```
#           final "next" pointer of each individual branch points; each
#           branch starts with the operand node of a BRANCH node.
#
BRANCH      node    Match this alternative, or the next...

# BACK      Normal "next" pointers all implicitly point forward; BACK
#           exists to make loop structures possible.
# not used
BACK        no      Match "", "next" ptr points backward.

# Literals
EXACT       sv      Match this string (preceded by length).
EXACTF      sv      Match this string, folded (prec. by length).
EXACTFL     sv      Match this string, folded in locale (w/len).

# Do nothing
NOTHING     no      Match empty string.
# A variant of above which delimits a group, thus stops optimizations
TAIL        no      Match empty string. Can jump here from outside.

# STAR,PLUS '?', and complex '*' and '+', are implemented as circular
#           BRANCH structures using BACK.  Simple cases (one character
#           per match) are implemented with STAR and PLUS for speed
#           and to minimize recursive plunges.
#
STAR        node    Match this (simple) thing 0 or more times.
PLUS        node    Match this (simple) thing 1 or more times.

CURLY       sv 2    Match this simple thing {n,m} times.
CURLYN      no 2    Match next-after-this simple thing
#                   {n,m} times, set parens.
CURLYM      no 2    Match this medium-complex thing {n,m} times.
CURLYX      sv 2    Match this complex thing {n,m} times.

# This terminator creates a loop structure for CURLYX
WHILEM      no      Do curly processing and see if rest matches.

# OPEN,CLOSE,GROUPP ...are numbered at compile time.
OPEN        num 1   Mark this point in input as start of #n.
CLOSE       num 1   Analogous to OPEN.

REF         num 1   Match some already matched string
REFF        num 1   Match already matched string, folded
REFFL       num 1   Match already matched string, folded in loc.

# grouping assertions
IFMATCH     off 1 2 Succeeds if the following matches.
UNLESSM     off 1 2 Fails if the following matches.
SUSPEND     off 1 1 "Independent" sub-regex.
IFTHEN      off 1 1 Switch, should be preceded by switcher .
GROUPP      num 1   Whether the group matched.

# Support for long regex
LONGJMP     off 1 1 Jump far away.
BRANCHJ     off 1 1 BRANCH with long offset.
```