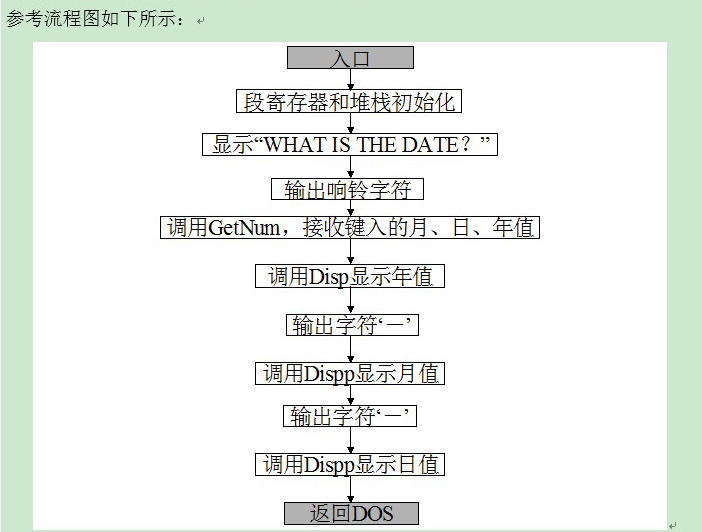
**汇编作业三**

**作业一：打印年月日**

要求：接收年月日信息并进行显示

目的：掌握响铃符方法；掌握年月日的输入方法；掌握过程调用方法

内容：先显示提示信息“WHAT IS THE DATE(MM/DD/YY):”，并响铃依次，然后键盘接收输入的月/年/日信息，并显示



**1 思路分析**

本作业要求使用过程调用实现，以及通过响铃法来实现。过程调用类似于c语言中的函数，在实际的使用中没有遇到太大的困难。所谓响铃法，响铃通常用于引起用户的注意或者在某些情况下表示发生了某种事件。在汇编语言中，通过使用INT 21H中断调用，并将AH寄存器设置为2，可以实现在DOS环境下发出响铃。根据下图的流程图，其实我们完全按照这个步骤来即可，先读取字符串，然后进行处理，分别放到年月日各自的子串中，然后最后按年月日的顺序输出即可实现。

**2 具体实现**

**响铃操作的实现**

在汇编语言中，通过使用INT 21H中断调用，并将AH寄存器设置为2。响铃可以用作用户界面的一部分，向用户传达某种信息。例如，在命令行程序中，如果用户输入了无效的命令或者发生了错误，可以通过响铃来提示用户。在程序执行过程中发生错误时，响铃可以用作一种通知机制，提醒程序员或者用户发生了异常情况。在一些交互式应用程序中，响铃可以用来强调用户需要关注的事项，或者用于实现某种特殊效果。在一些应用中，响铃可以与定时器结合使用，用来提醒用户或者程序在执行某些特定的操作或者定时任务。在调试过程中，响铃也可以作为一种简单的调试信息输出方式，用于指示程序执行到了某个特定的位置或者条件满足时。

部分代码实现如下所示

|  |
| --- |
| Assembly language  ; 响铃操作  MOV AH, 2 ; 响铃  MOV DL, 7  INT 21H ; 用02号功能，输出一个响铃字符 |

**GetNum过程调用的实现**

我们第二步就是需要调用GetNum进行读取输入的月日年。在这里我采用的方法是先设置三个string变量，来分别存放年月日，然后一位一位读取输入字符串的字符，然后分别存放到对应的string中即可。因为我们在输入时需要严格按照“月/日/年”的顺序输出，所以当读到“/”即可结束一个时间的读取。需要注意的是，字符串的结尾必须是“$”，所以我们还要通过getlength，给字符串的最后添加一个“$”。

部分代码实现如下所示

|  |
| --- |
| Assembly language GetNum PROC  INPUT\_M:   MOV AH, 1 ; 一位一位读取  INT 21H  CMP AL, 2FH ; 判断是否是/  JZ GETLENGTH\_M ; 字符串输入结束，转入getlength处  MOV [STRING\_M + BX], AL  INC BX  JMP INPUT\_M  GETLENGTH\_M:  MOV [STRING\_M + BX], '$'  MOV BX, 0  INPUT\_D:   MOV AH, 1  INT 21H  CMP AL, 2FH ; 判断是否是/  JZ GETLENGTH\_D ; 字符串输入结束，转入getlength处  MOV [STRING\_D + BX], AL  INC BX  JMP INPUT\_D  GETLENGTH\_D:  MOV [STRING\_D + BX], '$'  MOV BX, 0  INPUT\_Y:   MOV AH, 1  INT 21H  CMP AL, 0DH ; 判断是否是回车  JZ GETLENGTH\_Y ; 字符串输入结束，转入getlength处  MOV [STRING\_Y + BX], AL  INC BX  JMP INPUT\_Y  GETLENGTH\_Y:  MOV [STRING\_Y + BX], '$'  MOV BX, 0   EXIT:   RET  GetNum ENDP |

**Disp过程调用的实现**

接下来，就是输出即可。需要注意的是我们输入的顺序是月/日/年，但是我们输出需要按照年/月/日的顺序输出，因为直接将结果存放在字符串中，所以直接使用AH 9输出即可

部分实现代码如下

|  |
| --- |
| Assembly language Disp PROC  ; 输出输出提示  MOV AH, 9  MOV DX, OFFSET OUTPUT\_STRING  INT 21H    MOV AH, 9  MOV DX, OFFSET STRING\_Y  INT 21H ;     MOV AH, 2  MOV DL, '-'  INT 21H   MOV AH, 9  MOV DX, OFFSET STRING\_M  INT 21H ;    MOV AH, 2  MOV DL, '-'  INT 21H   MOV AH, 9  MOV DX, OFFSET STRING\_D  INT 21H ;    RET Disp ENDP |

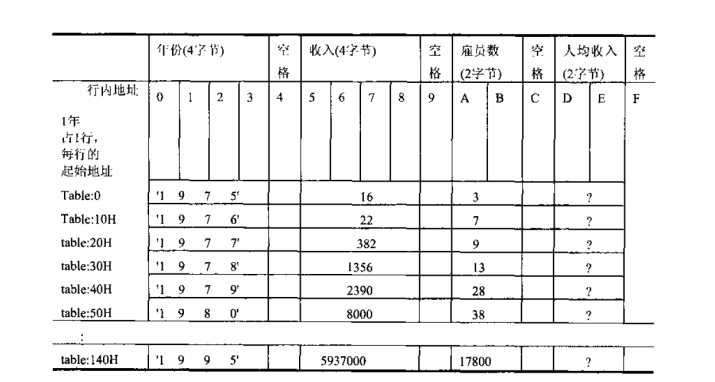
**3 心得体会**

本作业主要是熟悉call过程调用的使用，以及熟悉输入输出操作，在程序编写中没有太大的困难。但是，在开始的时候对响铃法有些不太理解，导致卡在使用响铃法这一步。最后，发现使用与否并不影响结果，便没有过分纠结这个问题。

**作业二：工资计算**

要求：将 data 段中的数据写入 table 段，并计算每年的人均收入（取整），结果一并填入下表所 示的 table 段：

程序中数据已定义好，如下：



data segment

;以下是表示 21 年的 21 个字符串

db '1975','1976','1977','1978','1979','1980','1981','1982','1983'

db '1984','1985','1986','1987','1988','1989','1990','1991','1992'

db '1993','1994','1995'

;以下是表示 21 年公司总收的 21 个 dword 型数据

dd 16,22,382,1356,2390,8000,16000,24486,50065,97479,140417,197514

dd 345980,590827,803530,1183000,1843000,2759000,3753000,4649000,5937000

;以下是表示 21 年公司雇员人数的 21 个 word 型数据

dw 3,7,9,13,28,38,130,220,476,778,1001,1442,2258,2793,4037,5635,8226

dw 11542,14430,45257,17800

data ends

table segment

db 21 dup('year summ ne ?? ')

table ends

**1 思路分析**

本作业的实现主要分为两部分，一是把data中的数据存放到table表中，二是需要打印出table表中的数据，这两个的实现都十分麻烦。对于第一个的实现，需要先搞清楚data中每个数据存放的地址段，然后按照顺序一个字节一个字节存放到table中对应的位置，第一步的实现思路较为清晰。而我本次遇到的主要难点在第二步的实现，我们需要打印table中三种类型的数据，分别是db，dw，dd。打印db的方法很简单，我们只需要一个字符一个字符输出即可，打印dw和dd的过程类似，需要我们转换为十进制后再输出，这里的实现比较复杂。

**2 具体实现**

**2.1 将data中的数据存放到table表中**

首先data中第0-83位存放的是年份，一个年份占4个字节，数据类型是db；第84-167位存放的是收入，一个收入占4字节，数据类型是dd；第168-209存放的是人数，一个人数占2字节，数据类型是dw。然后table中第0-3位存放年份，5-8位存放收入，10-11存放员工数，13-14存放平均收入。

在这基础上，我们即可开始一点一点从data中读取数据，存放到table中去。年份，收入，人数这三个数据可以直接读取，而剩下的人均收入，需要用收入/人数 来计算求得。

这里可以将data段中的数据看成是多个数组，而将table中的数据看成是一个结构型数据的数据，用偏移量SI的方式来读取数据

本程序中这部分的代码由GetNum过程实现，部分代码如下所示

|  |
| --- |
| Assembly language GetNum PROC   MOV CX, 21 ; 循环21次  MOV BX, 0  MOV SI, 0  MOV DI, 0  LOOP\_1:  ; 写入年份  MOV AX, DS:0[SI]  MOV ES:[0], AX  MOV AX, DS:0[SI+2]  MOV ES:[2], AX   ; 写入空格  MOV AL, 20H  MOV ES:[4], AL  MOV ES:[9], AL  MOV ES:[12], AL  MOV ES:[15], AL   ; 写入收入数据  MOV AX, DS:84[SI]  MOV ES:[5], AX  MOV DX, DS:84[SI+2]  MOV ES:[7], DX  ; 由于这里是dword数据类型，需要将其转换为字符串后输出   ; 写入人数  MOV BX, DS:168[DI]  ; 由于这里是word数据类型，需要将其转换为字符串后输出  MOV ES:[10], BX   ; 写入平均收入  DIV BX ; 求平均值  MOV ES:[13], AX   ; 进入下一个循环  MOV AX, ES  INC AX  MOV ES, AX  ADD SI, 4  ADD DI, 2  LOOP LOOP\_1   RET  GetNum ENDP |

**2.2 打印table表中的内容**

总体上来看，第一部分没有太大的困难，但是由于我们不能直观地看到我们在修改之后table段中的结果，在实际上我是先完成了第二步打印table中的内容后，再来实现第一步的。打印的内容主要分为两类，但是在后面的打印中，我发现除了db可以直接打印出来以外，另外三个数据并不能直接打印，而是需要转换为十进制后再输出。

首先是年份的打印，因为年份是db数据类型，一个年份是4字节，直接一位一位打印四个字符即可

部分实现代码如下

|  |
| --- |
| Assembly language INPUT\_1:; 输出年份  MOV DL,[BX+SI]  INT 21H  INC SI  CMP SI,4  JNE INPUT\_1 |

然后是收入和人数以及平均收入的打印，这三部分的打印思路相同，都是先转化为十进制后输出，这部分实际上在前几次作业中已经实现。但是，我在起初写的时候对dd和dw这两种数据类型在table中的存放存在误解，误认为是需要在读入的时候直接把dd类型的数据转换为dw存放到table中，然后导致在进行编写的时候遇到了较大的困难，浪费了比较多的时间，而后在查阅相关资料之后采用前面提到的做法，在输出的时候转换为十进制输出。

这部分在本程序中通过CONVERT过程实现，在CONVERT过程中包含CONVERT\_LOOP过程转换为十进制，以及OUTPUT来输出数据两个部分，部分代码实现如下

|  |
| --- |
| Assembly language CONVERT PROC ; 转化为十进制后输出 CONVERT\_LOOP:  MOV CX, 10 ; 每次除十取余   PUSH BX    MOV BX,AX  MOV AX,DX  MOV DX,0  DIV CX   PUSH AX  MOV AX,BX  DIV CX   MOV CX,DX  POP DX  POP BX  PUSH CX ; 余数入栈    CMP AX, 0 ; AX和DX都为零时结束  JNE CONVERT\_LOOP  CMP DX, 0  JNE CONVERT\_LOOP   ; 打印栈内数据 OUTPUT:  POP DX  MOV AH, 2  ADD DL, '0'  INT 21H  CMP SP, 94 ; 这里因为前面有三个数据在栈中，sp=94时表示余数全部输出  JNE OUTPUT  RET CONVERT ENDP |

在这里需要说明的一点是，前几次作业是采用的方法是把余数存放在一个新的字符串中，然后统一输出，这次我采用栈的方法，存放余数，然后直接出栈输出即可，在实现上更简单方便。

**3 心得体会**

本次作业的难度可以说是这几次作业中最难的一个，用到了栈，循环，过程等方法，尤其是涉及到读取数据段中的值并输出数据段中的值的部分，花费了我大量的时间。本次作业最大的收获是，让我熟悉了过程的使用，这种类似C语言中函数的思想，极大提升了代码的可读性。虽然本次作业最后写了200多行，但是由于使用过程的方法，可以清晰地看出每一个部分的意图，而且可以很大程度上避免操作过程中对寄存器或者栈的破坏。对于日后更复杂的程序而言，使用这种过程的方法也是必不可少的。