Computer Architecture HW2. cache Simulator

컴퓨터공학부 201901208 김현동

1.코드 구현 설명

Main 함수에서 mode, address를 불러와 trace 2차원 리스트에 저장합니다. Cache에 명령을 입력하는 역할을 합니다. cache, block, associative 크기 별로 solution 함수를 실행합니다.

solution 함수는 trace의 입력 전체를 실행합니다. 입력 mode에 따라 read, write, fetch 기능을 수행합니다. 기능에 따라 data, instruction cache 접근 횟수가 증가합니다. 또한 각 기능에서는 time_count를 증가시켜 높을수록 최근에 접근했음을 알립니다. 이는 다음에 LRU방식에 사용됩니다. 전체 명령을 시행한 후 data, instruction miss rate를 계산 후 출력형식에 맞게 출력합니다.

read 기능은 data cache에 접근합니다. direct mapped 방식이 아니라면 set이 형성됩니다. data cache valid==1, 같은 tag가 존재하면 hit로 판단합니다. miss라면 d_miss를 증가시키고 evit 함수를 통해 데이터를 저장할 block의 index를 계산하여 cache에 입력합니다. 또한 메모리와 내용이 다르다면(dirty 상태) memory write를 실행합니다.

write 기능은 read 기능과 유사하나 데이터가 dirty 상태로 변한다는 점에서 차이가 있습니다. write back 방식을 사용해서 cache에 write하는 순간 메모리와 다른 데이터가 되기 때문입니다.

fetch 기능은 instruction을 불러옵니다. read 기능과 유사하나 data cache가 아닌 instruction cache에 접근한다는 점에서 차이가 있습니다. 또한 dirty 상태를 고려하지 않습니다.

evict 함수는 LRU 기능을 사용하기 위함입니다. 각 cache에서 time 정보를 불러옵니다. 낮을수록 접근을 안 한 데이터로 판단됩니다. time이 가장 낮은 데이터를 찾아 index를 반환합니다. cache의 각 기능에서 데이터 삽입 또는 수정 위치를 계산합니다.

2.출력결과 표

Trace 1

trace1-out.txt | trace2-out.txt | trace1-out-result.csv × | trace2-out-result.csv

1 to 40 of 40 entries Filter

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	0.092	1916
1024	64	4	0.167	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	0.108	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	0.081	0.1012	651
4096	64	1	0.1048	0.0575	831
4096	64	2	0.0876	0.0502	578
4096	64	4	0.0752	0.048	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	0.049	0.033	239
8192	64	8	0.0469	0.0325	230
16384	16	1	0.0559	0.0655	191
16384	16	2	0.048	0.0564	107
16384	16	4	0.0434	0.049	31
16384	16	8	0.0421	0.0464	21
16384	64	1	0.0432	0.0295	213
16384	64	2	0.0306	0.026	115
16384	64	4	0.0267	0.0234	85
16384	64	8	0.0252	0.0233	73

Show 100 per page

Trace 2

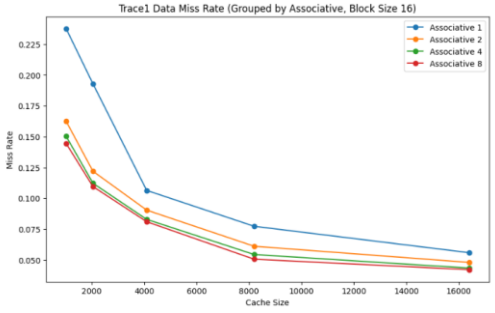
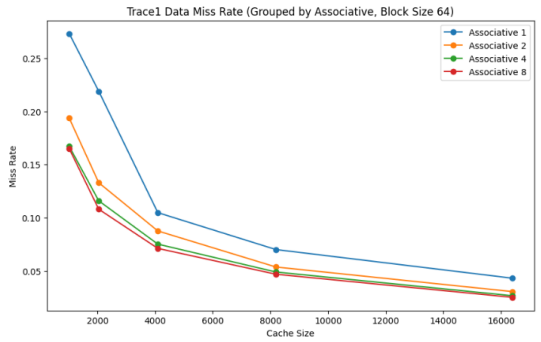
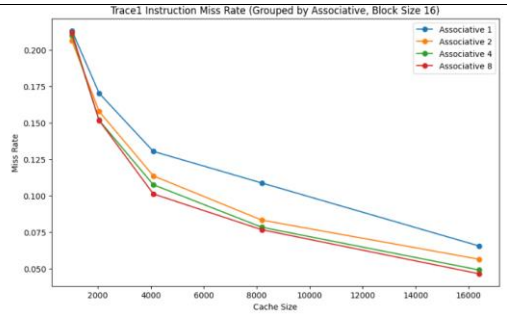
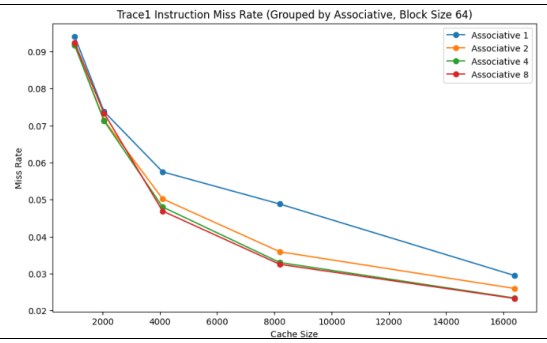
trace1-out.txt | trace2-out.txt | trace1-out-result.csv | trace2-out-result.csv ×

1 to 40 of 40 entries Filter

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	0.079	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	0.025	0.0305	183
2048	64	4	0.0234	0.0247	170
2048	64	8	0.0193	0.0216	126
4096	16	1	0.031	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	0.02	151
4096	64	2	0.0135	0.015	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	0.021	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	0.015	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

Show 100 per page

3. 그래프

	16 byte	64 byte
Data		
Instruction		

1. cache size가 증가할수록 Miss rate가 감소하는 경향을 보입니다. 이는 시간 지역성, 공간 지역성으로 설명 가능합니다. cache size 증가는 더 많은 정보를 저장할 수 있게 합니다. 인접한 데이터 접근 확률을 증가시고, 데이터를 더 오래 유지함으로써 cache miss를 줄일 수 있습니다. 작은 cache size는 cache에 저장할 수 있는 block블수가 제한되어 높은 충돌과 교체로 인해 Miss rate이 증가하는 것을 확인할 수 있습니다.

2. associate이 증가할수록 miss rate이 감소하는 경향을 보입니다. 이는 충돌 가능성이 낮아지기 때문입니다. 충돌은 서로 다른 메모리 주소가 같은 index에 매핑 될 때 발생합니다. n이 증가할 수록 동일한 index에 여러 개의 블록을 저장할 수 있게되어 충돌 가능성이 감소합니다.

이번 실험에서 이 경향은 Data cahce에서 더 크게 보여줍니다. 접근 패턴이 다르기 때문이라고 예측합니다. Data cache에서는 read, write 기능을 수행하지만, instruction cache에서는 fetch(read) 기능만을 주로 사용합니다. read, write에서 충돌이 더 자주 발생하기 때문에 이런 결과가 발생했습니다.

3. 블록 크기가 증가할수록 miss rate가 감소하는 경향을 보입니다. 공간 지역성을 증가시켜 cache miss를 줄입니다.