6.1

-Pipelining: Individual instructions are executed through a pipeline of stages so that while one instruction is executing in one stage of the pipeline, another instruction is executing in another stage of the pipeline.

-Superscalar: Multiple pipelines are constructed by replicating execution resources. This enables parallel execution of instructions in parallel pipelines, so long as hazards are avoided.

-Simultaneous multithreading (SMT): Register banks are replicated so that multiple threads can share the use of pipeline resources.

6.2:

In the case of pipelining, simple 3-stage pipelines were replaced by pipelines with 5 stages, and then many more stages, with some implementations having over a dozen stages. There is a practical limit to how far this trend can be taken, because with more stages, there is the need for more logic, more interconnections, and more control signals.

With superscalar organization, performance increases can be achieved by increasing the number of parallel pipelines. Again, there are diminishing returns as the number of pipelines increases. More logic is required to manage hazards and to stage instruction resources.

Eventually, a single thread of execution reaches the point where hazards and resource dependencies prevent the full use of the multiple pipelines available. This same point of diminishing returns is reached with SMT, as the complexity of managing multiple threads over a set of pipelines limits the number of threads and number of pipelines that can be effectively utilized.

6.3

The growing trend towards giving an increasing fraction of chip area to cache memory is mainly due to the fact that Cache memory uses less power than logic. This makes it an efficient option to use in chip design.

6.4:

The main design variables in a multicore organization are:

-The number of core processors on the chip.

-The number of levels of cache memory.

-The amount of cache memory that is shared.

6.6

Constructive interference can reduce overall miss rates. That is, if a thread on one core accesses a main memory location, this brings the frame containing the referenced location into the shared cache. If a thread on another core soon thereafter accesses the same memory block, the memory locations will already be available in the shared on-chip cache.

A related advantage is that data shared by multiple cores is not replicated at the shared cache level.

With proper frame replacement algorithms, the amount of shared cache allocated to each core is dynamic, so that threads that have a less locality can employ more cache.

Interprocessor communication is easy to implement, via shared memory locations.

The use of a shared L2 cache confines the cache coherency problem to the L1 cache level, which may provide some additional performance advantage.