CS 590: Topics in Computer Science
Assignment 08: MIPS Assembly Instructions and MIPS GCC

Load MIPS_ex1.s into MARS and assemble the code. Assume that:
`fib[0] = 0; fib[1] = 1; fib[n] = fib[n-1] + fib[n-2]`
Use Help (icon with the question mark) in order to answer the following questions about how to use MARS.

What do the .data, .word, .text directives mean (i.e., what do you put in each section)?
How do you set a breakpoint in MARS? Set breakpoint on line 15
After your program stops because of a breakpoint, how do you continue to execute your code?
How do you step through your code?
How can you find out the contents of a register? How do you modify the value of a register?
At what address is n stored in memory? Calculate the 13th fib number by modifying this memory location.
line 16 and 18 use syscall instruction. What is it and how do you use it? (hint: syscall inside Help!))

Exercise 2: A short MIPS program

Write a piece of MIPS code that, given values in `$s0` and `$s1`, put into the `$t?` registers the following:

```
$t0 = $s0
$t1 = $s1
$t2 = $t0 + $t1
$t3 = $t1 + $t2
...
$t7 = $t5 + $t6
```

In other words, for each register from `$t2` to `$t7`, it stores the sum of the previous two `$t?` register values. The `$s0` and `$s1` registers contain the initial values.

Don't set the value of `$s0` and `$s1` in your code. Instead, learn how to set it manually with MARS. Save your code in a file called `MIPS_ex2.s`

Exercise 3: Debugging a MIPS program

Debug the loop in the program in `MIPS_ex3.s`. It is meant to copy integers from memory address `$a0` to memory address `$a1`, until it reads a zero value. The number of integers copied (up to, but NOT including the zero value), should be stored in `$v0`.

Describe in a file `MIPS_ex3.txt` the bug(s) of the code. Create a file `MIPS_ex3_ok.s` that contains a bug fixed version of `MIPS_ex3.s`

Exercise 4: Compiling from C to MIPS

The file `MIPS_ex4.c` is a C version of the program in the previous part of the lab. Compile this program into MIPS code using the command (if working in the UXB5 Linux server):

```
mips-linux-gnu-gcc -S -O2 -fno-delayed-branch MIPS_ex4.c -o MIPS_ex4.s
```

If working in Toolman use:

```
mips64-linux-gnu-gcc -S -O2 -fno-delayed-branch MIPS_ex4.c -o MIPS_ex4.s
```

The `-O2` option (letter "O" and 2) turns on a level of optimization. The `-S` option generates assembly code. The above command should generate assembly language output for the C code. Please note that you will not be able to run this code through MARS.

Find the assembly code for the loop that copies source values to destination values. Then, find where the source and dest pointers you see in `MIPS_ex4.c` are originally stored in the assembly file. Finally, explain how these pointers are manipulated through the loop.
Find the section of code in `MIPS_ex4.s` that corresponds to the copying loop and explain how each line is used in manipulating the pointer.

Questions:

1. Explain in a paragraph how the Spectre malware exploits speculative execution in the CPU.
2. Why does Spectre not require to exploit any software vulnerabilities?
3. Explain why vulnerable speculative execution implementations, found in most microprocessors, violate the security assumptions.
4. What is a side channel attack? Give one example.

SUBMISSION
Write your report including screenshots of your program running and explaining how it works. Include comments in your code too and submit your program source code files (.c and .asm). Submit your report as a DOCX or PDF document through Western Online. Include in the report the answers to the questions writing the corresponding numbers and questions (or at least the numbers) in **bold** and in the proper order before every answer. At the end include a conclusion (properly labeled as that) explaining the importance of all this.

Name your report using the following format:
`YourLastName_YourFirstName_CS590_Lab08.pdf`

----------///