EMBEDDED APPLICATIONS: MIGRATING TO INTEL X86 ARCHITECTURE

NAVEEN G V
SOFTWARE & SERVICES GROUP
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804
Agenda

• Introduction
• About Intel Architecture
• Application porting considerations
• Taking advantage of Intel Software Tools for application migration
• Summary
• References
Intel® Architecture for IoT/Embedded Delivering Growth Beyond the PC & Server
Intel Architecture for Embedded products

The foundation for all computing
Silvermont Microarchitecture
in Next Generation Intel Products

Enabling Exceptional Experiences
Core Block Diagram
Core Block Diagram
Silvermont Architecture Highlights

**Better Performance**
- Out of Order Execution engine
- New multi-core and system fabric architecture
- New IA instructions extensions (Core™ Westmere level)
- New security and virtualization technologies

**Better Power Efficiency**
- Wide dynamic operating range
- Enhanced active and idle power management

**Full Advantage of Intel 22nm SoC Process Technology**
- 3-D Tri-gate transistors tuned for SoC products
- Architecture and design co-optimized with the process

*Building upon Intel's Strengths and Expertise in Defining Microarchitecture*
**Migration Guide Steps**

**Part 1** – Execute code correctly on one Intel® architecture (IA) core:
1. Port code to target OS.
2. Execute code correctly on one IA core.
3. Optimize for power and performance on one IA core.

**Part 2** – Implement additional IA benefits:
4. Apply technology updates: Multi-core, Virtualization...
5. Optimize code for power and multi-core IA performance.
Embedded Products – Software Stack

- **SW Tools** – Often drive decisions of the architecture conversion, such as OS and real-time support, and cost requirements.
- **Applications / Libraries** – Programming language, OS and processor architecture support, APIs.
- **OS's / Device Drivers / Graphics Drivers** – Processor/platform/tools support, real-time, embedded requirements.
- **Firmware** - Other architectures provide boot loaders for free. Intel provides commercial solutions from IBVs and solutions to assist boot loader development.
- **JTAG** (On-chip debuggers) – Used for board bring up and useful for all types of software debugging, especially time critical.

Various levels of software in the system stack from controlling the hardware to performing end-application functions.
Embedded Application Migration Planning

Feasibility study
- Survey the current SW stack to determine portability of existing code and available solutions for migration.
- Determine scope and effort of the based on survey of the SW stack.
- Understand requirements and available solutions throughout the SW stack.

Project plan
- Address feasibility study findings, choose solutions, and ensure to include any training that may be required.
- Apply to the recommended steps in the Migration Guide.

Project goals & requirements will drive the feasibility, scope and effort required for the SW migration.
Application Migration: Considerations

- Assembly language code.
- May not use OS abstraction layer. Makes changing OS's more difficult.
- May interface directly to devices instead of using OS APIs.
- May assume (hard code) endianness memory architecture.
- Serial code (not multi-core ready).
<table>
<thead>
<tr>
<th>Instructions</th>
<th>Instructions supported from Intel® Architecture are different compare to non IA, for some instructions there is no one-to-one (RISC to Intel architecture) equivalent. Refer to <a href="https://www.intel.com/content/www/us/en/develop/documentation.html">Intel® 64 and IA-32 Architectures Software Developer Manuals</a>.</th>
</tr>
</thead>
</table>
| Alignment    | • Intel architecture instructions vary in size and therefore do not require alignment.  
• Pointer alignment, E.g. 1 byte aligned on x86  
• Structure size and alignment. E.g. a struct with 3 characters on IA is 3 bytes; |
| Signed vs. unsigned char | char is signed on x86. CHAR_MIN and CHAR_MAX have different values on x86. gcc compiler can force all char types to be signed: -fsigned-char |
## Intel® Architecture - Instruction set

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Byte order (Endianness)</strong></td>
<td>Intel architecture is little-endian.</td>
</tr>
<tr>
<td><strong>Bit Fields</strong></td>
<td>Intel architecture is “normal bit ordered” aka “up bit ordered.”</td>
</tr>
<tr>
<td></td>
<td>(MSB is 0 for big-endian, LSB is 0 for little-endian – always the case for Intel architecture).</td>
</tr>
<tr>
<td><strong>Divide-by-zero</strong></td>
<td>On Intel architecture, executing this operation is fatal.</td>
</tr>
<tr>
<td><strong>Calling Conventions</strong></td>
<td>For Intel architecture, arguments are passed on the stack and registers for Intel® 64.</td>
</tr>
<tr>
<td><strong>Specified by ABI</strong></td>
<td>Intel architecture has fewer registers, also local variables may be stored on the stack as well.</td>
</tr>
<tr>
<td><strong>Calling Conventions</strong></td>
<td>For Intel architecture, arguments are passed on the stack and registers for Intel® 64.</td>
</tr>
<tr>
<td><strong>Specified by ABI</strong></td>
<td>C-function's calling conversion: IA uses left most argument first.</td>
</tr>
</tbody>
</table>
Application Migration : Best Known Methods

- Rewrite assembly code in high level language, and use translation tools as needed.
- For high level programming languages, simply recompile for IA as required using **Intel® C++ Compiler**
- Adopt multi-core supported libraries, such as **Intel® Integrated Performance Primitives** and **Intel® Math Kernel Library**.
- Implement multithreading using **Intel® Cilk™ Plus** and **Intel® Threading Building Blocks**
- Analyze the performance of your application using **Intel® VTune™ Amplifier**
Porting Alternatives to Intel Architecture

**Manual Conversion**
If **Performance** is a higher priority than **TTM/$**, then manual conversion obtains the most efficient code, **but requires the most work.**

**Use Bi-Endian Compiler**
If the program is specified cleanly in bi-endian features, the compiler can generate code correctly while maintaining high-performance.

**Recompile**
If code is endian neutral, it may just need a recompile. For example the EEMBC benchmarks are endian-agnostic.

Multiple migration options to address endianness
Intel® C++ Compiler Standard Edition for Embedded Systems with Bi-Endian Technology

What is it?

- Special compiler that enables faster architecture migration to Intel® Architecture
- Maintains Endianness of code
- Inserts bswap instructions into binary when necessary

The Bi-Endian Compiler enables applications to execute with big-endian semantics on little-endian processors
Bi-Endian Compiler Functionality

Code specified as Big-Endian...

- Behaves as though executing on a Big-Endian machine
- bswaps inserted at crucial points
  - E.g. after variable stores and loads
- When possible instruction bswap is eliminated to increase performance
  - E.g. variables without address taken and not part of a union

Code specified as Little-Endian...

- Runs as native code
- No byte swaps necessary

Minimal performance impact

- Intelligently inserts byte swaps only when necessary

Can mix & match Big-Endian and Little-Endian code

Reduced implementation and validation effort
Who is using Bi-Endian Compiler?

Networking & Telecommunication

• Porting code from non-IA (e.g. PPC) to IA
• Manual code migration was not an option
  • Large source code base
  • Difficult to debug
  • Required big resource commits
  ➞ Delayed or canceled product releases
• Millions of lines of legacy code ported to IA
• Today, customers successfully use Bi-Endian Compiler to release products based on IA
Intel® C++ Compiler- Compatibility

• Multiple OS and Cross Compilation Support
  • HOST OS: Windows*, Linux*
  • TARGET OS: Linux*, WindRiver Linux*, Yocto Project, Tizen*, Android*, customized Linux* and more

• Integration with major IDE for different targets
  • Eclipse* Integration on Linux*
  • WindRiver Linux Workbench Integration
  • Android NDK Integration
  • Windows* Visual Studio* integration for Android

• No source-level changes for quick & simple gains
  • Just recompile for a quick ROI!

• Source and binary compatibility
  • Compatible with GNU compiler
Intel® C++ Compiler- Performance

- Code generation tuned for latest micro architecture, Latest Instructions for newest IA processors (SSE, AVX, AVX2)

Support the newest Intel Architecture features

- Latest Instructions for newest IA processors (SSE, AVX, AVX2)
- Code generation tuned for latest micro architecture
- Advanced optimization features
- Highly Optimized Compiler libraries – (libirc, libimf, libsvml, etc)
- Good optimization report helps performance tuning
Advanced Optimizations & Standards Support

Maximum application performance through advanced optimization features:

- Interprocedural Optimization (IPO)
- Profile-Guided Optimization (PGO)
- Full support for Intel Streaming SIMD Extensions 3, Intel SSE 4
- Automatic Vectorizer
- Runtime support for Intel® processor generations: processor dispatch

ISO and ANSI C/C++ compatible

Source and binary compatible with GCC, including most GNU C language extensions
**Auto-Vectorization**

**SIMD – Single Instruction Multiple Data**

- **Scalar mode**
  - one instruction produces one result

- **SIMD processing**
  - with SSE or AVX instructions
  - one instruction can produce multiple results

```c
for (i=0; i<=MAX; i++)
    c[i]=a[i]+b[i];
```

![Diagram showing scalar and SIMD operations](image)
Intel C++ Compiler works with GNU binary utilities

- Intel C++ Compiler does not provide real bin utilities like "ld", "ar", but provide the driver to invoke the "ld" or "ar" automatically
- icc/icpc: first calls real Intel compiler(mcpcom) for compilation, then invokes the GNU "ld" if necessary
- xild: first calls real Intel compiler for optimization, then invokes GNU "ld"
- xiar: first calls real Intel compiler for optimization, then invokes GNU "ar"
Porting from GCC to ICC compiler (I)

• Change the compilation command using predefined platform configuration with –platform option
  • `<prefix, e.g. i586-poky-linux>-gcc → icc -platform=<val>`
  • `<prefix, e.g. i586-poky-linux>-g++ → icpc -platform=<val>`
  • `<prefix, e.g. i586-poky-linux>-ar → xiar -qplatform=<val>`
  • `<prefix, e.g. i586-poky-linux>-ld → xild -qplatform=<val>`
• This normally can be changed with environment variable like CC, CXX, AR, LD, or in the Makefile
Porting from GCC to ICC compiler (II)

- Starting from Intel C++ Compiler 15.0 (Intel System Studio 2015), use –gnu-prefix and --sysroot option
  - icc –gnu-prefix=i586-poky-linux--sysroot=<val>
  - icpc –gnu-prefix=i586-poky-linux--sysroot=<val>
  - xiar –qgnu-prefix=i586-poky-linux--sysroot=<val>
  - xild –qgnu-prefix=i586-poky-linux--sysroot=<val>
- More details on –gnu-prefix and –sysroot, refer to compiler documents, and online articles
Intel® Integrated Performance Primitives (Intel® IPP)

Enhances Developer Productivity
- Optimized image, signal and data processing routines
- Thread-safe functions

Industry Leading Performance
- Instruction set-level optimizations (SIMD)
- Efficient parallelism on multicore platforms

Support for Latest Processor Architectures
- Optimized for current multi-core processors
- Applications benefit seamlessly

Cross Platform and Operating System Support

**Multi OS:**
- Windows®
- Linux®
- OSX®
- Android®
- VxWorks®

**Multi Platform:**
- Mobile and Embedded (Intel® Quark, Intel® Atom™)
- Tablet (Intel® Atom™, Intel® Core™)
- Ultrabook/PC (Intel® Core™)
- Servers and Workstations (Intel® Xeon® and Intel® Xeon® Phi™)

Cross platform support and optimized for current and future processors

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Fastest and most used math library

Intel® Math Kernel Library (Intel® MKL)

- Highly optimized threaded math routines
- Optimizations using Intel® AVX and the new Intel® Advanced Vector Extensions 2 (Intel® AVX2)
- Use Intel® MKL on embedded Linux* targets in communications, medical, industrial, and other segments
- Use Intel MKL with Intel® C++ Compiler and GCC*

#1 used math library in the world
Source: Evans Data 2011-2013 WW Developer Surveys

Unleash processor performance with Intel MKL
# C++ template library for task parallelism

**Intel® Threading Building Blocks (Intel® TBB)**

## Generic Parallel Algorithms
- parallel_for
- parallel_reduce
- parallel_for_each
- parallel_do
- parallel_invoke
- parallel_sort
- parallel_deterministic_reduce
- parallel_scan
- parallel_pipeline
- pipeline

## Flow Graph
- graph
- continue_node
- source_node
- function_node
- multifunction_node
- overwrite_node
- write_once_node
- limiter_node
- buffer_node
- queue_node
- priority_queue_node
- sequencer_node
- broadcast_node
- join_node
- split_node
- indexer_node

## Concurrent Containers
- concurrent_unordered_map
- concurrent_unordered_multimap
- concurrent_unordered_set
- concurrent_unordered_multiset
- concurrent_hash_map
- concurrent_queue
- concurrent_bounded_queue
- concurrent_priority_queue
- concurrent_vector
- concurrent_lru_cache (preview)

## Synchronization Primitives
- atomic
- mutex
- recursive_mutex
- spin_mutex
- spin_rw_mutex
- speculative_spin_mutex
- speculative_spin_rw_mutex
- queuing_mutex
- queuing_rw_mutex
- null_mutex
- null_rw_mutex
- reader_writer_lock
- critical_section
- condition_variable
- aggregator (preview)

## Task Scheduler
- task
- task_group
- structured_task_group
- task_group_context
- task_scheduler_init
- task_scheduler_observer
- task_arena

## Timers and Exceptions
- tick_count
- tbb_exception
- captured_exception

## Threads
- Thread
  - combinable
  - enumerable_thread_specific

## Thread Local Storage
- aligned_space
- memory_pool (preview)

## Memory Allocation
- tbb_allocator
- scalable_allocator
  - cache_aligned_allocator
  - zero_allocator

---

**Optimization Notice**

Copyright © 2015, Intel Corporation. All rights reserved.

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

---

**Simplify Parallelism with a Scalable Parallel Model**
Performance Profiler
Intel® VTune™ Amplifier for Systems

Get the Tuning Data You Need
- Low overhead “hotspot” analysis with call stacks
- Advanced analysis for cache, branching, ...

Find Answers Fast
- Powerful analysis & data mining
- Results mapped to C/C++ or Java source

Easy to Use
- Remote analysis from the User Interface
- Windows or Linux Host analyzes Linux or Android target

Optimize Your Software Performance after porting application on IA
Energy and Power Profiler for System Software Developers

Intel® Energy Profiler

Optimize Software for Extended Battery Life

Find the Cause of Wake Ups That Waste Energy

- Interrupts mapped to the IRQ/device
- Timers mapped to the scheduling process
- Data correlated with Android Wake Locks

Available now for Linux and Android

Get Actionable Data to Extend Battery Life

Requires specific SOCs. On Android, a rootable OS is required with version compatible device drivers. See release notes for details.
What’s Included in Intel® System Studio

**Build & Optimize**

**Systems, Embedded Applications**

- Intel® System Debugger
- UEFI, OS, drivers through JTAG
- Intel® VTune™ Amplifier
- Intel® Energy Profiler
- Intel® Frame Analyzer
- Intel® Platform Analyzer
- Intel® System Analyzer
- WinDbg* support
- Intel® Math Kernel Library
- Intel® Threading Building Blocks
- Intel® Integrated Performance Primitives
- Intel® Enhanced GDB*
- Eclipse*-based, Visual Studio*
- Intel® C++ Compiler incl. Intel® Graphics Technology offload

**Analyse**

**Performance**

- CPU/GPU workloads
  - In real-time
  - Offline and detailed
- Code performance on CPU
time-, event-based
- Graphics performanceOpenGL ES, DirectX
- System-wide power efficiency
  - Wake-up, sleep-state, frequency, temp.
- Application robustness
  - Memory leaks

**Power**

**Correctness**

- Composer Edition
- Professional Edition
- Ultimate Edition

**Debug & Trace**

**System Software**

- Intel® System Debugger
- UEFI, OS, drivers through JTAG
- Intel® Debug Extensions for WinDbg*
- Windows* stack
  - WinDbg* over JTAG

Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
## Intel® System Studio: Editions, Components, and Operating Systems

### Target Operating Systems

<table>
<thead>
<tr>
<th>Category</th>
<th>Component</th>
<th>Linux* 1, 2</th>
<th>Android* 2</th>
<th>Windows*</th>
<th>VxWorks* 3</th>
<th>FreeBSD*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Composer</td>
<td>Composer</td>
<td>Composer</td>
<td>Composer</td>
<td>Composer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Professional</td>
<td>Professional</td>
<td>Ultimate</td>
<td>Professional</td>
<td>Ultimate</td>
</tr>
<tr>
<td>Host Operating Systems</td>
<td>Linux* Windows*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Linux* Windows*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Windows*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Integrated Development Environment</td>
<td>Eclipse*, Workbench*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Eclipse*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Visual Studio*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Workbench*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Eclipse*</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Compiler &amp; Libraries</td>
<td>Intel® C++ Compiler</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Integrated Performance Primitives</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Math Kernel Library</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Threading Building Blocks</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>System &amp; Application Debuggers</td>
<td>Intel® System Debugger 4</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Debug Extensions for WinDbg* 4</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel®-enhanced GDB* Application Debugger</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Debugger for Heterogeneous Compute</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>Performance, Power &amp; Correctness Analyzers</td>
<td>Intel® VTune™ Amplifier 6</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Energy Profiler</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Intel® Inspector</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>System Analyzer</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Platform Analyzer 5</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td></td>
<td>Frame Analyzer 6</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

### Notes

1. Linux*, Embedded Linux, Wind River* Linux*, Yocto Project*
2. Linux* and Android* target support available in a single product
3. Available from Wind River* with VxWorks*
4. Via Intel® ITP-XDP3 probe, OpenOCD*, Intel® SVT Closed Chassis Adapter* and EDKII* for UEFI*
5. Available for Windows* host
6. Also available for OS X* host as a separate download
7. Intel® System Debugger provides VxWorks* OS awareness – available with Ultimate Editions

---

**Optimization Notice**

Copyright © 2015, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.
Embedded Applications:
Migrating to Intel x86 Architecture

Intel provides the building blocks you need to develop hardware and software for the Internet of Things

Intel System Studio is a complete tools suite to build and optimize your application on Intel Architecture

- Increases performance with expertly optimized compiler and libraries
- Enhances power efficiency and performance with enhanced analyzers
- Improves developer productivity with expanded usability and capabilities

MIGRATE SMARTER CODE — SMARTER TO IA, WITH INTEL SYSTEM STUDIO

Learn more at: http://intel.ly/system-studio
Educating with Webinar series about “2016” tools
Expert talks about the new features
Series of live webinars Sept 22\textsuperscript{nd} – Oct 21\textsuperscript{st}, 2015
Attend live, or watch after the fact.


<table>
<thead>
<tr>
<th>Topics</th>
<th>Time (Asia) - Tuesdays</th>
<th>Time (USA) – Wednesdays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrating embedded applications to Intel x86 Architecture</td>
<td>Oct 13\textsuperscript{th}, 11 PM (PST)</td>
<td>Oct 14\textsuperscript{th}, 9AM (PST)</td>
</tr>
<tr>
<td>Get Deep System Insight for 6\textsuperscript{th} Generation Intel® Core™ Processors</td>
<td>Oct 20\textsuperscript{th}, 11PM (PST)</td>
<td>Oct 21\textsuperscript{st}, 9AM (PST)</td>
</tr>
</tbody>
</table>
References

- **PowerPC* to Intel® Architecture Migration**

- **Translation of PowerPC*/AltiVec* SIMD Macros to IA32/SSE**

- **Tool for Translating PowerPC*/AltiVec* SIMD Macros to IA32/AVX**

- **Convert Commercial Automation Systems to Intel® Architecture**

- **Using DSP Software Conversion Tools for Intel® Processors**

- **Moving AltiVec* Signal Processing Apps to Intel® Processors**
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804