CSE 401
Intro Compilers

Final Review

(post-Midterm)

Larry Snyder
Spring 2003

Interpret vs. compile

- Tradeoffs
- Run-time and compile-time
- Advantages of one over the other
- Basic structure of an interpreter

Jobs of a compiler (backend)

- Representation and placement of run-time values
- Generate machine code
- Optimization

Compile- vs Run-Time

- procedures vs activation record/stack frame
- scope vs environment
- symbol table vs stack frame
- variable vs memory/stack/register location
- lexically enclosing scope vs static link
- caller vs dynamic link

Run Time Storage

- Representation of data - scalars, aggregates
- memory areas: static, stack (lifo), heap
- layout of stack frame: formals, locals, links, etc.
- calling conventions – handling registers, return values, etc.
- parameter passing modes:
  call-by-value vs call-by-reference vs ...

Parameter passing

- Call-by-value, call-by-reference, etc.
- The mechanisms
- The consequences of the mechanisms on programming language design and on programs

Intermediate Code Gen

- Structure of code generation, and benefits of that structure
- Intermediate vs. target code generation (temps, machine (in)dependence, ...)
- 3-address code: what and why
- Generation of IR from AST:
  l- vs r-value, exprs, assign, arrays, ...
- Short circuit code

7

Target Code Gen

- Instruction selection (RISC/CISC)
- Register allocation
- Impact of basic architectural features

8

Optimization

- Deduce as much as possible at compile time about run time bindings, values, control flow,...
- Use it to:
  - Simplify/specialize unnecessarily general code
  - Reorder code
  - Exploit target machine
- Scope:
  - Peephole
  - Local
  - Global (intra-procedural)
  - Inter-procedural
- Examples

harder ↓ better

9

Activation records

- Distinguish from symbol tables
- What goes in them
- Static/dynamic links
  - What they are, why they are, and how they are managed

10

Implementation of optimization

- Analyses
  - live variable analysis
- Control and data flow graph representations
  - What and why
- Iterative dataflow analysis

11