Pledge: I pledge my honor that I have abided by the Stevens Honor System.
Name: Haig Emirzian

Homework #4

1.1
P1: 3.2 x 10^9 / 1.5 = **2.13 x 10^9 instructions per second**
P2: 2.0 x 10^9 / 1.0 = **2.0 x 10^9 instructions per second**
P3: 4.0 x 10^9 / 2.3 = **1.7 x 10^9 instructions per second**

1.2
Instructions: 2.13 x 10^9 * 10 = **2.13 x 10^10 instructions per 10 seconds**
Cycles: 3.2 x 10^9 * 10 = **3.2 x 10^10 cycles**

Instructions: 2.0 x 10^9 * 10 = **2.0 x 10^10 instructions per 10 seconds**
Cycles: 2.0 x 10^9 * 10 = **2.0 x 10^10 cycles**

Instructions: 1.7 x 10^9 * 10 = **1.7 x 10^10 instructions per 10 seconds**
Cycles: 4.0 x 10^9 * 10 = **4.0 x 10^10 cycles**

1.3
10 * 0.7 = 7 seconds
CPI = 1 * 1.2 = 1.2
New clock rate = (Old clock rate * CPI) / Execution time
2.0 x 10^10 * 1.2 / 7
= **3.4 x 10^9 Hertz**

2.1
Class A: 3 x 10^5
(1 x 10^6) * 0.3
Class B: 2 x 10^5
(1 x 10^6) * 0.2
Class C: 3 x 10^5
(1 x 10^6) * 0.3
Class D: 2 x 10^5
(1 x 10^6) * 0.2

P1: 1*(3 x 10^5) + 2*(2 x 10^5) + 3*(3 x 10^5) + 3*(2 x 10^5)
= (3 x 10^5) + (4 x 10^5) + (9 x 10^5) + (6 x 10^5)

= 2.2 x 10^6 Hertz

P2: 2*(3 x 10^5) + 2*(2 x 10^5) + 2*(3 x 10^5) + 2*(2 x 10^5)
= (6 x 10^5) + (4 x 10^5) + (6 x 10^5) + (4 x 10^5)
= 2.0 x 10^6 Hertz

P1: (2.2 x 10^6) / (1 x 10^6) = **2.2 cycles per instruction**
P2: (2.0 x 10^6) / (1 x 10^6) = **2.0 cycles per instruction**

2.2
CPI / clock rate = clock cycle time

P1: 2.2 x 10^6 / 2.5 x 10^9 = **8.8 x 10^-4 seconds**
P2: 2.0 x 10^6 / 3.0 x 10^9 = **6.7 x 10^-4 seconds**

2.3
8.8 x 10^-4 seconds
6.7 x 10^-4 seconds
P2 is faster.

3.1
Anew[j][i], i, j, m, n, and err

3.2
A[j][i+1], A[j][i-1], A[j-1][i], A[j+1][i]

3.3
It would be slower because elements in the same row are stored contiguously, and the program is filling the matrix by the rows first then by the columns. By flipping it, you would no longer be storing items contiguously.

4.1
No offset because it's a one-word and word-addressed
log2(16) = 4 bits

	Tag	**Index**	**Hit/Miss:** Check for prior similar tag and index
0x43:	0100	0011	Miss
0xc4:	1100	0100	Miss
0x2b:	0010	1011	Miss
0x42:	0100	0010	Miss
0xc5:	1100	0101	Miss

0x28: 0010 1000 Miss
0xbe: 1011 1110 Miss
0x05: 0000 0101 Miss
0x92: 1001 0010 Miss
0x2a: 0010 1010 Miss
0xba: 1011 1010 Miss
0xbd: 1011 1101 Miss

4.2
log2(2) = 1
Offset size: 1 bit
log2(8) = 3
Set Index size: 3 bit
8-3-1 = 4
Tag size: 4 bits

	Tag	**Set Index**	**Offset**	**Hit/Miss**
0x43:	0100	001	1	Miss
0xc4:	1100	010	0	Miss
0x2b:	0010	101	1	Miss
0x42:	0100	001	0	Hit
0xc5:	1100	010	1	Hit
0x28:	0010	100	0	Miss
0xbe:	1011	111	0	Miss
0x05:	0000	010	1	Miss
0x92:	1001	001	0	Miss
0x2a:	0010	101	0	Hit
0xba:	1011	101	0	Miss
0xbd:	1011	110	1	Miss

5.1
1 word: 4 bytes
Cache size: 2^n = 4
log2(4) = 2 bits
Offset: 2 bits
log2(512) = 9
Set Index: 9 bits
64-9-2 = 53
Tag: 53 bits

Tag: 52 bits—--Set Index: 9 bits—-----Offset: 2 bits

5.2
8 word block: 32 bytes
log2(32) = 5
Offset: 5 bits
log2(64) = 6
Set Index: 6 bits
64-5-6 = 53 bits
Tag: 53 bits

Tag: 53 bits—--Set Index: 6 bits—-----Offset: 5 bits

5.3
32 bytes per (53+1+32)
256 bytes per (53+1+256)
RatioA: 32/86
RatioB: 256/310

5.4
1 word block = 4 bytes
log2(4) = 2
Offset: 2 bits
512/2 = 256
log2(256) = 8
Set index: 8 bits
64-2-8 = 54 bits
Tag: 54 bits

Tag: 54 bits—--Set Index: 8 bits—-----Offset: 2 bits

6
16 blocks 4 way
16/4 = 4 blocks
log2(4) = 2 bits
Offset: 2 bits
log2(4) = 2 bits
Set Index: 2 bits
12-2-2 = 6 bits
Tag: 8 bits

Tag: 8 bits—--Set Index: 2 bits—-----Offset: 2 bits

Address	Tag	Set Index	Offset	Hit/Miss
0x143: 000101000011	00010100	00	11	Miss
0xc4a: 110001001010	11000100	10	10	Miss
0x22b: 001000101011	00100010	10	11	Miss
0x42f: 010000101111	01000010	11	11	Miss
0x492: 010010010010	01001001	00	10	Miss
0x2a2: 001010100010	00101010	00	10	Miss
0x3ba: 001110111010	00111011	10	10	Miss
0xb2d: 101100101101	10110010	11	01	Miss

	00			01			10			11		
Set Index	**V**	**Tag**	**Data**	**V**	**Tag**	**Data**	**V**	**Tag**	**Data**	**V**	**Tag**	**Data**
00	0			0			1	0010 1010	M[0x2a2]	1	0001 0100	M[0x143]
01	0			0			0			0		
10	0			0			1	0011 1011	M[0x3ba]	1	0010 0010	M[0x22b]
11	0			1	1011 0010	M[0xb2d]	0			1	0100 0010	M[0x42f]

- 0xc4a and 0x492 were evicted because they took the space that the new values were supposed to take. Since they were the least recently used, I decided to evict them.