improvement is to add hardware floating-point arithmetic units to speed up floating-point arithmetic instructions. It is estimated that the time taken by each floating-point instruction can be reduced by a factor of 10 with the new hardware. The second improvement is to add more first-level data cache to speed up the execution of loads and stores. It is estimated that, with the same amount of additional on-chip cache real-estate as for the floating-point units, loads and stores can be speeded up by a factor of 2 over the base machine.

Let F_{fp} and F_{ls} be the fraction of execution time spent in floating-point and load/store instructions respectively. The executions of these two sets of instructions are non-overlapping in time.

(a) Using Amdahl's speedup, what should the relation be between the fractions F_{fp} and F_{ls} such that the addition of the floating-point units is better than the addition of cache space?

(b) Suppose that, instead of being given the values of fractions F_{fp} and F_{ls}, you are given the fraction of floating-point instructions and the fraction of loads and stores. You are also given the average number of cycles taken by floating-point operations and loads/stores. Can you still find out which improvement is better based on these numbers? Explain why and how. Can you still estimate the maximum speedups for each improvement using Amdahl's law? Why?

(c) What are fractions F_{fp} and F_{ls} such that a speedup of 50% (or 1.5) is achieved for each improvement deployed separately?

(d) It is decided to deploy the floating-point unit first and to add cache space later on. In the original workload, fractions F_{fp} and F_{ls} are 30% and 20%, respectively. What is the maximum speedup obtained by upgrading to the floating-point units? Assuming that this maximum speedup is achieved by the floating-point unit upgrade, what is the maximum speedup of the cache upgrade with respect to the floating-point unit upgrade?

1.2 A multiprocessor machine has 1024 processors. On this machine we map a computation in which N iterate values must be computed and then exchanged between the processors. Values are broadcast on a bus after each iteration. Each iteration proceeds in two phases. In the first phase each processor computes a subset of the N iterates. Each processor is assigned the computation of $K = N/P$ iterates, where P is the number of processors involved. In the second, communication phase each processor broadcasts its results to all other processors, one by one. Every processor waits for the end of the communication phase before starting a new computation phase. Let T_c be the time to compute one iterate and let T_b be the time to broadcast one value on the bus. We define the computation-to-communication ratio R as T_c/T_b. Note that, when $P = 1$, no communication is required.

At first, we use the premise of Amdahl's speedup (i.e., the same workload spread across an increasing number of processors). Under these conditions:

(a) Compute the speedup as a function of P and R, for $K = 1, 2, \ldots, 1024$.

(b) Compute the maximum possible speedup as a function of P and R.

(c) Compute the minimum number of processors needed to reach a speedup greater than 1 as a function of P and R.

Second, we use the premise of Gustafson's law, namely that the uniprocessor workload grows with the number of processors so that the execution time on the multiprocessor is the same as that on the uniprocessor. Assume that the uniprocessor workload computes 1024 iterates.

(d) What should the size of the workload be (as a number of iterates) when P processors are used, as a function of P and R? Pick the closest integer value for the number of iterates.

(e) Reconsider (a)–(c) above in the context of growing workload sizes, according to Gustafson's law.

Third, we now consider the overhead needed to broadcast values over the bus. Because of software and bus protocol overheads, each bus transfer requires a fixed amount of time, independent of the size of the transfer. Thus the time needed to broadcast K iterate values on the bus by each processor at the end of each iteration is now $T_0 + K \times T_b$.

(f) Using the constant workload size assumption (as in Amdahl's law), what is the maximum possible speedup?

(g) Using growing workload size assumption (as in Gustafson's law), what is the maximum possible speedup?

1.3 This problem is about the difficulty of reporting average speedup numbers for a set of programs or benchmarks. We consider four ways of reporting average speedups of several programs:

- take the ratio of average execution times, S_1;
- take the arithmetic means of speedups, S_2;
- take the harmonic means of speedups, S_3;
- take the geometric means of speedups, S_4.

Consider the two improvements of a base machine in Exercise 1.1, one improving floating-point performance and one improving memory performance. Three programs are simulated: one with no floating-point operations (Program 1), one dominated by floating-point operations (Program 2), and one with balance between memory accesses and floating-point operations (Program 3). The execution time of each program on the three machines is given in Table 1.4.

can be tested. We can use barrier synchronizations, as described in Section 5.2.1, to make this happen. Show where barrier synchronizations must be inserted for the correct execution of the parallel algorithm in a shared-memory system. Explain what problems could arise if barriers are not inserted.

(c) Assume that the matrix contains 1024×1024 elements. The computation of each element takes 10 ns, and 16 processors are available for executing the algorithm in parallel. It takes 100 ns to start a thread, and threads are started sequentially from the main thread. Assume that sequential parts such as running critical sections take 100 ns each and that ten iterations of the outer while-loop are executed before convergence. Calculate the speedup obtained.

5.2 In the design exploration of a new chip multiprocessor system an important design decision is the choice of cache organization. In Section 5.4.1 the trade-off between shared and private caches is discussed, and in this exercise we will investigate it quantitatively. Assume that a chip multiprocessor has four processor cores. Our options are between using a private or a shared, first-level cache organization, where both of them consume about the same chip resources. The detailed assumptions for each cache organization are given below. The block size for both organizations is 16 bytes.

Private cache organization Each private cache contains eight block entries, is direct-mapped, and the cache hit time is one cycle. Cache coherence across the private caches is maintained using a simple protocol, according to Section 5.4.2. The time to carry out a snooping action is ten cycles. If a block has to be retrieved from memory it takes 100 cycles.

Shared cache organization The shared cache organization has as many block entries as the total number of block entries in the private caches. Further, it is partitioned into as many banks as the number of processors, and the mapping of memory blocks to banks is round-robin; block address i is mapped to bank i modulo B, where B is the number of banks. The banks are accessed by the processors using a $B \times B$ cross-bar switch (see Chapter 6). To access a cache bank takes two cycles if there is no contention. If a block has to be retrieved from memory it takes 100 cycles.

(a) Determine the average memory-access time for each of the organizations in the case when a *single* processor sequentially accesses memory blocks 0 up to 15 twice in a row. Which organization yields the shortest memory access time, and by how much?

(b) Determine the average memory-access time for each of the organizations in the case when *each* processor sequentially accesses memory blocks 0 up to 15 twice in a row. Ignore contention effects. Which organization yields the shortest memory access time, and by how much?

(c) Number the processors $1, \ldots, 4$. Determine the average memory-access time for each of the organizations in the case when processor i sequentially accesses memory blocks $8 \times (i-1)$ up to $8 \times (i-1) + 7$ twice in a row. Ignore contention effects. Which organization yields the shortest memory access time, and by how much?

(d) Assume that memory blocks 0 to 7 are initially present in both cache organizations. Now assume that processor 1 modifies all blocks, and processor 2 reads them subsequently. Ignore contention effects. Which organization yields the shortest memory access time, and by how much?

(e) Based on your quantitative findings in the previous assignments, under what conditions does a private cache have a performance advantage over a shared cache? Why do many chip multiprocessor designs favor a private first-level organization while using a shared second- or tertiary-level cache?

5.3 Consider a shared-memory multiprocessor that consists of eight processors and a private cache associated with each. Cache coherence is maintained by a simple cache protocol according to Section 5.4.2. The bus arbitration mechanism uses a fixed priority based on the identity of the processor/cache unit; when two units with identities i and j simultaneously issue a bus request and $i < j$, unit i will get access to the bus. However, the bus arbitration mechanism "remembers" the last unit that gained access and that unit cannot get access to the bus in the next bus cycle. Now let each of the eight processors issue a write followed by a read operation to the same location, where each processor writes the same number as its unit identity to that location. What will be returned by the read request issued by each processor?

Table 5.6 **Timing and traffic parameters for protocol actions**
B is the block size

Request type	Time to carry out protocol action	Traffic
Read hit	1 cycle	N/A
Write hit	1 cycle	N/A
Read request serviced by next level	40 cycles	6 bytes + B
Read request serviced by private cache	20 cycles	6 bytes + B
Read-exclusive request serviced by next level	40 cycles	6 bytes + B
Read-exclusive request serviced by private cache	20 cycles	6 bytes + B
Bus upgrade/update request	10 cycles	10 bytes
Ownership request	10 cycles	6 bytes
Snoop action	5 cycles	N/A

5.4 Assume that a shared-memory multiprocessor using private caches connected to a shared bus uses an MSI cache protocol to maintain cache coherence. The time it takes to carry out various protocol actions is listed in Table 5.6. While a read and write hit take only a single cycle, a read request takes 40 cycles as it has to bring the block from the next level

of the cache hierarchy. A bus upgrade request takes less time as it does not involve a block transfer but rather invalidates other shared copies. This action consists of transferring the request on the bus and making a snoop action in each cache; the time for the latter is also shown in Table 5.6.

Determine for each of the cases below how long it takes to carry out the following sequence of reads and writes to blocks X and Y, where the notation Ri/B and Wi/B means a read and write operation, respectively, by processor/cache unit i to block B: R1/X, R2/X, R3/Y, R4/X, W1/X, R2/X, R3/Y, R4/X.

(a) Determine how long it takes to carry out all memory requests under the assumption that snoop actions get a higher priority than processor read/write requests from that same unit; they have to wait until the snoop action is done. The tag directory is not duplicated.

(b) Determine how long it takes to carry out the memory requests from each individual processor under the assumption that we duplicate the tag directory to allow concurrency between inbound snoop actions and outgoing processor read/write generated protocol actions. In this case, concurrent tag lookups are only possible when the state of the block in the cache does not change as a result of the snoop action.

5.5 Assume a shared-memory multiprocessor with a number of processor/private cache units connected by a shared single-transaction bus. Our baseline cache coherence protocol is an MSI protocol, but we want to investigate what performance gain can be achieved by adding an exclusive state to make it a MESI protocol according to Section 5.4.3. We want to determine the time it takes to execute a sequence of accesses with the same assumptions and notations as in Exercise 5.4 with an MSI and with a MESI protocol. Consider the following sequence of accesses by the processors:

R1/X, W1/X, W1/X, R2/X, W2/X, W2/X, R3/X, W3/X, W3/X, R4/X, W4/X, W4/X.

(a) Now suppose that a transition from state E to state M brings no access cost. How many cycles does it take to execute the access sequence under MSI vs. MESI, assuming the access costs for the protocol transactions to be as in Table 5.6?

(b) Compare the traffic generated by the MSI and MESI protocols counted in bytes transferred using the data in Table 5.6, and assuming that B is 32 bytes.

5.6 Assume a shared-memory multiprocessor with a number of processor/private cache units connected by a shared single-transaction bus. Our baseline cache coherence protocol is an MSI protocol, but we want to investigate what performance gain can be achieved by adding an Ownership state to make it a MOESI protocol according to Section 5.4.3. We want to determine the time and traffic under the execution of a sequence of accesses with an MSI and with a MOESI protocol by using the parameters in Table 5.6. Consider the following sequence of accesses by the processors:

R1/X, W1/X, W1/X, R2/X, W2/X, W2/X, R1/X, W1/X, W1/1X, R2/X, W2/X, W2/X.

(c) We can unroll the vector loop twice (there are enough vector registers). Show the code for the dot-product and calculate the number of clocks needed to compute the multiplication of two 1024×1024 matrices (neglect the final scalar phases).

3.27 In this chapter, we have assumed a simple interleaved memory system for a vector processor, with four banks, a bank access latency of 4, and a stride of 1. However, in practice there will be more banks, and moreover the stride may vary to access complex structures.

(a) Assume 32 banks, a bank access time of 8 clocks, and one vector load or one vector store at a time. What are the permissible vector strides (from 1 to 32) that will avoid conflict, using the simple interleaved scheme in Figure 3.40. Can you generalize this result to any stride by considering the stride modulo 32?

(b) Assume now 31 banks, a bank access time of 8 clocks and one vector load or one vector store at a time. What are the permissible vector strides (from 1 to 31) that will avoid conflict, using the simple interleaved scheme in Figure 3.40. Can you generalize this result to any stride by considering the stride modulo 31?

(c) Discuss the advantages and inconveniences in using 31 banks rather than 32 banks.

3.28 Not all vector machines are load/store. In fact, the first vector machine, the CDC Star-100 (built at the beginning of the 1970s by a now defunct company called Control Data Corporation) was a pure memory-to-memory vector machine. Input vectors were streamed from memory directly into pipelined units and the results were streamed directly back

2019 Re-exam Problem 3

A design team needs to choose an appropriate cache coherence protocol to be used for a shared memory multiprocessor with a number of processor/private cache units connected by a shared single-transaction bus. The team is considering three invalidation-based snoopy protocols: MSI, MESI and MOESI.

In order to select the protocol, the team is analyzing the following representative sequence of accesses happening on the bus in the following order:

R1/X, R2/X, R2/Y, W1/X, W2/Y

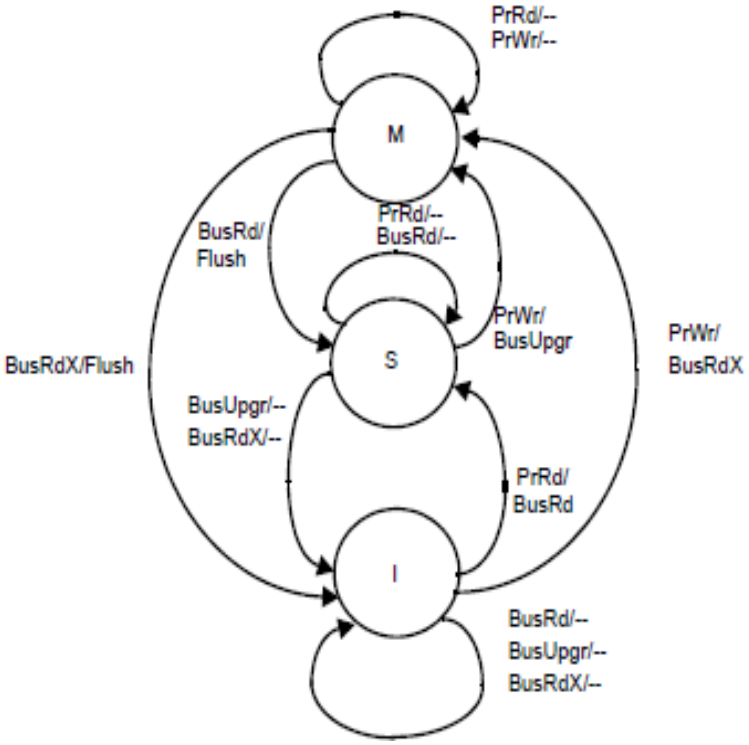
where the notation R*i*/X, W*i*/X means a read and write from processor *i* to cache block X. The latency and traffic costs of different operations considering a block size of B bytes are:

Operation	**Latency**	**Bus Traffic**
Read hit	1	N/A
Write hit	1	N/A
Request serviced by next level (BusRd,Flush)	100	6+B
Request serviced by different cache (BusRd/Flush)	20	6+B
Bus upgrade (BusUpgr)	10	10
Snoop action	5	N/A

Assumptions:

1. A bus upgrade consists of transferring the request on the bus and making a snoop action in each cache (the latency of the latter is shown in the table)
2. There is no contention on the tag directory (duplicated tag directory)
3. X and Y are not the same cache block
4. The block size B is 32 bytes
5. Read-exclusive requests (BusRdX) have the same latency and traffic as Read requests

The MSI state-transition diagram is shown below



To assess the performance of the three protocols the team must construct a table with the latency (cycles) and bus traffic (bytes) needed to complete the aforementioned access sequence (shown below).

Protocol	Latency (cycles)	Traffic (bytes)
MSI		
MESI		
MOESI		

Your task is as follows:

(a) Describe in 1-2 sentences the meaning of the states ‘E’ and ‘O’. What problems do they address? (b) Show the protocol transitions in each cache for the lines X and Y
(c) Complete the table with latency and traffic values