ISPC: clang-based front-end

2014 LLVM Developers' Meeting, Oct 28-29

Dmitry Babokin
James Brodman

ISPC Language

ISPC is a C-based language built on the SPMD (single program, multiple data) programming model. The ISPC programming model efficiently maps to the SIMD vector units of modern CPUs without the need for complex analysis and autovectorization. The key concept for achieving this is varying data types, which in cases of built-in types are simply equivalent to vectors of built-in types. For user-defined data types, ISPC generates Structure-of-Arrays style data layout that better maps to SIMD hardware. Varying data types are also allowed in control-flow statements (such as if statements), triggering "varying control flow".

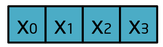
ISPC Language Concepts

Varying types

- Varying types get a separate copy for each SIMD lane, while uniform types are shared by all instances.

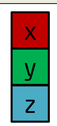
```
uniform float x;
```


```
varying float x;
```



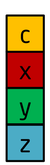
- Varying structures propagate variability to their members, producing Structure-Of-Arrays style types, which are optimal for vector code.

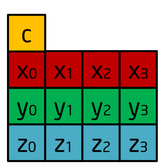
```
struct Point { float x, y, z; };
uniform Point uP;          varying Point vP;
```



- Structures may have members with bound variability.

```
struct Point2 {
  uniform int c;        // bound to uniform
  float x, y, z;        // unbound variability
};
uniform Point2 uP2;     varying Point2 vP2;
```





Varying control flow

- SIMD hardware has to do the same thing in each lane, by definition.

- Conditional statements based on varying values may have different results in each lane. When this occurs, control flow is divergent because different lanes might take different paths.

- ISPC transforms divergent control flow into data flow. An execution mask is used to only commit results in lanes that are actively participating in computation.

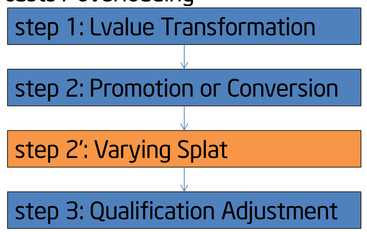
```
varying int i, j;
...
if (i > 0) {
  j++;
}
else {
  j--;
}
```

```
oldMask = Mask;
test = (i > 0);

Mask = oldMask & test;
if (Any(Mask)) {
  j = select(Mask, j+1, j);
}

Mask = oldMask & ~test;
if (Any(Mask)) {
  j = select(Mask, j-1, j);
}

Mask = oldMask;
```

- Masking is also used in foreach statements for any remainder iterations when the total number of iterations is not even divisible by the vector width.

Implementation of ISPC concepts in clang

- ISPC language definition is derived from "CPlusPlus".
- Implicit casts / overloading



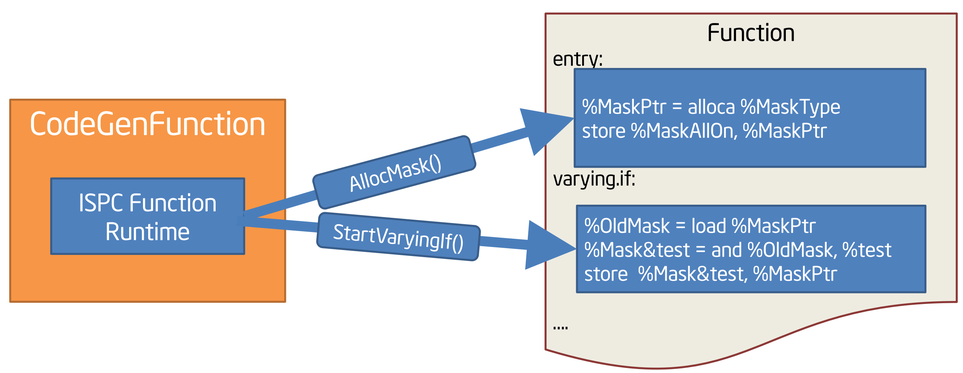
Category	Rank
Lvalue Transformation Quality Adjustment	Exact Match
Varying Splat	Varying Splat
Promotion	Promotion
Conversion	Conversion

- Type system tweaks:
  - Variability is a new property of Type class, but not a new type in the Type hierarchy.
  - Variability may be varying/uniform/unbound.
  - Variability is "enabled" only for selected types, when it makes sense (BuiltinType, PointerType, etc)
- All structs/classes are parsed as templates with implicit variability arguments

```
struct S {
  int int_field;          // unbound
};
varying struct S varying_s;
```

```
TranslationUnitDecl
|-ClassTemplateDecl S
| |-VariabilityTemplateParmDecl variability
| |-CXXRecordDecl struct S definition
| | |-CXXRecordDecl struct S
| | `-FieldDecl int_field 'unbound int'
| `-ClassTemplateSpecializationDecl struct S definition
|   |-TemplateArgument expr
|   | `-CXXBoolLiteralExpr '_Bool' false
|   |-CXXRecordDecl prev struct S
|   |-FieldDecl int_field 'varying int'
|   |-CXXConstructorDecl S 'void (void)'
|   | `-CompoundStmt
|   `-CXXConstructorDecl S 'void (const struct S<varying> &)'
|     `-ParmVarDecl 'const struct S<varying> &'
`-VarDecl varying_s 'struct S<varying>':'struct S<varying>'
  `-CXXConstructExpr 'struct S<varying>':'struct S<varying>' 'void (void)'
```

- ISPC Code Generation



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