HPC Hardware Assignment 3

Michael Tierney 11348436

November 2015

Task 1

In a 4 byte address there are 32 bits. This address is divided into sections in order to figure out its placement in cache. The offset determines where the data is in the cache line, the set determines which set the data belongs to, the tag is used to check if the address is already in the cache. The example cache used in this assignment has 128 bytes and consists of cache lines of size 16 bytes. The number of sets of the cache is determined using the following equation.

$$N_S = \frac{S_c}{S_l A}$$

Where N_s is the number of sets, S_c is the size of the cache, S_l is the size of a cache line and A is the associativity. For a direct mapped cache $A = 1$ leading to the following expression for our example cache.

$$N_s = 8 = \frac{128}{16}$$

In order to determine the set the minimum number of bits needed to represent all the set indices is taken from the address to the left of the bits used to determine the offset. The number of bits needed for the offset for our cache is equal to the number of bits required to represent the cache line size - 1. In this case it is (1111) which is 4 bits. Since we have 8 sets the minimum amount of bits required to represent the indices is (111) which is 3 bits. These leaves 9 bits for the tag. The breakdown of the address for this system is shown in the following diagram.

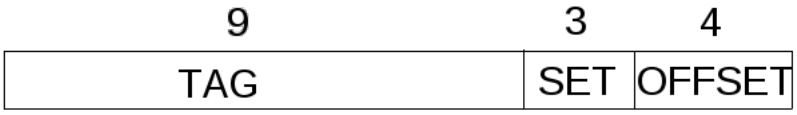

Figure 1: Diagram showing layout of bits for an address going to a cache with 8 sets and cache lines of size 16 bytes.

A piece of data is in the cache if its set and tag match another entry in the cache. This is referred to as a hit. If it isn't there the data has to be fetched from main memory. This new data replaces the least recently used cache line in the set. The second part of this assignment involved implementing the above system in c. In order to extract the relevant bits one has to convert the hexadecimal addresses to base ten and then use bitwise operators. In order to use these operators the number of bits needed to express the set,offset and tag must be determined. These are found by shifting the bits to the right until the number equals zero, the number of shifts is the number of bits. In order to determine the set one has to shift the bits in the address to the right by the number of bits used for the offset (2^{ϕ}). Once the bits are in place they can by logical & with the number $2^s - 1$ where s is the number of bits required to represent the set. This results in an index which represents the set the address belongs to. A diagram showing this process is shown in 2. To extract the tags one just has to logically & the address with the number $\sum_{i=1}^{16-s-\phi} 2^{16-i}$ where ϕ is the number of bits for the offset and s is the number of bits for the set. This tag is then compared with the tags already in cache. If the tags aren't matched then a cacheline with the lowest timestamp in the set is found and replaced with the new address.

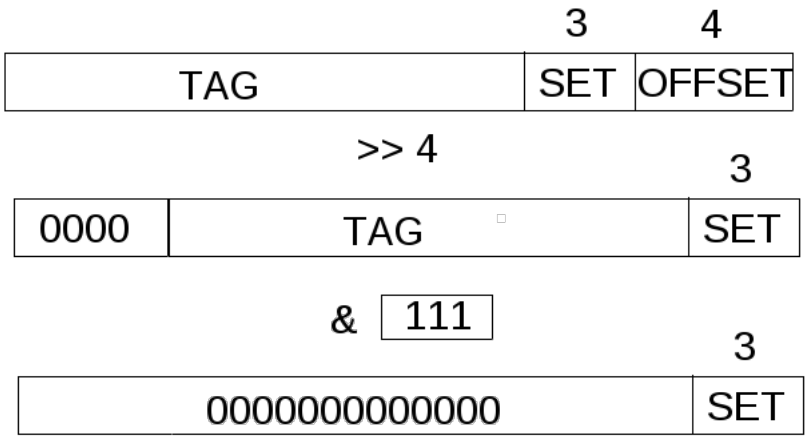

Figure 2: The process for extracting the set number.

The hits and misses for the addresses supplied for a direct mapped, 2-way, 4-way and fully associative cache calculated and are shown in table 1. These result in a hit rate of 28.125% for a direct mapped cache, 40.625% for a 2-way cache, 46.875% for a 4-way cache and 50.000% for a fully associative cache. This increase in cache hits as the associativity increases is to be expected.

Address	Set a=1	Hit/Miss a=1	Set a=2	Hit/Miss a=2	Set a=4	Hit/Miss a=4	Set a=8	Hi/Miss a=8
0000	0	0	0	0	0	0	0	0
0004	0	1	0	1	0	1	0	1
000c	0	1	0	1	0	1	0	1
2200	0	0	0	0	0	0	0	0
00d0	5	0	1	0	1	0	0	0
00e0	6	0	2	0	0	0	0	0
1130	3	0	3	0	1	0	0	0
0028	2	0	2	0	0	0	0	0
113c	3	1	3	1	1	1	0	1
2204	0	1	0	1	0	1	0	1
0010	1	0	1	0	1	0	0	0
0020	2	1	2	1	0	1	0	1
0004	0	0	0	1	0	1	0	1
0040	4	0	0	0	0	0	0	0
2208	0	0	0	0	0	1	0	1
0008	0	0	0	0	0	1	0	1
00a0	2	0	2	0	0	0	0	0
0004	0	1	0	1	0	1	0	1
1104	0	0	0	0	0	0	0	0
0028	2	0	2	1	0	0	0	1
000c	0	0	0	1	0	1	0	1
0084	0	0	0	0	0	0	0	0
000c	0	0	0	1	0	1	0	1
3390	1	0	1	0	1	0	0	0
00b0	3	0	3	0	1	0	0	0
1100	0	0	0	0	0	1	0	1
0028	2	1	2	1	0	1	0	1
0064	6	0	2	0	0	0	0	0
0070	7	0	3	0	1	0	0	0
00d0	5	1	1	0	1	0	0	0
0008	0	0	0	1	0	1	0	1
3394	1	1	1	1	1	1	0	1

Table 1: Hit and Miss results for various addresses at a=1,2,4,8 associativity.

The results are found to be the same as when the process is performed on paper.