

Block construction			Nameless definition		
(		Start block for IF or WHILE	[		Start nameless definition
)		End block for IF or WHILE	]	v	End nameless definitions
Control flow					
;		End of Word	EX	v --	Run a word from address
Conditional					
0?	a -- a	is TOS=Zero? conditional	1?	a -- a	is TOS<>Zero? conditional
+?	a -- a	is TOS>=0?	-?	a -- a	is TOS<0?
<?	a b -- a	is a<b? remove TOS	>?	a b -- a	is a>b? remove TOS
=?	a b -- a	is a=b? remove TOS	>=?	a b -- a	is a>=b? remove TOS
<=?	a b -- a	is a<=b? remove TOS	<>?	a b -- a	is a<>b? remove TOS
AND?	a b -- c	is a AND b? remove TOS	NAND?	a b -- c	is a NAND b? remove TOS
BT?	a b c -- a	is a<=b<=c? remove TOS			
Stack movements					
DUP	a -- aa	duplicate TOS	DROP	a --	remove TOS
OVER	ab -- aba	duplicate Second of Stack	PICK2	abc -- abca	Pick 3 element
PICK3	abcd -- abcd a	Pick 4 element	PICK4	abcde -- abcde a	Pick 5 element
SWAP	ab -- ba	swap TOS and NOS	NIP	ab -- b	remove NOS
ROT	abc -- bca	Rotate 3 top element	2DUP	ab -- abab	Duplicate 2 values of top
2DROP	ab --	Remove 2 elements	3DROP	abc --	Remove 3 elements
4DROP	abcd --	Remove 4 elements	2OVER	abcd -- abcdab	Copy 2 lower elements
2SWAP	abcd -- cdab	Swap 4 elements			
Return Stack					
>R	a --	rstack: -- a	R>	-- a	rstack: a --
R@	-- a	rstack: a -- a			
Logic operators					
AND	a b -- c	c=a AND b	OR	a b -- c	c=a OR b
XOR	a b -- c	c=a XOR b	NOT	a -- b	b=NOT a
Arithmetic operators					
+	a b -- c	d=a+b	-	a b -- c	d=a-b
*	a b -- c	d=a*b	/	a b -- c	d=a/b
<<	a b -- c	d=a shift left b	>>	a b -- c	d=a shift right b
>>>	a b -- c	d=a shift right b w/o sign	MOD	a b -- c	d=a mod b
/MOD	a b -- c d	c=a/b d=a mod b	*/	a b c -- d	d=a*b/c - not bit loss
*>>	a b c -- d	d=(a*b)>>c - not bit loss	<</	a b c -- d	d=(a<<c)/b - not bit loss
NEG	a -- b	b=-a	ABS	a -- b	b= a
SQRT	a -- b	b=square root(a)	CLZ	a -- b	b=count lead zeros of a
Memory fetch and store					
@	a -- [a]	fetch dword address	C@	a -- byte[a]	fetch byte from address
W@	a -- word[a]	fetch word address	D@	a -- dword[a]	fetch dword address
@+	a -- b [a]	fetch qword and inc 8	C@+	a -- b byte[a]	fetch byte and inc 1
W@+	a -- b word[a]	fetch word and inc 2	D@+	a -- b dword[a]	fetch dword and inc 4
!	a b --	store A in address B	C!	a b --	store byte A in address B
W!	a b --	store word A in address B	D!	a b --	store dword A in address B
!+	a b -- c	store A in B and inc 8	C!+	a b -- c	store byte A in B and inc 1
W!+	a b -- c	store word A in B and inc 2	D!+	a b -- c	store dword A in B and inc 4
+	a b --	increment in mem B, A	C+	a b --	increment in mem B, byte A
W+	a b --	increment in mem B, word A	D+	a b --	increment in mem B, dword A
Auxiliary registers					
>A	a --	load register A	B>	-- a	push register B
A>	-- a	push register A	>B	a --	load register B
A+	a --	add to A	B+	a --	add to B
A@	-- a	fetch from A	B@	-- a	fetch from B
A!	a --	store in mem A	B!	a --	store in mem B
A@+	-- a	fetch A and inc 8	B@+	-- a	fetch B and inc 8
A!+	a --	store in mem A, inc 8	B!+	a --	store in mem A, inc 8
CA@	-- a	fetch from A	CB@	-- a	fetch from B
CA!	a --	store in mem A	CB!	a --	store in mem B

CA@+	-- a	fetch A and inc 1	CB@+	-- a	fetch B and inc 1
CA!+	a --	store in mem A, inc 1	CB!+	a --	store in mem A, inc 1
DA@	-- a	fetch from A	DB@	-- a	fetch from B
DA!	a --	store in mem A	DB!	a --	store in mem B
DA@+	-- a	fetch A and inc 4	DB@+	-- a	fetch B and inc 4
DA!+	a --	store in mem A, inc 4	DB!+	a --	store in mem A, inc 4
<b>Memory copy and fill</b>					
MOVE	d s c --	copy S to D, C qword	MOVE>	d s c --	copy from S to D, C qword in rev.
FILL	d v c --	fill D, C qword with V	CMOVE	d s c --	copy from S to D, C bytes
CMOVE>	d s c --	copy S to D, C bytes in rev.	CFILL	d v c --	fill from D, C bytes with V
DMOVE	d s c --	copy S to D, C dwords	DMOVE>	d s c --	copy from S to D, C dwords in rev.
DFILL	d v c --	fill D, C dwords with V			
<b>Operating System</b>					
MEM	-- a	start memory free			

Prefix	
:	define CODE, :: Export word
#	define DATA, ## Export word
^	Include source code in filename
'	Adress of word, code or data
	Commento to end of the line
"	String to next ", "" for " character
\$	Hex numbers
%	Binary numbers, 0 can be .

Data Definition	
qword	#var 0
qword list	#list 1 2 3 4 5
byte list	#blist ( 1 2 3 4 )
dword list	#dlist [ 1 2 3 4 ]
memory	#buffer * 1024   1kb size
vectors	#vector 'actionword
list jump	#listj 'a1 'a2 'a3

Control Flow	
REPEAT	( loop )
IF	?? ( true branch )
WHILE	( while ?? loop )
MULTI WHILE	( while ?? while ?? loop )
IF-ELSE	factoring to new word :ifelse ?? ( true ; ) false ;

Comment work like option switches	
WIN	in win, the line is not a comment
LIN	in linux,...
WEB	In web,...
MAC	In MAC,...
RPI	In Raspberry Pi,...
MEM 640	data memory size (in kb) min 1kb