libzahl

Unless specified otherwise, returns are `void` and all parameters are of type `z_t`.

Initialisation

Initialise libzahl	`zsetup(env)`	must be called before any other function is used, `env` is a `jmp_buf` all functions will `longjmp` to — with value 1 — on error
Deinitialise libzahl	`zunsetup()`	will free any pooled memory
Initialise a	`zinit(a)`	call once before use in any other function
Deinitialise a	`zfree(a)`	must not be used again before reinitialisation

Error handling

Get error code	`zerror(a)`	returns `enum zerror`, and stores description in `const char **a`
Print error description	`zperror(a)`	behaves like `perror(a)`, `a` is a, possibly `NULL` or ε, `const char *`

Arithmetic

$a \leftarrow b + c$	`zadd(a, b, c)`	
$a \leftarrow b - c$	`zsub(a, b, c)`	
$a \leftarrow b \cdot c$	`zmul(a, b, c)`	
$a \leftarrow b \cdot c \mod d$	`zmodmul(a, b, c, d)`	$0 \le a \text{ sgn } bc < \lvert d \rvert$
$a \leftarrow b/c$	`zdiv(a, b, c)`	rounded towards zero
$a \leftarrow c/d$	`zdivmod(a, b, c, d)`	rounded towards zero
$b \leftarrow c \mod d$	`zdivmod(a, b, c, d)`	$0 \le b \text{ sgn } c < \lvert d \rvert$
$a \leftarrow b \mod c$	`zmod(a, b, c)`	$0 \le a \text{ sgn } b < \lvert c \rvert$
$a \leftarrow b^2$	`zsqr(a, b)`	
$a \leftarrow b^2 \mod c$	`zmodsqr(a, b, c)`	$0 \le a < \lvert c \rvert$
$a \leftarrow b^2$	`zsqr(a, b)`	
$a \leftarrow b^c$	`zpow(a, b, c)`	
$a \leftarrow b^c$	`zpowu(a, b, c)`	`c` is an `unsigned long long int`
$a \leftarrow b^c \mod d$	`zmodpow(a, b, c, d)`	$0 \le a \text{ sgn } b^c < \lvert d \rvert$
$a \leftarrow b^c \mod d$	`zmodpowu(a, b, c, d)`	ditto, `c` is an `unsigned long long int`
$a \leftarrow \lvert b \rvert$	`zabs(a, b)`	
$a \leftarrow -b$	`zneg(a, b)`	

Assignment

$a \leftarrow b$	`zset(a, b)`	
$a \leftarrow b$	`zseti(a, b)`	`b` is an `int64_t`
$a \leftarrow b$	`zsetu(a, b)`	`b` is a `uint64_t`
$a \leftarrow b$	`zsets(a, b)`	`b` is a decimal `const char *`
$a \leftrightarrow b$	`zswap(a, b)`	

Comparison

Compare a and b	`zcmp(a, b)`	returns `int` $\text{sgn}(a - b)$
Compare a and b	`zcmpi(a, b)`	ditto, `b` is an `int64_t`
Compare a and b	`zcmpu(a, b)`	ditto, `b` is a `uint64_t`
Compare $\lvert a \rvert$ and $\lvert b \rvert$	`zcmpmag(a, b)`	returns `int` $\text{sgn}(\lvert a \rvert - \lvert b \rvert)$

Bit operation

$a \leftarrow b \wedge c$	`zand(a, b, c)`	bitwise
$a \leftarrow b \vee c$	`zor(a, b, c)`	bitwise
$a \leftarrow b \oplus c$	`zxor(a, b, c)`	bitwise
$a \leftarrow \neg b$	`znot(a, b, c)`	bitwise, cut at highest set bit
$a \leftarrow b \cdot 2^c$	`zlsh(a, b, c)`	`c` is a `size_t`
$a \leftarrow b/2^c$	`zrsh(a, b, c)`	ditto, rounded towards zero
$a \leftarrow b \mod 2^c$	`ztrunc(a, b, c)`	ditto, a shares signum with b
Get number of used bits	`zbits(a)`	returns `size_t`, 1 if $a = 0$
Get index of lowest set bit	`zlsb(a)`	returns `size_t`, `SIZE_MAX` if $a = 0$
Is bit b in a set?	`zbtest(a, b)`	`b` is a `size_t`, returns `int`
$a \leftarrow b$, set bit c	`zbset(a, b, c, 1)`	`c` is a `size_t`
$a \leftarrow b$, clear bit c	`zbset(a, b, c, 0)`	ditto
$a \leftarrow b$, flip bit c	`zbset(a, b, c, -1)`	ditto
$a \leftarrow c/2^d$	`zsplit(a, b, c, d)`	`d` is a `size_t`, rounded towards zero
$b \leftarrow c \mod 2^d$	`zsplit(a, b, c, d)`	ditto, b shares signum with c

Conversion to string

Convert a to decimal	`zstr(a, b, c)`	returns the resulting `const char *` — `b` unless `b` is `NULL`, — c must be either 0 or at least the length of the resulting string but at most the allocation size of `b` minus 1
Get string length of a	`zstr_length(a, b)`	returns `size_t` length of a in radix b

Marshallisation

Marshal a into b	`zsave(a, b)`	returns `size_t` number of saved bytes, `b` is a `void *`
Get marshal-size of a	`zsave(a, NULL)`	returns `size_t`
Unmarshal a from b	`zload(a, b)`	returns `size_t` number of read bytes, `b` is a `const void *`

Number theory

$a \leftarrow \operatorname{sgn} b$	`zsignum(a, b)`	
Is a even?	`zeven(a)`	returns `int` 1 (true) or 0 (false)
Is a even?	`zeven_nonzero(a)`	ditto, assumes $a \neq 0$
Is a odd?	`zodd(a)`	returns `int` 1 (true) or 0 (false)
Is a odd?	`zodd_nonzero(a)`	ditto, assumes $a \neq 0$
Is a zero?	`zzero(a)`	returns `int` 1 (true) or 0 (false)
$a \leftarrow \gcd(c, b)$	`zgcd(a, b, c)`	$a < 0$ if $b < 0 \wedge c < 0$
Is b a prime?	`zptest(a, b, c)`	c runs of Miller–Rabin, returns `enum zprimality` `NONPRIME` (0) (and stores the witness in `a` unless `a` is `NULL`), `PROBABLY_PRIME` (1), or `PRIME` (2)

Randomness

$a \xleftarrow{\$} \mathbf{Z}_d$	`zrand(a, b, UNIFORM, d)`	`b` is a `enum zranddev`, e.g. `DEFAULT_RANDOM`, `FASTEST_RANDOM`