BRK 00

break

`--`

ends evaluation of current vector

INC2kr 01

increment

`a -- a+1`

increment value by 1

POP2kr 02

pop

`a --`

discard value

NIP2kr 03

nip

`a b -- b`

discard second value in stack

SWP2kr 04

swap

`a b -- b a`

swap values at top of stack

ROT2kr 05

rotate

`a b c -- b c a`

bring third value to the top of the stack, pushing others down

DUP2kr 06

duplicate

`a -- a a`

duplicate value

OVR2kr 07

over

`a b -- a b a`

duplicate second value in stack

EQU2kr 08

equal

`a b -- bool8`

push 01 if a and b are equal, otherwise 00

NEQ2kr 09

not equal

`a b -- bool8`

push 01 if a and b are not equal, otherwise 00

GTH2kr 0a

greater than

`a b -- bool8`

push 01 if a is greater than b, otherwise 00

LTH2kr 0b

less than

`a b -- bool8`

push 01 if a is less than b, otherwise 00

JMP2kr 0c

jump

`addr --`

adjust PC by the signed value addr8

short mode: set PC to addr16

2

JCN2kr 0d

jump conditional

`cond8 addr --`

if cond is non-zero, adjust PC by ±addr8

short mode: set PC to addr16

2

JSR2kr 0e

jump stash return

`addr --`

stash PC16 to return stack, adjust PC by ±addr8

short mode: set PC to addr16

2

return mode: to working stack

r

STH2kr 0f

stash

`a --`

stash value to return stack

return mode: instead stash to the working stack

r

LDZ2kr 10

load zero page

`addr8 -- a`

push value from 8-bit absolute address

STZ2kr 11

store zero page

`a addr8 --`

store value to 8-bit absolute address

LDR2kr 12

load relative

`addr8 -- a`

push value from 8-bit signed address relative to PC

STR2kr 13

store relative

`a addr8 --`

store value to 8-bit signed address relative to PC

LDA2kr 14

load absolute

`addr16 -- a`

push value from 16-bit absolute address

STA2kr 15

store absolute

`a addr16 --`

store value to 16-bit absolute address

DEI2kr 16

device input

`device8 -- a`

push value read from 8-bit device address

DEO2kr 17

device output

`a device8 --`

write value to 8-bit device address

ADD2kr 18

add

`a b -- a+b`

push the sum of the top two values on the stack

SUB2kr 19

subtract

`a b -- a-b`

push the difference between the values a and b

MUL2kr 1a

multiply

`a b -- a*b`

push the product of the top two values on the stack

DIV2kr 1b

divide

`a b -- a/b`

push the quotient of value a over b, rounding down

AND2kr 1c

bitwise and

`a b -- a&b`

push the bitwise AND of the top two values on the stack

ORA2kr 1d

bitwise or

`a b -- a|b`

push the bitwise OR of the top two values on the stack

EOR2kr 1e

exclusive or

`a b -- a^b`

push the bitwise XOR of the top two values on the stack

SFT2kr 1f

shift

`a shift8 -- c`

shift a right by the low nibble of the shift byte, then left by the high nibble, giving c

JCI 20

jump conditional immediate

`cond8 --`

if cond is non-zero, jump to the relative address16 following this instruction

JMI 40

jump immediate

`--`

jump to the relative address16 following this instruction

JSI 60

jump stash return immediate

`--`

push (PC+2)16, then jump to the relative address16 following this instruction

LIT2r 80

literal

`-- a`

push value following this instruction

2 +20

short mode

all values read and written are 2-byte shorts, unless marked explicitly like cond8 or addr16

r +40

return mode

values are read from and written to the return stack unless otherwise specified

k +80

keep mode

values read from the stack are not removed

hint: in "--" notation, the end result is both sides of the "--" marker

Padding

`|nn` absolute

`$nn` relative

Labels

`@x` label "x", parent

`&y` label "x/y", child

Addressing

`,label` LIT relative8

`_label` raw relative8

`.label` LIT absolute8

`-label` raw absolute8

`;label` LIT2 absolute16

`=label` raw absolute16

`!label` JMI relative8

`?label` JCI relative8

combine with & to refer to a child label of the last parent, e.g. ?&child means JCI to the label "parent/child"

Hex values

`#nn` LIT hex value8

`nn` raw hex value8

`#nnnn` LIT2 hex value8

`nnnn` raw hex value16

ASCII

`"word` raw ASCII bytes

Preprocessor

`%NAME {...}` macro-define

`~file` include

uxntal cheat-sheet