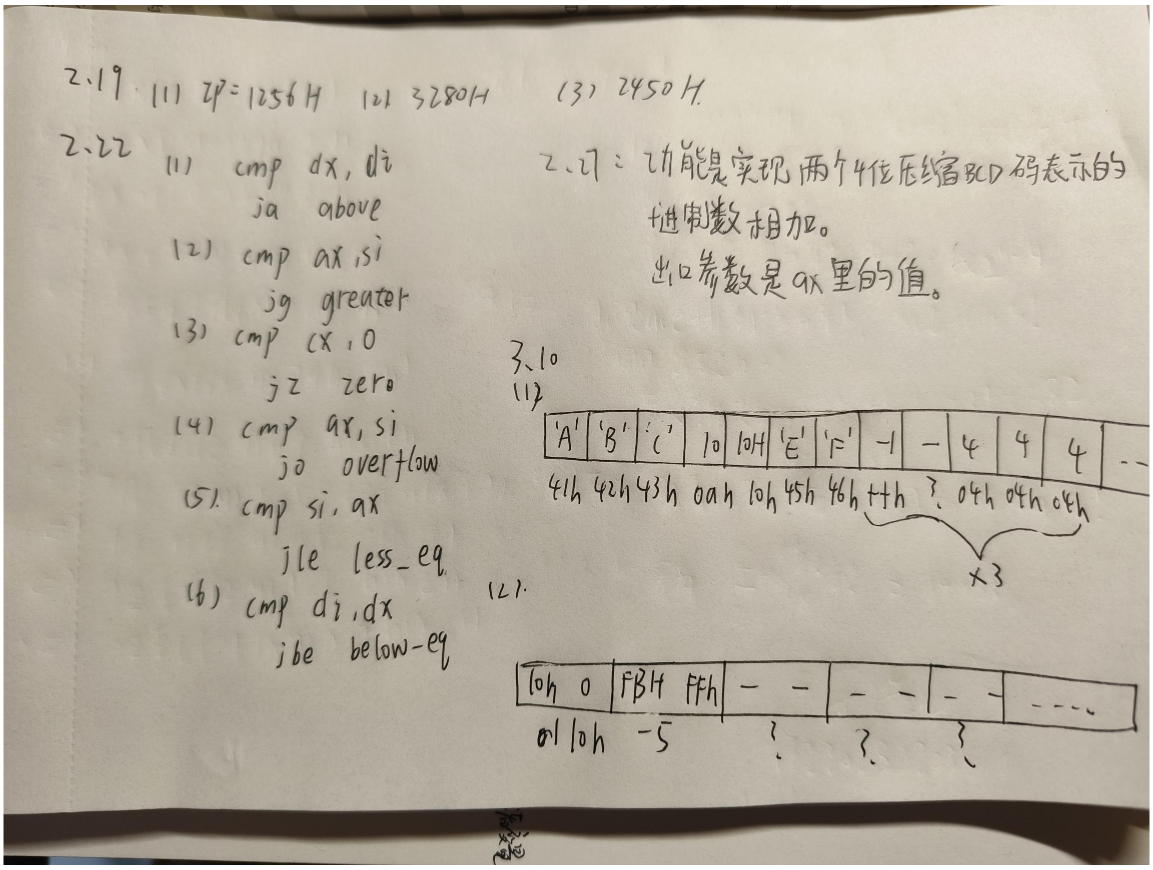


2.19. (1) IP=1256H (2) 3280H (3) 2450H

2.22
(1) cmp dx, di
    ja above
(2) cmp ax, si
    jg greater
(3) cmp cx, 0
    jz zero
(4) cmp ax, si
    jo overflow
(5) cmp si, ax
    jle less_eq
(6) cmp di, dx
    jbe below-eq

2.27：功能是实现两个4位压缩BCD码表示的十进制数相加。
出口参数是ax里的值。

3.10
(1)

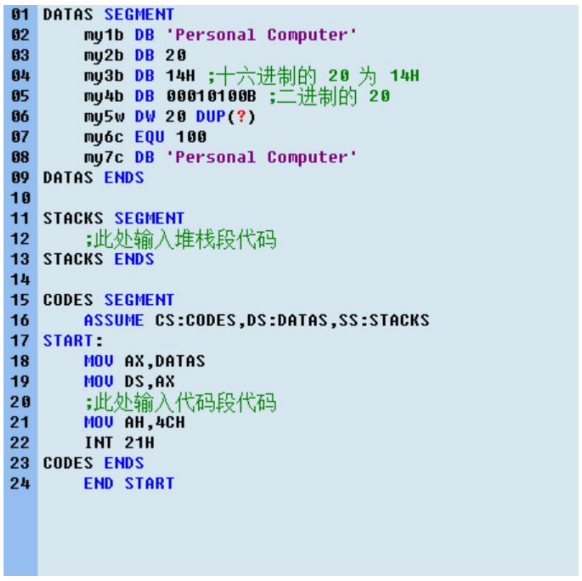
'A'	'B'	'C'	10	10H	'E'	'F'	-1	-	4	4	4	--
41h	42h	43h	0ah	10h	45h	46h	ffh	?	04h	04h	04h	

$\underbrace{\text{ffh ? 04h 04h 04h}}_{\times 3}$

(2).

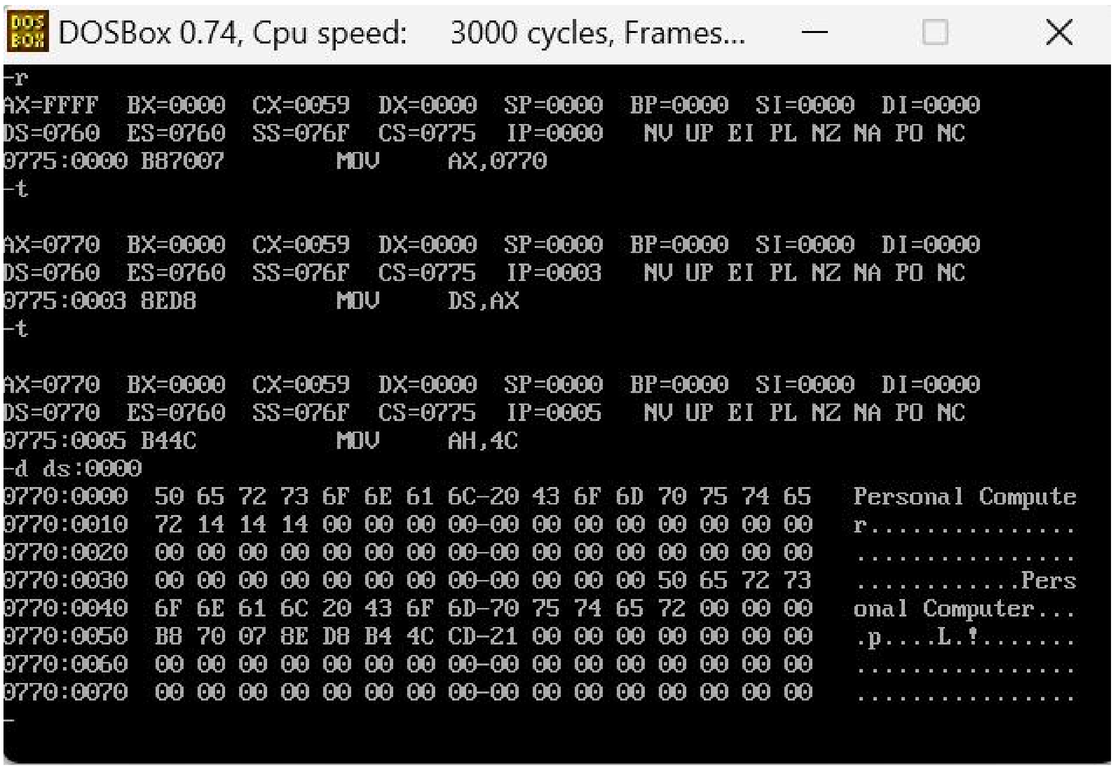
10h	0	FBH	FFh	-	-	-	-	-	-	....
01	10h	-5		?		?		?		

3.11



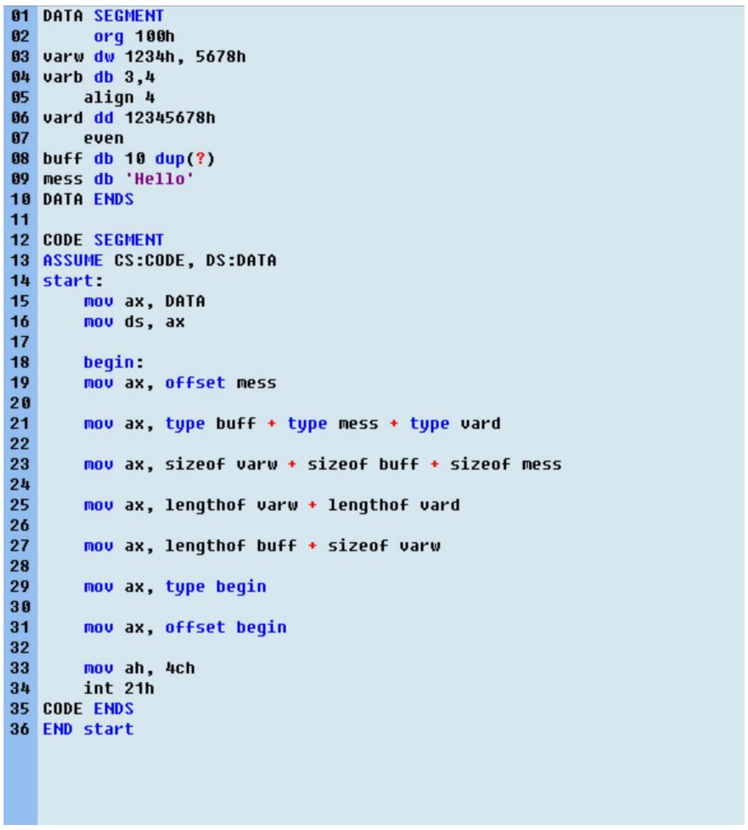
```
DATAS SEGMENT
    my1b DB 'Personal Computer'
    my2b DB 20
    my3b DB 14H ;十六进制的 20 为 14H
    my4b DB 00010100B ;二进制的 20
    my5w DW 20 DUP(?)
    my6c EQU 100
    my7c DB 'Personal Computer'
DATAS ENDS

STACKS SEGMENT
    ;此处输入堆栈段代码
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX
    ;此处输入代码段代码
    MOV AH,4CH
    INT 21H
CODES ENDS
    END START
```

通过 debug 可以看到，ds 数据段最开始是 Personal Computer，接着是十进制 20（即 14h），然后是十六进制的 20，和二进制的 20，随后是 20 个字节型未定义变量，然后是常量 100 最后是 Personal Computer。



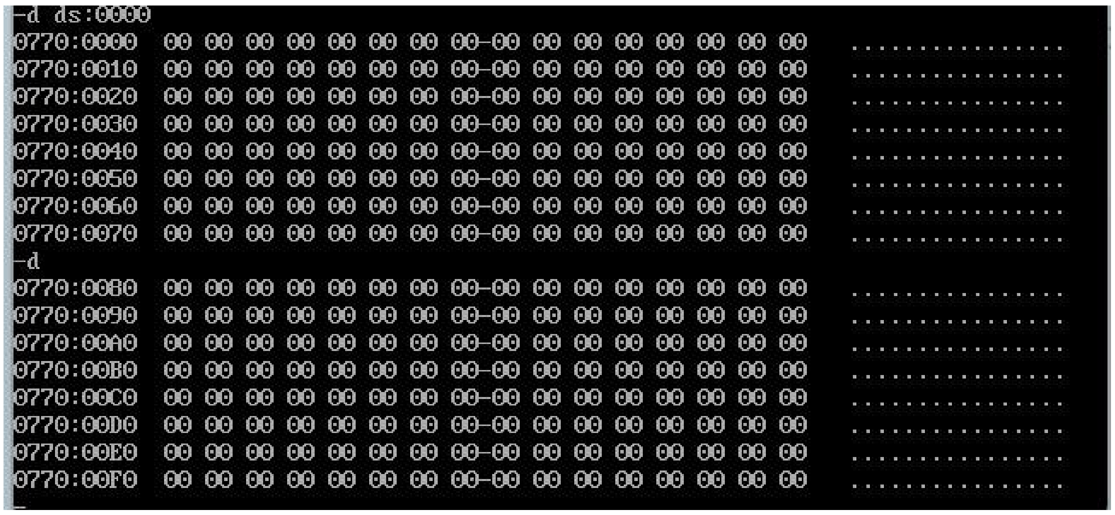
```
-r
AX=FFFF  BX=0000  CX=0059  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0760  ES=0760  SS=076F  CS=0775  IP=0000   NV UP EI PL NZ NA PO NC
0775:0000 B87007        MOV     AX,0770
-t

AX=0770  BX=0000  CX=0059  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0760  ES=0760  SS=076F  CS=0775  IP=0003   NV UP EI PL NZ NA PO NC
0775:0003 8ED8          MOV     DS,AX
-t

AX=0770  BX=0000  CX=0059  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0775  IP=0005   NV UP EI PL NZ NA PO NC
0775:0005 B44C          MOV     AH,4C
-d ds:0000
0770:0000  50 65 72 73 6F 6E 61 6C-20 43 6F 6D 70 75 74 65   Personal Compute
0770:0010  72 14 14 14 00 00 00 00-00 00 00 00 00 00 00 00   r...............
0770:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0030  00 00 00 00 00 00 00 00-00 00 00 00 50 65 72 73   ............Pers
0770:0040  6F 6E 61 6C 20 43 6F 6D-70 75 74 65 72 00 00 00   onal Computer...
0770:0050  B8 70 07 8E D8 B4 4C CD-21 00 00 00 00 00 00 00   .p....L.!.......
0770:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

3.15

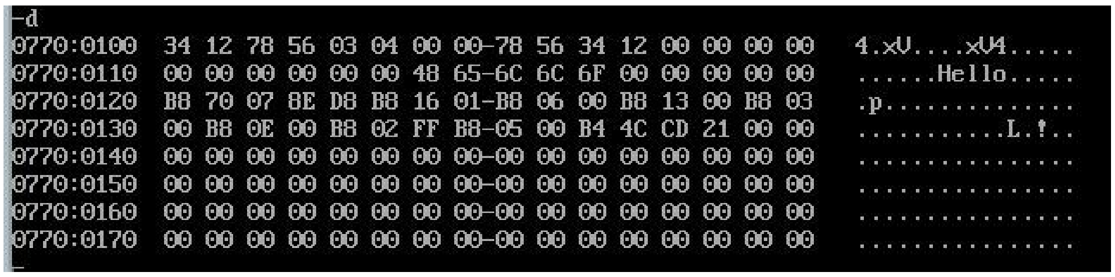


```
DATA SEGMENT
     org 100h
varw dw 1234h, 5678h
varb db 3,4
    align 4
vard dd 12345678h
    even
buff db 10 dup(?)
mess db 'Hello'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
start:
    mov ax, DATA
    mov ds, ax

    begin:
    mov ax, offset mess

    mov ax, type buff + type mess + type vard

    mov ax, sizeof varw + sizeof buff + sizeof mess

    mov ax, lengthof varw + lengthof vard

    mov ax, lengthof buff + sizeof varw

    mov ax, type begin

    mov ax, offset begin

    mov ah, 4ch
    int 21h
CODE ENDS
END start
```

内存情况：
一百个空数据

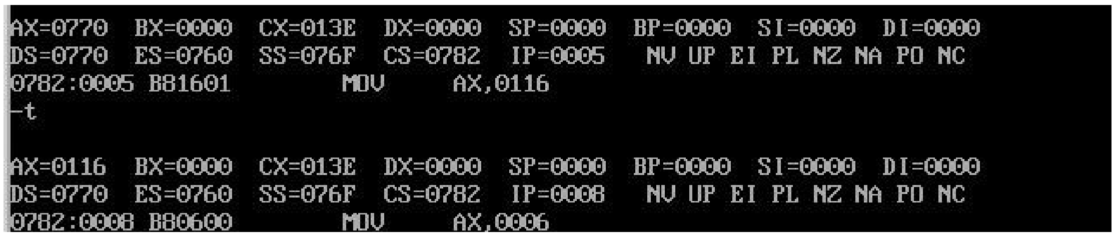

```
-d ds:0000
0770:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-d
0770:0080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:00F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

之后定义的数据，1234h，5678h ••••••

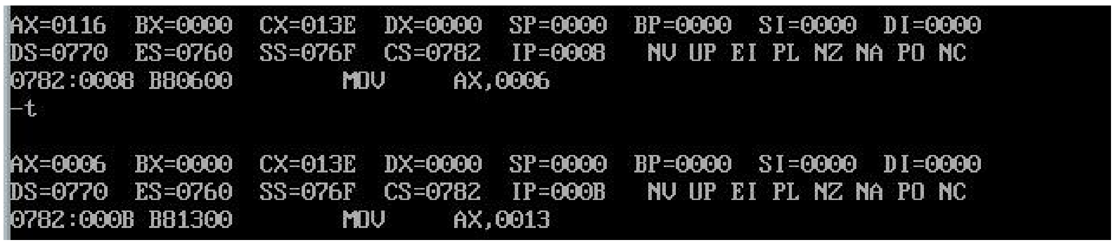

```
-d
0770:0100  34 12 78 56 03 04 00 00-78 56 34 12 00 00 00 00   4.xV....xV4.....
0770:0110  00 00 00 00 00 00 48 65-6C 6C 6F 00 00 00 00 00   ......Hello.....
0770:0120  B8 70 07 8E D8 B8 16 01-B8 06 00 B8 13 00 B8 03   .p..............
0770:0130  00 B8 0E 00 B8 02 FF B8-05 00 B4 4C CD 21 00 00   ...........L.!..
0770:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0770:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

begin:

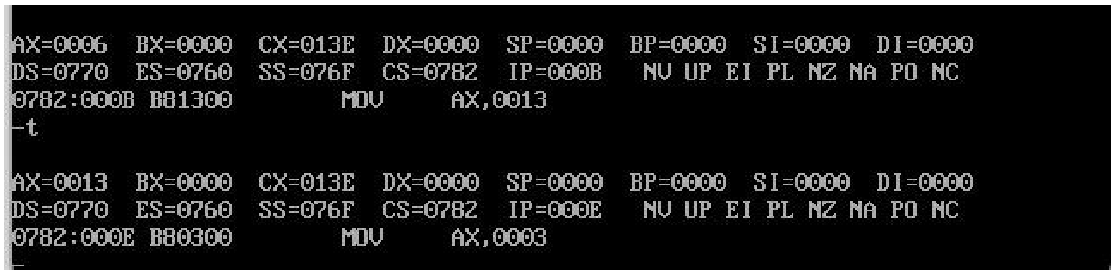
mov ax, offset mess 运算后 ax 是 0116h


```
AX=0770  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0005   NV UP EI PL NZ NA PO NC
0782:0005 B81601        MOV     AX,0116
-t

AX=0116  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0008   NV UP EI PL NZ NA PO NC
0782:0008 B80600        MOV     AX,0006
```

mov ax, type buff + type mess + type vard 运算完后 ax 是 6h， ax=1 + 1 + 4

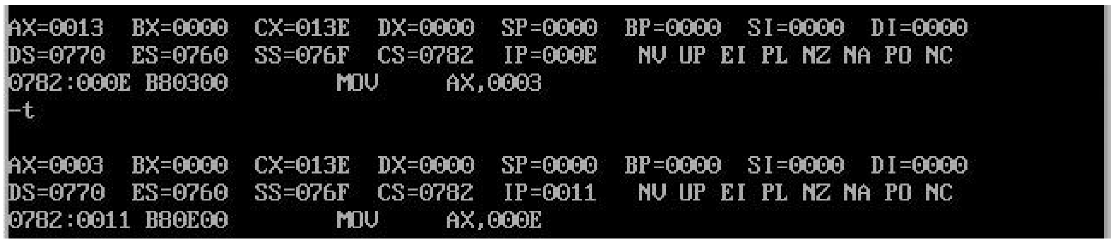

```
AX=0116  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0008   NV UP EI PL NZ NA PO NC
0782:0008 B80600        MOV     AX,0006
-t

AX=0006  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=000B   NV UP EI PL NZ NA PO NC
0782:000B B81300        MOV     AX,0013
```

mov ax, sizeof varw + sizeof buff + sizeof mess 运算完后 ax 是 13h，ax=4+10+5。



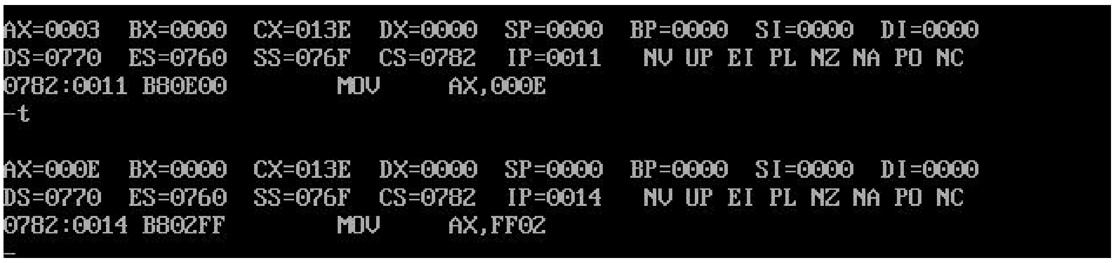
```
AX=0006  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=000B   NV UP EI PL NZ NA PO NC
0782:000B B81300        MOV     AX,0013
-t

AX=0013  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=000E   NV UP EI PL NZ NA PO NC
0782:000E B80300        MOV     AX,0003
```

mov ax, lengthof varw + lengthof vard 运算完后 ax 为 3h， ax=2+1。



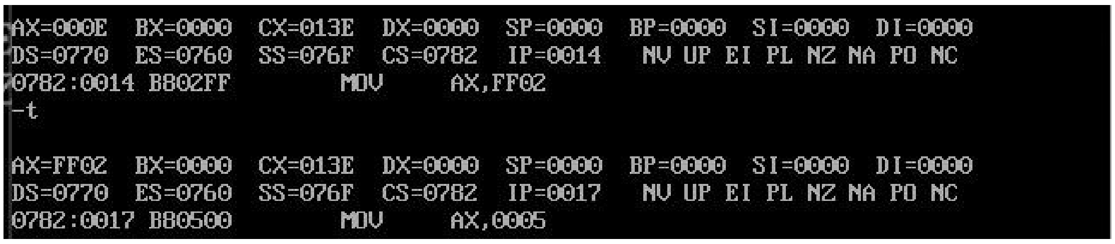
```
AX=0013  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=000E   NV UP EI PL NZ NA PO NC
0782:000E B80300        MOV     AX,0003
-t

AX=0003  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0011   NV UP EI PL NZ NA PO NC
0782:0011 B80E00        MOV     AX,000E
```

mov ax, lengthof buff + sizeof varw 运算完后 ax 为 0eh。



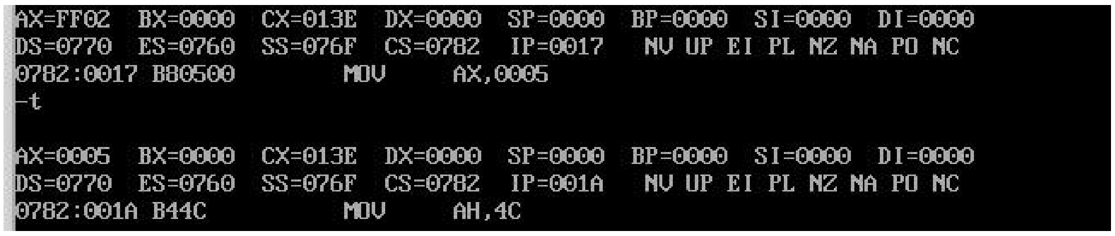
```
AX=0003  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0011   NV UP EI PL NZ NA PO NC
0782:0011 B80E00        MOV     AX,000E
-t

AX=000E  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0014   NV UP EI PL NZ NA PO NC
0782:0014 B802FF        MOV     AX,FF02
```

mov ax, type begin 运算完后 ax 为 0ff02h。



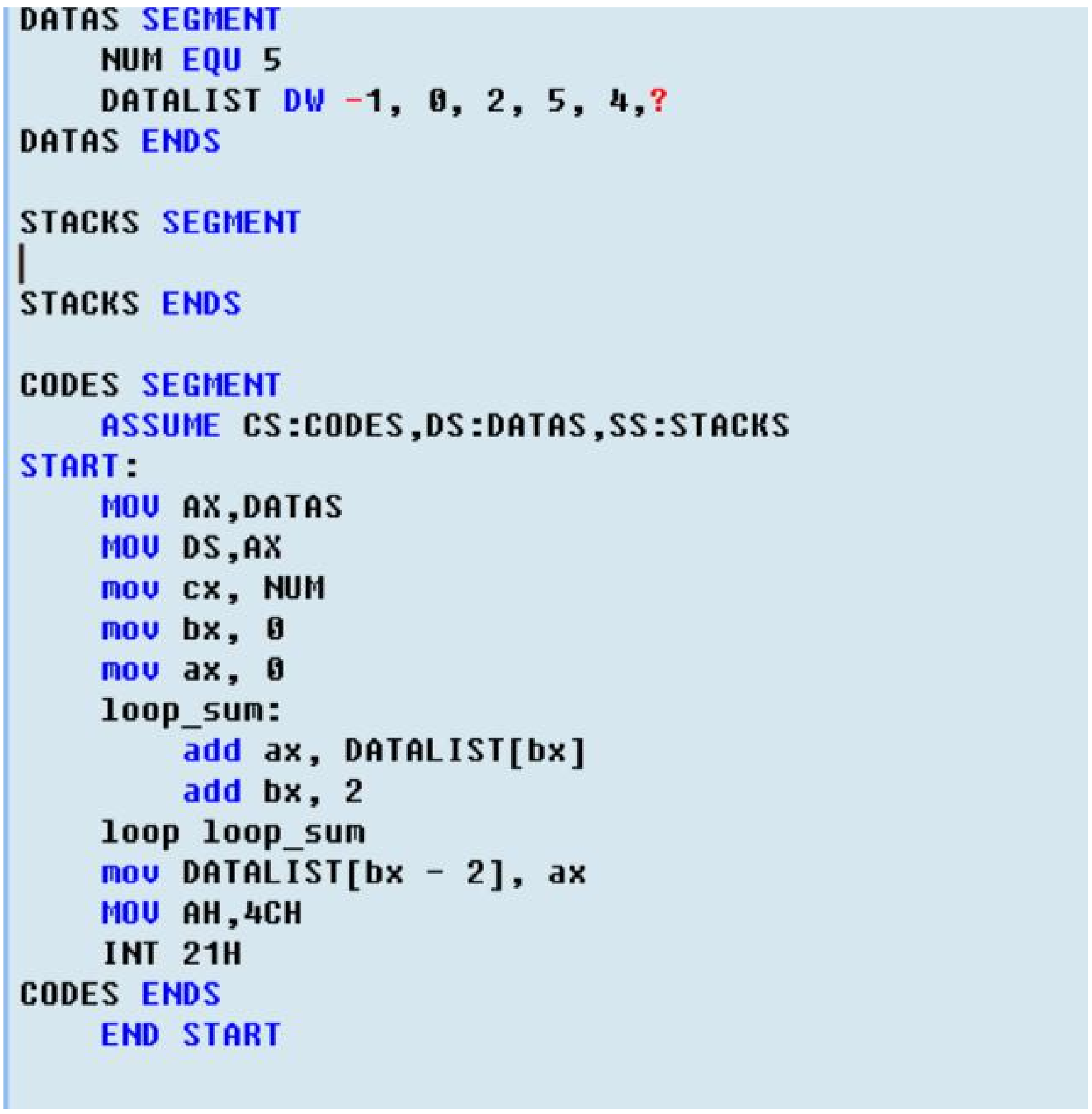
```
AX=000E  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0014   NV UP EI PL NZ NA PO NC
0782:0014 B802FF        MOV     AX,FF02
-t

AX=FF02  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0017   NV UP EI PL NZ NA PO NC
0782:0017 B80500        MOV     AX,0005
```

mov ax, offset begin 运算完后 ax 为 5h。



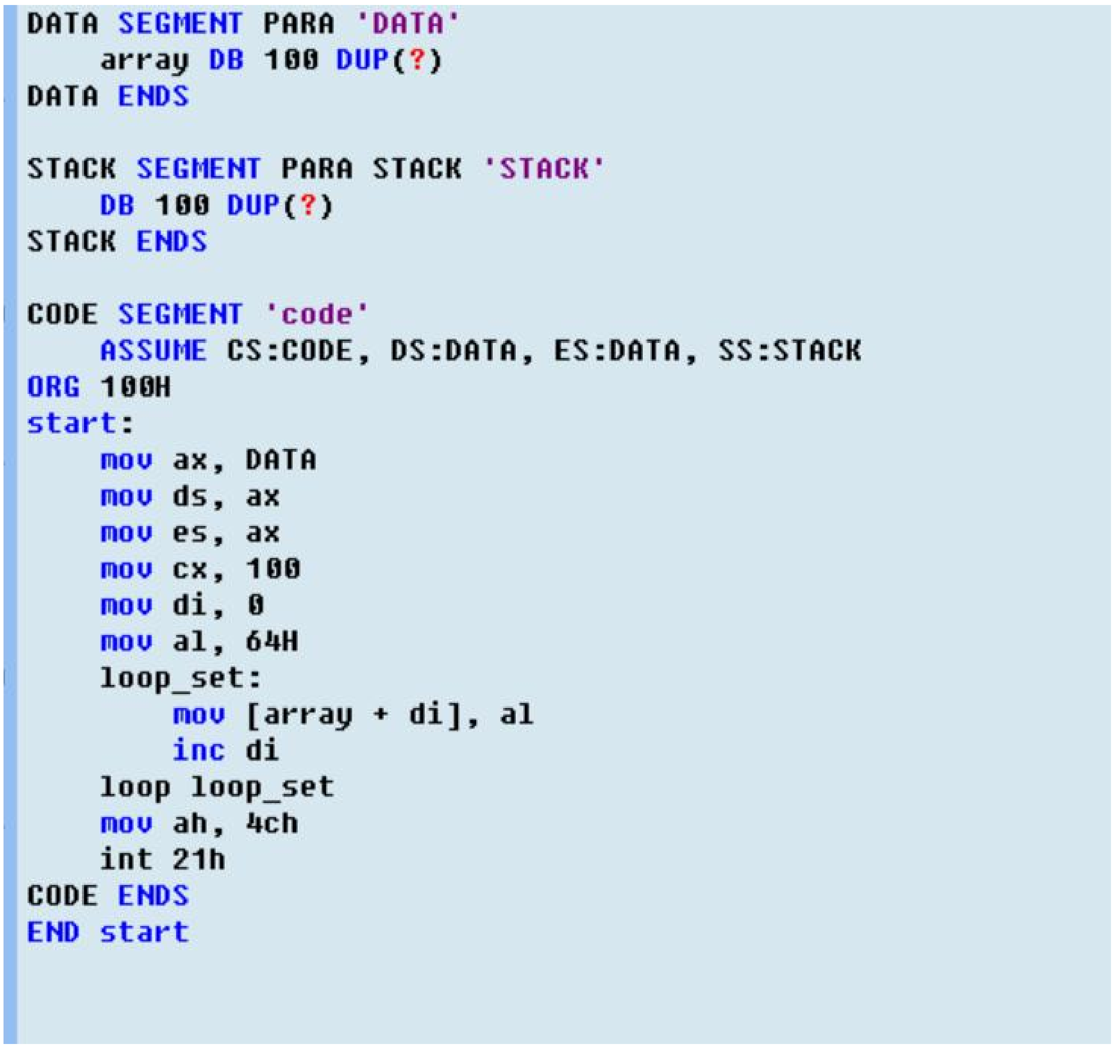
```
AX=FF02  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=0017   NV UP EI PL NZ NA PO NC
0782:0017 B80500        MOV     AX,0005
-t

AX=0005  BX=0000  CX=013E  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0770  ES=0760  SS=076F  CS=0782  IP=001A   NV UP EI PL NZ NA PO NC
0782:001A B44C          MOV     AH,4C
```

3.21

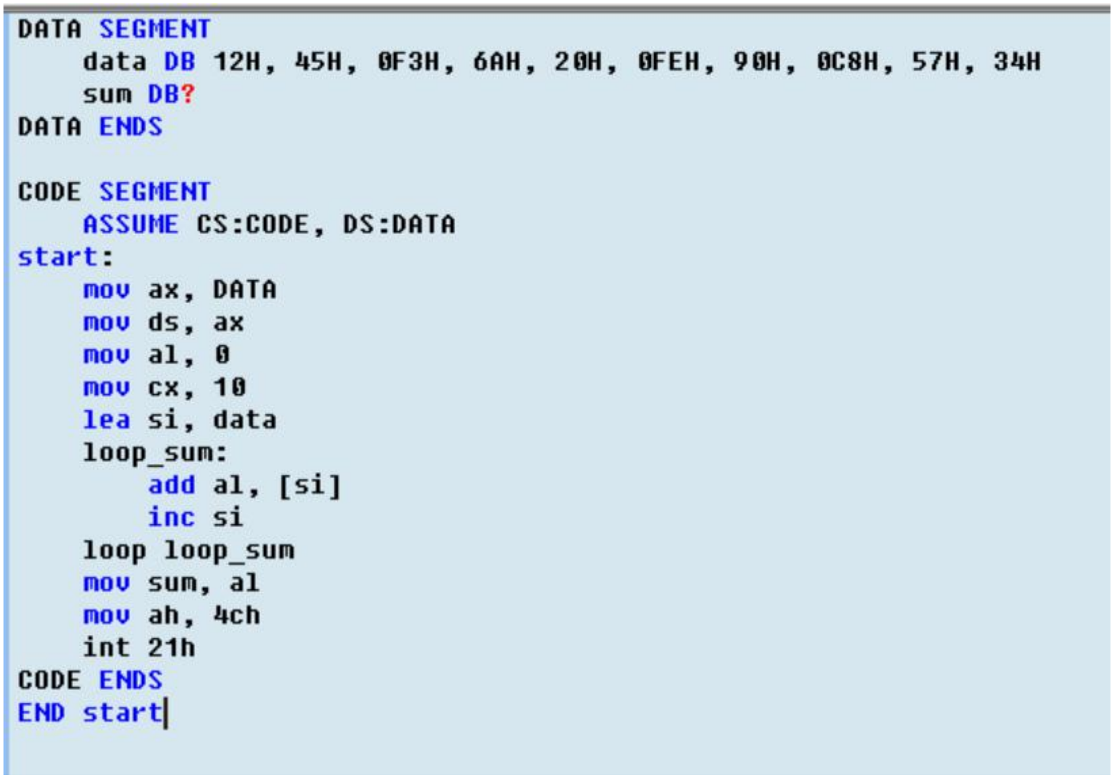


```
DATAS SEGMENT
    NUM EQU 5
    DATALIST DW -1, 0, 2, 5, 4,?
DATAS ENDS

STACKS SEGMENT

STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX
    mov cx, NUM
    mov bx, 0
    mov ax, 0
    loop_sum:
        add ax, DATALIST[bx]
        add bx, 2
    loop loop_sum
    mov DATALIST[bx - 2], ax
    MOV AH,4CH
    INT 21H
CODES ENDS
    END START
```

3.22


```
DATA SEGMENT PARA 'DATA'
    array DB 100 DUP(?)
DATA ENDS

STACK SEGMENT PARA STACK 'STACK'
    DB 100 DUP(?)
STACK ENDS

CODE SEGMENT 'code'
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
ORG 100H
start:
    mov ax, DATA
    mov ds, ax
    mov es, ax
    mov cx, 100
    mov di, 0
    mov al, 64H
    loop_set:
        mov [array + di], al
        inc di
    loop loop_set
    mov ah, 4ch
    int 21h
CODE ENDS
END start
```

3.24



```
DATA SEGMENT
    data DB 12H, 45H, 0F3H, 6AH, 20H, 0FEH, 90H, 0C8H, 57H, 34H
    sum DB?
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
start:
    mov ax, DATA
    mov ds, ax
    mov al, 0
    mov cx, 10
    lea si, data
    loop_sum:
        add al, [si]
        inc si
    loop loop_sum
    mov sum, al
    mov ah, 4ch
    int 21h
CODE ENDS
END start
```