### **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.

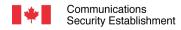




## What did we do yesterday?



## 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?

```
insert: ;insert(**entryNode, *newNode)
       push ebp
       mov ebp, esp
       mov eax, [ebp + 8]
10
11
       mov ecx, [ebp + 0 \times C]
12
       cmp DWORD [eax], 9
13
       inz evaluateInsert
14
       mov [eax], ecx
15
       jmp jobDone
16
     evaluateInsert:
17
       mov edx, [eax]
       mov ecx, [ecx]
```

```
cmp ecx, [edx]
       ile le Insert
       lea eax, [edx + 4]
       jmp attempt insert
     le Insert:
       lea eax, [edx + 8]
     attempt insert:
       mov ecx, [ebp + \Theta \times C]
       push ecx
       push eax
       call insert
       add esp, 0x08
     jobDone:
32
       pop ebp
```

Communications

# How is OOP organized when translated to assembly?

