Exposing More ILP

- Motivated by VLIW, which needs tons of ILP to work at all – but useful for superscalar/dynamic/speculative processors, as well.
- Software Techniques
  - Software Pipelining
  - Trace Scheduling

Compiler support for ILP: Software Pipelining

- Observation: if iterations from loops are independent, then can get ILP by taking instructions from different iterations
- Software pipelining: reorganizes loops so that each iteration is made from instructions chosen from different iterations of the original loop

SW Pipelining Example

Unrolled 3 times
1 LD F0,0(R1)
2 ADDD F4,F0,F2
3 SD 0(R1),F4
4 LD F6,-8(R1)
5 ADDD F8,F6,F2
6 SD -8(R1),F8
7 LD F10,-16(R1)
8 ADDD F12,F10,F2
9 SD -16(R1),F12
10 SUBI R1,R1,#24
11 BNEZ R1,LOOP

Software Pipelined
1 SD 0(R1),F4; Stores M[i]
2 ADDD F4,F0,F2; Adds to M[i-1]
3 LD F0,-16(R1); loads M[i-2]
4 SUBI R1,R1,#8
5 BNEZ R1,LOOP

Compiler Support for ILP: Trace Scheduling

- Creates long basic blocks by finding long paths in the code
Trace Scheduling

- Parallelism across IF branches vs. LOOP branches
- Two steps:
  - **Trace Selection**
    - Find likely sequence of basic blocks (trace) of (statically predicted) long sequence of straight-line code
  - **Trace Compaction**
    - Squeeze trace into few VLIW instructions
    - Need bookkeeping code in case prediction is wrong

HW support for More ILP

- Avoid branch prediction by turning branches into *conditionally executed instructions*: (aka predicated instructions)
  
  ```
  if (x) then A = B op C else NOP
  ```
  
  - If false, then neither store result or cause exception
  - Expanded ISA of Alpha, MIPS, PowerPC, SPARC have conditional move; PA-RISC can annul any following instr, IA64 can predicate any instruction (even have multiple predicates)
  
  ```
  ld F2, 0(R2)
  addd F4, F2, F0
  muld F6, F4, F4
  beqz R3, go_on
  addd F10, F0, F8
  addi R2, R2, #8
  ```
  
  go_on:
  ```
  addi R2, R2, #8
  bnez F6, loop
  ```

Predicated Execution

- Drawbacks to conditional instructions
  - Still takes a clock & alu even if “annulled”
  - Stall if condition evaluated late
  - Complex conditions reduce effectiveness; condition becomes known late in pipeline
  - Requires more operands! Typically only available as conditional move.

- Advantages
  - Eliminate prediction, misprediction
  - Longer basic blocks, ...

ILP in real code

- Based on all kinds of ideal assumptions. Further limited by:
  - Realistic branch prediction
  - Finite renaming registers
  - Imperfect alias analysis for memory operations
Window Size

Renaming Registers

Memory Aliasing

PowerPC 620

- speculative, out-of-order, superscalar processor
- fetches 4, issues 4, completes 4, 6 independent functional units
- Reservation stations, reorder buffer, register renaming in the register file
PowerPC 620 Pipeline Structure

- Fetch
- Decode
- Issue
- Execute
- Commit

FIGURE 4.52 The average number of instructions that the fetch unit can provide to the issue unit varies between 3.2 and 4, with an average of 3.4 for the integer benchmarks and 3.8 for the FP benchmarks.

FIGURE 4.54 The IPC throughput rate for the issue stage is arrived at by subtracting stalls that arise in issue from the IPC rate sustained by the fetch stage.

FIGURE 4.57 The breakdown of the ideal IPC of 4.0 into its components.
Pentium 4

- 20-stage pipeline
- IA32 (x86) ISA translated to RISC-like uops
- Uops stored in trace cache
- Decode/retire 3 uops/cycle
- Execute 6 uops/cycle
- Dynamically scheduled
- Explicit Register Renaming
- Simultaneous Multithreading (hyper-threading)

ILP Summary

- Parallelism is absolutely critical to modern computer system performance, but at a very fine level.
- Mechanisms that create, or expose parallelism: loop unrolling, software pipelining, code motion
- Mechanisms that allow the machine to exploit ILP: pipelining, superscalar, dynamic scheduling, speculative execution