bits

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 1 | Imported by: 18,566

Details Valid go.mod file · Redistributable license · Tagged version · Stable version · Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Documentation

Overview

Package bits implements bit counting and manipulation functions for the predeclared unsigned integer types.

Functions in this package may be implemented directly by the compiler, for better performance. For those functions the code in this package will not be used. Which functions are implemented by the compiler depends on the architecture and the Go release.

Index

func Len32(x uint32) (n int)
func Len64(x uint64) (n int)
func Len8(x uint8) int
func Mul(x, y uint) (hi, lo uint)
func Mul32(x, y uint32) (hi, lo uint32)
func Mul64(x, y uint64) (hi, lo uint64)
func OnesCount(x uint) int
func OnesCount16(x uint16) int
func OnesCount32(x uint32) int
func OnesCount64(x uint64) int
func OnesCount8(x uint8) int
func Rem(hi, lo, y uint) uint
func Rem32(hi, lo, y uint32) uint32
func Rem64(hi, lo, y uint64) uint64
func Reverse(x uint) uint
func Reverse16(x uint16) uint16
func Reverse32(x uint32) uint32
func Reverse64(x uint64) uint64
func Reverse8(x uint8) uint8
func ReverseBytes(x uint) uint
func ReverseBytes16(x uint16) uint16
func ReverseBytes32(x uint32) uint32
func ReverseBytes64(x uint64) uint64
func RotateLeft(x uint, k int) uint
func RotateLeft16(x uint16, k int) uint16
func RotateLeft32(x uint32, k int) uint32
func RotateLeft64(x uint64, k int) uint64
func RotateLeft8(x uint8, k int) uint8
func Sub(x, y, borrow uint) (diff, borrowOut uint)
func Sub32(x, y, borrow uint32) (diff, borrowOut uint32)
func Sub64(x, y, borrow uint64) (diff, borrowOut uint64)
func TrailingZeros(x uint) int
func TrailingZeros16(x uint16) int
func TrailingZeros32(x uint32) int
func TrailingZeros64(x uint64) int
func TrailingZeros8(x uint8) int

Examples

Add32
Add64
Div32
Div64
LeadingZeros16
LeadingZeros32
LeadingZeros64
LeadingZeros8

Len16
Len32
Len64
Len8
Mul32
Mul64
OnesCount
OnesCount16
OnesCount32
OnesCount64
OnesCount8
Reverse16
Reverse32
Reverse64
Reverse8
ReverseBytes16
ReverseBytes32
ReverseBytes64
RotateLeft16
RotateLeft32
RotateLeft64
RotateLeft8
Sub32
Sub64
TrailingZeros16
TrailingZeros32
TrailingZeros64
TrailingZeros8

Constants

View Source

```
const UintSize = uintSize
```

UintSize is the size of a uint in bits.

Variables

This section is empty.

Functions

func Add

added in go1.12

```
func Add(x, y, carry uint) (sum, carryOut uint)
```

Add returns the sum with carry of x, y and carry: sum = x + y + carry. The carry input must be 0 or 1; otherwise the behavior is undefined. The carryOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

func Add32

added in go1.12

```
func Add32(x, y, carry uint32) (sum, carryOut uint32)
```

Add32 returns the sum with carry of x, y and carry: sum = x + y + carry. The carry input must be 0 or 1; otherwise the behavior is undefined. The carryOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

▶ Example

func Add64

added in go1.12

```
func Add64(x, y, carry uint64) (sum, carryOut uint64)
```

Add64 returns the sum with carry of x, y and carry: sum = x + y + carry. The carry input must be 0 or 1; otherwise the behavior is undefined. The carryOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

▶ Example

func Div

added in go1.12

```
func Div(hi, lo, y uint) (quo, rem uint)
```

Div returns the quotient and remainder of (hi, lo) divided by y: quo = (hi, lo)/y, rem = (hi, lo)%y with the dividend bits' upper half in parameter hi and the lower half in parameter lo. Div panics for y == 0 (division by zero) or y <= hi (quotient overflow).

func Div32

added in go1.12

```
func Div32(hi, lo, y uint32) (quo, rem uint32)
```

Div32 returns the quotient and remainder of (hi, lo) divided by y: quo = (hi, lo)/y, rem = (hi, lo)%y with the dividend bits' upper half in parameter hi and the lower half in parameter lo. Div32 panics for y == 0 (division by zero) or y <= hi (quotient overflow).

▶ Example

func Div64

added in go1.12

```
func Div64(hi, lo, y uint64) (quo, rem uint64)
```

Div64 returns the quotient and remainder of (hi, lo) divided by y: quo = (hi, lo)/y, rem = (hi, lo)%y with the dividend bits' upper half in parameter hi and the lower half in parameter lo. Div64 panics for y == 0 (division by zero) or y <= hi (quotient overflow).

▶ Example

func LeadingZeros

```
func LeadingZeros(x uint) int
```

LeadingZeros returns the number of leading zero bits in x; the result is UintSize for x == 0.

func LeadingZeros16

```
func LeadingZeros16(x uint16) int
```

LeadingZeros16 returns the number of leading zero bits in x; the result is 16 for x == 0.

▶ Example

func LeadingZeros32

```
func LeadingZeros32(x uint32) int
```

LeadingZeros32 returns the number of leading zero bits in x; the result is 32 for x == 0.

▶ Example

func LeadingZeros64

```
func LeadingZeros64(x uint64) int
```

LeadingZeros64 returns the number of leading zero bits in x; the result is 64 for x == 0.

▶ Example

func LeadingZeros8

```
func LeadingZeros8(x uint8) int
```

LeadingZeros8 returns the number of leading zero bits in x; the result is 8 for x == 0.

▶ Example

func Len

```
func Len(x uint) int
```

Len returns the minimum number of bits required to represent x; the result is 0 for x == 0.

func Len16

```
func Len16(x uint16) (n int)
```

Len16 returns the minimum number of bits required to represent x; the result is 0 for x == 0.

▶ Example

func Len32

```
func Len32(x uint32) (n int)
```

Len32 returns the minimum number of bits required to represent x; the result is 0 for x == 0.

▶ Example

func Len64

```
func Len64(x uint64) (n int)
```

Len64 returns the minimum number of bits required to represent x; the result is 0 for x == 0.

▶ Example

func Len8

```
func Len8(x uint8) int
```

Len8 returns the minimum number of bits required to represent x; the result is 0 for x == 0.

▶ Example

func Mul

added in go1.12

```
func Mul(x, y uint) (hi, lo uint)
```

Mul returns the full-width product of x and y: (hi, lo) = x * y with the product bits' upper half returned in hi and the lower half returned in lo.

This function's execution time does not depend on the inputs.

func Mul32

added in go1.12

```
func Mul32(x, y uint32) (hi, lo uint32)
```

Mul32 returns the 64-bit product of x and y: (hi, lo) = x * y with the product bits' upper half returned in hi and the lower half returned in lo.

This function's execution time does not depend on the inputs.

▶ Example

func Mul64

added in go1.12

```
func Mul64(x, y uint64) (hi, lo uint64)
```

Mul64 returns the 128-bit product of x and y: (hi, lo) = x * y with the product bits' upper half returned in hi and the lower half returned in lo.

This function's execution time does not depend on the inputs.

▶ Example

func OnesCount

```
func OnesCount(x uint) int
```

OnesCount returns the number of one bits ("population count") in x.

▶ Example

func OnesCount16

```
func OnesCount16(x uint16) int
```

OnesCount16 returns the number of one bits ("population count") in x.

▶ Example

func OnesCount32

```
func OnesCount32(x uint32) int
```

OnesCount32 returns the number of one bits ("population count") in x.

▶ Example

func OnesCount64

```
func OnesCount64(x uint64) int
```

OnesCount64 returns the number of one bits ("population count") in x.

▶ Example

func OnesCount8

```
func OnesCount8(x uint8) int
```

OnesCount8 returns the number of one bits ("population count") in x.

▶ Example

func Rem

added in go1.14

```
func Rem(hi, lo, y uint) uint
```

Rem returns the remainder of (hi, lo) divided by y. Rem panics for y == 0 (division by zero) but, unlike Div, it doesn't panic on a quotient overflow.

func Rem32

added in go1.14

```
func Rem32(hi, lo, y uint32) uint32
```

Rem32 returns the remainder of (hi, lo) divided by y. Rem32 panics for y == 0 (division by zero) but, unlike Div32, it doesn't panic on a quotient overflow.

func Rem64

added in go1.14

```
func Rem64(hi, lo, y uint64) uint64
```

Rem64 returns the remainder of (hi, lo) divided by y. Rem64 panics for y == 0 (division by zero) but, unlike Div64, it doesn't panic on a quotient overflow.

func Reverse

```
func Reverse(x uint) uint
```

Reverse returns the value of x with its bits in reversed order.

func Reverse16

```
func Reverse16(x uint16) uint16
```

Reverse16 returns the value of x with its bits in reversed order.

▶ Example

func Reverse32

```
func Reverse32(x uint32) uint32
```

Reverse32 returns the value of x with its bits in reversed order.

▶ Example

func Reverse64

```
func Reverse64(x uint64) uint64
```

Reverse64 returns the value of x with its bits in reversed order.

▶ Example

func Reverse8

```
func Reverse8(x uint8) uint8
```

Reverse8 returns the value of x with its bits in reversed order.

▶ Example

func ReverseBytes

```
func ReverseBytes(x uint) uint
```

ReverseBytes returns the value of x with its bytes in reversed order.

This function's execution time does not depend on the inputs.

func ReverseBytes16

```
func ReverseBytes16(x uint16) uint16
```

ReverseBytes16 returns the value of x with its bytes in reversed order.

This function's execution time does not depend on the inputs.

▶ Example

func ReverseBytes32

```
func ReverseBytes32(x uint32) uint32
```

ReverseBytes32 returns the value of x with its bytes in reversed order.

This function's execution time does not depend on the inputs.

▶ Example

func ReverseBytes64

```
func ReverseBytes64(x uint64) uint64
```

ReverseBytes64 returns the value of x with its bytes in reversed order.

This function's execution time does not depend on the inputs.

▶ Example

func RotateLeft

```
func RotateLeft(x uint, k int) uint
```

RotateLeft returns the value of x rotated left by (k mod UintSize) bits. To rotate x right by k bits, call RotateLeft(x, -k).

This function's execution time does not depend on the inputs.

func RotateLeft16

```
func RotateLeft16(x uint16, k int) uint16
```

RotateLeft16 returns the value of x rotated left by (k mod 16) bits. To rotate x right by k bits, call RotateLeft16(x, -k).

This function's execution time does not depend on the inputs.

▶ Example

func RotateLeft32

```
func RotateLeft32(x uint32, k int) uint32
```

RotateLeft32 returns the value of x rotated left by (k mod 32) bits. To rotate x right by k bits, call RotateLeft32(x, -k).

This function's execution time does not depend on the inputs.

▶ Example

func RotateLeft64

```
func RotateLeft64(x uint64, k int) uint64
```

RotateLeft64 returns the value of x rotated left by (k mod 64) bits. To rotate x right by k bits, call RotateLeft64(x, -k).

This function's execution time does not depend on the inputs.

▶ Example

func RotateLeft8

```
func RotateLeft8(x uint8, k int) uint8
```

RotateLeft8 returns the value of x rotated left by (k mod 8) bits. To rotate x right by k bits, call RotateLeft8(x, -k).

This function's execution time does not depend on the inputs.

▶ Example

func Sub

added in go1.12

```
func Sub(x, y, borrow uint) (diff, borrowOut uint)
```

Sub returns the difference of x, y and borrow: diff = x - y - borrow. The borrow input must be 0 or 1; otherwise the behavior is undefined. The borrowOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

func Sub32

added in go1.12

```
func Sub32(x, y, borrow uint32) (diff, borrowOut uint32)
```

Sub32 returns the difference of x, y and borrow, diff = x - y - borrow. The borrow input must be 0 or 1; otherwise the behavior is undefined. The borrowOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

▶ Example

func Sub64

added in go1.12

```
func Sub64(x, y, borrow uint64) (diff, borrowOut uint64)
```

Sub64 returns the difference of x, y and borrow: diff = x - y - borrow. The borrow input must be 0 or 1; otherwise the behavior is undefined. The borrowOut output is guaranteed to be 0 or 1.

This function's execution time does not depend on the inputs.

▶ Example

func TrailingZeros

```
func TrailingZeros(x uint) int
```

TrailingZeros returns the number of trailing zero bits in x; the result is UintSize for x == 0.

func TrailingZeros16

```
func TrailingZeros16(x uint16) int
```

TrailingZeros16 returns the number of trailing zero bits in x; the result is 16 for x == 0.

▶ Example

func TrailingZeros32

```
func TrailingZeros32(x uint32) int
```

TrailingZeros32 returns the number of trailing zero bits in x; the result is 32 for x == 0.

▶ Example

func TrailingZeros64

```
func TrailingZeros64(x uint64) int
```

TrailingZeros64 returns the number of trailing zero bits in x; the result is 64 for x == 0.

▶ Example

func TrailingZeros8

```
func TrailingZeros8(x uint8) int
```

TrailingZeros8 returns the number of trailing zero bits in x; the result is 8 for x == 0.

▶ Example

Types

This section is empty.

Source Files

View all

bits.go bits_errors.go bits_tables.go