The University of Alabama in Huntsville
ECE Department
CPE 431 01/01R, CPE 531 01/91
Fall 2022

Due November 1, 2022

1. 0(15), 2.0.1(10), 2.0.2(10), 2.0.3(20), 3.0.1(5), 3.0.2(10), 4.0.1(5), 4.0.2(10)

1.0 <5.3> Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit hexadecimal memory addresses, given as byte addresses. 74, A0, 78, 38C, AC, 84, 88, 8C, 7C, 34, 38, 13C, 388, 18C

For each of these references, identify the index and the tag, given a three-way set associative cache with two word blocks and a total of 24 words. List if each reference is a hit or a miss, assuming the cache is initially empty and show every entry to the cache, including the tag value and the addresses of all data items stored. Use hexadecimal or binary, whichever is easier.

24 words | 1 block /2 words | 1 set / 3 blocks = 4 sets
Index = 2
Block off = 1
Byte = 2
Tag = 27

0	M[A0…A7]	M[80...87]	
1	M[388..38F]	~~M[A8..AF]~~ M[188..18F]	M[88..8F]
2	M[70..77]	M[30..37]	
3	M[78..7F]	M[38..3F]	M[130..13F]

Address(Hex)	Index (Decimal)	Tag	Hit/Miss
74	2	0000_0000_0000_0000_0000_ 0000_011	M
A0	0	0000_0000_0000_0000_0000_ 0000_101	M
78	3	0000_0000_0000_0000_0000_ 0000_011	M
38C	1	0000_0000_0000_0000_0000_ 0011_100	M
AC	1	0000_0000_0000_0000_0000_ 0000_101	M
84	0	0000_0000_0000_0000_0000_ 0000_100	M
88	1	0000_0000_0000_0000_0000_ 0000_100	M
8C	1	0000_0000_0000_0000_0000_ 0000_100	H

7C	3	0000_0000_0000_0000_0000_ 0000_011	H
34	2	0000_0000_0000_0000_0000_ 0000_001	M
38	3	0000_0000_0000_0000_0000_ 0000_001	M
13C	3	0000_0000_0000_0000_0000_ 0001_001	M
388	1	0000_0000_0000_0000_0000_ 0011_100	H
18C	1	0000_0000_0000_0000_0000_ 0001_100	M

2.0 **<5.4.>** This exercise examines the impact of different cache designs, specifically comparing associative caches to the direct-mapped caches from Section 5.4. For these exercises, use the word address stream given in hexadecimal: 15, A6, C9, 8F, 3D, A6, 3E, 85, 6F, 8F, 90, 3D

2.0.1 Using the references given and a fully associative cache with two-word blocks and a total size of 16 words, identify the index bits, the tag bits, and if it is a hit or miss. Use LRU replacement. Also show all entries made in the cache.

No index.
16 words | 1 block/2 words = 8 blocks
No byte offset.
Block Offset = 1.

Address(Hex)	Tag	Hit/Miss	Block Fill
15	0000_0000_0000_0000_0000_0000_0001_010	M	1(M[14,15])
A6	0000_0000_0000_0000_0000_0000_1010_011	M	2(M[A6,A7])
C9	0000_0000_0000_0000_0000_0000_1100_100	M	3(M[C8,C9])
8F	0000_0000_0000_0000_0000_0000_1000_111	M	4(M[8E,8F])
3D	0000_0000_0000_0000_0000_0000_0011_110	M	5(M[3C,3D])
A6	0000_0000_0000_0000_0000_0000_1010_011	H	Hit block 2
3E	0000_0000_0000_0000_0000_0000_0011_111	M	6(M[3E,3F])
85	0000_0000_0000_0000_0000_0000_1000_010	M	7(M[84,85])
6F	0000_0000_0000_0000_0000_0000_0110_111	M	8(M[6E,6F])
8F	0000_0000_0000_0000_0000_0000_1000_111	H	Hit block 4
90	0000_0000_0000_0000_0000_0000_1001_000	M	1(M[90,91])
3D	0000_0000_0000_0000_0000_0000_0011_110	H	Hit block 5

Multilevel caching is an important technique to overcome the limited amount of space that a first level cache can provide while still maintaining its speed. Consider a processor with the following parameters.

Base CPI, no memory stalls	Processor speed	Main memory access time	First-level cache miss rate per instruction	Second-level cache, direct-mapped speed	Global miss rate with second-level cache, direct-mapped	Second-level cache, eight-way s et associative speed	Global miss rate with second-level cache, eight-way set associative
1.5	2.4 GHz	70 ns	4.5 %	22 cycles	2.5 %	28 cycles	1.5 %

2.0.2 Calculate the CPI for the processor in the table using: 1) only a first level cache, 2) a second level direct-mapped cache, and 3) a second level eight-way set associative cache. How do these numbers change if main memory access time is doubled? If it is cut in half?

Main Memory Miss: 70ns/1/2.4 = 168 clock cycles
Double main memory access time = 336
Half memory access time = 84
1st level:

Normal- CPI = 1.5 + 0.045 * 168 = 9.06
-Double Time-CPI = 1.5 + 0.045 * 336 = 16.62
-Half Time-CPI = 1.5 + 0.045 * 84 = 5.28

2nd level direct mapped:
Base + Main Time*global rate + cache penalty* 1st level miss rate

Normal- CPI = 1.5 + 168*0.025 +0.045* 22 = 6.69
-Double Time-CPI = 1.5 + 0.025 * 336 + 0.045*22 = 10.89
-Half Time-CPI = 1.5 + 0.025 * 84 + 0.045*22 = 4.59

2nd level 8 way set:
Base + Main time * global rate + cache penalty * 1st level miss rate

Normal- CPI = 1.5 + 168*0.015 +0.045* 28 = 5.28
-Double Time-CPI = 1.5 + 0.015 * 336 + 0.045*28 = 7.8
-Half Time-CPI = 1.5 + 0.015 * 84 + 0.045*28 = 4.02

2.0.3 In older processors such as the Intel Pentium or Alpha 21264, the second level of cache was external (located on a different chip) from the main processor and the first-level cache. While this allowed for large second-level caches, the latency to access the cache was much higher, and the bandwidth was typically lower because the second-level cache ran at a lower frequency. Assume a 512 KiB off-chip second-level cache has a global miss rate of 4 %. If each additional 512 KiB of cache lowered the global miss rate by 0.5 %, and the cache had a total access time of 50 cycles, how big would the cache have to be to match the performance of the second-level direct-mapped cache listed in the table? Of the eight-way set-associative cache?

2nd Level DM: 1.5 + 120* 0.04 + 0.045 * 22 = 7.29 (ignore)
2nd level 8 way set: 1.5 + 120*0.04 + 0.045 *28 = 7.56 (ignore)

Now L2 missrate = .04 – 0.005n

2nd Level DM: 1.5 + 120* 0.04-0.005n + 0.045 * 22 = 6.69
1.5 + 120(0.035n) + 0.045 * 22 = 6.69

1.5 + (4.2n) + 0.99 =6.69
2.49 + 4.2n = 6.69
4.2n = 4.2
N = 1, Compared to original CPI of 6.69 you wouldn't need any extra caches.

2nd level 8 way set: 1.5 + 120*0.04-0.005n + 0.045 *28 = 5.28
1.5 + 120(0.035n) + 0.045 * 28 = 5.28
1.5 + (4.2n) + 1.26 = 5.28
2.76 + 4.2n = 5.28
4.2n = 2.52
N = 0.6

Compared to the original cpi of 5.28 you would need 1 512 KB off chip second level caches

3.0 **<5.5>** This exercise examines the single error correcting, double error detecting (SEC/DED) Hamming code.

3.0.1 What is the minimum number of parity bits required to protect a 256-bit word using the SEC/DED code?

SEC/DED: $2^P \geq p + d + 1$
Data bits in a bit- word = 256
$2^P \geq p + 257$ -> p = 9 (which is$512 \geq 266$)

3.0.2 Consider an SEC code that protects 8 bit words with 4 parity bits. If we read the value 0x876, is there an error? If so, correct the error.

1 2 3 4 ;5 6 7 8;9 1 11 2 3;4 5 6 (bit numbering)
1 0 0 0_0 1 1 1_0 1 1 0
_ _ 1 _ ; 0 0 0 _;0 1 1 1;0 1 1 0 (with parity space) – ignore used for encoding and my reference

P_1: XOR(1,0,0,1,0,1) = 1
P_2: XOR(0,0,1,1,1,1) = 0 (odd parity)
P_4: XOR(0,0,1,1,0) = 0 (odd parity)
P_8: XOR(1,0,1,1,0) = 1

$P_8P_4P_2P_1$ = 1001 = 9

So should be 1000_0111_1110 = 0x87E

4.0 Media applications that play audio or video files are part of a class of workloads called "streaming" workloads (i.e., they bring in large amounts of data but do not reuse much of it). Consider a video streaming workload that accesses a 2048 KiB working set sequentially with the following address stream:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, …

4.0.1 Assume a 128 KiB direct-mapped cache with a 16-byte block. What is the miss rate for the address stream above?

Block Size: 2048 KB/16B block = 128 K
Size of direct mapped cache/size of block = 128/16 = 8 * 8 = 64 bits
Every 64 bits is a miss. 1/64 = 0.015625 = 1.5625%

4.0.2 “Prefetching” is a technique that leverages predictable address patterns to speculatively bring in additional cache blocks when a particular cache block is accessed. One example of prefetchcing is a stream buffer that prefetches sequentially adjacent cache blocks into a separate buffer when a particular cache block is brought in. If the data is found in the prefetch buffer, it is considered a hit and moved into the cache and the next cache block is prefetched. Assume a two-entry stream buffer and assume that the cache latency is such that a cache block can be loaded before the computation on the previous cache block is completed. What is the miss rate for the address stream above?

Miss rate = misses/access

Misses =1.

Access = 2048 KB/2 = 2048 * 1024/2 = 1048576

Miss rate = 1/1048576 = 0.000095367%