Compilers 12/11/22


**SIT Compilation**

- "just-in-time"

**Delay Compilation**

- Maybe don't need to compile
- Maybe: Take advantage of runtime info

**Units of Compilation**

- Functions
- Traces

**Runtime:**

- Compiler! Produces machine bits directly into memory
- Source code (or SR)

Start by compiling `main` (only!)

```
main()

which = 1

main:

"main": ...

CALL F = F

CALL F = F

3 ...

3 RET "TRAP POLYE"

F = 1: Compiles F and puts code at address F
2. Overwrites its own code to jump to F
2a. Overwrite our call
3. Jump to F
```

---

**Interpreter + SIT Compiler**

- Start by interpreting `main`, etc.
- Keep track of execution counts
- When a function becomes "hot" we compile it
**TRACING JIT**

\[
f(\_I) 
\]

\[
\text{while } (...) \text{ ] -> TRAIC }\]

\[
\text{speculative} \quad \text{optimization} \]

\[
f(\_I) \quad f(x) \frac{1}{2} \]

\[
\begin{aligned}
\text{if } x = 42 \\
\text{call } g
\end{aligned}
\]

\[
\text{must deal with mis-speculation} \quad \text{guard } (x = 42) \rightarrow \text{bad out}
\]

\[
\text{on-state replacement}
\]