MI-SP-SP-12 (MI-GEN) (lecture 7 and 8)

Alokace registrů v generátorech kódu. Lokální optimalizace generovaného kódu v rámci základního bloku a v rámci funkce.

Register allocation

- registers are much faster than memory
  → compiler uses huge amount of virtual registers → those must be mapped
- solution using graph coloring (NP-hard problem)
- three states of temporary variables:
  - unallocated - not assigned yet
  - live - allocated and will be used in the future
  - dead

→ two variables can share register only if they are not alive at the same moment

Register interference graph

- node for each variable - edge between those alive simultaneously
  → create from the sets of living at each point of the program

→ assign colors to registers → heuristic solution
- k-colorable graph - can be colored with k colors
  - remove nodes with less than k neigh.
  → go back through stack and assign colors

- if we don't have enough space → register spilling (save into memory with load and store)
  → recompute the interference graph
  → try again

- try to spill temporaries with the most conflicts & try to avoid spilling in inner loops

Linear scan reallocation - simpler, faster, used by JITs (VM)

- based on live ranges of variables
- all variables in registers → if one must be spilled, choose the one with longest live range
  - reserved registers for spilled variables

Optimizations

- performed over various levels of IR
  - local - inside BB
  - global - whole procedure
  - interprocedural - whole program

[machine dependent × independent]

- "premature optimization is the root of all evil"
- 80% of execution time is spent executing 20% of the code
- good algorithms FTW

General optimization techniques

- strength reduction - e.g. shifts instead of multiplication
- common sub-expression elimination - compute common things just once
- constant propagation & constant folding
- dead code elimination
- code motion - move invariants from loops above them
- loop unrolling - do more steps within one iteration (with bigger steps)
- algebraic identities
- elimination of useless instructions

+ peephole optimization techniques