LSU EE 4720 **Homework 6** **Due: 23 April 2025**

Solution *Formatted 18:21, 30 April 2025*

Collaboration Rules

Each student is expected to complete his or her own assignment. It is okay to work with other students and to ask questions in order to get ideas on how to solve the problems or how to overcome some obstacle (be it a question of MIPS or assembler syntax, interpreting error messages, how a part of the problem might be solved, etc.) It is also acceptable to seek out assembly language resources for help on MIPS, etc. It is okay to make use of AI LLM tools such as ChatGPT and Copilot to generate sample code. (Do not assume LLM output is correct. Treat LLM output the same way one might treat legal advice given by a lawyer character in a movie: it may sound impressive, but it can range from sage advice to utter nonsense.)

After availing oneself to these resources **each student is expected to be able to complete the assignment alone**. Test questions will be based on homework questions and **the assumed time needed to complete the question will be for a student who had solved the homework assignment on which it was based**.

Student Expectations

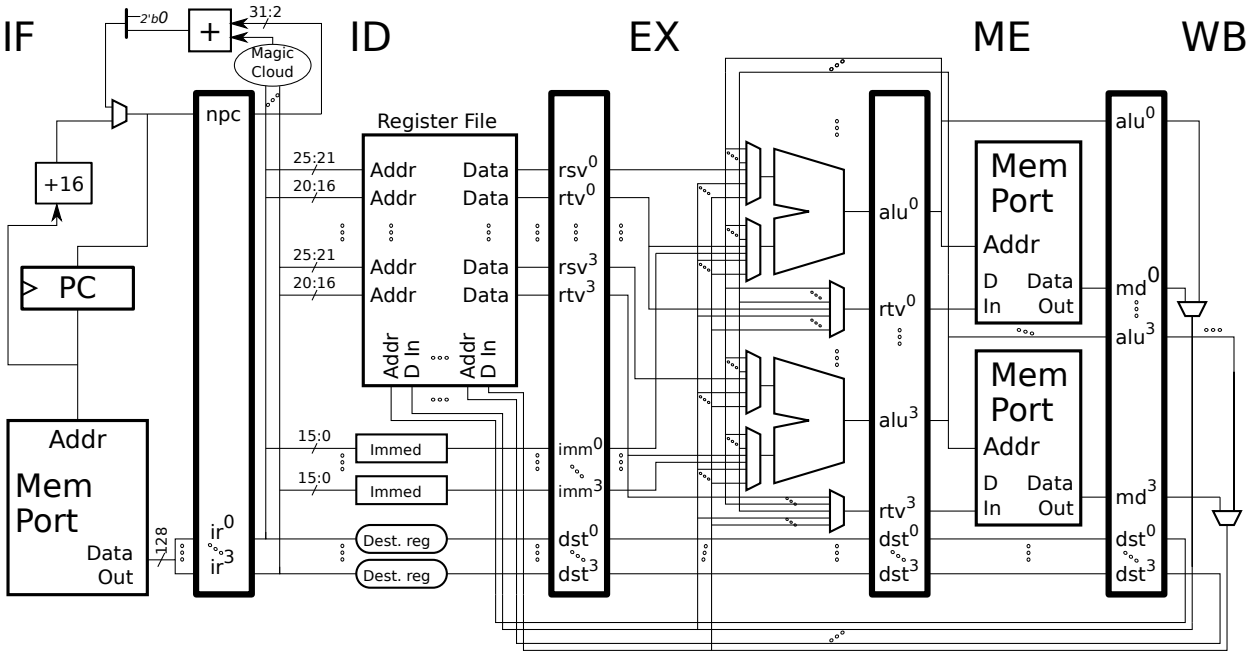
Some of the problems require thought, and students are expected to persevere until they find a solution. It is each student's duty to him or herself to resolve frustrations and roadblocks quickly, hopefully helped along by the satisfaction of making progress. There are plenty of old problems and solutions to look at. One way to resolve issues is to ask Dr. Koppelman or others for help.

Resources

Questions about superscalar MIPS implementations can be found in most final exams.

Problem 1: The following questions are based on 2021 Final Exam Problem 2(c), but **it is not identical**.

(*a*) Appearing below is a 4-way superscalar MIPS implementation which is **slightly different in an important way** from the one appearing in the 2021 Final Exam. In both this implementation and the one on the 2021 exam fetch is not aligned (which makes things easier). Also, there is no branch prediction, which is how we have been doing things in class.



☑ Show the execution of the code below for enough iterations to determine instruction throughout (IPC). (Note: There is no need to put slot numbers on the stage labels.) ☑ Don't forget that it is 4-way superscalar.

The solution is on the next page.

The difference between the implementation above and the one in the exam is that the one above has multiplexors on the path to the memory port D In connections, and so store instructions can bypass store values. This reduces the number of stalls suffered by the **sw** instruction.

Solution appears below. To keep instructions in order in **ID**, the stall of one instruction in **ID** stalls all instructions ahead in **ID**. So, in cycle 1 and 2 the **add** is stalling for the **lw** value. That forces the **sw** and **addi** to stall too, even though the **bne** and **addi** are not waiting for anything. The branch resolves in **ID** and so the target is not fetched until the branch is in **EX**, resulting in the squash of six instructions, the first two are shown, the next four are the instructions after the **xor** which are not shown.

The first iteration starts in cycle 0, the second starts in cycle 6, and so the instruction throughput is $\frac{6}{6-0} = 1\,\text{insn/cycle}$.

```
LOOP: # Cycle      0  1  2  3  4  5  6  7  8  9  10 11 12   First Iteration
 lw R10, 0(r1)     IF ID EX ME WB
 add R3, R10, r3   IF ID ----> EX ME WB
 sw R3, 0(r5)      IF ID -------> EX ME WB
 addi r5, r5, 4    IF ID -------> EX ME WB
 bne r1, r9, LOOP     IF -------> ID EX ME WB
 addi r1, r1, 4       IF -------> ID EX ME WB
 lb r8, 0(r9)         IF -------> IDx
 xor r11, r8, r10     IF -------> IDx
LOOP: # Cycle      0  1  2  3  4  5  6  7  8  9  10 11 12 13 14   Second Iter
 lw R10, 0(r1)                     IF ID EX ME WB
 add R3, R10, r3                   IF ID ----> EX ME WB
 sw R3, 0(r5)                      IF ID -------> EX ME WB
 addi r5, r5, 4                    IF ID -------> EX ME WB
 bne r1, r9, LOOP                     IF -------> ID EX ME WB
 addi r1, r1, 4                       IF -------> ID EX ME WB
 lb r8, 0(r9)                         IF -------> IDx
 xor r11, r8, r10                     IF -------> IDx
```

(*b*) The code from the solution to Final Exam 2021 2(c) has an instruction throughput of $\Theta_c = 0.75\,\text{insn/cycle}$. The solution to part 2(d) did not give the instruction throughput of the solution but did explain that the unrolled code is four times faster.

☑ What is the instruction throughput (IPC) of the 2(d) solution? (The pipeline execution diagram is in the solution, use that!)

The instruction throughput is $\frac{10\,\text{insn}}{(4-0)\,\text{cyc}} = 2.5\,\text{insn/cycle}$

.

☑ Why can't we use the instruction throughput of parts (c) and (d) to show how much faster part (d) is?

In general, one can't compare the execution time of two different code fragments by comparing their instruction throughput (IPC) because one also need to know how many instructions each code fragment executes. It *does* make sense to use IPC to compare the execution time of the same code fragment on two different implementations, which is something that is frequently done in this class.

Also note that the code for part (d) is unrolled degree 2, and so it operates on two elements per iteration, while the original code in (c) just operates on one element per iteration. Just comparing IPC we would conclude that the part-(d) code is $\frac{2.5}{0.75} = 3.33$ times faster, which isn't bad. But to compare the two we should look at how much work is done per cycle. For the part (c) code that would be $\frac{1}{8-0} = 0.125$ work items per cycle. For part (d) it is $\frac{2}{4-0} = 0.5$ work items per cycle, and so the part (d) code is $\frac{0.5}{0.125} = 4$ times faster, even better than 3.33.

(*c*) In part (d) the loop was to be unrolled degree 2. Here, unroll the loop degree 3 (start with three copies of the loop body) but for the implementation shown here (not from the exam). A correct solution should execute without stalls, but instructions will be squashed due to the branch (which can't be avoided in a 4-way superscalar without branch prediction).

☑ Unroll degree 3 and optimize so there are no stalls.

The solution appears below. The prologue and epilogue are omitted. Notice that the `add` instructions are positions so that they will be in different fetch groups.

```
 # SOLUTION
LOOP: # Cycle      0  1  2  3  4  5  6  7  8  9  10
 sw r13, 0(r5)     IF ID EX ME WB
 sw r3, 4(r5)      IF ID EX ME WB
 sw r23, 8(r5)     IF ID EX ME WB
 add r13, r10, r3  IF ID EX ME WB
 addi r5, r5, 12      IF ID EX ME WB
 add r14, r12, r13    IF ID EX ME WB
 lw r10, 0(r1)        IF ID EX ME WB
 lw r12, 4(r1)        IF ID EX ME WB
 add r3, r22, r14        IF ID EX ME WB
 lw r22, 8(r1)           IF ID EX ME WB
 bne r1, r9, LOOP        IF ID EX ME WB
 addi r1, r1, 12         IF ID EX ME WB
LOOP: # Cycle      0  1  2  3  4  5  6  7  8  9  10
 sw r13, 0(r5)                 IF ID EX ME WB
 sw r3, 4(r5)                  IF ID EX ME WB
 sw r23, 8(r5)                 IF ID EX ME WB
 add r13, r10, r3              IF ID EX ME WB
 addi r5, r5, 12                  IF ID EX ME WB
 add r14, r12, r13                IF ID EX ME WB
```

```
 lw r10, 0(r1)                    IF ID EX ME WB
 lw r12, 4(r1)                    IF ID EX ME WB
 add r3, r22, r14                    IF ID EX ME WB
 lw r22, 8(r1)                       IF ID EX ME WB
 bne r1, r9, LOOP                    IF ID EX ME WB
 addi r1, r1, 12                     IF ID EX ME WB
LOOP: # Cycle      0  1  2  3  4  5  6  7  8  9  10
```

Problem 2: Solve 2024 Final Exam Problem 2 (all parts), in which code for a 2-way superscalar MIPS implementation is to be completed (a) and the execution of code on a 4-way superscalar MIPS implementation is to be found.

See the final exam solution at `https://www.ece.lsu.edu/ee4720/2024/fe_sol.pdf`.