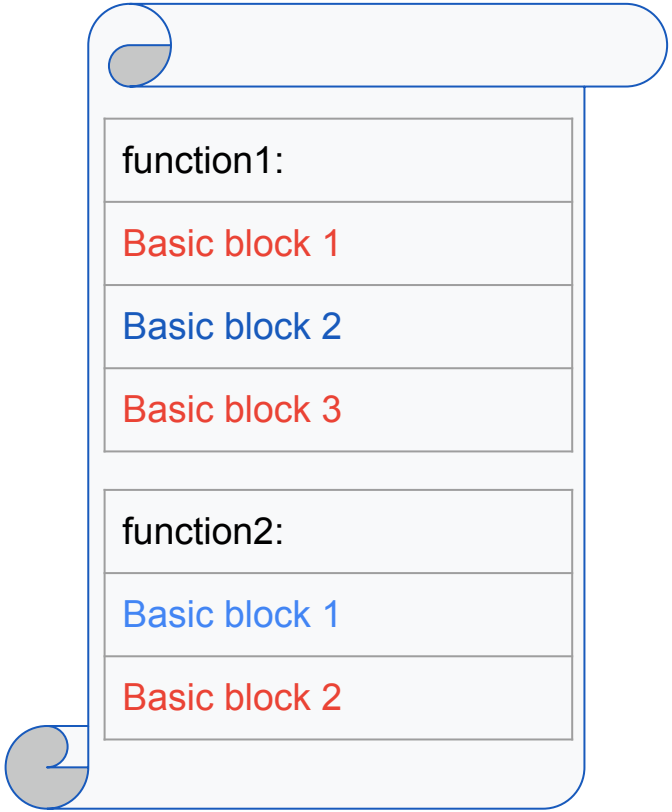
Google

LLDB support for Propeller optimized code

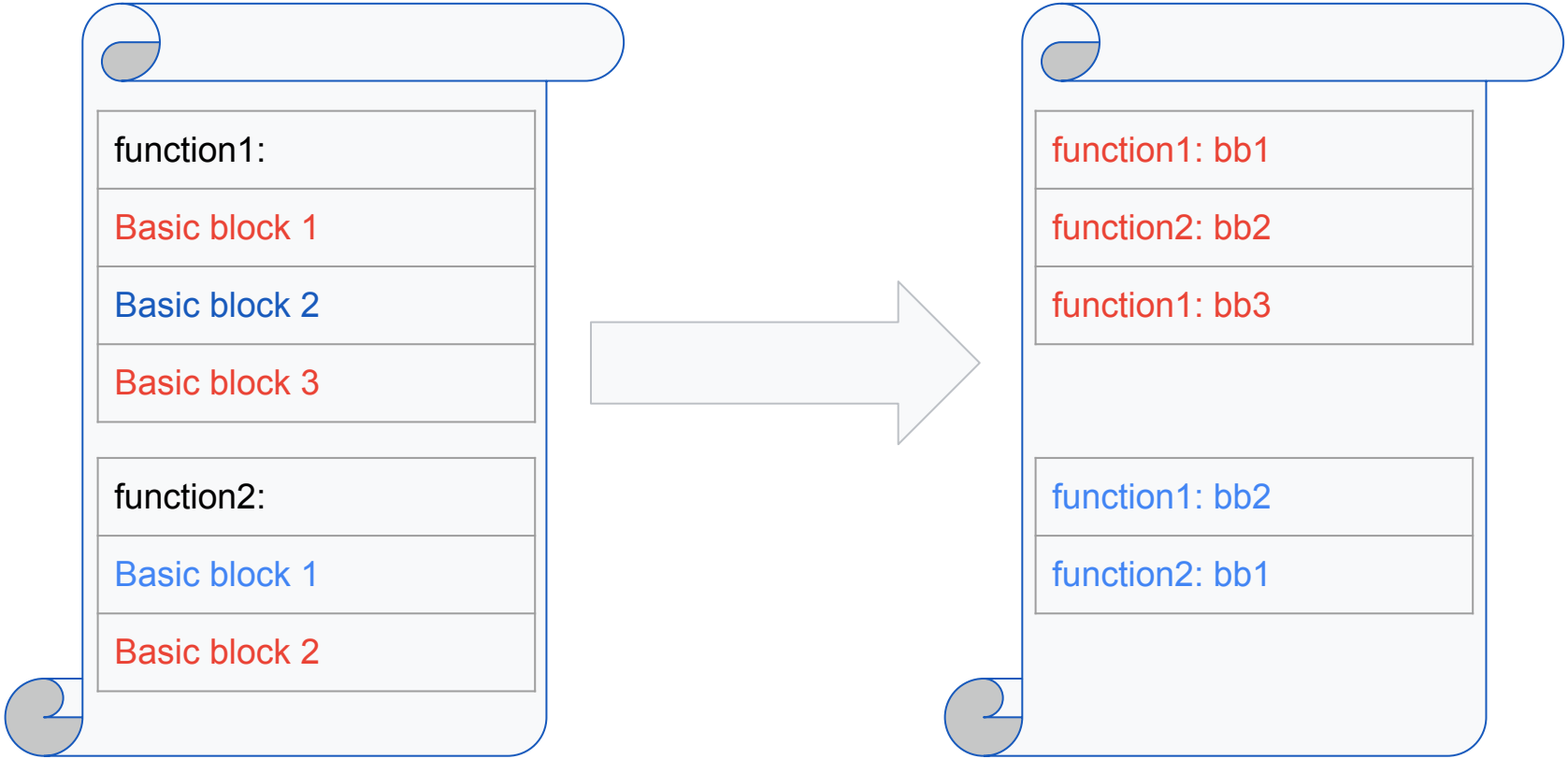
(things programmers believe about functions)

Pavel Labath

Propeller (Bolt, etc.)



Propeller (Bolt, etc.)



Debug info (DWARF)

```
DW_TAG_subprogram
    DW_AT_name    ("function1")
    DW_AT_low_pc  (0x00001000)
    DW_AT_high_pc
(0x00001030)
    ...
DW_TAG_subprogram
    DW_AT_name    ("function2")
    DW_AT_low_pc  (0x00002000)
    DW_AT_high_pc
(0x00002020)
    ...
```

Debug info (DWARF)

```
DW_TAG_subprogram
    DW_AT_name    ("function1")
    DW_AT_low_pc  (0x00001000)
    DW_AT_high_pc
(0x00001030)
    ...
DW_TAG_subprogram
    DW_AT_name    ("function2")
    DW_AT_low_pc  (0x00002000)
    DW_AT_high_pc
(0x00002020)
    ...
```

```
DW_TAG_subprogram
    DW_AT_name    ("function1")
    DW_AT_ranges  (
        [0x00001000, 0x00001010)
        [0x00002000, 0x00002010)
        [0x00001020, 0x00001030))
    ...
DW_TAG_subprogram
    DW_AT_name    ("function2")
    DW_AT_ranges  (
        [0x00002010, 0x00002020)
        [0x00001010, 0x00001020))
    ...
```

The problem

```
class Function {
    const AddressRange &GetAddressRange() { return m_range; }
    ...
};

function1.GetAddressRange() = [0x1000, 0x2010)
function2.GetAddressRange() = [0x1010, 0x2020)
```

The problem

```
class Function {
    const AddressRange &GetAddressRange() { return m_range; }
    ...
};

function1.GetAddressRange() =  [0x1000, 0x2010)
function2.GetAddressRange() =  [0x1010, 0x2020)

function1.GetAddressRange().GetBaseAddress() =  0x1000
function2.GetAddressRange().GetBaseAddress() =  0x1010
```

The problem

```
class Function {
    const AddressRange &GetAddressRange() { return m_range; }
    ...
};

function1.GetAddressRange() = [0x1000, 0x2010)
function2.GetAddressRange() = [0x1010, 0x2020)

function1.GetAddressRange().GetBaseAddress() = 0x1000
function2.GetAddressRange().GetBaseAddress() = 0x1010

class SymbolContext {
    bool GetAddressRange(uint32_t scope, uint32_t range_idx,
        bool use_inline_block_range, AddressRange &range) const;
};
```

Solution

```
class Function {
    AddressRanges GetAddressRanges() { return m_block.GetRanges(); }
    const Address &GetAddress() const { return m_address; }
};


class SymbolContext {
    bool GetAddressRange(uint32_t scope, uint32_t range_idx,
        bool use_inline_block_range, AddressRange &range) const;
    Address GetFunctionOrSymbolAddress() const;
};
```

Recommendations

- Do not assume that functions are contiguous

Recommendations

- Do not assume that functions are contiguous
- Do not assume that all parts (address ranges) of the function are within a single section
  - ```
    if (addr1.GetSection() == addr2.GetSection())
        return addr1.GetOffset() - addr2.GetOffset();
    ```
  - ```
    if (addr1.GetModule() == addr2.GetModule())
        return addr1.GetFileAddress() - addr2.GetFileAddress();
    ```

Recommendations

- Do not assume that functions are contiguous
- Do not assume that all parts (address ranges) of the function are within a single section
  - ```
    if (addr1.GetSection() == addr2.GetSection())
        return addr1.GetOffset() - addr2.GetOffset();
    ```
  - ```
    if (addr1.GetModule() == addr2.GetModule())
        return addr1.GetFileAddress() - addr2.GetFileAddress();
    ```
- Do not assume that a function is described by a single `eh_frame` record (line table sequence, etc.)

Recommendations

- Do not assume that functions are contiguous
- Do not assume that all parts (address ranges) of the function are within a single section
  - ```
    if (addr1.GetSection() == addr2.GetSection())
        return addr1.GetOffset() - addr2.GetOffset();
    ```
  - ```
    if (addr1.GetModule() == addr2.GetModule())
        return addr1.GetFileAddress() - addr2.GetFileAddress();
    ```
- Do not assume that a function is described by a single `eh_frame` record (line table sequence, etc.)
- Do not assume that function entry point is its lowest address

Recommendations

- Do not assume that functions are contiguous
- Do not assume that all parts (address ranges) of the function are within a single section
  - ```
    if (addr1.GetSection() == addr2.GetSection())
        return addr1.GetOffset() - addr2.GetOffset();
    ```
  - ```
    if (addr1.GetModule() == addr2.GetModule())
        return addr1.GetFileAddress() - addr2.GetFileAddress();
    ```
- Do not assume that a function is described by a single `eh_frame` record (line table sequence, etc.)
- Do not assume that function entry point is its lowest address
  - Corollary: Do not assume that offsets from the entry point are positive

Current state

- Most things “just work”
- Main exception: unwinding

Current state

- Most things “just work”
- Main exception: unwinding
- There are rough edges:

```
(lldb) disassemble --name foo
my_binary`foo:
0xdead00 <-1179648>: cmpl   $0x0, %eax
0xdead03 <-1179645>: jmp    0xdead33        ; <-1179597>
...
```

Thank you