Simulating DLX pipeline Hazards

-------------------------------

In this simulation, 2 types of hazards namely - Data & Control hazards were concentrated on.

A set of DLX instructions (program) may contain both data and control hazard.

As part of this implementation both are treated sperately. But, whenever control hazard is detected and if data hazard too present data hazard is handled first and then the control hazard is considered.

Wrt to Data Hazard, the choice is given to user for selection from 2 recovery algorithms

1 - Forwarding

2 - Compiler Instruction Re-orderdering

In forwarding technique, if a instruction is dependent on execution/memory/write back of another instruction with in the specified window, the result from the dependent instruction is passed to depending instruction prior to writing it back to memory. This reduces the number of stallings required. Its not always that forwarding technique eliminates stallings but it may reduce it.

In Compiler Instruction re-ordering is a dynamic scheduling technique, where the hardware mechanism of the processor rearranges the instruction execution to reduce the stalls.

Wrt to Control Hazard, initially the DLX instructions are checked and handled for data hazard(if present) and then control hazard is concentrated on. Control Hazard plays a crucial role in the execution of the program as a branch instruction may result in the branching or may not. So, in this simulation both the scenarios - branch taken and branch not taken are considered.

As per optimized data path, the branch condition and respective target address is pre-computed during the Instruction decode stage(ID).

Considering branch to be taken requires 3 stalls cycles in addition for each loop count. With branch not taken prediction, next instructions in the sequence is selected and executed, which will not require stalling as part of control hazard.

Wrt Branch delay slot scheduling algorithm, as in optimized datapath, the branch condition and respective target address is pre-computed during the Instruction decode stage(ID). Considering the branch to be taken all time, the next instruction after the branch instruction(if present) is chosen to execute before branch instruction to reduce unnecessary stalls needed by branch determination instruction.

Code Execution

---------------

Execution File : DLXPipelineHazard.java

Command line argument for the input file(instructions file) needs to be passed like - java DLXPipelineHazard DataHazard.txt

Instructions files added : 1 each for data & control hazard.

DataHazard.txt

ControlHazard.txt

Output listed on screen : Instructions as in input file

Pipelined instructions

Choice of algorithm

Selected Algorithm assumptions

Number of Clock cycles