Cache Experiments

Aditya Tripathy, EE24BTECH11001

November 2025

1 Basic Assumptions about the Tunable Cache Options

- Lines : This refers to the number of unique indices in the cache
- Ways : Refers to the associativity of the cache, with 1 being the option corresponding to a direct mapped cache
- Words per Line : This gives us the block size, which is the amount of data fetched from the main memory in the case of a cache miss.

2 Cache Reads

2.1 Program 1

2.1.1 Understanding the Memory Access Pattern

```
.data
L1: .dword 8, 8
A:

.text
la x3, L1
ld x4, 0(x3)
ld x5, 8(x3)
addi x3, x3, 16

addi x10, x0, 0
add x12, x3, x0
If1: beq x4, x10, end1
    addi x11, x0, 0
If2: beq x5, x11, end2
    ld x20, 0(x12)
    addi x12, x12, 8
```

```
18      addi  x11, x11, 1
19      beq  x0, x0, If2
20  end2: addi  x10, x10, 1
21      beq  x0, x0, If1
22  end1: nop
```

Listing 1: RISC-V Assembly Code

The access pattern is linear, where memory locations are accessed in the order `0x1010`, `0x10.5`, `0x1020`, $\cdots$. So, there is no temporal locality. Therefore we would like to fetch higher memory addresses in advance on a cache miss to improve hit rate.

2.1.2 Varying Number of Lines

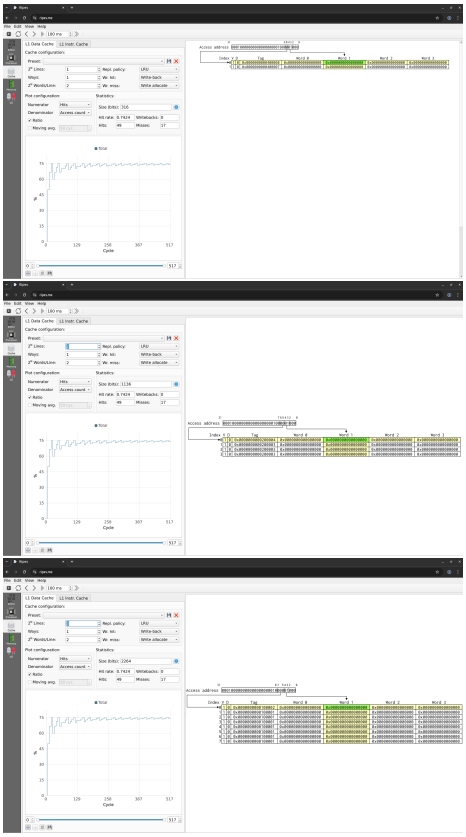


Figure 1: Increasing number of indices does not affect hit rate

We see that the cache hit rate remains exactly the same. This is expected due to the access pattern. Since the block size is 4, after every fourth memory access,

we will incur a cache miss, so it really does not matter if we are replacing an existing cache line (which was anyways not going to be used) or using one of the increased cache lines, for the new block. The cache hit rate is bottle-necked by the blocksize.

The same is true for the (16, 16) case.

2.1.3 Varying Block Size

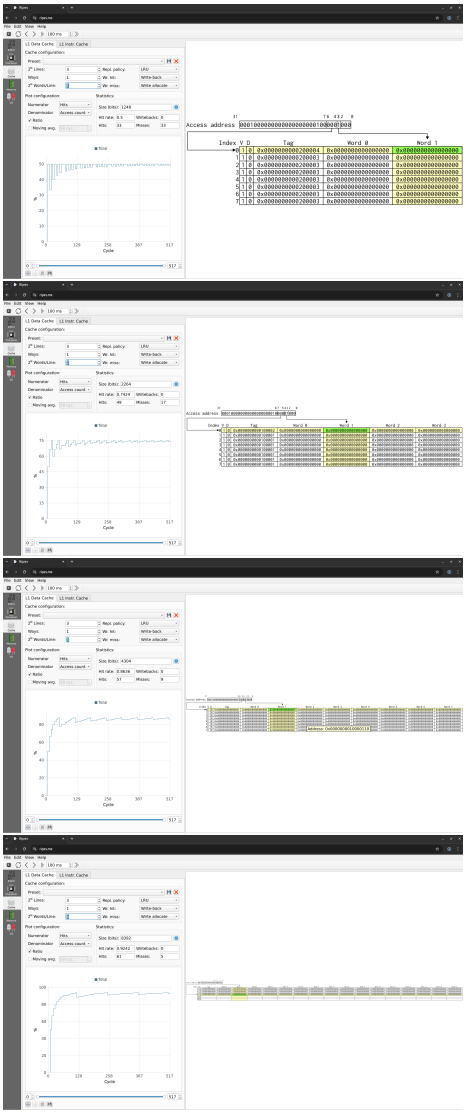


Figure 2: Increasing block size improves hit rate

Here we see a direct correlation between blocksize and the cache hit rate. This reaffirms our previous statement about the bottleneck being the blocksize for this kind of access pattern. The hit rate increases because only the first word of the block incurs a cache miss, all the rest are cache hits. So, increasing the blocksize will increase hit rate. This also holds true for the (16, 16) case.

2.1.4 Varying Associativity

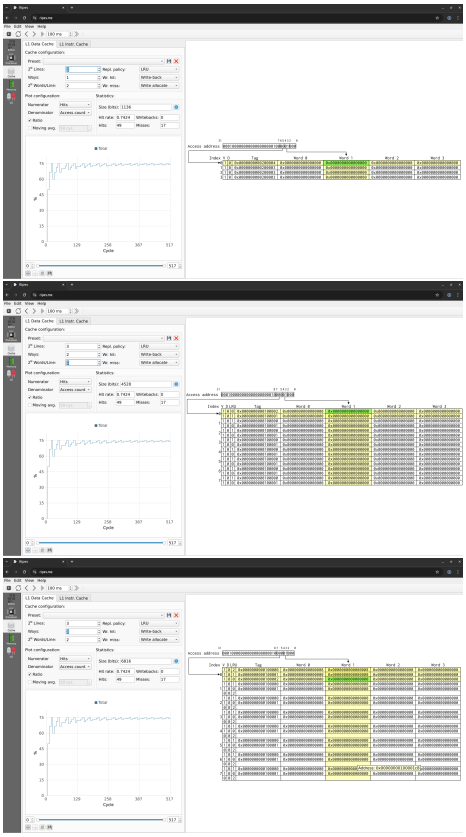


Figure 3: Increasing Associativity does not affect hit rate

Again, the cache hit rate is not affected by associativity, since it does not increase the blocksize which is the main factor in this access pattern. This also holds true for the (16, 16) case.

2.2 Program 2

2.2.1 Understanding the Memory Access Pattern

```
.data
L1: .dword 8, 8
A:

.text
la x3, L1
ld x4, 0(x3)
ld x5, 8(x3)
addi x3, x3, 16

    addi x10, x0, 0
    add x12, x3, x0
If1: beq x4, x10, end1
       addi x11, x0, 0
       slli x15, x10, 3
       add x12, x3, x15
If2:    beq x5, x11, end2
           ld x20, 0(x12)
           slli x15, x5, 3
           add x12, x12, x15
           addi x11, x11, 1
           beq x0, x0, If2
end2:   addi x10, x10, 1
       beq x0, x0, If1
end1: nop
```

Listing 2: RISC-V Assembly Code

The memory access pattern is such that each subsequent memory access differs by 8 bytes. So, any cache with block size less than 8 will see a penalty in more cache misses. Still, there is no temporal locality, so number of lines should not make a difference.

2.2.2 Varying Number of Lines

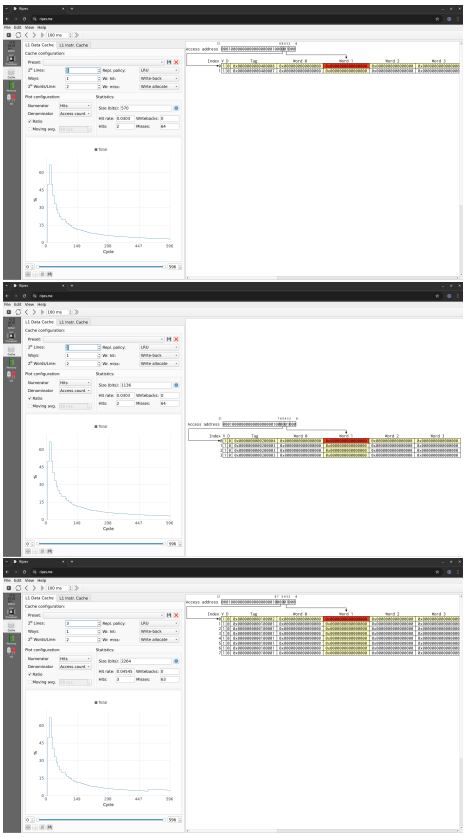


Figure 4: Increasing number of indices does not affect hit rate

Cache hit rate is not change by the linesize due the same reasons as program 1, namely lack of temporal locality. This also holds true for the (16, 16) case.

2.2.3 Varying Block Size

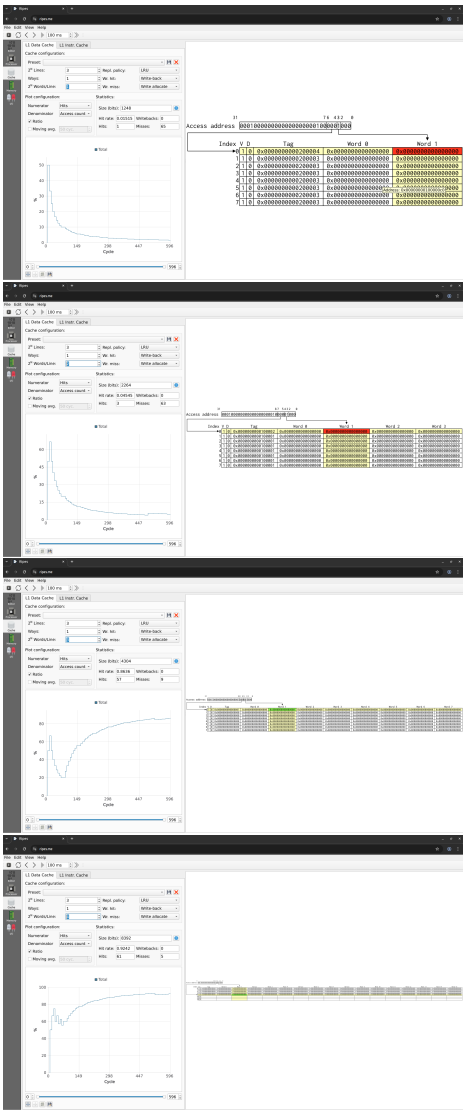


Figure 5: Increasing block size improves hit rate

Since the memory access jumps by 8 bytes, the lower blocksize see abysmal hit rate, but one block size reaches 8 and above, it sees the normal behavior as in program 1. This also holds true for the (16, 16) case.

2.2.4 Varying Associativity

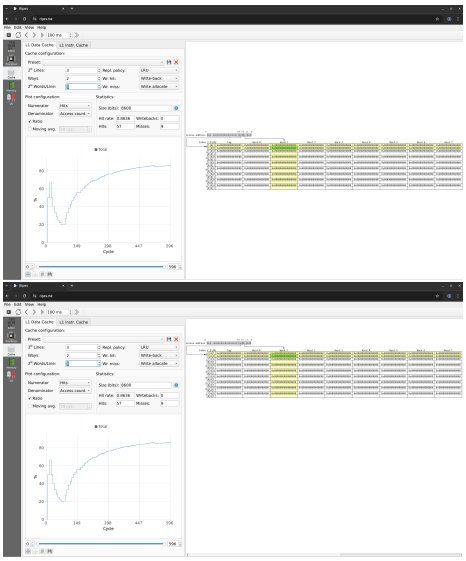


Figure 6: Increasing Associativity does not affect hit rate

The associativity in itself does not change the cache hit rate. For a fixed size cache, increasing associativity would increase the hit rate since blocksize would increase. This also holds true for the (16, 16) case.

3 Write Policies

3.1 Program 1

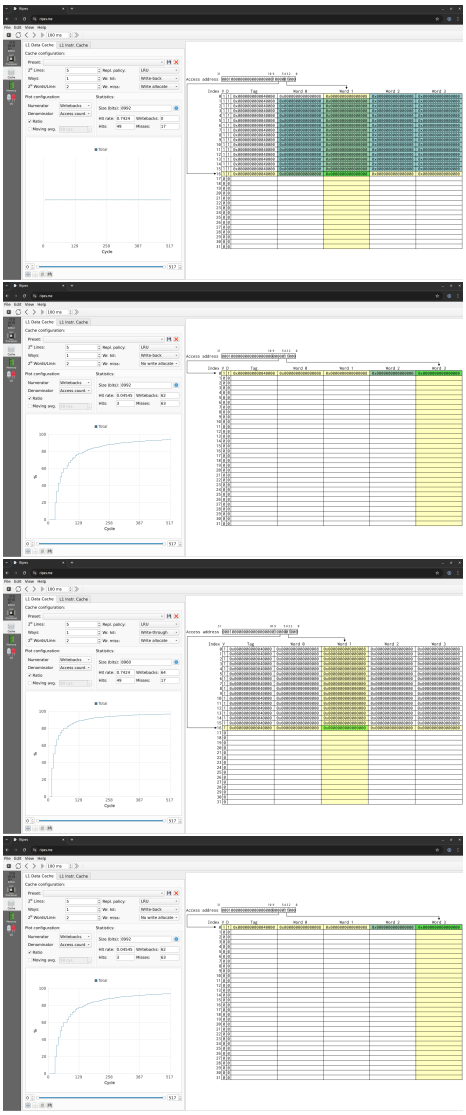


Figure 7: Program 1

3.2 Program 2

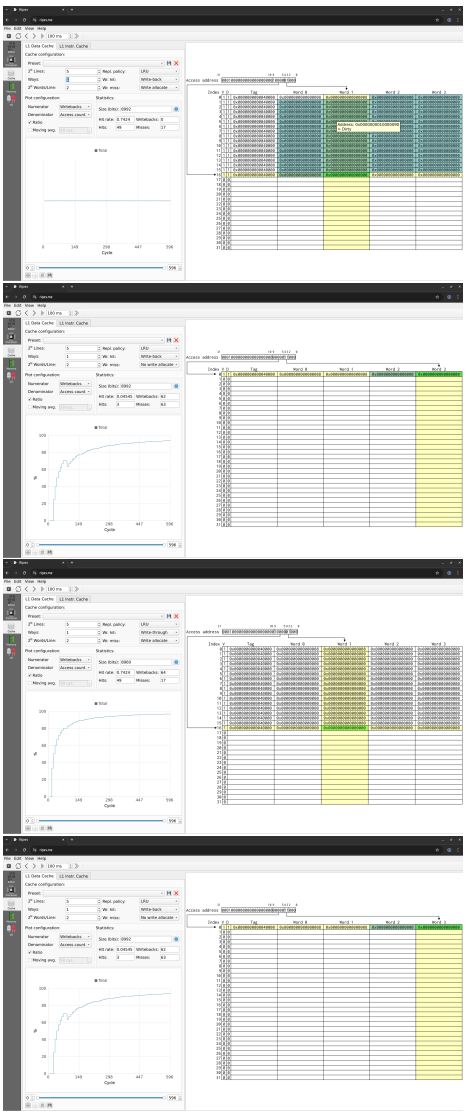
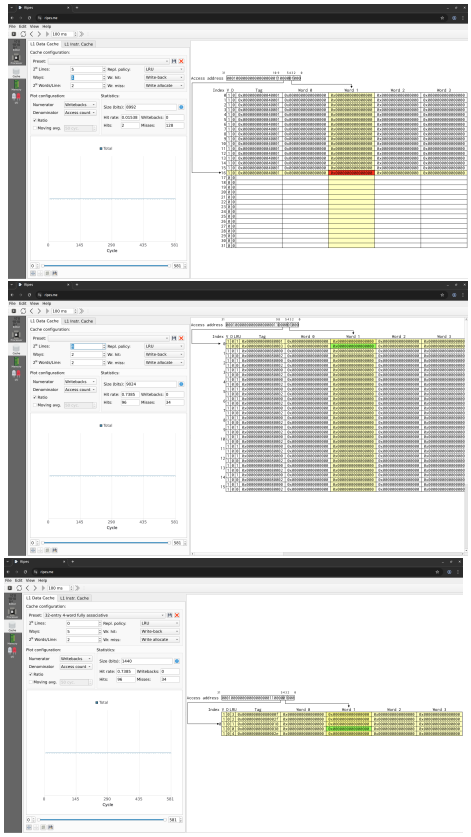


Figure 8: Program 2

Both the programs showcase similar behavior in all the configurations(both for (8, 8) and (16, 16) versions). Write allocate is a much better option since allocating the memory in the cache allows us to write to it in the cache itself without trying for more main memory transactions. No-allocate fails miserably since it performs best in the case of high temporal locality, which is completely

absent in both the programs.

4 Associativity



Direct mapped cache fails terribly since the cache access pattern, queries memory locations 2^{10} bytes away. For direct mapped cache to be of use, the block size would need to be of the order 1024 bytes which is impractical. For 2-way and fully associative caches, the hit rate is much higher since the program still alternatively exploits spatial locality, hence increased associativity increases the number of hits.