RISC vs. CISC

LESLIE DAHLBERG, Mälardalens University

JONATHAN LARSSON, Mälardalens University

Abstract

• Computer systems organization → Architectures → Serial architectures → Reduced instruction set computing • Computer systems organization → Architectures → Serial architectures → Complex instruction set computing

Additional Key Words and Phrases: RISC, CISC, Agile, DSP, Embedded

# An introduction to the CISC vs. RISC debate

Today computer architecture can be divided into two separate categories based upon the ISA: first the CISC architecture which stands for Complex Instruction Set Computer, second the RISC architecture which stands for Reduced Instruction Set Computer. Intel’s x86 chipset is a good example of CISC while ARM, MIPS, SPARC and POWER PC are common representations of the RISC architecture. [[BO](#Liu11)]

RISC architecture designs usually focus on large register files, simple STORE/LOAD operations and a limited set of simple fixed-length instructions. CISC on the other hand utilizes many complicated instructions, fewer registers, micro-coded operations and complex memory addressing modes. [[BO](#Geo90)]

The advantages of using a RISC design are: a simpler CPU design, faster development, efficient parallel execution, better compiler optimization support and reduced pipeline branching penalties. The advantages of using a CISC design are: faster context-switching, powerful assembly language programming, easier compiler design, flexibility through micro-coded instructions, fast floating point instructions, reduced memory requirements, better cache performance, reduced bus traffic and compatibility with widely used architectures. [[BO](#Geo90)]

During the 1990s and 1980s the advantages of the two ISAs we’re heavily debated. It was thought that CISC would remain the dominant ISA in the short term because of binary compatibility but that RISC would eclipse it in the long term because of improved compiler optimizations and high levels of parallelism which suited RISC. [[BO](#Geo90)]This debate has recently resurfaced because the fundamental requirements and technical limitations of the field have changed and mobile devices have taken over a large part of the market for computer chips. While the main constraints in 1990s we’re chip area and design complexity, today energy consumption and power constraints dominate. Today RISC is mainly used in embedded systems, microcontrollers, smartphones and tablets while CISC is prominent in personal computers and servers. CISC has however found it’s way into mobile devices too and RISC has been utilized in server operations. Moreover Blum et al. carried out a quantitative study in which they concluded that the ISA is irrelevant to performance and energy consumption. They assert that RISC and CISC ISAs are just optimizations suited for different performance levels. [[BO](#Emi13)]

# A brief history of two ISAs

A brief history of two ISAs

Both Intel and Motorola have released new CISC microprocessors in parallel during the years. Intel started with a 4-bit microprocessor in 1971 that was mostly used in calculators. Intel then continued with an 8-bit microprocessor in 1972, a 16-bit in 1978 and a 32-bit in 1985. At almost the same time, Motorola joined the CISC microprocessor business with their 8-bit processor in 1974. Motorola then continued with their first 16-bit in 1979 and then with their 32-bit in 1984. [[BO](#Jam95)]

A few years after Intel´s release of their first CISC microprocessor, IBM started with their development of RISC microprocessors. IBM and other organizations then continued with this development in parallel with the Intel and Motorola RISC-families. The first RISC microprocessor that was released was from the University of California, Berkeley with their RISC project in 1980 even though IBM started their development in 1975. Sun Microsystems incorporated some of the features from the Berkeley RISC project and released SPARC (Scalable Processors Architecture) in 1987. Motorola then started with it´s RISC development and released their first version in 1988. In 1993, three of the largest companies in the computer industry (Apple, Motorola and IBM) joined hands to develop the RISC PowerPC family. [[BO](#Jam95)]

## Motorola 88000

Starting in the 1970s people started to question if the complexities of CISC ISA designs were justified and the alternative, RISC, started to gain popularity. The promise of RISC seemed to good to be true: half the development time, half the cost and double the performance. Intel, Motorola and National Semiconductor all had vested interests in CISC ISAs and found it hard to switch to RISC without jeopardizing their existing designs but Motorola started developing a true RISC processor called 88100 belonging to the 88000 series of processors. The 88100 was a 32-bit microprocessor with an on-board floating point unit and 51 instructions. All instructions executed in a single cycle and the processor contained separate execution units to handle the heavily pipelined execution model. The 88200 extended this design with on chip memory management and caches. [[BO](#RDe88)]

# Modern risc implementations

RISC processors are used heavily in many industries and play a vital role in mobile computing. This section details some new developments RISC design.

## RISC-V

RISC-V is new open-source ISA which is intended to be a simple general-purpose commercial ISA with support for 64-bit address spaces on servers and desktops and 32-bit address spaces on mobile devices. It has a load-store architecture and utilizes 45 basic instructions but is extendible through ISA extensions. A RISC-V processor has be manufactured by IBM at 45 nm and runs at >1 GHz. The ISA implements the revised IEEE 754 floating point standard and is fully virtualizable. The Rocket implementation of RISC-V features a 6 point fixed pipeline, full supervisor support, a 64-bit fixed point datapath and a double precision FPU. [[BO](#And13)]

## Floating Point Coprocessor

Floating point operations such as additions, subtraction, multiplication, division and square root are important in scientific and engineering implementations and significantly slow down processing on RISC processors. A floating point coprocessor (FPU) is therefore often used in conjunction with RISC architectures to speed up these operations. Patil et al. have described how an FPU could be used in the RISC-V architecture reaching a frequency of 240MHz on single precision and 180 MHz on double precision operations. [[BO](#Pat15)]

## Agile RISC development

Lee et al. proposed a new approach to developing chip architectures using agile methods which are popular in software engineering. The main principles are: 1. Incomplete, fabricatable prototypes over fully featured models 2. Collaborative, flexible teams over rigid silos 3. Improvement of tools and generators over improvement of the instance 4. Response to change over following a plan. Their approach focuses on using a set of customized tools to quickly iterate over the design-production-testing cycle. They developed Chisel, an extension to the Scala programming language, which makes it possible to describe hardware components in Scala in an object-oriented fashion and convert the code into executable c++ simulations of the processor or low-level Verilog which can used to manufacture the chip hardware. The Raven-3 RISC-V processor is an example of a processor which was developed using the Agile methodology and which benefited greatly from reusability and sharing of expertise inherent to Agile development processes. [[BO](#Lee16)]

## Compressed code RISC architecture

Embedded systems are often heavily constrained when it comes to memory. This has spawned some research into compressible machine code in order to reduce the size of executable files. In these systems the executable is compressed in main memory and gets decompressed before entering the cache memory. The speed is therefore correlated to the number of cache misses. Since the program can contain jumps and braches conventional compression algorithms can’t be used as any one word or cache block in memory must be decompressible independently. Lekatsas et al. have compared three different compression algorithms on both x86 and MIPS. The compression ratios are better on MIPS than on x86 giving RISC ISAs an advantage in embedded applications with high memory constraints. [[BO](#Har98)]

# hybrid architectures

Using pure CISC or RISC approaches to architecture can work well in general cases but further performance can be gained by combined the two approaches in specialized systems.

## Hobbit

Already in the early 1990s, developers built hybrid versions of CISC and RISC. P. V. Argade et al developed their version which is called “Hobbit”. The RISC parts of the implementation in Hobbit is a reduced instruction set, it only retains the most common instructions. Hobbit also uses a single cycle instruction execution and 64 32-bit registers. The CISC parts of the implementation is variable length instruction format to produce more code density. It uses a memory to memory architecture, no delayed branching, no load delay slot and complex instructions implemented in the hardware, such as divide and multiplication of integers. [[BO](#Arg02)]

## MISA

There are a number of different instruction set architecture standards with the CISC x86 ISA and the RISC ARM being the most successful. The problem with having different ISA:s is that computers running one of them can not run the other ones applications without recompiling the source code. MISA is a multiple instruction set architecture processor design which can run both ARM and x86 binaries. To make precompiled binaries platform-independent, it is required to have a layer to handle the translation for the binary code from one architecture to another. There are two different ways to handle the translation of precompiled binaries. Interpretation, dynamic translation and static translation. Interpretation means each instruction is mapped at runtime to the equivalent instruction of the target ISA without saving any caches.  
Dynamic translation is almost the same as interpretation with the difference that a cache is saved to reuse some of the translated code. Static translation instead translates without having to run the code first, this makes the translation more complex and the translator can loose some of the code. [[BO](#Hus11)] uses their own hardware binary interpreter (XAM) to translate from x86 to ARM instructions in their implementation of MISA. The MISA design is the same as a standard ARM architecture but with x86 decoder and the XAM. It works by first decoding the x86 instructions which are then sent to the XAM which then translates to ARM instructions at runtime. MISA has the same performance as compiled ARM binaries but with a 14% power overhead. [[BO](#Hus11)]

## RCISC

Implementing advanced DSP applications on embedded systems faces a large problem with the load/store concept because DSP operations such as Fast Fourier Transforms require fast access to large swaths of memory. Lozano et al. have demonstrated that this can be overcome by reading the memory directly like in CISC processors. Though one may think that this would impact performance negatively, modern SRAM chips are fast enough to make this feasible. They demonstrated that what they call a  Reduced Complexity Instruction Set Computer can be constructed without significantly higher power consumption while at the same time improving performance by a factor of 2.0x - 3.5x depending on the application. RCISC combined the simple set of instructions of RISC with the direct memory access of CISC. [[BO](#Loz15)]

## CRISC

According to H. Krad and A. Y. Al-Taie, the RISC vs. CISC discussion is no longer relevant. This is because the new and also the future architectures are hybrids of both CISC and RISC. One of these implementations is Intel’s Pentium series chips which are described as CRISC (Complex-Reduce Instruction Set Computer). The Intel Pentium-Pro is an example from this family, it executes normal CISC instructions but uses an internal implementation which is a Post RISC CPU. In recent years, RISC architectures have selected to use a larger instruction set and CISC architectures have understood the profit with implementing a basic set of instructions that can be executed within a single CPU cycle. [[BO](#Has07)]

# CONCLUSIONS

The CISC vs. RISC debate has been active since the 1980s but recent research has shown that the choice of ISA is not relevant to general performance or energy efficiency. The choice between CISC and RISC should rather be based on specific performance requirements and optimized to the situation at hand. Combining the two approaches can also yield boosted performance in special cases like embedded systems.

REFERENCES

x

|  |  |
| --- | --- |
| BO | Argade, P. V., Aymeloglu, S., Berenbaum, A. D. et al. Hobbit: a high-performance, low-power microprocessor. In *Compcon Spring '93, Digest of Papers.* ( 2002). |
| BO | Blem, Emily, Menon, Jaikrishnan, and Sankaralingam, Karthikeyan. Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures. ( 2013), High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium, 1-12. |
| BO | Dettmer, R. Making RISC- respectable the Motorola 88000. *IEE Review*, 34, 6 (June 1988), 237-240. |
| BO | George, Alan D. An overview of RISC vs. CISC. ( 1990), System Theory, 1990., Twenty-Second Southeastern Symposium. |
| BO | Jamil, Tariq. RISC vs CISC Why less is more (September 1995), 13-16. |
| BO | Karaki, Hussein and Akkary, Haitham. Multiple instruction sets architecture (MISA). In *Energy Aware Computing (ICEAC), 2011 International Conference on* ( 2011), IEEE, 1 - 6. |
| BO | Krad, Hasan and Al-Taie, Aws Yousif. A New Trend for CISC and RISC Architecture. *Asian Journal of Information Technology*, 6, 11 (2007), 1125--1131. |
| BO | Lee, Yunsup, Waterman, Andrew, Cook, Henry et al. An Agile Approach to Building RISC-V Microprocessors. *IEEE Micro* , 36, 2 (March 2016), 8 - 20. |
| BO | Lekatsas, Haris and Wolf, Wayne. Code Compression for Embedded Systems. In *Design Automation Conference, 1998. Proceedings* ( 1998), IEEE. |
| BO | Liu, Ya, Zhou, Yanling, Zheng, Xianping, and Xiong, Wangping. The research and development of reduced instruction set computer. ( 2011), Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. |
| BO | Lozano, Hanni and Ito, Mabo. A Reduced Complexity Instruction Set architecture for low cost embedded processors. In *IEEE* ( 2015), 400-407. |
| BO | Patil, Vinayak, Raveendran, Aneesh, Sobha, P M, Selvakumar, A. David, and Vivian, D. Out of order floating point coprocessor for RISC V ISA. In *VLSI Design and Test (VDAT), 2015 19th International Symposium on* ( 2015), IEEE. |
| BO | Waterman, Andrew, Lee, Yunsup, Avizienis, Rimas, Cook, Henry, Patterson, David, and Asanovic, Krste. The RISC-V instruction set. In *Hot Chips 25 Symposium (HCS), 2013 IEEE* ( 2013), IEEE. |

x