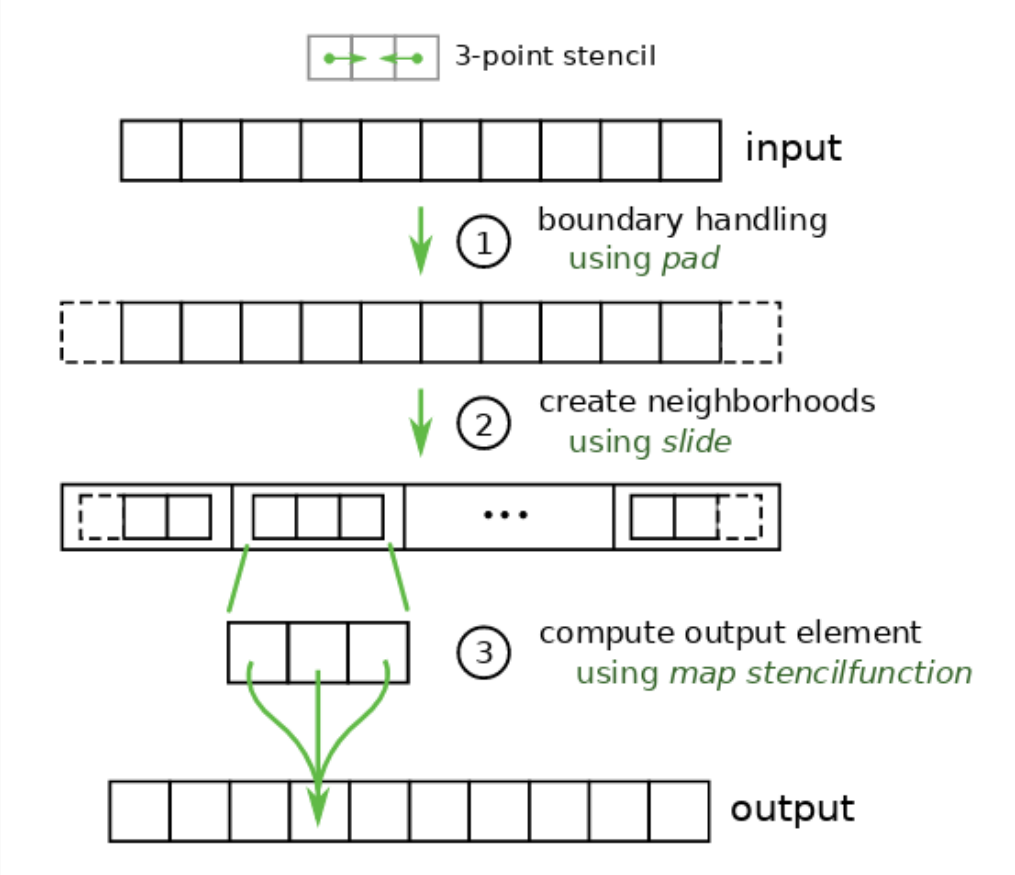
Lift Tutorial: View System

Bastian Hagedorn

Views

A Simple Expression

```
val highLevel = fun(
  ArrayType(Float, N), input ⇒
    Map(Reduce(add, 0.0f)) o
      Slide(3,1) o
        Pad(1,1,clamp) $ input )
```



Data Layout Primitives

```
val highLevel = fun(
  ArrayType(Float, N), input ⇒
    Map(Reduce(add, 0.0f)) o
      Slide(3,1) o
        Pad(1,1,clamp) $ input )
```

Observations:

- *Pad* and *Slide* only modify the data layout
  - → How to avoid unnecessary temporary result?
- *Slide* increases the dimension of our one-dimensional input array
  - → How to generate accesses to multi-dimensional arrays with a flat representation in memory?

Data Layout Primitives

```
val lowLevel = fun(
  ArrayType(Float, N), input ⇒
    MapGlb(MapSeq(toGlobal(id)) o ReduceSeq(add, 0.0f)) o
      Slide(3,1) o
        Pad(1,1,clamp) $ input )
```

Observations:

- *Pad* and *Slide* only modify the data layout
  - → How to avoid unnecessary temporary result?
- *Slide* increases the dimension of our one-dimensional input array
  - → How to generate accesses to multi-dimensional arrays with a flat representation in memory?

3-Point Stencil Code

```
float add(float x, float y) { return x + y; }
float id(float x) { return x; }
kernel void KERNEL(const global float *restrict IN, global float *OUT, int N) {
  float acc;
    // Map
    for (int globalID = get_global_id(0); (globalID < N);
         globalID = (globalID + get_global_size(0))) {
        acc = 0.0f;
        // Reduce
        for (int i = 0; i < 3; i++) {

            acc = add(acc, IN[???]);
        }
        OUT[globalID] = id(acc);
    }
}
```

Introducing Views

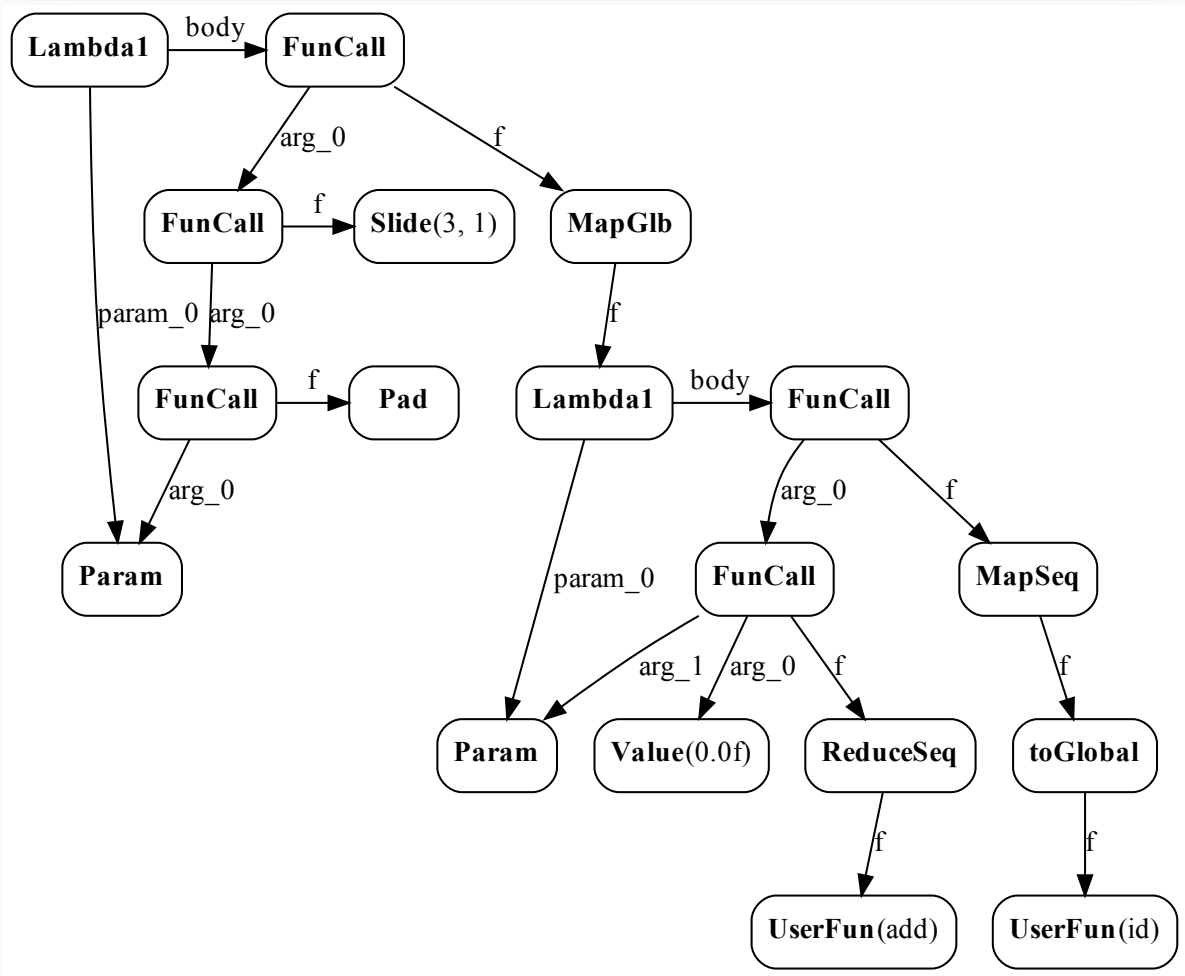
```
val lowLevel = fun(
  ArrayType(Float, N), input ⇒
    MapGlb(MapSeq(toGlobal(id)) o ReduceSeq(add, 0.0f)) o
      Slide(3,1) o
        Pad(1,1,clamp) $ input )
```

Views:

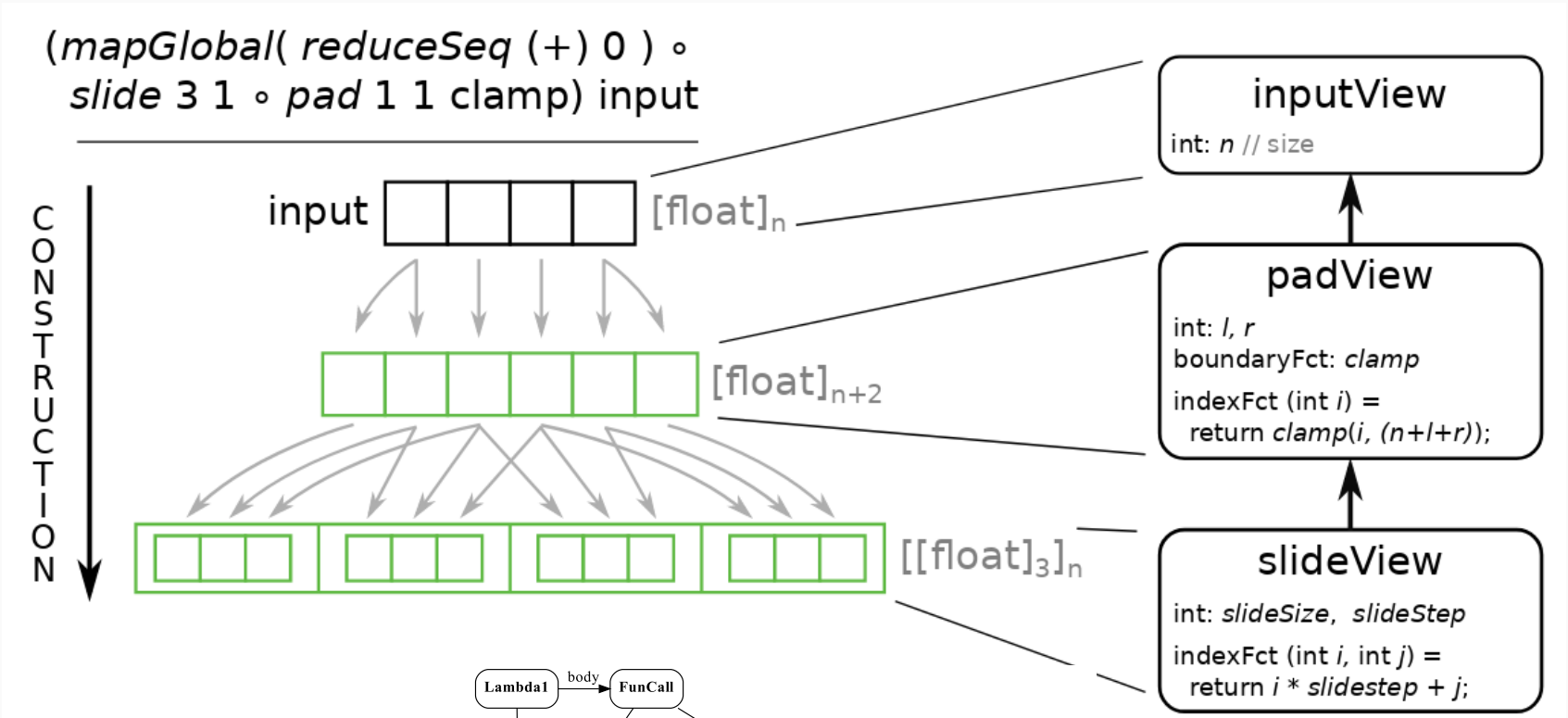
- **Construct** a representation of the effects of data layout functions
- **Consume** the views to generate correct array indices

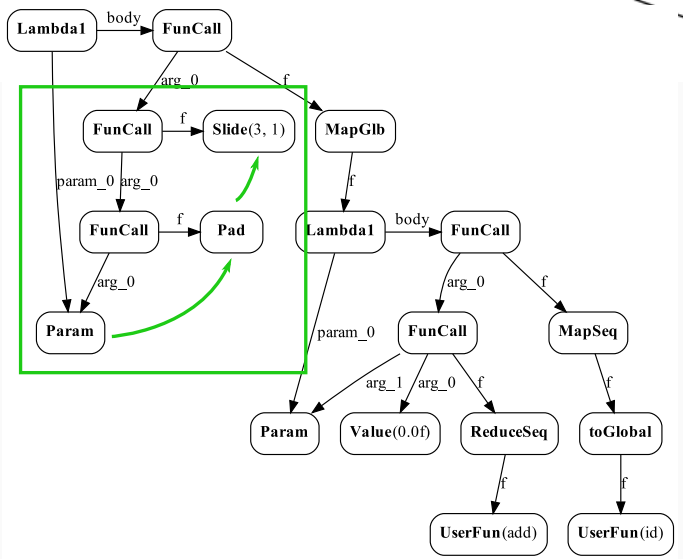
AST

```
val lowLevel = fun(
  ArrayType(Float, N), input ⇒
    MapGlb(MapSeq(toGlobal(id)) o ReduceSeq(add, 0.0f)) o
      Slide(3,1) o
        Pad(1,1,clamp) $ input )
```

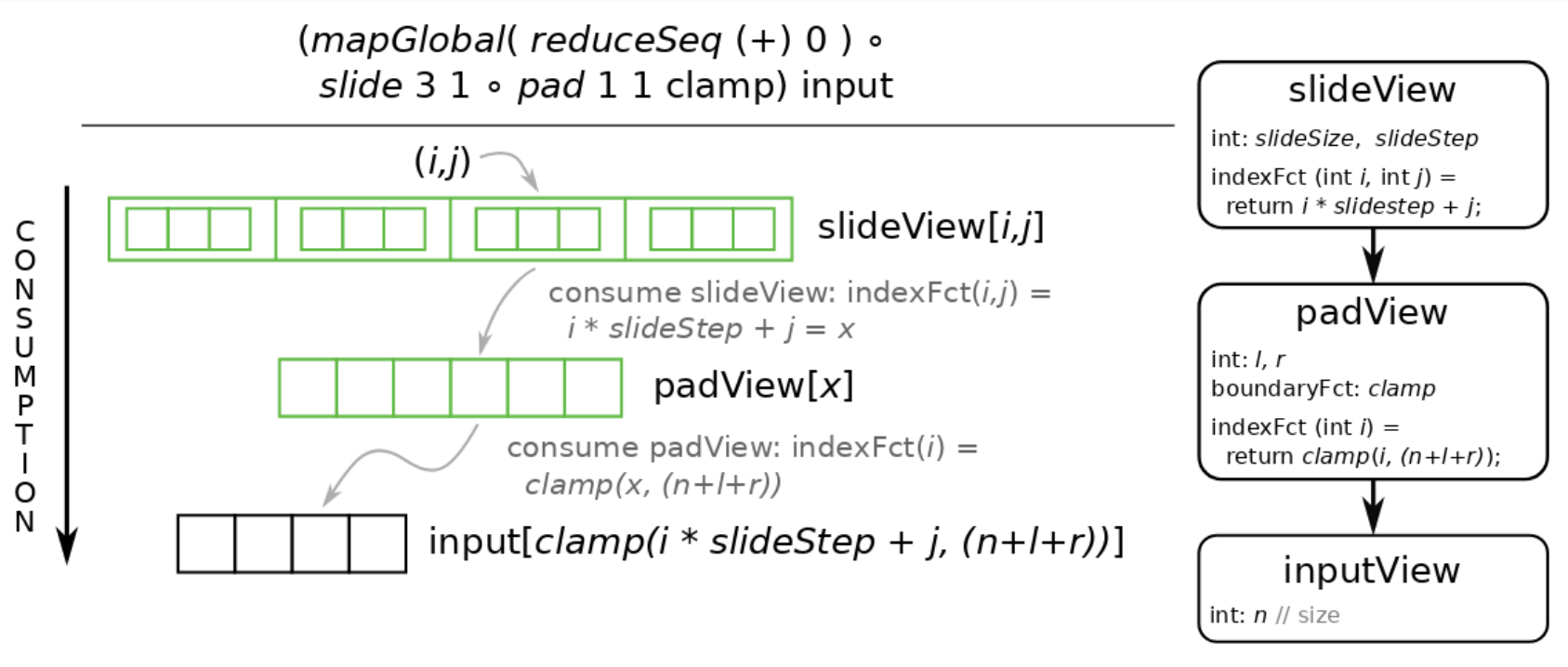


View Construction: Follow the Dataflow

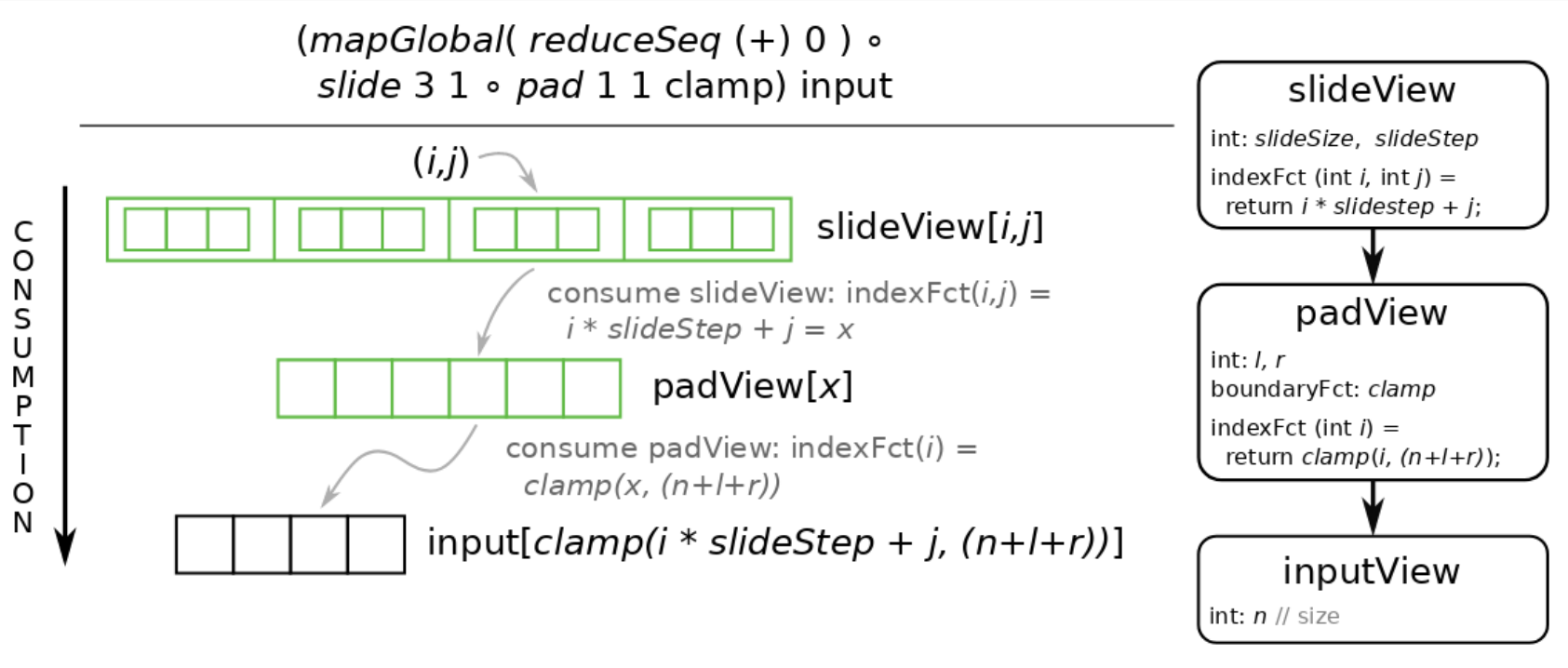




View Consumption: Generate Flat Accesses

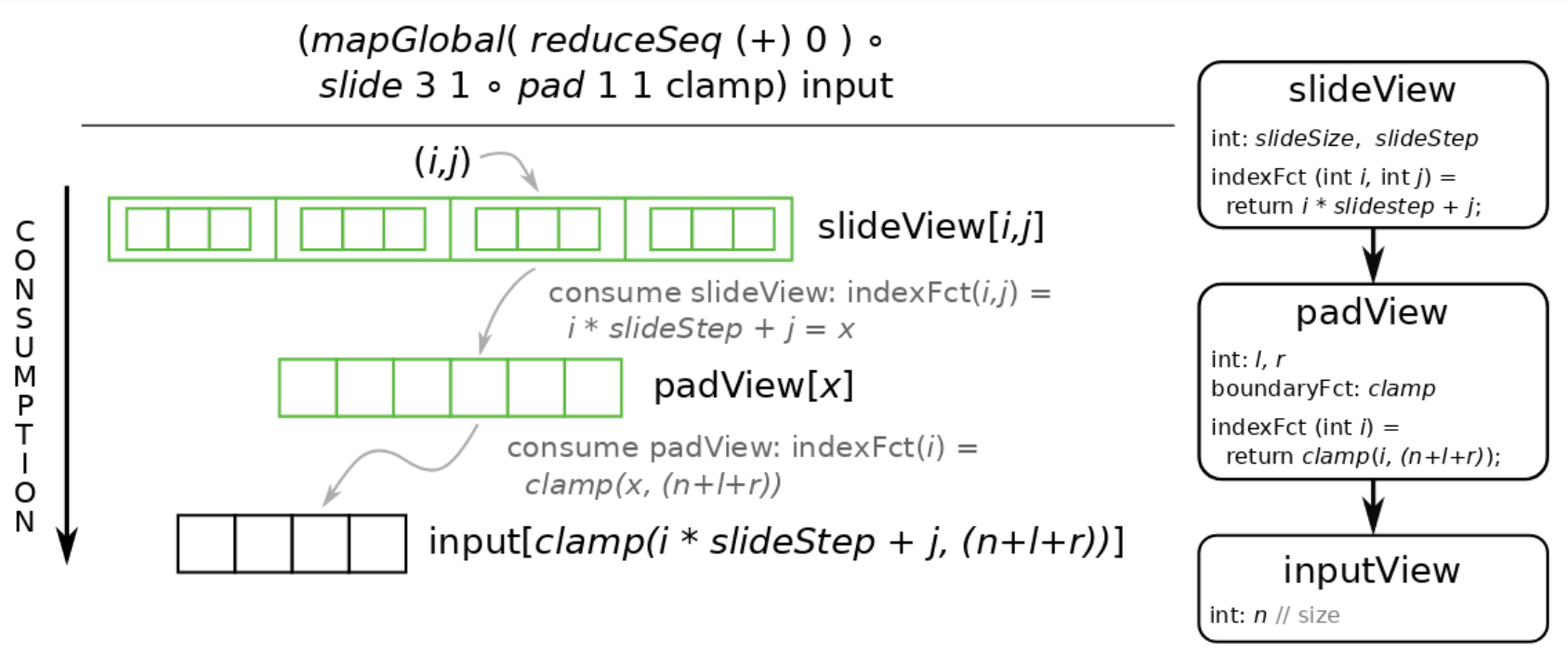


View Consumption: Generate Flat Accesses



Demo: `emitView` for `ViewSlide`

View Consumption: Generate Flat Accesses



Demo: `emitView` for `ViewSlide`

```
for(int i = 0; i < 3; i++) {
    acc = add(acc, IN[(((-1 + globalID + i) ≥ 0) ?
                       (((-1 + globalID + i) < N) ?
                         (-1 + globalID + i) : (-1 + N)) : 0)])
}
```

View Consumption: Generate Flat Accesses

```
// Reduce unrolled
acc = add(acc, IN[(((-1 + globalID + 0) ⩾ 0) ? (((-1 + globalID + 0) < N) ? (-1 + globalID + 0) : (-1 + N)) : 0)]);
acc = add(acc, IN[(((-1 + globalID + 1) ⩾ 0) ? (((-1 + globalID + 1) < N) ? (-1 + globalID + 1) : (-1 + N)) : 0)]);
acc = add(acc, IN[(((-1 + globalID + 2) ⩾ 0) ? (((-1 + globalID + 2) < N) ? (-1 + globalID + 2) : (-1 + N)) : 0)]);
```

Are all these operations necessary or can we do better?

Arithmetic Expression Simplification

`ArithExpr` Library

Lift comes with a powerful `ArithExpr` Library.

- performs simple arithmetic simplifications ($1 + 1 = 2$)
- keeps track of range information for variables
  - e.g., `0 <= globalID < N`
- handles arithmetic operations including integer division and modulo
  - e.g., $((2M + 1) \mod M) = 1 \mod M$

`ArithExpr` Library

Lift comes with a powerful `ArithExpr` Library.

- performs simple arithmetic simplifications ($1 + 1 = 2$)
- keeps track of range information for variables
  - e.g., `0 <= globalID < N`
- handles arithmetic operations including integer division and modulo
  - e.g., $((2M + 1) \mod M) = 1 \mod M$

Demo:

1. Library
2. `ArithExpr` Type Hierarchy
3. `Cst`, `Var` (including ranges), `?`, `Mod`
4. `SimplifySum`
5. Examples

Generated Indices Revisited

Question: Can we simplify the second array access?

```
// Reduce unrolled
acc = add(acc, IN[(((-1 + globalID + 0) ⩾ 0) ? (((-1 + globalID + 0) < N) ? (-1 + globalID + 0) : (-1 + N)) : 0)]);
acc = add(acc, IN[(((-1 + globalID + 1) ⩾ 0) ? (((-1 + globalID + 1) < N) ? (-1 + globalID + 1) : (-1 + N)) : 0)]);
acc = add(acc, IN[(((-1 + globalID + 2) ⩾ 0) ? (((-1 + globalID + 2) < N) ? (-1 + globalID + 2) : (-1 + N)) : 0)]);
```

Arithmetic Simplification

```
IN[ // predicate
    (((-1 + globalID + 1) >= 0) ?

    // true
    (((-1 + globalID + 1) <  N) ?
    (-1 + globalID + 1) : (-1 + N)) :

    // false
     0)
  ];
```

Arithmetic Simplification

```
IN[ // predicate
    (((-1 + globalID + 1) >= 0) ?

    // true
    (((-1 + globalID + 1) <  N) ?
    (-1 + globalID + 1) : (-1 + N)) :

    // false
     0)
  ];
```

Additions with constants cancel out

Arithmetic Simplification

```
IN[ // predicate
    ((globalID >= 0) ?

    // true
    ((globalID <  N) ?
    globalID : (-1 + N)) :

    // false
     0)
  ];
```

Arithmetic Simplification

```
IN[ // predicate
    ((globalID >= 0) ?

    // true
    ((globalID <  N) ?
    globalID : (-1 + N)) :

    // false
     0)
  ];
```

Predicate is always true (requires range information about the variable)

Arithmetic Simplification

```
//     predicate          true          false
IN[(globalID <  N) ? globalID : (-1 + N)];
```

Arithmetic Simplification

```
//for (int globalID = get_global_id(0);
//      globalID < N; ...

//     predicate          true         false
IN[(globalID <  N) ? globalID : (-1 + N)];
```

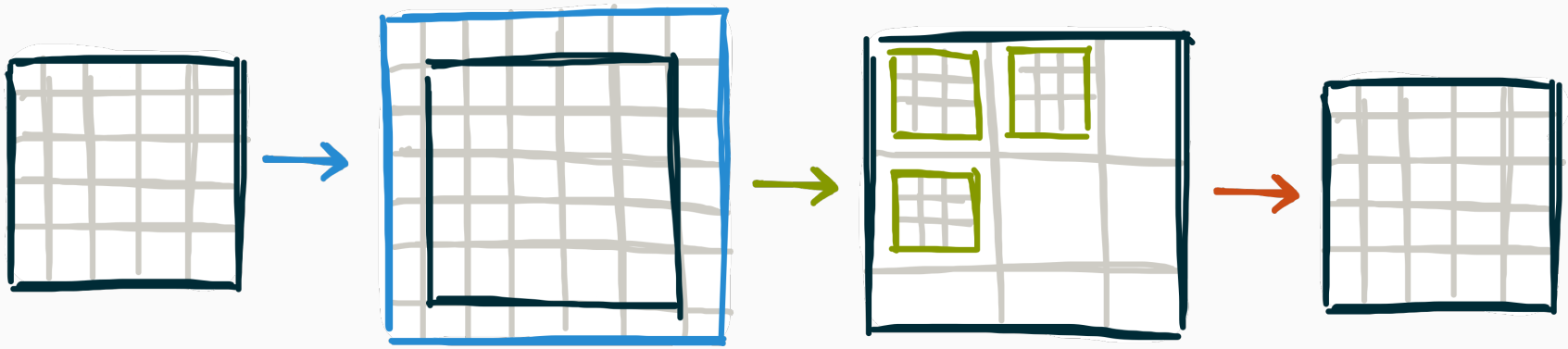
Arithmetic Simplification

```
IN[globalID];
```

Conclusion

```
val lowLevelExpression = fun(
    ArrayType(ArrayType(Float, M), N), input ⇒

    MapGlb(1)(MapGlb(0)(
        MapSeq(toGlobal(id)) o
            ReduceSeqUnroll(add, 0.0f) o Join()
    )) o Slide2D(3,1) o
      Pad2D(1,1,clamp) $ input)
```



Conclusion

```
float add(float x, float y){{{ return x+y;};}}
float id(float x){{{ return x;};}}
kernel void KERNEL(const global float* restrict v__20, global float* v__25, int v_M_0, int v_N_1){
        float v__21;
        for (int v_gl_id_16 = get_global_id(1); (v_gl_id_16 < (((1 + (1 + v_N_1)) + (-1 * (3 + (-1 * 1)))) / 1)); v_gl_id_16 = (v_gl_id_16 + get_global_size(1))){
            for (int v_gl_id_17 = get_global_id(0); (v_gl_id_17 < (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)); v_gl_id_17 = (v_gl_id_17 + get_global_size(0))){
                float v_tmp_38 = 0.0f;
                v__21 = v_tmp_38;
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((0 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((0 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((0 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((0 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((0 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((0 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((0 % 3) + (1 * (((((0 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((0 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((0 % 3) + (1 * (((((0 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((0 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((1 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((1 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((1 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((1 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((1 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((1 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((1 % 3) + (1 * (((((1 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((1 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((1 % 3) + (1 * (((((1 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((1 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((2 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((2 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((2 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((2 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((2 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((2 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((2 % 3) + (1 * (((((2 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((2 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((2 % 3) + (1 * (((((2 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((2 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((3 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((3 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((3 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((3 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((3 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((3 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((3 % 3) + (1 * (((((3 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((3 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((3 % 3) + (1 * (((((3 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((3 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((4 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((4 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((4 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((4 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((4 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((4 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((4 % 3) + (1 * (((((4 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((4 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((4 % 3) + (1 * (((((4 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((4 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((5 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((5 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((5 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((5 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((5 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((5 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((5 % 3) + (1 * (((((5 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((5 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((5 % 3) + (1 * (((((5 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((5 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((6 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((6 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((6 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((6 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((6 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((6 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((6 % 3) + (1 * (((((6 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((6 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((6 % 3) + (1 * (((((6 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((6 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((7 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((7 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((7 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((7 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((7 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((7 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((7 % 3) + (1 * (((((7 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((7 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((7 % 3) + (1 * (((((7 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((7 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__21 = add(v__21, v__20[((0 + ((0 + 0) + ((0 + (1 * v_M_0)) * ( (((((((8 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((8 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) ⩾ 0) ? ( (((((((8 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((8 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) < v_N_1) ? (((((((8 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((8 / 3) + (3 * v_gl_id_17)) % 3))) / (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)) + (1 * v_gl_id_16)) + (-1 * 1)) : (v_N_1 + (-1 * 1)) ) : 0 )))) + (1 * ( ((((8 % 3) + (1 * (((((8 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((8 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) < v_M_0) ? (((8 % 3) + (1 * (((((8 / 3) + (3 * v_gl_id_17)) / 3) + ((((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1) * (((8 / 3) + (3 * v_gl_id_17)) % 3))) % (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))) + (-1 * 1)) : (v_M_0 + (-1 * 1)) ) : 0 ))))];
                v__25[((0 + ((0 + ((0 + 0) + (v_gl_id_16 * (0 + ((0 + (1 * 1)) * (((1 + (1 + v_M_0)) + (-1 * (3 + (-1 * 1)))) / 1)))))) + (v_gl_id_17 * (0 + (1 * 1))))) + (1 * v_i_19))] = id(v__21);
            }
        }
}
```

Conclusion

```
kernel void KERNEL(const global float* restrict IN, global float* OUT, int M, int N){
    float acc;
    for (int y = get_global_id(1); (y < N); y = (y + get_global_size(1))){
        for (int x = get_global_id(0); (x < M); x = (x + get_global_size(0))){
            acc = 0.0f;
            // NW
            acc += IN[((M * ( ((-1 + y) ⩾ 0) ? (-1 + y) : 0 )) + ( ((-1 + x) ⩾ 0) ? (-1 + x) : 0 ))];
            // N
            acc += IN[(x + (M * ( ((-1 + y) ⩾ 0) ? (-1 + y) : 0 )))];
            // NE
            acc += IN[((M * ( ((-1 + y) ⩾ 0) ? (-1 + y) : 0 )) + ( ((1 + x) < M) ? (1 + x) : (-1 + M) ))];
            // W
            acc += IN[((M * y) + ( ((-1 + x) ⩾ 0) ? (-1 + x) : 0 ))];
            // C
            acc += IN[(x + (M * y))];
            // E
            acc += IN[((M * y) + ( ((1 + x) < M) ? (1 + x) : (-1 + M) ))];
            // SW
            acc += IN[((M * ( ((1 + y) < N) ? (1 + y) : (-1 + N) )) + ( ((-1 + x) ⩾ 0) ? (-1 + x) : 0 ))];
            // S
            acc += IN[(x + (M * ( ((1 + y) < N) ? (1 + y) : (-1 + N) )))];
            // SE
            acc += IN[((M * ( ((1 + y) < N) ? (1 + y) : (-1 + N) )) + ( ((1 + x) < M) ? (1 + x) : (-1 + M) ))];
          // write back result
            OUT[(x + (M * y))] = acc;
        }
    }
}
```