**电 子 科 技 大 学**

软件开发环境实验二：函数调用栈帧布局

**学生姓名：任振华 学号：2017060801023**

**指导教师：李林 实验时间：2019/12/22**

## 1实验目的

本实验总体目的是，通过使用Visual Studio 2017查看函数调用时参数、局部变量等在栈上的分布情况，以达到掌握函数调用时栈帧布局的目的。

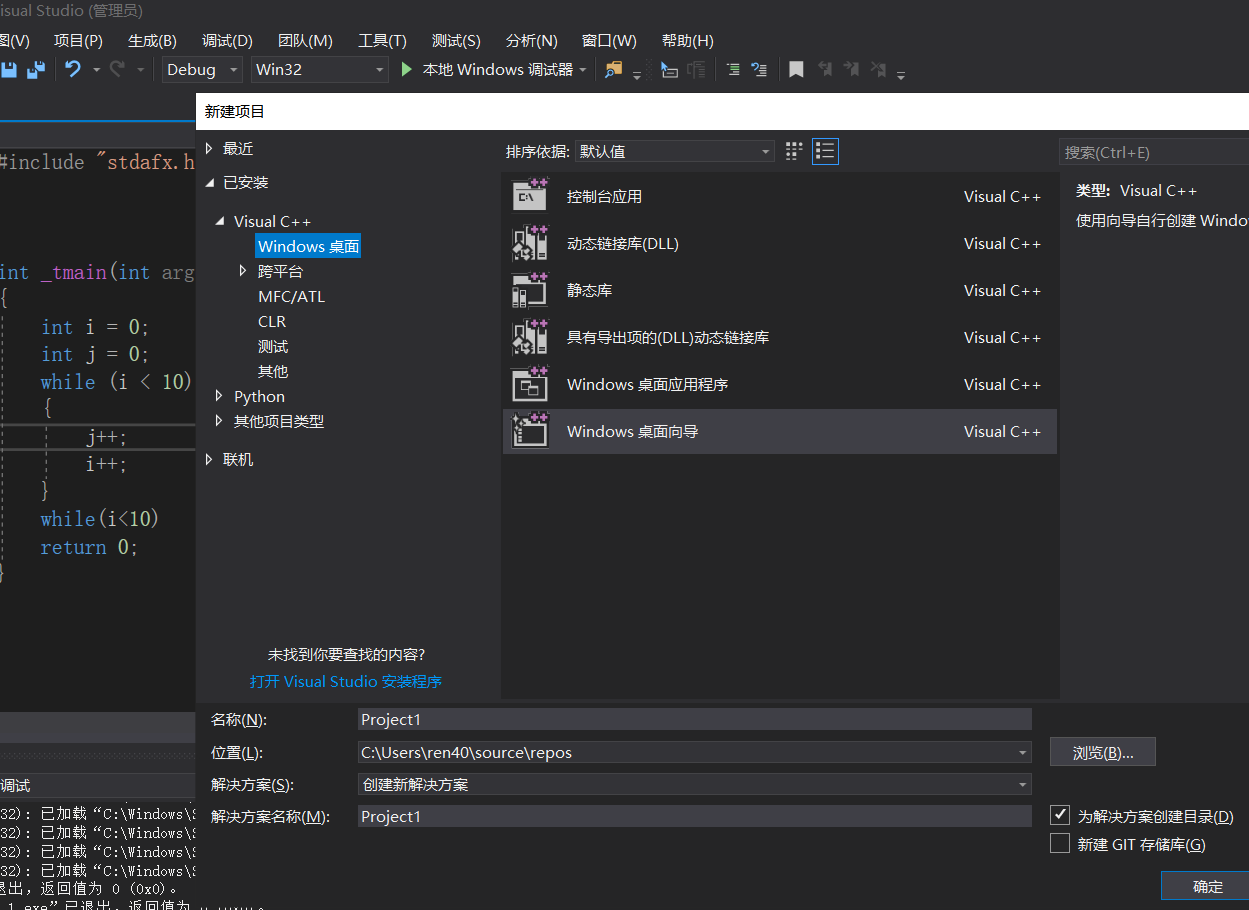
## 2实验原理

编译器对函数调用的支持，通常情况下都会使用栈。例如使用栈传递传参，保存函数返回地址。另外，局部变量也通常位于栈上。Visual Studio 2017为了防止栈上局部数组溢出，又采取了特殊的保护措施。本实验就需要通过观察栈帧布局，来了解这些保护措施。

本实验的环境是Visual Studio 2017。

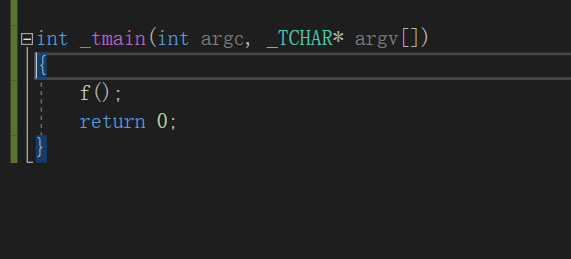
## 3实验步骤及要求

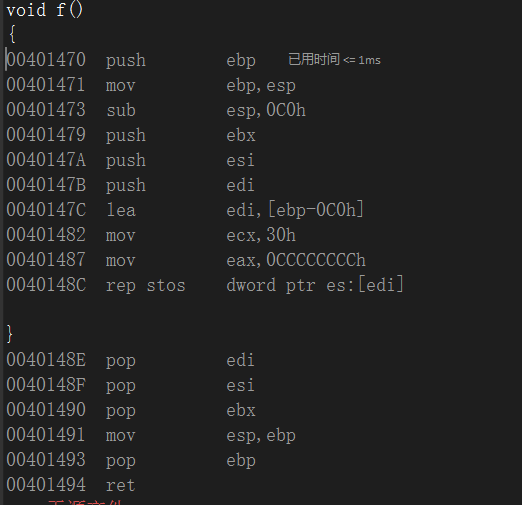
## 3.1工程的创建



## 3.2.1函数中无任何局部变量的情况

## 代码清单1





void f()

{

00401470 push ebp

将ebp寄存器的值存入栈中

00401471 mov ebp,esp

将esp寄存器的值存入ebp寄存器

00401473 sub esp,0C0h

sub指令是减法指令，该指令是将esp-12,再把结果送到esp

00401479 push ebx

将ebx出栈

0040147A push esi

；将esi压栈

0040147B push edi

；将edi压栈

0040147C lea edi,[ebp-0C0h]

；lea是取地址指令，该指令使edi = [ebp-0C0h]

00401482 mov ecx,30h

；使寄存器ecx的值为30h

00401487 mov eax,0CCCCCCCCh

；使寄存器eax的值为0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

；rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

}

0040148E pop edi

；将edi寄存器出栈

0040148F pop esi

；将esi寄存器出栈

00401490 pop ebx

；将ebp寄存器出栈

00401491 mov esp,ebp

；把ebp寄存器的值送到esp中

00401493 pop ebp

将ebp寄存器出栈

00401494 ret

；结束f()函数，返回到调用源

根据代码清单1的反汇编代码，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图2。

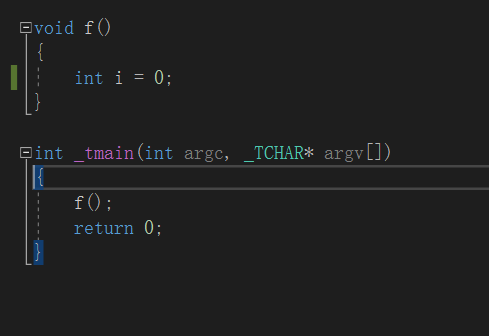
|  |
| --- |
|  |
|  |  |

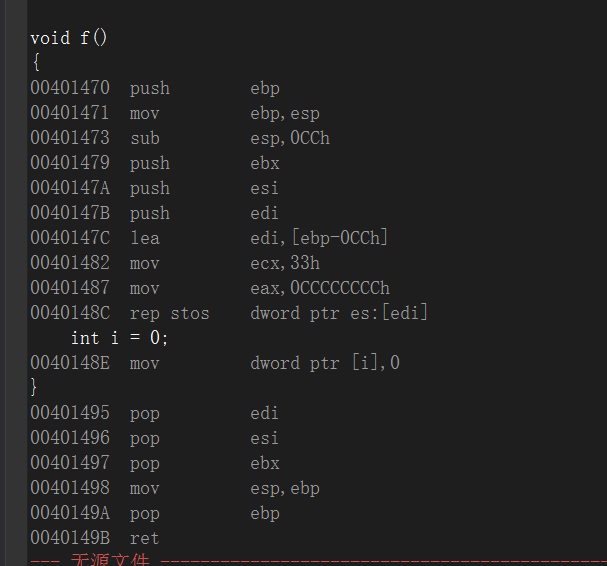
图2无局部变量时的布局

从图2可以知道，即使没有任何局部变量，栈上仍然有0xc0个字节的0xcc存在。这个是为了检测是否有溢出而写的。

## 3.2.2函数中只有一个局部变量的情况

## 代码清单3





void f()

{

00401470 push ebp

;将ebp寄存器压栈

00401471 mov ebp,esp

;将esp寄存器的数据送到ebp

00401473 sub esp,0CCh

;sub是减法指令,esp = esp - 12

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp-0CCh]

;lea是取地址指令,edi = ebp – 0CCh

00401482 mov ecx,33h

;使exc = 33h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

int i = 0;

0040148E mov dword ptr [i],0

；使 i = 0

}

00401495 pop edi

；将edi寄存器出栈

00401496 pop esi

；将esi寄存器出栈

00401497 pop ebx

；将ebx寄存器出栈

00401498 mov esp,ebp

；使esp = ebp

0040149A pop ebp

；将ebp寄存器出栈

0040149B ret

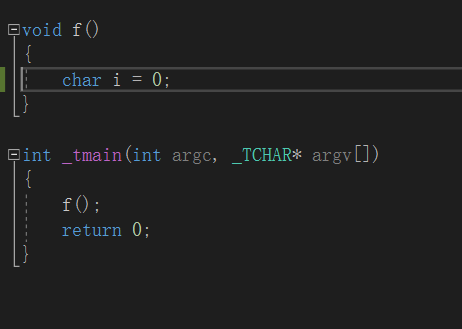
；结束f()函数，返回到调用源

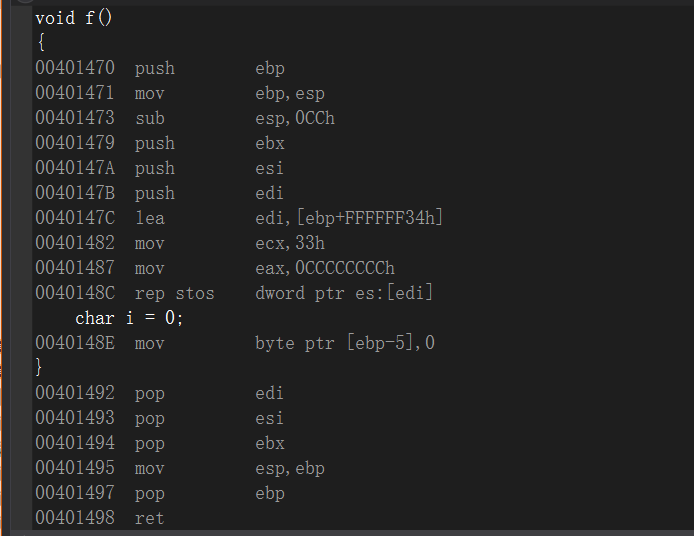
根据代码清单4，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图4。

|  |
| --- |
|  |
|  |  |

**图4 只有一个整型局部变量时的布局**

## 代码清单5





void f()

{

00401470 push ebp

;将ebp寄存器压栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0CCh

;sub是减法指令，使esp = esp – 0CCh

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF34h]

;lea是取地址指令，这条指令使edi = ebp + FFFFFF34h

00401482 mov ecx,33h

;使ecx = 33h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

char i = 0;

0040148E mov byte ptr [ebp-5],0

;使\*(ebp - 5) = 0,byte是字节，ptr是属性修饰符

}

00401492 pop edi

;将edi寄存器出栈

00401493 pop esi

;将esi寄存器出栈

00401494 pop ebx

;将ebx寄存器出栈

00401495 mov esp,ebp

; 使esp = ebp

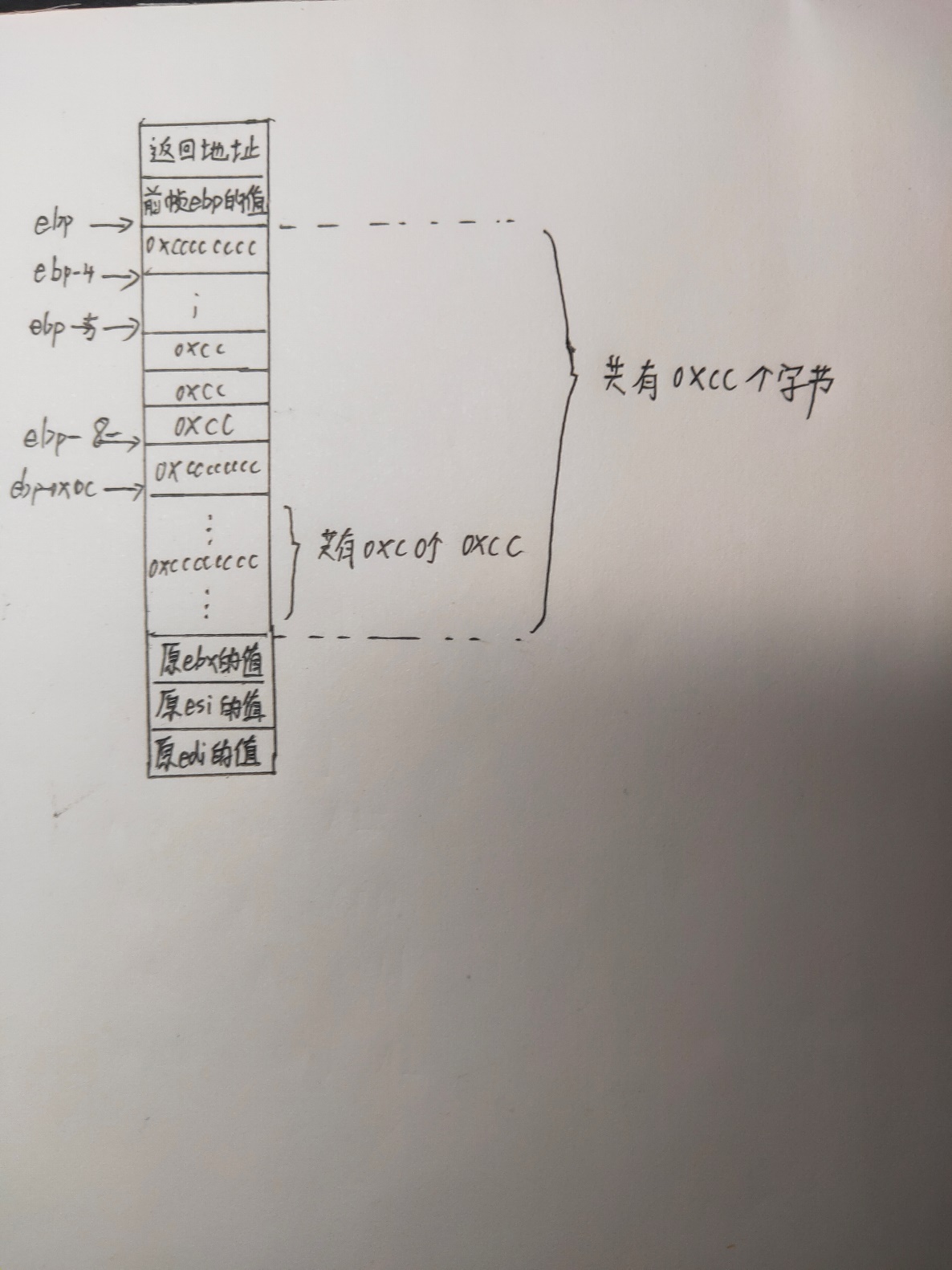
00401497 pop ebp

; 将ebp寄存器出栈

00401498 ret

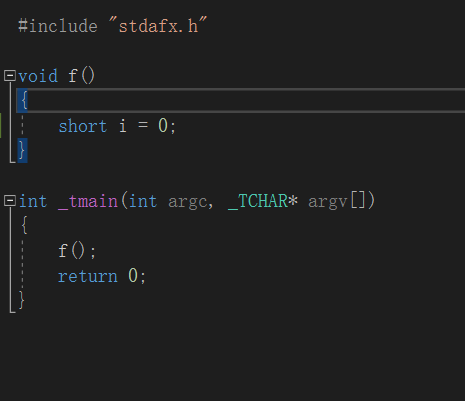
; 结束f()函数，返回到调用源

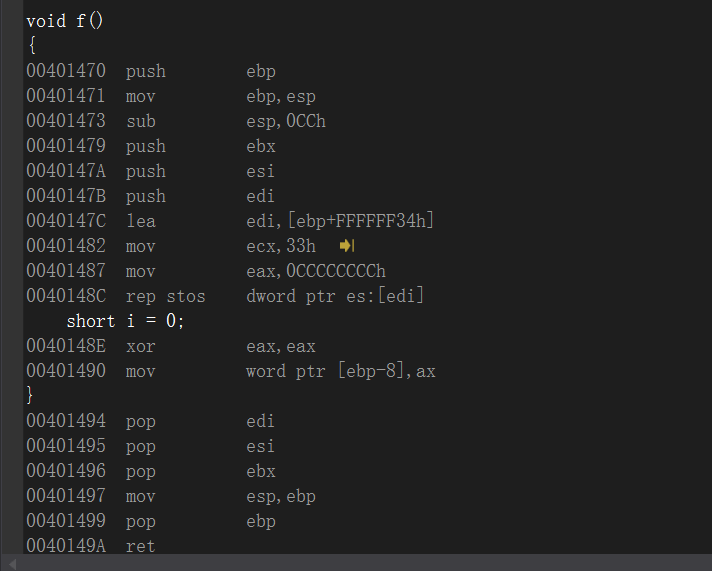
根据代码清单5，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图5



**图5 只有一个char型局部变量时的布局**

## 代码清单6





void f()

{

00401470 push ebp

;将ebp寄存器压栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0CCh

; sub是减法指令，使esp = esp – 0CCh

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF34h]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF34h

00401482 mov ecx,33h

;使ecx = 33h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

short i = 0;

0040148E xor eax,eax

;使eax寄存器清0

00401490 mov word ptr [ebp-8],ax

;使\*(ebp-8) = ax ,而eax寄存器已经被清0

}

00401494 pop edi

;将edi寄存器出栈

00401495 pop esi

;将esi寄存器出栈

00401496 pop ebx

;将ebx寄存器出栈

00401497 mov esp,ebp

;使esp = ebp

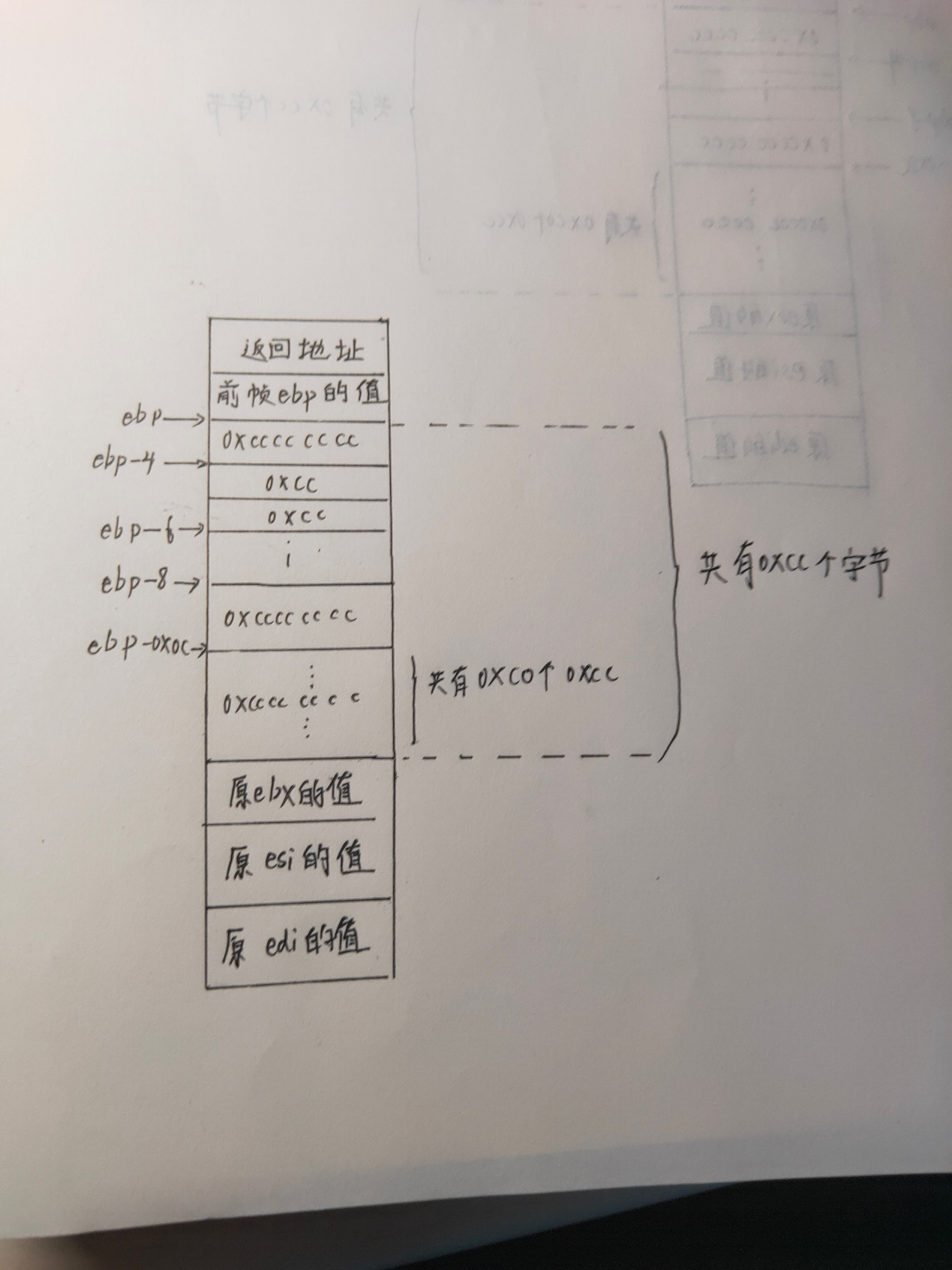
00401499 pop ebp

;将ebp寄存器出栈

0040149A ret

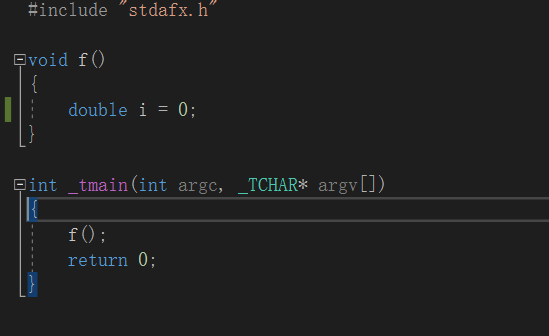
; 结束f()函数，返回到调用源

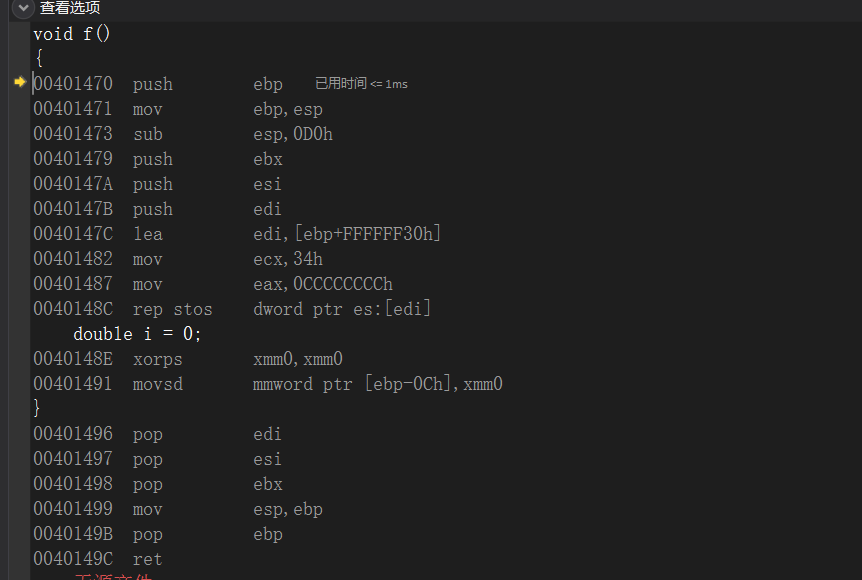
根据代码清单6，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图6



**图6 只有一个short型局部变量时的布局**

## 代码清单7





void f()

{

00401470 push ebp

;ebp寄存器压栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0D0h

; sub是减法指令，使esp = esp – 0D0h

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF30h]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF30h

00401482 mov ecx,34h

;使ecx = 34h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

double i = 0;

0040148E xorps xmm0,xmm0

;xorps—压缩单精度浮点值的按位逻辑异或, xmm0是SSE的128位寄存器，该指令使xmm0寄存器清0

00401491 movsd mmword ptr [ebp-0Ch],xmm0

;movsd是数据传送指令，传送一个双字

;mmword用于具有MMX和SSE(XMM)命令的64位多媒体操作数

}

00401496 pop edi

;将edi寄存器出栈

00401497 pop esi

;将esi寄存器出栈

00401498 pop ebx

;将ebx寄存器出栈

00401499 mov esp,ebp

;使esp = ebp

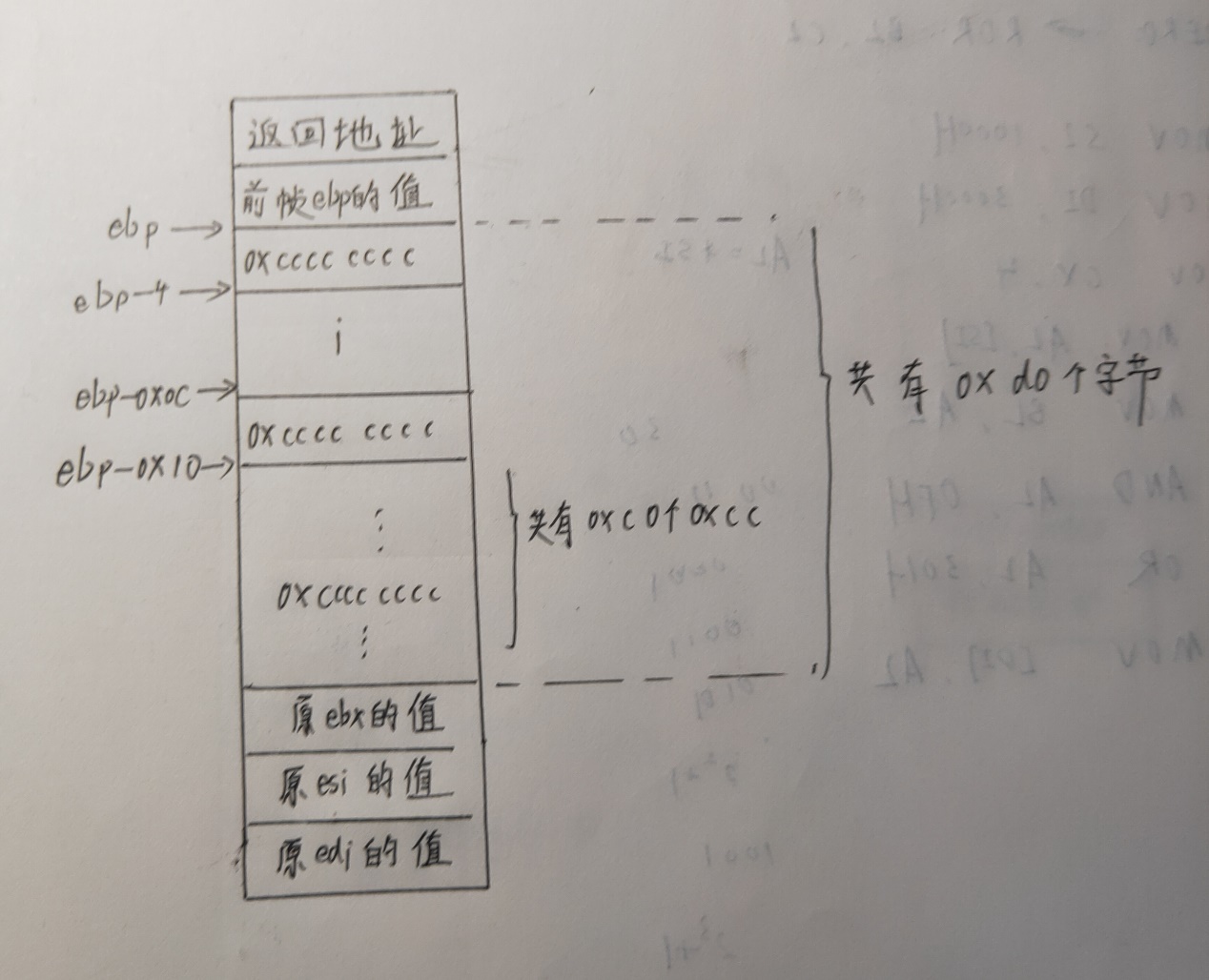
0040149B pop ebp

将ebp寄存器出栈

0040149C ret

; 结束f()函数，返回到调用源

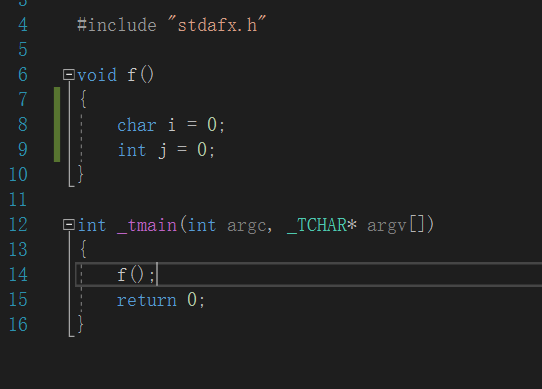
根据代码清单7，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图7

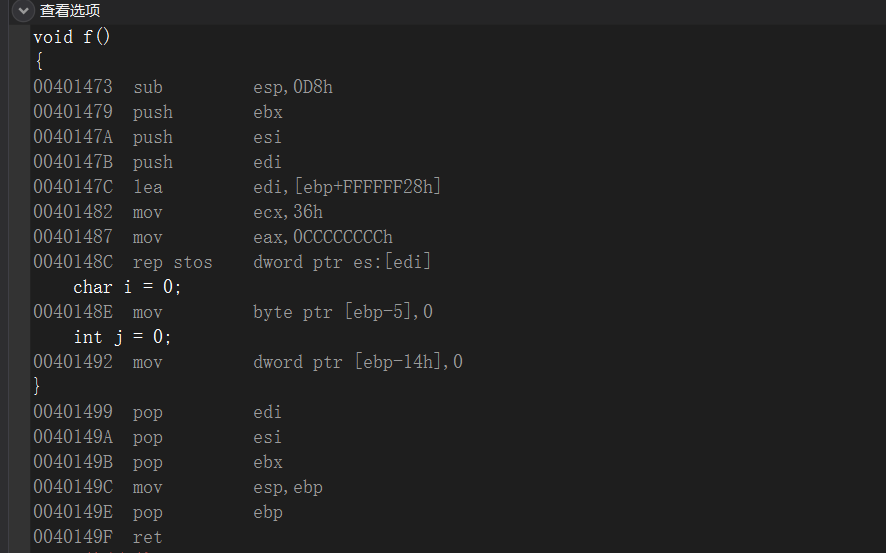
**图7 只有一个double型局部变量时的布局**

|  |
| --- |
|  |
|  |  |

## 3.2.3函数中有多个局部变量的情况

## 代码清单8





void f()

{

00401473 sub esp,0D8h

; sub是减法指令，使esp = esp – 0D8h

00401479 push ebx

;将ebx寄存器入栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF28h]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF28h

00401482 mov ecx,36h

;使ecx = 36h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

char i = 0;

0040148E mov byte ptr [ebp-5],0

;这条指令使 \*(ebp-5) = 0,byte是字节

int j = 0;

00401492 mov dword ptr [ebp-14h],0

;这条指令使\*(ebp-14h) = 0，dword是双字，即四字节

}

00401499 pop edi

;将edi寄存器出栈

0040149A pop esi

;将esi寄存器出栈

0040149B pop ebx

;将ebx寄存器出栈

0040149C mov esp,ebp

;使esp = ebp

0040149E pop ebp

;将ebp寄存器出栈

0040149F ret

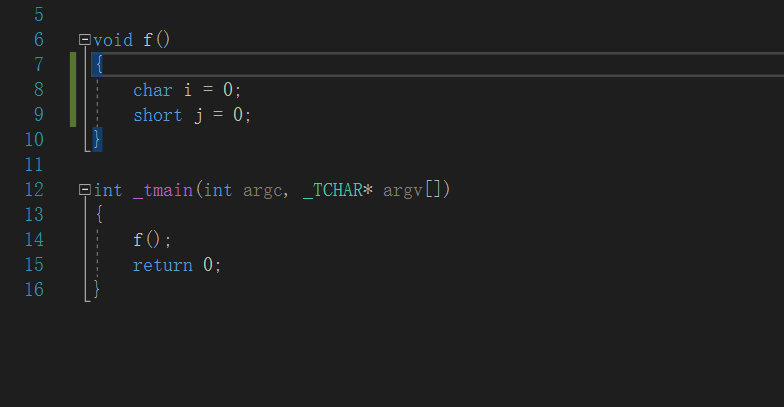
; 结束f()函数，返回到调用源

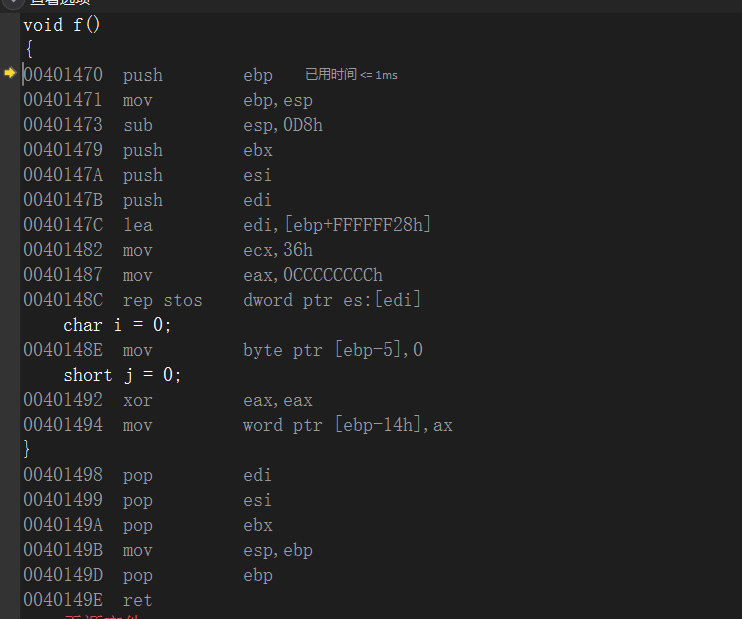
根据代码清单8，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图8。

|  |
| --- |
|  |
|  |  |

**图8代码清单8对应的栈帧布局**

## 代码清单10





void f()

{

00401470 push ebp

;将ebp寄存器压栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0D8h

; sub是减法指令，使esp = esp – 0D8h

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF28h]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF28h

00401482 mov ecx,36h

;使ecx = 36h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

char i = 0;

0040148E mov byte ptr [ebp-5],0

;使\*(ebp-5) = 0, byte是字节

short j = 0;

00401492 xor eax,eax

;xor是异或指令,使eax为0

00401494 mov word ptr [ebp-14h],ax

;使\*(ebp-14h) = ax,而ax为0

}

00401498 pop edi

;将edi寄存器出栈

00401499 pop esi

;将esi寄存器出栈

0040149A pop ebx

;将ebx寄存器出栈

0040149B mov esp,ebp

;使esp = ebp

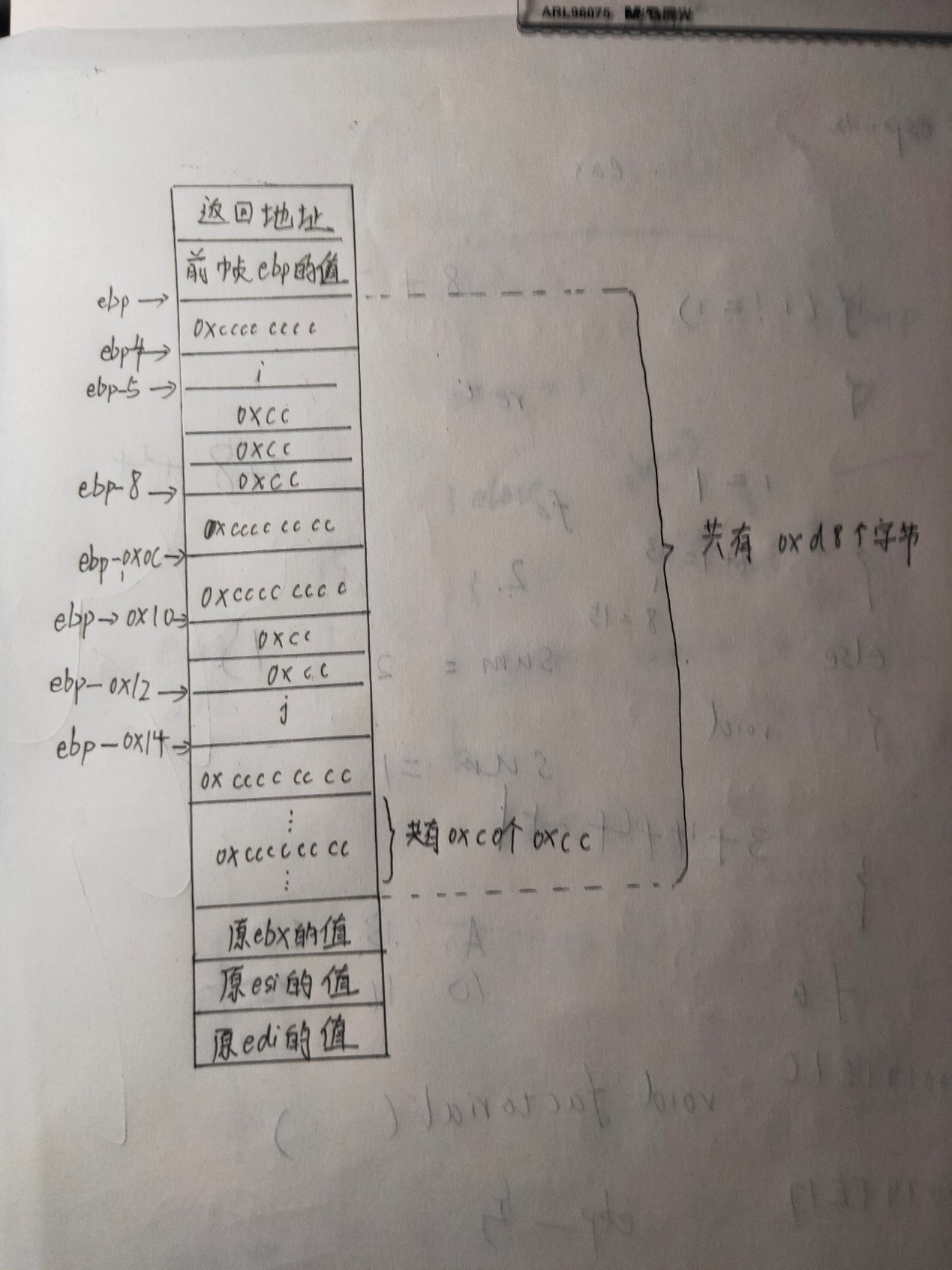
0040149D pop ebp

;将ebp寄存器出栈

0040149E ret

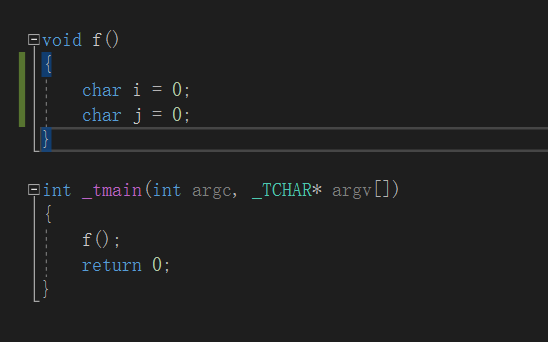
; 结束f()函数，返回到调用源

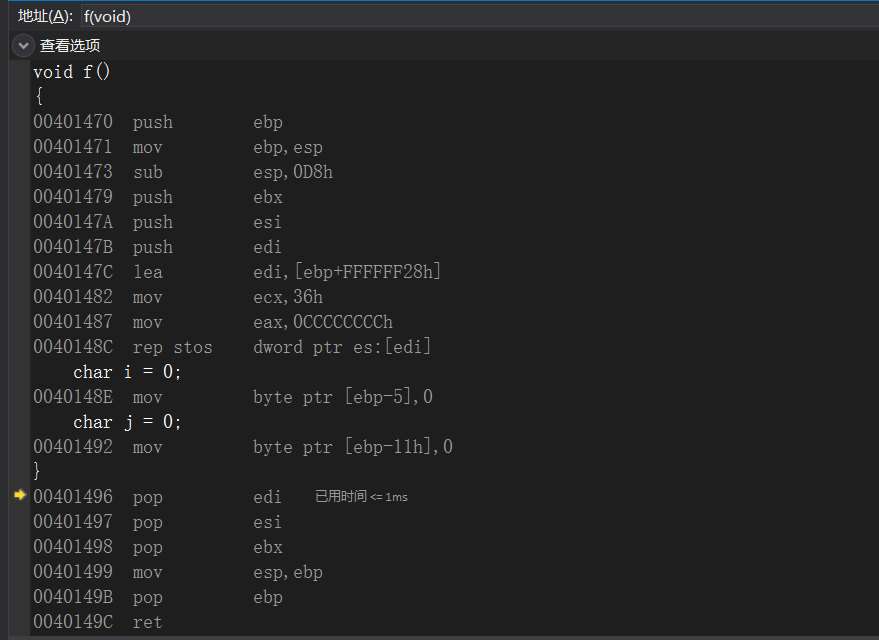
根据代码清单10，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图10。



**图10代码清单10对应的栈帧布局**

## 代码清单11





void f()

{

00401470 push ebp

;将ebp寄存器入栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0D8h

; sub是减法指令，使esp = esp – 0D8h

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF28h]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF28h

00401482 mov ecx,36h

;使ecx = 36h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

char i = 0;

0040148E mov byte ptr [ebp-5],0

;使\*(ebp-5) = 0, byte是字节

char j = 0;

00401492 mov byte ptr [ebp-11h],0

;使\*(ebp-11h) = 0, byte是字节

}

00401496 pop edi

;将edi寄存器出栈

00401497 pop esi

;将esi寄存器出栈

00401498 pop ebx

;将ebx寄存器出栈

00401499 mov esp,ebp

;使esp = ebp

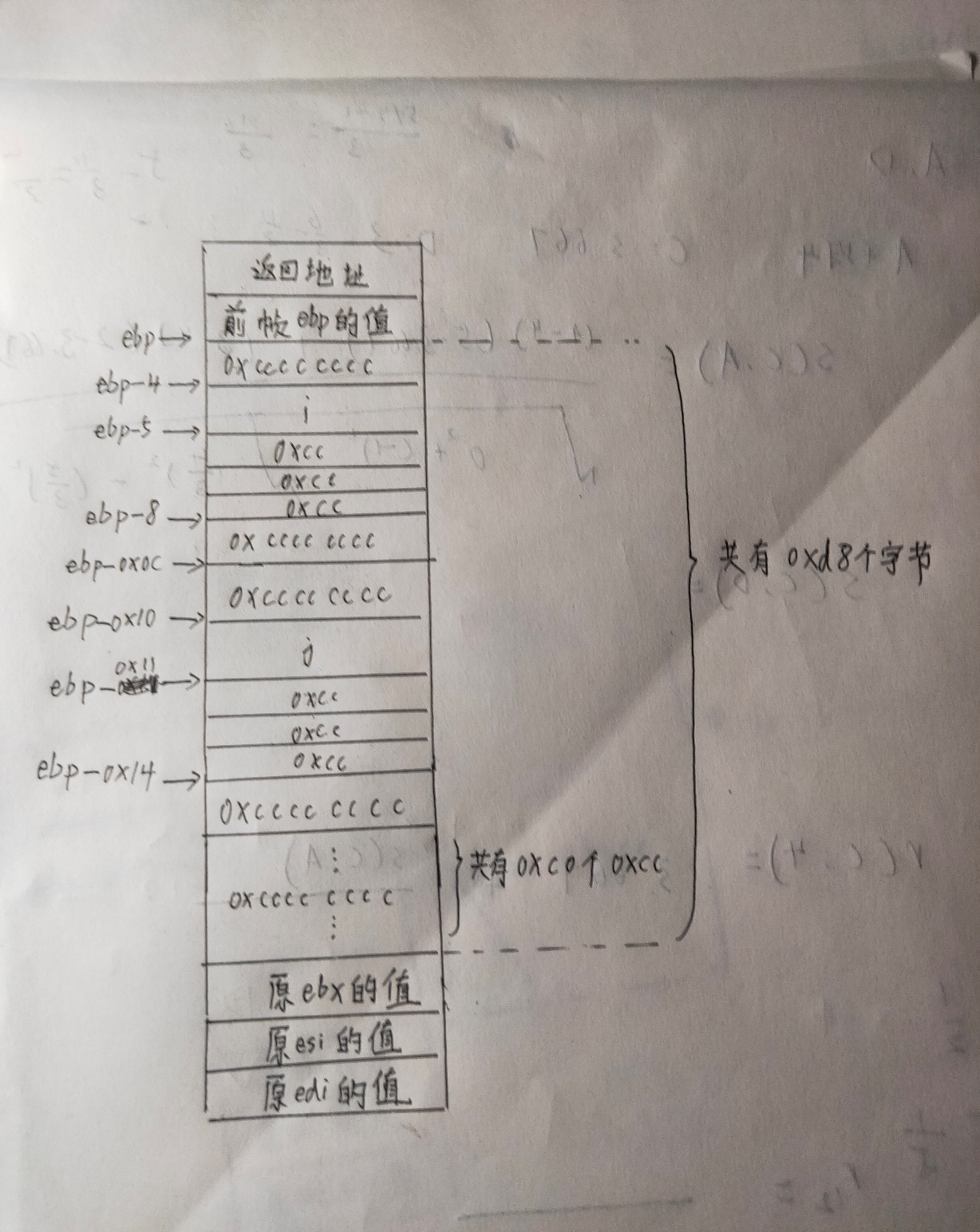
0040149B pop ebp

;将ebp寄存器出栈

0040149C ret

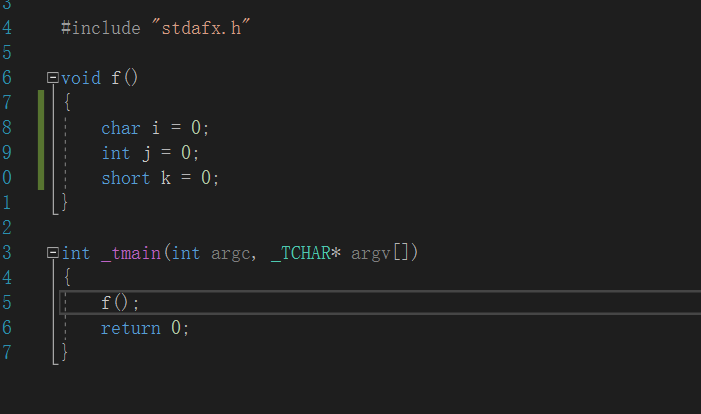
; 结束f()函数，返回到调用源

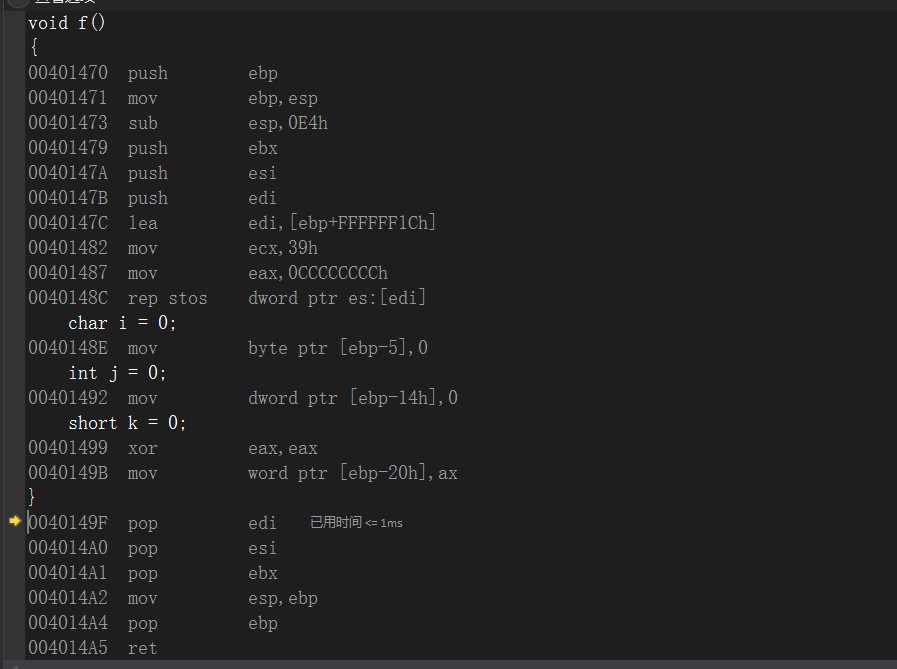
根据代码清单10，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图10。



**图11代码清单11对应的栈帧布局**

## 代码清单12





void f()

{

00401470 push ebp

;将ebp寄存器入栈

00401471 mov ebp,esp

;使ebp = esp

00401473 sub esp,0E4h

;sub是减法指令,使esp = esp – 0E4h

00401479 push ebx

;将ebx寄存器压栈

0040147A push esi

;将esi寄存器压栈

0040147B push edi

;将edi寄存器压栈

0040147C lea edi,[ebp+FFFFFF1Ch]

; lea是取地址指令，这条指令使edi = ebp + FFFFFF1Ch

00401482 mov ecx,39h

;使ecx = 39h

00401487 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

0040148C rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

char i = 0;

0040148E mov byte ptr [ebp-5],0

;使\*(ebp-5) = 0,byte是字节

int j = 0;

00401492 mov dword ptr [ebp-14h],0

;使\*(ebp-14h) = 0

short k = 0;

00401499 xor eax,eax

;xor是异或指令，使eax = 0

0040149B mov word ptr [ebp-20h],ax

;使\*(ebp-20h) = ax

}

0040149F pop edi

;将edi寄存器出栈

004014A0 pop esi

;将esi寄存器出栈

004014A1 pop ebx

;将ebx寄存器出栈

004014A2 mov esp,ebp

;使esp = ebp

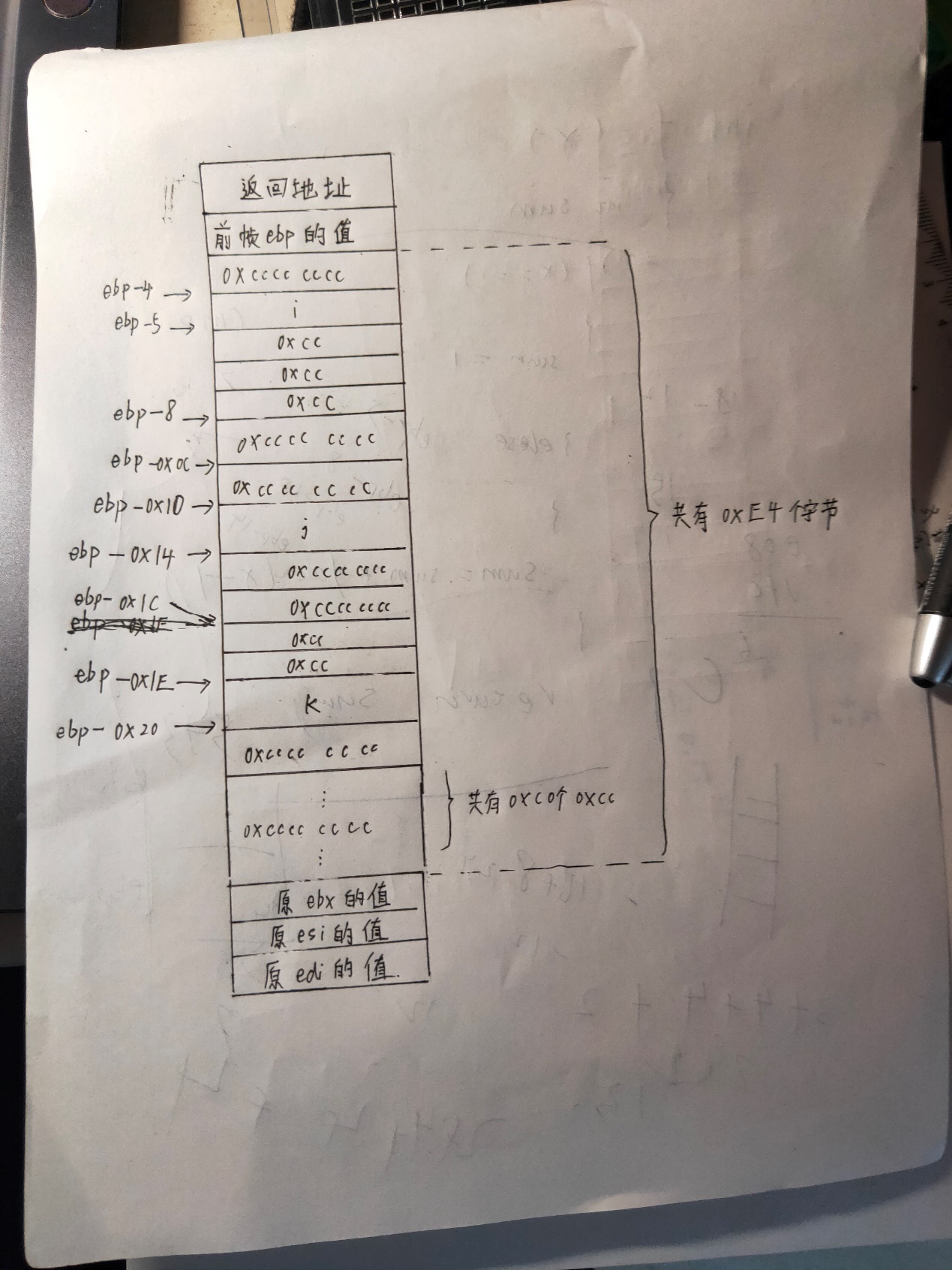
004014A4 pop ebp

;将ebp寄存器出栈

004014A5 ret

; 结束f()函数，返回到调用源

根据代码清单12，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图12。



**图12代码清单12对应的栈帧布局**

## 栈帧布局的规律总结

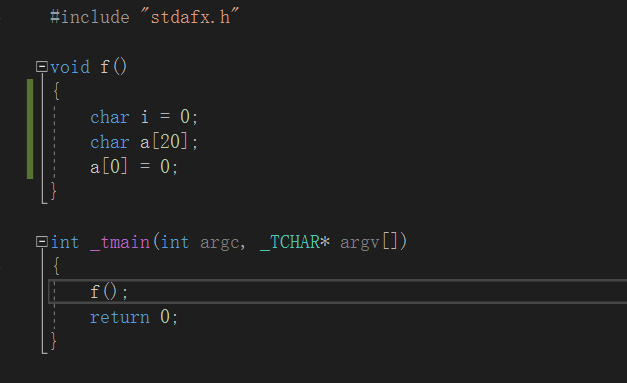
1.在函数栈帧结构中ebp是栈底，即高地址，esp是栈顶，即低地址。

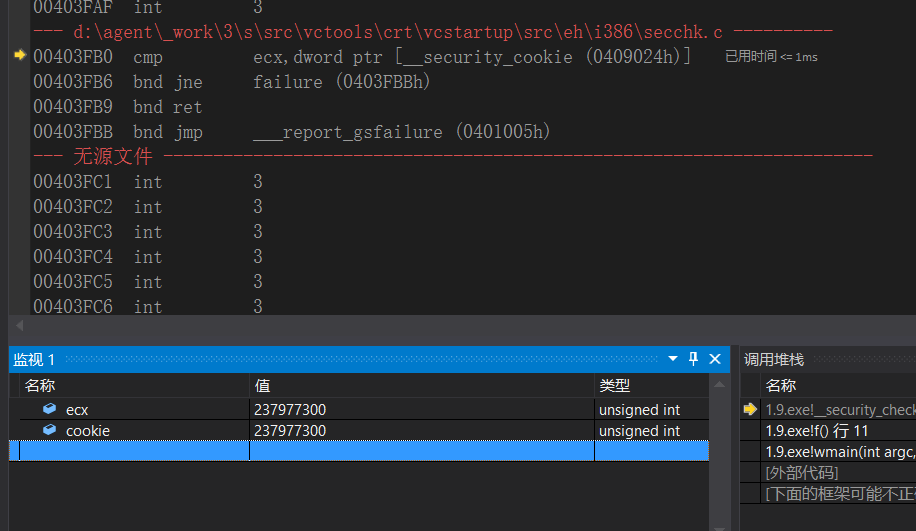
2.任何函数都只有一份ebp和esp，可以通过内存窗口发现ebp和esp永远保存最新当前函数的数值。用函数时，原函数的ebp和esp要保存起来，以便返回时恢复。

3.在栈帧布局中,局部变量两边都有0xcccccccc,以4字节来看，char型具备变量占据了4字节的最高一个字节的地方，short型占据了4字节低位的两个字节。

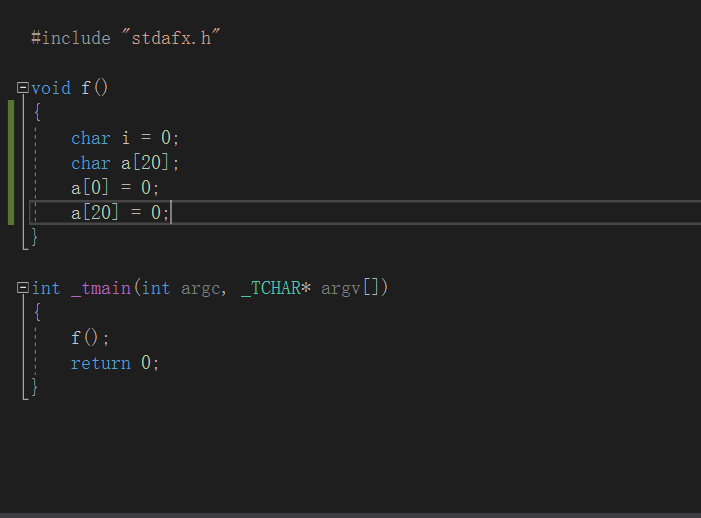
## 3.3函数中有局部数组的情况

## 代码清单13



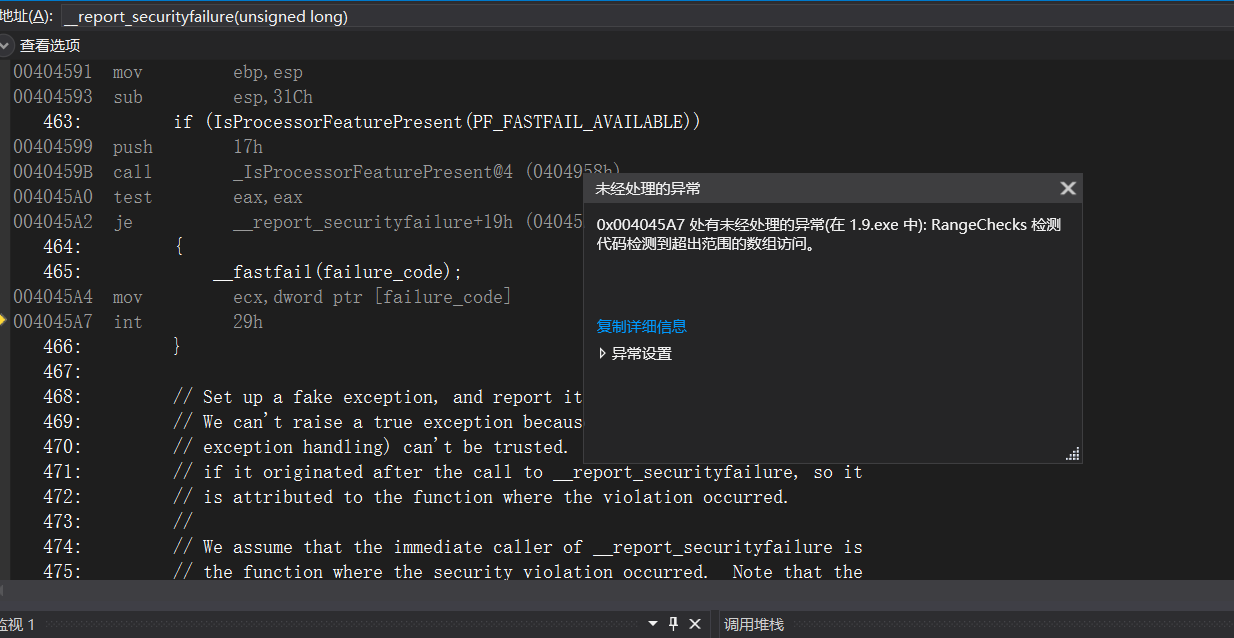


这时候ecx的值同\_\_security\_cookie一致

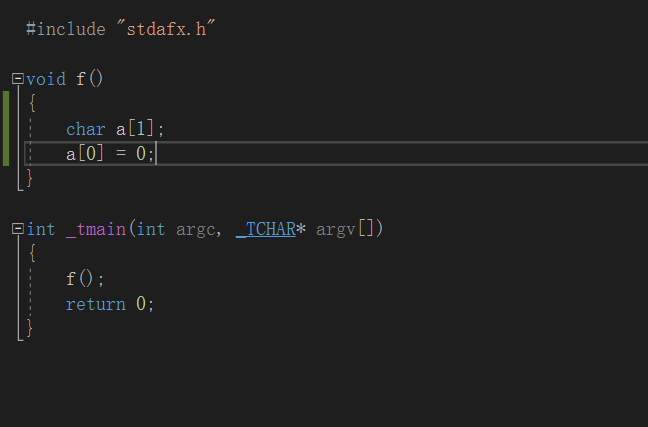


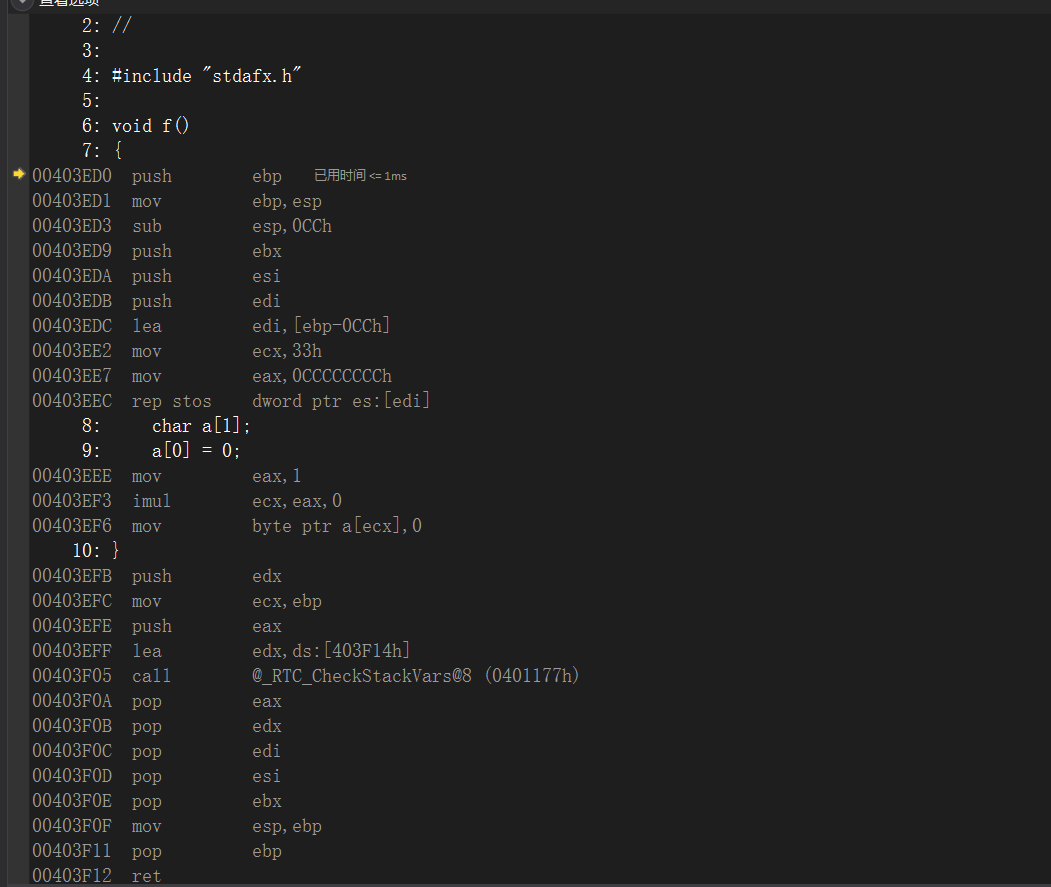
在函数添加a[20] = 0;a[20]可以造成越界写的效果

因为声明变量char a[20],访问下标是从0到19，不包括20



## 代码清单16





6: void f()

7: {

00403ED0 push ebp

;将ebp寄存器入栈

00403ED1 mov ebp,esp

;使ebp = esp

00403ED3 sub esp,0CCh

; sub是减法指令,使esp = esp – 0CCh

00403ED9 push ebx

;将ebx寄存器入栈

00403EDA push esi

;将esi寄存器压栈

00403EDB push edi

;将edi寄存器入栈

00403EDC lea edi,[ebp-0CCh]

;lea是取地址指令，该指令使edi = ebp-0CCh

00403EE2 mov ecx,33h

;使ecx = 33h

00403EE7 mov eax,0CCCCCCCCh

;使eax = 0CCCCCCCCh

00403EEC rep stos dword ptr es:[edi]

; rep是重复前缀，stos指令是将eax的值放入[edi]（es:edi所指向的地址）中，之后在这里将edi+4。rep stos 是循环指向stos，直到ecx为,每次循环ecx都会减1

8: char a[1];

9: a[0] = 0;

00403EEE mov eax,1

;使eax = 1

00403EF3 imul ecx,eax,0

;imul是有符号乘法指令，将eax与0相乘放入ecx

00403EF6 mov byte ptr a[ecx],0

;使a[0] = 0

10: }

00403EFB push edx

;将edx寄存器压栈

00403EFC mov ecx,ebp

;使ecx = ebp

00403EFE push eax

;将eax寄存器压栈

00403EFF lea edx,ds:[403F14h]

;lea是取地址指令，使edx = 403F14h, ds是数据段寄存器

00403F05 call @\_RTC\_CheckStackVars@8 (0401177h)

; 调用了\_RTC\_CheckStackVars函数。该函数的主要任务是检查局部数组是否存在越界写的问题。

00403F0A pop eax

;将eax寄存器出栈

00403F0B pop edx

;将edx寄存器出栈

00403F0C pop edi

;将edi寄存器出栈

00403F0D pop esi

;将esi寄存器出栈

00403F0E pop ebx

;将ebx寄存器出栈

00403F0F mov esp,ebp

;使esp = ebp

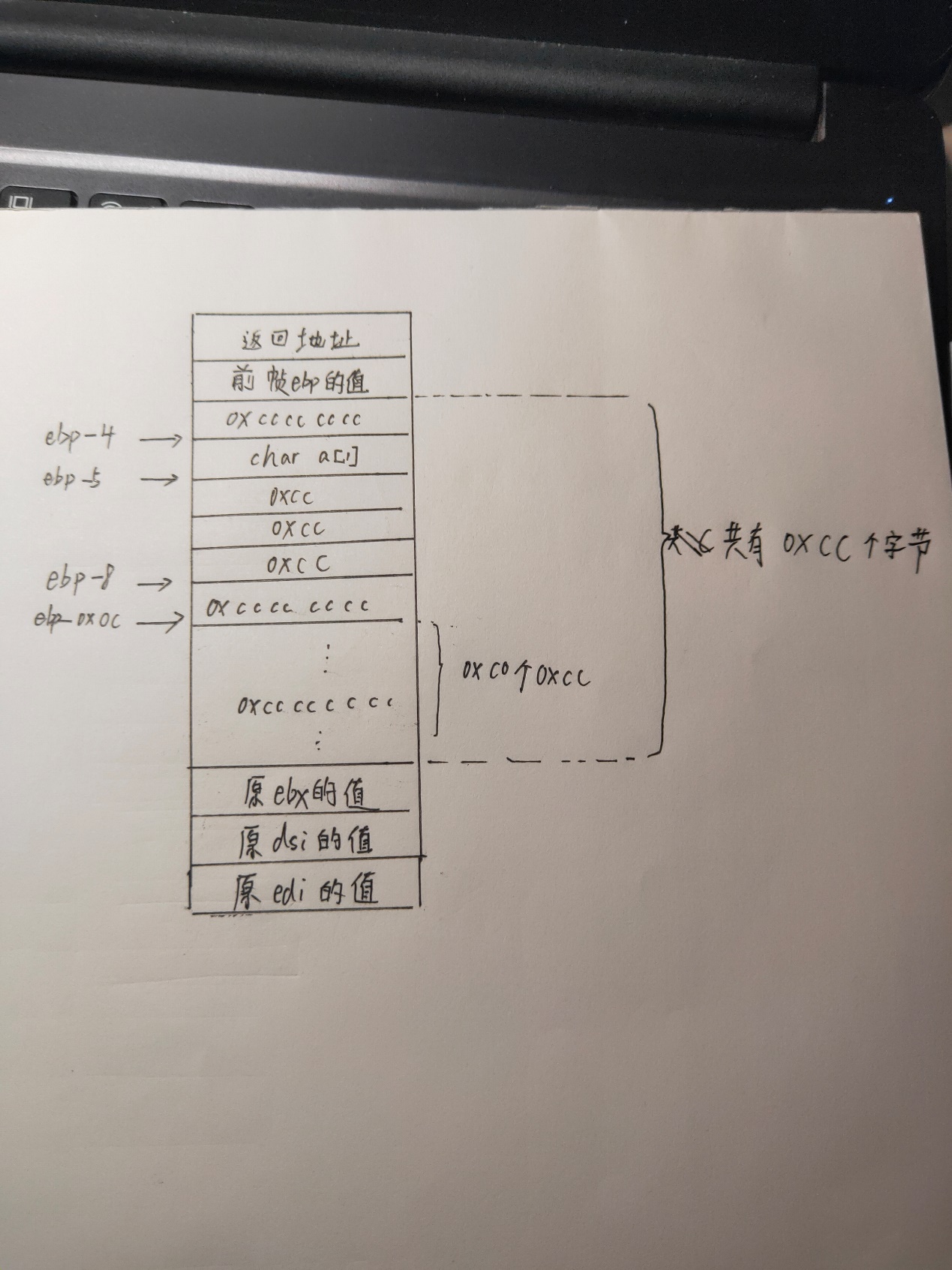
00403F11 pop ebp

;将ebp寄存器出栈

00403F12 ret

; 结束f()函数，返回到调用源

根据代码清单16，以及内存映像（调式->窗口->内存）可画出栈帧的布局，即图16。



**图16代码清单16对应的栈帧布局**