

汇编语言学习笔记：工资计算程序

一、实验目的

通过实现工资计算程序，掌握汇编语言中数据段操作、数组索引计算、32位除法运算和过程调用的关键技术。

二、程序功能

将21年的公司数据（年份、总收入、雇员人数）从data段转移到table段，并计算每年的人均收入，最后格式化输出结果。

三、关键技术点

1. 数据段定义与内存布局

```
data segment
    years db '1975','1976','1977'...
    incomes dd 16,22,382...
    employees dw 3,7,9...
data ends

table segment
    db 21 dup('year summ ne ?? ')
table ends
```

内存布局分析：

- **years**: 84字节 (21×4)
- **incomes**: 84字节 (21×4)
- **employees**: 42字节 (21×2)
- **table**: 336字节 (21×16)

2. 数组索引计算

由于不同数组元素大小不同，需要分别计算偏移量：

```
; 年份偏移量 = 索引 × 4
mov ax, si
mov cx, 4
mul cx
mov si, ax

; 收入偏移量 = 索引 × 4
mov ax, si
mov cx, 4
mul cx
mov si, ax
```

```
; 雇员偏移量 = 索引 × 2
mov ax, si
mov cx, 2
mul cx
mov si, ax
```

关键理解：

- 年份是字符串，按字节连续存储
- 收入是dword，每个元素4字节
- 雇员是word，每个元素2字节
- 必须正确计算每个数组的偏移量

3. 32位除法运算

```
; 设置被除数 (32位)
mov dx, word ptr temp_income      ; 高16位
mov ax, word ptr temp_income+2    ; 低16位

; 设置除数 (16位)
mov temp_employee, ax

; 执行除法
div temp_employee                ; 结果在AX
```

除法规则：

- 被除数：DX:AX（32位）
- 除数：16位寄存器或内存
- 结果：商在AX，余数在DX
- 必须确保DX:AX包含正确的32位数

4. 过程调用与寄存器保护

```
print_number proc
    push ax
    push bx
    push cx
    push dx
    push si

    ; 过程主体代码

    pop si
    pop dx
    pop cx
    pop bx
```

```

pop ax
ret
print_number endp

```

寄存器保护原则：

- 在过程开始时保存所有使用的寄存器
- 在过程结束时按相反顺序恢复
- 避免破坏调用者的数据

四、调试过程与问题解决

问题1：年份显示错误

现象：年份显示为乱序 (1975, 9751, 7519...) **原因：**年份索引步进错误，使用`inc si`而不是`add si, 4` **解决：**改为`add si, 4`，因为每个年份占4字节

问题2：人均收入显示为0

现象：最后一列全部显示0 **原因：**除法运算设置错误 **解决：**确保被除数正确设置在DX:AX中，除数在16位寄存器中

问题3：编译错误"constant expected"

现象：复杂数组表达式编译失败 **原因：**MASM不支持`incomes[si*2]`这样的复杂索引 **解决：**使用显式计算偏移量

五、重要指令详解

1. MUL 指令

```

mov ax, si      ; 被乘数
mov cx, 4       ; 乘数
mul cx          ; AX = AX × CX

```

- 用于计算数组偏移量
- 结果在AX (8位×8位) 或DX:AX (16位×16位)

2. DIV 指令

```

mov dx, high_word ; 被除数高16位
mov ax, low_word  ; 被除数低16位
div divisor       ; AX = 商, DX = 余数

```

- 用于32位÷16位运算
- 必须正确设置DX:AX

3. LOOP 指令

```
mov cx, 21
process_loop:
    ; 循环体
    loop process_loop ; CX--，如果CX≠0则循环
```

- 简化循环控制
- CX作为计数器

六、编程技巧总结

1. 数据访问技巧

- 使用`word ptr`和`dword ptr`明确数据类型
- 数组访问：基地址 + 偏移量
- 多字节数据：分高低字节处理

2. 调试技巧

- 分模块测试：先测试数据转移，再测试计算
- 使用DEBUG单步执行观察寄存器变化
- 检查关键计算：如 $1975 \div 3 = 5$

3. 代码组织技巧

- 使用过程封装重复功能
- 清晰的注释和标签命名
- 合理的寄存器分配

七、学习收获

1. **深入理解内存布局**：掌握了不同数据类型在内存中的存储方式
2. **熟练数组操作**：学会了处理不同大小元素的数组索引计算
3. **掌握数值运算**：理解了32位除法的原理和实现
4. **提升调试能力**：通过实际问题调试，加深了对汇编执行流程的理解
5. **强化过程设计**：学会了如何设计可重用的过程模块