Comp4300 - Project 3
Pipelined Datapath with Interlocks and Forwarding
FAQs

1. Are we assuming all instructions take five clock cycles to complete?

   Answer:
   Yes, it takes 5 stages (5 cycles) to complete each instruction.

2. How should we run the Syscall operation with the five functional units of the simulator? Should we use the IF stage to fetch the values of the registers that it will use?

   Answer:
   The syscall operation is very similar to the branch instruction.
   You must fetch the instruction and decode it before handling the system call.

3. When implementing the pipelined simulator, at what stage should syscall's main operations be performed?

   Answer: A system call can be executed in the EXE stage.

4. **SYSCALL in Pipelined Simulation**
   System calls are proving a little confusing when trying to convert my groups simulator into 5 execution stages. Namely this is because certain system calls can do different things:
   - access registers (print integer)
   - access memory (print and read string)
   - terminate execution (exit).

   In MIPS they are handled by interrupts to the OS, which we aren't simulating.
   My question is how should we approach these system calls?
   Answer: A system call instruction (i.e., `SYSCALL`) should be executed in the EXE stage (i.e., the third stage of your pipeline). To simply our implementation, during the execution of the `SYSCALL` instruction, your simulator simply performs what the system call is supposed to do. For example, if the system call is to print an integer, then your simulator will display an integer in the EXE stage. In our implementation, let us ignore clock cycles spent in running the system call, which is handled by an operating system.

5. **Addi & Subi: Immediate or pointer values?**
   Just to make sure, `addi` and `subi` add and subtract immediate values from the register operands given.

Example:

```
li $2, 5
li $3, 8
addi $2, $2, 1 # $2 = $2 + 1
subi $3, $3, 5 # $3 = $3 - 5
```

In retrospect, my partner and I thought that the simulator would have to interpret this as a pointer to a value in memory. Is the pointer necessary or just a more complex way of interpreting the instruction?

Answer: This instruction adds the immediate value (i.e., 1) into the value held in $2. Please note that $2 isn't a pointer to a value in the memory; $2 stores the value to be increased by 1 in this case.

6. **Instruction Cycles**

Using the clock cycle/instruction table in Project 2's pdf, do we need to have each step take the number of cycles listed in the table? Or can we make each step take the same amount of cycles like we did in class (where we made each step take 40ms)?

Answer: Assuming there is no delays, each instruction takes five clock cycles (i.e., five stages, each stage per clock cycle) to execute regardless of its type. In your pipeline design, you have to ensure that each stage (a.k.a., step) in your pipeline is completed within one clock cycle regardless of the stage. For example, the MEM stage only takes one clock cycle, so does the ID stage and the other three stages.