Wasm指令到源码的映射

2021年9月8日 王纪开

概述

- Source map是由承载Debug info的DWARF信息生成。emcc自身也是使用clang-dwarfdump得到文本格式的DWARF信息再通过python正则解析得到source map，在工程上并不是高效的解决办法。因此不如直接解析dwarf信息，而且可能还可以获得更多信息用于之后的其他功能。
- 由于Wasm的函数体并不在内存空间内，因此函数的指令并没有对应的地址，DWARF（source map）中给出的是源码和WASM文件中，指令相对于Code段的偏移的对应关系。这个偏移并不能很容易地转换为wat文件的行号。

目录

- emscripten生成source map过程
- code offset在实现时的问题

DWARF

- DWARF是有比较完善的标准的Debug info格式。
- 独立于可执行文件的封装格式，作为字节流封装在自定义段。
- 在wasm文件中作为custom section
- https://webassembly.github.io/spec/core/binary/modules.html#binary-customsec

Custom Section

- 目前只包含在二进制文件中，WAT格式中还没有对应的表示形式，转为WAT时会被丢弃
- 相关的表示方法还没有纳入标准：
- https://github.com/WebAssembly/proposals/issues/13
- https://github.com/WebAssembly/design/issues/1153 相关讨论

emcc生成source map

- `emcc ./rot13.c -g -gsource-map -o r.wasm`
- 这条命令背后：
  - 首先生成带dwarf的wasm文件
  - `llvm-dwarfdump -debug-info -debug-line --recurse-depth=0`
  - `python3 emsdk/upstream/emscripten/tools/wasm-sourcemap.py` 解析上个命令的stdout

llvm-dwarfdump

```
.debug_line contents:
debug_line[0x00000000]
Line table prologue:
    total_length: 0x000000fa
          format: DWARF32
         version: 4
 prologue_length: 0x0000001f
 min_inst_length: 1
max_ops_per_inst: 1
 default_is_stmt: 1
       line_base: -5
      line_range: 14
     opcode_base: 13
standard_opcode_lengths[DW_LNS_copy] = 0
standard_opcode_lengths[DW_LNS_advance_pc] = 1
standard_opcode_lengths[DW_LNS_advance_line] = 1
standard_opcode_lengths[DW_LNS_set_file] = 1
standard_opcode_lengths[DW_LNS_set_column] = 1
standard_opcode_lengths[DW_LNS_negate_stmt] = 0
standard_opcode_lengths[DW_LNS_set_basic_block] = 0
standard_opcode_lengths[DW_LNS_const_add_pc] = 0
standard_opcode_lengths[DW_LNS_fixed_advance_pc] = 1
standard_opcode_lengths[DW_LNS_set_prologue_end] = 0
standard_opcode_lengths[DW_LNS_set_epilogue_begin] = 0
standard_opcode_lengths[DW_LNS_set_isa] = 1
file_names[  1]:
           name: "rot13.c"
      dir_index: 0
       mod_time: 0x00000000
         length: 0x00000000

Address            Line   Column File   ISA Discriminator Flags
------------------ ------ ------ ------ --- ------------- -------------
0x0000000000000008      3      0      1   0             0  is_stmt
0x0000000000000081      6      9      1   0             0  is_stmt prologue_end
0x000000000000008a      6      7      1   0             0
0x0000000000000091      7      9      1   0             0  is_stmt
0x0000000000000098      7     11      1   0             0
0x00000000000000ab      7      9      1   0             0
0x00000000000000bd      8      7      1   0             0  is_stmt
0x00000000000000c5     10      9      1   0             0  is_stmt
0x00000000000000cc     10     11      1   0             0
0x00000000000000e0     10     18      1   0             0
0x00000000000000f2     10     21      1   0             0
0x00000000000000f9     10     23      1   0             0
0x000000000000010d     10      9      1   0             0
0x000000000000011d     11      9      1   0             0  is_stmt
0x0000000000000136     12     11      1   0             0  is_stmt
0x000000000000013d     12     13      1   0             0
0x0000000000000151     12     11      1   0             0
0x0000000000000163     13     11      1   0             0  is_stmt
0x000000000000017e      5      3      1   0             0  is_stmt end_sequence

```

Source Map V3

- JSON格式


```
{
    "version": 3,
    "sources": [
        "rot13.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/libc/crt1.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/standalone/__original_main.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/standalone/__main_void.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/standalone/__main_argc_argv.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/libc/musl/src/exit/_Exit.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/libc/musl/src/exit/exit.c",
        "../../../../../../../b/s/w/ir/k/install/emscripten/system/lib/libc/musl/src/errno/__errno_location.c"
    ],
    "names": [],
    "mappings": "yZAEA,2CAGQ,SAAF,OACE,OAAE,mBAAF,kBACF,QAEE,OAAE,oBAAO,kBAAG,OAAE,oBAAd,gBACA,yBACE,OAAE,oBAAF,kBACA,2
}
```

wasm-sourcemap.py mappings的生成

```
304       address_delta = address - last_address
305       source_id_delta = source_id - last_source_id
306       line_delta = line - last_line
307       column_delta = column - last_column
308       mappings.append(encode_vlq(address_delta) + encode_vlq(source_id_delta) + encode_vlq(line_delta) + encode_vlq(column_delta))
309       last_address = address
310       last_source_id = source_id
311       last_line = line
312       last_column = column
313     return OrderedDict([('version', 3),
314                         ('names', [])
```

```
265     for entry in entries:
266       line = entry['line']
267       column = entry['column']
268       # ignore entries with line 0
269       if line == 0:
270         continue
271       # start at least at column 1
272       if column == 0:
273         column = 1
274       address = entry['address'] + code_section_offset
275       file_name = entry['file']
```

总结

- emcc它生成source map的过程似乎并不高效
- 考虑直接解析dwarf信息：
  - 载入wasm文件时，引入python库解析dwarf信息保存映射关系，在报错时查询使用。
- pyelftools说它特意将dwarf解析部分和elf解析部分分离开了
- https://yurydelendik.github.io/webassembly-dwarf/ webassembly-dwarf标准

Code Offset

- 由于Wasm的函数体并不在内存空间内，因此函数的指令并没有对应的地址。
- 因此DWARF（source map）中给出的是源码和WASM文件中，指令相对于Code段的偏移的对应关系。加上code段相对于文件的偏移可以得到指令相对于整个文件的偏移。
- 如果需要获取对应wat文件的行号，需要考虑和disassembler的交互