PowerPC 620 Case Study

- First-generation out-of-order processor
- Developed as part of Apple-IBM-Motorola alliance
- Aggressive goals, targets
- Interesting microarchitectural features
- Hopelessly delayed
- Led to future, successful designs

IBM/Motorola/Apple Alliance

- Alliance began in 1991 with a joint design center (Somerset) in Austin
  - Ambitious objective: unseat Intel on the desktop
  - Delays, conflicts, politics...hasn’t happened, alliance largely dissolved today
- **PowerPC 601**
  - Quick design based on RISC compatible with POWER and PowerPC
- **PowerPC 603**
  - Low power implementation designed for small uniprocessor systems
    - 5 FUs: branch, integer, system, load/store, FP
- **PowerPC 604**
  - 4-wide machine
    - 6 FUs, each with 2-entry RS
- **PowerPC 620**
  - First 64-bit machine, also 4-wide
    - Same 6 FUs as 604
    - Next slide, also chapter 5 in the textbook
- **PowerPC G3, G4**
  - Newer derivatives of the PowerPC 603 (3-issue, in-order)
    - Added AltiVec multimedia extensions
PowerPC 620
- Joint IBM/Apple/Motorola design
- Aggressively out-of-order, weak memory order, 64 bits
- Hopelessly delayed, very few shipped, but influenced later designs

PowerPC 620 Pipeline

- Dispatch Stage
  - Rename
  - Allocate rename buffer, completion buffer
  - Dispatch to reservation station
- Reservation Stations
  - 2 to 4 entries per functional unit, depending on type
  - RS holds instruction payload, operands
- Execute Stage
  - Six functional units
  - Execute, bypass to waiting RS entries, write rename buffer
- Completion Buffer
  - Sixteen entries, holds instruction state until in-order completion
PowerPC 620 Pipeline

- Fetch stage
  - Instruction buffer (8)
- Dispatch stage
  - Reservation stations (6)
  - Execute stage(s)
- Completion buffer (16)
- Complete stage
- Writeback stage

- Complete Stage
  - Maintains precise exceptions by buffering out-of-order instructions
  - 4-wide
- Writeback Stage
  - In-order writeback: results copied from rename buffer to architectured register file

Benchmark Performance

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Dynamic Instructions</th>
<th>Execution Cycles</th>
<th>IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress</td>
<td>6,884,247</td>
<td>6,062,494</td>
<td>1.14</td>
</tr>
<tr>
<td>Eqnott</td>
<td>3,147,233</td>
<td>2,188,331</td>
<td>1.44</td>
</tr>
<tr>
<td>espresso</td>
<td>4,615,085</td>
<td>3,412,653</td>
<td>1.35</td>
</tr>
<tr>
<td>Li</td>
<td>3,376,415</td>
<td>3,399,293</td>
<td>0.99</td>
</tr>
<tr>
<td>alvinn</td>
<td>4,861,138</td>
<td>2,744,098</td>
<td>1.77</td>
</tr>
<tr>
<td>hydro2d</td>
<td>4,114,602</td>
<td>4,293,230</td>
<td>0.96</td>
</tr>
<tr>
<td>tomcatv</td>
<td>6,858,619</td>
<td>6,494,912</td>
<td>1.06</td>
</tr>
</tbody>
</table>

Branch Prediction Accuracy

<table>
<thead>
<tr>
<th>BranchPrediction</th>
<th>compress</th>
<th>Eqnott</th>
<th>espresso</th>
<th>Li</th>
<th>alvinn</th>
<th>hydro2d</th>
<th>tomcatv</th>
</tr>
</thead>
<tbody>
<tr>
<td>BranchResolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Taken</td>
<td>40.35%</td>
<td>31.84%</td>
<td>40.05%</td>
<td></td>
<td>6.38%</td>
<td>17.51%</td>
<td>6.12%</td>
</tr>
<tr>
<td>Taken</td>
<td>59.65%</td>
<td>68.16%</td>
<td>59.95%</td>
<td></td>
<td>93.62%</td>
<td>82.49%</td>
<td>93.88%</td>
</tr>
<tr>
<td>BTACPrediction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>84.18%</td>
<td>82.64%</td>
<td>81.99%</td>
<td></td>
<td>94.49%</td>
<td>88.31%</td>
<td>93.31%</td>
</tr>
<tr>
<td>Incorrect</td>
<td>15.82%</td>
<td>17.36%</td>
<td>18.01%</td>
<td></td>
<td>5.51%</td>
<td>11.69%</td>
<td>6.69%</td>
</tr>
<tr>
<td>BHT Prediction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolved</td>
<td>19.71%</td>
<td>18.30%</td>
<td>17.09%</td>
<td></td>
<td>28.83%</td>
<td>17.49%</td>
<td>26.18%</td>
</tr>
<tr>
<td>Correct</td>
<td>68.86%</td>
<td>72.16%</td>
<td>72.27%</td>
<td></td>
<td>81.58%</td>
<td>68.00%</td>
<td>52.50%</td>
</tr>
<tr>
<td>Incorrect</td>
<td>11.43%</td>
<td>9.54%</td>
<td>10.64%</td>
<td></td>
<td>0.92%</td>
<td>5.92%</td>
<td>4.00%</td>
</tr>
<tr>
<td>BTAC Incorrect and BHT Correct</td>
<td>0.01%</td>
<td>0.79%</td>
<td>1.13%</td>
<td>7.79%</td>
<td>0.07%</td>
<td>0.19%</td>
<td>0.00%</td>
</tr>
<tr>
<td>BTAC Correct and BHT Incorrect</td>
<td>0.00%</td>
<td>0.12%</td>
<td>0.37%</td>
<td>0.26%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Overall Branch Prediction Accuracy</td>
<td>88.35%</td>
<td>98.40%</td>
<td>89.36%</td>
<td>91.24%</td>
<td>98.97%</td>
<td>94.10%</td>
<td>97.95%</td>
</tr>
</tbody>
</table>
Wasted Fetch Cycles

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Misprediction</th>
<th>I-Cache Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress</td>
<td>6.65%</td>
<td>0.01%</td>
</tr>
<tr>
<td>eqntott</td>
<td>11.78%</td>
<td>0.08%</td>
</tr>
<tr>
<td>espresso</td>
<td>10.84%</td>
<td>0.52%</td>
</tr>
<tr>
<td>li</td>
<td>8.92%</td>
<td>0.09%</td>
</tr>
<tr>
<td>alvinn</td>
<td>0.39%</td>
<td>0.02%</td>
</tr>
<tr>
<td>hydro2d</td>
<td>5.24%</td>
<td>0.12%</td>
</tr>
<tr>
<td>tomcatv</td>
<td>0.68%</td>
<td>0.01%</td>
</tr>
</tbody>
</table>

Buffer Utilization

- Instruction buffer
  - Decouples fetch/dispatch
- Completion buffer
  - Supports OOO execution

Dispatch Stalls

<table>
<thead>
<tr>
<th>Source of Dispatch Stalls</th>
<th>Frequency of dispatch stall cycles.</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress</td>
<td>24.35%</td>
</tr>
<tr>
<td>eqntott</td>
<td>41.40%</td>
</tr>
<tr>
<td>espresso</td>
<td>35.28%</td>
</tr>
<tr>
<td>li</td>
<td>30.00%</td>
</tr>
<tr>
<td>alvinn</td>
<td>30.09%</td>
</tr>
<tr>
<td>hydro2d</td>
<td>17.13%</td>
</tr>
<tr>
<td>tomcatv</td>
<td>6.25%</td>
</tr>
</tbody>
</table>

Issue Stalls

<table>
<thead>
<tr>
<th>Source of Issue Stalls</th>
<th>Frequency of issue stall cycles.</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress</td>
<td>62.67%</td>
</tr>
<tr>
<td>eqntott</td>
<td>65.61%</td>
</tr>
<tr>
<td>espresso</td>
<td>51.94%</td>
</tr>
<tr>
<td>li</td>
<td>46.15%</td>
</tr>
<tr>
<td>alvinn</td>
<td>78.70%</td>
</tr>
<tr>
<td>hydro2d</td>
<td>60.29%</td>
</tr>
<tr>
<td>tomcatv</td>
<td>93.64%</td>
</tr>
</tbody>
</table>
### Parallelism Achieved

<table>
<thead>
<tr>
<th></th>
<th>0%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issuing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finishing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Summary of PowerPC 620
- First-generation OOO part
- Aggressive goals, poor execution
- Interesting contributions
  - Two-phase branch prediction (also in 604)
  - Short pipeline
  - Weak ordering of memory references
- PowerPC evolution
  - 1998: Power3 (630FP)
  - 2001: Power4
  - 2004: Power5

### 620 vs. Power3 vs. Power4

<table>
<thead>
<tr>
<th>Attribute</th>
<th>620</th>
<th>Power3</th>
<th>Power4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>172 MHz</td>
<td>450 MHz</td>
<td>1.3 GHz</td>
</tr>
<tr>
<td>Pipeline depth</td>
<td>5+</td>
<td>5+</td>
<td>15+</td>
</tr>
<tr>
<td>Branch predictor</td>
<td>Bimodal BHT + BTAC</td>
<td>Same</td>
<td>3x16 To combining</td>
</tr>
<tr>
<td>Fetch/issue/completion width</td>
<td>4/8/4</td>
<td>4/8/5</td>
<td></td>
</tr>
<tr>
<td>Rename/physical registers</td>
<td>8 Int, 8 FP</td>
<td>16 Int, 24 FP</td>
<td>80 Int, 72 FP</td>
</tr>
<tr>
<td>in-flight instructions</td>
<td>16</td>
<td>32</td>
<td>Up to 100</td>
</tr>
<tr>
<td>FP Units</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Instruction Cache</td>
<td>32K 8e SA</td>
<td>32K 128e SA</td>
<td>54K 6M</td>
</tr>
<tr>
<td>Data Cache</td>
<td>32K 8e SA</td>
<td>94K 128e SA</td>
<td>32K 7e SA</td>
</tr>
<tr>
<td>L1 data cache</td>
<td>16M</td>
<td>16M</td>
<td>128M/256M</td>
</tr>
<tr>
<td>L2 bandwidth</td>
<td>1GB/s</td>
<td>6.4GB/s</td>
<td>100+ GB/s</td>
</tr>
<tr>
<td>Store queue entries</td>
<td>5 x 16B</td>
<td>16 x 16B</td>
<td>12 x 64B</td>
</tr>
<tr>
<td>MSHRs</td>
<td>1:1:1:1</td>
<td>1:2:4</td>
<td>1:2:4:8</td>
</tr>
<tr>
<td>Hardware prefetch</td>
<td>None</td>
<td>4 streams</td>
<td>8 streams</td>
</tr>
</tbody>
</table>

### IBM Power4
- IBM POWER4, began shipping in 2001
  - Deep pipeline: 15 stages minimum
  - Aggressive combining branch prediction
  - Over 100 instructions in flight, tracked in 20 groups of 5 in ROB
  - Aggressive memory hierarchy, memory bandwidth
Chapter 7: Intel’s P6 Architecture
Modern Processor Design: Fundamentals of Superscalar Processors

Pentium Pro Case Study

• Microarchitecture
  – Order-3 Superscalar
  – Out-of-Order execution
  – Speculative execution
  – In-order completion
• Design Methodology
• Performance Analysis

Goals of P6 Microarchitecture

IA-32 Compliant
Performance (Frequency - IPC)
Validation
Die Size
Schedule
Power

P6 – The Big Picture
Memory Hierarchy

- Level 1 instruction and data caches - 2 cycle access time
- Level 2 unified cache - 6 cycle access time
- Separate level 2 cache and memory address/data bus
- Level 2 cache fill policy - implications

Instruction Fetch

- L2 Cache (256Kb)
- Instruction Buffer (256Kb)
- Instruction Fetch
- Instruction Data
- Branch Target Buffer (512)
- Branch Target
- Branch Prediction
- Prediction Control Logic
- Pattern History Table (PHT)

- Pattern History Table (PHT) is not speculatively updated
- A speculative Branch History Register (BHR) and prediction state is maintained
- Uses speculative prediction state if it exist for that branch
Branch Prediction Algorithm

- Current prediction updates the speculative history prior to the next instance of the branch instruction
- Branch History Register (BHR) is updated during branch execution
- Branch recovery flushes front-end and drains the execution core
- Branch mis-prediction resets the speculative branch history state to match BHR

Instruction Decode - 1

- Branch instruction detection
- Branch address calculation - Static prediction and branch always execution
- One branch decode per cycle (break on branch)

Instruction Decode - 2

- Instruction Buffer contains up to 16 instructions, which must be decoded and queued before the instruction buffer is re-filled
- Macro-instructions must shift from decoder 2 to decoder 1 to decoder 0

What is a uop?

Small two-operand instruction - Very RISC like.

IA-32 instruction
add (eax), (ebx) \( \text{MEM(eax)} \leftarrow \text{MEM(eax)} + \text{MEM(ebx)} \)

Uop decomposition:

- \( \text{ld guop0(eax)} \rightarrow \text{guop0} \leftarrow \text{MEM(eax)} \)
- \( \text{ld guop1(ebx)} \rightarrow \text{guop1} \leftarrow \text{MEM(ebx)} \)
- \( \text{add guop0, guop1} \rightarrow \text{guop0} \leftarrow \text{guop0} + \text{guop1} \)
- \( \text{sta eax} \)
- \( \text{std guop0} \rightarrow \text{MEM(eax)} \leftarrow \text{guop0} \)
### Instruction Dispatch

- Register Renaming
- Allocation requirements:
  - "3-or-none" Reorder buffer entries
  - Reservation station entry
  - Load Buffer or store buffer entry
- Dispatch buffer "probably" dispatches all 3 uops before re-fill

### Register Renaming - Example

#### Register Renaming - Example cont’d

- Similarly to Tomasulo’s Algorithm - Uses ROB entry number as tags
- The register alias tables (RAT) maintain a pointer to the most recent data for the renamed register
- Execution results are stored in the ROB
Challenges to Register Renaming

Real Register File (RRF) | Integer RAT | Reorder Buffer (ROB)
--- | --- | ---
8 | EAX | 6
8 | ECX | 1
8 | FST0 | 2
12 | FST1 | 3
4 | FST2 | 4
9 | FST7 | 5

Floating Point RAT

8-bit code
- mov AL, #data1
- mov AH, #data2
- add AL, #data3
- add AL, #data4

Byte addressable registers

Out-of-Order Execution Engine

- In-order branch issue and execution
- In-order load/store issue to address generation units
- Instruction execution and result bus scheduling
- Is the reservation station “truly” centralized & what is “binding”??
Instruction Completion

- Handles all exception/interrupt/trap conditions
- Handles branch recovery
  - OOO core drains out right-path instructions, commits to RRF
  - In parallel, front end starts fetching from target/fall-through
  - However, no renaming is allowed until OOO core is drained
  - After draining is done, RAT is reset to point to RRF
  - Avoids checkpointing RAT, recovering to intermediate RAT state
- Commits execution results to the architectural state in-order
  - Retirement Register File (RRF)
  - Must handle hazards to RRF (writes/reads in same cycle)
  - Must handle hazards to RAT (writes/reads in same cycle)
- "Atomic" IA-32 instruction completion
  - steps are marked as 1st or last in sequence
  - exception/interrupt/trap boundary
- 2 cycle retirement

Pentium Pro Design Methodology - 1

Pentium Pro Performance Analysis

- Observability
  - On-chip event counters
  - Dynamic analysis

- Benchmark Suite
  - BAPco Sysmark32 - 32-bit Windows NT applications
  - Winstone97 - 32-bit Windows NT applications
  - Some SPEC95 benchmarks

Performance – Run Times

Total of 27.5 billion cycles
Conclusions

IA-32 Compliant
Performance (Frequency - IPC)
366.0 ISpec92
283.2 FSpec92
8.09 SPECint95
6.70 SPECfp95
Validation
Die Size - Fabable
Schedule - 1 year late
Power -