Write your name, surname, student identification number (numero di matricola), e-mail. The answers can be written in English or in Italian. Please, present the work in a legible and readable form. All the answers must be properly and clearly explained, focusing on the requested issues.

1. A system \( \Sigma = (P_1, P_2, P_3) \) is composed of three processes executing the following actions on two shared integer variables \( x \) and \( y \) both initialized to zero:

\[
\begin{align*}
P_1 &:: \{ \text{Write}(x, 1); \text{Read}(x, 1); \text{Read}(y, 0); \} \\
P_2 &:: \{ \text{Write}(y, 1); \text{Read}(y, 1); \text{Read}(x, 1); \} \\
P_3 &:: \{ \text{Read}(y, ?); \text{Read}(x, ?); \}
\end{align*}
\]

The notation \( \text{Write}(x, v) \) corresponds to a code writing the value \( v \) in the variable \( x \), and \( \text{Read}(x, v) \) is a code reading the value of \( x \) and returning the value \( v \). Suppose that the computation is executed on a CMP architecture respecting the Sequential Consistency memory model.

(a) Describe in detail the general definition of the Sequential Consistency memory model and its properties through same basic examples;

(b) Related to system \( \Sigma \), explain which are the possible and the forbidden outcomes of the execution of \( P_3 \): i.e. the admissible values of \( y \) and \( x \) seen by the third process (question marks in the code).

2. A unidirectional ring network connects \( N > 1 \) nodes through switch units each connected to a corresponding node. Suppose that the network is emulated at the processes level by expressing the highest bandwidth as possible, and assume that processes cannot overlap the inter-process communication latency with their internal calculation (no communication processor is available).

(a) Show the emulation structure with particular attention to the design of the switch module in order to maximize the number of messages that can be routed in parallel;

(b) Provide a formal proof of the switch throughput in the average case. Hint: use basic results of Graph Analysis to study the switch decomposition in terms of interconnected processes.

3. A process \( P \) receives a stream of arrays of size \( M \) integers and encapsulates an array \( B \) of the same size statically initialized. The computation is described by the following pseudo-code:

\[
P:: \{ ... \text{while true do} \text{receive}(\text{ch\_in}, A); \text{sum} = 0; \text{for} \ i=0 \text{ to} \ M-1 \text{ do} \ B[i] = F(A[i], B[i]); \text{send}(\text{ch\_out}, \text{op}(B)); \}
\]

The notation \( \text{op}(B) \) denotes the application of a commutative and associative binary operator \( \oplus \) on all the elements of the array \( B \). The process is executed in a multi-CMP SMP architecture with directory-based cache coherence with home flushing semantics.

(a) Assume that the process is a bottleneck and describe a feasible parallelization of the problem;

(b) Discuss the parallel semantics of the reduce phase, i.e. whether it is synchronous or asynchronous with respect to the other computation phases;

(c) Present two different ways to parallelize the reduce phase by making different assumptions on the ratio \( T_F/T_B \) (i.e. the average processing time of the function \( F \) and of the binary operator \( \oplus \) respectively) and the optimal parallelism degree \( n > 1 \) of the computation.
Solution

1. The Sequential Consistency memory model was introduced by Lamport. It consists in a very rigid memory model for multiprocessors that prevents many of the possible optimizations to hide memory access latencies that are actually a common practice in existing architectures.

   The definition states that in a Sequential Consistent architecture all the PEs see the very same ordering of memory accesses. Furthermore, in this ordering the memory accesses executed by the same PE must appear in the program order while accesses by different PEs can be mixed in any possible way. As an example, suppose that a processor P1 executes two accesses (A; B) and the second processor P2 executes accesses (C; D). The model guarantees that both the processors will see the same ordering of the four memory accesses (STOREs and LOADs). This unique ordering is one of these six possible outcomes:

   \[
   \begin{array}{cccc}
   A & A & A & C \\
   B & C & C & D \\
   C & B & D & A \\
   D & D & B & B \\
   \end{array}
   \]

   For example, the ordering BACD is not admissible because A and B must be completed and made visible to all the processors in the program order (A; B).

   According with the previous definition, it is easy to understand which are the possible outcomes expected by executing Σ in a Sequential Consistent architecture. From the code of P1, we observe that the code WRITE(x, 1) must be executed and completed (made visible) before the instruction READ(y, 0). This means that when the first process reads the value of the variable y, the code WRITE(y, 1) executed by P2 is not complete and visible yet. Therefore, we have:

   \[
   \text{READ}(y, 0) \rightarrow \text{WRITE}(y, 1)
   \]

   Where A->B indicates that access A precedes B in the ordering. Since in the Sequential Consistent memory model the instructions are completed and made visible in the program order, we also know that:

   \[
   \text{WRITE}(x, 1) \rightarrow \text{READ}(y, 0)
   \]

   and by transitive property we have \(\text{WRITE}(x, 1) \rightarrow \text{WRITE}(y, 1)\). Therefore, the STORE instruction writing the value 1 on x must precede the one writing 1 on y. This allows us to conclude that the possible outcomes in the P3’s execution are:

   1. P3 finds \(y=0\) and \(x=0\) (both the two STOREs are not visible to P3 yet);
   2. P3 finds \(y=0\) and \(x=1\) (the first STORE by P1 is visible to P3 while the one by P2 not yet);
   3. P3 finds \(y=1\) and \(x=1\) (both the STOREs are visible to P3).

   The outcome \(y=1\) and \(x=0\) is not admissible in a Sequential Consistent machine: if the STORE instruction updating y to 1 by P2 is visible to P3, for sure also the STORE by P1 updating x with the value 1 must be visible.

2. The emulation consists in a set of processes emulating the nodes \(P_0, ..., P_{N-1}\), each connected to a switch module \(SW_0, ..., SW_{N-1}\) as in the figure below:

   Each switch \(SW_i\) module listens non-deterministically from two input channels, one from process \(P_i\) and one from the switch \(SW_{(i+N-1)\%N}\). In order to maximize the bandwidth of the switch, it needs to be decomposed in a network of processes properly interconnected. To understand the problem, suppose that each switch is implemented by a single process. If a switch process finds simultaneously two messages in
the two input channels, according to the semantics of the alternative command, only one of the two guards is executed and in the corresponding code the message is sent to the proper destination. Therefore, the switch process is not able to send in parallel more than one message unless proper architectural supports are provided (in the exercise we assume that no communication processor exists).

To solve the problem, each switch module is further decomposed into two internal switch processes as in the figure below:

![Switch Module Diagram]

In case of a message coming from the previous switch module in the ring, it can be routed to the process \( P_i \) with probability \( p \) or, if its destination is a next node in the ring, it is routed to process \( S_2 \) in charge of forwarding it to the next switch module. In case of a message coming from the process \( P_i \), this must be forwarded to the next switch module only. Therefore, \( S_2 \) works non-deterministically (alternative command).

In the average case, the probability distribution that a message coming from \( SW_{i-1} \) is routed either to \( P_i \) or to \( SW_{i+1} \) is the uniform one, i.e. \( p = 0.5 \). If we assume that the inter-arrival time from each input channel to \( SW_i \) is the minimum one (i.e. at the processes level this lower bound is the inter-process communication latency \( L_{com} \)), the inter-departure time from \( S_1 \) to \( P_i \) is:

\[
T_{D-P_i} = \frac{L_{com}}{p} = 2 \cdot L_{com}
\]

Conceptually, \( S_1 \) is not a bottleneck because its inter-arrival time \( L_{com} \) is equal to its ideal service time (which consists in the send primitive only, i.e. \( L_{com} \)), thus its utilization factor is \( \rho = 1 \). The inter-arrival time to process \( S_2 \) is:

\[
T_{A-2} = \frac{1}{2 \cdot L_{com} + L_{com}} = \frac{2}{3} \cdot L_{com}
\]

The second switch process \( S_2 \) is a bottleneck with \( \rho = \frac{3}{2} = 1.5 \), so its inter-departure time to the next switch module is \( L_{com} \). The total inter-departure time from the switch module is the inverse of the total output bandwidth from the system (sum of the departure rates from \( S_1 \rightarrow P_i \) and \( S_2 \rightarrow SW_{i+1} \)):

\[
T_{D-SW} = \frac{1}{2 \cdot L_{com} + L_{com}} = \frac{2}{3} \cdot L_{com}
\]

So, the total bandwidth of the switch module is:

\[
B_{SW} = \frac{1}{T_{D-SW}} = 1.5 \cdot L_{com}
\]

As expected, the average number of messages that the switch module is able to forward per service time is \( 1.5 \).

3. The computation must be parallelized with a data-parallel approach since the internal array \( B \) is modifiable. At the Virtual Processor level, we can recognize \( M \) virtual processors, each one encapsulating one element of \( A \) and one element of \( B \):

\[
VP_{i=0,...,M-1} = \{A[i],B[i]\}
\]

According with this definition, the computation is a map followed by a reduce phase. Passing to the actual version with real workers, one process is responsible for scattering each input array to \( n \) workers,
while the internal array $B$ is statically partitioned. The reduce phase is fully asynchronous w.r.t the map: i.e. the workers can start working on the next stream element without waiting for the completion of the global reduce phase.

Depending on the ratio $T_F/T_\oplus$, we can design three possible implementations:

i. in case the ratio is very high, the computation cost of the function is much greater than the one of the binary reduce operator. Therefore, the impact of the reduce could very small or even negligible. A simple implementation consists in a scatter process, $n$ workers, and a reduce process. The workers apply the map phase on their partition by reading their partitions of $A$ and $B$ and updating their partitions of $B$. Then, each worker computes the local reduce result on its partition of $B$. To compute the global reduce result, each worker transmits to the reduce process its local result and the reduce process is in charge of computing the global result, with ideal service time approximately equal to $(n - 1) \cdot T_\oplus$;

ii. in case the ratio is relatively small and if the parallelism degree is relatively high, it might be possible that the reduce process becomes a bottleneck and a centralized reduce is no longer a valid solution to optimize throughput. In that case, the reduce can be implemented by the workers themselves by exchanging local reduce results through a binary tree mapped on the workers. The communication grain is small (one word is the message exchanged by workers for the reduce) and it is likely to be overlapped with the workers’ internal computation. However, the latency of the reduce impacts on the workers’ ideal service time, in particular on the ideal service time of the slowest worker, i.e. the one in charge of producing the final reduce result. The latency of the reduce is in the order of $\log n \cdot (T_{send} (1) + T_\oplus)$, thus smaller than the one of the centralized reduced. The worker ideal service time (of the slowest worker) can be approximated as:

$$T_{id-W} \cong \frac{M}{n} \cdot (T_F + T_\oplus) + \log n \cdot (T_{send} (1) + T_\oplus)$$

Since the latency of the reduce is an additive factor of the worker ideal service time, and since $n$ is the optimal parallelism degree, it might be possible that the condition $T_{id-W} \leq T_A$ is still valid, and the parallelization is still able to eliminate the bottleneck, or that $T_{id-W} > T_A$ and the parallelization slightly remains a bottleneck due to the reduce overhead paid by the workers.