CO 224 – LAB 6

BUILDING A MEMORY HIERARCHY

COMPARISON REPORT

DISSANAYAKE D.M.I.G. : E/19/090

WIJERATHNA I.M.K.D.I. : E/19/446

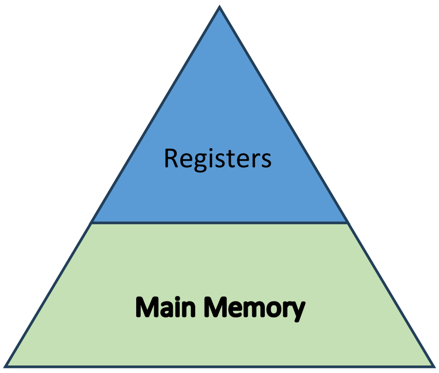
GROUP 13

SEMESTER 4

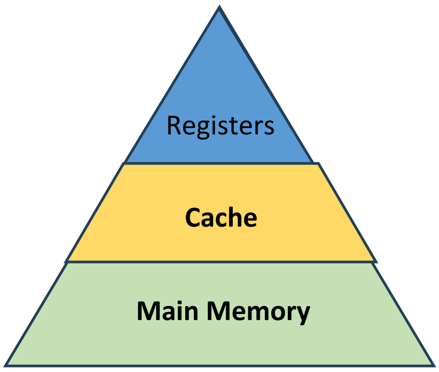
28/06/2023

INTRODUCTION

In this lab we are intended to build a memory sub-system for the CPU that have been built on the previous lab. In this report only the data memory (not considering instruction) sub section is considered. First part of the lab expects us to design the memory sub-system only using data memory. Then, in the later part it suggests us to introduce a data cache in between registers and data memory.



Lab 6 Part 1 (without cache)



Lab 6 Part 2 (with cache)

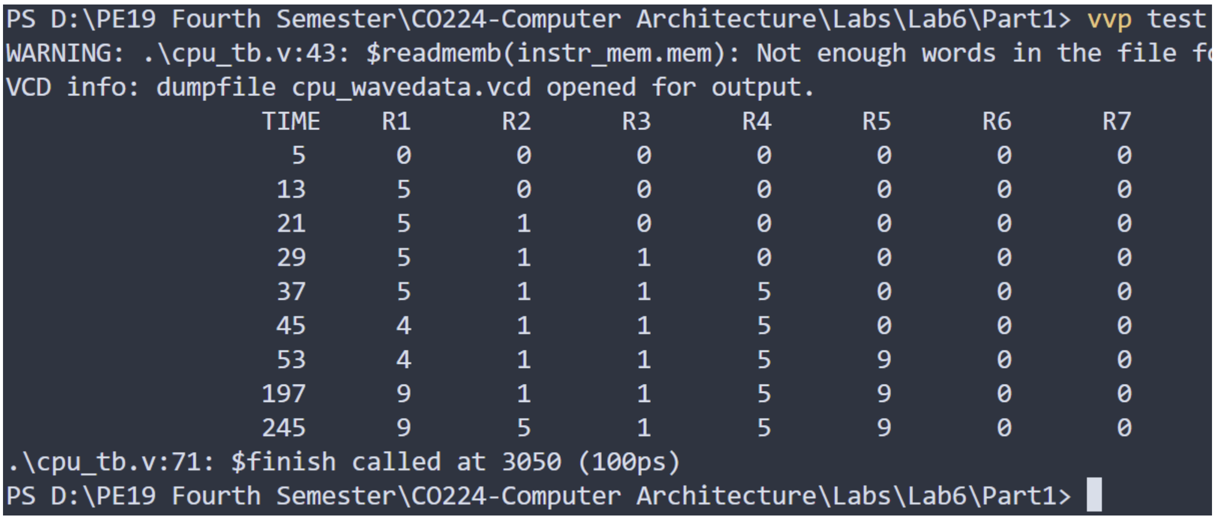
In a cache-less system, the CPU directly accesses the main memory, leading to longer memory access latencies and potential performance slowdowns. On the other hand, a cache-full system incorporates a cache between the CPU and main memory, enabling faster access to frequently used data and instructions. By exploiting temporal and spatial locality, cache-full systems can reduce memory stalls and improve overall system performance. However, cache-full systems are more complex in design and require additional hardware for cache management. The choice between cache-less and cache-full designs depends on factors such as performance requirements, cost considerations, and design complexity.

SAMPLE CODE USED : sample_program.s

```
loadi 1 0x05
loadi 2 0x01
loadi 3 0x01
loadi 4 0x05
sub 1 1 2
add 5 1 4
swd 5 4     // store r5  val in memory[r4=0x05]
swi 4 0x03  // store r4 val in memory[0x03]
lwd 1 4     // load to r1 val in (memory[r4=0x05])=9
lwi 2 0x03  // load to r2 val in (memory[0x03])=5
```

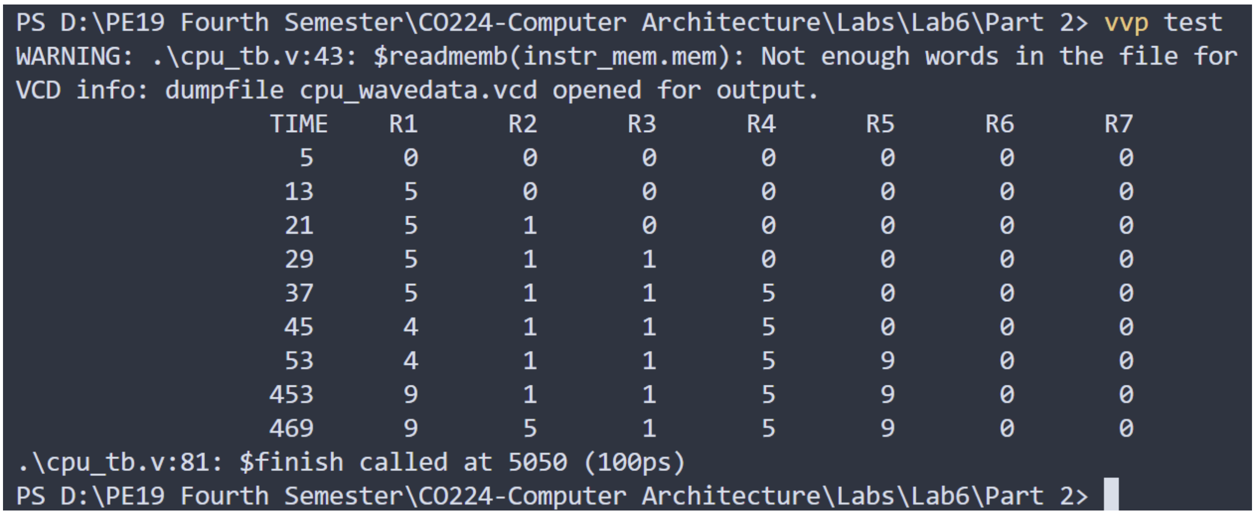
Selected register values monitoring w.r.t. time

Part 1 (without cache)



```
PS D:\PE19 Fourth Semester\CO224-Computer Architecture\Labs\Lab6\Part1> vvp test
WARNING: .\cpu_tb.v:43: $readmemb(instr_mem.mem): Not enough words in the file f
VCD info: dumpfile cpu_wavedata.vcd opened for output.
                 TIME     R1       R2       R3       R4       R5       R6       R7
                    5      0        0        0        0        0        0        0
                   13      5        0        0        0        0        0        0
                   21      5        1        0        0        0        0        0
                   29      5        1        1        0        0        0        0
                   37      5        1        1        5        0        0        0
                   45      4        1        1        5        0        0        0
                   53      4        1        1        5        9        0        0
                  197      9        1        1        5        9        0        0
                  245      9        5        1        5        9        0        0
.\cpu_tb.v:71: $finish called at 3050 (100ps)
PS D:\PE19 Fourth Semester\CO224-Computer Architecture\Labs\Lab6\Part1>
```

Part 2 (with cache)

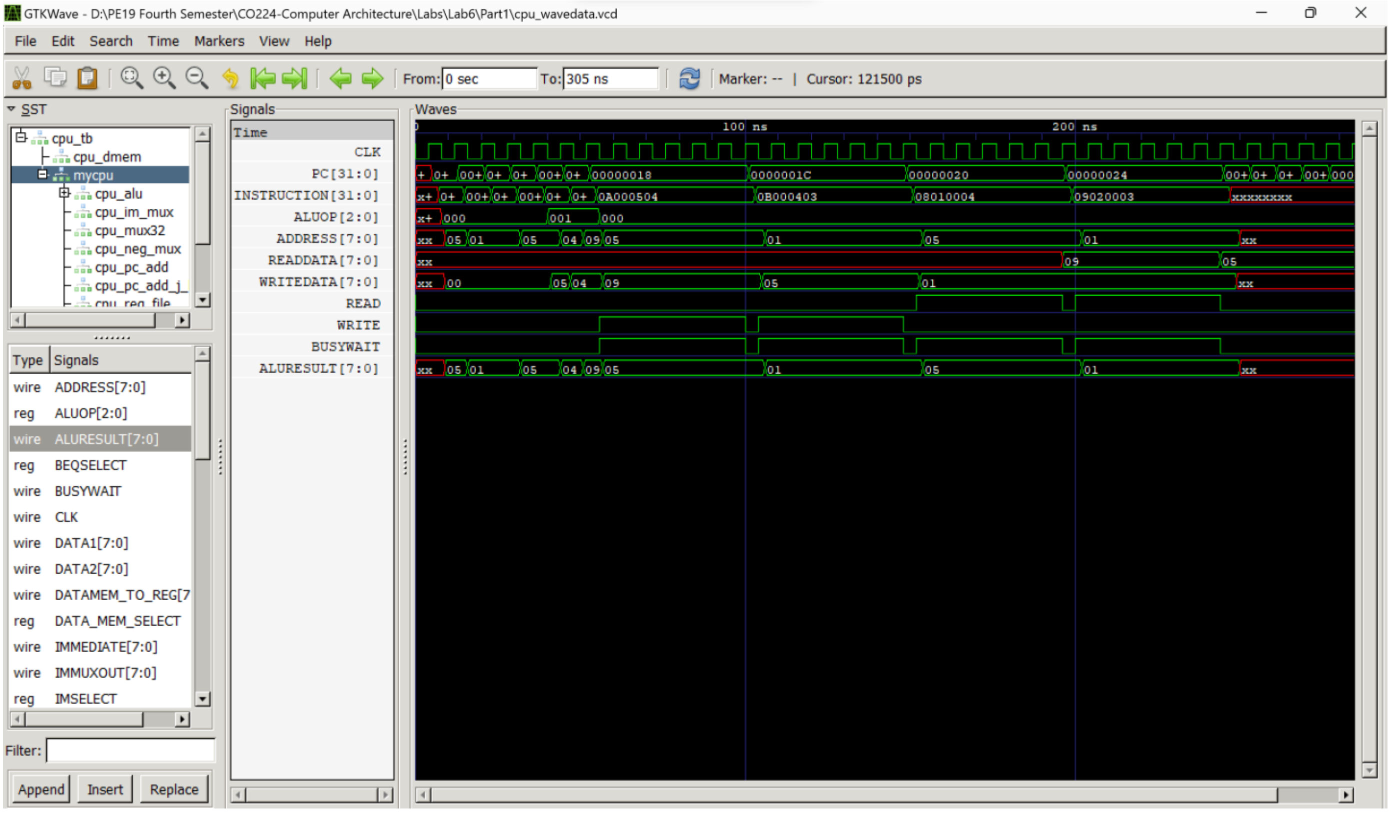


```
PS D:\PE19 Fourth Semester\CO224-Computer Architecture\Labs\Lab6\Part 2> vvp test
WARNING: .\cpu_tb.v:43: $readmemb(instr_mem.mem): Not enough words in the file for
VCD info: dumpfile cpu_wavedata.vcd opened for output.
                 TIME     R1       R2       R3       R4       R5       R6       R7
                    5      0        0        0        0        0        0        0
                   13      5        0        0        0        0        0        0
                   21      5        1        0        0        0        0        0
                   29      5        1        1        0        0        0        0
                   37      5        1        1        5        0        0        0
                   45      4        1        1        5        0        0        0
                   53      4        1        1        5        9        0        0
                  453      9        1        1        5        9        0        0
                  469      9        5        1        5        9        0        0
.\cpu_tb.v:81: $finish called at 5050 (100ps)
PS D:\PE19 Fourth Semester\CO224-Computer Architecture\Labs\Lab6\Part 2>
```

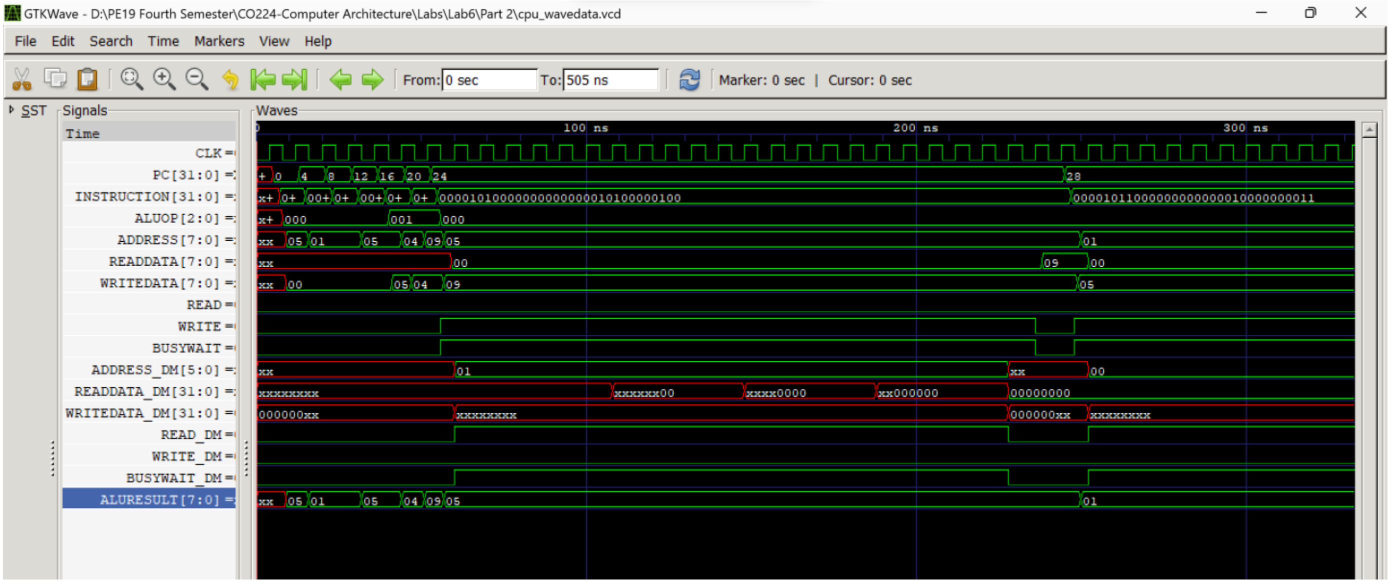
The same time scale is used in both parts. (`` `timescale  1ns/100ps ``)

GTKWave outputs:

Part 1 (without cache)



Part 2 (with cache)

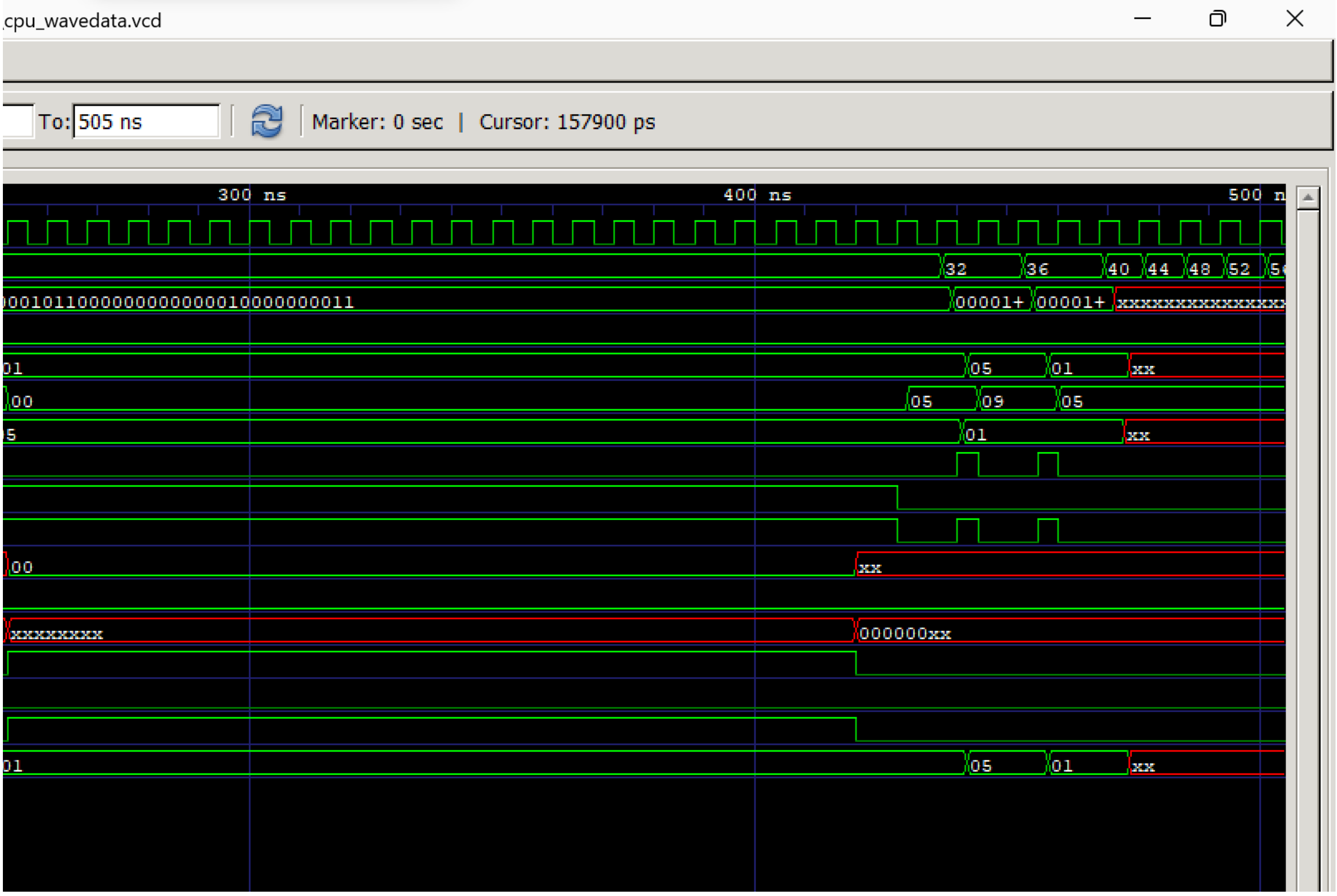


GTKWave - D:\PE19 Fourth Semester\CO224-Computer Architecture\Labs\Lab6\Part 2\cpu_wavedata.vcd

From: 0 sec To: 505 ns

Marker: 0 sec | Cursor: 0 sec

part 2 continued…



cpu_wavedata.vcd

To: 505 ns

Marker: 0 sec | Cursor: 157900 ps

Cache-less Memory Subsystem:

The cache-less memory subsystem refers to a design where the CPU directly accesses main memory without any intermediate cache. In this case, every memory access requires a long latency to fetch or store data from the main memory.

Cache-full Memory Subsystem:

The cache-full memory subsystem incorporates a cache between the CPU and main memory. The cache is designed to store frequently accessed data and instructions, reducing the memory access latency for subsequent requests.

Performance Analysis:

- Cache-less: 3050 (100ps)
- Cache-full: 5050 (100ps)

The cache-full memory subsystem is expected to improve the performance of the cache-full memory subsystem because of the introduced caching mechanism. The cache stores recently accessed instructions and data, allowing the CPU to retrieve them quickly without waiting for the main memory access. As a result, the cache-full system reduces the number of memory stalls and improves overall execution time. However, when a read-miss or write-miss occurs; approximately 21 clock cycles are additionally spent as the miss-penalty. The penalty occurs due to the fact that it has to bring back a whole block of memory.

In conclusion, the cache-full memory subsystem demonstrated superior performance compared to the cache-less system for the given assembly code in terms of hit rate. On the other hand, in terms of miss penalties the cache-less system is far ahead than the cache-full one. Hence, we can conclude that the performance depends on the nature of the assembly code we are providing. More specifically, the hits and misses in the provided assembly code.