CO224 - Lab 06 – Part 02

Title: Building a Memory Hierarchy

Group 02 (E/18/397, E/18/402)

In this Part we developed a cache memory for our 8-bit processor. To build that we used a finite state machine for our cache controller.

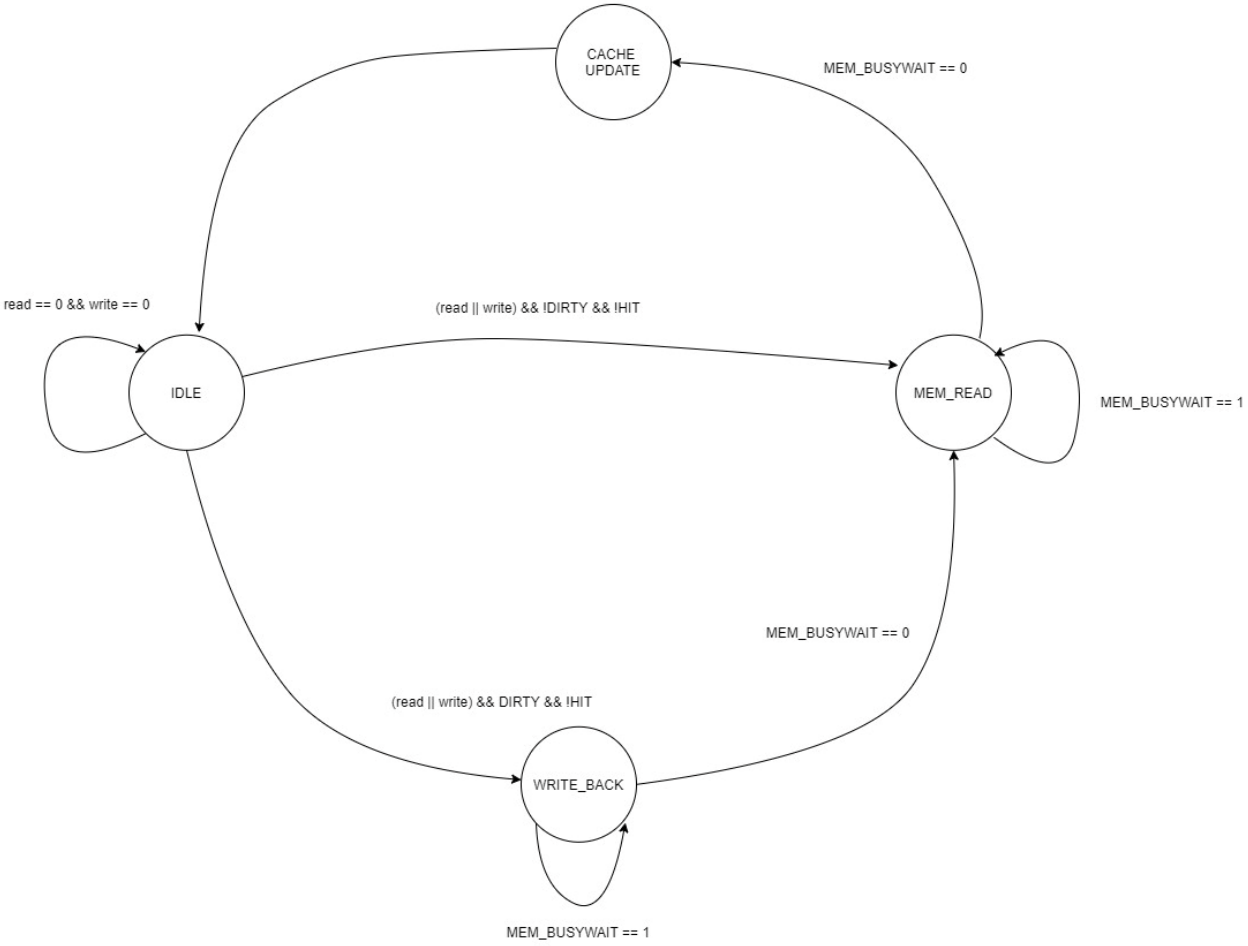

Figure 01: Finite state machine for cache controller

We used a sample program to compare our CPU with cache and without cache.

sample_program.s

```
1    loadi 0 0x09
2    loadi 1 0x01
3    swd 0 1
4    swi 1 0x00
5    lwd 2 1
6    lwd 3 1
7    sub 4 0 1
8    swi 4 0x07
9    lwi 5 0x07
10   lwi 6 0x20
11   swi 4 0x20
```

Figure 02: used sample program

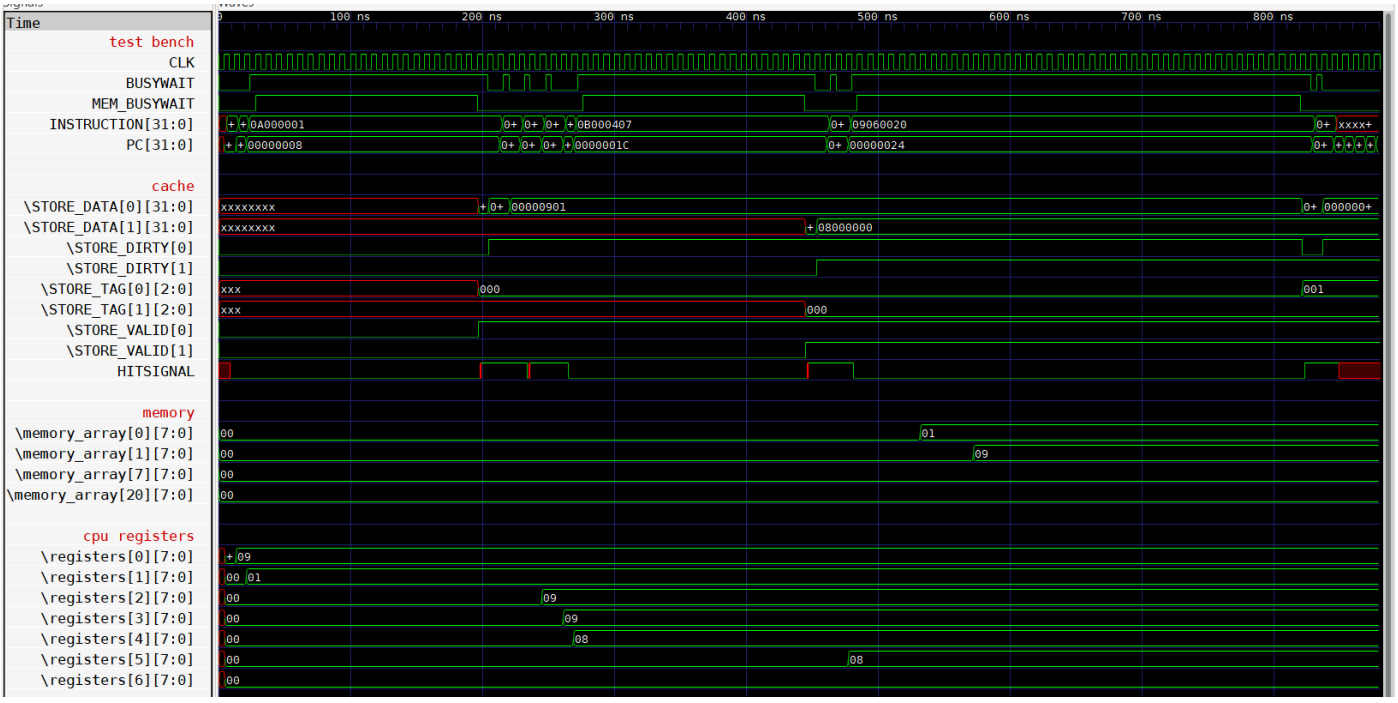

Figure 03: observed timing diagram for sample code where CPU with a cache memory.

First 2 instructions are loadi instructions. Since loadi instructions doesn't interfere with memory modules there are no busy waits. When it comes to first swd instruction compulsory miss happens. This happened because initially we don't have any values in the cache. Since Next four Instructions are read hits and write hits. Therefore, no stalling for CPU and writing data to cache block. Those four instructions have same index (000). For 8th instruction we have deal with memory address 0x07 and that memory address contains index of 001. Therefore, cache will retrieve data from main memory and store in 2nd slot of the cache. So, CPU will stall. Then again read hit in the 9th instruction and in the 10th instruction we must

perform write back operation because we must replace values in cache block. Then again in the 11th instruction contains write hit. Therefore, no stall for CPU.

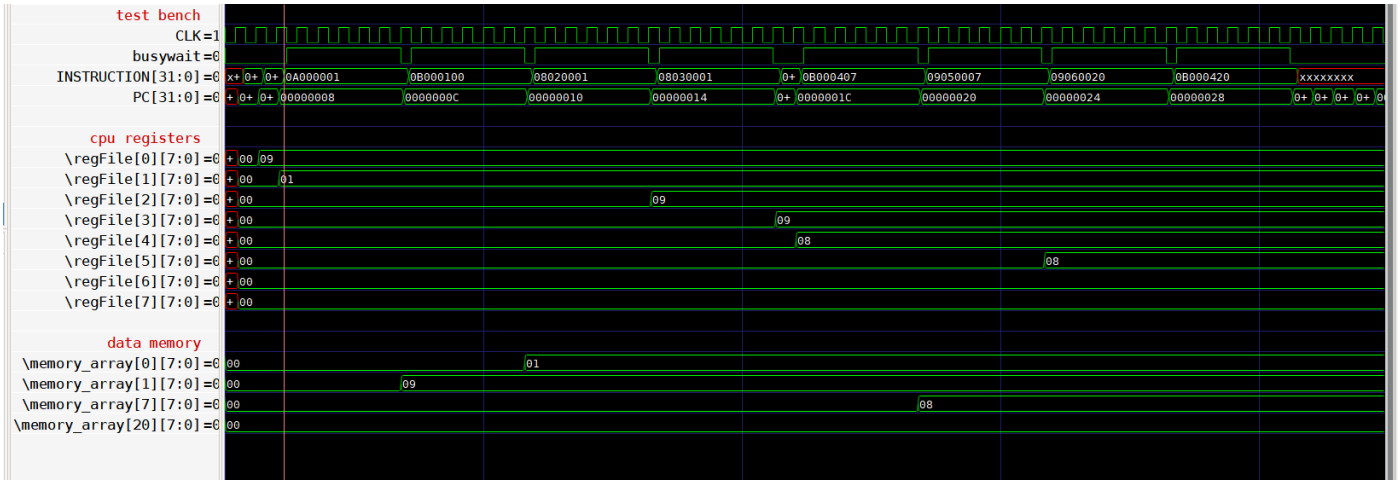

Figure 04: observed timing diagram for sample code where CPU without a cache memory.

In the non-cache statement, all the instructions that deal with data memory consume more than clock cycle. When using cache, some instructions that deal with data memory consume only one clock cycle time. The reason is that those corresponding values are stored in the cache.