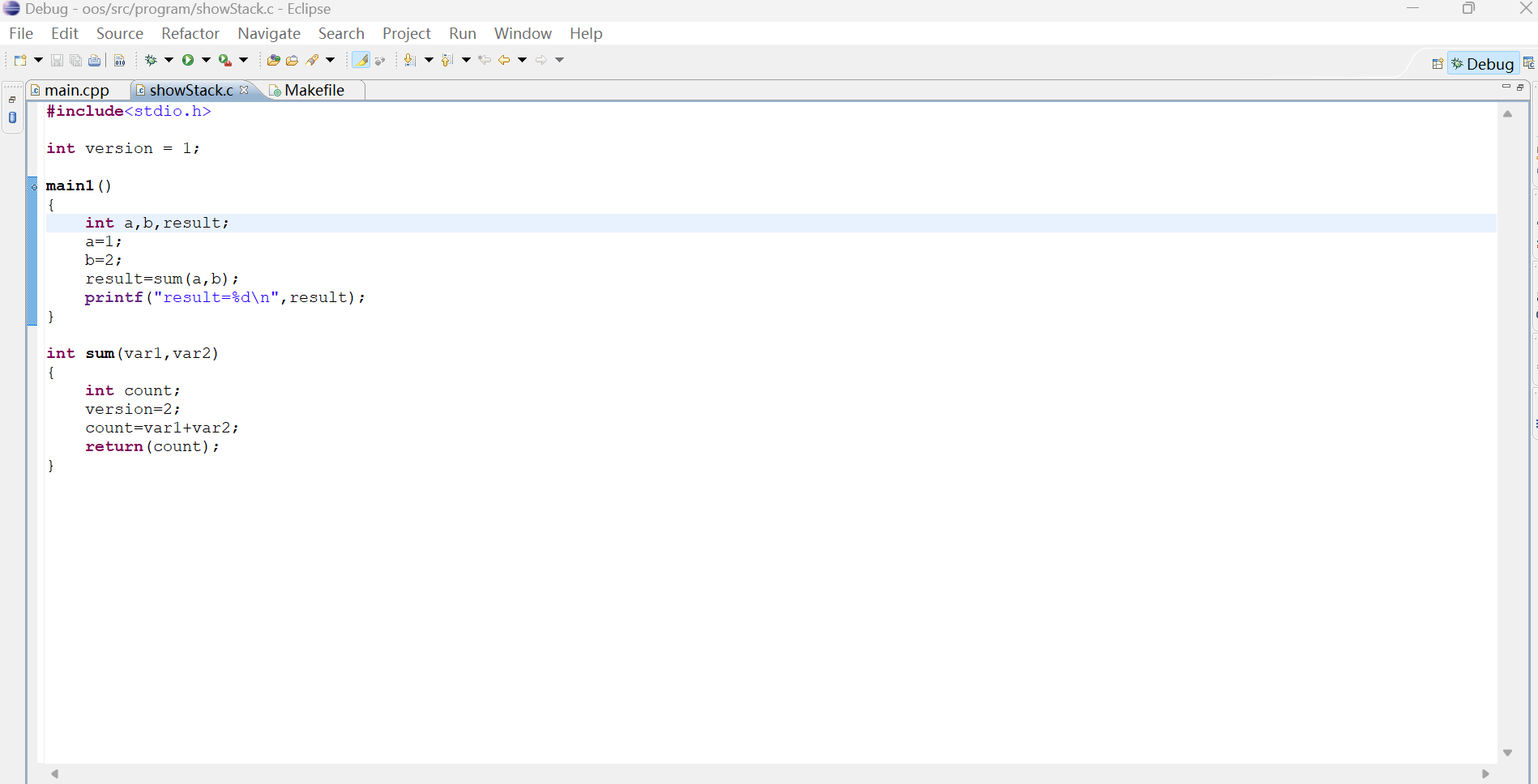
**实验报告：UNIX V6++进程的栈帧**

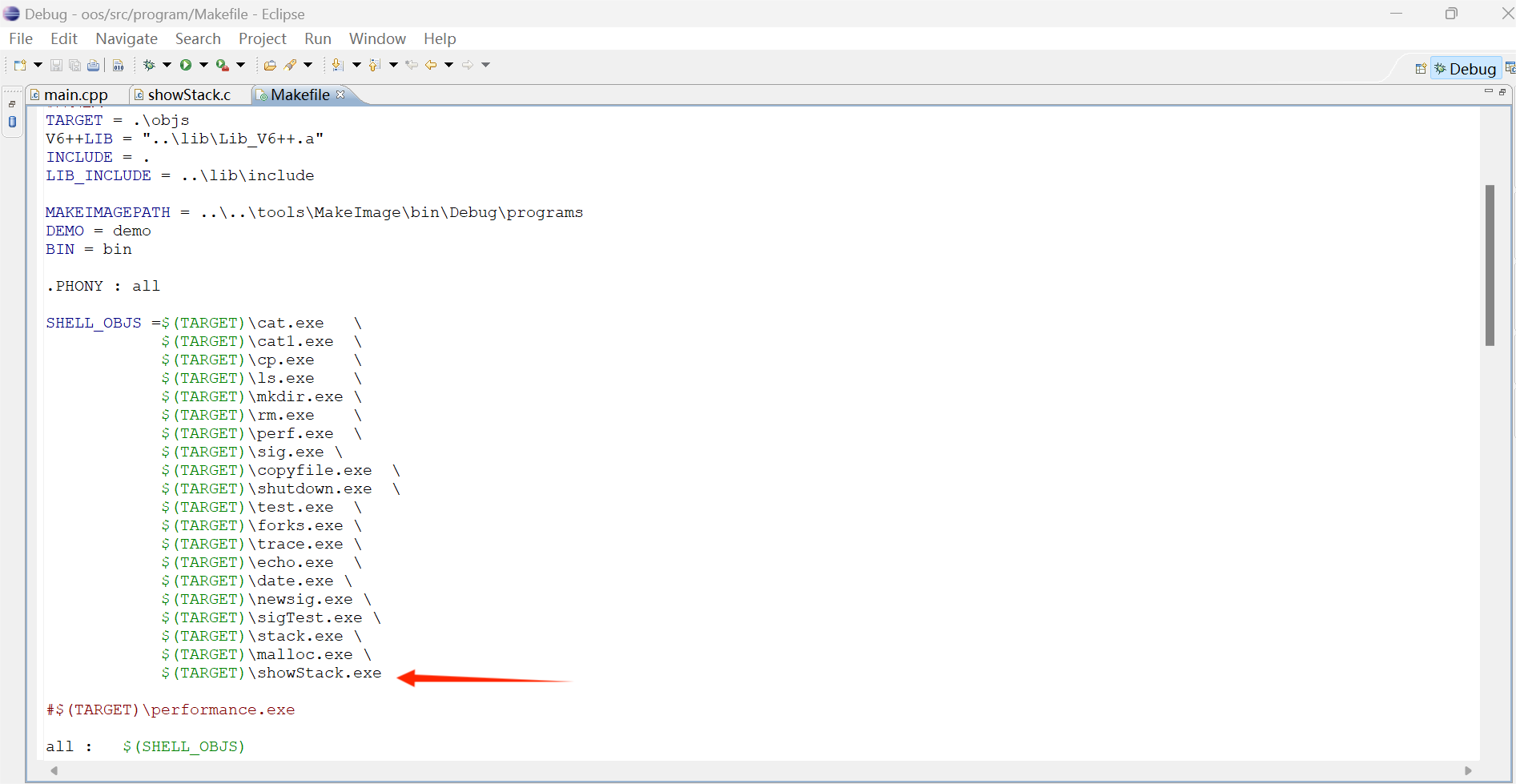
一、完成实验 4.1~4.2，掌握在 UNIX V6++中添加自定义程序及编译、链接与运行的全过程，掌握 UNIX V6++中调试运行与观察结果的常规操作

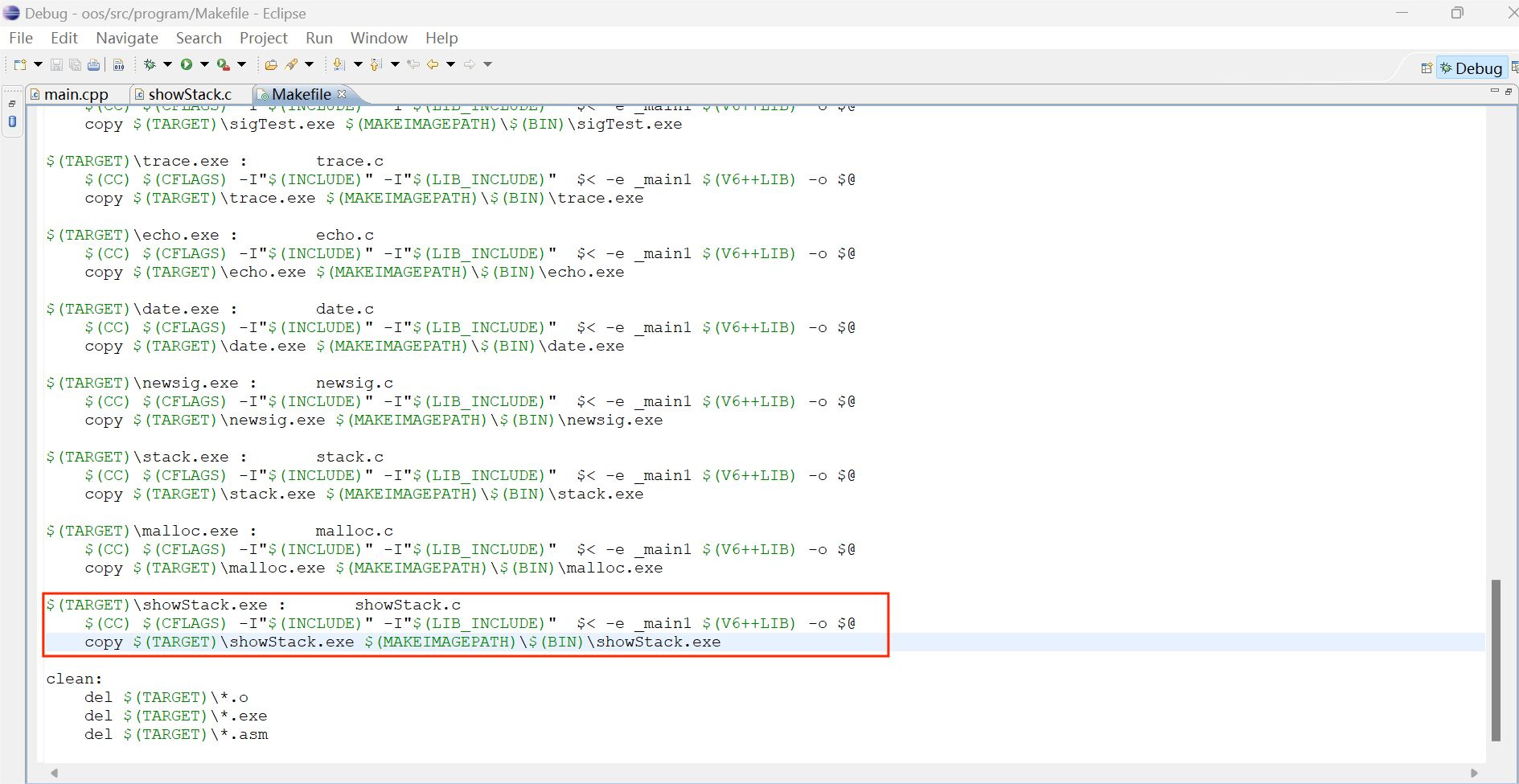
1.1在 UNIX V6++中编译链接运行一个 C 语言程序

1.1.1在 program 目录中加入一个新的文件showStack.c

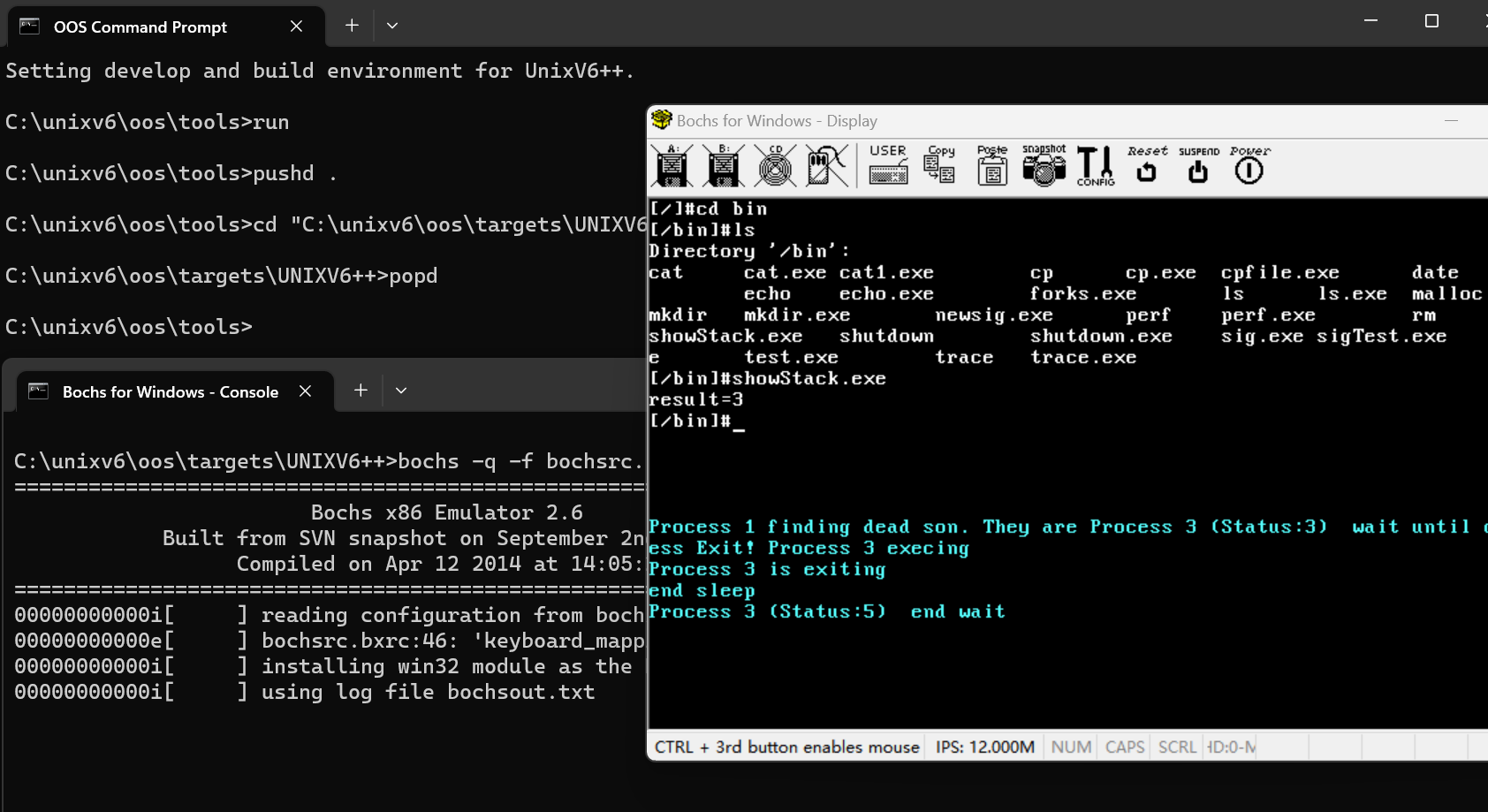


1.1.2修改编译需要使用的 Makefile 文件



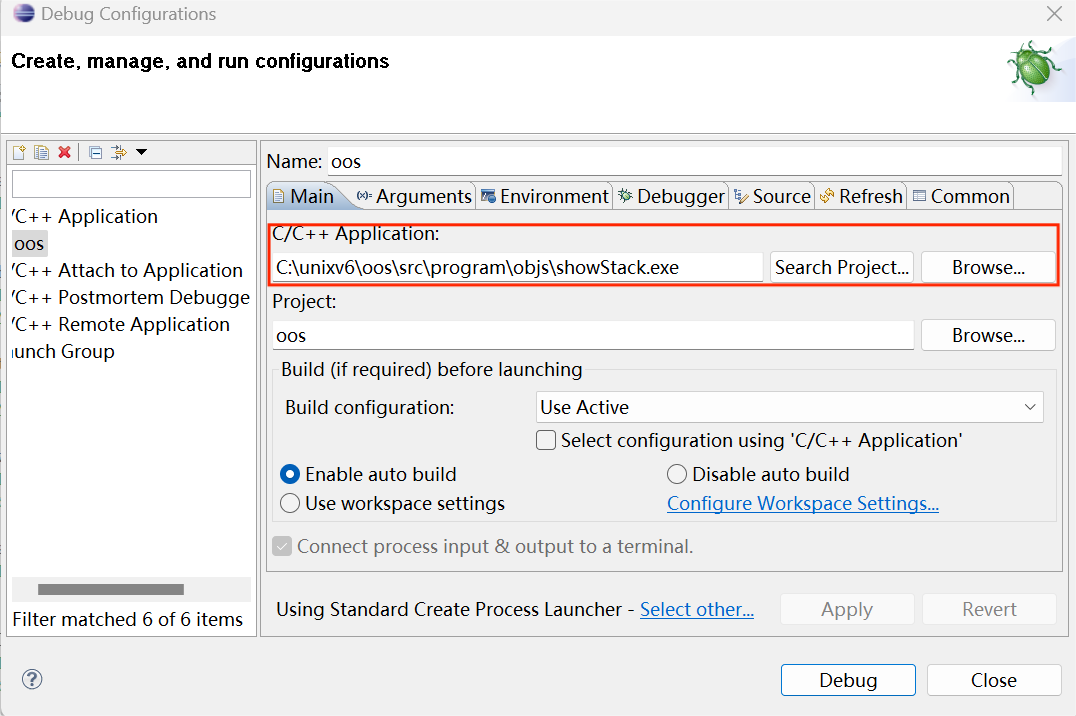


1.1.3重新编译运行 UNIX V6++代码；编译成功后在运行模式下启动UNIX V6++，进入 bin 目录输入命令 showStack.exe，可以看到程序的运行结果“result=3”

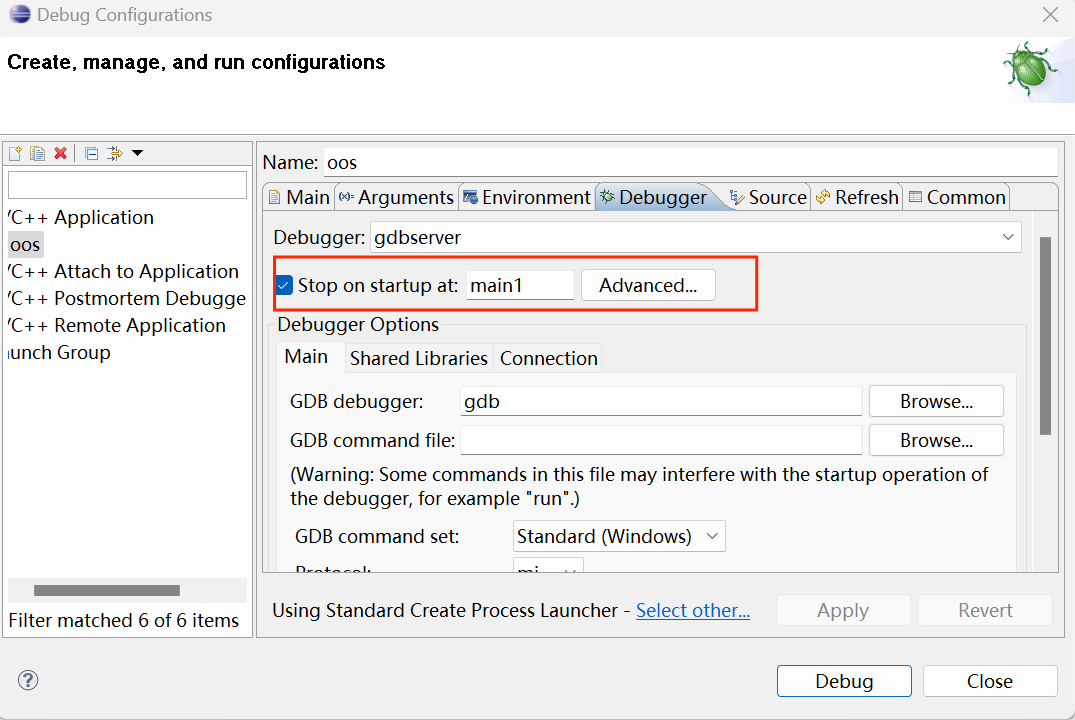


1.2完成程序的调试运行

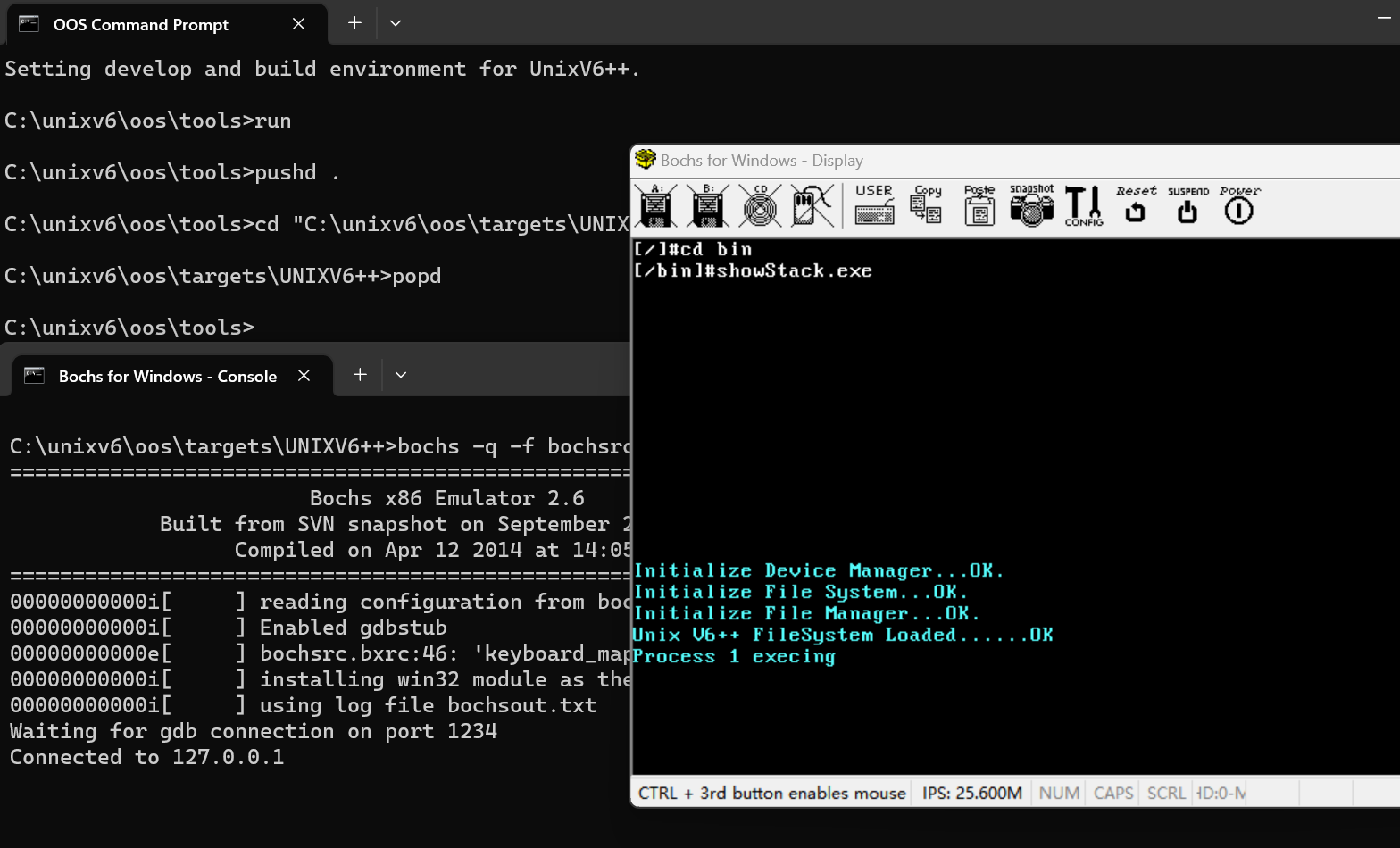
1.2.1将调试对象设置为 showStack.exe 程序



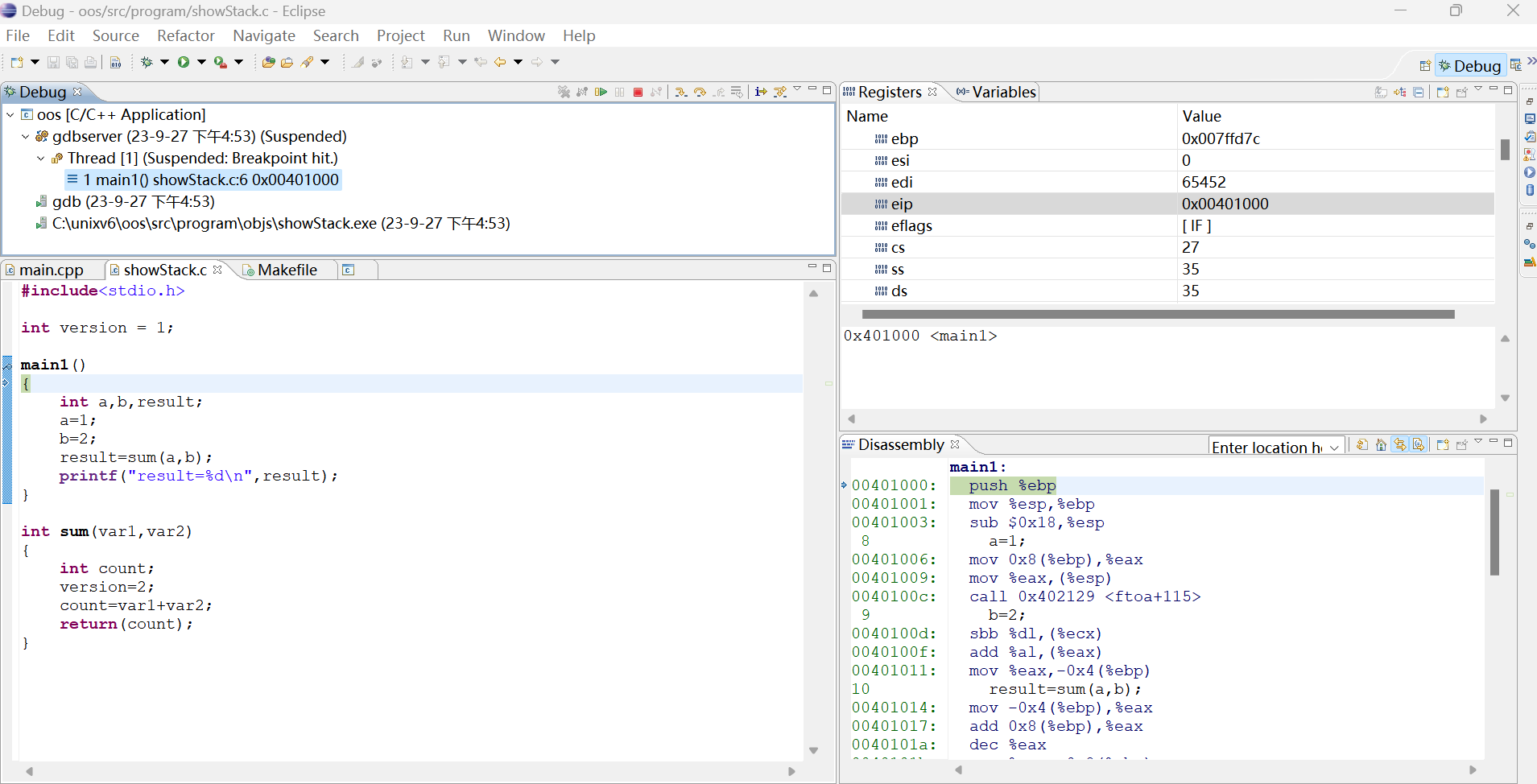
1.2.2调试起点设置为该程序的入口 main1



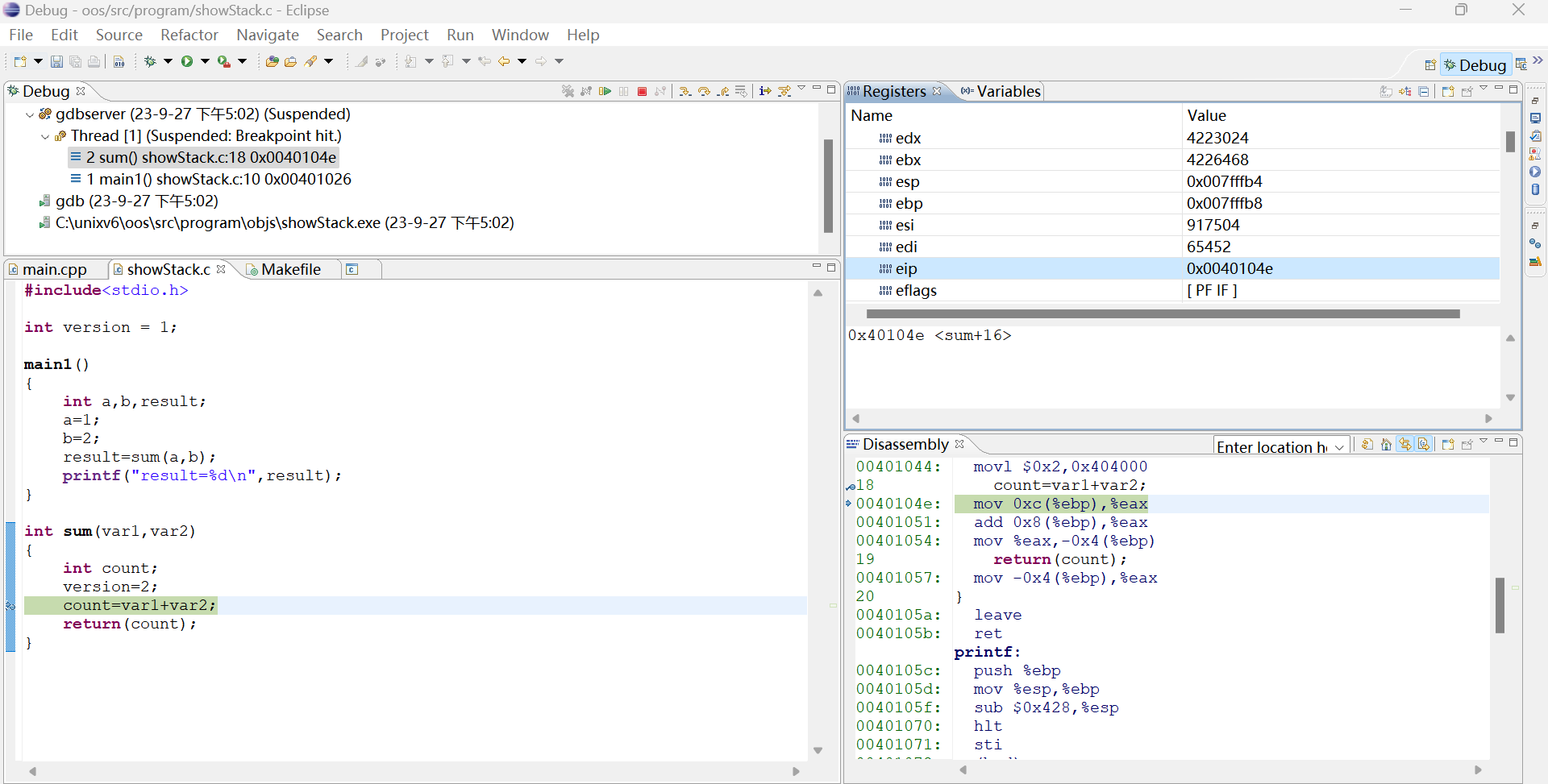
1.2.3以调试模式启动 UNIX V6++，执行命令“cd bin”和“showStack.exe”



1.2.4程序停在main1 处，此时eip寄存器的值指向该位置

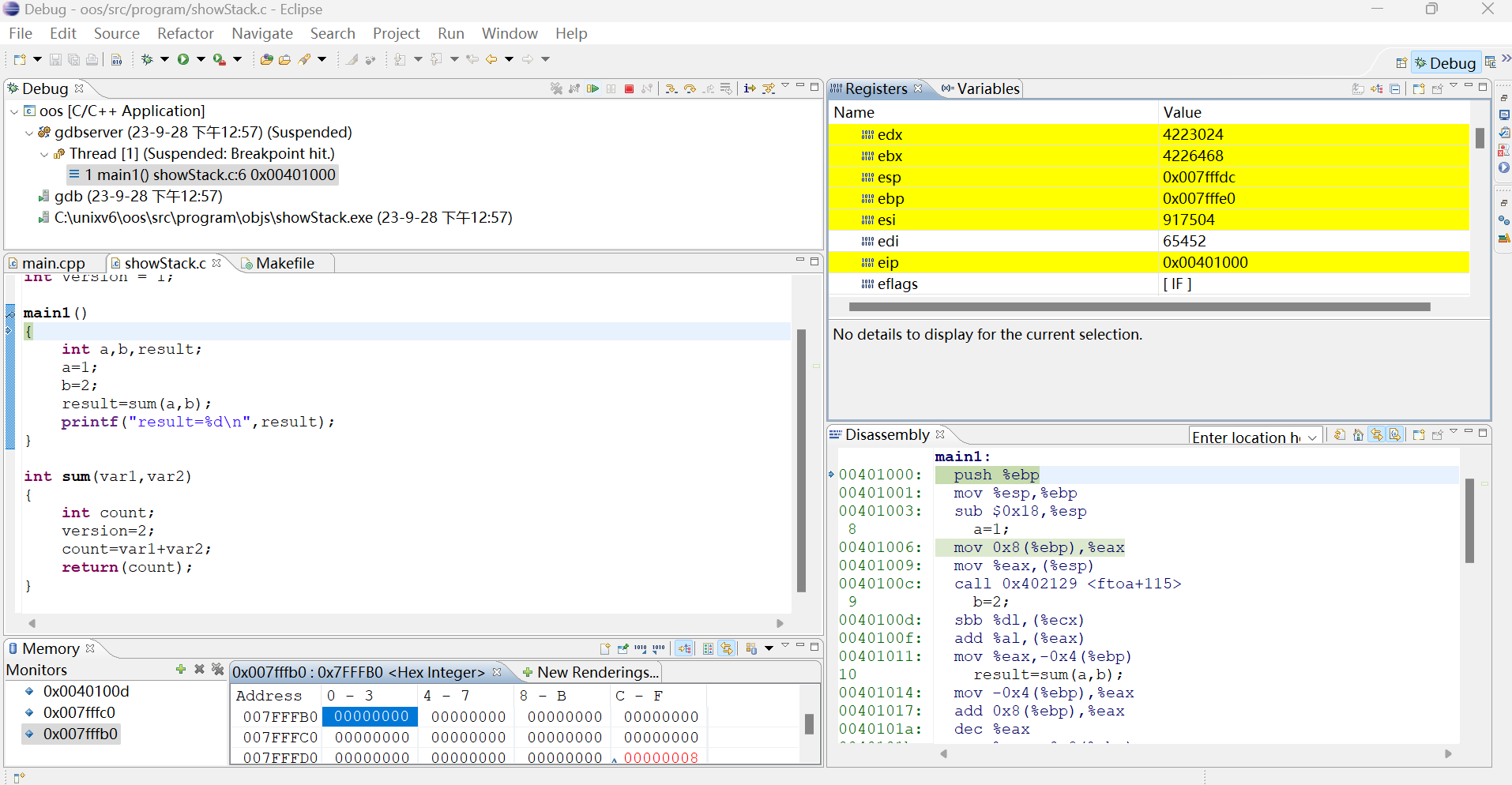


1.2.5设置断点再次调试，程序停在断点处，可以查看对应的汇编指令以及寄存器内的值

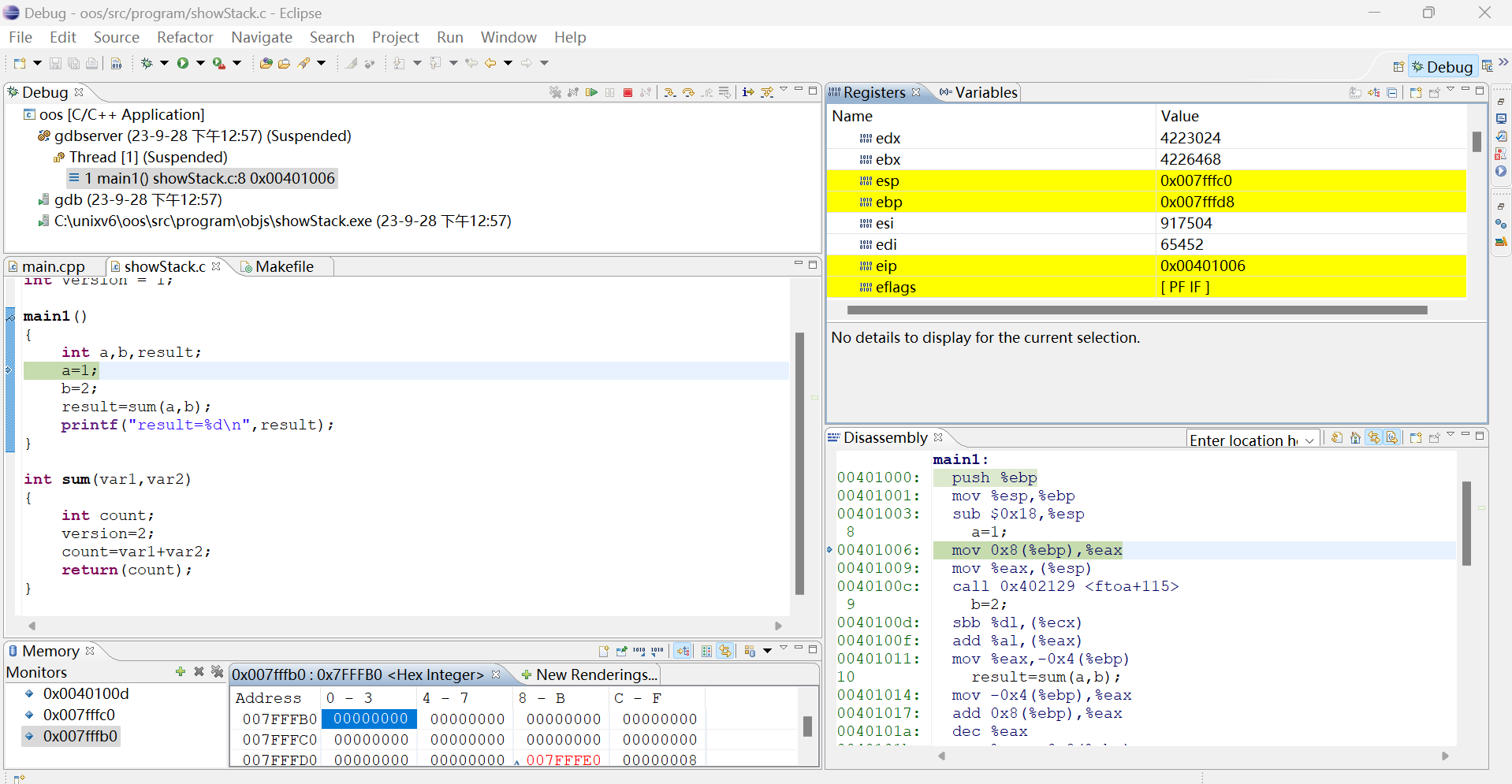


二、复现实验 4.3 中 main1 函数核心栈的变化，通过在 Memory 窗口中观察地址单元的值验证核心栈的变化

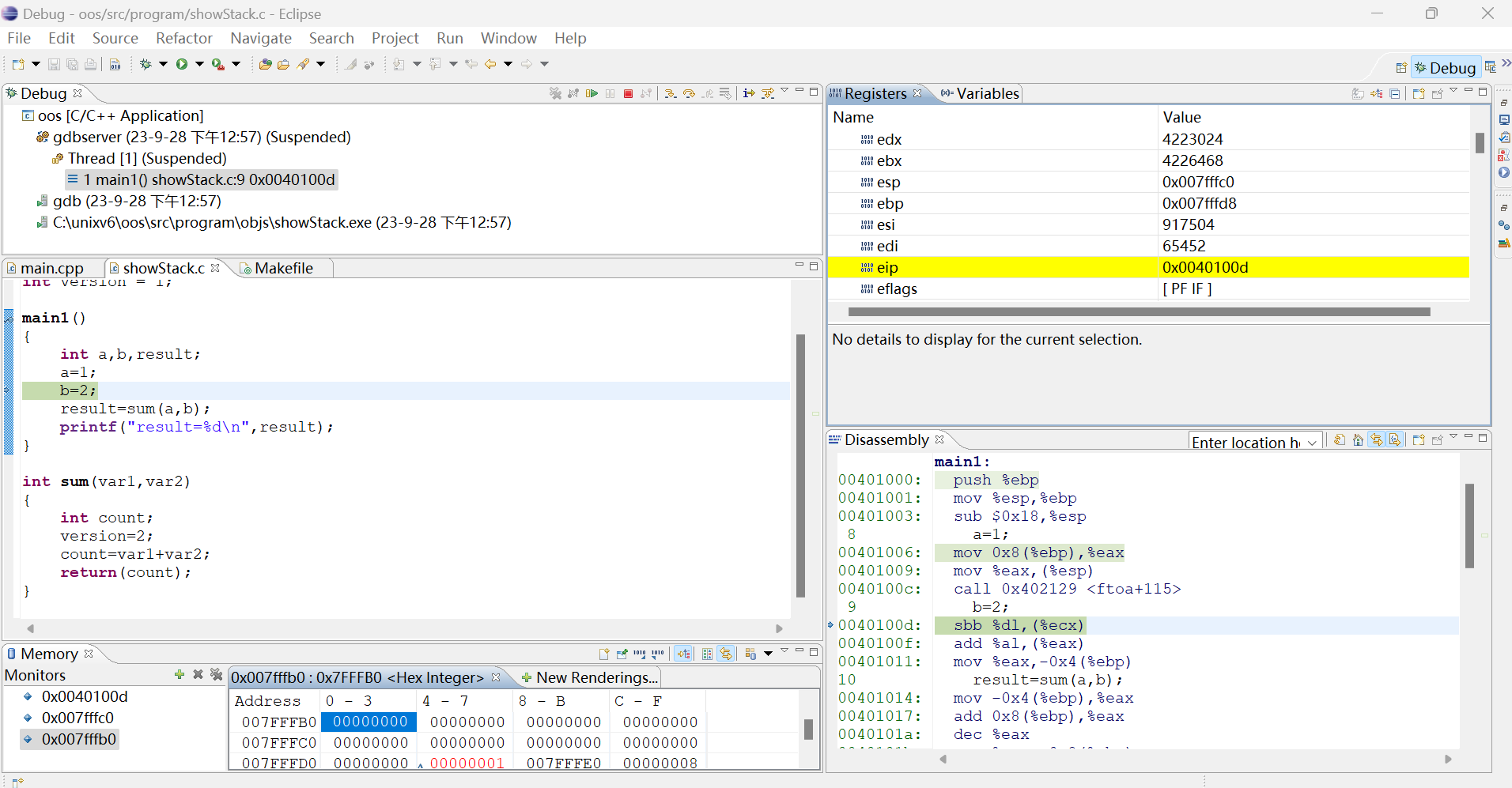
2.1前一栈帧的 ebp 存入当前栈，007fffdc存入main的返回地址00000008

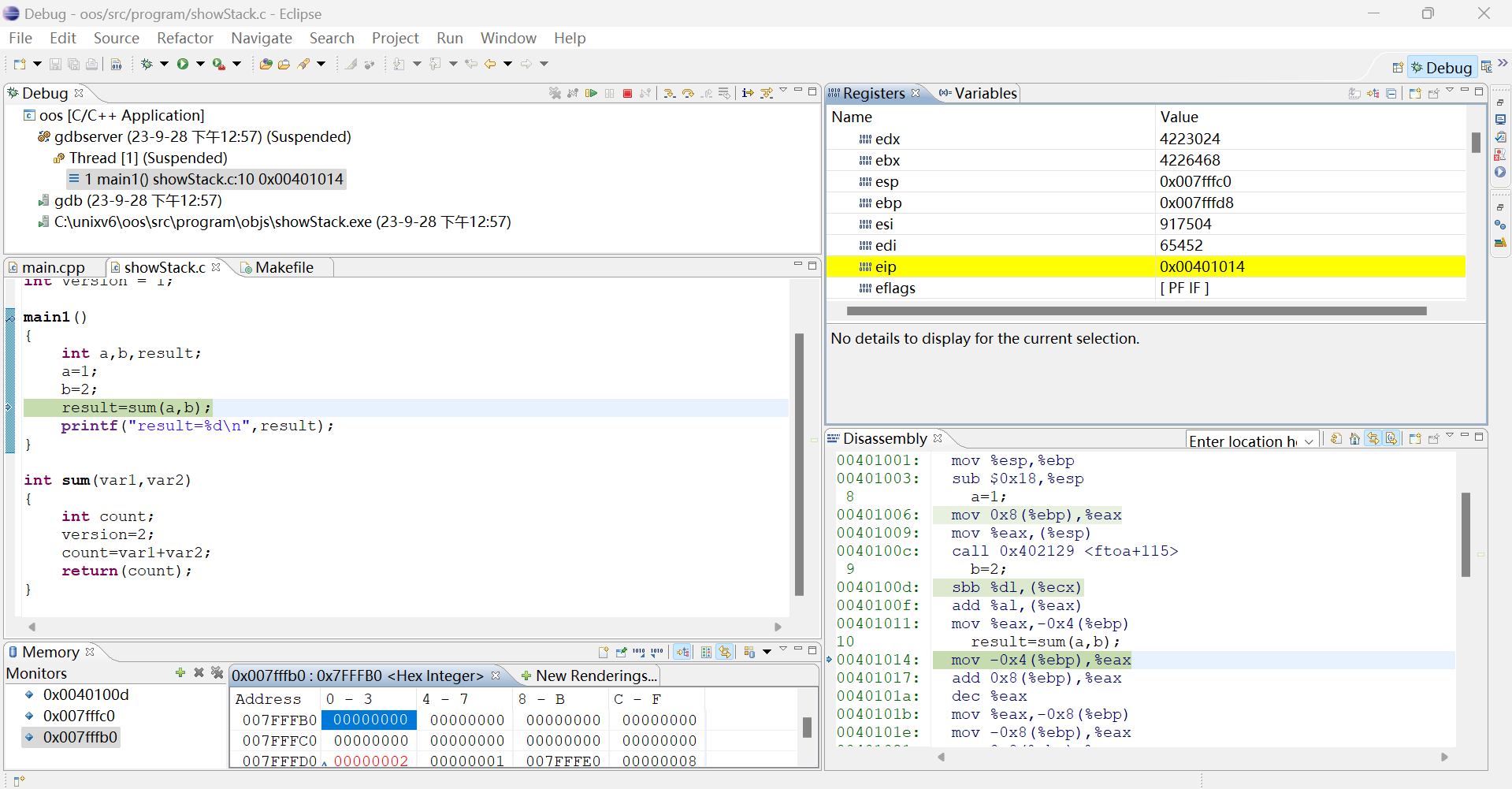


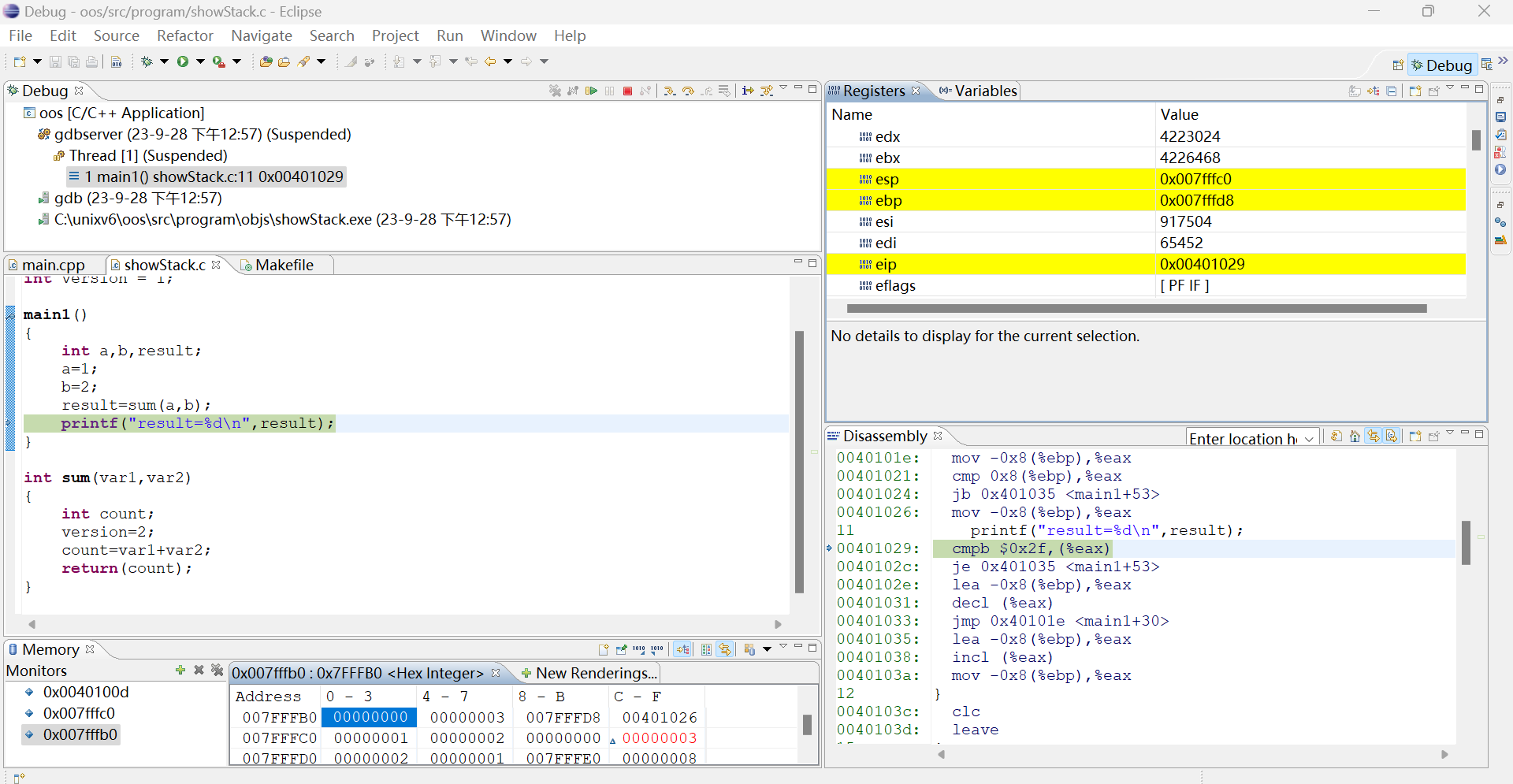
2.2修改ebp指向007fffd8，斌且esp上移指向007fffc0，即空出24个字节



2.3局部变量a=1，存入地址007fffd4

2.4局部变量b=2，存入地址007fffd0

2.5 sum函数运行完之后，局部变量result=3，存入地址007fffcc



三、完整分析 sum 的汇编代码，添加注释，并绘制完整的堆栈

3.1分析sum函数的汇编代码

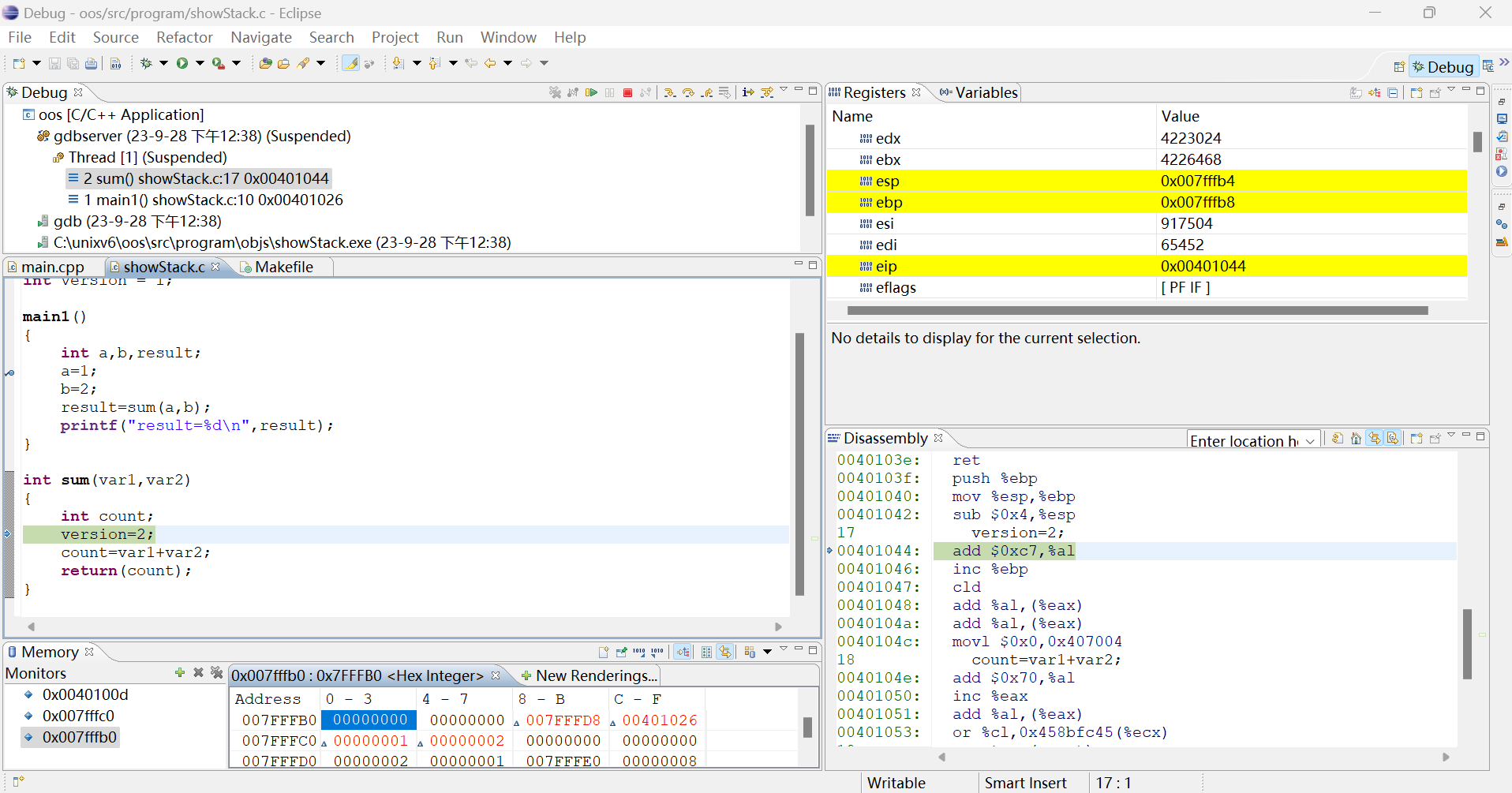
|  |
| --- |
| sum:  0040103e: push %ebp //前一栈帧的 ebp 存入当前栈  0040103f: mov %esp,%ebp //修改 ebp 指向当前栈帧(0x007fffb8)  00401041: sub $0x4, %esp //esp 上移 1 个字，空出局部变量count的位置  17 version=2;  00401044: movl $0x2,0x404000 //将立即数2移动到内存地址0x404000处  18 count=var1+var2;  0040104e: mov 0xc(%ebp), %eax //将0x007fffc4中的数据移动到寄存器 eax 中  00401051: add 0x8(%ebp), %eax //将寄存器 eax 中的值与0x007fffc0的数据相加，结果存到eax中  00401054: mov %eax,-0x4(%ebp) //将eax 的值移到0x007fffb4，即局部变量count  19 return(count);  00401057: mov -0x4(%ebp), %eax //将局部变量count的值送入eax  0040105a: leave  0040105b: ret |

3.2 sum函数堆栈示意图

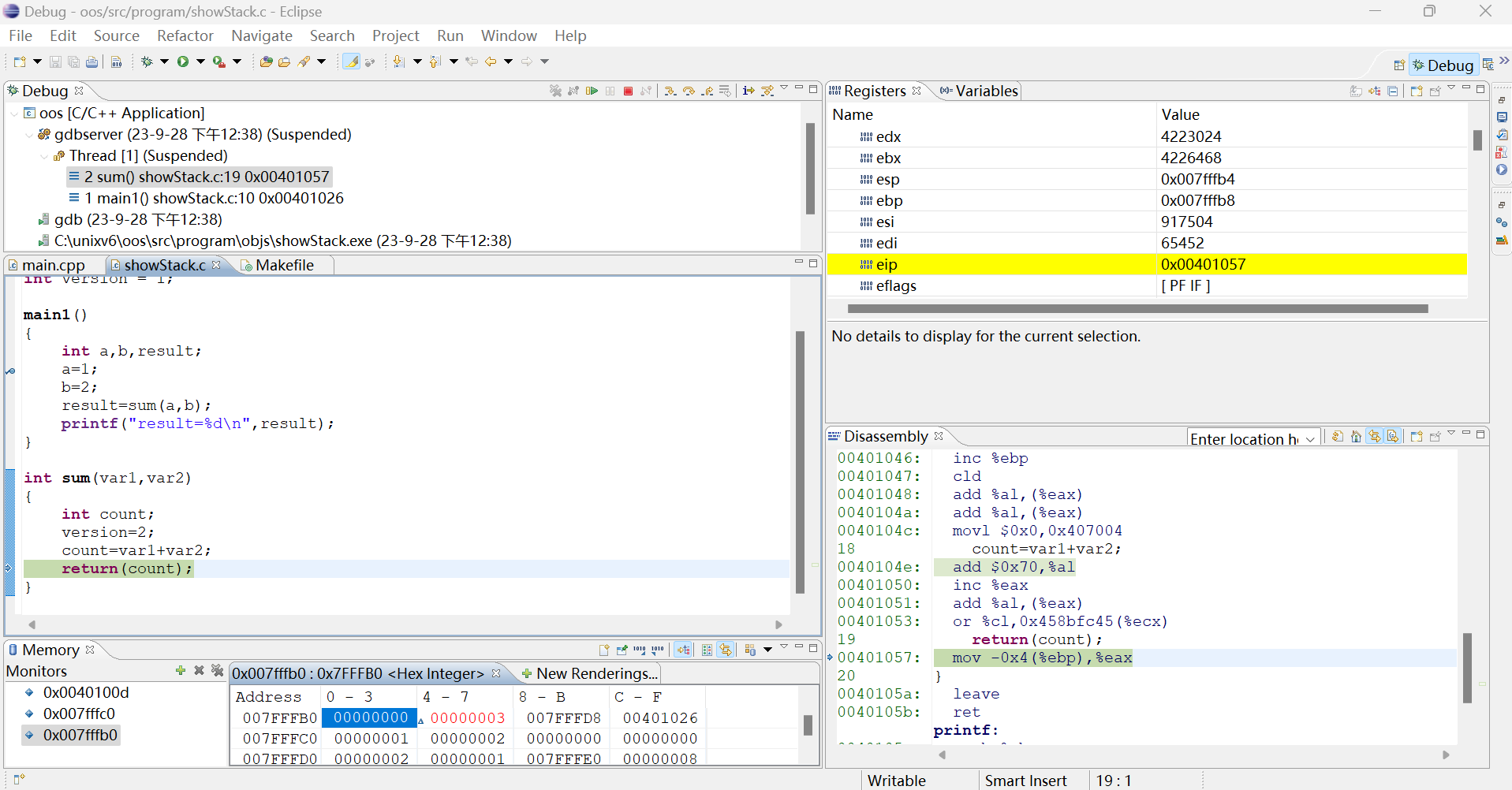
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| esp | 0x007fffb4 | 3 | 00401041 语句将 esp 修改到此处，空出 sum的局部变量count的位置；此处即为 count，00401054语句执行完后修改为 3 |
| ebp | 0x007fffb8 | 007fffd8 | 上一栈帧基址，0040103e 语句压栈；0040103f 语句将 ebp 修改到此处 |
|  | 0x007fffbc | 00401026 | sum的返回地址 |

3.3 sum函数内部运行过程中核心栈的变化

3.3.1进入sum函数之后，007fffc0和007fffc4分别存入1和2，007fffbc存入函数返回地址00401026，007fffc8存入上一栈帧基址007fffd8



3.3.2 局部变量count-3，存入地址007fffb4



四、在 main1 的汇编代码中出现了“sub $0x18,%esp”语句来预留出 6 个字，请查阅资料，解释原因

在这段汇编代码中，main1 函数会调用 sum 函数，且调用前后会有存储局部变量等操作，这些操作可能会用到堆栈空间。通过 "sub $0x18,%esp" 指令预留 24 字节的堆栈空间，确保有足够的空间来存储调用 sum 函数时所需的参数、局部变量以及其他可能需要存储的信息，以保证程序的正确执行和运行时的稳定性。

虽然实际运行过程中只用到了 20 个字节的堆栈空间（分别是局部变量a,b,result以及函数参数var1,var2），但可能是为了满足对齐要求、函数调用规范和代码生成优化，选择了预留 24 个字节的堆栈空间。