Assignment 2

Due: 11:59 PM, February 27

Solve Problem 2.1, 2.3, and 2.4 in the textbook (1st edition).

2.1. When we were discussing floating point addition, we made the simplifying assumption that each of the functional units took the same amount of time. Suppose that fetch and store each take 2 nanoseconds and the remaining operations each take 1 nanosecond.

a. How long does a floating point addition take with these assumptions?

b. How long will an unpipelined addition of 1000 pairs of floats take with these assumptions?

c. How long will a pipelined addition of 1000 pairs of floats take with these assumptions?

d. The time required for fetch and store may vary considerably if the operands/results are stored in different levels of the memory hierarchy. Suppose that a fetch from a level 1 cache takes two nanoseconds, while a fetch from a level 2 cache takes five nanoseconds, and a fetch from main memory takes fifty nanoseconds. What happens to the pipeline when there is a level 1 cache miss on a fetch of one of the operands? What happens when there is a level 2 miss?

Time	Operation	Operand 1	Operand 2	Result
0	Fetch operands	9.87×10^4	6.54×10^3	
1	Compare exponents	9.87×10^4	6.54×10^3	
2	Shift one operand	9.87×10^4	0.654×10^4	
3	Add	9.87×10^4	0.654×10^4	10.524×10^4
4	Normalize result	9.87×10^4	0.654×10^4	1.0524×10^5
5	Round result	9.87×10^4	0.654×10^4	1.05×10^5
6	Store result	9.87×10^4	0.654×10^4	1.05×10^5

Table 2.3 Pipelined Addition. Numbers in the Table Are Subscripts of Operands/Results

Time	Fetch	Compare	Shift	Add	Normalize	Round	Store
0	0						
1	1	0					
2	2	1	0				
3	3	2	1	0			
4	4	3	2	1	0		
5	5	4	3	2	1	0	
6	6	5	4	3	2	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
999	999	998	997	996	995	994	993
1000		999	998	997	996	995	994
1001			999	998	997	996	995
1002				999	998	997	996
1003					999	998	997
1004						999	998
1005							999

2.3. Recall the example involving cache reads of a two-dimensional array (page 22). How does a larger matrix and a larger cache affect the performance of the two pairs of nested loops? What happens if `MAX = 8` and the cache can store four lines? How many misses occur in the reads of `A` in the first pair of nested loops? How many misses occur in the second pair?

```
double A[MAX][MAX], x[MAX], y[MAX];
. . .
/* Initialize A and x, assign y = 0 */
. . .
/* First pair of loops */
for (i = 0; i < MAX; i++)
   for (j = 0; j < MAX; j++)
      y[i] += A[i][j]*x[j];
. . .
/* Assign y = 0 */
. . .
/* Second pair of loops */
for (j = 0; j < MAX; j++)
   for (i = 0; i < MAX; i++)
      y[i] += A[i][j]*x[j];
```

To better understand this, suppose `MAX` is four, and the elements of `A` are stored in memory as follows:

Cache Line	Elements of `A`			
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]
3	A[3][0]	A[3][1]	A[3][2]	A[3][3]

2.4. In Table 2.2, virtual addresses consist of a byte offset of 12 bits and a virtual page number of 20 bits. How many pages can a program have if it's run on a system with this page size and this virtual address size?

Table 2.2 Virtual Address Divided into Virtual Page Number and Byte Offset

Virtual Address								
Virtual Page Number				Byte Offset				
31	30	⋯	13	12	11	10	⋯	1	0
1	0	⋯	1	1	0	0	⋯	1	1