



Boolean Expressions

- What do we do with this?
 - x > y
- It is an expression that evaluates to true or false
 - Could generate the value (0/1 or whatever the local convention is)
 - But normally we don't want/need the value; we're only trying to decide whether to jump

11/8/2005 © 2002-05 Hal Perkins & UW CSE



Code for exp1 > exp2

- Basic idea: designate jump target, and whether to jump if the condition is true or if it is false
- Example: exp1 > exp2, target L123, jump on false

<evaluate exp1 to eax> <evaluate exp2 to edx> cmp eax,edx jng L123

11/8/2005 © 2002-05 Hal Perkins & UW CSE



Boolean Operators: !

- Source
 - ! exp
- Context: evaluate exp and jump to L123 if false (or true)
- To compile !, reverse the sense of the test: evaluate exp and jump to L123 if true (or false)

11/8/2005

© 2002-05 Hal Perkins & UW CSE



K-21

K-23

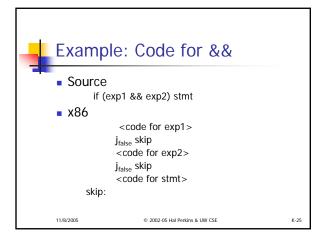
Boolean Operators: && and ||

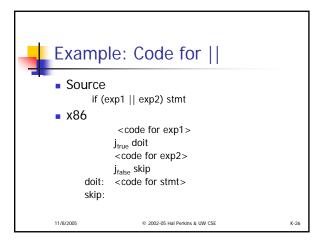
- In C/C++/Java/C#, these are shortcircuit operators
 - Right operand is evaluated only if needed
- Basically, generate the if statements that jump appropriately and only evaluate operands when needed

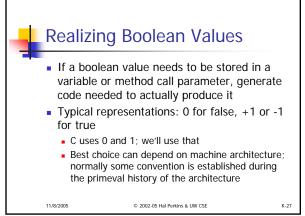
11/8/2005

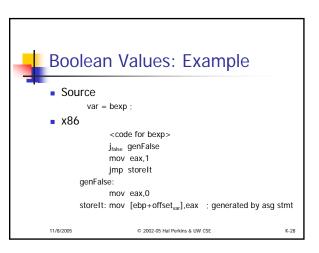
© 2002-05 Hal Perkins & UW CSE

K-22







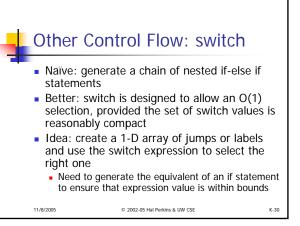


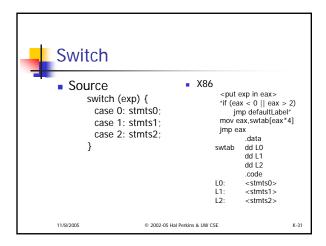
```
Faster, If Enough Registers

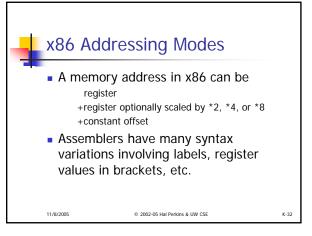
Source
var = bexp;
x86

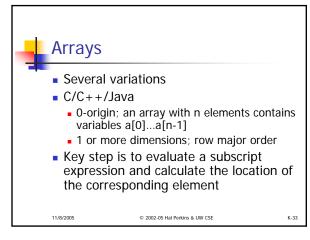
xor eax,eax
<code for bexp>
jralse storelt
inc eax
storelt: mov [ebp+offset_var],eax ; generated by asg stmt

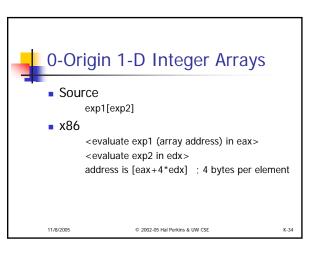
Or use conditional move (movecc) instruction if available – avoids pipeline stalls due to conditional jumps
```

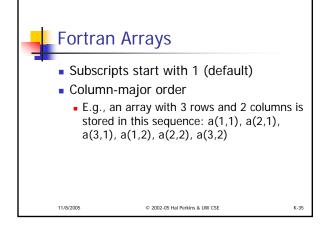


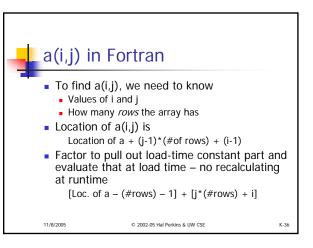














Coming Attractions

- Code Generation for Objects
 - Representation
 - Method calls
 - Inheritance and overriding
- Strategies for implementing code generators
- Code improvement optimization

11/8/2005

© 2002-05 Hal Perkins & UW CSE