Package: c64asm (via r-universe)

October 16, 2024

Type Package

Title 6502 Assembler

Version 0.1.4

Author mikefc

Maintainer mikefc <mikefc@coolbutuseless.com>

Description A simple 6502 assembler written purely in R and leveraging R data structures for pre-computing character sets and images.

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 2.10)

Imports stringi, dplyr, purrr

Suggests testthat, knitr, rmarkdown, jpeg, magick, glue

VignetteBuilder knitr

Repository https://coolbutuseless.r-universe.dev

RemoteUrl https://github.com/coolbutuseless/c64asm

RemoteRef HEAD

RemoteSha 5ff248e03451e95771fa865873996a4378ae5048

Contents

`asm_patterns` *Regex patterns for parsing 6502 assembly*

Description

Regex patterns for parsing 6502 assembly

Usage

```
asm_patterns
```

Format

An object of class `character` of length 24.

`compile` *Compile ASM to a c64 exectuable (in PRG format)*

Description

Compile ASM to a c64 exectuable (in PRG format)

Usage

```
compile(asm, prg = TRUE)
```

Arguments

`asm`	single character string containing all ASM (including newlines)
`prg`	Should compilation return the bytes of the compiled PRG? Default: TRUE. If FALSE, return the main assembler data.frame which is more useful for debugging.

Value

prg bytes as a raw vector or `prg_df` main assembler data.frame

`create_line_tokens` *Split each ASM line into tokens*

Description

Split each ASM line into tokens

Usage

```
create_line_tokens(asm)
```

Arguments

`asm`	a single character string containing full ASM code, including newlines

Value

A list where element contains the tokens for a single row. Rows with just comments or whitespace are discarded

`create_prg_df` *Create the main compiler datastructure 'prg_df' from a list of 'line_tokens'*

Description

Create the main compiler datastructure 'prg_df' from a list of 'line_tokens'

Usage

```
create_prg_df(line_tokens)
```

Arguments

`line_tokens`	list of line tokens created by 'create_line_tokens()'

Value

prg_df the main compiler datastructure

`create_prg_df_row` *Determine the opcode, address mode etc for a sequence of tokens representing a single instruction*

Description

Determine the opcode, address mode etc for a sequence of tokens representing a single instruction

Usage

```
create_prg_df_row(tokens)
```

Arguments

`tokens`	A sequence of ASM tokens for one row of the ASM file.

Value

single row data.frame containing information about the row

`extract_prg_bytes` *Extract the raw bytes for the assembled PRG*

Description

Extract the raw bytes for the assembled PRG

Usage

```
extract_prg_bytes(prg_df)
```

Arguments

`prg_df`	main assembler data.frame

Value

raw vector of bytes

get_opcode_info — *Determine the opcode, address mode etc for a sequence of tokens representing a single instruction*

Description

Determine the opcode, address mode etc for a sequence of tokens representing a single instruction

Usage

```
get_opcode_info(tokens)
```

Arguments

tokens	A sequence of ASM tokens. This must start with a valid opcode, and not be a ‘symbol‘ or an assembler directive like ‘.text‘

Value

single row data.frame containing information about the instruction

h2i — *Hex (character strings) to integer values*

Description

Any leading ’#’ will be dropped. Values prefixed with "$" will be treated as hexadecimal. Values with a leading ’b’ will be treated as binary (MSB bit first)

Usage

```
h2i(h)
```

Arguments

h	character vector of values e.g. "#$1a", "$12", "#127", "#b00111111"

Value

vector of integers

hi — *Fetch the high byte of a 16bit address*

Description

Fetch the high byte of a 16bit address

Usage

```
hi(address)
```

Arguments

address	address from $0000 to $ffff

Value

integer value for the high byte of this address

lo — *Fetch the low byte of a 16bit address*

Description

Fetch the low byte of a 16bit address

Usage

```
lo(address)
```

Arguments

address	address from $0000 to $ffff

Value

integer value for the low byte of this address

opcodes *Reference list of opcode information*

Description

A dataset containing opcode information for 6502 instructions

Usage

```
opcodes
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 217 rows and 15 columns.

Source

http://www.oxyron.de/html/opcodes02.html

process_symbols *Process any cross-referenced symbols to insert actual addresse/values*

Description

Process any cross-referenced symbols to insert actual addresse/values

Usage

```
process_symbols(prg_df)
```

Arguments

`prg_df`	main assembler data.frame created by ‘create_prg_df()‘

Value

udpated prg_df with actual addresses added for each row, and cross-referenced values resolved

process_zero_padding *Pad out with zero bytes between address blocks*

Description

Pad out with zero bytes between address blocks

Usage

```
process_zero_padding(prg_df)
```

Arguments

prg_df	main assembler data.frame after ‘process_symbols()‘ has been run

Value

prg_df updated with zero padding rows between address blocks so that every byte from start to finish is accounted for

s2i *String to ASM for ".text" code*

Description

This is a complete hack to create CBMASCII bytes from plain text

Usage

```
s2i(string)
```

Arguments

string	single string

Value

Vector of integers

save_prg *Save the PRG to file*

Description

Save the PRG to file

Usage

```
save_prg(prg_df, filename)
```

Arguments

prg_df	main assembler data.frame
filename	c64 PRG file to write to

w2b *Convert a 16bit address to 2 bytes (lo_byte, hi_byte) as is the 6502 way*

Description

Convert a 16bit address to 2 bytes (lo_byte, hi_byte) as is the 6502 way

Usage

```
w2b(address)
```

Arguments

address	address from $0000 to $ffff

Value

integer vector of length 2 with (lo_byte, hi_byte)

Index