REVERSE ENGINEERING OF L1 CACHE MEMORY - CS20B048 , CS20B080

Latency:

-> For this problem, we initially declared an array of type double of size 400. Later we flushed out the cache memory associated with the array using the command "_mm_clflush()"
-> For measuring latency, first we calculated the initial time by using the commands "RDTSC" and "CPUID"
Here the time taken for the first 32 bits is stored in 'edx' register and for next 32 bits is stored in register 'eax'
Then we started accessing elements of the array, and then we calculated the end time using the commands "RDTSCP" and "CPUID"
And then we calculated the time difference between start and end times and printed out them as the net time which is latency to access that particular element of the array.
We used "RDTSC" and "RDTSCP" commands along "CPUID" instructions to increase accuracy.

Specifications of cache:

-> The actual specifications of L1 cache present in our system is:

Cache block size = 64B

Associativity = 8

The system command used to find this is ' getconf -a | grep CACHE

```
maddirala@LAPTOP-UK10GSOM:~$ getconf -a | grep CACHE
LEVEL1_ICACHE_SIZE                 32768
LEVEL1_ICACHE_ASSOC                8
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 49152
LEVEL1_DCACHE_ASSOC                12
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  524288
LEVEL2_CACHE_ASSOC                 8
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  6291456
LEVEL3_CACHE_ASSOC                 12
LEVEL3_CACHE_LINESIZE              64
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
maddirala@LAPTOP-UK10GSOM:~$
```

Cache block size inferred by reverse engineering:

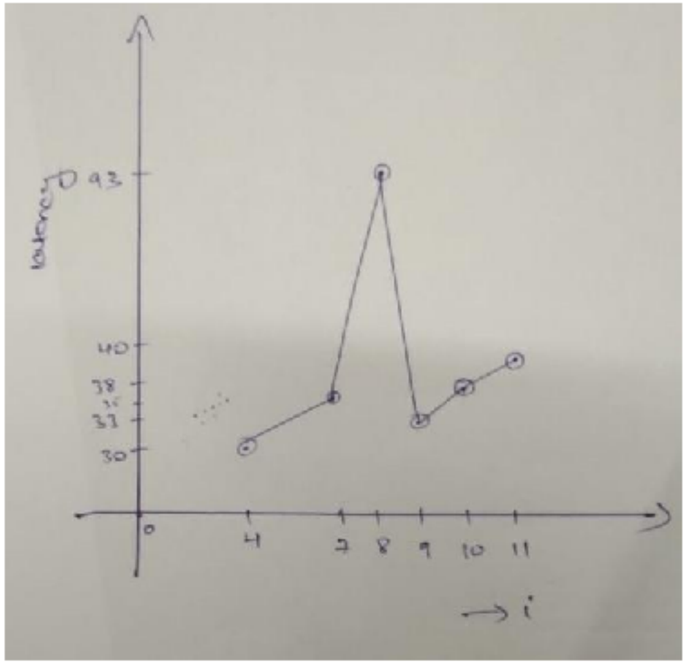
Procedure:

-> After finishing the above process, upon observing the values we got for latency and corresponding index of that element and since we know every double element of the array is of size 8 bytes,we can observe that first element will have more latency because initially cache memory is flushed and it brings a block of data into L1 cache.

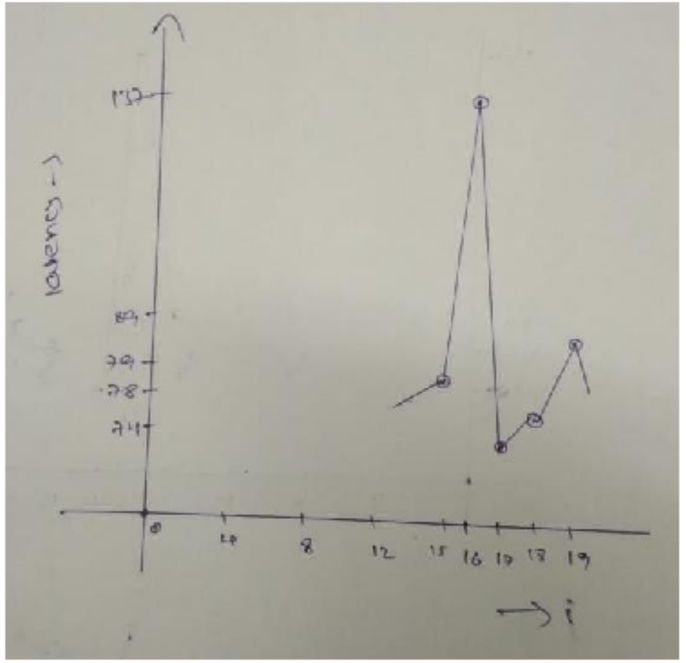
->Now we can say that after finishing each block the latency increases suddenly between two consecutive indices which occurs due to getting a new block.

->Upon observation it is obtained that the latency has this sudden jump in its value for every 7th and 8th index of the array which means after every 64B(double is of 8B and after 8 indices means 64B) it is accessing a new block that is the cache block size is 64B.

Plots:




```
35,7
93,8
33,9
38,10
40,11
```




```
79,15
137,16
74,17
78,18
89,19
```

Justification:

It matches with the actual block size. Here we got some output instances where there is a jump in latency of 8th element with respect to 7th element and similar jump again in latency of 16th element with respect to 15th element.

This shows that for every 8 elements of array, a new block is used and hence resulting the size of cache block as 8*8=64B.