Some project proposals

Peter Sestoft

2008-04-14

Compiling MicroC to x86 code

- Idea: Compile MicroC to x86 code
- Topics:
  - Understand the x86 instruction set
  - Decide on runtime (stack) layout
  - Design new Compile methods for MicroC
  - Generate x86 code as text input to nasm
- Challenges:
  - Use extreme care, or spend May debugging
  - Once it works, make it run as fast as possible

Extend MicroC compiler

- Idea: Learn more about C and compilation by covering more of C
- Topics:
  - C features, such as struct types, switch, break, continue, malloc/free, fct pointers
  - Design compile-time type representation and run-time data represenation
  - Modify parser, checker, compiler, (machine)
  - Write example programs and test
- Challenges:
  - Investigate compiler literature and transfer ideas to MicroC compiler

Icon (or SETL?) interpreter

- Idea: Icon interpreter in Java or C#
- Topics
  - The Icon language, with backtracking; e.g. (1 to 3) * (4 to 5) gives 4 5 8 10 12 15
  - Continuations – very cool technique
  - Implement parser, optional
  - Implement interpreter using continuations
  - Write and run Icon examples
  - See Programming Language Concepts section 8.9
- Challenges
  - Understand continuations
  - Consult http://www.cs.arizona.edu/icon/
  - Find or invent clever Icon programs
  - (If in .NET, using tail. prefix on calls)

Code duplication discovery in Microsoft Dynamics NAV (Navision)

- Idea: Find common patterns in NAV source code
- Topics:
  - Understand Michael Schou Christensen's CodeDup
  - Design C/AL (NAV) grammar from code examples
  - Adaptation to Coco/R and testing on code examples
  - Build on CodeDup
  - Empirical study of large C/AL source base
  - Collaboration with Microsoft Dev Center under NDA
  - Possibly: Propose refactorings – MSc thesis?
- Challenges:
  - Understand and reverse engineer C/AL
  - What CodeDup parameters (constants,…) work?

Performance engineering

- Idea:
  - Tune software for modern platforms: .NET/JVM and x86-processors
  - Improve *Numeric performance in C, C# and Java*
- Topics
  - Consider e.g. matrix computations, or the C5 collection library, and large problem sizes
  - Measure time and space performance
  - Compare platforms (.NET vs JVM, hardware)
  - Rewrite code for better performance
- Challenges
  - Find good profiling tools; be systematic
  - Draw general conclusions

Fast serialization using runtime code generation

- Idea: To serialize array of 100 000 structs, use reflection once to generate serialization code, then run that code 100 000 times
- Topics
  - Reflection to discover a type's components
  - Generating .NET or JVM code at runtime
  - Benchmarking
- Challenges
  - Understand .NET or Java bytecode in detail
  - Debugging the serialization code generator
  - Cover as much of C#/Java as possible