**Lab 3 Milestone**

Q1:

Perform the following tasks for your non-pipelined processor and include them in your written report:

* List the reported Fmax of your design before pipelining.

**Fmax = 53.7 MHz**

* List the cycle time of your design before pipelining. Show your work.

**Cycle Time = 1/Fmax = 1/53.7 = 0.01862 μs**

* List the instruction count for miner\_tb from the ModelSim simulation before pipelining.

**Instruction Count = 350,961**

* List the cycle count for miner\_tb from the ModelSim simulation before pipelining.

**Cycle Count = 361,890**

* Compute the CPI and execution time (# of cycles x cycle time) before pipelining.

**CPI = Cycles / Instructions = 361,890 / 350,961 = 1.03114**

**Execution Time = 361,890 \* (1/53.7) = 6738.106 μs**

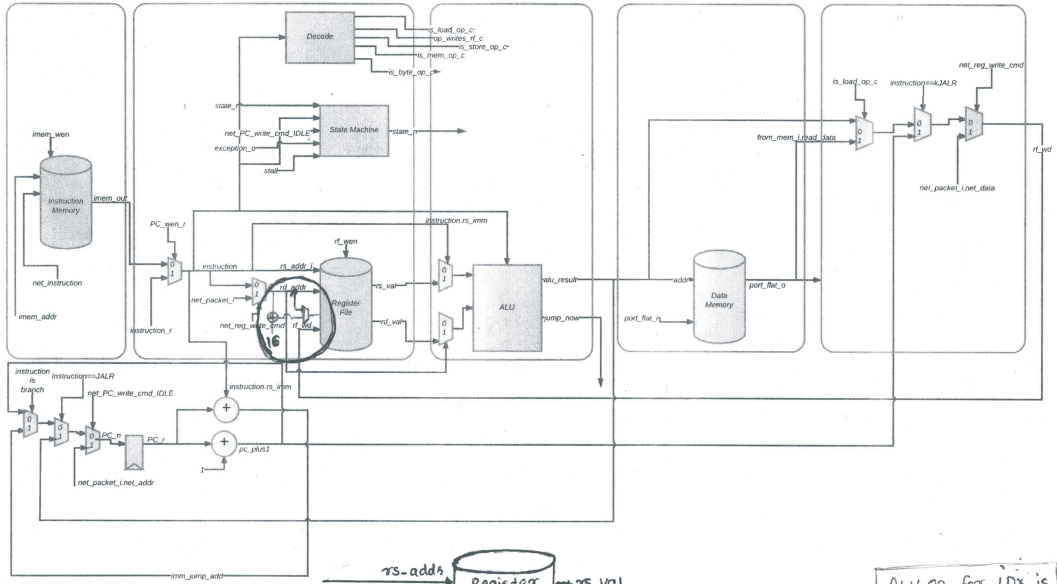
* List the number of registers used, the number of combinational functions used, and the number of memory bits used before pipelining.

**Registers = 2,069**

**Combinational Functions = 3,913**

**Memory Bits = 16,384**

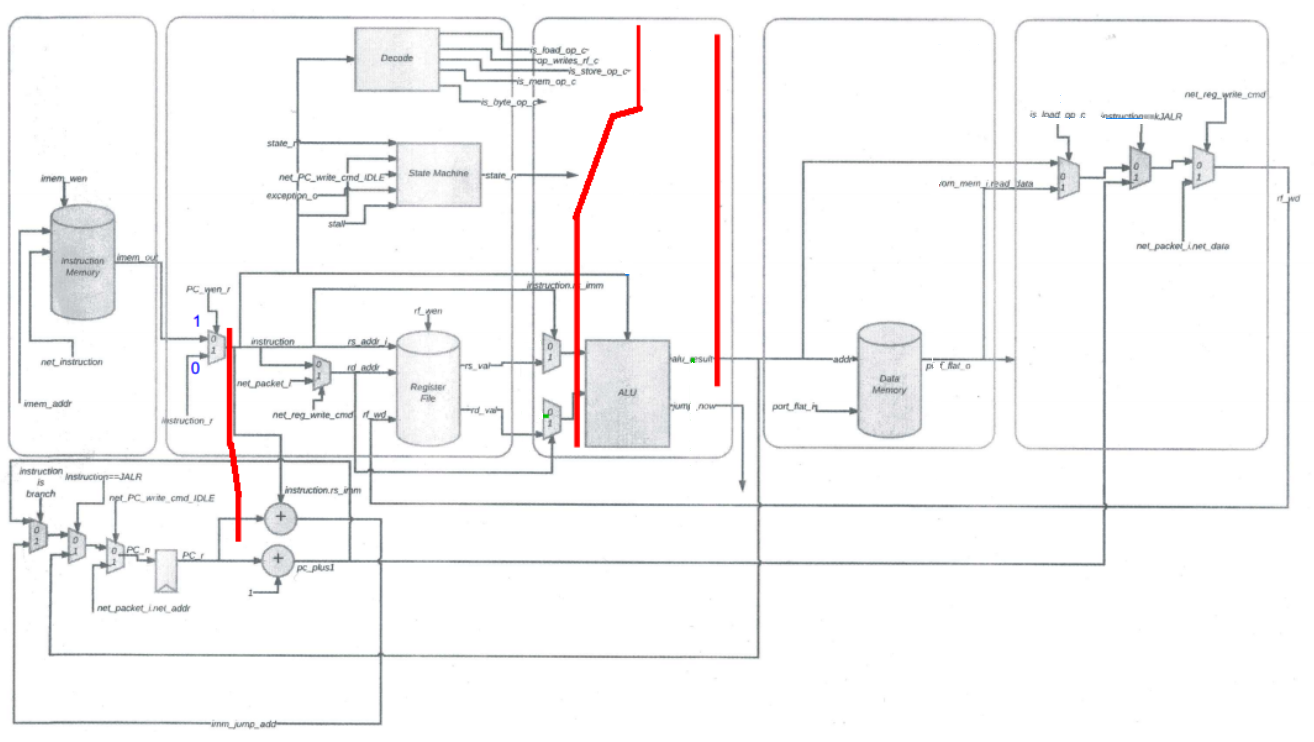
* Include a datapath diagram of your core. We strongly encourage you to draw this by hand. It will become a very useful tool in debugging your core.



Q2:

Include answers to the following questions in your written report:

* On your datapath diagram, define the boundaries of stages in your pipeline by drawing in the required registers. Spend some time doing this as it will be a useful tool in debugging the core.



* Discuss the tradeoffs of having a pipeline with more vs. less stages.

**A pipeline with more stages will usually have a lower cycle time and higher latency than a pipeline with less stages.**

Q3:

Include answers to the following questions in your written report:

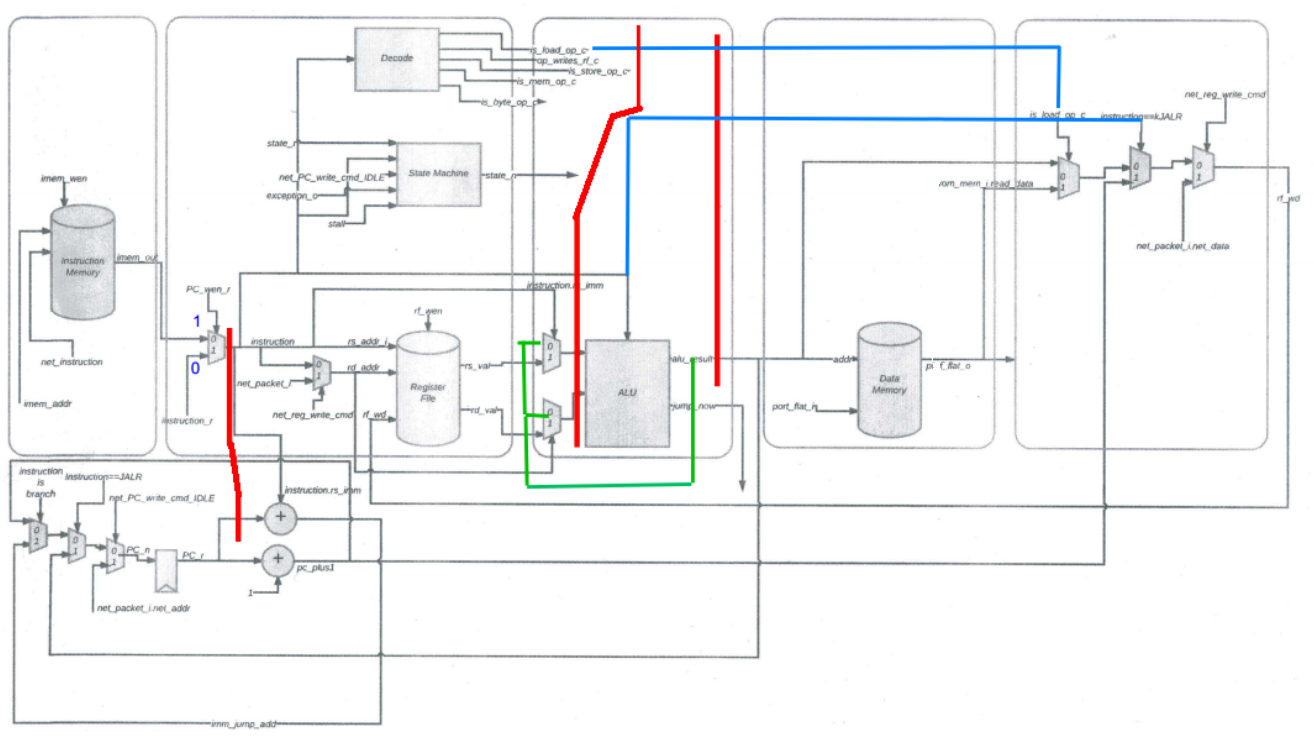
* You will need to pass the proper control signals to each stage of your pipeline. List which control signals are necessary for each stage of your pipeline.

**IF/ID Stage: 32 bit instruction, PC register**

**ID/EXE Stage: instruction, rd register, rs register**

**EXE/MEM Stage: alu result, jump flag**

* Update your datapath diagram to show the additional datapaths, registers, and muxes needed to implement pipeline forwarding.



Q4:

Include answers to the following questions in your written report:

* List all data hazards in your pipeline, and how they should be resolved (which stage data should be forwarded from, etc).

**Input of an instruction relies on the output of a previous instruction.**

**We forward from the EXE stage when an ALU instruction is in progress (and we forward from the MEM stage when a memory instruction is in progress).**

* List all control hazards in your pipeline, and how they should be resolved.

**Not knowing if a branch instruction will be taken or not taken, and executing the following incorrect instructions.**

**We assume branches will always be not taken, and flush instructions if there is a branch mis-predict.**

* List all structural hazards in your pipeline, and how they should be resolved.

**When a register is being read from and written to in the same cycle.**

**We would need to stall if a structural hazard is detected. (If possible, we would write at the start of a clock edge, and read in the later part of a cycle.)**

Q5:

Why would you want to have an independent write address port on the register file for a pipelined processor (i.e. why would you need to write to an address other than the rd address of the instruction in the decode stage)? Hint: think about what instructions may be in the write back stage (the stage which writes to the register file) and the the decode stage (the stage that reads from the register file).

**We would want an independent write address port for the register file for the case where the same register address is being written to and read from in the same cycle, so that the rd address from a previous instruction does not get overwritten.**