컴퓨터 구조

PA4 Write-up

2013-11826 임주경

1. Introduction

Cache simulator 구현을 통해 Cache structure 를 이해하고, Cache policy 를 직접 구현하며 개념을 이해한다.

2. Implementation

구현에 사용한 헤더 파일은 stdio.h / stdlib.h / string.h / getopt.h / "cachelab.h" 이다.

전역 변수로 int 형 변수 hits, misses, evictions, E, S, B, s, b 를 선언하였다. E, S, B, s, b 는 과제 스펙 pdf 에 표기된 내용을 따랐다.

메인 함수의 지역 변수로 line 의 property 값을 저장할 valid, tag, used_order 는 int 형 2 차원 배열로 선언하였다. 이때, valid 와 tag 는 수업시간에 배운 개념과 같으며, used_order 는 LRU replacement 를 따라 eviction 을 구현하기 위한 참조 변수이다. 2 차원 배열의 크기는 int[S][E]로 구현하였다.

Cache policy 는 LRU replacement policy 를 따르도록 구현하여야 하므로, eviction 에 사용하기에 참조할 변수 used_order 를 control 하기 위한 함수 void renew(int, int, int[][], int[][])를 구현하였다. 함수의 인자들은 차례대로 Set index, 최근에 접근된 Line index, valid, used_order 이다. Renew 함수에서는 접근한 Set 에 있는 valid 가 1 인 line 들에 대해서 가장 최근에 접근한 line 의 used_order 는 E - 1 에서부터 시작하며, 사용한지 오래될수록 1 씩 감소하도록 설정한다. 즉, Set 이 eviction 이 필요한 경우, used_order 가 0 값을 갖는 line 이 replacement 되어야 한다. 이 함수는 access 가 일어날 때마다 호출된다.

다음 함수는 cache 에 access 하는 주 함수 void access(unsigned long long, int[][], int[][], int[][]) 함수이다. 함수 인자는 차례대로 명령어 address, valid, tag, used_order 이다. 명령어 주소를 읽어 bit 연산을 통해 해당하는 Set index 와 tag bit 를 얻어낸다. 다음, 해당하는 Set 에서 line 수 만큼 loop 를 통해 필요한 tag 가 존재하는지 Hit, Miss, Eviction 여부를 판별한다. 각 case 에 맞게 전역변수 hits, misses, evictions 중 하나가 증가하게 되며, 호출마다 renew 함수를 호출해서 used_control 을 새롭게 설정한다.

마지막 메인 함수에서는 trace file 을 getopt 함수를 통해 명령어를 읽어 변수 S, E, B, s, b 값을 초기화한다. 다음, 명령어를 한 줄 씩 읽으며, Simulation 을 진행한다. 이때, 명령어 연산 'I'는 무시하며,

'L'과 'S'는 access 함수를 한 번 호출하고, 'M'는 access 함수를 두 번 호출하도록 한다.

최종적으로 변수 hits, misses, evictions 값을 출력하며 마친다.

3. Result

구현 결과 test-csim 실행을 통해, csim-ref 의 구현과 같은 결과를 얻을 수 있으며, 최종 점수 54 점이 확인되었다. Valgrind 를 통해 sample.trace 를 얻어서 마찬가지로 시뮬레이션을 통해, s, E, b 를 동일한 조건으로 비교해보아도 같은 결과를 얻을 수 있었다.