**CS4238 Lab: IDA and Code Structure**

The **goal** of this lab is to get familiar with **basic utilities in IDA** and **code structures** in PE files.

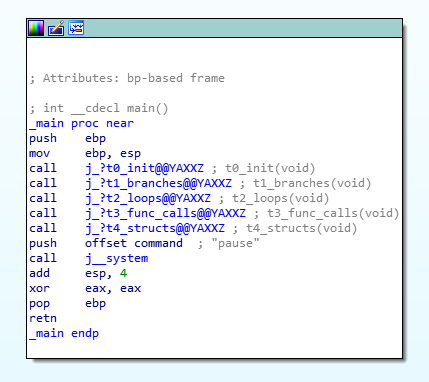
**Lab Set-up**

You will need the IDA Free for this lab (already integrated into the Flare VM). A CodeStructureExample.exe file will be used for analysis (see this lab’s attachments).

**A. IDA Free**

\* In lectures and labs, we mainly focus on IDA. If you prefer other disassemblers such as Gindra or Binary Ninja, please feel free to use them in assignments and your group project.

1. Load the CodeStructureExample.exe into IDA
2. Then load the [PDB](https://en.wikipedia.org/wiki/Program_database#:~:text=Program%20database%20(PDB)%20is%20a,from%20source%20files%20during%20compilation.) file in File->Load File->PDB File…
3. Search for function “\_main” in the left (click arbitrary function and press Ctrl+F), then you should be able to see the following view.



**B. Branching**

The goal of this task is to view a **branching** in IDA graph view.

1. Go to the function *j\_?t1\_branches@@YAXXZ* by double-clicking it
2. *j\_?t1\_branches@@YAXXZ* is a jump thunk of *?t1\_branches@@YAXXZ*

This is inserted by MSVC Compiler in [incremental linking](https://docs.microsoft.com/en-us/cpp/build/reference/incremental-link-incrementally?view=msvc-160).

1. Go to *?t1\_branches@@YAXXZ*. There are two functions inside. One is for an *if* statement, while another is for *case-switch* statements.
2. Go to *if\_statement\_func*.
   1. Is *current\_sec* a global variable, a variable on the heap, or a variable on the stack?
   2. How many branches are there?
   3. What is the branch condition? (*[jz](https://www.aldeid.com/wiki/X86-assembly/Instructions/jz)* [instruction](https://www.aldeid.com/wiki/X86-assembly/Instructions/jz))
3. Go to *switch\_statements\_func*.
   1. How many cases are there in total?
   2. Can you distinguish this switch statement from nested *if* statements?
   3. (Advanced) How can this switch statement be optimized to avoid comparing?

**C. Loops**

The task aims to view a **loop** in IDA graph view.

1. For loop: Go to *?* *for\_loop\_func@YAXXZ*.
   1. Which edge in the graph view implies this function contains a loop?
2. While loop, do-while loop, goto-loop
   1. *while\_loop\_func, do\_while\_loop\_func, goto\_loop\_func*
   2. Can you tell the difference by viewing their structures?

**D. Function calls**

The task aims to view different styles of function calls in IDA graph view.

1. Direct call and indirect call
   1. Goto *t3\_func\_calls* and locate a direct invocation to *test\_call* and the indirect one.
2. *cdecl* and *stdcall*
   1. Observe *cdel\_call* and *std\_call*. What is the difference? See [callee cleanup](https://en.wikipedia.org/wiki/X86_calling_conventions#Callee_clean-up).
3. *Fastcall*
   1. Observe *fast\_call*. How are arguments passed?

**E. Data structures**

1. Check *arr\_n\_str.* How are arrays, c strings, C++ strings structured?
2. Check *structures* and *classes.* What’s the difference between structures and classes?