Sam Kleist

Final Project Report

ROB

Inputs:

The reorder buffer receives instruction inputs from the issue module. These inputs contain the instruction operation, the destination, and the stall bit which is used to determine whether or not to add the incoming instruction. This stall bit is set in the issue module. There are also inputs from the common data bus, these inputs contain the rob destination and the corresponding data. Also,

Outputs:

The reorder buffer outputs data to the register file, this is rob\_id, and rob\_data. This is sent to the regfile and updates the entries with the corresponding tag. There is also a control bit called CTRL\_flushRegFile. The control bit CTRL\_regFileDataReady specifies if data is ready to be sent to the register file. Similarly, the control bit CTRL\_storeDataReady specifies if data is ready to be sent to the memory unit to be stored. The data for storing is outputted in the bits rob\_store\_addr and rob\_store\_val. The Control bit CTRL\_rob\_full simply states if the reorder buffer is full.

Implementation:

7 different registers are added that account for the 4 different slots in the reorder buffer. These register store the operation occurring, the destination, the destination type (No Destination, Register, or Memory), the value, a ready bit indication if that slot is ready, an open bit displaying whether or not that slot is open, and a branch taken bit which specifies the branch was taken if that rob slot was holding a branch instruction.

When data is incoming from the instruction queue, a for loop iterates through the ROB and looks for an open slot, when an open slot is found, the incoming data is added to that slot.

When data is incoming from the common data bus, it is first checked if the operation being performed was a branch instruction. If so, the branch taken bit for that is set. Next the value in the rob slot is set to the incoming data from the common data bus.

At every positive edge of the clock, the first slot in the rob is checked if it is ready. The destination type is checked. If it is no destination, the branch taken bit is checked. If the bit is taken, the rob is flushed and the bi CTRL\_flushRegFile is set. Next if the destination type is a register, the reg file data ready bit is set to one and rob operation and rob data bits are set to equal the data corresponding with the first slot in the reorder buffer. Finally, if the destination type is memory, the store control bit is set and the corresponding store address and values are assigned. After the correct bits have been set, everything in the reorder is shifted up one level. This is causing the first slot in the rob to be overwritten and the last slot to be emptied. Finally the bit CTRL\_rob\_full is set. This bit specifies if the reorder buffer is full by checking if none of the slots are open.

Issue

Inputs:

The issue module is now inputted the CTRL bit specifying if the reorder buffer is full.

Implementation:

By orring the stall bit with the CTRL bit specifying if the reorder buffer is full, issuing will now stall whenever the reorder buffer is full. This is because you do not want to issue to the reorder buffer and be waiting for a reservation. We must stall until both the reservation station and rob are ready.

Regfile

Inputs:

Instead of regfile inputs coming from the common data bus, they are now from the reorder buffer. The regfile is also inputted the control bit CTRL\_flushRegFile. This bit is set in the reorder buffer when a branch has been taken and the register file needs to be flushed.

Implemetaion:

If the bit to flush the register file has been set, a for loop iterates through all the tags in the register file and sets them to 0.