← Lab 3: Calling conventions and debugging

Finish by midnight on Friday, 9/28

This lab is a little different. I am giving you a broken program, and you will fix it. We learned everything you need to know to fix all the bugs with this program.

Also, I will be giving a solution to this lab after it's due, so you can see how well you did.

The starting point

Download this file and rename it `username_lab3.asm` like the other labs.

Using an external editor

You have probably discovered by now that the MARS editor is kind of terrible. This is not a big problem, because it works well with external editors.

Find a MIPS syntax highlighting plugin for your editor of choice (I know they exist for sublime and atom). Then open the file in your editor and in MARS **at the same time.**

Now, when you save your file in your editor, you can switch to MARS to assemble and run.

> You don't need to close and reopen the file in MARS! Just keep it open. It will see the changes when you assemble.

What it does

> Turn on Settings > Popup dialog for input syscalls.

When you first assemble and run, it will output something like this:

```
0
**** user input : 1
10
**** user input : 1
```

and then it will loop infinitely. This is bad.

Your task

There are 6 things to fix in this program. Read the comments in `main` to see what is wrong, and to see the pseudocode of what it *should* be doing.

The problems here mostly have to do with register usage and the calling convention. Some of them are very obvious, and some will require some investigation.

Some more info on the problems:

1. A simple, but common mistake in register use.
    - The problem is in `main`.
2. The right registers are being used, but the value isn't making it through.
    - The problem is in `main`.
3. **Bookmarks…**
    - The problem is in `read_int_plus_one`.
4. A violation of the saved/unsaved registers contract.
    - The problem is in `main`.
5. Not actually a calling convention/register usage issue! See below.
6. Another violation of the saved/unsaved registers contract.
    - The problem is in `print_array`.
    - You will have to change *and* add code to fix this correctly.

Debugging a crashed program

When you get to problem 5, the issue is that the program crashes. Debugging a program crash in assembly is very tricky, but there are things to make it easier.

- **Read the error message.**
    - It shows the line of code where the error happened.
    - For invalid memory accesses, it shows the address that is invalid (like "address out of range 0x00000004")
- **Take a step back… literally:**

- When the program crashes, use the step back button to "undo" instructions.
- That will let you inspect the contents of the registers right before the crash happened.
- Then you can step forward again to see the crash happen in "slow motion."
- This usually makes it pretty obvious why it's crashing.

Correct output

When all the bugs are fixed, the program will output the following:

```
45
**** user input : 1
1
**** user input : 1
5
5
101
201
301
401
501
601
701
801
901
1001
```

Submitting

Make sure your file is named `username_lab3.asm`, like `jfb42_lab3.asm`.

Submit here.

Drag your asm file into your browser to upload. **If you can see your file, you uploaded it correctly!**

You can also re-upload if you made a mistake and need to fix it.