Requirements, Bottlenecks, and Good Fortune Agents for Microprocessor Evolution

(Summary by Muhammad Hamad)
(MSCS – Semester 1 -MS-IC-21354)

Introduction : In past 30 years , microprocessors has passed through a great deal of transformation from 2300 transistors with 108 kHz frequency to 200 million transistors with frequency of 1GHz , This evolution was based on new requirements , bottleneck and good fortune.

Explanation : Computer architecture is considered as art more than science . Architects from there previous experience improve there capabilities and insights . If it is science at all it is science of trade offs. Number of transistors provided by microprocessor is actual resource , in fact electrons are actual workers in processors , as we can't speak to them so we transform our problem to instructions that they understand. First of all we convert our natural language problem to algorithms and then it is encoded into **ISA(Instruction Set Architecture) .ISA** is agreed upon interface that the compiled program uses to tell the microprocessor what it should do , and the microprocessor knows what it should do. Mostly the trade off is between higher performance and lower cost. The design of a microprocessor is about making relevant trade offs. Performance, cost, heat dissipation, and power consumption are examples of characteristics that strongly affect a design point. Another is "high availability" .Another one is that the set of applications for which we wish to use microprocessors. Other examples of the application space that continue to drive the need for unique design point are scientific applications, transaction-based applications, network applications, media applications.

Microprocessor processes instructions by supply instructions to the core of the processor where each instruction can do its job and supply data needed by each instruction and perform the operations required by each instruction. Earlier one instruction was fetched at a time now it has been advanced to four , Three problems in supplying instructions are instruction cache misses, fetch breaks, and conditional branch mispredictions. To supply data needed by an instruction, one needs the ability to have available an infinite supply of needed data,to supply it in zero time, and at reasonable cost. Storage can't accommodate all three. Improvements in processor cycle time continue to grow at a much faster rate than memory cycle time. To perform instructions functional units must collaborate but as on-chip cycle times decrease, the latency become worse. High performance, was the need for more than one instruction to be processed at the same time. Tommorrow's new demand is HCI now.

Bottlenecks are there that prevent three components of instruction processing from doing their jobs, e.g. instruction supply requires fetching some instructions each cycle. If these instructions were stored in memory, the time to fetch would be too long. The instruction cache was invented because of slow memory. The bottle-neck, which is caused by the conditional branch, is the arrangement of instructions in the order produced by the compiler, rather than the (dynamic) order in which the instructions are executed.

Good fortune happens when something allows you add more features to the processors like because of technology shrinks e.g. on-chip floating point accelerator and the multimedia instruction extension capability. First step towards pipelining by prefetching the next instruction while the current instruction was being executed. The latency to get instructions and data from off-chip memory to the on-chip processing elements was too long. The result: an on-chip cache. A cache can either be fast or large, not both. Since the cache had to be fast, it had to also be small. Two levels of cache on-chip was made, so that a miss in the fast, small first level cache could be satisfied by a larger, slower second level cache that was still a lot faster than going off-chip. The benefits of pipelining are lost if conditional branches produce pipeline stalls waiting for the condition on which the branch is based to be resolved. As transistors got smaller and chips got larger, the transistor count reached the point where the floating point unit could be placed on the same chip with the main processing unit thus saving unnecessary off-chip communication. Now multiprocessors have multiple functional unit that helps us in achieving concurrency e.g. separate address ALU’s, sophisticated load/store functional unit containing structures like write buffers, a miss pending queue, and a mechanism for handling memory disambiguation were added .Another mechanism used in HPC is out of order processing of instructions if the subsequent instruction has all that it needs to begin execution. Register aliasing and reservation stations has been used for this purpose. To do so with precise exceptions, the microprocessor had to add the distinction between instruction execution and instruction retirement. Instructions were allowed to execute whenever their resources (data and functional units) became available independent of their order in the program, but were forced to retire in the same order that they occurred in the executing program. As Single die size continues to increase, feature size continues to decrease, and on-chip frequencies continue to increase. The result is that a value produced by a functional unit at one corner of the chip can not traverse the chip and be available as a source to a functional unit at the opposite corner of the chip in the next cycle. This results partition the execution core into clusters. An alternative approach is to partition the chip into regions, with an identical processor occupying each region. The paradigm is referred to as CMP, for chip multiprocessor. Run the execution core at a frequency much faster than the rest of the microprocessor to remove the flow dependencies of source operands waiting for the results .On-chip frequencies are expected to be so high that their exist a challenge to redesign the data path in light of the new constraint of wire length. Future microprocessors will have to provide functionality to check for and correct these soft errors as they occur. There should be system so that we will see structures that operate asynchronously for some fixed period of time after which they synchronize with the global clock. Off-chip bandwidth is expensive, on-chip bandwidth is plentiful. My expectation: we will more effectively harness on-chip bandwidth. The expanded use of microcode is a way to do that.

The list of features that author wants to see in high performance microprocessor are expanded use of the trace cache,On-chip microcode for using the spare capacity, Dynamic recompilation , Multiple (at least three) levels of cache with corresponding ISA additions, Performance monitoring hardware and structure for monitoring and energy usage of the chip