Intel® OpenCL Implicit Vectorization Module

Nadav Rotem Software Developer, Intel® November 2011

#### **Intel OpenCL Team**

- Responsible for Intel® OpenCL SDK for Intel® Architecture.
- Develop the LLVM-based OpenCL\* compiler, debugger, etc.
- Enable future architectures in LLVM.
- The group is centered in Haifa, Israel.



Software & Services Group Developer Products Division



#### **Intel® OpenCL SDK 1.5**

- Publicly Available Today
- Features:
  - Conformant with the OpenCL\* 1.1 specification.
  - Supports 32/64-bit Microsoft Windows\* operating systems and 64-bit Linux\* operating systems.
  - Unique Implicit Vectorization Module.
  - Available at: www.intel.com/software/opencl





Software & Services Group Developer Products Division



#### **Intel® OpenCL SDK Offline Compiler**

- The Intel® OpenCL SDK Offline Compiler exposes LLVM-IR and enables users to examine the vectorized OpenCL\* kernel.
- Tool <u>demo</u>

Developer Products Division

Software

OpenCL Code	InstDy OpenCL IDE Offices Complet (IL MC - EndEqual For Yess Inst 1949 CoperCL Code For Yess Inst 1949 CoperCL Code For Y - LY GY IS presser team my, use my as the count or steps man p - DDS // estatly the same steps but with different coordinates. DDS // Estatly the same steps but with different coordinates. DDS // Estatly the same steps but with different coordinates. DDS // Estatly the same steps but with different coordinates. DDS // Estatly the same steps but with different coordinates. DDS // Estatly the same steps but with different coordinates. DDS // Fix Decay by the step length, which is computed as the r Time Decay - exp(_decay - exp(_decay - exp( to onvert_float( dx ) * DIS convert_float( dx ) + convert_float( dx ) * DIS // Det up the Breenham delts effor. DIS // Det up the Breenham delts effor. DIS // If the vertical ray (dy > dx) crosses either a bottom or // Decaer, coop it by the step count. DIS // Detect the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DIS // Select the count of steps that should be desited. DI	declare void (Barrier(132) declare void (Barrier(132) declare 10* @get_special_buffer.() declare 132 @get_iter_count.() declare 132 @get_new_global_id.(132, 21) define void (EvaluateBag(<4 x 1 mt) = addropace(1 ; <label>:0 %1 = extracteLement &lt;2 x 132&gt; %imgSize, 132 0 %2 = mik 132 %1, 2 %3 = extracteLement &lt;2 x 132&gt; %imgSize, 132 1 %4 = edix 132 %1, 2 %3 = mail new 132 %1, 3 %4 = add new 132 %1, -1 %6 = add new 132 %1, -1 %6 = add new 132 %1, -1 %6 = add new 132 %3, -1</label>	LLVM-IR
e) Software & Services	Adding     Default instruction set architecture.     Intel Openct CFU device was found:     Device measure intel(R) Core(TH) 17-14000 CFU B 3.4008x     Device version: OpenCL 1.1 (David State)     Device version: OpenCL 1.1 (David State)     Device profile: FULL_RECFLIC     Device profile: FULL_RECFLIC     Device state(R)     Rersel (GodDays> was not vertorized     Device     Sild succesded:	<pre>%.not = icmp sqt 132 %1, -2 %15 = icmp sqt 132 %1, -2 %15 = icmp sqt 132 %4, %8 %sr.comd23 = set 11 %9, %10 %11 = sub new 132 %4, %8 %13 = select 11 %9, 132 %11, \$32 %12 %14 = shl 132 %13, 1 %13 = add 132 %14, -2 %16 = icmp sqt 132 %2, 41 %17 = icmp sqt 132 %2, 47 %sr.comd26 = sr 11 %17, %18 %19 = sub new 132 %, %2 %sr.comd26 = sr 13 %17, %18 %sr.comd56 = sr 13 %sr.comd56 = sr 33 %sr.comd56 = sr 35 %sr.comd56 %sr.com</pre>	

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners. Optimization

Notice





### **OpenCL\***

- A framework for developing data-parallel programs for multiple kinds of devices.
- An open standard by Khronos\*.
- C-based language for compute shaders (called kernels):
  - Derived from ISO C99
  - Few restrictions, e.g. recursion, function pointers
  - Short vector types e.g., float4, short2, int16
  - Built-in functions: math (e.g., sin), geometric, common (e.g., min, clamp)



Software & Services Group Developer Products Division



#### **OpenCL\* Kernels**

#### Serial

#### Data Parallel

```
kernel void
cl_mul(global const float *a,
    global const float *b,
    global float *result) {
```

```
int id = get_global_id(0);
```

result[id] = a[id] \* b[id];



}

Software & Services Group Developer Products Division

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners.

}



Vectorization

Iterations – the "work" that needs to be done.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

•The OpenCL runtime executes the kernel on each one of the work-items.

•A kernel processes a single work-tem.

kernel void cl mul(global const float \*a, global const float \*b, global float \*result) { int id = get global id(0); float k = sin(b[id]);

result[id] = a[id] \* k;



Software & Services Group Developer Products Division



Iterations – the "work" that needs to be done.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

	<pre>brief widd af mat(ichal const first *s,     gidaal const first *s,     gidaal const first *s,     gidaal flost *cesult) ( ist id = get global_id(0); flost k = sin(b[id]); result[id] = a[id] * k; )</pre>	hered weld al_meliphical const float *s, _pickal const float free _pickal float *result) { last id = get_global_id(0;; float k = sin(b(d()); result[id] = a[id] * k; }	hereal wold al mailphana const first *s, planal const first *b, planal set first *b, planal first *result) { ist id = get glabal_id(0); float k = sin(b[id]); result[id] = a(id] * k; }	<pre>hereal widd al_mil(pichai const float *a, pichai const float *a, pichai float *result) ( ist id = get_global_id(b); float k = sin(b(id)); result[id] = a(id] * k; )</pre>	<pre>hered wid al_mlightabl const float "s, global most float "s, global float "result) ( ist id = get_global_id(b); float k = sin(b(d)); result[id] = s[id] + k; ]</pre>	<pre>hereal widd al_mil(pichai const float *a, pichai const float *a, pichai float *result) ( ist id = get_global_id(b); float k = sin(b(id)); result[id] = a(id] * k; )</pre>	<pre>herrel wid al milphohai const float "a, global suss float "a, global float "result) ( ist id = get_global_id(b); float k = sin(b(id)); result[id] = a(id] * k; ]</pre>	<pre>herrel wid al_mdightah omst float *s, global sumst float *s, global float *creatly { ist id = get_global_id(0;) float k = sin(b[d]); result[id] = a[id] * k; }</pre>
--	---	---	---	--	---	--	---	---

- •Vectorized kernels process multiple work-items at once.
- •Less kernel invocations are needed.

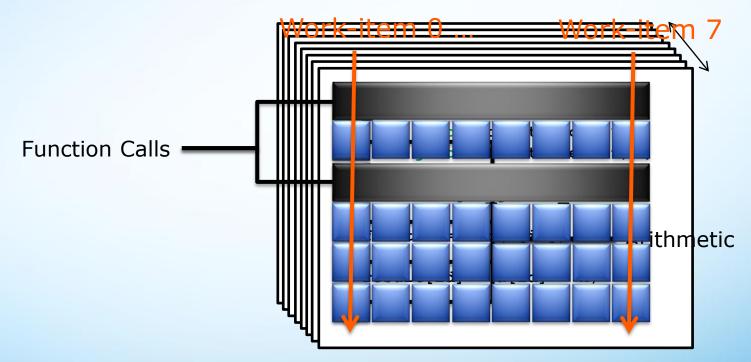


Software & Services Group Developer Products Division



#### What is Data-Parallel Vectorization ?

- 1. Input: LLVM-IR
- 2. Widen each instruction to make use of SSE4/AVX
- 3. Reduce the number of kernel invocations





Software & Services Group Developer Products Division



#### **Vectorization in LLVM-IR**



becomes

%b = add <8 x i32> %a, %a

Vectorization factor

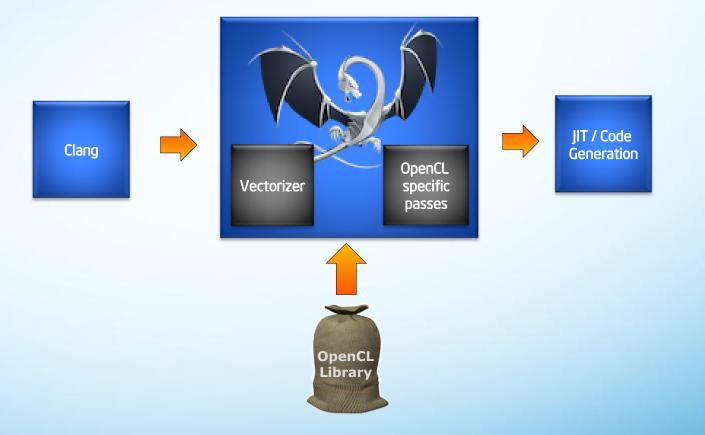


Software & Services Group Developer Products Division



#### **The OpenCL\* Implicit Vectorization Module**

#### **Intel® OpenCL SDK Structure**



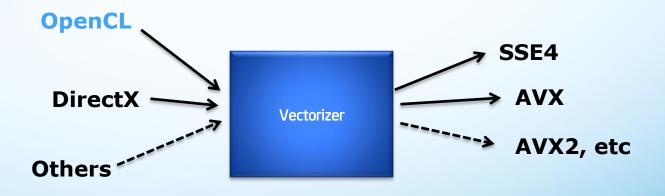


Software & Services Group Developer Products Division



#### Vectorizer

- Used in Intel® OpenCL SDK as well as other dataparallel products.
- Generates code for multiple Intel devices.
- A sequence of LLVM transformation and analysis passes.





Software & Services Group Developer Products Division



#### **Portable Vectorizer**

- The vectorizer has multiple data-parallel "users".
- Language dependent information is provided by an external analysis pass.
- For example:
  - In OpenCL\* get\_global\_id() returns a consecutive index.
  - In Microsoft DirectX\* API, 'sync' is a synchronization function.
  - List of built-in functions.
- LLVM allows high flexibility and enables vectorization that is language and platform independent.



Software & Services Group Developer Products Division



# **Packetization**



- Widens a single element into a vector of elements.
- Many details are handled by the packetizer:
  - Widening of arithmetic scalar LLVM instructions to vector LLVM instructions.
  - Replacing function calls with wide versions.
    - For example: sin(float) -> sin4(float4)
  - Optimized widening of load/stores.



Software & Services Group Developer Products Division



#### **Packetization of 'Vector' Code**

- OpenCL\* supports vector data types and some users write vector code.
- We scalarize all incoming user vectors and revectorize to make use of the wide instruction set.

Work-item 0 ...

X

Work-item 7

X

• AOS -> SOA

V V V V V V X Ζ Ζ Ζ Ζ Ζ Ζ W W W W W W

X

X

X

X

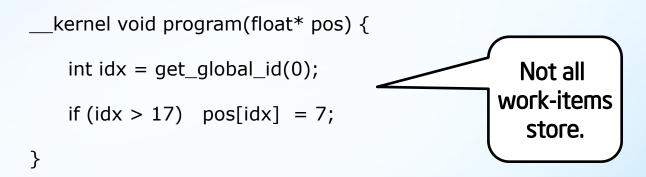


Software & Services Group Developer Products Division



#### Predication

When control-flow diverges, it is illegal to widen instructions.



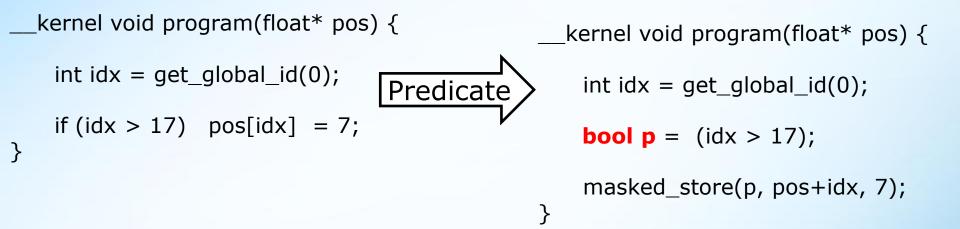
- Need to use masks (predicates) to make all of the work-items execute the same control flow path.
- Need to handle diverging branches, loops, etc.



Software & Services Group Developer Products Division



#### **Vectorization Flow**



\_\_kernel void program(float\* pos) {

int4 idx = get\_global\_id4(0);

**bool4 p** = (idx > 17);

masked\_store4(p, pos+idx, 7);



Software & Services Group Developer Products Division

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners.

Packetize



# **Control Flow in 'while' Loops**

- In loops, work-items enter at the same time, but exit at different times.
- Work items need to wait for the last work item to finish executing.
- Mask starts with all-one, and drops to zero for each work-item leaving the loop.



Software & Services Group Developer Products Division

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners.



Loop mask

## **Masking of Functions and Memory Ops**

- We mask: function calls, loads, stores, etc.
- LLVM does not support IR-level predication.
- Use Intrinsic functions:
  - Each architecture requires a different set of intrinsics.
  - Dedicated code to handle the widening of masked functions.
  - Dedicated code to implement unsupported memory ops, scatter/gather, etc.



Software & Services Group Developer Products Division

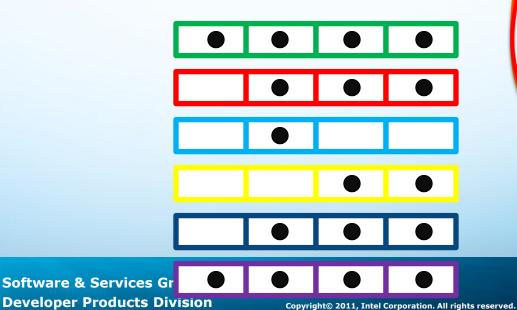


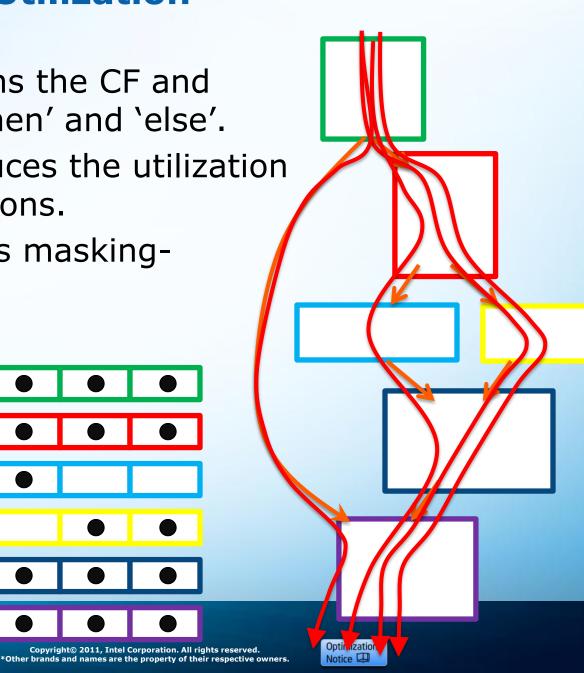
# **Partial Vector Utilization**

- Predication flattens the CF and execs both the 'then' and 'else'.
- Diverging CF reduces the utilization of vector instructions.
- Vectorization adds maskingoverhead.

inte

Software





### **Vectorization Width Decision**

- A heuristics pass is used to decide what is the ideal vectorization width (1, 4, 8, etc.)
- Perform static-analysis to collect different parameters of the program.
- We decide what vectorization-factor is best for the program:
  - Programs with low vector-utilizations do not get vectorized.
  - Integer-heavy kernels usually run better on vectorization factor of 4 because AVX is focused on floating-point types.
- Tuned using a large number of OpenCL\* programs.

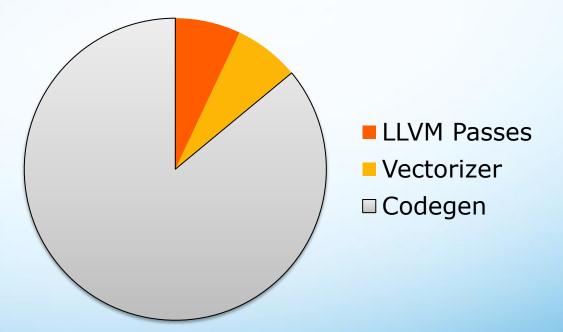


Software & Services Group Developer Products Division



# **Typical Build Time**

- The vectorizer is a fast pass.
- The vectorizer increases the codegen build time.





Software & Services Group Developer Products Division



#### **Working with the LLVM community**

#### **Vector-Select Codegen support**

 Our vectorizer uses the vector-select instruction extensively.

```
%C = fcmp oeq <8 x float> %A, %B
```

```
%V = select <8 x i1> %C, <8 x float> %A, <8 x float> %B
```

- Initially, we implemented a pass to lower vectorcompare and vector-select sequences to x86 (AVX and SSE) intrinsics.
- Worked with the community on implementing codegen support for vector-select.

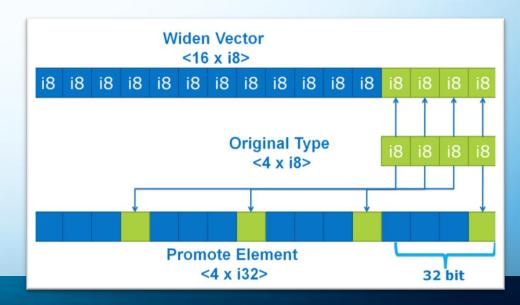


Software & Services Group Developer Products Division



#### **Vector Select and Type-Legalization**

- Added a new kind of type-legalization: "vector-promotion"
- Added support for vector-select on top:
  - Added x86 optimized blend sequences for AVX, SSE4
  - Implemented using XOR, AND, for other targets.
- Enabled in ToT.





Software & Services Group Developer Products Division



# **LLVM Wishlist**

- Predicated instructions:
  - Needed for vector architectures, which will likely to dominate future processors
  - Need IR changes
- Improved Microsoft Windows\* operating systems support:
  - Calling Conventions, MC, etc.
- Backward compatibility of the IR:
  - Use LLVM-IR as a cross-platform IR for OpenCL
  - Ease migration between versions of LLVM



Software & Services Group Developer Products Division



#### **Intel OpenCL Team and the Community**

- We've contributed bug fixes and features to LLVM.
- We've developed our own AVX support on top of LLVM 2.8.
  - Started contributing our improvements.
- We plan to work on AVX2 together with the community.
- We intend to work with the LLVM community on improving LLVM for Intel Platforms.



Software & Services Group Developer Products Division



#### **Questions ?**



# Software



Software & Services Group Developer Products Division

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners.

**Intel Confidential** 



32

#### **Optimization Notice**

#### **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



Software & Services Group Developer Products Division

Copyright© 2011, Intel Corporation. All rights reserved. \*Other brands and names are the property of their respective owners.

#### Intel Confidential



#### **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.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Cilk, Core Inside, FlashFile, i960, InstantIP, Intel, the Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside 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 © 2011. Intel Corporation.

#### http://intel.com/software/products



Software & Services Group Developer Products Division



