DIGITAL RESEARCH™

Programmer's Utilities Guide for the IBM® Personal Computer Disk Operating System

Digital Research
P.O. Box 579
160 Central Avenue
Pacific Grove, CA 93950
(408) 649-3896
TWX 910 360 5001

The Programmer's Utilities Guide for the IBM Personal Computer Disk Operating System was prepared using the Digital Research TEX Text Formatter and printed in the United States of America.

****
* First Edition: April 1983 *
****

Foreword

This manual describes several utility programs that aid programmers and system designers in the software development process. Collectively, these utilities allow you to assemble 8086 assembly language modules, link them together to form a program that runs, and generate a cross-reference map of the variables used in a program. You can also use these utilities to create and manage your own libraries of subroutines and program modules, as well as create large programs by breaking them into separate overlays.

The Programmer's Utilities Guide assumes that you are familiar with the DOS operating system environment. It also assumes that you are familiar with the basic elements of 8086 assembly language programming.

RASM-86™ is an assembler that translates 8086 assembly language statements into a relocatable object file in the Intel® format. RASM-86 facilities include assembly of Intel 8086 mnemonics, assembly-time expressions, conditional assembly, page formatting of listing files, and powerful code-macro capabilities.

Section 1 describes the overall operation of RASM-86 and its optional run-time parameters. Section 2 describes elements of RASM-86 assembly language, including the character set, delimiters, constants, identifiers, operators, expressions, and statements.

Section 3 describes the various RASM-86 directives that control the assembly process. Section 4 contains a brief description of the RASM-86 instructions for data transfer, mathematical operations, string manipulation, control transfer, and processor control. Section 5 describes the code-macro facilities of RASM-86.

Section 6 describes XREF-86™ , an assembly language cross-reference program used with RASM-86. Section 7 describes LINK-86™ , the linkage editor that combines relocatable object modules into an absolute file that runs under DOS. Section 8 explains how to use LIB-86™ , the software librarian that creates and manages libraries.

The appendixes contain a complete list of error messages output by each of the utility programs.

Table of Contents

1 Introduction to RASM-86

1.1 Assembler Operation 1-1

1.2 Invoking RASM-86 1-2

1.3 Optional Run-time Parameters 1-3

2 Elements of RASM-86 Assembly Language

2.1 RASM-86 Character Set 2-1

2.2 Tokens and Separators 2-1

2.3 Delimiters 2-1

2.4 Constants 2-3

2.4.1 Numeric Constants 2-3
2.4.2 Character Strings 2-3

2.5 Identifiers 2-4

2.5.1 Keywords 2-5
2.5.2 Symbols and Their Attributes 2-6

2.6 Operators 2-7

2.6.1 Operator Examples 2-11
2.6.2 Operator Precedence 2-13

2.7 Expressions 2-14

2.8 Statements 2-15

3 Assembler Directives

3.1 Segments 3-2

3.2 The Segment Directive 3-2

3.2.1 <segment name> 3-3
3.2.2 <align type> 3-3
3.2.3 <combine type> 3-4

3.3 The GROUP Directive 3-5

3.4 The ORG Directive 3-5

Table of Contents
(continued)

Table of Contents
(continued)

Table of Contents
(continued)

Table of Contents
(continued)

Appendixes

Tables, Figures, and Listings

Tables

Tables, Figures, and Listings (continued)

Figures

Listing

Section 1
Introduction to RASM-86

1.1 Assembler Operation

RASM-86 processes an 8086 assembly language source file in three passes and produces an 8086 machine language object file. RASM-86 can optionally produce three output files from one source file as shown in Figure 1-1.

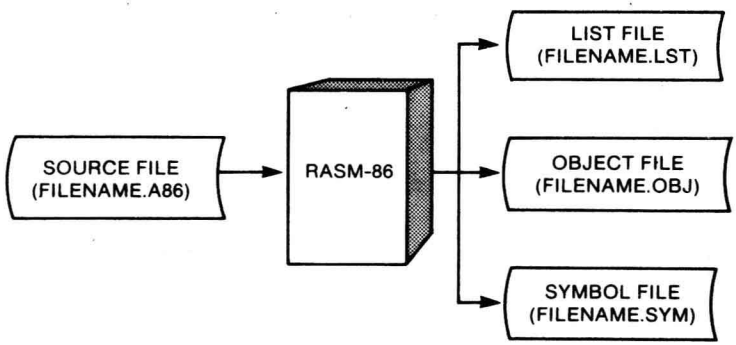

Figure 1-1. RASM-86 Source and Object Files

The LST list file contains the assembly language listing with any error messages. The OBJ object file contains the object code in Intel 8086 relocatable object format. The SYM symbol file lists any user-defined symbols.

The three files have the same filename as the source file. For example, if the name of the source file is BIOS88.A86, RASM-86 produces the files BIOS88.OBJ, BIOS88.LST, and BIOS88.SYM.

1.2 Invoking RASM-86

Invoke RASM-86 with a command in the form:

```
RASM86 source file ($ optional parameters)
```

The filespec has the form:

```
[d:]filename[.typ]
```

where

d:	is an optional drive specification denoting the source file's location. The drive specification is not needed if the source is on current drive.
filename	is a valid filename of 1 to 8 characters.
typ	is a valid filetype of 1 to 3 characters, usually A86.

RASM-86 accepts a source file with any filetype. If you omit the filetype from the command line, RASM-86 searches the directory for the specified filename with the filetype A86.

The following are some examples of valid RASM-86 commands:

```
A>rasm86 b:bios88

A>rasm86 bios88.a86  $ aa ob pb sb

A>rasm86 d:test
```

Once invoked, RASM-86 responds with the message:

```
-------------------------------------------------
RASM-86 Relocating Assembler           Version X.X
Serial No. xxxx-0000-654321    All Rights Reserved
Copyright (C) 1982,1983     Digital Research, Inc.
-------------------------------------------------
```

RASM-86 then attempts to open the source file. If the file does not exist on the designated drive or does not have the correct filetype, RASM-86 displays the message:

```
NO FILE
```

and stops processing.

By default, RASM-86 creates the output files on the currently logged-in disk drive. However, you can redirect the output files by using the optional parameters, or by a drive specification in the source filename. In the latter case, RASM-86 directs the output files to the drive specified in the source filename.

When the assembly is complete, RASM-86 displays the message:

```
END OF ASSEMBLY. NUMBER OF ERRORS: n  USE FACTOR: pp%
```

The Use Factor indicates how much of the available Symbol Table space was actually used during the assembly. The Use Factor is expressed as a decimal percentage ranging from 0 to 99.

1.3 Optional Run-time Parameters

The dollar sign character, $, denotes an optional string of run-time parameters. A parameter is a single-letter followed by a single-letter device name specification. The parameters are shown in Table 1-1.

Table 1-1. RASM-86 Run-time Parameters

Parameter	Specifies	Valid Arguments
A	Source file device	A, B, C, ... P
L	Local symbols in object file	O
O	Object file device	A ... P, Z
P	List file device	A ... P, X, Y, Z
S	Symbol file device	A ... P, X, Y, Z

All the parameters are optional, and you can enter them in the command line in any order. Enter the dollar sign only once at the beginning of the parameter string. Spaces can separate parameters, but are not required. However, no space is permitted between a parameter and its device name.

If you specify an invalid parameter in the parameter list, RASM-86 displays the message:

```
SYNTAX ERROR
```

RASM-86 then echoes the command tail up to the point where the error occurs and follows with a question mark. (Appendix F contains the complete list of RASM-86 error messages.)

A device name must follow the parameters A, O, P, and S. The devices are labeled as follows:

```
A, B, C, ... P  or  X, Y, Z
```

Device names A through P specify disk drives A through P, respectively. X specifies the user console, Y specifies the list device, and Z suppresses output.

If you direct the output to the console, you can temporarily stop the display at any time by typing a CTRL-S, and then restart it by typing CTRL-Q.

The LO parameter directs RASM-86 to include local symbols in the object file so that they appear in the SYM file created by LINK-86. Otherwise, only public symbols appear in the SYM file. You can use the SYM file with a symbolic instruction debugger to simplify program debugging.

Table 1-2. RASM-86 Command Line Examples

Command Line	Result
rasm86 io	Assembles file IO.A86 and produces IO.OBJ, IO.LST, and IO.SYM, all on the default drive.
rasm86 io.asm $ ad sz	Assembles file IO.ASM on drive D and produces IO.LST and IO.OBJ. Suppresses the symbol file.
rasm86 io $ py sx	Assembles file IO.A86, produces IO.OBJ, and sends listing directly to printer. Also outputs symbols on console.
rasm86 io $ lo	Includes local symbols in IO.OBJ.

End of Section 1

Section 2
Elements of RASM-86 Assembly Language

2.1 RASM-86 Character Set

RASM-86 recognizes a subset of the ASCII character set. The valid characters are the alphanumerics, special characters, and nonprinting characters shown below:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9

+ - * / = ( ) [ ] ; ' . ! , _ : @ $ ?

space, tab, carriage return, and line-feed
```

RASM-86 treats lower-case letters as upper-case except within strings. Only alphanumerics, special characters, and spaces can appear in a string.

2.2 Tokens and Separators

A token is the smallest meaningful unit of a RASM-86 source program, much as a word is the smallest meaningful unit of a sentence. Adjacent tokens within the source are commonly separated by a blank character or space. Any sequence of spaces can appear wherever a single space is allowed. RASM-86 recognizes horizontal tabs as separators and interprets them as spaces. RASM-86 expands tabs to spaces in the list file. The tab stops are at each eighth column.

2.3 Delimiters

Delimiters mark the end of a token and add special meaning to the instruction; separators merely mark the end of a token. When a delimiter is present, separators need not be used. However, using separators after delimiters can make your program easier to read.

Table 2-1 describes RASM-86 separators and delimiters. Some delimiters are also operators; these are explained in greater detail in Section 2.6.

Table 2-1. Separators and Delimiters

Character	Name	Use
20H	space	separator
09H	tab	legal in source files, expanded in list files
CR	carriage return	terminates source lines
LF	line-feed	legal after CR; if in source lines, it is interpreted as a space
;	semicolon	starts comment field
:	colon	identifies a label; used in segment override specification
.	period	forms variables from numbers
$	dollar sign	notation for present value of location counter; legal, but ignored in identifiers or numbers
+	plus	arithmetic operator for addition
-	minus	arithmetic operator for subtraction
*	asterisk	arithmetic operator for multiplication
/	slash	arithmetic operator for division
@	at	legal in identifiers
_	underscore	legal in identifiers
!	exclamation point	logically terminates a statement, allowing multiple statements on a single source line
'	apostrophe	delimits string constants