Introduction to Instruction Set Architecture

Take your first steps into Instruction Set Architecture as well as a comparison of CISC and RISC design practices

If you decided that learning what goes on under the hood of a computer is valuable, we would like to congratulate you on taking your knowledge of computers and data processing systems to a higher level. The first fundamental level in overall computer design theory is the ISA or Instruction Set Architecture.

What is ISA?

We can think of the ISA as a translator which helps two people speaking different languages communicate. In this case, the two "people" trying to communicate are our computer's hardware and our computer's software.

In practice, the hardware of our computer, such as the *Central Processing Unit* (CPU), works with a series of electrical signals called *binary*, meaning 'composed of two things:

- 0 representing 'OFF' or FALSE
- 1 representing 'ON' or TRUE

Our computer's software, the programming languages we typically write in like C++, Java, or Python, uses human-readable code such as double priceOfGas = 2.49; and if-else statements. The hardware and software of our computers are not speaking the same language because 0's and 1's do not directly equate into phrases that we programmers can read.

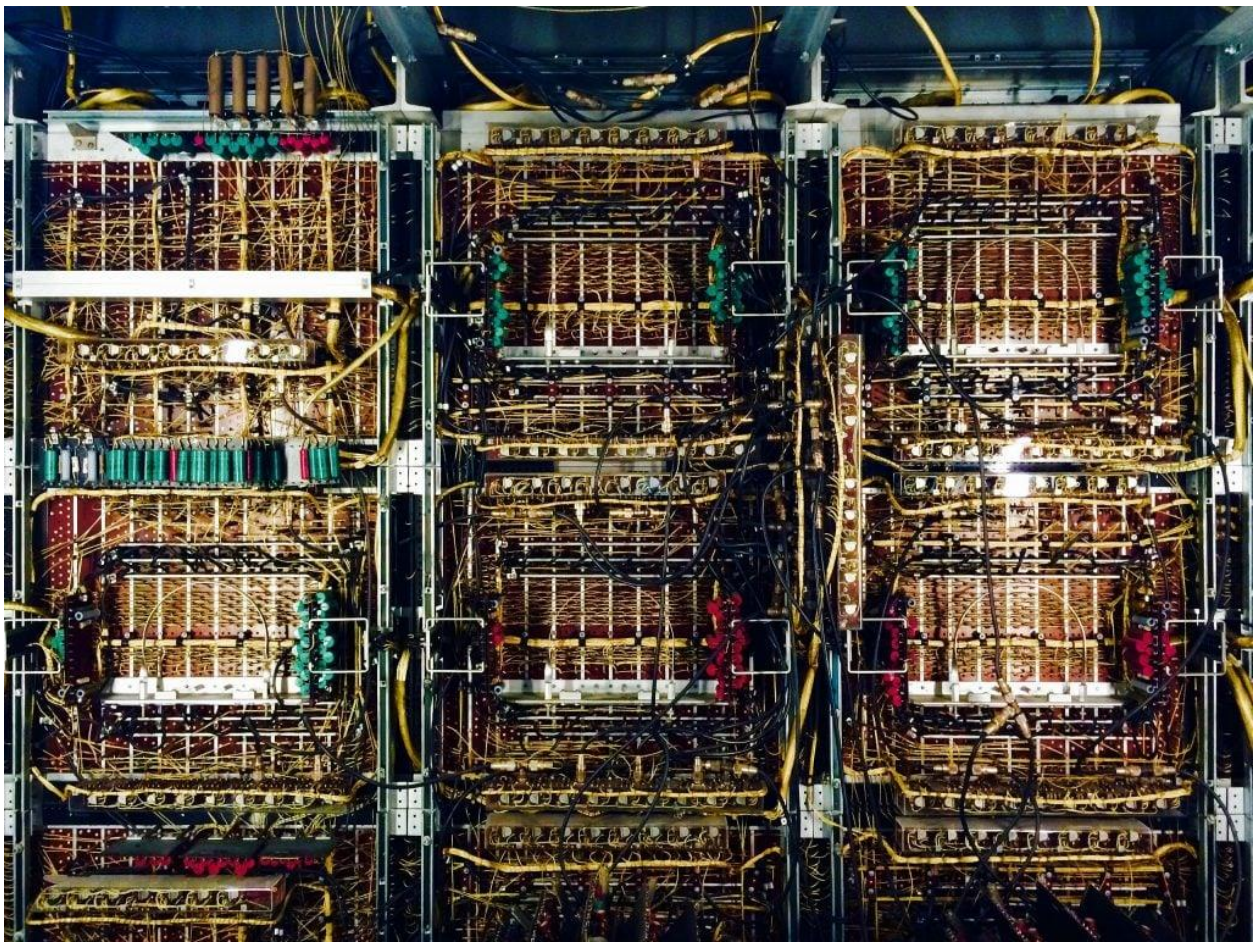
It is ultimately the job of the ISA to manage the translation of these values into the hardware readable code. The ISA defines the list of all the functions that the CPU can understand and how to translate a message between the hardware and software so that both entities understand each other.

Challenges of Connecting Hardware to Software

Picture yourself as a computer scientist in the middle of the 20th century faced with the following problem:

"How can we figure out how to make operating a computer easier? How can we set up these computers so our programmers can focus on writing computer programs and not have to relearn the computer components? We want the electrical engineers to work with the hardware and the software engineers to work with the logic without the two having to talk to each other. We need to come up with a standard for these communications."

You can already see how great a challenge this would become and how, through painful trial and error problem solving, we are still evaluating ways to increase the efficiency of the process. The fact that our hardware is constantly getting smaller, faster, and able to run at a higher capacity has been a constant challenge in arriving at the perfect ISA solution. What used to take up an entire room, such as the UNIVAC I computer below, can now be contained in the world's "dumbest" smartphone.



Credit: Computer History Museum

The UNIVAC I was praised for being able to compute a hundred years of math in two hours. The fastest computer today, the Summit, now can compute “in one second, what would take humans 6 billion years.” It still takes up about the same space as the UNIVAC, but with much better performance characteristics. Just as software design practices evolve to change the industry, hardware advancements make those changes obsolete, and vice versa. This constant battle has driven the speed of advancement to an incredible pace.

Credit: Carlos Jones/ORNL, CC BY 2.0

Early History: CISC vs RISC

CISC

The first attempts at designing ISAs focused on reducing the number of instructions that a computer would have to process. Many of the data transfer channels were moved by hand, physically disconnecting wires and plugging them into different components, so the logic went that the fewer times people had to move the cables, the faster this would go. Memory space was also very limited, meaning that the fewer lines of code, the better.

Inside the computer, this was implemented with more hardware such as cables, components, and a significant increase in power consumption to processes complex instructions. It was generally viewed that combining instructions such as “load a number from memory, perform an arithmetic calculation,

and store to another memory location" into one instruction was the most efficient way to get things done.

This new standard of architecture models came to be known as *Complex Instruction Set Computer*, or CISC for short, and for quite some time reigned supreme in design theory. The design focused on having computer scientists write as few lines of code as possible to maximize their performance.

RISC

It wasn't until the mid-1970s that another approach came along, the *Reduced Instruction Set Computer* or RISC. John Cocke, widely recognized as the founder of RISC, theorized that if they simplified the instructions to a very limited number of easy calculations, the computer could process them in a single machine cycle, allowing for multiple processes to be "pipelined" (something we'll learn in an upcoming module), and in turn resulting in faster overall speeds and reducing the amount of power needed to operate the computer.

Dr. Cocke, when reviewing the inputs and outputs of a CISC computer at a telegraph switchboard, realized that the computer rarely chose the most advantageous method of performing a task. By reducing the number and complexity of the instructions he could improve the overall performance while at the same time reducing components, power use, and heat generation. The downside was that the programmers would have to write more code for the computer. We might not think twice nowadays about creating a several thousand line program, disk space is nearly limitless and RAM is abundant, but in the days of Dr. Cocke, these resources were just beginning to show signs of being capable of handling large programs.

Modern and Future ISA Design

Three primary ISAs have emerged out of the technology revolution of the late 20th century:

- x86 (CISC design)
- ARM (RISC design)
- MIPS (modified RISC design for embedded processors)

Each design has its advantages, disadvantages, and parts of the industry where they excel and will continue to do so for the foreseeable future. Eventually, quantum computing will force the development of new ISAs as hardware begins relying on more than two states (OFF and ON) and transforms into multi-state logic processors (OFF, ON, and both). Companies such as IBM, Intel, and Microsoft are working diligently to develop these solutions as the technology arrives.

Conclusion

Let's review some of the key topics you learned throughout this article:

- An ISA, or Instruction Set Architecture, is the framework that defines the communication between the hardware and software of our computers.
- It is the set of commands that our computers can perform and these definitions fall into two distinct design practices, CISC and RISC.