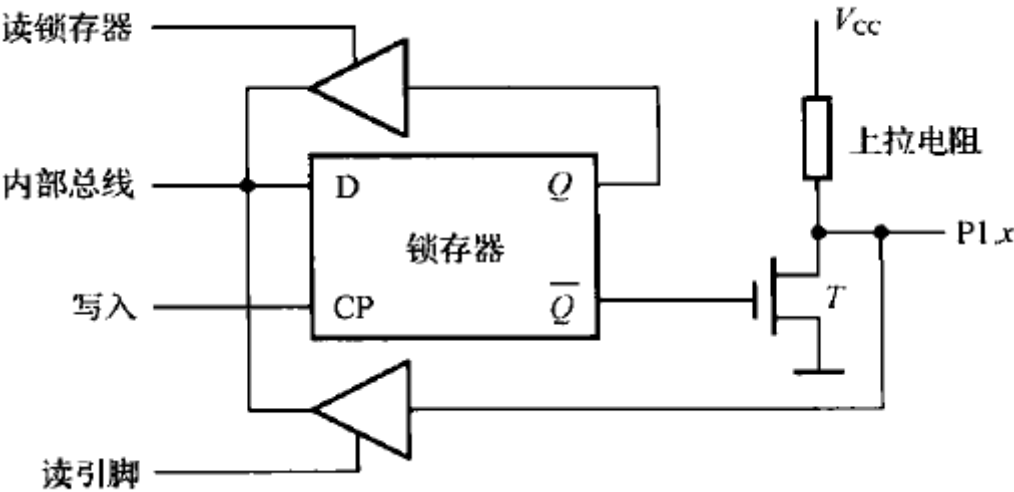
$\overline{PROG}$ 拉低, VPP接25V

3.备用电源

$V_{PD}$ 保护内部RAM不掉电

2.2 MCS-51单片机的并行I/O端口结构

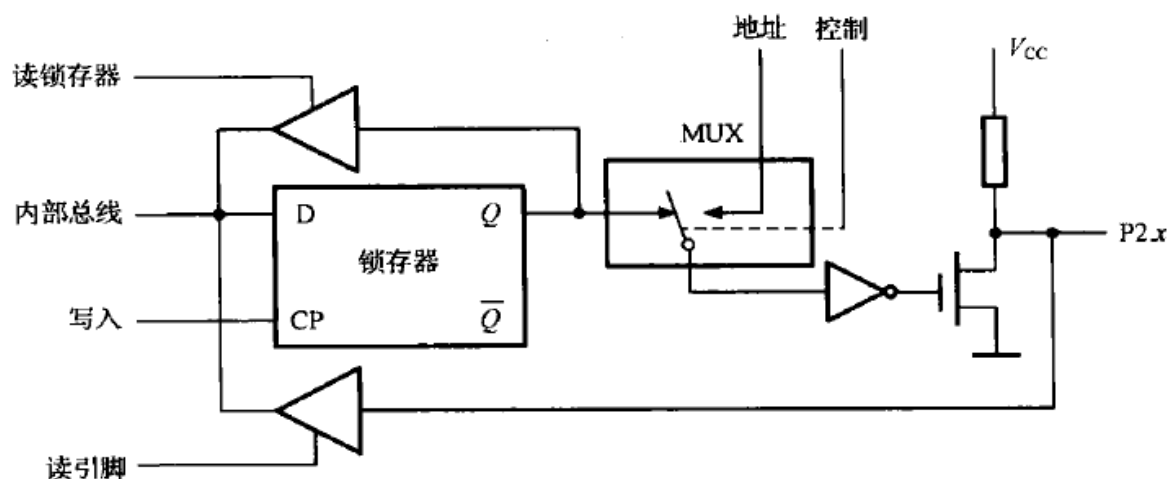
2.2.1 P1口



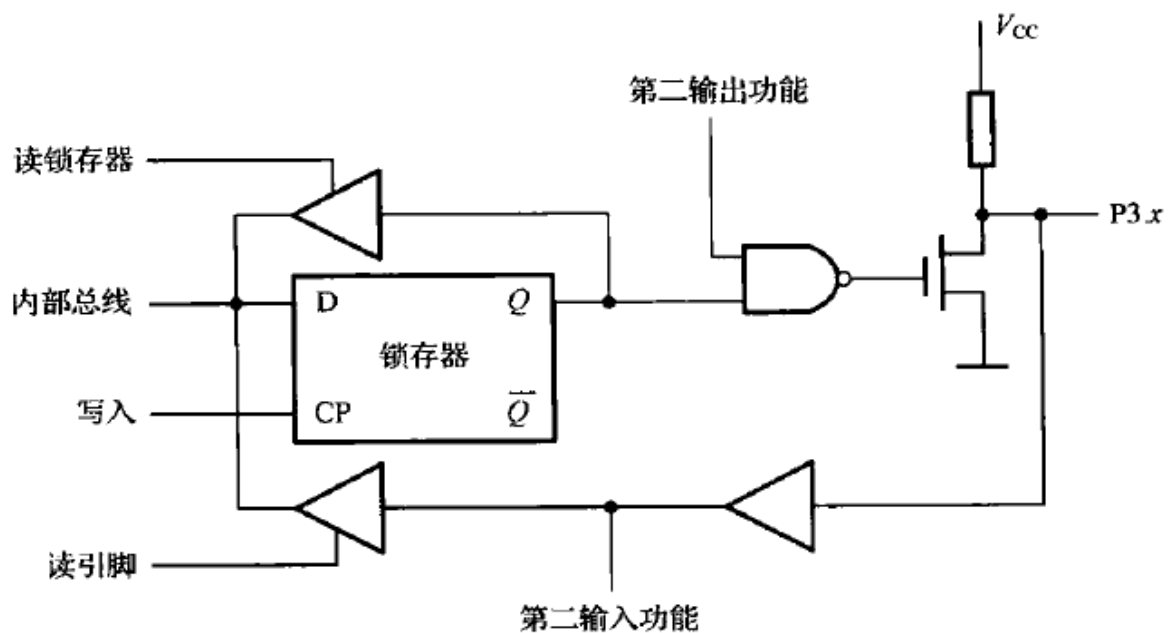
上拉电阻为场效应管等效

输入前应输出1防止短路（准双向I/O口）

## 2.2.2 P2口



## 2.2.3 P3口



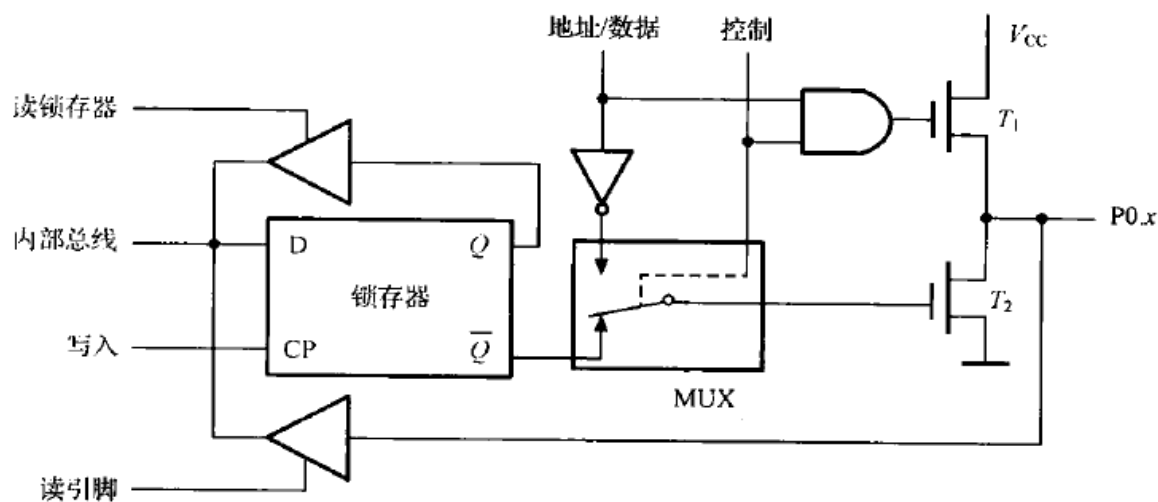
第一输出功能：第二输出功能信号线置1，与非门充当非门

第二输出功能：锁存器置1，与非门充当非门

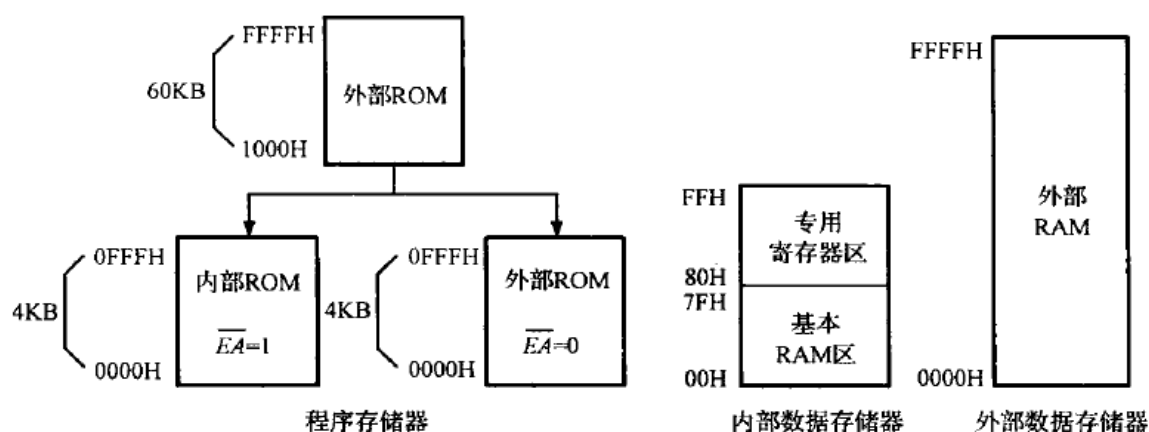
第一输入功能：锁存器置1，第二输出功能信号线置1，场效应管截止

第二输入功能：锁存器置1，第二输出功能信号线置1，场效应管截止

## 2.2.4 P0口



## 2.3 MCS-51单片机的存储器结构



### 2.3.1 程序存储器ROM

$\overline{EA} = 1$ , 单片机先执行片内4KB, 地址超出后自动转向片外

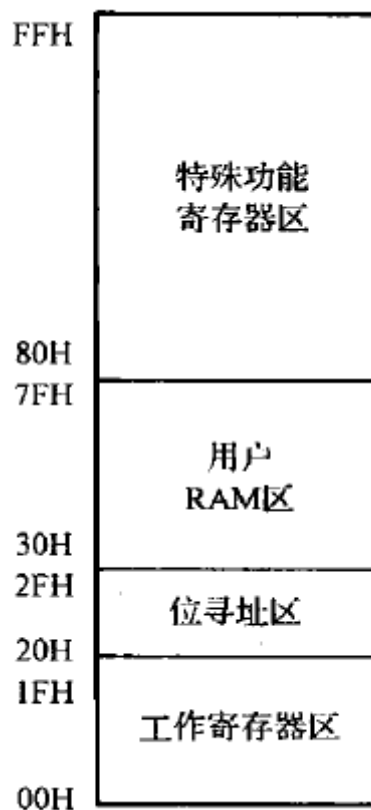
$\overline{EA} = 0$ , 单片机执行片外程序存储器内容

0003H-002AH单元均匀分为5段

特殊地址	功 能
0000H	主程序入口
0003H	外部中断 0 入口地址
000BH	定时/计数器 0 溢出中断入口地址
0013H	外部中断 1 入口地址
001BH	定时/计数器 1 溢出中断入口地址
0023H	串行口中断入口地址

主程序从0030H单元之后开始

### 2.3.2 数据存储器RAM



内部数据存储器：256B，00H-FFH，使用 `MOV` 访问

外部数据存储器：最大64KB，0000H-FFFFH，使用 `MOVX` 访问

## 一、基本RAM区

### 1.工作寄存器区 00H-1FH

RS1	RS0	工作寄存器组	工作寄存器地址
0	0	工作寄存器组 0	R0~R7 对应的地址为 00~07H
0	1	工作寄存器组 1	R0~R7 对应的地址为 08~0FH
1	0	工作寄存器组 2	R0~R7 对应的地址为 10~17H
1	1	工作寄存器组 3	R0~R7 对应的地址为 18~1FH

### 2.位寻址区 20H-2FH

16个单元，共16\*8=128位，既可以按字节寻址，也可以按位寻址

### 3.用户RAM区 30H-7FH

80个单元，按字节寻址

## 二、特殊功能寄存器区

### 1.程序计数器PC

16位，存放下一条指令地址，自动加1

## 2.累加器A/ACC

8位，E0H，可存放操作数

## 3.B寄存器

8位，F0H，主要用于乘除运算，也可作为一般数据寄存器使用

## 4.数据指针寄存器DPTR

16位

DPH：DPTR高位字节，83H

DPL：DPTR低位字节，82H

访问外部数据存储器时作地址指针使用

## 5.程序状态字寄存器PSW

8位，D0H

位序	PSW. 7	PSW. 6	PSW. 5	PSW. 4	PSW. 3	PSW. 2	PSW. 1	PSW. 0
位标志	CY	AC	F0	RS1	RS0	OV	—	P

### CY

进位标志位

### AC

辅助进位标志位

### F0

用户标志位，自行决定

### RS1、RS0

选择工作寄存器组

### OV

溢出标志位

### P

奇偶标志位，累加器A中1为偶数置0，反之置1

## 6.堆栈指针寄存器SP

8位，81H，存放堆栈栈顶地址

堆栈遵循LIFO，向上生长

进栈操作：PUSH，先SP加1，后写入数据

出栈操作：POP，先读出数据，后SP减1

SP初值为07H，但一般设置为30H-7FH地址空间的高端



## 2.4 MCS-51单片机的时钟电路与时序

### 2.4.1 时钟电路

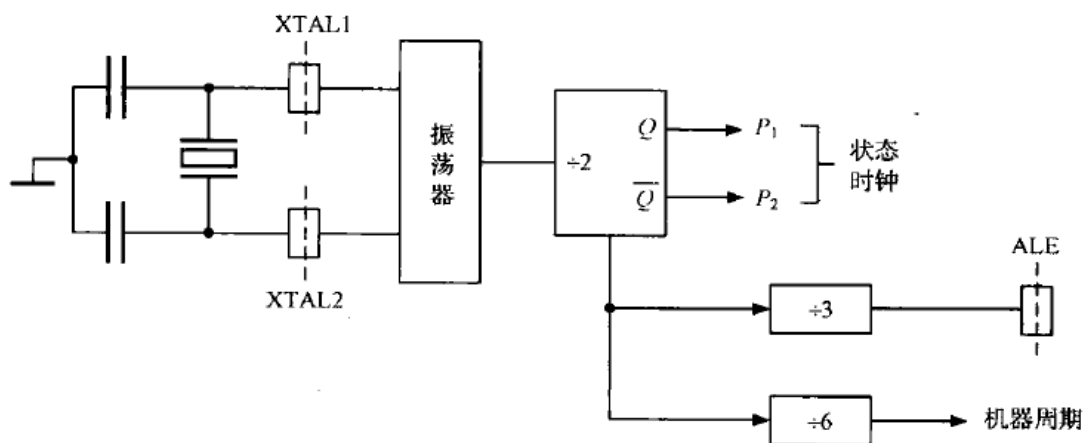
#### 一、内部时钟

XTAL1与XTAL2跨接晶体或陶瓷振荡器

晶体振荡器选择20-40pF电容，陶瓷振荡器选择30-50pF电容

#### 二、外部时钟

针对AT89，XTAL1接外部时钟，XTAL2悬空



### 2.4.2 时序

#### 一、振荡周期

晶振的振荡周期

#### 二、时钟周期

振荡脉冲二分频后的信号周期

#### 三、机器周期

一个机器周期 - 6个状态 - 12个节拍（振荡周期）

#### 四、指令周期

执行一条指令需要的时间

## 2.5 MCS-51单片机的工作方式

### 2.5.1 复位方式

## 一、复位信号

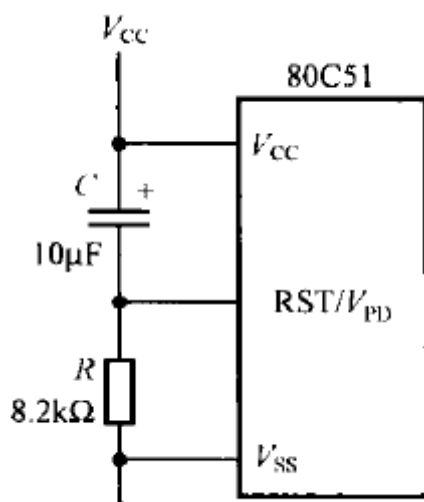
RST保持两个机器周期（24个振荡周期）以上高电平触发复位

## 二、复位操作

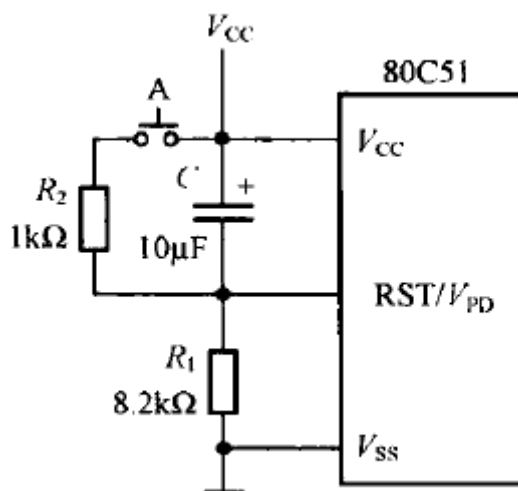
复位使得PC初始化至0000H，ALE与 $\overline{PSEN}$ 无效，不影响RAM

## 三、复位方式

### 1. 上电自动复位



### 2. 按键手动复位



## 2.5.2 程序执行方式

复位后系统指针指向0000H，在0000H处放置长转移指令转移至中断服务区后面（0030H）

## 2.5.3 单步执行方式

# 一、外部电路

按钮连接 $\overline{INT0}$ ，按钮按下高电平，放开低电平

# 二、中断程序

```
1  JNB P3.2,$ ;INT0为0时阻塞
2  JB  P3.2,$ ;INT0为1时阻塞
3  RETI      ;返回
```

## 2.5.4 低功耗方式

### 一、HMOS单片机掉电方式

主电源电压小于备用电源时，进入掉电中断，将有用数据保存至内部RAM中后由备用电源向RAM供电，时钟与CPU皆停止

主电源恢复后自动恢复现场

### 二、CHMOS单片机节电运行方式

由电源控制寄存器PCON（87H）控制

位序	PCON. 7	PCON. 6	PCON. 5	PCON. 4	PCON. 3	PCON. 2	PCON. 1	PCON. 0
位符号	SMOD	...	..	—	GF1	GF0	PD	IDL

PD=1，进入掉电方式

IDL=1，进入待机方式

#### 待机方式

振荡器工作，切断时钟到CPU的通路，中断、串口、定时器正常允许

退出：激活任意中断；复位

#### 掉电方式

振荡器停止

退出：复位

## 2.5.5 EPROM的编程和校验工作方式

# 3.1 指令格式及其符号说明

## 3.1.1 指令格式

[标号:] 助记符 [目的操作数],[源操作数] [:注释]

## 3.1.2 常用符号说明

符 号	含 义
Rn	表示当前选定寄存器组的工作寄存器 R0~R7, n=0~7
Ri	表示作为间接寻址的寄存器,只有 R0、R1 两个,即 i=0,1
A	累加器 A, ACC 则表示累加器 A 的地址
# data	表示 8 位立即数,即 00H~FFH
# data16	表示 16 位立即数,即 0000H~FFFFH
addr16	表示 16 位地址,可用于 64KB 范围内寻址,用于 LCALL 和 LJMP 指令中
addr11	表示 11 位地址,可用于 2KB 范围内寻址,用于 ACALL 和 AJMP 指令中
direct	片内 8 位 RAM 单元地址,包括特殊功能寄存器的地址或符号名称
rel	带符号的 8 位地址偏移量(−128~+127),用于 SJMP 和条件转移指令中
bit	位寻址区的直接寻址位,即内部 RAM(包括 SFR)中的可寻址位
(×)	表示×地址单元中的内容,或由×所指定的某寄存器的内容
((×))	由×间接寻址的单元中的内容
←	将箭头后面的内容传送到箭头前面去
\$	当前指令所在地址
/	加在位地址之前,表示该位状态取反
@	间接寻址寄存器或基址寄存器的前缀

## 3.1.3 指令的字节

一、单字节指令

二、双字节指令

三、三字节指令

# 3.2 寻址方式

## 3.2.1 立即寻址

```
1 MOV A , #25H ;将8位立即数25H赋值给累加器A
2 MOV DPTR , #1234H ;将16位立即数赋值给数据指针DPTR
```

## 3.2.2 直接寻址

1	MOV A , 50H	;将50H所存的数据赋值给累加器A。只能使用8位地址，因此只能寻址内部RAM
2	MOV A , P1	;直接寻址是访问特殊功能寄存器的唯一方法

### 3.2.3 寄存器寻址

1	MOV A , R7	;将寄存器R7的内容赋值给累加器A
---	------------	-------------------

### 3.2.4 寄存器间接寻址

1	MOV A , @R0	;将寄存器R0的内容作为地址，把该地址内容赋值给累加器A
---	-------------	------------------------------

R0, R1, DPTR可用于寄存器寻址

Ri可访问内部RAM低128字节与外部RAM低256字节

DPTR访问全部64KB外部RAM

### 3.2.5 变址寻址

1	MOVC A , @A+DPTR	;将累加器A与DPTR内容相加作为地址，将地址内容赋值给累加器A，@作用于A+DPTR
---	------------------	---

### 3.2.6 相对寻址

1	SJMP 50H	;将读取当前指令后的PC与操作数相加，得到地址后跳转
---	----------	----------------------------

### 3.2.7 位寻址

## 3.3 MCS-51单片机指令系统

### 3.3.1 数据传送类指令

#### 一、普通传送指令

##### 1. 片内数据存储器传送指令

1	MOV A , #data	;立即寻址
2	MOV A , direct	;直接寻址
3	MOV A , Rn	;寄存器寻址
4	MOV A , @Ri	;寄存器间接寻址
5		
6	MOV Rn , #data	;立即寻址
7	MOV Rn , direct	;直接寻址
8	MOV Rn , A	;寄存器寻址
9		
10	MOV direct , #data	;立即寻址
11	MOV direct , direct	;直接寻址
12	MOV direct , A	;寄存器寻址
13	MOV direct , Rn	;寄存器寻址

```

14 MOV direct , @Ri      ;寄存器间接寻址
15
16 MOV @Ri , #data      ;立即寻址
17 MOV @Ri , direct      ;直接寻址
18 MOV @Ri , A           ;寄存器寻址
19
20 MOV DPTR , #data16     ;立即寻址

```

## 2.片外数据存储器传送指令

只能通过累加器A与片外数据存储器数据传送，且只能通过@Ri与@DPTR间接寻址

```

1 MOVX A , @Ri           ;只能寻址低8位
2 MOVX @Ri , A           ;只能寻址低8位
3 MOVX A , @DPTR
4 MOVX @DPTR , A

```

## 3.程序存储器传送指令

```

1 MOVC A , @A+DPTR       ;寻址64KB
2 MOVC A , @A+PC         ;寻址256B

```

# 二、数据交换指令

## 1.整字节交换指令

8位数据交换

```

1 XCH A , Rn
2 XCH A , direct
3 XCH A , @Ri

```

## 2.半字节交换指令

低4位数据交换

```

1 XCHD A , @Ri

```

## 3.累加器高低半字节交换指令

```

1 SWAP A

```

# 三、堆栈操作指令

```

1 PUSH direct           ;SP+1,存入direct
2 POP direct             ;读取direct,SP-1

```

## 3.3.2 算术运算类指令

## 一、加法指令

### 1.不带进位

1	ADD A , #data	;A=(A)+data
2	ADD A , direct	;A=(A)+(direct)
3	ADD A , Rn	;A=(A)+(Rn)
4	ADD A , @Ri	;A=(A)+((Ri))

第7位有进位, CY=1

第3位有进位, AC=1

第6、7位有一个进位, OV=1

### 2.带进位

1	ADDC A , #data	;A=(A)+data+(CY)
2	ADDC A , direct	;A=(A)+(direct)+(CY)
3	ADDC A , Rn	;A=(A)+(Rn)+(CY)
4	ADDC A , @Ri	;A=(A)+((Ri))+(CY)

### 3.加1指令

1	INC A
2	INC direct
3	INC Rn
4	INC @Ri
5	INC DPTR

## 二、减法指令

### 1.带借位的减法指令

1	SUBB A , #data	;A=(A)-data-(CY)
2	SUBB A , direct	;A=(A)-(direct)-(CY)
3	SUBB A , Rn	;A=(A)-(Rn)-(CY)
4	SUBB A , @Ri	;A=(A)-((Ri))-(CY)

第7位有借位, CY=1

第3位有借位, AC=1

第6、7位有一个借位, OV=1

### 2.减1指令

1	DEC A
2	DEC direct
3	DEC Rn
4	DEC @Ri

## 三、乘法指令

```
1 | MUL AB
```

将累加器A与寄存器B相乘，得到16位乘积，高位在A，低位在B

## 四、除法指令

```
1 | DIV AB
```

被除数为A，除数为B；执行后商为A，余数为B

## 五、十进制调整指令

```
1 | MOV A , #93H
2 | ADD A , #59H
3 | DA A
```

将A调整为BCD码

## 3.3.3 逻辑运算及移位类指令

### 一、与

```
1 | ANL A , #data
2 | ANL A , direct
3 | ANL A , Rn
4 | ANL A , @Ri
5 | ANL direct , #data
6 | ANL direct , A
```

将源操作数与目标操作数按位取与

### 二、或

```
1 | ORL A , #data
2 | ORL A , direct
3 | ORL A , Rn
4 | ORL A , @Ri
5 | ORL direct , #data
6 | ORL direct , A
```

将源操作数与目标操作数按位取或

### 三、异或

```
1 | XRL A , #data
2 | XRL A , direct
3 | XRL A , Rn
4 | XRL A , @Ri
5 | XRL direct , #data
6 | XRL direct , A
```



将源操作数与目标操作数按位取异或

## 四、累加器清零与取反

1	CLR A	; 清零
2	COL A	; 取反

## 五、循环移位

1	RL A	; 累加器循环左移
2	RR A	; 累加器循环右移
3	RLC A	; 带进位累加器循环左移
4	RRC A	; 带进位累加器循环右移

## 3.3.4 控制转移类指令

### 一、无条件转移

#### 1. 长转移指令

1	LJMP addr16	; 将16位地址送PC
---	-------------	-------------

#### 2. 绝对转移指令

1	AJMP addr11	; PC高5位不变，赋值低11位
---	-------------	------------------

#### 3. 短转移指令

1	SJMP rel	; PC当前值加rel，rel是补码形式有符号数
---	----------	--------------------------

#### 4. 变址寻址转移指令

1	JMP @A+DPTR	; 将A与DPTR内容相加得到地址
---	-------------	-------------------

### 二、条件转移

#### 1. 累加器判零转移指令

1	JZ rel	; 若A=0则PC=PC+2+rel
2	JNZ rel	; 若A≠0则PC=PC+2+rel

#### 2. 比较转移指令

1	CJNE A, #data, rel	; 不等时转移，若目的操作数大于等于源操作数，CY=0；目的操作数小于源操作数，CY=1
2	CJNE A, direct, rel	; 可使用该特性实现数值比较
3	CJNE Rn, #data, rel	
4	CJNE @Ri, direct, rel	

### 3.减1非零转移指令

```
1 DJNZ Rn , rel ;Rn-1后不为0则跳转
2 DJNZ direct , rel
```

## 三、子程序调用及返回

### 1.长调用指令

```
1 LCALL addr16 ;PC+3; 分两次将PC压入堆栈，先压低位；将地址送入PC
```

### 2.绝对调用指令

```
1 ACALL addr11 ;PC+2; 分两次将PC压入堆栈，先压低位；将11位地址送入PC低位
```

### 3.子程序返回指令

```
1 RET ;自动从堆栈中取出PC
```

### 4.中断返回指令

```
1 RETI ;自动从堆栈中取出PC
```

## 四、空指令

```
1 NOP ;延迟一个机器周期
```

## 3.3.5 位操作类指令

### 一、位传送指令

```
1 MOV C , bit ;实现可寻址位与位累加器CY之间的传送
2 MOV bit , C
```

### 二、位置位与清零指令

```
1 SETB C ;置1
2 SETB bit
3 CLR C ;置0
4 CLR bit
```

### 三、位运算指令

```
1 ANL C , bit ;CY与bit
2 ANL C , /bit ;CY与bit非
3 ORL C , bit ;CY或bit
4 ORL C , /bit
5 CPL C ;C非
6 CPL bit
```

## 四、位控制转移指令

### 1.以C为条件

1	JC rel	;若CY=1则跳转
2	JNC rel	;若CY=0则跳转

### 2.以位地址内容为条件

1	JB bit , rel	;bit=1则跳转
2	JNB bit , rel	;bit=0则跳转
3	JBC bit , rel	;bit=1则跳转且bit清零

## 4.1 汇编语言的概述

### 4.1.1 汇编语言的特点

### 4.1.2 汇编语言的伪指令

#### 1.ORG

确定下一条指令的地址

#### 2.DB

保存字节常数或字符或表达式

```
1 DB 73H,01H,90H
2 DB "Hello!"
```

#### 3.DW

保存字数据（16位）

```
1 DB 1234H,90H
```

#### 4.EQU

赋值标号

```
1 AREA EQU 1000H
2 STK EQU AREA
```

#### 5.END

源程序结束

#### 6.DATA

指定的数据地址赋予规定的字符名称

#### 7.DS

保留指定表达式的若干字节空间

```
1 ORG 1000H
2 DS 07H
3 DB 20H,25H,33H,46H ;1007H为20H
```

## 8.BIT

将位地址赋予字符名

1	FLG BIT 20H	;标志位FLG定义在位地址20H
2	AI BIT P1.0	;把P1.0的位地址赋值给AI

## 4.2 汇编语言源程序的编辑和汇编

### 4.2.1 手工编程和汇编

### 4.2.2 机器编辑和交叉汇编

## 4.3 汇编语言程序设计

### 4.3.1 简单程序设计

#### 1.BCD转十进制

片内RAM内35H单位中BCD码转换至片内RAM中40H单元中

1	ORG 0800H	;代码所在位置
2	BCD EQU 35H	;数据源地址
3	BIN EQU 40H	;数据目标地址
4	START:	
5	MOV A , BCD	;BCD内容赋值A
6	ANL A , #F0H	;取高四位
7	SWAP A	;交换高低四位
8	MOV B , #0AH	;B=10
9	MUL AB	;10*原本高位，得16位乘积，高位在A
10	MOV R0 , A	;R0=A
11	MOV A , BCD	;BCD内容赋值A
12	ANL A , #0FH	;取低四位
13	ADD A , R0	;A=A+R0
14	MOV BIN , A	;BIN地址赋值A
15	SJMP \$	;跳转至当前地址（死循环）
16	END	

#### 2.BCD转ASCII

片内RAM内20H单位中BCD码转换至片内RAM中21H、22H单元中

1	ORG 1000H
2	BCD EQU 20H
3	ASC EQU 21H
4	START:
5	MOV A , BCD
6	ANL A #0FH
7	ADD A #30H

```

8      MOV ASC , A
9      MOV A , BCD
10     ANL A #F0H
11     SWAP A
12     ADD A #30H
13     MOV ASC+1 , A
14     SJMP $
15     END

```

## 4.3.2 分支程序设计

### 1.单/双分支

在片内RAM中40H处存在X，按规则将Y写入片内RAM中41H处

$$Y = \begin{cases} 1, X > 0 \\ 0, X = 0 \\ -1, X < 0 \end{cases}$$

```

1      ORG 1000H
2  START:
3      MOV A 40H
4      JZ DONE ;A=0时跳转
5      JNB ACC.7 , POST ;A[7]=0（正数）时跳转
6      MOV A , #FFH ;A=-1
7      SJMP DONE
8  POST:
9      MOV A , #01H
10     DONE:
11     MOV 41H , A
12     SJMP $
13     END

```

### 2.多分支

P1口：被操作数、结果低8位

P3口：操作数、结果高8位

R2：0-加 1-减 2-乘 3-除

```

1      ORG 1000H
2  START:
3      MOV P1 , #DATA1
4      MOV P2 , #DATA2
5      MOV DPTR , #TABLE
6      MOV A , R2
7      RL A
8      JMP @A+DPTR
9  TABLE:
10     AJMP PRG0
11     AJMP PRG1
12     AJMP PRG2
13     AJMP PRG3
14  PRG0:
15     MOV A , P1 ;A=P1
16     ADD A , P3 ;A=A+P3

```

```

17      MOV P1 , A          ;P1=A
18      CLR A
19      ADDC A , #00H       ;A=A+CY
20      MOV P3 , A          ;P3=A
21      RET
22  PRG1:
23      MOV A , P1
24      CLR C
25      SUBB A , P3
26      MOV P1 , A
27      CLR A
28      RLC A
29      MOV P3 , A
30      RET
31  PRG2:
32      MOV A , P1
33      MOV B , P3
34      MUL AB
35      MOV P1 , A
36      MOV P3 , B
37      RET
38  PRG3:
39      MOV A , P1
40      MOV B , P3
41      DIV AB
42      MOV P1 , A
43      MOV P3 , B
44      RET

```

### 4.3.3 循环程序设计

#### 一、单重循环

1. 片内RAM38H-47H存放16个二进制无符号数，求和存放R4、R5

```

1      ORG 0800H
2  START: MOV R0,#38H      ;起始地址
3          MOV R2,#10H     ;计次循环
4          MOV R4,#00H
5          MOV R5,#00H
6  LOOP:  MOV A,R5          ;低位送A
7          ADD A,@R0        ;A=A+R0
8          MOV R5,A         ;R5=A
9          CLR A            ;A=0
10         ADDC A,R4         ;A=A+R4+CY
11         MOV R4,A         ;R4=A
12         INC R0           ;R0++
13         DJNZ R2,LOOP      ;if R2!=0 goto LOOP
14         SJMP $           ;阻塞
15         END

```

2. 片内RAM的30H-4FH传送至片外1800H开始

```

1      ORG 1000H
2  START: MOV R0, #30H
3          MOV DPTR, #1800H      ;访问片外RAM应使用MOVX，并使用DPTR做地址
4          MOV R2, #20H
5  LOOP: MOV A, @R0
6          MOVX @DPTR, A
7          INC R0
8          INC DPTR
9          DJNZ R2, LOOP
10         SJMP $
11        END

```

## 二、多重循环

设计50ms延时程序

```

1      ORG 0800H
2  DELAY: MOV R7, #200      ;外循环次数，1T
3  DLY1:  MOV R6, #123      ;内循环次数，1T
4  DLY2:  DJNZ R6, DLY2      ;R6不为1时循环，2T
5          NOP              ;1T
6          DJNZ R7, DLY1      ;R7不为1时循环，2T
7          RET              ;2T

```

$f_{OSC} = 12MHz$ 时，1T为1us

总延迟=(246+2+1+1)T\*200+2T+1T=50.003ms

### 4.3.4 数值转换程序

#### 一、BIN - BCD

入口：16bit无符号数R3、R2

出口：R6、R5、R4

```

1  BINBCD: CLR A
2          MOV R4, A
3          MOV R5, A
4          MOV R6, A
5          MOV R7, #10H      ;计次
6  LOOP:   CLR C
7          MOV A, R2
8          RLC A
9          MOV R2, A
10         MOV A, R3
11         RLC A
12         MOV R3, A
13         MOV A, R4
14         ADDC A, R4
15         DA A
16         MOV R4, A
17         MOV A, R5
18         ADDC A, R5
19         DA A
20         MOV R5, A

```



```

21      MOV A,R6
22      ADDC A,R6
23      MOV R6,A
24      DJNZ R7,LOOP
25      RET

```

## 二、BCD - BIN

## 三、ASCII - BIN

```

1  ASCBIN: MOV A,R1      ;A=R1
2          CLR C
3          SUBB A,#30H    ;A-30H
4          MOV R1,A      ;R1=A
5          SUBB A,#0AH    ;A-10
6          JC LOOP      ;IF CY=1(A<10) GOTO LOOP
7          XCH A,R1      ;A=R1
8          SUBB A,#07H    ;A-7
9          MOV R1,A      ;R1=7
10 LOOP:  RET

```

## 4.3.5 查表程序设计

查表计算  $Y = X^2$

```

1      ORG 1000H
2  START: MOV A,30H
3          MOV DPTR,#TABLE
4          MOVC A,@A+DPTR
5          MOV 31H,A
6  TABLE: DB 0,1,4,9,16,25,36,49,64,81
7          END

```

```

1      ORG 1000H
2  START: MOV A,30H
3          ADD A,#02H
4          MOVC A,@A+PC
5          MOV 31H,A
6  TABLE: DB 0,1,4,9,16,25,36,49,64,81
7          END

```