CO224 Computer Architecture

Lab 6 Building a Memory Hierarchy

Part 2 : Data Cache

Report

Group 12 :

E/19/455 YASHAN W.V.

E/19/124 GUNASEKARA M.H.

Instruction Set

Let’s consider the time consumptions for the same instructions set on,

1. System without cache
2. System with cache

```
1   loadi 4 0x05    //Load 5 into R4 Register
2   swi 4 0x01      //Write the value of R4 into memory address @ 0x01
3   loadi 5 0x0A    //Load 10 into R5 Register
4   loadi 6 0x20    //Load 32 into R6 Register
5   swi 5 0x20      //Write the value of R5 into memory address @ 0x20
6   loadi 6 0x20    //Load 32 into R6 Register
7   lwi 8 0x01      //Read mem address @ 0x01, Store it into R8
8   loadi 7 0x14    //Load 20 into R7 Register
9   lwi 9 0x20      //Read mem address @ 0x20, Store it into R9
10  loadi 5 0x05    //Load 5 into R5 Register
11  swi 7 0x20      //Write the value of R7 into memory address @ 0x20
12  lwi 4 0x20      //Read mem address @ 0x20, Store it into R4
```

When analyzing the instruction set,

- 2nd Instruction : Write miss But not Dirty
- 5th Instruction : Write miss and Dirty
- 7th Instruction : Read miss and Dirty
- 9th Instruction : Read miss But not Dirty
- 11th Instruction : Write hit
- 12th Instruction : Read hit

With Data Cache Memory

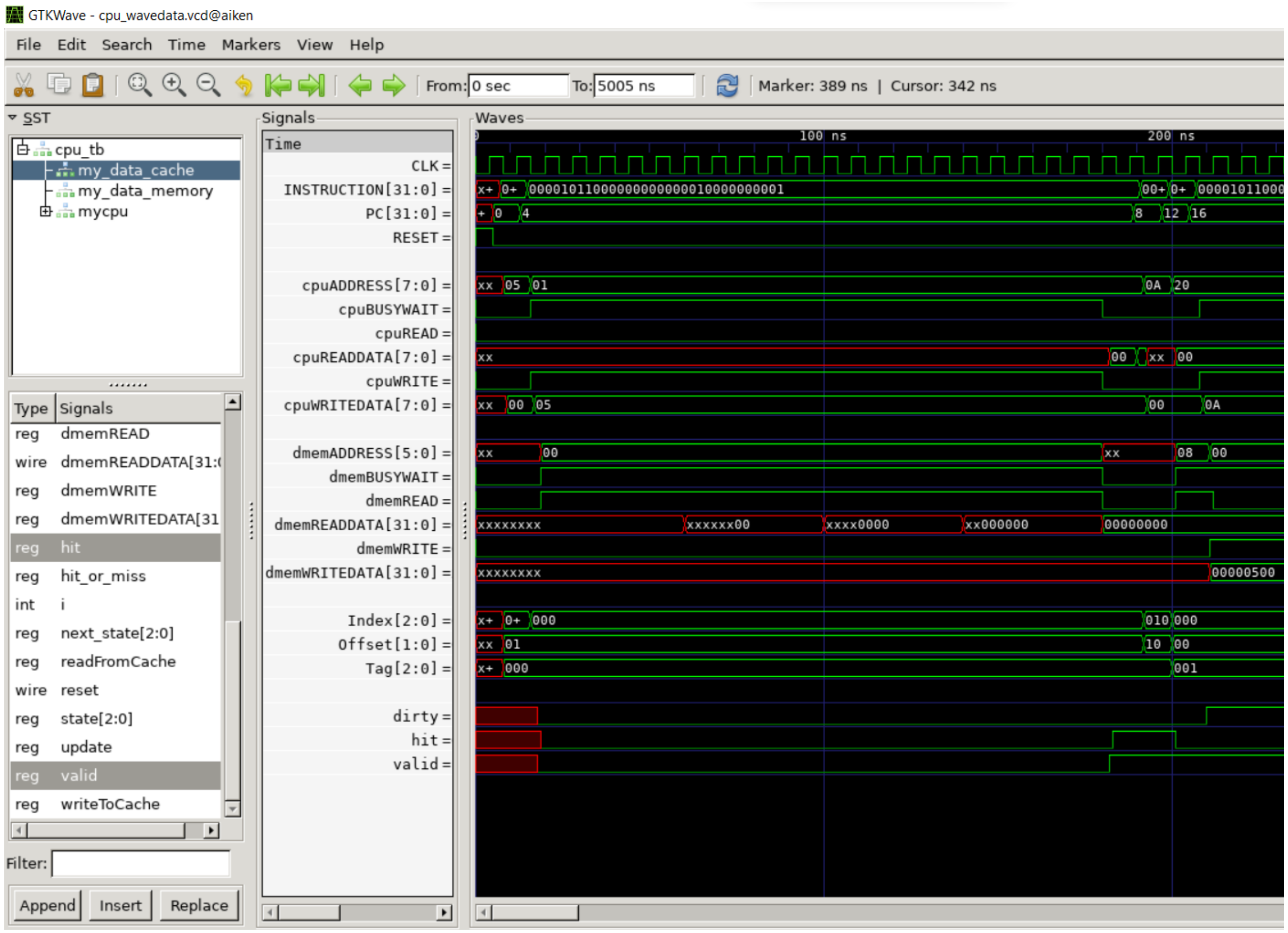


Figure 01: Timing Diagram of Write miss but not Dirty

Here, we can observe that when there’s a write miss but not dirty, a 21 cycles miss penalty is added.

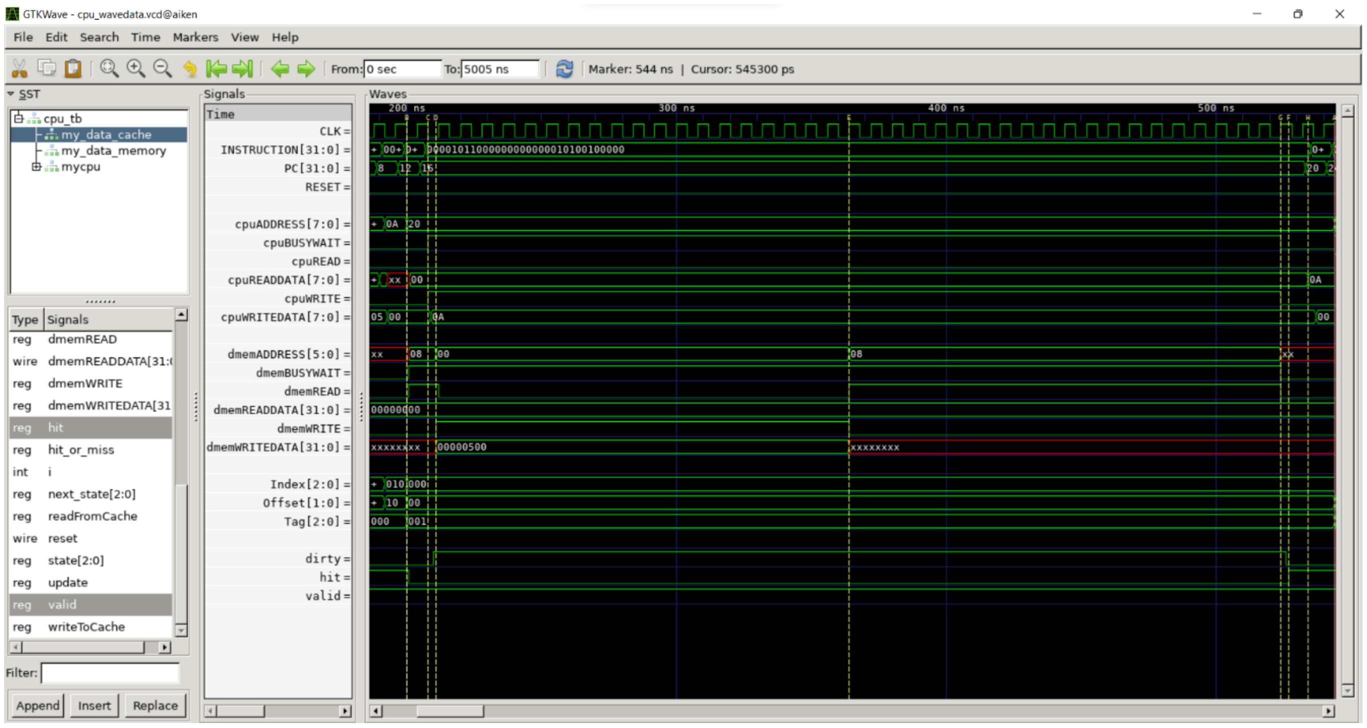


Figure 02: Timing Diagram of Write miss and Dirty

Here, we can observe that when there's a write miss and dirty, a 42 cycles miss penalty is added.

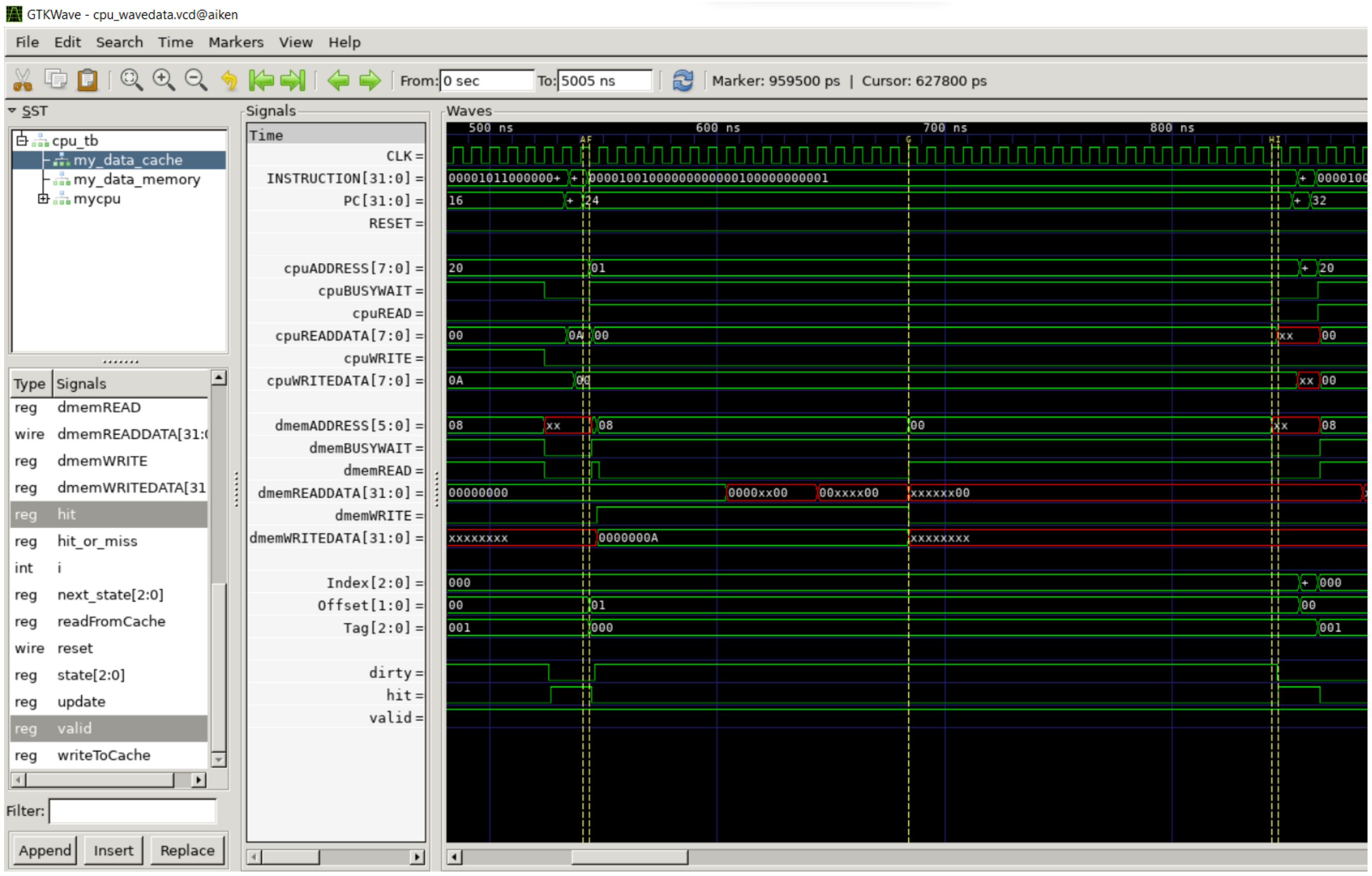
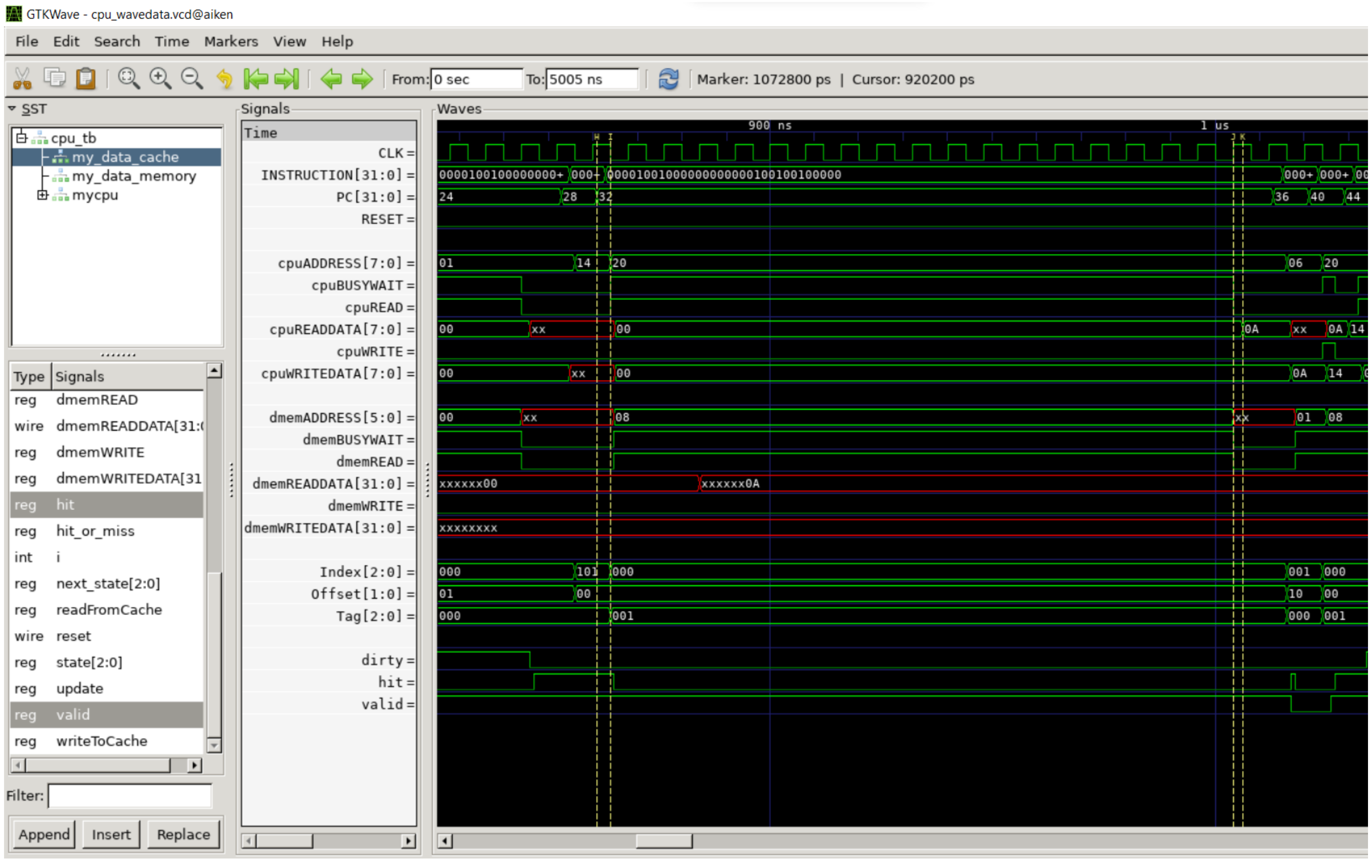


Figure 03: Timing Diagram of Read miss and Dirty

Here, we can observe that when there's a read miss and dirty, a 42 cycles miss penalty is added.



Type	Signals
reg	dmemREAD
wire	dmemREADDATA[31:(
reg	dmemWRITE
reg	dmemWRITEDATA[31
reg	hit
reg	hit_or_miss
int	i
reg	next_state[2:0]
reg	readFromCache
wire	reset
reg	state[2:0]
reg	update
reg	valid
reg	writeToCache

Figure 04: Timing Diagram of Read miss but not Dirty

Here, we can observe that when there's a read miss but not dirty, a 21 cycles miss penalty is added.

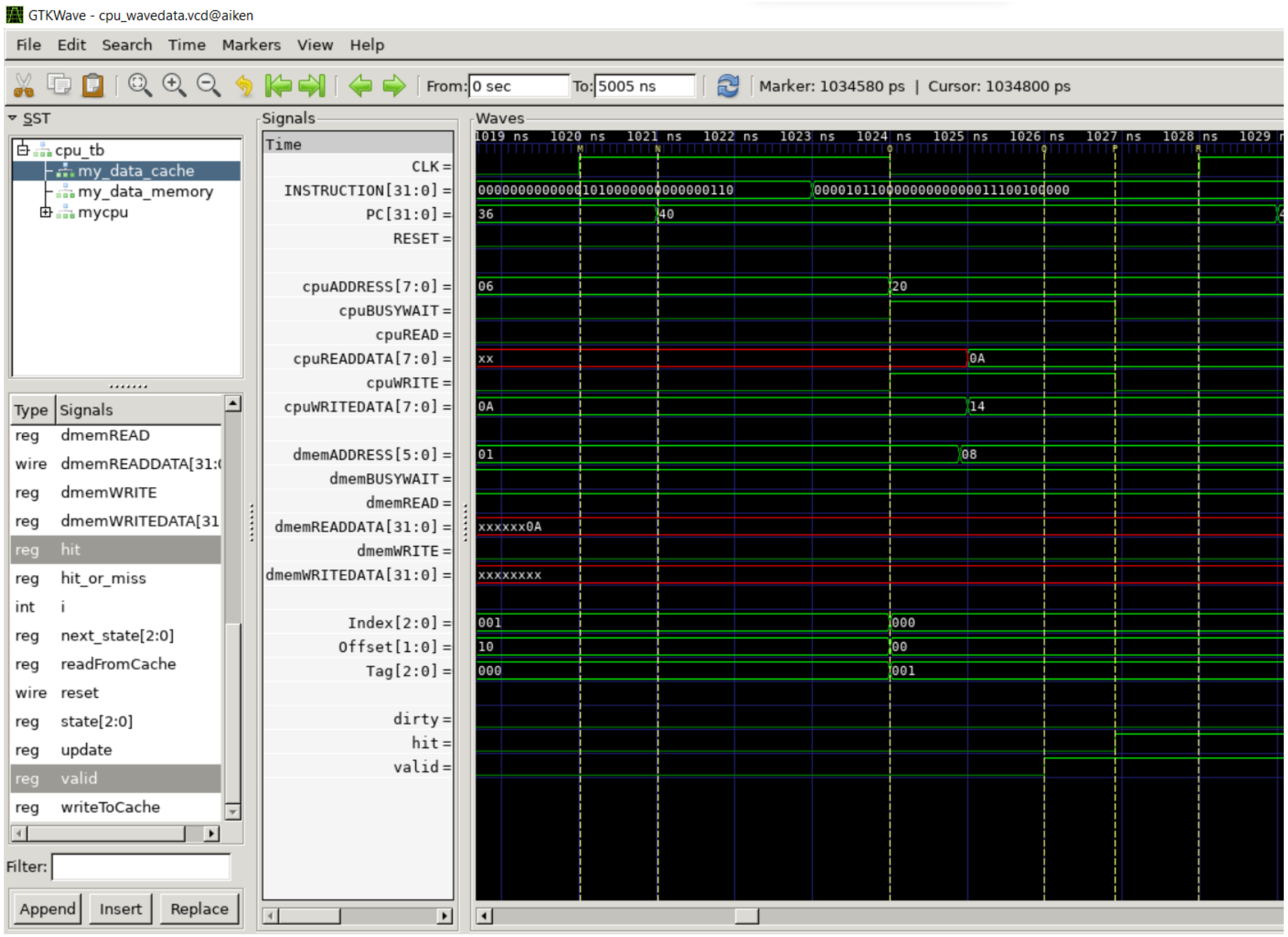


Figure 05: Timing Diagram of Write Hit

Here, we can observe that when there's a write hit, it takes only 1 clock cycle to execute next instruction.

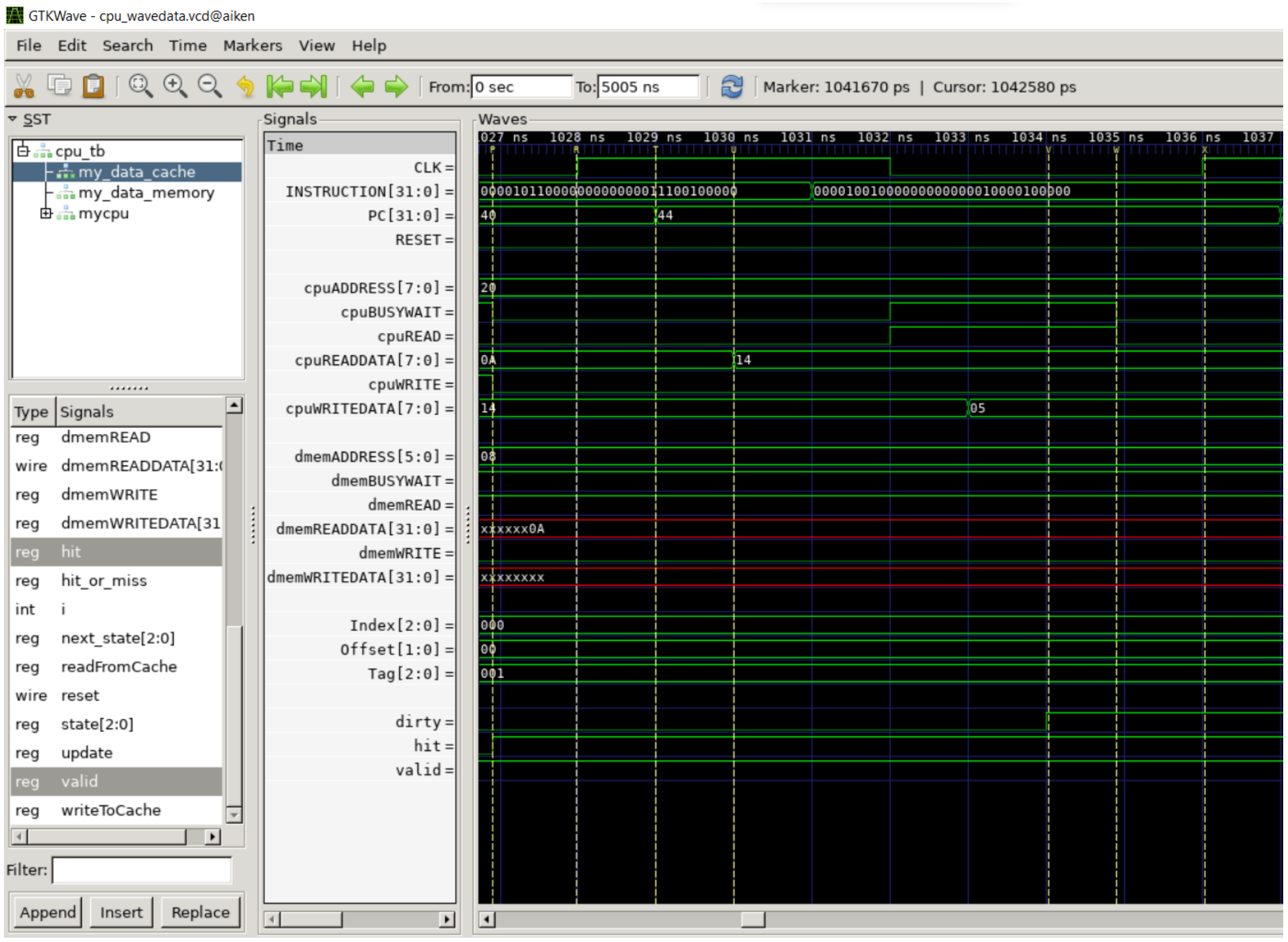


Figure 06: Timing Diagram of Read Hit

Here, we can observe that when there's a read hit, it takes only 1 clock cycle to execute the instruction.

Summary

Instruction Execution (With Cache Memory)	No of Clock Cycles that takes
2nd Instruction : Write miss But not Dirty	21
5th Instruction : Write miss and Dirty	42
7th Instruction : Read miss and Dirty	42
9th Instruction : Read miss But not Dirty	21
11th Instruction : Write hit	1
12th Instruction : Read hit	1

Without Data Cache Memory

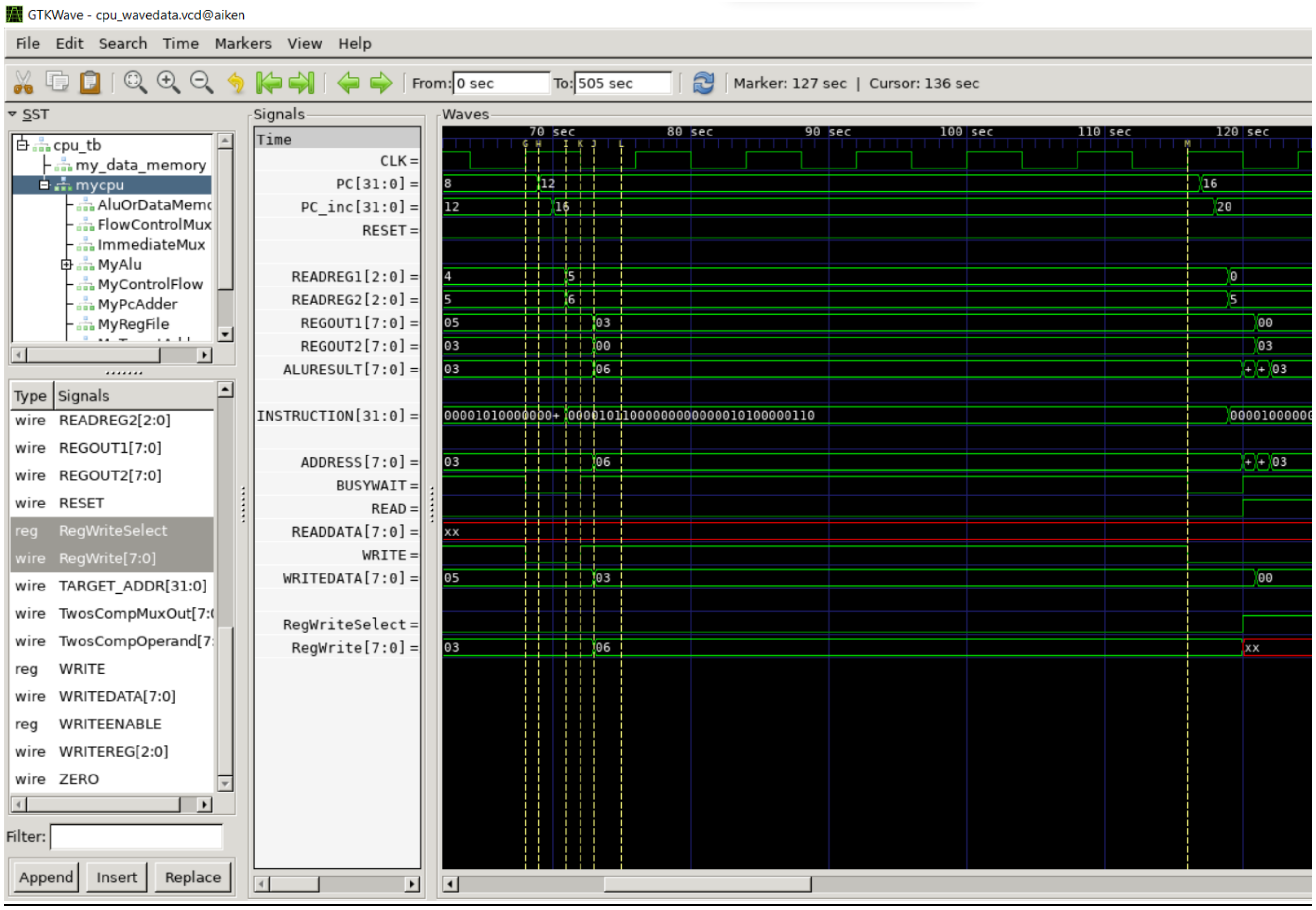


Figure 07: Timing Diagram of Write the value of R7 into memory

Here, we can observe that when there's a Write to memory(LOAD), it takes 6 clock cycle to execute the next instruction. With 5 clock cycles to memory read.

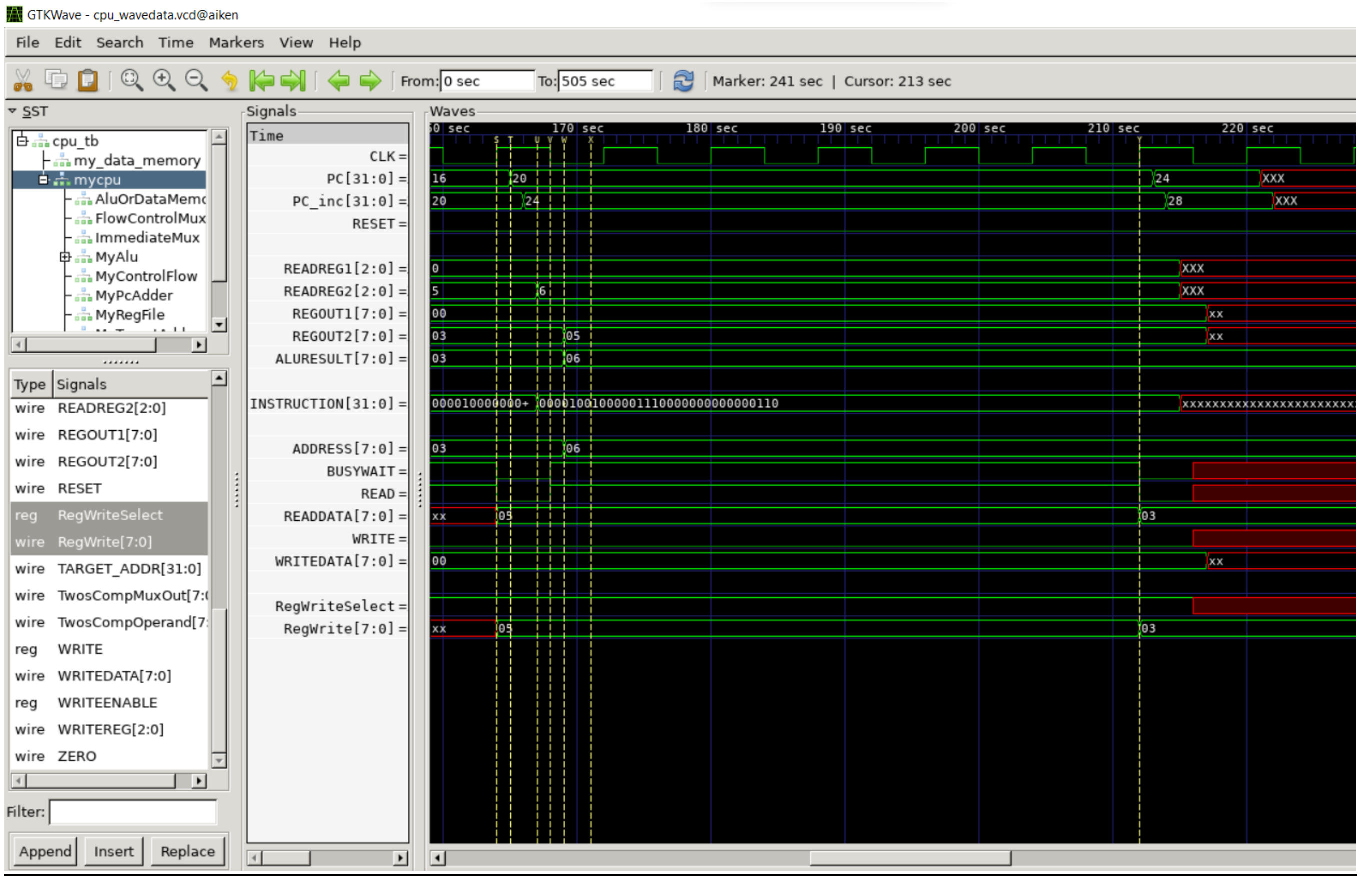


Figure 08: Timing Diagram of Read and Store into R4

Here, we can observe that when there's a Read and Store(STORE), it takes 6 clock cycle to execute the next instruction. With 5 clock cycles to memory read.

Comparison between System with cache and System without cache

Instruction	Clock Cycles (With cache)	Clock Cycles (Without cache)
11th Instruction : Write hit	1	6
12th Instruction : Read hit	1	6

Conclusion

Both systems execute the same instruction set.

In a system without cache, every memory access requires a direct access to the main memory every time it is required. This can result in a higher number of memory accesses compared to a system with a cache. Main memory accesses typically have higher latency compared to cache accesses.

By storing frequently accessed data in cache memory, it reduces the time needed to fetch data from the main memory.

Caches are designed to exploit the principles of spatial and temporal locality. Spatial locality refers to the tendency of a program to access data near the currently accessed data, while temporal locality refers to the tendency to access the same data again in the near future. Caches store recently accessed data and nearby data to satisfy subsequent requests more quickly.

Without a cache, the system loses the ability to take full advantage of these locality properties, leading to increased memory access overhead.

To wrap it up, in a system with a cache, the cache can store frequently accessed data, reducing memory latency and improving performance. The exact performance difference between the system without cache and the system with a cache would depend on factors such as cache size, cache hit/miss rates, and the specific memory access patterns of the program.

A system without cache typically experiences increased memory latency, more frequent main memory accesses, limited exploitation of locality, and potentially higher power consumption compared to a system with a cache. These factors can contribute to slower overall performance, but the exact impact will depend on the workload and memory access characteristics of the specific system.