Binary Lab

Description

The main aim of this lab is to understand provided assembly code and generate C code that corresponds to that assembly. You are provided with a C file (bin-lab.c) that contains a filled main() function and empty five functions f1(), f2(), f3(), f4(), and f5(). Your job is to read the object files (more on this shortly), understand the assembly of each function, and write the corresponding C code of each of the 5 functions in the 5 C files. The main goal is that if we compile and execute your code, it must generate the same output as the original executable ./bin-lab-ref (provided).

Note: The fact that you have one function in each of the provided empty C files does not mean that this function does not call another function. For example, when you examine the code for f1() you may find that f1() calls another function b1(). In this case, you need to include b1() in your code too.

Steps:

1. Download the file **bin-lab.c**
2. Download the executable **bin-lab-ref**
3. Execute bin-lab-ref to see the different outputs. You just need to type: **./bin-lab-ref**
4. You need to fill-in the blanks of file bin-lab.c such that when executed, its output is similar to ./bin-lab-ref
5. You compile your code with the following command:

   ```
   gcc -o bin-lab -fomit-frame-pointer bin-lab.c
   ```

How to work on this lab?

Your main source of help is a tool called objdump. Type:

```
objdump - d ./bin-lab-ref > output.txt
```

Open the file output.txt with any text editor and you will find the assembly of the code. Your best starting point is to look for the labels f1, f2, f3, f4, and f5 on at a time and read the assembly of each corresponding function. There may be extra assembly code, related to other libraries; you do not need to worry about them.

Note: The aim is to make your executable generate the same output as ./bin-lab-ref but the assembly generated from your code does NOT need to match the assembly generated from the original code as this is not easily achievable.

Grading

There are 5 functions to implement. Each one is worth 20 points, for a total of 100 points.

Submission

You just need to submit bin-lab.c through NYU classes.

Important Notes

If, by trying to execute ./bin-lab-ref you receive “command not found” or the like, then type:
chmod 777 ./bin-lab-ref
and then re-try to execute./bin-lab-ref

Some instructions that we did not cover in class but you may find in the object files (not included in the exam though):

- **repz**: This is used due to some strange behavior with old AMD K8 regarding its branch prediction. To make long story short, neglect it! So if you see, for example, repz retq assume it is retq only.
- **nopl**: this is a no-operation instruction. That is, do nothing instruction. It can take argument but does nothing with it. It is mainly used as a delay instruction waiting for an event to happen, such as incrementing rip register (more in-depth explanation will take us a bit deeper into hardware). Also, it is used as “padding” in the code to make following instruction start at specific address.

Enjoy!