The ARM Instruction Set Architecture

Dean Vidafar (Z5205847)

UNSW SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Author Note

For Any Enquiries Contact

<Z5205847@student.unsw.edu.au>

Table of Contents

[Table of Figures 3](#_Toc523133834)

[Abstract 4](#_Toc523133835)

[ARM7TDMI Core Diagram 5](#_Toc523133836)

[The ARM Instruction Set Architecture 6](#_Toc523133837)

[The Thumb Instruction Set 6](#_Toc523133838)

[Why do we use thumb? 7](#_Toc523133839)

[ARM Instruction Set 8](#_Toc523133840)

[Arithmetic and the ARM7TDMI 9](#_Toc523133841)

[Registers in the ARM7TDMI 10](#_Toc523133842)

# Table of Figures

[Figure 1: The ARM7DMI Core Diagram (ATMEL, 2004) 5](file:///C:\Users\QB1T\Documents\GitHub\2121-Assignment---ARM7TDMI\The%20ARM%20Instruction%20Set%20Architecture.docx#_Toc523133843)

[Figure 2: Example of how the ADD instruction is converted 7](file:///C:\Users\QB1T\Documents\GitHub\2121-Assignment---ARM7TDMI\The%20ARM%20Instruction%20Set%20Architecture.docx#_Toc523133844)

[Figure 3: The conversion of High-Level C, to Low Level ARM. (Suh, 2015) 9](file:///C:\Users\QB1T\Documents\GitHub\2121-Assignment---ARM7TDMI\The%20ARM%20Instruction%20Set%20Architecture.docx#_Toc523133845)

[Figure 4: Program status register format ATMEL (2004) 10](file:///C:\Users\QB1T\Documents\GitHub\2121-Assignment---ARM7TDMI\The%20ARM%20Instruction%20Set%20Architecture.docx#_Toc523133846)

[Figure 5:ARM7TDMI Architecture (Techcon, 2012) 11](file:///C:\Users\QB1T\Documents\GitHub\2121-Assignment---ARM7TDMI\The%20ARM%20Instruction%20Set%20Architecture.docx#_Toc523133847)

Abstract

The ARM processor is a 32-bit Reduced Instruction Set Computer (RISC), a microprocessor that recognises a limited number of instructions. An advantage of RISC microprocessors is that they can execute instructions relatively quick since the instructions are elementary. RISC chips also require fewer transistors and other components, making both RND and manufacturing more cost-efficient. This architecture is significantly different from other CISC (Complex instruction set computer) microprocessors and is primarily designed for embedded systems. In this report, the ARM Instruction Set Architecture, Memory Models, Registers, Instruction Set, and Data Types will be discussed.(Shee, 2004)

# ARM7TDMI Core Diagram

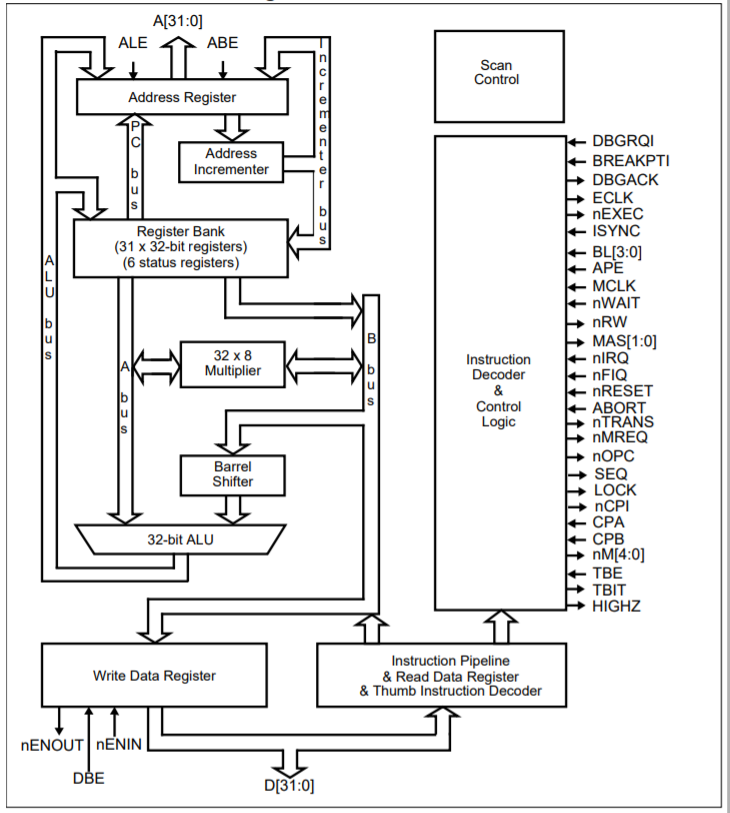


Figure 1: The ARM7DMI Core Diagram (ATMEL, 2004)

The ARM Instruction Set Architecture

The drawback of most RISC processors, including the ARM7TDMI however, is code density. In many cost-critical applications, one of the most expensive system components has always been memory. Therefore, the less memory a program occupies, the more cost effective. Since RISC processors have simple instructions, multiple instructions are needed to perform what may be done in a single CISC instruction. (Segars et al., 1995) The ARM microprocessor is based on the load/store architecture. ARM7TDMI offers no support for unaligned memory accesses. This microprocessor features two instruction sets, with both having mostly single clock-cycle execution.(McDermott, 2008). These instruction sets are:

* A 32-bit ARM instruction set
* A 16-bit Thumb instruction set

# The Thumb Instruction Set

The thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Each thumb instruction is 16 bits long, with a corresponding 32-bit ARM instruction that preforms the same operation. There is excellent interoperability between ARM and Thumb states due to the thumb instructions operating with the standard ARM register configuration. 16-bit thumb instructions are transparently decompressed to full 32-bit instructions on execution without performance loss. (ATMEL, 2004)

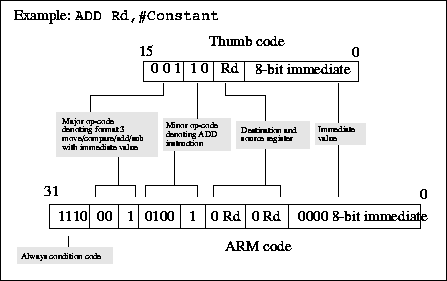


Figure 2: Example of how the ADD instruction is converted

from Thumb to ARM. (Bong-Ho, 1998)

## Why do we use thumb?

Thumb has all the advantages of using a 32-bit core such as:

* 32-bit registers
* 32-bit shifter
* 32-bit Arithmetic Logic Unit (ALU)
* 32-bit address space
* 32-bit memory transfer

Thumb offers a long branch range, powerful arithmetic operations and a large address space. Thumb Code is typically 65% of the size of ARM code and provides 160% of the performance of ARM code when running from a 16-bit memory system. Therefore, the ARM7TDMI core is ideally suited to embedded applications where there is restricted memory bandwidth and code density as well as footprint is integral.

# ARM Instruction Set

The ARM Instruction set offers a fixed instruction width of 32 bit to ease decoding and pipelining, at the cost of decreased code density.

# Arithmetic and the ARM7TDMI

The ARM7TDMI Microprocessor includes integer arithmetic operations for add, subtract and multiply, however, no support is provided for division. The microprocessor supports 32-bit by 32-bit multiplication with either a 32-bit result or 64-bit result. Arithmetic instructions can either set or preserve the conditions bits at the will of the programmer. The ARM7TDMI offers a range of addressing modes such as:

* Direct Addressing: ADD r3, r0, r1
* Immediate Addressing ADD r3, r0, #7
* Direct Addressing with A Shift or Rotate ADD r3, r0, r1, LSL#2

The Microprocessor supports some basic logical operators: AND, OR, XOR and BIC (Diba, 2014)

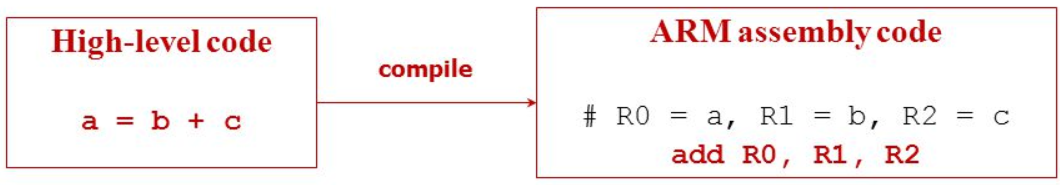


Figure 3: The conversion of High-Level C, to Low Level ARM. (Suh, 2015)

# Registers in the ARM7TDMI

The ARM7TDMI Microprocessor has a total of thirty-seven registers – thirty-one general-purpose 32-bit registers and six status registers - but these cannot all be seen at once. The processor state and operating mode dictate which registers are available to the programmer. In ARM state, 16 general registers and one or two status registers are visible at any one time. The ARM state register set contains 16 directly accessible registers: R0 to R15. All of these except R15 are general-purpose and may be used to hold either data or address values. In addition to these, there is a seventeenth register used to store status information. (ATMEL, 2004) For thumb code however conditional execution is not supported since the condition code would not have left enough space for a useful set of instructions. Another compromise made was how the register bank is access. ARM code has free access to 16 registers at once and uses three or four operand instructions with 4-bit register specifiers. Again, to allow enough opcode space, there is free access to only eight registers. Instructions consist of the more conventional two or three operands and use 3- bit register specifiers. (Segars et al., 1995)

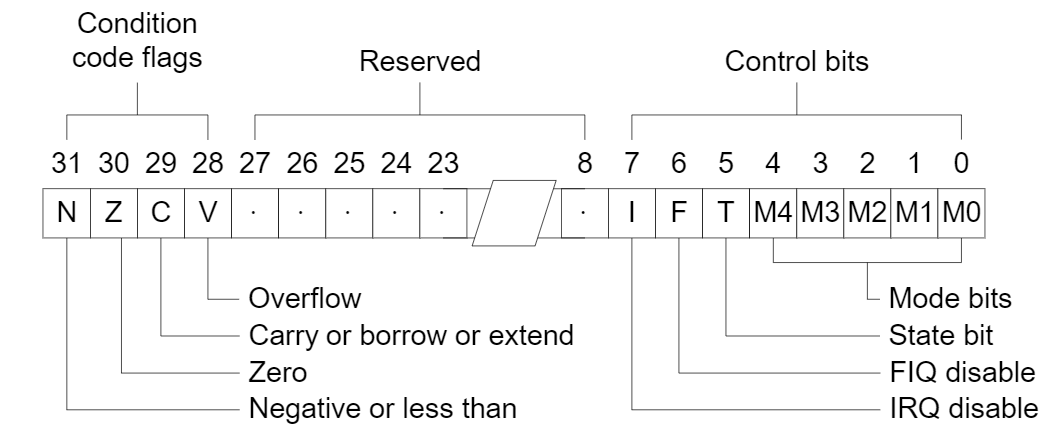


Figure 4: Program status register format ATMEL (2004)

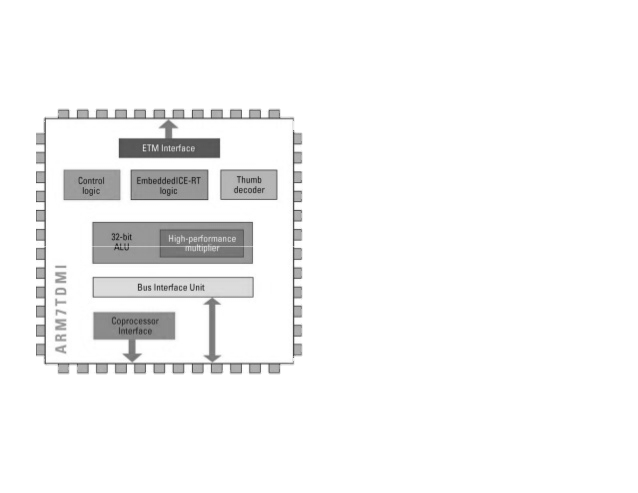


Figure 5:ARM7TDMI Architecture (Techcon, 2012)

BIBLIOGRAPHY

ATMEL. 2004. ARM7TDMI Technical Reference Manual. *Infocentre ARM* [Online]. Available: <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0210c/CACBCAAE.html>.

BONG-HO, K. 1998. Introduction to Thumb. *Characteristics of ARM Processor***,** 2-10.

DIBA, M. 2014. ARM Arithmetic and Bitweise Instructions. *Computer Organization and Systems Programming***,** 5-20.

MCDERMOTT, M. 2008. The ARM Instruction Set Architecture. *The University of Texas at Austin***,** 2-2.

SEGARS, S., CLARKE, K. & GOUDGE, L. J. I. M. 1995. Embedded control problems, Thumb, and the ARM7TDMI. 22-30.

SHEE, S. L. 2004. The ARM Architecture. Sydney: UNSW.

SUH, P. T. 2015. Arm Instructions I. *Computer Science Engineering, Korea University***,** 5.

TECHCON. Architecture. 2012 Sydney. Techcon.