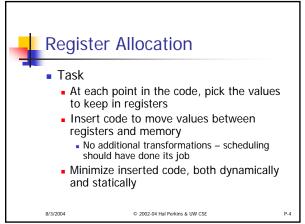
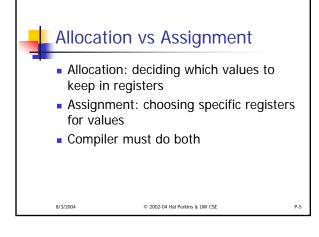
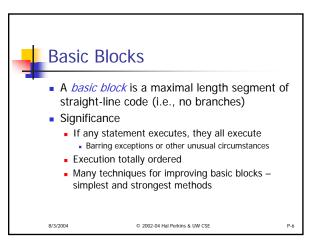


© 2002-04 Hal Perkins & UW CSE

8/3/2004







CSE P 501 Su04 P-1



## **Local Register Allocation**

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

/3/2004

© 2002-04 Hal Perkins & UW CSF



### **Allocation Constraints**

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

2004 © 2002-04 Hal Perkins & UW CSE



#### Liveness

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

8/3/2004

© 2002-04 Hal Perkins & UW CSE



## Top-Down Allocator

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

8/3/2004 © 2002-04 Hal Perkins & UW CSE P-10



## **Bottom-Up Allocator**

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

8/3/2004

© 2002-04 Hal Perkins & UW CSE



## **Bottom-Up Allocator**

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

8/3/2004

© 2002-04 Hal Perkins & UW CSE

P-12

CSE P 501 Su04 P-2



# **Global Register Allocation**

- A standard technique is graph coloring
- Use control and dataflow graphs to derive interference graph
  - Nodes are virtual registers (the infinite set)
  - Edge between (t1,t2) when t1 and t2 cannot be assigned to the same register

    Most commonly, t1 and t2 are both live at the same time

    Can also use to express constraints about registers, etc.
- Then color the nodes in the graph
  - Two nodes connected by an edge may not have same color
  - If more than k colors are needed, insert spill code

© 2002-04 Hal Perkins & UW CSE

# **Coming Attractions**

- Dataflow and Control flow analysis
- Overview of optimizations

8/3/2004

© 2002-04 Hal Perkins & UW CSE

CSE P 501 Su04 P-3