Very Long Instruction Words (VLIW)

Goal: intstruction level parallelism (ILP)

Given these independent operations:

1) r1 = r2 + r3
2) r4 = r5 XOR r6
3) r7 = r8 / r9

execute them in parallel using different functional units

Superscalar Review

Superscalar: dynamic ILP from sequential stream

- small (64) instruction issue buffer

- dependency-checking hardware

- register-renaming hardware

- load/store reorder hardware

- etc.

VLIW: software-discovered ILP

VLIW compiler statically knows:
- available functional units
- operation latencies

Compiler must:
- group independent operations: Very Long Instruction Words
- optimize for common-case control-flow

Advantage:
- much less hardware!

Disadvantages:
- static predictions may not match dynamic behavior
- recompile for every new processor revision

VLIW Papers

Fisher: (HP)
- early description of VLIW
- description of "trace scheduling" compilation for VLIW

Rau: (HP)
- split-issue model of VLIW instructions
  + decouples VLIW code from hardware
  + enables dynamic out-of-order VLIW

Klaiber: (Transmeta)
- processor as hardware/software combination
- dynamic translation: CISC x86 -> RISC VLIW

VLIW issues

General programming issues:
- "magic blue smoke" early release programs
- atomicity of operations
- exception/interrupt handling
- trace-scheduling through method-dispatch / anonymous function calls

Static compilation:
- code portability
- compile-time branch-prediction

Dynamic compilation:
- runtime translation costs
- highly unpredictable code (e.g. "go" from spec95)