redhat

Consistent Views at Recommended Breakpoints

Alexandre Oliva

aoliva@redhat.com

http://identi.ca/lxoliva/

http://people.redhat.com/~aoliva/

GCC Summit, October, 2010

Summary

- Motivation
- Background
  - Line numbers
  - Variable locations
- Consistent views
- Statement frontiers
- DWARF extensions

Motivation

- Set breakpoint at a line...
  - Stop at a later line, results clobbered
  - Computation of previous lines not complete
  - Stepping bounces back and forth
- Recommended inspection points

Sample Program

	C		RISC asm
1	int f(a, b, c, d) {		f:
2	int x = a + b;		r2(a) ← *(sp+ 4) r3(b) ← *(sp+ 8) r4(x) ← r2(a) + r3(b)
3	int y = c / d;	⇒	r5(c) ← *(sp+12) r6(d) ← *(sp+16) r7(y) ← r5(c) / r6(d)
4	x -= y;		
5	return x;		r1(x) ← r4(x) - r7(y)
6	}		ret

Optimization

Before sched	
r2(a) ← *(sp+ 4)	2
r3(b) ← *(sp+ 8)	2
r4(x) ← r2(a) + r3(b)	2
r5(c) ← *(sp+12)	3
r6(d) ← *(sp+16)	3
r7(y) ← r5(c) / r6(d)	3
r1(x) ← r4(x) - r7(y)	5
ret	5

⇒

After sched	
3	r5(c) ← *(sp+12)
3	r6(d) ← *(sp+16)
2	r2(a) ← *(sp+ 4)
2	r3(b) ← *(sp+ 8)
3	r7(y) ← r5(c) / r6(d)
2	r4(x) ← r2(a) + r3(b)
5	r1(x) ← r4(x) - r7(y)
5	ret

Line Numbers

PC	loc: F L is_stmt S
	.loc 1 3 is_stmt 1
0	r5(c) ← *(sp+12)
4	r6(d) ← *(sp+16)
	.loc 1 2 is_stmt 1
8	r2(a) ← *(sp+ 4)
12	r3(b) ← *(sp+ 8)
	.loc 1 3 is_stmt 1
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)
28	ret

⇒

PC = &f - .text
PC += 0, L += 3
PC += 8, L += -1
PC += 8, L += 1
PC += 4, L += -1
PC += 4, L += 3

≡

PC	L	S
0	3	1
.		0
.	3	0
8	2	1
16	3	1
20	2	1
24	5	1

Variable Location Lists

	.loc 1 3 is_stmt 1
0	r5(c) ← *(sp+12)
4	r6(d) ← *(sp+16)
	.loc 1 2 is_stmt 1
8	r2(**a**) ← *(**sp**+ **4**)
12	r3(b) ← *(sp+ **8**)
	.loc 1 3 is_stmt 1
16	r**7**(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)
28	ret

⇒

:	≤PC<		Len	Expr
a	0	**32**	2	breg**0 4**
	12	32	1	reg2
	0	0		
b	0	32	2	breg0 **8**
	16	32	1	reg3
	0	0		
y	20	32	1	reg**7**
	0	0		
x	24	28	1	reg4
	28	32	1	reg1
	0	0		

Variable Tracking at Assignments

1	f(a,b,c,d) {
1	# *a* ⇒ **a**
1	# *b* ⇒ b
1	# *c* ⇒ c
1	# *d* ⇒ d
2	int x = a + b;
2	# *x* ⇒ **x**
3	int y = c / d;
3	# *y* ⇒ y
4	x -= y;
4	# *x* ⇒ **x**
5	return x;
6	}

⇒

	# *a* ⇒ ***(sp+ 4)**
...	.loc 1 2 is_stmt 1
8	r2(a) ← *(sp+ 4)
12	r3(b) ← *(sp+ 8)
	.loc 1 3 is_stmt 1
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# *x* ⇒ **r4**(x)
	# *y* ⇒ r7(y)
	# *x* ⇒ **r4(x) - r7(y)**
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)
28	ret

Variable Tracking at Assignments

	# *a* ⇒ *(sp+ 4)
...	.loc 1 2 is_stmt 1
8	r2(a) ← *(sp+ 4)
12	r3(b) ← *(sp+ 8)
	.loc 1 3 is_stmt 1
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# *x* ⇒ r4(x)
	# *y* ⇒ **r7**(y)
	# *x* ⇒ **r4(x) - r7(y)**
	.loc 1 5 is_stmt 1
24	**r1**(x) ← r4(x) - r7(y)
28	ret

⇒

a	0	32	2	breg0 4
	12	32	1	reg2
	0	0		
y	**24**	32	1	reg**7**
	0	0		
x	24	**24**	1	reg4
	24	**32**	6	breg4 0
				breg7 0
				minus
				stack_value
	28	32	1	reg**1**
	0	0		

Consistent Views

	.loc 1 2 is_stmt **1**
8	r2(a) ← *(sp+ 4)
...	.loc 1 3 is_stmt **1**
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# x ⇒ r4(x)
	# y ⇒ r7(y)
	# x ⇒ r4(x) - r7(y)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)

⇒

	.loc 1 2 is_stmt **0**
8	r2(a) ← *(sp+ 4)
...	.loc 1 3 is_stmt **0**
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# x ⇒ r4(x)
	.loc 1 3 is_stmt 1
	# y ⇒ r7(y)
	.loc 1 4 is_stmt 1
	# x ⇒ r4(x) - r7(y)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)

Statement Frontiers

1	int f(a, b, c, d) {
2	int x = a + b;
2	# *x* ⇒ x
3	int y = c / d;
3	# *y* ⇒ y
4	x -= y;
4	# *x* ⇒ x
5	return x;
6	}

⇒

1	int f(a, b, c, d) {
2	**# BOS**
2	int x = a + b;
2	# *x* ⇒ x
3	**# BOS**
3	int y = c / d;
3	# *y* ⇒ y
4	**# BOS**
4	x -= y;
4	# *x* ⇒ x
5	**# BOS**
5	return x;
6	}

DWARF v4 Encoding

...	.loc 1 2 is_stmt 0
8	r2(a) ← *(sp+ 4)
...	.loc 1 3 is_stmt 0
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# x ⇒ r4(x)
	.loc 1 3 is_stmt 1
	# y ⇒ r7(y)
	.loc 1 4 is_stmt 1
	# x ⇒ r4(x) - r7(y)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)

⇒

PC	L	S
8	2	0
16	3	0
20	2	1
24	3	1
24	4	1
24	5	1

+

a	0	32	2	breg0 4
	12	32	1	reg2
	0	0		
y	24	32	1	reg7
	0	0		
x	24	24	1	reg4
	24	32	6	breg4 0
				breg7 0
				minus
				stack_value
	28	32	1	reg1
	0	0		

DWARF Extensions

- Discriminating same-PC labels
- Compact, compatible, asm .loc-able
- Column in line-number table: discriminator?
- New opcodes in location expressions?
- New fields in location lists?
- New attribute for variables?

DWARF v4+ Encoding

...	.loc 1 2 is_stmt 0
8	r2(a) ← *(sp+ 4)
...	.loc 1 3 is_stmt 0
16	r7(y) ← r5(c) / r6(d)
	.loc 1 2 is_stmt 1
20	r4(x) ← r2(a) + r3(b)
	# x ⇒ r4(x)
	.loc 1 3 is_stmt 1
	# y ⇒ r7(y)
	.loc 1 4 is_stmt 1
	# x ⇒ r4(x) - r7(y)
	.loc 1 5 is_stmt 1
24	r1(x) ← r4(x) - r7(y)

⇒

PC.**V**	L	S
8.**0**	2	0
16.**0**	3	0
20.**0**	2	1
24.**0**	3	1
24.**1**	4	1
24.**2**	5	1

variable y

	location_has_views

variable x

	location_has_views

y	24	32	1	reg7
	0	0	. **1 0**	
x	24	24	1	reg4
	24	32	6	breg4 0
				breg7 0
				minus
				stack_value
	28	32	1	reg1
	0	0	. **0 2 2 0 0 0**	

Final remarks

- Assembler-assigned `.view` numbers
- "Works" with old debug info consumers
- Advance views at same PC (J. Kratochvil)
- View per line, stmt, side-effect: "-gO0d"
- Supports reordering, not restructuring
- Implementation underway

Thank you!