Improving Vectorization of Select Reduction

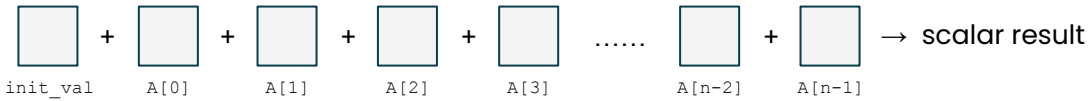
Mel Chen, Kolya Panchenko
SiFive Compiler Team
October 11th, 2023

What is Select Reduction?

Categorizing Reduction Patterns

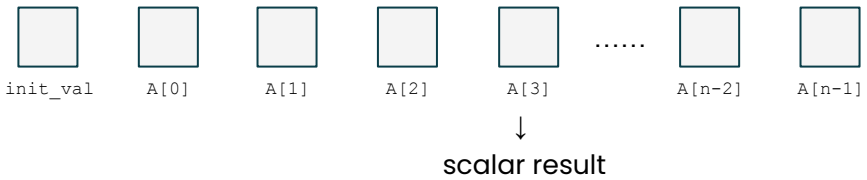
- Combine reduction (Arithmetic reduction)
  - The scalar result is obtained by combining the values

```
int sum = init_val;
for (int i = 0; i < n; i++)
  sum += A[i];
```



- Select reduction
  - The scalar result is directly chosen from the set of values

```
int max = init_val;
for (int i = 0; i < n; i++)
  if (A[i] > max)
    max = A[i];
```



1 Quoted from a comment by Ayal Zaks.

Outline

Preface What is Select Reduction?

A. Select Reduction Beyond Min/Max Reduction
   1. Current State: AnyOf Reduction Pattern (Author: David Sherwood)
   2. The New Thing: FindLastIV Reduction Pattern
   3. Our Goal: Min/Max with Index Pattern

B. Opportunities Left on the Table

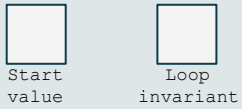
Select Reduction Beyond Min/Max Reduction

Current State: IAnyOf (SelectICmp), FAnyOf (SelectFCmp)1

- Select the integer loop invariant if the condition is matched for any of the iterations, and the start value otherwise.

```
int IAnyOf(int *a, int *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}

int FAnyOf(float *a, float *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}
```

AnyOf



1 David Sherwood, [LoopVectorize] Permit vectorisation of more select(cmp(), X, Y) reduction patterns

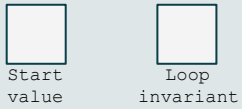
Select Reduction Beyond Min/Max Reduction

Current State: IAnyOf (SelectICmp), FAnyOf (SelectFCmp)1

- Select the integer loop invariant if the condition is matched for any of the iterations, and the start value otherwise.

```
int IAnyOf(int *a, int *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}

int FAnyOf(float *a, float *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}
```

AnyOf



1 David Sherwood, [LoopVectorize] Permit vectorisation of more select(cmp(), X, Y) reduction patterns

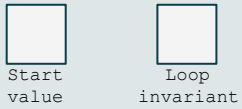
Select Reduction Beyond Min/Max Reduction

Current State: IAnyOf (SelectICmp), FAnyOf (SelectFCmp)1

- Select the integer loop invariant if the condition is matched for any of the iterations, and the start value otherwise.

```
int IAnyOf(int *a, int *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}

int FAnyOf(float *a, float *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ? 3 : rdx;
  return rdx;
}
```

AnyOf

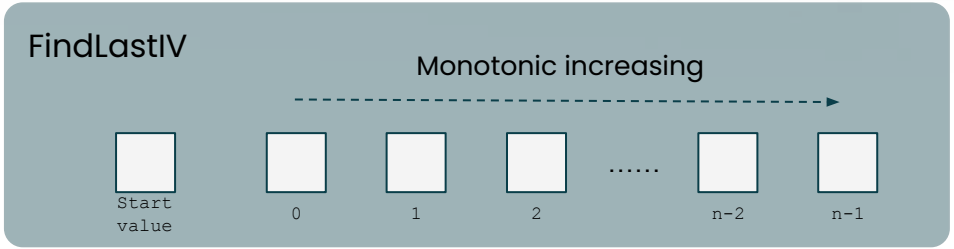


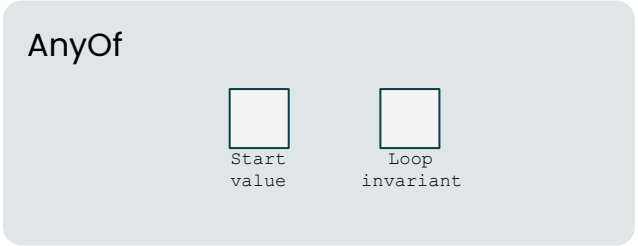
1 David Sherwood, [LoopVectorize] Permit vectorisation of more select(cmp(), X, Y) reduction patterns

Select Reduction Beyond Min/Max Reduction

The New Thing: IFindLastIV, FFindLastIV

- If the condition is never matched in any of iteration, select the start value.
- Select the maximum value of increasing induction variable that matches the condition.



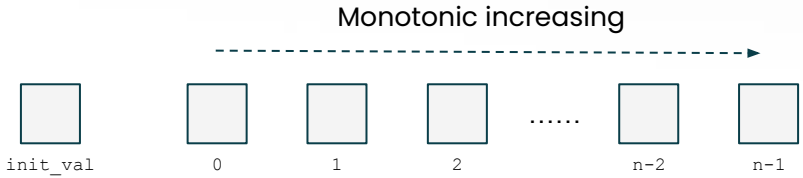



```
int IFindLastIV(int *a, int *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ?i : rdx;
  return rdx;
}
```

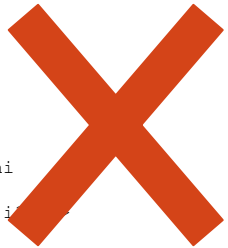
```
int FFindLastIV(float *a, float *b, int init_val, int n) {
  int rdx = init_val;
  for (int i = 0; i < n; i++)
    rdx = (a[i] > b[i]) ?i : rdx;
  return rdx;
}
```

FindLastIV Reduction Pattern

Vectorization Legality



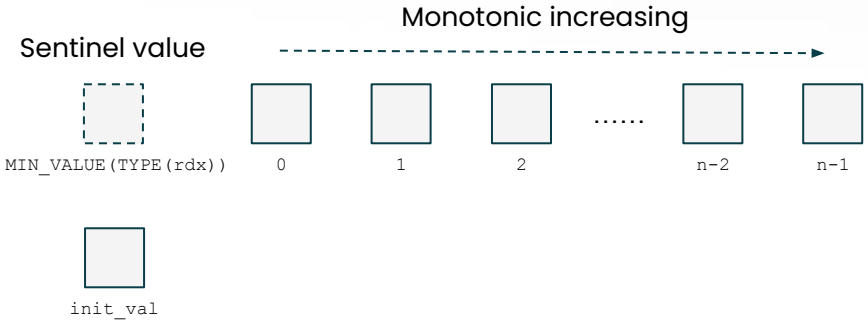

```
vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  ; Initialize vector by splatting the start value
  %vec.phi = phi <4 x i32> [ <i32 %init_val, i32 %init_val, i32 %init_val, i32 %init_val>, %vector.ph ], [ %6, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4
  %3 = getelementptr inbounds i32, ptr %b, i32 %0
  %4 = getelementptr inbounds i32, ptr %3, i32 0
  %wide.load11 = load <4 x i32>, ptr %4, align 4
  %5 = icmp sgt <4 x i32> %wide.load, %wide.load11
  %6 = select <4 x i1> %5, <4 x i32> %vec.ind, <4 x i32> %vec.phi
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i
  %7 = icmp eq i32 %index.next, %n.vec
  br i1 %7, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  ; Fix the reduction result by max reduction operation
  %8 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %6)
```



No, since the init_val is unknown, it might overlap with elements of the increasing sequence.

FindLastIV Reduction Pattern

Vectorization Legality



- The sentinel value cannot be an element in the sequence.
- The sentinel value is better to be less than all elements in the sequence.

sentinel value = MIN_VALUE(TYPE(rdx))

- The range of the sequence must be:

(sentinel value, MAX_VALUE(TYPE(rdx))]

FindLastIV Reduction Pattern

Vectorized IR Generation

```
vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  ; Initialize vector by splatting the sentinel value
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %6, %vector.body ]
```

FindLastIV Reduction Pattern

Vectorized IR Generation

```
vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  ; Initialize vector by splatting the sentinel value
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %6, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4
  %3 = getelementptr inbounds i32, ptr %b, i32 %0
  %4 = getelementptr inbounds i32, ptr %3, i32 0
  %wide.load11 = load <4 x i32>, ptr %4, align 4
  %5 = icmp sgt <4 x i32> %wide.load, %wide.load11
  %6 = select <4 x i1> %5, <4 x i32> %vec.ind, <4 x i32> %vec.phi
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %7 = icmp eq i32 %index.next, %n.vec
  br i1 %7, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  ; Fix the reduction result by max reduction operation
  %8 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %6)
```

FindLastIV Reduction Pattern

Vectorized IR Generation

```
vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  ; Initialize vector by splatting the sentinel value
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %6, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4
  %3 = getelementptr inbounds i32, ptr %b, i32 %0
  %4 = getelementptr inbounds i32, ptr %3, i32 0
  %wide.load11 = load <4 x i32>, ptr %4, align 4
  %5 = icmp sgt <4 x i32> %wide.load, %wide.load11
  %6 = select <4 x i1> %5, <4 x i32> %vec.ind, <4 x i32> %vec.phi
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %7 = icmp eq i32 %index.next, %n.vec
  br i1 %7, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  ; Fix the reduction result by max reduction operation
  %8 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %6)
  ; Sentinel value handling
  %rdx.select.cmp = icmp ne i32 %8, -2147483648
  %rdx.select = select i1 %rdx.select.cmp, i32 %8, i32 %init_val
```

Select Reduction Beyond Min/Max Reduction

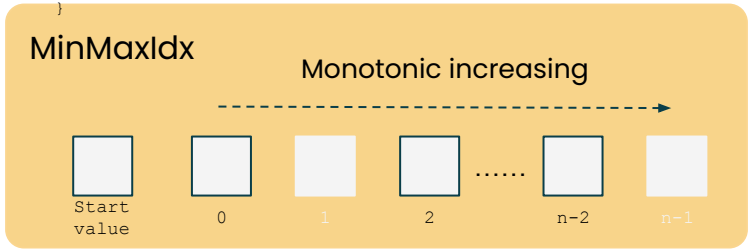
Our Goal: Min/Max with Index

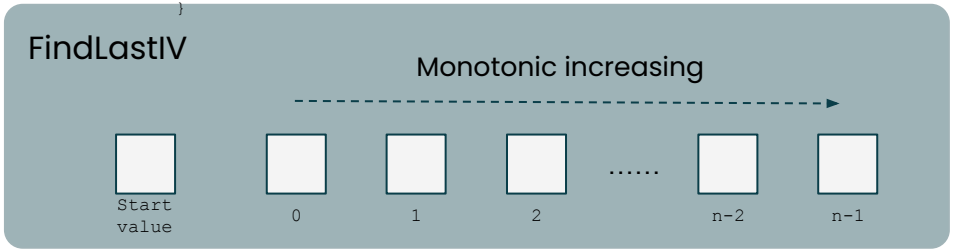
- Signed integer max with the first index

```
int SMaxFirstIdx(int *a, int init_max_val, int init_idx_val, int
n,
                 int *max_result) {
    int max = init_max_val, idx = init_idx_val;
    for (int i = 0; i < n; ++i)
        if (max < a[i]) {
            max = a[i];
            idx = i;
        }
    *max_result = max;
    return idx;
}
```

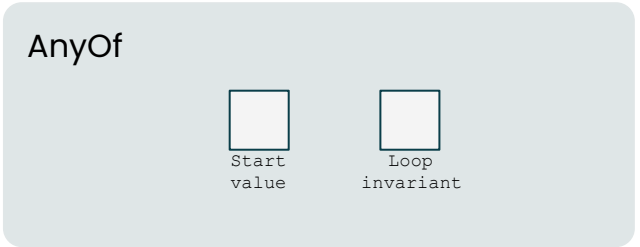
- Signed integer max with the last index

```
int SMaxLastIdx(int *a, int init_max_val, int init_idx_val, int
n,
                 int *max_result) {
    int max = init_max_val, idx = init_idx_val;
    for (int i = 0; i < n; ++i)
        if (max <= a[i]) {
            max = a[i];
            idx = i;
        }
    *max_result = max;
    return idx;
}
```





- Topics
  - Two-Stage Min/Max with Index Pattern Recognition
  - The Vectorized IR Generation of Min/Max with Index



Two-Stage Min/Max with Index Pattern Recognition

The First Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

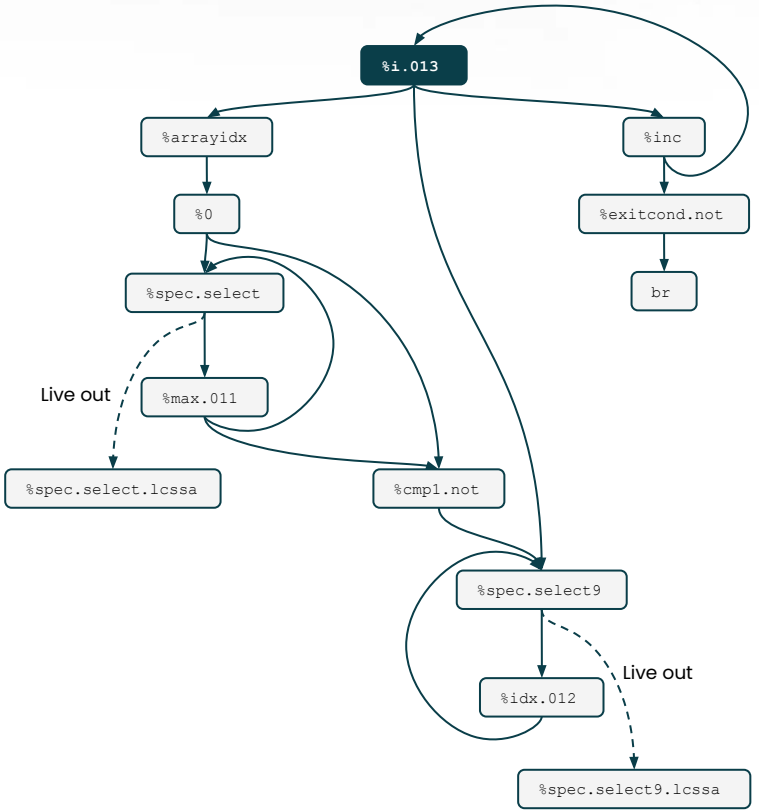


Fig. The def-use graph of signed max with last index

Two-Stage Min/Max with Index Pattern Recognition

The First Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

❶ RecurKind::IFindLastIV

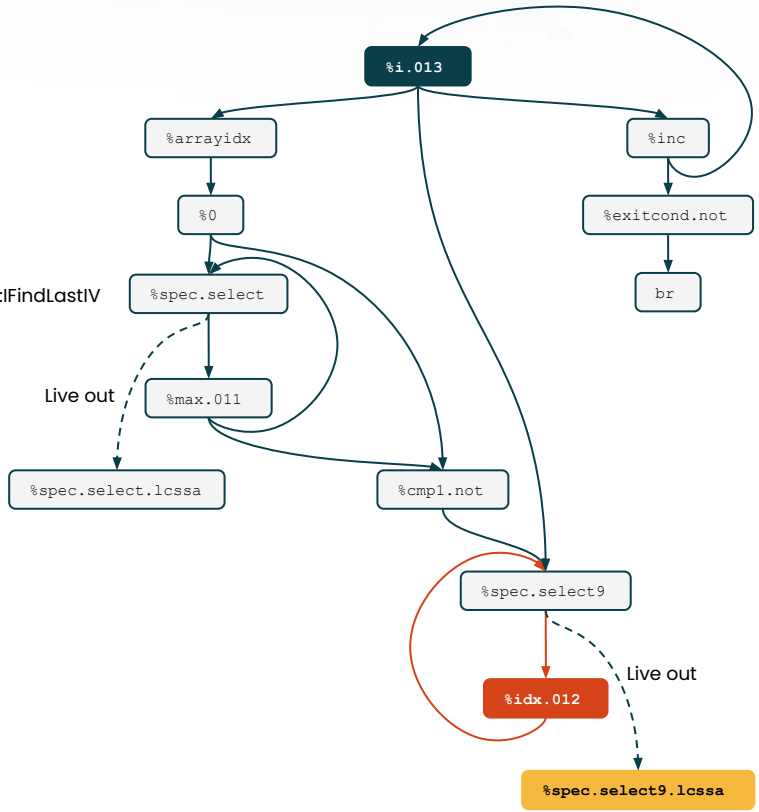


Fig. The def-use graph of signed max with last index

Two-Stage Min/Max with Index Pattern Recognition

The First Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]   ❶ RecurKind::IFindLastIV
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013      ❷ RecurKind::SMax
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

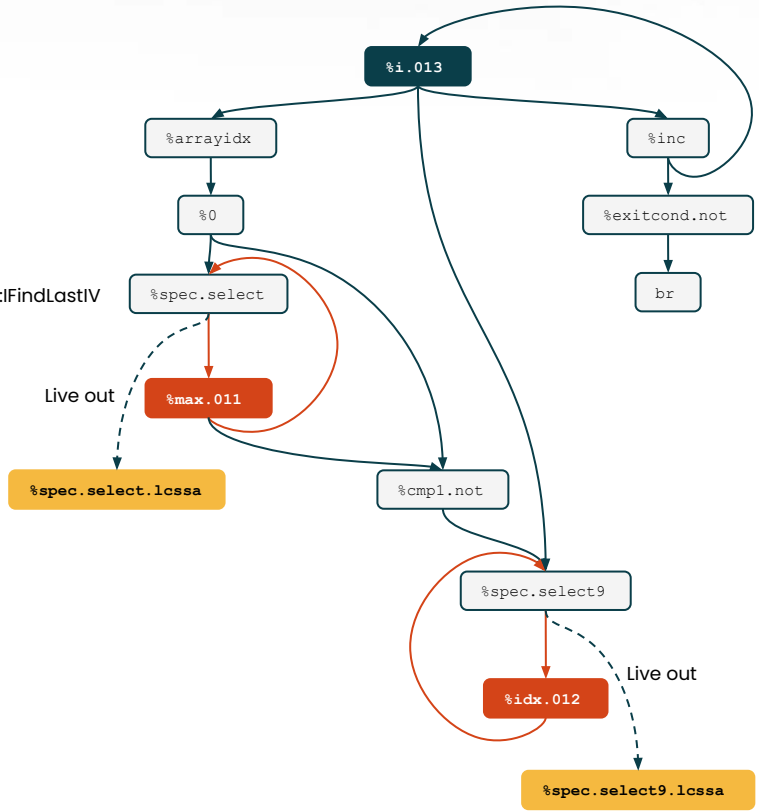


Fig. The def-use graph of signed max with last index

Two-Stage Min/Max with Index Pattern Recognition

The First Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]   ❶ RecurKind::IFindLastIV
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013      ❷ RecurKind::SMax
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

1. First index or last index
2. The index reduction PHI

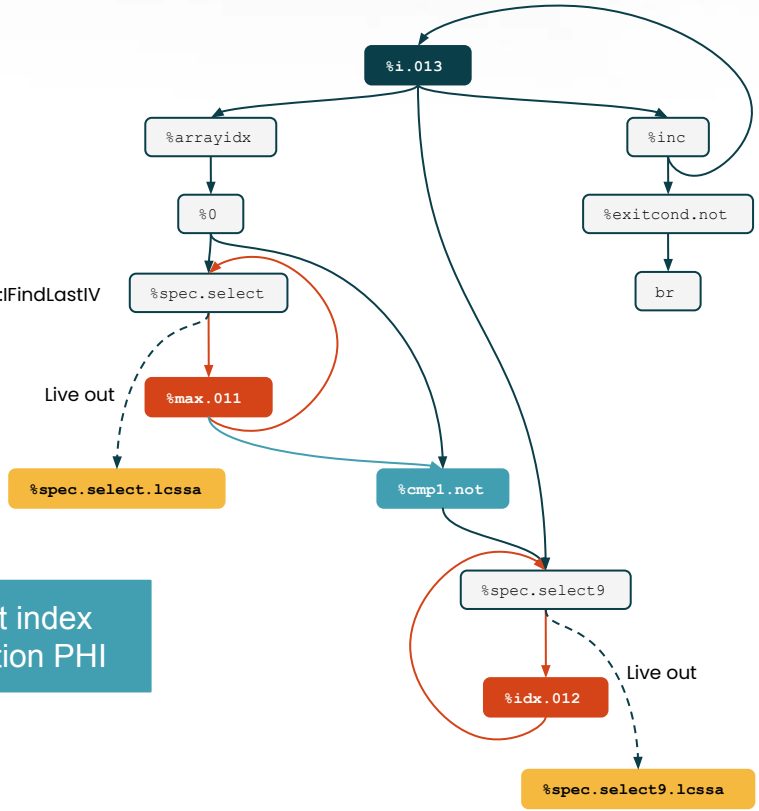
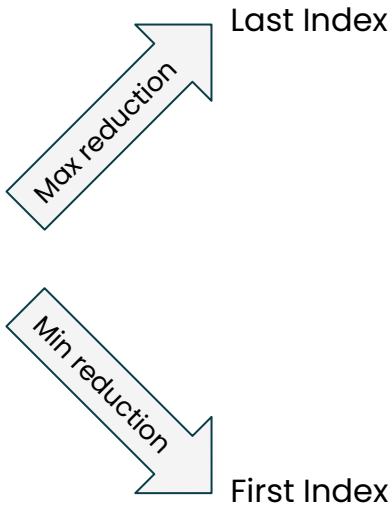


Fig. The def-use graph of signed max with last index

First Index or Last Index

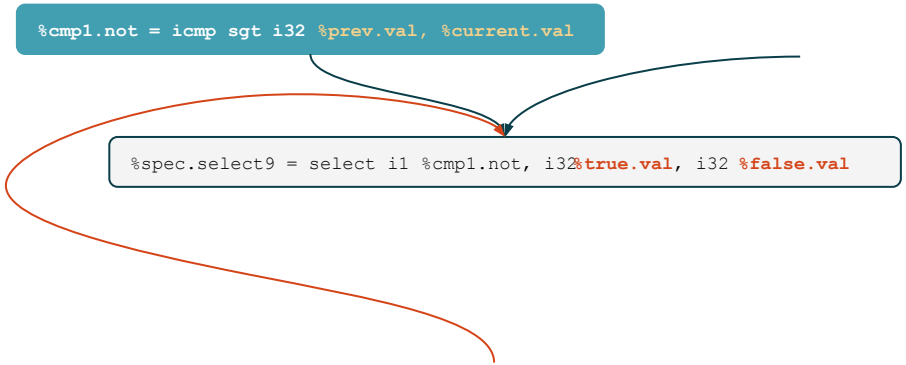
- It depends on two factors:
  - Max or Min reduction
  - cmp instruction

```
%cmp1.not = icmp sgt i32 %prev.val, %current.val
```



The Index Reduction PHI

- It depends on two factors:
  - Max or Min reduction
  - cmp instruction



Max reduction

Last Index
The index reduction PHI should define %true.val

Min reduction

First Index
The index reduction PHI should define %false.val

Two-Stage Min/Max with Index Pattern Recognition

The First Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

❶ RecurKind::IFindLastIV

❷ RecurKind::SMax

Index reduction phi = %idx.012 (true value)

1. First index or last index
2. The index reduction PHI

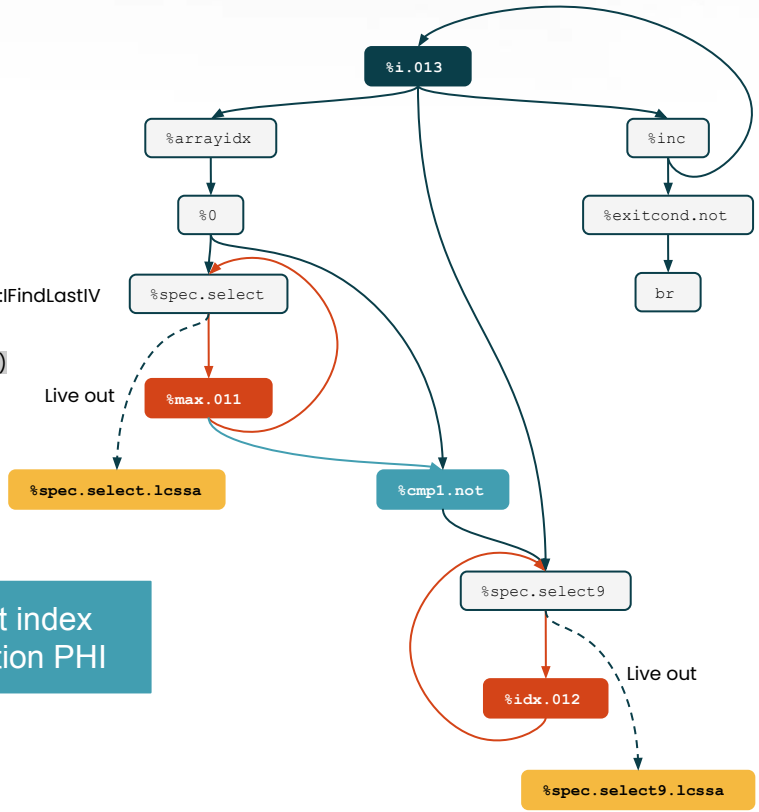


Fig. The def-use graph of signed max with last index

Two-Stage Min/Max with Index Pattern Recognition

The Second Stage Recognition

- Induction PHI
- Reduction PHI
- External user

```
...
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.013 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.012 = phi i32 [ %spec.select9, %for.body ], [ %init_idx_val, %for.body.preheader ]
  %max.011 = phi i32 [ %spec.select, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds i32, ptr %a, i32 %i.013
  %0 = load i32, ptr %arrayidx, align 4
  %cmp1.not = icmp sgt i32 %max.011, %0
  %spec.select = tail call i32 @llvm.smax.i32(i32 %max.011, i32 %0)
  %spec.select9 = select i1 %cmp1.not, i32 %idx.012, i32 %i.013
  %inc = add nuw nsw i32 %i.013, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body

for.cond.cleanup.loopexit:                        ; preds = %for.body
  %spec.select.lcssa = phi i32 [ %spec.select, %for.body ]
  %spec.select9.lcssa = phi i32 [ %spec.select9, %for.body ]
  br label %for.cond.cleanup
...
```

❶ ~~RecurKind::IFindLastIV~~
→ RecurKind::MinMaxLastIdx

❷ RecurKind::SMax
Index reduction phi = %idx.012 (true value)

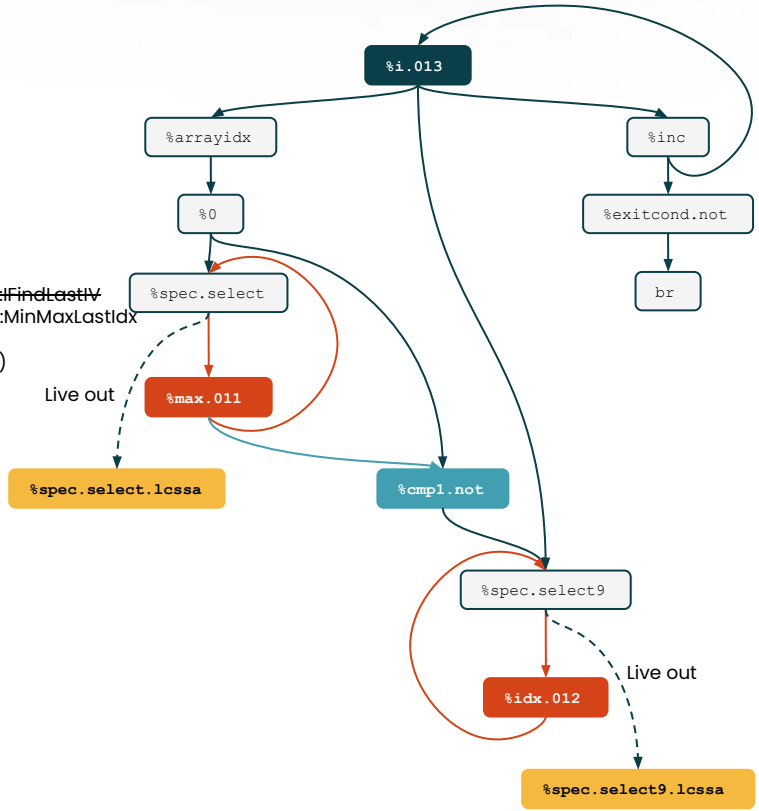


Fig. The def-use graph of signed max with last index

The IR Generation of Min/Max with Index

Signed Max with Last Index

```
  %minmax.ident.splatinsert = insertelement <4 x i32> poison, i32 %init max val, i64 0
  %minmax.ident.splat = shufflevector <4 x i32> %minmax.ident.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  br label %vector.body

vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %5, %vector.body ]
  %vec.phi15 = phi <4 x i32> [ %minmax.ident.splat, %vector.ph ], [ %4, %vector.body ]
```

❷ RecurKind::MinMaxLastIdx: Initialize vector by splatting the sentinel value

❶ RecurKind::SMax: Initialize vector by splatting the start value

The IR Generation of Min/Max with Index

Signed Max with Last Index

```
  %minmax.ident.splatinsert = insertelement <4 x i32> poison, i32 %init max val, i64 0
  %minmax.ident.splat = shufflevector <4 x i32> %minmax.ident.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  br label %vector.body

vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %5, %vector.body ]
  %vec.phi15 = phi <4 x i32> [ %minmax.ident.splat, %vector.ph ], [ %4, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4, !tbaa !4
  %3 = icmp sgt <4 x i32> %vec.phi15, %wide.load
  %4 = call <4 x i32> @llvm.smax.v4i32(<4 x i32> %vec.phi15, <4 x i32> %wide.load)
  %5 = select <4 x i1> %3, <4 x i32> %vec.phi, <4 x i32> %vec.ind
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %6 = icmp eq i32 %index.next, %n.vec
  br i1 %6, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  %7 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %4)
```

❷ RecurKind::MinMaxLastIdx: Initialize vector by splatting the sentinel value

❶ RecurKind::SMax: Initialize vector by splatting the start value

❸ Fix RecurKind::SMax: Emit the smax reduction operation

The IR Generation of Min/Max with Index

Signed Max with Last Index

```
  %minmax.ident.splatinsert = insertelement <4 x i32> poison, i32 %init max val, i64 0
  %minmax.ident.splat = shufflevector <4 x i32> %minmax.ident.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  br label %vector.body

vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %5, %vector.body ]
  %vec.phi15 = phi <4 x i32> [ %minmax.ident.splat, %vector.ph ], [ %4, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4, !tbaa !4
  %3 = icmp sgt <4 x i32> %vec.phi15, %wide.load
  %4 = call <4 x i32> @llvm.smax.v4i32(<4 x i32> %vec.phi15, <4 x i32> %wide.load)
  %5 = select <4 x i1> %3, <4 x i32> %vec.phi, <4 x i32> %vec.ind
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %6 = icmp eq i32 %index.next, %n.vec
  br i1 %6, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  %7 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %4)
  %.splatinsert = insertelement <4 x i32> poison, i32 %7, i64 0
  %.splat = shufflevector <4 x i32> %.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  %mask.cmp = icmp eq <4 x i32> %.splat, %4
```

❷ RecurKind::MinMaxLastIdx: Initialize vector by splatting the sentinel value

❶ RecurKind::SMax: Initialize vector by splatting the start value

❸ Fix RecurKind::SMax: Emit the smax reduction operation

❹ Fix RecurKind::SMax: Generate the mask representing the lanes where the maximum value is produced

The IR Generation of Min/Max with Index

Signed Max with Last Index

```
  %minmax.ident.splatinsert = insertelement <4 x i32> poison, i32 %init max val, i64 0
  %minmax.ident.splat = shufflevector <4 x i32> %minmax.ident.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  br label %vector.body

vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %5, %vector.body ]
  %vec.phi15 = phi <4 x i32> [ %minmax.ident.splat, %vector.ph ], [ %4, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4, !tbaa !4
  %3 = icmp sgt <4 x i32> %vec.phi15, %wide.load
  %4 = call <4 x i32> @llvm.smax.v4i32(<4 x i32> %vec.phi15, <4 x i32> %wide.load)
  %5 = select <4 x i1> %3, <4 x i32> %vec.phi, <4 x i32> %vec.ind
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %6 = icmp eq i32 %index.next, %n.vec
  br i1 %6, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  %7 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %4)
  %.splatinsert = insertelement <4 x i32> poison, i32 %7, i64 0
  %.splat = shufflevector <4 x i32> %.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  %mask.cmp = icmp eq <4 x i32> %.splat, %4
  %mask.select = select <4 x i1> %mask.cmp, <4 x i32> %5, <4 x i32> <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>
  %8 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %mask.select)
```

❷ RecurKind::MinMaxLastIdx: Initialize vector by splatting the sentinel value

❶ RecurKind::SMax: Initialize vector by splatting the start value

❸ Fix RecurKind::SMax: Emit the smax reduction operation

❹ Fix RecurKind::SMax: Generate the mask representing the lanes where the maximum value is produced

❺ Fix RecurKind::MinMaxLastIdx: Emit the masked smax reduction operation

The IR Generation of Min/Max with Index

Signed Max with Last Index

```
  %minmax.ident.splatinsert = insertelement <4 x i32> poison, i32 %init max val, i64 0
  %minmax.ident.splat = shufflevector <4 x i32> %minmax.ident.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  br label %vector.body

vector.body:                                      ; preds = %vector.body, %vector.ph
  %index = phi i32 [ 0, %vector.ph ], [ %index.next, %vector.body ]
  %vec.ind = phi <4 x i32> [ <i32 0, i32 1, i32 2, i32 3>, %vector.ph ], [ %vec.ind.next, %vector.body ]
  %vec.phi = phi <4 x i32> [ <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>, %vector.ph ], [ %5, %vector.body ]
  %vec.phi15 = phi <4 x i32> [ %minmax.ident.splat, %vector.ph ], [ %4, %vector.body ]
  %0 = add i32 %index, 0
  %1 = getelementptr inbounds i32, ptr %a, i32 %0
  %2 = getelementptr inbounds i32, ptr %1, i32 0
  %wide.load = load <4 x i32>, ptr %2, align 4, !tbaa !4
  %3 = icmp sgt <4 x i32> %vec.phi15, %wide.load
  %4 = call <4 x i32> @llvm.smax.v4i32(<4 x i32> %vec.phi15, <4 x i32> %wide.load)
  %5 = select <4 x i1> %3, <4 x i32> %vec.phi, <4 x i32> %vec.ind
  %index.next = add nuw i32 %index, 4
  %vec.ind.next = add <4 x i32> %vec.ind, <i32 4, i32 4, i32 4, i32 4>
  %6 = icmp eq i32 %index.next, %n.vec
  br i1 %6, label %middle.block, label %vector.body

middle.block:                                     ; preds = %vector.body
  %7 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %4)
  %.splatinsert = insertelement <4 x i32> poison, i32 %7, i64 0
  %.splat = shufflevector <4 x i32> %.splatinsert, <4 x i32> poison, <4 x i32> zeroinitializer
  %mask.cmp = icmp eq <4 x i32> %.splat, %4
  %mask.select = select <4 x i1> %mask.cmp, <4 x i32> %5, <4 x i32> <i32 -2147483648, i32 -2147483648, i32 -2147483648, i32 -2147483648>
  %8 = call i32 @llvm.vector.reduce.smax.v4i32(<4 x i32> %mask.select)
  %rdx.select.cmp = icmp ne i32 %8, -2147483648
  %rdx.select = select i1 %rdx.select.cmp, i32 %8, i32 %init_idx_val
```

❷ RecurKind::MinMaxLastIdx: Initialize vector by splatting the sentinel value

❶ RecurKind::SMax: Initialize vector by splatting the start value

❸ Fix RecurKind::SMax: Emit the smax reduction operation

❹ Fix RecurKind::SMax: Generate the mask representing the lanes where the maximum value is produced

❺ Fix RecurKind::MinMaxLastIdx: Emit the masked smax reduction operation

❻ Fix RecurKind::MinMaxLastIdx: Emit sentinel value handling

Opportunities Left on the Table

Journey has just begun

Opportunities Left on the Table

Opportunities for Improvement		TSVC	SPEC2006	SPEC2017
FindLastIV	Emit sentinel value handling on demand	s331	464.h264ref	531.deepsjeng_r
	Non-constant IV start value		403.gcc 445.gobmk 447.dealII	531.deepsjeng_r
MinMaxIdx	Single-exit min/max with index		445.gobmk 447.dealII 458.sjeng	
	FP Min/Max with index	s315 s318	447.dealII	
	Min/Max with 2-dimensions index	s3110		

The Opportunities of FindLastIV

```
// int rdx = start;
// for (int i = 0; i < n; i++)
//   rdx = <condition> ? i : rdx;

%select.iv = splat (MIN(<rdx-type>))
for () {
  ...
}
%max.rdx = reduce.smax %select.iv
%rdx = %max.rdx != MIN(<rdx-type>) ? %max.rdx : %start
```

- FindLastIV: Emit sentinel value handling on demand

```
// int rdx = -1;
// for (int i = 0; i < n; i++)
//  rdx = <condition> ? i : rdx;


%select.iv = splat (-1) // initial by the start value of reduction
for () {
  ...
}
%rdx = reduce.smax %select.iv
// No need to emit sentinel value handling
```

- FindLastIV: Non-constant IV start value

```
// int rdx = start;
// for (int i = iv_start; i < n; i++)
//  rdx = <condition> ? i : rdx;


%vmask = <all-false>
%select.iv = splat (MIN(<rdx-type>))
for () {
  ...
  %vmask |= %widen.condition // emit virtual or reduction
  ...
}
%cond.rdx = reduce.or %vmask
%max.rdx = reduce.smax %select.iv
%rdx = %cond.rdx ? %max.rdx : %start
```

The Opportunities of Min/Max with Index

```
int SMaxLastIdx(int *a, int init_max_val, int init_idx_val, int n,
                int *max_result) {
    int max = init_max_val;
    int idx = init_idx_val;
    for (int i = 0; i < n; ++i) {
        if (max <= a[i]) {
            max = a[i];
            idx = i;
        }
    }
    *max_result = max; // The external user of smax reduction
    return idx;
}
```

- MinMaxIdx: Single-exit min/max with index

```
int SMaxLastIdx(int *a, int init_max_val, int init_idx_val, int n) {
    int max = init_max_val;
    int idx = init_idx_val;
    for (int i = 0; i < n; ++i) {
        if (max <= a[i]) {
            max = a[i];
            idx = i;
        }
    }
    // No external user of smax reduction
    return idx;
}
```

The Opportunities of Min/Max with Index

- MinMaxIdx: FP Min/Max with index

```
for.body:                                         ; preds = %for.body.preheader, %for.body
  %i.012 = phi i32 [ %inc, %for.body ], [ 0, %for.body.preheader ]
  %idx.011 = phi i32 [ %idx.1, %for.body ], [ %init_idx_val, %for.body.preheader ]
  %max.010 = phi float [ %max.1, %for.body ], [ %init_max_val, %for.body.preheader ]
  %arrayidx = getelementptr inbounds float, ptr %a, i32 %i.012
  %0 = load float, ptr %arrayidx, align 4
  %cmp1 = fcmp fast ugt float %max.010, %0; cmp is not allowed to have multiple users so far
  %max.1 = select i1 %cmp1, float %max.010, float %0; select of FMax reduction
  %idx.1 = select i1 %cmp1, i32 %idx.011, i32 %i.012; select of MinMaxLastIdx reduction
  %inc = add nuw nsw i32 %i.012, 1
  %exitcond.not = icmp eq i32 %inc, %n
  br i1 %exitcond.not, label %for.cond.cleanup.loopexit, label %for.body
```

- MinMaxIdx: Min/Max with 2-dimensions index

```
int SMaxLast2DimIndex(int **a, int init_max_val,
                      int init_idx_i_val, int init_idx_j_val,
                      int n) {
    int max = init_max_val;
    int idx_i = init_idx_i_val;
    int idx_j = init_idx_j_val;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (max <= a[i][j]) {
                max = a[i][j];
                idx_i = i; // RecurKind::IAnyOf
                idx_j = j; // RecurKind::MinMaxLastIdx
            }

    return max + idx_i + idx_j;
}
```

Conclusion

- The FindLastIV and MinMaxIdx reduction patterns provide more opportunities for vectorization.
- The patch for the FindLastIV is ready for review:
  - Github PR: https://github.com/llvm/llvm-project/pull/67812
  - Phabricator: https://reviews.llvm.org/D150851
- The patch for min/max with index is work-in-progress: https://reviews.llvm.org/D143465

- Thanks to Alexey Bataev, Ayal Zaks, Florian Hahn, Ramkumar Ramachandra, Shiva Chen for their reviewing and suggestions to make this patch better.

SiFive