NAME

`assembler` - assemble code to self-written RISC-V based CPU

SYNOPSIS

`assembler` [*OPTIONS*...] `--input` *file*...
`assembler` `-i`|`--input` *file*...
`assembler` `-i`|`--input` *file*... [`-o`|`--output` *file*]
`assembler` `-i`|`--input` *file*... [`-f`|`--format` [`debug`|`mif`|`raw`]]
`assembler` `-i`|`--input` *file*... [`-f`|`--format` `mif`] [`-c`]

DESCRIPTION

Assembler translates assembly code to machine code for a self-written RISC-V based CPU. assembler-format(5) is based on a modified MIPS syntax that includes instructions and macros from RISC-V, especially RV32I and RV32M.

By default, `assembler` translates one or more input *file*s to binary or MIF output files. The `-f` or `--format` option dictates which format should be used for the output. The output is either binary (`raw`), MIF in accordance to src_mif(5) (`mif`) or none (`debug`), in which case it will be printed to stderr(3). The MIF format is used by default. Output files are named after and written to the same directory as input *file*s.

The `-o` or `--output` option change the directory and file name of the output files. If the input files contain .data sections, a second output *file* is generated that contains the data.

WARNING: Parser errors currently are very rudimentary, not simple and not helpful. Make sure to use assembler-format(5) correctly. As last resort, you may open an issue in our repository with sufficient details.

FILES

The `assembler` command expects input to be valid assembler-format(5) code. Source files are normally named *name*.asm (ex. `example.asm`).

Source files must be in ASCII or UTF-8 encoding. Other encodings have not been tested and may not work.

When assembling output files, destination filenames are either a.*ext* for text output and a.mem.*ext* for data output or *name* for text output and *name*.mem.*ext* for data output, if the `-o` or `--output` option is used. Data output is only generated, if .data sections are used in the input *file*s. If specifying a output with an extension, the text output is written to that location but the data output is always written to the stem of the filename with the extension mem.*ext*. For example, executing `assembler -i example.asm -o test.example` will write the text output to `test.example` and the code output to `test.mem.mif`.

OPTIONS

These options control the format, location and type of the output.

`-f`=[`raw`|`mif`|`debug`], `--format`=[`raw`|`mif`|`debug`]
: Specify the format of the output. The default is `mif`. `debug` only prints the output to stderr(3). As the name implies, it should only be used for debugging purposes. Do not expect stability in the format.

  `raw` writes the machine code and data as binary to the output files.

  `mif` writes and formats the machine code and data as MIF, see src_mif(5) for details. The MIF format can be commented with the instruction assembly names using option `-c` or `--comment`. The memory *depth* and word *width* can be changed using `--depth` and `--width` respectively.

`-c`, `--comment`
: Indicate that MIF output should be commented. By default MIF output is not commented. When used with `-f=mif` or `--format=mif`, every machine code instruction includes a human readable representation as comment. Note that pseudo instructions or macros are not represented as such and only hardware instructions are used for representation in comments.

`--depth`=*depth*
: Sets the memory *depth* for the MIF format. By default *depth* equals 1024. Valid values are between 1 and 65535 (including). See src_mif(5) for details.

`--width`=[8|32]
: Sets the word *width* for the MIF format. By default *width* equals 32 (bit). See src_mif(5) for details.

`-o`=*file*, `--output`=*file*
: Write output to *file* instead of the default location `a`.*ext*. Does not do anything if `-f=debug` or `--format=debug`. If .data sections are used in input files, then data output will be written to the directory and stem of filename with the extension of `.mem`.*ext*. If `-f=mif` or `--format=mif`, then *ext*=`mif`, otherwise *ext*=`bin`.

These options control how the assembly code is assembled.

`--no-nop-insertion`
: Indicate that no nop insertions should be done by the assembler. By default nop's are inserted to circumvent data, control and memory hazards. By using this flag, subroutines cannot be used since they contain hazards!

`--no-sp-init`
: By default the stack is initialized to 4096. This flag disallows stack initialization. Note that the stack must be initialized when using stack operations.

Input option:

`-i`=*file*,[*file*]..., `--input`=*file*,[*file*]...
: The list of input assembly files to use for assembling. At least one input file must be used. Multiple input files can be used and must be separated by a space.

  The input files will be linked together in the order of specification in this option. The first file specified comes first, the last file specified comes last.

Miscellaneous options:

`-h, --help`
: Show help.

`-v, --version`
: Show version and exit.

ENVIRONMENT

`RUST_LOG`
: Used to set the log level for the assembler. Valid log levels are [`warn|error|info|debug|tracing`]. Different log levels for different modules may also be specified by using *module=log_level*. Only `debug` and below are used currently. Expect log output format to change. See https://docs.rs/env_logger/0.11.1/env_logger/ for more information on logging.

EXIT STATUS

This section is WIP. Some operational errors will have different error codes. These are going to be documented here.

`0`
: Successful program execution.

`1`
: General error.

EXAMPLES

Assemble one assembly file to output in MIF format:

```
$ assembler -i example.asm
Assembled a.mif (/path/to/example)
Finished [==========================================================] 5/5 Success
```

Assemble more assembly files to output in binary format:

```
$ assembler -i example.asm example2.asm example3.asm -f raw
Assembled a.mif (/path/to/example)
Finished [==========================================================] 5/5 Success
```

Assemble one assembly file to output in MIF format, which is commented and uses 8 bit word width:

```
$ assembler -i example.asm --width 8 -c
Assembled a.mif (/path/to/example)
Finished [==========================================================] 5/5 Success
```

Assemble one assembly file, view debug messages and do not output anything to files:

```
$ RUST_LOG=debug assembler -i example.asm -f debug
```

BUGS

https://git.mafiasi.de/Prj-MR/Assembler/issues

COPYRIGHT

AUTHORS

Written by Steven Becker and Jan Julius.

SEE ALSO

src_mif(5), stderr(3), assembler-format(5)