

# Macro processors and their use in implementing software

P.J. Brown

November 1968 (revised April 1971)

Originally submitted for the degree of Doctor of  
Philosophy in the University of Cambridge

Copyright © 1968,1971 P.J. Brown

Permission is granted to copy and/or modify this document for private use only. Machine readable versions must not be placed on public web sites or FTP sites, or otherwise made generally accessible in an electronic form. Instead, please provide a link to the original document on the official ML/I web site (<http://www.ml1.org.uk>).

# Table of Contents

<b>Preface</b> .....	<b>1</b>
<b>Overall Introduction</b> .....	<b>2</b>
Summary .....	2
Acknowledgements .....	2
Originality .....	2
<b>Part I — A Survey of Macro Processors</b> .....	<b>4</b>
<b>1 Introduction</b> .....	<b>5</b>
1.1 What is a Macro Processor? .....	5
1.2 Why “Macro Processor” rather than “Macro Language”? ..	5
1.3 Uses of Macro Processors .....	5
1.4 Organisation of this Discussion .....	6
1.5 Language Extension .....	6
1.6 Language Translation .....	7
1.7 Text Generation .....	7
1.8 Systematic Editing and Searching .....	7
1.9 Introduction to the Design of a Macro Processor .....	8
<b>2 Base Language</b> .....	<b>9</b>
2.1 Syntactic Macros .....	9
2.2 Computation Macros .....	11
2.3 Difficulties in Categorising Compiler-Dependent Macros ...	11
2.4 Advantages of Base Language Dependence .....	12
<b>3 Syntax</b> .....	<b>14</b>
3.1 Syntax of Macro Calls .....	14
3.2 Recognition of Macro Calls .....	14
3.3 Recognition of Start First .....	15
3.4 Macro Identification First .....	16
3.5 End First .....	17
3.6 Separation of Arguments .....	17
3.7 XPOP Syntax .....	18
3.8 LIMP Syntax .....	19
3.9 ML/I Syntax .....	19
3.10 Notation for Formal Parameters .....	20
3.11 Syntax of Macro-Time Statements .....	21
3.12 Comparison of Syntaxes .....	21
3.13 Notation-Independence .....	21
3.14 Context-Dependent Replacement .....	22

3.15	Use of Newline . . . . .	22
3.16	Notational Restrictions . . . . .	23
3.17	Errors and Error Recovery . . . . .	24
<b>4</b>	<b>Text Evaluation . . . . .</b>	<b>25</b>
4.1	Name or Value . . . . .	25
4.2	Multi-Level Calls . . . . .	26
4.3	Conclusions on Multi-Level Calls . . . . .	28
4.4	Output . . . . .	28
4.5	The Scope of Macro Definitions . . . . .	29
4.6	Definitions with Restricted Scope . . . . .	30
4.7	Skips . . . . .	31
<b>5</b>	<b>Macro-Time Facilities . . . . .</b>	<b>32</b>
5.1	Macro-Time Variables . . . . .	32
5.2	Character Variables . . . . .	32
5.3	Integer Variables . . . . .	32
5.4	Boolean Variables . . . . .	32
5.5	Further Facilities . . . . .	33
5.6	Macro-Time Statements . . . . .	33
5.7	Arithmetic and Control Statements . . . . .	34
5.8	Further Statements . . . . .	35
5.9	Macro-Time Dictionaries . . . . .	35
<b>6</b>	<b>Implementation Methods . . . . .</b>	<b>37</b>
6.1	Multi-Pass Macro Processors . . . . .	37
6.2	Pre-Compiling of Macro-Time Statements . . . . .	37
6.3	List Processing Techniques . . . . .	38
6.4	Further Considerations of Speed . . . . .	39
6.5	Further Storage Considerations . . . . .	39
<b>7</b>	<b>Conclusions . . . . .</b>	<b>41</b>
7.1	Conclusions on Language Extension . . . . .	41
7.2	Conclusions on Language Translation . . . . .	43
7.3	Conclusions on Text Generation . . . . .	44
7.4	Conclusions on Editing and Searching . . . . .	45
7.5	Final Conclusions . . . . .	45
	<b>Part II — A Critique of ML/I . . . . .</b>	<b>46</b>
<b>8</b>	<b>A Critique of ML/I . . . . .</b>	<b>47</b>
8.1	Innovations in ML/I . . . . .	47
8.2	Defects of ML/I . . . . .	47
8.3	The Logic of ML/I . . . . .	48
8.4	Uses of ML/I . . . . .	49

<b>Part III — An Exercise in Machine Independence</b>	<b>51</b>
.....	
<b>9 The Use of Macro Processors in Implementing Machine-Independent Software</b>	<b>52</b>
9.1 Introduction	52
9.2 Discussion of Machine-Independence	52
9.3 Simple Criteria for Machine-Independence	52
9.4 The Problem	53
9.5 A Method of Solution	53
9.6 Advantages over Use of High-Level Languages	54
9.7 Advantages over Hand-Coding	55
9.8 Further Mappings	55
<b>10 The Language L</b>	<b>56</b>
10.1 Basic Details of L	56
10.2 The Mapping of L	58
10.3 Features of L That Specially Aid Machine-Independence	59
10.4 Constant-Defining Macros	59
10.5 Statement Prefixes	60
10.6 Example of a Statement Prefix	60
<b>11 Comparison with Other DLIMPs</b>	<b>62</b>
11.1 The Design of a Descriptive Language	62
11.2 Other DLIMPS	62
11.3 Implementing of SNOBOL	62
11.4 Implementing of Meta-Assemblers	63
11.5 DLIMPs Using WISP	63
11.6 Comparison with Other Methods	64
11.7 Advantages of using ML/I for a DLIMP	64
11.8 Flexibility	65
11.9 Power	65
<b>12 Results of Transfers</b>	<b>67</b>
12.1 Inefficiency	67
12.2 Time Taken	67
12.3 Representation of Data Types	68
12.4 Acknowledgements	68
12.5 The PDP-7 Assembly Language L-map	69
12.6 The IIT L-map for Titan	70
12.7 The PL/I L-map	71
12.8 The PLAN L-map for ICT 1900	73
12.9 The IBM System/360 Assembly Language L-map	75
12.10 The EASYCODER L-map for Honeywell 200	75
12.11 Summary of Results	76

<b>Bibliography and Appendices</b> .....	<b>77</b>
<b>Bibliography</b> .....	<b>78</b>
Main List of References .....	78
Cross-References .....	80
<b>Appendix A ML/I User's Manual</b> .....	<b>81</b>
<b>Appendix B L-Map Implementor's Manual</b> ...	<b>82</b>

## Preface

This is the complete dissertation, apart from the two Appendices (see below). The following points should be noted:

- a. The whole dissertation has been converted to Texinfo format. This has necessitated changing the Appendices from “I and II” to “A and B”.
- b. Part I was published in *Annual Review in Automatic Programming*, Volume 6, Pergamon Press, 1968.
- c. Each of the three Parts is self-contained and can be read independently of the other two. A knowledge of ML/I as summarised in the paper “The ML/I Macro Processor” [*Comm. ACM* **10**, 10 (Oct. 1967), 618–623] is assumed in Part II. The above paper described the main principles of ML/I and may be looked upon as a precis of Appendix B.
- d. The contents of Part III are also summarised in a published paper, which is entitled “Using a macro processor to aid software implementation”, [*Computer J.* **12**, 4 (Nov. 1969) 327–331].
- e. There have been two different versions of the Bibliography. These have been combined, and numbered references in the text have been adjusted accordingly.
- f. The Appendices are separate documents. Both of them have been updated at least twice, and the originals are no longer easily available.

# Overall Introduction

## Summary

The main research for this dissertation has consisted of designing and implementing a macro processor, which I have called *ML/I*, and in using it to generate versions of itself for several computers. A description of *ML/I* has been published [*Comm. ACM* **10**, 10 (Oct. 1967), 618–623].

This dissertation consists of three Parts and two large Appendices.

Part I is a survey of existing macro processors, including *ML/I*, with an evaluation of some of their uses, their achievements and their failings. The survey introduces the four main application areas for macro processors, which are considered to be language extension, language translation, text generation, and systematic editing and searching. It then considers the design factors that make macro processors fundamentally different from one another. These design factors are: relationship with base language, syntax, text evaluation, macro-time facilities and implementation methods. The bulk of the survey (Chapter 2, Chapter 3, Chapter 4, Chapter 5 and Chapter 6) is devoted to considering these design factors in turn. The last Chapter, Chapter 7, reviews the four application areas in the light of what has been said in the preceding Chapters.

Part II is a short critique of *ML/I*. It is a general discussion rather than a very detailed one. A complete description of *ML/I* is given as Appendix A.

Part III introduces the idea of a *DLIMP*. This is a means whereby machine-independent software can be implemented by describing it in a special-purpose language and then using a macro processor to map this language into any desired object language, in particular into the assembly language of any desired machine. The implementation of *ML/I* itself by means of a *DLIMP* is described and results for several implementations are presented. Full details of how to implement *ML/I* by this method are given in Appendix B. The object of Part III is to show that a *DLIMP* can be a very good method of software implementation and that *ML/I* is an especially suitable vehicle for performing a *DLIMP*.

## Acknowledgements

I would like to thank Professor M.V. Wilkes, who was my supervisor during the first and third years of my research, and Professor D.W. Barron who was my supervisor during the second year. I would like to thank John Nicholls and other members of the staff of IBM (UK) Laboratories for useful discussions and for arranging the typing of the draft of Part I. I would also like to thank the people who helped with the various implementations of *ML/I*. Full acknowledgements to these people are given in Chapter 12. Finally I would like to thank the Science Research Council for the financial support of my work.

## Originality

No survey of macro processors nor any analysis of the basic principles in the design of macro processors has ever been published before. Part I of this dissertation is intended to fill this

gap. Chapter 3 and Chapter 4, in particular, contain new insights into the design of macro processors.

ML/I, like any new piece of software, contains many facilities that have been in use before. However, its most central characteristic is original. The degree to which ML/I is original and the extent to which it has taken ideas from other macro processors is discussed at more length in Chapter 8.

The idea of a DLIMP as presented in Part III is not new, though no analysis of its virtues as a general method for software implementation has been published previously. The implementing of ML/I by means of a DLIMP is new in the following ways:

- a. It is a much more elaborate operation than the only previously published account of a DLIMP.
- b. New techniques such as statement prefixes and constant-defining macros have been introduced.
- c. The same descriptive language has been mapped into both a high-level language and an assembly language.
- d. Detailed results of a DLIMP have never been published before but the results quoted in this dissertation are thought to set a high standard of efficiency.

I hereby declare that this dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University. I further state that no part of my dissertation has already or is being currently submitted for any such degree, diploma or other qualification.

University Mathematical Laboratory  
Cambridge, England.

March 1968.