CO224 – Computer Architecture
Lab 06 – Part 2 (Report)

Group 44 :
E/20/024 - D.B.S. Ariyarathna
E/20/420 - Wanasinghe J.K.

In part 2 of lab 6, the data memory hierarchy for the CPU was further modified by adding a data cache module between the CPU and the data memory to minimize delays of data access. The performance between the previous cache-less setup and the current setup was tested by executing sample programs on both and observing the differences in the delays.

The following diagram depicts the finite state machine for the cache controller.

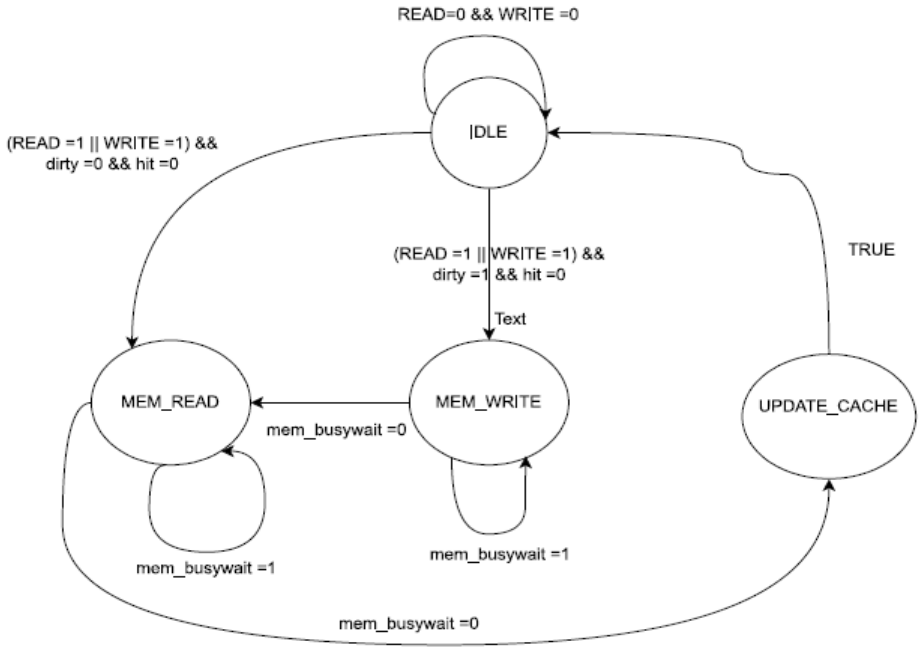

FIGURE 1 : FINITE STATE MACHINE FOR CACHE CONTROLLER

- This finite state machine consists of 4 states.

  IDLE: The default state when no data requests are being processed.

  MEM_READ: Activated when data needs to be fetched from the main memory.

  MEM_WRITE: Triggered when data needs to be written to the main memory.

  UPDATE_CACHE: Occurs when the cache needs to be updated with new data.

- Transitions: The state changes are based on various conditions:

From **IDLE** to **READ**: Initiated by a read request when the cache is clean.

From **IDLE** to **MEM_WRITE**: Occurs when there's a write request and the cache is dirty, indicating that the cached data has been modified and needs to be written back to the main memory.

- Control Signals: These include READ, WRITE, and the status of the dirty bit and memory bus (mem_busywait).

This report contains some comparative details which were observed during the lab 6 part 1 and part 2. The instruction sets given for part 2 and 1 are used in the implementations respectively.

Set of Instructions given for part 2

Considering the time taken for the two systems for the same set of instructions.

```
loadi 0 0x09      //load 9 into register 0
loadi 1 0x01    //load 1 into register 1
swd 0 1           //store the value in register 0 into the memory address in register 1
swi 1 0x00        //store the value in register 1 into memory address 0x00
lwd 2 1           //load the value at memory address 0x01 into register 2
lwd 3 1           //load the value at memory address 0x01 into register 3
sub 4 0 1         //subtract the value in register 0 from the value in register 1 and store
the result in register 4
swi 4 0x02        //store the value in register 4 into memory address 0x02
lwi 5 0x02        //load the value at memory address 0x02 into register 5
swi 4 0x20  //store the value in register 4 into memory address 0x20
lwi 6 0x20  //load the value at memory address 0x20 into register 6
```

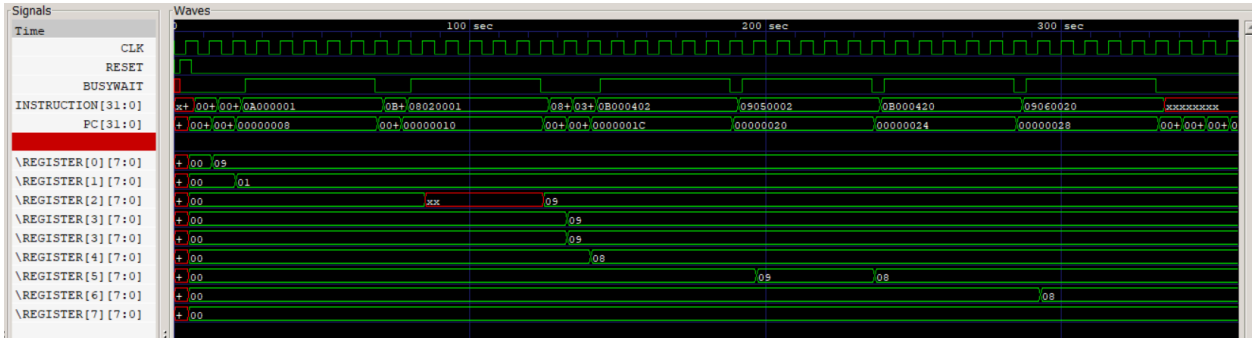

FIGURE 2 : SYSTEM WITHOUT CACHE

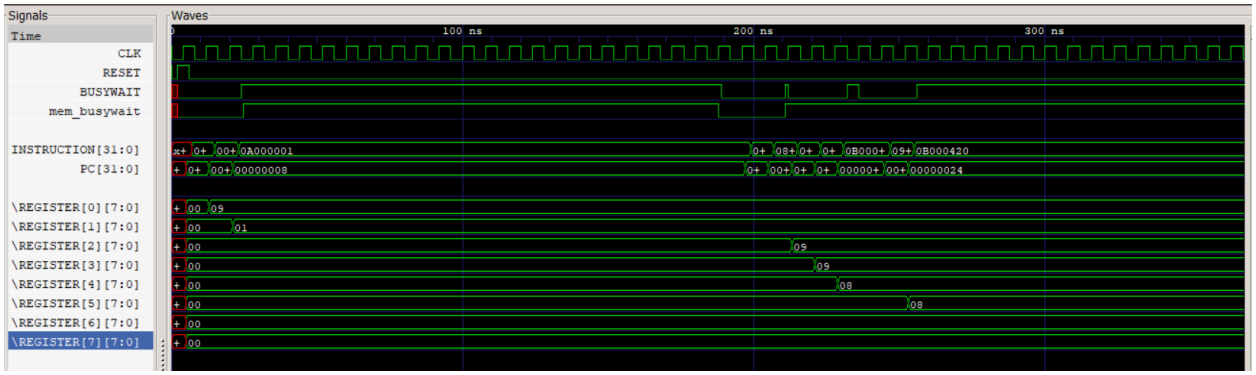

FIGURE 3 : SYSTEM WITH CACHE

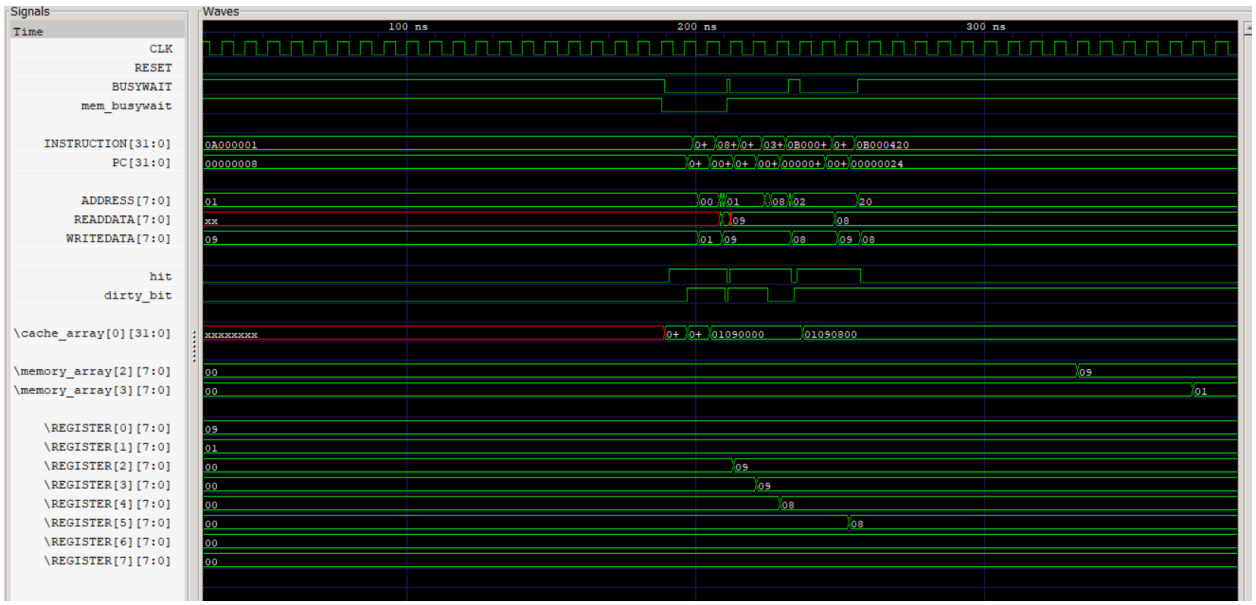

FIGURE 4 : SYSTEM WITH CACHE IN WRITE HIT / READ HIT

When there is a miss, it is being checked in the memory (at the absence of cache). If absent at both places, they are being filled. If the value is present, it is taken and given to the CPU. So there is a 20 clock cycle time spent here.

Set of Instructions given for part 1

Considering the time taken for the two systems for the same set of instructions.

```
loadi 2 0x0A //load 0x0A into register 2
swi 2 0x01 //store 0x0A into memory location 0x01
lwi 1 0x01 //load the value at memory location 0x01 into register 1
loadi 3 0x02 //load 0x02 into register 3
swd 2 3 //store the value in register 2 into memory location 0x02
lwd 4 3 //load the value at memory location 0x02 into register 4
```

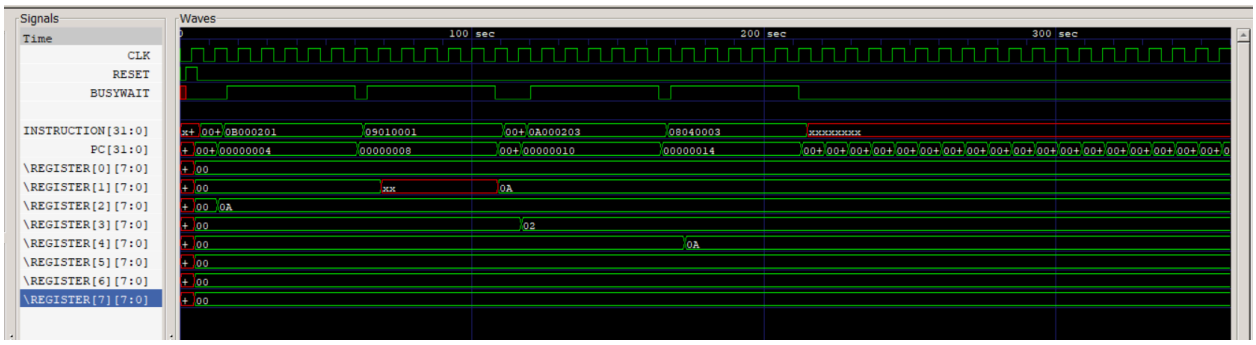

FIGURE 5 : SYSTEM WITHOUT THE CACHE

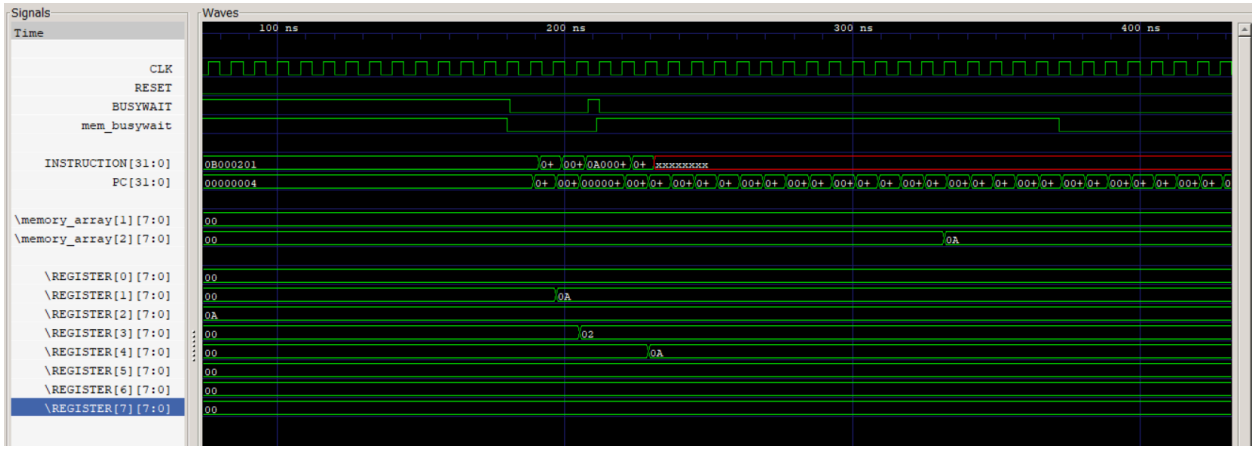

FIGURE 6 : SYSTEM WITH THE CACHE

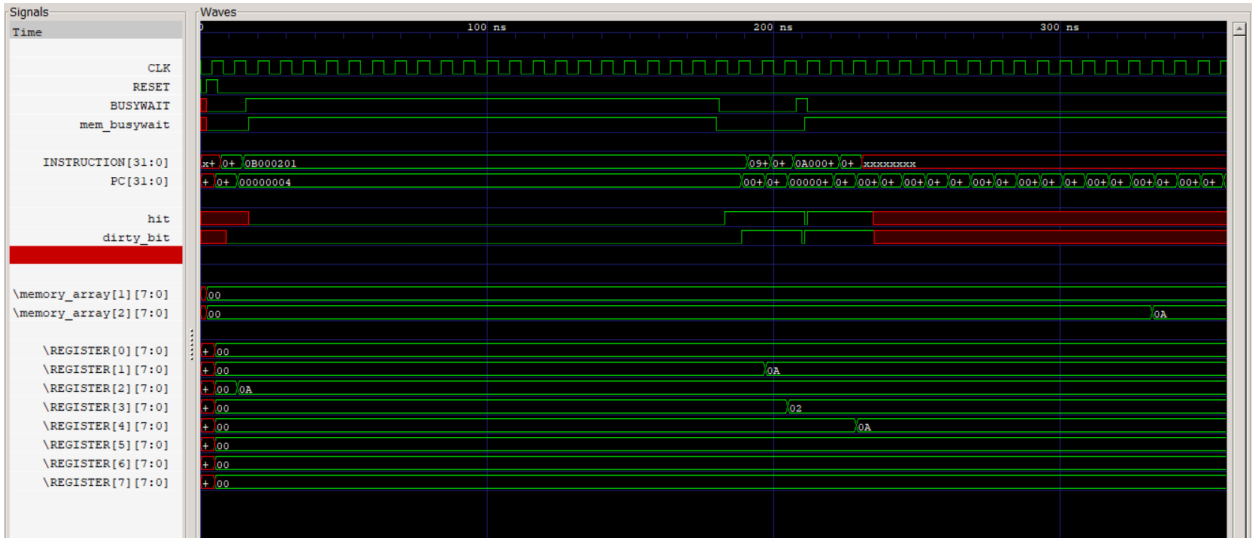

FIGURE 7 : SYSTEM WITH CACHE IN WRITE HIT / READ HIT

‘Hitting’ is checking the cache for the required data. In part 1, loading is done to cache, so to check whether there is a hit, no need to go for the memory. So, it directly shows as hits. Here cache memory acts as a data memory.

Conclusion

When there are more hits than misses, the cache implementation is more efficient than the cacheless.

The reasons for this are;

- Temporal locality (Availability of recently accessed data in cache)
- Spatial locality (Data located closer to the data which was accessed recently, would also be available in the cache since the blocks of data were fetched from the memory)

Therefore, if the hit rate is high, implementation of the cache will greatly reduce the time taken to execute a program because memory reading and writing can be completed within 1 clock cycle, which would otherwise always take 20 clock cycles to complete without a cache in between the data memory and the CPU.