컴퓨터 구조 실험

Project 4 과제

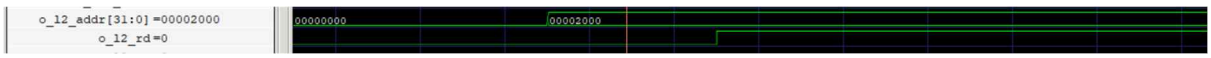
수업 명 : 컴퓨터구조실험
과제 이름 : project4
담당 교수님 : 이성원 교수님
학 번 : 2019202005
이 름 : 남종식

■ 실험 내용

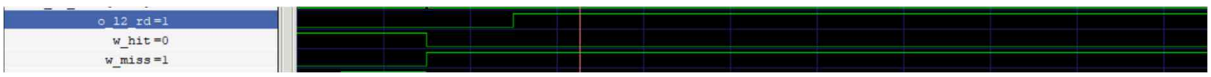
- **정렬 프로그램과 Benchmarks 프로그램에 각각 적합한 cache를 찾고, 각각의 프로그램의 적합한 cache가 왜 다른 건지 프로그램을 분석하여 비교**

먼저 insertion sort의 gtk wave부터 확인해보겠습니다.
Instruction cache는 해당 명령어들을 access할 때 처음 읽어온다면 miss 되면서 cache에 저장하는데, 이때 블록 사이즈가 4개의 word기 때문에 4개씩 읽어옵니다. 따라서 명령어 한 개씩 수행할 때 4개가 각각 Hit되면서 access됩니다.

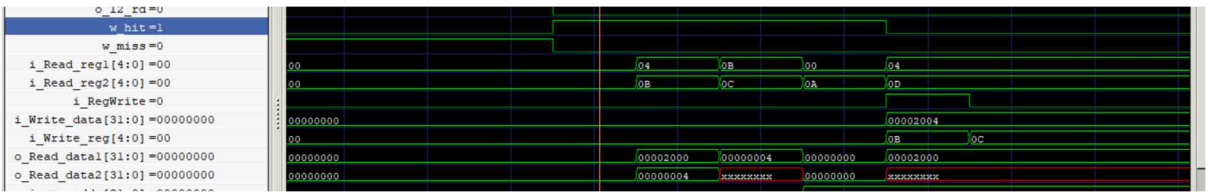
일단 위 사진에서 hit이 일어난 모습을 확인할 수 있고 이때는 당연히 miss가 일어나지 않은 모습을 확인할 수 있습니다. 이 상항은 L1 instruction cache에서 hit가 일어났으므로 L2 instruction cache를 사용할 필요가 없기 때문에 다음 사진에서 o_l2_rd의 값은 0으로 되어 있는 것을 알 수 있습니다.



만약에 miss가 일어났다면 L2 instruction cache에 접근할 것입니다. 그래서 아래 사진에서 o_l2_rd의 값이 1로 설정된 모습을 확인할 수 있습니다.



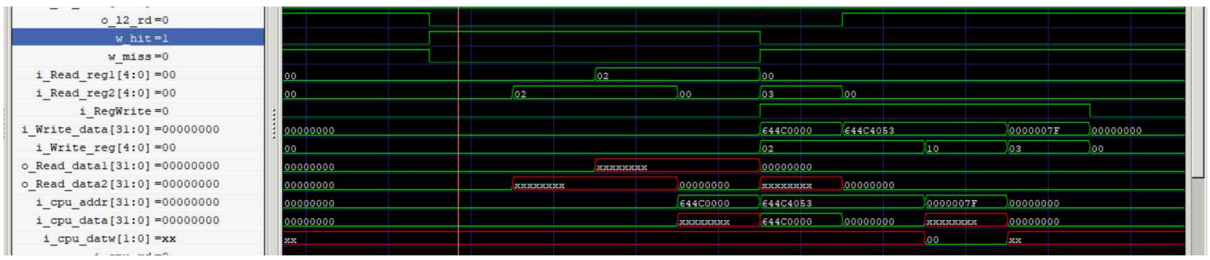
그 후 다시 hit이 일어나고 L2 instruction cache에서 instruction을 잘 가져오는 모습을 아래 사진을 통해 확인할 수 있습니다.
이 때 한번 읽은 instruction은 반복문에 의해 다시 수행될 때는 cache에 접근하여 사용할 수 있게 됩니다.



Instruction cache를 확인했으니 data cache부분을 확인해보겠습니다.
Data cache는 instruction이 Load나 Store일 때 access하고, 이때 data를 읽어와서 cache에 저장합니다. L1 data cache에서 hit이 일어났다면 당연히 miss의 값은 0일 것이고 L2 data cache에 접근하지 않을 것입니다. 하지만 miss가 발생한다면 L2 data cache에 접근할 것이고 o_l2_rd의 값이 1이 될 것입니다. 그리고 hit이 일어나는 상황을 확인할 수 있습니다.

다음은 random access상황을 보겠습니다.
Random access의 instruction cache의 동작방식은 insert sort의 instruction cache의 동작방식과 Miss/Hit의 규칙이 동일합니다.

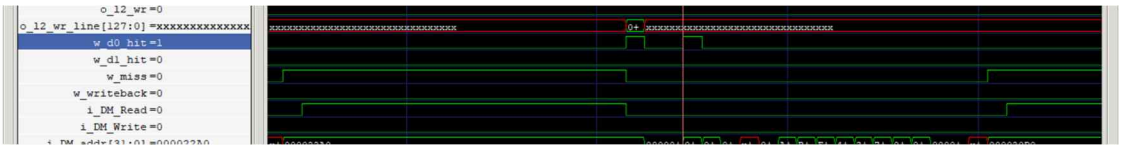
Hit가 일어나서 instruction을 가져오는 모습을 아래 사진에서 확인할 수 있습니다.



Miss가 발생해서 instruction을 L2cache에서 가져오는 모습을 확인할 수 있습니다.

다음은 data cache를 보겠습니다.
마찬가지로 Load와 Store 명령어가 올 때만 접근하게 됩니다. 하지만 Random

access 같은 경우에는 insert sort와 달리 계속 랜덤으로 값에 접근하게 되므로 cache에 저장되어 있는 값이 아닙니다. 앞서 access 했던 메모리와 현재의 메모리가 아예 상관이 없기 때문에 따라서 Miss와 Hit가 계속 같이 발생하게 됩니다.



그리고 각각의 Benchmark들의 Suitable Cache가 다른 이유는 당연히 각 프로그램마다 구현된 용도가 다르기 때문이라고 생각합니다. 일 처리 부분에서 각각이 최적화된 부분이 다르기 때문에 적합한 Cache가 다를 수 밖에 없다고 생각합니다.

- How I/D L1 hit operates, how I/D L1 miss operates?
  캐시에 데이터 또는 명령어가 요청되면, L1 캐시는 주소를 검색하여 해당 데이터를 찾습니다. 요청한 데이터가 L1 캐시에 존재하는 경우, 이를 hit이라고 하며, 해당 데이터는 즉시 제공됩니다.
  요청한 데이터가 L1 캐시에 존재하지 않는 경우, 이를 miss라고 합니다. 이 경우, L1 캐시는 주로 L2 캐시로부터 해당 데이터를 가져와 L1 캐시에 저장합니다.

- When and how main memory (L2) operates?
  L1 캐시가 Miss를 발생시키면, L2 캐시로부터 데이터를 가져오게 됩니다. L2 캐시는 L1 캐시보다 큰 용량을 가지므로, 보다 많은 데이터를 저장할 수 있습니다. 만약 L2 캐시에서 데이터를 찾을 수 없는 경우, 이는 L2 캐시 Miss가 발생하며, 데이터는 다음 단계의 메모리에서 가져와 L2 캐시에 저장됩니다.

- The differences between Insertion sort and Random access in terms of cache behaviors?
  삽입 정렬은 연속적인 요소들을 삽입하면서 정렬하는 반면, 랜덤 액세스는 임의의 위치에서 데이터에 액세스합니다.

삽입 정렬은 인접한 요소들을 자주 교환하고 액세스하는 반면, 랜덤 액세스는 요소들을 임의로 접근하므로 캐시의 Locality 특성을 더 잘 활용할 수 있습니다.
따라서 삽입 정렬은 Hit가 더 자주 발생할 가능성이 있으며, 랜덤 액세스는 Miss가 더 자주 발생할 가능성이 있습니다.

■ What if data size increase?
데이터 크기가 증가하면 캐시의 성능에 영향을 줄 수 있습니다.
데이터 크기가 L1 캐시보다 큰 경우, L1 캐시에 모든 데이터를 저장할 수 없으므로 Miss가 더 자주 발생할 가능성이 높아집니다.
이로 인해 액세스 지연이 발생하고, 메인 메모리 액세스 비용이 상승하게 됩니다. 대용량 데이터의 경우 L2 캐시와 메인 메모리 간의 데이터 전송 비용도 증가할 수 있으며, 이로 인해 전체적인 실행 시간이 증가할 수 있습니다.

■The original purpose of the benchmark programs given (not to measure the performance)?

1. **cc1 컴파일러**
cc1 컴파일러는 GNU C 컴파일러 버전 2.5.3을 기반으로 한 컴파일러입니다. 이는 C 소스 코드를 GIMPLE이라는 중간 표현 형식으로 번역하는 컴파일러입니다. cc1 컴파일러는 코드를 파싱하고 타입 체크, 심볼 해결 등의 최적화와 분석 작업을 수행하며 최적화된 어셈블리 코드를 생성합니다.

2. **ijpeg 유틸리티**
ijpeg 유틸리티는 인메모리 이미지의 압축 및 해제를 위한 도구입니다. 이 도구는 특히 JPEG이미지 형식에 사용됩니다. 이 유틸리티를 사용하면 고해상도 이미지를 더 작은 파일 크기로 압축하여 저장하거나 네트워크를 통해 전송하기가 쉬워집니다. 또한 JPEG 이미지를 해제하여 보거나 편집할 수 있는 기능을 제공합니다.

3. **Perl 인터프리터**

Perl 인터프리터는 Perl 프로그래밍 언어로 작성된 프로그램을 실행하는 도구입니다. Perl은 고수준의 동적이고 다목적인 프로그래밍 언어로, 강력한 텍스트 처리 능력과 정규 표현식 지원으로 알려져 있습니다. Perl 인터프리터는 Perl 스크립트를 읽고 해석하여 각 줄을 실행합니다. 내장 함수와 모듈의 다양한 기능을 제공하여 시스템 관리, 웹 개발 및 데이터 조작과 같은 다양한 작업에 적합합니다.

■ Discuss about the memory access patterns of the benchmark programs in the aspect of algorithm (for example: there is a matrix multiplication, so the program may show 2d array access)

삽입 정렬은 배열에서 원소를 순서대로 선택하여 이미 정렬된 부분 배열에 삽입하여 최종적으로 정렬된 배열을 만드는 간단한 정렬 알고리즘입니다. - 액세스 패턴을 살펴보면, 이 정렬은 temporal locality와 spatial locality를 이용합니다. 이 알고리즘에서는 각 원소가 배열에서 이전 원소들과 비교되며 올바른 위치에 삽입됩니다. 작은 원소는 오른쪽으로 이동하여 올바른 위치를 찾을 때까지 이동합니다. 이 과정은 배열 전체가 정렬될 때까지 반복됩니다. 메모리 액세스 측면에서 삽입 정렬은 일반적으로 배열의 인접한 원소에 접근합니다. 배열은 왼쪽에서 오른쪽으로 순차적으로 접근하며, 각 원소는 이전 원소와 비교됩니다. 이러한 순차적인 액세스 패턴은 인접한 원소들이 연속적으로 접근되는 지역성을 가지게 됩니다. 삽입 정렬은 원소를 배열 내에서 이동시키면서 작동하기 때문에 배열 원소에 대한 반복적인 읽기 및 쓰기 작업이 필요합니다. 이는 배열의 각 원소를 여러 번 액세스하여 비교하고 이동합니다.

각각의 조건에 따라 sets와 block, associativity를 수정해주며 시뮬레이터를 돌렸고, 3가지의 Benchmark를 모두 실행해서 AMAT를 구하였습니다.
AMAT를 구하는 공식은 Level 1에서는 $L1Hit\_time + L1Miss\_rate \times L1Miss\_penalty$ 이고, Level 2의 cache는 $L1Miss\_penalty = L2Hit\_time + L2Miss\_rate \times L2Miss\_penalty$ 입니다.
이번 과제에서의 Level 1 Hit time은 1 Cycle, Level 2 Hit time은 20 Cycles로 주어졌고, Miss penalty의 경우에는 200으로 주어졌습니다.

■ Perform experiments with several cache configurations and discuss about the

results &Discuss about the best fit cache configuration for each benchmark program and compare and discuss about the differences among the configurations

Simulation 1 Unified vs splits

Block size = 16, Associativity = 1 고정

CC1

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.3411	69.5015	0.3697	0.2248	66.7822
128	0.2745	56.2328	0.3118	0.1654	55.1577
256	0.2203	45.4461	0.2558	0.1154	44.3371
512	0.1667	34.7815	0.2131	0.0793	36.2120

최적의 cache는 512set의 unified cache입니다.

ijpeg

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.1880	38.8179	0.2585	0.2571	52.6390
128	0.1006	21.3866	0.1089	0.2194	27.6353
256	0.0457	10.4573	0.0212	0.1484	10.8647
512	0.0322	7.8100	0.0047	0.0791	5.3070

256set의 split cache를 사용할 때 128set을 사용할 때 보다 set이 2배 2증가하였는데 더 많은 감소를 보이고 있습니다. 이때가 최적일 것이라고 예상됩니다.
최적의 cache는 256set의 split cache입니다.

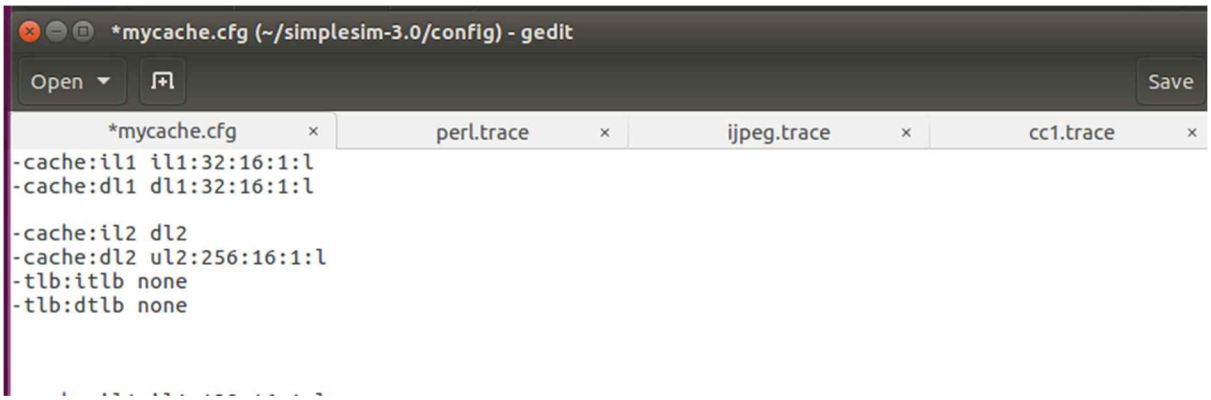
perl

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.3711	75.4952	0.4051	0.2391	72.8841
128	0.2903	59.3862	0.3521	0.1807	62.0269
256	0.2303	47.4392	0.2793	0.1388	49.2091
512	0.1816	37.7544	0.2134	0.1081	38.0096

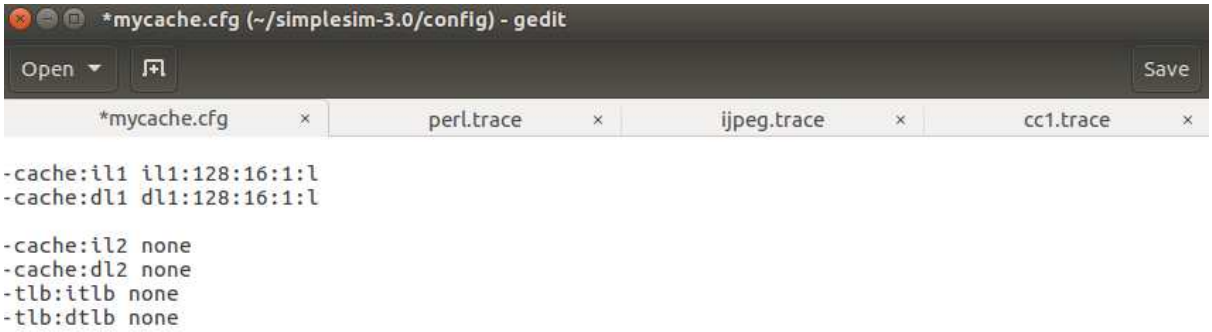
최적의 cache는 512set의 unified cache입니다.

Simulation 2. L1/L2 size

Block size = 16, Associativity = 1 고정



L2 cache가 none이 아닐 때 mycache.cfg파일을 설정한 화면입니다.



L2 cache가 none일 때 mycache.cfg파일을 설정한 화면입니다.
L1, L2의 값을 변화시키며 결과 값을 측정했습니다.

CC1

L1/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss Rate	AMAT
8/8/1024	0.4827	0.3761	0.2450	32.2358
16/16/512	0.4724	0.2954	0.3911	39.0694
32/32/256	0.3697	0.2248	0.5968	46.4223
64/64/128	0.3118	0.1654	0.8267	50.6750
128/128/0	0.2558	0.1154	1	44.4254

최적의 cache는 8/8/1024의 unified cache입니다.

ijpeg

L1/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss Rate	AMAT
8/8/1024	0.4912	0.4193	0.0310	13.4590
16/16/512	0.3733	0.3511	0.0729	13.4980
32/32/256	0.2586	0.2571	0.1261	12.3710
64/64/128	0.1089	0.2194	0.4078	14.3348
128/128/0	0.0212	0.1484	1	10.9529

최적의 cache는 32/32/256의 unified cache입니다.

perl

L1/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss Rate	AMAT
8/8/1024	0.4579	0.3748	0.3073	36.4291
16/16/512	0.4342	0.2963	0.4151	41.5493
32/32/256	0.4052	0.2390	0.5761	49.1390
64/64/128	0.3523	0.1806	0.8108	56.0159
128/128/0	0.2795	0.1387	1	49.2919

최적의 cache는 8/8/1024의 unified cache입니다.

Simulation 3. Associativity

Block size = 16 고정

Set의 값을 변화시키며 결과 값을 확인했습니다.

CC1

# of Sets	Split Cache Miss rate / AMAT					
	1-way			2-way		
	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.3697	0.2248	66.7822	0.2965	0.1320	51.0587
128	0.3118	0.1654	55.1577	0.2456	0.0881	41.3136
256	0.2558	0.1154	44.3371	0.1986	0.0564	32.8174
512	0.2131	0.0793	36.2120	0.1499	0.0362	24.7261
1024	0.1617	0.0524	27.3563	0.0990	0.0229	16.7088
2048	0.1172	0.0338	19.9613	0.0563	0.0140	10.1195

# of Sets	Split Cache Miss rate / AMAT	
	4-way	8way

	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.2403	0.0778	39.9517	0.1926	0.0456	31.3051
128	0.1949	0.0483	31.8085	0.1378	0.0298	22.5833
256	0.1408	0.0313	23.1205	0.0858	0.0192	14.5582
512	0.0901	0.0200	15.2437	0.0383	0.0122	7.3843
1024	0.0440	0.0126	8.2493	0.0145	0.0062	3.6742
2048	0.0182	0.0064	4.2415	0.0082	0.0021	2.5877

최적의 cache는 1024set의 8way split cache입니다.

ijpeg

# of Sets	**Split Cache Miss rate / AMAT**					
	1-way			2-way		
	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.2585	0.2571	52.6390	0.1172	0.1812	27.2490
128	0.1089	0.2194	27.6353	0.0170	0.1537	10.4179
256	0.0212	0.1484	10.8647	0.0054	0.0711	5.0463
512	0.0047	0.0791	5.3070	0.0015	0.0517	3.6350
1024	0.0035	0.0517	3.9703	0.0010	0.0126	1.8988
2048	0.0025	0.0157	2.2919	0.0003	0.0097	1.7106

# of Sets	**Split Cache Miss rate / AMAT**					
	4-way			8way		
	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.0173	0.0474	5.8121	0.0050	0.0251	2.9371
128	0.0055	0.0357	3.4981	0.0012	0.0154	1.9625
256	0.0014	0.0160	2.0415	0.0004	0.0113	1.7028
512	0.0004	0.0114	1.7297	0.0003	0.0090	1.6328

1024	0.0003	0.0091	1.6606	0.0003	0.0084	1.6544
2048	0.0003	0.0085	1.6831	0.0003	0.0081	1.6910

최적의 cache는 512set의 8way split cache입니다.

perl

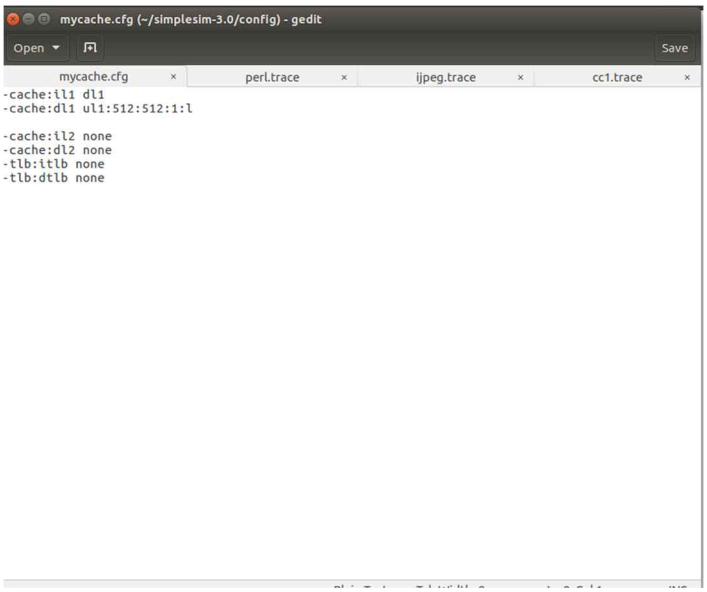
# of Sets	**Split Cache Miss rate / AMAT**					
	1-way			2-way		
	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.4051	0.2391	72.8841	0.3521	0.1507	60.3559
128	0.3522	0.1807	62.0269	0.2807	0.1108	47.8503
256	0.2794	0.1387	49.2091	0.2029	0.0820	35.0294
512	0.2134	0.1081	38.0096	0.1405	0.0593	24.7785
1024	0.1685	0.0716	29.5369	0.0871	0.0399	16.0156
2048	0.1418	0.0589	25.0142	0.0615	0.0332	15.6946

# of Sets	**Split Cache Miss rate / AMAT**					
	4-way			8way		
	Ist miss rate	Data miss rate	AMAT	Ist miss rate	Data miss rate	AMAT
64	0.2676	0.1039	45.5511	0.1978	0.0693	33.5492
128	0.1999	0.0728	34.0670	0.1251	0.0460	21.7704
256	0.1298	0.0513	22.7650	0.0670	0.0359	12.8362
512	0.0729	0.0376	13.8076	0.0343	0.0300	7.8171
1024	0.0372	0.0303	8.2774	0.0214	0.0288	5.9287
2048	0.0240	0.0289	6.3355	0.0184	0.0287	5.5380

최적의 cache는 2048set의 8way split cache입니다.

Simulation 4. Block size

Number of Sets = 512, Associativity = 1 고정



Simulation4를 진행하기 위한 mycache.cfg파일을 설정한 화면입니다. Block size의 값을 변화시키며 결과 값을 측정했습니다.

CC1

Block size	Unified cache Miss rate	AMAT

16	0.1667	34.6215
64	0.0410	9.5861
128	0.0206	5.5615
256	0.0118	3.8592
512	0.0064	2.8391

최적의 cache는 512block size의 unified cache입니다.

ijpeg

Block size	Unified cache Miss rate	AMAT
16	0.0322	7.6579
64	0.0093	3.1773
128	0.0014	1.6500
256	0.0008	1.5848
512	0.0006	1.6017

최적의 cache는 256block size의 unified cache입니다.

perl

Block size	Unified cache Miss rate	AMAT
16	0.1816	37.5952
64	0.0493	11.2392
128	0.0254	6.5144
256	0.0153	4.5518
512	0.0072	2.9914

최적의 cache는 512block size의 unified cache입니다.

■ 검증 전략, 분석 및 결과

● AMAT를 이용하여 적합한 Cache configuration 분석

먼저 위에서 AMAT를 이용하여 적합한 cache를 구했으니 이를 토대로 분석을 진행했습니다.

CC1과 perl의 block크기가 큰 것으로 봤을 때 연속적인 값을 읽는 경우가 많을 것으로 예상이 됩니다. 그리고 ijpeg의 경우에는 반복문이 상대적으로 적고 instruction의 수가 많을 것으로 예상이 되고 perl의 경우에는 associativity가 큰 것으로 보아 메모리의 주소 이동이 많을 것으로 예상됩니다. 그러므로 insertion sort와 random access는 각각 cc1과 perl의 최적의 cache를 사용할 경우 성능이 가장 좋을 것으로 예상이 됩니다.

■ 문제점 및 고찰

● 프로젝트 내용 전체 정리 및 고찰

먼저 이번 프로젝트는 컴퓨터 구조 강의 시간에 배운 내용인 cache에 대해서 직접 Set, associativity, Block size등을 조절해가며 각 benchmark에 따른 cache의 성능을 비교해 최적의 cache를 분석해보는 과제였습니다. Insertion sort와 Random access를 Simulate하여 Cache가 어떻게 작동하는지 분석하고 비교하였습니다. Waveform을 보면서 언제 Inst와 Data가 각각 Access되는지, Hit과 Miss되는 경우를 확인하였고 이해하는 데에는 오래 걸리지 않았으며, 직접 보고 확인할 수 있었기 때문에 비교적 쉽게 분석하고 비교할 수 있었습니다.

일단 우분투에서 주어진 파일들을 복사 후 .cfg파일을 수정하면서 각 benchmark의 성능을 분석하기 위해서 각각의 파일의 경로를 달리하여 실행했습니다. 실행 후 각 benchmark에 해당하는 tracefile에서 miss_rate를 중심으로 보았습니다. 그리고 이 miss_rate는 misses를 accesses로 나눈 값이라는 것을 알 수 있었습니다. 그리고 각 cache 단계에서의 access time과 hit time, miss_rate를 이용해 AMAT 또한 구할 수 있었습니다. 이때, 엑셀을 이용해 AMAT를 구하는 계산기를 만들어서 miss rate를 이용해 좀 더 편리하게 Set, associativity, Block size등에 따라 각 bechmark에 해당하는 AMAT를 구할 수 있었습니다. 이번 과제에서는 정해진 답이 없이 과제에서 주어진 프로그램을 돌려 miss rete를 분석하고 AMAT를 구하는 과제였기 때문에 이전 과제들에 비해서는 많이

수월하게 진행할 수 있었습니다.

unified cache

cc1	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT	inst miss rate*(inst hit tin
64	0.7185	1.0000	0.3411	200.0000	0.2815	2.0000	0.3411	200.0000	0.3411	69.5015	
128	0.7185	1.0400	0.2745	200.0000	0.2815	2.0800	0.2745	200.0000	0.2745	56.2328	
256	0.7185	1.0816	0.2203	200.0000	0.2815	2.1632	0.2203	200.0000	0.2203	45.4461	
512	0.7185	1.1249	0.1667	200.0000	0.2815	2.2497	0.1667	200.0000	0.1667	34.7815	

ijpeg	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT
64	0.7821	1.0000	0.1880	200.0000	0.2179	2.0000	0.1880	200.0000	0.1880	38.8179
128	0.7821	1.0400	0.1006	200.0000	0.2179	2.0800	0.1006	200.0000	0.1006	21.3866
256	0.7821	1.0816	0.0457	200.0000	0.2179	2.1632	0.0457	200.0000	0.0457	10.4573
512	0.7821	1.1249	0.0322	200.0000	0.2179	2.2497	0.0322	200.0000	0.0322	7.8100

perl	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT
64	0.7248	1.0000	0.3711	200.0000	0.2752	2.0000	0.3711	200.0000	0.3711	75.4952
128	0.7248	1.0400	0.2903	200.0000	0.2752	2.0800	0.2903	200.0000	0.2903	59.3862
256	0.7248	1.0816	0.2303	200.0000	0.2752	2.1632	0.2303	200.0000	0.2303	47.4392
512	0.7248	1.1249	0.1816	200.0000	0.2752	2.2497	0.1816	200.0000	0.1816	37.7544

split cache

cc1	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7185	1.0000	0.3697	200.0000	0.2815	1.0000	0.2248	200.0000	66.7822
128	0.7185	1.0400	0.3118	200.0000	0.2815	1.0400	0.1654	200.0000	55.1577
256	0.7185	1.0816	0.2558	200.0000	0.2815	1.0816	0.1154	200.0000	44.3371
512	0.7185	1.1249	0.2131	200.0000	0.2815	1.1249	0.0793	200.0000	36.2120

ijpeg	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7821	1.0000	0.2585	200.0000	0.2179	1.0000	0.2571	200.0000	52.6390
128	0.7821	1.0400	0.1089	200.0000	0.2179	1.0400	0.2194	200.0000	27.6353
256	0.7821	1.0816	0.0212	200.0000	0.2179	1.0816	0.1484	200.0000	10.8647
512	0.7821	1.1249	0.0047	200.0000	0.2179	1.1249	0.0791	200.0000	5.3070

perl	instr %	instr hit time	instr miss rate	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7248	1.0000	0.4051	200.0000	0.2752	1.0000	0.2391	200.0000	72.8841
128	0.7248	1.0400	0.3521	200.0000	0.2752	1.0400	0.1807	200.0000	62.0269
256	0.7248	1.0816	0.2793	200.0000	0.2752	1.0816	0.1388	200.0000	49.2091
512	0.7248	1.1249	0.2134	200.0000	0.2752	1.1249	0.1081	200.0000	38.0096

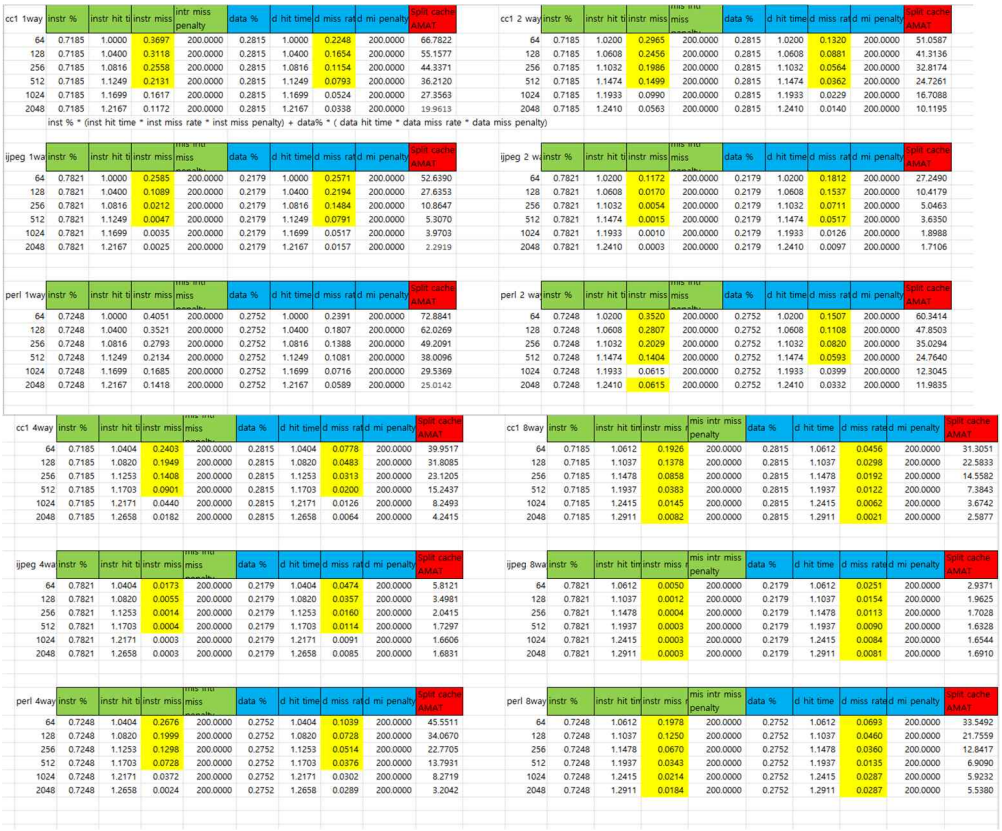
<simulation 1>

cc1	inst % L1	data % L1	Hit time L1	Data Miss rate L1	Inst Miss rate L1	Hit time L2	Miss rate L2	Miss panalty	AMAT
8/8/1024	0.7185	0.2815	1.0000	0.3761	0.4827	20.0000	0.2450	200.0000	32.2358
16/16/512	0.7185	0.2815	1.0400	0.2954	0.4274	19.2308	0.3911	200.0000	39.0694
32/32/256	0.7185	0.2815	1.0816	0.2248	0.3697	18.4911	0.5968	200.0000	46.4223
64/64/128	0.7185	0.2815	1.1249	0.1654	0.3118	17.7799	0.8267	200.0000	50.6750
128/128/0	0.7185	0.2815	1.1699	0.1154	0.2558	17.0961	1.0000	200.0000	44.4254

ijpeg	inst % L1	data % L1	Hit time L1	Data Miss rate L1	Inst Miss rate L1	Hit time L2	Miss rate L2	Miss panalty	AMAT
8/8/1024	0.7821	0.2179	1.0000	0.4193	0.4912	20.0000	0.0310	200.0000	13.4590
16/16/512	0.7821	0.2179	1.0400	0.3511	0.3733	19.2308	0.0729	200.0000	13.4980
32/32/256	0.7821	0.2179	1.0816	0.2571	0.2586	18.4911	0.1261	200.0000	12.3710
64/64/128	0.7821	0.2179	1.1249	0.2194	0.1089	17.7799	0.4078	200.0000	14.3348
128/128/0	0.7821	0.2179	1.1699	0.1484	0.0212	17.0961	1.0000	200.0000	10.9529

perl	inst % L1	data % L1	Hit time L1	Data Miss rate L1	Inst Miss rate L1	Hit time L2	Miss rate L2	Miss panalty	AMAT
8/8/1024	0.7248	0.2752	1.0000	0.3748	0.4579	20.0000	0.3072	200.0000	36.4291
16/16/512	0.7248	0.2752	1.0400	0.2963	0.4342	19.2308	0.4150	200.0000	41.5493
32/32/256	0.7248	0.2752	1.0816	0.2390	0.4052	18.4911	0.5760	200.0000	49.1390
64/64/128	0.7248	0.2752	1.1249	0.1806	0.3523	17.7799	0.8108	200.0000	56.0159
128/128/0	0.7248	0.2752	1.1699	0.1387	0.2793	17.0961	1.0000	200.0000	49.2919

<simulation 2>



cc1 1way	instr %	instr hit ti	instr miss	intr miss penalty	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7185	1.0000	0.3697	200.0000	0.2815	1.0000	0.2248	200.0000	66.7822
128	0.7185	1.0400	0.3118	200.0000	0.2815	1.0400	0.1654	200.0000	55.1577
256	0.7185	1.0816	0.2558	200.0000	0.2815	1.0816	0.1154	200.0000	44.3371
512	0.7185	1.1249	0.2131	200.0000	0.2815	1.1249	0.0793	200.0000	36.2120
1024	0.7185	1.1699	0.1617	200.0000	0.2815	1.1699	0.0524	200.0000	27.3563
2048	0.7185	1.2167	0.1172	200.0000	0.2815	1.2167	0.0338	200.0000	19.9613

inst % * (inst hit time * inst miss rate * inst miss penalty) + data% * (data hit time * data miss rate * data miss penalty)

cc1 2 way	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7185	1.0200	0.2965	200.0000	0.2815	1.0200	0.1320	200.0000	51.0587
128	0.7185	1.0608	0.2456	200.0000	0.2815	1.0608	0.0881	200.0000	41.3136
256	0.7185	1.1032	0.1986	200.0000	0.2815	1.1032	0.0564	200.0000	32.8174
512	0.7185	1.1474	0.1499	200.0000	0.2815	1.1474	0.0362	200.0000	24.7261
1024	0.7185	1.1933	0.0990	200.0000	0.2815	1.1933	0.0229	200.0000	16.7088
2048	0.7185	1.2410	0.0563	200.0000	0.2815	1.2410	0.0140	200.0000	10.1195

ijpeg 1wa	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7821	1.0000	0.2585	200.0000	0.2179	1.0000	0.2571	200.0000	52.6390
128	0.7821	1.0400	0.1089	200.0000	0.2179	1.0400	0.2194	200.0000	27.6353
256	0.7821	1.0816	0.0212	200.0000	0.2179	1.0816	0.1484	200.0000	10.8647
512	0.7821	1.1249	0.0047	200.0000	0.2179	1.1249	0.0791	200.0000	5.3070
1024	0.7821	1.1699	0.0035	200.0000	0.2179	1.1699	0.0517	200.0000	3.9703
2048	0.7821	1.2167	0.0025	200.0000	0.2179	1.2167	0.0157	200.0000	2.2919

ijpeg 2 wa	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7821	1.0200	0.1172	200.0000	0.2179	1.0200	0.1812	200.0000	27.2490
128	0.7821	1.0608	0.0170	200.0000	0.2179	1.0608	0.1537	200.0000	10.4179
256	0.7821	1.1032	0.0054	200.0000	0.2179	1.1032	0.0711	200.0000	5.0463
512	0.7821	1.1474	0.0015	200.0000	0.2179	1.1474	0.0517	200.0000	3.6350
1024	0.7821	1.1933	0.0010	200.0000	0.2179	1.1933	0.0126	200.0000	1.8988
2048	0.7821	1.2410	0.0003	200.0000	0.2179	1.2410	0.0097	200.0000	1.7106

perl 1way	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7248	1.0000	0.4051	200.0000	0.2752	1.0000	0.2391	200.0000	72.8841
128	0.7248	1.0400	0.3521	200.0000	0.2752	1.0400	0.1807	200.0000	62.0269
256	0.7248	1.0816	0.2793	200.0000	0.2752	1.0816	0.1388	200.0000	49.2091
512	0.7248	1.1249	0.2134	200.0000	0.2752	1.1249	0.1081	200.0000	38.0096
1024	0.7248	1.1699	0.1685	200.0000	0.2752	1.1699	0.0716	200.0000	29.5369
2048	0.7248	1.2167	0.1418	200.0000	0.2752	1.2167	0.0589	200.0000	25.0142

perl 2 wa	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7248	1.0200	0.3520	200.0000	0.2752	1.0200	0.1507	200.0000	60.3414
128	0.7248	1.0608	0.2807	200.0000	0.2752	1.0608	0.1108	200.0000	47.8503
256	0.7248	1.1032	0.2029	200.0000	0.2752	1.1032	0.0820	200.0000	35.0294
512	0.7248	1.1474	0.1404	200.0000	0.2752	1.1474	0.0593	200.0000	24.7640
1024	0.7248	1.1933	0.0615	200.0000	0.2752	1.1933	0.0399	200.0000	12.3045
2048	0.7248	1.2410	0.0615	200.0000	0.2752	1.2410	0.0332	200.0000	11.9835

cc1 4way	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7185	1.0404	0.2403	200.0000	0.2815	1.0404	0.0778	200.0000	39.9517
128	0.7185	1.0820	0.1949	200.0000	0.2815	1.0820	0.0483	200.0000	31.8085
256	0.7185	1.1253	0.1408	200.0000	0.2815	1.1253	0.0313	200.0000	23.1205
512	0.7185	1.1703	0.0901	200.0000	0.2815	1.1703	0.0200	200.0000	15.2437
1024	0.7185	1.2171	0.0440	200.0000	0.2815	1.2171	0.0126	200.0000	8.2493
2048	0.7185	1.2658	0.0182	200.0000	0.2815	1.2658	0.0064	200.0000	4.2415

cc1 8way	instr %	instr hit tin	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7185	1.0612	0.1926	200.0000	0.2815	1.0612	0.0456	200.0000	31.3051
128	0.7185	1.1037	0.1378	200.0000	0.2815	1.1037	0.0298	200.0000	22.5833
256	0.7185	1.1478	0.0858	200.0000	0.2815	1.1478	0.0192	200.0000	14.5582
512	0.7185	1.1937	0.0383	200.0000	0.2815	1.1937	0.0122	200.0000	7.3843
1024	0.7185	1.2415	0.0145	200.0000	0.2815	1.2415	0.0062	200.0000	3.6742
2048	0.7185	1.2911	0.0082	200.0000	0.2815	1.2911	0.0021	200.0000	2.5877

ijpeg 4wa	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7821	1.0404	0.0173	200.0000	0.2179	1.0404	0.0474	200.0000	5.8121
128	0.7821	1.0820	0.0055	200.0000	0.2179	1.0820	0.0357	200.0000	3.4981
256	0.7821	1.1253	0.0014	200.0000	0.2179	1.1253	0.0160	200.0000	2.0415
512	0.7821	1.1703	0.0004	200.0000	0.2179	1.1703	0.0114	200.0000	1.7297
1024	0.7821	1.2171	0.0003	200.0000	0.2179	1.2171	0.0091	200.0000	1.6606
2048	0.7821	1.2658	0.0003	200.0000	0.2179	1.2658	0.0085	200.0000	1.6831

ijpeg 8wa	instr %	instr hit tin	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7821	1.0612	0.0050	200.0000	0.2179	1.0612	0.0251	200.0000	2.9371
128	0.7821	1.1037	0.0012	200.0000	0.2179	1.1037	0.0154	200.0000	1.9625
256	0.7821	1.1478	0.0004	200.0000	0.2179	1.1478	0.0113	200.0000	1.7028
512	0.7821	1.1937	0.0003	200.0000	0.2179	1.1937	0.0090	200.0000	1.6328
1024	0.7821	1.2415	0.0003	200.0000	0.2179	1.2415	0.0084	200.0000	1.6544
2048	0.7821	1.2911	0.0003	200.0000	0.2179	1.2911	0.0081	200.0000	1.6910

perl 4way	instr %	instr hit ti	instr miss	mis intr miss	data %	d hit time	d miss rat	d mi penalty	Split cache AMAT
64	0.7248	1.0404	0.2676	200.0000	0.2752	1.0404	0.1039	200.0000	45.5511
128	0.7248	1.0820	0.1999	200.0000	0.2752	1.0820	0.0728	200.0000	34.0670
256	0.7248	1.1253	0.1298	200.0000	0.2752	1.1253	0.0514	200.0000	22.7705
512	0.7248	1.1703	0.0728	200.0000	0.2752	1.1703	0.0376	200.0000	13.7931
1024	0.7248	1.2171	0.0372	200.0000	0.2752	1.2171	0.0302	200.0000	8.2719
2048	0.7248	1.2658	0.0024	200.0000	0.2752	1.2658	0.0289	200.0000	3.2042

perl 8way	instr %	instr hit tin	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Split cache AMAT
64	0.7248	1.0612	0.1978	200.0000	0.2752	1.0612	0.0693	200.0000	33.5492
128	0.7248	1.1037	0.1250	200.0000	0.2752	1.1037	0.0460	200.0000	21.7559
256	0.7248	1.1478	0.0670	200.0000	0.2752	1.1478	0.0360	200.0000	12.8417
512	0.7248	1.1937	0.0343	200.0000	0.2752	1.1937	0.0135	200.0000	6.9090
1024	0.7248	1.2415	0.0214	200.0000	0.2752	1.2415	0.0287	200.0000	5.9232
2048	0.7248	1.2911	0.0184	200.0000	0.2752	1.2911	0.0287	200.0000	5.5380

<simulation 3>

cc1	instr %	instr hit time	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT
16	0.7185	1.0000	0.1667	200.0000	0.2815	2.0000	0.1667	200.0000	0.1667	34.6215
64	0.7185	1.0816	0.0410	200.0000	0.2815	2.1632	0.0410	200.0000	0.0410	9.5861
128	0.7185	1.1249	0.0206	200.0000	0.2815	2.2497	0.0206	200.0000	0.0206	5.5615
256	0.7185	1.1699	0.0118	200.0000	0.2815	2.3397	0.0118	200.0000	0.0118	3.8592
512	0.7185	1.2167	0.0064	200.0000	0.2815	2.4333	0.0064	200.0000	0.0064	2.8391

ijpeg	instr %	instr hit time	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT
16	0.7821	1.0000	0.0322	200.0000	0.2179	2.0000	0.0322	200.0000	0.0322	7.6579
64	0.7821	1.0816	0.0093	200.0000	0.2179	2.1632	0.0093	200.0000	0.0093	3.1773
128	0.7821	1.1249	0.0014	200.0000	0.2179	2.2497	0.0014	200.0000	0.0014	1.6500
256	0.7821	1.1699	0.0008	200.0000	0.2179	2.3397	0.0008	200.0000	0.0008	1.5848
512	0.7821	1.2167	0.0006	200.0000	0.2179	2.4333	0.0006	200.0000	0.0006	1.6017

perl	instr %	instr hit time	instr miss	mis intr miss penalty	data %	d hit time	d miss rate	d mi penalty	Unified cache Miss rate	Unified cache AMAT
16	0.7248	1.0000	0.1816	200.0000	0.2752	2.0000	0.1816	200.0000	0.1816	37.5952
64	0.7248	1.0816	0.0493	200.0000	0.2752	2.1632	0.0493	200.0000	0.0493	11.2392
128	0.7248	1.1249	0.0254	200.0000	0.2752	2.2497	0.0254	200.0000	0.0254	6.5144
256	0.7248	1.1699	0.0153	200.0000	0.2752	2.3397	0.0153	200.0000	0.0153	4.5518
512	0.7248	1.2167	0.0072	200.0000	0.2752	2.4333	0.0072	200.0000	0.0072	2.9914

<simulation 4>

위 사진에서 볼 수 있듯이 엑셀을 통해 AMAT계산기를 만든 후 AMAT을 쉽게 구할 수 있었습니다.

- **Benchmarks에 최적화된 cache와 정렬 프로그램에 최적화된 cache 비교 및 분석**

  위의 분석 결과를 통해 Set, associativity, Block size가 클수록 성능이 좋다는 점을 알 수 있습니다.
  그리고 L2 cache가 존재하고 L2의 크기가 클수록 성능이 좋아진다는 점 또한 확인할 수 있습니다. 하지만, 크기가 커질수록 성능이 상승하는 비율이 작아지는 점을 확인할 수 있습니다.
  CC1 프로그램의 경우 컴파일러이기 때문에 instruction cache가 빨라야지 성능이 잘 나오고, ijpeg는 이미지 압축 관련 벤치마크이기 때문에 픽셀 값을 2d matrix로 계산을 진행합니다. 그러므로 cc1과는 반대로 data cache가 좋아야 성능이 잘 나옵니다. Perl의 경우에는 instruction cache와 data cache가 모두 좋아야 성능이 잘 나옵니다.
  이렇게 눈에 보이는 명확히 변화들에 대해서는 분석이 쉬운 편이었고, 쉽게 이해할 수 있었습니다.
  하지만 number of sets, Associativity, Level2의 여부, Block size, 그리고 cost와 같이 많은 기준이 있기 때문에 다 종합했을 때 어떤 것이 가장 적합한지 찾는 것이 어려웠습니다. 주관적으로 세운 기준에 따라 고른 것이기 때문에 사람들에 따라 모두 다른 결과가 나올 수 있을 것 같다는 생각이 들었습니다.
  CC1과 perl의 block크기가 큰 것으로 봤을 때 연속적인 값을 읽는 경우가 많을 것으로 예상이 됩니다. 그리고 ijpeg의 경우에는 반복문이 상대적으로 적고 instruction의 수가 많을 것으로 예상이 되고 perl의 경우에는 associativity가 큰 것으로 보아 메모리의 주소 이동이 많을 것으로 예상됩니다. 그러므로 insertion sort와 random access는 각각 cc1과 perl의 최적의 cache를 사용할 경우 성능이 가장 좋을 것으로 예상이 됩니다.
  참고로 저는 정말로 최고의 성능만을 따지고 싶어서 cost에 비해서 효율이 안 나오더라도 성능이 가장 좋은 cache를 선택했습니다.