NAME

AS − the portable GNU assembler.

SYNOPSIS

as [**−a**[**cdghlns**][=*file*]] [**−−alternate**] [**−D**]
[**−−compress−debug−sections**] [**−−nocompress−debug−sections**]
[**−−debug−prefix−map** *old=new*]
[**−−defsym** *sym=val*] [**−f**] [**−g**] [**−−gstabs**]
[**−−gstabs+**] [**−−gdwarf−<N>**] [**−−gdwarf−sections**]
[**−−gdwarf−cie−version**=*VERSION*]
[**−−help**] [**−I** *dir*] [**−J**]
[**−K**] [**−L**] [**−−listing−lhs−width**=*NUM*]
[**−−listing−lhs−width2**=*NUM*] [**−−listing−rhs−width**=*NUM*]
[**−−listing−cont−lines**=*NUM*] [**−−keep−locals**]
[**−−no−pad−sections**]
[**−o** *objfile*] [**−R**]
[**−−statistics**]
[**−v**] [**−version**] [**−−version**]
[**−W**] [**−−warn**] [**−−fatal−warnings**] [**−w**] [**−x**]
[**−Z**] [**@***FILE*]
[**−−sectname−subst**] [**−−size−check=[error|warning]**]
[**−−elf−stt−common=[no|yes]**]
[**−−generate−missing−build−notes=[no|yes]**]
[**−−multibyte−handling=[allow|warn|warn−sym−only]**]
[**−−target−help**] [*target-options*]
[**−−**|*files* ...]

TARGET

Target AArch64 options:
[**−EB**|**−EL**]
[**−mabi**=*ABI*]

Target Alpha options:
[**−m***cpu*]
[**−mdebug** | **−no−mdebug**]
[**−replace** | **−noreplace**]
[**−relax**] [**−g**] [**−G***size*]
[**−F**] [**−32addr**]

Target ARC options:
[**−mcpu=***cpu*]
[**−mA6**|**−mARC600**|**−mARC601**|**−mA7**|**−mARC700**|**−mEM**|**−mHS**]
[**−mcode−density**]
[**−mrelax**]
[**−EB**|**−EL**]

Target ARM options:
[**−mcpu**=*processor*[+*extension*...]]
[**−march**=*architecture*[+*extension*...]]
[**−mfpu**=*floating-point-format*]
[**−mfloat−abi**=*abi*]
[**−meabi**=*ver*]
[**−mthumb**]
[**−EB**|**−EL**]
[**−mapcs−32**|**−mapcs−26**|**−mapcs−float**|
−mapcs−reentrant]
[**−mthumb−interwork**] [**−k**]

Target Blackfin options:
[**−mcpu**=*processor*[*−sirevision*]]
[**−mfdpic**]
[**−mno−fdpic**]
[**−mnopic**]

Target BPF options:
[**−EL**] [**−EB**]

Target CRIS options:
[**−−underscore** | **−−no−underscore**]
[**−−pic**] [**−N**]
[**−−emulation=criself** | **−−emulation=crisaout**]
[**−−march=v0_v10** | **−−march=v10** | **−−march=v32** | **−−march=common_v10_v32**]

Target C−SKY options:
[**−march=***arch*] [**−mcpu=***cpu*]
[**−EL**] [**−mlittle−endian**] [**−EB**] [**−mbig−endian**]
[**−fpic**] [**−pic**]
[**−mljump**] [**−mno−ljump**]
[**−force2bsr**] [**−mforce2bsr**] [**−no−force2bsr**] [**−mno−force2bsr**]
[**−jsri2bsr**] [**−mjsri2bsr**] [**−no−jsri2bsr**] [**−mno−jsri2bsr**]
[**−mnolrw**] [**−mno−lrw**]
[**−melrw**] [**−mno−elrw**]
[**−mlaf**] [**−mliterals−after−func**]
[**−mno−laf**] [**−mno−literals−after−func**]
[**−mlabr**] [**−mliterals−after−br**]
[**−mno−labr**] [**−mnoliterals−after−br**]
[**−mistack**] [**−mno−istack**]
[**−mhard−float**] [**−mmp**] [**−mcp**] [**−mcache**]
[**−msecurity**] [**−mtrust**]
[**−mdsp**] [**−medsp**] [**−mvdsp**]

Target D10V options:
[**−O**]

Target D30V options:
[**−O**|**−n**|**−N**]

Target EPIPHANY options:
[**−mepiphany**|**−mepiphany16**]

Target H8/300 options:
[−h−tick−hex]

Target i386 options:
[**−−32**|**−−x32**|**−−64**] [**−n**]
[**−march**=*CPU*[*+EXTENSION...*]] [**−mtune**=*CPU*]

Target IA−64 options:
[**−mconstant−gp**|**−mauto−pic**]
[**−milp32**|**−milp64**|**−mlp64**|**−mp64**]
[**−mle**|**mbe**]
[**−mtune=itanium1**|**−mtune=itanium2**]
[**−munwind−check=warning**|**−munwind−check=error**]
[**−mhint.b=ok**|**−mhint.b=warning**|**−mhint.b=error**]
[**−x**|**−xexplicit**] [**−xauto**] [**−xdebug**]

Target IP2K options:
[**−mip2022**|**−mip2022ext**]

Target M32C options:
[**−m32c**|**−m16c**] [−relax] [−h−tick−hex]

Target M32R options:
[**−−m32rx**|**−−[no−]warn−explicit−parallel−conflicts**|
−−W[n]p]

Target M680X0 options:
[**−l**] [**−m68000**|**−m68010**|**−m68020**|...]

Target M68HC11 options:
[**−m68hc11**|**−m68hc12**|**−m68hcs12**|**−mm9s12x**|**−mm9s12xg**]
[**−mshort**|**−mlong**]
[**−mshort−double**|**−mlong−double**]
[**−−force−long−branches**] [**−−short−branches**]
[**−−strict−direct−mode**] [**−−print−insn−syntax**]
[**−−print−opcodes**] [**−−generate−example**]

Target MCORE options:
[**−jsri2bsr**] [**−sifilter**] [**−relax**]
[**−mcpu=[210**|**340]**]

Target Meta options:
[**−mcpu=***cpu*] [**−mfpu=***cpu*] [**−mdsp=***cpu*] *Target MICROBLAZE options:*

Target MIPS options:
[**−nocpp**] [**−EL**] [**−EB**] [**−O**[*optimization level*]]
[**−g**[*debug level*]] [**−G** *num*] [**−KPIC**] [**−call_shared**]
[**−non_shared**] [**−xgot** [**−mvxworks−pic**]
[**−mabi**=*ABI*] [**−32**] [**−n32**] [**−64**] [**−mfp32**] [**−mgp32**]
[**−mfp64**] [**−mgp64**] [**−mfpxx**]
[**−modd−spreg**] [**−mno−odd−spreg**]
[**−march**=*CPU*] [**−mtune**=*CPU*] [**−mips1**] [**−mips2**]
[**−mips3**] [**−mips4**] [**−mips5**] [**−mips32**] [**−mips32r2**]
[**−mips32r3**] [**−mips32r5**] [**−mips32r6**] [**−mips64**] [**−mips64r2**]
[**−mips64r3**] [**−mips64r5**] [**−mips64r6**]
[**−construct−floats**] [**−no−construct−floats**]
[**−mignore−branch−isa**] [**−mno−ignore−branch−isa**]
[**−mnan=***encoding*]
[**−trap**] [**−no−break**] [**−break**] [**−no−trap**]
[**−mips16**] [**−no−mips16**]
[**−mmips16e2**] [**−mno−mips16e2**]
[**−mmicromips**] [**−mno−micromips**]
[**−msmartmips**] [**−mno−smartmips**]
[**−mips3d**] [**−no−mips3d**]
[**−mdmx**] [**−no−mdmx**]
[**−mdsp**] [**−mno−dsp**]
[**−mdspr2**] [**−mno−dspr2**]
[**−mdspr3**] [**−mno−dspr3**]
[**−mmsa**] [**−mno−msa**]
[**−mxpa**] [**−mno−xpa**]
[**−mmt**] [**−mno−mt**]
[**−mmcu**] [**−mno−mcu**]
[**−mcrc**] [**−mno−crc**]
[**−mginv**] [**−mno−ginv**]
[**−mloongson−mmi**] [**−mno−loongson−mmi**]
[**−mloongson−cam**] [**−mno−loongson−cam**]
[**−mloongson−ext**] [**−mno−loongson−ext**]

[**−mloongson−ext2**] [**−mno−loongson−ext2**]
[**−minsn32**] [**−mno−insn32**]
[**−mfix7000**] [**−mno−fix7000**]
[**−mfix−rm7000**] [**−mno−fix−rm7000**]
[**−mfix−vr4120**] [**−mno−fix−vr4120**]
[**−mfix−vr4130**] [**−mno−fix−vr4130**]
[**−mfix−r5900**] [**−mno−fix−r5900**]
[**−mdebug**] [**−no−mdebug**]
[**−mpdr**] [**−mno−pdr**]

Target MMIX options:
[**−−fixed−special−register−names**] [**−−globalize−symbols**]
[**−−gnu−syntax**] [**−−relax**] [**−−no−predefined−symbols**]
[**−−no−expand**] [**−−no−merge−gregs**] [**−x**]
[**−−linker−allocated−gregs**]

Target Nios II options:
[**−relax−all**] [**−relax−section**] [**−no−relax**]
[**−EB**] [**−EL**]

Target NDS32 options:
[**−EL**] [**−EB**] [**−O**] [**−Os**] [**−mcpu=***cpu*]
[**−misa=***isa*] [**−mabi=***abi*] [**−mall−ext**]
[**−m[no−]16−bit**] [**−m[no−]perf−ext**] [**−m[no−]perf2−ext**]
[**−m[no−]string−ext**] [**−m[no−]dsp−ext**] [**−m[no−]mac**] [**−m[no−]div**]
[**−m[no−]audio−isa−ext**] [**−m[no−]fpu−sp−ext**] [**−m[no−]fpu−dp−ext**]
[**−m[no−]fpu−fma**] [**−mfpu−freg=***FREG*] [**−mreduced−regs**]
[**−mfull−regs**] [**−m[no−]dx−regs**] [**−mpic**] [**−mno−relax**]
[**−mb2bb**]

Target PDP11 options:
[**−mpic**|**−mno−pic**] [**−mall**] [**−mno−extensions**]
[**−m***extension*|**−mno−***extension*]
[**−m***cpu*] [**−m***machine*]

Target picoJava options:
[**−mb**|**−me**]

Target PowerPC options:
[**−a32**|**−a64**]
[**−mpwrx**|**−mpwr2**|**−mpwr**|**−m601**|**−mppc**|**−mppc32**|**−m603**|**−m604**|**−m403**|**−m405**|
−m440|**−m464**|**−m476**|**−m7400**|**−m7410**|**−m7450**|**−m7455**|**−m750cl**|**−mgekko**|
−mbroadway|**−mppc64**|**−m620**|**−me500**|**−e500x2**|**−me500mc**|**−me500mc64**|**−me5500**|
−me6500|**−mppc64bridge**|**−mbooke**|**−mpower4**|**−mpwr4**|**−mpower5**|**−mpwr5**|**−mpwr5x**|
−mpower6|**−mpwr6**|**−mpower7**|**−mpwr7**|**−mpower8**|**−mpwr8**|**−mpower9**|**−mpwr9−ma2**|
−mcell|**−mspe**|**−mspe2**|**−mtitan**|**−me300**|**−mcom**]
[**−many**] [**−maltivec**|**−mvsx**|**−mhtm**|**−mvle**]
[**−mregnames**|**−mno−regnames**]
[**−mrelocatable**|**−mrelocatable−lib**|**−K PIC**] [**−memb**]
[**−mlittle**|**−mlittle−endian**|**−le**|**−mbig**|**−mbig−endian**|**−be**]
[**−msolaris**|**−mno−solaris**]
[**−nops=***count*]

Target PRU options:
[**−link−relax**]
[**−mnolink−relax**]
[**−mno−warn−regname−label**]

Target RISC-V options:

[**−fpic**|**−fPIC**|**−fno−pic**]
[**−march**=*ISA*]
[**−mabi**=*ABI*]
[**−mlittle−endian**|**−mbig−endian**]

Target RL78 options:
[**−mg10**]
[**−m32bit−doubles**|**−m64bit−doubles**]

Target RX options:
[**−mlittle−endian**|**−mbig−endian**]
[**−m32bit−doubles**|**−m64bit−doubles**]
[**−muse−conventional−section−names**]
[**−msmall−data−limit**]
[**−mpid**]
[**−mrelax**]
[**−mint−register=***number*]
[**−mgcc−abi**|**−mrx−abi**]

Target s390 options:
[**−m31**|**−m64**] [**−mesa**|**−mzarch**] [**−march**=*CPU*]
[**−mregnames**|**−mno−regnames**]
[**−mwarn−areg−zero**]

Target SCORE options:
[**−EB**][**−EL**][**−FIXDD**][**−NWARN**]
[**−SCORE5**][**−SCORE5U**][**−SCORE7**][**−SCORE3**]
[**−march=score7**][**−march=score3**]
[**−USE_R1**][**−KPIC**][**−O0**][**−G** *num*][**−V**]

Target SPARC options:
[**−Av6**|**−Av7**|**−Av8**|**−Aleon**|**−Asparclet**|**−Asparclite**
−Av8plus|**−Av8plusa**|**−Av8plusb**|**−Av8plusc**|**−Av8plusd**
−Av8plusv|**−Av8plusm**|**−Av9**|**−Av9a**|**−Av9b**|**−Av9c**
−Av9d|**−Av9e**|**−Av9v**|**−Av9m**|**−Asparc**|**−Asparcvis**
−Asparcvis2|**−Asparcfmaf**|**−Asparcima**|**−Asparcvis3**
−Asparcvisr|**−Asparc5**]
[**−xarch=v8plus**|**−xarch=v8plusa**]|**−xarch=v8plusb**|**−xarch=v8plusc**
−xarch=v8plusd|**−xarch=v8plusv**|**−xarch=v8plusm**|**−xarch=v9**
−xarch=v9a|**−xarch=v9b**|**−xarch=v9c**|**−xarch=v9d**|**−xarch=v9e**
−xarch=v9v|**−xarch=v9m**|**−xarch=sparc**|**−xarch=sparcvis**
−xarch=sparcvis2|**−xarch=sparcfmaf**|**−xarch=sparcima**
−xarch=sparcvis3|**−xarch=sparcvisr**|**−xarch=sparc5**
−bump]
[**−32**|**−64**]
[**−−enforce−aligned−data**][**−−dcti−couples−detect**]

Target TIC54X options:
[**−mcpu=54[123589]**|**−mcpu=54[56]lp**] [**−mfar−mode**|**−mf**]
[**−merrors−to−file** <*filename*>|**−me** <*filename*>]

Target TIC6X options:
[**−march=***arch*] [**−mbig−endian**|**−mlittle−endian**]
[**−mdsbt**|**−mno−dsbt**] [**−mpid=no**|**−mpid=near**|**−mpid=far**]
[**−mpic**|**−mno−pic**]

Target TILE-Gx options:
[**−m32**|**−m64**][**−EB**][**−EL**]

Target Visium options:

[**−mtune=***arch*]

Target Xtensa options:
[**−−[no−]text−section−literals**] [**−−[no−]auto−litpools**]
[**−−[no−]absolute−literals**]
[**−−[no−]target−align**] [**−−[no−]longcalls**]
[**−−[no−]transform**]
[**−−rename−section** *oldname=newname*]
[**−−[no−]trampolines**]
[**−−abi−windowed**|**−−abi−call0**]

Target Z80 options:
[**−march=***CPU[−EXT][+EXT]*]
[**−local−prefix=***PREFIX*]
[**−colonless**]
[**−sdcc**]
[**−fp−s=***FORMAT*]
[**−fp−d=***FORMAT*]

DESCRIPTION

GNU **as** is really a family of assemblers. If you use (or have used) the GNU assembler on one architecture, you should find a fairly similar environment when you use it on another architecture. Each version has much in common with the others, including object file formats, most assembler directives (often called *pseudo-ops*) and assembler syntax.

as is primarily intended to assemble the output of the GNU C compiler `gcc` for use by the linker `ld`. Nevertheless, we've tried to make **as** assemble correctly everything that other assemblers for the same machine would assemble. Any exceptions are documented explicitly. This doesn't mean **as** always uses the same syntax as another assembler for the same architecture; for example, we know of several incompatible versions of 680x0 assembly language syntax.

Each time you run **as** it assembles exactly one source program. The source program is made up of one or more files. (The standard input is also a file.)

You give **as** a command line that has zero or more input file names. The input files are read (from left file name to right). A command-line argument (in any position) that has no special meaning is taken to be an input file name.

If you give **as** no file names it attempts to read one input file from the **as** standard input, which is normally your terminal. You may have to type **ctl-D** to tell **as** there is no more program to assemble.

Use **−−** if you need to explicitly name the standard input file in your command line.

If the source is empty, **as** produces a small, empty object file.

as may write warnings and error messages to the standard error file (usually your terminal). This should not happen when a compiler runs **as** automatically. Warnings report an assumption made so that **as** could keep assembling a flawed program; errors report a grave problem that stops the assembly.

If you are invoking **as** via the GNU C compiler, you can use the **−Wa** option to pass arguments through to the assembler. The assembler arguments must be separated from each other (and the **−Wa**) by commas. For example:

```
gcc −c −g −O −Wa,−alh,−L file.c
```

This passes two options to the assembler: **−alh** (emit a listing to standard output with high-level and assembly source) and **−L** (retain local symbols in the symbol table).

Usually you do not need to use this **−Wa** mechanism, since many compiler command-line options are automatically passed to the assembler by the compiler. (You can call the GNU compiler driver with the **−v** option to see precisely what options it passes to each compilation pass, including the assembler.)

OPTIONS

@file

Read command-line options from *file*. The options read are inserted in place of the original *@file* option. If*file* does not e xist, or cannot be read, then the option will be treated literally, and not removed.

Options in *file* are separated by whitespace. A whitespace character may be included in an option by surrounding the entire option in either single or double quotes. Any character (including a backslash) may be included by prefixing the character to be included with a backslash. The *file* may itself contain additional *@file* options; any such options will be processed recursively.

−a[cdghlmns]

Turn on listings, in any of a variety of ways:

−ac
omit false conditionals

−ad
omit debugging directives

−ag
include general information, like as version and options passed

−ah
include high-level source

−al include assembly

−am
include macro expansions

−an
omit forms processing

−as include symbols

=file
set the name of the listing file

You may combine these options; for example, use **−aln** for assembly listing without forms processing. The **=file** option, if used, must be the last one. By itself, **−a** defaults to **−ahls**.

−−alternate

Begin in alternate macro mode.

−−compress−debug−sections

Compress DWARF debug sections using zlib with SHF_COMPRESSED from the ELF ABI. The resulting object file may not be compatible with older linkers and object file utilities. Note if compression would make a given section *larger* then it is not compressed.

−−compress−debug−sections=none
−−compress−debug−sections=zlib
−−compress−debug−sections=zlib−gnu
−−compress−debug−sections=zlib−gabi

These options control how DWARF debug sections are compressed. **−−compress−debug−sections=none** is equivalent to **−−nocompress−debug−sections**. **−−compress−debug−sections=zlib** and **−−compress−debug−sections=zlib−gabi** are equivalent to **−−compress−debug−sections**. **−−compress−debug−sections=zlib−gnu** compresses DWARF debug sections using zlib. The debug sections are renamed to begin with **.zdebug**. Note if compression would make a given section *larger* then it is not compressed nor renamed.

−−nocompress−debug−sections

Do not compress DWARF debug sections. This is usually the default for all targets except the x86/x86_64, but a configure time option can be used to override this.

-D Ignored. This option is accepted for script compatibility with calls to other assemblers.

--debug-prefix-map *old=new*
When assembling files in directory *old*, record debugging information describing them as in *new* instead.

--defsym *sym=value*
Define the symbol *sym* to be *value* before assembling the input file. *value* must be an integer constant. As in C, a leading **0x** indicates a hexadecimal value, and a leading **0** indicates an octal value. The value of the symbol can be overridden inside a source file via the use of a `.set` pseudo-op.

-f "fast"---skip whitespace and comment preprocessing (assume source is compiler output).

-g
--gen-debug
Generate debugging information for each assembler source line using whichever debug format is preferred by the target. This currently means either STABS, ECOFF or DWARF2. When the debug format is DWARF then a `.debug_info` and `.debug_line` section is only emitted when the assembly file doesn't generate one itself.

--gstabs
Generate stabs debugging information for each assembler line. This may help debugging assembler code, if the debugger can handle it.

--gstabs+
Generate stabs debugging information for each assembler line, with GNU extensions that probably only gdb can handle, and that could make other debuggers crash or refuse to read your program. This may help debugging assembler code. Currently the only GNU extension is the location of the current working directory at assembling time.

--gdwarf-2
Generate DWARF2 debugging information for each assembler line. This may help debugging assembler code, if the debugger can handle it. Note---this option is only supported by some targets, not all of them.

--gdwarf-3
This option is the same as the **--gdwarf-2** option, except that it allows for the possibility of the generation of extra debug information as per version 3 of the DWARF specification. Note - enabling this option does not guarantee the generation of any extra information, the choice to do so is on a per target basis.

--gdwarf-4
This option is the same as the **--gdwarf-2** option, except that it allows for the possibility of the generation of extra debug information as per version 4 of the DWARF specification. Note - enabling this option does not guarantee the generation of any extra information, the choice to do so is on a per target basis.

--gdwarf-5
This option is the same as the **--gdwarf-2** option, except that it allows for the possibility of the generation of extra debug information as per version 5 of the DWARF specification. Note - enabling this option does not guarantee the generation of any extra information, the choice to do so is on a per target basis.

--gdwarf-sections
Instead of creating a .debug_line section, create a series of .debug_line.*foo* sections where *foo* is the name of the corresponding code section. For example a code section called *.text.func* will have its dwarf line number information placed into a section called *.debug_line.text.func*. If the code section is just called *.text* then debug line section will still be called just *.debug_line* without any suffix.

--gdwarf-cie-version=*version*
Control which version of DWARF Common Information Entries (CIEs) are produced. When this flag is not specificed the default is version 1, though some targets can modify this default. Other possible

values for *version* are 3 or 4.

--size-check=error
--size-check=warning
Issue an error or warning for invalid ELF .size directive.

--elf-stt-common=no
--elf-stt-common=yes
These options control whether the ELF assembler should generate common symbols with the `STT_COMMON` type. The default can be controlled by a configure option **--enable-elf-stt-common**.

--generate-missing-build-notes=yes
--generate-missing-build-notes=no
These options control whether the ELF assembler should generate GNU Build attribute notes if none are present in the input sources. The default can be controlled by the **--enable-generate-build-notes** configure option.

--help
Print a summary of the command-line options and exit.

--target-help
Print a summary of all target specific options and exit.

-I *dir*
Add directory *dir* to the search list for `.include` directives.

-J Don't warn about signed overflow.

-K Issue warnings when difference tables altered for long displacements.

-L
--keep-locals
Keep (in the symbol table) local symbols. These symbols start with system-specific local label prefixes, typically **.L** for ELF systems or **L** for traditional a.out systems.

--listing-lhs-width=*number*
Set the maximum width, in words, of the output data column for an assembler listing to *number*.

--listing-lhs-width2=*number*
Set the maximum width, in words, of the output data column for continuation lines in an assembler listing to *number*.

--listing-rhs-width=*number*
Set the maximum width of an input source line, as displayed in a listing, to *number* bytes.

--listing-cont-lines=*number*
Set the maximum number of lines printed in a listing for a single line of input to *number* + 1.

--multibyte-handling=allow
--multibyte-handling=warn
--multibyte-handling=warn-sym-only
Controls how the assembler handles multibyte characters in the input. The default (which can be restored by using the **allow** argument) is to allow such characters without complaint. Using the **warn** argument will make the assembler generate a warning message whenever any multibyte character is encountered. Using the**war n-sym-only** argument will only cause a warning to be generated when a symbol is defined with a name that contains multibyte characters. (References to undefined symbols will not generate a warning).

--no-pad-sections
Stop the assembler for padding the ends of output sections to the alignment of that section. The default is to pad the sections, but this can waste space which might be needed on targets which have tight memory constraints.

−o *objfile*
Name the object-file output from **as** *objfile*.

−R Fold the data section into the text section.

−−sectname−subst
Honor substitution sequences in section names.

−−statistics
Print the maximum space (in bytes) and total time (in seconds) used by assembly.

−−strip−local−absolute
Remove local absolute symbols from the outgoing symbol table.

−v
−version
Print the **as** version.

−−version
Print the **as** version and exit.

−W
−−no−warn
Suppress warning messages.

−−fatal−warnings
Treat warnings as errors.

−−warn
Don’t suppress warning messages or treat them as errors.

−w Ignored.

−x Ignored.

−Z Generate an object file even after errors.

−− |*files* **...**
Standard input, or source files to assemble.

The following options are available when as is configured for the 64−bit mode of the ARM Architecture (AArch64).

−EB
This option specifies that the output generated by the assembler should be marked as being encoded for a big-endian processor.

−EL
This option specifies that the output generated by the assembler should be marked as being encoded for a little-endian processor.

−mabi=*abi*
Specify which ABI the source code uses. The recognized arguments are: `ilp32` and `lp64`, which decides the generated object file in ELF32 and ELF64 format respectively. The default is `lp64`.

−mcpu=*processor***[+***extension***...]**
This option specifies the target processor. The assembler will issue an error message if an attempt is made to assemble an instruction which will not execute on the target processor. The following processor names are recognized: `cortex−a34`, `cortex−a35`, `cortex−a53`, `cortex−a55`, `cortex−a57`, `cortex−a65`, `cortex−a65ae`, `cortex−a72`, `cortex−a73`, `cortex−a75`, `cortex−a76`, `cortex−a76ae`, `cortex−a77`, `cortex−a78`, `cortex−a78ae`, `cortex−a78c`, `cortex−a510`, `cortex−a710`, `ares`, `exynos−m1`, `falkor`, `neoverse−n1`, `neoverse−n2`, `neoverse−e1`, `neoverse−v1`, `qdf24xx`, `saphira`, `thunderx`, `vulcan`, `xgene1` `xgene2`, `cortex−r82`, `cortex−x1`, and `cortex−x2`. The special name`all` may be used to allow the assembler to accept instructions valid for any supported processor, including all

optional extensions.

In addition to the basic instruction set, the assembler can be told to accept, or restrict, various extension mnemonics that extend the processor.

If some implementations of a particular processor can have an extension, then then those extensions are automatically enabled. Consequently, you will not normally have to specify any additional extensions.

-march=*architecture*[**+***extension*...]
This option specifies the target architecture. The assembler will issue an error message if an attempt is made to assemble an instruction which will not execute on the target architecture. The following architecture names are recognized: `armv8-a`, `armv8.1-a`, `armv8.2-a`, `armv8.3-a`, `armv8.4-a armv8.5-a`, `armv8.6-a`, `armv8.7-a`, `armv8.8-a`, `armv8-r`, `armv9-a`, `armv9.1-a`, `armv9.2-a`, and `armv9.3-a`.

If both **-mcpu** and **-march** are specified, the assembler will use the setting for **-mcpu**. If neither are specified, the assembler will default to **-mcpu=all**.

The architecture option can be extended with the same instruction set extension options as the **-mcpu** option. Unlike **-mcpu**, extensions are not always enabled by default,

-mverbose-error
This option enables verbose error messages for AArch64 gas. This option is enabled by default.

-mno-verbose-error
This option disables verbose error messages in AArch64 gas.

The following options are available when as is configured for an Alpha processor.

-m*cpu*
This option specifies the target processor. If an attempt is made to assemble an instruction which will not execute on the target processor, the assembler may either expand the instruction as a macro or issue an error message. This option is equivalent to the `.arch` directive.

The following processor names are recognized: `21064`, `21064a`, `21066`, `21068`, `21164`, `21164a`, `21164pc`, `21264`, `21264a`, `21264b`, `ev4`, `ev5`, `lca45`, `ev5`, `ev56`, `pca56`, `ev6`, `ev67`, `ev68`. The special name `all` may be used to allow the assembler to accept instructions valid for any Alpha processor.

In order to support existing practice in OSF/1 with respect to `.arch`, and existing practice within **MILO** (the Linux ARC bootloader), the numbered processor names (e.g. 21064) enable the processor-specific PALcode instructions, while the ‘‘electro-vlasic’’ names (e.g. `ev4`) do not.

-mdebug
-no-mdebug
Enables or disables the generation of `.mdebug` encapsulation for stabs directives and procedure descriptors. The default is to automatically enable `.mdebug` when the first stabs directive is seen.

-relax
This option forces all relocations to be put into the object file, instead of saving space and resolving some relocations at assembly time. Note that this option does not propagate all symbol arithmetic into the object file, because not all symbol arithmetic can be represented. However, the option can still be useful in specific applications.

-replace
-noreplace
Enables or disables the optimization of procedure calls, both at assemblage and at link time. These options are only available for VMS targets and `-replace` is the default. See section 1.4.1 of the OpenVMS Linker Utility Manual.

-g This option is used when the compiler generates debug information. When **gcc** is using **mips-tfile** to generate debug information for ECOFF, local labels must be passed through to the object file.

Otherwise this option has no effect.

−G*size*
A local common symbol larger than *size* is placed in `.bss`, while smaller symbols are placed in `.sbss`.

−F
−32addr
These options are ignored for backward compatibility.

The following options are available when as is configured for an ARC processor.

−mcpu=*cpu*
This option selects the core processor variant.

−EB | −EL
Select either big-endian (−EB) or little-endian (−EL) output.

−mcode−density
Enable Code Density extension instructions.

The following options are available when as is configured for the ARM processor family.

−mcpu=*processor***[+***extension***...]**
Specify which ARM processor variant is the target.

−march=*architecture***[+***extension***...]**
Specify which ARM architecture variant is used by the target.

−mfpu=*floating-point-format*
Select which Floating Point architecture is the target.

−mfloat−abi=*abi*
Select which floating point ABI is in use.

−mthumb
Enable Thumb only instruction decoding.

−mapcs−32 | −mapcs−26 | −mapcs−float | −mapcs−reentrant
Select which procedure calling convention is in use.

−EB | −EL
Select either big-endian (−EB) or little-endian (−EL) output.

−mthumb−interwork
Specify that the code has been generated with interworking between Thumb and ARM code in mind.

−mccs
Turns on CodeComposer Studio assembly syntax compatibility mode.

−k Specify that PIC code has been generated.

The following options are available when as is configured for the Blackfin processor family.

−mcpu=*processor*[*−sirevision*]
This option specifies the target processor. The optional*sir evision* is not used in assembler. It's here such that GCC can easily pass down its `−mcpu=` option. The assembler will issue an error message if an attempt is made to assemble an instruction which will not execute on the target processor. The following processor names are recognized: `bf504`, `bf506`, `bf512`, `bf514`, `bf516`, `bf518`, `bf522`, `bf523`, `bf524`, `bf525`, `bf526`, `bf527`, `bf531`, `bf532`, `bf533`, `bf534`, `bf535` (not implemented yet), `bf536`, `bf537`, `bf538`, `bf539`, `bf542`, `bf542m`, `bf544`, `bf544m`, `bf547`, `bf547m`, `bf548`, `bf548m`, `bf549`, `bf549m`, `bf561`, and `bf592`.

−mfdpic
Assemble for the FDPIC ABI.

−mno−fdpic
−mnopic
Disable −mfdpic.

The following options are available when as is configured for the Linux kernel BPF processor family.

`@chapter` BPF Dependent Features

Options

−EB
This option specifies that the assembler should emit big-endian eBPF.

−EL
This option specifies that the assembler should emit little-endian eBPF.

Note that if no endianness option is specified in the command line, the host endianness is used. See the info pages for documentation of the CRIS-specific options.

The following options are available when as is configured for the C−SKY processor family.

−march=*archname*
Assemble for architecture *archname*. The**−−help** option lists v alid values for *archname*.

−mcpu=*cpuname*
Assemble for architecture *cpuname*. The**−−help** option lists v alid values for *cpuname*.

−EL
−mlittle−endian
Generate little-endian output.

−EB
−mbig−endian
Generate big-endian output.

−fpic
−pic
Generate position-independent code.

−mljump
−mno−ljump
Enable/disable transformation of the short branch instructions `jbf`, `jbt`, and `jbr` to `jmpi`. This option is for V2 processors only. It is ignored on CK801 and CK802 targets, which do not support the `jmpi` instruction, and is enabled by default for other processors.

−mbranch−stub
−mno−branch−stub
Pass through `R_CKCORE_PCREL_IMM26BY2` relocations for `bsr` instructions to the linker.

This option is only available for bare-metal C−SKY V2 ELF targets, where it is enabled by default. It cannot be used in code that will be dynamically linked against shared libraries.

−force2bsr
−mforce2bsr
−no−force2bsr
−mno−force2bsr
Enable/disable transformation of `jbsr` instructions to `bsr`. This option is always enabled (and **−mno−force2bsr** is ignored) for CK801/CK802 targets. It is also always enabled when **−mbranch−stub** is in effect.

−jsri2bsr
−mjsri2bsr
−no−jsri2bsr

−mno−jsri2bsr
Enable/disable transformation of `jsri` instructions to `bsr`. This option is enabled by default.

−mnolrw
−mno−lrw
Enable/disable transformation of `lrw` instructions into a `movih`/`ori` pair.

−melrw
−mno−elrw
Enable/disable extended `lrw` instructions. This option is enabled by default for CK800−series processors.

−mlaf
−mliterals−after−func
−mno−laf
−mno−literals−after−func
Enable/disable placement of literal pools after each function.

−mlabr
−mliterals−after−br
−mno−labr
−mnoliterals−after−br
Enable/disable placement of literal pools after unconditional branches. This option is enabled by default.

−mistack
−mno−istack
Enable/disable interrupt stack instructions. This option is enabled by default on CK801, CK802, and CK802 processors.

The following options explicitly enable certain optional instructions. These features are also enabled implicitly by using `−mcpu=` to specify a processor that supports it.

−mhard−float
Enable hard float instructions.

−mmp
Enable multiprocessor instructions.

−mcp
Enable coprocessor instructions.

−mcache
Enable cache prefetch instruction.

−msecurity
Enable C−SKY security instructions.

−mtrust
Enable C−SKY trust instructions.

−mdsp
Enable DSP instructions.

−medsp
Enable enhanced DSP instructions.

−mvdsp
Enable vector DSP instructions.

The following options are available when as is configured for an Epiphany processor.

−mepiphany
Specifies that the both 32 and 16 bit instructions are allowed. This is the default behavior.

−mepiphany16
Restricts the permitted instructions to just the 16 bit set.

The following options are available when as is configured for an H8/300 processor. `@chapter` H8/300 Dependent Features

Options
The Renesas H8/300 version of `as` has one machine-dependent option:

−h−tick−hex
Support H'00 style hex constants in addition to 0x00 style.

−mach=*name*
Sets the H8300 machine variant. The following machine names are recognised: `h8300h`, `h8300hn`, `h8300s`, `h8300sn`, `h8300sx` and `h8300sxn`.

The following options are available when as is configured for an i386 processor.

−−32 | −−x32 | −−64
Select the word size, either 32 bits or 64 bits. **−−32** implies Intel i386 architecture, while **−−x32** and **−−64** imply AMD x86−64 architecture with 32−bit or 64−bit word-size respectively.

These options are only available with the ELF object file format, and require that the necessary BFD support has been included (on a 32−bit platform you have to add −−enable−64−bit−bfd to configure enable 64−bit usage and use x86−64 as target platform).

−n By default, x86 GAS replaces multiple nop instructions used for alignment within code sections with multi-byte nop instructions such as leal 0(%esi,1),%esi. This switch disables the optimization if a single byte nop (0x90) is explicitly specified as the fill byte for alignment.

−−divide
On SVR4−derived platforms, the character / is treated as a comment character, which means that it cannot be used in expressions. The**−−di vide** option turns / into a normal character. This does not disable / at the beginning of a line starting a comment, or affect using # for starting a comment.

−march=*CPU***[+***EXTENSION***...]**
This option specifies the target processor. The assembler will issue an error message if an attempt is made to assemble an instruction which will not execute on the target processor. The following processor names are recognized: `i8086`, `i186`, `i286`, `i386`, `i486`, `i586`, `i686`, `pentium`, `pentiumpro`, `pentiumii`, `pentiumiii`, `pentium4`, `prescott`, `nocona`, `core`, `core2`, `corei7`, `l1om`, `k1om`, `iamcu`, `k6`, `k6_2`, `athlon`, `opteron`, `k8`, `amdfam10`, `bdver1`, `bdver2`, `bdver3`, `bdver4`, `znver1`, `znver2`, `znver3`, `btver1`, `btver2`, `generic32` and `generic64`.

In addition to the basic instruction set, the assembler can be told to accept various extension mnemonics. For example, `−march=i686+sse4+vmx` extends *i686* with *sse4* and *vmx*. The following extensions are currently supported: `8087`, `287`, `387`, `687`, `no87`, `no287`, `no387`, `no687`, `cmov`, `nocmov`, `fxsr`, `nofxsr`, `mmx`, `nommx`, `sse`, `sse2`, `sse3`, `sse4a`, `ssse3`, `sse4.1`, `sse4.2`, `sse4`, `nosse`, `nosse2`, `nosse3`, `nosse4a`, `nossse3`, `nosse4.1`, `nosse4.2`, `nosse4`, `avx`, `avx2`, `noavx`, `noavx2`, `adx`, `rdseed`, `prfchw`, `smap`, `mpx`, `sha`, `rdpid`, `ptwrite`, `cet`, `gfni`, `vaes`, `vpclmulqdq`, `prefetchwt1`, `clflushopt`, `se1`, `clwb`, `movdiri`, `movdir64b`, `enqcmd`, `serialize`, `tsxldtrk`, `kl`, `nokl`, `widekl`, `nowidekl`, `hreset`, `avx512f`, `avx512cd`, `avx512er`, `avx512pf`, `avx512vl`, `avx512bw`, `avx512dq`, `avx512ifma`, `avx512vbmi`, `avx512_4fmaps`, `avx512_4vnniw`, `avx512_vpopcntdq`, `avx512_vbmi2`, `avx512_vnni`, `avx512_bitalg`, `avx512_vp2intersect`, `tdx`, `avx512_bf16`, `avx_vnni`, `avx512_fp16`, `noavx512f`, `noavx512cd`, `noavx512er`, `noavx512pf`, `noavx512vl`, `noavx512bw`, `noavx512dq`, `noavx512ifma`, `noavx512vbmi`, `noavx512_4fmaps`, `noavx512_4vnniw`, `noavx512_vpopcntdq`, `noavx512_vbmi2`, `noavx512_vnni`, `noavx512_bitalg`, `noavx512_vp2intersect`, `notdx`, `noavx512_bf16`, `noavx_vnni`, `noavx512_fp16`, `noenqcmd`, `noserialize`, `notsxldtrk`, `amx_int8`, `noamx_int8`, `amx_bf16`,

`noamx_bf16`, `amx_tile`, `noamx_tile`, `nouintr`, `nohreset`, `vmx`, `vmfunc`, `smx`, `xsave`, `xsaveopt`, `xsavec`, `xsaves`, `aes`, `pclmul`, `fsgsbase`, `rdrnd`, `f16c`, `bmi2`, `fma`, `movbe`, `ept`, `lzcnt`, `popcnt`, `hle`, `rtm`, `invpcid`, `clflush`, `mwaitx`, `clzero`, `wbnoinvd`, `pconfig`, `waitpkg`, `uintr`, `cldemote`, `rdpru`, `mcommit`, `sev_es`, `lwp`, `fma4`, `xop`, `cx16`, `syscall`, `rdtscp`, `3dnow`, `3dnowa`, `sse4a`, `sse5`, `snp`, `invlpgb`, `tlbsync`, `svme` and `padlock`. Note that rather than extending a basic instruction set, the extension mnemonics starting with `no` revoke the respective functionality.

When the `.arch` directive is used with **−march**, the `.arch` directive will take precedent.

−mtune=*CPU*
This option specifies a processor to optimize for. When used in conjunction with the **−march** option, only instructions of the processor specified by the **−march** option will be generated.

Valid *CPU* values are identical to the processor list of **−march=***CPU*.

−msse2avx
This option specifies that the assembler should encode SSE instructions with VEX prefix.

−muse−unaligned−vector−move
This option specifies that the assembler should encode aligned vector move as unaligned vector move.

−msse−check=*none*
−msse−check=*warning*
−msse−check=*error*
These options control if the assembler should check SSE instructions. **−msse−check=***none* will mak e the assembler not to check SSE instructions, which is the default. **−msse−check=***warning* will make the assembler issue a warning for any SSE instruction. **−msse−check=***err or* will make the assembler issue an error for any SSE instruction.

−mavxscalar=*128*
−mavxscalar=*256*
These options control how the assembler should encode scalar AVX instructions. **−ma vxscalar=***128* will encode scalar AVX instructions with 128bit vector length, which is the default. **−mavxscalar=***256* will encode scalar AVX instructions with 256bit vector length.

WARNING: Don't use this for production code − due to CPU errata the resulting code may not work on certain models.

−mvexwig=*0*
−mvexwig=*1*
These options control how the assembler should encode VEX.W−ignored (WIG) VEX instructions. **−mvexwig=***0* will encode WIG VEX instructions with vex.w = 0, which is the default. **−mvexwig=***1* will encode WIG EVEX instructions with vex.w = 1.

WARNING: Don't use this for production code − due to CPU errata the resulting code may not work on certain models.

−mevexlig=*128*
−mevexlig=*256*
−mevexlig=*512*
These options control how the assembler should encode length-ignored (LIG) EVEX instructions. **−mevexlig=***128* will encode LIG EVEX instructions with 128bit vector length, which is the default. **−mevexlig=***256* and **−mevexlig=***512* will encode LIG EVEX instructions with 256bit and 512bit vector length, respectively.

−mevexwig=*0*
−mevexwig=*1*
These options control how the assembler should encode w−ignored (WIG) EVEX instructions. **−mevexwig=***0* will encode WIG EVEX instructions with evex.w = 0, which is the default. **−mevexwig=***1* will encode WIG EVEX instructions with evex.w = 1.

−mmnemonic=*att*
−mmnemonic=*intel*
This option specifies instruction mnemonic for matching instructions. The `.att_mnemonic` and `.intel_mnemonic` directives will take precedent.

−msyntax=*att*
−msyntax=*intel*
This option specifies instruction syntax when processing instructions. The `.att_syntax` and `.intel_syntax` directives will take precedent.

−mnaked−reg
This option specifies that registers don't require a **%** prefix. The `.att_syntax` and `.intel_syntax` directives will take precedent.

−madd−bnd−prefix
This option forces the assembler to add BND prefix to all branches, even if such prefix was not explicitly specified in the source code.

−mno−shared
On ELF target, the assembler normally optimizes out non-PLT relocations against defined non-weak global branch targets with default visibility. The**−mshar ed** option tells the assembler to generate code which may go into a shared library where all non-weak global branch targets with default visibility can be preempted. The resulting code is slightly bigger. This option only affects the handling of branch instructions.

−mbig−obj
On PE/COFF target this option forces the use of big object file format, which allows more than 32768 sections.

−momit−lock−prefix=*no*
−momit−lock−prefix=*yes*
These options control how the assembler should encode lock prefix. This option is intended as a workaround for processors, that fail on lock prefix. This option can only be safely used with single-core, single-thread computers **−momit−lock−prefix=***yes* will omit all lock prefixes. **−momit−lock−prefix=***no* will encode lock prefix as usual, which is the default.

−mfence−as−lock−add=*no*
−mfence−as−lock−add=*yes*
These options control how the assembler should encode lfence, mfence and sfence. **−mfence−as−lock−add=***yes* will encode lfence, mfence and sfence as **lock addl `$0x0`, (%rsp)** in 64−bit mode and **lock addl `$0x0`, (%esp)** in 32−bit mode. **−mfence−as−lock−add=***no* will encode lfence, mfence and sfence as usual, which is the default.

−mrelax−relocations=*no*
−mrelax−relocations=*yes*
These options control whether the assembler should generate relax relocations, R_386_GOT32X, in 32−bit mode, or R_X86_64_GOTPCRELX and R_X86_64_REX_GOTPCRELX, in 64−bit mode. **−mrelax−relocations=***yes* will generate relax relocations. **−mrelax−relocations=***no* will not generate relax relocations. The default can be controlled by a configure option **−−enable−x86−relax−relocations**.

−malign−branch−boundary=*NUM*
This option controls how the assembler should align branches with segment prefixes or NOP. *NUM* must be a power of 2. It should be 0 or no less than 16. Branches will be aligned within *NUM* byte boundary. **−malign−branch−boundary=0**, which is the default, doesn't align branches.

−malign−branch=*TYPE***[+***TYPE***...]**
This option specifies types of branches to align. *TYPE* is combination of **jcc**, which aligns conditional jumps, **fused**, which aligns fused conditional jumps, **jmp**, which aligns unconditional jumps, **call** which aligns calls, **ret**, which aligns rets, **indirect**, which aligns indirect jumps and calls. The default

is **−malign−branch=jcc+fused+jmp**.

−malign−branch−prefix−size=*NUM*
This option specifies the maximum number of prefixes on an instruction to align branches. *NUM* should be between 0 and 5. The default *NUM* is 5.

−mbranches−within−32B−boundaries
This option aligns conditional jumps, fused conditional jumps and unconditional jumps within 32 byte boundary with up to 5 segment prefixes on an instruction. It is equivalent to **−malign−branch−boundary=32 −malign−branch=jcc+fused+jmp −malign−branch−prefix−size=5**. The default doesn't align branches.

−mlfence−after−load=*no*
−mlfence−after−load=*yes*
These options control whether the assembler should generate lfence after load instructions. **−mlfence−after−load=***yes* will generate lfence. **−mlfence−after−load=***no* will not generate lfence, which is the default.

−mlfence−before−indirect−branch=*none*
−mlfence−before−indirect−branch=*all*
−mlfence−before−indirect−branch=*register*
−mlfence−before−indirect−branch=*memory*
These options control whether the assembler should generate lfence before indirect near branch instructions. **−mlfence−before−indirect−branch=***all* will generate lfence before indirect near branch via register and issue a warning before indirect near branch via memory. It also implicitly sets **−mlfence−before−ret=***shl* when there's no explicit **−mlfence−before−ret=**. **−mlfence−before−indirect−branch=***register* will generate lfence before indirect near branch via register. **−mlfence−before−indirect−branch=***memory* will issue a warning before indirect near branch via memory. **−mlfence−before−indirect−branch=***none* will not generate lfence nor issue warning, which is the default. Note that lfence won't be generated before indirect near branch via register with **−mlfence−after−load=***yes* since lfence will be generated after loading branch target register.

−mlfence−before−ret=*none*
−mlfence−before−ret=*shl*
−mlfence−before−ret=*or*
−mlfence−before−ret=*yes*
−mlfence−before−ret=*not*
These options control whether the assembler should generate lfence before ret. **−mlfence−before−ret=***or* will generate generate or instruction with lfence. **−mlfence−before−ret=***shl/yes* will generate shl instruction with lfence. **−mlfence−before−ret=***not* will generate not instruction with lfence. **−mlfence−before−ret=***none* will not generate lfence, which is the default.

−mx86−used−note=*no*
−mx86−used−note=*yes*
These options control whether the assembler should generate GNU_PROPERTY_X86_ISA_1_USED and GNU_PROPERTY_X86_FEATURE_2_USED GNU property notes. The default can be controlled by the **−−enable−x86−used−note** configure option.

−mevexrcig=*rne*
−mevexrcig=*rd*
−mevexrcig=*ru*
−mevexrcig=*rz*
These options control how the assembler should encode SAE-only EVEX instructions. **−mevexrcig=***rne* will encode RC bits of EVEX instruction with 00, which is the default. **−mevexrcig=***rd*, **−mevexrcig=***ru* and **−mevexrcig=***rz* will encode SAE-only EVEX instructions with 01, 10 and 11 RC bits, respectively.

−mamd64
−mintel64
This option specifies that the assembler should accept only AMD64 or Intel64 ISA in 64−bit mode. The default is to accept common, Intel64 only and AMD64 ISAs.

−O0 | −O | −O1 | −O2 | −Os
Optimize instruction encoding with smaller instruction size. **−O** and **−O1** encode 64−bit re gister load instructions with 64−bit immediate as 32−bit register load instructions with 31−bit or 32−bits immediates, encode 64−bit register clearing instructions with 32−bit register clearing instructions, encode 256−bit/512−bit VEX/EVEX vector register clearing instructions with 128−bit VEX vector register clearing instructions, encode 128−bit/256−bit EVEX vector register load/store instructions with VEX vector register load/store instructions, and encode 128−bit/256−bit EVEX packed integer logical instructions with 128−bit/256−bit VEX packed integer logical.

−O2 includes **−O1** optimization plus encodes 256−bit/512−bit EVEX vector register clearing instructions with 128−bit EVEX vector register clearing instructions. In 64−bit mode VEX encoded instructions with commutative source operands will also have their source operands swapped if this allows using the 2−byte VEX prefix form instead of the 3−byte one. Certain forms of AND as well as OR with the same (register) operand specified twice will also be changed to TEST.

−Os includes **−O2** optimization plus encodes 16−bit, 32−bit and 64−bit register tests with immediate as 8−bit register test with immediate. **−O0** turns off this optimization.

The following options are available when as is configured for the Ubicom IP2K series.

−mip2022ext
Specifies that the extended IP2022 instructions are allowed.

−mip2022
Restores the default behaviour, which restricts the permitted instructions to just the basic IP2022 ones.

The following options are available when as is configured for the Renesas M32C and M16C processors.

−m32c
Assemble M32C instructions.

−m16c
Assemble M16C instructions (the default).

−relax
Enable support for link-time relaxations.

−h−tick−hex
Support H'00 style hex constants in addition to 0x00 style.

The following options are available when as is configured for the Renesas M32R (formerly Mitsubishi M32R) series.

−−m32rx
Specify which processor in the M32R family is the target. The default is normally the M32R, but this option changes it to the M32RX.

−−warn−explicit−parallel−conflicts or −−Wp
Produce warning messages when questionable parallel constructs are encountered.

−−no−warn−explicit−parallel−conflicts or −−Wnp
Do not produce warning messages when questionable parallel constructs are encountered.

The following options are available when as is configured for the Motorola 68000 series.

−l Shorten references to undefined symbols, to one word instead of two.

−m68000 | −m68008 | −m68010 | −m68020 | −m68030

| −m68040 | −m68060 | −m68302 | −m68331 | −m68332
| −m68333 | −m68340 | −mcpu32 | −m5200
Specify what processor in the 68000 family is the target. The default is normally the 68020, but this can be changed at configuration time.

−m68881 | −m68882 | −mno−68881 | −mno−68882
The target machine does (or does not) have a floating-point coprocessor. The default is to assume a coprocessor for 68020, 68030, and cpu32. Although the basic 68000 is not compatible with the 68881, a combination of the two can be specified, since it's possible to do emulation of the coprocessor instructions with the main processor.

−m68851 | −mno−68851
The target machine does (or does not) have a memory-management unit coprocessor. The default is to assume an MMU for 68020 and up.

The following options are available when as is configured for an Altera Nios II processor.

−relax−section
Replace identified out-of-range branches with PC-relative `jmp` sequences when possible. The generated code sequences are suitable for use in position-independent code, but there is a practical limit on the extended branch range because of the length of the sequences. This option is the default.

−relax−all
Replace branch instructions not determinable to be in range and all call instructions with `jmp` and `callr` sequences (respectively). This option generates absolute relocations against the target symbols and is not appropriate for position-independent code.

−no−relax
Do not replace any branches or calls.

−EB
Generate big-endian output.

−EL
Generate little-endian output. This is the default.

−march=*architecture*
This option specifies the target architecture. The assembler issues an error message if an attempt is made to assemble an instruction which will not execute on the target architecture. The following architecture names are recognized: `r1`, `r2`. The default is `r1`.

The following options are available when as is configured for a PRU processor.

−mlink−relax
Assume that LD would optimize LDI32 instructions by checking the upper 16 bits of the *expression*. If they are all zeros, then LD would shorten the LDI32 instruction to a single LDI. In such case `as` will output DIFF relocations for diff expressions.

−mno−link−relax
Assume that LD would not optimize LDI32 instructions. As a consequence, DIFF relocations will not be emitted.

−mno−warn−regname−label
Do not warn if a label name matches a register name. Usually assembler programmers will want this warning to be emitted. C compilers may want to turn this off.

The following options are available when as is configured for a MIPS processor.

−G *num*
This option sets the largest size of an object that can be referenced implicitly with the `gp` register. It is only accepted for targets that use ECOFF format, such as a DECstation running Ultrix. The default value is 8.

−EB
Generate ‘‘big endian’’ format output.

−EL
Generate ‘‘little endian’’ format output.

−mips1
−mips2
−mips3
−mips4
−mips5
−mips32
−mips32r2
−mips32r3
−mips32r5
−mips32r6
−mips64
−mips64r2
−mips64r3
−mips64r5
−mips64r6
Generate code for a particular MIPS Instruction Set Architecture level. **−mips1** is an alias for **−march=r3000**, **−mips2** is an alias for **−march=r6000**, **−mips3** is an alias for **−march=r4000** and **−mips4** is an alias for **−march=r8000**. **−mips5**, **−mips32**, **−mips32r2**, **−mips32r3**, **−mips32r5**, **−mips32r6**, **−mips64**, **−mips64r2**, **−mips64r3**, **−mips64r5**, and **−mips64r6** correspond to generic MIPS V, MIPS32, MIPS32 Release 2, MIPS32 Release 3, MIPS32 Release 5, MIPS32 Release 6, MIPS64, MIPS64 Release 2, MIPS64 Release 3, MIPS64 Release 5, and MIPS64 Release 6 ISA processors, respectively.

−march=*cpu*
Generate code for a particular MIPS CPU.

−mtune=*cpu*
Schedule and tune for a particular MIPS CPU.

−mfix7000
−mno−fix7000
Cause nops to be inserted if the read of the destination register of an mfhi or mflo instruction occurs in the following two instructions.

−mfix−rm7000
−mno−fix−rm7000
Cause nops to be inserted if a dmult or dmultu instruction is followed by a load instruction.

−mfix−r5900
−mno−fix−r5900
Do not attempt to schedule the preceding instruction into the delay slot of a branch instruction placed at the end of a short loop of six instructions or fewer and always schedule a `nop` instruction there instead. The short loop bug under certain conditions causes loops to execute only once or twice, due to a hardware bug in the R5900 chip.

−mdebug
−no−mdebug
Cause stabs-style debugging output to go into an ECOFF-style .mdebug section instead of the standard ELF .stabs sections.

−mpdr
−mno−pdr
Control generation of `.pdr` sections.

−mgp32
−mfp32
The register sizes are normally inferred from the ISA and ABI, but these flags force a certain group of registers to be treated as 32 bits wide at all times. **−mgp32** controls the size of general-purpose registers and **−mfp32** controls the size of floating-point registers.

−mgp64
−mfp64
The register sizes are normally inferred from the ISA and ABI, but these flags force a certain group of registers to be treated as 64 bits wide at all times. **−mgp64** controls the size of general-purpose registers and **−mfp64** controls the size of floating-point registers.

−mfpxx
The register sizes are normally inferred from the ISA and ABI, but using this flag in combination with **−mabi=32** enables an ABI variant which will operate correctly with floating-point registers which are 32 or 64 bits wide.

−modd−spreg
−mno−odd−spreg
Enable use of floating-point operations on odd-numbered single-precision registers when supported by the ISA. **−mfpxx**implies **−mno−odd−spr eg**, otherwise the default is **−modd−spreg**.

−mips16
−no−mips16
Generate code for the MIPS 16 processor. This is equivalent to putting `.module mips16` at the start of the assembly file. **−no−mips16** turns off this option.

−mmips16e2
−mno−mips16e2
Enable the use of MIPS16e2 instructions in MIPS16 mode. This is equivalent to putting `.module mips16e2` at the start of the assembly file. **−mno−mips16e2** turns off this option.

−mmicromips
−mno−micromips
Generate code for the microMIPS processor. This is equivalent to putting `.module micromips` at the start of the assembly file. **−mno−micr omips** turns off this option. This is equivalent to putting `.module nomicromips`at the start of the assembly file.

−msmartmips
−mno−smartmips
Enables the SmartMIPS extension to the MIPS32 instruction set. This is equivalent to putting `.module smartmips`at the start of the assembly file. **−mno−smartmips** turns of f this option.

−mips3d
−no−mips3d
Generate code for the MIPS−3D Application Specific Extension. This tells the assembler to accept MIPS−3D instructions. **−no−mips3d** turns off this option.

−mdmx
−no−mdmx
Generate code for the MDMX Application Specific Extension. This tells the assembler to accept MDMX instructions. **−no−mdmx** turns off this option.

−mdsp
−mno−dsp
Generate code for the DSP Release 1 Application Specific Extension. This tells the assembler to accept DSP Release 1 instructions. **−mno−dsp** turns off this option.

−mdspr2

−mno−dspr2
Generate code for the DSP Release 2 Application Specific Extension. This option implies **−mdsp**. This tells the assembler to accept DSP Release 2 instructions. **−mno−dspr2** turns off this option.

−mdspr3
−mno−dspr3
Generate code for the DSP Release 3 Application Specific Extension. This option implies **−mdsp** and **−mdspr2**. This tells the assembler to accept DSP Release 3 instructions. **−mno−dspr3** turns of f this option.

−mmsa
−mno−msa
Generate code for the MIPS SIMD Architecture Extension. This tells the assembler to accept MSA instructions. **−mno−msa** turns off this option.

−mxpa
−mno−xpa
Generate code for the MIPS eXtended Physical Address (XPA) Extension. This tells the assembler to accept XPA instructions. **−mno−xpa** turns off this option.

−mmt
−mno−mt
Generate code for the MT Application Specific Extension. This tells the assembler to accept MT instructions. **−mno−mt** turns off this option.

−mmcu
−mno−mcu
Generate code for the MCU Application Specific Extension. This tells the assembler to accept MCU instructions. **−mno−mcu** turns off this option.

−mcrc
−mno−crc
Generate code for the MIPS cyclic redundancy check (CRC) Application Specific Extension. This tells the assembler to accept CRC instructions. **−mno−crc** turns off this option.

−mginv
−mno−ginv
Generate code for the Global INValidate (GINV) Application Specific Extension. This tells the assembler to accept GINV instructions. **−mno−ginv** turns off this option.

−mloongson−mmi
−mno−loongson−mmi
Generate code for the Loongson MultiMedia extensions Instructions (MMI) Application Specific Extension. This tells the assembler to accept MMI instructions. **−mno−loongson−mmi** turns of f this option.

−mloongson−cam
−mno−loongson−cam
Generate code for the Loongson Content Address Memory (CAM) instructions. This tells the assembler to accept Loongson CAM instructions. **−mno−loongson−cam** turns off this option.

−mloongson−ext
−mno−loongson−ext
Generate code for the Loongson EXTensions (EXT) instructions. This tells the assembler to accept Loongson EXT instructions. **−mno−loongson−ext** turns off this option.

−mloongson−ext2
−mno−loongson−ext2
Generate code for the Loongson EXTensions R2 (EXT2) instructions. This option implies **−mloongson−ext**. This tells the assembler to accept Loongson EXT2 instructions. **−mno−loongson−ext2** turns off this option.

−minsn32
−mno−insn32
Only use 32−bit instruction encodings when generating code for the microMIPS processor. This option inhibits the use of any 16−bit instructions. This is equivalent to putting `.set insn32` at the start of the assembly file. **−mno−insn32** turns of f this option. This is equivalent to putting `.set noinsn32` at the start of the assembly file. By default **−mno−insn32** is selected, allowing all instructions to be used.

−−construct−floats
−−no−construct−floats
The **−−no−construct−floats** option disables the construction of double width floating point constants by loading the two halves of the value into the two single width floating point registers that make up the double width register. By default **−−construct−floats** is selected, allowing construction of these floating point constants.

−−relax−branch
−−no−relax−branch
The **−−relax−branch** option enables the relaxation of out-of-range branches. By default **−−no−relax−branch** is selected, causing any out-of-range branches to produce an error.

−mignore−branch−isa
−mno−ignore−branch−isa
Ignore branch checks for invalid transitions between ISA modes. The semantics of branches does not provide for an ISA mode switch, so in most cases the ISA mode a branch has been encoded for has to be the same as the ISA mode of the branch's target label. Therefore GAS has checks implemented that verify in branch assembly that the two ISA modes match. **−mignor e−branch−isa** disables these checks. By default **−mno−ignore−branch−isa** is selected, causing any invalid branch requiring a transition between ISA modes to produce an error.

−mnan=*encoding*
Select between the IEEE 754−2008 (**−mnan=2008**) or the legacy (**−mnan=legacy**) NaN encoding format. The latter is the default.

−−emulation=*name*
This option was formerly used to switch between ELF and ECOFF output on targets like IRIX 5 that supported both. MIPS ECOFF support was removed in GAS 2.24, so the option now serves little purpose. It is retained for backwards compatibility.

The available configuration names are: **mipself**, **mipslelf** and **mipsbelf**. Choosing**mipself** no w has no effect, since the output is always ELF. **mipslelf**and **mipsbelf** select little− and big-endian output respectively, but **−EL** and **−EB** are now the preferred options instead.

−nocpp
as ignores this option. It is accepted for compatibility with the native tools.

−−trap
−−no−trap
−−break
−−no−break
Control how to deal with multiplication overflow and division by zero. **−−trap** or **−−no−br eak** (which are synonyms) take a trap exception (and only work for Instruction Set Architecture level 2 and higher); **−−break** or **−−no−trap** (also synonyms, and the default) take a break exception.

−n When this option is used, **as** will issue a warning every time it generates a nop instruction from a macro.

The following options are available when as is configured for a LoongArch processor.

−fpic

-fPIC
Generate position-independent code

-fno-pic
Don't generate position-independent code (default)

The following options are available when as is configured for a Meta processor.

`-mcpu=metac11`
Generate code for Meta 1.1.

`-mcpu=metac12`
Generate code for Meta 1.2.

`-mcpu=metac21`
Generate code for Meta 2.1.

`-mfpu=metac21`
Allow code to use FPU hardware of Meta 2.1.

See the info pages for documentation of the MMIX-specific options.

The following options are available when as is configured for a NDS32 processor.

`-O1`
Optimize for performance.

`-Os`
Optimize for space.

`-EL`
Produce little endian data output.

`-EB`
Produce little endian data output.

`-mpic`
Generate PIC.

`-mno-fp-as-gp-relax`
Suppress fp-as-gp relaxation for this file.

`-mb2bb-relax`
Back-to-back branch optimization.

`-mno-all-relax`
Suppress all relaxation for this file.

`-march=<arch name>`
Assemble for architecture <arch name> which could be v3, v3j, v3m, v3f, v3s, v2, v2j, v2f, v2s.

`-mbaseline=<baseline>`
Assemble for baseline <baseline> which could be v2, v3, v3m.

`-mfpu-freg=`*`FREG`*
Specify a FPU configuration.

```
0      8 SP /  4 DP registers
1     16 SP /  8 DP registers
2     32 SP / 16 DP registers
3     32 SP / 32 DP registers
```

`-mabi=`*`abi`*
Specify a abi version <abi> could be v1, v2, v2fp, v2fpp.

`-m[no-]mac`
Enable/Disable Multiply instructions support.

`-m[no-]div`
Enable/Disable Divide instructions support.

`-m[no-]16bit-ext`
Enable/Disable 16-bit extension

`-m[no-]dx-regs`
Enable/Disable d0/d1 registers

`-m[no-]perf-ext`
Enable/Disable Performance extension

`-m[no-]perf2-ext`
Enable/Disable Performance extension 2

`-m[no-]string-ext`
Enable/Disable String extension

`-m[no-]reduced-regs`
Enable/Disable Reduced Register configuration (GPR16) option

`-m[no-]audio-isa-ext`
Enable/Disable AUDIO ISA extension

`-m[no-]fpu-sp-ext`
Enable/Disable FPU SP extension

`-m[no-]fpu-dp-ext`
Enable/Disable FPU DP extension

`-m[no-]fpu-fma`
Enable/Disable FPU fused-multiply-add instructions

`-mall-ext`
Turn on all extensions and instructions support

The following options are available when as is configured for a PowerPC processor.

-a32
Generate ELF32 or XCOFF32.

-a64
Generate ELF64 or XCOFF64.

-K PIC
Set EF_PPC_RELOCATABLE_LIB in ELF flags.

-mpwrx | -mpwr2
Generate code for POWER/2 (RIOS2).

-mpwr
Generate code for POWER (RIOS1)

-m601
Generate code for PowerPC 601.

-mppc, -mppc32, -m603, -m604
Generate code for PowerPC 603/604.

-m403, -m405
Generate code for PowerPC 403/405.

-m440
Generate code for PowerPC 440. BookE and some 405 instructions.

-m464
Generate code for PowerPC 464.

−m476
Generate code for PowerPC 476.

−m7400, −m7410, −m7450, −m7455
Generate code for PowerPC 7400/7410/7450/7455.

−m750cl, −mgekko, −mbroadway
Generate code for PowerPC 750CL/Gekko/Broadway.

−m821, −m850, −m860
Generate code for PowerPC 821/850/860.

−mppc64, −m620
Generate code for PowerPC 620/625/630.

−me500, −me500x2
Generate code for Motorola e500 core complex.

−me500mc
Generate code for Freescale e500mc core complex.

−me500mc64
Generate code for Freescale e500mc64 core complex.

−me5500
Generate code for Freescale e5500 core complex.

−me6500
Generate code for Freescale e6500 core complex.

−mspe
Generate code for Motorola SPE instructions.

−mspe2
Generate code for Freescale SPE2 instructions.

−mtitan
Generate code for AppliedMicro Titan core complex.

−mppc64bridge
Generate code for PowerPC 64, including bridge insns.

−mbooke
Generate code for 32−bit BookE.

−ma2
Generate code for A2 architecture.

−me300
Generate code for PowerPC e300 family.

−maltivec
Generate code for processors with AltiVec instructions.

−mvle
Generate code for Freescale PowerPC VLE instructions.

−mvsx
Generate code for processors with Vector-Scalar (VSX) instructions.

−mhtm
Generate code for processors with Hardware Transactional Memory instructions.

−mpower4, −mpwr4
Generate code for Power4 architecture.

−mpower5, −mpwr5, −mpwr5x
Generate code for Power5 architecture.

−mpower6, −mpwr6
Generate code for Power6 architecture.

−mpower7, −mpwr7
Generate code for Power7 architecture.

−mpower8, −mpwr8
Generate code for Power8 architecture.

−mpower9, −mpwr9
Generate code for Power9 architecture.

−mpower10, −mpwr10
Generate code for Power10 architecture.

−mcell
−mcell
Generate code for Cell Broadband Engine architecture.

−mcom
Generate code Power/PowerPC common instructions.

−many
Generate code for any architecture (PWR/PWRX/PPC).

−mregnames
Allow symbolic names for registers.

−mno−regnames
Do not allow symbolic names for registers.

−mrelocatable
Support for GCC's −mrelocatable option.

−mrelocatable−lib
Support for GCC's −mrelocatable−lib option.

−memb
Set PPC_EMB bit in ELF flags.

−mlittle, −mlittle−endian, −le
Generate code for a little endian machine.

−mbig, −mbig−endian, −be
Generate code for a big endian machine.

−msolaris
Generate code for Solaris.

−mno−solaris
Do not generate code for Solaris.

−nops=*count*
If an alignment directive inserts more than *count* nops, put a branch at the beginning to skip execution of the nops.

The following options are available when as is configured for a RISC-V processor.

−fpic
−fPIC
Generate position-independent code

−fno−pic
Don’t generate position-independent code (default)

−march=ISA
Select the base isa, as specified by ISA. For example −march=rv32ima. If this option and the architecture attributes aren’t set, then assembler will check the default configure setting −−with−arch=ISA.

−misa−spec=ISAspec
Select the default isa spec version. If the version of ISA isn’t set by −march, then assembler helps to set the version according to the default chosen spec. If this option isn’t set, then assembler will check the default configure setting −−with−isa−spec=ISAspec.

−mpriv−spec=PRIVspec
Select the privileged spec version. We can decide whether the CSR is valid or not according to the chosen spec. If this option and the privilege attributes aren’t set, then assembler will check the default configure setting −−with−priv−spec=PRIVspec.

−mabi=ABI
Selects the ABI, which is either ‘‘ilp32’’ or ‘‘lp64’’, optionally followed by ‘‘f’’, ‘‘d’’, or ‘‘q’’ to indicate single-precision, double-precision, or quad-precision floating-point calling convention, or none to indicate the soft-float calling convention. Also, ‘‘ilp32’’ can optionally be followed by ‘‘e’’ to indicate the RVE ABI, which is always soft-float.

−mrelax
Take advantage of linker relaxations to reduce the number of instructions required to materialize symbol addresses. (default)

−mno−relax
Don’t do linker relaxations.

−march−attr
Generate the default contents for the riscv elf attribute section if the .attribute directives are not set. This section is used to record the information that a linker or runtime loader needs to check compatibility. This information includes ISA string, stack alignment requirement, unaligned memory accesses, and the major, minor and revision version of privileged specification.

−mno−arch−attr
Don’t generate the default riscv elf attribute section if the .attribute directives are not set.

−mcsr−check
Enable the CSR checking for the ISA-dependent CRS and the read-only CSR. The ISA-dependent CSR are only valid when the specific ISA is set. The read-only CSR can not be written by the CSR instructions.

−mno−csr−check
Don’t do CSR checking.

−mlittle−endian
Generate code for a little endian machine.

−mbig−endian
Generate code for a big endian machine.

See the info pages for documentation of the RX-specific options.

The following options are available when as is configured for the s390 processor family.

−m31
−m64
Select the word size, either 31/32 bits or 64 bits.

−mesa
−mzarch
Select the architecture mode, either the Enterprise System Architecture (esa) or the z/Architecture mode (zarch).

−march=*processor*
Specify which s390 processor variant is the target, **g5** (or **arch3**), **g6**, **z900** (or **arch5**), **z990** (or **arch6**), **z9−109**, **z9−ec** (or **arch7**), **z10** (or **arch8**), **z196** (or **arch9**), **zEC12** (or **arch10**), **z13** (or **arch11**), **z14** (or **arch12**), **z15** (or **arch13**), or **z16** (or **arch14**).

−mregnames
−mno−regnames
Allow or disallow symbolic names for registers.

−mwarn−areg−zero
Warn whenever the operand for a base or index register has been specified but evaluates to zero.

The following options are available when as is configured for a TMS320C6000 processor.

−march=*arch*
Enable (only) instructions from architecture *arch*. By default, all instructions are permitted.

The following values of *arch* are accepted: `c62x`, `c64x`, `c64x+`, `c67x`, `c67x+`, `c674x`.

−mdsbt
−mno−dsbt
The **−mdsbt** option causes the assembler to generate the `Tag_ABI_DSBT` attribute with a value of 1, indicating that the code is using DSBT addressing. The **−mno−dsbt** option, the default, causes the tag to have a value of 0, indicating that the code does not use DSBT addressing. The linker will emit a warning if objects of different type (DSBT and non-DSBT) are linked together.

−mpid=no
−mpid=near
−mpid=far
The **−mpid=** option causes the assembler to generate the `Tag_ABI_PID` attribute with a value indicating the form of data addressing used by the code. **−mpid=no**, the def ault, indicates position-dependent data addressing, **−mpid=near** indicates position-independent addressing with GOT accesses using near DP addressing, and **−mpid=far** indicates position-independent addressing with GOT accesses using far DP addressing. The linker will emit a warning if objects built with different settings of this option are linked together.

−mpic
−mno−pic
The **−mpic** option causes the assembler to generate the `Tag_ABI_PIC` attribute with a value of 1, indicating that the code is using position-independent code addressing, The `−mno−pic` option, the default, causes the tag to have a value of 0, indicating position-dependent code addressing. The linker will emit a warning if objects of different type (position-dependent and position-independent) are linked together.

−mbig−endian
−mlittle−endian
Generate code for the specified endianness. The default is little-endian.

The following options are available when as is configured for a TILE-Gx processor.

−m32 | −m64
Select the word size, either 32 bits or 64 bits.

−EB | −EL
Select the endianness, either big-endian (−EB) or little-endian (−EL).

The following option is available when as is configured for a Visium processor.

−mtune=*arch*
This option specifies the target architecture. If an attempt is made to assemble an instruction that will not execute on the target architecture, the assembler will issue an error message.

The following names are recognized: `mcm24` `mcm` `gr5` `gr6`

The following options are available when as is configured for an Xtensa processor.

−−text−section−literals | −−no−text−section−literals
Control the treatment of literal pools. The default is **−−no−text−section−literals**, which places literals in separate sections in the output file. This allows the literal pool to be placed in a data RAM/ROM. With **−−text−section−literals**, the literals are interspersed in the text section in order to keep them as close as possible to their references. This may be necessary for large assembly files, where the literals would otherwise be out of range of the `L32R` instructions in the text section. Literals are grouped into pools following `.literal_position` directives or preceding `ENTRY` instructions. These options only affect literals referenced via PC-relative `L32R` instructions; literals for absolute mode `L32R` instructions are handled separately.

−−auto−litpools | −−no−auto−litpools
Control the treatment of literal pools. The default is **−−no−auto−litpools**, which in the absence of **−−text−section−literals** places literals in separate sections in the output file. This allows the literal pool to be placed in a data RAM/ROM. With **−−auto−litpools**, the literals are interspersed in the text section in order to keep them as close as possible to their references, explicit `.literal_position` directives are not required. This may be necessary for very large functions, where single literal pool at the beginning of the function may not be reachable by `L32R` instructions at the end. These options only affect literals referenced via PC-relative `L32R` instructions; literals for absolute mode `L32R` instructions are handled separately. When used together with**−−text−section−literals**, **−−auto−litpools** takes precedence.

−−absolute−literals | −−no−absolute−literals
Indicate to the assembler whether `L32R` instructions use absolute or PC-relative addressing. If the processor includes the absolute addressing option, the default is to use absolute `L32R` relocations. Otherwise, only the PC-relative `L32R` relocations can be used.

−−target−align | −−no−target−align
Enable or disable automatic alignment to reduce branch penalties at some expense in code size. This optimization is enabled by default. Note that the assembler will always align instructions like `LOOP` that have fixed alignment requirements.

−−longcalls | −−no−longcalls
Enable or disable transformation of call instructions to allow calls across a greater range of addresses. This option should be used when call targets can potentially be out of range. It may degrade both code size and performance, but the linker can generally optimize away the unnecessary overhead when a call ends up within range. The default is **−−no−longcalls**.

−−transform | −−no−transform
Enable or disable all assembler transformations of Xtensa instructions, including both relaxation and optimization. The default is **−−transform**; **−−no−transform** should only be used in the rare cases when the instructions must be exactly as specified in the assembly source. Using **−−no−transform** causes out of range instruction operands to be errors.

−−rename−section *oldname=newname*
Rename the *oldname* section to *newname*. This option can be used multiple times to rename multiple sections.

−−trampolines | −−no−trampolines
Enable or disable transformation of jump instructions to allow jumps across a greater range of addresses. This option should be used when jump targets can potentially be out of range. In the absence of such jumps this option does not affect code size or performance. The default is **−−trampolines**.

−−abi−windowed | −−abi−call0
Choose ABI tag written to the `.xtensa.info` section. ABI tag indicates ABI of the assembly code. A warning is issued by the linker on an attempt to link object files with inconsistent ABI tags. Default ABI is chosen by the Xtensa core configuration.

The following options are available when as is configured for an Z80 processor.

`@chapter` Z80 Dependent Features

Command-line Options

−march=*CPU***[−***EXT***...][+***EXT***...]**
This option specifies the target processor. The assembler will issue an error message if an attempt is made to assemble an instruction which will not execute on the target processor. The following processor names are recognized: `z80`, `z180`, `ez80`, `gbz80`, `z80n`, `r800`. In addition to the basic instruction set, the assembler can be told to accept some extention mnemonics. For example, `−march=z180+sli+infc` extends *z180* with *SLI* instructions and *IN F,(C)*. The following extentions are currently supported: `full` (all known instructions), `adl` (ADL CPU mode by default, eZ80 only), `sli` (instruction known as *SLI*, *SLL* or *SL1*), `xyhl` (instructions with halves of index registers: *IXL*, *IXH*, *IYL*, *IYH*), `xdcb` (instructions like *RotOp (II+d),R* and *BitOp n,(II+d),R*), `infc` (instruction *IN F,(C)* or *IN (C)*), `outc0` (instruction *OUT (C),0*). Note that rather than extending a basic instruction set, the extention mnemonics starting with `−` revoke the respective functionality: `−march=z80−full+xyhl` first removes all default extentions and adds support for index registers halves only.

If this option is not specified then `−march=z80+xyhl+infc` is assumed.

−local−prefix=*prefix*
Mark all labels with specified prefix as local. But such label can be marked global explicitly in the code. This option do not change default local label prefix `.L`, it is just adds new one.

−colonless
Accept colonless labels. All symbols at line begin are treated as labels.

−sdcc
Accept assembler code produced by SDCC.

−fp−s=*FORMAT*
Single precision floating point numbers format. Default: ieee754 (32 bit).

−fp−d=*FORMAT*
Double precision floating point numbers format. Default: ieee754 (64 bit).

SEE ALSO

gcc (1), **ld** (1), and the Info entries for *binutils* and *ld*.

COPYRIGHT