1.1 汇编语言基本概念

1.汇编语言源程序格式

汇编语言源程序的结构是分段结构形式,一个汇编语言源程序是由若干个段组成，每个段以 SEGMENT 语句开始，以 ENDS 结束.整个程序的结尾是 END 语句.

汇编语言源程序的段有四种类型:代码段.数据段、堆栈段和附加段。

2.代码段结构框架

(1)定义段(使用 SEGMENTENDS 语句定义)
(2)说明物理段和逻辑段的关系，使用一个或多个 ASSUME 语句实现
(3)装填段寄存器(只装填数据型段寄存器)DS、ES
(4)完成所需功能的程序段
(5)设置返回 DOS 的方法

```
DATA SEGMENT   ; 定义数据段起始语句
.........      ; 定义数据
DATA ENDS      ; 定义数据段终止语句

CODE SEGMENT                  ; 定义代码段起始语句
   ASSUME CS:CODE, DS:DATA    ; 段关系说明
   START: MOV AX,DATA         ; 装填相应的段寄存器
          MOV DS,AX
          .......             ; 完成所需功能的程序段
          MOV AH,4CH          ; 设置返回 DOS
          INT 21H
CODE ENDS                     ; 定义代码段终止语

END  START     ; 程序结束
```

3.汇编语言语句类型

汇编语言源程序中的语句可分为三种类型:

①实指令语句:汇编时会产生目标代码的CPU可执行语句。

②伪指令语句:不会产生目标代码的CPU不可执行语句。其作用是指示汇编程序如何完成数据定义、符号定义、段定义、存储区分配、指示程序结束等。

③宏指令语句:是为简化源程序编写而设计的自定义语句。

4.汇编语言语句格式

[名字] 助记符/定义符 [操作数] [;注释]

①名字实为符号地址。在实指令语句中称为标号:在伪指令语句中称为变量名(还有段名、过程名等），它们都有三个主要属性:段属性、偏移属性和类型属性。

②助记符/定义符:指明该指令的功能。

③操作数:操作的对象。可以是立即数、寄存器、存储器、标号、变量和表达式。

2.1 数据定义伪指令

[变量名] 定义符 操作数[， 操作数，..]

这里的定义符有DB、DW、DD、DQ、 DT等。其功能是定义字节数据(BYTE类型)、字数据(WORD类型)、双字数据(DWORD类型）、四字数据(QWORD类型)、十字节数据(TBYTE类型)等。

而操作数可以是常量，变量、表达式、字符串、?和重复操作符“DUP”

(n DUP(初值[.初值，..))

如: X DB 3 DUP(?) --> X DB ?,?,?

2.2 符号定义伪指令

符号定义伪指令的用途是给一个符号重新命名或定义新的类型属性等。符号包括汇编语言的变量名、标号名、过程名、寄存器名以及指令助记符等。

常用的符号定义伪指令有EQU、= (等号)和LABLE

1. EQU伪指令

格式:名字 EQU 表达式

功能:将表达式的值赋给一个名字，可以用这个名字来代替给定表达式。

表达式可以是一个常数、变量、寄存器名、指令助记符、数值表达式或地址表达式等

2.=(等号)伪指令

格式:名字 = 表达式

功能:与EQU伪指令基本相同，主要区别是“=” 可以对同一个名字重复定义，而EQU伪指令不允许。

3. label伪指令

格式:名字 label 类型

功能:定义名字的类型，名字可以是标号或变量。

标号的类型可以是near或far:变量的类型可以是BYTE、 WORD等。

label伪指令并不占内存单元。

利用label伪指令可以使同一个数据区兼有byte和word两种属性，这样，在以后的程序中可以根据需要分别以字节或字为单位存取其中的数据。

2.3 段定义伪指令
段定义伪指令的用途是在汇编语言程序中定义逻辑段，用它来指定段的名称和范围，并指明段的定位类型、组合类型及类型。

segment/ends 伪指令
格式:段名 SEGMENT [定位类型] [组合类型][类别]
...指令语句或伪指令语句
段名 ENDS

定位类型。说明段的起始边界值(物理地址)。有以下 4 种定位类型:
①BYTE:表示逻辑段从字节的边界开始，即可以从任何地址开始。此时本段的起始地址紧接在前一个段的后面。
②WORD:表示逻辑段从字的边界开始。2 个字节为 1 个字，此时本段的起始地址最低一位为 0， 即段起始地址必须为偶数。

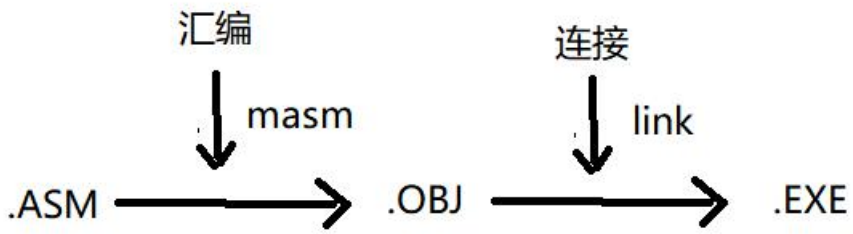
2.4 过程定义伪指令
在程序设计中，经常将一些重复出现的语句组定义为子程序，又称为过程， 可以采用 CALL 指令调用。
过程定义伪指令 proc/endp

过程定义格式:过程名 PROC [NEAR/FAR]
.....
RET
过程名 ENDP

(1)过程名:是编程人员给过程起的名称，是提供给其他程序调用时用的，不能省略。过程名有段地址、偏移地址和类型三个属性。段地址和偏移地址是过程中第 1 个语句的段地址和偏移地址，类型有格式指定。
(2)类型:包括 near 和 far 两种。near 表示过程与调用程序在同一段内: far 表示过程与调用程序不在同一段内。默认为 near.
(3)格式中的 proc 和 endp 是过程定义的关键字，它们前面的过程名必须-一致，且成对出现。
(4) ret 是过程的返回指令，是段内返回还是段间返回由格式中的类型决定。ret 指令控制返回到原来调用指令(call) 的下一条指令。

2.5 段说明伪指令
assume 伪指令作用是告诉汇编程序，将某一个段寄存器设置为存放某一个逻辑段的段地址，即明确指出源程序中的逻辑段与物理段之间的关系。
格式: assume 段寄存器名: 段名[段寄存器名:段名....]
格式中的段寄存器可以是 CS、DS、ES、 SS。

3.1 汇编语言上机过程



1.将汇编语言源程序翻译成目标程序的过程称为 汇编 过程，产生的目标文件的扩展名为 .OBJ
2.汇编语言源程序的语句格式包括名字、助记符、操作数、注释。

3.2 DOS 功能调用

程序员编写汇编程序，不可避免地要和底层的各类设备打交道，如简单的键盘输入、显示器的输出、磁盘文件的读/写或是打印机的打印等，而要完成这些工作需要弄清楚有关设备的结构组成、与 CPU 接口等系列问题，非常繁杂。

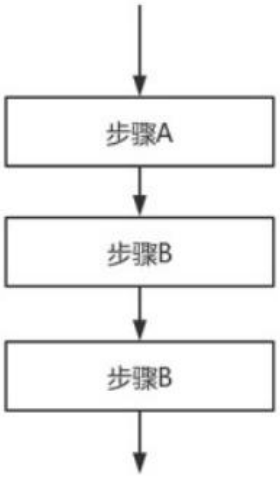
1.如何使用 DOS 系统功能调用(掌握)
在汇编程序中使用系统功能调用一般按以下几个步骤操作:
①根据所需的功能调用设置好相应的调用参数。有部分功能调用不需要调用参数，此步骤可以省略，但大部分功能调用都需要调用参数，则调用前应按要求准备好调用参数，通常将所需参数置入对应的寄存器中。
②将调用的系统子程序的编号即功能号送去 AH 寄存器。
③用" INT 21H"软件 中断指令转到系统子程序的总入口，然后通过分析 AH 的内容，再转向相应的子程序，即调用 DOS 功能子程序。
④限据有关功能调用说明取得返回参数，最后分析和处理返回参数。返回参数也是部分功能调用有部分没有。

例如:

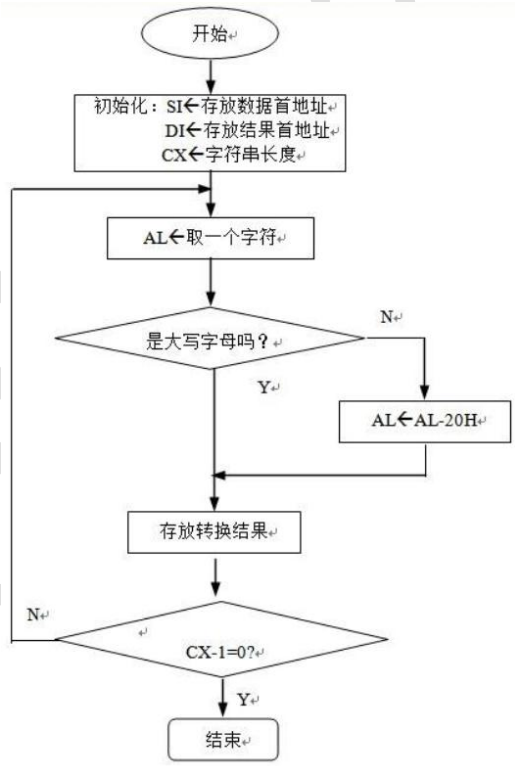
```
MOV  AH,01H        ;提供调用功能号
INT  21H           ; 系统功能调用
```

执行上述指令，将扫描键盘，等待有键按下，一旦有键按下就将键值（相应字符的 ASCII 码值）读入，并送入 al 寄存器，同时将这个字符显示在屏幕上。

4.1 顺序结构程序设计
顺序是是是一种最简单的程序。也称为直线程序。它的执行自始至终按照语句出现的先后顺序进行。



4.2 分支结构程序设计
通常是在两种或两种以上的不同的操作中的其中的一个。



编写分支程序需要解决三个问题，即:判断、定标号和转向。
判断通常是采用检查标志寄存器的标志位来判断条件是否成立。
一次判断形成两个分支，连续的判断则会形成多个分支。一般情况下，需要给每个分支单独编写一段程序，而且在程序段开始处给出标号，以便转向它，这就是定标号。
转向就是指使用“JE/JA/JC/JS/.... 标号”等使指令转向相应的分支程序段。

1、无条件转移指令只有 JMP

2、条件转移指令分为三种：基于算数标志位的转移指令，基于无符号的转移指

令，基于有符号数的转移指令

（1）基于算数标志位的转移指令

① JC CF=1 进/借位转移
② JNC CF=0 没有进/借位转移
③ JO OF=1 溢出转移
④ JNO OF=0 无溢出转移
⑤ JP PF=1 低八位偶数个 1 转移
⑥ JNP PF=0 低八位奇数个 1 转移
⑦ JS SF=1 结果为负转移
⑧ JNS SF=1 结果为正转移

（2）基于无符号的转移指令

① JE = 则转移
② JNE != 则转移
③ JA > 则转移
④ JNA <= 则转移
⑤ JB < 则转移
⑥ JNB >= 则转移

（3）基于有符号数的转移指令

① JE = 则转移
② JNE != 则转移
③ JG > 则转移
④ JNG <= 则转移
⑤ JL < 则转移
⑥ JNL >= 则转移

4.3 循环结构程序设计

重复执行某一段程序，直到某个条件出现为止。

1、循环程序的组成。

一个循环程序通常由四部分组成。

(1) 初始化部分。（启动循环所需的初始参数）
(2) 工作部分。（需要重复执行的程序段）
(3) 修改部分。（主要是修改循环条件）
(4) 控制部分。（根据条件决定是继续循环还是跳出循环）

2、循环控制条件。

（1）用计数控制循环。（适用于已知循环次数的循环）
（2）用条件控制循环。（适用于未知循环次数的循环）

3、循环指令

（1）LOOP CX!=0,则循环
（2）LOOPZ CX!=0,ZF=1,则循环
（3）LOOPNZ CX!=0,ZF=0,则循环