ARM7TDMI-S processor

The ARM7TDMI-S processor is a member of the ARM family of general-purpose 32-bit microprocessors. The ARM family offers high performance for very low-power consumption and gate count.

The ARM architecture is based on *Reduced Instruction Set Computer* (RISC) principles. The RISC instruction set, and related decode mechanism are much simpler than those of *Complex Instruction Set Computer* (CISC) designs. This simplicity gives:

• a high instruction throughput

• an excellent real-time interrupt response

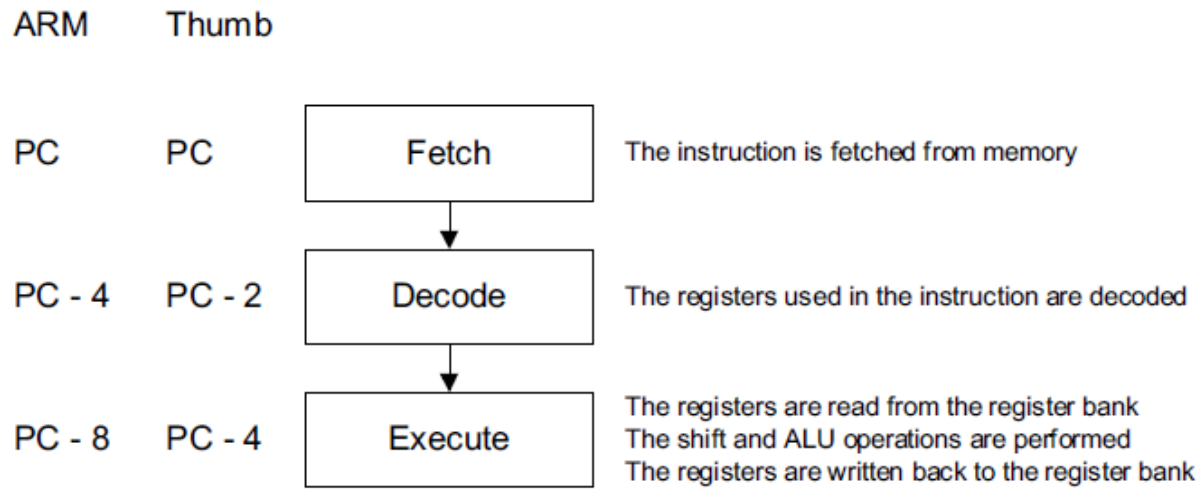
• a small, cost-effective, processor macrocell.

The instruction pipeline

The ARM7TDMI-S processor uses a pipeline to increase the speed of the flow of instructions to the processor. This enables several operations to take place simultaneously, and the processing, and memory systems to operate continuously.

A three-stage pipeline is used, so instructions are executed in three stages:
• Fetch
• Decode
• Execute.

The three-stage pipeline is shown in Figure below.



Note: The *Program Counter* (PC) points to the instruction being fetched rather than to the instruction being executed.

During normal operation, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

Memory access

The ARM7TDMI-S processor has a Von Neumann architecture, with a single 32-bit data bus carrying both instructions and data. Only load, store, and swap instructions can access data from memory.
\
Data can be 8-bit bytes, 16-bit halfwords, or 32-bit words. Words must be aligned to 4-byte boundaries. Halfwords must be aligned to 2-byte boundaries.

ARM7TDMI-S architecture

The ARM7TDMI-S processor has two instruction sets:
• the 32-bit ARM instruction set
• the 16-bit Thumb instruction set.

The ARM7TDMI-S processor is an implementation of the ARM architecture v4T.

Instruction compression

Microprocessor architectures traditionally had the same width for instructions and data.

Therefore, 32-bit architectures had higher performance manipulating 32-bit data and could address a large address space much more efficiently than 16-bit architectures.

16-bit architectures typically had higher code density than 32-bit architectures, and greater than half the performance.

Thumb implements a 16-bit instruction set on a 32-bit architecture to provide:
• higher performance than a 16-bit architecture
• higher code density than a 32-bit architecture.

The Thumb instruction set

The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions.

Thumb instructions are each 16 bits long, and have a corresponding 32-bit ARM instruction that has the same effect on the processor model.

Thumb instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and Thumb states.

On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bit ARM instructions in real time, without performance loss.

Thumb has all the advantages of a 32-bit core:
• 32-bit address space
• 32-bit registers
• 32-bit shifter and *Arithmetic Logic Unit* (ALU)
• 32-bit memory transfer.

Thumb therefore offers a long branch range, powerful arithmetic operations, and a large address space.

ARM7TDMI-S block diagram

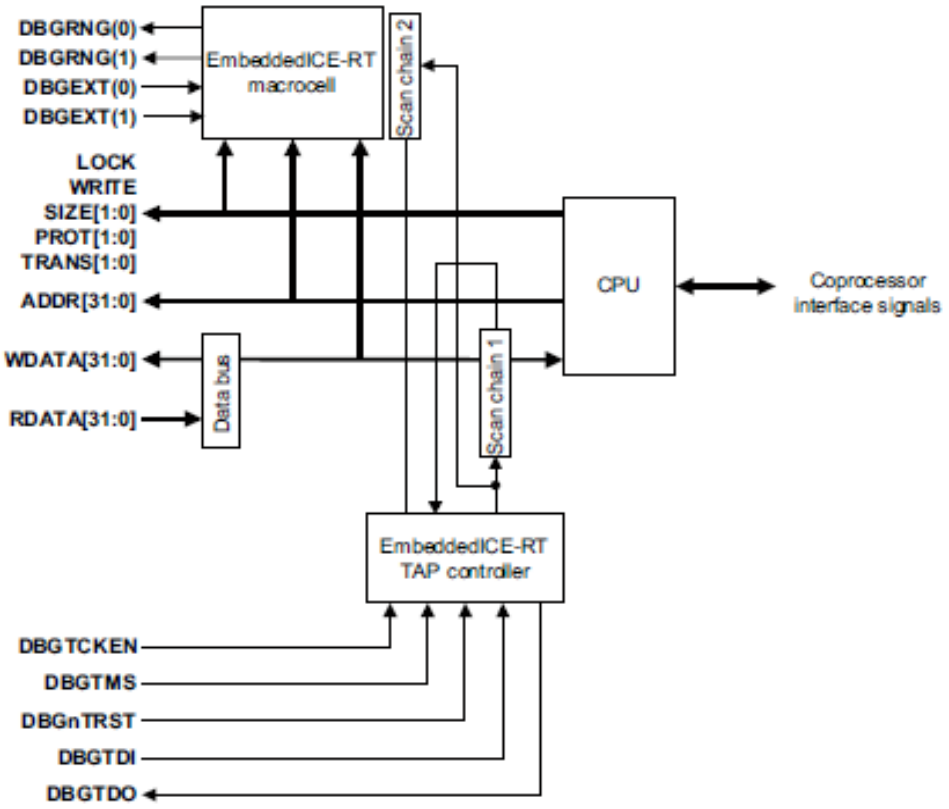


Figure ARM7TDMI-S block diagram

About the memory interface

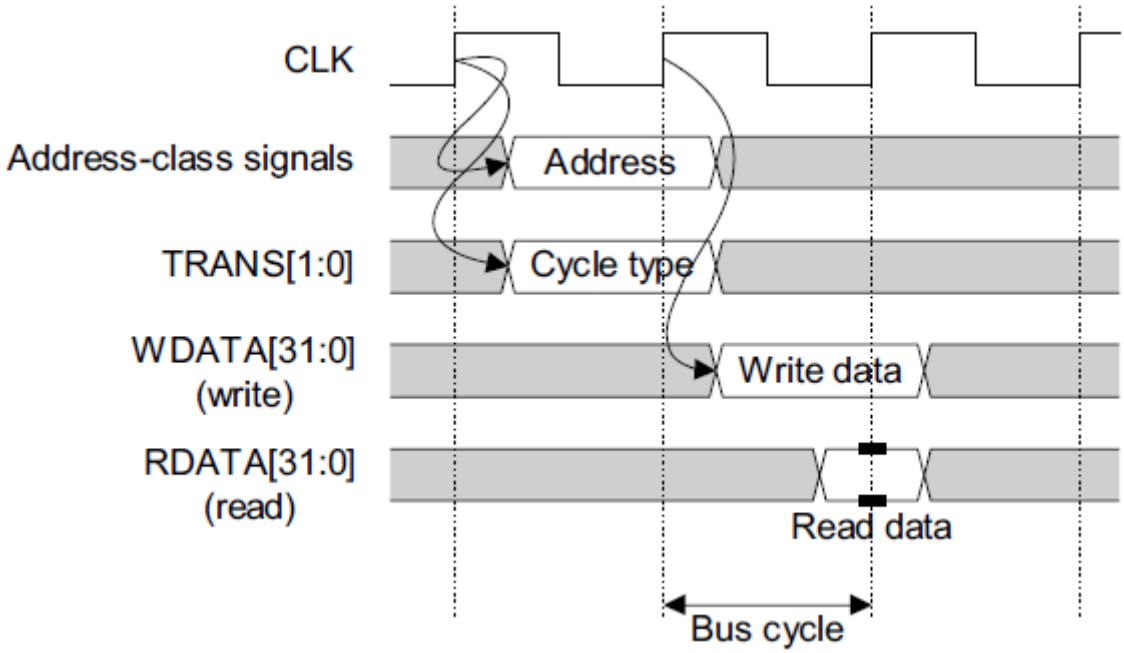
The ARM7TDMI-S processor has a Von Neumann architecture, with a single 32-bit data bus carrying both instructions and data. Only load, store, and swap instructions can access data from memory.

The ARM7TDMI-S processor supports four basic types of memory cycle:
• nonsequential
• sequential
• internal
• coprocessor register transfer.

Bus cycle types

The ARM7TDMI-S processor bus interface is pipelined, and so the address class signals, and the memory request signals are broadcast in the bus cycle ahead of the bus cycle to which they refer. This gives the maximum time for a memory cycle to decode the address, and respond to the access request.

A single memory cycle is shown in Figure below.



The ARM7TDMI-S processor bus interface can perform four different types of memory cycle. These are indicated by the state of the **TRANS[1:0]** signals. Memory cycle types are encoded on the **TRANS[1:0]** signals as shown in Table below.

TRANS[1:0]	Cycle type	Description
00	I cycle	Internal cycle
01	C cycle	Coprocessor register transfer cycle
10	N cycle	Nonsequential cycle
11	S cycle	Sequential cycle

The ARM7TDMI-S processor has four basic types of memory cycle:

Nonsequential cycle

During this cycle, the ARM7TDMI-S core requests a transfer to, or from an address which is unrelated to the address used in the preceding cycle.

Sequential cycle

During this cycle, the ARM7TDMI-S core requests a transfer to or from an address that is either one word or one half word greater than the address used in the preceding cycle.

Internal cycle

During this cycle, the ARM7TDMI-S core does not require a transfer because it is performing an internal function and no useful prefetching can be performed at the same time.

Coprocessor register transfer cycle

During this cycle, the ARM7TDMI-S core uses the data bus to communicate with a coprocessor but does not require any action by the memory system.

Ref: ARM7TDMI-S Revision: r4p3 Technical Reference Manual