Basic Blocks and Traces
CS 352
Basic blocks and Traces

Issues:

- To simplify translation there are mismatches between tree code and actual machine instructions:
  1. CJUMP to two labels; machine conditionals fall through on false
  2. ESEQ and CALL order evaluation of subtrees for side-effects – constrains optimization
  3. CALL as argument to another CALL causes interference between register arguments

- Can rewrite equivalent trees without these cases:
  - SEQ can only be subtree of another SEQ
  - SEQs clustered at top of tree
  - might as well turn into simple linear list of statements

- 3-stage transformation:
  1. to linear list of *canonical trees* without SEQ/ESEQ
  2. to *basic blocks* with no internal jumps or labels
  3. to *traces* with every CJUMP immediately followed by false target
Canonical trees

1. No SEQ or ESEQ
2. CALL can only be subtree of EXP(...) or MOVE(TEMP t,...)

Transformations:
- lift ESEQs up tree until they can become SEQs
- turn SEQs into linear list
\begin{align*}
\text{ESEQ}(s_1, \text{ESEQ}(s_2, e)) &= \text{ESEQ}(\text{SEQ}(s_1, s_2), e) \\
\text{BINOP}(\text{op}, \text{ESEQ}(s, e_1), e_2) &= \text{ESEQ}(s, \text{BINOP}(\text{op}, e_1, e_2)) \\
\text{MEM}(\text{ESEQ}(s, e_1)) &= \text{ESEQ}(s, \text{MEM}(e_1)) \\
\text{JUMP}(\text{ESEQ}(s, e_1)) &= \text{SEQ}(s, \text{JUMP}(e_1)) \\
\text{CJUMP}(\text{op}, \text{ESEQ}(s, e_1), e_2, l_1, l_2) &= \text{SEQ}(s, \text{CJUMP}(\text{op}, e_1, e_2, l_1, l_2)) \\
\text{BINOP}(\text{op}, e_1, \text{ESEQ}(s, e_2)) &= \text{ESEQ}(\text{MOVE}(\text{TEMP} \ t, e_1), \\
& \quad \text{ESEQ}(s, \text{BINOP}(\text{op}, \text{TEMP} \ t, e_2))) \\
\text{CJUMP}(\text{op}, e_1, \text{ESEQ}(s, e_2), l_1, l_2) &= \text{SEQ}(\text{MOVE}(\text{TEMP} \ t, e_1), \\
& \quad \text{SEQ}(s, \text{CJUMP}(\text{op}, \text{TEMP} \ t, e_2, l_1, l_2))) \\
\text{MOVE}(\text{ESEQ}(s, e_1), e_2) &= \text{SEQ}(s, \text{MOVE}(e_1, e_2)) \\
\text{CALL}(f, a) &= \text{ESEQ}(\text{MOVE}(\text{TEMP} \ t, \text{CALL}(f, a)), \text{TEMP}(t))
\end{align*}
Taming conditional branches

1. Form basic blocks: sequence of statements always entered at the beginning and exited at the end:
   - first statement is a LABEL
   - last statement is a JUMP or CJUMP
   - contains no other LABELs, JUMPS or CJUMPs

2. Order blocks into trace:
   - every CJUMP followed by false target
   - JUMPs followed by target, if possible, to eliminate JUMP
Basic blocks

*Control flow analysis* discovers basic blocks and control flow between them:

1. scan from beginning to end:
   - LABEL \( l \) starts a new block and previous block ends (append JUMP \( l \) if necessary)
   - JUMP or CJUMP ends a block and starts next block (prepend new LABEL if necessary)

2. prepend new LABELs to blocks with non-LABEL at beginning

3. append JUMP(NAME done) to last block
Traces

1. Pick an untraced block, the start of some trace
2. Follow a possible execution path, choosing false targets first
3. Repeat until all blocks are traced

Cleaning up:

- CJUMP followed by true target: switch targets, negate condition
- CJUMP\((o, a, b, l_t, l_f)\) followed by neither \(l_t\) nor \(l_f\):
  1. create new \(l'_f\)
  2. rewrite as CJUMP\((o, a, b, l_t, l'_f)\), LABEL \(l'_f\), JUMP \(l_f\)
- JUMP \(l\), LABEL \(l \rightarrow LABEL \ l\)