02nd Feb 2022

Discussion revolved around the Modern Architecture, and on Scalar Architecture and its importance. Before getting into it several issues with it are elaborated such as the number of execution units needed for the Integer arithmetic and Floating point too. It needs to create the pipelines for execution and to execute them. It is hard to maintain and use them. Thus, we must find the relevant independent unit which is quite a cumbersome task. It allows to achieve parallelism with the change of order of the instructions which is much more friendly to attain the level of parallelism.

Although with many advantages it is backed with several drawbacks which includes Branches in this architecture are problematic due to the reasons where we need to work on guessing whether the branch is true/ false for enabling the pipeline. If the guess isn't right, then all the work in it is dropped often this is termed as *Pipeline Miss* and can be expensive. To resolve this issue, Branch prediction to choose it or not is needed, this brings to the big question of how to come up with one of it? How to code to make one branch predictable? There are several data structures that can be used to mitigate it.

With the consideration of Heap and Array, there is a problem with merge when there are 1000 sorted lists which are to be changed to single sorted list. By assuming that we insert elements, and we find the maximum out of it and helps in finding the branch mispredictions. Arrays, to find the minimum we need to keep track of all the elements by scanning all the elements in O(k), nextly, we can insert the element at the Position i. It can have 1 branch misprediction for the last loop iteration whereas Heap has log(k)/2 branch mispredictions. An important thing to be noted is there is a limitation to the branch memory for 300-500 branches and there are chances for the predictor to fail if attempts to go over larger number of branches. This shows that many traditionally sophisticated algorithms tend to be slower on Scalar Architecture. We can follow couple of advice to make it better, such as using Performance calculator tools "perf" to measure branch mispredictions and arrays help better and process all the elements in order.

09th Feb 2022

Understanding types of Cache memory and its nuances involved in their abilities and usages. There are 3 types of them namely, L1, L2, L3 Cache memory. L1 doesn't have any conflicts with the instructions with 16/32 KB and has a latency of 1 clock cycle whereas L2 and has 52 KB and 12-14 latency cycles. This increased with the L3 Cache to 20-60 MB and 20-25 cycles and is often shared among cores. The smallest amount of the continuous data that can cached is 64 Bytes (Cache line). Based on the time period of the data that is retained in the cache for a while gives temporal locality cache. Row store often has an issue with the inefficient usage of the cache line and is resolved using column store. A pre-fetcher anticipated the memory requests to the data and pre-fills the cache for improving the execution speeds. To make program faster, we need to emulate benchmarks one such thing is SPEC-Int.

The advancements of the cores include developing multi-cores in 2002. Also, there are several complications with it namely, they are hard to make concurrent programs, also required synchronizations to avoid race problems and it is quite costly, problems with maintaining cache consistency and coherency with the traffic.

Major issue with the pipelining is Division operation as it requires at least 80 clock cycles and only way to make this easy is to make division easy by 2^k by shifting k bits to left.

11th Feb 2022

Today's discussion is all about multi cores, EPYC – a multi core processor that uses 8 core bundles called ciplet. Each ciplet comes with its own memory controller to attain the control over it. As, a result of this memory bandwidth grows with the number of the ciplets. Maximum memory size and bandwidth is directly proportional to the number of ciplets. Understanding about NUMA- which is a solution for access of one ciplet memory access to the others.

Basically, each of the ciplet has 4 lanes for the Hyper transport Protocol, to establish a link to other ciplets and also to connect to I/O. A best to achieve good memory access is by placing compute and data on the same NUMA node. Memory transfers on NUMA are busty with 64 Bytes at a time. This also enables to have virtual memory via an indirect mechanism. One virtual memory can result in 4 real memory accesses. Using nmap function we can manage to get memory directly not through the memory allocator.