Exposing More ILP

- These techniques were originally 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
- Hardware/Software Technique
  - Predicated execution

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

```
Unrolled 3 times
1 LD F0,0(R1)  ADD F4,F0,F2  LD F0,-8(R1)
2 ADD F4,F0,F2 1LP:SD 0(R1),F4; Stores M[i]
3 SD 0(R1),F4  ADD F4,F0,F2; Adds to M[i-1]
4 LD F6,-8(R1)  SD 0(R1),F4
5 ADD F8,F6,F2  ADD F4,F0,F2
6 SD -8(R1),F8  SD -8(R1),F4
7 LD F10,-16(R1) SUBI R1,R1,#8
8 ADD F12,F10,F2  BNEZ R1,LP
9 SD -16(R1),F12  BNEZ R1,LOOP
10 SUBI R1,R1,#24
11 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

Predication: HW support for More ILP

- Avoid branch prediction by turning branches into conditionally executed instructions: (aka predicated instructions)
  
  ```
  add c, a, b (x)  \Rightarrow \text{if (x) then } a = b + c \text{ else NOP}
  ```
  
  - If false, then neither store result nor 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)
add  F4, F2, F0
multd F6, F4, F4
beqz R3, go_on
add  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, ...
- Critical technology for VLIW, sw pipelining. Why?

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

![SPEC benchmarks](image-url)
Window Size

Renaming Registers

Memory Aliasing

Pentium Pro (II, III, etc.) microarchitecture

- 40 uncommitted instructions
- 20 unissued instructions
Pentium 4 microarchitecture

Pentium 4 front-end

Pentium 4 back-end

Pentium 4 Summary

- 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)

- 126 in-flight instructions (ROB size)
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