Animations of Computers as Teaching Aids

Thomas R. Leap
Computer Science Department
Elizabethtown College
Elizabethtown, Pennsylvania 17022

This paper discusses several programs which are used as teaching aids for teaching computer science students. The programs are animations of the internal workings of a central processing unit. They should be particularly useful in assembly language and computer organization courses or to give introductory students a more tangible example of what is going on inside the computer. The animation techniques use only the capabilities of common dumb conversational computer terminals and can easily be implemented on many different computer systems.

INTRODUCTION

Simple character graphics can be used to turn a computer terminal into an automated blackboard. When combined with animation, these illustrations can produce dramatic simulations of normally intangible operations. Teaching aids for teaching Computer Science using the computer terminal and these techniques include: animated arrays for demonstration of sorting or list processing; an animated computer to demonstrate the internal operation of computers; illustrated computer and equipment simulations where laboratory stations are unavailable; and animated tracing aids for teaching program execution principles and debugging techniques. All of these teaching aids have been implemented or are being developed at Elizabethtown College. They have been implemented on the college's Digital Equipment Corp. VAX computer using common dumb terminals and are programmed in the Pascal language. All of these teaching aids should be easily transportable to any computer or microcomputer with Pascal language capability.

The graphics and animation techniques necessary to implement these aids can easily be created on most computer terminals connected to timesharing computers and on microcomputers. The basic technique for animation consists of drawing a picture or displaying information on the terminal, then redrawing a changed picture or new information over the old to simulate motion or a continuing operation. This animation can be used to indicate the movement of data along buses in a computer or the modification of information in a memory element or register.

HARDWARE REQUIREMENTS

The computer terminal is the most important piece of hardware for performing animations. When microcomputers are used, the microcomputer's display can be considered to be a computer terminal. The features and capabilities of this device determine the quality of the displays that can be generated. Graphical illustrations can be created using line drawing character sets or simple characters such as "+" for corners, "-" for horizontal lines and "!" for vertical lines. Even though character graphics is not as impressive as solid line graphics, the result is equally effective.

When performing animations, the speed of the terminal becomes important. 300 baud operation is too slow to perform acceptable movement of objects on the terminal. Generally, 2400 baud operation is both acceptable and optimal.

TERMINAL INDEPENDENCE

Terminal independence may not be very important in many schools where there is only one kind of computer terminal in use. However, some schools are using several different types of terminals. Also, if a program is to be transportable, it should be written to handle different types of terminals.

Even though there is an ANSI standard for terminal commands, there are still many different command sequences which different types of terminals recognize. Dependency on any one command sequence can be eliminated by implementing a set of centralized routines in a program which issue the terminal dependent commands. The program can inquire about the terminal type using one of several common methods. These include: asking the operating system (this requires that the operating system maintain a list of terminals and their types), issuing an identification request command sequence to the terminal and examining the response (this will only work with terminals which respond to such a command), or asking the user what type of terminal he is using. The terminal type can then be used in the command sequence generation routines to select the proper command sequence.

It is important to develop a profile of a standard terminal which has features that are common among all the different types of terminals with which the program should work. If this set of features is kept simple, it is probable that commands for new terminals can easily be added to the program. The common features among most terminals include: a 24 line by 80 column display screen, direct cursor addressing and screen clearing (erasing the entire screen). A terminal must at least have direct cursor addressing to be be used for animation.

The size of the screen is only important in determining the size of the picture or the amount of information that can be displayed. If a terminal has a larger screen size, columns past 80 and lines past 24 can probably be ignored. If an animation is to work on a terminal with a smaller display size (such as 16 by 40 in some microcomputers), there may not be enough display area to show everything.

Screen clearing is not an absolute necessity since it can be performed by writing spaces across the entire screen. Most terminals also have the additional functions for clearing to the end of a line and to the end of a page. These two functions can also be performed by writing spaces to clear the respective areas of the screen.

Video attributes such as reverse video, bold or blinking text are nice additions to an animation but are somewhat less standard among terminals. Some terminals only allow one video attribute to be displayed at any one time. On some terminals, one character position on the screen is used every time the video attribute is changed. In order to remain applicable to many terminals, the standard terminal profile should allow only one video attribute at a time and always use one character position when changing attributes. A blank space should be left immediately before and after any field on a screen that will be displayed using a video attribute. This will allow room for the video attribute change character.

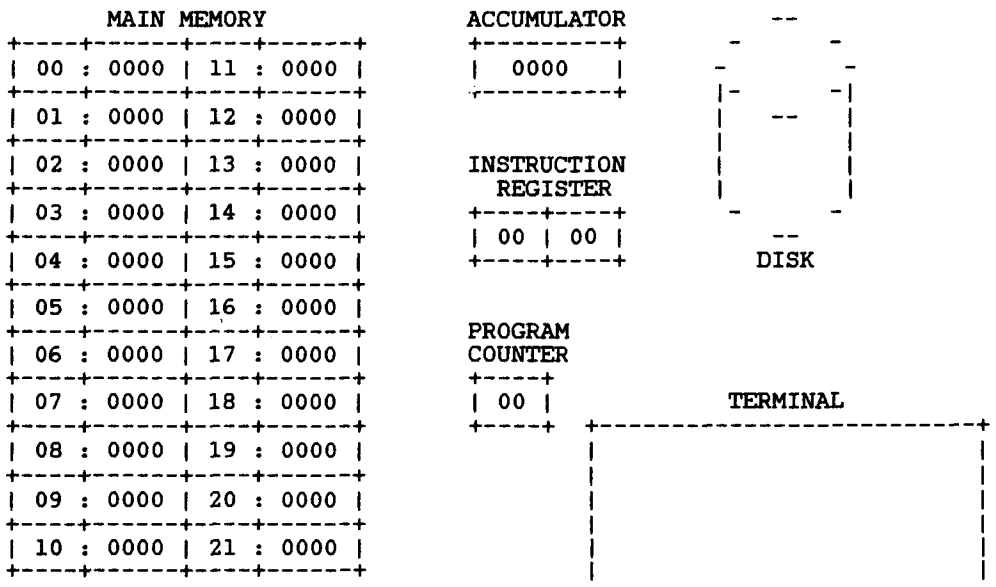
The animations described in the rest of this paper use a fairly simple standard terminal profile. The features that are needed are a 24 line by 80 column screen size, direct cursor addressing and reverse video capability. All other features are simulated when necessary. Scrolling of the terminal screen is even simulated by redrawing the portion of the screen that is to be scrolled. The terminals that the animations have been implemented on include: the Digital Equipment Corp. VT100 and VT52; the Applied Digital Data Systems Viewpoint A1 and 60; the Lear Seigler ADM 5; the Hazeltine Esprit I and III; and the IBM personal computer running a program emulating the ADDS Viewpoint A1.

AN ANIMATED COMPUTER

The Cardiac "cardboard" computer from Bell Telephone Laboratories has been very useful in teaching students the fundamentals of information storage and instruction execution in a central processing unit. One attempt to improve on this teaching aid is to simulate the Cardiac computer on a real computer. Programs can be entered from a terminal, executed by the simulator program and the result printed on the terminal. Unfortunately, this bypasses one of the best features and original purposes of the Cardiac computer. The student no longer sees what is going on inside the computer during execution.

Character graphics and animation can be used to restore the view of the internal parts of the computer. This animated computer is similar to the "visible" automobile engine or "visible" man where the internal parts are visible.

One animated computer in use at Elizabethtown College is used in introductory Computer Science courses. The animated computer consists of 100 words of memory, an accumulator, an instruction register and a program counter. A terminal for input and output



Animated Computer Display Layout
Figure 1.

and a disk for program storage are also included in the animation.

Since it is intended for use in an introductory course, the animated computer is kept very simple. The instruction set of the computer consists of basic instructions for: loading and storing the accumulator; integer addition, subtraction, multiplication and division operations; jump and jump if accumulator is negative; terminal input and output; and halt. Memory words and the accumulator hold 4 digit decimal values. Instructions are decoded by breaking a word in half to give an operation code and an operand.

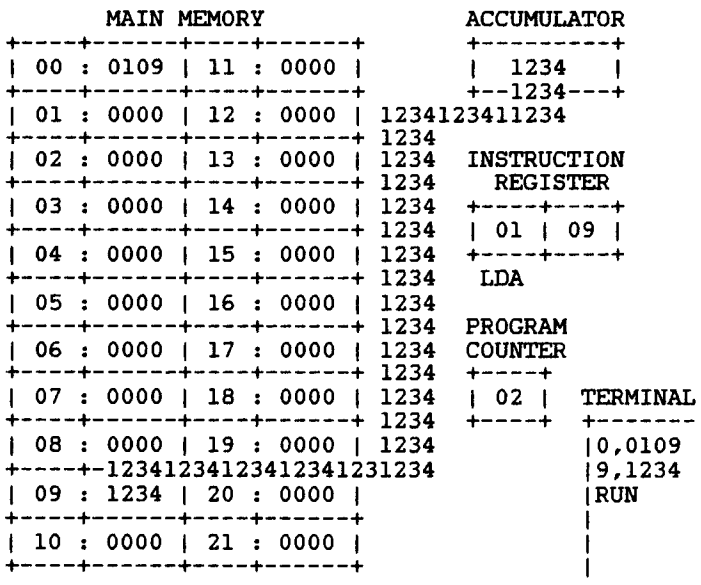
The contents of the three registers and the first 22 memory locations are displayed on the screen (see Figure 1 for the screen layout). Generally, most programs used in an introductory course need no more than the 22 displayed memory locations.

A boxed area in the lower righthand corner is used to display terminal input and output. Only basic operations of a computer terminal are carried out in this box. Short lines of information can be entered by the user or displayed as a result of computer operations. As new lines of information appear at the bottom of the terminal box, the existing lines scroll up and disappear from the top of the box. This is done by redrawing the entire contents of this area each time scrolling is performed. The terminal is used by the student to enter commands which control the operation of the computer, to enter instructions and data values into memory and to enter input and display output resulting from program execution.

Program execution in the animated computer consists of a simple execute cycle. Each step in the execution of an instruction is animated as fully as possible to illustrate what is occuring inside the computer. An execution cycle begins with an instruction being fetched from memory. The program counter is displayed in reverse video as well as the corresponding memory address to clarify which memory location is being accessed. The contents of the selected memory location are then moved to the instruction register.

All movement of data between memory and registers is animated by drawing and then erasing the data at successive positions on the display (see figure 2). This makes the data appear to move from the memory location to the register. The result shows clearly where the data came from and that it was copied to its destination rather than being moved.

The drawing and redrawing of data when animating works best at terminal speeds of 2400 baud or greater. 2400 baud is probably the optimal speed because the animation is fast enough to appear fairly smooth without being too fast for students to follow. At 9600 baud, the movements appear quite smooth but are too fast for most people to follow closely. No matter how slowly the terminal is operated, some capability is needed to control the animation speed. There are always times during a classroom demonstration or when a beginning student is using the computer



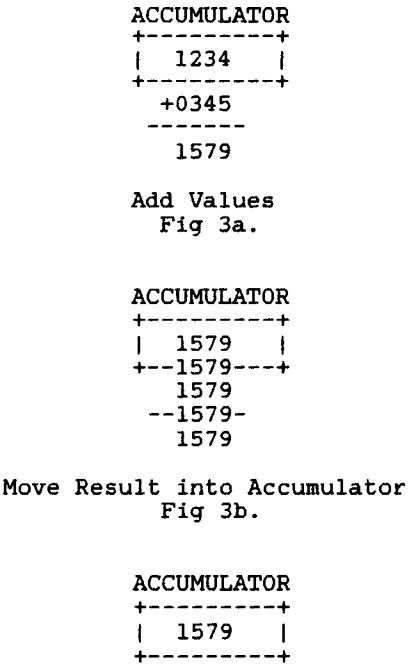
Animated Computer Data Movement
During Execution of a Load Instruction
Figure 2.

that the animation should be slowed down or temporarily halted.

There are two ways for controling the speed of the animation. One method is to utilize an operating system's wait facility to pause briefly between each redrawing of a piece of data being moved. The duration of this pause can be adjusted to slow down or speed up the animation. This method is subject to variations in pause time depending on the current response time when being used in a timesharing computer system. The other method is to output a series of characters to the display. The characters printed should be spaces so that they are not noticeable to the person watching the animation. The number of characters printed will determine the length of the pause. The duration of pauses in this method is highly dependent on the baud rate at which the terminal is being operated.

After fetching the instruction from memory, animation is used to show the incrementing of the program counter. An addition problem is displayed under the program counter which shows that 1 is added to the value in the program counter. The result is then moved up to and replaces the contents of the program counter.

Instruction decoding and execution is animated by breaking the instruction into its operation code and operand parts. The mnemonic name for the operation code is then displayed under that portion of the instruction register.



Accumulator Display for
an Add Instruction
Figure 3.

Execution of load and store instructions consists of moving data to or from the accumulator. Arithmetic operations are illustrated by moving a data value to a position just beneath the accumulator and drawing an arithmetic problem beneath the accumulator (see figure 3). After an appropriate pause to allow the student to examine the problem that has be set up, the answer is moved up into the accumulator. Animation of a jump instruction consists of moving the instruction's operand down to the program counter. Input and output instructions consist of moving data between memory and the terminal box on the display screen.

The commands for operating the animated computer are very similar to those used in the Basic language. Instructions and data are entered into memory by typing a memory address followed by the contents. This is similar to entering a Basic statement by typing a line number followed by the statement. The other commands are:

RUN - Begin program execution starting at memory location zero
BYE or EXIT - Exit from the animated computer
SCRATCH - Zero the contents of memory

LIST - Lists the memory contents on a nearby printer
SAVE - Saves a program on disk
OLD - Reloads a program from disk
CONT - Continues execution from where it stopped

When the SAVE and OLD commands are executed, a trap door opens on the side of the disk on the display screen. The file name is written next to this door and a stream of numbers move between memory and the disk to show the transfer.

There are three additional commands used for controling the speed of animation. Two of these commands (FAST and SLOW) are used to shorten or lengthen pauses at critical points in the animation (such as during arithmetic instruction displays). These commands operate in an incremental manner in which each time the command is issued, the animation slows or speeds up by one increment. The remaining command (STEP) is used to enter single step mode in which the animation halts after the execution of each instruction.

ANALYSIS OF THE ANIMATED COMPUTER'S USE

The animated computer has proven to be most useful in helping introductory students to comprehend the internal parts of a computer and how they operate. It gives the student a more tangible example of how memory works and how a computer processes instructions sequentially. A typical assignment might be to give the students a program and ask them to trace it both with and without the aid of the animated computer. Students are generally interested in the animated computer because of the novelty of the animation. This makes them more apt to spend time outside class using it.

The animated computer can also be used in class to help demonstrate the relationship between machine language, assembly language and higher level languages. This is done by showing how a simple Basic program can be compiled or rewritten in assembly language and then in machine language.

The amount of emphasis that instructors at Elizabethtown College have put on teaching the animated computer and on using it as a teaching aid has varied greatly from semester to semester. The variation goes from requiring students to play with the computer on their own without classroom lectures on it to spending several weeks teaching students to write programs for it. Experience has demonstrated that the animated computer is not a teaching machine but is just a teaching aid. It cannot replace classroom lectures. Students who are made to use the animated computer without benefit of lectures and armed only with a handout often get little out of the exercise and frequently are unable to understand it. The other extreme of teaching introductory students to program in the animated computer's machine language is a bit too much for students. Many introductory students struggle to learn to write small programs in Basic. Requiring them to write a machine language program is beyond their ability and is best left to an assembly language course where the animated computer serves well as a short introduction to the course.

A MICROCOMPUTER WORK STATION

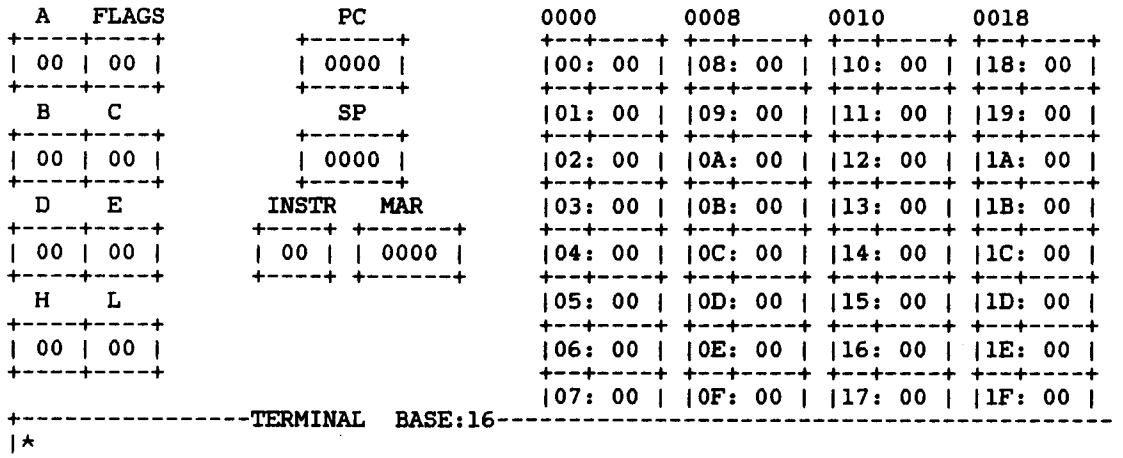
Single board microcomputer work stations are often used in assembly language programming courses. These stations are relatively inexpensive but usually provide only the capability for machine language programming. A computer simulation of one of these microcomputers is providing as many work stations as there are terminals on the timesharing system at Elizabethtown College.

The microcomputer simulator, called AN8080, is a simulation of the Intel Corporation 8080 microcomputer. With AN8080, as in the animated computer, the terminal's screen is used to display the contents of internal registers and memory and terminal input and output (see figure 4). When AN8080 is operating, the registers and memory displayed on the screen are updated at the end of each instruction. The movement of data between memory and the registers is not shown as in the animated computer. This is because AN8080 is meant for use by more advanced students who already understand the concepts illustrated by showing data movements. This movement would only slow the operation AN8080 to a point where it would frustrate the student.

Memory is displayed on the screen using four memory windows consisting of eight bytes each. The simulation can be configured to actually have as much memory as desired. The starting addresses for each of the memory windows can be set independently allowing four different sections of memory to be displayed simultaneously.

Programs can be loaded either by hand from the terminal or from a file. The contents of memory can also be saved in a file. Programs in memory can be modified from the terminal by loading new contents for memory locations. To facilitate programming, an assembler exists which generates object code which can then be loaded into memory. Object files use standard Intel hex format.

There are three speeds of operation available to the user. The default speed is slow because all changed registers and memory locations must be written to the

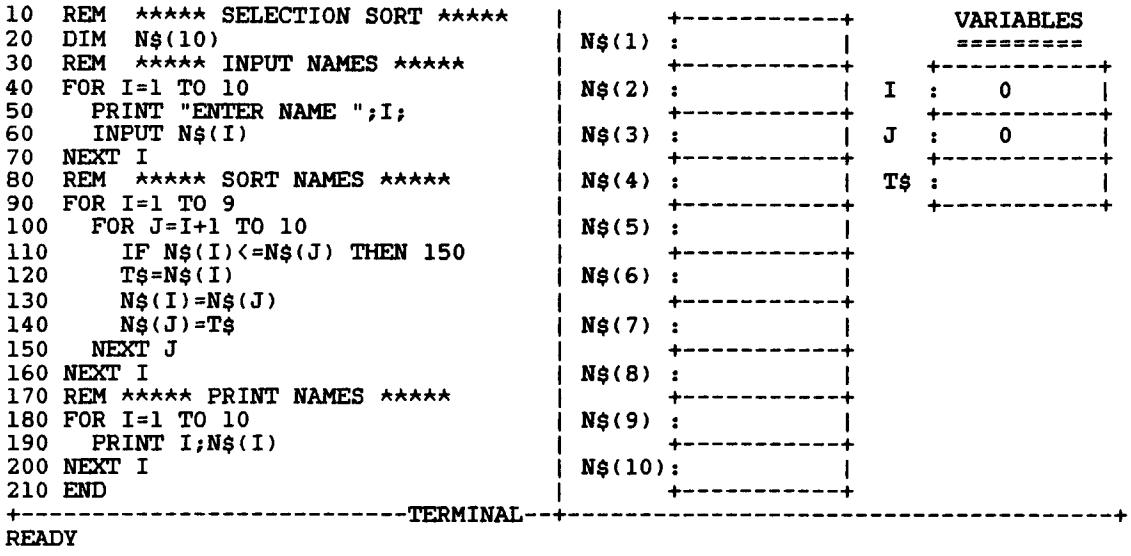


AN8080 Display Layout
Figure 4.

display screen after each instruction is executed. This speed varies according to the baud rate of the terminal. In the fast speed, only the program counter is updated during execution. Only when execution halts are all changed registers and displayed memory locations updated on the display screen. In the very fast mode, the internal parts of the microcomputer are not displayed at all and no updating of the display screen is ever done. Single step mode and program breakpoints are also available to aid in debugging.

The AN8080 program is really a framework for simulating microcomputers. The program consists of a command language interpreter, an animation driver, register and memory structure tables, and an instruction decoder/interpreter. It should be possible to modify the program to simulate any microcomputer by specifying the register structure and writing a new instruction decoder.

Several students have used both the AN8080 work station and the Intel SDK-85 computer. They emphatically prefer the AN8080 both for learning to program and for program development. One student's statement, "It's easy to think you are losing control when using the SDK-85 compared to the AN8080" sums up the comparison of the two.



Animated High Level Language Computer
Display Layout
Figure 5.

AN ANIMATED HIGH LEVEL LANGUAGE COMPUTER

Another animated computer that executes either Basic or Pascal language programs is currently being developed. In this computer animation, the Basic program being executed will be displayed on the left side of the terminal screen (see figure 5). The variables used in the program and their current values will be displayed on the right hand side. The bottom few lines of the terminal will be used to display terminal input and output.

Execution of a program in this animation will consist of highlighting the statement currently being executed and the variables being used in its execution. In the case of arrays, only the array elements involved will be highlighted. For assignment statements, the variable to receive the data will be flashed at the end of the instruction.

This animation should be very useful in teaching both sequential program logic and loops. It should also be helpful as a tracing tool and should help clear up the confusion about how variables work.

CONCLUSIONS

The animated computer provides an excellent teaching aid to help the introductory student understand the internal parts of a computer. The animated computer and animations in general are far superior to messy blackboard work when frequent changes must be made to illustrate continuous operations. The techniques of animation can be extended to advanced computer simulations for use by advanced students such as in the case of the AN8080 computer simulation. The animated high level language computer should provide the same benefits when teaching programming in high level languages.

Contact the author at the address given above for information on acquiring any of these programs.

ACKNOWLEDGEMENTS

My thanks to Elizabethtown College students Eric Luckenbaugh, for his help in writing the animated computer, and Kelly Williams, for his help in writing the Intel 8080 assembler. I also want to thank Barbara Tulley of the Elizabethtown College Computer Science Department for her helpful advice and opinions regarding the animated computer.