TITLE

Development of a Minimal Operating System for the ALTAIR 8800 Microcomputer.

AUTHOR

George T. Dupont
Student
Computer Science and Statistics Department
University of Southern Mississippi
Hattiesburg, Mississippi

ABSTRACT

ATMOS is a minimal operating system for the ALTAIR 8800 microcomputer. The program is being written to provide the Computer Science and Statistics Department of the University of Southern Mississippi with a minimal operating system which will be used in the development of ALTAIR 8800 software.

The program when completed will permit an ALTAIR user to interact with the University of Southern Mississippi's Xerox SIGMA 9 computer, will provide the user with the capability to dynamically debug ALTAIR software, and will permit the user to use the SIGMA 9 system as a file storage device.

INTRODUCTION

This paper presents a description of ATMOS, an ALTAIR 8800 Minimal Operating System.

ATMOS is being developed to provide the Computer Science and Statistics Department of the University of Southern Mississippi with a rudimentary operating system which will be used in the development of software for the Department's two ALTAIR 8800 microcomputers. Because ATMOS is primarily a software development tool and only secondarily an operating system, the emphasis placed on some of the more traditional operating system services has been minimized in favor of an increase in the emphasis placed on debugging services.

While ATMOS itself is not meant to be a full-scale operating system, however, it is being written in such a way as to permit it to serve as a nucleus around which such an operating system can be built. This is partciularly true in the area of I/O handlers where stubs are being purposefully designed into the program to permit the later addition with a minimum of program revision of an I/O monitor capability.

ATMOS has two modes of operation:

a. The transparent mode permits an ALTAIR user to interact with the University's Xerox SIGMA 9 computer and to thus take advantage of the larger system's hardware and software;

b. The executive mode permits the user to execute programs in the ALTAIR taking advantage of the ATMOS debugging capabilities and to use the SIGMA 9 as a storage device for programs and other files.

THE ALTAIR 8800

The ALTAIR 8800 is a microcomputer built around Intel's 8080 microprocessor chip. This chip contains an 8-bit accumulator, six 8-bit data registers, a 16-bit program counter, a 16-bit stack pointer register, an 8-bit arithmetic logic unit, four testable flag bits, and all of the other registers, busses, and timing and control lines necessary to permit it to provide the ALTAIR with a complete 8-bit central processing unit. The six data registers can be treated as three 16-bit registers and these can be used for double word arithmetic operations and as address registers for directly and indirectly addressing the ALTAIR's memory. The stack pointer register and the 8080's stack architecture permit any section of the ALTAIR's memory to be used as a last in/first out stack.

Memory for the University's two ALTAIR's is provided by static MOS RAM's. One of the computers has 12K 8-bit bytes of

memory; the other has 16K bytes.

The ALTAIR can be interfaced with a variety of input/output devices. Although there is little likelihood that the University's ALTAIR's would be expanded to the maximum capability, the Intel 8080 chip does permit up to 256 input devices and 256 output devices to be interfaced with it.

The interfaces with which ATMOS is concerned are provided by an RS-232 compatible serial input/output board. Each of these boards provides two input ports and two output ports. Each of the ALTAIR's has been interfaced through one of these boards with a Texas Instruments Silent 700 terminal and with a modem which in turn is interfaced with the University's Xerox SIGMA 9 computer.

ATMOS - THE TRANSPARENT MODE

Because of the ALTAIR's interface with the SIGMA 9, all of the larger system's hardware and software have been made available to the ALTAIR user. ATMOS' transparent mode includes the software necessary to permit the ALTAIR user to contact and to interact with the SIGMA 9.

The SIGMA 9 programs which are of the most interest to the ATMOS user are the SIGMA 9's Terminal Executive Language (TEL) which provides extensive file manipulation capabilities, the SIGMA 9's Editor, a cross-assembler for ALTAIR assembly language, an ALTAIR simulator, and a conversion program which transforms the output of the cross-assembler into a format which can be more efficiently transmitted through the modem to the ATMOS executive mode loader routine. Cross-compilers will, of course, be added to this list as they become available.

ATMOS - THE EXECUTIVE MODE

In the executive mode, ATMOS will provide the user with several options which include the capability to store and retrieve files using the SIGMA 9 system as the storage device and to execute programs in the ALTAIR using the dynamic debugging capabilities of ATMOS.

To provide an indication of the capabilities which ATMOS will possess when completed, this section presents a description of each of the executive mode commands. As will be seen, these commands permit the ALTAIR user to examine or change any section of memory, to examine or change the contents of any of the registers or any of five flag bits, to set break points in the user's program and to clear those break points, to initiate execution of the user's program or to resume execution after the program is suspended at a break point, to load the ALTAIR's memory from the SIGMA 9 or to create a file on the SIGMA 9's storage devices from the

ALTAIR's memory, and to shift from the executive mode to the transparent mode.

The format of the ATMOS executive mode commands follows that of the ALTAIR simulator in the SIGMA 9. Where practical, the commands are identical to those of the simulator.

All digits used in ATMOS commands are assumed by the program to be octal. Addresses are specified as two 8-bit bytes expressed by three octal digits for each byte. The address 011 122, for example, refers to memory location 004522 (i.e., $011_8\ 122_8 = 00\ 001\ 001_2\ 01\ 010\ 010_2 = 0\ 000\ 100\ 101\ 010\ 010_2 = 004522_8$). This convention follows that of the ALTAIR cross-assembler listing and is therefore considered the most natural for the ATMOS user to use.

In the list of commands which follows, characters which are not underlined are those typed by the operator; those characters which are underlined are those initiated by the ALTAIR. The symbol "@" is the ATMOS cue and is typed by the ALTAIR whenever a command is expected from the user. The numbers used in the following examples are representative only. All input strings to ATMOS are terminated by a carriage return.

COMMAND	DESCRIPTION
@D,001,022,001,326	Dump Memory.
	This command will cause ATMOS to print on the terminal the contents of locations 001 022 through 001 326. Ten groups of three octal digits (i.e., bytes) are printed on each line. The first two bytes printed on each line form the address of the first memory location whose contents are printed on that line; the remaining eight bytes are the contents of that location and the next seven memory locations.
@I,001,323	Input to memory.
001 323 215 001 324 207 001 325 @	This command permits the user to modify from the terminal the contents of the ALTAIR's memory. When the operator types the basic command and terminates the line with a carriage return, ATMOS types the address of the memory location to be changed. The operator types the new value for

that location and terminates the line with a carriage return. ATMOS types the address of the next sequential location in memory; the operator types a new value and a carriage return. This process continues until the operator types a carriage return without first typing a new value. This causes ATMOS to terminate the input operation.

In the example, location 001 323 was changed to 215, location 001 324 was changed to 207, and location 001 325 was not changed.

@E,A,213

@E,H,315

@E,SP,013,367

Enter Register.

The enter command is used to modify the contents of a register. There are two formats for this instruction because of the different lengths of the ALTAIR's registers. The accumulator and the six data registers are by Intel's convention referred to by letter designations. The accumulator is register A; the other six are B, C, D, E, H, and L. The format used in the first two examples is used to modify these 8-bit registers. The stack pointer register (SP) is a 16-bit register and is modified using the two-byte format shown in the third example. The effect of the three examples would be to change the contents of register A to 213, the contents of register H to 315, and the contents of the stack pointer to 013 367.

@F,C,0

@F,Z,1

@F,P,0

Modify Flag Bit.

The Intel 8080 chip contains four testable flag bits. They are carry (C), zero (Z), sign (S), and parity (P). There is a fifth flag bit, auxiliary carry (A), used in adjusting the accumulator for decimal arithmetic. The "F" command modifies these flag bits. In the examples, the carry bit is reset (0 or logic false), the zero flag is set, and the parity flag is reset.

@B,002,304

Set Break Point.

LEN 2

This command is used to set a break point in the user's program. After execution of the user's program is begun, an attempt to execute the instruction in the given location (in the example, 002 304) will cause control to be transferred back to ATMOS. ATMOS will then print on the terminal the contents of the program counter, the accumulator, the six data registers, the five flag bits, and the stack pointer. ATMOS will then print the cue and wait for a command from the operator.

Because ALTAIR instructions can be one, two, or three bytes in length, the operator is requested to type the length (LEN) of the instruction for which the break point is being inserted. This information is readily available from the cross-assembler listing.

@C,002,304

Clear Break Point.

@C

This command is used to clear break points previously set by the "Set Break Point" command. The first example clears the break point at location 002 304 (if none exists, the command has no effect). The second example clears all break points currently in the user's program.

@R,001,307

Run.

@R

This command permits execution of the user's program. In the first example, a starting address is specified. This is the format of the command which would be used to initiate execution of the user's program. The second example is the command which is used to resume execution of the user's program after execution was suspended at a break point.

@S,001,023,002,321

Store.

FID filename

This command is used to create a file in the SIGMA 9 system from the ALTAIR's memory. The command causes ATMOS to shift into the transparent mode, to form and transmit to the SIGMA 9 the appropriate TEL commands to permit building of a file, to begin transmitting the contents of the specified memory locations, and at the end of the store operation to shift back to the executive mode. In the example, the contents of ALTAIR memory locations 001 023 through 002 321 are stored in the SIGMA 9 system under the file identifier (FID) "filename". The format in which the file is stored is the same as that produced by the ALTAIR cross-assembler conversion program mentioned earlier and is suitable for loading using the "Load" command.

@L filename

Load.

This command loads the memory of the ALTAIR from the SIGMA 9. The command causes ATMOS to shift into the transparent mode, to form and transmit to the SIGMA 9 the appropriate TEL command to cause dumping of a file, to begin loading of the file at the addresses specified within the file, and to shift back to the executive mode upon completion of the load operation.

CTRL A

Shift Modes.

The character formed by depressing both the CONTROL key and the A key on the terminal simultaneously is used as the command which causes ATMOS to shift modes. If in the executive mode, the shift is to the transparent mode; if in the transparent mode, the shift is to the executive mode.

ATMOS DESIGN

While an attempt is being made to keep the memory requirements and execution time of the program at a minimum, this is not being done at the expense of program clarity or program flexibility. Even though the program is being written in ALTAIR assembly language, extensive use of modular programming and a top-down approach to the problem of program design have resulted in a relatively straightforward and easily understood program.

Of the routines which make up ATMOS, only two have been less than routine in their implementation. These two are the routines which set program break points and which transfer control back to ATMOS upon sensing of one of these break points. The difficulties which have arisen with user program break points have been caused by a combination of three factors -- the three different ALTAIR instruction lengths, the necessity for maintaining program flexibility, and the fact that in an 8-bit computer there are very few spare bits available.

The method by which break points are implemented in ATMOS involves the substitution of a branching instruction for the instruction in the user's program for which the break point is to be inserted. Because most ALTAIR instructions are either one byte or two bytes in length, however, the 3-byte jump instructions and the 3-byte call instructions are of little practical use in the implementation of break points by this substitution method. The ALTAIR does have one one-byte branching instruction in its repertoire -- the restart (RST) instruction. This instruction when executed pushes the contents of the program counter onto the LIFO stack and performs a branch to one of eight reserved locations in the first 64 words of memory. The specific location to which the branch is taken depends on a 3-bit field within the instruction.

While one possible approach to the implementation of break points would be to assign a different 3-bit code to each break point, this would effectively remove the RST instruction from the repertoire for other uses. ATMOS, it will be remembered, is being designed not only to meet existing needs, but also to permit easy expansion as future requirements dictate. Future requirements will almost certainly include the addition of an I/O monitor capability and an expanded use of the ALTAIR's program interrupt facility. Not only is the RST instruction one of the most natural to use as an I/O monitor call, but also the eight reserved locations to which the RST branches are the very ones to which program interrupts trap. Obviously, program flexibility would best be served by taking another approach.

The approach taken in ATMOS uses only one of the eight reserved locations by causing all break point RST's to have the same 3-bit code. The ATMOS "Set Break Point" command causes this

RST command to be substituted for each byte of the specified instruction in the user's program and causes the latter instruction to be moved to a holding area in memory. When the user's program executes the break point RST command, the value of the program counter at that time is placed on the stack and control is transferred to the break point reserved location. ATMOS simply pops the stack and uses the value of the program counter which is stored there to determine which break point was executed. ATMOS then saves the value of all of the other registers and flags and transfers control to the operator at the terminal. When program execution is resumed using the executive mode "R" command, ATMOS restores the registers and flags (including any changes made during the break), executes the original instruction in the holding area, and performs a branch to the next instruction in the user's program.

CONCLUSION

This paper has presented a description of a minimal operating system for the ALTAIR 8800. While ATMOS itself is indeed minimal, the design of the program is such as to permit its eventual growth.

The discussion of the implementation of break points pointed up the conflict that occasionally arose between the needs of the present version of the program and the anticipated needs of future versions. In such instances, satisfaction of the requirements of the future program was the first concern.

REFERENCES

Information presented in this paper concerning the Intel 8080 microprocessor chip was abstracted from the following two sources:

1. Intel 8080 Microcomputer System Manual, Intel Corp., Santa Clara, Ca., 1975.

2. Soucek, B. Microprocessors and Microcomputers. John Wiley and Sons, New York, 1976, pp. 251-298.