The 'Little Man Storage' Model

Larry Brumbaugh William Yurcik

National Center for Supercomputing Applications (NCSA)
University of Illinois Urbana-Champaign
{ljbrumb,byurcik}@ncsa.uiuc.edu

Abstract

A simple but powerful storage model is described that has close correlation to generic storage systems. Extending the Little Man Computer paradigm developed by Stuart Madnick and John Donovan during the 1960s at MIT (where it was taught to all undergraduate computer science students), this paper describes a comparable development undertaken for disk and tape storage devices. A "Little Man Storage" paradigm is proposed to simplify the explanation of how storage devices function and how data is maintained by those devices.

1. Overview

For over forty years the Little Man Computer (LMC) paradigm has proved to be a simple but powerful and long-lived tool for teaching computer architecture to undergraduates in a field where a product is considered obsolete after 5 years (8 generations!). The authors of this paper have taught for many years with LMC simulators and have documented how LMC simulators can be useful teaching tools [1-4]. However, as computer architectures have evolved over time, subsystems within computers have also grown in complexity and capability such that their operation can no longer be effectively explained to undergraduates without new educational support.

In this paper we propose a new paradigm for teaching about storage systems, a core embedded subsystem coordinated with the larger computer architecture that has grown in complexity and capability to necessitate separate treatment. In fact many storage systems today have under-utilized processor capabilities such that we feel teaching storage systems may actually have an impact on future developments.

We propose a "Little Man Storage" model for teaching about storage systems consisting of elements similar to "Little Man Computer". By using ecological design in which model elements have intuitive meaning from human experience, we believe that a Little Man Storage (LMS) model may provide benefit in courses where storage systems are studied comparable to the impact of Little Man Computer. The LMS paradigm is consistent with the SNIA Shared Storage Model [5] that was developed to help standardize storage concepts across vendor platforms. This paper provides a conceptual overview of LMS as a precursor to a simulator implementation. It is our hope for feedback that can be incorporated into near-term development. This paper is meant as a discussion of educational techniques for communicating complex concepts in a learning environment and not as a tutorial, we assume readers a basic understanding of disk storage devices and how they store and manage data.

The remainder of the paper is organized as follows: after reviewing the LMC paradigm in Section 2, the LMS model is described in Section 3. In Section 4 the relevant LMS conceptual elements identified. Section 5 compares/ contrasts LMC and LMS to highlight our contribution. In Sections 6 and 7 file storage and data management are modeled. The discussion and examples focus exclusively on disk storage. An example is given of a typical storage processing operation that illustrates the individual steps within the operation and examples are also given that show the changes that occur in the storage device itself. Although not discussed in this paper, a small subset of this material can be used to illustrate tape/cartridge processing.

2. The Little Man Computer Paradigm

The LMC paradigm has stood the test of time as a conceptual device for helping students understand the processing that takes place inside a computer. One of its greatest strengths is its simplicity. The paradigm consists of a walled mailroom, 100 mailboxes numbered 00 through 99, a calculator, a two digit location counter, an input basket, and an output basket. Each mailbox is designed to hold a single slip of paper upon which is written a three digit decimal number. Note that each mailbox has a unique address and the contents of each mailbox are separate from its address. The calculator can be used for input/output operations, temporarily store numbers, and to add and subtract numbers. The two-digit

location counter is used to increment the count each time the Little Man executes an instruction. The location counter has a reset located outside of the mailroom. Finally there is the "Little Man" himself, depicted as a cartoon character, who performs tasks within the walled mailroom. Figure 1 illustrates the major components of the LMC paradigm. Other than the reset switch for the location counter, the only communication a user has with the Little Man is via slips of paper with three digit numbers put into the input basket or retrieved from the output basket.

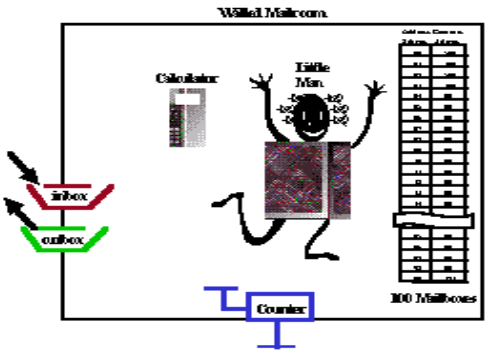

Figure 1. Little Man Computer and the Walled Mailroom

The authors have written several papers [1-4] describing use of a LMC simulator to enhance the quality of computer science courses, specifically those that emphasize architecture, hardware/software, and operating systems concepts. The two simulators developed by the authors are part of a larger worldwide effort to construct LMC simulators some of which are described in [3]. We feel these widespread developments validate both the utility and continuous interest in the LMC paradigm.

3. The LMS Model

We intend to leverage the LMC paradigm with corresponding conceptual analogies. In particular, the basic philosophy utilized in the LMC model is to minimize the functional details and physical structure while still allowing the important conceptual features to be clearly illustrated. The LMS model described here would have been valid with the disks of 30 years ago. However, more importantly it provides insight into modern storage systems. Furthermore, this paper describes a model, not a working simulation, but all the moveable pieces for the working simulation are presented.

Recall that a disk storage device contains several moveable components including: a) the revolving platters where data are stored, b) an access arm that moves to the designated location for the data and c) a mechanism for copying data between the buffers and the hard drive during input and output operations. Little Man Storage itself, again depicted as a cartoon character, performs all three of these functions.

4. LMS Hardware

The LMS disk device consists of two platters where data can be stored on both sides of a platter. Both the top and bottom surfaces of each platter surface contain three concentric tracks. Hence, the storage device consists of three cylinders. Each track consists of eight areas and all areas store exactly 512 bytes of data. Table 1 specifies the numbering scheme used to identify actual locations on the device. Figure 2 shows both sides of a platter.

Table 1. Basic Hardware Components of LMS

Storage Device Components	ID Numbering Scheme for the Component
3 cylinders	0, 1, 2 (independent of platter surface)
4 tracks per cylinders	0, 1, 2, 3 (0/1 1st platter & 2/3 2nd platter)
8 areas per track	0, 1, 2, 3, 4, 5, 6, 7 (same data each area)

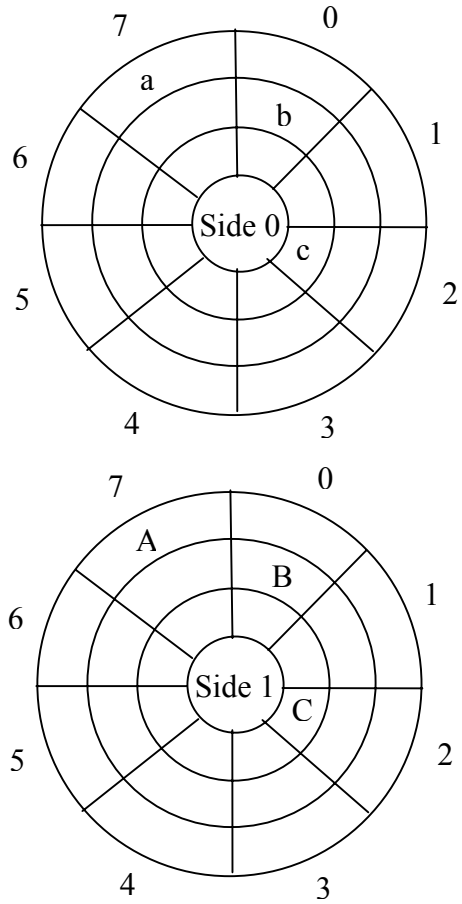

Figure 2. Both Sides of a Disk Platter

Areas can be referenced with values from 000 to 237. Address *xyz* identifies the location of the cylinder, platter, and area respectively. The small size of the storage device allows decimal numbers to be used for all three values, which simplifies addressing. Total disk capacity is 48K (=3 cylinders * 4 tracks/cylinder * 8 areas/ track * .5K bytes/area. Figure 2 shows one of the two platters in the storage device. The three area locations denoted by a, b, and c in Figure 2 have addresses of 007, 100 and 202 respectively. Area locations A, B and C have addresses of 017, 110 and 212 respectively. An alternative approach that was briefly considered that numbered the areas from 00 to 95.

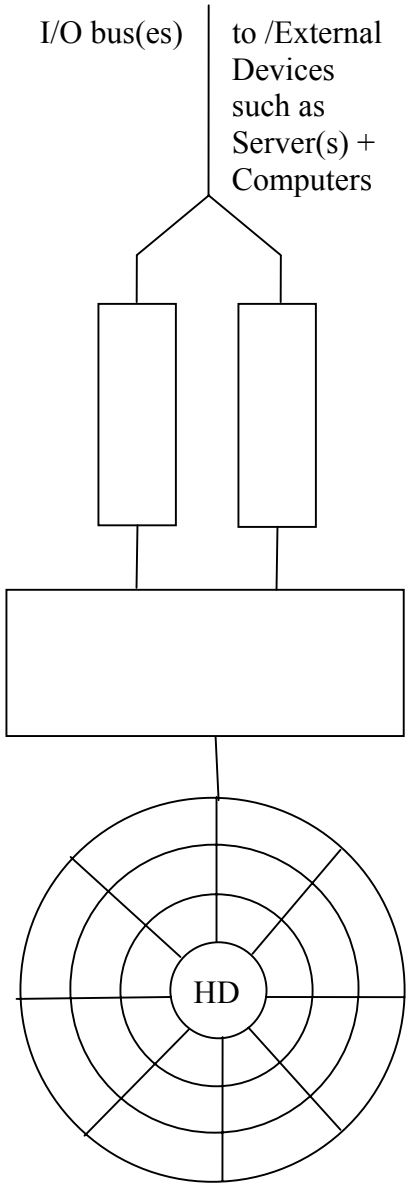

Figure 3. Physical Components that comprise an LMS Storage Device

The LMS model consists of the physical components shown in Figure 3. The disk controller is 'Little Man' (cartoon character) who provides the intelligence for disk operation and can perform a limited number of simple functions. In particular, LMS decodes and executes the commands sent to it from the attached server/computer. In implementing the commands, LMS uses one of its arms to read-data-from and write-data-to the hard drive (HD). The HD consists of the platters where data is actually stored. Communication paths called I/O buses connect the storage device to the source/destination of its data. Buffers are intermediate storage areas (pieces of paper) where data is placed both prior to copying it to storage and after retrieving it from storage and before sending it to the external device. There is one buffer (piece of paper) for data going in each direction.

In adhering to the LMC simplification principle, the disk contains no cache. Likewise, there are no auxiliary or reserved areas/tracks that can be used to replace parts of the disk that become defective. If part of an LMS device becomes inoperable, there is no way to designate processing options. No timing considerations are provided for any of the electromechanical components of the devices. Little Man Storage performs all the physical processing associated with the device. This includes using one arm to rotate the platters in the HD, using the other arm to move over a specific cylinder and then with the same arm copying the data to/from the HD.

5. Comparing Little Man Computer and Little Man Storage

Table 2 provides a comparison of the environments provided by the two paradigms and the types of physical acts that the Little Man must perform in each of them.

Table 2. Comparing LMC and LMS Characteristics

Environment/Physical Act Compared	Little Man Computer (LMC)	Little Man Storage (LMS)
historic relevance of paradigm	from 1960's to present	from 1970's to present
type of hardware device described	computer	disk storage device
actual hardware location of Little Man intelligence	CPU control unit	storage controller
locations where data is stored	100 mailboxes (00-99)	96 disk areas (000-237)
methods for performing I/O operations	read/write slips of paper	read/write disk areas
programmable device?	yes	no

6. File Storage and Data Management

The LMS storage device consists of 96 areas where 94 areas are used to store data and 2 areas are reserved to help manage the other 94. Area 000 contains a LIST of all files stored on the device. This is the only (the root) LIST on the disk. It is of fixed size (one area) and cannot be expanded. Table 3 shows the values stored in the LIST for several files. The location of the initial data in the file is specified in the Area Start Location as a (cylinder, platter, area) location. For simplicity, there are no attributes that can be assigned to a file. When a file is created, LMS adds a new row in the LIST. A new row is always added following the last or bottommost current LIST entry. If a file is deleted, its line in the LIST is erased. This is denoted as <blank> in Table 3.

Table 3. LIST Structure for the Disk

File Name	Size (Bytes)	Area Start	Creation Date
ALPHA.doc	10	006	06/06/2005
X.Y.Z	5000	128	09/18/1997
<blank>	-	-	-
NextFile1234.txt	0	225	12/25/2002
**********	-	-	-

Area 001 is used to manage the data areas that the device contains. Each of the 94 data areas either holds data associated with a file or is a free (unused) area. New files and additions to existing files obtain their storage from the free areas. It is the job of LMS to utilize this information in area 001 to retrieve and store files. LMS must also modify this information when necessary.

Initially, when the disk is first formatted, LMS marks areas 002 through 237 as free. This information is kept in a Free-Area-List. Whenever a file is created, one or more of the free areas are assigned to hold its data. When a file is deleted, the areas where its data were stored are returned to the Free-Area-List. Area 001 holds the Area Utilization List (AUL), where LMS stores information about the data areas. There are 96 entries in the AUL. The first two are used to manage the Free-Area-List and are described in the next section. The others entries are either used to identify the storage areas assigned to individual files or are a part of the Free-Area-List. Table 4 shows the initial portion of an AUL after 2 files have been written to the storage device. One file occupies 4 areas (002, 003, 005 and 006) while the second file occupies a single area (004). A value of 999 identifies the final area in a file. Note that areas 007 and 008 are either part of the same file or both are free areas. Free areas are shown in italics. LMS itself does all of this reading and writing of information.

Table 4. Contents of Area 001 Showing Storage Allocation after Two Files are Written

Area Number	Next Area Location in File
000	007 (first free area) *
001	00N (last free area) *
002	003 (file continuation)
003	005 (file continuation)
004	999 (end of file)
005	006 (file continuation)
006	999 (end of file)
007	008
. . .	. . .
237	999

Table 3 shows that the LIST entry for a file identifies only the first area assigned to it. The rest of the file location information is stored in the AUL. The AUL identifies the areas that are linked together to provide storage for the file. The final area contains a Next Area Location value of 999, meaning this is the last area associated with the file. Storage for a file need not be in contiguous areas. The areas that are not assigned to any file are tied together in the Free-Area-List. The areas at the beginning of this list are used to satisfy subsequent requests for storage. The Table 4 structure is actually an oversimplification used to clarify processing details. In reality, the AUL only needs to contain the rightmost column of values since LMS can determine the Area Number from its physical position in the list (by counting from the beginning of the list).

7. Additional Storage Model Parameters

LMS must remember three important values. It uses the first value to find an initial free area for new files and additions to existing files. This value is stored as the very first entry in the AUL (see Table 4). When additional storage is needed, LMS looks in this location and begins writing data to the corresponding area it identifies. Additional free areas can then be determined using the Free-Area-List. Once the last free area needed for the current processing operation is determined, its Next Area Location (the next free area) becomes the new first value in the AUL. Similarly, the second entry in the AUL identifies the final area in the Free-Area-List. When a file is deleted, its areas are added to the Free-Area-List following the area identified in the second AUL entry. The final area added to the list becomes the new value in location 2 of the AUL.

The third important value is the final entry in the LIST, which is identified by following it with a 'fake' file name entry of '**********'. The LIST is a white board where LMS writes entries for new files at the bottom of the board and erases entries for deleted files. Once the bottom of the board is reached, the LIST is

considered full and must be 'reorganized'. If there are unused erased rows on the board, rows on the bottom are copied to the currently erased rows and then erased from the bottom of the board. Following the LMC principle of simplicity, the LMS model places restrictions on the LIST structure and on the number of files that can be stored. With some effort this limit can be raised and subdirectories can also be used. Since this clearly will result in a more complexity, it is not discussed here.

Whenever a file is created, it is assigned one initial area. If no data are written to the file, LMS writes ***End-of-File*** at the beginning of the area. An area is never split between two distinct files. Hence, every file requires at least one area of storage and the maximum number of files is 94. An alternative approach that was strongly considered assigns the Start Location entry in the LIST for an empty file to a special value such as 999.

8. Storage Processing Operations

In the same manner that the CPU of a computer executes instructions, a storage device controller such as Little Man Storage is capable of executing a pre-defined group of commands that create, delete, store, retrieve and process data. Although some storage devices support a wider range of operations, we limit LMS to five commands as shown in Table 5. LMS processes complete files and individual records must be identified in the application programs (since storage devices are unaware of logical records). Each buffer can hold one area of data. A physical record consists of all the data in an area. LMS determines the actual location of a physical record that it needs by combining information from the command itself, the LIST, and the AUL. Each command is composed of steps in the same way that CPU instructions are composed of steps. EXAMPLE 1 illustrates the steps performed as part of a Read File command.

Table 5. Basic I/O Commands Supported by LMS

Command	OpCode	Processing Performed by Command
Create File	00	Write an entry in the LIST, including create date, etc. Initialize one Free-Area-List area to ***End-of-File***.
Delete File	01	Erase the file entry from the LIST. Return all AUL entries associated with the file to the Free-Area-List.
Read File	02	Begin in the LIST and then go through the corresponding AUL entries. With the alternative approach noted above, can also start in the AUL table.
Write File	03	Add data starting with the first area on the Free-Area-List. Write ***End-of-File*** after the last record is written.
Append File	04	Follow the AUL entries for the file to the one containing 999. Add new records in a new area and replace 999 with new area number.

All commands have the same basic syntax |op-code|filename|optional data|. In the case of Write and Append commands, the data to be written immediately follows the command code and file name. Op-codes are 1 byte in length, while file names are 20 bytes and can contain any printable characters. For example, |3|MY-NEW-INFO |*****| is a command to write 5 asterisks to a file called MY-NEW_INFO.

EXAMPLE 1: A paper is placed in the input buffer that says to get the data in the ALPHA.doc file. LMS looks at the command in the buffer and reads it, noting the command code (02) and the file name. LMS looks in the LIST and sees that initial data in ALPHA.doc begins in area 002. It rotates the disk until that area can be accessed. It copies the data from area 006 to the output buffer. LMS then looks in the AUL and notes the entry for area 006 identifies additional ALPHA.doc data in area 013. It uses one arm to move the disk to this location and the other arm to copy the data from 013 to the output buffer. This processing continues for every area where ALPHA.doc data is stored. When an AUL entry of 999 is found, the Read File operation is complete.

9. Detailed Processing Examples

Two examples are now given to illustrate all of the LMS components discussed to this point. Throughout all of these examples an unrealistic assumption is made that every operation is performed successfully. There is no way to recover from an invalid or incorrect operation.

EXAMPLE 2: It is assumed that the HD is formatted and all 94 data areas are free. File AA is created and several small records are written to it. File BB is created and enough records are written to it to fill three areas. Several additional records are then added to AA, requiring a new area to be allocated using an Append File command. A third file GG is created, but no records are written to it. Finally, file DD is created and three areas have data written to them. Figure 4 shows the relevant areas following the processing. The first two areas contain the LIST and the AUL.

	0	1	2	3	4	5	6	7
00	-	-	AA	BB	BB	BB	AA	GG
01	DD	DD	DD	-	-	-	-	-
02	-	-	-	-	-	-	-	-

Figure 4. Disk Status Following the I/O Operations in EXAMPLE 2

EXAMPLE 3: This example begins immediately after the processing in EXAMPLE 2 has completed. File AA is deleted. Two new files called SS and RR are created and one byte of data is written to each file. Additional records are then written to SS. Figure 5 shows the relevant areas following the processing.

	0	1	2	3	4	5	6	7
00	-	-	-	BB	BB	BB	-	GG
01	DD	DD	DD	SS	RR	SS	-	-
02	-	-	-	-	-	-	-	-

Figure 5. Disk Status Following the I/O Operations in EXAMPLE 3

10. Summary

We have introduced a new Little Man Storage model for teaching about computer storage systems. While this paper focuses primarily on conveying disk storage concepts, work is underway for developing a Little Man Storage software simulator that extends the storage concepts demonstrated beyond disks. Results from the educational use of this model will also provide feedback on the effectiveness of this model in targeted learning environments.

11. References

[1] W. Yurcik and L. Brumbaugh, “Using LMC Simulator Assembly Language to Illustrate Major Programming Concepts,” *Info. Systems Education Conf. (ISECON),* 2001.

[2] W. Yurcik and L. Brumbaugh, “A Web-Based Little Man Computer Simulator,” *32nd Technical Symposium of Computer Science Education (SIGCSE),* pp. 204-208, 2001.

[3] W. Yurcik and H. Osborne, “A Crowd of Little Man Computers: Visual Computer Simulator Teaching Tools,” *Winter Simulation Conference (WSC),* 2001.

[4] W. Yurcik, J. Vila, and L. Brumbaugh, "An Interactive Web-Based Simulation of a General Computer Architecture," *IEEE Intl. Conf. on Engineering & Computer Education (ICECE*), 2000.

[5] *SNIA Shared Storage Model White Paper.* <http://www.snia.org/tech_activities/shared_storage_model/SNIA-SSM-text-2003-04-13.pdf>