Major Microprocessors

A Unified Approach using CALM

J.D. Nicoud
and F. Wagner

North - Holland

MAJOR MICROPROCESSORS

A Unified Approach using CALM

J. D. NICOUD

EPFL Lausanne
Lausanne, Switzerland

and

F. WAGNER

Balzers AG
Balzers, Liechtenstein

1987

NORTH-HOLLAND
AMSTERDAM · NEW YORK · OXFORD · TOKYO

© ELSEVIER SCIENCE PUBLISHERS B.V., 1987

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

ISBN: 0 444 70116 8

Publishers:

ELSEVIER SCIENCE PUBLISHERS B.V.
P.O. Box 1991
1000 BZ Amsterdam
The Netherlands

Sole distributors for the U.S.A. and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY, INC.
52 Vanderbilt Avenue
New York, N.Y. 10017
U.S.A.

Library of Congress Cataloging-in-Publication Data

Nicoud, Jean-Daniel.
Major microprocessors.

Bibliography: p.
Includes index.
1. Microprocessors. 2. Microprocessors--Programming.
3. Assembler language (Computer program language)
4. CALM (Computer program language) I. Wagner, F.,
1941- . II. Title.
QA76.5.N5125 1987 005.26 86-24126
ISBN 0-444-70116-8

PRINTED IN THE NETHERLANDS

MAJOR MICROPROCESSORS
A Unified Approach using CALM

PREFACE

The microprocessor field is one of the fastest moving industrial fields but, if one considers the computer organization principles defined by von Neumann more than forty years ago, its core has not changed very much. The small number of transistors that could be put on an integrated circuit limited the performance of the first microprocessors. But the steady increase of chip transistor density has resulted in new implementations, more or less software compatible with the earlier devices, every three or five years. And although several less powerful microprocessors having weak points or insufficient support have disappeared, a limited number of microprocessors have had really great impacts in the computing world. But even these are superseded by new products, keeping however the same philosophy and partial compatibility.

For years, efforts were made to establish a common assembler notation which could become an internationally acknowledged standard [BIEW79] [DUNC81] [FISCH79] [BALD84]. The Common Assembly Language for Microprocessors CALM was designed at the Ecole Polytechnique of Lausanne (Switzerland) and has been used since 1974 [NIC75] [NIC76] [SCHN81] [NIC83] [STROH86] [ZELT86]. It is applicable to the currently available 8-bit, 16-bit and 32-bit microprocessors, has been standardized by DIN [DIN84] and is under consideration by IEC. Over the last ten years, CALM assemblers for more than 20 microprocessors have been implemented, thousands of students have learned CALM, assemblers running on widely spread systems are available [FAH85], many companies have developed industrial software using CALM, and several megabytes of code have been written. The educational and professional efficiency gained through the use of CALM has been proven many times over.

Assembly language is used in the development of system software and in demanding applications such as those of real time control. The knowledgable programmer must have a good understanding of the basic features of each machine so that notations which are logical and easy to remember are of major interest, especially now when he spends a major part of his time writing in a high-level language such as Fortran, C, Pascal, or Modula.

此为试读，需要完整PDF请访问：www.ertongbook.com

Each manufacturer has its own assembler mnemonics and notations, though there are no essential reasons for doing so. Thus, the programmer when using a different processor must simultaneously understand the architecture and the features of the processor, and learn the new specific notations describing these. Since each processor implements a subset of what an ideal processor should do, a universal notation could be used to describe any of them. Manufacturers restrict their notation to the functions their processors execute; this means that they contain fewer symbols than universal notations; these short-hand notations are simple to use after several weeks of usage, but they are quickly forgotten. CALM notations specify the data type and all elements of the addressing mode; confusion and errors are therefore minimized, especially for the programmer who uses assembly language infrequently, or has to program different processors in the same period of time.

This book covers the principles behind microprocessors and describes the main features of seven of the more widely used devices. It is primarily a book about programming in assembly language. CALM is used to describe the basic assembly language concepts and the machine languages of selected microprocessors. The machine languages of other micros, minis and even large computers could be described using the same notation, which focuses on operand sizes and addressing modes.

Beginners may have difficulties understanding all concepts at first reading. The reader should have some basic understanding of computers and programming, especially in assembly language. The book should help him to get a much better understanding of programming microprocessors, and to convince him of the purpose and usefulness of a standard software implementation of an assembler notation such as CALM.

This book consists of two parts. The first part is introductory, and the second part describes seven selected microprocessors. The four chapters of the first part cover number representations, processor operations and programming. They are not intended to teach the basics of computers but rather to introduce and define the ideas and names used in the following chapters. Programming is detailed in seventeen examples for a hypothetically perfect processor. These seventeen programs are rewritten in the following chapters for each of the selected processors; this illustrates the features of each microprocessor, and demonstrates that a good programming practice uses the methodology of successive refinement; it however does not cover all the techniques used when programming large programs.

The second part of the book consists of seven chapters, each of which describes one processor. The 6809 is presented first, due to its more complete features. The 6809 ended the line of true 8-bit devices, but came too late to get the success it deserved. The 8085 is still very much in use and has some historical importance. With its predecessors, the 8008 and 8080, it greatly influenced the microprocessor world of the '70s. The Z80 and its CMOS versions have been very successful not only among professionals, but also with thousands of microprocessor fans. Two major monolithic microcomputers are presented. The well known 8048 has a particular architecture, but CALM notations are also adequate for this processor. The 6801 is straightforward and easy to understand. Two widely used 16-bit microprocessors, the 8086 and the 68000 complete the book. The 32-bit upward compatible versions bring only few new concepts at the instruction level, but imply new system programming approaches which are not the purpose of this book. The index covers only the first four chapters, in order to help the readers who study a given processor without having carefully read the definition chapters. The appendix includes the CALM reference cards for the seven processors. Reference cards for other processors may be obtained at the address below or from the CALM assembler distributors.

It is hoped that each chapter of the second part will provide enough material to allow the writing of complex programs with a minimum of additional information about the CALM assembler used. Learning to use the manufacturer's notation and the assembler pseudo-instructions simply requires a better familiarization with the manufacturer's notation and documentation.

We are grateful to the many friends who have carefully read parts of the book and checked its content. We would like to mention specially R. Beuchat, J. Borawski, P. Fäh, K. Hoyer, E. Skodaralakis, R. Sommer, B. Szafnicki and A. Wegmann. However, there is so much precise and specific information in this book that the risk of minor errors is high. The authors will be obliged to readers who signal immediately these errors; please send comments to LAMI-EPFL, Cour 37, CH-1007 Lausanne.

This book was edited and printed as it is, using the equipment developed at the "Laboratoire de Microinformatique" of the Swiss Federal Institute of Technology (EPFL). We thank cordially the engineers who wrote the text formatting software and guaranteed the maintenance of the equipment, especially D. Dumoulin, P. Fäh, A. Guignard and G. Vaucher.

J.D. Nicoud F. Wagner

14 INDEX

ADDC 48
ADD 48
AND 49
ASCII 14
ASL 50
ASR 50
Accumulator-based 20
Accumulator 16
Address or Index Register 19
Address specifier 35
Arithmetic and Logic Unit 16
Arithmetic numbers 6
BASE 37
BCD (Binary Coded Decimal) 1
Based addressing 37
Based-indexed addressing 38
Binary file 57
Binary system 1
CALL 50
CHECK 48
CLR 50
CLR 50
COMP 48
CONV 47
Cache memory 19
Control Unit 16
Control register 25
DAA 49
DEC 49
DIV 48
DJ,NE 52
DMA Controller 27
Direct memory access 27
Direct memory addressing 35
Direct register addressing 34
Dynamic range 9
EX 47
Error file 57
Excess-2^{e-1} 10
Execution cycle 17
Exponent 9
Fetch cycle 17
Fixed-point numbers 9
Flag Register 16
Flags 21
Floating-point numbers 9
General-register 20
Half carry 14
Halt state 17
Harvard type 18
Hexadecimal system 1
Hold 27
INC 49
IOFF 52
ION 52
IS0-7 14
Immediate addressing 33
Index register 38
Indexed addressing 37
Indexed addressing 37
Indirect addressing 36
Inherent 34
Instruction Register 16
Interface circuits 24
Interrupt Controller 26
Interrupt request 25
Interrupt service routine 26

JUMP 50
Label 58
Linker 60
Load effective address 47
Loader 60
Local label 58
Logical numbers 6
MOVE 47
MUL 48
Macroassemblers 57
Mantissa 9
Memory-mapped 24
NEG 49
NOP 52
NOT 49
OR 49
One's-complement 5
Overflow 7
POP 47
PUSH 47
Page register 23
Paged addressing 42
Pass 57
Polling 26
Pop 23
Post-increment (register indirect) 39
Post-indexed indirect 41
Pre-decrement (register indirect) 39
Pre-indexed indirect 41
Prefetch queue 19
Processor 16
Program Counter 16
Programmed transfer 25
Programming model 29
Pseudo-instructions 59
Push 23
RESET 17
RESET 52
RET 50
RLC 50
RRC 50
Recursive 64
Reentrant 63
Register indirect 37
Relative addressing 44
Relocatable 62
SET 50
SET 50
SL 50
SR 50
SUBC 48
SUB 48
SWAP 47
Short direct memory addressing 35
Sign extension 8
Signed decimal integers 12
Signed decimal integers 12
Signed-magnitude 5
Software interrupt 7
Software interrupts 27
Source file 57
Stack Pointer 19
Stack 23
Static range 9
Status register 25
Symbol 57
TCLR 50
TEST 50
TNOT 50
TRAP 52
TSET 50
Trap 76
Traps 27
Two's-complement 5
Von Neumann's architecture 15
WAIT 52
XOR 49

CONTENTS

1 **Binary Arithmetic** 1

- 1.1 POSITIONAL NUMBER SYSTEMS 1
- 1.2 NUMBER CONVERSIONS 3
- 1.3 BINARY REPRESENTATION OF SIGNED INTEGERS 5
- 1.4 BINARY ARITHMETIC FOR INTEGERS 7
  - 1.4.1 Unsigned Operations 7
  - 1.4.2 Two's-complement operation 8
  - 1.4.3 Sign-extension 8
- 1.5 FIXED- AND FLOATING-POINT NUMBERS 9
  - 1.5.1 Floating-point numbers 9
  - 1.5.2 Arithmetic for floating-point numbers 11
- 1.6 SIGNED DECIMAL INTEGERS 12
- 1.7 REPRESENTATION OF NONNUMERICAL DATA 14

2 **Microcomputer Operations** 15

- 2.1 VON NEUMANN ARCHITECTURE 15
- 2.2 PROCESSOR 16
  - 2.2.1 Basic architecture 16
  - 2.2.2 Other features 19
- 2.3 REGISTERS 19
- 2.4 ARITHMETIC UNIT AND FLAGS 21
- 2.5 MEMORY 22
- 2.6 STACK 23
- 2.7 INPUT-OUTPUT OPERATIONS 24
  - 2.7.1 Programmed transfer 25
  - 2.7.2 Data transfer with interrupt 25
  - 2.7.3 Direct memory access 27

3 **Microprocessor Instructions** 29

- 3.1 PROGRAMMING MODEL 29
- 3.2 REGISTERS 29
- 3.3 DATA TYPES 30
  - 3.3.1 Data specification 30
  - 3.3.2 8-bit microprocessors 31
  - 3.3.3 16-bit microprocessors 31
  - 3.3.4 Examples of data types 32
- 3.4 ADDRESSING MODES 33

3.5 IMMEDIATE ADDRESSING 33
3.6 DIRECT ADDRESSING 34
3.6.1 Direct register addressing 34
3.6.2 Direct memory addressing 35
3.6.3 Short direct memory addressing 35
3.7 INDIRECT ADDRESSING 36
3.7.1 Register indirect addressing 37
3.7.2 Post-increment and pre-decrement addressing 39
3.7.3 Memory indirect addressing 40
3.7.4 Paged addressing 42
3.8 RELATIVE ADDRESSING 44
3.9 INSTRUCTION FORMAT 45
3.10 INSTRUCTION SET 46
3.10.1 Data transfers 46
3.10.2 "Load Address" instruction 47
3.10.3 Arithmetic operations 48
3.10.4 Logical operations 49
3.10.5 Shift and rotate operations 49
3.10.6 Bit operations 50
3.10.7 Program control 50
3.10.8 Interrupt control 52
3.10.9 Specific instructions 52
3.11 DATA STRUCTURES 52
3.11.1 Tables and Arrays 53
3.11.2 Queues 53
3.11.3 Lists 54
3.11.4 Records 55

4 Assembly Language Programs 57

4.1 ASSEMBLER OPERATIONS 57
4.1.1 Symbols 57
4.1.2 Instructions 58
4.1.3 Code-generation pseudo-operations 59
4.1.4 Other pseudo-instructions 59
4.1.5 Comments 59
4.1.6 Linker and loader 60
4.2 PROGRAMMING 60
4.2.1 Processor model 60
4.2.2 Addressing modes 61
4.2.3 Programming techniques 62
4.2.4 Relocation 62
4.2.5 Register management 62
4.2.6 Parameter passing 63
4.2.7 Reentrance 63
4.2.8 Recursivity 64
4.2.9 Error handling 65

此为试读，需要完整PDF请访问： www.ertongbook.com

4.3 BRANCH AND LOOP STRUCTURES 65
4.3.1 Initialization loop 65
4.3.2 Move block 66
4.3.3 Lower to uppercase conversion 67
4.4 TABLE HANDLING 68
4.4.1 General lower to uppercase conversion 68
4.4.2 Branch on value 69
4.4.3 Bubble sort 70
4.5 ARITHMETIC 72
4.5.1 Multiple increment 72
4.5.2 Binary to Gray conversion 73
4.5.3 Gray to Binary conversion 74
4.5.4 Binary to decimal (BCD) conversion 75
4.6 INDIRECT ADDRESSING MODES 76
4.6.1 System call 76
4.6.2 Linked list 79
4.7 Conclusion 80

5 Motorola 6809 81

5.1 HARDWARE 81
5.1.1 Data transfer and instruction execution 81
5.1.2 Memory and peripheral interface 82
5.1.3 Processor control 82
5.1.4 Bus control 83
5.1.5 Interrupt control 84
5.2 SOFTWARE 85
5.2.1 Programming model 85
5.2.2 Addressing modes 86
5.2.3 Immediate addressing 86
5.2.4 Register direct addressing 88
5.2.5 Memory direct and relative addressing 88
5.2.6 Register indirect addressing 89
5.2.7 Memory indirect addressing 90
5.2.8 Instruction format 90
5.2.9 Instruction set 92
5.2.10 Transfer instructions 92
5.2.11 2-operand arithmetic and logical instructions 94
5.2.12 1-operand arithmetic and logical instructions 95
5.2.13 Jump instructions 96
5.2.14 Miscellaneous instructions 97
5.3 PROGRAMMING EXAMPLES 98
5.3.1 Clear a bit-map window 99
5.3.2 Move non-overlapping block 100
5.3.3 Convert lowercase letter to uppercase 100
5.3.4 Convert any 7-bit code 101

5.3.5 Jump according to input character 101
5.3.6 Jump according to consecutive values 102
5.3.7 Bubble sort 103
5.3.8 Multiple increment 105
5.3.9 Binary to Gray conversion 106
5.3.10 Gray to binary conversion 106
5.3.11 Binary to decimal conversion 108
5.3.12 System call 108
5.3.13 Linked list 109

6 Intel 8085 111

6.1 HARDWARE 111
6.1.1 Data transfer and instruction execution 112
6.1.2 Memory and peripheral interface 114
6.1.3 Processor control 114
6.1.4 Bus control 115
6.1.5 Serial I/O 115
6.1.6 Interrupts 115
6.2 SOFTWARE 117
6.2.1 Programming model 117
6.2.2 Addressing modes 118
6.2.3 Instruction format 120
6.2.4 Instruction set 121
6.2.5 Transfer instructions 121
6.2.6 2-operand arithmetic and logical instructions 122
6.2.7 1-operand arithmetic and logical instructions 123
6.2.8 Jump instructions 123
6.2.9 Miscellaneous instructions 124
6.3 PROGRAMMING EXAMPLE 125
6.3.1 Clear a bit-map window 125
6.3.2 Move non-overlapping block 126
6.3.3 Convert lowercase letter to uppercase 127
6.3.4 Convert any 7-bit code 127
6.3.5 Jump according to input character 128
6.3.6 Jump according to consecutive values 129
6.3.7 Bubble sort 129
6.3.8 Multiple increment 130
6.3.9 Binary to Gray conversion 132
6.3.10 Gray to binary conversion 132
6.3.11 Binary to decimal conversion 133
6.3.12 System call 135
6.3.13 Linked list 136

7 Zilog Z80 and National Semiconductor NSC800 139

7.1 Z80 HARDWARE 139
7.1.1 Data transfer and instruction execution 140
7.1.2 Memory and peripheral interface 141

7.1.3 Processor control 141
7.1.4 Bus control 141
7.1.5 Interrupts 141
7.2 NSC800 HARDWARE 142
7.2.1 Data transfer and instruction execution 143
7.2.2 Memory interface and processor control 143
7.2.3 Bus control 143
7.2.4 Interrupts 143
7.3 SOFTWARE 143
7.3.1 Programming model 144
7.3.2 Addressing modes 145
7.3.3 Instruction format 147
7.3.4 Instruction set 147
7.3.5 Transfer instructions 147
7.3.6 Block transfer and compare instruction 149
7.3.7 2-operand arithmetic and logic instructions 151
7.3.8 1-operand arithmetic and logic instructions 151
7.3.9 Jump and call instructions 152
7.3.10 Miscellaneous instructions 153
7.4 PROGRAMMING EXAMPLES 154
7.4.1 Clear a bit-map window 154
7.4.2 Move non-overlapping blocks 155
7.4.3 Convert lower case to upper case 156
7.4.4 Convert any 7-bit code 156
7.4.5 Jump according to input character 157
7.4.6 Jump according to consecutive values 157
7.4.7 Bubble sort 158
7.4.8 Multiple increment 159
7.4.9 Binary to Gray conversion 160
7.4.10 Gray to binary conversion 161
7.4.11 Binary to decimal conversion 162
7.4.12 System call 163
7.4.13 Linked list 164

8 Intel 8048 167

8.1 HARDWARE 169
8.1.1 Initialization 170
8.1.2 Transfer control signals 170
8.1.3 Execution of instructions 171
8.1.4 External data memory 172
8.1.5 Input-output ports 173
8.1.6 Input-output port expansion 173
8.1.7 T0 and T1 lines 174
8.1.8 Counter 175
8.1.9 Interrupts 176
8.1.10 External memories 177

8.2 SOFTWARE 178
8.2.1 Programming model 178
8.2.2 Addressing modes 182
8.2.3 Instruction format 184
8.2.4 Instruction set 185
8.2.5 Transfer instructions 185
8.2.6 2-operand arithmetic and logical instructions 186
8.2.7 1-operand arithmetic and logical instructions 186
8.2.8 Jump instructions 187
8.2.9 1-bit logical instructions 188
8.3 PROGRAMMING EXAMPLE 189
8.3.1 Clear a bit-map window 189
8.3.2 Move non-overlapping block 190
8.3.3 Convert lowercase letters to uppercase 190
8.3.4 Convert any 7-bit code 191
8.3.5 Jump according to input character 191
8.3.6 Jump according to consecutive values 192
8.3.7 Bubble sort 192
8.3.8 Multiple increment 193
8.3.9 Binary to Gray conversion 195
8.3.10 Gray to binary conversion 195
8.3.11 Binary to decimal conversion 196
8.3.12 System call 197
8.3.13 Linked list 197
8.3.14 Interrupt handling 198

9 Motorola 6801 199

9.1 HARDWARE 200
9.1.1 Data transfer in modes 2 and 3 203
9.1.2 Processor control 203
9.1.3 Parallel I/O 204
9.1.4 Serial I/O 205
9.1.5 Timer 205
9.1.6 Interrupts 206
9.2 SOFTWARE 207
9.2.1 Programming model 207
9.2.2 Addressing modes 208
9.2.3 Instruction format examples 209
9.2.4 Instruction set 209
9.2.5 Transfer instructions 210
9.2.6 2-operand arithmetic and logic instructions 211
9.2.7 1-operand arithmetic and logic instructions 211
9.2.8 Jump and call instructions 212
9.2.9 Miscellaneous instructions 213
9.3 PROGRAMMING EXAMPLES 213
9.3.1 Clear a bit-map window 214
9.3.2 Move non-overlapping block 215

9.3.3 Convert lowercase letter to uppercase 216
9.3.4 Convert any 7-bit code 216
9.3.5 Jump according to input character 217
9.3.6 Jump according to consecutive values 217
9.3.7 Bubble sort 219
9.3.8 Multiple increment 220
9.3.9 Binary to Gray conversion 221
9.3.10 Gray to binary conversion 222
9.3.11 Binary to decimal conversion 223
9.3.12 System call 224
9.3.13 Linked list 226

10 Intel 8086 229

10.1 HARDWARE 230
10.1.1 Minimum and maximum mode 230
10.1.2 Data transfer and instruction execution 232
10.1.3 Memory and peripheral interface 234
10.1.4 Processor control 235
10.1.5 Bus control 236
10.1.6 Interrupt control 237

10.2 SOFTWARE 240
10.2.1 Programming model 240
10.2.2 Addressing modes 243
10.2.3 Instruction format 245
10.2.4 Instruction set 248
10.2.5 Transfer instructions 248
10.2.6 2-operand arithmetic and logical instructions 250
10.2.7 1-operand arithmetic and logical instructions 251
10.2.8 Jump instructions 252
10.2.9 String manipulation instructions 253
10.2.10 Miscellaneous instructions 254

10.3 PROGRAMMING EXAMPLES 255
10.3.1 Clear a bit-map window 256
10.3.2 Move non-overlapping block 257
10.3.3 Convert lowercase letter to uppercase 257
10.3.4 Convert any 7-bit code 258
10.3.5 Jump according to input character 258
10.3.6 Jump according to consecutive values 259
10.3.7 Bubble sort 259
10.3.8 Multiple increment 260
10.3.9 Binary to Gray conversion 261
10.3.10 Gray to binary conversion 262
10.3.11 Binary to decimal conversion 263
10.3.12 System call 264
10.3.13 Linked list 265

11 Motorola 68000 **267**
11.1 HARDWARE 267
11.1.1 Data transfers 268
11.1.2 Privileged modes 270
11.1.3 6800 compatibility mode 270
11.1.4 Interrupts 270
11.1.5 Processor control 271
11.1.6 Bus control 272
11.1.7 Vector assignment 272
11.2 SOFTWARE 273
11.2.1 Programming model 273
11.2.2 Data types 274
11.2.3 Addressing modes 275
11.2.4 Immediate and direct addressing 275
11.2.5 Register indirect addressing 277
11.2.6 Relative addressing 277
11.2.7 Generalized immediate addressing 278
11.2.8 Bit sub-addresses 278
11.2.9 Group of address mode 279
11.2.10 Instruction format 280
11.2.11 Instruction set 281
11.2.12 Transfer instructions 281
11.2.13 2-operand arithmetic and logic instructions 283
11.2.14 1-operand arithmetic and logic instructions 285
11.2.15 Jump and call instructions 287
11.2.16 Miscellaneous instructions 288
11.3 PROGRAM EXAMPLES 288
11.3.1 Clear a bit-map window 288
11.3.2 Move non-overlapping blocks 289
11.3.3 Convert lower case to upper case 290
11.3.4 Convert any 7-bit code 290
11.3.5 Jump according to input character 291
11.3.6 Jump according to consecutive values 291
11.3.7 Bubble sort 292
11.3.8 Multiple increment 293
11.3.9 Binary to Gray conversion 294
11.3.10 Gray to binary conversion 294
11.3.11 Binary to decimal conversion 295
11.3.12 System call 296
11.3.13 Linked list 297
12 Appendix **299**
13 References **329**
14 Index **331**

此为试读，需要完整PDF请访问：www.ertongbook.com