INF2270 — Spring 2011

Lecture 8: Superscalar CPUs

UNIVERSITETET I OSLO

content

From Scalar to Superscalar

content

From Scalar to Superscalar

Scalar Processors

The concept of a CPU that we have discussed so far where all scalar processors, in as far as they do not execute operations in parallel and produce only a single result data item at a time.

Vector processors

High performance computing led to vector processors, most prominently the Cray-1 in 1976 that had 8 vector registers of 64 words of 64-bit length. Vector processors perform 'single instruction multiple datastream' (SIMD) computations, i.e. they execute the same operation on a vector instead of a scalar. Some machines used parallel ALU's but the Cray-1 used a dedicated pipelining architecture that would fetch a single instruction and then execute it efficiently, e.g. 64 times, saving 63 fetches.

Multi processor

Vector computers lost popularity with the introduction of multi-processor computers such as Intels's Paragon series of *massively parallel supercomputers*: Combineing multiple (standard) CPU's is cheaper than designing powerfull vector processors, even considering a bigger *communication overhead*: in some architectures with a single shared memory/system bus the instructions and the data need to be fetched and written in sequence for each processor, making the von Neumann bottleneck more severe. However, many more clever solutions were introduced, e.g. local memory/cache and/or parallel memory access.

Clusters/Grids

But even cheaper and obtainable for the common user are Ethernet clusters of individual computers, or even computer grids connected over the internet. Both of these, obviously, suffer from massive communication overhead and espescially the latter are best used for so called 'embarassingly parallel problems', i.e. computation problems that do require no or minimal communication of the computation nodes.

Multi Core

Designing more complicated integrated circuits has become cheaper with progressing miniaturization, such that several processing units can now be accomodated on a single chip which has now become standard with AMD and Intel processors. These multi-core processors have many of the advantages of multi processor machines, but with much faster communication between the cores, thus, reducing communication overhead. (Although, it has to be said that they are most commonly used to run individual independent processes, and for the common user they do not compute parallel problems.)

Superscalar Processor Principle

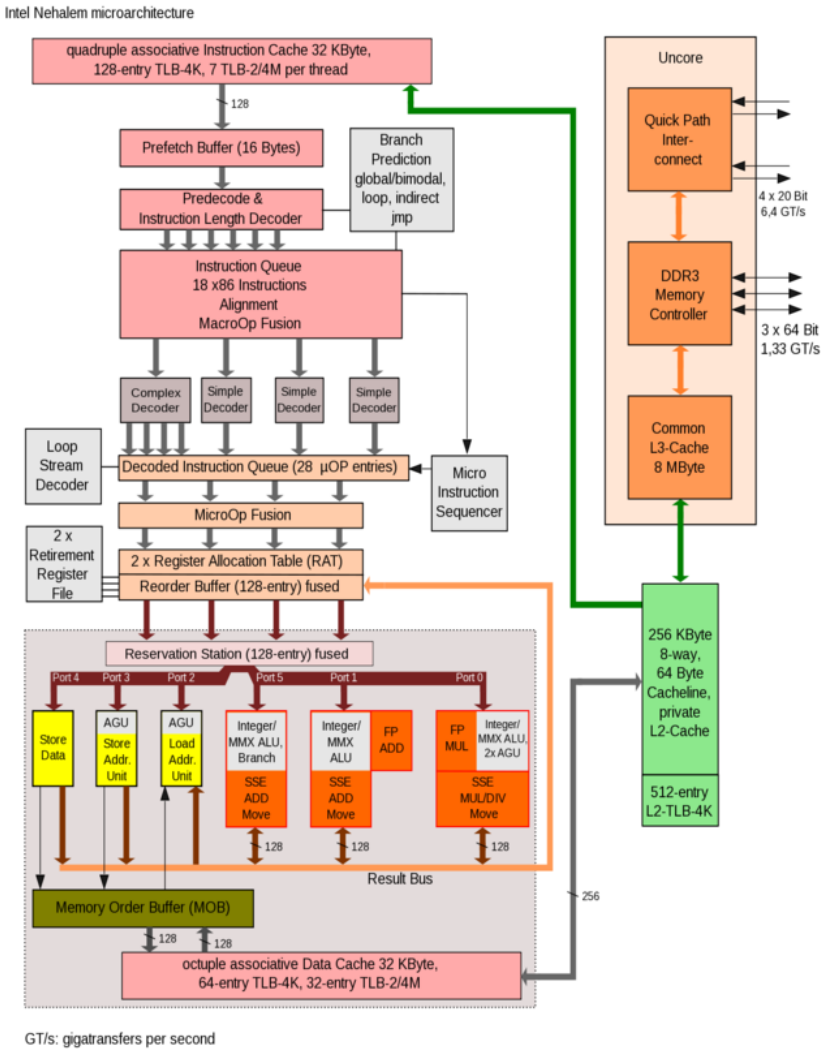
Superscalar processors were introduced even before multi-core and all modern designs belong to this class. Like vector processeors they are actually capable of executing instructions *in parallel*, but in contrast to vector computers, they are *different* instructions. Instead of replication of the basic functional units n-times in hardware (e.g. the ALU), superscalar processors exploit the fact that there already *are* multiple functional units. For example, many processors do sport both an ALU and a FPU. Thus, they should be able to execute an integer- and a floating-point operation *simultaneously*. Data access operations do not require the ALU nor the FPU (or have a dedicated ALU for address operations) and can thus also be executed at the same time.

Superscalar Processor

For this to work, several instructions have to be fetched in parallel, and then *dispatched*, either in parallel, if possible, or in sequence, if necessary. A sentral additional stage is an *reorder buffer (ROB)* that reorders the microinstructions for better throughput. The pipeline is then divided into parallel execution units for differnt types of instructions.
Superscalar processors can ideally achive an average clock cycle per instruction (CPI) smaller than 1, and a speedup higher than the number of pipelining stages k (which is saying the same thing in two different ways).
Compiler level support can group instructions to optimize the potential for parallel execution.

Intel Nehalem Architecture

As an example: the Intel Core i7 uses the 'Nehalem' architecture, has has 14 pipeline stages and can execute up to 6 instructions in parallel.



Some Elements in Superscalar Architectures (1/3)

Micro-instruction reorder buffer (ROB): Stores all instructions that await execution and dispatches them for *out-of-order execution* when appropriate. Note that, thus, the order of execution may be quite different from the order of your assembler code. Extra steps have to be taken to avoid and/or handle hazards caused by this reordering.

Some Elements in Superscalar Architectures (2/3)

Retirement stage: The pipelining stage that takes care of finished instructions and makes the result appear consistent with the execution sequence that was intended by the programmer. In particular it delays write back to the register file of results along a branch taken due to barnch prediction until it is clear that the prediction was correct.

Some Elements in Superscalar Architectures (3/3)

Reservation station registers: A single instruction reserves a set of these registers for all the data needed for its execution on its functional unit. Each functional unit has several slots in the reservation station. Once all the data becomes available and the functional unit is free, the instruction is executed.

Test Yourself (Exam 2010)

Riktig eller feil?

- Teoretisk kan en superskalar datamaskin med et skreddersyd program fåen speedup som er større enn antall trinn i pipelinen.