

### Legend

$\Gamma$ comptime	$\sigma$ []const u8	$\tau$ type
$\perp$ void	@V Vector	$\forall$ anytype
when $\forall$ is returned it refers to the type of the $\forall$ arguments		

### Compiler

```
@src() Source Location
@setAlignment Stack( $\Gamma$  alignment: u29)  $\perp$ 
@setCold( $\Gamma$  is_cold: bool)  $\perp$ 
@setExecutableQuota( $\Gamma$  new_quota: u32)  $\perp$ 
@setRuntimeSafety( $\Gamma$  safety_on: bool)  $\perp$ 
@in $\Gamma$ () bool
@trap() noreturn
@breakpoint()  $\perp$ 
@frameAddress() usize
@hasDecl( $\Gamma$  Container: $\tau$ ,  $\Gamma$  name:  $\sigma$ ) bool
@hasField( $\Gamma$  Container: $\tau$ ,  $\Gamma$  name:  $\sigma$ ) bool
@import( $\Gamma$  path:  $\sigma$ )  $\tau$ 
@export(decl,  $\Gamma$  opts: Export Options)  $\perp$ 
@extern( $T$ : $\tau$ ,  $\Gamma$  opts: Extern Options)  $T$ 
@prefetch(ptr:  $\forall$ ,  $\Gamma$  opts: Prefetch Options)  $\perp$ 
@returnAddress() usize
@field(lhs:  $\forall$ ,  $\Gamma$  name:  $\sigma$ ) (field)
@fieldParentPtr( $\Gamma$  fname:  $\sigma$ , ptr: * $T$ )  $\forall$ 
@alignOf( $\Gamma$  T: $\tau$ )  $\Gamma$ _int
@sizeof( $\Gamma$  T: $\tau$ )  $\Gamma$ _int
@bitSizeOf( $\Gamma$  T: $\tau$ )  $\Gamma$ _int
@bitOffsetOf( $\Gamma$  T: $\tau$ ,  $\Gamma$  field:  $\sigma$ )  $\Gamma$ _int
@bitOffsetOf( $\Gamma$  T: $\tau$ ,  $\Gamma$  field:  $\sigma$ )  $\Gamma$ _int
@call(mod: CallModifier, func:  $\forall$ , args:  $\forall$ )  $\forall$ 
@tagName(value:  $\forall$ ) [:0]const u8
@unionInit( $\Gamma$  Union: $\tau$ ,  $\Gamma$  active:  $\sigma$ , init) Union
@embedFile( $\Gamma$  path:  $\sigma$ ) *const [N:0]u8
@compileError( $\Gamma$  msg:  $\sigma$ ) noreturn
@errorMessage(err: anyerror) [:0]const u8
@errorReturnTrace() ?*StackTrace
@panic(msg:  $\sigma$ ) noreturn
@This()  $\tau$ 
```

### Data and Logic

```
@memcpy(n oalias dst, noalias src)  $\perp$ 
@memset(dest, elem)  $\perp$ 
@popCount(op erand:  $\forall$ )  $\forall$ 
@shlExact(value:  $T$ , shift_amt: Log2T)  $T$ 
@shlWithOverflow(a:  $\forall$ , amt: Log2T) struct {  $\forall$ , u1 }
@shrExact(value:  $T$ , shift_amt: Log2T)  $T$ 
@byteSwap(op erand:  $\forall$ )  $T$ 
@bitReverse(int eger:  $\forall$ )  $T$ 
@clz(op erand:  $\forall$ )  $\forall$ 
@ctz(op erand:  $\forall$ )  $\forall$ 
```

### C Interop

```
@cDefine( $\Gamma$  name:  $\sigma$ , value)  $\perp$ 
@cImport(expr) $\tau$ 
@cInclude( $\Gamma$  path:  $\sigma$ )  $\perp$ 
@cUndef( $\Gamma$  name:  $\sigma$ )  $\perp$ 
@cVaArg(op erand: *VaList,  $\Gamma$  T: $\tau$ )  $T$ 
@cVaCopy(src: *VaList) VaList
@cVaEnd(src: *VaList)  $\perp$ 
@cVaStart() VaList
```

### GPU

```
@workGroupId( $\Gamma$  dim: u32) u32
@workGroupSize( $\Gamma$  dim: u32) u32
@workItemId( $\Gamma$  dim: u32) u32
```

### Atomics

---

@fence (order: AO) ↓

@atomicLoad

(Γ T:τ, p: \*const T, Γ order: AO) T

@atomicStore

(Γ T:τ, p: \*T, val: T, Γ order: AO) ↓

@atomicRmw

(Γ T:τ, p: \*T, Γ op: ARO, op: T, Γ order: AO) T

@cmpxchgWeak

(Γ T:τ, p: \*T, exp: T, new: T, scs: AO, fail: AO) ?T

@cmpxchgStrong

(Γ T:τ, p: \*T, exp: T, new: T, scs: AO, fail: AO) ?T

---

AO = std.builtin.AtomicOrder

RMO = std.builtin.AtomicRmwOp

---



By **nyc** (nyc)  
[cheatography.com/nyc/](https://cheatography.com/nyc/)

Not published yet.  
Last updated 19th April, 2024.  
Page 2 of 3.

Sponsored by **CrosswordCheats.com**  
Learn to solve cryptic crosswords!  
<http://crosswordcheats.com>

### Casting

```
@as(Γ T:τ, expr) T
@bitCast(value: V) V
@ptrCast(value: V) V
@ptrFromInt(address: usize) V
@addrSpaceCast(ptr: V) V
@alignCast(ptr: V) V
@enumFromInt(integer: V) V
@intCast(int: V) V
@intFromBool(value: bool) u1
@intFromEnum(enum_tag union: V) V
@intFromError(err: V) Int
@intFromFloat(float: V) V
@intFromPtr(ptr: V) usize
@floatFromInt(int: V) V
@floatCast(value: V) V
@errorFromInt(value: Int) anyerror
@volatileCast(value: V) DestType
@constCast(value: V) DestType
@errorCast(value: V) V
@truncate(integer: V) V
```

### SIMD

```
@Vector(len: Γ_int, Element:τ)τ
@shuffle(Γ E:τ, a: @V, b: @V, Γ mask: @V) @V
@splat(scalar: V) V
@reduce(Γ op: ReduceOp, value: V) E
@select(Γ T:τ, pred: @V, a: @V, b: @V) @V
```

### Math

```
@setFloatMode(Γ mode: FloatMode) ↓
@max(a: T, b: T) T
@min(a: T, b: T) T
@addWithOverflow(a: V, b: V) struct { V, u1 }
@mulWithOverflow(a: V, b: V) struct { V, u1 }
@subWithOverflow(a: V, b: V) struct { V, u1 }
@divExact(num: T, denom: T) T
@divFloor(num: T, denom: T) T
```

### Math (cont)

```
@divTrunc(num: T, denom: T) T
@mulAdd(Γ T:τ, a: T, b: T, c: T) T
@mulAdd(Γ T:τ, a: T, b: T, c: T) T
@abs(value: V) V
@mod(num: T, denom: T) T
@rem(num: T, denom: T) T
@sqrt(value: V) V
@sin(value: V) V
@cos(value: V) V
@tan(value: V) V
@exp(value: V) V
@exp2(value: V) V
@log(value: V) V
@log2(value: V) V
@log10(value: V) V
@floor(value: V) V
@ceil(value: V) V
@trunc(value: V) V
@round(value: V) V
```

### \*WASM

```
@wasmMemorySize(i: u32) u32
@wasmMemoryGrow(i: u32, d: u32) i32
```

### Types

```
@Type(Γ info: Type)τ
@TypeOf(operand)
@typeName(Γ T:τ) Type
@typeName(T:τ) *const [N:0]u8
```

