FLAP

Forth & LISP on the PDP-1

Angelo Papenhoff
aap

PiDP-1

replica of PDP-1
computer from 1960

18 bit, 4kw (extension: 64kw) memory

needed a fun hacking project for the machine!

Forth

imperative

von-Neumann-like

stack based

extensible

virtual machine

Lisp

functional

λ-calculus-like

list based

extensible

function evaluator

Meta-circular Lisp (1960)

```
apply[fn;x;a] =
    [atom[fn] → [eq[fn;CAR] → caar[x];
                 eq[fn;CDR] → cdar[x];
                 eq[fn;CONS] → cons[car[x];cadr[x]];
                 eq[fn;ATOM] → atom[car[x]];
                 eq[fn;EQ] → eq[car[x];cadr[x]];
                 T → apply[eval[fn;a];x;a]];
     eq[car[fn];LAMBDA] → eval[caddr[fn];pairlis[cadr[fn];x;a]];
     eq[car[fn];LABEL] → apply[caddr[fn];x;cons[cons[cadr[fn];
                                                caddr[fn]];a]]]
eval[e;a] = [atom[e] → cdr[assoc[e;a]];
     atom[car[e]] →
              [eq[car[e],QUOTE] → cadr[e];
              eq[car[e];COND] → evcon[cdr[e];a];
              T → apply[car[e];evlis[cdr[e];a];a]];
     T → apply[car[e];evlis[cdr[e];a];a]]
```

pairlis and assoc have been previously defined.

```
evcon[c;a] = [eval[caar[c];a] → eval[cadar[c];a];
              T → evcon[cdr[c];a]]
```

and

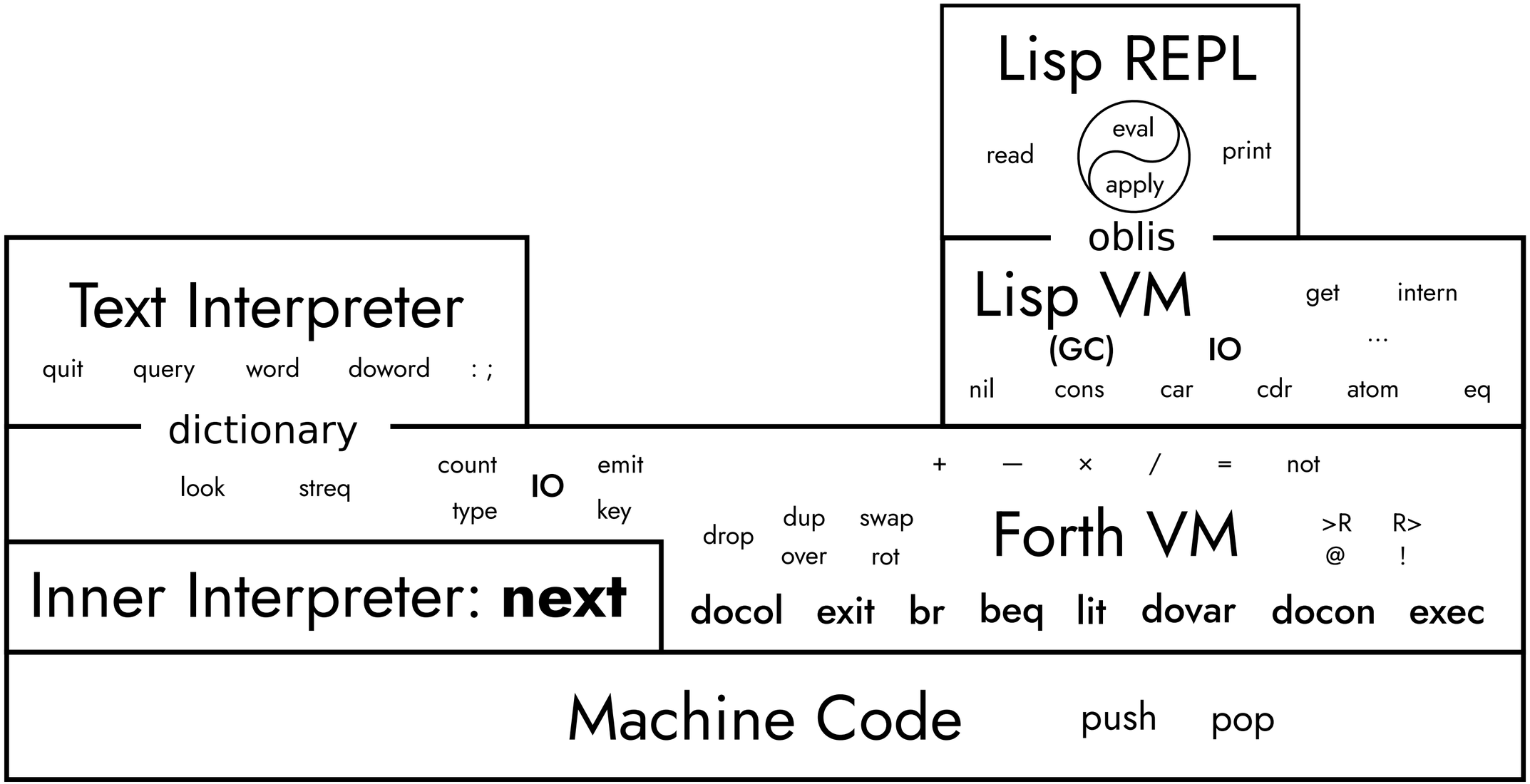
```
evlis[m;a] = [null[m] → NIL;
              T → cons[eval[car[m];a];evlis[cdr[m];a]]]
```

Forth VM

```
next,   lio i ip
        idx ip
        dio t
        lac i t
        dac tt
        jmp i tt

docol,  lio ip
        jsp rpush
        idx t
        dac ip
        jmp next

exit,   .+1
        jsp rpop
        dio ip
        jmp next
```


Lisp REPL
read
eval
apply
print
oblis
Lisp VM
get
intern
(GC)
IO
...
nil
cons
car
cdr
atom
eq
Text Interpreter
quit
query
word
doword
: ;
dictionary
look
streq
count
type
IO
emit
key
+
—
×
/
=
not
drop
dup
over
swap
rot
Forth VM
>R
R>
@
!
Inner Interpreter: next
docol
exit
br
beq
lit
dovar
docon
exec
Machine Code
push
pop

hlt

https://obsolescence.dev/pdp1.html

https://github.com/obsolescence/pidp1/tree/main/tapes/flap