Iterations
An Interdisciplinary Journal of Software History

211 Andersen Library, 222 – 21st Avenue South, Minneapolis, MN 55455 USA

Software Development at the Eckert-Mauchly Computer Company Between 1947 and 1955

Arthur L. Norberg
Charles Babbage Institute

Date published: 31 December 2003

Abstract: Histories of Eckert-Mauchly Computer Company (EMCC) center on the hardware design and development of the BINAC and UNIVAC. Important as this side of the story is, it is not the whole story. Unlike several companies entering the new area, which designed computer systems and left the software development to the customer, EMCC tried to provide more to the customer. In addition to their hardware design and development program, John Mauchly and J. Presper Eckert organized a software development program to educate the customer about the ability and use of a computer system. This software program influenced and was influenced by hardware developments. Mauchly, especially, saw the importance of software needs even before he and Eckert embarked on their industrial adventure and he worked for many years on providing the software customers would need if EMCC were to play a major role in the computer market.

Keywords: coding techniques, programming systems, Eckert-Mauchly Computer Company (EMCC), John Mauchly, J. Presper Eckert, software development, EDVAC II, Frances Elizabeth Holberton, Betty Jean Bartik, ENIAC, UNIVAC I, BINAC, Jean Jennings, Hubert M. Livingston, Arthur J. Gehring, Grace Murray Hopper, UNITYPER, IBM Defense Calculator, Laplace boundary value problem, automatic programming, subroutines, differentiator, compilers, assembly routines, Short Code, William F. Schmitt, Albert B. Tonik, Robert Logan, A-O, neutral corner concept, Richard Ridgway, Lloyd Stowe, Pseudo-code, Math-Matic, Flow-Matic, UNIVAC Fac-tronic System

Histories of Eckert-Mauchly Computer Company (EMCC) center on the hardware design and development of the BINAC and UNIVAC, systems developed originally for the Northrup Aviation Company and the Census Bureau. As there was no market for computer systems outside government circles in the second half of the 1940s, Eckert and Mauchly pursued their dream by garnering contracts for machines in the hope that they could

spread the development cost over enough contracts. Repeatedly, they underestimated the costs and design obstacles of a new technology. Eckert and Mauchly saw the BINAC, part of a military missile system contract, as a prototype of the UNIVAC, and another way to spread the costs. Try as they might, there was simply not enough funds to achieve their goal. Yet they were close in late 1949 when they realized new funds could only come if they sold a portion or all of the company. Remington Rand bought the company in early 1950, paid the accumulated debts, laid out funds for construction of UNIVAC systems, and tried to renegotiate the contracts in force to stem the tide of losses. Remington Rand needed a number of years to turn the corner of profitability for computer systems, something they achieved by the early 1960s.

There is certainly enough drama in these events as evidenced in the several books and numerous articles on them that have appeared. EMCC represents a quintessential example of the startup company in a new technological area that encounters financial difficulty and can only continue by selling out to a group with deeper pockets. Important as this side of the story is, it is not the whole story. Unlike several companies entering the new area, which designed computer systems and left the software development to the customer, EMCC tried to provide more to the customer. In addition to their hardware design and development program, they organized a software development program to educate the customer about the ability and use of a computer system. This software program influenced and was influenced by hardware developments. Mauchly, especially, saw the importance of software needs even before he and Eckert embarked on their industrial adventure and he worked for many years on providing the software customers would need if EMCC were to play a major role in the computer market.

Early Developments

Addressing the history of software development of the earliest period of electronic digital computing has yielded some useful information about the approaches used by the early companies to incorporate new abilities into computer system designs. Historians placed an early focus on instruction sets and structures, and virtually always noted the absence of applications programs for any of these new systems.[1] They, then, seemed to have hurried on to describe what has come to be seen as the major revolutionary development of the mid-1950s: higher-level languages such as FORTRAN and COBOL. Readers often get the impression that little or no software development went on in the first decade of the industry's history. This article, focused on the software efforts of the Eckert-Mauchly Computer Company (EMCC) in the early decade, is intended to illustrate the types of software issues of concern to a new manufacturer in this new industry, a situation not uncommon in other companies as well.

Very early in the history of EMCC, John Mauchly assumed responsibility for programming, coding, and applications for the company's planned computer systems. His early interaction with representatives of the Census Bureau in 1944 and 1945, and discussion with people interested in statistics, weather prediction, and various business problems in 1945 and 1946 focused his attention on the need to provide new users with the software to accomplish their objectives. He knew it would be difficult to sell computers without application materials, and without training on how to use the systems. As others in the company designed the hardware and logic of the EDVAC II, as it was then called, Mauchly focused his attention on software questions.[2] There was no organized software and applications department during 1947; Mauchly designed the organization as time transpired and machine design determined needs. Nevertheless, in early 1947, he began to recruit a staff of mathematicians interested in coding for a new department he intended to organize as president of the company.

The early members came with experience in coding for various computer systems including ENIAC. Frances Elizabeth Snyder (later Holberton) joined EMCC in 1947. She had already had an illustrious career in computing, as one of the original members of the ENIAC computing group. In 1942, shortly after graduation from the University of Pennsylvania with a degree in journalism, she joined the Philadelphia Computing Unit at the Moore School. This group worked on problems associated with the tables being produced by the Aberdeen Proving Ground. Snyder, and another member of the group, Betty Jean Jennings (later Bartik), developed a trajectory program used to control the operation of the ENIAC during the public demonstration in February 1946. In 1947, Snyder transferred to Aberdeen when the ENIAC was moved there. She served EMCC as a part time consultant in February through April 1947, and in July of 1947, she left the civil service and became an employee of EMCC.[3] She stated later that she had actually asked Mauchly if she could join EMCC.[4]

Jean Jennings Bartik graduated from Missouri State Teachers College (now University) in 1945, with a mathematics major. She studied analytic geometry, trigonometry, and physics. In summer 1944, she worked at Pratt-Whitney Aircraft in Kansas City on engine work, but did not want to engage in that type of work after graduation. She applied for a job at the Aberdeen Proving Ground for their location at the University of Pennsylvania. After two months, she received a letter offering her the job and she left for Philadelphia the next evening. Not long after, Jennings applied for a position as an ENIAC coder, and she, Snyder, and several others were chosen and sent to Aberdeen in June 1945 for training. After several months at Penn coding for the ENIAC, Jennings was selected to

head a group to generate programs to turn ENIAC into a stored-program system. She decided not to move to Aberdeen when ENIAC moved there, and after completing this programming work she accepted a job at EMCC in early 1948.[5]

With these accomplished mathematicians, the department immediately engaged in a number of projects while they acquired knowledge of the needs of users to whom they expected to sell UNIVAC systems. Many conversations occurred between Mauchly and Snyder, on the one hand, and prospective customers on the other. Snyder remembered early trips to the Census Bureau, NBS, and Martin Marietta. Later, she visited many UNIVAC I sites for consultation on programming problems.[6] As she pointed out in her 1983 interview for the Charles Babbage Institute (CBI), she had no training in computing, no courses in formal logic. Indeed, when she interviewed potential programmers for EMCC and elsewhere, she asked such questions as "Did you like plane geometry?" in an effort to establish whether the person had a puzzle-solving ability, which might make them good programmers.[7] But this line of questioning did not always produce the desired result. In an interview in 1990, Jean Jennings reported that in EMCC

> We used to argue about what kind of a person [made the best programmer]. We had this little test that Art Katz worked out. We used to give these people this little test… I don't know what anybody else used, but their enthusiasm for doing something new was what always impressed me personally. …Hildegard Nidecker came along, who had much experience in doing calculations for the Army. And she flunked his test, so nobody wanted to hire her. Then he [Katz] said, 'this is ridiculous. This just proves to me that the test is ridiculous. We know that she is going to do a good job,' and in fact [we hired her] and she retired from UNIVAC [in the 1980s]. So the truth is we did it by the seat of our pants, and I personally did it if I liked the person.[8]

Whatever the evaluation scheme, an effective group was assembled at EMCC.

The applications group grew slowly between 1947 and 1950. M. Jacoby joined the firm in December 1947. Dr. Arthur Katz came to EMCC shortly thereafter in February 1948 and, as mentioned, Jean Jennings became an employee at the end of March 1948. Four more people joined in the second half of 1948 and early 1949—M. League, V. Hovsepian, Hubert M. Livingston, and Arthur J. Gehring. Besides the special studies for prospective customers noted above, Snyder worked closely with Mauchly on the early codes (instruction sets) for EDVAC II and BINAC, codes C-1 through C-5. More special studies involved work for the Army Map Service, Oak Ridge, and Glenn L. Martin Company, all completed by early 1949. Snyder worked on a floating decimal routine (at one point along with Katz), a reciprocal routine, double precision operations, and reciprocal square roots. Mauchly, while working with Snyder on the code,

keeping contact with potential customers, and overseeing the programming group, worked with several people on such problems as the generation of random numbers, bi-harmonic problems, and programs for BINAC. Jennings spent most of her energies on test routines for BINAC. She even used some time programming chess and gin rummy games.[9]

Rounding out the early programming group, Grace Murray Hopper, who joined EMCC in 1949, possessed substantial training in mathematics and experience in computer development and coding. She studied mathematics at Vassar College in the 1920s, and in 1934 received a Ph.D. in mathematics from Yale. Hopper was elected to Phi Beta Kappa and Sigma Xi for her accomplishments. For a spell in the 1930s, she taught mathematics at Vassar and held a postdoctoral fellowship award from Vassar, which she used at New York University in 1941. She entered the US Navy in 1943, and, after training, was posted to the Bureau of Ordnance Computation Project at Harvard University. Working with Howard Aiken on the Mark I and Mark II, she developed several programs, and served as co-author of the Mark I and Mark II computer manuals.[10] Hopper joined the Harvard staff as a Research Fellow in 1946, continuing to work on Mark II and Mark III for the navy. From this position, she moved to EMCC. Rounding out the early group was Herbert F. Mitchell, who joined EMCC in 1949 and became head of the new Laboratory for Computational Analysis. Mitchell received a Ph.D. from Harvard in 1948, where his dissertation advisor was Howard Aiken and he worked on developments in numerical analysis. Hopper and Mitchell had worked together on various Mark designs at Harvard after the war.

The EMCC programming activities fell into three categories. First, there were the requests of the customers for programs to accomplish their tasks of payroll, procedural flows, accounting, and purchasing. Second, there was commercial research undertaken by EMCC to enhance sales of computer systems: collation, sorting, merging, editing, tabulated examples, and integrated systems for payroll. And third, EMCC engaged in engineering studies in collaboration with the engineering design group. Besides the logical design of BINAC and UNIVAC, they investigated the logical design of the UNITYPER, card-to-tape converter, supervisory control, speed comparisons of codes, error detection, and reliability.[11]

To understand this group's activity better, it is useful to return to March 1947, an important month in EMCC, as the engineering and programming groups made many decisions about basic design, which no doubt is the reason Snyder served as a consultant during this time and the applications group began to grow in the following summer. For example, March 11 and 12 were devoted to a design conference on the EDVAC II. During this conference the staff discussed the basic activities of their design—sorting speed, conversion of data from tape to memory and back, instructions for

operating the tape system, displaying the memory, instructions for moving from one point in a program to another, and starting and stopping. The give and take during the meeting, as reflected in summary minutes, resulted in some major decisions of how the storage system would be designed and operated.[12] Census problems would require significant sorting, and if the EDVAC II used binary sorting instead of decimal, then gains over punch card systems would be offset making the EDVAC II and the punch card systems near equal for sorting. Moreover, in systems with a long latency period, sorting would be very time consuming. Times of recovery of data with tapes were estimated to be about ½ millisecond. "If mercury tanks holding 20 words are used, the average time lost in obtaining a word from such a memory is ½ millisec." This meant that the internal speed of discriminating was comparable with the speed of the input and output.[13] Calculation revealed that the gain in sorting using EDVAC II would be in the neighborhood of 2.5, a bad comparison when other tasks on the system showed gains of 100 to 1000. Some adjustments would have to be made. They discussed decreasing the latency time by using 10 word tanks instead of 20 word tanks. They believed they could insert some parallelism by providing for simultaneous use of internal operations and tape reading and writing. Data could then be transferred in blocks rather than individually. The discussion at the meeting was done under an assumption of a pulse rate of 1 megacycle. In further consideration of the input/output issues, the group evaluated doubling the pulse rate and returning to the 20 word tanks, but this raised reliability questions. Further study seemed needed. Virtually all of these ideas were tried over the next year.

Some of the instructions to accomplish these operations were obvious to the group, such as the arithmetic functions. Therefore, they only spent time considering additional orders to transfer data in the most efficacious manner. "It is desirable to make the orders used for reading and writing on tapes as simple as possible for the operator to use, and as simple as possible to 'mechanize' in control equipment."[14] Of four tapes, the system could run two at a time, with provision for running forward and reverse and reading and writing on different tapes simultaneously. Several instructions were to be designed for this purpose.[15] Over the next six weeks, Mauchly and Snyder designed the first instruction set of 26 instructions, five of which would later be dropped.[16]

Between March 1947 and May 1949, the applications group developed and analyzed ten variations of the code, or instruction set, for EMCC designs and design changes. Many similarities exist across these codes, although there are some important differences. C-1 through C-4 were based on a 2-megacycle pulse repetition rate; the next four schemes involved a 4-megacycle rate. C-9 dropped back to 2 megacycles, and C-10, the final instruction set for UNIVAC I, was to operate on 2.25

megacycles.[17] Word sizes varied from 52 pulses per word to 104, with C-10 ending at 91 pulses per word. C-1 called for 50 20-word mercury delay tanks, which in C-10 became 100 10-word tanks. Block size climbed from 20 words initially to 60 at the end. As Snyder noted, the characteristics that remained fixed throughout the code definitions were:

> The handling of decimal quantities
> Using excess 3 addition
> The handling of coded alphabetic
> Memory size from 1,000 to 100,000 words (4 digits)
> Tape servos for input and output
> Parallel read-write and compute
> Buffer between input, output, and memory
> 12-character digit words
> 2 instructions per word
> directly connected typewriter.[18]

To illustrate the differences in characteristics among the three major EMCC designs of the 1940s, Figure 1 shows a comparison of the features of the various Eckert-Mauchly designs compiled by Nancy Stern for her study of EMCC and the sources for the data.[19]

Programming the IBM Defense Calculator

It is instructive at this point to compare these activities inside EMCC with those occurring inside IBM while they designed programming for the new Defense Calculator, the system delivered at the end of 1952. Developed in 1951-52, the Defense Calculator, later called the IBM 701, was similar to the Princeton IAS design, but deviated from it in a number of ways. Designs for tape systems to be used with other systems suggested a 36-bit word length would allow the computer to use these tape systems, rather than requiring design of new storage. This word length allowed two 18-bit instructions per word. Addresses in this design could be assigned to each 18-bit half-word, and branch instructions fell naturally into the addressing scheme. Various techniques were devised to maximize the use of input/output systems without delaying internal operations of the system. Punch cards contained both data and instructions and these were read into the system while the CPU continued to operate on instructions.[20]

With IBM's Engineering-Applied Science group, a mathematical programming study proceeded to review the Defense Calculator design to develop programs to enhance the utility of the system to customers. The group either developed or adopted programs from other IBM systems for use with the Defense Calculator. Among these were programs for floating-point, complex and double precision arithmetic, and the extraction of roots. They designed complete application programs, a solution of Laplace's equation, and a partial differential equation program applicable to determining the distribution temperature in a nuclear reactor. Many new

A COMPARISON OF ARCHITECTURE, PERFORMANCE, AND PHYSICAL CHARACTERISTICS OF THE ECKERT-MAUCHLY COMPUTERS

	EDVAC	**BINAC**	**UNIVAC**
ARCHITECTURE			
Programming	Stored program	Stored program	Stored program
Data Transmission	Serial	Serial	Serial
Number representation	Binary	Binary	Decimal
Word Length	44 bits	31 bits	11 digits + sign
Other data types	-	-	12 characters/words
Instruction length	44 bits	14 bits	6 characters
Instruction format	4-address	1-address	1-address
Instruction set size*	12(16)	25(32)	45(63)
Accumulators/programmable registers	4	2	4
Main memory size	1,024 words	512 words	1,000 words
Main memory type	Delay line	Delay line	Delay line
Secondary memory	Magnetic drum	Magnetic tape	Magnetic tape
Other I/O devices	Cards, paper tape	Typewriter	Typewriter, cards, paper tape
Error detection	Redundant CPUs	Redundant CPUs	Redundancy, parity
PERFORMANCE			
Clock rate	1 MHz	4 MHz‡	2.25 MHz
Add time	0.864 ms†	0.285 ms†	0.525 ms†
Multiply time	2.9 ms†	0.654 ms†	2.15 ms†
Divide Time	2.9 ms†	0.633 ms†	3.89 ms†
PHYSICAL CHARACTERISTICS (approximate measurements)			
Vacuum tube count	3,600	1,400	5,400
Diode count	12,000	N/A	18,000
Power consumption	50 kW	13 kW	81 kW
Floor space of computer only	490 sq. ft.	N/A	352 sq. ft.

Figure 1. Adapted from Nancy Stern, *From Eniac to UNIVAC: An Appraisal of the Eckert-Mauchly Computers* (Bedford, Mass.: Digital Press, 1981).

KEY:

N/A—data not available.
*—number of instructions used (number encoded).
†—includes memory access time for instructions and operands.
‡—later reduced to 2.5 M

instructions were proposed, not necessarily to provide for new problem solution techniques, but rather to offer programs that would make previously attempted problems easier to solve.[21]

Two significant comparisons should be made here. First, a number of the programs developed were similar or identical to those proposed by EMCC

for use on UNIVAC I. This suggests that the approach to programming inside IBM was similar to that inside EMCC, where IBM programmers saw needs similar to those perceived by EMCC programmers. Second, IBM programmers traded on their advanced knowledge of business and engineering problem solution developed for tabulator operations over the previous two decades, an experience that no one in EMCC possessed outside of the Aberdeen problems area. Here EMCC stood at a disadvantage, and therefore took longer to develop a strong approach to programming. Thus, UNIVAC I's delivered possessed few applications programs, though they did arrive with basic programming structures available for programming applications by the customers, along with instructions on how to program the applications.

EMCC Reaching Out

The applications group, especially Mauchly, identified a need to provide training for incoming programmers, engineers, customers, and sales personnel, an area already deeply embedded in the business approach of IBM. As a result, several members of the group worked on a training manual and developed a course to be offered either at EMCC or at the customer's site.[22] The course began with defining the operating code for UNIVAC and an introduction to programming in which several short examples of coded operations were presented. Subsequent lectures included description of flow-charting, types of subroutines, collation, and matrix algebra. Students spent substantial time on specific examples to understand operations like floating point, round off, problems of tape wear, etc.[23]

While all this coding activity was going on, Mauchly, as President of EMCC, was extremely busy visiting many customer and potential customer sites. For example, between October 28 and November 14, 1947, Mauchly hosted visitors and took two trips to New York City and one to Chicago. There were multiple visits with representatives from A. C. Nielsen, Northrup, and Prudential Insurance. He participated in drafting proposals for sales, interviewing candidates for positions at EMCC, conferences with staff on design questions, and oversaw the applications group.[24] EMCC at the time ran a six-day work week, and often Eckert and Mauchly were in on Sundays. Eckert's idiosyncratic work habits sometimes led him to stay at work around the clock.[25]

Returning to the discussion of applications efforts at EMCC, as the use of ENIAC and the other computer systems of the middle 1940s showed, there was a large class of engineering and scientific problems that could be attacked using electronic computers. Mauchly, with his interest in weather problems, which had led him to be interested in electronic computation methods in the first place, was in a good position to know this. Thus, it is

no surprise that among the applications group there were people thinking about the solution to such problems. The better posed problems involved partial differential equations that could be solvable using finite difference techniques. Betty Snyder and Hubert M. Livingston investigated various solutions for the plane potential problem. This pair published an article in 1949 in which they presented a computer program for the UNIVAC to solve the Laplace boundary value problem.[26] They set up a two-dimensional space, and used a finite difference method originally proposed by H. Liebmann in 1918.[27] The article opened with a very brief description of the UNIVAC system, including a list of the instructions to be used in the solution of this problem. As is typical in the solution of such problems, the authors set up a region, in the example a rectangular space, though they argued how irregular spaces could be examined as well, with a mesh of horizontal and vertical lines, and set up the equations to calculate approximations for the partial differential equation that led to a set of difference equations. The number of equations resulting is equal to the number of interior mesh points. Either a direct or iterative method can be used to solve the linear system of equations. The authors discussed the availability of subroutines for use in these problems, and in a company document on the subject intended for circulation, discussed truncation errors, scale factors, and times of solution.[28] This problem is representative of the types of research going on in this group in the late 1940s.

Programming the UNIVAC

From 1947 on, coders at EMCC developed a number of subroutines for both mathematical and business use. By 1951, the number of subroutines had increased to the point where they needed to put some order into them to increase efficiency of use. While the other members of the Applications Department continued their work on programs and routines, including diagnostic routines, for UNIVAC I, Hopper assumed an interest in automatic programming. She attempted to meld the operations of the computer system and its programs with the use of subroutines. This idea of subroutines was exploited at EMCC before she arrived, as we saw above in the work of Betty Snyder. Hopper's contribution was to make it possible not just to call up a routine from memory, but, if necessary, actually construct a subroutine program, insert it into a program, and carry out the computation. The process of translating a subroutine into a program received the name "compiler." The needed information was delivered by UNIVAC under the control of an operation they called a "differentiator." The differentiator delivered the information necessary to program the computation of a function and its derivatives. The actual derivation of the function was done by the computer system, not by the programmer as before. Thus, the UNIVAC became capable of developing a completed program.[29]

Over the next few years, coding or programming became a burgeoning area of activity in university computer projects and in the few companies focused on machine developments. As Knuth and Pardo summarized in 1977, the first important programming tools were developed, focused initially on "general-purpose subroutines for such commonly needed processes as input-output conversions, floating-point arithmetic, and transcendental functions."[30] In their brief summary of the history of compilers, Knuth and Pardo drew attention to two developments of the early 1950s: assembly routines and interpretive routines.[31] The publication of the Maurice Wilkes, David Wheeler, and Stanley Gill volume on programming in 1951 became a classic in the training of programmers over the next decade at least.[32] An early version of the book reached programmers around the world in September 1950. Over the next few years, a range of interpretive routines appeared, perhaps the most notable for its influence being John Backus' IBM 701 Speedcoding System, published in 1954.

The EMCC group investigated the development of assembly routines for use with BINAC and UNIVAC in the late 1940s. Indeed, Knuth and Pardo credited Mauchly with development of the first "high-level" programming language that he implemented in a program called Short Code,[33] a program that could accept algebraic equations as written and the program would perform the indicated operations. William F. Schmitt coded this type of problem for BINAC. In 1950, he and Albert B. Tonik recoded the program for UNIVAC I, and in 1951 Robert Logan took the task a step further.[34] In this program, the twelve-digit word was broken into six two-digit packets. This can be illustrated with a simple example.

Evaluate x = a + b	
In Short Code:	00 S0 03 S1 07 S2

S0, S1, and S2 represent the memory locations of the quantities x, a, and b, and 03 stands for the operation of equality and 07 for addition. Read from right to left S2 is added to S1, which is placed in S0. Thirty operations were provided, including bracket indicators for evaluation of expressions, floating point arithmetic operations, finding integral roots, the basic mathematical functions, use of routines from a library, such as trigonometric and logarithm calculations, and input/output operations.[35] This program was an effort to introduce more flexibility into problem solving. And as long as the problems were small scale in which computer time was efficiently used, the code worked well. As the scale of problems grew, a point of diminishing returns arose where it was more efficient to code in the regular way.

From 1947 on, coders at EMCC developed a number of subroutines for both mathematical and business use. By 1951, this number had increased to the point where they needed to put some order into them to increase

efficiency of use. Hopper took on this task in October 1951, and between then and May 1952 she and her associates wrote the first Remington Rand compiler A-0.[36] As Jean Sammet pointed out, "a compiler must perform at least the following functions: Analysis of the source code, retrieval of appropriate routines from a library, storage allocation, and creation of actual machine code."[37] From Sammet's perspective, A-0, developed for UNIVAC I does exactly this, and she claimed that A-0 was the first compiler. Hopper, however, speaking in 1978, commented on A-0 in the following way.

> It wasn't what you'd call [a compiler] today, and it wasn't what you'd call a 'language' today. It was a series of specifications. For each subroutine you wrote some specs. The reason it got called a compiler was that each subroutine was given a 'call word,' because the subroutines were in a library, and when you pull stuff out of a library you compile things. It's as simple as that.[38]

While the other members of the Applications Department continued their work on programs and routines, including diagnostic routines, for UNIVAC I, Hopper assumed an interest in what she called "automatic programming." She attempted to meld the operations of the computer system and its programs with the use of subroutines. This idea of subroutines was exploited at EMCC before she arrived, as we saw above in the work of Betty Snyder. Hopper's contribution was to make it possible not just to call up a routine from memory, but, if necessary, actually construct a subroutine program from basic mathematical information supplied to the system by a mathematician or programmer, insert it into a program, and carry out the computation of the needed values of the function. After the needed information was inserted into memory, it could be delivered at any later time directly by UNIVAC. UNIVAC delivered the information necessary to program the computation of a function and its derivatives or values. Just as in the case of the differentiator, the actual derivation of the function was done by the computer system, not by the programmer as before. The process of translating a subroutine into a program received the name "compiler."

When coding the compiling routine A-0 (and A-1), the coder needed to keep in mind three sets of memory locations.[39]

(1) those locations used by the compiler, concerned with compilation, input of information and subroutines, and output of running tape and record.
(2) those locations used by the running tape, concerned with numerical computation, input of numerical data, and output of results.
(3) those locations of the individual subroutines.

Thus, at any given instant during compilation, a particular word usually had at least three addresses associated with it. For a given problem, it was assumed there would be four tapes (UNISERVOs) available. Tape number 1 contained the instructions for compilation and data handling; number 2

held the input data; number 3 contained the data called for by specific subroutines; and number 4 received the output data. Along with memory segments, say 000-059 for the initial read, for input, working storage, program, constants, and output blocks, the compiler had an area called the "neutral corner." The neutral corner contained certain transfer instructions. The concept of the neutral corner arose because Hopper quickly encountered the problem that in some cases the program needed to jump back for something previously processed, and at other times the need was to jump forward to a section of the program still unknown, therefore whose location was unknown. That is, there were two types of jumps to be coped with. Jumping back was simple; jumping forward was impossible. The telling of her solution to this problem bears presenting completely in her own words.

> And here comes in the curious fact that sometimes something totally extraneous to what you are doing will lead you to an answer. It so happened that when I was an undergraduate at college I played basketball under the old women's rules which divided the court into two halves, and there were six on a team; we had both a center and a side center, and I was the side center. Under the rules, you could dribble only once and you couldn't take a step while you had the ball in your hands. Therefore, if you got the ball and you wanted to get down there under the basket, you used what we called a 'forward pass.' You looked for a member of your team, threw the ball over, ran like the dickens up ahead, and she threw the ball back to you. So it seemed to me that this was an appropriate way of solving the problem I was facing of the forward jumps! I tucked a little section down at the end of the memory which I called the 'neutral corner.' At the time I wanted to jump forward from the routine I was working on, I jumped to a spot in the 'neutral corner.' I then set up a flag for the [forward operation] which said, 'I've got a message for you.' This meant that each routine, as I processed it, had to look and see if it had a flag; if it did, it put a second jump from the neutral corner to the beginning of the routine, and it was possible to make a single-pass compiler and the concept did come from playing basketball.[40]

A-0, however, was the only single-pass compiler built. Hopper believed A-0 should be a one-pass compiler. The information defining a problem came from one tape unit and the program was written on another because UNIVAC I had only 1,000 words of storage, leaving little room for anything but the basic steps of the compiling process.

In the technical language of the instruction manual, the neutral corner was described thusly.

> When the compiler, processing operation a, is informed that one of the exits of operation a must transfer control to operation a+b (b >1), a 'forward pass' is required, [Figure 2]. The compiler inserts in the exit line of operation a, an instruction transferring control to memory location g in the neutral corner. The compiler then records the fact that, the neutral corner has in storage a forward pass destined for operation a+b. As each successive operation is treated, the compiler looks to see whether or not a forward pass has been tossed to that operation. Hence, it will find a pass to operation a+b. The instruction

> transferring control to operation a+b is generated and delivered to position g in the neutral corner.[41]

All of this was on the Running or Program Tape, tape #2.

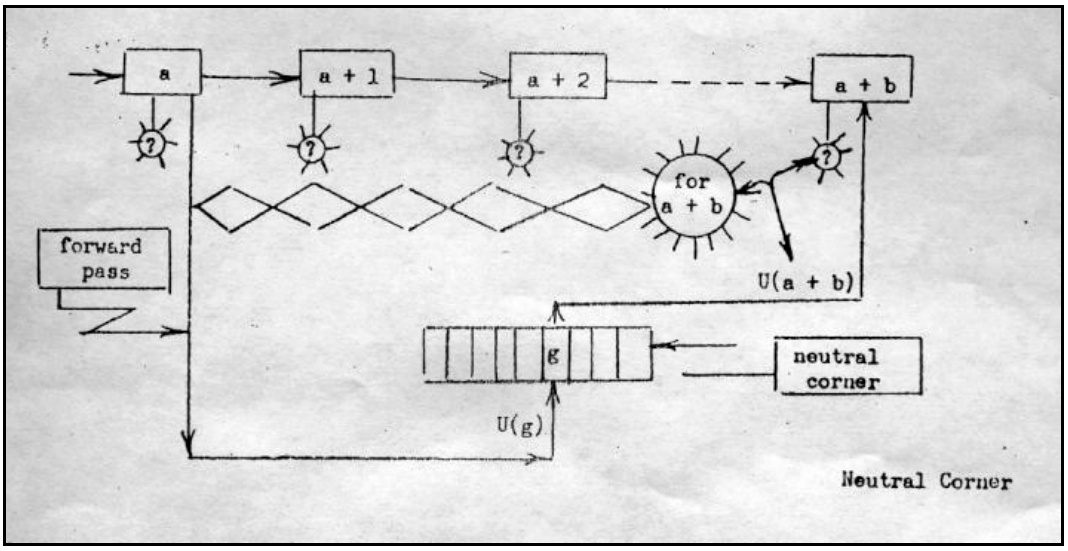


Figure 2. Use of the Neutral Corner concept in A-0.

Similarly, for the Compiling Tape, tape number 1, it was assumed that four UNISERVOs were available. These tapes were for the compiler, information defining the problem, the subroutine library, and the running tape. The memory sections were broken down in the same way on this tape, though each section might contain different kinds of information than on the running tape. The information defining the problem consisted of a set of operations, each defined by three or more words (Figure 3). The information contained the call number of the operation and subroutine, one or more "argument words" identifying quantities entering the operation, "control words" if the normal exit to the next operation was to be altered or if the subroutine had more than one exit, and, lastly, one or more "result words" identifying the results produced by the operation. The subroutines in the library were in alphabetical order. As each subroutine was entered in the running program, a record was kept by the computer, including its call number, the number of the operation, and the memory location in running memory at which the subroutine began.

The compiling tape was constructed in blocks from 1 to 13, where blocks 1 to 4 contained the data and instructions for the compilation and blocks 5 through 13 contained the compiling program. The A-1 compiler was more extensive than A-0, and provided for longer programs, a larger number of transfers of control to the neutral corner, and more working storage.

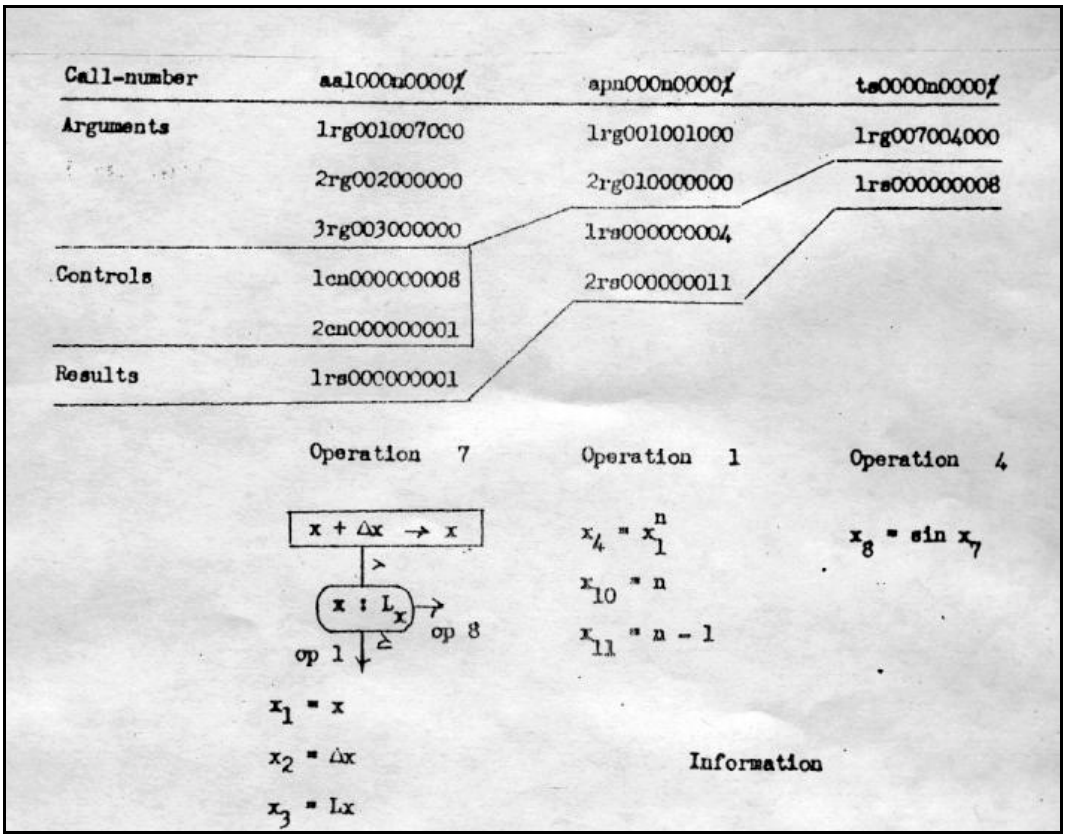


Figure 3. Compiling operation in A-0.

Hopper presented the A-0 compiler at the Association for Computing Machinery (ACM) meeting in Pittsburgh in May 1952. She carried 200 copies of her presentation to the meeting, and returned with 100 copies to Philadelphia.[42] Richard Ridgway in a paper delivered in September 1952 offered some comparative data for the use of A-0 acquired around the time of Hopper's talk. The group compared the calculation of the problem discussed above using the conventional method of program development and running and using the compiler. In the conventional method, 740 minutes were needed of the programmer's time, 105 minutes for auxiliary manpower and equipment, and 35 minutes to run the problem on the UNIVAC. The equivalent numbers using the compiler were 20, 20, and 8.5 minutes, respectively. Thus, problem solution required 880 minutes conventionally and 48.5 minutes using a compiler.[43] In spite of this advantage in time, the A-0 compiler did not come into general use, because EMCC was in process of increasing its generality (A-1) and then developed a more effective compiler by 1955 (A-2).

The time specified for programmer minutes in a given problem is only useful in this comparison. The time translates into only about an hour-and-a-half. A more detailed analysis of the conventional method for UNIVAC

would show that the problem setup could sometimes take weeks, especially with a new problem. Consider the four stages in addressing problem analysis in the EMCC programming group. Lloyd Stowe noted in early 1951 that the programmer's task could be divided into four parts: analysis of the problem, preparation of block diagrams and flow charts, coding, checking, and preparation of time estimates, and, if needed, running of sample problems through the code, and "bookkeeping."[44] He noted that in one case he spent seven weeks on preparation of a problem. After the first preparation, he received more information about the problem. And after a second preparation, more information came that the people with the problem had not previously recognized as necessary. So a third effort was required. This example is reminiscent of the later problem in the 1960s of trying to obtain an expert's knowledge for developing an expert system. In the latter case, understanding a problem requires consultation with the proposers of the problem and analysis of the information provided. Since Stowe concentrated on commercial problems, he was particularly interested in the nature and amount of input data. The Census Bureau data, which Stowe, Snyder, and Gilpin were working on at this time, expected an input of 151,000,000 punch cards.[45] Programmers asked themselves such questions as: what form does the data have? What is the required form and volume of the output data? How will the output be used? Will the output be reused? Etc. From this, the programmer could craft a block diagram or flow chart of the problem, in classic von Neumann style. The block diagrams helped Stowe, and presumably other programmers, to identify omissions, inconsistencies, and errors. The general order of problem solution was set down, sometimes in great detail: "take this number, add it to that one, divide it by two, and get a percentage."[46] Next came the flow chart procedure.

The most laborious part of the process followed. The flow charts had to be translated into the language of the machine, a process the compiler was designed to circumvent. To accomplish the simple task of adding two numbers, a significant number of lines of code were written.

> The programmer must instruct the machine in its particular code to bring one number from the storage. He must tell it in another operation to add another number from the storage and must further instruct it to take the sum and send it back to storage. These three operations for the computer are implied by one operation in the flow chart. It is frequently difficult to look at the flow chart and say it is going to need 752 lines of coding; it is almost impossible without experience with the particular type of problem. Some of the most innocuous looking boxes on the flow chart may take lines and lines of coding.[47]

It was at this point in the process that a decision could be made between two possible solutions, if multiple possibilities were under investigation.

After coding, an independent programmer at EMCC checked the entire program. At EMCC, the coders tried to have at least two independent

checks made of the code at this point. In the event the problem needed to be put aside for a higher priority concern, Stowe would try to write a report on the work to that point so that when he returned to the problem he would not have to start anew. Then came specific operating instructions. These instructed the operator what to do if something went wrong with the routine, how the tapes were to be mounted, how long was the expected run. Even these instructions were occasionally not enough. Sometimes, the programmer actually accompanied the operator during the program run to be able to handle programming errors if they turned up.

One last point about the conventional programming activity should be noted. As computer people emphasize repeatedly, internal memory space and computer running time were at a premium. To conserve time when running a new problem for the first time, the coder entered rerun aspects (i.e., checkpoints) to the program to prevent having to go back to the beginning of the problem each time an error was corrected and the problem was rerun. "The problem should be arranged in such a logical form that it is necessary to go back only just so far and start over, not go back and completely rerun the entire problem."[48] This procedure saved time, and, perhaps more important, money.

In his talk to the seminar, Stowe emphasized the need for training of coders and programmers, a point also stressed by Mauchly to Remington Rand management at this time. At this time, the programmer needed to be familiar with the logic of the computer system, but did not need to be a computer design specialist. They could be taught programming. A background in particular problems would be a decided advantage. He stopped short of calling for a training program. If nothing else, Stowe's presentation illustrated the attitude of EMCC programming people of the need and desire for more sophisticated programming tools. The Hopper group's work on A-0 was designed to meet this need. But in 1951 A-0 did not go far enough.

Hopper realized that even writing the input specifications for the A-0 compiler was long and cumbersome. She and her group adopted a three-address code for the 12 alpha-decimal characters. They imposed this on top of A-0, and wrote a translator to put on the front end of A-0. Thus, the A-2 compiler came to be.[49] In the previous compilers (A-0 and A-1), the problem analyst prepared the problem for solution and submitted the steps to a coder for preparation, just as was done in coding any problem. In A-2, the analyst circumvented this step through the use of a "Pseudo-code." The Pseudo-code instructions were recorded on tape and read into the computer. The compiler read these instructions and assembled the entire program for running. By this time, the C-10 code was in use on the UNIVAC.

The structure of this compiler resembled the earlier compilers with the added feature of the Pseudo-code.[50] The set of easily accessible subroutines was contained in a library in alphabetical order. Information from tape was read in the same sixty word units, called blocks. Arithmetic was done in floating point. Data was expressed in "two-word" form, which represented the complete numeric quantity to be expressed. The first word contained a numeric quantity without the decimal point and the second word gave the information for the placement of the decimal point. The data and instructions occupied the same block locations as in the previous compilers.

In solving a problem such as

$$Y = e^{-x2} \sin cx$$

Where x ranged between –0.99 and +0.99 in increments of $\Delta x = 0.01$, y needed to be determined for each of 199 values of x throughout the range. Consider only one such calculation in the set to appreciate the flavor of the Pseudo-code. One operation was to increase x by the increment Δx and test to determine if all the values of x throughout the range have been used to calculate the y result required. The description of this operation is "ADD to A LIMIT." Three lines were needed, expressed in a three-address code.

```
1 2 3 4 5 6 7 8 9 10 11 12
AAL  (xi)   (Δx)   (Lx)
1CN  0 0 0 0 ( ≠ OPN #)
2CN  0 0 0 0 ( = OPN #)
```

The first line required 3 relative working storage addresses. The values needed were x, Δx, and Lx, the last being the limit of the range of x. X for this problem was known to be in 000, Δx was in 002, and Lx was in 004. Thus, the first line was expressed

```
AAL 000 002 004
```

On the basis of this instruction, the routine would add x (in 000) to Δx (in 002) and place the sum (the new x) back in 000. This new x would then be tested against the limit of x (LX in 004) to determine whether or not the limit had been reached. All the coder was required to do was send "control" back to the beginning and calculate the next y with the new value of x. This was the purpose of the 1CN line. The symbols ≠ OPN # meant that if the new x was not equal to the limit, control would be transferred to the operation number (OPN #) placed here. Since actual calculation began with Operation 1, this line would read

```
1CN 000 0 00001
```

Five digits were allocated to the operation number because the compiler could handle up to 99,999 operations. When the limit for this problem was reached, control went to the next operation, in this case Operation 11.

```
2CN 000 0 00011
```

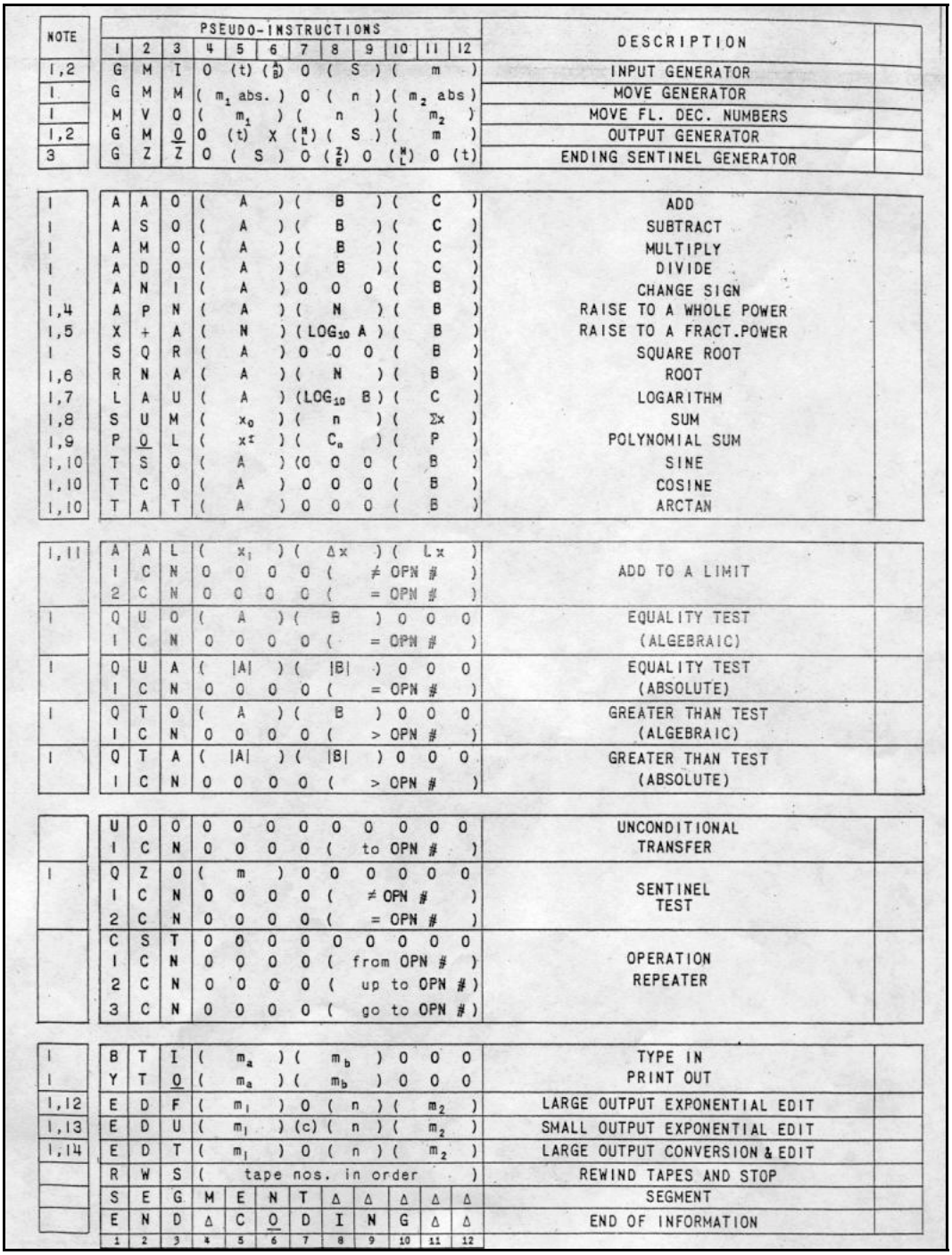
In all, there were 35 Pseudo-code instructions, including an end coding instruction. Figure 4. Later compilers for business program compilation received names, Math-Matic (a form of A-3) and Flow-Matic, in response to requests of the sales staff. Flow-Matic allowed the user to write instructions in English pseudo-code, which UNIVAC I could translate and use to generate the program. Figure 5 shows the set of instructions for Flow-Matic, a compiler that became an important input to the design of the later higher-level programming language COBOL.

Dupont was the first company to use A-0. From there, it spread to the David Taylor Model Basin, where Betty Snyder now resided, to the Army Map Service, and the Census Bureau, all purchasers of UNIVAC I computer systems.[51]

The Mauchly group determinedly tried to convince the community to use these techniques. There was the Philadelphia meeting mentioned above where Hopper spoke on A-0. Mauchly repeatedly addressed groups in various professional settings. Not surprisingly, he participated in the symposium on large-scale digital calculating machinery at Harvard in 1947, where he spoke about "Preparation of Problems for Edvac-type Machines."[52] In September 1949, he presented details about the UNIVAC system to the American Chemical Society. Mauchly was invited to address the Chesapeake Section of the Society of Naval Architects and Marine Engineers in January 1951 on the subject of the computer's value in various engineering problems of interest to the society. Before the American Gas Association-Edison Electric Institute Joint Accounting Conference in April 1953, Mauchly focused on the system's usefulness to business and noted a new training program offered by Remington Rand.[53]

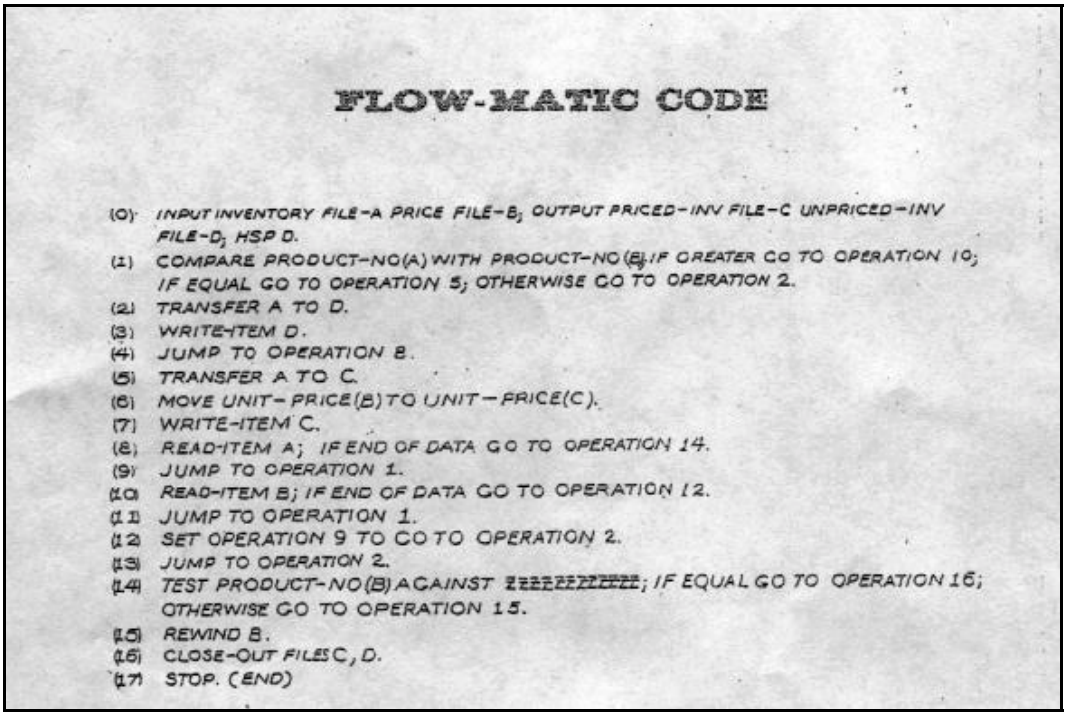
Hopper was on the lecture circuit at least as much as Mauchly. In May 1952, she spoke to the Association for Computing Machinery (ACM) on "The Education of a Computer," a title she used often but with slightly revised text each time to keep up with developments in EMCC. For example, as we noted above, one talk was to the Symposium on Industrial Application for Automatic Computing Equipment held in Kansas City, MO, by the Midwest Research Institute in January 1953. Richard Ridgeway delivered a paper on "Compiling Routines" to a meeting of the ACM in September 1952 in which he did a detailed analysis of EMCC compilers. The Census Bureau organized a workshop on coding and programming for July 1953, attended by several Remington Rand

applications personnel. In September 1955, Mary K. Hawes, Supervisor of Commercial Programming of Remington Rand, presented a talk entitled "Automatic Routines for Commercial Installations" at a meeting of the ACM. After 1955, attendance at meetings by Remington Rand employees became too numerous to catalog here for any useful purpose.



NOTE	PSEUDO-INSTRUCTIONS 1	2	3	4	5	6	7	8	9	10	11	12	DESCRIPTION
1,2	G	M	I	0	(t)	($\frac{A}{B}$)	0	(	S	) (	m	)	INPUT GENERATOR
1	G	M	M	(m_1 abs.)			0	(	n	) (	m_2 abs)		MOVE GENERATOR
1	M	V	0	(	m_1	)	(	n	) (	m_2	)		MOVE FL. DEC. NUMBERS
1,2	G	M	O	0	(t)	X	($\frac{M}{L}$)	(	S	) (	m	)	OUTPUT GENERATOR
3	G	Z	Z	0	(	S	)	0	($\frac{Z}{E}$)	0	($\frac{M}{L}$)	0 (t)	ENDING SENTINEL GENERATOR
1	A	A	0	(	A	) (	B	) (	C	)			ADD
1	A	S	0	(	A	) (	B	) (	C	)			SUBTRACT
1	A	M	0	(	A	) (	B	) (	C	)			MULTIPLY
1	A	D	0	(	A	) (	B	) (	C	)			DIVIDE
1	A	N	I	(	A	) 0	0	0	(	B	)		CHANGE SIGN
1,4	A	P	N	(	A	) (	N	) (	B	)			RAISE TO A WHOLE POWER
1,5	X	+	A	(	N	) (	LOG_{10} A	) (	B	)			RAISE TO A FRACT.POWER
1	S	Q	R	(	A	) 0	0	0	(	B	)		SQUARE ROOT
1,6	R	N	A	(	A	) (	N	) (	B	)			ROOT
1,7	L	A	U	(	A	) (LOG_{10}	B) (	C	)				LOGARITHM
1,8	S	U	M	(	x_0	) (	n	) (	Σx	)			SUM
1,9	P	O	L	(	$x^{\pm}$	) (	C_n	) (	P	)			POLYNOMIAL SUM
1,10	T	S	0	(	A	) (C	0	0	(	B	)		SINE
1,10	T	C	0	(	A	) 0	0	0	(	B	)		COSINE
1,10	T	A	T	(	A	) 0	0	0	(	B	)		ARCTAN
1,11	A	A	L	(	x_1	) (	Δx	) (	L x	)			ADD TO A LIMIT
	1	C	N	0	0	0	0	(	≠ OPN #	)			
	2	C	N	0	0	0	0	(	= OPN #	)			
1	Q	U	0	(	A	) (	B	) 0	0	0			EQUALITY TEST
	1	C	N	0	0	0	0	(	= OPN #	)			(ALGEBRAIC)
1	Q	U	A	(	|A|	) (	|B|	) 0	0	0			EQUALITY TEST
	1	C	N	0	0	0	0	(	= OPN #	)			(ABSOLUTE)
1	Q	T	0	(	A	) (	B	) 0	0	0			GREATER THAN TEST
	1	C	N	0	0	0	0	(	> OPN #	)			(ALGEBRAIC)
1	Q	T	A	(	|A|	) (	|B|	) 0	0	0			GREATER THAN TEST
	1	C	N	0	0	0	0	(	> OPN #	)			(ABSOLUTE)
	U	0	0	0	0	0	0	0	0	0	0	0	UNCONDITIONAL
	1	C	N	0	0	0	0	(	to OPN #	)			TRANSFER
1	Q	Z	0	(	m	) 0	0	0	0	0	0		SENTINEL
	1	C	N	0	0	0	0	(	≠ OPN #	)			TEST
	2	C	N	0	0	0	0	(	= OPN #	)			
	C	S	T	0	0	0	0	0	0	0	0	0	OPERATION
	1	C	N	0	0	0	0	(	from OPN #	)			REPEATER
	2	C	N	0	0	0	0	(	up to OPN #	)			
	3	C	N	0	0	0	0	(	go to OPN #	)			
1	B	T	I	(	m_a	) (	m_b	) 0	0	0			TYPE IN
1	Y	T	O	(	m_a	) (	m_b	) 0	0	0			PRINT OUT
1,12	E	D	F	(	m_1	) 0	(	n	) (	m_2	)		LARGE OUTPUT EXPONENTIAL EDIT
1,13	E	D	U	(	m_1	) (c)	(	n	) (	m_2	)		SMALL OUTPUT EXPONENTIAL EDIT
1,14	E	D	T	(	m_1	) 0	(	n	) (	m_2	)		LARGE OUTPUT CONVERSION & EDIT
	R	W	S	(	tape nos. in order							)	REWIND TAPES AND STOP
	S	E	G	M	E	N	T	Δ	Δ	Δ	Δ	Δ	SEGMENT
	E	N	D	Δ	C	O	D	I	N	G	Δ	Δ	END OF INFORMATION
	1	2	3	4	5	6	7	8	9	10	11	12	

Figure 4. Pseudocode instructions for A-0.



FLOW-MATIC CODE

(0) INPUT INVENTORY FILE-A PRICE FILE-B; OUTPUT PRICED-INV FILE-C UNPRICED-INV FILE-D; HSP D.
(1) COMPARE PRODUCT-NO(A) WITH PRODUCT-NO(B); IF GREATER GO TO OPERATION 10; IF EQUAL GO TO OPERATION 5; OTHERWISE GO TO OPERATION 2.
(2) TRANSFER A TO D.
(3) WRITE-ITEM D.
(4) JUMP TO OPERATION 8.
(5) TRANSFER A TO C.
(6) MOVE UNIT-PRICE(B) TO UNIT-PRICE(C).
(7) WRITE-ITEM C.
(8) READ-ITEM A; IF END OF DATA GO TO OPERATION 14.
(9) JUMP TO OPERATION 1.
(10) READ-ITEM B; IF END OF DATA GO TO OPERATION 12.
(11) JUMP TO OPERATION 1.
(12) SET OPERATION 9 TO GO TO OPERATION 2.
(13) JUMP TO OPERATION 2.
(14) TEST PRODUCT-NO(B) AGAINST ZZZZZZZZZZZZ; IF EQUAL GO TO OPERATION 16; OTHERWISE GO TO OPERATION 15.
(15) REWIND B.
(16) CLOSE-OUT FILES C, D.
(17) STOP. (END)

Figure 5. Flow-matic Code.

All of these examples of codes, compilers, and outreach indicate the high level of software activity within EMCC and later Remington Rand. The company assembled a group of highly effective programmers to provide programs that would make the UNIVAC more attractive to potential customers and to attach customers to Remington Rand. The group was effective, and after the sales force received education about the use of computers, sales began to rise, such that the late 1950s and 1960s ensured the future of Sperry Rand in the computer field.

Almost immediately upon EMCC's joining Remington Rand, difficulties about programming arose within the firm. Remington Rand inaugurated a Department of Program Planning (outside of the Eckert-Mauchly subsidiary) connected with the sales activity, and hired personnel from Harvard Computation Laboratory, Aberdeen Proving Ground, Institute for Advanced Study, and Dahlgren Proving Ground, all people with experience with computers. In addition, meetings about programming training with senior management of Remington Rand by EMCC representatives began on May 5, 1950, when Mitchell, Wistar Brown and Mauchly met with Al Seares to discuss the need for a new training program. Noting that this activity had some urgency to it, EMCC offered cooperation. Brown repeated this plea in a second meeting on May 31

with Millang. Nothing happened for the next two months. In August, members of Department of Program Planning (DPP) visited Philadelphia for six weeks of training, without consultation or indication of purpose, after which they were recalled to New York. On September 1st, David Savidge was appointed to head Program Planning. No mention was made of EMCC. In the fall of 1950, Herbert Mitchell learned that this department had undertaken to "invent, develop and promote a different method of programming" and had organized a course in which they advertised this new method for programming the UNIVAC. Betty Snyder visited the Census Bureau on November 2nd and found advertising materials from DPP sent to Census by Savidge. Mauchly asserted that the advertising materials circulated were misleading and contained serious omissions and errors. He telephoned Millang requesting an appointment to discuss the materials. Millang promised to call back, but he never did. In November, the Philadelphia group also learned from a customer that a "trial course" date had been set. This intelligence set in motion a chain of telephone calls and visits to Philadelphia. Savidge came and met with Mitchell and Hopper. Mitchell and Hopper noted that every page of the text material Savidge brought with him needed correction. When asked about the purpose of the course and the new methods, Savidge invited them to attend the course, though without answering their question. Mitchell (and Mauchly for two sessions) visited the course. The students in the course believed the methods were those of EMCC. Mauchly called for a test of the methods developed by DPP and the EMCC Computation Analysis Laboratory, successor to the Applications Group. This entire contretemps was remarkable when one considers that the DPP group had access only to descriptions of the UNIVAC. Mauchly carefully criticized the materials and DPP, noting that the DPP group was composed of knowledgeable and experienced people, and all he wanted to do was cooperate with them to improve the process, the materials, and training of people to use UNIVAC, and promote Remington Rand.[54] The records do not show how this problem was resolved, but the problem is symptomatic of Remington Rand management's approach to its new subsidiary.[55]

After the acceptance of UNIVAC I by the Census Bureau in March 1951, the Eckert-Mauchly Division began to issue UNIVAC system information in the Remington Rand style. The company published booklets describing the UNIVAC system, problems that could be solved using it and the software developed by EMCC, and the availability of training programs designed to teach customer personnel how to operate and program the system. Besides adopting names for compilers like Math-Matic and Flow-Matic, EMCC described their product as the "UNIVAC Fac-tronic System." This description presented information on the capability of the parts of the system and how they related to each other, the reliability of the system, sorting with UNIVAC, the range of applications programs availability, and the seminars and training activities designed for the

customer.[56] The specific applications discussed in detail were used at the Census, materials control in manufacturing, from the development of a production schedule to service schedules, and payroll preparation. EMCC indicated that a wider range of applications was possible with the Factronic system. In the commercial area, they cited programs for billing, sorting, collating, interfiling, nearly 100 statistical and accounting report possibilities, and all the elements of reporting taxes, social security, and deductions necessary in payroll accounts. EMCC had developed programs for statistical analysis for both military and civilian users. Various logistics programs included production scheduling, building requirements, stock control, etc. The UNIVAC could also be used for scientific and engineering applications, such as the solution of matrix algebra problems, several elliptic partial differential equations, including LaPlace's and Poisson's equations with various shaped boundaries and boundary values, and the rapid reduction of test data from experiments. The standard library programs could be compiled using Math-Matic and Flow-Matic in the mid-1950s. The customer's personnel could design specialized programs using some of the library of routines developed by EMCC, after they mastered the system in the training courses offered by Remington Rand.

All of these programs, both systems software and applications, had been developed by the mid-1950s by the various programming groups of EMCC and Remington Rand to be delivered with the UNIVAC I. Sometimes the customer could not wait for Remington Rand to produce programs they needed for their operations. When this happened, the customer using systems like Flow-Matic could develop the needed programs. EMCC and Remington Rand were very conscious that sales of UNIVAC would only happen if applications programs came with the UNIVAC system, and the company expended a substantial effort in the years 1947 to 1956 on coding techniques and programming systems.

Arthur L. Norberg, "Software Development at the Eckert-Mauchly Computer Company Between 1947 and 1955," *Iterations: An Interdisciplinary Journal of Software History* 2 (December 31, 2003): 1-26.

---

[1] See, for example, Paul E. Ceruzzi, *A History of Modern Computing*, (Cambridge, MA: MIT Press, 1998), Chapter 3; Martin Campbell-Kelly, "Programming the EDSAC: Early Programming Activity at the University of Cambridge," *Annals of the History of Computing*, *2*(1980): 7-36 and *From Airline Reservations to Sonic the Hedgehog (History of Computing Series)*, op. cit.; Emerson Pugh, *Building IBM*, op. cit.; and Stuart S. Shapiro, "Computer Software as Technology: An Examination of Technological Development," Ph.D. dissertation, 1990, Carnegie-Mellon University, Chapter 2.

[2] EMCC employed the name EDVAC II to distinguish their company design for a computer system from the University of Pennsylvania Moore

School EDVAC design, which they helped design before leaving the Moore School in 1946. Later, the EDVAC II became the UNIVAC I.
[3] W. Barkley Fritz, "The Women of ENIAC," *Annals of the History of Computing, 18*(Fall 1996): 13-28.
[4] UNIVAC Conference Transcript, OH 200, CBI. Comment by Frances E. Holberton, p. 52.
[5] Fritz, "The Women of ENIAC," op. cit., p. 18-19. This article contains substantial information about Bartik, her training, and the programming of ENIAC.
[6] Interview with Frances E. Holberton, OH 50, passim, CBI.
[7] UNIVAC Conference, op. cit., p. 65.
[8] Ibid., p. 68.
[9] T. W. Brown to H. L. Strauss, "Report on Applications Dept.," 31 March 1949, Sperry Corporation Records, Acquisition 1825, Box 83, Chronological File, Hagley.
[10] Association f or Computing Machinery, "A Quarter-Century View, ACM71," (New York, 1971), and "An Analysis of the Eckert-Mauchly Computer Corporation," op. cit., Biographical Summary Section.
[11] Ibid.
[12] "Conferences on EDVAC II Design," March 11, 12, 1947, Sperry Corporation Records, Acquisition 1825, Box 37, Hagley.
[13] Ibid.
[14] Ibid.
[15] Ibid.
[16] Data taken from a comparative chart in the Frances E. Holberton Papers, CBI 94, Box 13, File: UNIVAC Code Development.
[17] Ibid.
[18] Ibid.
[19] Nancy Stern, *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*, (Bedford, MA: Digital Press, 1981), p. 133.
[20] C. J. Bashe, L. R. Johnson, J. H. Palmer, and E. W. Pugh, *IBM's EarlyComputers*, (Cambridge, MA: MIT Press, 1986), pp. 138-142.
[21] Ibid., pp. 143-44.
[22] The Holberton papers at CBI contain outlines and some lectures from the course offered to EMCC engineers in early 1950, with lectures by Herbert Mitchell, Grace Hopper, and Betty Snyder.
[23] "Training Course for EMCC's Engineers," spring 1950, Holberton Papers, Box 5.
[24] Mauchly, "Chronology," October/November 1947, Mauchly Papers, University of Pennsylvania Archives, Box 3:C:1, Folder 5.
[25] Lukoff, *From Dits to Bits*, op. cit., passim on Eckert's presence at the office and how he interacted with personnel.
[26] B. Snyder and H. M. Livingston, "Coding of a Laplace Boundary Value Program for the UNIVAC," *MTAC, 3*(January 1949): 341-50.

[27] H. Liebmann, "Die ausgenährte Ermittlung harmonischer Funktionen und konformer Abbildungen (nach Ideen von Boltzmann and Jacobi)," Adad. D. Wissen., Munich, *Berichte*, 1918, pp. 385-416.
[28] "Coding of a Laplace Boundary Value Problem," EMCC 1948, Holberton Papers, Box 13.
[29] G. M. Hopper, "The Education of a Computer," *Proceedings of the Association for Computing Machinery* (Pittsburgh meeting, May 2 and 3, 1952), (New York: ACM, 1952), pp. 243-249. Hopper used this title for several presentations in this period, but she claimed the presentations differed. A comparison of her published paper "The Education of a Computer," with "The Education of a Computer" in Proceedings, Symposium on Industrial Applications of Automatic Computing Equipment, Midwest Research Institute, Kansas City, Missouri, January 8 and 9, 1953 (copy at CBI In Hopper file), illustrates that the basis of the many papers was the description of the UNIVAC system, but the examples of what the system could calculate were chosen to appeal to the audience Hopper was addressing. The description in the text can be found in either article.
[30] D. Knuth and L. T. Prado, "The Early Development of Programming Languages," *Encyclopedia of Computer Science and Technology*, *7* (1977): 419-493.
[31] "Early American 'Compilers'," Ibid.
[32] M. V. Wilkes, D. J. Wheeler, and S. Gill, *The Preparation of Programs for an Electronic Digital Computer, with special reference to the EDSAC and the use of a library of subroutines*, (Cambridge, MA: Addison-Wesley, 1951; reprint edition, Los Angeles: Tomash Publishers, 1982). See Martin Campbell-Kelly's introduction to the reprint edition for an assessment of the significance of this publication.
[33] Knuth and Pardo, op. cit., p 434.
[34] "UNIVAC Short Code [Instruction Manual]," "Preface," Computer Product Manuals Collection, CBI 60, Box 186.
[35] "UNIVAC Short Code," passim, and Sammet, *Programming Languages*, op. cit., pp. 129-130.
[36] G. M. Hopper, "Keynote Address," In Richard L. Wexelblat, ed., *History of Programming Languages*, (New York: Academic Press, 1981), pp. 7-20, p. 10.
[37] Jean E. Sammet, *Programming Languages: History and Fundamentals*, (Englewood Cliffs, NJ: Prentice-Hall, 1969), p. 12.
[38] Ibid.
[39] The following description is taken from the Instruction manual "A-0 Compiler," Computer Products Manual Collection, CBI 60, Box 205.
[40] Hopper, "Keynote," op. cit., p. 11.
[41] "A-0 Compiler" Instruction Manual, op. cit., p, 3.
[42] Hopper, "Keynote," p. 12.

[43] Richard K. Ridgway, "Compiling Routines," presented at ACM meeting September 8-9, 1952, Holberton Papers, CBI 94, Box 5.
[44] L. Stowe, "Programming," *Summary of Papers Presented at the Seminar on Data Handling and Automatic Computer*, 26 February to 6 March 1951, Office of Naval Research, US Government Computing Collection, CBI 63, Box 2.
[45] "Progress Report on Bureau of the Census Problem, 4 January 1951," Holberton Papers, CBI 94, Box 23. Stowe, "Programming," p. 80.
[46] Ibid., p. 81.
[47] Ibid.
[48] Ibid., p. 83.
[49] Hopper's group included James McGarvey, Adele "Millie" Koss, F. M. Delaney, Margaret H. Harper, and Richard K. Ridgway. Hopper, "Keynote," op. cit., pp. 12-13.
[50] The following description comes from a Remington Rand "Instruction Manual: The A-2 Compiler System" published in 1955 In the Holberton Papers, CBI 94, Box 18.
[51] Hopper, "Keynote," op. cit., p. 14.
[52] *Proceedings*, Harvard 1948; reprinted as Volume 7 In the CBI Reprint Series, MIT Press and Tomash Publishers, 1985.
[53] Mauchly Papers, Box 3:C:12, Folder 258, University of Pennsylvania Archives.
[54] Mauchly to A. N. Seares, Internal evidence points to a series of memoranda written toward the very end of 1950, Mauchly Papers, Box 3:C:6, Folder 140, University of Pennsylvania Archives.
[55] When Remington Rand agreed to purchase EMCC, it was with the understanding that EMCC would be an independent part of the Remington Rand company, i.e., a subsidiary, reporting directly to the head of the company James H. Rand.
[56] One of many examples that can be found in the Computer Product Literature Collection (CBI 12) is a 20-page brochure "UNIVAC Fac-tronic System by Remington Rand Inc. Eckert-Mauchly Division," Box 99. While there is no date on the brochure, internal evidence about available subsystems suggests 1953.