Nicole Standiford

11/15/18

HNRS option CS270- ARM Architecture

Advanced RISC Machine (ARM) architecture is a type of Instruction Set Architecture that utilizes RISC, or Reduced Instruction Set Computer, methods to create fast processing and a higher capacity for advanced computations that improves the user interface. ARM architecture has four major components: instructions, a pipeline, registers, and load/store memory access with the arithmetic logic unit (ALU). Instructions are a basic set of commands that connect the hardware and the software of a computer. A pipeline is a system which utilizes the output of one process as the input of the next process. Registers allow for quick memory access. Load and store capabilities are essential to processing commands for any operation. ARM architecture is not the only RISC architecture, but it is unique.

Other RISC machines exist bit ARM architecture has several features that allow it to be superior to other RISC machines. A basic RISC machine has the capability to use a system that runs less cycles per instruction which speeds up processing time. ARM architecture uses this as well as different cycles for certain instruction from using multiple registers and load/store sets which increases speed and density of code and can usually reduce to a single cycle. It also uses a barrel shifter which shifts data with combinational logic instead of sequential logic, utilizing current data instead of data in memory. This makes instructions more complex which improves performance. ARM architecture also uses 16-bit Thumb instructions to improve code density when more bits are not needed. This architecture can also conditionally execute instructions to improve performance. It also has Digital Signal Processing (DSP) instructions that shorten time to understand user input and external input. Utilizing auto-increment and auto-decrement addressing modes is another superior feature. ARM architecture supports seven different types of exceptions and has an exception mode that utilizes specific registers. In short, it has many built in algorithms to make processing time faster and smoother.

The initial ARM architecture only came in a 32-bit version, but after generation seven a 62-bit version was implemented. The 32-bit version came with three profiles labelled Application (A), Real Time (R), and Microcontroller (M). Each of these profiles offered different CPU modes that allowed different levels of access to high risk aspects of the computer such as low memory and service routines. This was the main feature of the 32-bit version. When the 64-bit version was introduced, so were a various array of improvements that increased processing speed and the quality of the user interface. Changing the architecture from 32 to 64-bit didn't take away any of the features of the 32-bit architecture. When the 64-bit version was implemented so was a compatibility set of instructions so that both could still be used on any device. The improvements that came with the 64-bit version included a variety of new instructions that allowed more kinds of analytical processing. Some of these are, advanced floating-point instructions (VFP) and single instruction, multiple data instructions (SIMD), as well as new cryptography instructions.

There are some conflicting ideas in the explanation of ARM architecture in terms of instruction bits. It both claims to have 16-bit Thumb instructions that improves performance as well as a new and improved 64-bit version that improves performance. It leaves it unclear whether more or less bits are ideal. The answer is that it depends. Certain instruction only take 16 bits and to use 64 would be a waste of processing time and memory. On the other hand, some instructions are incredibly complex and require 64 bits. The ability of ARM architecture to conditionally use the appropriate number of bits to ideally fit the instruction situation allows it to be incredibly efficient without taking away functionality. Its ability to switch between data types is also carried into its ability to switch between types of instructions. ARM incorporates several different types of instruction sets and can access them when necessary but doesn't need to have them always available. This grants high functionality without using excess functioning time to sort through all the instructions contained.

Like any other instruction set architecture, ARM architecture uses registers, the stack, and subroutine calls to function. It can also map to C and C++ languages. Meaning that all the data types in the higher-level languages can be directly correlated to a data type used by ARM architecture at low levels. It uses bit codes as any other low-level language would, but it's complexity allows it to be much more easily linked to higher level languages, another feature that improves performance. Registers set can be extended by instructions in ARM architecture if more are necessary for certain functions. The stack is used to reduce the need for registers within this architecture as well. To handle data larger the 32 or 64 bits, ARM architecture uses consecutive registers to represent the larger data. In the 32-bit version of ARM there were only 16 registers available for use and now has a total of 37 registers. Of the 37, six are used as status registers. It can also utilize interrupt processes instead of polling to increase processing speed. By using a variety of different instructions, ARM has an algorithm for a vast number of functions. It even includes sub-architectures that allow it to access other instructions in certain states when necessary. While it has many of the same features and core aspects as any other instructions set architecture, ARM utilizes all of the best and most efficient aspects of instruction set architecture and combines them without encumbering its processes.

Many different kinds of technology have used ARM architecture over the years since it was released. Some older examples include PDA's and iPods. Newer examples include tablets, iPhones, cameras, navigation systems, and gaming consoles. Even Raspberry Pi's and other single board computers utilize ARM. On the other end of the spectrum, a highly specific and advanced computer built by Manchester University, the SpiNNaker, used ARM to simulate the human brain. The wide range of products that use ARM architecture give insight into how well it preforms compared to other architectures. It is clearly a frontrunner in terms of performance. ARM architecture has been used since 1981 and has continued to be developed and used up to this day. It has gone through 21 different versions with various improvements. It continuous to be improved upon and utilized in the newest technology.

The LC-3 gives a good micro example of the happenings in an ARM architecture while simplifying it enough to be understood. To fully understand ARM architecture, one could simply imagine what they know of the LC-3 and then multiply it by a million times of complexity. It is possible to write in the machine code for ARM architecture, as it is to do so in the LC-3, however it would be incredibly difficult and time consuming. Higher level languages are much easier to use to program functions. This is simply because instruction set architecture breaks down what we want to do into tiny units that we call bits. To use it is incredibly difficult, to create it even more so. The point being, ARM architecture is a feat of technology and truly incredible.

Figure 1: ARM1 2nd processor for the BBC Micro

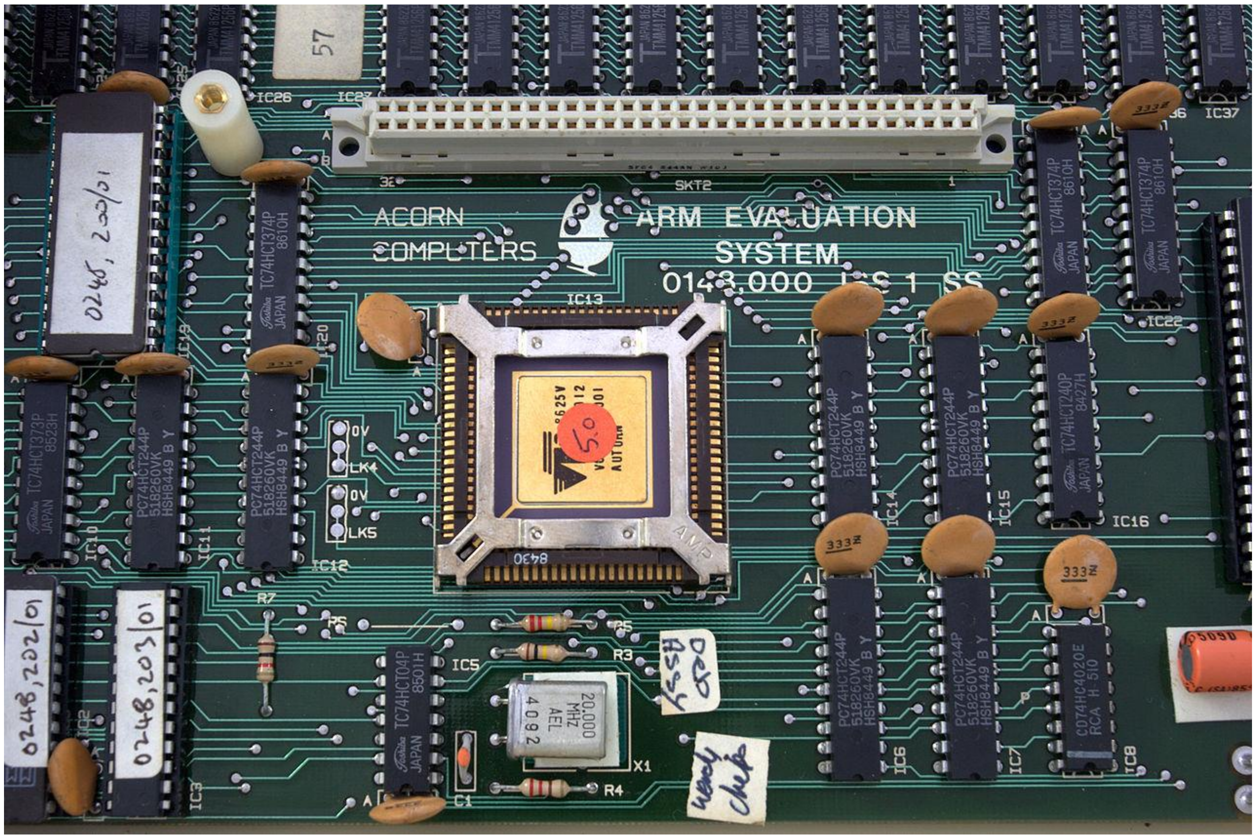
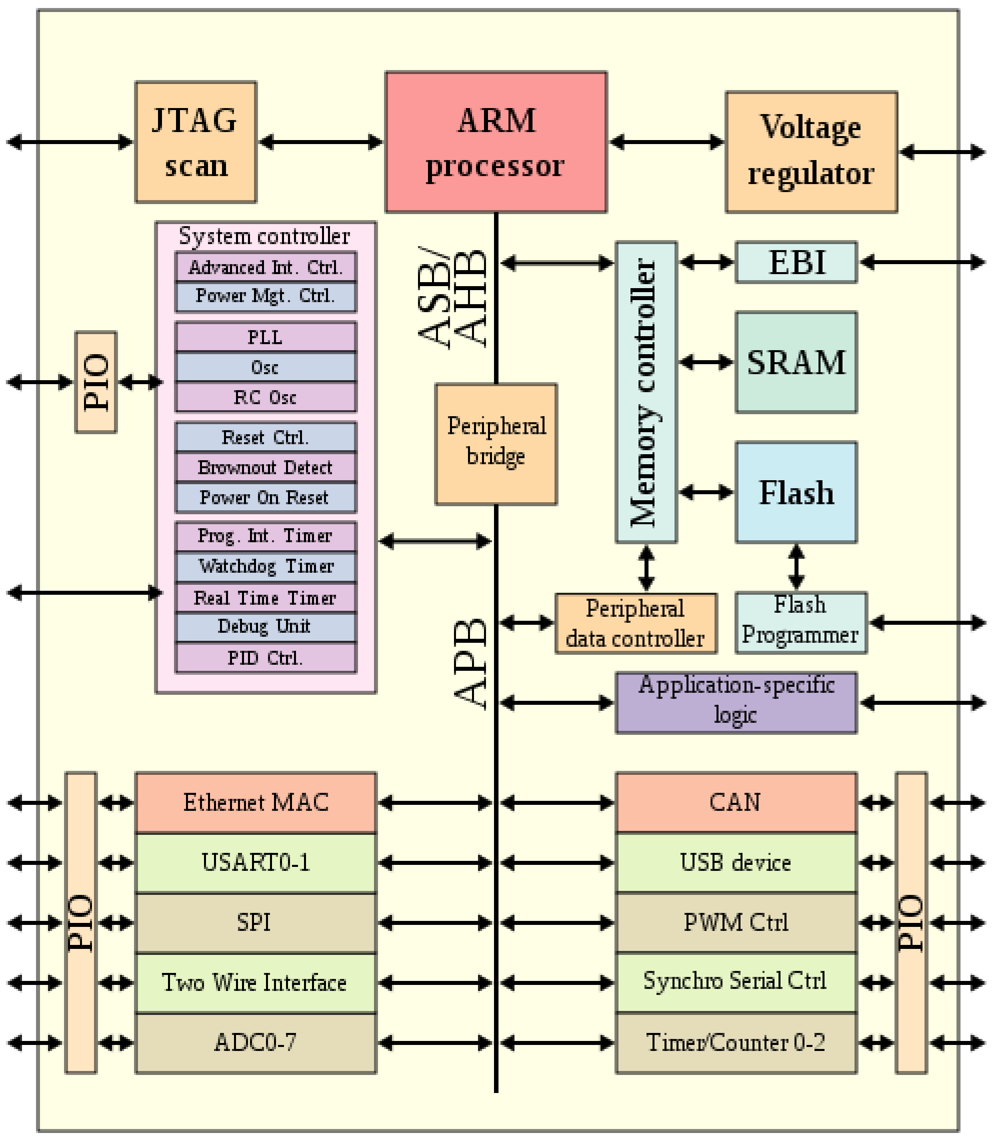


Figure 2: Microprocessor-based system on a chip



Works Cited

1. Andrews, Jason (2005). "3 SoC Verification Topics for the ARM Architecture". *Co-verification of hardware and software for ARM SoC design*. Oxford, UK: Elsevier. p. 69. ISBN 0-7506-7730-9. ARM started as a branch of Acorn Computer in Cambridge, England, with the formation of a joint venture between Acorn, Apple and VLSI Technology. A team of twelve employees produced the design of the first ARM microprocessor between 1983 and 1985. Retrieved 1 October 2018.
2. ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition. ARM Limited. p. B4-8. Archived from the original (PDF) on 6 February 2009. APX and XN (execute never) bits have been added in VMSAv6 [Virtual Memory System Architecture] Retrieved 12 October 2018.
3. "ARM Corporate Backgrounder" Archived 4 October 2006 at the Wayback Machine., *ARM Technology*. Retrieved 1 October 2018.
4. "ARM Cortex-R Architecture" (PDF). ARM Holdings. October 2013. Retrieved 1 October 2018.
5. "ARM Discloses Technical Details Of The Next Version Of The ARM Architecture" (Press release). ARM Holdings. 27 October 2011. Retrieved 20 September 2018.
6. "ARM DSP Instruction Set Extensions". *arm.com*. Archived from the original on 14 April 2009. Retrieved 18 September 2018.
7. Brash, David (August 2010). "Extensions to the ARMv7-A Architecture" (PDF). ARM Ltd. Retrieved 6 October 2018
8. "GCC 7 Release Series - Changes, New Features, and Fixes". The ARMv8.3-A architecture is now supported. It can be used by specifying the -march=armv8.3-a option.

[..] The option -msign-return-address= is supported to enable return address protection using ARMv8.3-A Pointer Authentication Extensions. Retrieved 20 September 2018.

9. “Procedure Call Standard for the ARM Architecture" (PDF). ARM Holdings. 30 November 2013. Retrieved 27 October 2018.
10. “ Processor mode". ARM Holdings. Retrieved 26 October 2018.
11. “Windows is coming back to ARM, this time with 32-bit x86 compatibility". *Ars Technica*. Retrieved 12 October 2018.