This page is intentionally left blank.

Kazimir Majorinc

PDP-1, BBN I PDP-36 LISP

Povijest Lispa 36.

Razmjena vještina
Hacklab u mami
19. listopada 2013.

PDP-1 LISP

Peter L. Deutsch je tijekom 1960-64 implementirao Lisp 1.5 u „TX-0“ i „PDP-1“ laboratorijima (čini se da su laboratoriji dobijali bar neslužbena imena po računalima) MIT, jer su ta računala omogućavala interaktivni pristup.

Po **Deutschu**, u retrospektivi, najzanimljiviji dio „filozofije dizajna“ je da se, koristeći interpreter, neke operacije koje su definirane kao primitivne mogu definirati unutar samog jezika.

Primjer: umjesto CAAR može se koristiti (LAMBDA(X)(CAR(CAR X))). U vrijeme u koje je **Deutsch** to napisao (prije 2006, ali možda ne puno prije), **Deutsch** se više nije sjećao je li u to vrijeme to bilo opće prihvaćeno znanje (je, još od **McCarthyja**!)

Vidi se da je PDP-1 Lisp dizajniran za interpretiranje ali razlike su vrlo male.

PDP Lisp je „interaktivan.“ Cjepidlačimo li, interaktivniji nego današnji Lispovi: interpreter nije čekao na „return“ nego je prepoznao kraj S-izraza, odmah ga izračunao i ispisao dobivenu vrijednost.

atom

car

cdr

cond

(cons x y…) „returns the concatenation of its argument list.“ (???)

eq

(eval x) - eval sa jednim argumentom

(gensym) - g0001, g0002, ….

(greaterp a b)

go - vidi primjer kod prog

list

(loc x) - adresa na kojoj se nalazi podatak x.

(logand x y) -„ ...returns the logically and of the argument list.“

(logor x y) „...returns the logical or of the argument list.“

(minus x) - prima samo jedan argument.

(null x) - „t if the argument is nil or empty, otherwise t.“

(numberp x)

(plus x y...)

(prin1 x) ispisuje atome

(print x) ispisuje S-izraze

(prog (u) …)

```
(prog (x y)
      (cond ((greaterp x y) (return x))
            (t (return y))))
```

```
(prog ()
      ten (go ten))
```

quote

(read)

return

rplaca

rplacd

sassoc

```
(sassoc (quote y) (quote ((x.1) (y.2))) (quote gensym))
```

setq - za razliku od Lispa 1.5 PDP-1 nema set.

(stop x) - ekvivalentno zaustavlja računalo sa x u akumulatoru

terpri

(times x y...)

(xeq c a i) izvršava mašinsku instrukciju c sa a i i u najvažnijim registrima.

oblist - lista postojećih atomskih simbola

nil

t

expr

subr

fexpr

fsubr

apval

Čini se da Deutsch ignorira FUNARG problem. Nema macroa.

BBN-LISP

Dokument „The BBN LISP system“ u izdanju BOLT BERANEK AND NEWMAN INC., Cambridge, Massachusetts iz veljače 1966. Autori: **Dainel Bobrow**, D. Lucille Darley, Daniel L. Murphy, Cynthia Solomon, **Warren Teitelman**. Liste se spremaju na disk i koristi se keširanje u RAM-u. Dok je RAM imao oko 8k riječi (18 bita), magnetski bubanj je imao 256 k. Usporenje „začuđujuće malo.“ (3 puta, dok je inače pristup disku i bubnju oko 1000 puta sporiji).

oblist[] - funkcija a ne konstanta

not

prog1[x;y], prog2[x;y], progn[x;y; …;z]

set, setq, setn, setnq (vrlo egzotično), setqq

putd[x;y] - isto kao set - ali vrijednost y je definicija funkcija

putdq - isto, samo se niti jedan od argumenata ne izračunava

getd[x] - vrijednost definicije funkcije

fntyp[x] - vraća EXPR, FEXPR, SUBR, FSUBR, NIL, ovisno o tipu funkcije.

eval[x], evala[x;a]

nconc - append, ali uništava originalne liste

nnconc - baš isto kao nconc (u suprotnom ne bi bio moguć trace)

nth[x;n], last[x]

tconc[x;p], lconc[x;p] - p je pointer na zadnji element liste.

remob[x] - uništava sve tragove atoma x iz sistema

disp[x;y] - piše točku na grafičkom terminalu na kordinatama x i y.

displis[l]

mapcar[x;fn]

(mapcar (quote (1 2 3)) (lambda(x)(+ x x))) => (2 4 6).

map - isto kao mapcar ali ne stvara listu

reclaim[]

PDP 36 LISP

Među dokumentima objavljenima na Computer History Museum, Software Preservation Center nalazi se i PDP-36 LISP, interni dokument Electrical Engineering Department (dakle, ne AI) MIT objavljen 20. maja 1966. Autor dokumenta nije naveden i nije poznat. Dokument ima 19 strana + popis Lisp funkcija + cca 40 strana programa u asembleru.

Lisp je opisan kao „izvorni jezik za pisanje algoritama“. Za razliku od ostalih jezika u kojima se programi sastoje od niza instrukcija u Lispu se obično (ali ne uvijek) program sastoji od definicija funkcija. Osnovni objekt u Lispu je *struktura stabla* koja se naziva S-izraz. Sve definicije funkcija, argumenti funkcija, vrijednosti funkcija, programi, instrukcija, varijable i podaci su S-izrazi. Na svakom čvoru S-izraza postoje dvije grane, koje eventualno završavaju u atomu.

S-izrazi se nazivaju konkatenacijom S-izraza.

valp - predikat koji provjerava ima li simbol vrijednost

setq (nema funkcije set)

nlambda - za izračunavanje funkcija koje ne izračunavaju argumente. Takve funkcije se nazivaju fexpr-ovi. Fexprovi imaju samo jedan argument u listi argumenata, ali tom argumentu se pridružuje cijela lista.

((nlambda (x) (plus (car x) (car (cdr x)))) 1 2) => 3

„Functions which do not evaluate their arguments are usually used only on the top level for „utillity" purposes. Other functions do not, as a rule, call them and they are not usually recursive.“

dex

```
 (dex reverse
      (x)
      (prog (y)
            z (cond ((null x (return y)))
              (setq y (cons (car x) y))
              (setq x ( cdr x))
              (go z)
))
```

.

kraj