



汇编语言程序设计

第四章 汇编语言程序组织

张华平 副教授 博士

Email: kevinzhang@bit.edu.cn

Website: <http://www.nlpir.org/>

@ICTCLAS张华平博士

大数据搜索挖掘实验室 (wSMS@BIT)





(2015-2016-1)-COM07023-t04745-1 张华平 汇编语言程序设计
15 第15周周2 (2015-12-15) 15:10-17:10 111
55 信3008
(2015-2016-1)-COM07023-t04745-1 张华平 汇编语言程序设计
15 第15周周2 (2015-12-15) 15:10-17:10 111
56 信3006 张华平

大数据时代的创新 研究生楼101 吴甘沙 Intel中国研究院院长





本章要求：

- ① 编写简单的、完整的汇编程序。
- ② DEBUG程序调试。

主要内容：

汇编源程序结构、常用伪指令及操作符、汇编语言程序上机过程、数据的输入输出等。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

4.1 汇编语言语句格式

汇编语句：**指令、伪指令、宏指令**。

每条**指令**语句都生成机器代码，各对应一种CPU操作，在程序运行时执行。

伪指令语句由汇编程序在汇编过程中执行，数据定义语句分配存储空间，其它伪指令不生成目标码。

宏指令是用户按照宏定义格式编写的一段程序，可以包含指令、伪指令、甚至其他宏指令。





汇编语言语句格式:

[名字] 助记符 <操作数> [;注释]
]

其中带[]的内容是可选的。





名字域是语句的符号地址，可以由26个大小写英文字母、0~9数字、_、\$、@、? 等字符组成，数字不能出现在名字的第一个字符位置。

指令的名字叫做标号，必须以冒号（:）结束。它提供给循环或转移指令的转向地址。

伪指令的名字可以是变量名、过程名、段名等。通常，名字具有三属性：段基址、偏移量和类型。

标号的类型有NEAR型和FAR型，变量的类型有字节、字、双字、四字等。

[返回](#)



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



助记符域给出操作的符号表示，可以是指令助记符、伪指令助记符等。例如加法指令的助记符是ADD。

操作数域为操作提供必要的信息。每条指令语句的操作数个数已由系统确定，例如加法指令有两个操作数。

注释域用以说明本条语句在程序中的功能，要简单明了。注释以分号（;）开始。

[返回](#)



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



4.2 汇编语言源程序结构

用汇编语言编写程序可以使用两种基本格式。

一种是使用完整段定义；

一种是使用简化段定义。

本节介绍使用完整段定义格式书写的汇编语言源程序。

. EXE文件结构 /FDISK XCOPY

. COM文件结构 /FORMAT SYS





一、典型的 .EXE 文件结构

.EXE 文件 是一种可执行程序，它是一个可重定位的装入模块，可以包含多个段，文件的总长度可以超过 64K。

.EXE 程序 由文件头和程序本身的二进制代码两部分组成。

.EXE 结构 是 DOS 普遍采用的一种格式，DOS 的大多数应用程序采用 .EXE 结构，例如 FDISK。





DOS装入.EXE文件的过程:

- ① DOS的装入程序为.EXE程序建立一个256字节的程序段前缀PSP。其中
- ② 把文件头读入内存工作区。
- ③ 计算可执行模块的大小。
- ④ 计算装入的起始段地址。
- ⑤ 完成重定位。
- ⑥ 初始化段寄存器和指针寄存器。
- ⑦ 把控制权交给.EXE程序。






装入程序对段和指针寄存器的设置为：

CS: IP为主程序的入口地址(程序装入后执行的第一条指令地址)。**SS**为堆栈段的段基址，**SP**指向栈顶。其它段寄存器全部被初始化为指向**PSP**的段基址，以便用户能够访问**PSP**中的信息。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



例. 编程实现 $123+456 \rightarrow \text{SUM}$ 单元的功能。(程序4.1)





1. 段定义伪指令

基本格式:

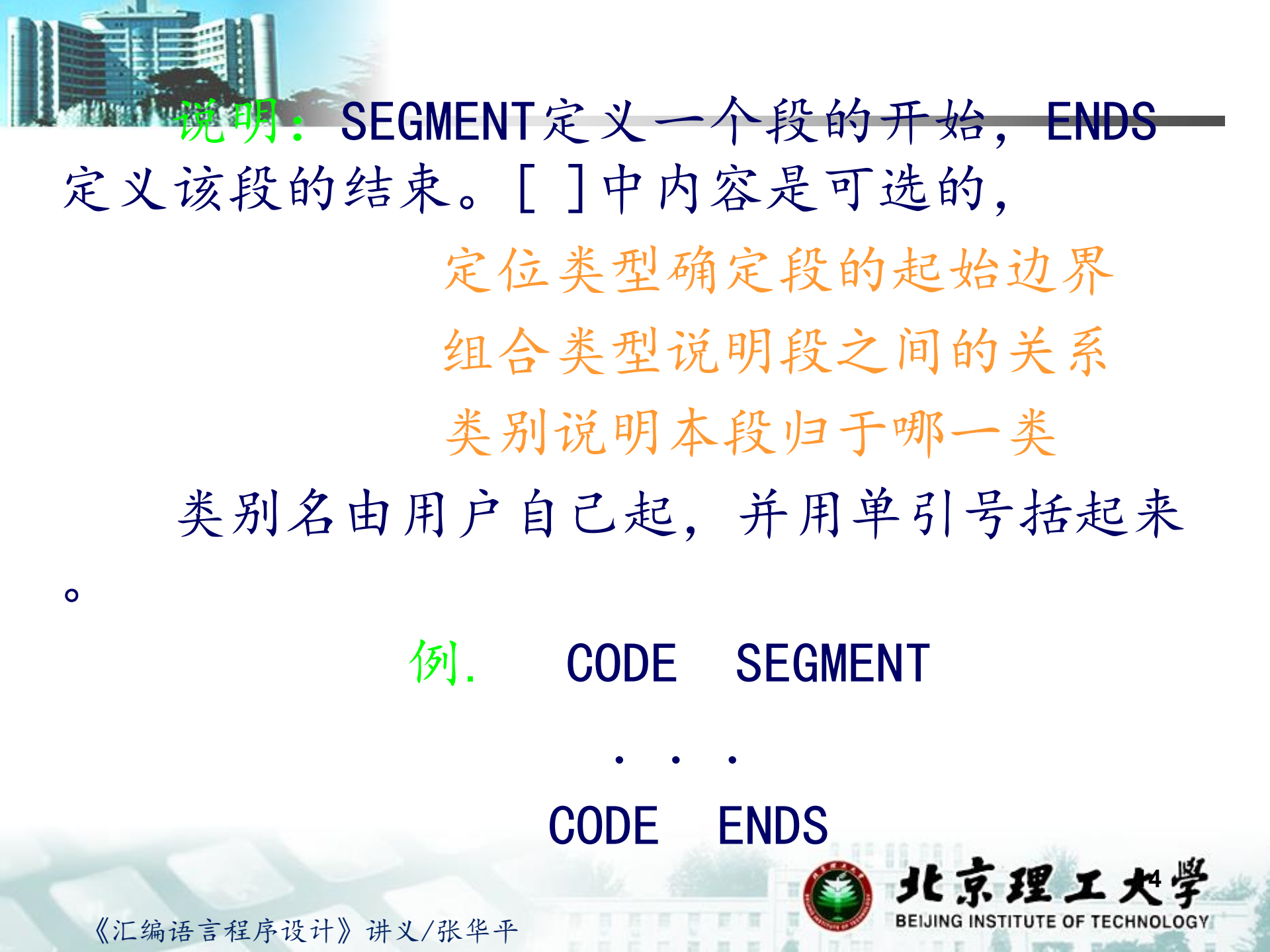
段名 SEGMENT [定位类型][组合类型][‘类别’]

• • •

段名 ENDS

功能: 定义一个段。





说明: SEGMENT 定义一个段的开始, ENDS
定义该段的结束。[] 中内容是可选的,

定位类型确定段的起始边界

组合类型说明段之间的关系

类别说明本段归于哪一类

类别名由用户自己起, 并用单引号括起来

。

例. CODE SEGMENT

. . .

CODE ENDS



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



2. ASSUME伪指令

格式：ASSUME 段寄存器名:段名
[,]

功能：说明段寄存器和段之间的关系。

说明：若一个段寄存器与NOTHING关联,则表示取消前边对该段寄存器的假设,可以省略。

ASSUME语句并不给段寄存器赋值,它应放在引用段寄存器之前,通常放在代码段或主过程的第一个语句位置。





3. 过程定义伪指令

格式:

过程名 **PROC** [类型]

· · ·

过程名 **ENDP**

功能：定义一个过程。





说明：汇编语言中无论是主程序还是子程序都以**过程**形式出现。一个代码段可以含有多个过程，具有.EXE结构的主过程必须是**FAR**型。过程名由用户自己起。类型选项指明该过程类型，可以是：

NEAR（或缺省）——说明该过程是近型的，只能在段内被调用。

FAR ——说明该过程是远型的，可以在段间被调用，也可以在段内被调用。



4. 程序结束伪指令

格式：END [过程名]

功能：表示源程序结束。

说明：过程名指示程序执行起始地址。
。[]中过程名是可选的。只有主过程模块的END后可带过程名，它必须是主程序名。若一个程序由多个模块组成，则除主模块外，其它模块的END语句不能带过程名。

例如。END MAIN



5. 使DS指向用户程序的数据段

开始：DS→PSP

在用户程序运行过程中，DS应指向程序自己的数据段以便访问其中内容，例如我们要访问A、B和SUM变量。为此，应在程序中用指令为DS等段寄存器赋值。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

6. 如何返回DOS

MAIN PROC FAR

PUSH DS ;PSP段基址入栈

XOR AX, AX ;清0

PUSH AX ;数字0入栈

... ;完成程序指定功能

RET

MAIN过程是FAR型, 执行RET时从栈中弹出0给IP, 再弹出一个字(PSP段基址)给CS, 现在CS:IP指向PSP:0处的指令INT 20H。

INT 20H指令功能: 退出应用程序, 释放所占内存并返回DOS。调用时要求CS指向PSP段基址。





系统调用实现返回DOS

方法是功能号4CH→AH寄存器, 返回码→AL, 正常返回时返回码为0。使用这种方法。程序4.1的代码段可以改写为如[程序4.2](#)所示的结构。





CODE SEGMENT : 程序4.2

```
MAIN PROC FAR
    ASSUME CS:CODE , DS:DATA
    ASSUME SS:STASG
    MOV AX , DATA ;无程序前奏
    MOV DS , AX
    MOV AX , A
    ADD AX , B
    MOV SUM, AX
    MOV AX , 4C00H
    INT 21H
MAIN ENDP
CODE ENDS
END
```

返回



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



程序4.1和程序4.2中大部分语句都是**任何一个 .EXE结构所必须的**。编写 .EXE程序时可把它们作为一个标准框架。

以上两种 .EXE结构编程时都可以使用。程序4.1结构使得对汇编语言编程及内部运行机制有更深入的了解，程序4.2使得程序更简洁。





二、典型的.COM文件结构

.COM文件是一种可执行程序，它的总长度不能超过64K，整个文件只能由一个段组成。

它没有文件头，只包含程序本身的二进制代码。这种结构代码紧凑，与实现同功能的.EXE文件相比，占用内存更少，速度更快，因此适合编制较小的程序，例如DOS的外部命令SYS、FORMAT等都是.COM结构。





DOS装入.COM文件时先建立一个程序段前缀PSP, 其段长为100H字节, PSP:0处存放一条INT 20H指令, 把程序的二进制代码紧跟PSP之后装入。但是因为.COM程序的代码、数据及堆栈数据在同一段中, 所以对所有的段寄存器都初始化为指向PSP的段基址。 $IP=100H$, 为PSP之后的下一个地址偏移量, SP 指向栈顶, 栈顶中存入一个字型数字0, 如图4-1所示。

对于实现上例的.COM结构的程序如下。





CS、DS、ES、SS

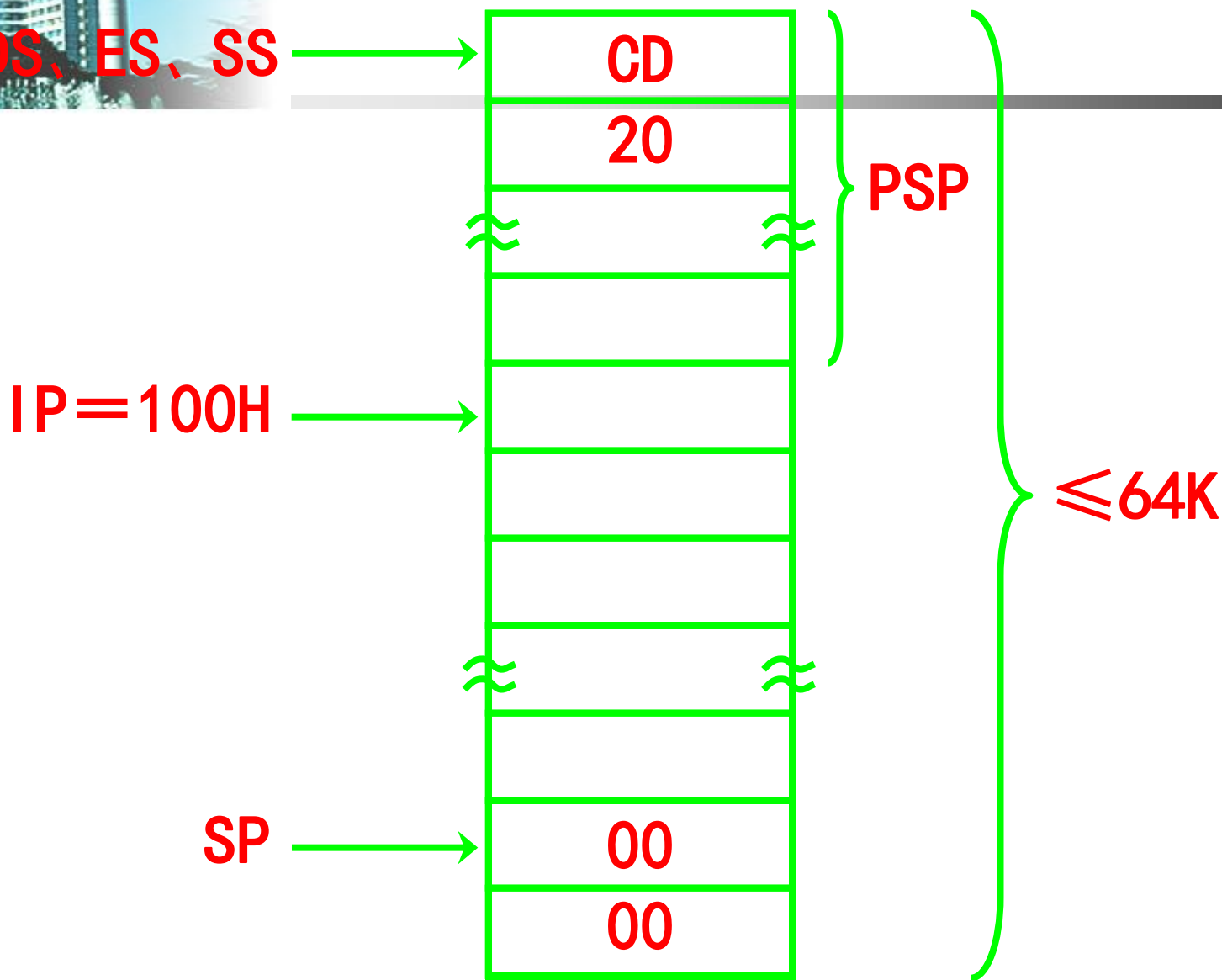


图4 -1 .COM文件装入内存示意图

返回



BEIJING INSTITUTE OF TECHNOLOGY



CODE

SEGMENT

;程序4.3

ORG 100H

ASSUME CS:CODE, DS:CODE

MAIN

PROC NEAR

JMP START

A DW 123

B DW 456

SUM DW ?

START: MOV AX, A

ADD AX, B

MOV SUM, AX

RET

MAIN ENDP

CODE ENDS

END MAIN



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



ORG伪指令格式：ORG 数值表达式

功能：设置地址计数器内容为数值表达式的值。

说明：在汇编程序对源程序汇编的过程中，使用地址计数器保存当前正在汇编的语句地址（段内偏移量），汇编语言允许用户直接用\$引用地址计数器的当前值。

例. ORG 100H

；设置地址计数器的值为100H

例. ORG \$+6

；跳过6个字节的存储区域

返回



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



DS: CODE

令

```
CODE SEGMENT ;程序4.4
    ORG      100H
    ASSUME   CS: CODE ,
MAIN PROC     NEAR
    JMP      START
    . . .    ;数据定义伪指令
START:       . . .    ;指令语句
    . . .
    MOV      AX , 4C00H
    INT      21H
MAIN ENDP
CODE ENDS
END MAIN
```

[返回](#)



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



通过以上讨论可以看出，主程序为**NEAR型**、**ORG 100H**语句及所出现的位置、它之后第一个占用存储空间的语句应该是一条可执行的指令语句都是任何一个**COM**结构所**必须**的。





4.3 常用伪指令

- 一、数据定义伪指令
- 二、LABEL伪指令
- 三、符号定义伪指令
- 四、对准伪指令





一、数据定义伪指令

数据定义伪指令用来定义程序中使用的数据。这是一组使用频率很高的语句，因为大部分程序都会涉及到数据问题。

格式：**[变量名]** **助记符** **操作数**

功能：为变量分配单元，并为其初始化或者只预留空间。





说明：

①变量名是可选的，需要时由用户自己起。它是该数据区的符号地址，也是其中第一个数据项的偏移量。程序通过变量名引用其中的数据。





②

助记符是数据类型的符号表示。

助记符	数据类型	一个数据项字节数
DB (BYTE)	字节型	1
DW (WORD)	字型	2
DD (DWORD)	双字型	4
DQ (QWORD)	四字型	8
DF (FWORD)	六字节型	6
DT (TBYTE)	10字节型	10

注：()中是在MASM6.11版本中可以使用
的助记符。必须掌握DB、DW、DD。





③ 操作数

操作数可以是数字常量、数值表达式、字符串常量、地址表达式、?、<n> DUP（操作数，……）形式。





a. 数字常量及数值表达式

操作数可以是数值表达式，数字中若出现字母形式，不区分大小写。如下所示：

十进制数：以D结尾，汇编语言中缺省值是十进制数，所以D可以省略不写。

二进制数：以B结尾。例如，10100011B，10100011b。

十六进制数：以H结尾。例如，12H，12h，0AB56H，0ab56h。

八进制数：以Q或O（字母）结尾。例如，352Q。





b. 字符串常量

在汇编语言中字符需要用单引号括起来，其值为字符的ASCII值。因为每个字符占用一个字节，所以最好用DB助记符定义字符串。例如，
'A' 的值为41H。'abc' 的值为616263H。





c. 地址表达式

操作数可以是地址符号。若只定义符号的偏移量部分，则使用DW助记符。若要定义它的双字长地址指针（既含16位偏移量又含段基址），则使用DD助记符，其中低字中存放偏移量，高字中存放段基址。

例如，“VAR DW LAB”语句在汇编后VAR中含有LAB的偏移量。





d. ?

在程序中使用“?”为变量预留空间而不赋初值。

e. <n> DUP (操作数,)

若要对某些数据重复多次，可以使用这种格式。其功能是把（ ）中的内容复制n次。
DUP可以嵌套。



例:

M1	DB	15, 67H, 11110000B, ?
M2	DB	'15' , ' AB\$'
M3	DW	4*5
M4	DD	1234H
M5	DB	2 DUP (5, ' A')
M6	DW	M2 ;M2的偏移量
M7	DD	M2 ;M2的偏移量、

段基址

设以上数据自1470:0000开始存放, 则为:

0F 67 F0 00 31 35 41 42 24 14 00 34 12 00 00 05 41
05 41 04 00 04 00 70 14。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



可直接通过变量名引用变量，但要注意类型匹配。例如以下程序片段：

```
MOV AL, M1           ; (AL) = 15
MOV BX, M3           ; (BX) = 20
ADD M3, 6            ; (M3) = 26
MOV AL, M2           ; (AL) = '1' =
```

31H

```
MOV BL, M2+2 ; (BL) = 'A' = 41H
```

```
MOV M1+3, BL ; (M1+3) = 41H
```

M2+2的这种表示形式?



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



二、LABEL伪指令

格式：名字 LABEL 类型

功能：为下一个存储单元起一个名字并定义它的类型。

说明：对于变量，类型可以是BYTE、WORD、DWORD等。对于标号，其类型可以是NEAR、FAR。LABEL伪指令并不为名字分配空间，但它可以为下一个存储单元另起名字而且另定义它的类型，从而使同一地址具有不同类型的名字，便于引用。





例.

```
REDEW          LABEL    WORD
DEBYTE         DB   25H, 32H, 56H, 42H
MOV            BL, DEBYTE      ; (BL) = 25H
MOV            CX, REDEW
               ; (CX) = 3225H
```

以上程序片段执行情况见下图4-2。



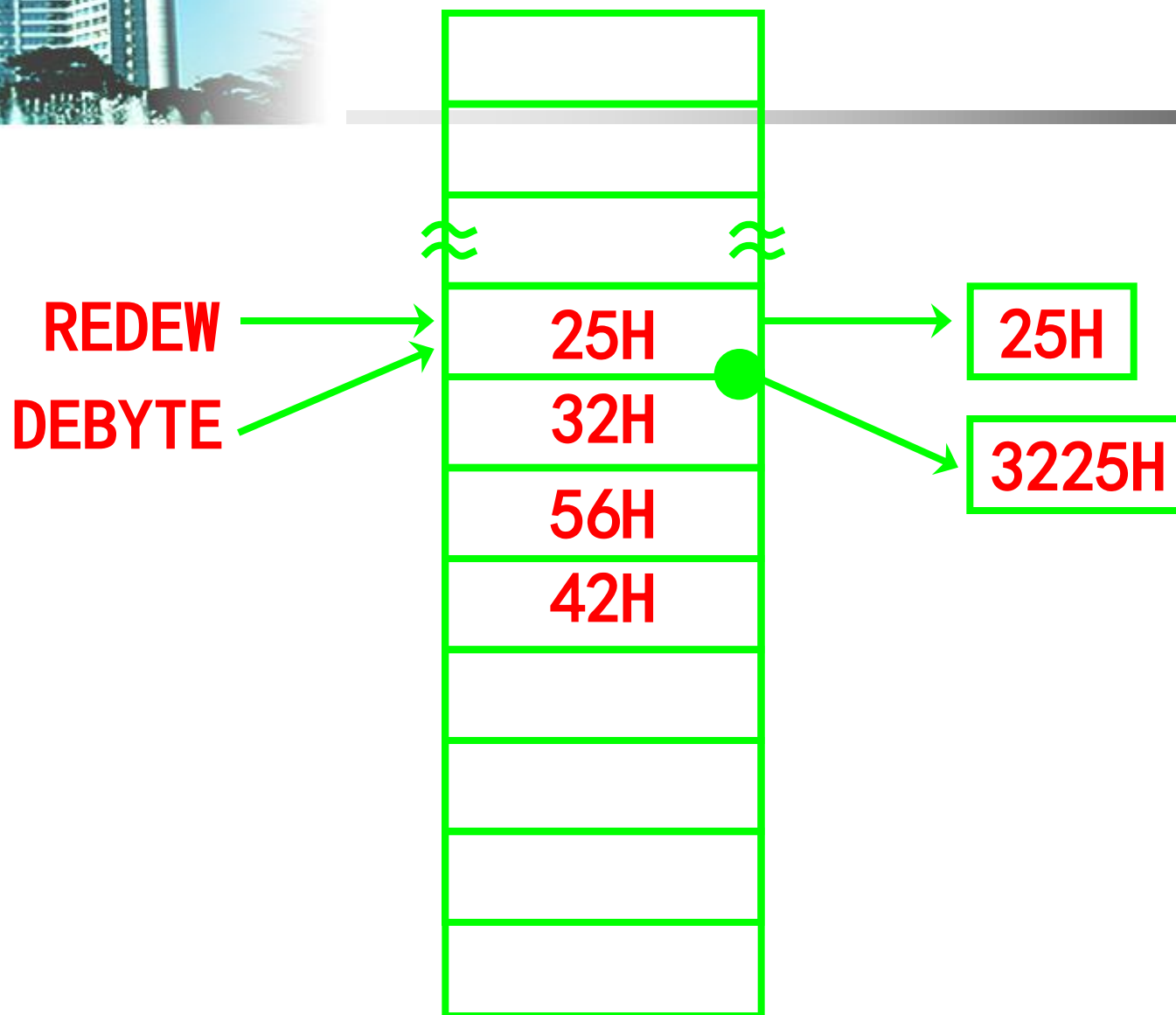


图4—2 LABEL语句功能示意图

[返回](#)



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



三、符号定义伪指令

有时程序中会多次出现同一个表达式，为方便起见可以用符号定义伪指令给该表达式定义一个符号，以便于引用及减少程序修改量，并提高程序的可读性。**汇编后该符号代表一个确定的值。**





1. 等值EQU伪指令

格式：符号名 EQU 表达式

功能：用符号名代表表达式或表达式的值。

说明：表达式可以是任何有效的操作数格式。例如常数、数值表达式、另一符号名或助记符。

注意：用EQU定义的符号在同一个程序中不能再定义。





例:

CR EQU 0DH ;回车符的ASCII值
LF EQU 0AH ;换行符的ASCII值
BEL EQU 07H ;响铃符的ASCII值
PORT_B EQU 61H ;定义PORT_B端口
B EQU [BP+6] ;[BP+6]用B表示

程序中可以通过符号引用这些值, 例如:

MOV AL, CR ;等价于 MOV
AL, 0DH
ADD BL, B ;等价于 ADD
BL, [BP+6])
OUT PORT_B, AL ;输出到61H端口





EQU用途：增加程序可读性、缩短程序书写长度、避免因某些修改而带来的程序不一致性。

EQU伪指令除了以上用途外，经常使用它的一个场合是与\$配合，得到变量分配的字节数。如下所示：

```
MSG      DB      'This is first  
string.'  
  
Count    equ     $-msg  
Mov      cl, count ; (CL)=MSG的串  
长=21
```





这样做可以由汇编程序在汇编过程中**自动计算字符串的长度**，避免了编程者计数。特别是当因字符串改变而串长改变时，取串长的语句无需做任何修改。

由于用EQU定义的符号在同一个程序中不能再定义，所以以下语句是错误的：

```
CT EQU 1
```

```
CT EQU CT+1
```



2. 等号(=) 伪指令

格式: 符号名 = 数值表达式

功能: 用符号名代替数值表达式的值

说明: 等号伪指令与EQU伪指令功能相似, 其区别是等号伪指令的表达式只能是常数或数值表达式。

用“=”定义的符号在同一个程序中可以再定义。通常在程序中用“=”定义常数。

例. DPL1 = 20H

 K = 1

 K = K+1





四、对准伪指令

格式：EVEN

功能：定位到偶地址。

说明：在80X86系列中，由于硬件所采用的设计技术，最好的存储安排是字型数据从字边界开始存放，双字数据从双字边界开始存放，这样可以极大地提高程序的运行效率。

也可以用ALIGN 4定位到双字地址。





例.

ORG 50H

A1 DB 3

EVEN

A2 DW 5

50H

51H

52H

53H

03

05

00

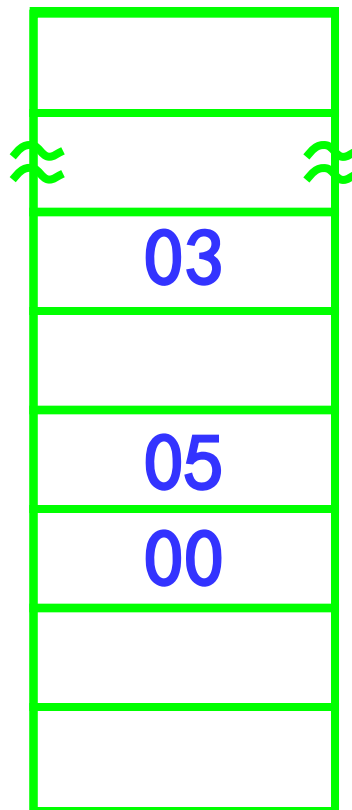


图4-3 EVEN语句的效果





4.4 结构伪指令

结构定义使得多个存储区具有相同的数据格式。例如在日常生活中经常用到的二维表格数据，就可以用结构来定义。





例. 为某单位设计一个工资表，工资表中有三项内容：姓名、职称、工资。该单位共有32个职工。工资表格式和每项数据的长度或范围如下所示。

数据项	姓名	职称	工资
字段名	NAM	POST	WAGES
字段长度、类型或数值范围	10字节 字符	12字节 字符	0~65535 数字

表4-1 工资表格式





1. 结构定义伪指令

格式: 结构名 STRUC

·

·

·

 结构名 ENDS

功能: 定义一种结构。

说明: 结构体内数据定义语句中的变量名叫做结构字段名, 简称字段名。结构一般应放在为结构数据分配空间之前定义, 例如在数据段之前。





例如，对于以上工资表，可以
定义如下结构：

```
PAYROLL          STRUC
NAM              DB    10
DUP(' A')
POST            DB
'ENGINEER '
WAGES          DW    ?
PAYROLL        ENDS
```





在这个结构中，NAM和POST字段是字节型数据，WAGES是字型数据。POST和WAGES是简单字段，**简单字段**是指只有一个数据项的字段（一个？、一个数字、一个字符或一个字符串）。NAM是多重字段，**多重字段**是指含有DUP操作数或具有多个数据项的字段。

语句“ADR DB 2, 5, 6”因为有三个数据项，所以也是多重字段。





注意：结构定义伪指令中虽然包含了数据定义语句，但仅仅是定义了某种格式的结构（例如PAYROLL），**并没有给它分配空间**。为了给结构数据分配空间，需要用结构预置语句。





2. 结构预置语句

格式：结构变量名 结构名 <字段值表>

功能：为结构变量分配存储空间及初始化。

说明：结构预置语句中的结构变量名是程序中可以直接引用的名字。

结构名是结构定义中的结构名，它告诉汇编程序该变量具有什么样的结构。

字段值表用于给结构字段赋初值。





对字段值的说明:

① 字段值表必须用<>括起来;

② 若给出字段值表, 则字段值间要用逗号隔开。

③ 只能对简单字段赋初值, 对于多重字段的位置应直接用逗号隔开, 相当于占位, 不可省略。

④ DUP操作数可以出现在结构预置中, 用以分配多个同样结构的数据。





为上例工资表分配32条记录空间的程序段如下：

N01 PAYROLL <>

;1号职工的记录，各字段采用结构定义初值

N02 PAYROLL <,'WORKER',1200>

;为2号职工的记录初始化

N03_32 PAYROLL 30 DUP(<>)

;DUP为3~32号职工定义30条相同结构的记录





经过定义，工资表的初始存储分配情况如下所示：

偏移量 结构变量名

N01 N01
N02 N02
N03_32 N03_32

N03_32+24*1
...

N03_32+24*29

NAM	POST	WAGES
AAAAAAAAAA	ENGINEER □ □ □ □	
AAAAAAAAAA	WORKER □ □ □ □ □ □	1200
AAAAAAAAAA	ENGINEER □ □ □ □	
AAAAAAAAAA	ENGINEER □ □ □ □	
...	...	
AAAAAAAAAA	ENGINEER □ □ □ □	
注：□表示空格，占用一个字符位置。		

表4-2 工资表初值





3. 访问结构变量

对结构分配空间后，在程序中就可以访问它们了。存取格式可以有以下几种：

(1) 结构变量名

这种格式可以访问到结构变量级。它访问的是结构变量中的**第一个字节值**（视结构中的第一个字段的类型决定，例如若是用DW定义，则为第一个字）。

例. LEA BX, N01

;把结构变量N01的首地址送BX





(2) 结构变量. 字段名

这种格式可以访问到结构变量中的字段值。

例如：

```
MOV          NO1. WAGES , 1200
```

；为1号职工的工资字段赋值1200

```
MOV          AX , NO1. WAGES
```

；取1号职工的工资送给AX





从以上可以看出，这里引用的结构变量**实际上是使用它的偏移量**，偏移量可以通过基址寄存器BX或BP表示。若使用BX，则段寄存器默认为DS；若使用BP，则段寄存器默认为SS。允许使用段超越前缀。

例如：

```
LEA          BX , N03_32
```

```
MOV          [BX]. WAGES, 1500
```

；为3号职工的工资字段赋值1500



(3) 结构变量. 结构字段[变址寄存器]

这种格式可以访问到结构变量中某字段的某个字节，该字节距离字段首的位移由变址寄存器SI或DI给出。例如：

```
LEA          BX , N03_32
MOV          [BX]. WAGES , 1500
XOR          SI , SI
MOV          [BX]. NAM[SI] , ' C'
```

；字符C送给3号职工姓名字段的一个字节



注意结构定义、预置和存取三者的顺序及位置一般应遵从：**结构定义在数据段前出现；预置在数据段中出现；访问在代码段中出现。**





运行以上程序4.5后，工资表的数据如下所示：

偏移量 结构变量名

N01 N01
N02 N02
N03_32 N03_32

N03_32+24*1
...

N03_32+24*29

NAM	POST	WAGES
AAAAAAAAAA	ENGINEER □ □ □ □	1200
AAAAAAAAAA	WORKER □ □ □ □ □ □	1200
CAAAAAAAAA	ENGINEER □ □ □ □	1500
AAAAAAAAAA	ENGINEER □ □ □ □	
...	...	
AAAAAAAAAA	ENGINEER □ □ □ □	
注：□表示空格，占用一个字符位置。		

表4-3 程序4.5运行后的工资表





编写一个循环程序并递增修改SI的值，
可以对一个字段的各个字节实现存取。同样
道理，通过每次给BX增加一条记录的长度（
本例是加24），可以存取到不同的记录。





4.5 汇编语言操作符

操作符可以出现在语句的操作数表达式中。该操作在汇编程序汇编时实现。包括算术、逻辑、关系、属性、返回值操作符。





一、算术操作符

算术操作符包括 $+$ 、 $-$ 、 $*$ 、 $/$ 和
MOD（取模）操作符。

算术操作符可以用在数值表达式或
地址表达式中。





例.

X DW 12, 34, 56

CT EQU (\$-X)/2

MOV CX, CT ; (CX

) = 3

MOV AX, X

ADD AX, X+2 ; (AX) =

46





二、逻辑操作符

逻辑操作符包括AND、OR、XOR和NOT。逻辑操作符是按位操作的，它只能用在数值表达式中。

例.

```
PORT EQU 0FH
```

```
AND DL, PORT AND 0FEH
```

汇编后: AND DL, 0EH



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



三、关系操作符

关系操作符包括EQ、NE、LT、LE、GT、GE。其操作结果为一个逻辑值，若关系成立结果为真（全1），否则结果为假（0）。其中的操作数必须是数字或同段内的两个存储器地址。

例. 指令 MOV AL, CH LT 20的汇编结果：

MOV AL, 0FFH ; 当CH < 20时
或: MOV AL, 0 ; 当CH ≥ 20时





四、返回值操作符

这组操作符可以把一些特征或内存单元地址的组成部分作为数值返回。它们是TYPE、LENGTH、SIZE、SEG、OFFSET和\$，其格式和功能见表4-4。



格 式	功 能
TYPE 变量 TYPE 标号	返回变量或者标号的类型代码 ^[1]
LENGTH 变量	返回由<n> dup (操作数, ...) 格式定义的n值, 其它情况回送1
SIZE 变量	返回分配给变量的字节数, 只对由<n> dup (操作数, ...) 格式定义的变量有实际意义 ^[2]
SEG 变量 SEG 标号	返回变量或者标号的段基址
OFFSET 变量 OFFSET 标号	返回变量或者标号的偏移量
\$	返回地址计数器的当前值
<p>注[1]: 若使用TYPE 变量, 则返回一个数据项占用的字节数。例若变量用DB定义, 则返回1; 用DW定义返回2.....; 若用结构定义则返回该结构占用的字节数。若使用TYPE 标号, 则NEAR型标号返回-1, FAR型标号返回-2。</p> <p>注[2]: $SIZE = TYPE * LENGTH$, 只有在变量用<n> dup (操作数, ...)时表示该变量占用的字节数。因为LENGTH在其它情况下总是回送1, 此时SIZE也就失去实际意义。</p>	

表4-4

返回值操作符





例.

```
VAR      DW      50      DUP (12)
ARY      DW      4, 5, 6
COUNT EQU $-OFFSET ARY    ; COUNT=6
MOV      AX, SEG VAR
MOV      DS, AX                ; VAR的
段基址→DS
ADD      SI, TYPE ARY          ; (SI) +
2→SI
MOV      CX, LENGTH VAR        ; 50→CX
MOV      CX, SIZE VAR
; 100→CX
MOV      CX, LENGTH ARY        ; 1→CX
MOV      CX, SIZE ARY          ; 2→CX
```





五、属性操作符

属性操作符指定操作数的属性。包括PTR、THIS、SHORT、HIGH和LOW操作符。





1. PTR操作符

格式：类型 PTR 地址表达式

功能：指定地址表达式的类型。

说明：若是变量的地址表达式, 则类型可以是BYTE、WORD、DWORD等。若是标号的地址表达式, 则类型可以是FAR、NEAR。PTR经常用在临时改变地址类型或必须明确指出类型的场合。





例.

BUF DB 31H, 32H

MOV AL, BUF ; (AL) = 31H

MOV BX, WORD PTR BUF
; 临时指定BUF为字型, (BX) = 3231H

LEA DI, BUF

MOV BYTE PTR [DI], 6

; 明确指出DI指向字节型单元, 否则汇编时出错



2. THIS操作符

格式：THIS 类型

功能：为存储器操作数指定类型。

该操作数地址与下一个存储单元具有相同的段基址和偏移量。

说明：它并不为该存储器操作数分配空间。与LABEL伪指令的功能有相似之处,但其区别是所出现的位置不同。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



例.

```
REDEW EQU THIS WORD
```

```
DEBYTE DB 25H, 32H, 56H, 42H
```

REDEW和DEBYTE的段基址和偏移量均相同, 但类型不同前者是字型, 后者是字节型。

例.

```
F_JMP EQU THIS FAR
```

```
N_JMP: DEC CX
```

使“DEC CX”指令具有一个FAR属性的地址F_JMP

。





3. SHORT操作符

用来修饰JMP指令中转向地址的属性, 指出转向地址是在下一条指令地址的 $-128 \sim +127$ 字节范围之内。

例如: **JMP SHORT LAB**





4. HIGH、LOW操作符

这两个操作符被称为字节分离操作符,它接收一个数字或地址表达式, HIGH取其高字节, LOW取其低字节。

例.

```
COUNT EQU 1234H
```

```
MOV AL, LOW COUNT
```

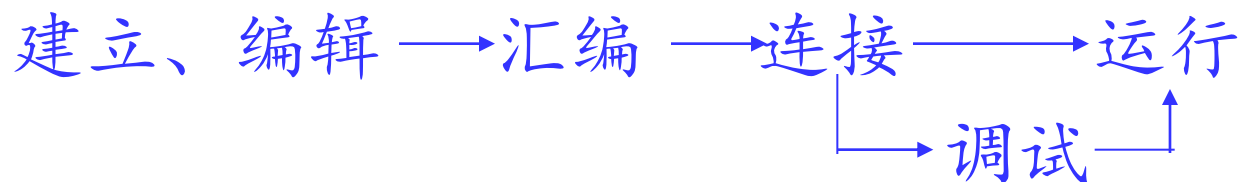
汇编结果为: MOV AL, 34H



4.6 汇编语言程序上机过程

汇编语言实践环境基于DOS平台。

当汇编语言源程序编好后，要使其实现功能，需经过以下过程：



具体实现过程（略）





4.7 数据的输入输出

在汇编语言程序中，系统软件资源可以通过**ROM BIOS中断调用**和**DOS系统功能调用**中断来使用。

ROM BIOS和DOS中断的层次关系图见下图。



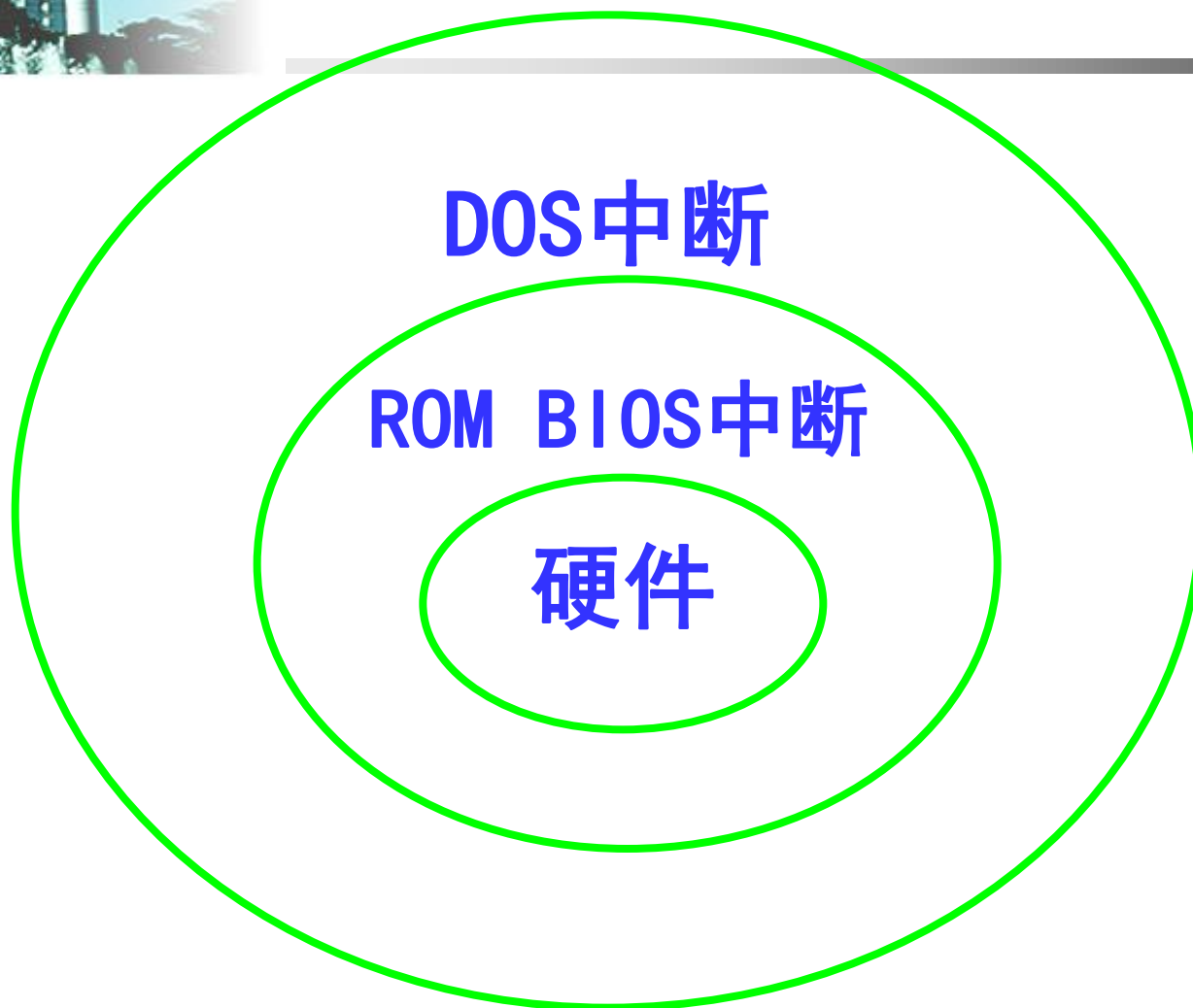


图4-4 ROM BIOS和DOS中断层次图





一、使用ROM BIOS中断调用

BIOS——基本输入输出系统。

主要由许多设备驱动程序组成，在PC系统中这组程序被固化在ROM中，所以常称为**ROM BIOS**。本节只介绍键盘输入和显示器输出的BIOS中断调用方法。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



1. 键盘中断调用 16H

16H中断服务子程序提供了多个功能, 每个功能对应一个功能号。调用它们的方法是:

- ① 功能号→AH
- ② INT 16H



主要功能简述如下:

功能号	功 能	返 回 参 数
0	等待从键盘读字符	AL=字符的ASCII值, AH=扫描码
1	读键盘缓冲区字符	若ZF=1, 表示缓冲区空;否则表示缓冲区不空, 则AL=ASCII值, AH=扫描码
2	返回键盘状态字节	AL=键盘状态字节





说明：对于2号功能，AL中返回的键盘状态字节各位含义如下：

位7 6 5 4 3 2 1 0



1 = 按下右Shift键

1 = 按下左Shift键

1 = 按下Ctrl键

1 = 按下Alt键

1 = Scroll Lock 状态改变

1 = Num Lock 状态改变

1 = Caps Lock 状态改变

1 = Insert 状态改变



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



例. 从键盘接收一个字符，并送入CHAR变量。

CHAR DB ?

MOV AH, 0

INT 16H ;等待从键盘接收一个

字符

MOV CHAR, AL;接收到的字符保存





也可以用1号功能完成上述功能，如下所示：

CHAR DB ?

LOP: MOV AH, 1

INT 16H ;读键盘缓冲区

JZ LOP

;若缓冲区无字符可取则循环

等待

MOV CHAR, AL

;接收到的字符保存





例. 编写能够实现以下要求的程序段: 若按了 Insert 键, 则转入插入处理, 否则继续处于常规状态。

```
MOV    AH, 2
INT     16H
TEST   AL, 80H
JNZ    INSERT
      ...
```

INSERT: ... ; 插入处理





2. 显示器中断调用10H

ROM BIOS的10H中断子程序提供了多种功能以支持屏幕处理及显示。

以普遍使用的80×25彩色文本方式为例说明。





80×25显示方式坐标:

屏幕左上角位置是(0, 0),

屏幕右上角位置是(0, 79),

屏幕左下角位置是(24, 0),

屏幕右下角位置是(24, 79)。

屏幕上的每个字符在显示RAM中有两个存储单元相对应, 一个存放该字符的ASCII值, 另一个存放其属性, 所以一屏共需4000个单元。这就是显示器一页的内容, 实际上分配给一页是4K字节。都支持0~7页。





字符的属性确定了每个要显示字符的特性，例如字符是否闪烁、彩色字符的颜色等。见下图。

BL为**闪烁**位， $BL=1$ 为字符闪烁。

I为**亮度**位， $I=1$ 为高亮度。

亮度和闪烁只用于前景。表4-5给出了16种字符颜色的组合，它也适用于图形方式。当前景和背景选择相同颜色组合时，字符便无法看见。



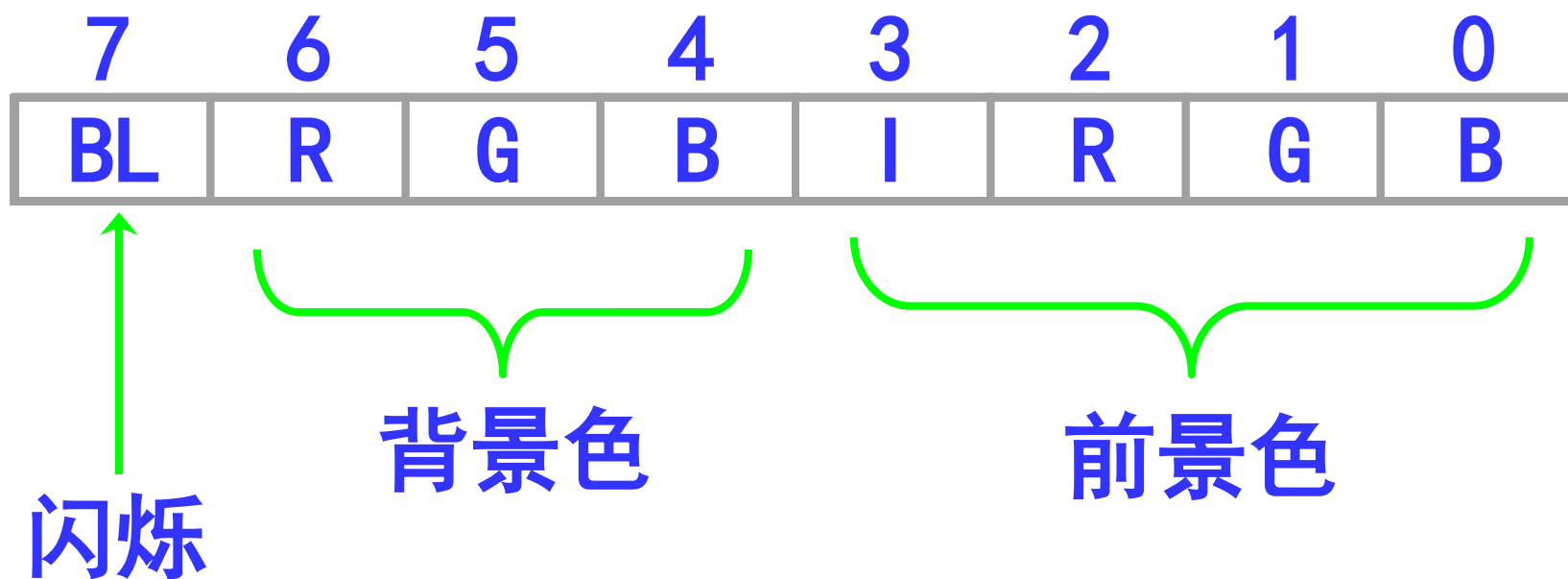


图4-5 彩色文本属性字节





二、使用DOS系统功能调用

每种操作系统都为用户提供
了使用输入输出设备的系统资源。这些资源
不同于ROM BIOS，它们对硬件的依赖性
更少，兼容性较好、使用更方便。





在DOS环境下, 汇编语言除了通过调用ROM BIOS中断使用输入输出设备外, 还可以通过DOS系统功能调用中断 (INT 21H) 使用它们。

采用的系统功能调用方法: 功能号送AH寄存器, 调用参数送所要求的位置, 然后发INT 21H系统功能调用中断。





1. 带回显的字符输入并识别Ctrl-C键

功能号：1

返回参数：AL=输入字符的
ASCII值

说明：等待从标准输入设备(通常为键盘)输入一个字符，把接收到的字符的ASCII值送给AL，并显示到显示器的当前光标位置。该功能识别Ctrl-C键并做相应处理。





例. 输入一个字符到CHAR变量中。

```
CHAR DB ?
```

```
MOV AH, 1
```

```
INT 21H
```

```
MOV CHAR, AL
```





从此例中可以看出使用DOS系统功能调用比使用BIOS中断更方便, 因为它的功能更强。1号功能在输入的同时还显示到了屏幕上, 并且适当处理光标。另外它还能够识别Ctrl- C键。但使用该功能得不到字符的扫描码。

能够输入一个字符的DOS功能调用还有6、7、8号, 它们之间在功能上有一些差别 (见附录A)。





2. 输出一个字符

功能号：2

调用参数：DL=字符的ASCII值

说明：该输出功能使光标跟随移动。

例. 输出一个字符Y。

```
MOV    AH , 2
```

```
MOV    DL , ' Y'
```

```
INT    21H
```



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



3. 输入字符串

功能号：0AH

调用参数：DS:DX指向自定义的
输入缓冲区首地址

功能：从标准输入设备（例如键盘）输入一串字符到用户定义的缓冲区，直到按下Enter键为止。在接收的同时显示到屏幕上，并且识别Ctrl-C键，也处理DOS编辑键。





说明：自定义的输入缓冲区应是字节型，其格式如表4-7所示。**第一个字节**是用户自定义的缓冲区最大长度，以字节计数。**第二个字节**是实际输入的字符个数，这个字节由系统自动计数并回填，计数值不包括最后一个回车键，用户程序只需要为该字节预留空间即可。**从第三个字节开始存放输入的字符串**，每个字符占用一个字节，内容为该字符的ASCII值，输入以回车键结束，输入字符的个数可以少于缓冲区长度。



地址

内容

说明

0

缓冲区长度M

$M \leq 255$

1

实际输入的字符个数

系统自动计数并回填
不包括最后一个回车键

2

输入的第一个字符

3

输入的第二个字符

4

输入的第三个字符

...

. . .

含回车键在内的
M个字节缓冲区

M+1

表4-7 输入缓冲区格式

返回





例. 定义60个字符的缓冲区，并输入一串字符。

BUFFER **DB 60** ; 定义缓冲区长

度

DB ?

DB 60 DUP (?)

; 设DS已是BUFFER的段基址

MOV AH, 0AH ; 接收一串字符

LEA DX, BUFFER

INT 21H





当执行INT 21H指令时，便等待从键盘输入字符直到按下回车键结束。

在输入后经常需要把实际输入的字符个数送给CX寄存器，以便后续处理。这可以用以下程序片段实现。注意不能把第二个字节的内容直接送给CX。

MOV CL, BUFFER+1 ;实际输入字符数送CX

XOR CH, CH

把输入的第一个字符送给AL寄存器

的方法是：

MOV AL, BUFFER+2



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



4. 输出字符串

功能号：9

调用参数：DS:DX指向要输出的以\$结尾的字符串首地址

功能：输出字符串到标准输出设备

说明：要输出的字符串必须以\$结束





例.

PRINT

DB ' To input: ', '\$'

;以\$结尾的要输出的字符串

;设DS已指向PRINT的段基址

MOV AH, 9

LEA DX, PRINT

INT 21H

;输出字符串“To input: ”到显示器





综合举例：编程实现输出“To
input:”的提示信息,并等待从键盘输入一串字符
送给BUFFER缓冲区,见[程序4.7](#)。

使用DOS功能调用比使用ROM BIOS中断调用更简单方便,但INT 21H却不能完全覆盖BIOS提供的功能,如它无法确定屏幕的光标位置,而INT 10H中断则不仅可以做到这一点,还可以更全面地控制屏幕,并且运行速度也要快得多。





缓冲区

字符串

信息

STACKSG SEGMENT STACK 'S'

DW 64 DUP('ST')

STACKSG ENDS

DATA SEGMENT

BUFFER DB 60,?,60 DUP(?) ;输入

PRINT DB 'To input: ','\$' ;输出

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:STACKSG

MAIN PROC FAR

MOV AX,DATA

MOV DS,AX

MOV AH,9 ;输出一串提示

LEA DX,PRINT

INT 21H

MOV AH,0AH ;接收一串字符

LEA DX,BUFFER

INT 21H

MOV AX,4C00H

INT 21H

MAIN ENDP

CODE ENDS

END MAIN



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

返回



4.8 微处理器伪指令(不要求)

汇编程序在缺省情况下只接受8086 / 8088的指令系统,即使在386以上机器也是如此,因此为8086 / 8088编写的程序在286以上的处理器都可以顺利执行。

为了能够使用其它微处理器或协处理器的指令系统编写高级的32位机软件,需要在程序中增加选择微处理器的伪指令。



较常使用的选择微处理器伪指令有以下几种：

伪指令	功 能
. 286	选择80286微处理器指令系统
. 386	选择80386微处理器指令系统
. 486	选择80486微处理器指令系统
. 586	选择80586微处理器指令系统
. 8087	选择8087数字协处理器指令系统
. 287	选择80287数字协处理器指令系统
. 387	选择80387数字协处理器指令系统





注意**较低版本**的汇编程序并不完全支持这一组伪指令。

例如即使在Pentium上使用MASM5.0，也不支持**.586**伪指令。由于向下兼容，所以在Pentium机器上仍然可以使用**.386**伪指令。





4.9 简化段定义

简化段定义是MASM 5.0版以后提供的，它较容易使用，和高级语言连接也比较容易，但这种格式并不适用于大多数汇编程序。

本节简介简化段定义常用结构及与其有关的伪指令。





一、简化段定义常用结构

注意：当 . 386 等选择处理器伪指令出现在 . model 之前时表示缺省选择 32 位指令模式，出现在 . model 之后时表示缺省选择 16 位指令模式。

下例给出简化段定义常用结构。





二、简化段定义常用结构中的伪指令

1. 定义存储模型伪指令

常用格式：`.model` 存储模型

功能：定义存储模型。





常用的存储模型有：

- ①TINY：所有代码和数据放置在一个段中。
- ②SMALL：所有代码在一个段内，所有数据在另一个段。
- ③MEDIUM：代码放置在多个段内。数据限制在一个段。
- ④COMPACT：代码在一个段内，数据可以在多个段内。
- ⑤LARGE：代码和数据被放置在多个段内。
- ⑥HUGE：单个数据项可以超过64K，其它同LARGE模型。
- ⑦FLAT：与TINY模型类似，所有代码和数据放置在一个段中。TINY模型的段是16位，FLAT32位。





2. 定义堆栈段尺寸伪指令

格式: `.stack size`

功能: 建立一个堆栈段并定义其大小

说明: 若不指定 `size` 参数, 则使用缺省值 1KB。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

三、有关的预定义符号

与简化段定义伪指令相关的预定义符号：

@CODE： 代码段段名。

@DATA： 由 .DATA 和 .STACK 定义的段集合成的组名。

@FARDATA： 独立数据段的段名。

这些预定义符号类似于用 EQU 伪指令定义的符号，它们可以在程序中被引用。

例如：

```
MOV    AX, @DATA
```

```
MOV    DS, AX
```





四、简化段定义举例

.MODEL SMALL

.586

.STACK 64

.DATA

HI DB 'HELLO!', 0DH, 0AH, '\$'

.CODE

.STARTUP

MOV AH, 9

LEA DX, HI

INT 21H

.EXIT

END



与完整段定义相比，`.startup`和`.exit`伪指令的引入方便程序设计人员。

注意：当`.386`等选择处理器伪指令出现在`.model`伪指令之前时**不能用**`.startup`和`.exit`，否则汇编时出错。



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



4.8 Windows 程序的执行环境

➤ Windows 内存布局

虚拟内存

“平坦”内存模式

内存布局

段选择符和段描述符信息示例

➤ Windows 的保护机制

特权级+类型检查

➤ 用户界面



4.9 Windows汇编源程序基本格式

- 控制台界面的汇编源程序
- Windows界面的汇编源程序





.386

.model flat, stdcall

option casemap:none

includelib msvcrt.lib

printf PROTO C :ptr sbyte,:VARARG

.data

szMsg byte “Hello World! %c” ,0ah,0

a byte 'Y' b byte "hello"

.code

start:

 invoke printf,offset szMsg,a

 invoke printf,offset szMsg,offset b

ret

end start





BUF1 DB 1,2,3

BUF2 DW 5,6,7

L1 EQU \$-BUF2

L2 EQU BUF2-BUF1

L3 EQU \$-BUF1



感谢关注聆听！



张华平

Email: kevinzhang@bit.edu.cn

微博: @ICTCLAS张华平博士

实验室官网:

<http://www.nlpir.org>



大数据千人会

