

SEGGER Assembler

An assembler for
Arm microcontrollers

User Guide & Reference Manual

Document: UM20006
Software Version: 2.10 Beta
Revision: 0
Date: July 30, 2021



A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

Specifications written in this document are believed to be accurate, but are not guaranteed to be entirely free of error. The information in this manual is subject to change for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions. SEGGER makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. SEGGER specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Copyright notice

You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of SEGGER. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2018-2020 SEGGER Microcontroller GmbH, Monheim am Rhein / Germany

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49 2173-99312-0
Fax. +49 2173-99312-28
E-mail: support@segger.com*
Internet: www.segger.com

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at <https://www.segger.com/legal/privacy-policy/>.

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please report it to us and we will try to assist you as soon as possible.

Contact us for further information on topics or functions that are not yet documented.

Print date: July 30, 2021

Software	Revision	Date	By	Description
2.10	0	180514	PC	Initial release.
1.10	0	180427	PC	T32 + A32 internal release.
1.00	0	180410	PC	T32 internal release.

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler).
- The C programming language.
- The target processor.
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend *The C Programming Language* by Kernighan and Richie (ISBN 0--13--1103628), which describes the standard in C programming and, in newer editions, also covers the ANSI C standard.

How to use this manual

This manual describes all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
Parameter	Parameters in API functions.
Sample	Sample code in program examples.
Sample comment	Comments in program examples.
User Input	Text entered at the keyboard by a user in a session transcript.
Secret Input	Text entered at the keyboard by a user, but not echoed (e.g. password entry), in a session transcript.
Reference	Reference to chapters, sections, tables and figures or other documents.
Emphasis	Very important sections.
SEGGER home page	A hyperlink to an external document or web site.

Table of contents

1	About the assembler	10
1.1	Introduction	11
1.1.1	What is the SEGGER Assembler?	11
1.1.2	Assembler features	11
2	Command-line options	12
2.1	Input file options	13
2.1.1	--include	14
2.1.2	--via	15
2.2	Output file options	16
2.2.1	--dependency-file	17
2.2.2	--keep-empty-sections	17
2.2.3	--no-keep-empty-sections	18
2.2.4	--keep-local-symbols	19
2.2.5	--no-keep-local-symbols	20
2.2.6	--mapping-symbols	21
2.2.7	--no-mapping-symbols	22
2.2.8	--output	23
2.2.9	--symbols	24
2.2.10	--no-symbols	25
2.2.11	--pad-sections	26
2.2.12	--no-pad-sections	27
2.3	Target selection options	28
2.3.1	--cpu=name	28
2.3.2	-marm	29
2.3.3	-mbig-endian	30
2.3.4	-mlittle-endian	31
2.3.5	-mthumb	32
2.4	Control options	33
2.4.1	--auto-import	33
2.4.2	--no-auto-import	34
2.4.3	--define	35
2.4.4	--input-syntax	36
2.5	List file options	37
2.5.1	--list-diagnostics	37
2.5.2	--no-list-diagnostics	38
2.5.3	--list-file	39
2.5.4	--list-form-feed	40
2.5.5	--no-list-form-feed	41
2.5.6	--list-html	42

2.5.7	--list-page-length	43
2.5.8	--list-symbols	44
2.5.9	--no-list-symbols	45
2.5.10	--list-text	46
2.6	Diagnostic options	47
2.6.1	--remarks	47
2.6.2	--remarks-are-errors	48
2.6.3	--remarks-are-warnings	49
2.6.4	--no-remarks	50
2.6.5	--silent	51
2.6.6	--verbose	52
2.6.7	--warnings	53
2.6.8	--warnings-are-errors	54
2.6.9	--no-warnings	54
3	Expressions	55
4	GNU syntax	56
4.1	Source file directives	57
4.1.1	.INCBIN	57
4.2	Mode directives	58
4.2.1	.ARM	58
4.2.2	.CODE	59
4.2.3	.THUMB	60
4.3	Section directives	61
4.3.1	.DATA	61
4.3.2	.BSS	62
4.3.3	.TEXT	63
4.3.4	.RODATA	64
4.4	Symbol directives	65
4.4.1	.GLOBL, .GLOBAL, .EXTERN	65
4.4.2	.LOCAL	66
4.4.3	.WEAK	67
4.4.4	.HIDDEN	68
4.4.5	.PROTECTED	69
4.4.6	.INTERNAL	70
4.4.7	.TYPE	71
4.4.8	.SIZE	72
4.5	Data allocation directives	73
4.5.1	.BYTE, .ASCII	73
4.5.2	.STRING, .ASCIZ	74
4.5.3	.2BYTE	75
4.5.4	.4BYTE	76
4.5.5	.HWORD, .SHORT	77
4.5.6	.WORD, .INT, .LONG	78
4.5.7	.INST	79
4.5.8	.INST.N	80
4.5.9	.INST.W	81
4.6	Alignment directives	82
4.6.1	.ALIGN	82
4.6.2	.P2ALIGN, .P2ALIGNW, .P2ALIGNL	83
4.7	Conditional directives	84
4.7.1	.IF, .IFNE	84
4.7.2	.IFEQ	85
4.7.3	.IFGE	86
4.7.4	.IFGT	87
4.7.5	.IFLE	88
4.7.6	.IFLT	89
4.7.7	.IFDEF	90

4.7.8	.IFNDEF, .IFNOTDEF	91
4.8	Listing directives	92
4.8.1	.LIST	92
4.8.2	.NOLIST	93
4.8.3	.TITLE	94
4.8.4	.SUBTTL	95
4.9	Miscellaneous directives	96
4.9.1	.ERR	96
4.9.2	.ABORT	97
4.9.3	.IDENT	98
4.9.4	.POOL, .LTORG	99
4.9.5	.SYNTAX	100
5	Arm syntax	101
5.1	Mode directives	102
5.1.1	CODE16	102
5.1.2	CODE32	103
5.2	Listing directives	104
5.2.1	TTL	105
5.2.2	SUBT	106
6	IAR syntax	107
6.1	Mode directives	108
6.1.1	ARM, CODE32	108
6.1.2	CODE	109
6.1.3	THUMB, CODE16	110
6.2	Data allocation directives	111
6.2.1	DC8, DCB	111
6.2.2	DC16, DCW	112
6.2.3	DC24	113
6.2.4	DC32, DCD	114
6.2.5	DCI16, DCIW	115
6.2.6	DCI32, DCID	116
6.2.7	DS8, DS	117
6.2.8	DS16, DSW	118
6.2.9	DS24	119
6.2.10	DS32, DSD	120
7	Instructions	121
7.1	Alphabetical list of instructions	122
7.2	Instructions by architecture	123
8	Differences between assemblers	124
8.1	Syntax and Encoding differences	125
8.1.1	MUL and MULS	125
8.1.2	MULS	125
9	Indexes	126
9.1	Subject index	127

Chapter 1

About the assembler

1.1 Introduction

This section presents an overview of the SEGGER Assembler and its capabilities.

1.1.1 What is the SEGGER Assembler?

The SEGGER Assembler is a fast assembler that processes Cortex-M assembly language source files. It is designed to be very flexible, yet simple to use.

The assembler accepts GNU, Arm, and IAR syntax source files and will assemble them to Arm ELF object files that can be linked by a standard Arm linker.

1.1.2 Assembler features

The SEGGER Assembler has the following features:

- Highly efficient and very fast to assemble.
- Supports GNU, Arm, and IAR syntax.

Chapter 2

Command-line options

Command line option naming is generally compatible with the following toolsets:

- GNU assembler `as`
- Arm assembler `armasm`
- IAR assembler `iasmarm`

2.1 Input file options

2.1.1 --include

Summary

Add include directory.

Syntax

`--include path`
`-Ipath`

Description

This option adds the path *path* to the end of the list of directories to search for included source files and included binary files.

2.1.2 --via

Synopsis

Read additional options and input files from file.

Syntax

```
--via filename  
--via=filename  
-f filename  
@filename
```

Description

This option reads the file *filename* for additional options and input files. Options are separated by spaces or newlines, and file names which contain special characters, such as spaces, must be enclosed in double quotation marks.

Notes

This option can only be provided on the command line and cannot appear in an indirect file.

2.2 Output file options

2.2.1 --dependency-file

Summary

Write dependencies to file.

Syntax

`--dependency-file filename`
`--MD filename`

Description

This option instructs the assembler to write the list of dependencies (included source files and included binary files) to the file *filename*.

2.2.2 --keep-empty-sections

Summary

Keep empty sections in object file.

Syntax

`--keep-empty-sections`

Description

This option instructs the assembler to emit empty sections that have been created during assembly. An empty section is a section that is created using a section directive but has no instructions or data allocated to it.

The default is to eliminate empty sections.

See also

`--no-keep-empty-sections` on page 18

2.2.3 --no-keep-empty-sections

Summary

Eliminate empty sections from object file.

Syntax

--no-keep-empty-sections

Description

This option instructs the assembler to eliminate empty sections that have been created during assembly. An empty section is a section that is created using a section directive but has no instructions or data allocated to it.

The default is to eliminate empty sections.

See also

--keep-empty-sections on page 17

2.2.4 --keep-local-symbols

Summary

Keep local symbol definitions in object file.

Syntax

--keep-local-symbols

Description

This option instructs the assembler to include local symbol definitions created during assembly in the object file's symbol table. Local symbols can be either an equate or a local label definition that is not exported as a global symbol.

The default is eliminate local symbols.

Note

Local symbols that start `".L"` are never emitted to the object even with **--keep-local-symbols** in effect.

See also

--no-keep-local-symbols on page 20

2.2.5 --no-keep-local-symbols

Summary

Do not keep local symbol definitions in object file.

Syntax

`--no-keep-local-symbols`

Description

This option instructs the assembler to exclude local symbol definitions created during assembly from the object file's symbol table. Local symbols can be either an equate or a local label definition that is not exported as a global symbol.

See also

--keep-local-symbols on page 19

2.2.6 --mapping-symbols

Summary

Include mapping symbols.

Syntax

--mapping-symbols

Description

This option includes mapping symbols in the Arm ELF file.

See also

--no-mapping-symbols on page 22

2.2.7 --no-mapping-symbols

Summary

Do not generate mapping symbols.

Syntax

`--no-mapping-symbols`

Description

This option produces an Arm ELF file that contains no code and data mapping symbols.

See also

--mapping-symbols on page 21

2.2.8 --output

Summary

Set output file name.

Syntax

`--output filename`

`-o filename`

`--output=filename`

`-o=filename`

Description

This option sets the Arm ELF output filename, typically with extension “o”.

2.2.9 --symbols

Summary

Include symbol table.

Syntax

`--symbols`

Description

This option includes the symbol table in the Arm ELF file. This does not automatically enable mapping symbol inclusion.

See also

`--no-symbols` on page 25

2.2.10 --no-symbols

Summary

Discard symbol table.

Syntax

`--no-symbols`

Description

This option eliminates all symbols from the Arm ELF file that are not required which includes the mapping symbols.

See also

`--symbols` on page 24

2.2.11 --pad-sections

Summary

Pad sections to alignment boundary.

Syntax

--pad-sections

Description

This option instructs the assembler to emit additional data to a section such that its size is a multiple of its alignment. Padding a section to a multiple of its alignment may waste space that a linker could otherwise utilize for section packing.

The default is not to pad sections.

See also

--no-pad-sections on page 27

2.2.12 --no-pad-sections

Summary

Do not pad sections to alignment boundary.

Syntax

--no-pad-sections

Description

This option instructs the assembler not to emit additional data to a section such that its size is a multiple of its alignment. Padding a section to a multiple of its alignment may waste space that a linker could otherwise utilize for section packing.

The default is not to pad sections.

See also

--pad-sections on page 26

2.3 Target selection options

2.3.1 --cpu=name

Summary

Set target core or architecture.

Syntax

--cpu=*name*
-cpu=*name*
-mcpu=*name*

Description

This option selects the target processor for the application and controls the construction of appropriate veneers when required.

The core names accepted are:

- cortex-m0, cortex-m0plus, cortex-m0+, cortex-m1, cortex-m3, cortex-m4, cortex-m7, cortex-m23, cortex-m33
- cortex-a5, cortex-a7, cortex-a8, cortex-a9, cortex-a12, cortex-a15, cortex-a17, cortex-a32, cortex-a35, cortex-a53, cortex-a55, cortex-a57, cortex-a72, cortex-a73, cortex-a75
- cortex-r4, cortex-r4f, cortex-r5, cortex-r7, cortex-r8, cortex-r52

The architecture names accepted are:

- 4t, 5t, 5te
- 6, 6j, 6k, 6kz, 6t2, 6z, 6zk, 6-m, 6s-m
- 7, 7ve, 7-a, 7-r, 7-m, 7e-m
- 8-a, 8.1-a, 8.2-a, 8.3-a, 8.4-a, 8-r, 8-m.base, 8-m.main

The default is --cpu=cortex-m0.

2.3.2 -marm

Summary

Select Arm instruction set.

Syntax

`-marm`

Description

This option sets the default assembler instruction set to Arm (A32). The default instruction set is the instruction set if the source file does not select a specific instruction set using assembler source directives.

See also

`-mthumb` on page 32

2.3.3 -mbig-endian

Summary

Select big-endian byte order.

Syntax

`-mbig-endian`
`-EB`

Description

This option selects big-endian byte ordering for data and instructions.

The default is little-endian byte ordering.

See also

-mlittle-endian on page 31

2.3.4 -mlittle-endian

Summary

Select little-endian byte order.

Syntax

`-mlittle-endian`
`-EL`

Description

This option selects little-endian byte ordering for data and instructions.

The default is little-endian byte ordering.

See also

-mbig-endian on page 30

2.3.5 -mthumb

Summary

Select Thumb instruction set.

Syntax

`-mthumb`

Description

This option sets the default assembler instruction set to Thumb (T32). The default instruction set is the instruction set if the source file does not select a specific instruction set using assembler source directives.

See also

`-marm` on page 29

2.4 Control options

2.4.1 --auto-import

Summary

Automatically import symbols.

Syntax

`--auto-import`

Description

This option instructs the assembler to automatically define a symbol as an external (imported) symbol if it is used and has not been declared with a different storage class.

See also

`--no-auto-import` on page 34

2.4.2 --no-auto-import

Summary

Do not automatically import symbols.

Syntax

--no-auto-import

Description

This option instructs the assembler not to automatically define as symbol as an external (imported) symbol and to require that all symbols are declared as local or external before their first use.

See also

--auto-import on page 33

2.4.3 --define

Summary

Define symbol.

Syntax

`--define symbol=value`

`-Dsymbol=value`

Description

This option defines the symbol *symbol* to the numeric value *value*.

2.4.4 --input-syntax

Summary

Set source file syntax.

Syntax

`--input-syntax=name`
`-fsyntax=name`

Description

This option selects the source file syntax. The syntaxes accepted by the assembler are:

- **arm** — Format is compatible with ARM Limited's `armasm` assembler.
- **iar** — Format is compatible with IAR's `iasmarm` assembler.
- **gnu** — Format is compatible with The Free Software Foundation's `gas` assembler.

There is one additional syntax identifier:

- **auto** — Source format is automatically derived from the input file.

The default is **auto** syntax where the assembler automatically determines the appropriate source file syntax using heuristics.

2.5 List file options

2.5.1 --list-diagnostics

Summary

Print diagnostics to listing.

Syntax

--list-diagnostics

Description

This option instructs the assembler to print all diagnostics at the end of the listing.

See also

--no-list-diagnostics on page 38

2.5.2 --no-list-diagnostics

Summary

Do not print diagnostics to listing.

Syntax

--no-list-diagnostics

Description

This option instructs the assembler not to print diagnostics at the end of the listing.

See also

--list-diagnostics on page 37

2.5.3 --list-file

Summary

Generate an assembler listing file.

Syntax

```
--list-file filename  
--list-file=filename  
-Lfilename
```

Description

Generates an assembler listing to the given filename.

2.5.4 --list-form-feed

Summary

Issue form feed between pages.

Syntax

--list-form-feed

Description

This option instructs the assembler to write a form feed between listing pages.

See also

--list-page-length on page 43, *--no-list-form-feed* on page 41

2.5.5 --no-list-form-feed

Summary

Do not issue form feed between pages.

Syntax

`--no-list-form-feed`

Description

This option inhibits the assembler writing a form feed between listing pages.

See also

--list-form-feed on page 40

2.5.6 --list-html

Summary

Format listing as an HTML document.

Syntax

`--list-html`

Description

This option instructs the assembler to format the assembler listing as an HTML document. In this mode, listing pagination controls are ignored.

See also

`--list-text` on page 46

2.5.7 --list-page-length

Summary

Set number of lines per page.

Syntax

--list-page-length=*value*

Description

This option sets the number of lines per page to *value*. If *value* is zero, which is the default, the listing is not divided into pages.

See also

--list-form-feed on page 40

2.5.8 --list-symbols

Summary

Print symbol table to listing.

Syntax

--list-symbols

Description

This option instructs the assembler to print a symbol table at the end of the listing.

See also

--no-list-symbols on page 45

2.5.9 --no-list-symbols

Summary

Do not print symbol table to listing.

Syntax

`--no-list-symbols`

Description

This option instructs the assembler not to print a symbol table at the end of the listing.

See also

--list-symbols on page 44

2.5.10 --list-text

Summary

Format listing as a text document.

Syntax

--list-text

Description

This option instructs the assembler to format the assembler listing as an HTML document. In this mode, listing pagination controls are effective.

This is the default.

See also

--list-html on page 42

2.6 Diagnostic options

2.6.1 --remarks

Summary

Issue remarks.

Syntax

`--remarks`

Description

This option instructs the assembler to issue remarks for potential issues during assembly. This is the default.

See also

--no-remarks on page 50, *--remarks-are-warnings* on page 49, *--remarks-are-errors* on page 48

2.6.2 --remarks-are-errors

Summary

Elevate remarks to errors.

Syntax

`--remarks-are-errors`

Description

This option elevates all remark diagnostics issued by the assembler to errors.

See also

`--no-remarks` on page 50, `--remarks` on page 47, , `--remarks-are-warnings` on page 49

2.6.3 --remarks-are-warnings

Summary

Elevate remarks to warnings.

Syntax

`--remarks-are-warnings`
`--remarks_are_warnings`

Description

This option elevates all remark diagnostics issued by the assembler to warnings.

See also

`--no-remarks` on page 50, `--remarks` on page 47, , `--remarks-are-errors` on page 48

2.6.4 --no-remarks

Summary

Suppress remarks.

Syntax

--no-remarks

Description

This option disables all remark diagnostics issued by the assembler. Although remarks are suppressed, the total number of remarks that are suppressed by the assembler is shown at the end of linking:

```
C:> segger-as --via=app.ind
Copyright (c) 2017-2018 SEGGER Microcontroller GmbH    www.segger.com
SEGGER Assembler 2.10 compiled Apr 11 2018 10:50:34

Assembly complete: 0 errors, 0 warnings, 2 remarks suppressed

C:> _
```

See also

--remarks-are-warnings on page 49, *--remarks-are-errors* on page 48

2.6.5 --silent

Summary

Do not show output.

Syntax

`--silent`

`-q`

Description

This option inhibits all assembler status messages; only diagnostic messages are shown.

See also

`--verbose` on page 52

2.6.6 --verbose

Summary

Increase verbosity.

Syntax

--verbose

-v

Description

This option increase the verbosity of the assembler by one level.

See also

--*silent* on page 51

2.6.7 --warnings

Summary

Issue warnings.

Syntax

`--warnings`

Description

This option instructs the assembler to issue warnings for dubious use or inputs. This is the default.

See also

`--no-warnings` on page 54, `--warnings-are-errors` on page 54

2.6.8 --warnings-are-errors

Summary

Elevate warnings to errors.

Syntax

`--warnings-are-errors`
`--fatal-warnings`

Description

This option elevates all warning diagnostics issued by the assembler to errors.

See also

`--no-warnings` on page 54, `--warnings` on page 53

2.6.9 --no-warnings

Summary

Suppress warnings.

Syntax

`--no-warnings`

Description

This option disables all warning diagnostics issued by the assembler. Although warnings are suppressed, the total number of warnings that are suppressed by the assembler is shown at the end of linking:

```
C:> segger-as --via=app.ind
SEGGER Macro Assembler V2.10 compiled Jul  9 2020 14:30:42
Copyright (c) 2018-2020 SEGGER Microcontroller GmbH    www.segger.com

Assembly complete: 0 errors, 1 warnings suppressed, 0 remarks

C:> _
```

See also

`--warnings` on page 53, `--warnings-are-errors` on page 54

Chapter 3

Expressions

GNU	IAR	ARM
(), prefix +, -, !, ~	(), prefix +, -, !, ~, LOW, HIGH, BYTE1, BYTE2, BYTE3, BYTE4, LWRD, HWRD, DATE, SFB, SFE, SIZEOF	(), prefix +, -, :NOT:, :LNOT:
*, /, %, <<, >>	*, /, %	*, /, %, :MOD:
, &, ^, !	+, -	<<, >>, :SHL:, :SHR:, :ROL:, :ROR:
+, -, =, <>, ≠, <, >, ≤, ≥	<<, >>	+, -, &, ^, :AND:, :OR:, :EOR:
&&,	&&, &	=, <>, ≠, /=, <, >, ≤, ≥
	, , ^, XOR	&&, , :LAND:, :LOR:, :LEOR:
	=, =, <>, ≠, <, >, ≤, ≥, UGT, ULT	

Chapter 4

GNU syntax

4.1 Source file directives

4.1.1 .INCBIN

Include binary file.

Syntax

.INCBIN "*filename*"

Description

Searches for the binary file *filename* using include paths. Inserts the binary file into the current section as data with alignment 1.

4.2 Mode directives

4.2.1 .ARM

Select Arm instruction set.

Syntax

.ARM

Description

Selects the Arm instruction set for subsequent instructions.

See also

.THUMB on page 60

4.2.3 .THUMB

Select Thumb instruction set.

Syntax

.THUMB

Description

Selects the Thumb instruction set for subsequent instructions.

See also

.ARM on page 58

4.3 Section directives

4.3.1 .DATA

Switch to data section

Syntax

.DATA

Description

Switches to the .data section.

4.3.2 .BSS

Switch to bss section

Syntax

.BSS

Description

Switches to the .bss section.

4.3.3 .TEXT

Switch to text section

Syntax

`.TEXT`

Description

Switches to the `.text` section.

4.3.4 .RODATA

Switch to data section

Syntax

.RODATA

Description

Switches to the .rodata section.

4.4 Symbol directives

4.4.1 .GLOBL, .GLOBAL, .EXTERN

Set external symbol binding.

Syntax

```
.GLOBL name, name...  
.GLOBAL name, name...  
.EXTERN name, name...
```

Description

Defines all symbols in the symbol list to have global binding so that they are externally visible.

4.4.2 .LOCAL

Set local symbol binding.

Syntax

.LOCAL *name, name...*

Description

Defines all symbols in the symbol list to have local binding so that they are not externally visible.

4.4.3 .WEAK

Set weak symbol binding.

Syntax

.WEAK *name*, *name*...

Description

Defines all symbols in the symbol list to have weak binding so that they can be overridden when linked.

4.4.4 .HIDDEN

Set symbol visibility to hidden.

Syntax

.HIDDEN *name, name...*

Description

Defines all symbols in the symbol list to have hidden visibility and is stored as **STV_HIDDEN** in the **st_other** member of the ELF symbol table entry.

4.4.5 .PROTECTED

Set symbol visibility to protected.

Syntax

.PROTECTED *name, name...*

Description

Defines all symbols in the symbol list to have protected visibility and is stored as **STV_PROTECTED** in the **st_other** member of the ELF symbol table entry.

4.4.6 .INTERNAL

Set symbol visibility to internal.

Syntax

.INTERNAL *name, name...*

Description

Defines all symbols in the symbol list to have internal visibility and is stored as **STV_INTERNAL** in the **st_other** member of the ELF symbol table entry.

4.4.7 .TYPE

Set type of ELF symbol.

Syntax

`.TYPE name, [ % | @ ] type`

Description

Sets the type of the symbol *name* to the the type *type* which can be **object** or **function**.

Example

```
.type SEGGER_DSP_GetVersion, %function  
.type SEGGER_DSP_aTwiddleFactors, %object
```

4.4.8 .SIZE

Set size of ELF symbol.

Syntax

.SIZE *name*, *expr*

Description

Sets the size of the symbol *name* to *size*.

Example

```
.size SEGGER_DSP_aTwiddleFactors, 1024
```

4.5 Data allocation directives

4.5.1 .BYTE, .ASCII

Allocate 8-bit data.

Syntax

.BYTE *expr*, *expr*...
.ASCII *expr*, *expr*...

Description

Place the initialized 8-bit values in the expression list into memory at the location counter.

Example

SEGGER Assembler V2.10 www.segger.com
GNU_BYTE.s - Assembled Thu Nov 7 11:46:23 2019

```
1                                     .DATA
2
3      00000000 4120736D      .BYTE "A small list of primes:"
      00000004 616C6C20
      00000008 6C697374
      0000000C 206F6620
      00000010 7072696D
      00000014 65733A
4      00000017 02030507      .BYTE 2, 3, 5, 7, 11, 13
      0000001B 0B0D
5
6                                     .END
```

ASSEMBLY COMPLETE

No errors, no warnings, no remarks

See also

.STRING, **.ASCIZ** on page 74

4.5.2 .STRING, .ASCIZ

Allocate string data.

Syntax

.STRING *expr*, *expr*...

.ASCIZ *expr*, *expr*...

Description

Place the initialized string values in the expression list into memory at the location counter, each string followed by a trailing zero byte.

Example

SEGGER Assembler V2.10 www.segger.com
GNU_STRING.s - Assembled Thu Nov 7 11:46:23 2019

```

1                               .DATA
2
3                               @ String is terminated by a zero byte
4      00000000 49742073      .STRING "It simply works!"
      00000004 696D706C
      00000008 7920776F
      0000000C 726B7321
      00000010 00

5
6                               @ Each string is terminated by a zero byte
7      00000011 53454747      .STRING "SEGGER", "The Embedded Experts"
      00000015 45520054
      00000019 68652045
      0000001D 6D626564
      00000021 64656420
      00000025 45787065
      00000029 72747300

8
9                               .END
```

ASSEMBLY COMPLETE

No errors, no warnings, no remarks

See also

.BYTE, *.ASCII* on page 73

4.5.4 .4BYTE

Allocate 32-bit data.

Syntax

.4BYTE *expr*, *expr*...

Description

Place the initialized 32-bit values in the expression list into memory at the location counter.

These directives do not force the location counter to be word aligned when placing the data, and do not issue a warning diagnostic when the location counter is not correctly aligned.

Example

```
SEGGGER Assembler V2.10      www.segger.com
GNU_4BYTE.s - Assembled Thu Nov  7 11:46:23 2019

1                               .DATA
2
3                               @ Data that is correctly aligned
4      00000000 DEADBEEF      .4BYTE 0xEFBEADDE, 0xDEADBEEF
5      00000004 EFBEADDE
6
7                               @ Data that is not aligned on a word
7      00000008              .SPACE 1
8      00000009 DEADBEEF      .4BYTE 0xEFBEADDE, 0xDEADBEEF
9      0000000D EFBEADDE
10
10                              .END
```

ASSEMBLY COMPLETE

No errors, no warnings, no remarks

See also

.WORD, *.INT*, *.LONG* on page 78

4.5.5 .HWORD, .SHORT

Allocate 16-bit data.

Syntax

.HWORD *expr*, *expr*...

.SHORT *expr*, *expr*...

Description

Place the initialized 16-bit values in the expression list into memory at the location counter.

These directives do not force the location counter to be halfword aligned when placing the data, but a warning diagnostic is issued when the location counter is not correctly aligned.

Example

```
SEGGER Assembler V2.10      www.segger.com
GNU_HWORD.s - Assembled Thu Nov  7 11:46:23 2019
```

```

1                                .DATA
2
3                                @ Data that is correctly aligned
4      00000000 DEADBEEF        .HWORD 0xADDE, 0xEFBE, 0xDEAD, 0xBEEF
      00000004 ADDEEFBE
5
6                                @ Data that is not aligned on a halfword
7      00000008                .SPACE 1
8      00000009 DEADBEEF        .HWORD 0xADDE, 0xEFBE, 0xDEAD, 0xBEEF
      0000000D ADDEEFBE
9
10                               .END
```

DIAGNOSTICS

```
GNU_HWORD.s:8: warning: 16-bit data does not fall on natural alignment at address 00000009
GNU_HWORD.s:8: warning: 16-bit data does not fall on natural alignment at address 0000000B
GNU_HWORD.s:8: warning: 16-bit data does not fall on natural alignment at address 0000000D
GNU_HWORD.s:8: warning: 16-bit data does not fall on natural alignment at address 0000000F
```

ASSEMBLY COMPLETE

No errors, 4 warnings, no remarks

See also

.2BYTE on page 75

4.5.6 .WORD, .INT, .LONG

Allocate 32-bit data.

Syntax

.WORD *expr*, *expr*...

.INT *expr*, *expr*...

.LONG *expr*, *expr*...

Description

Place the initialized 32-bit values in the expression list into memory at the location counter.

These directives do not force the location counter to be word aligned when placing the data, but a warning diagnostic is issued when the location counter is not correctly aligned.

Example

```
SEGGER Assembler V2.10      www.segger.com
GNU_WORD.s - Assembled Thu Nov  7 11:46:23 2019
```

```

1                                .DATA
2
3                                @ Data that is correctly aligned
4      00000000 DEADBEEF        .WORD 0xEFBEADDE, 0xDEADBEEF
      00000004 EFBEADDE
5
6                                @ Data that is not aligned on a word
7      00000008                .SPACE 1
8      00000009 DEADBEEF        .WORD 0xEFBEADDE, 0xDEADBEEF
      0000000D EFBEADDE
9
10                               .END
```

DIAGNOSTICS

```
GNU_WORD.s:8: warning: 32-bit data does not fall on natural alignment at address 00000009
GNU_WORD.s:8: warning: 32-bit data does not fall on natural alignment at address 0000000D
```

ASSEMBLY COMPLETE

No errors, 2 warnings, no remarks

See also

.4BYTE on page 76

4.5.7 .INST

Allocate 16-bit or 32-bit instruction.

Syntax

`.INST expr, expr...`

Description

Place the initialized values in the expression list into memory at the location counter as instruction codes. For values that fit into 16 bits, a 16-bit instruction is generated, and for others a 32-bit instruction is generated.

4.5.8 .INST.N

Allocate 16-bit instruction.

Syntax

.INST.N *expr*, *expr*...

Description

Place the initialized 16-bit values in the expression list into memory at the location counter as instruction codes.

4.5.9 .INST.W

Allocate 32-bit instruction.

Syntax

`.INST.W expr, expr...`

Description

Place the initialized 32-bit values in the expression list into memory at the location counter as instruction codes.

4.6 Alignment directives

4.6.1 .ALIGN

Align location counter.

Syntax

.ALIGN *align-power*, *fill-value*, *max-bytes*

Description

The *align-power* expression, which must be absolute, is the power of two to align the location counter to. An *align-power* of 3, for instance, will ensure that the location counter is a multiple of 2^3 , i.e. 8.

The optional *fill-value* defines the filler byte to use when aligning the location counter. If omitted, a default value of is selected: the default value for code sections is a nop instruction and for other sections it is zero.

The optional *max-bytes* defines the maximum number of bytes inserted to align the location counter. If more than *max-bytes* bytes are required for alignment, no alignment is performed.

4.6.2 .P2ALIGN, .P2ALIGNW, .P2ALIGNL

Align location counter.

Syntax

```
.P2ALIGN align-power [ , fill-value [ , max-bytes ] ]  
.P2ALIGNW align-power [ , fill-value [ , max-bytes ] ]  
.P2ALIGNL align-power [ , fill-value [ , max-bytes ] ]
```

Description

The *align-power* expression, which must be absolute, is the power of two to align the location counter to. An *align-power* of 3, for instance, will ensure that the location counter is a multiple of 2^3 , i.e. 8.

The optional *fill-value* defines the filler value to use when aligning the location counter. If omitted, a default value of is selected: the default value for code sections is a nop instruction and for other sections it is zero. If it is specified, the fill value is a one-byte pattern for .P2ALIGN, a two-byte pattern for .P2ALIGNW, and a 4-byte pattern for .P2ALIGNL.

The optional *max-bytes* defines the maximum number of bytes inserted to align the location counter. If more than *max-bytes* bytes are required for alignment, no alignment is performed.

4.7 Conditional directives

4.7.1 .IF, .IFNE

Assemble if nonzero.

Syntax

```
.IF expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a nonzero value, assemble *true-statements* otherwise assemble the optional *false-statements*.

See also

.IFEQ on page 85

4.7.2 .IFEQ

Assemble if zero.

Syntax

```
.IFEQ expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a zero value, assemble *true-statements* otherwise assemble the optional *false-statements*.

See also

.IF, *.IFNE* on page 84

4.7.3 .IFGE

Assemble if greater than or equal zero.

Syntax

```
.IFGE expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a value greater than or equal to zero, assemble *true-statements* otherwise assemble the optional *false-statements*.

4.7.4 .IFGT

Assemble if greater than zero.

Syntax

```
.IFGT expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a value greater than zero, assemble *true-statements* otherwise assemble the optional *false-statements*.

4.7.5 .IFLE

Assemble if less than or equal zero.

Syntax

```
.IFLE expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a value less than or equal to zero, assemble *true-statements* otherwise assemble the optional *false-statements*.

4.7.6 .IFLT

Assemble if less than zero.

Syntax

```
.IFLT expr
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the expression *expr* evaluates to a value less than zero, assemble *true-statements* otherwise assemble the optional *false-statements*.

4.7.7 .IFDEF

Assemble if symbol defined.

Syntax

```
.IFDEF id
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the symbol *id* is defined, assemble *true-statements* otherwise assemble the optional *false-statements*.

See also

.IFNDEF, *.IFNOTDEF* on page 91

4.7.8 .IFDEF, .IFNOTDEF

Assemble if symbol not defined.

Syntax

```
.IFDEF id
    true-statements
[ .ELSE
    false-statements ]
.ENDIF
```

Description

If the symbol *id* is not defined, assemble *true-statements* otherwise assemble the optional *false-statements*.

See also

.IFDEF on page 90

4.8 Listing directives

4.8.1 .LIST

Turn on listing.

Syntax

.LIST

Description

Each line subsequently assembled will be printed to the listing file.

4.8.2 .NOLIST

Turn off listing.

Syntax

.LIST

Description

Each line subsequently assembled will be not be printed to the listing file.

4.8.3 .TITLE

Set listing title.

Syntax

`.TITLE string-expr`

Description

Sets the title that is shown at the top of each page in the listing.

4.8.4 .SUBTTL

Set listing subtitle.

Syntax

`.SUBTTL string-expr`

Description

Sets the subtitle that is shown below the title at the top of each page in the listing.

4.9 Miscellaneous directives

4.9.1 .ERR

Generate user-defined error diagnostic.

Syntax

`.ERR string`

Description

Generate a user-defined error diagnostic with the message defined by *string*. The `.ERR` directive is usually combined with one of the conditional directives to indicate a user configuration error.

Example

```
.ifne Align % 4
    .err "Alignment is not a multiple of 4"
.endif
```

4.9.2 .ABORT

Terminate assembly.

Syntax

.ABORT

Description

Immediately terminate assembly with a fatal error indicating that the user requested an abort.

Example

```
.ifndef ENTRY_SYMBOL
    .abort
#endif
```

4.9.3 .IDENT

Place data in comment section.

Syntax

`.IDENT string`

Description

Add *string* to the end of the `.comment` section. The `.comment` section is automatically created if it does not exist.

Example

```
.ident "Copyright (C) 2021 SEGGER Microcontroller GmbH"
```

4.9.4 .POOL, .LTORG

Dump literal pool.

Syntax

.POOL

Description

Dump the literal pool of the active section at the current location counter.

4.9.5 .SYNTAX

Set assembler syntax.

Syntax

.SYNTAX *mode*

Description

Set the input syntax to *mode*. The only allowable mode is **unified**, selecting Unified Assembler Language (UAL): divided syntax, signified by **divided**, is not supported.

Chapter 5

Arm syntax

5.1 Mode directives

5.1.1 CODE16

Select Thumb instruction set.

Syntax

CODE16

Description

Selects the Thumb instruction set for subsequent instructions.

Example

```
SEGGER Assembler V2.10    www.segger.com
ARM_CODE16.s - Assembled Thu Nov  7 11:46:23 2019

1                          AREA |.text|, CODE
2
3                          CODE16
4
5          00000000 0120    MOVS R0, #1
6          00000002 4018    ADDS R0, R0, R1
7
8                          END
```

ASSEMBLY COMPLETE

No errors, no warnings, no remarks

See also

CODE32 on page 103

5.1.2 CODE32

Select Arm instruction set.

Syntax

CODE32

Description

Selects the Arm instruction set for subsequent instructions.

Example

```
SEGGER Assembler V2.10    www.segger.com
ARM_CODE32.s - Assembled Thu Nov  7 11:46:23 2019

1                          AREA |.text|, CODE
2
3                          CODE32
4
5          00000000 0100B0E3    MOVS R0, #1
6          00000004 010090E0    ADDS R0, R0, R1
7
8                          END
```

ASSEMBLY COMPLETE

No errors, no warnings, no remarks

See also

CODE16 on page 102

5.2 Listing directives

5.2.1 TTL

Set listing title.

Syntax

TTL *open-string*

Description

Sets the title that is shown at the top of each page in the listing.

5.2.2 SUBT

Set listing subtitle.

Syntax

SUBT *open-string*

Description

Sets the subtitle that is shown below the title at the top of each page in the listing.

Chapter 6

IAR syntax

6.1 Mode directives

6.1.1 ARM, CODE32

Select Arm instruction set.

Syntax

ARM
CODE32

Description

Selects the Arm instruction set for subsequent instructions.

See also

THUMB, *CODE16* on page 110

6.1.2 CODE

Select default instruction set.

Syntax

`CODE`

Description

Selects the Arm or Thumb instruction set according to that specified on the command line by the `-mthumb` or `-marm` options. If no instruction set is specified on the command line, the instruction defaults to Thumb.

6.1.3 THUMB, CODE16

Select Thumb instruction set.

Syntax

THUMB
CODE16

Description

Selects the Thumb instruction set for subsequent instructions.

See also

ARM, CODE32 on page 108

6.2 Data allocation directives

6.2.1 DC8, DCB

Allocate 8-bit data.

Syntax

DC8 *expr*, *expr*...

DCB *expr*, *expr*...

Description

Place the initialized 8-bit values in the expression list into memory at the location counter.

Example

```
SEGGGER Assembler V2.10    www.segger.com
IAR_DC8.s - Assembled Thu Nov  7 11:46:23 2019

1                               SECTION `.data`:DATA(2)
2
3          00000000 4120736D      DCB "A small list of primes:"
          00000004 616C6C20
          00000008 6C697374
          0000000C 206F6620
          00000010 7072696D
          00000014 65733A
4          00000017 02030507      DC8  2, 3, 5, 7, 11, 13
          0000001B 0B0D

5
6                               END

ASSEMBLY COMPLETE

No errors, no warnings, no remarks
```

6.2.2 DC16, DCW

Allocate 16-bit data.

Syntax

DC16 *expr, expr...*

DCW *expr, expr...*

Description

Place the initialized 16-bit values in the expression list into memory at the location counter.

6.2.3 DC24

Allocate 24-bit data.

Syntax

DC24 *expr*, *expr*...

Description

Place the initialized 32-bit values in the expression list into memory at the location counter.

6.2.4 DC32, DCD

Allocate 32-bit data.

Syntax

`DC32 expr, expr...`

`DCD expr, expr...`

Description

Place the initialized 32-bit values in the expression list into memory at the location counter.

6.2.5 DCI16, DCIW

Allocate 16-bit instruction.

Syntax

DCI16 *expr*, *expr*...

DCIW *expr*, *expr*...

Description

Place the initialized 16-bit values in the expression list into memory at the location counter as instruction codes.

6.2.6 DCI32, DCID

Allocate 32-bit instruction.

Syntax

DCI32 *expr, expr...*

DCID *expr, expr...*

Description

Place the initialized 32-bit values in the expression list into memory at the location counter as instruction codes.

6.2.7 DS8, DS

Allocate space for 8-bit data.

Syntax

DS8 *expr*

DS *expr*

Description

Allocate space for *expr* 8-bit bytes. For zero-data sections such as **.bss**, the space is allocated; for sections that have content, such as **.text** and **.data**, the allocated space is initialized with zero bytes.

6.2.8 DS16, DSW

Allocate space for 16-bit data.

Syntax

DS16 *expr*

DSW *expr*

Description

Allocate space for *expr* 16-bit elements. For zero-data sections such as **.bss**, the space is allocated; for sections that have content, such as **.text** and **.data**, the allocated space is initialized with zero bytes.

6.2.9 DS24

Allocate space for 24-bit data.

Syntax

DS24 *expr*

Description

Allocate space for *expr* 24-bit elements. For zero-data sections such as **.bss**, the space is allocated; for sections that have content, such as **.text** and **.data**, the allocated space is initialized with zero bytes.

6.2.10 DS32, DSD

Allocate space for 32-bit data.

Syntax

DS32 *expr*

DSD *expr*

Description

Allocate space for *expr* 32-bit elements. For zero-data sections such as **.bss**, the space is allocated; for sections that have content, such as **.text** and **.data**, the allocated space is initialized with zero bytes.

Chapter 7

Instructions

7.1 Alphabetical list of instructions

7.2 Instructions by architecture

Chapter 8

Differences between assemblers

8.1 Syntax and Encoding differences

The following sections document the differences in UAL input syntax and the binary encoding of UAL source code. This is particularly important if you expect identical instruction encodings between assemblers, especially when using the SEGGER assembler to assemble existing source code.

8.1.1 MUL and MULS

UAL allows a multiply instruction to be written as `MUL Rd, Rn`. When the only encoding of this UAL input is a three-operand instructions, `MUL Rd, Rn` must be interpreted as `MUL Rd, Rd, Rn`.

The following are the encodings of `MUL Rd, Rn` and `MULS Rd, Rn` where only three-operand encoding is possible. The SEGGER assembler follows Arm's encoding, whereas GNU deviates and IAR rejects correct UAL.

SEGGER	Arm	GNU	IAR
<code>MUL Rd, Rd, Rn</code>	<code>MUL Rd, Rd, Rn</code>	<code>MUL Rd, Rn, Rd</code>	Rejects
<code>MULS Rd, Rd, Rn</code>	<code>MULS Rd, Rd, Rn</code>	<code>MULS Rd, Rn, Rd</code>	Rejects

8.1.2 MULS

UAL allows a multiply instruction to be written as `MULS Rd, Rn`. The assembler can choose between two-operand form and three-operand form based on the registers `Rd` and `Rn`.

The following are the encodings of `MULS R1, R2`:

SEGGER	Arm	GNU	IAR
<code>MULS.N R1, R2</code>	Rejects	<code>MULS.N R1, R2</code>	<code>MULS.N R1, R2</code>

And `MULS R1, R9`:

SEGGER	Arm	GNU	IAR
<code>MULS.W R1, R1, R9</code>	<code>MULS.W R1, R1, R9</code>	<code>MULS.W R1, R9, R1</code>	<code>MULS.W R1, R1, R9</code>

Chapter 9

Indexes

9.1 Subject index

- .2BYTE (GNU directive), 75
- .4BYTE (GNU directive), 76
- .ABORT (GNU directive), 97
- .ALIGN (GNU directive), 82
- .ARM (GNU directive), 58
- ARM (IAR directive), 108
- .ASCII (GNU directive), 73
- .ASCIZ (GNU directive), 74
- auto-import (command line), 33
- .BSS (GNU directive), 62
- .BYTE (GNU directive), 73
- .CODE (GNU directive), 59
- CODE (IAR directive), 109
- CODE16 (Arm directive), 102
- CODE16 (IAR directive), 110
- CODE32 (Arm directive), 103
- CODE32 (IAR directive), 108
- command line,
 - auto-import, 33
 - cpu=name, 28
 - define, 35
 - dependency-file, 17
 - include, 14
 - input-syntax, 36
 - keep-empty-sections, 17
 - keep-local-symbols, 19
 - list-diagnostics, 37
 - list-file, 39
 - list-form-feed, 40
 - list-html, 42
 - list-page-length, 43
 - list-symbols, 44
 - list-text, 46
 - mapping-symbols, 21
 - marm, 29
 - mbig-endian, 30
 - mlittle-endian, 31
 - mthumb, 32
 - no-auto-import, 34
 - no-keep-empty-sections, 18
 - no-keep-local-symbols, 20
 - no-list-diagnostics, 38
 - no-list-form-feed, 41
 - no-list-symbols, 45
 - no-mapping-symbols, 22
 - no-pad-sections, 27
 - no-remarks, 50
 - no-symbols, 25
 - no-warnings, 54
 - output, 23
 - pad-sections, 26
 - remarks, 47
 - remarks-are-errors, 48
 - remarks-are-warnings, 49
 - silent, 51
 - symbols, 24
 - verbose, 52
 - via, 15
 - warnings, 53
 - warnings-are-errors, 54
- cpu=name (command line), 28
- .DATA (GNU directive), 61
- DC16 (IAR directive), 112
- DC24 (IAR directive), 113
- DC32 (IAR directive), 114
- DC8 (IAR directive), 111
- DCB (IAR directive), 111
- DCD (IAR directive), 114
- DCI16 (IAR directive), 115
- DCI32 (IAR directive), 116
- DCID (IAR directive), 116
- DCIW (IAR directive), 115
- DCW (IAR directive), 112
- define (command line), 35
- dependency-file (command line), 17
- DS (IAR directive), 117
- DS16 (IAR directive), 118
- DS24 (IAR directive), 119
- DS32 (IAR directive), 120
- DS8 (IAR directive), 117
- DSD (IAR directive), 120
- DSW (IAR directive), 118
- .ERR (GNU directive), 96
- .EXTERN (GNU directive), 65
- .GLOBAL (GNU directive), 65
- .GLOBL (GNU directive), 65
- GNU syntax,
 - .2BYTE, 75
 - .4BYTE, 76
 - .ABORT, 97
 - .ALIGN, 82
 - .ARM, 58
 - .ASCII, 73
 - .ASCIZ, 74
 - .BSS, 62
 - .BYTE, 73
 - .CODE, 59
 - .DATA, 61
 - .ERR, 96
 - .EXTERN, 65
 - .GLOBAL, 65
 - .GLOBL, 65
 - .HIDDEN, 68
 - .HWORD, 77
 - .IDENT, 98
 - .IF, 84
 - .IFDEF, 90
 - .IFEQ, 85
 - .IFGE, 86
 - .IFGT, 87
 - .IFLE, 88
 - .IFLT, 89
 - .IFNDEF, 91
 - .IFNE, 84
 - .IFNOTDEF, 91
 - .INCBIN, 57
 - .INST, 79
 - .INST.N, 80
 - .INST.W, 81
 - .INT, 78
 - .INTERNAL, 70
 - .LIST, 92
 - .LOCAL, 66
 - .LONG, 78
 - .LTORG, 99
 - .NOLIST, 93
 - .P2ALIGN, 83
 - .P2ALIGNL, 83
 - .P2ALIGNW, 83
 - .POOL, 99
 - .PROTECTED, 69
 - .RODATA, 64
 - .SHORT, 77
 - .SIZE, 72
 - .STRING, 74
 - .SUBTTL, 95
 - .SYNTAX, 100
 - .TEXT, 63
 - .THUMB, 60
 - .TITLE, 94
 - .TYPE, 71
 - .WEAK, 67

- .WORD, 78
- .HIDDEN (GNU directive), 68
- .HWORD (GNU directive), 77
- IAR syntax,
 - ARM, 108
 - CODE, 109
 - CODE16, 102, 110
 - CODE32, 103, 108
 - DC16, 112
 - DC24, 113
 - DC32, 114
 - DC8, 111
 - DCB, 111
 - DCD, 114
 - DCI16, 115
 - DCI32, 116
 - DCID, 116
 - DCIW, 115
 - DCW, 112
 - DS, 117
 - DS16, 118
 - DS24, 119
 - DS32, 120
 - DS8, 117
 - DSD, 120
 - DSW, 118
 - SUBT, 106
 - THUMB, 110
 - TTL, 105
- .IDENT (GNU directive), 98
- .IF (GNU directive), 84
- .IFDEF (GNU directive), 90
- .IFEQ (GNU directive), 85
- .IFGE (GNU directive), 86
- .IFGT (GNU directive), 87
- .IFLE (GNU directive), 88
- .IFLT (GNU directive), 89
- .IFNDEF (GNU directive), 91
- .IFNE (GNU directive), 84
- .IFNOTDEF (GNU directive), 91
- .INCBIN (GNU directive), 57
- include (command line), 14
- input-syntax (command line), 36
- .INST (GNU directive), 79
- .INST.N (GNU directive), 80
- .INST.W (GNU directive), 81
- .INT (GNU directive), 78
- .INTERNAL (GNU directive), 70
- keep-empty-sections (command line), 17
- keep-local-symbols (command line), 19
- .LIST (GNU directive), 92
- list-diagnostics (command line), 37
- list-file (command line), 39
- list-form-feed (command line), 40
- list-html (command line), 42
- list-page-length (command line), 43
- list-symbols (command line), 44
- list-text (command line), 46
- .LOCAL (GNU directive), 66
- .LONG (GNU directive), 78
- .LTORG (GNU directive), 99
- mapping-symbols (command line), 21
- marm (command line), 29
- mbig-endian (command line), 30
- mlittle-endian (command line), 31
- mthumb (command line), 32
- no-auto-import (command line), 34
- no-keep-empty-sections (command line), 18
- no-keep-local-symbols (command line), 20
- no-list-diagnostics (command line), 38
- no-list-form-feed (command line), 41
- no-list-symbols (command line), 45
- no-mapping-symbols (command line), 22
- no-pad-sections (command line), 27
- no-remarks (command line), 50
- no-symbols (command line), 25
- no-warnings (command line), 54
- .NOLIST (GNU directive), 93
- output (command line), 23
- .P2ALIGN (GNU directive), 83
- .P2ALIGNL (GNU directive), 83
- .P2ALIGNW (GNU directive), 83
- pad-sections (command line), 26
- .POOL (GNU directive), 99
- .PROTECTED (GNU directive), 69
- remarks (command line), 47
- remarks-are-errors (command line), 48
- remarks-are-warnings (command line), 49
- .RODATA (GNU directive), 64
- .SHORT (GNU directive), 77
- silent (command line), 51
- .SIZE (GNU directive), 72
- .STRING (GNU directive), 74
- SUBT (Arm directive), 106
- .SUBTTL (GNU directive), 95
- symbols (command line), 24
- .SYNTAX (GNU directive), 100
- .TEXT (GNU directive), 63
- .THUMB (GNU directive), 60
- THUMB (IAR directive), 110
- .TITLE (GNU directive), 94
- TTL (Arm directive), 105
- .TYPE (GNU directive), 71
- verbose (command line), 52
- via (command line), 15
- warnings (command line), 53
- warnings-are-errors (command line), 54
- .WEAK (GNU directive), 67
- .WORD (GNU directive), 78