AMD ROCm™ Compiler Reference Guide

Publication # **1.0** Revision: **0802**
Issue Date: **August 2021**

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. In addition, any stated support is planned and is also subject to change. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

[This page left blank intentionally]

Table of Contents

Chapter 1 Overview of ROCmCC Compiler

The ROCmCC Compiler is based on the Clang/LLVM compiler technology. It is optimized for high-performance computing on AMD GPUs and CPUs and supports various heterogenous programming models such as HIP, OpenMP, and OpenCL.

Chapter 2 Command Options

The CPU compiler optimizations described in this chapter originate from the AMD AOCC compiler. They are available in ROCmCC after the llvm-amdgpu-alt package installation. AMD provides high performance compiler optimizations for Zen-based processors in *AMD Optimizing C/C++ Compiler* (AOCC).

For more information, refer to

https://developer.amd.com/amd-aocc/

2.1 Target Selection

Option	Description
`-march=<cpu>`	Use it to specify if Clang must generate code for a specific processor family member and later. For example, if you specify `-march=i486`, the compiler can generate instructions that are valid on i486 and later processors, but which may not exist on the earlier ones.
`-march=znver1`	Use this architecture flag for enabling the best code generation and tuning for AMD Zen based x86 architecture. All the x86 Zen ISA and associated intrinsic are supported.
`-march=znver2`	Use this architecture flag for enabling the best code generation and tuning for AMD Zen2 based x86 architecture. All x86 Zen2 ISA and associated intrinsic are supported.
`-march=znver3`	Use this architecture flag for enabling best code generation and tuning for AMD Zen3 based x86 architecture. All x86 Zen3 ISA and associated intrinsic are supported.

2.2 Code Generation

Use the following options to specify the optimization level:

Level	Description
`-O0`	It means no optimization: this level compiles the fastest and generates the most debuggable code.
`-O1`	Between levels `-O0` and `-O2`.
`-O2`	A moderate level of optimization, which enables most optimizations.
`-O3`	It is similar to the level `-O2`, except that it enables the optimizations, which take longer to perform or may generate larger code (in an attempt to make the program run faster). This `-O3` level has more optimizations when compared to the base LLVM version on which it is based. These optimizations include improved handling of indirect calls, advanced vectorization, and so on.
`-Ofast`	It enables all the optimizations from `-O3` along with other aggressive optimizations that may violate strict compliance with language standards. This `-Ofast` level has more optimizations when compared to the base LLVM version on which it is based. These optimizations include partial unswitching, improvements to inlining, unrolling, and so on.
`-Os`	It is similar to the level `-O2`, but with extra optimizations to reduce the code size.
`-Oz`	It is similar to the level `-Os` (and thus, `-O2`), but reduces the code size further.
`-O`	It is equivalent to the level `O2`.
`-O4 and higher`	It is equivalent to the level `O3`.

2.3 AMD Optimizations for AMD Zen Architectures

2.3.1 -famd-opt

Enables a default set of AMD proprietary optimizations for the AMD Zen CPU architectures.

`-fno-amd-opt` disables the AMD proprietary optimizations.

The `-famd-opt` flag is useful when a user wants to build with the proprietary optimization compiler and not have to depend on setting any of the other proprietary optimization flags. Note, `-famd-opt` can be used in addition to other proprietary CPU optimization flags.

The table of optimizations below will implicitly enable the invocation of the AMD proprietary optimizations compiler. Whereas the `-famd-opt` flag explicitly requires such.

2.3.2 -fstruct-layout=[1,2,3,4,5,6,7]

Analyzes the whole program to determine if the structures in the code can be peeled and if the pointer or integer fields in the structure can be compressed. If feasible, this optimization transforms the code to enable these improvements. This transformation is likely to improve cache utilization and memory bandwidth. It is expected to improve the scalability of programs executed on multiple cores.

This is effective only under `flto` as the whole program analysis is required to perform this optimization. You can choose different levels of aggressiveness with which this optimization can be applied to your application; with 1 being the least aggressive and 7 being the most aggressive level.

- **`fstruct-layout=1`** enables structure peeling.
- **`fstruct-layout=2`** enables structure peeling and selectively compresses self-referential pointers in these structures to 32-bit pointers, wherever safe.
- **`fstruct-layout=3`** enables structure peeling and selectively compresses self-referential pointers in these structures to 16-bit pointers, wherever safe.
- **`fstruct-layout=4`** enables structure peeling, pointer compression as in level 2 and further enables compression of structure fields, which are of integer type. This is performed under a strict safety check.
- **`fstruct-layout=5`** enables structure peeling, pointer compression as in level 3 and further enables compression of structure fields which are of integer type. This is performed under a strict safety check.
- **`fstruct-layout=6`** enables structure peeling, pointer compression as in level 2 and further enables compression of structure fields, which are of type 64-bit **signed int** or **unsigned int**. You must ensure that the values assigned to 64-bit **signed int** fields are in range -(2^31 - 1) to +(2^31 - 1) and 64-bit **unsigned int** fields are in the range 0 to +(2^31 - 1). Else, incorrect results may be obtained. This

compression is performed without considering any safety analysis. So, you must ensure the safety based on the program compiled.

- `fstruct-layout=7` enables structure peeling, pointer compression as in level 3 and further enables compression of structure fields, which are of type 64-bit signed int or unsigned int. You must ensure that the values assigned to 64-bit signed int fields are in range -(2^31 - 1) to +(2^31 - 1) and 64-bit unsigned int fields are in the range 0 to +(2^31 - 1). Else, incorrect results may be obtained. This compression is performed without considering any safety analysis. So, must ensure the safety based on the program compiled.

NOTE:

`fstruct-layout=4` and `fstruct-layout=5` are derived from `fstruct-layout=2` and `fstruct-layout=3` respectively, with the added feature of safe compression of integer fields in structures. Going from `fstruct-layout=4` to `fstruct-layout=5` may result in higher performance if the pointer values are such that the pointers can be compressed to 16-bits.

`fstruct-layout=6` and `fstruct-layout=7` are derived from `fstruct-layout=2` and `fstruct-layout=3` respectively, with the added feature of compression of the integer fields in structures. These are similar to `fstruct-layout=4` and `fstruct-layout=5`, but here, the integer fields of the structures are always compressed from 64-bits to 32-bits, without any safety guarantee.

2.3.3 -fitodcalls

It promotes indirect to direct calls by placing conditional calls. Application or benchmarks that have small and deterministic set of target functions for function pointers that are passed as call parameters benefit from this optimization. Indirect-to-direct call promotion transforms the code to use all possible determined targets under runtime checks and falls back to the original code for all the other cases. Runtime checks are introduced by the compiler for each of these possible function pointer targets followed by direct calls to the targets.

This is a link time optimization, which is invoked as `-flto -fitodcalls.`

2.3.4 -fitodcallsbyclone

Performs value specialization for functions with function pointers passed as an argument. It does this specialization by generating a clone of the function. The cloning of the function happens in the call chain as needed to allow conversion of indirect function call to direct call. This complements `-fitodcalls` optimization and is also a link time optimization, which is invoked as `-flto -fitodcallsbyclone`.

2.3.5 -fremap-arrays

Transforms the data layout of a single dimensional array to provide better cache locality. This optimization is effective only under *flto* as the whole program analysis is required to perform this optimization, which can be invoked as `-flto -fremap-arrays`.

2.3.6 -finline-aggressive

Enables improved inlining capability through better heuristics. This optimization is more effective when using with *flto* as the whole program analysis is required to perform this optimization, which can be invoked as `-flto -finline-aggressive`.

2.3.7 -fnt-store (non-temporal store)

Generates a non-temporal store instruction for array accesses in a loop with a large trip count.

2.3.8 -fnt-store=aggressive

This is an experimental option to generate non-temporal store instruction for array accesses in a loop, whose iteration count cannot be determined at compile time. In this case, compiler assumes the iteration count is huge.

2.4 Optimizations Through Driver -mllvm <options>

The following optimization options must be invoked through driver `-mllvm <options>`:

2.4.1 -enable-partial-unswitch

Enables partial loop un-switching, which is an enhancement to the existing loop unswitching optimization in LLVM. Partial loop un-switching hoists a condition inside a loop from a path for which the execution condition remains invariant, whereas the original loop un-switching works for a condition that is completely loop invariant. The condition inside the loop gets hoisted out from the invariant path and original loop is retained for the path where condition is variant.

2.4.2 -aggressive-loop-unswitch

Experimental option which enables aggressive loop unswitching heuristic (including `-enable-partial-unswitch`) based on the usage of the branch conditional values. Loop unswitching leads to code-bloat. Code-bloat can be minimized if the hoisted condition is executed more often. This heuristic prioritizes the conditions based on the number of times they are used within the loop. The heuristic can be controlled with the following options:

- `-unswitch-identical-branches-min-count=<n>`

Enables unswitching of a loop with respect to a branch conditional value (B), where B appears in at least <n> compares in the loop. This option is enabled with -aggressive-loop-unswitch. The default value is 3.

Usage: -mllvm -aggressive-loop-unswitch -mllvm -unswitch-identical-branches-min-count=<n>
Where, n is a positive integer and lower value of <n> facilitates more unswitching.

- -unswitch-identical-branches-max-count=<n>
  Enables unswitching of a loop with respect to a branch conditional value (B), where B appears in at most <n> compares in the loop. This option is enabled with -aggressive-loop-unswitch. The default value is 6.

  Usage: -mllvm -aggressive-loop-unswitch -mllvm -unswitch-identical-branches-max-count=<n>
  Where, n is a positive integer and higher value of <n> facilitates more unswitching.

NOTE: These options may facilitate more unswitching in some of the workloads. Since, loop-unswitching inherently leads to code-bloat, facilitating more unswitching may significantly increase the code size. Hence, it may also lead to longer compilation times.

2.4.3 -enable-strided-vectorization

Enables strided memory vectorization as an enhancement to the interleaved vectorization framework present in LLVM. It enables the effective use of gather and scatter kind of instruction patterns. This flag must be used along with the interleave vectorization flag.

2.4.4 -enable-epilog-vectorization

Enables vectorization of epilog-iterations as an enhancement to existing vectorization framework. This enables generation of an additional epilog vector loop version for the remainder iterations of the original vector loop. The vector size or factor of the original loop should be large enough to allow an effective epilog vectorization of the remaining iterations. This optimization takes place only when the original vector loop is vectorized with a vector width or factor of sixteen. This vectorization width of sixteen may be overwritten by -min-width-epilog-vectorization command line option.

2.4.5 -enable-redundant-movs

Removes any redundant mov operations including redundant loads from memory and stores to memory. This can be invoked using -Wl,-plugin-opt=-enable-redundant-movs.

2.4.6 -merge-constant

Attempts to promote frequently occurring constants to registers. The aim is to reduce the size of the instruction encoding for instructions using constants and obtain a performance improvement.

2.4.7 -function-specialize

Optimizes the functions with compile time constant formal arguments.

2.4.8 -lv-function-specialization

Generates specialized function versions when the loops inside function are vectorizable and the arguments are not aliased with each other.

2.4.9 -enable-vectorize-compares

Enables vectorization on certain loops with conditional breaks assuming the memory access are safely bound within the page boundary.

2.4.10 -inline-recursion=[1,2,3,4]

Enables inlining for recursive functions based on heuristics with level 4 being most aggressive. The default level will be 2. Higher levels may lead to code-bloat due to expansion of recursive functions at call sites.

- For level 1-2: Enables inlining for recursive functions using heuristics with inline depth 1. Level 2 uses more aggressive heuristics.
- For level 3: Enables inlining for all recursive functions with inline depth 1.
- For level 4: Enables inlining for all recursive function with inline depth 10.

This is more effective with flto as the whole program analysis is required to perform this optimization, which can be invoked as `-flto -inline-recursion=[1,2,3,4]`.

2.4.11 -reduce-array-computations=[1,2,3]

Performs array dataflow analysis and optimizes the unused array computations.

- `reduce-array-computations=1`: Eliminates the computations on unused array elements.
- `reduce-array-computations=2`: Eliminates the computations on zero valued array elements.
- `reduce-array-computations=3`: Eliminates the computations on unused and zero valued array elements (combination of 1 and 2).

This optimization is effective with flto as the whole program analysis is required to perform this optimization, which can be invoked as `-flto -reduce-array-computations=[1,2,3]`.

2.4.12 -global-vectorize-slp={true,false}

Vectorizes the straight-line code inside a basic block with data reordering vector operations. This option is set to **true** by default.

2.4.13 -region-vectorize

Experimental flag for enabling vectorization on certain loops with complex control flow which the normal vectorizer cannot handle.

This optimization is effective with *flto* as the whole program analysis is required to perform this optimization, which can be invoked as `-flto -region-vectorize`.

2.4.14 -enable-X86-prefetching

Enables the generation of x86 prefetch instruction for the memory references inside a loop/ inside an inner most loop of a loop nest to prefetch the second dimension of multidimensional array/memory references in the inner most of a loop nest. This is an experimental pass; its profitability is being improved.

2.4.15 -suppress-fmas

Identifies the reduction patterns on FMA and suppresses the FMA generation as it is not profitable on the reduction patterns.

2.4.16 -enable-licm-vrp

Enables estimation of the virtual register pressure before performing loop invariant code motion. This estimation is used to control the number of loop invariants that will be hoisted during the loop invariant code motion.

2.4.17 -loop-splitting

Enables splitting of loops into multiple loops to eliminate the branches, which compare the loop induction with an invariant or constant expression. This option is enabled under `-O3` by default. To disable this optimization, use `-loop-splitting=false`.

2.4.18 -enable-ipo-loop-split

Enables splitting of loops into multiple loops to eliminate the branches, which compares the loop induction with a constant expression. This constant expression can be derived through inter-

procedural analysis. This option is enabled under `-O3` by default. To disable this optimization, use `-enable-ipo-loop-split=false`.

2.4.19 -compute-interchange-order

Enables heuristic for finding the best possible interchange order for a loop nest. To enable this option, use `-enable-loopinterchange`. This option is set to **false** by default.

Usage: `-mllvm -enable-loopinterchange -mllvm -compute-interchange-order`

2.4.20 -convert-pow-exp-to-int={true,false}

Converts the call to floating point exponent version of `pow` to its integer exponent version if the floating-point exponent can be converted to integer. This option is set to **true** by default.

2.4.21 -do-block-reordering={none,normal,aggressive}

Reorders the control predicates in increasing order of complexity from outer predicate to inner when it is safe. The **normal** mode reorders simple expressions while the **aggressive** mode will reorder predicates involving function calls if it can determine that they have no side-effects. This option is set to **normal** by default.

2.4.22 -fuse-tile-inner-loop

Enables fusion of adjacent tiled loops as a part of loop tiling transformation. This option is set to **false** by default.

2.4.23 -Hz,1,0x1 [Fortran]

[Fortran] Helps to preserve array index information for array access expressions which get linearized in the compiler frontend. The preserved information is used by the compiler optimization phase in performing optimizations such as loop transformations. It is recommended that any user who is using optimizations such as loop transformations and other optimizations requiring de-linearized index expressions should use the Hz option. This option has no impact on any other aspects of the Flang frontend.

2.5 GPU Compilation

This table provides the most commonly-used compiler options for GPU code.

-x hip	Compiles the source file as a HIP program
-fopenmp	Enables the OpenMP support
-fopenmp-targets=<gpu>	Enables the OpenMP target offload support of the specified GPU architecture
--gpu-max-threads-per-block=<value>	Set default launch bounds for kernels
-munsafe-fp-atomics	Enable unsafe floating point atomic instructions (AMDGPU only)
-ffast-math	Allow aggressive, lossy floating-point optimizations
-mwavefrontsize64/-mno-wavefrontsize64	Set wavefront size to be 64 or 32 on Navi architectures
-mcumode	Switch between CU and WGP modes on Navi architectures
--offload-arch=<gpu>	HIP offloading target ID in the form of a device architecture followed by target ID features delimited by a colon. Each target ID feature is a pre-defined string followed by a plus or minus sign (e.g. gfx908:xnack+:sramecc-). May be specified more than once.
-g	Generate source-level debug information
-fgpu-rdc/-fno-gpu-rdc	Generate relocatable device code, also known as separate compilation mode

Chapter 3 Appendix A

3.1 Supported Status of Other Clang Options

Option	Support	Description
-###	Supported	Print (but do not run) the commands to run for this compilation
--analyzer-output <value>	Supported	Static analyzer report output format (html|plist|plist-multi-file|plist-html|sarif|text).
--analyze	Supported	Run the static analyzer
-arcmt-migrate-emit-errors	Unsupported	Emit ARC errors even if the migrator can fix them
-arcmt-migrate-report-output <value>	Unsupported	Output path for the plist report
-byteswapio	Supported	Swap byte-order for unformatted input/output
-B <dir>	Supported	Add <dir> to search path for binaries and object files used implicitly
-CC	Supported	Include comments from within macros in preprocessed output
-cl-denorms-are-zero	Supported	OpenCL only. Allow denormals to be flushed to zero.
-cl-fast-relaxed-math	Supported	OpenCL only. Sets -cl-finite-math-only and -cl-unsafe-math-optimizations, and defines __FAST_RELAXED_MATH__.
-cl-finite-math-only	Supported	OpenCL only. Allow floating-point optimizations that assume arguments and results are not NaNs or +-Inf.
-cl-fp32-correctly-rounded-divide-sqrt	Supported	OpenCL only. Specify that single-precision floating-point divide and sqrt used in the program source are correctly rounded.
-cl-kernel-arg-info	Supported	OpenCL only. Generate kernel argument metadata.
-cl-mad-enable	Supported	OpenCL only. Allow use of less precise MAD computations in the generated binary.
-cl-no-signed-zeros	Supported	OpenCL only. Allow use of less precise no signed zeros computations in the generated binary.
-cl-opt-disable	Supported	OpenCL only. This option disables all optimizations. By default optimizations are enabled.
-cl-single-precision-constant	Supported	OpenCL only. Treat double-precision floating-point constant as single precision constant.
-cl-std=<value>	Supported	OpenCL language standard to compile for.
-cl-strict-aliasing	Supported	OpenCL only. This option is added for compatibility with OpenCL 1.0.
-cl-uniform-work-group-size	Supported	OpenCL only. Defines that the global work-size be a multiple of the work-group size specified to clEnqueueNDRangeKernel

Option	Support	Description
-cl-unsafe-math-optimizations	Supported	OpenCL only. Allow unsafe floating-point optimizations. Also implies -cl-no-signed-zeros and -cl-mad-enable.
--config <value>	Supported	Specifies configuration file
--cuda-compile-host-device	Supported	Compile CUDA code for both host and device (default). Has no effect on non-CUDA compilations.
--cuda-device-only	Supported	Compile CUDA code for device only
--cuda-host-only	Supported	Compile CUDA code for host only. Has no effect on non-CUDA compilations.
--cuda-include-ptx=<value>	Unsupported	Include PTX for the following GPU architecture (e.g. sm_35) or 'all'. May be specified more than once.
--cuda-noopt-device-debug	Unsupported	Enable device-side debug info generation. Disables ptxas optimizations.
--cuda-path-ignore-env	Unsupported	Ignore environment variables to detect CUDA installation
--cuda-path=<value>	Unsupported	CUDA installation path
-cxx-isystem <directory>	Supported	Add a directory to the C++ SYSTEM include search path
-C	Supported	Include comments in preprocessed output
-c	Supported	Only run preprocess, compile, and assemble steps
-dD	Supported	Print macro definitions in -E mode in addition to normal output
-dependency-dot <value>	Supported	Filename to write DOT-formatted header dependencies to
-dependency-file <value>	Supported	Filename (or -) to write dependency output to
-dI	Supported	Print include directives in -E mode in addition to normal output
-dM	Supported	Print macro definitions in -E mode instead of normal output
-dsym-dir <dir>	Unsupported	Directory to output dSYM's (if any) to
-D <macro>	Supported	=<value> Define <macro> to <value> (or 1 if <value> omitted)
-emit-ast	Supported	Emit Clang AST files for source inputs
-emit-interface-stubs	Supported	Generate Interface Stub Files.
-emit-llvm	Supported	Use the LLVM representation for assembler and object files
-emit-merged-ifs	Supported	Generate Interface Stub Files, emit merged text not binary.
--emit-static-lib	Supported	Enable linker job to emit a static library.
-enable-trivial-auto-var-init-zero-knowing-it-will-be-removed-from-clang	Supported	Trivial automatic variable initialization to zero is only here for benchmarks, it'll eventually be removed, and I'm OK with that because I'm only using it to benchmark
-E	Supported	Only run the preprocessor
-fAAPCSBitfieldLoad	Unsupported	Follows the AAPCS standard that all volatile bit-field write generates at least one load. (ARM only).
-faddrsig	Supported	Emit an address-significance table
-faligned-allocation	Supported	Enable C++17 aligned allocation functions

Option	Support	Description
-fallow-editor-placeholders	Supported	Treat editor placeholders as valid source code
-fallow-fortran-gnu-ext	Supported	Allow Fortran GNU extensions
-fansi-escape-codes	Supported	Use ANSI escape codes for diagnostics
-fapple-kext	Unsupported	Use Apple's kernel extensions ABI
-fapple-link-rtlib	Unsupported	Force linking the clang builtins runtime library
-fapple-pragma-pack	Unsupported	Enable Apple gcc-compatible #pragma pack handling
-fapplication-extension	Unsupported	Restrict code to those available for App Extensions
-fbackslash	Supported	Treat backslash as C-style escape character
-fbasic-block-sections=<value>	Supported	Place each function's basic blocks in unique sections (ELF Only) : all | labels | none | list=<file>
-fblocks	Supported	Enable the 'blocks' language feature
-fborland-extensions	Unsupported	Accept non-standard constructs supported by the Borland compiler
-fbuild-session-file=<file>	Supported	Use the last modification time of <file> as the build session timestamp
-fbuild-session-timestamp=<time since Epoch in seconds>	Supported	Time when the current build session started
-fbuiltin-module-map	Unsupported	Load the clang builtins module map file.
-fcall-saved-x10	Unsupported	Make the x10 register call-saved (AArch64 only)
-fcall-saved-x11	Unsupported	Make the x11 register call-saved (AArch64 only)
-fcall-saved-x12	Unsupported	Make the x12 register call-saved (AArch64 only)
-fcall-saved-x13	Unsupported	Make the x13 register call-saved (AArch64 only)
-fcall-saved-x14	Unsupported	Make the x14 register call-saved (AArch64 only)
-fcall-saved-x15	Unsupported	Make the x15 register call-saved (AArch64 only)
-fcall-saved-x18	Unsupported	Make the x18 register call-saved (AArch64 only)
-fcall-saved-x8	Unsupported	Make the x8 register call-saved (AArch64 only)
-fcall-saved-x9	Unsupported	Make the x9 register call-saved (AArch64 only)
-fcf-protection=<value>	Unsupported	Instrument control-flow architecture protection. Options: return, branch, full, none.
-fcf-protection	Unsupported	Enable cf-protection in 'full' mode
-fchar8_t	Supported	Enable C++ builtin type char8_t
-fclang-abi-compat=<version>	Supported	Attempt to match the ABI of Clang <version>
-fcolor-diagnostics	Supported	Enable colors in diagnostics
-fcomment-block-commands=<arg>	Supported	Treat each comma separated argument in <arg> as a documentation comment block command
-fcommon	Supported	Place uninitialized global variables in a common block
-fcomplete-member-pointers	Supported	Require member pointer base types to be complete if they would be significant under the Microsoft ABI
-fconvergent-functions	Supported	Assume functions may be convergent
-fcoroutines-ts	Supported	Enable support for the C++ Coroutines TS
-fcoverage-mapping	Unsupported	Generate coverage mapping to enable code coverage analysis

Option	Support	Description
-fcs-profile-generate=<directory>	Unsupported	Generate instrumented code to collect context sensitive execution counts into <directory>/default.profraw (overridden by LLVM_PROFILE_FILE env var)
-fcs-profile-generate	Unsupported	Generate instrumented code to collect context-sensitive execution counts into default.profraw (overridden by LLVM_PROFILE_FILE env var)
-fcuda-approx-transcendentals	Unsupported	Use approximate transcendental functions
-fcuda-flush-denormals-to-zero	Supported	Flush denormal floating-point values to zero in CUDA device mode.
-fcuda-short-ptr	Unsupported	Use 32-bit pointers for accessing const/local/shared address spaces
-fcxx-exceptions	Supported	Enable C++ exceptions
-fdata-sections	Supported	Place each data in its section
-fdebug-compilation-dir <value>	Supported	The compilation directory to embed in the debug info.
-fdebug-default-version=<value>	Supported	Default DWARF version to use, if a -g option caused DWARF debug info to be produced
-fdebug-info-for-profiling	Supported	Emit extra debug info to make the sample profile more accurate
-fdebug-macro	Supported	Emit macro debug information
-fdebug-prefix-map=<value>	Supported	remap file source paths in debug info
-fdebug-ranges-base-address	Supported	Use DWARF base address selection entries in .debug_ranges
-fdebug-types-section	Supported	Place debug types in their section (ELF Only)
-fdeclspec	Supported	Allow __declspec as a keyword
-fdelayed-template-parsing	Supported	Parse templated function definitions at the end of the translation unit
-fdelete-null-pointer-checks	Supported	Treat usage of null pointers as undefined behavior (default)
-fdiagnostics-absolute-paths	Supported	Print absolute paths in diagnostics
-fdiagnostics-hotness-threshold=<number>	Unsupported	Prevent optimization remarks from being output if they do not have at least this profile count
-fdiagnostics-parseable-fixits	Supported	Print fix-its in machine parseable form
-fdiagnostics-print-source-range-info	Supported	Print source range spans in numeric form
-fdiagnostics-show-hotness	Unsupported	Enable profile hotness information in diagnostic line
-fdiagnostics-show-note-include-stack	Supported	Display include stacks for diagnostic notes
-fdiagnostics-show-option	Supported	Print option name with mappable diagnostics
-fdiagnostics-show-template-tree	Supported	Print a template comparison tree for differing templates

Option	Support	Description
-fdigraphs	Supported	Enable alternative token representations '<:', ':>', '<%', '%>', '%:', '%:%:' (default)
-fdiscard-value-names	Supported	Discard value names in LLVM IR
-fdollars-in-identifiers	Supported	Allow '$' in identifiers
-fdouble-square-bracket-attributes	Supported	Enable '[[]]' attributes in all C and C++ language modes
-fdwarf-exceptions	Unsupported	Use DWARF style exceptions
-feliminate-unused-debug-types	Supported	Do not emit debug info for defined but unused types
-fembed-bitcode-marker	Supported	Embed placeholder LLVM IR data as a marker
-fembed-bitcode=<option>	Supported	Embed LLVM bitcode (option: off, all, bitcode, marker)
-fembed-bitcode	Supported	Embed LLVM IR bitcode as data
-femit-all-decls	Supported	Emit all declarations, even if unused
-femulated-tls	Supported	Use emutls functions to access thread_local variables
-fenable-matrix	Supported	Enable matrix data type and related builtin functions
-fexceptions	Supported	Enable support for exception handling
-fexperimental-new-constant-interpreter	Supported	Enable the experimental new constant interpreter
-fexperimental-new-pass-manager	Supported	Enables an experimental new pass manager in LLVM.
-fexperimental-relative-c++-abi-vtables	Supported	Use the experimental C++ class ABI for classes with virtual tables
-fexperimental-strict-floating-point	Supported	Enables experimental strict floating point in LLVM.
-ffast-math	Supported	Allow aggressive, lossy floating-point optimizations
-ffile-prefix-map=<value>	Supported	remap file source paths in debug info and predefined preprocessor macros
-ffine-grained-bitfield-accesses	Supported	Use separate accesses for consecutive bitfield runs with legal widths and alignments.
-ffixed-form	Supported	Enable fixed-form format for Fortran
-ffixed-point	Supported	Enable fixed point types
-ffixed-r19	Unsupported	Reserve register r19 (Hexagon only)
-ffixed-r9	Unsupported	Reserve the r9 register (ARM only)
-ffixed-x10	Unsupported	Reserve the x10 register (AArch64/RISC-V only)
-ffixed-x11	Unsupported	Reserve the x11 register (AArch64/RISC-V only)
-ffixed-x12	Unsupported	Reserve the x12 register (AArch64/RISC-V only)
-ffixed-x13	Unsupported	Reserve the x13 register (AArch64/RISC-V only)
-ffixed-x14	Unsupported	Reserve the x14 register (AArch64/RISC-V only)
-ffixed-x15	Unsupported	Reserve the x15 register (AArch64/RISC-V only)
-ffixed-x16	Unsupported	Reserve the x16 register (AArch64/RISC-V only)
-ffixed-x17	Unsupported	Reserve the x17 register (AArch64/RISC-V only)
-ffixed-x18	Unsupported	Reserve the x18 register (AArch64/RISC-V only)
-ffixed-x19	Unsupported	Reserve the x19 register (AArch64/RISC-V only)
-ffixed-x1	Unsupported	Reserve the x1 register (AArch64/RISC-V only)
-ffixed-x20	Unsupported	Reserve the x20 register (AArch64/RISC-V only)

Option	Support	Description
-ffixed-x21	Unsupported	Reserve the x21 register (AArch64/RISC-V only)
-ffixed-x22	Unsupported	Reserve the x22 register (AArch64/RISC-V only)
-ffixed-x23	Unsupported	Reserve the x23 register (AArch64/RISC-V only)
-ffixed-x24	Unsupported	Reserve the x24 register (AArch64/RISC-V only)
-ffixed-x25	Unsupported	Reserve the x25 register (AArch64/RISC-V only)
-ffixed-x26	Unsupported	Reserve the x26 register (AArch64/RISC-V only)
-ffixed-x27	Unsupported	Reserve the x27 register (AArch64/RISC-V only)
-ffixed-x28	Unsupported	Reserve the x28 register (AArch64/RISC-V only)
-ffixed-x29	Unsupported	Reserve the x29 register (AArch64/RISC-V only)
-ffixed-x2	Unsupported	Reserve the x2 register (AArch64/RISC-V only)
-ffixed-x30	Unsupported	Reserve the x30 register (AArch64/RISC-V only)
-ffixed-x31	Unsupported	Reserve the x31 register (AArch64/RISC-V only)
-ffixed-x3	Unsupported	Reserve the x3 register (AArch64/RISC-V only)
-ffixed-x4	Unsupported	Reserve the x4 register (AArch64/RISC-V only)
-ffixed-x5	Unsupported	Reserve the x5 register (AArch64/RISC-V only)
-ffixed-x6	Unsupported	Reserve the x6 register (AArch64/RISC-V only)
-ffixed-x7	Unsupported	Reserve the x7 register (AArch64/RISC-V only)
-ffixed-x8	Unsupported	Reserve the x8 register (AArch64/RISC-V only)
-ffixed-x9	Unsupported	Reserve the x9 register (AArch64/RISC-V only)
-fforce-dwarf-frame	Supported	Always emit a debug frame section
-fforce-emit-vtables	Supported	Emits more virtual tables to improve devirtualization
-fforce-enable-int128	Supported	Enable support for int128_t type
-ffp-contract=<value>	Supported	Form fused FP ops (e.g. FMAs): fast (everywhere) | on (according to FP_CONTRACT pragma) | off (never fuse). Default is 'fast' for CUDA/HIP and 'on' otherwise.
-ffp-exception-behavior=<value>	Supported	Specifies the exception behavior of floating-point operations.
-ffp-model=<value>	Supported	Controls the semantics of floating-point calculations.
-ffree-form	Supported	Enable free-form format for Fortran
-ffreestanding	Supported	Assert that the compilation takes place in a freestanding environment
-ffunc-args-alias	Supported	Function argument may alias (equivalent to ansi alias)
-ffunction-sections	Supported	Place each function in its section
-fglobal-isel	Supported	Enables the global instruction selector
-fgnu-keywords	Supported	Allow GNU-extension keywords regardless of a language standard
-fgnu-runtime	Unsupported	Generate output compatible with the standard GNU Objective-C runtime
-fgnu89-inline	Unsupported	Use the gnu89 inline semantics
-fgnuc-version=<value>	Supported	Sets various macros to claim compatibility with the given GCC version (default is 4.2.1)
-fgpu-allow-device-init	Supported	Allow device-side init function in HIP
-fgpu-rdc	Supported	Generate relocatable device code, also known as separate compilation mode
-fhip-new-launch-api	Supported	Use new kernel launching API for HIP

Option	Support	Description
-fignore-exceptions	Supported	Enable support for ignoring exception handling constructs
-fimplicit-module-maps	Unsupported	Implicitly search the file system for module map files.
-finline-functions	Supported	Inline suitable functions
-finline-hint-functions	Supported	Inline functions that are (explicitly or implicitly) marked inline
-finstrument-function-entry-bare	Unsupported	Instrument function entry only, after inlining, without arguments to the instrumentation call
-finstrument-functions-after-inlining	Unsupported	Like -finstrument-functions, but insert the calls after inlining
-finstrument-functions	Unsupported	Generate calls to instrument function entry and exit
-fintegrated-as	Supported	Enable the integrated assembler
-fintegrated-cc1	Supported	Run cc1 in-process
-fjump-tables	Supported	Use jump tables for lowering switches
-fkeep-static-consts	Supported	Keep static const variables if unused
-flax-vector-conversions=<value>	Supported	Enable implicit vector bit-casts
-flto-jobs=<value>	Unsupported	Controls the backend parallelism of -flto=thin (default of 0 means the number of threads will be derived from the number of CPUs detected)
-flto=<value>	Unsupported	Set LTO mode to either 'full' or 'thin'
-flto	Unsupported	Enable LTO in 'full' mode
-fmacro-prefix-map=<value>	Supported	remap file source paths in predefined preprocessor macros
-fmath-errno	Supported	Require math functions to indicate errors by setting errno
-fmax-tokens=<value>	Supported	Max total number of preprocessed tokens for -Wmax-tokens.
-fmax-type-align=<value>	Supported	Specify the maximum alignment to enforce on pointers lacking an explicit alignment
-fmemory-profile	Supported	Enable heap memory profiling
-fmerge-all-constants	Supported	Allow merging of constants
-fmessage-length=<value>	Supported	Format message diagnostics so that they fit within N columns
-fmodule-file=[<name>=]<file>	Unsupported	Specify the mapping of module name to precompiled module file, or load a module file if name is omitted.
-fmodule-map-file=<file>	Unsupported	Load this module map file
-fmodule-name=<name>	Unsupported	Specify the name of the module to build
-fmodules-cache-path=<directory>	Unsupported	Specify the module cache path
-fmodules-decluse	Unsupported	Require declaration of modules used within a module
-fmodules-disable-diagnostic-validation	Unsupported	Disable validation of the diagnostic options when loading the module
-fmodules-ignore-macro=<value>	Unsupported	Ignore the definition of the given macro when building and loading modules

Option	Support	Description
-fmodules-prune-after=<seconds>	Unsupported	Specify the interval (in seconds) after which a module file will be considered unused
-fmodules-prune-interval=<seconds>	Unsupported	Specify the interval (in seconds) between attempts to prune the module cache
-fmodules-search-all	Unsupported	Search even non-imported modules to resolve references
-fmodules-strict-decluse	Unsupported	Like -fmodules-decluse but requires all headers to be in modules
-fmodules-ts	Unsupported	Enable support for the C++ Modules TS
-fmodules-user-build-path <directory>	Unsupported	Specify the module user build path
-fmodules-validate-input-files-content	Supported	Validate PCM input files based on content if mtime differs
-fmodules-validate-once-per-build-session	Unsupported	Don't verify input files for the modules if the module has been successfully validated or loaded during this build session
-fmodules-validate-system-headers	Supported	Validate the system headers that a module depends on when loading the module
-fmodules	Unsupported	Enable the 'modules' language feature
-fms-compatibility-version=<value>	Supported	Dot-separated value representing the Microsoft compiler version number to report in _MSC_VER (0 = don't define it (default))
-fms-compatibility	Supported	Enable full Microsoft Visual C++ compatibility
-fms-extensions	Supported	Accept some non-standard constructs supported by the Microsoft compiler
-fmsc-version=<value>	Supported	Microsoft compiler version number to report in _MSC_VER (0 = don't define it (default))
-fnew-alignment=<align>	Supported	Specifies the largest alignment guaranteed by '::operator new(size_t)'
-fno-addrsig	Supported	Don't emit an address-significance table
-fno-allow-fortran-gnu-ext	Supported	Allow Fortran GNU extensions
-fno-assume-sane-operator-new	Supported	Don't assume that C++'s global operator new can't alias any pointer
-fno-autolink	Supported	Disable generation of linker directives for automatic library linking
-fno-backslash	Supported	Treat backslash like any other character in character strings
-fno-builtin-<value>	Supported	Disable implicit builtin knowledge of a specific function
-fno-builtin	Supported	Disable implicit builtin knowledge of functions
-fno-c++-static-destructors	Supported	Disable C++ static destructor registration
-fno-char8_t	Supported	Disable C++ builtin type char8_t
-fno-color-diagnostics	Supported	Disable colors in diagnostics
-fno-common	Supported	Compile common globals like normal definitions
-fno-complete-member-pointers	Supported	Do not require member pointer base types to be complete if they would be significant under the Microsoft ABI
-fno-constant-cfstrings	Supported	Disable creation of CodeFoundation-type constant strings
-fno-coverage-mapping	Supported	Disable code coverage analysis

Option	Support	Description
-fno-crash-diagnostics	Supported	Disable auto-generation of preprocessed source files and a script for reproduction during a clang crash
-fno-cuda-approx-transcendentals	Unsupported	Don't use approximate transcendental functions
-fno-debug-macro	Supported	Do not emit macro debug information
-fno-declspec	Unsupported	Disallow __declspec as a keyword
-fno-delayed-template-parsing	Supported	Disable delayed template parsing
-fno-delete-null-pointer-checks	Supported	Do not treat usage of null pointers as undefined behavior
-fno-diagnostics-fixit-info	Supported	Do not include fixit information in diagnostics
-fno-digraphs	Supported	Disallow alternative token representations '<:', ':>', '<%', '%>', '%:', '%:%:'
-fno-discard-value-names	Supported	Do not discard value names in LLVM IR
-fno-dollars-in-identifiers	Supported	Disallow '$' in identifiers
-fno-double-square-bracket-attributes	Supported	Disable '[[]]' attributes in all C and C++ language modes
-fno-elide-constructors	Supported	Disable C++ copy constructor elision
-fno-elide-type	Supported	Do not elide types when printing diagnostics
-fno-eliminate-unused-debug-types	Supported	Emit debug info for defined but unused types
-fno-exceptions	Supported	Disable support for exception handling
-fno-experimental-new-pass-manager	Supported	Disables an experimental new pass manager in LLVM.
-fno-experimental-relative-c++-abi-vtables	Supported	Do not use the experimental C++ class ABI for classes with virtual tables
-fno-fine-grained-bitfield-accesses	Supported	Use large-integer access for consecutive bitfield runs.
-fno-fixed-form	Supported	Disable fixed-form format for Fortran
-fno-fixed-point	Supported	Disable fixed point types
-fno-force-enable-int128	Supported	Disable support for int128_t type
-fno-fortran-main	Supported	Don't link in Fortran main
-fno-free-form	Supported	Disable free-form format for Fortran
-fno-func-args-alias	Supported	Function argument may alias (equivalent to ansi alias)
-fno-global-isel	Supported	Disables the global instruction selector
-fno-gnu-inline-asm	Supported	Disable GNU style inline asm
-fno-gpu-allow-device-init	Supported	Don't allow device-side init function in HIP
-fno-hip-new-launch-api	Supported	Don't use new kernel launching API for HIP
-fno-integrated-as	Supported	Disable the integrated assembler
-fno-integrated-cc1	Supported	Spawn a separate process for each cc1
-fno-jump-tables	Supported	Do not use jump tables for lowering switches
-fno-keep-static-consts	Supported	Don't keep static const variables if unused
-fno-lto	Supported	Disable LTO mode (default)
-fno-memory-profile	Supported	Disable heap memory profiling
-fno-merge-all-constants	Supported	Disallow merging of constants

Option	Support	Description
-fno-no-access-control	Supported	Disable C++ access control
-fno-objc-infer-related-result-type	Supported	do not infer Objective-C related result type based on method family
-fno-operator-names	Supported	Do not treat C++ operator name keywords as synonyms for operators
-fno-pch-codegen	Supported	Do not generate code for uses of this PCH that assumes an explicit object file will be built for the PCH
-fno-pch-debuginfo	Supported	Do not generate debug info for types in an object file built from this PCH and do not generate them elsewhere
-fno-plt	Supported	Use GOT indirection instead of PLT to make external function calls (x86 only)
-fno-preserve-as-comments	Supported	Do not preserve comments in inline assembly
-fno-profile-generate	Supported	Disable generation of profile instrumentation.
-fno-profile-instr-generate	Supported	Disable generation of profile instrumentation.
-fno-profile-instr-use	Supported	Disable using instrumentation data for profile-guided optimization
-fno-register-global-dtors-with-atexit	Supported	Don't use atexit or __cxa_atexit to register global destructors
-fno-rtlib-add-rpath	Supported	Do not add -rpath with architecture-specific resource directory to the linker flags
-fno-rtti-data	Supported	Disable generation of RTTI data
-fno-rtti	Supported	Disable generation of rtti information
-fno-sanitize-address-poison-custom-array-cookie	Supported on Host only	Disable poisoning array cookies when using custom operator new[] in AddressSanitizer
-fno-sanitize-address-use-after-scope	Supported on Host only	Disable use-after-scope detection in AddressSanitizer
-fno-sanitize-address-use-odr-indicator	Supported on Host only	Disable ODR indicator globals
-fno-sanitize-blacklist	Supported on Host only	Don't use blacklist file for sanitizers
-fno-sanitize-cfi-canonical-jump-tables	Supported on Host only	Do not make the jump table addresses canonical in the symbol table
-fno-sanitize-cfi-cross-dso	Supported on Host only	Disable control flow integrity (CFI) checks for cross-DSO calls.
-fno-sanitize-coverage=<value>	Supported on Host only	Disable specified features of coverage instrumentation for Sanitizers
-fno-sanitize-memory-track-origins	Supported on Host only	Disable origins tracking in MemorySanitizer
-fno-sanitize-memory-use-after-dtor	Supported on Host only	Disable use-after-destroy detection in MemorySanitizer
-fno-sanitize-recover=<value>	Supported on Host only	Disable recovery for specified sanitizers
-fno-sanitize-stats	Supported on Host only	Disable sanitizer statistics gathering.

Option	Support	Description
-fno-sanitize-thread-atomics	Supported on Host only	Disable atomic operations instrumentation in ThreadSanitizer
-fno-sanitize-thread-func-entry-exit	Supported on Host only	Disable function entry/exit instrumentation in ThreadSanitizer
-fno-sanitize-thread-memory-access	Supported on Host only	Disable memory access instrumentation in ThreadSanitizer
-fno-sanitize-trap=<value>	Supported on Host only	Disable trapping for specified sanitizers
-fno-sanitize-trap	Supported on Host only	Disable trapping for all sanitizers
-fno-short-wchar	Supported	Force wchar_t to be an unsigned int
-fno-show-column	Supported	Do not include column number on diagnostics
-fno-show-source-location	Supported	Do not include source location information with diagnostics
-fno-signed-char	Supported	char is unsigned
-fno-signed-zeros	Supported	Allow optimizations that ignore the sign of floating point zeros
-fno-spell-checking	Supported	Disable spell-checking
-fno-split-machine-functions	Supported	Disable late function splitting using profile information (x86 ELF)
-fno-stack-clash-protection	Supported	Disable stack clash protection
-fno-stack-protector	Supported	Disable the use of stack protectors
-fno-standalone-debug	Supported	Limit debug information produced to reduce size of debug binary
-fno-strict-float-cast-overflow	Supported	Relax language rules and try to match the behavior of the target's native float-to-int conversion instructions
-fno-strict-return	Supported	Don't treat control flow paths that fall off the end of a non-void function as unreachable
-fno-sycl	Unsupported	Disable SYCL kernels compilation for device
-fno-temp-file	Supported	Directly create compilation output files. This may lead to incorrect incremental builds if the compiler crashes
-fno-threadsafe-statics	Supported	Do not emit code to make initialization of local statics thread safe
-fno-trigraphs	Supported	Do not process trigraph sequences
-fno-unique-section-names	Supported	Don't use unique names for text and data sections
-fno-unroll-loops	Supported	Turn off loop unroller
-fno-use-cxa-atexit	Supported	Don't use __cxa_atexit for calling destructors
-fno-use-flang-math-libs	Supported	Use Flang internal runtime math library instead of LLVM math intrinsics.
-fno-use-init-array	Supported	Use .ctors/.dtors instead of .init_array/.fini_array
-fno-visibility-inlines-hidden-static-local-var	Supported	Disables -fvisibility-inlines-hidden-static-local-var (this is the default on non-darwin targets)
-fno-xray-function-index	Unsupported	Omit function index section at the expense of single-function patching performance
-fno-zero-initialized-in-bss	Supported	Don't place zero initialized data in BSS

Option	Support	Description
-fobjc-arc-exceptions	Unsupported	Use EH-safe code when synthesizing retains and releases in -fobjc-arc
-fobjc-arc	Unsupported	Synthesize retain and release calls for Objective-C pointers
-fobjc-exceptions	Unsupported	Enable Objective-C exceptions
-fobjc-runtime=<value>	Unsupported	Specify the target Objective-C runtime kind and version
-fobjc-weak	Unsupported	Enable ARC-style weak references in Objective-C
-fopenmp-simd	Unsupported	Emit OpenMP code only for SIMD-based constructs.
-fopenmp-targets=<value>	Unsupported	Specify a comma-separated list of triples OpenMP offloading targets to be supported
-fopenmp	Unsupported	Parse OpenMP pragmas and generate parallel code.
-foptimization-record-file=<file>	Supported	Specify the output name of the file containing the optimization remarks. Implies -fsave-optimization-record. On Darwin platforms, this cannot be used with multiple -arch <arch> options.
-foptimization-record-passes=<regex>	Supported	Only include passes that match a specified regular expression in the generated optimization record (by default, include all passes)
-forder-file-instrumentation	Supported	Generate instrumented code to collect order file into default.profraw file (overridden by '=' form of option or LLVM_PROFILE_FILE env var)
-fpack-struct=<value>	Unsupported	Specify the default maximum struct packing alignment
-fpascal-strings	Supported	Recognize and construct Pascal-style string literals
-fpass-plugin=<dsopath>	Supported	Load pass plugin from a dynamic shared object file (only with new pass manager).
-fpatchable-function-entry=<N,M>	Supported	Generate M NOPs before function entry and N-M NOPs after function entry
-fpcc-struct-return	Unsupported	Override the default ABI to return all structs on the stack
-fpch-codegen	Supported	Generate code for uses of this PCH that assumes an explicit object file will be built for the PCH
-fpch-debuginfo	Supported	Generate debug info for types in an object file built from this PCH and do not generate them elsewhere
-fpch-instantiate-templates	Supported	Instantiate templates already while building a PCH
-fpch-validate-input-files-content	Supported	Validate PCH input files based on content if mtime differs
-fplugin=<dsopath>	Supported	Load the named plugin (dynamic shared object)
-fprebuilt-module-path=<directory>	Unsupported	Specify the prebuilt module path
-fprofile-exclude-files=<value>	Unsupported	Instrument only functions from files where names don't match all the regexes separated by a semi-colon
-fprofile-filter-files=<value>	Unsupported	Instrument only functions from files where names match any regex separated by a semi-colon
-fprofile-generate=<directory>	Unsupported	Generate instrumented code to collect execution counts into <directory>/default.profraw (overridden by LLVM_PROFILE_FILE env var)

Option	Support	Description
-fprofile-generate	Unsupported	Generate instrumented code to collect execution counts into default.profraw (overridden by LLVM_PROFILE_FILE env var)
-fprofile-instr-generate=<file>	Unsupported	Generate instrumented code to collect execution counts into <file> (overridden by LLVM_PROFILE_FILE env var)
-fprofile-instr-generate	Unsupported	Generate instrumented code to collect execution counts into default.profraw file (overridden by '=' form of option or LLVM_PROFILE_FILE env var)
-fprofile-instr-use=<value>	Unsupported	Use instrumentation data for profile-guided optimization
-fprofile-remapping-file=<file>	Unsupported	Use the remappings described in <file> to match the profile data against names in the program
-fprofile-sample-accurate	Unsupported	Specifies that the sample profile is accurate
-fprofile-sample-use=<value>	Unsupported	Enable sample-based profile guided optimizations
-fprofile-use=<pathname>	Unsupported	Use instrumentation data for profile-guided optimization. If pathname is a directory, it reads from <pathname>/default.profdata. Otherwise, it reads from file <pathname>.
-freciprocal-math	Supported	Allow division operations to be reassociated
-freg-struct-return	Unsupported	Override the default ABI to return small structs in registers
-fregister-global-dtors-with-atexit	Supported	Use atexit or __cxa_atexit to register global destructors
-frelaxed-template-template-args	Supported	Enable C++17 relaxed template argument matching
-freroll-loops	Supported	Turn on loop reroller
-fropi	Unsupported	Generate read-only position independent code (ARM only)
-frtlib-add-rpath	Supported	Add -rpath with architecture-specific resource directory to the linker flags
-frwpi	Unsupported	Generate read-write position independent code (ARM only)
-fsanitize-address-field-padding=<value>	Supported on Host only	Level of field padding for AddressSanitizer
-fsanitize-address-globals-dead-stripping	Supported on Host only	Enable linker dead stripping of globals in AddressSanitizer
-fsanitize-address-poison-custom-array-cookie	Supported on Host only	Enable poisoning array cookies when using custom operator new[] in AddressSanitizer
-fsanitize-address-use-after-scope	Supported on Host only	Enable use-after-scope detection in AddressSanitizer
-fsanitize-address-use-odr-indicator	Supported on Host only	Enable ODR indicator globals to avoid false ODR violation reports in partially sanitized programs at the cost of an increase in binary size
-fsanitize-blacklist=<value>	Supported on Host only	Path to blacklist file for sanitizers

Option	Support	Description
-fsanitize-cfi-canonical-jump-tables	Supported on Host only	Make the jump table addresses canonical in the symbol table
-fsanitize-cfi-cross-dso	Supported on Host only	Enable control flow integrity (CFI) checks for cross-DSO calls.
-fsanitize-cfi-icall-generalize-pointers	Supported on Host only	Generalize pointers in CFI indirect call type signature checks
-fsanitize-coverage-allowlist=<value>	Supported on Host only	Restrict sanitizer coverage instrumentation exclusively to modules and functions that match the provided special case list, except the blocked ones
-fsanitize-coverage-blacklist=<value>	Supported on Host only	Deprecated, use -fsanitize-coverage-blocklist= instead
-fsanitize-coverage-blocklist=<value>	Supported on Host only	Disable sanitizer coverage instrumentation for modules and functions that match the provided special case list, even the allowed ones
-fsanitize-coverage-whitelist=<value>	Supported on Host only	Deprecated, use -fsanitize-coverage-allowlist= instead
-fsanitize-coverage=<value>	Supported on Host only	Specify the type of coverage instrumentation for Sanitizers
-fsanitize-hwaddress-abi=<value>	Supported on Host only	Select the HWAddressSanitizer ABI to target (interceptor or platform, default interceptor). This option is currently unused.
-fsanitize-memory-track-origins=<value>	Supported on Host only	Enable origins tracking in MemorySanitizer
-fsanitize-memory-track-origins	Supported on Host only	Enable origins tracking in MemorySanitizer
-fsanitize-memory-use-after-dtor	Supported on Host only	Enable use-after-destroy detection in MemorySanitizer
-fsanitize-recover=<value>	Supported on Host only	Enable recovery for specified sanitizers
-fsanitize-stats	Supported on Host only	Enable sanitizer statistics gathering.
-fsanitize-system-blacklist=<value>	Supported on Host only	Path to system blacklist file for sanitizers
-fsanitize-thread-atomics	Supported on Host only	Enable atomic operations instrumentation in ThreadSanitizer (default)
-fsanitize-thread-func-entry-exit	Supported on Host only	Enable function entry/exit instrumentation in ThreadSanitizer (default)
-fsanitize-thread-memory-access	Supported on Host only	Enable memory access instrumentation in ThreadSanitizer (default)
-fsanitize-trap=<value>	Supported on Host only	Enable trapping for specified sanitizers
-fsanitize-trap	Supported on Host only	Enable trapping for all sanitizers
-fsanitize-undefined-strip-path-components=<number>	Supported on Host only	Strip (or keep only, if negative) a given number of path components when emitting check metadata.

Option	Support	Description
-fsanitize=<check>	Supported on Host only	Turn on runtime checks for various forms of undefined or suspicious behavior. See user manual for available checks
-fsave-optimization-record=<format>	Supported	Generate an optimization record file in a specific format
-fsave-optimization-record	Supported	Generate a YAML optimization record file
-fseh-exceptions	Supported	Use SEH style exceptions
-fshort-enums	Supported	Allocate to an enum type only as many bytes as it needs for the declared range of possible values
-fshort-wchar	Unsupported	Force wchar_t to be a short unsigned int
-fshow-overloads=<value>	Supported	Which overload candidates to show when overload resolution fails: best|all; defaults to all
-fsigned-char	Supported	char is signed
-fsized-deallocation	Supported	Enable C++14 sized global deallocation functions
-fsjlj-exceptions	Supported	Use SjLj style exceptions
-fslp-vectorize	Supported	Enable the superword-level parallelism vectorization passes
-fsplit-dwarf-inlining	Unsupported	Provide minimal debug info in the object/executable to facilitate online symbolication/stack traces in the absence of .dwo/.dwp files when using Split DWARF
-fsplit-lto-unit	Unsupported	Enables splitting of the LTO unit
-fsplit-machine-functions	Supported	Enable late function splitting using profile information (x86 ELF)
-fstack-clash-protection	Supported	Enable stack clash protection
-fstack-protector-all	Unsupported	Enable stack protectors for all functions
-fstack-protector-strong	Unsupported	Enable stack protectors for some functions vulnerable to stack smashing. Compared to -fstack-protector, this uses a stronger heuristic that includes functions containing arrays of any size (and any type), as well as any calls to alloca or the taking of an address from a local variable
-fstack-protector	Unsupported	Enable stack protectors for some functions vulnerable to stack smashing. This uses a loose heuristic that considers functions vulnerable if they contain a char (or 8bit integer) array or constant sized calls to alloca , which are of greater size than ssp-buffer-size (default: 8 bytes). All variable sized calls to alloca are considered vulnerable. A function with a stack protector has a guard value added to the stack frame that is checked on function exit. The guard value must be positioned in the stack frame such that a buffer overflow from a vulnerable variable will overwrite the guard value before overwriting the function's return address. The reference stack guard value is stored in a global variable.
-fstack-size-section	Supported	Emit section containing metadata on function stack sizes
-fstandalone-debug	Supported	Emit full debug info for all types used by the program

Option	Support	Description
-fstrict-enums	Supported	Enable optimizations based on the strict definition of an enum's value range
-fstrict-float-cast-overflow	Supported	Assume that overflowing float-to-int casts are undefined (default)
-fstrict-vtable-pointers	Supported	Enable optimizations based on the strict rules for overwriting polymorphic C++ objects
-fsycl	Unsupported	Enable SYCL kernels compilation for device
-fsystem-module	Unsupported	Build this module as a system module. Only used with -emit-module
-fthin-link-bitcode=<value>	Supported	Write minimized bitcode to <file> for the ThinLTO thin link only
-fthinlto-index=<value>	Unsupported	Perform ThinLTO importing using the provided function summary index
-ftime-trace-granularity=<value>	Supported	Minimum time granularity (in microseconds) traced by time profiler
-ftime-trace	Supported	Turn on time profiler. Generates JSON file based on output filename.
-ftrap-function=<value>	Unsupported	Issue call to specified function rather than a trap instruction
-ftrapv-handler=<function name>	Unsupported	Specify the function to be called on overflow
-ftrapv	Unsupported	Trap on integer overflow
-ftrigraphs	Supported	Process trigraph sequences
-ftrivial-auto-var-init-stop-after=<value>	Supported	Stop initializing trivial automatic stack variables after the specified number of instances
-ftrivial-auto-var-init=<value>	Supported	Initialize trivial automatic stack variables: uninitialized (default) | pattern
-funique-basic-block-section-names	Supported	Use unique names for basic block sections (ELF Only)
-funique-internal-linkage-names	Supported	Uniqueify Internal Linkage Symbol Names by appending the MD5 hash of the module path
-funroll-loops	Supported	Turn on loop unroller
-fuse-flang-math-libs	Supported	Use Flang internal runtime math library instead of LLVM math intrinsics.
-fuse-line-directives	Supported	Use #line in preprocessed output
-fvalidate-ast-input-files-content	Supported	Compute and store the hash of input files used to build an AST. Files with mismatching mtime's are considered valid if both contents is identical
-fveclib=<value>	Unsupported	Use the given vector functions library
-fvectorize	Unsupported	Enable the loop vectorization passes
-fverbose-asm	Supported	Generate verbose assembly output
-fvirtual-function-elimination	Supported	Enables dead virtual function elimination optimization. Requires -flto=full
-fvisibility-global-new-delete-hidden	Supported	Give global C++ operator new and delete declarations hidden visibility

Option	Support	Description
-fvisibility-inlines-hidden-static-local-var	Supported	When -fvisibility-inlines-hidden is enabled, static variables in inline C++ member functions will also be given hidden visibility by default
-fvisibility-inlines-hidden	Supported	Give inline C++ member functions hidden visibility by default
-fvisibility-ms-compat	Supported	Give global types 'default' visibility and global functions and variables 'hidden' visibility by default
-fvisibility=<value>	Supported	Set the default symbol visibility for all global declarations
-fwasm-exceptions	Unsupported	Use WebAssembly style exceptions
-fwhole-program-vtables	Unsupported	Enables whole-program vtable optimization. Requires -flto
-fwrapv	Supported	Treat signed integer overflow as two's complement
-fwritable-strings	Supported	Store string literals as writable data
-fxray-always-emit-customevents	Unsupported	Always emit __xray_customevent(...) calls even if the containing function is not always instrumented
-fxray-always-emit-typedevents	Unsupported	Always emit __xray_typedevent(...) calls even if the containing function is not always instrumented
-fxray-always-instrument=<value>	Unsupported	DEPRECATED: Filename defining the whitelist for imbuing the 'always instrument' XRay attribute.
-fxray-attr-list= <value>	Unsupported	Filename defining the list of functions/types for imbuing XRay attributes.
-fxray-ignore-loops	Unsupported	Don't instrument functions with loops unless they also meet the minimum function size
-fxray-instruction-threshold= <value>	Unsupported	Sets the minimum function size to instrument with XRay
-fxray-instrumentation-bundle= <value>	Unsupported	Select which XRay instrumentation points to emit. Options: all, none, function-entry, function-exit, function, custom. Default is 'all'. 'function' includes both 'function-entry' and 'function-exit'.
-fxray-instrument	Unsupported	Generate XRay instrumentation sleds on function entry and exit
-fxray-link-deps	Unsupported	Tells clang to add the link dependencies for XRay.
-fxray-modes= <value>	Unsupported	List of modes to link in by default into XRay instrumented binaries.
-fxray-never-instrument=<value>	Unsupported	DEPRECATED: Filename defining the whitelist for imbuing the 'never instrument' XRay attribute.
-fzvector	Supported	Enable System z vector language extension
-F <value>	Unsupported	Add directory to framework include search path
--gcc-toolchain=<value>	Supported	Use the gcc toolchain at the given directory
-gcodeview-ghash	Supported	Emit type record hashes in a .debug$H section
-gcodeview	Supported	Generate CodeView debug information
-gdwarf-2	Supported	Generate source-level debug information with dwarf version 2
-gdwarf-3	Supported	Generate source-level debug information with dwarf version 3

Option	Support	Description
-gdwarf-4	Supported	Generate source-level debug information with dwarf version 4
-gdwarf-5	Supported	Generate source-level debug information with dwarf version 5
-gdwarf	Supported	Generate source-level debug information with the default dwarf version
-gembed-source	Supported	Embed source text in DWARF debug sections
-gline-directives-only	Supported	Emit debug line info directives only
-gline-tables-only	Supported	Emit debug line number tables only
-gmodules	Supported	Generate debug info with external references to clang modules or precompiled headers
-gno-embed-source	Supported	Restore the default behavior of not embedding source text in DWARF debug sections
-gno-inline-line-tables	Supported	Don't emit inline line tables
--gpu-max-threads-per-block=<value>	Supported	Default max threads per block for kernel launch bounds for HIP
-gsplit-dwarf=<value>	Supported	Set DWARF fission mode to either 'split' or 'single'
-gz=<value>	Supported	DWARF debug sections compression type
-gz	Supported	DWARF debug sections compression type
-G <size>	Unsupported	Put objects of at most <size> bytes into small data section (MIPS / Hexagon)
-g	Supported	Generate source-level debug information
--help-hidden	Supported	Display help for hidden options
-help	Supported	Display available options
--hip-device-lib=<value>	Supported	HIP device library
--hip-link	Supported	Link clang-offload-bundler bundles for HIP
--hip-version=<value>	Supported	HIP version in the format of major.minor.patch
-H	Supported	Show header includes and nesting depth
-I-	Supported	Restrict all prior -I flags to double-quoted inclusion and remove the current directory from include path
-ibuiltininc	Supported	Enable builtin #include directories even when -nostdinc is used before or after -ibuiltininc. Using -nobuiltininc after the option disables it
-idirafter <value>	Supported	Add directory to AFTER include search path
-iframeworkwithsysroot <directory>	Unsupported	Add directory to SYSTEM framework search path, absolute paths are relative to -isysroot
-iframework <value>	Unsupported	Add directory to SYSTEM framework search path
-imacros <file>	Supported	Include macros from file before parsing
-include-pch <file>	Supported	Include precompiled header file
-include <file>	Supported	Include file before parsing
-index-header-map	Supported	Make the next included directory (-I or -F) an indexer header map
-iprefix <dir>	Supported	Set the -iwithprefix/-iwithprefixbefore prefix
-iquote <directory>	Supported	Add directory to QUOTE include search path
-isysroot <dir>	Supported	Set the system root directory (usually /)

Option	Support	Description
-isystem-after <directory>	Supported	Add directory to end of the SYSTEM include search path
-isystem <directory>	Supported	Add directory to SYSTEM include search path
-ivfsoverlay <value>	Supported	Overlay the virtual filesystem described by file over the real file system
-iwithprefixbefore <dir>	Supported	Set directory to include search path with prefix
-iwithprefix <dir>	Supported	Set directory to SYSTEM include search path with prefix
-iwithsysroot <directory>	Supported	Add directory to SYSTEM include search path, absolute paths are relative to -isysroot
-I <dir>	Supported	Add directory to include search path. If there are multiple -I options, these directories are searched in the order they are given before the standard system directories are searched. If the same directory is in the SYSTEM include search paths, for example, if also specified with -isystem, the -I option will be ignored
--libomptarget-nvptx-path=<value>	Unsupported	Path to libomptarget-nvptx libraries
-L <dir>	Supported	Add directory to library search path
-mabicalls	Unsupported	Enable SVR4-style position-independent code (Mips only)
-maix-struct-return	Unsupported	Return all structs in memory (PPC32 only)
-malign-branch-boundary=<value>	Supported	Specify the boundary's size to align branches
-malign-branch=<value>	Supported	Specify types of branches to align
-malign-double	Supported	Align doubles to two words in structs (x86 only)
-Mallocatable=<value>	Unsupported	Select semantics for assignments to allocatables (F03 or F95)
-mbackchain	Unsupported	Link stack frames through backchain on System Z
-mbranch-protection=<value>	Unsupported	Enforce targets of indirect branches and function returns
-mbranches-within-32B-boundaries	Supported	Align selected branches (fused, jcc, jmp) within 32-byte boundary
-mcmodel=medany	Unsupported	Equivalent to -mcmodel=medium, compatible with RISC-V gcc.
-mcmodel=medlow	Unsupported	Equivalent to -mcmodel=small, compatible with RISC-V gcc.
-mcmse	Unsupported	Allow use of CMSE (Armv8-M Security Extensions)
-mcode-object-v3	Supported	Legacy option to specify code object ABI V2 (-mnocode-object-v3) or V3 (-mcode-object-v3) (AMDGPU only)
-mcode-object-version=<version>	Supported	Specify code object ABI version. Defaults to 4. (AMDGPU only)
-mcrc	Unsupported	Allow use of CRC instructions (ARM/Mips only)
-mcumode	Supported	Specify CU (-mcumode) or WGP (-mno-cumode) wavefront execution mode (AMDGPU only)
-mdouble=<value>	Supported	Force double to be 32 bits or 64 bits
-MD	Supported	Write a depfile containing user and system headers

Option	Support	Description
-meabi <value>	Supported	Set EABI type, e.g. 4, 5 or gnu (default depends on triple)
-membedded-data	Unsupported	Place constants in the .rodata section instead of the .sdata section even if they meet the -G <size> threshold (MIPS)
-menable-experimental-extensions	Unsupported	Enable use of experimental RISC-V extensions.
-mexec-model=<value>	Unsupported	Execution model (WebAssembly only)
-mexecute-only	Unsupported	Disallow generation of data access to code sections (ARM only)
-mextern-sdata	Unsupported	Assume that externally defined data is in the small data if it meets the -G <size> threshold (MIPS)
-mfentry	Unsupported	Insert calls to fentry at function entry (x86/SystemZ only)
-mfix-cortex-a53-835769	Unsupported	Workaround Cortex-A53 erratum 835769 (AArch64 only)
-mfp32	Unsupported	Use 32-bit floating point registers (MIPS only)
-mfp64	Unsupported	Use 64-bit floating point registers (MIPS only)
-MF <file>	Supported	Write depfile output from -MMD, -MD, -MM, or -M to <file>
-mgeneral-regs-only	Unsupported	Generate code which only uses the general purpose registers (AArch64 only)
-mglobal-merge	Supported	Enable merging of globals
-mgpopt	Unsupported	Use GP relative accesses for symbols known to be in a small data section (MIPS)
-MG	Supported	Add missing headers to depfile
-mharden-sls=<value>	Unsupported	Select straight-line speculation hardening scope
-mhvx-length=<value>	Unsupported	Set Hexagon Vector Length
-mhvx=<value>	Unsupported	Enable Hexagon Vector eXtensions
-mhvx	Unsupported	Enable Hexagon Vector eXtensions
-miamcu	Unsupported	Use Intel MCU ABI
--migrate	Unsupported	Run the migrator
-mincremental-linker-compatible	Supported	(integrated-as) Emit an object file that can be used with an incremental linker
-mindirect-jump=<value>	Unsupported	Change indirect jump instructions to inhibit speculation
-Minform=<value>	Supported	Set error level of messages to display
-mios-version-min=<value>	Unsupported	Set iOS deployment target
-MJ <value>	Unsupported	Write a compilation database entry per input
-mllvm <value>	Supported	Additional arguments to forward to LLVM's option processing
-mlocal-sdata	Unsupported	Extend the -G behavior to object local data (MIPS)
-mlong-calls	Supported	Generate branches with extended addressability, usually via indirect jumps.
-mlong-double-128	Supported on Host only	Force long double to be 128 bits

Option	Support	Description
-mlong-double-64	Supported	Force long double to be 64 bits
-mlong-double-80	Supported on Host only	Force long double to be 80 bits, padded to 128 bits for storage
-mlvi-cfi	Supported on Host only	Enable only control-flow mitigations for Load Value Injection (LVI)
-mlvi-hardening	Supported on Host only	Enable all mitigations for Load Value Injection (LVI)
-mmacosx-version-min=<value>	Unsupported	Set Mac OS X deployment target
-mmadd4	Supported	Enable the generation of 4-operand madd.s, madd.d and related instructions.
-mmark-bti-property	Unsupported	Add .note.gnu.property with BTI to assembly files (AArch64 only)
-MMD	Supported	Write a depfile containing user headers
-mmemops	Supported	Enable generation of memop instructions
-mms-bitfields	Unsupported	Set the default structure layout to be compatible with the Microsoft compiler standard
-mmsa	Unsupported	Enable MSA ASE (MIPS only)
-mmt	Unsupported	Enable MT ASE (MIPS only)
-MM	Supported	Like -MMD, but also implies -E and writes to stdout by default
-mno-abicalls	Unsupported	Disable SVR4-style position-independent code (Mips only)
-mno-crc	Unsupported	Disallow use of CRC instructions (Mips only)
-mno-embedded-data	Unsupported	Do not place constants in the .rodata section instead of the .sdata if they meet the -G <size> threshold (MIPS)
-mno-execute-only	Unsupported	Allow generation of data access to code sections (ARM only)
-mno-extern-sdata	Unsupported	Do not assume that externally defined data is in the small data if it meets the -G <size> threshold (MIPS)
-mno-fix-cortex-a53-835769	Unsupported	Don't workaround Cortex-A53 erratum 835769 (AArch64 only)
-mno-global-merge	Supported	Disable merging of globals
-mno-gpopt	Unsupported	Do not use GP relative accesses for symbols known to be in a small data section (MIPS)
-mno-hvx	Unsupported	Disable Hexagon Vector eXtensions
-mno-implicit-float	Supported	Don't generate implicit floating point instructions
-mno-incremental-linker-compatible	Supported	(integrated-as) Emit an object file which cannot be used with an incremental linker
-mno-local-sdata	Unsupported	Do not extend the -G behaviour to object local data (MIPS)
-mno-long-calls	Supported	Restore the default behaviour of not generating long calls
-mno-lvi-cfi	Supported on Host only	Disable control-flow mitigations for Load Value Injection (LVI)
-mno-lvi-hardening	Supported on Host only	Disable mitigations for Load Value Injection (LVI)

Option	Support	Description
-mno-madd4	Supported	Disable the generation of 4-operand madd.s, madd.d and related instructions.
-mno-memops	Supported	Disable generation of memop instructions
-mno-movt	Supported	Disallow use of movt/movw pairs (ARM only)
-mno-ms-bitfields	Supported	Do not set the default structure layout to be compatible with the Microsoft compiler standard
-mno-msa	Unsupported	Disable MSA ASE (MIPS only)
-mno-mt	Unsupported	Disable MT ASE (MIPS only)
-mno-neg-immediates	Supported	Disallow converting instructions with negative immediates to their negation or inversion.
-mno-nvj	Supported	Disable generation of new-value jumps
-mno-nvs	Supported	Disable generation of new-value stores
-mno-outline	Unsupported	Disable function outlining (AArch64 only)
-mno-packets	Supported	Disable generation of instruction packets
-mno-relax	Supported	Disable linker relaxation
-mno-restrict-it	Unsupported	Allow generation of deprecated IT blocks for ARMv8. It is off by default for ARMv8 Thumb mode
-mno-save-restore	Unsupported	Disable using library calls for save and restore
-mno-seses	Unsupported	Disable speculative execution side effect suppression (SESES)
-mno-stack-arg-probe	Supported	Disable stack probes which are enabled by default
-mno-tls-direct-seg-refs	Supported	Disable direct TLS access through segment registers
-mno-unaligned-access	Unsupported	Force all memory accesses to be aligned (AArch32/AArch64 only)
-mno-wavefrontsize64	Supported	Specify wavefront size 32 mode (AMDGPU only)
-mnocrc	Unsupported	Disallow use of CRC instructions (ARM only)
-mnop-mcount	Supported	Generate mcount/__fentry__ calls as nops. To activate they need to be patched in.
-mnvj	Supported	Enable generation of new-value jumps
-mnvs	Supported	Enable generation of new-value stores
-module-dependency-dir <value>	Unsupported	Directory to dump module dependencies to
-module-file-info	Unsupported	Provide information about a particular module file
-momit-leaf-frame-pointer	Supported	Omit frame pointer setup for leaf functions
-moutline	Unsupported	Enable function outlining (AArch64 only)
-mpacked-stack	Unsupported	Use packed stack layout (SystemZ only).
-mpackets	Supported	Enable generation of instruction packets
-mpad-max-prefix-size=<value>	Supported	Specify maximum number of prefixes to use for padding
-mpie-copy-relocations	Supported	Use copy relocations support for PIE builds
-mprefer-vector-width=<value>	Unsupported	Specifies preferred vector width for auto-vectorization. Defaults to 'none' which allows target specific decisions.
-MP	Supported	Create phony target for each dependency (other than main file)
-mqdsp6-compat	Unsupported	Enable hexagon-qdsp6 backward compatibility

Option	Support	Description
-MQ <value>	Supported	Specify name of main file output to quote in depfile
-mrecord-mcount	Supported	Generate a __mcount_loc section entry for each __fentry__ call.
-mrelax-all	Supported	(integrated-as) Relax all machine instructions
-mrelax	Supported	Enable linker relaxation
-mrestrict-it	Unsupported	Disallow generation of deprecated IT blocks for ARMv8. It is on by default for ARMv8 Thumb mode.
-mrtd	Unsupported	Make StdCall calling convention the default
-msave-restore	Unsupported	Enable using library calls for save and restore
-mseses	Unsupported	Enable speculative execution side effect suppression (SESES). Includes LVI control flow integrity mitigations
-msign-return-address=<value>	Unsupported	Select return address signing scope
-msmall-data-limit=<value>	Supported	Put global and static data smaller than the limit into a special section
-msoft-float	Supported	Use software floating point
-msram-ecc	Supported	Legacy option to specify SRAM ECC mode (AMDGPU only). Should use --offload-arch with :sramecc+ instead
-mstack-alignment=<value>	Unsupported	Set the stack alignment
-mstack-arg-probe	Unsupported	Enable stack probes
-mstack-probe-size=<value>	Unsupported	Set the stack probe size
-mstackrealign	Unsupported	Force realign the stack at entry to every function
-msve-vector-bits=<value>	Unsupported	Specify the size in bits of an SVE vector register. Defaults to the vector length agnostic value of "scalable". (AArch64 only)
-msvr4-struct-return	Unsupported	Return small structs in registers (PPC32 only)
-mthread-model <value>	Supported	The thread model to use, e.g. posix, single (posix by default)
-mtls-direct-seg-refs	Supported	Enable direct TLS access through segment registers (default)
-mtls-size=<value>	Unsupported	Specify bit size of immediate TLS offsets (AArch64 ELF only): 12 (for 4KB) | 24 (for 16MB, default) | 32 (for 4GB) | 48 (for 256TB, needs -mcmodel=large)
-mtp=<value>	Unsupported	Thread pointer access method (AArch32/AArch64 only)
-mtune=<value>	Supported on Host only	Only supported on X86. Otherwise accepted for compatibility with GCC.
-MT <value>	Unsupported	Specify name of main file output in depfile
-munaligned-access	Unsupported	Allow memory accesses to be unaligned (AArch32/AArch64 only)
-MV	Supported	Use NMake/Jom format for the depfile
-mwavefrontsize64	Supported	Specify wavefront size 64 mode (AMDGPU only)
-mxnack	Supported	Legacy option to specify XNACK mode (AMDGPU only). Should use --offload-arch with :xnack+ instead

Option	Support	Description
-M	Supported	Like -MD, but also implies -E and writes to stdout by default
--no-cuda-include-ptx=<value>	Supported	Do not include PTX for the following GPU architecture (e.g. sm_35) or 'all'. May be specified more than once.
--no-cuda-version-check	Supported	Don't error out if the detected version of the CUDA install is too low for the requested CUDA gpu architecture.
-no-flang-libs	Supported	Do not link against Flang libraries
--no-offload-arch=<value>	Supported	Remove CUDA/HIP offloading device architecture (e.g. sm_35, gfx906) from the list of devices to compile for. 'all' resets the list to its default value.
--no-system-header-prefix=<prefix>	Supported	Treat all #include paths starting with <prefix> as not including a system header.
-nobuiltininc	Supported	Disable builtin #include directories
-nogpuinc	Supported	Do not add CUDA/HIP include paths and include default CUDA/HIP wrapper header files
-nogpulib	Supported	Do not link device library for CUDA/HIP device compilation
-nostdinc++	Unsupported	Disable standard #include directories for the C++ standard library
-ObjC++	Unsupported	Treat source input files as Objective-C++ inputs
-objcmt-atomic-property	Unsupported	Make migration to 'atomic' properties
-objcmt-migrate-all	Unsupported	Enable migration to modern ObjC
-objcmt-migrate-annotation	Unsupported	Enable migration to property and method annotations
-objcmt-migrate-designated-init	Unsupported	Enable migration to infer NS_DESIGNATED_INITIALIZER for initializer methods
-objcmt-migrate-instancetype	Unsupported	Enable migration to infer instancetype for method result type
-objcmt-migrate-literals	Unsupported	Enable migration to modern ObjC literals
-objcmt-migrate-ns-macros	Unsupported	Enable migration to NS_ENUM/NS_OPTIONS macros
-objcmt-migrate-property-dot-syntax	Unsupported	Enable migration of setter/getter messages to property-dot syntax
-objcmt-migrate-property	Unsupported	Enable migration to modern ObjC property
-objcmt-migrate-protocol-conformance	Unsupported	Enable migration to add protocol conformance on classes
-objcmt-migrate-readonly-property	Unsupported	Enable migration to modern ObjC readonly property
-objcmt-migrate-readwrite-property	Unsupported	Enable migration to modern ObjC readwrite property
-objcmt-migrate-subscripting	Unsupported	Enable migration to modern ObjC subscripting
-objcmt-ns-nonatomic-iosonly	Unsupported	Enable migration to use NS_NONATOMIC_IOSONLY macro for setting property's 'atomic' attribute

Option	Support	Description
-objcmt-returns-innerpointer-property	Unsupported	Enable migration to annotate property with NS_RETURNS_INNER_POINTER
-objcmt-whitelist-dir-path=<value>	Unsupported	Only modify files with a filename contained in the provided directory path
-ObjC	Unsupported	Treat source input files as Objective-C inputs
--offload-arch=<value>	Supported	CUDA offloading device architecture (e.g. sm_35), or HIP offloading target ID in the form of a device architecture followed by target ID features delimited by a colon. Each target ID feature is a pre-defined string followed by a plus or minus sign (e.g. gfx908:xnack+:sramecc-). May be specified more than once.
-o <file>	Supported	Write output to <file>
-parallel-jobs=<value>	Supported	Number of parallel jobs
-pg	Supported	Enable mcount instrumentation
-pipe	Supported	Use pipes between commands, when possible
--precompile	Supported	Only precompile the input
-print-effective-triple	Supported	Print the effective target triple
-print-file-name=<file>	Supported	Print the full library path of <file>
-print-ivar-layout	Unsupported	Enable Objective-C Ivar layout bitmap print trace
-print-libgcc-file-name	Supported	Print the library path for the currently used compiler runtime library ("libgcc.a" or "libclang_rt.builtins.*.a")
-print-prog-name=<name>	Supported	Print the full program path of <name>
-print-resource-dir	Supported	Print the resource directory pathname
-print-search-dirs	Supported	Print the paths used for finding libraries and programs
-print-supported-cpus	Supported	Print supported cpu models for the given target (if target is not specified, it will print the supported cpus for the default target)
-print-target-triple	Supported	Print the normalized target triple
-print-targets	Supported	Print the registered targets
-pthread	Supported	Support POSIX threads in generated code
--ptxas-path=<value>	Unsupported	Path to ptxas (used for compiling CUDA code)
-P	Supported	Disable linemarker output in -E mode
-Qn	Supported	Do not emit metadata containing compiler name and version
-Qunused-arguments	Supported	Don't emit warning for unused driver arguments
-Qy	Supported	Emit metadata containing compiler name and version
-relocatable-pch	Supported	Whether to build a relocatable precompiled header
-rewrite-legacy-objc	Unsupported	Rewrite Legacy Objective-C source to C++
-rewrite-objc	Unsupported	Rewrite Objective-C source to C++
--rocm-device-lib-path=<value>	Supported	ROCm device library path. Alternative to rocm-path.
--rocm-path=<value>	Supported	ROCm installation path, used for finding and automatically linking required bitcode libraries.

Option	Support	Description
-Rpass-analysis=<value>	Supported	Report transformation analysis from optimization passes whose name matches the given POSIX regular expression
-Rpass-missed=<value>	Supported	Report missed transformations by optimization passes whose name matches the given POSIX regular expression
-Rpass=<value>	Supported	Report transformations performed by optimization passes whose name matches the given POSIX regular expression
-rtlib=<value>	Unsupported	Compiler runtime library to use
-R<remark>	Unsupported	Enable the specified remark
-save-stats=<value>	Supported	Save llvm statistics.
-save-stats	Supported	Save llvm statistics.
-save-temps=<value>	Supported	Save intermediate compilation results.
-save-temps	Supported	Save intermediate compilation results
-serialize-diagnostics <value>	Supported	Serialize compiler diagnostics to a file
-shared-libsan	Unsupported	Dynamically link the sanitizer runtime
-static-flang-libs	Supported	Link using static Flang libraries
-static-libsan	Unsupported	Statically link the sanitizer runtime
-static-openmp	Supported	Use the static host OpenMP runtime while linking.
-std=<value>	Supported	Language standard to compile for
-stdlib++-isystem <directory>	Supported	Use directory as the C++ standard library include path
-stdlib=<value>	Supported	C++ standard library to use
-sycl-std=<value>	Unsupported	SYCL language standard to compile for.
--system-header-prefix=<prefix>	Supported	Treat all #include paths starting with <prefix> as including a system header.
-S	Supported	Only run preprocess and compilation steps
--target=<value>	Supported	Generate code for the given target
-Tbss <addr>	Supported	Set starting address of BSS to <addr>
-Tdata <addr>	Supported	Set starting address of DATA to <addr>
-time	Supported	Time individual commands
-traditional-cpp	Unsupported	Enable some traditional CPP emulation
-trigraphs	Supported	Process trigraph sequences
-Ttext <addr>	Supported	Set starting address of TEXT to <addr>
-T <script>	Unsupported	Specify <script> as linker script
-undef	Supported	undef all system defines
-unwindlib=<value>	Supported	Unwind library to use
-U <macro>	Supported	Undefine macro <macro>
--verify-debug-info	Supported	Verify the binary representation of debug output
-verify-pch	Unsupported	Load and verify that a pre-compiled header file is not stale
--version	Supported	Print version information
-v	Supported	Show commands to run and use verbose output

Option	Support	Description
-Wa,<arg>	Supported	Pass the comma separated arguments in <arg> to the assembler
-Wdeprecated	Supported	Enable warnings for deprecated constructs and define __DEPRECATED
-Wl,<arg>	Supported	Pass the comma separated arguments in <arg> to the linker
-working-directory <value>	Supported	Resolve file paths relative to the specified directory
-Wp,<arg>	Supported	Pass the comma separated arguments in <arg> to the preprocessor
-W<warning>	Supported	Enable the specified warning
-w	Supported	Suppress all warnings
-Xanalyzer <arg>	Supported	Pass <arg> to the static analyzer
-Xarch_device <arg>	Supported	Pass <arg> to the CUDA/HIP device compilation
-Xarch_host <arg>	Supported	Pass <arg> to the CUDA/HIP host compilation
-Xassembler <arg>	Supported	Pass <arg> to the assembler
-Xclang <arg>	Supported	Pass <arg> to the clang compiler
-Xcuda-fatbinary <arg>	Supported	Pass <arg> to fatbinary invocation
-Xcuda-ptxas <arg>	Supported	Pass <arg> to the ptxas assembler
-Xlinker <arg>	Supported	Pass <arg> to the linker
-Xopenmp-target=<triple> <arg>	Supported	Pass <arg> to the target offloading toolchain identified by <triple>.
-Xopenmp-target <arg>	Supported	Pass <arg> to the target offloading toolchain.
-Xpreprocessor <arg>	Supported	Pass <arg> to the preprocessor
-x <language>	Supported	Treat subsequent input files as having type <language>
-z <arg>	Supported	Pass -z <arg> to the linker