

CS 471 Test 2 Study Guide

Environments and Symbol Tables

- Implementation options
- Functional vs. imperative environments
- Tiger Implementation
 - Symbol table
 - Enter/exit scope
 - Environments

Type Checking

- Tiger implementation
 - Tenv vs. venv
 - transTy
 - transVar
 - transExp
 - transDec
 - recursive type declarations

Activation Records / Stack Frames

- Invocations vs. instantiations
- Stack pointer vs. frame pointer
- Address space: stack, heap, static data, code
 - Stack allocation vs. heap allocation vs. static allocation
- Handling local variables
 - Cannot use a stack for higher-order functions
 - Can use stack for nested functions
- Handling nested procedures
- Passing parameters
 - why we need calling conventions
 - register windows
- Calling sequence
- Caller save vs. callee save
 - Optimizations, leaf procedures
- Passing return values
- Return address – why we need it, when we store it
- Static link vs. dynamic link – why we need each
- Registers vs. memory – tradeoffs, why compilers care
- Tiger Frames
 - Frame and temp interfaces

Intermediate Representations

- Reasons for an IR
- What makes a good IR
- Components of an IR
- Designing an IR
- Three-address code
 - Shorthand: triples, quadruples
 - Converting sample code into three-address code
- Expression trees
 - Converting sample code into an expression tree

- The IR machine
- Multiple IR stages (HIR, MIR, LIR) – features, motivation, examples
- Graphical vs. linear IR
 - DAGs
 - Dependence graphs
- Stack machine code
- Tiger IR
 - Expressions
 - Statements
 - Translating (to IR): if, while, return
- Canonical IR
 - Properties, motivation
 - converting to canonical form
 - introducing temps, handling returns, flattening, cjmps

Basic Blocks and Traces

- Partitioning code into basic blocks
- Ordering basic blocks
- Trace creation – heuristics, benefits

Instruction selection

- Tiling
 - Designing tiles
 - Selecting tiles – optimal vs. optimum
- Strategies for fixed regs
- Maximal munch
 - Benefits and problems
- Dynamic programming
 - Cost model, problems
- Generating code for function calls, function bodies
 - Prologue vs. epilogue
- Generating code for dynamic structures

Register Allocation

- Live ranges
- Gen, kill, def, use, live, dead, webs
- Interference and interference graphs
- Graph coloring
- Spilling, spill costs
- Splitting and heuristics
- Other regalloc optimizations
 - Register coalescing
 - Register targeting
 - Presplitting
 - Interprocedural regalloc