目标代码动态调试—TurboDebugger

TD. EXE（简称TD）是一个具有窗口界面的程序调试器, 是Borland 公司产品Turob Debugger 的IA-16版本。利用TD，用户能够调试已有的可执行程序（后缀为EXE）；用户也可以在TD中直接输入程序指令，编写简单的程序。

- 一、如何启动TD
- 二、TD中的数制
- 三、TD的用户界面
- 四、代码区的操作
- 五、寄存器区和标志区的操作
- 六、数据区的操作
- 七、堆栈区的操作

处理器认识
环境检查修改
指令认识
直接I/O操作(硬件调试)
单步调试
断点调试

优点:彩色文本, 占用资源不多
缺点:不支持中断调试

二、如何在Windows环境下启动TD

在Windows中启动TD

- 仅启动TD而不载入要调试的程序

双击TD.EXE文件名，Windows就会打开一个DOS窗口并启动TD。启动TD后会显示一个版权对话框，这时按回车键即可关掉该对话框。

- 启动TD并同时载入要调试的程序

把要调试的可执行文件拖到TD.EXE文件名上，Windows就会打开一个DOS窗口并启动TD，然后TD会把该可执行文件自动载入内存供用户调试。

若建立可执行文件时未生成符号名表，**TD**启动后会显示“**Program has no symbol table**”的提示窗口，这时按回车键即可关掉该窗口。(可修改属性)

一、如何在命令行方式启动TD

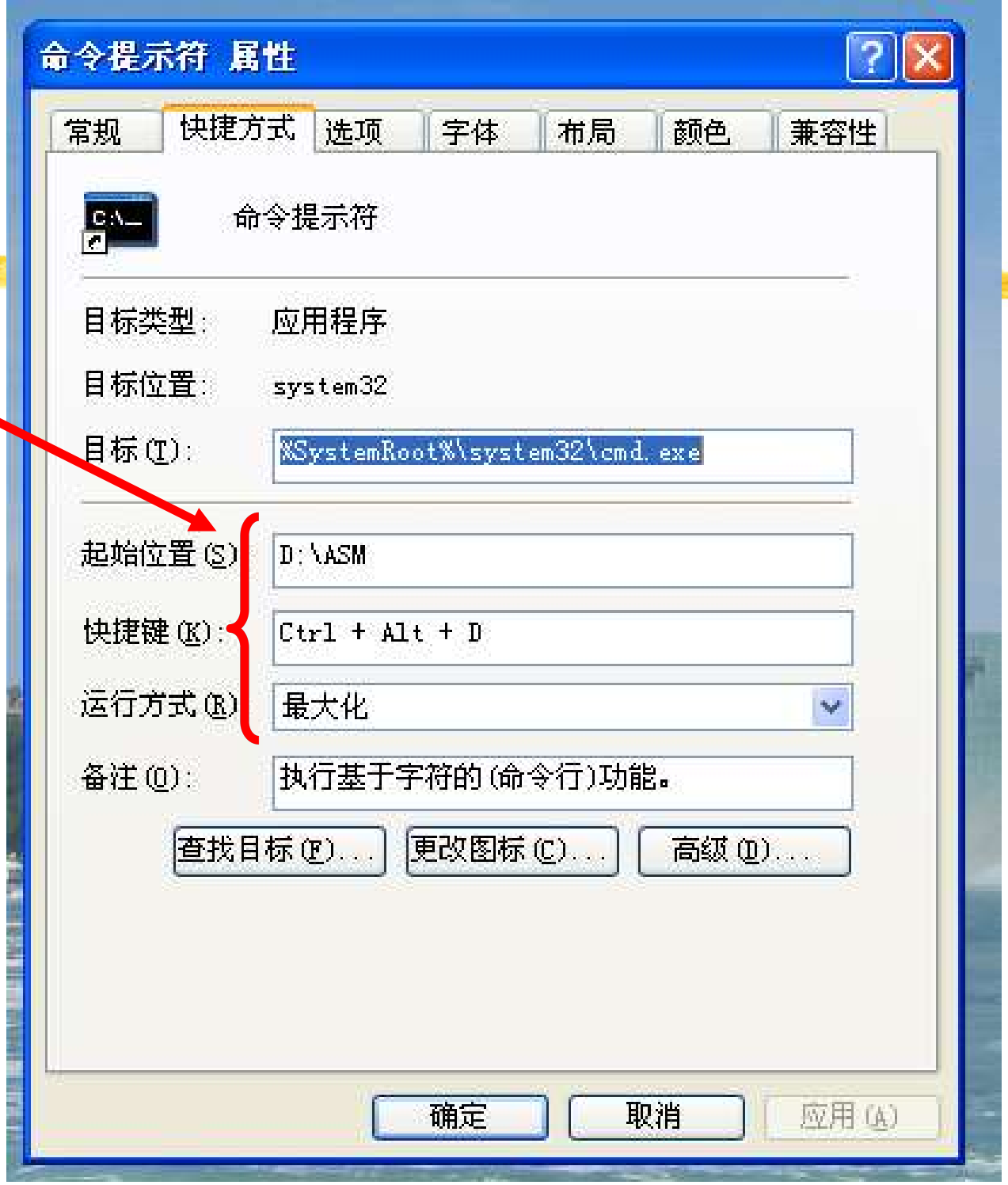
什么是命令行方式?

(1)Windows下进入命令行方式/常用MS-DOS命令

Windows下利用开始—运行(或按Win键+R）—输入cmd 或command (Win2K) ↙或在程序下进入MS-DOS方式(Win2K)或附件下(Win XP)进入命令提示符进入命令行方式。也可将相关命令复制到桌面。

可以根据爱好和需要修改MS-DOS方式或命令提示符的相关属性，包括起始（工作）目录（路径）、快捷键（默认CTRL + ALT +D键)、窗口/全屏选项、前景背景颜色等，

命令行快捷方式选择


命令提示符 属性
常规
快捷方式
选项
字体
布局
颜色
兼容性
命令提示符
目标类型: 应用程序
目标位置: system32
目标(T): %SystemRoot%\system32\cmd.exe
起始位置(S): D:\ASM
快捷键(K): Ctrl + Alt + D
运行方式(R): 最大化
备注(O): 执行基于字符的(命令行)功能。
查找目标(F)...
更改图标(C)...
高级(D)...
确定
取消
应用(A)

颜色属性选择

命令行样例模式

典型命令行操作

采用滚屏方式 **(ScreenUp)**

一、如何启动TD

在命令行方式-DOS窗口中启动TD

- 仅启动TD而不载入要调试的程序

D:\ASM＞TD↙

用这种方法启动TD，TD会显示一个版权对话框，这时按回车键即可关掉该对话框。

- 启动TD并同时载入要调试的程序

D:\ASM＞TD　HELLO.EXE↙

若建立可执行文件时未生成符号名表(带源代码调试)，TD启动后会显示“Program has no symbol table”的提示窗口，这时按回车键即可关掉该窗口。

二、TD中的数制

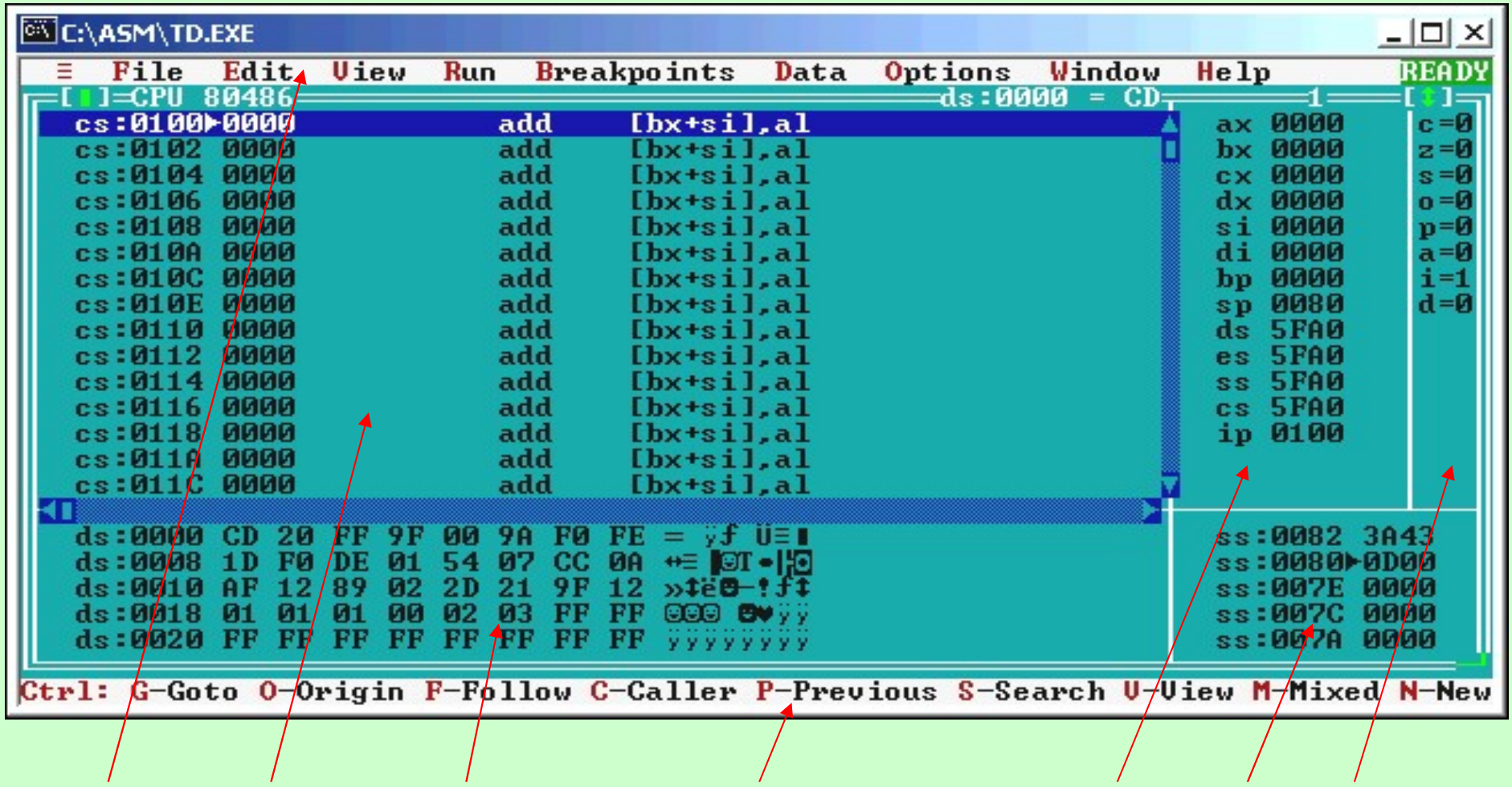
TD支持各种进位记数制，但通常情况下屏幕上显示的机器指令码、内存地址及内容、寄存器的内容等均按十六进制显示（数值后省略“H”）。在TD的很多操作中，需要用户输入一些数据、地址等，在输入时应遵循计算机中数的记数制标识规范。例如：

二进制数后面加“B”或“b”，如10010001b等；

十六进制数后面加“H”或“h”，如38h、0a5h、0ffh等。

TD允许在常数前面加上正负号,遵循补码格式。所有的实验在输入程序或数据时，若无特别说明，都可按十六进制数进行输入，若程序中需要输入负数，可按上述规则进行输入。

三、TD的用户界面—CPU窗口



全局菜单　代码区　数据区　功能键提示条　寄存器区 堆栈区 标志区

图3.1 TD的CPU窗口界面

三、TD的用户界面—CPU窗口

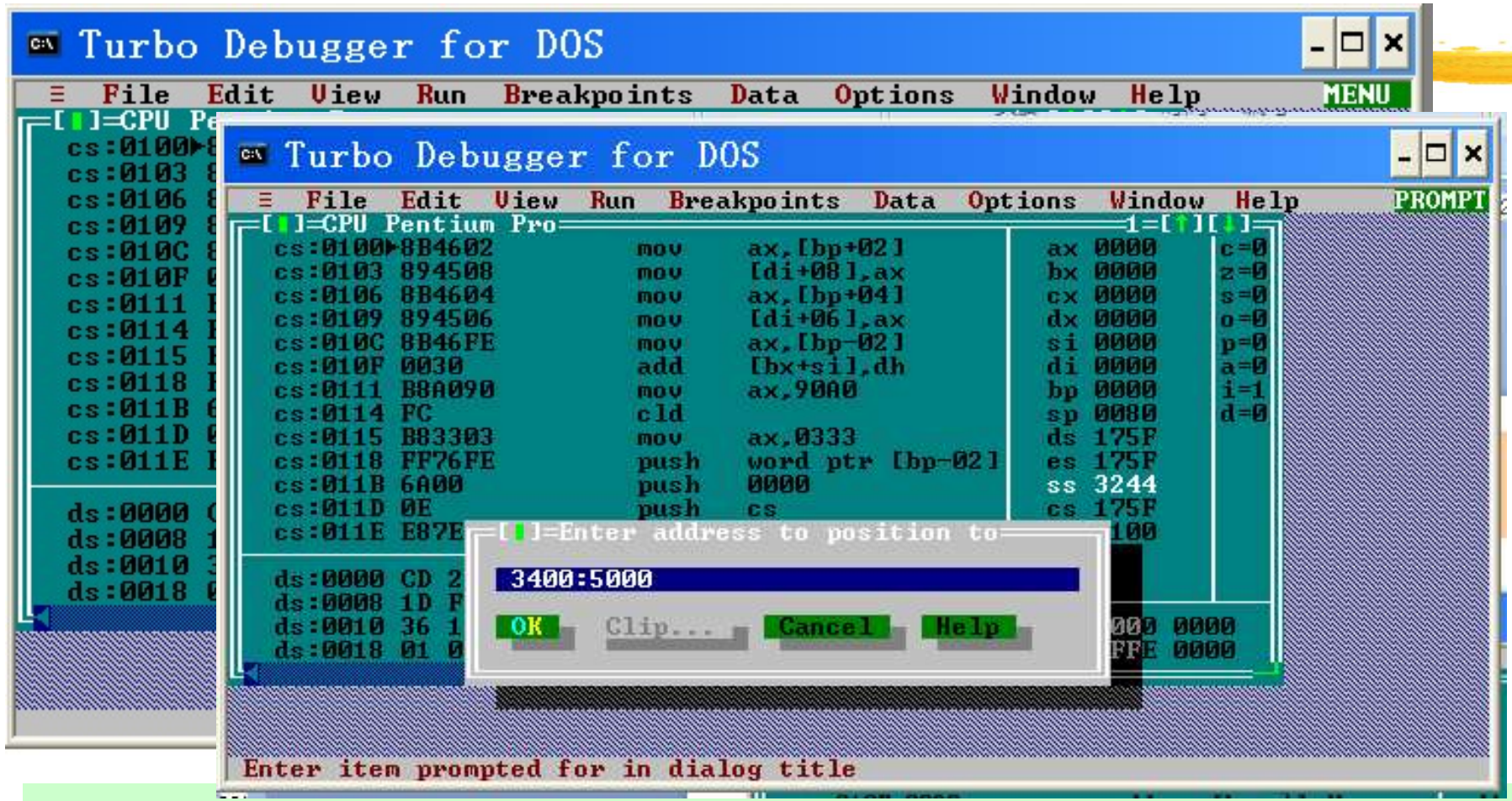
功能作用：--代码动态调试

- 显示CPU和内存整个状态；
- 在代码区内使用嵌入汇编，输入指令或对程序进行临时性修改。
- 存取数据区中任何数据结构下的字节，并以多种格式显示或改变它们。
- 检查和改变寄存器（包括标志寄存器）的内容。

五个区域：代码、寄存器、标志、数据和堆栈区。

光标所在区域称为当前区域，用户可以使用Tab键或Shift-Tab键切换当前区域，也可以在相应区中单击鼠标左键选中某区为当前区。

感兴趣的存储区域选择



鼠标右键点击感兴趣的区域（代码、数据和堆栈区之一），出现**GOTO…**下拉菜单（选择**Goto../G)**写入新的段基和偏移量地址（如**3400:5000,**或**ES：5000**）回车输入

三、TD的用户界面—全局菜单介绍1

CPU窗口的上面为TD的全局菜单条，可用“ALT键+菜单项首字符”打开菜单项对应的下拉子菜单。在子菜单中用“↑”、“↓”键选择所需的功能，按回车键即可执行选择的功能。为简化操作，某些常用的子菜单项后标出了对应的快捷键。

File菜单：文件操作

Open	载入可执行程序文件准备调试
Change dir	改变当前目录
Get info	显示被调试程序的信息
DOS shell	执行DOS命令解释器（用EXIT命令退回到TD）
Quit	退出TD（Alt-X）

三、TD的用户界面—全局菜单介绍2

Edit菜单：文本编辑

Copy 复制当前光标所在内存单元的内容到粘贴板（Shift-F3）

Paste把粘贴板内容粘贴到当前光标所在内存单元（Shift-F4）

View菜单：打开一个信息查看窗口

Breakpoints	断点信息
Stack	堆栈段内容
Watches	被监视对象信息
Variables	变量信息
Module	模块信息
File	文件内容
CPU	打开一个新的CPU窗口
Dump	数据段内容
Registers	寄存器内容

三、TD的用户界面—全局菜单介绍3

Run菜单：执行

Run　　从CS:IP开始运行程序直到程序结束(F9)

Go to cursor　从CS:IP开始运行程序到光标处(F4)

Trace into　　单步跟踪执行（对CALL指令将跟踪进入子程序）(F7)

Step over　　单步跟踪执行(对CALL指令将执行完子程序才停下)(F8)

Execute to　　执行到指定位置(Alt-F9)

Until return　执行当前子程序直到退出(Alt-F8) ）

Breakpoints菜单：断点功能

Toggle　　在当前光标处设置/清除断点(F2)

At　　在指定地址处设置断点（Alt-F2）

Delete all　清除所有断点

三、TD的用户界面—全局菜单介绍4

Data菜单：数据查看

Inspector　打开观察器以查看指定的变量或表达式

Evaluate/Modify　计算和显示表达式的值

Add watch　　增加一个新的表达式到观察器窗口

Option菜单：杂项

Display options　设置屏幕显示的外观

Path for source　指定源文件查找目录

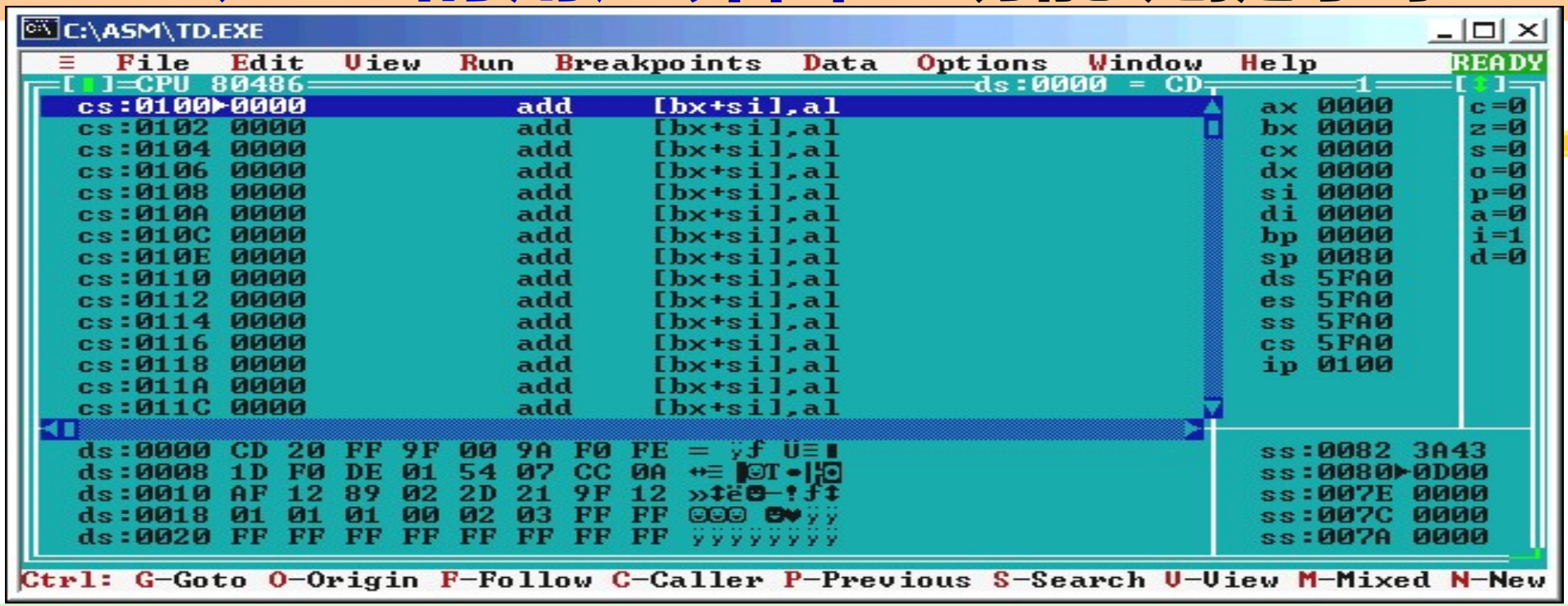
Save options　　保存当前选项

三、TD的用户界面—全局菜单介绍5

Window菜单：窗口操作

Zoom	放大/还原当前窗口（F5）
Next	转到下一窗口（F6）
Next Pane	转到当前窗口的下一区域（Tab）
Size/Move	改变窗口大小/移动窗口（Ctrl-F5）
Close	关闭当前窗口（Alt-F3）
User screen	查看用户程序的显示（Alt-F5）

三、TD的用户界面—功能键提示条



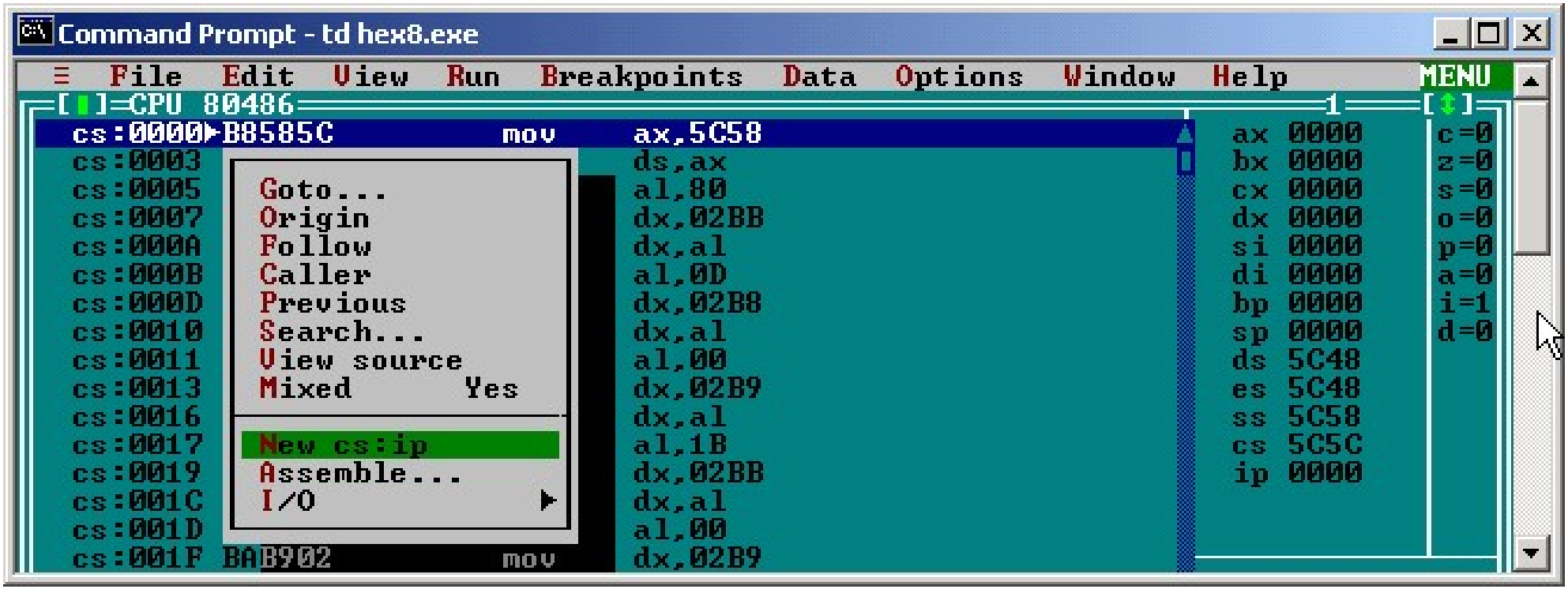
三组功能键：**F1～F10，Alt-F1～Alt-F10,Ctrl**功能键--代码区的局部菜单。**CPU**窗口下面的提示条中显示了这三组功能键对应的功能。通常情况下提示条中显示的是**F1～F10**功能键的功能。按住**Alt**不放，提示条中将显示**Alt-F1～Alt-F10**功能键的功能。按住**Ctrl**不放，提示条中将显示各**Ctrl**功能键的功能。

TD功能键对应功能表

键	功能	键	功能	键	功能
F1	帮助	Alt-F1	帮助	Ctrl-G	定位到指定地址
F2	设/清断点	Alt-F2	设置断点	Ctrl-O	定位到 CS:IP
F3	查看模块	Alt-F3	关闭窗口	Ctrl-F	定位到指令目的地址
F4	运行到光标	Alt-F4	Undo 跟踪	Ctrl-C	定位到调用者
F5	放大窗口	Alt-F5	用户屏幕	Ctrl-P	定位到前一个地址
F6	下一窗口	Alt-F6	Undo 关窗	Ctrl-S	查找指定的指令
F7	跟踪进入	Alt-F7	指令跟踪	Ctrl-V	查看源代码
F8	单步跟踪	Alt-F8	跟踪到返回	Ctrl-M	选择代码显示方式
F9	执行程序	Alt-F9	执行到某处	Ctrl-N	更新 CS:IP
F10	激活菜单	Alt-F10	局部菜单		

CPU窗口--局部菜单

TD的CPU窗口中，每个区域都有一个局部菜单，局部菜单提供了对本区域进行操作的各个命令。在当前区域中按Alt-F10键或单击鼠标右键即可激活本区域的局部菜单,进行修改等各种操作



代码区的局部菜单

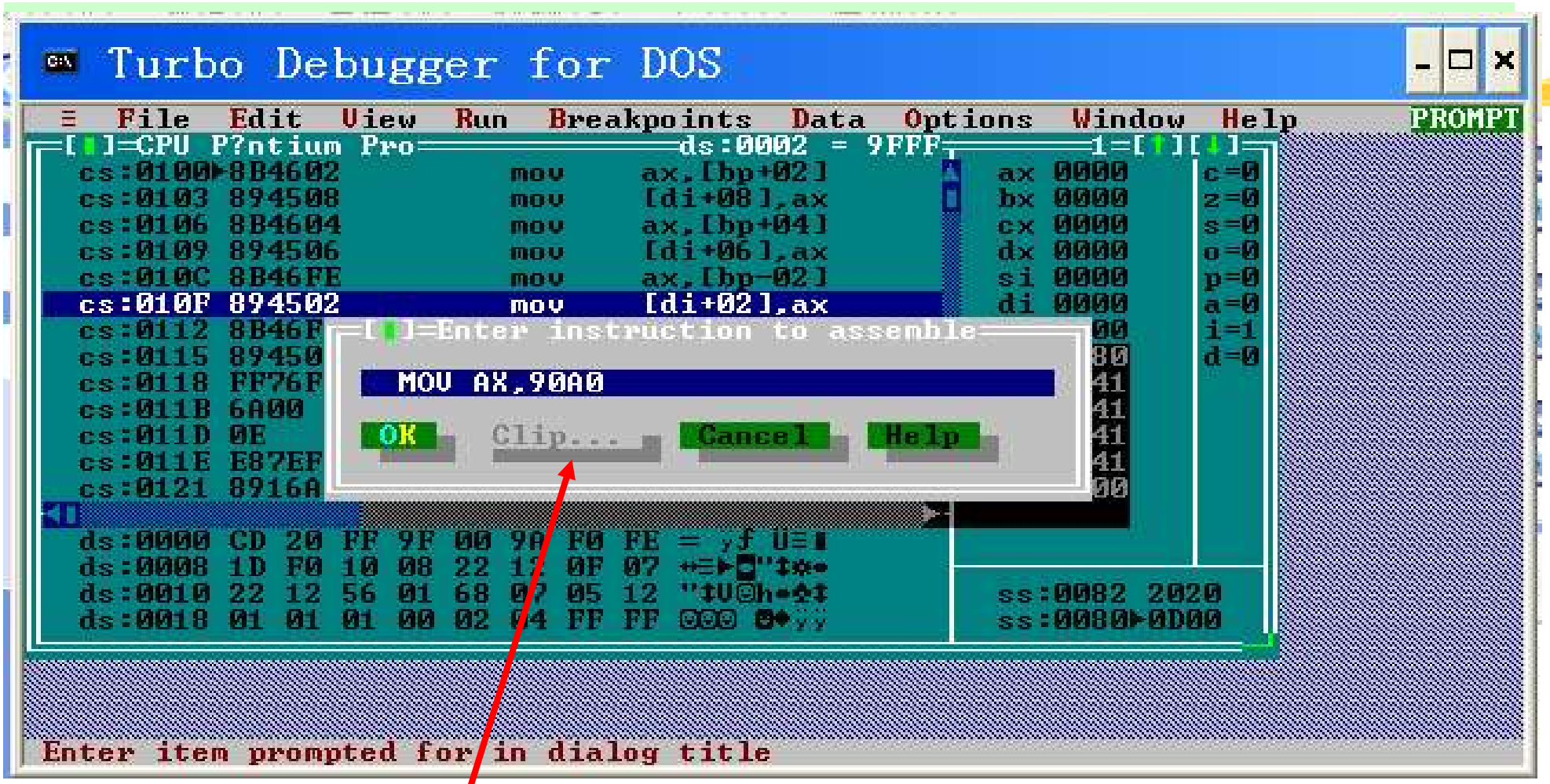
CPU窗口--代码区的操作

代码区用来显示代码（程序）的地址、代码的机器指令和代码的反汇编指令。本区中显示的反汇编指令依赖于所指定的程序起始地址。TD自动反汇编代码区的机器代码并显示对应的汇编指令。

每条反汇编指令的最左端是其地址，如果段地址与CS段寄存器的内容相同，则只显示字母“CS”和偏移量（CS:YYYY），否则显示完整的十六进制的段地址和偏移地址（XXXX:YYYY）。地址与反汇编指令之间显示的是指令的机器码。如果代码区当前光标所在指令引用了一个内存单元地址，　则该内存单元地址和内存单元的当前内容显示在CPU窗口顶部边框的右部，这样不仅可以看到指令操作码，还可看到指令要访问的内存单元的内容。

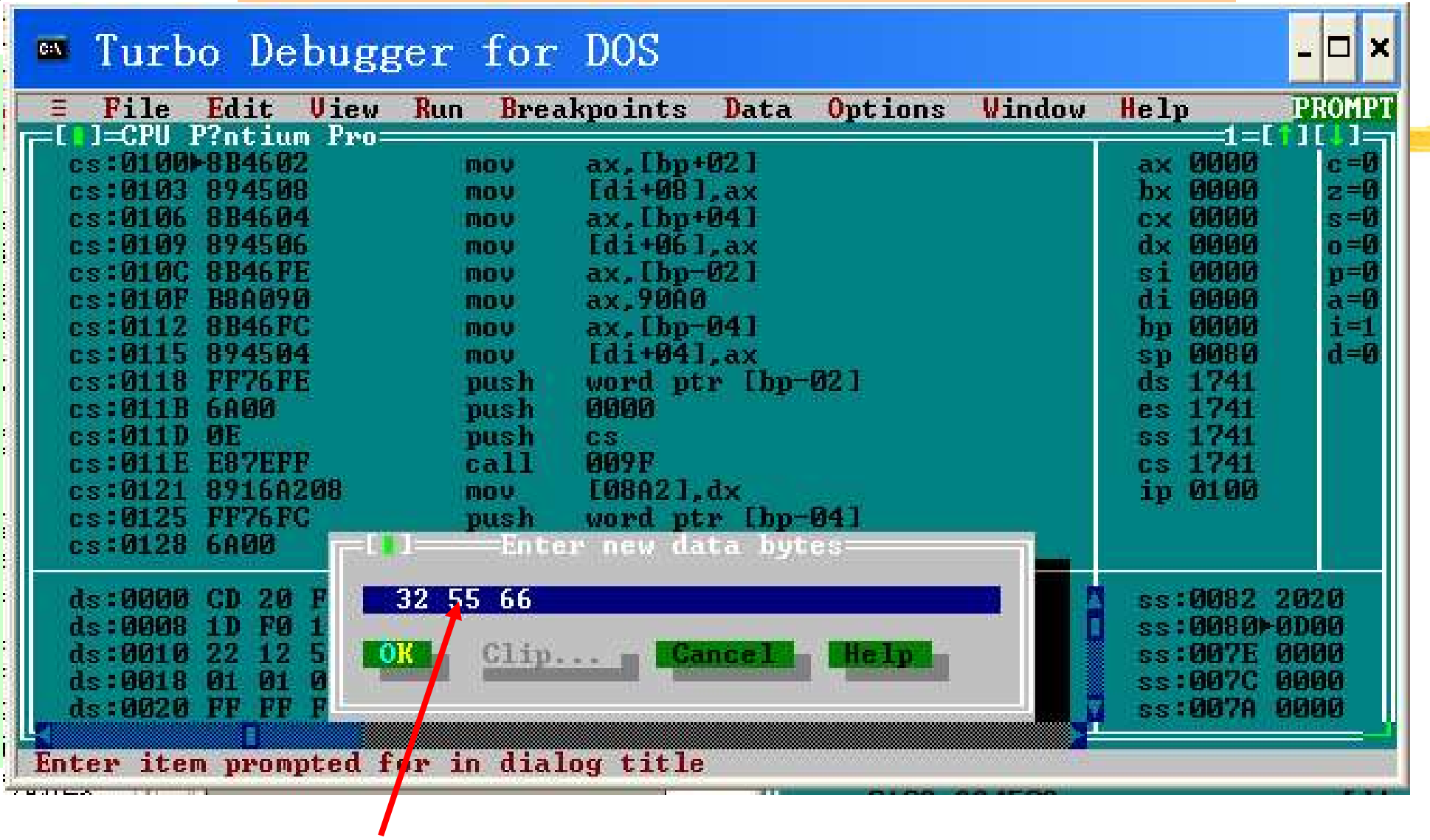
*当带源代码调试时可以看到指令符号

CPU窗口--代码区的操作



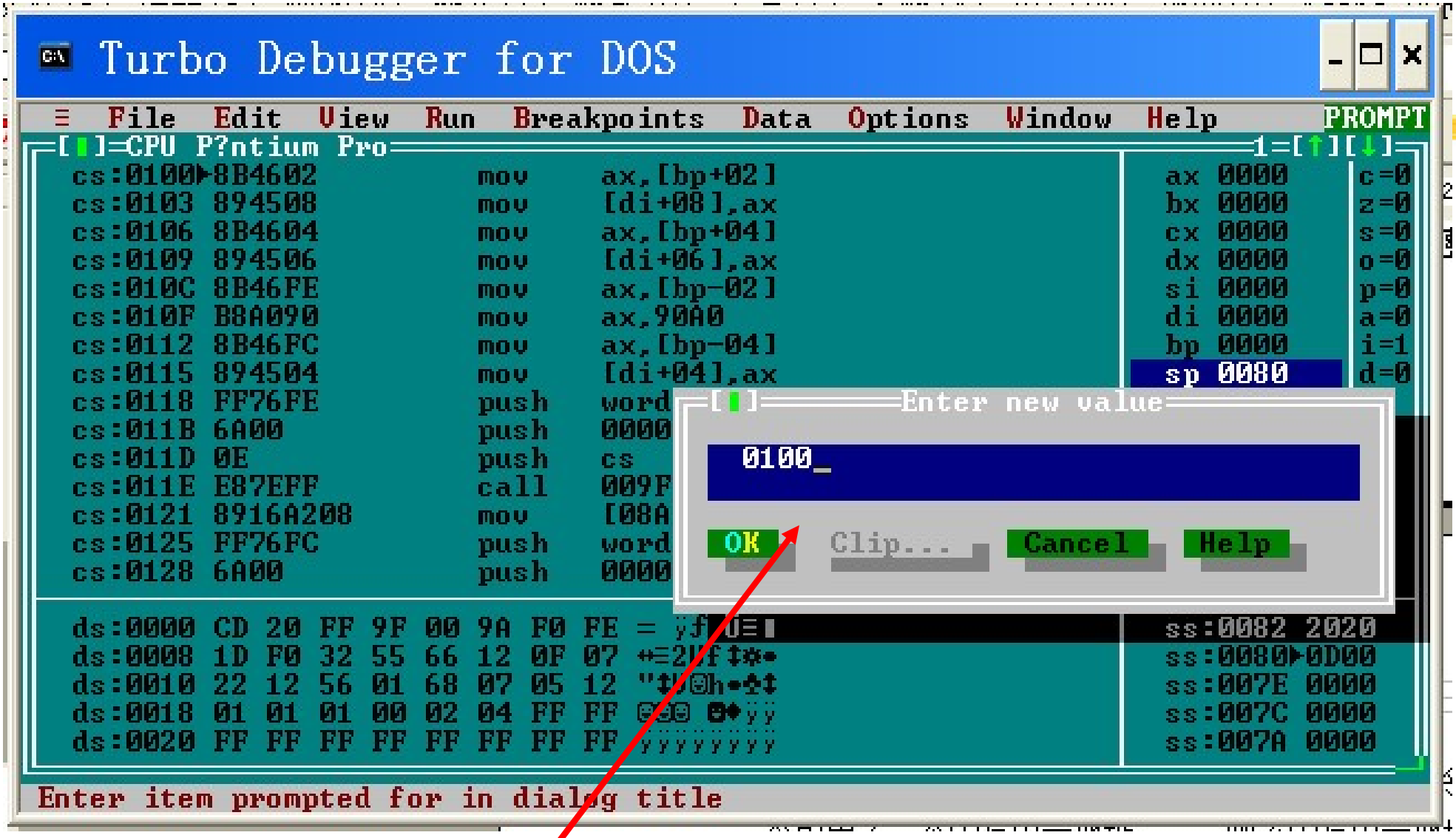
在光标行按空格键弹出输入框,汇编修改当前指令.数据区相同

CPU窗口—数据区的操作



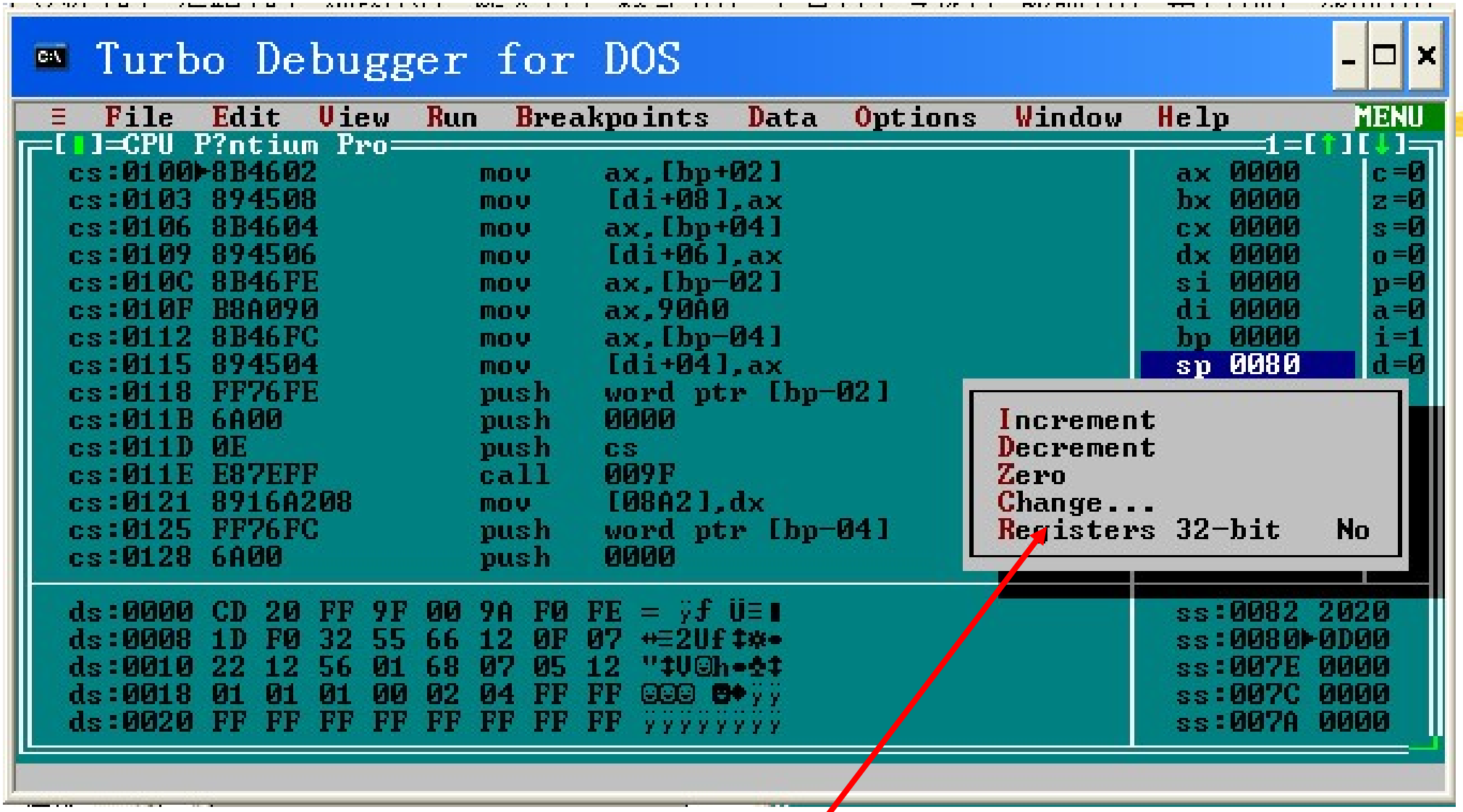
在光标处按空格键弹出输入框,修改内存数据(可连续输入,用空格隔开)

CPU窗口—寄存器区的操作1



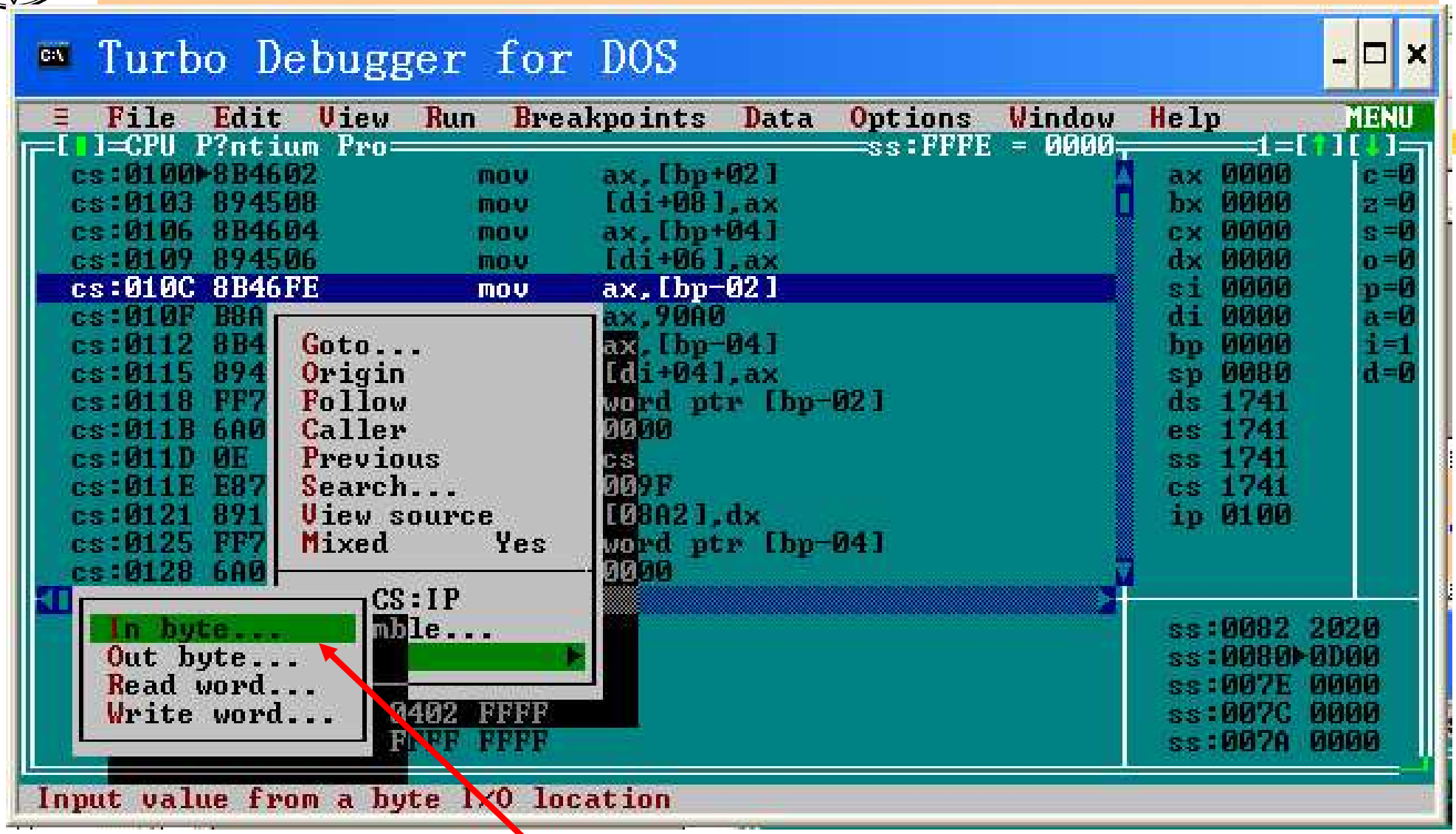
在光标处按空格键弹出输入框,修改内存数据(可连续输入,用空格隔开),堆栈区相同

CPU窗口—寄存器区的操作2



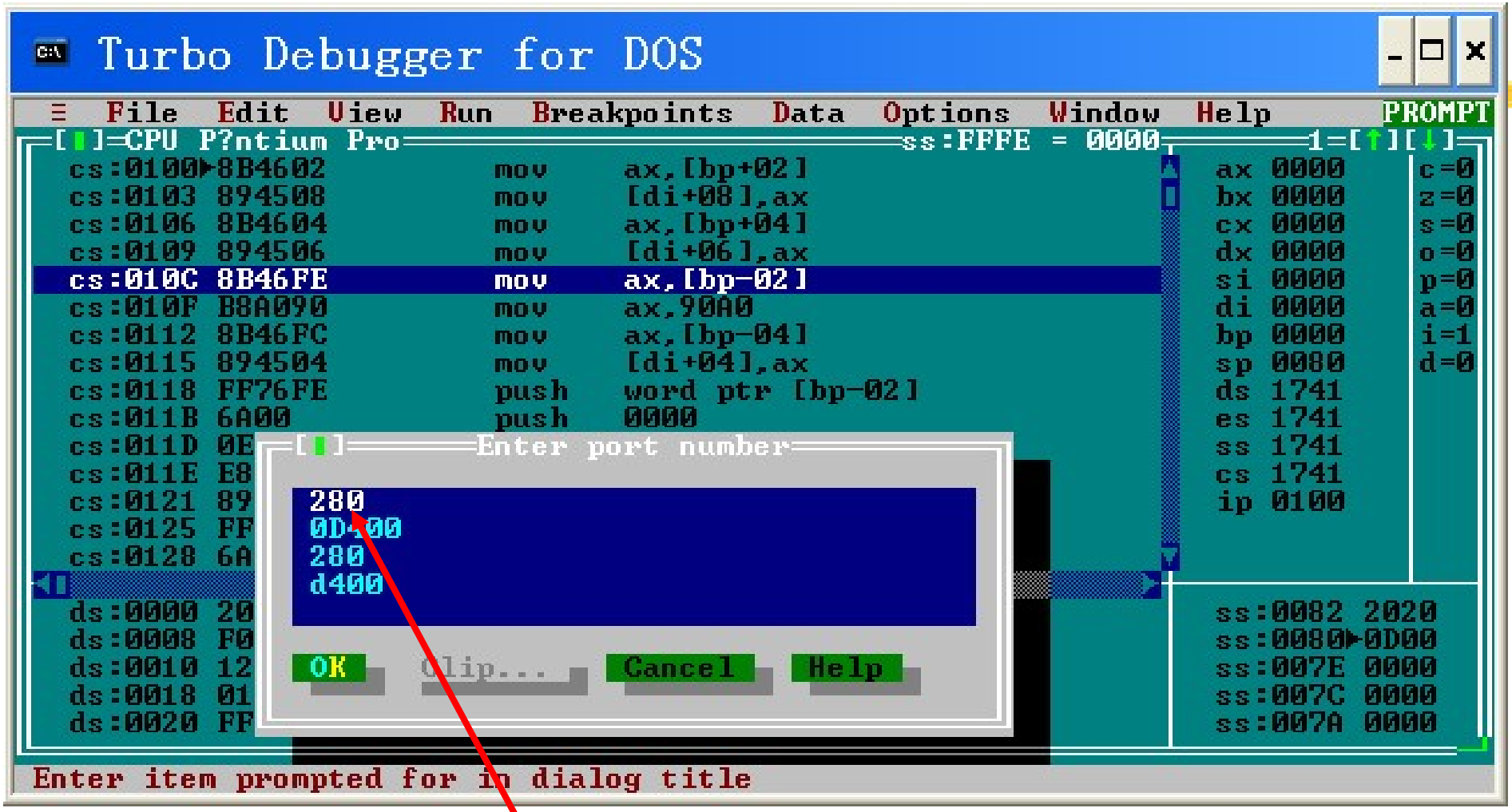
在光标处单击鼠标右键弹出局部菜单,选择控制功能,增减1或置零等

CPU窗口—代码区I/O操作:输入例



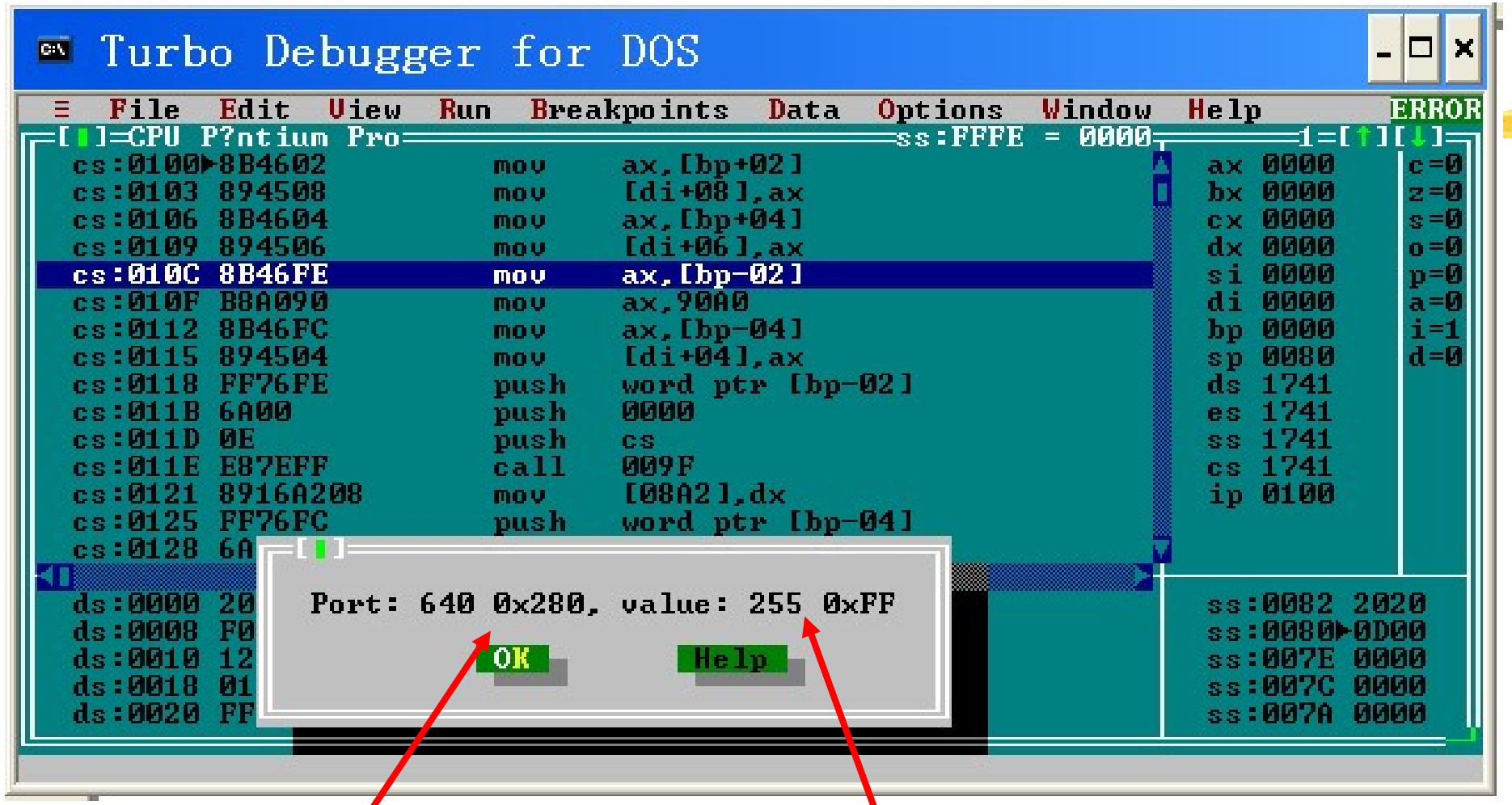
在光标处单击鼠标右键弹出局部菜单,选择**I/O --In Byte**

CPU窗口—代码区I/O操作:输入例



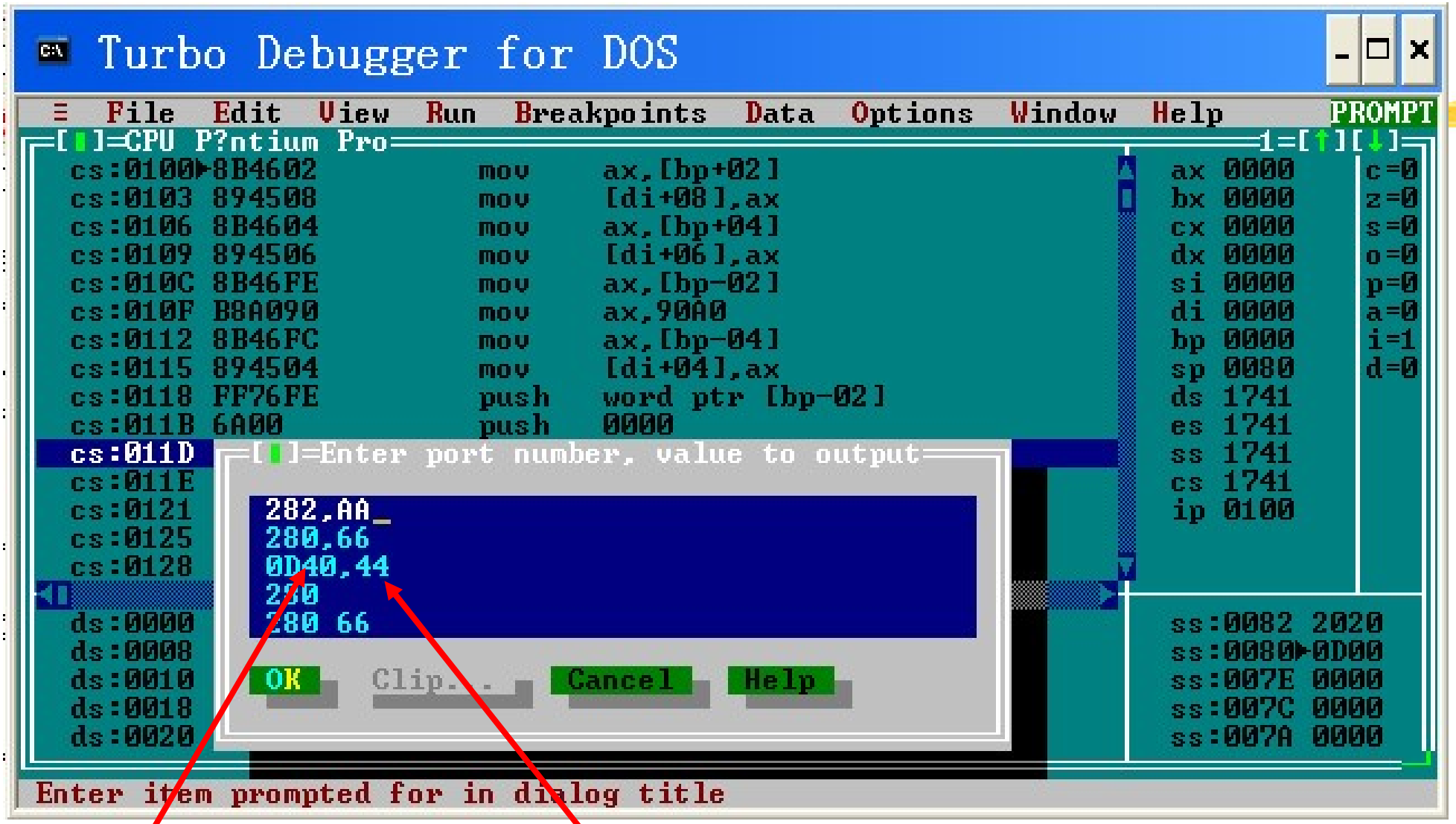
在光标处输入端口号或端口符号

CPU窗口—代码区I/O操作:输入例



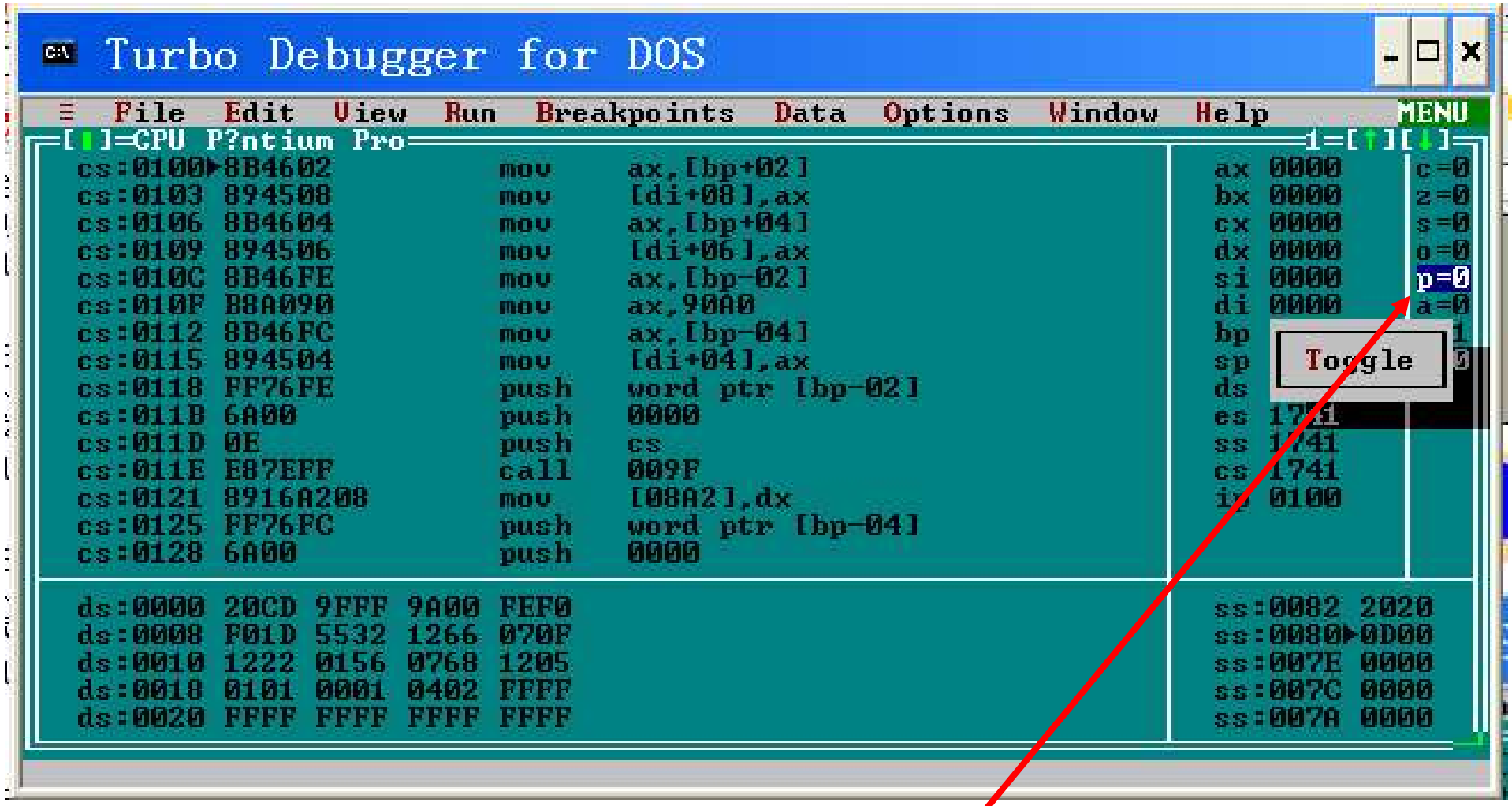
端口号十进制和十六进制　　输入数值十进制和十六进制数

CPU窗口—代码区I/O操作:输出例



端口号十进制和　　输出数值十六进制数,中间用逗号隔开

CPU窗口—标志区修改操作

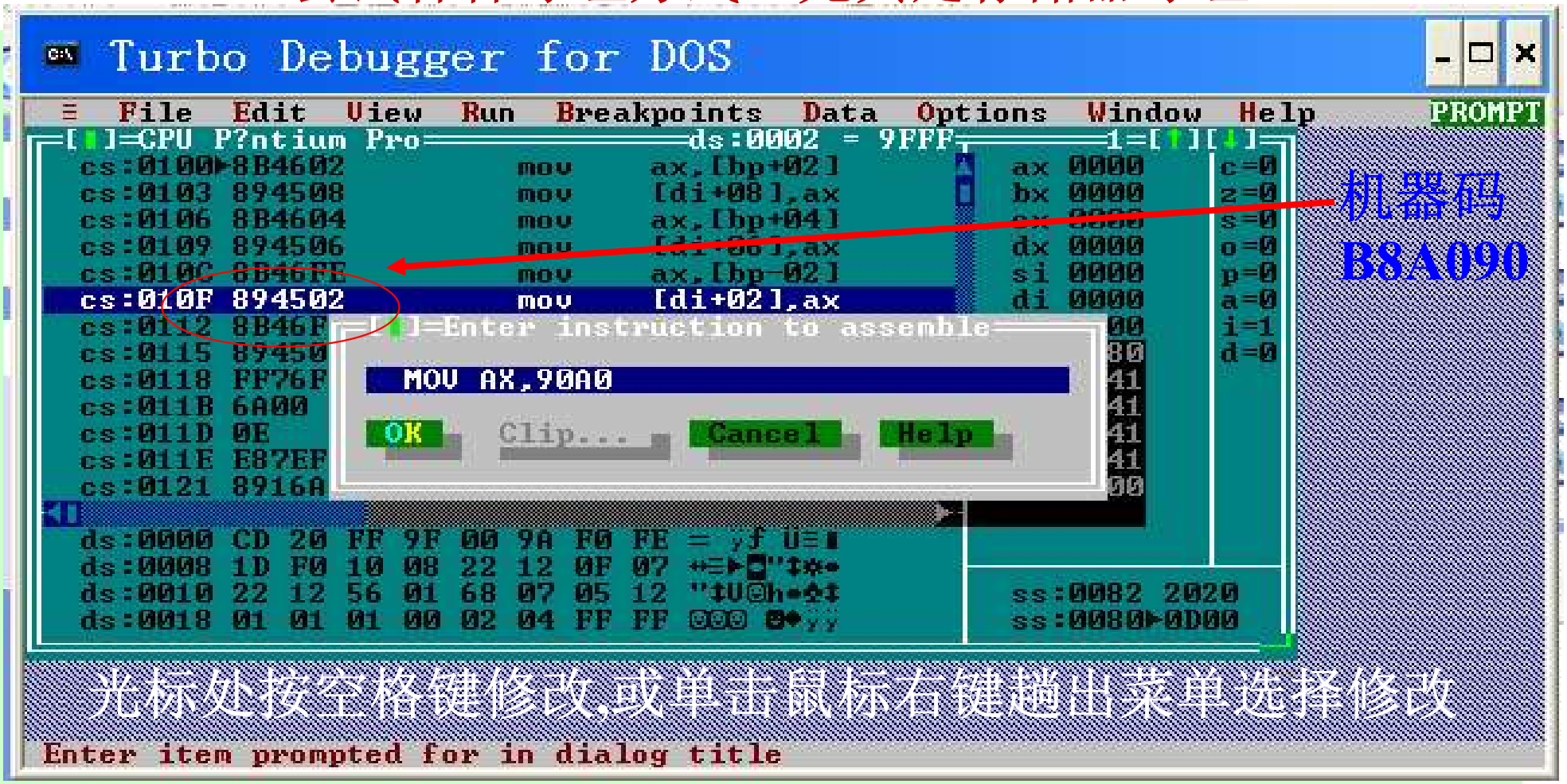


光标处按空格键修改,或单击鼠标右键趟出菜单选择修改

指令输入与功能认识

1.CPU窗口-代码区操作:在光标行按空格键弹出输入框,汇编或修改当前指令

例**MOV AX，90A0**，修改**CS，IP=010F**，观察**AX**内容，利用**RUN**下单步命令**F7/F8**，执行一条指令，结果：**IP=0112**，**AX=90A0**—尝试各种寻址方式，尤其是存储器寻址



指令输入与功能认识（默认十六进制数）

典型指令：MOV [2000]，AX ；如何确定DS：[2000]内容？

MOV CS：[2000]，BX

INC BYTE PTR [2000] （观察存储器字节单元变化）

DEC WORD PTR[BX] （观察存储器字单元变化，存储3000H）

LDS SI，[2000] 观察存储器双字单元内容=DS：SI？）

PUSH AX，PUSH BX，POP AX，POP BX

（观察堆栈指针和内容变化）

ADD [2000]，AX

AND AL，BX， SHL AX，1（CL）

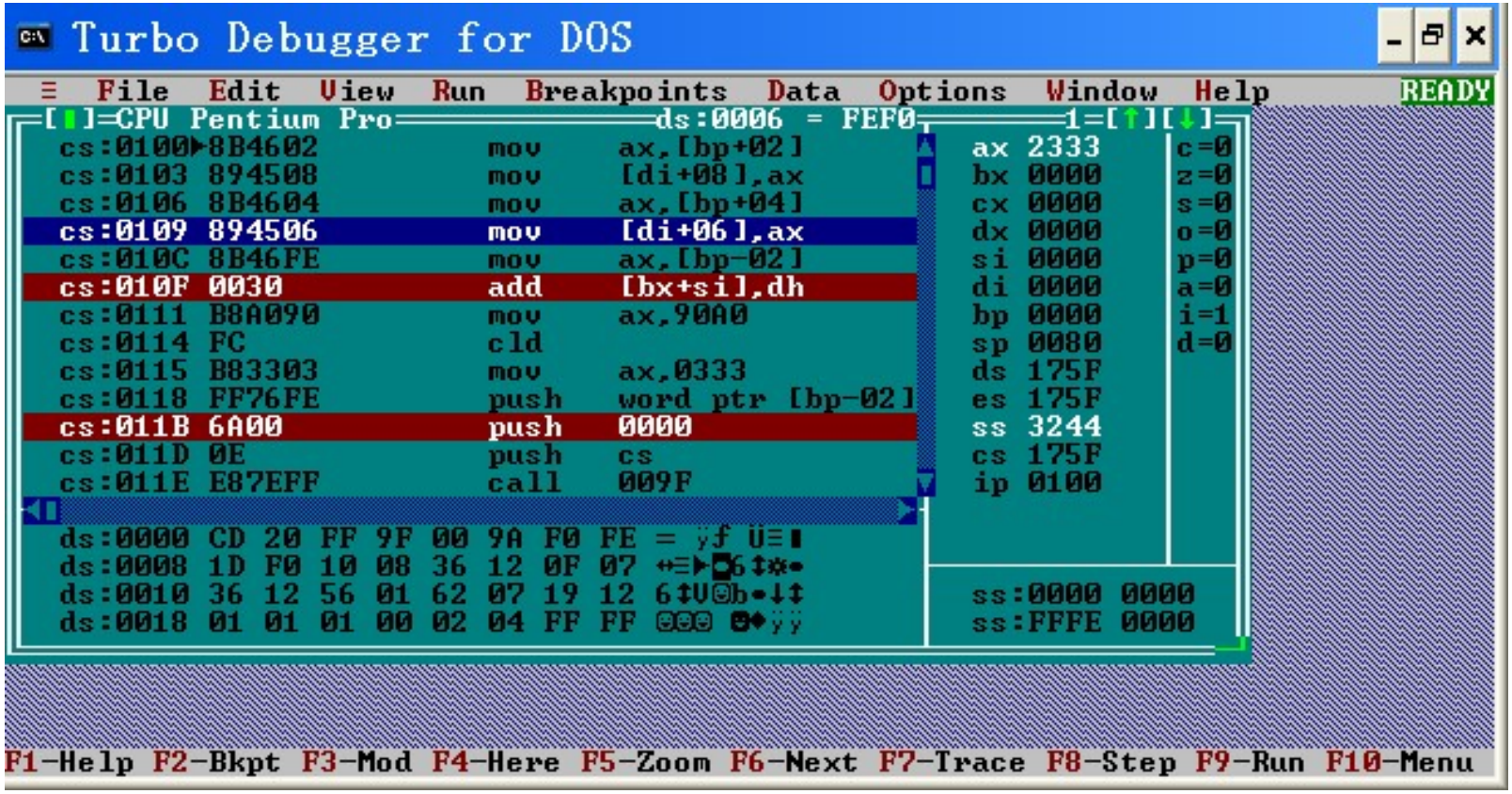
CLC， STD

REP MOVSB, MOVSW (先修改DS,SI;ES,DI,DF)

信息交换与运算指令

指令输入与功能认识（默认十六进制数）

顺序执行：设置断点：在蓝色光标行处按F2键指定或取消断点（或Breakpoints菜单下选Toggle),



一段程序：　程序流控制，顺序执行，断点结果检查

指令输入与功能认识（默认十六进制数）

条件转移：CS:1FC0　CMP AX,1000

JC 2000

JZ 2010

循环转移：　LOOP 1FA0

检查条件，目标CS:IP

无条件转移：　JMP 1FA8　(SHORT)

JMP 4020:809A

JMP NEAR PTR[2000]

JMP FAR PTR[2000]

寻址方式，检查目标CS:IP

程序流控制 1：执行过程

指令输入与功能认识（默认十六进制数）

过程调用：　CALL 2000H

CALL BX

CALL 2000:4321H

CALL NEAR/FAR PTR[BX]

检查堆栈
返回断点
目标CS:IP

过程返回：　：

RET

RETF

检查堆栈
返回断点

检查堆栈
返回断点
中断向量
目标CS:IP

中断调用：　INT 08H

中断向量：0000:0020H

中断返回：　…IRET

检查堆栈
返回断点

程序流控制2：小段过程程序 / 中断服务程序

指令输入与功能认识（默认十六进制数）

调试认识其他指令功能；

带符号调试功能：

支持符号调试宏汇编操作：

MASM/Zi myprog;

Link/codeview myprog;

TASM/zi myprog; 兼容MASM/Zi

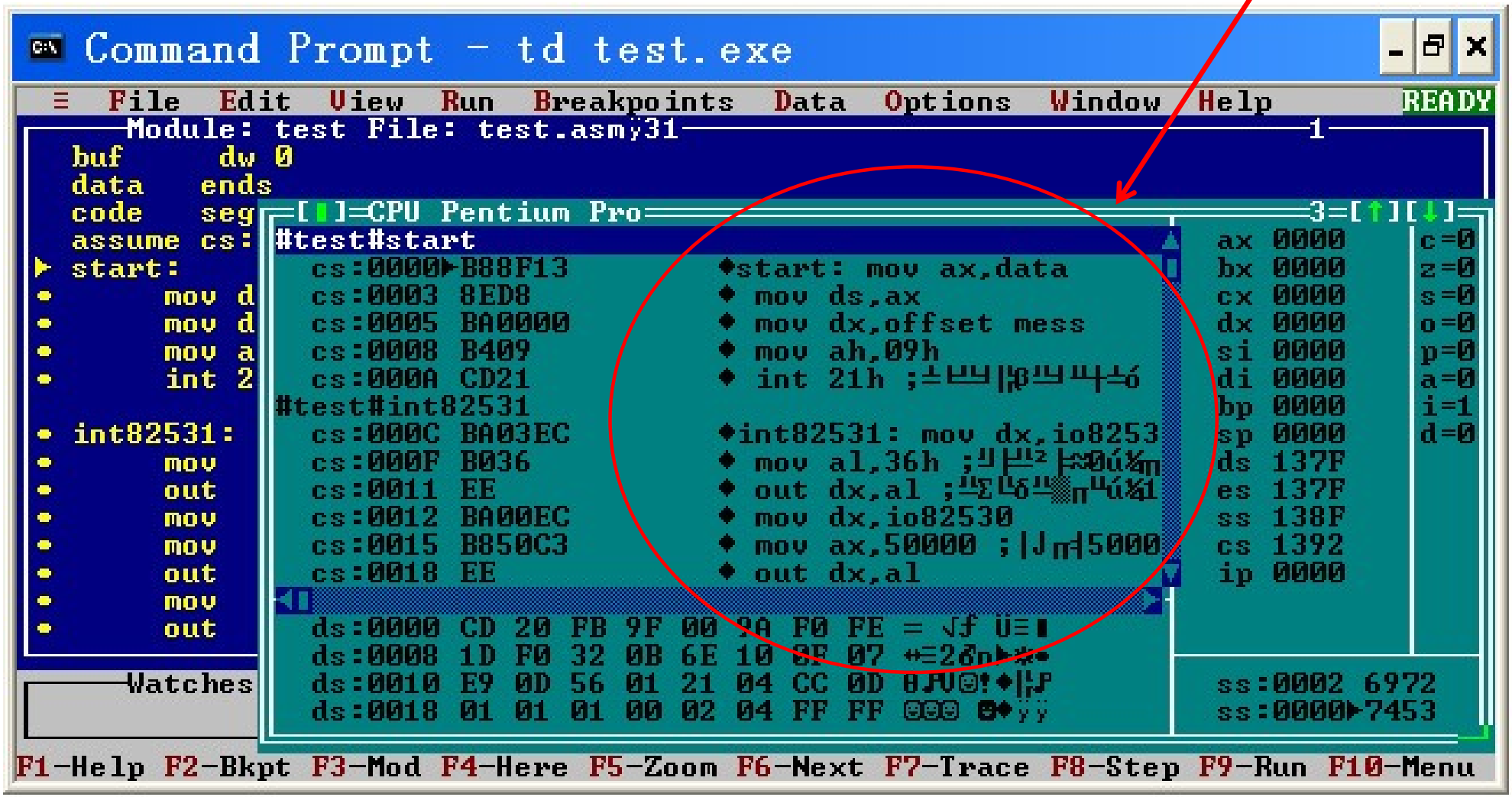
TLINK/v myprog+…;

可以直接操作\检查符号变量、标号！

指令输入与功能认识（默认十六进制数）

带符号调试功能：（打开CPU窗口可同时看到源程序）

支持符号调试宏汇编操作：



汇编过程及原理

运行汇编程序必备的软件环境：DOS操作系统；汇编软件系统。汇编系统盘应包含如下文件：

MASM	宏汇编程序文件	(TASM)
LINK	连接程序文件	(TLINK)
CREF	索引程序文件（也可不用）	
EDIT	文本编辑程序（或PE等文本编辑程序）	

用户通过屏幕编辑程序EDIT（各功能）键入源程序，检查无误，可将源程序存到汇编系统盘上，**该程序的扩展名为·ASM。（XXX.ASM）**

(2) MASM汇编程序：格式、宏、模块→OBJ: 浮动汇编（相对关系）、段、变量待定位

汇编链接过程

执行宏汇编程序MASM----生成目标文件.OBJ，辅助文件LST，CRF（可选）

用汇编语言编写的源程序必须是一个完整的源程序，才能经过宏汇编程序MASM的汇编，生成一个目标程序。为了完成汇编任务，汇编程序一般采用两遍扫描的方法，第一遍扫描源程序产生符号表、处理伪指令等，第二遍扫描产生机器指令代码、确定数据等。

OBJ将源程序的操作码部分变为机器码，但地址操作数是可浮动的相对地址，而不是实际地址，因此需经LINK连接文件进行连接才能形成可执行文件。

LST是列表文件把源程序和目标程序列表，以供检查程序用。

CRF是交叉索引文件，对源程序所用的各种符号进行前后对照的文件

汇编链接操作过程

C:\masm\masm ↙ [MYFILE[.asm][， mobj， mlst,mcrf]；

Microsoft （R） Macro Assemble Version 5.00

Copyright （C） Microsoft Corp 1981-1985,1987,All right reserved.

Source filename [.ASM]:MYFILE ↙

Object filename [MYFILE.OBJ]:MYFILE ↙

Source listing [NUL.LST]:MYFILE ↙

Cross-reference [NUL.CRF]:MYFILE ↙

50678＋410090 Bytes symbol space free

0 Warning Errors

0 Severe Errors

MASM/R　MASM/E----8087实模式 / 仿真库方式

LINK 多模块链接

用汇编语言编写的源程序经过汇编程序（MASM）汇编后产生了目标程序（.OBJ），该文件是将源程序操作码部分变成了机器码，但地址是可浮动的相对地址（逻辑地址），因此必须经过连接程序LINK连接后才能运行。连接程序LINK是把一个或多个独立的目标程序模块装配成一个可重定位的可执行文件，扩展名为.EXE文件。此外还可以产生一个内存映象文件，扩展名为.MAP。连接程序执行过程:

（LINK/HELP---帮助开关）

D>LINK ↙（直接链接 P1+P2+P3[, PEXE,PMAP,LIB1+LIB2] / M;）

Microsoft® overlay link Version 3.60

Copyright ©Microsoft Corp 1983-1987 All right reserved

Object Modules [.OBJ]：MYFILE ↙

Run File [MYFILE.EXE]：MYFILE ↙

List File [NUL.MAP]：MYFILE ↙[;]

Libraries [.LIB]： ↙

内存映象文件（.MAP）

由连接程序LINK产生的扩展名为.MAP文件，它实际上是连接程序的列表文件，它给出了每个段的地址分配情况及长度，（加/M开关—外部变量相对地址。例如：

D>TYPE MYFILE.MAP ↙

```
Start       Stop      Length      Name          Class
00000H     0000FH    0010H       DATA
00010H     0004FH    0040H       STACK
00050H     0005FH    0010H       CODE
Origin       Group
[Address Publics by name
0900:0002    mmm
0010:0070    VV1
 Address Publics by value
0010:0070            VV1
0900:0002    mmm  ]
Program entry point at 0005:0000
```

交叉索引文件（.CRF）

汇编后产生的交叉索引文件，扩展名为.CRF,它列出了源程序中定义的符号（包括：标号、变量等）和程序中引用这些符号的情况。

如果要查看这个符号表，必须使用CREF.EXE的文件，它根据.CRF文件建立一个扩展名为.REF的文件，而后再用DOS的TYPE命令显示，就可以看到这个符号使用情况表。具体操作方法如下：

D>CREF ↙

cref filename [.CRF]：MYFILE ↙

list filename [MYFILE.REF]： ↙

D>TYPE MYFILE.REF ↙

执行、目标代码格式比较与调试

.EXE装入后各寄存器（段的分配）、已定义变量（数据）地址，N / F 指针变量；

.各伪指令的作用；

.多模块（PUBLIC/EXTRN）链接； / M,8087指令实验。

.汇编指令与目标代码指令的差别。

.重要DOS/BIOS功能调用（控制台输入 / 输出）

.不要试图跟踪 DOS/BIOS调用，尽量用断点或过程P命令

.注意：高级语言与汇编语言接口

调试工具

(1)DEBUG----命令行目标代码调试器

(2)CV—CodeView　Microsoft公司源代码调试器

MASM/Zi　myprog;

Link/codeview　myprog;

(3)TD—Turbo Debuger Borland公司源代码调试器

TASM/zi　myprog;　兼容MASM/Zi

TLINK/v myprog+...;

(4) Compuware Co.的NUMEGA Soft-ICE

实际操作

MASM T18;

MASM T19;

LINK T18+T19;

TASM/zi T18;

TLINK/v T18;

TD T18(.exe)

F3重复命令行