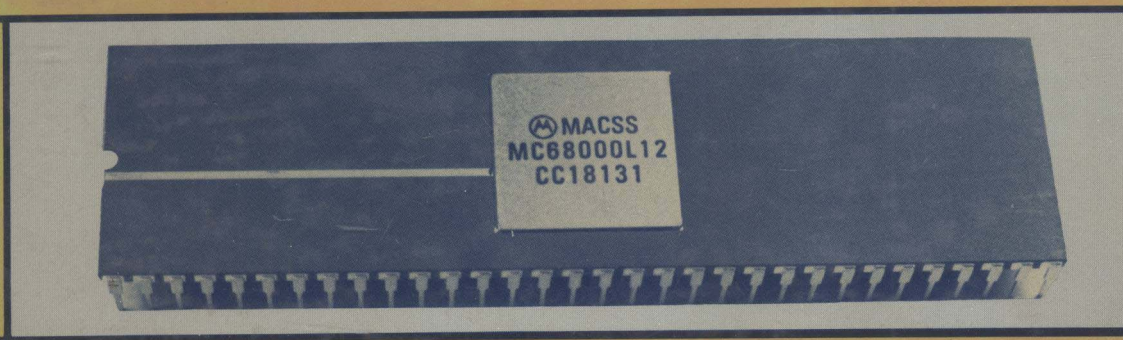
Assembly Language Programming FOR THE 68000



ARTHUR GILL
EDWARD CORWIN
ANTONETTE LOGAR

ASSEMBLY LANGUAGE PROGRAMMING FOR THE 68000

Arthur Gill
Edward Corwin
Antonette Logar

Prentice-Hall, Inc.
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data
Gill, Arthur (date)
Assembly language programming for the 68000.
Includes index.
1. Motorola 68000 (Microprocessor)—Programming.
2. Assembler language (Computer program language)
I. Corwin, Edward (date) . II. Logar,
Antonette (date) . III. Title.
QA76.8.M6895G55 1987 005.265 86-15118
ISBN 0-13-049529-8

To our parents

Editorial/production supervision: Linda Zuk
Cover design: Diane Saxe
Manufacturing buyer: Ed O'Dougherty
Cover photo courtesy of Motorola, Inc.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-049529-8 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

PREFACE

The objective of this book is to familiarize the reader with the basic organization and operational features of the Motorola 68000 and to present assembly language techniques for this computer. It is not, *per se*, a general text on machine structures, and the authors do not attempt to provide a comprehensive treatment of available computer organizations and assemblers. However, to the extent that the concepts and methods governing the operation and programming of the 68000 are used in many other computers, the material in this book should provide the reader with good preparation for the operation and programming of other machines.

Chapter 1 provides an outline of algorithms for converting numbers from one system (binary, hex, decimal) to another. The algorithms are not provided with proofs, and are intended to serve only as reference. Chapter 2 describes the organizational structure of the 68000 (the central memory, central processor, and peripheral devices). Chapter 3 explains how numbers (integer and floating-point), characters, and strings are represented in the 68000. Chapter 4 describes the 68000's instruction formats and addressing modes. And Chapter 5 introduces the reader to assembly language programming.

The first five chapters should provide the reader with sufficient background to write simple programs for the 68000. The remaining chapters delve deeper into operational details and describe additional techniques. Chapter 6 introduces stacks and subroutines (including recursion). Chapter 7 looks more closely at the 68000's arithmetic (including double-precision) and other operations, such as the test, comparison, branch, and shift operations. Chapter 8 explains the trap and interrupt mechanisms. Chapter 9 describes the workings of assemblers and linkage editors and the notion of relocation. (No particular assembler or linkage editor is used, and the concepts discussed are quite general.) Chapter 10 introduces some advanced assembler facilities, such as macros, repeated assembly, and conditional assembly.

The book ends with a number of appendixes, which consist of reference lists and tables (character codes, summary of addressing modes, lists of instructions, and so forth). An appendix on programming style is also included, which should be carefully read by the beginner.

Each chapter concludes with a set of exercises that serve to illustrate and sometimes complement the material in the text. The reader is encouraged to solve the problems and

run the programs included in these exercises. True assimilation of the material in this book can come about only through practice—by the actual writing and execution of programs.

The only prerequisite to this book is some experience with high-level language programming. No particular language is assumed, but it is taken for granted that the reader is familiar with the notions of an algorithm, a flowchart, and a stored program.

It is not our intent that this book stand alone as a course text. Since it does not describe all the fine details of the 68000 instructions and assembler directives, students should have access to a 68000 microprocessor handbook and the assembler manual appropriate to their particular installation, where these details can be found when needed. Little is said in the book regarding peripheral equipment (only the terminal and clock device are treated in detail), and students doing I/O programming may wish to refer to peripherals handbooks and local manuals for assistance.

The assembler used in this text contains features common to most 68000 assemblers. It is named TASTE, and was written by two South Dakota Tech graduate students, John Kjellevold and Jon Sundfjord.

The authors are indebted to Dr. Karen Whitehead for reviewing the manuscript and offering much useful advice. Thanks are also due Dr. David Ballew and Professors Harold Cards, Julie Dahl, and Roger Opp for helpful comments and suggestions. And, finally, thanks to Tara and Rob Powles for their help with the manuscript.

CONTENTS

1

NUMBER SYSTEMS

As we work with the 68000, we shall make extensive use of the binary and hexadecimal (hex) number systems, as well as the decimal system. It is important that the student acquire, as soon as possible, the facility to convert from one system to another. In this chapter we shall outline, without proof, some algorithms for carrying out these conversions. Students familiar with these algorithms may proceed directly to Chapter 2.

In this chapter, a "number" will mean a non-negative integer (0, 1, 2, . . .). The number N will be denoted by N_{10}, N_{16}, or N_2 if it is in the decimal, hexadecimal, or binary system, respectively. However, the subscript may be dropped if it is understood from the context.

An m-digit number will be written symbolically as $D_{m-1} \ldots D_1 D_0$ [D_i being the (i+1)st digit from the right].

In all flowcharts, an oval-shaped box will represent an entry or an exit point, and a diamond-shaped box will represent a branching point.

1.1 DECIMAL-TO-BINARY CONVERSION

The flowcharts in Figures 1.1 and 1.2 describe algorithms for converting a decimal number N into its binary equivalent M.

Example (Subtraction-of-powers method)

$$N = 217_{10}$$

$$217 - 2^7 = 217 - 128 = 89 \qquad (D_7 = 1)$$

$$89 - 2^6 = 89 - 64 = 25 \qquad (D_6 = 1)$$

$$25 - 2^4 = 25 - 16 = 9 \qquad (D_4 = 1)$$

$$9 - 2^3 = 9 - 8 = 1 \qquad (D_3 = 1)$$

$$1 - 2^0 = 1 - 1 = 0 \qquad (D_0 = 1)$$

$$M = 11011001_2$$

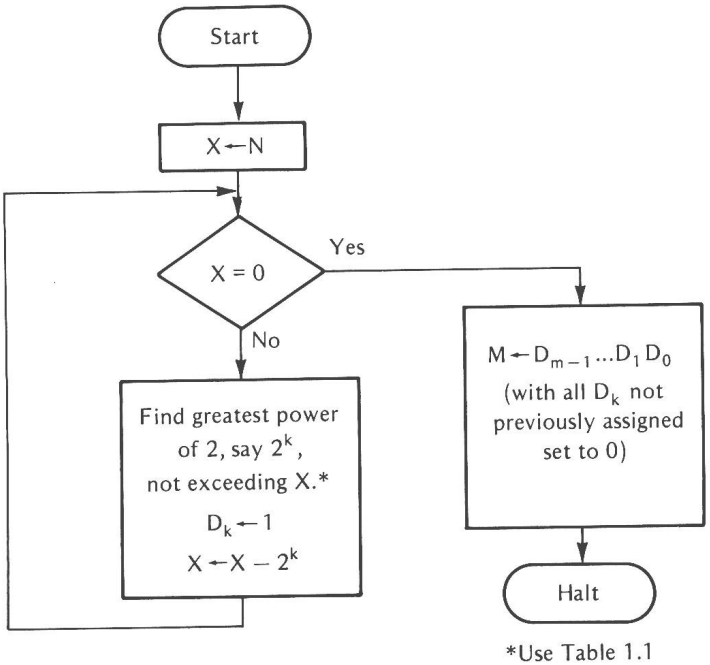


Figure 1.1 Decimal-to-binary conversion by subtraction of powers.

Example (Division method)

$N = 217_{10}$

217 is odd	$(D_0 = 1)$
$217/2 = 108$ is even	$(D_1 = 0)$
$108/2 = 54$ is even	$(D_2 = 0)$
$54/2 = 27$ is odd	$(D_3 = 1)$
$27/2 = 13$ is odd	$(D_4 = 1)$
$13/2 = 6$ is even	$(D_5 = 0)$
$6/2 = 3$ is odd	$(D_6 = 1)$
$3/2 = 1$ is odd	$(D_7 = 1)$
$1/2 = 0$	

$M = 11011001_2$

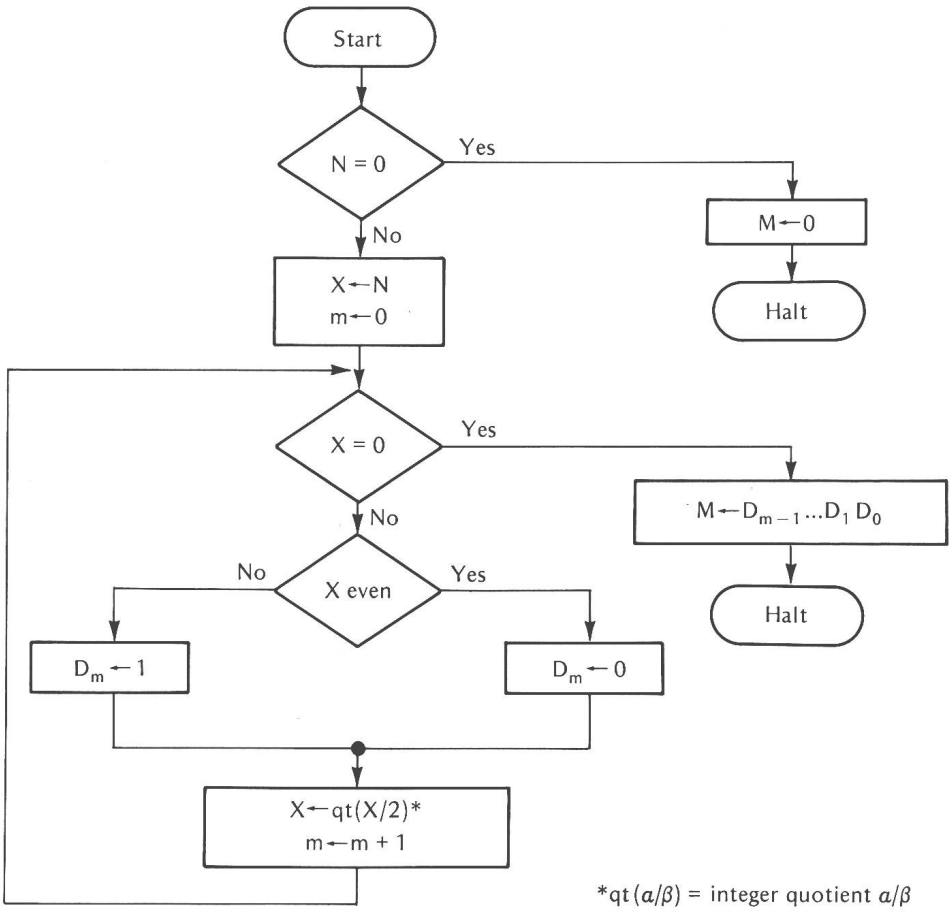


Figure 1.2 Decimal-to-binary conversion by division.

1.2 DECIMAL-TO-HEXADECIMAL CONVERSION

The flowcharts in Figures 1.3 and 1.4 describe algorithms for converting a decimal number N into its hexadecimal equivalent M. Powers of 16 are listed in Table 1.1, hexadecimal digits are listed in Table 1.2.

Example (Subtraction-of-powers method)

$$N = 2591_{10}$$

$$2591 - 10*16^2 = 2591 - 2560 = 31 \qquad (D_2 = 10 = A_{16})$$

$$31 - 1*16^1 = 31 - 16 = 15 \qquad (D_1 = 1)$$

$$15 - 15*16^0 = 15 - 15 = 0 \qquad (D_0 = 15 = F_{16})$$

$$M = A1F_{16}$$

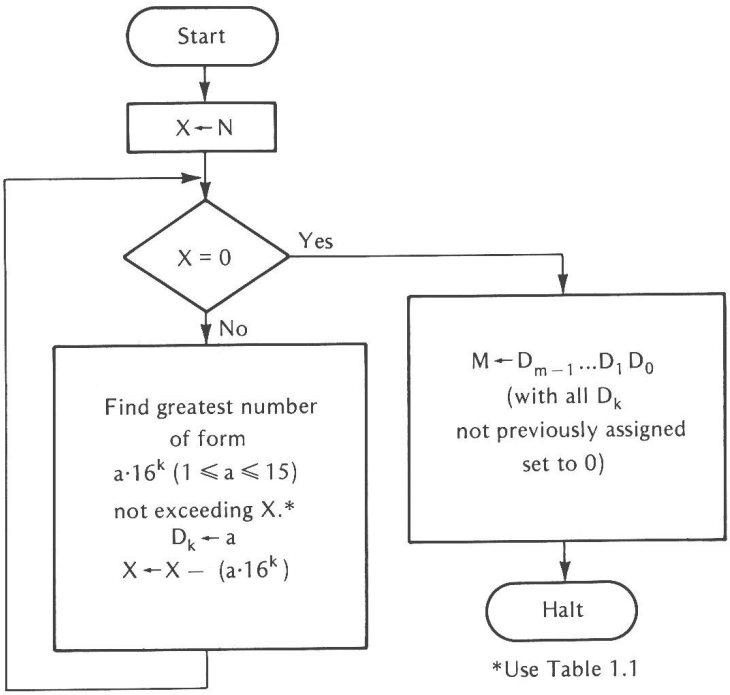


Figure 1.3 Decimal-to-hex conversion by subtraction of powers.

Example (Division method)

$$N = 2591_{10}$$

$$2591/16 = 161 \text{ (remainder } 15) \qquad (D_0 = F)$$

$$161/16 = 10 \text{ (remainder } 1) \qquad (D_1 = 1)$$

$$10/16 = 0 \text{ (remainder } 10) \qquad (D_2 = A)$$

$$M = A1F_{16}$$

An alternative method consists of first converting N into binary as shown in Section 1.1, and then converting the result into hexadecimal as shown in Section 1.6.

1.3 BINARY-TO-DECIMAL CONVERSION

If $N = D_{m-1} \ldots D_1 D_0$ is a binary number, then its decimal equivalent is:

$$M = \sum_{i=0}^{m-1} D_i * 2^i$$

(Powers of 2 are listed in Table 1.1)

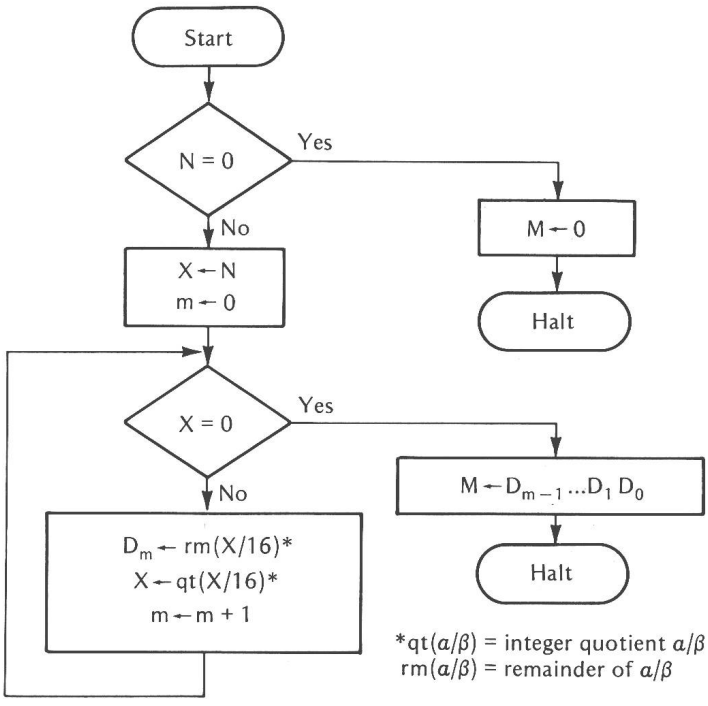


Figure 1.4 Decimal-to-hex conversion by division.

Example

$$N = 1011100$$

$$M = 2^2 + 2^3 + 2^4 + 2^6 = 4 + 8 + 16 + 64 = 92_{10}$$

1.4 HEXADECIMAL-TO-DECIMAL CONVERSION

If $N = D_{m-1} \ldots D_1 D_0$ is a hexadecimal number, then its decimal equivalent is:

$$M = \sum_{i=0}^{m-1} D_i * 16^i$$

(Powers of 16 are listed in Table 1.1.)

Alternatively, the decimal equivalent can be written as

$$M = (\ldots((D_{m-1} * 16 + D_{m-2}) * 16 + D_{m-3}) * 16 + \ldots + D_1) * 16 + D_0$$

The following examples illustrate how these two forms can be used to compute the decimal equivalent of a hexadecimal number.

Example (Sum-of-powers method)

$$N = 3107_{16}$$

$$M = 7*16^0 + 0*16^1 + 1*16^2 + 3*16^3$$

$$= 7*1 + 0*16 + 1*256 + 3*4096$$

$$= 7 + 0 + 256 + 12288$$

$$= 12551_{10}$$

Example (Multiply-and-add method)

$$N = 3107_{16}$$

$$3*16 + 1 = 48 + 1 = 49$$

$$49*16 + 0 = 784 + 0 = 784$$

$$784*16 + 7 = 12544 + 7 = 12551$$

$$M = 12551_{10}$$

1.5 HEXADECIMAL-TO-BINARY CONVERSION

The binary equivalent M of the hexadecimal number N is obtained by replacing each digit in N with a group of four binary digits as shown in Table 1.1. (Leading 0s in the leftmost group can be deleted.)

Example

$$N = 3D6C_{16}$$

$$M = 11\ 1101\ 0110\ 1100_2$$

TABLE 1.1 POWERS OF 2 AND 16

$16^0 = 2^0 = 1$	$16^2 = 2^8 = 256$
$2^1 = 2$	$2^9 = 512$
$2^2 = 4$	$2^{10} = 1024$
$2^3 = 8$	$2^{11} = 2048$
$16^1 = 2^4 = 16$	$16^3 = 2^{12} = 4096$
$2^5 = 32$	$2^{13} = 8192$
$2^6 = 64$	$2^{14} = 16384$
$2^7 = 128$	$2^{15} = 32768$

1.6 BINARY-TO-HEXADECIMAL CONVERSION

The hex equivalent M of the binary number N is obtained by partitioning N into four-digit groups from right to left and then replacing each group by a hex digit as shown in Table 1.2. (If the length of N is not divisible by 4, add enough leading 0s to make it so.)

TABLE 1.2 DECIMAL-HEXADECIMAL-BINARY CONVERSION

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Example

$N = 1101011010001111 = 1101\ 0110\ 1000\ 1111$

$M = D68F_{16}$

1.7 BINARY AND HEXADECIMAL ADDITION

Binary numbers are added as if they were decimal, except for the following rules:

$$1 + 1 = 0 \text{ (carry 1)}$$

$$1 + 1 + 1 = 1 \text{ (carry 1)}$$

Example

$$\begin{array}{r} 101101_2 \\ +\ 100111_2 \\ \hline 1010100_2 \end{array}$$

The most straightforward method for adding two hex numbers is to add as if they were decimal, except for the following rules:

1. If a digit is a letter (A to F), convert it to decimal.

2. If the sum, S, of two digits (added as if they were decimal) is 16 or more, then replace S with S-16 and carry 1 (e.g., 9 + 9 = 2 carry 1).
3. If the sum of two digits (added as if they were decimal) is 10 to 15, then replace it with the appropriate letter, A through F (e.g., $5 + 7 = 12_{10} = C_{16}$).

Example

$$\begin{array}{r} 7A34D_{16} \\ +\ B2C36_{16} \\ \hline 12CF83_{16} \end{array}$$

EXERCISES

1.1. Using both the subtraction-of-powers and the division method, convert the following decimal numbers into their binary and hex equivalents.
- a. 2337_{10}
- b. 10000_{10}
- c. 16383_{10}

1.2. Convert the following binary numbers into their decimal and hex equivalents.
- a. 1111111_2
- b. 10000001_2
- c. 1010011100101110_2

1.3. Convert the following hex numbers into their decimal and binary equivalents.
- a. $8FF_{16}$
- b. $9A8_{16}$
- c. $CAFE_{16}$

1.4. Perform the following binary additions and check the results by converting the numbers into decimal.
- a. $1011010_2 + 11010_2$
- b. $1111101_2 + 1110_2$

1.5. Perform the following hex additions and check the results by converting the numbers into decimal.
- a. $16A_{16} + FFF_{16}$
- b. $18F_{16} + 2EC_{16}$

1.6. Compute the following binary product and check the result by converting the numbers into decimal.

$$11010111_2 * 100101_2$$

1.7. Show that the binary equivalent of $2^k - 1$ is 111 . . . 1 (k times).

2

THE ORGANIZATION OF THE 68000

A Motorola 68000-based computer system consists of a Motorola 68000 *central processor* (CP), where all computations take place, the *central memory* (CM), where the data and program are stored; and *peripheral devices*, such as the terminal, clock, and the devices needed to communicate between the 68000 and the terminal. In this text we shall assume a very rudimentary 68000 configuration with only a terminal, a clock, and a communication device as peripheral devices. Details on these and other devices can be found in the manuals for the particular devices.

Strictly speaking, "68000" refers to the microprocessor chip itself (consisting of the CP only). However, for the sake of brevity, in the remainder of this text the designation 68000 will refer to the entire system—CP, CM, and peripherals—and not just the chip.

Figure 2.1 outlines the general structure of the 68000 CP along with central memory and peripheral devices. In this chapter we shall describe the main features of the various components shown in this figure with the exception of the clock control and status registers. These registers, together with the interrupt feature of the communication device, will be discussed in Chapter 8.

A symbol by itself, for example X, will represent a CM address or a register name. The notation (X) will stand for "the contents of X". Similarly, ((X)) will stand for the "contents of (X)" (i.e., the contents of the memory location whose address is found in X). For the sake of readability, the name of a register will sometimes be used to indicate its contents when no other meaning is possible. Thus, D5 and (D5) will both be used to indicate the contents of register D5. For example, "add D5 to D3" and "add (D5) to (D3)" will both be used and will mean the same thing. The latter is perhaps more technically accurate, but the former is easier to read.

2.1 THE CENTRAL MEMORY

The central memory of a computer is also called "main memory", "primary memory", "RAM" (random access memory), and "core" (a term left over from the days when most central memories were constructed from magnetic core elements).

The basic memory element of a computer is the *bit*, which is capable of holding