LSU EE 4720 **Homework 5** **Due: 8 April 2026**

Solution *Formatted 13:29, 17 April 2026*

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, RISC-V, or assembler syntax, interpreting error messages, how a part of the problem might be solved, etc.) It is also acceptable to seek out resources for help on MIPS, RISC-V, etc. It is okay to make use of AI LLM tools such as ChatGPT, Claude, 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

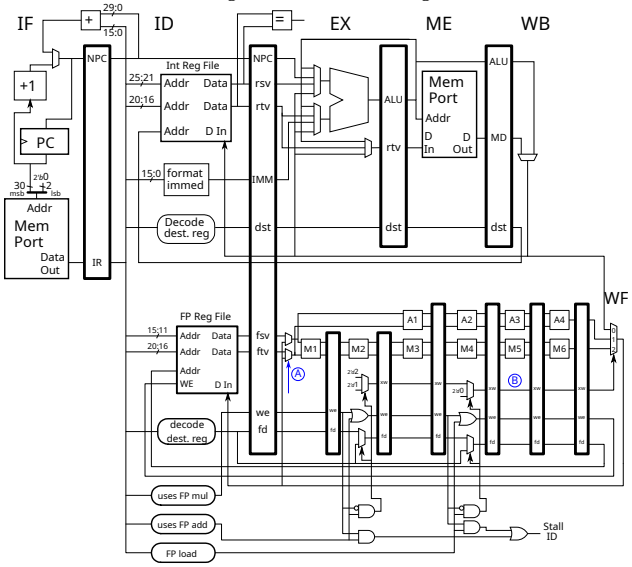
To solve this assignment you are expected to avail yourself of references provided in class and on the Web site, and to learn how to handle references that are at first hard to understand, and to keep looking (and asking) when the answer isn't in the first place you look. Some of the problems require thought, and you are expected to persevere until you find a solution. It is each student's duty to him or herself to resolve frustrations and roadblocks, 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

Many past final exams and homework assignments have problems on FP pipeline variations and control logic.

Problems start on the next page.

Larger on Next Page



Larger on Next Page

Problem 1: *This problem is based on one that appeared on a recent exam, with some enhancements.* Appearing to the right and much larger on the next page is our pipelined MIPS implementation with floating point hardware. This hardware includes multiplexors to bypass values to the FP multiply and addition units. Two wires in the diagram are labeled (in blue) A and B.

☑ Show the execution of the code below on the implementation to the right. ☑ Check for dependencies. ☑ Don't forget to avoid `WF` structural hazards.

☑ Show the value on the wires labeled A and B **only** for the cycles in which the respective values are used. ☑ Leave a cycle blank when it doesn't matter what the value is (execution would be correct for any value).

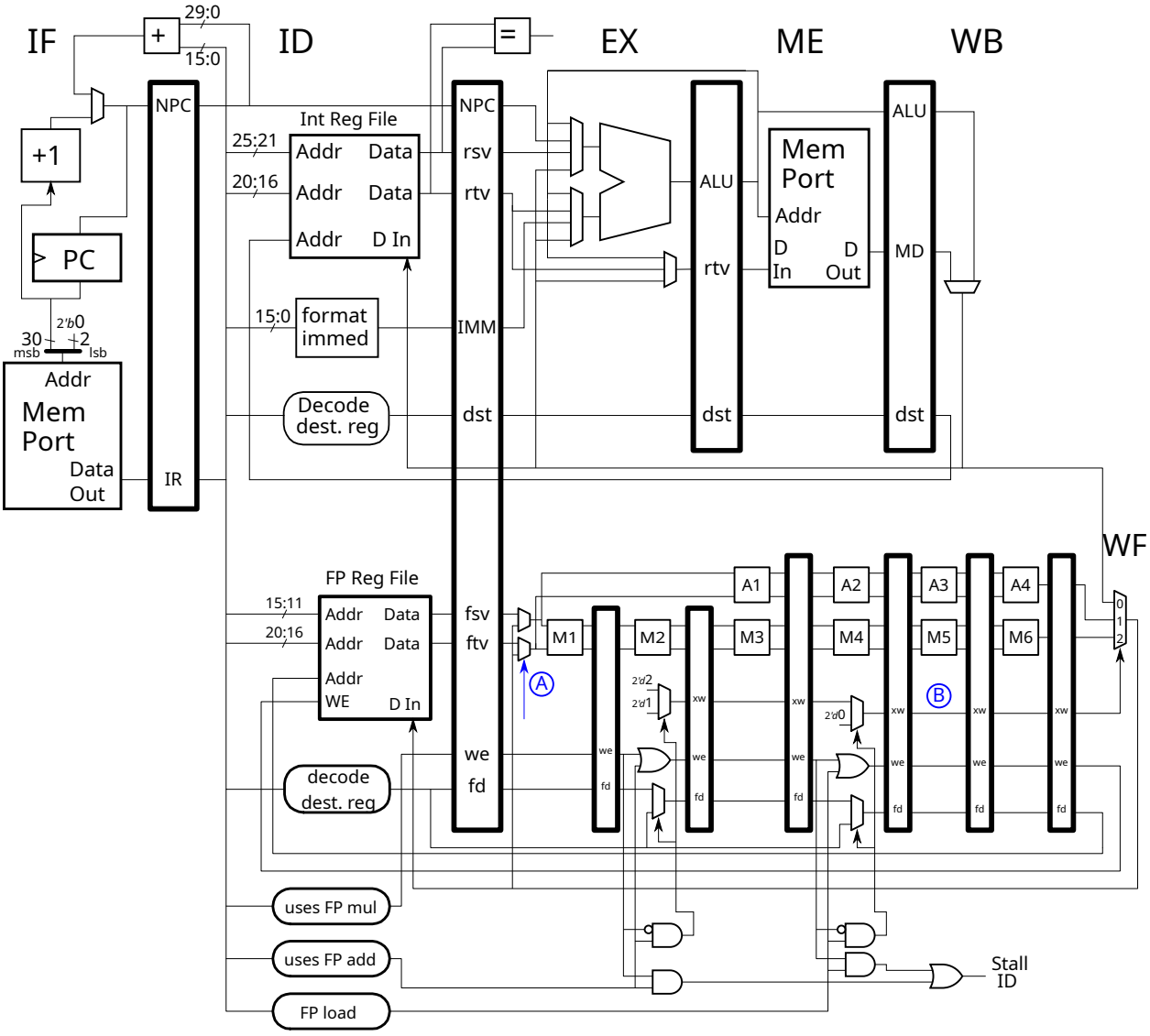
The solution appears below. The `add.s f8` has to stall two cycles to avoid the structural hazard at `WF`. Note that the stall occurs in `ID`, not in `A4`. The `add.s f7` stalls because of the dependence carried by `f4`, and the `add.s f10` stalls due to the dependence carried by `F11`.

The A signal is used for instructions in `A1` and `M1` (which are two names for the same stage). If no instruction is there, as in cycles 0, 1, 4, etc., then the value should be left blank. When an instruction is in `A1` or `M1` then the value is `1` if a bypass to the lower input is needed, otherwise it's `0`. The `add.s f7` instruction receives a bypassed value (`f4`) in cycle 1, and the `add.s f10` in cycle 12. Otherwise there are no bypasses.

The B signal, which is in the `A3` `M5` stage, carries the select signal for the `WF` mux. It's value is used if there is an `add.s` or `mul.s` instruction in that stage *or* if there is an `lwc1` in `EX`. For the cycles in which `M5` is occupied B should be `2` since it uses input 2 to the `WF` mux. When `A3` is occupied B should be `1`, and when there is an `lwc1` in `EX` B should be `0`.

```
# SOLUTION
# Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12
mul.s f1, f2, f3   IF ID M1 M2 M3 M4 M5 M6 WF
mul.s F4, f5, f6      IF ID M1 M2 M3 M4 M5 M6 WF
add.s f8, f15, f16       IF ID ----> A1 A2 A3 A4 WF
add.s f7, f9, F4            IF ----> ID ----> A1 A2 A3 A4 WF
lwc1 F11, 0(r1)                      IF ----> ID EX ME WF
add.s f10, f12, F11                          IF ID -> A1 A2 A3 A4 WF
# Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
A                        0  0        0        1        1
B                                    2  2  1     0  1        1
# Cycle            0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
```

Use the diagram below for Problem 1:



There's another problem on the next page.

Problem 2: Problem 2 from the 2025 Final Exam asks for completing the design of a MIPS FP pipeline in which FP add instructions pass though the same six stages as the FP multiply, but only perform computation in the first four.

SVG versions of the diagrams from the exam problem can be found at `https://www.ece.lsu.edu/ee4720/2025/fe-h-part-a.svg` and `https://www.ece.lsu.edu/ee4720/2025/fe-h-part-b.svg`, a collection of logic gates is at `https://www.ece.lsu.edu/ee4720/2026/c.svg`. All of these can be edited with your favorite SVG editor, mine is Inkscape.

(*a*) Solve 2025 Final Exam Problem 2 (both parts). Though this problem was based on a homework assignment and past exam questions, it can be solved without referring to them. That said, there's nothing wrong with looking at those problems and their solutions.

See the posted final exam solution for a detailed explanation, or look at the solution to part b of this problem which also shows the final exam solution but only discusses part b of this problem.

(*b*) Part b of the exam problem mentions a possible 2026 homework problem in which stall logic is to be designed. This is **not** that problem. Notice that in the illustrated hardware for the exam problem there is no path from the data memory port (in `ME`) to the FP register file, which would be needed for `lwc1`. (Such a path does exist in the FP hardware for Problem 1 of this homework assignment, but that path won't work for this problem.) Add such hardware for FP load instructions, and do it in a way that avoids structural hazard stalls in the same way that the hardware already avoids structural hazards with `add.s` and `mul.s`. That is, a `lwc1` instruction should go through the same number of stages as a FP add and multiply.

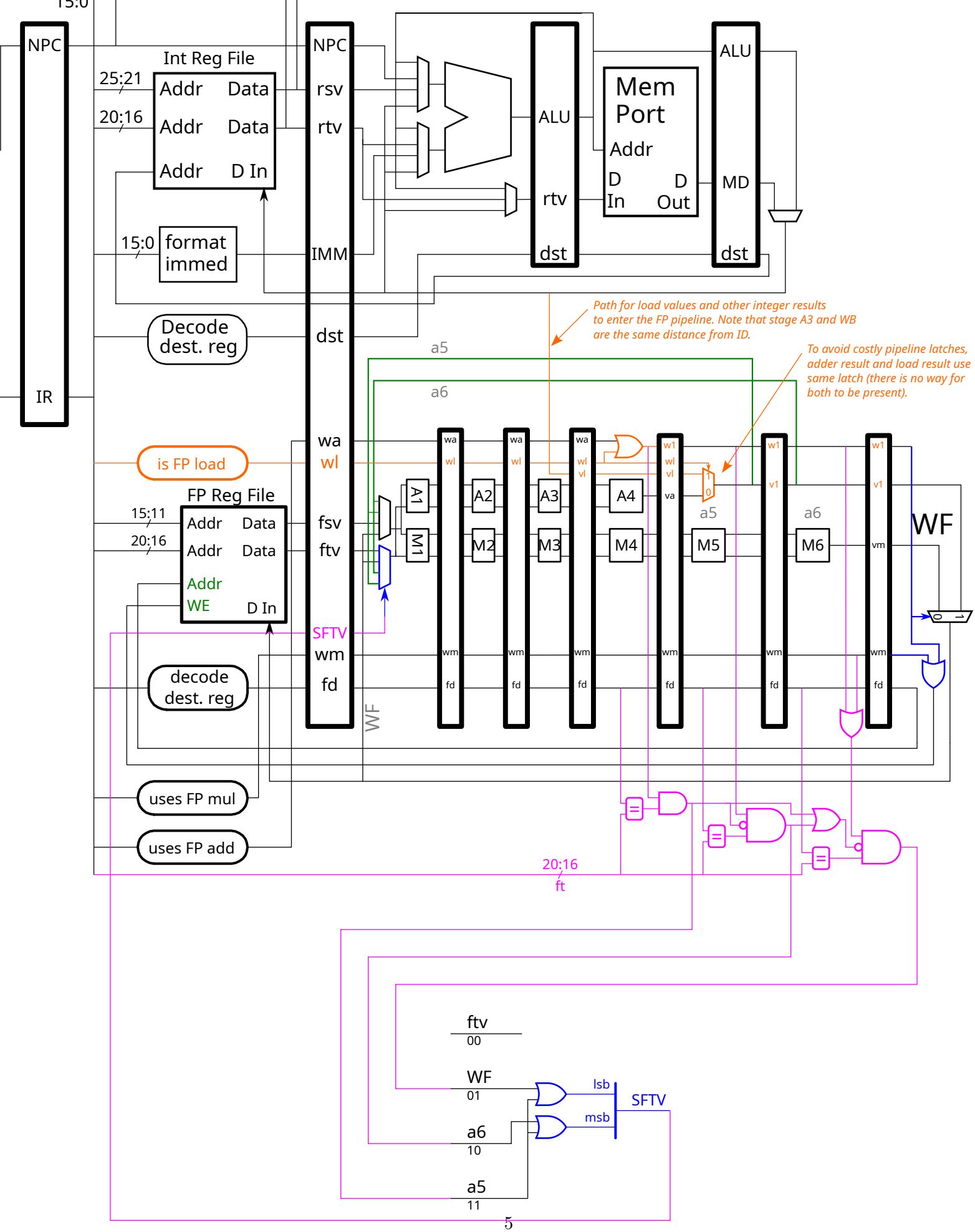
☑ Add connections so that `lwc1` can execute without risk of a structural hazard at `WF` ☑ by going through the same number of stages as `add.s` and `mul.s`.

☑ Connections should include the value to write back, control signals, and the register number.

☑ Pay attention to cost and critical path. ▷ Assume that per-bit, pipeline latches cost twice as much a 2-input multiplexor.

The solution appears on the next page. A path for load values and other integer results has been added from the `WB` stage to `A3`. Note that both `WB` and `A3` are the fifth stage (where `IF` is the first stage). The load value is put in a new pipeline latch, labeled `vl` (value load, though another good label would be `vi` since the value is from the integer pipeline but if I did that I'd need to emphasize that the `i` means the value is from the integer pipeline, not that it's in an integer format nor that it is destined for an integer register). In `a5` a multiplexor routes either the adder or integer pipeline value (`vl`) into the pipeline latch to reduce the number of pipeline latch bits needed starting at this point. The reason that a mux was not used in stages `A3` and `A4` was because the problem stated these units were on the critical path and so any additional delay would slow down the system. Because the two values start sharing a wire at `a5` a single write-back bit, `w1`, is used. (The `1` indicates the `WF`-stage mux input that it will use.) Note that the changes were done is such a way that our bypass hardware can now bypass a `lwc1` from stages `a5`, `a6`, and `WF`.

Grading Note: Some student solutions used values from `WB.MD` rather than the mux output. Though this would work for `lwc1`, it would not for `mtc1`. No points were taken off because `mtc1` was not part of the problem.


NPC
IR
Int Reg File
25:21
20:16
Addr
Data
Addr
Data
Addr
D In
15:0
format immed
NPC
rsv
rtv
IMM
dst
ALU
rtv
dst
Mem Port
Addr
D In
D Out
ALU
MD
dst
Path for load values and other integer results to enter the FP pipeline. Note that stage A3 and WB are the same distance from ID.
To avoid costly pipeline latches, adder result and load result use same latch (there is no way for both to be present).
Decode dest. reg
dst
a5
a6
wa
wl
is FP load
FP Reg File
15:11
20:16
Addr
Data
Addr
Data
Addr
WE
D In
fsv
ftv
SFTV
wm
fd
decode dest. reg
WF
A1
M1
A2
M2
A3
M3
A4
M4
M5
M6
WF
uses FP mul
uses FP add
20:16
ft
ftv
00
WF
01
a6
10
a5
11
lsb
msb
SFTV