Verifying Code Generation is unaffected by -g/-S

Russell Gallop

Senior Software Engineer in Test

Problem

- PlayStation®4 (PS4) developers make extensive use of debug info and assembly code
  - Assume `-g` generates debug info without side effects on code gen
  - Assume that compiling direct to an object file is the same as compiling to `.s` then assembling
- But is this true?

Check CFC (Compile Flow Consistency)

- We have implemented a checker for this as a compiler wrapper:
  - `cfe/utils/check_cfc/check_cfc.py`
- Rename to `clang` and add to PATH before the real `clang`:

```
cp check_cfc.py clang
cp check_cfc.py clang++
export PATH=<path to check_cfc>:$PATH
```

- Then use as if it's the compiler
- Intercepts `-c` commands

Check CFC (Compile Flow Consistency)

- Runs:
  - `clang -c` **`<args>`** `-o a.o`
  - `clang -c` **`<modified args>`** `-o b.o`
  - Compare `objdump -d` of `a.o` and `b.o`
- Returns non-zero if
  - The modified compile fails
  - The comparison fails
- Easy to integrate into build systems

Example (PR21807)

```
$ cat test.cpp
char a;
struct C { int f(char ,char ,char ,...); };
void foo(){ C c;  char lc = a; c.f(0,a,0,lc); c.f(0,a,0,lc); }
$ clang -O2 -c test.cpp
Check CFC, checking: dash_g_no_change
Code difference detected due to using -g
--- /tmp/tmpTKVDdi.o
+++ /tmp/tmpwWlqII.o
   14:   31 c9              xor     %ecx,%ecx
   16:   31 c0              xor     %eax,%eax
   18:   4c 89 f7           mov     %r14,%rdi
-  1b:   89 da              mov     %ebx,%edx
-  1d:   41 89 d8           mov     %ebx,%r8d
+  1b:   41 89 d8           mov     %ebx,%r8d
+  1e:   89 da              mov     %ebx,%edx
```

Example (PR23098)

```
$ cat test.c
int a; void fn1() { a = a << 1 & 255; }
$ clang -c test.c
Check CFC, checking: dash_s_no_change
Code difference detected due to using -S
--- /tmp/tmptzxZed.o
+++ /tmp/tmp6Vwjnc.o
    0:  55                          push   %rbp
    1:  48 89 e5                    mov    %rsp,%rbp
    4:  8b 04 25 00 00 00 00        mov    0x0,%eax
-   b:  c1 e0 01                    shl    $0x1,%eax
+   b:  d1 e0                       shl    %eax
```

Results

- Ran our regression tests with Check CFC
- dash_g_no_change
  - Bugs found in peephole optimizer, branch folding, machine scheduler
  - 18590, 19051, 21807 (all fixed)
- dash_s_no_change
  - Found bugs in Isel, FastISel
  - 22854, 22995, 23098 (all fixed)

Summary

- Simple method of testing user expectations
- Finds subtle bugs across large parts of the compiler
- Future work
  - Testing Intel vs AT&T x86 asm syntax
  - Separating preprocess and compile steps
  - Comparison of debug information and data
- Please try it out

- Poster afterwards