



VAX

STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING

SECOND EDITION

ROBERT W. SEBESTA

VAX

STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING

Second Edition

Robert W. Sebesta

University of Colorado
at Colorado Springs



The Benjamin/Cummings Publishing Company, Inc.

Redwood City, California • Menlo Park, California
Reading, Massachusetts • New York • Don Mills, Ontario
Wokingham, U.K. • Amsterdam • Bonn • Sydney
Singapore • Tokyo • Madrid • San Juan

To Joanne

Sponsoring Editor: John Thompson
Production Coordinator: Eleanor Renner Brown
Cover and Chapter Opener Designer: Eleanor Mennick
Electronic Composition and Art: Ocean View Technical Publications

VAX is a registered trademark of the Digital Equipment Corporation

Copyright © 1991 by The Benjamin/Cummings Publishing Company, Inc.
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 written permission of the publisher.
Printed in the United States of America. Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data

Sebesta, Robert W.

VAX : structured assembly language programming / Robert W.
Sebesta. — 2nd ed.

p. cm.

Rev. ed. of VAX II. © 1984.

Includes index.

ISBN 0-8053-7122-2

1. VAX-11 (Computer)—Programming. 2. Assembler language
(Computer program language) 3. Structured programming.

I. Sebesta, Robert W. VAX 11 II. Title.

QA76.8.V37S4 1991

005.2'45—dc20

90-26190

CIP

ISBN 0-8053-7122-2

2 3 4 5 6 7 8 9 10 HA 95 94 93 92 91



The Benjamin/Cummings Publishing Company, Inc.
390 Bridge Parkway
Redwood City, California 94065

VAX

STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING

Second Edition

THE BENJAMIN/CUMMINGS SERIES IN COMPUTER SCIENCE

- G. Booch,
Object-Oriented Design with Applications (1989)
- G. Brookshear
Computer Science: An Overview, Third Edition (1991)
- F. Carrano
Assembler Language Programming for the IBM 370 (1988)
- D. M. Etter
Structured FORTRAN 77 for Engineers and Scientists, Third Edition (1990)
- P. Helman, R. Veroff, and F. Carrano
Intermediate Problem Solving and Data Structures: Walls and Mirrors, Second Edition (1991)
- P. Helman and R. Veroff
Walls and Mirrors: Intermediate Problem Solving and Data Structures—Modula II (1988)
- N. Miller and C. G. Peterson
File Structures With Ada (1990)
- A. Kelley and I. Pohl
A Book on C: An Introduction to Programming in C, Second Edition (1990)
- A. Kelley and I. Pohl
C by Dissection: The Essentials of C Programming (1988)
- I. Pohl
C++ for C Programmers
- I. Pohl
C++ for Pascal Programmers
- I. Pohl
Turbo C++ (1991)
- W. J. Savitch
Pascal: An Introduction to the Art and Science of Programming, Third Edition (1990)
- W. J. Savitch
Turbo Pascal 4.0/5.0 with 5.5 Supplement (1990)
- R. Sebesta
VAX Structured Assembly Language Programming, Second Edition (1991)
- R. Sebesta
Concepts of Programming Languages (1989)
- M. Thorne
Computer Organization and Assembly Language Programming for IBM PCs and Compatibles (1991)
- Titles of Related Interest**
- M. Sobell
A Practical Guide to the UNIX System, Second Edition (1989)
- M. Sobell
A Practical Guide to UNIX System V, Version 4 (1991)

PREFACE

This book is intended to be used for the standard course on assembly language programming, with the assumption that the reader has had at least one course in high-level language programming.

Learning an assembly language has two related yet distinct benefits: the acquisition of the skill of producing software in assembly language and gaining an in-depth knowledge of the architecture of a particular computer.

In this book the VAX series of superminicomputers is presented as a vehicle for discussing these two topics, because we believe that the VAX is particularly suited to the study of assembly language. First, the VAX architecture represents that of many other computers. The VAX instruction set includes many of the features of the instruction sets of other computers. As a result, the assembly languages of most other computers are easy to learn for those who know VAX assembly language. The second advantage of studying VAX assembly language lies in the complexity of the VAX architecture. The VAX has a rich set of instructions and addressing modes that make programming it in assembly language far easier than on simpler computers. The obvious price of this higher degree of writeability is the increased difficulty in learning the assembly language. However, the orthogonality of the VAX instruction set goes a long way toward balancing the increase in complexity with a measure of elegant simplicity.

Our approach to software development in assembly language is as follows. A pseudocode solution to a given problem is created first. This pseudocode is then translated, using consistent techniques, to an assembly language program. The pseudocode is included as comments in the final program, providing documentation. This program design methodology is described in detail and used in all example programs. This method, which is clearly a top-down approach, results in well-structured programs that are reasonably easy to read, while sacrificing very little in efficiency.

Macroinstructions are used for terminal input and output. This allows the reader to write complete programs very early in the course. Getting such an early start is important, because we believe that a significant amount of programming experience is necessary to acquire the expertise to be an effective assembly language programmer.

Features of the Book

Program Examples

This book contains a large number of program examples, many of which are complete stand-alone programs. Most of the complete examples are preceded by pseudocode algorithms that describe their actions.

The VAX/VMS Debugger

There is an entire chapter devoted to the VAX/VMS Debugger. This chapter describes the most useful debugger commands, including screen mode use of the debugger. The complete text of a debugging session is also included.

Input/Output Macroinstructions

Macroinstructions for terminal input and output of three integer type values and character strings, and also output for floating-point values, are used for the example programs. The software package that provides these services also includes a macro for dumping registers and memory. The method of obtaining this software is described below.

Student Aids

Every chapter except the first has a summary, lists of new terms and new instructions, and a problem set. Nearly all of the problem sets have both programming and nonprogramming exercises. We have included a description of all of the nonprivileged VAX instructions. Instructors who feel that the entire instruction set cannot be effectively covered in a course can choose sections to exclude according to their tastes.

The VAX Record Management System

The book has a complete chapter on RMS, the VMS subsystem that the VAX/VMS high-level languages use for input and output.

An Alternative Introduction

Appendix A contains a chapter-length introduction to computer organization, machine language, and assembly language programming. This appendix uses a very simple and idealized computer named SIMCOM. The SIMCOM approach to introducing the important concepts of this book is intended to be used for classes in which the students have relatively weak backgrounds in computing (for example, those who have not had a course in computer organization and have never learned anything about instruction sets and low-level programming).

Instructional Software

Both the SIMCOM simulator and the VAX input/output package are available through an anonymous FTP account from node HAPPY.UCCS.COLO-RADO.EDU. The user id is ANONYMOUS and the password is GUEST. All of the required files are in the directory named MACRO. For further information about software please contact the author at the University of Colorado at Colorado Springs or your local Benjamin/Cummings representative.

Using the Book in the Classroom

Chapter 1 contains background information on the evolution of computer architecture and computer languages, along with some justification for learning any assembly language, and VAX assembly language in particular. Chapter 2 covers the necessary material on binary and hexadecimal numbers, the addition and subtraction operations on binary and hexadecimal numbers, number base conversions, and twos complement notation. Chapters 1 and 2 may be skipped by classes whose students have some background in computer organization and binary and hexadecimal arithmetic.

Most of the essential material of the book appears in Chapters 3-11. Chapter 3 first describes general computer architecture and CPU operation, and then introduces the VAX architecture and a small collection of VAX instructions and directives. The process of writing and running complete programs is also described.

Chapter 4 introduces VAX assembly language implementations of the fundamental program control constructs. The methodology uses standard techniques of implementing pseudocode versions of selection and looping structures.

Chapter 5 is a description of the VAX/VMS debugger, including its use in screen mode. A complete debugging session on an example program is a significant part of the chapter.

Chapter 6 introduces the other VAX integer data types, operand expressions, and simple macros. Chapter 7 discusses the use of indexing for array processing and the VAX implementation of indexing. Chapter 8 introduces the concept of indirect addressing and covers the VAX addressing modes that implement it.

Chapter 9 describes the character manipulation instructions of the VAX and how they can be used for programming solutions to simple problems.

Chapter 10 is a thorough discussion of VAX assembly language subprograms, including the various parameter passing methods, recursion, and subprogram libraries. Chapter 11 covers the remaining features of macros, along with all of the VAX techniques for conditional assembly.

Chapter 12 describes VAX facilities for dealing with floating-point and decimal data. Bit and logic instructions and their applications are discussed in Chapter 13. Chapter 14 briefly describes the fundamental features of the RMS input/output system of VAX/VMS.

Acknowledgments

Textbooks are created over a long period of time by a large number of people. Some of the people who were involved in the development of this book deserve mention here.

The first edition of this book was reviewed by Theodore Bashkow of Columbia University, M. Faiman of the University of Illinois, George Rice of De Anza College, Arthur Gill of the University of California at Berkeley, Henry Leitner of Harvard University (Aiken Computation Laboratory), and Robert Muller of Boston University. Richie L. Lary of Digital performed a technical review of the first edition.

The second edition was reviewed by John Sheehan of San Francisco State University and Myers L. Foreman of Lamar University.

Alan Apt, former editor at Benjamin/Cummings, encouraged and advised me throughout the efforts to produce the two editions of this book.

The UCCS Computer Center provided the use of their VAX systems for the development of the input/output package and the example programs that appear in this book.

Finally, I thank my wife Joanne for her patience during the seemingly endless hours it has taken me to write the two editions of this book.

ROBERT W. SEBESTA

BRIEF CONTENTS

1	Introduction	1
2	Nondecimal Numbers and Arithmetic	13
3	Introduction to Computer Architecture and Assembly Language	37
4	Repetition and Selection Structures	77
5	Using the VAX/VMS Debugger	107
6	Integer Data Types, Operand Expressions, and Simple Macros	135
7	Arrays and Indexing	167
8	Indirect Addressing	189
9	Character Manipulation	225
10	Subprograms	251
11	Macros and Conditional Assembly	287
12	Floating-Point and Decimal Instructions	311
13	Bit and Logic Operations	341
14	VAX Input/Output	367
15	Additional Features	385
	Appendix A A Simple Computer	405
	Appendix B HEX/Decimal Conversion Table	437
	Appendix C ASCII Codes	439
	Appendix D Answers to Selected Problems	441
	Appendix E Using the TPU Editor	447
	Appendix F VAX Instruction Summary	455
	Index	475

CONTENTS

1	Introduction	1
1.1	The Evolution of Contemporary Computer Architecture	2
1.2	The Evolution of Computer Languages	4
1.3	Reasons to Study Assembly-Language Programming	9
1.4	The VAX Family of Computers	10
1.5	Why Study the VAX?	11
2	Nondecimal Numbers and Arithmetic	13
2.1	Positional Number Systems	14
2.2	Binary and Hexadecimal Numbers	14
2.3	Addition and Subtraction	17
2.3.1	Addition	17
2.3.2	Subtraction	20
2.4	Conversions Between Number Bases	22
2.4.1	Conversion to Decimal Numbers	23
2.4.2	Conversion Between Binary and Hex Numbers	24
2.4.3	Conversion to Nondecimal Numbers	26
2.4.4	Conversion of Fractions	27
2.5	Twos Complement Notation	29
3	Introduction to Computer Architecture and Assembly Language	37
3.1	General Computer Architecture	38
3.1.1	Introduction	38
3.1.2	Main Memory	38
3.1.3	Byte-Forward and Byte-Backward Representations	41
3.1.4	Central Processing Unit	43
3.1.5	Input and Output Devices	43
3.1.6	Machine Instruction Formats	44
3.2	CPU Operation—The Fetch-Execute Cycle	45

3.3	Introduction to VAX Architecture	47
3.4	LONGWORD Integer Instructions	50
3.4.1	Symbols and Storage Allocation	51
3.4.2	Simple Machine Instruction Formats	53
3.4.3	Data Move Instructions	56
3.4.4	Arithmetic Instructions	57
3.4.5	An Example Program Segment	59
3.4.6	Constant Operands	60
3.4.7	Input and Output	61
3.5	Running VAX Assembly-Language Programs	62
3.5.1	Required Assembler Directives	63
3.5.2	Program Creation, Assembly, Linking, and Execution	65
3.6	Runtime Errors	70
4	Repetition and Selection Structures	77
4.1	Branch Instructions	78
4.1.1	Unconditional Branch Instructions	78
4.1.2	Conditional Branch Instructions	79
4.2	Pretest Logical Loops	83
4.3	Selection Structures	86
4.4	Counter-Controlled Loops	89
4.5	More Loop Instructions	95
4.6	Compound Conditions	97
4.7	The Assembly Process	99
4.8	More Directives	101
5	Using the VAX/VMS Debugger	107
5.1	What Is a Debugger?	107
5.2	Breakpoints, Tracepoints, and Watchpoints	108
5.2.1	Address Points	108
5.2.1	Breakpoints	109
5.2.3	Tracepoints	111
5.2.4	Watchpoints	113
5.3	Running Programs with the Debugger	114
5.4	EXAMINE and DEPOSIT Commands	116

5.5	Getting Into and Out of the Debugger	119
5.5.1	Changing Parameter Default Values	120
5.5.2	Breakpoint DO Options	121
5.6	A Sample Debugging Session	122
5.7	Using the Debugger in Screen Mode	127
5.7.1	Display Configurations	127
5.7.2	Using the Keypad	129
5.8	The DUMP Instruction	131
6	Integer Data Types, Operand Expressions, and Simple Macros	135
6.1	The Other Integer Data Types	136
6.1.1	Integer Data Types	136
6.1.2	Directives	137
6.1.3	Move and Arithmetic Instructions	138
6.1.4	Test, Compare, and Loop Instructions	143
6.1.5	Size Conversion Instructions	144
6.1.6	The Debugger and Non-LONGWORD Integers	145
6.1.7	Input and Output of Non-LONGWORD Integers	145
6.2	Operand Expressions	146
6.3	Constants of Nondecimal Bases	148
6.4	Direct Assignment Statements	148
6.5	A Sample Program	150
6.6	The Overflow and Carry Indicators	152
6.7	Simple Macros	154
6.1.7	The Macro Concept	154
6.7.2	Parameterless Macros	155
6.7.3	Passing Parameters to Macros	156
6.7.4	Macro Listing Control	159
7	Arrays and Indexing	167
7.1	The Need for Arrays	167
7.2	The Concept of Indexing	169
7.3	Indexing on the VAX	170
7.3.1	VAX Assembly-Language Indexing Syntax	171
7.3.2	Indexing Operation	173

7.4	The INDEX Instruction	181
7.5	Matrices	182
8	Indirect Addressing	189
8.1	The Concept of Indirect Addressing	190
8.2	Register-deferred Addressing Mode	190
8.3	Autoincrement and Autodecrement Addressing Modes	193
8.4	Displacement Mode	203
8.5	Relative-deferred Mode	207
8.6	Two Levels of Indirectness	211
8.7	Summary of Effective Address Computation	215
8.8	Relative Speeds of Some Addressing Modes	217
8.9	Controlling the Sizes of Displacements	218
8.10	Indirect Addressing and the Debugger	219
9	Character Manipulation	225
9.1	Character Codes and Character Data	225
9.2	Character Input/Output	230
9.3	Character Manipulation	231
9.3.1	Sorting Character Data	231
9.3.2	Finding Words	238
9.3.3	Finding Specific Substrings	245
10	Subprograms	251
10.1	Stack Operations	252
10.2	Simple Subprograms	257
10.3	Passing Parameters with a General Argument List	262
10.4	Passing Parameters in the Stack	269
10.5	Recursive Procedures	271
10.6	Subprogram Libraries	278
10.7	Subprograms and the Debugger	279
11	Macros and Conditional Assembly	287
11.1	Macro Parameters	288
11.1.1	Default Parameters	288
11.1.2	Keyword Parameters	289
11.1.3	String Parameters	290
11.1.4	Catenation of Parameters	290

11.2	Unique Symbol Generation	291
11.3	String Operations in Macros	294
11.4	Macro Libraries	295
11.5	Conditional Assembly	297
11.5.1	Repeat Loops	297
11.5.2	Values of Symbols of Parameters	298
11.5.3	List-directed Repeat Loops	299
11.5.4	Assembly-Time Selection Structures	301
11.6	More Directives	304
11.7	Recursive Macros	306
12	Floating-Point and Decimal Instructions	311
12.1	Single-Precision Floating-Point Data Type	312
12.1.1	Single-Precision Floating-Point Notation	312
12.1.2	Floating-Point Constant Notation	314
12.1.3	Floating-Point Non-Arithmetic Operations	316
12.1.4	Floating-Point Arithmetic Instructions	317
12.2	Additional Floating-Point Data Types	322
12.3	Errors with Floating-Point Operations	324
12.4	Dealing with Decimal Data	326
12.4.1	Decimal Data Formats	326
12.4.2	Instructions for Decimal Data	329
12.4.3	Conversion Instructions for Decimal Data	332
13	Bit and Logic Operations	341
13.1	Bit String Data	342
13.2	Branch Instructions for Bit Strings	342
13.3	Bit String Searching	345
13.4	Manipulating Bit Strings	347
13.5	Shift Instructions	351
13.6	Logic Operations	354
13.7	Unsigned Branch Instructions	361
14	VAX Input/Output	367
14.1	Overview of RMS	367
14.2	Control Blocks	369
14.2.1	Allocation of Space for Control Blocks	370
14.2.2	File-level Operations	372

14.2.3	Record-level Operations	372
14.3	Example Programs	374
14.4	Terminal Input/Output	379
15	Additional Features	385
15.1	Program Sections	386
15.2	Queues	388
15.3	Local Labels	393
15.4	Multiple-Selection Structures	394
15.5	Multiple-Precision Integer Arithmetic	395
15.6	Character Code Translation	397
15.7	The EDITPC Instruction	399
15.8	Memory Interlock	399
15.9	Odds and Ends	400
Appendix A	A Simple Computer	405
Appendix B	HEX/Decimal Conversion Table	437
Appendix C	ASCII Codes	439
Appendix D	Answers to Selected Problems	441
Appendix E	Using the TPU Editor	447
Appendix F	VAX Instruction Summary	455
Index		475



1

INTRODUCTION

- 1.1 *The Evolution of Contemporary Computer Architecture*
- 1.2 *The Evolution of Computer Languages*
- 1.3 *Reasons to Study Assembly-Language Programming*
- 1.4 *The VAX Family of Computers*
- 1.5 *Why Study the VAX?*

There are a few important preliminaries to a serious study of assembly-language programming. The first two chapters and part of the third chapter of this book cover the most important of these. Chapters 1 and 3 include discussions of the fundamentals of the architecture of digital computers and their operation. Chapter 2 covers the required information on nondecimal numbers and the methods of storing integer data in a computer's memory.

In this chapter we discuss the fundamentals of computers in rather general terms. Included is a brief historical introduction to the development of the hardware and architecture of contemporary computers, and also a brief discussion of the early history of programming languages. With this background we can put assembly language and the VAX family of computers in perspective. This chapter also includes some reasons for studying assembly-language programming in general, and the VAX architecture and assembly language in particular.