Case Studies: The PowerPC 620 and Intel P6

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

- Fetch stage
- Instruction buffer (8)
- Branch target buffer (2)
-Reservation stations (6)
- Execute stage(s)
- Finisher buffer (16)
- Complete stage
- Writeback stage

- Dispatch Stage
  - Rename
  - Allocate: rename buffer, completion buffer
  - Dispatch to reservation station
  - Branches: match (if operands available) or predict with BTB
- 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**

<table>
<thead>
<tr>
<th>MUX</th>
<th>XSU0</th>
<th>XSU1</th>
<th>MC-FXU</th>
<th>LSU</th>
<th>FPU</th>
</tr>
</thead>
</table>

- **Reservation station (6)**
- **Execute stage(s)**
- **Completion buffer (16)**
  - **Complete stage**
  - **Writeback stage**

- **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>2,493,230</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**

  - **Two-phase branch prediction**
    - **BTAC**
      - Holds targets of taken branches only
      - On miss, fetch sequential (not-taken) path
      - Accessed in single cycle in fetch stage
      - Generates fetch address for next cycle
    - **BHT**
      - Accessed in dispatch stage
      - 256 entries, 2-way set-associative
  - **Interactions**
    - \{BTAC right, wrong\} x \{BHT right, wrong\} = 4 cases
    - BHT overrides BTAC

- **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 Not Taken</td>
<td>40.35%</td>
<td>33.44%</td>
<td>60.07%</td>
<td>33.08%</td>
<td>63.54%</td>
<td>17.23%</td>
<td>61.12%</td>
</tr>
<tr>
<td>Taken</td>
<td>59.65%</td>
<td>66.56%</td>
<td>39.93%</td>
<td>66.92%</td>
<td>36.46%</td>
<td>82.77%</td>
<td>38.88%</td>
</tr>
<tr>
<td>BTAC Prediction Correct</td>
<td>84.18%</td>
<td>81.64%</td>
<td>74.71%</td>
<td>94.49%</td>
<td>88.31%</td>
<td>63.31%</td>
<td>93.08%</td>
</tr>
<tr>
<td>Incorrect</td>
<td>15.82%</td>
<td>18.36%</td>
<td>25.29%</td>
<td>55.51%</td>
<td>11.69%</td>
<td>36.69%</td>
<td></td>
</tr>
<tr>
<td>BHT Prediction Resolved Correct</td>
<td>19.71%</td>
<td>18.30%</td>
<td>28.83%</td>
<td>17.49%</td>
<td>26.18%</td>
<td>45.39%</td>
<td></td>
</tr>
<tr>
<td>Incorrect</td>
<td>80.29%</td>
<td>71.70%</td>
<td>71.18%</td>
<td>82.51%</td>
<td>68.82%</td>
<td>54.61%</td>
<td></td>
</tr>
<tr>
<td>Overall Branch Prediction Accuracy</td>
<td>88.57%</td>
<td>90.46%</td>
<td>89.36%</td>
<td>91.28%</td>
<td>99.07%</td>
<td>94.18%</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

- Frequency of dispatch stall cycles.
  - Source of Dispatch Stalls: compress eqntott espresso li alvinn hydro2d tomcatv
  - Serialization: 0.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00%
  - Move to special register constraint: 0.00% 4.49% 0.94% 3.49% 0.00% 0.94% 0.00%
  - Read port saturation: 0.00% 0.00% 0.37% 0.37% 0.00% 0.37% 0.00%
  - Reservation station saturation: 36.07% 22.36% 31.50% 34.40% 22.81% 42.70% 36.51%
  - Rename buffer saturation: 24.06% 7.60% 13.93% 17.26% 1.36% 16.98% 34.13%
  - Completion buffer saturation: 5.54% 3.64% 2.42% 4.27% 21.12% 7.89% 9.65%
  - Another to same unit: 9.72% 20.51% 18.31% 10.57% 24.38% 12.01% 7.17%
  - No dispatch stalls: 24.35% 41.00% 35.28% 30.00% 30.00% 17.53% 6.35%

### Issue Stalls

- Frequency of issue stall cycles.
  - Source of Issue Stalls: compress eqntott espresso li alvinn hydro2d tomcatv
  - Out of order disallowed: 0.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00%
  - Serialization: 1.69% 1.81% 3.21% 10.81% 0.03% 4.47% 0.01%
  - Waiting for source: 21.97% 29.30% 37.79% 32.03% 17.74% 22.71% 3.52%
  - Waiting for execution unit: 13.67% 3.28% 7.66% 11.01% 2.81% 1.50% 1.30%
  - No issue stalls: 62.07% 6.61% 51.94% 46.15% 78.70% 60.29% 93.64%
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>8+</td>
<td>8+</td>
<td>16+</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>4/4/4</td>
<td>4/8/5</td>
<td></td>
</tr>
<tr>
<td>Rename/physical registers</td>
<td>6 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>instructions units</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Instruction Cache</td>
<td>32K 8e 8a</td>
<td>32K 128e 8a</td>
<td>64K 64M</td>
</tr>
<tr>
<td>Data Cache</td>
<td>32K 8e 8a</td>
<td>96K 128e 8a</td>
<td>256K 64e</td>
</tr>
<tr>
<td>L1 Data Cache</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB/2MB</td>
</tr>
<tr>
<td>L1 bandwidth</td>
<td>16GB/s</td>
<td>64GB/s</td>
<td>100+ GB/s</td>
</tr>
<tr>
<td>Store queue entries</td>
<td>6 x 8B</td>
<td>16 x 8B</td>
<td>12 x 64B</td>
</tr>
<tr>
<td>MSHRs</td>
<td>L1/D:1</td>
<td>L1/D:4</td>
<td>L1/D: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

- 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

Instruction Cache Unit

Branch Target Buffer
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) MEM(eax) <- MEM(eax) + MEM(ebx)
```

Uop decomposition:

```
Id guop0, (eax)  guop0  <- MEM(eax)  
Id guop1, (ebx)  guop1  <- MEM(ebx)  
add guop0,guop1 guop0  <- guop0 + guop1  
sta eax           MEM(eax)  <- guop0  
std guop0         MEM(eax)  <- 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
Challenges to Register Renaming

Real Register File (RRF)

<table>
<thead>
<tr>
<th>Integer RAT</th>
<th>Reorder Buffer (ROB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAX</td>
<td>EAX</td>
</tr>
<tr>
<td>ECX</td>
<td>ECX</td>
</tr>
<tr>
<td>EBX</td>
<td>EBX</td>
</tr>
<tr>
<td>ECX</td>
<td>ECX</td>
</tr>
<tr>
<td>FST0</td>
<td>FST0</td>
</tr>
<tr>
<td>FST1</td>
<td>FST1</td>
</tr>
<tr>
<td>FST2</td>
<td>FST2</td>
</tr>
<tr>
<td>FST3</td>
<td>FST3</td>
</tr>
</tbody>
</table>

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”?

Reservation Station

- Cycle 1
  - Order checking
  - Operand availability
- Cycle 2
  - Writeback bus scheduling

Memory Ordering Buffer (MOB)

- Load buffer retains loads until completed, for coherency checking
- Store forwarding out of store buffers
- 2 cycle latency through MOB
- “Store Coloring” - Load instructions are tagged by the last store
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
  - uops 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
Performance – IPC vs. uPC

Instructions and Uops retired per cycle

Performance – Branch Prediction

Branch Miss/Prediction Rate

6.8% avg

Branch Miss Rate
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 -