과제명 : #3 MIPS with Cache

과목명 : 컴퓨터구조론(유시환)

분반 : 2

학번 : 32202177

이름 : 서주민

제출일자 : 22/06/20

Free day : No

Project#3 MIPS with Cache

(비주얼 스튜디오, 윈도우 환경에서 c언어와 C++로 작업했습니다!!)

(목표)

- 지난 과제 #2 single cycle MIPS는 Green Sheet에 정의된 ISA에 따라 컴파일러가 변환하는 과정을 에뮬레이터로 대체해보는 것이었다. 그 CPU에뮬레이터는 배열로 구현된 메모리와 레지스터, 변수로 pc를 구현한 형태였다. 그런데 실제로는 메모리에 직접 access하는데엔 시간이 오래 걸린다는 문제가 있다. 이를 보완하기 위해 실제 컴퓨터엔 캐시가 존재한다. 자주 사용하는 데이터를 캐시에 옮겨 두고 사용하면 매번 메모리에 접근하지 않아도 되어 필요한 시간이 줄어든다. 이 캐시의 작동원리를 배우고, #2에서 만든 에뮬레이터에 캐시를 추가하는 것이 이 과제의 목표이다.
-

(구현 내용)

- 사용하는 데이터를 캐시에 저장해서, 매번 메모리에 접근하지 않고 캐시를 먼저 확인한 후 동작하도록 만드는 것이 최소 목표이다. #2와 마찬가지로 4개의 simple과 fib, gcd, input4 총 7개의 예제를 사용할 것이다. 먼저 pc가 명령의 주소를 읽고, 메모리로 가는 게 아닌 캐시로 간다. 캐시에서 tag와 offset을 통해 명령의 캐시에서의 위치를 파악할 수 있다. 캐시에 찾는 명령이 존재하면 hit, 아니면 miss라고 한다. hit일때는 캐시에서 바로 데이터를 가져다 쓰면 되고, Miss일 때는 메모리에서 데이터를 가져와서 캐시에 쓰고 데이터를 사용하면 된다.
- 이때 캐시에 valid한 데이터가 없어서 일어나는 cold hit인 경우엔 그냥 데이터를 가져오면 된다. 캐시는 데이터 뿐 아니라 명령 위치를 나타내는 tag, 유효한 값인지를 나타내는 valid, 메모리와 같은 데이터인지를 나타내는 dirty값이 존재한다. SW명령의 경우 값을 수정하는 것도 캐시에서 하는데, 이 값이 캐시에서 지워지게 되면 메모리는 업데이트 되지 않은 값이므로 계산이 틀어진다. 따라서 캐시에서 값을 수정하면 Dirty비트를 1로 해주고, 이 값을 캐시에서 지워야 한다면 메모리에 제대로 업데이트 시킨 뒤 지워야 한다.

(구현 과정)

- Valid비트는 유효한 값인지를 나타낸다. 처음에는 쓰레기 값 때문이라고 생각해서 캐시를 0으로 초기화 시키고 구현하지 않으려 했다. 그런데 tag가 0인 명령어가 존재하므로, 0으로 초기화 해도 그게 0인 값인지, 빈 값인지 알 수 없음을 알았다.
- 파일을 나눌 만큼 큰 프로그램을 만들어 본 적이 없어서 이론만 알고 있었는데, 이번에 처음으로 헤더와 파일을 여러 개로 나누어 작성해보았다. java로는 클래스별로 나누어

작성하는 게 익숙하지만, C는 클래스 단위가 아니라서 헤더로 연결해서 #include해주어야 한다.

- 기존의 명령어를 정의하는 const_list.h, cpu관련 구조체와 함수가 선언된 cpu.h, 기본적인 cache의 틀을 가진 cache.h, 세컨드 찬스로 더 자세히 정의된 cache_sca.h 4개의 헤더파일이 있다.

- 기존의 cpu가 적힌 cpu.cpp, cache.cpp, 이를 상속받는 cache_sca.cpp과 실행하는 main.cpp 4개의 cpp파일까지 총 8개의 파일로 구성되어있다. 클래스를 사용하기 위해 cpp로 바꾸었다. 구현한 방식은 second chance방식으로, miss가 났을 때 바로 지우지 않고

(마무리)

- #3은 이전 과제 #2에서 만든 cpu에뮬레이터를 업그레이드하는 방식으로 만들게된다. 그런데 내가 이전에 만든 프로그램에서 아쉬운 점이 많아서 신경쓰였다. 그중에서 제일 신경쓰인건 명령을 모두 switch case로 구분해 둔 것이다. 작동은 하지만 instruction의 구조와는 상관없으니 아쉽다는 생각이 들어서, 명령어마다 control signal를 입력하고 알아서 작동되는 구조로 대공사를 시도했었다. 결과적으론 시간이 모자라서 완성하지 못하고 이전 cpu에 cache를 만들게 되었지만, 이 과정도 나에게는 의미가 있어서 지우지 않고 자료로 두었다

- 사실 cpp을 잘 몰라서 c에서 struct로 클래스를 구현하려고 시도했었는데 함수 포인터를 쓰는게 너무 어렵고 가독성이 떨어져서 포기했다..