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:

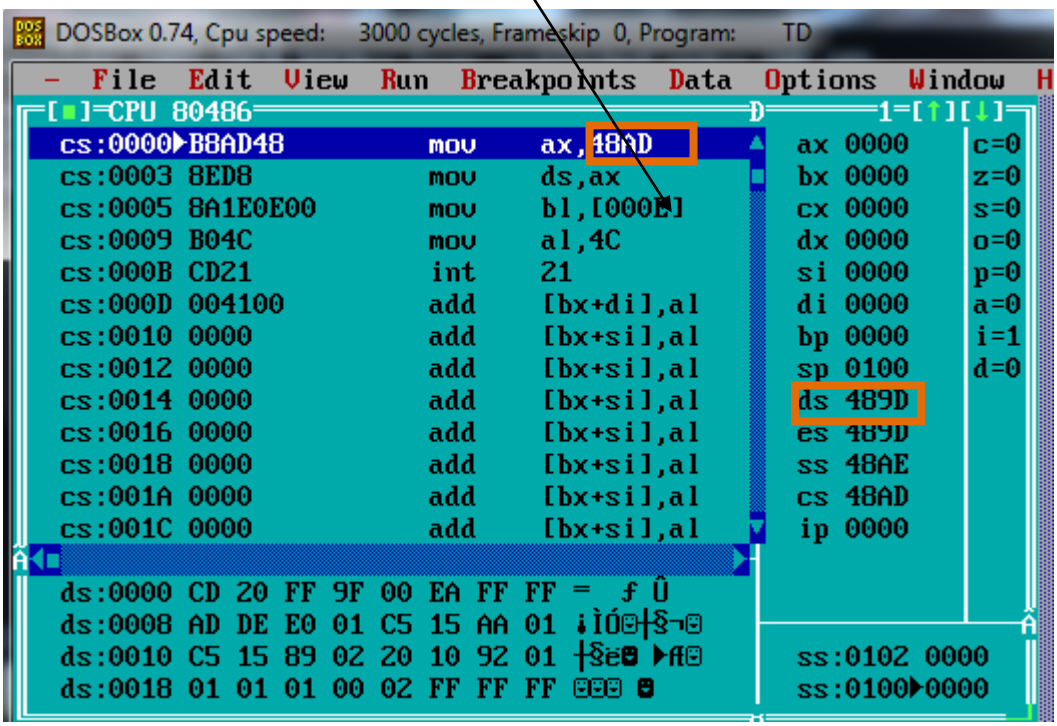
```
num DB 16      ; riserva 1 byte contenente 16 decimale
num DB 16h     ; riserva 1 byte contenente 16 esadecimale
num DB 16o     ; riserva 1 byte contenente 16 ottale
num DB 'A'     ; riserva 1 byte di valore pari al codice ASCII di A
num DB 'CIAO'
num DB 'C', 'I','A','O'
```

- **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

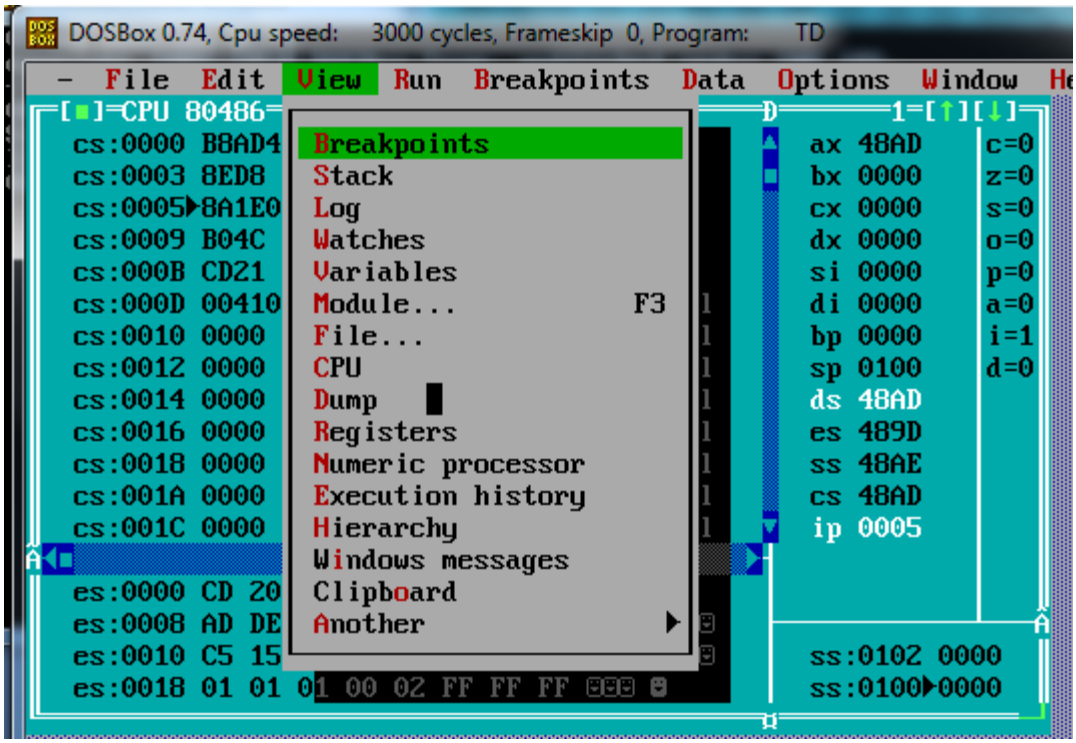
Es2. Scrivi il seguente programma ed esegui con td.

```
    .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
```

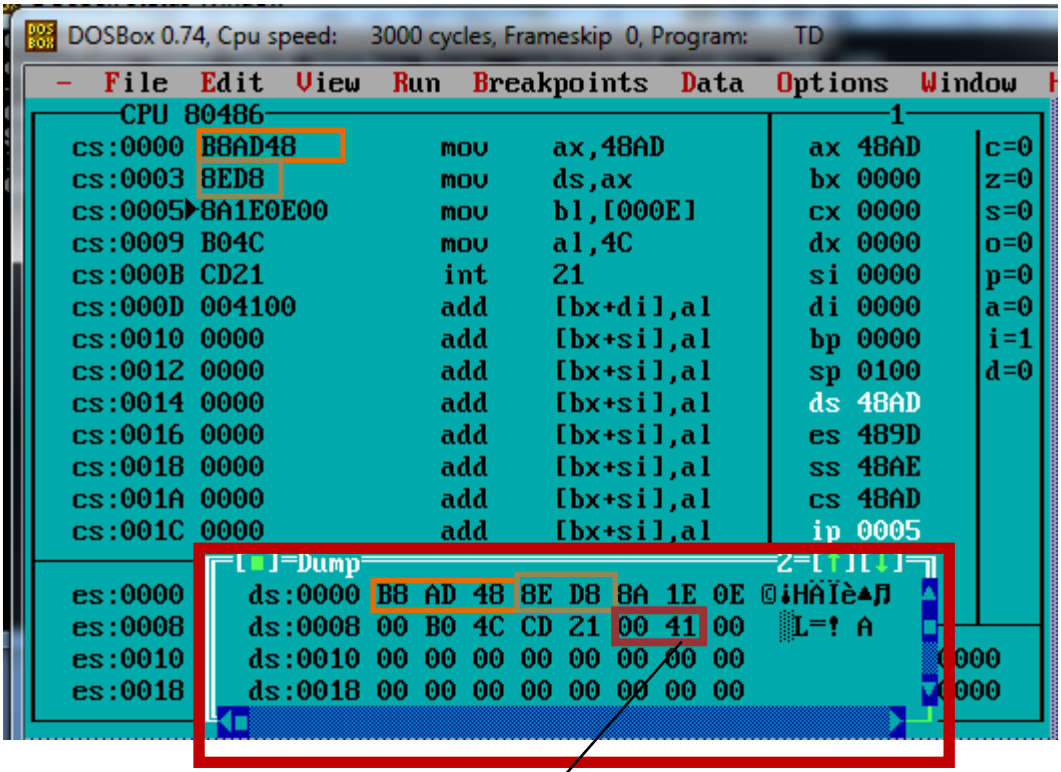
- Assembla
- esegui con TD
- Appare la finestra riportata sotto.
- Notare i: @data è diverso da ds
- Notare: nell'istruzione mov al, car non viene visualizzato il valore contenuto nella variabile car ma il primo indirizzo libero della memoria dove viene memorizzato il dato.



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



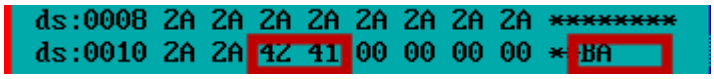
- Vi appare la **finestra Dump**



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,



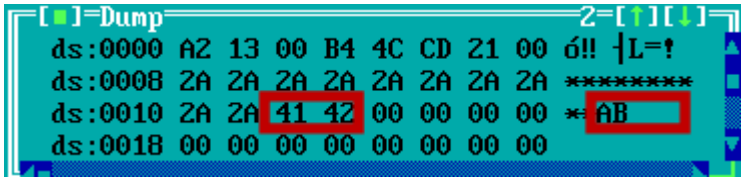
   Con l'operatore **DUP** nella sezione DATA

   NomeVar DB n°Caratteri DUP('*')

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

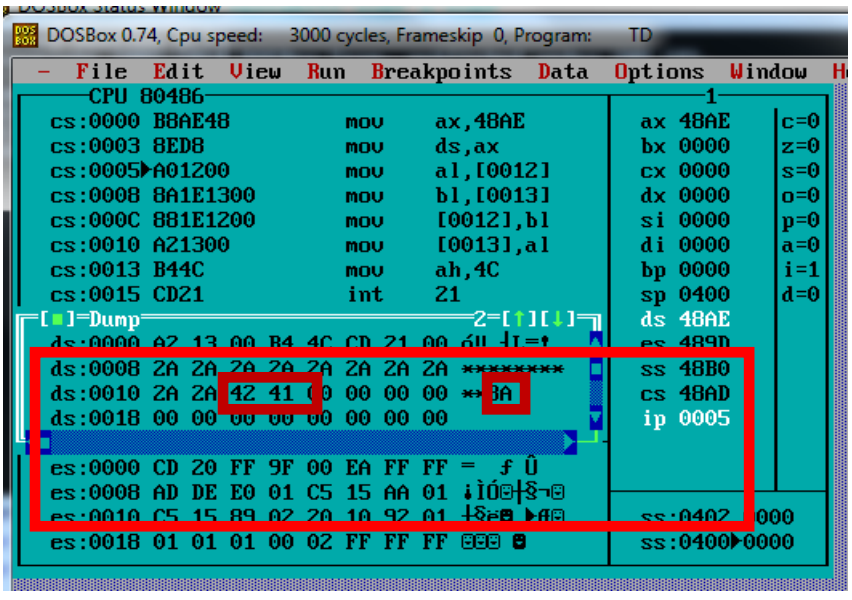
- Per creare uno spazio di 10 zero: spazio DB 0Ah DUP(0)

2. OUTPUT esercizio:

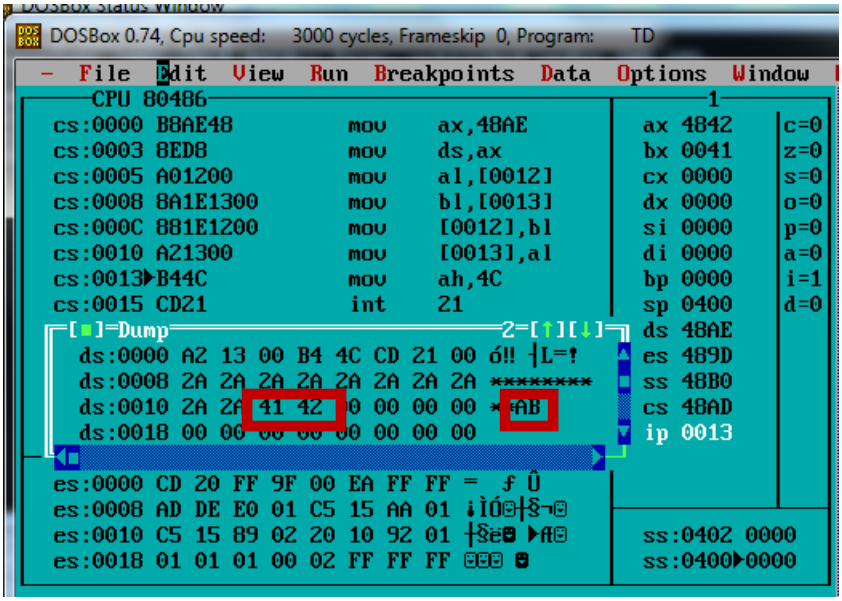



```
;programma che scambia due variabili num1 e num2
; in c:     c=num1                <- in Assembly non posso spostare una variabile in un'altra variabile ovvero
;           num1=num2                non posso spostare una locazione di memoria in un'altra locazione di memoria
;           num2=c    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

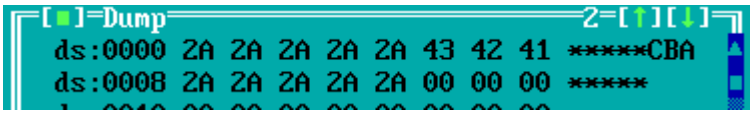


Continuando con F8 ottengo:

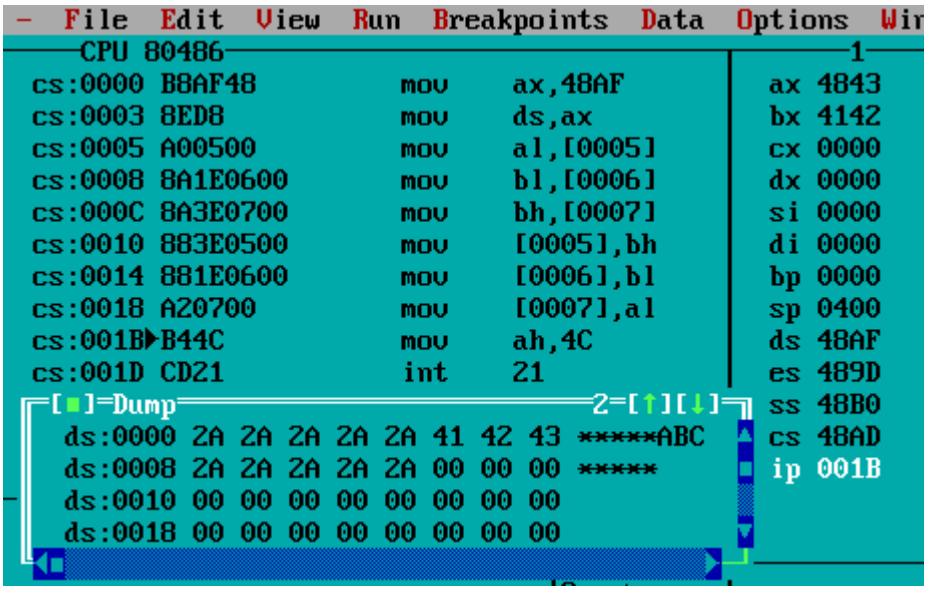


Es4. PROVA TU. Modifica l'esercizio in modo da scambiare tre variabili stampando prima e dopo cinque asterischi.

Prima dello scambio:



Dopo lo scambio



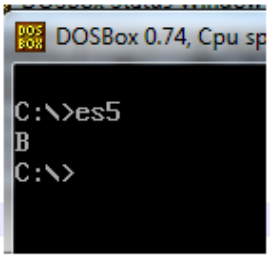
STAMPARE UN CARATTERE A VIDEO

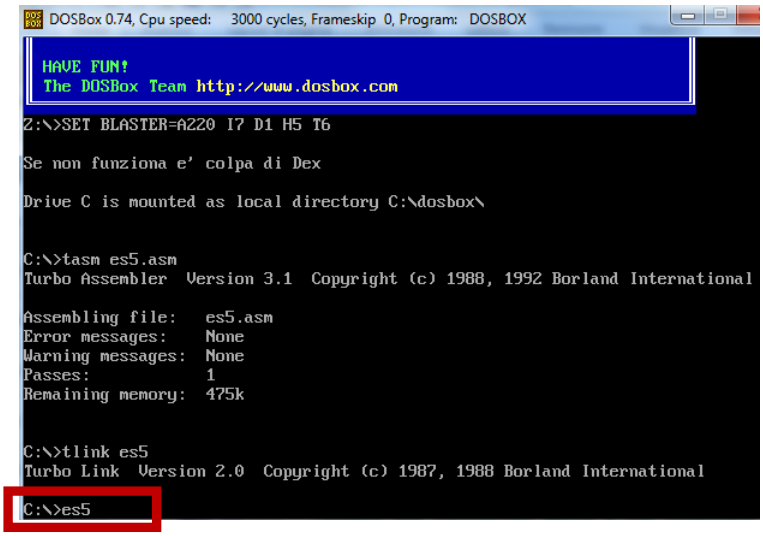
```
MOV DL , carattereDaStampare   ;memorizzo il carattere da stampare in DL
MOV AH, 02h                    ; stampa a video il carattere in DL
INT 21h
```

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

```
;programma che stampa a video la lettera B contenuta nella variabile

     .MODEL SMALL
     .STACK
     .DATA
          lettera DB 'B'
     .CODE
BEGIN:
       mov ax, @data
       mov ds,ax

       mov dl,lettera
       mov ah,02h
       int 21h

       mov AH,4CH    ;Queste due istruzioni serve a far terminare correttamente il progra
       int 21H       ; Ritorno al Sistema Operativo
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 promp non con td.

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

```
;programma che stampa a video la scritta Hello World!

     .MODEL SMALL
     .STACK

     .DATA
          msg DB 'Hello world!$'
     .CODE
BEGIN:

       mov ax,@data
       mov ds,ax

       mov dx, offset msg
       mov ah,09h         ; function 09h - visualizza stringa
       int 21h

       mov AH,4CH    ;Queste due istruzioni serve a far terminare correttamente il progra
       int 21H       ; Ritorno al Sistema Operativo
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 prograr
       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

```
;programma che legge un carattere da tasiera e lo visuyalizza a video

     .MODEL SMALL
     .STACK

     .DATA

     .CODE
BEGIN:

       mov ah, 08h
       int 21h

       mov dl,al          ; sposto il carattere in dl per visualizzarlo a video
       mov ah,02h         ; function 02h - visualizza carattere
       int 21h

       mov AH,4CH    ;Queste due istruzioni serve a far terminare correttamente il prograı
       int 21H       ; Ritorno al Sistema Operativo
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)

```
;programma che legge un carattere da tasiera e lo visuyalizza a video

      .MODEL SMALL
      .STACK

      .DATA
            numero DB ?
            msg DB 'Inserisci un numero compreso tra 0 e 9 $'
      .CODE
BEGIN:
        mov ax, @data
        mov ds,ax

        mov dx,offset msg
        mov ah,09h
        int 21h

        mov ah, 08h
        int 21h

        mov numero, al
        inc numero
        mov dl,numero
        mov ah,02h          ; function 02h - visualizza carattere
        int 21h

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