第四章　汇编语言程序设计

主要内容

- 汇编语言整体结构
- 分支程序设计技术
- 循环程序设计技术
- 子程序设计技术

4.1 汇编语言程序设计基础

1.段定义伪指令

格式： 段名 **SEGMENT** [定位类型][组合类型]['类别']

;段定义开始伪指令

⋮ } 指令语句或伪指令语句组成的段的实体

段名 **ENDS** ;段定义结束伪指令

4.1 汇编语言程序设计基础

①段名： 段名是所定义的段的名称，其构成规则与语句的名称一样。

段名一但定义，就具备5个属性
- 段地址
- 段内偏移地址
- 定位类型
- 组合类型
- 类别

- 格式中的定位类型、组合类型和类别外面的方括号不是语法符号，它表示该项是可以省略的；
- 在段定义时，SEGMENT与ENDS必须成对出现；
- SEGMENT与ENDS左边的段名必须一致。

4.1 汇编语言程序设计基础

②定位类型

- 告诉汇编程序（MASM.EXE），对该段汇编时，该段的起始边界的要求。
- 其类型有PAGE、PARA、WORD、BYTE四种，这四种类型的边界地址的要求如下：
  - PAGE = XXXX XXXX XXXX 0000 0000
  - PARA = XXXX XXXX XXXX XXXX 0000 (缺省型)
  - WORD = XXXX XXXX XXXX XXXX XXX0
  - BYTE = XXXX XXXX XXXX XXXX XXXX
    - ✓ 即边界地址（20位地址）应分别可以被256、16、2、1除尽，分别称为以页、节、字、字节为边界；
    - ✓ 在实际应用中，每个段的定位类型常选PARA（节）型。

③组合类型

告诉连接程序（LINK.EXE）在进行多模块目标程序连接时，该段与其它段连接的有关信息，如本段与其它段是否组合为同一段；组合后，本段信息与其他段信息的关系，如地址的高低端等。

4.1 汇编语言程序设计基础

④类别

- 类别可以使用任何一个合法的名称，但必须用单引号括起来；
- 在多模块程序设计中，连接时，将把不同模块中相同‘类别’的各段在物理上相邻地连接在一起，其顺序亦与LINK时提供的各模块顺序一致。这样做的一个好处是便于程序的固化；
- 在单模块程序设计中，类别可有可无。若有，它只是告知程序阅读者本段信息的含义。

4.1 汇编语言程序设计基础

2.汇编语言源程序的完整结构

```
STACK  SEGMENT  STACK
       DB   256 DUP(?)
       TOP  LABEL  WORD
STACK  ENDS
DATA1  SEGMENT
       ⋮
DATA1  ENDS
```

用DB、DW等伪指令定义的段的实体

4.1 汇编语言程序设计基础

```
DATA2  SEGMENT
```

用DB、DW等伪指令定义的段的实体

```
DATA2  ENDS
CODE  SEGMENT
      ASSUME  CS:CODE, DS:DATA1
      ASSUME  ES:DATA2, SS:STACK
START:  MOV  AX, DATA1
        MOV  DS, AX     ；DS初始化
        MOV  AX, DATA2
        MOV  ES, AX     ；ES初始化
```

4.1 汇编语言程序设计基础

```
    MOV  AX, STACK
    MOV  SS, AX      ；SS初始化
    MOV  SP, OFFSET  TOP
    ……
    MOV  AH, 4CH
    INT   21H        ；程序结束，返回DOS操作系统
CODE ENDS            ；代码段定义结束
    END  START       ；整个程序结束
```

用指令语句编写的完成某一功能的程序体。

4.1 汇编语言程序设计基础

3. ASSUME伪指令

该指令告诉汇编程序（MASM.EXE）在对源程序汇编时，源程序中的段名与哪个段寄存器建立关系。这种关系只是一种承诺关系，汇编程序对源程序汇编时，承认这种关系，但段寄存器的值并未确定，用户必须在代码段一开始用MOV指令对DS、ES、SS初始化。程序开始部分：

```
START：MOV  AX，DATA1
       MOV  DS，AX    ；对DS初始化
       MOV  AX，DATA2
       MOV  ES，AX    ；对ES初始化
       MOV  AX，STACK
       MOV  SS，AX    ；对SS初始化
```

4. LABEL伪指令

格式: 名称 LABEL 类型

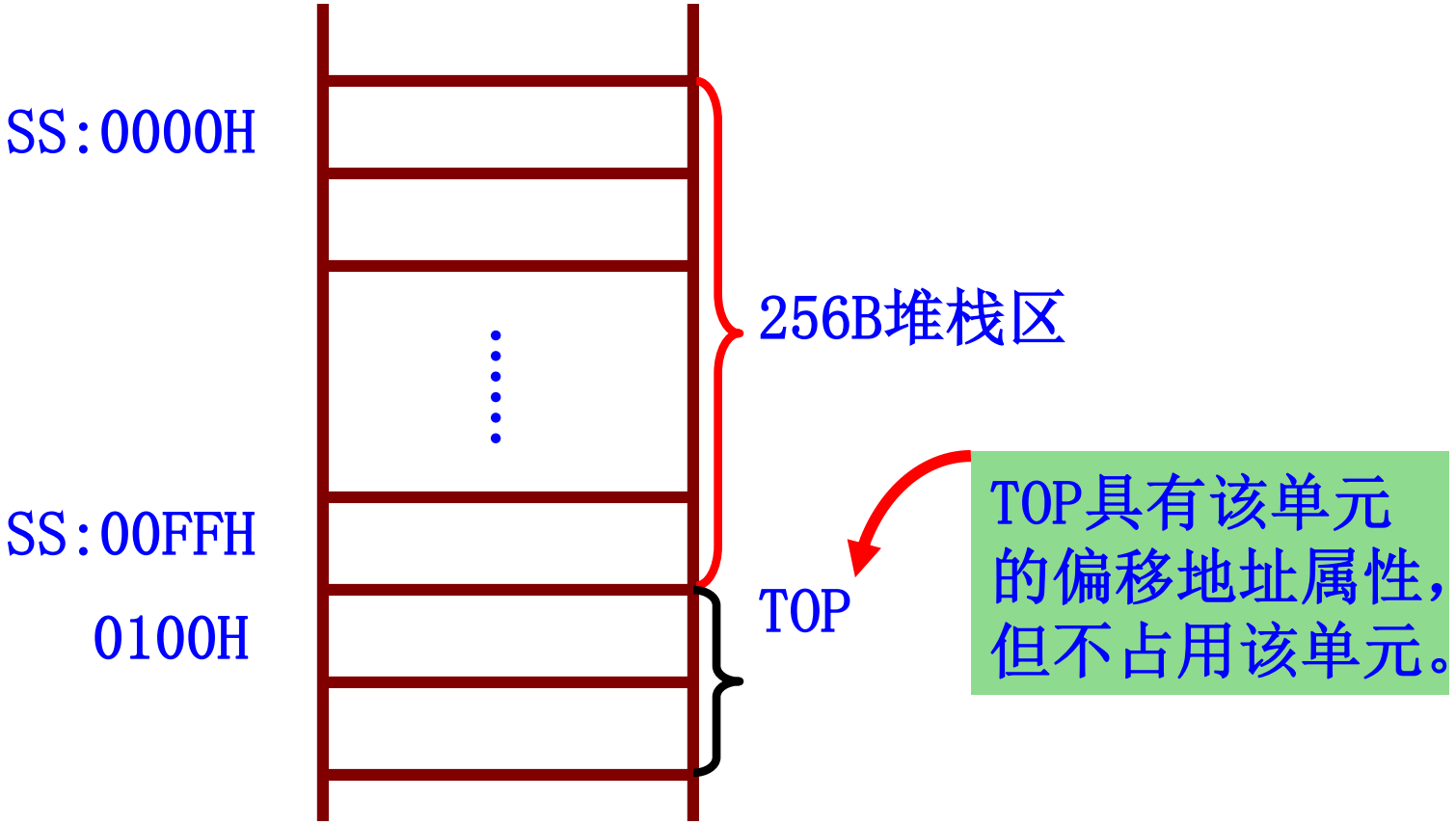
- LABEL伪指令的功能是定义某变量名或标号的类型。它虽具有段地址与偏移地址的属性，但它不占内存单元。
- 如前面定义的堆栈段：

```
STACK   SEGMENT   STACK
        DB    256  DUP(?)
        TOP  LABEL  WORD
STACK  ENDS
```

注：由于STACK段内的偏移地址开始为0000H，段内留出了256个字节作为堆栈区，因此汇编到TOP处时，偏移地址为0100H。

4.1 汇编语言程序设计基础

汇编情况如下图所示：



由于堆栈指针SP初始化后要指向栈底+1单元，所以上述程序段中有：

……

```
MOV  AX, STACK
MOV  SS, AX
MOV  SP, OFFSET  TOP
```

此段程序完成对SS、SP的初始化

……

4.1 汇编语言程序设计基础

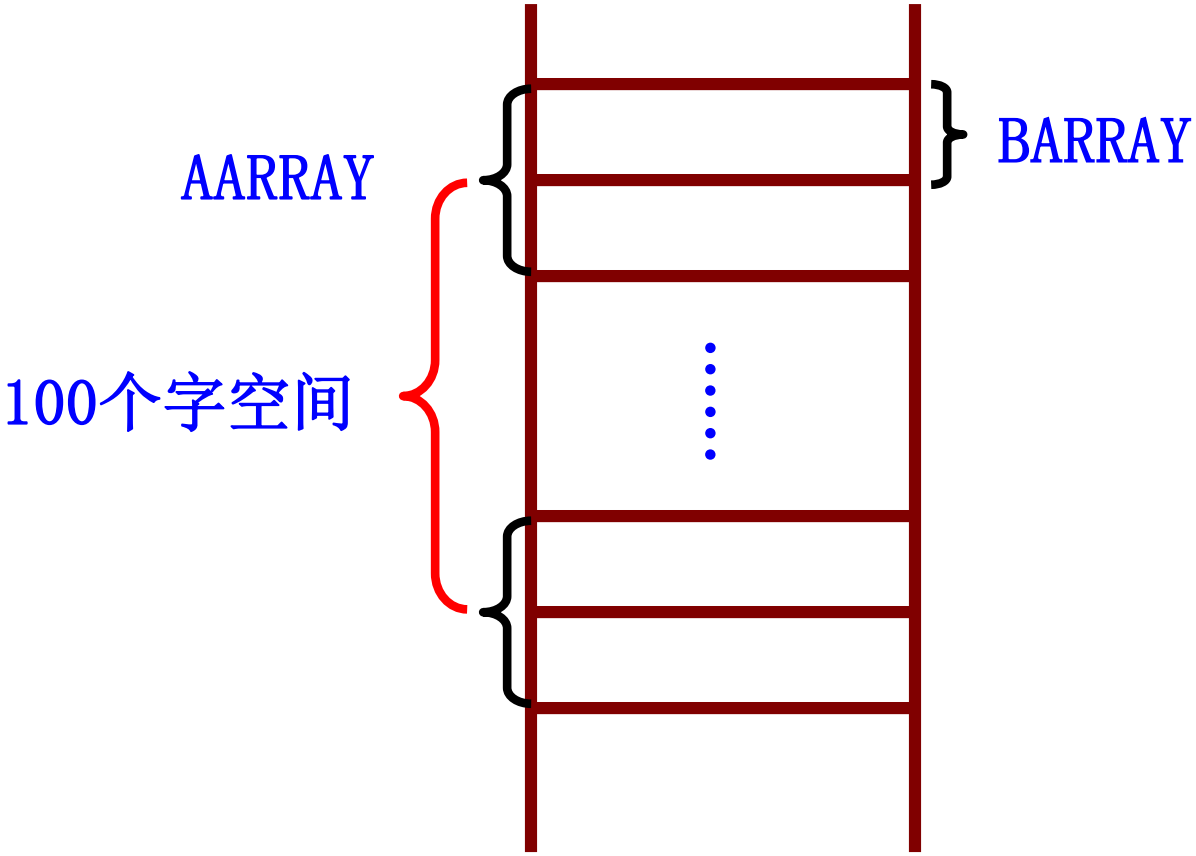
LABEL伪指令的功能是定义某变量名或标号的类型，具有段地址与偏移地址的属性，但它不占内存单元。

例： BARRAY LABEL BYTE

AARRAY DW 100 DUP(?)

上面定义了两种类型的变量，BARRAY为字节类型，AARRAY为字类型，它们的段和偏移地址属性完全相同，都是下面保留的100个字空间的首地址，其目的是为了程序中可以对这100个字空间作两种不同类型的操作。这一点上，LABEL的作用与前面介绍的PTR操作符的作用相类似。

4.1 汇编语言程序设计基础



4.1 汇编语言程序设计基础

当需要对该100个字空间进行字操作时，可利用AARRAY字变量，如：

```
MOV   AX, ARRAY
```

当需要对该100个字空间进行字节操作时，可利用BRRAY字节变量，如：

```
MOV   AL, BRRAY
```

等效于

```
MOV   AL, BYTE  PTR  ARRAY
```

5.END伪指令

格式： **END** 表达式

- 该伪指令标志整个源程序的结束，告诉汇编程序汇编到此结束；
- 每个单独汇编的源程序的结尾必须有**END**伪指令。格式中的表达式是该程序运行时的启动地址，它通常是可执行语句的标号。

4.1 汇编语言程序设计基础

如前面完整结构程序中的最后有：

```
        ……

        MOV   AH，4CH
        INT    21H
CODE ENDS
        END   START
```

总汇编结束

起始地址表达式

6. = 伪指令和EQU伪指令

格式： 名称 = 表达式

名称 EQU 表达式

功能：将表达式的值赋给左边的名称，但表达式的值不能超过65535。

- 该指令本身不占内存空间，是为格式中的表达式部分赋一个名称；
- 在编写源程序时，凡用到表达式值的地方都可以用名称（符号常量）来代替。汇编时，在出现名称的地方又用表达式的值取代了该名称。

注：EQU伪指令定义的名称在程序中只能定义一次，而用 = 伪指令定义的名称可以重新定义。

示例：**Avalue = 5**

……

Avalue = Avalue+3 ✓

表示开始把符号名Avalue定义为数值5，后来把符号名Avalue重新定义为代表数值8。

但是不能写成：

Avalue EQU 5

……

Avalue EQU Avalue+3 ✗

4.1 汇编语言程序设计基础

示例：

```
        ⋮
COUNT  EQU 5*8
BPT  = BYTE PTR
        ⋮
MOV  CX，COUNT  ；等效于 MOV CX，5*8
MOV  BPT[BX]，0  ；等效于 MOV BYTE PTR [BX]，0
        ⋮
```

7. ORG伪指令

格式： ORG 表达式

- 格式中的表达式的值是一个2字节的无符号数；
- ORG伪指令的功能是指明该语句下面的指令或者变量在段内的偏移地址。

例： ORG 0100H

该伪指令指出，下面指令或变量的偏移地址为0100H。

ORG伪指令一般常用于数据段中来确定某变量的偏移地址。