

## 10 Innovations in the Intel Pentium 4

- 1) Execution Trace Cache – The trace cache sits between the instruction decode and execution core and is able to store already decoded instructions reducing the load on the decoder.
- 2) Out of order Execution – The Out of Order execution engine lets the processor reorder instructions to optimize throughput as they go through the pipeline. It uses retirement logic to reorder the results back to their original forms.
- 3) Microcode ROM – The microcode ROM is used to translate complex IA-32 instructions such as string moves, and exception handling into simple microcode that can be executed directly by the processor.
- 4) Advanced Register Renaming – The register renaming uses a front end RAT and a retirement RAT decoupling the Data and ROB buffers. Upon retirement, no result data values are actually moved from one physical structure to another, increasing performance.
- 5) Uop Scheduling – There are two queues—one for memory and one for non-memory operations. Each queue stores the uops in FIFO order, but can execute out-of-order with respect to the other queue allowing the dynamic out-of-order scheduling window to be larger than on previous implementations.
- 6) Double Pumped ALU – The double pumped ALU allows the Pentium 4 to do fully dependent ALU operations at twice the main clock rate. One such example is a staggered add that can add a double wide number in a single processor clock.
- 7) Clock Rates – The Pentium 4 processor implements a 20-stage pipeline enabling a much higher clock rate than previously possible.
- 8) Low Latency L1 Data Cache – The L1 data cache operates with a 2 clock load-use latency for integer and a 6 clock load-use latency for floating point loads. This is achieved with a new access algorithm that leverages the fact that almost all accesses hit the first-level data cache and the data table.
- 9) Store-to-Load Forwarding – With the very deep pipeline of the Pentium 4, it can take many cycles for a store to retire and commit. To enable loads to use the value of pending stores, loads use data written into the L1 data cache by stores that have not yet committed.
- 10) Bandwidth – The Pentium 4 processor has a 64-bit wide system bus clocked at 400 MHz capable of transferring 3.2 Gbytes per second.

## 10 factors that may limit the performance of the IA-64 architecture

- 1) Lower Frequency – The IA-64 claims to be less complex than traditional superscalar processors, but Intel has made no claims on its ability to support high frequencies that can increase the instruction throughput.
- 2) Predication – Predication depends on the existence of many functional units because approximately half the predicated instructions are discarded effectively wasting hardware, power, and processor calculations.
- 3) Explicit addressing – The IA-64 processors have only base registers and lack offset fields forcing the explicit computation of an address in advance.
- 4) Sign-Extending – The IA-64 has no sign-extended loads greatly increasing the path length
- 5) Integer Operations – The IA-64 has no integer multiply or divide instructions available. Instead it must explicitly copy data to and from the floating-point registers to perform these operations.
- 6) Instruction Bundle Placement – The IA-64 architecture places sever limitations on instruction placement which can cause a larger code footprint and/or more cycles wasted due to less compact code.
- 7) Pollution of Cache – Code bloat can pollute the instruction cache while speculative loads will pollute data cache and use bandwidth.
- 8) In order execution – In-order execution defined by the IA-64 can result in stalling the pipeline when a superscalar processor would not.
- 9) Recovery Code – To recover from speculative operations, recovery code needs to be used. This recovery code may not be in cache and consequently may cause page faults that would not have happened in a superscalar processor.
- 10) Dynamic Information – A superscalar machine makes extensive use of dynamic information while the iA-64 relies almost entirely on decisions made statically at compile time. Dynamic information can provide a wealth of data that is not available at compile time.