Nehalem

Intel Micro-architecture
Features:

- **Wide Dynamic Execution:**
  Every processor core can fetch, dispatch, execute and retire up to four instructions per clock cycle.

- **Advanced Smart Cache:**
  improved bandwidth from the second level cache to the core, and improved support for single- and multi-threaded applications computation.

- **Smart Memory Access:**
  which pre-fetches data from memory responding to data access patterns, reducing cache-miss exposure of out-of-order execution.

- **Advanced Digital Media Boost:**
  for improved execution efficiency of most 128-bit SIMD instruction with single-cycle throughput and floating-point operations.
Instruction and Data Flow Process:

- The early stages of the processor **fetch**-in several macro-instructions at a time.
- **decode** them into sequences of micro-ops.
- The micro-ops are buffered at various places where they can be picked up and scheduled to use in parallel if data dependencies are not violated. In Nehalem, micro-ops are issued to stations where they reserve their position for subsequent.
- dispatching as soon as their input operands become available.
- Finally, completed micro-ops retire and post their results to permanent storage.
Hardware implementation

- four identical compute cores
- UIU: Un-Core Interface Unit (switch connecting the 4 cores to the 4 L3 cache segments, the IMC and QPI ports)
- L3: level-3 cache controller and data block memory
- IMC: 1 integrated memory controller with 3 DDR3 memory channels
- QPI: 2 Quick-Path Interconnect ports
- auxiliary circuitry for cache-coherence, power control, system management and performance monitoring logic
Software Access

- a 64-bit linear (“flat”) logical address space,
- uniform byte-register addressing,
- 16 64-bit-wide General Purpose Registers (GPRs) and instruction pointers
- 16 128-bit “XMM” registers for streaming SIMD extension instructions, in addition to 8 64-bit MMX registers or the 8 80-bit x87 registers, supporting floating-point or integer operations,
- fast interrupt-prioritization mechanism,
- a new instruction-pointer relative-addressing mode.
Front-End In-order Pipeline

- Retrieve blocks of macro-instruction from memory
- Translate instruction
- Handle instruction in-order
- Decode 4 instruction per cycle
- Decode instruction streams of threads in alternate cycles
Execution Engine Out-of-order Pipelines

- Dynamically schedule micro-ops for dispatching and execution
- Dispatch up to 6 micro-ops per cycle
- Four micro-ops can retire per cycle
- Result written-back rate up to one register per port per cycle