Arm Glossary

2.0

Non-Confidential

Copyright © 2022–2023 Arm Limited (or its affiliates).
All rights reserved.

Issue 02

105565_0200_02_en

Arm Glossary

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	8 December 2022	Non-Confidential	First release under new document ID, 105565. Previously published under AEG0014.
0200-01	19 June 2023	Non-Confidential	Second release under new document ID, 105565.
0200-02	24 July 2023	Non-Confidential	Third release under new document ID, 105565.

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND

REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at https://www.arm.com/company/policies/trademarks.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. A

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

A32

The instruction set used by an Armv8 processor that is in AArch32 execution state. A32 is a fixed-width instruction set that uses 32-bit instruction encoding. It is compatible with the Armv7 Arm instruction set.

A32 instruction

An instruction executed by a core that is in AArch32 Execution state and A32 Instruction set state. A32 is a fixed-width instruction set that uses 32-bit instruction encodings. Previously, this instruction set was called the Arm instruction set.

A32 state

When a core is in the AArch32 Execution state, if it is in the A32 Instruction set state then it executes A32 instructions.

A64

The instruction set used by an Armv8 processor that is in AArch64 execution state. A64 is a fixed-width instruction set that uses 32-bit instruction encoding.

A64 instruction

The instruction set used by an Armv8-A core that is in AArch64 Execution state. A64 is a fixed-width instruction set that uses 32-bit instruction encodings.

AAPCS, Arm Architecture Procedure Call Standard

Defines how registers and the stack are used for subroutine calls.

AArch32

The Arm 32-bit Execution state that uses 32-bit general purpose registers, and a 32-bit program counter (PC), stack pointer (SP), and link register (LR). AArch32 Execution state provides a choice of two instruction sets, A32 and T32. In implementations of versions of the Arm architecture before Armv8, and in the Arm R and M architecture profiles, execution is always in AArch32 state.

AArch64

AArch64 is the 64-bit extension to the Arm architecture, first introduced with the Armv8-A architecture profile in October 2011. It adds an optional 64-bit architecture, AArch64, and an associated instruction set, A64. The Arm 64-bit Execution state uses:

- 64-bit general purpose registers
- a 64-bit program counter (PC)
- stack pointer (SP)
- exception link registers (ELR)

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

AArch64 state is supported only in the Armv8-A architecture profile.

ABI, Application Binary Interface

A collection of specifications, some open and some specific to the Arm architecture, that regulate the inter-operation of binary code in a range of execution environments for Arm processors. The base standard specifies those aspects of code generation that must conform to a standard that supports inter-operation. It is aimed at authors and vendors of C and C++ compilers, linkers, and runtime libraries.

ACBC, Assertive Content and Backlight Control

A histogram-based power reduction engine, a constituent part of Assertive Display.

ACE, AXI Coherency Extensions

Provide additional channels and signaling to an AXI interface to support system level cache coherency.

ADC, Assertive Display Control

The entire backlight pipeline of Assertive Display.

ADI, Arm Debug Interface

The ADI connects a debugger to a device. The ADI is used to access memory-mapped components in a system, such as processors and CoreSight components. The ADI protocol defines the physical wire protocols that are permitted, and the logical programmers model.

Advanced SIMD, Advanced Single Instruction Multiple Data

A feature of the Arm architecture that provides Single Instruction Multiple Data (SIMD) operations on a dedicated bank of registers. If an implementation also supports scalar floating-point instructions, the floating-point and Advanced SIMD instructions use a common register bank. Arm Neon technology provides the Advanced SIMD instructions, and therefore these are often called the Neon instructions.

AHB-AP, AHB Access Port

An optional component of the Debug Access Port (DAP) that provides an AHB interface to an SoC. CoreSight supports access to a system bus infrastructure using the AHB Access Port (AHB-AP) in the DAP. The AHB-AP provides an AHB master port for direct access to system memory. Other bus protocols can use AHB bridges to map transactions. For example, you can use AHB to AXI bridges to provide AHB access to an AXI bus matrix.

AHB, Advanced High-performance Bus

An AMBA bus protocol supporting pipelined operation, with the address and data phases occurring during different clock periods. This means the address phase of a transfer can occur during the data phase of the previous transfer. The AHB provides a subset of the functionality of the AMBA AXI protocol.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

AHB-Lite

A subset of the full AMBA AHB protocol specification. AHB-Lite provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect.

AMBA, Advanced Microcontroller Bus Architecture

The AMBA family of protocol specifications is the Arm open standard for on-chip buses. AMBA provides solutions for the interconnection and management of the functional blocks that make up a System-on-Chip (SoC). Applications include the development of embedded systems with one or more processors or signal processors and multiple peripherals.

APB

An AMBA bus protocol for ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Use APB to connect to the main system bus through a system-to-peripheral bus bridge to help reduce system power consumption.

APB-AP, APB Access Port

An optional component of the DAP that provides an AHB interface to an SoC. CoreSight supports access to a system bus infrastructure using the AHB Access Port in the Debug Access Port (DAP). The AHB-AP provides an AHB master port for direct access to system memory. Other bus protocols can use AHB bridges to map transactions. For example, you can use AHB to AXI bridges to provide AHB access to an AXI bus matrix.

APSR, Application Processor Status Register

In AArch32 User mode, a restricted form of the CPSR.

architecturally executed

An instruction is architecturally executed only if it is executed in a simple sequential execution of the program. When such an instruction has been executed and retired, it has been architecturally executed. Any instruction that is treated as a **NOP** because it fails its condition code check, is an architecturally executed instruction. In a processor that performs Speculative execution, an instruction is not architecturally executed if the processor discards the result of the Speculative execution of that instruction.

Arm profiler

A plug-in to the Arm Workbench Integrated Development Environment that provides non-intrusive analysis of embedded software over time, on targets running at frequencies which are typically as high as 250MHz. Targets can be Real-Time System Models (RTSMs) and hardware targets.

Arm TrustZone technology

The hardware and software that enable the integration of enhanced security features throughout an SoC. In Armv6K, Armv7-A and Armv8-M, the Security Extensions implement the TrustZone hardware. In Armv8, EL3 incorporates the TrustZone hardware.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

armar

An object file archiver tool, which might be included as part of a compiler toolchain or development environment that Arm provides. armar enables you to archive ELF object files into a library. See also armclang, armclang integrated assembler, armcc, armasm, armlink, fromelf.

armasm

An assembler, for legacy armasm syntax assembly language source files, which might be included as part of a compiler toolchain or development environment that Arm provides. The assembler produces object files containing machine code. See also armclang, armclang integrated assembler, armcc, armlink, fromelf, armar.

armcc

A compiler, for C and C++ language source files, which might be included as part of a compiler toolchain or development environment that Arm provides. The compiler can produce object files containing machine code or assembly language source files. See also armclang, armclang integrated assembler, armasm, armlink, fromelf, armar.

armclang

A compiler, for C and C++ language source files, which might be included as part of a compiler toolchain or development environment that Arm provides. The compiler can produce object files containing machine code or assembly language source files. This compiler is based on LLVM and Clang. See also armflang, armclang integrated assembler, armcc, armasm, armlink, fromelf, armar.

armclang integrated assembler

An assembler, for GNU syntax assembly source files, which might be included as part of a compiler toolchain or development environment that Arm provides. The assembler produces object files containing machine code. This assembler is based on LLVM and Clang. See also armclang, armcc, armasm, armlink, fromelf, armar.

armflang

A compiler, for Fortran language source files, which might be included as part of a compiler toolchain or development environment that Arm provides. The compiler can produce object files containing machine code or assembly language source files. This compiler is based on LLVM and Flang. See also armclang, armclang integrated assembler, armlink, fromelf, armar.

armlink

A linker, for combining object files and library files by placing code and data at specific addresses, which might be included as part of a compiler toolchain or development environment that Arm provides. The linker can produce executable images, partially linked object files, or shared object files. See also armclang, armclang integrated assembler, armcc, armasm, fromelf, armar.

ATB

An AMBA bus protocol for trace data. The ATB is a common bus used by the trace components to pass trace data in a system in a data-agnostic format. A trace device can use an ATB to share CoreSight capture resources.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

ATB bridge

A synchronous ATB bridge provides a register slice that meets timing requirements by adding a pipeline stage. It provides a unidirectional link between two synchronous ATB domains.
An asynchronous ATB bridge provides a unidirectional link between two ATB domains with asynchronous clocks. This means it connects components in different clock domains.

atomicity

Describes actions that appear to happen as a single operation. In the Arm architecture, atomicity refers to either single-copy atomicity or multi-copy atomicity. The Arm Architecture Reference Manuals define these forms of atomicity.

authentication asynchronous bridge

Transfers authentication signals between two asynchronous clock domains.

authentication synchronous bridge

Transfers authentication signals between two synchronous clock domains.

AWIDE, Arm Workbench IDE

Arm Workbench IDE is based around the Eclipse IDE, and provides additional features to support the Arm development tools that are provided in RVDS.

AXI, Advanced eXtensible Interface

An AMBA bus protocol that supports:

- Separate phases for address or control and data
- Unaligned data transfers using byte strobes
- Burst-based transactions with only start address issued
- Separate read and write data channels
- Issuing multiple outstanding addresses
- Out-of-order transaction completion
- Optional addition of register stages to meet timing or repropagation requirements.

The AXI protocol includes optional signaling extensions for low-power operation.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. B

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

base porting layer

A platform-dependent base driver software component that communicates with the Mali GPU. For example, the base porting layer controls the Mali GPU registers. You implement, or port, the base porting layer onto different target platforms.

BCD file, Board and Chip Definition file

For RealView Debugger, a BCD file enables you to define the memory map and memory-mapped registers for a target development board or processor. Arm provides various BCD files with RVDS for Arm development boards.

beat

An individual transfer within a burst. For example, an INCR4 burst comprises four beats.

big-endian

In Arm architecture, big-endian is defined as the memory organization in which the least significant byte of a word is at a higher address than the most significant byte. For example:

- A byte or halfword at a word-aligned address is the most significant byte or halfword in the word at that address
- A byte at a halfword-aligned address is the most significant byte in the halfword at that address

board file

A debugger uses this term to refer to the top-level configuration file, normally called rvdebug.brd. This file references one or more other configuration files. A board file contains:

- The debug configuration connection-level settings
- References to the debug interface configuration file which identifies the targets on the development platform
- References to any BCD files assigned to a debug configuration

breakpoint unit

In the context of an Arm debugger, a unit in a Chained breakpoint that combines with other breakpoint units to create a complex hardware breakpoint. In an M-profile processor, a hardware debug component that can be part of the Flash Patch and Breakpoint unit.

BSABI, Base Standard Application Binary Interface

Application Binary Interface. A collection of specifications, some open and some specific to the Arm architecture, that regulate the inter-operation of binary code in a range of execution environments for Arm processors. The base standard specifies those aspects of code generation that must conform to a standard that supports inter-operation. It is aimed at authors and vendors of C and C++ compilers, linkers, and runtime libraries.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. C

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

captive thread

Captive threads are all threads that can be brought under the control of RVDS. Special threads, called non-captive threads, are essential to the operation of Running System Debug (RSD) and so are not under debugger control.

chained breakpoint

In the context of an Arm debugger, a unit in a Chained breakpoint that combines with other breakpoint units to create a complex hardware breakpoint. In an M-profile processor, a hardware debug component that can be part of the Flash Patch and Breakpoint unit.

chained tracepoint

In the context of an Arm debugger, a complex tracepoint that comprises multiple tracepoint units.

CI, channel interface

In an ECT device, the channel interface is one of the interfaces on CTI.

CMM

In the context of an Arm debugger, a scripting language provided for compatibility with other debuggers. If you are writing new scripts, Arm recommends that you use the GNU Debugger (GDB) scripting commands because these offer more functionality in the Arm debuggers.

CMSIS, Cortex Microcontroller Software Interface Standard

Defines a common way to access peripheral registers and define exception vectors, the names of the registers of the core peripherals , and the core exception vectors. It is a device-independent interface for RTOS kernels, including a debug channel. CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M Series processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system.

communications channel

The hardware used for communicating between the software running on the processor, and an external host, using the debug interface. When this communication is for debug purposes, it is called the Debug Communications Channel (DCC).

CRF, Condensed Reference Format

An Arm proprietary file format for specifying test vectors. Typically, Arm supplies a script to convert CRF format to Verilog Reference Format (VRF).

condition code field

A 4-bit field in an Arm instruction that specifies the condition under which the instruction executes.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

condition flags

The condition flags are the N, Z, C, and V bits of PSTATE or of a Program Status Register (PSR).

CONSTRAINED UNPREDICTABLE

Where an instruction can result in **UNPREDICTABLE** behavior, the Arm architecture can specify a narrow range of permitted behaviors. This range is the range of **CONSTRAINED UNPREDICTABLE** behavior. All implementations that are compliant with the architecture must follow the **CONSTRAINED UNPREDICTABLE** behavior. However, software must not rely on any **CONSTRAINED UNPREDICTABLE** behavior. When **CONSTRAINED UNPREDICTABLE** appears in body text, it is always in SMALL CAPITALS.

context synchronization event

A context synchronization event is one of:

- In all versions of the Arm architecture:
  - The execution of an ISB instruction that does not fail its condition code check
  - The taking of an exception
  - The return from an exception
- In addition, in Armv8:
  - Exit from Debug state
  - Executing a DCPS instruction
  - Executing a DRPS instruction

The architecture requires a Context synchronization event to guarantee visibility of any change to a System register.

coprocessor

A processor, or conceptual processor, that supplements the main processor to carry out additional functions. In AArch32 Execution state, the Arm architecture defines an interface to up to 16 coprocessors, CP0-CP15. In Armv8, AArch32 state supports only conceptual coprocessors CP10, CP11, CP14, and CP15. In previous versions of the architecture, coprocessors CP0-CP7 are available for implementation defined features, and coprocessors CP8-CP15 are reserved for use by Arm. In all architecture versions for the A and R architecture profiles, in AArch32 state:

- CP15 instructions access the System registers. Some documentation describes this set of registers as the System Control Coprocessor
- CP14 instructions access System registers for debug, trace, and execution environment features.
- The CP10 and CP11 instruction space is for floating-point and Advanced SIMD instructions if supported, including the instructions for accessing the floating-point and Advanced SIMD System registers.

core

The abstract machine defined in the Arm architecture, as documented in an Arm Architecture Reference Manual. A processing element (PE) implementation that is compliant with the Arm

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

architecture must conform with the behaviors described in the corresponding Arm Architecture Reference Manual. Arm processor documentation usually describes a PE as a Core.

core module

In the context of an Arm Integrator development board, an add-on development board that contains an Arm processor and local memory. Core modules can run standalone, or can be stacked onto Integrator development boards.

core register

Processing registers used in AArch32 Execution state, comprising:

- 13 general-purpose registers, R0 to R12, that software uses for all data processing when using the base instruction set instructions
- SP, the Stack Pointer, that can also be referred to as R13
- LR, the Link Register, that can also be referred to as R14
- PC, the Program Counter, that can also be referred to as R15

In some situations, software can use SP, LR, and PC for processing. The instruction descriptions include any constraints on the use of SP, LR, and PC.

CoreSight ECT, Embedded Cross Trigger

A modular system that supports the interaction and synchronization of multiple triggering events with an SoC. It comprises:

- *Cross Trigger Interface* (CTI)
- *Cross Trigger Matrix* (CTM)

CoreSight ETB, Embedded Trace Buffer

A Logic block that extends the information capture functionality of a trace macrocell.

CoreSight ETM, Embedded Trace Macrocell

A hardware macrocell that, when connected to a processor, outputs trace information on a trace port. The ETM provides processor driven trace through a trace port compliant to the ATB protocol. An ETM always supports instruction trace, and might support data trace.

CoreSight STM, System Trace Module

A trace source designed primarily for high-bandwidth trace of instrumentation embedded into software. This instrumentation is made up of memory-mapped writes to the STM, which carry information about the behavior of the software.

CoreSight TMC, Trace Memory Controller

Controls the capturing or buffering of trace that is generated by trace sources within a system. The TMC receives trace from an ATB interface and can be configured to perform one of the following:

- Route the trace out over an AXI master interface, to allow trace to be captured in system memory or in other peripherals

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- Capture the trace in a circular buffer in dedicated SRAM
- Buffer the trace in a First In First Out (FIFO) style, to smooth over peaks in trace bandwidth

CPAK, Cycle Model Performance Analysis Kit

Pre-built, ready-to-simulate virtual systems that include models of Arm IP plus software.

CPSR, Current Program Status Register

In AArch32 state, the register that holds the current program status.

CRF, Condensed Reference Format

An Arm proprietary file format for specifying test vectors. Typically, Arm supplies a script to convert CRF format to Verilog Reference Format (VRF).

cross-path blocking

Cross-path blocking occurs when a divergent node has congestion on one of its output nodes which blocks bus traffic to its other output nodes.

CTI, Cross Trigger Interface

Part of an Embedded Cross Trigger (ECT) device. In an ECT, the CTI provides the interface between a processor or ETM and the CTM.

CTM, Cross Trigger Matrix

Part of an Embedded Cross Trigger (ECT) device. In an ECT device, the CTM combines the trigger requests generated by CTIs and broadcasts them to all CTIs as channel triggers.

cycle model

Model of Arm IP that provides a cycle-accurate programmer's view for virtual prototyping.

Cycle Model Studio

Arm GUI that uses the Cycle Model Compiler to compile RTL into a Cycle Model for various simulation environments.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

4. D

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

DAPBUS interconnect

The DAPBUS interconnect connects a *Debug Port* (DP) to the *Access Ports* (APs) in a CoreSight DAP.

data side

The processor logic responsible for the movement and processing of data.

data-active write transaction

A transaction that has completed the address transfer or leading write data transfer, but has not completed all its data transfers.

DBGTAP, Debug Test Access Port

A debug control and data interface based on IEEE 1149.1 *JTAG Test Access Port* (TAP). The interface has four or five signals.

DCC, Debug Communications Channel

A channel that carries data between a debugger and debug logic in the target processor. It can do this without stopping the program flow or causing entry to Debug state, but can also be used when the target is in Debug state. The DCC is part of the debug register interface of the target.

debug agent

In the RealView Debugger, the debug agent provides target-side support for *Running System Debug* (RSD). The debug agent can be a thread or be built into the RTOS. The debug agent and RealView Debugger communicate with each other using the DCC. This passes data between the debugger and the target using a hardware debug interface, without stopping the program or entering debug state.

DTSL, Debug and Trace Services Layer

Software layer that sits between the debugger and the RDDI target access API.

debug configuration

In the context of an Arm debugger, a debug configuration defines a debugging environment for the development platform that is accessed through a particular debug interface. Multiple debug configurations can be created for a debug interface, each providing a separate debugging environment to different development platforms, or different debugging environments to the same development platform. All debug configurations are stored in the main debugger board file. Each configuration might reference one or more BCD files.

debug illusion

The view of the software being debugged that a debugger presents to its user. The features of the debug illusion include:

- Mapping between assembler code and source code, including displaying assembler and source code simultaneously if required
- Support for source-level stepping and breakpoints.*Visibility of the source-level function call stack, even when called functions are generated inline
- Display of variable values and structure field values, even when these values migrate between various locations. This includes displaying registers and the stack.

debug interface

In the context of RealView Debugger, the debug interface identifies the targets on your development platform, and provides the mechanism that enables RealView Debugger to communicate with those targets. The debug interface corresponds directly to a piece of hardware or a software simulator.

DEBUG_RECOV

Short name for the Debug recovery power mode that can apply to the cores or the DSU.

DebugBlock

The DebugBlock combines the functions, registers, and interfaces for debugging the DSU while the cores and cluster are powered down. Therefore, this component is separate from the cluster and in a separate power domain.

Development Studio 5

The suite of software development tools, together with supporting documentation and examples, that enable you to write and debug applications for the Arm family of processors. DS-5 supersedes RealView Development Suite.

DS-5

The suite of software development tools, together with supporting documentation and examples, that enable you to write and debug applications for the Arm family of processors. DS-5 supersedes RealView Development Suite.

DSM, Design Simulation Model

A functional simulation model of the device that is derived from the RTL but that does not reveal its internal structure. The DSM does not model any features added during synthesis such as internal scan chains.

DSU, DynamIQ Shared Unit

Comprises the L3 memory system, control logic, and external interfaces to support a DynamIQ cluster. The DynamIQ cluster microarchitecture integrates one or more cores with the DSU to form a cluster that is implemented to a specified configuration.

DTM, Data Timing Module

In the context of physical IP, a data timing module that synchronizes incoming and outgoing data. The DTM is a component of PHY to include I/Os and PLL.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

DVS, Device Validation Suite

A set of tests to check the functionality of a device against that defined in the Technical Reference Manual.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. E

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Eclipse for DS-5

Eclipse for DS-5 is based around the Eclipse IDE, and provides additional features to support the Arm development tools that are provided in DS-5.

ECT, Embedded Cross Trigger

Embedded Cross Trigger (ECT) is a modular system that supports the interaction and synchronization of multiple triggering events with an SoC. It comprises:

- Cross Trigger Interface (CTI)
- Cross Trigger Matrix (CTM)

ELR, Exception Link Register

In AArch64 state, there is a dedicated LR for each implemented Exception level, for example, ELR_EL1. The ELR holds the exception return address.

embedded assembler

Embedded assembler is a compiler feature that enables writing a C or C++ function entirely using assembly instructions, from within a C or C++ source language file.

Embedded-System Graphics Library

A standardized set of functions that communicate between graphics software, such as OpenGL ES or OpenVG drivers, and the platform-specific windowing system that displays the image.

EmbeddedICE logic

An on-chip logic block that provides TAP-based debug support for an Arm processor. It is accessed through the DAP on the Arm processor.

EmbeddedICE-RT

Hardware provided by an Arm processor to aid debugging in real-time.

ESSL compiler

The compiler that translates shaders, that are written in ESSL, into binary code for the shader units in the Mali GPU. There are two versions of the ESSL compiler:

- the on-target compile
- the offline compiler

ETB, Embedded Trace Buffer

A logic block that extends the information capture functionality of a trace macrocell.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

ETM, Embedded Trace Macrocell

A hardware macrocell that, when connected to a processor, outputs trace information on a trace port. The Embedded Trace Macrocell (ETM) provides processor driven trace through a trace port compliant to the ATB protocol. An ETM always supports instruction trace, and might support data trace.

ETV, Extended Target Visibility

Enables RealView Debugger to access features of the underlying target such as chip-level information provided by the hardware manufacturer or SoC designer.

event asynchronous bridge

Enables an event signal to cross a clock domain boundary.

Exception level

In the Armv8 architecture, a program executes at one of four Exception levels. In AArch64, the Exception level determines the level of execution privilege. Exception levels provide a logical separation of software execution privilege that applies across all operating states of the Armv8 architecture. System software determines the Exception level, and therefore the level of privilege, at which software runs.

Exception model

Armv8 Exception model, which defines up to four Exception levels, and that provides an execution privilege hierarchy.

exception vector

A fixed address that contains the address of the first instruction of the corresponding exception handler.

exceptional state

In an Armv7 implementation that includes a VFP subarchitecture, in floating-point operation, if the floating-point hardware detects an exceptional condition, the Arm floating-point implementation sets the FPEXC bit and loads a copy of the exceptional instruction to the FPINST register. When in the exceptional state, the issue of a trigger instruction to the floating-point extension causes a bounce.

execution vehicle

A part of the debug target interface that processes requests from the client tools to the target.

execution view

In Arm compiler toolchains, execution view describes the memory map layout by showing each region and section, in the image, in terms of the addresses where the regions and sections are located during execution.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

explicit access

A read from memory, or a write to memory, generated by a load or store instruction executed by the core. Reads and writes generated by hardware translation table accesses are not explicit accesses.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

6. F

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Fast Model

Instruction-accurate model for early software development on Arm systems. You can use the model, together with Arm profiler and an Arm debugger, to optimize and debug your applications early in the development cycle.

Fast Models library

Instruction-accurate model for early software development on Arm systems. You can use the model, together with Arm profiler and an Arm debugger, to optimize and debug your applications early in the development cycle.

Fast Models portfolio

Instruction-accurate model for early software development on Arm systems. You can use the model, together with Arm profiler and an Arm debugger, to optimize and debug your applications early in the development cycle.

Fastline

An MTI trace plug-in that allows you to instrument Fast Models for analysis in Streamline.

FCSE, Fast Context Switch Extension

Before Armv8, Fast Context Switch Extension (FCSE) is an extension to the Arm architecture that modifies the behavior of the memory system. It enables multiple programs running on the core to use identical address ranges, while ensuring that the addresses they present to the rest of the memory system differ.

From Armv6, Arm deprecates use of the FCSE. The FCSE is optional in Armv7, and obsolete from the Armv7 Multiprocessing Extensions.

FIQ

FIQ interrupt. nFIQ is one of two interrupt signals on many Arm processors.

flat address mapping

A memory system where the physical address for every access is equal to its virtual address. Sometimes called a flat mapping.

FOM data

FOM data is a comparison of the normalized performance and leakage characteristics among different standard cell libraries available within Arm POP IP.

formatter

In an ETB or TPIU, an internal input block that embeds the trace source ID in the data to create a single trace stream.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

FPB, Flash Patch and Breakpoint

In an Arm M-profile processor, a Flash Patch and Breakpoint (FPB) can:

- Remap sections of ROM, typically flash memory, to regions of RAM
- Set breakpoints on code in ROM. It can be used for debug, and to provide a code or data patch to an application that requires field updates to a product ROM.

FTC, Fragment Thread Creator

In a Mali GPU, a functional block that creates and issues threads to the tri-pipe processing unit pipeline. It is used for all fragment jobs output from the tiler.

fromelf

An object file format conversion tool, which might be included as part of a compiler toolchain or development environment that Arm provides. `fromelf` enables you to convert ELF images and object files, and to display information about those files. See also `armclang`, `armclang integrated assembler`, `armcc`, `armasm`, `armlink`, `armar`.

ftrace, function tracer

A tracing framework for the Linux kernel.

FULL_RET

Short name for the Core dynamic power mode for cores and for the full retention power mode for some DSUs.

FUNC_RET

Short name for the functional retention power mode that typically only applies to the DSU.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

7. G

general-purpose register

Processing registers used in AArch32 Execution state, comprising:

- 13 general-purpose registers, R0 to R12, that software uses for all data processing when using the base instruction set instructions
- SP, the Stack Pointer, that can also be referred to as R13
- LR, the Link Register, that can also be referred to as R14
- PC, the Program Counter, that can also be referred to as R15

In some situations, software can use SP, LR, and PC for processing. The instruction descriptions include any constraints on the use of SP, LR, and PC.

generic thread creator

In a Mali GPU, the Generic Thread Creator is a functional block that creates and issues threads to the tri-pipe processing unit pipeline. It is used for all non-fragment jobs, including vertex shading, geometry shading, and OpenCL jobs.

GIC, Generic Interrupt Controller

Abbreviation for Generic Interrupt Controller. A GIC is an exclusive block of IP that performs critical tasks of interrupt management, prioritization and routing. GICs are primarily used for boosting processor efficiency and supporting interrupt virtualization. GICs are implemented based on Arm GIC architecture which has evolved from GICv1 to the latest version GICv3/v4. Arm has a number of multi-cluster CPU interrupt controllers that provide a range of interrupt management solutions for all types of Arm Cortex processor systems.

GIC CPU interface

A common interrupt controller programming interface, defined by the GIC architecture.

GPU, Graphics Processing Unit

A hardware accelerator for graphics systems using OpenGL ES and OpenVG. The Mali-200, Mali-300, and Mali-400 MP Graphics Processing Units (GPUs) comprise of a vertex processor and one or more fragment processors. Mali-T600 series GPUs consist of one or more shader cores that can execute vertex or fragment shaders.

granular power requester

Used by a debugger to control powerup and powerdown of specific components within a CoreSight system.

graphics application

A custom program that executes in the Mali graphics system and displays content in a frame buffer for transfer to a display.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

GTC, Generic Thread Creator

In a Mali GPU, the Generic Thread Creator is a functional block that creates and issues threads to the tri-pipe processing unit pipeline. It is used for all non-fragment jobs, including vertex shading, geometry shading, and OpenCL jobs.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

8. H

Halted System Debug

In the context of an Arm debugger, this means that a target can only be debugged when it is not running. With the target stopped, RealView Debugger presents OS awareness information by reading and interpreting target memory.

Halting Debug

In Arm A-profile and R-profile processors, in AArch32 state, one of two mutually exclusive debug modes. When Halting debug is enabled, core execution halts when a breakpoint or watchpoint is encountered. You can use the debug interface to examine and alter all core state, memory, input and output locations.

hardware breakpoint

In the context of an Arm debugger, a unit in a Chained breakpoint that combines with other breakpoint units to create a complex hardware breakpoint. In an M-profile processor, a hardware debug component that can be part of the Flash Patch and Breakpoint unit.

hierarchical tiler

In the context of a Mali GPU, the hierarchical tiler sorts all the primitives in the scene into a hierarchical list structure. These lists are then processed by the shader cores.

high registers

In AArch32 state, core registers R8-R12, SP, LR, and PC. Some T32 instructions cannot access these registers.

high vectors

In AArch32 state, one of two possible locations for exception vectors. The high vector address range is near the top of the address space, rather than at the bottom.

hint instruction

A hint instruction provides information that the hardware can take advantage of.

Hit-Under-Miss

A Hit-Under-Miss (HUM) buffer that means a memory access can hit in the cache, even though there has been a data miss in the cache.

HSD

In the context of an Arm debugger, this means that a target can only be debugged when it is not running. With the target stopped, RealView Debugger presents OS awareness information by reading and interpreting target memory.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

HTM

A trace source that makes bus information visible. This information cannot be inferred from the processor using just a trace macrocell. HTM trace can provide:

- An understanding of multi-layer bus utilization
- Software debug, for example, visibility of access to memory areas and data accesses
- Bus event detection for trace trigger or filters, and for bus profiling.

HUM

A Hit-Under-Miss (HUM) buffer that means a memory access can hit in the cache, even though there has been a data miss in the cache.

HVC

HyperVisor Call instruction used in both the Armv7 and Armv8 architectures. Used to generate a synchronous exception that is handled by a hypervisor running in EL2.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

9. I

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

ICE Extension Unit

A hardware extension to the EmbeddedICE logic that provides more breakpoint units.

IDAU

The Armv8-M architecture defines a mechanism whereby an implementation can define whether any particular address is exempt from checking, is NSC or Secure. If the Armv8-M Security Extension is included, then the internal Security Attribution Unit (SAU) or an external Implementation Defined Attribution Unit (IDAU) determines the Security state attributed to each address.

IEM, Intelligent Energy Management

An energy manager solution consisting of both software and hardware components that function together to prolong battery life in an Arm processor-based device.

iIntegrator

A range of Arm hardware development platforms. Core modules are available that contain the processor and local memory.

implementation-specific

In the context of Arm trace macrocells, behavior that is not architecturally defined, and might not be documented by an individual implementation. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.

imprecise tracing

In an Arm trace macrocell, a filtering configuration where instruction or data tracing can start or finish earlier or later than expected. Most imprecise cases cause tracing to start or finish later than expected.

indirect read

When an instruction uses a System register value to establish operating conditions, that use of a System register is an indirect read of that System register.

indirect write

Occurs when the contents of a register are updated by some other mechanism than a direct write.

Input section

In Arm compiler toolchains, an input section is an individual section from an input object file. It contains code, initialized data, or describes a fragment of memory that is not initialized or that must be set to zero before the image can execute. `armlink` operates on input sections by grouping them into bigger building blocks called output sections and regions.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Instruction Abort

An indication to the core of an attempted instruction fetch that is not permitted. The Instruction Abort might be generated by access permission checks performed by the memory system on the core, or might be signaled by the memory system. An exception is taken only if the core attempts to execute the instruction. No exception is taken if the core does not execute an instruction that is attempted to fetch or prefetch from a faulting memory location. The AArch64 architecture definitions introduce the term Instruction Abort. Descriptions of AArch32 state use the term Prefetch Abort.

instruction breakpoint

In the context of an Arm debugger, a location in the image containing an instruction that, if executed, activates a breakpoint. The breakpoint activation can be delayed by assigning condition qualifiers, and subsequent execution of the image is determined by any actions assigned to the breakpoint.

instruction set system model

In the context of RVDS, ISSM is a set of models that simulate the Arm Cortex family of processors. These models are provided with RVDS.

instruction side

The processor logic responsible for the fetching of instructions and providing them to the core.

Instruction Synchronization Barrier

An operation to ensure that any instruction that comes after the ISB operation is fetched only after the ISB has completed.

Instrumentation Trace Macrocell

Hard real-time debugging requires close interaction with the processor. Tracing provides a chronological picture of a system’s inner workings up to, starting from or in the vicinity an event, mainly to guide a human in understanding a faulty program. For processors with Arm CoreSight such as Cortex-M3 and Cortex-M4, the Instrumentation Trace Macrocell (ITM) provides a lightweight, nonintrusive way to collect debug trace output.

intermediate physical address

In an implementation of virtualization, the intermediate physical address to which a Guest OS maps a virtual address.

IRI, Interrupt Redistribution Infrastructure

One of two parts defined by the GICv3 architecture.

ISSM

In the context of RVDS, ISSM is a set of models that simulate the Arm Cortex family of processors. These models are provided with RVDS.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

IT block

In AArch32 state, a block of up to four instructions following a T32 IT (If-Then) instruction. Each instruction in the block is conditional. The condition for each instruction is either the same as or the inverse of the condition that is specified by the IT instruction.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

10. J

Jazelle DBX

The Jazelle Extension technology is called Jazelle DBX.

Jazelle Extension

Some Arm processors before Armv8 included the Jazelle Extension, to provide hardware execution of some Java bytecodes in AArch32 state, as part of a Java Virtual Machine (JVM) implementation. From Armv8, the architecture supports only an AArch32 Trivial Jazelle implementation, meaning a JVM must be implemented entirely in software. The Jazelle Extension technology is called Jazelle DBX.

Jazelle RCT

On an Arm processor, a modification of the T32 instruction set to make it a better target for code generated at runtime. This is also called the T32 Execution Environment (T32EE). From Armv8, Arm processors do not support Jazelle RCT.

Jazelle state

In AArch32 state, in the Jazelle Instruction set state the core executes Java bytecodes as part of a Java Virtual Machine (JVM). From Armv8, Arm processors do not support Jazelle state.

Jazelle Technology Enabling Kit

A kit containing source code for integration with a Java Virtual Machine to enable Jazelle DBX on an Arm-based host platform.

job object

In the context of graphics processing, a Mali job system component that provides jobs with required content for Mali GPU execution.

job system back-end

In the context of graphics processing, a Mali job system component that shares some priority handling, but mainly requests jobs from queues and sends job requests to the Mali GPU.

JTAG, Joint Test Action Group

An IEEE group that is focused on silicon chip testing methods. Many debug and programming tools use a Joint Test Action Group (JTAG) interface port to communicate with processors.

JTAG Access Port

An optional component of the DAP that provides debugger access to on-chip scan chains.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

JTAG interface unit

In the context of Arm RealView tools, a protocol converter that converts low-level commands from RVDS debuggers into JTAG signals to the processor, for example to the EmbeddedICE logic and the ETM.

JTAG-AP

An optional component of the DAP that provides debugger access to on-chip scan chains.

JTAG-DP

In an external debugger, a block that acts as a master on a system bus and provides access to the target.

JTEK

A kit containing source code for integration with a Java Virtual Machine to enable Jazelle DBX on an Arm-based host platform.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

11. L

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

load view

In Arm compiler toolchains, load view describes the memory map layout by showing each region and section in the image, according to the addresses where the regions and sections are located when the image is loaded into memory, before execution starts.

LPI, Low-Power Interface

An interface used for controlling the transitions between power modes.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

12. M

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Mali MMU

A full-featured Memory Management Unit (MMU) that is present on Mali GPUs.

Monitor debug

In Arm A-profile and R-profile processors, in AArch32 state, one of two mutually exclusive debug modes. When Monitor debug is enabled, a debug event generates a debug exception, that is taken as a Prefetch Abort or Data Abort exception. Breakpoints and watchpoints are examples of debug events.

MPAM, Memory System Resource Partitioning and Monitoring

Memory System Resource Partitioning and Monitoring (MPAM). MPAM is designed to align the division of memory-system performance between software by means of following two approaches:

- Memory-system resource partitioning, by means of partition identifiers (PARTIDs)
- Memory-system resource usage monitoring, by means of a *Performance Monitoring Group* (PMG).

MPU, Memory Protection Unit

A hardware unit that controls a limited number of protection regions in memory. An MPU is the major component of an Arm Protected Memory System Architecture (PMSA).

MTB, Micro Trace Buffer

The MTB provides a simple execution trace capability to M-series processors. The MTB provides a lighter option for instruction trace requirements for software development. Unlike the Embedded Trace Macrocell(ETM) or the Program Trace Macrocell(PTM) trace solutions, the MTB does not require dedicated trace connection. However, the amount of trace history provided by the MTB is limited by the size of SRAM allocated for trace operations.

Multi-ICE

A JTAG-based tool for debugging embedded systems.

Multi-layer interconnect

An interconnect scheme similar to a cross-bar switch. Each manager on the interconnect has a direct link to each subordinate that is not shared with other managers. This means each manager can process transfers in parallel with other managers. Contention in a multi-layer interconnect only occurs at a payload destination, typically a subordinate.

Multi-master AHB

Typically a shared, not multi-layer, AHB interconnect scheme. More than one manager connects to a single AMBA AHB link. In this case, the bus is implemented with a set of full AMBA AHB

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

manager interfaces. Managers that use the AMBA AHB-Lite protocol must connect through a wrapper to supply full AMBA AHB manager signals to support multi-manager operation.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

13. N

Neon instruction set

The Arm technology that provides SIMD processing using a dedicated SIMD and floating-point register bank. Registers in this bank can be accessed as 128-bit registers, 64-bit registers, 32-bit registers, 16-bit registers, or 8-bit registers.

Neon SIMD, Neon Single Instruction Multiple Data

Arm Neon technology is an advanced Single Instruction Multiple Data (SIMD) architecture extension for the Arm Cortex-A series and Cortex-R52 processors.

Normal world

In software descriptions of the operation of an Arm core, effectively the environments that contain two virtual processors that run on a single core. The Secure world processes operations that are security-critical, and non security-critical operations are performed in the Normal world. Hardware descriptions use Secure state to describe a core that is executing in the Secure world, and Non-secure state to describe a core that is executing in the Normal world.

NSAID, Non-secure Access ID

To differentiate between Protected and Non-Trusted entities, Arm defines 16 states that mark all processes within hardware and software. These states are defined using the Non-secure Access ID (NSAID), and each initiating device in the SoC has one or more NSAID values assigned in hardware. The NSAID enables other components to identify the initiating device for a particular transaction, and to identify whether the device is treated as Non-protected and therefore permitted to read data from other Non-protected masters.

NVIC

The Nested Vectored Interrupt Controller (NVIC) facilitates low-latency exception and interrupt handling and controls power management. The Arm Architecture Reference Manual provides more information on the NVIC registers.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

14. O

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

OFF

Short name for the Off power mode that can apply to the cores or the DSU.

OFF_EMU

Short name for the Emulated off power mode that can apply to the cores or the DSU.

offline compiler

A command-line tool that translates vertex shaders and fragment shaders written in the ESSL into binary vertex shaders and binary fragment shaders that you can link and run on the Mali GPU.

ON

Short name for the On power mode that can apply to the cores or the DSU.

on-target compiler

A component of the Mali GPU OpenGL ES 2.0 driver that translates shader source files provided by the graphics application, into binary shader code, at runtime.

OpenGL ES driver

Part of a driver stack that translates OpenGL ES API commands into data and instructions for the Mali GPU. Only the device driver controls the Mali GPU directly.

OpenVG driver

Part of a driver stack that translates OpenVG API commands into data and instructions for the Mali GPU. Only the device driver controls the Mali GPU directly.

OS-awareness

OS-awareness is a feature provided by RealView Debugger that enables you to:

- debug applications running on an embedded OS development platform, such as a Real-Time Operating System (RTOS)
- present thread information and scope some debugging operations to specific threads.

output section

A contiguous sequence of input sections that have the same RO, RW, or ZI attributes. The sections are grouped together in larger fragments called regions. The regions are grouped together into the final executable image.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

15. P

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

PBHA, Page-Based Hardware Attributes

An optional, implementation-defined feature provided by cores. It allows software to set up to two bits in the translation tables, which are then propagated though the memory system with transactions, and can be used in the system to control system components.

PC, Program Counter

The PC holds the virtual address of the next instruction to be executed, and:

- In AArch32 state, the PC is a 32-bit register, and is register R15 in the general-purpose register file
- In AArch64 state, the PC is a 64-bit register, that is independent of the general-purpose registers X0-X30. In AArch64 state, software cannot write directly to the PC.

PE, Processing Element

The abstract machine that is defined in the Arm architecture, as documented in an Arm Architecture Reference Manual. A processing element implementation that is compliant with the Arm architecture must conform with the behaviors that are described in the corresponding Arm Architecture Reference Manual.

PFT, Program Flow Trace

The PFT architecture assumes that any trace decompressor has a copy of the program being traced, and generally outputs only enough trace for the decompressor to reconstruct the program flow. However, its trace output also enables a decompressor to reconstruct the program flow when it does not have a copy of parts of the program, for example because the program uses self-modifying code. A *Program Flow Trace Macrocell* (PTM) implements the Program Flow Trace architecture.

PIL, Processor Integration Layer

A level of design hierarchy that includes an Arm processor and its immediate CoreSight components ready for integration into a CoreSight debug and trace system.

PMU, Performance Monitoring Unit

The Performance Monitor architecture defines performance monitor counters and logic that can be used to gather data about the processing element (PE) operation during runtime. The Performance Monitors Extensions provide information about the possible options for a particular Architecture profile, the *Performance Monitoring Unit* (PMU) is a physical implementation of a Performance Monitors Extension.

POP, Performance Optimization Package

A package for the implementation of an Arm processor using Arm Artisan-optimized logic and memory physical IP

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

PPI, Private Peripheral Interrupt

In the context of the GIC, this is a peripheral interrupt that is specific to a single processor.

Prefetch Abort

An abort occurs when an illegal memory access causes an exception. An abort can be generated by the hardware that manages memory accesses, or by the external memory system. The hardware that generates the abort might be a *Memory Management Unit* (MMU) or a *Memory Protection Unit* (MPU).

Procedure Call Standard for the Arm Architecture

The Arm Architecture Procedure Call Standard defines how registers and the stack are used for subroutine calls.

profiling

In the context of RealView Trace, the accumulation of statistics during execution of a program to measure performance or to determine critical areas of code.

PSR, Program Status Register

Holds status and control information for a core, or for a program running on the core. When executing in AArch32 state, the *Current Program Status Register* (CPSR) is the active PSR that affects operation of the core. When executing in AArch64 state, PSTATE holds equivalent status information, but is not accessible as a single register. The *Saved Program Status Register* (SPSR) is a copy of the current state of the cores saved by the hardware on taking an exception. At the point where a core recognizes an exception:

- If the exception is taken to AArch32 state, it copies the CPSR into the SPSR
- If the exception is taken to AArch64 state, it saves the current PSTATE to the SPSR

PSTATE

In Armv8, an abstraction of process state. All of the instruction sets include instructions that operate directly on elements of PSTATE. In previous versions of the architecture, the CPSR provides the equivalent functionality.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

16. R

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

RAO, Read-As-One

Hardware must implement the field as reading as all 1s. Software can rely on the field reading as all 1s. This description can apply to a single bit that reads as 1, or to a field that reads as all 1s.

RAO/SBOP, Read As One, Should Be One or Preserved

Hardware must implement the field as Read-as-One, and must ignore writes to the field. Software can rely on the field reading as all 1s, but must use an SBOP policy to write to the field. This description can apply to a single bit that reads as 1, or to a field that reads as all 1s.

RAO/WI, Read As One, Writes Ignored

Hardware must implement the field as Read-as-One, and must ignore writes to the field. Software can rely on the field reading as all 1s, and on writes being ignored.This description can apply to a single bit that reads as 1, or to a field that reads as all 1s.

RAZ, Read As Zero

Hardware must implement the field as reading as all 0s. Software can rely on the field reading as all 0s. This description can apply to a single bit that reads as 0, or to a field that reads as all 0s.

RAZ/SBZP, Read-As-Zero, Should-Be-Zero-or-Preserved

Hardware must implement the field as Read-as-Zero, and must ignore writes to the field. Software can rely on the field reading as all 0s, but must use an SBZP policy to write to the field. This description can apply to a single bit that reads as 0, or to a field that reads as all 0s.

RAZ/WI, Read-As-Zero, Writes Ignored

Hardware must implement the field as Read-as-Zero, and must ignore writes to the field. Software can rely on the field reading as all 0s, and on writes being ignored. This description can apply to a single bit that reads as 0, or to a field that reads as all 0s.

RCT, Runtime Compilation Target

On an Arm processor, a modification of the T32 instruction set to make it a better target for code generated at runtime. This is also called the T32 Execution Environment (T32EE). From Armv8, Arm processors do not support Jazelle RCT.

Read-Allocate

A technique that deals with data store accesses to caches. In a Read-Allocate cache, the data is simply stored to main memory. Cache lines are only allocated to memory location when data is read or load

read, modify, write

In a read, modify, write sequence, a value is read to a general-purpose register, the relevant fields updated in that register, and the new value written back.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

RealMonitor

A small program that, when integrated into your target application or Real-Time Operating System (RTOS), enables you to observe and debug your target while parts of your application continue to run.

RealView Debugger Trace

Part of RVDS that extends the debugging capability with the addition of real-time program and data tracing. It is available from the RealView Debugger Code window.

RealView Trace and RealView Trace 2

Works in conjunction with RealView ICE to provide real-time trace functionality for software running in System-on-Chip devices with deeply embedded Arm processors. RealView Trace 2 also supports data streaming directly to Arm Profiler, providing real-time hardware platform profiling.

Redistributor

A Redistributor is a component of the GIC architecture. It is part of the interrupt routing infrastructure that is connected to the CPU interface, and thereby to the core.

replicator

In an Arm trace macrocell, enables two trace sinks to be wired together and to operate independently on the same incoming trace stream. The input trace stream is output onto two independent ATB ports.

RES0

A reserved bit or field with Should-Be-Zero-or-Preserved (SBZP) behavior. Used for fields in register descriptions. Also used for fields in architecturally defined data structures that are held in memory, for example in translation table descriptors. **RES0** is not used in descriptions of instruction encodings. Within the architecture, there are some cases where a register bit or bitfield can be **RES0** in some defined architectural context, but can have different defined behavior in a different architectural context. For any **RES0** bit or field, software must not rely on the bit or field reading as 0, and must use an SBZP policy to write to the bit or field.

RES0H

A reserved bit or field with Should-Be-Zero-or-Preserved (SBZP). This behavior uses the Hardwired to 0 subset of the **RES0** definition.

RES1

A reserved bit or field with Should-Be-One-or-Preserved (SBOP) behavior. Used for fields in register descriptions. Also used for fields in architecturally defined data structures that are held in memory, for example in translation table descriptors. **RES1** is not used in descriptions of instruction encodings. Within the architecture, there are some cases where a register bit or bitfield can be **RES1**in some defined architectural context, but can have different defined behavior in a different architectural context. For any **RES1** bit or bitfield, software must not rely on the bit or bitfield reading as 1, and must use an SBOP policy to write to the bit or bitfield.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

RES1H

A reserved bit or field with Should-Be-Zero-or-Preserved (SBZP). This behavior uses the Hardwired to 1 subset of the **RES1** definition.

root region

In Arm compiler toolchains, a root region refers to an ELF region, which is an execution region in the image, where the address in the load view and in the execution view are the same. The initial entry point of an image and the vector table of a processor must be in a root region.

ROPI, Read-Only Position Independent

In Arm compiler toolchains, a Read-Only Position Independent (ROPI) segment is often Position-Independent Code (PIC), but can be be read-only data, or a combination of PIC and read-only data. A program is Read-Only Position-Independent (ROPI) if all its read-only segments can be loaded to and used from any address in memory, without having to be recompiled. A read-only segment is often usually Position-Independent Code (PIC), but can be read-only data, or a combination of PIC and read-only data.

rounding mode

In floating-point operation, specifies how the exact result of a floating-point operation is rounded to a value that is representable in the destination format. The IEEE 754 standard defines the required rounding modes for compliant floating-point implementations, and Arm implementations support these rounding modes.

Note: The IEEE 754-2008 standard changes the term Rounding mode to Rounding-direction attribute. Arm documentation continues to use the term Rounding mode, as defined in the IEEE 754-1985 standard.

RSD, Running System Debug

In the context of software executing on a core that implements the Arm architecture, Read-Only Position Independent describes code or read-only data that can be placed at any address.

RVCT, RealView Compilation Tools

A suite of tools that, together with supporting documentation and examples, enables you to write and build applications for Arm processors.

RVD, RealView Debugger

An Arm debugger that enables you to examine and control the execution of software running on a debug target. RealView Debugger is supplied as part of RVDS in both Windows and Red Hat Linux versions.

RVDS, RealView Development Suite

The RealView suite of software development tools, together with supporting documentation and examples, that enable you to write and debug applications for the Arm family of processors.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

RVI, RealView ICE

An Arm JTAG interface unit for debugging embedded processor cores that uses a DBGTAP or Serial Wire interface.

RVISS, RealView Instruction Set Simulator

One of the Arm simulators supplied with RVDS. RealView Instruction Set Simulator is a collection of programs that simulate the instruction sets and architecture of various Arm processors. This provides instruction-accurate simulation and enables Arm and Thumb executable programs to be run on non-native hardware. RVISS provides modules that model:

- The Arm processor
- The memory used by the processor

There are alternative predefined models for each of these parts. However, you can create your own models if a supplied model does not meet your requirements.

RWPI, Read-Write Position Independent

In the context of software executing on a core that implements the Arm architecture, read-write code or data that can be placed at any address.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

17. S

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

SBO, Should Be One

Hardware must ignore writes to the field. Software should write the field as all 1s. If software writes a value that is not all 1s, it must expect an **UNPREDICTABLE** result. This description can apply to a single bit that should be written as 1, or to a field that should be written as all 1s.

SBOP, Should-Be-One-or-Preserved

The Armv7 Large Physical Address Extension modified the definition of SBOP to apply to register fields that are SBOP in some but not all contexts. From the introduction of Armv8 such register fields are described as **RES1**. The definition of SBOP given here applies only to fields that are SBOP in all contexts. Hardware must ignore writes to the field. When writing this field, software must either write all 1s to this field, or, if the register is being restored from a previously read state, this field must be written with the previously read value. If this is not done, then the result is **UNPREDICTABLE**. This description can apply to a single bit that should be written as its preserved value or as 1, or to a field that should be written as its preserved value or as all 1s.

SBZ, Should Be Zero

Hardware must ignore writes to the field. Software should write the field as all 0s. If software writes a value that is not all 0s, it must expect an **UNPREDICTABLE** result. This description can apply to a single bit that should be written as 0, or to a field that should be written as all 0s.

SBZP, Should Be Zero or Preserved

The Large Physical Address Extension modifies the definition of SBZP for register bits that are reallocated by the extension, and as a result are SBZP in some but not all contexts. The generic definition of SBZP given here applies only to bits that are not affected by this modification. Hardware must ignore writes to the field. When writing this field, software must either write all 1s to this field, or, if the register is being restored from a previously read state, this field must be written previously read value. If this is not done, then the result is **UNPREDICTABLE**. This description can apply to a single bit that should be written as its preserved value or as 0, or to a field that should be written as its preserved value or as all 0s.

SCC, Serial Configuration Controller

On CoreTile Express daughterboards, a register interface in the test chip or development chip on the daughterboard, that the Versatile Express configuration system uses to configure the chip. On LogicTile Express daughterboards, a register interface in the FPGA on the daughterboard, that the Versatile Express configuration system uses to configure the FPGA. The configuration system loads values into the registers at powerup or reset and during runtime.

SCS, System Control Space

On Cortex-M series processors, a memory-mapped region from `0xE000E000` to `0xE000EFFF` that provides system control and configuration registers, including control of the Nested Vectored Interrupt Controller (NVIC) and debug functions.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

section

In Arm Compiler toolchains, a section refers to an ELF file section, which is the smallest unit of code and data on which a linker can operate.

Secure monitor

In software descriptions, the module that switches the Arm core between Normal World and Secure World execution environments.

Secure world

In software descriptions of the operation of an Arm core, effectively the environments that contain two virtual processors that run on a single core. The Secure world processes operations that are security-critical, and non security-critical operations are performed in the Normal world. Hardware descriptions use Secure state to describe a core that is executing in the Secure world, and Non-secure state to describe a core that is executing in the Non-secure state.

semihosting

A mechanism to communicate Input/Output (I/O) requests from application code to a host workstation running a debugger. For example, you can use semihosting to enable functions in the C library, such as printf() and scanf(), to use the screen and keyboard on the host workstation instead of having a screen and keyboard on the target system.

Shared layer

In general, contains functions used by more than one Mali GPU driver. It contains math functions, texture processing and list utilities.

signaling NaN

In floating-point operation, the floating-point hardware causes an Invalid Operation exception whenever any floating-point operation receives a signaling NaN as an operand. You can use signaling NaNs in debugging, to track down some uses of uninitialized variables.

SIMD, Single Instruction, Multiple Data

In the Arm instruction sets, supported SIMD instructions can comprise:

- Instructions that perform parallel operations on the bytes or halfwords of the Arm core registers
- Instructions that perform vector operations. That is, they perform parallel operations on vectors held in multiword registers. Different versions of the Arm architecture support and recommend different instructions for vector operations.

simple sequential execution

The behavior of an implementation that fetches, decodes, and completely executes each instruction before proceeding to the next instruction. Such an implementation performs no speculative accesses to memory, including to instruction memory. The implementation does not pipeline any phase of execution. In practice, this is the theoretical execution model that the architecture is based on, and Arm does not expect this model to correspond to a realistic implementation of the architecture.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

SOM, Sign-Off Model

A SOM is an opaque, compiled simulation model generated from a technology-specific netlist of an Arm processor, derived after gate level synthesis and timing annotation, that you can use in back-annotated gate-level simulations to prove the function and timing behavior of the device. A SOM provides accurate timing simulation of an SoC, and supports simulation using production test vectors from the Automatic Test Pattern Generation (ATPG) tool. It only supports back-annotation using SDF files. The SOM includes timing information but provides slower simulation than a DSM.

SP, stack pointer

On Arm cores, SP refers to the stack pointer for the hardware-managed stack, and:

- In AArch32 state, the SP is register R13 in the general-purpose register file
- In AArch64 state, there is a dedicated SP for each implemented Exception level

SPC, Serial Power Controller

On Versatile Express CoreTile daughterboards, a register interface in an Arm test chip or development chip that the Versatile Express configuration system uses to configure the power controller within the chip. The SPC registers define the power status, supply levels and clock values of the internal power domains within the chip.

SPI

In the context of an Arm GIC, this is a peripheral interrupt that the Distributor can route to any of a specified combination of processors. Each peripheral interrupt is either:

- Edge-triggered This is an interrupt that is asserted on detection of a rising edge of an interrupt signal and then, regardless of the state of the signal, remains asserted until it is cleared by the conditions defined by this specification.
- Level-sensitive This is an interrupt that is asserted whenever the interrupt signal level is active, and deasserted whenever the level is not active.

SPI, Serial Peripheral Interface

SPI is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards.

SPSR, Saved Program Status Register

A register used to save the state of the core on taking an exception.

STL, Software Test Library

STL is used when testing the functional safety of Arm-based cores to detect failures in functional logic (excluding memory), such as during the startup and run time of the core.

STM, System Trace Macrocell

System Trace Macrocell - A trace source designed primarily for high-bandwidth trace of instrumentation embedded into software. This instrumentation is made up of memory-mapped writes to the STM, which carry information about the behavior of the software.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Supervisor Call

An instruction that causes the processor to take a Supervisor Call exception. Used by the Arm standard C library to handle semihosting.

support code

In a floating-point implementation that requires a floating-point subarchitecture, system software that complements the hardware floating-point implementation. The support code can provide a library of routines that perform operations beyond the scope of the hardware, and can include a set of exception handlers to process exceptional conditions in compliance with the IEEE 754 standard.

SVC, Supervisor Call

An instruction that causes the processor to take a Supervisor Call exception. Used by the Arm standard C library to handle semihosting.

SWI, Software Interrupt

An instruction that causes the processor to take a Supervisor Call exception. Used by the Arm standard C library to handle semihosting.

synchronization primitive

An instruction that is used to ensure memory synchronization, for example LDREX or STREX.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

18. T

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

T32
An instruction set that can be used by an Armv8 processor that is in AArch32 execution state. T32 is a variable-length instruction set that uses both 16-bit and 32-bit instruction encodings. It is compatible with the Armv7 Thumb instruction set.

T32 instruction
An instruction that can be used by a core that is in AArch32 execution state. T32(Thumb) is a variable-length instruction set that uses both 16-bit and 32-bit instruction encodings. It is the only instruction set supported by Arm M-profile processors.

T32 state
When a core is executing in AArch32 state, if it is in T32(Thumb) Instruction set state then it executes T32 instructions.

T32EE instruction
One or two halfwords that specify an operation to perform for a core that is in AArch32 state and in T32EE(ThumbEE) Instruction set state.

T32EE state
In AArch32 state, in the T32EE (ThumbEE) Instruction set state the core executes the T32EE instruction set.

TAP, Test Access Port
The collection of four mandatory and one optional terminals that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are TDI, TDO, TMS, and TCK. In the JTAG standard, the nTRST signal is optional, but this signal is mandatory in Arm processors because it is used to reset the debug logic.

TCK
The electronic clock signal that times data on the TAP data lines TMS, TDI, and TDO.

TDI, Test Data Input
TDI is the electronic signal input to a TAP controller from the data source (upstream). Usually this is seen connecting the RealView ICE run control unit to the first TAP controller.

TDO, Test Data Output
TDO is the electronic signal output from a TAP controller to the downstream data sink. Usually this connects the last TAP controller to the RealView ICE run control unit.

Texture Descriptor
Data structure used by the Mali GPU to describe one texture map.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Thumb-2

The technology, introduced in Armv6T2, that extends the Thumb instruction set to a variable-length instructions set that includes both 16-bit and 32-bit instructions.

tile buffer

A memory buffer inside the Mail GPU that holds the framebuffer contents for the tile that is currently being rendered. The tile buffer can be accessed without using the memory bus.

TLX,Thin Links

A protocol to reduce the number of signals in an AXI point-to-point connection to simplify routing.

TMC, Trace Memory Controller

Controls the capturing or buffering trace generated by trace sources within a system. The TMC receives trace from an ATB interface and can be configured to perform one of the following:

- Route the trace out over an AXI master interface, to allow trace to be captured in system memory or in other peripherals
- Capture the trace in a circular buffer in dedicated SRAM
- Buffer the trace in a First In First Out (FIFO) style, to smooth over peaks in trace bandwidth

TMS, Test Mode Select

A connector pin in a JTAG interface.

TPA, Trace Port Analyzer

A hardware device that captures trace information that is output on a trace port.

TPIU, Trace Port Interface Unit

Drains trace data and acts as a bridge between the on-chip trace data and the data stream captured by a Trace Port Analyzer.

trace driver

A remote debug interface target that controls a piece of trace hardware. That is, the trigger macrocell, trace macrocell, and trace capture tool.

trace funnel

In an Arm trace macrocell, a device that combines multiple trace sources onto a single bus.

trace hardware

A device that contains an Arm trace macrocell.

Tracepoint unit

In the context of an Arm debugger, a unit within a Chained tracepoint that combines with other tracepoint units to create a complex tracepoint.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

trap enable bits

For floating-point operation, the trap enable bits determine whether trapped or untrapped exception handling is selected. If trapped exception handling is selected, the way it is carried out is **IMPLEMENTATION DEFINED**.

trigger instruction

In a floating-point implementation that requires a floating-point subarchitecture, a trigger instruction is a floating-point instruction that causes a bounce when it is issued.

Trigger Interface

Part of an Embedded Cross Trigger (ECT) device. In an ECT, the CTI provides the interface between a processor or ETM and the CTM.

TrustZone Software

A secure software framework that uses the Arm architecture Security Extensions.

TrustZone technology

The hardware and software that enable the integration of enhanced security features throughout an SoC. In Armv6K, Arm7-A, and Armv8-M, the Security Extensions implement the TrustZone hardware. In Armv8, EL3 incorporates the TrustZone hardware.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

19. U

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

UAL, Unified Assembler Language

Unified Assembler Language (UAL) is a common assembly language syntax for both A32 and T32 instructions. Code written using UAL can be built for both A32 and T32 state. UAL syntax can be used with both armasm and armclang integrated assembler.

UMP, Unified Memory Provider

Unified Memory Provider (UMP). Provides a safe way to share memory across processes, drivers and hardware components, possibly using an MMU or MPU for memory protection. The Mali GPU driver stack uses the UMP API for certain optional functionality.

unconditional breakpoint

In the context of an Arm debugger, a breakpoint that does not have a conditional qualifier assigned. The breakpoint activates immediately when it is hit, but subsequent image execution is determined by any actions assigned to the breakpoint.

UNDEFINED

Indicates an instruction that is not architecturally defined. It generates an **UNDEFINED** Instruction exception.

UNK, UNKOWN

An abbreviation indicating that software must treat a field as containing an **UNKNOWN** value.

UNK/SBOP, UNKNOWN/Should Be One or Preserved

Hardware must implement the field as Read-As-One, and must ignore writes to the field.

Software must not rely on the field reading as all 1s, and except for writing back to the register it must treat the value as if it is **UNKNOWN**. Software must use an SBOP policy to write to the field.

This description can apply to a single bit that should be written as its preserved value or as 1, or to a field that should be written as its preserved value or as all 1s.

UNK/SBZP, UNKNOWN/Should Be Zero or Preserved

Hardware must implement the field as Read-As-Zero, and must ignore writes to the field.

Software must not rely on the field reading as all 0s, and except for writing back to the register it must treat the value as if it is unknown. Software must use an SBZP policy to write to the field.

This description can apply to a single bit that should be written as its preserved value or as 0, or to a field that should be written as its preserved value or as all 0s.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

UNP, UNPREDICTABLE

For an Arm processor, **UNPREDICTABLE** means the behavior cannot be relied upon. **UNPREDICTABLE** behavior must not perform any function that cannot be performed at the current or a lower level of privilege using instructions that are not **UNPREDICTABLE**.

UNPREDICTABLE behavior must not be documented or promoted as having a defined effect. An instruction that is **UNPREDICTABLE** can be implemented as **UNDEFINED**.

In an implementation that supports Virtualization, the Non-secure execution of **UNPREDICTABLE** instructions at a lower level of privilege can be trapped to the hypervisor, provided that at least one instruction that is not **UNPREDICTABLE** can be trapped to the hypervisor if executed at that lower level of privilege.

For an Arm trace macrocell, **UNPREDICTABLE** means that the behavior of the macrocell cannot be relied on. Such conditions have not been validated. When applied to the programming of an event resource, only the output of that event resource is **UNPREDICTABLE**. **UNPREDICTABLE** behavior can affect the behavior of the entire system, because the trace macrocell can cause the core to enter debug state, and external outputs can be used for other purposes.

user space gator

An agent installed on a Streamline Performance Analyzer target.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

20. V

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

veneer

In Arm compiler toolchains, veneers are small sections of code generated by the linker and inserted into a program. A linker might generate veneers when a branch involves a destination beyond the branching range.

VFP

The original name of the extension to the Arm architecture that provided floating-point arithmetic. From Armv8-A, the architecture includes support for floating-point instructions, rather than this being an architecture extension.

VFP Support Code

Vector Floating-Point (VFP) support code is code provided to handle cases that VFP hardware is unable to process.

VMSA, Virtual Memory System Architecture

A Virtual Memory System Architecture (VMSA) defines and describes how virtual addresses are translated into physical addresses, including the attributes and permissions that can be assigned to the addresses. In a processing element, the Memory Management Unit (MMU) performs this address translation and enforces these permissions.

0 to 9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

21. W

word-invariant

In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged.

The Arm architecture supports word-invariant systems in Armv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions with unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. Arm strongly recommends that word-invariant systems use the endianness that produces the required byte addresses at all times, possibly apart from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler uses only word-aligned memory accesses.

write completion

The memory system indicates to the core that a write is complete at a point in the transaction where the memory system can guarantee that the write is observable by all processors in the system. An additional recommendation for Device-nGnRnE memory (Strongly-ordered memory), is that a write to a memory-mapped peripheral is complete only when it reaches that memory-mapped peripheral and therefore can trigger any side effects caused by the memory-mapped peripheral. Write completion is not required to ensure that all side effects are globally visible, although some peripherals might define this as a required property of completed writes.

write interleave capability

For an interface to an interconnect, the number of data-active write transactions for which the interface can transmit data. This is counted from the earliest transaction.

write interleave depth

The number of data-active write transactions for which the interface can receive data.

Copyright © 2022–2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential