

# CSE 582 – Compilers

---

Register Allocation  
Hal Perkins  
Autumn 2002

11/21/2002 © 2002 Hal Perkins & UW CSE P-1

---

---

---


---

---

---

---

---



# Agenda

- Register allocation constraints
- Top-down and bottom-up local allocation
- Global allocation – register coloring

Credits: Adapted from slides by Keith Cooper, Rice University

11/21/2002 © 2002 Hal Perkins & UW CSE P-2

---

---

---


---

---

---

---

---



# k

- Intermediate code typically assumes infinite number of registers
- Real machine has  $k$  registers available
- Goals
  - Produce correct code that uses  $k$  or fewer registers
  - Minimize added loads and stores
  - Minimize space needed for spilled values
  - Do this efficiently –  $O(n)$ ,  $O(n \log n)$ , maybe  $O(n^2)$

11/21/2002 © 2002 Hal Perkins & UW CSE P-3

---

---

---


---

---

---

---

---



## Register Allocation

- Task
  - At each point in the code, pick the values to keep in registers
  - Insert code to move values between registers and memory
    - No additional transformations – scheduling should have done its job
  - Minimize inserted code, both dynamically and statically

11/21/2002 © 2002 Hal Perkins & UW CSE P-4

---

---

---


---

---

---

---

---



## Allocation vs Assignment

- Allocation: deciding which values to keep in registers
- Assignment: choosing specific registers for values
- Compiler must do both

11/21/2002 © 2002 Hal Perkins & UW CSE P-5

---

---

---


---

---

---

---

---



## Basic Blocks

- A *basic block* is a maximal length segment of straight-line code (i.e., no branches)
- Significance
  - If any statement executes, they all execute
    - Barring exceptions or other unusual circumstances
  - Execution totally ordered
  - Many techniques for improving basic blocks – simplest and strongest methods

11/21/2002 © 2002 Hal Perkins & UW CSE P-6

---

---

---


---

---

---

---

---



## Local Register Allocation

- n Transformation on basic blocks
- n Produces decent register usage inside a block
  - n Need to be careful of inefficiencies at boundaries between blocks
- n Global register allocation can do better, but is more complex

11/21/2002 © 2002 Hal Perkins & UW CSE P-7

---

---

---


---

---

---

---

---



## Allocation Constraints

- n Allocator typically won't allocate all registers to values
- n Generally reserve some minimal set of registers  $F$  used only for spilling (i.e., don't dedicate to a particular value)

11/21/2002 © 2002 Hal Perkins & UW CSE P-8

---

---

---


---

---

---

---

---



## Liveness

- n A value is *live* between its *definition* and *use*.
  - n Find definitions ( $x = \dots$ ) and uses ( $\dots = x \dots$ )
  - n Live range is the interval from definition to last use
    - n Can represent live range as an interval  $[i,j]$  in the block

11/21/2002 © 2002 Hal Perkins & UW CSE P-9

---

---

---


---

---

---

---

---



## Top-Down Allocator

- n Idea
  - n Keep busiest values in a dedicated registers
  - n Use reserved set, F, for the rest
- n Algorithm
  - n Rank values by number of occurrences
  - n Allocate first k-F values to registers
  - n Add code to move other values between reserved registers and memory

11/21/2002 © 2002 Hal Perkins & UW CSE P-10

---

---

---


---

---

---

---

---



## Bottom-Up Allocator

- n Idea
  - n Focus on replacement rather than allocation
  - n Keep values used "soon" in registers
- n Algorithm
  - n Start with empty register set
  - n Load on demand
  - n When no register available, free one
- n Replacement
  - n Spill value whose next use is farthest in the future
  - n Prefer clean value to dirty value
  - n Sound familiar?

11/21/2002 © 2002 Hal Perkins & UW CSE P-11

---

---

---


---

---

---

---

---



## Bottom-Up Allocator

- n Invented about once per decade
  - n Sheldon Best, 1955, for Fortran I
  - n Laslo Belady, 1965, for analyzing paging algorithms
  - n William Harrison, 1975, ECS compiler work
  - n Chris Fraser, 1989, LCC compiler
  - n Vincenzo Liberatore, 1997, Rutgers
- n Will be reinvented again, no doubt
- n Many arguments for optimality of this

11/21/2002 © 2002 Hal Perkins & UW CSE P-12

---

---

---


---

---

---

---

---



## Global Register Allocation

- n A standard technique is *graph coloring*
- n Use control and dataflow graphs to derive *interference graph*
  - n Nodes are virtual registers (the infinite set)
  - n Edge between (t1,t2) when t1 and t2 cannot be assigned to the same register
    - n Most commonly, t1 and t2 are both live at the same time
    - n Can also use to express constraints about registers, etc.
- n Then color the nodes in the graph
  - n Two nodes connected by an edge may not have same color
  - n If more than k colors are needed, insert spill code

11/21/2002 © 2002 Hal Perkins & UW CSE P-13

---

---

---


---

---

---

---

---



## Coming Attractions

- n Dataflow and Control flow analysis
- n Overview of optimizations

11/21/2002 © 2002 Hal Perkins & UW CSE P-14

---

---

---

---

---

---

---

---