ECE552 Lab4 report
Francis Guo 1005931397
Rongbo Zhang 1005273393

Q1.
For the choice of the microbenchmark for the next-line prefetcher, we had an array of 1,000,000 elements. The elements are int type. Then when we are looping through this array, when we set the step of looping equal to 1, which means that we are stepping 4 bytes everytime we loop, we are getting a very small miss rate. With this step size, the next data is in the next block, so the next-line prefetcher is prefetching the correct block. In the tests we got 0% of the missing rate. (0% is rounded number, there are some cache misses) And when we are stepping 128 elements which are 4*128 = 512 bytes. The step size is larger than the block size, so we are getting a higher cache miss rate which is 67.51%. There is a reduction of the miss rate when the step size is smaller than the block size. This shows that the next-line prefetcher is working correctly.

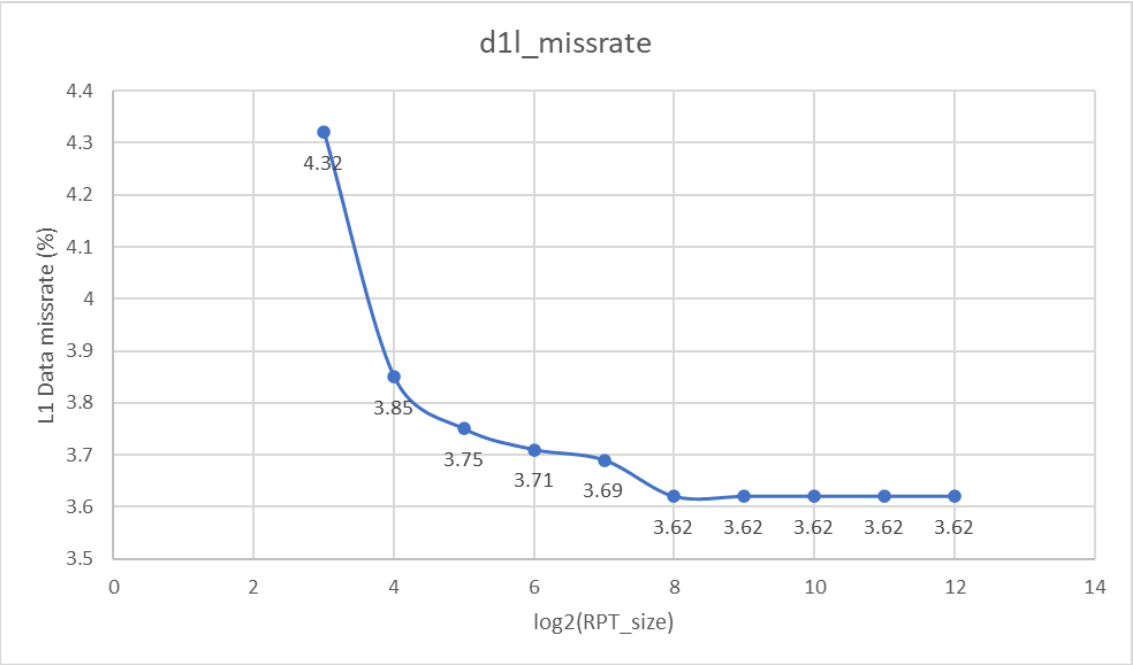
Q2
For the micro benchmark for the stride prefetcher, we had an array of 1,000,000 elements of int. We are looping through this array, and we used a switching step, which means when we are looping through the array. If the first time we used a step of 64 elements which are 256 bytes, then the next time we will use a step of 128 which are 512 bytes. In this case, we are getting a high miss rate which is 11.98%, because the constant switching causes the FSM to end up in the No-Pred state. Then, we change it to a constant step of 64 element which is 64*4 = 256 bytes. In this case we have a miss rate of 0.5%, because the constant switching causes the FSM to end up in the Steady state. With these results, we could confirm that the stride prefetcher is working properly.

Q3
$T_{avg} = T_{L1-hit} + MR_{L1} * (T_{L2-hit} + MR_{L2} * T_{Memory})$
$T_{avg} = 1 + MR_{L1} * (10 + MR_{L2} * 100)$

Config	L1 Miss Rate	L2 Miss Rate	Average access time
no-prefetch	4.16%	11.40%	1.89
next-line	4.19%	8.38%	1.77
stride	3.85%	5.78%	1.61

Q4

The following figure shows how changing the RPT table size affects the miss rate.
The size is in log2 of the RPT table size.



With the RPT table size increase in size, we have a miss rate decrease. However, with RPT table size larger than 2^8 we can see diminishing returns in reducing the miss rate due to aliasing in the RPT. Thus, 2^8 is a good size for the RPT for the given benchmark.

Q5

I am considering adding the accuracy and coverage of the prefetcher, and the average access time as a result of the prefetcher. By looking at the accuracy we can study if the prefetcher pollutes the cache with unnecessary data by evicting useful data. By looking at the coverage we can study how many misses did it save. By looking at the average access time we can study the performance gain of adding the prefetcher.

Q6

For the micro benchmark for the open-ended prefetcher. We had an array of 1,000,000 elements of int. Then we are looping through this array. In this micro benchmark, we used a patterned switching step, which means when we are looping through the array, we will have a pattern of 10 iter of the array. The steps are shown in a pattern of {10, 10, 10, 64, 64…}. In this case, we could see that our open prefetcher has 0.42% of miss rate and the stride prefetcher causes a 1.04% of miss rate. This is due to the fact that our open-ended prefetcher only pre-fetches when it is in steady state.

For the open-ended prefetcher, the open-ended prefetcher we implemented a modified stride prefetcher. Here, we have implemented a stride prefetcher that we change the size of the rpt table to 1K. We have also changed the output of the state for the stride predictor. We change

to only prefetch when we are in the Steady state. This means that prefetches are only allowed when the prefetcher is very confident about the prediction for the address of prediction.This leads to a 1.97% of average miss rate across the three test benches.

For this open predictor, we have the RPT with 1024 entries. The RPT table size is 16 Byte * 1024 = 16,384 Bytes. This is 16K bytes. This is similar to the L1 data cache. So, this open prefetcher is feasible to be manufactured and added to the processor. From the CACTIA result, we have the following for the RPT table:

```
Access time (ns): 0.492951
Cycle time (ns):  0.443181
Total dynamic read energy per access (nJ): 0.00433307
Total leakage power of a bank (mW): 5.33019
Cache height x width (mm): 0.134653 x 0.206732
```

The leakage power and size of the RPT is feasible to apply to the processor. The access time is similar to the L1 data cache. Since we assume there is a memory hierarchy, the prefetch happens when there is a cache miss in L1. The access time to the lower level is much longer, so the prefetch should not be the critical path that affects the clock frequency.

Contribution

Rongbo Zhang:

1. Next-line prefetcher debug
2. Stride prefetcher
3. Stride prefetcher debug
4. Open-ended prefetcher debug
5. Report

Francis:

1. Next-line prefetcher
2. Stride prefetcher
3. Open-ended prefetcher
4. Micro-testbench
5. Report