Assignment 4: Caching In

Team 45: Baiyoke Nateesuwan, Hill Justin W, Long Katie C, Andrew Lee
Department Computer Engineering
University of Kentucky, Lexington, KY USA
bna225@g.uky.edu, coolhilljr@uky.edu, katieclong@uky.edu, andrew.lee1@uky.edu

Abstract

This project was about building the cache memory to interface with a given slow memory module that can be used with our pipelined processor design.

GENERAL APPROACH

The solution for this problem can be broadly described in the following steps:

1. The slow memory module was initialized using the *$readmemh0()* function, which read the contents of VMEM into a 2D array that served as our memory. The input consisted of instructions and data, which data going into the lower half of memory. Thus, any accesses for instructions pulled from the top half of the memory.
2. For the instruction cache, the group figured out that whenever the instruction cache has a miss and needs to fetch from memory, the cache will output a NOP. This will effectively stall the processor until an actual instruction is returned, as nothing will happen.
3. The caches are direct mapped, with a variable width of *`CACHEWIDTH*. Each entry is one word long, and contains the instruction from the memory. In order to access the instructions, we used a simple modulo operation on the Program Counter. The result of the modulo is what index the cache will look for the data or instruction.
4. The instruction and data cache are connected through a wire name *addr*, that outputs the address that needs to be read from or written to. The only way this addr is to be read is if *strobe* is one and *rnotw* is one. Accordingly, we set those flags to be one on a miss in the instruction cache. The cache is then filled, using prefetching, with the number of instructions specified by the *`CACHEWIDTH* and *`TAG* global variables. This process is similar in the data cache as well.

2. ISSUES

The largest issue we encountered was in attempting to use Dr. Dietz's solution to the last assignment. Our team did not have a successful implementation from the last project, thus we grappled with understanding how Dietz's implementation worked.

Additionally, we had a hard time in implementing the caches. Specifically, the data cache write. This is mostly because of the issue mentioned above. We were unable to get the processor to drive values, and couldn't figure out the issue.

CONCLUSIONS

In attempting this project, we learned a lot about the theory of cache memory. However, when it came time to actually implement the caches, it was difficult to figure out what exactly needed to be done to get it all working.