bar - core	**description**	**irregular form**
|% *<arms>*	form a core with subject as the payload	
|~ {moss seed}	form an iron gate	
|= {moss seed}	form a gate, a dry one-armed core with sample	
|. seed	form a trap, a one-armed core with one arm $	
|- seed	form a trap and kick ("call") it	
|_ moss (map term foot)	form a door, a many-armed core with sample	
|* {moss seed}	form a gill, a wet one-armed core with sample	
|^ twig (map term foot)	form a core with battery and anonymous arm $ and kick it	
|: {seed seed}	form a core with burnt sample	
|? seed	form a lead trap	
buc - mold		
$: (list moss)	form a mold to recognize a tuple	{a/foo b/bar c/baz}
$= {@tas moss}	mold that wraps a face around another mold	foo/bar
$% (list {{aura @} moss})	mold recognizing a union tagged by head atom	
$@ {moss moss}	mold that normalizes a union tagged by depth	
$^ {moss moss}	mold that normalizes a union tagged by head depth	
$? (list moss)	mold that normalizes a generic union	?($foo $bar $baz)
$- {moss moss}	mold that normalizes to an example gate	
$_ seed	mold that normalizes to an example	_foo
cen - call		
%= {wing (list (pair wing seed))}	take a wing with changes	foo(x 1, y 2, z 3)
%_ {wing (list (pair wing seed))}	take a wing with changes, preserving type	
%~ {wing seed seed}	call with multi-armed door	~(arm core arg)
%- {seed seed}	call a gate (function)	(fun arg)
%. {seed seed}	call a gate, reversed	
%+ {seed seed seed}	call a gate with pair sample	
%^ {seed seed seed seed}	call a gate with triple sample	
%* {wing twig (list (pair wing seed))}	make with arbitrary twig	
col - cell		
:- {seed seed}	construct a cell (2-tuple)	[a b], a^b
:+ {seed seed seed}	construct a triple (3-tuple)	[a b c]
:^ {seed seed seed seed}	construct a quadruple (4-tuple)	[a b c d]
:* (list twig)	construct an n-tuple	[a b c d e …]
:~ (list seed)	construct a null-terminated list	~[a b c]
:_ {seed seed}	construct a cell, inverted	
dot - nock		
.* {seed seed}	evaluate with nock 2	
.? seed	check for cell or atom with nock 3	
.+ atom	increment an atom with nock 4	+(42)
.= {seed seed}	test for equality with nock 5	=(a b)
.^ {moss seed}	load from the arvo namespace with nock 11	
ket - cast		
^+ {seed seed}	typecast by example (seed)	
^- {moss seed}	typecast by mold	\`foo\`bar
^= {toga seed}	name a value	foo=bar
^? seed	convert any core to a lead core (bivariant)	
^| seed	convert a gold core to an iron core (contravariant)	
^~ seed	fold constant at compile time	
sem - make		
;; {moss seed}	normalize with a mold, asserting fixpoint	
;~ {seed (list seed)	glue a pipeline together with a product-sample adapter	
;: {seed (list seed)	call a binary function as an n-ary function	:(fun a b c d)
;/ seed	tape as XML element	
sig - hint		
~& {seed seed}	debugging printf	
~% {term wing (list {term seed}) seed}	jet registration	
~/ {term seed}	jet registration for gate with registered context	
~$ {term seed}	profiling hit counter	
~| {seed seed}	tracing printf	
~_ {seed seed}	user-formatted tracing printf	
~? {seed seed seed}	conditional debug printf	
~> $@(term {term seed}) seed}	raw hint, applied to computation	
~< $@(term {term seed}) seed}	raw hint, applied to product	

sig - hint (continued)	**description**	**irregular form**
~+ seed	cache a computation	
~= {seed seed}	detect duplicate	
~! {seed seed}	print type on compilation fail	
tis - flow		
=> {seed seed}	compose two twigs	
=^ {taco wing seed seed}	pin the head of a pair; change a leg with the tail	
=* {term seed seed}	define an alias	
=~ (list seed)	compose many twigs	
=< {seed seed}	compose two twigs, inverted	foo:bar
=+ {seed seed}	combine a new noun with the subject	
=- {seed seed}	combine a new noun with the subject, inverted	
=| {moss seed}	combine a defaulted mold with the subject	
=/ {taco seed seed}	combine a named and/or typed noun with the subject	
=; {taco seed seed}	combine a named and/or typed noun with the subject, inverted	
=. {wing seed seed}	change one leg in the subject	
=: {(list (pair wing seed)) seed}	change multiple legs in the subject	
wut - test		
?: {seed seed seed}	branch on a boolean test	
?< {seed seed}	negative assertion	
?> {seed seed}	positive assertion	
?- {wing (list (pair moss seed))}	switch against a union, with no default	
?+ {wing seed (list (pair moss seed))}	switch against a union, with a default	
?. {seed seed seed}	branch on a boolean test, inverted	
?~ {wing seed seed}	branch on whether a wing of the subject is null	
?@ {wing seed seed}	branch on whether a wing of the subject is an atom	
?^ {wing seed seed}	branch on whether a wing of the subject is a cell	
?= {moss wing}	test pattern match	
?! seed	logical not	!foo
?& (list seed)	logical and	&(foo bar baz)
?| (list seed)	logical or	|(foo bar baz)
zap - wild		
!! $~	crash	
!> seed	wrap a noun in its span (create a vase)	
!= seed	make the nock formula for a twig	
!? {@ seed}	restrict the hoon version	

other syntax

+1:[a b]	[a b]
+2:[a b]	a
+3:[a b]	b
+6:[a [b c]]	b
+7:[a [b c]]	c
.:[a b]	[a b]
-:[a b]	a
+:[a b]	b
+<:[a [b c]]	b
+>:[a [b c]]	c
-.core	battery
+.core	payload
+>.core	context (outer core)
+<.core	sample
^face	face in outer core
.	current subject
+	+:.
-	-:.
+>	+>:.
..arm	core in which ++arm is defined

~	0 (nil)
%.y &	yes (true)
%.n |	no (false)
`a	[~ a]
~[a b c]	[a b c ~]
[a b c]~	[[a b c] ~]
~2017.8.26	`@da`date
~marzod-taglux	`@p`pronounceable base-256 number
12.345.567	`@ud`decimal
--12.345.567	`@sd`signed decimal
0xdeadbeef	`@ux`hexadecimal
.1.23e4	`@rs`floating-point decimal
“hoon”	tape (text as list of characters)
%hoon ‘hoon’	cord (text as atom)
$hoon	mold form of cord
?=($hoon %hoon)	%.y
?=($hoon %loon)	%.n
foo+bar	[%foo bar]
/foo/bar	[%foo %bar ~] wire (path)