RISC vs CISC

By Douglas Smith and Mandeep Plaha

What is RISC/CISC

- Both are **Instruction Set Architectures (ISA)**
- **RISC** = Reduced Instruction Set Computer
  - Faster, simpler hardware
  - More instructions per programs
- **CISC** = Complex Instruction Set Computer
  - Slower, more complex hardware
  - Fewer instructions per program
- CISC processors came first with RISC processors first emerging in late 1970s/early 1980s sparking a lot of debate

The 1980s debate + historical context

- Original reasons for **CISC**
  - Memory small: few instructions a positive
  - Programmers working in assembly, able to take full advantage of more complex instructions
- **High Level Languages** (HLL) arise
  - Compilers writing assembly, unable to take advantage of complex instructions
- Major constraints are **chip area, processor design complexity**
- Arguments/Advantages of **RISC**
  - Shorter **design time**
  - Better use of **chip space** (more general purpose registers, caches, pipelining)
  - Greater **speed**
  - Assembly doesn't need to closely match with HLL
- Despite RISC advantages argument arose and CISC remained dominant due to
  - **Legacy** and **commercial interest**
  - Lack of major initial support to take a risk with RISC
- But over time RISC processors grew in presence/number and the debate's heat faded as
  - The line between RISC/CISC processors faded each **inheriting elements of the other** (Intel micro-ops, Arm Thumb and single instruction multiple data(SIMD), etc)
  - Improvements in technology made the major constraints less relevant

The current debate + conclusions

- Today's computing landscape is significantly shaped by **smartphones and tablets**.
- **Energy** and **power** are the primary design constraints.
- Cross market appearance of both ISAs, e.g., **ARM-based servers**, and **x86-based mobile devices**.
- Considering the dominance of ARM and x86 and the multipronged importance of, we need to compare ARM to x86 on the metrics of **power**, **energy**, and **performance**.
- Key findings:
  - **Performance differences** are generated by **ISA-independent** microarchitecture differences.
  - The **energy consumption** is again **ISA-independent**.
  - One ISA is not fundamentally more efficient.
  - ARM and x86 implementations are simply design points optimized for different performance levels.
  - Overall x86 implementations consume significantly more power than ARM implementations. However, the **choice** of **power** or **performance optimized core designs** impacts core power use more than ISA.
  - **Energy** use is also primarily impacted by **design** choice and not by the ISA.
- Conclusions
  - ISA being RISC or CISC is **largely irrelevant** for today's mature microprocessor design world.

Sources

- http://courses.cs.washington.edu/courses/cse470/17sp/readings/RISC_CISC_2.pdf
- http://courses.cs.washington.edu/courses/cse470/17sp/readings/RISC_CISC_1.pdf
- https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/
- https://cs.stackexchange.com/questions/269/why-would-anyone-want-cisc