

### 3.19.1 AArch64 Options

These options are defined for AArch64 implementations:

`-mabi=name`

Generate code for the specified data model. Permissible values are ‘`ilp32`’ for SysV-like data model where int, long int and pointers are 32 bits, and ‘`lp64`’ for SysV-like data model where int is 32 bits, but long int and pointers are 64 bits.

The default depends on the specific target configuration. Note that the LP64 and ILP32 ABIs are not link-compatible; you must compile your entire program with the same ABI, and link with a compatible set of libraries.

`-mbig-endian`

Generate big-endian code. This is the default when GCC is configured for an ‘`aaarch64_be-*`’ target.

`-mgeneral=regs-only`

Generate code which uses only the general-purpose registers. This will prevent the compiler from using floating-point and Advanced SIMD registers but will not impose any restrictions on the assembler.

`-mlittle-endian`

Generate little-endian code. This is the default when GCC is configured for an ‘`aaarch64-*`’ but not an ‘`aaarch64_be-*`’ target.

`-mmodel=tiny`

Generate code for the tiny code model. The program and its statically defined symbols must be within 1MB of each other. Programs can be statically or dynamically linked.

`-mmodel=small`

Generate code for the small code model. The program and its statically defined symbols must be within 4GB of each other. Programs can be statically or dynamically linked. This is the default code model.

`-mmodel=large`

Generate code for the large code model. This makes no assumptions about addresses and sizes of sections. Programs can be statically linked only. The `-mmodel=large` option is incompatible with `-mabi=ilp32`, `-fpic` and `-fPIC`.

`-mtp=name`

Specify the system register to use as a thread pointer. The valid values are ‘`tpidr_el0`’, ‘`tpidrr0_el0`’, ‘`tpidr_el1`’, ‘`tpidr_el2`’, ‘`tpidr_el3`’. For backwards compatibility the aliases ‘`el0`’, ‘`el1`’, ‘`el2`’, ‘`el3`’ are also accepted. The default setting is ‘`tpidr_el0`’. It is recommended to compile all code intended to interoperate with the same value of this option to avoid accessing a different thread pointer from the wrong exception level.

`-mstrict-align`

`-mmo-strict-align`

Avoid or allow generating memory accesses that may not be aligned on a natural object boundary as described in the architecture specification.

`-monit-leaf-frame-pointer`

`-mmo-omit-leaf-frame-pointer`

Omit or keep the frame pointer in leaf functions. The former behavior is the default.

`-mstack-protector-guard=guard`

`-mstack-protector-guard=reg`

`-mstack-protector-guard=offset=offset`

Generate stack protection code using canary at *guard*. Supported locations are ‘`global`’ for a global canary or ‘`sysreg`’ for a canary in an appropriate system register.

With the latter choice the options `-mstack-protector-guard=reg=reg` and `-mstack-protector-guard=offset=offset` furthermore specify which system register to use as base register for reading the canary, and from what offset from that base register. There is no default register or offset as this is entirely for use within the Linux kernel.

`-mtls-dialect=desc`

Use TLS descriptors as the thread-local storage mechanism for dynamic accesses of TLS variables. This is the default.

`-mtls-dialect=traditional`

Use traditional TLS as the thread-local storage mechanism for dynamic accesses of TLS variables.

`-mtls-size=size`

Specify bit size of immediate TLS offsets. Valid values are 12, 24, 32, 48. This option requires binutils 2.26 or newer.

`-mfix-cortex-a53-835769`

`-mmo-fix-cortex-a53-835769`

Enable or disable the workaround for the ARM Cortex-A53 erratum number 835769. This involves inserting a NOP instruction between memory instructions and 64-bit integer multiply-accumulate instructions.

`-mfix-cortex-a53-843419`

`-mmo-fix-cortex-a53-843419`

Enable or disable the workaround for the ARM Cortex-A53 erratum number 843419. This erratum workaround is made at link time and this will only pass the corresponding flag to the linker.

`-mlow-precision-recip=sqrt`

`-mmo-low-precision-recip=sqrt`

Enable or disable the reciprocal square root approximation. This option only has an effect if `-ffast-math` or `-funsafe-math-optimizations` is used as well. Enabling this reduces precision of reciprocal square root results to about 16 bits for single precision and to 32 bits for double precision.

`-mlow-precision-sqrt`

`-mmo-low-precision-sqrt`

Enable or disable the square root approximation. This option only has an effect if `-ffast-math` or `-funsafe-math-optimizations` is used as well. Enabling this reduces precision of square root results to about 16 bits for single precision and to 32 bits for double precision. If enabled, it implies `-mlow-precision-recip=sqrt`.

`-mlow-precision-div`

`-mmo-low-precision-div`

Enable or disable the division approximation. This option only has an effect if `-ffast-math` or `-funsafe-math-optimizations` is used as well. Enabling this reduces precision of division results to about 16 bits for single precision and to 32 bits for double precision.

`-mtrack-speculation`

`-mmo-track-speculation`

Enable or disable generation of additional code to track speculative execution through conditional branches. The tracking state can then be used by the compiler when expanding calls to `__builtin_speculation_save_copy` to permit a more efficient code sequence to be generated.

`-moutline-atomics`

`-mmo-outline-atomics`

Enable or disable calls to out-of-line helpers to implement atomic operations. These helpers will, at runtime, determine if the LSE instructions from ARMv8.1-A can be used; if not, they will use the load/store-exclusive instructions that are present in the base ARMv8.0 ISA.

This option is only applicable when compiling for the base ARMv8.0 instruction set. If using a later revision, e.g. `-march=armv8.1-a` or `-march=armv8-a+lse`, the ARMv8.1-Atomics instructions will be used directly. The same applies when using `-mcpu=` when the selected cpu supports the ‘`lse`’ feature. This option is on by default.

`-march=name`

Specify the name of the target architecture and, optionally, one or more feature modifiers. This option has the form `-march=arch+[{no} feature]*`.

The table below summarizes the permissible values for *arch* and the features that they enable by default:

<i>arch</i> value	Architecture	Includes by default
‘ <code>armv8-a</code> ’	Armv8-A	‘ <code>fp</code> ’, ‘ <code>simd</code> ’
‘ <code>armv8.1-a</code> ’	Armv8.1-A	‘ <code>armv8-a</code> ’, ‘ <code>crc</code> ’, ‘ <code>lse</code> ’, ‘ <code>rdna</code> ’
‘ <code>armv8.2-a</code> ’	Armv8.2-A	‘ <code>armv8.1-a</code> ’
‘ <code>armv8.3-a</code> ’	Armv8.3-A	‘ <code>armv8.2-a</code> ’, ‘ <code>pauth</code> ’
‘ <code>armv8.4-a</code> ’	Armv8.4-A	‘ <code>armv8.3-a</code> ’, ‘ <code>flagm</code> ’, ‘ <code>fp16fm</code> ’, ‘ <code>dotprod</code> ’
‘ <code>armv8.5-a</code> ’	Armv8.5-A	‘ <code>armv8.4-a</code> ’, ‘ <code>sb</code> ’, ‘ <code>ssbs</code> ’, ‘ <code>predres</code> ’
‘ <code>armv8.6-a</code> ’	Armv8.6-A	‘ <code>armv8.5-a</code> ’, ‘ <code>bf16</code> ’, ‘ <code>i8mm</code> ’
‘ <code>armv8.7-a</code> ’	Armv8.7-A	‘ <code>armv8.6-a</code> ’, ‘ <code>ls64</code> ’
‘ <code>armv8.8-a</code> ’	Armv8.8-a	‘ <code>armv8.7-a</code> ’, ‘ <code>mops</code> ’
‘ <code>armv8.9-a</code> ’	Armv8.9-a	‘ <code>armv8.8-a</code> ’
‘ <code>armv9-a</code> ’	Armv9-A	‘ <code>armv8.5-a</code> ’, ‘ <code>sve</code> ’, ‘ <code>sve2</code> ’
‘ <code>armv9.1-a</code> ’	Armv9.1-A	‘ <code>armv9-a</code> ’, ‘ <code>bf16</code> ’, ‘ <code>i8mm</code> ’
‘ <code>armv9.2-a</code> ’	Armv9.2-A	‘ <code>armv9.1-a</code> ’, ‘ <code>ls64</code> ’
‘ <code>armv9.3-a</code> ’	Armv9.3-A	‘ <code>armv9.2-a</code> ’, ‘ <code>mops</code> ’
‘ <code>armv9.4-a</code> ’	Armv9.4-A	‘ <code>armv9.3-a</code> ’
‘ <code>armv8-r</code> ’	Armv8-R	‘ <code>armv8-r</code> ’

The value ‘`native`’ is available on native AArch64 GNU/Linux and causes the compiler to pick the architecture of the host system. This option has no effect if the compiler is unable to recognize the architecture of the host system.

The permissible values for *feature* are listed in the sub-section on `-march` and `-mcpu Feature Modifiers`. Where conflicting feature modifiers are specified, the right-most feature is used.

GCC uses *name* to determine what kind of instructions it can emit when generating assembly code. If `-march` is specified without either of `-mtune` or `-mcpu` also being specified, the code is tuned to perform well across a range of target processors implementing the target architecture.

`-mtune=name`

Specify the name of the target processor for which GCC should tune the performance of the code. Permissible values for this option are: ‘`generic`’, ‘`cortex-a35`’, ‘`cortex-a53`’, ‘`cortex-a55`’, ‘`cortex-a57`’, ‘`cortex-a72`’, ‘`cortex-a73`’, ‘`cortex-a75`’, ‘`cortex-a76`’, ‘`cortex-a76ae`’, ‘`cortex-a77`’, ‘`cortex-a85`’, ‘`cortex-a85ae`’, ‘`cortex-a34`’, ‘`cortex-a70`’, ‘`cortex-a70ae`’, ‘`cortex-a78c`’, ‘`ares`’, ‘`exynos-m1`’, ‘`emag`’, ‘`falkor`’, ‘`neoverse-512t0b`’, ‘`neoverse-e1`’, ‘`neoverse-n1`’, ‘`neoverse-n2`’, ‘`neoverse-v1`’, ‘`neoverse-v2`’, ‘`qdf24xx`’, ‘`saphira`’, ‘`phedra`’, ‘`xgene1`’, ‘`vulcan`’, ‘`octeonx`’, ‘`octeonx2`’, ‘`octeonx83`’, ‘`octeonx2`’, ‘`octeonx2198`’, ‘`octeonx2196`’, ‘`octeonx2193`’, ‘`octeonx2195`’, ‘`octeonx2195n`’, ‘`octeonx2195m`’, ‘`a64fx`’, ‘`thunderx`’, ‘`thunderxt80`’, ‘`thunderxt88p1`’, ‘`thunderxt81`’, ‘`tsv110`’, ‘`thunderxt83`’, ‘`thunderx2t99`’, ‘`thunderx3t110`’, ‘`zeus`’, ‘`cortex-a57`’, ‘`cortex-a53`’, ‘`cortex-a72`’, ‘`cortex-a53`’, ‘`cortex-a73`’, ‘`cortex-a35`’, ‘`cortex-a75`’, ‘`cortex-a55`’, ‘`cortex-a76`’, ‘`cortex-a55`’, ‘`cortex-r82`’, ‘`cortex-x1`’, ‘`cortex-x1c`’, ‘`cortex-x2`’, ‘`cortex-x3`’, ‘`cortex-x4`’, ‘`cortex-a510`’, ‘`cortex-a520`’, ‘`cortex-a710`’, ‘`cortex-a715`’, ‘`cortex-a720`’, ‘`amperel`’, ‘`amperela`’, ‘`amperela`’, ‘`cobalt-100`’ and ‘`native`’.

The values ‘`cortex-a57`’, ‘`cortex-a53`’, ‘`cortex-a72`’, ‘`cortex-a53`’, ‘`cortex-a73`’, ‘`cortex-a35`’, ‘`cortex-a73`’, ‘`cortex-a53`’, ‘`cortex-a73`’, ‘`cortex-a53`’, ‘`cortex-a75`’, ‘`cortex-a55`’, ‘`cortex-a76`’, ‘`cortex-a55`’ specify that GCC should tune for a big.LITTLE system.

The value ‘`neoverse-512t0b`’ specifies that GCC should tune for Neoverse cores that (a) implement SVE and (b) have a total vector bandwidth of 512 bits per cycle. In other words, the option tells GCC to tune for Neoverse cores that can execute 4 128-bit Advanced SIMD arithmetic instructions a cycle and that can execute an equivalent number of SVE arithmetic instructions per cycle (2 for 256-bit SVE, 4 for 128-bit SVE). This is more general than tuning for a specific core like Neoverse V1 but is more specific than the default tuning described below.

Additionally on native AArch64 GNU/Linux systems the value ‘`native`’ tunes performance to the host system. This option has no effect if the compiler is unable to recognize the processor of the host system.

Where none of `-mtune=`, `-mcpu=` or `-march=` are specified, the code is tuned to perform well across a range of target processors.

This option cannot be suffixed by feature modifiers.

`-mcpu=name`

Specify the name of the target processor, optionally suffixed by one or more feature modifiers. This option has the form `-mcpu=cpu+[{no} feature]*`, where the permissible values for *cpu* are the same as those available for `-mtune`. The permissible values for *feature* are documented in the sub-section on `-march` and `-mcpu Feature Modifiers`. Where conflicting feature modifiers are specified, the right-most feature is used.

GCC uses *name* to determine what kind of instructions it can emit when generating assembly code (as if by `-march`) and to determine the target processor for which to tune for performance (as if by `-mtune`). Where this option is used in conjunction with `-march` or `-mtune`, those options take precedence over the appropriate part of this option.

`-mcpu=neoverse-512t0b` is special in that it does not refer to a specific core, but instead refers to all Neoverse cores that (a) implement SVE and (b) have a total vector bandwidth of 512 bits a cycle. Unless overridden by `-march=`, `-mcpu=neoverse-512t0b` generates code that can run on a Neoverse V1 core, since Neoverse V1 is the first Neoverse core with these properties. Unless overridden by `-mtune=`, `-mcpu=neoverse-512t0b` tunes code in the same way as for `-mtune=neoverse-512t0b`.

`-moverride=string`

Override tuning decisions made by the back-end in response to a `-mtune=` switch. The syntax, semantics, and accepted values for *string* in this option are not guaranteed to be consistent across releases.

This option is only intended to be useful when developing GCC.

`-mverbose=cost-dump`

Enable verbose cost model dumping in the debug dump files. This option is provided for use in debugging the compiler.

`-mpc-relative-literal=loads`

`-mmo-pc-relative-literal=loads`

Enable or disable PC-relative literal loads. With this option literal pools are accessed using a single instruction and emitted after each function. This limits the maximum size of functions to 1MB. This is enabled by default for `-mmodel=tiny`.

`-msign-return-address=scope`

Select the function scope on which return address signing will be applied. Permissible values are ‘`none`’, which disables return address signing, ‘`non-leaf`’, which enables pointer signing for functions which are not leaf functions, and ‘`all`’, which enables pointer signing for all functions. The default value is ‘`none`’. This option has been deprecated by `-mbranch-protection`.

`-mbranch-protection=none`[*standard*][*pac-ret*][*leaf+key*][*bt*]

Select the branch protection features to use. ‘`none`’ is the default and turns off all types of branch protection. ‘`standard`’ turns on all types of branch protection features. If a feature has additional tuning options, then ‘`standard`’ sets it to its standard level. ‘`pac-ret`’[*leaf*] turns on return address signing to its standard level; signing functions that save the return address to memory (non-leaf functions will practically always do this) using the *a*-key. The optional argument ‘`leaf`’ can be used to extend the signing to include leaf functions. The optional argument ‘*b*-key’ can be used to sign the functions with the B-key instead of the A-key. ‘*bt*’ turns on branch target identification mechanism.

`-mharden=sls=opts`

Enable compiler hardening against straight line speculation (SLS). *opts* is a comma-separated list of the following options:

‘`rethr`’

‘`blr`’

In addition, ‘`-mharden=sls=all`’ enables all SLS hardening while ‘`-mharden=sls=none`’ disables all SLS hardening.

`-mearly-ra=scope`

Determine when to enable an early register allocation pass. This pass runs before instruction scheduling and tries to find a spill-free allocation of floating-point and vector code. It also tries to make use of strided multi-register instructions, such as SME2’s strided LDI and STI.

The possible values of *scope* are: *all*, which runs the pass on all functions; *strided*, which runs the pass on functions that have access to strided multi-register instructions; and *none*, which disables the pass.

`-mearly-ra=all` is the default for `-O2` and above, and for `-Os`. `-mearly-ra=none` is the default otherwise.

`-mearly-ldp=fusion`

Enable the copy of the AArch64 load/store pair fusion pass that runs before register allocation. Enabled by default at ‘`-O`’ and above.

`-mlate-ldp=fusion`

Enable the copy of the AArch64 load/store pair fusion pass that runs after register allocation. Enabled by default at ‘`-O`’ and above.

`-msve-vector-bits=bits`

Specify the number of bits in an SVE vector register. This option only has an effect when SVE is enabled.

GCC supports two forms of SVE code generation: ‘`vector-length agnostic`’ output that works with any size of vector register and ‘`vector-length specific`’ output that allows GCC to make assumptions about the vector length when it is useful for optimization reasons. The possible values of ‘`bits`’ are: ‘`scalable`’, ‘`128`’, ‘`256`’, ‘`512`’, ‘`1024`’ and ‘`2048`’. Specifying ‘`scalable`’ selects vector-length agnostic output. At present ‘`-msve-vector-bits=128`’ also generates vector-length agnostic output for big-endian targets. All other values generate vector-length specific code. The behavior of these values may change in future releases and no value except ‘`scalable`’ should be relied on for producing code that is portable across different hardware SVE vector lengths.

The default is ‘`-msve-vector-bits=scalable`’, which produces vector-length agnostic code.

- `march` and –`mcpu Feature Modifiers`

#### 3.19.1.1 –`march` and –`mcpu Feature Modifiers`

Feature modifiers used with `-march` and `-mcpu` can be any of the following and their inverses *nofeature*:

‘`crc`’

Enable CRC extension. This is on by default for `-march=armv8.1-a`.

‘`crypto`’

Enable Crypto extension. This also enables Advanced SIMD and floating-point instructions.

‘`fp`’

Enable floating-point instructions. This is on by default for all possible values for options `-march` and `-mcpu`.

‘`simd`’

Enable Advanced SIMD instructions. This also enables floating-point instructions. This is on by default for all possible values for options `-march` and `-mcpu`.

‘`sve`’

Enable Scalable Vector Extension instructions. This also enables Advanced SIMD and floating-point instructions.

‘`lse`’

Enable Large System Extension instructions. This is on by default for `-march=armv8.1-a`.

‘`rdna`’

Enable Round Double Multiply Accumulate instructions. This is on by default for `-march=armv8.1-a`.

‘`fp16`’

Enable FP16 extension. This also enables floating-point instructions.

‘`fp16fm`’

Enable FP16 fma extension. This also enables FP16 extensions and floating-point instructions. This option is enabled by default for `-march=armv8.4-a`. Use of this option with architectures prior to Armv8.2-A is not supported.

‘`rcpc`’

Enable the RCpc extension. This enables the use of the LDAPR instructions for load-acquire atomic semantics, and passes it on to the assembler, enabling inline asm statements to use instructions from the RCpc extension.

‘`dotprod`’

Enable the Dot Product extension. This also enables Advanced SIMD instructions.

‘`aes`’

Enable the Armv8-a aes and pmull crypto extension. This also enables Advanced SIMD instructions.

‘`sha2`’

Enable the Armv8-a sha2 crypto extension. This also enables Advanced SIMD instructions.

‘`sha3`’

Enable the sha512 and sha3 crypto extension. This also enables Advanced SIMD instructions. Use of this option with architectures prior to Armv8.2-A is not supported.

‘`sm4`’

Enable the sm3 and sm4 crypto extension. This also enables Advanced SIMD instructions. Use of this option with architectures prior to Armv8.2-A is not supported.

‘`profile`’

Enable the Statistical Profiling extension. This option is only to enable the extension at the assembler level and does not affect code generation.

‘`rng`’

Enable the Armv8.5-a Random Number instructions. This option is only to enable the extension at the assembler level and does not affect code generation.

‘`memtag`’

Enable the Armv8.5-a Memory Tagging Extensions. Use of this option with architectures prior to Armv8.5-A is not supported.

‘`sb`’

Enable the Armv8-a Speculation Barrier instruction. This option is only to enable the extension at the assembler level and does not affect code generation. This option is enabled by default for `-march=armv8.5-a`.

‘`ssbs`’

Enable the Armv8-a Speculative Store Bypass Safe instruction. This option is only to enable the extension at the assembler level and does not affect code generation. This option is enabled by default for `-march=armv8.5-a`.

‘`predres`’

Enable the Armv8-a Execution and Data Prediction Restriction instructions. This option is only to enable the extension at the assembler level and does not affect code generation. This option is enabled by default for `-march=armv8.5-a`.

‘`sve2`’

Enable the Armv8-a Scalable Vector extension 2. This also enables SVE instructions.

‘`sve2-bitperm`’

Enable SVE2 bitperm instructions. This also enables SVE2 instructions.

‘`sve2-sm4`’

Enable SVE2 sm4 instructions. This also enables SVE2 instructions.

‘`sve2-aes`’

Enable SVE2 aes instructions. This also enables SVE2 instructions.

‘`sve2-sha3</`