COMP 4300 Project 2 FAQ

1. What is the exact functionality for the new functions on lab2. In addition, what will the syscall function do, and where will it take its implicit operands?

Answer: You need to simulate five separate functional units described in the Table on page 2.

Function Unit 1: Instruction Memory; **To support**: Instruction fetch
Function Unit 2: Register file; **To support**: register read
Function Unit 3: ALU; **To support**: computation
Function Unit 4: Data Memory; **To support**: data read/write
Function Unit 5: Register file; **To support**: register write

Each function unit aims to support one specific stage in a five-stage pipeline.

To find details on data/operands passed from one stage to the next stage, please refer to the datapath model of pipelining.

2. How is the palindrome string supposed to be written to memory? It is not apparent from the assembly code you gave us.

Answer: The input string can be loaded from a keyboard and stored in the data segment starting at "string_space:" (see the last line in the sample program below), where 1024 bytes are reserved for the string.

```
## read the string S:
la $a0, string_space
li $a1, 1024
li $v0, 8 # load "read_string" code into $v0.
syscall
```

The above code calls the "read_string" code, which is a system call that reads a string from a keyboard and saves the string in "string_space". If you are using the C programming language, you may use "`scanf()`" function to simulate the "read_string" system call.

3. How do you store strings into a data array? the array for the data is of mem_word type, which is an int32. do we have to convert the string to ints to get it into the array?

Answer: You may use a char array to represent the data segment.

4. Do we actually need to create some kind of pipeline in the simulator?

Answer: No, you do not need to implement a pipeline in your simulator. Please focus on how to simulate a multi-cycle machine with 32 general-purpose registers.

5. I just noticed that you haven't explicitly asked for our source code to be submitted; does this mean you don't want our source code, or should we submit it anyway?

Answer: <username>_gprSim should contain your source code.

6. Can we at least copy the lab2.s file into a txt file and remove the comments?

Answer: Yes, you can remove the comments from lab2.s.

7. What is palindrome.s?

Answer: palindrome.s to be submitted by you is a "partial assembly", where you relocate the code and data to reside at chosen locations in memory.

8. How are we supposed to put the string into a register that is of type int32?

Answer: A string cannot be directly placed into a register. You can only put each char in a register at a time.

9. In Project 2, should the GPR machine accept a hex encoded file as in Project 1, or should it accept ASCII?

Answer: Hex encoded file is easy to decode by your emulator. Nevertheless, you may choose to use ASCII as an input file.

10. Does the program counter increment over every line? Does it skip blank lines and Label only line?

Answer: When you load the program instructions into the text segment, please remove blank lines and label lines.
When you increase PC, it should point to the next instruction to be executed.

11. In the project description (page 5), in the list of deliverables it has:
"4) result.txt - It should contain the values you obtained for C, IC, and speed-up."
The exact values of C, IC, and speed-up are input dependent, so is this file just supposed to be a summary of average values and their significance?

Answer: C and IC depends on your input, whereas speedups are within a certain range. In your result file, you summarize C, IC and speedup values for your evaluated input strings. You may test at least two input strings (e.g., one

palindrome and one non-palindrome string) and report their corresponding C, IC, and speedups.

12. In the Project 2-group document, it says that palindrome.s must "lay out the code and data in memory and calculate the proper values fro the offset and label values." Does that mean we need to remove the loop labels, such as "length_loop:" and instead, just use the offset in the branch statement? Should it just be the number to add/subtract from the program counter or just number of lines to jump? For example, in the following block of code:

```
length_loop:
   lb $t3, ($t2)
   beqz $t3, end_length_loop

   addi $t2, $t2, 1
   b length_loop
```

Should "length_loop:" be removed and the branch statement look instead like this: `b -3`

Answer: You are right. All the labels should be replaced by specific offset values when you load the source file into your simulated machine. In your above example, the offset of the `beqz` instruction should be `-3`, meaning that PC will point to the `lb` instruction if `$t3`'s value equals to 0.

13. Should we ignore the case of an input string? So for example is "Anna" a palindrome?

Answer: The assembly code is what actually handles the character comparisons to determine if the string is a palindrome. Because 'A' and 'a' have different hex values, the program determines that "Anna" isn't a palindrome.