Program Analysis for Embedded Systems

Alex Aiken

A Confession

• I am here under false pretenses

• I know
  - little about embedded systems
  - a bit about programming languages and compilers

• This talk is all questions, no answers
How is Embedded Software Different from Ordinary Software?

• It has to work

• One or more (very) limited resources
  - Registers
  - RAM
  - Bandwidth
  - Time

Devil's Advocate

• So what's different?

• All software works with limited resources

• We have compiler technology to deal with it
  - Various forms of program analysis
Example: Registers

- All machines have only a few registers
- Compiler uses the registers as best it can
  - Spills the remaining values to main memory
  - Manages transfers to and from registers
- The programmer feels she has $\infty$ registers

The Standard Trick

- This idea generalizes
- For scarce resource $X$
  - Manage $X$ as best we can
  - If we need more, fall back to secondary strategy
  - Give the programmer a nice abstraction
The Standard Trick

• This idea generalizes

• For scarce resource X
  - Manage X as best we can
    - Any correct heuristic is OK, no matter how complex
  - If we need more, fall back to secondary strategy
    - Focus on average case behavior
  - Give the programmer a nice abstraction

Examples of the Standard Trick

• Compilers
  - Register allocation
  - Dynamic memory management

• OS
  - Virtual memory
  - Caches

Summary: abstract and hide complexity of resources
What's Wrong with This?

- Embedded systems have limited resources

- Meaning hard limits
  - Cannot use more time
  - Cannot use more registers

- The compiler must either
  - Produce code within these limits
  - Report failure

- The standard trick is anathema to embedded systems
  - Can't hide resources

Revisiting the Assumptions

- Any correct heuristic is OK, no matter how complex
  - Embedded programmer must understand reasons for failure
  - Feedback must be relatively straightforward

- Focus on average case behavior
  - Embedded compiler must reason about the worst case
  - Cannot improve average case at expense of worst case

- Give the programmer a nice abstraction
  - Still need abstractions, but likely different ones
Questions

• What are good models for programming with resource constraints?
  - Made explicit in the programming language

• How do we combine standard and embedded programming models?
  - Would like to build on what we know and have

• How do we give programmers good feedback?