CS24 Elementary Computer Organization
Chapter 5 Exercises: 5.1.1, 5.1.2, 5.1.3, 5.2.1, 5.2.2, 5.2.3, 5.7.1, 5.7.2, 5.9.1

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?

16bytes = 128bits
$\frac{128bits}{1block} \times \frac{1int}{32bits}$ = 4 32-bit integers

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

The variables I and J are constantly being changed through the use of loops in this code. This means that as they are being incremented, their values are being stored in the cache and therefore they are highly likely to be accessed many more times which demonstrates the use of temporal locality.

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

A[I][J] is exhibiting spatial locality as the values stored at the arrays address in memory are close together. Because of this, as the 2D array is indexed, the cache is likely to use values that are close to the ones being accessed through the means of this array.

5.2 Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit memory address references, given as word addresses.

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

5.2.1 [10] <5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Index	Valid	Tag	
0000	0	-	
0001	0	-	
0010	1	0000	2
0011	1	0000	3
0100	1	1011	180
0101	1	1011	181
0110	0	-	
0111	0	-	
1000	0101	88	
1001	0	-	
1010	1	1011	186
1011	1	0010	43
1100	1	0010	44
1101	1	1111	253
1110	1	0000	~~180~~ 14
1111	1	1011	191

$\overset{m}{3}\ \overset{m}{180}\ \overset{m}{43}\ \overset{m}{2}\ \overset{m}{191}\ \overset{m}{88}\ \overset{m}{190}\ \overset{m}{14}\ \overset{m}{181}\ \overset{m}{44}\ \overset{m}{186}\ \overset{m}{253}$

5.2.2 [10] <5.3> For each of these references, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Index	Valid	Tag	
000	0	-	
001	1	0000	2-3
010	1	1011	180-181
011	0	-	
100	1	0101	88-89
101	1	1011	~~42-43~~ 186-187
110	1	1111	~~44-45~~ 252-253
111	1	0000	~~190-191~~ 14-15

m	*m*	*m*	*HIT*	*m*	*m*	*HIT*	*m*	*HIT*	*m*	*m*	*m*
3	180	43	2	191	88	190	14	181	44	186	253

5.2.3 [20] <5.3, 5.4> You are asked to optimize a cache design for the given references. There are three direct-mapped cache designs possible, all with a total of 8 words of data: C1 has 1-word blocks, C2 has 2-word blocks, and C3 has 4-word blocks. In terms of miss rate, which cache design is the best? If the miss stall time is 25 cycles, and C1 has an access time of 2 cycles, C2 takes 3 cycles, and C3 takes 5 cycles, which is the best cache design?

CACHE C1:

Index	Valid	Tag	
000	1	01011	88
001	0	-	
010	1	10111	~~2~~ 186
011	1	00101	~~3~~ 43
100	1	00101	~~180~~ 44
101	1	11111	~~181~~ 253
110	1	00001	14
111	1	10111	~~191~~ 190

$\overset{m}{3}\ \overset{m}{180}\ \overset{m}{43}\ \overset{m}{2}\ \overset{m}{191}\ \overset{m}{88}\ \overset{m}{190}\ \overset{m}{14}\ \overset{m}{181}\ \overset{m}{44}\ \overset{m}{186}\ \overset{m}{253}$

CACHE C2:

Index	Valid	Tag	
00	1	01011	88-89
01	1	10111	~~2-3~~ ~~42-43~~ ~~2-3~~ 186-187
10	1	11111	~~180-181~~ ~~44-45~~ 252-253
11	1	00001	~~190-191~~ 14-15

$\overset{m}{3}\ \overset{m}{180}\ \overset{m}{43}\ \overset{m}{2}\ \overset{m}{191}\ \overset{m}{88}\ \overset{HIT}{190}\ \overset{m}{14}\ \overset{HIT}{181}\ \overset{m}{44}\ \overset{m}{186}\ \overset{m}{253}$

CACHE C3:

Index	Valid	Tag	
0	1	10111	(~~0-1-2-3~~) (~~40-41-42-43~~) (~~0-1-2-3~~) (~~88-89-90-91~~) 184-185-186-187
1	1	11111	(~~180-181-182-183~~) (~~188-189-190-191~~) (~~12-13-14-15~~) (~~180-181-182-183~~) (~~44-45-46-47~~) 252-253-254-255

$\overset{m}{3}\ \overset{m}{180}\ \overset{m}{43}\ \overset{m}{2}\ \overset{m}{191}\ \overset{m}{88}\ \overset{HIT}{190}\ \overset{m}{14}\ \overset{m}{181}\ \overset{m}{44}\ \overset{m}{186}\ \overset{m}{253}$

Out of all the caches, cache C2 has the best performance as it has the most amount of data values that resulted in a HIT.

The rate at which C1 misses when placing values in the cache is $\frac{12}{12} \times 100 = 100\%$. The total cycles for C1 $= totalmisses \times missstalltime + references \times accesstime = 12 \times 25 + 12 \times 2 = 324 cycles$

The rate at which C2 misses when placing values in the cache is $\frac{10}{12} \times 100 = 83\%$. The total cycles for C2 $= totalmisses \times missstalltime + references \times accesstime = 10 \times 25 + 12 \times 3 = 286 cycles$

The rate at which C3 misses when placing values in the cache is $\frac{11}{12} \times 100 = 92\%$. The total cycles for C3 $= totalmisses \times missstalltime + references \times accesstime = 11 \times 25 + 12 \times 5 = 335 cycles$

So, the second cache has the best performance since it requires the least amount of cycles.

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

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253

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 off set bits, and if it is a hit or a miss.

Index	Valid	Tag	Valid	Tag	Valid	Tag	
00	1	01011	0	-	0	-	88-89
01	1	00101	1	00000	1	10111	~~2-3~~ 42-43 2-3 186-187
10	1	10110	1	00101	1	11111	180-181 44-45 252-253
11	1	10111	1	00001	0	-	190-191 14-15

$$\begin{matrix} m & m & m & HIT & m & m & HIT & m & HIT & m & m & m \\ 3 & 180 & 43 & 2 & 191 & 88 & 190 & 14 & 181 & 44 & 186 & 253 \end{matrix}$$

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.

References:	~~3~~ 181	~~180~~ 44	~~2~~ 186	~~191~~ 253	191	88	190	14
Tags:	0xB5	0x2C	0xBA	0xFD	0xBF	0x58	0xBE	0x0E
Valid:	1	1	1	1	1	1	1	1

(I wrote the tags in hex because if they were written in binary the table wouldnt fit)

$$\overset{m}{3}\ \overset{m}{180}\ \overset{m}{43}\ \overset{m}{2}\ \overset{m}{191}\ \overset{m}{88}\ \overset{m}{190}\ \overset{m}{14}\ \overset{m}{181}\ \overset{m}{44}\ \overset{m}{186}\ \overset{m}{253}$$

5.9 This Exercise examines the single error correcting, double error detecting (SEC/ DED) Hamming code.

5.9.1 [5] <5.5> What is the minimum number of parity bits required to protect a 128-bit word using the SEC/DED code?

The following formula can be used to determine the minimum number of parity bits required for a 128-bit word:

$2^p >= p + d$
$2^p >= p + 128bits$

The minimum number that can be plugged into p in the expression to make it logically correct is the value 8:

$2^8 >= 8 + 128bits$
$256 >= 136$

So the minimum number of parity bits needed is 8.