NATIONAL BUREAU OF STANDARDS REPORT

1951

SYSTEM SPECIFICATIONS FOR THE DYSEAC

By

Alan L. Leiner
Electronic Computers Laboratory

September 1952

U. S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

U. S. DEPARTMENT OF COMMERCE
Charles Sawyer, *Secretary*

NATIONAL BUREAU OF STANDARDS
A. V. Astin, *Director*

THE NATIONAL BUREAU OF STANDARDS

The scope of activities of the National Bureau of Standards is suggested in the following listing of the divisions and sections engaged in technical work. In general, each section is engaged in specialized research, development, and engineering in the field indicated by its title. A brief description of the activities, and of the resultant reports and publications, appears on the inside of the back cover of this report.

Electricity. Resistance Measurements. Inductance and Capacitance. Electrical Instruments. Magnetic Measurements. Applied Electricity. Electrochemistry.

Optics and Metrology. Photometry and Colorimetry. Optical Instruments. Photographic Technology. Length. Gage.

Heat and Power. Temperature Measurements. Thermodynamics. Cryogenics. Engines and Lubrication. Engine Fuels. Cryogenic Engineering.

Atomic and Radiation Physics. Spectroscopy. Radiometry. Mass Spectrometry. Solid State Physics. Electron Physics. Atomic Physics. Neutron Measurements. Infrared Spectroscopy. Nuclear Physics. Radioactivity. X-Rays. Betatron. Nucleonic Instrumentation. Radiological Equipment. Atomic Energy Commission Instruments Branch.

Chemistry. Organic Coatings. Surface Chemistry. Organic Chemistry. Analytical Chemistry. Inorganic Chemistry. Electrodeposition. Gas Chemistry. Physical Chemistry. Thermochemistry. Spectrochemistry. Pure Substances.

Mechanics. Sound. Mechanical Instruments. Aerodynamics. Engineering Mechanics. Hydraulies. Mass. Capacity, Density, and Fluid Meters.

Organic and Fibrous Materials. Rubber. Textiles. Paper. Leather. Testing and Specifications. Polymer Structure. Organic Plastics. Dental Research.

Metallurgy. Thermal Metallurgy. Chemical Metallurgy. Mechanical Metallurgy. Corrosion.

Mineral Products. Porcelain and Pottery. Glass. Refractories. Enameled Metals. Concreting Materials. Constitution and Microstructure. Chemistry of Mineral Products.

Building Technology. Structural Engineering. Fire Protection. Heating and Air Conditioning. Floor, Roof, and Wall Coverings. Codes and Specifications.

Applied Mathematics. Numerical Analysis. Computation. Statistical Engineering. Machine Development.

Electronics. Engineering Electronics. Electron Tubes. Electronic Computers. Electronic Instrumentation.

Radio Propagation. Upper Atmosphere Research. Ionospheric Research. Regular Propagation Services. Frequency Utilization Research. Tropospheric Propagation Research. High Frequency Standards. Microwave Standards.

Ordnance Development.
Electromechanical Ordnance.
Electronic Ordnance.
These three divisions are engaged in a broad program of research and development in advanced ordnance. Activities include basic and applied research, engineering, pilot production, field testing, and evaluation of a wide variety of ordnance matériel. Special skills and facilities of other NBS divisions also contribute to this program. The activity is sponsored by the Department of Defense.

Missile Development. Missile research and development: engineering, dynamics, intelligence, instrumentation, evaluation. Combustion in jet engines. These activities are sponsored by the Department of Defense.

- Office of Basic Instrumentation
- Office of Weights and Measures.

NATIONAL BUREAU OF STANDARDS REPORT

NBS PROJECT	NBS REPORT
1203-34-5705	1951

September 1952

SYSTEM SPECIFICATIONS FOR THE DYSEAC

By

Alan L. Leiner

Electronics Division
Electronic Computers Laboratory

FOREWORD

Under the sponsorship of a tri-service steering committee of the Defense Department, a new digital data-processor is now being developed by the Electronic Computers Laboratory. This machine is intended to serve as a central control unit in a data-handling network, and also as a testing device for determining the operational properties of the network.

It is a full-scale, general-purpose, serial machine, operating at a megacycle repetition rate. Like the SEAC, it contains a serial acoustic delay-line memory with a minimum capacity of 500 words which can be expanded to 4000. However, it differs from the SEAC in the following particulars: (1) it contains an expanded repertoire of arithmetic operations; (2) it contains provision for annexing a greater variety of external storage and input-output devices; and (3) it is capable of carrying out automatically, in an integrated fashion, concurrent operations involving high-speed input-output and processing of data. For example, it will be capable of receiving non-synchronous data from telemetering sources (or of displaying and dispatching results) even while it is carrying out a complex program of data-processing. This last feature will make the new machine considerably more flexible and far more powerful than the SEAC, for complex data-handling operations.

Furthermore, the machine's high-speed memory will be almost immediately accessible to other units of the data-handling network. In fact, the machine can be used as a storage reservoir which can be shared among the other units of the network and which can be interrogated (or recorded in) by them independently, without disrupting the course of the internal data-processing program. Conversely, the machine can check up periodically on other units of the network, and re-direct them as the need arises in the same flexible manner.

The physical realization of the machine-system outlined in this report is being accomplished by assemblies of two main types of "packages" developed by this Laboratory. This packaged construction is aimed at simplifying both maintenance and supply procedures.

S. N. Alexander, Chief
Electronic Computers Laboratory

A. V. Astin
Director

CONTENTS

CONTENTS

LIST OF TABLES

SYSTEM SPECIFICATIONS FOR THE DYSEAC

By

Alan L. Leiner

I. INTRODUCTION

1. Summary of operating specifications

The DYSEAC is a full-scale general-purpose data-processor which operates at a basic repetition rate of one megacycle and uses the serial mode of data representation. Its internal high-speed memory is composed of mercury acoustic delay-line tanks containing eight words each, with a maximum access time of 384 microseconds. The memory capacity is, at the minimum, 512 words (stored in a 64-tank cabinet) and it can be expanded to 4096 words (stored in eight such cabinets). A list of the types of operations performed by the machine and their approximate average rates of execution is given below:

Operation	Code-Symbol	Time (milliseconds), including access time to the memory
Addition	2	0.9
Subtraction	1	0.9
Accumulate-and-overflow-check	4	0.7
Accumulate-and-store	5	0.9
Summation	3	0.8*
Multiplication, major (rounded)	A	3.0
Multiplication, minor	B	3.0
Division	C	3.0
Shift	8	1.1**
Justify	9	2.0**
Logical Transfer	D	0.9
Comparison, algebraic	6	0.7
Comparison, absolute	7	0.7
File, unconditional	F	0.4
Breakpoint File	E	0.4
Input-Output	0	0.3***

Notes: *Plus 0.05 msec per word.
**For half word-length shifts.
***Internal program time only.

2. Comparison with the SEAC

2.1 Operations:

Of the 16 types of operations performed by the DYSEAC, five types are the same as those in the SEAC, five types represent variations or expansions of their SEAC counterparts, and six are either entirely new or completely revised. In the latter class are the two Accumulation instructions and the Summation, Justify, and high-speed Shift operations which have been provided to facilitate coding and to speed up the performance of various arithmetic and data-handling processes. In the expanded class of operations, the facility for providing both major and minor products (or both quotient and remainder) in little more than a single multiplication (or division) time should be noted. Improved base-point and relative-address arrangements have also been provided.

The DYSEAC's input-output type of operation represents a completely new departure from the SEAC system in several respects. First, it can be carried out at the same time as, and independent of, computing operations. Second, this concurrent loading or printing-out of the memory can be made to apply to any word or group of words in the memory without any restriction as to location or quantity. Third, the course of a processing program can be tied to (and sequenced by) these input-output operations whenever desirable. Whenever the program is primarily concerned with data which is being transferred into or out of the system, it can be coupled to the input-output instructions, by means of certain automatic interlocks and program-jumping features, in such a way as to be automatically sequenced by the progress of the external input-output operations. This enables optimum use to be made of the machine's concurrent operating ability; at the same time, it relieves the programmer of the responsibility for working out, exactly, the timing interrelationship between the non-synchronous internal and external portions of the processing procedure.

Another important respect in which the DYSEAC differs from the SEAC is in the greatly extended repertoire of manual and monitoring operations. These facilities provide the operator (or, alternatively, an automatic external device) with the power of setting up and ordering a wide variety of special interpolated operations into the program being carried out. These include printing-out or loading various memory locations or storage registers in the computer, substitution of new instructions or new words instead of those indicated in the program, and so forth. In combination with the input-output features mentioned above, these facilities make it possible for the machine to communicate in an extremely flexible manner with the person operating it---and even to share its high-speed memory (and consequently any other part of its internal processing or external storage facilities) freely among a battery of remotely located external data-handling devices.

2.2 Pulse-codes for representing information:

Like the SEAC, the DYSEAC operates on a minor cycle of 48 microseconds and employs a pure binary word representation consisting of 45 binary numerical digits plus one sign digit. Numbers are expressed in the form: absolute value plus algebraic sign. The sign (+) is represented by 0, and the sign (-) is represented by 1. For details on the word-format, see Table 1.

DYSEAC also employs a three-address instruction system,-- that is, each instruction word contains three address numbers denoted as alpha, beta, and gamma, which in most cases have the following significance: alpha denotes the location (in the acoustic memory) of the first operand, beta denotes the location of the second operand, and gamma denotes the location to which the result is to be sent. Each address-number referring to these locations is represented, within the instruction word, by a 12-digit binary number-sequence composed of three parts:

(1) The smallest three digits indicate the position of the word in the tank, i.e., the timing number (0 to 7).

(2) The next six digits are the binary representation of the tank number within the cabinet of 64 tanks (0 to 63).

(3) The largest three digits designate which one of the eight possible cabinets is referred to (0 to 7).

These three combined parts form the binary representation of the address-number (0 to 4095), successive numbers generally indicating adjacent locations in the same tank. Successive instructions are generally located in consecutively-numbered memory locations.

The total information content of an instruction word is distributed as follows:

12 digits contain the address alpha,
12 digits contain the address beta,
12 digits contain the address gamma,
4 digits contain relative-address and program-sequencing information (described in Table 3),
4 digits contain the code-symbol specifying the type of operation to be performed,
1 digit contains a monitor signal (whose utilization is described in Section V, paragraph 3), and
1 digit is a check digit (whose utilization is described in paragraph 3 immediately below).

Table 1. Word-Format (Internal)

Each Instruction-Word Contains:		
Binary Digits	Digit Position in Word*	Content
1 bit	P-46	Check digit
12 bits	P-45 through P-34	Address alpha
12 bits	P-33 through P-22	Address beta
12 bits	P-21 through P-10	Address gamma
4 bits	P-9 through P-6	Relative-address and program-sequencing information
4 bits	P-5 through P-2	Operation code-symbol
1 bit	P-1	Monitor signal
46 bits total		

Each Number-Word Contains:		
1 bit	P-46	Check digit
44 bits	P-45 through P-2	Numerical digits
1 bit	P-1	Algebraic sign digit
46 bits total		

*The successive binary digit positions in a word are denoted by P-1, P-2, ... P-46, counting from right to left (least significant to most significant).

3. Checking features

Unlike SEAC, the new machine contains provision for automatically checking the performance of the high-speed memory. The design of this checking feature is based on observed experience of the operating reliability of the SEAC's acoustic delay-line memory. Its inclusion is based on the premise, supported by SEAC experience, that the faults most commonly encountered with this type of memory are those which result in an occasional picking-up or dropping of a single binary digit in some part of the memory. The checking feature in DYSEAC is designed to guard against the possibility that such errors may pass by undetected.

The system operates on the principle of inserting a single odd-even check digit at the end of each word, as it is being written into the memory. This check digit, which is chosen so as to make an odd number of the total sum of the one-digits appearing in each word, occupies the P-46 digit position of the word and is not susceptible to manipulation by the arithmetical (or other program) instructions. The consistency of each word stored in the memory is checked by a special detecting device which causes a halt (with appropriate warning signal) whenever a word is observed which fails to pass the odd-number checking test.

This error-detecting device can detect an error under any of the following circumstances:

(1) Whenever a defective word is read out of the memory or is found to occupy a memory location which is about to be written into.

(2) Whenever a defective word is observed in a memory tank during the period prior to the minor cycle in which the particular address sought for becomes available.

(3) Approximately once every second or so, at which time the program is temporarily halted for about 25 milliseconds and the entire memory is scanned from beginning to end in search of a defective word.

In all cases, the halt and error-signal indication occurs in time to prevent the execution of any further instructions. For example, when a memory location containing a defective word is written into, the halt occurs immediately after the writing has taken place, and before a new instruction has been selected.

Manual controls are provided which aid in locating the address which contains the defective word (See Section V, paragraph 2). Whenever a defective word is printed out on the supervisory printer, a characteristic error-indication character replaces the customary sign-indication character at the right-most end of the word.

Although intended primarily for application in connection with the mercury acoustic delay-line memory, the checking equipment is in no way restricted or limited to this type of memory and may be utilized equally well in connection with any other type suited for annexation to this machine.

II. Arithmetic and Processing Operations

1. Addition-type operations

Five addition-type operations are provided, namely: Addition (normal), Subtraction, Accumulate-and-overflow check, Accumulate-and-store, Summation. These operations are described in detail in the following paragraphs.

1.1 Addition, normal (code symbol, 2)

This is the usual algebraic addition operation, like the one used in SEAC. The instruction reads "Form the sum of the word in alpha plus the word in beta, and write the result in address gamma."

1.2 Subtraction (code symbol, 1)

The instruction reads "Form the difference of the word in alpha minus the word in beta, and write the difference in address gamma." No provision is made in either this operation or the preceding one for checking against the occurrence of overflow.

1.3 Accumulate-and-overflow-check (code symbol, 4)

This operation and the following operation are intended for use in situations where the accumulation of the sum of a long list of numbers is desired, or in situations where it is particularly difficult to predict the occurrence of overflow. It combines the characteristics of both an arithmetic operation and a choice operation. Both instructions involve the use of an arithmetic register for the purpose of accumulating a partial sum. This register is referred to as the arithmetic accumulator register, or simply the accumulator. The instruction reads: "Form the sum of the word in alpha plus the word in beta plus the contents of the accumulator, and, if the sum does not overflow, store the results in the accumulator, and proceed to the normal next instruction. If the indicated sum does produce an overflow, however, leave the previous contents of the accumulator unchanged, and take the next instruction from address gamma in the memory." For rules governing the clearing of the accumulator, see paragraph 1.4 below. Note that this operation does not involve the writing of any information into the memory.

Using the Accumulate-and-overflow-check instruction, lists of numbers may be summed at least twice as fast as is possible by means of the normal addition operation, because a single instruction allows two items rather than only one to be added to the accumulated partial sum.

1.4 Accumulate-and-store (code symbol, 5)

This instruction permits writing of the contents of the accumulator register into the memory. It is used for terminating a series of Accumulation-and-overflow-check operations. It does not in itself provide any check against overflow.

The instruction reads "Form the difference of the word in alpha minus the word in beta, add to it the contents of the accumulator, and write the sum into memory address gamma. Then clear the accumulator."

This instruction is also used for clearing the accumulator prior to starting an accumulation sequence. Another use occurs during double-precision operations or when division remainders are desired. These are described in the paragraphs below dealing with Multiplication and Division.

The rules governing the modification of the accumulator contents, and the effect of the accumulator contents on the various operations, are summarized in Table 2. Notice that the accumulator contents affect (or are affected by) only five out of the total of 16 instruction types. This fact contributes considerably to the power and flexibility with which these accumulation features can be used, because it allows the coder to interleave those accumulation instructions that actually add the successive items to the partial sum with other instructions, such as normal additions, comparisons, logical transfers, etc., which can be used for modifying addresses in some of the instruction words, keeping independent tallies, and so forth.

Notice also that this operation involves the difference (alpha minus beta) rather than the sum (alpha plus beta), as in the preceding operation. This feature is intended to facilitate the process of writing the contents of the accumulator, unmodified, into the memory. To accomplish this, the coder need only write the same arbitrary number for both of the addresses alpha and beta in the instruction word. This may be desirable for the final step in the accumulation process, if the coder fears the possibility that the addition of the last two items may cause an overflow.

Table 2. Summary of Rules Governing Use of Accumulator

Instructions Not Affecting (and not affected by) Contents of Accumulator	Instructions Affecting Contents of Accumulator	Modified Contents
Addition	Accumulate-and-overflow-check*	Partial sum
Subtraction	Accumulate-and store*	Zero
Summation	Multiplication, major (rounded)	Major product (rounded)
Shift	Multiplication, minor	Major product (unrounded)
Justify	Division	Remainder
Logical Transfer		
Comparison (algebraic)		
Comparison (absolute)		
File (unconditional)	*Except for these two accumulation operations, the previous contents of the accumulator do not affect the results of the operation.	
Breakpoint File		
Input-Output		

1.5 Summation (code symbol, 3)

This operation permits the extremely rapid summation of large blocks of words located in consecutively-numbered address positions in the memory. The instruction reads "Form the sum of the words located in consecutive address locations running from beta through alpha inclusive, and write the sum into address gamma in the memory." For this operation, address beta is ordinarily numerically less than address alpha. If this condition is not observed, however, the sum of the words in addresses beta and beta-plus-one is produced.

No provision is made in this operation for checking against overflow. In carrying out certain programs of checking procedures, however, which involve forming the sums of large groups of words, the occurrence of overflow is generally not material. It must be remembered that when overflow does occur, the sum obtained may be in one of two ambiguous forms depending upon the order and the grouping in which the words were summed. The equivalence of these two forms may be established, however, by checking the fact that their difference, as computed by a Subtraction operation, is numerically equal to zero.

Summation operations and input-output operations do not proceed concurrently. If a Summation instruction is given while an input-output operation is in progress, the program is temporarily halted until the input-output operation has been completed.

2. Multiplication-division type operations

2.1 Major multiplication, rounded (code symbol, A)

This operation produces a rounded-off major product. The binary point in both of the operands and in the result is located two binary positions from the left-hand end of the word (i.e., between P44 and P43). In other words, the largest magnitude that these numbers can reach is slightly less than four. The instruction reads "Form the product of the word in alpha and the word in beta and write it, rounded off, into address gamma in the memory, and into the accumulator.

Rounding-off is accomplished by adding a unit in the first rejected position on the right-hand end of the product and allowing the resultant carrys, if any, to propagate. Means for securing the major product unrounded are described immediately below.

2.2 Minor multiplication (code symbol, B)

This instruction produces the minor product. The instruction reads "Form the minor product of the word in alpha and the word in beta and write the result in address gamma in the memory." The location of the binary point in this operation is the same for the words in alpha and beta as in the preceding operation. The result, however, is positioned so that the least significant digit in the product is located in the right-most numerical position in the word, namely the P-2 position. An alternative interpretation of the binary point positions would be that the binary point for both of the factors and for the product is located at the extreme right-hand end of the words (that is, all of the words are integers).

Whenever a minor Multiplication instruction is performed, the major product, unrounded (with proper sign), is written into the arithmetic accumulator register. If such an operation is followed by an Accumulate-and-store instruction, the major product may be written into the memory in considerably less than the usual multiplication time. It should be noted that the right-most two binary digits of the major product and the left-most two binary digits of the minor product are identical.

2.3 Division (code symbol, C)

By means of the Division operation, both the quotient and the remainder may be obtained rapidly. The instruction reads "Form the quotient, word in beta divided by word in alpha, and write it into address gamma in the memory." The positions of the binary point in this operation are the same as for the major Multiplication operation described above. The quotient is not rounded off, and the process is carried out in such a way that the quotient may be in error by being numerically too small as a result of the lack of rounding off. Whenever a Division operation is performed, the remainder is written into the arithmetic accumulator register with proper sign, namely, the sign of the dividend. The remainder is never numerically greater than the divisor times two to the minus 42 power. The remainder may be rapidly written into the memory by means of an Accumulate-and-store instruction, as described above.

2.4 Shift (code symbol, 8)

By means of this operation, words may be shifted an arbitrary number of places either to the left or to the right in considerably less than a Multiplication or Division time. The instruction reads "Shift the word in alpha according to the code indicated in the word located in address beta in the memory, and write the result into address gamma in the memory." The six right-most numerical digits, P2 through P7, of the word in memory location beta indicate the number of binary places the word is to be shifted; the P-1, or sign digit, indicates the direction of the shift, -- i.e., for left-shift positive and for right-shift negative. In other words, the digits P-1 through P-7 indicate the power of two by which the word in alpha is to be multiplied without round-off. The sign of the shifted word is transferred unchanged.

The time required for a shift operation is a function of the number of places shifted, and ranges from 1 minor cycle to 12 minor cycles (average 6.5 minor cycles) longer than an addition operation.

3. Processing operations

3.1 Justify (code symbol, 9)

The Justify operation provides the coder with a convenient means for carrying out floating binary point operations, or other operations requiring frequent readjustment of scaling factors. The instruction reads "Determine the number, N, satisfying the following inequalities:

$$(\beta) \leqslant (\alpha)\ 2^{N} < 2\ (\beta)$$

$$-63 \leqslant N \leqslant 63$$

and write this number with proper sign into the digits P-1 through P-7 of the word in address gamma without in any way altering its other digits." (β) means "absolute value of word in memory address β," etc. By choosing (β) equal to a power of 2, this operation can be used to determine the number of places that the word in alpha must be shifted in order to make its most significant binary digit coincide with the most significant binary digit of the word in beta. Note that the result of a Justify operation is written into the memory in such a way as to make it possible to use it immediately as the beta operand in a subsequent Shift operation which will cause the shifted word to be lined up with some designated comparand in the manner described by the first inequality above.

When either or both of the operands in a Justify instruction are equal to zero, the result produced is N = 63, as indicated in hexadecimal notation in the following table:

Alpha	Beta	Gamma
Zero	Not Zero	3F+
Not Zero	Zero	3F-
Zero	Zero	3F-

The time required for a Justify operation is a function of the value of N produced, and averages N + 2.5 minor cycles longer than an addition operation.

3.2 Logical Transfer (code symbol, D)

The logical transfer instruction is similar to its counterpart in SEAC. The instruction reads "Write into address gamma in the memory those digits of the word in alpha which correspond to one-digits in the word in beta. Leave the word in gamma unchanged in those digit positions which correspond to zero-digits in the word in beta." This operation permits segments of one word to be transplanted into another word. Also, it contains as special cases the processes of logical multiplication and logical addition.

III. Program Control Operations

1. Program sequencing

1.1 Sequence of instructions

Under the three-address system employed in this machine, consecutive instructions are normally located in consecutively-numbered address positions in the memory. Means for interrupting this consecutive sequencing of instructions and for initiating a new sequence are provided by a variety of choice instructions and counter-setting instructions which are described in paragraphs 2 and 3 below. In addition to this ability to initiate new instruction sequences at any time, the programmer also has the facility of choosing between two possible alternative sequences of instructions at any time. That is, he may (if he wishes) interleave two distinct sequences, periodically jumping from one to the other in an arbitrary manner. This facility is made possible by the dual counter register scheme described below.

1.2 Dual counter-registers

As in the three-address control system in SEAC, two distinct counter-registers are provided for program sequencing. These counter-registers are designated as counter No. 0 and counter No. 1, respectively. Each counter holds a twelve-binary-digit address. The coder may select the address in either counter as the address of the next instruction to be performed. The counter from which the address of the next instruction is to be taken is indicated by the P-6 or "d" digit in the instruction being currently executed. (See Table 1 for the format of an instruction word.) In other words, this d-digit in every instruction indicates the source of the address of the instruction immediately to follow. Execution of this process may be temporarily suspended, however, by means of a program-jumping feature available in connection with input-output operations, which is described in Section IV, paragraph 3, below.

2. Choice instruction operations

2.1 Comparison, algebraic (code symbol, 6)

By means of this instruction, either of the counter registers may be re-set and a new program sequence initiated, starting at the new address setting. The instruction reads "If the word in alpha is algebraically greater than or equal to the word in beta, take the next instruction from the normal consecutive address position. If, however, the word in alpha is less than the word in beta, take the next instruction from address gamma." At the same time the counter specified by the P-6 (d-digit) is reset to gamma. Note that the indicated inequalities pertain to the algebraic magnitudes of the numbers.

2.2 Comparison, absolute (code symbol, 7)

This instruction is similar to the Comparison (algebraic) described above, differing from it only in that the indicated inequalities pertain to the absolute values of the numbers indicated.

2.3 Overflow check (code symbol, 4)

The Accumulation-and-overflow check operation described in Section II, paragraph 1.3, above, may be used as a choice-type operation as well as an arithmetic-type operation. Paraphrasing the previous description of this operation, it becomes: "If the addition of the word in alpha and the word in beta to the contents of the accumulator causes an overflow, take the next instruction from address gamma in the memory. Otherwise, take the normal next instruction."

3. Relative addresses

3.1 Definitions

The numbers written in the address segments of an instruction word may be interpreted in either of two ways, namely, as absolute addresses or as relative addresses. The intended interpretation is indicated by a set of code digits in each instruction word. (See Table 3.) A code digit 0 indicates an absolute address, and a code digit 1 indicates a relative address.

An absolute address is interpreted merely as a number identifying a specific fixed memory location. A relative address, however, identifies a position in the memory by specifying its displacement relative to a certain address number that is stored in one of the special counter-registers. That is, if the counter-register in question contains the number C, then whenever an address number A is written in an instruction word along with a relative designation, the memory location A plus C is referred to. The value of C may range from 0 to $2^{12}-1$, and the effect of negative displacements can be secured by using the complement, modulo 2^{12}, of the displacement desired.

Either of the two available counter-registers (No. 0 or No. 1) can be used in this fashion. The File instructions, which are described in paragraph 4 below, govern the adoption of either counter for relative address purposes and the insertion of the desired base number, C. After either counter is adopted for this purpose, it remains adopted until a countermanding File instruction is given. The following terminology is used in the subsequent discussion:

Relative counter, or adopted counter refers to the counter-register holding the number to which alpha, beta, or gamma are relative. (Chosen by means of File instructions.)

Instruction counter refers to the counter-register holding the address from which the next instruction is scheduled to be taken. (Chosen by means of d-digit in each instruction word.)

Table 3. Relative-Address Information and Program-Sequencing Information

Binary Digits	Digit Position in Word	Content
a-digit	P-9	indicates whether alpha is an absolute or relative address*.
b-digit	P-8	indicates whether beta is an absolute or relative address.
c-digit	P-7	indicates whether gamma is an absolute or relative address.
d-digit	P-6	indicates which counter register contains the address of the next instruction.
Total: 4 bits		

*Except for the operations Input-Output (see Table 7), File and Breakpoint (see Table 5).

3.2 Applications of relative address

3.21 Use of a fixed base

As a result of the dual counter-register system, two different general methods of applying the relative address facility are available. The first method makes use of a fixed base number, or base point, from which the address in the instruction word is used to indicate the relative displacement of the memory location referred to. In this type of application, the counter adopted for relative (i.e., Adopted Counter) and the counter used for program sequencing (i.e., Instruction Counter) are different. For example, suppose counter No. 0 is chosen as the Instruction Counter and counter No. 1 is the Adopted Counter. Since counter No. 0 is the Instruction Counter (i.e., all the instructions are written with the d-digit equal to zero), the counter reading is generally increased by unity after each instruction is executed. The contents of counter No. 1, however, since it is not being used for program sequencing, can be left unchanged. Thus all operands in the instruction being executed can remain relative to a fixed point.

From time to time, the contents of the adopted counter (No. 1) may be modified as the occasion arises. (Means for modifying counter-readings with File instructions are described in paragraph 4 below.) After each such modification, which requires only a single instruction word, all of the operands referred to in the set of instructions being executed are automatically selected from a new block of memory locations. This type of procedure is particularly valuable when large numbers of identical operations need to be performed on successive words, or groups of words, systematically arranged in the memory. Examples of this sort arise in matrix operations, a typical example being the multiplication of two matrices.

3.22 Use of a floating base

The second type of application of the relative address feature involves the use of a floating base point, rather than a fixed one. This floating base point is the address of the instruction word itself. Under this scheme, the adopted relative counter and the instruction counter are the same. With this procedure, the memory locations of the operands referred to may be indicated relative to the instruction word even when the address in which the instruction word is located is not known to the coder at the time the program is being written. This feature is particularly valuable in permitting the preparation of standardized sub-routines in an invariant form that is independent of the actual memory locations into which the sub-routines will ultimately be inserted.

3.3 Method of adopting counter-registers for relative addresses

A counter-register is adopted as a relative counter by means of one of the counter-setting and filing operations, namely, File (unconditional), or Breakpoint File which are described below.

4. Counter-setting and filing operations

4.1 File, unconditional (code symbol, F)

These counter-setting and filing operations permit the programmer to (1) adopt either given counter as the relative counter, (2) reset this counter to a new reading, and (3) record the old setting of the counter in the memory along with various other useful information. The File (F) instruction reads: "Write the file record (indicated in Table 4 below) into the memory location specified by address beta in this File instruction-word. (Format of the File instruction-word is indicated in Table 5). Then reset the counter specified by the P-9 digit of the File instruction word to the setting indicated in the gamma segment of the File instruction word, and adopt the reset counter for future relative references."

After a counter-register is set by means of a File instruction, its contents remain fixed until either (1) reset by another File instruction or (2) referred to as the source of the address of a next-instruction. As a general rule, whenever a given counter-register is referred to as the source of a next-instruction address, the counter reading does not receive its unit advance until after the specified instruction has been executed and its result has been written in the memory. Exception is made to this rule only in the case of the P-45 through P-34 digits of the file-record, in order to record the address of the scheduled next-instruction. If a counter has not been used as the source of a next-instruction address since it was last adopted, this fact is indicated in the P-4 digit of the file-record.

Table 4. File-Record: the information written into the memory as a result of a File instruction (F or E)

Information	Contained in Digit Position
Address of scheduled next instruction.	P-45 through P-34
New counter reading, after resetting.	P-33 through P-22
Prior counter-reading, plus the number in the alpha segment of the File instruction word.	P-21 through P-10
File instruction word's a-digit.	P-9
File instruction word's b-digit.	P-8
File instruction word's c-digit.	P-7
File instruction word's d-digit.	P-6
Which counter was previously adopted. (No. 0 or No. 1)	P-5
About counter No. 0: (if this counter has not been used as a next-instruction address source since it was last adopted or reset, the P-4 digit is 1; if it has been so used, the P-4 digit is 0).	P-4
About counter No. 1: (if this counter has not been used as a next-instruction address source since it was last adopted or reset, the P-3 digit is 1; if it has been so used, the P-3 digit is 0).	P-3
About the previous instruction: (whether the d-digit of the previous instruction was a 0 or a 1).	P-2
About the counter not being adopted: (if this counter is at present being reserved for an Input-Output program jump*, the P-1 digit is 1; if not, the P-1 digit is 0).	P-1

*Note: For a discussion of Input-Output program jumps, see Section IV, paragraph 3.

The File instruction-word, which is not to be confused with the resulting file-record, has the same general format as the other types of instruction-words (cf. Tables 1 and 5) but the significance of some of the digit positions is different.

Table 5. Word-Format of File Instructions

Digits	Word Segment	Contents
P-45 through P-34	Alpha	Contains the displacement constant to be added to prior counter reading in order to produce P-10 through P-21 of file-record.
P-33 through P-22	Beta	Contains the memory address, absolute or relative, into which the file-record is to be written.
P-21 through P-10	Gamma	Contains the new setting for counter, either absolute or relative.
P-9	a-digit	Indicates which counter to file and reset (No. 0 or No. 1).
P-8	b-digit	Indicates whether beta is absolute (0) or relative (1).
P-7	c-digit	Indicates whether gamma is absolute (0) or relative (1).
P-6	d-digit	Indicates which counter is the source of the next-instruction (No. 0 or No. 1).
P-5 through P-2	Operation symbol	Contains the File-operation code symbol.
P-1	Monitor signal	Either 0 or 1.

Some examples of several typical uses of the File instruction and the resultant file-record written into the memory are contained in Table 6.

Table 6. Typical Uses of the File Instruction

	Purpose of Instruction	P-Digits	Content of File Instruction	Content of File-Record
Example 1	File both counter readings in address X; the No. 0 counter to be retained as the instruction counter; the No. 1 counter to be retained as the adopted counter.	45--34	0	Counter No. 0, advanced
		33--22	X	Counter No. 1
		21--10	0	Counter No. 1
		9	1	1
		8	0	0
		7	1	1
		6	0	0
Example 2	File both counter readings in address X; the No. 1 counter to be retained as the instruction counter; the No. 0 counter to be retained as the adopted counter.	45--34	0	Counter No. 1, advanced
		33--22	X	Counter No. 0
		21--10	0	Counter No. 0
		9	0	0
		8	0	0
		7	1	1
		6	1	1
Example 3	File in memory address X the address for the present instruction (counter No. 0) reduced by Y (to be interpreted later as the location of the instruction initiating another iteration of program); the No. 0 counter to be retained unchanged as the adopted counter; the next-instruction address to be taken from counter No. 1	45--34	2^{12} - Y	Counter No. 1
		33--22	X	Counter No. 0
		21--10	0	Counter No. 0 minus Y
		9	0	0
		8	0	0
		7	1	1
		6	1	1
Example 4	Reset counter No. 1 to equal X; and take the next instruction from X.	45--34	0	X
		33--22	Unused address	X
		21--10	X	Previous setting of No. 1
		9	1	1
		8	0	0
		7	0	0
		6	1	1
Example 5	Advance counter No. 1 by X units and adopt it for relative references purposes; take next-instruction address from counter No. 0	45--34	0	Counter No. 0
		33--22	Unused address	Counter No. 1, advanced X.
		21--10	X	Counter No. 1, before advance
		9	1	1
		8	0	0
		7	1	1
		6	0	0

4.2 Breakpoint File (code symbol, E)

The Breakpoint File instruction provides an operation which is exactly similar to the unconditional File instruction (F) except that it requires the setting of an external switch to indicate that its activation is desired. If the external switch is not so set, Breakpoint File instructions are passed over in the program without causing the prescribed alteration in counter readings, etc., or writing into the memory. The next instruction is selected according to the usual rules.

The Breakpoint File instruction is intended to facilitate the carrying out of program monitoring operations such as are required in the initial debugging of a new program. For example, the programmer can insert Breakpoint File instructions liberally throughout the program, arranging them in such a way as to cause program jumps leading to interpolated automonitoring routines. The automonitoring (or program analysis) routines can be composed in the widest conceivable variety. The initial trial running of a program may then be carried out with the Breakpoint operations activated whenever the need for automonitoring appears to arise. For the normal running of a program, the Breakpoint File instructions can be rapidly passed over without causing any significant loss of time. From a slightly different point of view, the Breakpoint File may be used to cause a given program to be carried out in either of two previously specified ways depending upon the setting of the external selection switch. This feature may be used in applications analogous to those described in Section V, paragraph 3, below.

The unconditional File and Breakpoint File operations do not refer to addresses alpha or gamma in the memory, thereby avoiding the access time which would be otherwise required for these operations.

IV. Input-Output Transfer Operations

1. Instruction types

1.1 Load and Print instructions (code symbol, 0)

In using this system, the programmer can arrange to load or print out any consecutive series of memory locations ranging in size from a single word to the entire memory, even while a program of computation is proceeding. Automatic interlocks are provided in the event that the programmer inadvertently attempts to use any memory location in an inconsistent manner in this respect. For example, if the programmer directs that a given section of the memory be loaded from an external input unit and then some time afterwards directs that the contents of a memory location in this section be used as the source of an instruction, the program will automatically be temporarily halted until the sought-for memory location has actually received the word intended for it from the input unit. The details of these interlocks are described in paragraph 2 below.

The types of input-output operations have been chosen so as to be readily applicable for use with input-output units of widely varying types and characteristics. Among the types contemplated for this machine are (1) a collection of magnetic wire or tape drives, (2) one or more magnetic drum storage units, (3) mechanical keyboard and printing devices such as Flexowriter, and (4) special analog-type input or display devices.

1.2 Stages of the operations

A typical input-output operation proceeds in the following stages: First the computer selects the particular unit with which communication is to be made. The code-number signifying this unit is specified in the instruction word. The term "unit" as used in this context may mean a particular wire or tape drive, or a designated channel on a magnetic drum, etc. In the second stage of the process, the computer runs through a specified number of words or blocks of words on the selected unit, the starting point being either the word lying nearest the reading heads, in the case of a magnetic wire or tape unit, or, in the case of a magnetic drum, a fixed origin point on the periphery. For other types of applications, this step may be used to introduce a variable delay. In the third stage, which begins after the requisite number of words has been counted out, the transfer of words between the external unit and the internal high-speed memory takes place. The transfers commence with the numerically smallest address in the block indicated and proceed in consecutive order until the numerically highest address is reached, thereby concluding the operation. The format of an input-output instruction word is indicated in Table 7. The alpha segment of the instruction word is used to specify (1) the type of operation, i.e., load or print, (2) the type of unit, such as tape unit or drum unit, (3) the direction of motion of the unit, if relevant, (4) the running distance or location of the initial word sought for on the external unit, and (5) the code identifying the particular unit or channel desired. The gamma address segment in the instruction word specifies the initial address in the high-speed memory, and the beta address segment indicates the final address in the high-speed memory between which the transfers are to be made. For transfers to take place, the designated final address beta must be numerically greater than or equal to the designated initial address gamma. If this condition is not satisfied, no transfer of information takes place, and the instruction becomes merely an order to move the designated unit a specified distance in a given direction, and could be used for re-reeling tape drives, and so forth.

Table 7. Word-Format of a Typical Input-Output Instruction (For Tape or Wire Unit, and Drum Unit)

Digits	Word Segment	Contents
P-45	Alpha	Indicates type of operation: Load (0), or Print (1).
P-44	"	Indicates type of unit: Tape (0), or Drum (1).
P-43 through P-34	"	Indicates location of information on unit. On drum: The quadrant of drum is indicated by P-43 through P-42, and the drum channel is indicated by P-41 through P-34. On tape: The direction of motion of the tape unit, forward (0) or reverse (1), is indicated by P-43. The running distance in this direction is indicated by P-42 through P-38. The number of the tape unit is indicated by P-37 through P-34.
P-33 through P-22	Beta	Indicates the final address in the high-speed memory to (or from) which information is to be transferred.
P-21 through P-10	Gamma	Indicates the initial address in the high-speed memory to (or from) which information is to be transferred.
P-9	a-digit	Indicates program-jump: (If a program jump is desired after completion of the input-output operation, P-9 is a 1; if no such jump is desired, P-9 is a 0.)
P-8 through P-7	b, c-digit	Indicates whether beta, and gamma are relative or absolute.
P-6	d-digit	Indicates counter containing address of next instruction.
P-5 through P-2	Operation symbol	Contains code-symbol for input-output operation. (0)
P-1	Monitor Signal	See Section V, paragraph 2.

2. Concurrent regulation

2.1 Address conflict interlocks

As mentioned previously, automatic interlocks are provided as safeguards against inadvertent attempts to use a given memory location for two inconsistent purposes. For example, after initiating a Load operation the machine is not allowed to read from or write into any memory location affected by the Load instruction until the pending memory-loading operation has taken place in that particular location. Similarly, after a Print instruction has been given, the machine is not permitted to write into any location affected by the print-out order until the word in that location has been printed out. Whenever necessary, these automatic interlocks temporarily halt the computing program, but only for the minimum time needed to insure that these conditions are fulfilled. For example, if an instruction specifies that memory addresses 000 through 500 be loaded and that the next instruction be taken from address 010, the program will proceed to the next instruction immediately after address 010 has been loaded, even though the remaining 490 words have not yet been entered into the other memory locations. The interlock will be called into play again only in the event that premature reference to any other of the first 501 words be attempted.

2.2 Priority conflict interlocks

Automatic interlocks are also provided against the contingency that the program calls for an additional input-output operation before a previously-ordered input-output operation has been completed. In this event, the program halts temporarily, waits until the preceding operation is completed, and then proceeds in the usual fashion to carry out the second instruction.

3. Program jumping feature

For many applications which make use of concurrent loading or printing-out of the high-speed memory while computations are proceeding, the precise length of time required to complete the input-output operation is either indeterminate or else quite difficult to predict. A typical example of an instance of this type occurs when it is necessary to hunt for a particular word or group of words located in an unknown location on a long magnetic tape reel--where a word can be identified as one of the type sought-for only after it has been read into the computer and subjected to an analysis of its characteristics. Since efficiency requires that information be read into the high-speed memory in fairly large blocks, it is desirable (while the comparatively slow procedure of reading-in the information is taking place) to make use of the ability of the system to proceed on other independent phases of the program. By means of the special input-output program-jumping feature, the programmer is able to direct that these other concurrent independent phases of the program be allowed to proceed uninterrupted until the specified loading

operation shall have been completed, and that, as soon as the loading is completed, the program shall jump to an alternative specified routine which is designed to carry out the identification analysis upon the newly-received information and to decide whether or not to retain it or to continue hunting on the tape.

An analogous need arises when it is desired to check the accuracy of information printed out on the external medium by reading the printed-out information back into the computer and comparing it with its original source inside. A process of this sort would require a Print instruction followed by a Load instruction. In order to avoid having to force the program to stand by while the print-out operation is being completed, the programmer could arrange for other internal operations to be carried out concurrently which would be interrupted only when the print-out operation was concluded, at which time the input instruction would be initiated.

Program jumps of this sort are directed by means of the P-6 and P-9 digits in the instruction word. More explicitly, the P-6 digit (or d-digit) of each instruction word specifies the counter register (No. 0 or No. 1) which contains the address of the instruction to be executed immediately after completion of the current instruction. If the d-digit is 0, the address of the next instruction is contained in counter No. 0, and if the d-digit is 1, the source of the next instruction is contained in counter No. 1. When the current instruction is an input-output order (which generally specifies a lengthy external operation) and the programmer desires, prior to its completion, to proceed with the next instruction and other internal operations which can be carried out while the lengthy external operation is being executed, then the P-9 digit (or a-digit) of the current instruction word is written equal to 1. In effect the programmer thus, at the time that he orders the input-output unit to perform a specified task, includes in the order a request for the production of a signal when the task is completed. This signal, received from the input-output control, causes the program to jump; i.e., the counter register then being used as the source of the next instruction is temporarily abandoned, and the instruction word in the address location stored in the <u>other</u> counter register is executed in its stead.

Table 8. Summary of Program-Jump Rules

Input-Output Instruction	Source of Immediate Next Instruction	Source of Next Instruction After Input-Output Operation is Completed
d-digit equals 0 and a-digit equals 1	Counter No. 0	Counter No. 1
d-digit equals 1 and a-digit equals 1	Counter No. 1	Counter No. 0

(If the a-digit equals 0, no program jump occurs and, after completion of the input-output operation, the next instruction is selected in the usual way by the instruction word being executed at that time.)

After effecting a program jump, the new program that is initiated by the substitute instruction may, if the programmer so desires, be written with a constant d-digit in all of its instructions except the final one. As a result, as soon as the final instruction is executed, the program will return to the instruction located in the memory address stored in the temporarily-abandoned counter-register and resume the interrupted program where it left off. It should be noted that the programmer need make no provision for explicitly recording or determining the address at which the interrupted program is to be resumed. This address will automatically be held unchanged in the temporarily-abandoned counter register.

After the program has encountered an input-output instruction with the a-digit equal to 1 (which, in effect, indicates that the programmer wishes to preserve the address in the control counter-register for future reference), any premature attempt to alter the contents of this register by means of a counter re-setting instruction (or any instruction whose d-digit specifies the reserved counter) will result in the program being halted temporarily until the input-output operation in question has been completed and the originally directed program-jump is ready to be made. This automatic interlock relieves the programmer of the burden of having to estimate precisely how long a time will be required for the input-output operation to be completed and insures that all the desired steps in the program will be carried out in their proper sequence. The system is so designed that the order which appears first in the program must be satisfied before any subsequent conflicting order can be carried out.

V. Manual-Monitor Operations

1. Definitions

Under the general heading of manual-monitor operations are included all those operations which are either initiated manually by the machine operator, who for example presses a push-button, or else are initiated by the program itself under conditions which are specified by means of external switch settings. The first type is called a manual operation; the second type a monitor operation. The exact nature of the operations performed in both cases is specified by means of external switch settings.

2. Types of operations performed

The types of manual-monitor operations which may be performed fall into five main classes, namely (1) loading operations, which load new information into specified portions of the machine such as various memory locations and storage registers, (2) print-out operations, which print-out the contents of these storage locations inside the machine, (3) substitute-new-address directions that

interpolate new instructions between the members of the programmed instruction sequence, (4) substitute-new-address directions that initiate entirely new sequences of programmed instructions, and (5) halt-program orders, with warning signal.

A list of the various storage locations inside the machine to which some or all of the above-mentioned types of manual-monitor operations may be made to pertain is indicated in Table 9, below. Typical examples of possible manual-monitor operations which the operator may select are given in paragraph 3 below.

Table 9. Storage locations directly accessible by means of Manual-monitor operations

Memory locations:

1. Address 000.
2. Addresses 000 through 007, inclusive.
3. Addresses 000 through 00F, inclusive.
4. Addresses 000 through 017, inclusive.
5. Addresses 000 through 01F, inclusive.
6. Addresses 000 through end*.
7. Address alpha indicated in the current instruction word.
8. Address beta indicated in the current instruction word.
9. Address gamma indicated in the current instruction word.
10. Address scheduled for next reference by the current instruction word.
11. Address stored in Counter-register No. 0.
12. Address stored in Counter-register No. 1.
13. Address stored in Address Storage Register (ASR).
14. thru 20. Similar to items 7 through 13 above, except that all eight addresses in the tank in which the indicated word is located are included.
21. Addresses 000 through ASR**.
22. Addresses ASR through end, inclusive.
23. Addresses ASR through 007, inclusive.
24. Addresses ASR through 00F, inclusive.
25. Addresses ASR through 017, inclusive.
26. Addresses ASR through 01F, inclusive.

Storage Registers:

27. Instruction Register.
28. Arithmetic Accumulator Register.
29. Counter-Register No. 0.
30. Counter-Register No. 1.
31. Address Storage Register (ASR).

*"end" denotes largest available address number in memory.

**ASR denotes the address number stored in the Address Storage Register.

Three general methods are available to the operator for indicating the type of manual-monitor operation desired. First: for the most frequently used type, individual push-buttons are provided whereby the specified operation can be initiated manually at any time, with a minimum of effort. Second: for the less-frequently used types, a group of rotary selection switches are provided for indicating the combination of operations and addresses desired. Finally, in a somewhat less accessible manner, means of selecting any possible combination of specifications are provided by a batteryof small switches which can be set up in various specified patterns so as to produce those operations which are less-frequently useful to the operator or which might be needed only for certain special engineering test purposes.

3. Modes of initiating monitor operations

By means of a set of external switches, the operator may indicate the occasions or conditions under which the specified type of manual-monitor operation is to be initiated by the course of the internal program. For example, the operator may indicate that the monitor process is to be initiated by any one or more of the following conditions:

1) Every new instruction;
2) Every new reference to the memory;
3) An instruction to which a negative sign (P-1 equal to one) is attached;
4) A comparison instruction which results in the selection of the normal consecutive next instruction;
5) A comparison instruction which results in the selection of the jump alternative next instruction stored in address gamma;
6) Any address reference which exactly matches the twelve binary digit address number stored in the special Address Storage Register (ASR) provided for this purpose;
7) The same as the above except that only the nine most significant binary digits are matched;
8) An address reference lying within some indicated one (or more) of the eight possible memory cabinets;
9) A memory-error indication (See Section I, paragraph 3);
10) Various other special conditions.

It will be noted that item 8 above permits the three left-most binary digits of each address to be used as special code keys that signify the various classes of occasions on which a monitor operation might be desired. This feature is still applicable even if the full 4096-word complement of memory locations is not actually provided. For example, if only a single 512-word cabinet of memory is in operation, these code digits are capable of expressing seven classes of automonitor-initiating conditions.

4. Applications of manual-monitor system

Under Section III, paragraph 4.2, describing the Breakpoint File operation, it was noted that the initiation of automonitoring routines could be controlled by means of the external switch which is provided for activating Breakpoint File instructions,--provided such instructions had been inserted in the program in the proper strategic places. In the event that these Breakpoint File instructions have not been inserted in the program, the third type of monitor operation listed above, (substitute-next-instruction-interpolating-new-instruction) can be used for the same purpose. For example, one of the possible operations of this type can interpolate the word in address 000 as the next instruction, in place of the scheduled next instruction in the program, without in any way altering any counter re-setting or other operations which the preceding instruction had caused to be carried out. The word in address 000 would be chosen so as to cause the contents of the counter-registers to be filed in designated memory locations and would then initiate a prepared program of automonitor analysis. (This program might, for example, include printing-out the address of the instruction previously executed, the instruction word itself, various of the operands, or the result of the instruction, etc., etc.) The initiation of this program could be made dependent upon conditions involving the type of instruction, the characteristics of the addresses involved, or the operands themselves. When the desired information has been printed out, the program would cause the control to be re-set to the condition that had existed before the automonitor routine was started and would cause the next regularly scheduled instruction to be selected.

Among other simpler operations which can easily be performed, the following may be noted:

1) Various fixed addresses or groups of fixed addresses may be continually printed out even while the computation is proceeding. Since these operations are initiated by means of external switches which control only d-c voltages, operations of this type might well be intiated by mechanical devices remote from the computer itself. In other words, these remote external devices can direct that various sections of the computer's memory be made available to them. Similarly, they can direct that various sections of the computer's memory be loaded. Thus the computer can be said to share its high-speed memory (and consequently any other part of its internal computing or external storage facilities) with these remote devices,--at the option of either the external devices or the computation program or both, jointly.

2) The contents of a specified address can be printed out whenever and as soon as it may be referred to in the course of the program, with the option of halting the program at that time if desired.

3) Every word read-from or written-into the memory at each step of the program can be printed out. Neither this operation nor the preceding one require the preparation of any special routine.

4) Certain operations are available which pertain to more than one special address. Of these, the following may be taken as typical:

   a) Load the entire memory, and take the word in address 000 as the next instruction.

   b) Load the first tank (or first 2, 3, or 4 tanks) in the memory and take the word in address ASR as the next instruction.

   c) Let the program proceed until the word in address ASR is referred to, at which time print out the contents of the Instruction Register, and halt.

   d) Whenever address ASR is requested, substitute address 000 in its stead, and proceed with the program.

VI. Summary

As the foregoing outline of the DYSEAC system has shown, this machine is capable of carrying out (1) a balanced set of internal arithmetic and program-control operations, (2) a group of external input-output operations which can proceed simultaneously with the internal computing operations and, at the option of the programmer, can be automatically interlocked with them in such a way to to intermesh the sequencing of internal and external operations, and (3) an extended repertoire of manual and monitoring operations by means of which a wide variety of special input-output or program-branching operations may be interpolated at any time into the previously-prepared program instructions. These latter two features, acting in concert, make the internal storage facilities of the machine available to the outside world without appreciably interfering with the internal computations; also they permit unscheduled interchanges of data and program-information to take place between the machine and the outside world at the instigation of either, independently, or of both cooperatively. This communication betweeen the machine and the outside can be readily maintained whether the outside communicant is another automatic device or a human operator.

The human operator might be coupled to the machine system (1) through the medium of a visual display device which exhibits, graphically, numerical data stored inside the machine's memory, and (2) by means of a battery of manual control switches capable of inserting new numerical data or certain new control commands into the machine. For example, in an analytical-study application concerned with the control of air traffic at an airport terminal, the numerical data displayed by the machine would represent in real time the predicted traffic patterns in the neighborhood of the airport, e.g., the predicted locations (based on latest scheduling information) of all aircraft inside a certain area. The operator could, by means of the manual input controls, (1) enter newer data into the machine as the information became available, (2) interrogate the machine for more detailed information concerning certain individual flights as the need arose, and even (3) instruct the machine to exhibit the effect to be expected from issuing revised flight control orders. In applications of this sort which require a constant interchange of intelligence between the machine and its operator (also perhaps between the machine and other devices), the DYSEAC system could be fitted smoothly into the more comprehensive aircraft-communication system, and could serve as a critical link in it.

Because of its combination of flexible external facilities and powerful general-purpose internal facilities, the DYSEAC could also be used as a simulator. It could simulate one (or more) missing elements, e.g., special storage reservoirs, decoders, converters, etc., in a tentatively designed but only partially realized communication network. By acting as an interim substitute for such special-purpose missing links in the overall communication-system, the DYSEAC could test the operational effectiveness of proposed design variations for each link. The effect of variations in the operating properties of each link could easily be explored over a wide range merely by writing into the DYSEAC program of instructions a new set of numerical values for the parameters describing the link's operational specifications. Hence a series of comparative tests could be run off in rapid succession, and the relative efficiency of the proposed designs could be observed under realistic conditions. This procedure would be particularly effective for evaluating those links which are intended to be realized, physically, by inflexible highly-specialized equipment. It would also be particularly effective in evaluating the human operator's reactions to new designs. This latter factor is, of course, extremely difficult to assess by means of theoretical studies alone. Thus the DYSEAC could serve as a valuable analytical tool for carrying out a comprehensive program of system-studies.

Alan L. Leiner

A. L. Leiner
Chief, Machine Systems Branch
Electronic Computers Laboratory

THE NATIONAL BUREAU OF STANDARDS

Functions and Activities

The functions of the National Bureau of Standards are set forth in the Act of Congress, March 3, 1901, as amended by Congress in Public Law 619, 1950. These include the development and maintenance of the national standards of measurement and the provision of means and methods for making measurements consistent with these standards; the determination of physical constants and properties of materials; the development of methods and instruments for testing materials, devices, and structures; advisory services to Government Agencies on scientific and technical problems; invention and development of devices to serve special needs of the Government; and the development of standard practices, codes, and specifications. The work includes basic and applied research, development, engineering, instrumentation, testing, evaluation, calibration services, and various consultation and information services. A major portion of the Bureau's work is performed for other Government Agencies, particularly the Department of Defense and the Atomic Energy Commission. The scope of activities is suggested by the listing of divisions and sections on the inside of the front cover.

Reports and Publications

The results of the Bureau's work take the form of either actual equipment and devices or published papers and reports. Reports are issued to the sponsoring agency of a particular project or program. Published papers appear either in the Bureau's own series of publications or in the journals of professional and scientific societies. The Bureau itself publishes three monthly periodicals, available from the Government Printing Office: The Journal of Research, which presents complete papers reporting technical investigations; the Technical News Bulletin, which presents summary and preliminary reports on work in progress; and Basic Radio Propagation Predictions, which provides data for determining the best frequencies to use for radio communications throughout the world. There are also five series of nonperiodical publications: The Applied Mathematics Series, Circulars, Handbooks, Building Materials and Structures Reports, and Miscellaneous Publications.

Information on the Bureau's publications can be found in NBS Circular 460, Publications of the National Bureau of Standards ($1.00). Information on calibration services and fees can be found in NBS Circular 483, Testing by the National Bureau of Standards (25 cents). Both are available from the Government Printing Office. Inquiries regarding the Bureau's reports and publications should be addressed to the Office of Scientific Publications, National Bureau of Standards, Washington 25, D. C.


NBS