

Assembly programming

Welcome back! This is day 3!

© Government of Canada

This document is the property of the Government of Canada. It shall not be altered, distributed beyond its intended audience, produced, reproduced or published, in whole or in any substantial part thereof, without the express permission of CSE.



Communications
Security Establishment

Centre de la sécurité
des télécommunications

PAGE

Canada 

What did we do yesterday?



What exactly is a calling convention?



About cdecl, how are parameters passed to the function?



About cdecl, what registers are available for the programmer within a function? What happens with other registers?



Let's say a function with 2 (32 bits) parameters has just been called and EIP is on the very first instruction of the function.

How is the stack currently looking like?



What happens when a system call is made?



What is the calling convention for system calls on linux x86?



What data structure is used here? How does it work?

```
5 insert: ;insert(**entryNode, *newNode)
6     push ebp
7     mov ebp, esp
8
9     mov eax, [ebp + 8]           ; Getting pointer
10                                ; to entry node
11     mov ecx, [ebp + 0xC]
12     cmp DWORD [eax], 0
13     jnz evaluateInsert
14     mov [eax], ecx             ; Node inserted
15     jmp jobDone
16 evaluateInsert:
17     mov edx, [eax]
18     mov ecx, [ecx]
```

```
19     cmp ecx, [edx]
20     jle le_Insert
21     lea eax, [edx + 4]
22     jmp attempt_insert
23 le_Insert:
24     lea eax, [edx + 8]
25 attempt_insert:
26     mov ecx, [ebp + 0xC]
27     push ecx
28     push eax
29     call insert
30     add esp, 0x08
31 jobDone:
32     pop ebp
33     ret
```

How is OOP organized when translated to assembly?

