# 汇编语言课程作业

**————期中报告**

2251334 倪朗恩

## 基本要求

1. **已给数据段如下**

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  11542,14430,15257,17800

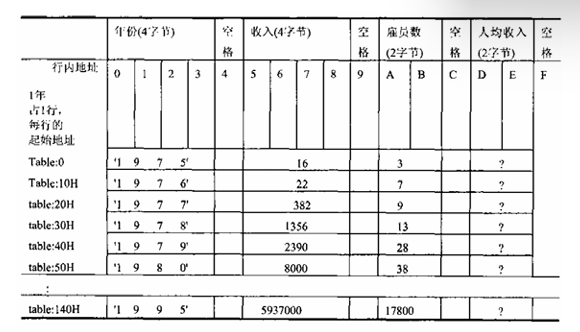
DATA ENDS

1. **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**

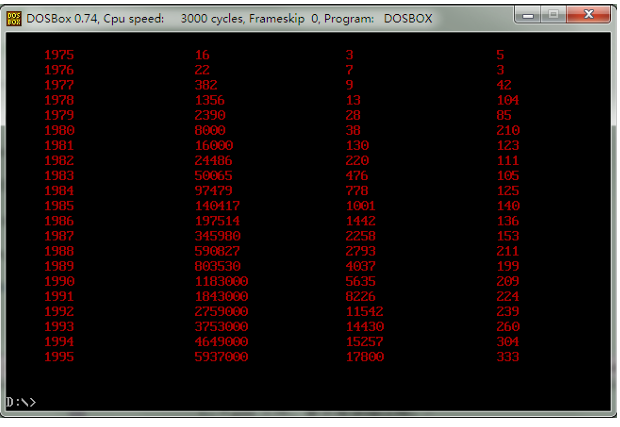
TABLE SEGMENT

    DB 21 DUP('year summ ne ?? ')

TABLE ENDS

****

1. **将table 段的数据打印出来，格式参照下图**

****

## 输出九九乘法表

### 框架

将具体流程分发至函数中，并集合功能模块相同的函数到同一文件，实现多模块编程。总共分为三个模块：

Main作为入口函数，负责调用函数以及传参，被调用的函数包括PIP\_PRINT及CLS及PIP\_OP(PIP表示批处理)，这三个函数作为PUBLIC处理；

Predeal文件用于实现数据的预处理，PIP\_OP位于其中；

Print文件用于实现数据的输出，PIP\_PRINT和CLS位于其中.

### 具体实现

#### Main.asm

先进行数据的预处理，然后清屏（CLS）并输出（PIP\_PRINT），默认DS段用于存储源数据，ES段用于存储目标位置。

ASSUME CS:CODESEG,DS:DATA,ES:TABLE,SS:STKSEG

EXTRN CLS:FAR

EXTRN PIP\_PRINT:FAR

EXTRN PIP\_OP:FAR

CODESEG SEGMENT

MAIN PROC FAR

    MOV AX,DATA

    MOV DS,AX

    MOV AX,TABLE

    MOV ES,AX

    MOV BX,4

    MOV AX,21

    MUL BX

    MOV BX,AX

    ADD AX,AX

    MOV CX,AX

    XOR SI,SI

    XOR DI,DI

    MOV AX,21

    call PIP\_OP

    MOV AH,00H    ;设置显示方式

    MOV AL,02H    ;80\*25 16色文本显示

    INT 10H

    MOV AX,ES

    MOV DS,AX

    MOV BX,0B800H

    MOV ES,BX

    MOV CX,840

    call CLS

    MOV SI,0

    MOV DI,170

    MOV AX,21

    MOV CH,04H

    call PIP\_PRINT

    ; 设置光标位置

    ; AH = 02h (Set Cursor Position)

    ; BH = 00h (Video Page Number)

    ; DH = Row (0-24)

    ; DL = Column (0-79)

    MOV AH, 02h

    MOV BH, 00h

    MOV DH, 16H  ; 设置行号

    MOV DL, 00H  ; 设置列号

    INT 10h

    MOV AX,4C00H

    INT 21H

MAIN ENDP

CODESEG ENDS

#### Predeal.asm

##### PIP\_OP

批处理函数，循环调用YEAROP，对每一组数据进行处理和保存。

参数：

1. AX输入循环次数/数据组数
2. BX保存DD类型的金额数组
3. CX保存DW类型的人数数组
4. SI指向DS：SI的源数据起始位置
5. DI指向ES：DI的存储起始位置

PIP\_OP PROC FAR

    PUSH AX

    PUSH BX

    PUSH CX

    PUSH SI

    PUSH DI

YEAR\_LOOP:

    call YEAROP

    ADD BX,4

    ADD CX,2

    ADD SI,4

    ADD DI,16

    DEC AX

    CMP AX,0

    JNE YEAR\_LOOP

    POP DI

    POP SI

    POP CX

    POP BX

    POP AX

    RET

PIP\_OP ENDP

##### YEAROP

组处理函数，对一组数据进行处理。

参数：

1. BX保存DD类型金额的位置
2. CX保存DW类型人数的位置
3. DS：SI指向年数
4. ES：DI指向输出起始位置

YEAROP PROC FAR

    PUSH AX

    PUSH CX

    PUSH SI

    PUSH DI

    PUSH CX

    MOV CX,4

    CLD

    REP MOVSB

    POP CX

    INC DI

    MOV AX,WORD PTR [BX]

    MOV ES:[DI],AX

    MOV AX,WORD PTR [BX+2]

    MOV ES:[DI+2],AX

    ADD DI,5

    XCHG BX,CX

    MOV AX,WORD PTR [BX]

    MOV ES:[DI],AX

    XCHG BX,CX

    ADD DI,3

    call DIVOP

    MOV ES:[DI],AX

    POP DI

    POP SI

    POP CX

    POP AX

    RET

YEAROP ENDP

##### DIVOP

用于处理人均工资，即DD除以DW。

参数：

1. BX保存DD类型金额的位置
2. CX保存DW类型人数的位置
3. AX用于返回计算结果

DIVOP PROC FAR

    PUSH BX

    PUSH DX

    MOV AX,WORD PTR [BX]

    MOV DX,WORD PTR [BX+2]

    XCHG BX,CX

    MOV CX,[BX]

    DIV CX

    XCHG BX,CX

    POP DX

    POP BX

    RET

DIVOP ENDP

#### Print.asm

##### CLS

清屏函数。

参数：

1. CX用于保存需要清除的字块数，一行80字，从左上循环清除

CLS PROC FAR

    PUSH BX

    PUSH CX

    PUSH DX

    XOR BX,BX

    XOR DX,DX

CLS\_LOOP:

    MOV ES:[BX],DX

    ADD BX,2

    LOOP CLS\_LOOP

    POP DX

    POP CX

    POP BX

    RET

CLS ENDP

##### PIP\_PRINT

批处理输出处理后的数据，循环调用YEAR\_PRINT函数对每一组数据输出。

参数：

1. AX输入循环次数/输出组数
2. CH选择输出字符格式（红）
3. DS：SI保存需要输出数据的起始位置
4. ES：DI保存输出位置（缓冲区）

PIP\_PRINT PROC FAR

    PUSH AX

    PUSH CX

    PUSH DI

    PUSH SI

YEAR\_PRINT\_LOOP:

    call YEAR\_PRINT

    ADD SI,16

    ADD DI,160

    DEC AX

    CMP AX,0

    JNE YEAR\_PRINT\_LOOP

    POP SI

    POP DI

    POP CX

    POP AX

    RET

PIP\_PRINT ENDP

##### YEAROP

组处理函数，对一组数据进行输出，包括年数，总金额，人数，平均工资，使用函数对不同类型数据进行输出。

参数：

1. CH选择输出字符格式（红）
2. DS：SI保存需要输出数据的起始位置
3. ES：DI保存输出位置（缓冲区）

YEAR\_PRINT PROC FAR

    PUSH AX

    PUSH BX

    PUSH CX

    PUSH DX

    PUSH DI

    MOV AH,CH

    MOV CX,4

    MOV BX,SI

    call STR\_PRINT

    MOV CH,AH

    ADD DI,32

    MOV AX,WORD PTR [SI+5]

    MOV DX,WORD PTR [SI+7]

    call DD2STR

    ADD DI,32

    MOV AX,WORD PTR [SI+10]

    call DW2STR

    ADD DI,32

    MOV AX,WORD PTR [SI+13]

    call DW2STR

    POP DI

    POP DX

    POP CX

    POP BX

    POP AX

    RET

YEAR\_PRINT ENDP

##### DD2STR

输出DD类型数字，将DD数据除以10000，分出高位和低位，并分别调用DW的输出函数（DW2STR）进行显示，从而保证数据不会溢出。

参数：

1. DX：AX保存要输出数据
2. CH保存输出格式
3. ES：DI指向输出位置/缓冲区

DD2STR PROC FAR

    PUSH AX

    PUSH BX

    PUSH DX

    PUSH DI

    MOV BX,10000

    DIV BX

    call DW2STR

    XCHG AX,DX

    call DW2STR

    POP DI

    POP DX

    POP BX

    POP AX

    RET

DD2STR ENDP

##### DW2STR

输出DW类型数字，循环除10将低位数字压栈，然后循环出栈并输出。

参数：

1. AX保存要输出DW数据
2. CH保存输出格式
3. ES：DI指向输出位置/缓冲区

DW2STR PROC FAR

    PUSH AX

    PUSH BX

    PUSH CX

    PUSH DX

    PUSH DI

    XOR CL,CL

    MOV BX,10

GET\_DIGIT:

    INC CX

    XOR DX,DX

    DIV BX

    MOV DH,CH

    PUSH DX

    CMP AX,0

    JNE GET\_DIGIT

    XOR CH,CH

OUTPUT\_DW:

    POP AX

    ADD AL,30H

    MOV ES:[DI],AX

    ADD DI,2

    LOOP OUTPUT\_DW

    POP DI

    POP DX

    POP CX

    POP BX

    POP AX

    RET

DW2STR ENDP

##### STR\_PRINT

输出指定数目的字符串。

参数：

1. CX保存输出字符数
2. AH保存输出格式
3. DS：BX指向输出数据起始位置
4. ES：DI指向输出位置/缓冲区

STR\_PRINT PROC FAR

    PUSH AX

    PUSH BX

    PUSH CX

    PUSH DI

OUTPUT\_STR\_LOOP:

    MOV AL,[BX]

    MOV ES:[DI],AX

    INC BX

    ADD DI,2

    LOOP OUTPUT\_STR\_LOOP

    POP DI

    POP CX

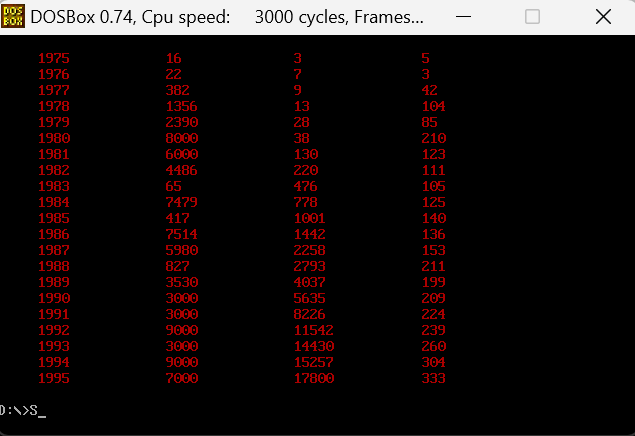
    POP BX

    POP AX

    RET

STR\_PRINT ENDP

## 实验结果



## 问题与心得

1. 函数说明信息

当对功能进行分割，函数会变得很多，适当、充足、准确的注释有效地帮助了代码编写以及debug。

要注意参数的输入格式/风格要明确。

1. 多模块编程

将不同的功能编写进入不同的文件，有利于明确编写以及维护。

本次作业中，提供了明确的存储方式（按内存），导致小函数虽然多但功能零散，依据阶段目标进行功能划分能更好维护，故选择如此划分。

实际上可以依据运算、输出、数据结构、主函数进行划分，为table编写明确的struct，并将对struct的数据处理整合在一个文件，其他分别存储在运算文件以及输出文件，这样有利于后期维护、代码复用和扩展，降低代码耦合性。

1. 对缓冲区有了更深刻的了解