Practical Foundations of Debugging

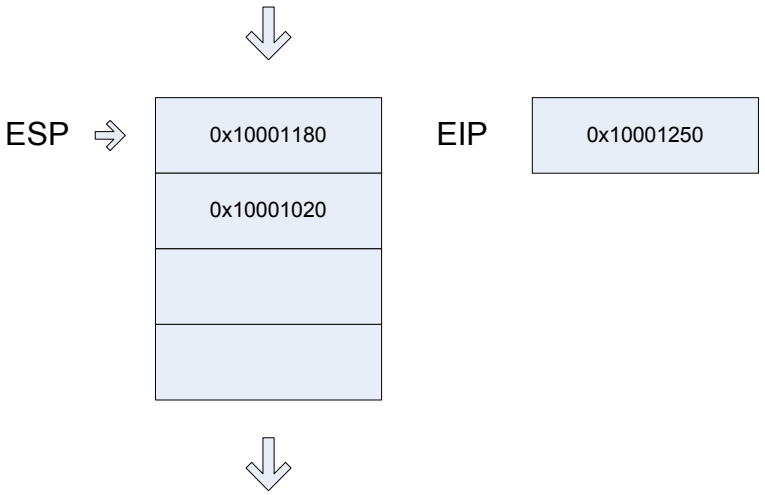
Chapter 6

Frame Pointer and Local Variables – Part 1

Designed, written and illustrated by Dmitry Vostokov

Call stack

func 0x10001000 – 0x10001100
func2 0x10001101 – 0x10001200
func3 0x10001201 – 0x10001300
(.MAP file or .PDB file)

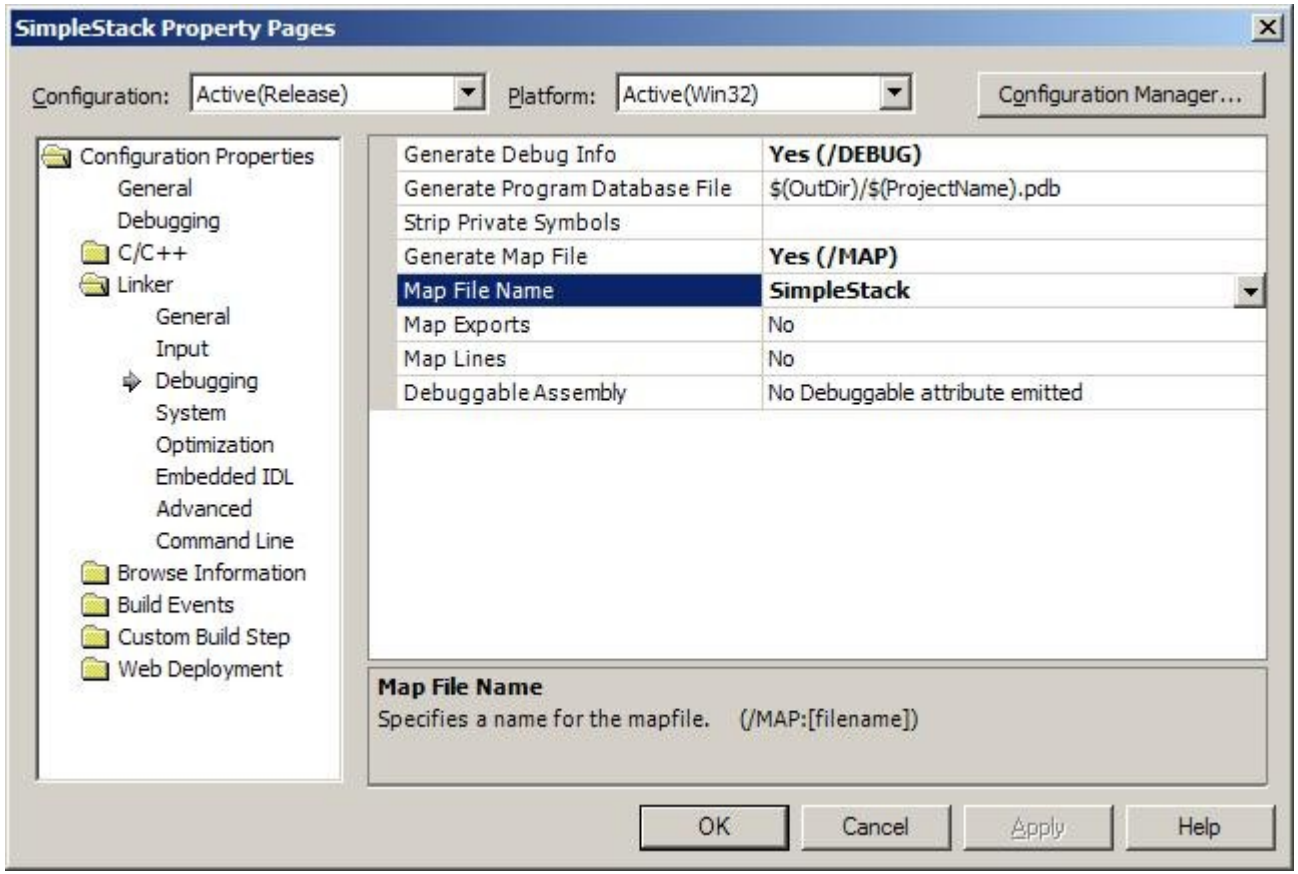



```
func3
func2
func
```

```
void func()
{
    func2();
}

void func2()
{
    func3();
}
```

Specifying .MAP file (VC++.NET project settings)



.MAP file structure

```
SimpleStack

Timestamp is 40e84445 (Sun Jul 04 18:54:13 2004)

Preferred load address is 00400000

Start         Length     Name                   Class
0001:00000000 00003b88H .text                   CODE
…
…
0003:00000030 00000274H .data                   DATA
0003:000002c0 00000598H .bss                    DATA

Address         Publics by Value              Rva+Base     Lib:Object

…
0001:00000000       _func                      00401000 f   func.obj
0001:00000010       _func2                     00401010 f   func2.obj
0001:00000020       _func3                     00401020 f   func3.obj
0001:00000030       _main                      00401030 f   SimpleStack.obj
…
```

Stack in WinDbg

```
0:000> r
eax=00321468 ebx=7ffdf000 ecx=00000001 edx=7ffe0304 esi=00000a28 edi=00000000
eip=00401023 esp=0012fecc ebp=0012fecc iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
SimpleStack!func3+0x3:
00401023 cc              int     3

0:000> dds esp
0012fecc  0012fed4
0012fed0  00401018 SimpleStack!func2+0x8
0012fed4  0012fedc
0012fed8  00401008 SimpleStack!func+0x8
0012fedc  0012fee4
0012fee0  00401038 SimpleStack!main+0x8
0012fee4  0012ffc0
0012fee8  004011d4 SimpleStack!mainCRTStartup+0x173
0012feec  00000001
0012fef0  00321418
```

Stack usage

- Stack is used to store return address for CALL instructions
- Stack is used to pass parameters to functions and store local variables
- **Stack is used to save and restore registers temporarily**

Using stack to save registers

```
mov  eax, 10
mov  edx, 20
push     eax
push     edx
imul     eax, edx
mov  dword ptr [result], eax
pop      edx  ; popping in reverse order
pop      eax ; stack is LIFO
```

Functions and saving registers

- Group 1: EBX, ESI, EDI, EBP
- Group 2: EAX, EDX, ECX

```
// func.c
void func()
{
   // may use group 1 registers
   // saves group 2 registers if uses values in them after
   // calling func2()
   func2();
}

// func2.c
void func2()
{
   // saves group 1 registers if it uses them
   // freely uses group 2 registers
}
```

Register review

General purpose

EAX
EBX
ECX
EDX

Instruction Pointer

EIP

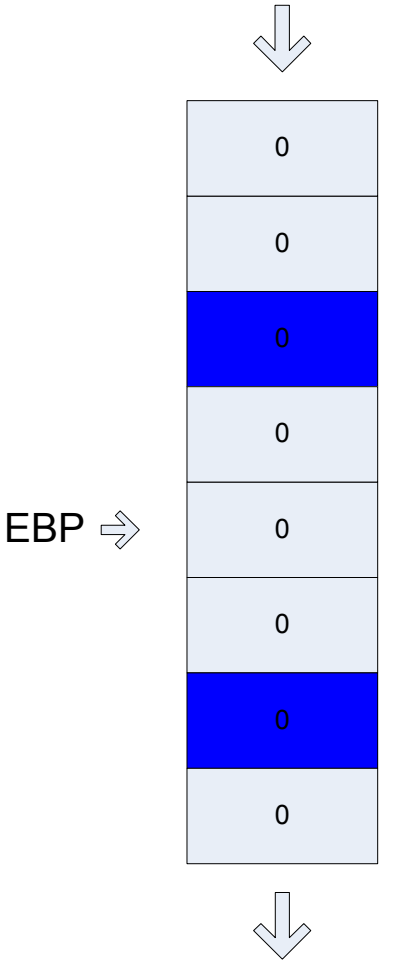
Stack Pointer

ESP

Stack Frame Pointer (Base Pointer)

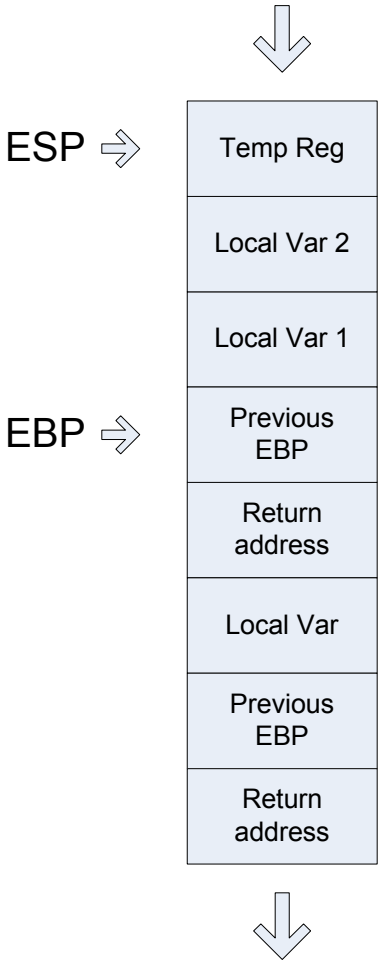
EBP

Addressing array elements



	Address of element	Value of element
0x10001000	EBP-10	[EBP-10]
0x10001004	EBP-C	[EBP-C]
0x10001008	EBP-8	[EBP-8]
0x1000100C	EBP-4	[EBP-4]
0x10001010	EBP	[EBP]
0x10001014	EBP+4	[EBP+4]
0x10001018	EBP+8	[EBP+8]
0x1000101C	EBP+C	[EBP+C]

Stack structure (no function parameters)



[EBP] contains previous EBP

[EBP-4] contains Local Var 1 (DWORD)

[EBP-8] contains Local Var 2 (DWORD)

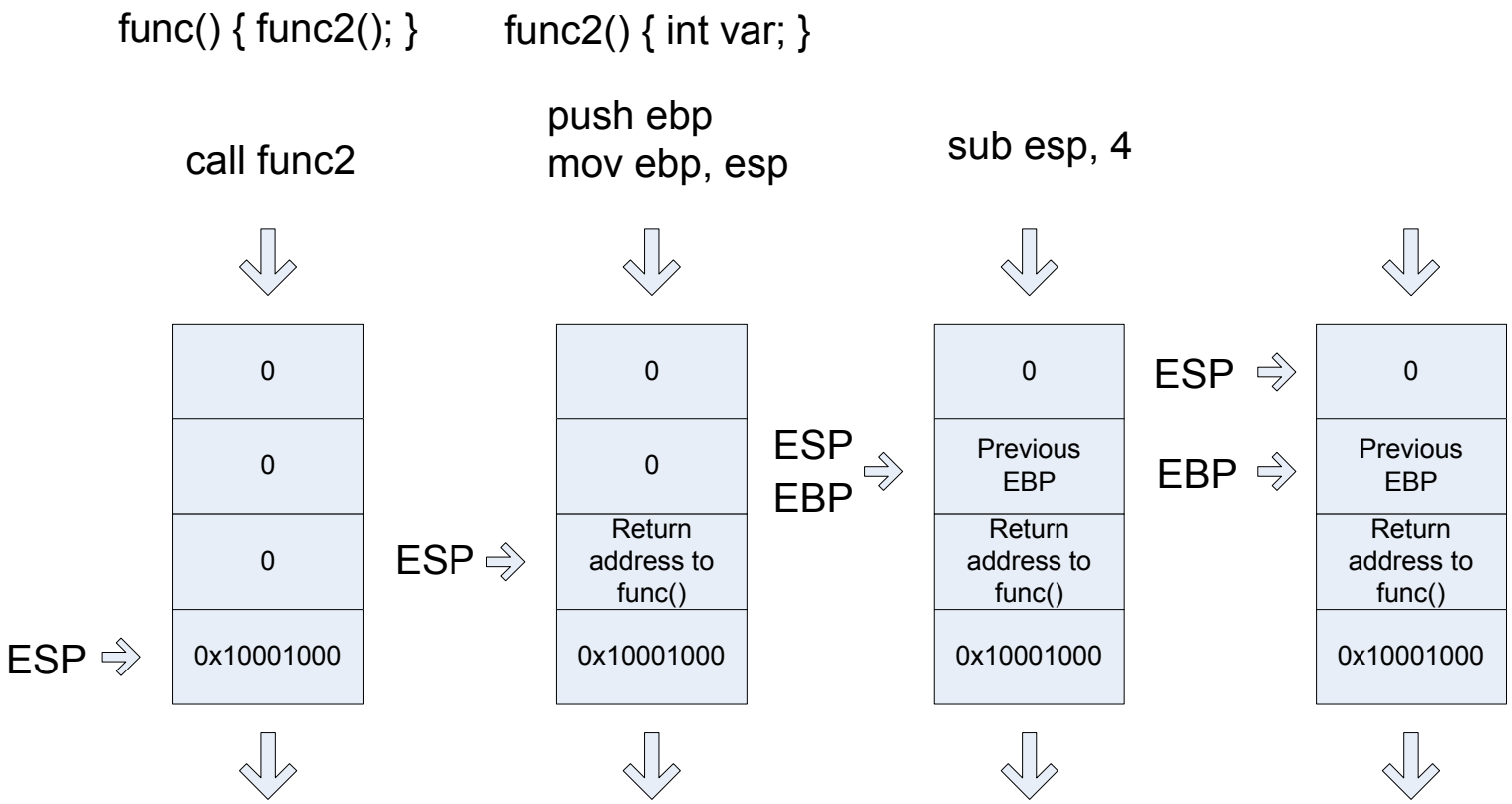
```
void func()
{
  int var1, var2;
}
```

Stack in WinDbg (no local variables and function parameters)

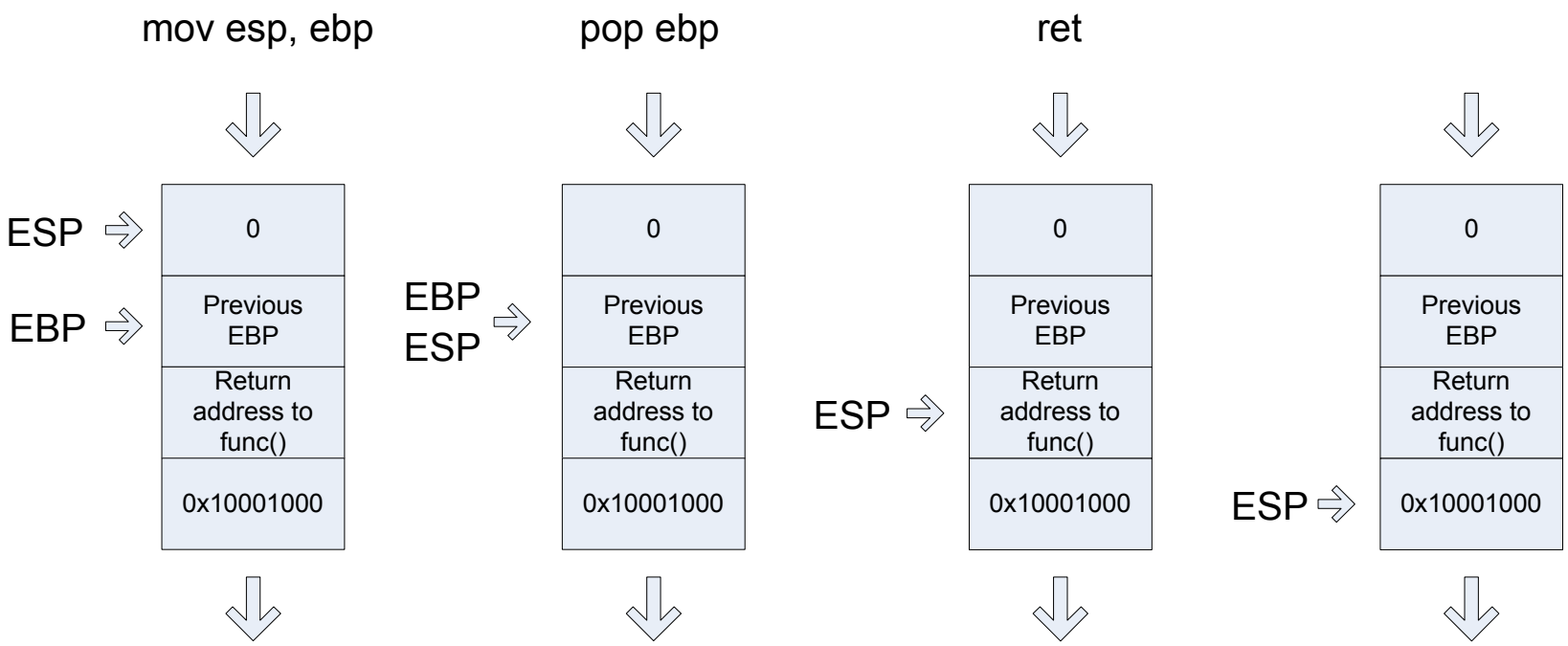
```
0:000> r
eax=00321468 ebx=7ffdf000 ecx=00000001 edx=7ffe0304 esi=00000a28 edi=00000000
eip=00401023 esp=0012fecc ebp=0012fecc iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
SimpleStack!func3+0x3:
00401023 cc              int     3

0:000> dds esp
0012fecc  0012fed4
0012fed0  00401018 SimpleStack!func2+0x8
0012fed4  0012fedc
0012fed8  00401008 SimpleStack!func+0x8
0012fedc  0012fee4
0012fee0  00401038 SimpleStack!main+0x8
0012fee4  0012ffc0
0012fee8  004011d4 SimpleStack!mainCRTStartup+0x173
0012feec  00000001
0012fef0  00321418
```

Function prolog



Function epilog



“Arithmetical” project

```
int main(int argc, char* argv[])
{
   int a, b;

   a = 1;
   b = 1;

   b = b + a;
   ++a;
   b = a * b;

   return 0;
}
```

```
LocalVariables!main:
push    ebp                         ; establishing stack frame
mov     ebp,esp                     ;
sub     esp,0xd8                    ; creating stack frame for locals
push    ebx                         ; saving registers that might be used
push    esi                         ;   outside
push    edi                         ;
lea     edi,[ebp-0xd8]              ; getting lowest address of stack frame
mov     ecx,0x36                    ; filling stack frame with 0xCC
mov     eax,0xcccccccc              ;
rep     stosd                       ;
mov     dword ptr [ebp-0x8],0x1     ; [a] = 1       ([ebp-0x8])
mov     dword ptr [ebp-0x14],0x1    ; [b] = 1       ([ebp-0x14])
mov     eax,[ebp-0x14]              ; eax := [b]
add     eax,[ebp-0x8]               ; eax := eax + [a]
mov     [ebp-0x14],eax              ; [b] := eax    (b = b + a)
mov     eax,[ebp-0x8]               ; eax := [a]
add     eax,0x1                     ; eax := eax + 1
mov     [ebp-0x8],eax               ; [a] := eax    (++a)
mov     eax,[ebp-0x8]               ; eax := [a]
imul    eax,[ebp-0x14]              ; eax := eax * [b]
mov     [ebp-0x14],eax              ; [b] := eax    (b = a * b)
xor     eax,eax                     ; eax := 0      (return value)
pop     edi                         ; restoring registers
pop     esi                         ;
pop     ebx                         ;
mov     esp,ebp                     ; restoring previous stack pointer
pop     ebp                         ; restoring previous stack frame
ret                                 ; return 0
```