第4章　汇编语言程序设计

本章主要内容

§ 4.1 汇编语言程序格式和伪指令

§ 4.2 DOS系统功能调用和BIOS中断调用

§ 4.3 汇编语言程序设计方法与实例

引子

- **什么是汇编语言（Assembly Language）？**
  - **定义**：使用指令的助记符、符号地址和标号等编写的程序设计语言。
  - **特点**：汇编指令与机器码一一对应；特定汇编语言只针对特定CPU
- **用汇编语言编程的优缺点**
  - **优点**：运行速度快，实时性好，占用内存空间小，能最大限度地发挥硬件的作用。
  - **缺点**：可读性差，移植性差，对程序员要求高
- **汇编语言的适用场合**
  - 绝大部分系统软件；涉及快速处理、位处理和访问硬件设备的高效程序——实时数据处理程序、实时控制程序、高级绘图程序、游戏程序等。

- 学习汇编语言的目的：

学会一种汇编语言，就能举一反三，触类旁通。学会8086汇编语言编程，就打好了学习32位高档机程序设计的基础，也便于从事单片机和嵌入式系统的设计开发。

- 汇编语言的汇编处理过程

1)编辑(EDIT.EXE)：按语法规则编写源程序PROG.ASM；

2)汇编(MASM.EXE)：用汇编程序将源程序翻译成目标文件PROG.OBJ；

3)链接(LINK.EXE)：用连接程序对1个或几个.OBJ模块连接后，生成能在机器上执行的程序PROG.EXE。

§4.1 汇编语言程序格式和伪指令

4.1.1 汇编语言程序格式

4.1.2 伪指令语句

4.1.3 完整的汇编语言程序框架

4.1.1 汇编语言程序格式

汇编语言程序有两类指令语句：

指令（instruction）

与机器码一一对应

伪指令（Pseudo Instruction）

指示性语句，不产生机器码，在汇编过程中完成特定功能。

4.1.1 汇编语言程序格式

1. 指令语句

指令语句格式

标号：指令助记符　操作数　；注释

1）标号

- 标号是指令的符号地址，后面须加冒号“:”。
- 标号以英文字母或特殊字符打头
- 标号具有段基址、偏移量及类型三种属性。

4.1.1 汇编语言程序格式

1. 指令语句

指令语句格式

标号：指令助记符　操作数　；注释

2）指令助记符：

表示指令的操作码，不可缺少。

4.1.1 汇编语言程序格式

1. 指令语句

指令语句格式

标号：指令助记符　操作数　；注释

3）操作数：1条指令可包含1个或2个操作数，也可没有操作数。

可以是：常量，字符和字符串(ASCII码)，变量，标号，存储器，表达式。

1. 指令语句

*操作数——具体内容

- 1条指令可包含1个或2个操作数，也可没有操作数。
- 操作数的组成：

➢ 常数　二进制数，加B；
　　10进制数，可加D或省略；
　　16进制数，加H，A~F前要加0；
　　2-10进制BCD数，加H，要用调整指令

➢ 字符或字符串　用单引号’ ’ 括起来

➢ 变量　程序运行期间可修改，数值可由DB、DW、DD等来定义

➢ 标号　如JMP　NEXT

➢ 存储器

➢ 表达式

4.1.1 汇编语言程序格式

1. 指令语句

指令语句格式

标号：指令助记符　操作数　；注释

4）注释：说明指令或程序的功能，增强程序可读性，可省略。注释前必须加分号";"

【例】 1）MOV CX,100　;传送100到CX

2）MOV CX,100　;循环计数器置初值

4.1.1 汇编语言程序格式

2. 伪指令语句

伪指令语句的格式如下：

名字　伪指令指示符　操作数　；注释

1）名字

- 是给伪指令语句起的名称，格式要求与标号类似，名字后不能跟冒号“:”。

4.1.1 汇编语言程序格式

2. 伪指令语句

伪指令语句的格式如下：

名字　伪指令指示符　操作数　；注释

2）**伪指令指示符：**代表伪指令功能，是伪指令语句中不可缺少部分，如：

段定义语句　　SEGMENT和ENDS

过程定义语句　PROC和ENDP

4.1.1 汇编语言程序格式

2. 伪指令语句

伪指令语句的格式如下：

名字　伪指令指示符　操作数　；注释

3）**操作数：**无操作数，或带1到多个操作数；

4）**注释：**说明功能，和指令注释相同。

【例】VAR DB 05H,06H,07H ;定义字节型（Data Byte）变量
;VAR,并将其初始化为05H,06H,07H

4.1.1 汇编语言程序格式

3. 表达式和运算符

- 表达式的定义：将常数、符号、寄存器等通过**运算符**连接起来的式子叫做表达式。
- 特点：不论是常数、变量还是标号，都可用表达式的形式给出。

表达式的运算符

表 4.1　MASM 表达式中的运算符

类型	符号	名称	运算结果	举例
算术运算符	+	加法	和	3+6=9
	−	减法	差	8−3=5
	*	乘法	乘积	4 * 6=24
	/	除法	商	28/5=5
	MOD	模除	余数	28 MOD 5=3
	SHL	按位左移(n 次)	左移后 2 进制数	0010B SHL 2=1000B
	SHR	按位右移(n 次)	右移后 2 进制数	1100B SHR 1=0110B
逻辑运算符	NOT	非运算	逻辑非结果	NOT 1100B=0011B
	AND	与运算	逻辑与结果	1011B AND 0011B=0011B
	OR	或运算	逻辑或结果	1011B OR 1100B=1111B
	XOR	异或运算	逻辑异或结果	1011B XOR 0010B=1001B

表达式的运算符(续)

表 4.1　MASM表达式中的运算符

类型	符号	名称	运算结果	举例
关系运算符	EQ	相等	结果为真,输出全1 结果为假,输出全0	5 EQ 10B=全0
	NE	不等		5 NE 10B=全1
	LT	小于		5 LT 2=全0
	LE	小于等于		5 LE 101B=全1
	GT	大于		5 GT 011B=全1
	GE	大于等于		5 GE 110B=全0
数值返回符	SEG	返回段基址	段基址	SEG N1=N1所在段基址
	OFFSET	返回偏移地址	偏移地址	OFFSET N1=N1的偏移地址
	LENGTH	返回变量单元数	单元数	LENGTH N1=N1的单元数
	TYPE	返回变量类型	(见表4.3)	
	SIZE	返回变量总字节数	总字节数	SIZE N1=N1的总字节数

表达式的运算符(续)

(续)表 4.1

类型	符号	名称	运算结果	举例
修改属性符	PTR THIS 段寄存器名	修改类型属性 指定类型属性 段超越前缀	修改后类型 指定后类型 修改段	BYTE PTR [BX] ALPHA EQU THIS BYTTE ES:[BX]
其它运算符	HIGH LOW SHORT () [ ]	分离高字节 分离低字节 短转移说明 圆括号 方括号	取高字节 取低字节 -128~127 字节间转移 改变运算优级先级 下标或间接寻址	HIGH 1234H=12H LOW 1234H=34H JMP SHORT LABEL (8-3)*6=30 MOV AX,[BX]

运算符的优先级：表达式运算符也有优先级

表 4.2　运算符的优先级别

优先级		运算符
高级	0	(),[ ],LENGTH,SIZE
↑	1	PTR,OFFSET,SEG,TYPE,THIS,CS:,DS:,ES:,SS:(4 个段超越前缀)
	2	HIGH,LOW
	3	* ,/,MODE,SHL,SHR
	4	+,−
	5	EQ,NE,LT,LE,GT,GE
	6	NOT
	7	AND
	8	OR,XOR
低级	9	SHORT

◆ 常用运算符举例

1）算术运算符

〖例4.1〗利用**现行地址符“$”**和减法运算符“–”**求数组的长度**。程序段：

```
DATA    SEGMENT                    ;数据段
LIST    DB 12，38，5，29，74   ;LIST数组（变量）
COUNT   EQU $-LIST                  ;COUNT=现行地址-
                                   ;LIST的偏移地址
DATA    ENDS
                ⋮
        MOV CX，COUNT              ;CX←LIST数组长度
```

- **解：LIST变量的起始地址偏移量为0，“$”符表示本指令的现行地址偏移量**，它等于5，所以$-LIST=5-0=5，并赋予常量COUNT，这样可很方便地求得变量**字节长度**。

◆ 常用运算符举例

2）逻辑运算符和关系运算符

〖例4.2〗将表达式的运算结果送到寄存器中。

```
MOV  AL，NOT  10110101B
     ; AL←01001010B
MOV  BL，10H  GT  20H
     ; BL←00H，因10H>20H为假，输出全0
MOV  BX，6  EQ  0110B
     ; BX←FFFFH，因6=6为真，输出全1
```

◆ 常用运算符举例

3）数值返回运算符

□数值返回运算符OFFSET和SEG

〖例4.3〗 将TABLE变量的段基址：偏移量送入 DS：BX。

```
TABLE  DB  40H，79H，24H，30H，19H     ;数字0~9的
           12H，02H，78H，00H，18H     ;七段代码表
           ┇
       MOV  BX，OFFSET TABLE    ;BX←TABLE的偏址
       MOV  AX，SEG TABLE       ;AX←TABLE的段址
       MOV  DS，AX              ;DS←TABLE的段址
```

◆ 常用运算符举例

4）修改属性运算符

〖例4.6〗对存储单元的属性进行修改。

```
INC     BYTE  PTR [BX]
        ; 将字节存储单元的内容增1
```

- 用“BYTE PTR”指明存储单元[BX]为字节单元。

```
MOV     BX，ES：[DI]
        ; BX ←（16×ES+DI）的内容
```

- 未加段超越前缀ES时，存储单元[DI]默认DS为段基地址，加了ES:操作符后，段基地址修改成了ES。

4.1.1 汇编语言程序格式

4.1.2 伪指令语句

4.1.3 完整的汇编语言程序框架

4.1.2 伪指令语句

1. 段定义语句

- 段定义语句：**SEGMENT**和**ENDS**

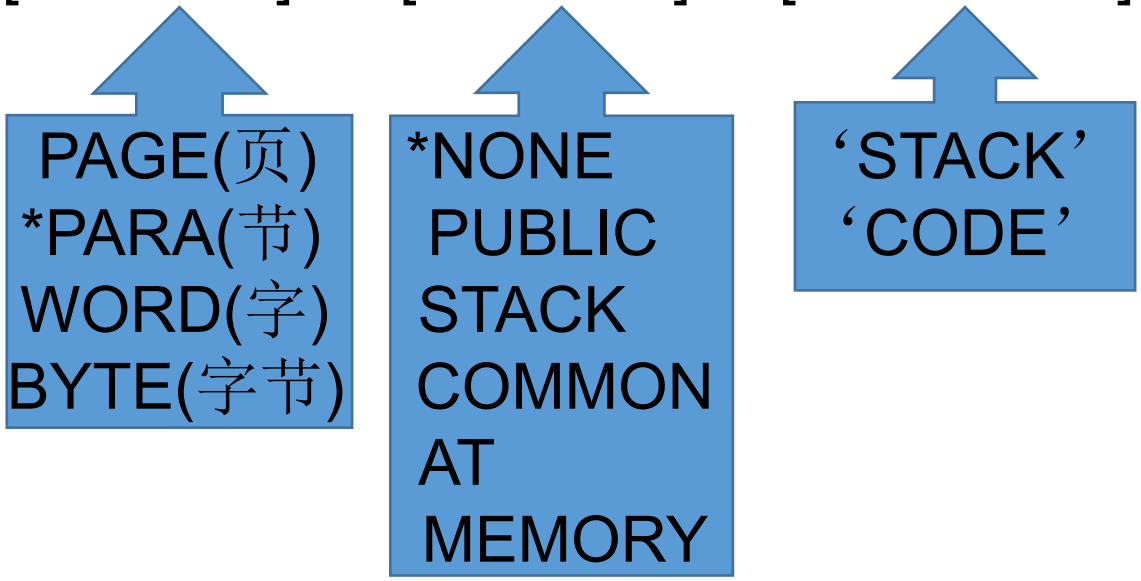
〖例4.7〗用段定义语句定义一个数据段，段名为DATA，段中包含X、Y两个变量。

```
DATA    SEGMENT          ;数据段开始，DATA为段名
                         ;表示该段的基址
    X       DW  1234H    ;变量X的段基址：偏移量
                         ;=DATA：0000，内容1234H
    Y       DB  56H      ;变量Y的段基址：偏移量
                         ;=DATA：0002，内容为56H
DATA    ENDS             ;数据段结束
```

1.段定义语句

- 段定义语句的一般形式：

段名　SEGMENT [定位类型]　[组合类型]　[‘分类名’]



段名　ENDS

- **加“[　]”项可省略**，但堆栈段的组合类型是STACK，不可省略。

1.段定义语句

1）定位类型（Align Type）

- **用处：用LINK程序将程序中的段相互衔接时，用定位类型来确定该段存储器的起始边界要求。**
- 定位类型有四种:

PAGE 该段起始地址能被256（页）整除

PARA 该段起始地址能被16（节）整除

WORD该段起始地址能被2（字）整除

BYTE 起始地址可从任何地方开始

1.段定义语句

2）组合类型（Combine-Type）

- 用处：组合类型告诉LINK程序本段与其它段关系

- 包括：

NONE　与其它段不连接，各段有独立段基址和偏移量。

PUBLIC　同名同类别模块段连接成一段，段基址同, 偏移量不同。

COMMON　本段与其它段覆盖，偏移地址名称不同。

STACK　这是堆栈段，不可省略。

MEMORY　连接时该段放在所有段最后（最高地址）。

AT　定义本段的段基地址。如AT 2000H定义该段的段基地址为20000H。

1.段定义语句

3）分类名（‘class’）

用处：LINK将分类名相同的逻辑段组成1个段组

- 分类名有‘STACK’，‘CODE’和‘DATA’等。

2.段分配语句 ASSUME

- **用处：** 告诉汇编程序，4个段寄存器CS、DS、SS、ES分别与哪些段有关。
- **格式：**

  ASSUME CS：代码段名，DS：数据段名

  SS：堆栈段名，ES：附加段名

◆**注：** 只是描述相关性，并没有给段寄存器具体赋值，**程序中还需要段寄存器赋值语句**。

3.过程定义语句PROC和ENDP

- **功能：**定义过程（Procedure），或称子程序（Sub Function），用CALL语句来调用。
- **格式：**

过程名　**PROC**　[NEAR]/FAR

⋮

RET

过程名　**ENDP**

3.过程定义语句PROC和ENDP

- **功能：** 定义过程（Procedure）
- **格式：** 过程名 **PROC** [NEAR]/FAR

RET

过程名 **ENDP**

过程属性： 过程名像标号一样，有3种属性：段基址、偏移地址和距离属性（NEAR或FAR），距离属性默认值为NEAR

4.变量定义语句(DB、DW、DD、DQ和DT)

- 变量定义语句的一般形式为：

  变量名 伪指令指示符 操作数；注释

- **变量名：**用符号表示，也可以省略。
- **伪指令：**包括DB、DW、DD、DQ和DT，分别定义字节（**B**yte）、字（**W**ord）、双字（**D**ouble Word）、4字（**Q**uater Word）和10字节（**T**en Byte）变量。
- **操作数：**可以设置具体**初值**，也可用**问号“?”**来表示**随机值**。可以用**DUP**来定义**重复值**。

4.变量定义语句--举例

〖例4.8〗　变量定义语句举例。

```
FIRST     DB    ?          ; 定义一个字节变量
                           ; 初始值不确定
SECOND    DB    20H，33H   ; 定义两个字节变量
THIRD     DW    1122H，3344H ;定义两个字变量
FOUR      DD    12345678H  ;定义一个双字变量
```

4.变量定义语句--重复值DUP举例

- 用**DUP**来定义**重复变量**，其格式为：

  变量名　**伪指令指示符**　*n* DUP (操作数)

  其中*n*为重复变量的个数。

〖例4.9〗

```
N1    DB    100     DUP (?)
                ; 分配100个字节单元，初值不确定
N2    DW    10      DUP (0)
                ; 定义10个字单元，初值均为0
N3    DB    100 DUP (3 DUP(8), 6)
                ; 定义100个“8，8，8，6”的数据项
```

4.变量定义语句

- 操作数项（初始化数据项）也可写成**单个字符或字符串**的形式，通常用**字节**来表示。

〖例4.10〗字符串变量举例

```
DB    ‘Welcome’
```

4.变量定义语句---变量存放情况

〖例4.11〗数据在存储单元中的存放形式

```
DATA1    DB    '3', 'A'
DATA2    DW    98, 100H, -2
DATA3    DD    12345678H
DATA4    DB    100 DUP (0)
```

5.程序结束语句

- **功能：** 指示源程序结束。一般位于程序的最后一行，汇编程序遇到END则停止汇编。
- **格式：**

  END [标号名或名字]

- **说明：** 标号名或名字可省略。

6.其它伪指令

1）等值伪指令EQU

- 使用EQU语句可使程序更清晰、易读，其格式为：

符号名 **EQU** 变量、标号、常数等

〔例4.12〕

```
Profit   EQU   10    ;常数值10赋给符号名Profit
CNT1  EQU   41H   ;常数值41H赋给符号名CNT1
COUNT  EQU    8   ;常数值8赋给COUNT
```

说明：汇编程序遇到EQU执行直接替代的操作。

6.其它伪指令

2）对准伪指令 EVEN

- **功能：**将下一语句指向的地址调整为偶地址，确保存取一个字数据只要进行一次操作。

〖例4.14〗对准伪指令举例。

```
DATA    SEGMENT
X       DB ‘M’ ; X变量的偏移地址为0
        EVEN    ; 将下一语句指向地址调整为偶数
Y       DW  100  DUP(?)
                ; Y变量从地址为02H处开始存放
DATA    ENDS
```

6.其它伪指令

3）ORG伪指令

- **功能：**为ORG下面一条语句**指定起始偏移地址**，可放在程序的任何位置上。

〖例4.15〗 ORG伪指令举例。

```
DATA    SEGMENT
        ORG  1200H
    A1  DB  12H，34H        ; A1变量偏移地址为1200H
        ORG   2000H
    A2  DW  3040H，2830H; A2变量偏移地址为2000H
DATA    ENDS
```

6.其它伪指令

```
CODE    SEGMENT
        ORG   400H
    ;此段代码段起始地址偏移量为400H
        ASSUME  CS：CODE，DS：DATA
        ┇
CODE    ENDS
```

4.1.1 汇编语言程序格式

4.1.2 伪指令语句

4.1.3 完整的汇编语言程序框架

4.1.3 完整的汇编语言程序框架

- **描述：** 完整的汇编语言程序包含**数据段**、**代码段**、**堆栈段**和**附加数据段**。

➢ **说明：**

代码段： 是**必须要有**的；

堆栈段： 根据情况设置；

数据段： 代码段中要用到变量或数据时，应该设置数据段。

附加段： 用到字符串时需要设置。

1.完整的汇编语言程序框架

〖例4.17〗 汇编语言程序框架。

```
DATA    SEGMENT                    ;数据段
 X      DB   ?
 Y      DW   ?
DATA    ENDS

EXTRA   SEGMENT   ;附加段
ALPHA   DB    ?
BETA    DW   ?
EXTRA   ENDS

STACK   SEGMENT  PARA  STACK  'STACK'   ;堆栈段
STAPN   DB    100 DUP(?)   ;定义100字节空间
TOP     EQU  100
STACK   ENDS
```

1.完整的汇编语言程序框架

```
;代码段
CODE SEGMENT
MAIN  PROC   FAR                    ; 过程定义语句
                                    ; 说明4个段寄存器分别与哪些段有关
        ASSUME  CS：CODE，  DS：DATA
                ES：EXTRA，SS：STACK
START:
        MOV    AX，STACK            ; 设堆栈段寄存器SS: SP
        MOV    SS，AX
        MOV    SP，TOP
        PUSH   DS                   ; DS入栈保护
        SUB    AX，AX               ; AX=0
        PUSH   AX                   ; 段内偏移量“0”入栈
        MOV    AX，DATA             ; AX ← 数据段基址DATA
        MOV    DS，AX               ; DS ← 数据段基址DATA
```

1.完整的汇编语言程序框架

```
        MOV   AX，EXTRA
        MOV   ES，AX      ；ES← 附加段基址EXTRA
                          ；用户要编写的程序内容
        RET               ；返回DOS
MAIN    ENDP              ；MAIN过程结束
CODE    ENDS              ；代码段结束
        END   MAIN        ；整个源代码结束
```

2.关于返回DOS系统的说明

- 1）DOS程序应该能正确返回DOS：程序在DOS下运行结束后，如不能正确返回DOS，则其它程序将无法运行，还会导致死机。因为**DOS是单任务操作系统**。
- 2）返回DOS的3种方法：

  【1】按程序框架设定的方法返回

  【2】执行4CH号DOS功能调用返回

  【3】用INT 20H指令直接返回DOS

2.关于返回DOS系统的说明

- 2）返回DOS的3种方法：

【1】按程序框架设定的方法返回

- 先将主程序定义为一个远过程，再执行3条指令：

```
PUSH    DS
SUB     AX，AX
PUSH    AX
  ┇
RET
```

▶ 将DS和00H推入栈，再执行RET指令，转去执行INT 20H指令，返回DOS。这是返回DOS的常规方法。

2.返回DOS操作系统

【2】执行4CH号DOS功能调用

程序结束前按如下方法使用4CH号DOS功能调用指令，返回DOS。

```
MOV   AX，4C00H      ; AH=4CH，是DOS功能号
                     ; AL通常置为0
INT    21H
```

- 这种方法功能更强，更安全，使用也比较方便，建议使用这种方法返回DOS。

【3】用INT 20H指令

若编写的程序要以.COM文件的形式执行

可用INT 20H指令直接返回DOS。

§4.2 DOS系统功能调用和BIOS中断调用

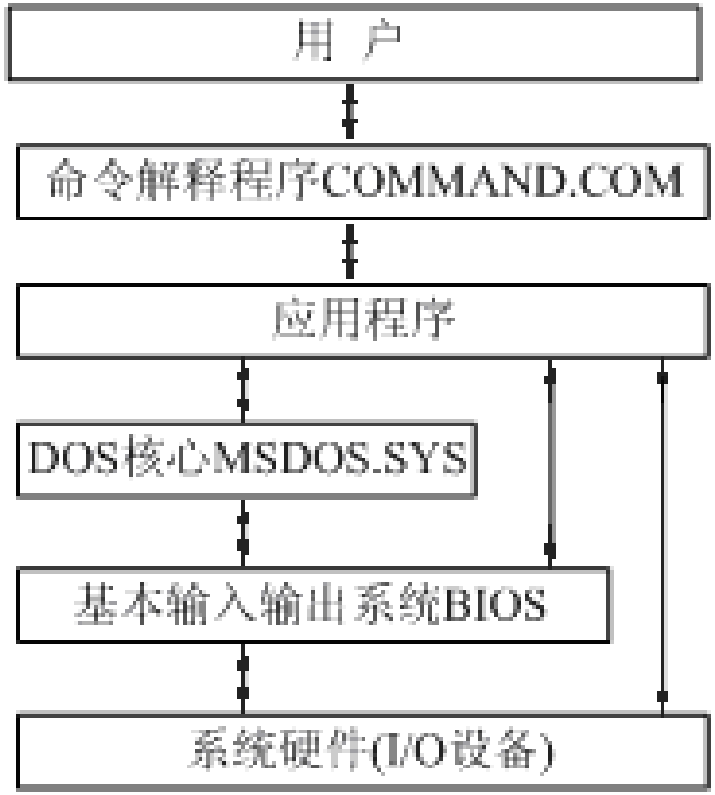
4.2.1 概述

4.2.2 DOS系统功能调用

4.2.3 BIOS中断调用

4.2.1 概述

- 磁盘操作系统DOS采用模块化、层次化结构。



4.2.1 概述

组成DOS的各主要模块的功能：

- 系统硬件

  主要指I/O设备，如CRT显示器、打印机、键盘、硬盘和鼠标等。

- 基本输入输出系统 **BIOS(Basic Input Output System)**

  包含了能**直接与底层硬件**打交道的**设备驱动程序**。此外还包含系统设置信息、开机后自检程序和系统自启动程序。利用**中断调用指令INT *n***，可以直接调用BIOS中的外设驱动程序，实现对主要I/O设备的控制管理。

- DOS核心 **MSDOS.SYS**
  - 该模块以BIOS为基础的一组的服务程序；
  - 称为**DOS系统功能调用**；
  - 用**INT 21H**指令来调用，以**功能号**来区分不同的服务程序；
  - **功能比BIOS更齐全、更完整**。
- 命令处理程序 **COMMAND.COM**

  它执行后，出现**DOS命令提示符“>”**；

  它接收用户命令，进行分析处理后，让机器执行各种应用程序，并在CRT上显示执行结果。

4.2.1 概述

4.2.2 DOS系统功能调用

4.2.3 BIOS中断调用

4.2.2 DOS系统功能调用

1. 中断处理程序分类

◆软件中断：**INT *n***指令，***n* =00~FFH**，256个中断；

- ***n*=00~04H**——为专用中断，处理除法错、单步、不可屏蔽中断NMI、断点、溢出中断；
- ***n*=10H~1AH、2FH、31H、33H**——为**BIOS中断**，即保存在系统**ROM BIOS**中的**BIOS功能调用**。
- ***n*=20H~2EH**——为**DOS中断**，即**DOS系统功能调用**，可对显示器、键盘、打印机、串行通信等字符设备提供输入输出服务。

1. 中断处理程序分类

◆DOS系统功能调用（*n*=**20H~2EH**）

- *n*=**20H**为**程序结束中断**，利用INT 20H中断可返回DOS操作系统。
- *n*=**21H**则为功能最强大的DOS中断，它**包含了很多子功能**，给每个子功能程序赋一个编号，称为**功能号**，调用前要**送到AH寄存器**中。

2. DOS系统功能调用方法

1）**功能号**送到**AH**寄存器中，AH=**00~6CH**。

2）**入口参数**送到指定的寄存器中（不带参数则不用设置）。

3）执行**INT 21H**指令。

4）得到**出口参数**，或将结果显示在CRT上。

3. DOS系统功能调用举例

1）DOS键盘功能调用

- 利用DOS功能调用，可将读入的键值送进AL，并显示在CRT上，或检查是否有键压下等，还可将从键盘输入的一串字符输入到内存缓冲区中。

例4.18　DOS功能调用1，等待从键盘输入一个字符。

```
MOV   AH，01 H     ; AH 功能调用号01H
INT     21H        ; AL 读入键值，并显示该字符
```

- 若有键压下，读入键值，并检查是否为Ctrl-Break键?若是，自动调用INT 23H中断，执行退出命令；否则将键值送入AL，并显示该字符。

3. DOS系统功能调用举例

例4.19　交互式程序中，用户键入字母键Y或N，分别转入不同的程序去处理，并在CRT上显示键入字符；若按了Ctrl-Break，则结束程序，否则继续等待。

```
GET_KEY：
   MOV       AH，01H      ; AH 功能调用号01H
   INT 21H                ; AL← 读入键值
   CMP       AL，  ‘Y’    ; 键值是Y吗？
   JE        YES          ; 是，转YES
   CMP       AL，  ‘N’    ; 不是Y，是N吗？
   JE        NO           ; 是，转NO
   JNE        GET_KEY     ; 不是N，返回继续等待
YES：          ┇          ; 按Y键的处理程序
NO：           ┇          ; 按N键的处理程序
```

4.2.1 概述

4.2.2 DOS系统功能调用

4.2.3 BIOS中断调用

4.2.3 BIOS中断调用

- 在80X86微型计算机中，从内存地址**0FE000H**开始的**8KB**存储空间中，用**EPROM**固化了ROM BIOS程序。（现代的PC机用的是**EEPROM**）

- ROM BIOS模块包含了系统加电自检程序、引导装入程序、基本**I/O**设备驱动程序以及接口控制等功能模块，它们以**中断服务程序**的形式向程序员开放。

- 关于**BIOS** 和**DOS** 调用的几点说明：
  - 有些DOS系统功能调用和BIOS中断调用能完成同样的功能。例如，要打印一个字符，可以用INT 21H的5号DOS功能调用，也可用BIOS的INT 17H的0号中断调用。
  - BIOS更接近硬件，使用起来要复杂一些，应尽量使用DOS系统功能调用。
  - 有些情况下，必须使用BIOS中断调用。例如，INT 17中断的2号调用为读打印机状态，DOS功能调用无这种功能，只能使用BIOS中断调用。

◆ ROM BIOS中断调用的方法

1）功能号送**AH**

2）设置入口参数

3）执行**INT *n***指令（*n*=**10H~1AH、2FH、31H、33H**）

4）分析出口参数及状态

§4.3 汇编语言程序设计方法与实例

引子—结构化程序设计方法

- 汇编语言程序设计采用**结构化程序设计方法。**

◆每个程序**只有一个入口**，必须要有出口，中间内容**不能含有死循环**语句。

◆程序都按照**顺序结构**、**条件分支结构**和循环结构等3种基本结构进行构建。

◆设计时先考虑总体、全局目标，再考虑细节、局部问题，把**复杂问题分解**为一个个**模块**或**子目标**，一步步进行设计。

◆将这些基本结构、子模块合理组合起来，就可构成一个大的程序。

引子—结构化程序设计方法

- 编程时要在程序行上**适当加注释**。这样设计出来的程序层次分明，结构清楚，可读性强，便于调试。
- 编写较复杂的程序时，**一般应先画出程序流程图**，将设计步骤细化，再按流程图设计编写程序。

4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用

4.3.1 顺序结构程序设计

- 顺序结构程序也称为简单程序，这种程序按指令排列的先后顺序逐条执行。

例4.33　编写显示一个笑脸字符在显示器上的程序，程序命名为HAPPY.ASM。

```
PROG1    SEGMENT
         ASSUME  CS：PROG1      ; 只有1个代码段
START:   MOV     DL，1          ; DL← 要显示字符
                                ; 的ASCII码
         MOV     AH，2          ; AH ←功能号2
         INT     21H            ; 显示笑脸符
         MOV     AX，4C00H
         INT     21H            ; 返回DOS
PROG1    ENDS
         END     START
```

4.3.1 顺序结构程序设计

例4.34 由人机对话从键盘输入1个10进制数（0~9），查表求键入数字的平方值，存入AL寄存器中，并显示有关的提示信息。试编写汇编语言程序。

程序如下：

```
DATA  SEGMENT
        TABLE  DB  0, 1, 4, 9, 16, 25, 36, 49, 64, 81
                                                ; 数字0~9的平方值
        BUF   DB ‘Please input a number(0~9):’,0DH,0AH, ‘$’
                                                ; 提示信息
DATA  ENDS
CODE SEGMENT
        ASSUME   CS：CODE，DS：DATA
START:
        MOV   AX，DATA
        MOV   DS，AX                            ; 设置DS
        MOV   DX，OFFSET  BUF
                    ; 设置DX，使字符串首地址=DS：DX
        MOV   AH，9H                            ; 9号DOS功能调用
        INT    21H                              ; 显示提示信息
```

```
        MOV   AH，1          ; 1号功能调用，等待键入字符
        INT    21H           ; AL← 键入数字的ASCII码
        AND   AL，0FH        ; AL ←截下数字值
                             ;（表内元素序号）
        MOV   BX，OFFSET  TABLE
                             ; BX指向表头地址TABLE
        MOV   AH，0          ; AX寄存器高字节清0
        ADD   BX，AX
              ; 表头地址+键入数字(AL)，结果存入BX
         MOV  AL，[BX]       ; 查表求得平方值

        MOV   AX，4C00H
        INT    21H           ; 返回DOS
CODE    ENDS
        END   START
```

4.3.1 顺序结构程序设计

例4.35　在存储单元A1和A2中，各存有一个2字节的无符号数，低字节在前，高字节在后。编程将两数相加，结果存入SUM单元，也要求低字节在前，高字节在后，进位存入最后一个字节单元。

```
DATA   SEGMENT
       A1      DB     56H，78H          ；数A1
       A2      DB     4FH，9AH          ；数A2
       SUM     DB     3 DUP(0)
                                ；存两数相加之和，考虑进位位
DATA   ENDS

CODE   SEGMENT
       ASSUME  CS：CODE，DS：DATA
```

```
BEGIN：
        MOV   AX，DATA
        MOV   DS，AX              ；设置数据段基址
        MOV   BX，0               ；BX为地址指针,初值清0
        CLC                       ；进位位清0
        MOV   AL，A1 [BX]         ；取低字节A1
        ADC   AL，A2 [BX]         ；与A2低字节相加
        MOV   SUM [BX]，AL        ；存入SUM单元(低字节)
        INC    BX                 ；调整指针
        MOV   AL，A1 [BX]         ；取高字节相加
        ADC   AL，A2 [BX]
        MOV   SUM [BX]，AL        ；存高字节
        JNC    STOP               ；无进位，转STOP
```

```
        INC    BX                    ; 有进位
        MOV  AL，0
        INC    AL
        MOV  SUM [BX]，AL     ; 进位存入SUM+2单元
STOP：MOV  AX，4C00H
        INT    21H
CODE   ENDS
END    BEGIN
```

4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

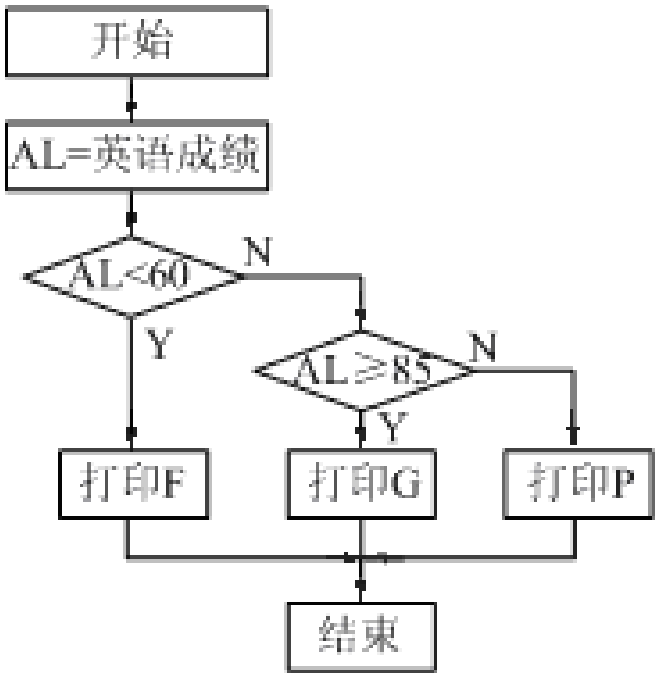
4.3.5 过程调用

4.3.2 分支程序设计

- 要求程序根据不同条件选择不同的处理方法，即程序处理步骤中出现了分支，应根据某一特定条件，选择其中一个分支执行。

【例4.36】

设某学生的英语成绩已存放在AL寄存器中，如果分数低于60分，则打印F，如高于等于85分，则打印G，否则打印P。这就是一个分支程序。

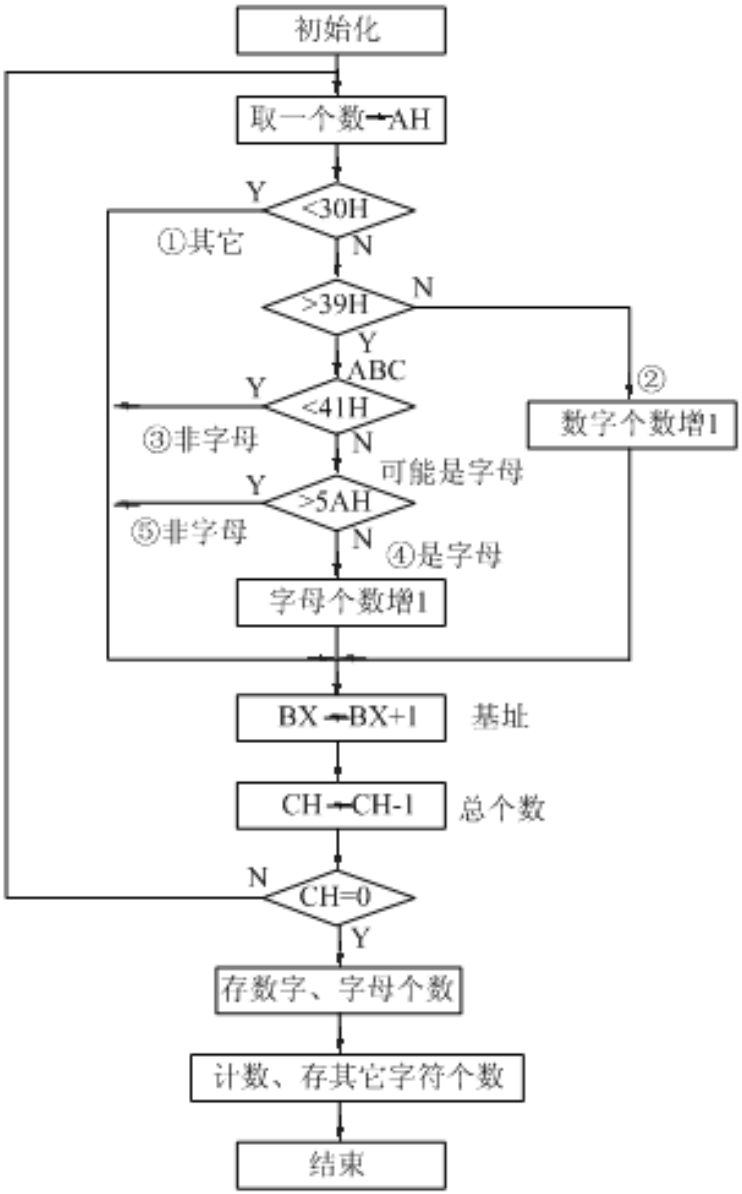


4.3.2 分支程序设计

例4.37 在存储器中以首地址BUF开始存有一串字符，字符串个数用COUNT表示。要求统计数字0~9、字母A~Z和其它字符的个数，并分别将它们的个数存储到NUM开始的3个内存单元中去。

- 在ASCII码表中，数字0~9的ASCII码为30H~39H，大写字母A~Z的ASCII码为41H~5AH，其余值为其它字符或控制符的ASCII码值。

4.3.2 分支程序设计



- 数字个数存放在**DL**中，字母个数存放在**DH**中。

```
DATA      SEGMENT
BUF       DB  '+36', 'PRINT', 'abc',  '2A0CH', '#'
                                                    ; 一串字符
COUNT   EQU  $-BUF                  ; COUNT=字符总个数
NUM       DB            3 DUP(?)    ; 先后存放存数字、字母
                                    : 和其它字符个数
DATA      ENDS
;
CODE      SEGMENT
          ASSUME CS：CODE，DS：DATA
START：   MOV  AX，DATA
          MOV  DS，AX               ; 设置数据段
          MOV  CH，COUNT            ; CH ←数组长度
          MOV  BX，0                ; BX为基址指针,初值清0
          MOV  DX，0
                    ; DH 数字个数，DL字母个数，初值清0
```

```
LOOP1：MOV   AH，BUF [BX]    ; AH ←取一个数
       CMP   AH，30H         ; <30H?
       JL    NEXT            ; ①是，转
       CMP   AH，39H         ; >39H?
       JG    ABC             ; 是，转
       INC   DH              ;②否，数字个数增1
       JMP   NEXT
ABC：  CMP   AH，41H         ; <41H?
       JL    NEXT            ; ③是，非字母，转
       CMP   AH，5AH         ; >5AH?
       JG    NEXT            ; ⑤是，非字母，转
       INC   DL              ; ④否，字母个数增1
NEXT： INC   BX              ; 基地址指针加1
       DEC   CH              ; 字符串长度减1
       JNZ   LOOP1           ; 未完，取下一个数
```

例4.37

```
        MOV     NUM，DH          ；已完，存数字个数
        MOV     NUM+1，DL        ；存字母个数
        MOV     AH，COUNT
        SUB      AH，DH
        SUB      AH，DL           ；计算出其它字符个数
        MOV     NUM+2，AH        ；存其它字符个数
        MOV     AX，4C00H
        INT      21H
CODE    ENDS
END     START
```

4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用

4.3.3 循环结构程序

- 要求某段程序反复执行多次，直到满足某些条件时为止，这种程序称为循环结构程序。

- 在循环程序中，常用计数器（如CX寄存器）来控制循环次数。

4.3.3 循环结构程序

例4.38　在一串给定个数的数据中寻找最大值，存放到MAX存储单元中。

```
DATA    SEGMENT
BUF     DW   1234H, 3200H, 4832H, 5600H
COUNT   EQU  ($-BUF)/2   ;字数据个数（循环次数）
MAX     DW   ?           ;存最大值
DATA    ENDS

STACK   SEGMENT  'STACK'
STAPN   DB     100  DUP(?)
TOP     EQU    $-STAPN
STACK   ENDS
```

4.3.3 循环结构程序

```
CODE    SEGMENT
MAIN    PROC   FAR
        ASSUME  CS: CODE,  SS: STACK
START:  MOV     AX, STACK
        MOV     SS, AX
        MOV     SP, TOP
        PUSH    DS
        SUB     AX, AX
        PUSH    AX
        MOV     AX, DATA
        MOV     DS, AX
        MOV     CX, COUNT      ; CX← 个数
        LEA     BX, BUF        ; BX ← BUF的偏移地址
        MOV     AX, 0          ; AX ←0
```

4.3.3 循环结构程序

```
AGAIN：  CMP   AX, [BX]    ; AX与后取的数比较
         JGE   NEXT        ; 如AX中数大于等于后者,则转
         MOV   AX, [BX]    ; 如后取的数大，则将其送AX
NEXT：   INC   BX          ;修改地址指针
         INC   BX
         DEC   CX          ;循环次数减1
         JNZ   AGAIN       ; 没处理完，转（循环操作）
         MOV   MAX,AX
         RET               ; 返回DOS
MAIN     ENDP              ; 处理完，结束
CODE     ENDS
         END   MAIN
```

4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用

4.3.4 代码转换程序

- 不同数据形式之间转换

4.3.4 代码转换程序

例4.41　将AL寄存器中的二进制数转换成非压缩BCD数，存入AX中，再转换成ASCII码后在CRT上显示。

4.3.4 代码转换程序

```
BIN_ASC  PROC  NEAR
         MOV    AH, 0
         MOV    BL, 10        ; 除数
         DIV    BL            ; AL ←商(9),  AH ← 余数(8)
         XCHG   AH, AL        ; AX=0908H(非压缩BCD数)
         ADD    AX, 3030H     ; AX=3938H（ASCII码）
         MOV    CX, AX        ; CX ← 3938H
         MOV    DL, CH
         MOV    AH, 2
         INT    21H           ; 显示9
         MOV    DL, CL
         MOV    AH, 2
         INT    21H           ; 显示8
         RET
BIN_ASC  ENDP
```

4.3.4 代码转换程序

例4.42　编写将BX中的二进制数转换成16进制数，并在显示器上显示的子程序。

4.3.4 代码转换程序

```
; 程序入口已为BX存入了一个二进制数。
BIN_HEX     PROC NEAR
            MOV  CH, 4
; 转换后产生4个16进制数字（大循环次数）
ROTATE:     MOV  CL, 4          ; 小循环次数（左移4次）
            ROL  BX, CL         ; 对BX左移4次
            MOV  AL, BL         ; AL ←BL
            AND  AL, 0FH        ; 截得一个16进制数字
            ADD  AL, 30H        ; 加30H，转换成ASCII码
            CMP  AL, 3AH        ; 与‘A’比，>9?
            JL   DISPLAY        ; ≤9，转显示
            ADD  AL, 7H         ; >9，将数字0AH~0FH
                                ; 转换成ASCII码
```

4.3.4 代码转换程序

```
DISPLAY：   MOV   DL, AL        ; DL←待显数字ASCII码
            MOV   AH, 2
            INT   21H           ; 显示DL中数字
            DEC   CH            ; 4个数字都显示完？
            JNZ   ROTATE        ; 没有，转大循环
            RET                 ; 显示完，退出
BIN_HEX     ENDP
```

4.3.1 顺序结构程序设计

4.3.2 分支程序设计

4.3.3 循环结构程序

4.3.4 代码转换程序

4.3.5 过程调用

4.3.5 过程调用

- 过程或子程序
- 过程调用
- 子程序嵌套

例4.46　内存中有两个数组ARY1和ARY2，数组长度为20和10，要求编写一个程序，分别累加两个数组的值，存入SUM1和SUM2开始的单元中，低字节在前，高字节在后。

```
;数据段
DATA    SEGMENT                ;数据段
ARY1    DB  20 DUP(?)          ;数组1，20个随机数
SUM1    DB  2 DUP(?)           ;存数组1各数相加之和
ARY2    DB  10 DUP(?)          ;数组2，10个随机数
SUM2    DB  2 DUP(?)           ;存数组2相加之和
DATA    ENDS
```

```
; 堆栈段
STACK   SEGMENT   STACK
SPACE   DB     100 DUP(?)
TOP       EQU   $-SPACE
STACK   ENDS

CODE    SEGMENT                    ; 代码段
MAIN    PROC        FAR            ; 主程序
        ASSUME  CS: CODE, DS: DATA, SS: STACK
BEGIN:
        MOV     AX, STACK
        MOV     SS, AX
        MOV     AX, TOP
        MOV     SP, AX
        PUSH    DS
        SUB     AX,AX
        PUSH    AX
```

```
        MOV   AX, DATA
        MOV   DS, AX
        LEA   SI, ARY1          ; 转子过程前入口参数，
                                ; SI←ARY1首地址
        MOV   CX, 20            ; CX← ARY1长度
        MOV   BX, OFFSET SUM1 ; BX←和单元首址
        CALL  SUM               ; 转子过程，求数组1之和
        LEA   SI, ARY2          ; 转子过程前入口参数
        MOV   CX, 10
        MOV   BX, OFFSET SUM2
        CALL  SUM               ; 转子过程，求数组2之和
        RET                     ; 返回DOS
MAIN    ENDP                    ; 主程序结束

; 子程序SUM
```

```
SUM   PROC    NEAR          ;求和子过程SUM
      XOR     AL, AL        ;AX清0，CF标志清0
      MOV     AH, 0         ;AH存进位，初值清0
LOOP1:
      ADC     AL, [SI]      ;数组中取一元素
                            ;带进位累加到AL
      ADC     AH, 0         ;进位累加到AH中
      INC     SI            ;修改地址指针
      DEC     CX
      JNZ     LOOP1         ;未完，继续
      MOV     [BX], AL      ;已处理完，存和数
      MOV     [BX+1], AH    ;存进位累加值
      RET
SUM   ENDP                  ;SUM子过程结束

CODE  ENDS
      END   MAIN            ;整个程序结束
```