Pipelining and Exceptions

- Exceptions represent another form of control dependence.
- Therefore, they create a potential branch hazard.
- Exceptions must be recognized early enough in the pipeline that subsequent instructions can be flushed before they change any permanent state.
- As long as we do that, everything else works the same as before.
- Exception-handling that always correctly identifies the offending instruction is called *precise interrupts*.

Pipelining in Today’s Most Advanced Processors

- Not fundamentally different than the techniques we discussed.
- Deeper pipelines.
- Pipelining is combined with
  - superscalar execution
  - out-of-order execution
  - VLIW (very-long-instruction-word)
A modest superscalar MIPS

- what can this machine do in parallel?
- what other logic is required?
- Represents earliest superscalar technology (e.g., circa early 1990s)

Superscalar Execution

- To execute four instructions in the same cycle, we must find four independent instructions
- If the four instructions fetched are guaranteed by the compiler to be independent, this is a VLIW machine
- If the four instructions fetched are only executed together if hardware confirms that they are independent, this is an in-order superscalar processor.
- If the hardware actively finds four (not necessarily consecutive) instructions that are independent, this is an out-of-order superscalar processor.
- What do you think are the tradeoffs?

Superscalar Scheduling

- assume in-order, 2-issue, ld-store followed by integer
  lw $6, 36($2)
  add $5, $6, $4
  lw $7, 1000($5)
  sub $9, $12, $5
- assume 4-issue, any combination (VLIW?)
  lw $6, 36($2)
  add $5, $6, $4
  lw $7, 1000($5)
  sub $9, $12, $5
  sw $5, 200($6)
  add $3, $9, $9
  and $11, $7, $6
- When does each instruction begin execution?

Superscalar vs. superpipelined

(multiple instructions in the same stage, same CR as scalar)

(more total stages, faster clock rate)
Dynamic Scheduling  
or  
Out-of-Order Scheduling

- Issues (begins execution of) an instruction as soon as all of its dependences are satisfied, even if prior instructions are stalled. (assume 2-issue, any combination)

```
lw $6, 36($2)  
add $5, $6, $4  
lw $7, 1000($5)  
sub $9, $12, $8  
sw $5, 200($6)  
add $3, $9, $9  
and $11, $5, $6
```

Reservation Stations

- are a mechanism to allow dynamic scheduling

![Reservation Station Diagram]

PowerPC 604, Pentium Pro (II, III)

- Deep pipeline
- Dynamically Scheduled (out-of-order scheduling)
- Trace Cache
- Simultaneous Multithreading (HyperThreading)

![PowerPC 604, Pentium Pro Diagram]

Pentium 4

- Deep pipeline
- Dynamically Scheduled (out-of-order scheduling)
- Trace Cache
- Simultaneous Multithreading (HyperThreading)
Modern Processors

- Pentium II, III – 3-wide superscalar, out-of-order, 14 integer pipeline stages
- Pentium 4 – 3-wide superscalar, out-of-order, simultaneous multithreading, 20+ pipe stages
- AMD Athlon, 3-wide ss, out-of-order, 10 integer pipe stages
- AMD Opteron, similar to Athlon, with 64-bit registers, 12 pipe stages, better multiprocessor support.
- Alpha 21164 – 2-wide ss, in-order, 7 pipe stages
- Alpha 21264 – 4-wide ss, out-of-order, 7 pipe stages
- Intel Itanium – 3-operation VLIW, 2-instruction issue (6 ops per cycle), in-order, 10-stage pipeline

Pipelining -- Key Points

- $ET = \text{Number of instructions} \times \text{CPI} \times \text{cycle time}$
- *Data hazards* and *branch hazards* prevent CPI from reaching 1.0, but *forwarding* and *branch prediction* get it pretty close.
- Data hazards and branch hazards need to be detected by hardware.
- Pipeline control uses combinational logic. All data and control signals move together through the pipeline.
- Pipelining attempts to get CPI close to 1. To improve performance we must reduce CT (superpipelining) or CPI below one (superscalar, VLIW).