



中南大學
CENTRAL SOUTH UNIVERSITY

第一单元 第四讲

汇编语言程序格式

盛 羽

中南大学计算机学院

shengyu@csu.edu.cn

1. 汇编和运行
2. 伪操作

■将汇编程序源代码转换为目标代码

- 将指令集的助记符（ Mnemonics ）翻译为操作码
- 解析符号名称（ Symbolic names ）成为存储器地址（ 偏移地址，具体数值 ）以及其它的实体
- 解析伪指令

- 将目标代码拼装成可执行程序
- 多模块程序



■ 开发工具

■ 源代码编辑

- 各种纯文本编辑工具
- 记事本、Sublime、Editplus、Notepad++、VS

■ 汇编和连接

- masm
- link

■ 调试程序

- debug

■ 64 位机器

- 需安装 Dosbox 等工具进行环境模拟

1. 汇编和运行

```
demo.asm
1  data segment ;数据段开始
2      x dw 5,14,20 ;存储空间申请 (变量定义)
3      y dw 6
4      z dw 7
5      w dw ?
6  data ends ;数据段结束
7
8  code segment ;代码段开始
9      main proc far ;主程序定义开始
10         assume cs:code,ds:data ;指定缺省段
11     start: ;整个代码开始
12         push ds ;缺省操作
13         sub ax,ax ;缺省操作
14         push ax ;缺省操作
15         mov ax,data ;将数据段起始地址通过ax赋给ds
16         mov ds,ax ;将数据段起始地址通过ax赋给ds
17         mov ax,x ;将内存以x值为偏移地址的单元的值赋给ax
18         add ax,y ;将内存以y为偏移地址的单元的值与ax相加并赋给ax
19         add ax,z ;将内存以z值为偏移地址的单元的值与ax相加并赋给ax
20         mov w,ax ;将ax寄存器的值赋给内存以z值为偏移地址的单元
21         ret ;主函数返回
22     main endp ;主程序定义结束
23 code ends ;代码段结束
24 end start ;代码结束
25
```

```
template.asm
data segment ;数据段开始

data ends ;数据段结束

code segment ;代码段开始
main proc far ;主程序定义开始
    assume cs:code,ds:data ;指定缺省段
start: ;整个代码开始
    push ds ;缺省操作
    sub ax,ax ;缺省操作
    push ax ;缺省操作
    mov ax,data ;将数据段起始地址通过ax赋给ds
    mov ds,ax ;将数据段起始地址通过ax赋给ds

    ret ;主函数返回
main endp ;主程序定义结束
code ends ;代码段结束
end start ;代码结束
```

1. 汇编和运行
2. 伪操作

2. 伪操作

■ 伪操作（伪指令）

```
data segment
    x dw 5,14,20
    y dw 6
    z dw 7
    w dw ?
data ends

code segment
    main proc far
        assume cs:code,ds:data
    start:
        push ds
        sub ax,ax
        push ax
        mov ax,data
        mov ds,ax
        mov ax,x
        add ax,y
        add ax,z
        mov w,ax
        ret
    main endp
code ends
end start
```

分
进
编
汇
性

```
0000      data segment
0000 0005 000E 0014      x dw 5,14,20
0006 0006      y dw 6
0008 0007      z dw 7
000A 0000      w dw ?
000C      data ends

0000      code segment
0000      main proc far
0000      assume cs:code,ds:data
0000      start:
0000 1E      push ds
0001 2B C0      sub ax,ax
0003 50      push ax
0004 B8 ---- R      mov ax,data
0007 8E D8      mov ds,ax
0009 A1 0000 R      mov ax,x
000C 03 06 0006 R      add ax,y
0010 03 06 0008 R      add ax,z
0014 A3 000A R      mov w,ax
0017 CB      ret
0018      main endp
0018      code ends
0018      end start
```

令都被处理了

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0500 0e00 1400 0600 0700 0000 0000 0000
1e2b c050 b800 008e d8a1 0000 0306 0600
0306 0800 a30a 00cb
```

■ 段定义伪操作

- 定义段，数据段、堆栈段、代码段
- 同一段的数据（代码）存在连续的内存空间中

■ 定义方法

段名 segment

; 内容

段名 endp

- 段名：程序员给段取的名字，尽量能反映其类别。如 data,data1,code,code1
- 成对出现
- 一个程序中可以有多个数据段，多个代码段

2. 伪操作

■ 过程定义伪操作

```
1  code segment
2  main proc far
3      assume cs:code
4  start:
5      push ds
6      sub ax,ax
7      push ax
8
9      mov ax,-5
10     call tmp
11     mov bx,ax
12     ret
13 main endp
14 tmp proc
15     neg ax
16     ret
17 tmp endp
18 code ends
19 end start
```

数

```
0000      code segment
0000      main proc far
      assume cs:code
0000      start:
0000 1E      push ds
0001 2B C0      sub ax,ax
0003 50      push ax
8
0004 B8 FFFB      mov ax,-5
0007 E8 000D R      call tmp
000A 8B D8      mov bx,ax
000C CB      ret
000D      main endp
000D      tmp proc
000D F7 D8      neg ax
000F C3      ret
0010      tmp endp
0010      code ends
      end start
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
1e2b c050 b8fb ffe8 0300 8bd8 cbf7 d8c3
```

2. 伪操作

■ 假定伪指令 `assume`

- 用于指定变量名的缺省段寄存器
- 右侧程序，`x,y,z` 变量所在的段名绑定给了 `es` 段寄存器，所以这几个变量段寄存器的值使用的是 `es` 段寄存器的值。（虽然程序中 `es` 段寄存器的值被赋予了 `d2` 段的起始地址）
- 如果同一段绑定同时指定到 `ds` 和 `es`，默认 `ds`

```
d2 segment
    a dw 4
d2 ends
d1 segment
    x dw 5,14,20
    y dw 6
    z dw 7
    w dw ?
d1 ends
code segment
    main proc far
        assume cs:code,es:d1,ds:d2
    start:
        push ds
        sub ax,ax
        push ax
        mov ax,d1
        mov ds,ax
        mov ax,d2
        mov es,ax
        mov ax,x
        add ax,y
        add ax,z
        mov w,ax

        mov bx,a
        ret
    main endp
code ends
end start
```

```
C:\CODE>debug demo.exe
-u
076E:0000 1E          PUSH    DS
076E:0001 2BC0          SUB     AX,AX
076E:0003 50          PUSH    AX
076E:0004 B86D07       MOV     AX,076D
076E:0007 8ED8       MOV     DS,AX
076E:0009 B86C07       MOV     AX,076C
076E:000C 8EC0       MOV     ES,AX
076E:000E 26          ES:
076E:000F A10000       MOV     AX,[0000]
076E:0012 26          ES:
076E:0013 03060600    ADD     AX,[0006]
076E:0017 26          ES:
076E:0018 03060800    ADD     AX,[0008]
076E:001C 26          ES:
076E:001D A30A00       MOV     [000A],AX
```

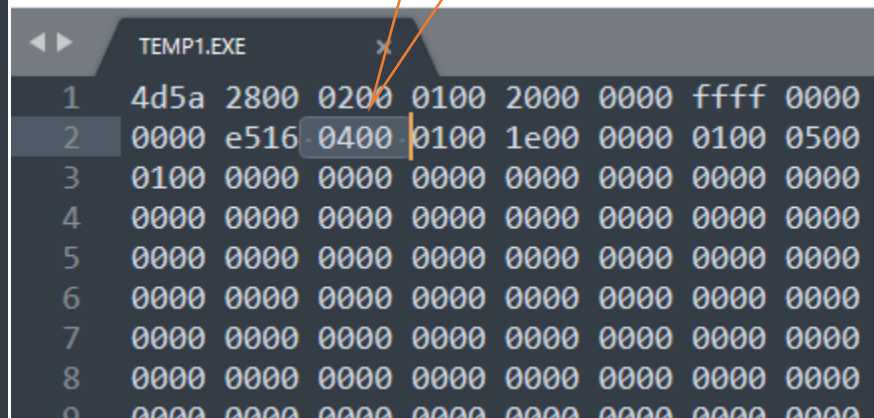
2. 伪操作

■ 伪操作符 start (或其他)

- 用于表示程序从那一行语句开始执行代码
- 名称自定义，但是需要与 end 相匹配
- 建议用 start
- 如果没有放在第一行？
- 一定要 main 过程吗？
 - 不需要，从 start 标识的语句开始执行

```
data segment
x dw 5,14,20
y dw 6
z dw 7
w dw ?
data ends
code segment
main proc far
    assume cs:code,ds:data
    push ds
    sub ax,ax
    push ax
start:
    mov ax,data
    mov ds,ax
    mov ax,x
    add ax,y
    add ax,z
    mov w,ax
    ret
main endp
code ends
end start
```

IP 寄存器初始值



	TEMP1.EXE								
1	4d5a	2800	0200	0100	2000	0000	ffff	0000	
2	0000	e516	0400	0100	1e00	0000	0100	0500	
3	0100	0000	0000	0000	0000	0000	0000	0000	
4	0000	0000	0000	0000	0000	0000	0000	0000	
5	0000	0000	0000	0000	0000	0000	0000	0000	
6	0000	0000	0000	0000	0000	0000	0000	0000	
7	0000	0000	0000	0000	0000	0000	0000	0000	
8	0000	0000	0000	0000	0000	0000	0000	0000	
9	0000	0000	0000	0000	0000	0000	0000	0000	

■ 数据定义及存储器分配伪操作

[变量名] 数据类型 操作数 1 , 操作数 2 , 操作数 3

■ 变量名 , 可选 , 仅用于更便捷的访问内存单元

■ 数据类型 : DB,DW,DD....

■ DB : 每个操作数单元占 1 个字节

■ DW : 每个操作数单元占 2 个字节

■ DD : 每个操作数单元占 4 个字节

■ 操作数 : 根据操作数进行内存单元的分配

1. 具体数值 : 分配一个内存单元 , 并给其赋予该数值为初始值

2. ? : 分配一个内存单元 , 不赋初始值

3. n dup(操作数) : 将操作数重复 n 遍 , 操作数规则同 1 、 2 、 3 , 可嵌套

■ n dup(1,10,100,?, m dup(?)) : $(1+1+1+1+m)*n$ 个内存单元

■数据定义及存储器分配伪操作

```
33  0500 0600 0700 0800 0009 0009 0000 0000
34  cdab cdab cdab cdab cdab cdab cdab cdab
35  030f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f
36  6162 6364 6162 6261 7856 3412 0000 0000
37  1e2b c050 b800 008e d8b8 0100 8ec0 a100
38  0003 0602 0003 0604 00a3 0700 b402 b24f
39  cd21 b24b cd21 cb
```

■数据定义及存储器分配伪操作

```
push    ds
sub     ax,ax
push    ax
mov     ax,d1
mov     ds,ax
mov     ax,d2
mov     es,ax
mov     ax,x
mov     bx,x+1
mov     cx,x+2
mov     dx,y
mov     si,x+16
mov     di,x+64
```

33	0500	0600	0700	0800	0009	0009	0000	0000
34	cdab	cdab	cdab	cdab	cdab	cdab	cdab	cdab
35	030f	0f0f	0f0f	0f0f	0f0f	0f0f	0f0f	0f0f
36	6162	6364	6162	6261	7856	3412	0000	0000
37	1e2b	c050	b800	008e	d8b8	0100	8ec0	a100
38	008b	1e01	008b	0e02	008b	1602	008b	3610
39	008b	3e40	00b4	02b2	4fcd	21b2	4bcd	21cb

```
AX=0005 BX=0600 CX=0006 DX=0006 SP=FFFC BP=0000 SI=ABCD DI=2B1E
DS=076C ES=076D SS=076B CS=0770 IP=0025  NU UP EI PL ZR NA PE NC
```

2. 伪操作

■ 数据定义及存储器分配伪操作

```
d1 segment
  x dw 5
  y dw 6
  z dw 7
  q db 8
  w dw ?
  p dw 9,9
d1 ends
d2 segment
  a dw 8 dup (0abcdh)
  b db 3,3 dup(5 dup(15))
  c1 db 'abcd'
  c2 db 'a','b'
  c3 dw 'ab'
  c4 dd 12345678h
d2 ends
```

```
mov x,0101h
mov q,33h
mov a+4,1234h
```

```
AX=0005 BX=0600 CX=0006 DX=0006 SP=FFFC BP=0000 SI=ABCD DI=2B1E
DS=076C ES=076D SS=076B CS=0770 IP=0025 NU UP EI PL ZR NA PE NC
0770:0025 C70600000101 MOV WORD PTR [0000],0101 DS:0000=0005
-t
```

```
AX=0005 BX=0600 CX=0006 DX=0006 SP=FFFC BP=0000 SI=ABCD DI=2B1E
DS=076C ES=076D SS=076B CS=0770 IP=002B NU UP EI PL ZR NA PE NC
0770:002B C606060033 MOV BYTE PTR [0006],33 DS:0006=08
-t
```

```
AX=0005 BX=0600 CX=0006 DX=0006 SP=FFFC BP=0000 SI=ABCD DI=2B1E
DS=076C ES=076D SS=076B CS=0770 IP=0030 NU UP EI PL ZR NA PE NC
0770:0030 26 ES:
0770:0031 C70604003412 MOV WORD PTR [0004],1234 ES:0004=ABCD
```

```
AX=0005 BX=0600 CX=0006 DX=0006 SP=FFFC BP=0000 SI=ABCD DI=2B1E
DS=076C ES=076D SS=076B CS=0770 IP=0037 NU UP EI PL ZR NA PE NC
0770:0037 B402 MOV AH,02
-d 076c:0000
076C:0000 01 01 06 00 07 00 33 00-00 09 00 09 00 00 00 00 .....3.....
076C:0010 CD AB CD AB 34 12 CD AB-CD AB CD AB CD AB CD AB ....4.....
076C:0020 03 0F 0F 0F 0F 0F 0F 0F-0F 0F 0F 0F 0F 0F 0F .....
076C:0030 61 62 63 64 61 62 62 61-78 56 34 12 00 00 00 00 abcdabbaxU4....
076C:0040 1E 2B C0 50 B8 6C 07 8E-D8 B8 6D 07 8E C0 A1 00 .+.P.l....m....
076C:0050 00 8B 1E 01 00 8B 0E 02-00 8B 16 02 00 8B 36 10 .....6.
076C:0060 00 8B 3E 40 00 C7 06 00-00 01 01 C6 06 06 00 33 ..>@.....3
076C:0070 26 C7 06 04 00 34 12 B4-02 B2 4F CD 21 B2 4B CD &....4....0.!.K.
```


2. 伪操作

■ 数据定义及存储器分配伪操作

■ 字符串定义

```
d1 segment  
  
    hello db 'Hello World!',0dh,0ah,'$'  
  
d1 ends
```

■ 字符串的访问

```
mov al,hello[5]
```


■ 强制属性操作符 PTR

```
data segment
    xb db 12,13,20
    yw dw 5678h
data ends
code segment
    main proc far
        assume cs:code,ds:data
    start:
        push ds
        sub ax,ax
        push ax
        mov ax,data
        mov ds,ax
        ;mov ax,xb ;会提示类型不匹配的警告,不要使用
        mov al,xb
        mov bx,word ptr xb

        ;mov cl,yw ;会提示类型不匹配的警告,不要使用
        mov cx,yw
        mov dl,byte ptr yw

        mov si,1
        ;mov [si],0abh;直接这样写, 报错: 操作数必须明确类型
        mov [si],byte ptr 0abh;
        add si,1
        mov [si],word ptr 0cdh;
```

DW...) 等进行数据存取

```
AX=070C BX=0D0C CX=5678 DX=0078 SP=FFFC BP=0000 SI=0002 DI=0000
DS=076C ES=075C SS=076B CS=076D IP=0025  NU UP EI PL NZ NA PO NC
```

```
-d 076c:0000
076C:0000  0C AB CD 00 56 00 00 00-00 00 00 00 00 00 00 00
076C:0010  1E 2B C0 50 B8 6C 07 8E-DB A0 00 00 8B 1E 00 00
076C:0020  8B 0E 03 00 8A 16 03 00-BE 01 00 C6 04 AB 83 C6
076C:0030  01 C7 04 CD 00 CB 5D C3-B8 80 01 50 FF 76 04 E8
076C:0040  F6 73 82 04 04 0B 16 B8-40 75 20 02 2B 12 02 10
```

■ 符号常量

■ “=”

- 符号名 = 数值表达式
- 数值表达式：汇编器负责解析，在汇编阶段（生成 obj 之前）能够得出具体数值
- 为编程方便，某些常数建议设为常量，如

count=10

mov cl,count

■ 如

■ count=count * 2

■ “EQU”

- 与 “=” 类似
- 不能重复定义

2. 伪操作

■ 符号常量

■ “\$”

■ 表示当前

list BYTE

ListSize =

```
d1 segment
    list db 10,20,30,40
    listSize=$-list
    array dw 10,20,30,$,$+2,60
    arraySize=$-array
d1 ends

code segment
    main proc far
        assume cs:code,ds:d1
    start:
        push ds
        sub ax,ax
        push ax
        mov ax,d1
        mov ds,ax

        mov cx,listSize
        arraySize=arraySize/2
        mov cx,arraySize
```

```
0000      d1 segment
0000  0A 14 1E 28      list db 10,20,30,40
= 0004      listSize=$-list
0004  000A 0014 001E 000A R      array dw 10,20,30,$,$+2,60
      000E R 003C
= 000C      arraySize=$-array

0010      d1 ends
0000      code segment
0000      main proc far
      assume cs:code,ds:d1
0000      start:
0000  1E      push ds
0001  2B C0      sub ax,ax
0003  50      push ax
0004  B8 ---- R      mov ax,d1
0007  8E D8      mov ds,ax

0009  B9 0004      mov cx,listSize
= 0006      arraySize=arraySize/2
000C  B9 0006      mov cx,arraySize
```