RUHR
UNIVERSITÄT
BOCHUM

RUB

Name: **Tim Barsch**
Student ID: **108019210718**
Assignment Number: **1**
Date: October 28, 2021
Course: **ARM Processors for Embedded Cryptography**

1.1: Explain the differences between the RISC and the CISC processor architecture. Which architecture does our STM32F407VG μC have?

The difference between *Complex Instruction Set Architecture* and *Reduced Instruction Set Architecture* is self-explanatory when you look at the names. The RISC architecture use a reduced instruction set and perform each instruction in one single clock cycle. On one side each instruction is simple and fast performed but on the other hand it means that the number of instruction for each high level logic is increased what cause larger code sizes. Other properties of the RISC architecture are that instructions comes under size of one word, it has often more general purpose registers and has usually less data types. In difference to RISC, the CISC architecture use a complex instruction set. This means that one single CISC instruction can perform a higher level logic. For example, the CISC instruction can perform a multiplication in one single instruction, but the RISC instruction set does not have a instruction for multiplication. So you have to load to different registers and perform the multiplication 'manually'. But the CISC architecture does all this steps in one single instruction. A CISC code has less instruction and reduce the code size in comparison to RISCs but can not more perform each instruction in one clock cycle. So one CISC instruction are larger than one word, has less general purpose registers, has usually more data types and can perform operations in memory itself.

So in conclusion we can say that a CISC architecture is more focused on hardware and RISC has a focus on software. Through the reduced instruction set of RISC architecture it is harder to design a program on this architecture and needs more software support. The CISC architecture is through the complex instruction set easier to understand.

The STM32F407VG μC is a RISC.

1.2: Explain the differences between the Von-Neumann and the Harvard memory architecture.

The difference between the *Von-Neumann* and the *Harvard* architecture lays in the memory where data and instructions are saved. Instructions and data is saved in the same memory

in *Von-Neumann* architecture. In the *Harvard* architecture, there is a separation between data and instruction memory and buses and is developed to overcome the bottleneck of *Von-Neumann* architecture. This bottleneck is overcome by the separation because the CPU is now able to read/write data and access instructions at the same time.

1.3: Explain the advantages of using a pipeline in your own words.

The benefit of a pipeline is obvious: higher performance. Through dividing the instruction execution into the smaller the steps *fetch, decode* and *execute* it is possible to perform more instruction per given time and the clock cycle speed can be increased. Without pipelining, one instruction can be performed at once, all smaller steps has to perform in one clock cycle and the units for the small steps are often idle while other works. With pipelining it is possible to perform more instruction in the same time because all units for the small steps can perform the next instruction step without being idle.

But the usage of a pipeline does not have only positive effects. Through the beginning performance of the next instruction before the last instruction is executed, the branch of instruction has to be predicted and this cause a lot of problems. The best examples for this issues are the security vulnerabilities *Meltdown* and *Spectre*. These vulnerabilities allows an attacker to get secret data from other programs in execution. In a nutshell, this was possible through CPUs use branch prediction and speculative execution, what cause leakages of sensitive data. For example, if the destination of a branch depends on a memory value that is in the process of being read, CPUs will try to guess the destination and attempt to execute ahead. Speculative logic is unfaithful in how it executes, can access the victim's memory and registers, and can perform operations with measurable side effects. [3]

1.4: How many pipeline stages does our Cortex-M4 processor have?

The Cortex-M4 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. [4]

1.5: The Cortex-A series is targeted at computationally challenging applications. How many pipeline stages does the Cortex-A12 have? Compare this to the amount of pipeline stages in Intel Ice Lake processors.

At a high level, Cortex A12 features a 10 - 12 stage integer pipeline - a lengthening of Cortex A9's 8 - 11 stage pipeline. [5] The Intel Ice Lake client processors have 14 - 19 pipeline stages. [6] That means that a Intel Ice Leak processor has up to 9 stages more than a Cortex A12 processors. This has to result in a higher clock speed of the Intel processors, but that is not surprising, because the Intel Ice Leak platform is a 64-bit, 10nm microarchitecture to be implemented in modern mobile devices like the Cortex A12 as well. The Cortex A12 is a 32-bit, 28nm microarchitecture [7] and just these values let you know that the Intel processors use newer technology in comparison to the Cortex A12.

1.6: To which ARM family does the Cortex-M4 processor belong?

The Cortex M4 is a ARMv7-M architecture. [8]

1.7: 32-bit ARM instruction sets.

In the lecture, you learned the format of 32-bit ARM data process- ing instructions (A32 instruction set), but also saw that there are less powerful 16-bit variants, optimized for the common case. These instructions are implemented in the thumb instruction set. Most smaller ARM processors are actually limited to only support thumb or thumb2.

i) thumb instruction set in comparison to the A32 instruction set

"The Thumb instruction set consists of 16-bit instructions that act as a compact shorthand for a subset of the 32-bit instructions of the standard ARM. Every Thumb instruction could instead be executed via the equivalent 32-bit ARM instruction. However, not all ARM instructions are available in the Thumb subset; for example, there's no way to access status or coprocessor registers. Also, some functions that can be accomplished in a single ARM instruction can only be simulated with a sequence of Thumb instructions." [9] So the most important difference is the operation size of both instruction sets. Be while A32 can access 32-bit registers as well, the thumb instruction set can just operate on 16-registers. A other major difference is that thumb has unique stack mnemonics[1] - PUSH and POP - that do not exist in A32. [9] But what is the advantage of using thumb instead of A32? It is the reduced code density and the higher performance in comparison to A32. [9]

ii) thumb instruction set in comparison to the thumb2 instruction set

Thumb2 introduces a 32-bit instructions that are intermixed with the thumb 16-bit instructions. So the thum2 instruction set is fully compatible with the thumb instruction set. "Thumb-2 has the performance close to or better than that of the ARM instruction set and has the code density of the original Thumb." [10]

iii) Which instruction set(s) does our STM32F407VG μC support?

"The Cortex-M4 processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements." [11]

So the STM32F407VG μC supports the Thumb-2 instruction set and thought the downwards compatibility Thumb as well.

[1] A human readable assembly instruction

References

[1] Himanshu B., Ayush N., Pulkit A. & Anshul T., Last Updated on Feb. 19, 2021, geeksforgeeks.org/computer-organization-risc-and-cisc/

[2] Pankaj P. & Gurukiran S., Last Updated on Aug. 04, 2021, geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/

[3] Paul Kocher et al., September 16, 2019, *Spectre Attacks: Exploiting Speculative Execution* ieeexplore.ieee.org/document/8835233

[4] Cortex-M4 User Guide, P. 11

[5] Anand Lal Shimpi, Published on July 17, 2013, anandtech.com/show/7126/the-arm-diaries-part-2-understanding-the-cortex-a12/2

[6] Undefined Author, Last Updated on June 20, 2021, en.wikichip.org/wiki/intel/microarchitectures/ice_lake_(client)

[7] Undefined Author, Last Updated on Dec. 31, 2018, en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a12

[8] Cortex-M4 User Guide, P. 8

[9] Undefined Author, Sep. 24, 2003, embedded.com/introduction-to-arm-thumb/

[10] ARM1165T2-S Technical Reference Manual, developer.arm.com/documentation/ddi0338/g/ch01s02s01

[11] STM32 Cortex-M4 MCUs and MPUs programming manual, P. 14, st.com/resource/en/programming_manual/dm00046982-stm32-cortex-m4-mcus-and-mpus-programming-manual-stmicroelectronics.pdf