redhat

Consistent Views at Recommended Breakpoints (bis)

Alexandre Oliva

aoliva@redhat.com

http://identi.ca/lxoliva/

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

GNU Tools Cauldron, 2017

Summary

- Motivation
- Debug Information
- Statement Frontiers
- Consistent Views
- Representation
- Status

Motivation

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

Sample Program

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

⇒

RISC asm
f:
.loc 1 **2** is_stmt 1 r2(a) ← *(sp+ 4) r3(b) ← *(sp+ 8) r4(x) ← r2(a) + r3(b)
.loc 1 3 is_stmt 1 r5(c) ← *(sp+12) r6(d) ← *(sp+16) r7(y) ← r5(c) * r6(d)
.loc 1 **4** is_stmt 1
.loc 1 5 is_stmt 1 r1(x) ← r4(x) - r7(y) ret

```
(gdb) break f
...at line 2
(gdb) run
2 int x = a + b;
(gdb) next
3 int y = c * d;
(gdb) next
5 return x;
(gdb) print a
1
(gdb) print b
2
(gdb) print x
3
```

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

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

⇒

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

```
(gdb) b f
at line 3
(gdb) run
3 int y =
(gdb) n
2 int x =
(gdb) n
3 int y =
(gdb) p x
optimized
(gdb) n
2 int x =
(gdb) n
5 return
```

Variable Tracking at Assignments

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

⇒

	# *a* ⇒ ***(sp+ 4)**
...	.loc 1 3 is_stmt **1**
...	.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

```
(gdb) b f
at line 3
(gdb) run
3 int y =
(gdb) u 4
5 return
(gdb) p a
1
(gdb) p b
2
(gdb) p y
12
(gdb) p x
-9
```

Statement Frontier Notes

...	# *d* ⇒ d
2	**# STMT**
2	int x = a + b;
	# *x* ⇒ x
3	**# STMT**
3	int y = c * d;
	# *y* ⇒ y
4	**# STMT**
4	x -= y;
	# *x* ⇒ x
5	**# STMT**
5	return x;

⇒

...	# *d* ⇒ *(sp+16)
	.loc 1 3 is_stmt **0**
...	.loc 1 2 is_stmt 1
8	r2(a) ← *(sp+ 4)
12	r3(b) ← *(sp+ 8)
	.loc 1 3 is_stmt **0**
16	r7(y) ← r5(c) * r6(d)
	.loc 1 2 is_stmt **0**
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)
28	ret

```
(gdb) b f
at line 2
(gdb) run
2 int x = a +
(gdb) n
5 return x;
(gdb) b 3
at line 5
(gdb) set x=3
not an lvalue
```

loclist(x):

24	24	r4
24	32	r4 - r7
28	32	r1

Location View Numbering

0.0	.loc 1 3 is_stmt 0 **view -0**
...	.loc 1 2 is_stmt 1 **view 0**
8.0	r2(a) ← *(sp+ 4)
12.0	r3(b) ← *(sp+ 8)
	.loc 1 3 is_stmt 0 **view 0**
16.0	r7(y) ← r5(c) * r6(d)
	.loc 1 2 is_stmt 0 **view 0**
20.0	r4(x) ← r2(a) + r3(b)
	# *x* ⇒ r4(x)
	.loc 1 3 is_stmt 1 **view .lvui**
24.**lvui**	# *y* ⇒ r7(y)
	.loc 1 4 is_stmt 1 **view .lvuj**
24.**lvuj**	# *x* ⇒ r4(x) - r7(y)
	.loc 1 5 is_stmt 1 **view .lvuk**
24.**lvuk**	r1(x) ← r4(x) - r7(y)
28.0	ret

```
8.0:  2 int x = a +
(gdb) n
24.0:  3 int y = c *
(gdb) set x = 175
175
(gdb) n
24.1:  4 x -= y;
(gdb) n
24.2:  5 return x;
(gdb) b 4
at line 4, 24.1
```

locviewlist(x):

24.**0**	24.**2**	r4
24.**2**	32.**0**	r4 - r7
28.**0**	32.**0**	r1

Location View Numbering

- Compiler- or assembler-computed views
- Reinterpreting line-number programs
  - Change PC → reset view
  - Same PC → incremented view
  - Exception: `DW_LNS_fixed_advance_pc`

```
# at view N
.balign 32 # or asm
# at view N+1 or 0?
```

DWARF v5- GNU Extensions

8.0	r2(a) ← *(sp+ 4)
12.0	r3(b) ← *(sp+ 8)
16.0	r7(y) ← r5(c) * r6(d)
20.0	r4(x) ← r2(a) + r3(b)
	# x ⇒ r4(x)
24.**0**	# y ⇒ r7(y)
24.**1**	# x ⇒ r4(x) - r7(y)
24.**2**	r1(x) ← r4(x) - r7(y)
28.0	ret

DIE for y:	
LLSTy	DW_AT_location
LVSTy	**DW_AT_GNU_locviews**
DIE for x:	
LLSTx	DW_AT_location
LVSTx	**DW_AT_GNU_locviews**

LVSTy:		LLSTy:		
1	0	24	32	r7
		0	0	

LVSTx:		LLSTx:		
0	2	24	24	r4
2	0	24	32	r4 - r7
0	0	28	32	r1
		0	0	

DWARF v6+ Proposal

8.0	r2(a) ← *(sp+ 4)
12.0	r3(b) ← *(sp+ 8)
16.0	r7(y) ← r5(c) * r6(d)
20.0	r4(x) ← r2(a) + r3(b)
	# *x* ⇒ r4(x)
24.**0**	# *y* ⇒ r7(y)
24.**1**	# *x* ⇒ r4(x) - r7(y)
24.**2**	r1(x) ← r4(x) - r7(y)
28.0	ret

no `view_pair` ⇒ **implied 0, 0** →

DIE for *y*:	
`location`	LLSTy

DIE for *x*:	
`location`	LLSTx

LLSTy:

`view_pair`	`.uleb128 1, 0`		
`start_end`	24	32	r7
`end`			

LVSTx:

`view_pair`	`.uleb128 0, 2`		
`start_end`	24	24	r4
`view_pair`	`.uleb128 2, 0`		
`start_end`	24	32	r4 - r7
`start_end`	28	32	r1
`end`			

Conclusion

- Inlined Entry Point Markers; other markers?
- Views in other addresses and ranges
- GNU binutils 2.30 (master, users/aoliva/SFN)
- GCC 8? (GIT branch aoliva/SFN)
- GDB, Systemtap, ...?

Thank you!