컴퓨터 SW 개론
Assignment #5 보고서
20180038
박형규

Part A.

먼저 cache가 이루어져 있는 구조대로 struct를 만들었다. cache를 총괄하는 구조체인 cache구조체 안에는 cache에 포함되는 set들을 가르키는 set type의 포인터인 sets와 전체 set의 개수와 line의 개수인 setnum과 linenum을 담았다. set 구조체에는 set안에 포함되어 있는 line들을 가르키는 lines포인터를 담았고, line 구조체 안에는 valid bit을 나타내는 bool type의 v와 tag, 그리고 얼마나 최근에 사용되었는지를 나타내는 LRU값을 담았다.

main 함수에서는 가장 먼저 while문과 getopt함수를 이용하여 s, E, b, t를 받아왔다. 그리고 받아온 s, E, b에 따라 cache의 동적할당을 진행하였다. 그 후, fscanf함수를 이용하여 while문을 통하여 스캔을 진행하였고, operation이 'I'인 경우에는 메모리에 접근하지 않으므로 continue를 이용하여 process함수가 실행되지 않게 하였으며, operation이 'M'일 경우에는 load와 store의 두 번의 접근이 이루어지므로 process함수가 두 번 실행되도록 하였다.

process함수에서는 먼저 set index bit과 tag bit를 분리하여 저장하였다. 그 뒤에 먼저 hit이 발생하였는지 확인하고, hit이 발생하지 않았을 경우에는 miss가 필연적으로 발생하므로 missnum을 하나 증가시켰다. 그 뒤에는 해당하는 set에 빈 line이 존재하는지 확인하였고, 빈 line이 존재한다면 그 line에 집어넣고, 존재하지 않는다면 evicnum을 하나 증가시키고 LRU에 의하여 어느 line과 바꿀지를 정하였다.

update 함수는 line에 포함된 LRU 부분을 지속적으로 업데이트시켜주는 작업을 수행하였다. LRU값은 0이상 E 이하의 값을 가지게 하였는데, 특정 라인에 접근이 발생할때마다 그 라인보다 최근에 쓰이지 않은 라인들의 LRU가 1씩 감소하도록 하여 가장 최근에 쓰이지 않은 LRU값이 0이 되도록 하였다.

Part B.

먼저 우리가 사용하는 캐시의 구조를 살펴보면 s=5, E=1, b=5이므로 32개의 set이 존재하고 각 set에는 단 하나의 line만 존재한다. 또한 각 line은 32byte크기이므로 int형 값이 8개 들어간다. 또한 address bit의 구조를 살펴보면 가장 낮은 5자리의 bit은 block offset bit이고, 그 뒤의 5자리 bit은 set index bit이며 나머지 bit은 tag bit이다.

transpose_submit함수에서 if문을 활용하여 32x32 matrix일 경우에는 Mis32함수가, 64x64 matrix일 경우에는 Mis64함수가, 61x67 matrix일 경우에는 Mis67함수가 호출되도록 하였다.

먼저 Mis32함수는 32x32 matrix를 다룬다. matrix를 8x8로 구분하여, 즉 block의 크기를 8로 하여 과정을 진행하였다. 이는 32x32 matrix에서 8줄마다 set bit가 겹치게 되기 때문이다. 이 때 배열의 row index와 column index가 같을 때(diagonal일 때) set bit이 같아지게 되므로 diagonal일 경우만 뒤로 빼주어 계산하여 miss의 발생을 줄여주었다.

Mis64함수는 64x64 matrix를 다룬다. 64x64 matrix는 matrix를 8x8로 나누어 주었을 때 위의 4줄과 아래 4줄의 set index가 겹치게 된다. 따라서 4칸씩 나누어 transformation을 진행하게 되는데, 이때 4x4 matrix 안에서 먼저 transformation을 진행한 뒤, 추가적으로

block끼리 transformation을 진행할 필요가 있다면 transformation을 진행하도록 하였다. 예를 들어  와 같은 block matrix가 있을 때 1번과 4번 block은 block 안에서 transformation을 진행하고 추가적인 block transformation이 필요하지 않으므로 그대로 두고, 2와 3block은 block안에서 transformation을 진행한 뒤, 2와 3블록이 서로 transformation되어야 하므로 추가적인 transformation을 진행하였다.

Mis61함수는 61x67 matrix를 다룬다. 61x67 matrix는 block의 size를 16으로 하여 32x32 matrix를 다룰 때와 동일하게 diagonal일 경우에 나중으로 빼어 계산하는 방법을 통하여 miss의 발생을 줄였다. 또한 3, 4번째 for문에서 bj < N, bi < M의 조건을 추가해 줌으로써 예외 발생 상황을 control하였다.