CS546 “Parallel and Distributed Processing”
Homework 9

*Submission:*
*Due by 11:59pm of 11/23/2016*
*Total points 100 - Late penalty: 10% penalty for each day late*
*Please upload your assignment on Blackboard with the following name:*
*CS546_SectionNumber_LastName_FirstName_HW9.*
*Please do NOT email your assignment to the instructor and/or TA!*

1. (12 points) Consider a memory system with a level 1 cache of 32 KB and DRAM of 512 MB with the processor operating at 1 GHz. The latency to L1 cache is one cycle and the latency to DRAM is 100 cycles. In each memory cycle, the processor fetches four words (cache line size is four words). What is the peak achievable performance of a dot product of two vectors? Note: Where necessary, assume an optimal cache placement policy.

```
1        /* dot product loop */
2        for (i = 0; i < dim; i++)
3                dot_prod += a[i] * b[i];
```

Ans : 1 cycle takes 1/(1x109 ) seconds.

One access to memory takes 100/(1x$10^9$ ) seconds. =100ns.

Performance = 4 FLOPS / (4x100/(1x$10^9$ )) = 10 MFLOPS.

2. (12 points) Extending this further, consider the problem of multiplying two dense matrices of dimension 4K x 4K. What is the peak achievable performance using a three-loop dot-product based formulation? (Assume that matrices are laid out in a row-major fashion.)

```
1        /* matrix-vector product loop */
2        for (i = 0; i < dim; i++)
3                for (j = 0; i < dim; j++)
4                        c[i] += a[i][j] * b[j];
```

Ans: 5 cache lines will have to fetched, one for the matrix X and 4 for the matrix Y (since the access is performed in a column-major fashion). Thus, the peak performance = 8 FLOPs/500 ns = 16 MFLOPS.

3. (10 points) How does cache associativity affect the cache performance?

Ans: Increasing associativity decreases miss rate but with diminishing returns. Assume higher associativity would increase clock cycle time over direct mapped cache as follows: 2-way 1.36 times, 4-way 1.44 times, 8-way 1.52 times. Hit time is 1 clock cycle. Miss penalty for direct mapped cache is 25 clock cycles.

4. (10 points) Just providing deeper memory hierarchies does NOT bridge the gap between processor and memory performance. Why?

Ans: This is due to the memory wall problem. An important reason for this disparity is the limited communication bandwidth beyond chip boundaries, which is also referred to as *bandwidth wall* CPU speed improved at an annual rate of 55% while memory speed only improved at 10%. Given these trends, it was expected that memory latency would become an overwhelming bottleneck in computer performance.

5. (12 points) Derive the formula for calculating the current average access time (C-AMAT) for a word in the first level cache of a system. Assume the following values for a theoretical system containing an L1 cache.

| Location | Hit Time | Hit Concurrency | Pure Miss Rate | Pure Miss Concurrency | Pure Miss Penalty |
|---|---|---|---|---|---|
| L1 | 5ns | 3 | 35% | 5 | 1000ns |

Determine the current average access time for a memory word in the described system.

Ans: C-AMAT = HitCycle/CH+ pMR×pAMP/Cm

= 5/3+.35*1000/5=1.67+50=51.67

6. (12 points) Derive the **recursive** formula for calculating the current average access time (CAMAT) for a word in a system with three levels of cache. Assume the following values for a theoretical system containing an L1, L2, and L3 cache.

| Location | Hit Time | Hit Concurrency | Pure Miss Rate | $\eta$ |
|---|---|---|---|---|
| L1 | 5ns | 2.75 | 45% | 1.5 |
| L2 | 10ns | 3.25 | 30% | 2.25 |
| L3 | 35ns | 4.5 | 15% | 4 |
| Main Memory | 100ns | 6 | 0 | 0 |

Determine the current average access time for a memory word in the described system.

Ans: $C-AMAT = \frac{H_{L1}}{C_{H_{L1}}} + pMR_{L1} * \eta_{L1} * \left(\frac{H_{L2}}{C_{H_{L2}}} + pMR_{L2} * \eta_{L2} * \left(\frac{H_{L3}}{C_{H_{L3}}} + pMR_{L3} * \eta_{L3} * \frac{H_{Mem}}{C_{H_{Mem}}}\right)\right)$

$$= \frac{5}{2.75} + .45 * 1.5 * \left(\frac{10}{3.25} + .30 * 2.25 * \left(\frac{35}{4.5} + .15 * 4 * \frac{100}{6}\right)\right)$$

= 10.177

7. (12 points) How to optimize the following code to reduce the **miss rate** of a memory system? Please give the optimized code and explain why briefly.

```
for (i = 0; i < N; i++) {
        A[i] = B[i] + C[i];
                }
    for (i = 0; i < N; i++) {
        D[i] = B[i] + A[i];
                }
```

Ans: for(i=0;i<N;i++){

```
        A[i] = B[i] + C[i];
        D[i] = B[i] + A[i];

    }
```

By this Both B and C will be cache and hence improve locality. By improving locality this will automatically reduce miss rate.

8. (10 points) What are the hardware technologies which can increase **Hit Concurrency** in the formula of C-AMAT and explain why briefly?

Ans: 1. Peer Pure Misses : Pure misses in a shared memory component which are issued by the same processing element

2. Multi-banked caches: Dividing cache into independent blocks to support simultaneous access

3. Computing concurrence: such as multi-core, multi-thread, multi-issue also increase hit concurrence.

4. The number of pipeline stages is larger, the number of cache accesses is larger in a fixed period of time.

9. (10 points) What is the Non-block cache and what are its characteristics?

Ans: A non-blocking cache can work on other requests while waiting for memory to supply any misses. Completion buffer controls the entries of requests and ensures that departures take place in order even if loads complete out of order Split the non-blocking cache in two parts .The front end attaches tags to requests and ensures FIFO delivery of responses to the processor .The backend is a non-locking cache which can return responses out-of-order .One may merge the front end with the processor and directly expose the backend interface.

Note: We encourage collaboration between you and your classmates. Discuss various approaches and techniques to better understand the questions. However, we do NOT allow copying solutions or code. This is considered as cheating and falls under IIT code of honor. Penalties will be enforced. Please make sure you write your own solutions.
GOOD LUCK!