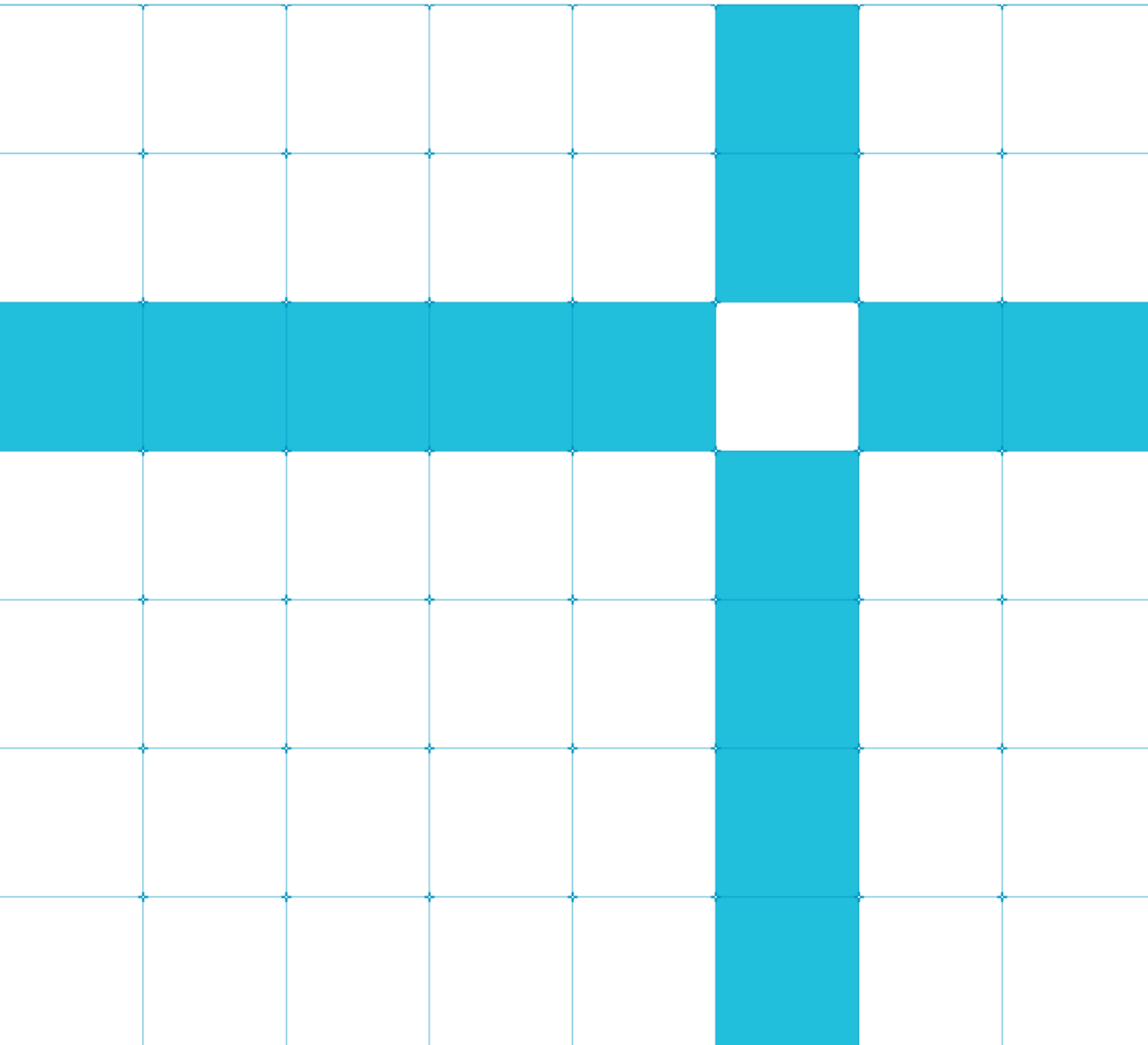
arm

Introducing the Arm architecture

Version 1.0



Introducing the Arm architecture

Release Information

Document History

Version	Date	Confidentiality	Change
1.0	1 April 2019	Non-confidential	First release

Non-Confidential Proprietary Notice

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

1 Overview

The Arm architecture provides the foundations for the design of a processor or core, things we refer to as a Processing Element (PE).

The Arm architecture is used in a range of technologies, integrated into System-on-Chip (SoC) devices such as smartphones, microcomputers, embedded devices, and even servers.

The architecture exposes a common instruction set and workflow for software developers, also referred to as the Programmer's model. This helps to ensure interoperability across different implementations of the architecture, so that software can run on different Arm devices.

This guide introduces the Arm architecture for anyone with an interest in it. No prior knowledge of the Arm architecture is needed, but a general familiarity with processors and programming and their terminologies is assumed.

At the end of this guide you can **check your knowledge**. You will have learned about the different profiles of the Arm architecture and whether certain features are architecture or micro-architecture specific. document contains a guide to the Cortex-A55 micro-architecture with a view to aiding software optimization.

2 About the Arm architecture

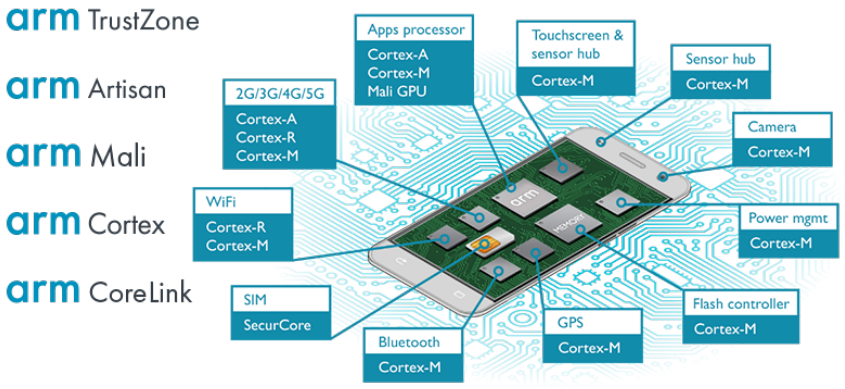
The Arm architecture is one of the most popular processor architectures in the world today, with several billion Arm-based devices shipped every year.

There are three architecture profiles: A, R and M.

A-profile (Applications)	R-profile (Real-time)	M-profile (Microcontroller)
• High performance	• Targeted at systems with real-time requirements.	• Smallest/lowest power. Small, highly power-efficient devices.
• Designed to run a complex operating system, such as Linux or Windows.	• Commonly found in networking equipment, and embedded control systems.	• Found at the heart of many IoT devices.

These three profiles allow Arm architecture to be tailored to the needs of different use cases, while still sharing several base features.

Note: Arm Cortex is the brand name used for Arm's processor IP offerings. Our partners offer other processor brands using the Arm architecture.



3 What do we mean by architecture?

When we use the term architecture, we mean a functional specification. In the case of the Arm architecture, we mean a functional specification for a processor. An architecture specifies how a processor will behave, such as what instructions it has and what the instructions do.

You can think of an architecture as a contract between the hardware and the software. The architecture describes what functionality the software can rely on the hardware to provide. Some features are optional, as we will discuss later in the section on micro-architecture.

The architecture specifies:

Instruction set	• The function of each instruction • How that instruction is represented in memory (its encoding).
Register set	• How many registers there are. • The size of the registers. • The function of the registers. • Their initial state.
Exception model	• The different levels of privilege. • The types of exceptions. • What happens on taking or returning from an exception.
Memory model	• How memory accesses are ordered. • How the caches behave, when and how software must perform explicit maintenance.
Debug, trace, and profiling	• How breakpoints are set and triggered. • What information can be captured by trace tools and in what format.

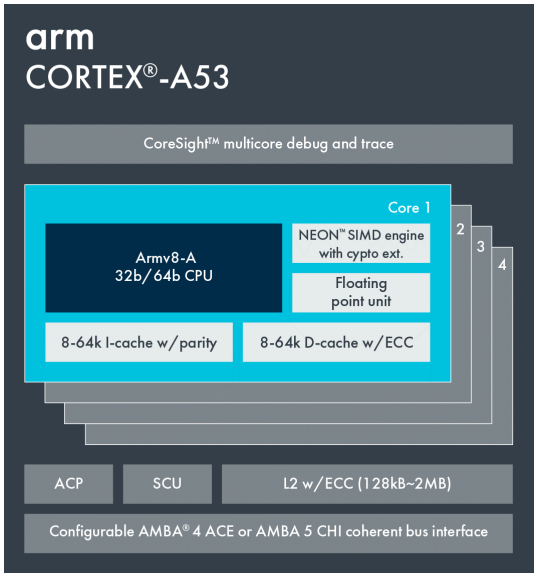
4 Architecture and micro-architecture

Architecture does not tell you how a processor is built and works. The build and design of a processor is referred to as micro-architecture. Micro-architecture tells you how a processor works.

Micro-architecture includes things like:

- Pipeline length and layout.
- Number and sizes of caches.
- Cycle counts for individual instructions.
- Which optional features are implemented.

For example, Cortex-A53 and Cortex-A72 are both implementations of the Armv8-A architecture. This means that they have the same architecture, but they have very different micro-architectures, as shown in the following image:



Target	Optimized for power efficiency	Optimized for performance
Pipeline	8 stages In-order	15+ stages Out-of-order
Caches	L1 I cache: 8KB - 64KB L1 D cache: 8KB - 64KB L2 cache: optional, up to 2MB	L1 I cache: 48KB fixed L1 D cache: 48KB fixed L2 cache: mandatory, up to 2MB

Software that is architecturally-compliant can run on either the Cortex-A53 or Cortex-A72 without modification, because they both implement the same architecture.

5 Development of the Arm architecture

The Arm architecture is developed over time and each version builds on what came before.

You will commonly see the architecture referred to as something like:

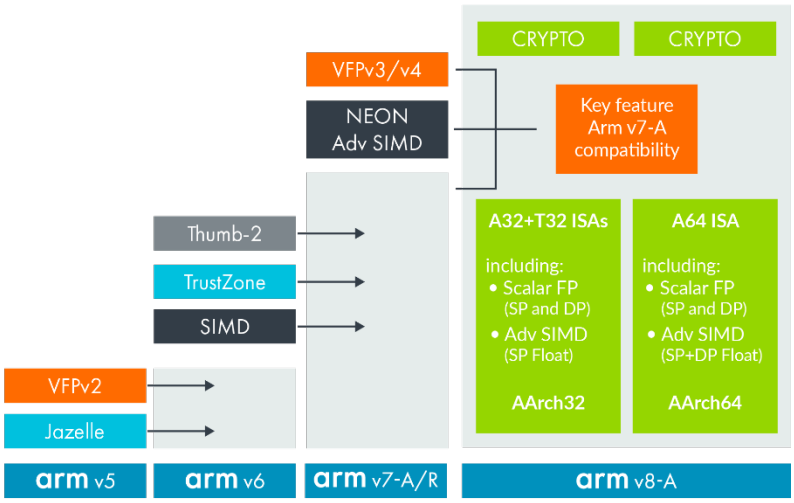
```
Armv8-A
```

This means Version 8 of the architecture, for A-Profile.

Or, in short form:

```
v8-A
```

This figure shows the development of the Arm architecture from version 5 to version 8, with the new features that were added each time. In this guide, we will only look at Armv8-A.

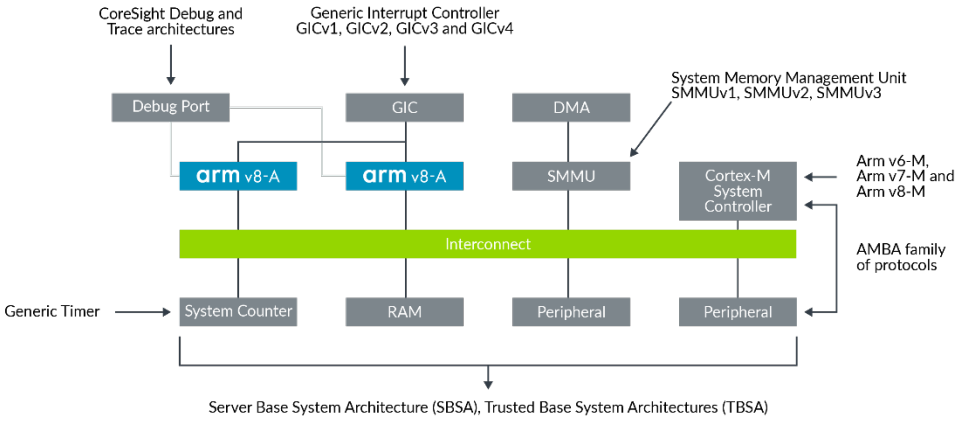


Armv8-A was a major milestone for Arm. Up to and including Armv7-A/R, the Arm architecture was a 32-bit architecture. Armv8-A is a 64-bit architecture, although it still supports 32-bit execution to provide backwards compatibility for legacy software (for example, v7, v6, and v5).

We will not discuss all the features listed on the diagram here, but we will introduce them in later topics.

6 Other Arm architectures

The Arm architecture is the best-known Arm specification, but it is not the only one. Arm has similar specifications for many of the components that make up a modern System-on-Chip (SoC). This diagram provides some examples:



Generic Interrupt Controller

The Generic Interrupt Controller (GIC) specification is a standardized interrupt controller for use with Armv7-A/R and Armv8-A/R.

System Memory Management Unit

A System Memory Management Unit (SMMU or sometimes IOMMU) provides translation services to non-processor masters.

Generic Timer

The Generic Timer provides a common reference system count to all the processors in the system. These provide timer functionality, which is used for things like the operating system's scheduler tick. The Generic Timer is part of the Arm architecture, but the system counter is a system component.

Server Base System Architecture and Trusted Base System Architecture

The Server Base System Architecture (SBSA) and Trusted Base System Architecture (TBSA) provide system design guidelines for SoC developers.

Advanced Microcontroller Bus Architecture

The Advanced Microcontroller Bus Architecture (AMBA) family of bus protocols control how components in an Arm-based system are connected, and the protocols on those connections.

7 Understanding Arm documentation

Arm provides a lot of documentation to developers. We will explain where to find documentation and other information for developing on Arm.

7.1. Where is the documentation?

The **Arm developer website** - This is where you can download the Arm architecture and processor manuals.

The **Arm community** is where you can ask development questions, and find articles and blogs on specific topics from Arm experts.

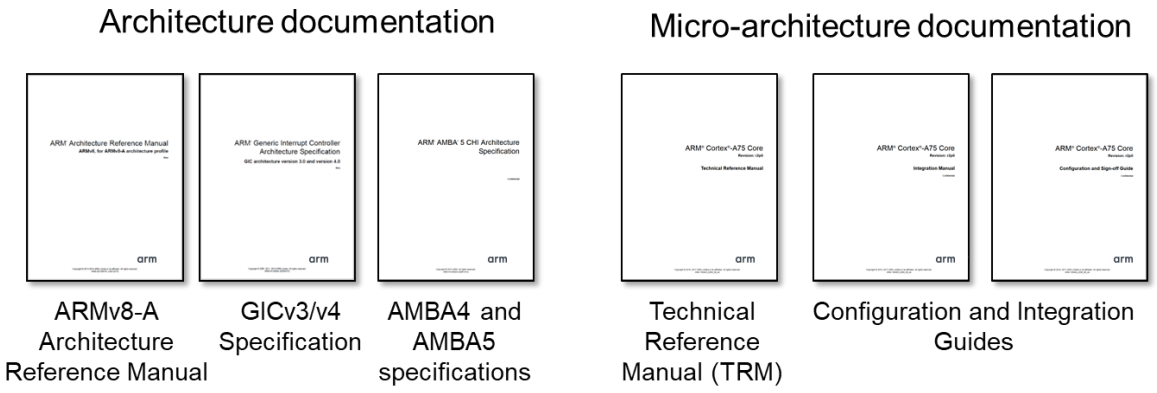
7.2. Which document describes what?

- Each Arm Architecture Reference Manual (Arm ARM) describes an architecture specifications. An Arm ARM is relevant to any implementation of that architecture.
- Each Arm Cortex processor has a Technical Reference Manual (TRM). The TRM describes the features specific to that processor. In general, the TRMs will not repeat any information given in the Arm ARMs.
- Each Arm Cortex processor also has a Configuration or Integration Manual (CIM). The CIM describes how to integrate the processor into a system. Generally, this information is only relevant to SoC designers.

Note: The CIMs are only available to IP licensees. The TRMs are available to download from developer.arm.com without a license.

7.3. So, what does this mean for me?

If you are looking for information on a particular processor, you might need to refer to several different documents. Here we can see the different documents you might need to use with a Cortex-A75 processor.



Cortex-A75 implements ARMv8.2-A, a GICv4 CPU interface and AMBA bus interfaces, so you would need to refer to separate documents for each element. Plus, you would need to refer to the documents detailing the micro-architecture.

If you are working with an existing SoC, you will also use documentation from the SoC's manufacturer. This documentation is typically referred to as a datasheet. The datasheet gives information specific to that SoC.

7.4. What information will I find in each document?

	Architecture			Microarchitecture		SoC datasheet
	Arm ARM	**GIC specifications**	**AMBA specifications**	**TRM**	**CIM**	
Instruction set	X					
Instruction cycle timings				X		
Architectural registers	X	X				
Processor specific registers				X		
Memory model	X					
Exception model	X					
Support for optional features				X	X (some might be synthesis choice)	
Size of caches/TLBs				X		
Power management				X		
Bus ports				X	X	
All legal bus transactions			X			
Bus transactions generated by processor				X		
Memory map						X
Peripherals						X
Pin-out of SoC						X

7.5. Differences between reference manuals and user guides

The documents we have looked at so far, Arm ARMs, TRMs and CIMs, are reference manuals. This means that they do not provide guidance on how to use the processor. For example, the Arm ARM does not have a section on how to turn on an MMU.

This structure is deliberate, and is intended to keep a clear divide between the technical detail of what the architecture requires, which is found in reference manuals, and documents that provide more general guidance, such as this guide. Some general guidance documents will introduce concepts, and others provide instructions for you to follow.

8 Common architecture terms

The architecture uses a number of terms, usually written in small capital letters in documentation, which have very specific meanings. While the Arm Architecture Reference Manuals (Arm ARMs) provide a full definition of each term, here we will look at the most common terms and what they mean to programmers.

PE Processing Element

Processing Element (PE) is a generic term for an implementation of the Arm architecture. You can think of a PE as anything that has its own program counter and can execute a program. For example, the Arm ARM states:

> *The states that determine how a PE operates, including the current Exception level and security state, and in AArch32 state, the PE mode.*

Manuals use the generic term PE because there are many different potential micro-architectures. For example, the following micro-architectures are possible in the Arm Cortex-A processors:

- Cortex-A8 is a single core, single-thread processor. The entire processor is a PE.
- Cortex-A53 is a multi-core processor, each core is a single thread. Each core is a PE.
- Cortex-A65AE is a multi-core processor, each core has two threads. Each thread is a PE.

By using the term PE, the architecture is kept separate from the specific design decisions that are made in different processors.

IMPLEMENTATION DEFINED

A feature which is IMPLEMENTATION DEFINED (IMP DEF for short) is defined by the specific micro-architecture. The implementation must present a consistent behavior/value.

For example, the size of the caches is IMP DEF. The architecture provides a defined mechanism for software to query what the cache sizes are, but the size of the cache is up to the processor designer.

Similarly, support for the cryptography instructions is IMP DEF. Again, there are registers to allow software to determine if the instructions are present or not.

In both examples, the choice is static. That is, a given processor either will, or will not, support the features and instructions. The presence of the feature cannot change at runtime.

For Cortex-A processors, some IMP DEF choices will be fixed, and some will be synthesis options. For example, on Cortex-A57 the size of the L1 caches is fixed, and the size of the L2 cache is a synthesis option. However, the decision about the size of the L2 cache is made at design time. It is still static at runtime.

Full details of the IMP DEF options will be documented in the TRM.

UNPREDICTABLE and CONSTRAINED UNPREDICTABLE

UNPREDICTABLE and CONSTRAINED UNPREDICTABLE are used to describe things that software should not do.

When something is UNPREDICTABLE or CONSTRAINED UNPREDICTABLE, the software cannot rely on the behavior of the processor. The processor might also exhibit different behaviors if software carried out the bad action multiple times.

For example, providing a misaligned translation table is CONSTRAINED UNPREDICTABLE. This represents bad software. Bad software is software that violates the architectural rule that translation tables should adhere to.

Unlike IMP DEF behaviors, the TRM does not usually describe all the UNPREDICTABLE behaviors.

DEPRECATED

Sometimes, we will remove a feature from the architecture. There are several reasons that might happen, such as performance or because the feature is no longer commonly used and is unnecessary. However, there may still be some legacy software that relies upon the feature. Therefore, before removing a feature completely, we will first mark it as DEPRECATED. For example, the Arm ARM states:

The uses of the IT instruction, and use of the CP15DMB, CP15DSB and CP151SB barrier instructions, are deprecated for performance reasons.

DEPRECATED is a warning to developers that a feature will be removed in the future, and that they should start removing it from their code.

Often, a control will be added to the architecture at the same time, allowing the feature to be disabled. This control allows developers to test for use of the feature in legacy code.

RES0/RES1 Reserved, should be Zero/Reserved, should be One

Reserved, should be Zero/Reserved, should be One (RES0/RES1) is used to describe a field that is unused and has no functional effect on the processor.

A reserved field might be used in some future version of the architecture. In this instance, the RES0/RES1 field value of 1 will give the new behavior.

A RES0 field will not always read as 0, and a RES1 field might not always read as 1. RES0/1 only tells you that the field is unused.

There are times when RES0/RES1 fields must be stateful. Stateful means that the fields read back the last written value.

9 Check your knowledge

Q: If you saw Armv7-R referred to in a document, which version and profile of the architecture is being referred to?

A: Version 7, R-Profile

Q: In which version of the Arm architecture was 64-bit support added to the A-Profile?

A: Version 8 (Armv8-A)

Q: For each of the following, would you classify them as architecture or micro-architecture: instruction encodings, cache size, and memory ordering?

A:

- Instruction encodings: **Architecture**
- Cache size: **Micro-architecture**
- Memory ordering: **Architecture**

Q: What is a PE?

A: A PE is a Processing Element: a machine that implements the Arm architecture.

10 Related information

Here are some resources related to material in this guide:

- **Arm architecture and reference manuals**
- **Arm Community** - Ask development questions, and find articles and blogs on specific topics from Arm experts.

Here are some resources related to topics in this guide:

Other Arm architectures

- **Generic Interrupt Controller (GIC)**
- **Server Base System Architecture (SBSA)**
- **System Memory Management Unit (SMMU or sometimes IOMMU)**
- **Trusted Base System Architecture (TBSA)**

Useful links to training

- **The Arm architecture**
- **What does the architecture consist of?**
- **What is architecture?**

11 Next steps

This guide introduced the fundamental principles of what the Arm architecture is, how it had evolved, and its profiles and their applications. This knowledge helps provide a foundation on which you can build as you learn more about Arm technologies.

We have discussed some of the common terms and concepts that are key to understanding the Arm architecture, and the different profiles of the Arm architecture. We have described features that are specific to architecture and micro-architecture, and how Arm architecture terms and concepts appear in Arm architecture reference manuals (Arm ARMs) and other Arm documentation and resources. We have also learned about the different profiles of the Arm architecture and other Arm architectures.

Further guides in this series introduce aspects of the Arm architecture in detail, and provide examples and commentary.

To keep learning about the Armv8-A architecture, see more in our **series of guides**.