컴퓨터구조 HW2

중요!

제출 파일중 202002398lru.c는 lru 알고리즘을 적용한 코드이고, 202002398radom.c는 랜덤 알고리즘을 적용한 코드입니다.

202002398 이도환

2.1

캐시 블록 구조체(CacheBlock) 정의

valid: 캐시 블록이 유효한지 여부를 나타냅니다.

tag: 태그 비트로, 메모리 주소의 일부분을 저장합니다.

timestamp: LRU(Least Recently Used) 알고리즘을 구현하기 위한 시간 값을 저장합니다.

dirty: 쓰기 동작이 발생했는지 여부를 나타냅니다.

글로벌 변수 정의

time_count: LRU 알고리즘을 구현하기 위한 시간 카운터.

total_set: 캐시의 세트 수.

i_total, i_miss: instruction 캐시의 총 접근 횟수 및 miss 횟수.

d_total, d_miss, d_write: data 캐시의 접근 횟수, miss 횟수 및 메모리 쓰기 횟수.

trace: 메모리 접근 트레이스를 저장하는 배열.

trace_length: 트레이스 길이.

instruction_cache, data_cache: instruction 캐시 및 data 캐시 배열.

main 함수

입력으로부터 트레이스를 읽어와 trace 배열에 저장합니다.

다양한 캐시 크기, 블록 크기, 연관도 조합에 대해 캐시 시뮬레이션을 수행합니다.

각 조합에 대해 solution 함수를 호출하여 결과를 출력합니다.

solution 함수

캐시 크기, 블록 크기, 연관도를 매개변수로 받아 캐시를 초기화합니다.

트레이스의 각 접근에 대해 읽기(read_op), 쓰기(write_op), 인스트럭션 페치(fetch_inst)를 수행합니다.

각 동작이 끝난 후 miss rate와 메모리 쓰기 횟수를 계산하여 출력합니다.

시뮬레이션이 끝난 후 캐시 메모리를 해제합니다.

read_op 함수

주어진 주소에 대해 데이터 캐시에서 읽기 동작을 수행합니다.

캐시 히트 시 타임스탬프를 업데이트하고, miss 시 LRU 알고리즘을 사용하여 교체 블록을 결정합니다.

write_op 함수

주어진 주소에 대해 데이터 캐시에서 쓰기 동작을 수행합니다.

캐시 히트 시 타임스탬프를 업데이트하고, miss 시 LRU 알고리즘을 사용하여 교체 블록을 결정합니다.

쓰기 동작의 경우 해당 블록을 dirty 상태로 설정합니다.

fetch_inst 함수

주어진 주소에 대해 인스트럭션 캐시에서 페치 동작을 수행합니다.

캐시 히트 시 타임스탬프를 업데이트하고, miss 시 LRU 알고리즘을 사용하여 교체 블록을 결정합니다.

find_lru_block 함수

LRU 알고리즘에 따라 가장 오랫동안 사용되지 않은 블록을 찾습니다.

update_timestamp 함수

주어진 블록을 제외한 모든 블록의 타임스탬프를 증가시키고, 사용된 블록의 타임스탬프를 0으로 설정합니다.

2.2 결과 표

Trace1.txt 결과 표

cache size	block size	associative	d-miss rate	i-miss rate	mem write
1024	16	1	0.2376	0.2133	2636
1024	16	2	0.1625	0.2064	1623
1024	16	4	0.1503	0.2101	1461
1024	16	8	0.1445	0.2118	1409
1024	64	1	0.2731	0.0941	2856
1024	64	2	0.1936	0.0920	1916
1024	64	4	0.1670	0.0917	1467
1024	64	8	0.1649	0.0924	1436
2048	16	1	0.1928	0.1704	2080
2048	16	2	0.1219	0.1579	1126
2048	16	4	0.1122	0.1518	1008
2048	16	8	0.1095	0.1517	998
2048	64	1	0.2189	0.0738	2368
2048	64	2	0.1329	0.0715	1140
2048	64	4	0.1159	0.0713	847
2048	64	8	0.1080	0.0734	778
4096	16	1	0.1064	0.1304	902
4096	16	2	0.0904	0.1136	728
4096	16	4	0.0829	0.1075	660
4096	16	8	0.0810	0.1012	651
4096	64	1	0.1048	0.0575	831
4096	64	2	0.0876	0.0502	578
4096	64	4	0.0752	0.0480	461
4096	64	8	0.0712	0.0469	425
8192	16	1	0.0773	0.1087	536
8192	16	2	0.0611	0.0832	288
8192	16	4	0.0544	0.0783	244
8192	16	8	0.0506	0.0766	217
8192	64	1	0.0701	0.0488	507
8192	64	2	0.0537	0.0359	281
8192	64	4	0.0490	0.0330	239
8192	64	8	0.0469	0.0325	230
16384	16	1	0.0559	0.0655	191
16384	16	2	0.0480	0.0564	107
16384	16	4	0.0434	0.0490	31
16384	16	8	0.0421	0.0464	21
16384	64	1	0.0432	0.0295	213
16384	64	2	0.0306	0.0260	115
16384	64	4	0.0267	0.0234	85
16384	64	8	0.0252	0.0233	73

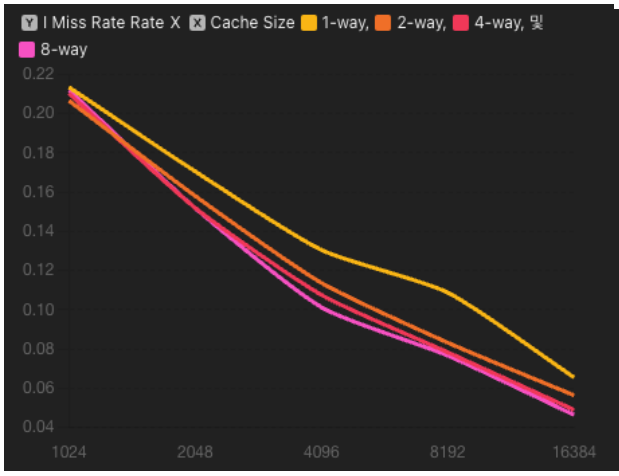
Trace2.txt 결과 표

cache size	block size	associative	d-miss rate	i-miss rate	mem write
1024	16	1	0.1335	0.0937	1177
1024	16	2	0.0596	0.0941	531
1024	16	4	0.0482	0.0790	398
1024	16	8	0.0467	0.0776	379
1024	64	1	0.1378	0.0521	1305
1024	64	2	0.0662	0.0536	674
1024	64	4	0.0504	0.0438	489
1024	64	8	0.0503	0.0451	430
2048	16	1	0.0445	0.0583	315
2048	16	2	0.0362	0.0527	253
2048	16	4	0.0359	0.0398	253
2048	16	8	0.0353	0.0378	241
2048	64	1	0.0314	0.0353	220
2048	64	2	0.0250	0.0305	183
2048	64	4	0.0234	0.0247	170
2048	64	8	0.0193	0.0216	126
4096	16	1	0.0310	0.0421	152
4096	16	2	0.0214	0.0294	87
4096	16	4	0.0172	0.0272	56
4096	16	8	0.0165	0.0275	49
4096	64	1	0.0229	0.0200	151
4096	64	2	0.0135	0.0150	69
4096	64	4	0.0144	0.0109	78
4096	64	8	0.0137	0.0111	81
8192	16	1	0.0229	0.0249	77
8192	16	2	0.0162	0.0210	19
8192	16	4	0.0145	0.0188	4
8192	16	8	0.0145	0.0195	3
8192	64	1	0.0144	0.0116	79
8192	64	2	0.0077	0.0096	19
8192	64	4	0.0062	0.0081	11
8192	64	8	0.0057	0.0082	3
16384	16	1	0.0172	0.0158	36
16384	16	2	0.0150	0.0131	9
16384	16	4	0.0145	0.0102	0
16384	16	8	0.0145	0.0093	0
16384	64	1	0.0075	0.0068	24
16384	64	2	0.0063	0.0055	9
16384	64	4	0.0057	0.0043	0
16384	64	8	0.0057	0.0041	0

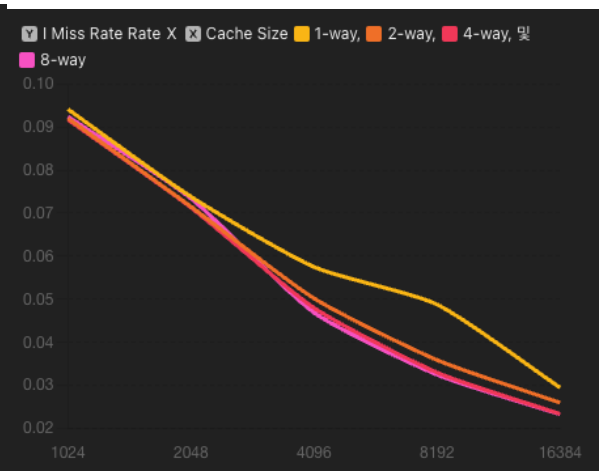
2.3 캐시 크기가 증가할수록 missrate는 감소합니다. 또한 연관도가 높을수록 miss rate가 낮아지는 경향을 파악할 수 있습니다.

LRU 알고리즘

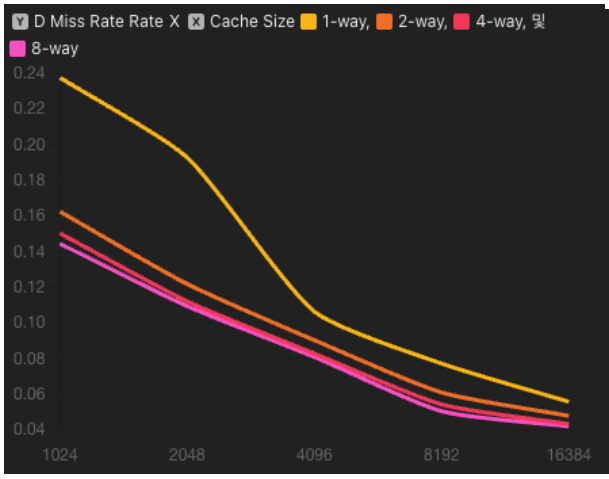
I cache 16byte



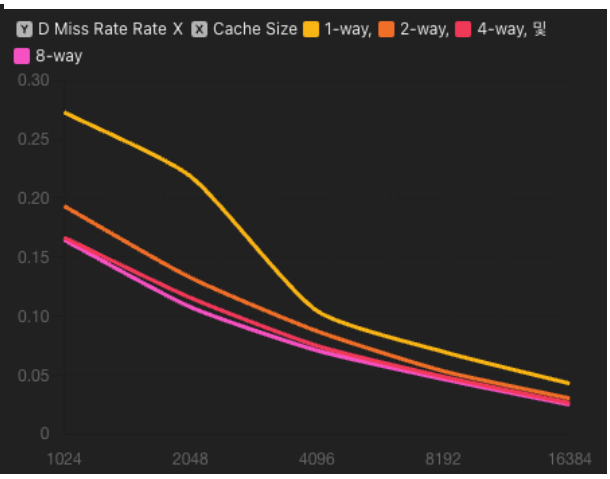
I cache 64byte



D cache 16byte

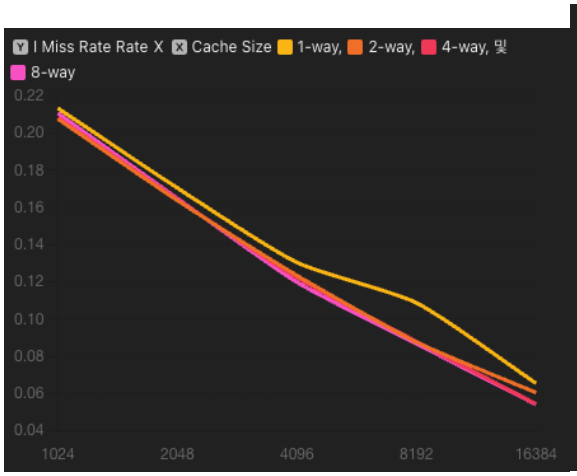


D cache 64byte

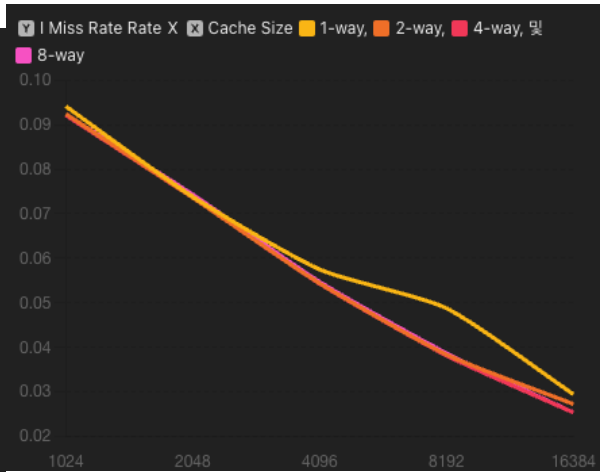


랜덤 알고리즘

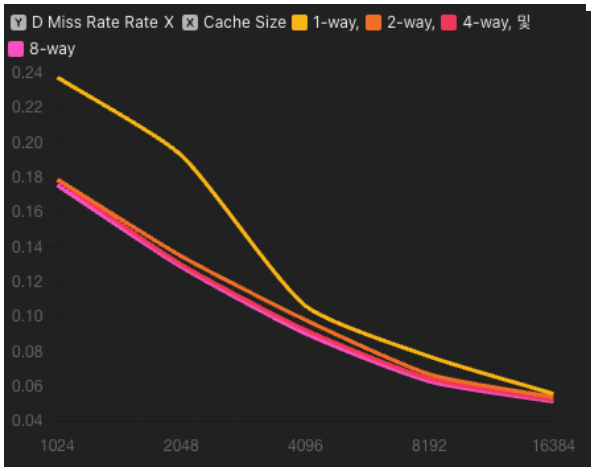
I cache 16byte



I cache 64byte



D cache 16byte



D cache 64byte