

Exam

NAME:

Instructions: Write your solutions with a pen *clearly* and *succinctly* directly on this booklet. You may not use extra resources such as books, notepads or portable electronic devices.

Please sign below to confirm that you understand that, according to the regulations of the Faculty of Informatics, violation of the above instructions can lead to you failing this exam.

Signature: _____

Grades		
Sec. #	you got	out of
1		6
2		12
3		8
4		7
5		4
total		37

Choose the correct answer (only one is correct):

/12

1 OpenMP

Questions	Answers
1. OpenMP is	<ul style="list-style-type: none"><input type="checkbox"/> a programming language<input type="checkbox"/> a parallelization library<input checked="" type="checkbox"/> a set of compiler directives accompanied with a runtime library that allows you to easily parallelize a code on shared memory architectures<input type="checkbox"/> a utility that allows you to parallelize your code on distributed memory architectures
2. Assuming OMP_NUM_THREADS=8, when an OpenMP parallel region (“omp parallel”) is encountered by the master thread,	<ul style="list-style-type: none"><input checked="" type="checkbox"/> a set of 7 additional threads will be created and will execute in parallel the following block of code along with the master thread<input type="checkbox"/> a set of 8 additional threads will be created and will execute in parallel the following block of code along with the master thread<input type="checkbox"/> a set of as many threads as the number of available cores will be created and will execute in parallel the following block of code<input type="checkbox"/> no threads will be created until an “omp for” directive is encountered
3. If an “omp barrier” directive is encountered in an OpenMP parallel region	<ul style="list-style-type: none"><input type="checkbox"/> a single random thread will execute the next block of code<input type="checkbox"/> all the threads will execute the next block of code one after the other sequentially<input type="checkbox"/> the master thread only will execute the following block of code<input checked="" type="checkbox"/> the execution will not proceed further until all the threads of the parallel region reach the barrier
4. If a variable is declared as “firstprivate” in an OpenMP parallel region, this means that	<ul style="list-style-type: none"><input type="checkbox"/> a copy of the variable will be placed by the compiler in the L2 cache of each core<input checked="" type="checkbox"/> the variable will be copied from the master thread to each individual thread of the parallel region<input type="checkbox"/> a variable of the same type will be created in each individual thread of the parallel region<input type="checkbox"/> the variable will be private for the first subsequent for loop but shared for the rest

Questions	Answers
<p>5. When an “omp for” / "omp do" directive precedes a for / do loop, then</p>	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> the iteration space of the following for / do loop will be partitioned across all the threads of the current OpenMP parallel region <input type="checkbox"/> a new thread will be created for each iteration of the loop <input type="checkbox"/> you can safely assume that the first iteration of the loop will always be executed by the master thread <input type="checkbox"/> each thread of the parallel region will execute the whole loop
<p>6. The Amdahl's law states that the execution time $T(N)$ of a parallel application with N threads</p>	<ul style="list-style-type: none"> <input type="checkbox"/> will always be equal to $T(1)/T(N)$, where $T(1)$ is the execution time with a single thread <input type="checkbox"/> will be proportional to the time it takes for a main memory request to be served <input checked="" type="checkbox"/> will always be bound by the fraction of the application that remained strictly serial <input type="checkbox"/> cannot be predicted

2 MPI

Questions	Answers
1. MPI is useful	<input type="checkbox"/> only for shared memory systems <input type="checkbox"/> only for distributed memory systems <input checked="" type="checkbox"/> both for shared memory and distributed memory systems <input type="checkbox"/> only for Cray systems
2. What is the last MPI routine that all processes must call in an MPI program?	<input type="checkbox"/> MPI_Abort <input checked="" type="checkbox"/> MPI_Finalize <input type="checkbox"/> MPI_Init <input type="checkbox"/> MPI_Initialized
3. When an MPI_Send routine returns, which of the following is always true?	<input type="checkbox"/> The receiver has received all of the message <input checked="" type="checkbox"/> The variable that has been sent (the send buffer) is safe to be modified by the caller <input type="checkbox"/> The send was a "synchronous" send <input type="checkbox"/> The receiver must have called MPI_Recv
4. The MPI_Send routine	<input type="checkbox"/> is always synchronous <input type="checkbox"/> is always buffered <input checked="" type="checkbox"/> might be synchronous or buffered, depending on the implementation <input type="checkbox"/> is an example of non-blocking communication
5. A message can be sent and will be received	<input type="checkbox"/> between different communicators if the sender and receiver ranks are the same in both communicators <input type="checkbox"/> between different communicators if the sender and receiver tags are the same for both communicators <input type="checkbox"/> between different communicators if the sender and receiver tags are the same and the ranks are the same in each communicator <input checked="" type="checkbox"/> only within the same communicator
6. Wildcarding allows	<input type="checkbox"/> the sender to broadcast a messages to all processes instead of just one <input type="checkbox"/> the sender to send a message to the next receiving process <input checked="" type="checkbox"/> the receiver to receive from any source or with any tag <input type="checkbox"/> two messages sent from one rank to another rank to overtake each other

Questions	Answers
7. With which MPI routine can I determine my left and right neighbours in a Cartesian topology?	<input type="checkbox"/> MPI_Cart_coords <input type="checkbox"/> MPI_Cart_create <input checked="" type="checkbox"/> MPI_Cart_shift <input type="checkbox"/> MPI_Cart_create
8. If I have a total of n processes, what function can I call that will create a sensible division of these processors in a Cartesian grid?	<input checked="" type="checkbox"/> MPI_Dims_create <input type="checkbox"/> MPI_Cart_rank <input type="checkbox"/> MPI_Cart_coords <input type="checkbox"/> MPI_Graph_create
9. All MPI programs must contain a call to MPI_Init. This routine must be called before all other MPI routines, with one exception:	<input type="checkbox"/> MPI_Abort <input checked="" type="checkbox"/> MPI_Initialized <input type="checkbox"/> MPI_Finalize <input type="checkbox"/> MPI_Comm_size
10. MPI_Reduce will	<input type="checkbox"/> send chunks of an array from the root process to other processes <input type="checkbox"/> send a single piece of data from the root process to all other processes <input checked="" type="checkbox"/> compute an operation on elements from different processes and store the result on the root process <input type="checkbox"/> compute an operation on elements from different processes and store the result on all processes
11. If you are using MPI with OpenMP, how should you initialize MPI?	<input type="checkbox"/> MPI_Init <input checked="" type="checkbox"/> MPI_Init_thread <input type="checkbox"/> MPI_Funneled <input type="checkbox"/> MPI_Initialize_OpenMP
12. MPI_Barrier:	<input checked="" type="checkbox"/> Stops all MPI processes in a communicator until all reach the barrier <input type="checkbox"/> Completes all communications within a communicator <input type="checkbox"/> Ensures it is safe to use a sent buffer <input type="checkbox"/> Finalizes an MPI session

3 I/O

Choose the correct answer (only one is correct):

/8

Questions	Answers
1. Parallelizing I/O	<ul style="list-style-type: none"><input type="checkbox"/> Will lead to perfect scalability of the code<input checked="" type="checkbox"/> Helps reducing the total computational time<input type="checkbox"/> Does not matter, it is always a minor part of the execution<input type="checkbox"/> It is not possible
2. Writing files in parallel	<ul style="list-style-type: none"><input type="checkbox"/> Is possible only using MPI<input type="checkbox"/> Always improves performance, independently from the file system<input checked="" type="checkbox"/> Is always safe if implemented with MPI<input type="checkbox"/> Always improves performance, if implemented with MPI
3. A binary file written with MPI	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Can be read by a non MPI program<input type="checkbox"/> Can be read only using the same number of MPI tasks it was written with<input type="checkbox"/> Follows a precise standard<input type="checkbox"/> Can be read only by a C/C++ program
4. MPI non blocking I/O	<ul style="list-style-type: none"><input type="checkbox"/> Is not supported by MPI<input type="checkbox"/> Cannot be combined to collective I/O<input type="checkbox"/> Is supported since it avoids deadlocks<input checked="" type="checkbox"/> Can improve the performance by overlapping I/O to computation
5. MPI collective I/O	<ul style="list-style-type: none"><input type="checkbox"/> Improves performance because all MPI tasks can write at the same time to disk<input type="checkbox"/> Improves performance since only one MPI task write files<input type="checkbox"/> Improves performance since MPI tasks do not communicate<input checked="" type="checkbox"/> Improves performance since the number I/O operations is optimized
6. HDF5 files	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Are portable but needs HDF5 library to be read<input type="checkbox"/> Are portable and can be easily read by any program<input type="checkbox"/> Are portable and can be easily read only by Fortran programs<input type="checkbox"/> Are portable and their size is always smaller than any other file format

Questions	Answers
7. HDF5 provides	<input type="checkbox"/> An OpenMP based library to do parallel I/O <input checked="" type="checkbox"/> An MPI based library to do parallel I/O <input type="checkbox"/> Does not support parallel I/O <input type="checkbox"/> A CUDA based library to do parallel I/O
8. Strong scalability test:	<input type="checkbox"/> Measures the scalability of a MPI code <input type="checkbox"/> Measures how performance changes increasing the problem size together with the number of parallel tasks <input type="checkbox"/> Measures how performance changes increasing the problem size and keeping the number of parallel tasks the same <input checked="" type="checkbox"/> Measures the scalability of a parallel code for a specific problem size with increasing the number of parallel tasks

4 CUDA

Questions	Answers
1. In CUDA terminology, when a kernel is launched, the set of threads that run on the same SMX is called a	<input type="checkbox"/> thread grid <input type="checkbox"/> thread set <input type="checkbox"/> thread gang <input checked="" type="checkbox"/> thread block
2. A reduction between n threads in the same thread block can be optimally performed with complexity	<input type="checkbox"/> constant $O(1)$ <input type="checkbox"/> linear $O(n)$ <input checked="" type="checkbox"/> logarithmic $O(\log(n))$ <input type="checkbox"/> quadratic $O(n^2)$
3. If host and gpu access managed memory simultaneously the following occurs	<input type="checkbox"/> a segmentation fault <input type="checkbox"/> performance slowdown <input checked="" type="checkbox"/> undefined behavior <input type="checkbox"/> managed memory ensures ensures that all reads and writes are correctly ordered
4. To make the host code wait for all kernels to finish before proceeding call	<input type="checkbox"/> __syncthreads() <input checked="" type="checkbox"/> cudaDeviceSynchronize() <input type="checkbox"/> MPI_Barrier() <input type="checkbox"/> nothing: kernel launches return when the kernel has finished
5. A race condition may occur when	<input checked="" type="checkbox"/> multiple threads access an address with at least one write <input type="checkbox"/> multiple threads read the same location in device memory <input type="checkbox"/> host and device write to the same managed memory <input type="checkbox"/> all of the above
6. Thread A can reliably read a memory update by thread B if	<input type="checkbox"/> thread B was in the same thread block and __syncthreads() has been called <input type="checkbox"/> thread A and thread B both access the memory using atomics (e.g. atomicAdd()) <input type="checkbox"/> thread B was in another kernel that finished before thread A's kernel launched <input checked="" type="checkbox"/> all of the above
7. Memory reads for a kernel are fastest from	<input type="checkbox"/> host memory <input checked="" type="checkbox"/> shared memory <input type="checkbox"/> hard disk <input type="checkbox"/> device memory

5 OpenACC

Questions	Answers
1. Currently, OpenACC	<ul style="list-style-type: none"><input type="checkbox"/> Is supported only by NVIDIA<input checked="" type="checkbox"/> Is supported by various compilers<input type="checkbox"/> Is supported only by PGI<input type="checkbox"/> Is supported only on CRAY systems
2. When you declare a data region	<ul style="list-style-type: none"><input checked="" type="checkbox"/> You have to ensure that the needed arrays are properly created and initialized on the GPU<input type="checkbox"/> You are sure that the compiler will copy all the needed arrays to the GPU<input type="checkbox"/> All the instructions until the end of the data region will be executed on the GPU<input type="checkbox"/> The compiler creates a CUDA kernel
3. The “parallel” directive	<ul style="list-style-type: none"><input type="checkbox"/> Executes the first following loop on the GPU assigning one iteration per thread<input type="checkbox"/> Parallelizes the first loop that follows on the CPU<input type="checkbox"/> Distributes the work that follows among all the available threads<input checked="" type="checkbox"/> Creates a kernel that is executed by all gangs redundantly
4. Assuming A is an array of 128 real number elements, then the directive “acc data create(A[0:128])” in C or “acc data create(A)” in Fortran	<ul style="list-style-type: none"><input type="checkbox"/> Will allocate an array of 128 “real” elements on the main memory of the CPU (host)<input checked="" type="checkbox"/> Will allocate an array of 128 “real” elements on the main memory of the GPU<input type="checkbox"/> Will allocate an array of 128 “real” elements on the shared memory of the GPU<input type="checkbox"/> Will allocate an array of 128 “real” elements on the main memory of the GPU and copy the values stored to the CPU