CS 5341

Oscar Galindo

**A New Golden Age for Computer Architecture**

The paper starts by briefly mentioning the early history of ISAs and how a man by the name of Maurice Wilkins proposed to simplify the control of machines using something called “control store”. As mentioned, “control store” gave rise to a new programming paradigm called microprogramming which in essence allowed for the use of pre-programmed microinstructions stored in memory instead of logic gates to perform instructions. While technically Wilkin’s work was an improvement it allowed IBM to merge different ISAs into a single ISA.

The paper then proceeds to describe the evolution from Wilkin’s work done in the 1960s to the implementations given to microprogramming in the decades of the 1970s and the 1980s. As the paper mentions, many advances in memory allowed for more complex ISAs, for example and notably Intel’s 8800 and 80386, all of this during the 1970s. By the 1980s computers having much bigger control stores allowed for the investigation of the limits of microprogramming through Complex Instruction Sets, which were implemented in Complex Instruction Sets Computers or CISC. By then -1980s- the critical question became “what instructions would compilers generate?” So, with this idea in mind different researchers invented machines that could use complex instructions sets, among them David Patterson. Researchers quickly realized that few instructions generated much of the microcode, which suggested interpretation bugs were occurring. David Patterson’s solution to the issue was to suggest a correcting mechanism for CISC to prevent bugs. Patterson attempted to publish his work about the microinstruction interpreter correcting mechanism but failed, his work was received with much dislike among the reviewers at the journal *Computer.* Patterson’s failure had two unintended consequences, first Patterson got the needed motivation to work in RISC (Reduced Instruction Set Computers) and the CISCs were “doomed” -not really- to be dominated by the RISCs. RISCs allowed for a reduced base of instructions and the elimination of the needed interpreter to convert instructions into microcode, among other advances notably among such advances was performance. RISCs implementations allowed for a 4x faster execution with respect to similar programs interpreted with CISCs implementations. Nevertheless, CISC remained the standard although inferior in terms of performance to RISC.

Later, as the document mentions, more innovations like VLIW (Very Long Instruction Word) and EPIC (Explicitly Parallel Instruction Computers) were attempted to replace the current reference ISA called x86 32-bit. The idea behind the VLIW and EPIC ISAs was to increase the number of encoded instructions in a single row of memory from 1 to many. Although, both VLIW and EPIC failed to provide a faster execution with most programs while in some very specific cases the architectures worked as intended, this last force the rise of the x86 64-bit as the new reference ISA. x86 would then become the dominant ISA until 2011, when x86 peaked and ever-since has been losing 10% of market share every year. With the launch of the iPhone in 2007 Apple opened the market to RISCs that implemented a SoC (System on a Chip), so, as the document mentions, CISC dominated the PC era, but the post-PC era driven by mobile devices is driven by RISC.

As mentioned in the document, the new era of computing faces some challenges. For example, the unattainable scale-up of the number of transistors in a fixed density and the increase of energy used by each of the transistors in a cheap, known as Moore’s Law and Dennard Scaling, as well as a historic lack of focus of integrating security as part of the execution within ISAs. In turn these issues have created the opportunities to explore DSA (Domain Specific Architectures) and DSL (Domain Specific Languages) as areas of research that could offer new ideas to continue the progress of computing capacity. The document makes an emphasis on the idea that open ISAs or open architectures could greatly benefit our evolution by making the interested community be a player on developing security, perfecting performance, etc. In addition, the document concludes with a proposal as to how the Agile Software Development methodology could be replicated with hardware development. Finally, the document offers an optimistic outlook for the future, assuming that the future will be interesting and innovative for ISAs.

While the paper has and impeccable and thorough summarization of the events that gave rise to modern computing ISAs, I find that few additions could be done to the paper in terms of providing an outlook for the future. For example, I believe the authors could have added what’s to come in the immediate future because although the gap between projected transistor density and actual transistor density (as stated in Moore’s Law) is significant isn’t there a chance we could develop a technology to fix this problem? Also, I believe there is a significant value at mentioning security as a fundamental challenge that we must address as a community, could the development of security measures be a part of the Agile Hardware development? I believe other discussions that could have been “provoked” are can we as a community survive creating Domain Specific Languages that tend to super-specialization like MATLAB with Matrix Multiplications? What could be a realistic definition of a correct implementation of an ISA with respect to instruction execution sequences, if any? Is there or are there means to fulfill the suggested conversion of interpreted languages to compiled languages as suggested with the Python-C example? Otherwise, I believe the paper does a great job, as intended, at surveying a quite historic and relevant topic as computer architectures, it catches the attention by naturally showing the progression of ideas that were used to build upon to construct every type of ISAs. In addition, I believe the paper is a great demonstration of a Turing Lecture.