CS2610 LAB #4 : REVERSE ENGINEERING CPU CACHES

Measuring Access latency

The hurdles in measuring access latency are cpu interrupts, non- serial execution order and shared cpu cores. We can force instructions to be executed in serial order by using _mm_lfence() instruction in c.

And, we can use taskset in linux to make our program run on single cpu core. And we can isolate the single core from running other programs before starting the program with isolcpus . We can disable the interrupts in kernel mode. But, I have accidentally found most of the above is already available in a single extension to vs code . CPH judge , an extension used by competitive programmers, by far has provided me with the most stable output.

But, I found a small issue with its cache handling, on a few trial runs, it somehow retained the cache , even after force flushed it in the program. Fortunately, just by running the program again solves it.
Hence, To measure the access latency of the instruction we can use __rdtsc() just before and after the instruction and _mm_lfence() to force serialize it. And, we can separately calculate the overhead caused by the _mm_lfence() and subtract it from the time measured above.

It would yield us the time taken. But , sometimes we get negative time due to the error in overhead value. We are separately calculating the overhead , hence this occurs. But, no worries, It happens once in 200 instructions. Whereas a typical block size is 64B(which requires two dozen instructions if you use int * type).

SYSTEM CACHE Specifications

In linux terminal , type “getconf -a | grep CACHE”.

The output will be something like this :

LEVEL1_ICACHE_SIZE 32768
LEVEL1_ICACHE_ASSOC 8
LEVEL1_ICACHE_LINESIZE 64
LEVEL1_DCACHE_SIZE 32768
LEVEL1_DCACHE_ASSOC 8
LEVEL1_DCACHE_LINESIZE 64
LEVEL2_CACHE_SIZE 262144
LEVEL2_CACHE_ASSOC 4
LEVEL2_CACHE_LINESIZE 64
LEVEL3_CACHE_SIZE 3145728
LEVEL3_CACHE_ASSOC 12
LEVEL3_CACHE_LINESIZE 64
LEVEL4_CACHE_SIZE 0
LEVEL4_CACHE_ASSOC 0
LEVEL4_CACHE_LINESIZE 0

Here Level1_DCACHE is our item of interest. Size refers to the total capacity of cache in Bytes.. Assoc refers to Associativity of Cache . LINESIZE refers to Blocklength in Bytes.

My L1_cache size is 32,768B.
My L1_cache is 8 way associative cache(i.e., each set can hold upto 8 blocks.)
My L1_cache block length is 64B.

Measuring Cache Block Size

Method:
-> Create an int array of size 200.
-> flush all those elements from cache using _mm_clflush();
->Now iterate over each element of the array and find its access time and output.
->Now, look at the date for regular spikes in access time.
-> In such data, the difference between indices of two consecutive spikes yields block length.

The output data from my code is appended as table 1.

The spikes indices are mentioned in the table.
Notice the hit time is around 20 , the variance is due to system noise. The negative value of time is due to error in the value of overhead calculation.

The measured cache block size is 16 * 4 = 64 Bytes (Since each int element is of 4 bytes).

It matches the actual size of the actual CacheBlock size(64 B). Since we are iterating over the array, we need to get a spike each time we access a new block. Thus , the difference between spikes yields block size.

Measuring Cache Associativity

Method:

-> initialize an int array of size 10,000.
-> Find the number of cache sets from the system specs . (No of cache L1 sets in my device is 64).
-> Infer the block size of cache L1 from the above measurement which is 64B.
-> Now, addresses with a difference of 4096 = (64*64) bytes go to the same set i.e., x, x + 4096, x + 2*(4096) , x + 3*(4096),.....
-> lets call the block with address x + k*(4096) as block k.
-> Now , start with i as 1 and access all i blocks once and then once again access all of them but this time calculate their access times also. If there is an outlier in the access times for a particular i , then the cache associativity is (i - 1).
-> Note, on access after eviction from the set, it brings data from cache level 2 to 1. Thus, the spike is really small, try to infer from previous cache access time
->Also, generally cache associativity is in powers of 2.
-> Now , you can stop the program.

The output for my code is appended as table 2.

The measured Cache L1 associativity is (9 - 1) = 8

The actual Cache L1 associativity is 8 .

Justification for the method:

If Cache size is less than or equal to i, it could accommodate all the i blocks and hence their access times will not have outlier. The moment i just crosses the associativity size , we shall have a outlier in access time data.
Hence, associativity size = i - 1;

Table 1

Offset 0	**--- Access time :- 160**	**//Cache miss**
Offset 1	**--- Access time :- -8**	**//error in overload**
Offset 2	**--- Access time :- 16**	
Offset 3	**--- Access time :- 16**	
Offset 4	**--- Access time :- 16**	
Offset 5	**--- Access time :- 16**	
Offset 6	**--- Access time :- 16**	
Offset 7	**--- Access time :- 16**	
Offset 8	**--- Access time :- 16**	
Offset 9	**--- Access time :- 16**	
Offset 10	**--- Access time :- 16**	
Offset 11	**--- Access time :- 10**	
Offset 12	**--- Access time :- 18**	
Offset 13	**--- Access time :- 16**	
Offset 14	**--- Access time :- 16**	
Offset 15	**--- Access time :- 16**	
Offset 16	**--- Access time :- 608**	**//Cache miss**
Offset 17	**--- Access time :- 18**	
Offset 18	**--- Access time :- 16**	
Offset 19	**--- Access time :- 16**	
Offset 20	**--- Access time :- 8**	
Offset 21	**--- Access time :- 16**	
Offset 22	**--- Access time :- 18**	
Offset 23	**--- Access time :- 16**	
Offset 24	**--- Access time :- 16**	
Offset 25	**--- Access time :- 14**	
Offset 26	**--- Access time :- 14**	
Offset 27	**--- Access time :- 16**	
Offset 28	**--- Access time :- 16**	
Offset 29	**--- Access time :- 16**	
Offset 30	**--- Access time :- 16**	
Offset 31	**--- Access time :- 16**	
Offset 32	**--- Access time :- 24**	**//Cache miss**
Offset 33	**--- Access time :- 14**	
Offset 34	**--- Access time :- 16**	
Offset 35	**--- Access time :- 14**	
Offset 36	**--- Access time :- 16**	
Offset 37	**--- Access time :- 16**	
Offset 38	**--- Access time :- 16**	

Offset 39 --- Access time :- 16
Offset 40 --- Access time :- 16
Offset 41 --- Access time :- 16
Offset 42 --- Access time :- 10
Offset 43 --- Access time :- 14
Offset 44 --- Access time :- 16
Offset 45 --- Access time :- 18
Offset 46 --- Access time :- 16
Offset 47 --- Access time :- 16
Offset 48 --- Access time :- 198 //Cache miss
Offset 49 --- Access time :- 16
Offset 50 --- Access time :- 16
Offset 51 --- Access time :- 16
Offset 52 --- Access time :- 12
Offset 53 --- Access time :- 16
Offset 54 --- Access time :- 16
Offset 55 --- Access time :- 18
Offset 56 --- Access time :- 20
Offset 57 --- Access time :- 14
Offset 58 --- Access time :- 14
Offset 59 --- Access time :- 16
Offset 60 --- Access time :- 16
Offset 61 --- Access time :- 18
Offset 62 --- Access time :- 10
Offset 63 --- Access time :- 14
Offset 64 --- Access time :- 28 //Cache miss
Offset 65 --- Access time :- 18
Offset 66 --- Access time :- 16
Offset 67 --- Access time :- 18
Offset 68 --- Access time :- 18
Offset 69 --- Access time :- 16
Offset 70 --- Access time :- 16
Offset 71 --- Access time :- 16
Offset 72 --- Access time :- 16
Offset 73 --- Access time :- 14
Offset 74 --- Access time :- 16
Offset 75 --- Access time :- 16
Offset 76 --- Access time :- 16
Offset 77 --- Access time :- 18
Offset 78 --- Access time :- 16
Offset 79 --- Access time :- 16
Offset 80 --- Access time :- 194 //Cache miss
Offset 81 --- Access time :- 16
Offset 82 --- Access time :- 16
Offset 83 --- Access time :- 20
Offset 84 --- Access time :- 16
Offset 85 --- Access time :- 16
Offset 86 --- Access time :- 16
Offset 87 --- Access time :- 18

Offset 88 --- Access time :- 20
Offset 89 --- Access time :- 18
Offset 90 --- Access time :- 16
Offset 91 --- Access time :- 14
Offset 92 --- Access time :- 16
Offset 93 --- Access time :- 16
Offset 94 --- Access time :- 20
Offset 95 --- Access time :- 16
Offset 96 --- Access time :- 28 //cache miss
Offset 97 --- Access time :- 18
Offset 98 --- Access time :- 16
Offset 99 --- Access time :- 16
Offset 100 --- Access time :- 16
Offset 101 --- Access time :- 16
Offset 102 --- Access time :- 14
Offset 103 --- Access time :- 16
Offset 104 --- Access time :- 16
Offset 105 --- Access time :- 20
Offset 106 --- Access time :- 16
Offset 107 --- Access time :- 16
Offset 108 --- Access time :- 16
Offset 109 --- Access time :- 16
Offset 110 --- Access time :- 18
Offset 111 --- Access time :- -4 //error in overload
Offset 112 --- Access time :- 32 //cache miss
Offset 113 --- Access time :- 18
Offset 114 --- Access time :- 16
Offset 115 --- Access time :- 16
Offset 116 --- Access time :- 20
Offset 117 --- Access time :- 14
Offset 118 --- Access time :- 16
Offset 119 --- Access time :- 16
Offset 120 --- Access time :- 16
Offset 121 --- Access time :- 16
Offset 122 --- Access time :- 14
Offset 123 --- Access time :- 18
Offset 124 --- Access time :- 16
Offset 125 --- Access time :- 16
Offset 126 --- Access time :- 16
Offset 127 --- Access time :- 16
Offset 128 --- Access time :- 28 //cache miss
Offset 129 --- Access time :- 12
Offset 130 --- Access time :- 16
Offset 131 --- Access time :- 16
Offset 132 --- Access time :- 12
Offset 133 --- Access time :- 16
Offset 134 --- Access time :- 18
Offset 135 --- Access time :- 12
Offset 136 --- Access time :- 18

Offset 137 --- Access time :- 12
Offset 138 --- Access time :- 14
Offset 139 --- Access time :- 16
Offset 140 --- Access time :- 16
Offset 141 --- Access time :- 18
Offset 142 --- Access time :- 18
Offset 143 --- Access time :- 10
Offset 144 --- Access time :- 32 //Cache miss
Offset 145 --- Access time :- 18
Offset 146 --- Access time :- 18
Offset 147 --- Access time :- 18
Offset 148 --- Access time :- 16
Offset 149 --- Access time :- 16
Offset 150 --- Access time :- 16
Offset 151 --- Access time :- 16
Offset 152 --- Access time :- 18
Offset 153 --- Access time :- 14
Offset 154 --- Access time :- 14
Offset 155 --- Access time :- 16
Offset 156 --- Access time :- 16
Offset 157 --- Access time :- 18
Offset 158 --- Access time :- 14
Offset 159 --- Access time :- 16
Offset 160 --- Access time :- 28 //Cache miss
Offset 161 --- Access time :- 20
Offset 162 --- Access time :- 10
Offset 163 --- Access time :- 16
Offset 164 --- Access time :- 16
Offset 165 --- Access time :- 12
Offset 166 --- Access time :- 18
Offset 167 --- Access time :- 16
Offset 168 --- Access time :- 16
Offset 169 --- Access time :- 12
Offset 170 --- Access time :- 14
Offset 171 --- Access time :- 14
Offset 172 --- Access time :- 18
Offset 173 --- Access time :- 14
Offset 174 --- Access time :- 16
Offset 175 --- Access time :- 12
Offset 176 --- Access time :- 192 //Cache miss
Offset 177 --- Access time :- 14
Offset 178 --- Access time :- 18
Offset 179 --- Access time :- 16
Offset 180 --- Access time :- 20
Offset 181 --- Access time :- 18
Offset 182 --- Access time :- 16
Offset 183 --- Access time :- 16
Offset 184 --- Access time :- 16
Offset 185 --- Access time :- 18

Offset 186 --- Access time :- 16
Offset 187 --- Access time :- 18
Offset 188 --- Access time :- 16
Offset 189 --- Access time :- 12

Table 2

Blocks loaded 1
Block - 0 -- Access Time:- -6 //error due to overhead

Blocks loaded 2
Block - 0 -- Access Time:- 14
Block - 1 -- Access Time:- 16

Blocks loaded 3
Block - 0 -- Access Time:- 16
Block - 1 -- Access Time:- 14
Block - 2 -- Access Time:- 18

Blocks loaded 4
Block - 0 -- Access Time:- 16
Block - 1 -- Access Time:- 20
Block - 2 -- Access Time:- 24
Block - 3 -- Access Time:- 14

Blocks loaded 5
Block - 0 -- Access Time:- 22
Block - 1 -- Access Time:- 16
Block - 2 -- Access Time:- 16
Block - 3 -- Access Time:- 14
Block - 4 -- Access Time:- 20

Blocks loaded 6
Block - 0 -- Access Time:- 24
Block - 1 -- Access Time:- 24
Block - 2 -- Access Time:- 18
Block - 3 -- Access Time:- 16
Block - 4 -- Access Time:- 18
Block - 5 -- Access Time:- 12

Blocks loaded 7
Block - 0 -- Access Time:- 20
Block - 1 -- Access Time:- 22
Block - 2 -- Access Time:- 14
Block - 3 -- Access Time:- 20
Block - 4 -- Access Time:- 16
Block - 5 -- Access Time:- 8
Block - 6 -- Access Time:- 22

Blocks loaded 8
Block - 0 -- Access Time:- 26
Block - 1 -- Access Time:- 34
Block - 2 -- Access Time:- 36
Block - 3 -- Access Time:- 18
Block - 4 -- Access Time:- 30
Block - 5 -- Access Time:- 18
Block - 6 -- Access Time:- 30
Block - 7 -- Access Time:- 18

Blocks loaded 9
Block - 0 -- Access Time:- 14
Block - 1 -- Access Time:- 38
Block - 2 -- Access Time:- 32
Block - 3 -- Access Time:- 36
Block - 4 -- Access Time:- 18
Block - 5 -- Access Time:- 38
Block - 6 -- Access Time:- 40 **//This is a cycle time for bringing data from cache level 2 to 1**
Block - 7 -- Access Time:- 20
Block - 8 -- Access Time:- 36

Blocks loaded 10
Block - 0 -- Access Time:- 22
Block - 1 -- Access Time:- 28
Block - 2 -- Access Time:- 38
Block - 3 -- Access Time:- 30
Block - 4 -- Access Time:- 38
Block - 5 -- Access Time:- 32
Block - 6 -- Access Time:- 26
Block - 7 -- Access Time:- 16
Block - 8 -- Access Time:- 36
Block - 9 -- Access Time:- 24

Blocks loaded 11
Block - 0 -- Access Time:- 28
Block - 1 -- Access Time:- 38
Block - 2 -- Access Time:- 34
Block - 3 -- Access Time:- 34
Block - 4 -- Access Time:- 36
Block - 5 -- Access Time:- 16
Block - 6 -- Access Time:- 22
Block - 7 -- Access Time:- 34
Block - 8 -- Access Time:- 28
Block - 9 -- Access Time:- 36
Block - 10 -- Access Time:- 20

Blocks loaded 12
Block - 0 -- Access Time:- 32

Block - 1 -- Access Time:- 40
Block - 2 -- Access Time:- 34
Block - 3 -- Access Time:- 30
Block - 4 -- Access Time:- 28
Block - 5 -- Access Time:- 42
Block - 6 -- Access Time:- 34
Block - 7 -- Access Time:- 30
Block - 8 -- Access Time:- 18
Block - 9 -- Access Time:- 36
Block - 10 -- Access Time:- 28
Block - 11 -- Access Time:- 22