Lab 9

Lab 9

Task 1:

1. Find out the number of bits in block offset, cache index, and tag for the default cache configuration.
2. Address, Cache index, Tag, Block offset, Hit/Miss

Part 1 Table					
Attributes	Address	Cache index	Tag	Block Offset	Hit/Miss
1	0x10010000	0	0x00200200	0	Miss
2	0x10010004	0	0x00200200	0	Hit
3	0x10010008	0	0x00200200	0	Hit
4	0x1001000c	0	0x00200200	0	Hit
5	0x10010010	0	0x00200200	1	Miss
6	0x10010014	0	0x00200200	1	Hit
7	0x10010018	0	0x00200200	1	Hit
8	0x1001001c	0	0x00200200	1	Hit
9	0x10010020	0	0x00200200	2	Miss
10	0x10010024	0	0x00200200	2	Hit

The pattern is miss Hit Hit Hit, miss Hit Hit Hit,... until the cache is filled. This means I predict a hit rate of 75%, and this is corroborated by the data, which had a hit rate of 7/10.

1. Pattern = #bitsInIndex = log_2(#blocks)
2. #bits in offset = log_2(BlockSize(bytes))
3. #bits in tag = 32-b-l

The Report Items:

- 32 bits per field because this is the size of the location in memory, and subsequently the size of riskv commands. We are on a 32 bit machine.
- The cache hit rate was 75%, which makes sense becasue every iteration the cache would follow the pattern of missing the first, then hitting three. This happened every itteration and not only the first because the cache was not able to store the entire array, and the array was being looped through each iteration. When the cache was larger than the array, the hit rate will be 100%, as each item could be stored, meaning there were only misses on the first itteration of the array loop.
- The cache rate observed was 75% and 100% respectively, because of the aforementioned reasons.

Task 2:

Part 2 Table				
Attributes	Number of Block	Cache Size	Hit Rate	Miss Count
1	8	128 Bytes	75%	2560
2	16	256 Bytes	75%	2560
3	32	512 Bytes	75%	2560
4	64	1024 Bytes	99%	64
5	128	2048 Bytes	99%	64

Pattern: Miss, Hit, Hit, Hit - Until the cache is full, then full hits

- Cache hit rates predicted: 75%
- Explaination for cache hit rate: Since the size is 4, and the first search is always a miss, 1/4 times the search is a miss, giving it a hit rate of 75%. Once the cache size is larger than the array size, then there are only misses on the first run through, meaning the hit rate is near 100% after it misses the first run through 64 locations in memory.
- Explaination for miss count: The miss count is 2560 because when the array size is below the cache size, there is a miss every 4th search, whereas when the cache size is greater than the array size, the cache can be filled and then this mitigates any misses, meaning only 64 misses on the first run through are encountered.
- Method for predicting hit rates and miss counts: You know the hit rate will be constant if the cache size is lower than the size of the array, as it will have to reload items of the array slowing down the effectiveness of the cache, however if the size of the cache is greater than the size of the array, it can store all of the items for easy access and get full hits after the first itteration.

Task 3:

Part 3 Table (Miss, Hit, Hit, Hit, Hit, Hit, Hit, Hit)				
Attributes	Number of Block	Cache Size	Hit Rate	Miss Count
1	8	256 Bytes	88%	1280
2	16	512 Bytes	88%	1280
3	32	1024 Bytes	100%	32
4	64	2048 Bytes	100%	32
5	128	4096 Bytes	100%	32

Pattern: Miss, Hit, Hit, Hit, Hit, Hit, Hit, Hit- Until the cache is full, then full hits

- Cache hit rates predicted: 7/8 or 88%
- Explaination for cache hit rate: Since the size is now 8 instead of 4, and the first search is always a miss, now 7/8 times there is a hit, giving a percentage of 88%. This also changes the default cache sizes, doubling them becasue the size of each word doubled.

- Explaination for miss count: The miss count is dependant on the cache size vs the array size. If the cache size is larger than the array size, it will be capped at 32, as once it has itterated once it will go over previously stored memory. If the cache size is smaller than the array size, it will be capped out at 1280.
- Method for predicting hit rates and miss counts: Test to see if the cache size is equal to or greater than the array size.