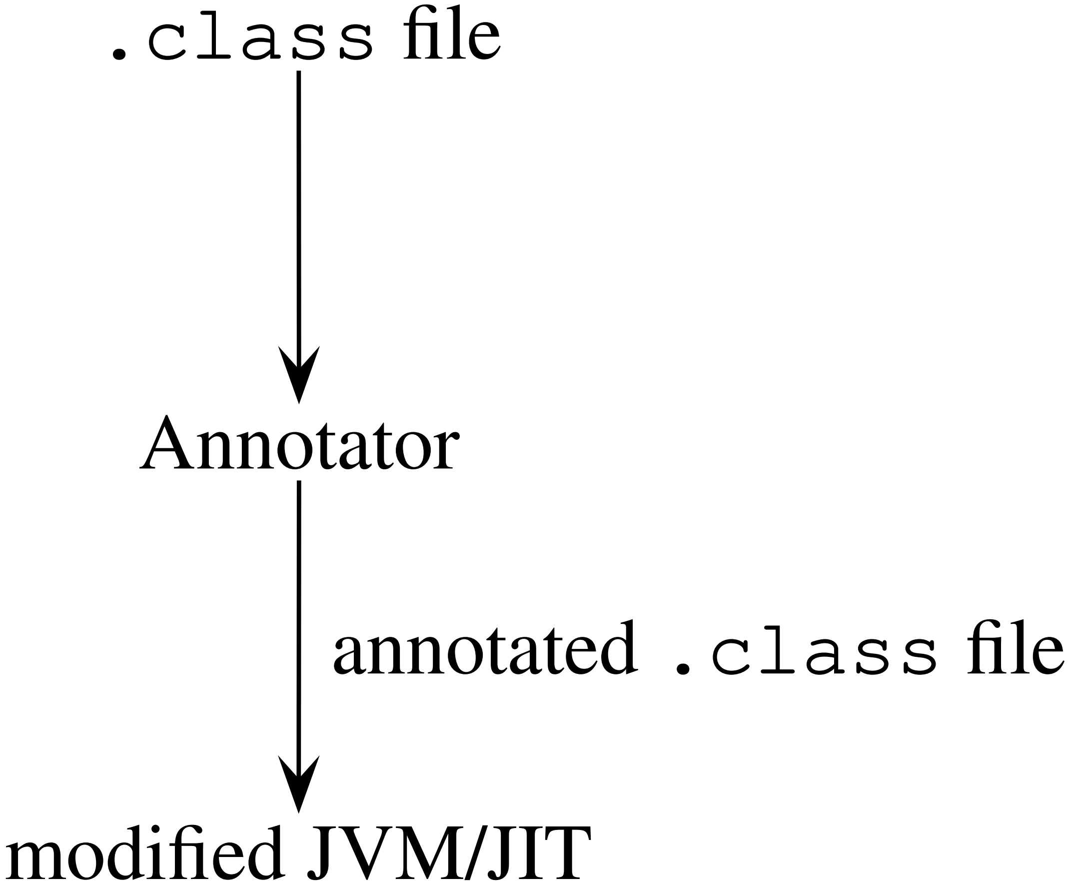
Joel Jones (jjones@uiuc.edu)
Samuel Kamin

Department of Computer Science
University of Illinois at Urbana-Champaign
http://mtdoom.cs.uiuc.edu/PressPot.html

Annotation Process



The Annotations

VRs Virtual register assignment on the bytecode level

swaps Compact representation of spill values and locations. Indicates VR priority change or "swap."

Superlinear Analysis / Linear Exploitation: Register Allocation

- Superlinear Server — Analysis:

  Reachability Analysis $O(N \cdot A)$ — N is number of basic blocks and A is maximal number of back edges

  Liveness Analysis like reachability

  Loop Detection $O(E \lg N)$ — E is number of edges and N is number of basic blocks

  Graph Coloring Register Allocator allocation is $O(N \ \lg \ N)$ in number of nodes in interference graph (Briggs)

- Linear Client — Exploitation:

  Linear Verification linear addition to normal verification

  Linear Assignment $O(n)$ in number of VRs

Verifiable:
Ensure Inviolability of the JVM

VRs During standard verification, type of each VR is checked.

swaps Swaps constrained to be between values of the same type.

Portable:
Any JVM, Any Register Model

Ignorable annotations to `.class` file must be safely ignored

Independent VRs are independent of the size and structure of register files

Compact encoding of swaps doesn't favor machines with large number of registers

Flexible swaps indicates *potential* spills, not *required* ones

Optimized:
Improves Even Small Problems

problem size		`.class` file size	code gen time	execution time
2	w/o VRs	1252	11ms	21ms
	w VRs	1645	13ms	19.7ms
10	w/o VRs	1800	27ms	47.7ms
	w VRs	2050	18ms	37.7ms

Results from program which conditionally assigns the contents of array to temporary variables, then sums the temporary variables. (Chaitin) (Sparcstation I, in 1000 loop iterations)