Diagrams or charts are not needed. Therefore it is highly recommended to type the answers. Please keep your answers organised and tidy. If you are not sure about the answers, you could look up from the Hennessy and Patterson book or directly from the Internet. If you could not find the answer, please make it clear that you are not sure.

1 Cache Memories

(A) How might a programmer improve the performance of a program given detailed knowledge of a processors memory hierarchy?

(B) A 4KB, blocking, private L1 cache with 16B lines sees the following sequence of accesses from its core.

\[
\begin{align*}
0x00001000 & \text{ Load} \\
0x00001010 & \text{ Store} \\
0x00002000 & \text{ Load} \\
0x00001010 & \text{ Load} \\
0x00003000 & \text{ Load} \\
0x00001010 & \text{ Store} \\
0x00001010 & \text{ Store} \\
0x00002000 & \text{ Load} \\
0x00001000 & \text{ Load} \\
0x00002000 & \text{ Load} \\
0x00002000 & \text{ Load}
\end{align*}
\]

(i) Assuming a write-allocate cache that is empty at first and implements the least-recently-used (LRU) replacement algorithm, what is the hit rate if the cache is:

- Direct-mapped;
- Fully-associative;
- 2-way set-associative

(ii) (Optional) If the core supports out-of-order execution, how might a non-blocking cache bring performance benefits?

(iii) (Optional) How might the cores load/store queue be used to reduce the number of memory accesses seen by the cache?

(iv) Assume that this core and cache are part of a chip-multiprocessor, with the cache connected to a shared L2 via a bus that maintains coherence through a snooping MESI protocol. What sequence of steps would be taken if another core wanted to load from 0x00001010 after the given sequence had finished?
(C) A naive programmer writes the following code for performing the matrix multiply-add function $C = AB + C$ on square matrices:

```plaintext
for (i=0; i<N; ++i) {
    for (j=0; j<N; ++j) {
        for (k=0; k<N; ++k) {
            C[k][i] = C[k][i] + ( A[k][j] * B[j][i] );
        }
    }
}
```

Where $X[v][u]$ refers to the element in row $v$, column $u$. Arrays are stored in memory row by row, i.e.

$X[0][0], X[0][1], X[0][2], \ldots X[0][N], X[1][0], \ldots$

(i) When used to multiply very large matrices, performance of the programmers algorithm is very poor. Explain what might be happening.

(ii) The algorithm can be improved simply by changing the order of the loops. Demonstrate how and why?

(iii) (Optional) Show how further improvement can be obtained through a technique known as cache blocking?

(iv) (Optional) Could the algorithm be successfully parallelised to run a on a microprocessor supporting Simultaneous Multithreading (SMT)? Briefly justify your answer.

(D) (Optional) How might a hardware prefetcher that is capable of detecting and prefetching non-unit strides be implemented?

### 2 Vector Execution

(A) Why does a vector processor offer a particularly energy efficient solution to execute some types of program?

(B) In which situations might a vector processor perform worse than a simple pipelined processor?

### 3 Cache Coherence Protocols I

(A) Why are multi-level caches often used in preference to a single larger cache? How might the parameters of an L1 and L2 data cache typically differ?

(B) Considering the three shared cache organisations listed in Figure 1, what are the benefits and downsides to each approach? Justify your answers according to the cache performance formulae in the lecture 10 (slide 7).

(C) A processors multi-level cache hierarchy consists of L1 and L2 caches with the following characteristics: L1 miss rate is 2%, L1 hit time is 2 cycles, local L2 miss rate is 20%, L2 hit time is 10 cycles, L2 miss penalty is 200 cycles. It is suggested that reducing the size of the L1 cache will improve overall performance by reducing the L1 caches hit time to a single cycle. The reduction in L1 cache size will increase the L1s miss rate to
3% 3. Will average memory access time actually be improved? Use the same formulae and show your calculations.

(D) Cache coherence protocols are classified as either invalidate or update protocols. What are the potential advantages and disadvantages of adopting an update rather than an invalidate protocol?

(E) The MESI protocol extends the MSI protocol by adding the E state. When is a cache line in state E? And what optimisation does the addition of this extra state permit?

(F) (Optional) How can the MESI cache coherency protocol be exploited to ensure that a test-and-set instruction is performed atomically without the need to lock down the bus for multiple cycles?

(G) More about cache coherence protocols will be discussed in the next supervision.

End of Problems