# 第4章
# 汇编语言及其程序设计(I)

# 4.1汇编语言概述

## ❖汇编语言

是一种面向CPU指令系统的程序设计语言，采用指令助记符来表示操作码和操作数、用符号地址表示操作数地址。

用汇编语言编写的程序能够利用硬件系统的特性，直接对位、字节、字寄存器、存储单元、I/O端口等进行处理，同时也能直接使用CPU指令系统及其提供的各种寻址方式编制出高质量的程序。

这种程序占用内存空间少，执行速度快。

## ❖ 汇编程序

是将汇编语言源程序翻译成机器能够识别和执行的目标程序的一种系统程序。

- 能够根据用户的要求自动分配存储区域，包括程序区、数据区、暂存区；
- 自动把各种进制数转换成二进制数，把字符转换成ASCII码，计算表达式的值等；
- 自动对源程序进行检查，给出错误信息等。

具有这些基本功能的汇编程序一般称为基本汇编(Assembler，ASM)。

包含全部ASM的功能，并还增加了伪指令、宏指令、结构、记录等高级汇编语言功能的汇编程序称为宏汇编(MacroAssembler，MASM)。

目前PC中大多采用宏汇编程序。

## ❖ 汇编语言语句的种类

- **指令性语句：** 能产生目标代码，CPU可以执行的能完成特点功能的语句；
- **指示性语句：** 不产生目标代码，仅在汇编过程中告诉汇编程序应如何完成数据定义、符号定义、段定义、存储器分配、过程定义、程序结束等操作；
- **宏指令语句：** 可以多次使用的同一个程序段定义为一条宏指令、一条宏指令包含若干条指令，当一条宏指令作为语句出现，该语句称为宏指令语句。调用时可简单地用宏指令名来代替。在汇编时，遇到宏指令体中的指令来代替这条宏指令语句。

## ❖ 汇编语言语句的格式

### 1) 指令性语句的格式

[标号：] 助记符 [操作数1，[操作数2]] [；注释]

其中方括号[ ]内的项可省略；

指令语句中，标号位于指令之前，是用户为程序中某条指令所指定的名称，是指令的符号地址，标号后面必须紧跟“：”。标号代表该指令在存储器中的首地址，标号可以作为转移指令的目标操作数，转移的目标地址可放一标号；

注释是以“；”开始的字符串，仅用于增加源程序的可读性；

【例】A1：MOV AL，10H

标号A1是指令性语句的名字，是该指令第一字符的符号地址；

标号具有3种属性：

- 段属性：表示标号所在代码段的起始地址，即CS值；
- 偏移属性：表示标号所代表的指令相对于代码段起始地址的字节数；
- 类型属性：分FAR、NEAR两种，分别表示标号所代表的指令与使用该标号做目标地址的控制转移指令，处于不同的代码段还是处于同一代码段中。

## 2) 指示性语句的格式

[名字]　伪指令指示符　　操作数1[，操作数2，…，操作数n] [；注释]

其中方括号[ ]内的项可省略；

指示性语句中，名字可以是符号常量名、变量名、过程名、段名，名字后面不能有“：”；位于伪指令之前；

【例】A1　　DB　　10H

变量A1是伪指令语句的名字，A1后面不跟冒号“：”。DB将一个字节10H定义给A1单元

变量名是用户为存放数据的存储单元所指定的名称。变量作为操作数出现在指令中时，相当于直接寻址方式；

具有3种属性：

- 段属性：该单元所在段的起始地址，可以用段寄存器DS、ES、SS、CS表示；
- 偏移属性：该单元相对于段内起始地址的字节数；
- 类型属性：该单元所存放的数据长度，可能是字节(BYTE)、字(WORD)、双字(DWORD)等类型。

### 3) 宏指令语句的格式

宏用MACRO和ENDM来定义，宏名后面不能有“：”。

宏指令名MACRO [形式参数1，形式参数2，…]
语句组
…　　；宏定义体
ENDM

经过宏定义的宏指令可以被调用，调用时使用宏指令名调用该宏定义。

宏调用的格式为:

宏指令名[实际参数1，实际参数2，...]

在需要使用这一程序段的地方只需要写出宏的名字即可，称为宏调用。

# 4.2 汇编语言标识符、表达式和运算符

## 1、标识符

汇编语言语句格式第一个字段是名字字段，可以是标号或变量，这两者又称为标识符。标号和变量可用LABLE和EQU伪指令来定义，相同的标号或变量的定义在同一程序中只能出现一次。

## 2、表达式和运算符

①算术运算符

7个，+、-、*、/、MOD(求余)、SHL(左移)、SHR(右移)

②关系运算符

6个，EQ、NE、LT、GT、LE、GE;

都是双操作数运算符，运算对象只能是两个性质相同的项目;

关系运算的结果只能是关系成立或不成立两种情况。成立时运算结果为1，否则为0。

③逻辑运算符

4个，AND(与)、OR(或)、XOR(异或)、NOT(非)(单操作数运算符)；

作用是对操作数进行按位操作。

与指令系统中的逻辑运算指令不同，运算后产生一个逻辑运算值，供给指令操作数使用，它不影响标志位。

## ④分析运算符

5个，SEG、OFFSET、TYPE、SIZE、LENGTH；对存储器地址进行运算；可将存储器地址的段、偏移量和类型3个重要属性分离出来，返回到所在位置做操作数使用。前3个运算符对变量、标号有效，后2个仅对变量有效。分析运算符的操作对象：必须是存储器操作数，即变量、标号。

### 1) SGE运算符

取段地址运算符，该运算符返回变量或标号所在段的段地址。

格式：SEG　变量名或标号名

【例】MOV　SI，SEG DATA；

SI←变量DATA的段地址

## 2) OFFSET运算符

取段内偏移量符，该运算返回变量或标号所在段的段内偏移量。

格式：OFFSET　　变量名或标号名

【例】MOV　BX，OFFSET BUF；

BX←标号BUF的偏移地址

## 3) LENGTH运算符

格式：LENGTH　　变量

LENGTH运算符用来回送分配给该变量的单元数。

当变量是用重复数据操作符DUP定义的，则返回DUP前面的数值(即重复次数)；

如果没有DUP说明，则返回值总是“1”。

4) TYPE运算符

格式：TYPE 变量名或标号名

取类型属性运算符，该运算返回变量或标号的段类型属性。若运算对象是标号，则返回标号的距离属性值，标号NEAR、FAR的类型值TYPE分别为-1、-2；

若运算对象是变量，则返回比例类型所占字节数。变量类型分别是BYTE、WORD、DWORD、QWORD和TWORD的类型值TYPE分别为1、2、4、8和10个。

5) SIZE运算符

格式：SIZE 变量

该运算符返回变量所占的总字节数，

即：SIZE = TYPE × LENGTH

⑤合成运算符

6个，“：”、PTR、THIS、SHORT、HIGH、LOW。对已定义的单个操作数、重新生成段基址、偏移量相同而类型不同的新操作数。PTR、THIS对存储单元、变量、标号有效。

1) “：”运算符

格式：段超越前缀：变量或地址表达式

用来给变量、标号或地址表达式临时指定一个段属性。

【例】MOV　CX，ES：[3000H]

将附加数据段ES中偏移地址为3000H字存储单元的内容送CX;

若没有段超越前缀ES，默认是将数据段DS中的偏移地址为3000H的字存储单元的内容送CX。

## 2) PRT运算符

格式：类型 PRT 表达式

赋予变量或地址表达式一个指定的“类型”属性

【例】DATA DB，12H，34H

MOV AX，WORD PTR DATA

在第一条语句中，DATA被定义为字节变量；

在第二天语句中，DATA被临时指定为字类型。

### 3) THIS运算符

格式：THIS　类型

EQU与THIS连用，给指定变量、标号定义新的类型或距离属性，与它下一个数据定义语句的段地址和偏移地址相同。

【例】　DATA　　EQU THIS　　BYTE

　　DATB　　DW　2233H

由EQU　THIS定义的字节变量DATA，与它下一个的字变量DATB的段地址和偏移地址完全相同，执行指令MOV　　AL，DATA后，AL的值是33H

## 4)　HIGH和LOW

格式：HIGH ／LOW　变量或标号

称为字节分离运算符，将一个16位的数或表达式的高字节和低字节分离出来

HIGH从中分离出高位字节，LOW分离出低位字节。

```
例      K1      EQU  0ABCDH
        K2      EQU  1234H
                MOV     AH，HIGH  K1
                MOV     BL，LOW  K2
汇编时，计算表达式形成指令为：
                MOV     AH，0ABH
                MOV     BL，34H
```

## 5)　SHORT

格式：SHORT 标号

用来修饰JMP指令中跳转地址的属性，指出跳转地址是在下一条指令地址的-128～+127个字节范围之内。

```
例      L1:     JMP     SHORT L2
                .........
        L2:     MOV     AX，0
```

## 3、运算符的优先级别

汇编过程中，汇编程序先计算表达式的值，然后再翻译指令。

在计算表达式的值时，如果一个表达式同时具有多个运算符，则按以下规则进行运算：

1) 优先级高的先运算，优先级低的后运算；
2) 优先级相同时，按表达式中从左到右的顺序运算；
3) 括号可以提高运算的优先级，括号内的运算总是在相邻的运算之前进行。

## 4、汇编语言源程序结构

汇编语言源程序是分段结构形式。

一个汇编语言源程序有若干个逻辑段组成，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。

通常，一个汇编语言源程序由**数据段、堆栈段和代码段3个逻辑段组成**。

1) 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、变量等程序需要对其进行操作的数据；

2) 堆栈段

用来在内存中建立一个适当的堆栈区，以便在中断处理、子程序调用时使用。堆栈段一般设定几十字节至几千字节。若太小可能导致程序执行中的堆栈溢出错误;

3) 代码段

包括许多以符号表示的指令，其内容就是程序要执行的指令。作为汇编源程序的主模块，下面几部分不可缺少：

①必须用ASSUME伪指令告诉汇编程序，某一段地址应该放入哪一个段寄存器，这样才能确定段中各项的偏移量;

②DOS的装入程序在执行时，将把CS初始化为正确的代码段地址，把SS初始化为正确的堆栈段地址，在源程序中不需要再对它们进行初始化；

因装入程序已经将DS寄存器留作他用，这是为了保证程序段在执行过程中数据段地址的正确性，所以在源程序中应有以下两条指令，对DS进行初始化：

```
MOV     AX，DATA                ；
MOV     DS，AX                  ；初始化DS
```

指令中DATA为用户名设定的数据段段名。

③在DOS环境下，通常采用DOS中断功能调用的4CH使汇编语言返回DOS，即

```
MOV     AH, 4CH                 ；返回DOS
INT     21H
```

如果不是主模块，这两条指令可以不用

```
DATA SEGMENT                                    ; 定义数据段
     BUF  DB   23H，16H，08H，20H               ; 初始化变量
     CN   EQU  '$'-BUF
DATA ENDS
STRCK     SEGMENT                               ; 定义堆栈段
          STA  DB   10    DUP(?)
          TOP  EQU  '$'-BUF
STACK     ENDS
CODE      SEGMENT                               ; 定义代码段
          ASSUME   CS：CODE，DS：DATA，SS：STACK
START:    MOV AX，DATA                          ; 汇编开始
          MOV DS，AX                            ; 初始化DS
          MOV BX，OFFSET     BUF  ; 取数据区偏移地址
          MOV CX，CN                            ; 设定数据个数
          ……
          …….
          MOV AH, 4CH                           ; 返回DOS
          INT  21H
CODE      ENDS
          END       START                       ; 汇编结束
```

# 4.3 汇编语言程序上机过程

## 1、汇编语言的工作环境

❖ 硬件环境

8086汇编语言程序对机器硬件要求不高，一般多在PC及其兼容机上运行，要求计算机具有一些基本配置就可以。

## ❖软件环境

指支持汇编语言程序运行和帮助建立汇编语言源程序的一些软件，包括以下几个方面：

1) DOS操作系统：汇编语言程序的建立和运行都是在DOS操作系统的支持下进行的；
2) 编辑程序：用来输入、建立或编辑汇编语言源程序；
3) 汇编程序：可用宏汇编MASM.EXE将源程序汇编成目标程序；
4) 连接程序：使用LINK.EXE进行程序的连接；
5) 调试程序：一种辅助工具，采用动态调试程序DEBUG.COM帮助编程者进行程序调试。

## 2、汇编语言上机步骤

1) 用编辑程序EDIT.COM建立扩展名为.ASM的汇编语言源程序文件；
2) 用汇编程序MASM.EXE将源程序文件汇编成用机器码表示的目标程序文件，扩展名为.OBJ；
3) 若在汇编过程中出现语法错误，根据错误的信息提示(位置、类型、说明)，可用编辑软件重新调入源程序进行修改；
4) 汇编无误时采用连接程序LINK.EXE把目标文件转化成可执行文件，扩展名为.EXE；
5) 生成可执行文件后，在DOS命令状态下直接输入文件名执行该文件，也可采用调试程序DEBUG.COM进行相应处理。

## ❖ 汇编语言程序设计的基本步骤

1) 分析问题，建立数学模型：
2) 确定算法或解题思路：
3) 画出流程图：
4) 存储空间和工作单元初始化：8086存储器结构要求存储空间分段使用，因此要分别定义数据段、堆栈段、代码段和附加段；
5) 编写汇编语言源程序：
6) 静态检测：
7) 动态调试：

# 4.4 程序的基本结构与基本程序设计

每一个结构只有一个入口和一个出口，3种结构的任意组合和嵌套构成了结构化的程序。

## 1、顺序结构

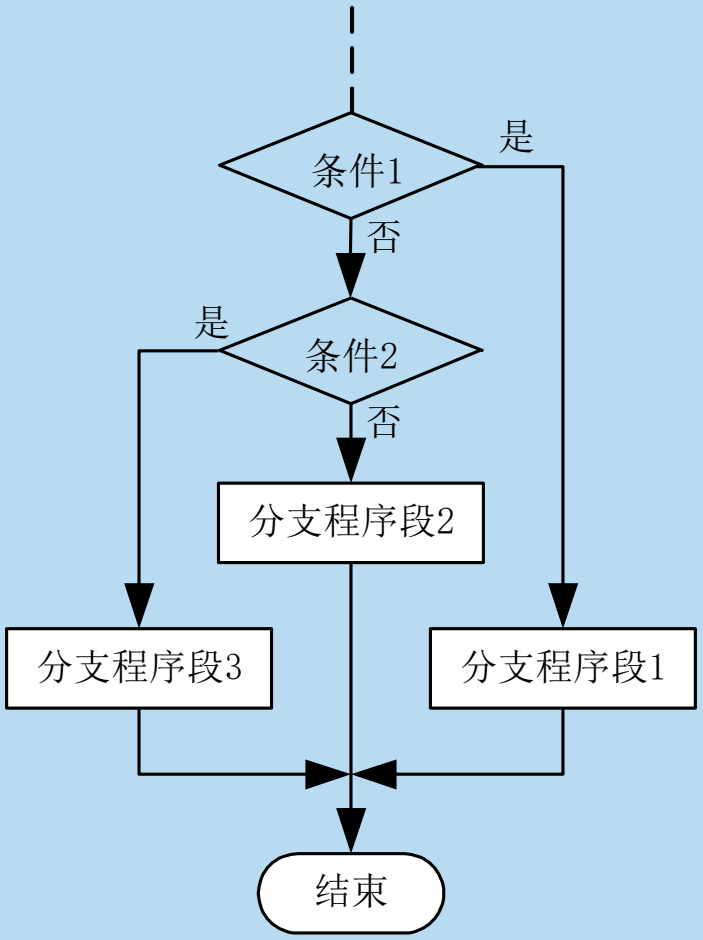
最简单的程序结构，按指令的书写顺序一个语句紧跟一个语句执行，到最后一条指令为止，指令指针IP中的内容呈线性增加。

一个起始框 →

一个或几个执行框 →

一个终止框。

【例1】已知X、Y是数据段中的两个无符号字节数据，要求完成表达式$Z = (X^2 + Y^2)/2$的计算。




# 2、分支/选择结构程序

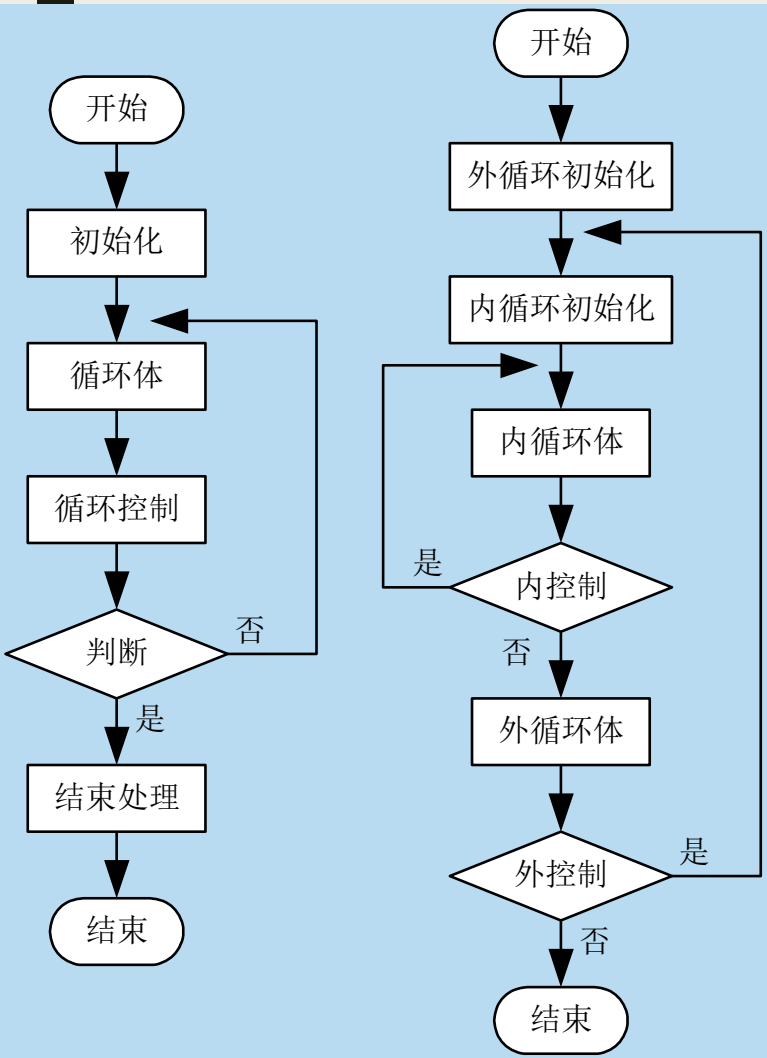
条件的判断是先由执行指令如CMP、TEST后产生的状态标志位，再由条件转移指令Jcc根据标志位的各种情况进行转移，以确定不同的处理过程。分支是通过条件转移指令Jcc和JMP实现的。分支结构有单分支结构、双分支结构、多分支结构。

条件转移指令可以测试下述条件：>、≥、=、≠、<、≤、溢出、未溢出、正、负、奇和偶等。

【例2】设AX和BX中分别存放一个无符号字数据，找出其中小数送入MINUS单元。

【例3】在内存单元BUF中存放有着一个8位带符号二进制数X，假定为-25，根据下列函数关系编写程序求Y，并将结果存入RESULT单元

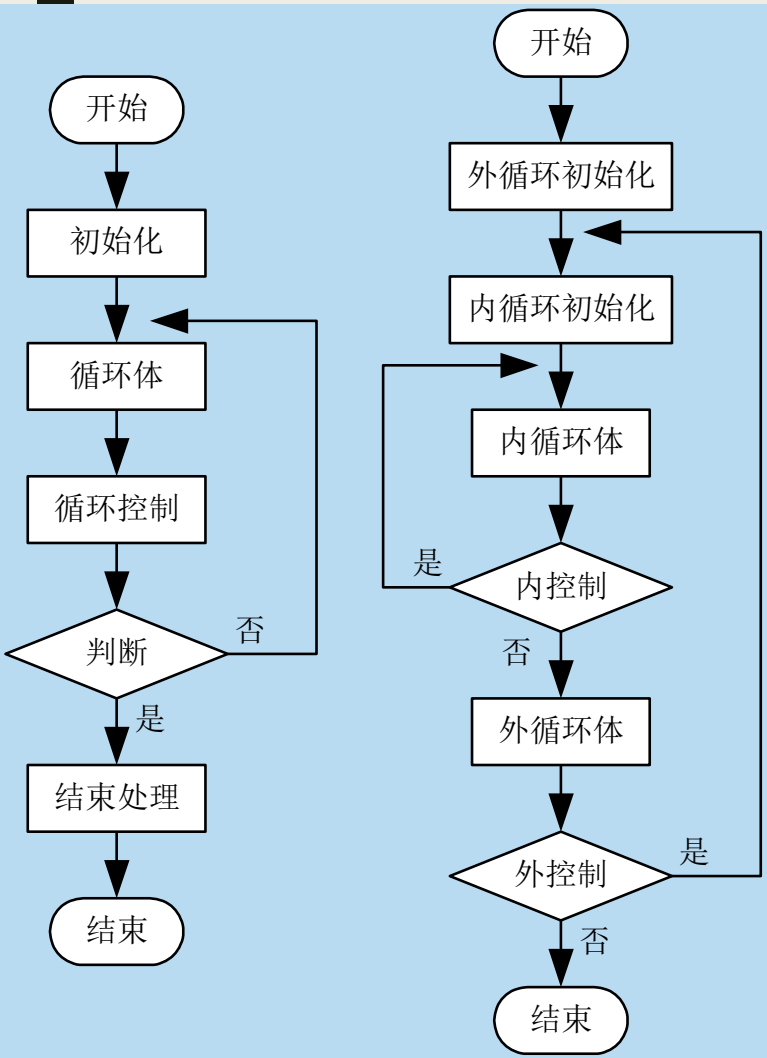
$$Y=\begin{cases}1 & (X>0)\\ 0 & (X=0)\\ -1 & (X<0)\end{cases}$$




# 3、循环程序设计

凡是重复执行的操作均可用循环程序来实现。通常4个部分组成：

1) 初始化部分：
2) 循环体：
3) 参数修改部分：为确保每次循环都能正确运行，必须对计数器的值、操作数的地址指针及控制变量发生有规律的变化；
4) 控制部分：需要选择一个恰当的循环条件保证循环程序按预定的循环次数或某种预定的条件正常循环和结束




循环结构程序的设计关键是循环控制部分，循环控制可以在进入循环之前进行，也可在循环体后进行，于是形成两种循环结构：

①是先执行循环体，再判断循环是否结束；

②是先判断是否符合循环条件，符合则执行循环体，否则退出循环。

常用的控制方法有：

①计数法、

②寄存器终值法、

③条件控制法。

①计数器控制法——当循环次数是已知的情况下，将计数器的初值设置为规定的循环次数，每执行一遍循环体，计数器值-1，减到0，则退出循环，否则进行执行循环体。

【例4】从DATA单元开始连续存放了50个单字节带符号数，编写程序统计这50个数据中负元素的个数，将统计结果存入DATB单元。

②寄存器终值控制法——用一个寄存器存放初值，每执行一次循环体该寄存器的值都按某种规律而有所变化，直到该寄存器值达到某一终值退出循环。

【例5】编程求解1+2+3+...+N < 100时最大的N值，将N值送NUM单元中，同时将1+2+3+...+N的和送SUM单元。

③条件控制法——在循环次数未知的情况，利用本身的结束条件来控制循环结束的方法。

【例6】若自内存STRING单元开始存放若干个ASCII码字符串，以字符‘$’结尾，编制程序将每个字符的最高位作为奇校验位后送入原单元。

# 4、子程序结构

一个完整的独立的程序段，可以多次被其他程序调用，并在这个程序段执行完毕后返回到原调用的程序处。

调用子程序的程序段称为主程序。主程序中调用指令的下一条指令的地址称为返回地址，也称为断点。

子程序用一对过程伪指令PROC和ENDP声明，格式

```
过程名   PROC     [NEAR]/[FAR]  ;  过程开始
.....                            ;  过程体
过程名   ENDP                    ;  过程结束
```

# 4.1 主-子程序调用及功能分析

子程序基本结构包括:

1) 子程序说明：说明子程序的名称、功能、入口参数、出口参数、占用工作单元的情况，明确该子程序的功能和调用方法;

2) 现场保护及恢复：通常采用堆栈操作;

保护现场：子程序执行完毕返回主程序后，为保证主程序按原有的状态继续正常执行，需对寄存器的内容加以保护;

恢复现场：子程序执行完毕后再恢复这些被保护的寄存器的内容；

3) 子程序体：实现响应的子程序功能；

4) 子程序返回：子程序返回语句RET和主程序(调用程序)调用语句CALL一一对应。返回指令用来恢复被中断位置的地址。

# 4.2 子程序的参数传递

主程序向子程序传递输入参数 及 子程序向主程序传递输出参数称为主程序和子程序间的参数传递。

汇编语言中实现参数传递的方法：

1) **寄存器传递：** 适用于传递的参数较少的情况。在调用子程序之前，把参数放到规定的寄存器中，有这些寄存器将参数带入子程序中；执行子程序结束后的结果也是放到规定的寄存器中带回主程序。

2) **堆栈传递：**通过堆栈这个临时存储区来实现参数传递。主程序将入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口参数压入堆栈，主程序从堆栈中取出参数。汇编语言与高级语言混合编程是经常采用。

3) **存储器传递：**若调用程序和子程序在同一程序模块内，采用存储器传递参数是最简单的方法。通常在数据段内定义出要传递的数据变量，也称为数据缓冲区，子程序对这些定义的变量可直接访问。

# 5、DOS与BIOS功能调用程序设计

DOS(Disk Operating System)是PC上重要的操作系统。

DOS功能调用程序入口地址已由系统置入中断向量表中，编写汇编语言程序可用INT中断指令直接调用DOS功能。

中断调用号参看教材表4.5，或附录。

BIOS是BASIC INPUT/OUTPUT SYSTEM的缩写，即基本输入/输出系统，指固化在只读存储器(ROM)中的一组程序，以称ROM BIOS。

# 5.1 DOS功能调用过程

（全部采用INT 21H中断）：

- 由AH给出功能号；
- 根据相应功能的要求，设置入口参数。入口参数是子程序运行所需要的数据，DOS系统功能调用的入口参数通常是放在指定的内部寄存器器中，少数功能调用也可以没有入口参数；
- 执行中断指令INT 21H；
- 分析和使用出口参数。

# 5.2 DOS调用举例

## ①显示一个字符

功能号：02H

入口参数： AH=02H；DL=要显示字符的ASCII码。

出口参数：无

功能：在当前光标位置显示DL中的字符，光标右移。

调用方法示例：

```
MOV         DL，‘A’      ；显示字符’A’
MOV         AH，02H
INT         21H
```

## ②显示一个字符串

功能号：09H

入口参数：AH=09H，DS:DX=欲显示字符串在内存的首地址，且字符串必须以‘$’(24H)作为结束符。

出口参数：无

功能：在当前光标位置，显示由DS：DX所指的、以‘$’结尾的字符串，且光标右移。其中，‘$’不算在显示的字符串之内。

## ③键盘读入一个字符

功能号：01H

入口参数：AH=01H

出口参数：AL=输入字符的ASCII码。

功能：等待从键盘读入一个字符，将其ASCII码送入AL，同时将字符显示在屏幕上。

调用方法：

```
MOV AH，01H
INT  21H
```

说明：输入一个字符后，不需要回车。若只输入回车，则AL=0DH。

## ④从键盘读入一个字符串

功能号：0AH

入口参数：DS：DX=输入缓冲区首地址。

出口参数：无

功能：从键盘读入一个字符串，存放DS：DX所指的缓冲区。

## ⑤与日期时间调用有关

1、设置、读取日期：2BH，2AH；

CX：年份，DH：月份，DL：日期；

AL=00H(有效)，FFH(无效) — 对2BH而言

2、设置、读取时间：2DH，2CH；

CH：小时，CL：分钟，DH：秒，DL：1/100秒；

AL=00H(有效)，FFH(无效) — 对2DH而言

注意参数的设置！

# 5.3 BIOS中断和功能调用

BIOS提供了访问主要I/O设备的服务程序。

BIOS与系统无关；

BIOS中断和功能调用可以提供更低、更接近于硬件的指令；

DOS调用其实是通过调用BIOS完成的。

调用过程与DOS调用过程类似：

设置入口参数，执行中断指令取得服务；

# 5.4 BIOS中断和功能调用

## 显示器输出服务和键盘输入服务:

1) INT 10H：显示器输出

INT 10H包含了与显示器有关的功能，可用来设置显示方式，设置光标大小和位置，显示字符等。

①AH=0AH：显示字符

入口参数：AL=欲显示字符的ASCII码。

②AH=0EH：显示字符

入口参数：AL=欲显示字符的ASCII码。

功能：类似于0AH功能，但显示字符后，光标随之移动，并可解释回车，换行和退格等控制符。

2)INT=16H：键盘输入

①AH=0：从键盘读一键

出口参数：AL=ASCII码，AH=扫描码。

功能：从键盘读入一个键后返回，按键不显示在屏幕上。对于无相应ASCII码的键，如功能键等，AL返回0。

扫描码是表示按键所在位置的代码，有关细节将在以后讨论。

②AH=1：判断是否有键可读

出口参数：若ZF=0，则有键可读，AL=ASCII码，AH=扫描码；否则，无键可读。

③AH=2：返回变换键的当前状态

出口参数：AL=变换键的状态。