LSU EE 4720 **Homework 5** **Due: 7 April 2025**

Solution *Formatted 15:45, 29 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 MIPS floating-point implementation can be found in most final exams.

Problem 1: Solve the last part of 2024 Final Exam Problem 1, in which the execution of FP code on an ordinary MIPS FP implementation is to be shown.

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

Next problem on next page.

Problem 2: Consider 2024 Final Exam Problem 3, in which some control logic is to be designed for a FP implementation in which an `add.s` instruction normally goes through the same number of stages as a `mul.s`, though only using four of those for computation, but in which an `add.s` instruction can also hop ahead if `WF` is available in an earlier cycle, thus reaching `WF` one or two cycles earlier.

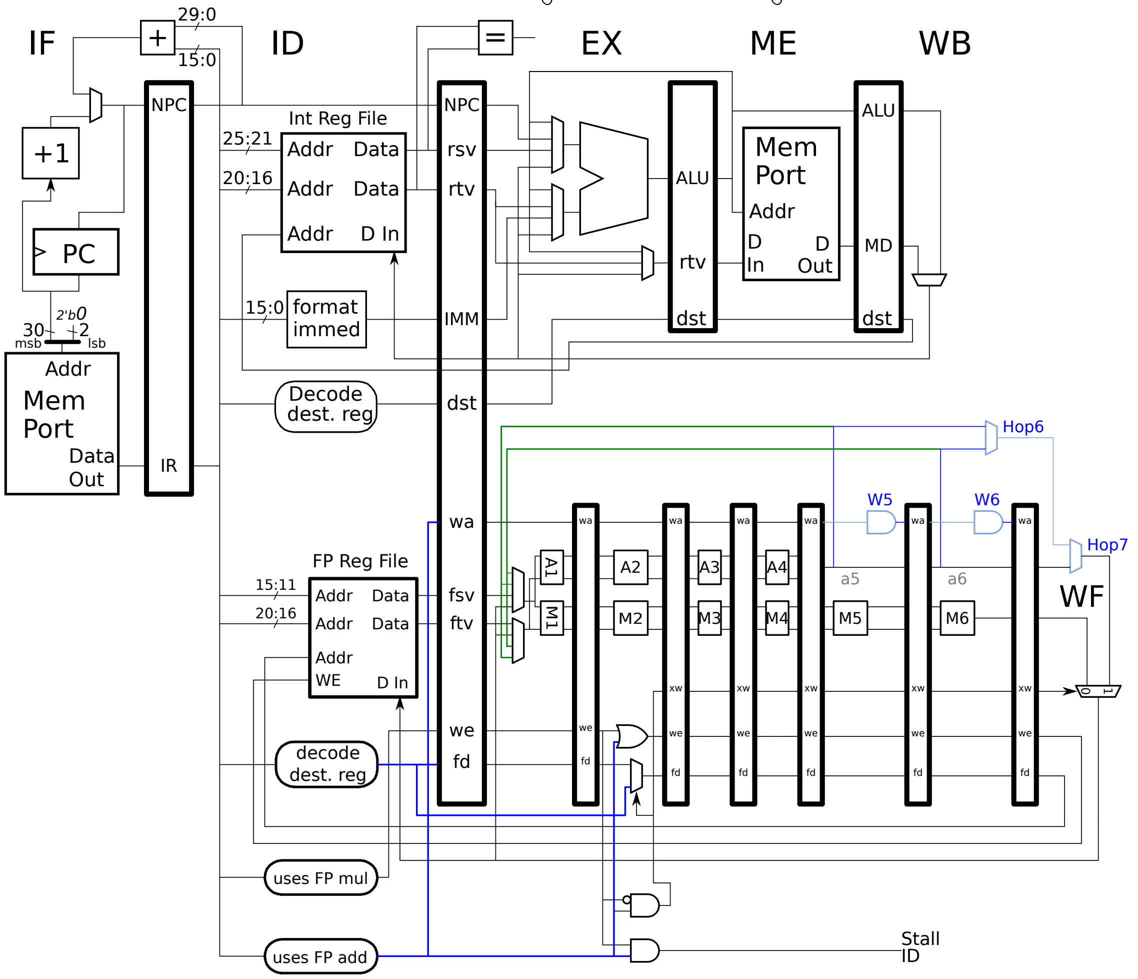
A copy of the illustration used in the exam, in SVG format, can be found at `https://www.ece.lsu.edu/ee4720/2024/fe-fp-hop.svg`. It can be edited using Inkscape or any other SVG editor.

(*a*) Solve 2024 Final Exam Problem 3.

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

(*b*) The italicized text in Problem 3 mulls that maybe hopping isn't such a good idea, that it might be less costly to bypass than hop. Modify the implementation so that rather than hopping bypass paths are provided for instructions in `a5` and `a6`. This is actually an easy problem since there is no need to show control logic for this part. *Maybe for the final exam.*

Solution appears below in green. All one needed to do is connect the pipeline latches carrying the data to the `A1`-stage multiplexors. The connections shown allow maximum flexibility, for example, bypassing the value written by the instruction in `a5` to the `fs` operand and the value written by the instruction in `a6` to the `ft` operand. Using the output of the `Hop6` multiplexor would have lowered costs but would have limited bypass to only one value. Note that the `WF`-stage mux could not be used since it would be needed for writing a value to the `FP` register file.


IF
ID
EX
ME
WB
29:0
15:0
+
=
+1
PC
NPC
IR
30
2'b0
+2
msb
lsb
Addr
Mem Port
Data Out
Int Reg File
25:21
20:16
Addr Data
Addr Data
Addr D In
15:0
format immed
Decode dest. reg
NPC
rsv
rtv
IMM
dst
ALU
rtv
dst
Mem Port
Addr
D In
D Out
ALU
MD
dst
Hop6
Hop7
W5
W6
a5
a6
WF
wa
fsv
ftv
we
fd
FP Reg File
15:11
20:16
Addr Data
Addr Data
Addr
WE
D In
decode dest. reg
uses FP mul
uses FP add
A1
A2
A3
A4
M1
M2
M3
M4
M5
M6
xw
Stall ID