컴퓨터 구조 개론

LAB6 Cache Report

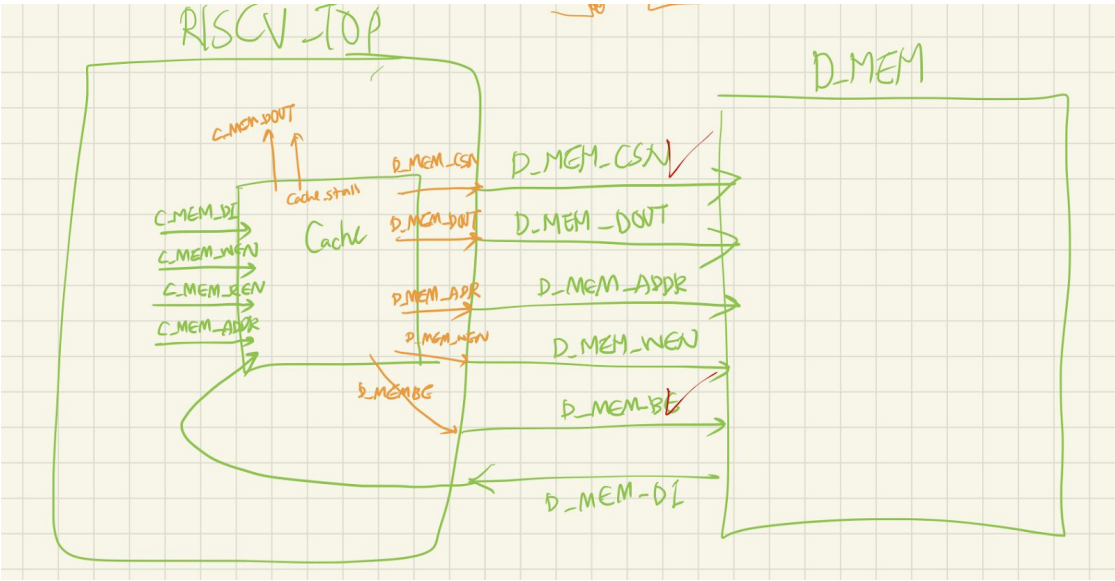
20160542 이현지 20160707 형준하

Introduction

이번 랩에서는 캐시의 구조를 이해하고 이를 구현해보는 것을 목표로 하였다. 특히 캐시 중에서도 direct mapped cache와 set associative cache의 차이에 대해 알아보았고, write back, write allocate 등의 caching method 등을 더 정확하게 이해하고 실제로 구현해보았다.

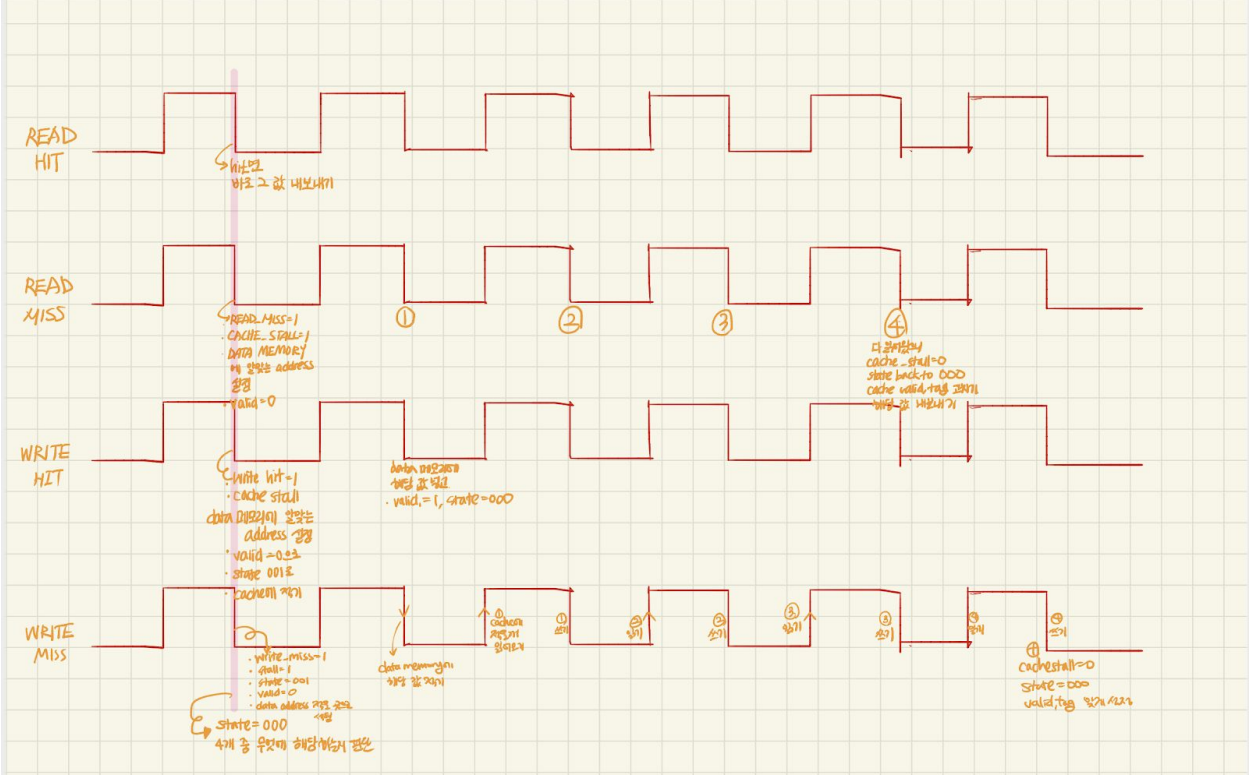
Design

캐시는 Riscv-top 에게 있어서 메모리와 똑같이 보여야 하기 때문에 DI, WEN, ADDR등의 시그널들이 들어가도록 해주었고, DOUT이 나오도록 했다. 여기서 한가지 다른 점은, 캐시가 read 할때를 명시해주어야 cache miss등이 read하지 않는 상황에 발생하지 않기 때문에, REN 시그널 또한 추가해주었다.



Implementation

캐시가 동작하는 경우는 크게 4가지로 나누어 생각하였다. 1) read miss 2) read hit 3) write hit 4) write miss.



Read hit의 경우 주소에 따른 값이 그대로 나가도록 했다.

Read miss의 경우 4개의 state를 만들고, 각 state마다 block offset이 00->01->10->11로 바뀌며 메모리에서 캐시로 negative edge에서 block을 읽어왔다. 이때 pipeline을 stall 시켰다.

Write hit의 경우 한번의 stall을 한다. 첫번째 negative edge에서 cache에 값을 적고, 두번째 neg edge에서 메모리에 값을 쓴다.

Write miss의 경우 똑같이 state를 나누고, 두번째 negative edge에서 메모리에 값을 쓴다. 그 이후로 메모리에서 캐시로 각 block씩 4개의 block을 읽어온다.

Cache miss 에 의한 Stall의 경우에는 IF ID EX MEM stage는 stall 이 되어야 하기 때문에 pipleline register와 control signal들이 앞으로 이동하지 못하게 막아주고, WB stage에는 dummy control signal 을 채움으로서 구현하였다.

Evaluation

Inst, forloop, sort 3가지 경우에 대해서 다 통과하였습니다.

inst의 경우 기존에 25 cycle에서 유지되어 25cycle이 나왔고,

forloop은 기존 97 cycle에서 121로,

sort은 기존 13991 cycle에서 17966로 각각 증가하였습니다.

Discussion

이번 과제에서 6 cycle이 무엇을 의미하는지 이해하느라 오래 걸렸습니다. 이외에는 큰 문제 없이 잘 진행했습니다.

Conclusion

Cache가 무엇인지 배울 수 있었고, write and read 각각의 경우 miss나 hit인 경우 어떻게 clock을 분배해야하는지 배울 수 있었습니다. 다만, data memory access가 주어진 과제에서처럼 ideal하지 않고, 오래 걸리는 경우에서 cache를 통해 시간이 단축되는 것을 직접 확인할 수 있었다면 더 재미있었을 꺼 같다는 아쉬움이 약간 있습니다.