University of North Alabama

Computer Architecture and Orginization

Cortex A15 Processor

Author:

Jeffrey Allen

Professor:

Dr. Patricia Roden

October 15, 2014

Contents

1 History/Background of ARM Cortex-A15

In 1985, the Acorn Computer Group birthed the ARM architecture in the United Kingdom. This was only the beginning to the quick pace ARM began developing and evolving their architectural designs. Two short years afterwards, Acorn released its first RISC processor that was optimized for low-cost personal computers. Following this, in 1990, ARM which originally stood for Acorn RISC Machine developed its Advanced RISC Machine. This design defined its 32-bit RISC-like architecture generally used today.

According to Linda Null and Julia Lobor there are multiple families of ARM architectures and processors. The reason for different families depend on the applications of the architectures and processors. The architectures include include the ARM1, ARM2, which span to the ARM11 architecture. The processors are also categorized into different "series". This includes the M series, R series, and A series. The M series processors are optimized for microcontrollers. The R series has been designed for embedded systems and real time applications. Whereas the A series, which includes the A15, has been designed to handle full operating systems in third party applications. So just by looking that the name of the A15, it is implictly understood that it was designed and optimized for the latter domain of applications.

2 Memory Specifications

Memory in the ARM architecture models memory as a single array of 2^{32} 8-bit bytes. In total, that is 4,294,967,296 bytes of memory (or 34,359,738,368 bits. Whoo. That's quite alot to take care of).

In reviewing the ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition, the list of following data types are supported in memory:

Byte ́8 bits

Halfword ́16 bits

Word ́32 bits

Doubleword ́64 bits

Where the byte resides in memory, termed the byte address, ranges from 2^{32}

Load and store operations are able to transfer bytes, halfwords, or words to and from memory. The loading of bytes or halfwords zero-extend (pad with zeroes) or sign-extend (increasing number of bits of a binary number while preserving the number's sign and value) the data as it is loaded, or specified by the load instruction.

The A15's ARM instructions are word-aligned. This means that the size of the instructions are and must be divisible by 4. While its Thumb instructions are halfword-aligned. As for Java bytecode instructions, they are simply be byte aligned. Meaning it has no real limitations of where memory can be accessed.

Unaligned memory is supported with special instructions with ARMv7's instruction set.

3 Instruction Sets

I'll begin this section with explaining that an interface is an intermediary between two entities. Humans have utilized this concept in order to build rockets which have reached mars, create heart rate monitors, and even predict weather. A computer's processor is only designed to fetch, decode and execute various instructions. Before all of that, it must first be able to understand what its supposed to do. Since humans and computers are worlds apart when it comes to communicating with each other, the Instruction Set Architecture (ISA) interface was created. This interface saves humans from learning the exact strings of 0's and 1's the processor understands in order to carry out an instruction. Essentially, this is the agreed-upon interface designed for a machine essentially allows software to able to communicate with the hardware that executes it.

3.1 RISC Vs. CISC

Two types of processor design philosophies include Reduced Instruction Set Computers (RISC) and Complex Instruction Set Computers (CISC). CISC systems can be described through an example. Subtracting two integers is considered "simple". Then there is a type of instruction which can be found in a bubble sort. In array A, copying one element α located at A_x and "bubbling" it up in order to swap it with another element located at A_y found to be the α's correct position. CISC systems are based off of these complex instructions that performed multiple operations in a single instruction. These instructions which were used primarily in the 1950's through the 1970's and early 1980's because of the limitations of memory and cost (16KB of memory $\approx$ \$500) which architecture designers were able to optimize all instructions for a specific task.

RISC systems were designed with simplicity in mind, keeping all instructions small in order to execute the more efficiently. A notable difference between the two philosophies is that RISC assumes that required operands are not in memory, but directly in the processor's internal registers.

3.2 Main ISA: ARMv7-A

At the end of the day, it is all about communication. Moore's Law, states that the density of transistors on an integrated circuit is doubling ever 18 months. This law explains one of the main reasons why communication between humans and computers has changed, and continues to change so rapidly. The concept applied to the range of years between 1960 to 2014 describes a fair amount exponential increase in transitors, or rather, methods of communication. As Linda Null and Julia Lobor point out though, RISC in today's society something of a misnomer. They say that almost all recently created instruction sets are a mixture of RISC's and CISC's. It can be seen in practice with the Cortex-A15 which supports its main ARMv7-A architecture, supplemented with the Thumb, Advanced SIMD, and VFP architectures.

3.3 Extentions to ARMv7-A Instructions

Being that the A15 is the fourth generation of processors designed by ARM, it has evolved extensively. Good designs are kept and streamlined, while the bad died out. To keep up with the evolution of computation the A series began to be designed with ability to decode different instruction sets. Depending on what state, or "CPU mode", of operation of execution the processor is in, will determine how the instruction will be translated. The state indication is controlled by a simple switch of the designated T (Thumb) bit and J (Java) bit in the processor's Current Program Status Register (CSPR). This will be explained later in the "Registers" section. With the A15 having the ability to support multiple ISA's, this allows for is the Large Physical Address Extension (LPAE) architecture to enable virtual environments.

Why have support for virtual environments though? As stated before, the Cortex-A series is a family of processors which is only evolving with the times. More and more computation must be supported, so a processor needs to be versitale enough to keep up with the designs. Being that utilizes it multiple instruction sets, or extensions of them. These includes the ARMv7-A, Thumb, ThumbEE, NEON, and VFP instruction sets. ARM's A15 reference manual indicates on various occasions when to refer to a repective to describe it's own features.

3.3.1 Thumb & ThumbEE

This instruction set basically truncates the most commonly used 32-bit ARM instructions into 16-bit instructions. Therefore, creating simpler and more quickly executed instructions.

3.3.2 Jazelle

3.3.3 Advanced SIMD (Single Instruction Multiple Data)

Advanced SIMD extension is a media and signal processing architecture that adds instructions targeted primarily at audio, video, 3-D, graphics, image, and speech processing.

3.3.4 VFP (Vector Floating Point Coprocessor Extension)

This is instruction set that extends the ARMv7-A instructions. ALL VFP extension performs single-precision and double-precision floating-point operations.

3.3.5 NEON

4 Instruction Format

Since the ARMv7-A architecture is based off of the RISC design, all instructions are the same size. However, since the Cortex A15 extends this base ARMv7-A architecture, 64-bit instructions also exist. This is only in virtual states of processing though. In the many layers of abstraction that exist in a computer, the endianess of an architecture must be carefully observed. The endianess of of an architecture refer to the "byte order" of the computer. This can also be explained as the method of which a computer stores multiple-byte data elements. For example, the 4-bit string made up of 2 2-bit data elements 0001, can be read by a computer in little-endian as 4_{10}. Alternatively, read as big-endian it would be 1_{10} The ARMv7-A architecture exclusivley exhibits little-endian practices. The A15 however, has an interesting but dangerous characteristic which gives it the ability to configure itself between little-endian or big-endian through the use of ARM and Thumb instruction $SETEND\ BE$ which translates to 0, or $SETEN\ LE$ which translates to 1.

5 Registers

Essentially, a register is a place on the processor where data, instructions and state information is held. At any given moment in time, the processor is said to have access to

16 of these high speed memory locations which are 32-bits wide. Drawing from the concepts introduced in chapter 3, the need for virtualization is essential to todays computing standards. The different modes of execution supported by the A15 allows for this small set of 15 registers to actually be viewed as having a total of 37. Hurray for abstraction.

The A15 manual quite often times refers to the ARMv7 manual in order to get details regarding it. This is one of those times. The ARMv7 manual explains how these 15 registers are utilized:

- **Banked**
- 0: General Purpose
- 1: General Purpose
- 2: General Purpose
- 3: General Purpose
- 4: General Purpose
- 5: General Purpose
- 6: General Purpose
- 7: General Purpose
- **Unbanked**
- 8:
- 9:
- 10:
- 11:
- 12:
- 13: Stack Pointer

- 14: Link Register
- **Register 15: Program Counter**

6 Data Types

The following data types use these 32 bits to represent:

- 32-bit pointers
- Unsigned or signed 32-bit integers
- Unsigned 16-bit or 8-bit integers, held in zero-extended form
- Signed 16-bit or 8-bit integers, held in sign-extended form
- Two 16-bit integers packed into a register
- Four 8-bit integers packed into a register
- Unsigned or signed 64-bit integers held in two registers

Unsigned according to this processor's specification manual is described as a data value representing a non-negative integer in the range 0 to 2^{N-1}.

7 Addressing Modes

The method in which the address in memory is generated used by the load or store

instruction is defined as an addressing mode. For the sake of simplicity, I'll be only outlining main topics related to how addressing modes

8 I/O

9 Unusual Features

9.1 Limited Registers

One of the characteristics that make the ARM architecture unique is that only supports 16 general purpose registers. Compared to other architectures, this amount of registers can be called limited. All of the other major architectures, including the MIPS, SPARC, and the PowerPC architectures all support 32 registers.

To compensate for this limitation of registers though, virtualized environment instructions have been designed and implemented for this processor.

9.2 big.LITTLE with the A15 & A9

10 Uses/Applications

10.1 Mobile Computing

Ph.D. Steve Furber, one of the principal designers of the BBC Microcomputer System and ARM microprocessors when the "A" in ARM stood for "Acorn", had this to say when interviewed by Jason Fitzpatrick. "The ARM architecture is the most widely used 32-bit RISC architecture and the ARM processor's power efficiencyperforming the same amount of work as other 32-bit processors while consuming one-tenth the amount of electricityhas resulted in the widespread dominant use of the ARM processor in mobile devices and embedded systems." This whole research paper is based off of fascination related to ubiquitous computing. So when I decided to conduct research on what was behind the popular Samsung's Galaxy S4 successful mobile phone design, it was not a coincidence to find out it was ARM behind the processor design.

11 Contributions to the computer architecture landscape

11.1 Low Energy Computing