



武汉纺织大学
WUHAN TEXTILE UNIVERSITY

单片机原理及应用

课程性质：必修（考试） 学时：48 学分：3
考核方式：闭卷考试



第4章 汇编语言程序设计知识

§ 4.1 编程的步骤、方法和技巧

§ 4.2 伪指令



§ 4.1 编程的步骤、方法和技巧

§ 4.1.1 编程的步骤

§ 4.1.2 编程的方法和技巧

§ 4.1.3 汇编语言程序的基本结构



§ 4.1.1 编程的步骤

- 一、分析问题
- 二、确定算法
- 三、画程序流程图
- 四、编写程序



一、分析问题

- 对需要解决的问题进行分析，以求对问题由正确的理解。
- 解决问题的任务是什么？
- 工作过程？
- 现有的条件，已知数据，对运算的精度和速度方面的要求？
- 设计的硬件结构是否方便编程？



二、确定算法

- 算法是如何将实际问题转化成程序模块来处理。
- 在编程以前，先要对几种不同的算法进行分析、比较，找出最适宜的算法。

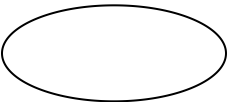
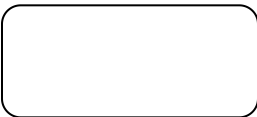


三、画程序流程图

- 程序流程图是使用各种图形、符号、有向线段等来说明程序设计过程的一种直观表示。
- 流程图步骤分得越细致，编写程序是也越方便。
- 画流程图是程序结构设计是采用的一种重要手段。
- 一个系统软件有总的流程图（主程序框图）和局部的流程图。
- 流程图常采用的图形和符号。



三、画程序流程图

椭圆框  或桶形框  : 表示程序的开始或结束。

矩形框  : 表示要进行的工作。

菱形框  : 表示要判断的事情，菱形框内的表达式表示要判断的内容。

圆圈  : 表示连接点

指向线  : 表示程序的流向



四、编写程序

- 用89C51汇编语言编写的源程序行（一条语句）
包括四个部分，也叫四个字段：

〔标号：〕 〔操作码〕 〔操作数〕； 〔注释〕

- 每个字段之间要用分隔符分隔，而每个字段内部不能使用分隔符。可以用作分隔符的符号：空格“ ”、冒号“：”、、逗号“，”、分号“；”等。

例：LOOP: MOV A, #00H; 立即数00H→A



标号

- 标号是用户定义的符号地址。
- 一条指令的标号是该条指令的符号名字，标号的值是汇编这条指令时指令的地址。
- 标号由以英文字母开始的1~8个字母或数字组成，以冒号“:”结尾。
- 标号可以由赋值伪指令赋值，如果没有赋值，汇编程序把存放该指令目标码第一字节的存储单元的地址赋给该标号，所以，标号又叫指令标号。



操作码

- 操作码是必不可少的。
- 它用一组字母符号表示指令的操作码。在89C51中，由89C51的指令助记符组成。



操作数

- 汇编语言指令可能要求或不要求操作数，所以这一字段可能有也可能没有。
- 若有两个操作数，操作数之间用逗号“，”分开。
- 操作数包括的内容有：
 - (1) 工作寄存器：由PSW.3和PSW.4规定的当前工作寄存器区中的R0~R7。
 - (2) 特殊功能寄存器：21个SFR的名字。
 - (3) 标号名：赋值标号—由汇编指令EQU等赋值的标号；指令标号—指令标号指示的指令的第一字节地址是该标号的值。
 - (4) 常数：可用二进制（B）、十进制、十六进制（H），若常数以字符开头，前面加0。
 - (5) \$：用来表示程序计数器的当前值。
 - (6) 表达式：汇编时，计算出表达式的值填入目标码。



注释

- 注释部分不是汇编语言的功能部分，只是用语增加程序的可读性。
- 良好的注释是汇编语言程序编写中的重要组成部分。

§ 4. 1. 2 编程的方法 and 技巧



一、模块化的程序设计方法

二、编程技巧



一、模块化的程序设计方法

- 1、程序功能模块化的优点
- 2、划分模块的原则



1、程序功能模块化的优点

- 单个模块结构的程序功能单一，易于编写、调试和修改。
- 便于分工，从而可使多个程序员同时进行程序的编写和调试工作，加快软件研制进度。
- 程序可读性好，便于功能扩充和版本升级。
- 对程序的修改可局部进行，其它部分可以保持不变。
- 对于使用频繁的子程序可以建立子程序库，便于多个模块调用。



2、划分模块的原则

- 每个模块应具有独立的功能，能产生一个明确的结果，即单模块的功能高内聚性。
- 模块之间的控制耦合应尽量简单，数据耦合应尽量少，即模块间的低耦合性。控制耦合是指模块进入和退出的条件及方式，数据耦合是指模块间的信息交换方式、交换量的多少及交换频繁程度。
- 模块长度适中。20条~100条的范围较合适。



二、编程技巧

- 1、尽量采用循环结构和子程序。
- 2、尽量少用无条件转移指令。
- 3、对于通用的子程序，考虑到其通用性，除了用于存放子程序入口参数的寄存器外，子程序中用到的其他寄存器的内容应压入堆栈（返回前再弹出），即保护现场。
- 4、在中断处理程序中，除了要保护处理程序中用到的寄存器外，还要保护标志寄存器。
- 5、用累加器传递入口参数或返回参数比较方便，在子程序中，一般不必把累加器内容压入堆栈。



§ 4.1.3 汇编语言程序的基本结构

一、顺序程序

二、分支程序

三、循环程序



一、顺序程序

- 顺序程序是最简单的程序结构，即顺序结构。
- 程序按顺序一条一条地执行指令。



例4-1：双字节加法子程序段。设被加数存放于片内RAM的addr1(低字节)和addr2(高字节)，加数存放于片内RAM的addr3(低字节)和addr4(高字节)，结果放在addr1(低字节)和addr2(高字节)中。

```
解：      ORG      1000H  
START: PUSH  ACC;  
      MOV      R0, #addr1;  
      MOV      R1, #addr3;  
      MOV      A, @R0;  
      ADD      A, @R1;  
      MOV      @R0, A;  
      INC      R0;  
      INC      R1;  
      MOV      A, @R0;  
      ADDC     A, @R1;  
      MOV      @R0, A;  
      POP      ACC;
```



例4-2：拆字。将片内RAM 20H单元的内容拆成两段，每段四位。并将它们分别存入21H与22H单元中。程序如下：

```
解：          ORG    2000H  
START:      MOV    R0, #21H ; 21H→R0  
            MOV    A, 20H  ; (20H) →A  
            ANL    A, #0FH ; A∧#0FH→A  
            MOV    @R0, A  ; (A) → (R0)  
            INC    R0      ; R0+1→R0  
            MOV    A, 20H  ; (20H) →A  
            SWAP   A,      ; A0~3 ←→A4~7  
            ANL    A, #0FH ; A∧#0FH→A  
            MOV    @R0, A  ; (A) → (R0)  
            SJMP   $
```



例4-3：16位数求补。设16位二进制数在R1和R0中，求补结果存于R3和R2中。

解：

```
                ORG      1000H
START:  MOV      A, R0
        CPL      A
        ADD      A, #01H
        MOV      R2, A
        MOV      A, R1
        CPL      A
        ADDC     A, #00H
        MOV      R3, A
        SJMP     $
```



二、分支程序

- 程序分支是通过条件转移指令实现的，即根据条件对程序的执行进行判断、满足条件则进行程序转移，不满足条件就顺序执行程序。
- 分支程序又分为单分支和多分支结构。
- 多分支程序是首先把分支程序按序号排列，然后按序号值进行转移。
- 在89C51指令系统中，通过条件判断实现单分支程序转移的指令有：JZ、JNZ、CJNE、DJNZ等。此外还有以位状态作为条件进行程序分支的指令，如JC、JNC、JB、JNB、JBC等。使用这些指令可以完成0、1、正、负，以及相等、不相等作为各种条件判断依据的程序转移。
- 结构如图4-1所示。



二、分支程序

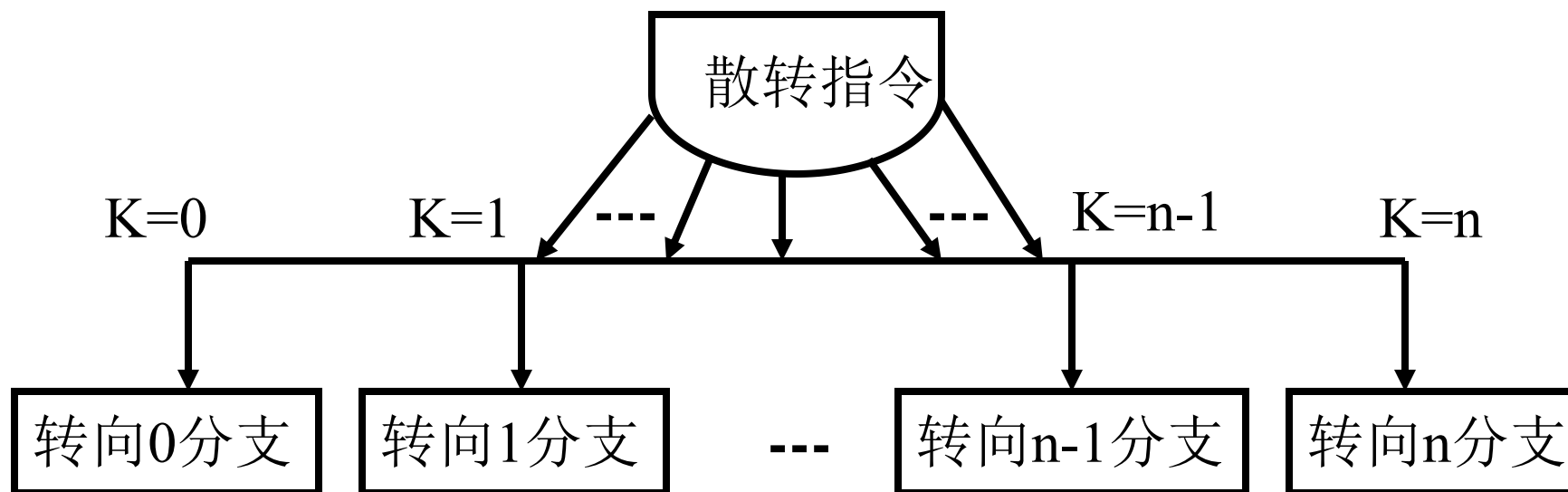


图4-1 分支程序结构



例4-4： 128种分支转移程序。功能：根据入口条件转移到128个目的地址。

入口：（R3）=转移目的地址的序号00H~7FH。出口：转移到相应子程序入口。

解：

JMP_128: MOV A, R3

RL A

MOV DPTR, #JMPTAB

JMP @A+DPTR

JMPTAB: AJMP ROUT00

AJMP ROUT01

⋮

AJMP ROUT7F

} 128个子程序首址

•说明：此程序要求128个转移目的地址（ROUT00 ~ROUT7FH）必须驻留在与绝对转移指令AJMP相同的一个2KB存储区内。

•RL指令对变址部分乘以2，因为每条AJMP指令占两个字节。

例4-5：存放于addr1和addr2中的两个无符号二进制数，求其中的大数并存放于addr3中。程序如下：



```
解：  ORG 1000H
      addr1 DATA 31H ; 定义 addr1
      addr2 DATA 32H ; 定义 addr2
      addr3 DATA 30H ; 定义 addr3
START: MOV     A, addr1
        CJNE   A, addr2, LOOP1; 两数比较，不相等则转LOOP1
        SJMP   LOOP2
LOOP1:  JC     LOOP2             ; 当CY=1, 转LOOP2
        MOV    addr3, A         ; CY=0, (A)>(addr2)
        SJMP   LOOP3           ; 转结束
LOOP2:  MOV    addr3, addr2     ; CY=1, (addr2)>(A)
LOOP3:  END
```



例4-6：片内RAM中ONE和TWO两个单元中存有两个无符号数，将两个数中的小者存入RES单元。程序如下：

```
解：  ORG 1000H
      ONE DATA 22H ; 定义ONE
      TWO DATA 23H ; 定义TWO
      RES DATA 30H ; 定义RES
          MOV     A, ONE
START: CJNE     A, TWO, BIG
          SJMP    STORE
BIG:     JC     STORE
          MOV     A, TWO
STORE:  MOV     RES, A
          SJMP    $
```



例4-7：设变量x存放于VAR单元中，函数值y存放于FUNC中，按下式赋值

解：程序如下：

ORG 0800H

VAR DATA 30H ;定义VAR

FUNC DATA 31H ;定义FUNC

START: MOV A, VAR ;取x

JZ COMP ;为0转COMP

JNB ACC.7, POSI ;x>0转POSI

MOV A, #0FFH ;x<0,-1→A

SJMP COMP

POSI: MOV A, #01H

COMP: MOV FUNC, A

END

$$y = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$



三、循环程序

- 在程序运行时，有时需要连续重复执行某段程序，可以使用循环程序。其结构包括四部分：
 - 1、置循环初值
 - 2、循环体（循环工作部分）
 - 3、修改控制变量
 - 4、循环控制部分
- 其组织方式如图4-5所示。



1、置循环初值

- 对于循环程序中所使用的工作单元，在循环开始时应置初值。
- 例如，工作寄存器设置计数初值，累加器A清0，以及设置地址指针、长度等。



2、循环体（循环工作部分）

- 重复执行的程序段部分，分为循环工作部分和循环控制部分。
- 循环控制部分每循环一次，检查结束条件，当满足条件时，就停止循环，往下继续执行其他程序。



3、修改控制变量

- 在循环程序中，必须给出循环结束条件。
- 常见的是计数循环，当循环了一定的次数后，就停止循环。
- 在单片机中，一般用一个工作寄存器Rn作为计数器，对该计数器赋初值作为循环次数。每循环一次，计数器的值减1，即修改循环控制变量，当计数器的置件为0时，就停止循环。

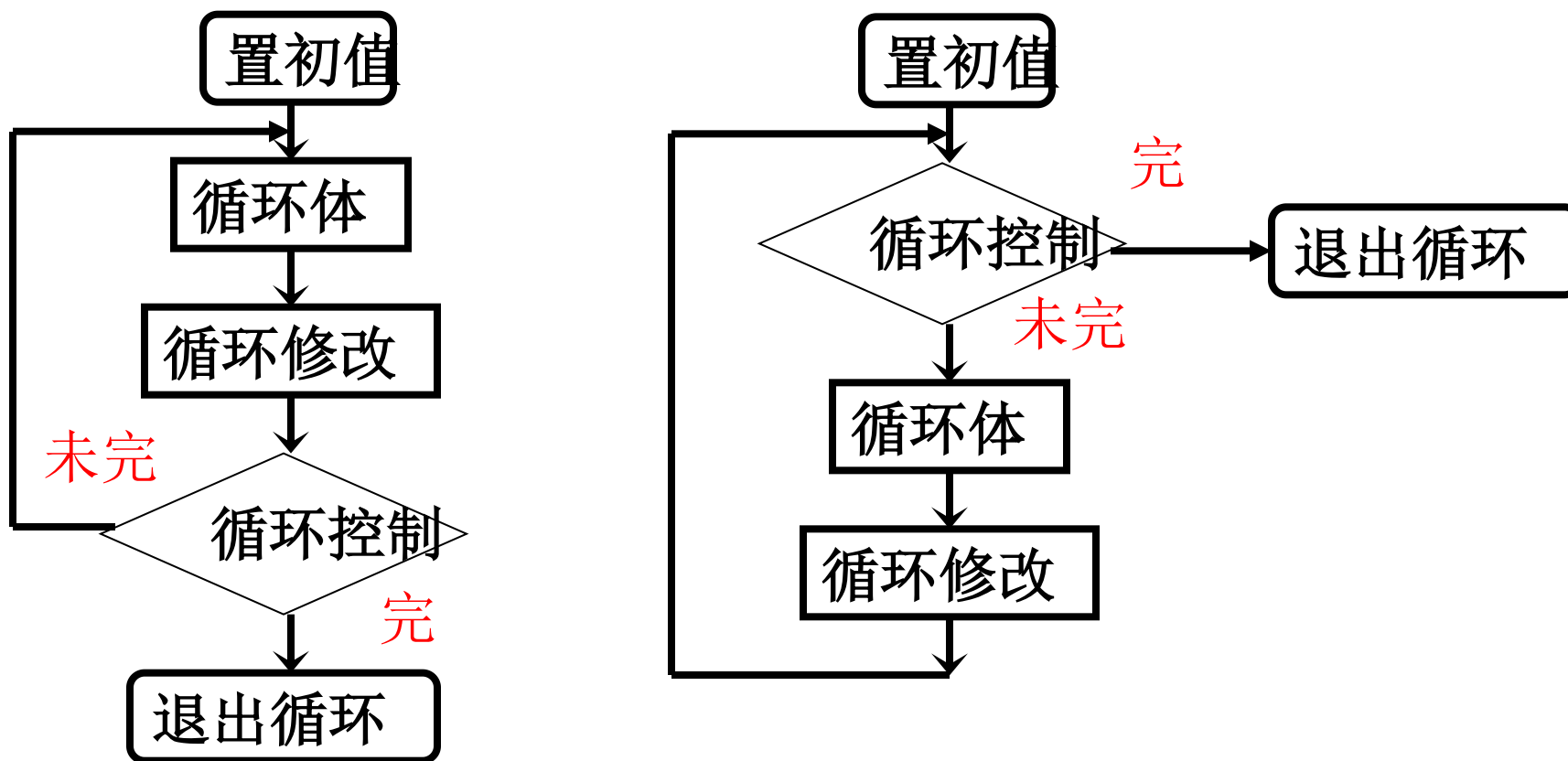


4、循环控制部分

- 根据循环结束条件，判断是否结束循环。
- **89C51**可采用**DJNZ**指令来自动修改控制变量并能结束循环。



三、循环程序



(a)

(b)

图4-5 循环组织方式流程图



1) 采用循环程序进行软件延时子程序

执行**MOV Rn,#data**要一个机器周期

执行**DJNZ Rn,\$** 要两个机器周期

执行**RET** 要两个机器周期

因此可以精确算出程序执行时间。

例：以机器周期为单位，计算以下程序的执行时间。

DELAY: MOV R2,#data ; 预置循环控制常数

DELAY1: DJNZ R2, DELAY1 ; 当(R2) 不等于0时，转向本身

RET



查表注意到执行DJNZ指令要用两个机器周期

执行MOV Rn,#data要一个机器周期

执行MOV dir,#data要两个机器周期

因此可以精确算出程序执行时间。

例：以机器周期为单位，计算以下程序的执行时间。

	MOV R7,#64H	;1个机器周期
LOOP:	MOV R6,#0FAH	;1
	DJNZ R6,\$;(2), $250 \times 2 = 500$
	DJNZ R7,LOOP	;(2), $503 \times 100 = 50300$
	RET	;2
		共50303个机器周期



TIME:	MOV R1,#0FAH	;1个机器周期
L1:	MOV R0,#0FFH	;1
W1:	DJNZ R0,W1	;(2), $255*2=510$
	DJNZ R1,L1	;(2), $513*250=128250$
	NOP	;1
	NOP	;1
	RET	;2
		共128255个机器周期



问题：将以上程序的第二句 **LOOP: MOV R6,#0FAH**

改为 **LOOP: MOV R6,#0**

则 **DJNZ R6,\$** 语句会执行多少次？

要实现延时功能，有时还需要用到空操作指令：

关 键 字	功能简述	字节数	机器周期
NOP	空操作	1	1

空操作指令不进行任何操作，但它在程序存储器中占了一个字节的位置，执行时也需要占用一个机器周期的时间。



例：编写延时4ms的子程序。设晶振为12MHz。

解：晶振为12MHz，则1个机器周期为1微妙（ $1\mu\text{s}$ ），一条DJNZ指令为2个机器周期，则执行该指令2000次为4ms， $2000=20*100$

ORG 1000H

Delay: MOV R6,#20

DLY1: MOV R7,#100

DLY2: DJNZ R7,DLAY2

DJNZ R6,DLY1

RET



例4-9：多字节无符号加法程序。设被加数低字节地址存于R0中，加数低字节地址存于R1中，字节数存于R3中，相加结果存于原被加数单元。

```
START: MOV  A,R0      ;保存首地址
        MOV  R5,A
        MOV  A,R3      ;保存字节数
        MOV  R7,A
        CLR  C
ADDA:   MOV  A,@R0
        ADDC A,@R1      ;作加法
        MOV  @R0,A      ; 部分和存入对应的被加数单元
        INC  R0
        INC  R1
        DJNZ R7,ADDA    ; 当(R7) 不等于0时，则继续作加法
        JNC  ADDB       ;处理最高字节进位
        INC  R3
        MOV  @R0,#01H
ADDB:   MOV  A,R5      ;增加内容
        MOV  R0,A
        END
```



例4-10：搜索最大值。从片内 RAM 的 BLOCK 单元开始有一个无符号数据块,其长度存于 LEN 单元中,试求出其中最大的。

```
START:  LEN DATA 20H  
        MAX DATA 21H  
        BLOCK DATA 22H  
        CLR A  
        MOV R2, LEN           ;数据块长度送 R2  
        MOV R1, # BLOCK      ;置地址指针  
LOOP:   CLR C  
        SUBB A, @ R1         ;用减法做比较  
        JNC NEXT            ;无借位 A 大  
        MOV A, @ R1         ;否则大者送 A  
        SJMP NEXT1  
NEXT:   ADD A, @ R1         ;A 大恢复 A  
NEXT1:  INC R1              ;修改地址指针  
        DJNZ R2, LOOP       ;未完继续  
        MOV MAX, A         ;若完则存大数
```



§ 4.2 伪指令

- 伪指令不是真正的指令，无对应的机器码，在汇编时不产生目标程序，只是用来对汇编过程进行某种控制。
- **89C51有8个伪指令：**

ORG

END

EQU

DATA

DB

DW

DS

BIT



ORG 汇编起始命令

- 格式：ORG 16位地址
- 功能：规定该伪指令后面程序的汇编地址，即汇编后生成目标程序存放的起始地址。例如：

ORG 2000H

START: MOV A, #64H

⋮

- 规定了START的地址是2000H，又规定了汇编后的第一条指令码从2000H开始存放。



END 汇编结束指令

- 格式: **END**
- 功能: 通知汇编程序结束汇编。在**END**之后所有的汇编指令均不予以处理。



EQU 赋值命令

- 格式：字符名称 EQU 项（数或汇编符号）
- 功能：把“项”赋给“字符名称”。
- 注意：字符名称不等于标号（其后没有冒号）；其中的项，可以是数，也可以是汇编符号。EQU赋值过的符号名可以用作数据、代码地址、位地址或一个立即数。可以是8位的，也可以是16位的。



EQU 赋值命令

例1:

AA EQU R1

MOV A, AA ; AA代表工作寄存器R1

例2:

A10 EQU 10

DELY EQU 07EBH

MOV A, A10 ; A10作为片内的一个直接地址

LCALL DELY ; DELY作为一个16位子程序的入口地址



DATA 数据地址赋值命令

- 格式：字符名称 DATA 表达式
- 功能：与EQU类似，但有以下差别：
 - 1、EQU定义的字符名必须先定义后使用，而DATA定义的字符名可以后定义先使用。
 - 2、用EQU伪指令可以把一个汇编符号赋给一个名字，而DATA只能把数据赋给字符名。
 - 3、DATA语句可以把一个表达式的值赋给字符名称，其中的表达式应是可求值的。DATA伪指令在程序中用来定义数据地址



DB 定义字节命令

- 格式： **DB** （项或项表）
- 功能： 通知汇编程序从当前**ROM**地址开始， 保留一个字或字节串的存储单元， 并存入**DB**后的数据。
- 注意： 项或项表可以是一个字节， 用逗号隔开的字节串或括在单引号中的**ASCII**字符串。



DB 定义字节命令

例如: **ORG 2000H**
 DB 0A3H

LIST: DB 26H, 03H

STR: DB ‘ABC’
 ⋮

经汇编后 (2000H) =A3H,
 (2001H) =26H,
 (2002H) =03H,
 (2003H) =41H,
 (2004H) =42H,
 (2005H) =43H, (41H, 42H, 43H分别为A, B, C的ASCII码)



DW 定义字命令

- 格式: **DW** 16位数据项或项表
- 功能: 把**DW**后的16位数据项或项表从当前地址连续存放。每项数值为16位二进制数, 高8位先放, 低8位后存放。**DW**用于定义一个地址表。
- 例如:

```
ORG 1500H  
TABLE: DW 7234H, 8AH, 10H  
      ⋮
```

经汇编后 (1500H) =72H,
(1501H) =34H,
(1502H) =00H,
(1503H) =8AH,
(1504H) =00H,
(1505H) =10H,



DS 定义存储空间命令

- 格式: **DS 表达式**
- 功能: 在汇编时, 从指定地址开始保留**DS**之后表达式的值所规定的存储单元以备后用。
- 例如:

ORG 1000H

DS 08H

DB 30H, 8AH

汇编后, 从**1000H**保留8个单元, 然后从**1008H**按**DB**命令给内存赋值, 即 **(1008H) = 30H**

(1009H) = 8AH



BIT 位地址符号命令

- 格式：字符名 **BIT** 位地址
- 功能：把**BIT**后的位地址值赋给字符名。其中字符名不是标号，其后没有冒号，但字符名是必须的。
- 例如：

A1 BIT P1.0

A2 BIT 02H

汇编后，P1口第0位的位地址90H就赋给了A1，而A2的值则为02H。



§ 4.3 思考题与习题

下列程序段经汇编后，从1000H开始的各有关存储单元的内容将是什么？

```
TAB1    ORG    1000H
        EQU    1234H
TAB2    EQU    3000H
        DB     "START"
        DW     TAB1,TAB2,70H
```



第四章结束