216 Assignment 2

Sanya Mittal(2021CS10565) and **Anshik Sahu(2021CS10577)**

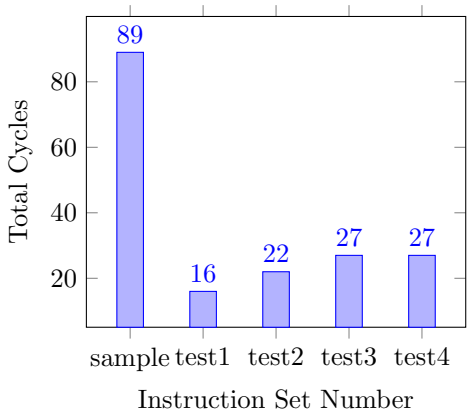
April 15, 2023

1 Abstract

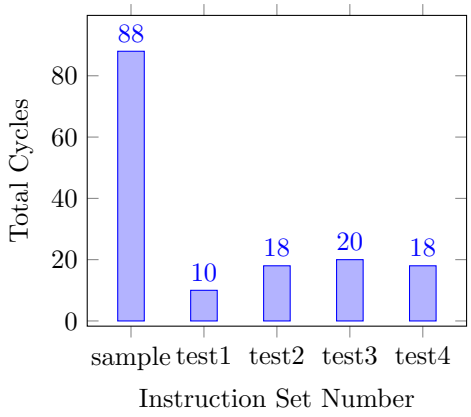
In this project, we have designed a pipeline simulator for MIPS Instructions and used it to simulate a 5 stage and a 7-9 stage pipeline with and without bypassing. We have analysed and compared the observations for all the cases on 5 different input files. We have also implemented a Branch Predictor which can optimise the pipeline and decrease the number of required cycles. We have designed three types of Predictors based on 3 different algorithms and reported their accuracies.

2 Graphs

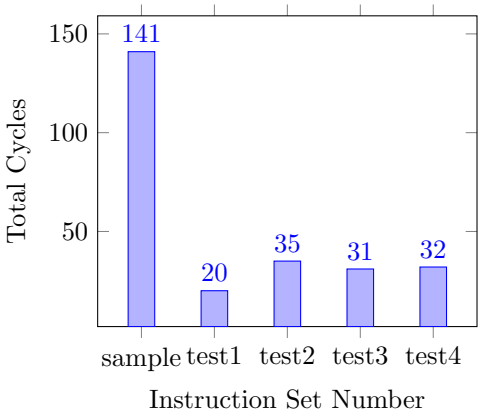
This is the graph for the number of cycles in **5 stage pipeline without bypassing**.



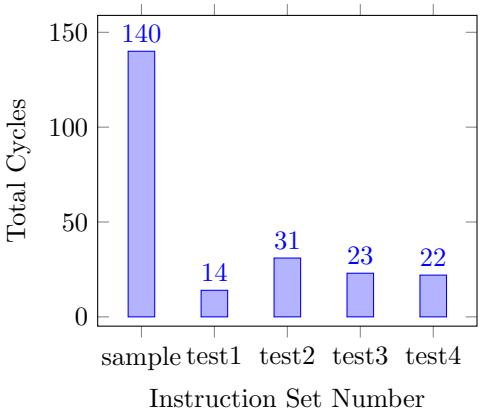
This is the graph for the number of cycles in **5 stage pipeline with bypassing**.



This is the graph for the number of cycles in **7 - 9 stage pipeline without bypassing**.



This is the graph for the number of cycles in **7-9 stage pipeline with bypassing**.



3 List of Observations

- As expected theoretically, for all the test cases when **forwarding/bypassing is active, the number of cycles reduces significantly**. As in the case of Data Dependency, an instruction does not have to wait for a previous instruction to complete to get a value from it, we can bypass the value from the latch at the point of production to the point of consumption; reducing the number of stalls.

- From the graphs, we can observe that the decrement in the number of cycles is the maximum for the file **public_test4** when switching from without bypassing to bypassing(both in 5 and 7-9 pipeline). This shows that there is **maximum data dependency in that instruction set**. And similarly the **minimum data dependency is in sample.asm**/

- In a 7-9 pipeline, the number of stages increased for each instruction over the 5-stage pipeline. Thus the **number of cycles for a given instruction set should increase in a 7-9 stage pipeline** as can be verified from the graphs.

- The number of instructions for files public_test1, public_test2, public_test3 and public_test4 are 6,9,15 and 13 respectively. Public_test2 takes more cycles than public_test3 despite having less number of instructions. This is because the first file has **beq and bne instructions as we need to wait till the EX stage** to know if the branch was taken or not.

- In going from 5 to 7-9 stage pipeline the total time increases when the branching does not dominate and when the Instruction set has significant branching then total time increases as number of cycles increase significantly.

File Name	sample	test1	test2	test3	test4
5stage	19580ns	3520ns	4840ns	5940ns	5940ns
5stage_bypass	19360ns	2200ns	3960ns	4400ns	3960ns
79stage	23970ns	3400ns	5950ns	5270ns	5440ns
79stage_bypass	23800ns	2380ns	5270ns	3910ns	3740ns

Table 1: Total Execution Times for various input files and pipelines

Length of stages:
For 5 stage pipeline:= IF: 70ns, ID: 100ns, EX: 150ns, MEM: 200ns, WB: 120ns
For 7-9 stage pipeline:= IF1: 35ns, IF2: 35ns, ID1: 33ns, ID2: 33ns, EX: 150ns, MEM1: 100ns, MEM2: 100ns, WB: 120ns
Length of latches: 20ns

As mentioned above there can be a significant amount of branching expected in files sample.asm and public_test2.asm, which is supported by the increase in total execution times for both with and without bypassing. Green deviates from the trend.

4 Representation of the Execution of Pipeline

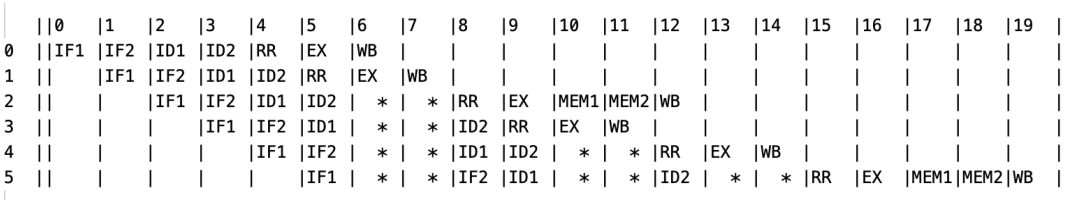
These are representations for the output of public_test1.asm

```
   ||0   |1   |2   |3   |4   |5   |6   |7   |8   |9   |10  |11  |12  |13  |14  |15  |
0  ||IF  |ID  |EX  |ME  |WB  |    |    |    |    |    |    |    |    |    |    |    |
1  ||    |IF  |ID  |EX  |ME  |WB  |    |    |    |    |    |    |    |    |    |    |
2  ||    |    |IF  |ID  |  * |  * |EX  |ME  |WB  |    |    |    |    |    |    |    |
3  ||    |    |    |IF  |  * |  * |ID  |EX  |ME  |WB  |    |    |    |    |    |    |
4  ||    |    |    |    |    |    |IF  |ID  |  * |  * |EX  |ME  |WB  |    |    |    |
5  ||    |    |    |    |    |    |    |IF  |  * |  * |ID  |  * |  * |EX  |ME  |WB  |
```

Figure 1: 5 Stage Pipeline Without Bypassing.

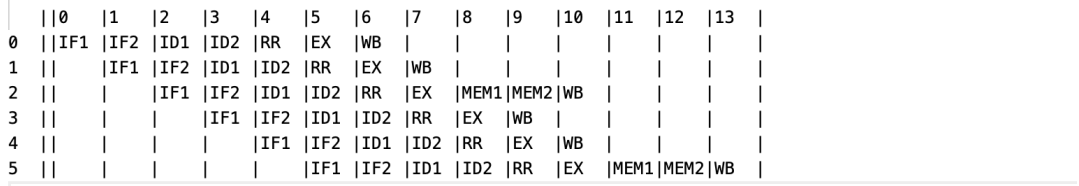
```
   ||0   |1   |2   |3   |4   |5   |6   |7   |8   |9   |
0  ||IF  |ID  |EX  |ME  |WB  |    |    |    |    |    |
1  ||    |IF  |ID  |EX  |ME  |WB  |    |    |    |    |
2  ||    |    |IF  |ID  |EX  |ME  |WB  |    |    |    |
3  ||    |    |    |IF  |ID  |EX  |ME  |WB  |    |    |
4  ||    |    |    |    |IF  |ID  |EX  |ME  |WB  |    |
5  ||    |    |    |    |    |IF  |ID  |EX  |ME  |WB  |
```

Figure 2: 5 Stage Pipeline With Bypassing.



```
   ||0   |1   |2   |3   |4   |5   |6   |7   |8   |9   |10  |11  |12  |13  |14  |15  |16  |17  |18  |19  |
0  ||IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |    |    |    |    |    |    |    |    |    |    |
1  ||    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |    |    |    |    |    |    |    |    |    |
2  ||    |    |IF1 |IF2 |ID1 |ID2 |  *  |  *  |RR  |EX  |MEM1|MEM2|WB  |    |    |    |    |    |    |    |
3  ||    |    |    |IF1 |IF2 |ID1 |  *  |  *  |ID2 |RR  |EX  |WB  |    |    |    |    |    |    |    |    |
4  ||    |    |    |    |IF1 |IF2 |  *  |  *  |ID1 |ID2 |  *  |  *  |RR  |EX  |WB  |    |    |    |    |    |
5  ||    |    |    |    |    |IF1 |  *  |  *  |IF2 |ID1 |  *  |  *  |ID2 |  *  |  *  |RR  |EX  |MEM1|MEM2|WB  |
```

Figure 3: 7-9 Stage Pipeline Without Bypassing.



```
   ||0   |1   |2   |3   |4   |5   |6   |7   |8   |9   |10  |11  |12  |13  |
0  ||IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |    |    |    |    |
1  ||    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |    |    |    |
2  ||    |    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |MEM1|MEM2|WB  |    |    |    |
3  ||    |    |    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |    |
4  ||    |    |    |    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |WB  |    |    |    |
5  ||    |    |    |    |    |IF1 |IF2 |ID1 |ID2 |RR  |EX  |MEM1|MEM2|WB  |
```

Figure 4: 7-9 Stage Pipeline With Bypassing.

5 Table for Prediction Data

	00	**01**	**10**	**11**
SaturatingBranchPredictor	79.0146%	83.9416%	87.9562%	86.6788%
BHRBranchPredictor	71.5328%	72.2628%	72.6277%	72.8102%
SaturatingBHRBranchPredictor	71.5328%	82.2993%	87.5912%	86.1314%
SaturatingBHRBranchPredictor2	79.014%	83.9416%	87.2263%	85.7664%

Table 2: Accuracies for various prediction strategies

- In the above table, the columns are the start states of the counter and the rows are the various prediction strategies used. The four strategies used are as follows :
  - **Saturating Branch Predictor** maintains 2^{14} counters, each indexed by the last 14 bits of the Program Counter. Corresponding to each branch, we go to its counter and predict branch taken if value of the counter is greater than or equal to 2 and not taken otherwise. If the branch is taken, the counter is incremented by one (if not in the highest state i.e. 3) else the counter is decremented by one (if not in the lowest state i.e. 0)
  - **BHR Branch Predictor** maintains 4 counters, each corresponding to one value of the Branch History Register(bhr). Corresponding to each bhr, we go to its counter and predict branch taken if value of the counter is greater than or equal to 2 and not taken otherwise. If the branch is taken, the counter is incremented by one (if not in the

highest state i.e. 3) else the counter is decremented by one (if not in the lowest state i.e. 0). Bhr is updated according to the expected result of the latest branch.

- **SaturatingBHRBranch Predictor** maintains 2^{16} counters, each indexed by the bit combination of the last 14 bits of the Program Counter and value stored in bhr. Corresponding to each index, we go to its counter and predict the branch to be taken if the value of the counter is greater than or equal to 2 and not taken otherwise. If the branch is taken, the counter is incremented by one (if not in the highest state i.e. 3) else the counter is decremented by one (if not in the lowest state i.e. 0)
- **SaturatingBHRBranch Predictor2** maintains 2^{16} counters, each indexed by the bit combination of the last 14 bits of the Program Counter and value stored in bhr. Corresponding to each index, we go to its counter and predict if we would use Saturating Branch Predictor or BHR Branch Predictor for predicting taken/not taken. We choose to predict using Saturating Branch Predictor in case the counter value is less than 1 and go to the value of table and predict the value according to its counter, and do the same in the corresponding case for BHR Branch Predcitor.

- Saturating Branch Predictor is expected to be more accurate than BHR Branch Predictor as it makes predictions based on each branch's specific history over the overall history. This is evident from the data.
- Ideally the maximum accuracy is expected from the SaturatingBHRBranch Predictor2 as it predicts whether to use Saturating Predictor or BHR Predictor and then makes the prediction. But since predictions are dataset dependent thus a slight error can be explained.

6 Token Distribution

Name	Tokens
Sanya Mittal(2021CS10565)	50%
Anshik Sahu(2021CS10577)	50%