COL216 Assignment-5

Satyam Kumar Modi , 2019CS50448
Siddhesh Kalekar, 2019CS50436

The aim of this assignment is to develop a multicore simulator with a efficiency in the throughput.

APPROACH USED

We have tried to maintain an **instruction buffer** in each core which stores the pending DRAM instructions of the core (which belongs to the same Row number) and then at the end of the instruction, we execute them. We alot the DRAM to the core in the first come first serve basis. We have only raised the DRAM request if the DRAM is busy in other core. We have use a **last pointer** in the instruction to maintain a check of the last non-executed instruction and then at the end of the DRAM cycle, we have updated the PC of the core.
At the beginning of the cycle, the instruction of each core is checked (if it is a DRAM instruction) which is done by the MRM. Whenever there is a data hazard, we put the core in a halt condition and then unhalt it when the DRAM instruction is complete.

BOUNDS ON THE INPUT

The no. of cores is bounded by 8.
The no. of cycles is bounded by 1000.

SOME DESIGN DECISIONS

1. Maximum size of instruction buffer has been set to 3.
2. The DRAM is allocated on the first come first serve basis.
3. The MRM takes a fixed amount of cycles per check which is 5 in our case.
4. Put the core into halt whenever we encounter a data hazard.
5. When the DRAM is executing the instructions in the buffer, then other cores are put on a halt.
6. The branch instructions cannot be executed simultaneously in a core which is using the DRAM.

Pros of the implementation :-
1. We have tried to use each clock cycle in an efficient way by implementing the forwarding technique.
2. We have used a finite buffer size to overcome the situation when the instruction of one core is executed very late.
3. Our technique uses the first-come-first-serve technique in the MRM, thus removing the case when any particular core being given more priority. Also, it takes care of the fact that the core which is having frequent DRAM Instructions gets executed first.
4. We have implemented the stalling in case of a Data hazard.

Cons of the implementation :-
1. We have not removed the redundant instructions from the instruction set.
2. We have not used the COL_ACCESS_DELAY in case of the instructions that are being executed in the buffer.
3. We have restricted forwarding only to the I and R-set instrictions. Using forwarding in the J-instructions may lead to unhandled addresses.
4. The performance may be a constant times less than the most optimal solution as we have used stalling in our technique.