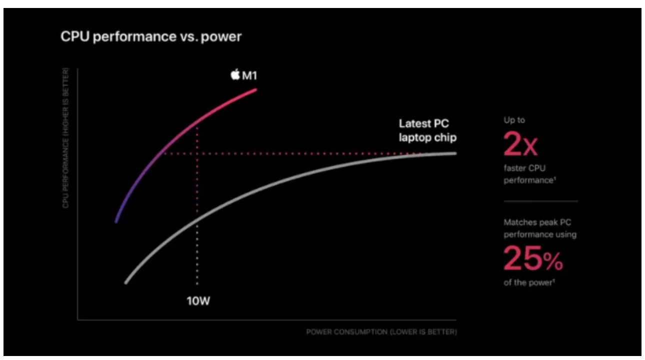
CS159-N: Apple in ARM Architecture

Silin Meng
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000

silin.meng@sjsu.edu

ABSTRACT
The competition between ARM architecture and x86 architecture has never stopped in the past decades. ARM architecture uses the logic of RISC in its design, achieving high efficiency in battery consumption, and on the opposite, x86 architecture based on the logic of CISC represents the high performance of the computer. However, the release of the M1 chip seems to break the balance between these two architectures and achieve performance while maintaining efficiency simultaneously through collaboration with its operating system MacOS. In this report, I will discuss what Apple has done with its M chip, and its performance compared to other Intel chips representing x86 architecture.

1. YEAR 2020: APPLE M1 RELEASED



Back on November 10 in 2022, this graph given by Apple shocked the industry. The new Apple PC with Arm architecture M1 chip first-time beat the intel chip in terms of performance, while maintaining a low power consumption. Having low power consumption in an arm architecture chip over x86 architecture is never a surprise to the industry but performance is the key. To understand better the new Apple chip, we need to first understand what is the difference between arm architecture and x86 according to our experience.

2. ARM vs X86

The main difference between ARM and x86 is indeed the design, in which ARM uses RISC architecture and x86 uses CISC architecture. Both architectures have their advantages and disadvantages over each other. To fully understand the differences, knowing the features of RISC and CISC is very crucial.

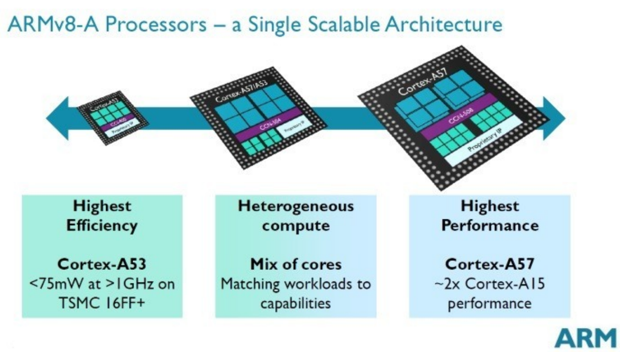
Over the last few decades, x86 and CISC undoubtedly dominate the PC industry. Before the release of the instruction set concept, old software couldn't run on the new computer. With the instruction set, the new computer model would obtain the same instruction set as the old computer model so that the old software can run on the new computer model. The idea of CISC and RISC is based on this instruction set concept. CISC represents complex instruction set computer, and RISC represents reduced instruction set computer. According to the name, we can know designing CISC means that we need more variety of complex instruction sets, and designing RISC means that we just need to focus on less simple instruction sets, where it's reduced. And there is a huge controversy about which of these two is better but CISC has more advantages due to its historical status.

Each instruction set of CISC is more powerful than RISC, which can do much more stuff. On the opposite, CISC consumes more power than RISC in each instruction. CISC architecture is created with many powerful instruction sets, and each instruction set takes a varying amount of time, making it mostly run under sequential processing (hard for pipelining). RISC is created with many simple instruction sets, and each instruction set takes an identical amount of time. This feature makes RISC easier to perform pipelining, and parallelism gives a big boost in performance. However, the challenging part of RISC is redirected to the software level, and writing an efficient program and compiler would be important for the performance. And I think the great combination of software and hardware is one of the key reasons why Apple M1 has so much performance while maintaining good power consumption.

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

For the tradeoff between power consumption and performance, ARM also came up with the big.LITTLE architecture early in the day. ARM's big.LITTLE architecture is an innovation that Intel couldn't follow for a while, and recently Intel has this architecture with its 12th Gen Alder Lake CPUs. In the big.LITTLE architecture, processors can obtain different types of cores.

Traditional multi-core processors contain identical cores with the same power consumption. ARM brings heterogeneous computing to mobile devices with big.LITTLE. This means that the cores in a processor can have different performances and power consumption. When the device is running lightly, use low-power cores, and on the opposite, use high-performance cores.



When designing a processor, it is necessary to consider the adoption of a large number of technical designs, which determine the performance and power consumption of the processor. Both Intel and ARM processors use pipelines when an instruction is decoded and ready to be executed, which means that the decoding is done in parallel.

To execute instructions faster, these pipelines can be designed to allow instructions to be executed out of program order (out-of-order execution /multi-lookahead). And we have tricks in the structure that can determine the dependency between instructions. Both Intel and ARM provide out-of-order execution logic structures. Of course, this structure is very complicated and costs more power consumption.

For example, the ARM Cortex-A53 uses sequential execution with less power consumption. The ARM Cortex-A57 uses out-of-order execution, which is faster but consumes more power. Processors using big.LITTLE architecture can have both Cortex-A53 and Cortex-A57 cores at the same time, and how to use these cores is determined according to specific needs (by the operating system).

Ideally, this architecture design works perfectly but it could also go much worse if the cores are not assigned well. The pressure goes to the operating system, and the challenge is back to the software level. Repeating my statement previously, I think the great combination of software and hardware is the key to why Apple M1 has so much performance while maintaining good power consumption in ARM architecture. In the future, the cooperation between hardware and software will be a major focus for ARM-based computer producers while competing with x86-based computer producers in terms of performance.

3. OPINION FOR THE FUTURE

The future of CISC and RISC (ARM and x86) is controversial. I would like to present some major opinions for furthering this discussion. “Moore’s Law is dead,” said David Patterson. Opinion from this side indicates that we are experiencing a slowdown in the development of hardware, where the number of transistors is no longer double every two years. However, we still require more computation, especially with the rise of machine learning in the past few years. RISC architecture seems to be a better path, which it’s easier for parallelism and manufacturing technology development. In addition to that, some advantages of CISC from the past are no longer holding, such as less register memory and faster programming speed. A few decades ago, programmers were still programming with these instruction sets called assembly language. CISC’s power instruction sets brought huge convenience to programmers. One line of code from CISC can do the same job as several lines of code from RISC. However, to this day, most programmers are no longer programming in assembly language. The compiler also grew stronger over time, and high-level programming languages can also run very efficiently on computers. Meanwhile, less register is not that attractive anymore, as the cost of memory has dropped a lot now. Process manufacturing technology development is also considered easier with RISC architecture, and it will be discussed in the next section.

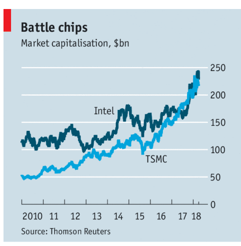
On the opposite, the supporter of CISC argues that x86 has higher historical status, and is compatible with basically all old software. There are many well-known software applications for industrial production, such as Autodesk Moldflow, Unigraphics NX, and AutoCAD. Many of these software applications were written in the 90s, under the x86 architecture. Moving from x86 to ARM will indeed take too much work for them, and this is a big advantage of x86 in historical status. In the early competition between CISC and RISC, ARM with RISC’s pipelining outperformed the x86’s CISC for a while. However, Intel micro-ops (since intel PENTIUM Pro) can disassemble its complex instruction set into several simple instructions, and x86 can now pipeline upon it. Intel micro-ops eliminated this advantage from ARM and gained back its No.1 position in PC. If there is no significant difference, CISC with higher historical status should remain in its position. CISC and x86 will always find the way.

Now, we exclude the above two controversial opinions from the industry and look at some academic experiments. According to the research “ISA wars: Understanding the relevance of ISA being RISC or CISC to performance, power, and energy on modern architectures” in the University of Wisconsin-Madison, they have concluded that if we exclude the impact of process manufacturing technology and operating system, the use of whether RISC or CISC doesn’t matter to the performance and power consumption of processors. The difference is all due to ISA-independent factors. This conclusion rejects the concept that the competition between ARM and x86 is mainly over the competition between the architectures both selected. If the architecture selected is not the key, then what’s the difference between them in competition? There can be a possible answer if we simply reverse the result of this research, and process manufacturing technology and the operating system/software should be the key.

3.1 Process Manufacturing Technology

According to the University of Wisconsin-Madison, the competition between ARM and x86 is not about the instruction set architecture selected, but instead the process manufacturing technology and operating system. In the past decades, Intel had the best process manufacturing technology for a long period. Things changed after Intel refused to enter the chip production industry for mobile phones. The consequence is that Taiwan Semiconductor Manufacturing Co., Ltd took up the boat of producing ARM chips for companies like Apple, and a large amount of investment went into it.

Today, Taiwan Semiconductor Manufacturing Co., Ltd offers a 3nm process manufacturing technology for Apple's M3. And Intel reached Intel 4 (equivalent to 7nm) for large production. Taiwan Semiconductor Manufacturing Co., Ltd's recent surpass in process manufacturing technology over Intel gives Apple better performance and power consumption on its chip while competing with Intel. In addition, for process manufacturing technology development, I think (just speculation) that Intel with CISC is more disadvantages than Taiwan Semiconductor Manufacturing Co., Ltd with RISC. The complexity of its instruction set will make Intel harder in making development. With similar manufacturing limitations, it's easier to fit simpler instruction sets into the chip, and the development will be easier for RISC. For example, two people are competing to draw as many figures as possible on a piece of paper. These two people have the same size of paper but one person is simply drawing a square figure and another is drawing an anime character figure. The job is much easier for the first person. This is currently one huge advantage of ARM architecture in competition with x86 architecture. On the opposite, Intel is also trying to find its way to keep Moore's law alive, such as by adding more dimensions and stacking up the transistors on the chip. It's very difficult to tell who will succeed eventually but time gives us an answer.



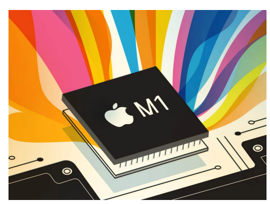
3.2 Software/Operating System

Except for better process manufacturing technology, the operating system and software can be considered another big advantage of Apple. The release of M1 not only stresses Intel but also Microsoft. Wintel Union now works closer than in the past because of the pressure. As I mentioned early in this paper, the cooperation of software and hardware in pipelining/parallelism is another key to the performance of the computer in the future, where both ARM and x86 (with Micro-Ops) can achieve it. And the work of parallelism must rely on better-written software and operating system. The level of cooperation between hardware and software determines how well the computer is going to perform.

Before Apple, Microsoft and Qualcomm have a PC product, which is Windows 10 plus Qualcomm snapdragon (ARM architecture). However, the level of cooperation between hardware and software is not high, and the completion level is low for this PC. The performance of this computer doesn't have much impact on the industry. I think that it's mainly due to the less attentive attitude of Microsoft toward this product. A main feature of RISC is the emphasis on software and its huge potential for optimization of software in this architecture. If Microsoft spends more effort working on the software for this product upon Qualcomm snapdragon, it's possible that it can reach a similar level to Apple. Unfortunately, this is not likely to happen since Microsoft and Qualcomm are two separate companies, and they can never work like Apple. Self-development in both hardware and software has become the greatest advantage for Apple in software and operating system, and Apple's dedication to optimizing the efficiency of cooperation between hardware and software made it today's success.

4. APPLE'S WORK

ARM and RISC architecture give Apple huge potential to work on its software for performance boosting. Rosetta 2 is the latest dynamic binary translator developed by Apple for its M chip in PC, which allow x86-based software efficiently run on PC. Apple also spends a lot of effort on its hardware and drivers, increasing communication speed. For the operating system, the joint optimization between macOS and M chip also brings the efficiency to a higher level, which is the typical optimization in cooperation between hardware and software as I mentioned above. Combining these works with some technological advantages of ARM (such as big.LITTLE architecture) in power consumption and better process manufacturing technology from Taiwan Semiconductor Manufacturing Co., Ltd, Apple shocked the whole industry in 2020, with the release of the M1 chip. It's indeed very impressive and innovative.



5. CONCLUSION

The success of Apple in the M1 chip is accomplished for many reasons. Its focus on optimizing the cooperation between its hardware and software and better process manufacturing technology makes Apple become the biggest threat to the Wintel union. As a company with such a market cap, it is not easy to achieve innovation like this in the industry. The difference between RISC and CISC seems not the case for the success of Apple. Under ARM/RISC architecture, the potential in software optimization is huge, making it possible to replicate the performance of x86 for Apple through a smarter dynamic binary translator Meanwhile, TSMC's current advantage with Intel's competition on process manufacturing technology further Apple's advantage.

6. REFERENCES

[1] Sagi, S. S. (2021). Difference between CISC and RISC - use, characteristics & advantages. EduKedar. Retrieved March 27, 2023, from https://edukedar.com/difference-between-cisc-and-risc/

[2] Sims, G. (2014). 冰峰雪晴. ARM vs X86 – Key differences explained! Retrieved March 27, 2023, from https://www.amkevin.com/272.html

[3] Intel. (n.d.). 12th gen Intel® core™ processor family: Adaptable scalability. Retrieved March 31, 2023, from https://www.intel.com/content/www/us/en/products/docs/processors/core/12th-gen-processors.html

[4] The Economist Newspaper. (2018). TSMC is about to become the world's most advanced chipmaker. The Economist. Retrieved April 1, 2023, from https://www.economist.com/business/2018/04/05/tsmc-is-about-to-become-the-worlds-most-advanced-chipmaker

[5] Blem, E., Menon, J., Vijayaraghavan, T., & Sankaralingam, K. (2015). ISA wars: Understanding the relevance of ISA being RISC or CISC to performance, power, and energy on modern architectures. ACM Transactions on Computer Systems, 33(4), 12:1-12:36. doi: 10.1145/2699682