实验二、程序栈帧与内存布局（5 分）

同济大学计算机系《操作系统课程实验》 国豪班、拔尖班 邓蓉 2024 年 9 月 12 日第 1 版

一、Unix V6++的编译规则是：

- 虚空间中，代码段、数据段、只读数据段、bss 段紧贴放置，每个逻辑段按 4096 字节（4K）对齐。代码段中的指令 4 字节对齐；只读数据段中的常量 8 字节对齐；数据段和 bss 段中的变量 4 字节对齐。
- 可执行文件中，代码段、数据段、只读数据段，每个逻辑段按 512 字节对齐（512 字节是一个磁盘扇区的容量），size 登记的是逻辑段的实际长度（未对齐）。
- imageBegin 是代码段的首地址，等于 0x401000。栈底，0x800000（8M）。

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000020e8  00401000  00401000  00000400  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  00404000  00404000  00002600  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .rdata        00000204  00405000  00405000  00002800  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .bss          00000020  00406000  00406000  00000000  2**2
                  ALLOC
```

二、实验目的

熟悉 Unix V6++系统中，子程序调用、返回过程中栈帧的建立和销毁过程。
观察 Unix V6++系统中可执行文件的格式和内存布局。
掌握指令和数据的编址过程。

三、实验准备

配置完成的 Unix V6++系统。

四、实验报告要求与评分标准

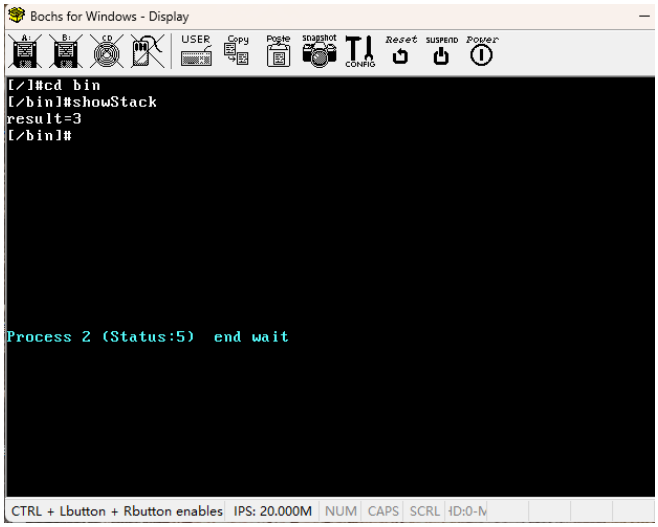
截图关键步骤，展示应用程序的调试过程。
回答文末列出的问题。

Part 1、调试 Unix V6++应用程序

一、用实验一 Part3 的方法，在 Unix V6++系统中添加应用程序 showStack。不

调试，Bochs 虚拟机上直接运行，结果如图所示。

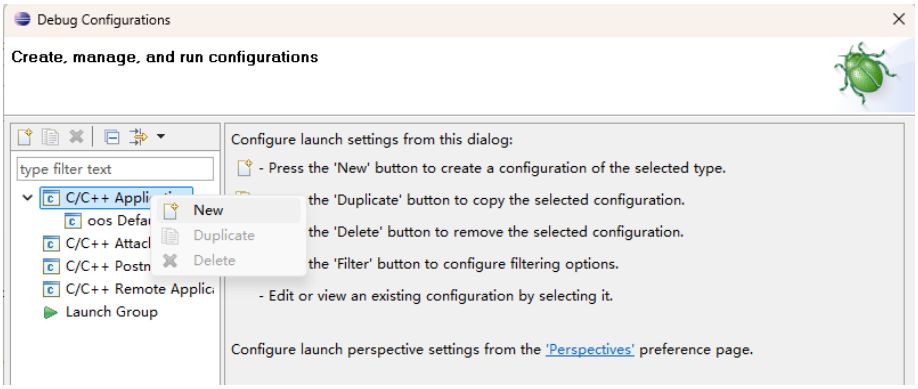
```
#include <stdio.h>

int version=1;

main1( )
{
    int a,b,result;
    a = 1;
    b = 2;
    result = sum(a,b);
    printf("result=%d\n",result);
}

int sum(var1, var2)
{
    int count;
    version = 2;
    count = var1 + var2;
    return(count);
}
```



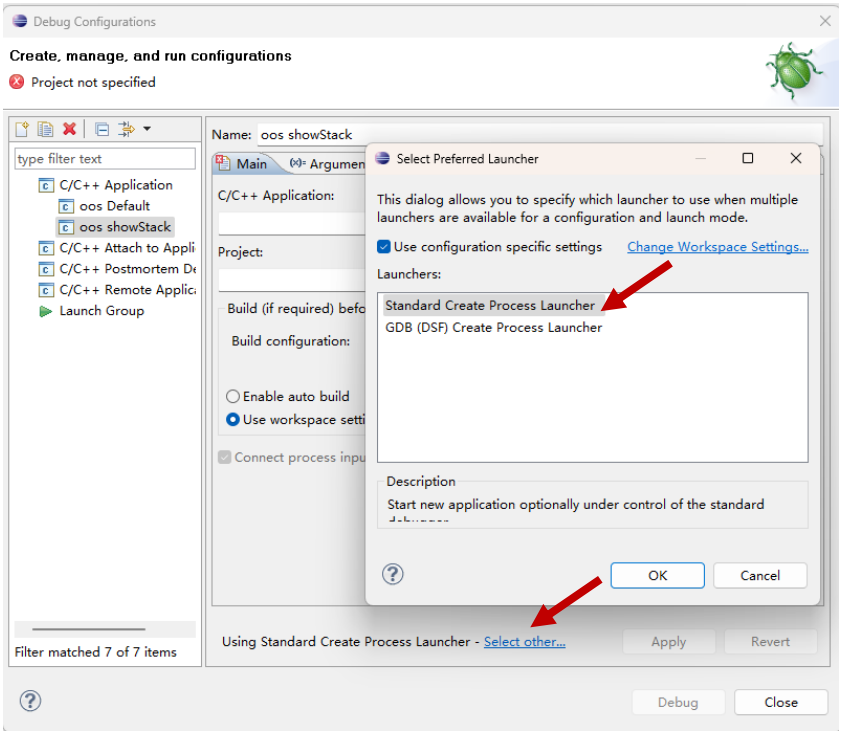
二、调试 showStack.exe，观察程序运行过程中栈帧的变化，关注寄存器 EIP、ESP、EBP 和 EAX。

1、 确保内核 kernel.exe 调试无误，系统可以在调试状态下运行 showStack.exe。

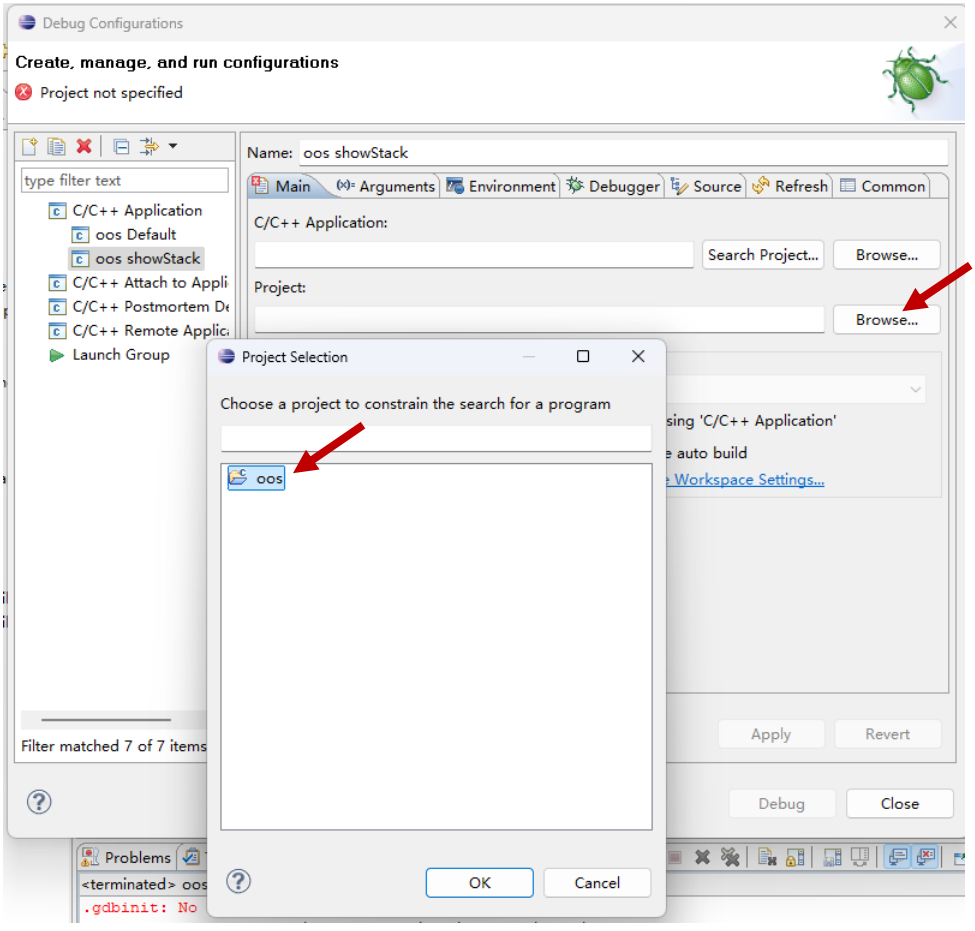
2、 新建一个调试配置（debug configuration），用来调试 showStack.exe。



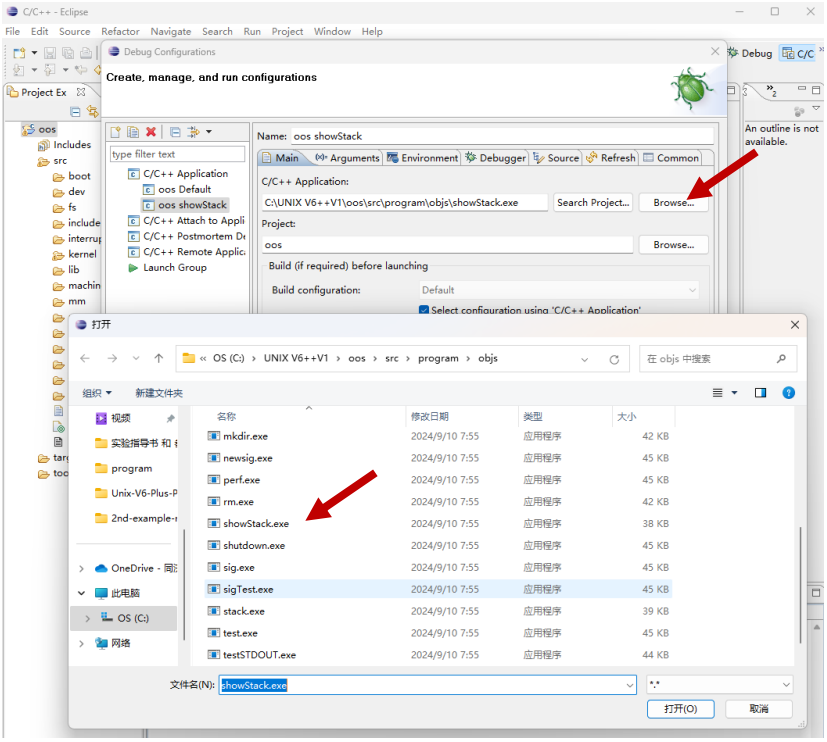
2.1 命名这个调试配置为 oos showStack，点 Select other。所有选择与实验一相同。点 OK。



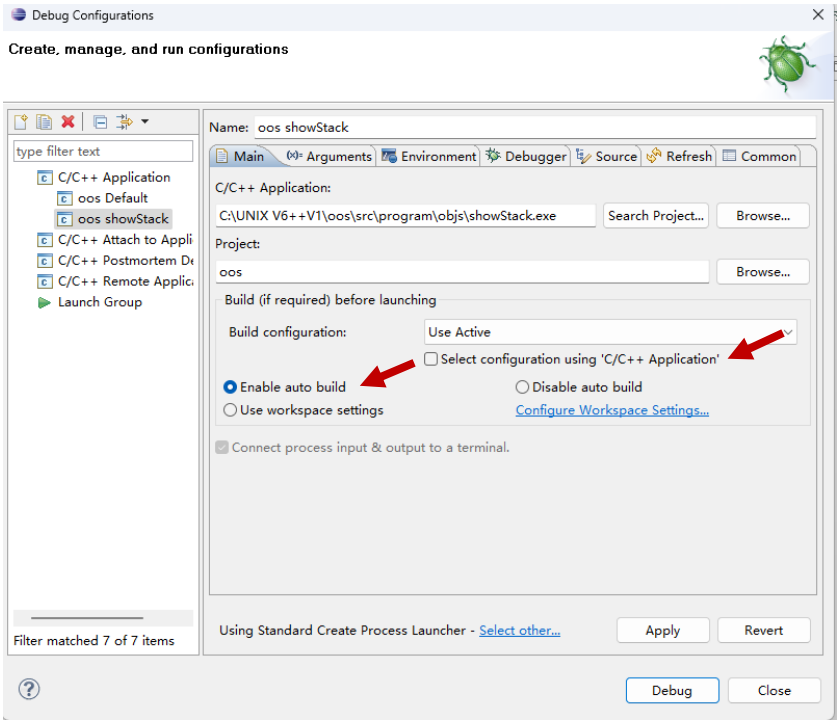
2.2 选一个 Project，oos



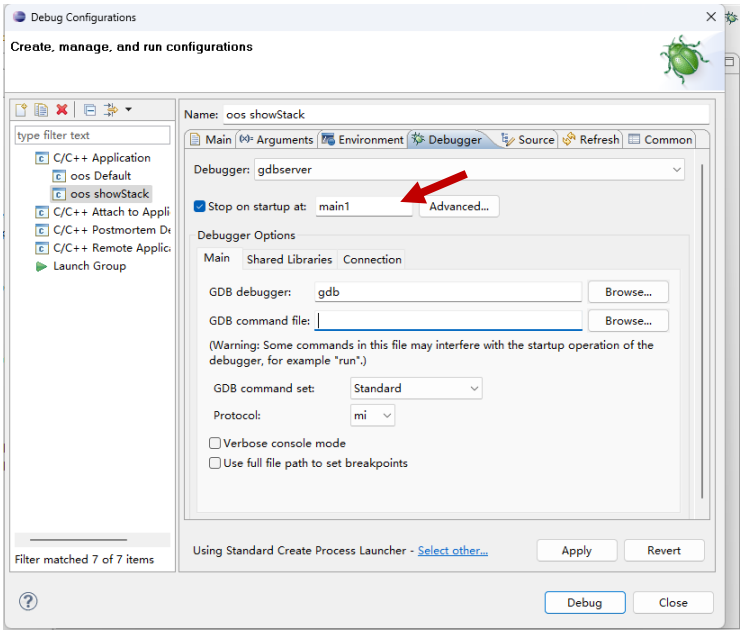
2.3　设置调试目标：src/program/objs/showStack.exe

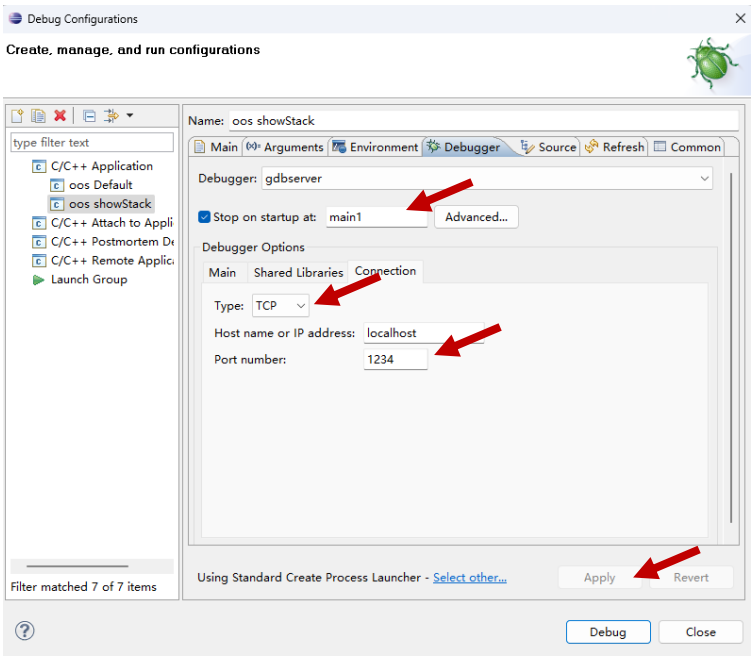


2.4　不选 Select configuration using ‘C/C++ Application
　　选中 Enable auto build

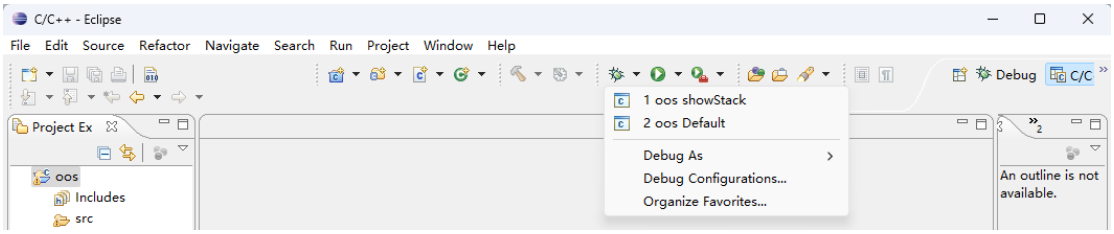


2.5　Debugger 选项卡设置调试的起始点。Unix V6++系统中，应用程序的入口是 main1，所以，**调试的起始点设为 main1**。其余设置与实验一完全相同。



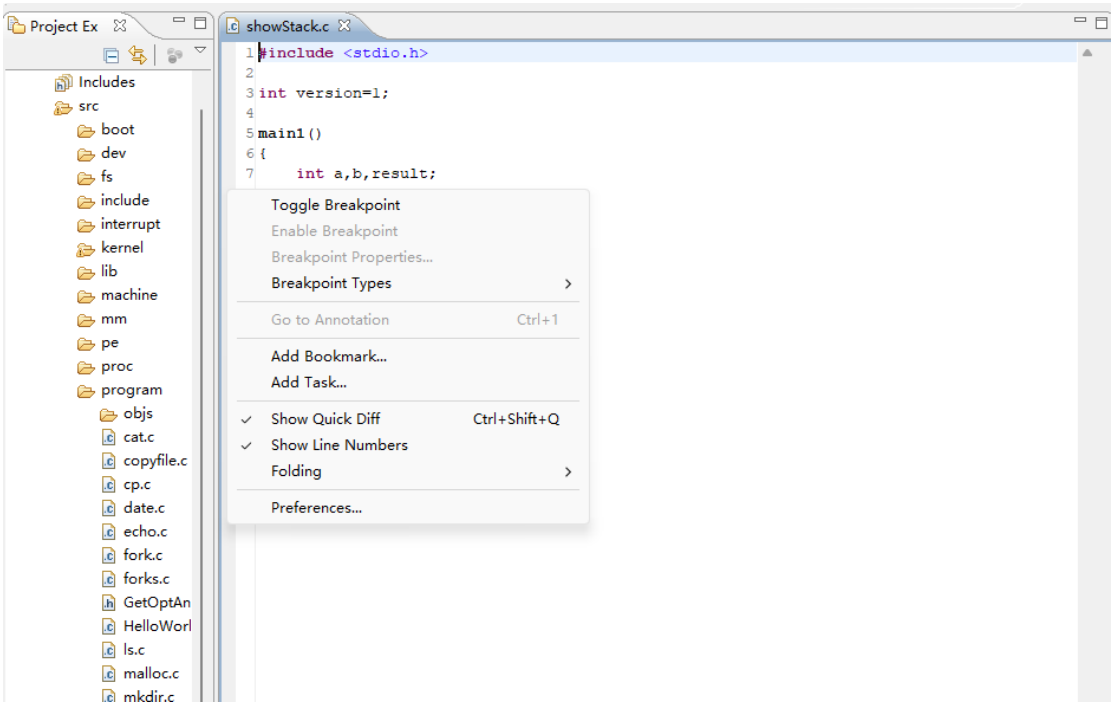


2.6　Apply，应用程序 showStack debug 配置设置成功。现在，我们有了 2 个调试配置。点 oos Default，调试内核，点 oos showStack，调试 showStack.exe 程序。想要调试其它应用程序，仿照 showStack 便可。

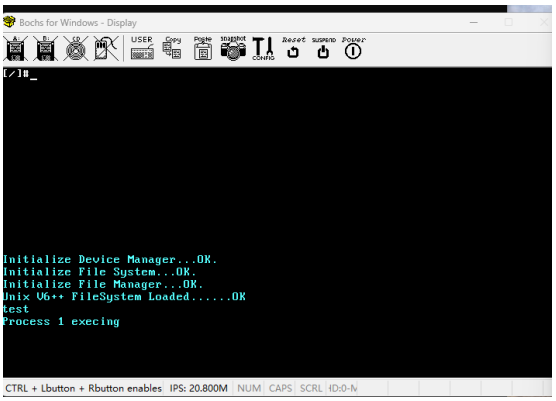


三、调试 showStack.exe 程序

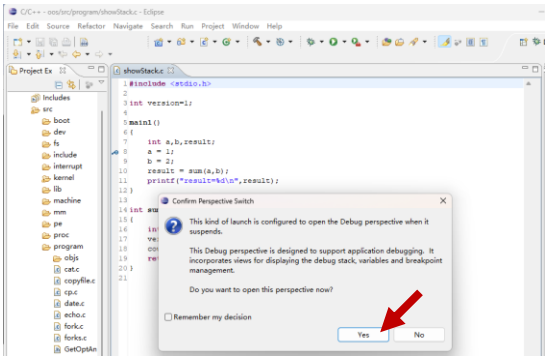
3.1　打断点。打开 showStack.c 程序，Toggle Breakpoint 在想停的位置。



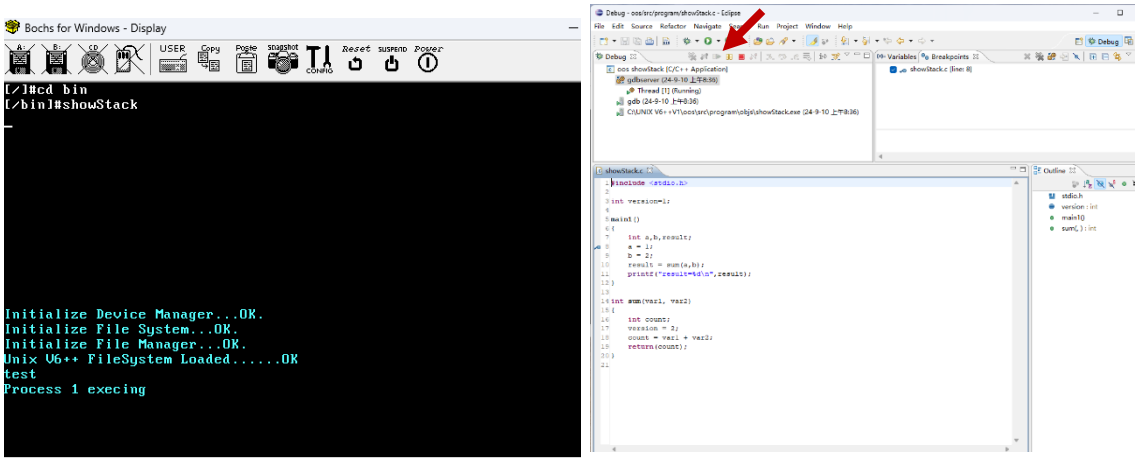
3.2　打开 bochs 虚拟机，让它等待调试命令。虚拟机会运行，至 shell 程序输出命令行提示符。如下图。



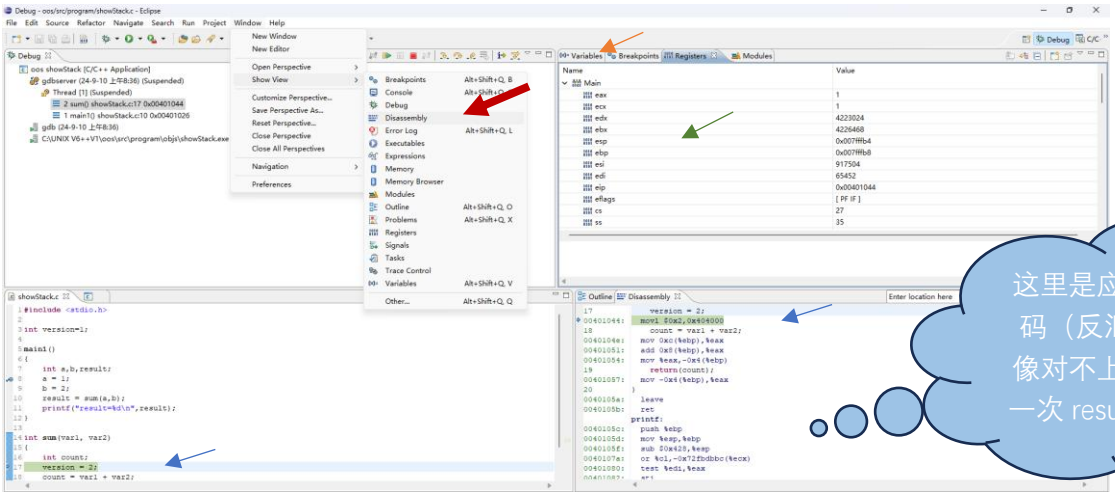
3.3　Eclipse，'Yes' 打开 Debug 视图。



3.4　回到 bochs 虚拟机，显示屏依次执行命令行 cd bin，showStack。其间，遇到停顿，按 Resume 按钮。

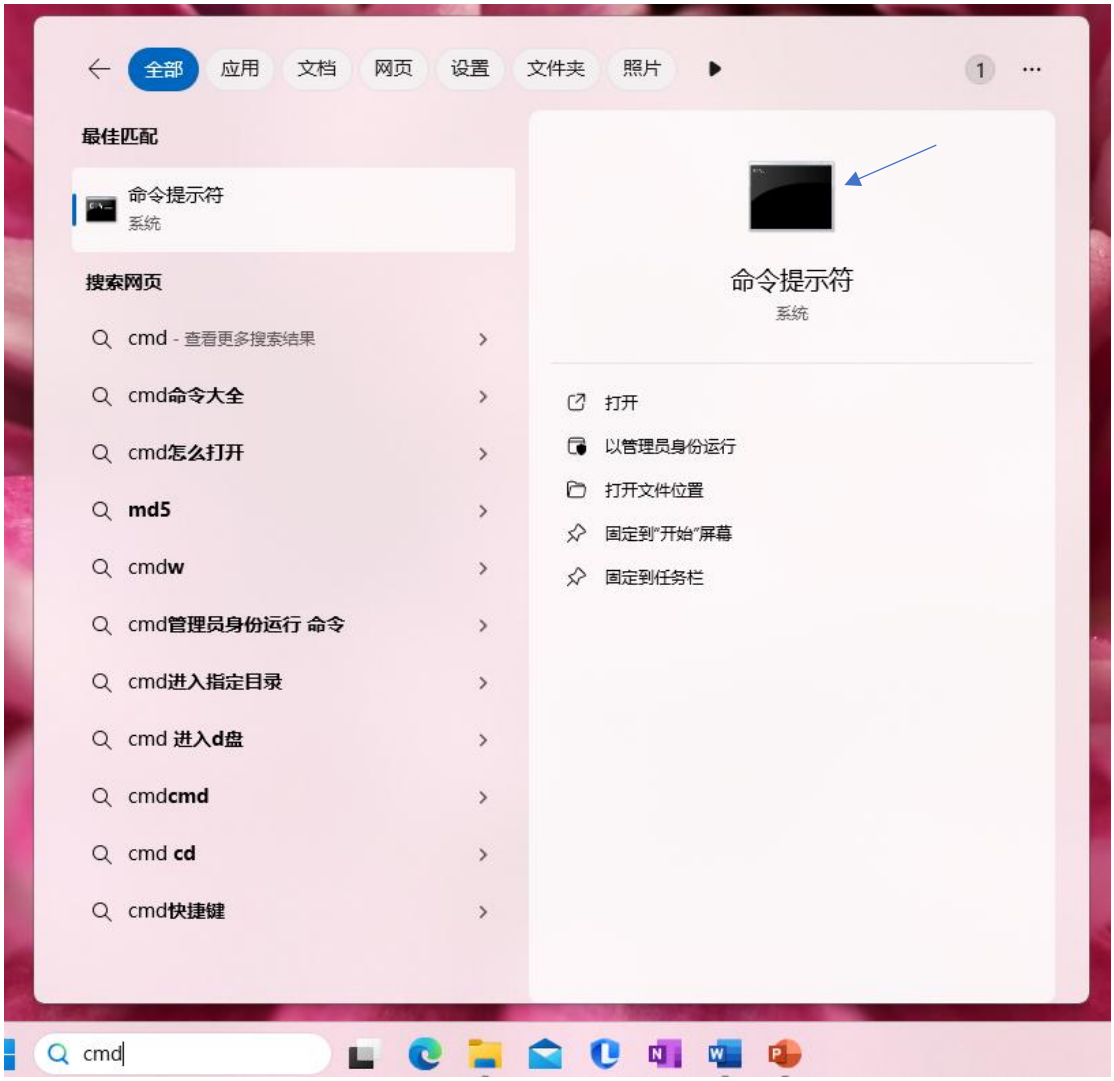


showStack 停下来了。我们可以单步执行它，同步观察它执行的每条汇编指令。为此，我们需要 Show View，打开 Disassembly 视图。蓝色箭头是正在执行的语句和它对应的汇编代码。绿色箭头处是所有寄存器的值。橘黄色箭头处可以看局部变量的值。随着单步执行，寄存器的值会变，变量的值也会变。观察每条指令执行期间，尤其是调用 sum( )函数期间，变化中的寄存器 EIP、ESP、EBP 和 EAX。



Part 2、用工具 objdump 探索 Unix V6++可执行程序

showStack 程序的源文件名：src/program/showStack.c，
可执行程序是 src/program/obj/showStack.exe。把它复制出来，保存在 windows 系统中。双击蓝色箭头指向的 cmd 图标，启动 windows 系统中的 cmd 程序。用工具 objdump 查看这个可执行文件的细节。



一、section headers，描述所有逻辑段的起始地址和长度

```
D:\UNIX V6++V1\oos\tools>objdump -h showStack.exe

showStack.exe:     file format pei-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000020e8  00401000  00401000  00000400  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  00404000  00404000  00002600  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .rdata        00000204  00405000  00405000  00002800  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .bss          00000020  00406000  00406000  00000000  2**2
                  ALLOC
  4 .idata        00000014  00407000  00407000  00002c00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .stab         00003fe4  00408000  00408000  00002e00  2**2
                  CONTENTS, READONLY, DEBUGGING, EXCLUDE
  6 .stabstr      0000124d  0040c000  0040c000  00006e00  2**0
                  CONTENTS, READONLY, DEBUGGING, EXCLUDE
```

程序是 pe 格式的，可以运行在 i386 平台上。有 6 个段。0#是代码段，1#是数据段，2#是只读数据段，3#是 bss 段。4、5、6 暂时不管它。4 用来引用外部数据，也就是其它源程序中定义的符号。

IDX，段号码；Name，段的名字；Size，段的实际长度；VMA，段在虚空间中的起始地址；LMA，？？？；File off，段的内容在可执行文件中的偏移量（起始地址）；Algn，段中所有元素的对齐规则。

段的名字：	代码段又叫正文段	.text
	数据段	.data
	只读数据段	.rdata
	bss 段	.bss

段的属性：
ALLOC 表示执行程序时要为这个段分配内存空间，
LOAD 表示需要从磁盘加载数据，
READONLY 只读。
CODE、DATA 是段的类型，代码 或 数据。

二、可执行码

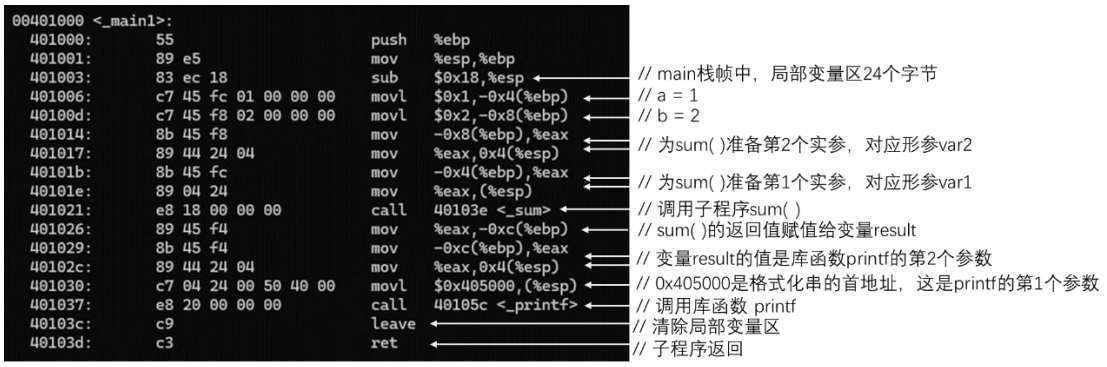
```
D:\UNIX V6++V1\oos\tools>objdump -d showStack.exe

showStack.exe:     file format pei-i386


Disassembly of section .text:

00401000 <_main1>:
  401000:       55                      push   %ebp
  401001:       89 e5                   mov    %esp,%ebp
  401003:       83 ec 18                sub    $0x18,%esp
  401006:       c7 45 fc 01 00 00 00    movl   $0x1,-0x4(%ebp)
  40100d:       c7 45 f8 02 00 00 00    movl   $0x2,-0x8(%ebp)
  401014:       8b 45 f8                mov    -0x8(%ebp),%eax
  401017:       89 44 24 04             mov    %eax,0x4(%esp)
  40101b:       8b 45 fc                mov    -0x4(%ebp),%eax
  40101e:       89 04 24                mov    %eax,(%esp)
  401021:       e8 18 00 00 00          call   40103e <_sum>
  401026:       89 45 f4                mov    %eax,-0xc(%ebp)
  401029:       8b 45 f4                mov    -0xc(%ebp),%eax
  40102c:       89 44 24 04             mov    %eax,0x4(%esp)
  401030:       c7 04 24 00 50 40 00    movl   $0x405000,(%esp)
  401037:       e8 20 00 00 00          call   40105c <_printf>
  40103c:       c9                      leave
  40103d:       c3                      ret

0040103e <_sum>:
  40103e:       55                      push   %ebp
  40103f:       89 e5                   mov    %esp,%ebp
  401041:       83 ec 04                sub    $0x4,%esp
  401044:       c7 05 00 40 40 00 02    movl   $0x2,0x404000
  40104b:       00 00 00
  40104e:       8b 45 0c                mov    0xc(%ebp),%eax
  401051:       03 45 08                add    0x8(%ebp),%eax
  401054:       89 45 fc                mov    %eax,-0x4(%ebp)
  401057:       8b 45 fc                mov    -0x4(%ebp),%eax
  40105a:       c9                      leave
  40105b:       c3                      ret

0040105c <_printf>:
  40105c:       55                      push   %ebp
  40105d:       89 e5                   mov    %esp,%ebp
  40105f:       81 ec 28 04 00 00       sub    $0x428,%esp
  401065:       8d 45 0c                lea    0xc(%ebp),%eax
  401068:       89 85 f4 fb ff ff       mov    %eax,-0x40c(%ebp)
  40106e:       8b 85 f4 fb ff ff       mov    -0x40c(%ebp),%eax
  401074:       89 44 24 08             mov    %eax,0x8(%esp)
```

…… 还有很多 ……

入口地址，main1，0x401000。

注意，栈帧与我们上课讲的有差异（它有对齐）。

main1( )函数，汇编指令注释。



三、需要回答的问题（每题 1 分）

1、注释 sum( )函数汇编码。
2、绘制 showStack 程序的栈。在 debug 视图单步，观察栈帧的创建和销毁过程。绘图展示堆栈的变化过程。
3、使用 objdump 命令，发现应用程序的更多细节。
4、修改 showStack( )程序，printf 出来 main，sum，printf，version 以及所有局部变量的地址，这些地址是否与你期待的相符。绘制 showStack( )程序的内存布局，标出[0,8M)内存空间中，代码段、数据段，堆栈段……各个逻辑段的具体位置，标出本题列出的所有符号的位置。
5、（1）在 Windows 系统中打开 cmd 窗口，重新编译 showStack.c，依然使用我们包里 MinGW/bin 中的 gcc。观察问题 4 中程序的输出结果。解释你观察到的现象。（2*）再次修改 showStack 程序，启用堆，制造 OOM 错误。
6、你的发现与思考。