**Cache\_sim**

**13 신우현**

**14 안태건**

**14 부승혁**

**<코드 설명>**

**Cache를 구조체로 선언**

**typedef struct \_Cache {**

**int valid;**

**long long tag;**

**long long data;**

**int counter;**

**}Cache;**

**Cache \*\*cache**

Cache\_size, block\_size, assoc의 값에 따라 set\_index의 개수를 구하고 cache를 동적할당하여 초기화한다.

cache= (Cache \*\*)malloc(sizeof(\*cache)\*set\_index);

for (i = 0; i < set\_index; i++)

cache[i] = (Cache \*)malloc(sizeof(\*\*cache)\*assoc);

그리고 index\_bits, byteoffset\_bits, tag\_bits를 구한다.

각각의 비트들로 address의 tag, index값을 추출한다. 구한 index로 cache에 접근하며

address의 tag와 cache의 tag값이 같고 valid비트도 1이라면 hit된다 그렇지 않으면 miss이다.

Replacement policy가 LRU인 경우는 counter값을 hit될때마다 조절한다.

Miss되었을 때 LRU의 경우는 counter가 0인 cache에 address의 tag값을 대입한다.

(counter값도 조절한다)

반면에 Random인 경우 무작위로 선택되어진 cache에 address tag값을 대입한다.

p.s) log2()함수와 pow()함수를 사용하기 위해 math.h를 include 하였고

time.h를 include 하여 난수 생성시 시드값으로 time()함수를 사용했다.

**Trace file : test\_cachesize\_blocksize.trc**

**Trace file : test\_cachesize\_blocksize.trc**

**Trace file : test\_assoc.trc**

**Trace file : test\_repl.trc**

**<contribution >**

전체 코드 작성, 코드 분석 및 오류수정 – 안태건 (5)

결과값 검토, graph1,3 작성 – 신우현 (2.5)

결과값 검토, graph2,4 작성 – 부승혁 (2.5)