Superscalar processor
Speculation, Reordering and ILP Limit

Outline
• Prev: Instruction renaming and scheduling
• Reordering
• Speculative execution
• Limit of ILP on a Wide Issue Processor

Tasks of superscalar processing
- Parallel Decoding and issue
- Parallel instruction execution
- Preserving the sequential consistency of instruction execution and exception processing

Checking in issue bound fetch
- Decoded instruction
- Rs1, Rs2, Rd
- Reset V bit of Rd
- Register File
- Update Rd, set V bit
- O(1) operand value
- EU
- O(2) operand
- Reservation station
- Check V1, V2
- O(C, O(1), O(2), Rd
- Association update of V1, V2 with Rd, set V bits
- Rd, result
- Common Data Bus

Part III
Instructions Reordering (RO Buffer)

Handling interrupts
- Status of instruction execution at the time of interrupt
- Completed
- Under execution
- Not started
- Program order
- These can "commit"
Speculated Execution

- Simple pipeline
  - Branch prediction reduces stalls due to control dependence
- Wide issue processor
  - Mere branch prediction is not sufficient
  - Speculated execution: Instructions in the predicted path need to be fetched and EXECUTED

What is required for speculation?

- Branch prediction to choose which instructions to execute
- Execution of instructions before control dependences are resolved
- Ability to undo the effects of incorrectly speculated sequence
- Preserving of correct behaviour under exceptions

Speculative execution

predicted branch

don't commit till correctness of prediction is determined

Types of speculation

- Hardware based speculation
  - done with dynamic branch prediction and dynamic scheduling
  - used in Superscalar processors
- Compiler based speculation
  - done with static branch prediction and static scheduling
  - used in VLIW processors

Reordering

i: issued
x: in execution
f: finished

instructions commit/retire

Using ROB with RF

from FUs

Register File
to reservation stations/FUs

Register File

ROB

from FUs
to reservation stations/FUs
Extending Tomasulo’s scheme for speculative execution

- Introduce re-order buffer (ROB)
- Add another stage – “commit”

Normal execution

- Issue
- Execute
- Write result

Speculative execution

- Issue
- Execute
- Write result
- Commit

How much to speculate?

- Handle exceptions in speculated instructions?
  - handle only low cost exception events such as first level cache miss
  - wait if expensive exceptional event occurs such as second level cache miss or TLB miss
- Speculating through multiple branches
  - needed when branches are frequent or clustered
  - even handling multiple branches in a cycle may be required

Case Study (Intel Pentium Pro)

- Super scalar CISC Processor with a RISC core
- It issue upto three RISC operations/Cycle and dispatch upto 5 operations/Cycle
- Unified RS with 20 entries for both FP & FX
- Strict Sequential consistency: using a ROB
- Renaming is implemented in ROB

Available ILP

- Available ILP: Program dependent
- 1st Case: ILP of 100, Assume A[i], B[i], C[i] in regs or Loop is unrolled 100 time
- 2nd Case, ILP is limited...you can’t do in efficiently..(inherent data dependency)
Limits to ILP

- Conflicting studies of amount
  - Benchmarks (vectorized Fortran FP vs. integer C programs)
  - Hardware sophistication

- How much ILP is available using existing mechanisms with increasing HW budgets?

- Do we need to invent new HW/SW mechanisms to keep on processor performance curve?
  - Intel MMX, SSE (Streaming SIMD Extensions): 64 bit ints
  - Intel SSE2: 128 bit, including 2 64-bit Fl. Pt. per clock
  - Motorola AltVec: 128 bit ints and FP
  - Supersparc Multimedia ops, etc.

Overcoming Limits

- Advances in compiler technology + significantly new and different hardware techniques may be able to overcome limitations assumed in studies

- However, unlikely such advances when coupled with realistic hardware will overcome these limits in near future

Limits to ILP HW Model comparison

<table>
<thead>
<tr>
<th>Model</th>
<th>Power 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions Issued per clock</td>
<td>Infinite</td>
</tr>
<tr>
<td>Instruction Window Size</td>
<td>Infinite</td>
</tr>
<tr>
<td>Renaming Registers</td>
<td>Infinite</td>
</tr>
<tr>
<td>Branch Prediction Perfect</td>
<td>Perfect</td>
</tr>
<tr>
<td>Cache</td>
<td>Perfect</td>
</tr>
<tr>
<td>Memory Alias Analysis</td>
<td>Perfect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Power 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions Issued per clock</td>
<td>Infinite</td>
</tr>
<tr>
<td>Instruction Window Size</td>
<td>Infinite</td>
</tr>
<tr>
<td>Renaming Registers</td>
<td>Infinite</td>
</tr>
<tr>
<td>Branch Prediction Perfect</td>
<td>Perfect</td>
</tr>
<tr>
<td>Cache</td>
<td>Perfect</td>
</tr>
<tr>
<td>Memory Alias Analysis</td>
<td>Perfect</td>
</tr>
</tbody>
</table>

Limits to ILP

Ideal (Initial) HW Model and Ideal compilers.

Assumptions for ideal/perfect machine to start:

- **Window size**
  - infinite (compiler can see whole program for analysis), infinite issue count
  - Unlimited Resource: 1 cycle latency for all instructions (FP *, /), unlimited instructions issued/clock cycle

- **Register renaming**
  - infinite virtual registers
  => all register WAW & WAR hazards are avoided

- **Branch prediction**
  - perfect; no miss predictions, no control dependencies; perfect speculation & an unbounded buffer of instructions available

- **Memory-address alias analysis**
  - all memory access addresses known & a load can be moved before a store provided addresses not equal

- **Perfect caches**
  - All memory access takes 1 cycle

Issue Window

- Suppose issue window size is 4
- All are register instructions

- Number of possible dependency
  
  \[ Rd_{i} (9) + Rs_{1}(9)+Rs_{2}(9) \] // 9 test
  +Rd_{i}(6) + Rs_{1}(6)+Rs_{2}(6) // 6 test
  +Rd_{i}(3) +Rs_{1}(3)+Rs_{2}(3) // 3 test
  \[
  = (4-1)*3 + (4-2)*3 + (4-3)*3 = 18 \]

- Window size N require

  \[
  (N-1)*3 + (N-2)*3 + ... + 3 = 3(1+2+...+N-1) = 3(N^2-N)/2
  \]
Limits to ILP HW Model comparison

<table>
<thead>
<tr>
<th></th>
<th>New Model</th>
<th>Model</th>
<th>Power 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions issued per clock</td>
<td>Infinite</td>
<td>Infinite</td>
<td>4</td>
</tr>
<tr>
<td>Instruction Window Size</td>
<td>Infinite, 2K, 512, 128, 32</td>
<td>Infinite</td>
<td>200</td>
</tr>
<tr>
<td>Renaming Registers</td>
<td>Infinite</td>
<td>Infinite</td>
<td>48 integer + 40 Fl. Pt.</td>
</tr>
<tr>
<td>Branch Prediction</td>
<td>Perfect</td>
<td>Perfect</td>
<td>2% to 6% misprediction (Tournament Branch Predictor)</td>
</tr>
<tr>
<td>Cache</td>
<td>Perfect</td>
<td>Perfect</td>
<td>64KI, 32KD, 1.92MB L2, 36 MB L3</td>
</tr>
<tr>
<td>Memory Alias</td>
<td>Perfect</td>
<td>Perfect</td>
<td>??</td>
</tr>
</tbody>
</table>

More Realistic HW: Window Impact

Change from Infinite window
2048, 512, 128, 32

Integer: 8 - 63

FP: 9 - 150

Upper Limit to ILP: Ideal Machine

To be continued