Selected Question: 5.1.1~5.1.3, 5.3.1~5.3.5, 5.5.1~5.5.6, 5.7.1~5.7.5

5.1 In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 32-bit integer.

```
for    (I=0; I<8; I++)
  for    (J=0; J<8000; J++)
    A[ I ][ J ] = B[ I ][ 0 ]+A[ J ][ I ];
```

5.1.1 [5] <§5.1> How many 32-bit integers can be stored in a 16-byte cache block?

5.1.1. 4

5.1.2 [5] <§5.1> References to which variables exhibit temporal locality?

5.1.2 I,J, and B[I][0]

A[0][0] = B[0][0] + A[0][0]
A[0][1] = B[0][0] + A[1][0]
A[0][2] = B[0][0] + A[2][0]
….
A[0][7999]= B[0][0] + A[7999][0]
A[1][0]=B[1][0]+ B[0][1]
A[1][1]=B[1][0]+ B[1][1]
A[1][2]=B[1][0]+ B[2][1]
….
Therefore, I, J, and B[I][0] exhibit temporal locality

5.1.3 [5] <§5.1> References to which variables exhibit spatial locality?

5.1.3
A[I][J] only. Note that A[J][I] doesn't exhibit spatial locality because only data within the same row are stored contiguously. Access pattern for A[J][I] are A[0][0], A[1][0], A[2][0]….. It

Access pattern for A[I][J] = A[0][0], A[0][1], A[0][2] ….
Access pattern for B[I][0] = B[0][0], B[0][0]….
Access pattern for A[J][I] = A[0][0], A[1][0], A[2][0]
Therefore, only A[I][J] exhibits spatial locality

5.3 For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31-10	9-5	4-0

5.3.1 [5] <§5.3 >What is the cache block size (in words)?

5.3.1. Offset has 5 bits. 2^5=32 bytes = 8 words

5.3.2 [5] <§5.3> How many entries does the cache have?

5.3.2 Index has 5 bits. 2^5=32 entries

5.3.3 [5] <§5.3>What is the ratio between total bits required for such a cache implementation over the data storage bits? (Do not include valid bit)

5.3.3

Data size = 32 * 32*8 = 8192 bits

Total storage bits if no valid bit = 32* (22+ 32*8)= 8896

Total storage bits if there are valid bits = 32 * (22+ 32*8+1) = 8928 bits

8896/8192 = 1.086 (without valid bits)

8928/8192 = 1.089 (with valid bits)

Starting from power on, the following byte-addressed cache references are recorded.

Address											
0	4	16	132	232	160	1024	30	140	3100	180	2180

5.3.4 [10] <§5.3> How many blocks are replaced?

5.3.4 3 blocks are replaced.

Byte Address	Binary Address	Tag	Index	Hit/Miss
0	0000 0000 0000	0	00000	M
4	0000 0000 0100	0	00000	H
16	0000 0001 0000	0	00000	H
132	0000 1000 0100	0	00100	M
232	0000 1110 1000	0	00111	M
160	0000 1010 0000	0	00101	M
1024	0100 0000 0000	1	00000	M
30	0000 0001 1110	0	00000	M
140	0000 1000 1100	0	00100	H
3100	1100 0001 1100	3	00000	M
180	0000 1011 0100	0	00101	H
2180	1000 1000 0100	2	00111	M

5.3.5 [10] <§5.3>What is the hit ratio?

5.3.5 4 hits in 12 accesses. Hit ratio = 4/12 = 0.33

5.5 Media applications that play audio or video ales 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 512 KiB working set sequentially with the following address stream:

0,2,4,6,8,10,12,14,16,…

5.5.1[5] <§§5.4, 5.8>Assume a 64 KiB direct- mapped cache with a 32-byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

5.5.1 Assuming the addresses given as byte addresses, each group of 16 accesses will map to the same 32-byte block so the cache will have a miss rate of 1/16. For example, accesses 0, 2, 4, 6, …….30 will be mapped into the same 32-byte. All misses are compulsory misses. The miss rate is not sensitive to the size of the cache or the size of the working set. It is, however, sensitive to the access pattern and block size.

5.5.2[5] <§§5.1,5.8>Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

5.5.2The miss rates are 1/8, 1/32, and 1/64, respectively. The workload is exploiting spatial locality.

5.5.3[10] <§5.13> “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 prefetching 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 as 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?

5.5.3 In this case the miss rate is 0.

Cache block size (B) can affect both miss rate and miss latency. Assuming a 1-CPI machine with an average of 1.35 references (both instruction and data) per instruction, help find the optimal block size given the following miss rates for various block sizes.

8：4%	16：3%	32：2%	64：1.5%	128：1%

5.5.4 [10] <§5.3>What is the optimal block size for a miss latency of 20×B cycles?

5.5.4

Average Memory Access Time (AMAT) = (Time for a Hit) + (Miss Rate) x (Miss Latency). Since CPI is given as one, the time for a hit in this case is one cycle.

Block size (B)	Miss Rate	Latency	
8 bytes	4%	8*20=160 cycles	1+4%*160=7.4 cycles
16 bytes	3%	320 cycles	1+3%*320=10.6 cycles
32 bytes	2%	640 cycles	1+2%*640=13.8
64 bytes	1.5%	1280 cycles	1+1.5%*1280 = 20.2
128 bytes	1%	2560 cycles	1+1%*2560=26.6

Therefore, block size=16 bytes is optimal.

5.5.5 [10] <§5.3>What is the optimal block size for a miss latency of 24+B cycles?

5.5.5

Average Memory Access Time (AMAT) = (Time for a Hit) + (Miss Rate) x (Miss Latency). Since CPI is given as one, the time for a hit in this case is one cycle.

Block size (B)	Miss Rate	Latency	
8 bytes	4%	8+24=32 cycles	1+4%*32=2.28 cycles
16 bytes	3%	16+24=40 cycles	1+3%*40=2.20 cycles
32 bytes	2%	32+24=56 cycles	1+2%*56=2.12

64 bytes	1.5%	64+24=88 cycles	1+1.5%*88 = 2.32
128 bytes	1%	128+24=152 cycles	1+1%*153=2.53

Therefore, block size = 32 is optimal.

5.5.6 [10] <§5.3>For constant miss latency, what is the optimal block size?

5.5.6 B=128

Average Memory Access Time (AMAT) = (Time for a Hit) + (Miss Rate) x (Miss Latency). Since CPI is given as one, the time for a hit in this case is one cycle.

Assume Latency = L

Block size (B)	Miss Rate	Latency	
8 bytes	4%	L cycles	1+4%*L cycles
16 bytes	3%	L cycles	1+3%*L cycles
32 bytes	2%	L cycles	1+2%*L cycles
64 bytes	1.5%	L cycles	1+1.5%*L cycles
128 bytes	1%	L cycles	1+1%*L cycles

Therefore, block size = 128 is optimal.

5.7 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, refer to the address stream shown in Exercise 5.2. (Note: the address stream is 3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253. Each one is word address)

5.7.1 [10] <§5.4> Using the sequence of references from Exercise 5.2, show the final cache contents for a three-way set associative cache with two-word blocks and a total size of 24 words. Use LRU replacement. For each reference identify the index bits, the tag bits, the block offset bits, and if it is a hit or a miss.

5.7.1 The cache would have 24 / 3 = 8 blocks per way and thus an index field of 3 bits.

The word address is 3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

Word Address	Binary Address	Tag	Index	Block offset	Hit/Miss	Way 0	Way 1	Way 2
3	0000 0011	0000	001	1	M	T(1)=0		
180	1011 0100	1011	010	0	M	T(1)=0, T(2)=11		
43	0010 1011	0010	101	1	M	T(1)=0, T(2)=11 T(5)=2		
2	0000 0010	0000	001	0	M	T(1)=0, T(2)=11 T(5)=2	T(1)=0	
191	1011 1111	1011	111	1	M	T(1)=0, T(2)=11 T(5)=2, T(7)=11	T(1)=0	
88	0101 1000	0101	100	0	M	T(1)=0,	T(1)=0	

						T(2)=11 T(5)=2, T(7)=11 T(4)=5		
190	1011 1110	1011	111	0	H	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5	T(1)=0	
14	0000 1110	0000	111	0	M	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5	T(1)=0 T(7)=0	
181	1011 0101	1011	010	1	H	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5	T(1)=0 T(7)=0	
44	0010 1100	0010	110	0	M	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5, T(6)=2	T(1)=0 T(7)=0	
186	1011 1010	1011	101	0	M	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5, T(6)=2	T(1)=0 T(7)=0 T(5)=11	
253	1111 1101	1111	110	1	M	T(1)=0, T(2)=11 T(5)=2, T(7)=11 T(4)=5, T(6)=2	T(1)=0 T(7)=0 T(5)=11 T(6)=15	

5.7.2 [10] <§5.4> Using the references from Exercise 5.2, show the final cache contents for a fully associative cache with one-word blocks and a total size of 8 words. Use LRU replacement. For each reference identify the index bits, the tag bits, and if it is a hit or a miss.

5.7.2

Each block has 1 word, and the cache would have 8/ 1 = 8 blocks.

Since this cache is fully associative and has one-word blocks, the word address is equivalent to the tag. The only possible way for there to be a hit is a repeated reference to the same word, which does not occur for this

sequence.

Tag	Hit/Miss	Contents
3	M	3
180	M	3,180
43	M	3,180,43
2	M	3,180,43,2
191	M	3,180,43,2,191
88	M	3,180,43,2,191,88
190	M	3,180,43,2,191,88,190
14	M	3,180,43,2,191,88,190,14
181	M	181,180,43,2,191,88,190,14
44	M	181,44,43,2,191,88,190,14
186	M	181,44,186,2,191,88,190,14
253	M	181,44,186,253,191,88,190,14

5.7.3 [15] <§5.4> Using the references from Exercise 5.2, what is the miss rate for a fully associative cache with two-word blocks and a total size of 8 words, using LRU replacement? What is the miss rate using MRU (most recently used) replacement? Finally, what is the best possible miss rate for this cache, given any replacement policy?

5.7.3

Each block has 2 word, and the cache would have 8/ 2 = 4 blocks.

Tag has 7 bits.

Address	Binary Address	Tag	Hit/Miss	Contents
3	0000 0011	1	M	1
180	1011 0100	90	M	1,90
43	0010 1011	21	M	1,90,21
2	0000 0010	1	H	1,90,21
191	1011 1111	95	M	1,90,21,95
88	0101 1000	44	M	1,90,21,95,44
190	1011 1110	95	H	1,90,21,95,44
14	0000 1110	7	M	1,90,21,95,44,7
181	1011 0101	90	H	1,90,21,95,44,7
44	0010 1100	22	M	1,90,21,95,44,7,22
186	1011 1010	143	M	1,90,21,95,44,7,22,143
253	1111 1101	126	M	1,90,126,95,44,7,22,143

The final reference replaces tag 21 in the cache, since tags 1 and 90 had been reused at time=3 and time=8 while 21 hadn't been used since time=2.

Miss rate = 9/12 = 75%

Miss rate if MRU is used = 75%

This is the best possible miss rate, since there were no misses on any block that had been previously evicted

from the cache. In fact, the only eviction was for tag 21, which is only referenced once.

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 Set Associative Speed	Global Miss Rate with Second Level Cache, Eight-Way Set Associative
1.5	2GHz	100ns	7%	12cycles	3.5%	28cycles	1.5%

5.7.4 [10] <§5.4>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?

5.7.4 (The answer in the solution manual is incorrect)

Clock cycle = 0.5ns

Base CPI = 1.5

Memory Access Cycle = 100ns /0.5 = 200 cycles

L1 only:	Direct mapped L2:	8-way set associated L2:
CPI =1.5 + 7% * 200 = 15.5	CPI= 1.5+ 7%* 12+ 3.5% *200 = 9.34	CPI= 1.5+ 7%*28 + 1.5% * 200 =6.46

Doubled memory access time,

L1 only:	Direct mapped L2:	8-way set associated L2:
CPI =1.5 + 7% * 400 = 29.5	CPI= 1.5+ 7%* 12+ 3.5% *400 = 16.34	CPI= 1.5+ 7%*28 + 1.5% * 400 =9.46

Half memory access time

L1 only:	Direct mapped L2:	8-way set associated L2:
CPI =1.5 + 7% * 100 = 8.5	CPI= 1.5+ 7%* 12+ 3.5% *100 = 5.84	CPI= 1.5+ 7%*28 + 1.5% * 100 =4.96

5 .7.5 [10] <§5.4>It is possible to have an even greater cache hierarchy than two levels. Given the processor above with a second level, direct-mapped cache, a designer wants to add a third level cache that takes 50

cycles to access and will reduce the global miss rate to 1.3%. Would this provide better performance? In general, what are the advantages and disadvantages of adding a third level cache?

5.7.5

New memory hierarchi

Base CPI, No Memory Stalls	First Level Cache Miss Rate per Instruction	Second Level Cache, Direct-Mapped Speed	Global Miss Rate with Second Level Cache, Direct-Mapped	Third Level Cache	Global Miss Rate with Third Level Cache, Direct Mapped	Memory Access
1.5	7%	12cycles	3.5%	50 cycles	1.3%	200 Cycle

1.5+ 7% *12 + 3.5%* 50 + 1.3% * 200 = 5.39 Cycles per instruction (CPI)

Adding the L3 cache does reduce the overall memory access time, which is the main advantage of having a L3 cache. The disadvantage is that the L3 cache takes real estate away from having other types of resources, such as functional units.