Measuring the User Debugging Experience

Greg Bedwell

Sony Interactive Entertainment

introducing

DExTer

introducing

Debugging EXperience Tester

introducing

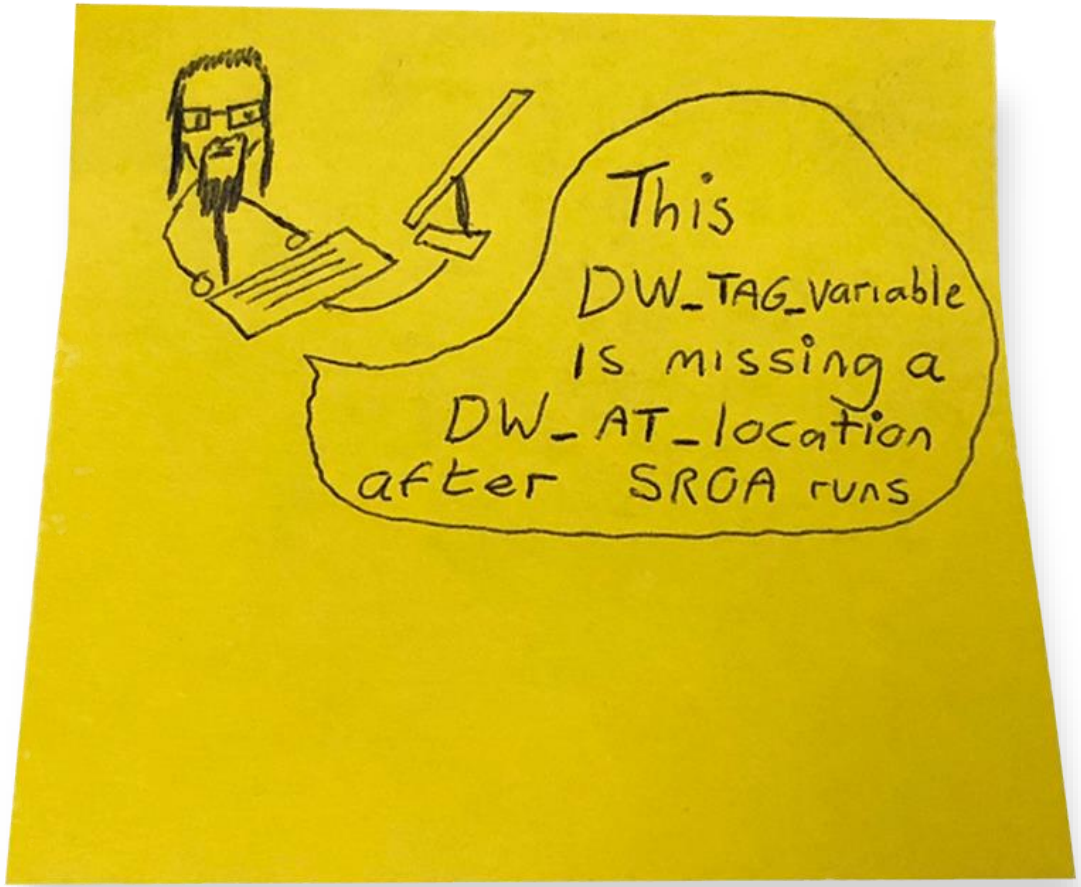
Debugging EXperience Tester

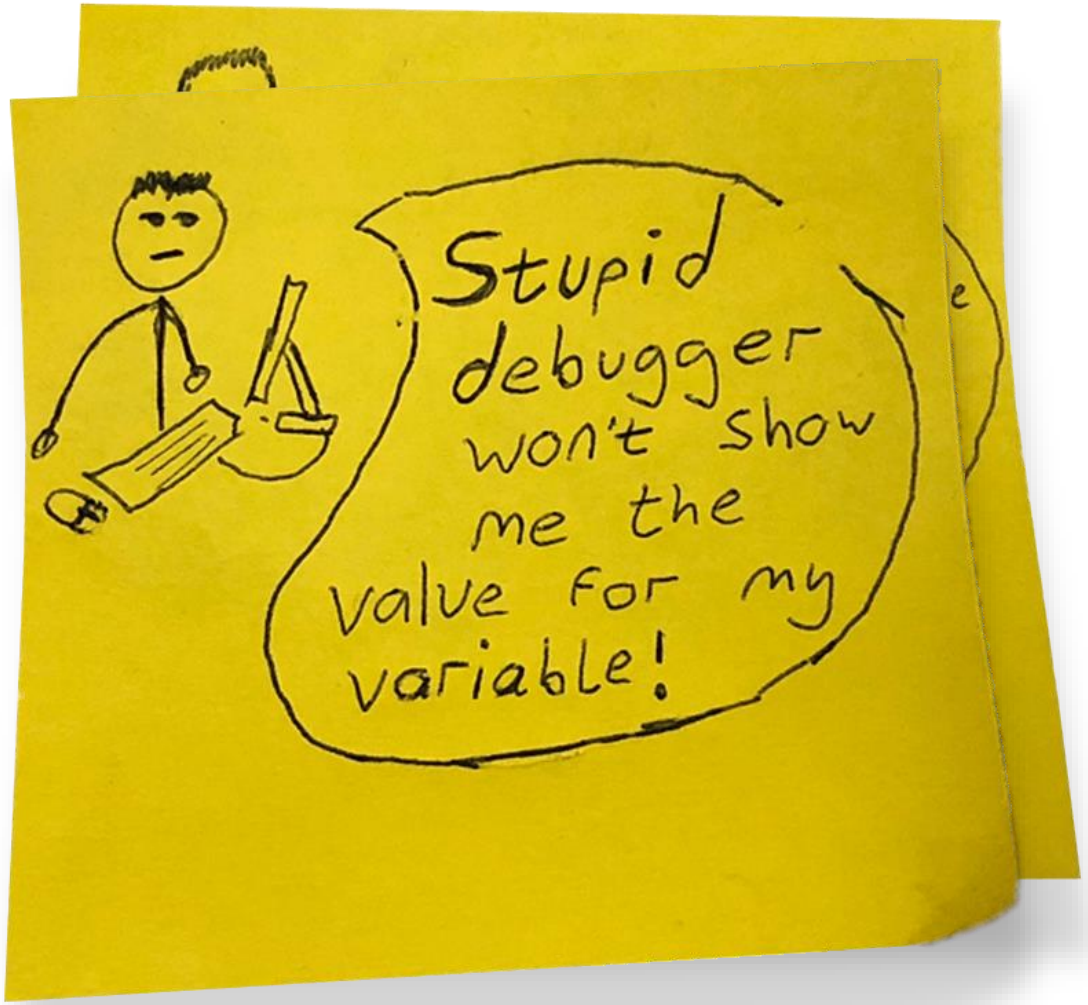
(currently in internal review in preparation for open sourcing)

A suite of tools for viewing and measuring the quality of the user debugging experience

Developed by Greg Bedwell & Stephen Wilks

Python v2.7 and v3.5+


This
DW_TAG_variable
is missing a
DW_AT_location
after SROA runs


Stupid debugger won't show me the value for my variable!

```
1 // ========================================
2 NOINLINE int foo(int x) {
3   return x * 2;  // DexWatch('x')
4 }
5
6 int main(int argc, char**) {
7   return foo(argc);
8 }
9 // ========================================
```

```
1 // ========================================
2 NOINLINE int foo(int x) {
3   return x * 2;  // DexWatch('x')
4 }
5
6 int main(int argc, char**) {
7   return foo(argc);
8 }
9 // ========================================
```

```
$ python dexter.py view example.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 14, "BREAKPOINT", "FUNC", {}]
.   [2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 10, "BREAKPOINT", "FUNC", {"x": "1"}]
[3, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 3, "STEP", "FUNC", {}]
## END (3 steps) ##
```

```
1 // ========================================
2 NOINLINE int foo(int x) {
3   return x * 2;  // DexWatch('x')
4 }
5
6 int main(int argc, char**) {
7   return foo(argc);
8 }
9 // ========================================
```

```
$ python dexter.py view example.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 14, "BREAKPOINT", "FUNC", {}]
.   [2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 10, "BREAKPOINT", "FUNC", {"x": "1"}]
[3, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 3, "STEP", "FUNC", {}]
## END (3 steps) ##
```

```
1 // ============================================
2 NOINLINE int foo(int x) {
3   return x * 2;  // DexWatch('x')
4 }
5
6 int main(int argc, char**) {
7   return foo(argc);
8 }
9 // ============================================
```


```
{
    "step_index": 2,
    "step_kind": "FUNC",
    "stop_reason": "BREAKPOINT",
    "frames": [
      {
        "function": "foo(int)",
        "is_inlined": false,
        "loc": {
          "path": "/home/greg/dexter/tests/example/test.cpp",
          "lineno": 3,
          "column": 10
        }
      },
      {
        "function": "main",
        "is_inlined": false,
        "loc": {
          "path": "/home/greg/dexter/tests/example/test.cpp",
          "lineno": 7,
          "column": 10
        }
      }
    ],
    "watches": {
      "x": {
        "expression": "x",
        "value": "1",
        "type": "int",
        "could_evaluate": true,
        "is_optimized_away": false,
        "could_retrieve_data": true
      }
```

```
1  // ========================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x', '1', on_line=3)
11
12 // DexExpectStepKind('FUNC_EXTERNAL', 0)
13 // DexExpectStepKind('SAME', 0)
14 // DexExpectStepKind('BACKWARD', 0)
15 // ========================================
```

```
1  // ===========================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x
11
12 // DexExpectStepKind('FUN
13 // DexExpectStepKind('SAM
14 // DexExpectStepKind('BAC
15 // ======================
```

-O0 -g

Greg's Linux Machine

```
$ python dexter.py view example-O0g.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 14, "BREAKPOINT", "FUNC", {}]
.   [2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 10, "BREAKPOINT", "FUNC", {"x": "1"}]
[3, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 3, "STEP", "FUNC", {}]
## END (3 steps) ##

$ python dexter.py calc -v example-O0g.json
example-O0g.json = 0/10 (0.0000)

  step kind differences [0/3]
    BACKWARD:
      0

    FUNC_EXTERNAL:
      0

    SAME:
      0

  test.cpp:3 [x] [0/7]
    expected encountered values:
      1


$
```

```
1  // ============================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x
11
12 // DexExpectStepKind('FUN
13 // DexExpectStepKind('SAM
14 // DexExpectStepKind('BAC
15 // ======================
```

-O0 -g

Greg's Linux Machine

```
$ python dexter.py view example-O0g.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 14, "BREAKPOINT", "FUNC", {}]
.   [2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 10, "BREAKPOINT", "FUNC", {"x": "1"}]
[3, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 3, "STEP", "FUNC", {}]
## END (3 steps) ##

$ python dexter.py calc -v example-O0g.json
example-O0g.json = 0/10 (0.0000)

  step kind differences [0/3]
    BACKWARD:
      0

    FUNC_EXTERNAL:
      0

    SAME:
      0

  test.cpp:3 [x] [0/7]
    expected encountered values:
      1


$
```

```
1  // ============================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x
11
12 // DexExpectStepKind('FUN
13 // DexExpectStepKind('SAM
14 // DexExpectStepKind('BAC
15 // ======================
```

-O2 -g

Greg's Linux Machine

```
$ python dexter.py view example-O2g.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 10, "BREAKPOINT", "FUNC", {}]
[2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 12, "BREAKPOINT", "FUNC", {"x": "1"}]
## END (2 steps) ##

$ python dexter.py calc -v example-O2g.json
example-O2g.json = 0/10 (0.0000)

  step kind differences [0/3]
    BACKWARD:
      0

    FUNC_EXTERNAL:
      0

    SAME:
      0

  test.cpp:3 [x] [0/7]
    expected encountered values:
      1


$
```

```
1  // ========================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x
11
12 // DexExpectStepKind('FUN
13 // DexExpectStepKind('SAM
14 // DexExpectStepKind('BAC
15 // ========================
```

-O2 -gmlt

Greg's Linux Machine

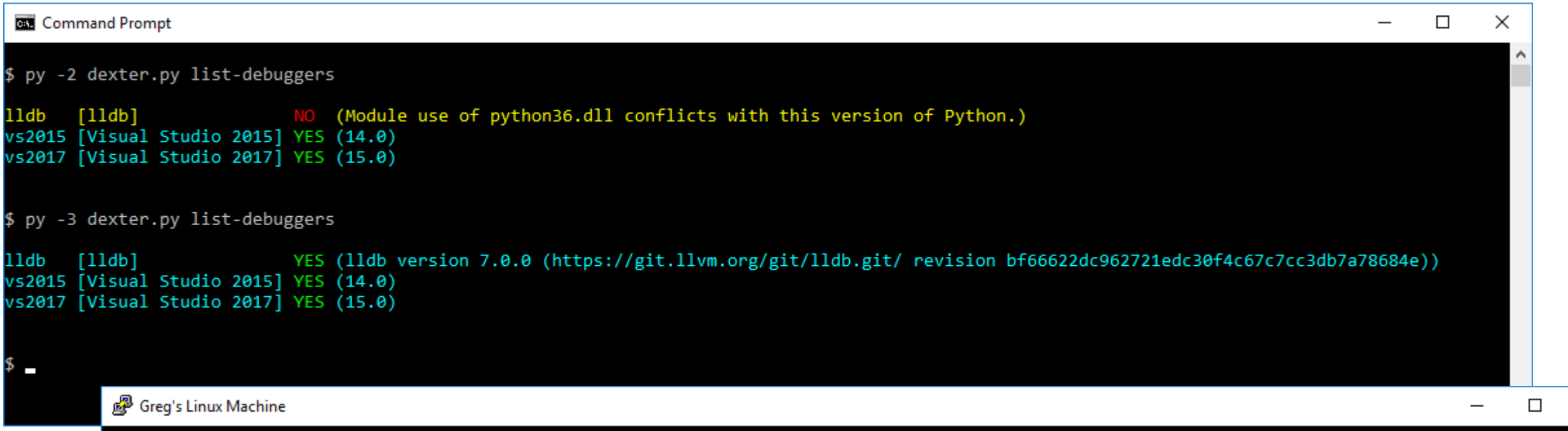
```
$ python dexter.py view example-O2gmlt.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 10, "BREAKPOINT", "FUNC", {}]
[2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 12, "BREAKPOINT", "FUNC", {"x": null}]
## END (2 steps) ##

$ python dexter.py calc -v example-O2gmlt.json
example-O2gmlt.json = 5/10 (0.5000)

  step kind differences [0/3]
    BACKWARD:
      0

    FUNC_EXTERNAL:
      0

    SAME:
      0

  test.cpp:3 [x] [5/7]
    could not evaluate:
      step 2 [+5]


$
```

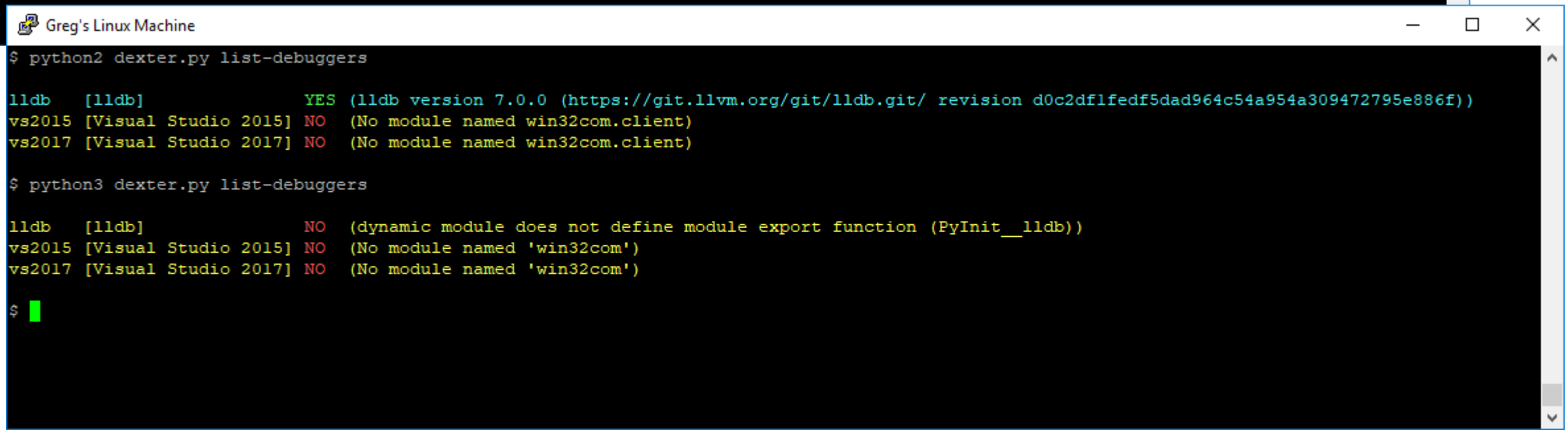
```
1  // ========================================
2  NOINLINE int foo(int x) {
3    return x * 2;  // DexWatch('x')
4  }
5
6  int main(int argc, char**) {
7    return foo(argc);
8  }
9
10 // DexExpectWatchValue('x
11
12 // DexExpectStepKind('FUN
13 // DexExpectStepKind('SAM
14 // DexExpectStepKind('BAC
15 // ========================
```

-O2 -gmlt

Greg's Linux Machine

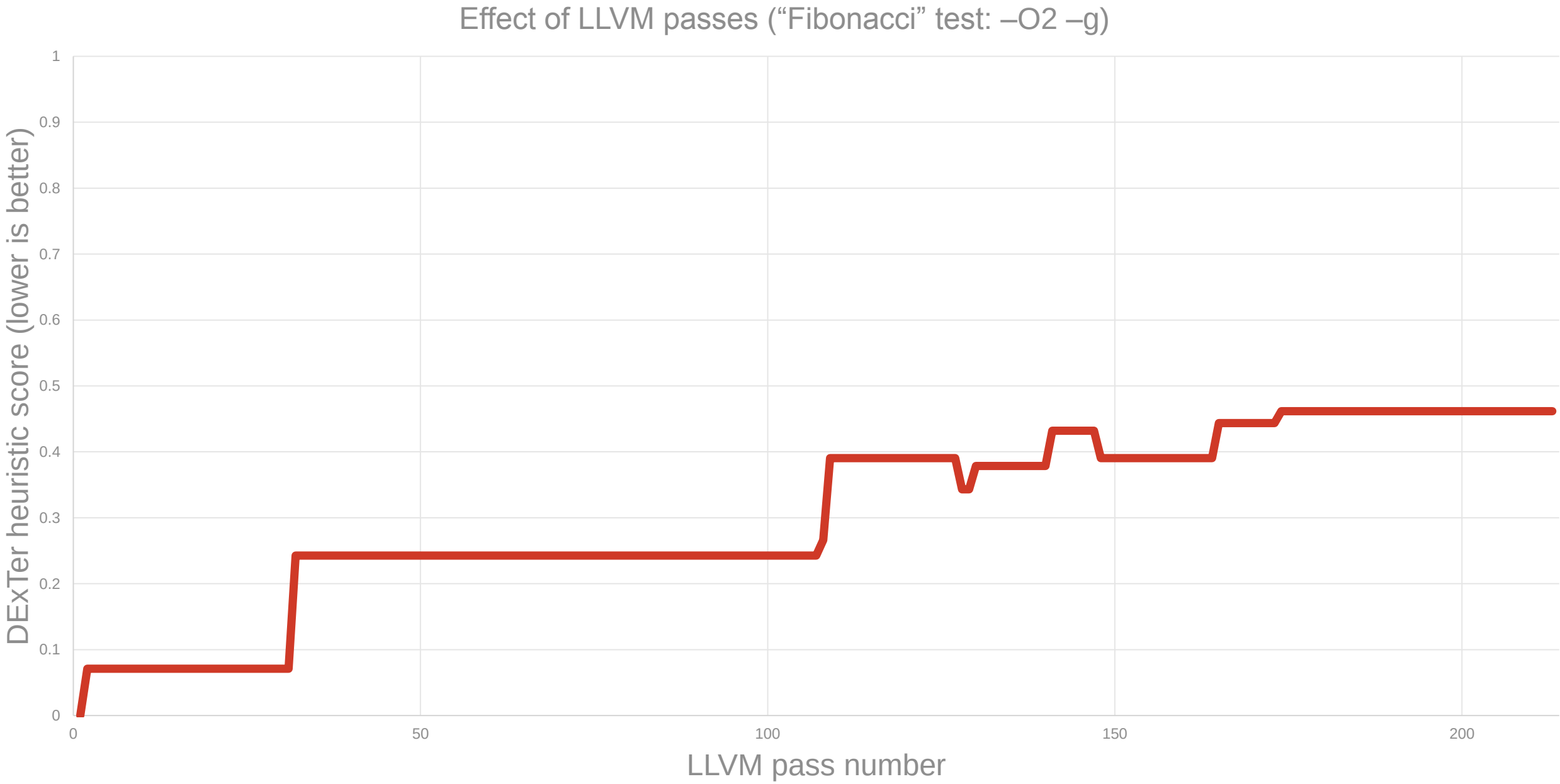
```
$ python dexter.py view example-O2gmlt.json
## BEGIN ##
[1, "main", "/home/greg/dexter/tests/example/test.cpp", 7, 10, "BREAKPOINT", "FUNC", {}]
[2, "foo(int)", "/home/greg/dexter/tests/example/test.cpp", 3, 12, "BREAKPOINT", "FUNC", {"x": null}]
## END (2 steps) ##

$ python dexter.py calc -v example-O2gmlt.json
example-O2gmlt.json = 5/10 (0.5000)

  step kind differences [0/3]
    BACKWARD:
      0

    FUNC_EXTERNAL:
      0

    SAME:
      0

  test.cpp:3 [x] [5/7]
    could not evaluate:
      step 2 [+5]

$
```

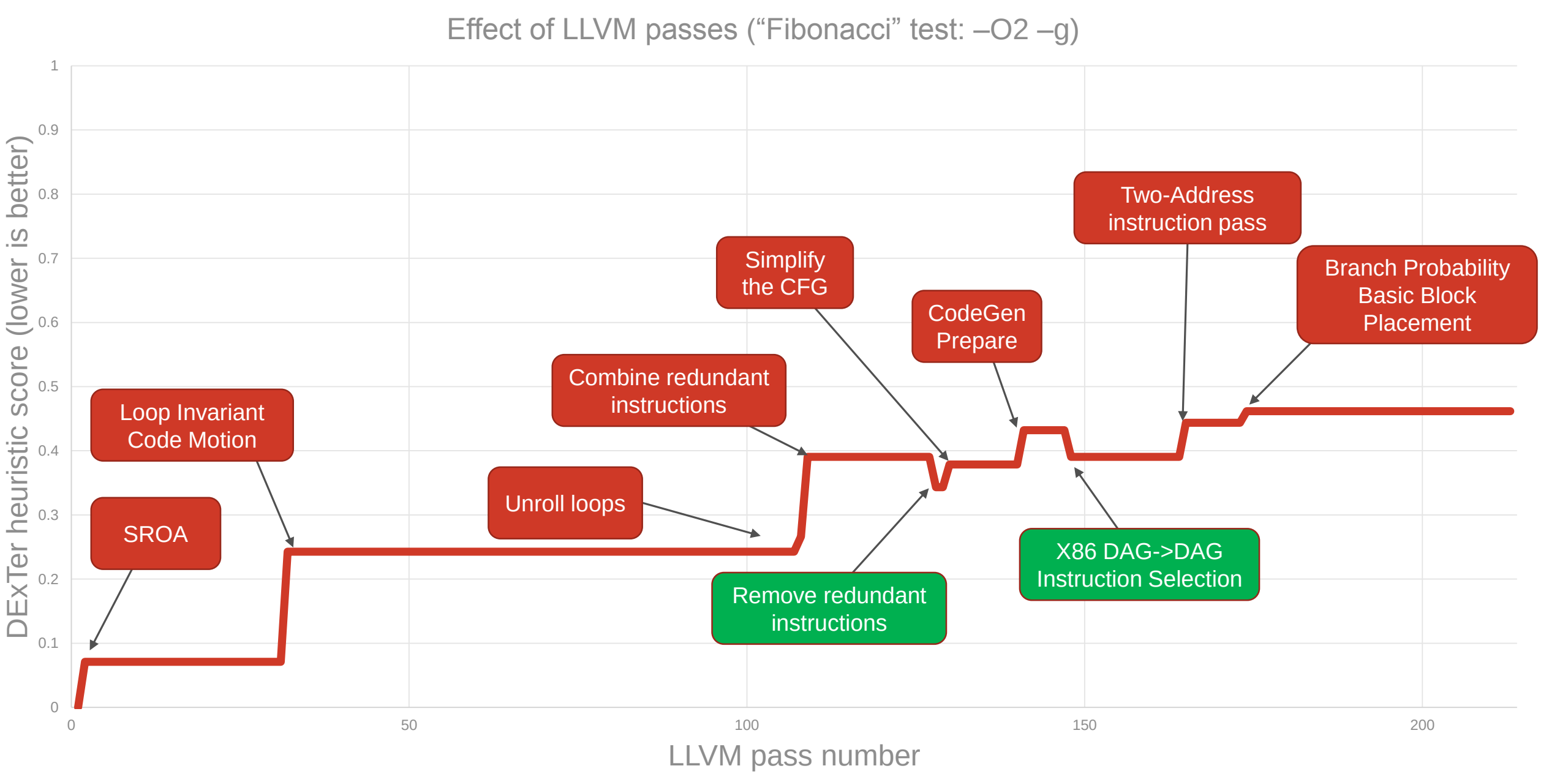

Command Prompt
$ py -2 dexter.py list-debuggers
lldb [lldb] NO (Module use of python36.dll conflicts with this version of Python.)
vs2015 [Visual Studio 2015] YES (14.0)
vs2017 [Visual Studio 2017] YES (15.0)
$ py -3 dexter.py list-debuggers
lldb [lldb] YES (lldb version 7.0.0 (https://git.llvm.org/git/lldb.git/ revision bf66622dc962721edc30f4c67c7cc3db7a78684e))
vs2015 [Visual Studio 2015] YES (14.0)
vs2017 [Visual Studio 2017] YES (15.0)
$


Greg's Linux Machine
$ python2 dexter.py list-debuggers
lldb [lldb] YES (lldb version 7.0.0 (https://git.llvm.org/git/lldb.git/ revision d0c2df1fedf5dad964c54a954a309472795e886f))
vs2015 [Visual Studio 2015] NO (No module named win32com.client)
vs2017 [Visual Studio 2017] NO (No module named win32com.client)
$ python3 dexter.py list-debuggers
lldb [lldb] NO (dynamic module does not define module export function (PyInit__lldb))
vs2015 [Visual Studio 2015] NO (No module named 'win32com')
vs2017 [Visual Studio 2017] NO (No module named 'win32com')
$


Clang trunk over time (“Fibonacci” test: –O2 –g)
DExTer heuristic score (lower is better)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
19 April 2017
08 June 2017
28 July 2017
16 September 2017
05 November 2017
25 December 2017
13 February 2018
Date

```
$ python dexter.py clang-opt-bisect --builder clang --debugger lldb --cflags="-O2 -g" --tests-directory=tests/fibonacci

pass 1/213 = 0/169 (0.0000, 0.0000) [Simplify the CFG on function (_Z9FibonacciiRi)]
pass 2/213 = 12/169 (0.0710, 0.0710) [SROA on function (_Z9FibonacciiRi)]
pass 3/213 = 12/169 (0.0710, 0.0000) [Early CSE on function (_Z9FibonacciiRi)]
pass 4/213 = 12/169 (0.0710, 0.0000) [Simplify the CFG on function (main)]
pass 5/213 = 12/169 (0.0710, 0.0000) [SROA on function (main)]
pass 6/213 = 12/169 (0.0710, 0.0000) [Early CSE on function (main)]
pass 7/213 = 12/169 (0.0710, 0.0000) [Infer set function attributes on module (/home/greg/dexter/tests/fibonacci/test.cpp)]
pass 8/213 = 12/169 (0.0710, 0.0000) [Interprocedural Sparse Conditional Constant Propagation on module (/home/greg/dexter/tests/fibonacci/test.cpp)]
pass 9/213 = 12/169 (0.0710, 0.0000) [Called Value Propagation on module (/home/greg/dexter/tests/fibonacci/test.cpp)]
pass 10/213 = 12/169 (0.0710, 0.0000) [Global Variable Optimizer on module (/home/greg/dexter/tests/fibonacci/test.cpp)]
pass 11/213 = 12/169 (0.0710, 0.0000) [Promote Memory to Register on function (_Z9FibonacciiRi)]
pass 12/213 = 12/169 (0.0710, 0.0000) [Promote Memory to Register on function (main)]
pass 13/213 = 12/169 (0.0710, 0.0000) [Dead Argument Elimination on module (/home/greg/dexter/tests/fibonacci/test.cpp)]
pass 14/213 = 12/169 (0.0710, 0.0000) [Combine redundant instructions on function (_Z9FibonacciiRi)]
pass 15/213 = 12/169 (0.0710, 0.0000) [Simplify the CFG on function (_Z9FibonacciiRi)]
pass 16/213 = 12/169 (0.0710, 0.0000) [Combine redundant instructions on function (main)]
pass 17/213 = 12/169 (0.0710, 0.0000) [Simplify the CFG on function (main)]
pass 18/213 = 12/169 (0.0710, 0.0000) [Remove unused exception handling info on SCC (_Z9FibonacciiRi)]
pass 19/213 = 12/169 (0.0710, 0.0000) [Function Integration/Inlining on SCC (_Z9FibonacciiRi)]
pass 20/213 = 12/169 (0.0710, 0.0000) [Deduce function attributes on SCC (_Z9FibonacciiRi)]
pass 21/213 = 12/169 (0.0710, 0.0000) [SROA on function (_Z9FibonacciiRi)]
pass 22/213 = 12/169 (0.0710, 0.0000) [Early CSE w/ MemorySSA on function (_Z9FibonacciiRi)]
pass 23/213 = 12/169 (0.0710, 0.0000) [Speculatively execute instructions if target has divergent branches on function (_Z9FibonacciiRi)]
pass 24/213 = 12/169 (0.0710, 0.0000) [Jump Threading on function (_Z9FibonacciiRi)]
pass 25/213 = 12/169 (0.0710, 0.0000) [Value Propagation on function (_Z9FibonacciiRi)]
pass 26/213 = 12/169 (0.0710, 0.0000) [Simplify the CFG on function (_Z9FibonacciiRi)]
pass 27/213 = 12/169 (0.0710, 0.0000) [Combine redundant instructions on function (_Z9FibonacciiRi)]
pass 28/213 = 12/169 (0.0710, 0.0000) [Tail Call Elimination on function (_Z9FibonacciiRi)]
pass 29/213 = 12/169 (0.0710, 0.0000) [Simplify the CFG on function (_Z9FibonacciiRi)]
pass 30/213 = 12/169 (0.0710, 0.0000) [Reassociate expressions on function (_Z9FibonacciiRi)]
pass 31/213 = 12/169 (0.0710, 0.0000) [Rotate Loops on loop]
pass 32/213 = 41/169 (0.2426, 0.1716) [Loop Invariant Code Motion on loop]
pass 33/213 = 41/169 (0.2426, 0.0000) [Unswitch loops on loop]
pass 34/213 = 41/169 (0.2426, 0.0000) [Simplify the CFG on function (_Z9FibonacciiRi)]
```


Effect of LLVM passes (“Fibonacci” test: –O2 –g)
DExTer heuristic score (lower is better)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200
LLVM pass number


Effect of LLVM passes (“Fibonacci” test: –O2 –g)
DExTer heuristic score (lower is better)
LLVM pass number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
Loop Invariant Code Motion
SROA
Unroll loops
Combine redundant instructions
Simplify the CFG
CodeGen Prepare
Two-Address instruction pass
Branch Probability Basic Block Placement
Remove redundant instructions
X86 DAG->DAG Instruction Selection

Overall effect of LLVM passes across multiple DExTer tests (-O2 –g)