The Little Man Computer

Documentation and User's Guide

Version 3.5
Prepared by Professor Susan A. Riedel, January 6, 1994
Revised by Professor Jeffrey L. Hock, August 30, 1994
Put in html format by Professor Richard J. Povinelli, August 20, 1997

Table of Contents

Introduction
LMC Components
Start Button
Instruction Counter
Calculator
Paper and Pencil
Mailboxes
IN and OUT Baskets
Executing LMC Programs
LMC INSTRUCTION SET
A Sample LMC Program
The LMC Number System
The LMC directory files
The LMC Menu System
CREATE
LOAD
EDIT
ASSEMBLE
STORE
DISPLAY LMC CONTENTS
RUN LAlC PROGRAMS
DEBUG LMC PROGRAMS
HELP SCREENS
DOS
The Little Man Assembly Language
The Little Man Assembler
Summary
Acknowledgements

The Little Man Computer

Introduction

The Little Man Computer (LMC) is a simplified example of computer hardware and software which can be used to explain the fundamental principles of computer engineering. The LMC contains all of the components of modern computers: Memory, a Central Processing Unit (CPU), and input/output capability. A small but powerful programming language is used which allows the programmer to define a computation or operation for the Little Man to perform. By writing and executing

simple programs, the student is able to understand the function of each component of the hardware. Writing, debugging and executing LMC programs is supported by a color menu-driven software package written in the C programming language for the DOS environment. This paper describes the organization of the LMC, the machine language of the LMC, the LMC assembly language, the LMC Assembler, and the menu-driven user interface for the LMC.

LMC Components

Little Man: This is the heart of the LMC. The Little Man can perform a finite set of simple operations, one at a time. He performs any given operation using the following steps:

1. Read the number displayed in the Instruction Counter.
2. Go to the mailbox with that same number.
3. Look at the slip of paper in that mailbox, remember the number which appears on that paper, and put the slip of paper back into the same mailbox.
4. Push a button on the Instruction Counter which increments the number which is displayed.
5. Perform the operation designated by the number on the slip of paper.
6. Go back to step 1.

NOTE: The order of steps 4 and 5 are particularly important when considering the branching instructions. See page 4, instructions B, BZ, BP and the NOTE associated with instruction BP.

Start Button:	The operator of the LMC (not the Little Man) presses the START BUTTON to zero the Instruction Counter and ring a bell which wakes up the Little Man so he can start executing the program which as previously been stored in the mailboxes.
Instruction Counter:	This display contains the number of the mailbox in which the NEXT instruction is stored. The Little Man can change the number in the instruction counter (usually by incrementing but sometimes by replacement), and the user of the LMC can reset the Instruction Counter to 00 using the start button.
Calculator:	This calculator performs simple arithmetic (addition and subtraction only) and also can be used for temporary storage of a single three digit number. The display is limited to three digits. The calculator has ten numerical keys (0-9) and two operation keys (+ and -). The calculator also has two lights which can be seen by the Little Man. One of these lights (the one at the upper left) turns on whenever the number being displayed is exactly zero. The other light (upper right) turns on whenever the number being displayed is positive. It is important to note that the number ZERO is considered a positive number. Thus when ZERO is displayed on the calculator, both lights will be on. These lights are sometimes referred to as flags.
Paper and Pencil:	The Little Man is assumed to have an unlimited supply of paper and pencils for transferring data between different segments of the LMC.
Mailboxes:	Mailboxes are identified using two-digit numbers (00 99) and each can hold a single slip of paper containing a single three digit number.
IN and OUT Baskets:	The IN and OUT Baskets are used by the Little Man to communicate with the outside world, i.e. the LMC user. The IN basket can hold an unlimited number of slips of paper, each with a three-digit number written on it. The Little Man always takes the slip of paper that has been in the IN basket the longest, and this slip of paper can never be returned to the IN basket. The OUT basket can also hold an unlimited number of slips of paper, each with a three-digit number written on it. When the Little Man is instructed to place a slip of paper in the OUT basket, he always puts the paper at the bottom of the stack of paper in the OUT basket. These are called FIRST- IN/FIRST-OUT (FIFO) baskets.

Executing LMC Programs

1. To actually run an LMC program we must carry out the following steps.
2. Load the instructions into the mailboxes, starting with mailbox 00. (See the LMC INSTRUCTIONS given on the next page.)
3. Place the data to be used by the program in the IN basket, in the order in which the program will use these data.
4. Press the START BUTTON to set the Instruction Counter to 00 and to also wakeup the Little Man.
5. Wait for the result to appear in the OUT basket.

LMC INSTRUCTION SET

FORMAT	MNEMONIC	MEANING
000	STOP	Stops the Computer - the Little Man rests.
1xx	ADD xx	Adds the contents of mailbox xx to the calculator display.
2xx	SUB xx	Subtracts the contents of mailbox xx from the calculator display.
3xx	STO xx	Stores the calculator value into mailbox xx.
4xx	STA xx	Stores the address portion of the calculator value (last 2 digits) into the address portion of the instruction in mailbox xx.
5xx	LOAD xx	Loads the contents of mailbox xx into the calculator.
6xx	B xx	This instruction sets the instruction counter to the number xx, thus effectively branching to mailbox xx See the note for instruction BP
7xx	BZ xx	IF the calculator value is zero, THEN set the instruction counter to the number xx, thus effectively branching to mailbox xx. See the note for instruction BP
8xx	BP	IF the calculator value is positive, THEN set the instruction counter to the number xx, thus effectively branching to mailbox xx. NOTE: zero is considered positive. NOTE: Because of the three branching instructions, it is important for the LITTLE MAN to FIRST increment the instruction counter and THEN carry out the instruction. If he FIRST set the instruction counter to xx and THEN incremented the instruction counter the next instruction to be executed would be in mailbox xx+1, not in mailbox xx as was intended.
901	READ	Read a number from the IN basket and key it into the calculator.
902	PRINT	Copy the number in the calculator onto a slip of paper and place it into the OUT basket.

A Sample LMC Program

Suppose we wish to write a LMC program which adds two numbers together. In order to write such a program, we need to think about how to describe each simple step that the Little Man needs to perform to result in the addition of two numbers. These simple steps are described below:

1. The first step is to decide on where to place the two numbers that we want the Little Man to add. There are two places to store numbers in the LMC: in the mailboxes and/or in the IN basket. If we place the two numbers in two different mailboxes, the Little Man will only be able to add those two numbers each time we run the program - we won't be able to change the numbers without changing the slips of paper in the mailboxes (i.e. without having to rewrite the program). On the other hand, if we place the two numbers in the IN basket, and have the Little Man place the result of the addition in the OUT basket, then we have written a program that can add any two numbers. That is, each time we run the ADDITION program, we can place the two numbers to be added into the IN basket, and the sum will appear in the OUT basket. Using the IN basket is a much more flexible approach to addition than using the mailboxes to hold

the numbers.
2. Now we need to understand how the Little Man adds numbers. Notice that the ADD instruction, lxx, adds the contents of a mailbox (the one numbered xx) to the contents of the calculator. Thus, we need to move the first number which has been placed into the IN basket into a mailbox, and the other number in the IN basket into the calculator. Think about this. Why did we place the first number in a mailbox and the second in the calculator instead of placing the second number into a mailbox and leaving the first number in the calculator? Does any of this matter?
3. Once the addition has been completed, we can instruct the Little Man to place the result in the OUT basket by using the PRINT instruction.

Now, let's look at an actual LMC program which adds the two numbers presented in the IN basket and places the resulting summation in the OUT basket.

MAILBOX Number	Mailbox Contents (INSTRUCTION)	COMMENTS
00	901	Read the first number from the IN basket into the calculator.
01	306	Store the number from the calculator to mailbox 06. The first free mailbox after the end of this program.
02	901	Read the second number from the IN basket into the calculator.
03	106	Add the number in mailbox 06 to the number in the calculator.
04	902	Print the contents of the calculator onto a slip of paper and place it in the OUT basket.
05	000	Stop and let the Little Man rest.

An Example LMC Machine Language Program

Note that the first number in the IN basket was placed in the calculator and then moved into a mailbox. This took two steps because the Little Man does not know how to move numbers from the IN basket directly into a mailbox. Note further that we used mailbox 06 to hold this first number - why? Because we needed mailboxes 00-05 to hold the LMC instructions, so the first empty mailbox was 06. In fact, we could have used any of the empty mailboxes 06-99 to hold this number. Mentally step through this LMC program to make sure you understand how it works. For practice, you may wish to try and modify the above program to, subtract two numbers, add three numbers, multiply two numbers (this last problem being significantly more difficult).

The LMC Number System

The slips of paper that the Little Man uses internally can only hold a three digit number. There is no space on the paper for a minus sign (-). To accommodate negative numbers, a coding scheme, known as "10's (ten's) complement" is used. The positive numbers 0-499 are represented in their usual form. The negative numbers -1 through -500 are represented in 10's complement as 999 through 500. The 10's complement number is computed by adding the negative number to 1000 - that is, -14 would be represented as 986, since-14 + 1000 = 986.

It is not usually necessary for the LMC user to worry about the number system. Numbers can be represented in the usual signed form in the IN basket. The READ (901) instruction converts the number in the IN basket to 10's complement form. Similarly, numbers appearing in the OUT basket will appear in their signed form, since the PRINT (902) instruction converts from 10's complement to signed form. Numbers in the mailboxes must be in 10's complement form, however. This is important for the programmer to understand when trouble-shooting a program. It is also important for the LMC user to

remember that the range of numbers represented in the LMC is from -500 to 499. Numbers outside of this range, even if they are generated as intermediate results, will generate an overflow error and cause immediate termination of the LMC execution.

The LMC directory files

The following files are needed to run the Little Man Computer system:

LMC.EXE	This is the LMC, which can be run by typing c:\lmc\lmc at the DOS prompt on the computers in Haggerty Hall.
MASSEMB.EXE	This is the LMC Assembler, which can be run from the LMC Menu System (see below) by selecting the appropriate option from the CREATE submenu, or can be run directly from DOS by typing MASSEMB at the DOS prompt
TDE.EXE	s is the public domain editor which can be run from the LMC by selecting the appropriate option from the CREATE submenu, or can be run directly from DOS by typing TDE at the DOS prompt.
INSTR.TXT	This is an ASCII II which contains the LMC help files. It should not be modified.
MENU.TXT	This is an ASCII file which contains the MENU options used by the LMC program. It is read into the computer at the start of the LMC program execution and should not be modified.

The LMC Menu System

A color menu system is used to support the creation, execution, and debugging of LMC programs. This software assumes that you have an 80286 based (or better) PC with a VGA monitor and the DOS operating system. Using this menu system, the LMC programmer can enter Little Man machine language programs, edit them, run them, debug them line by line, and look at the contents of the mailboxes, the IN and OUT baskets, the calculator display, and the instruction counter. Little Man Assembly Language (LMAL) programs can also be created, assembled, run, and debugged from within the LMC menu system. There is even a simple on-line help system for beginning LMC users. The menu system has the following options:

MAIN MENU

1. CREATE an LMC program
2. DISPLAY the contents of the LMC
3. RUN the LMC program
4. DEBUG the LMC program
5. HELP Screens
6. DOS shell
7. Exit

Before the CREATE submenu appears, the programmer must know whether the program to be examined is in Little Man Machine Language or in Little Man Assembly Language. The CREATE submenu has the following options:

CREATE

1. LOAD an LMC program
2. EDIT an LMC program
3. ASSEMBLE an LMC program (only if LMAL is used)
4. STORE an LMC program
5. EXIT to the main menu

LOAD

The LOAD submenu permits the programmer to input the program from a disk file or line by line. This submenu also allows the data to be loaded into the IN basket.

INPUT PROGRAM FROM FILE:

If you chose option <1> "Input Program From File", you will be presented with a dialog box which asks: "Enter full pathname of file." If you keep your program in the LMC working directory, then only the file name and extension need be given. If the program cannot find the file you have specified, it will simply ask for the file name again (no error message is given). Also, there is no easy way out of the dialog box. If you cannot remember the program's name you may have to press CTRL- BREAK to terminate the program, and then restart the program by typing LMC.

INPUT PROGRAM FROM TERMINAL:

You may enter an LMC program directly from the terminal and later save it to a disk file. Once an LMC program has been saved, it can be reloaded using Option 1 from this submenu. Your program will then be displayed on the screen and you will be prompted for any changes you would like to make. If you enter your program directly from the terminal, you will be asked to follow the following format:

Columns 1-3	Mailbox Number. Type 014 for Mailbox 14.
Column 4	This column must be left blank
Columns 5-7	LMC Instruction. Type 901 for the READ instruction
Column 8	This column must be left blank
Columns 9-80	Comments, which will be ignored by the LMC but might be useful to anyone reading your program.

Program entry stops when you type 999 in columns 1-3. The program is then displayed and you have the opportunity to make changes.

When you have entered your LMC program, either from disk or at the terminal, and have made all necessary changes to the program, the following submenu appears:

1. Save LMC Program to Disk File
2. Load IN Baskets

Option 1 will allow you to enter a filename for saving you LMC program, which can then be reloaded another time. Once the program is saved, you are prompted to load the IN basket. You can choose not to save your LMC program by selecting Option 2.

EDIT

The EDIT function invokes the TDE public domain text editor which permits changes, insertions and deletions. Once the TDE editor is called upon, use the F1 key to display the help screen, the F2 key to save your edited file and the F3 key to exit the editor and return to the LMC. Follow the same column spacing as given under INPUT PROGRAM FROM TERMINAL above.

ASSEMBLE

The ASSEMBLE function runs the LMC Assembler, a program which translates Little Man Assembly Language (LMAL) into Little Man Machine Language. When called, the LMC Assembler asks you to "Enter the name of the input file without extension." The LMAL file must have the extension .ASM. When the Assembler runs, it produces a file containing the LMC machine language program with the same file name and the extension .LMC. To help in the debugging process, the Assembler also produces a file with the extension .DBG (debug) which contains both the LMAL program and its LMC translation, and a file with the extension .ERR containing any error messages generated during the assembling process. If the file specified could not be found an error message "open file failed, program aborted. Hit any key to return to LMC" appears. For additional information on assembly language programming see The Little Man Assembly Language.

STORE

The STORE function is used to write an LMC program out to disk. The full file name and extension must be given at the prompt.

DISPLAY LMC CONTENTS

This submenu has the following options which are used to examine the contents- of the LMC before or after running a LMC program:

1. Contents of OUT basket
2. Contents of IN basket
3. Contents of Mail Boxes (displays 20 mailboxes at a time)
4. Instruction Counter
5. Calculator Contents
6. Return to Main Menu

RUN LAlC PROGRAMS

This function runs the LMC program. If there were any errors encountered during the run, error messages will be displayed on the screen. Detectable errors include numeric overflow, illegal instructions, attempt to read from an empty IN basket, and attempt to access an empty mailbox. You can interrupt the running program by hitting any key on the keyboard.

DEBUG LMC PROGRAMS

When errors are encountered in a LMC program, the DEBUG option can often help to detect and correct those errors. The DEBUG option allows the programmer to run the LMC program line-by-line, stopping after each line to view the contents of the LMC. There are two DEBUG modes: Regular DEBUG starts the line-by-line execution at mailbox 00; quick DEBUG allows the programmer to input a mailbox number, executes the program to that mailbox and then starts the line by line execution of the program. The contents of the Instruction Counter, the current mailbox and the Calculator are automatically displayed. The DEBUG submenu, shown below, can be used to display other LMC contents.

1. Next Program Step
2. Additional Display

HELP SCREENS

This option provides the programmer with access to a simple on-line help facility. Help is available on all aspects of the LMC, including creating LMC programs, displaying the contents of the LMC, running LMC programs and debugging them. The HELP submenu is shown below:

1. Programming the LMC
2. LMC Instruction Set
3. Running Your Program
4. Displaying LMC Contents
5. Debugging Your Program

6. Return to Main Menu

DOS

This option gives the programmer access to all of the functions of the DOS environment without having to exit the LMC environment. For example, if the programmer created an LMC program and saved it to a file, it could then be printed out by exiting to DOS and running a PRINT program or performing a COPY to the printer (PRN or LPT1). When the printing was completed, the LMC environment could be re-entered by typing EXIT at the DOS prompt.

The Little Man Assembly Language

Although the Little Man is very comfortable remembering numeric instruction and numeric mailbox location, LMC programmers often have trouble with all of these numbers. To make programming easier, the LMC also supports a mnemonic form for programs, called Little Man Assembly Language (LMAL). In LMAL, mnemonic form is used for all instructions (see table on page 4). LMAL also supports the use of names, rather than numbers, for mailboxes. This is accomplished in two ways. First, the programmer can identify one or more mailbox names using a special LMAL instruction called DC (for "Define Constant"). These mailboxes are often used to store LMC program data. Second, the programmer can associate a label with any given LMAL instruction, thereby identifying the mailbox holding that instruction for future reference.

As an example, let's extend the previous program to add many numbers. Suppose we wish to write a program to add all numbers in the IN baskets, until we encounter a zero. When the 0 is read, the sum of all the previous numbers will be printed out. Examine the following LMAL program: (This can be generated using any ASCII editor such as TDE which is accessible under option <2> "Edit an LMC program" of the CREATE submenu of the LMC.)

filename.ASM

Label	**Instruction**	**Comment**
.LMC		;A directive for the assembler
.LOOP	READ	;Read in the next number
	BZ PRINTOUT	;If it's a zero, jump to printing
	ADD SUM	;If it's not a zero, add to sum
	STO SUM	;Remember to store the running sum
	B LOOP	;Go back to the beginning
.PRINTOUT	LOAD SUM	;Place the sum in the calculator
	PRINT	;Print the calculator display
	STOP	;Give the Little Man a break
.DC	SUM 000	;Reserve a mailbox for SUM and initialize zero
.END		;A directive for the assembler

This SOURCE code file must be stored to disk using the extension of .ASM (filename.LMC) and will be read by the Little Man Assembler program (MASSEMB.EXE) to generate the LMC machine language file (filename.LMC). As explained above, all of these operations (editing, assembling, running, etc.) can be performed within the LMC menuing system.

Note that in this program, there are no numeric LMC operation codes and no numeric references to mailboxes. Instead, each instruction is represented by a mnemonic such as READ, ADD and STOP. Mailboxes are given names; the mailboxes LOOP and PRINTOUT hold instructions, while the mailbox SUM holds data. There are three special instructions in this program, each of which begins with a period, ".". These are instructions to the Assembler which are sometimes called directives. The Assembler is the program which translates the LMAL program above into an LMC machine language program which the LMC can then execute. The .LMC instruction must be on the first line of your LMAL program, and identifies the start of your LMAL code. The .END instruction must be on the last line of your LMAL program, and identifies the end of your LMAL code. The .DC instruction allows you to name a mailbox and place a value into it which can later be used by your program. In the example above, the mailbox is named SUM and the value loaded into this mailbox is ZERO. The mailbox gets loaded with the value of ZERO before program execution starts.

The Little Man Assembler

The LMAL program above would not be understood by the Little Man. The Little Man only understands numbers! Therefore, a translator is needed to turn the LMAL program into the numeric form that the Little Man understandsÑLittle Man machine language. This translator is known as the Little Man Assembler, because it takes a Little Man Assembly Language program and "assembles" a Little Man machine language program from it. The assembly process includes translating the mnemonic instructions into numbers, identifying a mailbox number for every instruction, finding a mailbox for all the data defined by DC directive, and resolving all references to named mailboxes by replacing the names with the assigned mailbox numbers. The input to the Little Man Assembler is a LMAL program, filename.ASM, and the output is a Little Man machine language program named filename.LMC which is ready to be loaded into the mailboxes starting at mailbox 00. The result of assembling the LMAL program above is the following Little Man machine language program file:

filename.LMC

0	901	***	READ
1	705	***	BZ
2	108	***	ADD
3	308	***	STO
4	600	***	B
5	508	***	LOAD
6	902	***	PRINT
7	000	***	STOP
8	000	***	
9	000	***	
10	000	***	
11	000	***	
	:		
99	000	***	

Now compare the assembly language version of this program with the machine language version. You will probably find that the assembly language version is easier to understand, as it uses mnemonics which remind you of the instruction and does not require the programmer to know the mailbox number for each instruction or each data element. It is usually a good idea to write all but the simplest LMC programs in assembly language.

The Little Man Assembler (MASSEMB.EXE) also produces two other output files: filename.DBG and filename.ERR. The .DBG file is an output file which contains information for debugging the assembler software and the .ERR output file contains any errors the assembler detected during the assembly process. The two file should be used when debugging your software.

Summary

That's about it for the description of the Little Man Computer. There will be a number of homework problems concerning the programming of the LMC using both machine language and assembly language.

Acknowledgements

My Acknowledgements

I would like to thank Dr. Jeffrey L. Hock for the paper version of this document. I have only converted the paper version into html format.

Dr. Hock's Acknowledgements

I would like to thank Professor Susan A. Riedel of the Department of Electrical and Computer Engineering at Marquette University for the vast majority of this handout. In a handout she had prepared on January 6, 1994 she had included the following acknowledgements:

Professor Riedel's Acknowledgements

The idea for the Little Man Computer originated with Professors Stuart Madnick and John Donovan at the Massachusetts Institute of Technology. The author is indebted to these wonderful teachers for this and many other inspiring pedagogical tools.

Much of the LMC software was written by the author's students at Marquette University, to satisfy course requirements in software design or as special projects. The earliest version of the LMC was written by Keith Fehl. The assembler was written by Anthony Tintera. The menu system was written by David McChristie, Denice Rocca, Cynthia Simpson and Jonathan Spangler. The Quick Debugger was written by Steven Betchner, Joanne Manders and Michael Terwelp. The editor and file management was written by Curtin Fleischer, Johathan Ley, Andrew Nault, Vincent Sablan and Paul Tarnow.
