**Utilizare Debugger**

```
#include<iostream>
using namespace std;
int main(void){
004113C0  push        ebp
004113C1  mov         ebp,esp
004113C3  sub         esp,0E4h
004113C9  push        ebx
004113CA  push        esi
004113CB  push        edi
004113CC  lea         edi,[ebp-0E4h]
004113D2  mov         ecx,39h
004113D7  mov         eax,0CCCCCCCCh
004113DC  rep stos    dword ptr es:[edi]
      int a,b,c;
      a=64;
004113DE  mov         dword ptr [a],40h
      b=65;
004113E5  mov         dword ptr [b],41h
      c=a+b;
004113EC  mov         eax,dword ptr [a]
004113EF  add         eax,dword ptr [b]
004113F2  mov         dword ptr [c],eax
      cout<<c<<endl;
004113F5  mov         esi,esp
004113F7  mov         eax,dword ptr [__imp_std::endl (418298h)]
004113FC  push        eax
004113FD  mov         edi,esp
004113FF  mov         ecx,dword ptr [c]
00411402  push        ecx
00411403  mov         ecx,dword ptr [__imp_std::cout (418290h)]
00411409  call        dword ptr
[__imp_std::basic_ostream<char,std::char_traits<char> >::operator<<
(418294h)]
0041140F  cmp         edi,esp
00411411  call        @ILT+325(__RTC_CheckEsp) (41114Ah)
00411416  mov         ecx,eax
00411418  call        dword ptr
[__imp_std::basic_ostream<char,std::char_traits<char> >::operator<<
(41829Ch)]
0041141E  cmp         esi,esp
00411420  call        @ILT+325(__RTC_CheckEsp) (41114Ah)
      return 0;
00411425  xor         eax,eax
}
00411427  pop         edi
00411428  pop         esi
00411429  pop         ebx
0041142A  add         esp,0E4h
00411430  cmp         ebp,esp
00411432  call        @ILT+325(__RTC_CheckEsp) (41114Ah)
00411437  mov         esp,ebp
00411439  pop         ebp
0041143A  ret
```

Programul de depanare, debagger-ul, ne prezintă traducerea în limbaj de asamblare a codului sursă C++ (scris aici cu **bold**). Fiecare linie începe cu adresa (pe 4 octeți, adică 8 cifre hexa) a locației de memorie unde este depus codul maşină (cod înțeles de microprocesor) al respectivei instrucțiuni. După adresă urmează mnemonica (push, mov, add, etc.) şi argumentele instrucțiunii.

In exemplul dat aici codul maşină al funcției **main** este depus în memorie între adresele 004113C0 şi 0041143A. Inainte de a-l depune în memorie, debagger-ul setează toți octeții zonei rezervate codului programului cu valoarea CC, aşa observăm mai uşor codul nostru:

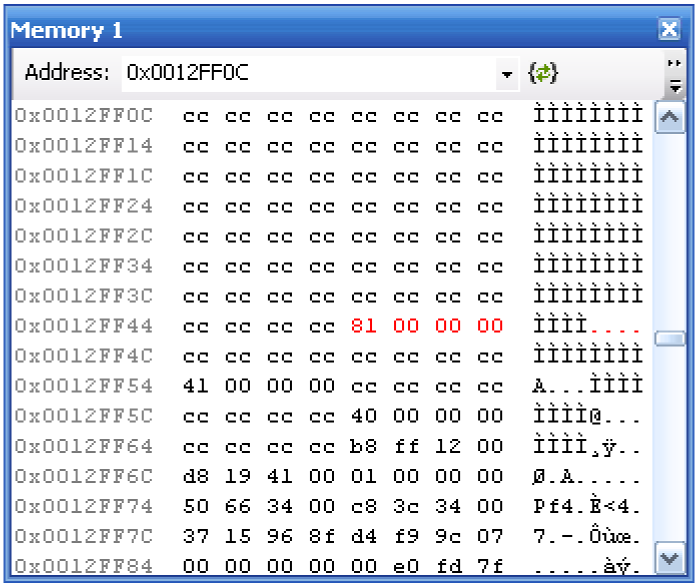
```
Memory 1
Address: 0x004113A8
0x004113A8  cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌ
0x004113B0  cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌ
0x004113B8  cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌ
0x004113C0  55 8b ec 81 ec e4 00 00  U.ì.ìä..
0x004113C8  00 53 56 57 8d bd 1c ff  .SVW...ÿ
0x004113D0  ff ff b9 39 00 00 00 b8  ÿÿ.9...¸
0x004113D8  cc cc cc cc f3 ab c7 45  ÌÌÌÌó«ÇE
0x004113E0  f8 40 00 00 00 c7 45 ec  ø@...ÇEì
0x004113E8  41 00 00 00 8b 45 f8 03  A....Eø.
0x004113F0  45 ec 89 45 e0 8b f4 a1  Eì.Eà.ô¡
0x004113F8  98 82 41 00 50 8b fc 8b  ˜.A.P.ü.
0x00411400  4d e0 51 8b 0d 90 82 41  MàQ....A
0x00411408  00 ff 15 94 82 41 00 3b  .ÿ.”.A.;
0x00411410  fc e8 34 fd ff ff 8b c8  üè4ýÿÿ.È
0x00411418  ff 15 9c 82 41 00 3b f4  ÿ.œ.A.;ô
0x00411420  e8 25 fd ff ff 33 c0 5f  è%ýÿÿ3À_
0x00411428  5e 5b 81 c4 e4 00 00 00  ^[.Ää...
0x00411430  3b ec e8 13 fd ff ff 8b  ;ìè.ýÿÿ.
0x00411438  e5 5d c3 cc cc cc cc cc  å]ÃÌÌÌÌÌ
0x00411440  cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌ
```

Iată şi starea regiştrilor microprocesorului şi a zonei de memorie alocată stivei de execuție a programului după execuția instrucțiunii C++

**c=a+b;**

chiar înainte de a fi executată instrucțiunea microprocesor de la adresa 004113F5 (vezi valoarea registrului EIP - Extended Instruction Pointer - ).

```
Registers
EAX = 00000081 EBX = 7FFDE000
ECX = 00000000 EDX = 00000001
ESI = 00000000 EDI = 0012FF68
EIP = 004113F5 ESP = 0012FE78
EBP = 0012FF68 EFL = 00000206
```




Octeții în roşu sunt cei patru octeți alocați variabilei **c**.