What does Dyninst need for DWARF rewriting?

Example 1

```
[0x10] int foo(int a, int b) {
     if(a>b) return a+b;
     throw “bad”;
  }
```

Dyninst just relocates function

```
[0x10] int foo(int a, int b) {
     [0x15] tail_call DYNINST_foo;
  }

[0x20] int DYNINST_foo(int a, int b) {
     if(a>b) return a+b;
     [0x25] throw “bad”;
  }
```

DWARF updates here:

New line map entry, eh_table, CFI for DYNINST_foo (0x20)

Copy line map, eh_table, CFI for foo (0x10) and update locations IN SITU by adding offset of new function relative to old one (0x20 - 0x10).

Example 2

Dyninst mutates function, adds function call

```
[0x10] int foo(int a, int b) {
     [0x15] tail_call DYNINST_foo;
  }

[0x2000] int DYNINST_foo(int a, int b) {
     if(a>b) {
[0x2005] traceme();
     return a+b;
     }
     [0x2025] throw “bad”;
  }
```

DWARF updates here:

New line map entry, eh_table, CFI for DYNINST_foo (0x2000)

What should we do with 'traceme' at 0x2005? Where should this point to in the line map table? It has no source location. Can we make up a file "<DYNINST_INSERTED_DYNINST_foo.cpp:NN>".

Copy line map, eh_table, CFI for foo (0x10) and update locations IN SITU by adding offset of new function relative to old one (0x2000 - 0x10) plus offset added by insertion of call to 'traceme'.

Example 3

```
[0x10] int foo(int a, int b) {
     int x = a + b;
     return a>b?x:b;
  }

[0x10] int foo(int a, int b) {
     tail_call DYNINST_foo;
  }

[0x1000] int DYNINST_foo(int a, int b) {
     int x = a + b;
     return a>b?x:b;
  }
```

Here, we need to update the location list for x.

Timeline

handle rewritten binaries first!

Short Term: **Don't crash**
    Update ELF EH table

Next:
    Update line map table
    Update DWARF CFI

Long Term:
    Update callsite entries for DYNINST_foo calling bar. Name, location, callsite parameter, and others. Here be dragons with GNU_callsite vs callsite.
    Add variables

Long long term (nice to have):
    Add functions using BPatch AST
    Dynamic instrumentation adds difficulties we don't want to handle yet. Could require interaction with glibc to get exception handling correct.
    Create new callsite information (e.g., adding call to 'traceme').

Ben Woodard
Tim Haines``
Bolo
Jim
Jonathon Anderson anderson.jonathonm@gmail.com
Cedric Valensi cedric.valensi@gmail.com

Highlights:
First, don’t crash.
Later, do things to make debugging easier.
Later later, do things that make Dyninst’s new information present.

Action items:

1. [Ben] Convert operations into a possible libdw ideas
2. [Tim] Make a real example with data dumps from DWARF/ELF

Using yaml2obj to make a DWARF writer was a 2020 GSoC
https://www.llvm.org/OpenProjects.html#llvm_dwarf_yaml2obj