Scope compatibility for symbolic files

TS symbolic files are written in 7-bit, 8 character/word symbolic code, left justified and with multiple blank encoding. Scope programs expect to see a totally different format: 6-bit, 10 character/word card images. Therefore, the Scope interface must provide the necessary conversion. This turns out to cost about 9 µs/character when implemented by the following program:

\$ X6 = \text{word being assembled} \\
\$ B6 = \text{shift count for assembly} \\
\$ X5 = \text{word being disassembled} \\
\$ B5 = \text{count for disassembly} \\
\$ B7 = \text{word address for store} \\
\$ E4 = \text{word address for load} \\
\$ X4 = 1778 \\
\$ B3 = 2nd of input buffer \\
\$ B7 = 2nd of output buffer \\
\$ E1 = 1 \\
\$ X1 = \text{temp} \$
*get new input word

5 L5  S85  8
3 5  S54  B4+B1
5 5  SA5  B4

* main loop for every character, unpick

5 L3  LX5  8
2 5  RX1  X5 * X4
17 * convert
17  SA1  X1 + CONVTAB
5 5  NG  X1, L6
2 * pack
6  LX1  B6 X1
5 5  RX6  X6+X1
5 5  EQ  B6, O, L1
5 5  SB6  B6-6
5 5  to next input char
5 5  L3  S85  B5-1
5 5  NE  B5, O, L2

* need new input word

5 5  NG  X5, L4

13  NE  84, S2, 65

* end of input buffer

Negative move special (ignore or multiple blank)
4. Store output variables

10 L I  S46 B7
5    S B6  60-6
5    S X6  0
5    S B7  B7+A0
12    L E  B7, B8, L3

\[ T_{\text{min}} = 8.1 + \frac{1}{10} \cdot 4.0 + \frac{1}{8} \cdot 3.4 \]

\[ = 8.9 \text{ ms on } 1 \text{ second/12K full card} \]

Probably a similar conversion would be required for output, unless the output is destined solely for printing and we want to provide a special

m

\text{m}

at a w

for this purpose.

Of course, nearly all of this cost can be eliminated by embedding the

conversion into the program reading the input, especially if it is hard
e

g

t

enough to unpick each line into a label/word buffer.