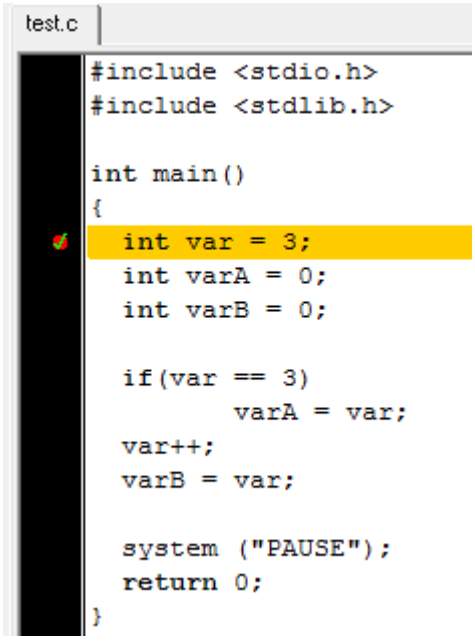
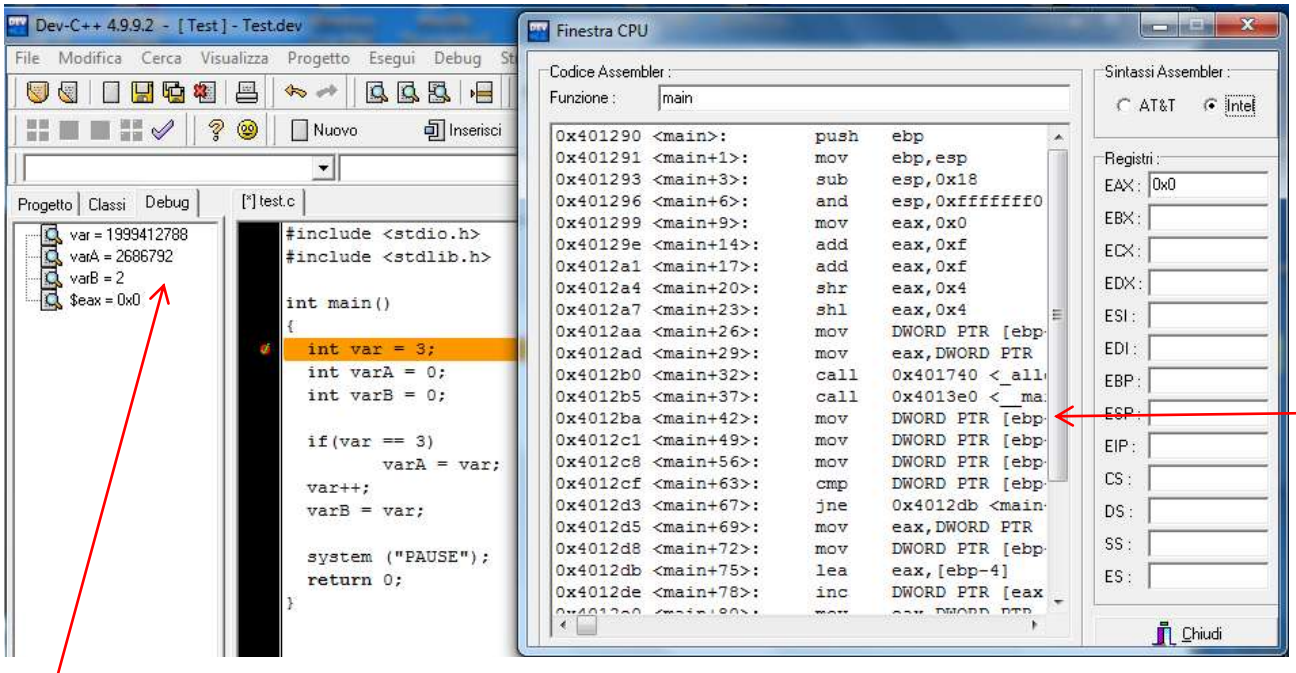
USO DEV_C++: DEBUGGING

Codice

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int var = 3;
 int varA = 0;
 int varB = 0;

 if(var == 3)
      varA = var;
 var++;
 varB = var;

 system ("PAUSE");
 return 0;
}
```



Primo step

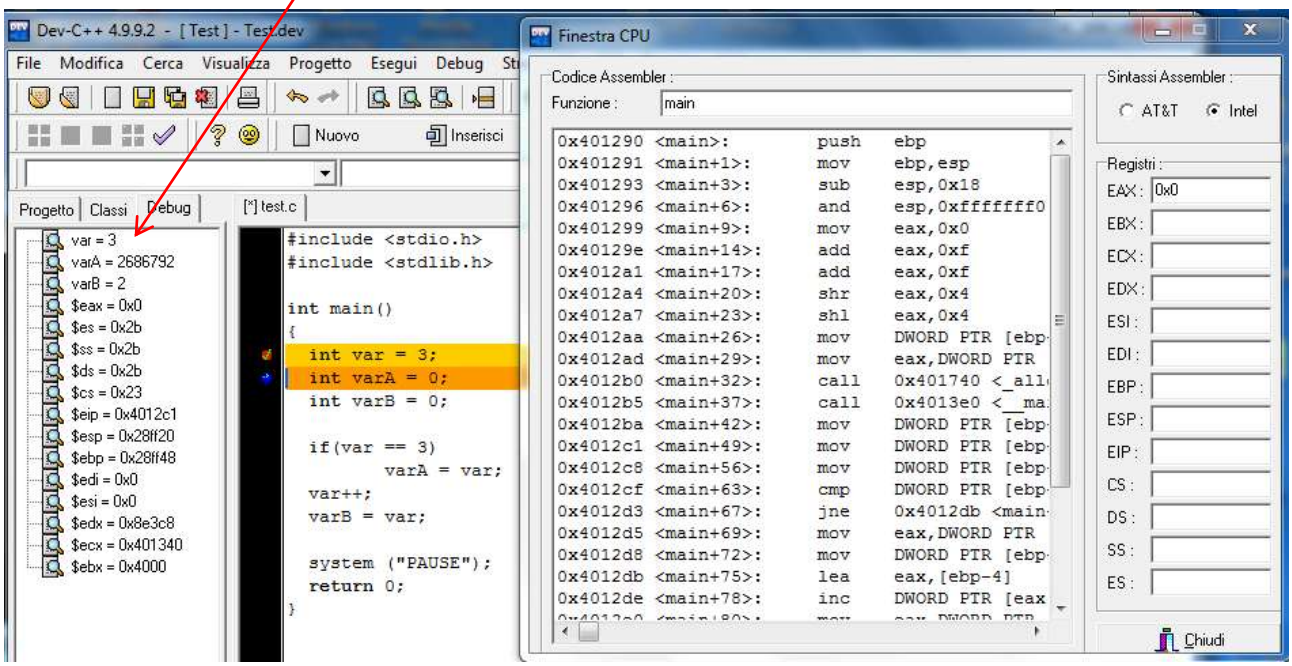


Le variabili contengono valori casuali

L'accumulatore (registro EAX) è inizializzato a zero

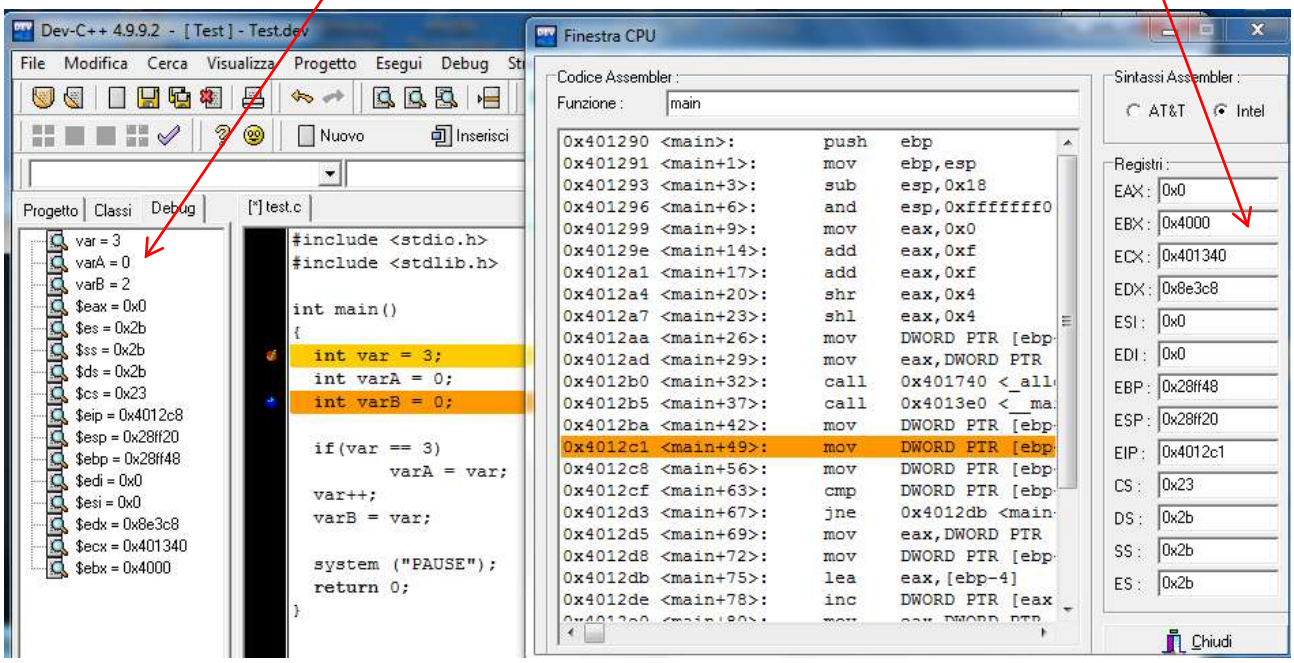
Secondo step

Ora **var** contiene il valore 3

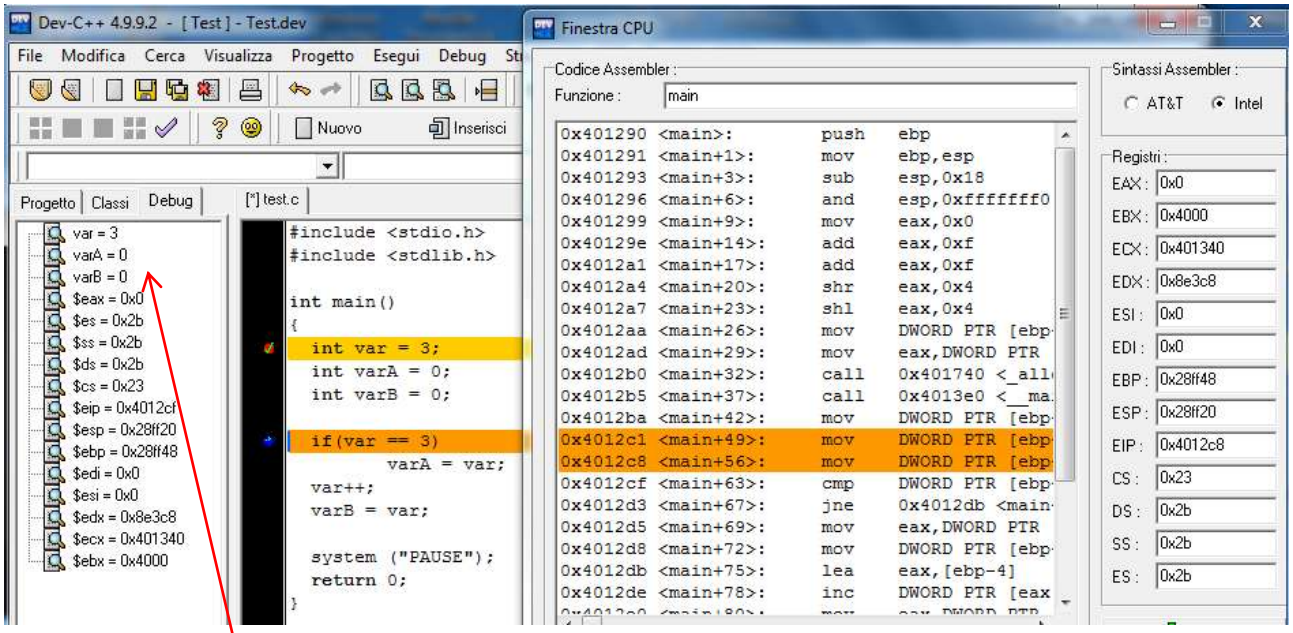


Terzo step

Ora **varA** contiene il valore 0 e vengono **visualizzate** le variazioni di contenuto degli **altri registri**



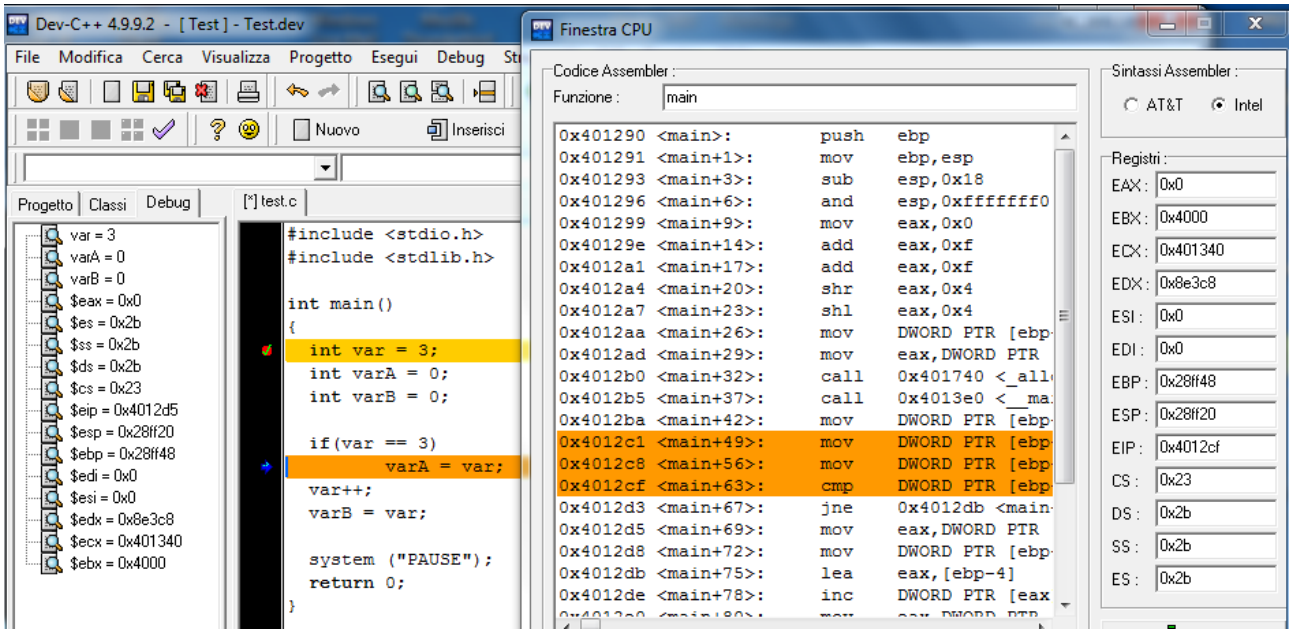
Quarto step (quarta e quinta istruzione)



Anche le variabili **varA** e **varB** vengono inizializzate

Quinto step

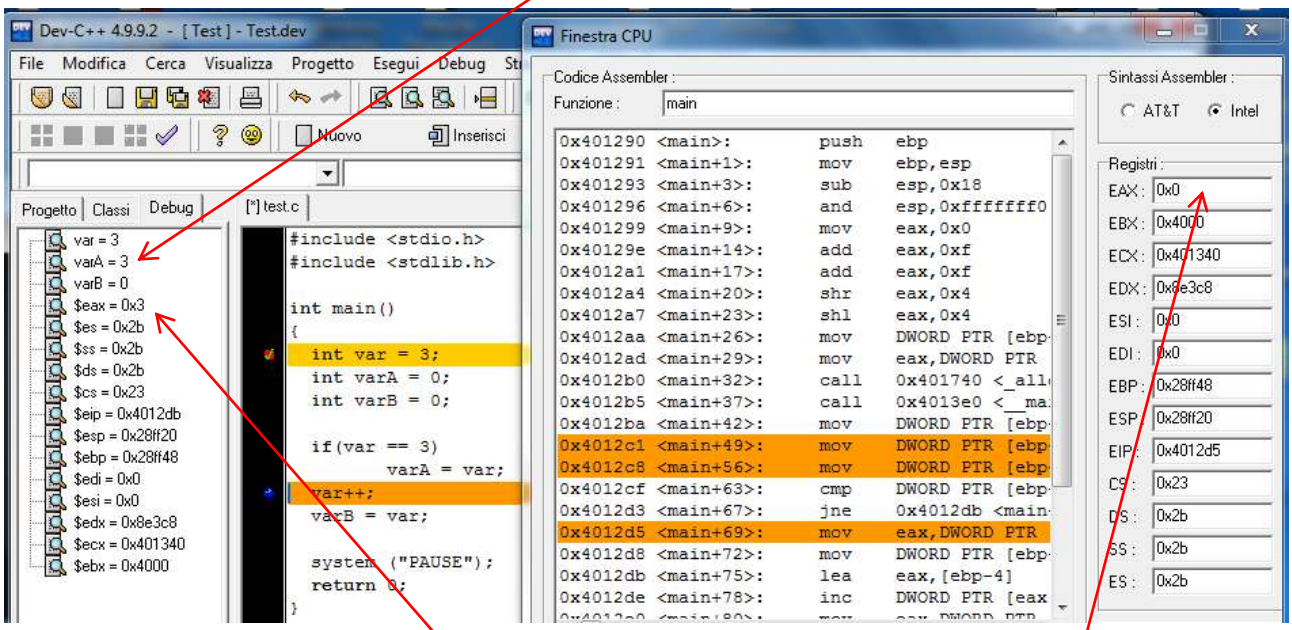
Confronto tra contenuto di **var** ed il valore 3



Essendo vera la condizione, non viene eseguito il salto ma si *punta* l'istruzione per copiare il valore di **var** in **varA**

Sesto step

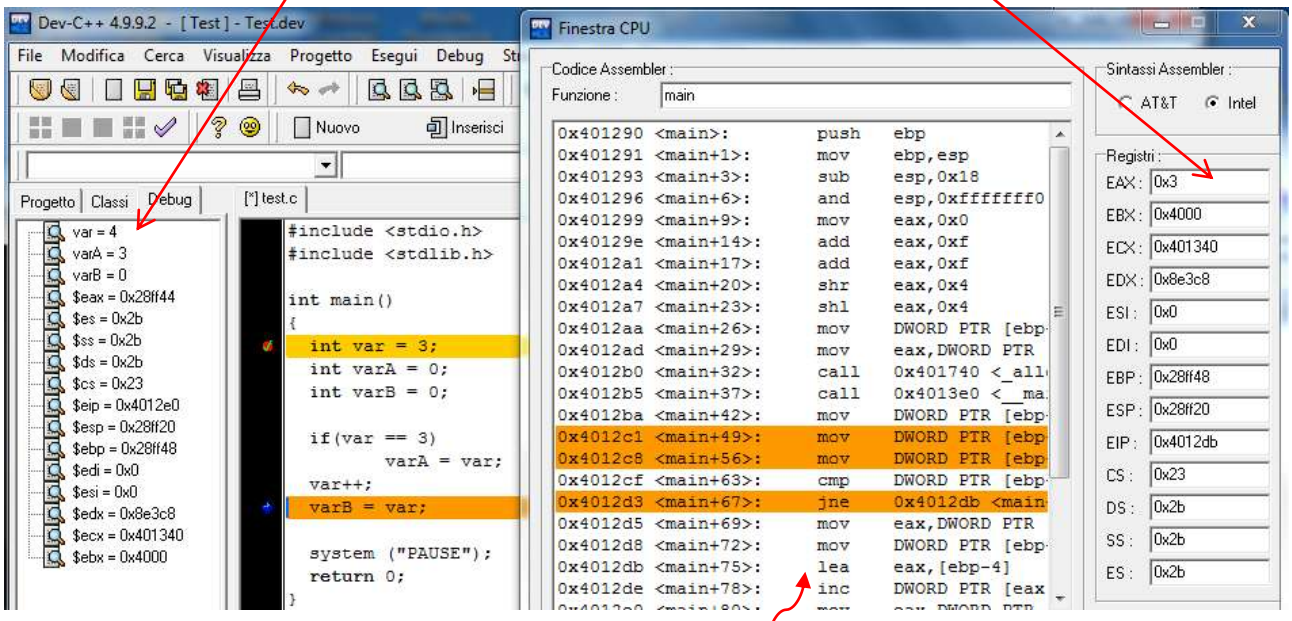
Ora **varA** contiene lo stesso valore di **var** cioè 3



Nb: nell'accumulatore è memorizzato tale valore(3) ma nella finestra CPU sarà visualizzato allo step successivo

Settimo step

Il contenuto di **var** è **incrementato** interessando il contenuto dell'accumulatore (valore prima dell'incremento)

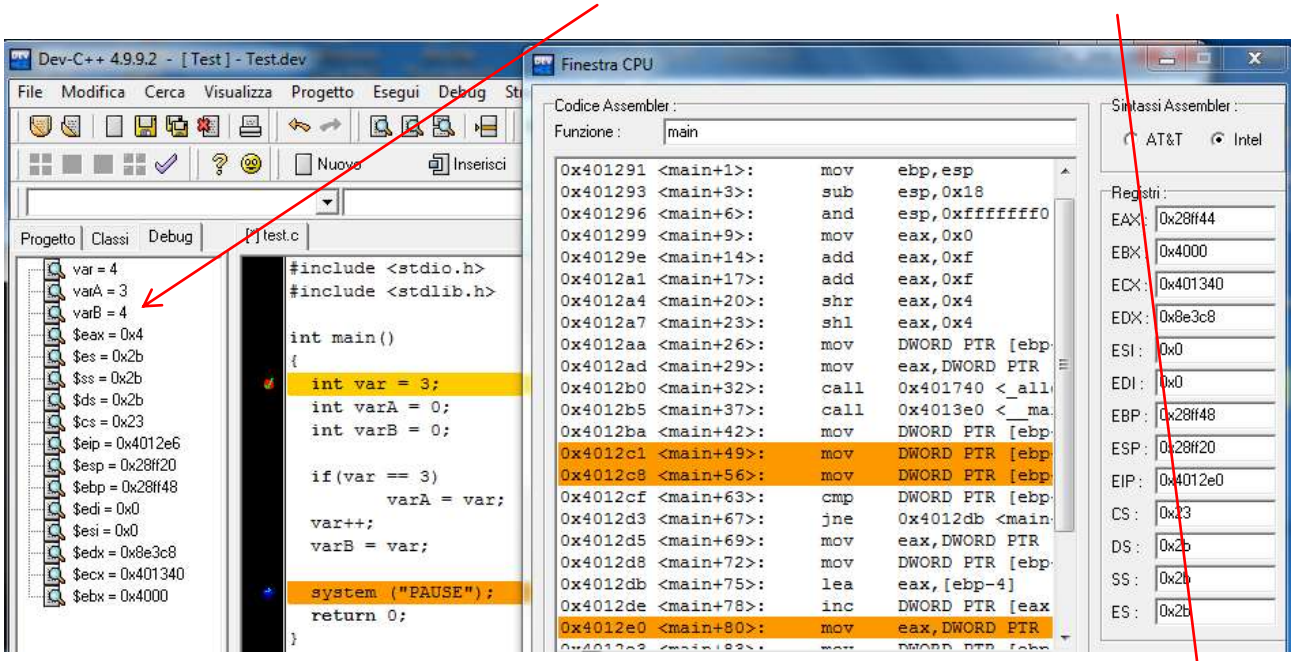


LEA Load Effective Address

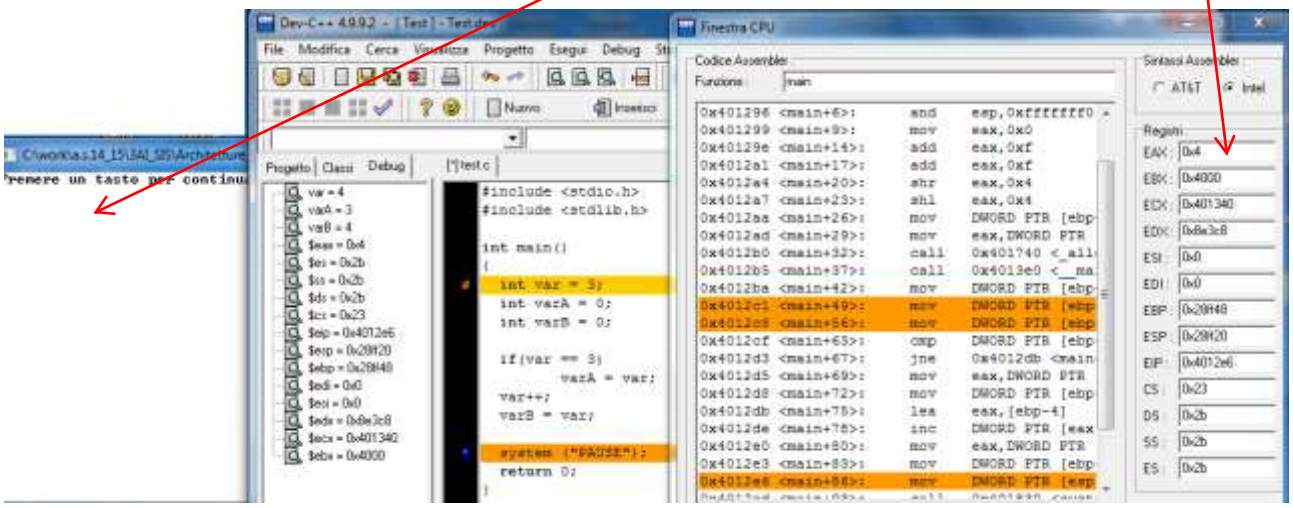
Nb: nell'accumulatore è memorizzato nuovo valore (indirizzo ebp-4) ma nella finestra CPU sarà visualizzato allo step successivo; registro BP (Base Pointer) punta ad un punto all'interno dello stack

Ottavo step

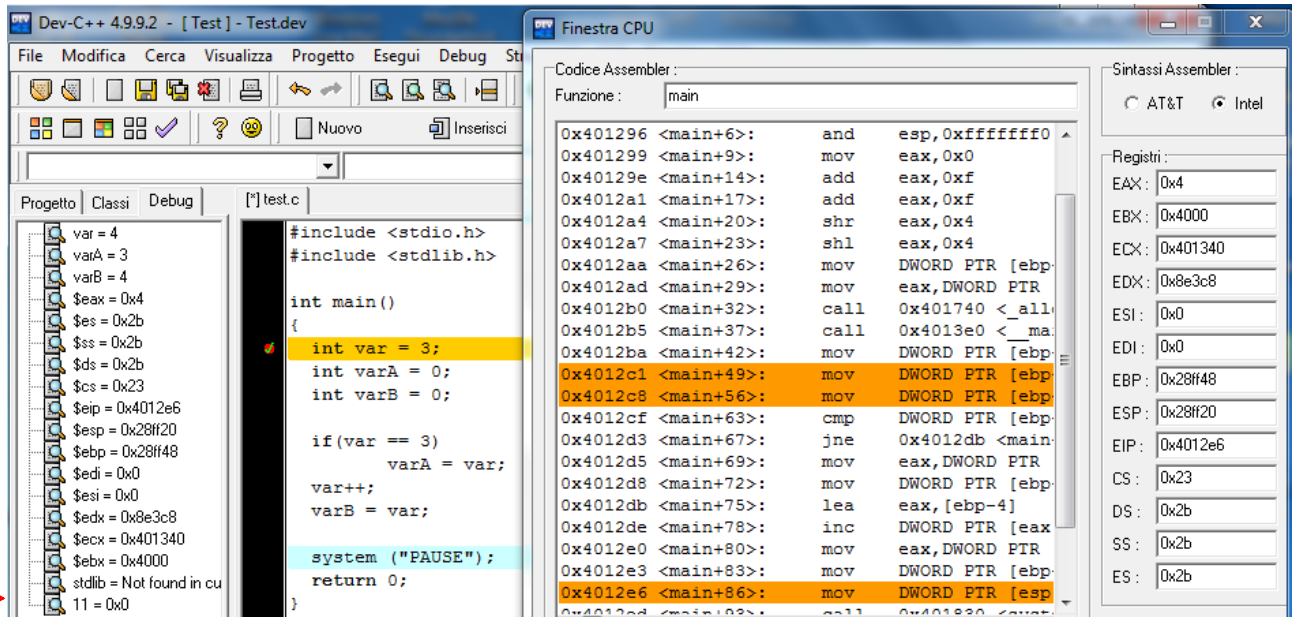
Ora la variabile **varB** contiene il valore di **var** cioè **4** influenzando il contenuto dell'accumulatore



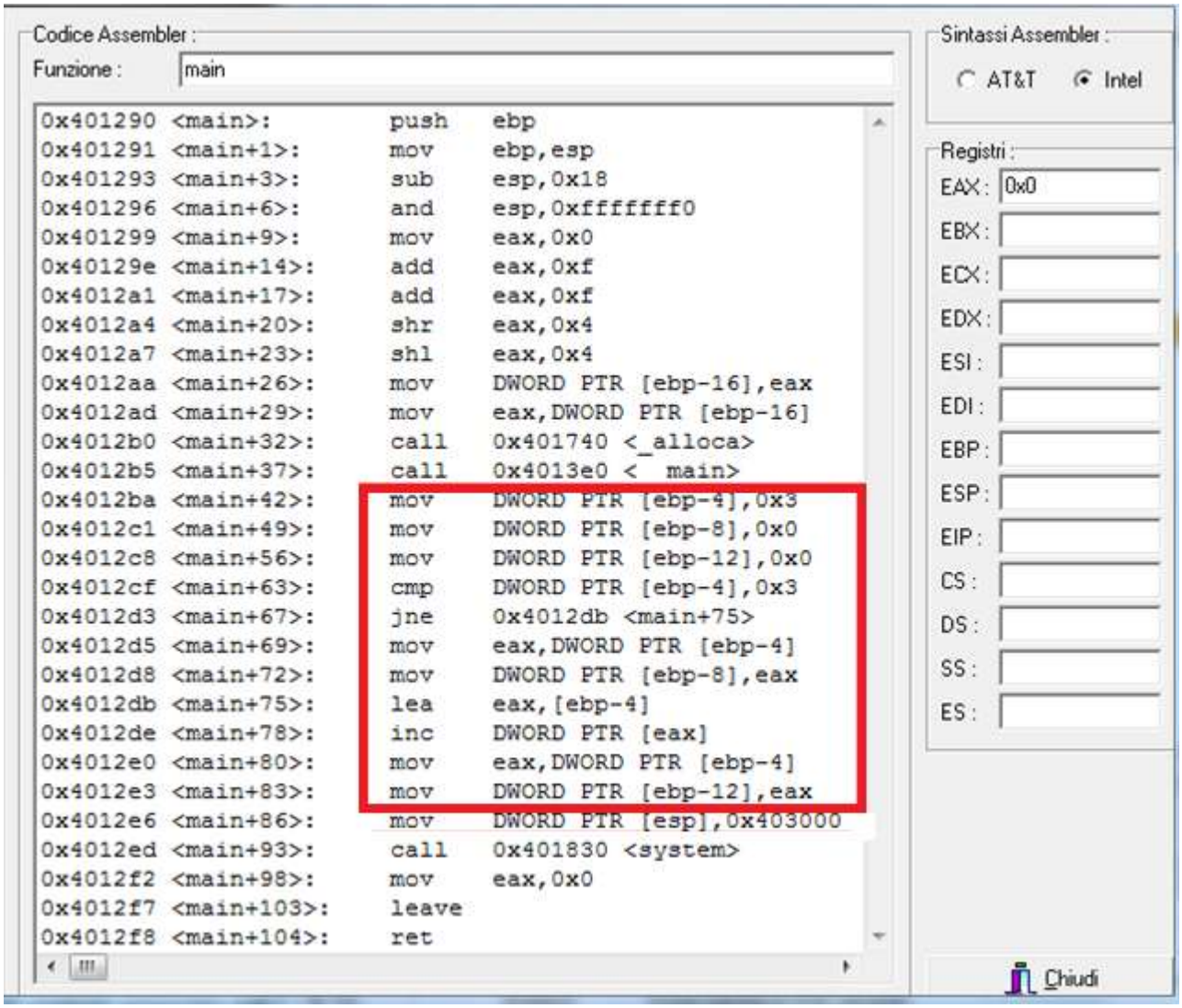
L'esecuzione della chiamata a SO apre la finestra console e attende la pressione di un tasto:



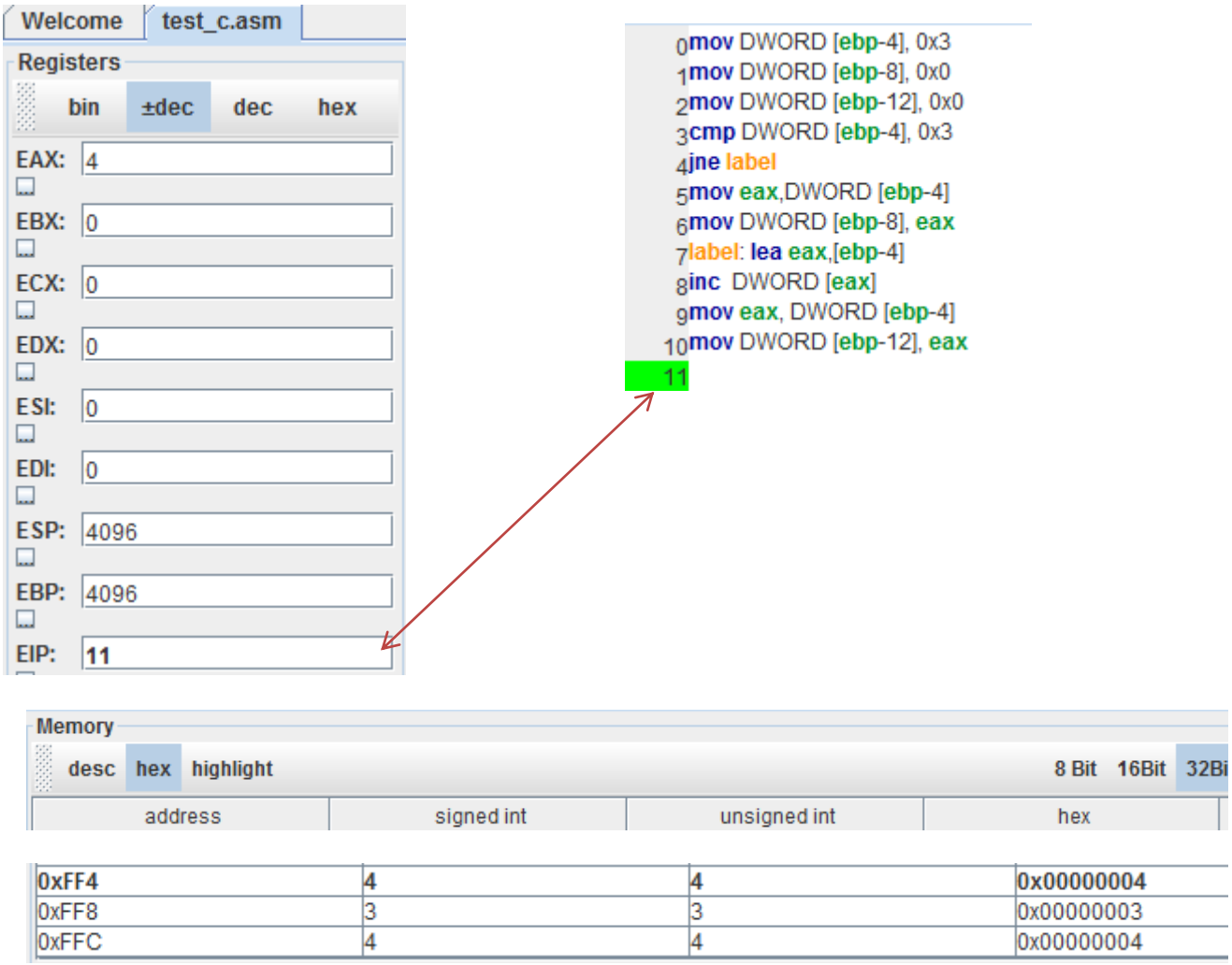
Dopo aver premuto un tasto nella finestra console, l'esecuzione termina (11 linee di codice) :



Codice Assembly - Intel



Simulando in JASMIN



EBP: ***il Rregistro BP (Base Pointer)*** punta ad un punto all'interno dello stack
4096 decimale corrisponde a 1000H ed EBP-4 cioè 4092 corrisponde a FFCH

Anche, allocando all'inizio della memoria

```
MOV [0x4], 03; assegnazione var = 3
MOV [0x8], 00; assegnazione varA = 0
MOV [0xC], 00; assegnazione varB = 0
CMP [0x4], 03; confronta con dato
JNZ esci
; istruzione del costrutto alternativa
MOV EAX, [0x4]
MOV [0x8], EAX ; assegna ad altra variabile varA il valore di var
esci:
; proseguo progrmma
MOV EBX, [0x4]
INC EBX
MOV [0x4], EBX ; incrementa var
MOV [0xC], EBX ; assegna ad altra variabile varB il valore di var incrementato
```

Nb: Senza poter emulare le chiamate a SO