

AArch64 ELF Conventions for Binary Analysis

2025Q4

Date of Issue: 23rd January 2026

The ARM logo, consisting of the lowercase letters 'arm' in a bold, blue, sans-serif font.

1 Preamble

1.1 Abstract

This document specifies code patterns and ELF conventions for AArch64 binaries, enabling Binary Analysis tools to recognize them more easily.

1.2 Keywords

ELF, AArch64 ELF, Binary Analysis tools

1.3 Latest release and defects report

Please check [AArch64 ELF Conventions for Binary Analysis](#) for the latest release of this document.

Please report defects in this specification to the [issue tracker page on GitHub](#).

1.4 Licence

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Grant of Patent License. Subject to the terms and conditions of this license (both the Public License and this Patent License), each Licensor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Licensed Material, where such license applies only to those patent claims licensable by such Licensor that are necessarily infringed by their contribution(s) alone or by combination of their contribution(s) with the Licensed Material to which such contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Licensed Material or a contribution incorporated within the Licensed Material constitutes direct or contributory patent infringement, then any licenses granted to You under this license for that Licensed Material shall terminate as of the date such litigation is filed.

1.5 About the license

As identified more fully in the [Licence](#) section, this project is licensed under CC-BY-SA-4.0 along with an additional patent license. The language in the additional patent license is largely identical to that in Apache-2.0 (specifically, Section 3 of Apache-2.0 as reflected at <https://www.apache.org/licenses/LICENSE-2.0>) with two exceptions.

First, several changes were made related to the defined terms so as to reflect the fact that such defined terms need to align with the terminology in CC-BY-SA-4.0 rather than Apache-2.0 (e.g., changing “Work” to “Licensed Material”).

Second, the defensive termination clause was changed such that the scope of defensive termination applies to “any licenses granted to You” (rather than “any patent licenses granted to You”). This change is intended to help maintain a healthy ecosystem by providing additional protection to the community against patent litigation claims.

1.6 Contributions

Contributions to this project are licensed under an inbound=outbound model such that any such contributions are licensed by the contributor under the same terms as those in the [Licence](#) section.

1.7 Trademark notice

The text of and illustrations in this document are licensed by Arm under a Creative Commons Attribution-Share Alike 4.0 International license (“CC-BY-SA-4.0”), with an additional clause on patents. The Arm trademarks featured here are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Please visit <https://www.arm.com/company/policies/trademarks> for more information about Arm’s trademarks.

1.8 Copyright

Copyright (c) 2026, Arm Limited and its affiliates. All rights reserved.

Contents

1 Preamble	2
1.1 Abstract	2
1.2 Keywords	2
1.3 Latest release and defects report	2
1.4 Licence	3
1.5 About the license	3
1.6 Contributions	3
1.7 Trademark notice	3
1.8 Copyright	3
2 About this document	5
2.1 Change Control	5
2.1.1 Current Status and Anticipated Changes	5
2.1.2 Change history	5
2.2 References	5
2.3 Terms and abbreviations	5
3 AArch64 Veneer Types Recognized by Binary Analysis Tools	7
3.1 Range Extension Veneers	7
3.1.1 Absolute Address Veneers	7
3.1.2 Position-independent Veneers	7
3.1.3 Execute-only Veneers	7
3.2 Short Range Veneers	8
3.3 BTI Landing Pad Veneers	9

2 About this document

2.1 Change Control

2.1.1 Current Status and Anticipated Changes

The following support level definitions are used by the Arm ABI specifications:

Release

Arm considers this specification to have enough implementations, which have received sufficient testing, to verify that it is correct. The details of these criteria are dependent on the scale and complexity of the change over previous versions: small, simple changes might only require one implementation, but more complex changes require multiple independent implementations, which have been rigorously tested for cross-compatibility. Arm anticipates that future changes to this specification will be limited to typographical corrections, clarifications and compatible extensions.

Beta

Arm considers this specification to be complete, but existing implementations do not meet the requirements for confidence in its release quality. Arm may need to make incompatible changes if issues emerge from its implementation.

Alpha

The content of this specification is a draft, and Arm considers the likelihood of future incompatible changes to be significant.

This document is at **Alpha** release quality.

2.1.2 Change history

If there is no entry in the change history table for a release, there are no changes to the content of the document for that release.

Issue	Date	Change
2025Q4	23 rd January 2026	Alpha draft release.

2.2 References

This document refers to the following documents.

Ref	URL or other reference Title	
AAELF64	IHI 0056	ELF for the Arm 64-bit Architecture
SYSVABI64	sysvabi64	System V Application Binary Interface (ABI) for the Arm 64-bit Architecture

2.3 Terms and abbreviations

AArch64 ELF Conventions for Binary Analysis document uses the following terms and abbreviations:

GOT

Global offset table. It holds addresses of symbols referenced from code, containing both link-time constants and entries updated by dynamic relocation.

.got.plt

The part of GOT containing symbol addresses referenced by Procedure Linkage Table (PLT) entries.
Other terms may be defined when first used.

3 AArch64 Veneer Types Recognized by Binary Analysis Tools

As described in [AAELF64](#), section Call and Jump relocations, the linker may insert veneers (also referred to as thunks, stubs, or trampolines) to implement call and jump relocations. This section defines the commonly used types and instruction sequences for such veneers on AArch64, enabling better recognition by binary analysis tools.

Toolchains are encouraged to follow these patterns to ensure veneers can be reliably recognized during binary analysis. These pattern sets are intended to remain open, and toolchains may introduce new veneer forms.

Additionally, if veneers emit static relocations, they become indistinguishable from regular code. In such cases, binary analysis tools should rely on relocation information to identify veneer targets.

3.1 Range Extension Veneers

3.1.1 Absolute Address Veneers

A 64-bit absolute target address is loaded from a literal pool into a register, and an indirect branch is used to transfer control to the target:

```
<caller>:
    B/BL __AArch64AbsLongThunk_<target>

__AArch64AbsLongThunk_<target>:
    LDR X16, =<target>
    BR  X16
```

Where <target> denotes the ultimate branch destination.

3.1.2 Position-independent Veneers

Position independent version uses an R_AARCH64_PREL64 relocation to reference the target address through literal pool:

```
<caller>:
    B/BL __<target>_veneer

__<target>_veneer:
    LDR X16, 1f
    ADD X17, #0
    ADD X16, X16, X17
    BR  X16
1:
    .xword R_AARCH64_PREL64(<target>) + 12
```

3.1.3 Execute-only Veneers

In pure code (execute-only) sections, an absolute 64-bit target address is constructed using move wide instructions, which combine four 16-bit immediate fields:

```

<caller>:
    B/BL __AArch64AbsX0LongThunk_<target>

__AArch64AbsX0LongThunk_<target>:
    MOVZ X16, #:abs_g0_nc:<target>
    MOVK X16, #:abs_g1_nc:<target>, LSL #16
    MOVK X16, #:abs_g2_nc:<target>, LSL #32
    MOVK X16, #:abs_g3:<target>,    LSL #48
    BR   X16

```

Note that some of the MOVK instructions may be omitted if their corresponding 16-bit segments of the address are zero and do not need to be explicitly set.

The following instruction sequence may be used to construct 64-bit PC-relative address:

```

<caller>:
    B/BL $thunk<decimal suffix>

$thunk<decimal suffix>:
    ADR  X16, #:prel_g0_nc:<target>
    MOVK X17, #:prel_g1_nc:<target>, LSL #16
    MOVK X17, #:prel_g2_nc:<target>, LSL #32
    MOVK X17, #:prel_g3:<target>,    LSL #48
    ADD  X16, X16, X17
    BR   X16

```

3.2 Short Range Veneers

When the target is within range of a direct branch, the linker may generate a simple short range branch veneer using the same name as a range extension veneer. The optional infix X0 identifies execute-only variants:

```

__AArch64Abs[X0]LongThunk_<target>:
    B <target>

```

PC-relative veneer is generated using ADRP and ADD:

```

<caller>:
    B/BL __AArch64ADRPThunk_<target>

__AArch64ADRPThunk_<target>:
    ADRP X16, <target>
    ADD  X16, X16, :lo12:<target>
    BR   X16

```

It is recognized by alternative names: __<target>_veneer, \$thunk<decimal suffix>, <hexadecimal prefix>-tramp<decimal suffix>

When the .got.plt entry for the <target> is known to exist an direct load from GOT may be used:

```

<caller>:
    B/BL <hexadecimal prefix>-tramp<decimal suffix>

<hexadecimal prefix>-tramp<decimal suffix>:
    ADRP X16, :got:<target>
    LDR  X16, [X16, :got_lo12:<target>]
    BR   X16

```

3.3 BTI Landing Pad Veneers

When Branch Target Identification (BTI) is enabled, any of the veneers above may route through an additional Landing Pad veneer `__AArch64BTIThunk_<target>`. The rules governing generation of these veneers are specified in [SYSVABI64](#), section Tool Requirements for generating BTI instructions. A Landing Pad veneer begins with a BTI instruction to ensure a valid landing and then branches to the target:

```
__AArch64BTIThunk_<target>:  
    BTI C  
    B <target>
```

In cases where the veneer is placed immediately before the target, the B instruction may be omitted:

```
__AArch64BTIThunk_<target>:  
    BTI C  
<target>:
```

Recognized by alternative names: `__<target>_bti_veneer`