

Es1. DICHIARAZIONE DI VARIABILI

;per visualizzare il risultato viene utilizzato il turbo debug TD

.MODEL SMALL

.STACK

.DATA

num1 DB 5H ; dichiarazione variabile

.CODE

BEGIN:

MOV AX,@data ; imposto il registro DS con @data= l'indirizzo inizio del segmento dati
MOV DS,AX ;

mov al, num1 ; posso copiare un byte su registri da 8 bit e word su registri da 16 bit
mov ax, num1 ; dà errore – provare!!!

MOV AX,4C00H ;funzione DOS Exit
INT 21H

END BEGIN

TIPI DI VARIABILI

Nome_variabile **DB** valore

può assumere sono quattro e ne definiscono la massima grandezza. Essi sono:

- **DB** (define byte): la variabile è grande 1 byte;

Esempi:

num DB 16 ; riserva 1 byte contenente 16 decimale
num DB 16h ; riserva 1 byte contenente 16 esadecimale
num DB 16o ; riserva 1 byte contenente 16 ottale
num DB 'A' ; riserva 1 byte di valore pari al codice ASCII di A
num DB 'CIAO'
num DB 'C', 'I', 'A', 'O'

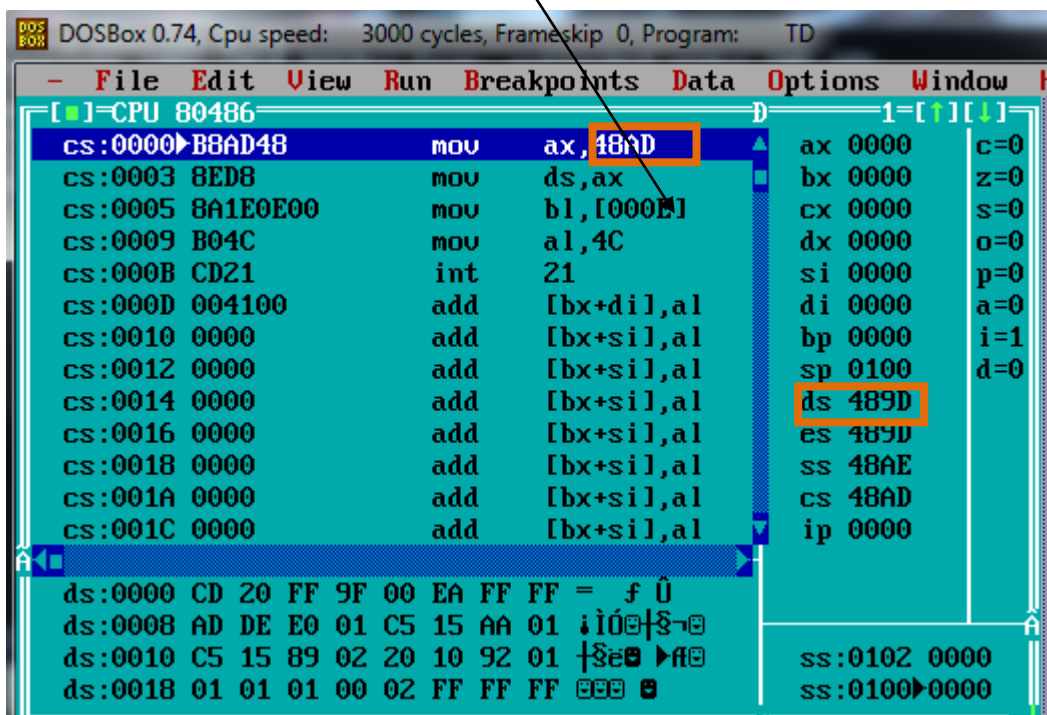
- **DW** (define word): la variabile è grande 2 byte;
- **DD** (define double): la variabile è grande 4 byte;
- **DQ** (define quad): la variabile è grande 8 byte
- **DT** (define ten): la variabile è grande 10 byte

Es2. Scrivi il seguente programma ed esegui con td.

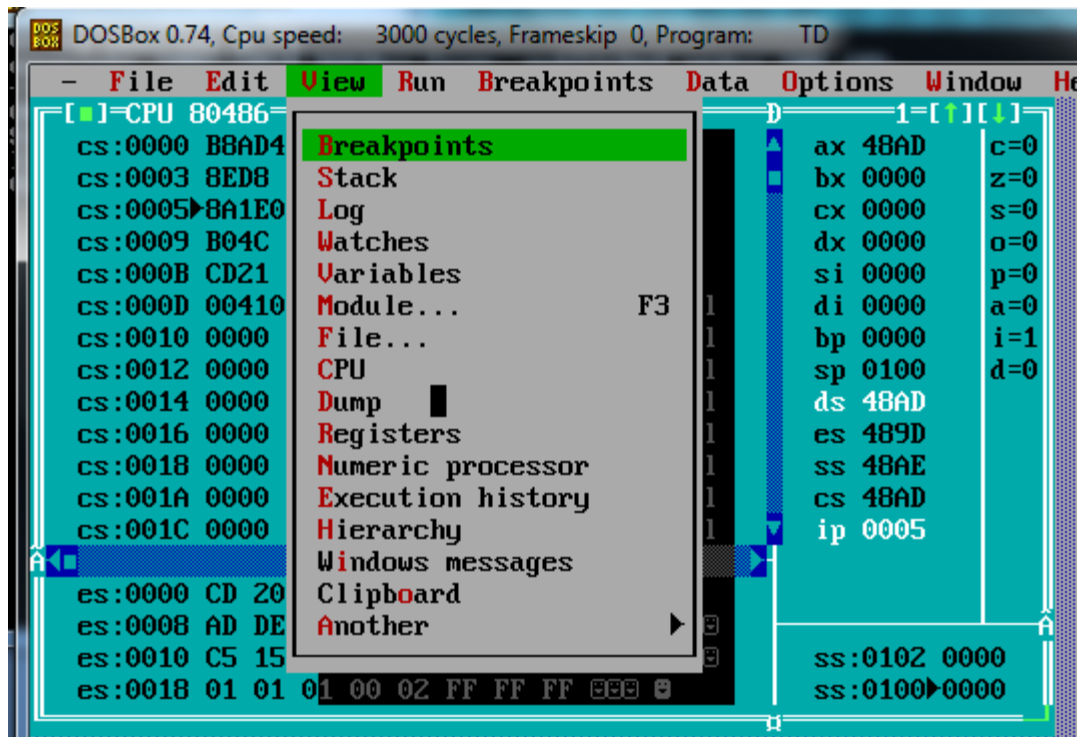
```
.MODEL tiny
.STACK
.DATA
    car DB 'A' ; dichiarazione variabile
.CODE
BEGIN:

    MOV AX,@data
    MOV DS,AX
    mov al, car
    MOV AX,4C00H ;funzione DOS Exit
    INT 21H
END BEGIN
```

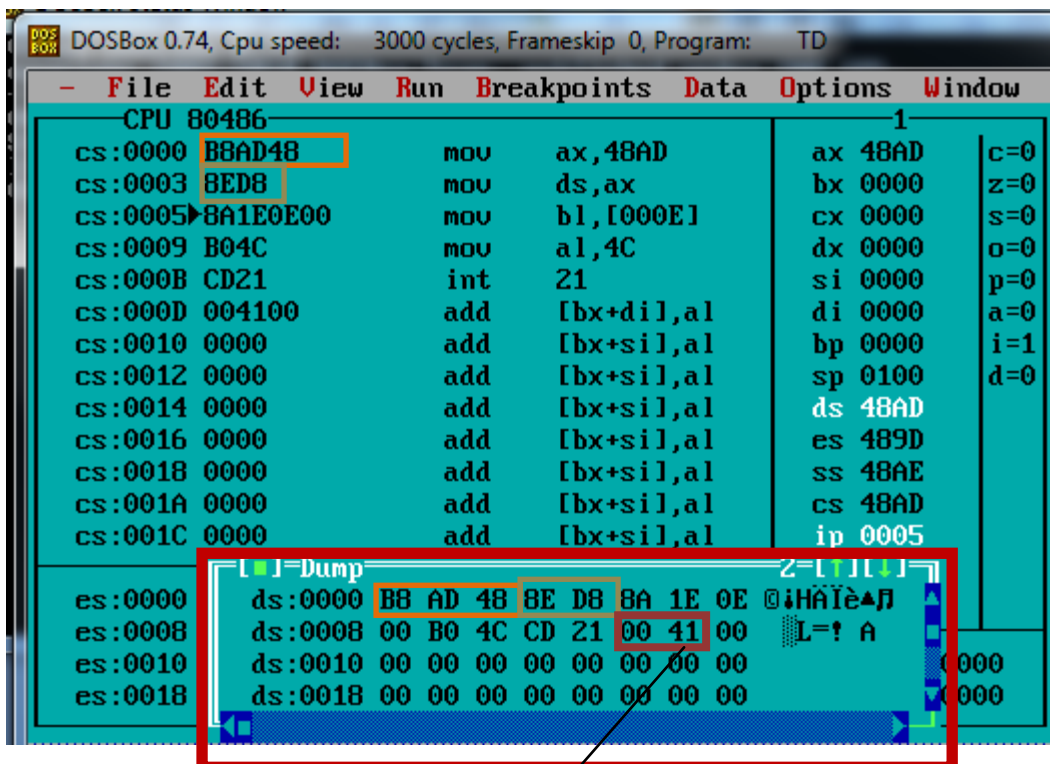
- Assembla
- esegui con TD
- Appare la finestra riportata sotto.
- Notare i: @data è diverso da ds
- Notare: nell'istruzione `mov al, car` non viene visualizzato il valore contenuto nella variabile `car` ma il primo indirizzo libero della memoria dove viene memorizzato il dato.



- Eseguire due F8 e visualizzare la memoria (view → dump)



- Vi appare la **finestra Dump**



In questa finestra è riportato il segmento dati (ds) che contiene le istruzioni del programma e nella prima cella libera il contenuto della variabile (codice ascii del carattere A)

Es3. SCAMBIO DI DUE VARIABILI

1. Prima di memorizzare le due variabili num1='B' e num2='A' voglio memorizzare 10 asterischi,

```
ds:0008 2A 2A 2A 2A 2A 2A 2A 2A *****
ds:0010 2A 2A 42 41 00 00 00 00 * BA
```

Con l'operatore **DUP** nella sezione DATA

NomeVar DB n°Caratteri DUP('*')

DUP crea uno spazio di stringa di 10 caratteri specificati dentro parentesi, in questo caso *

- Per creare uno spazio di 10 zero: spazio DB 0Ah DUP(0)

2. OUTPUT esercizio:

```
[J]=Dump 2=[↑][↓]
ds:0000 A2 13 00 B4 4C CD 21 00 6!! |L=?
ds:0008 2A 2A 2A 2A 2A 2A 2A 2A *****
ds:0010 2A 2A 41 42 00 00 00 00 * AB
ds:0018 00 00 00 00 00 00 00 00
```

```
;programma che scambia due variabili num1 e num2
; in c:      c=num1          <- in Assembly non posso spostare una variabile in un'altra variabile ovvero
;           num1=num2       non posso spostare una locazione di memoria in un'altra locazione di memoria
;           num2=c          ma devo passare attraverso un registro

.MODEL SMALL
.STACK
.DATA
    spazio DB 0Ah dup('*') ; corrisponde ascii 2Ah
    num1 DB 'B'            ; corrisponde ascii 42
    num2 DB 'A'            ; carattere ascii 41

.CODE
BEGIN:
    mov ax,@data ;@data contiene l'indirizzo di inizio del segmento dati
    mov ds,ax

    mov al,num1
    mov bl,num2
    mov num1,bl
    mov num2,al

    mov AH,4CH ;Queste due istruzioni serve a far terminare correttamente il programma
    int 21H    ; Ritorno al Sistema Operativo
END BEGIN
```

Prima dello scambio:

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: TD
- File Edit View Run Breakpoints Data Options Window H
CPU 80486
cs:0000 B8AE48 mov ax,48AE ax 48AE c=0
cs:0003 8ED8 mov ds,ax bx 0000 z=0
cs:0005 A01200 mov al,[0012] cx 0000 s=0
cs:0008 8A1E1300 mov bl,[0013] dx 0000 o=0
cs:000C 8B1E1200 mov [0012],bl si 0000 p=0
cs:0010 A21300 mov [0013],al di 0000 a=0
cs:0013 B44C mov ah,4C bp 0000 i=1
cs:0015 CD21 int 21 sp 0400 d=0
ds:0000 A2 13 00 B4 4C CD 21 00 6!! |L=?
ds:0008 2A 2A 2A 2A 2A 2A 2A 2A *****
ds:0010 2A 2A 42 41 00 00 00 00 * BA
ds:0018 00 00 00 00 00 00 00 00
ss:48B0 2A 2A 2A 2A 2A 2A 2A *****
ss:48B8 2A 2A 42 41 00 00 00 00 * BA
ip 0005
es:0000 CD 20 FF 9F 00 EA FF FF = f 0
es:0008 AD DE E0 01 C5 15 AA 01 i00|S-0
es:0010 C5 15 89 02 20 10 92 01 |S-0 000
es:0018 01 01 01 00 02 FF FF FF 000 0
```

Continuando con F8 ottengo:

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: TD
- File Edit View Run Breakpoints Data Options Window
CPU 80486
cs:0000 B8AE48      mov     ax,48AE      ax 4842  c=0
cs:0003 8ED8        mov     ds,ax        bx 0041  z=0
cs:0005 A01200      mov     al,[0012]    cx 0000  s=0
cs:0008 8A1E1300    mov     bl,[0013]    dx 0000  o=0
cs:000C 8B1E1200    mov     [0012],bl    si 0000  p=0
cs:0010 A21300      mov     [0013],al    di 0000  a=0
cs:0013 B44C        mov     ah,4C        bp 0000  i=1
cs:0015 CD21        int     21          sp 0400  d=0

[ ]=Dump 2=[↑][↓]
ds:0000 A2 13 00 B4 4C CD 21 00 6!! |L=?
ds:0008 2A 2A 2A 2A 2A 2A 2A 2A *****
ds:0010 2A 2A 41 42 00 00 00 00 *AB
ds:0018 00 00 00 00 00 00 00 00

es:0000 CD 20 FF 9F 00 EA FF FF = f U
es:0008 AD DE E0 01 C5 15 AA 01 i00|S-@
es:0010 C5 15 89 02 20 10 92 01 |Se@ >A@
es:0018 01 01 01 00 02 FF FF FF @@@ @

ss:0402 0000
ss:0400 0000
```

Es4. PROVA TU. Modifica l'esercizio in modo da scambiare tre variabili stampando prima e dopo cinque asterischi.

Prima dello scambio:

```
[ ]=Dump 2=[↑][↓]
ds:0000 2A 2A 2A 2A 2A 43 42 41 *****CBA
ds:0008 2A 2A 2A 2A 2A 00 00 00 *****
ds:0010 00 00 00 00 00 00 00 00
```

Dopo lo scambio

```
- File Edit View Run Breakpoints Data Options Window
CPU 80486
cs:0000 B8AF48      mov     ax,48AF      ax 4843
cs:0003 8ED8        mov     ds,ax        bx 4142
cs:0005 A00500      mov     al,[0005]    cx 0000
cs:0008 8A1E0600    mov     bl,[0006]    dx 0000
cs:000C 8A3E0700    mov     bh,[0007]    si 0000
cs:0010 8B3E0500    mov     [0005],bh    di 0000
cs:0014 8B1E0600    mov     [0006],bl    bp 0000
cs:0018 A20700      mov     [0007],al    sp 0400
cs:001B B44C        mov     ah,4C        ds 48AF
cs:001D CD21        int     21          es 489D

[ ]=Dump 2=[↑][↓]
ds:0000 2A 2A 2A 2A 2A 41 42 43 *****ABC
ds:0008 2A 2A 2A 2A 2A 00 00 00 *****
ds:0010 00 00 00 00 00 00 00 00
ds:0018 00 00 00 00 00 00 00 00

ss 48B0
cs 48AD
ip 001B
```

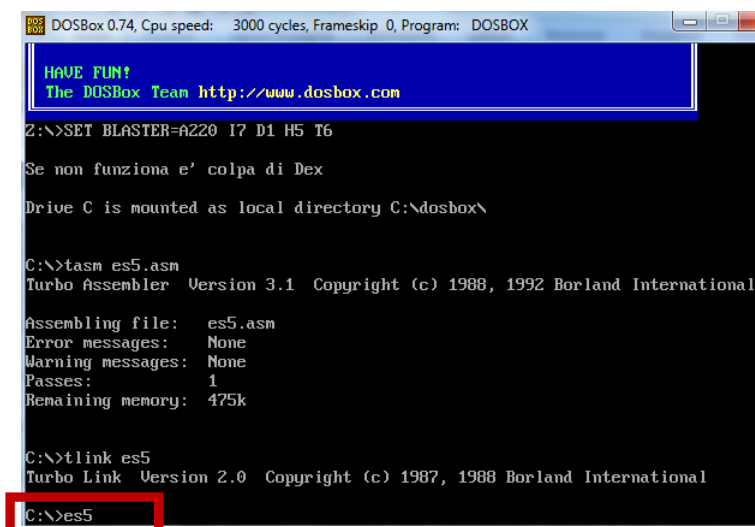
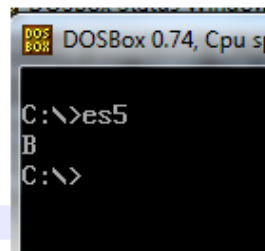
STAMPARE UN CARATTERE A VIDEO

```
MOV DL, carattereDaStampare ;memorizzo il carattere da stampare in DL
MOV AH, 02h                 ; stampa a video il carattere in DL
INT 21h
```

- Per utilizzare la funzione 02h dell'interrupt 21 bisogna memorizzare il carattere in **DL**.
- Per visualizzare il risultato, eseguire il programma direttamente dal prompt non con td.

Es 5. Stampare la lettera B.

```
1 ;programma che stampa a video la lettera B
2
3 .MODEL SMALL
4 .STACK
5 .DATA
6 .CODE
7 BEGIN:
8
9     mov dl, 'B'
10    mov ah, 02h
11    int 21h
12
13    mov AH, 4CH ;Queste due istruzioni serve a far terminare correttamente :
14    int 21H    ; Ritorno al Sistema Operativo
15 END BEGIN
16
```



Es6. Stampare la lettera B contenuta in una variabile.

```
1 ;programma che stampa a video la lettera B contenuta nella variabile
2
3 .MODEL SMALL
4 .STACK
5 .DATA
6     lettera DB 'B'
7 .CODE
8 BEGIN:
9     mov ax, @data
10    mov ds, ax
11
12    mov dl, lettera
13    mov ah, 02h
14    int 21h
15
16    mov AH, 4CH ;Queste due istruzioni serve a far terminare correttamente il progra
17    int 21H    ; Ritorno al Sistema Operativo
18 END BEGIN
```

STAMPARE UNA STRINGA A VIDEO

```
.DATA
    msg DB 'Ciao!$'      ; la stringa deve terminare con il simbolo dollaro $
.CODE
...

MOV DX, OFFSET msg      ; indicare in DX l'inizio della stringa da stampare
MOV AH, 09h             ; stampa a video la stringa
INT 21h

...
```

- La stringa deve terminare con \$
- L'operatore **offset** restituisce l'offset di una variabile, cioè la distanza in byte della variabile dall'inizio del segmento dati, in questo caso ci dice l'indirizzo di inizio frase da stampare.
- L'offset va memorizzato in DX.
- Per visualizzare il risultato, eseguire il programma direttamente dal prompt non con td.

Es7. Stampare la scritta Hello world! memorizzata nella variabile msg

```
1 ;programma che stampa a video la scritta Hello World!
2
3 .MODEL SMALL
4 .STACK
5
6 .DATA
7     msg DB 'Hello world!$'
8 .CODE
9 BEGIN:
10
11     mov ax, @data
12     mov ds, ax
13
14     mov dx, offset msg
15     mov ah, 09h      ; function 09h - visualizza stringa
16     int 21h
17
18     mov AH, 4CH      ;Queste due istruzioni serve a far terminare correttamente il programma
19     int 21H          ; Ritorno al Sistema Operativo
20 END BEGIN
```

Es8. Stampare il proprio nome.

Es9. Stampa la scritta 'Anno 2015' con i numeri in ASCII.

```
;programma che stampa a video la scritta Anno 2015

.MODEL SMALL
.STACK

.DATA
    msg DB 'Anno',32,50,48,49,53,'$'      ; 32=spazio, 50=2, 48=0, 49=1, 53=5
.CODE
BEGIN:

    mov ax, @data ;@data contiene l'indirizzo di inizio del segmento dati
    mov ds, ax

    mov dx, offset msg ; indico indirizzo d'inizio della stringa in DX
    mov ah, 09h        ; function 09h - visualizza stringa
    int 21h

    mov AH, 4CH        ;Queste due istruzioni serve a far terminare correttamente il program
    int 21H            ; Ritorno al Sistema Operativo
END BEGIN
```

Es10. Stampa BUON NATALE usando solo i caratteri ascii.

LETTURA DI UN CARATTERE DA TASTIERA

```
MOV AH, 08h          ; legge il carattere premuto e memorizza il carattere
INT 21h              ; ASCII in AL
```

- Il carattere digitato da tastiera viene inserito in AL

Con echo (il carattere viene ripetuto due volte):

```
MOV AH, 01h          ; legge il carattere premuto e memorizza il carattere
INT 21h              ; ASCII in AL
```

Es11. Programma che legge un carattere da tastiera e lo visualizza a video

```
1 ;programma che legge un carattere da tastiera e lo visualizza a video
2
3 .MODEL SMALL
4 .STACK
5
6 .DATA
7
8 .CODE
9 BEGIN:
10
11     mov ah, 08h
12     int 21h
13
14     mov dl, al          ; sposto il carattere in dl per visualizzarlo a video
15     mov ah, 02h         ; function 02h - visualizza carattere
16     int 21h
17
18     mov AH, 4CH         ;Queste due istruzioni serve a far terminare correttamente il programma
19     int 21H            ; Ritorno al Sistema Operativo
20 END BEGIN
```

Es12. Programma che chiede di inserire un numero da tastiera compreso tra 0 e 9, lo legge, lo incrementa e lo visualizza a video

INC

INC OP1

Incrementa di 1 il contenuto dell' operando (OP1)

```
1 ;programma che legge un carattere da tastiera e lo visualizza a video
2
3 .MODEL SMALL
4 .STACK
5
6 .DATA
7     numero DB ?
8     msg DB 'Inserisci un numero compreso tra 0 e 9 $'
9
10 .CODE
11 BEGIN:
12     mov ax, @data
13     mov ds, ax
14
15     mov dx, offset msg
16     mov ah, 09h
17     int 21h
18
19     mov ah, 08h
20     int 21h
21
22     mov numero, al
23     inc numero
24     mov dl, numero
25     mov ah, 02h        ; function 02h - visualizza carattere
26     int 21h
27
28     mov AH, 4CH        ;Queste due istruzioni serve a far terminare correttamente il p
29     int 21H           ; Ritorno al Sistema Operativo
30
31 END BEGIN
```