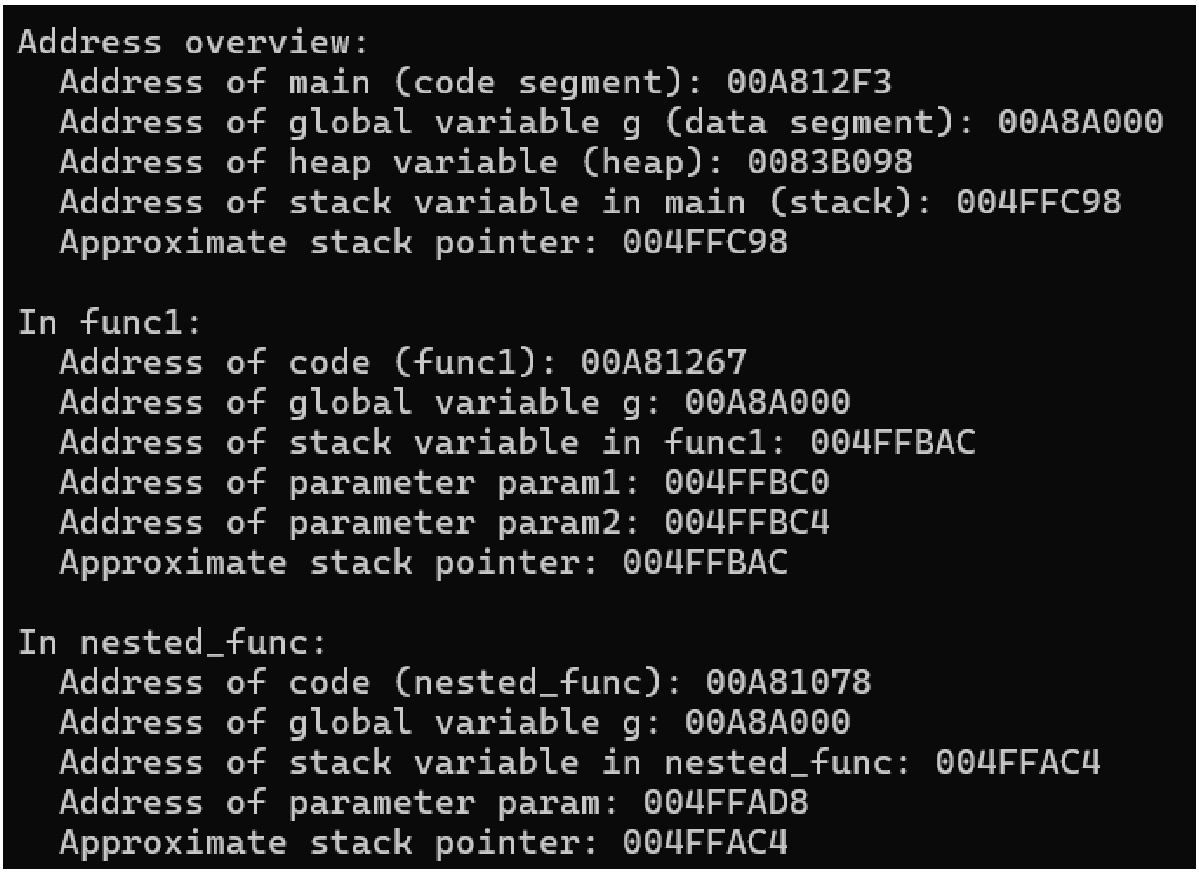
MSVC 编译器下代码各部分地址

C++源代码如下：

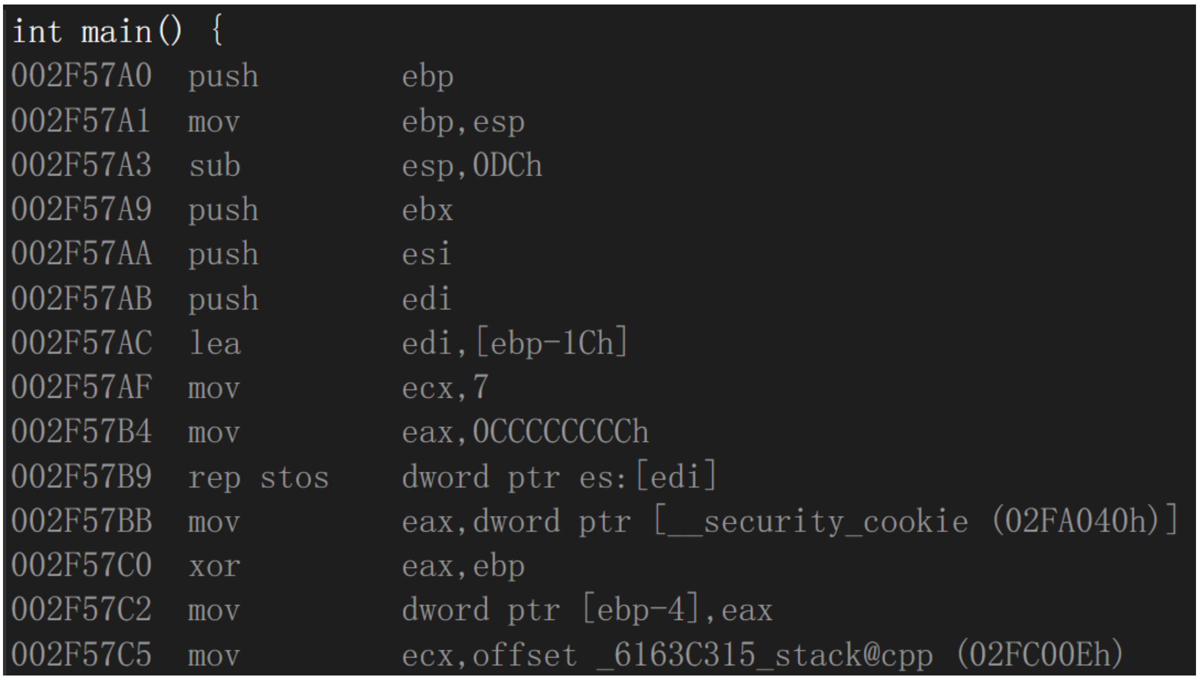
```
#include <stdio.h>
#include <stdlib.h>

int g = 42; // 全局变量，位于数据段（.data 或 .bss）

void nested_func(int param) {
    int nested_stack_var = 0; // 局部变量，位于栈区
    printf("In nested_func:\n");
    printf("  Address of code (nested_func): %p\n", (void*)nested_func);
    printf("  Address of global variable g: %p\n", (void*)&g);
    printf("  Address of stack variable in nested_func: %p\n",
(void*)&nested_stack_var);
    printf("  Address of parameter param: %p\n", (void*)&param);
    printf("  Approximate stack pointer: %p\n\n", (void*)&nested_stack_var);
}

void func1(int param1, int param2) {
    int func1_stack_var = 0; // 局部变量，位于栈区
    printf("In func1:\n");
    printf("  Address of code (func1): %p\n", (void*)func1);
    printf("  Address of global variable g: %p\n", (void*)&g);
    printf("  Address of stack variable in func1: %p\n",
(void*)&func1_stack_var);
    printf("  Address of parameter param1: %p\n", (void*)&param1);
    printf("  Address of parameter param2: %p\n", (void*)&param2);
    printf("  Approximate stack pointer: %p\n\n", (void*)&func1_stack_var);

    // 调用嵌套函数
    nested_func(param1 + param2);
}

int main() {
    printf("Address overview:\n");

    // 打印代码段起始地址
    printf("  Address of main (code segment): %p\n", (void*)main);

    // 打印全局变量（数据段）
    printf("  Address of global variable g (data segment): %p\n", (void*)&g);

    // 打印堆区地址
    void* heap_var = malloc(1);
    printf("  Address of heap variable (heap): %p\n", heap_var);
    free(heap_var);

    // 打印栈区地址
    int main_stack_var = 0;
    printf("  Address of stack variable in main (stack): %p\n",
(void*)&main_stack_var);
    printf("  Approximate stack pointer: %p\n\n", (void*)&main_stack_var);
    // 调用一级函数，传递两个参数
    func1(10, 20);

    return 0;
}
```



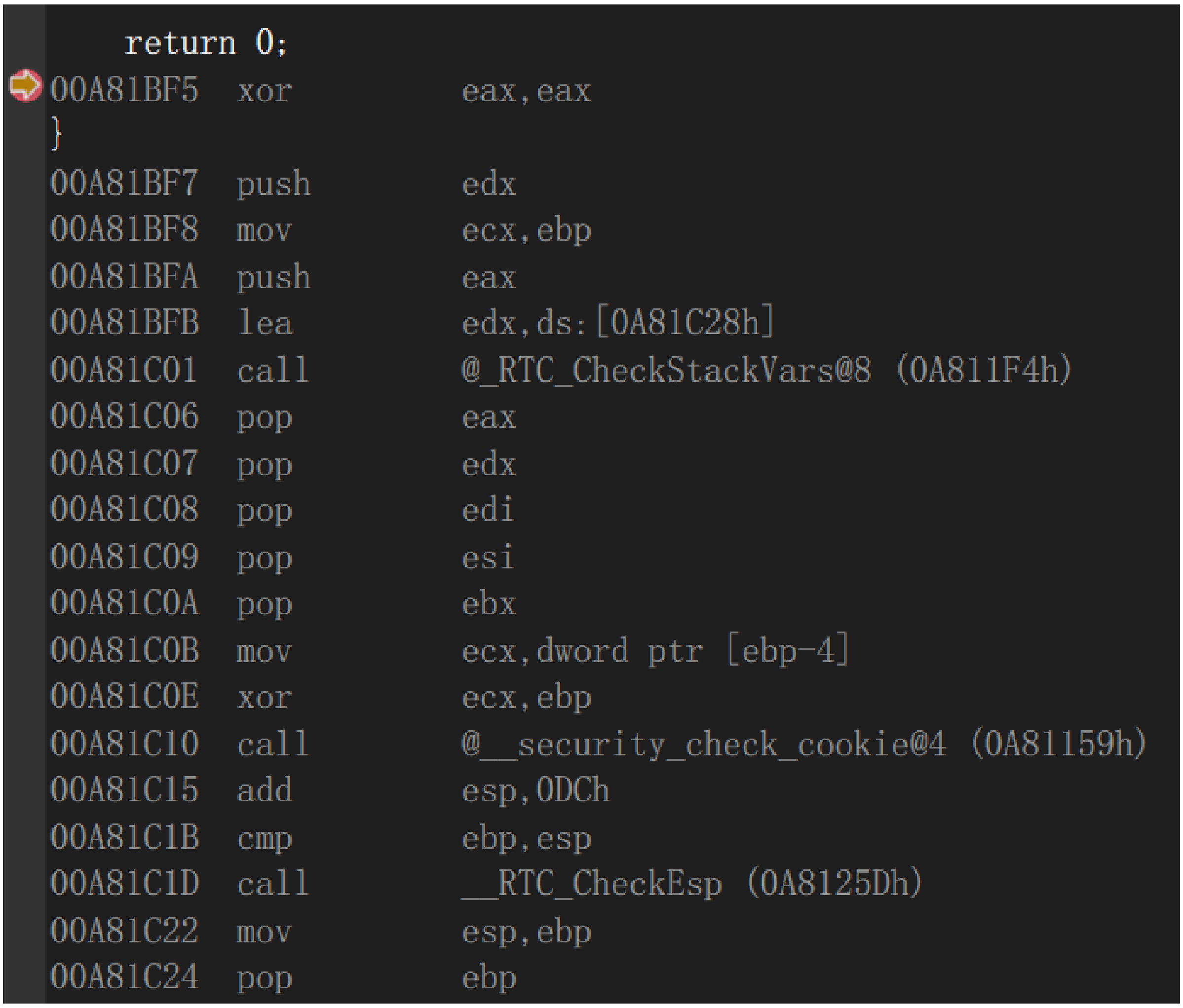
```
Address overview:
  Address of main (code segment): 00A812F3
  Address of global variable g (data segment): 00A8A000
  Address of heap variable (heap): 0083B098
  Address of stack variable in main (stack): 004FFC98
  Approximate stack pointer: 004FFC98

In func1:
  Address of code (func1): 00A81267
  Address of global variable g: 00A8A000
  Address of stack variable in func1: 004FFBAC
  Address of parameter param1: 004FFBC0
  Address of parameter param2: 004FFBC4
  Approximate stack pointer: 004FFBAC

In nested_func:
  Address of code (nested_func): 00A81078
  Address of global variable g: 00A8A000
  Address of stack variable in nested_func: 004FFAC4
  Address of parameter param: 004FFAD8
  Approximate stack pointer: 004FFAC4
```

如上图可以看到，地址从小到大排序为：**栈，堆，代码段，数据段**。此外，栈区地址是**向低地址增长**的。

在Visual Studio的项目下，分配给程序的栈空间大小默认为**1MB**，可以在项目设置中调整大小。



```
int main() {
002F57A0  push        ebp
002F57A1  mov         ebp,esp
002F57A3  sub         esp,0DCh
002F57A9  push        ebx
002F57AA  push        esi
002F57AB  push        edi
002F57AC  lea         edi,[ebp-1Ch]
002F57AF  mov         ecx,7
002F57B4  mov         eax,0CCCCCCCCh
002F57B9  rep stos    dword ptr es:[edi]
002F57BB  mov         eax,dword ptr [__security_cookie (02FA040h)]
002F57C0  xor         eax,ebp
002F57C2  mov         dword ptr [ebp-4],eax
002F57C5  mov         ecx,offset _6163C315_stack@cpp (02FC00Eh)
```



```
    return 0;
00A81BF5  xor         eax,eax
}
00A81BF7  push        edx
00A81BF8  mov         ecx,ebp
00A81BFA  push        eax
00A81BFB  lea         edx,ds:[0A81C28h]
00A81C01  call        @_RTC_CheckStackVars@8 (0A811F4h)
00A81C06  pop         eax
00A81C07  pop         edx
00A81C08  pop         edi
00A81C09  pop         esi
00A81C0A  pop         ebx
00A81C0B  mov         ecx,dword ptr [ebp-4]
00A81C0E  xor         ecx,ebp
00A81C10  call        @__security_check_cookie@4 (0A81159h)
00A81C15  add         esp,0DCh
00A81C1B  cmp         ebp,esp
00A81C1D  call        __RTC_CheckEsp (0A8125Dh)
00A81C22  mov         esp,ebp
00A81C24  pop         ebp
```

通过函数开关和结尾反汇编可以看到，每个函数都被分配了一部分**“栈帧”**，用于专门存储该函数的**参数，返回地址，基址**等信息。这个栈帧同时也是栈空间的一部分。