

# Parallel Programming Methods and Tools

*Hubert Haberstock*  
*Technical Consulting Engineer*



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# Agenda



- 10:00 Welcome & Introduction
- 10:15 Developing for Multi-Core on Windows with Intel® Parallel Studio (Intel)
- 11:00 Intel® Parallel Studio - Un caso di studio (C. Fiorillo)
- 12:00 Parallel Architectures and Parallel Programming, today and tomorrow (Intel)
- 13:00 - 14:00 Lunch break*
- 14:00 Parallel Programming Methods and Tools (Intel)**
- 14:45 Intel Parallel Studio – Creating parallel code (Intel)
- 16:00 Intel Parallel Studio - Debug ed ottimizzazione del codice parallelo (C. Fiorillo)
- 17:00 Wrap up, seminar evaluation, Q&A



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



## Simplifies threading for improved developer productivity

- "Think Parallel" and "Code Parallel" without writing the low-level thread management

Best choice because:

### Easier Parallel Implementation

- Included libraries and built-in capabilities reduce amount of code and increase scalability

### High Performance

- Built-in optimization features and libraries extract the most from multicore processors

### Interoperable

- Supports variety of threading methods and compilers
- Seamless integration to Microsoft Visual Studio (Windows\*)
- Command line, source and binary compatibility with GCC (Linux\*, Mac OS\*X)
- Integrates in XCode development environment (Mac OS\* X)



# More than a C++ Compiler ...



Simplifies threading for improved developer productivity

- Simple concurrent functionality (`__task/__taskcomplete`)
- Vectorization support for SSE2/SSE3/SSSE3/SSE4
- OpenMP™ 3.0, Auto-Parallelization
- Cilk (New generation tools)
- Intel® Parallel Debugger Extensions - A Plug-in to Visual Studio\*
- Intel® IDB Debugger (Linux\*, Mac OS\* X)
- Intel® Threading Building Blocks
- C++ lambda function support
- Intel® Integrated Performance Primitives (IPP)
- Intel® Math Kernel Library (MKL) (HPC tools only)
- Parallel build (`/MP`) feature
- `/MP` enables use of multiple cores to speed up build
- Diagnostics to help develop parallel programs (`/Qdiag-enable:thread`)



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



# Compatibility with Microsoft

## Source and binary compatible

- Full compatibility with .Net\* Build Environment
- Binary compatibility
- Name Mangling and Calling Convention
- Debug Format Compatibility
- Mix and match object files or DLLs

## Limitations

- No support of *.pch* file; instead use *.pch* file
- No support for attributed programming, managed C++





- Optimized library for multiple problem domains
  - Image Processing
    - JPEG, JPEG2000
    - Image Search Descriptors (MPEG7), Color layout, Edge histogram
    - 3D support
  - Audio/Video
    - Codecs - H.264, MPEG4
    - Video enhancement - Denoise, Deinterlace, Demosaic
    - Microsoft\* RT Audio
  - Data Compression
    - LZO, zlib, gzip, bzip2
- Single and Multi-core processor optimizations and support
  - Intel® Atom™ & Intel® Core™ processor family including Intel® Core™ i7™



# (Auto) Vectorization

using the SIMD registers, let the  
Compiler parallelize ...



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.



- For vectorization we add eight *128-bit* registers known as **XMM0-XMM7**. For the 64-bit extensions additional eight registers **XMM8-XMM15** are added
- Operations on this registers are an addition to the X86 instruction set
- The Intel® Compiler can automatically generate these instructions called SSEx (Streaming SIMD Extensions)

# Compiler Based Vectorization



## Group 1: /Qx<extension>, -x<extension>

- Targeting Intel® processors; specific optimizations for Intel® processors
- Compiler will try to make use of all instruction set extensions up to and including <extension>;
- Processor check added to main-program
- Application will not start in case feature not available

## Group 2: -arch:<extension>

- Compatible to MS compiler
- Adds Intel® processor check only for “newer: extensions SSSE3, SSE4.1, ...
- Disables Intel-specific optimizations
- Should be used in case application is supposed to run on Intel and non-Intel processors
- Missing check can cause application to fail in case extension not available
- For legacy processors without SSE2 (Pentium® III etc) , use extension -arch: IA32

## Group 3: /Qax<extension>, -ax<extension>

- Dual-code paths – a ‘generic’ and ‘optimized’ path
- ‘optimized’ path for Intel® processors defined by <extension>
- ‘generic’ code path defaults to -arch:SSE2
- ‘generic’ code path can be changed using additionally switch of group 1 or 2



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# SIMD – SSEx support



**SSE**



**4x floats**

**SSE-2**



**2x doubles**



**16x bytes**



**8x 16-bit shorts**



**4x 32-bit integers**



**2x 64-bit integers**



**1x 128-bit(!) integer**



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

- Example vector addition, were you add two single precision, 4-component vectors together
- Using x86 operation requires four floating point additions using a FADD instructions

```
sum_vector [x] = vector1[x] + vector2[x];  
sum_vector [y] = vector1[y] + vector2[y];  
sum_vector [z] = vector1[z] + vector2[z];  
sum_vector [w] = vector1[w] + vector2[w];
```

- Using SSE registers a single 128 bit *packed-add* instruction can be used to replace the four scalar additions:

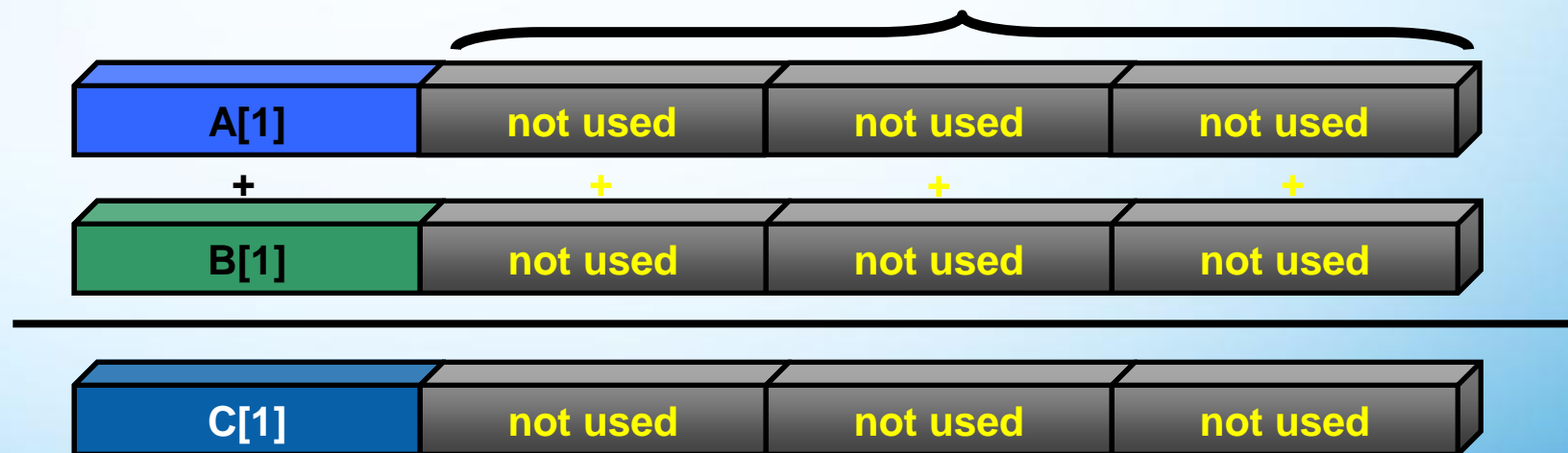
```
movaps xmm0, vector1  
addps  xmm0, vector2  
movaps sum_vector, xmm0
```

# Using SSE3 - Your Task: Convert This...



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

e.g. 3 x 32-bit unused integers

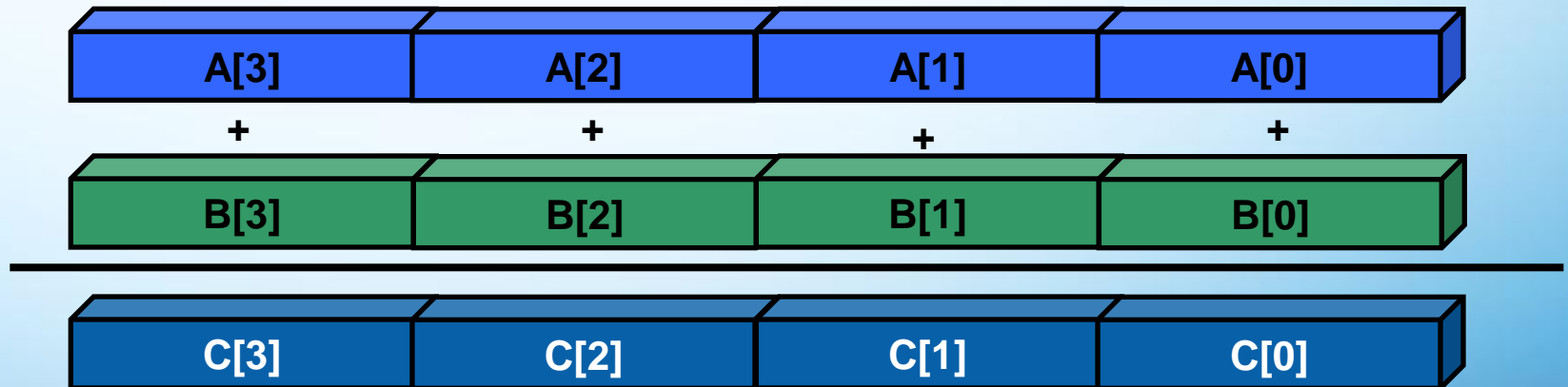




# ...Into This...

- Processor switch is vectorizing loops for fp and scalar ops.
- Usage: Linux\* `-xSSE3` Windows\* `/QxSSE3`

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







# Example: Vectorization Report

Driver.c

Driver.c(64) : (col. 2) remark: loop was not vectorized: **contains unvectorizable statement** at line 65.

Driver.c(74) : (col. 2) remark: **LOOP WAS VECTORIZED.**

Driver.c(39) : (col. 2) remark: **LOOP WAS VECTORIZED.**

Driver.c(28) : (col. 10) remark: loop was not vectorized: **statement cannot be vectorized.**

Driver.c(30) : (col. 3) remark: **LOOP WAS VECTORIZED.**

Driver.c(12) : (col. 2) remark: loop was not vectorized: **not inner loop.**

Driver.c(14) : (col. 14) remark: loop was not vectorized: **statement cannot be vectorized.**

Driver.c(18) : (col. 3) remark: loop was not vectorized: **not inner loop.**

Driver.c(19) : (col. 4) remark: **LOOP WAS VECTORIZED.**

Multiply.c

Multiply.c(7) : (col. 2) remark: loop was not vectorized: **not inner loop.**

Multiply.c(9) : (col. 3) remark: loop was not vectorized: **unsupported loop structure.**

-out:MatVector.exe



Software & Services Group, Developer Products Division

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.

# The future of SIMD



- Now we have SSEx with 128 bit vectors
- Next is AVX with 256 bit vectors (Sandy Bridge)
- Or LRBni with 512 bit vectors
- ...



# Legal Disclaimer



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.

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. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference [www.intel.com/software/products](http://www.intel.com/software/products).

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright © 2010. Intel Corporation.

<http://intel.com/software/products>



**Software & Services Group, Developer Products Division**

Copyright © 2010, Intel Corporation. All rights reserved.

\*Other brands and names are the property of their respective owners.