report of the pipelining manipulation on a QuAC ISA CPU

Pipeline stages

The experimenter identified five stages in this CPU. In stage one, CPU fetches instruction from RAM. In stage two, CPU will decode that instruction into control signals, so it can know which two registers it needs to read, which executions' result it need to writeback later, and how will those execution operate. After requiring above information, it enters stage three and four, that the third do arithmatic operations, and the fourth handle memory store or load. Totally, these two stages will produce four different results, one of it also contains data that updates flag register.

The pipelining manipulation experimenter asserted on the single cycle CPU only fragments the CPU into four stages, that he combined above discussed memory stage and ALU stage together. His reason on this decision is that, these results are gonna be seleted by one control signal DMUX, and be written back to register at once, treating them as one unit reduces design complexity. He rename this combined stage as Operations.

Three pipeline registers are inserted in the CPU, one between instruction fetch and decode, register read named with IF/ID. One between Decode, Register Read and Operations, named ID/OP. The last one is located between operations and Writeback, named OP/WB.

For now, if user want to change r1 to 9, change r2 to 1, and increase r1 with r2, his compiler can place two movl next to each other, but has to insert three nops between movl r2 and add r1.

This deign doesn't have hardware nop. All nops are inserted by compiler.

Handle Data Hazards

If the pipeline design stops at above, then, for every dependency, the compiler has to insert three nops for data write backs, the optimisation designer implemented is a forwarding feature.

The feature exhibits at the end of operation stage, connects backwards to ID/OP register. Every time the operation produces one result, it will be refered into ID/OP register, with the information of which register is meant to be written by above result. ID/OP register will check the whether the register read select is equal to that refered write select, if so, then the register data will be changed with refered data.

Doing so will allow each instruction wait one clk less for dependent data's write back.

Under the effect of above stage fragmentation, the forwarding unit doesn't need to check weather it is "forward back in time", both MEM and ALU outputs information in the same time.

The data forwarded not only includes results, but also include alu flags.

Handle Control Hazards

This design handles control hazards by simply flushing.

Starts at ID/OP and OP/ID register, everytime it receives an instruction for manipulating PC register, it will sends a signal to IF/ID to clear what ever the new instruction it is receiving. Doing so allows the whole pipeline stall for PC calculation for two clk cycles.