Intel® Next Generation Nehalem Microarchitecture

Andrey Semin
HPC Technology Manager
Intel Corporation, EMEA
Agenda

• Nehalem Design Philosophy
• Enhanced Processor Core
• New Instructions
• Optimization Guidelines and Software Tools
• New Platform Features

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Penryn, Nehalem, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

Intel, Intel Inside, Xeon, Core, Pentium, AVX and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2009 Intel Corporation.
Intel Tick-Tock Development Model: Delivering Leadership Multi-Core Performance

Silicon and Software Tools Unleash Performance
Nehalem: Scalable Cores

Same core for all segments
Common software optimization
Common feature set

Nehalem 45nm

Servers/Workstations
Energy Efficiency, Performance, Virtualization, Reliability, Capacity, Scalability

Desktop
Performance, Graphics, Energy Efficiency, Idle Power, Security

Mobile
Battery Life, Performance, Energy Efficiency, Graphics, Security

Optimized cores to meet all market segments
Core Microarchitecture Recap

Wide Dynamic Execution
- 4-wide decode/rename/retire

Advanced Digital Media Boost
- 128-bit wide SSE execution units

Intel HD Boost
- New SSE4.1 Instructions

Smart Memory Access
- Memory Disambiguation
- Hardware Prefetching

Advanced Smart Cache
- Low latency, high BW shared L2 cache

Nehalem builds on the great Core microarchitecture
# Nehalem Micro-Architecture

A new dynamically scalable microarchitecture

## Key Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>45nm Intel® multi-core processors</td>
<td>(2, 4, 8 core implementations planned)</td>
</tr>
<tr>
<td>Greater Instruction per clock and improved cache hierarchy</td>
<td></td>
</tr>
<tr>
<td>Simultaneous Multi-Threading</td>
<td></td>
</tr>
<tr>
<td>Dynamic Resource Scaling</td>
<td>Any unneeded cores automatically put into sleep mode; remaining operating cores get access to ALL cache, bandwidth and power/thermal budgets</td>
</tr>
<tr>
<td>Turbo Mode</td>
<td>CPU operates at higher-than-stated frequency when operating below power and thermal design points</td>
</tr>
</tbody>
</table>

## Benefits

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy efficient multi-core processing</td>
<td></td>
</tr>
<tr>
<td>Faster Processing / core</td>
<td></td>
</tr>
<tr>
<td>More Threads / core</td>
<td></td>
</tr>
<tr>
<td>Lower power consumption during periods of low utilization</td>
<td></td>
</tr>
<tr>
<td>Additional Processing boost during peak demand periods</td>
<td></td>
</tr>
</tbody>
</table>

**FASTER cores … MORE cores/threads … DYNAMICALLY ADAPTABLE**

Source: Intel. All future products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
Agenda

- Nehalem Design Philosophy
- **Enhanced Processor Core**
- New Instructions
- Optimization Guidelines and Software Tools
- New Platform Features
Designed for Performance

- New SSE4.2 Instructions
- Improved Lock Support
- Additional Caching Hierarchy

- Execution Units
- L1 Data Cache
- L2 Cache & Interrupt Servicing

- Memory Ordering & Execution
- Instruction Decode & Microcode
- Paging
- Branch Prediction

- Out-of-Order Scheduling & Retirement
- Instruction Fetch & L1 Cache

- Deeper Buffers
- Improved Loop Streaming

- Simultaneous Multi-Threading
- Faster Virtualization

- Better Branch Prediction
Enhanced Processor Core

- Instruction Fetch and Pre Decode
- Instruction Queue
- Decode
- Rename/Allocate
- Reservation Station
- Execution Units
- ITLB
- 32kB Instruction Cache
- 2nd Level TLB
- 256kB 2nd Level Cache
- DTLB
- 32kB Data Cache

Front End Execution Engine Memory

L3 and beyond
Loops are very common in most software
Take advantage of knowledge of loops in HW
- *Decoding the same instructions over and over*
- *Making the same branch predictions over and over*
Loop Stream Detector identifies software loops
- Stream from Loop Stream Detector instead of normal path
- Disable unneeded blocks of logic for **power savings**
- **Higher performance** by removing instruction fetch limitations
Front-end: Loop Stream Detector in Nehalem

Same concept as in prior implementations

**Higher performance:** Expand the size of the loops detected

**Improved power efficiency:** Disable even more logic

**Nehalem Loop Stream Detector**

Branch Prediction \(\rightarrow\) Fetch \(\rightarrow\) Decode

28 Micro-Ops
Execution Engine

Start with powerful Core 2 execution engine
- Dynamic 4-wide Execution
- Advanced Digital Media Boost
  - 128-bit wide SSE
- HD Boost (Penryn)
  - SSE4.1 instructions
- Super Shuffler (Penryn)

Add Nehalem enhancements
- Additional parallelism for higher performance
Execution Unit Overview

Unified Reservation Station
- Schedules operations to Execution units
- Single Scheduler for all Execution Units
- Can be used by all integer, all FP, etc.

Execute 6 operations/cycle
- 3 Memory Operations
  - 1 Load
  - 1 Store Address
  - 1 Store Data
- 3 “Computational” Operations

Unified Reservation Station

Port 0
- Integer ALU & Shift
- FP Multiply
- Divide
- SSE Integer ALU
  - Integer Shuffles

Port 1
- Integer ALU & LEA

Port 2
- Load
- Complex Integer
- SSE Integer Multiply

Port 3
- Store Address
- Integer ALU & Shift

Port 4
- Store Data

Port 5
- Branch
- FP Shuffle
- SSE Integer ALU
  - Integer Shuffles
Increased Parallelism

Goal: Keep powerful execution engine fed
Nehalem increases size of out of order window by 33%
Must also increase other corresponding structures

| Structure        | Merom | Nehalem | Comment                                                        |
|------------------|-------|---------|                                                               |
| Reservation Station | 32    | 36      | Dispatches operations to execution units                      |
| Load Buffers     | 32    | 48      | Tracks all load operations allocated                           |
| Store Buffers    | 20    | 32      | Tracks all store operations allocated                          |

Increased Resources for Higher Performance
Enhanced Cache Subsystem – New Memory Hierarchy

New 3-level cache hierarchy

- 1st level remains the same as Intel Core Microarchitecture
  - 32KB instruction cache
  - 32KB data cache
- New L2 cache per core
  - 256 KB per core – holds data + instructions
  - Very low latency
- New shared last level cache
  - Large size (8MB for 4-core)
  - Shared between all cores
    - Allows lightly threaded applications to use the entire cache
  - Inclusive Cache Policy
    - Minimize traffic from snoops
    - On cache miss, only check other cores if needed (data in modified state)
Multi-threaded software becoming more prevalent

**Scalability** of multi-thread applications can be limited by synchronization

Synchronization primitives: LOCK prefix, XCHG

Reduce synchronization latency for legacy software

Greater thread scalability with Nehalem
Other Performance Enhancements
Intel Xeon® 5500 Series Processor (Nehalem-EP)

**Intel® Turbo Boost Technology**

Increases performance by increasing processor frequency and enabling faster speeds when conditions allow.

- **Normal**: All cores operate at rated frequency.
- **4C Turbo**: All cores operate at higher frequency.
- **<4C Turbo**: Fewer cores may operate at even higher frequencies.

Higher performance on demand

**Intel® Hyper-Threading Technology**

Increases performance for threaded applications delivering greater throughput and responsiveness.

Up to 30% higher†

Higher performance for threaded workloads

† For notes and disclaimers, see performance and legal information slides at end of this presentation.
Hyper-Threading Implementation Details for Nehalem

Multiple policies possible for implementation of SMT

**Replicated** – Duplicate state for SMT
- Register state
- Renamed RSB
- Large page ITLB

**Partitioned** – Statically allocated between threads
- Key buffers: Load, store, Reorder
- Small page ITLB

**Competitively shared** – Depends on thread’s dynamic behavior
- Reservation station
- Caches
- Data TLBs, 2\textsuperscript{nd} level TLB

**Unaware**
- Execution units
Agenda

- Nehalem Design Philosophy
- Enhanced Processor Core
- New Instructions
- Optimization Guidelines and Software Tools
- New Platform Features
Extending Performance and Energy Efficiency
- SSE4.2 Instruction Set Architecture (ISA) Leadership

Accelerated String and Text Processing
- Faster XML parsing
- Faster search and pattern matching
- Novel parallel data matching and comparison operations

Accelerated Searching & Pattern Recognition of Large Data Sets
- Improved performance for Genome Mining, Handwriting recognition.
- Fast Hamming distance / Population count

New Communications Capabilities
- Hardware based CRC instruction
- Accelerated Network attached storage
- Improved power efficiency for Software I-SCSI, RDMA, and SCTP

What should the applications, OS and VMM vendors do?:
Understand the benefits & take advantage of new instructions in 2008.
Provide us feedback on instructions ISV would like to see for next generation of applications
STTNI - STring & TExt New Instructions

STTNI MODEL

Source2 (XMM / M128)

Equal Each Instruction
True for each character in Src2 if same position in Src1 is equal
Src1: Test\tday
Src2: tad tseT
Mask: 01101111

Equal Any Instruction
True for each character in Src2 if any character in Src1 matches
Src1: Example\nSrc2: atad tseT
Mask: 10100000

Ranges Instruction
True if a character in Src2 is in at least one of up to 8 ranges in Src1
Src1: AZ’O’9zzz
Src2: taD tseT
Mask: 00100001

Equal Ordered Instruction
Finds the start of a substring (Src1) within another string (Src2)
Src1: ABCA0XYZ
Src2: S0BACBAB
Mask: 00000010

Projected 3.8x kernel speedup on XML parsing & 2.7x savings on instruction cycles
STTNI Model

**EQUAL ANY**

Source1 (XMM)

- Bit 0
- OR results down each column

Source2 (XMM / M128)

- Check each bit in the diagonal

**EQUAL EACH**

Source1 (XMM)

- First Compare does GE, next does LE
- AND GE/LE pairs of results
- OR those results

Source2 (XMM / M128)

- AND the results along each diagonal

**RANGES**

Source1 (XMM)

- IntRes1

Source2 (XMM / M128)

- IntRes1

**EQUAL ORDERED**

Source1 (XMM)

- IntRes1

Source2 (XMM / M128)

- IntRes1
Example Code For strlen()

```c
int sttni_strlen(const char * src)
{
    char eom_vals[32] = {1, 255, 0};
    __asm{
        mov         eax, src
        movdqu      xmm2, eom_vals
        xor         ecx, ecx
        topofloop:
        add         eax, ecx
        movdqu      xmm1, OWORD PTR[eax]
        pcmpistri   xmm2, xmm1, imm8
        jnz         totopofloop
        byte_2:
        lea         eax,[ecx - 2]
        mov         ecx,string
        sub         eax,ecx
        ret
    }
}
```

Current Code: Minimum of 11 instructions; Inner loop processes 4 bytes with 8 instructions

STTNI Code: Minimum of 10 instructions; A single inner loop processes 16 bytes with only 4 instructions
ATA - Application Targeted Accelerators

**CRC32**
Accumulates a CRC32 value using the iSCSI polynomial

One register maintains the running CRC value as a software loop iterates over data. Fixed CRC polynomial = 11EDC6F41h

Replaces complex instruction sequences for CRC in Upper layer data protocols:
- iSCSI, RDMA, SCTP

Enables enterprise class data assurance with high data rates in networked storage in any user environment.

**POPCNT**
POPCNT determines the number of nonzero bits in the source.

POPCNT is useful for speeding up fast matching in data mining workloads including:
- DNA/Genome Matching
- Voice Recognition

ZFlag set if result is zero. All other flags (C,S,O,A,P) reset.
Preliminary Performance

CRC32 optimized Code

crc32c_sse42_optimized_version(uint32 crc, unsigned char const *p, size_t len)
{
    // Assuming len is a multiple of 0x10
    asm("pusha");
    asm("mov %0, %eax" :: "m" (crc));
    asm("mov %0, %ebx" :: "m" (p));
    asm("mov %0, %ecx" :: "m" (len));
    asm("1:");
    // Processing four byte at a time: Unrolled four times:
    asm("crc32 %eax, 0x0(%ebx)" );
    asm("crc32 %eax, 0x4(%ebx)" );
    asm("crc32 %eax, 0x8(%ebx)" );
    asm("crc32 %eax, 0xc(%ebx)" );
    asm("add $0x10, %ebx")2;
    asm("sub $0x10, %ecx");
    asm("jecxz 2f");
    asm("jmp 1b");
    asm("2:");
    asm("mov %eax, %0" : "m" (crc));
    asm("popa");
    return crc;
}

- Preliminary tests involved Kernel code implementing CRC algorithms commonly used by iSCSI drivers.
- 32-bit and 64-bit versions of the Kernel under test
- 32-bit version processes 4 bytes of data using 1 CRC32 instruction
- 64-bit version processes 8 bytes of data using 1 CRC32 instruction
- Input strings of sizes 48 bytes and 4KB used for the test

<table>
<thead>
<tr>
<th>Input Data Size</th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 bytes</td>
<td>6.53 X</td>
<td>9.85 X</td>
</tr>
<tr>
<td>4 KB</td>
<td>9.3 X</td>
<td>18.63 X</td>
</tr>
</tbody>
</table>

Preliminary Results show CRC32 instruction outperforming the fastest CRC32C software algorithm by a big margin
Agenda

- Nehalem Design Philosophy
- Enhanced Processor Core
- New Instructions
- **Optimization Guidelines and Software Tools**
- New Platform Features
Software Optimization Guidelines

Most optimizations for Core microarchitecture still hold

Examples of new optimization guidelines:
- 16-byte unaligned loads/stores
- Enhanced macrofusion rules
- NUMA optimizations

Nehalem SW Optimization Guide are published

Intel Compiler supports settings for Nehalem optimizations (e.g. -xSSE4.2 option)
Simplified Many-core Development with Intel® Tools

**Insight**
- VTune™ Analyzer
  - Find the code that can benefit from threading and multicore
  - Find hotspots that limit performance

**Methods**
- Compilers / Libraries
  - MKL
  - TBB
  - IPP
- Clients
  - OpenMP
  - Ct research
  - Hybrid methods
- Clusters
  - MPI
  - Hybrid methods

**Confidence**
- Intel® Thread Checker
  - Find deadlocks and race conditions
- Intel® Trace Analyzer and Collector
  - Event based tracing

**Performance**
- VTune Analyzer
  - Tune for performance and scalability
- Intel® Thread Profiler
  - Visualize efficiency of threaded code

---

Architectural Analysis | Introduce Parallelism | Confidence/Correctness | Optimize/Tune
---|---|---|---
Windows; Linux; Mac OS
Tools Support of New Instructions

- Intel Compiler 10.x+ supports the new instructions
  - SSE4.2 supported via intrinsics
  - Inline assembly supported on both IA-32 and Intel64 targets
  - Necessary to include required header files in order to access intrinsics
    - `<tmmintrin.h>` for Supplemental SSE3
    - `<smmintrin.h>` for SSE4.1
    - `<nmmintrin.h>` for SSE4.2

- Intel Library Support
  - XML Parser Library released in Fall ’08
  - IPP is investigating possible usages of new instructions

- Microsoft Visual Studio 2008 VC++
  - SSE4.2 supported via intrinsics
  - Inline assembly supported on IA-32 only
  - Necessary to include required header files in order to access intrinsics
    - `<tmmintrin.h>` for Supplemental SSE3
    - `<smmintrin.h>` for SSE4.1
    - `<nmmintrin.h>` for SSE4.2
  - VC++ 2008 tools masm, msdis, and debuggers recognize the new instructions


```c
#include "multiply_d.h"

void dgemm(
    const double *A, const double *B, double *C)
{
    unsigned i, j, k;

    for (i = 0; i < NUM; ++i) {
        for (j = 0; j < NUM; ++j) {
            const double *B_j = B + j * NUM;
            double cij = *(C + j * NUM + i);
            for (k = 0; k < NUM; ++k) {
                cij += *(A_i + k * NUM) * *(B_j + k);
            }
            *(C + j * NUM + i) = cij;
        }
    }

    /*
    */
    
    void dgemm
}
```
VTune Tuning Assist View

- High branch mispredictions impact
- The CPI is high
- Many L2 Demand Misses

Use specific events to focus on instructions of interest.
VTune Sampling Over Time View

Sampling Over Time Views Show How Sampling Data Changes Over Time
• Detect hidden potential non-deterministic multithreading errors such as deadlocks and data races

• Analyze the results using Visual Studio* integration or a standalone graphical interface.

• Quickly drill down to the source to identify problematic lines of code
Use the Same Toolset for 32/64 bit on Windows*, Linux* and Mac OS* X

<table>
<thead>
<tr>
<th>Intel® Software Development Products</th>
<th>Operating Systems</th>
<th>Operating Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows*</td>
<td>Linux*</td>
</tr>
<tr>
<td><strong>Development Environments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compilers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Fortran</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Performance Analyzers</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>VTune® Performance Analyzer</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Performance Libraries</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Integrated Performance Primitives</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Math Kernel Library</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Mobile Platform SDK</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Threading Analysis Tools</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Thread Checker</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Thread Profiler</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Cluster Tools</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>MPI Library</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Trace Analyzer and Collector</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Math Kernel Library Cluster Edition</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Cluster Toolkit</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>XML Tools**</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>

** = Currently Available

From Servers to Mobile / Wireless Computing, Intel® Software Development Products Enable Application Development Across Intel® Platforms

** Additional XML tools information can be found at www.intel.com/software/xml
Agenda

• Nehalem Design Philosophy
• Enhanced Processor Core
• New Instructions
• Optimization Guidelines and Software Tools
• New Platform Features
Feeding the Execution Engine

Powerful 4-wide dynamic execution engine
Need to keep providing fuel to the execution engine

Nehalem Goals
- **Low latency** to retrieve data
  - Keep execution engine fed w/o stalling
- High data **bandwidth**
  - Handle requests from multiple cores/threads seamlessly
- **Scalability**
  - Design for increasing core counts

Combination of great **cache hierarchy** and **new platform**

*Nehalem designed to feed the execution engine*
Designed For Modularity

Optimal price / performance / energy efficiency for server, desktop and mobile products

Differentiation in the “Uncore”:

- # cores
- # mem channels
- # QPI Links
- Size of cache
- Type of Memory
- Power Management
- Integrated graphics

2009 Servers & Desktops
Today’s Platform Architecture

Front-Side Bus Evolution

CPU → MCH → memory → ICH

CPU → MCH → memory → ICH

CPU → MCH → memory → ICH
Nehalem Based System Architecture

Nehalem Microarchitecture
Integrated Intel® QuickPath Memory Controller
Intel® QuickPath Interconnect
Buffered or Un-buffered Memory
PCI Express* Generation 2
Optional Integrated Graphics

Source: Intel. All future products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
Integrated Memory Controller (IMC)

Memory controller optimized per market segment

Initial Nehalem products
- Native DDR3 IMC
- Up to 3 channels per socket
- Speeds up to DDR3-1333
  - Massive *memory bandwidth*
- Designed for *low latency*
- Support RDIMM and UDIMM
- RAS Features

Future products
- **Scalability**
  - Vary # of memory channels
  - Increase memory speeds
  - Buffered and Non-Buffered solutions
- Market specific needs
  - Higher memory capacity
  - Integrated graphics

*Significant performance through new IMC*
IMC Memory Bandwidth (BW)

3 memory channels per socket
Up to DDR3-1333 at launch
- Massive memory BW
- HEDT: 32 GB/sec peak
- 2S server: 64 GB/sec peak

**Scalability**
- Design IMC and core to take advantage of BW
- Allow performance to scale with cores
  - Core enhancements
    - Support more cache misses per core
    - Aggressive hardware prefetching w/ throttling enhancements
  - Example IMC Features
    - Independent memory channels
    - Aggressive Request Reordering

Massive memory BW provides performance and scalability
QuickPath Interconnect

Nehalem introduces new QuickPath Interconnect (QPI)

*High bandwidth, low latency* point to point interconnect

Up to 6.4 GT/sec initially
- 6.4 GT/sec -> 12.8 GB/sec
- Bi-directional link -> 25.6 GB/sec per link
- Future implementations at even higher speeds

Highly *scalable* for systems with varying # of sockets
Non-Uniform Memory Access (NUMA)

FSB architecture
- All memory in one location
Starting with Nehalem
- Memory located in multiple places
Latency to memory dependent on location
Local memory
- Highest BW
- Lowest latency
Remote Memory
- Higher latency

Ensure software is NUMA-optimized for best performance
Local Memory Access

CPU0 requests cache line X, not present in any CPU0 cache
- CPU0 requests data from its DRAM
- CPU0 snoops CPU1 to check if data is present

Step 2:
- DRAM returns data
- CPU1 returns snoop response

Local memory latency is the maximum latency of the two responses
Nehalem optimized to keep key latencies close to each other
Remote Memory Access

CPU0 requests cache line X, not present in any CPU0 cache
- CPU0 requests data from CPU1
- Request sent over QPI to CPU1
- CPU1’s IMC makes request to its DRAM
- CPU1 snoops internal caches
- Data returned to CPU0 over QPI

Remote memory latency a function of having a low latency interconnect
**Memory Latency Comparison**

*Low memory latency* critical to high performance

Design integrated memory controller for low latency

Need to optimize both local and remote memory latency

Nehalem delivers

- Huge reduction in local memory latency
- Even remote memory latency is fast

Effective memory latency depends per application/OS

- Percentage of local vs. remote accesses
- Nehalem has lower latency regardless of mix

![Relative Memory Latency Comparison](chart.png)
Summary

Nehalem – The 45nm Tock designed for
- *Power Efficiency*
- *Scalability*
- *Performance*

Enhanced Processor Core
Brand New Platform Architecture
Extending x86 ISA Leadership
Tools Available to support new processors feature and ISA

More web based info:
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

Intel may make changes to specifications and product descriptions at any time, without notice. All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Penryn, Nehalem, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel’s internal code names is at the sole risk of the user.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

Intel, Intel Inside, Xeon, Core, Pentium, AVX and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2009 Intel Corporation.
Risk Factors

This presentation contains forward-looking statements that involve a number of risks and uncertainties. These statements do not reflect the potential impact of any mergers, acquisitions, divestitures, investments or other similar transactions that may be completed in the future. The information presented is accurate only as of today’s date and will not be updated. In addition to any factors discussed in the presentation, the important factors that could cause actual results to differ materially include the following: Factors that could cause demand to be different from Intel’s expectations include changes in business and economic conditions, including conditions in the credit market that could affect consumer confidence; customer acceptance of Intel’s and competitors’ products; changes in customer order patterns, including order cancellations; and changes in the level of inventory at customers. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Additionally, Intel is in the process of transitioning to its next generation of products on 45 nm process technology, and there could be execution issues associated with these changes, including product defects and errata along with lower than anticipated manufacturing yields. Revenue and the gross margin percentage are affected by the timing of new Intel product introductions and the demand for and market acceptance of Intel’s products; actions taken by Intel’s competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; Intel’s ability to respond quickly to technological developments and to incorporate new features into its products; and the availability of sufficient components from suppliers to meet demand. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; excess or obsolete inventory; manufacturing yields; changes in unit costs; impairments of long-lived assets, including manufacturing, assembly/test and intangible assets; and the timing and execution of the manufacturing ramp and associated costs, including start-up costs. Expenses, particularly certain marketing and compensation expenses, vary depending on the level of demand for Intel’s products, the level of revenue and profits, and impairments of long-lived assets. Intel is in the midst of a structure and efficiency program that is resulting in several actions that could have an impact on expected expense levels and gross margin. Intel is also in the midst of forming Numonyx, a private, independent semiconductor company, together with STMicroelectronics N.V. and Francisco Partners L.P. A change in the financial performance of the contributed businesses could have a negative impact on our financial statements. Intel’s equity proportion of the new company’s results will be reflected on its financial statements below operating income and with a one quarter lag. The results could have a negative impact on Intel’s overall financial results. Intel’s results could be affected by the amount, type, and valuation of share-based awards granted as well as the amount of awards cancelled due to employee turnover and the timing of award exercises by employees. Intel’s results could be impacted by adverse economic, social, political and physical/infrastructure conditions in the countries in which Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel’s results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel’s SEC reports. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the report on Form 10-Q for the quarter ended Sept. 29, 2007.