第四章　汇编语言程序格式

清華大學
Tsinghua University

主要内容：

- 汇编程序功能
- 伪操作
- 汇编语言程序格式
- 汇编语言程序的上机过程

1. 汇编程序功能

汇编程序的主要功能：

- 检查源程序，给出出错信息
- 产生目标文件(.obj)和列表文件(.lst)
- 展开宏指令

2. 伪操作（伪指令）

是汇编程序对源程序进行汇编时处理的操作，完成处理器选择、存储模式定义、数据定义、存储器分配、指示程序开始结束等功能。

- 处理器选择伪操作
- 段定义伪操作
- 程序开始和结束伪操作
- 数据定义及存储器分配伪操作
- 表达式赋值伪操作
- 地址计数器与对准伪操作
- 基数控制伪操作

处理器选择伪操作：

.8086　选择 8086 指令系统　默认

.286　选择 80286 指令系统

.286P　选择保护模式下的 80286 指令系统

.386　选择 80386 指令系统

.386P　选择保护模式下的 80386 指令系统

.486　选择 80486 指令系统

.486P　选择保护模式下的 80486 指令系统

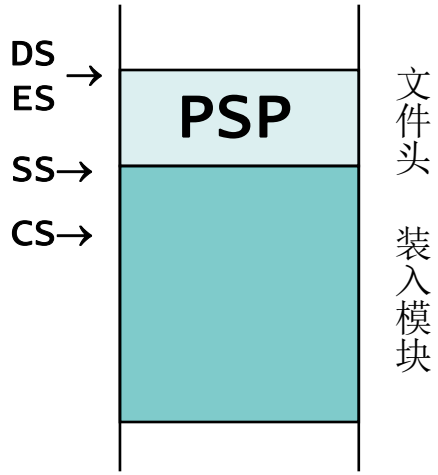
.586　选择 Pentium 指令系统

.586P　选择保护模式下的 Pentium 指令系统

段定义伪操作：

完整的段定义伪操作

```
data    segment                 ;定义数据段
        …
data    ends
extra   segment                 ;定义附加段
        …
extra   ends
code    segment                 ;定义代码段
        assume cs:code, ds:data,
es:extra
start:
        mov    ax, data
        mov    ds, ax           ;段地址 → 段寄存器
        …
code    ends
        end    start
```



EXE 程序的内存映象图

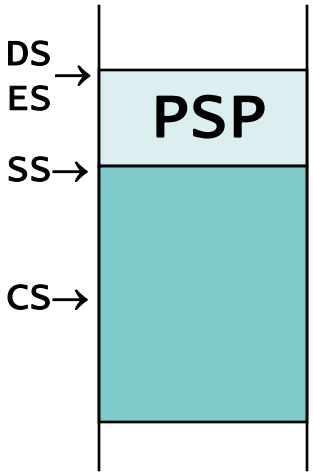
汇编源程序的其它形式：

```
……
code   segment
main   proc   far
       assume ……
start:
       push   ds
       mov    ax, 0
       push   ax
       ……
       ret
main   endp
code   ends
       end    start
```

```
……
code   segment
main   proc   far
       assume ……
start:
       ……
       ……
       mov    ax,4c00h
       int    21h
main   endp
code   ends
       end    start
```

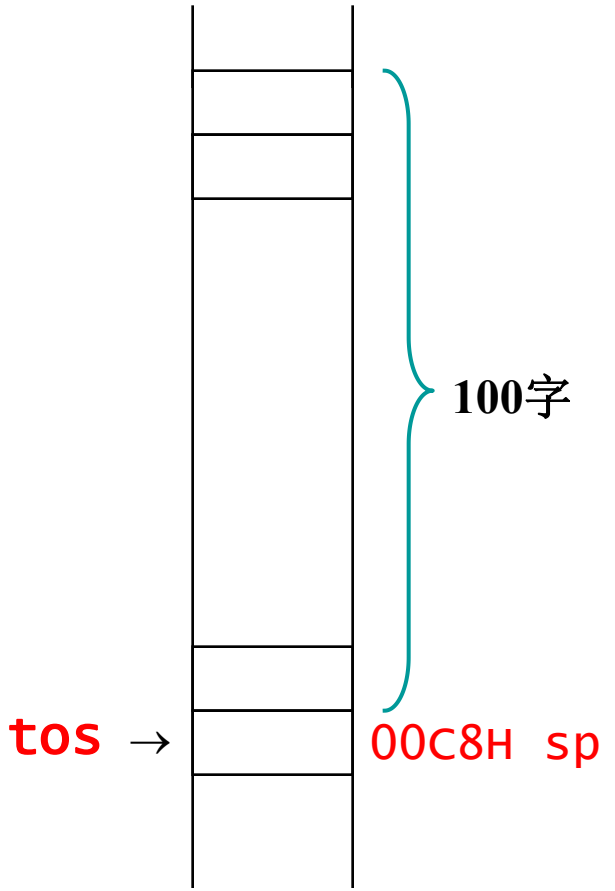
带堆栈段的汇编源程序：

```
data    segment
        ……
data    ends
stack   segment
        dw      100     dup (?)
        tos     label   word
stack   ends
code    segment
main    proc    far
        assume  cs:code, ds:data, ss:stack
start:
        mov     ax,  stack
        mov     ss,  ax
        mov     sp,  offset  tos
        push    ds
        sub     ax,  ax
        push    ax
        mov     ax,  data
        mov     ds,  ax
        ……
        ret
main    endp
code    ends
        end     start
```



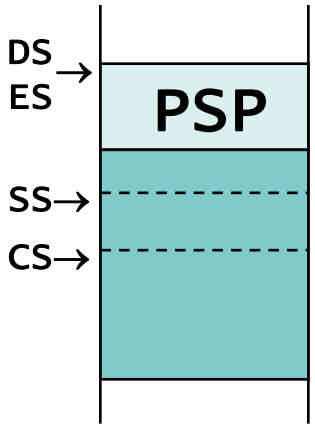
堆栈段的情况：

```
……
stack   segment
        dw      100     dup (?)
        tos     label   word
stack   ends
……
        mov     ax,  stack
        mov     ss,  ax
        mov     sp,  offset  tos

        push    ds
        sub     ax,  ax
        push    ax
……
```

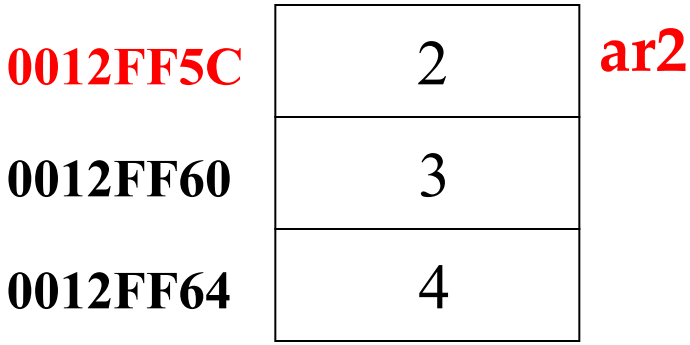


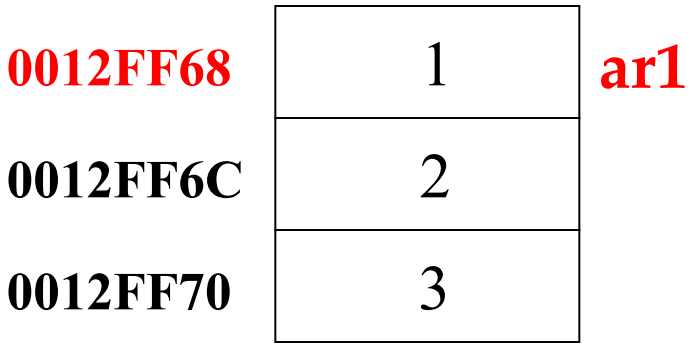

```
LINK : warning L4021: no stack segment
```

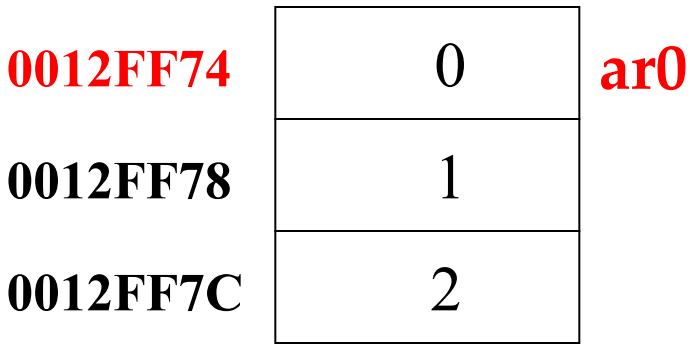
```
data    segment
        ……
data    ends

stack   segment   stack
        dw        100      dup (?)
        tos       label    word
stack   ends

code    segment
……
start:
        push      ds
        sub       ax,   ax
        push      ax
        mov       ax,   data
        mov       ds,   ax
        ……
        ret
……
code    ends
        end       start
```



```
int main()
{    int ar0[]={0,1,2};    int ar1[]={1,2,3};    int ar2[]={2,3,4};
     int *p[]={ar0,ar1,ar2};
……
}
```







完整段定义的格式:

段名　**SEGMENT**　[定位类型] [组合类型] [使用类型] [‘类别’]

……

……　　; 语句序列

段名　**ENDS**

(1) 定位类型 align_type

PARA　BYTE　WORD　DWORD　PAGE

(2) 组合类型 combine_type

PRIVATE　PUBLIC　COMMON　STACK　AT exp

(3) 使用类型 use_type

USE16　USE32

(4) 类别 ‘class’

例：

```
; proadd1.asm

        extrn   proadd : far
data    segment common
        ary     dw  1,2,3,4,5,6,7,8,9,10
        count   dw  10
        sum     dw  ?
data    ends

code1   segment
main    proc    far
        assume  cs:code1, ds:data
start:  push    ds
        sub     ax, ax
        push    ax
        mov     ax, data
        mov     ds, ax
        call    far ptr proadd
        ret
main    endp
code1   ends
        end     start
```

```
; proadd2.asm

public  proadd
data    segment  common
        ary      dw  1,2,3,4,5,6,7,8,9,10
        count    dw  10
        sum      dw  ?
data    ends

code2   segment
proadd  proc   far
        assume cs:code2,ds:data
        mov    ax, data
        mov    ds, ax
        ……
proadd  endp
code2   ends
        end
```

存储模式与简化段定义伪操作

(1) MODEL 伪操作

. MODEL 存储模式 [,语言类型] [,操作系统类型] [,堆栈选项]

存储模式：tiny small medium compact large huge flat

```
.model    small
.stack    100H
.data
    ……
.code
.startup
    ……
.exit 0
end
```

(2) 简化的段定义伪操作

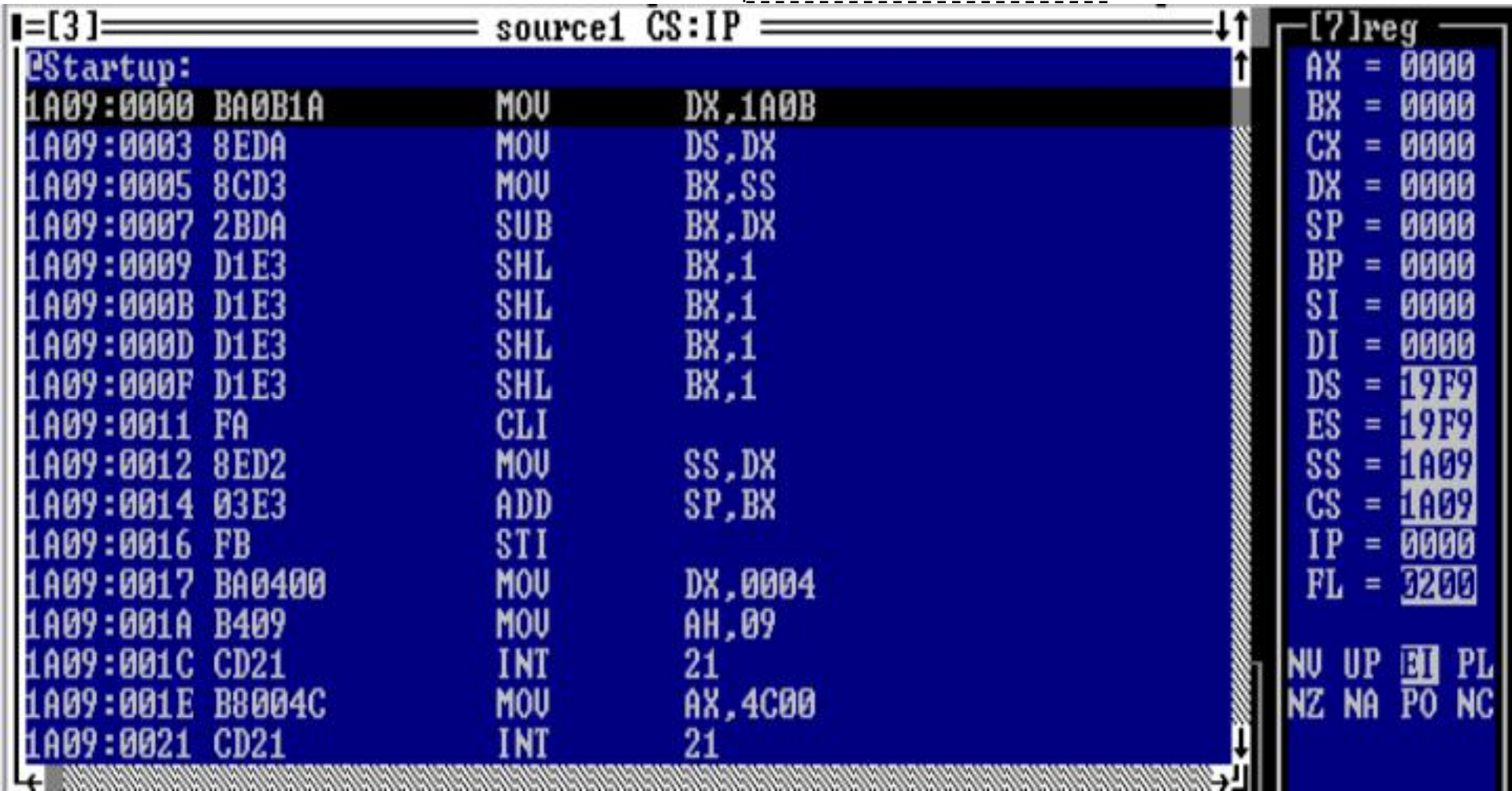
```
.code [name]
.data
.data?
.fardata [name]
.fardata? [name]
.const
.stack [size]
```

```
.model small
.data
……
.code
.startup
    ……
.exit 0
end
```

```
        .model small
        .data
        ……
        .code
start:  mov  ax, @data
        mov  ds, ax
        ……
        mov  ax, 4c00h
        int  21h
        end  start
```

```
        .model small
        .const
        ……
        .data
        ……
        .code
start:  mov  ax, DGROUP
        mov  ds, ax
        ……
        mov  ax, 4c00h
        int  21h
        end  start
```

```
.model small
.data
……
.code
.startup
    ……
.exit 0
end
```



段组定义伪操作

```
dseg1   segment   word   public   'data'
        ……
dseg1   ends
dseg2   segment   word   public   'data'
        ……
dseg2   ends

datagroup   group   dseg1, dseg2

cseg    segment   para   public   'code'
        assume    cs:cseg, ds:datagroup
start:
        mov   ax, datagroup
        mov   ds, ax
        ……
        mov   ax, 4c00h
        int   21h
cseg    ends
        end   start
```

```
data1  segment  word
       const1   dw    100
data1  ends

data2  segment  word
       var1     dw    ?
data2  ends

datagroup  group  data1, data2

code   segment
       assume  cs:code, ds:datagroup

start:
       mov   ax, datagroup
       mov   ds, ax

       mov   ax, const1
       mov   var1, ax

       mov   bx, offset var1     ; (bx) = 2
       mov   bx, offset data1    ; (bx) = 2
       mov   bx, offset data2    ; (bx) = 4

       assume  ds:data2
       mov   ax, data2
       mov   ds, ax
       mov   ax, var1
       mov   bx, offset var1     ; (bx) = 2

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

```
data1  segment
       const1   dw    100
data1  ends

data2  segment
       var1     dw    ?
data2  ends

datagroup  group  data1, data2

code   segment
       assume  cs:code, ds:datagroup

start:
       mov  ax, datagroup
       mov  ds, ax

       mov  ax, const1
       mov  var1, ax

       mov  bx, offset var1      ; (bx) = 0
       mov  bx, offset data1     ; (bx) = 2
       mov  bx, offset data2     ; (bx) = 2

       assume  ds:data2
       mov  ax, data2
       mov  ds, ax
       mov  ax, var1
       mov  bx, offset var1      ; (bx) = 0

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

程序开始和结束伪操作：

TITLE text

NAME module_name

END [label]

. STARTUP

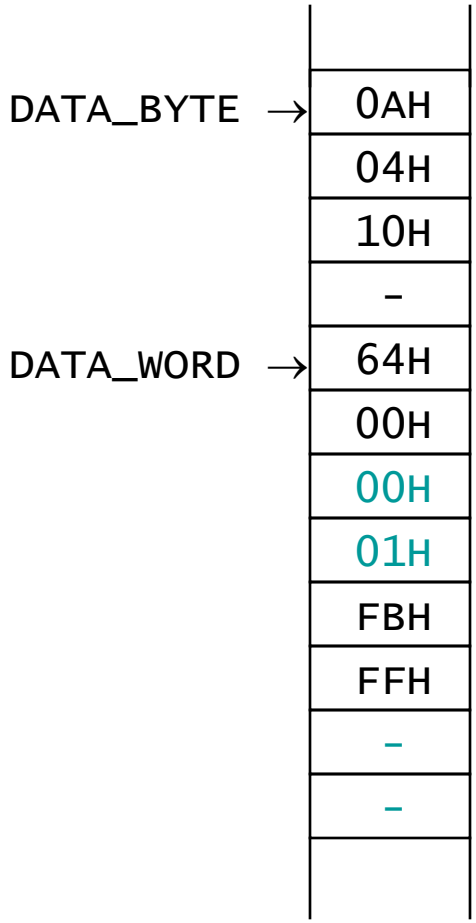
. EXIT [return_value]

数据定义及存储器分配伪操作：

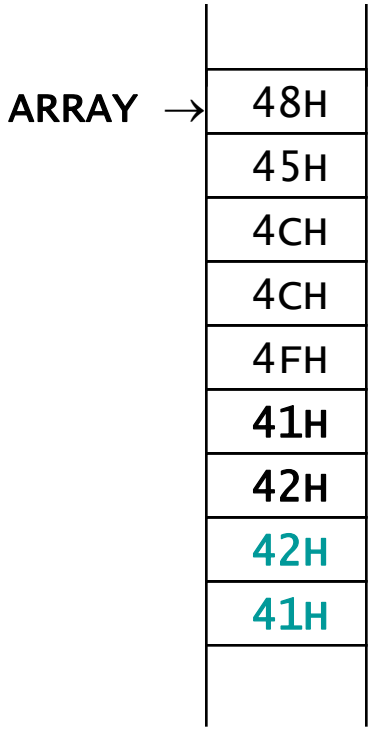
[变量] 助记符 操作数 [,操作数,…] [;注释]

助记符：DB DW DD DF DQ DT

```
DATA_BYTE   DB   10, 4, 10H, ?
DATA_WORD   DW   100, 100H, -5, ?
```



```
ARRAY   DB   'HELLO'
        DB   'AB'
        DW   'AB'
```



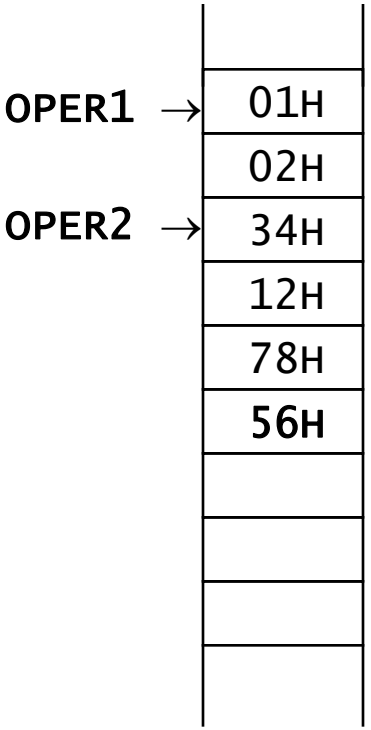

```
        PAR1   DW   100,200
        PAR2   DW   300,400
ADDR_TABLE   DW   PAR1,PAR2
```

```
VAR   DB   100 DUP (?)
      DB   2 DUP (0,2 DUP(1,2),3)
```

```
OPER1   DB   ?, ?
OPER2   DW   ?, ?
        ……
        MOV   OPER1, 0     ;字节指令
        MOV   OPER2, 0     ;字指令
```

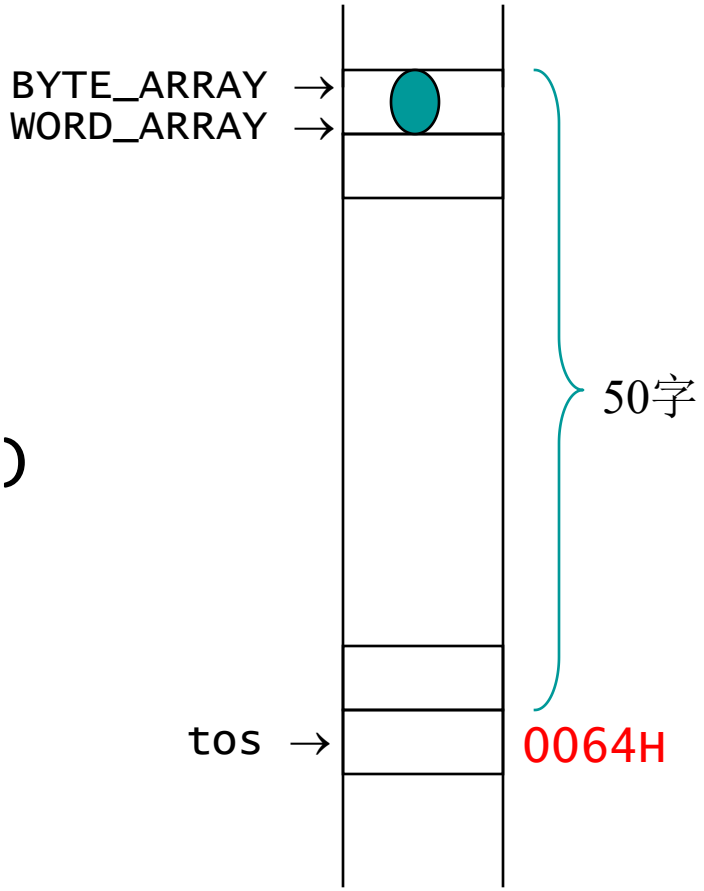
```
OPER1   DB   1, 2
OPER2   DW   1234H, 5678H
        ……
        MOV   AX, OPER1+1    ×
        MOV   AL, OPER2      ×    类型不匹配

        MOV   AX, WORD PTR   OPER1+1
        MOV   AL, BYTE PTR   OPER2
```

(AX)=3402H　　(AL)=34H



LABEL 伪操作： name LABEL type

```
BYTE_ARRAY   LABEL   BYTE
WORD_ARRAY   DW 50   DUP (?)
        tos  LABEL   WORD
```



表达式赋值伪操作：

表达式名　EQU　表达式

```
B        EQU   [BP+8]
ALPHA    EQU   9
BETA     EQU   ALPHA+18
```

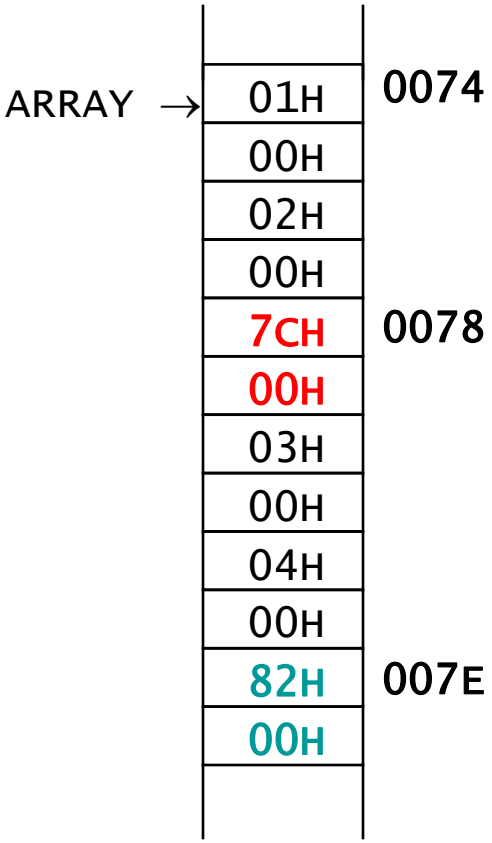
“＝”伪操作 （允许重复定义）

```
.....
EMP  =  7
.....
EMP  =  EMP  +  1
.....
```

地址计数器与对准伪操作：

地址计数器 $ ： 保存当前正在汇编的指令的地址

```
ORG    $+8          ; 跳过 8 个字节的存储区
JNE    $+6          ; 转向地址是 JNE 的首址 + 6
```



$ 用在伪操作的参数字段：

表示地址计数器的当前值

```
ARRAY  DW  1,2,$+4,3,4,$+4
```

ORG 伪操作：

```
SEG1    SEGMENT
        ORG     10
        VAR1    DW    1234H
        ORG     20
        VAR2    DW    5678H
        ORG     $+8
        VAR3    DW    1357H
SEG1    ENDS
```

```
BUFFER   LABEL   BYTE
         ORG     $+8
```

⟺

```
BUFFER   DB   8 DUP (?)
```

```
         ORG   100H
START:   .....
```

EVEN　　;使下一地址从偶地址开始

```
A   DB   'morning'
    EVEN
B   DW   2   DUP   (?)
```

ALIGN　boundary

```
        ALIGN   4
        ALIGN   2       ; EVEN
```

基数控制伪操作：

. RADIX 表达式　　；规定无标记数的基数

```
MOV   BX, 0FFH
MOV   BX, 178
```

```
.RADIX   16
MOV   BX, 0FF
MOV   BX, 178D
```

3. 汇编语言程序格式

源程序的每条语句可表示为：

[名字]	操作	操作数	[; 注释]
↓	↓	↓	↓
标号	指令	寄存器	说明程序或语句的功能
变量	伪指令	标号	
	宏指令	变量	
		常数	
		表达式	

标号/变量：段值、偏移量、类型三种属性

表达式：数字表达式　地址表达式

表达式操作符：

(1) **算术操作符：** +、−、*、/、mod

```
MOV   DX, BLOCK+(6-1)*2
```

```
ARRAY    DW    1,2,3,4,5,6,7
ARYEND   DW    ?
         MOV   CX, (ARYEND-ARRAY)/2
```

```
ADD   AX, BLOCK+2      ; 符号地址±常数  有意义
                       ; * / 时意义不明确
MOV   AX, BX+1         ; ×
```

(2) 逻辑和移位操作符： AND、OR、XOR、NOT、SHL、SHR

```
OPR1   EQU   25
OPR2   EQU   7
       AND   AX, OPR1 AND OPR2        ; AND AX, 1

MOV    AX,   0FFFFH SHL 2             ; MOV AX, 0FFFCH

IN     AL, PORT_VAL
OUT    PORT_VAL AND 0FEH, AL
```

(3) 关系操作符： EQ、NE、LT、LE、GT、GE

计算结果为逻辑值：真 0FFFFH

假 0000H

```
MOV FID, (OFFSET Y - OFFSET X)  LE  128
```

```
X:    ……

      ……

Y:    ……
```

若 ≤128 (真) 汇编结果： `MOV FID, -1`

若 >128 (假) 汇编结果： `MOV FID, 0`

(4) 数值回送操作符： OFFSET、SEG、TYPE、LENGTH、SIZE

OFFSET / SEG 变量 / 标号

功能：回送变量或标号的偏址 / 段址

TYPE 变量 / 标号 / 常数

DB	DW	DD	DF	DQ	DT	NEAR	FAR	常数
1	2	4	6	8	10	-1	-2	0

LENGTH 变量

功能：回送由 **DUP** 定义的变量的单元数，其它情况回送1

SIZE 变量

功能：**LENGTH * TYPE**

```
ARRAY   DW   100  DUP  (?)
TABLE   DB   'ABCD'

ADD  SI ,  TYPE    ARRAY      ; ADD  SI, 2
ADD  SI ,  TYPE    TABLE      ; ADD  SI, 1
MOV  CX ,  LENGTH  ARRAY      ; MOV  CX, 100
MOV  CX ,  LENGTH  TABLE      ; MOV  CX, 1
MOV  CX ,  SIZE    ARRAY      ; MOV  CX, 200
MOV  CX ,  SIZE    TABLE      ; MOV  CX, 1
```

(5) **属性操作符：** PTR、段操作符、SHORT、THIS、HIGH、LOW、HIGHWORD、LOWWORD

类型 PTR 表达式	MOV WORD PTR [BX], 5
段操作符	MOV ES:[BX], AL
SHORT 标号	JMP SHORT NEXT
THIS 类型	TA EQU THIS BYTE NEXT EQU THIS FAR

```
CONS   EQU   1234H
MOV    AH,   HIGH   CONS
MOV    AL,   LOW    CONS
```

第四章　上 机 过 程

清華大學
Tsinghua University

4. 汇编语言程序的上机过程

用 MASM 6.11 和 CodeView 汇编和调试汇编语言程序

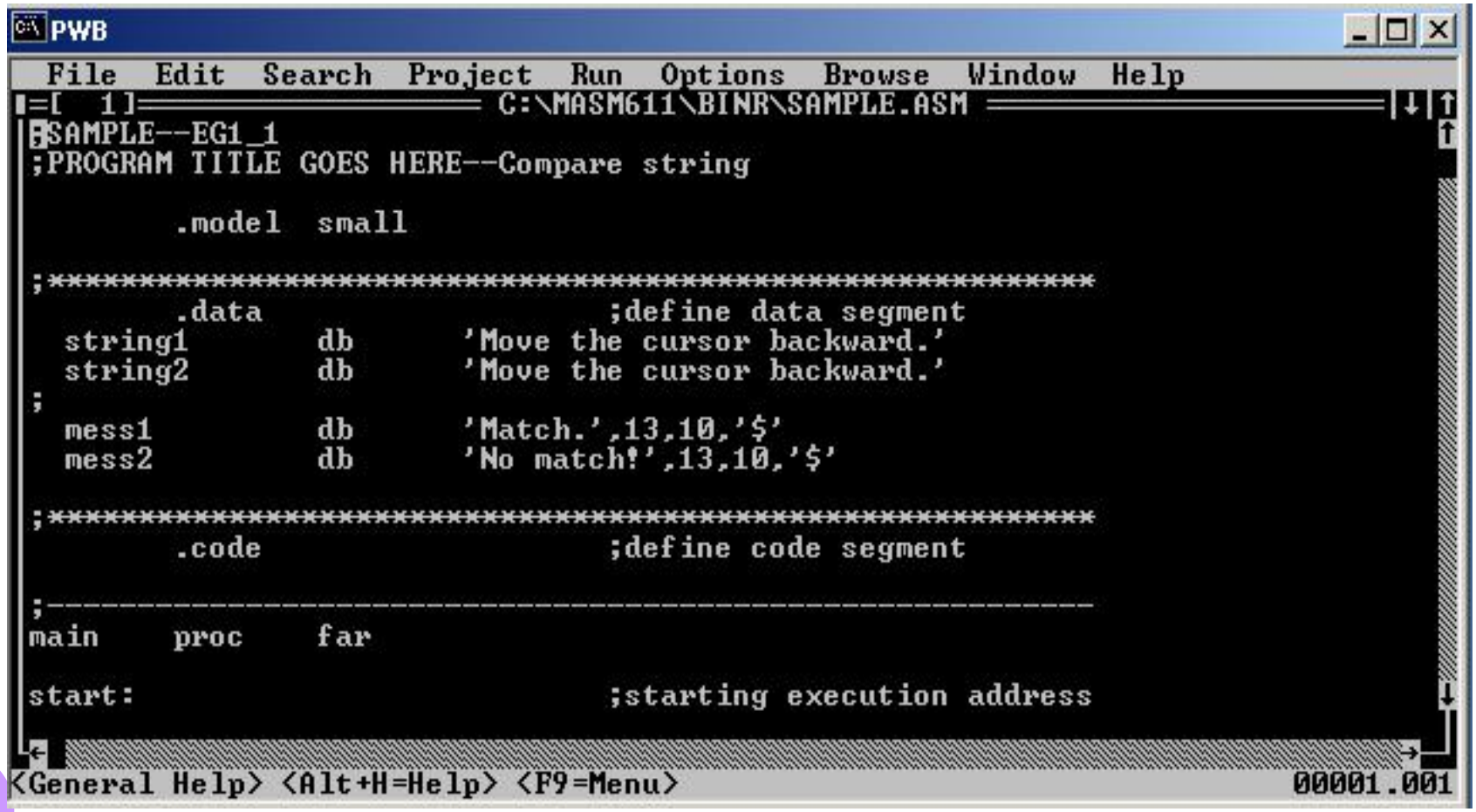
在 MASM 6.11 中，Microsoft 公司提供了程序员工作平台 PWB（programmer's work bench），这是一个集成化开发环境，程序员可以方便地完成汇编语言源程序的编辑、汇编、连接调试和执行等工作。

MASM 6 提供的调试工具是 CodeView，CodeView 是一个源码级调试工具，能支持 16 位和 32 位指令的调试。其用户界面是多窗口全屏幕的，用户可方便地使用菜单和鼠标来调试程序。

上机步骤：

1　设置环境变量

在 MASM611\BINR 的目录下，执行 new-vars.bat 即可建立环境参数和搜索路径，然后执行 pwb.exe 即可进入 PWB 的主窗口。用其文件 File 菜单中的 Exit 命令可退出 PWB。



2　用 File 菜单下的有关命令建立 asm 文件

3　用 Project 和 Options 菜单下的某些命令建立汇编语言程序运行环境

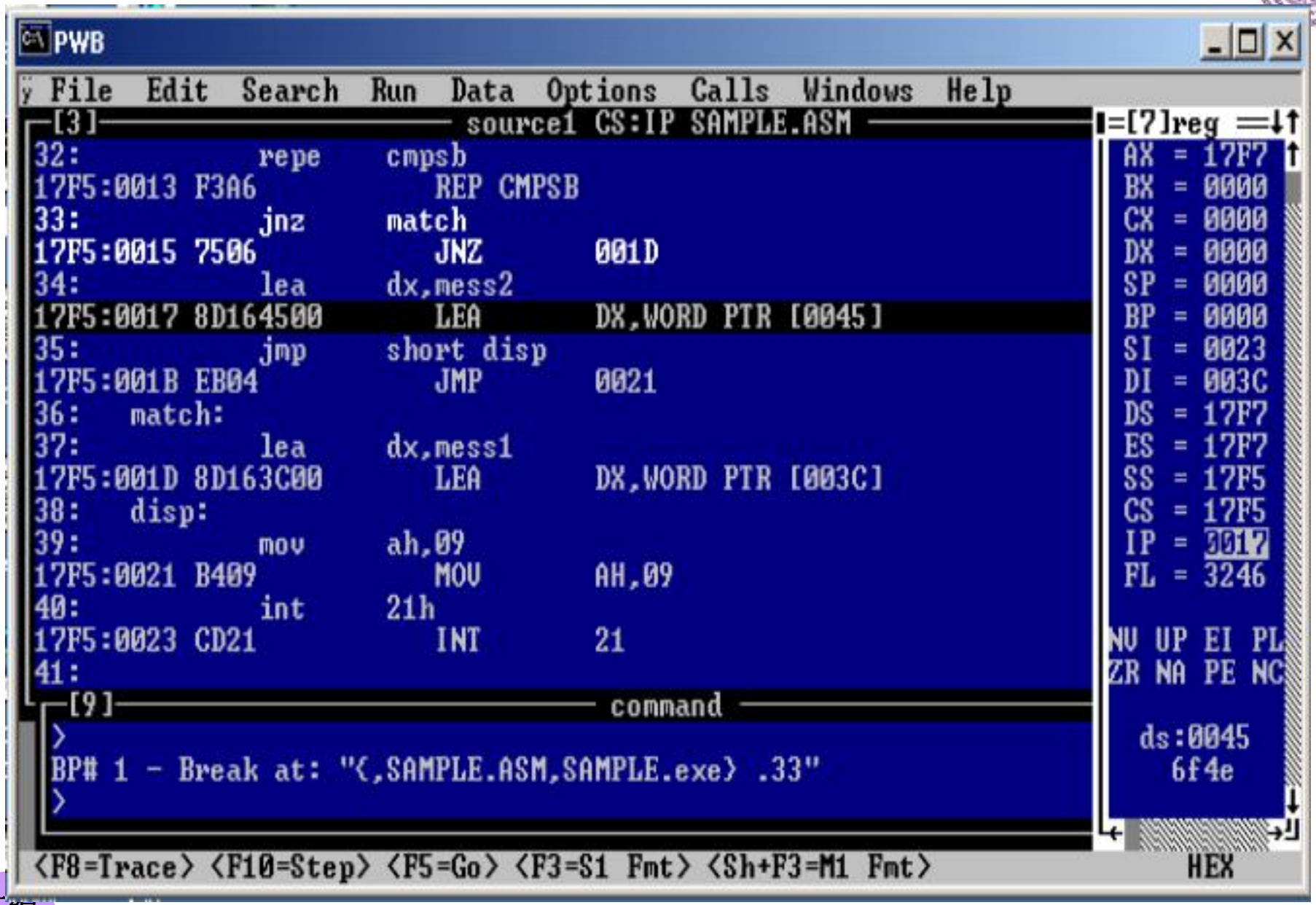
建立工程文件，设置所建立的可执行文件类型的 Build Options，设置编译器选项的 Language Options，设置连接器选项的 Link Options，设置调试器选项的 CodeView Options 等。由于用户使用时大部分选项都可采用 PWB 提供的默认值，因而不必作任何修改。但对于汇编语言程序，还有必要对 MASM 的选项作出某些选择。

4　用 Project 菜单下的有关命令对源文件进行汇编和连接

Compile File	编译（汇编）当前的源文件，生成目标文件
Build	连接目标文件，生成可执行文件
Build All	汇编当前源文件，连接工程中的所有目标文件，生成可执行文件

5　用 Run 菜单下的有关命令执行程序

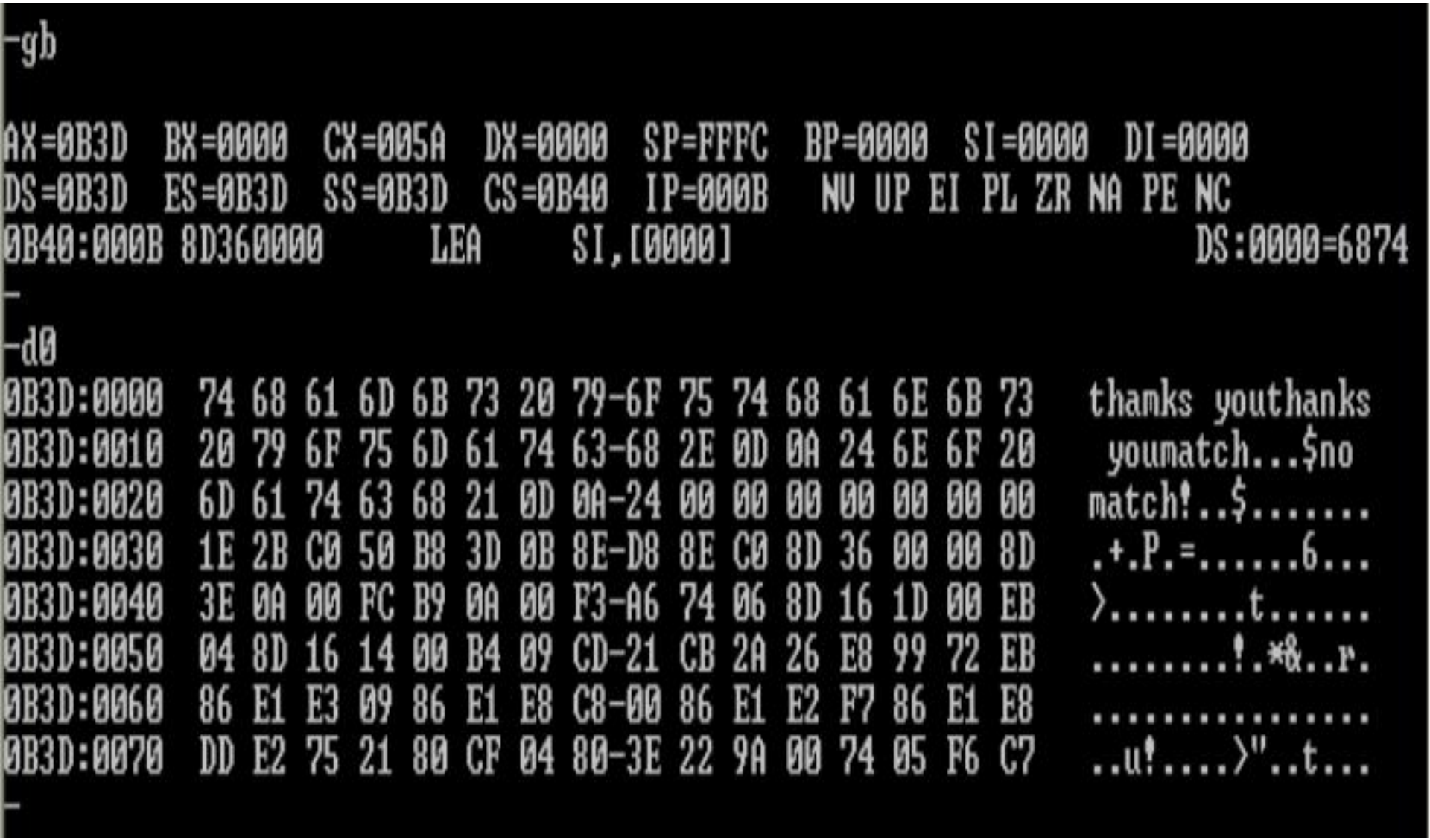
6　用 CodeView 调试程序


PWB
File Edit Search Run Data Options Calls Windows Help
[3] source1 CS:IP SAMPLE.ASM
32: repe cmpsb
17F5:0013 F3A6 REP CMPSB
33: jnz match
17F5:0015 7506 JNZ 001D
34: lea dx,mess2
17F5:0017 8D164500 LEA DX,WORD PTR [0045]
35: jmp short disp
17F5:001B EB04 JMP 0021
36: match:
37: lea dx,mess1
17F5:001D 8D163C00 LEA DX,WORD PTR [003C]
38: disp:
39: mov ah,09
17F5:0021 B409 MOV AH,09
40: int 21h
17F5:0023 CD21 INT 21
41:
[9] command
>
BP# 1 - Break at: "(,SAMPLE.ASM,SAMPLE.exe) .33"
>
=[7]reg
AX = 17F7
BX = 0000
CX = 0000
DX = 0000
SP = 0000
BP = 0000
SI = 0023
DI = 003C
DS = 17F7
ES = 17F7
SS = 17F5
CS = 17F5
IP = 0017
FL = 3246
NV UP EI PL
ZR NA PE NC
ds:0045
6f4e
<F8=Trace> <F10=Step> <F5=Go> <F3=S1 Fmt> <Sh+F3=M1 Fmt>
HEX

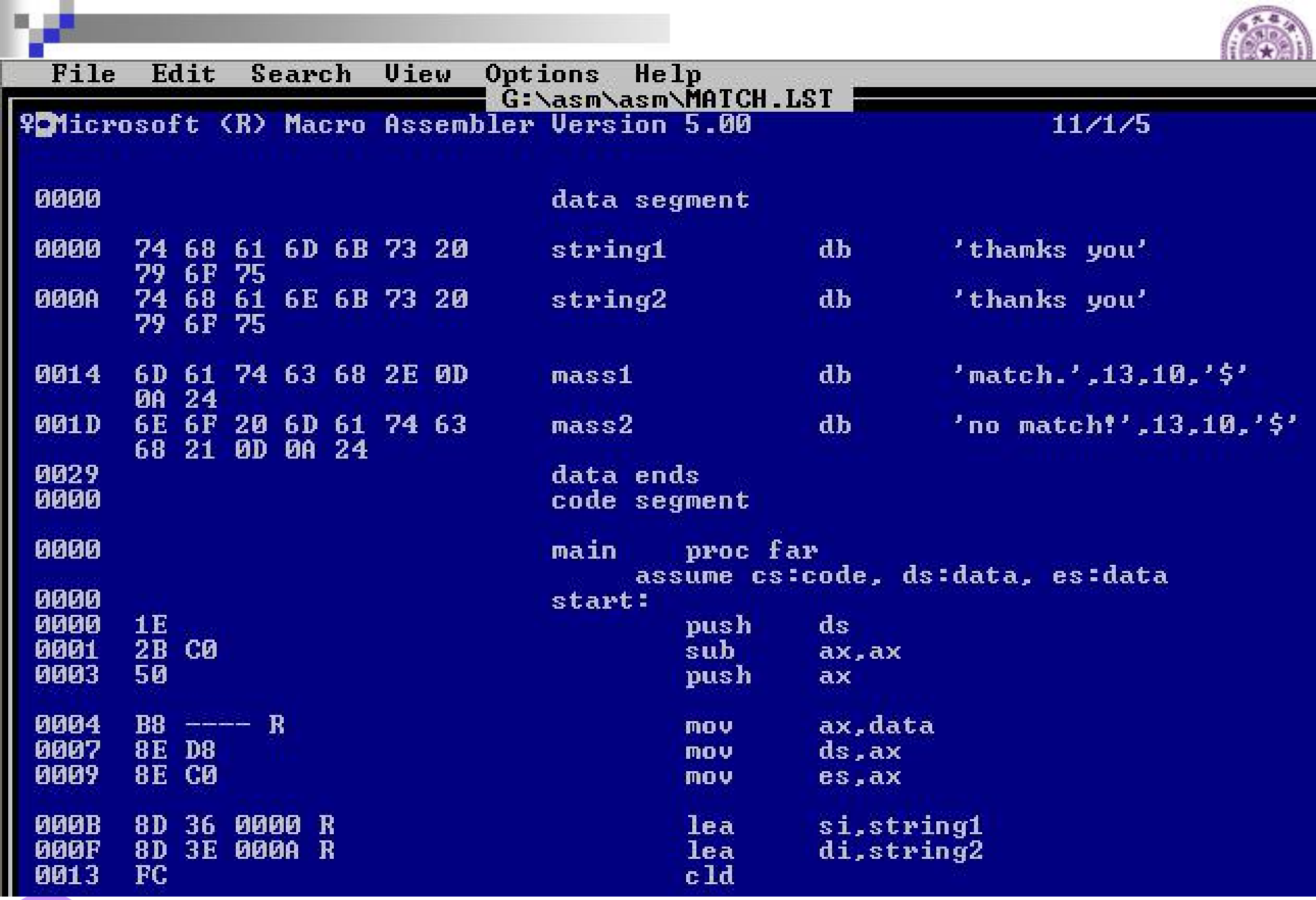
match.asm

```
data    segment
        string1   db   'thamks you'
        string2   db   'thanks you'
        mass1     db   'match.', 13, 10, '$'
        mass2     db   'no match!', 13, 10, '$'
data    ends

code    segment
main    proc    far
        assume  cs:code, ds:data, es:data
start:
        push    ds
        sub     ax, ax
        push    ax

        mov     ax, data
        mov     ds, ax
        mov     es, ax

        lea     si, string1
        lea     di, string2
        cld
        mov     cx, 10
        repz    cmpsb
        jz      match
        lea     dx, mass2
        jmp     short  disp
match:  lea     dx, mass1
disp:   mov     ah, 09
        int     21h
        ret
main    endp
code    ends
        end     start
```

```
G:\asm\asm>debug match.exe
-
-u
0B40:0000 1E              PUSH    DS
0B40:0001 2BC0            SUB     AX,AX
0B40:0003 50              PUSH    AX
0B40:0004 B83D0B          MOV     AX,0B3D
0B40:0007 8ED8            MOV     DS,AX
0B40:0009 8EC0            MOV     ES,AX
0B40:000B 8D360000        LEA     SI,[0000]
0B40:000F 8D3E0A00        LEA     DI,[000A]
0B40:0013 FC              CLD
0B40:0014 B90A00          MOV     CX,000A
0B40:0017 F3              REPZ
0B40:0018 A6              CMPSB
0B40:0019 7406            JZ      0021
0B40:001B 8D161D00        LEA     DX,[001D]
0B40:001F EB04            JMP     0025
-
-d0
0B2D:0000  CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 4E 05 8A 03    . .........O.N...
0B2D:0010  4E 05 17 03 4E 05 3D 05-01 01 01 00 02 FF FF FF   N...N.=.........
0B2D:0020  FF FF FF FF FF FF FF FF-FF FF FF FF FB 0A 4C 01   ..............L.
0B2D:0030  0E 0A 14 00 18 00 2D 0B-FF FF FF FF 00 00 00 00   ......-.........
0B2D:0040  05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B2D:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20   .!...........
0B2D:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20            .....
0B2D:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00            ........
-
-r
AX=0000  BX=0000  CX=005A  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0B2D  ES=0B2D  SS=0B3D  CS=0B40  IP=0000   NV UP EI PL NZ NA PO NC
0B40:0000 1E              PUSH    DS
-
```


-gb
AX=0B3D BX=0000 CX=005A DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000
DS=0B3D ES=0B3D SS=0B3D CS=0B40 IP=000B NV UP EI PL ZR NA PE NC
0B40:000B 8D360000 LEA SI,[0000] DS:0000=6874
-
-d0
0B3D:0000 74 68 61 6D 6B 73 20 79-6F 75 74 68 61 6E 6B 73 thamks youthanks
0B3D:0010 20 79 6F 75 6D 61 74 63-68 2E 0D 0A 24 6E 6F 20 youmatch...$no
0B3D:0020 6D 61 74 63 68 21 0D 0A-24 00 00 00 00 00 00 00 match!..$.....
0B3D:0030 1E 2B C0 50 B8 3D 0B 8E-D8 8E C0 8D 36 00 00 8D .+.P.=.....6...
0B3D:0040 3E 0A 00 FC B9 0A 00 F3-A6 74 06 8D 16 1D 00 EB >.....t.....
0B3D:0050 04 8D 16 14 00 B4 09 CD-21 CB 2A 26 E8 99 72 EB ....!.*&..r.
0B3D:0060 86 E1 E3 09 86 E1 E8 C8-00 86 E1 E2 F7 86 E1 E8 ....
0B3D:0070 DD E2 75 21 80 CF 04 80-3E 22 9A 00 74 05 F6 C7 ..u!....>"..t...
-

```
-u
0B40:000B 8D360000      LEA     SI,[0000]
0B40:000F 8D3E0A00      LEA     DI,[000A]
0B40:0013 FC            CLD
0B40:0014 B90A00        MOV     CX,000A
0B40:0017 F3            REPZ
0B40:0018 A6            CMPSB
0B40:0019 7406          JZ      0021
0B40:001B 8D161D00      LEA     DX,[001D]
0B40:001F EB04          JMP     0025
0B40:0021 8D161400      LEA     DX,[0014]
0B40:0025 B409          MOV     AH,09
0B40:0027 CD21          INT     21
0B40:0029 CB            RETF
0B40:002A 2A26E899      SUB     AH,[99E8]
-
-g29
no match!

AX=0924  BX=0000  CX=0006  DX=001D  SP=FFFC  BP=0000  SI=0004  DI=000E
DS=0B3D  ES=0B3D  SS=0B3D  CS=0B40  IP=0029   NV UP EI NG NZ AC PE CY
0B40:0029 CB            RETF
-
```


File Edit Search View Options Help
G:\asm\asm\MATCH.LST
Microsoft (R) Macro Assembler Version 5.00 11/1/5
0000 data segment
0000 74 68 61 6D 6B 73 20 string1 db 'thamks you'
79 6F 75
000A 74 68 61 6E 6B 73 20 string2 db 'thanks you'
79 6F 75
0014 6D 61 74 63 68 2E 0D mass1 db 'match.',13,10,'$'
0A 24
001D 6E 6F 20 6D 61 74 63 mass2 db 'no match!',13,10,'$'
68 21 0D 0A 24
0029 data ends
0000 code segment
0000 main proc far
assume cs:code, ds:data, es: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 8E C0 mov es,ax
000B 8D 36 0000 R lea si,string1
000F 8D 3E 000A R lea di,string2
0013 FC cld

match.lst (1)

```
0014  B9 000A                       mov     cx,10
0017  F3/ A6                        repz    cmpsb
0019  74 06                        jz      match
001B  8D 16 001D R                  lea     dx,mass2
001F  EB 04                         jmp     short disp
0021                          match:
0021  8D 16 0014 R                  lea     dx,mass1
0025                          disp:
0025  B4 09                         mov     ah,09
0027  CD 21                         int     21h

0029  CB                            ret
002A                          main endp

002A                          code ends

                              end start
```

match.lst (2)

```
Microsoft (R) Macro Assembler Version 5.00                  11/1/5

Segments and Groups:

                N a m e                 Length   Align   Combine Class

CODE . . . . . . . . . . . . . .        002A     PARA    NONE
DATA . . . . . . . . . . . . . .        0029     PARA    NONE

Symbols:

                N a m e                 Type     Value   Attr

DISP . . . . . . . . . . . . . .        L NEAR   0025    CODE

MAIN . . . . . . . . . . . . . .        F PROC   0000    CODE    Length = 002A
MASS1  . . . . . . . . . . . . .        L BYTE   0014    DATA
MASS2  . . . . . . . . . . . . .        L BYTE   001D    DATA
MATCH  . . . . . . . . . . . . .        L NEAR   0021    CODE

START  . . . . . . . . . . . . .        L NEAR   0000    CODE
STRING1  . . . . . . . . . . . .        L BYTE   0000    DATA
STRING2  . . . . . . . . . . . .        L BYTE   000A    DATA

@FILENAME  . . . . . . . . . . .        TEXT  match


     42 Source  Lines
     42 Total   Lines
     12 Symbols

  50436 + 451052 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

match1.asm

```
data    segment
        string1   db   'thamks you'
        mass1     db   'match.', 13, 10, '$'
        mass2     db   'no match!', 13, 10, '$'
data    ends

extra   segment
        string2   db   'thanks you'
extra   ends

code    segment
main    proc      far
        assume    cs:code, ds:data, es:extra
start:
        ……
        mov       ax, data
        mov       ds, ax

        mov       ax, extra
        mov       es, ax
        ……
code    ends
        end       start
```

```
G:\asm\asm>debug match1.exe
-
-u
0B40:0000 1E              PUSH    DS
0B40:0001 2BC0            SUB     AX,AX
0B40:0003 50              PUSH    AX
0B40:0004 B83D0B          MOV     AX,0B3D
0B40:0007 8ED8            MOV     DS,AX
0B40:0009 B83F0B          MOV     AX,0B3F
0B40:000C 8EC0            MOV     ES,AX
0B40:000E 8D360000        LEA     SI,[0000]
0B40:0012 8D3E0000        LEA     DI,[0000]
0B40:0016 FC              CLD
0B40:0017 B90A00          MOV     CX,000A
0B40:001A F3              REPZ
0B40:001B A6              CMPSB
0B40:001C 7406            JZ      0024
0B40:001E 8D161300        LEA     DX,[0013]
-
-r
AX=0000  BX=0000  CX=005D  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0B2D  ES=0B2D  SS=0B3D  CS=0B40  IP=0000   NV UP EI PL NZ NA PO NC
0B40:0000 1E              PUSH    DS
-
-ge

AX=0B3F  BX=0000  CX=005D  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=0B3D  ES=0B3F  SS=0B3D  CS=0B40  IP=000E   NV UP EI PL ZR NA PE NC
0B40:000E 8D360000        LEA     SI,[0000]                      DS:0000=6874
-
-d0
0B3D:0000  74 68 61 6D 6B 73 20 79-6F 75 6D 61 74 63 68 2E   thamks youmatch.
0B3D:0010  0D 0A 24 6E 6F 20 6D 61-74 63 68 21 0D 0A 24 00   ..$no match!..$.
0B3D:0020  74 68 61 6E 6B 73 20 79-6F 75 00 00 00 00 00 00   thanks you......
0B3D:0030  1E 2B C0 50 B8 3D 0B 8E-D8 B8 3F 0B 8E C0 8D 36   .+.P.=....?....6
0B3D:0040  00 00 8D 3E 00 00 FC B9-0A 00 F3 A6 74 06 8D 16   ...>........t...
0B3D:0050  13 00 EB 04 8D 16 0A 00-B4 09 CD 21 CB D1 E3 8B   ...........!....
0B3D:0060  87 BE 22 0B 87 C0 22 74-E1 8B 9E FE FE D1 E3 D1   ..".."t.........
0B3D:0070  E3 8B 87 BE 22 8B 97 C0-22 89 86 FA FE 89 96 FC   ....".."........
-
```

match2.asm

```
.model small

.data
        string1   db   'thamks you‘
        string2   db   'thanks you'
        mass1     db   'match.', 13, 10, '$'
        mass2     db   'no match!', 13, 10, '$'

.code
main    proc      far
start:
        mov       ax, @data
        mov       ds, ax
        mov       es, ax
        ……
main    endp
        end       start
```

```
-d
0B3D:0000  1E 2B C0 50 B8 3F 0B 8E-D8 8E C0 8D 36 0A 00 8D   .+.P.?......6...
0B3D:0010  3E 14 00 FC B9 0A 00 F3-A6 74 06 8D 16 27 00 EB   >........t...'..
0B3D:0020  04 8D 16 1E 00 B4 09 CD-21 CB 74 68 61 6D 6B 73   ........!.thamks
0B3D:0030  20 79 6F 75 74 68 61 6E-6B 73 20 79 6F 75 6D 61    youthanks youma
0B3D:0040  74 63 68 2E 0D 0A 24 6E-6F 20 6D 61 74 63 68 21   tch...$no match!
0B3D:0050  0D 0A 24 FE FE 73 7D 8B-9E FE FE D1 E3 D1 E3 8B   ..$..s}.........
0B3D:0060  87 BE 22 0B 87 C0 22 74-E1 8B 9E FE FE D1 E3 D1   ..".."t.........
0B3D:0070  E3 8B 87 BE 22 8B 97 C0-22 89 86 FA FE 89 96 FC   ....".."........
-
-gb

AX=0B3F  BX=0000  CX=0053  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000
DS=0B3F  ES=0B3F  SS=0B3D  CS=0B3D  IP=000B   NV UP EI PL ZR NA PE NC
0B3D:000B 8D360A00      LEA     SI,[000A]                     DS:000A=6874
-d0
0B3F:0000  04 8D 16 1E 00 B4 09 CD-21 CB 74 68 61 6D 6B 73   ........!.thamks
0B3F:0010  20 79 6F 75 74 68 61 6E-6B 73 20 79 6F 75 6D 61    youthanks youma
0B3F:0020  74 63 68 2E 0D 0A 24 6E-6F 20 6D 61 74 63 68 21   tch...$no match!
0B3F:0030  0D 0A 24 FE FE 73 7D 8B-9E FE FE D1 E3 D1 E3 8B   ..$..s}.........
0B3F:0040  87 BE 22 0B 87 C0 22 74-E1 8B 9E FE FE D1 E3 D1   ..".."t.........
0B3F:0050  E3 8B 87 BE 22 8B 97 C0-22 89 86 FA FE 89 96 FC   ....".."........
0B3F:0060  FE C4 9E FA FE 26 8A 47-0C 2A E4 40 50 8B C3 05   .....&.G.*.@P...
0B3F:0070  0C 00 52 50 E8 19 46 83-C4 04 50 8D 86 00 FF 50   ..RP..F...P....P
-
```

例：

```
        .model  tiny
        .code
        org     100h
begin:  jmp     main
;*************************
num1    dw      1199H
num2    dw      1166H
sum     dw      ?
;*************************
main    proc    near
        mov     ax,  num1
        add     ax,  num2
        mov     sum, ax
        mov     ax,  4c00h
        int     21h
main    endp
        end     begin
```

例：

```
code    segment   'code'
        assume    cs:code,ds:code,ss:code,es:code
            org   100H
begin:      jmp   main

num1        dw    1199H
num2        dw    1166H
sum         dw    ?

main        proc  near
            mov   ax,  num1
            add   ax,  num2
            mov   sum, ax
            mov   ax,  4c00h
            int   21h
main        endp
code        ends
            end   begin
```

DOS 功能调用 INT 21H

用户在程序中调用 DOS 提供的一些子功能：

（1）一般设备的输入输出
（2）磁盘的输入输出及磁盘文件的管理
（3）其它

调用方法：

（1）设置调用参数
（2）MOV AH，功能号
（3）INT 21H

（1）DOS 键盘功能调用

例：单字符输入（AH = 1）

```
get-key :  mov   ah ,  1
           int   21h
           cmp   al ,  'Y'
           je    yes
           cmp   al ,  'N'
           je    no
           jne   get_key
yes:
           ……
no:
           ……
```

例：输入字符串 (AH = 0ah)

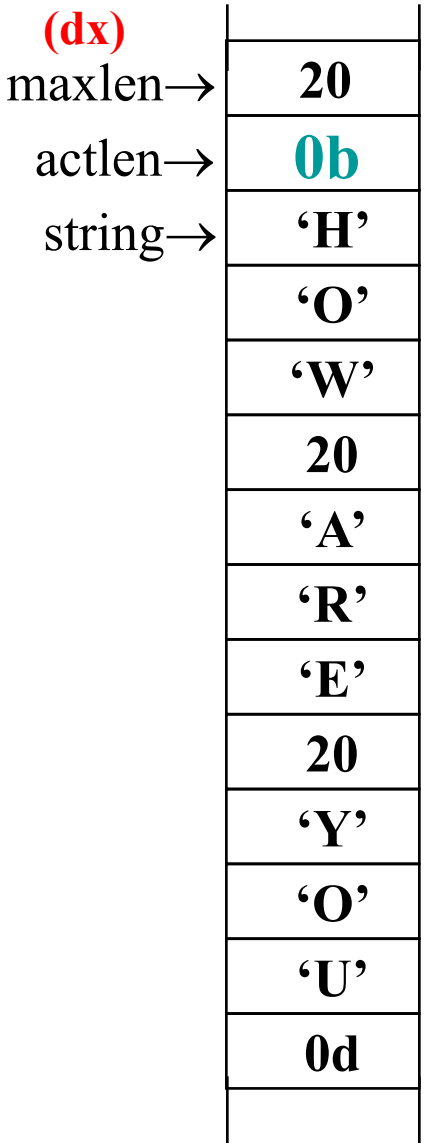
定义缓冲区：

（1） maxlen db 32
actlen db ?
string db 32 dup (?)

（2） maxlen db 32, 0, 32 dup (?)

（3） maxlen db 32, 33 dup (?)

```
lea   dx , maxlen
mov   ah , 0ah
int   21h
```



（2） DOS 显示功能调用

例：显示单个字符 (AH = 2)

```
mov    ah ,  2
mov    dl ,  ‘A’
int    21h
```

例：显示字符串 (AH = 9)

```
string   db   ‘HELLO’, 0dh, 0ah, ‘$’
……
mov    dx , offset string
mov    ah , 9
int    21h
```

（3） DOS 打印功能 (AH = 5)