Module 12 Example Set 2

1. What are chip multiprocessors (CMPs) and how do they differ from multi-threaded systems?
Chip multi-processors (CMPs) are also known as multi-core processors. They are examples of one version of multi-threaded systems. CMPs can execute instructions from multiple threads at the same time, but on distinct cores. No execution units are shared among the threads. Each core has a separate L1 cache and separate execution units but may share the L2 or L3 cache and the memory system.

3. How do multi-core systems differ from multi-processor systems?
Each processor within a multi-processor system is implemented on a separate chip and employs separate caches while interfacing with other processors through some type of interconnect system that allows the separate processors to share memory. The cores within a multi-core system are on the same chip and tend to be simpler in nature than the multi-processor's more complex superscalar processors that provide dynamic instruction scheduling. Each core usually has a separate L1 cache but may share the L2 and L3 caches with other cores but the processors within a multi-processor system would have separate caches that must be kept consistent or coherent.

4. What are some of the factors that have prompted the development and use of multi-core processors?
The management of reservation stations and the dynamic scheduling of instructions within superscalar systems together with the control of very deep pipelines (20 or more stages) require the use of a vast number of transistors which when operated at extremely high clock rates consume large amounts of energy and produce a lot of heat which becomes more of a problem to handle. Also the design time and test/verification time for such systems adds dramatically to their cost. So, the use of multiple cores, each of which has a much simpler design and runs at a lower clock rate, has become viewed as a better and more cost effective alternative to complex powerful single core systems. It has also been observed that most programs often do not have enough instructions that can be executed in parallel due to various types of dependencies and due to cache misses which force accesses to the relatively slow memory. So the sophisticated superscalar systems are often starved for work to do and many of their functional units tend to sit idle. Implementing multiple cores on a single chip often takes no more chip area or real estate than a single complex superscalar processor.

5. Describe a typical application of a multi-core system.
One example is a web server system that needs to handle a large number of requests over a network. The individual requests, whether for web pages, database access or file service, are typically independent tasks that can be handled by threads spread across separate cores or processors.

6. How does the type of parallelism exploited by multi-core systems differ from that exploited by superscalar systems?
Each of the separate cores within a multi-core system executes a separate thread within a program or from different programs at the same time; this is called thread-level parallelism (TLP). Superscalar systems, on the other hand, dynamically examine the instructions within a single program or thread and try to identify multiple instructions that can execute at the same time; this is called instruction level parallelism (ILP). The superscalar systems often execute the instructions out of order (i.e., in a different order than they appear within the program), but insure that the instructions write their results in the proper order using a re-order buffer (ROB).

7. How are a processor's clock rate and voltage level related to its power consumption?
The power consumed is proportional to the clock rate and to the square of the voltage level used.
Systems that run at lower clock rates can usually use lower voltage levels as well. For example, if cutting the clock rate in half also allows the voltage to be cut in half, then the overall power consumption will be cut by a factor of 8 (since ½ * ¼ = 1/8).

8. Do multi-core processors tend to execute programs at a faster rate than superscalar uniprocessors running at comparable clock rates?
No, the individual cores usually do not execute as many instructions in parallel as the heavyweight superscalar systems. However, the multi-core systems tend to greatly increase the number of threads or tasks that can be completed per unit time (i.e., the throughput) and provide a higher performance per watt advantage.

9. How does a vector processor differ from a scalar processor?
A vector processor contains vector registers and has the ability to perform the same operation on the elements in two vector registers at the same time.
Scalar processors can only apply one operation at a time using two scalar (i.e., single element) operands as inputs.

10. What are the options for loading vector registers?
One option is to load all elements within the vector registers before the operation begins, the other to employ "chaining" in which the first element in each input vector register is loaded and while each subsequent element is loaded, the operation is applied to the previously loaded elements (in a pipelined fashion).

11. A double precision floating point vector A of length 64 (i.e., A is a single dimensional array containing 64 elements) with stride 2 resides in memory at address 0x8c040020. What would be the memory address of elements A[3] and A[9]?

The “stride” is the separation between consecutive elements. A stride of 1 means that consecutive elements are adjacent in memory, A stride of N means that consecutive elements are separated by N*W bytes, where W is the width of each element. Since each element is 8 bytes in size, the distance between consecutive elements 2*8 = 16 bytes (0x10). The address of A[0] is 0x8C040020, the address of A[1] is 0x8C040020 + 1*2*0x10,
so the address of A[3] is 0x8C040020 + 3*2*0x10 = 0x8C040080.
The address of A[9] = 0x8C040020 + 9*2*0x10 = 0x8C040140.