컴퓨터구조 (2017년 2학기)

과제 #2 – Cache Simulator 구현

0. 과제 제출 및 유의 사항

A. 수업 홈페이지에 링크된 과제 제출 사이트에 제출 기한에 맞춰서 온라인으로 제출한다. 제출 사이트에는 다른 과목들의 과제들도 같이 제출할 수 있도록 되어 있으므로 해당 과목과 과제를 정확하게 선택하도록 주의한다.

B. **소스 코드와 MS 워드로 작성된 보고서의 pdf 파일들을 하나의 zip 파일로** 묶고 파일 이름은 "이름_hw1.zip"으로 한다. 보고서에 코드의 실행 방법을 자세히 명시해야 한다. 사용 요령을 명확하게 명시하지 않아 실행이 되지 않은 코드의 경우 과제를 0 점 처리한다.

C. 제출 마감 시간을 기준으로 24 시간씩 지날 때마다 만점에서 25%씩 감점처리한다.

1. 개요

이번 과제는 C나 C++을 이용하여 cache simulator 프로그램을 구현하는 것이다. Cache simulator는 주어진 memory trace와 cache 설정(cache 전체 크기, block 크기, associativity)에 대해 hit rate를 출력한다.

2. 프로그램 입력

Cache simulator는 다음과 같이 동작해야 한다. Cache simulator의 실행 파일이름을 simple_cache_sim (윈도우에서는 simple_cache_sim.exe)이라고 하면 윈도우 커맨드 창이나 리눅스의 터미널 창에서 아래과 같은 입력에 대해 동작해야 한다.

```
$ simple_cache_sim swim.trace 1024 16 4
```

위와 같이, cache simulator는 4개의 입력을 필요로 한다. 각 입력들은 다음과 같다.

- 메모리 트레이스 파일
- cache의 전체 바이트 단위 크기
- cache block 하나의 바이트 단위 크기
- associativity, 즉 Direct mapped cache의 경우에는 1이며, 4-way set associative cache는 4가 된다. 일반적으로 N-way associative cache에서 N 값을 입력한다.

(주의 사항: associativity가 1 이상인 경우 block replacement는 Least Recently Used 방법을 이용해서 구현한다.)

3. 메모리 트레이스 파일

메모리 트레이스 파일은 cache simulator의 첫 번째 입력으로 주어진다. 트레이스 파일은 어떤 프로그램을 수행하면서 발생한 데이터 메모리 접근들을 담고 있다. 명령어 fetch는 포함되어 있지 않다. 트레이스 파일의 각 라인들은 하나의 메모리 접근을 의미한다. 각 라인들은 3개의 필드로 구분되어 있는데, 이 중 두 번째 필드는 메모리에 접근할 때의 32-bit 바이트 주소를 의미하며, 트레이스 파일에서는 16진수로 표현되어 있다. 예를 들어 0xff32e100는 메모리 주소 4281524480번지를 의미한다. 각 필드들은 space 한 칸으로 구분되어 있다. 프로그램 작성을 위해 필요한 예제 트레이스 파일들은 수업 홈페이지에 링크되어 있다.

4. 프로그램 출력

Cache simulator는 아래와 같은 형식으로 출력되어야 한다. 1-3번째 출력들은 입력으로 주어진 cache 설정에 대해 tag, index, offset 에 해당하는 필드의 길이들을 나타낸다. 4번째 출력은 cache simulation의 결과인데 총 메모리 접근 횟수, hit count, hit rate의 세 개의 정보를 출력한다. 총 메모리 접근 횟수는 트레이스 파일에 들어있는 메모리 접근 횟수를 나타내며 이는 트레이스 파일의 라인 수와 동일하다. Hit rate는 총 메모리 접근 횟수와 hit count로부터 계산할 수 있다.

```
$ simple_cache_sim gcc.trace 1024 16 2
tag: 23 bits
index: 5 bits
offset: 4 bits
Result: total access 515683, hit 468811, hit rate 0.91
```

5. 컴파일 및 실행 환경

과제를 작성하는 환경은 어떤 환경이든 상관없다. 제출한 소스 코드는 Ubuntu 리눅스 14.04 LTS 64-bit 버전에서 g++ 4.8.2를 이용하여 컴파일하고 테스트될 것이다. 그러므로 cache simulator는 표준 C/C++를 이용하여 작성하여 어떤 플랫폼에서는 컴파일이 될 수 있도록 해야 한다. 또한 예제 트레이스 이외에 다른 트레이스 파일들도 이용해서 동작을 검증할 것이다.

프로그램은 다음과 같은 경우 에러 메시지를 출력하고 종료해야 한다. 각 상황에 필요한 에러 메시지들은 각자 알아서 정하도록 한다.

- 프로그램의 입력 인자가 4개가 아닐 경우
- 존재하지 않은 트레이스 파일 입력했을 경우
- Cache block의 크기가 2의 지수승이 아닌 경우
- Associativity가 1, 2, 4, 8 이외의 값이 입력된 경우
- Cache 전체의 크기가 (cache block 크기)x(associativity)의 배수가 아닐 경우

6. 제출물

- 보고서: 구현한 cache simulator 의 전체 구조를 설명하고, 중요한 코드 부분들에 대해 구현

방법을 설명해야 함. 분량은 3페이지 이내로 하며, 보고서 표지는 사용하지 말 것 (사용시 감점). 보고서는 파일을 1차 과제와 마찬가지로 웹디스크에 업로드 해야 하고, 출력물은 조교에게 제출한다. (제출 기한은 수업 홈페이지 참고)

- 소스코드: 소스코드는 한 개의 파일로 만들어서 1차 과제 때와 같이 웹디스크에 업로드 할 것. 파일 이름은 lab2_학번_이름.cc 로 할 것.

7. 평가 주안점

- 컴파일이 에러 없이 잘 되는가
- 예외 사항에 대한 처리가 구현되어 있는가
- 다양한 cache 설정에 대해 프로그램이 동작하는가
- 결과가 정확한가