ARM in Nintendo 3DS

ECE332 Project Final Draft

Rose-Hulman Institute of Technology

Weimu Song
Yuqi Zhou

Abstract

The Nintendo 3DS is a handheld game console announced by Nintendo in 2010. Unlike any other game consoles in the market, Nintendo 3DS inherited the traditional double-screen design from Nintendo DS, and also added a brand new fancy function called autostereoscopy which allows user to view stereoscopic (3D) graphics without wearing any devices, and the performance is quite powerful in 2010 when smartphones are not as popular as recent years. Because of its special design and portability, 3DS quickly became the most popular handheld game console at that time. This paper is going to look into the architecture of 3DS and analyze how ARM11 and ARM9 CPU works in this console.

3DS Architecture Introduction

3DS is powered by two ARM11 CPUs running at 266MHz (Red in Figure 1), and a DMP PICA200 GPU clocked at 133MHz. It features 6MB of VRAM dedicated to graphics, 2*64MBs of RAM, and 1GBs of flash memory for storage [1].

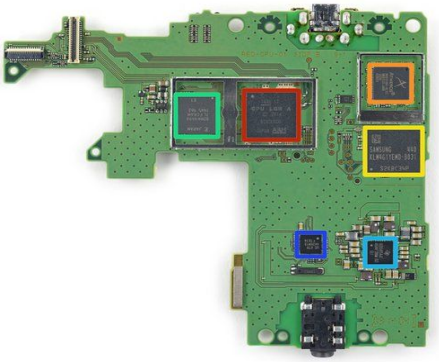

Figure1: 3DS Motherboard [1]

First is the GPU, the DMP PICA200 GPU is a 2006 chip designed for portable devices. It is not on the same level as the GPU of iPhone 4, which was announced in the same year. However, it is relatively cheap and provides efficient graphics to the device. In normal condition, the chip's vertex performance has a maximum 15.3M polygons/sec rate when running at 200MHz. However, the vertex performance was limited to no more than 5M polygons/sec in 3DS.

One of the reasons that Nintendo chose to lock the GPU speed is to exchange for longer battery life. The power will be quickly consumed if the GPU is in full speed. The other reason is about memory, the 3DS does not have enough graphic RAM in order to hold that many textures. Most of the computation power is used to obtain a stable 60fps graphics, so users can feel comfortable when they are using autostereoscopy. Therefore, in some aspect, the new autostereoscopic functionality is an obstacle that somehow limited the full power of the DMP PICA200 GPU[2].

Next is the CPU(s), there are two processors in 3DS. One is the ARM11 MPCore processor and the other is ARM9 processor. The ARM11 processor is a popular portable CPU that has already been used for many years. The first version of ARM11 was produced in 2002. Some famous products like iPhone 2G and 3G also chose to use this CPU. The one that is used on 3DS is ARM11 MPCore, which is a dual-core processor, and it makes 3DS became the first dual-core portable device in the world. These cores were divided to work on different functionality. One core only works on system softwares and the other core is used for applications, such as games and general softwares [3]. This two-processor architecture is inherited from the Nintendo DSi, which is the previous generation of Nintendo 3DS, though DSi used older processors (ARM9 and ARM7). One advantage of using the same architecture is it allows 3DS compatible with the original DS, which makes it more competitive in the market.

ARM Architecture

The last part of the article presents that ARM processors have not only served many generations of Nintendo portable consoles, but also a popular choice for other portable devices. The following sections about ARM processor are going to show the advantage of this processor and the reasons why Nintendo has this special preference to the ARM processors.

a. History

Advanced RISC machine (ARM) is "a family of reduced instruction set computing (RISC) architectures for computer processors [4]." In the beginning, it was named "Acorn RISC machine", developed by the British computer manufacturer Acorn Computers in the 1980s. The architecture was at first used only in the company's personal computers. The architecture was so successful that the team considered designing a processor and publishing it officially. Sophie Wilson, a key member of the team, convinced the CEO of the company who then assembled a team and let them implement the hardware.

The first ARM-based computer was published in 1987, named Acorn Archimedes. Acorn Archimedes is also known as ARM2, which featured "32-bit data bus, 26-bit address space, and 27 32-bit registers [5]." The address spaces only had 26 bits so the top six bits of the program counter register served as status flags. Even though the address bus was extended to 32 bits in the later products, "program code still had to lie within the first 64 MB of memory in 26-bit compatibility mode."

From the 1980s, Apple computer and VLSI Technology, which became Philips Semiconductor, chose to work with Acorn Computers on the following versions of ARM. The product developed by the three companies came out in 1992, named ARM6. ARM610, an ARM60-based processor, was used in Apple's newest product at that time, Apple Newton PDA.

After ARM6 was published, the company published more versions of ARM processors in a short period of time. From [5], it turned out that between 1996 and 1999, the ARM7 family and ARM9 family were published. In the following year, the company published ARM10 and ARM11 family.

Since ARM6, the ARM processor has been widely used in mobile phones, laptops, and other mobile devices. In 2005, less than 2 percent of all mobile phones did not use ARM processors. In 2013, a report showed that over 60 percent of mobile devices use ARM processors.

It is not surprising that nintendo kept using the ARM processors for its machines. Both 3DS and new 3DS used ARM11 and ARM9 processors.

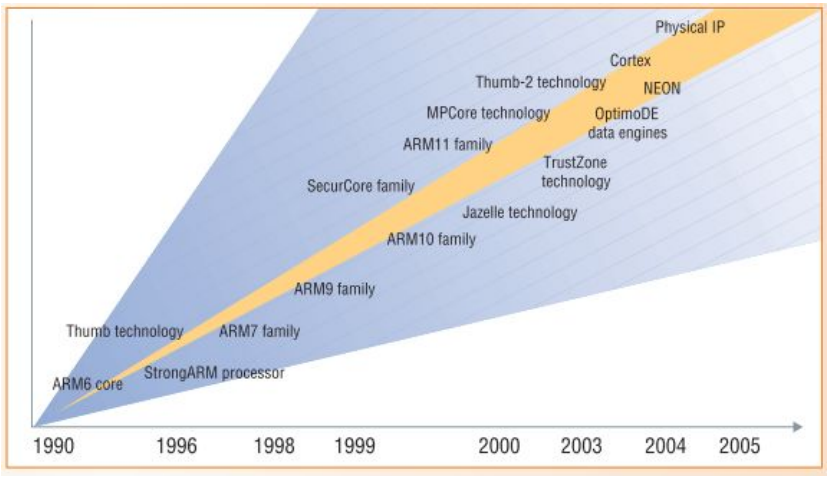

Figure2: the history of ARM family since ARM6, taken from [5]

b. Special Features

i. Variable cycle execution

Compared to reduced-instruction-set computing (RISC) processors, like MIPS, one of the biggest improvements ARM family have made is variance cycle execution [5]. In MIPS, the instructions can only load and store one register every time, which will be very inefficient and time-consuming. In ARM, however, some instructions can load and store several registers at the same time, which is similar to what we learned in class.

With shorter waiting time, the programs can finish earlier, requiring less instruction fetch and execution. At the same time, the system does not have to pay additional power for the "bubbles" in the pipeline, so the overall power consumption decreases.

In conclusion, this approach can eliminate the waiting time for the instructions, increase code density, and save power.

ii. Inline barrel shift

Inline barrel shift is a technique ARM uses to make the operations faster. It preprocesses one of the operands before putting them into ALU and calculates them [5]. As we understand it, the data of one operand is shifted or rotated before operations so that it takes less time to calculate in ALU. As we learned in class, some operations in ALU can take several cycles to complete, such as multiplication and division. Although rearranging the instructions or combining two loops into one loop can also solve the problem and increase the code density (code density refers to the combined size of instructions of a task [14]), the best way should be directly reducing the number of cycles.

From our understanding, this approach is similar to the extensions of MIPS, but in ARM, as the toolchain says, the preprocessing is a part of the instructions, so it gives the system more flexibility.

iii. Conditional execution

Another useful feature of ARM is conditional execution. If some conditions of the instructions are not fulfilled, the system can run other instructions and keep the instructions lacking conditions waiting until they have all the resources needed. For example, an instruction adds values from r0 and r1, but the value of r1 has not been loaded. Then the instruction will not do the calculation before the value of r1 is available and executable instructions after it can use the ALU instead.

This approach can be done by "ALU status flag" [6], which is set by the previous instructions, telling current instruction whether it has all the data needed or not. The flag is always set to be true for the instructions at first. If one instruction depending on a flag "thinks" other instructions depending on

the same flag need to wait, it will set the flag to false. In this way, the other instructions will wait until the flag is set to true again.

Conditional branch instructions can also be used to control the execution flow [6]. They are not counted as "real" instructions in the original program, but they execute in the same way as other branch instructions. Instead of comparing registers, they will check the flags.

iv. Thumb instruction set

Another approach ARM uses to reduce code density is thumb instruction set [5], using 16-bit instructions instead of 32. The structure and compiled instructions of thumb are different from the normal ARM. With a lack of 16 bits, the thumb has less functionality. For example, instructions for loading storing can use registers r0 to r12 in normal ARM, but in the thumb, they can only use r0 to r7. However, other than that, the thumb is as powerful as the normal ARM while saving much more spaces and significantly reducing the code density.

The same assembly code can be compiled to both ARM and thumb instructions for any processor [6]. The processor will tell the difference based on the least significant bit of the instructions. It does not determine the type based on length because thumb2, an evolution of thumb, use both 32-bit and 16-bit instructions. Thumb2 has almost the same functionalities as the normal ARM but reduces code density just like the thumb. Fig.3 [7] shows an example of the thumb instructions.

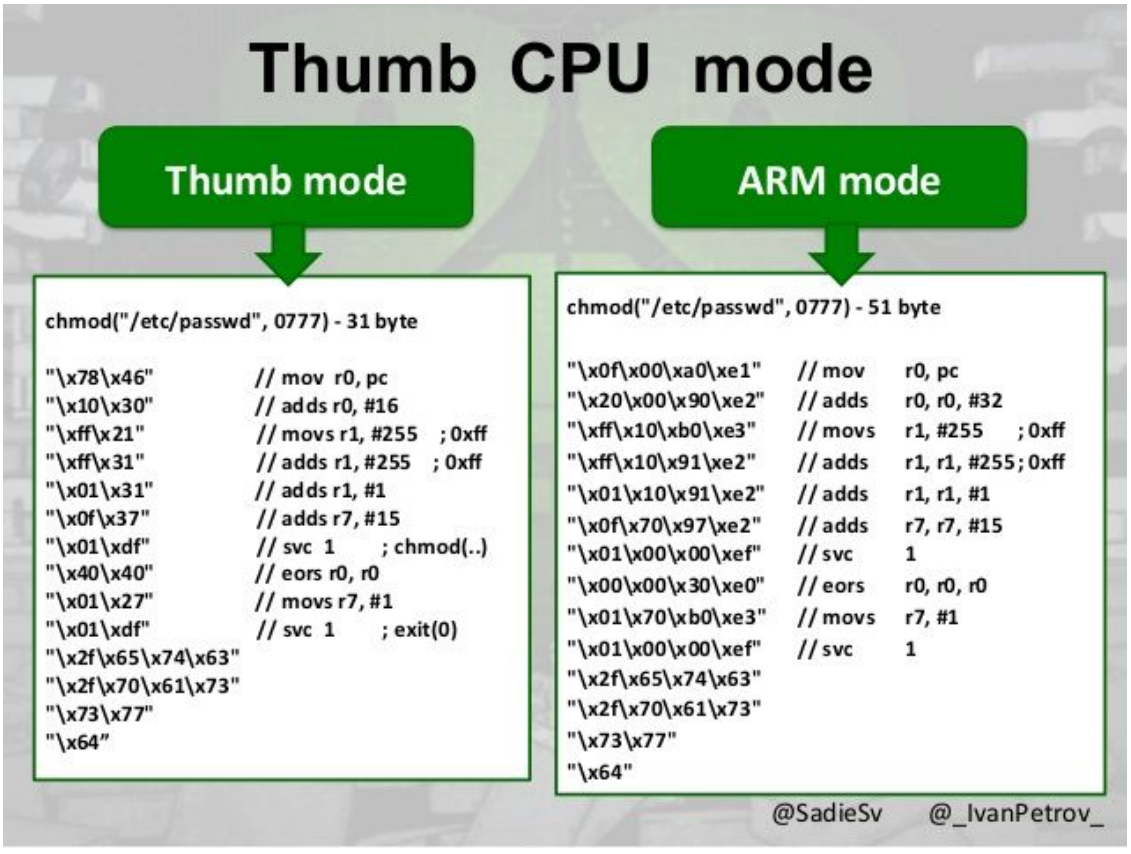

Fig.3 comparation of normal ARM instructions and thumb instructions [7]

ARMs in 3DS

a. 3DS and ARM9

i. Why 3DS Uses Two CPUs

One of the most important reasons to add an ARM9 processor is that NDS (Nintendo DS), the handheld game console before 3DS, uses ARM9 as the main processor [8]. Nintendo wanted the players to be able to play the games on NDS, which was like a "tradition" of its game consoles (players can also play games of GBA on NDS). Therefore, they kept the ARM9 processor for those games. Since ARM11 was more powerful than ARM9, they used an ARM11 processor as the main processor.

ii. Security

To further explain how ARM9 works on 3DS, we have to start with the issues of security. In fact, NDS is known for its insecurity. Since it was published, hackers had been trying to emulate its gamecards, wifi-connections, etc. For example, in 2007, 800, 000 units of games of NDS were sold, but the pirate hardwares were also "transacted" for more than half of the number [9].

To avoid making the same mistake, Nintendo decided to use a new architecture to protect the security of its new console. First, the ARM9 processor can only do some low-level stuff, like file accessing to NAND (a storage technology to store data without power) and AES/RSA engine (used for encryption of electronic data based on AES/RSA, two specifications for that) [8, 10, 11].

Second, when it needs to "communicate" with the ARM11 processor, it should pass through a series of a trust hierarchy. Users can run old games in the compatibility mode of 3DS, which is not harmful and does not need to be checked before passing to ARM11. However, when the user wants to communicate with ARM11 through ARM9 in user mode, a unit called Process9 will first check the security and then passing code to Kernel 9. The process is shown in Fig.4 [8].

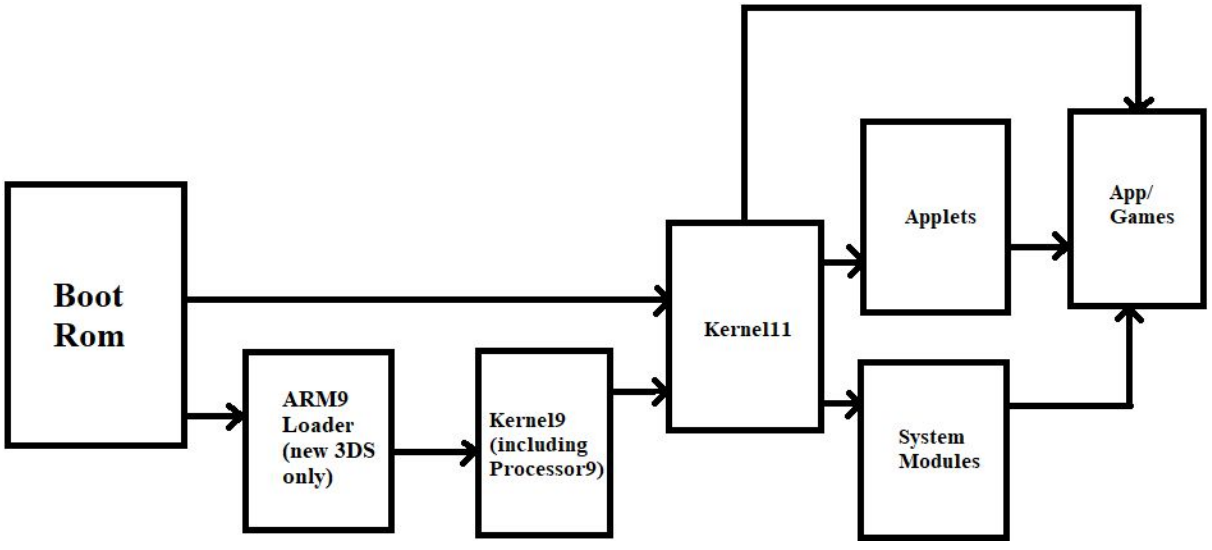

Fig.4 ARM path

b. 3DS, ARM11, and Modules

ARM11 deals with everything a 3DS should do, allowing the players to play the latest games, take 3D pictures, etc. However, the developers paid attention to security in order to prevent their machines from being hacked and stop other people from selling pirate games. One of the best ways is to make the kernel know as little as possible. Therefore, 3DS does not let the kernel, Kernel11, to do most of the jobs. In fact, Kernel11 is only used for some basic tasks, such as memory management,

multi-threading, and I/O access [12]. The developer of 3DS leaves the most important jobs to modules, which are much more secure. In this way, even though the hackers "controls" the kernel, they can hardly map the right address of the memory to patch. Each process in the modules has its own virtual memory and allocates the memory in such a way that people have to "manually" catch and follow the kernel layout. This is complicated and inconvenient, but it is much better than looping through the whole memory [12].

In the paper [12], the author groups the modules into three categories based on their security properties, system modules, aplets, and applications, shown in Fig. 4. System modules serve as backend worker, communicating with the Nintendo server, making sure the connection security with Transport Layer Security (a cryptographic protocol designed to provide communications security over a computer network [13]), and downloading contents. Aplet modules are responsible for displaying images, videos and everything shown on the screen. Application modules run the games, setting apps, camera apps, etc.

c. The New 3DS and ARM9 Loader

The new 3DS is an upgraded model of 3DS, which came out in 2015. In this new model, Nintendo added an additional layer of encryption for the Kernel9/Process9 code and they added ARM9 Loader to derive the new keys and decrypt the new layer [8], which make the system much safer. Basically, ARM9 loader provides a way to check if the machine is hacked. It will compare a "key" stored in the system with the one sent by boot rom. If they do not match, then the system refuses to pass the commands to Kernel9. It is easier for the Nintendo company to protect their products by simply changing the key every time the system updates. In order not to hack the machine again every time it updates, especially for the users who know nothing about programming, the users have to manually cancel the automatic update, which means they are unable to play the latest games.

Conclusion

All in all, There is no doubt that the ARM processor is an appropriate choice for Nintendo 3DS. It provides efficient execution process, reliable battery life and sufficient game performance. It reduces code density and allows load/store multiple registers at the same time, preprocesses the registers before execution skips instructions.

In the 3DS system, with the protection of system modules and loaders, ARM processors can run in a secure environment without worry being hacked. Therefore, even though ARM9 and ARM11 are no longer a cutting edge technology these years. There is no doubt that they were been used in a wide range of devices in the past, so it is still worth to dig into these processors and understand their specialties.

References:

[1] IFixit, “Nintendo 3DS Teardown.” *IFixit*, 16 Oct. 2013. [Online]. https://www.ifixit.com/Teardown/Nintendo+3DS+Teardown/5029#s22696 [Accessed April 12, 2019].

[2] Bierton, David. "Nintendo 3DS GPU Revealed." *IQGamer.* June 01, 2010. [Online] http://imagequalitymatters.blogspot.com/2010/06/nintendo-3ds-gpu-revealed.html [Accessed April 18, 2019].

[3] "Nintendo 3DS ARM Documentation." *About the 3DS CPU(s) | Nintendo 3DS ARM Documentation*. [Online] https://lioncash.github.io/ARMBook/about_the_3ds.html [Accessed April 18, 2019].

[4] Wikipedia contributors. "ARM architecture." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 12 Apr. 2019. Web. 12 Apr. 2019.

[5] J. Goodacre and A. N. Sloss, “Parallelism and the ARM instruction set architecture,” Computer, vol. 38, no. 7, pp. 42–50, Jul. 2005. [Abstract]. Available: Google Scholar, https://scholar.google.com/. [Accessed April 12, 2019].

[6] ARM, “ARM11 MPCore™ Processor Technical Reference Manual,” 2007. [Online]. Available:http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.arm11/index.htm&_ga=2.207180756.691661817.1553217435-1633743312.1553177519 [Accessed April 12, 2019].

[7] Svetlana Gaivoronski, “Shellcodes for ARM: Your Pills Don't Work on Me, x86.”, SlideShare, LinkedIn Corporation, 10 Sep 2014, https://www.slideshare.net/SvetlanaGaivoronski/shellcodes-for-arm-your-pills-dont-work-om. [Accessed April 17, 2019]

[8] Lu, Yifan. "The 3ds cryptosystem." (2016). https://yifan.lu/2016/04/06/the-3ds-cryptosystem/. [Accessed May 3]

[9] Wikipedia contributors. "Nintendo DS." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 29 Apr. 2019. Web. 6 May. 2019.

[10] Wikipedia contributors. "Advanced Encryption Standard." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 1 May. 2019. Web. 6 May. 2019.

[11] Wikipedia contributors. "RSA (cryptosystem)." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 5 May. 2019. Web. 6 May. 2019.

[12] Lu, Yifan. "3DS Code Injection through ‘Loader’." (2016). https://yifan.lu/2016/03/28/3ds-code-injection-through-loader/. [Accessed May 6]

[13] Wikipedia contributors. "Transport Layer Security." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 7 May. 2019. Web. 8 May. 2019.

[14] Wikipedia contributors. "Instruction set architecture." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 19 Apr. 2019. Web. 8 May. 2019.