第10章 单片机的C语言编程

内 容 提 要

- ★ C51程序结构
- ★ C51的数据类型
- ★ 数据的存贮类型和存贮模式
- ★ C51对SFR、可寻址位、存储器和I/O口的定义
- ★ 函 数
- ★ 单片机资源的C语言编程实例
- ★ 汇编语言和C语言的混合编程

51系列单片机支持三种高级语言，即PL/M，C和BASIC。

C语言是一种通用的程序设计语言，其代码率高，数据类型及运算符丰富，并具有良好的程序结构，适用于各种应用的程序设计，是目前使用较广的单片机编程语言。

单片机的C语言采用C51编译器(简称C51)。由C51产生的目标代码短、运行速度高、所需存储空间小、符合C语言的ANSI标准，生成的代码遵循Intel目标文件格式，而且可与A51汇编语言或PL/M51语言目标代码混和使用。

应用C51编程具有以下优点：

（1）C51管理内部寄存器和存贮器的分配，编程时，无需考虑不同存储器的寻址和数据类型等细节问题；

（2）程序由若干函数组成，具有良好的模块化结构；

（3）有丰富的子程序库可直接引用，从而大大减少用户编程的工作量。

（4）C语言和汇编语言可以交叉使用. 汇编语言程序运行速度快、但复杂运算编程耗时。如果用汇编语言编写与硬件有关的部分程序，用C语言编写与硬件无关的运算部分程序，充分发挥两种语言的长处，可以提高开发效率。

10.1 C51程序结构

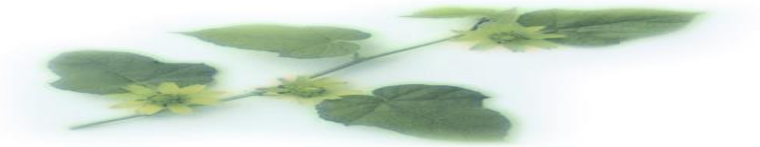
同标准C一样，C51的程序由一个个函数组成，这里的函数和其他语言的“子程序”或“过程 ”具有相同的意义。

其中必须有一个主函数main( )，程序的执行从main()函数开始，调用其他函数后返回主函数main()，不管函数的排列顺序如何，最后在主函数中结束整个程序。

C语言程序的组成结构如下所示：

```
全局变量说明    /*可被各函数引用*/
main()          /*主函数*/
{
局部变量说明    /*只在本函数引用*/
执行语句(包括函数调用语句)
}

fun1(形式参数表)  /*函数1*/
形式参数说明
{
局部变量说明
执行语句(包括调用其他函数语句)
}
…
funn(形式参数表)        /*函数n*/
形式参数说明
{
局部变量说明
执行语句
}
```



10.2 C51的数据类型

C51的数据有常量和变量之分。

常量—在程序运行中其值不变的量，可以为字符，十进制数或十六进制数(用0x表示)。

变量—在程序运行中其值可以改变的量。

一个变量由变量名和变量值构成，变量名即是存贮单元地址的符号表示，而变量的值就是该单元存放的内容。

定义一个变量，如果没有指定地址，编译系统就会自动为它安排一个存贮单元。

10.2.1 C51变量的数据类型

	数 据 类 型	长 度	值 域
位 型	bit	1Bit	0 或 1
字符型	signed char	1Byte	-128~127
	unsigned char	1Byte	0~255
整型	signed int	2Byte	-32768~+32767
	unsigned int	2Byte	0~65535
	signed long	4Byte	-2147483648~+2147483647
	unsigned long	4Byte	0~4294967295
实型	Float	4Byte	1.176E-38~3.40E+38
指针型	data/idata/ pdata	1Byte	1 字节地址
	code/xdata	2Byte	2 字节地址
	通用指针	3Byte	其中 1 字节为储存器类型编码，2,3 字节为地址偏移量
访问 SFR 的数据类型	sbit	1Bit	0 或 1
	sfr	1Byte	0~255
	sfr16	2Byte	0~65535

10.2.2 关于指针型数据

(1)关于指针型变量

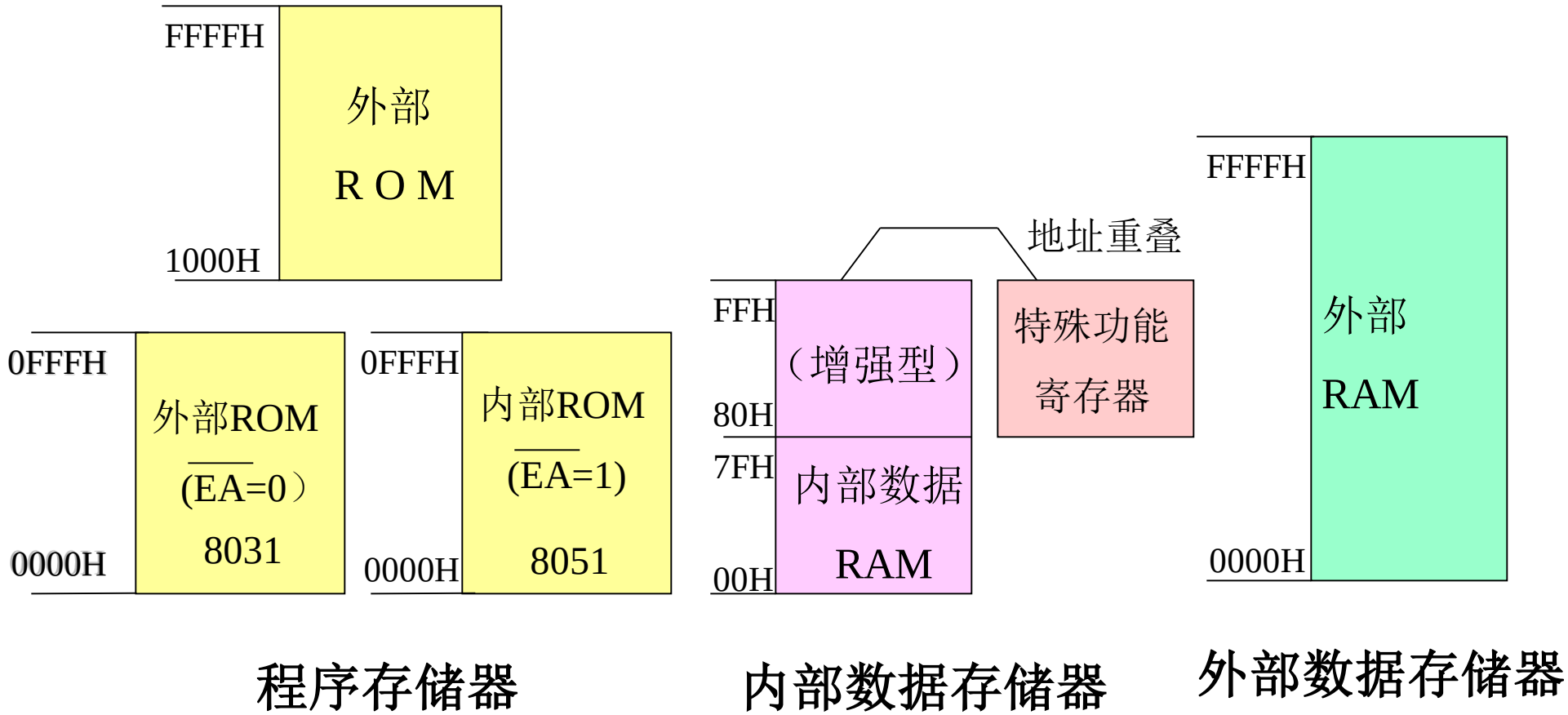
下面表格表示两种语言将m单元的内容送n单元的对照语句。

直接寻址		间 接 寻 址	
汇编语言	C语言	汇编语言	C语言
mov n,m 传送语句	n=m; 赋值语句	mov R1,#m ; m的地址送R1 mov n,@R1 ; m的内容送n	P=&m /*m的地址送P*/ n=*P /*m的内容送n*/

10.3 数据的存贮类型和存贮模式

10.3.1数据的存贮位置类型

C51是面向8XX51系列单片机及硬件控制系统的开发语言，它定义的任何变量必须以一定的存贮类型的方式定位在8XX51的某一存贮区中，否则便没有意义。因此在定义变量类型时，还必须定义它的存贮类型，C51的变量的存贮类型如表10-4所示：



程序存储器　　内部数据存储器　　外部数据存储器

表10-4 C51的变量的存贮类型

存贮器类型	描　　述
data	直接寻址内部数据存贮区，访问变量速度最快(128bytes)
bdata	可位寻址内部数据存贮区，允许位与字节混合访问(16bytes)
idata	间接寻址内部数据存贮区，可访问全部内部地址空间(256bytes)
pdata	分页(256bytes)外部数据存贮区，由操作码MOVX@Ri访问
xdata	外部数据存贮区(64K)，由操作码MOVX@DPTR访问
code	代码存贮区(64K)，由操作码MOVC@A+DPTR访问

访问内部数据存贮器(idata)比访问外部数据存贮器(xdata)相对要快一些。

例如定义变量x语句：**data char x** (等价于**char data x**)。

10.3.2 程序的存贮器模式

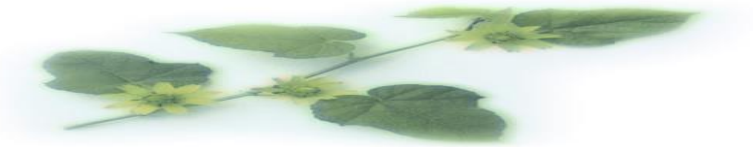
存贮器模式决定了变量的默认存贮器类型、参数传递区和无明确存贮区类型的说明。C51的存贮器模式有SMALL、LARGE和COMPACT（见表10-

存贮器模式	描　　述
SMALL	参数及局部变量放入可直接寻址的内部存贮器(最大128bytes，默认存贮器类型DATA)
COMPACT	参数及局部变量放入分页外部存贮区(最大256bytes，默认存贮器类型是PDATA)
LARGE	参数及局部变量直接放入外部数据存贮器(最大64K)，默认存贮器类型为XDATA)

例如设C语言源程序为PROR.C，若使程序中的变量类型和参数传递区限定 在外部数据存贮区，有两种方法：

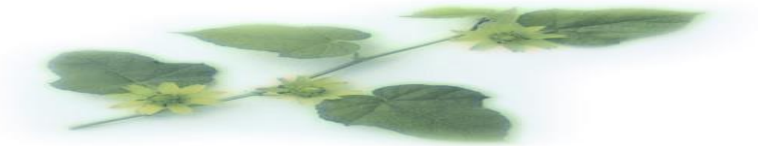
方法1：用C51对PROR.C进行编译时，使用命令C51 PROR.C COMPACT。

方法2：在程序的第一句加预处理命令 #pragma compact



10.3.3 变量说明举例

```
data char var;                          /*字符变量var定位在片内数据存贮区*/
char code MSG[]="PARAMETER:";           /*字符数组MSG[ ]定位在程序存 贮区*/
unsigned long xdata array[100];         /*无符号长型数组定位在片外RAM区*/
float idata x,y,z;                      /*定位在片内用间址访问的内部RAM区*/
bit lock;                               /*定位在片内RAM可位寻址区*/
unsigned int pdata sion;                /*定位在分页的外部RAM*/
unsigned char xdata ve[10][4][4]        /*定位在片外RAM区*/
sfr P0=0x80;                            /*定义P0口，地址为80H*/
char bdata flags;                       /*字符变量flags定位在可位寻址内部RAM区*/
sbit flag0=flags^0;                     /*定义flag0为flags.0 */
```



如果在变量说明时略去存贮器类型标志符，编译器会自动选择默认的存贮器类型。

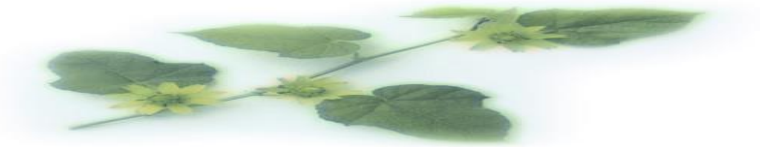
默认的存贮器类型由控制指令SMALL、COMPACT和LARGE限制。

例如如果声明char var

则默认的存贮器模式为SMALL，var放在data存贮区；

如果使用COMPACT模式，var放入idata存贮区；

在使用LARGE模式的情况下，var被放入外部数据存贮区(xdata存贮区)。



指针变量的类型和存贮位置

long xdata *px;　　/*指针px指向long型xdata区(每个数据占四个单元，指针自身在默认存贮器(如不指定编译模式在data区)，指针长度为2个字节*/

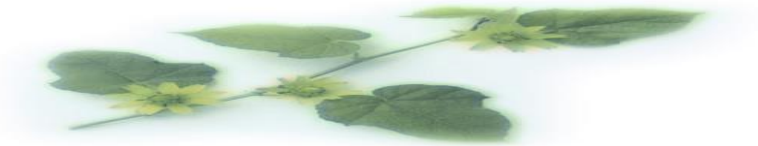
char xdata *data pd；/*指针pd指向xdata区，指针在data区，指针长度2字节*/

data char xdata *pd;　　/*与上例等效*/

data int *pn；(或int *data pn)　　/*定义一个类型为int型的通用型指针，指针1个字节在data区，总长度为3字节*/

比如：外部地址4080H开始的int类型存储器，int **xdata *px；px=0x4080;**

比如：内部地址20H开始的char类型存储器，data **char *pn；　pn=0x20;**



10.4 C51对SFR、可寻址位、存储器和I/O口的定义

10.4.1 特殊功能寄存器SFR定义

C51提供了一种自主形式的定义方式，使用特定关键字sfr

如 sfr SCON=0x98； /*串行通信控制寄存器地址98H*/

sfr TMOD=0x89； /*定时器模式控制寄存器地址89H*/

sfr ACC=0xe0； /*A累加器地址E0H*/

sfr P1=0x90； /*P1端口地址90H*/

定义了以后，程序中就可以直接引用寄存器名。

C51也建立了一个头文件reg51.h (增强型为reg52.h)，在该文件中对所有的特殊功能寄存器的进行了sfr定义，在编程时可以直接引用特殊功能寄存器名，或直接引用位名称。

要特别注意：在引用 时特殊功能寄存器或者位名称必须大写。

10.4.2 对位变量的定义

C51对位变量的定义有三种方法：

1. 将变量用bit类型的定义符定义为bit类型：

如 bit mn;

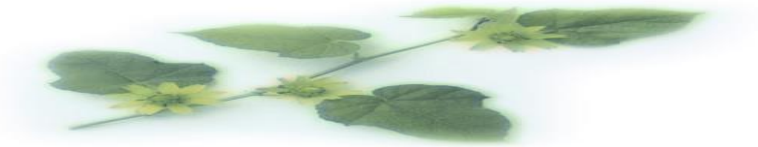
mn为位变量，其值只能是“0”或“1”，其位地址C51自行安排在可位寻址区的bdata区。

2. 采用字节寻址变量位的方法：

如 bdata int ibase; /*ibase定义为整型变量*/

sbit mybit=ibase^15; /*mybit定义为ibase的D15位*/

这里位是运算符“^”相当于汇编中的“.”，其后的最大取值依赖于该位所在的字节寻址变量的定义类型，如定义为char最大值只能为7。



3. 对特殊功能寄存器的位的定义

方法1：使用头文件及sbit定义符；多用于无位名的可寻址位。

例如 #include <reg51.h>

```
sbit P1-1=P1^1;      /*P1-1为P1口的第1位*/
sbit ac=ACC^7;       /*ac定义为累加器A的第7位*/
```

方法2：使用头文件reg51.h，再直接用位名称。

例如 #include <reg51.h>

```
RS1=1;
RS0=0;
```

方法3：用字节地址位表示

例如 sbit OV=0xD0^2;

方法4：用寄存器名.位定义

```
例如 sfr PSW=0xd0;        /*定义PSW地址为d0H*/
     sbit CY=PSW^7;       /*CY为PSW 7*/
```

10.4.3 C51对存贮器和外接I/O口的绝对地址访问

1.对存贮器的绝对地址访问

利用绝对地址访问的头文件absacc.h可对不同的存贮区进行访问。

该头文件的函数有：

CBYTE （访问code区字符型）　　DBYTE （访问data区字符型）

PBYTE （访问pdata或I/O区字符型）

XBYTE （访问xdata或I/O区字符型）

还有CWORD、DWORD、PWORD和XWORD四个函数，它们的访问区域同上，只是访问的类型为int 型。

例10-1

```
#include<absacc.h>
#define com XBYTE [0x07ff]
```

那么后面程序com变量出现的地方，就是对地址为07ffH的外部RAM或I/O口进行访问。

例10-2 XWORD［0］=0x9988；

即将9988H(int类型)送入外部RAM的0号和1号单元。

使用中要注意：absacc.h一定要包含进程序，XBYTE必须大写。

2.对外部I/O口的访问

由于单片机的I/O口和外部RAM统一编址，因此对I/O口地址的访问可用XBYTE (MOVX @ DPTR)或PBYTE (MOVX @Ri)进行。

例10-3 XBYTE［0xefff］=0x10；将10H输出到地址为EFFFH端口

10.6 函 数

C语言程序由函数组成，下面介绍函数的要点。

10.6.1 函数的分类及定义

从用户使用角度划分，函数分为库函数和用户自定义函数。

库函数是编译系统为用户设计的一系列标准函数(见本书附录二)，用户只需调用，而无需自己去编写这些复杂的函数。

使用#include包含语句，然后才能调用。

用户自定义函数是用户根据任务编写的函数

从参数形式上函数分为无参函数和有参函数。

有参函数即是在调用时，调用函数用实际参数代替形式参数，调用完返回结果给调用函数。

10.6.2 函数的定义

函数以“{”开始，以“}”结束。

无参函数的定义：

返回值类型 函数名( )
{函数体语句}

如果函数没有返回值，可以将返回值类型设为void。

有参函数的定义：

返回值类型 函数名(形式参数表列)
形式参数类型说明
{ 函数体语句
return(返回形参名)
}

10.6.3 函数的调用

函数调用的形式为：函数名(实际参数表列)；

实参和形参的数目相等类型一致,对于无参函数当然不存在实际参数表列。

函数的调用方式有三种:

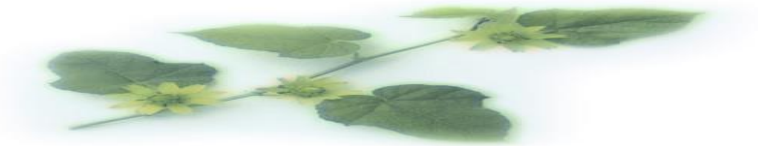
① 函数调用语句：即把被调函数名作为调用函数的一个语句；如fun1()。

② 被调函数作为表达式的运算对象，如 result=2*get(a,b)

此时get函数中的a,b应为实参，其以返回值参予式中的运算。

③ 被调函数作为另一个数的实际参数

如 m=max(a,get(a,b))； 函数get(a,b)作为函数max()的一个实际参数。



10.6.4对被调函数的说明

如果被调函数出现在主调函数之后，在主调函数前应对被调函数作以说明，形式为：

返回值类型 被调函数名(形参表列)；

如果被调函数出现在主调函数之前，可以不对被调函数说明。

下例被调函数在主调函数前，不用说明。

```
int fun1(a,b)
int a,b;
 {
int c;
c=a+b;
return(c);
}
```

```
  main( )
 {
int d,u=3,v=2;
d=2*fun(u,v);
  }
```

```
int fun1(int a,int b);
main( )
 {
 int d,u=3,v=2;
 d=2*fun1(u,v);
 }
 int fun1(int a,int b)
  { int c;
   c=a+b;
    return(c);
  }
```

本例中被调函数在主调函数后,在前面对被调函数进行说明。

10.7 C语言编程实例

常用的结构：直线程序、分支程序、循环程序、子程序

例10－5 完成19805×24503的编程

分析：两个乘数比较大，其积更大，采用unsigned long类型，设乘积存放在外部数据存贮器0号开始的单元。程序如下：

```
main()
 {   unsigned long xdata *p;   /*设定指针p指向类型为unsigned long的外部RAM区*/
     unsigned long a=19805;    /* 设置a为unsigned long类型，并赋初值 */
     unsigned long b=24503,c;  /*设置b和积为unsigned long类型，并赋初值 */
    p=0;                       /*设地址指向0号单元*/
    c=a*b;
    *p=c;                      /*积存入外部RAM 0号单元*/
    }
```

上机在变量观察窗口看到运算结果c=48528195，即为乘积的十进制数。观察XDATA区(外部RAM)的0000H～0003H单元分别为1C EC D0 7B，即存放的为乘积的十六进制数。

观察XDATA区：

地址	0000	0001	0002	0003
	1C	EC	D0	7B

*P变量(积)

可见定义为unsigned long类型，给每个变量分配四个单元。

例10－6 片内RAM 20H单元存放着一个0～05H的数，用查表法，求出该数的平方值放入内部RAM21H单元。

```
main( )
 { char x,*p
   char code tab [6] = {0,1,4,9,16,25} ;
   p=0x20;
   x=tab [*p] ;
   p++;
   *p=x;
    }
```

10.7.2 循环程序的设计

C语言的循环语句有以下几种形式

1. while(表达式){语句；}

其中表达式为循环条件，语句为循环体，当表达式值为真(值为非0)，重复执行“语句”。语句可只一条以“；”结尾；可以多条组成复合语句，复合语句必须用{}括起；也可以没有语句，通常用于等待中断，或查询。

2. do{语句；}while(表达式)

表达式为真执行循环体“语句”，直至表达式为假，退出循环执行下一个语句。

3. for(表达式1；表达式2；表达式3；){语句；}

其中语句为循环体。执行过程是：执行表达式1后进入循环体，如表达式2为假，按表达式3修改变量，再执行循环体，直到表达式2为真.

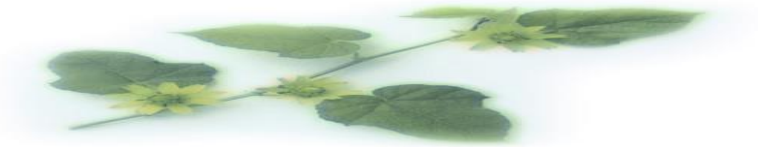
例10-8 分析下列程序的执行结果：

```
main()
 {
int sum=0,i;
do {
    sum+=i;
    i++;
    } while(i<=10);
}
```

本程序完成0+1+2+...+10的累加，执行后sum=55

例10-9 将例10-8改用for语句编程

```
main
 {
  int sum=0,i;
 for(i=0;i<10;i++)
 sun+=i;
      }
```



10.7.3 分支程序的设计

C语言的分支选择语句有以下几种形式：

1. if(表达式){语句；}

句中表达式为真执行语句，否则执行下一条语句。当花括号中的语句不只一条，花括号不能省。

2. if(表达式){语句1；}else{语句2；}

句中表达式为真执行语句1，否则执行语句2

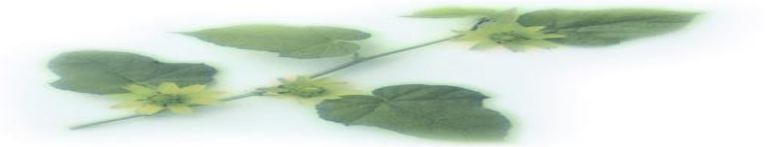
为了能无论哪种情况，执行完后都执行下一条语句。if语句可以嵌套。

3. switch(表达式) {

case 常量表达式1:{语句1；}break；...

case 常量表达式n:{语句n；}break；

default:{语句n+1;}



例10-10 片内RAM的20H单元存放一个有符号数x，函数y与x有如下关系式：

$$y=\begin{cases} x & x>0 \\ 20H & x=0 \\ x+5 & x<0 \end{cases}$$

设y存放于21H单元，程序如下

```
main()
     {
   char x,*p,*y;
   p=0x20;
   y=0x21;
   for(;;)
    {
        x=*p;
        if(x==0) *y=0x20;
        else if(x&0x80) *y=x+5;
        else *y=x;
   }
}
```

程序中为观察不同数的执行结果，采用了死循环语句for(;;)，上机调试时退出死循环可用Ctrl+C。

10.8 单片机资源的C语言编程实例

例10-12 在3.1节曾用汇编语言完成了外部RAM的000EH单元和000FH单元的内容交换，现改用C语言编程。

C语言对地址的指示方法可以采用指针变量，也可以引用absacc.h头文件作绝对地址访问，下面采用绝对地址访问方法。

```
#include<absacc.h>
 main()
 {
    char c;
     for(;;)
      {
           c=XBYTE［14］;
           XBYTE［14］=XBYTE［15］;
           XBYTE［15］=c;
    }
       }
```

10.9 汇编语言和C语言的混合编程

本节介绍不同的模块，不同的语言相结合的编程方法。通常情况下以高级语言编写主程序，用汇编语言编与硬件有关的子程序。高级语言不同的编译程序对汇编的调用方法不同，在KEIL C51中，是将不同的模块(包括不同语言的模块)分别汇编或编译，再通过连接生成一个可执行文件。

C语言程序调用汇编语言程序要注意以下几点。

1. 被调函数要在主函数中说明，在汇编程序中，要使用伪指令使CODE选项有效并声明为可再定位段类型，并且根据不同情况对函数名作转换，见表10.6。

表10.6 函数名的转换

说　明	符号名	解　释
void func(void)	FUNC	无参数传递或不含寄存器参数的函数名不作改变转入目标文件中，名字只是简单的转为大写形式
void func(char)	_FUNC	带寄存器参数的函数名加入"_" 字符前缀以示区别，它表明这类函数包含寄存器内的参数传递
void func(void)reentrant	_?FUNC	对于重入函数加上"_?"字符前缀以示区别，它表明这类函数包含寄存器内的参数传递

2. 汇编程序中对为其他模块使用的符号进行PUBLIC声明，C语言程序前对外来符号进行EXTERN声明。

3. 要注意参数的正确传递。

10.9.1 C语言程序和汇编语言程序参数的传递

在混合语言编程中，关键是入口参数和出口参数的传递，KEIL C51编译器可使用寄存器传递参数，也可以使用固定存贮器或使用堆栈，由于8XX51的堆栈深度有限，因此多用寄存器或存贮器传递。用寄存器传递最多只能传递三个参数，选择固定的寄存器，见表10.7。

表10.7 参数传递的寄存器选择

参数类型	char	int	long,float	一般指针
第1个参数	R7	R6,R7	R4-R7	R1，R2，R3
第2个参数	R5	R4,R5	R4-R7	R1，R2，R3
第3个参数	R3	R2,R3	无	R1，R2，R3

例如 funcl(int a) “a”是第一个参数，在R6,R7传递

func2(int b, int c, int *d) “b”在R6，R7中传递，“c”在R4，R5中传递，指针变量“d”在R1，R2，R3中传递。

如果传递参数寄存器不够用，可以使用存贮器传送，通过指针取得参数。

汇编语言通过寄存器或存贮器传递参数给C语言程序，汇编语言通过寄存器传递给C语言 的返回值见表10.8 。

返 回 值	寄 存 器	说 明
bit	C	进位标志
(unsigned)char	R7	
(unsigned)int	R6，R7	高位在R6，低位在R7
(unsigned)long	R4-R7	高位在R4，低位在R7
float	R4-R7	32位IEEE格式，指数和符号位R7
指针	R1，R2，R3	R3放存储器类型，高位在R2，低位在R1

下面通过实例说明混合编程的方法及参数传递过程。

C 语言程序调用汇编语言程序举例

例1. 用P1.0产生周期为4ms的方波，同时用P1.1产生周期为8ms的方波。

说明：模块一：主程序，用C语言编写，使P1.1产生周期为8ms的方波；

模块二：子程序，用C语言编写，使P1.0产生周期为4ms的 方波；

模块三：用汇编语言编写的延时1ms程序。

模块一调用模块二获得8ms方波， 模块二调模块三时向汇编程序传递字符型参数(x=2)，延时2ms，程序如下：

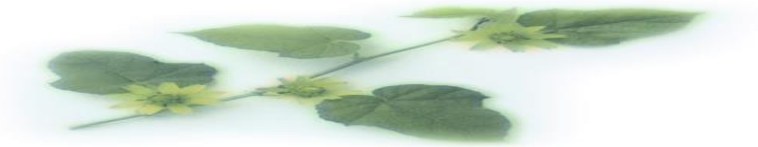
C语言程序

模块一：

```
#include<reg51.h>
  #define uchar unsigned char
  sbit P1-1=P1^1;
 extern void delay4ms(void);   /* 定义延时4ms
                          函数(模块二) */
  main( ){    uchar i;
    while(1)   { P1-1=0;
delay4ms(  );/* 调模块二延时
4ms */
P1-1=1;
delay4ms( );/* 调模块二延时
4ms*/
 }
}
```

模块二：

```
#include<reg51.h>
   #define uchar unsigned char
    sbit P1-0=P1^0;
 extern   void delaylms(uchar x);   /* 定义延时1ms函数(模块三) */
   void delay4ms(void)
 { P1-0=0;
   delaylms(2);/* 调汇编函数(模块三) */
   P1-0=1;
   delaylms(2);/* 调汇编函数(模块三)*/
 }
```



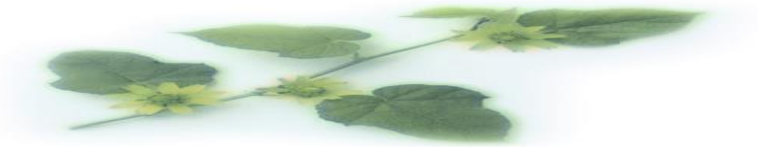
模块三：

```
DE   SEGMENT CODE          ；定义DE段为再定位程序段
PUBLIC   _delaylms         ；delaylms为其他模块调用
RSEG   DE                  ；选择DE为当前段
_delaylms：  NOP
DELA：   MOV R1，#0F8H   ；延时
LOP1：   NOP
         NOP
         DJNZ R1，LOP1
         DJNZ R7，DELA   ；R7为C程序传递过来的参数
EXIT：  RET
         END
```

例2.在汇编程序中比较两数大小，C程序为主调函数

模块一：C语言程序

```
#define uchar unsigned char
extern  uchar max(uchar a, uchar b);   /* 定义汇编函数 */
main( ){
        uchar a=5,b=35,c;
        c=max(a,b);/* 调汇编函数，a,b为传递的参数 */
        }
```



模块二：汇编语言程序

```
DE    SEGMENT CODE          ；定义为再定位程序段
      PUBLIC  _max          ；max为其 他模块调用
      RSEG  DE              ；选择DE为当前段
_max: MOV A，R7             ；取模块一的参数a
      MOV 30H，R5           ；取模块一的参数b
      CJNE A，30H，TAG1     ；比较a,b的大小
TAG1: JNC EXIT
      MOV R7，30H           ；大数存于R7中返回
EXIT: RET
      END
```

此例中，C语言程序通过R7和R5传递字符型参数a和b到汇编语言程序，汇编语言程序将返回值放在R7中，主调函数通过R7取出返回值。