# Intel compiler overview and optimizations

## Intel – UNN Performance Optimization Winter Camp

Khalik Kasimov
Technical Consulting Engineer
khalik.kasimov@intel.com

# Требования ПО для практики

- Microsoft Visual Studio* 2019 или 2022 Community/Enterprise/Professional + 'Разработка классических приложений на C++' ('Desktop development with C++')
- Intel oneAPI Base Toolkit version 2022.1 for Windows
- Intel oneAPI HPC Toolkit version 2022.1 for Windows
- Intel oneAPI command prompt for Intel 64 (IA32) for Visual Studio 2019 (2022) запускается

# Подготовка к работе

- Загрузить и распаковать архив с кодом:

  https://github.com/ivorobts/compiler-optimization/archive/refs/heads/master.zip

- Настроить Intel C++ компилятор:
  - Опция 1: Запустить пакетный файл <oneAPI install-dir>\setvars.bat [<arg1>] [<arg2>]
    - arg1 (опциональный): intel64 или ia32
    - arg2 (опциональный): vs2019 или vs2017
  - Опция 2 (рекомеднованная компанией Intel): запустить Intel oneAPI command prompt for <target architecture> for Visual Studio <year>

# Тестовое задание 1:
# Базовые оптимизации Intel C++ компилятора

# Тестовое задание 1

- Перейдите в папку compiler-optimization-master\basic
- Изучите код

fx.c:

```
#include <stdio.h>


double f(double x){
  double ret;
  ret = 4.0 / (x*x + 1.0);
  return  ret;
}
```

# Основной исходный файл pi.c

```
#include <stdio.h>
#include <time.h>

#define N 1000000000

double f( double x );

main ()
{
      double sum, pi, x, h;
      clock_t start, stop;
      int i;

      h = (double)1.0/(double)N;
      sum = 0.0;

      start = clock();
      for ( i=0; i<N ; i++ ){
            x = h*(i-0.5);
            sum = sum + f(x);
      }
      stop = clock();

      // print value of pi to be sure multiplication is correct
      pi = h*sum;
      printf("    pi is approximately : %f \n", pi);
      // print elapsed time
      printf("Elapsed time = %lf seconds\n",((double)(stop - start)) / CLOCKS_PER_SEC);
}
```

# Использование Intel C++ компилятора

- Базовые значения: Оптимизация отключена

```
> icl /Od pi.c fx.c /o Od_pi
...
> Od_pi.exe
   pi is approximately : 3.141593
Elapsed time = 12.691000 seconds
```

https://tinyurl.com/intel-cpp-compiled-od-option

# Высокоуровневые оптимизации (High-Level Optimizations)

- Включите уровень оптимизации O2 (исп. по умолчанию)

```
>icl /O2 pi.c fx.c /o O2_pi
...
> O2_pi.exe
   pi is approximately : 3.141593
Elapsed time = 1.654000 seconds
```

7.67x быстрее

- Посмотрите отчет об оптимизации:

```
>icl /O2 pi.c fx.c /o O2_pi /Qopt-report:3 /Qopt-report-file:O2.txt
```

# Межпроцедурная оптимизация (IPO)

- Добавьте /Qipo и посмотрите на результат

```
> icl /O2 /Qipo pi.c fx.c /o ipo_pi
...
> ipo_pi.exe
   pi is approximately : 3.141593
Elapsed time = 0.541000 seconds
```

3.06x быстрее

- Сгенерируйте отчёт об оптимизации:

```
> icl /O2 /Qipo pi.c fx.c /o ipo_pi /Qopt-report:3 /Qopt-report-file:ipo.txt
```

# Отключение нескольких оптимизаций

- Отключите векторизацию:

```
>icl /O2 /Qipo /Qvec- pi.c fx.c /o ipo_novec_pi /Qopt-report:3 /Qopt-report-file:ipo_novec.txt
...
> ipo_novec_pi.exe
   pi is approximately : 3.141593
Elapsed time = 1.174000 seconds
```

- Отключите возможность встраивания пользовательских функций:

```
> icl /O2 /Qipo /Ob0 pi.c fx.c /o ipo_noinline_pi /Qopt-report:3 /Qopt-report-file:ipo_noinline.txt
...
> ipo_noinline_pi.exe
   pi is approximately : 3.141593
Elapsed time = 1.640000 seconds
```

https://tinyurl.com/intel-cpp-compiler-qvec https://tinyurl.com/intel-cpp-compiler-ob-option

# Оптимизации, специфичные для процессоров Intel

- Добавьте /QxHost в компиляционную строку:

```
> icl /O2 /Qipo /QxHost pi.c fx.c /o xhost_pi /Qopt-report:5 /Qopt-report-file:xhost.txt
...
> xhost_pi.exe
   pi is approximately : 3.141593
Elapsed time = 0.481000 seconds
```

- Сравните отчёты для случаев с/без оптимизации под железо:

```
> fc ipo.txt xhost.txt
```

В чём заметная разница?

# Профилировка приложения компилятором

- Выполните инструментацию приложения

```
> icl /Qprof-gen pi.c fx.c /o pgen_pi
```

- Запустите приложение чтобы собрать статистические данные

```
> pgen_pi.exe
   pi is approximately : 3.141593
Elapsed time = 1.974000 seconds
```

- Используйте собранные данные и соберите приложение

```
> icl /O2 /Qipo /QxHost /Qprof-use pi.c fx.c /o puse_pi
...
> puse_pi.exe
   pi is approximately : 3.141593
Elapsed time = 0.470000 seconds
```

https://tinyurl.com/intel-cpp-compiler-qprof-gen https://tinyurl.com/intel-cpp-compiler-qprof-use

# Автоматическое распараллеливание

- Добавьте опцию для автоматического распараллеливания к HLO

```
> icl /O2 /Qparallel pi.c fx.c /o par_pi /Qopt-report:5 /Qopt-report-phase:par /Qopt-report-file:par.txt
...
> par_pi.exe
   pi is approximately : 3.141593
Elapsed time = 1.435000 seconds
```

- Добавьте опцию для автоматического распараллеливания к примеру с оптимизацией, специфичной для процессоров Intel

```
> icl /O2 /Qipo /QxHost /Qparallel pi.c fx.c /o par_xhost_pi /Qopt-report:5 /Qopt-report-phase:par
/Qopt-report-file:par_xhost.txt
...
> par_xhost_pi.exe
   pi is approximately : 3.141593
Elapsed time = 0.147000 seconds
```

- Сравните отчеты об оптимизации

# Добавление OpenMP

- Рассмотрите код pi_par.c
- Скомпилируйте его с добавлением опции /Qopenmp

```
> icl /O2 /Qipo /QxHost /Qopenmp pi_par.c fx.c /o openmp_pi /Qopt-report:5
/Qopt-report-phase:openmp /Qopt-report-file:omp.txt
...
> openmp_pi.exe
   pi is approximately : 3.141593
Elapsed time = 0.142147 seconds
```

89x быстрее, чем исходная версия!

# Тестовое задание 2: Векторизация

# Тестовое задание 2

- Перейдите в папку compiler-optimization-master\vectorization\matvector\c
- Изучите код

# Векторизация отключена

- Скомпилируйте с выключенной векторизацией:

```
> icl /O2 /QxHost /Qvec- multiply.c driver.c /o matvector
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 3.048349 seconds
GigaFlops = 2.645366
Sum of result = 77172000000.000000
```

# Векторизации быть?

- Скомпилируйте в включённой векторизацией:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 3.108680 seconds
GigaFlops = 2.594027
Sum of result = 77172000000.000000
```

- Соберите отчёт об оптимизации и изучите его:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qopt-report:5
```

- Нет ускорения. Почему?

# Единичные шаги в цикле

- Посмотрите на код в multiply.c:

  Чему равны inc_i и inc_j? Компилятор знает их значения во время компиляции?

- Перейдите в vectorization/matvector/c/solutions/unit-stride, скомпилируйте с включённой векторизацией и изучите отчёт об оптимизации:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qopt-report:5
```

- Запустите: :

```
> matvector.exe
```

- Теперь цикл векторизован?

# Векторизация внутреннего цикла [1]

- Можно ли векторизовать без изменения кода?
- Перейдите в vectorization/matvector/c/solutions/multiversioning_off и скомпилируйте с опцией /Qalias-args- :

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qalias-args- /Qopt-report:5
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.822794 seconds
GigaFlops = 9.800748
Sum of result = 77172000000.000000
```

- Есть ускорение? Цикл векторизовался?

https://tinyurl.com/intel-cpp-compiler-qalias-args

# Генерация ассемблера

- Но а что же происходит внутри? Какие инструкции генерирует компилятор?
  - Скомпилируйте multiply.c с опцией /S:

```
> icl /O2 /QxHost multiply.c /Qalias-args- /S
```

  - ИЛИ перейдите по ссылке: https://godbolt.org/z/qqznK66cv

# Векторизация внутреннего цикла [2]

- Можно ли передать информацию о конкретном цикле?
  - Да! Нужно добавить #pragma ivdep в нужное место
- Перейдите в vectorization/matvector/c/solutions/ivdep и скомпилируйте:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qopt-report:5
```

- Запустите:

```
>matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.821833 seconds
GigaFlops = 9.812217
Sum of result = 77172000000.000000
```

- Есть ускорение? Цикл векторизовался? https://godbolt.org/z/YWP7Ydbzj

# Векторизация внутреннего цикла [3]

- Можно ли передать информацию о конкретном указателе?
  - Да! Используйте ключевое слово restrict с этим указателем
- Перейдите в vectorization/matvector/c/solutions/restrict и скомпилируйте с опцией /Qrestrict :

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qrestrict /Qopt-report:5
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.810324 seconds
GigaFlops = 9.951574
Sum of result = 77172000000.000000
```

- Есть ускорение? Цикл векторизовался? https://godbolt.org/z/nsj8vnca8

# Выравнивание данных [1]

- Взгляните на driver.c и проверьте как выделена память на a, b и c
  - Можем ли мы гарантировать выравнивание этих массивов? Как?
- Перейдите в vectorization/matvector/c/solutions/align и скомпилируйте:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qopt-report:5
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.905035 seconds
GigaFlops = 8.910154
Sum of result = 77172000000.000000
```

- Нет ускорения. Почему? https://godbolt.org/z/oMb56xYPT

# Выравнивание данных [2]

- Взгляните на multiply.c
  - Указана ли какая-либо информация о выравнивании a, b, c? Нужно это исправить!
- Перейдите в vectorization/matvector/c/solutions/assume_aligned и скомпилируйте:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector /Qopt-report:5
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.753137 seconds
GigaFlops = 10.707220
Sum of result = 77172000000.000000
```

- Видим ускорение!

4x быстрее, чем исходная версия

https://godbolt.org/z/44f8dzfnW

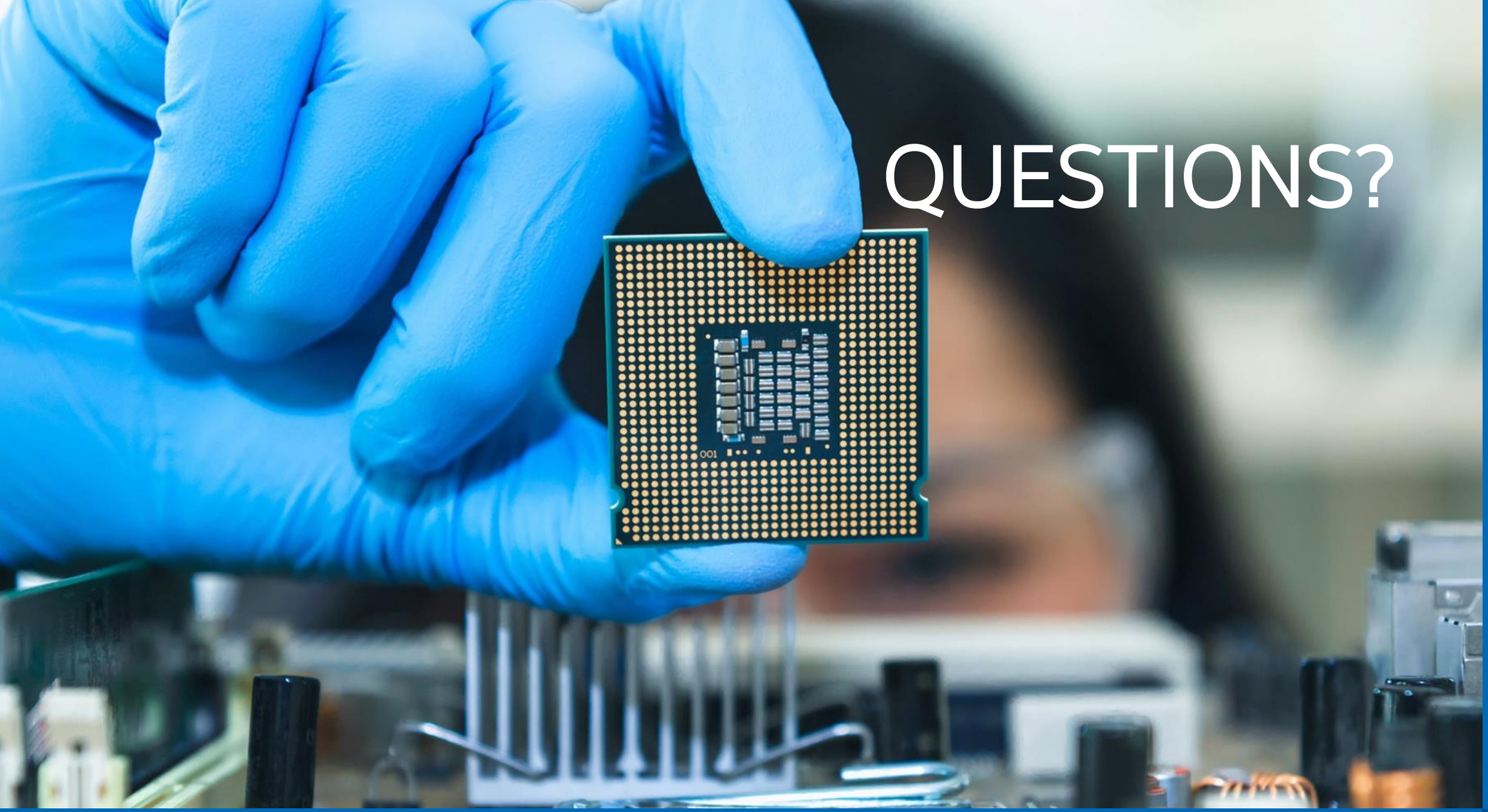
# Дополнительное ускорение

- Перейдите в vectorization/matvector/c/solutions/vector_aligned и посмотрите какие изменения сделаны в коде
- Скомпилируйте:

```
> icl /O2 /QxHost multiply.c driver.c /o matvector
```

- Запустите:

```
> matvector.exe
ROW: 64 COL: 63
Elapsed time = 0.420657 seconds
GigaFlops = 19.474289
Sum of result = 77172000000.000000
```


QUESTIONS?

# Notices & Disclaimers

- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- 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. For more complete information visit www.intel.com/benchmarks.
- 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.
- 

**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


intel