第6章 伪指令与源程序格式

- **重点**：伪指令语法和格式；
- **内容**：6.1 伪指令
  6.2 语句格式
- 汇编语言程序的语句：指令、伪指令、宏指令。
- 伪指令（伪操作）：主要用来定义数据变量和程序结构。

6.1 伪指令

- 指令是在程序运行期间由计算机的CPU来执行的。
- 伪指令是在汇编程序对源程序进行汇编期间由汇编程序处理的操作。

6.1.1 段定义伪指令

6.1.2 程序开始和结束伪指令

6.1.3 数据定义与存储器单元分配伪指令

6.1.4 类型属性操作符

6.1.5 This和Label

6.1.6 EQU和=

6.1.7 地址计数器$与定位伪指令

6.1.8 基数控制伪指令

6.1.9 过程定义伪指令

6.1.1 段定义伪指令

- 例6.1 段定义

```
data        segment                    ；定义数据段data
             buff   db ‘hello,world!$’
data        ends
code        segment                    ；定义代码段code
assume cs:code,ds:data                 ；指定段寄存器和段
                                          的关系
start：  mov ax,data                   ；对ds赋data段基地址
         mov ds,ax
```

```
    lea bx,buff
    mov ax, 'AB'
    mov [bx],ax
    mov dx,offset buff
    mov ah,9
    int 21h
    mov ah,4ch
    int 21h
code    ends
    end start
```

```
C:\>debug sam6_1.exe
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 8D1E0000      LEA     BX,[0000]
076B:0009 B84241        MOV     AX,4142
076B:000C 8907          MOV     [BX],AX
076B:000E BA0000        MOV     DX,0000
076B:0011 B409          MOV     AH,09
076B:0013 CD21          INT     21
076B:0015 B44C          MOV     AH,4C
076B:0017 CD21          INT     21
076B:0019 0000          ADD     [BX+SI],AL
076B:001B 0000          ADD     [BX+SI],AL
076B:001D 0000          ADD     [BX+SI],AL
076B:001F 0000          ADD     [BX+SI],AL
-g
BAllo,world!
Program terminated normally
```

1. 段定义伪指令

- 段定义伪指令格式：

  segment_name　SEGMENT

  ……

  segment_name　ENDS

- **ASSUME伪指令格式：**

  ASSUME register_name : segment_name ，register_name : segment_name

- assume伪指令只是指定把某个段分配给哪个段寄存器，并不能把段地址装入段寄存器中，所以在代码段中，还必须把段地址装入相应的段寄存器中，通常用两条MOV指令完成这一操作，但是代码段不需要这样做，它在程序初始化的时候完成。

2. 简化的段定义伪指令

- . MODEL定义存储模型。
- .DATA定义数据段，默认段名为_DATA。
- @DATA表示段名_DATA，在指令中表示段地址。

- . MODEL定义存储模型
  - tiny（微型模式）：cs、ss、ds的初值将被设置为1个，意味着只能使用一个段。程序大小不能超过64kb
  - small（小型模式）：只能至多有一个代码段和一个数据段。因此程序总大小不能超过128kb。
  - compact（紧凑模式）：代码段仅有一个，数据段可以有多个。数据的指针默认为远转移。

- . MODEL定义存储模型
  - medium（中型模式）：代码段可以有多个，数据段仅有一个。调用类型默认为远转移。
  - large（大型模式）：代码段和数据段都可以是多个。但是一开始定义好的静态数据段仍然只能只能不超过64kb，保证在一个段内。
  - huge（巨型模式）：代码段和数据段都可以是多个。无限制。
  - flat（平展模式）：创建一个32位的程序。DOSBOX不能运行该程序。

- 例6.2

```
·model   small          ；定义存储模型为small
·data                   ；定义数据段data
      string   db ‘hello,world!$’
·code                           ；定义代码段code
start：mov ax,@data     ；对ds赋data段基地址
        mov ds,ax
        mov dx,offset string
        mov ah,9
        int 21h
        mov ah,4ch
        int 21h
end     start
```

6.1.2 程序开始和结束伪指令

- 表示源程序结束的伪指令格式：

  END [标号]

- 标号指示程序开始执行的起始地址。
- 若有多个程序模块，则只需指明主程序开始时的标号，其它子程序模块则只用**END**而不能指定标号。

6.1.3 数据定义与存储器单元分配伪指令

- 指令语句的一般格式：

  [标号:] 操作码 操作数 [; 注释]

- 和指令语句格式类似，这一类伪指令的格式：

  [变量名] 伪指令 N个操作数 [;注释]

- 标号就是指令的符号地址。
- 标号的三种属性：

① 段值（指明存在于哪个段中）
标号对应存储段的段地址

② 偏移值（指明段内的位置）
标号在存储段内的偏移地址

③ 类型（指明标号的类型属性）
标号的类型可以是NEAR和FAR两种

- 变量实质上就是某一个或几个存储单元。
- 变量的三种属性：

① 段值（指明存在于哪个段中）
变量（名字）对应存储单元的段地址

② 偏移值（指明段内的位置）
变量（名字）对应存储单元的偏移地址

③ 类型（指明每个变量占用的字节数）
变量名的类型可以是**DB(1字节)、DW(2字节)、DD(4字节)、DF(6字节)、DQ(8字节)、DT(10字节)等**

标号与变量区别

- 变量
  - 定义一般在非代码段，是数据在内存中存放的符号地址，是在程序运行期间可随时被修改数值的数据对象，是内存的一个数据区的名字。
  - 由标识符构成，其后不带冒号。
- 标号
  - 定义一般出现在代码段中，表示指令在内存中存放的符号地址。它对应的值在汇编时自动计算。
  - 是由标识符及后一个冒号构成。

定义数据类型的伪指令：

- DB：用来定义字节，其后的每个操作数都占用1个字节。
- DW：用来定义字，其后的每个操作数都占用1个字。
- DD：用来定义双字，其后的每个操作数都占用2个字。
- DF：用来定义六个字节的字，其后的每个操作数都占用48位。
- DQ：用来定义4个字，其后的每个操作数都占用4个字。
- DT：用来定义10个字节，其后的每个操作数都占用10个字节。

- 程序中默认的数据为十进制数。
- 当数据第一位不是数字时，应在前面加0。
- 负数均为补码形式存放。
- 字符串用‘’括起来。
- ‘？’表示只分配存储单元，不存入数值。

- DUP　　复制伪指令
- 格式：count DUP (operand, …, operand)
- 操作：将括号中的操作数重复count次，count可以是一个表达式，其值应该是一个正数。
- DUP操作可嵌套。

- 例6.3 操作数为常数、数据表达式。

```
D_BYTE      DB  10, 10H
D_WORD      DW  14,100H,-5，0ABCDH
D_DWORD     DD  4×8
```

- 例6.4 操作数为字符串。问号‘?’仅预留空间。

```
MESSAGE   DB ‘HELLO?’,?     ；问号?通常
                                被系统置0
          DB ‘AB’，？
          DW ‘AB’          ；注意这里‘AB’作为
                             串常量按字类型存放

MESSAGE
```

- 例6.5 用操作符复制操作数。

  ARRAY DB 2 DUP(1,3,2 DUP(4,5))

- 例6.6 根据需要自己定义的各类数据,含义由自己决定。

  X1 DB 14，3 ；十进制小数3.14

  Y2 DW 1234H,5678H ；32位数据十六进制数
  56781234H

  Y3 DW 22，9 ；32位数据十六进制数00090016H

6.1.4 类型属性操作符

- WORD PTR ； 字类型
- BYTE PTR ； 字节类型
- 类型属性操作符仅是指定变量的“访问类型”，并不改变变量本身的类型。

- 例6.7 在指令中用类型属性操作符指定对内存变量的访问类型，以匹配两个操作数。

```
OPER1   DB  3, 4
OPER2   DW  5678H, 9
  ⋮
MOV   AX，OPER1      ；操作数类型不匹配
MOV   BL, OPER2      ；操作数类型不匹配
MOV   [DI], 0        ；操作数类型不明确
```

- 这三条指令可改为：

```
MOV   AX，WORD PTR OPER1     ；从OPER1处取一个字使
                               AX=0403H
MOV   BL, BYTE PTR OPER2     ；从OPER2处取一个字节使BL=78H
MOV   BYTE PTR[DI], 0        ；常数0送到内存字节单元
```

6.1.5 THIS操作符和LABEL伪操作

- 一个变量可以定义成不同的访问类型，THIS操作符或LABEL伪操作都可以实现。
- 格式：THIS type
- 格式：name LABEL type
- 操作：指定一个类型为type的操作数，使该操作数的地址与下一个存储单元地址相同。

- 例6.8 把变量定义成不同访问类型，以便指令中可灵活选用。指令执行结果如图6-6所示。

```
BUF=THIS  WORD
DAT   DB  8,9
OPR_B    LABEL  BYTE
OPR_W    DW  4 DUP(2)
 ┇
MOV   AX,  1234H
MOV   OPR_B, AL
MOV   OPR_W+2, AX
MOV   DAT+1, AL
MOV   BUF, AX
```

表达式BUF=THIS WORD使BUF和DAT指向同一个内存单元。
LABE伪操作使得OPR_B和OPR_W指向同一个内存单元。

6.1.6 表达式赋值伪指令“EQU”和 “=”

- 可以用赋值伪操作给表达式赋予一个常量或名字。格式如下：

  Expression_name EQU Expression

  Expression_name = Expression

- 表达式中的变量或标号，必须先定义后引用。
- EQU伪操作中的表达式名是不允许重复定义的，而“＝”伪操作则允许重复定义。

6.1.7 汇编地址计数器$与定位伪指令

1. 地址计数器 $

- 地址计数器是一个16位的变量，用$表示
- 开始汇编或在每一段开始时，将地址计数器初始化为零。
- 当在指令中用到$时，它只代表此指令的首地址，而与$本身无关。
- 当$用在伪操作的参数字段时，它所表示的是地址计数器的当前值。

- 例：jmp $+6；表示转向地址是Jmp指令的首地址加上6。

- 例：array dw 1,2,$+4,3,4,$+4，array分配的偏移地址 为0074。

  [0074] 01,00 , 02,00, 7C,00, 03,00, 04,00, 82,00

  0078+4 007E+4

- 注：ARRAY数组中的两个$+4得到的结果是不同的，这是由于$的值在不断变化。

- 例6.9 考察$的作用,假定$初值=0。

```
ARRAY   DW 3,$+7,7
COU=$
NEW     DW COU
```

2. ORG 伪操作

- ORG伪操作用来设置当前地址计数器的值。
- 格式：ORG constant expression
- 操作：如常数表达式的值为n，则该操作指示下一个字节的存放地址为n。

- 例6.10 考察ORG伪操作,数据在内存中的存放如图6-8所示。

```
ORG  0
DB   3
ORG  4
BUFF   DB  6
ORG    $+6
VAL    DB  9
```

3. EVEN 伪操作

- EVEN伪操作使下一个变量或指令开始于偶数地址。

4. ALIGN 伪操作

- ALIGN伪操作使下一个变量的地址从4的倍数开始。

6.1.8 基数控制伪指令

- 汇编程序默认的数为十进制数，程序中使用其它基数表示的常数要标记。
- .RADIX伪操作可以把默认的基数改变为2~16范围内的任何基数。
- 格式：.RADIX　expression　；　表达式表示基数值（用十进制数表示）。
- 注意：　在用.RADIX把基数定为十六进制后，十进制数后面都应该跟字母D。

例如：

```
MOV     BX, 0FFH        ; 16进制数标记为H
MOV     BL, 10000101B   ; 二进制数标记为B
MOV     BX, 178         ; 10进制为默认的基数，可无标记
.RADIX  16              ; 以下程序默认16进制数
MOV     BX, 0FF         ; 16进制为默认的基数，可无标记

MOV     BX, 178D        ; 10进制数应加标记D
```

应当注意，在用 **.RADIX 16**把基数定为十六进制后，十进制数后面都应跟字母**D**。在这种情况下，如果某个十六进制数的末字符为**D**，则应在其后跟字母**H**，以免与十进制数发生混淆。

6.1.9 过程定义伪指令

- 过程相当于高级语言程序中的子程序，是能完成特定功能的独立的代码模块。
- 过程是进行模块化程序设计的基础。
- 80x86中调用过程指令是CALL，从过程返回的指令是RET。

- 过程定义包含两条伪指令：PROC和ENDP。PROC表示过程的开始，ENDP表示过程的结束。
- 过程定义语句的格式：

```
过程名  PROC  [属性]          ；过程开始
                              ；过程体
过程名  ENDP                  ；过程结束
```

- 功能：定义一个过程(子程序)。

- 过程名是标识符，起到标号的作用，是子程序入口的符号地址。

- 过程的属性可以是FAR或NEAR类型。NEAR为近，是段内调用。FAR类型为远，是跨段调用，缺省时为NEAR。

- 例 6.11

```
data    segment                ；定义数据段data
        string   db 'hello,world!$'
data    ends
code    segment                ；定义代码段code
assume  cs:code,ds:data
main    proc   far             ；定义过程main
        mov  ax,data
        mov  ds,ax
        mov  dx,offset string
        mov  ah,9
        int  21h
        mov  ah,4ch
        int  21h
main    endp
code    ends
        end   main             ；汇编结束, 程序起始点main
```

6.2 语句格式

- 语句格式：

[name] operation operand [;comment]

名字 操作 操作数 注释

可选，不是必须

- 各项之间必须用空格隔开。

- 名字项是一个符号，可以是指令的标号，也可以是变量名。
- 操作项是一个操作码的助记符，可以是指令、伪指令或宏指令名。
- 操作数项由一个或多个表达式组成。
- 注释项用来说明程序或语句的功能。

6.2.1 名字项和操作项

1. 名字项

- 由字母、数字和专用字符（?,.,@,_,$）组成。
- 以字母和特殊字符开头（@、$、.、_、？）。
- 名字中如果用到·，则必须是第一个字符。
- 不能与保留字相同。

- 名字项可以是标号或变量，用来表示符号地址，有三种属性：

  段属性：定义该符号地址的段起始地址，此值必须在一个段寄存器中。

  偏移属性：从段起始地址到定义该地址符号的位置之间的字节数。

  类型属性：对于标号指明是段内还是段外引用；对于变量指明该变量所占字节数。
- 标号或变量不能重复定义。

2. 操作项

- 指令：汇编程序将其翻译为机器语言指令。
- 伪指令：汇编程序根据其要求进行处理。
- 宏指令：根据宏定义展开。

6.2.2 表达式和操作符

- 算术操作符
- 逻辑与移位操作符
- 关系操作符
- 数值回送操作符

1. 算术操作符

- 算术运算符主要有+、—、*、/、MOD。
- MOD也称为取模，它得到除法之后的余数。
- 减法运算可用于段内两个操作数地址（以变量名表示）的运算，其结果是一个常数，表示这两个变量之间相距的字节数。

- 例6.12 算术操作符的使用，设有如下定义：

```
ORG 0
VAL=4
DA1  DW  6，2，9，3
DA2  DW  15，17，24
COU=$-DA2
```

- VAL是常数，无需确定它的位置就可以使用。DA1和DA2是变量的符号地址，它们在内存中有确定的位置，我们只能根据它们的地址才能访问。

```
MOV   AX，DA1*4          ；错，地址乘或除，没有意义
MOV   AX，DA1*DA2        ；错，地址乘或除，没有意义
```

```
MOV   AX，DA1+DA2      ；错，地址相加，没有意义
MOV   AX，BX+VAL       ；错，BX+VAL须用指令实现
MOV   AX，[BX+VAL]     ；地址表达式，汇编成MOV AX，
                         [BX+4]
MOV   AX，DA1+VAL      ；地址表达式，汇编成MOV AX，[4]
MOV   AX，[DA1+VAL]    ；地址表达式，汇编成MOV AX，[4]
MOV   AX，VAL*4/2      ；数字表达式，汇编成MOV AX，8
MOV   AX，[VAL*4/2]    ；数字表达式，汇编成MOV AX，8
MOV   CX，(DA2-DA1)/2  ；得到DA1区数据个数，
                         汇编成MOV CX，4
MOV   BX，COU          ；得到DA2区的字节数，
                         汇编成MOV BX，6
```

2. 逻辑与移位操作符

- 逻辑操作符：AND，OR，NOT, XOR。
- 移位操作符：SHL和SHR。
- 格式：expression 操作符 number
- 逻辑与移位操作符都是按位进行的。
- 逻辑与移位操作符都只能用于数字表达式中。

- 例6.13 逻辑操作符的使用

```
ARY  DW   8
VAL=4
MOV   AX，BX AND 0FFH    ；错，BX AND VAL
                          须用指令实现
MOV   AX，ARY AND 0FFH   ；错，ARY AND
                           VAL须用指令实现
MOV   AX，VAL AND 0F0H   ；汇编成MOV  AX，0
AND   AX，VAL OR 0F0H    ；汇编成AND  AX，
                           0F4H
```

- 例6.14 移位操作符的使用

```
ARY  DW   8
VAL=4
MOV   AX，BX SHL 2     ；错，BX 左移须用指令实现
MOV   AX，ARY SHL 2    ；错，ARY 左移须用指令实现
MOV   AX，VAL SHL 2    ；汇编成MOV  AX，10H
MOV   AX，8  SHL 2     ；汇编成MOV  AX，20H
MOV   AX，VAL SHL 15   ；汇编成MOV  AX，00H
```

3. 关系操作符

- 关系操作符用来对两个操作数的大小关系作出判断。

  EQ（相等）

  NE（不相等）

  LT（小于）

  LE（小于等于）

  GT（大于）

  GE（大于等于）

- 关系操作符的两个操作数必须都是数字，或是同一段内的两个存储器地址。
- 计算结果为逻辑值，结果为真表示为0FFFFH，结果为假表示为0。

- 例6.15 关系操作符的使用

```
VAL=4
MOV   AX，BX  GT 2    ；错，BX 是否大于2须用
                        指令实现判断
MOV   AX，VAL GE 2    ；汇编成MOV  AX，FFFFH
MOV   AX，8  LE VAL   ；汇编成MOV  AX，0
```

4. 数值回送操作符

- TYPE
- LENGTH
- SIZE
- OFFSET
- SEG

（1）TYPE

- 格式：TYPE expression
- 表达式为变量，则汇编程序回送该变量的以字节数表示的类型。

  DB　回送1　　DW　回送2
  DD　回送4　　DF　回送6
  DQ　回送8　　DT　回送10
- 表达式为标号，则汇编程序回送代表该标号类型的数值。

  NEAR　回送-1
  FAR　回送-2
- 表达式为常数则回送0。

（2） LENGTH

- 格式： LENGTH variable
- 若变量用DUP定义，则返回总变量数，否则为1。
- 嵌套的DUP不计。所以，对于使用嵌套的DUP复制的数据不能据此得到正确的总变量数。

（3）SIZE

- 格式：SIZE variable
- 若变量用DUP定义，则返回总字节数，否则为单个变量的字节数。
- 嵌套的DUP不计，所以，对于使用嵌套的DUP复制的数据不能据此得到正确的总字节数。

（4） OFFSET

- 格式：OFFSET　variable或label
- 操作：回送变量或标号的偏移地址。

（5） SEG

- 格式：SEG　variable或label
- 操作：回送变量或标号的段地址。

- 例6.16 数值回送操作符的使用

  设有如下定义：

  ```
  ORG  0
  VAL=4
  ARR   DW  4  DUP(3)
  BUF   DW  4  DUP( 4 DUP(3))
  DAT   DW  15，17，24
  STR   DB  ‘ABCDEF’
  ```

  汇编程序对下面的指令汇编结果为：

  ```
  MOV   AX，TYPE ARR          ；汇编成MOV  AX，2
  MOV   AX，LENGTH  ARR       ；汇编成MOV  AX，4
  ```

```
MOV   AX，LENGTH BUF      ；汇编成MOV AX，4
MOV   AX，LENGTH DAT      ；汇编成MOV AX，1
MOV   AX，SIZE ARR        ；汇编成MOV AX，8
MOV   AX，SIZE BUF        ；汇编成MOV AX，8
                            （不是32）
MOV   AX，SIZE DAT        ；汇编成MOV AX，2
MOV   AL，SIZE STR        ；汇编成MOV AX，1
MOV   AX，OFFSET ARR      ；不完整的机器指令
MOV   BX，SEG ARR         ；不完整的机器指令
```

运算符的优先级

- 当一个表达式中同时有几个运算符时，按运算符优先级顺序执行。
- 汇编源程序时，汇编程序按照下列规则计算表达式的值：

（1）先执行优先级高的运算

（2）优先级相同的操作，从左至右顺序进行

（3）可以用圆括号改变运算的顺序