PORTIONS OF THIS DOCUMENT ARE ILLEGIBLE

LA-UR -82-1133

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE DISKLESS LSI-11 SYSTEMS

AUTHOR(S) Wendell Ford
Donald G. Shirk

SUBMITTED TO Digital Equipment Computer Users Society Symposium, Atlanta, Georgia, May 10-14, 1982

Los Alamos
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO 836 R4
ST NO 2629 5/81

DISKLESS LSI-11 SYSTEMS

Wendell Ford and Donald G. Shirk
Los Alamos National Laboratory
Los Alamos, New Mexico

ABSTRACT

Programs for dedicated LSI-11 based systems can easily be stored in ROM instead of floppy disks, yet execute the same as disk-stored programs. Two sample systems are described.

INTRODUCTION

Certain applications can benefit greatly by the powerful intelligence available through use of a mini or microcomputer. The benefits are often negated if the resulting system is too complicated for operation by unsophisticated users or contains components not suitable for the application environment. An example is equipment that uses a general purpose LSI-11 computer complete with full feature terminal and floppy disk drives, in an industrial environment that must be operated by production personnel. It is unreasonable to assume the operator can ensure the correct floppy is loaded, bring up the operating system, enter the proper command to run the application program, and be able to resolve problems at any step.

We are developing a number of instruments for a chemical process line. The instruments are to be operated by a variety of persons with diverse backgrounds, i.e., starting grade technicians to floor supervisors. Simple, straightforward operation of all equipment is needed. The environment is that of a modern, well-maintained processing plant. The atmosphere is controlled to be safe for human occupation, although abnormal conditions occasionally occur. Small magnitude vibration may be expected.

Several goals are common to the various finished systems. The following goals were to be met at minimum cost and with a minimum number of components.

(1) Create a very 'friendly' user environment; one that would not intimidate unskilled users. Simply turning power on is all that is required to begin operation.
(2) Achieve reliable operation in an industrial environment. Maintenance needs to be accomplished quickly and easily in the field.
(3) Protect the program and data from accidental or malicious corruption by the operator or others.
(4) Any communications with other processors or devices should be via simple networks.

These goals translated into more specific requirements.

(1) If possible, the need for a disk drive and full feature terminal should be eliminated.
(2) All hardware components should be readily available commercial units.
(3) The size should be as compact as possible.
(4) Maintenance and repair would consist of swapping components or circuit boards.
(5) All communications should be by existing industry-wide standards.
(6) Software development, not necessarily developed on the field system, should be portable.
(7) Installation and/or revision of the software in the field should be easy and direct.
(8) Where possible programs should be written in high-level language.
(9) The cost of the field-installed system was to be kept low.

SYSTEM DEVELOPMENT

We implemented two systems meeting the above requirements in which the program is stored in EPROM, loaded into RAM at turn-on by the bootstrap loader, and executed out of RAM. One system is used to read data from intelligent scientific instruments. This system successfully interfaces an electronic balance and a solution mass measuring device. The other system illustrates use of an existing host computer for control and mass storage. This system is a speaker-identity-verification system acting as a satellite to a PDP-11/34. The host PDP-11/34 controls security system operation and stores voice templates for each user. The satellite LSI-11 performs voice verification and returns the decision to the host.

To meet the listed requirements, the following system attributes were developed. The software resides in EPROM. To eliminate any concerns over conflicts between ROM and RAM address space, the program is downloaded from ROM to RAM for execution. The program auto loads at turn-on. Different programs are executed by entering the bootstrap loader at different entry points. The operating 'panel' is a compact or hand-held terminal. The communications are serial EIA or current loop using standard baud rates. A small thermal printer is available for hard-copy output.

Hardware

A basic LSI-11 computer hardware configuration was selected for use in the systems. The enclosure is a BA11-M with front panel switches and lights. This chassis allows up to eight dual-wide boards in 3 1/2" of rack space. Four dual-wide boards are used: an LSI-11/2 KD11-HA CPU, a MSV11-DD 16 bit by

32K RAM memory, a DLV11-J 4-channel serial interface, and a MRV11-C 64K-byte ROM board. DLV11-KAs are used to convert from EIA to current loop where required. The bootstrap is contained in EPROM on the MRV11-C. For the speaker-identity-verification system, the two remaining quad slots are available for the voice recognition module and speech synthesizer module. The current system configuration uses external modules connected to serial ports. Single quantity list prices for the LSI-11 components is about $4200, the voice recognition module is about $2100 and the speech synthesizer custom built is less than $1000.

Software

The majority of the applications software was written in RT-11 FORTRAN IV. Use of MACRO-11 was restricted to the channel driver routines. Under stand-alone operations, console terminal FORTRAN I/O was provided by the $SIMRT module, which was included at LINK time.

Production of task image .SAV format files was straight forward. Standard FORTRAN compilation procedures were used. The LINK step involved one additional switch selection in the command line. The standard LINK command line was modified with the '/INCLUDE' switch to allow inclusion of the $SIMRT module.

The .SAV task image now serves as input for the EPROM programmer software procedures. These software procedures read the task image file from the floppy, convert it to the required high and low byte format, then program the high and low bytes EPROMs. Used in this manner both the high and low byte EPROMs are programmed simultaneously.

Code for the bootstrap loader is written in MACRO-11 and is assembled to produce a machine instruction listing. These binary instruction codes are then entered manually into a standard EPROM programmer and the bootstrap EPROMs are programmed.

Program Storage and Execution Environment

The executable programs are stored in EPROM on the MRV11-C board. Maximum capacity of the MRV11-C is 64K-bytes and up to 16 MRV11-C's may be included in a system. Provision is also made for storing and accessing the bootstrap loader in a MRV11-C. The MRV11-C provides two different addressing modes: direct and window mapped. Direct addressing places all ROM locations in the LSI-11 address space the same as any other memory. Address decoding options allow considerable freedom in positioning the ROMs in the address space. Window mapping provides access to the ROM locations through two independent 2K-byte windows that may be located anywhere in the 64K-byte address space. The segments that are viewed through each window can be varied under program control. The bootstrap option may be used by either mode or disabled. When the bootstrap option is used, configuration jumpers allow several choices of bootstrap starting address. When the window mapped mode of operation is selected, only two 2K-byte segments of the ROM are visible to the LSI-11 address space.[1] To prevent conflicts with RAM address space, we used only one window located in the bottom of the I/O page, and thus the EPROM space is invisible to the LSI-11 address space. CPU power-up mode 2 was selected and the bootstrap located at address 173000. The bootstrap is designed to load a specified section of the ROM space through the window into the RAM space and begin execution. Thus, at power-up the program is automatically transferred from ROM to RAM and execution begun. Other programs can be loaded by different bootstrap entry points, which specify loading different ROM locations. There are various options one could use for entering the bootstrap.

SPEAKER IDENTITY VERIFICATION SYSTEM

A brief description of the speaker-identity-verification system is given here (a more complete discussion will be the subject of a future paper). An early version of the system and some preliminary results appear in Ref. 2. A PDP-11/34 acts as controller for a complete security and access control system. The LSI-11 is a satellite to the PDP-11/34. A person is enrolled in the system by a training process that generates a set of voice templates. At the end of training the voice templates are stored on disk in the PDP-11/34 under a user ID number. When a person wishes to be verified (for example, enter a controlled area), he enters his ID number via a numeric keypad. The 11/34 verifies that the number is valid and downloads the appropriate voice templates to the LSI-11, which in turn downloads them to the voice recognition module. The user is then prompted through a series of utterances and his identity checked. All control of the LSI-11 is by commands from the 11/34; communications is by serial ASCII characters.

At power-up the LSI-11 comes up in a quiescent state awaiting an 11/34 command. The program that is loaded by the bootstrap is the normally used verify routine. After executing the 11/34 command, the LSI-11 replies and returns to the await-command mode. If the command is to enroll a new person, the program jumps into the bootstrap at the point that loads and executes the training program. At the end of the training session a jump is again made to the bootstrap to load and execute the verification program. All this action is transparent to the 11/34 and to the user as all loading and starting of programs requires no operator intervention.

INSTRUMENT MONITOR SYSTEM

The other application (reading scientific instruments) is a single program loaded at power-up (Figure 1). The user interacts through a compact (or hand held) ASCII terminal. When power is turned on, the program is loaded, starts executing, and a start-up message is displayed on the terminal. The operator may then select the desired operation (reading the instrument of interest) and the result is displayed on the terminal. If a hard copy is needed, the result can be printed on the compact printer. Because the terminal is used in the console channel, simulated FORTRAN I/O was used. The other channel drivers are written in MACRO-11 and allow asynchronous operation. Another feature is the use of the line-time clock. If the interogated device does not respond within a prescribed time (as timed by the line-time clock), the program alerts the operator and awaits the next command. Ultimate use for the system would allow for control from a host computer to place the instruments in an on-line data acquisition mode. Operator interaction would be limited to indicating a measurement is ready to be taken or taking care of special cases.

Figure 1. Instrument monitor system.

In both systems the full 56K-byte RAM space is available for the applications programs (64K minus the 8K I/O space). No partioning of the memory space into RAM or ROM areas was needed. The number of application programs could be expanded by the addition of up to 16 MRV11-Cs (up to 64K-byte each). As more programs are added, more clever use of the bootstrap space may be required. The power-up program could be a menu program for selecting one of several alternatives.

SOFTWARE DEVELOPMENT ENVIRONMENT

The software development was done on a PDP-11/04 host operating under RT11 V4.0. The 11/04 system contains two RL01 hard disks and an RX02 floppy disk. Thus, the speed of the hard disk was used for program development and the convenience of the floppy was used for program transfer. The development LSI-11 system also contains an RX02 and runs RT11. A third party Q-bus compatible EPROM programmer was purchased for transfering the programs from floppy disk to EPROM. The programming system provided a word programming mode (two 8-bit wide EPROMs simultaneously) so that partioning into high-byte, low-byte was not necessary. The jumpers on the MRV11-C allow use of a wide variety of EPROMs, limited to the single power supply variety. We have successfully used the Intel 2716 and the TI TMS2532. Some care is required to select proper jumpers for the 129 wire-wrap pins.

SUMMARY AND CONCLUSIONS

Our work has shown that compact LSI-11 systems for dedicated (but flexible) applications can be built. No disk need be present; the program can begin operation at power-up. The painful process of ROM-RAM partioning can be avoided and most of the software can be written in a high-level language (FORTRAN in our examples). However, the FORTRAN code did require considerably more memory for the console I/O than a corresponding assembly language. Development could be done on a LSI-11 system with floppy disk, or the faster speed of a hard disk system could be used and the programs then transferred to the floppy disk.

REFERENCES

1. Microcomputers and Memories (Digital Equipment Corporation, 1981).

2. W. Ford and D. G. Shirk, "Applications of Voice Input/Output," Nucl. Mater. Manage X, 432-437 (1982).