

AccessionIndex: TCD-SCSS-V.20160121.002

Accession Date: 21-Jan-2016

Accession By: Peter Canavan

Object name: Programming the Z80

Vintage: c.1979

Synopsis: Rodney Zaks, ISBN 0-89588-013-X, Sybex Inc, 2344 Sixth Street, Berkeley, California 94710.

Description:

Rodney Zaks books (also see *Microprocessor Interfacing Techniques* elsewhere in this catalog) were popular guides for design engineers, students and enthusiasts in the late 1970s.

When introduced in Jul-1976 the Zilog Z80 was the most powerful 8-bit CPU on the market. Its internal circuitry was designed by the legendary Masatoshi Shima, while Federico Faggin designed the instruction set, a superset of that of the Intel i8080.

Programming the Z80 describes the Z80 CPU hardware, instruction set and address modes, and basic I/O techniques, then covers application examples, data structures and program development.

The instruction set is described in very great detail, so this book was an excellent reference for such details. It is also a good snapshot of the simplicity of a mid-to-late 1970s microprocessor, just before the evolution of increasingly complex 16/32/64-bit architectures.

Many thanks to Peter Canavan, Network Manager, Broadcast Operations, Australian Broadcasting Commission (ABC), who donated this item from his personal collection.

The homepage for this catalog is at: <https://www.scss.tcd.ie/SCSSTreasuresCatalog/>
Click 'Accession Index' (1st column listed) for related folder, or 'About' for further guidance. Some of the items below may be more properly part of other categories of this catalog, but are listed here for convenience.

Accession Index	Object with Identification
TCD-SCSS-V.20160121.002	Rodney Zaks, Programming the Z80, ISBN 0-89588-013-X, Sybex Inc, 2344 Sixth Street, Berkeley, California 94710, c.1979.



Figure 1: Programming the Z80 front cover

PROGRAMMING THE Z80

has been designed both as an educational text and as a self-contained reference book. As such, it can be used as a complete introductory book on programming, ranging from the basic concepts to advanced data structures manipulations.

It also contains a comprehensive description of all the Z80 instructions as well as its internal operation, and should provide a comprehensive reference for the reader who is already familiar with the principles of programming, but wishes to learn the Z80.

This book is the result of extensive experience by the author in the field of education and programming. As such, it has been designed to be clear and easy to read. All concepts are explained in simple yet precise terms, building progressively towards more complex techniques. The reader will gain not only an understanding of programming in the language of the Z80 but also a detailed understanding of the way a microprocessor such as the Z80 actually executes instructions. The reader will follow the flow of execution between the various registers and along the buses. This is indispensable for effective programming at machine level in the world of microprocessors. Because programming is not just the skill of coding an algorithm into a programming language but also the art of designing appropriate data structures, an extensive chapter on data structures is presented, which both introduces the concepts and actual application programs. The reader will find there lists, tables, binary trees, and the required algorithms.

After reading this book, the reader should have acquired all the basic skills required to program not just at the elementary level, but in most practical cases.

ABOUT THE AUTHOR

Dr. Rodney Zaks has been involved with the industrial use of microprocessors since they first appeared. He is the author of a number of best-selling books on all aspects of microprocessors. He has taught microprocessor courses to more than 5,000 people internationally, ranging from the introductory level to bit-slice microprogramming techniques. He holds a Ph. D. in Computer Science from U.C. Berkeley, and is a member of ACM and IEEE.

NOTICE: "Z80" is a registered trademark of ZILOG Inc. SYBEX is not connected in any way to ZILOG Inc.

ISBN#0-89588-013-X

Figure 2: Programming the Z80 rear cover

CANAUVAN

JUNE 80

2710320

programming the Z80

RODNAY ZAKS



Figure 3: Programming the Z80 title page

ACKNOWLEDGEMENTS

Designing a programming textbook is always difficult. Designing it so that it will teach elementary programming as well as advanced concepts while covering both hardware and software aspects makes it a challenge. The author would like to acknowledge here the many constructive suggestions for improvements or changes made by: O.M. Barlow, Dennis L. Feick, Richard D. Reid, Stanley E. Erwin, Philip Hooper, Dennis B. Kitsz.

A special acknowledgement is also due to Chris Williams for his contribution to the instruction-set and the data structures section.

Any additional suggestions for improvements or changes should be sent to the author, and will be reflected in forthcoming editions.

Several tables in Chapter Four showing hexadecimal codes for the Z80 instructions have been reprinted by permission of Zilog Inc.

Cover Design by Daniel le Noury

Every effort has been made to supply complete and accurate information. However, Sybex assumes no responsibility for its use; nor any infringements of patents or other rights of third parties which would result. No license is granted by the equipment manufacturers under any patent or patent rights. Manufacturers reserve the right to change circuitry at any time without notice.

In particular, technical characteristics and prices are subject to rapid change. Comparisons and evaluations are presented for their educational value and for guidance principles. The reader is referred to the manufacturer's data for exact specification.

Copyright © 1979, SYBEX Inc. World Rights reserved. No part of this publication may be stored in retrieval system, transmitted, or reproduced in any way, including but not limited to, photocopy, photography, magnetic or other record, without the prior written permission of the publisher.

Library of Congress Card Number: 78-73741

ISBN Number: 0-89588-013-X

Printed in the United States of America

Printing 10 9 8 7 6 5 4 3 2 1

Figure 4: Programming the Z80 publishers page

TABLE OF CONTENTS

PREFACE	13
I. BASIC CONCEPTS	15
<i>Introduction, What is programming?, Flowcharting, Information Representation</i>	
II. Z80 HARDWARE ORGANIZATION	46
<i>Introduction, System Architecture, Internal Organization of the Z80, Instruction Formats, Execution of Instructions with the Z80, Hardware Summary</i>	
III. BASIC PROGRAMMING TECHNIQUES	94
<i>Introduction, Arithmetic Programs, BCD Arithmetic Multiplication, Binary Division, Instruction Summary, Subroutines, Summary</i>	
IV. THE Z80 INSTRUCTION SET	154
<i>Introduction, Classes of Instructions, Summary, Individual Descriptions</i>	
V. ADDRESSING TECHNIQUES	438
<i>Introduction, Possible Addressing Modes, Z80 Addressing modes, Using the Z80 Addressing Modes, Summary</i>	
VI. INPUT/OUTPUT TECHNIQUES	460
<i>Introduction, Input/output, Parallel Word Transfer, Bit Serial Transfer, Peripheral Summary, Input/Output Scheduling, Summary</i>	
VII. INPUT/OUTPUT DEVICES	511
<i>Introduction, The Standard PIO, The Internal Control Register, Programming a PIO, The Zilog Z80 PIO</i>	

Figure 5: Programming the Z80 table of contents page 1

VIII. APPLICATION EXAMPLES	520
<i>Introduction, Clearing a Section of Memory, Polling I/O Devices, Getting Characters In, Testing A Character, Bracket Testing, Parity Generation, Code Conversion: ASCII to BCD, Convert Hex to ASCII, Finding the Largest Element of a Table. Sum of N Elements, A Checksum Computation, Count the Zeroes, Block Transfer, BCD Block Transfer, Compare Two Signed 16-Bit Numbers, Bubble-Sort, Summary</i>	
IX. DATA STRUCTURES	539
<i>PART 1—THEORY</i>	
<i>Introduction, Pointers, Lists, Searching and Sorting, Section Summary</i>	
<i>PART 2—DESIGN EXAMPLES</i>	
<i>Introduction, Data Representation for the List, A Simple List, Alphabetic Set, Linked List, Summary</i>	
X. PROGRAM DEVELOPMENT	579
<i>Introduction, Basic Programming Choices, Software Support, The Program Development Sequence, Hardware Alternatives, The Assembler, Conditional Assembly, Summary</i>	
XI. CONCLUSION.....	602
<i>Technological Development, The Next Step</i>	
APPENDIX A.....	604
<i>Hexadecimal Conversion Table</i>	
APPENDIX B.....	605
<i>ASCII Conversion Table</i>	
APPENDIX C.....	606
<i>Relative Branch Tables</i>	
APPENDIX D.....	607
<i>Decimal to BCD Conversion</i>	
APPENDIX E.....	608
<i>Z80 Instruction Codes</i>	
APPENDIX F.....	615
<i>Z80 to 8080 Equivalence</i>	
APPENDIX G.....	616
<i>8080 to Z80 Equivalence</i>	
INDEX.....	617

Figure 6: Programming the Z80 table of contents page 2

LIST OF ILLUSTRATIONS

Figure 1.1:	A Flowchart for Keeping Room Temperature Constant	17
Figure 1.2:	Decimal-Binary Table	21
Figure 1.3:	2's Complement Table	29
Figure 1.4:	BCD Table	35
Figure 1.5:	Typical Floating-Point Representation	38
Figure 1.6:	ASCII Conversion Table	40
Figure 1.7:	Octal Symbols	42
Figure 1.8:	Hexadecimal Codes	43
Figure 2.1:	Standard Z80 System	47
Figure 2.2:	"Standard" Microprocessor Architecture	49
Figure 2.3:	Shift and Rotate	50
Figure 2.4:	The 16-bit Address Registers Create the Address Bus	52
Figure 2.5:	The Two-Stack Manipulation Instructions	54
Figure 2.6:	Fetching an Instruction from the Memory	56
Figure 2.7:	Automatic Sequencing	56
Figure 2.8:	Single-Bus Architecture	57
Figure 2.9:	Execution of an Addition—R0 into ACC	58
Figure 2.10:	Addition—Second Register R1 into ALU	58
Figure 2.11:	Result Is Generated and Goes into R0	59
Figure 2.12:	The Critical Race Problem	60
Figure 2.13:	Two Buffers Are Required	61
Figure 2.14:	Internal Z80 Organization	65
Figure 2.15:	Typical Instruction Formats	67
Figure 2.16:	The Register Codes	68
Figure 2.17:	Instruction Fetch—(PC) Is Sent to the Memory	70
Figure 2.18:	PC Is Incremented	71
Figure 2.19:	The Instruction Arrives from the Memory into IR	72
Figure 2.20:	Transferring C into D	73
Figure 2.21:	The Contents of C Are Deposited INTO TMP	73
Figure 2.22:	The Contents of TMP Are Deposited into D	74
Figure 2.23:	Two Transfers Occur Simultaneously	76
Figure 2.24:	End of ADD r	77
Figure 2.25:	Fetch-Execute Overlap during T1-T2	78
Figure 2.26:	Intel Abbreviations	79
Figure 2.27:	Intel Instruction Formats	80
Figure 2.28:	Transfer Contents of HL to Address Bus	85
Figure 2.29:	LD A, (ADDRESS) Is a 3-Word Instruction	86
Figure 2.30:	Before Execution of LD A	87
Figure 2.31:	After Execution of LD A	87
Figure 2.32:	Second Byte of Instruction Goes into Z	88
Figure 2.33:	Z80 MPU Pinout	91

Figure 7: Programming the Z80 list of illustrations page 1

Figure 3.0:	The Z80 Registers.....	95
Figure 3.1:	Eight-Bit Addition $RES = OP1 + OP2$	96
Figure 3.2:	LD A, (ADR1): OP1 is Loaded from Memory.....	97
Figure 3.3:	ADD A, (HL).....	98
Figure 3.4:	LD (ADR3), A (Save Accumulation in Memory).....	98
Figure 3.5:	16-Bit Addition—The Operands.....	100
Figure 3.6:	Storing Operands in Reverse Order.....	102
Figure 3.7:	Pointing to the High Byte.....	103
Figure 3.8:	A 32-Bit Addition.....	104
Figure 3.9:	16-Bit Load—LD HL, (ADR1).....	106
Figure 3.10:	Storing BCD Digits.....	109
Figure 3.11:	Packed BCD Subtract: $N1 \leftarrow N2 - N1$	111
Figure 3.12:	The Basic Multiplication Algorithm—Flowchart.....	114
Figure 3.13:	8 x 8 Multiplication Program.....	115
Figure 3.14:	8 x 8 Multiplication—The Registers.....	116
Figure 3.15:	LD BC, (MPRAD).....	117
Figure 3.16:	LD DE (MPDAD).....	118
Figure 3.17:	Shift and Rotate.....	120
Figure 3.18:	Shifting from E into D.....	121
Figure 3.19:	Form for Multiplication Exercise.....	123
Figure 3.20:	Multiplication: After One Instruction.....	124
Figure 3.21:	Multiplication: After Two Instructions.....	124
Figure 3.22:	Multiplication: After Five Instructions.....	125
Figure 3.23:	One Pass Through The Loop.....	125
Figure 3.24:	Improved Multiply, Step 1.....	127
Figure 3.25:	Registers for Improved Multiply.....	128
Figure 3.26:	Improved Multiply, Step 2.....	129
Figure 3.27:	16 x 16 Multiply—The Registers.....	130
Figure 3.28:	16 x 16 Multiplication Program.....	131
Figure 3.29:	16 x 16 Multiply with 32 Bit Result.....	133
Figure 3.30:	8-Bit Binary Division Flowchart.....	134
Figure 3.31:	16 x 8 Division—The Registers.....	134
Figure 3.32:	16 x 8 Division Program.....	135
Figure 3.33:	Form for Division Program.....	137
Figure 3.34:	Non-Restoring Divisor—The Registers.....	138
Figure 3.35:	Subroutine Call.....	143
Figure 3.36:	Nested Calls.....	145
Figure 3.37:	The Subroutine Calls.....	146
Figure 3.38:	Stack vs. Time.....	146
Figure 3.39:	Multiplication: A Complete Trace.....	151
Figure 3.40:	The Multiplication Program (Hex).....	153
Figure 3.41:	Two Iterations Through the Loop.....	153
Figure 4.1:	Shift and Rotate.....	156
Figure 4.2:	Eight-Bit Load Group—'LD'.....	160
Figure 4.3:	16-Bit Load Group—'LD', 'PUSH' and 'POP'.....	161
Figure 4.4:	Exchanger 'EX' and 'EXX'.....	162
Figure 4.5:	Block Transfer Group.....	163
Figure 4.6:	Block Search Group.....	165
Figure 4.7:	Eight-Bit Arithmetic and Logic.....	166

Figure 8: Programming the Z80 list of illustrations page 2

Figure 4.8:	Sixteen-Bit Arithmetic and Logic	167
Figure 4.9:	Shift and Rotate	170
Figure 4.10:	Rotates and Shifts	171
Figure 4.11:	Nine-Bit Rotation	171
Figure 4.12:	Eight-Bit Rotation	171
Figure 4.13:	Digit Rotate Instructions	172
Figure 4.14:	Bit Manipulation Group	173
Figure 4.15:	General-Purpose AF Operation	174
Figure 4.16:	The Flags Register	174
Figure 4.17:	Summary of Flag Operations	180
Figure 4.18:	Jump Instructions	182
Figure 4.19:	Restart Group	183
Figure 4.20:	Output Group	186
Figure 4.21:	Input Group	186
Figure 4.22:	Miscellaneous CPU Control	187
Figure 5.1:	Basic Addressing Modes	440
Figure 5.2:	Addressing (Pre-Indexing)	442
Figure 5.3:	Indirect Indexed Addressing (Post-Indexing)	443
Figure 5.4:	Indirect Addressing	444
Figure 5.5:	Indexed Addressing Has 2-Byte Opcode	447
Figure 5.6:	Character Search Flowchart	450
Figure 5.7:	Block Transfer: Initializing the Register	451
Figure 5.8:	A Block Transfer-Memory Map	453
Figure 5.9:	Adding Two Blocks: $BLK1 = BLK1 + BLK2$	456
Figure 5.10:	Memory Organization for Block Transfer	458
Figure 6.1:	Turning on a Relay	462
Figure 6.2:	A Programmed Pulse	462
Figure 6.3:	Basic Delay Flowchart	464
Figure 6.4:	Parallel Word Transfer: The Memory	468
Figure 6.5:	Parallel Word Transfer: Flowchart	469
Figure 6.6:	Bit Serial Transfer-Flowchart	472
Figure 6.7:	Serial-to-Parallel: The Registers	474
Figure 6.8:	Handshaking (Output)	478
Figure 6.8a:	Handshaking (Input)	478
Figure 6.9:	Printer—Data Paths	479
Figure 6.10:	Seven Segment LED	481
Figure 6.11:	Hexadecimal Characters Generated with a Seven-Segment LED	481
Figure 6.12:	Format of a Teletype Word	485
Figure 6.13:	TTY Input with Echo	486
Figure 6.14:	Teletype Program	487
Figure 6.15:	Teletype Input	488
Figure 6.16:	Teletype Output	489
Figure 6.17:	Printing a Memory Block	491
Figure 6.18:	Three Methods of I/O Control	492
Figure 6.19:	Polling Loop Flowchart	494
Figure 6.20:	Reading from a Paper-Tape Reader	494
Figure 6.21:	Printing on a Punch or Printer	495
Figure 6.22:	Z80 Stack After Interruption	496
Figure 6.23:	Saving Some Working Registers	496
Figure 6.24:	Interrupt Sequence	497

Figure 9: Programming the Z80 list of illustrations page 3

	NMI Forces Automatic Vectoring	499
Figure 6.25:	Interrupt Mode 0	501
Figure 6.26:	Saving the Registers	502
Figure 6.27:	Mode 1 Interrupt	503
Figure 6.28:	Mode 2 Interrupt	504
Figure 6.29:	Mode 2: A Practical Example	505
Figure 6.30:	Polled vs. Vectored Interrupt	506
Figure 6.31:	Several Devices May Use the Same Interrupt Line	507
Figure 6.32:	Stack Contents During Multiple Interrupts.	508
Figure 6.33:	Interrupt Logic	510
Figure 6.34:	Typical PIO	512
Figure 7.1:	Using a PIO—Load Control Register	513
Figure 7.2:	Using a PIO—Load Data Direction	514
Figure 7.3:	Using a PIO—Read Status	514
Figure 7.4:	Using a PIO—Read INPUT	515
Figure 7.5:	Z80 PIO Pinout	516
Figure 7.6:	Z80 Block Diagram	517
Figure 7.7:	Largest Element in a Table	527
Figure 8.1:	Sum of N Elements	528
Figure 8.2:	BCD Block Transfer-the Memory	530
Figure 8.3:	Comparing Two Signed Numbers	532
Figure 8.4:	Bubble-Sort Examples: Phases 1 to 12	534
Figure 8.5:	Bubble-Sort Example: Phases 13 to 21	535
Figure 8.6:	Bubble-Sort Flowchart	536
Figure 8.7:	Bubble-Sort	537
Figure 8.8:	Bubble-Sort	537
Figure 9.1:	An Indirection Pointer	540
Figure 9.2:	A Directory Structure	541
Figure 9.3:	A Linked List	542
Figure 9.4:	Inserting a New Block	542
Figure 9.5:	A Queue	543
Figure 9.6:	Round Robin is Circular List	545
Figure 9.7:	Genealogical Tree	545
Figure 9.8:	Doubly-Linked List	546
Figure 9.10:	Typical List Entries in the Memory	549
Figure 9.11:	The Simple List	550
Figure 9.12:	Table Search Flowchart	551
Figure 9.13:	Table Insertion Flowchart	552
Figure 9.14:	Deleting an Entry (Simple List)	553
Figure 9.15:	Table Deletion Flowchart	554
Figure 9.16:	Simple List- The Programs	555
Figure 9.17:	Simple List-A Sample Run	556
Figure 9.18:	Binary Search Flowchart	559
Figure 9.19:	A Binary Search	561
Figure 9.20:	Insert "BAC"	563
Figure 9.21:	Delete "BAC"	564
Figure 9.22:	Deletion Flowchart (Alphabetic List)	565
Figure 9.23:	Binary Search Program	566
Figure 9.24:	Alphabetic List—A Sample Run	569
Figure 9.25:	Linked List Structure	571

Figure 10: Programming the Z80 list of illustrations page 4

Figure 9.26:	Linked List—A Search	573
Figure 9.27:	Linked List: Example of Insertion	573
Figure 9.28:	Example of Deletion (Linked List)	574
Figure 9.29:	Linked List-The Programs	575
Figure 9.30:	Linked List-A Sample Run	577
Figure 10.1:	Programming Levels	581
Figure 10.2:	A Typical Memory Map	586
Figure 10.3:	Microprocessor Programming Form	591
Figure 10.4:	Assembler Output-An Example	593
Figure 10.5:	Operator Precedence	597

Figure 11: Programming the Z80 list of illustrations page 5

