

电子科技大学信息与软件工程学院

实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 汇编语言程序设计

理论教师 赵 洋

实验教师 赵 洋

电子科技大学

实验报告

学生姓名：袁昊男 学号：2018091618008 指导教师：赵洋

实验地点：在线实验 实验时间：2020.04.26

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：寻址方式在结构化数据访问中的应用

三、实验学时：2 学时

四、实验原理：

计算机是进行数据处理、运算的工具，使用过程中需要明确两个基本问题：

1、处理的数据在什么地方？

当数据存在内存中的时候，我们可以用多种方式来给定这个内存单元的偏移地址，这种定位内存单元的方法一般被称为寻址方式。8086CPU 提供了多种寻址方式，包括：直接寻址、寄存器间接寻址、寄存器相对寻址、基址变址寻址、相对基址变址寻址。

2、要处理的数据有多长？

8086CPU 的指令可以处理两种尺寸的数据，byte 和 word。所以在指明指令中进行的是字节操作还是字操作。对于这个问题，汇编语言中用以下方式进行处理：

- (1) 通过寄存器名指明要处理的数据尺寸。
- (2) 在指令中没有寄存器名存在的情况下，用操作符 (Xptr) 指明内存单元的长度，X 在汇编指令中可以为 word 或 byte。
- (3) 指令中默认了访问的是字节单元还是字单元，如 push 和 pop 指令只对字单元进行操作。

五、实验目的：

- 1、掌握各种寻址方式的使用。
- 2、掌握汇编语言中复杂数据结构的定义和使用。
- 3、掌握正确分配与使用寄存器与存储单元。
- 4、掌握 div 指令的使用。

5、掌握 dd、dw、dup 等伪指令的使用。

六、实验内容：

编程实现：将 datasg 段中的数据按如下格式写入到 table 表中，并计算 21 年中的人均收入（取整），结果也保存在 table 表中。

```
assume cs:codesg
```

```
datasg segment
```

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

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

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

```
;以上是表示 21 年的 21 个字符串
```

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

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

```
;以上是表示 21 年公司总收的 21 个 dword 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
```

```
;以上是表示 21 年公司雇员人数的 21 个 word 型数据
```

```
datasg ends
```

```
table segment
```

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

```
table ends
```

编程，将 data 段中的数据按照如下格式写入到 table 段中，并计算 21 年中的人均收入（取整），结果也按照下面的格式保存在 table 段中。

表 1 table 段数据结构

| | 年份 (4 字节) | | | | 空 格 | 收入 (4 字节) | | | | 空 格 | 雇员数 (2 字节) | | 空 格 | 人均 收入 (2 字节) | | 空 格 |
|--|--------------|---|---|----|--------|--------------|---|---|---|--------|---------------|---|--------|--------------------|---|--------|
| 行内 地址 1 年 占1行, 每行的 起始地址 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| table:0 | '1 | 9 | 7 | 5' | | 16 | | | | | 3 | | | ? | | |
| | | | | | | | | | | | | | | | | |
| table:140H | '1 | 9 | 9 | 5' | | 5937000 | | | | | 17800 | | | ? | | |

七、实验器材（设备、元器件）：

PC 微机一台

八、实验步骤：

- 1、编辑源程序，建立一个以后缀.ASM 的文件。
- 2、汇编源程序，检查程序有否错误，有错时回到编辑状态，修改程序中错误行，无错时继续第 3 步。
- 3、连接目标程序，产生可执行程序。
- 4、用 DEBUG 程序调试可执行程序，记录数据段的内容。

九、实验数据及结果分析

1、思路分析

(1) 总体思路

将 data 段中的数据看成是多个数组，将 table 中的数据看成是一个结构型数据的数组，每个结构型数据中包含多个数据项。可用 bx 定位每个结构型数据，用 idata 定位数据项，用 si 定位数据项中的每个元素，对于 table 中的数据的访问采用[bx].idata 和[bx].idata[si]的寻址方式。

(2) 数据存储地址分析

表 2 data 段中的数据结构

| | | | | | | | | | |
|-----------|--------|--------|--------|--------|-------|---------|---------|---------|---------|
| data:0h | '1975' | '1975' | '1975' | '1975' | | '1975' | '1975' | '1975' | '1975' |
| data:54h | 16 | 22 | 382 | 1356 | | 2759000 | 3753000 | 4649000 | 5937000 |
| data:0a8h | 3 | 7 | 9 | 13 | | 11542 | 14430 | 15257 | 17800 |

表格中 1、2 行每一数据占 4 个字节，第三行每一数据占 2 个字节。因此第一行偏移地址范围是 data:0h~data:53h（年份），第二行偏移地址范围是 data:54h~data:0a7h（收入），第三行偏移地址范围是 data:a8h~data:d0h（雇员数）。

把表 2 中的三行数据按照顺序依次 mov 到表 1 中并进行 div 运算。所以可以先将表 2 中的数据按行 mov 到表 1 相应的单元中在进行 div 运算或者将表 2 中的数据按列 mov 到表 1 相应的单元中在进行 div 运算。因此有两种方式实现本实验。这里采用按行 mov 的方法。

(3) 寻址方式

字符串型数据（年份）的寻址方式为[bx+idata]（idata 取值为 0、2）；
dword 型数据（收入）的寻址方式为[bx+idata]（idata 取值为 84、86）；
word 型数据（雇员数）的寻址方式为[bx+idata]（idata 取值为 168）。

2、流程图

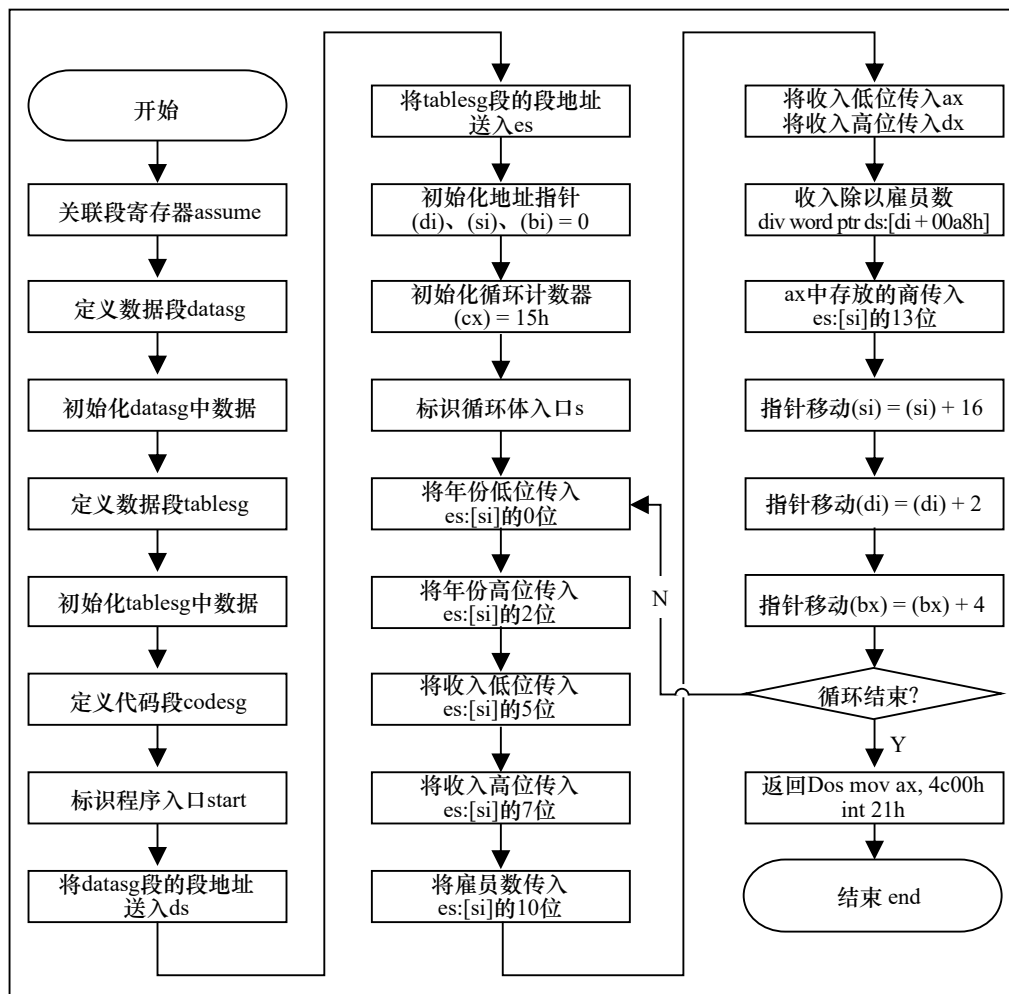


图 1 实验一流程图

3、实验截图

```

DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c ~/MYDOSBox
Drive C is mounted as local directory /Users/tommy/MYDOSBox/

Z:\>c:

C:\>masm ex_7:
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51490 + 465054 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link ex_7:
Microsoft (R) Segmented-Executable Linker Version 5.13
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.

LINK : warning L4021: no stack segment

C:\>_
  
```

图 2 编译、连接

```

DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
04D1:0003 8ED8      MOV     DS,AX
-t
AX=04AE BX=0000 CX=02B6 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=04AE ES=049E SS=04AD CS=04D1 IP=0005  NU UP DI PL NZ NA PO NC
04D1:0005 B8BC04      MOV     AX,04BC
-d 04ae:0
04AE:0000 31 39 37 35 31 39 37 36-31 39 37 37 31 39 37 38 1975197619771978
04AE:0010 31 39 37 39 31 39 38 30-31 39 38 31 31 39 38 32 1979198019811982
04AE:0020 31 39 38 33 31 39 38 34-31 39 38 35 31 39 38 36 1983198419851986
04AE:0030 31 39 38 37 31 39 38 38-31 39 38 39 31 39 39 30 1987198819891990
04AE:0040 31 39 39 31 31 39 39 32-31 39 39 33 31 39 39 34 1991199219931994
04AE:0050 31 39 39 35 10 00 00 00-16 00 00 00 7E 01 00 00 1995.....~...
04AE:0060 4C 05 00 00 56 09 00 00-40 1F 00 00 80 3E 00 00 L...U...@....>..
04AE:0070 A6 5F 00 00 91 C3 00 00-C7 7C 01 00 81 24 02 00 &_...C...G!...$.
-d 04ae:0080
04AE:0080 8A 03 03 00 7C 47 05 00-EB 03 09 00 CA 42 0C 00 ....!G...k...JB..
04AE:0090 18 0D 12 00 38 1F 1C 00-58 19 2A 00 28 44 39 00 ....8...X...*(D9.
04AE:00A0 28 F0 46 00 68 97 5A 00-03 00 07 00 09 00 0D 00 (pF.h.Z.....
04AE:00B0 1C 00 26 00 82 00 DC 00-DC 01 0A 03 E9 03 A2 05 ..&...N...i..".
04AE:00C0 D2 08 E9 0A C5 0F 03 16-22 20 16 2D 5E 38 99 3B R.i.E..."^8.;
04AE:00D0 88 45 00 00 00 00 00-00 00 00 00 00 00 00 00 .E.....
04AE:00E0 79 65 61 72 20 73 75 6D-6D 20 6E 65 20 3F 3F 20 year summ ne ??
04AE:00F0 79 65 61 72 20 73 75 6D-6D 20 6E 65 20 3F 3F 20 year summ ne ??

```

图 3 跟踪、调试（程序执行前）

```

DOSBox 0.74-3-1, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
04AE:0020 31 39 38 33 31 39 38 34-31 39 38 35 31 39 38 36 1983198419851986
04AE:0030 31 39 38 37 31 39 38 38-31 39 38 39 31 39 39 30 1987198819891990
04AE:0040 31 39 39 31 31 39 39 32-31 39 39 33 31 39 39 34 1991199219931994
04AE:0050 31 39 39 35 10 00 00 00-16 00 00 00 7E 01 00 00 1995.....~...
04AE:0060 4C 05 00 00 56 09 00 00-40 1F 00 00 80 3E 00 00 L...U...@....>..
04AE:0070 A6 5F 00 00 91 C3 00 00-C7 7C 01 00 81 24 02 00 &_...C...G!...$.
-d 04ae:0080
04AE:0080 8A 03 03 00 7C 47 05 00-EB 03 09 00 CA 42 0C 00 ....!G...k...JB..
04AE:0090 18 0D 12 00 38 1F 1C 00-58 19 2A 00 28 44 39 00 ....8...X...*(D9.
04AE:00A0 28 F0 46 00 68 97 5A 00-03 00 07 00 09 00 0D 00 (pF.h.Z.....
04AE:00B0 1C 00 26 00 82 00 DC 00-DC 01 0A 03 E9 03 A2 05 ..&...N...i..".
04AE:00C0 D2 08 E9 0A C5 0F 03 16-22 20 16 2D 5E 38 99 3B R.i.E..."^8.;
04AE:00D0 88 45 00 00 00 00 00-00 00 00 00 00 00 00 00 .E.....
04AE:00E0 31 39 37 35 20 10 00 00-00 20 03 00 20 05 00 20 1975 .....
04AE:00F0 31 39 37 36 20 16 00 00-00 20 07 00 20 03 00 20 1976 .....
-d 04ae:0100
04AE:0100 31 39 37 37 20 7E 01 00-00 20 09 00 20 2A 00 20 1977 ~... ..*.
04AE:0110 31 39 37 38 20 4C 05 00-00 20 0D 00 20 68 00 20 1978 L... ..h.
04AE:0120 31 39 37 39 20 56 09 00-00 20 1C 00 20 55 00 20 1979 U... ..U.
04AE:0130 31 39 38 30 20 40 1F 00-00 20 26 00 20 D2 00 20 1980 @... ..&.R.
04AE:0140 31 39 38 31 20 80 3E 00-00 20 82 00 20 7B 00 20 1981 .>... ..{.
04AE:0150 31 39 38 32 20 A6 5F 00-00 20 DC 00 20 6F 00 20 1982 &_... \. o.
04AE:0160 31 39 38 33 20 91 C3 00-00 20 DC 01 20 69 00 20 1983 .C... \. i.
04AE:0170 31 39 38 34 20 C7 7C 01-00 20 0A 03 20 7D 00 20 1984 G!... ..}.

```

图 4 跟踪、调试（程序执行后）

4、结果分析

(1) 程序执行前

使用 D 命令查看数据段内容，可见数据已经正确存入相应位置（图 3）。04AE:0000~04AE:0053 存储的是 21 年的 21 个字符串；04AE:0054~04AE:00A7 存储的是 21 年公司总收入的 21 个 dword 型数据；04AE:00A8~04AE:00D1 存储的是 21 年公司雇员人数的 21 个 word 型数据；04AE:00E0~04AE:1020 存储的是 21 行 'year summ ne ??'。

(2) 程序执行后

使用 D 命令查看数据段内容，可见数据已经正确存入相应位置（图 4）。以 04AE:00E0~04AE:00EF 内数据为例分析，16 个字节内存储的内容依次是“31 39 37 35 20 10 00 00 00 20 03 00 20 05 00 20”，其中“31 39 37 35”依次为“1”、“9”、“7”、“5”四个字符的十六进制 ASCII 码，对应年份（占 4 字节）；“20”为空格字符的十六进制 ASCII 码；“10 00 00 00”为数值 16 的十六进制，对应收入（占 4 字节）；“03 00”为数值 3 的十六进制，对应雇员数（占 2 字节）；“05 00”为数值 5 的十六进制，对应人均收入（ $16 \div 3 = 5.33$ ，取整）。其余 20 组数据同理可得结果正确。

十、实验结论

代码经 masm 编译通过，通过 Debug 调试发现：程序执行前，初始数据正确存入目的地址中；程序执行后，程序正确计算出人均收入，并结构化地存入目的地址中。

由此可得出：将数据看作数组，分析其存储结构与地址关系，借助 bx、si 等寄存器综合使用 [bx].idata 和 [bx].idata[si] 的寻址方式，采用简单的循环结构，可较高效地对结构型数据进行操作。

十一、总结及心得体会

- 1、将具有明显结构化特征的数据看成结构型数据的数组，每个数组中包含多个数据项，综合使用 [bx].idata 和 [bx].idata[si] 的寻址方式可较高效地对数据进行定位与操作。
- 2、数据在内存中实际并没有“行”存储的概念，但可以将顺序存储的数据抽象成按“行”存储的方式，便于分析数据地址之间的关系与数据定位。
- 3、注意 div 除法指令对被除数、除数的大小限制及商与余数的存储位置。在本实验中，被除数总收入占 4 字节（32 位）、除数雇员数占 2 字节（16 位），因此 DX 中存放总收入的高 16 位、AX 中存放总收入的低 16 位、雇员数存放在任一内存单元中；商存放于 AX 中、余数存放于 DX 中。
- 4、使用 Debug 工具对代码进行调试，如 D、U 命令等查看内存与指令，有助于对代码细节的调整，能较好地对命令执行情况作出反馈。

十二、对本实验过程及方法、手段的改进建议

在使用 Debug 工具对内存单元存放数据进行查看时，一些数值型数据被当作 ASCII 码输出，不便于判断目的单元存放的数据是否正确。因此可以将相关数

据格式化输出，便于编程人员调试。汇编语言输出功能可以调用 DOS 功能中的 09 号功能，也就是显示 DS:DX 中的字符串，采用中断方式输出。

报告评分：

指导教师签字：

附录：实验程序源码

```
assume cs:codesg,ds:datasg,es:tablesq

datasg segment
    db '1975','1976','1977','1978','1979','1980','1981','1982','1983'
    db '1984','1985','1986','1987','1988','1989','1990','1991','1992'
    db '1993','1994','1995'
    ;以上是表示 21 年的 21 个字符串

    dd 16,22,382,1356,2390,8000,16000,24486,50065,97479,140417,197514
    dd 345980,590827,803530,1183000,1843000,2759000,3753000,4649000,5937000
    ;以上是表示 21 年公司总收的 21 个 dword 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
    ;以上是表示 21 年公司雇员人数的 21 个 word 型数据
datasg ends

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

codesg segment

start: mov ax,datasg
       mov ds,ax                ;将 data 段地址传入 ds
       mov ax,tablesq
       mov es,ax                ;将 table 段地址传入 es
       mov bx,0
       mov si,0
       mov di,0
       mov cx,15h               ; 循环次数

s:     mov ax,ds:[bx]             ; 将年份低位传入 ax 寄存器
       mov es:[si],ax            ; 将 ax 中的年份低位传入 es:[si]的 0 位
       mov ax,ds:[bx + 2]        ; 将年份高位传入 ax 寄存器
       mov es:[si + 2],ax        ; 将 ax 中的年份高位传入 es:[si]的 2 位

       mov ax,ds:[bx + 54h]       ; 将收入低位传入 ax 寄存器
       mov es:[si + 5],ax        ; 将 ax 中的收入低位传入 es:[si]的 5 位
       mov ax,ds:[bx + 56h]       ; 将收入高位传入 ax 寄存器
       mov es:[si + 7],ax        ; 将 ax 中的收入高位传入 es:[si]的 7 位

       mov ax,ds:[di + 00a8h]     ; 将雇员数传入 ax 寄存器
       mov es:[si + 0ah],ax       ; 将雇员数传入 es:[si]的 10 位

       mov ax,[bx + 54h]          ; 取收入的低位传入 ax
       mov dx,[bx + 56h]          ; 取收入的高位传入 dx
       div word ptr ds:[di + 00a8h] ; 收入除以雇员数
       mov es:[si + 0dh],ax       ; 将 ax 中收入的整数传入 es:[si]的 13 位

       add si,16                  ; 下一行
       add di,2                   ; 雇员数和人均收入均为 2 字节
       add bx,4                   ; 年份和收入均为 4 字节

       loop s
```

```
    mov ax,4c00h  
    int 21h  
  
codesg ends  
end start
```