# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

## 一. 汇编语言程序设计的基本步骤

### 1. 分析问题

首先必须明确求解问题的意义和任务。对题目给出的已知条件和要完成的任务进行详细地了解和分析，将一个实际的问题转化为计算机可以处理的问题。

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

### 2. 确定算法

所谓算法，简单地说就是计算机能够实现的有限的解题步骤。我们知道，计算机只能进行最基本的算术运算和逻辑运算，要完成较为复杂的运算和控制操作，必须选择合适的算法，这是正确编程的基础。

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

若题目涉及到某种运算，则必须写出适合程序设计的正确算法，若题目要完成的功能未涉及到运算，也要写出编程思想。

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

### 3. 设计流程

将提出的算法或编程思想用流程图的方式画出来。图4.3.1给出了流程图中较为通用的几种符号。

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

起始和终止框

执行框

判断框

连接框

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

### 4. 根据流程图编写程序

编写程序是采用程序设计语言来实现上面已确定的算法，此过程有些书上称为编码。本书所介绍的是采用汇编语言编写程序。采用汇编语言编写程序应注意以下几个问题：

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

（1）必须详细了解CPU的编程模型、指令系统、寻址方式及相关伪指令;

（2）必须进行存储空间和工作单元的合理分配;

（3）多次使用的程序段可采用子程序或宏指令;

（4）尽可能用标号或变量来代替绝对地址和常数;

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

### 5. 程序的检验

程序编写好以后，必须经过书面检查和上机调试，以便说明程序是否正确。检验时，应预先选择典型数据，检查是否可以得到预期结果。

# 4.3 汇编语言程序设计的基本方法

## ——汇编语言程序设计的基本步骤

### 6. 编写说明文件

一个完整的软件应有相应的说明文件，这不仅便于用户使用，也便于对程序的维护和扩充。说明文件主要应包括程序的功能和使用方法，程序的基本结构和所采用的主要算法以及程序必要说明和注意事项等。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

## 二. 8086汇编语言源程序的完整结构及伪指令

我们知道，8086/8088 CPU的地址空间是分段结构的，程序中出现的数据与代码以及程序中用到的堆栈都必须纳入某个段中。那么，如何告诉汇编程序，源程序中的哪些内容属于数据段，哪些属于代码段呢？这自然是由汇编系统中提供的伪指令来实现的。下面我们首先介绍构成完整程序的有关伪指令。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

## 1. 段定义伪指令

格式:

```
段名   SEGMENT   [定位类型][组合类型]['类型']
                                  ;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
```

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

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，就具备了以下5个属性。

- 段地址
- 段内偏移地址
- 定位类型
- 组合类型
- 类别

其中，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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

- 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。

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:

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

```
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
```

即它们的边界地址（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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

在实际应用中，每个段的定位类型常选PARA（节）型。因为若选PAGE（页）型，将会使相邻的段间有较大空间的浪费；而选WORD或BYTE型，又很难做到使一个段的偏移地址从0000H开始。

## 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何等。组合类型有以下6种不同的类型：

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

NONE型：表示本段与不同模块中的其它段在逻辑上不发生关系。连接后各模块中的各段都有自己的段地址（也称基地址）。

STACK型：组合后的这个段用作堆栈。当段定义中指明了STACK类型后，说明堆栈段已经确定，所以，在可执行文件装入内存后段寄存器SS中已是该段的段地址，堆栈指针SP已指向堆栈底。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

PUBLIC型

COMMON型

AT表达式型

MEMORY型

④类别:

类别可以使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。当类别相同的各段的段名不同时，它们连接后虽在同一物理段内，但

它们仍不属于同一段，也就是它们的段基址不相同。这样做的一个好处是便于程序的固化。在编程时，它们都是独立的代码段，各段有各自的段基址，但连接后，他们却在同一物理段，从而可以固化在一起。

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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

## 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
          ⋮   } 用DB、DW等伪指令定义的段的实体
DATA1   ENDS
```

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

```
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
```

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

```
        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。

## 3. LABEL伪指令

格式：名称　LABEL　类型

格式中的类型有BYTE、WORD、DWORD、结构名、记录名、NEAR、FAR共7种。前5种属于变量的类型，后两种是属于标号的类型。结构和记录是由伪指令定义的两种数据类型(关于这两种数据类型的定义在此就不在叙述，请参阅有关资料）。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

格式中的名称就是语句的名称，为一标识符，若后面的类型是前5种之一，那么该名称就是变量名；当类型为后两种时，该名称就是标号。我们已经知道，变量与标号除具有类型属性外，还具有段地址和偏移地址的属性，名称的这两个属性就是汇编程序汇编到该语句时语句所在的段地址和偏移地址。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

如前面定义的堆栈段:

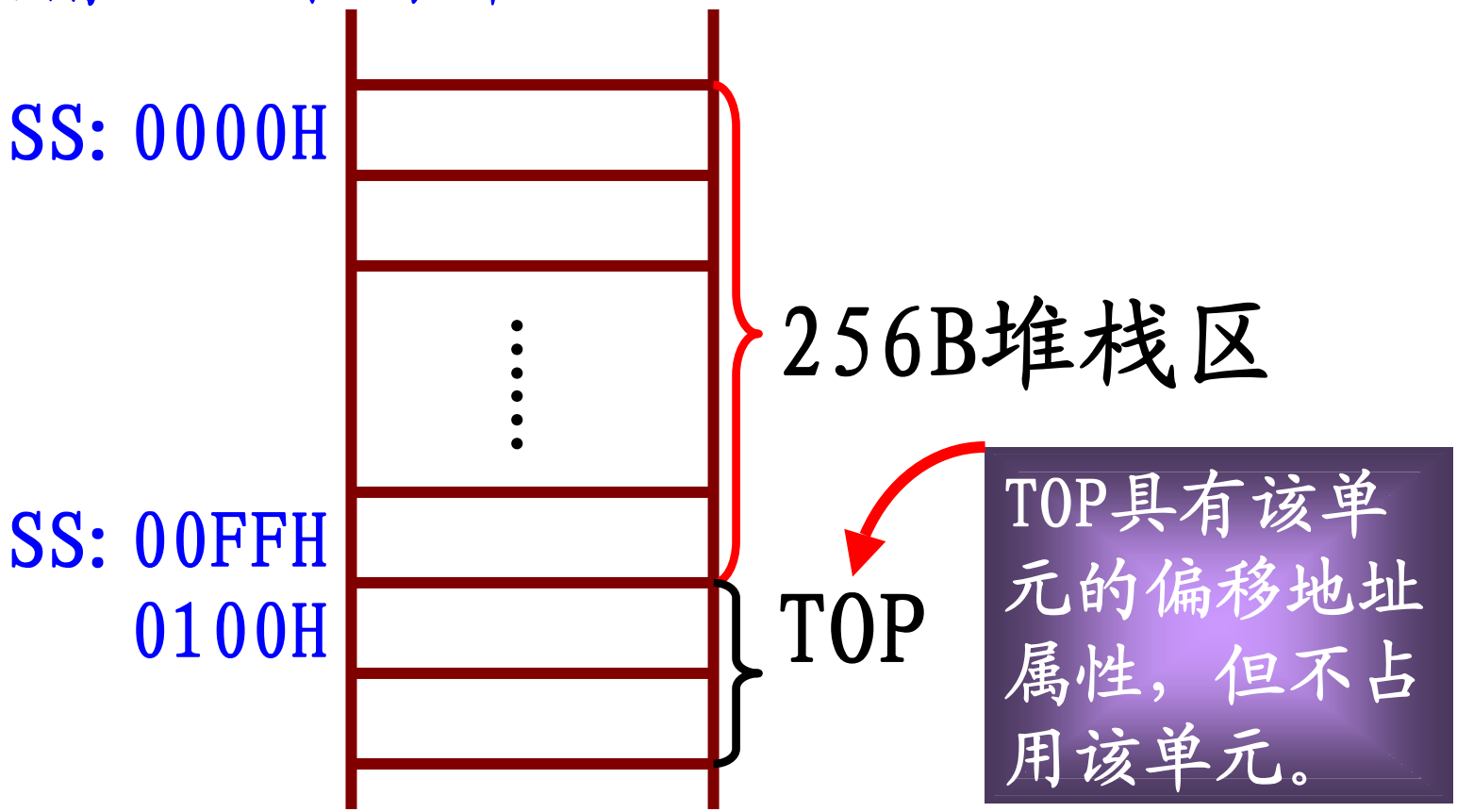
```
STACK   SEGMENT   STACK
        DB   256   DUP(?)
  TOP   LABEL   WORD
STACK   ENDS
```

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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

汇编情况如下图所示：



# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

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

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

又由于前面介绍组合类型时，介绍了若某段定义时，其组合类型选为STACK型，系统默认该段为堆栈段。所以，在可执行文件装入内存后，段寄存器SS中已是该段的段基址，堆栈指针SP已指向堆栈底+1单元的偏移地址。因此，在程序设计时，程序段就不用给SS、SP 初始化了。

即：

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

这段程序可以缺省。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

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

例如:

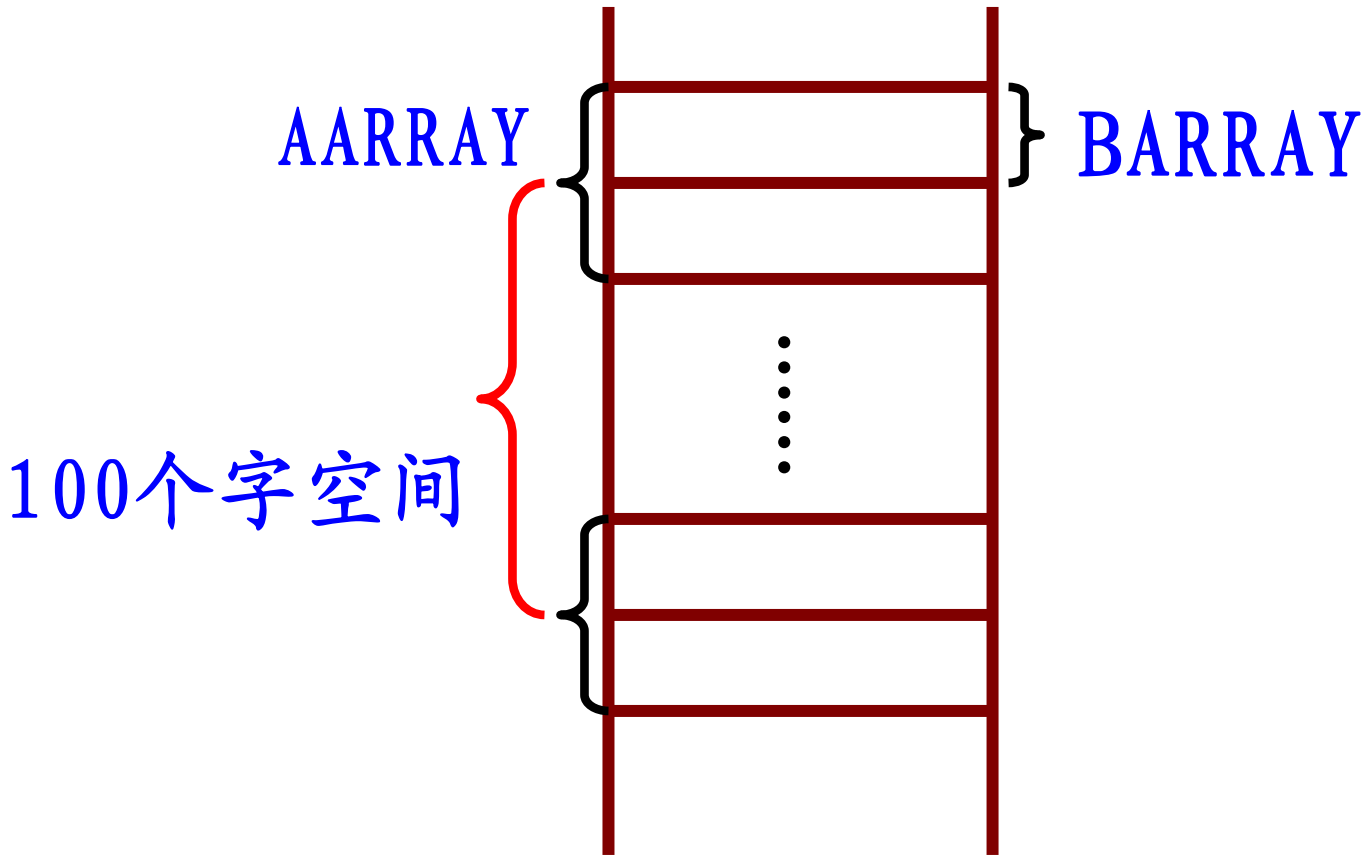
```
BARRAY   LABEL   BYTE

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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令



当需要对该100个字空间进行字操作时，可利用AARRAY字变量。如下面指令格式是正确的。

```
MOV   AX, ARRAY
```

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

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。如下面指令格式是正确的。

```
MOV   AL,BRRAY
```

等效于

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

## 4. ASSUME伪指令

ASSUME伪指令告诉汇编程序（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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

如前面完整结构程序中的下段程序：

```
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
```

因为CS不能做DST，所以用户不能对CS初始化，CS和IP的初始化是系统赋给的，其方法见下面的END伪指令介绍。

# 4.3 汇编语言程序设计的基本方法
## ——8086汇编语言源程序的完整结构及伪指令

- 格式:

**ASSUME** 段寄存器:名称[,段寄存器:名称,...]

其中:名称可以有以下几种情况

① 由**SEGMENT**伪指令定义的段名.

如:**ASSUME CS:CODE,DS:DATA;**

② 表达式:**SEG**变量名或**SEG**标号.

如:变量**ADR1**,可作为名称,

**ASSUME DS:SEG ADR1;**

③ **GROUP**伪指令定义的段组名.

如:**GCODE**为段组名,**ASSUME CS:GROUP**将**CS**指向该段组的组头.

## 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。它告诉汇编程序汇编到此结束。所以，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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

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。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

伪指令本身不占内存空间。它的功能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。但汇编时，在出现名称的地方又用表达式的值取代了该名称，例如是一个变量名，那么它将被这个变量名取代。

# 4.3 汇编语言程序设计的基本方法

## ——8086汇编语言源程序的完整结构及伪指令

- 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。

```
      ……
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。

例如:

```
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。

# 4.3 汇编语言程序设计的基本方法
## ——8086汇编语言源程序的完整结构及伪指令

### 8. TITLE伪指令

格式: TITLE　正文

该伪指令是为程序指定一个标题的. 在列表文件的每一页的第一行将打印出这个标题.

格式中的正文即指定的标题, 不超过60个字符.

## 9. PAGE伪指令

格式: PAGE　　参数1, 参数2

该伪指令是为汇编程序所产生的列表文件指定每页的行数和每行的字符数的. 参数1表示每页的行数, 参数2表示每行的字符数.

通常PAGE伪指令是程序的第一条语句.

作业：

P188 12.