CSE3666 — Lab 9

Mike Medved

April 30th, 2022

1 Prompt

In this lab, we experiment with data cache simulator in RARS. It helps us understand how cache works.

The RISC-V program we will use in experiment is lab9-cache.s. The program reads words from a word array warray for 10240 times, starting from warray[0]. The index is incremented by the stride size after each read. The default stride size is 1. If the index is out of range, it is reset to 0. The pseudocode is as follows.

```
index = 0
for i in range(10240)
   read warray[index]
   index += stride
   if index >= len(warray)
      index = 0
```

2 Task 1

I predict that cache hit rate will be 75%, as there are four cache operations per block offset, and three of them will succeed since the first operation is in place to put the element into cache.

Table 1: Cache Simulation

address	**cache index**	**tag**	**block offset**	**result**
0x10010000	0	0x00200200	0	miss
0x10010004	0	0x00200200	0	hit
0x10010008	0	0x00200200	0	hit
0x1001000C	0	0x00200200	0	hit
0x10010010	0	0x00200200	1	miss
0x10010014	0	0x00200200	1	hit
0x10010018	0	0x00200200	1	hit
0x1001001C	0	0x00200200	1	hit
0x10010020	0	0x00200200	2	miss
0x10010024	0	0x00200200	2	hit

The number of bits per field is given by the above table, with the complete cache address being 32 bits, the cache index being **3 bits**, the tag being **25 bits**, and the block offset being **4 bits**. I predicted that the cache hit rate will be 75%, and by observing the above data, we can clearly see that the observation was confirmed since 3/4 operations per block offset were successful, while one missed.

3 Task 2

I predict that the cache hit rate will be 75%, as before due to the same reason - since the block size of 4 words, the first access will always miss, and the subsequent accesses will hit, giving a 3/4 success rate.

Table 2: Cache Simulation

blocks	**cache size**	**hit rate**	**misses**
8	128 bytes	75%	2560
16	256 bytes	75%	2560
32	512 bytes	75%	2560
64	1024 bytes	99%	64
128	2048 bytes	99%	64

By observing the above collected data, we can clearly see that the cache hit rate is 75% when the array size is greater than that of the cache size. However, once the cache size exceeds that of the array size, the cache hit rate increases substantially to nearly 100% since the cache can be populated with all array values, thus mitigating the inherent reason that the smaller cache missed 25% of the time.

In order to predict the cache hit and miss rates if the size of *warray* is increased by arbitrary amounts we can use the information we previous learned about caches. For example, if $size_{cache} < size_{array}$, then the cache hit rate will be constant since the cache will need to be repopulated periodically when it fills up. However, if $size_{cache} > size_{array}$, all array items can be populated into the cache, and therefore the cache hit rate will be close to 100%.

4 Task 3

I predict that the cache hit rate will be 88%, since the cache block size is doubled the ratio of hit/miss will be 7/8 which comes out to be roughly 88%.

Table 3: Cache Simulation

blocks	**cache size**	**hit rate**	**misses**
8	256 bytes	88%	1280
16	512 bytes	88%	1280
32	1024 bytes	100%	32
64	2048 bytes	100%	32
128	4096 bytes	100%	32

As before, the above data confirms that the cache hit rate is 88% when the cache size is smaller than the array size. However, if the cache size is larger than the array size, the cache hit rate will be close to 100% since the cache can be populated with all array values, thus mitigating the inherent reason that the smaller cache missed 25% of the time.