

ARM, THUMB, M68000 Architecture Comparative Analysis

**Steven Battle
CSEE 4824
10/17/2003**

1. Processor Organization

1.1 ARM

The ARM7TDMI architecture is based on modern RISC principles. This processor is a load/store machine featuring 37 registers: 31 general purpose 32-bit registers and six status registers. Not all of these registers are visible at all times; the available register set depends on the state and operating mode of the processor.

In the ARM state, the available register set contains 16 general purpose registers, R0 to R15. These general purpose registers may be used to hold 32-bit data or addresses. Additionally, a status register, R16, the current program state register (CPSR), contains condition code flags and mode bits describing the state of the processor. Specific to the ARM architecture, the Program Counter (PC) is stored in Register 15 with zeros in bits [1:0] indicating that instructions are word-aligned and are 4 bytes in size.

1.2 THUMB

Like the ARM architecture, the THUMB processor is an advanced RISC load/store machine. The THUMB shares many properties with the ARM as it operates as a subset of the ARM architecture. These two processors are fundamentally the same; they run on the same silicon chip and operate in much the same way, indeed, one can switch between the two modes in the same program. What makes the THUMB different from the ARM is the register set, register size, and instruction size.

The THUMB register set is a subset of the ARM register set mentioned above. Instead of 16 GPRs, only eight general purpose registers, R0-R7, are available. The register set also differs in size, 16-bits for THUMB vs. 32-bits for ARM. In addition to these GPRs, as in the ARM state, are the PC, SP, SPSR, and Link registers. The THUMB registers are directly mapped from the ARM state registers, facilitating a convenient transfer of data when switching between the two modes. It should also be noted that the THUMB program counter is different from ARM to accommodate its smaller instruction size. THUMB instructions are half-word aligned, with bit [0] of the PC set to 0 compared to the word aligned instructions of the ARM processor.

1.3 Motorola M68000

The Motorola M68K processor is a classic CISC processor. It has a register set containing 32-bit registers, eight of which are data registers, D0-D7, and another eight address registers, A0-A7. Unlike the more recent ARM and THUMB processors, the Motorola M68K only has an 8-bit condition code register. Whereas the ARM and THUMB status registers contained flags to indicate the state of the program and the processor and various link and process registers, the Motorola processor contains status bits relative to the previously executed instruction, indicating whether the result was negative, zero, overflow, carry, etc. Like the ARM, the M68K has a 32-bit PC register and like the THUMB, it has a stack pointer register (though it is 32-bits compared to THUMB's 16-bit register). Additionally, the M68K has a built in Floating Point Unit (FPU) with a separate 32-bit register set including eight floating point registers, FP0-FP7, and its own 32-bit control and status registers. This more complicated hardware was left out of the ARM and THUMB architectures.

2. Instruction Set Architecture

2.1 Instructions:

The ARM architecture supports a set of approximately 34 base instructions¹ determined by the designers to be the most frequently used, yet robust enough to implement fully featured systems. These instructions range from simple load (LDR) and store (STR) instructions to the most complex multiply (MUL) and multiply accumulate (MLA) (both signed and unsigned). Included are arithmetic, logical, conditional, and data manipulation instructions. Unique to the ARM architecture is the conditional execution of every

¹ ARM7TDMI DATA SHEET, ARM, 1995. Pg 4-1 ARM Instruction set table

instruction. Adding a two character suffix to the instruction tells the processor to execute the instruction only if the condition indicated by the suffix is true. In the absence of a suffix, the processor assumes AL or “always” execute. These suffixes represent the normal conditions usually associated with branch instructions. The format of these instructions is discussed in section 2.6.

The THUMB instruction set, like the register set, is taken directly from the ARM architecture. THUMB implements a condensed subset of the ARM instructions and reduces them in size from 32-bits to 16-bits. By doing so, the number and strength of instructions are similarly reduced. The THUMB ISA consists of a base of 19 Opcode formats² representing instructions ranging from data processing LDR, STR to the most complex arithmetic, MUL. The instructions only work on the limited register set of the THUMB and are no longer conditionally executed by default. Additionally, every instruction now affects the processor status register, whereas in the ARM architecture, the S bit in the instruction must be set in order for this to occur. Also, nearly every operation in THUMB is executed unconditionally, while the reverse is true for ARM instructions.

As a CISC machine, the Motorola M68000 has a more complex ISA with 56 instructions designed around functionality rather than frequency of use. The instruction set was designed to allow almost every instruction to support as many addressing modes as possible while performing logical, data, arithmetic, and conditional operations. To keep the number of instructions manageable, only one mnemonic is needed to operate on byte, word, and long word data sizes. This is similar to ARM and THUMB where instructions are appended to indicate the data size.

2.2 Addressing Modes

The Motorola M68K supports many more addressing modes than the ARM and THUMB architectures. These 14 supported modes fall into six basic categories including: register direct addressing, absolute data addressing, program counter relative addressing, register indirect addressing, immediate data addressing, and implied addressing³. Motorola decided to implement as many modes as possible, allowing instructions to access data in a variety of ways. ARM took the opposite approach, simplifying the way data is accessed by limiting the addressing modes to immediate offset, register offset, and scaled register offset. Both ARM and Motorola saw the need to include post and pre-increment offset options; however this mode is not supported in THUMBs limited architecture. The ARM and THUMB architectures contain some useful addressing modes not present in the M68K. One of these is the LDMIA/STMIA mode, in which multiple registers can be loaded from memory and stored to memory at a time. Another is due to the fact that the ARM can perform load and shift operations in one cycle. Limited support for this exists in THUMB as an ARM MOVS R1, R2, R3 LSL#IMM is translated to LSL R1, R2, #IMM. As the THUMB is a compressed version of ARM, it supports fewer addressing modes than ARM as is evident from the instruction formats of each where ARM instructions include 2 or 3 registers, and THUMB instructions include at most 2 registers.

2.3 Data Types

Each processor supports a limited amount of data types. The both ARM and THUMB supports byte (8-bit), half-word (16-bit) and word (32-bit) data types. The M68K supports data of 8, 16, and 32-bits like the ARM and THUMB, but it also supports bit and BCD data. This architecture also defines its data types differently; as a word is 16-bits and a long-word is 32-bits in length.

2.4 Endianism and Alignment

The representation of data in memory differs between the three processors. Both the ARM and THUMB architectures supports big and little endian data storage. However, the byte alignment differs between these two processors due to the different data bus sizes. In ARM, 32-bit words are aligned by every four bytes and half-words on every even byte. In Thumb, data is aligned by half-words on every even byte due to its

² *ibid* Pg 5-2 ARM Instruction set table

³ M68000, 5th ed. Prentice-Hall, NJ,1986. pg 1-4 Table 1-1

smaller data bus. While in the Motorola architecture, bytes are individually addressable with the high order byte aligned on the even bytes in big endian form. Multibyte data are accessed only on word (even byte) boundaries.

2.5 Conditional Instructions

Conditional instructions are where the ARM architecture really shows its form. A four bit field in the Instruction Format (sec. 2.6) defines the conditions under which the instruction can be executed. If the condition is not met, the instruction is not executed. This can result in very efficient code and allow for greater flexibility when dealing with exceptions and conditional situations where many separate ‘compare and branch’ instructions would be necessary to leapfrog over short code segments.

In all three architectures, conditional instructions and branches are evaluated against the condition code or status register. This register stores flags asserted from the result of instructions. In the THUMB architecture, nearly every instruction sets the appropriate condition flags. In ARM however, the S bit must be explicitly asserted as in the ‘ADDS’ or else the register will not be written to. In the M68K architecture conditions are set with logical and arithmetic operations in addition to compare instructions. In all three architectures, branches operate against these flags allowing the program to jump to target points in the program. Semantically, the conditions vary slightly from the M68K to the ARM and Thumb architectures, but the fundamental operations are the same.

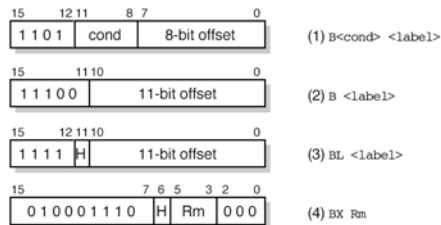
2.6 Instruction Format

Both ARM and THUMB architectures were designed with a uniform instruction format in mind. This greatly reduces the complexity of the instruction decoding hardware at the cost of increased code size. The Motorola M68K uses variable instruction sizes, optimizing for code density.

The ARM can be characterized as a (0,3) machine, with no direct memory access and three operand instructions. These instructions are required to be aligned in memory. The format of these instructions is as follows:

- 31:28 – condition of execution
- 27:26 – Pseudo Opcode
- 25 – Immediate bit – I = 0, src in register, I = 1, src in immediate
- 24:22 – Opcode (together with 27:26 determine operation)
- 21 – S bit, set condition code
- 20:16 – Rn – Source1 register
- 15:12 – Rd – Destination register
- 11:0 – “operand 2” dependant on I bit.

The THUMB instruction format is based on this format, reduced to 16-bits. This reduction is achieved by removing several fields which are deemed not important enough or not necessary for use in the architecture. By reducing the instruction size, the architects have made the format much less regular than the standard ARM processor. By observing the branch instructions, one can see how irregular the THUMB format can be:



Most THUMB instructions have two registers as operands, compared to three with ARM. The format follows this general pattern:

(16-bit instructions vs. 32-bit), resulting in much fewer instruction bytes required for THUMB in exercise 1 than ARM.

Comparing the two RISC architectures, with 32-bit memory, ARM will be faster than THUMB, since it can operate in its native 32-bit mode. With 16-bit memory, Thumb code will be faster, since that is its native mode, though it will require more instructions to perform the same operations. In general, for best performance one would use 32-bit memory and ARM code, though for best cost and power-efficiency, THUMB code paired with 16-bit memory would be optimum. THUMB could also be used in non-speed critical situations, since it will not execute as fast on typical systems as the ARM.

The M68K produces very compact code, at 16-bits per instruction and with a complex instruction set often reducing instruction count. It offers a powerful instruction set with minimal memory use for instruction storage. However much of this advantage has been minimized with the THUMB extension to the ARM architecture. The ISA advantages also depend on the complexity of instructions required in the embedded system. For instructions typically designated as complex, or addressing modes that are complex, the M68K could provide a better solution than THUMB, though ARM emulates complex addressing modes which could eliminate this advantage of the M68K. Exercise 2 shows an example of the tradeoffs between architectures. ARM can perform the function in 4 instructions, while both THUMB and M68K require 8. All three processors require the same number of instruction bytes, it just depends on the size of the operands. For 32-bit operands, ARM would be much faster, for 16-bit, it may be better to use either THUMB or the M68K. In general, the ARM paired with THUMB can create much more robust systems, with the ARM operating on 32-bit memory, and THUMB operating in its own 16-bit memory space, but it depends on the type of system being designed, the memory space, and the operations required.