CSCI 6461:

Computer Systems and Architecture

Instructor: Dr. Lei He

Project I

Team Members:

1. Nahom Melaku *(G30588499)*
2. Shubham Jadhav *(G30570862)*
3. Amish Raj *(G48495262)*

Table of Contents

1. Locality of Different Programs

Objective: Show that the programs have different locality, and there are programs with “good” or “bad” locality.

General Setup:

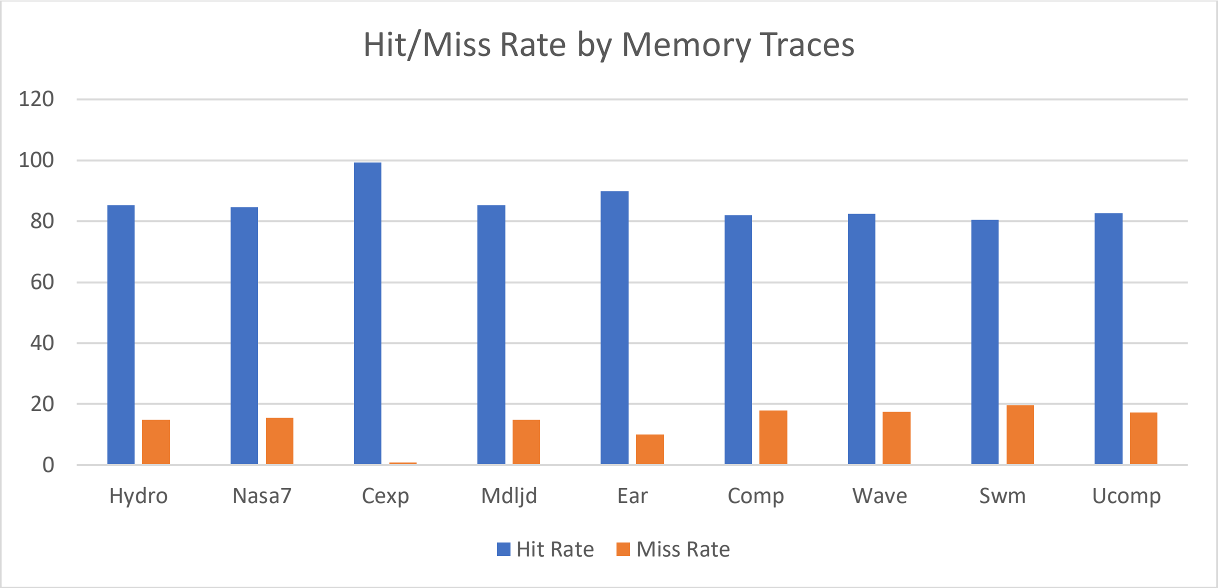
- Processors in SMP=1
- Cache Coherence Protocol=MESI
- Scheme for bus arbitration= Random
- Word wide (bits)=16
- Words by block = 16 (block size = 32 bytes)
- Blocks in main memory = 8192 (main memory size = 256 KB)
- Blocks in cache = 128 (cache size = 4KB)
- Mapping = Fully Associative
- Replacement Policy = LRU

Variable Setup:

Obtain the miss rate using the following traces: Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm and UComp.

Result:

Trace	Hit Rate (%)	Miss Rate (%)
Hydro	85.19	14.810
Nasa7	84.636	15.364
Cexp	99.26	0.740
Mdljd	85.22	14.780
Ear	89.969	10.098
Comp	82.092	17.908
Wave	82.521	17.479
Swm	80.37	19.630
Ucomp	82.73	17.270



Conclusion:

1. It can be seen from the experimental results above that **all the programs have different locality grades**. As per data collected, even the programs using the same amount of instruction have a different hit/miss ratio, block transfers, and total replacement.
2. The program with the best locality is **CEXP** as it has the least Miss Rate of 0.740% or the highest Hit Rate of 99.26%.
3. The program with the worst locality is **SWM** as it has the highest Miss Rate of 19.630% or the lowest Hit Rate of 80.37%.
4. Yes, the design of memory systems that exploit the locality of specific programs can improve system performance because **exploiting temporal locality helps keep data closer to the processor and exploiting spatial locality helps move important data that is needed next to the upper level**. Memory systems also take advantage of locality by having larger block sizes and cache.
5. The miss rate decreases as the **cache begins to carry the most frequently accessed requests by the processor**. Furthermore, it decreases as the bus transaction and block transfer increase, optimizing the most relevant data that the processor requests.

In conclusion, through this experiment we were able to observe the locality of different programs and we were able to show that programs have different locality. CEXP is a program having good locality and SWM is an example of a program with bad locality. We also discussed the impact of exploiting locality on the system performance as well as analyzed the behavior of decreasing miss rate with the forward execution of a program.

2. Influence of Different Cache Sizes

Objective: Analyze the influence of various cache sizes on the miss rate for different memory traces.

General Setup:

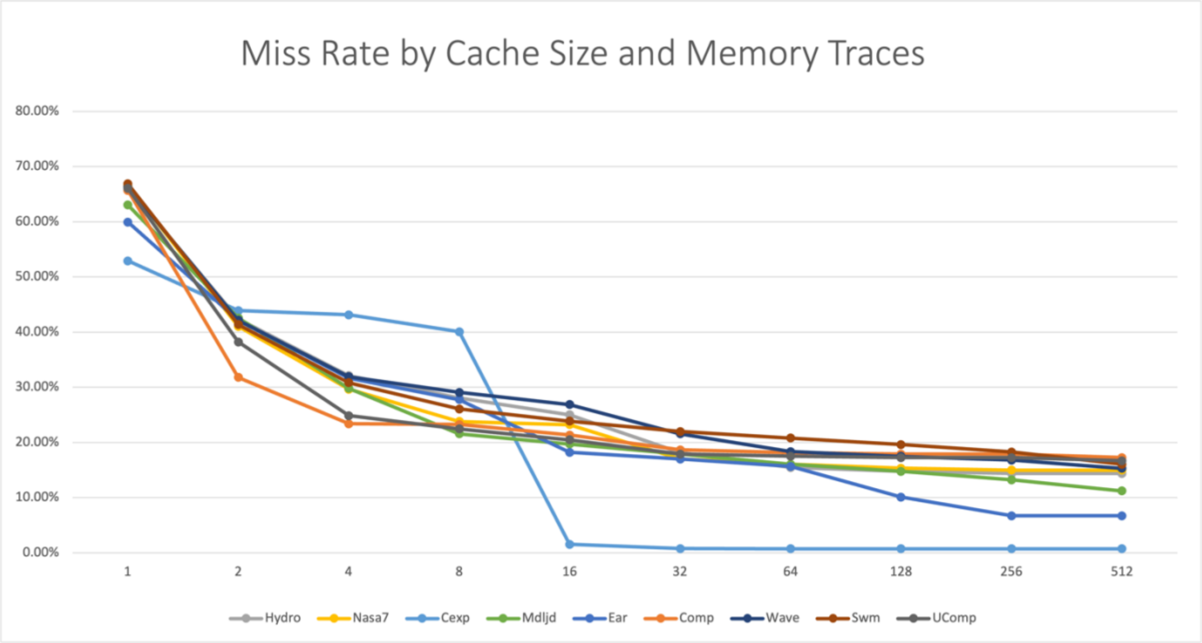
- Processors in SMP = 1
- Cache Coherence Protocol = MESI
- Scheme for bus arbitration = Random
- Word wide (bits) = 16
- Words by block = 16
- Blocks in main memory = 8192
- Mapping = Fully Associative
- Replacement policy = LRU

Variable Setup:

- For each setup, configure the number of blocks in cache to 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512
- Obtain the miss rate using the following traces: Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm and UComp

Results:

Trace	Miss Rate (%) for Influence of Block in cache of:									
	1	2	4	8	16	32	64	128	256	512
Hydro	66.20	42.36	32.11	28.06	25.01	18.19	15.46	14.81	14.38	14.38
Nasa7	66.58	41.02	29.65	23.82	23.24	17.04	16.07	15.36	14.99	14.99
Cexp	52.92	43.90	43.15	40.07	1.54	0.77	0.74	0.74	0.74	0.74
Mdljd	63.06	42.57	29.82	21.58	19.72	17.89	16.04	14.78	13.24	11.23
Ear	59.95	42.14	31.62	27.77	18.22	16.99	15.67	10.10	6.71	6.71
Comp	65.69	31.80	23.40	23.26	21.36	18.66	18.19	17.91	17.91	17.27
Wave	66.36	42.14	31.92	29.09	26.88	21.56	18.35	17.48	16.81	15.32
Swm	66.87	41.38	30.84	26.09	23.88	21.99	20.82	19.63	18.29	16.06
UComp	66.05	38.24	24.88	22.47	20.49	17.89	17.53	17.27	17.27	16.69



Conclusion:

1. The **miss rate decreases** as the cache size decreases and this is because **capacity misses decrease as cache size** is increased and this reduces miss rate
2. The decrease in miss rates happened **for all the benchmarks** and this indicates that all programs have the same general behavior **independently of different locality grades**
3. When we increase the size of the cache, **capacity and conflict (collision) misses are reduced.**
4. There is a conflict miss when we need to put a block in an **already occupied cache space**. This might happen when our **cache is full**. Therefore, **we can say that there are conflict misses in our experiment**
5. As we keep increasing the cache size, **the rate of conflict misses reduces**. Although the benefits are always **less noticeable with larger caches**, the miss rate does improve as the associativity grade rises. This result is consistent with the theory because increasing associativity **reduces conflict misses**; nevertheless, **in big caches, this type of miss is less common** and **after a point we can see stabilization** in the miss rates even if cache size keeps on increasing
6. We can see **substantial differences for all the benchmarks in miss rates** when the block size changes from **1 to 2, and also from 2 to 4** where there is a **drastic reduction in miss rates**. As initially we have only one block, for every access to memory, there is a **high probability of misses**. And, when we increase the cache size (which is from the increased number of blocks from 1 to 2 and from 2 to 4), the **probability of misses reduces at a higher degree** as compared to the change from 256 to 512

In conclusion, the increase of cache size **improves the performance** of our system, and this is more evident when we increase the **size of smaller caches than larger caches** for a fully associative mapping.

3. Influence of Block Size

Objective: Study the influence of block size on the miss rate

General setup:

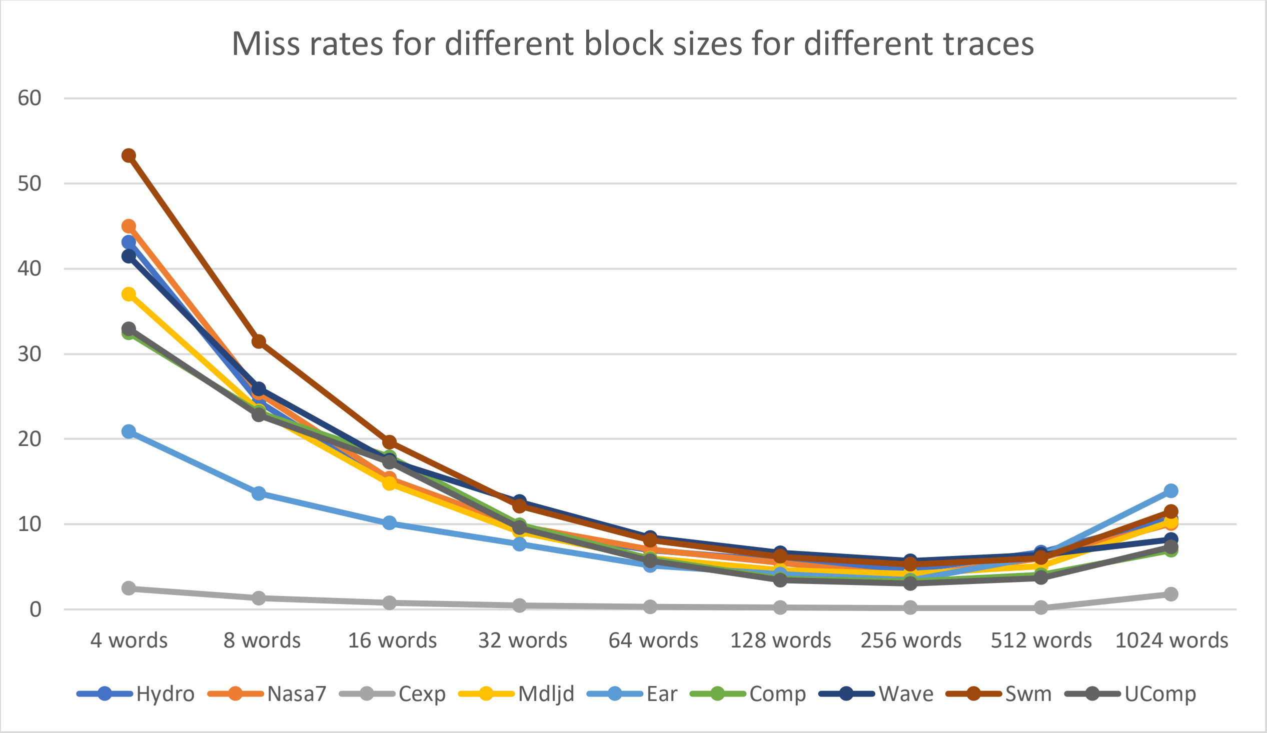
- Processor in SMP = 1
- Cache coherence protocol = MESI
- Scheme for bus arbitration = Random
- Word wide (bits) = 16
- Main memory size = 256KB (the number of blocks in main memory varies)
- Cache size = 4KB (the number of blocks in main memory varies)
- Mapping = Fully Associative
- Replacement policy = LRU

Variable setup:

- Word by block = 4 (block size = 8bytes), 8, 16, 32, 64, 128, 256, 512, 1024 (block size = 2048bytes)
- For each word by block, use the following traces: Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm and UComp

Results:

Trace	Miss Rate (%) for words by block of:								
	4	8	16	32	64	128	256	512	1028
Hydro	43.11	24.31	14.81	9.40	6.96	5.69	4.61	6.72	10.63
Nasa7	44.96	25.39	15.36	9.76	7.06	5.50	3.88	5.98	10.08
Cexp	2.44	1.31	0.74	0.46	0.30	0.22	0.15	0.19	1.78
Mdljd	37.01	23.29	14.78	9.08	6.01	4.60	4.18	5.07	10.31
Ear	20.84	13.59	10.10	7.65	5.12	4.11	3.43	6.27	13.92
Comp	32.45	23.10	17.91	9.91	5.90	3.57	3.29	4.04	6.93
Wave	41.44	25.88	17.48	12.61	8.43	6.62	5.69	6.45	8.20
Swm	53.25	31.41	19.63	12.12	8.13	6.22	5.27	6.05	11.50
UComp	32.93	22.80	17.27	9.55	5.67	3.40	3.04	3.70	7.32



Conclusion:

1. The miss rate decreases as the block size increases because of the principle of locality.
   - **Temporal locality:** if something is accessed once, it'll probably be accessed again soon
   - **Spatial locality:** if something is accessed, something nearby it will probably be accessed

   And larger block size helps with spatial locality.
2. The increment happens for all the benchmarks.
3. **Compulsory misses decrease as we increase the block size** because increasing the block size allows more data to be fetched into the Cache. This improves the chance of a new memory location request already begin present in the Cache.
4. When we increase the block size to **512**, the miss rate **starts increasing** and this is because of the conflict misses that arise because of our fixed cache size. **This is our pollution point.**

In conclusion, the increase of block size improves the system performance. We have to be cautious about this point. High latency and high bandwidth encourage large block size. This means, if it takes a long time to process miss, bring in as much information as you can when you do miss. Low latency and low bandwidth encourage small block size. This means, if it doesn't take long to process miss, but the bandwidth is small, use small block sizes.

4. Influence of Block Size for Different Cache Sizes

Objective: Study the influence of block size on the miss rate for different cache sizes

General setup:

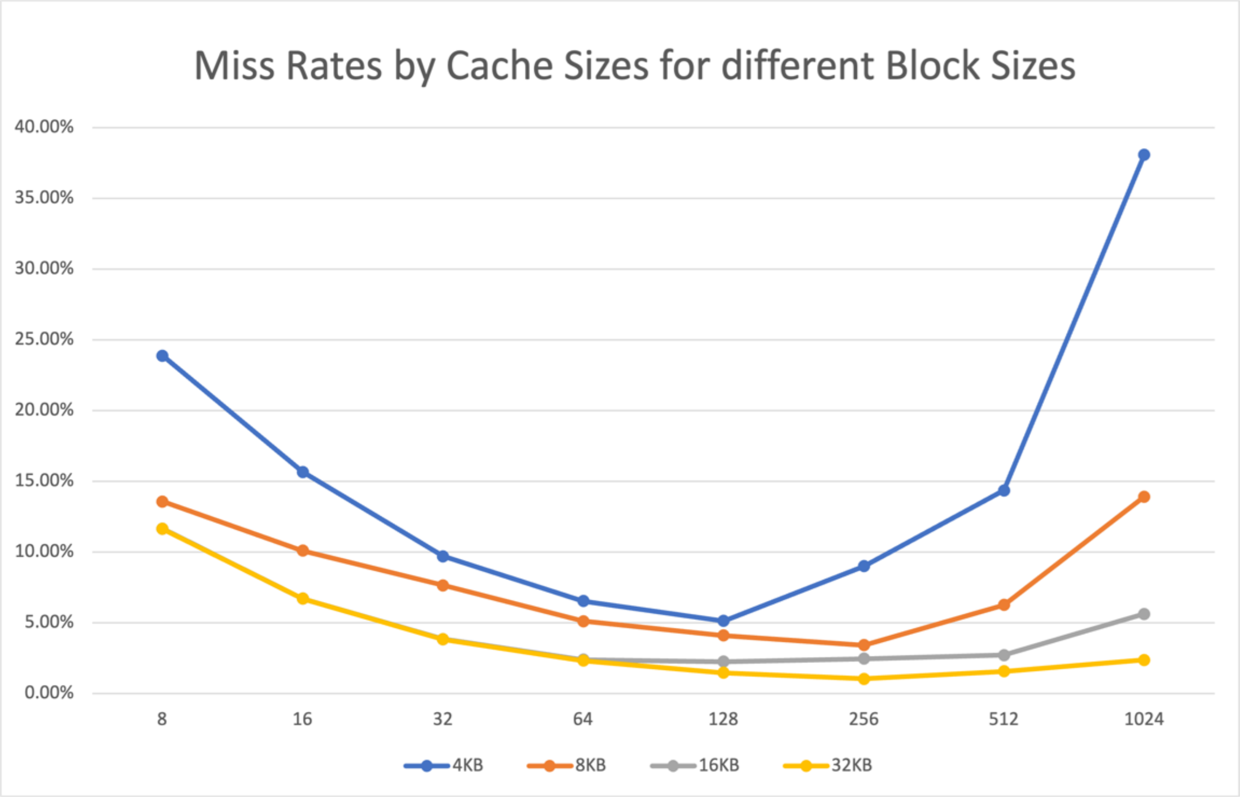
- Processor in SMP = 1
- Cache coherence protocol = MESI
- Scheme for bus arbitration = Random
- Word wide (bits) = 32
- Main memory size = 1024KB (the number of blocks in main memory varies)
- Mapping = Fully Associative
- Replacement policy = LRU
- For each word by block, use the following the trace, Ear.

Variable setup:

- Word by block = 8 (block size = 32 bytes), 16, 32, 64, 128, 256, 512, 1024 (block size = 4096bytes)
- Cache size = 4KB, 8KB, 16KB, 32KB

Results:

Cache Size	Miss Rates (%) For Words Per Block of:							
	8	16	32	64	128	256	512	1024
4KB	23.89	15.67	9.70	6.54	5.14	9.01	14.38	38.09
8KB	13.59	10.10	7.65	5.12	4.11	3.43	6.27	13.92
16KB	11.66	6.71	3.86	2.39	2.26	2.47	2.73	5.63
32KB	11.64	6.71	3.84	2.34	1.47	1.06	1.58	2.37



Conclusion:

1. The miss rate decreases as the block size increases because of the principle of locality
    a. **Temporal locality:** if something is accessed once, it'll probably be accessed again soon
    b. **Spxatial locality:** if something is accessed, something nearby it will probably be accessed
    c. **And larger block size helps with spatial locality**
2. The increment happens for the benchmark we used which is Ear
3. **Compulsory misses decrease as we increase the block size** because increasing the block size allows more data to be fetched into the Cache
4. This improves the chance of a new memory location request already begin present in the Cache
5. When we increase the block size to **512**, the miss rate **starts increasing** and this is because of the conflict misses that arise. **This is our pollution point**

In conclusion, the influence of the pollution point decreases as the cache size increases and this is because if we have larger space in cache, the chance of conflicts happening decreases, and this reduces the conflict misses which will affect the miss rate

5. Influence of the Mapping for Different Cache Sizes

Objective: Analyze the influence of the mapping on the miss rate for several cache sizes.

General Setup:

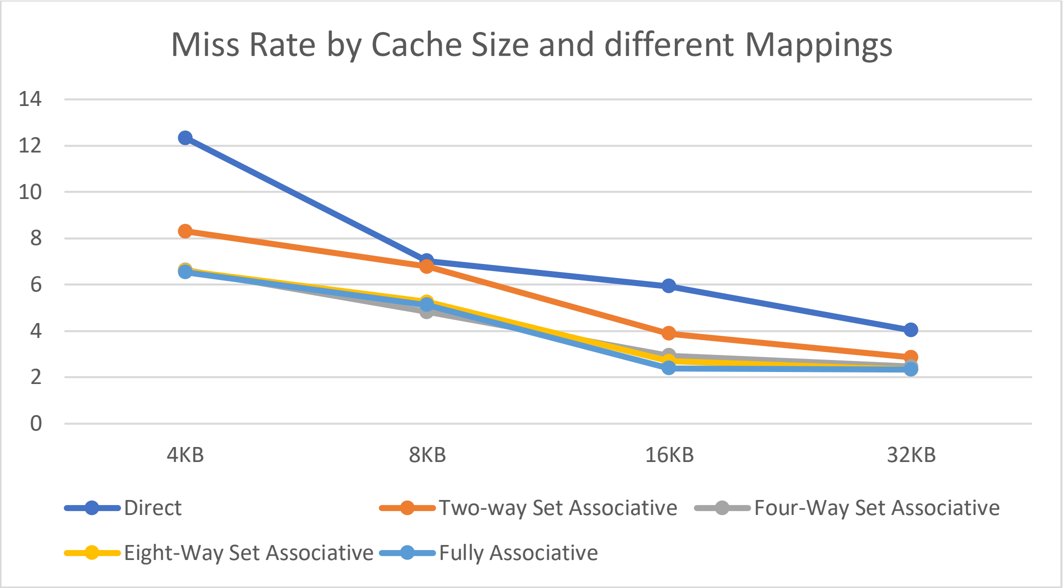
- Processors in SMP = 1
- Cache Coherence Protocol = MESI
- Scheme for bus arbitration = Random
- Word wide (bits) = 32.
- Blocks in main memory = 4096
- Replacement policy = LRU

Variable Setup:

- Words by block = 64 (Block Size = 256 bytes)
- For each configuration of mappings, configure the number of blocks in cache to get cache sizes of 4KB, 8KB, 16KB, 32KB
- For each trace obtain the miss rate using the memory trace Ear

Results:

Mapping	Miss Rate (%) by Cache Size of:			
	4KB	8KB	16KB	32KB
Direct	12.34	7.03	5.93	4.03
Two-way set associative	8.31	6.78	3.88	2.86
Four-way set associative	6.63	4.82	2.94	2.47
Eight-way set associative	6.59	5.26	2.69	2.34
Fully Associative	6.54	5.12	2.39	2.34



Conclusion:

As we can observe from the results above, the miss rate decreases as associativity increases. This is because, if the cache is direct mapped (one-way), then a block in memory is direct mapped to a set which can only contain one block. If another block that is mapped to that set is needed by the CPU, the previous block is evicted. Thus, the next time we need this evicted block, it will be a miss. If the cache is two-way mapped, then one set can contain two blocks. This means, in our previous example, both blocks in the set can be kept and there is not going to be a miss. If the associativity increases, the chance of missing a block decreases.

When there is more data in the cache, the associativity grade has a stronger influence than when there is less data. This is because a bigger cache holds a higher number of possible cache lines, each of which can retain the required information. This shows that the associativity grade has a bigger influence on the number of incorrectly answered questions. Finally, it is conceivable to claim that increasing the system's associativity improves its performance.

The most advantages may be seen in one-way to two-way communication. In a direct-mapped cache that can hold 16 lines of data in their entirety, if the cache is completely associative, there are 16 separate cache lines with the ability to contain the needed data. If the cache is not completely associative, just one cache line may retain the necessary data. Consequently, the cache must search over a larger number of lines to find the data that is being requested, resulting in a higher miss rate. The frequency of undiscovered conflicts, on the other hand, is reduced since it is less likely that two unique bits of data would be mapped to the same cache line. Therefore, fewer confrontations go unreported.

In conclusion, we were able to analyze the influence of mapping on the miss rate based on several cache sizes and explore how associativity impacts system performance and cache accesses.

6. Influence of Different Replacement Policy

Objective: Analyze the influence of various replacement policy on the miss rate for different memory traces.

General Setup:

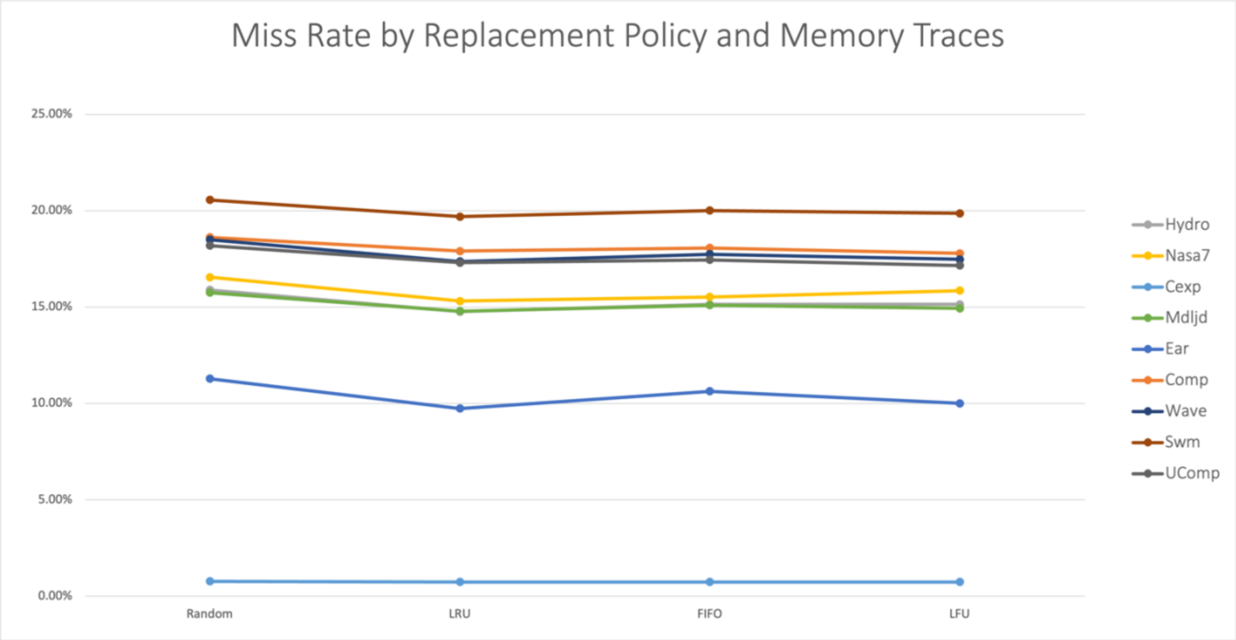
- Processors in SMP = 1
- Cache Coherence Protocol = MESI
- Scheme for bus arbitration = Random
- Word wide (bits) = 16
- Words by block = 16
- Blocks in main memory = 8192
- Blocks in Cache = 128
- Mapping = 8-way set associative

Variable Setup:

- For each setup, configure the replacement policy using Random, LRU, LFU, and FIFO
- Obtain the miss rate using the following traces: Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm and UComp

Results:

Trace	Influence of the Replacement policy on miss rate			
	Random	LRU	FIFO	LFU
Hydro	15.89%	14.76%	15.14%	15.14%
Nasa7	16.55%	15.31%	15.53%	15.85%
Cexp	0.78%	0.74%	0.74%	0.74%
Mdljd	15.75%	14.79%	15.10%	14.93%
Ear	11.29%	9.74%	10.63%	10.00%
Comp	18.62%	17.91%	18.07%	17.79%
Wave	18.50%	17.36%	17.74%	17.48%
Swm	20.56%	19.69%	20.01%	19.86%
UComp	18.19%	17.31%	17.45%	17.16%



Conclusion:

1. **LRU is the best** replacement policy and **Random is the worst** replacement policy
2. The results we got from **LFU, and FIFO were not that much different with LRU**, and we were able to notice that the benefits of those **do not depend on different locality grades**
3. We **do not expect** the results for different replacement policies to be different because in a direct-mapped cache, we have only one block in a set and that is the only one to be replaced. All the policies **would work the same in this case** and remove the same block
4. Using a concrete replacement policy **improves** the system performance. The step with **more benefits in terms of performance** is from **Random to LRU**. As the principle of temporal locality suggests, the best choice is to evict the least recently used block, because it is least likely to be used again soon
5. **Random** is the **cheapest** choice and **LRU** is the most **expensive**. Choosing the cheapest is a trade off in cache performance, as Random policy may have high number of misses as compared to LRU and this tradeoff can be worthy if efficiency is of importance