第四章 汇编语言程序设计基本方法

内 容 简 介

全面掌握8086/8088指令系统的使用，包括指令的功能、寻址方式及其书写格式、对标志位的影响、使用注意事项。掌握汇编语言程序设计所必须的伪指令，并由此构成汇编语言程序的完整结构。掌握变量、常量及伪指令的使用和一些常用的基本程序设计方法。在分支程序设计中，要特别注意每个分支的完整性和分支条件的合理使用；在循环程序设计中，掌握循环程序的基本结构，特别要注意应避免出现死循环；在子程序设计中，着重掌握参数的

第四章 汇编语言程序设计基本方法

内 容 简 介

各种传递方式及其实现，对堆栈这种特殊的存储区域进行了详细的描述，切实掌握堆栈的使用。宏指令和字符串操作是汇编语言程序设计中的两个难点，教材中对此也作了详细的介绍，要求掌握正确使用宏指令和字符串操作指令。

教材中简要介绍了DOS功能调用的方法和常用的一些DOS功能，要求能熟练使用INT 21H 的01、02、09、0AH和4CH号等功能。

4.1 汇编语言基础

一. 汇编语言与机器语言的相关概念

汇编语言与机器语言属于低级语言，它们与高级语言有较大的区别，汇编语言中的语句与机器的型号密切相关。如Intel 8086系列CPU、Intel8031系列单片CPU等，若CPU型号不同，其指令系统就不同，当然语句的书写格式也就不同。

4.1 汇编语言基础

在高级语言中，完成某个加法功能，我们可采用语句 X=A+B ，只要给变量A和B赋一确定值，此加法就可以实现了。在汇编语言中则不同，程序必须指出A、B存放在何处，相加后的结果又存放在何处，然后才能实现这一加法运算。显然，汇编语言在通过程序告诉计算机做什么和如何做时，显得更加具体。正是这种具体，使得该语言与计算机（处理器）紧密相关，从而也要求学习和使用汇编语言的人对处理器的结构有更加深入的了解。

4.1 汇编语言基础

1. 机器语言

→ 机器（CPU）能直接认识的一种二进制代码语言。CPU能认识的一组二进制代码就是一条指令。

4.1 汇编语言基础

如：

就是一条指令代码，或者机器语言代码。

B0H
12H
；这一组二进制代码，就是告诉CPU将AL ← 12H。

05H
02H
00H
；这一组二进制代码，告诉CPU将（AL）+2 → AX。

4.1 汇编语言基础

2. 机器语言程序

→ 采用机器语言编写的程序，即二进制代码程序。

优点：程序送入计算机后，CPU可以直接执行。

缺点：不易书写，不易检查，编写程序十分困难。

4.1 汇编语言基础

3. 汇编语言

→ 为了克服机器语言不易书写、记忆复杂等缺点，人们采用一组字母、数字和符号来代替一条二进制代码指令，这种表示指令的符号称为助记符，这种用一组符号来代替一条指令编写程序时采用的语言，称为汇编语言。

4.1 汇编语言基础

4. 汇编语言程序

→ 用汇编语言编写的程序称为汇编语言程序，或者称为汇编语言源程序。这种编程方法称为汇编语言程序设计。汇编语言源程序名必须为文件名.ASM。

4.1 汇编语言基础

如:

用MOV AL，12H 代替 {B0H, 12H}

用ADD AX，0002H 代替 {05H, 02H, 00H}

4.1 汇编语言基础

优点：要比一串二进制代码清晰多了，书写容易，记忆也方便。

缺点：CPU不能直接执行。用汇编语言编写的汇编语言源程序必须经过汇编，将其翻译成机器语言格式， CPU才能执行。

5. 汇编

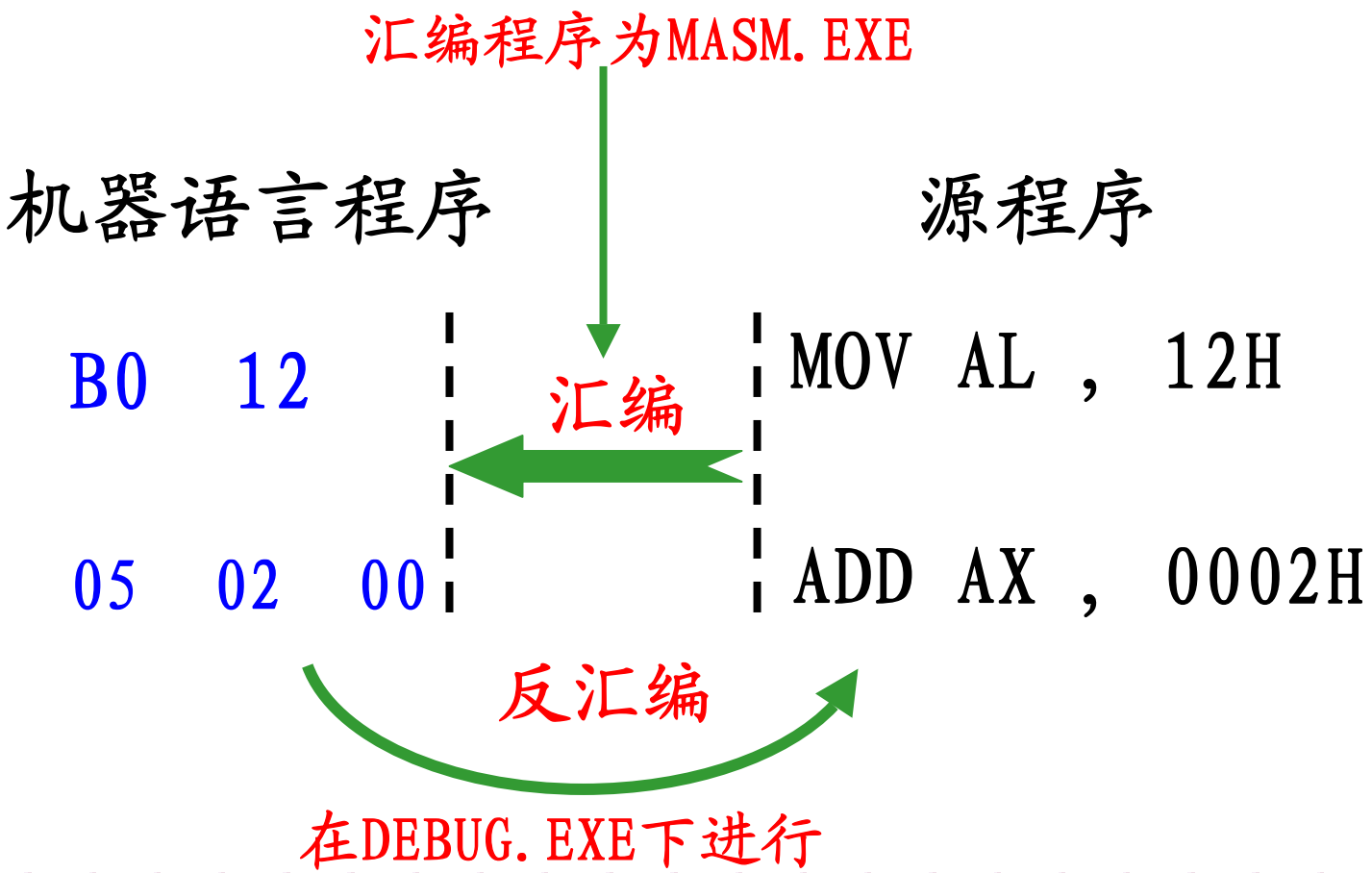
→ 把汇编语言源程序翻译成机器语言程序的过程称为汇编。

6. 汇编程序

→ 能把汇编语言源程序翻译成机器语言程序的系统程序（语言加工程序）。8086宏汇编程序为MASM.EXE。

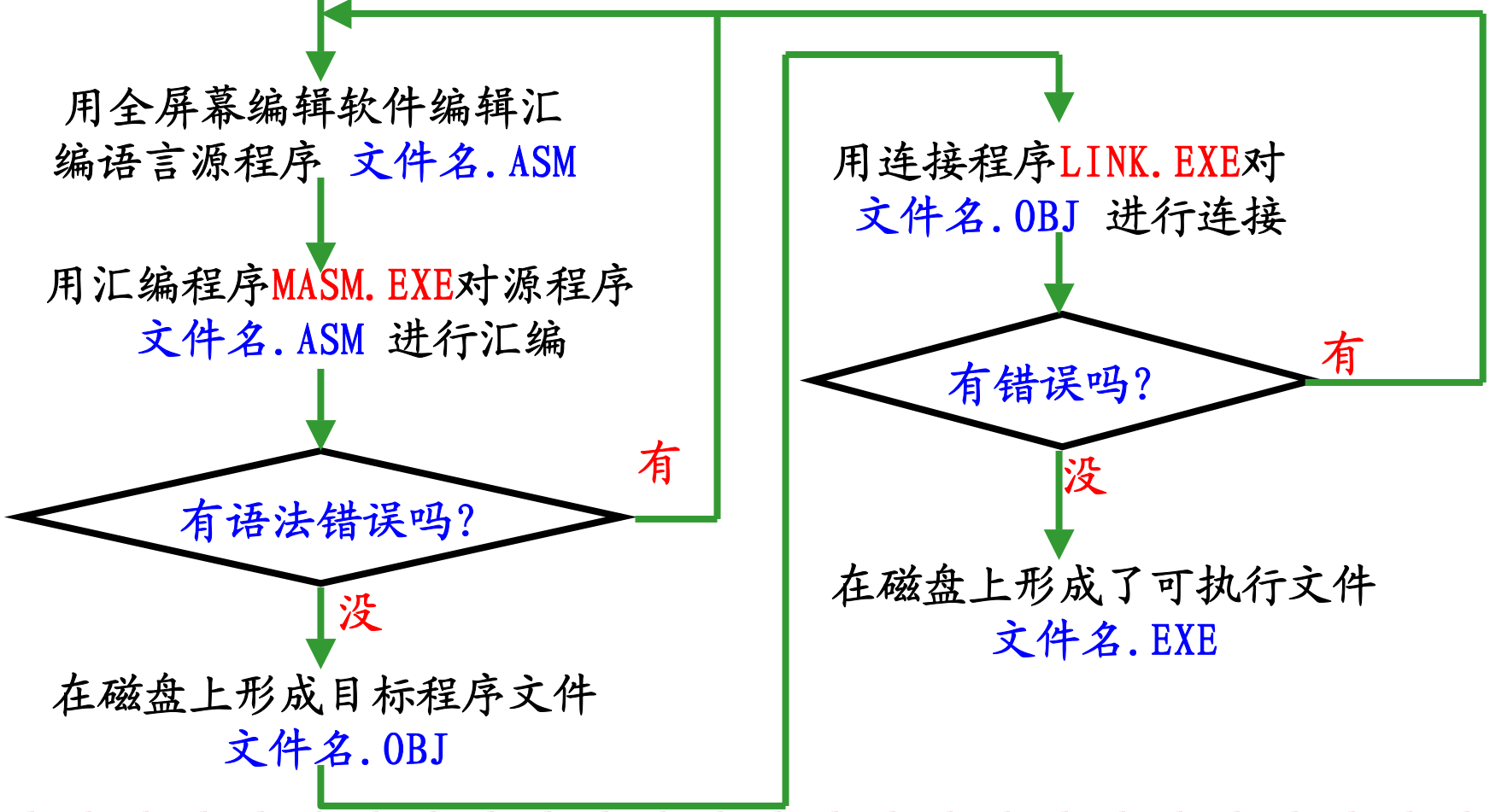
4.1 汇编语言基础

如：


汇编程序为MASM.EXE
机器语言程序
源程序
B0 12
MOV AL , 12H
汇编
05 02 00
ADD AX , 0002H
反汇编
在DEBUG.EXE下进行

4.1 汇编语言基础

7. 从汇编语言源程序到可执行程序所经过的处理过程



4.1 汇编语言基础

二. 汇编语言程序中语句的种类

在汇编语言程序设计中，程序中的语句有

三类：
- 指令语句
- 伪指令语句
- 宏指令语句

4.1 汇编语言基础

1. 指令语句

→ 汇编后能产生机器语言代码，是CPU能执行的语句。

2.伪指令语句

→ 汇编后不能产生机器语言代码，是CPU不能执行的语句。它只是告诉汇编程序（MASM.EXE）应如何汇编。

3. 宏指令语句

→ 它是8086指令系统中没有的指令，是用户自己根据宏指令定义的方法定义的一条能完成某一特定功能的新的指令。

4.1 汇编语言基础

三. 汇编语言中语句的组成

汇编语言源程序（文件名.ASM）是由一条条语句组成的。语句则由

- 标识符
- 操作助记符
- 操作数
- 注释

四部分组成。

其基本格式如下：

4.1 汇编语言基础

[标识符]	操作助记符	空格	[操作数]	[; 注释]

其中，操作助记符指出该条语句的基本操作功能，是必须有的部分。而[ ]项可有可无，视情况而定。

若是指令语句，标识符就是一个标号名，以冒号结尾；若是伪指令语句，标识符就是变量名或者段名等，以空格结尾。

4.1 汇编语言基础

标识符的第一个字符必须是字母，不能为数字，总字符个数不能超过31个。在给标识符起名时，不能用8086指令系统中的专用符来给标识符起名称，如 ADD、MOV等。起名时，尽量起的有点意义。

4.1 汇编语言基础

四. 汇编语言中的常数与表达式

在汇编语言程序中，语句中的操作数项既可以是常数或表达式（立即数），也可以是指明操作数所在处的一种说明。如果操作数是常数或表达式，则有以下几种形式。

4.1 汇编语言基础

1. 数值常数

若为数值常数，则按其基数的不同，可

有
- 二进制数
- 十进制数
- 八进制数
- 十六进制数

等不同的表达式。

4.1 汇编语言基础

如：将（AL）← 12，则有：

```
MOV AL , 12          ;12为十进制数
MOV AL , 0CH         ;0CH为十六进制数
MOV AL , 00001100B   ;00001100B为二进制数
MOV AL , 140         ; 140为八进制数
```

2. 字符串常数

字符串常数是由单引号“ ′ …… ′ ”括起来的一串字符或者单个字符。

如：
```
MOV DL , ′ A ′        ; DL ←41H

BUF DB ′ 12Aa ′       ; 将′ 12Aa′ 字符串定义给
                      ; 变BUF以下连续的（4个字
                      ; 节）存储器单元。
```

3. 表达式

语句中的操作数项也可以是表达式。表达式由操作数和操作符组成。操作符有:

4.1 汇编语言基础

算术操作符：+、-、*、/、MOD

逻辑操作符：ADD、OR、XOR、NOT

关系操作符：EQ(相等)、NE(不等)、LT(小于)、GT(大于)、LE(小于或等于)、GE(大于或等于)

属性操作符：SEG、OFFSET、TYPE、LENGTH、SIZE

属性修改操作符：PTR

4.1 汇编语言基础

①算术操作符

MOV AL , 5+2*3 等效于 MOV AL , 11

MOV AL , 11/2 等效于 MOV AL , 05H;取商

MOV AL , 11 MOD 2 等效于MOV AL , 01H;取余

4.1 汇编语言基础

②逻辑操作符

等效于

```
MOV AL , 0CCH AND 0F0H
MOV AL , 0C0H
```

等效于

```
AND AL , 0CCH OR 0F0H
AND AL , 0FCH
```

CPU执行时完成的操作

汇编程序汇编时完成的操作

4.1 汇编语言基础

③关系操作符

若关系成立，则为真，取值全1;
若关系不成立，则为假，取值全0。

如:

```
MOV AL , 04H LT 05H ;关系成立为真
```

等效于

```
MOV AL , 0FFH
```

4.1 汇编语言基础

④属性操作符

```
MOV BX , OFFSET TABLE ;BX
MOV AX , SEG   TABLE  ;AX
MOV DL , TYPE  TABLE  ;DL
```

取变量TABLE单元的偏移地址

取变量TABLE单元的段地址

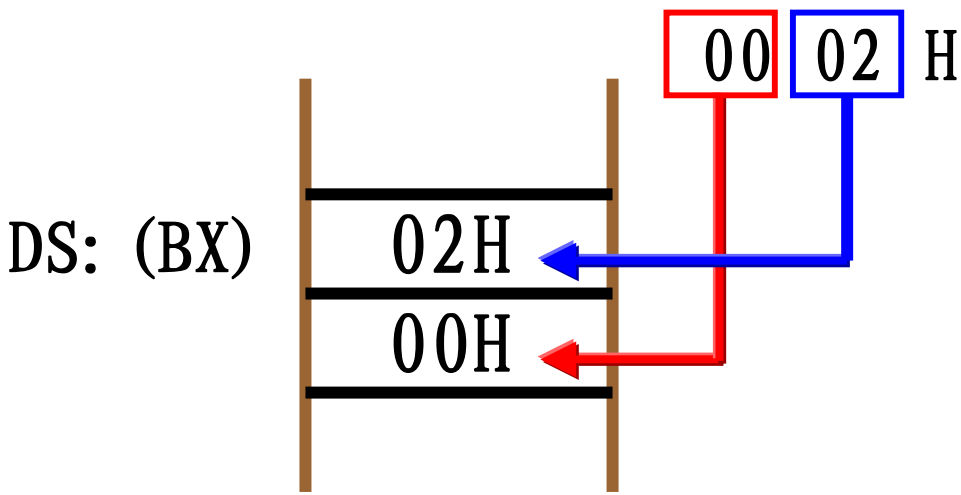
取变量TABLE的类型

SRC均为立即数寻址

4.1 汇编语言基础

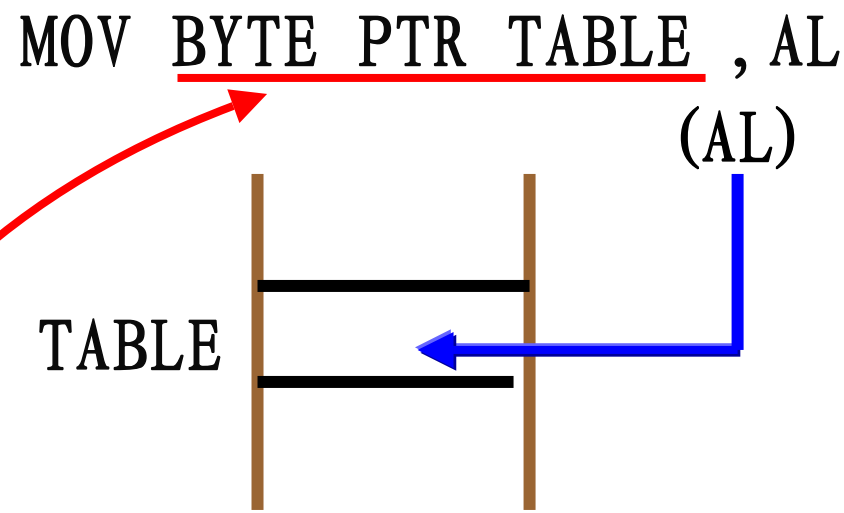
⑤属性修改操作符

```
MOV WORD PTR [BX] ,02H
```



4.1 汇编语言基础

⑤属性修改操作符



只是在本条语句中用PTR将TABLE变量临时修改为字节型变量，脱离了本条语句，变量TABLE的类型恢复原样。

4.1 汇编语言基础

五. 标号、变量及伪指令

前面已经讲过，在汇编语言程序设计中，为了方便记忆，将直接访问的存储器单元的实际地址符号化，即给要访问存储器单元起一个标识符名，而标识符有标号名、变量名等。

4.1 汇编语言基础

1. 标号

→ 用以指示某条指令语句的位置（地址）。其定义方法就是在指令语句的操作助记符前加上标号名，以冒号结尾。它可以作为程序转移指令的操作数。

4.1 汇编语言基础

如：

```
        ●
        ●
        ●
        JMP  FMAX
        ●
        ●
        ●
        ●
        ●
FMAX:   MOV  AX ,0
        ●
        ●
        ●
```

4.1 汇编语言基础

➡ 标号一但定义，就具有了以下三个属性：

- 段地址 —— 标号对应的指令所在段的段地址
- 段内偏移地址 —— 标号对应的指令所在的段内EA
- 类型
  - NEAR型 —— 该标号与转移指令在同一代码段。
  - FAR型 —— 该标号与转移指令不在同一代码段。

2. 变量

→ 用以指示存放数据的存储器单元的符号地址。变量所指明的存储器单元的值，在程序运行期间是可以改变的。

4.1 汇编语言基础

- 变量定义伪指令（也称为数据定义伪指令或者称为存储器分配伪指令）

变量定义伪指令主要应用在数据段，是用来给变量名所对应的存储器单元分配数据或预留空间。变量定义伪指令有以下五种：

4.1 汇编语言基础

[变量名] DB 表达式 ; 定义字节型变量

[变量名] DW 表达式 ; 定义字型变量

[变量名] DD 表达式 ; 定义双字型变量

[变量名] DQ 表达式 ; 定义长字型变量

[变量名] DT 表达式 ; 定义一个10字节的变量

常用的变量定义伪指令有DB、DW、DD。伪指令左边的变量名可有可无，若有必须以空格结尾。

4.1 汇编语言基础

变量定义伪指令语句中的表达式有以下几种情况:

- 1个或多个常数或表达式。当为多个时，其间用逗号分割。
- 带引号的字符串。
- 一个问号（？）。（用来将此单元保留，存放结果）
- 重复方式。其格式为:

  重复次数 DUP （表达式）

4.1 汇编语言基础

变量定义举例

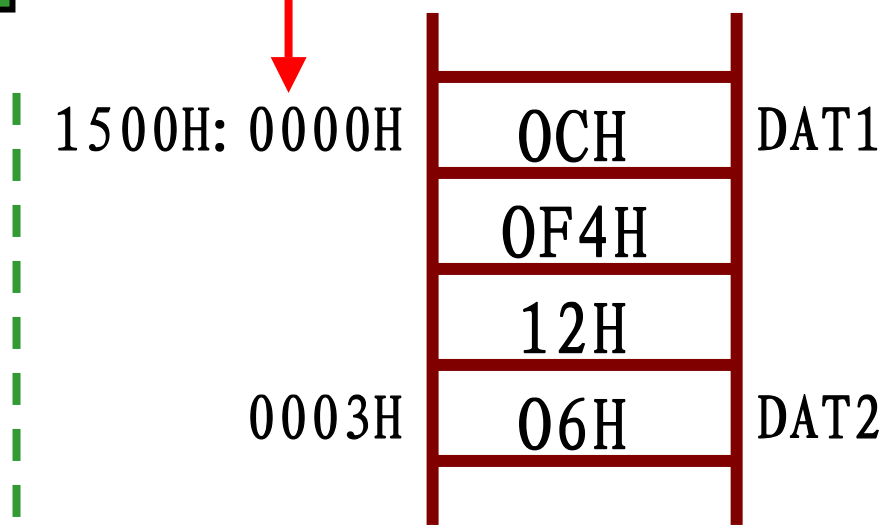
若（DS）=1500H，且在数据段0000H偏移地址开始有以下变量定义。

4.1 汇编语言基础

汇编程序对本段汇编后，各变量对应存储器单元的内容如下图。

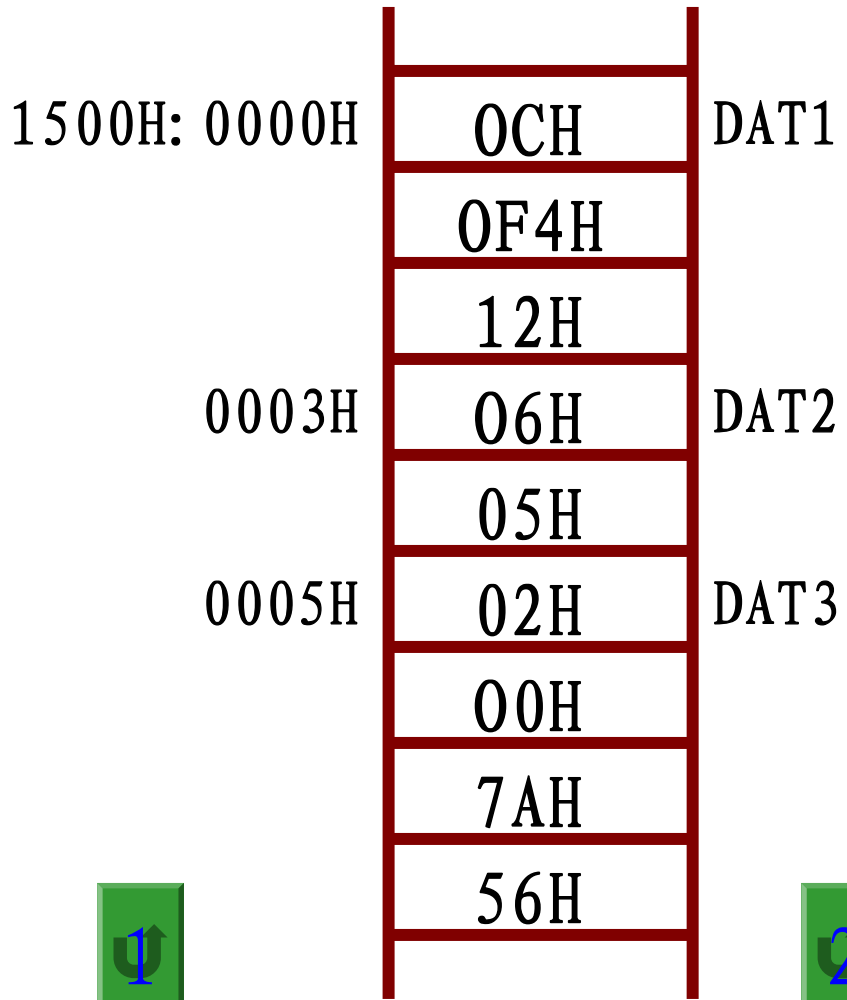
此时，MASM.EXE中的软件位置计数器$=0000H

```
DAT1 DB 12, -12, 12H
DAT2 DB 2*3, $+2
```



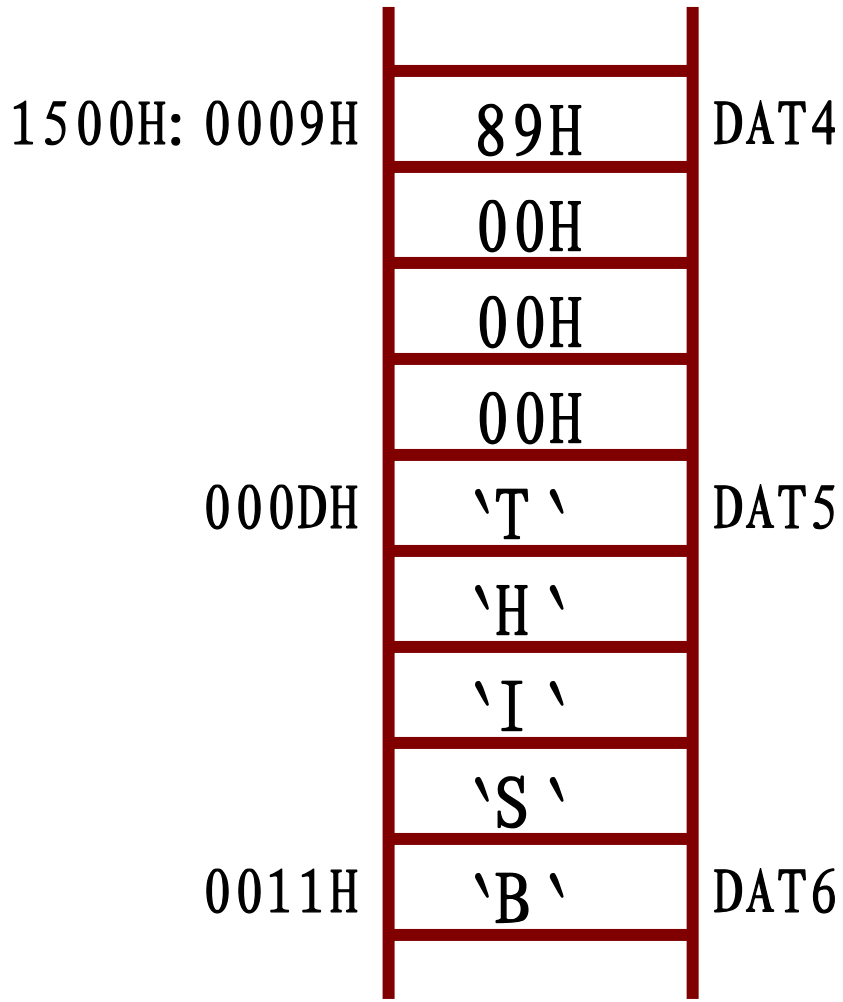
4.1 汇编语言基础

```
DAT1  DB  12, -12, 12H
DAT2  DB  2*3, 15/3
DAT3  DW  02H, 567AH
DAT4  DD  89H
DAT5  DB  ' THIS'
DAT6  DW  ' AB' , ' C'
DAT7  DB  3 DUP (00H)
DAT8  DW  3 DUP (?)
DAT9  DW  DAT6
DAT10 DD  DAT8
DAT11 DB  ?
```



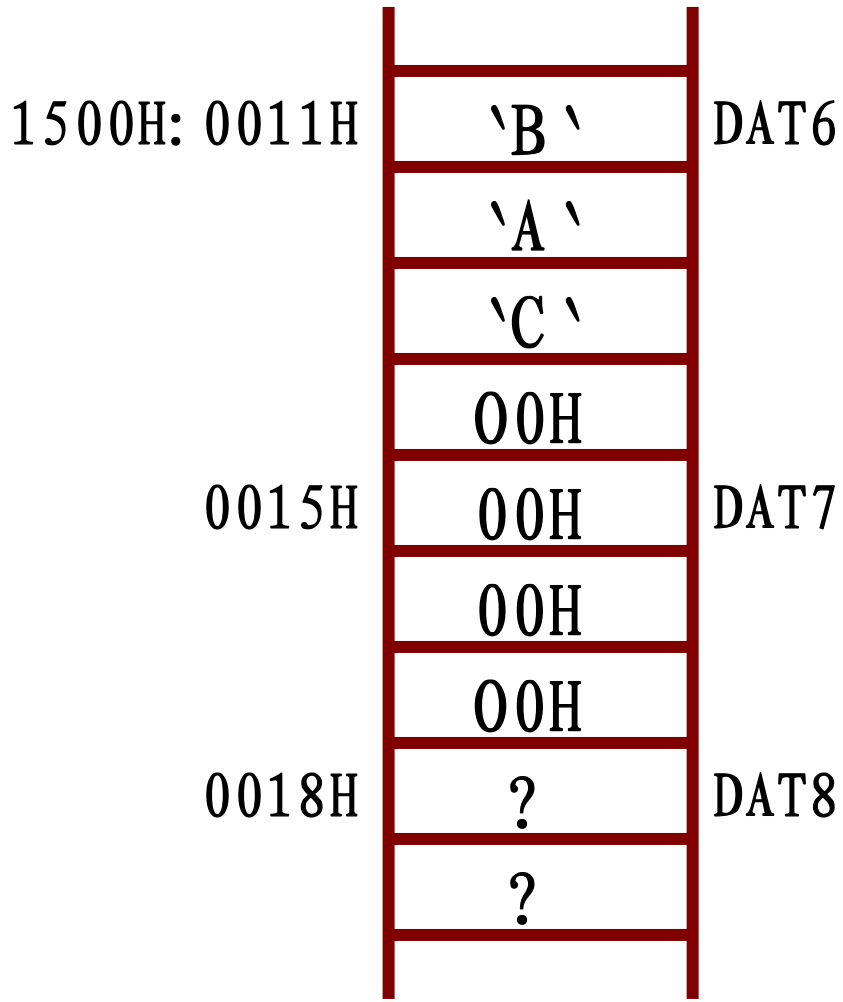
4.1 汇编语言基础

```
DAT1  DB  12, -12, 12H
DAT2  DB  2*3, $+2
DAT3  DW  02H, 567AH
DAT4  DD  89H
DAT5  DB  ' THIS'
DAT6  DW  ' AB' , ' C'
DAT7  DB  3 DUP (00H)
DAT8  DW  3 DUP (?)
DAT9  DW  DAT6
DAT10 DD  DAT8
DAT11 DB  ?
```



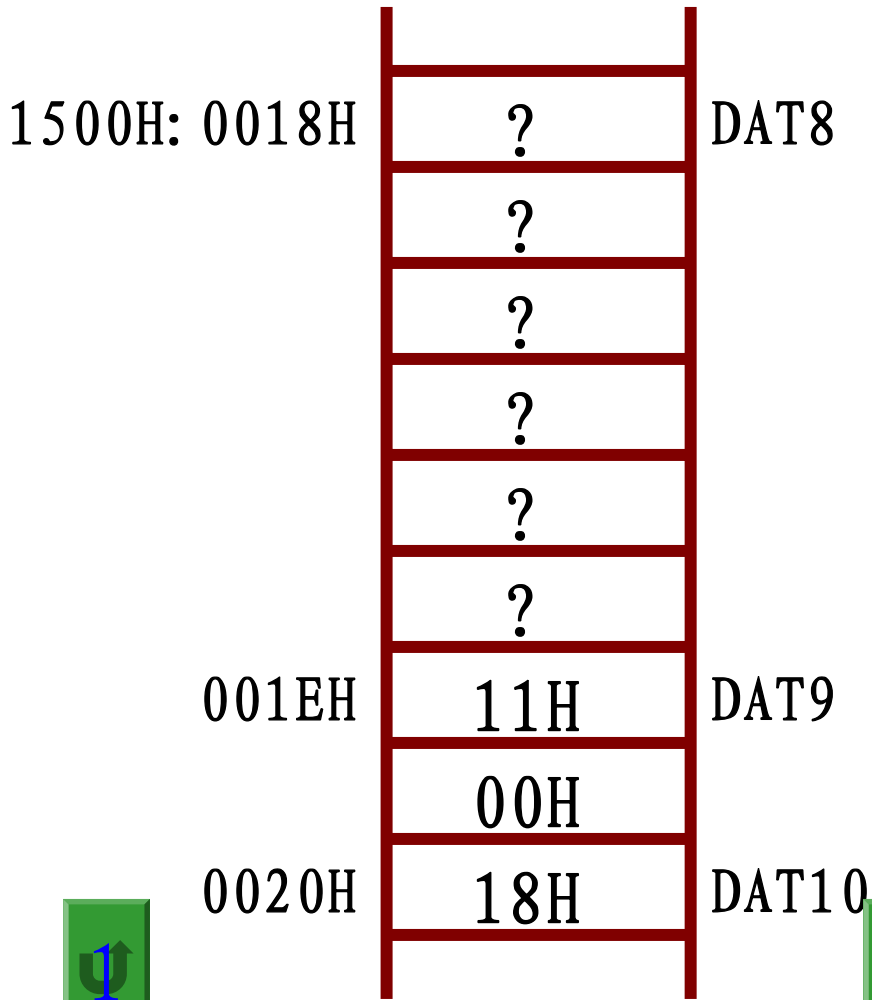
4.1 汇编语言基础

```
DAT1  DB  12, -12, 12H
DAT2  DB  2*3, $+2
DAT3  DW  02H, 567AH
DAT4  DD  89H
DAT5  DB  ' THIS'
DAT6  DW  ' AB' , ' C'
DAT7  DB  3 DUP (00H)
DAT8  DW  3 DUP (?)
DAT9  DW  DAT6
DAT10 DD  DAT8
DAT11 DB  ?
```



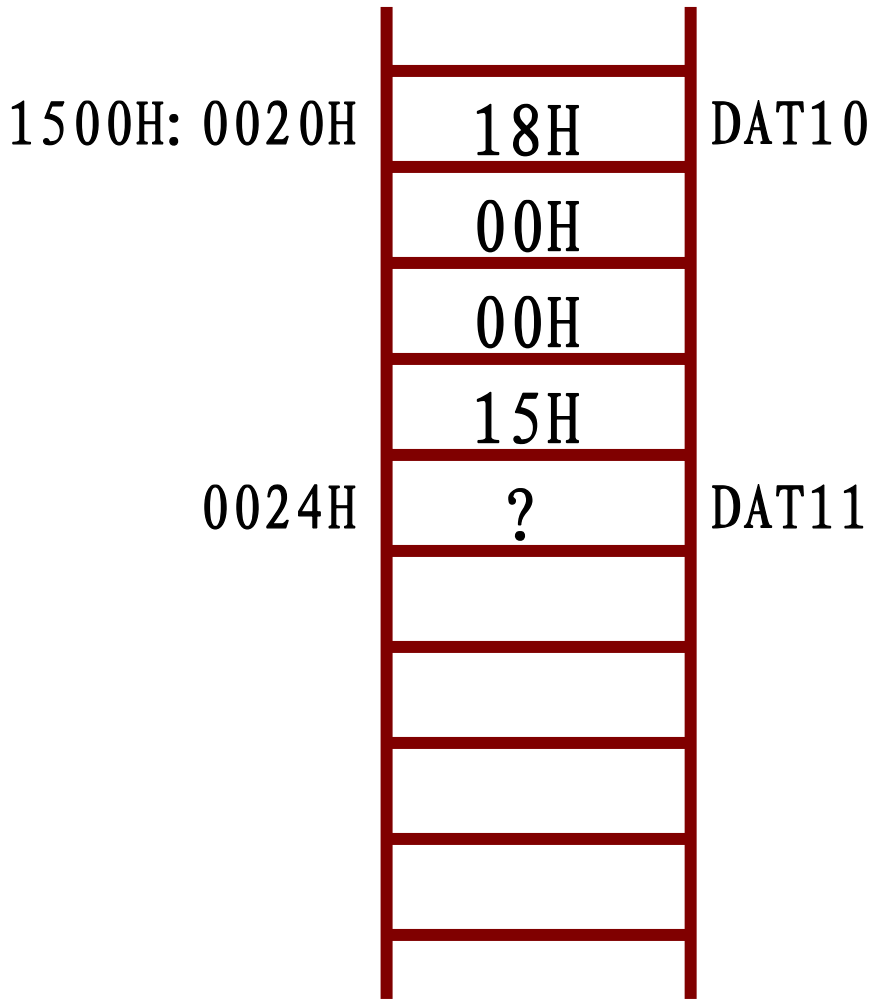
4.1 汇编语言基础

```
DAT1  DB  12, -12, 12H
DAT2  DB  2*3, $+2
DAT3  DW  02H, 567AH
DAT4  DD  89H
DAT5  DB  ' THIS'
DAT6  DW  ' AB' , ' C'
DAT7  DB  3 DUP (00H)
DAT8  DW  3 DUP (?)
DAT9  DW  DAT6
DAT10 DD  DAT8
DAT11 DB  ?
```



4.1 汇编语言基础

```
DAT1  DB  12, -12, 12H
DAT2  DB  2*3, $+2
DAT3  DW  02H, 567AH
DAT4  DD  89H
DAT5  DB  ' THIS'
DAT6  DW  ' AB' , ' C'
DAT7  DB  3 DUP (00H)
DAT8  DW  3 DUP (?)
DAT9  DW  DAT6
DAT10 DD  DAT8
DAT11 DB  ?
```



4.1 汇编语言基础

变量的属性

一个变量一旦定义了，就具有了以下五个属性：

- 段地址（SEG）
- 段内偏移地址（OFFSET）
- 类型（TYPE）
- 长度（LENGTH）
- 大小（SIZE）

4.1 汇编语言基础

其中，
- 段地址为变量所在段的段地址
- 段内偏移地址为变量对应单元的偏移地址

4.1 汇编语言基础

● 类型

为每个变量所占的字节数，对于DB、DW、DD、DQ、DT定义的变量其类型分别为1，2，4，8，10。通常又将DB、DW、DD所定义的变量称为BYTE类型，WORD类型和DWORD类型变量。

4.1 汇编语言基础

- 长度:

变量定义时，一个变量名所定义的变量的个数。在含有DUP操作符的变量定义中，变量名所定义的变量个数为定义格式中的重复次数。在其它各种变量定义中，一个变量名所定义的变量个数均为1。

- 大小

变量定义时，分配给同一个变量名的所有变量的总的字节数。所以某变量名的大小为该变量名的长度与它的类型的乘积。

即：SIZE=LENGTH*TYPE

4.1 汇编语言基础

例：对于前面变量定义例子中各变量的定义，我们有：

```
MOV AX , SEG DAT1       ;  (AX)=1500H
MOV AX , SEG DAT10      ;  (AX)=1500H
MOV AX , OFFSET DAT3    ;  (AX)=0005H
MOV AL , TYPE DAT3      ;  (AL)=02H
```

前 后

4.1 汇编语言基础

```
MOV AX , LENGTH DAT3   ;  (AX)=0001H
MOV AX , LENGTH DAT8   ;  (AX)=0003H
MOV AX , SIZE DAT3     ;  (AX)=0002H
MOV AX , SIZE DAT8     ;  (AX)=0006H
```

以上指令中，
SRC均为立即数寻址