COL-216 ASSIGNMENT-4

HARIKESH 2019CS10355

SOURAV 2019CS10404

APPROACH FOR CODE

- AS WE HAVE ALREADY DESIGNED A BASIC INTERPRETER IN C++ FOR HANDLING A SUBSET OF THE INSTRUCTIONS AND A MIPS SIMULATOR TOO. BASICALLY, WE ARE DEVELOPING A MAIN MEMORY AND THEN ARE INTEGRATING IT INTO MORE ADVANCED INTERPRETER THAN THE PREVIOUS ONE.
- NOW OUR MAIN MEMORY HAS TO TAKE UPTO 2^20 BYTES. SO, WE ARE BASICALLY STORING ADDRESSES INTO AN 2-D ARRAY OF SIZE 1024*1024.
- NOW OUR OTHERS INSTRUCTIONS I.E. EXCEPT LW AND SW ARE NOT AFFECETED BY THIS DRAM (DYNAMIC RANDOM-ACCESS MEMORY) IMPLEMENTATION AND STILL REQUIRE ONLY 1 CYCLE TO COMPLETE THAT OPERATION.
- IN CASE OF LW AND SW COMMANDS WE HAVE TO TAKE THIS CASE OF DRAM AND HENCE WE HAVE TO FIRST CHECK THE ROW NUMBER IN WHICH THE WORK OF OUR IMPORTANCE IS THERE AND AFTER THAT CHECK THE COLUMN OFFSET AND THESE 2 THINGS ARE BASICALLY GIVEN AS AN INPUT TO US WHICH IS ROW_ACCESS_DELAY AND COLUMN_ACCESS_DELAY AND HENCE WE ARE THEN CALCULATING THE TOTAL NUMBER OF CYCLES REQUIRED IN THIS CASE. NOT ONLY THIS, BUT THIS TIME WE ARE ALSO TRYING TO REDUCE THE TOTAL NUMBER OF CYCLES TAKEN IN SOME CASES OF MEMORY READ AND WRITE OPERATIONS. WE ARE DOING THAT BY STORING ALL SUCH REQUESTS IN A DATA STRUCTURE AND THEN RE-ORDERING ALL THOSE DRAM REQUESTS ACCORDING TO THE CONCEPT THAT WILL TAKE TWO SUCH INSTRUCTIONS OF SAME ROW TOGETHER AND THEN IMPLEMENT THAT.
- APART FROM THAT WE ARE MAPPING ALL THOSE INSTRUCTIONS AND THEIR REGISTERS WITH SOMETHING SO AS TO FOUND THOSE EASILY.
- THE TOTAL CYCLES REQUIRED IN THAT CASE CAN BE EASILY FOUND BY CONSIDERING THESE THINGS. NOW SUPPOSE WE ARE HAVING A PARTICULAR

row in our rowbuffer and the next instruction details also lie in that than we don't need to get that row back and then do the computation. We can basically update without going back. Else we have to go back in if it is not the same row and bring the other one.

- So, we have to do this till our file is completely read by us and during each cycle we have to print some basic details and finally after completing that we have to print the number of row updations and total number of cycles as in the previous assignment. We do this by keeping track of row number and cycle number.

Assumptions and Basic Notes

- I have assumed that there is no comment in the text file given to us but I am dealing the case of empty lines.
- Here we are taking all type of instructions unlike minor i.e. jump and other type of statements are included.
- Printing "Wrong Output" in case some register is out of scope or there is some kind of error in MIPS instructions. Also have taken the input format in the context that was given and asked for.
- The MIPS file must end with "exit:".

Testing Strategy

- We have tested our code for some easy text files apart from the one given by you on moodle which includes file with only 2 to 3 instructions and empty file (required error is printed in this case).
- Tested by giving wrong commands or out of scope registers and print necessary error.
- Taken some files with only sw and lw commands and otherwise.
- Take a file with a good number of instructions and just shuffled the instructions and tried to reduce the number of cycles as much as we can to make our code efficient.
- Checked the value of each and every register after running the code and confirmed it by doing some basic manual calculations.

STRENGTHS AND WEAKNESSES OF OUR CODE

- THIS IMPLEMENTATION IS HANDLING JUMP AND BRANCH STATEMENTS TOO UNLIKE THE PREVIOUS ONE. THERE IS ALSO A POSSIBILITY OF LITTLE MUCH INEFFICIENCY IN REORDING IN SOME TESTCASES.
- AS WE ARE FINDING OUT HOW MANY OTHER DRAM REQUESTS ARE THERE IN NEXT LINES OF THE GIVEN FILE (WE ARE SEARCHING UPTO THE EXTENT OF THE CYCLES TAKE IN FIRST LOAD OR STORE INSTRUCTION).SO IF SOME TESTCASE HAS THESE INSTRUCTIONS REPEATING CORRECTLY THEN OUR CODE IS INEFFICIENT BUT OUR CODE IS HIGHLY INEFFICIENT IN THOSE CASES WHERE THESE COMMANDS ARE NOT REPEATING SIMULTANEOUSLY.
- ALSO, WE ARE JUST GOING UPTO THE DELAYS TO CHECK ANY OTHER INSTRUCTION BELONGING TO THE SAME ROW SO AS TO REORDER THOSE REQUESTS TO MAKE OUR IMPLEMENTATION EFFICIENT BUT ONE CAN SAY THAT THERE ARE MILLIONS OF LINES IN REAL WORLD CODES SO MAYBE OUR CODE IS NOT SO EFFICIENT. BUT STRENGTH OF OUR CODE LIES IN THE FACT THAT ANY OTHER CODE GIVING EFFICIENT OUTPUT IN THAT CASE WILL REQUIRE ADVANCED HARDWARE AND HENCE WILL HAVE MORE COST THAN OURS AND ALSO IT WILL REQUIRE MORE SPACE THAN OURS.
- WE ARE USING NON-BLOCKING MEMORY IN OUR CASE SO ANY INDEPENDENT INSTRUCTION WILL NOT REQUIRE AN EXTRA CYCLE AND WILL BE DONE ALONGSIDE DRAM AND THUS MAKE OUR CODE EFFICIENT.