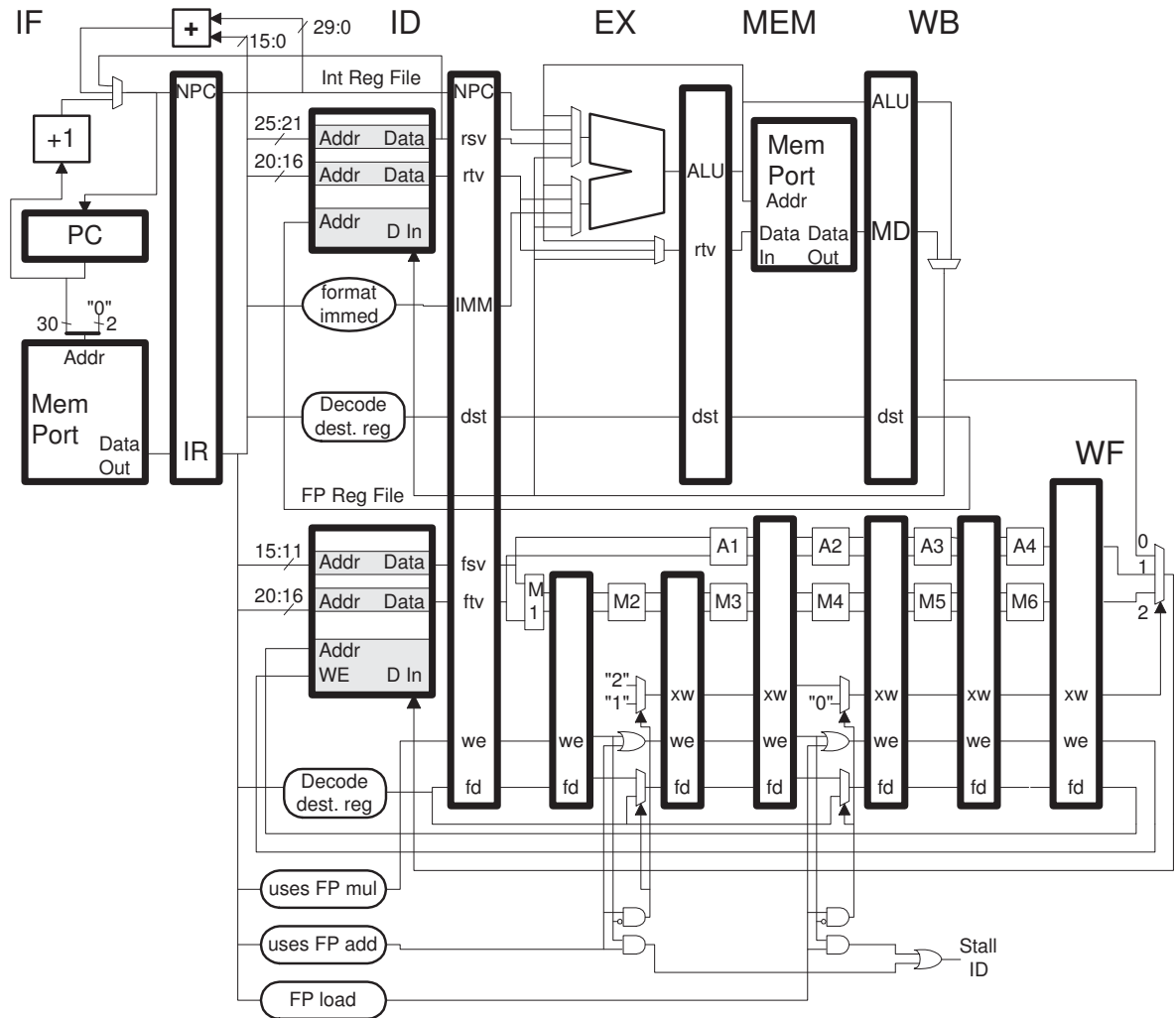**Problem 1:** The code below executes on the illustrated MIPS implementation. Assume that any reasonable bypasses needed for the FP operands are available, even though they are not shown in the illustration. A bypass is reasonable if it does not have a significant impact on clock frequency and if it does not use circuitry that can predict the future.


```
LOOP:
 ldc1 f0, 0(r1)
 mul.d f2, f0, f4
 add.d f6, f6, f2
 bne r1, r2, LOOP
 addi r1, r1, 8
```

(*a*) Show a pipeline execution diagram covering enough iterations to compute the CPI. Don't forget to check code for dependencies.

(*b*) Compute the CPI.

(*c*) Remember, that some bypass paths are assumed present though not illustrated. Add the needed paths to the implementation and show when they are used.

Problem 2: Precise exceptions are necessary for integer instructions, but only Nice To Have for floating-point instructions. Suppose exception conditions, such as overflow, were detected in `A4` and `M6` in the pipeline from the previous problem.

```
mul.d f2, f0, f4
add.d f6, f6, f2
and r3, r3, r5
addi r1, r1, 8
```

(*a*) For the code fragment above, would a `mul.d` exception detected in `M6` be precise? Explain in terms of architecturally visible storage (register and memory values) when the handler starts. (Note that in general exceptions detected in `M6` would not be precise, but the question is only asking about the fragment above.)

(*b*) For the code fragment above, would a `add.d` exception detected in `A4` be precise? Explain in terms of architecturally visible storage when the handler starts.

Problem 3: The MIPS implementation below has a fully pipelined FP add unit. Replace the FP add unit with one that has an initiation interval of 2 and a total computation time of 4 cycles. Note that the time to compute a floating point sum is the same on the original and replacement adder.
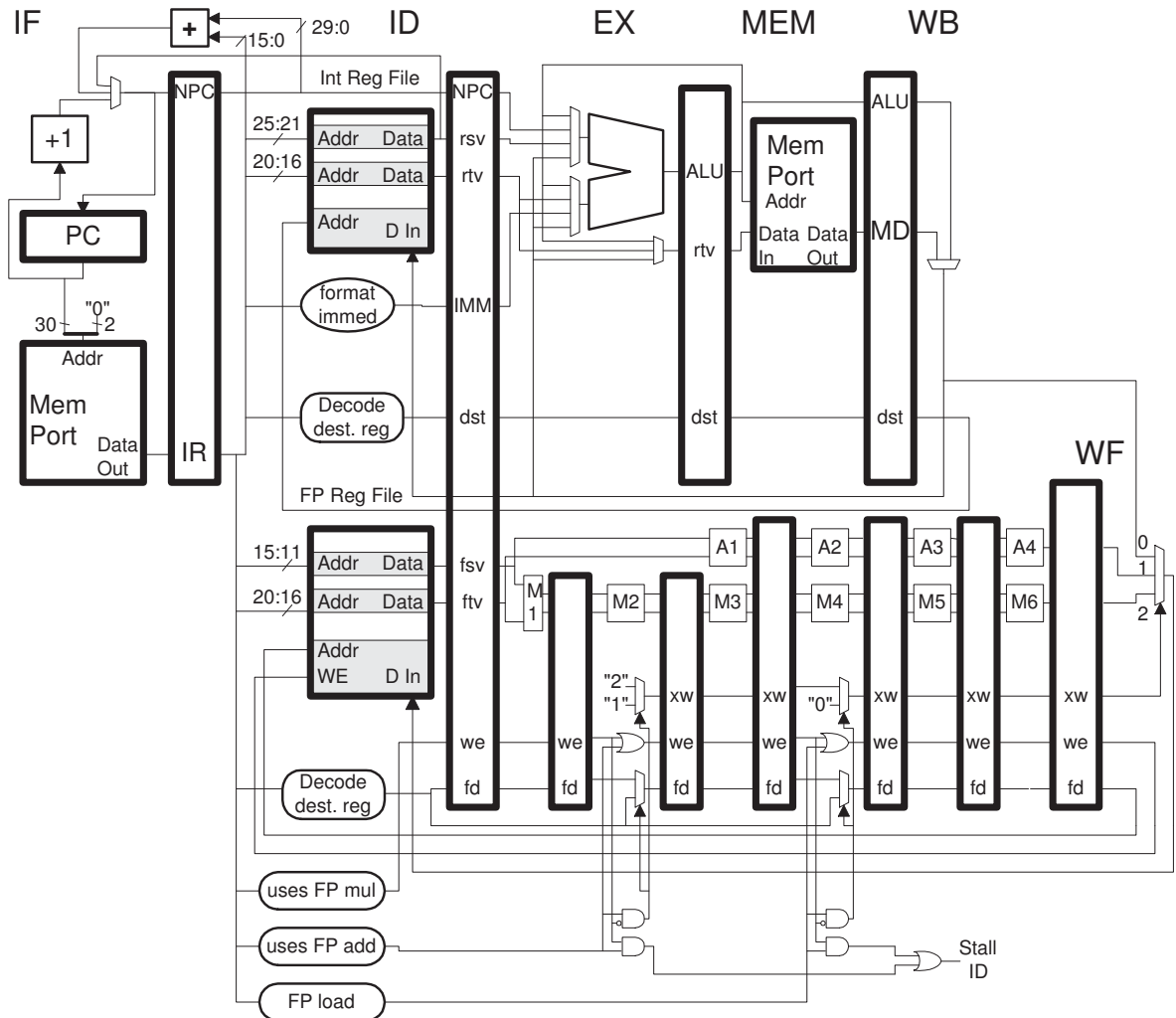
The new adder has two stages, `A1` and `A2`, each has two inputs (like their fully pipelined counterparts), and each has two outputs. In the first cycle of computation the source operands are placed at the inputs to `A1`, in the second cycle of computation the values at the outputs of `A1` at the end of the first cycle are placed at the inputs to `A1`. In the third cycle the values at the outputs of `A1` at the end of the second cycle are placed at the inputs `A2`, and in the fourth cycle the inputs to `A2` are the values at the outputs of `A2` at the end of the third cycle. The sum is available from the upper output of `A2` at the end of the fourth cycle.

(*a*) Replace the FP adder datapath with the one described above.

(*b*) Modify the control logic for the new adder. Be sure to account for the structural hazard when there are two consecutive FP add instructions.