# Package: rbytecode (via r-universe)

August 26, 2024

**Type** Package

**Title** R Byte Code Assembler/Disassembler

**Version** 0.1.1

**Maintainer** Mike Cheng <mikefc@coolbutuseless.com>

**Description** Assembler/Disassembler for R's byte code.

**URL** https://github.com/coolbutuseless/rbytecode

**BugReports** https://github.com/coolbutuseless/rbytecode/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Imports** compiler, stringr

**Depends** R (>= 2.10)

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

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

**RemoteUrl** https://github.com/coolbutuseless/rbytecode

**RemoteRef** HEAD

**RemoteSha** 0b47240dbd62509fb29049e20050adb0dfebdcce

## Contents

---

`as.character.bcdf` *Convert a bytecode data.frame to a text representation*

---

## Description

Convert a bytecode data.frame to a text representation

## Usage

```
## S3 method for class 'bcdf'
as.character(x, incl_expr = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | bytecode data.frame produced by `dis()`, `disq()` and `parse_code()` |
| `incl_expr` | Include the stored expression? Default: FALSE |
| `...` | ignored |

## Examples

```
## Not run:
as.character(disq(1 + 1))

## End(Not run)
```

---

`asm` *Compile bytecode assembly into an R bytecode object*

---

## Description

Compile bytecode assembly into an R bytecode object

## Usage

```
asm(code, expr_default = quote(stop("[rbytecode]")))
```

## Arguments

| | |
|---|---|
| `code` | R bytecode assembly code as a single string |
| `expr_default` | The default expression stored with the bytecode. In general this should be a quoted ‘call‘ object, and if you are running a standard modern version of R, then it will almost never get evaluated/called. Default: quote(stop("[rbytecode])). |

## Value

bytecode object

## Examples

```
## Not run:
code <- "LDCONST 1\nLDCONST 2\nADD\nRETURN"
bc <- asm(code)
eval(bc) # 3

## End(Not run)
```

---

`compile_bcdf` *Compile a bytecode data.frame to an executable R bytecode object*

---

## Description

Compile a bytecode data.frame to an executable R bytecode object

## Usage

```
compile_bcdf(bcdf, expr_default = quote(stop("[rbytecode]")))
```

## Arguments

| | |
|---|---|
| `bcdf` | bytecode data.frame |
| `expr_default` | The default expression stored with the bytecode. In general this should be a quoted 'call' object, and if you are running a standard modern version of R, then it will almost never get evaluated/called. Default: quote(stop("[rbytecode])). |

## Value

bytecode object

## Examples

```
## Not run:
code <- "LDCONST 1\nLDCONST 2\nADD\nRETURN"
bcdf <- parse_code(code)
bc   <- compile_bcdf(bcdf)
eval(bc)

## End(Not run)
```

## dis *Disassemble R objects to a bytecode data.frame*

### Description

Disassemble R objects to a bytecode data.frame

### Usage

```
dis(x, incl_expr = FALSE, depth = 0L, env = NULL)
```

### Arguments

| | |
|---|---|
| x | Possible inputs: language object, call object, compiled bytecode, functionquoted expression. |
| incl_expr | include the expression index in the output. Default: FALSE. This is an advanced feature |
| depth | internal variable tracking recursion depth. Not usually set by user. |
| env | environment for keeping track of labels. Not usually set by user. |

### Value

bytecode data.frame (bcdf)

### Examples

```
## Not run:
dis(quote(1 + x))
fc <- compiler::cmpfun(\(x) {2 * x + 1})
dis(fc)

## End(Not run)
```

## disq *A wrapper for* dis() *which accepts unquoted expressions.*

### Description

A wrapper for dis() which accepts unquoted expressions.

### Usage

```
disq(x, incl_expr = FALSE)
```

### Arguments

| | |
|---|---|
| x | unquoted expression |
| incl_expr | include the expression index in the output. Default: FALSE. This is an advanced feature |

### Examples

```
## Not run:
disq(1 + 3)
dis(quote(1 + 3))

## End(Not run)
```

---

## eval_special *Evaluate an expression with particular handling for SPECIAL functions*

---

### Description

R has many SPECIAL functions in the base R package that should be called in bytecode with CALLSPECIAL

### Usage

```
eval_special(bc)
```

### Arguments

| | |
|---|---|
| bc | bytecode object to be evaluated |

### Details

But given it's a bit difficult to construct a CALLSPECIAL without just writing actual R code, there are some SPECIALs which can be handled by wrapping the special functions in a non-special wrapper.

## ops *Information about all bytecode operators*

### Description

Information about all bytecode operators

### Usage

```
ops
```

### Format

An object of class `list` of length 129.

## parse_code *Parse R bytecode assembly into a bytecode data.frame*

### Description

Parse R bytecode assembly into a bytecode data.frame

### Usage

```
parse_code(code)
```

### Arguments

| | |
|---|---|
| `code` | Single string containing bytecode instructions |

### Value

bytecode data.frame (`bcdf`)

### Examples

```
## Not run:
code <- "LDCONST 1\nLDCONST 2\nADD\nRETURN"
parse_code(code)

## End(Not run)
```

print.bcdftxt *Print a bytecode data.frame*

## Description

Print a bytecode data.frame

## Usage

```
## S3 method for class 'bcdftxt'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | Character representation of a bytecode data.frame |
| ... | arguments passed to cat() |

## Examples

```
## Not run:
bcdf <- disq(1 + x)
txt  <- as.character(bcdf)
txt

## End(Not run)
```

# Index