Apple IIc and IIe Assembly Language

Apple IIc and IIe Assembly Language

Jules H. Gilder

Chapman and Hall
New York London

First published 1986
by Chapman and Hall
29 West 35 Street, New York, N.Y. 10001

Published in Great Britain by
Chapman and Hall Ltd
11 New Fetter Lane, London EC4P 4EE

ISBN-13: 978-0-412-01121-4

e-ISBN-13: 978-1-4684-6424-5

DOI: 10.1007/978-1-4684-6424-5

©1986 Chapman and Hall

All Rights Reserved. No part of this book may be
reprinted, or reproduced or utilized in any form
or by any electronic, mechanical or other means,
now known or hereafter invented, including
photocopying and recording, or in any information
storage or retrieval system, without permission in
writing from the publishers.

Library of Congress Cataloging-in-Publication Data

Gilder, Jules H., 1947-

Apple IIc and IIe assembly language.

Bibliography: p.

1. Apple IIc (Computer)--Programming.
2. Apple IIe (Computer--Programming. 3. Assembler
language (Computer program language) I. Title.

II. Title: Apple 2c and 2e assembly language.

III. Title: Apple two c and 2 e assembly language.

QA76.8.A66225G55 1986 005.265 86-2592

DEDICATION

Dedicated with love to Miriam.

Contents

Preface	1
1 Machine Code or Assembly Language (Why machine code?)	3
2 Numbers (Binary, hex and decimal, Binary to decimal conversion, Decimal to binary conversion, Binary to hex conversion, Hex to decimal conversion)	5
3 It All Adds Up! (Binary arithmetic, Addition, Subtraction, Binary coded decimal (BCD), BCD addition, BCD subtraction)	10
4 It's Logical (Logical operations, AND, OR, EOR)	17
5 The Registers (The accumulator, The index registers, The program counter)	20
6 A Poke at Machine Code (Code — the program counter, Entering machine code, The hex loader program, Calling machine code, Saving it out to disk, The Apple ROMs)	22
7 Status Symbols (The status register)	31
8 Addressing Modes I (Zero page addressing, Immediate addressing)	35
9 Bits and Bytes (Load, store and transfer, Paging memory)	38
10 Arithmetic in Assembler (Addition, Subtraction, Negation, Using BCD)	42
11 Addressing Modes II (Absolute addressing, Zero page indexed addressing, Absolute indexed addressing, Indirect addressing, Post-indexed indirect addressing, Pre-indexed absolute addressing, Implied and relative addressing)	51

12 Stacks of Fun (The stack, Stack instructions for saving data)	60
13 Looping (Loops, Counters, Comparisons, Branches, FOR . . . NEXT, Memory counters)	65
14 Subroutines and Jumps (Subroutines, Passing parameters, Jumps)	75
15 Shifts and Rotates (Arithmetic shift left, Logical shift right, Rotate left, Rotate right, Logically speaking, Printing binary!, BIT)	81
16 Multiplication and Division (Multiplication, Division)	90
17 Assembly Types (Conditional assembly, Look-up tables)	95
18 Floating a Point (The floating point accumulators, Using USR, Integer to floating point, Floating point to integer, Floating memory, The subroutines)	99
19 Speeding Up and Slowing Down	108
20 Interrupts and Breaks (Interrupts, Breaks)	110
21 Prepacked Utilities (Hex to binary conversion, Binary to hex conversion, Output ASCII string)	112
 Appendices	
1 The Screen	119
2 The 6502 and 65C02	121
3 The Instruction Set	123
4 Instruction Cycle Times	166
5 Apple // Memory Map	169
6 Branch Calculators	170
7 6502 and 65C02 Opcodes	171
General Index	175
Program Index	178

PREFACE

The Apple // series of computers represents one of the most versatile and powerful home computers available. If you've used your computer for a while, you've probably become quite familiar with Applesoft BASIC. That's good, because once you know that, this book will show you how to graduate from BASIC programming to assembly language programming. There are many reasons to program your Apple in assembly language. First and foremost is speed. Assembly language is about 100 times faster than BASIC. If you're thinking of writing games or business programs that do sorting, speed is of the essence and assembly language is a must. Assembly language programs usually also require less memory. Thus you can squeeze more complex programs into a smaller amount of memory. Finally, assembly language programs offer you a considerable amount of security, because they are more difficult to trace and change.

While assembly language is powerful, it doesn't have to be difficult to learn. In fact, if you can write programs in Applesoft BASIC, you're already half-way home. This book assumes you know BASIC and absolutely nothing about assembly language or machine language. Every effort has been made to write in nontechnical language and to set the chapters out in a logical manner, introducing new concepts in digestible pieces as and when they are needed, rather than devoting whole chapters to specific items. Wherever possible, practical programs are included to bring home the point being made, and in most instances they are analyzed and the function and operation of each instruction explained.

Apple //c and //e Assembly Language is completely self-contained, includes a full description of all the machine-code instructions available, and suggests suitable applications for their use. After a bit of theory in the opening chapters, the main registers of the 6502 and 65C02 are introduced and descriptions given of how, when, and where machine code routines can be entered. There is also a simple machine code monitor program to facilitate the entry of such routines.

After discussing the ways in which the microprocessor flags certain conditions to the outside world, some of the modes of addressing the chip are described. Machine code addition and subtraction routines are introduced, and the easiest ways of manipulating and saving data for future use by the program and the processor are described. Machine code loops, (equivalent to BASIC's FOR...NEXT...STEP) show how sections of code may be repeated, and subroutines and jumps take the place of BASIC's GOTO and GOSUB. Also included is a look at some of the more complicated procedures, such as multiplication and division, using the shift and rotate instructions.

Details of the routines in the Apple's ROMs are provided to make your programming task easier. And, finally, a comprehensive set of appendices provides quick reference to the sort of things you'll need to "want to know quickly" when you start writing your very own original machine code programs.

You've just taken your first step into an exciting, new world of computing. Good luck and happy computing.