COL216 Assignment 2 report

Team members :

Piyush Chauhan 2021CS11010

Shankh Gupta 2021CS50604

Overview and Design Decisions

The goal of the assignment was to simulate a 5 stage and 7-9 stage pipeline in C++. Following is the design decision made by us in our implementation.

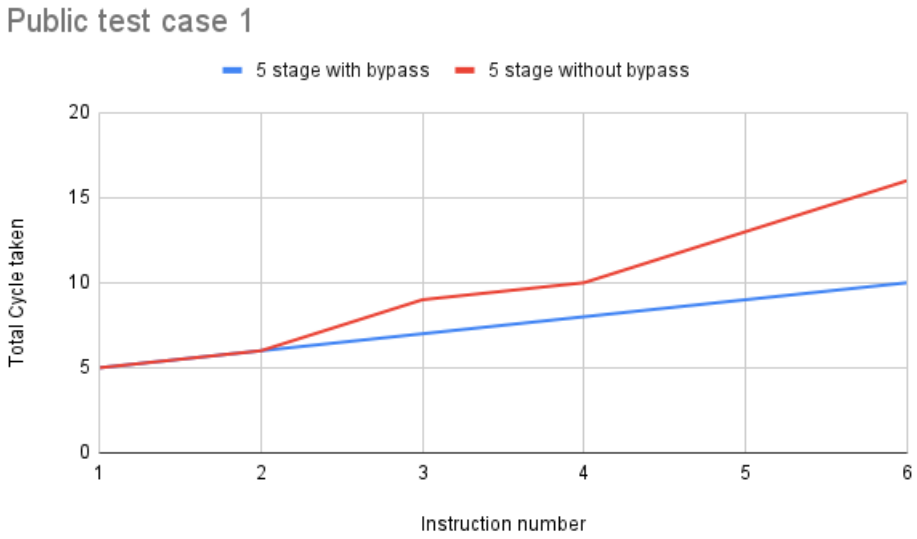
1) We have implemented 23 instructions in our simulator.
2) We had made a header file for each state, and they have used it in the main files. The header files of stage 5 without forwarding were reused in with forwarding with slight modifications. The functions in the header files were split accoringly to make a 7-9 pipeline. IF EX and WB were used in all the main files as it is.
3) We have made a test file for registers value and data memory for easy debugging.
4) A new instruction will begin from a new line
5) The register name in the instruction can be either the standard MIPS register name or it can be $0, $1 to $ 31. Note that one cannot write in a $zero or $0 register.
6) A special instruction is being added called “end” which will denote the end of the instruction set. Users need not write “end”, it will automatically be appended by our program.
7) Instruction decode state just decodes the values of the register and checks for stalls and forwarding.
8) All the branch decisions are made in the Execution stage.
9) If there is a jump instruction, then the instruction of the destination can be fetched in the same cycle. Thus, it reduces the cycle count.

Testing the pipeline

Testing 5 stage pipeline.

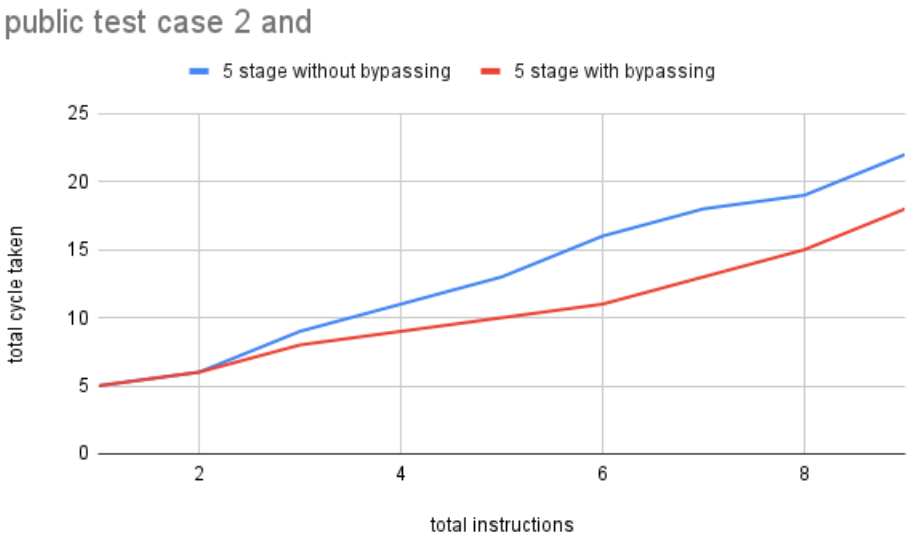
PUBLIC TEST CASE 1:

- For public test case 1 total cycle taken without forwarding was 16.
- Cycles taken with forwarding were 11.
- This is because instruction 3 was dependent on the output of the 2nd instruction. Similarly with instruction 5 was dependent on 4 and 6th was dependent on 5th instruction.



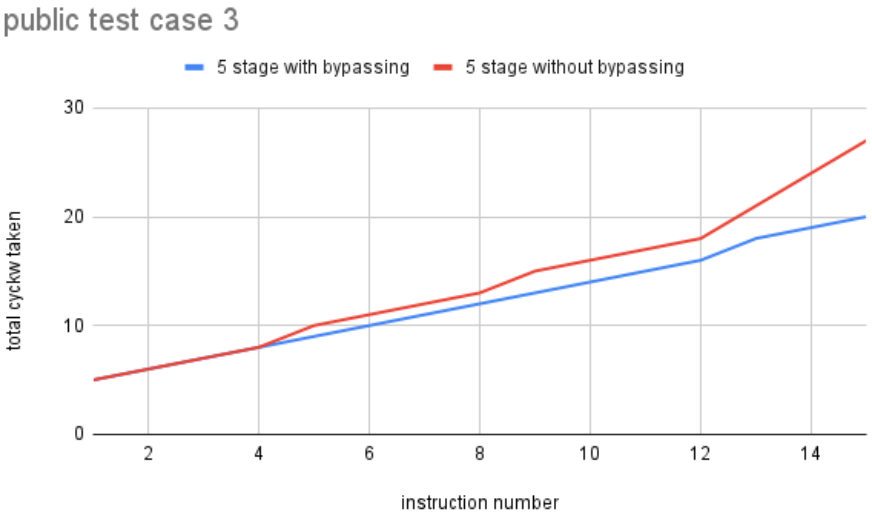
PUBLIC TEST CASE 2:

- For public test case 2 total cycle taken without forwarding was 22 (including the initial all nil stage)
- With forwarding the total cycle time reduces to 19 (including the initial all nil stage).



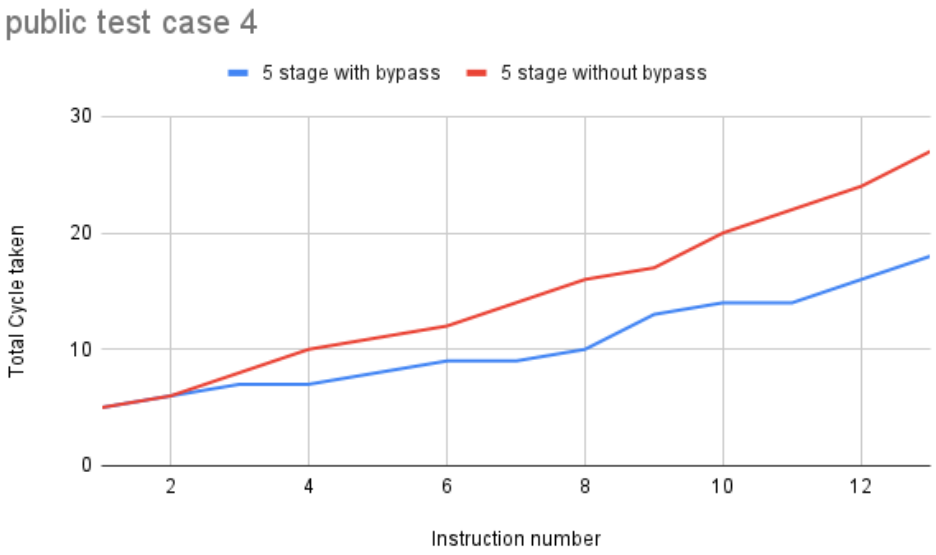
PUBLIC TEST CASE 3:

- For public test case 2 total cycle taken without forwarding was 28 (including the initial all nil stage)
- With forwarding the total cycle time reduces to 21 (including the initial all nil stage).



PUBLIC TEST CASE 4:

- For public test case 2 total cycle taken without forwarding was 28 (including the initial all nil stage)
- With forwarding the total cycle time reduces to 19 (including the initial all nil stage).

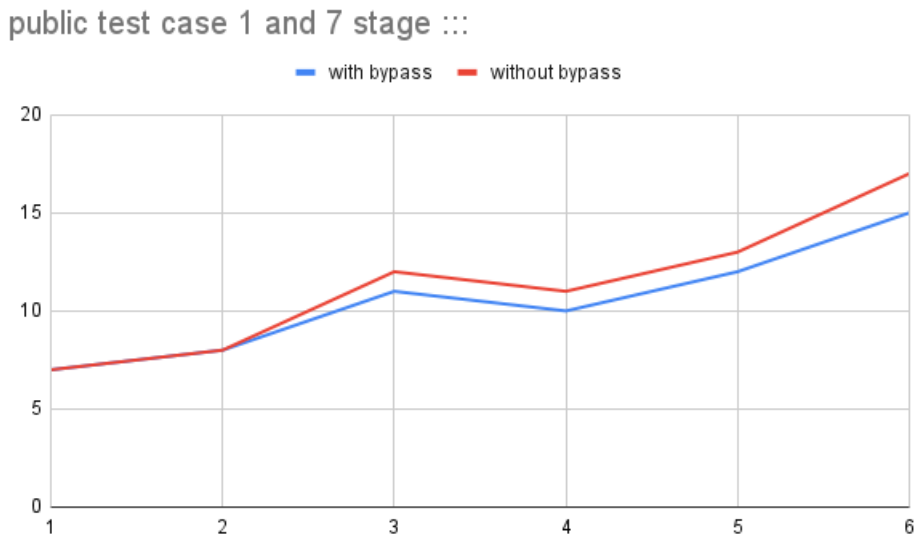


Conclusions:

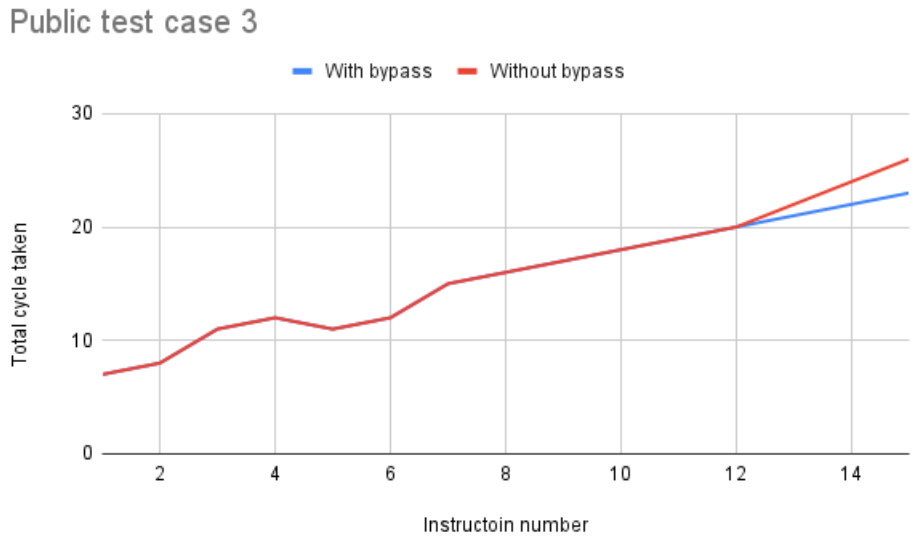
- From all the cases it is prominent that 5 stage pipelines with bypassing performs better than a non-bypassed implementation.
- From test case 3 and 4 we can conclude that if a “sw” is followed by a “lw” instruction then it reduces the cycle time as compared to the case when a “sw” are together and “lw” are together.
- In branched instruction (Test 2), if the registers used in branch instruction are calculated just above the branch instruction, then bypassing helps in reducing the 2 stalls before the branch instruction.

Testing the 7-9 stage pipeline:

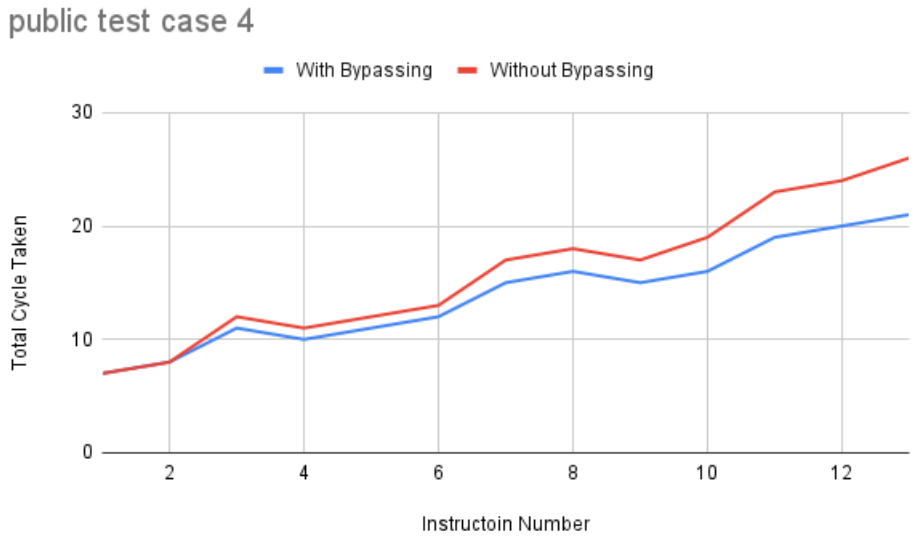
Public test case 1:



Public test case 3:



Public Test Case 4:



BRANCH PREDICTOR:

We were required to make 3 branch predictors in this part and test their accuracy. The branch trace file given contained 548 branch addresses out of which 379 were marked as taken and 169 were marked as not taken.

Saturating Branch predictor

Accuracy of the SaturatingBranchPredictor is

Starting state of the counter (in binary)	00	01	10	11
Correct Prediction	433	490	513	475
Accuracy (%)	79.0146	83.9416	87.9562	86.6788

BHR Branch predictor

Accuracy of the BHRBranchPredictor is:

Starting state of the bhr(in binary)	00	01	10	11
Correct Prediction	392	396	398	399
Accuracy (%)	71.5328	72.2628	72.6227	72.8102

SaturatingBHR Branch Predictor:

Our accuracy is

Starting state of the bhr(in binary)	00	01	10	11
Correct Prediction	396	410	409	399
Accuracy (%)	72.2628	74.8175	74.6300	74.635