COMPENG 4DM4: Computer Communication Networks

Lab 2: Processor Pipline

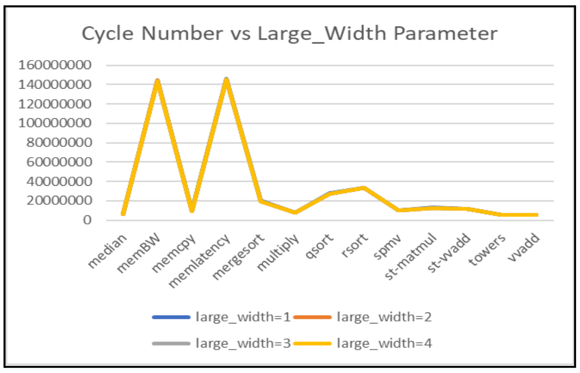
Shanzeb Immad - 400382928 - immads

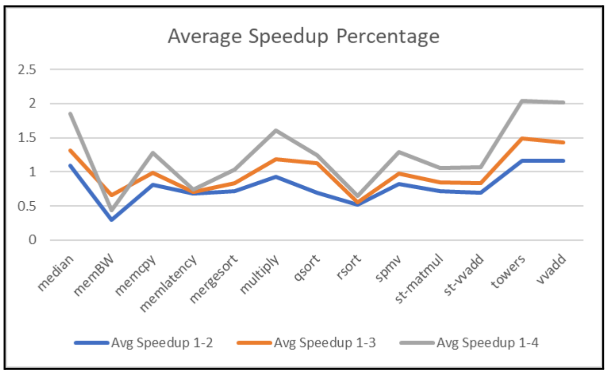
Josh Umansky - 400234265 - umanskyj

Exercise 3 - Superscalar Processors and Pipeline width Impact:

3.1 Whole Pipeline Width

The large width parameter, which by default is set as 2, but varies depending on the param.in file, controls the pipeline width (where the entire pipelines use the same width, as per the Macsim documentation). The width determines the number of pipelines available for instructions to be run in conjunction with one another. As shown, the impact on the benchmarks is visible as a small decrease in the overall cycles taken (*Image 1)*, and better illustrated by looking at the average speedup per benchmark (*Image 2)*, with the large_width = 4 representing the largest speedup for all benchmarks by an average of 1.25% compared to the baseline of large_width = 1.





Based on the findings experimentally, it would be reasonable to assume that the individual benchmarks should execute in less cycle times, as more instructions could be done in the additional pipelines available as the large_width parameter increases.

3.2 Width of different functional units

The groupings of the specified parameters are listed below:

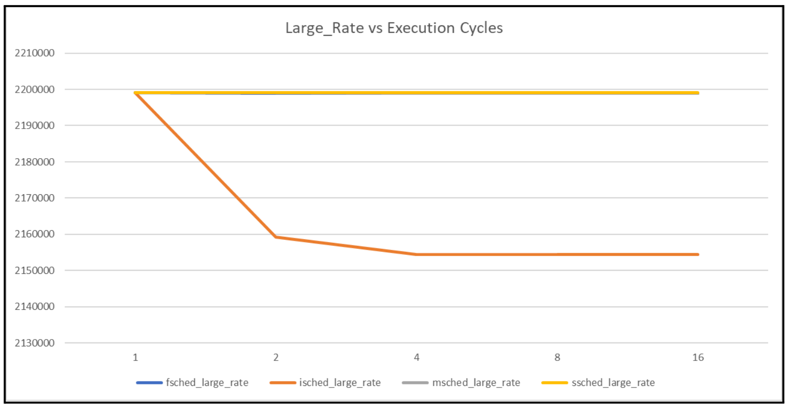
***sched_large_size group;**

- Fsched_large_size
- Isched_large_size
- Msched_large_size
- ssched_large_size

***sched_large_rate group:**

- Fsched_large_rate - Number of floating point instructions that can be executed per cycle
- Isched_large_rate - Number of integer instructions that can be executed per cycle
- msched_large_rate - Number of memory instructions that can be executed per cycle
- Ssched_large_rate - Not referenced in documentation

By executing a sweep of values from 1-16, in 2^x intervals, the following data can be obtained. It would follow that at a higher rate, the less number of cycles a program would take, as we allow Macsim to run more instructions per cycle of a specific instruction type. Summarising the graph, the data shows that three of the rate's have very little effect on the overall cycle count of the benchmark (Fsched - 205 cycles saved, Msched - 72 cycles saved, Ssched - 0 cycles saved). The isched_large_rate has the largest cycle saving effect on the benchmark, saving ~45000 cycles. This correlates to what we would expect, considering from lab 1, the most common UOP used in mergesort was UOP_IADD, which is an integer addition. This relates to the isched_large_rate, as it allows more integer instructions to be executed per cycle.

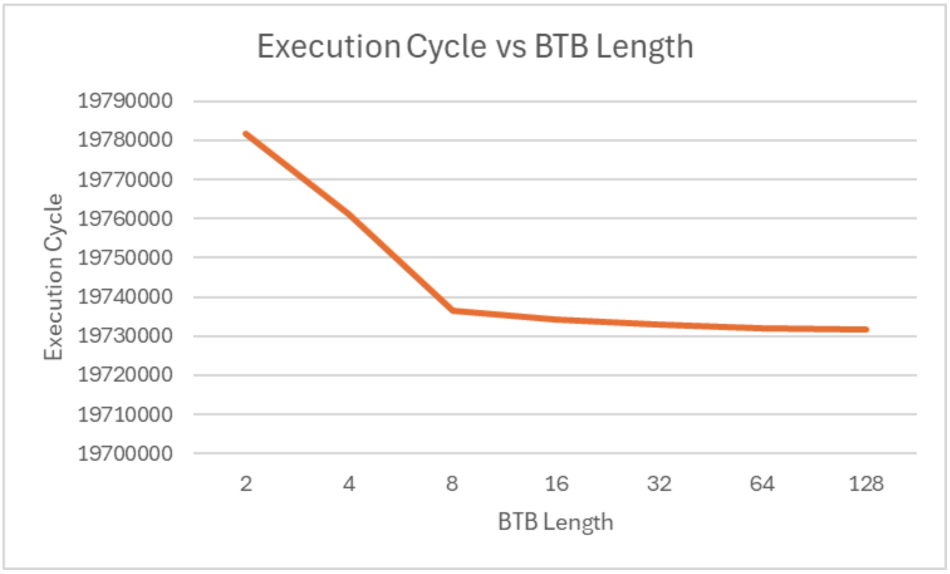


Exercise 4 - Branch Predictor:

4.1.1 BTB Impact

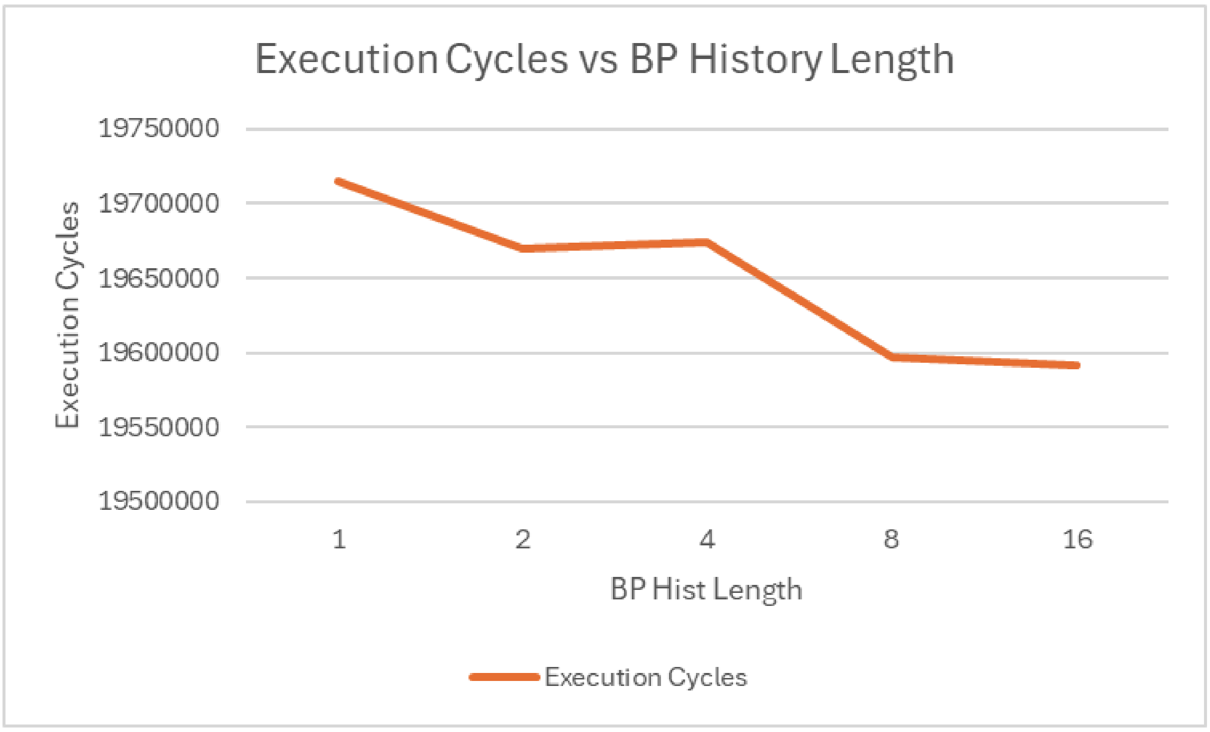
In the first part of the experiment, the execution cycle was observed for when BTB was disabled, on and perfect and on and imperfect. The results revealed the execution cycles to be at the largest number when the BTB is disabled. This follows the theory that without the Branch Target Buffer, the CPU cannot predict the target address so more cycles occur to figure it out. When BTB is on but not perfect, the cycle count drops significantly by a value of 300,000. This makes sense as now target addresses can be predicted and less cycles will have to be used on unnecessary computations. Finally the BTB is set to perfect and the cycle count decreased yet again by another 20,000, this indicates that when it was non perfect, some of the predictions made were incorrect.

The second part requires the entry count for the BTB to be swept with the values of 1,2,4,8,16,32,64,128. The plot below shows the relation of the execution count with the BTB entries. The trend observed is that with lower entries, the table can not hold many predictions which must lead to many mispredictions. As the entries increase, the instruction count approaches the same value as when BTB was perfect with 1024 entries..



4.2 gshare Branch Predictor Impact
In this part, the same process is followed from 4.1 but instead of BTB we look at the results from the Branch Predictor. In the first part, the instruction count is obtained for the BP when it is not in use, perfect and imperfect. When BP is disabled there is a higher instruction count, this is due to the fact that it must wait for the branch to finish to move on instead of predicting the results. As the predictor is then made to be enabled yet imperfect, the cycles decrease by a bit, this occurs as some predictions are being made but since it is not perfect, some predictions are wrong. Finally when BP is enabled and perfect, the cycle is even lower than the cases before. This theoretically makes sense as in this circumstance the predictions being made are always correct.

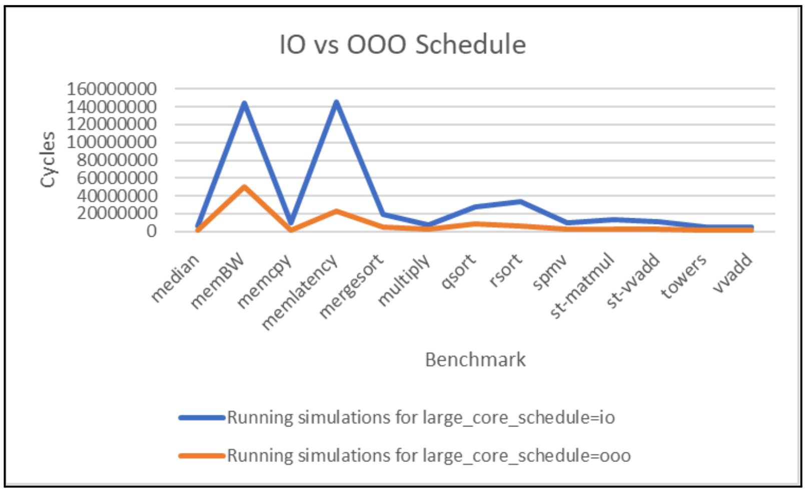
In this case, the variable BP_hist_length is what is now being swept, with the values of, 1, 2, 4, 8, 16. This parameter being changed, is the length of the history the branch predictor uses, so by increasing it, theoretically the execution count should decrease. This is due to the fact that with more access to previous branches, more accurate predictions can then be made. This is confirmed by the plot below showing the drastic decrease while the history increases.



Exercise 5 - Pipeline Type: In Order vs Out-of-Order:

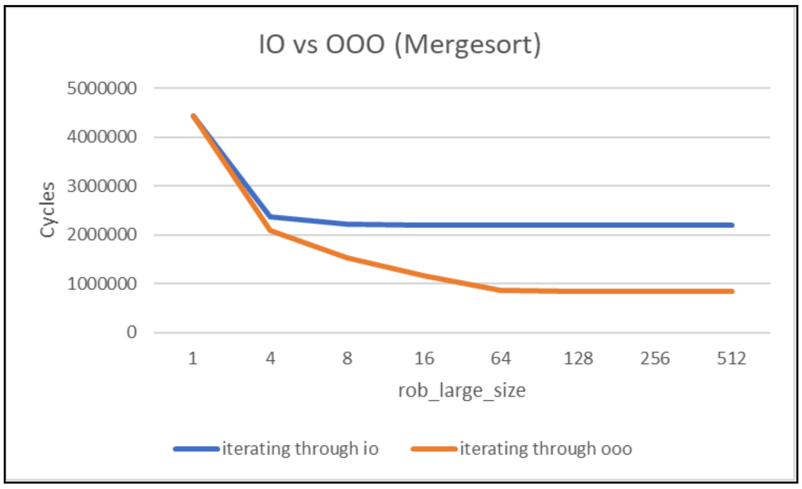
5.1 IO vs OOO

As can be noted from the results shown below, the impact of IO (in order) and OOO (out of order) operation is evident. The out of order execution results in a faster cycle time across every benchmark, given to the fact that instructions can be overlapped, and ran in a faster order then the in order execution. This results in massive savings in total cycles, especially for the larger benchmarks like memBW, memlatency, qsort and rsort.



5.2 Impact of ROB

From the data collected, the ROB buffer size has a direct effect on the cycles that the mergesort benchmark takes to execute. The effect is more pronounced with the out of order execution style, as a larger buffer size allows more instructions to be reordered, and processed out of order. The effect on the in-order execution is less pronounced as the ROB size increases, as the instructions cannot be re-ordered, and must only be executed at most four at a time (since the large_width parameter is set to 4).


IO vs OOO (Mergesort)
Cycles
rob_large_size
iterating through io
iterating through ooo