```
; a 128-line compiler for L

   ; tkval          value of token num
   ; last           the end of last fun
   ; numlocal       no. of locals of current fun
   ; freecell       pointer to free sym
   ; CH             current input char
   ; TK             attribute of current token
   ; CP             code pointer
   ; CS             code segment, array 1000
   ; sym            symbol table, array 1000
   ; lvrec          record of locals, array 20

 1 to isNum c { } (c >= 48) & (c <= 57) ; 0..9
 2 to isSpace c { } (c < 33)          ; tab space nl
 3
 4 to readc { }       ; read one char
 5   CH = sys 3
 6   CH
 7
 8 ; a cell has 4 fields, they are: char,right,next,atr
 9 to newcell c { k }
10   k = freecell
11   freecell = freecell + 4
12   sym[k] = c
13   k
14
15 to search i c { }   ;  if sym[i]=0 insert c at i
16   if sym[i] == 0 { sym[i] = newcell c }
17   sym[i]
18
19 ; return index to symtab[], 1 if numeric
20 to lex { i }
21   while readc
22     if CH == 59
23       while CH != 10 { readc }  ; skip comment
24     if ! isSpace CH { break }   ; skip blank
25   if isNum CH
26     tkval = CH - 48
27     while ! isSpace readc
28       tkval = tkval*10 + CH - 48
29       1 break                ; it is numeric
30   i = 0
31   while ! isSpace CH         ; check space
32     i = search i+2 CH          ; next of i
33       while sym[i] != CH { i = search i+1 CH } ;
match right
34     CH = readc               ; read next
35   i
36
37 to outc op arg { }
38   CS[CP] = (arg << 8) | op
39   CP = CP + 1
40
41 to patch a1 a2 { op }          ; relative address
42   op = CS[a1] & 255
43   CS[a1] = ((a2-a1) << 8) | op
44
45 to rename d { } numlocal – d + 1 ; rename 1..n to
n..1
46 to typeof i { } (sym[i+3] >> 20) & 15 ; bits 23..20
47 to getref i { } sym[i+3] & 1048575    ; last 20 bits
48 to setattr i ty ref { } sym[i+3] = (ty << 20) | ref
49
50 to parse { i j a b }
51   i = lex
52   TK = sym[i+3]              ; get attr
53   if i == 1  outc 31 tkval      ; lit tkval
54   else if TK == 0
55     numlocal = numlocal + 1
56     setattr i 3 numlocal        ; declare local
57     lvrec[numlocal] = i
58   else if TK < 24  outc TK 0    ; op no-arg
59   else if TK < 50              ; op arg
60     j = lex                    ; get arg (numeric)
61     outc TK tkval
62   else if TK < 54  break    ; stop { } else
63   else if TK == 56             ;   ->
64     j = lex                    ;  name
65     if (typeof j) == 2
66       outc 27 getref j         ; gvar, st ref
67     else
68       outc 25 rename getref j   ; lvar, put ref
69   else if TK == 53           ;   to
70     last = CP                ; record fun
71     j = lex                  ; fname
72     setattr j 1 CP
73     parse                    ; get pv until {
74     a = numlocal             ; set arity
75     parse                     ; get lv until }
76     outc 38 numlocal-a     ; fun lv
77   else if TK == 54             ;   end
78     outc 20 numlocal         ; ret n
79     i = 1
80     while i <= numlocal
81       setattr lvrec[i] 0 0
82       i = i + 1
83     numlocal = 0             ; clear locals
84   else if TK == 57             ;  while
85     a = CP
86     parse                    ; cond {
87     b = CP
88     outc 30 0                ; jf 0
89     parse                    ; body }
90     outc 28 a-CP             ; jmp a
91     patch b CP
92   else if TK == 55           ;  if
93     parse                     ; cond {
94     a = CP
95     outc 30 0                 ; jf 0
96     parse
97     if TK == 52               ;   else
98       patch a CP+1
99       a = CP
100      outc 28 0                ;   jmp 0
101      parse
102    patch a CP
103  else if TK == 60            ;   var
104    j = lex                     ; global name
105    setattr j 2 (array 1)      ; gvar, alloc DS
106  else                        ; TK > 100, names
107    j = typeof i
108    a = getref i
109    if j == 1  outc 32 a              ; fun, call ref
110    if j == 2  outc 26 a              ; gvar, ld ref
111    if j == 3  outc 24 rename a    ; lvar, get ref
112  parse
113
114 to main { i k }
115   freecell = 4
116   CP = 1
117   CS = array 1000
118   sym = array 1000
119   lvrec = array 20
```

```
120  i = lex            ; init symtab
121  while i > 1        ; until 0 (end with num)
122    k = lex          ; get attr
123    sym[i+3] = tkval
124    i = lex          ; get sym
125  outc 32 0          ; call to main
126  outc 23 0          ; end
127  parse
128  patch 1 last+1
```

For a full detail of this compiler, please refer to
Chongstitvatana, P., "Self-generating systems: how to a 10,000,000_2 line compiler assembles itself", invited paper, National Computer Science and Engineering Conf., Bangkok, Thailand, October 27-28, 2005.