DAVID MOON
RICHARD M. STALLMAN
DANIEL WEINREB

LISP
CHINE
NUAL

MA
M

Lisp Machine Manual

Sixth Edition, System Version 99

June 1984

Richard Stallman
Daniel Weinreb
David Moon

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract number N00014-80-C-0505.

Preface

The Lisp Machine manual describes both the language and the operating system of the Lisp Machine. The language, a dialect of Lisp called Zetalisp, is completely documented by this manual. The software environment and operating-system-like parts of the system contain many things which are still in a state of flux. This manual confines itself primarily to the stabler parts of the system. It describes how to program, but not for the most part how to operate the machine. The window system is documented separately in the Lisp Machine Window System manual.

Any comments, suggestions, or criticisms will be welcomed. Please send Arpa network mail to BUG-LMMAN@MIT-MC.

Those not on the Arpanet may send U.S. mail to
Richard M. Stallman
Artificial Intelligence Lab
545 Technology Square
Cambridge, Mass. 02139

Portions of this manual were written by Mike McMahon and Alan Bawden. The chapter on the LOOP iteration macro is mostly a reprint of Laboratory for Computer Science memo TM-169, by Glenn Burke. Sarah Smith, Meryl Cohen and Richard Ingria of LMI, and Richard Mlynarik of MIT, helped to correct the manual.

Personal Note from Richard Stallman

The Lisp Machine is a product of the efforts of many people too numerous to list here and of the former unique unbureaucratic, free-wheeling and cooperative environment of the M.I.T. Artificial Intelligence Laboratory. I believe that the commercialization of computer software has harmed the spirit which enabled such systems to be developed. Now I am attempting to build a software-sharing movement to revive that spirit from near oblivion.

Since January 1984 I have been working primarily on the development of GNU, a complete Unix-compatible software system for standard hardware architectures, to be shared freely with everyone just like EMACS. This will enable people to use computers and be good neighbors legally (a good neighbor allows his neighbors to copy any generally useful software he has a copy of). This project has inspired a growing movement of enthusiastic supporters. Just recently the first free portable C compiler compiled itself. If you would like to contribute to GNU, write to me at the address above. Restrain social decay—help get programmers sharing again.

Summary Table of Contents

1. Introduction 1
2. Primitive Object Types 9
3. Evaluation 24
4. Flow of Control. 62
5. Manipulating List Structure 86
6. Symbols 129
7. Numbers 135
8. Arrays 162
9. Generic Sequence Functions 188
10. Characters and Strings 203
11. Functions 223
12. Closures. 250
13. Stack Groups 256
14. Locatives 267
15. Subprimitives 270
16. Areas 296
17. The Compiler 301
18. Macros 320
19. The LOOP Iteration Macro 350
20. Defstruct 372
21. Objects, Message Passing, and Flavors 401
22. The I/O System 448
23. Expression Input and Output 506
24. Naming of Files 545
25. Accessing Files 580
26. The Chaosnet 611
27. Packages 636
28. Maintaining Large Systems 660
29. Processes 682
30. Errors and Debugging 698
31. How to Read Assembly Language 752
32. Querying the User 768
33. Initializations 772
34. Dates and Times. 776
35. Miscellaneous Useful Functions 784
Concept Index. 816
Flavor Index. 820
Operation Index 821
Keyword Index 826
Object Creation Options 833
Meter Index 834
Variable Index. 835
Function Index 840
Condition Name Index 858

Table of Contents

1. Introduction. 1
1.1 General Information 1
1.2 Structure of the Manual. 1
1.3 Notational Conventions and Helpful Notes 3
1.4 Common Lisp Support 7

2. Primitive Object Types 9
2.1 Data Types 9
2.2 Data Type Predicates 11
2.3 Type Specifiers 14
2.3.1 Standard Type Specifiers 14
2.3.2 User-Defined Type Specifiers. 19
2.3.3 Testing Types with Type Specifiers 20
2.3.4 Coercion with Type Specifiers 21
2.3.5 Comparing Type Specifiers. 22

3. Evaluation 24
3.1 Variables 25
3.1.1 Variables and Bindings. 25
3.1.2 Setting Variables. 30
3.1.3 Variable Binding Constructs 31
3.1.4 Defining Global Variables 33
3.1.5 The Global Binding 35
3.2 Generalized Variables. 35
3.2.1 setf 36
3.2.2 locf 38
3.3 Functions. 38
3.3.1 Lambda-List Keywords 43
3.3.2 Local Functions 45
3.4 Some Functions and Special Forms 46
3.5 Declarations 51
3.6 Tail Recursion 55
3.7 Multiple Values. 55
3.8 Evaluation and Function Calling Errors 59

4. Flow of Control 62
4.1 Compound Statements 62
4.2 Conditionals 63
4.2.1 Comparison Predicates. 69
4.3 Iteration 70
4.4 Static Non-Local Exits 75
4.5 Tags and Gotos 78
4.6 Dynamic Non-Local Exits. 80
4.7 Mapping 84

5. Manipulating List Structure 86

5.1 Conses . . . 87
5.2 Lists . . . 89
5.3 Cons Cells as Trees . . . 97
5.4 Cdr-Coding . . . 100
5.5 Tables . . . 102
5.6 Lists as Tables . . . 102
5.7 Lists as Sets. . . . 107
5.8 Association Lists . . . 109
5.9 Stack Lists . . . 112
5.10 Property Lists . . . 113
5.11 Hash Tables. . . . 116
5.11.1 Hash Table Functions . . . 117
5.11.2 Hash Table Operations . . . 120
5.11.3 Hash Tables and the Garbage Collector . . . 121
5.11.4 Hash Primitive . . . 121
5.12 Resources. . . . 123
5.12.1 Defining Resources. . . . 124
5.12.2 Allocating Resource Objects . . . 126
5.12.3 Accessing the Resource Data Structure . . . 127
5.12.4 Fast Pseudo-Resources . . . 128

6. Symbols . . . 129
6.1 The Value Cell . . . 129
6.2 The Function Cell . . . 130
6.3 The Property List. . . . 131
6.4 The Print Name . . . 132
6.5 The Package Cell . . . 132
6.6 Creating Symbols. . . . 133

7. Numbers . . . 135
7.1 Numeric Predicates. . . . 138
7.2 Numeric Comparisons . . . 139
7.3 Arithmetic . . . 141
7.4 Complex Number Functions . . . 146
7.5 Transcendental Functions. . . . 147
7.6 Numeric Type Conversions. . . . 149
7.7 Floating Point Numbers . . . 150
7.8 Logical Operations on Numbers . . . 151
7.9 Byte Manipulation Functions. . . . 155
7.10 Random Numbers. . . . 157
7.11 Information on Numeric Precision. . . . 158
7.12 Arithmetic Ignoring Overflow . . . 160
7.13 24-Bit Arithmetic . . . 160
7.14 Double-Precision Arithmetic . . . 161

8. Arrays . . . 162
8.1 Array Types . . . 163
8.2 Extra Features of Arrays . . . 166

8.2.1 Displaced Arrays 166
8.3 Constructing Arrays 167
8.4 Accessing Array Elements 170
8.5 Getting Information About an Array 173
8.6 Changing the Size of an Array 176
8.7 Arrays Overlaid With Lists 177
8.8 Adding to the End of an Array 178
8.9 Copying an Array 179
8.10 Bit Array Functions 180
8.11 Order of Array Elements 182
8.12 Matrices and Systems of Linear Equations 183
8.13 Planes 185
8.14 Maclisp Array Compatibility 186

9. Generic Sequence Functions 188
9.1 Primitive Sequence Operations 188
9.2 Simple Sequence Operations 189
9.3 Mapping On Sequences 191
9.4 Operating on Selected Elements 193
9.4.1 Removing Elements from Sequences 194
9.4.2 Substitution Functions 196
9.4.3 Searching for Elements 197
9.5 Comparison Functions 199
9.6 Sorting and Merging 200

10. Characters and Strings 203
10.1 Characters 204
10.1.1 Components of a Character 205
10.1.2 Constructing Character Objects 207
10.1.3 The Character Set 208
10.1.4 Classifying Characters 210
10.1.5 Comparing Characters 210
10.1.6 Character Names 211
10.2 Conversion to Upper or Lower Case 212
10.3 Basic String Operations 214
10.4 String Searching 218
10.5 Maclisp-Compatible Functions 221

11. Functions 223
11.1 What Is a Function? 223
11.2 Function Specs 223
11.3 Simple Function Definitions 227
11.4 User Operations on Functions 228
11.5 Kinds of Functions 229
11.5.1 Interpreted Functions 229
11.5.2 Lambda Macros 230
11.5.3 Compiled Functions 232
11.5.4 Other Kinds of Functions 232

11.5.5 Special Forms and Functions 233
11.6 Function-Defining Special Forms 234
11.6.1 Maclisp Lexprs 238
11.7 How Programs Manipulate Function Specs 239
11.8 How Programs Examine Functions 242
11.9 Encapsulations 244
11.9.1 Rename-Within Encapsulations 249

12. Closures 250
12.1 What a Closure Is 250
12.2 Examples of the Use of Closures 252
12.3 Closure-Manipulating Functions 253
12.4 Entities 255

13. Stack Groups 256
13.1 Resuming of Stack Groups 257
13.2 Stack Group States 258
13.3 Stack Group Functions 259
13.4 Analyzing Stack Frames 261
13.5 Input/Output in Stack Groups 264
13.6 An Example of Stack Groups 265

14. Locatives 267
14.1 Cells and Locatives 267
14.2 Functions That Operate on Locatives 268
14.3 Mixing Locatives with Lists 268

15. Subprimitives 270
15.1 Data Types 270
15.2 Forwarding 272
15.3 Pointer Manipulation 274
15.4 Special Memory Referencing 275
15.5 Storage Layout Definitions 278
15.6 Analyzing Structures 279
15.7 Creating Objects 280
15.8 Copying Data 280
15.9 Returning Storage 281
15.10 Locking Subprimitive 282
15.11 CADR I/O Device Subprimitives 282
15.12 Lambda I/O-Device Subprimitives 283
15.13 Function-Calling Subprimitives 283
15.14 Special-Binding Subprimitive 284
15.15 The Paging System 284
15.16 Closure Subprimitives 287
15.17 Distiguishing Processor Types 288
15.18 Microcode Variables 289
15.19 Microcode Meters 292
15.20 Miscellaneous Subprimitives 295

16. Areas 296
16.1 Area Functions and Variables 297
16.2 Interesting Areas. 299
16.3 Errors Pertaining to Areas 300

17. The Compiler 301
17.1 The Basic Operations of the Compiler 301
17.2 How to Invoke the Compiler. 301
17.3 Input to the Compiler 303
17.4 Compile-Time Properties of Symbols. 307
17.5 Using Compiler Warnings 308
17.5.1 Controlling Compiler Warnings. 309
17.5.2 Recording Warnings 311
17.6 Compiler Source-Level Optimizers. 313
17.7 Maclisp Compatibility 314
17.8 Putting Data in QFASL Files. 317
17.9 Analyzing QFASL Files 319

18. Macros. 320
18.1 Introduction to Macros. 320
18.2 Aids for Defining Macros 322
18.2.1 Defmacro. 322
18.2.2 Backquote 325
18.3 Local Macro Definitions 329
18.4 Substitutable Functions 329
18.5 Hints to Macro Writers. 331
18.5.1 Name Conflicts 332
18.5.2 Block-Name Conflicts. 334
18.5.3 Macros Expanding into Many Forms 334
18.5.4 Macros that Surround Code. 336
18.5.5 Multiple and Out-of-Order Evaluation. 337
18.5.6 Nesting Macros. 339
18.5.7 Functions Used During Expansion 341
18.6 Aids for Debugging Macros 342
18.7 Displacing Macro Calls. 342
18.8 Functions to Expand Macros. 343
18.9 Definitions of Macros 344
18.10 Extending setf and locf 345

19. The LOOP Iteration Macro. 350
19.1 Introduction. 350
19.2 Clauses 351
19.2.1 Iteration-Driving Clauses 352
19.2.2 Bindings 354
19.2.3 Entrance and Exit. 355
19.2.4 Side Effects. 356
19.2.5 Values 356
19.2.6 Endtests 358

19.2.7 Aggregated Boolean Tests 358
19.2.8 Conditionalization 359
19.2.9 Miscellaneous Other Clauses 360
19.3 Loop Synonyms. 361
19.4 Destructuring 361
19.5 The Iteration Framework 362
19.6 Iteration Paths. 363
19.6.1 Pre-Defined Paths 365
19.6.1.1 The Interned-Symbols Path 365
19.6.1.2 The Hash-Elements Path. 365
19.6.1.3 Sequence Iteration. 366
19.6.2 Defining Paths 367
19.6.2.1 Path Definition Example. 369

20. Defstruct 372
20.1 Introduction to Structure Macros 372
20.2 How to Use Defstruct 374
20.3 Options to Defstruct. 376
20.4 Using the Constructor and Alterant 385
20.4.1 Constructors 385
20.4.2 By-Position Constructors 386
20.4.3 Alterant Macros 387
20.5 Byte Fields 387
20.6 Grouped Arrays. 389
20.7 Named Structures. 390
20.8 Common Lisp Defstruct. 393
20.9 The si:defstruct-description Structure 394
20.10 Extensions to Defstruct. 396
20.10.1 Example 396
20.10.2 Options to si:defstruct-define-type. 397

21. Objects, Message Passing, and Flavors 401
21.1 Objects 401
21.2 Modularity 402
21.3 Generic Operations 405
21.4 Generic Operations in Lisp 406
21.5 Simple Use of Flavors. 408
21.6 Mixing Flavors 411
21.7 Flavor Functions 414
21.8 Defflavor Options. 424
21.9 Flavor Families. 431
21.10 Vanilla Flavor 432
21.11 Method Combination 433
21.12 Implementation of Flavors. 441
21.12.1 Order of Definition 442
21.12.2 Changing a Flavor. 443
21.13 Useful Editor Commands. 443

21.14 Property List Operations 445
21.15 Printing Flavor Instances Readably 446
21.16 Copying Instances 446

22. The I/O System 448
22.1 Input Functions 449
22.1.1 String Input Functions 449
22.1.2 Character-Level Input Functions 450
22.1.3 Maclisp Compatibility Input Functions 451
22.1.4 Interactive Input with Prompting 453
22.2 Output Functions 456
22.3 I/O Streams 459
22.3.1 Standard Streams 459
22.3.2 Standard Input Stream Operations 461
22.3.3 Standard Output Stream Operations 463
22.3.4 Asking Streams What They Can Do 465
22.3.5 Operations for Interactive Streams 466
22.3.6 Cursor Positioning Stream Operations 467
22.3.7 Operations for Efficient Pretty-Printing 468
22.3.8 Random Access File Operations 468
22.3.9 Buffered Stream Operations 469
22.3.10 Obtaining Streams to Use 470
22.3.11 String I/O Streams 473
22.3.12 Implementing Streams 474
22.3.13 Implementing Streams with Flavors 475
22.3.14 Implementing Streams Without Flavors 479
22.4 Formatted Output 483
22.4.1 The Format Function 483
22.4.2 The Output Subsystem 496
22.5 Rubout Handling 500

23. Expression Input and Output 506
23.1 What the Printer Produces 506
23.2 Options that Control Printing 514
23.3 What The Reader Accepts 516
23.3.1 Sharp-sign Constructs 521
23.4 Expression Output Functions 527
23.4.1 Pretty-Printing Output Functions 528
23.4.2 Non-Stream Printing Functions 529
23.5 Expression Input Functions 530
23.5.1 Non-Stream Parsing Functions 533
23.5.2 Input Error Conditions 534
23.6 The Readtable 535
23.7 Read-Macro Characters 539
23.8 The :read and :print Stream Operations 543

24. Naming of Files 545
24.1 Pathnames 545

24.2 Pathname Components 547
24.2.1 Raw Components and Interchange Components 549
24.2.2 Pathname Component Operations 550
24.2.3 Canonical Types 551
24.3 Defaults and Merging 554
24.4 Pathname Functions 557
24.5 Generic Pathnames 561
24.6 Pathname Operations 563
24.7 Host File Systems Supported 568
24.7.1 ITS 568
24.7.2 TOPS-20 (Twenex), Tenex, and VMS 569
24.7.3 Unix and Multics Pathnames 570
24.7.4 Lisp Machine File Systems 571
24.7.5 Logical Pathnames 572
24.7.6 Editor Buffer Pathnames 575
24.8 Hosts 576
24.8.1 Parsing Hostnames 576
24.8.2 Host Object Operations 578

25. Accessing Files 580
25.1 Opening and Closing File Streams 580
25.2 File Stream Operations 587
25.3 Manipulating Files 588
25.3.1 Loading Files 591
25.4 Pathname Operations That Access Files 592
25.5 File Attribute Lists 594
25.6 Accessing Directories 598
25.7 Errors in Accessing Files 604
25.8 File Servers 607
25.8.1 Errors in Communication with File Servers 609

26. The Chaosnet 611
26.1 Chaosnet Overview 611
26.2 Conns 614
26.3 Opening and Closing Connections 615
26.3.1 User-Side 615
26.3.2 Server-Side 616
26.4 Stream Input and Output 617
26.5 Packet Input and Output 619
26.6 Connection Interrupts 622
26.7 Chaosnet Errors 622
26.7.1 Local Problems 623
26.7.2 Problems Involving Other Machines' Actions 623
26.8 Information and Control 624
26.9 Higher-Level Protocols 626
26.9.1 Status 626
26.9.2 Routing Information 627

26.9.3 Telnet and Supdup 627
26.9.4 File Access 628
26.9.5 Mail 628
26.9.6 Send 629
26.9.7 Name. 629
26.9.8 Time 629
26.9.9 Uptime 630
26.9.10 Arpanet Gateway 630
26.9.11 Host Table. 630
26.9.12 Dover 631
26.9.13 Remote Disk. 632
26.9.14 The Eval Server 632
26.10 Using Higher Level Protocols 633

27. Packages 636
27.1 The Current Package. 637
27.2 Package Prefixes 638
27.3 Home Packages of Symbols 639
27.4 Keywords 640
27.5 Inheritance between Name Spaces 641
27.6 Packages and Interning. 644
27.7 Shadowing and Name Conflicts 647
27.8 Styles of Using Packages 648
27.9 Package Naming. 649
27.9.1 Local Nicknames for Packages. 651
27.10 Defining Packages 652
27.11 Operating on All the Symbols in a Package 655
27.12 Packages as Lisp Objects 656
27.13 Common Lisp and Packages 657
27.14 Initialization of the Package System 657
27.15 Initial Packages 658

28. Maintaining Large Systems. 660
28.1 Defining a System 660
28.2 Transformations 663
28.3 Making a System. 666
28.4 Adding New Keywords to make-system 668
28.5 Adding New Options for defsystem. 669
28.6 More Esoteric Transformations. 670
28.7 Common Lisp Modules 671
28.8 The Patch Facility 672
28.8.1 Defining a System. 675
28.8.2 Loading Patches 675
28.8.3 Making Patches 676
28.8.4 Private Patches 678
28.8.5 System Status 679
28.8.6 Patch Files 680

29. Processes 682
29.1 The Scheduler. 683
29.2 Locks 687
29.2.1 Process Queues. 689
29.3 Creating a Process. 689
29.4 Process Generic Operations 691
29.4.1 Process Attributes 691
29.4.2 Run and Arrest Reasons 694
29.4.3 Bashing the Process. 694
29.5 Process Flavors 695
29.6 Other Process Functions. 696

30. Errors and Debugging 698
30.1 Conditions 698
30.2 Handling Conditions 700
30.3 Standard Condition Flavors 705
30.4 Condition Operations 706
30.4.1 Condition Operations for the Debugger. 707
30.5 Signaling Conditions 708
30.5.1 Convenience Functions for Signaling 708
30.5.2 Creating Condition Instances 713
30.5.3 Signaling a Condition Instance 715
30.6 Proceeding 717
30.6.1 Proceeding and the Debugger. 718
30.6.2 How Signalers Provide Proceed Types. 722
30.6.3 Nonlocal Proceed Types 723
30.7 The Debugger. 726
30.7.1 Entering the Debugger 726
30.7.2 How to Use the Debugger 728
30.7.3 Debugger Commands 730
30.7.4 Summary of Commands 734
30.7.5 Deexposed Windows and Background Processes. 736
30.7.6 Debugging after a Warm Boot 737
30.8 Tracing Function Execution 737
30.9 Breakon. 741
30.10 Advising a Function 742
30.10.1 Designing the Advice 744
30.10.2 :around Advice 744
30.10.3 Advising One Function Within Another 745
30.11 Stepping Through an Evaluation 746
30.12 Evalhook. 748
30.13 The MAR 750

31. How to Read Assembly Language 752
31.1 Introduction. 752
31.2 A More Advanced Example 755
31.3 The Rest of the Instructions 758

31.4 Function Entry . . . 761
31.5 Special Class IV Instructions . . . 763

32. Querying the User . . . 768

33. Initializations . . . 772
33.1 System Initialization Lists . . . 772
33.2 Programming Initializations . . . 774

34. Dates and Times . . . 776
34.1 Getting and Setting the Time. . . . 777
34.1.1 Elapsed Time in 60ths of a Second. . . . 777
34.1.2 Elapsed Time in Microseconds . . . 778
34.2 Printing Dates and Times . . . 778
34.3 Reading Dates and Times . . . 780
34.4 Reading and Printing Time Intervals . . . 781
34.5 Time Conversions . . . 782
34.6 Internal Functions . . . 782

35. Miscellaneous Useful Functions . . . 784
35.1 Documentation . . . 784
35.2 Hardcopy . . . 785
35.3 Metering . . . 787
35.4 Poking Around in the Lisp World . . . 790
35.5 Utility Programs . . . 793
35.6 The Lisp Listen Loop . . . 795
35.7 The Garbage Collector. . . . 797
35.8 Logging In. . . . 800
35.9 Dribble Files. . . . 802
35.10 Version Information . . . 803
35.11 Booting and Disk Partitions. . . . 804
35.11.1 Manipulating the Label . . . 805
35.11.2 Updating Software. . . . 807
35.11.3 Saving Personal Software. . . . 808
35.11.4 Copying Bands . . . 809
35.12 Site Options and Host Table . . . 810
35.12.1 Updating Site Information . . . 814
35.12.2 Accessing Site Options. . . . 814
35.12.3 The LMLOCS File. . . . 815

Concept Index . . . 816

Flavor Index . . . 820

Operation Index . . . 821

Keyword Index . . . 826

Object Creation Options . . . 833

Meter Index . . . 834

Variable Index. 835
Function Index 840
Condition Name Index 858