Limits on ILP

Achieving Parallelism

- Techniques
  - Scoreboarding / Tomasulo's Algorithm
  - Pipelining
  - Speculation
  - Branch Prediction
- But how much more performance could we theoretically get? How much ILP exists?
- How much more performance could we realistically get?

Methodology

- Assume an ideal (and impractical) processor
- Add limitations one at a time to measure individual impact
- Consider ILP limits on a hypothetically practical processor

- Analysis performed on benchmark programs with varying characteristics

Hardware Model – Ideal Machine

- Remove all limitations on ILP
  - Register renaming
    - Infinite number of registers available, eliminating all WAW and WAR hazards
  - Branch prediction
    - Perfect, including targets for jumps
  - Memory address alias analysis
    - All addresses known exactly, a load can be moved before a store provided that the addresses are not identical
  - Perfect caches
    - All memory accesses take 1 clock cycle
  - Infinite resources
    - No limit on number of functional units, buses, etc.

Ideal Machine

- All control dependencies removed by perfect branch prediction
- All structural hazards removed by infinite resources
- All that remains are true data dependencies (RAW) hazards
- All functional unit latencies: one clock cycle
  - Any dynamic instruction can execute in the cycle after its predecessor executes
- Initially we assume the processor can issue an unlimited number of instructions at once looking arbitrarily far ahead in the computation

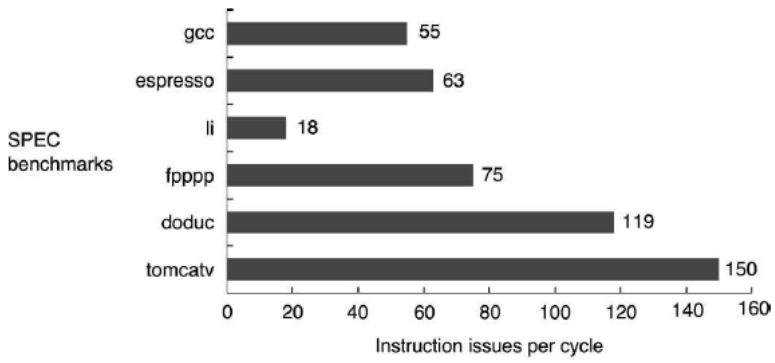
Experimental Method

- Programs compiled and optimized with standard MIPS optimizing compiler
- The programs were instrumented to produce a trace of instruction and data references over the entire execution
- Each instruction was subsequently re-scheduled as early as the true data dependencies would allow
  - No control dependence

Benchmark Programs – SPECINT92

Program	*Type*	*Description*
gcc	INT	Gnu C compiler
espresso	INT	Boolean espression minimizer
li	INT	Lisp interpreter
fpppp	FP	Quantum chemistry problem
doduc	FP	Monte-carlo simulation
tomcatv	FP	Generation of 2D coordinate system around a geometric region

ILP on the Ideal Processor



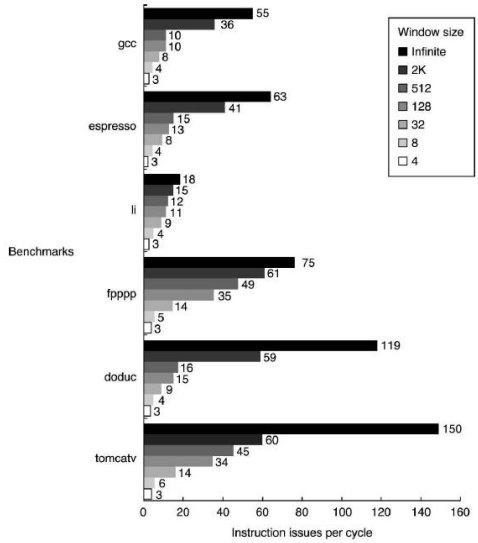
How close could we get to the ideal?

- The perfect processor must do the following:
  1. Look arbitrarily far ahead to find a set of instructions to issue, predicting all branches perfectly
  2. Rename all registers to avoid WAR and WAW hazards
  3. Resolve data dependencies
  4. Resolve memory dependencies
  5. Have enough functional units for all ready instructions

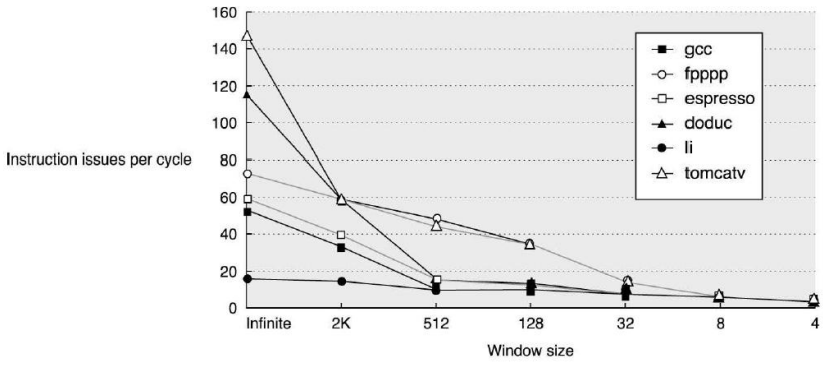
Limiting the Instruction Window

- Limit window size to n (no longer arbitrary)
  - Window = number of instructions that are candidates for concurrent execution in a cycle
- Window size determines
  - Instruction storage needed within the pipeline
  - Maximum issue rate
  - Number of operand comparisons needed for dependence checking is $O(n^2)$
    - To try and detect dependences among 2000 instructions would require some 4 million comparisons
    - Issuing 50 instructions requires 2450 comparisons

Effect of Reduced Window Size



Effect of Reduced Window Size



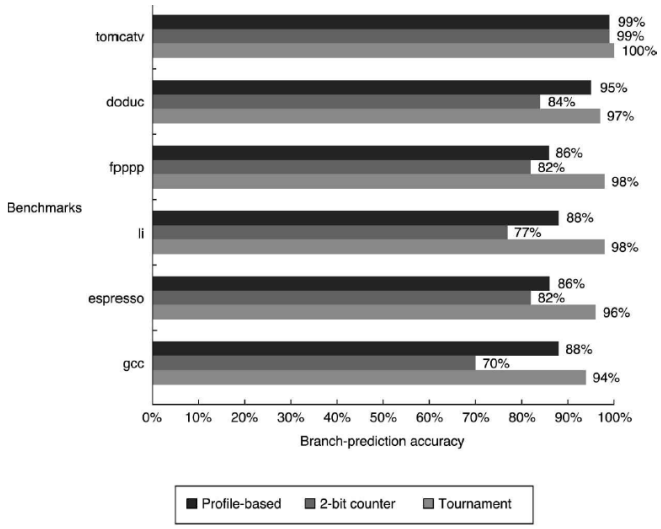
Effect of Reduced Window Size

- Integer programs do not have as much parallelism as floating point programs
  - Scientific nature of the program
- Highly dependent on loop-level parallelism
  - Instructions that can execute in parallel across loop iterations cannot be found with small window sizes without compiler help
- From now on, assume:
  - Window size of 2000
  - Maximum of 64 instructions issued per cycle

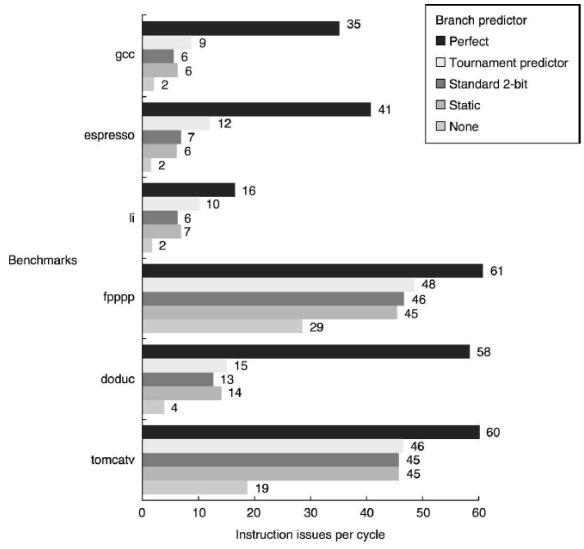
Effects of Branch Prediction

- So far, all branch outcomes are known before the first instruction executes
  - This is difficult to achieve in hardware or software
- Consider 5 alternatives
  1. Perfect
  2. Tournament predictor
     - (2,2) prediction scheme with 8K entries
  3. Standard (non-correlating) 2-bit predictor
  4. Static (profile-based)
  5. None (parallelism limited to within current basic block)
- No penalty for mispredicted branches except for unseen parallelism

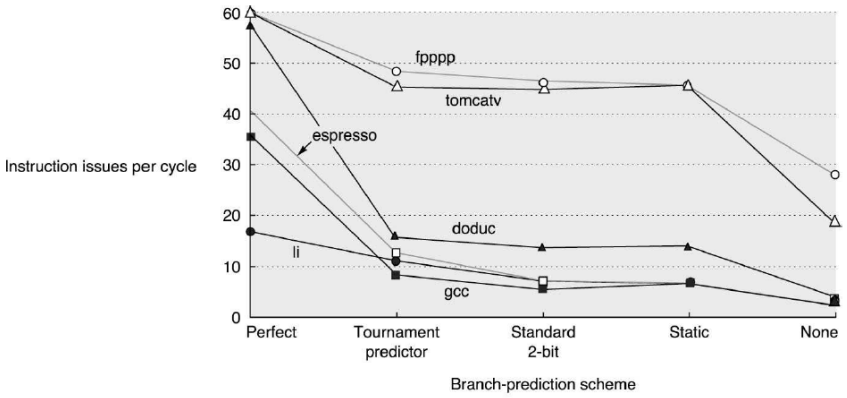
Branch Prediction Accuracy



Effects of Branch Prediction



Effects of Branch Prediction



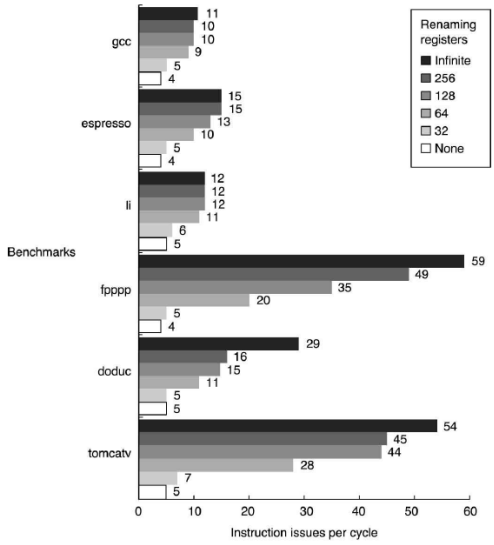
Branch Prediction

- Accurate prediction is critical to finding ILP
- Loops are easy to predict
- Independent instructions are separated by many branches in the integer programs and doduc
- From now on, assume tournament predictor
  - Also assume 2K jump and return predictors

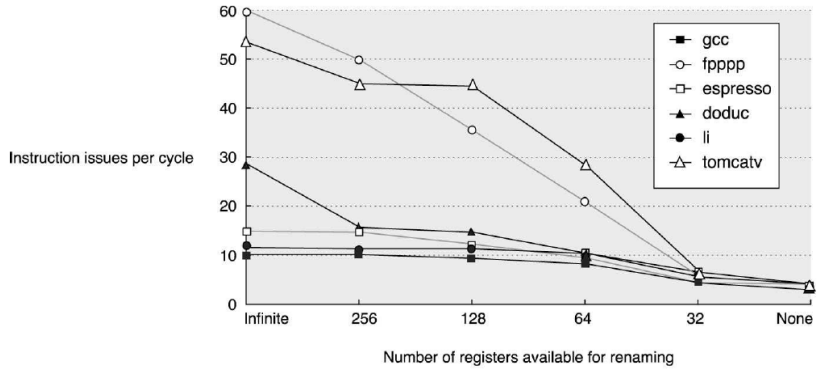
Effect of Finite Registers

- What if we no longer have infinite registers? Might have WAW or WAR hazards
- Alpha 21264 had 41 gp and integer renaming registers
- IBM Power5 has 88 fp and integer renaming registers

Effect of Renaming Registers



Effects of Renaming Registers



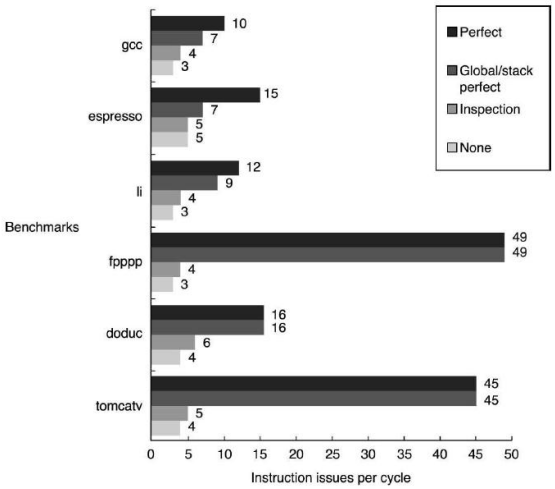
Renaming Registers

- Not a big difference in integer programs
  - Already limited by branch prediction and window size, not that many speculative paths where we run into renaming problems
- Many registers needed to hold live variables for the more predictable floating point programs
- Significant jump at 64
- We will assume 256 integer and FP registers available for renaming

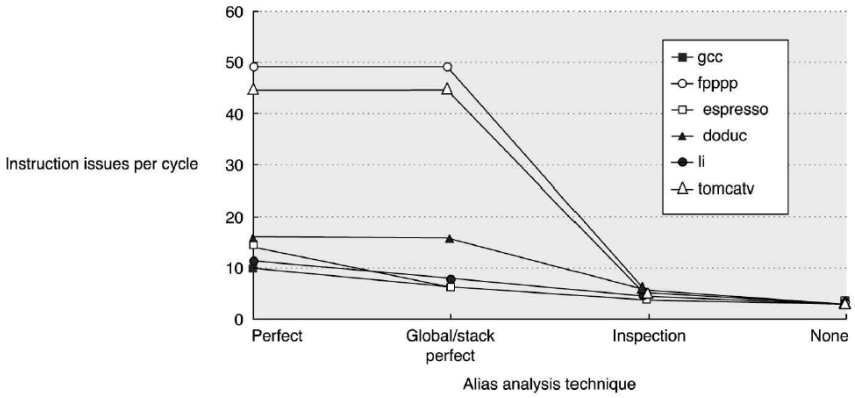
Imperfect Alias Analysis

- Memory can have dependencies too, so far assumed they can be eliminated
- So far, memory alias analysis has been perfect
- Consider 3 models
  - Global/stack perfect: idealized static program analysis (heap references are assumed to conflict)
  - Inspection: a simpler, realizable compiler technique limited to inspecting base registers and constant offsets
  - None: all memory references are assumed to conflict

Effects of Imperfect Alias Analysis



Effects of Imperfect Alias Analysis



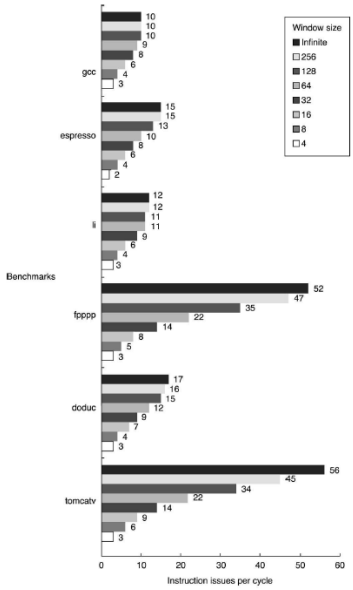
Memory Disambiguation

- Fpppp and tomcatv use no heap so perfect with global/stack perfect assumption
  - Perfect analysis here better by a factor of 2, implies there are compiler analysis or dynamic analysis to obtain more parallelism
- Has big impact on amount of parallelism
- Dynamic memory disambiguation constrained by
  - Each load address must be compared with all in-flight stores
  - The number of references that can be analyzed each clock cycle
  - The amount of load/store buffering determines how far a load/store instruction can be moved

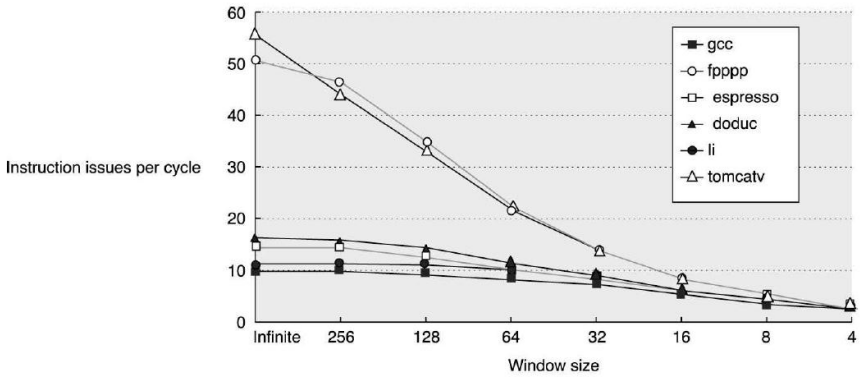
What is realizable?

- If our hardware improves, what may be realizable in the near future?
  - Up to 64 instruction issues per clock (roughly 10 times the issue width in 2005)
  - A tournament predictor with 1K entries and a 16 entry return predictor
  - Perfect disambiguation of memory references done dynamically (ambitious but possible if window size is small)
  - Register renaming with 64 int and 64 fp registers

Performance on Hypothetical CPU



Performance on Hypothetical CPU



Hypothetical CPU

- Ambitious/impractical hardware assumptions
  - Unrestricted issue (particularly memory ops)
  - Single cycle operations
  - Perfect caches
- Other directions
  - Data value prediction and speculation
    - Address value prediction and speculation
  - Speculation on multiple paths
  - Simpler processor with larger cache and higher clock rate vs. more emphasis on ILP with a slower clock and smaller cache