COMPUTER TECHNOLOGY

SIMPLE guidelines for developing a computer-based laboratory*

DORIS AARONSON and STEVEN BRAUTH
New York University, New York, New York 10003

Information relevant to setting up a computer-based psychology research laboratory is reviewed. The pros and cons of computer usage and of various hardware configurations are considered. Also, several approaches to program development are discussed. Guidelines for hardware and software development are put forth, and an example of a laboratory based on these principles is described.

Today over 50 companies manufacture small computers suitable for use in psychology laboratories (Sidowski, 1972). Researchers now must seriously consider the alternatives of a general-purpose computer lab, a do-it-yourself special-purpose automated lab, or a more traditional lab of stimulus-response equipment controlled primarily by the E. Decisions among these alternatives are often difficult because an integrated picture based on various criteria is hard to obtain. This paper outlines the thinking that led to the development of a computer-based lab and concludes with an illustrative example of one such working lab.

In the following sections, we attempt to answer questions frequently asked by psychologists who are newcomers to computer technology. Section I deals with hardware (equipment) decisions. They concern reasons for buying a computer and the amount of computer to buy. Section II concerns software (programming) decisions. They concern the functions a program must perform, who should write it, and the pros and cons of writing very specific, quickly codable programs or more complex but general-purpose languages. Section III describes our SIMPLE solution for these hardware and programming decisions.

I. HARDWARE QUESTIONS

1. Why Use a Computer?

Relative to conventional

laboratories, a computer laboratory can decrease experimental variability and increase flexibility in experimental design. Variability can be decreased in several ways: (a) First, equipment variability is reduced far below that for slide projectors, memory drums, and cam timers. The resolution of computer (crystal) clocks and the time to activate solid-state switches or to execute an instruction are generally a few microseconds, and the reliability is one or more orders of magnitude better than this. (b) Because the E and his differential social interactions can be removed from a fully automated laboratory, between-S variability is reduced. (c) S-paced trials, immediate feedback, and active on-line participation raise the motivational level and consequently decrease within-S variance. Further, within a session, the smooth-running automated system often permits a 50% larger data sample than does standard equipment. (d) Finally, variability due to E errors in setting knobs and recording data are eliminated.

Flexibility in experimental design occurs because event sequencing (e.g., feedback and stimulus ordering) and stimulus attributes (e.g., quantity, duration, intensity) can easily be varied under program control or can be made contingent on S's responses and latencies. For example, this facilitates using complex reinforcement schedules, linking designs, stimulus conditions based on individual performance differences, and rapidly converging threshold determinations. This flexibility allows rapid pilot work to determine the optimal experimental conditions and easy parametric variation within and between studies. Detailed criteria for flexible programmed auditory research have been discussed by Doorne and Sanders (1968) and by Nelson, Barry, & Perrott (1969). A detailed comparison of flexibility in conventional and computer-controlled

visual apparatus is presented by Cumming (1971).

Several disadvantages of a computer laboratory relate to time and money. If you are not prepared to deal with them, do not attempt a computer laboratory. Several companies now offer small laboratory computers for \$5,000-\$10,000, in the financial range of conventional equipment (Theis & Hobbs, 1971). It is heartening to note that such computers cost about \$25,000 in 1965 and are projected to cost about \$2,000 in 1975. However, these prices are for the economy model, and the "luxury" options on peripheral devices turn out to be necessities for most research projects. A workable computer laboratory may often cost in excess of \$20,000 in hardware, depending on the user's needs. A second expense is maintenance and repair service, generally from \$1,000 to \$5,000 per year. There are three options: a complete yearly service contract (parts and labor); individual service calls (generally slow and poor as companies encourage full contracts); and service by your own (highly skilled) shop personnel (or some combinations of Options 2 and 3). Finally, environmental changes are often needed in the way of temperature, humidity, electric power, and noise control.

A nontrivial disadvantage is that computers are time consuming. Far more time is needed in learning how to use them than conventional apparatus, and totally new programs and languages may take several months or more to get working. "Down time" should be less than for other apparatus, but the repair schedule is often not under your control. A final problem arises because computers are generally shared. Hence, research time must be scheduled in advance, may not be as much as you desire, or may be at odd hours.

So far, our discussion primarily concerns the two extremes of experimental control by a general-purpose computer or by a human researcher. There is a middle ground of purchasing computer modules and assembling them for your own special purposes. Uttal (1972) wisely cautions against building such special-purpose "noncomputers," although many researchers have done

*Preparation of this paper and the associated programming was supported in part by USPH Grant MH-16,496 to New York University. We thank Hollis Shapiro and Carol Geisler for helpful comments on drafts of this paper, and for help in testing SIMPLE user commands. Further information on SIMPLE can be obtained by writing the authors at NYU Psychology Department, 4 Washington Place, Room 858, New York, New York 10003, or by consulting Brauth & Aaronson (1972).

so quite successfully. Such an effort can be fruitful if (and only if) most of the following criteria are met: (a) Your research requires very special nonstandard components and circuitry, and a general-purpose computer would need extensive expansion or interfacing to accomplish your goals. (b) Your experimental operations occur in a fixed, predetermined sequence, and you do not need the flexibility of control contingent on S- or E-determined events from one trial to the next. (c) You plan a long sequence of experiments all using very closely related hardware designs. (d) You are skilled in electrical engineering and can make competent decisions on the specifics of equipment design and maintenance. If you are not, and your engineer leaves, you may have a major catastrophe. (e) You have available lots of inexpensive skilled technical labor and/or have received a gift of free surplus modules (which could turn out to be the classical "gift horse" if the other criteria are not met).

2. What Hardware is Necessary?

Laboratory computers fall roughly into two classes: small and medium size. These categories refer both to properties of the basic machine and to the number and types of peripheral devices it carries (Buchbinder, 1971a). The large high-speed computational machines of university computing centers are excellent for statistical analyses of large data matrices, precision parameter estimation (e.g., by iteration) for mathematical models, and simulation procedures for theory testing. However, these machines generally cannot compete with laboratory computers on inexpensive flexible capabilities for handling stimulus displays and for sensing responses. This division of computer functions creates a problem, as experiments are run on one machine while much of the data analyses may occur elsewhere. Links between computers can be made, but these links are sometimes difficult electronically, financially, and politically, depending upon the joint interests of the involved parties.

A "basic" small computer usually has a core memory of about 4,000 words of 12 or 16 bits each, plus a Teletype. Most research requires at least double the memory, at about \$2,000 to \$5,000 for each additional 4,000 words. Information goes in and out of a basic computer via the Teletype keyboard and printer and its associated low-speed paper-tape reader and punch (about 10 characters/sec). One immediately discovers that the Teletype is the weakest link in a

computer system, in terms of speed and durability. Its marginal and slow performance is intolerable for many purposes. Precision high-speed reader-punch combinations for paper tape with fewer moving parts (often photoelectric) sell for about \$5,000, and high-speed printers cost \$10,000-\$25,000 (Buchbinder, 1972). These permit you to accomplish in a few minutes what may take an hour on a Teletype. Instead, for about \$10,000 or more, you may obtain an IBM-style card reader-punch or various types of magnetic tape systems, including miniaturized cartridge systems.

For controlling experimental apparatus, one generally needs a programmable clock, a relay buffer or other switching device to activate stimulus displays, and a data collector to sense responses (\$2,000-\$5,000 together). Most basic small computers are not equipped with these items, although they do have the capability built in to easily connect and use them. Computer companies often sell such interface items as extras, offer to build them for your special needs, or sell you the modules plus free advice on putting them together (Mintz, 1972).

For most cognition laboratories, a computer-controlled visual display oscilloscope, plus its interface, is important (\$3,000-\$30,000 or more). Generally, pictures generated on these cathode ray tubes must be refreshed or intensified by computer command 30 to 60 times/sec. This may involve a nonnegligible amount of time and memory from a computer that should be doing other things. This computer-refresh problem is avoided by storage display scopes which maintain a picture continuously until an erase signal is received. However, most models currently available have several limitations on the amount of information displayed, its nature (characters, points, or lines), and its display parameters (intensity, scale size, position, brightness, timing). Some models erase by flooding the entire screen with light, which may not be desired for many perception experiments. For both types of scopes, decisions must be made about phosphor properties (Sperling, 1971b): color, phosphorescence decay time, efficiency (brightness), and durability (small patches may burn out when repeatedly pulsed). The standard TV phosphor, P-4, is reasonable for many psychology applications. It appears white (composed of yellow and blue, which might be separated with appropriate filters), decays to 1% of its initial intensity in less than .5 msec (i.e., a good tachistoscope, but also able to display much information

free of flicker), and its durability is relatively high.

Such small computers are fine for about six users with similar needs. For a larger or more diversified user group, a "medium-sized" computer might be considered. These are faster and have more bits/word (18-36) and more words in memory (16,000-32,000). They can control more peripheral devices, have a larger command set, and special-purpose registers. This leads to a number of programming advantages: larger program "pages" in memory, more direct (rather than indirect) addressing, greater arithmetic accuracy, greater microprogramming possibilities, and consequently shorter programs (Lode, 1972). Hence, a user can accomplish his purposes with greater precision in far less time than that required with a small machine. Medium-sized computers with appropriate interface circuitry make it possible for several Es to leave the computer almost completely set up for their experiments while others are running. Limited types of time sharing are sometimes possible. Such endeavors should be attempted cautiously; some pros and cons are considered in the next section. A final advantage of medium-sized computers is that they can often handle a useful FORTRAN-type of system, permitting some kinds of data analyses in the laboratory rather than at the computer center. In contrast, FORTRAN on basic small machines is rather limited in logical capability and in amount of space left in memory for the user's program and data.

The variety and quality of laboratory computers and display terminals are increasing and the prices are decreasing as the demand from both universities and industries mounts (Buchbinder, 1971b; Haavind, 1971). The behavioral scientist must carefully evaluate these products in light of his particular needs: no existing computer system fulfills the psychologist's dreams or illusions. Based on your present and future research needs, and your present and future finances, consider whether to purchase a small, medium, or large computer, to use someone else's (Sperling, 1971a), or to continue with conventional apparatus.

3. When is Time Sharing Useful?

So far, we have implicitly assumed "stand-alone" use of small- and medium-sized computers, i.e., only one user at a time. There are at least four ways to put a computer to multiple use, each with advantages and disadvantages.

(a) *Multiple subjects.* In some cases, the researcher may want to run several Ss simultaneously on the same single

computer program. This is quite possible, but with severe limitations. First, there would be restrictions on the stimulus and feedback displays. The Ss might all view the same stimulus display, or several separate devices might receive the same display signals, e.g., several TV-type video monitors or light panels wired in parallel. No complex response-contingent displays could be programmed, but a few feedback lights might be. Second, there would be limitations on the nature, amount, or arrangement of response data if space in the computer memory and peripheral storage devices were small. Instead, Ss might respond with paper and pencil or speak into tape recorders. Also, Ss might respond via switch closures sensed by a computer data collector or by multiplexed Teletypes. Most small computers may have insufficient storage to hold responses for all Ss for the entire experiment. A trial-by-trial punched output tape could be reentered after the experiment to sort out data for the separate Ss. Third, limitations on temporal aspects of the experiment would occur. Stimulus and feedback displays could not be S-paced, and precision reaction-time data might not be obtainable.

(b) *Foreground-background systems.* These systems, for medium-sized computers, permit a complex working experiment to receive machine priority as the foreground program, while a background user edits new programs or does statistics when the machine is not fully used by the foreground. If the foreground user has a high information load (e.g., a lot of time spent painting pictures on a scope), the background user might be locked out for several minutes at a time. Some foreground-background systems supplied by computer companies do not have adequate memory-protect features, so that the foreground program could possibly write data over the background user's program. Further, these systems sometimes require that foreground and background user schedules be identical: at the beginning of each user period, all devices (e.g., Teletypes, magnetic tape units, etc.) are assigned to foreground or background, and new assignments cannot be made until both user programs have ended. Finally, the amount of core memory available to the background user may often be severely limited.

(c) *Complete time sharing.* When several potential computer owners get together, they are often tempted to pool funds, time, and skills to obtain a well-equipped medium-sized time-sharing system. This is rarely

advisable (Uttal, 1972), although a few attempts have developed into very active and productive psychology labs. The experiments for which such efforts can save time and money in the long run are limited to those having both very low information rates and very weak timing demands: e.g., concept formation, problem solving, decision making, and instrumental learning. They do not include most experiments using scope displays, real-time speech, reaction times, or physiological data (EEG, EKG, etc.). Given the appropriate experiments, a bonus is often gained in data analyses. A time-sharing machine includes a lot of core, disk, and tape storage not generally needed on smaller stand-alone systems. This can make possible complete data analyses on the same machine that runs the experiments. A further disadvantage of time-sharing is that the system programs needed to coordinate the users take a long time to write (often 2 years), need costly extra hardware to operate, and are so complex that the psychologist must depend completely on a skilled programmer (resulting in major problems if he leaves). If the hardware goes down, all experiments are terminated. Two to five smaller machines could be purchased for the same total cost and would rarely all be out of commission at once.

Sometimes stand-alone research, e.g., complex visual scope displays and physiological response data, requires computer hardware almost suitable for time sharing. If these experiments are run less than 20 hours/week, it may pay to use some of the remaining time in a time-sharing mode if the disadvantages can be tolerated.

A very different time-sharing arrangement is a remote telephone terminal connected to a large computer center (Green, 1972). The terminal might include a Teletype and an alphanumeric storage display scope. The class of possible experiments is limited in the ways described above. However, in this case, the computer center is responsible for systems programming and maintenance. They generally have available high-level programming languages such as FORTRAN, BASIC, FOCAL, and LISP, and will provide magnetic tapes to store your data for later analyses. This might be the right thing for a new computer user with \$1,000-\$3,000 per year. Note, however, that remote terminal services vary drastically in quality. For some very busy services, the chances of getting a free telephone line and adequate core memory whenever you want it may be low. At prime hours, there may be long delays for the computer to respond, and the chances of system "crashes" or of

becoming disconnected may be high. Check with past users on these points before getting involved in any particular system.

(d) *"Off-line" machine sharing.* A small computer may run your experiments but may not have the memory space or the arithmetic precision to analyze your data easily. In this case, you may want to establish connections to use a medium or large computer when you are not actually running Ss, i.e., in "batch-processing" mode. Such connections are also useful (1) to generate your stimulus displays, e.g., when your machine has insufficient storage to hold all items simultaneously for randomization; (2) to compile or assemble a program too large to fit in your machine in "source" form along with the assembler (but a fine size for execution in binary form).

There are several ways to establish connections that vary in convenience and cost. Some large machines can be equipped with a low-speed Teletype or a high-speed paper-tape reader-punch that is relatively compatible with the laboratory computer. Compatibility factors include the tape quality, the physical spacing and number of channels of holes, and the logical code assigned to the characters (programming can often handle incompatibilities in codes or parity checks). Alternatively, the laboratory computer can be equipped with a card reader-punch, but this is often more expensive than the former option. If a magnetic tape link is used, compatibility in code format, bit-density, and number of channels must be considered. Sometimes slight misalignments of the magnetic read-write heads can cause problems for two tape systems that are otherwise logically compatible. Finally, the laboratory computer can be connected as a remote terminal to the large machine. If it is connected via hard wiring, one must consider the increases in electrical noise and decreases in signal strength with distance between computers. If it is connected via telephone lines, one must consider the nontrivial costs of phone bills, terminal rental, and the probability of being disconnected. In either case, details of buffering and timing the information transmission must be considered, but are now handled in standard ways at the large computer center.

II. SOFTWARE QUESTIONS

1. What Types of Programs Should You Write?

From the standpoint of the psychology computer user, there is a continuum of programming generality, flexibility, and expandability. This

continuum is related to the ease of coding experiments and to the "level" of coding at which the user specifies his experiment, ranging from numeric machine instructions to commands resembling English.

At the bottom of this continuum is the *special-purpose* program, generally coded in a combination of numeric and assembly languages. All information necessary to conduct the given experiment is contained in the specific program. Coding a single well-defined experiment is probably the quickest way to get computer-based research started. However, the effort invested in some of these programs may not be worthwhile, as their specificity may require time-consuming changes even in going from pilot to final runs of an experiment.

A moderate level of generality is provided by programs which accept *control data*. This gives the computer user the option of specifying some aspects of his experiment as "input data" rather than restricting him to an inflexible self-contained program. Possible input data might be the display content, its physical parameters, or its sequencing. For example, for different groups of Ss in a verbal learning task, the user might load at "run time" different sets of stimulus words, presentation rates, or accuracy criteria for task completion.

A higher level of flexibility is attained with a *subroutine package*. The user specifies his needs by selecting a subset of the available programmed subroutines. For example, an audio frequency scaling task might contain subroutines for generating random tones and for accepting category judgments from a keyboard. These may accept as control data the tone duration and the number of categories. Such subroutines might be selected from a general psychophysics package including stimulus subroutines for varying loudness or temporal spacing and response subroutines for collecting paired comparison or reaction time data.

The highest level of generality is the creation of *programming languages* for the user. Such languages should provide easy-to-use commands having mnemonics close to English or related to the user's research jargon. Two categories of user languages are compilers and interpreters. Compilers translate (and then eliminate) the user's program (the source program) into another language (the object program), which is directly decodable by the machine instructions of the computer. Interpreters do not engage in this translation process. Rather, they maintain in storage the user's

complete source program, which they scan, immediately interpreting and executing one instruction at a time. Compiler and interpreter commands are generally flexible and user-oriented. For example, CAN (Churchill, Naess, & Olivier, 1971), an interpreter for authors of computer-aided instruction materials, permits user commands in English sentences. In LEXIGRAPH, an interpreter for the PDP-4 (Forsyth, 1967), the psycholinguist might generate a techistoscopic display of one word at a time, halfway down the screen, with on and off times of 500 and 1,000 msec, as follows:

```
MODE Tach, LINE 10, TIME 500, 1000
TEXT (This/is/a/stimulus/sentence./)
```

With these "psycho-English" commands, the user does not have to worry about the details of setting up counters and clock registers, determining the coordinates of each point painted on the screen, initializing scope parameters, etc. Such high-level flexible commands are designed to produce broad classes of psychological experiments. Compilers and interpreters are easily expanded by adding additional commands and special-purpose subroutines for the user. If the original version of the language did not foresee all future experiments, it may be far easier to add a new command for a particular user than to generate a completely new program tailored for his single experiment. Haber (1970, 1971) has taken advantage of the widespread knowledge of the FORTRAN compiler to develop visual display systems that are easily modified.

Interpreters are generally far easier for the systems programmer to develop than are compilers. Also, they often have "interactive" features helpful to the user who has little computer background. As the user enters his commands from a Teletype (or a tape reader), the computer may signal when errors are made and the user can correct them on the spot. The user can start to execute his experiment or some pieces of it. If it runs into trouble, the user can modify his program on-line and immediately try again. When a final version is determined, a copy on punched or magnetic tape can be reentered in a few seconds for later execution. On the other hand, compilers are often superior to interpreters on two grounds: minimal amount of core memory occupied during execution and minimal time to execute the user's instructions. But compilers must receive an entire finished program before beginning to execute any of it. With a small machine based on

punched paper-tape input, the user must punch his program, read in the source tape to the compiler, wait for the object program to be punched, resubmit that to a machine assembler, which punches out a third version in binary format. Some 30 min later, the user can attempt to execute his experiment. It then may fail because of a missing comma. The user has no interactive editing options and must start the entire procedure again. Such difficulties with a computer have totally discouraged users before their first experiment was running. However, on a medium-sized computer with a disk, the time involved in these procedures is greatly reduced. Further, the larger core memory and disk may permit a system that combines features of a compiler and an interpreter.¹

2. What Commands Are Needed?

To program his experiment, the user needs commands to serve five classes of computer functions. These functions are briefly described below, along with some general principles for command format.

(a) To operate input-output devices.

The psychologist needs commands to transfer information into and out of the computer before, during, and after the experimental run. For example, at the start of a project, the user must type his program at the Teletype, edit it for format, logical, and design errors, and obtain tapes for quick reloading during experiments. Data might be put out in relatively raw form on the fastest possible device, so that many Ss can be run during "prime" daytime hours. These data can be reloaded to obtain summary statistics and readable listings during nonprime hours at nights and weekends, when human Ss are less available.

(b) To use memory storage buffers.

For the purposes of an experiment, the computer memory can be conceptualized as being partitioned into five special-purpose storage areas, or buffers: for the program, for input data (e.g., display content, parameter values, design information); for special-purpose registers (e.g., counters, clocks, limit values); for temporary working area (e.g., variable values needed later for branching decisions); and for permanent response output data. The researcher needs memory reference commands to store, access, and use these types of information in conjunction with the appropriate input-output devices during the course of his experiment.

(c) To perform arithmetic operations. At the start of an experiment, commands are needed to initialize special registers, data buffers,

parameter values, etc. As the experiment progresses, commands are needed to operate on these values: e.g., to index event counters, to sum response errors, to compare S response times to preset criteria, to generate random numbers, or to convert binary data to a more usable decimal form.

(d) *To control the flow of events.* Commands are needed to control the ordering and timing of events during an experiment, conditional on S responses or E restrictions. These are of two types: transfers and subroutine jumps. Transfer commands switch control directly to a remote (forward or backward) program location, and execution continues from there in serial order until another control command is encountered. Subroutine jumps, however, execute a group of commands out of order and then immediately return control to the next command listed after the jump. The same subroutine may be "called" or "entered" many times from many different program locations, thereby reducing the amount of code necessary for a given experiment.

(e) *To operate special devices.* Peripheral devices must be programmed to present stimuli (e.g., display scopes, switches controlling tones and lights), to sense responses (e.g., keyboards, switch closures activated by voice keys), and to time these stimulus and response events. Each command might control only one device or might control a combination of devices to serve one type of experimental design.

(f) *Command format.* Some decisions must be made about the coding of commands to be optimal for the inexperienced computer user and to be efficient for the computer to handle. From the user's view, commands should be as close to English as possible. From the machine's view, because interpreters maintain a program in memory in almost the original form, English sentences would rapidly tax the memory. One compromise is to represent each user command by the first letter of an appropriate English mnemonic. The memory space used is only one coding criterion. A second is the functional level of coding. Commands should be compatible with the way the psychologist conceptualizes his experiment. Each command must perform an experimental operation without burdening the user with details of executing that operation by the computer. For each user command, these details might occupy 5 to 50 machine assembly language instructions in the underlying systems program. A third coding criterion is the ease of combining commands. User

commands must be in appropriately sized "chunks" to build a large class of experiments and must have few constraints on order, other than those of the particular experimental design. A fourth coding criterion is simplicity of syntax within a command. Syntax should be minimal and relatively consistent for all commands. Users of the FORTRAN language know well the number of syntax errors resulting from missing or misplaced commas and parentheses.

3. What Programmers Should Be Used?

Three types of programmers are generally available to the psychology researcher: himself, his psychology students, and professional programmers. All of these might be used, but in different ways. The researcher will benefit by doing some of his own programming and by knowing programming details well enough to communicate his experimental needs unambiguously to his programmers. This is not a trivial task, and the time spent learning will pay off in the long run. Some researchers unskilled in programming have found out only after their experiment was run that their display contents were determined by a very nonrandom number generator and that equipment and program execution times were confounded with display duration or response latency. Some psychology students like programming, become very good at it, and will make fewer errors than professionals in translating verbalized experimental designs into computer code. However, the reinforcement ratio for programming is often higher than that for studying psychology, so that the student programmer may endanger his own career goals and must at times be guided. Good professional programmers are expensive, hard to find, and often difficult to communicate with. However, it pays in time and money to use them for complex jobs such as developing operating systems and user-oriented languages. Professionals are often valuable in getting research started with a new computer, and their experience enables them to set conventions for program compatibility. This pays off later in being able to combine quickly pieces of old programs to form a new experiment. Be sure that professionals provide complete documentation of their work with both program comment statements and English manuscripts. Without this, a new programmer or psychology user may have to start over from the beginning.

III. A SIMPLE ILLUSTRATION

SIMPLE is an interpretive programming language for conducting experiments on human information processing. SIMPLE, Scope Interpreter for Memory, Perception, and Learning Experiments, permits programmed control of audiovisual displays and also senses response times and errors from human Ss processing those displays. We feel that SIMPLE provides a good illustration of the guidelines put forth in the preceding sections on hardware and software.

1. SIMPLE Hardware

SIMPLE runs on a PDP-8I computer with 8,192 core locations, Teletype, crystal clock, and high-speed reader-punch (Fig. 1). The PDP-8I is interfaced with a Computer Communications Corp. CC-30 communications station. This includes a programmable visual display on a standard TV monitor (Sony, with P-4 phosphor), a hardware character generator (20 lines of 40 characters each), a cursor that can point to any one of the 800 screen locations by illuminating a short line under it (by program or keyboard control), an alphanumeric response keyboard, a controller, and a 1,024-word memory buffer for the characters and control functions. Displays are easily programmed using ASCII alphanumeric letter codes rather than point-by-point relative coordinates. The CC-30 scope is self-refreshed 60 times/sec until the PDP signals erasure, thus maintaining a bright picture without PDP time or memory. SIMPLE COMMANDS determine the following CC-30 display attributes: content of the elements (letters, numbers, special symbols), the number of elements (0 to 800, flicker free), the display duration, the interdisplay interval, and the display terminator (a fixed or random programmed time limit, or a response input from the human S). In addition, a 12-bit relay buffer was added to activate external displays such as lights, tape recorders, tactile displays, audio generators, movie or slide projectors.

SIMPLE senses and operates on the human S's response signals from the CC-30 typewriter keyboard (photoelectric). Responses (i.e., switch closures) are also sensed from auxiliary terminals substituted into the CC-30 keyboard interface, and from a 12-bit data collector directly connected to the PDP. Hence, many response sources are possible, including voice operated relays, telegraph keys, special pens, styluses, and pressure-sensitive writing boards. SIMPLE records these events as well as their latencies from the onset or offset of any specified stimulus display.

Fig. 1. PDP-8/I computer system used with SIMPLE at New York University.

With this hardware, SIMPLE permits the user to conduct a wide variety of experiments, for example: (1) learning tasks such as event prediction, free operant responding, or probabilistic concept formation; (2) memory and linguistic tasks such as recognition, recall, or running-span determinations; (3) perception tasks such as discrimination, monitoring, searching, vigilance, and reaction-time determinations.

2. SIMPLE Software

SIMPLE has four parts. The SERVICE PROGRAMS aid the researcher in preparing and editing tapes (1) of the user program for his experimental design and (2) for the text materials for his stimulus displays. The MONITOR receives up to 511 user commands from the Teletype or high-speed paper-tape reader, and interprets or decodes them into subroutine calls. The RUNNER executes these user commands to conduct on-line interactive psychology experiments. It controls the logical flow of events, and keeps track of where information is stored in memory. The COMMAND SUBROUTINES are the detailed assembly language instructions to drive

peripheral devices and to manipulate the logical memory registers under the user's control. The user's COMMANDS calling the command subroutines are formatted according to the four criteria considered in Section II2f: they are single-letter mnemonics plus a four-digit octal argument or argument string contained in parentheses. Below, we briefly describe the functional capabilities of SIMPLE's 14 classes of user commands. The command organization below roughly parallels Sections II2a-e above, although some commands do have secondary functions, spanning two or more sections.

(a) *Input-output commands: Buffer (B).* The input to buffer storage includes display contents and variable values needed to determine the specific experimental conditions for the given run. Output includes data on S responses, response times, and variable values. B and B+ read the input tape into a general input text buffer available to D (display) and R (read) commands, or into a special list buffer available to N (number) and P (position) display commands. Bnnnn waits for the S to type nnnn characters on the CC-30 keyboard, writes them into the output data buffer, and

displays them on the TV screen (unless suppressed by Pn). B* punches onto the paper tape the contents of the output data buffer and resets it for new data.

(b) *Memory commands: Read (R); position and put (P).* R commands read information from the input text memory buffer or directly from the user's program (already stored in memory) into special variable registers (available to other commands) or into the output data buffer. The characters to be read are generally formatted by a slash (/) as a termination signal to R. RC, RK, and Rn read items from the input buffer into the counter variable, the keypress variable, or the nth arithmetic variable ($0 < n < 7$), which are available, respectively, to the C (counter), K (keypress), and A (arithmetic) commands. R* and R() read items from the input buffer or from the user's program (i.e., enclosed within the parentheses of the R) into the output buffer for identification of experimental conditions. R:nn compares a character from the input buffer to the nnth value of the special response keypress array and skips the next command if they match, but not if they differ. R/ reads input and ignores it, e.g., when an alternative feedback message must be skipped.

Some P commands position the TV display beam for text selected by N (number) commands. Pn gets the vertical and horizontal coordinates from the left and right digit pairs of arithmetic register n, while PA modifies only the vertical values. P/ and P* erase from the CC-30 memory buffer the contents of the current line or the entire scope. Other P commands put information into new memory locations. P, PK, PC, PZ, and PN put information into arithmetic register 0, respectively, from the S's keyboard, the keypress variable, the counter variable, special register Z (which may contain response-time data), and the random number generator.

(c) *Arithmetic (A) and load (L) commands.* A commands operate on the eight variable arithmetic registers. Contents of these registers can be set to a constant (An:mmmm), negated (A-), or transferred between each other (A:nm). Also, the A commands permit program branching contingent on arithmetic register values, which may be based on previous response events or latencies. The next program command is skipped if two registers are equal (A?nm), if one is greater ($A > nm$), or if one equals a predetermined constant (A:/mmmm).

L (load) commands access the 64 keypresses and S response times (RTs) in the special keypress array, and permit arithmetic tests on these data, often in conjunction with the

Table 1
A Sample Simple Reading Time Experiment

PROGRAM	USER COMMENTS
E	Enter commands
A1 :1	Put 1 in register A1 to use for indexing A2
A2 :7634	Put -100 octal in A2 to count trials
L*	Clear and initialize keypress array
B	Read text tape into input buffer
R*	Read trial number into output buffer
C*	Set "word" counter to zero
RC	Read word count into counter variable
AK :	Set keypress variable to space, i.e., " "
D/	Display a word from the input buffer
K+	Wait for space key; record reading time
J (-1)	No key in 4 sec; wait again
J (+1)	Wrong key pressed and recorded; proceed
T*	Key pressed; reset K-timing counter
C/	Index "word" counter; skip if done
J (-6)	Sentence not finished; jump to display
D (*.../)	Sentence finished; display terminator
W1750	Maintain display for 1 sec octal
RK	Read the "correct" response key variable
D/	Display yes-no test question
K+	Wait for answer key (Y, N)
J (-1)	No key in 4 sec; wait again
J (+3)	Wrong key; jump to display
D (*RIGHT/)	Right key; display "correct" feedback
J (+2)	Skip over "wrong" feedback
D (*WRONG/)	Display feedback for error
W1750	Maintain display for 1 sec octal
B*	Punch data output tape
A;12	Index A2; skip if zero
J5	More trials left; start again
D (*THANKS/)	100 trials done; end experiment
W1750	Maintain display for 1 sec octal
F	Finish commands

One trial of input text tape: 1/6/10*READY/10#THE/10#DOG/10#BIT/10#THE/10#BOY/N 10**DID THE DOG BITE THE CAT?*/ etc. for 100 trials /\$

arithmetic variables, to determine branching conditions. The last RT (L/) or the *n*th RT (L > *nn*) can be compared to the combined contents of registers Z and A₀ (taken as one double precision register), and the next command is skipped if the RT is greater than those contents. Similarly, the last keypress (L) or the *n*th keypress (L:*nn*) can be compared to the keypress variable, and the next command is skipped if they match. L_{nn} loads the *n*th RT into the combined registers Z and A₀ if further arithmetic manipulations are desired, and L* clears all values in the keypress array (but not the output data buffer).

(d) *Control commands*: Jump (J); counter (C). Some commands (e.g., keypress) have response-contingent branching to particular locations. Frequently, the command at those locations is a JUMP to yet another program location for arithmetic operations or feedback displays. Also, J is helpful in repeating or looping over a set of commands several times. Program control can jump unconditionally to line *nnnn* (J_{nnnn}), or to a line *n* commands away from the current command J(*±n*). J:*nnn* permits a jump to a remote subroutine and back again to the next user command when the subroutine is completed.

The J and C commands are often used together to count experimental events and branch contingent on such counts. The counter is incremented by one each time C_{nnnn} or C/ is executed, and the next command is skipped when the counter equals, respectively, the value of the constant *nnnn* or of the preset counter variable (using RC or A12). C* resets the counter to zero for the next use.

(e) *Special device commands*: Wait (W), expect (X), display (D), number (N), keypress (K), time (T), interface (I). These commands control experimental peripherals: the clock, the CC-30 display and keyboard, the data collector, and relay interface. Alphanumeric text can be displayed on the scope with the predetermined contents taken from the input text buffer (using D_{nnnn} or D/), from the program (using D()) with the text enclosed in parentheses, or from the S's keyboard response (using D). Also, displays can contain items from the special list buffer selected from a predetermined position (N), selected randomly with replacement (N_{nnn}) or without replacement (N:*nnn*).

Display duration can be determined by W_{nnnn} (Wait) which initiates a program pause for *nnnn* msec. Also, display duration (among other things) can be determined by the

programmable interrupt (X_{nnnn}). When responses are expected over some time interval, this activates the clock, skips the next command, and executes subsequent commands for *nnnn* sec. During this time, all response inputs from the data collector are timed. After *nnnn* sec, control returns to the command following the X. However, if T* is encountered, before *nnnn* sec elapse, the clock is turned off and the program continues in order rather than branching back.

S responses and their latencies are obtained with the K (keypress) and T (time) commands. K commands sense the CC-30 keyboard (or any devices substituted into that interface) and permit program branching dependent on elapsed time and whether the response matches a predetermined character. The T commands sense responses from the NYU data collector. For the K and T commands, responses and RTs are written into the output data buffer and also into the special keypress array. Subsequent program branching can be made, based on the contents of this keypress array, using L commands described above.

The In_{nnn} (interface) command sets a pattern of 12 relays according to the binary equivalent of the octal number *nnnn*. In this way, auxiliary equipment such as tape recorders, lights, etc., can be sequenced and timed under program control.

3. A Sample SIMPLE Reading Time Experiment

In order to illustrate how the above commands can be combined, we list in Table 1 an actual experimental program. The left column contains the program read by the PDP-8I and the right column contains descriptive comments for the psychologist. This is a 100-trial experiment to obtain S-paced word-by-word reading times for sentences displayed on the CC-30 scope. At the start of each trial, "READY" is displayed and then S presses the space-bar to display each successive word, while the computer clock times his display durations. After each sentence, a YES-NO question is displayed to test S's comprehension. His response time to press Y or N is recorded, his response is compared to the correct key, and feedback on accuracy is displayed. Before each trial, some pre-punched text tape is read into the input text buffer. In this example, it contains the trial number to be written in the output buffer, the number of sentence words (six, including READY) to set for the counter variable, the actual words to be displayed, the correct answer key (N), and the test question.

The slashes (/) are block-terminators for the input buffer, and the * and # are scope format characters to clear the entire screen or correct line. Indeed, SIMPLE is simple for both the computer and the researcher.

REFERENCES

- BRAUTH, S., & AARONSON, D. SIMPLE: Scope interpreter for memory, perception and learning experiments. DECUSCOPE, 1972, in press.
- BRYDEN, J. E. Visual displays for computers. Computer Design, October 1971, 55-77.
- BUCHBINDER, H. G. (Ed.) Minicomputers and peripherals: Blind date-happy marriage. Digital Design, 1971a, 1, 32-35.
- BUCHBINDER, H. G. (Ed.) Minicomputers: What happens when technology changes? Digital Design, 1971b, 1, 36-39.
- BUCHBINDER, H. G. (Ed.) Computer-driven printers: A survey of types, how they perform, and what's available in the marketplace. Digital Design, 1972, 2, 15-29.
- CHURCHILL, S., NAESS, L., & OLIVIER, W. P. CAN-4, An advanced author language for CAI, computer-based testing and psychological experimentation: PDP-9 implementation. Behavior Research Methods & Instrumentation, 1971, 3, 95-99.
- CUMMING, G. LINC-8 presents and controls visual experiments. Behavior Research Methods & Instrumentation, 1971, 3, 24-29.
- DOORNE, H. V., & SANDERS, A. F. PSARP, A programmable signal and response processor. Behavior Research Methods & Instrumentation, 1968, 1, 29-32.
- FORSYTH, D. Lexigraph. Harvard Psychology Computer-Based Laboratory Document No. 24, November 3, 1967.
- GREEN, B. F., JR. The use of time-shared terminals in psychology. Behavior Research Methods & Instrumentation, 1972, 4, 51-54.
- HAAVIND, R. C. (Ed.) Man's window to interactive computing. Computer Decisions, 1971, 3, 10-15.
- HABER, R. N. An alphanumeric display program for a PDP-8. Behavior Research Methods & Instrumentation, 1971, 3, 141-145.
- HABER, R. N., BARRY, S. H., & UHLMAN, T. On-line Fortran for the PDP-8. Behavior Research Methods & Instrumentation, 1970, 2, 248-252.
- LODE, T. 18-bit vs. 16-bit small computers: How to choose the right one. Digital Design, 1972, 2, 34-41.
- MINTZ, R. S. Simple processors: Interface link in a minicomputer control system. Digital Design, 1972, 2, 42-45.
- NELSON, M. A., BARRY, S. H., & PERROTT, D. R. Automated system for acoustic research. Behavior Research Methods & Instrumentation, 1969, 1, 298-299.
- SIDOWSKI, J. B. Various uses of minicomputers in psychology. Behavior Research Methods & Instrumentation, 1972, 4, 43-50.
- SPERLING, G. Computer parasites and hosts: Practical advice on how to be a successful parasite at your host's computer installation. Behavior Research Methods & Instrumentation, 1971a, 3, 147-148.
- SPERLING, G. Flicker in computer-generated visual displays: Selecting a CRO phosphor and other problems. Behavior Research Methods & Instrumentation, 1971b, 3, 151-153.
- THEIS, D. J., & HOBBS, L. C. The minicomputer revisited. Datamation, 1971, 10, 24-34.
- UTTAL, W. R. Misuse, abuse, overuse, and unuse of on-line computer facilities by psychologists. Behavior Research Methods & Instrumentation, 1972, 4, 55-60.

NOTE

1. Such a visual display language, NYUSHO, contains an efficient and flexible combination of interpreter and compiler features. It runs on a medium-sized PDP-15 computer with disk, magnetic tapes, an independent scope-processor, and 16,000 words of core memory. NYUSHO was developed by Miriam Kaplan and Abhinav Dwivedi and was an outgrowth of George Sperling's GSLetters.