**Assignment 5: MIPS Pipeline**

This assignment is based on MIPS pipeline simulation using WinMIPS64 simulator. The simulator and its documentation/tutorial is available at<http://indigo.ie/~mscott/>. Since the simulator is based on windows, you will have to use something like Wine ([https://www.winehq.org](https://www.winehq.org/)) to run it on ubuntu or macOS.

**Submission Instructions:** Make a directory named with your roll number. For each problem *i,* write your code (if required by the problem) in a file named *i.*s and include them in your submission directory. Also write answer to each question just below it and include this file in your submission. Finally submit a tar.gz i.e., [roll-no.].tar.gz (as follows) on moodle:

[roll-no.]

|----1.s

|----2.s

:

|----n.s

|----a5.docx (or any other format as you like)

1. Write a snippet of assembly code and use the simulator timing diagram to deduce whether each of the following is pipelined and, if so, the number of stages involved in each operation.

(i) Integer multiply- **Pipelined, number of stages = 7**

(ii) integer divide- **Not Pipelined, number of stages = 1**

(iii) FP add- **Pipelined, number of stages = 4**

(iv) FP multiply- **Pipelined, number of stages = 7**

(v) FP divide- **Not Pipelined, number of stages = 1**

2. Configure the pipeline to (a) suppress forwarding and (b) enable forwarding for different snippets of code.

**(a) Suppressed forwarding = 19 cycles, 5 instructions, 3800 CPI**

**(b) Enabled forwarding = 16 cycles, 5 instructions, 3200 CPI**

3. Write a 2-instruction sequence to maximize the time between the start of the first instruction and the end of the last. Repeat for a 3-instruction sequence.

**2 instruction sequence => 52 cycles, 2 insructions, 17.333 CPI (3-1.s)**

**3 instruction sequence => 76 cycles, 3 insructions, 19.000 CPI (3-2.s)**

4. Write a snippet of code with the potential to create a structural hazard in the multi-cycle execution case. Is this resolved in the pipeline and, if so, how?

**If we write add.d instruction after mul.d such that both add.d and mul.d will finish at the same time. Hence they will access the memory at the same time creating structural hazard if pipelining is not used. In case of pipelining, the second instruction will wait for some time hence preventing structural hazard (creating structural stalls)**

5. What is the time to execute a pair of multiply instructions with the second having a RAW dependency on the first, with and without forwarding?

**With forwarding => 18 cycles**

**Without forwarding => 20 cycles**

6. Write a snippet of code to determine the branch penalty in each case – (a) branch not taken and (b) branch taken? What are the penalties in each case and what do you conclude about the stage in which the branch condition is evaluated and the stage in which the target address is computed?

**Penalty when branch not taken = 0**

**Penalty when branch taken = 4**

**branch condition is evaluated and the target address is computed in the instruction decode stage of branch.**

7. Experiment by configuring the pipeline to support a branch delay slot. Show a snippet of code in which this feature is useful and one where this feature is not.

8. Write a snippet to illustrate a potential WAW hazard. Is this situation handled in the pipeline and, if so, how?

**This situation is handled in the pipeline using WAW stall.**

9. Can a WAR hazard exist in this pipeline? If so, how? If no, why not?

**Yes.**

**For example-**

**.text**

**add.d f7,f7,f3**

**add.d f7,f7,f4**

**mul.d f4,f5,f6 ; WAR on f4**

**If the mul.d is allowed to issue, it could "overtake" the second add.d and write to f4 first. Therefore in this case the mul.d must be stalled in ID.**

10. It is required to sum 12 FP numbers residing in registers f1 through f12. Write the snippet of code to perform the summation with the minimum number of clock cycles. What is the minimum time required? Assume that the final sum should be in f13 and that the contents of registers f1 through f12 should not be destroyed.

**23 cycles, 12 instructions, 1.917 CPI**