SCALE:
Software-Controlled Architectures for Low Energy

Krste Asanovic, Mark Hampton, Seongmoo Heo, Ronny Krashinsky, Albert Ma, Gong Ke Shen, Jessica Tseng, Michael Zhang

MIT Laboratory for Computer Science
Performance-Oriented Architectures

• Implementations of modern RISC/VLIW ISAs perform a large number of microarchitectural operations for each user instruction
  – For integer add instruction on 5-stage RISC pipeline only ~2% of energy is the 32-bit adder circuit itself
  – Rest includes cache tags and data, TLBs, register files, pipeline registers, exception state management, ...
Performance-Oriented Architectures

• Implementations of modern RISC/VLIW ISAs perform a large number of microarchitectural operations for each user instruction
  – For integer add instruction on 5-stage RISC pipeline only ~2% of energy is the 32-bit adder circuit itself
  – Rest includes cache tags and data, TLBs, register files, pipeline registers, exception state management, ...

• Modern microarchitectures pipeline and parallelize microarch ops such that they have no user-visible performance impact

⇒ No incentive to expose these microarch ops in a purely performance-oriented ISA
Performance-Oriented Architectures

- Implementations of modern RISC/VLIW ISAs perform a large number of microarchitectural operations for each user instruction
  - For integer add instruction on 5-stage RISC pipeline only ~2% of energy is the 32-bit adder circuit itself
  - Rest includes cache tags and data, TLBs, register files, pipeline registers, exception state management, ...

- Modern microarchitectures pipeline and parallelize microarch ops such that they have no user-visible *performance* impact
  ⇒ No incentive to expose these microarch ops in a purely performance-oriented ISA

*Energy-consumption is hidden from software*
Energy-Exposed Architectures

- Allow energy-conscious compiler to remove superfluous microarch ops
Energy-Exposed Architectures

- Allow energy-conscious compiler to remove superfluous microarch ops

**Reward compile-time analysis with run-time energy savings**
Tag-Unchecked Loads and Stores

Allow software to avoid cache tag check when successive memory accesses are to same cache line

\[
\text{ld } r1, (r2) \\
\text{ld.nochk } r3, 4(r2) \quad \# \quad \text{Must be to same cache line}
\]

Energy reductions:
- no tag RAM read/compare or no tag CAM search
- only low order address bits need to be computed
- no TLB lookup for physically tagged caches

⇒ Reduces cache access energy to just RAM read
Instruction Chains

- Many register values have lifetime of exactly one instruction

  ```assembly
  ld  r1, (r2)
  addi r1, r1, 1
  st  r1, (r2)
  ```
Instruction Chains

• Many register values have lifetime of exactly one instruction
  
  \[ \text{ld} \ r1, (r2) \]
  \[ \text{addi} \ r1, r1, 1 \]
  \[ \text{st} \ r1, (r2) \]

• Create hybrid RISC-accumulator architecture to pass values between instructions
  
  \[ \text{ld} \ (r2) \ | \ \text{addi} \ 1 \ | \ \text{st} \ (r2) \]
Instruction Chain Benefits

\[ \text{ld (r2)} \mid \text{addi 1} \mid \text{st (r2)} \]

- **Reduced register file activity**
  - only write to bypass latches, not regfile
  - reduce reads from reg file

- **Reduced instruction fetch bandwidth**
  - compact encoding for accumulator operands

- **Reduced exception state management**
  - only update exception PC at head of chain
  - exceptions always restart at head of chain