CORE word set (133 words):

!	(x a-addr)	#	(ud1 ud2)	#>	(<i>xd c</i> - <i>addr u</i>)	#s	(ud1 ud2)
•	(" <spaces>name" xt)</spaces>	(Execution: ("ccc <paren>")</paren>	*	$(n1 u1 \ n2 u2 n3 u3)$	*/	$(n1 \ n2 \ n3 n4)$
*/MOD	(n1 n2 n3 n4 n5)	+	$(n1 u1 \ n2 u2 n3 u3)$	+!	$(n u \ a-addr)$		
+LOOP	Compilation: (C: do-sys) Run-ti		(R: <i>loop-sys1</i> <i>loop-sys2</i>)		(x-)	_	$(n1 u1 \ n2 u2 n3 u3)$
•	<u>(n)</u>	. "	Compilation: ("ccc <quote>"</quote>) Run-time	e: ()		$(n1 \ n2 n3)$
/MOD	$(n1 \ n2 - n3 \ n4)$	0<	(n flag)	0=	(x - flag)	1+	(n1 u1 - n2 u2)
<u>1-</u>	(n1 u1 - n2 u2)	2!	(x1 x2 a-addr)	2*	(x1-x2)	2/	(x1-x2)
20	(a-addr x1 x2)	2DROP	(x1 x2)	2DUP	(x1 x2 - x1 x2 x1 x2)	20VER	(x1 x2 x3 x4 x1 x2 x3 x4 x1 x2)
2SWAP	(x1 x2 x3 x4 x3 x4 x1 x2)	:	(C: " <spaces>name" colon-s</spaces>	vs) Initiati	ion: $(i*x i*x)(R: nest-sys)$ nam	e Executio	n: (i*x j*x)
<i>;</i>	Compilation: (C: colon-sys) Run	n-time: ()) (R: <i>nest-sys</i>)	<	(n1 n2 - flag)	<#	<u>()</u>
=	(x1 x2 - flag)	>	(n1 n2 flag)	>BODY	(<i>xt a-addr</i>)	<u>>IN</u>	<u>(a-addr)</u>
>NUMBE	R (udl c-addrl ul ud2 c-addr2 u2)		>R	Execution: $(x)(R x)$?DUP	$(x-0 \mid xx)$
<u>@</u>	(a-addr x)	ABORT	(i*x)(R:j*x)	ABORT"	Compilation: ("ccc <quote>") R</quote>	<u>lun-time: (</u>	$i^*x x 1 - i^*x (R: j^*x - j^*x)$
ABS	(n-u)	ACCEPT	(c-addr + n1 - + n2)	ALIGN	<u>()</u>	ALIGNE	D (addr a-addr)
ALLOT	<u>(n)</u>	AND	$(x1 \ x2 - x3)$	BASE	<u>(a-addr)</u>		
BEGIN	Compilation: (C: dest) Run-time			BL	<u>(char)</u>	<u>C!</u>	<u>(char c-addr)</u>
<u>C,</u>	(char)	C@	<u>(c-addr char)</u>	CELL+	<u>(a-addr1 a-addr2)</u>	CELLS	(n1 - n2)
CHAR	(" <spaces>name" char)</spaces>	CHAR+	<u>(c-addr1 c-addr2)</u>	CHARS	(n1 - n2)		
CONSTA				COUNT	(c-addr1 c-addr2 u)	CR	<u>()</u>
CREATE	_ · · · · · · · · · · · · · · · · · · ·			DECIMA	<u>L ()</u>	DEPTH	(+n)
DO	Compilation: (C: do-sys) Run-ti						
DOES>	Compilation: (C: colon-sys1 colo	n-svs2) Ri	in time: $()(\mathbf{R} \cdot \mathbf{noct} \mathbf{cucl}) $ In	itiation: (i*v i*vaaddv)(D· nacteue?) v	ame Execu	ution: (i*r i*r)
		• /					
DROP	(x-)	DUP	(x-xx)	ELSE	Compilation: (C: orig1 orig2)	Run-time: (<u></u>)
DROP EMIT	(x) (x)	DUP ENVIRON	<u>(x x x)</u> IMENT? (c-addr u false i*x tr	ELSE ue)	Compilation: (C: orig1 orig2) F	Run-time: (EVALUA) TE (i*x c-addr u j*x)
DROP EMIT EXECUT	$\frac{(x)}{(x)}$ $\underline{E}(i^*xxtj^*x)$	DUP ENVIRON EXIT	(xxx) NMENT? $(c-addr u false \mid i*x tr$ Execution: $()(R: nest-sys)$	ELSE ue) FILL	Compilation: (C: orig1 orig2) F	Run-time: (<u></u>)
DROP EMIT EXECUTI FM/MOD		DUP ENVIRON EXIT HERE	<u>(x x x)</u> IMENT? (c-addr u false i*x tr	ELSE ue) FILL HOLD	Compilation: (C: orig1 orig2) F (c-addr u char) (char)	Run-time: (EVALUA FIND) TE (i*x c-addr u j*x)
DROP EMIT EXECUT FM/MOD I	(x) (x) $E(i^*x xt j^*x)$ (dl nl n2 n3) Execution: $(n u)$ (R: loop-sys	DUP ENVIRON EXIT HERE loop-sys)	(xxx) IMENT? (c-addr u false i*x tr Execution: () (R: nest-sys) (addr)	ELSE ue) FILL HOLD IF	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim	EVALUA FIND e: (x)) TE $(i*x c-addr u j*x)$ $(c-addr c-addr 0 xt 1 xt -1)$
DROP EMIT EXECUTI FM/MOD I IMMEDIA	(x) (x) $E(i^*x xt - j^*x)$ (dl nl n2 n3) Execution: $(n u)(R: loop-sys$ ATE $()$	DUP ENVIRON EXIT HERE loop-sys) INVERT	$\frac{(xxx)}{\text{IMENT?} (c-addr u false \mid i*x tr}$ $\text{Execution:} () (R: nest-sys)$ $\frac{(addr)}{(xlx2)}$	ELSE ue) FILL HOLD IF J	Compilation: (C: orig1 orig2) F (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: (n u) (R: loop-sys1 l	EVALUA FIND e: (x) oop-sys2) TE $(i*x c-addr u j*x)$ $(c-addr c-addr 0 xt 1 xt -1)$
DROP EMIT EXECUT: FM/MOD I IMMEDIA	$ \begin{array}{c} (x) \\ (x) \\ \hline (x) \\ \hline E (i^*x xt j^*x) \\ (dl nl n2 n3) \\ \hline \text{Execution: } (n u) \text{ (R: loop-sys} \\ \hline \text{ATE } () \\ (char) \end{array} $	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE	(xxx) IMENT? (c-addr u false i*x tr Execution: () (R: nest-sys) (addr) (x1 x2) Execution: () (R: loop-sys)	ELSE ue) FILL HOLD IF J LITERAL	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: (n u) (R: loop-sys1 l Compilation: (x) Run-time: (Eun-time: () TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP	(x) (x) $E(i^*x xt - j^*x)$ (dl nl n2 n3) Execution: $(n u)(R: loop-sys$ ATE $()$ (char) Compilation: $(C: do-sys)$ Run-ti	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () ()	(xxx) IMENT? (c-addr u false i*x tr Execution: () (R: nest-sys) (addr) (x1 x2) Execution: () (R: loop-sys) R: loop-sys1 loop-sys2)	ELSE ue) FILL HOLD IF J LITERAL LSHIFT	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: (n u) (R: loop-sys1 l Compilation: (x) Run-time: ((x1 u x2)	Run-time: () TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d)
DROP EMIT EXECUTI FM/MOD I IMMEDIA KEY LOOP MAX	$\begin{array}{c} (x) \\ (x) \\ \hline (x) \\ \hline E \ (i^*x xt j^*x) \\ (d1 \ n1 n2 \ n3) \\ \hline \text{Execution: } (n u) \ (R: loop-sys \\ \hline \textbf{ATE} \ () \\ \hline (char) \\ \hline \text{Compilation: } (C: do-sys) \ \text{Run-ti} \\ \hline (n1 \ n2 n3) \end{array}$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (]	(xxx) IMENT? (c-addr u false i*x tr Execution: () (R: nest-sys) (addr) (x1 x2) Execution: () (R: loop-sys R: loop-sys1 loop-sys2) (n1 n2 n3)	ELSE ue) FILL HOLD IF J LITERAT LSHIFT MOD	Compilation: (C: $orig1 orig2$) R ($c-addr \ u \ char$) ($char$) Compilation: (C: $ orig$) Run-time Execution: ($ n u$) (R: $loop-sys1\ l$ L Compilation: (x) Run-time: ($ (x1\ u x2)$) ($n1\ n2 n3$)	Eun-time: () TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUTI FM/MOD I IMMEDIA KEY LOOP MAX NEGATE	(x) (x) (x) (x) (x) $(i^*x xt - j^*x)$ (dl nl n2 n3) Execution: $(n u)(R: loop-sys$ ATE $()$ (char) Compilation: $(C: do-sys)$ Run-ti (nl n2 n3) (nl n2)	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR	(xxx) IMENT? (c-addr u false i*x tr Execution: () (R: nest-sys) (addr) (x1 x2) Execution: () (R: loop-sys) R: loop-sys1 loop-sys2)	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: $(-n u)$ (R: loop-sys1 l Compilation: (x) Run-time: (x) (x1 u x2) (n1 n2 n3) (x1 x2 x1 x2 x1)	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPOI	(x) (x) $E(i^*x xt j^*x)$ (d1 n1 n2 n3) Execution: $(n u)(R: loop-sys$ ATE $()$ (char) Compilation: $(C: do-sys)$ Run-ti (n1 n2 n3) (n1 n2) NE Compilation: $(" < spaces > name")$	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () (1 MIN OR '')	(xxx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: $(-n u)$ (R: loop-sys1 l L Compilation: (x) Run-time: $((x1 u x2))$ $(n1 n2 n3)$ $(x1 x2 x1 x2 x1)$ $()$ (R: $i*x$)	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@	(x) (x) $E(i^*x xt j^*x)$ (dl nl n2 n3) Execution: $(n u)$ (R: loop-sys ATE () (char) Compilation: (C: do-sys) Run-ti (nl n2 n3) (nl n2) NE Compilation: (" <spaces>name"</spaces>	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () (] MIN OR ") RECURSE	$(x-xx)$ IMENT? $(c-addr u false \mid i*x tr$ Execution: $()$ $(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()$ $(R: loop-sys)$ $R: loop-sys1 loop-sys2$ $(nl n2 n3)$ $(xl x2 x3)$ E Compilation: $()$	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-time Execution: $(-n u)$ (R: loop-sys1 l Compilation: (x) Run-time: $()$ (x1 u x2) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run-	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> 1-time: ()) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@ ROT	(x) (DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () (] MIN OR ") RECURSE RSHIFT	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(nl n2 n3)$ $(xl x2 x3)$ E Compilation: $()$ $(xl u x2)$	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S"	Compilation: (C: origl orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-time Execution: $(-n u)$ (R: loop-sys1 l Compilation: (x) Run-time: $()$ (x1 u x2) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R</quote>	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> a-time: () cun-time: () TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x) 1 c-addr u)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@ ROT S>D	(x) (x) (x) (x) (x) (x) (x) (x) $(dl nl n2 n3)$ $(nl n2 n3)$ $(nl n2)$ Execution: $(n u)$ (R: loop-sys $(char)$ Compilation: $(char)$ $(char)$ Compilation: $(char)$ $(ch$	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () (] MIN OR ') RECURSE RSHIFT SIGN	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(x1x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n)	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S" SM/REM	Compilation: (C: origl orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: (n u) (R: loop-sys1 l L Compilation: (x) Run-time: ((x1 u x2) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: i*x) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R (d1 n1 n2 n3)</quote>	e: (x) oop-sys2 x) M* MOVE R> altime: () cun-time: (SOURCE) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x)) c-addr u) (c-addr u)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@ ROT S>D SPACE	(x) (x)	DUP ENVIRON EXIT HERE loop-sys INVERT LEAVE me: () (] MIN OR ') RECURSE RSHIFT SIGN SPACES	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(x1x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n)	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S" SM/REM STATE	Compilation: (C: origl orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: ($n u$) (R: loop-sys1 l Compilation: (x) Run-time: ($(x1 u x2)$) ($n1 n2 n3$) ($x1 x2 x1 x2 x1$) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R ($d1 n1 n2 n3$) (a-addr)</quote>	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> a-time: () cun-time: (SOURCE SWAP) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x)) c-addr u) (x1 x2 x2 x1)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@ ROT S>D SPACE THEN	(x) (x) $(x$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR ") RECURSE RSHIFT SIGN SPACES E: ()	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(x1x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n)	ELSE ue) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S" SM/REM STATE TYPE	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: ($n u$) (R: loop-sys1 l Compilation: (x) Run-time: ((x1 u x2)) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R (d1 n1 n2 n3) (a-addr) (c-addr u)</quote>	Eun-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> n-time: () cun-time: (SOURCE SWAP U.) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUT! FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO! R@ ROT S>D SPACE THEN U<	(x) (x) $(x$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR ") RECURSE RSHIFT SIGN SPACES e: () UM*	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(x1x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n)	ELSE we) FILL HOLD IF J LITERAT LSHIFT MOD OVER QUIT REPEAT S" SM/REM STATE TYPE UM/MOD	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-tim Execution: ($n u$) (R: loop-sys1 l Compilation: (x) Run-time: ((x1 u x2)) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R (d1 n1 n2 n3) (a-addr) (c-addr u) (ud u1 u2 u3)</quote>	Run-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> n-time: () cun-time: (SOURCE SWAP U. UNLOOP) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x) c-addr u) (c-addr u) (x1 x2 x2 x1) (u) Execution: () (R: loop-sys)
DROP EMIT EXECUT! FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO! R@ ROT S>D SPACE THEN U< UNTIL	(x) (x) $(x$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR ") RECURSE RSHIFT SIGN SPACES E: () UM* E: (x)	(x-xx) IMENT? $(c-addr u false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n) (n)	ELSE we) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S" SM/REM STATE TYPE UM/MOD VARIABI	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-time Execution: ($n u$) (R: loop-sys1 l L Compilation: (x) Run-time: ((x1 u x2)) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R (d1 n1 n2 n3) (a-addr) (c-addr u) (ud u1 u2 u3) LE ("<spaces>name") name Ex</spaces></quote>	Eun-time: (EVALUA FIND e: (x) oop-sys2 x M* MOVE R> -time: () cun-time: (SOURCE SWAP U. UNLOOP secution: () TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1) - loop-sys1 loop-sys2) (n1 n2 d) (addr1 addr2 u) Execution: (x) (R: x) c-addr u) (c-addr u) (x1 x2 x2 x1) (u) Execution: () (R: loop-sys)
DROP EMIT EXECUT: FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO: R@ ROT S>D SPACE THEN U< UNTIL WHILE	(x) (x) (x) $E (i*x xtj*x)$ $(dl nl n2 n3)$ $Execution: (n u) (R: loop-sys$ $ATE ()$ $(char)$ $Compilation: (C: do-sys) Run-ti$ $(nl n2 n3)$ $(nl n2)$ $NE Compilation: ("name"$ $Execution: (x) (R: x x)$ $(xl x2 x3 x2 x3 x1)$ (nd) $()$ $Compilation: (C: orig) Run-time$ $(ul u2 flag)$ $Compilation: (C: dest) Run-time$ $(compilation: (C: dest) Run-time$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR ") RECURSE RSHIFT SIGN SPACES E: () UM* E: (x)	(x-xx) IMENT? $(c-addr u - false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()(R: loop-sys)$ $R: loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n) (n) $(u1 u2 ud)$	ELSE we FILL HOLD IF J LITERAI LSHIFT MOD OVER QUIT REPEAT SM/REM STATE TYPE UM/MOD VARIABI WORD	Compilation: (C: origl orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-time Execution: ($n u$) (R: loop-sys1 l L Compilation: (x) Run-time: ((x1 u x2)) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") Run Compilation: ("ccc<quote>") Run (d1 n1 n2 n3) (a-addr) (c-addr u) (ud u1 u2 u3) LE ("<spaces>name") name Ex (char "<chars>ccc<char>" c-addr</char></chars></spaces></quote></quote>	Run-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> a-time: () cun-time: (SOURCE SWAP U. UNLOOP Recution: (ddr)) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)
DROP EMIT EXECUT! FM/MOD I IMMEDI: KEY LOOP MAX NEGATE POSTPO! R@ ROT S>D SPACE THEN U< UNTIL	(x) (x) $(x$	DUP ENVIRON EXIT HERE loop-sys) INVERT LEAVE me: () (1 MIN OR ") RECURSI RSHIFT SIGN SPACES e: () UM* e: (x) Run-time:	(x-xx) IMENT? $(c-addr u - false i*x tr$ Execution: $()(R: nest-sys)$ $(addr)$ $(xl x2)$ Execution: $()(R: loop-sys)$ R: $loop-sys1 loop-sys2)$ $(n1 n2 n3)$ $(x1 x2 x3)$ E Compilation: $()$ $(x1 u x2)$ (n) (n) $(u1 u2 ud)$ Execution: $()$	ELSE we) FILL HOLD IF J LITERAL LSHIFT MOD OVER QUIT REPEAT S" SM/REM STATE TYPE UM/MOD VARIABI	Compilation: (C: orig1 orig2) R (c-addr u char) (char) Compilation: (C: orig) Run-time Execution: ($n u$) (R: loop-sys1 l L Compilation: (x) Run-time: ((x1 u x2)) (n1 n2 n3) (x1 x2 x1 x2 x1) () (R: $i*x$) Compilation: (C: orig dest) Run Compilation: ("ccc <quote>") R (d1 n1 n2 n3) (a-addr) (c-addr u) (ud u1 u2 u3) LE ("<spaces>name") name Ex</spaces></quote>	Run-time: (EVALUA FIND e: (x) oop-sys2 x) M* MOVE R> a-time: () cun-time: (SOURCE SWAP U. UNLOOP Recution: (ddr)) TE (i*x c-addr u j*x) (c-addr c-addr 0 xt 1 xt -1)

CORE EXT word set (46 words):

#TIB (a-addr)	. (Execution: ("ccc <paren>")</paren>		0 <> (x - flag)		
0 > (n - flag)	2>R Execution: (x1 x2) (R: x1				
Execution: $(-x1 x2)(R; x1 x2 -$					
: NONAME (C: colon-sys) (S: xt) Initiation	• •		$\langle \rangle$ $(x1 \ x2 flag)$		
	time: $(n1 u1 \ n2 u2) (R: loop-sys)$	AGAIN Compilation: (C: dest) Run-tir			
C" Compilation: ("ccc <quote>") I</quote>		CASE Compilation: (C: case-sys) Ru			
COMPILE , Execution: (xt)	CONVERT (d1 c-addr1 ud2 c-addr2)	ENDCASE Compilation: (C: case-sys) Ru			
ENDOF Compilation: (C: case-sys1 of-sys		ERASE (addr u)	EXPECT $(c-addr + n)$		
FALSE (false)	<u>HEX ()</u>	MARKER (" <spaces>name") name Exec</spaces>			
$NIP \qquad (x1 x2 x2)$	OF Compilation: (C: of-sys) R		$\underline{PAD} \qquad (c-addr)$		
PARSE $(char "ccc < char > " c - addr u)$	$\underline{\mathbf{PICK}} \qquad (xu \dots x1 \ x0 \ u xu \dots x1 \ x0 \ xu$		REFILL (flag)		
RESTORE-INPUT $(xn \dots xl \ n flag)$		u)SAVE-INPUT $(-xn \dots xl n)$	SOURCE-ID $(-0 \mid -1)$		
SPAN $(a-addr)$		nterpretation: (x " <spaces>name") Compilat</spaces>	*		
TRUE (true)	TUCK $(x1 x2 - x2 x1 x2)$	$\mathbf{U.R} \qquad (u n)$	U> $(u1 u2 flag)$		
<u>UNUSED (u)</u>	VALUE $(x " \le spaces > name") name$		WITHIN $(n1 u1 n2 u2 n3 u3 flag)$		
[COMPILE] Compilation: (" <spaces>name</spaces>	2")	\ Execution: ("ccc <eol>")</eol>			
BLOCK word set (8 words):	BLK (a-addr)	BLOCK $(u - a - addr)$	BUFFER $(u a - addr)$		
	EVALUATE $(i*x c-addr u j*x)$	FLUSH ()	LOAD $(i^*x u - j^*x)$		
	SAVE-BUFFERS ()	<u>UPDATE ()</u>			
BLOCK EXT word set (6 words):	EMPTY-BUFFERS ()	$\underline{\textbf{LIST}} \qquad (u)$	REFILL (flag)		
	$\underline{SCR} \qquad (a\text{-}addr)$	THRU $(i*x ul u2 - j*x)$	\ Execution: ("ccc <eol>")</eol>		
DOUBLE word set (20 words):					
200222 Word 00t (20 Wordo).					
2CONSTANT (x1 x2 " <spaces>name") name Execution: (x1 x2) 2LITERAL Compilation: (x1 x2) Run-time: (x1 x2)</spaces>					
2VARIABLE (" <spaces>name") name E</spaces>	xecution: (a-addr)	D+ (d1 ud1 d2 ud2 - d3 ud3)	D- $(d1 ud1 d2 ud2 d3 ud3)$		
\mathbf{D} . $(d-)$	$D.R \qquad (dn-)$	D0< (<i>d flag</i>)	D0 = (xd - flag)		
D2* $(xd1 - xd2)$	D2/ $(xd1 - xd2)$	D< (d1 d2 flag)	$\mathbf{D} = \frac{(xd1 xd2 flag)}{(xd1 xd2 flag)}$		
$\overline{D>S}$ $(d-n)$	$\overline{\text{DABS}} (d-ud)$	$\frac{\text{DMAX} \qquad (d1 \ d2 \ d3)}{}$	$\frac{\text{DMIN}}{(d1 d2 - d3)}$		
$\overrightarrow{\text{DNEGATE}} (d1 - d2)$	$\underline{\mathbf{M}} \star / \underbrace{(d1 n1 + n2 - d2)}$	$\mathbf{M+} \qquad (d1 ud1 \ n - d2 ud2)$			
DOUBLE EXT word set (2 words):	2ROT $(x1 x2 x3 x4 x5 x6 x3 x4 x5 x6)$	x6x1x2) DU< (udl ud2 flag)			
	•				
EXCEPTION word set (2 words):	CATCH $(i*x xt - j*x 0 i*x n)$	THROW $(k^*x n - k^*x i^*x n)$			
EXCEPTION EXT word set (2 words):	ABORT $(i*x)(R: j*x)$	ABORT" Compilation: ("ccc <quote>")</quote>	Run-time: $(i^*x x 1 - i^*x) (R: j^*x - j^*x)$		
,		· · · · · · · · · · · · · · · · · · ·			
FACILITY word set (3 words):	$\mathbf{AT}\mathbf{-XY} \qquad (u1\ u2\\)$	KEY? $(flag)$	<u>PAGE ()</u>		
FACILITY EXT word set (6 words):	EKEY $(-u)$	EKEY>CHAR $(u u false char true)$	EKEY? (flag)		
•					

MS (*u* --)

(-- flag)

EMIT?

TIME &DATE (--+n1+n2+n3+n4+n5+n6)

Standard Forth (ANSI INCITS 215-1994) Reference

(**<u>RED</u>** = no defined interpretation semantics)

FILE word set (21 words):

("ccc <paren>")</paren>	BIN (fam1 fam2)	CLOSE-FILE (fileid ior)	CREATE-FILE (c-addr u fam fileid ior)
DELETE-FILE (c-addr u ior)	FILE-POSITION (fileid ud ior)	FILE-SIZE (fileid ud ior)	INCLUDE-FILE $(i*x fileid j*x)$
INCLUDED $(i*x c-addr u j*x)$	OPEN-FILE (c-addr u fam fileid ior)	R/O (fam)	$\mathbf{R/W} \qquad (fam)$
READ-FILE (c-addr u1 fileid u2 ior)	READ-LINE (c-addr u1 fileid u2 flag i	i <u>or)</u>	REPOSITION-FILE (ud fileid ior)
RESIZE-FILE (ud fileid ior)	S" Interpretation: ("ccc <quote>" -</quote>	c-addr u) Compilation: ("ccc <quote>") F</quote>	Run-time: (c-addr u)
SOURCE-ID (0 -1 <i>fileid</i>)	<u>w/o (fam)</u>	WRITE-FILE (c-addr u fileid ior)	WRITE-LINE (c-addr u fileid ior)

RENAME-FILE (c-addrl ul c-addr2 u2 -- ior) (F: before -- after)

FLOATING word set (31 words):

>FLOAT (<i>c-addr u true</i> <i>false</i>) (F: <i>r</i>)	D>F (<i>d</i>) (F: <i>r</i>)	$\mathbf{F!} \qquad (f\text{-}addr) (F: r)$	F * (F: r1 r2 r3)
F+ (F: r1 r2 r3)	F - (F: r1 r2 r3)	F / (F: r1 r2 r3)	F0< (flag) (F: r)
F0= $(flag)(F:r)$	F < (flag) (F: r1 r2)	F>D $(-d)(F:r-)$	F @ $(f-addr)(F:r)$
FALIGN ()	FALIGNED (addr f-addr)	FCONSTANT (" <spaces>name") (F: r)</spaces>) <i>name</i> Execution: () (F: r)
FDEPTH $(+n)$	FDROP (F: r)	FDUP $(F: r r r)$	
FLITERAL Compilation: (F: r) Run-tin	ne: (F: r)	FLOAT+ (f-addr1 f-addr2)	FLOATS $(n1 - n2)$
FLOOR (F: r1 r2)	FMAX $(F: r1 \ r2 r3)$	FMIN $(F: r1 \ r2 - r3)$	FNEGATE (F: r1 r2)
FOVER (F: r1 r2 r1 r2 r1)	FROT (F: r1 r2 r3 r2 r3 r1)	FROUND (F: r1 r2)	FSWAP (F: r1 r2 r2 r1)
FVARIABLE (" <spaces>name") name Ex</spaces>	xecution: (f-addr)	REPRESENT (c-addr u n flag1 flag2) (F:	<u>r)</u>

FLOATING EXT word set (41 words):

DF! $(df-addr) (F: r)$	DF @ (<i>df-addr</i>) (F: <i>r</i>)	<u>DFALIGN ()</u>	DFALIGNED (addr df-addr)
<pre>DFLOAT+ (df-addr1 df-addr2)</pre>	<u>DFLOATS (n1 n2)</u>	F** (F: r1 r2 r3)	F. $()(F:r)$
FABS (F: r1 r2)	FACOS (F: r1 r2)	FACOSH (F: r1 r2)	FALOG (F: r1 r2)
FASIN $(F: rl r2)$	FASINH $(F: r1 r2)$	FATAN $(F: r1 r2)$	FATAN2 (F: r1 r2 r3)
FATANH (F: r1 r2)	FCOS $(F: r1 - r2)$	FCOSH $(F: r1 r2)$	FE. $()(F:r)$
FEXP $(F: r1 - r2)$	FEXPM1 (F: r1 r2)	FLN $(F: r1 r2)$	FLNP1 (F: $r1 - r2$)
FLOG $(F: rl r2)$	FS. $()(F:r)$	FSIN $(F: r1 r2)$	FSINCOS (F: r1 r2 r3)
FSINH (F: r1 r2)	FSQRT $(F: r1 - r2)$	FTAN (F: $r1 - r2$)	FTANH (F: r1 r2)
F~ (flag) (F: r1 r2 r3)	PRECISION (u)	SET-PRECISION (u)	SF! $(sf-addr)(F:r)$
SF @ $(sf$ -add r $)$ $(F: r)$	SFALIGN ()	SFALIGNED (addr sf-addr)	SFLOAT+ (sf-addr1 sf-addr2)
SFLOATS (n1 n2)			

LOCAL word set (2 words): (LOCAL) (c-addr u --) local Execution: (-- x)

TO Interpretation: (x "< spaces > name" --) Compilation: ("< spaces > name" --) Run-time: (x --)

LOCAL EXT word set (1 word): Locals | Compilation: ("<spaces>name1" "<spaces>name2" ... "<spaces>namen" "|" --) Run-time: (xn ... x2 x1 --)

MEMORY word set (3 words): ALLOCATE $(u - a - addr \ ior)$ FREE (a - addr - - ior) RESIZE $(a - addr \ 1 \ u - - a - addr \ 2 \ ior)$

Standard Forth (ANSI INCITS 215-1994) Reference

(**<u>RED</u>** = no defined interpretation semantics)

TOOLS word set (5 words):

WORDS (--)

TOOLS EXT word set (13 words):

Compilation: (C: colon-sys --) Run-time: (--) (R: nest-sys --) name Execution: (i*x -- j*x)

AHEAD Compilation: (C: -- orig) Run-time: (--)

ASSEMBLER (--)

BYE (--)

CODE ("<spaces>name" --) name Execution: (i*x -- j*x)

CS-PICK Execution: (C: destu ... orig0|dest0 -- destu ... orig0|dest0 destu) (S: u --)

[ELSE] Execution: ("<spaces>name ... " --)

[THEN] Execution: (--)

SEARCH word set (9 words):

DEFINITIONS (--) **FIND** $(c-addr - c-addr 0 \mid xt \mid xt - 1)$ **FORTH-WORDLIST** (--wid)

GET-CURRENT (-- wid) GET-ORDER (-- widn ... widl n) SEARCH-WORDLIST (c-addr u wid -- $0 \mid xt \mid xt - 1$)

SET-CURRENT (wid--) SET-ORDER (widn ... widl n --) WORDLIST (-- wid)

SEARCH EXT word set (5 words):

ALSO (--) FORTH (--) ONLY (--) ORDER (--)

PREVIOUS (--)

STRING word set (8 words):

-TRAILING (c-addr ul -- c-addr u2) /STRING (c-addrl ul n -- c-addr2 u2) BLANK (c-addr u --) CMOVE (c-addrl c-addr2 u --)

CMOVE> (c-addr1 c-addr2 u --) COMPARE (c-addr1 u1 c-addr2 u2 -- n) SEARCH (c-addr1 u1 c-addr2 u2 -- c-addr3 u3 flag)

SLITERAL Compilation: (c-addr1 u --) Run-time: (-- c-addr2 u)

Table 3.1 – Data types			Symbol	Symbol Data type Size on stack		Table 2.1 – Parsed text abbreviations	
Symbol	Data type	Size on stack	$d ud^2$	double-cell number	2 cells	Abbreviation	Description
flag	flag	1 cell	xd	unspecified cell pair	2 cells	<char></char>	the delimiting character marking the end of the string being
true	true flag	1 cell	colon-sys	definition compilation	implementation dependent		parsed
false	false flag	1 cell	do-sys	do-loop structures	implementation dependent	<chars></chars>	zero or more consecutive occurrences of the character char
char	character	1 cell	case-sys	CASE structures	implementation dependent	<space></space>	a delimiting space character
n	signed number	1 cell	of-sys	OF structures	implementation dependent	<spaces></spaces>	zero or more consecutive occurrences of the character space
+n	non-negative number	1 cell	orig	control-flow origins	implementation dependent	<quote></quote>	a delimiting double quote
u	unsigned number	1 cell	dest	control-flow destinations	implementation dependent	<pre><paren></paren></pre>	a delimiting right parenthesis
$n u^1$	number	1 cell	loop-sys	loop-control parameters	implementation dependent	< <i>eol></i>	an implied delimiter marking the end of a line
x	unspecified cell	1 cell	nest-sys	definition calls	implementation dependent	ccc	a parsed sequence of arbitrary characters, excluding the delimiter
xt	execution token	1 cell	i*x, j*x, k*	x ³ any data type	0 or more cells		character
addr	address	1 cell				name	a token delimited by space, equivalent to ccc < space > or
a-addr	aligned address	1 cell	1 May be e	ither a signed number or an ur	signed number depending on context.		ccc < eol >
c-addr	character-aligned address	1 cell	2 May be e	ither a double-cell signed num	ber or a double-cell unsigned number		
d	double-cell signed number		depending on context.				
+d	double-cell non-negative nu	umber 2 cells	3 May be a	n undetermined number of sta	ck entries of unspecified type. For		
ud	double-cell unsigned numb	er 2 cells	examples	of use, see 6.1.1370 EXECUT	E, 6.1.2050 QUIT.		