Problem Set 1 - ECE391 Fall 2021

Logistics

Problem Set 1 is due **Tuesday 9/7 at 05:59:59 PM** in the master branch. Only one person per group should have a `partners.txt` with **all** the netids of your partners on separate lines; In order to finishe this PS, you need to form a group of **at least 4 students**, no cap on maximum number of collaborators in the same group. If your name appears in 2 or more different groups, you will get a 0 for PS1. An example `partners.txt` would look like this if `yanmiao2` was submitting the group's solution.

> yanmiao2
>
> ch5
>
> sjeon12
>
> hc19
>
> xiangl5

Note that if your netid is part of more than one `partners.txt`, you will recieve a 0 for the problem set.

You can attempt this problem set on the Class VM (devel) or on any EWS Linux computer.

Problem 1: GNU Debugger (GDB) (5 pt)

Please write the command(s) you should use to achieve the following tasks in GDB.

1. Show the value of variable "test" in hex format.
2. Show the top four bytes on your stack word-by-word, e.g. it should look something like this "0x0102 0x0304", NOT "0x01020304".
3. Check all register values.
4. Set a breakpoint at "ece.c" line 391.
5. Connect to the test_(no)debug vm in the lab setup.

Please write your solution in p1_soln.txt with answers to each question on a separate line. For example, your p1_soln.txt should be of the form

> answer 1
>
> answer 2
>
> ...
>
> answer 5

Problem 2: C to Assembly Translation (10 pt)

Write x86 assembly code for the body of the `merge` function found in `merge.c`. Make sure to set up and tear down the stack frame as well as save and restore any callee/caller-saved registers yourself (if you use them). Assume caller-saved registers are saved prior to the `merge_asm` function being called for the first time. Include comments (but don't overdo it!) in your assembly code to show the correspondence between the C code and your x86 code.

Also note that:

1. The `merge_asm` function in `merge_asm.S` is partially filled out for you, your job is to complete the function.
2. Please make sure your code and comments are easy to read. We reserve the right to take points off if your answer is too hard to follow.
3. You must synthesize your answer without the help of a computer. For example, you may not write the C code and then use the compiler to disassemble it. If you are caught doing this, you will receive a 0.
4. There might not be a one-to-one mapping between C and Assembly codes, but you have to translate using the exact same algorithm(E.g. if the provided merge function uses recursion method, you cannot translate using iteration method). If you failed to do this, you will receive a 0.
5. You must write your solution in `p2/merge_asm.S` and submit it through gitlab.

Problem Constraints:

- You can assume all input txt files are valid(E.g. only numbers, no letters, no overflow)
- You can assume the 2 linked lists provided to you are already sorted separately.
- You can assume all numbers in the linked list are non-negative integers.

To build the code (no debug flag): `$ make clean && make`

To run the code: `$ ./merge ./1_list.txt ./2_list.txt`

To build the code (debug flag): `$ make clean && make debug`

To run the code (debug): `$ gdb --args ./merge ./1_list.txt ./2_list.txt`

Problem 3: Assembly to C Translation (10 pt)

Write a C function equivalent to the x86 assembly function, `mystery_asm` found in `mystery_asm.S`.

1. Please make sure your code and comments are easy to read. We reserve the right to take points off if your answer is too hard to follow.
2. There might not be a one-to-one mapping between C and Assembly codes, but you must **translate** your code, a functionally equivalent algorithm with a different structure will recieve a 0.(E.g. if the provided function uses recursion method, you cannot translate using iteration method)
3. You must write your solution in `p3/mystery.c` and submit it through gitlab.
4. Note this problem is just for practise use for ECE391 students. So it might not be the most organized/optimized.

Problem Constraints:

- You can assume `x` and `y` are non-negative integers.
- You don't need to worry about overflow situations.

To build the code (no debug flag): `$ make clean && make`

To run the code: `$ ./mystery ./1_input.txt`

To build the code (debug flag): `$ make clean && make debug`

To run the code (debug): `$ gdb --args ./mystery ./1_input.txt`