History of CRMS APL

Paul McJones
paul@mcjones.org

Draft of 18 July 2025
To appear in *IEEE Annals of the History of Computing*

Abstract

The Center for Research in Management Science (CRMS) at the University of California, Berkeley pioneered the use of interactive computerized multi-person simulation experiments for social science research. The facility, referred to as the Management Science Laboratory, was conceived by Professor Austin Hoggatt and included a computer system with linked DEC PDP-5 and PDP-8 computers. An advanced APL-based version was designed and built during 1972–1976. By January 1974 it could support one experimenter developing and running an experiment connected to multiple terminals. By 1976 the system was complete, with an easy-to-use interactive APL subsystem, and a multi-user time-shared operating system. The project was a technical success, making it possible to write experiments, debug them, run them, and analyze the results. It established the Management Science Laboratory as the prototype for experimentation in economics and behavioral science. However the lab failed to jell as a community of experimenters and support personnel, and the hardware and software quickly became obsolete.

This is my attempt to reconstruct the full project history. I was a project member during 1972. Reference [24] is an archive of surviving project-related documents and source code listings, which are expected to become available in digitized form through the Computer History Museum.

Introduction

UC Berkeley professors Frederick Emory Balderston [23] and Austin Curwood Hoggatt [7] were pioneers in the use of computer-based simulation in economics. Hoggatt's Ph.D. thesis [12] was said to be "the first thesis in the United States that used simulations with a human-to-computer interface." In 1962 the two published a monumental monograph [2] on a (noninteractive) simulation of the lumber industry, which included 35 pages of FORTRAN II source code.

Management Science Laboratory

In 1964 Balderston and Hoggatt began planning the Management Science Laboratory. In a 1969 paper [14] demonstrating its use, Hoggatt described it as follows:

> The Management Science Laboratory was conceived by faculty in the Center for Research in Management Science at the University of California at Berkeley as a facility for behavioral research. A facilities grant from the National Science Foundation was awarded to the Center, and an interest group was formed to design and construct a laboratory for experiments with human subjects in situations based on models from the behavioral disciplines.
>
> Our thinking was guided by results from a prototype laboratory and by the emerging literature in experimental gaming. Two principal considerations were that the environment should not detract from the experiments and, whenever possible, automatic control should be substituted for human control over experimental variables. The use of a digital computer as the heart of the experimental control system would achieve the generality which was required. The design of the laboratory was given over to architects who consulted with the faculty, while the computer control system was designed by technicians coordinated by one faculty member.

The lab occupied about 2800 square feet, and much of the space could be reconfigured with wall sections suspended from overhead tracks; it was usually set up with a cubicle for each experimental subject. There were connections for terminals (originally Model 33 Teletypes) and video cameras and monitors.

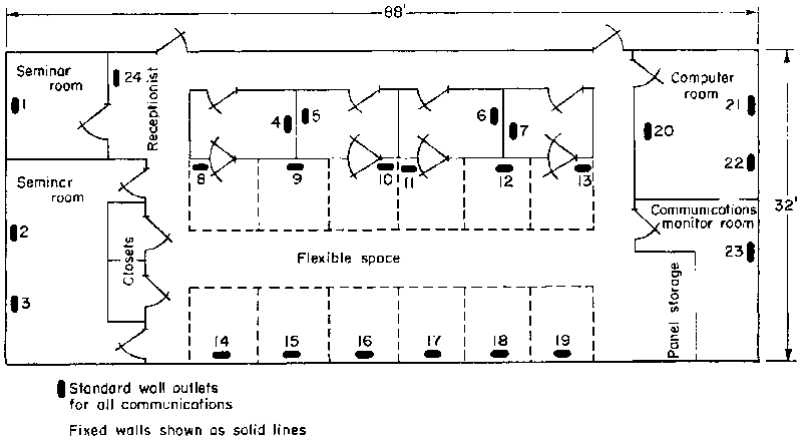

Figure 1. Floor Plan of the Management Science Laboratory. From [14].

Hoggatt described the computer system: [14]

> At the center is a time-shared PDP-5 with DECTAPES, IBM compatible tape, full-duplex teletype multiplexor, analog-digital digital-analog conversion, micro clock, voltage out (one shot) and sense channels and a memory-to-memory link to a PDP-8 computer. The PDP-5 is quiescent, responding to a 100-millisecond interrupt to update its clock or to a demand for service from one of the devices which it serves. On demand the PDP-5 slavishly responds or transmits to a peripheral device or stores data. The PDP-8, which has extended arithmetic, operates on a variable quantum break of 10 milliseconds up to 100 milliseconds and can be tuned to match the environmental characteristics of any given experiment. Control over the experiment resides in the PDP-8 which can issue

commands to the PDP-5 to send character strings or operate external devices. For example, in creating a log of the experiment the PDP-8 may select characters from the input string of a teletype, add control characters and the real clock time to the string and have the PDP-5 write this information on the DECTAPE.

Figure 2. PDP-5/PDP-8 circa 1973. Photo by Paul McJones.

Hoggatt described in detail a duopoly experiment using the system in [15]. The computers were programmed using a language ECL/3 (Environmental Control Language, version 3) designed by A. Hoggatt, D. Steingart, J. Schlesinger, and J. Moore.[1]

Researching APL

Apparently, the PDP-8 had been added to the system in 1967. [17] Even still, the combined PDP-5/8 system was small and slow. By 1970, there was interest in replacing it with a more powerful computer system. Hoggatt had become an advocate of the APL programming language, which IBM had released for the System/360 in 1968. [6] APL offered a concise, powerful notation that Hoggatt believed would be appropriate for the experiment control programs (via extensions to handle files, multiple terminals and other devices), analysis of the resultant data, and as a publication language for the models underlying the experiments. [16]

Two students who had contributed to the software for the PDP-5/8 system, David S. Steingart from Computer Science and Jeffrey Moore from Business Administration, joined forces with Computer Science graduate student Rodnay Zaks to explore an APL implementation on a microprogrammable computer. [32] The section of their paper entitled "A Time-Sharing System for Behavioral Science Experimentation" lays out detailed requirements for a system to be used for experimentation. Processors were evaluated and an emulator and other tools were written in Snobol. A Digital Scientific META 4 Series 16 processor, [5] memory boards, and I/O boards

[1] ECL/3 is described in [22]. For a large example, see Appendix IV of: Mark B. Garman, "Trading Floor/1: A Prototype of an Automated Securities Exchange," Institute of Business and Economic Research, Working Paper No. 7, Graduate School of Business Administration, University of California, Berkeley, July 1972. Available: https://crms-apl.computerhistory.org/#Garman1972

were acquired, with a memory mapping board being contracted out. A complex dynamic interpreter was designed and partially implemented, but the system was not completed.[2]

CRMS APL begins

By winter of 1971–1972 a new project was begun to build a complete computer system. The project was administered by Balderston, chair of CRMS, and by the Laboratory Advisory Committee, chaired by Hoggatt. An announcement by Balderston noted the project would be "headed by Mark L. Greenberg (PhD, Electrical Engineering and Computer Science, UC-Berkeley, 1971) and Jeffrey Moore (doctoral student in information science, Graduate School of Business Administration, UC-Berkeley)."[3]

By April 1972, Charles A. Grant (also a 1971 Berkeley EECS PhD) returned from a post-doctoral fellowship and joined Greenberg as co-technical leader; Moore continued for a time as a consultant and continued to supervise the PDP-5/8 system.[4] One of the next hires was George Morrow, [20] to work on hardware. Morrow had dropped out of high school, but returned to school in his late 20s, earning bachelor's and master's degrees, and entering the Ph.D. program in mathematics at Berkeley. David Redell and I also joined the staff at this time. Redell was in the Computer Science Ph.D. program, and I had just completed my B.S. in Engineering Mathematics; we had worked together on the newly terminated CAL Timesharing System at Berkeley's Computer Center. [21]

Balderston and Hoggatt played complementary roles, with Hoggatt focusing on detailed requirements and technical approaches and Balderston securing institutional support. Both believed strongly in the Laboratory, incorporated it into their personal research agendas, and were eager to spread the ideas.

Hardware

The META 4 processor used by Zaks' investigation was originally designed to emulate an IBM 1130, but its read-only control store could be reprogrammed. [5] Given the limited size of this control store (2048 instructions), it was decided to use two META 4s sharing a single main core memory. One processor would be dedicated to APL while the other executed a general-purpose instruction set as well as controllers for I/O devices. The general-purpose processor would run a timesharing system for program development, supervising the APL processor, etc. (Grant and Greenberg had sketched how a suitable microprocessor could be used to implement a CPU and I/O devices in a 1971 paper. [9]) Perhaps to reassure Balderston and Hoggatt, the system programming language for the second machine was named SIMPLE. In addition to the two

[2] [32], [33], [26], and: R. Zaks, Personal communication, 14 October 2024.
[3] F. Balderston, "Press Release," *bus ad bulletin*, Schools of Business Administration, Volume 7, Number 4, 26 January 1972. [25]
[4] Center for Research in Management Science, "Annual Report," 1971-72. [25]

processors and shared memory, interfaces for serial lines,[5] printer, disks,[6] and tape drives were built. Computer Science Professor Martin Graham was on the Laboratory Advisory Committee and consulted on a few electrical problems such as ground loops.[7]

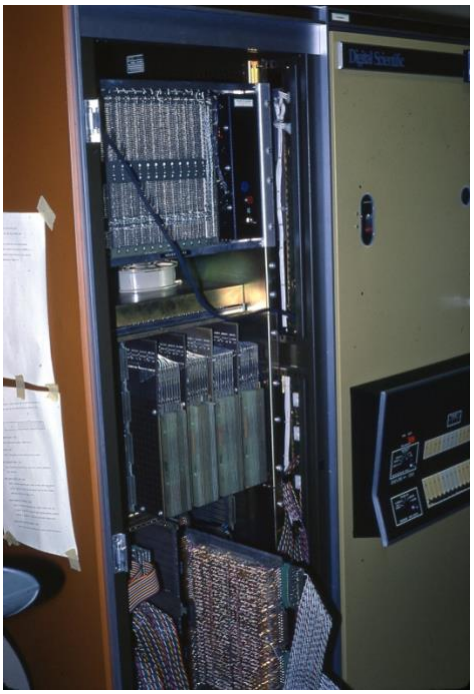

Figure 3. CRMS META 4 system circa 1973. Photo by Paul McJones.

Microcode

The original plan was to use the APL microcode from Zaks' investigation. My initial assignment was to get it running, but I had problems understanding the design from its source code and limited documentation. I proposed to do a new design based on static rather than dynamic name localization, which would allow parsing at compile time rather than runtime. There would be a virtual machine with a set of instructions for a subset of a dialect of APL. APL pioneered array-valued expressions: most functions accept operands that are vectors, matrices, or higher-rank arrays. Scalar functions such as addition or multiplication are generalized elementwise if both operands are arrays with the same shape (set of dimensions); if one operand is a scalar and the other is an array, the scalar is applied to each element of the array. Other APL primitive functions include reshaping an array (e.g., converting a matrix to a vector with the same elements), catenating or joining arrays, transposing a matrix, and many more. Because the control store of the META 4 processor was so small, a subset of the primitives had to be chosen, with the remainder implemented using that subset. There were a few other changes: [10]

[5] G. Morrow, "Serial Line Input/Output System," Systems Group, Center for Research in Management Science, 24 July 1972. [24] Available: https://crms-apl.computerhistory.org/#Morrow1972

[6] M. L. Greenberg, "Disk Interface and Controller Reference Manual," Technical Document, Systems Group, Center for Research in Management Science, 12 July 1972. [24] Available: https://crms-apl.computerhistory.org/#Greenberg1972

[7] C. A. Grant, Personal communication, 18 September 2024.

- User-defined functions could be defined with up to 15 extra parameters; at runtime, any missing optional parameters would be replaced with an "undefined" value that could be detected using a built-in function. (Zaks had used a similar scheme.)
- A function parameter could be declared to be "by reference," allowing the function to assign a new value to the corresponding actual parameter variable.
- Arrays could have elements of mixed type: character, integer, and float-point. This turned out to be useful with interprocess mailboxes -- see APL Runtime Services below.

My plan was approved, and I started to design. There was a preliminary machine specification by July;[8] the microcode was complete, documented, and tested by February 1973[9,10,11] Given that the projected users would not be numerical analysts, his recommendation was to implement decimal floating-point. I was concerned that this would occupy too much control store, so instead implemented 32-bit binary floating-point. I incorporated correct rounding using a "sticky bit" based on Kahan's lectures.[12]

Since the SIMPLE machine wasn't running yet, cross-development was done on an SDS 940 computer running the Berkeley Timesharing System. [19] There was an assembler for the META 4 microcode and a simulator with debugging features (e.g., single-stepping and examining registers). We may have started out using the original Project Genie [30] SDS 940 in Cory Hall, but we soon switched to the machine belonging to Resource One, which was part of Project One, an intentional community in San Francisco. [31]

I left the project after delivering the APL microcode but learned that when the microcode was installed in the hardware read-only memory (which involved manually peeling little foil squares from printed circuit boards), it worked the first time. The Appendix lists four changes that were made later. Around 1974 Grant contacted me for help in understanding an "anomalous" arithmetic result: dividing by 10 and then multiplying by 10 in some cases gave a number differing by one: an example of (correct) rounding.[13]

The two machines were usually referred to as the AIPU and SIPU (IPU = Instruction Processing Unit). Grant designed and implemented the SIPU.[14]

---

[8] P. McJones, "Preliminary Reference Manual for the CRMS System APL Processor," Technical Document, Systems Group, Center for Research in Management Science, 12 July 1972. [24] Available: https://crms-apl.computerhistory.org/#McJones1972b

[9] P. McJones, "APL\META 4 Microcode," Digital Scientific META 4 Series 16 assembly language, 28 January 1973. [24] Available: https://crms-apl.computerhistory.org/#McJones1973a

[10] P. McJones, "Test programs for CRMS APL Processor," 1972-1973. [24] Available: https://crms-apl.computerhistory.org/#McJones1973b

[11] P. McJones, "CRMS APL Processor Reference Manual," Technical Document, Systems Group, Center for Research in Management Science, 2 February 1973. [24] Available: https://crms-apl.computerhistory.org/#McJones1973c

[12] P. McJones, "Meta APL Floating-Point Arithmetic [design notes]," 13 June 1972. [24] Available: https://crms-apl.computerhistory.org/#McJones1972a

[13] [C. Grant and P. McJones,] [Example of floating-point rounding, circa 1974.] [24]Available: https://crms-apl.computerhistory.org/#GrantMcJones1974

[14] C. A. Grant, Personal communication, 18 September 2024.

SIMPLE operating system

In early discussions it was decided to build a message-based system, with support in the SIPU microcode for basic process scheduling and message sending. Rather than switching between user and supervisor mode, there would be a kernel process with extra privileges to handle system activities that could not be handled in microcode. I/O devices would appear to software as regular processes sending and receiving messages, so there would be no interrupt mechanism. An early working document by Redell described messages as being sent to a (process, port) pair and described fairly complicated mechanisms to be implemented in microcode.[15] A later working document described a much simpler mechanism for the microcode, but noted that it was sufficient to enable a trusted "kernel" process to implement flexible message channels.[16] The microcode actually implemented included a basic process scheduler and a test-and-set instruction.[17]

A short document from 1974 called Claims Manager demonstrates several aspects of the operating system design.[18] The Claims Manager provided voluntary shared/exclusive locking for objects. It was implemented as a server process accessed via messages. A request message included a command word, a port capability for the response, and a capability for the object to be locked. Capabilities were unforgeable references to operating system objects such as processes, memory segments, message ports, terminals, and open files.

APL processes communicated by sending and receiving messages through mailboxes. There was no microcode support for this; multiplexing of the APL processor was performed by APL Runtime Services running on the SIPU, which could perform start and stop commands and access data structures in the shared memory.

Language Services

Early design work envisioned what is now known as an Integrated Development Environment (IDE) combining editing, compiling, and debugging within a single command-line interface. Furthermore, it was proposed to share many underlying components between the SIMPLE and APL language systems. Text editing was based on a line-oriented file structure with printing

[15] D. Redell, “Interprocess Communication,” Working Document, 20 July 1972 [24] Available: https://crms-apl.computerhistory.org/#Redell1972a

[16] D. Redell and Paul McJones, “CRMS Process Synchronization Mechanism,” 11 November 1972. [24] Available: https://crms-apl.computerhistory.org/#RedellMcJones1972

[17] C. A. Grant, Personal communication, 18 September 2024.

[18] P. Glassco, “Claims Manager,” Technical Document, Systems Group, Center for Research in Management Science, University of California, Berkeley, 5 July 1974. [24] Available: https://crms-apl.computerhistory.org/#Glassco1974

terminals. Editing commands consisted of a two-letter command name followed by one or more expressions typically designating a line or range of lines. Early design memos give details.[19,20,21]

As the system evolved, it was decided that the APL interface should follow standard APL\360 conventions as closely as possible for editing, system commands, and so on.[22] The SIMPLE interface retained the general form sketched in the early design memos, but was refined with a rich set of editing, debugging, and performance measurement commands.[23]

A reference manual for the SIMPLE language survives.[24] A brief document lists the proposed APL syntax;[25] appendices of "Writing Behavioral Experiments" (see below) list primitive functions not yet implemented as well as new CRMS primitives. Listings of the source code for the SIMPLE and APL language services (but not compilers) survive.[26,27]

APL Runtime Services

The document "Writing Behavioral Experiments in CRMS APL: Programmer's Manual" by Gee et al.[28] explains how an experiment program was organized and describes the application programming interfaces. An experiment consisted of a set of cooperating processes: isolated instances of running programs. The experimenter's process could start subject and robot processes by calling a provided function. Each subject process had an attached terminal and was connected via a pair of mailboxes to the experimenter's process; no variables were shared between processes. A mailbox allowed one-way transmission of a sequence of APL objects (scalars, vectors, or matrices) between two endpoints. The endpoints could be processes, files, and timers. An experiment could include one or more robot processes. A robot process substituted for a person at a terminal and thus interacted with the corresponding subject process.

---

[19] D. Redell, "CRMS APL/SIMPLE Integrated Language Processing System: Design Considerations," 22 July 1972. [24] Available: https://crms-apl.computerhistory.org/#Redell1972b

[20] D. Redell, "Language System: Interim Command Language," 5 December 1972. [24] Available: https://crms-apl.computerhistory.org/#Redell1972d

[21] D. Redell, "Preliminary Language System User Manual," 2 February 1973. [24] Available: https://crms-apl.computerhistory.org/#Redell1973a

[22] D. Redell, "Proposal for Final CRMS APL Interface—APL Subcommittee," 6 August 1974. [24] Available: https://crms-apl.computerhistory.org/#Redell1974

[23] M. Greenberg, "SIMPLE Language System Terminal Commands Reference Manual," Technical Document, Systems Group, Center for Research in Management Science, University of California, Berkeley, May 1975. [24] Available: https://crms-apl.computerhistory.org/#Greenberg1975

[24] M. Greenberg, "SIMPLE Language Specification Reference Manual," Technical Document R1, Systems Groups, Center for Research in Management Science. 26 July 1973; revised 5 September 1973. [24] Available: https://crms-apl.computerhistory.org/#Greenberg1973

[25] P. McJones, "CRMS APL Syntax," 30 October 1972. [24] Available: https://crms-apl.computerhistory.org/#McJones1972c

[26] ?, "SIMPLE Interactive Language System source," SIMPLE source code, undated, circa 1974. [24] Available: https://crms-apl.computerhistory.org/#Xx1974a

[27] ?, "APL Runtime Supervisor source," SIMPLE source code, undated, circa 1974. [24] Available: https://crms-apl.computerhistory.org/#Xx1974b

[28] P. Gee, W. Greiner, S. Linker, and D. Redell, "Writing Behavioral Experiments in CRMS APL: Programmer's Manual (Preliminary Version)," Technical Document, Systems Group, Technical Document Center for Research in Management Science, University of California, Berkeley, 15 July 1974. [24] Available: https://crms-apl.computerhistory.org/#GeeEtAl1974

The code to implement these facilities included the APL Runtime Supervisor, which was written in SIMPLE, and also a set of library functions (as described in Gee et al.) written in APL.

Wiley Greiner, who had extensive previous experience programming in APL\360, joined the project shortly after I left and led the effort to implement the full set of APL primitives in terms of the provided subset.[29] Sheldon Linker contributed to this effort and also worked on the plotting package,[30] assisted in writing experiment programs, and helped run experiments.[31]

Greiner's masters thesis[32] compared two approaches for generalizing scalar expressions to arrays—"vertical" (array-at-a-time) and "horizontal" (element-at-a-time)—and demonstrated that the facilities provided by the APL processor provided an effective foundation, mostly using the "vertical" approach.

Development progresses

As mentioned previously, design and implementation began in 1972. In a departmental meeting on November 22 of that year, Balderston discussed the new system, saying that sometime in 1973 APL was expected to be running.[33] He also noted the new system would have the capacity to be used for general time-sharing (e.g., document preparation and general interactive APL usage) in addition to the behavioral experimentation which had been going on for almost a decade.

By July 1973, most of the hardware was complete (core memory and disk and terminal controllers). The SIMPLE and APL compilers were operational and a single-user operating system supported the execution of programs.[34]

By January 1974, a prototype experiment had been developed using the APL language services. Most of the full set of APL primitive functions had been programmed. The operating system could only support one experiment at a time. Editing and debugging facilities were available in preliminary form. Still to come were an easy-to-use interactive APL subsystem and a full multi-user timesharing system. [10]

Six months later, the system had been augmented with eighteen hard-copy terminals [Data Terminals and Communications “daisy wheel” devices], a magnetic tape drive, and modems allowing remote dial-in. Planning was underway for a connection to the Campus Computer Center's remote job entry system, analog-to-digital converters, and graphic display terminals. The full interactive APL environment was available and had supported multiple-process and

[29] W. Greiner, Personal communication, 7 and 15 October 2024 and 21 May 2025.
[30] S. Linker, “APLOT: APL Functions for Plotting on DTC Terminals,” Technical Document, Systems Group, Center for Research in Management Science, University of California, Berkeley, 21 August 1974. [24] Available: https://crms-apl.computerhistory.org/#Linker1974a
[31] S. Linker, Personal communication, 3 June and 5 October 2024.
[32] W. Greiner, “Scalar Functions: An APL Analysis of the Vertical and Horizontal Implementations,” Master's Thesis, Electrical Engineering and Computer Science, University of California, Berkeley, 12 June 1975. [24] Available: https://crms-apl.computerhistory.org/#Greiner1975
[33] J. M. Carman, presiding, “Departmental Luncheon,” 22 November 1972. [25], Carton 2.
[34] Center for Research in Management Science, “Annual Report,” 1972-73. [25], Carton 8.

multiple-terminal experiments since spring 1974. An interim version of the timesharing system was available. Oligopoly and macroeconomic game programs had been converted from the PDP-5/8 system to the new system in order both to check out the new system and to support a development effort in classroom use. At this point, the PDP-5/8 system had been largely phased out.[35]

By February 1975 Hoggatt and Balderston organized a task force with subcommittees to "pool knowledge and judgements concerning the several problem areas".[36] Apparently they were concerned with how to allocate funds for maintenance versus extensions to the system. By July of that year, the system was being used for both experiments and classroom exercises, although using the system while it was being developed caused some difficulties. The full Phase II operating system was in use and was being used for text editing and formatting as well as APL development. Twelve RAMTEK displays with light pens and keyboards had been acquired; they were expected to be integrated into the system by that fall, which would complete the work envisioned for the system.[37]

The final report of the 1971–1976 Laboratory Development grant described the state of the system as including an expanded core memory of 128K words, the link to the campus CDC 6400, and the RAMTEK display system.[38] The report noted: “All original major objectives for the computing system have been accomplished.” Peter Blatman, who had previously used APL\360 extensively, did a fair amount of APL programming on the system (see Usage, below) and found it to be quite usable, with the feel of “real APL.”[39]

The system was shut down in the late 1970s. Hoggatt lamented: “After spending large sums of NSF money to capitalize this facility, the University of California, Berkeley scrapped it for want of funds to pay for maintenance.” [1]

Usage

The difficulty of conceiving, developing, and carrying out a computer-based economics or social sciences experiment, especially with the technology of the 1960s and 1970s, seems to have limited the number of such experiments carried out. The following sections summarize what I've been able to discover, first about experimental usage of the CRMS APL system, and second about other usage.

Experiments

[35] Center for Research in Management Science, “Annual Report,” 1973-74. [4]
[36] A. C. Hoggatt and F. E. Balderston, “Task Force Subcommittees for CRMS Laboratory Functions,” 13 February 1975. The cover letter provides motivation for the task force. [24] Available: https://crms-apl.computerhistory.org/#HoggattBalderston1975
[37] Center for Research in Management Science, “Annual Report,” 1974-75. [4]
[38] F. E. Balderston, L. L. Vance, and A. Manza. “‘Control Organizations’ and Their Interactions with Operating Entities: An Experimental Investigation,” Appendix C, Research Proposal submitted to the National Science Foundation, 24 March 1977. [27], Box 48.
[39] P. J. Blatman, Personal communication, 6 October 2024.

- Professor Mark Garman was interested in the impact of structural alternatives of asset markets upon "the general social welfare"—efficiency, fairness, etc. His approach, described in the first of a multipart report "Laboratory Studies in Asset Trading,"[40] involved empirical studies, laboratory studies, theoretical studies, and simulation studies. As a basis for the laboratory studies, he began the Trading Floor/1 project in 1971, using the PDP 5/8 system. With three research assistants he built a detailed simulation of an automated securities exchange.[41,42] It defined roles for customers, brokers, and specialists. In 1974 the simulation was rewritten for the APL system, resulting in Trading Floor/2.[43,44,45] The more powerful computer system allowed various improvements. In particular, the companies being traded could now play an active role: under appropriate circumstances, they could "enter the auctions to float new equity issues at the going market prices." In a 1977 paper, Hoggatt noted one finding from this was that in the absence of the specialist (what is now called the market maker) in the market, "Trading creeps to a halt." He recommended the NYSE keep this in mind as they moved toward an automatic exchange. [17]

[40] M. B. Garman, "Laboratory Studies in Asset Trading," August-September 1973. [24] Available: https://crms-apl.computerhistory.org/#Garman1973

[41] [M. B. Garman,] "Experimental Stock Market Instructions." 20 August 1969. An early sketch of what became Trading Floor/1. [24] Available: https://crms-apl.computerhistory.org/#Garman1969

[42] M. B. Garman, "Trading Floor/1: A Prototype of an Automated Securities Exchange," Institute of Business and Economic Research, Working Paper No. 7, Graduate School of Business Administration, University of California, Berkeley, July 1972. [24] Available: https://crms-apl.computerhistory.org/#Garman1972

[43] M. B. Garman, "A Description of an Experimental Securities Exchange," Working Paper CP-372, Center for Research in Management Science, University of California, Berkeley, March 1975; revised October 1975. [24] Available: https://crms-apl.computerhistory.org/#Garman1975a

[44] [M. B. Garman,] "CLAIMS - Experimenter Documentation," circa 1975? [24] Available: https://crms-apl.computerhistory.org/#Garman1975b

[45] [M. B. Garman,] "CLAIMS Experiment Instruction Manual," circa 1975? [24] Available: https://crms-apl.computerhistory.org/#Garman1975c

Figure 4. Trading Floor recruiting poster, May 1974. Courtesy Mark Garman.

- Professor Hayne Leland studied the market in leases for offshore oil. The 1974-1975 CRMS Annual Report[46] noted: "Peter Blatman ... prepared a large simulation of oil-lease bidding which is to test the capability of the META to handle 150 subjects concurrently in a single simulating episode". In his 1977 paper on usage, Hoggatt noted: "A tentative result: the government captures most of the value of the oil in the ground (winning bids cluster near the net yield values for each tract), and the bidders go broke buying information which has zero value as soon as the bids are opened." [17]
- Shlomo Gill, a Ph.D. student under Hoggatt, carried out an organization theory experiment during 1975,[47] publishing his thesis in 1976. [8] In his 1977 paper on usage, Hoggatt reported: [17] "After the data were collected on one subject the initial analysis was in hand within hours. As the thesis was written, the text was input on the computer. When the full Ph.D. committee agreed that it was done, text editing programs produced in half a day a ribbon copy of the thesis acceptable to the registrar of the university."
- Hoggatt, Herman Brandstätter, and Peter Blatman investigated the influence of "robots" as participants in bargaining experiments. The system, programmed by Blatman in 1976, is described in [3]. Two generations of robot players (the first based on analysis of student subjects; the second based on play among businessmen, university administrators, and the first-generation robots) were included as participants in a Harsanyi-Selten bargaining game. These robot players were apparently not detected by the human players, and they did not affect the game play. [18]

[46] Center for Research in Management Science, "Annual Report," 1974-75. [4]
[47] Center for Research in Management Science, "Annual Report," 1974-75. [4]

- In a 1977 NSF research proposal, Balderston mentioned an interactive model for financial planning of a savings-and-loan association, developed by Clifford Olsen in fulfillment of a requirement for a master's degree. It's not clear this was used in an experimental mode.[48]
- In the same 1977 proposal, Balderston mentioned that he had developed a 'Little University' simulator with George Weathersby and Jeffrey Moore for the PDP-5/PDP-8 system that was rewritten for the APL system and used with a wide variety of student groups, business executives, and educational administrators.
- Finally, in the 1977-1978 Annual Report, Balderston noted:[49]

> A model of regulatory interaction with energy-using firms was programmed for experimentation in the Management and Behavioral Sciences Laboratory. During 1977-78, two Business Administration class groups and three groups of energy-policy professionals participated in laboratory sessions to make energy-policy decisions under different simulated policy restrictions. Direct rationing proved to be easier for the regulators to administer but produced greater hostility on the part of the managers of regulated firms and greater distortion of production patterns than occurred when price incentives were employed by the regulators.

Other uses

In addition to the game-style experiments that had been the original motivation for the Laboratory and its computer systems, usage of the APL system was extended to simulations, classroom teaching, and general-purpose time-sharing. An early example was a demonstration of the manipulation of the money supply and governmental budget of a mystical economy (EQUILIBRIA) conducted by doctoral student Peter Elias in March 1974.[50,51] In the summer of 1975 Hoggatt was planning to use extensions to EQUILIBRIA for teaching macroeconomics, extensions to oligopoly games for teaching microeconomics, and to develop a year-long sequence in quantitative methods for graduate and undergraduate education. He hoped to get back to experimentation after that.[52]

In 1975, then-Computer Science Masters student Peter Blatman used the APL system to perform cluster analysis of objects for robot path planning. His report includes a listing of the APL source code.[53] Blatman transferred from the Computer Science Department to Business Administration and programmed several of the experiments mentioned previously. His MBA thesis gave an

---

[48] F. E. Balderston, L. L. Vance, and A. Manza, "`Control Organizations' and Their Interactions with Operating Entities: An Experimental Investigation," pp. 12-13, Research Proposal submitted to the National Science Foundation, 24 March 1977. [27]

[49] Center for Research in Management Science, "Annual Report," 1977-78. [4]

[50] P. Elias with A. Hoggatt, "'EQUILIBRIA' (A Macroeconomic Computer Game)," B. A. 147 Classwork. Working Paper No. 1, School of Business Administration, University of California [, Berkeley] [, 15 March 1974]. [24] Available: https://crms-apl.computerhistory.org/#EliasHoggatt1974

[51] P. Elias, "Some Notes on a Learning Experience," Working Paper No. 3, [School of Business Administration, University of California, Berkeley], 27 March 1974. [24] Available: https://crms-apl.computerhistory.org/#Elias1974

[52] Center for Research in Management Science, "Annual Report," 1974-75. [4]

[53] D. T. Killen and P. J. Blatman, "A Report on Environment Modeling and Model Pre-Processing for Jason, the Berkeley Robot," Report, C.S. 282H, Department of Electrical Engineering and Computer Science, University of California, Berkeley, Winter 1975. [24] Available: https://crms-apl.computerhistory.org/#KillenBlatman1975

implementation and performance analysis of a data encryption algorithm, again with full source code included.[54]

Impact

Economists began conducting experiments with human subjects by the late 1940s. The approach grew in popularity through the 1950s. For example, Hoggatt described a paper-and-pencil version of an oligopoly experiment in [13]. Hoggatt's experimental work and his familiarity with computers led him much further. As Andrej Svorenčík has observed, the Management Science Laboratory pioneered the very concept of a laboratory for experiments in economics, with a controlled environment and computers mediating interactions and capturing and analyzing results. Hoggatt worked diligently to share the ideas with other experimental economists, which "informed their choices on how to organize their laboratories both as a physical space but also as a communal place". But it was that last step—forming an ongoing community—that did not come to fruition at CRMS. [28][29] Later laboratories founded in the 1980s by Charles Plott at Caltech and by future Nobel Prize winners Vernon Smith at the University of Arizona and Reinhard Selten at Bonn (both of whom were familiar with Hoggatt's lab) managed to establish a community of researchers and others who could sustain a practice of computer-based experimental economics. Nowadays a well-known software package called the Zurich Toolbox for Readymade Economic Experiments (z-Tree) makes it straightforward to develop and carry out economic experiments on a network of Windows laptops. [34]

Moving on

Morrow went on to found Thinker Toys and Morrow Designs and was one of the people who standardized the S-100 bus used in many early microcomputers. Grant and Greenberg founded North Star Computers. Zaks founded Sybex, a publisher of books on computer programming. Moore received his PhD in business and spent the rest of his career at Stanford. Redell completed his PhD in computer science and became an assistant professor at MIT. In two years he returned to the Bay Area, working successively at Xerox, DEC, AgileTV, and Google. I worked at a small software house named Virtual Memory Systems, then IBM, Xerox, Tandem, DEC, E.piphany, AgileTV, and Adobe.

Reflections

During the years of development from 1972 through 1975 or 1976, Moore's Law—the doubling of the number of transistors in integrated circuits every two years—was in its early stages. But there was already a visible change: the small integrated circuits and core memory used in the META 4 system were being replaced by large integrated circuits: single-chip microprocessors and dynamic RAM chips—the very components that Morrow, Grant, and Greenberg used in their subsequent companies. But the CRMS software wasn't easily portable, so the system died

[54] P. J. Blatman, “A Feasibility Study of a Data Encryption Algorithm for Use in an Electronic Funds Transfer System,” Master's thesis, Graduate School of Business Administration, University of California, Berkeley, 12 June 1978. [24] Available: https://crms-apl.computerhistory.org/#Blatman1978

when its hardware became unaffordable to service. In contrast, at about this same time the UNIX operating system was being rewritten in the programming language C, allowing it to be ported to an amazing variety of computers and to evolve to support many of today's phones, laptops, and cloud computers.

Bibliography

[1] M. Araten, et al., "The Winter Simulation Conference: Perspectives of the Founding Fathers", in *Proceedings of the 24th conference on Winter simulation (WSC '92)*, Association for Computing Machinery, 1992, pp. 37–62, doi: 10.1145/167293.167300

[2] F. E. Balderston and A. C. Hoggatt, "Simulation of market processes". Institute of Business and Economic Research, University of California, Berkeley, 1962. Available: https://babel.hathitrust.org/cgi/pt?id=umn.31951001579707d&seq=7

[3] P. Blatman, H. Brandstätter, and A. C. Hoggatt, "Operations manual for the Brandstatter Bargaining Game," Center for Research in Management Science, University of California, Berkeley, 1977.

[4] "Center for Research in Management Science Annual Reports", Bancroft Library, University Archives (308m.B814.ar).

[5] Digital Scientific Corporation, "The META 4 Series 16 computer system: preliminary system manual," Publication No. 7006MO, June 1970. Available: http://bitsavers.org/pdf/digitalScientific/7006MO_Meta16_SysMan_Jun70.pdf

[6] D. Falkoff and K. E. Iverson, "APL\360 user's manual," IBM Corporation, August 1968. Available: http://bitsavers.org/pdf/ibm/apl/APL_360_Users_Manual_Aug68.pdf

[7] U. Frey, "In memoriam: Austin Hoggatt 1929 – 2009." Available: https://senate.universityofcalifornia.edu/_files/inmemoriam/html/austinhoggatt.html

[8] S. Gill, "Experiments in the economic theory of teams," Ph.D. dissertation, University of California, Berkeley, 1976.

[9] C. A. Grant and M. L. Greenberg, "The uses of a microprocessor in an interactive computing system," in 4th Hawaii International Conference on System Sciences, January 12-14, 1971. Available: https://crms-apl.computerhistory.org/#GrantGreenberg1971

[10] C. A. Grant, M. L. Greenberg, and D. D. Redell, "A computer system providing microcoded APL," in Proceedings of the Sixth International Conference on APL (APL '74), Association for Computing Machinery, pp. 173–179, 1974. doi: 10.1145/800269.810809

[11] T. Haigh, "Oral history interview of William Kahan," 5–8 August, 2005, Berkeley, California. Society for Industrial and Applied Mathematics, Philadelphia, PA. Available: https://history.siam.org/oralhistories/kahan.htm

[12] A. C. Hoggatt, "Simulation of the firm," PhD thesis, University of Minnesota, 1957. Available: https://primo.lib.umn.edu/permalink/01UMN_INST/ijl1rs/alma9921164910001701

[13] A. C. Hoggatt, "An experimental business game," *Behavioral Science*, vol. 4, no. 3, pp. 199-203, July 1959. doi: 10.1002/bs.3830040303

[14] A. C. Hoggatt, J. Esherick and J. T. Wheeler, "A laboratory to facilitate computer-controlled behavioral experiments," *Administrative Science Quarterly*, vol. 14, no. 2, pp. 202–207, June 1969. doi: 10.2307/2391098 Also: Reprint No. 88, Center for Research in Management Science. Available: https://crms-apl.computerhistory.org/HoggattEtAl1969

[15] A. C. Hoggatt, “Response of paid student subjects to differential behaviour of robots in bifurcated duopoly games,” The Review of Economic Studies, vol. 36, no. 4, pp. 417-432, October 1969. doi: 10.2307/2296468
[16] A. Hoggatt, M. Greenberg, and J. Moore, “A micro-programmed APL language laboratory Control System,” (Abstract only) in A. C. Hoggatt, ed., *Proceedings of the 1973 Winter Simulation Conference*, Sponsored by ACM/AIEE/SHARE/SCi/TIMS, January 1973. The session was "Night in a Berkeley Laboratory"; the introduction noted: "A novel micro-programmed APL computer for laboratory control is being developed in the Laboratory under sponsorship of the National Science Foundation. Visitors will have an opportunity to see a demonstration of the system and hear a presentation of its design features. doi: 10.1145/800293.811620
[17] A. C. Hoggatt, “On the uses of computers for experimental control and data acquisition,” *American Behavioral Scientist*, vol. 20, no. 3, pp. 347-365, January/February 1977. doi: 10.1177/000276427702000303
[18] A. C. Hoggatt, H. Brandstätter, and P. Blatman, “Robots as instrumental functions in the study of bargaining behavior,” in *Bargaining Behavior*, Heinz Sauermann, Ed., Tübingen: J. C. B. Mohr (Paul Siebeck), 1978, pp. 179–210. doi: 10.23668/psycharchives.10193
[19] B. W. Lampson, W. W. Lichtenberger and M. W. Pirtle, “A user machine in a time-sharing system,” in Proceedings of the IEEE, vol. 54, no. 12, pp. 1766–1774, December 1966. doi: 10.1109/PROC.1966.5260 Available: https://bwlampson.site/02-UserMachine/WebPage.html
[20] J. Markoff, “George Morrow, a personal computer visionary, dies at 69,” *The New York Times*, 9 May 2003, page A29. Available: https://www.nytimes.com/2003/05/09/business/george-morrow-a-personal-computer-visionary-dies-at-69.html
[21] P. McJones and D. Redell, “History of the CAL Timesharing System,” *IEEE Annals of the History of Computing*, vol. 45, no. 3, pp 80-91, July-September 2023. doi: 10.1109/MAHC.2023.3282262
[22] J. Moore. ECL/3, “The environmental control language of the Management and Behavioral Sciences Laboratory time-sharing system,” Center for Research in Management Science, University of California, Research Report LR-15, 1971.
[23] J. G. Myers and R. E. Miles, “In memoriam: Frederick E. Balderston 1923 – 2007,” Academic Senate, University of California. Available: https://senate.universityofcalifornia.edu/_files/inmemoriam/html/frederickbalderston.html senate.universityofcalifornia.edu
[24] “Records of the CRMS APL Collection,” Computer History Museum, Acquisition #2025.0092. Available: https://crms-apl.computerhistory.org
[25] “Records of the Office of the Chancellor, University of California, Berkeley, CU-29,” The Bancroft Library, Univ. California, Berkeley, CA, USA, 1971-1973.
[26] K. Savetz, “Rodnay Zaks, founder of Sybex Books—interview,” YouTube, audio recording, 14 October 2016. Available: https://youtu.be/b73n-bElHH4?si=kS2Hp6nTiQn2ViCA
[27] “Martin Shubik Papers,” David M. Rubenstein Rare Book & Manuscript Library, Duke University.
[28] A. Svorenčík, “The experimental turn in economics: a history of experimental economics,” Ph.D. dissertation, School of Economics, University of Utrecht, Utrecht. Available: https://ssrn.com/abstract=2560026

[29] A. Svorenčík, "The role of computers in the emergence of experimental economics laboratories: material culture and moral economy," *Œconomia History, Methodology, Philosophy* vol. 13, no. 3, 2023. doi: 10.4000/oeconomia.15939
[30] Wikipedia Contributors, "Project Genie," *Wikipedia, The Free Encyclopedia*, last revised 27 March 2023 19:13 UTC. Available: https://en.wikipedia.org/w/index.php?title=Project_Genie
[31] Wikipedia Contributors, "Project One (San Francisco)," *Wikipedia, The Free Encyclopedia*, last revised 21 February 2024 20:57 UTC. Available: https://en.wikipedia.org/w/index.php?title=Project_One_(San_Francisco)
[32] R. Zaks, D. Steingart, and J. Moore, "A firmware APL time-sharing system," in *Proceedings of the May 18-20, 1971, Spring Joint Computer Conference (AFIPS '71 (Spring))*, Association for Computing Machinery, pp 179–190. doi: 10.1145/1478786.1478812
[33] R. Zaks, "A microprogrammed APL implementation," Ph.D. dissertation, University of California, Berkeley, June 1972. Published under the same title by Sybex, 1978.
[34] Zurich Toolbox for Readymade Economic Experiments, Universität Zürich. Available: https://www.ztree.uzh.ch/en.html

Acknowledgements

The original CRMS APL system was done as part of a Systems Development effort under National Science Foundation Grant GS-32138. Additional work was done under Grant SOC75 - 08177.

Dave Redell shared recollections, a number of documents, and the two SIMPLE source code listings. Chuck Grant shared recollections and assisted with research at the Bancroft Library. Sheldon Linker shared recollections. Wiley Greiner shared recollections and a number of documents. Mark Garman shared recollections and a number of documents. Peter Blatman shared recollections and scans of two documents. In addition to his insightful publications, Andrej Svorenčík provided scans of a document from [27].

Additional people who worked on the project include Paul Gee (APL subsystem), James L. Harp (operating system for the SIPU),[55] Ross Harrower (I/O interface hardware),[56] David Koch, Paul Glassco (Claims Manager), and Leon Goldberg (lab manager and some application programs).

Appendix: Errors in AIPU microcode

Over the life of the system, four changes (noted in pencil on the listing[57] on the pages indicated below) were made to the AIPU microcode, which I'd debugged on the simulator before leaving the project in January 1973:

[55] J. L. Harp, Personal communication, 12 June 2024.
[56] C. A. Grant, Personal communication, 5 October 2024.
[57] P. McJones, "APL\META 4 Microcode," Digital Scientific META 4 Series 16 assembly language, 28 January 1973. [24] Available: https://crms-apl.computerhistory.org/#McJones1973a

"Bug 1" (page 76)

If the APL processor was started up with an incorrectly formatted heap (a free list that loops back to its middle rather than to the starting point), it could loop forever, requiring a power reset to clear it. To deal with this, a counter was added to the code that searched the free storage list, so it would stop after 2^16 iterations.

"Bug 2" (page 14)

The wrong error number was generated for a particular error condition detected when the processor started up on a new APL process.

"Bug 3" (page 31)

The ACOPY function failed to generate a trap if given the address of a variable with an "undefined" value.

"Bug 4" (page 24)

When the reference count of a block of memory is decremented and becomes zero, it shouldn't actually be written back to memory as a zero, since the FREE routine might trap for some reason. (The AIPU's model is that if a trap occurs, the state must remain as it was at the beginning of the current instruction, to make it easy to restart after the trap is handled.)