微机原理和接口技术

第七讲 指令系统与汇编程序8

提 纲

13. 编程语言及汇编语言编程风格
14. 汇编程序设计中的伪指令
15. 汇编与调试过程
16. 汇编语言程序设计概述
17. 程序设计的结构化
18. 基本程序设计
19. 子程序设计概述
20. 子程序设计举例

提 纲

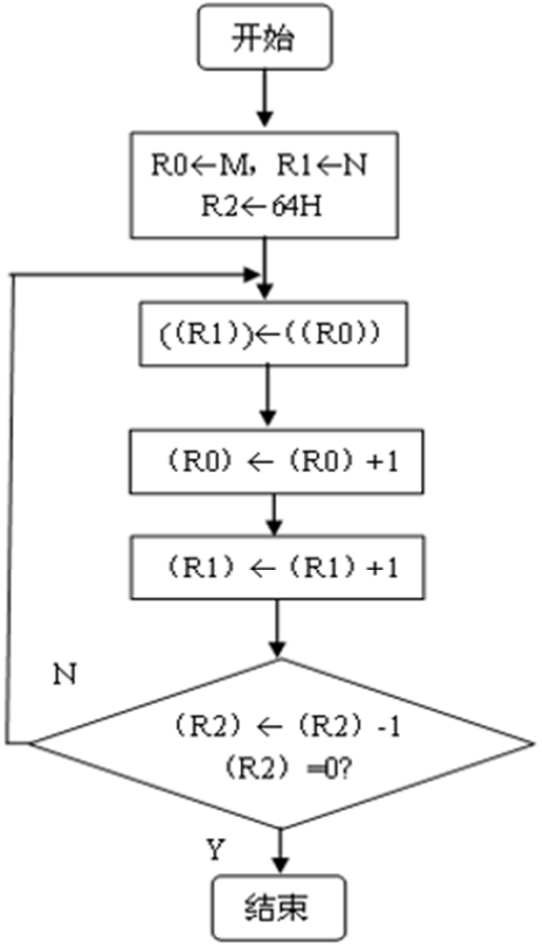
18. 基本程序设计

汇编程序基本设计

数据块传送程序。设在以M为起始地址的内部RAM中存有100个单字节数，试编一程序，把这100个数传送到以N为起始地址的外部RAM中。

解：以R0做为源数据所在内部RAM地址的指针，R1做为目标外部RAM的地址指针，R2做为循环计数控制变量。编写循环程序实现数据传送。

```
        ORG     0100H
START:  MOV     R0, #M
        MOV     R1, #N
        MOV     R2, #64H
LP:     MOV     A, @R0
        MOVX    @R1, A
        INC     R0
        INC     R1
        DJNZ    R2, LP
        SJMP    $
        END
```



汇编程序基本设计

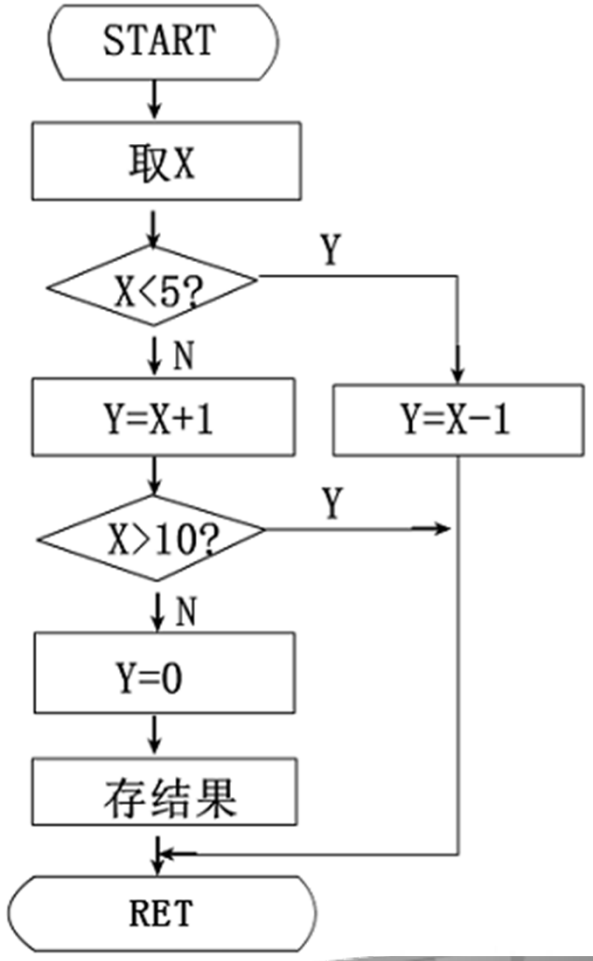
- 问题：设变量x存于内部RAM的30H单元，试编程按下式给y赋值，并存入内部RAM的31H单元

$$y=\begin{cases} x+1 & (x>10) \\ 0 & (10\geq x\geq 5) \\ x-1 & (x<5) \end{cases}$$

汇编程序基本设计

- 问题：设变量x存于内部RAM的30H单元，试编程按下式给y赋值，并存入内部RAM的31H单元

$$y=\begin{cases} x+1 & (x>10) \\ 0 & (10\geq x\geq 5) \\ x-1 & (x<5) \end{cases}$$



分析：要根据x的大小来决定y值，在判断x<5和x>10时，采用CJNE和JC以及CJNE和JNC指令进行判断，用R0暂存y的值。

汇编程序基本设计

程序：

```
x    EQU    30H
y    EQU    31H
     ORG    1000H
START: MOV     A,x          ;取X
       CJNE    A,#5,NEXT1   ;与5比较
NEXT1: JC      NEXT2        ;X<5,则转NEXT2
       MOV     R0,A
       INC     R0           ;设X>10，Y=X+1
       CJNE    A,#11,NEXT3  ;与11比较
NEXT3: JNC     NEXT4        ;X>10,则转NEXT4
       MOV     R0，#0       ;10≥x≥5,Y=0
       SJMP    NEXT4
NEXT2: MOV     R0，A
       DEC     R0           ;X<5，Y=X-1
NEXT4: MOV     y，R0    ;存结果
       RET
       END
```

汇编程序基本设计

- **例3-30：**已知内部RAM从BLOCK单元开始有一个无符号数的数据块，其长度在LEN单元，试编程求出数据块中的最大值，并存入MAX单元。

汇编程序基本设计

- **例3-30：**已知内部RAM从BLOCK单元开始有一个无符号数的数据块，其长度在LEN单元，试编程求出数据块中的最大值，并存入MAX单元。

分析：先将MAX单元清0，再把它和数据块中的数据逐一比较，若MAX中的数值大，则比较下一个，否则把数据块中的数据送入MAX；逐个比较，直到每个数都比较完毕。用R0作为数据块的地址指针。

```
分析：      ORG   2000H
            MOV   MAX, #00H        ;MAX清0
            MOV   R0, #BLOCK       ;R0指向数据块的首地址
LOOP:       MOV   A, @R0           ;取出数据块中数据送A
            CLR   C                ;C清0
            SUBB  A, MAX           ;（A）和（MAX）的数据相减，形
成Cy
            JC    NEXT             ;若（A）<（MAX），比较下一个
            MOV   MAX, @R0         ;若（A）>（MAX），则大的数送
MAX
NEXT:       INC   R0               ;指向下一数据
            DJNZ  LEN, LOOP
            END
```

提 纲

19. 子程序设计概述

子程序概述

- 在程序设计中，将那些需多次应用、具有某种相同作用的程序段从整个程序中独立出来，单独编制成一个程序，尽量使其标准化，并存放于某一存储区域，需要时通过指令进行调用。这样的程序段，称为子程序。

子程序概述

子程序是指能被主程序或其它程序调用，在实现某种功能后自动返回到调用程序去的程序。

调用子程序的程序称为主程序或调用程序。

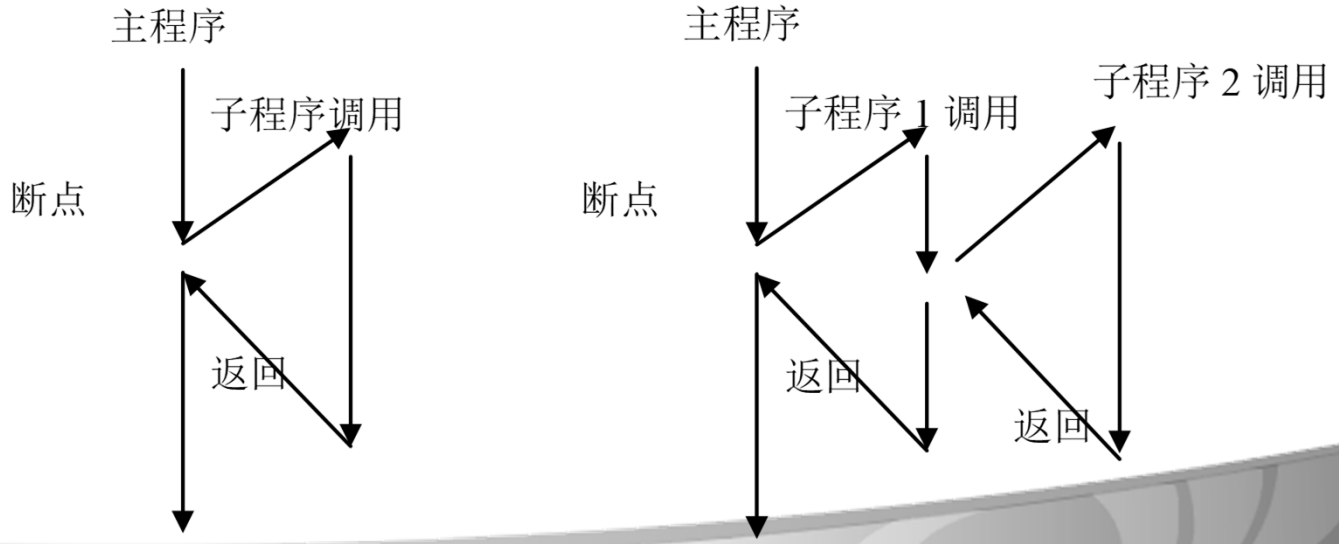
1.子程序的优点

- 不必重复书写同样的程序，提高编程效率。
- 可使程序的逻辑结构简单，便于阅读。
- 便于程序编写、调试和修改等。
- 缩短源程序和目标程序的长度，节省程序存储器空间。
- 使程序模块化、通用化，便于交流和继承。

子程序概述

2.子程序的调用和嵌套

- 通常将主程序中调用子程序指令的下一条指令的地址称为断点，子程序的第一条指令地址称为子程序首地址。
- **子程序调用：**CALL指令自动将断点地址压入堆栈保护，然后将子程序入口地址送PC，实现子程序的调用；子程序返回时，RET指令将使堆栈顶部的断点地址弹出到PC，实现子程序的返回。
- **子程序嵌套：** 在子程序的执行过程中，可能出现子程序调用其它子程序的情况，称为子程序**嵌套调用**。



3.子程序编写要点

(4)要保证子程序能够正确返回。

- 首先子程序必须以RET指令结束;
- 执行RET指令时，堆栈顶部的内容是调用时保存的断点地址;
- 子程序中，对堆栈的入栈和出栈操作次数必须相同，以保证返回时堆栈指针SP的值与调用进入时一致。

(5)子程序在功能上应具有通用性和完整性。

(6)子程序的注释要求。

- 子程序应有功能说明，标明子程序的资源占用情况，以便调用时参考;
- 子程序应注明入口参数和出口参数，以便在调用时赋值和返回时获取结果。

子程序概述

当一个程序调用子程序时，通常都向子程序传递若干个数据让它来处理；当子程序处理完后，一般也向调用它的程序传递处理结果，这种在调用程序和子程序之间的信息传递称为参数传递。

- 参数传递：进行调用程序和子程序之间的信息传递
- 入口参数：调用程序向子程序传递的参数
- 出口参数：子程序向调用程序传递的参数
- 常用的参数传递方法：寄存器法和约定存储单元法

子程序概述

主程序或调用程序和子程序之间的参数传递方法是程序员自己或和别人事先约定好的信息传递方法。常用参数传递方法有：寄存器法和约定存储单元法。

1.寄存器法：将入口参数和出口参数存放在约定的寄存器中。

特点：数据传递速度快、编程方便、节省内存单元，是最直接、简便的参数传递方式。但是MCU的寄存器个数有限，该方法适用于传递较少的参数信息。

2.约定存储单元法：把入口参数和出口参数都放在事先约定好的内存单元中

特点：不占用寄存器，参数个数可以较多，每个子程序要处理的数据和送出的结果都有独立的存储单元。但是该方法要用一定数量的存储单元，会增加编程中对变量定义的难度。

子程序概述

主程序的断点地址（即返回地址）是调用指令自动保护的；但是对于主程序中的数据（A、PSW、Rn等），为了防止执行子程序后遭到破坏，则需要编写程序加以保护。保护的内容为主程序中用到而子程序也要使用的寄存器、SFR、内存等。

数据保护与恢复的三种方法：

- **堆栈保护：**子程序首先将需要保护的数据依次压入堆栈保存，返回指令前，反序弹出堆栈以恢复现场；
- **切换工作寄存器组保护：**通过修改RS1、RS0，使主程序与子程序使用不同组别的工作寄存器；
- **内存保护：**子程序首先将需要保护的内容，保存到空闲的寄存器或内存单元暂存，返回指令前，从保存处取出以恢复现场。

子程序概述

主程序：

```
PROG1:  MOV     R2,#04H
PRO1:   LCALL   SUB1
        DJNZ    R2,PRO1
        RET
```

子程序：

```
SUB1:   MOV   R2,#20H
LOOP:   DJNZ  R2, LOOP
        RET
```

出现了死循环，主程序中的**R2**始终不会到**0**

由于**SUB1**子程序中，也要用到**R2**，因此必须要先保护后恢复。

问题：怎么保护？

子程序概述

主程序：

```
PROG1: MOV    R2,#04H
PRO1:  LCALL  SUB1
       DJNZ   R2,PRO1
       RET
```

子程序（堆栈保护）：

```
SUB1:  PUSH  02H      ；R2的内容入栈保护
       MOV   R2,#20H；子程序使用R2
LOOP:  DJNZ  R2, LOOP
       POP   02H      ；出栈恢复R2的内容
       RET
```

子程序（切换工作寄存器组保护）：

```
SUB1:  SETB  RS0      ;选择第1组的Rn
       MOV   R2,#20H
LOOP:  DJNZ  R2, LOOP
       CLR   RS0      ;恢复使用第0组的Rn
       RET
```

子程序（内存保护）：

```
B1: MOV   30H,R2   ; R2的内容暂存到30H
    MOV   R2,#20H
OP: DJNZ  R2, LOOP
    MOV   R2,30H   ; 恢复R2内容
    RET
```

提 纲

20. 子程序设计举例

子程序设计举例

- 例：微控制器主频为12MHz，试编写软件延时程序，延时时间分别为0.1s，1s，10s的程序。
- ；主频为12MHz，则机器周期为1微秒。
- 程序：　　助记符　　机器周期

```
DL1:  PUSH  30H        ; 2
      MOV   30H,#N     ; 1
D1:   NOP              ; 1
      NOP              ; 1
      DJNZ  30H,D1     ; 2
      POP   30H        ; 2
      RET              ; 2
```

- 要求程序执行时间1ms=1000μS=2+1+(1+1+2)×N+2+2
- 则有4N=993μS；　取 N=248=0F8H，代入后可得子程序真正延时为999μS，有千分之一误差。可在RET前加入NOP，即实现了1ms延时。

子程序设计举例

其它各档延时程序均可以通过调用1ms延时子程序来实现：

(1)0.1s子程序

```
DL100:  PUSH   30H          ;0.1s子程序:调用DL1共100次
        MOV    30H,#100
D100:   LCALL  DL1
        DJNZ   30H,D100
        POP    30H
        RET
```

(2)1s子程序

```
DL1S:   PUSH   30H          ;1s子程序:调用DL100共10次
        MOV    30H,#10
D1S:    LCALL  DL100
        DJNZ   30H,D1S
        POP    30H
        RET
```

子程序设计举例

其它各档延时程序均可以通过调用1ms延时子程序来实现：

(1)10s子程序

```
DL10S:    PUSH   30H          ;10s子程序，调用DL100ms共100次
          MOV    30H,#100
D10S:     LCALL  DL100
          DJNZ   30H,D10S
          POP    30H
          RET
```

由于各子程序的保护和恢复30H内容、循环初始化、调用和返回等都需要时间，所以存在一定的定时误差，可通过减少基准定时DL1ms的时间来补偿。更准确且不需要消耗CPU时间资源的延时，可利用MCU中的定时器/计数器来实现。

Thank you!