第2节 循环程序

对内存的寻址方式

形式	名称	特点	意义	示例
[idata]	直接寻址	用一个常量/立即数来表示地址	用于直接定位一个内存单元	mov ax, [200]
[bx]	寄存器间接寻址	用一个变量来表示内存地址	用于间接定位一个内存单元	mov bx, 0 mov ax, [bx]
[bx+idata]	寄存器相对寻址	用一个变量和常量表示地址	可在一个起始地址的基础上用变量间接定位一个内存单元	mov bx, 4 mov ax, [bx+200]
[bx+si]	基址变址寻址	用两个变量表示地址		mov ax, [bx+si]
[bx+si+idata]	相对基址变址寻址	用两个变量和一个常量表示地址		mov ax, [bx+si+200]

example02.asm

```
assume cs:code, ds:data, ss:stack

data segment
    a0: db 4 dup(0)  ; 数组a的第0行
    a1: db 4 dup(1)  ; 数组a的第1行
    a2: db 4 dup(2)  ; 数组a的第2行
    b:  db 12 dup(8) ; 需要复制到的地方
data ends

stack segment
    db 20 dup(0)     ; 预留的栈空间
stack ends

code segment
main:
    mov ax, data       ; 初始化数据段寄存器
    mov ds, ax

    mov ax, stack      ; 初始化栈寄存器
    mov ss, ax

    mov ax, 20         ; 初始化栈顶
    mov sp, ax

    mov bx, 0          ; i指针，用于遍历行
    mov cx, 3          ; 遍历行
s0:
    push cx            ; 保存现场
    mov si, 0          ; j指针，用于遍历数组的列
    mov cx, 4          ; 遍历列
```

```
s1:
    mov al, [bx+si]  ; 从数组a读取元素
    mov [bx+si]12, al  ; 写入数组b
    inc si             ; 列指针增加
    loop s1            ; 继续循环列
    pop cx             ; 恢复现场
    add bx,4           ; 行指针增加
    loop s0            ; 继续循环行
end main
code ends
```

1.寻址方式

简单来说就是常量作为数组的首地址，然后bx和si（di也行）作为两个指针进行寻址，注意，不能使用[si+di]，这两个指针只能各自单独配合bx使用。另外，对于不同的编译器，寻址的书写方式不同，注意观察编译后的代码是不是你想要的寻址方式。

以上面的例子来说，数组a首地址为0，数组b的首地址为12，然后bx用于指定行，si用于指定列，由于上面a的第一行占4个字节，因此bx每次+4，si每次+1 。

对于emu8086来说，

定位a[i][j]就是[bx][si]或者[bx+si]

定位b[i][j]就是[bx][si]12或者[bx+si]12，或者[bx+si]b

注意，这里寻址访问内存单元，b会直接翻译为地址，如果把b的地址移动到ax中，需要使用

mov ax,offset b

如果把b地址的内容移动到ax中，应该是

mov ax,[b]

所以在方括号寻址的地方，标号b就表示地址，而没有方括号的地方，需要加上offset才能取出地址，

这里对于数据的标号，使用起来似乎还不是特别方便，到后面我们会介绍直接定址表，介绍一种专门用于

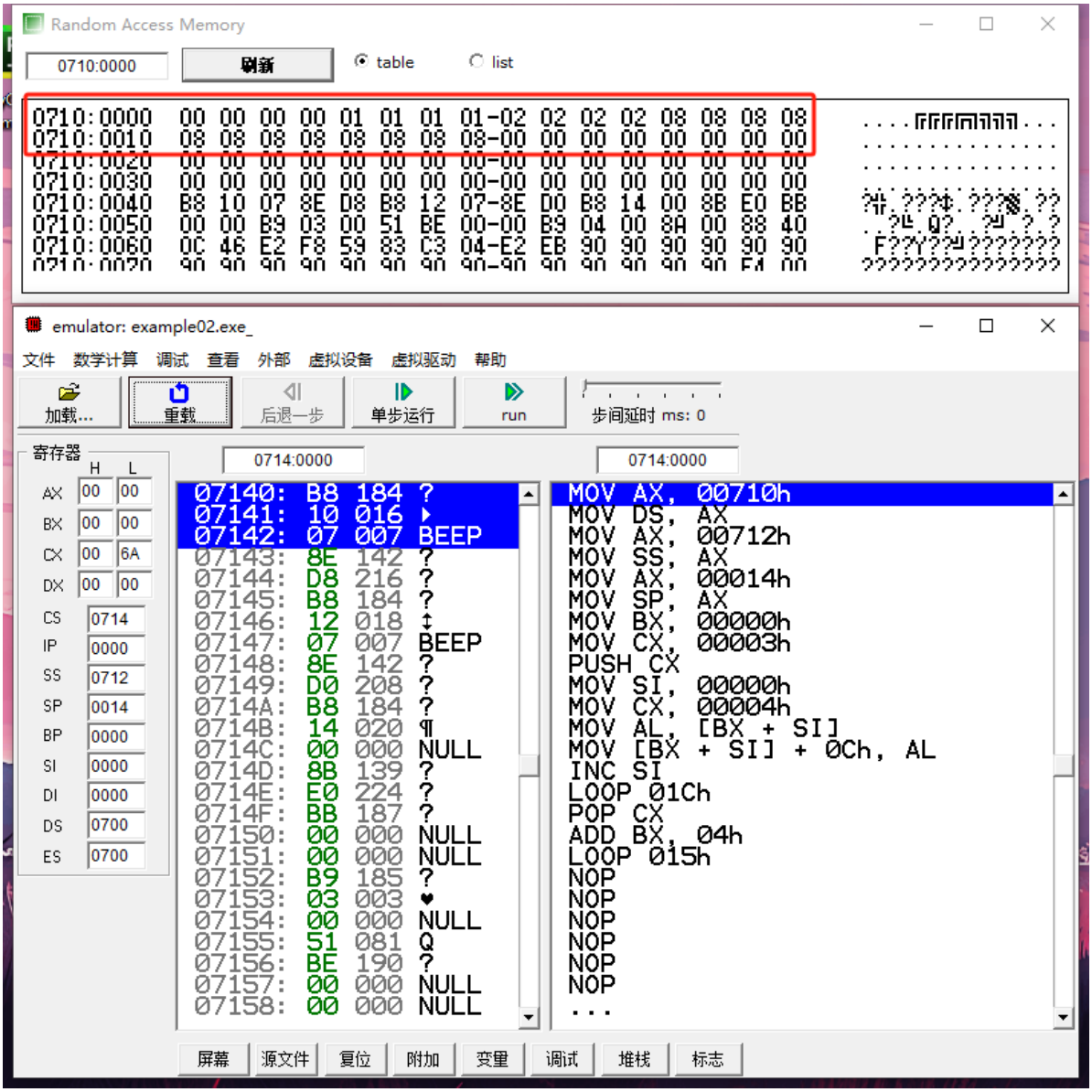
数据的标号，会使得数据访问变得十分方便，上面这种如果感觉不方便，直接忘记即可，知道有这么一个东西存在就行

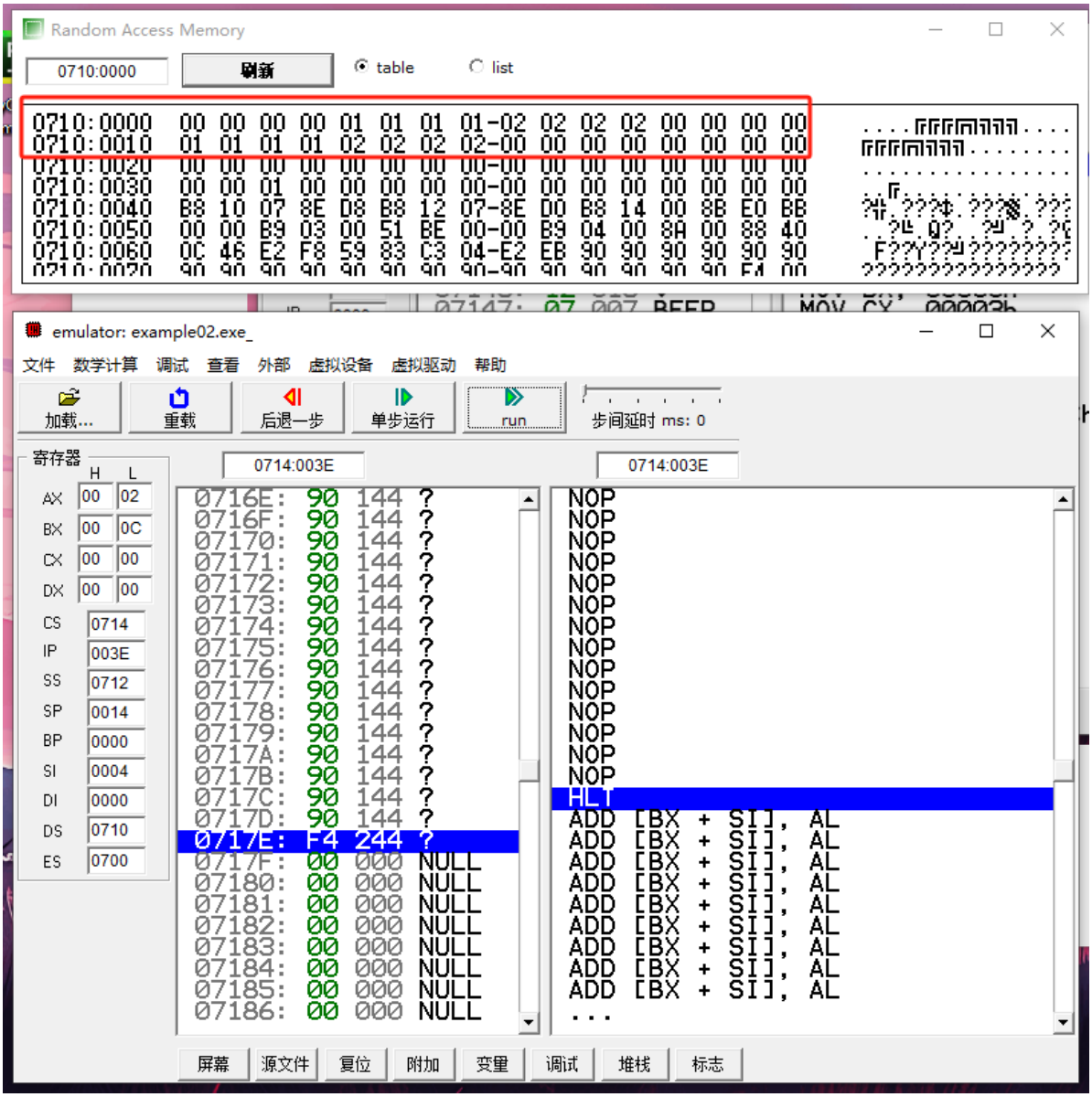
2.双重循环

由于cx用于循环次数计数，两次循环都要用到cx，为了互不影响，在进入第二次循环之前，保存cx，第二次循环结束之后还原cx，保存和还原需要用到的栈需要手动初始化栈的段地址和栈顶指针sp。

此外，复制每一行的时候，列指针si都要归0

下面两张图是代码执行前后，data段内存变化情况


Random Access Memory
0710:0000
刷新
table
list
0710:0000 00 00 00 00 01 01 01 01-02 02 02 02 08 08 08 08
0710:0010 08 08 08 08 08 08 08 08-00 00 00 00 00 00 00 00
0710:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0040 B8 10 07 8E D8 B8 12 07-8E D0 B8 14 00 8B E0 BB
0710:0050 00 00 B9 03 00 51 BE 00-00 B9 04 00 8A 00 88 40
0710:0060 0C 46 E2 F8 59 83 C3 04-E2 EB 90 90 90 90 90 90
emulator: example02.exe_
文件 数学计算 调试 查看 外部 虚拟设备 虚拟驱动 帮助
加载...
重载
后退一步
单步运行
run
步间延时 ms: 0
寄存器
H L
AX 00 00
BX 00 00
CX 00 6A
DX 00 00
CS 0714
IP 0000
SS 0712
SP 0014
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0714:0000
07140: B8 184 ?
07141: 10 016 ►
07142: 07 007 BEEP
07143: 8E 142 ?
07144: D8 216 ?
07145: B8 184 ?
07146: 12 018 ↕
07147: 07 007 BEEP
07148: 8E 142 ?
07149: D0 208 ?
0714A: B8 184 ?
0714B: 14 020 ¶
0714C: 00 000 NULL
0714D: 8B 139 ?
0714E: E0 224 ?
0714F: BB 187 ?
07150: 00 000 NULL
07151: 00 000 NULL
07152: B9 185 ?
07153: 03 003 ♥
07154: 00 000 NULL
07155: 51 081 Q
07156: BE 190 ?
07157: 00 000 NULL
07158: 00 000 NULL
0714:0000
MOV AX, 00710h
MOV DS, AX
MOV AX, 00712h
MOV SS, AX
MOV AX, 00014h
MOV SP, AX
MOV BX, 00000h
MOV CX, 00003h
PUSH CX
MOV SI, 00000h
MOV CX, 00004h
MOV AL, [BX + SI]
MOV [BX + SI] + 0Ch, AL
INC SI
LOOP 01Ch
POP CX
ADD BX, 04h
LOOP 015h
NOP
NOP
NOP
NOP
NOP
NOP
...
屏幕
源文件
复位
附加
变量
调试
堆栈
标志


Random Access Memory
0710:0000
刷新
table
list
0710:0000 00 00 00 00 01 01 01 01-02 02 02 02 00 00 00 00
0710:0010 01 01 01 01 02 02 02 02-00 00 00 00 00 00 00 00
0710:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0030 00 00 01 00 00 00 00 00-00 00 00 00 00 00 00 00
0710:0040 B8 10 07 8E D8 B8 12 07-8E D0 B8 14 00 8B E0 BB
0710:0050 00 00 B9 03 00 51 BE 00-00 B9 04 00 8A 00 88 40
0710:0060 0C 46 E2 F8 59 83 C3 04-E2 EB 90 90 90 90 90 90
emulator: example02.exe_
文件 数学计算 调试 查看 外部 虚拟设备 虚拟驱动 帮助
加载...
重载
后退一步
单步运行
run
步间延时 ms: 0
寄存器
H L
AX 00 02
BX 00 0C
CX 00 00
DX 00 00
CS 0714
IP 003E
SS 0712
SP 0014
BP 0000
SI 0004
DI 0000
DS 0710
ES 0700
0714:003E
0716E: 90 144 ?
0716F: 90 144 ?
07170: 90 144 ?
07171: 90 144 ?
07172: 90 144 ?
07173: 90 144 ?
07174: 90 144 ?
07175: 90 144 ?
07176: 90 144 ?
07177: 90 144 ?
07178: 90 144 ?
07179: 90 144 ?
0717A: 90 144 ?
0717B: 90 144 ?
0717C: 90 144 ?
0717D: 90 144 ?
0717E: F4 244 ?
0717F: 00 000 NULL
07180: 00 000 NULL
07181: 00 000 NULL
07182: 00 000 NULL
07183: 00 000 NULL
07184: 00 000 NULL
07185: 00 000 NULL
07186: 00 000 NULL
0714:003E
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
屏幕
源文件
复位
附加
变量
调试
堆栈
标志