Note on Lenier's 1953 Whirlwind program.

Knuth 12 Aug 76 - 14 Aug 76

Whirlwind upgrade (as detailed from their program, not having a manual available at the time) 16-bit words; 8 bits to thequad
ac = ac + (m).
set ac to result if 20 or more; else 93.
[16:17 set it; 0-0 don't-care]
ac = ac - (m).
toggle ac.
set m = 29.
div decide [used for interpreter call only]
er ac = ac (m)
wr multiply [for interpreter call only]
div decided integer constant 1 or -1 (ap)
rd read a character in Flaconet code into ac. [6-bit code]
rc normal ac and digit number of shifts in (m) [I guess; end only once, 2x1] uninform, something about 1/10?
sl shift left
sp jump and set I register to address of fillbyte instruction
sr shift right into extension of accumulator
ac = ac - (m).
toggle off result <= 0
ra add address (m) to register,
rd address (m) to address gc
rs (m) ac = ac

Assemble language. 4a3 means location of 4a3 plus 4; 2r means location of current ending plus 2; octal constants 0, 7, 4, 10.
location 0 contains 0.

There is a 128-entry switching table (unspecified position or filled with miscellaneous constants and subscripts of code),


decided between lower case and upper case. The codes are:

\[ \begin{array}{c|c}
0.9 & 9 \\
0.1 & 1.9 \\
0.1 & 1.19, 29 & \end{array} \]

e = 0 or period [20], a switch initially set 0
\( \Rightarrow \) common [24] or next 9c or something, a switch initially set to common
\( \Rightarrow \) upper [24] \( \Rightarrow \) lower [24]
S = use :: [20] \( \Rightarrow \) C = use :: [20] \( \Rightarrow \) \( \Rightarrow \) expression [100]9.
P = print, go to part [29] \( \Rightarrow \) a switch initial set to print a.

The code uses switched indentation, I'll use 'bt3' flags all initially false.

- flags: upper case?
- writing: processing a constant? if so, 2 = number :: [24] or constant [20]6
- flag: constant is not a label
- flag: decimal point occurred?
- flag: implied multiplication in effect rather than numeric label; actually, flag = flag, so drop it.
- flag: processing a negative exponent
- flag: exponent should be negative (or positive if in the denominator)
Start: 

```
1a. begin: for x in range(0, 10):
...```
Now, parenthese:

```c
0411 let x = close(x);
11
411
0411 open;
411
stack[k] = alloc;
711
if op < oid then go to next;
1111
stack[k+1] = stack[k] + x;
1111
reset x, go to next;
0411
if stack[k] < x then reset x, go to next;
289
else if flag < time; op < "av"; op < "mv";
9112
compile (x, stock[k]);
19112
stock[k] = stock[k+1]; compile ("sp", stock[k]);
28112
add x, stock[k];
32112
compile (op, stock[k]); k < k-1, go to next;
```

Interpretation: stack[k] is location to hold contents of this parenthese pair,
stock[k] is location of instruction "sp" after evaluating the end of this parenthese pair, it will either
jump to an instruction that uses the contents of this parenthese after evaluation (arc k < op) or wait for
stock[k] to be evaluated (arc k > op).

Example: a = ((b(c)de))((e)(f)),

```
```

leads to: S-expression

```
```

```
```

which is equal to:

```
```
Finally, control flow:

```
print A:  
    output ["",]:  
        print 1:  
            
        print 2:  
            
        print 3:  
            
        print 4:  
            
        print 5:  
            
        print 6:  
            
        print 7:  
            
        print 8:  
            
        print 9:  
            
        print 10:  
            
        print 11:  
            
        print 12:  
            
        print 13:  
            
        print 14:  
            
        print 15:  
            
        print 16:  
            
        print 17:  
            
        print 18:  
            
        print 19:  
            
        print 20:  
            
        print 21:  
            
        print 22:  
            
        print 23:  
            
        print 24:  
            
        print 25:  
            
        print 26:  
            
        print 27:  
            
        print 28:  
            
        print 29:  
            
```