第3-2章 伪指令和程序格式

主讲：许万茹

本章教学内容

- 伪指令的概念、定义和使用方法；
- 等值语句、数据定义语句、段定义语句、其它语句、地址计数器语句、过程定义语句等伪指令的格式和含义。

重点：各种伪指令的格式、含义和它们在程序设计中的作用

主要内容

3.2.1 伪指令的功能与格式

3.2.2 主要的伪指令类型

3.2.3 汇编语言程序格式

3.2.4 汇编语言程序的上机过程

1、汇编语言语句类型

语句类型

- **指令（性）语句**

  能够被直接翻译成机器代码并让CPU直接执行的语句

- **指示性语句（伪指令语句）**

  指导汇编程序如何编译汇编语言源程序，并不翻译成机器代码，CPU也不执行

1、汇编语言语句类型

汇编语句分为指令和伪指令。

指令

每一条指令语句对应一条可执行的CPU机器指令，其执行产生相应的CPU动作

- 指令性语句 – 可以产生相应的机器码:

MOV AX,1234H

对应的机器码：B8 34 12

- 有操作码和操作数2部分

伪指令

不对应CPU机器指令，是汇编程序对源程序进行汇编时处理的操作，完成存储模式定义、数据定义、存储器分配、指示程序开始结束等功能；不产生机器码。

- 指示性语句（伪操作）：

DATA1 DB 60H,0ACH,74H

例：C语言编程格式

用C语言编程实现 c = a + b，并在屏幕上显示出结果

```
#include "stdafx.h"
#include "stdio.h"
int main  (int argc, char* argv[])
{
    int a=1,b=2,c;

    c=a+b;
    printf("c=%d\n",c);
    return 0;
}
```

指示性语句

指令性语句

例：用汇编程序完成两个字节数据相加

```
DATA   SEGMENT          ；段定义开始（DATA段）
BUF1   DB  34H          ；第1个加数
BUF2   DB  2AH          ；第2个加数
SUM    DB  ?            ；准备用来存放和数的单元
DATA   ENDS             ；段定义结束（DATA段）
CODE   SEGMENT          ；段定义开始（CODE段）
       ASSUME CS:CODE，DS:DATA
                        ；规定DATA、CODE分别为数据段和代码段
START: MOV  AX，DATA
       MOV  DS，AX      ；给数据段寄存器DS赋值
       MOV  AL，BUF1    ；取第1个加数
       ADD  AL，BUF2    ；和第2个加数相加
       MOV  SUM，AL     ；存放结果
       MOV  AH，4CH
       INT  21H         ；返回DOS状态
CODE   ENDS             ；段定义结束（CODE段）
       END START        ；整个源程序结束
```

伪指令语句

指令语句

2. 汇编语言语句格式

- 指令语句(处理器指令、硬指令)

[名字:] 操作码 [操作数[,操作数]][;注释]

- 伪指令语句(汇编程序命令、伪指令)

[名字]伪操作码 [操作数[,操作数]][;注释]

含义：由用户按一定规则定义的标识符
组成：英文字母、数字、特殊符号
形式：标号和变量

语句的说明部分

2. 汇编语言语句格式

- 指令语句(处理器指令、硬指令)

名字定义满足的规则

(1) 数字不能作为第一个字符

(2) 单独的问号（?）不能作为名字

(3) 最大有效长度为31

(4) 保留字不能作为名字使用

[名字]伪操作码 [操作数[,操作数]][;注释]

含义：由用户按一定规则定义的标识符

组成：英文字母、数字、特殊符号

形式：标号和变量

语句的说明部分

操作码

含义：指明操作的性质或功能。

书写规则：**操作码与操作数之间用空格分开**

操作数

含义：指定参与操作的数据或者地址表达式。

个数：一般指令，1个或2个，也可以没有；
伪指令和宏指令，可以有多个。

书写规则：**操作数多于1个时，操作数之间用逗号分开**

3. 伪指令的功能

- **程序员提高编程效率**
  - 数据定义及存储器分配伪操作
  - 表达式赋值伪操作
  - 地址计数器与对准伪操作
- **指导编译器如何编译汇编程序**
  - 选择处理器型号
  - 定义存储模式
  - 定义段
  - 指示程序开始/结束

主要内容

3.2.1 伪指令的功能与格式

3.2.2 主要的伪指令类型

3.2.3 汇编语言程序格式

3.2.4 汇编语言程序的上机过程

1. 段定义伪操作

- 程序由4个逻辑段组成：数据段、堆栈段、附加段和代码段
- 每个逻辑段以SEGMENT语句开始，以ENDS语句结束

代码段

包括许多以符号表示的指令，源程序-其内容就是程序要执行的指令。

数据段

用来在内存中建立一个存储工作区，以存放常数、变量等操作数据。

堆栈段

用来在内存中建立一个堆栈存储区，以便在中断、子程序调用时使用。

1. 段定义伪操作

完整的段定义格式

```
DATA    SEGMENT                 ;  定义数据段
        …
DATA    ENDS
;-----------------------------------------
EXTRA   SEGMENT                 ;  定义附加段
        …
EXTRA   ENDS
;-----------------------------------------
CODE    SEGMENT                 ;  定义代码段
        ASSUME CS:CODE, DS:DATA, ES:EXTRA
START:
        MOV    AX, DATA
        MOV    DS, AX           ;  段地址 → 段寄存器
        …
CODE    ENDS
        END    START
```

段定义格式

段名 SEGMENT [定位类型] [组合类型] [使用类型] [‘类别’]

……

…… ；语句序列

段名 ENDS

一般采取默认设置，缺省：

段名 SEGMENT

……

…… ；语句序列

段名 ENDS

2. 指定段地址伪指令

ASSUME <段寄存器名>:<段名>[,<段寄存器名>:<段名>…]

功能：建立段寄存器与段的缺省关系

注意：ASSUME伪指令并不为段寄存器设定初值

例： assume cs:code, ds:data, es:extra

其中：code是代码段的段名，data是数据段的段名，extra是扩展段的段名

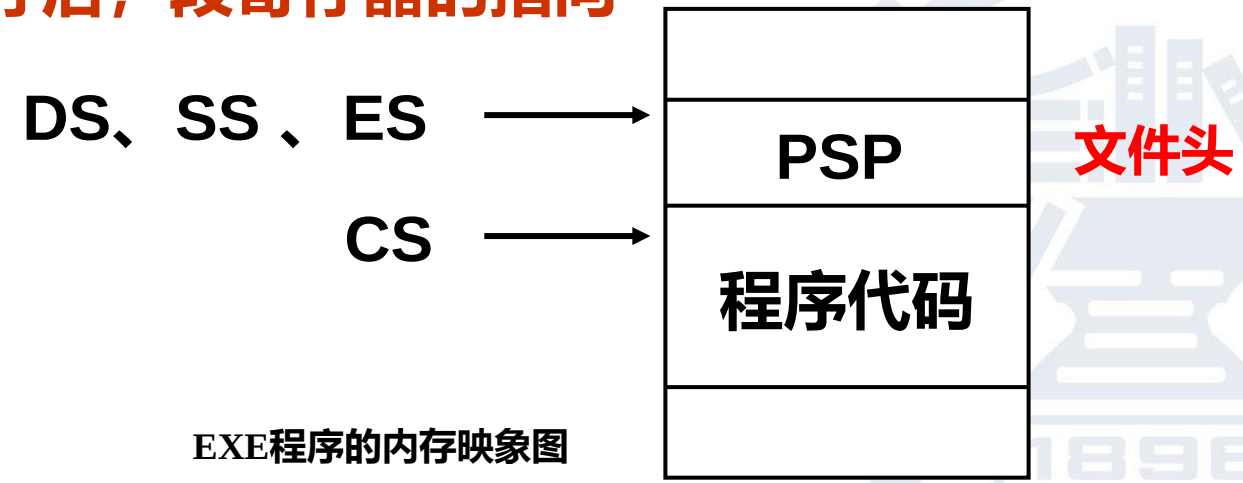
3. 设置段地址值

- 在代码段开始处进行DS、SS、ES的段地址装填

例：MOV AX, DATA

MOV DS, AX

程序装入内存后，段寄存器的指向



EXE程序的内存映象图

4. 结束伪操作

END [label]

例：

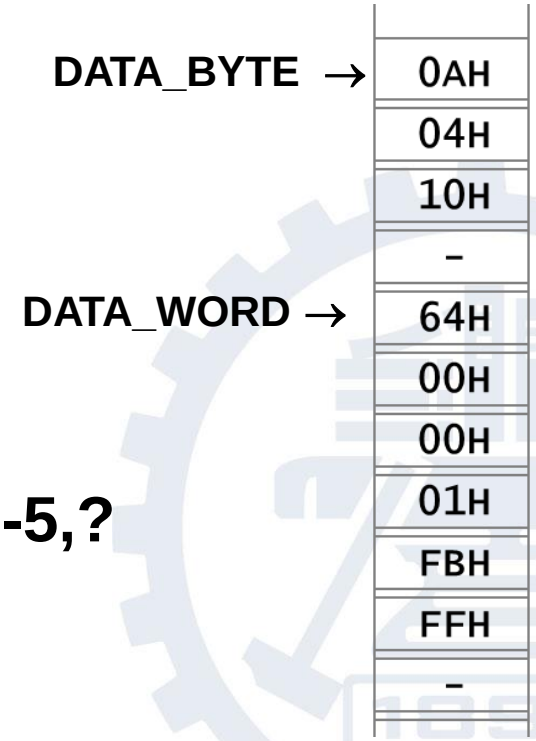
```
code   segment          ; 定义代码段
       assume cs:code, ds:data, es:extra
start:
       mov   ax, data
       mov   ds, ax      ; 段地址 → 段寄存器
       …
code   ends
       end   start
```

5. 数据定义及存储器分配伪操作

[变量] 助记符 操作数[,操作数,…][;注释]

助记符：DB DW DD DQ (8字节) DT (10字节)

DB： 定义字节
DW：定义字
DD： 定义双字



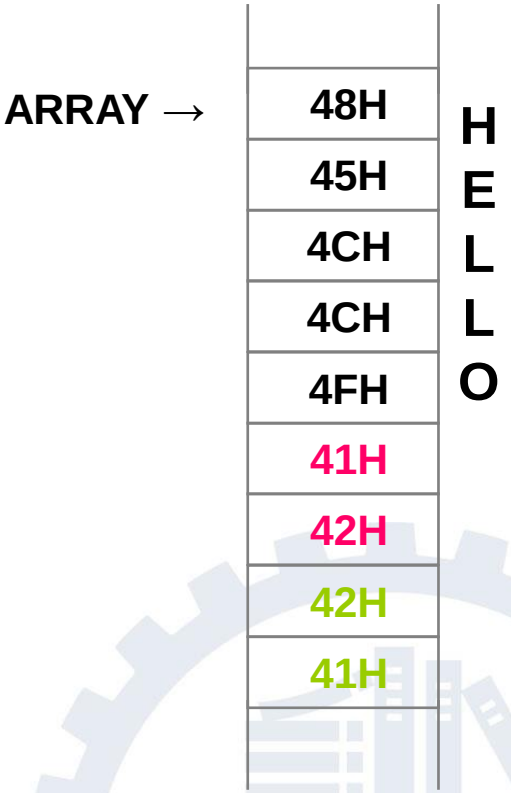
例：

```
DATA_BYTE  DB  10,4,10H,?
DATA_WORD  DW  100,100H,-5,?
```

例： ARRAY DB ‘HELLO’

DB ‘AB’

DW ‘AB’



例： 操作数用复制操作符DUP，
表示操作数重复若干次

VAR DB 100 DUP (?)

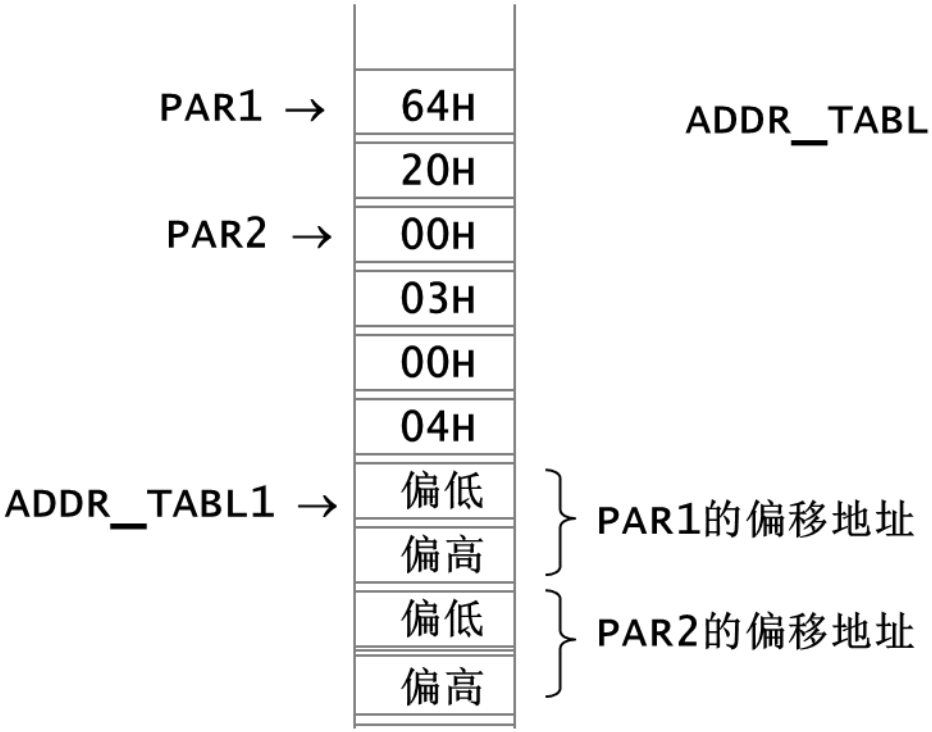
DB 2 DUP (0,2 DUP(1,2),3)

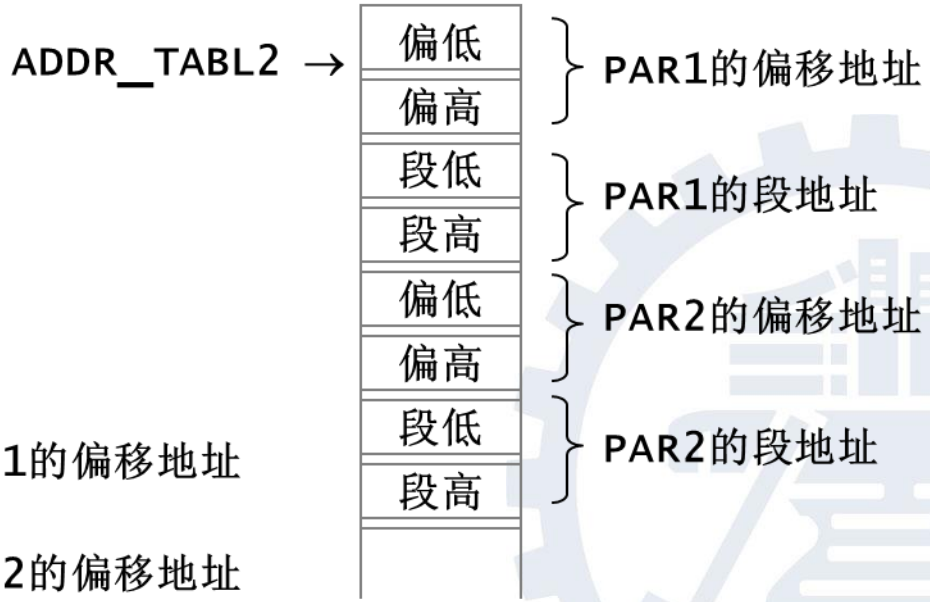
例： PAR1 DB 100,20H

PAR2 DW 300H,400H

ADDR_TABL1 DW PAR1,PAR2；存放偏移地址16位

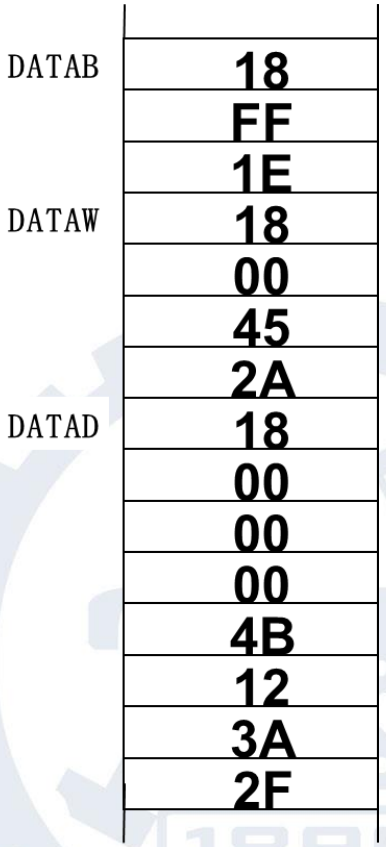
ADDR_TABL2 DD PAR1,PAR2；存放偏移和段地址各16位





操作数

- 常数
  - 字节数据 例：DATAB DB 18H，-1，30
  - 字数据 例：DATAW DW 18H，2A45H
  - 双字数据 例：DATAD DD 18H，2F3A124BH
- 代表数据



操作数

- 常数
  - 字节数据　例：DATAB DB 18H，-1，30
  - 字数据　例：DATAW DW 18H，2A45H
  - 双字数据　例：DATAD DD 18H，2F3A124BH
- 代表数据
- 表达式
- 代表内存单元地址

```
例：  ADDR1 DW NEXT   ；存放偏移地址
      ADDR2 DD NEXT   ；存放偏移和段地址
        …
NEXT：MOV AL，34H   ；NEXT为指令标号
```

操作数

- 常数
  - 字节数据　例：DATAB DB 18H，-1，30
  - 字数据　例：DATAW DW 18H，2A45H
  - 双字数据　例：DATAD DD 18H，2F3A124BH
- 代表数据
  - 表达式　代表内存单
  - 字符串　存放各字符的ASCII码
  - ?　只分配单元，不定义初值

例：
```
ADDR1 DW NEXT   ；存放偏移地址
ADDR2 DD NEXT   ；存放偏移和段地址
...
NEXT：MOV AL，34H   ；NEXT为指令标号
```

例：
```
STR1 DB ‘ABCD’
STR1 DB ‘A’, ‘B’, ‘C’, ‘D’
STR1 DB 41H,42H,43H,44H
```

例：
```
BUF1 DB 5,6,7,?
BUF2 DW 56H,78H,?,345FH
```

例：
```
OPER1  DB  ?, ?
OPER2  DW  ?, ?
       ……
MOV  OPER1, 0   ;字节指令
MOV  OPER2, 0   ;字指令
```

例：
```
OPER1  DB  1H, 2H,  3H
OPER2  DW  1234H, 5678H
       ……
MOV  AX, OPER1+1  ×
MOV  AL, OPER2    ×  类型不匹配

MOV  AX, WORD PTR  OPER1+1
MOV  AL, BYTE PTR  OPER2
```

6. 表达式赋值伪操作

(1) 表达式名 EQU 表达式

例：
```
ALPHA   EQU  9
BETA    EQU  ALPHA+18
```

(2) 表达式名=表达式

注意：EQU指令不占用内存空间！

如：
```
BUF1 DW 1,2,3
INTT EQU 5
BUF2 DW 4,5,6
```

那么3和4的地址是连续的！

EQU与=的差异

同一个程序中“=”可以对一个符号重复定义，但EQU不能对同一个符号重复定义

例：
Y1=7
Y1=128 √
Y1 EQU 7
Y1 EQU 128 ×

解除定义伪指令PURGE

格式：PURGE <符号1，符号2，…，符号n>

功能：解除指定符号的定义

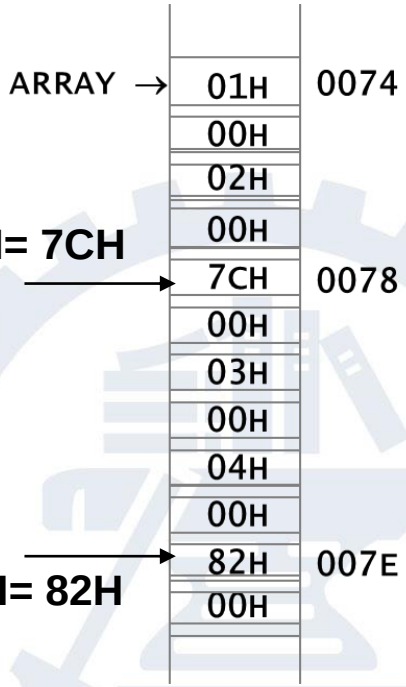
例：Y1 EQU 7
PURGE Y1
Y1 EQU 128

7. 地址计数器与对准伪操作

- 在代码段，$表示当前正在汇编的指令的地址

```
ORG  $+8      ；跳过8个字节的存储区
JNE  $+6      ；转向地址是 JNE 的地址 +6
JMP  $+2      ；转向下一条指令
```

$+4= 78H+4H= 7CH



- 在数据段，$表示当前地址计数器的值

```
ARRAY  DW  1, 2 , $+4 , 3 , 4 , $+4
```

$+4= 7EH+4H= 82H

- **地址计数器$的常用使用方法**

常用来确定数组中元素的个数

例： BUF1 DB 1，2，3，4，5

CNT1 EQU $-BUF1 （常用）

BUF2 DW 1，2，3，4，5

CNT2 EQU ($-BUF2)/2

CNT1、CNT2分别为数组BUF1、BUF2中数据元素的个数

- **ORG 伪操作**

指明下一条汇编语句的偏移地址，本身不占内存空间

```
SEG1    SEGMENT
        ORG    10   ; 设置$为10，此段目标代码从偏移地址10开始
        VAR1   DW   1234H; VAR1的偏移地址为10
        ORG    20   ; 在10和20地址之间是没有指令的
        VAR2   DW   5678H
        ORG    $+8  ; $增加8，即在5678H之后空出8个字节
        VAR3   DW   1357H
SEG1    ENDS
```

```
        ORG  100H
START:  ……
```

8. 表达式操作符

在指令中出现，主要是对常量进行运算

运算符类型	运算符及说明
算术运算符	+、 -、*(×)、/(÷)、MOD(取余数)
逻辑运算符	AND(与)、OR(或)、XOR(异或)、NOT(非)
位移运算符	SHL(逻辑左移)、SHR(逻辑右移)
关系运算符	EQ(=)、NE(≠)、GT(>)、LT(<)、GE(≥)、LE(≤)

8. 表达式操作符

(1) 算术操作符： + 、 - 、*、 /、Mod

```
VIDEO_BUF  DB  25*80*2  DUP(?)
ARRAY   DW   1,2,3,4,5,6,7
ARYEND  DW   ?
MOV  CX, (ARYEND-ARRAY)/2
BLOCK DB  1，2，3

MOV  AL, BLOCK+2   ；符号地址±常数有意义
```

等价于 MOV AL, [BLOCK+2]

(2) 逻辑和移位操作符

AND、OR、XOR、NOT、SHL、SHR

```
OPR1  EQU  25   ; 00011001B
OPR2  EQU  7    ; 00000111B

AND  AX, OPR1 AND OPR2  ; AND AX,1

MOV  AX,  0FFFFH SHL 2    ; MOV AX,0FFFCH
```

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

计算结果为逻辑值：不成立为0，即0H。成立为-1，即字节为FFH，字为FFFFH。

MOV FID, (OFFSET Y - OFFSET X) LE 128

```
X:  ……
    ……
Y:  ……
```

若 ≤ 128（真） 汇编结果： MOV FID, -1

若 > 128（假） 汇编结果： MOV FID, 0

(4) 数值回送操作符

OFFSET、SEG、LENGTH、SIZE

OFFSET / SEG　　变量 / 标号

功能：回送变量或标号的偏址 / 段址

如 MOV DX, SEG FUNC

LENGTH　变量

功能：回送由DUP定义的变量的单元数，其它情况回送1

SIZE　变量

功能：返回配送给该变量的字节数

等价于 LENGTH * TYPE　;DB的TYPE值为1，DW的TYPE值为2

例：

```
ARRAY   DW   100 DUP (?)
TABLE   DB   'ABCD'

MOV  CX,  LENGTH  ARRAY   ;  MOV  CX, 100
MOV  CX,  LENGTH  TABLE   ;  MOV  CX, 1
MOV  CX,  SIZE    ARRAY   ;  MOV  CX, 200
MOV  CX,  SIZE    TABLE   ;  MOV  CX, 1
```

(5) 属性修改的伪指令： PTR

- 用于暂时改变**内存变量或标号**的原有属性，但**不能修改寄存器**的原有属性

- **格式： 新属性 PTR （旧属性的）表达式**

  MOV AX, WORD PTR [BX]

- **方式：**与其它汇编指令配合使用，不能独立使用

PTR指令的应用场合

- 有一些指令中，操作数或表达式的属性是不明显的，需要加以明确。

  例：CALL DWORD PTR [BX]； 远调用

  CALL BYTE PTR [BX]； 段内近调用

- 需要修改操作数或表达式的属性的。

  例：F1 DW 1234H

  MOV AL，BYTE PTR F1； AL=34H

  例：F2 DB 23H，56H，18H

  MOV BX，WORD PTR F2； BX=5623H

(6) 定位类型的伪指令： PARA

- PARA（Paragraph,节)表明该段起始地址对齐到para，供OS使用

- 一般说来，各个逻辑段的首地址在‘节’的整数边界上（每16个存储单元叫做一节，1para = 16Bytes），即每个逻辑段的起始地址是16的整数倍，或地址的最低4位为0。

例如： code segment para ‘code’

主要内容

3.2.1 伪指令的功能与格式

3.2.2 主要的伪指令类型

3.2.3 汇编语言程序格式

3.2.4 汇编语言程序的上机过程

1. 汇编语言源程序结构

- 一个汇编源程序一般应该由3个逻辑段组成，即数据段、堆栈段和代码段
- 每个逻辑段以SEGMENT语句开始，以ENDS语句结束。整个源程序以END语句结束
  - 数据段用来在内存中建立一个适当容量的工作区，以存放常数、变量等操作数据。
  - 堆栈段用来在内存中建立一个适当的堆栈区，以便在中断、子程序调用时使用。
  - 代码段包括了许多以符号表示的指令，其内容就是程序要执行的指令。

1. 汇编语言源程序结构

- **完整段定义结构：用段定义伪指令**

```
数据段名 SEGMENT
 …
数据段名 ENDS
堆栈段名
SEGMENT
 …
堆栈段名 ENDS
代码段名 EGMENT
        ASSUME …
START：段地址装填
        …
        …

    MOV AH,4CH
    INT 21H
代码段名 ENDS
END     START
```

例：求字存储单元中两个数之差，结果存入下一个相邻的字单元中。

```
DATA    SEGMENT
BUF     DW 3483H,4596H
RES     DW  ?
DATA    ENDS
STACK   SEGMENT STACK 'STACK'
        STA  DW 100 DUP(?)
STACK   ENDS
CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA,SS:STACK
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,BUF
        SUB AX,BUF+2
        MOV RES,AX
        MOV AH,4CH
        INT  21H
CODE    ENDS
END     START
```

2. 程序段前缀结构FAR

过程定义伪指令：定义一个过程
格式：过程名 PROC [类型]
过程名 ENDP

例:三个数相加并把结果存放在SUM单元中

```
DATA  SEGMENT
BUF   DB 35H,78H,0A5H
SUM   DB ?
DATA  ENDS
CODE  SEGMENT
      ASSUME
CS:CODE,DS:DATA
ASUM  PROC FAR
START:PUSH DS
      MOV AX,0
      PUSH AX
      MOV AX,DATA
      MOV DS,AX
      MOV AL,0
      MOV SI,OFFSET BUF
      ADD AL,[SI]
      INC SI
      ADD AL,[SI]
      INC SI
      ADD AL,[SI]
      MOV SUM,AL
      RET
ASUM  ENDP
CODE  ENDS
      END START
```

3. 代码段中程序结束的两种方式

方式一：

```
......
CODE    SEGMENT
        ASSUME  ......
START:
            ......
            ......
            MOV    AH,4CH
            INT    21H
CODE    ENDS
            END    START
```

方式二：

```
......
CODE    SEGMENT
MAIN    PROC    FAR
        ASSUME  ......
START:
            PUSH   DS
            MOV    AX, 0
            PUSH   AX
                   ......
            RET
MAIN    ENDP
CODE    ENDS
            END    START
```

主要内容

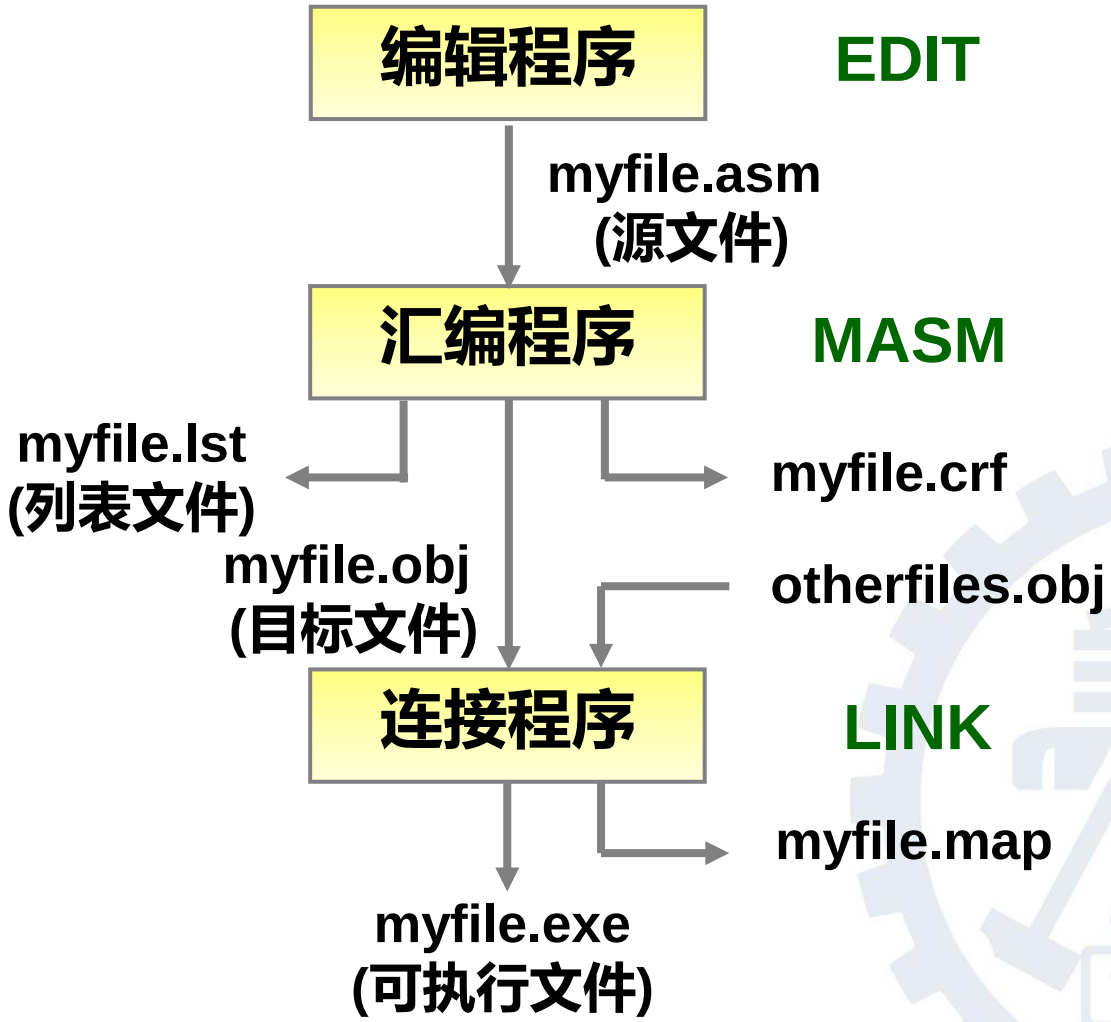
3.2.1 伪指令的功能与格式

3.2.2 主要的伪指令类型

3.2.3 汇编语言程序格式

3.2.4 汇编语言程序的上机过程

1. 程序运行步骤及生成的文件



2. 建立、运行汇编语言程序

```
C>EDIT  MYFILE . ASM ↙
C>MASM  MYFILE . ASM ↙
    Microsoft (R) Macro Assembler Version 5.10
    Copyright (C) Microsoft Corp 1981,1988.All rights reserved.
    Object filename [MYFILE.OBJ]:  ↙
    Source listing  [NUL.LST]: MYFILE.LST ↙
    Cross-reference [NUL.CRF]:  ↙
        47962 + 413345Bytes symbol space free
            0 Warning Errors
            0 Severe  Errors
```

```
C>LINK  MYFILE . OBJ  ↙
    Microsoft (R) Overlay Linker  Version 3.64
    Copyright (C) Microsoft Corp 1983-1988. All rights reserved.
    Run File [MYFILE.EXE]: ↙
    List File [NUL.MAP]: ↙
    Libraries [.LIB]:  ↙
    LINK : warning L4021: no stack segment
    C>MYFILE ↙
```

3. 汇编程序功能

汇编程序的主要功能：

- **检查源程序，给出出错信息**
- **产生目标文件(.obj)和列表文件(.lst)**
- **展开宏指令**

1、画图说明下列语句所分配的存储空间及初始化的数据

(1) AA DB 'BYTE',12,-12H,3 DUP (0,?,2 DUP(1,2),?)

(2) BB DW 5 DUP (0,1,2),?,-5, 'BY',TE',256H

2、假设程序中的数据定义如下:

PARTNO DW ?

PNAME DB 16 DUP (?)

COUNT DD ?

PLENTH EQU $-PARTNO

问PLENTH的值是多少? 它表示什么意义?

作 业

3、请设置一个数据段DATASG，其中定义以下变量

(1) FLD1为字符串变量： ‘computer’

(2) FLD2为十进制字节变量：32

(3) FLD3为十六进制字节变量：20

(4) FLD4为二进制字节变量：01011001

(5) FLD5为10个0的字节变量

(6) FLD6为数字的ASCII字符字节变量32654

(7) FLD7为包括5个十进制数的字变量：5,6,7,8,9

(8) FLD8为100个字变量