ARM Architecture Report

Perla Vanessa Jaime Gaytán
A00344428

Embedded Systems
(TE3059)

October 19th, 2020.

TABLE OF CONTENTS

Table of Contents **2**
Table of Figures **3**
1. Introduction **4**
1.1. Document Organization 4
1.2. Terminology 6
2. History **8**
3. Memory Formats **11**
3.1. Big-endian Format 11
3.2. Little-endian format 11
4. Processor Modes **13**
4.1. User Mode 13
4.2. FIQ Mode 13
4.3. IRQ Mode 13
4.4. SVC Mode 13
4.5. Abort Mode 14
4.6. Undefined mode 14
4.7. System Mode 14
5. Instruction Set **15**
6. Registers **16**
7. Pipelining **18**
7.1. Fetch Stages 18
7.2. Decode Stage 18
7.3. Execute Stage 18
8. Operating systems **19**
8.1. Historical OS 19
8.2. Embedded OS 19
8.3. Mobile Device and Desktop/Server OS 20
9. Power control **21**
9.1. Power management modes 21
9.1.1. Run Mode 21
9.1.2. Standby Mode 21
9.1.3. Shutdown Mode 21
10. Conclusion **22**

TABLE OF FIGURES

Figure	Page
Nokia 6610.	8
Semiconductor revenue.	9
ARM Cortex Processors.	10
Big-endian format	11
Little-endian format.	12
ARM Instruction Set.	15
Banked Register of an ARM CPU.	16
CPSR.	17
ARM Pipeline Stages.	18
Android OS which is primarily used on ARM architecture.	19

1. INTRODUCTION

In this document it will find information related to ARM Architecture: history, memory formats, processor modes, instruction set, registers, pipelining and OS. It also contains some conclusions regarding this investigation at the end and the references that were used in order to do this report.

1.1. DOCUMENT ORGANIZATION

This document is organized as follows:

Section 1 Introduction	It contains a short description of what content it can be fin don this document and the most important point on it. It also contains information relatable to understand this document.
Section 2 History	This section is related to the history of how ARM was developed and its motivation.
Section 3 Memory Formats	This section is about the different formats that ARM processors handle: big endian format and little-endian format.
Section 4 Processor Modes	This section explains the seven different processor modes that ARM has: User, FIQ, IRQ, SVC, Abort, Undefined and System.
Section 5 Instruction Set	In this section the Instruction Set is explained.
Section 6 Registers	This section is related to all the registers that ARM has and what are they for.
Section 7 Pipelining	This section explains how does pipelining works in ARM.

Section 8 **Operating systems**	This section shows some examples of which OS supports ARM and its most common ones.
Section 9 **Power Control**	This section shows all the power modes that this processor has in order to have a power control.
Section 10 **Conclusions**	In this section some conclusions are shown regarding the investigation on this document.
References	It contains the references that were used during the making of this document.

1.2. TERMINOLOGY

The following acronyms are used all along the document.

Acronym	Description
ADC	Analog to Digital Converter
CISC	Complex Instruction Set Computer
CPSR	Current Program Status Register
DAC	Digital to Analog Converter
DDR	Double Data Rate
DMA	Direct Memory Access
DSP	Digital Signal Processing
EMIO	Extended Multiplexed I/O
FIQ	Fast Interrupt Request
FPGA	Field Programmable Gate Array
GEM	Gigabit Ethernet MAC
GUI	Graphical User Interface
HDL	Hardware Description Language
HLS	High Level Synthesis
IRQ	Interrupt Request
LED	Light Emitting Diode
LSB	Least Significant Bit
LVDS	Low-Voltage Differential Signaling
MAC	Medium Access Control
Mbps	Mega Bits per Second
MMCM	Mixed-Mode Clock Manager

MSB	Most Significant Bit
NCO	Numerically Controlled Oscillator
OTR	Out of Range
OS	Operating System
PC	Personal Computer
PL	Programmable Logic
PLL	Phase Locked Loop
PS	Processing System
RISC	Reduced Instruction Set Computer
RMII	Reduced Media Independent Interface
Rx	Reception
SFP	Small Form-Factor Pluggable Transceptor
SoC	System on Chip
SoW	Statement of Work
SP	Stack Pointer
SVC	Supervisor Call
Tx	Transmission

2. HISTORY

ARM stood for ‘Advanced RISC Machines’ or ‘Acorn RISC Machines’ but nowadays it doesn’t stand for anything. This company was found in November 1990 as Advanced RISC Machines Ltd. This company was a joint between Acord Computers (the most popular creator of the BBC Micro in the 1980’s), Apple Computer and VLSI Technology, because Apple wanted to use ARM technology but didn’t want a based on Acorn IP, at that time considered as a competitor, product. Apple was the investor, VLSI Technology provided the tools, and Acorn provided 12 engineers. It’s first office was a barn in Cambridge. In 1993 the Apple Newton was launched on ARM architecture, but it has flaws which lowered its usability and they realized that the success doesn’t rely on only one product. After that, the ARM processor was licensed to many semiconductor companies trying to speed up their time in the market. [15]

The most important deal for the company was in 1993 with Texas Instruments (TI) which gave credibility to ARM and proved the successful viability of it. This event force them to formalize their licensing business model and make more cost-effective products. During the mobile revolution in 1994, Nokia wanted to use ARM based system design, but their greatest concerns were about memory because of overall system cost to produce. This led to ARM creating a custom 16 bit per instruction set that lowered the memory demands. The first ARM powered GSM phone was the Nokia6610, show in Figure 1, and it was a massive success. [17]

Figure 1. Nokia 6610.

In the late 1990's ARM was one of the highest value companies of technology companies valuing it at over 300 times its actual earnings for 1999. ARM was ranked 30th on the FTSE 100 index of more valuable companies. However, in the early 2000's, the technology sector crumbled and devalued overall on the stock market by 80-90%. In the Figure 2 this decline can be better illustrated. [18]

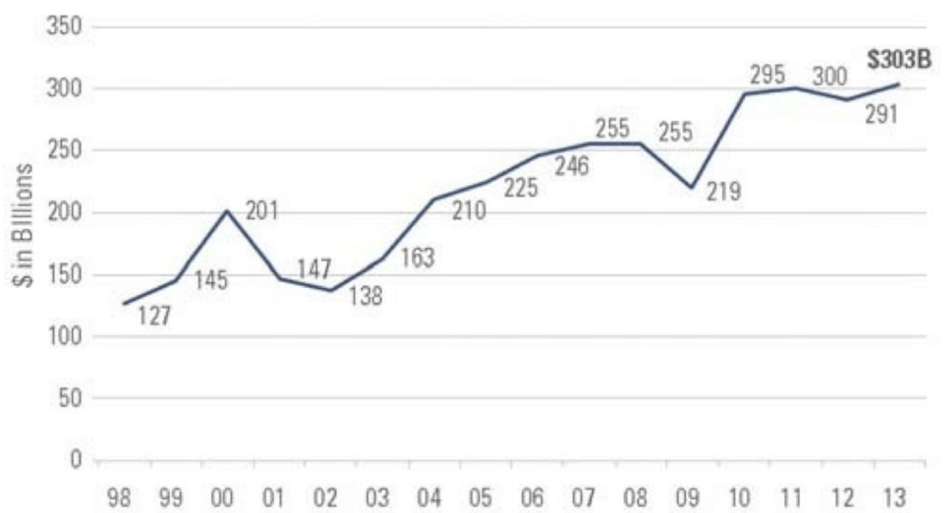

Figure 2. Semiconductor revenue.

By this time, microprocessors had become so small that they only occupied a small part of the chip, so the issue was how to build software-based systems on a single chip or SoC solutions. The majority of companies didn't have design teams with the ability to build their own microprocessor, or the tools needed to make them usable, therefore, microprocessors were one of the first to use IP license model and ARM was designed more and more SoCs. In 2001 the ARM926EJ-S was announced, and it was fully synthesizable with a 5-stage pipeline and an integrated MMU and some DSP operations. It became licensed by over 100 silicon vendors worldwide and has gone on to ship multiple billions of units. After that, the ARM9 became the new ARM7, which became the ARM9E, and then the ARM10 and ARM11. The ARM10 technology used was low power and high-performance processing, which was new ground for the industry. ARM decided to diversify the offering to cover all the need of the industry because they realized that they couldn't continue the upwards. [3]

The diversity that ARM brought to the industry was: Cortex family. Cortex-A continued the offering of mobile applications, followed by the higher performance demand. Cortex-R provided high performance, real time processors that catered for the highly specialized real-time requirements. Cortex-M provided extremely low power, low cost cores to the micro-controller industry.

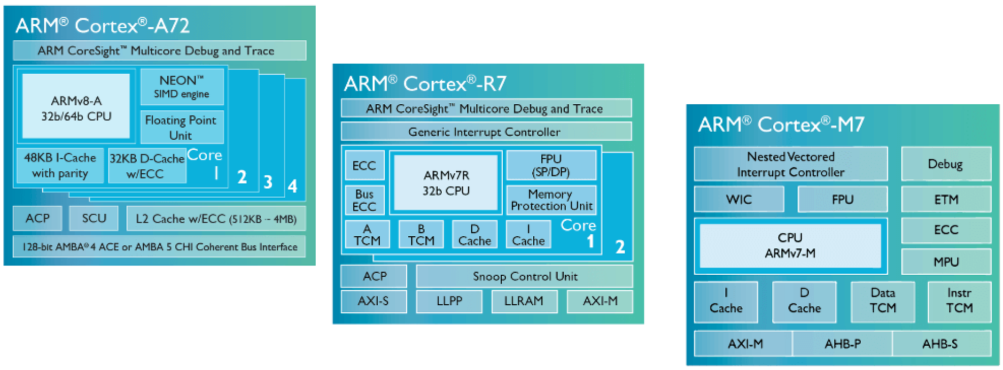

Figure 3. ARM Cortex Processors.

In 2008, the smartphone market demands an increased performance, while maintaining a long battery life presented a challenge. ARM responded with the Cortex-A9 MPCore, which was better able to address the huge dynamic range in processing. ARM currently has 96% share in the mobile market and shows no signs of slowing down.

3. Memory Formats

The ARM processor views memory as a linear collection of bytes numbered in ascending order from zero. Bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. This processor can treat words in memory as being stored in a 32-bit word-invariant big-endian format and Little-endian format. [4]

3.1. Big-endian Format

In 32-bit word-invariant big-endian format, the ARM processor stores the most significant byte of a word at the lowest-numbered byte, and the least significant byte at the highest-numbered byte. Therefore, byte 0 of the memory system connects to data lines 31-24. This process is shown in Figure 4. The architecture has evolved over time, and version seven of the architecture defines three architecture profiles. The application profile (A-profile) is implemented Cortex-A, the rela-time profile (R-profile) is implemented by Cortex-R and the microcontroller profile (M-profile) is implemented by Cortex-M.

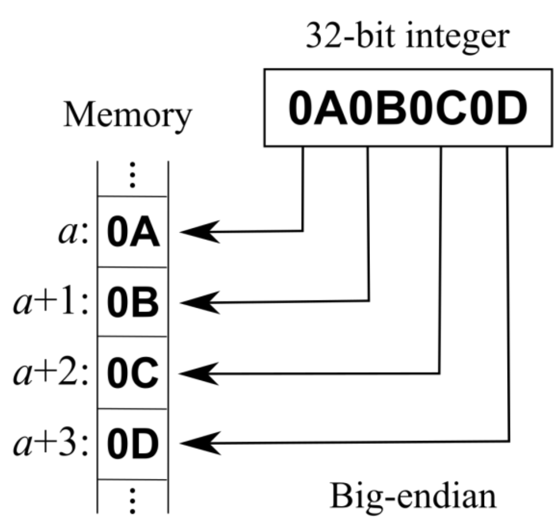

Figure 4. Big-endian format

3.2. Little-endian format

In little-endian format, the lowest-numbered byte in a word is the least significant byte of the word and the highest-numbered byte is the most significant.

Therefore, byte 0 of the memory system connects to data lines 7-0. This process in shown at the Figure 5.

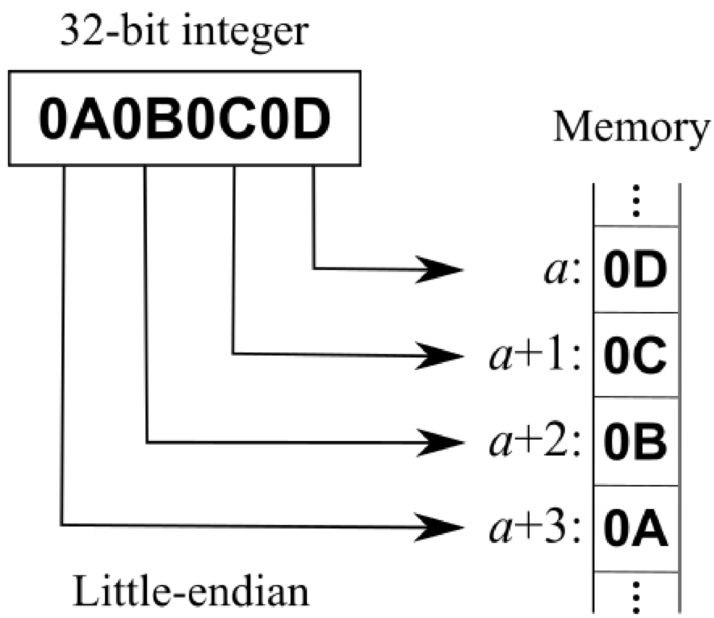

Figure 5. Little-endian format.

4. PROCESSOR MODES

There are seven processor modes: User, FIQ, IRQ, SVC, Abort, Undefined and System. All these modes are privileged modes except for User mode which is Unprivileged Mode. The Exception modes includes: SVC, FIQ, IRQ, Abort and Undefined. [14]

4.1. USER MODE

This mode is where most of the application or OS tasks run.

4.2. FIQ MODE

ARM supports two types of interrupting handling. The FIQ is the first one that ARM supports. It entered in FIQ Mode whenever a high priority, determined as fast, interrupt is raised. This mode provides a large number of shadow registers, from R8 to R14, and is useful when it must complete extremely quickly, or else, data loss is a possibility. The original ARM used FIQ used FIQ for networking and floppy disc which had to be serviced as soon as data was available. Modern ARMs would use FIW for high speed DMA-style transfer.

4.3. IRQ MODE

The IRQ is the second type of interrupting handling that ARM supports. It entered here whenever a normal priority interrupt is raised. For this mode only R13, R14 and CPSR are shadowed. All interrupts that do not require extreme speed will use IRQ Mode.

4.4. SVC MODE

OS calls set the processor to SVC Mode, and then the processor jumps to &8 (or &FFFF0008) and it resets. After the system is reset, the ARM begins processing at address &0 or &FFFF0000 (if high vectors configured). This address is the location of the Reset Vector, which should be a branch to the reset code. For this mode only R13, R14 and CPSR are shadowed.

4.5. ABORT MODE

This mode is used to handle memory access violations. It is done as result of a failure to loads either an instruction (Prefetch Abort) or data (Data abort). A Prefetch Abort occurs when the processor attempts to execute to load an instruction and failed. In ARMv5 this can occur programmatically using the breakpoint instruction. A Data Abort occurs when the processor attempts to fetch data, but the memory system says is unable to. The abort occurs before the failed instruction alters the processor state. For this mode only R13, R14 and CPSR are shadowed.

4.6. UNDEFINED MODE

It is used to handle undefined instruction when it is encountered. For this mode only R13, R14 and CPSR are shadowed.

4.7. SYSTEM MODE

This is the only mode that is not entered by an exception. It can only be entered by executing an instruction that explicitly writes to the mode bits of the CPSR from another privileged mode.

5. INSTRUCTION SET

The 32-bit ARM architecture includes the following RISC: Load/store architecture, does not support unaligned memory accesses, uniform 16x32-bit register file, fixed instruction of 32 bits to ease decoding and pipelining, and single clock-cycle execution. Some additional design features ar: arithmetic instructions, 32-bit barrel shifter, indexed addressing modes, link register, and a 2-priority-level interrupt. The general ARM Instruction Set is shown in Figure 6. By overlapping the various stages of operation, the ARM processor maximizes the clock rate achievable to execute each instruction. It delivers a throughput greater tha one instruction for each cycle.[6]

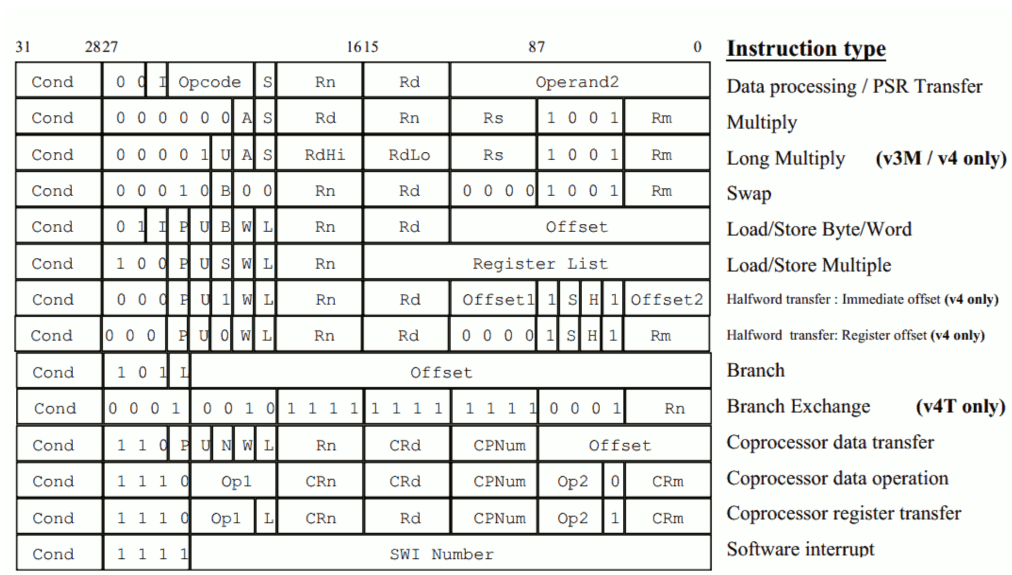

Figure 6. ARM Instruction Set.

6. REGISTERS

In the ARM processor there are sixteen general registers and one or two status registers that are accessible at any time. In privileged modes, mode-specific banked registers become available as show in Figure 7. The registers from R0 to R15 are directly accessible. The CPSR contains condition code flags, status bits, and current mode bits. Registers from R0 to R13 are general-purpose registers use to hold either data or address values. Registers R14, R15 and CPSR have special function such as: Link Register or Program Counter. Register R13 is referre to as SP. Register R14 is used as the subroutine Link Register which receives the return address when a Branch with Link instruction us executed. The R15 holds the Program Counter which state this is a word aligned. The CPSR has 32 bits which is organized as shown in Figure 8. [8][16]

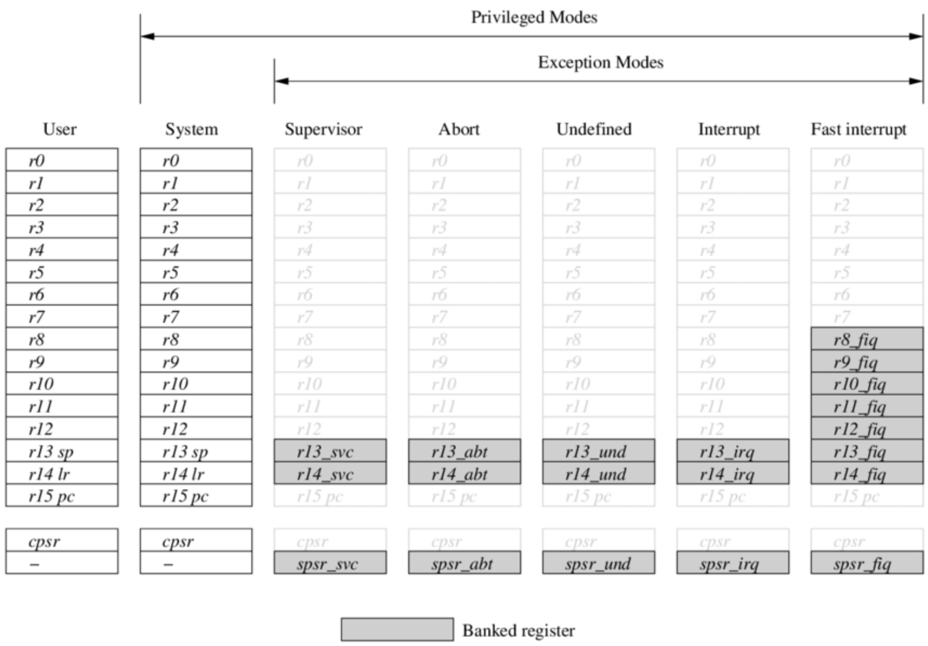

Figure 7. Banked Register of an ARM CPU.

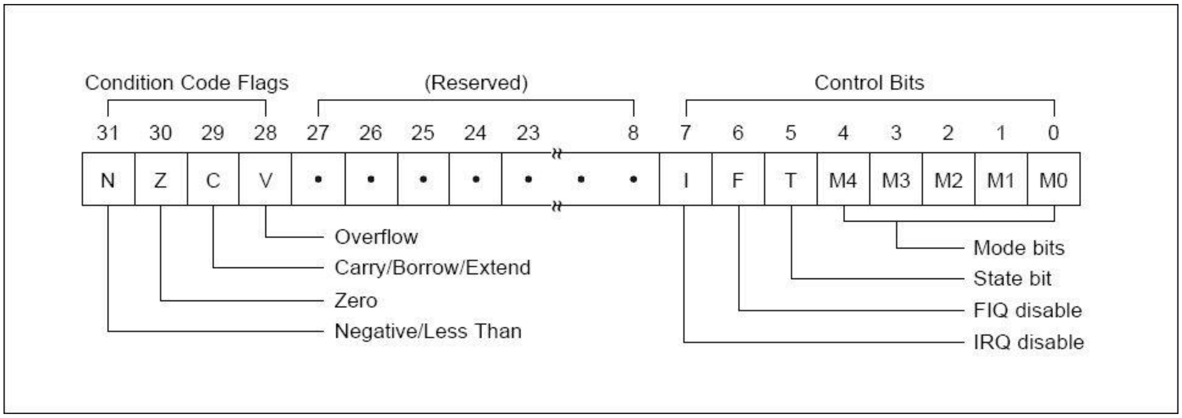

Figure 8. CPSR.

7. Pipelining

Instruction pipelines allow to overlap execution of multiple instructions with the same circuitry.[2] The circuitry is usually divided up into stages and each stage processes a specific part of one instruction at a time. The ARM processors have a three-stage pipeline: fetch, decode and execute. These stages are shown in Figure 9. [9][13]

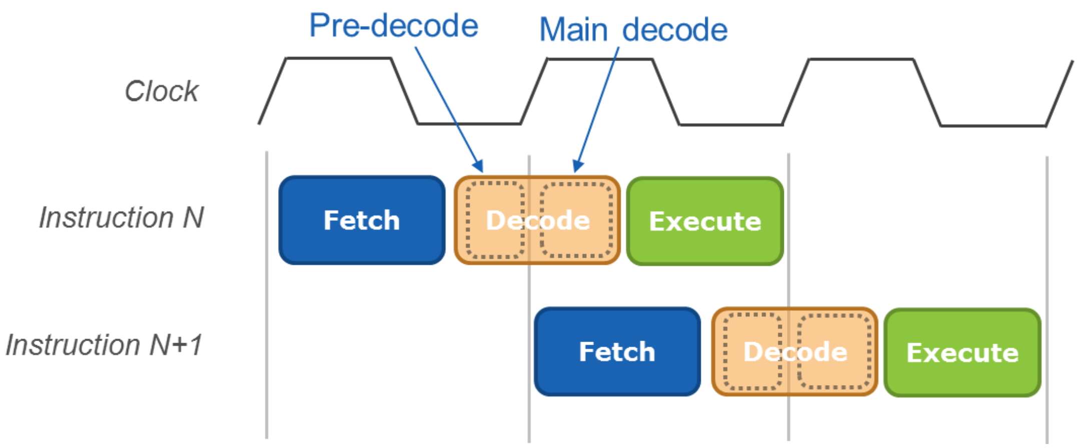

Figure 9. ARM Pipeline Stages.

7.1. Fetch Stages

The Fetch Stages can hold up to four instructions and it's where branch prediction is performed on instructions ahead of execution of earlier instructions. In the first stage of Instruction Fetch, the address is issued to memory, in the second stage the memory returns data to core, and in the third stage is for branch prediction and Thumb-2 instruction alignment.[5]

7.2. Decode Stage

The Decode stage can contain any instruction in parallel with a predicted branch. This stage is where the instruction is analyzed and interpreted it. [7]

7.3. Execute Stage

The Execute stage can obtain a predicted branch, an ALU or multiply instruction, a load/store multiple instruction or a coprocessor instruction un parallel execution. [12]

8. Operating systems

In this section it will be shown some examples of 32-bit operating systems that supports ARM architecture.[10]

8.1. Historical OS

The first 32-bit ARM-based personal computer was the Acorn Archimedes but it was originally intended to run an ambitious operating system called ARX. The machines shipped with RISC OS which was also used on later ARM-based systems from Acorn and other vendors. Some early Acorn machines were also able to run a Unix port called RISC iX. [11]

8.2. Embedded OS

The 32-bit ARM architecture is supported by many embedded and real-time OS. The most common are: Android (see Figure 10), A2, Linux, RIOT, Windows 10 IoT Core, and OS-9.

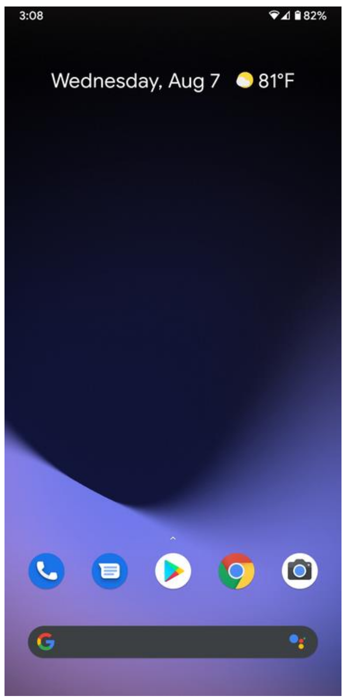

Figure 10. Android OS which is primarily used on ARM architecture.

8.3. Mobile Device and Desktop/Server OS

The 32-bit ARM architecture is the primary hardware environment for most mobile devices such as: Android, BlackBerry OS, Chrome OS, Firefox OS, Ubuntu Touch, Windows Mobile, Windows Phone, Windows 10 Mobile, among others. The 32-bit ARM architecture also supported by RISC OS and by multiple Unix-like OS like: OpenBSD, OpenSolaris, Debian, Gentoo, Ubuntu, Raspberry Pi Os, and others.

9. Power control

The ARM processor includes features that improve energy efficiency. These features include an accurate branch and return prediction which reduce the number of incorrect instructions fetch and decode operations. In order to reduce the number of cache flushes, saving energy in the system, include physically addressed caches. Also, the caches use sequential access information in order to reduce the number of accesses to the Tag RAMs. [1]

9.1. Power management modes

The ARM processor supports three types of power management modes: run mode, standby mode, and shutdown mode.

9.1.1. Run Mode

Run mode is the usual mode which all of the functionalities of the processor are available.

9.1.2. Standby Mode

Most of the clocks are disable of the device in this mode, while keeping the design powered up. The transition to this mode is caused by three main reasons which are: an interrupt, a debug request or reset.

9.1.3. Shutdown Mode

In this mode the entire device is powered down. The processor returns to the Run Mode by doing a reset.

10. CONCLUSIONS

In this document the ARM processor is being described. All the main features of this processor are discussed on it. There are some major important features presented such as: History, Registers and Pipelining. It was presented the beginning of the ARM company and how they become important on the industry. It also presents all the Registers that this processor uses and how they are being used. It also explains what the modes of this processor are, and which registers they use on each mode. This information is important to know in order to use whichever processor of this line because it explains how they work and the most important things that everyone should know before used any of these processors. For me it was important to did this investigation to know more about the processor that we are using on the laboratory class.

References

[1] *About power control.* . (2020). Arm Developer. Recovery from: https://developer.arm.com/documentation/ddi0290/g/power-control/about-power-control?lang=en

[2] Ahlawat, A. (2020). What is Pipelining?. Recovery from: https://www.studytonight.com/computer-architecture/pipelining#:~:text=Pipelining%20is%20a%20technique%20where,increases%20the%20overall%20instruction%20throughput.

[3] Alonso, R. (2020). *Todo lo que necesitas saber sobre los procesadores ARM.* Recovery from: https://hardzone.es/tutoriales/componentes/procesador-arm/

[4] *ARM architecture.* (2020). Wikipedia. Recovery from: https://en.wikipedia.org/wiki/ARM_architecture

[5] *ARM Cortex-M0+Pipeline.* (2020). Microchip Developer Help. Recovery from: https://microchipdeveloper.com/32arm:m0-pipeline

[6] *ARM Instruction Set.* (2020). ARM. Recovery from: https://iitd-plos.github.io/col718/ref/arm-instructionset.pdf

[7] *Decode.* (2020). LEXICO Powered by OXFORD. Recovery from: https://www.lexico.com/definition/decode

[8] Jinual, J. (2020). *ARM ORGANIZATION-REGISTER BANK.* Recovery from: https://letsembed.wordpress.com/2013/12/15/arm-organization-register-bank/

[9] *Memory Formats.* (2020). Arm Developer. Recovery from: https://developer.arm.com/documentation/ddi0290/g/programmer-s-model/memory-formats?lang=en

[10] *Operating modes.* (2020). Arm Developer. Recovery from:https://developer.arm.com/documentation/ddi0290/g/programmer-s-model/operating-modes?lang=en

[11] *Operating Systems.* (2020). Arm Developer. Recovery from:https://developer.arm.com/solutions/os

[12] *Pipeline stages.* (2020). Arm Developer. Recovery from: https://developer.arm.com/documentation/ddi0290/g/introduction/pipeline-stages?lang=en

[13] *Pipeline stages.* (2020). Arm Developer. Recovery from:https://developer.arm.com/documentation/ddi0290/g/introduction/pipeline-stages?lang=en

[14] *Processor modes.* (2012). ARM wiki. Recovery from: https://heyrick.eu/armwiki/Processor_modes

[15] Sengar. S. (2015). *ARM Architecture Basics.* Recovery from: https://saurabhsengarblog.wordpress.com/2015/12/05/arm-architecture-basics/

[16] *The register set.* (2020). Arm Developer. Recovery from:https://developer.arm.com/documentation/ddi0290/g/programmer-s-model/registers/the-register-set?lang=en

[17] Walshe, B. (2015). *History of Arm: Part 1.* Arm Community. Recovery from:https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/a-brief-history-of-arm-part-1

[18] Walshe, B. (2015). *History of Arm: Part 2.* Arm Community. Recovery from: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/a-brief-history-of-arm-part-2