Debug

Nous utilisons pour ce test un petit programme d'un seul module. Testons-le avec Turbo C++ Version 3.0

tstSomme.c

```
#include "stdio.h"
int a, b;
int somme( int n1, int n2)
{
  return n1 + n2;
}
void main()
{
 printf("Entrez le premier nombre: ");
  scanf("%d", &a);
  printf("Entrez un deuxieme nombre: ");
  scanf("%d", &b);
  a = somme(a, b);
 printf("la somme est %d", a);
  getch();
}
```

Demander la production de l'exécutable (Compile > Make)

L'environnement de développement est à la fois un éditeur et un debugger. Tout se passe comme si on était en présence d'un langage interprété. La seule différence est qu'il a fallu demander la construction de l'exécutable avant de pouvoir lancer la première instruction. Lançons l'exécution du programme pas à pas de manière à voir se dérouler les instructions une à une. (**R**un > **T**race into F7)

Les trois raccourcis intéressants sont **F7** pour avancer instruction par instruction et **ALT-F5** pour visualiser l'écran "utilisateur" autrement dit celui ou tourne l'application.



Agrandissez votre fenêtre pour pouvoir y afficher plus d'informations :

Option > Environment > Preference... > cochez l'option « Screen size » () 43/50 lines > [OK]

Une autre fonction intéressante de debug est d'afficher les données en cours d'exécution. Ouvrez pour ce faire la fenêtre de « Watch » Window > Watch

Testez la commande **Debug > Watches > Add** watch... ou son raccourci **Ctrl F7** pour saisir les noms des variables que vous voudriez voir évoluer.

Testez le programme pas à pas avec la touche F7 puis au moment d'entrer dans la fonction somme voyez l'effet de la commande **Ctrl F3** (= **D**ebug > **C**all stack...) Elle permet de retracer les appels de fonction en indiquant en outre les valeurs passées comme paramètres.

Debug avec turbo debugger

Le turbo debugger est un programme complémentaire lui aussi produit par Borland et qui permet le test de programmes conçus dans différents langages. Nous l'utiliserons pour des programme écrit s en C ou en assembleurs.

Le Turbo Debugger est un programme qui nécessite la présence de deux exécutables : TD.exe et RTM.exe. Il faut donc que ces fichiers se trouvent dans l'un des répertoires spécifié par la variable d'environnement PATH. La dernière condition pour que le debugger puisse se référer au code source est d'utiliser une version du linker pas trop ancienne. Nous utilisons la version 7 de TLink.exe au lieu de la version initiale de TC V3.0

Voici la commande en ligne à taper pour avoir en une seule commande la compilation et l'édition de lien :

TCC -v TstSomme.c

Lancer le debugger TD : TD TstSomme

Au départ, une seule fenêtre, dite fenêtre "CPU", affiche le segment de code de l'application. On y voit les instructions en assembleur et leurs codes en hexadécimal. Plus bas dans la même fenêtre figure une partie du segment de données tandis qu'à droite on peut voir les états des registres.

-[]]=CPU Pentium======		Đ=====1=[↑][↓]=,
cs:0000⊩BA8317	mov dx,1783	🔼 ax 0000 🛛 c=0
cs:0003 2E89168D02	mov cs:[DGROUP@],dx	bx 0000 z=0
cs:0008 B430	mov ah,30	cx 0000 s=0
cs:000A CD21	int 21	dx 0000 o=0
cs:000C 8B2E0200	mov bp,[0002]	si 0000 p=0
cs:0010 8B1E2C00	mov bx,[002C]	di 0000 a=0
cs:0014 8EDA	mov ds,dx	bp 0000 i=1
cs:0016 A39200	mov [0092],ax	sp 0080 d=0
cs:0019 8C069000	mov [0090],es	ds 150F
cs:001D 891E8C00	mov [008C],bx	es 150F
cs:0021 892EA800	mov [00A8],bp	ss 17DA
cs:0025 E88101	call 01A9	cs 151F
cs:0028_C43E8A00	les di,[008A]	🔽 ip 0000
ds:0000 CD 20 FB 9F 00) 9A FØ FE = ¹f Ü-∎	
ds:0008 1D F0 35 0B F9	11 OF 07 ↔ 58 🐄	9
ds:0010 6F 0F 56 01 09	04 52 0F o*U⊕0◆R*	ss:0082_0401
ds:0018 01 01 01 00 02	FF FF FF 😔 🖯	ss:0080►52FB

Pas facile évidemment de s'y retrouver dans ce charabia !

Il est heureusement possible de faire apparaître le code source, pressez la touche F3 Demandons l'affichage du module source F3 (ou ALT-View > Module ...) puis sélectionner le module TSTSOMME

Une seconde fenêtre affiche le code source en langage C. Redimensionner cette fenêtre et placer la de sorte à voir simultanément le code source en C et la fenêtre « CPU »

📾 80x60 - td tstsomme		_ 8 ×
- File Edit View Run Breakpoints Data Opt	ions Window Help	READY
CPU Pentium _somme: int somme(int n1, int n2) cs:0291 55 push bp cs:0292 8BEC mov bp,sp #TSTSOMME#7: return n1 + n2; cs:0297 034606 cs:0297 034606 add ax, [bp+04] cs:0297 034606 jmp #TSTSOMME#8 (029 #TSTSOMME#8: > cs:0290 EB00 cs:0290 C3 ret _main: void main() cs:029F 8BEC cs:029F 8BEC mov bp,sp	1 ax 0000 c=0 bx 0000 z=0 cx 0000 s=0 dx 0000 o=0 si 0000 p=0 di 0000 a=0 bp 0000 i=1 sp 0080 d=0 ds 150F ss ss 17DA cs cs 151F ip ip 0000 i	
<pre>ds:0008 1D F0 35 0B F9 11 0F 07 +-56" 4x* ds:0010 6F 0F 56 01 09 04 52 0F oxUUDO*R* ds:0018 01 01 01 00 02 FF FF FF DOGE 0 []=Module: TSTSOMME File: TSTSOMME.C 1 #include "stdio.h" int a, b; int somme(int n1, int n2) { return n1 + n2; } void main() { printf("Entrez le premier nombre: "); scanf("%d", &a); printf("Entrez un deuxieme nombre: "); scanf("%d", &b); a = somme(a, b); printf("la somme est %d", a); getch(); } </pre>	ss:0082 0401 ss:0080►52FB 2=[1][U]	

Si on lance le programme pas à pas [F7] on s'aperçoit qu'il faut beaucoup d'instructions avant d'entrer dans le module main(). Plaçons le curseur dans le module source sur la ligne main() et demander l'exécution jusqu'à cet endroit (RUN > Go to cursor F4)

A chaque fois que l'on presse la touche F7 on avance maintenant d'une ligne en code C. Allons ainsi jusqu'au début de la fonction somme().

Voyons maintenant en détail et en assembleur comment se passe la fonction somme. Cliquez pour cela dans la fenêtre qui affiche le code machine de cette fonction. La touche F7 fera avancer instruction par instruction.

Remarquer comment les paramètres sont passés sur la pile. Le pointeur BP sert à remonter dans la pile pour y reprendre les arguments passés à la fonction. En fin de fonction le résultat, un entier, est simplement retourné par l'accumulateur AX. Remarquer aussi l'instruction en CS:203, ce saut est inutile, c'est le résultat d'une compilation qui n'est pas optimisée.