Burak Yesil
“I pledge my honor that I have abided by the Stevens Honor System.”

Problem 1:

A. Variables i, j, m, n
B. The Variable A and the 2d array Anew
C. The program would be slower because the structure of the array is [j][i]. (further explanation below)
2d Array:

1	2	3	4
5	6	7	8

Memory Structure:

1
2
3
4
5
6
7
8

As you can see the matrix is stored in memory row by row. So if the outer loop is i and the inner loop is j, then we would be going column by column meaning we would first visit **(row 1 column 1)**, **(row 2 column 1)**, (**row 3 column 1)**, we would then move on to the next column. This means that instead of going in sequence down the stack/memory, we would have to jump to different parts of the memory, making the program with the new changes slower.

Problem 2:

A.

Reference	Binary Word Address	Index	Tag	hit/miss
0x43	**1000011**	1000011%10000 = **0011**	**100**	**miss**
0xc4	11000100	11000100%10000= **0100**	**1100**	**miss**
0x2b	101011	101011%10000 = **1011**	**10**	**miss**
0x42	1000010	1000010%10000 = **0010**	**100**	**miss**
0xc5	11000101	11000101%10000 = 0**101**	**1100**	**miss**
0x28	101000	101000 % 10000 = **1000**	**10**	**miss**
0xbe	10111110	10111110 % 10000 = **1110**	**1011**	**miss**
0x05	101	101%10000 = **0101**	**0**	**miss**
0x92	10010010	10010010% 10000 = **0010**	**1001**	**miss**
0x2a	101010	101010%10000 = **1010**	**10**	**miss**
0xba	10111010	10111010% 10000 = **1010**	**1011**	**miss**

0xbd	10111101	10111101% 10000 = **1101**	**1011**	**miss**

B.

hex	binary	tag	index	offset	hit/miss
0x43	1000011	100	001	1	miss
0xc4	11000100	1100	010	0	miss
0x2b	101011	10	101	1	miss
0x42	1000010	100	001	0	hit
0xc5	11000101	1100	010	1	hit
0x28	101000	10	100	0	miss
0xbe	10111110	1011	111	0	miss
0x05	101	0	010	1	miss
0x92	10010010	1001	001	0	miss
0x2a	101010	10	101	0	hit
0xba	10111010	1011	101	0	miss
0xbd	10111101	1011	110	1	miss

Problem 3:

a.)

(8*30).5 =12

(16*30).3 = 14.1

(32*30).02 = 19.2

(64*30).02 = 19.2

(128*30).011 = 42.24

8 bytes is the optimal block size

b.)

(8+24)*.05=1.6

(16+24)*.03=1.2

(32+24)*.02=1.12

(64+28)*.015=1.32

(128+24)*.011=1.672

32 bytes is the optimal block size.

c.)

The largest block size, 128, is the optimal one since the latency is constant.

Problem 4:

A.)

512 block = 2^n, n =9.

1-word block: 2^m, m=2

64-9 = 55

53 bits for the tag, 9 bits for the index, 2 bits for the offset

B.)

64 blocks = 2^n, n=6.

8-word blocks = 2^m, m =5.

64-(6+3) = 55

53 bits for the tag, 6 bits for the index, 5 bits for the offset

C.)

(Using the following equation: 2^n (2^m *35 +63-n-m))

1. 2^9(32 +63 -9) = 44,032

   44032/64 = 688/1 bit

2. 2^6(2^3 * 35 +63 -9 -3) = 19,648

   19648/64= 307/1 bit

D.)

2 way associative

54 for the tag

8 for the index

2 for the offset

Problem 5:

	Tag	Data	Tag	Data	Tag	Data	Tag	Data
00								
01	101101011	Mem[0xb2d]						
10	1100010010	Mem[0xc4a]	100100100	Mem[0x492]	001010 1000	Mem[0x2a2]	0011101110	Mem[0x3ba]
11	0001010000	Mem[0x143]	0010001010	Mem[0x22b]	010000 1011	Mem[0x42f]		