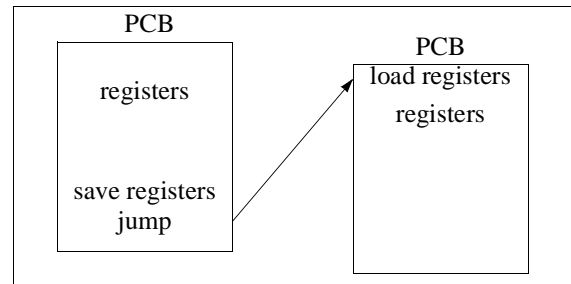


CS 518

The Synthesis Kernel: An Exercise in Minimalism

Randy Wang
Fall, 2001
Princeton University

Principle of Frugality



- How to get absolute maximum speedup?
 - By doing absolute minimum work
- Code synthesis
 - Generate tailor-made code on the fly
- “Executable data structure”
 - Embed code in data structures to avoid data structure traversals
- Code specialization
 - Special-casing and optimizing for common case

CS518

1

Randy Wang

Questions

- Does this approach generalize to a big real OS?
 - Assembly language as “fast prototyping language”
 - Self-changing code
 - Limited functionality
- How much does this matter for real applications?
 - Kernel primitives have impressive performance
 - How much does this translate to app performance?
 - System was never complete enough to run real benchmarks
- Implications
 - Bounding performance
 - The minimalism principle revisited many more times
 - Clean abstraction/framework is the key to make it usable
- Prelude to Exokernel
 - Specialized code
 - Dynamic code generation
 - Minimalist context switches
 - How do I do all this securely for arbitrary users?

CS518

2

Randy Wang