

CSCI-GA.3033-009
Multicore Processors: Architecture & Programming
 Homework Assignment1

1.

Type of Parallelism	Definition	Needs Programmer Help
Instruction Level parallelism	Executing several instructions in parallel, either through time (pipelining) or space (superscalar and VLIW)	No, hardware (superscalar and pipelining) and compiler (VLIW) exploit it.
Data Level Parallelism	Executing the same instruction (SIMD) or the same task/thread on different data items	Yes, GPUs are a clear example of that; although compiler can make use of vector instructions (vector processors).
Task Level Parallelism	Executing several threads/processes at the same time (MIMD)	Yes!

2. Multiprocessor systems have been around for several decades now. This means we should already have good experience dealing with parallel systems. Yet, we are facing challenges dealing with multicore processors.

- a) List all the differences you can think of between traditional multiprocessor systems and the current multicore processors.

There are several differences here. The major ones are:

- **Latency of communication among cores is much lower than in traditional multiprocessor systems. This gives much higher bandwidth on-chip.**
 - **Traditional multiprocessor systems were designed mainly for some niche, mostly scientific, applications. The current multicore processors are expected to execute those niche applications as well as traditional day-to-day applications.**
 - **Power density is a major problem in multicore processors.**
 - **The area is a very scarce resource in multicore processors. This means that some interconnections, that may be feasible in traditional multiprocessor systems, may not be feasible here.**
 - **Off-chip bandwidth is very severe for multicore processors.**
- b) List one or more cases where our expertise with traditional multiprocessor systems is helpful in dealing with multicore processors.
- **Writing programs for scientific applications on a multicore processor benefits a lot from our expertise in high performance computing.**
 - **Our expertise with consistency models and coherence are also applicable here (more on that later in the course).**
- c) List one or more cases where our expertise with traditional multiprocessor systems is NOT helpful in dealing with multicore processors.

- **Writing code for some day-to-day, non-scientific, applications such as an OS kernel.**

3. What do you think is/are the factor(s) that can make an application very hard (or sometimes impossible) to parallelize?

There are many reasons. The most important ones:

- **The instructions, or threads, of the application are strictly dependent on each other; and hence have to be executed sequentially.**
- **Very complicated control flow**
- **Extensive use of pointers (prohibits compiler from making many optimizations).**

4. If you are given a sequential program that you are required to parallelize, first you need to find the parts that are *parallelizable*. However, in some cases, it is not worth it to parallelize those parts, why?

This happens when those parts are seldom executed. If a part (a part can be a procedure, loop body, group of procedures, etc) is only 2%, for example, of the total execution time of the sequential version, then it is not worth the effort to parallelize it. This is why it is very important to *profile* the sequential application to see which parts are worth concentrating on.