agreement with Watson is not complete, and a value known to be in error in Watson is correct in the table under review.

J. C. P. MILLER

23 Bedford Square
London W.C. 1

[1] G. N. WATSON, *Treatise on the Theory of Bessel Functions*, Cambridge, 1922, second ed. 1944 [*MTAC*, v. 2, p. 49–51.]

94[V].—BALLISTIC RESEARCH LABORATORIES, Aberdeen Proving Ground, Md. *Supersonic Flow past Cone Cylinders.* [See Note **113.**]

AUTOMATIC COMPUTING MACHINERY

Edited by the Staff of the Machine Development Laboratory of the National Bureau of Standards. Correspondence regarding the Section should be directed to Dr. E. W. CANNON, 225 Far West Building, National Bureau of Standards, Washington 25, D. C.

TECHNICAL DEVELOPMENTS

Characteristics of the Institute for Numerical Analysis Computer

In January 1949, members of the staff of the Institute for Numerical Analysis[1] began the development and construction of a high-speed electronic digital computer. As of December 1, 1949, the central computer was approximately eighty per cent completed. The group responsible for the building of this machine is composed of, besides the author, three engineers, three junior engineers, and four technicians. In addition, one mathematician is assigned to the coding and programming of problems to be run on the machine.

Information is stored and processed in the computer in units called words, a word consisting of 41 binary digits. This word length is determined by the number of words which can be stored in the computer's high-speed memory.

Words in the machine sense may represent (1) numerical information, (2) instructions to the computer, and (3) alphabetic information.

In the case of numerical information, one binary digit of a word is used for the sign and 40 binary digits are available for numerical data. Numbers are stored in the memory as absolute value and sign. In the arithmetic unit, negative numbers may be converted to complementary[2] form to keep the operational algorithms relatively simple. Thus, negative numbers involved in addition, subtraction, and compare are complemented upon arrival in the arithmetic unit. In the multiplication, extract, input, and output commands, negative numbers are not complemented.

Numbers may be represented in many different ways. For example, a word may represent a signed-binary number lying somewhere between -2^{40} and $+2^{40}$, or the binary point may be ahead of the most significant digit in which case a word lies in the range -1 to $+1$; the built-in arithmetic operations handle numbers in either of these forms. The word may, on the other hand, represent a signed-ten-decimal-digit number where each decimal digit is represented as a four-digit-binary number. A floating representation may

be used where the first digit represents the sign, the next eight digits represent the exponent b, and the next 32 digits represent the significant digits of the number in binary form. A floating decimal representation may also be used giving numbers of eight significant digits ranging in absolute value from about 10^{-50} to 10^{+50}. More than one word can be used to represent a number to effect much greater precision or range of values.

Floating operations have been coded (as will be explained later in this paper) to provide for the addition, subtraction, multiplication, or division of two numbers of the form $a \cdot 2^b$ (with a and b stored in the same address). All four floating operations involve about 87 instructions. These floating operations are performed in about 3, 3, 6, and 14 milliseconds, respectively. Compare these times with 64, 64, and 384 *micro*-seconds, the times required for doing ordinary binary addition, subtraction, and multiplication.

Instructions are subclassified into three classes. These three classes are *command words*, *control words*, and *code words*.

The *command words*, commonly called commands, are explicitly "understood" and "obeyed" by the computer. A command causes the computer to perform a specific operation and gives the necessary information about where to get the needed data and what to do with the results. At present, the command system used by the Institute's computer consists of a set of thirteen commands.[3] Eight of the thirteen are what might be termed basic commands; the other five are variations of these eight. Such a command system is known as a four-five address code, with four addresses generally in the command word and the fifth address in a control counter. The function, or operation, of a particular command is denoted by F, the four addresses of the command by α, β, γ, and δ, and the fifth address by ϵ which normally determines the address of the next command to be obeyed.

The size of the memory determines the length of each address which, in the case of a 512 word memory, is nine binary digits. Thus, 36 binary digits are used to denote the four addresses, α, β, γ, and δ. Four digits are used to denote F; of these four, three are used to define the eight basic command words, and one is used to denote modifications of the eight. There is one spare digit in the command word.

The thirteen operations, or functions, of the Institute's computer, as well as the meanings of the addresses for the various command words, are given in Table 1.

In order to change automatically the course of operation in the calculator when certain bounds have been reached,[4] there are conditional and unconditional transfers of control commands. A conditional transfer is accomplished with compare commands. An unconditional transfer is accomplished with certain special commands (A_1, S_1, and M_1), wherein the fourth address, δ, determines the next command.

In the table, special compare might well be called absolute compare in that the absolute values of the numbers are compared. Since the result of the subtraction in the compare operation is put back into the memory, this command can be used to obtain the absolute value of a number in one operation by comparing the number with zero. The compare command can also be used for an operation called tally, as follows: Assume that it is desired to repeat a routine fifty times stored in memory locations 31 to 37, inclusive.

TABLE 1

Meanings of the Addresses for the Commands

Command	α	β	γ	δ	F
Add	Address of Augend	Address of Addend	Address of Sum	Address of Next Command if Overflow	A
Special Add	Address of Augend	Address of Addend	Address of Sum	Address of Next Command	A_1
Subtract	Address of Minuend	Address of Subtrahend	Address of Difference	Address of Next Command if Overflow	S
Special Subtract	Address of Minuend	Address of Subtrahend	Address of Difference	Address of Next Command	S_1
Multiply	Address of Multiplier	Address of Multiplicand	Address of Product Rounded Off		M
Special Multiply	Address of Multiplier	Address of Multiplicand	Address of Product Rounded Off	Address of Next Command	M_1
Product	Address of Multiplier	Address of Multiplicand	Address of Most Significant Part of Product	Address of Least Significant Part of Product	P
Compare	Address of Minuend	Address of Subtrahend	Address of Difference	Address of Next Command if Difference is Non-negative	C
Special Compare	Address of Minuend	Address of Subtrahend	Address of Difference of Absolute Values	Address of Next Command if Difference of Absolute Values is Non-negative	C_1
Extract	Address of Extractor (Determines Digits to be Extracted)	Address of Extractee	Address of Extracted and Shifted Result	If Second Digit of δ is {0—Shift Left; 1—Shift Right} Other Digits Tell Number of Places to Shift	E
Input	Address of Incoming Information	Drum Address if used		Selects Input Device	I
Special Input	(Incoming Information Goes to Address ϵ)			Selects Input Device	I_1
Output	Address of Outgoing Information	Drum Address if used		Selects Output Device	O

At the beginning of the routine, let the number "49" be placed in address 10. Suppose that address 1 stores the number "1." Place the command

10, 1, 10, 31, C

in address 38. The first time this command immediately following the routine is obeyed, the number in address 10 will be reduced to "48," and the next command will come from address 31. Each time the routine is repeated, the command in address 38 will be executed, with a reduction of the number in address 10 by one. After the routine has been performed 49 times, the number in address 10 will be zero; the 50th time the difference formed by the compare order will be negative, and the next command will come from address 39 (the value of ϵ). Thus, the desired routine will be repeated exactly fifty times.

In the case of the normal addition and subtraction commands, overflow[5] is automatically detected. An extra digit is provided in the arithmetic unit in the most significant end of the A and M[6] registers so that for normal addition and subtraction commands a "1" in this position will cause the next command to come from address δ instead of address ϵ. In the case of the compare command, the proper result of the subtraction is in the A register, but here, as in the addition and subtraction commands, the most significant digit (overflow digit) is not put back into the memory. In the compare command there is no means of detecting overflow.

The extract command provides for obtaining the logical product of two words[7] and shifting the result an arbitrary amount; it may also be used to delete arbitrary parts of a word. Its primary purpose is to assist automatically in fabricating new commands during the computation and to sort out the exponent from the significant figures in floating-point operations.

Special input is the command used to insert information into the computer when it is first put into operation. A word consisting entirely of zeros in the F portion is used to designate special input. Thus, when the memory is cleared, every word is a special input command, and, with these commands, the destination of the incoming data is determined by ϵ. After ϵ has been increased by "1," it determines the source of the next command. Once started, the calculator will count through its complete high-speed memory and read in information from the teletype tape.[8] After the memory has been filled, the ϵ counter steps to zero, and the calculator obeys the command stored in that position.

The input and output commands may specify the magnetic drum as a source or destination of words to be transferred. The drum itself will be about eight inches in diameter and two feet long and will hold about 8,192 words.[9] Initially only one word at a time can be transferred between it and the high-speed memory. The drum will rotate at 3600 revolutions per minute which means that the average access time for a word will be 8 milliseconds. As soon as possible after the computer is put into operation, counting facilities will be added to the control to enable the high-speed memory to be operated in synchronism with the drum. This will make it possible to transfer a whole vector in one revolution of the drum, thus greatly decreasing the average access time.

In general, the drum will serve as an auxiliary storage for numbers, instructions, and function tables. The drum is used instead of extra tape units because it offers better accessibility to information and requires no manual handling. The drum does not hold as much information as a magnetic tape unit, but its size seems adequate for the purposes mentioned above. For greater storage several drums may be operated in synchronism.[10]

The second class of instructions, *control words*, are not directly obeyed by the calculator, nor are they a direct part of the calculation; yet in various ways, they control the course of the computation or enter into the arithmetic-like operations which are performed upon command words. Control words may serve as parameters which determine the number of repetitions of certain routines; they may be the bounds used to stop certain computational processes; or they may serve as factors in the logical products or extraction operations.

The third class of instructions, *code words*,[11] specify (usually in one word)

a whole sequence of procedures for the computer to follow. Thus, one may specify, for example, a scalar multiplication with only one code word. This code word is made up of parameters which specify the common factor (that is, which specify the address in the high-speed memory), the location of the elements of the vector (say, by specifying the address of the first element and the number of terms in the vector), and the location for the result.

When prepared for the computer, a problem consists of a sequence of words called a *main routine.* This main routine is made up of instructions (commands, code words, and control words) together with the numerical constants appropriate to the problem.

The code words in the main routine contain the parameters or addresses necessary to call into action other sequences of instructions, usually called *subroutines.* Subroutines (and code words) exist for such procedures as floating operations, standard iteration formulae, vector operations, integration formulae, and so forth, which are frequently used in the course of doing a computation. A subroutine may itself contain code words which call into use other subroutines. (All of the more frequently-used subroutines will be stored on the magnetic drum, thus being comparatively easily accessible.)

A routine known as the *interpretation routine* keeps track of the place in the main routine, causes segments of the main routine to be brought into the high speed memory, inspects each successive instruction in the main routine to see if it is a command (in which case it is obeyed) or a code word (in which case the interpretation routine extracts an entry[12] from the code word and sends the computer to the appropriate subroutine). A subroutine may itself change the code word being considered by the interpretation routine and, thus, cause the computer to carry out other subordinate activities before going on with the main routine.

Once the appropriate subroutines are established, the task of coding a complex problem is very much simpler. For example, the problem of solving systems of simultaneous linear equations of orders up to 125 by the elimination method has been worked out with the five main subroutines listed in Table 2. The arithmetic operations performed by these subroutines are of

TABLE 2

Routines for Solving Simultaneous Linear Equations

Name	Code	No. of Instructions	Time for Execution in μsecs.	Purpose
Vector Input	α, γ, n, VI	8	$(n + \frac{1}{16})8000$	n words transferred from addresses $\gamma + i$ on drum to addresses $\alpha + i$ of high-speed memory, $i = 0, 1, \cdots, n - 1$.
Vector Output	α, γ, n, VO	9	$(n + \frac{1}{16})8000$	Converse of VI, that is, transfer from high-speed memory to drum.
Floating Operations	α, β, γ, A α, β, γ, S α, β, γ, M α, β, γ, D	88	3,000 3,000 6,000 14,000	Adds, subtracts, multiplies, or divides the operands and puts result in γ.
Vector Constant Product	α, β, γ, n, VC	10	$6500n + 448$	Multiplies word in β by words in $\alpha + i$; answers go into $\gamma + i$, $i = 0, 1, \cdots, n - 1$.
Vector Subtraction	α, β, γ, n, VS	10	$3500n + 448$	Subtracts words in $\beta + i$ from those in $\alpha + i$, answers go into $\gamma + i$, $i = 0, 1, \cdots, n - 1$.

the floating binary type, and each such arithmetic operation is in itself a subroutine. By using code words the coding for this simultaneous linear equation problem is reduced to laying out a sequence of about thirty instructions. It is convenient to have all routines and constants, and two rows of the matrix stored in the high-speed memory; the figure of 125 is based on a 512 word memory. The approximate time required to solve a set of sixty equations under various conditions is given in Table 3. In fixed-point operation, a division routine replaces the floating routines of Table 2. The largest pivot may be used in each reduction, and scale factors may have to be introduced. Table 2, Table 3, and some of the routines were worked out by ROSELYN LIPKIS of the Machine Development Unit at the Institute for Numerical Analysis.

TABLE 3

Time Required to Solve Sixty Simultaneous Linear Equations[13]

	Fixed Binary Point		Floating Binary Point	
	Synchronized Drum	Non-Synchronized Drum	Synchronized Drum	Non-Synchronized Drum
Computing Time	3.5 min.	3.5 min.	19 min.	19 min.
Transfer Time (to and from the drum)	1.0 min.	30.0 min.	1 min.	30 min.
Totals	4.5 min.	33.5 min.	20 min.	49 min.

H. D. HUSKEY

Institute for Numerical Analysis
Univ. of California, Los Angeles

[1] The Institute is one of four sections of the National Applied Mathematics Laboratories of the National Bureau of Standards. It is located on the campus of the University of California at Los Angeles. The computing machine discussed in this paper is financed by the Air Materiel Command of the United States Air Force.

[2] $-N$ is converted to $2^{42} - N$ or $2^{46} - N$, depending upon whether the size of the memory is 512 or 1024 words, respectively.

[3] These commands are a variation of a set proposed by E. F. MOORE while he was working in the National Applied Mathematics Laboratories of the National Bureau of Standards.

[4] For example, in the integration of the exterior ballistic equation, the procedure must change when a shell has completed its flight; or in a square root iteration, the procedure changes when two successive iterants are sufficiently close together.

[5] Addition, subtraction, and compare may produce results which exceed the capacity of the memory cells.

[6] This extra digit in M is needed for the complement process.

[7] Or it may be used for deleting arbitrary parts of a word.

[8] At a later date, magnetic tape may also be used for inserting information into the computer.

[9] Professor P. MORTON of the University of California at Berkeley is constructing such a drum.

[10] Professor F. C. WILLIAMS of Manchester University, England, has operated a drum in synchronism with a master oscillator.

[11] Code words have been variously termed abbreviated code instructions, quasi-commands, shorthand commands, abbreviated commands, and coded commands. The term "code word" has been selected in preference to these other terms by the author to distinguish more clearly this type of instruction from the explicit commands.

[12] An entry is the address of the command in the subroutine which should be obeyed first.

[13] This table is based on storing three rows of the matrix in the high-speed memory. For more than approximately eighty equations, only two rows can be stored, and input-output is increased by fifty per cent.