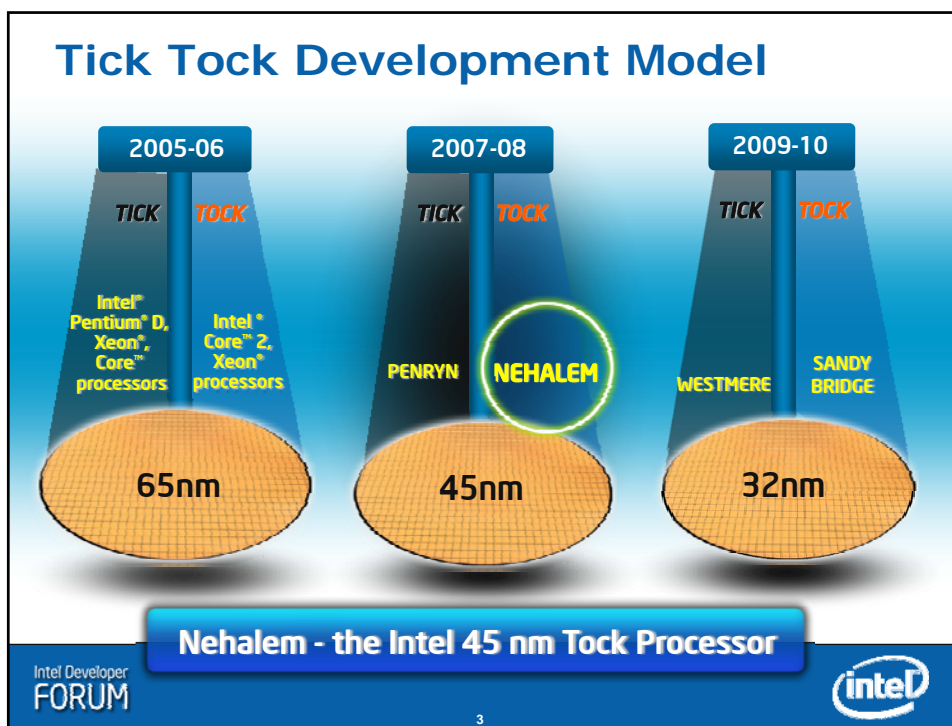
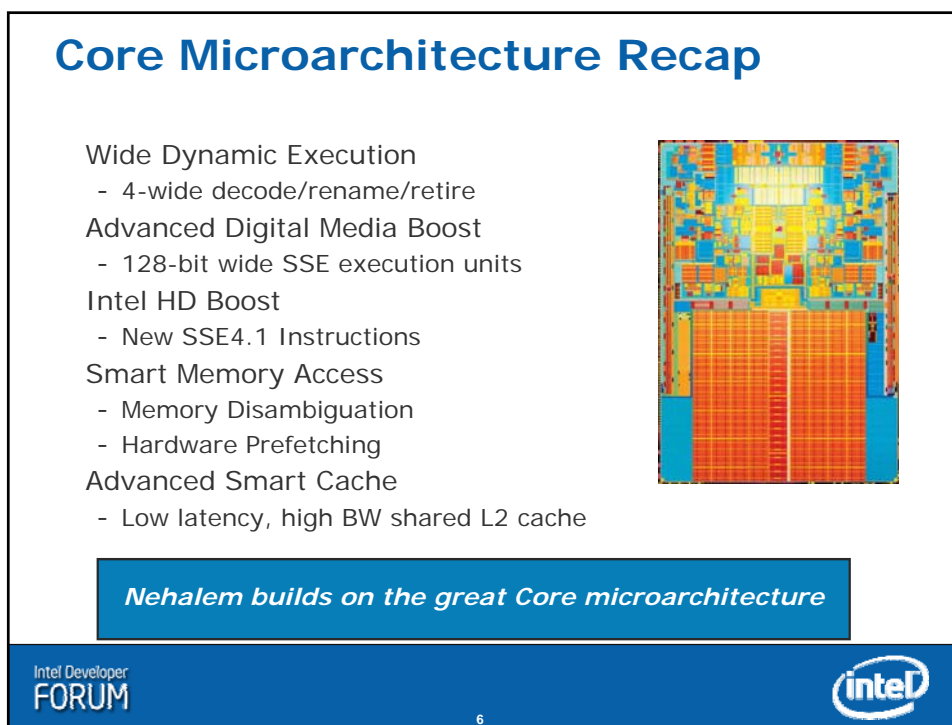
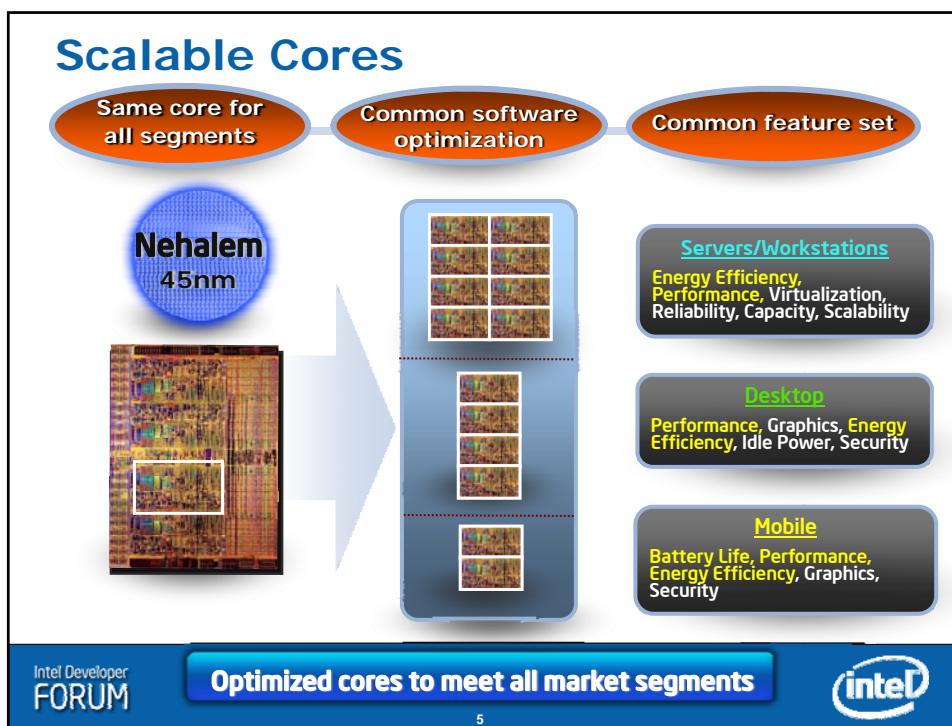


## Agenda

- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
  - Performance Features
  - Simultaneous Multi-Threading
- **New Platform**
  - New Cache Hierarchy
  - New Platform Architecture
- **Performance Acceleration**
  - Virtualization
  - New Instructions





## Agenda

- **Nehalem Design Philosophy**

- **Enhanced Processor Core**

- Performance Features
- Simultaneous Multi-Threading

- **New Platform**

- New Cache Hierarchy
- New Platform Architecture

- **Performance Acceleration**

- Virtualization
- New Instructions

Intel Developer  
FORUM



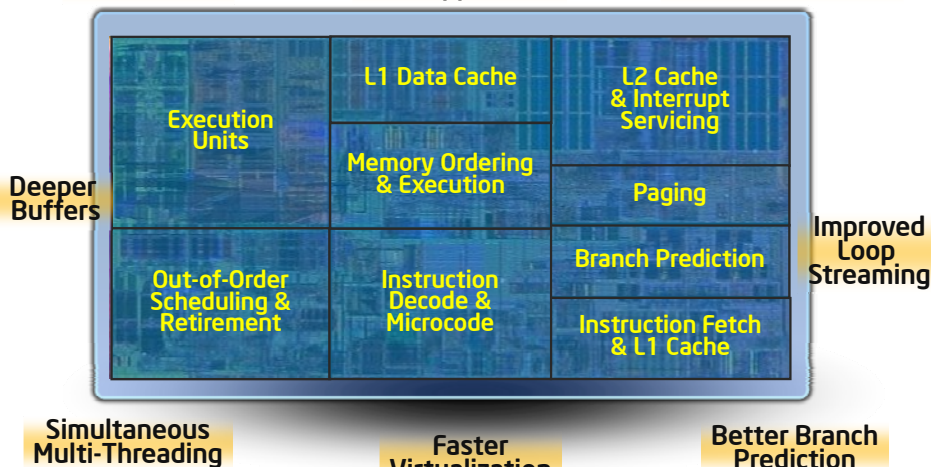
7

## Designed for Performance

New SSE4.2  
Instructions

Improved Lock  
Support

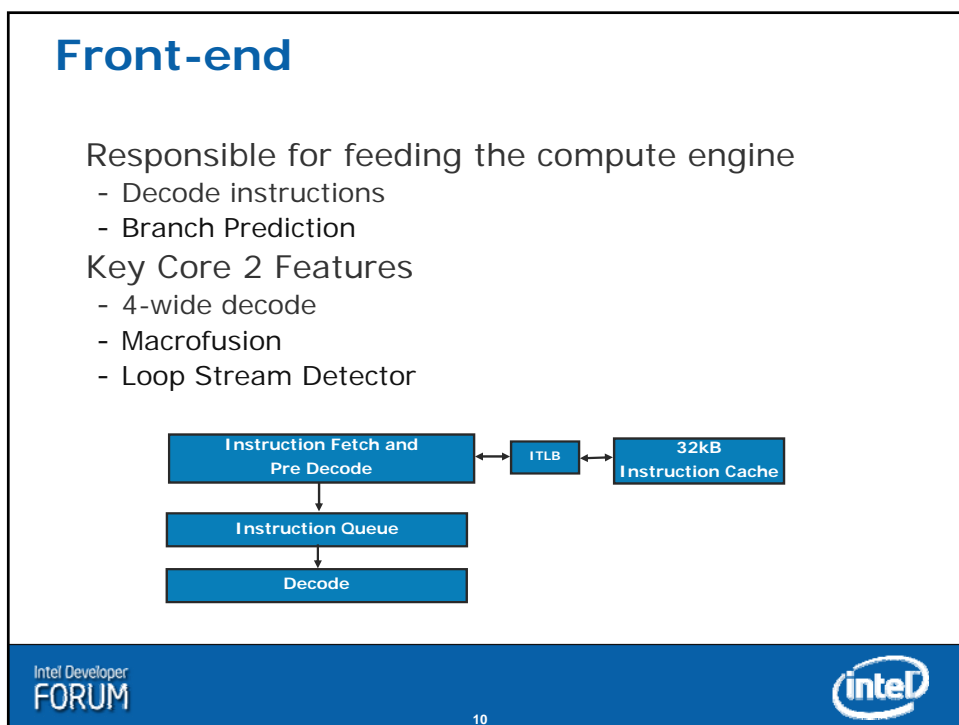
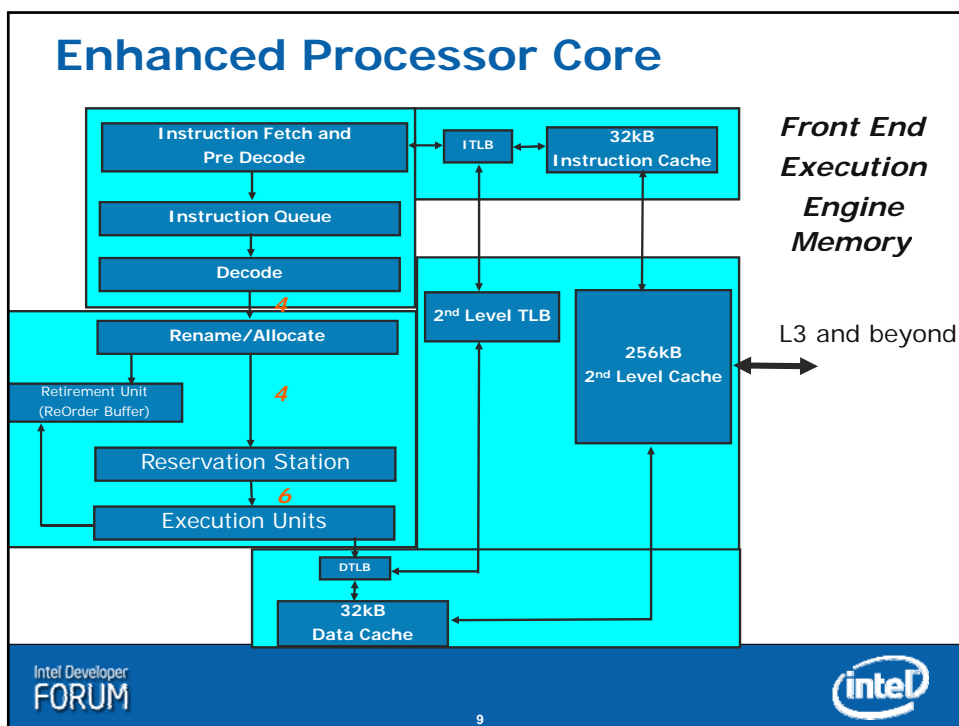
Additional Caching  
Hierarchy



Intel Developer  
FORUM



8



## Macrofusion Recap

Introduced in Core 2

TEST/CMP instruction followed by a conditional branch treated as a single instruction

- Decode as one instruction
- Execute as one instruction
- Retire as one instruction

Higher *performance*

- Improves throughput
- Reduces execution latency

Improved *power efficiency*

- Less processing required to accomplish the same work

## Nehalem Macrofusion

Goal: Identify more macrofusion opportunities for increased *performance* and *power efficiency*

Support all the cases in Core 2 **PLUS**

- CMP+Jcc macrofusion added for the following branch conditions
  - JL/JNGE
  - JGE/JNL
  - JLE/JNG
  - JG/JNLE

Core 2 only supports macrofusion in 32-bit mode

- Nehalem supports macrofusion in both 32-bit and 64-bit modes

*Increased macrofusion benefit on Nehalem*

## Loop Stream Detector Reminder

Loops are very common in most software

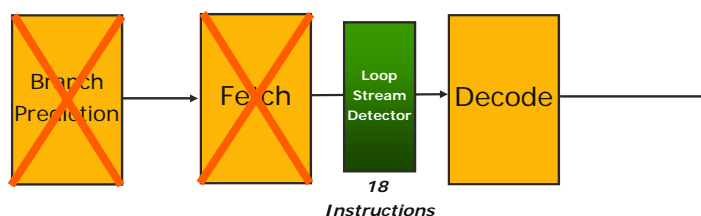
Take advantage of knowledge of loops in HW

- 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

### Core 2 Loop Stream Detector



Intel Developer  
FORUM

13



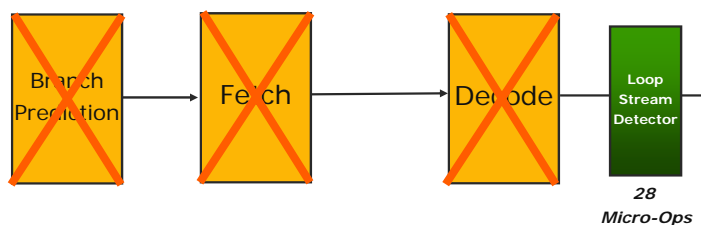
## Nehalem Loop Stream Detector

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



Intel Developer  
FORUM

14



## Branch Prediction Reminder

Goal: Keep powerful compute engine fed

Options:

- Stall pipeline while determining branch direction/target
- Predict branch direction/target and correct if wrong

Minimize amount of time wasted correcting from incorrect branch predictions

- **Performance:**

- Through higher branch prediction accuracy
- Through faster correction when prediction is wrong

- **Power efficiency:** Minimize number of speculative/incorrect micro-ops that are executed

*Continued focus on branch  
prediction improvements*

## L2 Branch Predictor

Problem: Software with a large code footprint not able to fit well in existing branch predictors

- Example: Database applications

Solution: Use multi-level branch prediction scheme

Benefits:

- Higher **performance** through improved branch prediction accuracy
- Greater **power efficiency** through less mis-speculation



## Renamed Return Stack Buffer (RSB)

### Instruction Reminder

- CALL: Entry into functions
- RET: Return from functions

### Classical Solution

- Return Stack Buffer (RSB) used to predict RET
- RSB can be corrupted by speculative path

### The *Renamed RSB*

- No RET mispredicts in the common case

## Execution Engine

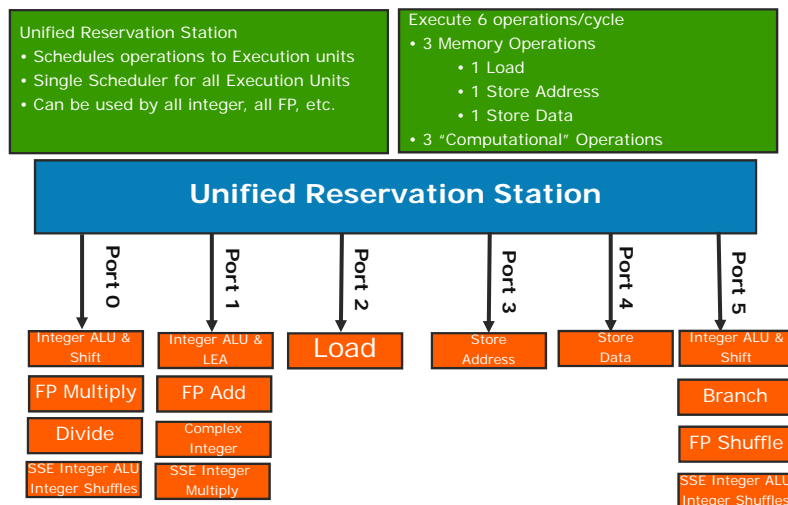
### Start with powerful Core 2 execution engine

- 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

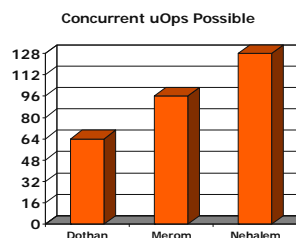


## Increased Parallelism

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 Memory Subsystem

Start with great Core 2 Features

- Memory Disambiguation
- Hardware Prefetchers
- Advanced Smart Cache

New Nehalem Features

- New TLB Hierarchy
- Fast 16-Byte unaligned accesses
- Faster Synchronization Primitives

## New TLB Hierarchy

Problem: Applications continue to grow in data size

Need to increase TLB size to keep the pace for performance

Nehalem adds new low-latency unified 2<sup>nd</sup> level TLB

	# of Entries
<b>1<sup>st</sup> Level Instruction TLBs</b>	
Small Page (4k)	128
Large Page (2M/4M)	7 per thread
<b>1<sup>st</sup> Level Data TLBs</b>	
Small Page (4k)	64
Large Page (2M/4M)	32
<b>New 2<sup>nd</sup> Level Unified TLB</b>	
Small Page Only	512

## Fast Unaligned Cache Accesses

Two flavors of 16-byte SSE loads/stores exist

- Aligned (MOVAPS/D, MOVDQA) -- Must be aligned on a 16-byte boundary
- Unaligned (MOVUPS/D, MOVDQU) -- No alignment requirement

Prior to Nehalem

- Optimized for Aligned instructions
- Unaligned instructions slower, lower throughput -- Even for aligned accesses!
  - Required multiple uops (not energy efficient)
- Compilers would largely avoid unaligned load
  - 2-instruction sequence (MOVSD+MOVHPD) was faster

Nehalem optimizes Unaligned instructions

- Same speed/throughput as Aligned instructions on aligned accesses
- Optimizations for making accesses that cross 64-byte boundaries fast
  - Lower latency/higher throughput than Core 2
- Aligned instructions remain fast

No reason to use aligned instructions on Nehalem!

Benefits:

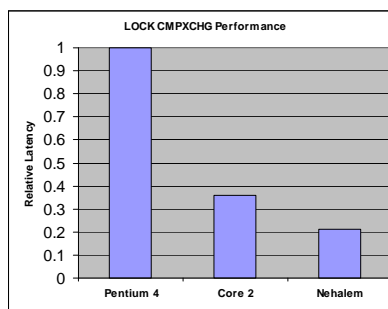
- Compiler can now use unaligned instructions without fear
- **Higher performance** on key media algorithms
- More **energy efficient** than prior implementations

## Faster Synchronization Primitives

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*

## Simultaneous Multi-Threading (SMT)

### SMT

- Run 2 threads at the same time per core

Take advantage of 4-wide execution engine

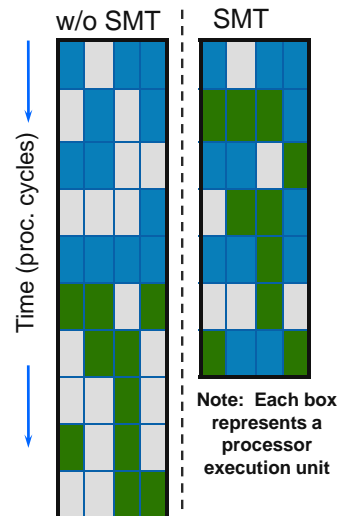
- Keep it fed with multiple threads
- Hide latency of a single thread

Most **power efficient** performance feature

- Very low die area cost
- Can provide significant performance benefit depending on application
- Much more efficient than adding an entire core

### Nehalem advantages

- Larger caches
- Massive memory BW



*Simultaneous multi-threading enhances performance and energy efficiency*

## SMT Implementation Details

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<sup>nd</sup> level TLB

Unaware

- Execution units

## Agenda

- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
  - Performance Features
  - Simultaneous Multi-Threading
- **Feeding the Engine**
  - New Memory Hierarchy
  - New Platform Architecture
- **Performance Acceleration**
  - Virtualization
  - New Instructions

## Feeding the Execution Engine

Powerful 4-wide dynamic execution engine

Need to keep providing fuel to the execution engine

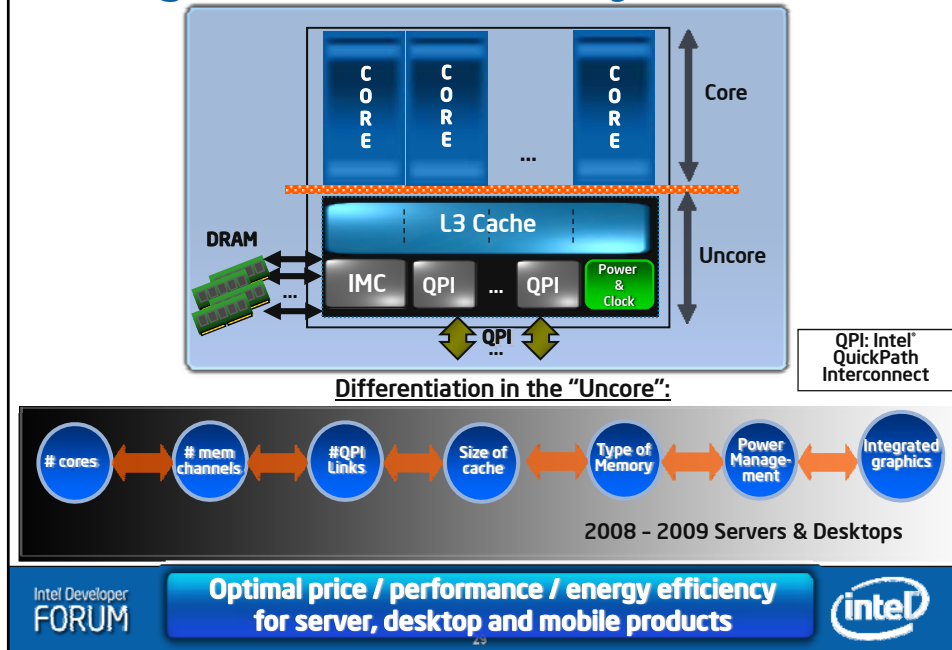
Nehalem Goals

- **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



## Intel Smart Cache – Core Caches

### New 3-level Cache Hierarchy

#### 1<sup>st</sup> level caches

- 32kB Instruction cache
- 32kB Data Cache
  - Support more L1 misses in parallel than Core 2

#### 2<sup>nd</sup> level Cache

- New cache introduced in Nehalem
- Unified (holds code and data)
- 256 kB per core
- **Performance:** Very low latency
- **Scalability:** As core count increases, reduce pressure on shared cache



## Intel Smart Cache -- 3<sup>rd</sup> Level Cache

New 3<sup>rd</sup> level cache

Shared across all cores

Size depends on # of cores

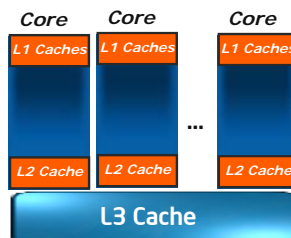
- Quad-core: Up to 8MB

- **Scalability:**

- Built to vary size with varied core counts
- Built to easily increase L3 size in future parts

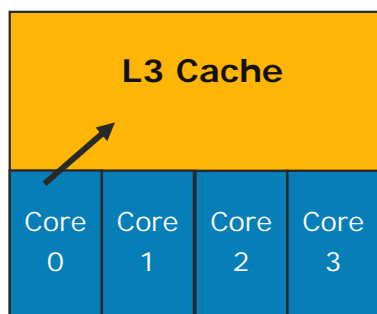
Inclusive cache policy for best **performance**

- Address residing in L1/L2 **must** be present in 3<sup>rd</sup> level cache

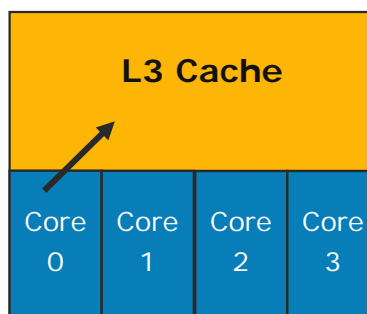


## Inclusive vs. Exclusive Caches – Cache Miss

### *Exclusive*



### *Inclusive*

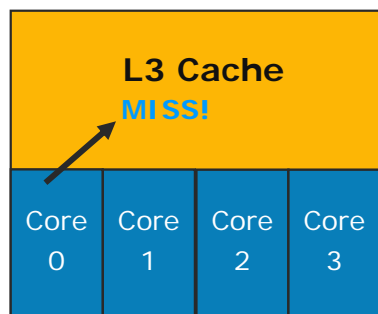


Data request from Core 0 misses Core 0's L1 and L2  
Request sent to the L3 cache

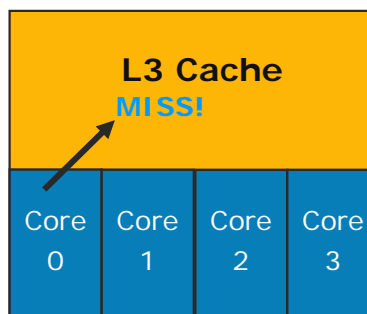


## Inclusive vs. Exclusive Caches – Cache Miss

*Exclusive*



*Inclusive*



Core 0 looks up the L3 Cache  
Data not in the L3 Cache

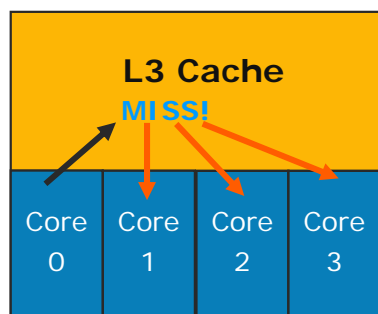
Intel Developer  
FORUM



33

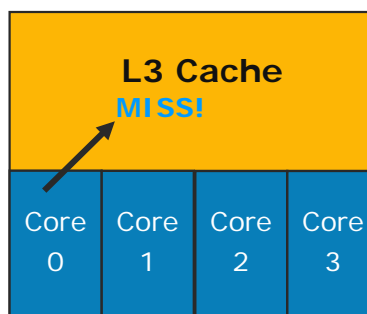
## Inclusive vs. Exclusive Caches – Cache Miss

*Exclusive*



Must check other cores

*Inclusive*



Guaranteed data is not on-die

Greater *scalability* from inclusive approach

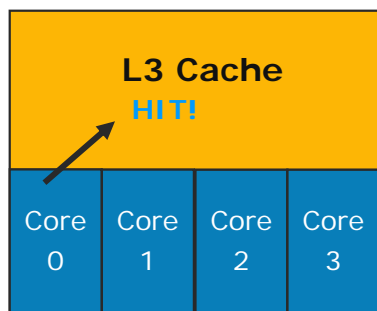
Intel Developer  
FORUM



34

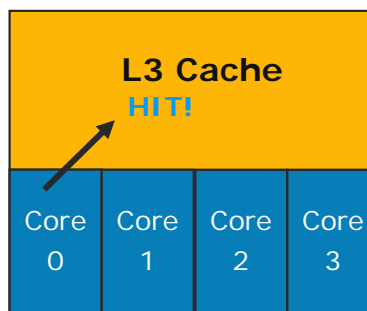
## Inclusive vs. Exclusive Caches – Cache Hit

### Exclusive



No need to check other cores

### Inclusive

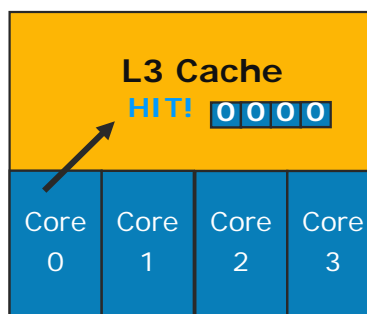


Data could be in another core  
**BUT** Nehalem is smart...

## Inclusive vs. Exclusive Caches – Cache Hit

### Inclusive

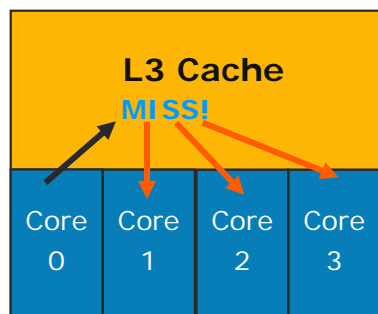
- Maintain a set of "core valid" bits per cache line in the L3 cache
- Each bit represents a core
- If the L1/L2 of a core *may* contain the cache line, then core valid bit is set to "1"
- No snoops of cores are needed if no bits are set
- If more than 1 bit is set, line cannot be in Modified state in any core



Core valid bits limit unnecessary snoops

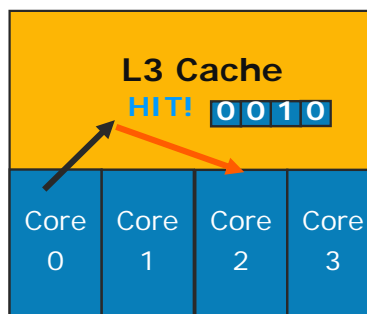
## Inclusive vs. Exclusive Caches – Read from other core

*Exclusive*



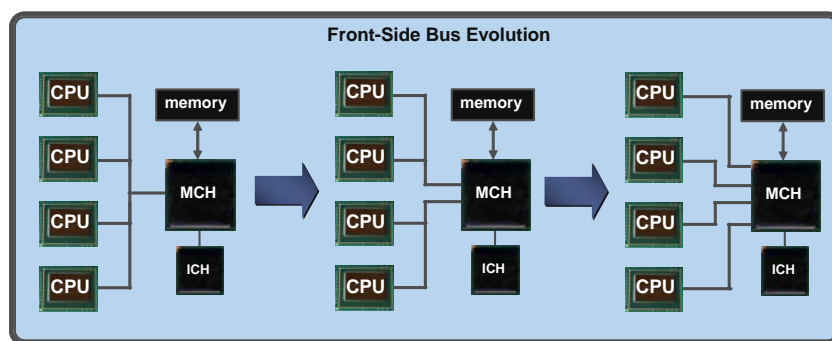
Must check all other cores

*Inclusive*



Only need to check the core whose core valid bit is set

## Today's Platform Architecture



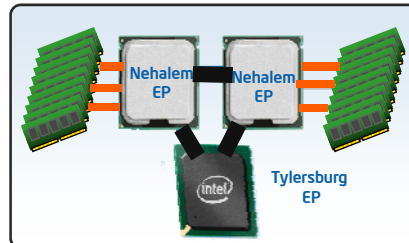
## Nehalem-EP Platform Architecture

### Integrated Memory Controller

- 3 DDR3 channels per socket
- Massive memory **bandwidth**
- Memory Bandwidth scales with # of processors
- Very **low memory latency**

### QuickPath Interconnect (QPI)

- New point-to-point interconnect
- Socket to socket connections
- Socket to chipset connections
- Build **scalable** solutions



*Significant performance leap from new platform*

## QuickPath Interconnect

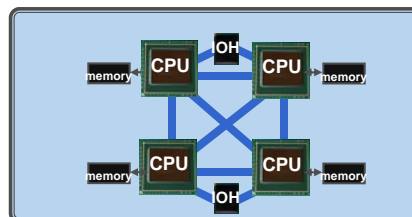
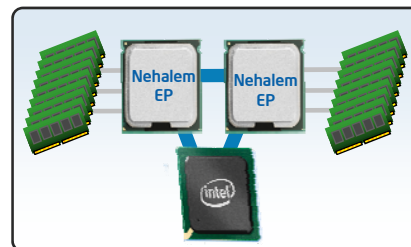
Nehalem introduces new QuickPath Interconnect (QPI)

**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



## Integrated Memory Controller (IMC)

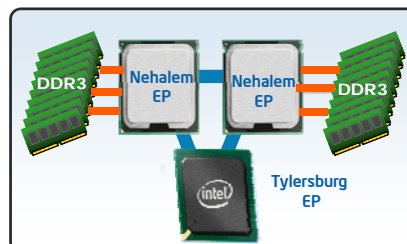
Memory controller optimized per market segment

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*

Intel Developer  
FORUM

41



## IMC Memory Bandwidth (BW)

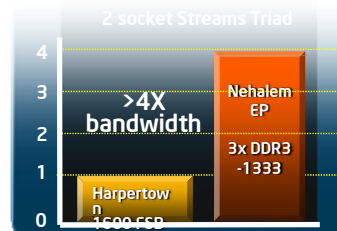
3 memory channels per socket

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*

Intel Developer  
FORUM

42



## Non-Uniform Memory Access (NUMA)

FSB architecture

- All memory in one location

Starting with Nehalem

- Memory located in multiple places

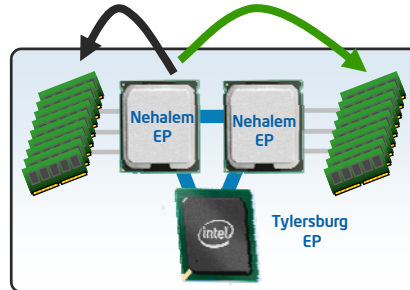
Latency to memory dependent on location

Local memory

- Highest BW
- Lowest latency

Remote Memory

- Higher latency



*Ensure software is NUMA-optimized for best performance*

Intel Developer  
FORUM

43



## Local Memory Access

CPU0 requests cache line X, not present in any CPU0 cache

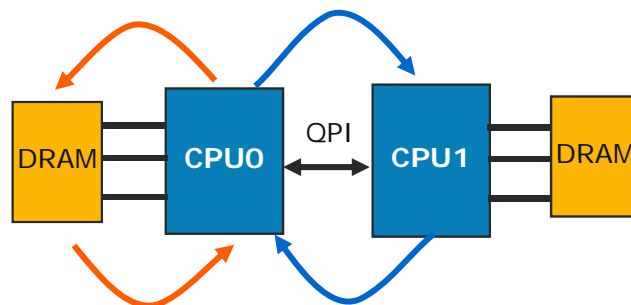
- CPU0 requests data from its DRAM
- CPU0 snoops CPU1 to check if data is present

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



Intel Developer  
FORUM

44

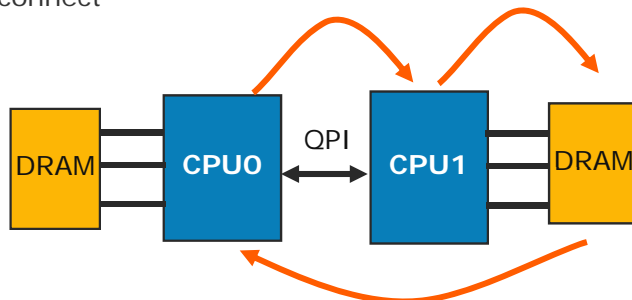


## Remote Memory Access

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



Intel Developer  
FORUM

45



## Memory Latency Comparison

**Low memory latency** critical to high performance

Design integrated memory controller for low latency

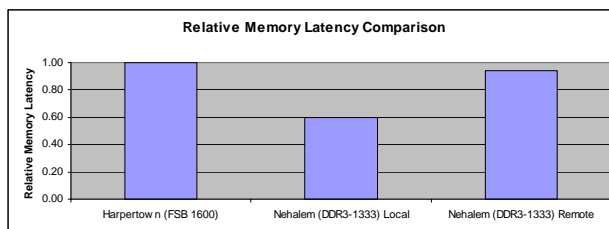
Need to optimize both local and remote memory latency

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
- NHM has lower latency regardless of mix



Intel Developer  
FORUM

46



## Agenda

- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
  - Performance Features
  - Simultaneous Multi-Threading
- **Feeding the Engine**
  - New Memory Hierarchy
  - New Platform Architecture
- **Performance Acceleration**
  - Virtualization
  - New Instructions

## Virtualization

To get best virtualized *performance*

- Have best native performance
- Reduce:
  - # of transitions into/out of virtual machine
  - Latency of transitions

Nehalem virtualization features

- Reduced latency for transitions
- Virtual Processor ID (VPID) to reduce effective cost of transitions
- Extended Page Table (EPT) to reduce # of transitions

*Great virtualization performance w/ Nehalem*



## Latency of Virtualization Transitions

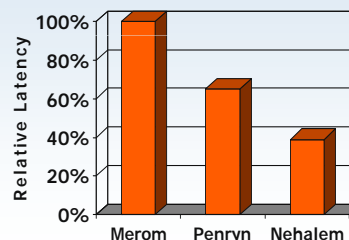
### Microarchitectural

- Huge latency reduction generation over generation
- Nehalem continues the trend

### Architectural

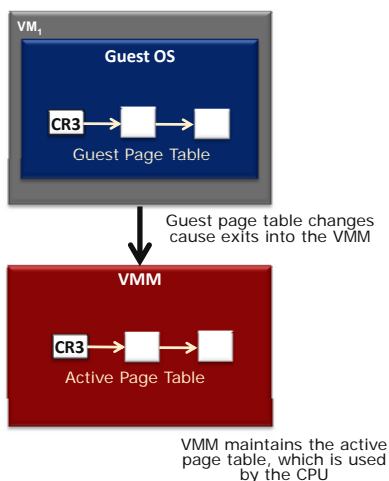
- Virtual Processor ID (VPID) added in Nehalem
- Removes need to flush TLBs on transitions

Round Trip Virtualization Latency



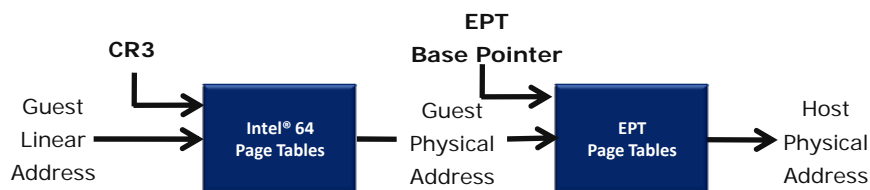
*Higher Virtualization Performance Through  
Lower Transition Latencies*

## Extended Page Tables (EPT) Motivation



- **A VMM needs to protect physical memory**
  - Multiple Guest OSs share the same physical memory
  - Protections are implemented through page-table virtualization
- **Page table virtualization accounts for a significant portion of virtualization overheads**
  - VM Exits / Entries
- **The goal of EPT is to reduce these overheads**

## EPT Solution



### Intel® 64 Page Tables

- Map Guest Linear Address to Guest Physical Address
- Can be read and written by the guest OS

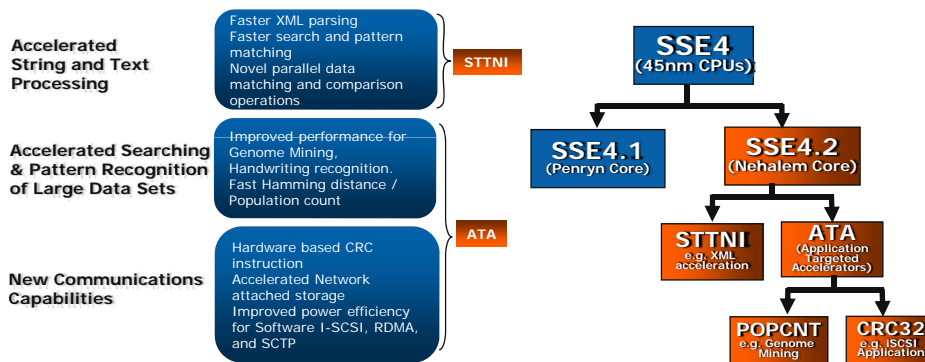
### New EPT Page Tables under VMM Control

- Map Guest Physical Address to Host Physical Address
- Referenced by new EPT base pointer

**No VM Exits due to Page Faults, INVLPG or CR3 accesses**

## Extending Performance and Energy Efficiency

### - SSE4.2 Instruction Set Architecture (ISA) Leadership in 2008



*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

Operates on strings of bytes or words (16b)

## 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\n  
Src2: atad tsT  
Mask: 10100000

## Ranges Instruction

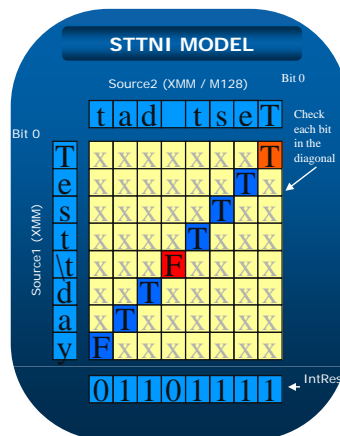
True if a character in Src2 is in at least one of up to 8 ranges in Src1

Src1: AZ'0'9zzz  
Src2: tad tseT  
Mask: 00100001

## Equal Ordered Instruction

Finds the start of a substring (Src1) within another string (Src2)

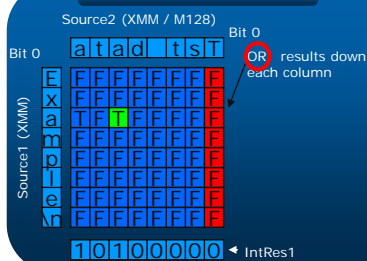
Src1: ABCA0XYZ  
Src2: SOBACBAB  
Mask: 00000010



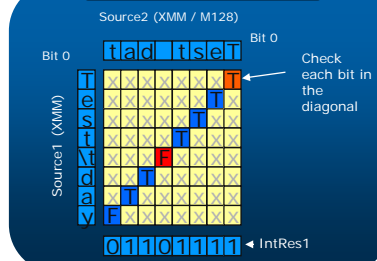
Projected 3.8x kernel speedup on XML parsing & 2.7x savings on instruction cycles

## STTNI Model

### EQUAL ANY



### EQUAL EACH



### RANGES



### EQUAL ORDERED



Intel Developer FORUM



## Example Code For strlen()

```

string equ [esp + 4]
mov ecx, string ; ecx
test ecx, 3 ; test if
je short main_loop
str_misaligned:
; simple byte loop until string is a
mov al, byte ptr [ecx]
add ecx, 1
test al, al
je short byte_3
test ecx, 3
jne short str_misaligned
add eax, dword ptr 0 ; 5 b
align 16 ; should l
main_loop:
mov eax, dword ptr [ecx] ; r
mov edx, 7efefeffh
add edx, eax
xor eax, -1
xor eax, edx
add ecx, 4
test eax, 81010100h
je short main_loop
; found zero byte in the loop
mov eax, [ecx - 4]
test al, al ; is it byte
je short byte_0
test ah, ah ; is it byte
je short byte_1
test eax, 0ff00000h ; is it byte

```

```

je short byte_2
test eax, 0ff000000h
; is it byte 3
je short byte_3
jmp short main_loop
; taken if bits 24-30 are clear and
; 31 is set
byte_3:
lea eax, [ecx - 1]
mov ecx, string
sub eax, ecx
ret
byte_2:
lea eax, [ecx - 2]
mov ecx, string
sub eax, ecx
ret
byte_1:
lea eax, [ecx - 3]
mov ecx, string
sub eax, ecx
ret
byte_0:
lea eax, [ecx - 4]
mov ecx, string
sub eax, ecx
ret
strlen endp
end

```

### STTNI Version

```

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, QWORD PTR[eax]
pcmpltri xmm2, xmm1, imm8
jnz     topofloop
endofstring:
add     eax, ecx
sub     eax, src
ret

```

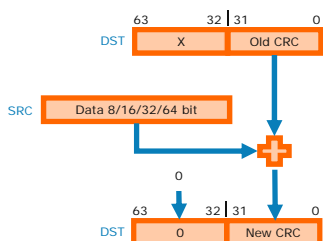
**Current Code:** Minimum of 11 instructions; Inner loop processes 4 bytes with 8 instructions

**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



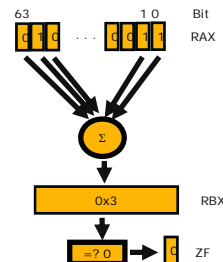
One register maintains the running CRC value as a software loop iterates over data.  
Fixed CRC polynomial = 11EDC6F41h

Replaces complex instruction sequences for CRC in Upper layer data protocols:

- iSCSI, RDMA, SCTP

### 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

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

Intel Developer  
FORUM



## CRC32 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("jecz 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

	32 - bit	64 - bit
Input Data Size = 48 bytes	6.53 X	9.85 X
Input Data Size = 4 KB	9.3 X	18.63 X

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

## 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
    - ✓ <smmmintrin.h> for SSE4.1
    - ✓ <nmmmintrin.h> for SSE4.2
- Intel Library Support
  - > XML Parser Library using string instructions will beta Spring '08 and release product 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
    - ✓ <smmmintrin.h> for SSE4.1
    - ✓ <nmmmintrin.h> for SSE4.2
  - > VC++ 2008 tools masm, msdis, and debuggers recognize the new instructions

## 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 will be published  
Intel Compiler will support settings for Nehalem optimizations

## Summary

Nehalem – The 45nm Tock

Designed for

- **Power Efficiency**
- **Scalability**
- **Performance**

Enhanced Processor Core

Brand New Platform Architecture

Extending ISA Leadership

## Additional sources of information on this topic:

NGMS002-Upcoming Intel® 64 Instruction Set Architecture Extensions

- **April 3 16:00 – 17:50**; Auditorium, 3rd floor

NGMC001-Next Generation Intel® Microarchitecture (Nehalem) and New Instruction Extensions - Chalk Talk

- **April 3 17:50 – 18:30**; Auditorium, 3rd floor

Demos in the Advance Technology Zone on the 3<sup>rd</sup> floor –

More web based info: <http://www.intel.com/technology/architecture-silicon/next-gen/index.htm>

Intel Developer  
FORUM

61



## Session Presentations - PDFs

The PDF of this Session presentation is available from our IDF Content Catalog:

<https://intel.wingateweb.com/SHchina/catalog/controller/catalog>

These can also be found from links on [www.intel.com/idf](http://www.intel.com/idf)

Intel Developer  
FORUM

62



## Please Fill out the Session Evaluation Form

Put in your lucky draw coupon to win the  
prize at the end of the track!

You must be present to win!

Thank You for your input, we use it to improve future  
Intel Developer Forum events

Intel Developer  
FORUM

63



## Legal Disclaimer

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.

**[ADD ANY CODE NAMES FROM FOILS]** 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, **[ADD ANY OTHER INTEL TRADEMARKS OR LOGO IN FOILS]** 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 © 2008 Intel Corporation.

Intel Developer  
FORUM

64





## 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.