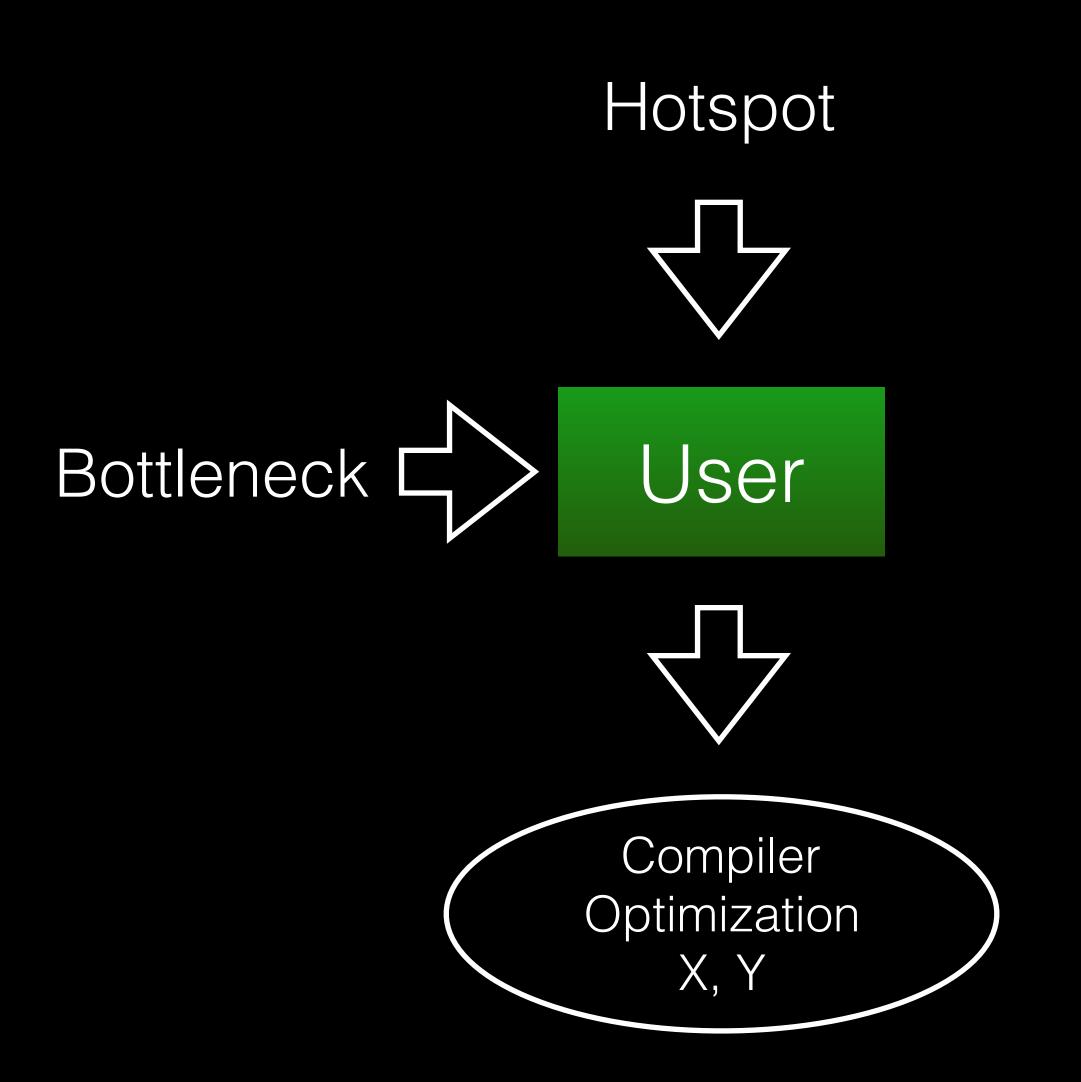
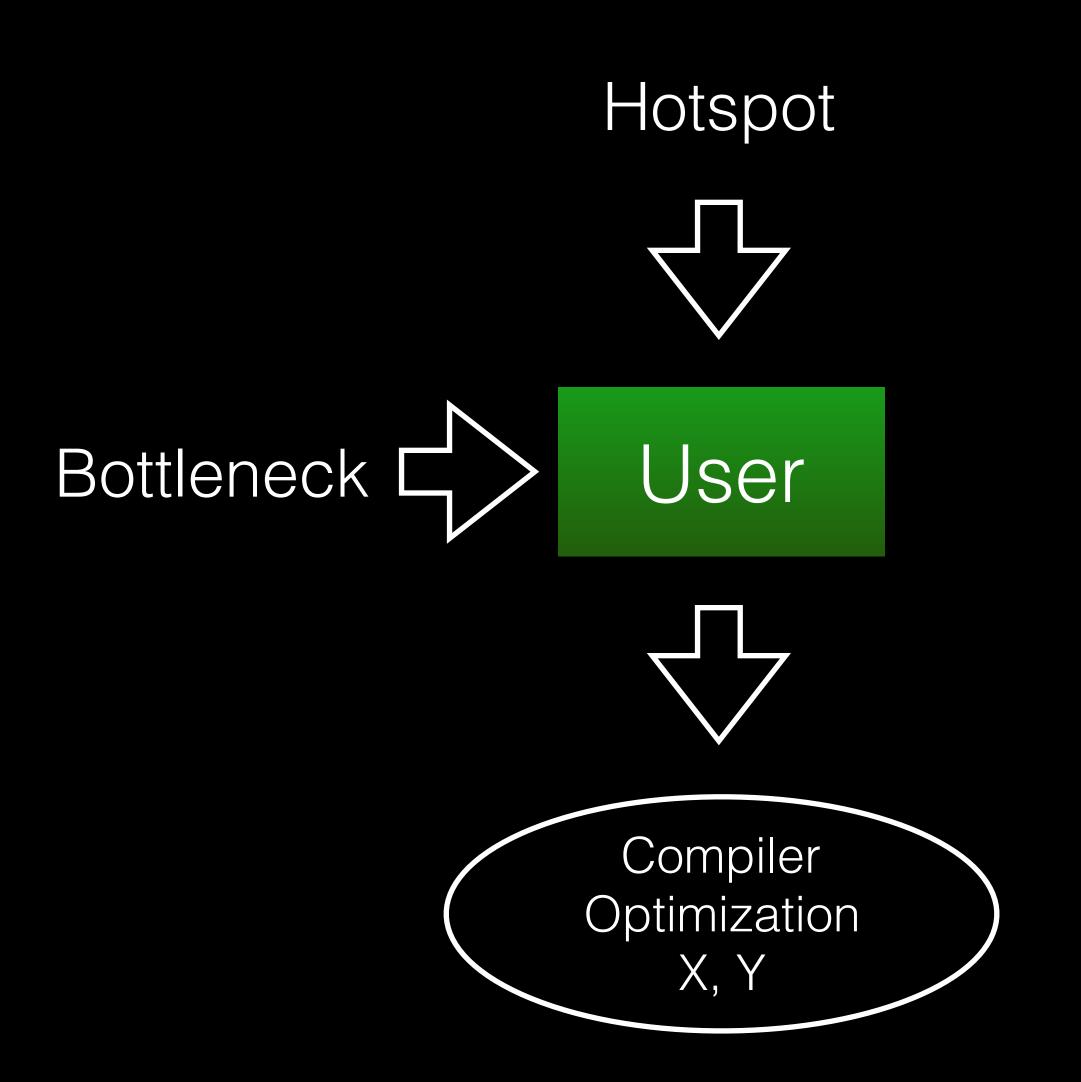
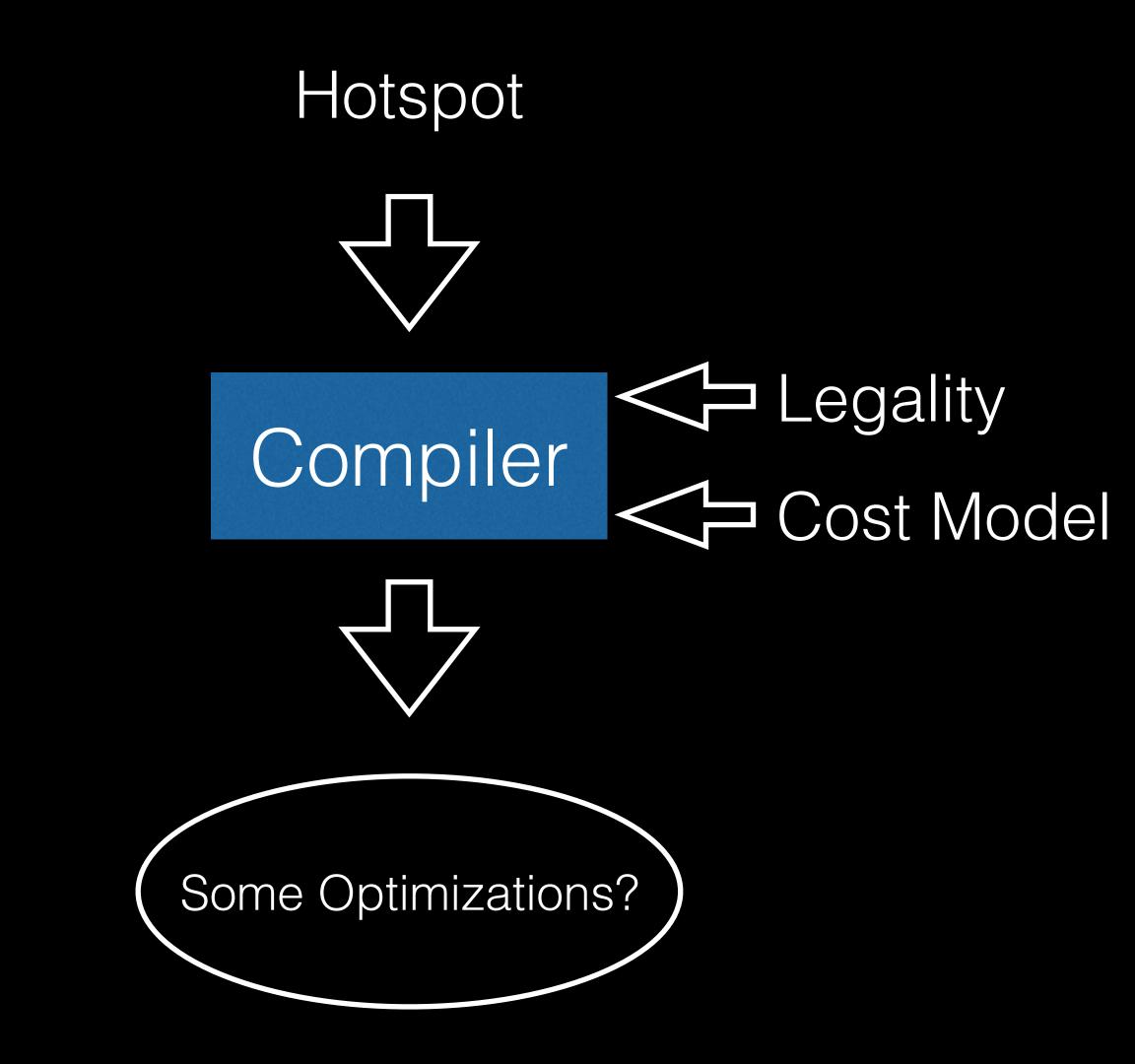
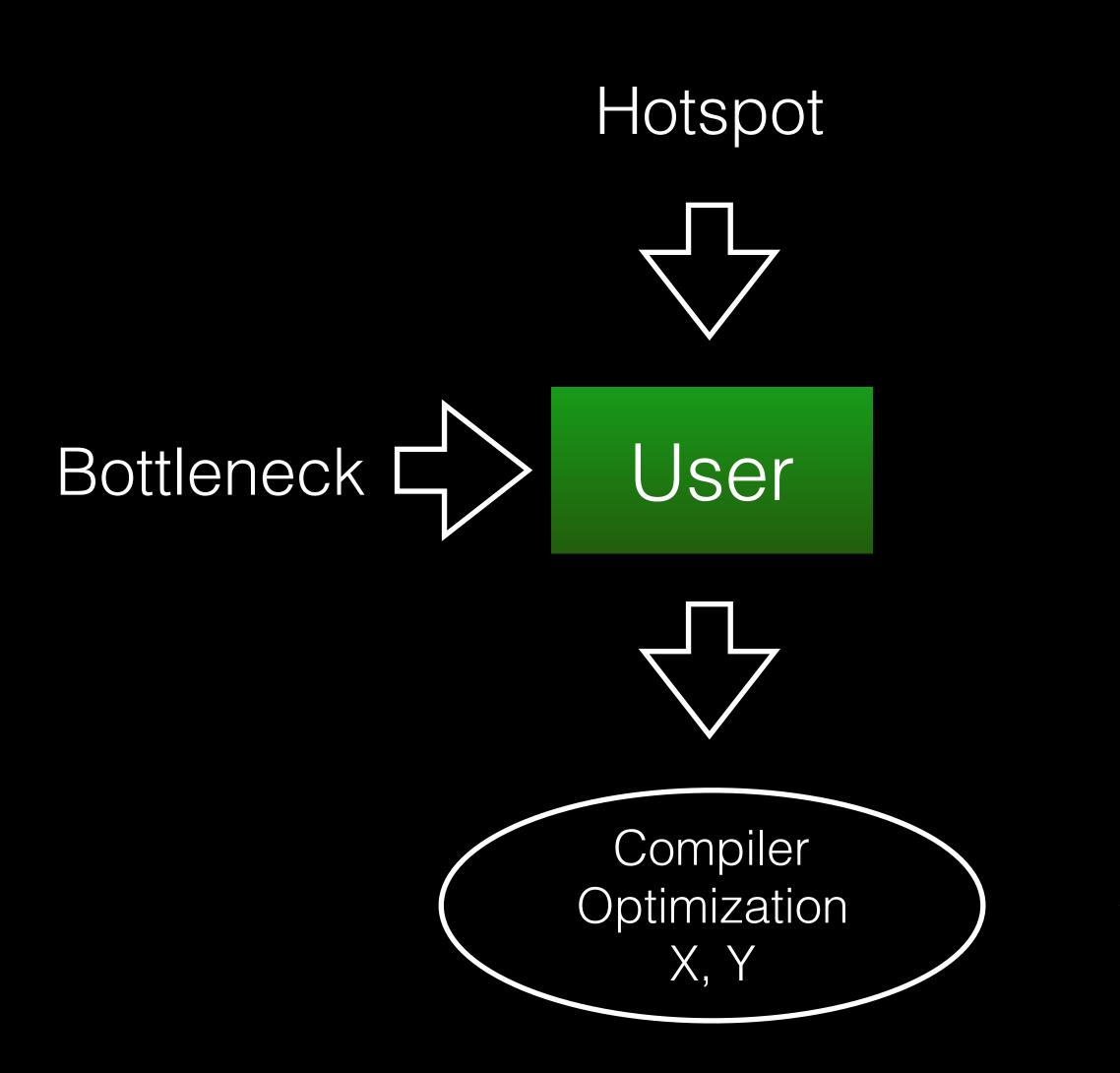
Compiler-assisted Performance Analysis

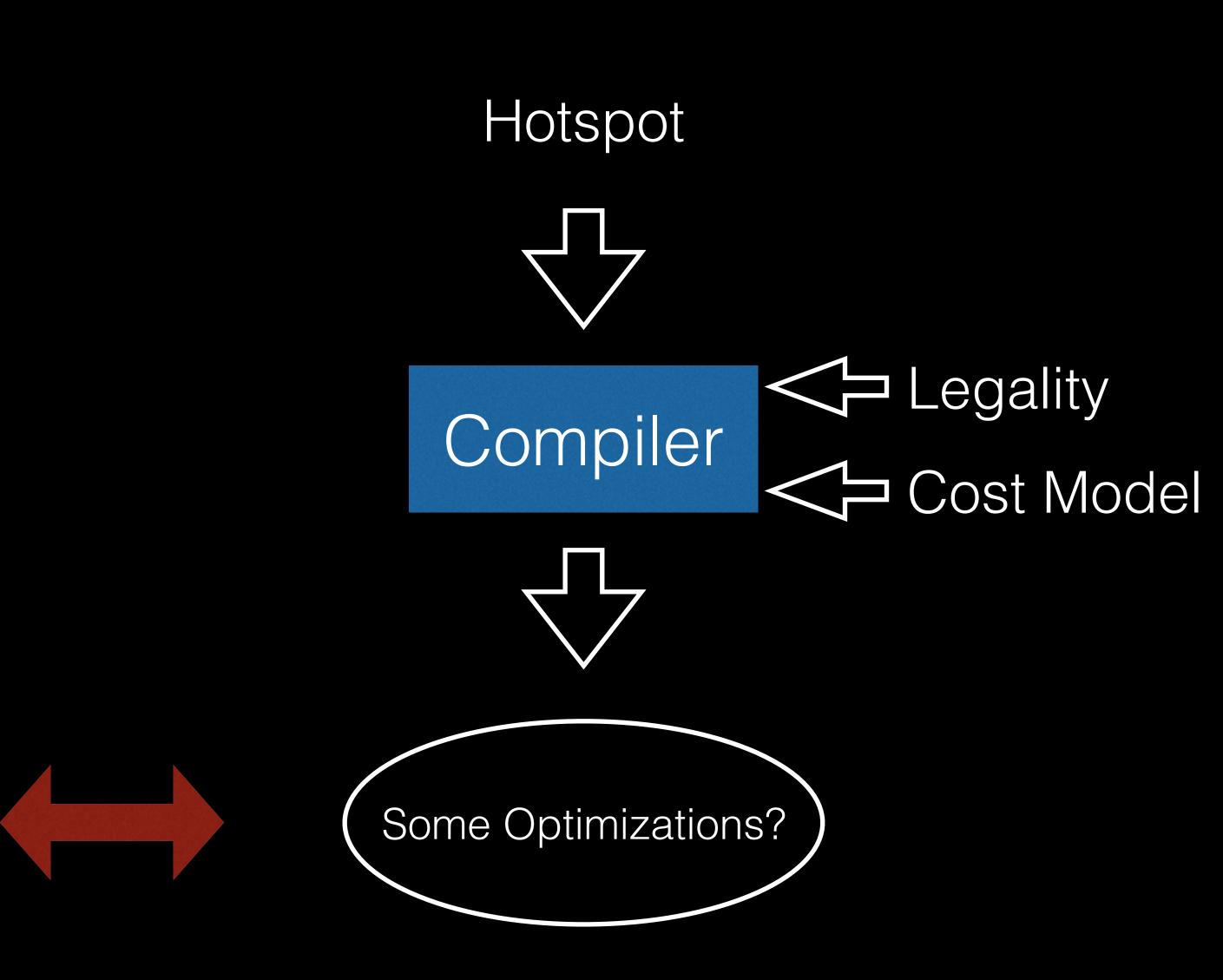
Adam Nemet Apple anemet@apple.com

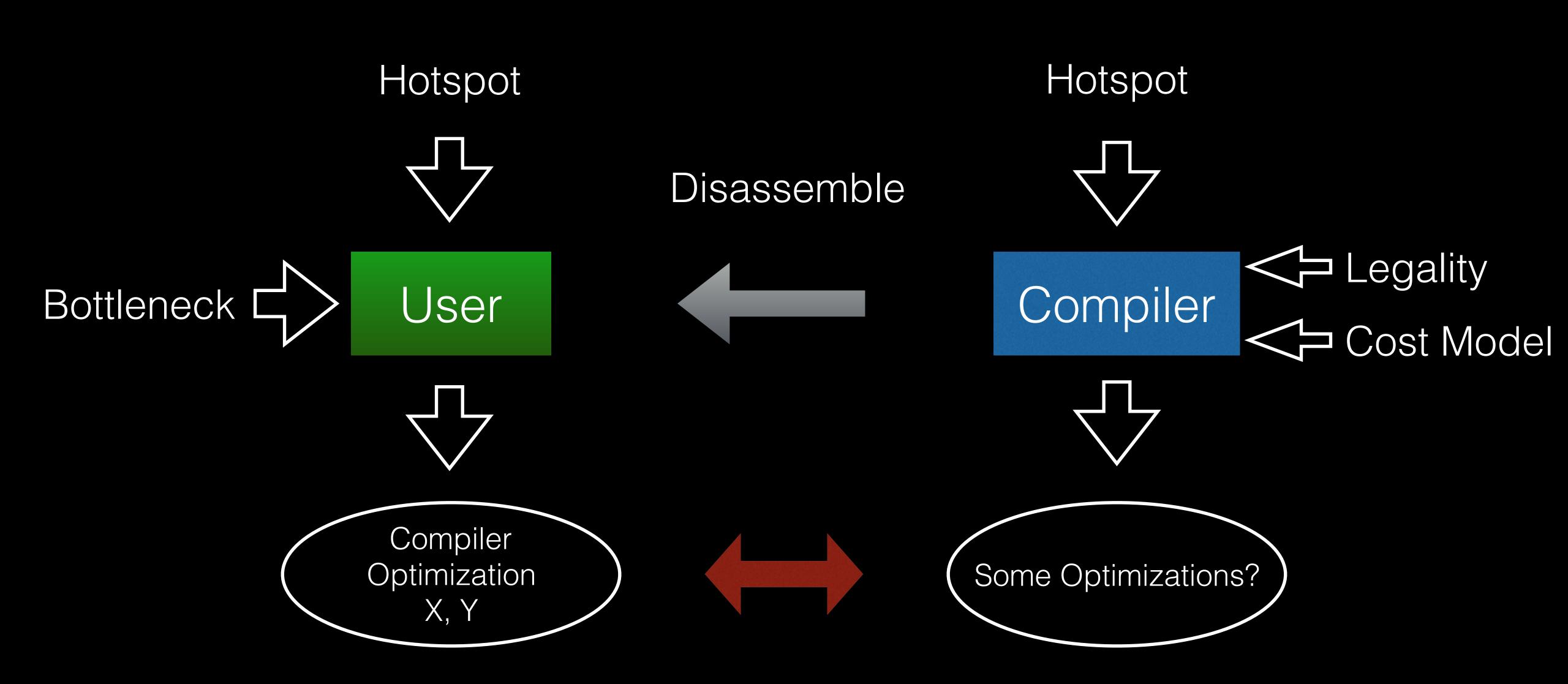


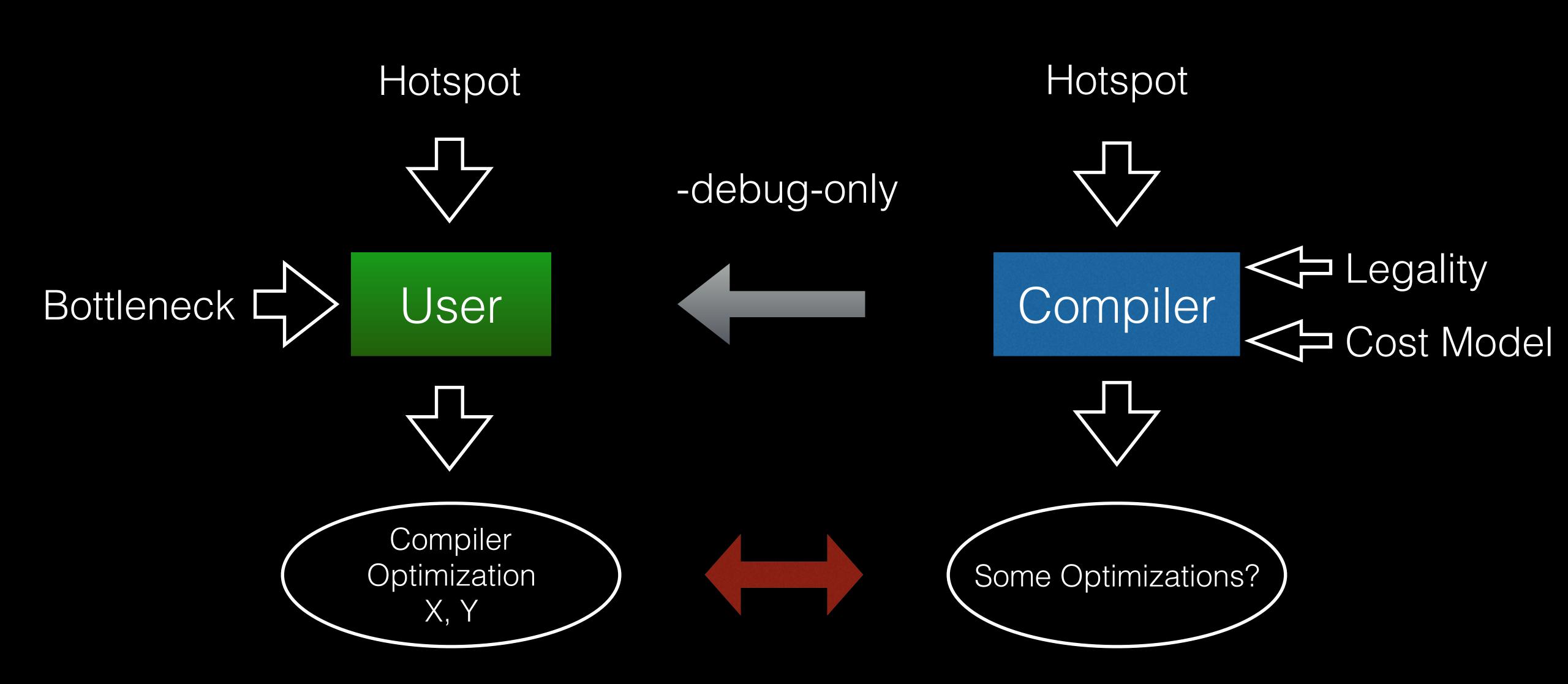


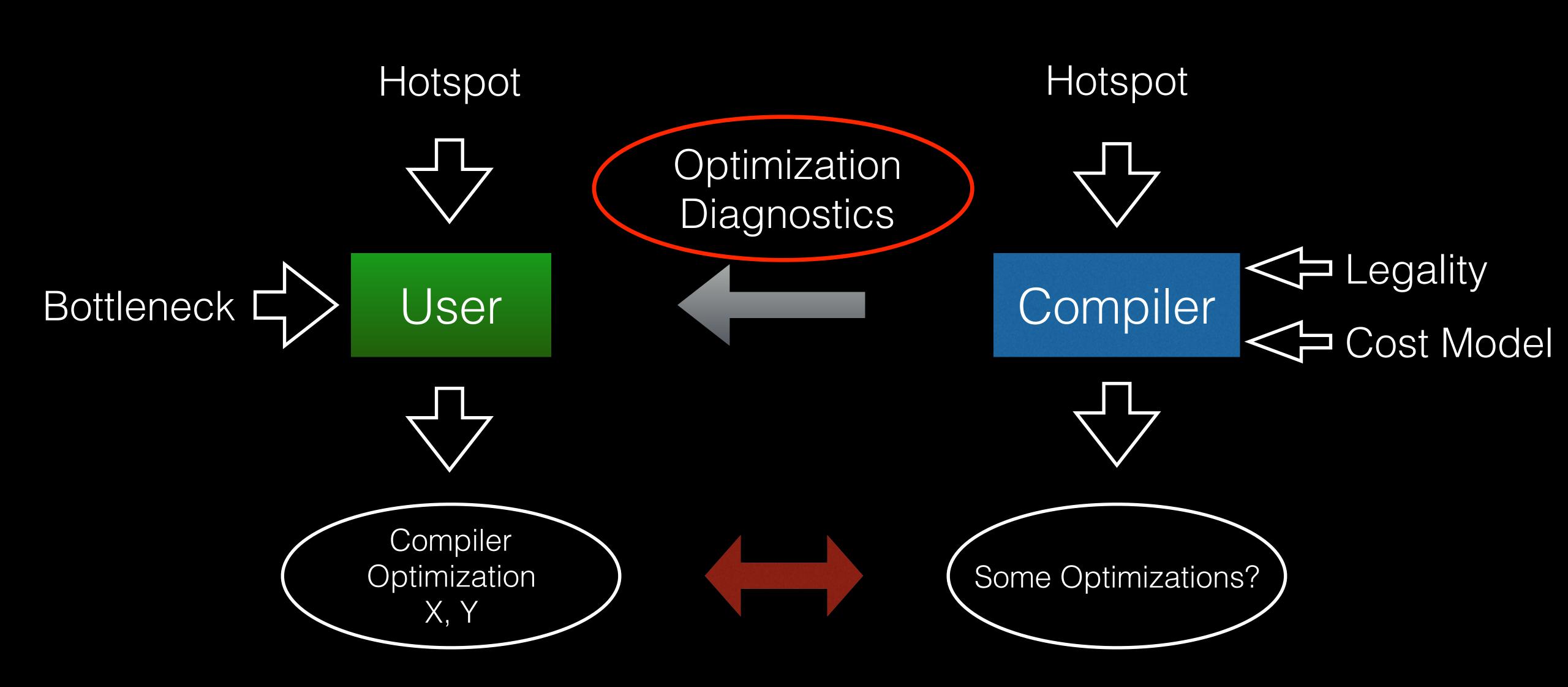












Optimization Diagnostics in LLVM

- Supported in LLVM
- Only a small number of passes emit them
- -Rpass options to enable them in the compiler output

foo.c:8:5: remark: accumulate inl
 accumulate(arr[i], sum);

foo.c:8:5: remark: accumulate inlined into compute_sum[-Rpass=inline]

Optimization Diagnostics in LLVM

- Supported in LLVM
- Only a small number of passes emit them
- -Rpass options to enable them in the compiler output
- For large programs, the output of -Rpass is noisy and unstructured

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:474:7: remark: Func1 can be inlined into Func2 with cost=10 (threshold=487) [-Rpass-analysis=inline] if (Func1(StrParI1[IntLoc], StrParI2[IntLoc+1]) == Ident1)

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:474:7: remark: Func1 inlined into Func2 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:485:7: remark: marked this call a tail call candidate [-Rpass=tailcallelim] if (strcmp(StrParI1, StrParI2) > 0)

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:443:2: remark: sext eliminated [-Rpass=gvn] for (IntIndex = IntLoc; IntIndex <= (IntLoc+1); ++IntIndex)</pre>

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:443:2: remark: sext eliminated [-Rpass=gvn] Array2Par[IntLoc+20][IntLoc] = Array1Par[IntLoc];

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:405:8: remark: Func3 can be inlined into Proc6 with cost=0 (threshold=412) [-Rpass-analysis=inline] if (! Func3(EnumParIn))

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:405:8: remark: Func3 inlined into Proc6 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:271:3: remark: Proc5 can be inlined into Proc0 with cost=10 (threshold=337) [-Rpass-analysis=inline] Proc5();

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:271:3: remark: Proc5 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:292:3: remark: Proc2 can be inlined into Proc0 with cost=25 (threshold=225) [-Rpass-analysis=inline] Proc2(&IntLoc1);

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:292:3: remark: Proc2 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:288:5: remark: Proc6 can be inlined into Proc0 with cost=-5 (threshold=412) [-Rpass-analysis=inline] Proc6(Ident1, &EnumLoc);

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:288:5: remark: Proc6 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19: remark: Func1 can be inlined into Proc0 with cost=10 (threshold=487) [-Rpass-analysis=inline] if (EnumLoc == Func1(CharIndex, 'C'))

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19: remark: Func1 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:285:3: remark: Proc1 can be inlined into Proc0 with cost=15 (threshold=337) [-Rpass-analysis=inline] Proc1(PtrGlb);

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:285:3: remark: Proc1 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:284:3: remark: Proc8 can be inlined into Proc0 with cost=125 (threshold=225) [-Rpass-analysis=inline] Proc8(Array1Glob, Array2Glob, IntLoc1, IntLoc3);

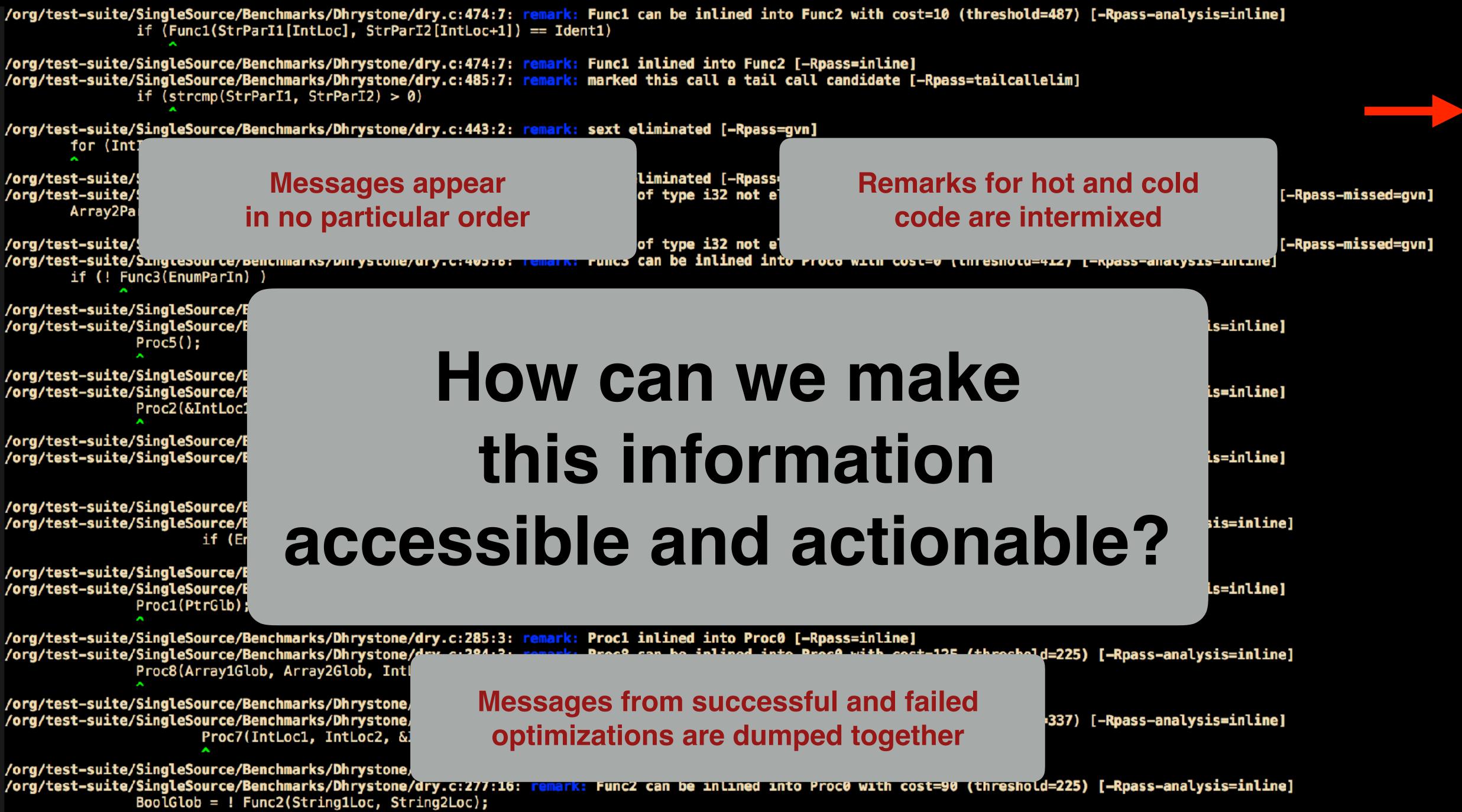
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:284:3: remark: Proc8 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:281:4: remark: Proc7 can be inlined into Proc0 with cost=-5 (threshold=337) [-Rpass-analysis=inline] Proc7(IntLoc1, IntLoc2, &IntLoc3);

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:281:4: remark: Proc7 inlined into Proc0 [-Rpass=inline] /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:277:16: remark: Func2 can be inlined into Proc0 with cost=90 (threshold=225) [-Rpass-analysis=inline] BoolGlob = ! Func2(String1Loc, String2Loc);

```
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:446:33: remark: load of type i32 not eliminated in favor of store because it is clobbered by store [-Rpass-missed=gvn]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:446:33: remark: load of type i32 not eliminated in favor of store because it is clobbered by store [-Rpass-missed=gvn]
```

4





Wish List

- All in one place: Optimizations Dashboard
- At a glance: See high-level interaction between optimizations for targeted low-level debugging
- Filtering: Noise-level should be minimized by focusing on the hot code
- Integration: Display hot code and the optimizations side-by-side

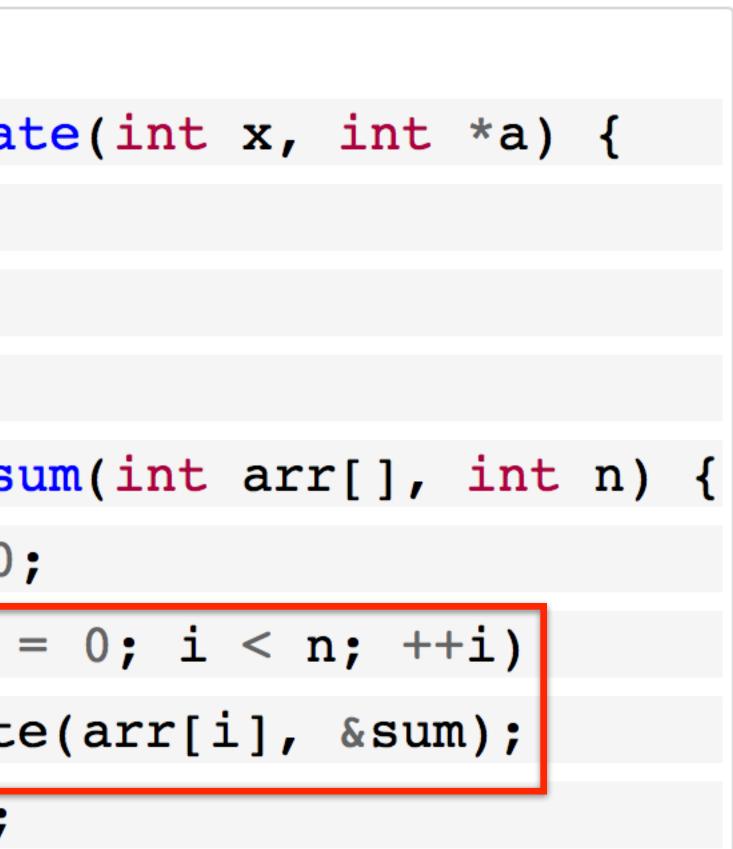
opt-viewer

Approach

- Extend existing optimization remark infrastructure
 - Add the new optimizations
 - Add ability to output remarks to a data file
- Visualize data in HTML
- Targeting compiler developers initially

Example

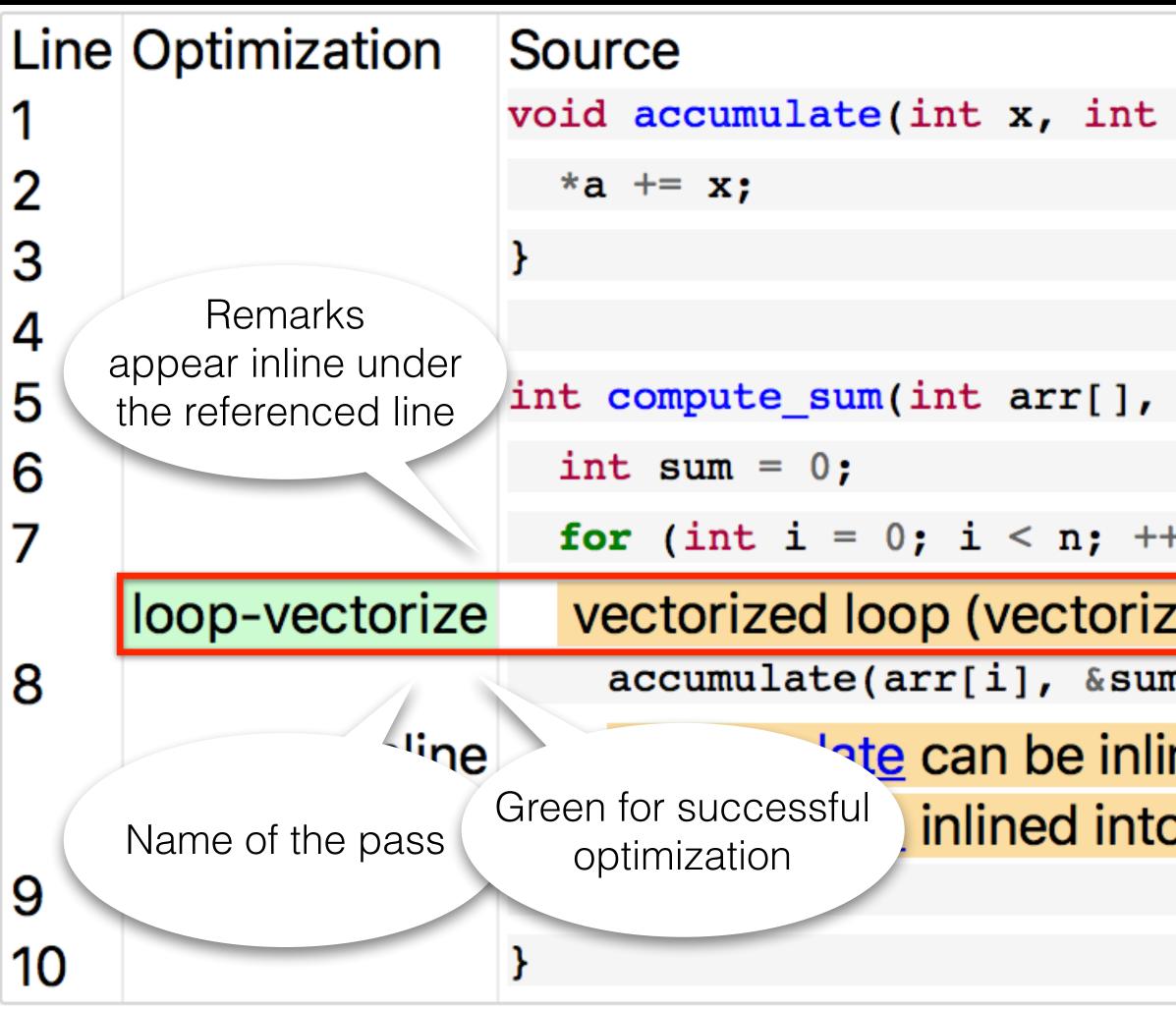
Line	Source
1	void accumula
2	*a += x;
3	}
4	
5	<pre>int compute_s</pre>
6	int sum = 0
7	for (int i
8	accumulat
9	return sum;
10	}



Work Flow

- \$ clang -03 -fsave-optimization-record -c foo.c
- \$ open html/foo.c.html

\$ utils/opt-viewer/opt-viewer.py foo.opt.yaml html

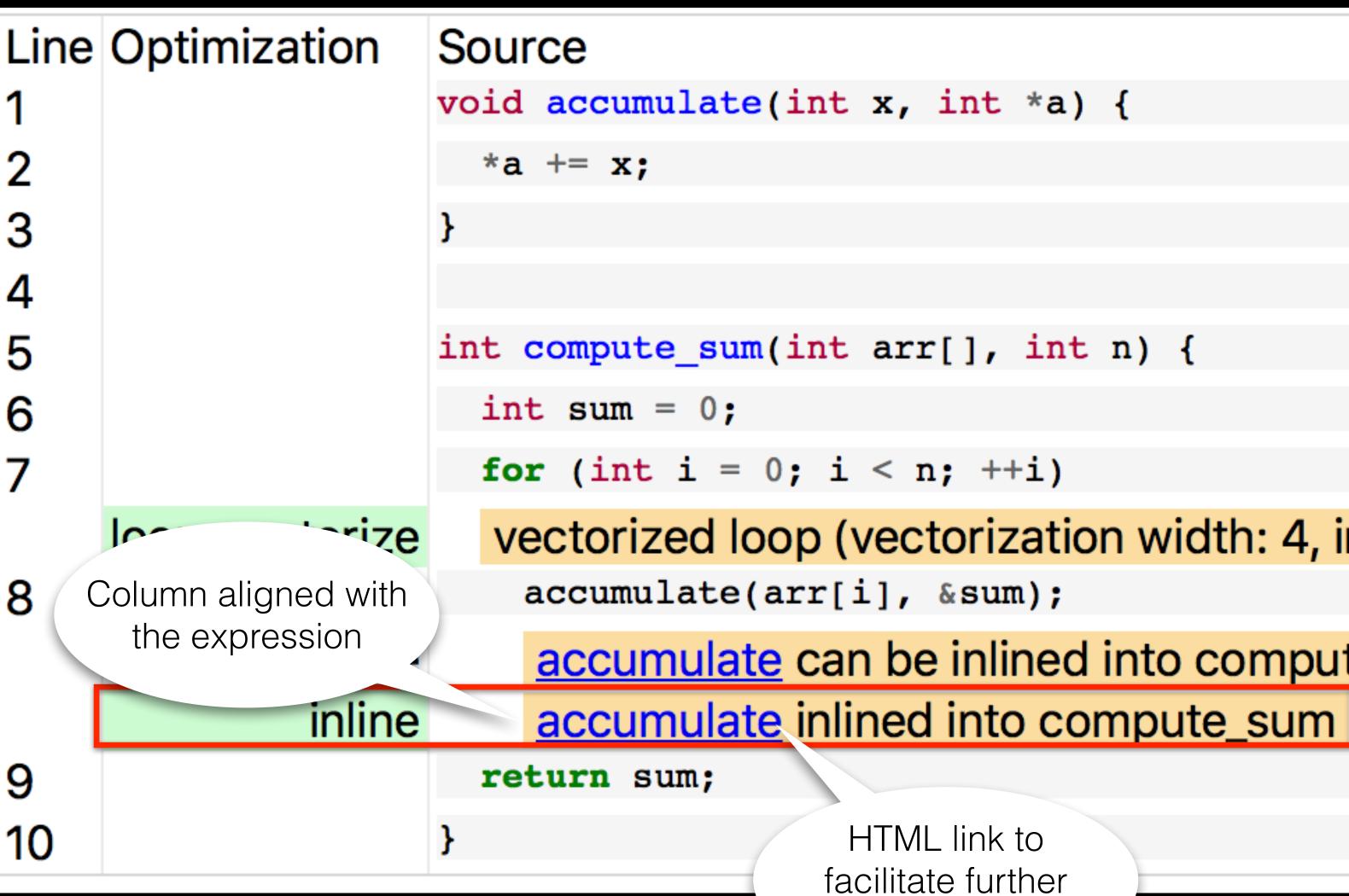


Successful Optimizations

*a) {	[
int n	1) {	Further details about the optimization	
⊦i)			
ation	width:	4, interleaved count: 2	2)
n);			
ned ir	nto com	pute_sum with cost=-	-5 (threshold

inlined into compute_sum





analysis

Successful Optimizations

vectorized loop (vectorization width: 4, interleaved count: 2)

<u>accumulate</u> can be inlined into compute_sum with cost=-5 (threshold=487)



Line	Optimization	Source
1	-	<pre>void accumulate(int x, int</pre>
2		*a += x;
3		}
4		
5 6		<pre>int compute_sum(int arr[],</pre>
		<pre>int sum = 0;</pre>
	emarks in white Analysis remarks	<pre>for (int i = 0; i < n; +</pre>
	Tize	e vectorized loop (vectorized
8		accumulate(arr[i], &su
	inline	e <u>accumulate</u> can be inl
	inline	accumulate inlined int
9		<pre>return sum;</pre>
10		}

Successful Optimizations

*a) {	
<pre>int n) {</pre>	Ontimizations can
	Optimizations can expose interesting
⊦i)	analyses
ation width: 4, in	nterleaver' count: 2)
a);	
ned into comput	te_sum with cost=-5 (threshold=
o compute_sum	



Missed Optimizations

Line	Source
1	void accumula
2	
3	int compute_s
4	int sum = 0
5	for (int i
6	accumulat
7	<pre>return sum;</pre>
8	}

te(int x, int *a); um(int arr[], int n) { ; = 0; i < n; ++i) ce(arr[i], &sum);

Missed Optimizations

Line	Optimization	Sou	irce
1			<pre>>ccumulate(int x, int *a</pre>
2	Red mea		
3	optim	izatio	on
4		lr	nt sum = 0;
5		fo	or (int i = 0; i < n; ++i)
	loop-vectorize	lo	oop not vectorized
6			<pre>accumulate(arr[i], ∑);</pre>
	inline		accumulate will not be inlin
	loop-vectorize		loop not vectorized: call in
7		re	eturn sum;
8		}	

);

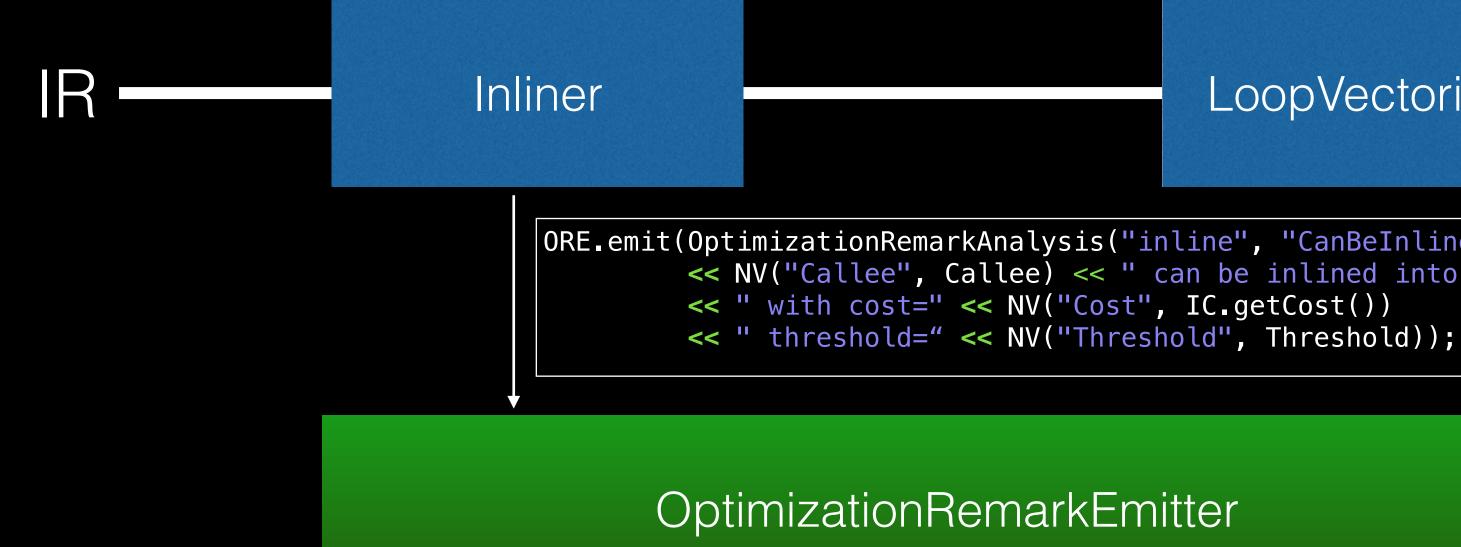
t n) {

ned into compute_sum because its definition is unavailable struction cannot be vectorized



LLVM Changes

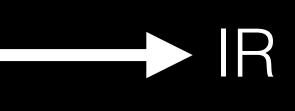
Pass pipeline



-Rpass-analysis=inline

foo.c:8:5: remark: accumulate can be inlined into compute_sum with cost=-5 (threshold=487) [-Rpass-analysis=inline] accumulate(arr[i], sum); $\mathbf{\wedge}$

LoopVectorizer

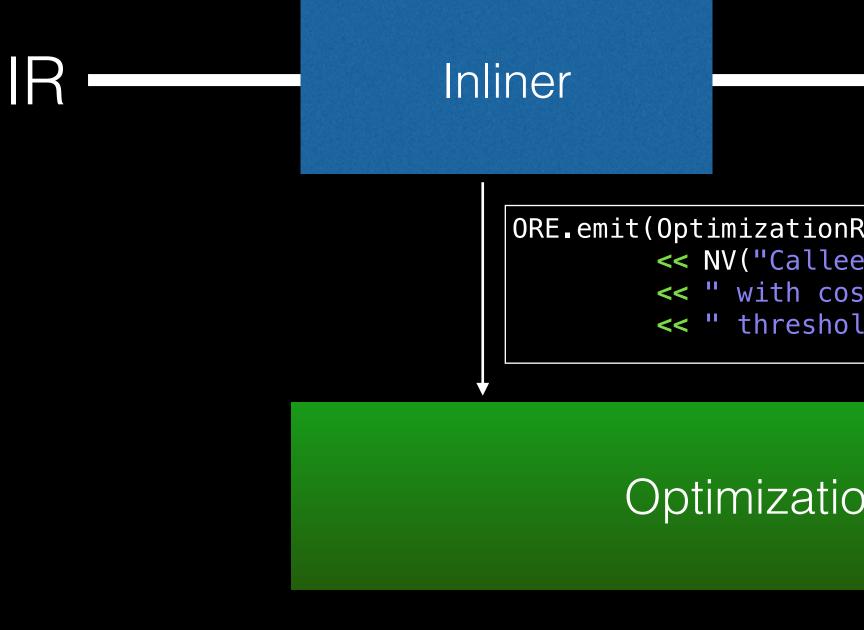


ORE.emit(OptimizationRemarkAnalysis("inline", "CanBeInlined", Call) << NV("Callee", Callee) << " can be inlined into " << NV("Caller", Caller)</pre>



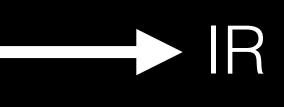
LLVM Changes

Pass pipeline



-fsave-optmization-record enables source line debug info (-gline-tables-only)

LoopVectorizer



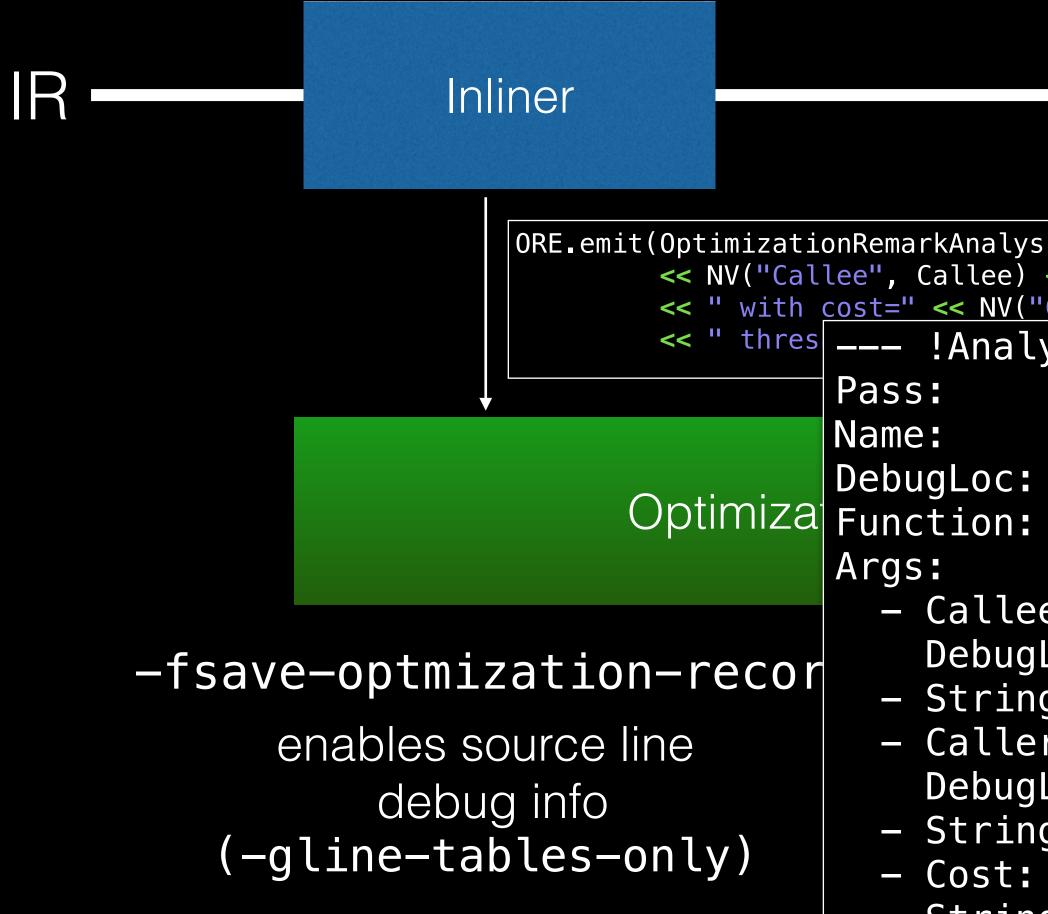
OptimizationRemarkEmitter



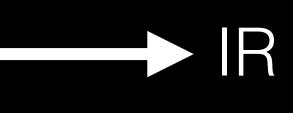


LLVIVI Changes

Pass pipeline



LoopVectorizer



```
ORE.emit(OptimizationRemarkAnalysis("inline", "CanBeInlined", Call)
       << NV("Callee", Callee) << " can be inlined into " << NV("Caller", Caller)
       << " with cost=" << NV("Cost", IC.getCost())</pre>
       << " thres ____ !Analysis</pre>
                                    inline
                Pass:
                                    CanBeInlined
                Name:
                                    { File: s.cc, Line: 8, Column: 5 }
                DebugLoc:
                                    compute_sum
                Args:
                   - Callee:
                                        accumulate
                                         { File: s.cc, Line: 1, Column: 0 }
                     DebugLoc:
                                           can be inlined into '
                   - String:
                   - Caller:
                                        compute_sum
                                         { File: s.cc, Line: 5, Column: 0 }
                     DebugLoc:
                                           with cost='
                   - String:
                                         '-5'
                   - Cost:
                                           (threshold='
                   - String:
                                         '487'
                   – Threshold:
                                         - String:
```

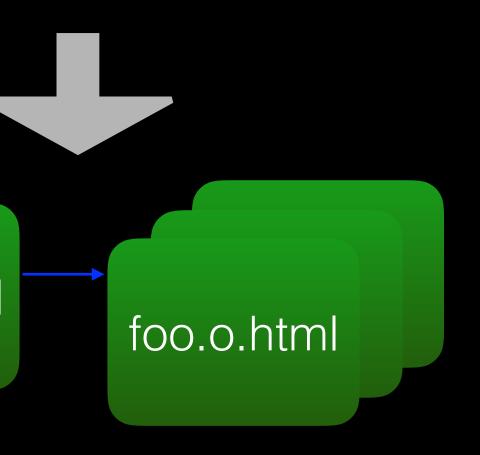


index.html

opt-viewer



utils/opt-viewer/opt-viewer.py







Source Location

/org/test-suite/SingleSource/Benchmarks/Dhrystone/ /org/test-suite/SingleSource/Benchmarks/Dhrystone/

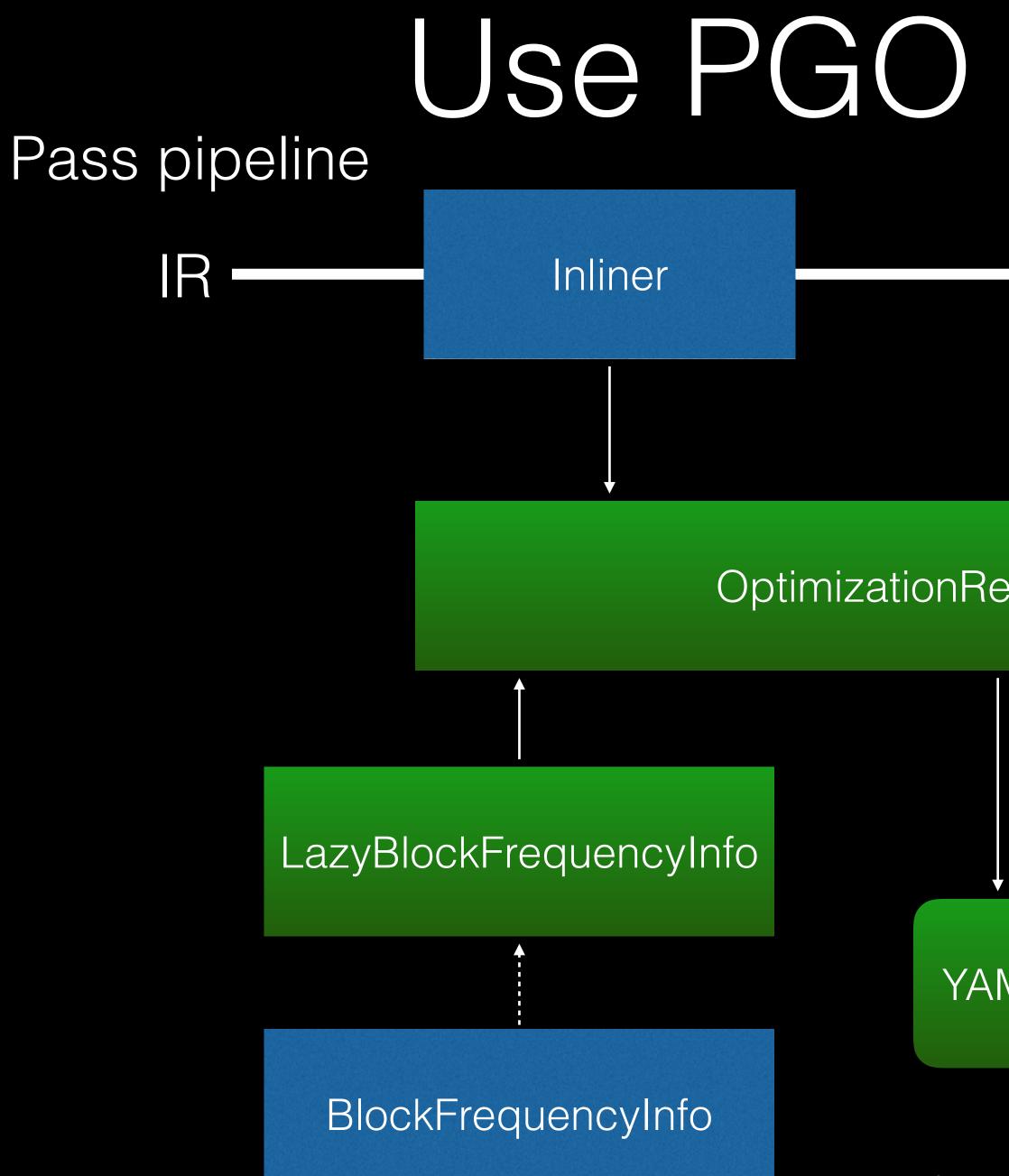
Index

	Function	Pass
<u>/dry.c:485:7</u>	Func2	inline
/dry.c:474:7	Func2	inline
/dry.c:474:7	Func2	inline
/dry.c:219:14	Proc0	inline
/dry.c:221:13	Proc0	inline
/dry.c:245:27	Proc0	inline
/dry.c:246:23	Proc0	inline
/dry.c:251:2	Proc0	inline
/dry.c:261:14	Proc0	inline
/dry.c:271:3	Proc0	inline
/dry.c:271:3	Proc0	inline
/dry.c:292:3	Proc0	inline
/dry.c:292:3	Proc0	inline
/dry.c:288:5	Proc0	inline
<u>/dry.c:288:5</u>	Proc0	inline
/dry.c:287:19	Proc0	inline

Source Location	
/org/test-suite/SingleSource/Benchma	<u>rks/Dhrystone/</u>
/org/test-suite/SingleSource/Benchma	rks/Dhrystone/
/org/test-suite/SingleSource/Benchma	rke/Dhrvetone/
/org/test-suite/SingleSource/Bend	
/org/test-suite/SingleSource/Bend	
/org/test-suite/SingleSource/Bend	Most of th
/org/test-suite/SingleSource/Benchma	<u></u>
/org/test-suite/SingleSource/Benc'	
/org/test-suite/SingleSource/Benc	
/org/test-suite/SingleSource/Benc	Sort
/org/test-suite/SingleSource/Benc	
/org/test-suite/SingleSource/Benchma	rks/Dnrystone/
/org/test-suite/SingleSource/Benchma	rks/Dhrystone/

Index

	Function	Pass	
<u>/dry.c:485:7</u>	Func2		inline
<u>/dry.c:474:7</u>	Func2		inline
Idn/ c·171.7	Eune?		inline
Neieur			inline
Noisy:			inline
this code I	not hot		inline
<u>/ al j. a. 2 - a. 2 a</u>			inline
			inline
			inline
t by hotne	SS		inline
			inline
<u>/ary.c:292:3</u>	Procu		inline
/dry.c:292:3	Proc0		inline
<u>/dry.c:288:5</u>	Proc0		inline
<u>/dry.c:288:5</u>	Proc0		inline
<u>/dry.c:287:19</u>	Proc0		inline
	-		

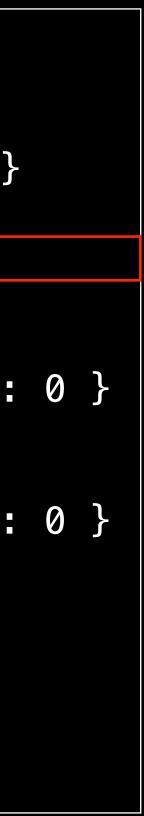


Use PGO for Hotness

	!Analysis	
	Pass: Name: DebugLoc: Function: Hotness:	<pre>inline CanBeInlined { File: s.cc, Line: 8, Column: 5 } compute_sum 3</pre>
	Args:	
em	- Callee: DebugLoc:	<pre>accumulate { File: s.cc, Line: 1, Column: ' can be inlined into ' compute_sum { File: s.cc, Line: 5, Column: ' with cost=' '-5' ' (threshold=' '487' ')'</pre>

YAML





Hotness

Source Location

/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:443:2 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:473:2 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:473:2 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:443:2 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:444:31 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19 /org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:474:7



Hotness	Function	Pass
100%	Proc8	loop-vectorize
65%	Func2	loop-vectorize
65%	Func2	loop-vectorize
52%	Proc0	loop-delete
52%	Proc0	loop-idiom
36%	Proc0	inline
36%	Proc0	inline
34%	Eunc2	inline

Optimizations Recorded

Function Inliner Loop Vectorizer Loop Unroller LoopDataPrefetch

... more to follow

LICM

Loop Idiom

Loop Deletion

SLP Vectorizer

Test Drive ON LLVM test suite

- Does the information presented in this high-level view contain É. sufficient detail to reconstruct what happened?
- 2. Can we discover the interactions between optimizations?
- 3. With the improved visibility, can we quickly find real performance opportunities?

Improve & Evaluate

DhryStone (SingleSource/Benchmark)

Interaction of Optimizations

DhryStone

430			Proc8(Array1	Par, Array2	Par,	IntParI1,	IntParI2
431			Array1Dim	· -	-		
432			Array2Dim	Array2P			
433			OneToFifty	IntParI	1;		
434			OneToFifty	IntParI	2;		
435			{				
436			REG	OneToFifty	IntL	oc;	
437			REG	OneToFifty	IntI	ndex;	
438							
439			IntI	Loc = IntPar	I1 +	5;	
440			Arra	aylPar[IntLo	c] =	<pre>IntParI2;</pre>	
441			Arra	ay1Par[IntLo	c +1]	<pre>= Array1Pa</pre>	ar[IntLoc
442			Arra	ay1Par[IntLo	c +30]	= IntLoc	;
443			for	(IntIndex =	IntL	oc; IntIn	dex <= (I
	52%	loop-delete	loop	deleted			
	100%	loop-vectorize	vect	torized loop (v	ectori	zation width	n: 4, interle
444				Array2P	ar[In	tLoc][Int]	[ndex] = 1
	52%	loop-idiom					fo
445			++Ar	ray2Par[Int]	Loc][IntLoc-1]	;
	17%	gvn		l of type i32 no			
	17%	gvn	load	l of type i32 no	ot elin	ninated in fa	vor of store
446			Arra	ay2Par[IntLo	c +20]	[IntLoc]	= Array1Pa
	34%	gvn					load of t
	17%	gvn					load of t
447			Inte	lob = 5;			
448			}				

' Inli	Inlining C		
;			
t call \			
<pre>htLoc+1); ++IntIndex)</pre>		roc0	
aved count: 2)		roc8	
IntLoc;			
med memset	P	roc0	
because it is clobbered by store	P	<u>roc0</u>	
because it is clobbered by <u>call</u>	P	<u>roc0</u>	
ar[IntLoc];		0	
pe i32 not eliminated in favor of store because it is clobbered by store		roc8	
pe i32 eliminated	P	<u>roc0</u>	



DhryStone

				_			
430			Proc8(Array	1Par, Array2	Par, IntParI1,	, IntParI2)	
431			Array1Dim	Array1P	er;		
432			Array2Dim	Array2P	ar;		
433			OneToFifty	IntParI	:1;		
434			OneToFifty	IntParI	2;		
435			{				
436			REG	OneToFifty	IntLoc;		
437			REG	OneToFifty	<pre>IntIndex;</pre>		
438							
439			Int	Loc = IntPar	:I1 + 5;		
440			<pre>Array1Par[IntLoc] = IntParI2;</pre>				
441			<pre>Array1Par[IntLoc+1] = Array1Par[IntLoc];</pre>				
442			<pre>Array1Par[IntLoc+30] = IntLoc;</pre>				
443			for	(IntIndex =	IntLoc; IntIr	ndex <= (IntLoc+1)	
	100%	loop-vectorize	vec	torized loop (v	ectorization widt	th: 4, interleaved cou	
444						tIndex] = IntLoc;	
445			++A:	rray2Par[Int	Loc][IntLoc-1]	;	
446			Arra	ay2Par[IntLo	c+20][IntLoc]	<pre>= Array1Par[IntLo</pre>	
	34%	gvn				load of type i32 n	
	U 170	9111					
447			Int	Glob = 5;			
448			}				
440			1				

IntLoc+1); ++IntIndex)

eaved count: 2)

Proc8

Par[IntLoc]; type i32 not eliminated in favor of store because it is clobbered by store

Proc8



DhryStone

430			Proc8(Array)	lPar, Array2	Par, IntP	arI1,	IntParI2	
431			Array1Dim	Array1P	ar;			
432			Array2Dim	Array2P	ar;			
433			OneToFifty	IntParI	1;			
434			OneToFifty	IntParI	2;			
435			{					
436			REG	OneToFifty	IntLoc;			
437			REG	OneToFifty	IntIndex	;		
438								
439			Intl	Loc = IntPar	I1 + 5;			
440			<pre>Array1Par[IntLoc] = IntParI2;</pre>					
441			Arra	ay1Par[IntLo	c+1] = Ar	ray1Pa	ar[IntLoc]	
442			Arra	ay1Par[IntLo	c+30] = I	ntLoc	;	
443			for	(IntIndex =	IntLoc;	IntInd	dex <= (In	
	100%	loop-vectorize	vec	torized loop (v	ectorizatio	n width	n: 4, interlea	
444				Array2P	ar[IntLoc][Int]	[ndex] = 1	
445			++A1	rray2Par[Int	Loc][IntL	oc-1];	;	
446			Arra	ay2Par[IntLo	c+20][Int	Loc] =	= Array1Pa	
	34%	gvn					load of t	
447			Into	lob = 5;				
448			}					

];

IntLoc+1); ++IntIndex)

eaved count: 2)

IntLoc;

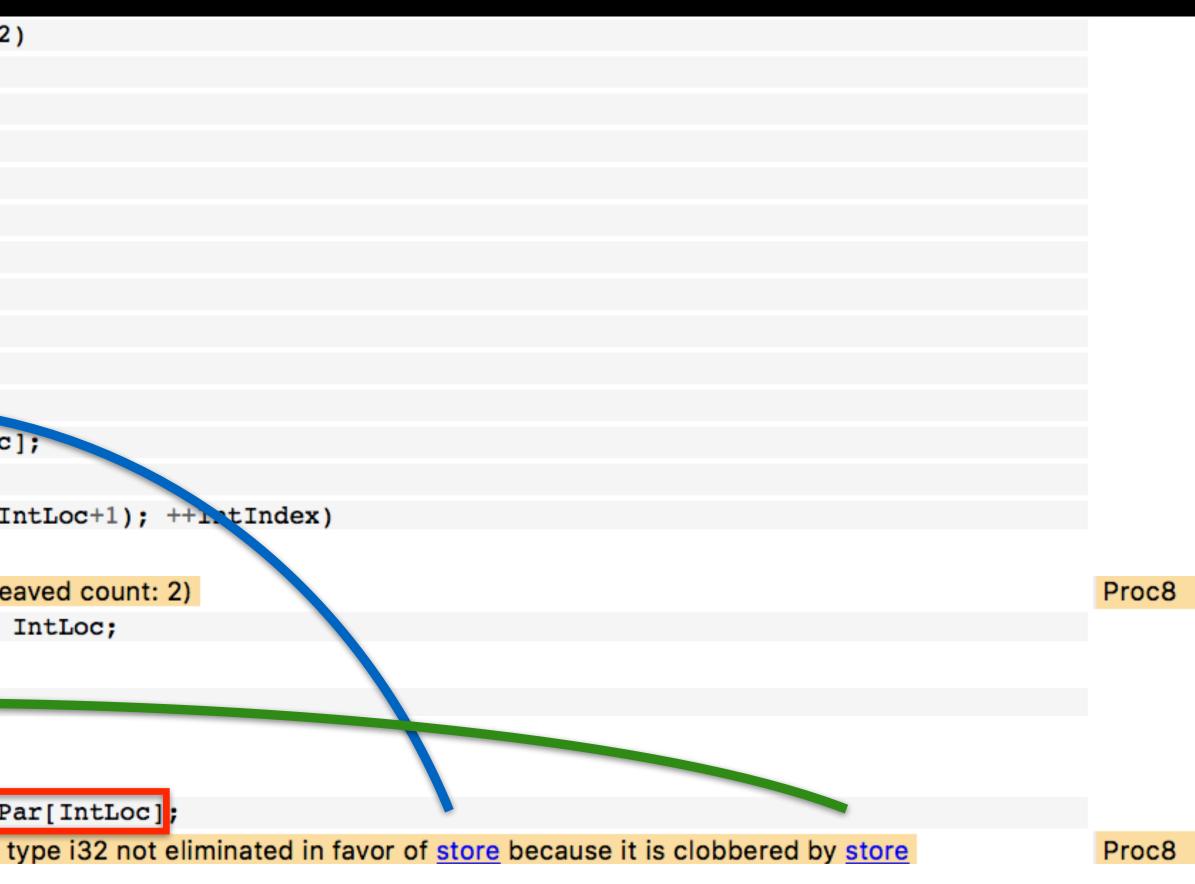
Proc8

Par[IntLoc]; type i32 not eliminated in favor of <u>store</u> because it is clobbered by <u>store</u> Proc8



430			Proc8(Array)	lPar, Array2	Par, IntParIl	, IntParI2
431		Array1Dim	Array1P	ar;		
432		Array2Dim	Array2P	ar;		
433			OneToFifty	IntParI	1;	
434			OneToFifty	IntParI	2;	
435			{			
436			REG	OneToFifty	IntLoc;	
437			REG	OneToFifty	<pre>IntIndex;</pre>	
438						
439			IntI	Loc = IntPar	I1 + 5;	
440			Arra	ay1Par[IntLo	c] = IntParI2	;
441			Arra	ay1Par[IntLo	c+1] = Array1	Par[IntLoc
442			Arra	aylPar[IntLo	c+30] = IntLo	с;
443			for	(IntIndex =	IntLoc; IntI	ndex <= (In
	100%	loop-vectorize	vec	torized loop (v	ectorization wid	th: 4, interlea
444				Array2P	ar[IntLoc][In	tIndex] = 1
			_			
445			+- A1	ray2Par[Int	Loc][IntLoc-1]
			_			
446			Arra	ay2Par[IntLo	c+20][IntLoc]	= Array1Pa
	34%	gvn				load of t
447			Int	lob = 5;		
448			}			







400			Drogg(Array1Dar Array2Dar Tr+DarT1 Tr+DarT2)
430			Proc8(Array1Par, Array2Par, IntParI1, IntParI2)
431			Array1Dim Array1Par;
432			Array2Dim Array2Par;
433			OneToFifty IntParI1;
434			OneToFifty IntParI2;
435			{
436			REG OneToFifty IntLoc;
437			REG OneToFifty IntIndex;
438			
439			<pre>IntLoc = IntParI1 + 5;</pre>
440			<pre>Array1Par[IntLoc] = IntParI2;</pre>
441			ArraylPar[IntLoc+1] = ArraylPar[IntLoc]
442			ArraylPar[IntLoc+30] = IntLoc;
443			<pre>for (IntIndex = IntLoc; IntIndex <= (Int</pre>
	52%	loop-delete	loop deleted
444			Array2Par[IntLoc][IntIndex] = In
	52%	loop-idiom	
445			++Array2Par[IntLoc][IntLoc-1];
	17%	gvn	
	17%	gvn	
446	1770	gvii	Array2Par[IntLoc+20][IntLoc] = Array1Par
440			mildy zi di [inchoo ; zo [[inchoo] – mildy ii di
	170/	C 1/10	load of tw
447	17%	gvn	
447			<pre>IntGlob = 5;</pre>
448			}

2)	
c];	
IntLoc+1); ++IntIndex)	
	Proc0
IntLoc;	
ormed memset	Proc0
re because it is clobbered by store	Proc0
re because it is clobbered by <u>call</u>	Proc0
Par[IntLoc];	
type i32 eliminated	Proc0



430			• =		Par, IntParI1,	IntParI2)
431			Array1Dim	Array1Pa	ar;	
432			Array2Dim	Array2Pa	ar;	
433			OneToFifty	IntParI	1;	
434			OneToFifty	IntParI	2;	
435			{			
436			REG	OneToFifty	IntLoc;	
437			REG	OneToFifty	IntIndex;	
438						
439			IntI	loc = IntParl	11 + 5;	
440			Arra	ylPar[IntLoc	c] = IntParI2;	
441			Arra	ylPar[IntLoc	c+1] = Array1Pa	ar[IntLoc];
442			Arra	ylPar[IntLoc	(+30] = IntLoc	;
443			for	(IntIndex =	IntLoc; IntInd	dex <= (Int
	52%	loop-delete	loop	deleted		
444				Array2Pa	ar[IntLoc][Int]	Index] = In
	52%	loop-idiom		_		form
445			++Ar	ray2Par[Int]	Loc][IntLoc-1]	
	17%	gvn			ot eliminated in fa	-
	17%	gvn			ot eliminated in fa	
446		5			c+20][IntLoc]	
	17%	gvn				load of typ
447	1770	gvii	Inte	lob = 5;		load of typ
448			1			
440			,			

2)	
c];	
IntLoc+1); ++IntIndex)	
	Proc0
IntLoc;	
ormed memset	Proc0
re because it is clobbered by store	Proc0
re because it is clobbered by <u>call</u>	Proc0
Par[IntLoc];	
type i32 eliminated	Proc0



400			Drogg(Array1Dar Array2Dar Tr+DarT1 Tr+DarT2)
430			Proc8(Array1Par, Array2Par, IntParI1, IntParI2)
431			Array1Dim Array1Par;
432			Array2Dim Array2Par;
433			OneToFifty IntParI1;
434			OneToFifty IntParI2;
435			{
436			REG OneToFifty IntLoc;
437			REG OneToFifty IntIndex;
438			
439			<pre>IntLoc = IntParI1 + 5;</pre>
440			<pre>Array1Par[IntLoc] = IntParI2;</pre>
441			ArraylPar[IntLoc+1] = ArraylPar[IntLoc]
442			ArraylPar[IntLoc+30] = IntLoc;
443			<pre>for (IntIndex = IntLoc; IntIndex <= (Int</pre>
	52%	loop-delete	loop deleted
444			Array2Par[IntLoc][IntIndex] = In
	52%	loop-idiom	
445			++Array2Par[IntLoc][IntLoc-1];
	17%	gvn	
	17%	gvn	
446	1770	gvii	Array2Par[IntLoc+20][IntLoc] = Array1Par
440			hildy 21 di [inchoo ; 20] [inchoo] — hildy il di
	170/	CU 10	load of tw
447	17%	gvn	
447			<pre>IntGlob = 5;</pre>
448			}

;	
tLoc+1); ++IntIndex)	
	Proc0
ntLoc;	
med memset	Proc0
because it is clobbered by store	Proc0
because it is clobbered by <u>call</u>	Proc0
r[IntLoc];	

type i32 eliminated Proc0



278			<pre>while (IntLoc1 < IntLoc2)</pre>	
	34%	loop-unroll	completely unrolled loop with 2 iterations	Proc0
279			{	
280			<pre>IntLoc3 = 5 * IntLoc1 - IntLoc2;</pre>	
281			<pre>Proc7(IntLoc1, IntLoc2, &IntLoc3);</pre>	
	18%	inline	Proc7 can be inlined into Proc0 with cost=-5 (threshold=337)	Proc0
	18%	inline	Proc7 inlined into Proc0	Proc0
282			++IntLoc1;	
283			}	
284			<pre>Proc8(Array1Glob, Array2Glob, IntLoc1, IntLoc3);</pre>	
	18%	inline	Proc8 can be inlined into Proc0 with cost=125 (threshold=225)	Proc0
	18%	inline	Proc8 inlined into Proc0	Proc0
285			<pre>Proc1(PtrGlb);</pre>	
	18%	inline	Proc1 can be inlined into Proc0 with cost=15 (threshold=337)	Proc0
	18%	inline	Proc1 inlined into Proc0	Proc0
286			<pre>for (CharIndex = 'A'; CharIndex <= Char2Glob; ++CharIndex)</pre>	
	34%	licm	load hoisted	Proc0
	17%	gvn	load of type i8 not eliminated in favor of store because it is clobbered by store	Proc0
	11%	gvn	load of type i8 eliminated	Proc0



430			<pre>Proc8(Array1Par, Array2Par, IntParI1, IntParI2)</pre>
431			Array1Dim Array1Par;
432			Array2Dim Array2Par;
433			OneToFifty IntParI1;
434			OneToFifty IntParI2;
435			{
436			REG OneToFifty IntLoc;
437			REG OneToFifty IntIndex;
438			
439			<pre>IntLoc = IntParI1 + 5;</pre>
440			<pre>Array1Par[IntLoc] = IntParI2;</pre>
441			Array1Par[IntLoc+1] = Array1Par[IntLoc];
442			<pre>Array1Par[IntLoc+30] = IntLoc;</pre>
443			<pre>for (IntIndex = IntLoc; IntIndex <= (Int</pre>
	52%	loop-delete	loop deleted
444			Array2Par[IntLoc][IntIndex] = In
	52%	loop-idiom	form
445			++Array2Par[IntLoc][IntLoc-1];
	17%	gvn	load of type i32 not eliminated in favor of store
	17%	gvn	load of type i32 not eliminated in favor of store
446			Array2Par[IntLoc+20][IntLoc] = Array1Par
	17%	gvn	load of ty
447			<pre>IntGlob = 5;</pre>
448			}

2)	
c];	
IntLoc+1); ++IntIndex)	
	Proc0
IntLoc;	
ormed memset	Proc0
re because it is clobbered by <u>store</u>	Proc0
re because it is clobbered by <u>call</u>	Proc0
Par[IntLoc];	
type i32 eliminated	Proc0



DhryStone: Summary

- Without low-level debugging, quickly reconstructed what happened
- Even though it involved interaction between multiple optimizations Inlining and Alias Analysis/GVN
- Missed optimizations: Extra analysis to manage with false positives
 - 1. Filter trivially false positives
 - 2. Expose enough information for quick detection by user

Freebench/distray (MultiSource/Benchmarks)

Finding Performance Opportunity

211			sta	atic double IntersectObjs(VECTOR *LinP, VECTOR *Lin
212				VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{	
237				<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
207	99%	loop-vectorize		loop not vectorized: vectorization is not beneficial and is not
238	00/0			Pos = objs[objn].pos;
239				Pos.x -= LinP->x; /* Translate
200	100%	licm		failed to hoist load with loop-invarian
240				Pos.y -= LinP->y;
	100%	licm		failed to hoist load with loop-invarian
241				Pos.z -= LinP->z;
	100%	licm		failed to hoist load with loop-invarian
	20%	gvn		getelementptr eliminated by PRE
242				A = 1.0 / (LinD - x + LinD - y
	100%	licm		failed to hoist load with loop-invari
	100%	licm		failed to hois
	100%	licm		
	100%	gvn		
	100%	gvn		
243				B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*Lind)
244				C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -
245				if($(A = C + B*B) > 0.0$) { /*else 1
246				A = sqrt(A);
	9%	inline		sqrt will not be inlined into IntersectObjs beca
247				if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248				if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249				t = ttmp;
250				Pnt->x = LinD->x*t; /* Calculate
	5%	slp-vectorizer		Stores SLP vectorized
251				Pnt->y = LinD->y*t;
	5%	licm		failed to hoist load with loo
	5%	gvn		load of type double elimina
252				Pnt->z = LinD->z*t;
	5%	licm		failed to hoist load with loo
	5%	gvn		load of type double elimina
	100%	gvn		load of type double not elin
	100%	gvn		load of type double not elin
253				Norm->x = Pnt->x-Pos.x; /* Calcualat
	5%	licm		failed to hoist load with loo
	5%	gvn		load of type double elimina
254				$Norm - > v = Pnt - > v - Pos \cdot v;$

```
InD,
TURE **txt )
```

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

riant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

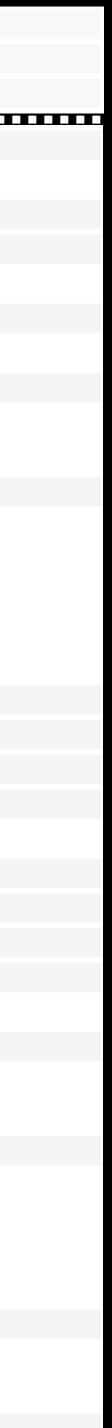
failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of load because it is clobbered by store load of type double not eliminated in favor of load because it is clobbered by store inD->z) * A; - Pos.y*Pos.y - Pos.z*Pos.z) * A; no hit */

```
cause its definition is unavailable
A;
```

```
e intersection point */
```

op-invariant address because the loop may invalidate its value ated in favor of <u>load</u>

```
op-invariant address because the loop may invalidate its value
ated in favor of load
iminated in favor of load because it is clobbered by <u>store</u>
iminated in favor of <u>load</u> because it is clobbered by <u>store</u>
ate surface normal */
op-invariant address because the loop may invalidate its value
ated in favor of <u>fmul</u>
```



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			Pos = objs[objn].pos;
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z -= LinP->z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRE
242			A = 1.0 / (LinD - x + LinD - y + LinD + LinD - y + LinD - y + LinD - y + Li
	100%	licm	failed to hoist load with loop-invari
	100%	licm	failed to hois
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*Lind - >y + Pos.z
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -
245			if($(A = C + B*B) > 0.0$) { /*else 1
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			<pre>t = ttmp;</pre>
250			$Pnt \rightarrow x = LinD \rightarrow x*t;$ /* Calculate
0.54	5%	slp-vectorizer	Stores SLP vectorized
251	= 0 /		Pnt->y = LinD->y*t;
	5%	licm	failed to hoist load with loo
050	5%	gvn	load of type double elimina
252	50/	liana	Pnt - z = LinD - z * t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
	100%	gvn	load of type double not elin
050	100%	gvn	load of type double not elin
253	5 %	lions	Norm->x = Pnt->x-Pos.x; /* Calcualate
	5% 5%	licm	failed to hoist load with loo
054	5%	gvn	load of type double elimina
254			$Norm - v = Pnt - v - Pos \cdot v;$

```
InD,
TURE **txt )
```

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

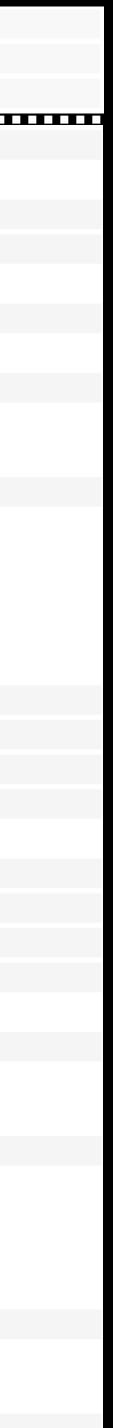
failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of load because it is clobbered by store load of type double not eliminated in favor of load because it is clobbered by store inD->z) * A; - Pos.y*Pos.y - Pos.z*Pos.z) * A; no hit */

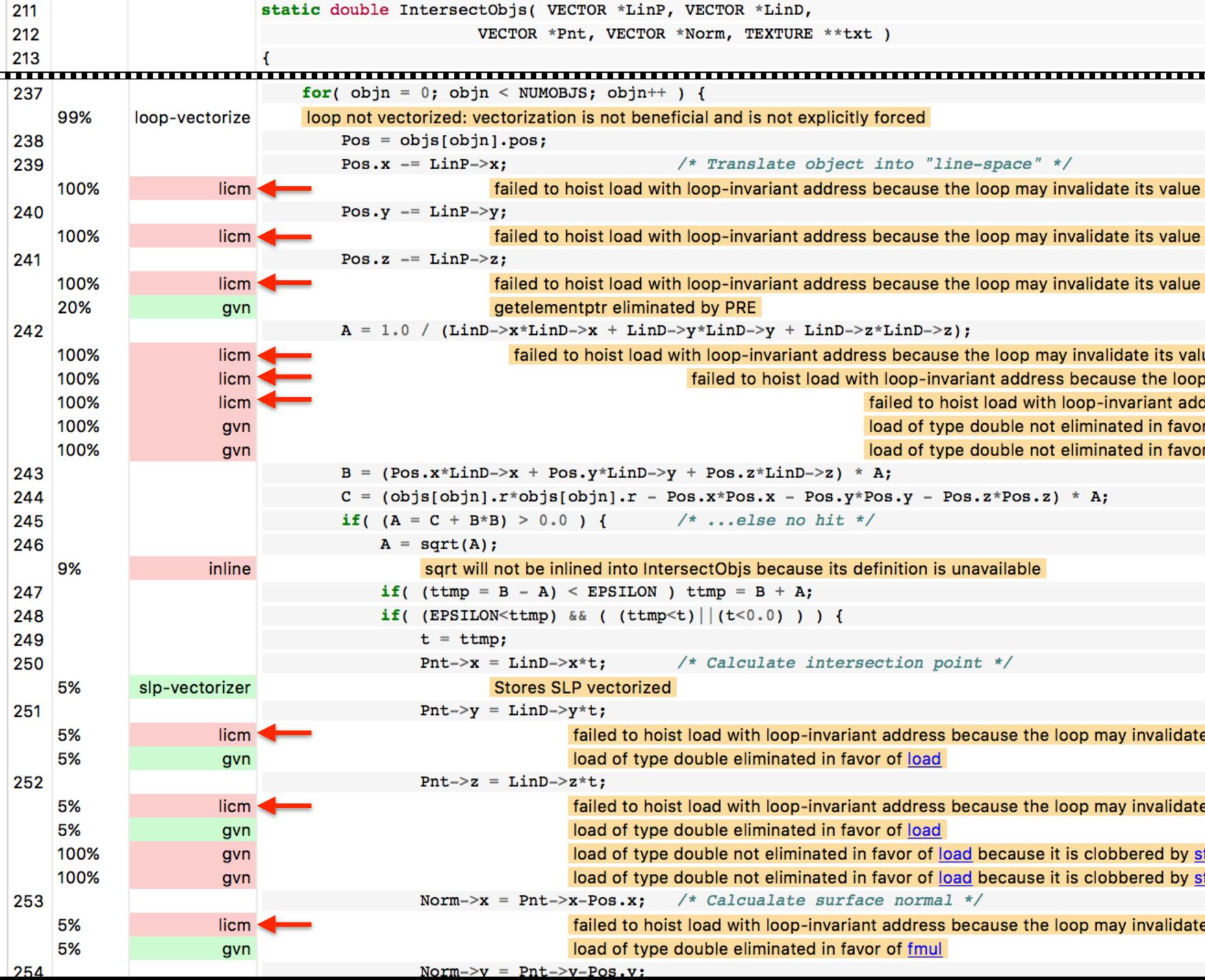
```
cause its definition is unavailable
+ A;
) ) ) {
```

```
te intersection point */
```

op-invariant address because the loop may invalidate its value nated in favor of load

bop-invariant address because the loop may invalidate its value nated in favor of load iminated in favor of load because it is clobbered by store iminated in favor of load because it is clobbered by store ate surface normal */ bop-invariant address because the loop may invalidate its value nated in favor of fmul





```
/* Translate object into "line-space" */
```

failed to hoist load with loop-invariant address because the loop may invalidate its value

failed to hoist load with loop-invariant address because the loop may invalidate its value

failed to hoist load with loop-invariant address because the loop may invalidate its value

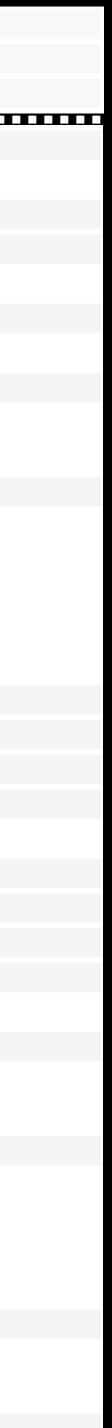
failed to hoist load with loop-invariant address because the loop may invalidate its value

failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of load because it is clobbered by store load of type double not eliminated in favor of load because it is clobbered by store

```
/* Calculate intersection point */
```

failed to hoist load with loop-invariant address because the loop may invalidate its value

```
failed to hoist load with loop-invariant address because the loop may invalidate its value
load of type double not eliminated in favor of load because it is clobbered by store
load of type double not eliminated in favor of load because it is clobbered by store
failed to hoist load with loop-invariant address because the loop may invalidate its value
```



212		
		VECTOR *Pnt, VECTOR *Norm, TEXT
213	{	
237	f	<pre>or(objn = 0; objn < NUMOBJS; objn++) {</pre>
99% loop-ved		oop not vectorized: vectorization is not beneficial and is no
238		Pos = objs[objn].pos;
239		Pos.x -= LinP->x; /* Translat
100%	licm 🔶	failed to hoist load with loop-invaria
240		Pos.y -= LinP->y;
100%	licm 🚽 🗕	failed to hoist load
241		Pos.z -= LinP->z;
100%	licm 🔶	failed to hoist load mayb
20%	gvn	getelementptr elimi
242		$A = 1.0 / (LinD \rightarrow x + LinD \rightarrow x + Li$
100%	licm	failed to hoist load with loop-inva
100%	licm	failed to ho
100%	licm	
100%	gvn	
100%	gvn	
243		B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*LinD - >y + Pos.z
244		<pre>C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x</pre>
245		if($(A = C + B*B) > 0.0$) { /*else
246		A = sqrt(A);
9%	inline	sqrt will not be inlined into IntersectObjs bed
247		if($(ttmp = B - A) < EPSILON$) $ttmp = B + A$
248		if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249		t = ttmp;
250		Pnt->x = LinD->x*t; /* Calculat
5% slp-vec	torizer	Stores SLP vectorized
251	liam	Pnt -> y = LinD -> y * t;
5% 5%	licm	failed to hoist load with load
252	gvn	<pre>load of type double elimin Pnt->z = LinD->z*t;</pre>
5%	licm	failed to hoist load with log
5%	gvn	load of type double elimin
100%	gvn	load of type double entitie
100%	gvn	load of type double not eli
253	911	Norm->x = Pnt->x-Pos.x; /* Calcuala
5%	licm	failed to hoist load with log
5%	gvn	load of type double elimin
254	3	Norm->v = $Pnt->v-Pos.v$;

```
InD,
TURE **txt )
```

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

```
ot modified via LinP,
be writes through other
pointers
```

value

value

riant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of <u>load</u> because it is clobbered by <u>store</u> load of type double not eliminated in favor of <u>load</u> because it is clobbered by <u>store</u> LnD->z) * A;

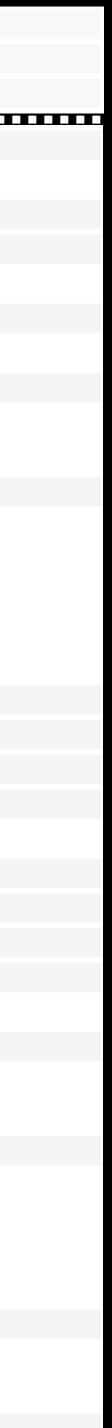
```
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

```
cause its definition is unavailable
A;
```

```
e intersection point */
```

op-invariant address because the loop may invalidate its value ated in favor of <u>load</u>

```
op-invariant address because the loop may invalidate its value
ated in favor of load
iminated in favor of load because it is clobbered by store
iminated in favor of load because it is clobbered by store
ate surface normal */
op-invariant address because the loop may invalidate its value
ated in favor of fmul
```



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *LinD,</pre>			
212			VECTOR *Pnt, VECTOR *Norm, TEXTURE **txt)			
213			{			
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>			
237	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not explicitly forced			
238	55%	loop-vectorize	Pos = objs[objn].pos;			
239			Pos.x -= LinP->x; /* Translate object into "line-space" */			
200	100%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
240	10070	nom	Pos.y $-=$ LinP->y;			
	100%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
241			Pos.z $-=$ LinP $->z$;			
	100%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	20%	gvn	getelementptr eliminated by PRE			
242			A = 1.0 / (LinD->x*LinD->x + LinD->y*LinD->y + LinD->z*LinD->z);			
	100%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	100%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	100%	licm	faile Not modified via LinD, y invalidate its value			
	100%	gvn	load			
	100%	gvn	load maybe writes through other bered by store			
243			B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LinD->z) * A; $C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - Pos.y*Pos$ $DOINTERS$			
244						
245			<pre>if((A = C + B*B) > 0.0) { /*else no hit */</pre>			
246	00/	in the s	A = sqrt(A);			
	9%	inline	sqrt will not be inlined into IntersectObjs because its definition is unavailable			
247 248			<pre>if((ttmp = B - A) < EPSILON) ttmp = B + A; if((EPSILON<ttmp) &&="" (="" (t<0.0)="" (ttmp<t)="")="" pre="" {<="" =""></ttmp)></pre>			
248			t = ttmp;			
249			Pnt->x = LinD->x*t; /* Calculate intersection point */			
200	5%	slp-vectorizer	Stores SLP vectorized			
251	070		Pnt->y = LinD->y*t;			
	5%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	5%	gvn	load of type double eliminated in favor of load			
252		J	Pnt->z = LinD->z*t;			
	5%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	5%	gvn	load of type double eliminated in favor of load			
	100%	gvn	load of type double not eliminated in favor of load because it is clobbered by store			
	100%	gvn	load of type double not eliminated in favor of load because it is clobbered by store			
253			Norm->x = Pnt->x-Pos.x; /* Calcualate surface normal */			
	5%	licm	failed to hoist load with loop-invariant address because the loop may invalidate its value			
	5%	gvn	load of type double eliminated in favor of <u>fmul</u>			
254			Norm - v = Pnt - v - Pos v;			



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			Pos = objs[objn].pos;
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z -= LinP->z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRF
242			$\mathbf{A} = 1.0 / (\text{LinD} - \mathbf{x} + \text{LinD} - \mathbf{y} + \text{LinD} - \mathbf{y} + \text{LinD} - \mathbf{y} + $
	100%	licm	failed to hoist load with loop-invar
	100%	licm	failed to hoi
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x ·
245			if($(A = C + B*B) > 0.0$) { /*else
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			t = ttmp;
250			$Pnt \rightarrow x = LinD \rightarrow x*t;$ /* Calculate
	5%	slp-vectorizer	Stores SLP vectorized
251			Pnt->y = LinD->y*t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
	100%	gvn	load of type double not elir
0.50	100%	gvn	load of type double not elir
253	504		Norm->x = Pnt->x-Pos.x; /* Calcuala
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
254			$Norm - > v = Pnt - > v - Pos \cdot v;$

```
inD,
TURE **txt )
```

ot explicitly forced

```
te object into "line-space" */
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

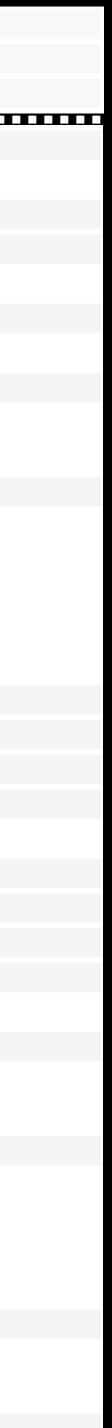
```
+ LinD->z*LinD->z);
ariant address because the loop may invalidate its value
bist load with loop-invariant address because the loop may invalidate its value
failed to hoist load with loop-invariant address because the loop may invalidate its value
load of type double not eliminated in favor of load because it is clobbered by store
load of type double not eliminated in favor of load because it is clobbered by store
load of type store
load of type store by store
store
inD->z) * A;
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

```
cause its definition is unavailable
+ A;
) ) ) {
```

```
te intersection point */
```

oop-invariant address because the loop may invalidate its value nated in favor of load

bop-invariant address because the loop may invalidate its value nated in favor of load liminated in favor of load because it is clobbered by store liminated in favor of load because it is clobbered by store ate surface normal */ bop-invariant address because the loop may invalidate its value nated in favor of fmul



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z -= LinP->z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRE
242			A = 1.0 / (LinD - x + LinD - y + LinD + LinD - y + LinD - y + LinD - y + Li
	100%	licm	failed to hoist load with loop-invar
	100%	licm	failed to hois
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*LinD - >y + Pos.z
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x ·
245			if($(A = C + B*B) > 0.0$) { /*else
246		1.11.1	A = sqrt(A);
0.47	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if((ttmp = B - A) < EPSILON) ttmp = B +
248 249			<pre>if((EPSILON<ttmp) &&="" (="" (ttmp<t)="" td="" ="" (t<0.0)<=""></ttmp)></pre>
249			Pnt - x = LinD - x t; / Calculate
200	5%	slp-vectorizer	Stores SLP vectorized
251	070	Sip-vectorizer	Pnt->y = LinD->y*t;
201	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
	100%	gvn	load of type double not elir
	100%	gvn	load of type double not elir
253			Norm->x = Pnt->x-Pos.x; /* Calcuala
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
254			Norm->v = Pnt->v-Pos.v:

```
inD,
TURE **txt )
```

Reads and writes don't alias

ot explicitly forced

```
te object into "line-space" */
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of load because it is clobbered by store load of type double not eliminated in favor of load because it is clobbered by store inD->z) * A; - Pos.y*Pos.y - Pos.z*Pos.z) * A;

```
no hit */
```

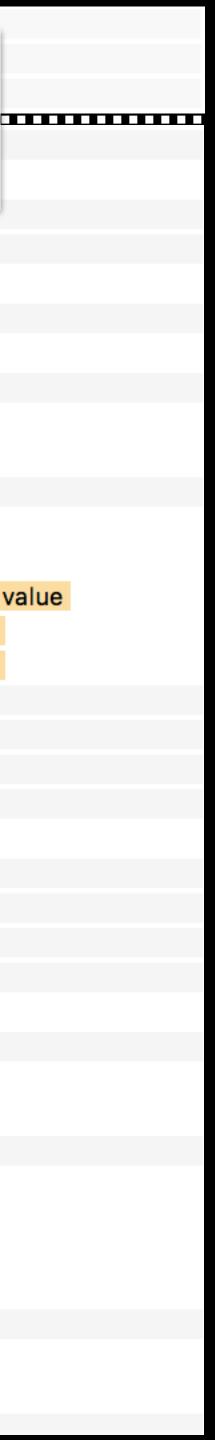
```
cause its definition is unavailable
+ A;
) ) ) {
```

```
te intersection point */
```

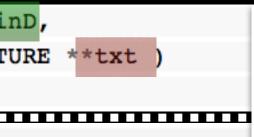
op-invariant address because the loop may invalidate its value nated in favor of <u>load</u>

hop-invariant address because the loop may invalidate its value hated in favor of load iminated in favor of load because it is clobbered by <u>store</u> iminated in favor of load because it is clobbered by <u>store</u> ate surface normal */

```
ated in favor of <u>fmul</u>
```



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z -= LinP->z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRE
242			A = 1.0 / (LinD - x + LinD - y + LinD + LinD - y + LinD - y + LinD - y + Li
	100%	licm	failed to hoist load with loop-invar
	100%	licm	failed to hois
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*LinD - >y + Pos.z
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x ·
245			if($(A = C + B*B) > 0.0$) { /*else
246		1.11.1	A = sqrt(A);
0.47	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if((ttmp = B - A) < EPSILON) ttmp = B +
248 249			<pre>if((EPSILON<ttmp) &&="" (="" (ttmp<t)="" td="" ="" (t<0.0)<=""></ttmp)></pre>
249			Pnt - x = LinD - x t; / Calculate
200	5%	slp-vectorizer	Stores SLP vectorized
251	070	Sip-vectorizer	Pnt->y = LinD->y*t;
201	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
	100%	gvn	load of type double not elir
	100%	gvn	load of type double not elir
253			Norm->x = Pnt->x-Pos.x; /* Calcuala
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimina
254			Norm->v = Pnt->v-Pos.v:



Loop versioning with array overlap checks?

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

nt address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

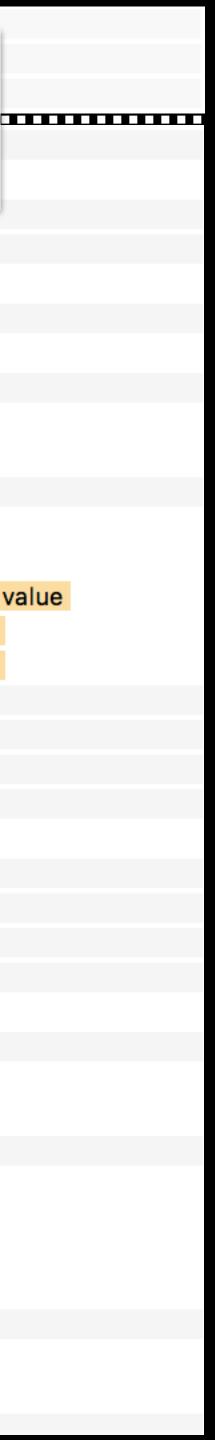
failed to hoist load with loop-invariant address because the loop may invalidate its value load of type double not eliminated in favor of load because it is clobbered by store load of type double not eliminated in favor of load because it is clobbered by store inD->z) * A; - Pos.y*Pos.y - Pos.z*Pos.z) * A; no hit */

```
cause its definition is unavailable
+ A;
) ) ) {
```

```
te intersection point */
```

op-invariant address because the loop may invalidate its value ated in favor of <u>load</u>

bop-invariant address because the loop may invalidate its value nated in favor of load iminated in favor of load because it is clobbered by store iminated in favor of load because it is clobbered by store ate surface normal */ bop-invariant address because the loop may invalidate its value nated in favor of fmul



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	version-licm	Loop memory access not suitable
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238		-	<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z = LinP > z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRE
242			$A = 1.0 / (LinD \rightarrow x + LinD \rightarrow y $
	100%	licm	failed to hoist load with loop-invari
	100%	licm	failed to hois
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lindar)
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -
245			if($(A = C + B*B) > 0.0$) { /*else 1
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			t = ttmp;
250	50/		$Pnt \rightarrow x = LinD \rightarrow x*t;$ /* Calculate
051	5%	slp-vectorizer	Stores SLP vectorized
251	E9/	liana	Pnt - y = LinD - y t;
	5%	licm	failed to hoist load with loo
252	5%	gvn	load of type double elimina Pnt->z = LinD->z*t;
252	E9/	liam	
	5%	licm	failed to hoist load with loo load of type double elimina
	5% 100%	gvn	
	100% 100%	gvn	load of type double not elin load of type double not elin
253	100%	gvn	Norm->x = Pnt->x-Pos.x; /* Calcualat
200	5%	licm	failed to hoist load with loo
	5% 5%		
	570	gvn	

```
inD,
TURE **txt )
```

LICM-based LoopVersioning (-enable-loop-versioning-licm)

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

```
failed to hoist load with loop-invariant address because the loop may invalidate its value
load of type double not eliminated in favor of load because it is clobbered by store
load of type double not eliminated in favor of load because it is clobbered by store
inD->z) * A;
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

cause its definition is unavailable + A;

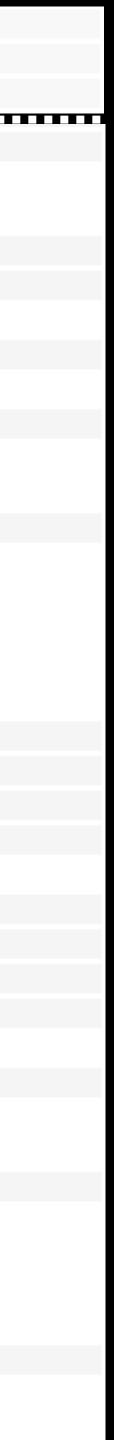
)){

```
te intersection point */
```

op-invariant address because the loop may invalidate its value nated in favor of <u>load</u>

```
op-invariant address because the loop may invalidate its value
nated in favor of <u>load</u>
iminated in favor of <u>load</u> because it is clobbered by <u>store</u>
iminated in favor of <u>load</u> because it is clobbered by <u>store</u>
ate surface normal */
```

nated in favor of fmul



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *LinD,</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTURE **txt)
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	version-licm	Loop memory access not suitable
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not explicitly forced
238			Pos = objs[objn].pos;
239			Pos.x -= LinP->x; /* Translate object into "lin
	100%	licm	failed to hoist load with loop-invariant address because th
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invariant address because th
241			Pos.z -= LinP->z;
	100%	licm	failed to hoist load with loop-invariant address because th
	20%	gvn	getelementptr eliminated by PRE
242			A = 1.0 / (LinD - x + LinD - y + LinD - y + LinD - z
	100%	licm	failed to hoist load with loop-invariant address because
	100%	licm	failed to hoist load with loop-inva
	100%	licm	failed to ho
	100%	gvn	load of type
	100%	gvn	load of type
243			B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LinD->z) * A;
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - Pos.y*Pos.y - Po
245			<pre>if((A = C + B*B) > 0.0) { /*else no hit */</pre>
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs because its definition is u
247			if $(ttmp = B - A) < EPSILON)$ ttmp = B + A;
248			if((EPSILON <ttmp) &&="" (="" (t<0.0)="" (ttmp<t)="")="" td="" {<="" =""></ttmp)>
249			<pre>t = ttmp; Pnt->x = LinD->x*t; /* Calculate intersection point</pre>
250	E9/	ala vectorizar	
251	5%	slp-vectorizer	<pre>Stores SLP vectorized Pnt->y = LinD->y*t;</pre>
201	5%	licm	failed to hoist load with loop-invariant address b
	5%	gvn	load of type double eliminated in favor of load
252	070	gvii	Pnt->z = LinD->z*t;
202	5%	licm	failed to hoist load with loop-invariant address b
	5%	gvn	load of type double eliminated in favor of load
	100%	gvn	load of type double not eliminated in favor of loa
	100%	gvn	load of type double not eliminated in favor of loa
253		J	Norm->x = Pnt->x-Pos.x; /* Calcualate surface normal
	5%	licm	failed to hoist load with loop-invariant address b
	5%	gvn	load of type double eliminated in favor of fmul

```
inD,
FURE **txt )
.....
```

Performance opportunity if we can improve this pass

ot explicitly forced

```
te object into "line-space" */
ant address because the loop may invalidate its value
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

ist load with loop-invariant address because the loop may invalidate its value

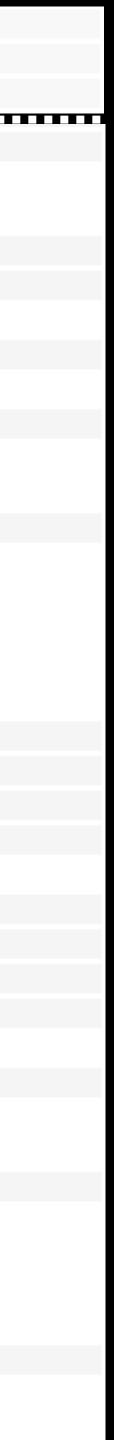
```
failed to hoist load with loop-invariant address because the loop may invalidate its value
          load of type double not eliminated in favor of load because it is clobbered by store
          load of type double not eliminated in favor of load because it is clobbered by store
inD->z) * A;
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

cause its definition is unavailable · A;

te intersection point */

op-invariant address because the loop may invalidate its value ated in favor of load

op-invariant address because the loop may invalidate its value ated in favor of load iminated in favor of <u>load</u> because it is clobbered by <u>store</u> iminated in favor of load because it is clobbered by store ate surface normal */ op-invariant address because the loop may invalidate its value



211			<pre>static double IntersectObjs(VECTOR *LinP, VECTOR *Lin</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	version-licm	Loop memory access not suitable
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238		-	<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	failed to hoist load with loop-invarian
240			Pos.y -= LinP->y;
	100%	licm	failed to hoist load with loop-invarian
241			Pos.z = LinP > z;
	100%	licm	failed to hoist load with loop-invarian
	20%	gvn	getelementptr eliminated by PRE
242			A = 1.0 / (LinD - x + LinD - y
	100%	licm	failed to hoist load with loop-invari
	100%	licm	failed to hois
	100%	licm	
	100%	gvn	
	100%	gvn	
243			B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lindar)
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -
245			if($(A = C + B*B) > 0.0$) { /*else 1
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			t = ttmp;
250	50/		$Pnt \rightarrow x = LinD \rightarrow x*t;$ /* Calculate
051	5%	slp-vectorizer	Stores SLP vectorized
251	E9/	liana	Pnt - y = LinD - y t;
	5%	licm	failed to hoist load with loo
252	5%	gvn	load of type double elimina Pnt->z = LinD->z*t;
252	E9/	liam	
	5%	licm	failed to hoist load with loo load of type double elimina
	5% 100%	gvn	
	100% 100%	gvn	load of type double not elin load of type double not elin
253	100%	gvn	Norm->x = Pnt->x-Pos.x; /* Calcualat
205	5%	licm	failed to hoist load with loo
	5% 5%		
	570	gvn	

```
inD,
TURE **txt )
ot explicitly forced
```

Approximate the opportunity by manually modifying the source

```
te object into "line-space" */
```

```
ant address because the loop may invalidate its value
```

ant address because the loop may invalidate its value

ant address because the loop may invalidate its value

```
+ LinD->z*LinD->z);
```

ariant address because the loop may invalidate its value

```
ist load with loop-invariant address because the loop may invalidate its value
```

```
failed to hoist load with loop-invariant address because the loop may invalidate its value
load of type double not eliminated in favor of load because it is clobbered by store
load of type double not eliminated in favor of load because it is clobbered by store
inD->z) * A;
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

```
cause its definition is unavailable
+ A;
```

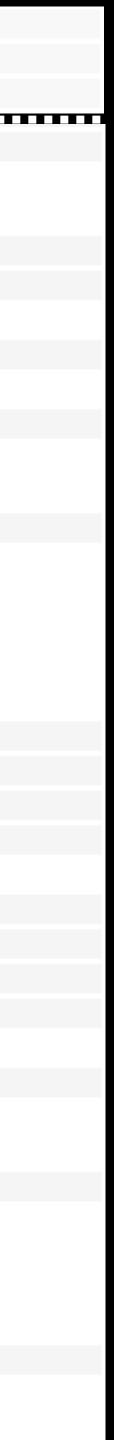
)){

```
te intersection point */
```

op-invariant address because the loop may invalidate its value nated in favor of <u>load</u>

```
oop-invariant address because the loop may invalidate its value
nated in favor of <u>load</u>
iminated in favor of <u>load</u> because it is clobbered by <u>store</u>
iminated in favor of <u>load</u> because it is clobbered by <u>store</u>
ate surface normal */
```

```
nated in favor of fmul
```



211			<pre>static double IntersectObjs(VECTOR * restrict LinP, V</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	load hoisted
240			Pos.y -= LinP->y;
	100%	licm	load hoisted
241			Pos.z -= LinP->z;
	100%	licm	load hoisted
	20%	gvn	load eliminated by PRE
	20%	gvn	getelementptr eliminated by PRE
242			$A = 1.0 / (LinD \rightarrow x + LinD \rightarrow y $
	100%	licm	load hoisted
	100%	licm	fmul hoisted
	100%	licm	load hoisted
	100%	licm	fmul hoiste
	100%	licm	fadd hoisted
	100%	licm	
	100%	licm	
	100%	licm	
	100%	licm	fdiv hoisted
	20%	gvn	
243			B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*Lind)
244			C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -
245			if($(A = C + B*B) > 0.0$) { /*else n
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if((ttmp = $B - A$) < EPSILON) ttmp = $B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			t = ttmp;
250			Pnt->x = LinD->x*t; /* Calculate
251			Pnt->y = LinD->y*t;
	5%	licm	failed to hoist load with loop
	5%	gvn	load of type double eliminat
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loop
	5%	gvn	load of type double eliminat
253			Norm->x = Pnt->x-Pos.x; /* Calcualat

```
VECTOR * restrict LinD,
TURE **txt )
```

ot explicitly forced

te object into "line-space" */

+ LinD->z*LinD->z);

d ed

> load hoisted fmul hoisted

fadd hoisted

load of type double eliminated in favor of load

```
inD->z) * A;
- Pos.y*Pos.y - Pos.z*Pos.z) * A;
no hit */
```

cause its definition is unavailable
A;
)) {

te intersection point */

op-invariant address because load is conditionally executed ated in favor of phi

```
op-invariant address because load is conditionally executed
ated in favor of <u>phi</u>
ate surface normal */
```

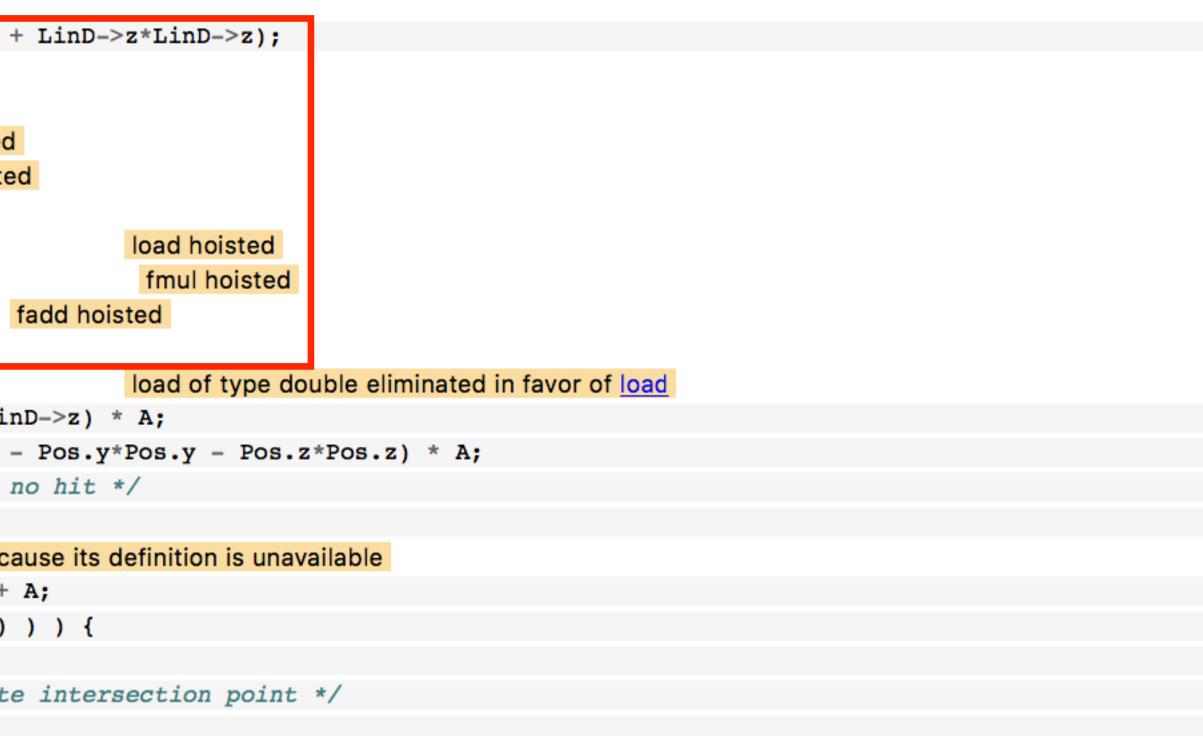


212VECTOR *Pnt, VECTOR *Norm, TEXTUR213for(objn = 0; objn < NUMOBJS; objn++) {239for(objn = 0; objn < NUMOBJS; objn++) {239loop-vectorize239Pos = objs[objn].pos;239Pos.x == LinP-x; /* Translate100%licm100%licm240Pos.y == LinP-y;100%licm20%gvn2100%licm100%lic	211			<pre>static double IntersectObjs(VECTOR * restrict LinP, V</pre>
237for(objn = 0; objn < NUMOBJS; objn++) {238loop-vectorize239 $for(objn = 0; objn < NUMOBJS; objn++) $ {239 $Pos = objs[objn].pos;$ 239 $Pos = objs[objn].pos;$ 239 $Pos * - \pm LinP > x;$ 100%licm240 $Pos * y = LinP > y;$ 100%licm20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn20%gvn100%licm	212			VECTOR *Pnt, VECTOR *Norm, TEXTU
99%loop-vectorizeloop not vectorized: vectorization is not beneficial and is not Pos = objs[objn].pos; Pos.x == LinP>x; Pos.x == LinP>x; (* Translate load hoisted240Pos.y == LinP>y; loo%load hoisted241Pos.z == LinP>z; load hoistedload hoisted20%gyn gyn load hoistedload hoisted20%gyn gyn load hoistedload hoisted20%gyn gyn load hoistedload hoisted20%gyn gyn load hoistedload hoisted20%gyn gyn load hoistedload hoisted20%gyn gynload hoisted100% licm 100%licm licmload hoisted100% licm 100% licmload hoistedload hoisted100% licm 100% licmload hoistedload hoisted100% licm 100% licmload hoistedload hoisted100% licm 100% licmload hoistedload hoisted100% licm 100% licmload hoistedload hoisted100% licm 100% licm 100% licm licm 100% licm licmfdiv hoisted243B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin C = (objs[objn].r*objs[objn].r = Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r A = sgrt(A);9% 244inline sgrt will not be inlined into IntersectObjs becc a = sgrt(A);9% 250inline failed to hoist load with loo load of type double elimina251 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5% 5	213			{
99%loop-vectorizeloop not vectorized: vectorization is not beneficial and is not Pos = objs[objn].pos; Pos.x = LinP>x;/* Translate load hoisted238100%licmload hoisted240Pos.y = LinP>y; loo%load hoisted241Pos.z = LinP>z; load hoisted20%gyn gynload hoisted20%gyn gynload hoisted20%gyn gynload hoisted20%gyn gynload hoisted20%gyn gynload hoisted20%gyn gynload hoisted100%licm licmload hoisted100%licm licmfmul hoisted100%licm licmload hoisted100%licm licmfdiv hoisted100%licm licmfdiv hoisted100%licm licmfdiv hoisted100%licm licmfdiv hoisted20%gynsqt will not be inlined into IntersectObjs becc20%gynsqt will not be inlined into IntersectObjs becc20%gynfif((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON+ttmp) && (ttmp <t) (t<0.0)<="" td="" ="">247if((EPSILON+ttmp) && (ttmp<t) (t<0.0)<="" td="" ="">248failed to hoist load with loo load of type double elimina250Pnt->z = LinD->z*t;5%licm failed to hoist load with loo load of type double elimina5%licm failed to hoist load with loo load of type double elimina5%gynlicem failed to hoist load with loo load of type do</t)></t)>	237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
238Pos = objs[objn].pos; Pos.x -= LinP->x;/* Translate Pos.y = LinP->x;100%licmload hoisted240Pos.y = LinP->y;load hoisted241Pos.z == LinP->z;load hoisted20%gynload hoisted20%gynload hoisted20%gynload hoisted20%gynload hoisted20%gynload hoisted20%gynload hoisted100%licmload hoisted100%licmfmul hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmload hoisted100%licmfadd hoisted20%gvnlice fadd hoisted20%gvnfallel hoisted20%gvnlice fadd hoisted100%licmfallel hoisted20%gvnlice fadd hoisted20%gvnlice hoisted20%gvnlice hoisted20%gvnlice hoisted24fallel hoistedlice hoisted <td>207</td> <td>99%</td> <td>loop-vectorize</td> <td></td>	207	99%	loop-vectorize	
239Pos.x -= LinP->x;/* Translate240Pos.y -= LinP->x;/* Translate241Pos.y -= LinP->z;100%licm20%gvn100%licm	238			
100%licmload hoisted240Pos.y -= LinP->y;100%licmload hoisted241Pos.z -= LinP->z;100%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn242A = 1.0 / (LinD->x*LinD->x + LinD->y+LinD->y +100%licm <td></td> <td></td> <td></td> <td></td>				
240Pos.y = LinP->y;100%licmload hoisted241Pos.z = LinP->z;100%gynload eliminated by PRE20%gyngyn20%gynload eliminated by PRE20%gynload hoisted100%licmload hoisted100%licmfmul hoisted100%licmload hoisted100%licmload hoisted100%licmload hoisted100%licmfadd hoisted20%gyngyn243B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -245if((A = C + B*B) > 0.0) { /*else r246if((EPSILON <ttmp) &="" ((ttmp<="" +="" <<="" a)="" b="" td="">247if((EPSILON<ttmp) &="" ((ttmp<t)="" (t<0.0)="" <="" td="" ="">249falled to hoist load with loo250Pnt->x = LinD->x*t; /* Calculate251gynload of type double elimina252Pnt->z = LinD->z*t;5%licmfailed to hoist load with loo</ttmp)></ttmp)>		100%	licm	•
241Pos.z -= LinP->z;100%licm20%gyn20%gyn20%gyn20%gyn20%gyn20%gyn242 $A = 1.0 / (LinD->x*LinD->x*LinD->y*LinD-$	240			
100%licmload hoisted20%gvngetelementpt eliminated by PRE20%gvngetelementpt eliminated by PRE20%gvnload hoisted20%gvnload hoisted100%licmfmul hoisted100%licmload hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfdiv hoisted100%licmfdiv hoisted100%licmfdiv hoisted100%licmfdiv hoisted100%licmfdiv hoisted20%gvnB = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted243fdiv hoistedfdiv hoisted244fdiv hoistedfdiv hoisted245ffit (A = C + B*B) > 0.0) { /*else r246fit ((EPSILON <ttmp) &="" ((tump<t))="" (t<0.0)="" *else="" r<="" td="" {="" ="">247fit ((EPSILON<ttmp) &="" ((tump<t))="" (t<0.0)="" t="ttmp;</td" {="" ="">250Pnt->x = LinD->x*t; /* Calculate251failed to hoist load with loo5%gvnload of type double elimina5%gvnload of type double elimina5%gvn</ttmp)></ttmp)>		100%	licm	load hoisted
20%gvnload eliminated by PRE20%gvngetelementpt eliminated by PRE242 $A = 1.0 / (LinD-x*LinD->x + LinD->y*LinD->y + 100%100%licmload hoisted100%licmfmul hoisted100%licmfmul hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted20%gvnfdiv hoisted243fdiv hoistedfdiv hoisted244fdiv hoistedfdiv hoisted245fdiv hoistedfdiv hoisted246fdiv hoisted into the inlined into IntersectObjs beca247fdiv (ttmp = B - A) < EPSILON) ttmp = B + fit$	241			
20%gvnload eliminated by PRE20%gvngetelementpt eliminated by PRE242 $A = 1.0 / (LinD->x*LinD->x + LinD->y*LinD-y*LinD-y+100%licmload hoisted100%licmfmul hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfdiv hoisted100%licmfdiv hoisted100%licmfdiv hoisted20%gvnB = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin24320%gvn244fif (A = C + B*B) > 0.0) { /*else r245if((A = C + B*B) > 0.0) { /*else r246if((trmp = B - A) < EPSILON) trmp = B +$		100%	licm	load hoisted
20%gvngetelementptreliminated by PRE242 $A = 1.0 / (LinD->x*LinD->x + LinD->y*LinD->y + 100%100%licm243B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r = Pos.x*Pos.x =245if((A = C + B*B) > 0.0) { /*else r246A = sqrt(A);9%inline247if((ttmp = B - A) < EPSILON) ttmp = B +$		20%	gvn	
242 $A = 1.0 / (LinD->x+LinD->x + LinD->y+LinD->y + linD->y+LinD->y + linD->y+LinD->y + linD->y+LinD->y + linD->y+LinD->y + linD->y+LinD->y + linD->y + linD->y+LinD->y + linD->y + l$		20%	-	
100%licmfmul hoisted100%licmload hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfdiv hoisted100%licmfdiv hoisted20%gvnfdiv hoisted20%gvnfdiv hoisted243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LinC = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);245A = sqrt(A);9%inline247if((tEPSILONt = tump;249T = tump;250Pnt->x = LinD->x*t; /* CalculatePnt->y = LinD->x*t;5%gvn252Pnt->z = LinD->z*t;5%licmSyn5%gvn252Pnt->z = LinD->z*t;5%gvn5%gvn$	242			A = 1.0 / (LinD - x + LinD - y
100%licmload hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfdiv hoisted100%licmfdiv hoisted20%gvnfdiv hoisted20%gvn $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin243B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r245A = sqrt(A);9%inline247sqrt will not be inlined into IntersectObjs beca248if((ttmp = B - A) < EPSILON) ttmp = B + if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON250Pnt->x = LinD->x*t; /* Calculated + Pnt->y = LinD->x*t;5%gvnload of type double elimina)252Pnt->z = LinD->z*t;5%licm5%gvn5%gvn$		100%	licm	load hoisted
100%licmload hoisted100%licmfadd hoisted100%licmfadd hoisted100%licmfdiv hoisted100%licmfdiv hoisted20%gvnfdiv hoisted20%gvn $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin243B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r245A = sqrt(A);9%inline247sqrt will not be inlined into IntersectObjs beca248if((ttmp = B - A) < EPSILON) ttmp = B + if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON250Pnt->x = LinD->x*t; /* Calculated + Pnt->y = LinD->x*t;5%gvnload of type double elimina)252Pnt->z = LinD->z*t;5%licm5%gvn5%gvn$		100%	licm	fmul hoisted
100%licmfadd hoisted100%licm100%licm100%licm100%licm100%gvn243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r245A = sqrt(A);9%inline247if((tmp = B - A) < EPSILON) ttmp = B + if((EPSILON249t = ttmp;250Pnt->x = LinD->x*t; /* Calculate251Pnt->y = LinD->y*t;5%licm5%gvn252Pnt->z = LinD->z*t;5%licm<$		100%	licm	
100%licm100%licm100%licm100%licm100%licm20%gvn243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r245if((A = C + B*B) > 0.0) { /*else r246A = sqrt(A);9%inline247if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON$		100%	licm	fmul hoiste
100%licm100%licm100%licm100%licm100%licm20%gvn243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin244C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r245if((A = C + B*B) > 0.0) { /*else r246A = sqrt(A);9%inline247if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON$		100%	licm	fadd hoisted
100% 100% 20%licm licmfdiv hoisted243 244 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LindrowC = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);2452469%inline247248249inline249250251inline5%5%5%licm9mt->z = LinD->z*t;5%5%5%9%licm9mt->z = LinD->z*t;5%5%9%licm9mt->z = LinD->z*t;$		100%	licm	
100% 20%licmfdiv hoisted243 244 244B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*Lin C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x - if((A = C + B*B) > 0.0) { /*else r A = sqrt(A);245 246 9%inline247 248 249sqrt will not be inlined into IntersectObjs beca if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON <ttmp) &&="" ((ttmp<t)="" (t<0.0)<br="" =""></ttmp)> 249 250 251250 251 5% 5% 5%licm 5% 9vn252 5% 5% 5% 6%licm 6 9vn253 5% 5% 9wnfailed to hoist load with loo load of type double elimination failed to hoist load with loo load of type double elimination		100%	licm	
20%gvn243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LinC = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);246A = sqrt(A);9%inline247if((ttmp = B - A) < EPSILON) ttmp = B +if((EPSILON249249t = ttmp;250Pnt->x = LinD->x*t; /* CalculatePnt->y = LinD->y*t;5%gvn251Fnt->z = LinD->z*t;5%licmS%5%gvn250Iicmfailed to hoist load with looload of type double elimina251Fnt->z = LinD->z*t;5%gvn$		100%	licm	
243 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LindC = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);2452469%inline247Sqrt will not be inlined into IntersectObjs becaif((ttmp = B - A) < EPSILON) ttmp = B +if((EPSILON249t = ttmp;250Pnt->x = LinD->x*t; /* CalculatePnt->y = LinD->y*t;5%gvn252Pnt->z = LinD->z*t;5%licm5%5%gvn250IntersectorDad of type double eliminaDad of type double elimina$		100%	licm	fdiv hoisted
243 244 245 245 $B = (Pos.x*LinD->x + Pos.y*LinD->y + Pos.z*LindC = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);2469%inline247248249sqrt will not be inlined into IntersectObjs becaif((ttmp = B - A) < EPSILON) ttmp = B +if((EPSILONx = LinD->x*t; /* CalculatePnt->y = LinD->y*t;5%5%5%5%5%Iicm5%9wn2525%5%5%Pnt->z = LinD->z*t;5%5%5%9wnIicmfailed to hoist load with loopload of type double eliminationload of type double elimination$		20%		
244 245 246 9% $C = (objs[objn].r*objs[objn].r - Pos.x*Pos.x -if((A = C + B*B) > 0.0) { /*else rA = sqrt(A);9%inline247248249if((ttmp = B - A) < EPSILON) ttmp = B +if((EPSILONx = LinD->x*t; /* CalculatePnt->y = LinD->y*t;5%licm5%5%Pnt->z = LinD->z*t;5%licmfailed to hoist load with looload of type double elimina5%gyn$	243		J • • •	B = (Pos.x*LinD - >x + Pos.y*LinD - >y + Pos.z*Lind)
246A = sqrt(A);9%inline247sqrt will not be inlined into IntersectObjs beca248if((ttmp = B - A) < EPSILON) ttmp = B +				
246A = sqrt(A);9%inline247sqrt will not be inlined into IntersectObjs beca248if((ttmp = B - A) < EPSILON) ttmp = B +	245			$if((A = C + B*B) > 0.0) \{$ /*else n
247 248 249if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON <ttmp) &&="" ((ttmp<t) (t<0.0)<br=""></ttmp)> t = ttmp;250 251 $T = ttmp;$ 250 251Pnt->x = LinD->x*t; /* Calculate Pnt->y = LinD->y*t;5% 5% 5% 5% 5%Icm failed to hoist load with loo load of type double eliminat failed to hoist load with loo load of type double eliminat failed to hoist load with loo load of type double eliminat				A = sqrt(A);
247 248 249if((ttmp = B - A) < EPSILON) ttmp = B + if((EPSILON <ttmp) &&="" ((ttmp<t) (t<0.0)<br=""></ttmp)> t = ttmp;250 251 $T = ttmp;$ 250 251Pnt->x = LinD->x*t; /* Calculate Pnt->y = LinD->y*t;5% 5% 5% 5% 5%Icm failed to hoist load with loo load of type double eliminat failed to hoist load with loo load of type double eliminat failed to hoist load with loo load of type double eliminat		9%	inline	sqrt will not be inlined into IntersectObjs beca
249 t = ttmp; 250 Pnt->x = LinD->x*t; /* Calculate 251 Pnt->y = LinD->y*t; 5% Icm 5% gvn 252 Pnt->z = LinD->z*t; 5% Icm 5% gvn 252 Pnt->z = LinD->z*t; 5% Icm 5% Icm 5% gvn	247			
250 251Pnt->x = LinD->x*t; /* Calculate Pnt->y = LinD->y*t;5%Iicm 5%5%gvn 2525%Iicm 5%5%Iicm 5%5%Iicm 5%5%Iicm 5%5%Iicm 65%Iicm 75%Iicm <td>248</td> <td></td> <td></td> <td>if((EPSILON<ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)></td>	248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
251Pnt->y = LinD->y*t;5%licm5%gvn252S%5%licm5%licm5%licm5%gvn5%gvn	249			<pre>t = ttmp;</pre>
5%licm5%gvn252T5%licm5%licm5%gvn5%gvn10ad of type double elimination5%gvn10ad of type double elimination5%gvn10ad of type double elimination5%gvn	250			$Pnt \rightarrow x = LinD \rightarrow x*t;$ /* Calculate
5%gvnload of type double elimination2525%licm5%licmfailed to hoist load with loop5%gvnload of type double elimination	251			Pnt->y = LinD->y*t;
5%gvnload of type double elimination2525%licm5%licmfailed to hoist load with loop5%gvnload of type double elimination		5%	licm	failed to hoist load with loo
252 Pnt->z = LinD->z*t; 5% licm 5% gvn			gvn	
5% gvn load of type double eliminat	252		J	
5% gvn load of type double eliminat		5%	licm	failed to hoist load with loo
			gvn	
	253		J	

```
VECTOR * restrict LinD,
TURE **txt )
```

ot explicitly forced

```
te object into "line-space" */
```



op-invariant address because load is conditionally executed ated in favor of phi

```
op-invariant address because load is conditionally executed
ated in favor of <u>phi</u>
ate surface normal */
```



211			<pre>static double IntersectObjs(VECTOR * restrict LinP, V</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238			<pre>Pos = objs[objn].pos;</pre>
239			Pos.x -= LinP->x; /* Translate
	100%	licm	load hoisted
240			Pos.y -= LinP->y;
	100%	licm	load hoisted
241			
	100%	licm	
	20%	gvn	
	20%	gvn	
242			Dynamic In
	100%	licm	
	100%	licm	
	100%	licm	
	100%	licm	Dod
	100%	licm	Red
	100%	licm	
	20%	gvn	
243		, , , , , , , , , , , , , , , , , , ,	
244			
245			
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			<pre>t = ttmp;</pre>
250			Pnt->x = LinD->x*t; /* Calculate
251			Pnt->y = LinD->y*t;
	5%	licm	failed to hoist load with loop
	5%	gvn	load of type double eliminat
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loop
	5%	gvn	load of type double eliminat
253			Norm->x = Pnt->x-Pos.x; /* Calcualat

VECTOR * restrict LinD, TURE **txt)

ot explicitly forced

e object into "line-space" */

Instruction Count luced by **11%**

cause its definition is unavailable
A;
)) {

e intersection point */

op-invariant address because load is conditionally executed ated in favor of phi

op-invariant address because load is conditionally executed ated in favor of phi ate surface normal */



211			<pre>static double IntersectObjs(VECTOR * restrict LinP, V</pre>
212			VECTOR *Pnt, VECTOR *Norm, TEXTU
213			{
237			<pre>for(objn = 0; objn < NUMOBJS; objn++) {</pre>
207	99%	loop-vectorize	loop not vectorized: vectorization is not beneficial and is not
238	5576	1000-1000120	Pos = objs[objn].pos;
239			Pos.x -= LinP->x; /* Translate
	100%	licm	load hoisted
240			Pos.y -= LinP->y;
	100%	licm	load hoisted
241			
	100%	licm	
	20%	gvn	
	20%	gvn	
242			
	100%	licm	
	100%	licm	Darfarmor
	100%	licm	Pertormar
	100%	licm	
	20%	gvn	
243			
244			
245			
246			A = sqrt(A);
	9%	inline	sqrt will not be inlined into IntersectObjs beca
247			if($(ttmp = B - A) < EPSILON$) $ttmp = B +$
248			if ((EPSILON <ttmp) &&="" (="" (t<0.0)<="" (ttmp<t)="" td="" =""></ttmp)>
249			t = ttmp;
250			Pnt->x = LinD->x*t; /* Calculate
251			Pnt->y = LinD->y*t;
	5%	licm	failed to hoist load with loo
	5%	gvn	load of type double elimination
252			Pnt->z = LinD->z*t;
	5%	licm	failed to hoist load with loo
0.50	5%	gvn	load of type double elimination
253			Norm->x = Pnt->x-Pos.x; /* Calcualat

VECTOR * restrict LinD, TURE **txt)

ot explicitly forced

te object into "line-space" */

nce headroom 11%

cause its definition is unavailable
A;
)) {

te intersection point */

op-invariant address because load is conditionally executed ated in favor of phi

op-invariant address because load is conditionally executed ated in favor of phi ate surface normal */



Freebench/distray: Summary

- Found optimization opportunity while staying in the high-level view
 - Reconstructed the reason for missed optimization
 - High-level view exposed that the gain may be substantial
 - Got immediate feedback of the desired effect on the prototype
 - Identified the pass for low-level debugging

Check Out More Examples

http://lab.llvm.org:8080/artifacts/opt-view_test-suite

Development Timeline Initial version on LLVM trunk Now Compiler Developer Tool

Code Author Tool

New tools using Optimization Records

Compiler Developer Tool: Status

- Written in Python
- Hook up new passes
- Improve diagnostics quality for existing passes
 - Perform extra analysis for insightful messages
- Improve UI

Compiler Developer Tool: Status

Hein

- Written in Python
- Hook up new passes
- Improve diagnostics statity for
 - Perform extra analysis for inst
- Improve

y for existing passes

arysis for insightful messages

Code Author Tool: Wishlist

- Suggest specific actions
 - E.g. for the LICM case: if the two pointers can never point to the same object consider using 'restrict'
 - Add new "recommendation" analysis passes to detect opportunity and suggest:
 - Source annotation to enable off-by-default passes (aggressive loop transformations, non-temporal stores)
 - Refactoring: data transformations

Code Author Tool: Wishlist

- Suggest specific actions
 - same object consider using '
 - Add new "recommend Collect and suggest:
 - loop transformations, non-temporal stores)
 - Refactoring: data transformations

• E.g. for the LICM case: if the two pointers can never point to the alysis passes to detect opportunity

Source an otation to enable off-by-default passes (aggressive)

Optimization Records: New Tools

- llvm-opt-report
- Performance regression analysis
- optimization
- Bottom-up search for performance opportunities
 - See all the LICM opportunities like in Freebench/distray

Optimization statistics with the ability to zoom into the particular

Optimization Records: New Tools

- llvm-opt-report
- Performance regression analysis
- Optimizatid SELECT optimizations FROM optimizatio pass = 'licm' AND WHERE
- Bottom-up ORDER BY hotness
 - See all the LICM opportunities like in Freebench/distray

benchmark, hotspot, hotness type = 'missed' AND name = 'LoadWithLoopInvariantAddressInvalidated'

Optimization Records: New Tools

- llvm-opt-report
- Performance regression analysis
- optimization
- Bottom-up search for performance opportunities
 - See all the LICM opportunities like in Freebench/distray
 - in the hottest code

Optimization statistics with the ability to zoom into the particular

Allows finding opportunities that occur with high frequency but not

Acknowledgement

- Tyler Nowicki
- John McCall
- Hal Finkel



SIBsim4 (MultiSource/Applications)

Finding Performance Opportunity

846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
0.10	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
050	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty;	
854	201	lie we	n->prev = i;	the large second
	3%	licm	shl hoisted	link_msps
	3%	licm	and hoisted	link_msps
855				
856 857				
857				



846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	· · · · · · · · · · · · · · · · · · ·	link_msps
	3%	gvn		link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm		link_msps
	3%	licm		link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm		link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty;	
854			n->prev = i;	
	3%	licm	shl hoisted	link_msps
	3%	licm	and hoisted	link_msps
855			}	
856			}	
857			}	

68



046			for $(i = i + 1, i < \text{stop}, i + 1)$	
846	10.0%	loop vesterize	for $(j = i + 1; j < stop; j++)$ {	link mono
	100% 100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847	100%	100p-vectorize	<pre>exon_p_t n = mCol->e.exon[j];</pre>	<u>link_msps</u>
047	99%	licm	failed to hoist load with loop-invariant address	link mene
	100%		load eliminated by PRE	link_msps link_msps
	99%	gvn licm	failed to hoist load with loop-invariant address	link_msps
848	0070	liciti	<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	<u></u>
0.0	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty;	
854			n->prev = i;	
	3%	licm	shl hoisted	link_msps
	3%	licm	and hoisted	link_msps
855			}	
856			}	
857			}	



846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
040	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			$exon_ptn = mCol -> e.exon[j];$	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
0.50	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty;	
854	00/	l'and	n->prev = i;	Parts and a
	3%	licm	shi hoisted	link_msps
055	3%	licm	and hoisted	link_msps
855			}	
856			,	
857			3	



12	s	tatic inline int	
13	1	ies_after_p(exon_p_t a, exon_p_t b)	
14	{		
15		/* When we have some overlap, make sure it is only a small part. */	
16		/*	
17			
18		p1 p2 p3 */	
19			
20		if (b->from1 > a->to1) {	
99%	licm	getelementptr hoisted	link_msps
99%	licm	failed to hoist load with loop-invariant address	link_msps
100%	licm	failed to hoist load with loop-invariant address	link_msps
21	norm	unsigned int p1;	IIIK_III303
22		unsigned int p2;	
23		unsigned int p3;	
23		if $(b \rightarrow from 2 > a \rightarrow to 2)$	
	liam		link mene
99%	licm	getelementptr hoisted	link_msps
99%	liem	failed to hoist load with loop-invariant address	link_msps
100%	licm	failed to hoist load with loop-invariant address	link_msps
25		return 1;	
26		if $(b \rightarrow from 2 < a \rightarrow from 2 b \rightarrow to 2 < a \rightarrow to 2)$	Patrice and a
41%	licm	getelementptr hoisted	link_msps
41%	liem	failed to hoist load with loop-invariant address	link_msps
41%	licm	failed to hoist load with loop-invariant address	link_msps
27		return 0;	
28		$p1 = b \rightarrow from 2 - a \rightarrow from 2;$	
29		p2 = a - to2 - b - from2;	
30		p3 = b - > to2 - a - > to2;	
1		if (pl > p2 && p3 > p2 && p1 > options.K && p3 > options.K)	
2		return 1;	
3		<pre>} else if (b->from2 > a->to2) {</pre>	
4		unsigned int p1;	
5		unsigned int p2;	
6		unsigned int p3;	
7		if $(b \rightarrow from 1 \le a \rightarrow from 1 \parallel b \rightarrow to 1 \le a \rightarrow to 1)$	
2%	licm	aetelementptr hoisted	link_msps
2%	liem	failed to hoist load with loop-invariant address	link_msps
2%	licm	failed to hoist load with loop-invariant address	link_msps
8		return 0;	
9		$pl = b \rightarrow from1 - a \rightarrow from1;$	
10		$p2 = a \rightarrow tol - b \rightarrow froml;$	
11		p3 = b - to1 - a - to1;	
2		if $(p1 > p2 \& \& p3 > p2 \& \& p1 > options.K \& \& p3 > options.K)$	
13		return 1;	
4		}	
5		return 0;	
6	1		
	,		

71

846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
0.0	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty;	
854			n->prev = i;	
	3%	licm	shl hoisted	link_msps
	3%	licm	and hoisted	link_msps
855			}	
856			}	
857			}	



846			<pre>for (j = i + 1; j < stop; j++) {</pre>		
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer		link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info		link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>		
	99%	licm	failed to hoist load with loop-invariant address		link_msps
	100%	gvn	load eliminated by PRE		link_msps
	99%	licm	failed to hoist load with loop-invariant address		link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>		
	97%	inline	<pre>lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)</pre>		link_msps
	97%	inline	lies_after_p inlined into link_msps		link_msps
	55%	licm	failed to hoist load with loop-invariant address	Look at	link_msps
	55%	licm	failed to hoist load with loop-invariant address		link_msps
849			unsigned int penalty;	the loads	
850			penalty = $abs(n-from1 - m-from1) >> 15;$		
	3%	licm	getelementptr hoisted		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
	3%	gvn	load of type i32 eliminated		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
851	0.07		penalty $+=$ abs(n->from2 - m->from2) >> 15;		1
	3%	licm	getelementptr hoisted		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
	3%	gvn	load of type i32 eliminated		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
852			<pre>if (penalty < m->Score) {</pre>		
853			n->Score = m->Score - penalty;		
854	29/	liom	n->prev = i; shl hoisted		link mono
	3% 3%	licm licm	and hoisted		link_msps
855	370	ncm	and noisted		link_msps
856					
857					
007			F		



846			<pre>for (j = i + 1; j < stop; j++) {</pre>		
040	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer		link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info		link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>		<u></u>
	99%	licm	failed to hoist load with loop-invariant address		link_msps
	100%	gvn	load eliminated by PRE		link_msps
	99%	licm	failed to hoist load with loop-invariant address		link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>		
	97%	inline	<pre>lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)</pre>		link_msps
	97%	inline	lies_after_p inlined into link_msps		link_msps
	55%	licm	failed to hoist load with loop-invariant address	Look at	link_msps
	55%	licm	failed to hoist load with loop-invariant address		link_msps
849			unsigned int penalty;	the loads	
850			penalty = $abs(n->from1 - m->from1) >> 15;$		
	3%	licm	getelementptr hoisted		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
	3%	gvn	load of type i32 eliminated		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
851	001		<pre>penalty += abs(n->from2 - m->from2) >> 15;</pre>		1
	3%	licm	getelementptr hoisted		link_msps
	3%	licm	failed to hoist load with loop-invariant address		link_msps
	3% 3%	gvn	load of type i32 eliminated		link_msps
852	3%	licm	<pre>failed to hoist load with loop-invariant address if (penalty < m->Score) {</pre>		<u>link_msps</u>
853			n->Score = m->Score - penalty;		
854			n->prev = i;		
004	3%	licm	shl hoisted		link_msps
	3%	licm	and hoisted		link_msps
855			}		
856			}		
857			}		



712			static inline int	
713			lies_after_p(exon_p_t a, exon_p_t b)	
714			{	
715			/* When we have some overlap, make sure it is only a small part. */	
716			/*	
717				
718			p1 p2 p3 */	
719			br br to 1	
720			if $(b \rightarrow from 1 \ge a \rightarrow to 1)$ {	
	0.0%	liam		link mone
	99%	licm	getelementptr hoisted	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	licm	failed to hoist load with loop-invariant address	link_msps
721			unsigned int p1;	
722			unsigned int p2;	
723			unsigned int p3;	
724			if (b->from2 a->to2)	
	99%	licm	getelementptr hoisted	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	licm	failed to hoist load with loop-invariant address	link_msps
725			return 1;	
726			if $(b \rightarrow from 2 < a \rightarrow from 2 b \rightarrow to 2 < a \rightarrow to 2)$	
4	41%	licm	getelementptr hoisted	link_msps
	41%	licm	failed to hoist load with loop-invariant address	link_msps
4	41%	licm	failed to hoist load with loop-invariant address	link_msps
727			return 0;	
728			$p1 = b \rightarrow from 2 - a \rightarrow from 2;$	
729			p2 = a - to2 - b - from2;	
730			p3 = b - > to2 - a - > to2;	
731			if (pl > p2 && p3 > p2 && pl > options.K && p3 > options.K)	
732			return 1;	
733			$else if (b->from 2 > a->to 2) {$	
734			unsigned int p1;	
735			unsigned int p2;	
736			unsigned int p3;	
737			if $(b \rightarrow from1 < a \rightarrow from1 b \rightarrow to1 < a \rightarrow to1)$	
1	2%	licm	getelementptr hoisted	link_msps
	2%	licm	failed to hoist load with loop-invariant address	link_msps
:	2%	licm	failed to hoist load with loop-invariant address	link_msps
738			return 0;	
739			$p1 = b \rightarrow from1 - a \rightarrow from1;$	
740			$p2 = a \rightarrow tol - b \rightarrow froml;$	
741			$p3 = b \rightarrow to1 - a \rightarrow to1;$	
742			if $(p1 > p2 \& \& p3 > p2 \& \& p1 > options.K \& \& p3 > options.K)$	
743			return 1;	
744			}	
745			return 0;	
746			1	
/***			1	

846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) { n->Score = m->Score - penalty; LOOK at</pre>	
853				
854			n->prev = i; shi hoisted the stores	
	3%	licm	Shi hoistea	link_msps
	3%	licm	and hoisted	link_msps
855			}	
856			}	
857			}	



846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
040	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			$exon_ptn = mCol -> e.exon[j];$	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	<pre>lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)</pre>	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
852			<pre>if (penalty < m->Score) { n->Score = m->Score - penalty; LOOK at</pre>	
853				
854	29/	liene	n->prev = i; shi hoisted the stores	link, mana
	3%	licm	Shi hoisted	link_msps
955	3%	licm	and hoisted	link_msps
855 856			۲ ٦	
			ן ז	
857			1	



846			<pre>for (j = i + 1; j < stop; j++) {</pre>	
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>	
	99%	licm	failed to hoist load with loop-invariant address	link_msps
	100%	gvn	load eliminated by PRE	link_msps
	99%	licm	failed to hoist load with loop-invariant address	link_msps
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>	
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps
	97%	inline	lies_after_p inlined into link_msps	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
	55%	licm	failed to hoist load with loop-invariant address	link_msps
849			unsigned int penalty;	
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;	
	3%	licm	getelementptr hoisted	link_msps
	3%	licm	failed to hoist load with loop-invariant address	link_msps
	3%	gvn	load of type i32 eliminated	link_msps
	3%	licm		link_msps
852			<pre>if (penalty < m->Score) {</pre>	
853			n->Score = m->Score - penalty; Can 'm' and 'n'	
854			n->prev = i;	
	3%	licm	shi hoisted really alias?	link_msps
	3%	licm	and hoisted	link_msps
855			}	
856			}	
857			}	



			$exon_pt m = mCol -> e.exon[i];$		
846			<pre>for (j = i + 1; j < stop; j++) { Probably not! </pre>		
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectorizer	link_msps	
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more info	link_msps	
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>		
	99%	licm	failed to hoist load with loop-invariant address	link_msps	
	100%	gvn	load eliminated by PRE	link_msps	
	99%	licm	failed to hoist load with loop-invariant address	link_msps	
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>		
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps	
	97%	inline	lies_after_p inlined into link_msps	link_msps	
	55%	licm	failed to hoist load with loop-invariant address	link_msps	
	55%	licm	failed to hoist load with loop-invariant address	link_msps	
849			unsigned int penalty;		
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$		
	3%	licm	getelementptr hoisted	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
	3%	gvn	load of type i32 eliminated	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;		
	3%	licm	getelementptr hoisted	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
	3%	gvn	load of type i32 eliminated	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
852			<pre>if (penalty < m->Score) {</pre>		
853			n->Score = m->Score - penalty;		
854			n->prev = i;		
	3%	licm	shl hoisted	link_msps	
	3%	licm	and hoisted	link_msps	
855			}		
856			}		
857			}		
			70		



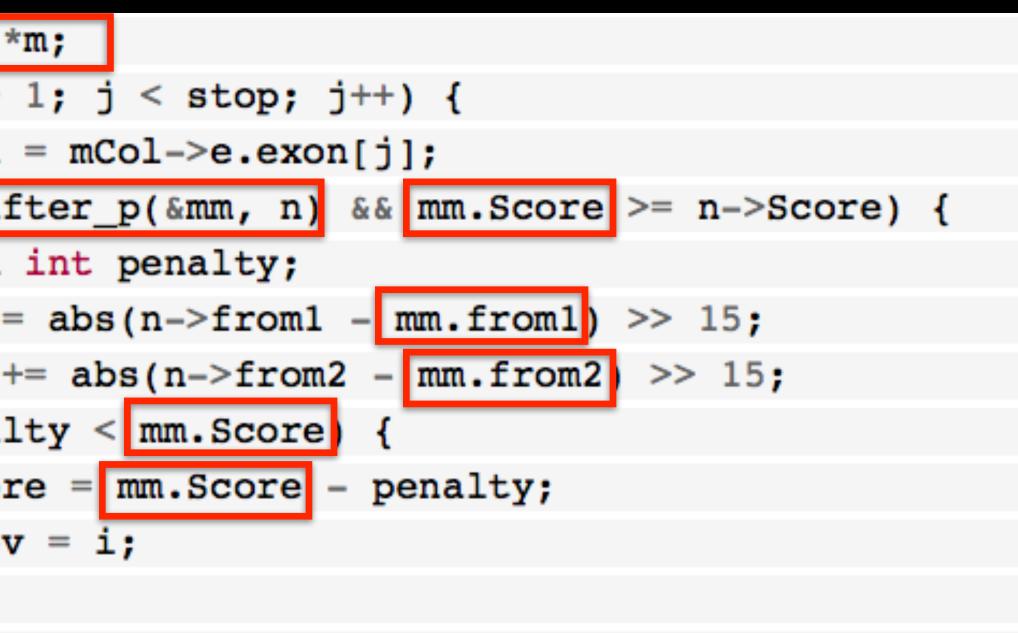
			<pre>exon_p_t m = mCol->e.exon[i];</pre>		
846			<pre>for (j = i + 1; j < stop; j++) { We need to use 'restrict' </pre>		
	100%	loop-vectorize	loop not vectorized: loop control flow is not ur	link_msps	
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop	link_msps	
847			<pre>exon_p_t n = mCol->e.exon[j];</pre>		
	99%	licm	failed to hoist load with loop-invariant address	link_msps	
	100%	gvn	load eliminated by PRE	link_msps	
	99%	licm	failed to hoist load with loop-invariant address	link_msps	
848			<pre>if (lies_after_p(m, n) && m->Score >= n->Score) {</pre>		
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14785 (threshold=325)	link_msps	
	97%	inline	lies_after_p inlined into link_msps	link_msps	
	55%	licm	failed to hoist load with loop-invariant address	link_msps	
	55%	licm	failed to hoist load with loop-invariant address	link_msps	
849			unsigned int penalty;		
850			penalty = $abs(n-sfrom1 - m-sfrom1) >> 15;$		
	3%	licm	getelementptr hoisted	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
	3%	gvn	load of type i32 eliminated	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
851			penalty $+=$ abs(n->from2 - m->from2) >> 15;		
	3%	licm	getelementptr hoisted	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
	3%	gvn	load of type i32 eliminated	link_msps	
	3%	licm	failed to hoist load with loop-invariant address	link_msps	
852			<pre>if (penalty < m->Score) {</pre>		
853			n->Score = m->Score - penalty;		
854			n->prev = i;		
	3%	licm	shl hoisted	link_msps	
075	3%	licm	and hoisted	link_msps	
855					
856					
857			}		



846		exon_t mm = *
847		for (j = i +
848		exon_p_t n
849		<pre>if (lies_af</pre>
850		unsigned
851		penalty =
852		penalty +
853		if (penal
854		n->Scor
855		n->prev
856		}
857		}
858		}

```
*m;
1; j < stop; j++) {
= mCol->e.exon[j];
fter_p(&mm, n) && mm.Score >= n->Score) {
int penalty;
= abs(n->from1 - mm.from1) >> 15;
+= abs(n->from2 - mm.from2) >> 15;
lty < mm.Score) {</pre>
re = mm.Score - penalty;
v = i;
```

846		exon_t mm =	*
847		for (j = i +	•
848		exon_p_t n	L
849		if (lies_a	f
850		unsigned	
851		penalty	=
852		penalty	+
853		if (pena	1
854		n->Sco	r
855		n->pre	v
856		}	
857		}	
858		}	



846			<pre>exon_t mm = *m;</pre>		
847			<pre>for (j = i + 1; j < stop; j++) {</pre>		
	100%	loop-vectorize	loop not vectorized: loop control flow is not understood by vectoriz		
	100%	loop-vectorize	loop not vectorized: use -Rpass-analysis=loop-vectorize for more i		
848			<pre>exon_p_t n = mCol->e.exon[j];</pre>		
	99%	licm	failed to hoist load with loop-invariant		
	100%	gvn	load of type %structexon_t** not elim		
	100%	gvn	load eliminated by PRE		
	0%	gvn	load of type %structexon_t** not elim		
	99%	licm	failed to hoist load with loop-invariant		
849			<pre>if (lies_after_p(&mm, n) && mm.Score >= n->Score) {</pre>		
	97%	inline	lies_after_p can be inlined into link_msps with cost=-14805		
	97%	inline	lies_after_p inlined into link_msps		
850		unsigned int penalty;			
851					
	3%	gvn	load of type i32 eliminated		
852			<pre>penalty += abs(n->from2 - mm.from2) >> 15;</pre>		
	3%	gvn	load of type i32 eliminated		
853			<pre>if (penalty < mm.Score) {</pre>		
854			n->Score = mm.Score - penalty;		
855			n->prev = i;		
	3%	licm	shI hoisted		
	3%	licm	and hoisted		
856			}		
857			}		
858			}		
000			·		

izer e info

t address because the loop may invalidate its value minated in favor of <u>load</u> because it is clobbered by <u>store</u>

minated in favor of <u>load</u> because it is clobbered by <u>store</u> t address because the loop may invalidate its value

5 (threshold=325)



```
static inline int
712
713
                               lies_after_p(exon_p_t a, exon_p_t b)
714
                                 /* When we have some overlap, make sure it is only a small part. */
715
716
                                 /* _____
717
                                         p1 p2 p3 */
718
719
                                 if (b->from1 > a->to1) {
720
                                   unsigned int pl;
721
                                   unsigned int p2;
722
                                   unsigned int p3;
723
                                   if (b \rightarrow from 2 > a \rightarrow to 2)
724
725
                                     return 1;
                                   if (b \rightarrow from 2 < a \rightarrow from 2 | | b \rightarrow to 2 < a \rightarrow to 2)
726
                                     return 0;
727
                                   p1 = b \rightarrow from 2 - a \rightarrow from 2;
728
                                   p2 = a - > to2 - b - > from2;
729
                                   p3 = b -> to2 - a -> to2;
730
                                   if (p1 > p2 && p3 > p2 && p1 > options.K && p3 > options.K)
731
732
                                     return 1;
                                 } else if (b->from2 > a->to2) {
733
734
                                   unsigned int pl;
                                   unsigned int p2;
735
                                   unsigned int p3;
736
                                   if (b \rightarrow from 1 < a \rightarrow from 1 | b \rightarrow to 1 < a \rightarrow to 1)
737
738
                                     return 0;
                                   p1 = b->from1 - a->from1;
739
                                   p2 = a \rightarrow to1 - b \rightarrow from1;
740
741
                                   p3 = b -> to1 - a -> to1;
                                   if (p1 > p2 \&\& p3 > p2 \&\& p1 > options.K \&\& p3 > options.K)
742
743
                                     return 1;
744
745
                                 return 0;
746
```

84