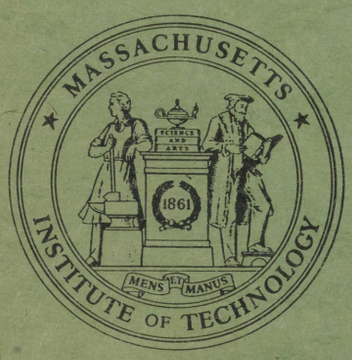
Report R-193

SELECTED DESCRIPTIVE MATERIAL

WHIRLWIND I COMPUTER

Volume 1 of 2: General

November 10, 1950



ELECTRONIC COMPUTER DIVISION
SERVOMECHANISMS LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Report R-193

SELECTED DESCRIPTIVE MATERIAL

WHIRLWIND I COMPUTER

Volume 1 of 2: General

November 10, 1950

Edited by
C. W. Adams

ELECTRONIC COMPUTER DIVISION
SERVOMECHANISMS LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

TABLE OF CONTENTS

Volume I

Volume II

FOREWORD (AND ABSTRACT)

This cover contains a selection of diagrams and memorandums from the reports that have been written about the Whirlwind I electronic computer. This material has been assembled for general reference use by people interested in applications of the computer.

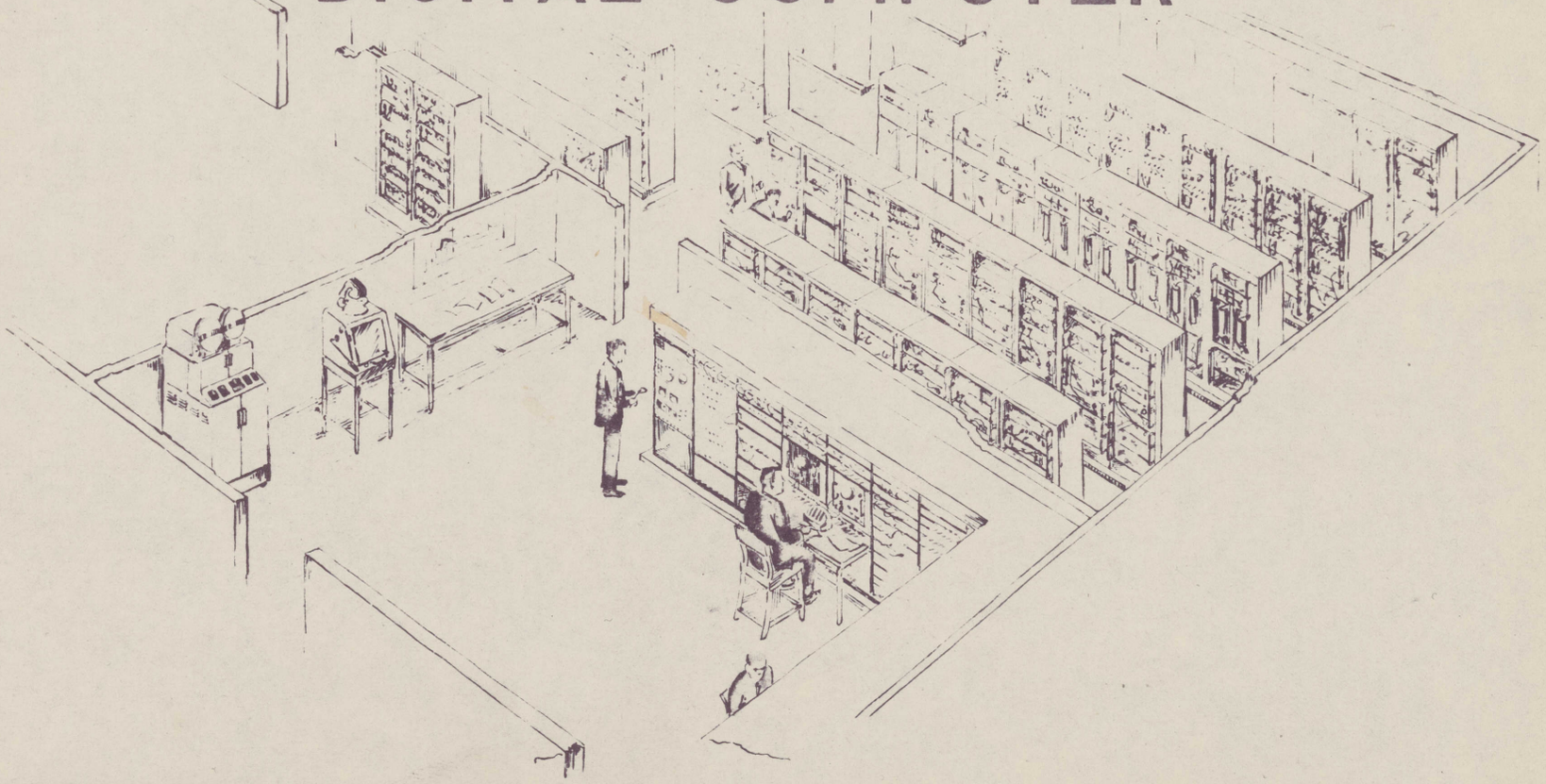
Volume 1 contains reproductions of a series of posters that present a simplified elementary picture of the computer, its operation, and some of its applications; two memorandums that list Project reports on mathematics and on applications; and three memorandums on the insertion of information into the computer, the use of subroutines, and a description of one display program.

Volume 2 contains a short guide to coding, a longer memorandum on coding, and a supplement to the latter.

WHIRLWIND I

is a high-speed electronic

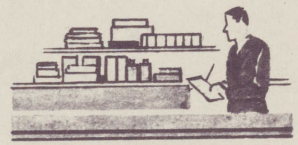
DIGITAL COMPUTER



A-3623
F 1031
SC265

INFORMATION-PROCESSING SYSTEMS using DIGITAL COMPUTERS will have many APPLICATIONS

gun control

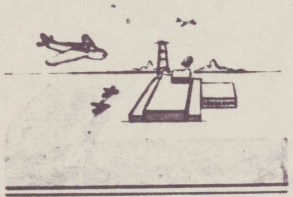


inventory

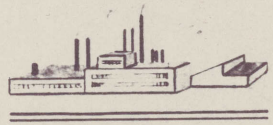
$$\frac{d^2\theta}{dt^2} + a\frac{d\theta}{dt} + b\theta = f(t)$$

mathematics

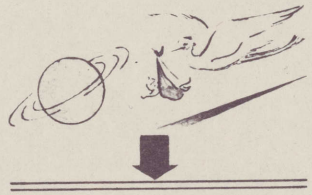
economic analysis



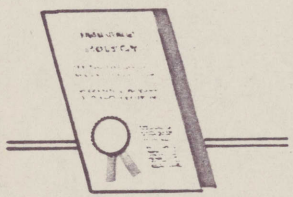
air traffic control



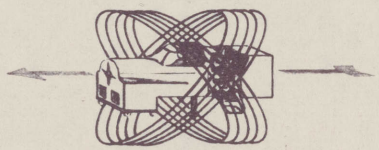
industrial process control



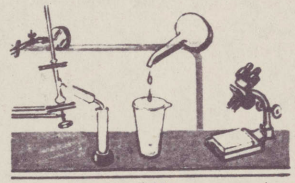
census



insurance handling



simulation

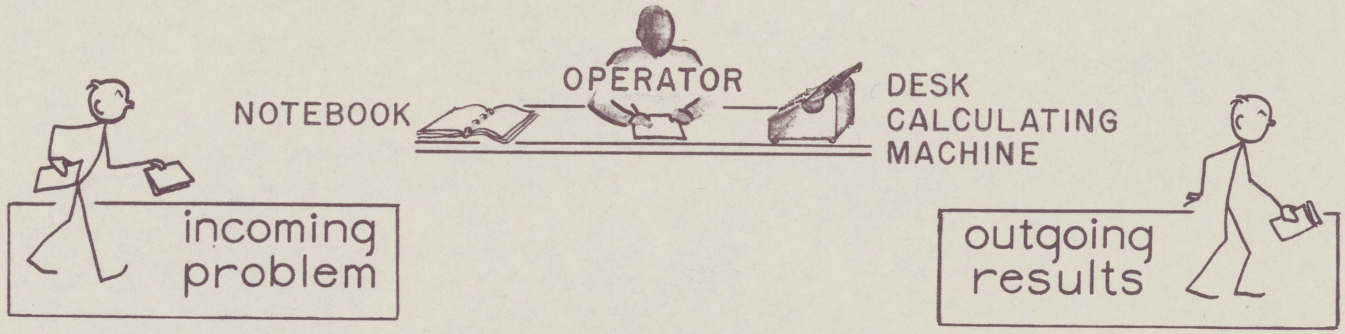


scientific and engineering computations

A-35676
F-1028
SC-207

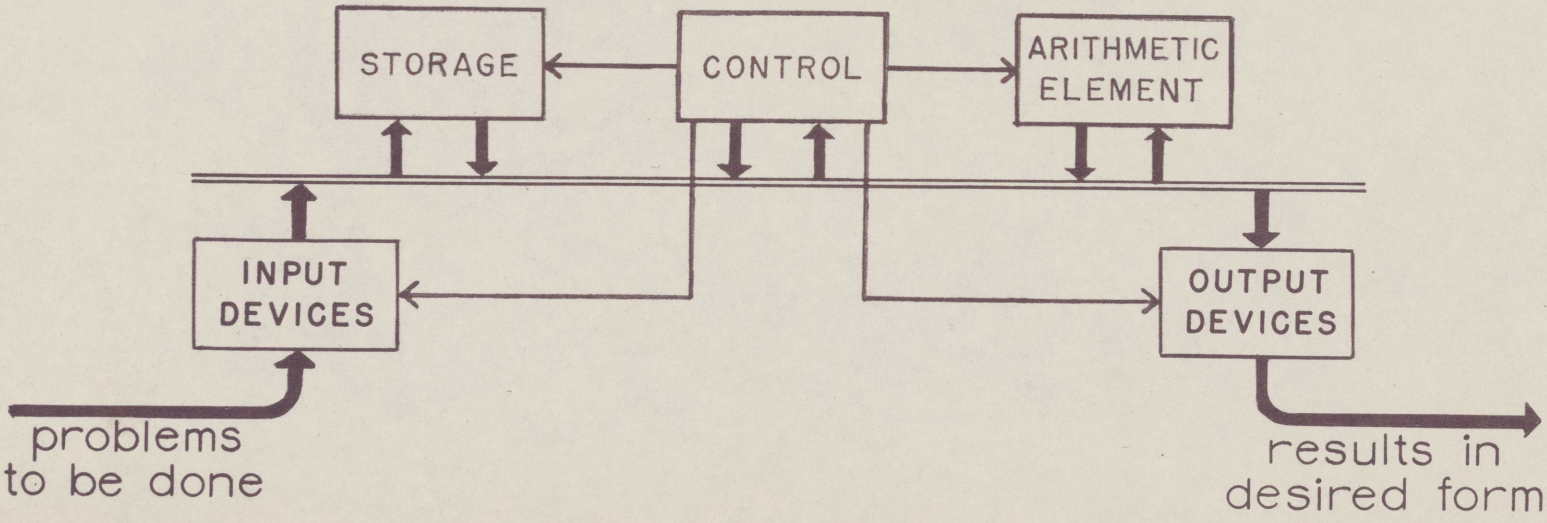
BASIC COMPUTER ELEMENTS

comparison between manual computation



and

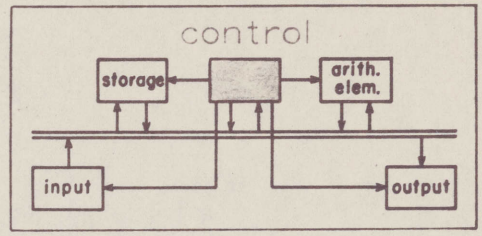
WHIRLWIND I computation



basic computer elements

CONTROL

Just as in manual computation the operator controls the steps to be performed • so in WHIRLWIND I computation



CONTROL

takes next instruction in sequence from storage • examines it and sends pulses at the proper times to the various parts of the computer to perform the necessary *PROCESSES*

instructs STORAGE to

SELECT A STORAGE REGISTER AND READ IN OR READ OUT EITHER NUMBERS OR INSTRUCTIONS

instructs INPUT to

SELECT EXTERNAL DEVICE START — READ — STOP

instructs OUTPUT to

SELECT EXTERNAL DEVICE START — RECORD — STOP

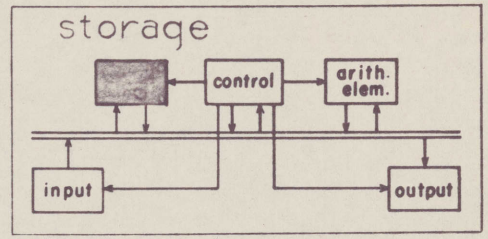
instructs **ARITHMETIC ELEMENT** to

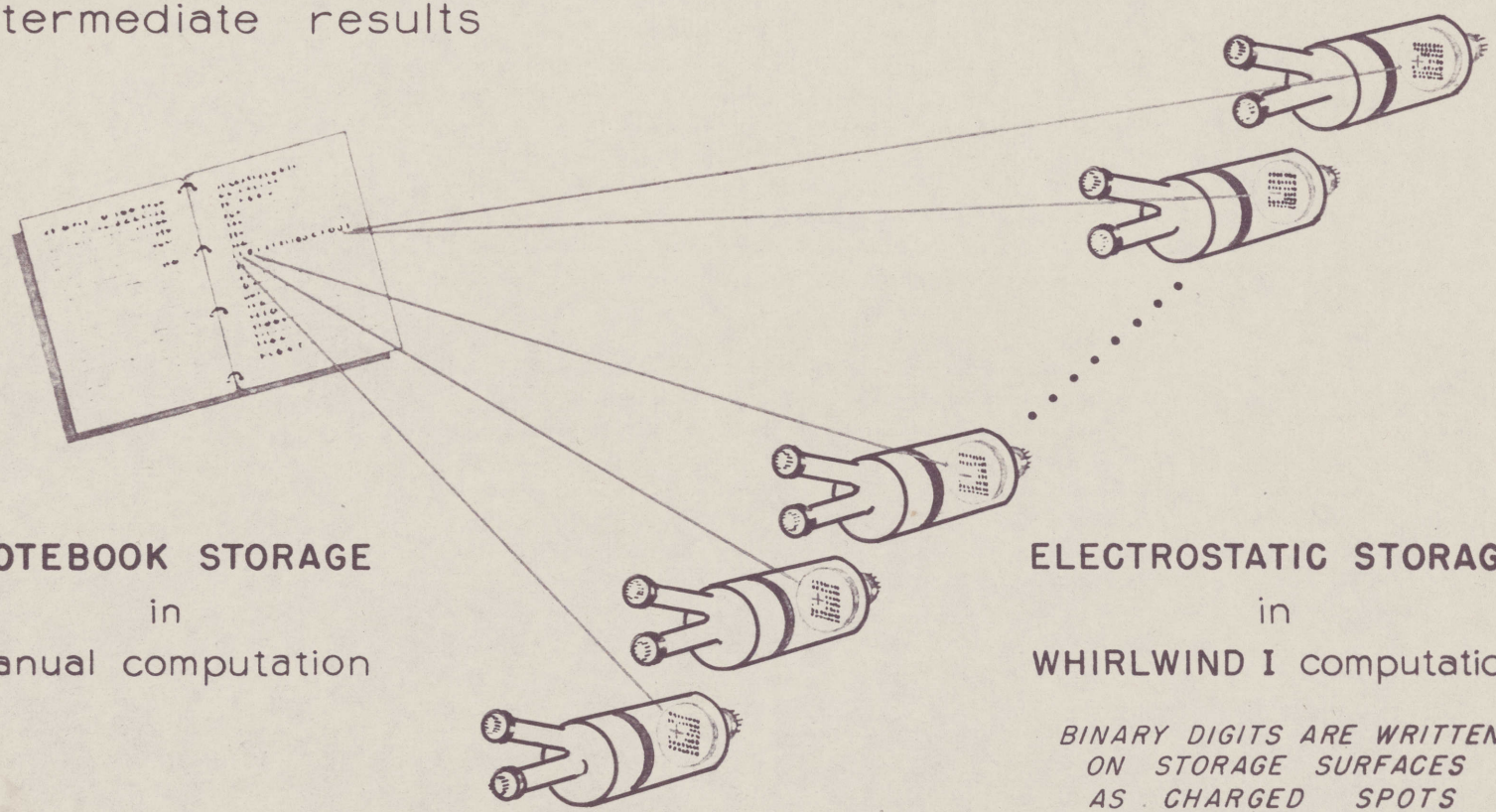
CLEAR REGISTERS · ADD · SUBTRACT · MULTIPLY DIVIDE · COMPARE

basic computer elements

STORAGE

the memory for initial data, instructions, and intermediate results





NOTEBOOK STORAGE
in
manual computation

ELECTROSTATIC STORAGE
in
WHIRLWIND I computation

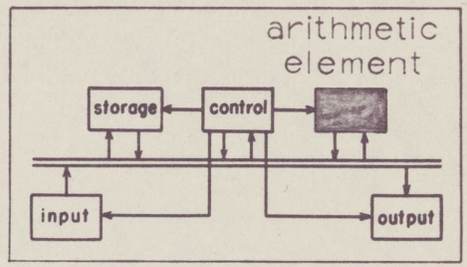
BINARY DIGITS ARE WRITTEN ON STORAGE SURFACES AS CHARGED SPOTS

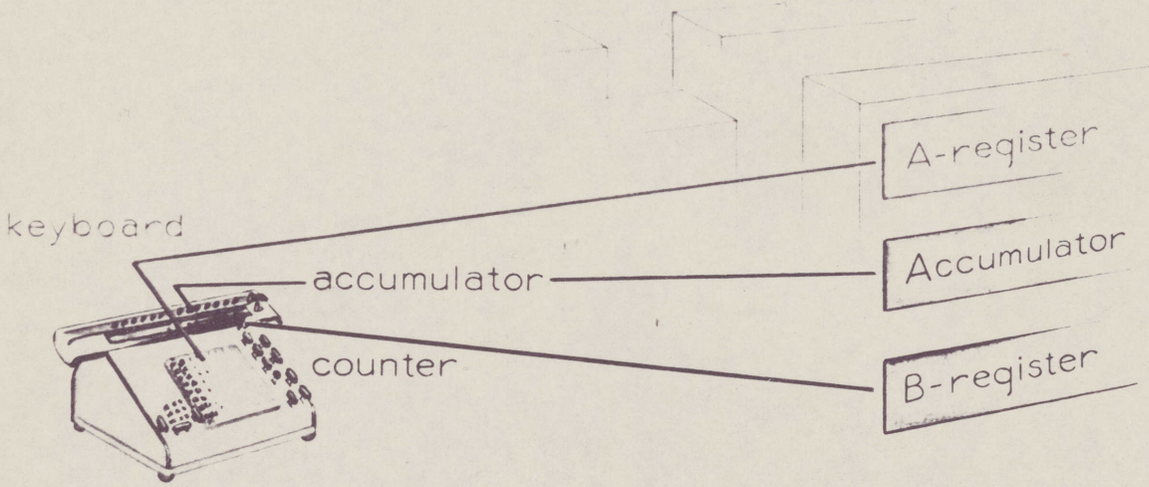
A-3623
F1034
SC-267

basic computer elements

ARITHMETIC ELEMENT

of WHIRLWIND I compared with a desk calculator





Receives number from the main bus-Holds addend, subtrahend, multiplicand or divisor

The adding unit-Holds the result of an addition, multiplication or subtraction and the dividend in division

Auxiliary register-Holds the multiplier or quotient

DESK CALCULATOR

ARITHMETIC ELEMENT

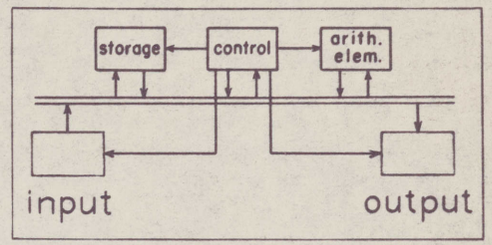
PERFORMS ACTUAL ARITHMETIC OPERATIONS SUCH AS —

add • subtract • multiply • divide

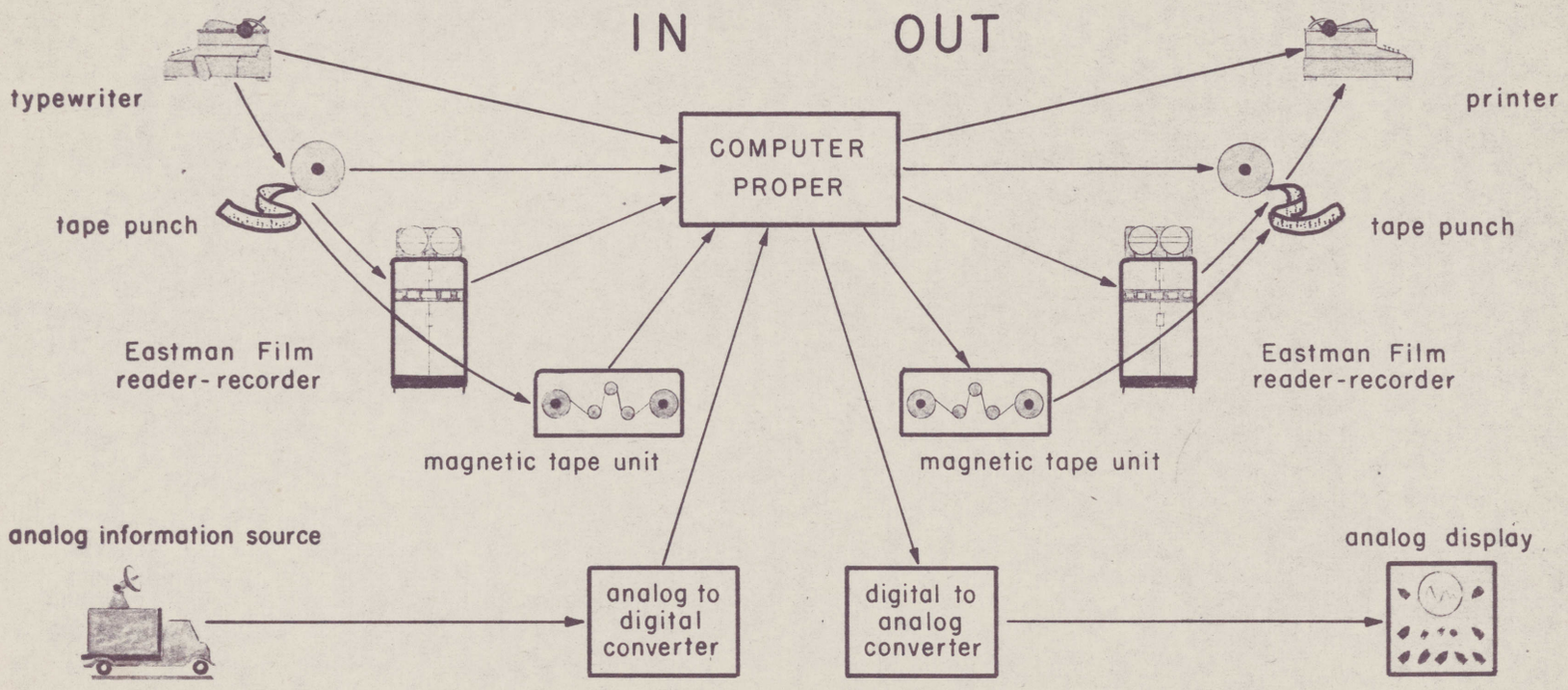
basic computer elements

INPUT AND OUTPUT

alternative ways of getting information into and out of WHIRLWIND I

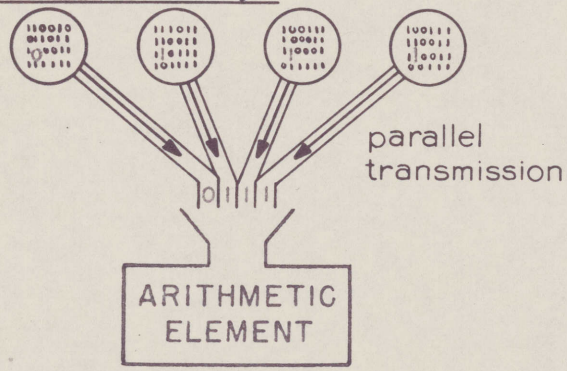


SELECTION OF DEVICE USED IS DONE BY CONTROL



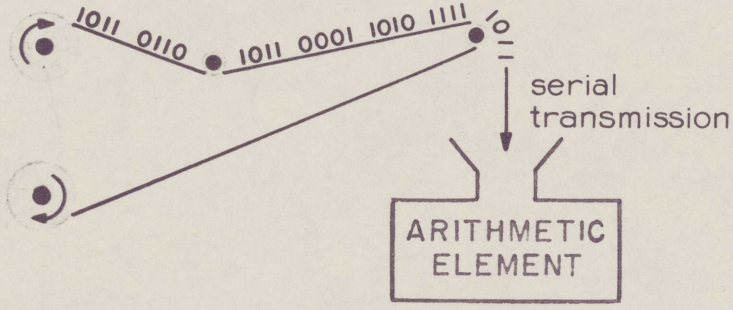
WHIRLWIND I
IS A PARALLEL COMPUTER

parallel storage



In a *PARALLEL* computer, all the digits of a number are stored, selected, and operated on *SIMULTANEOUSLY*

serial storage

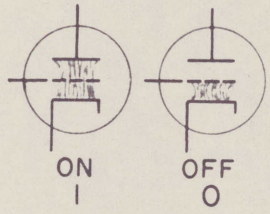


In a *SERIAL* computer, the digits of a number are stored, selected, and operated on *SEQUENTIALLY*

A-36236
F1038

WHIRLWIND I
USES BINARY ARITHMETIC

10
is the common
digital base



2
is more convenient
in electronic work

FOR EXAMPLE, IN COUNTING —

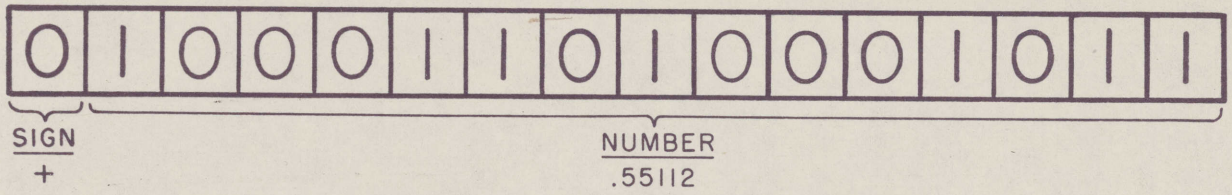
decimal	1	2	3	4	5	6	.....	10	 17
binary	1	10	11	100	101	110	.....	1010	 10001

WHIRLWIND I

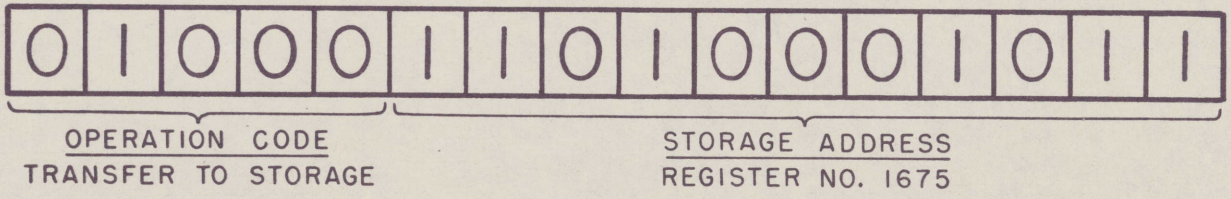
WORDS

are held in storage registers
and represent either
NUMBERS or *INSTRUCTIONS*

number

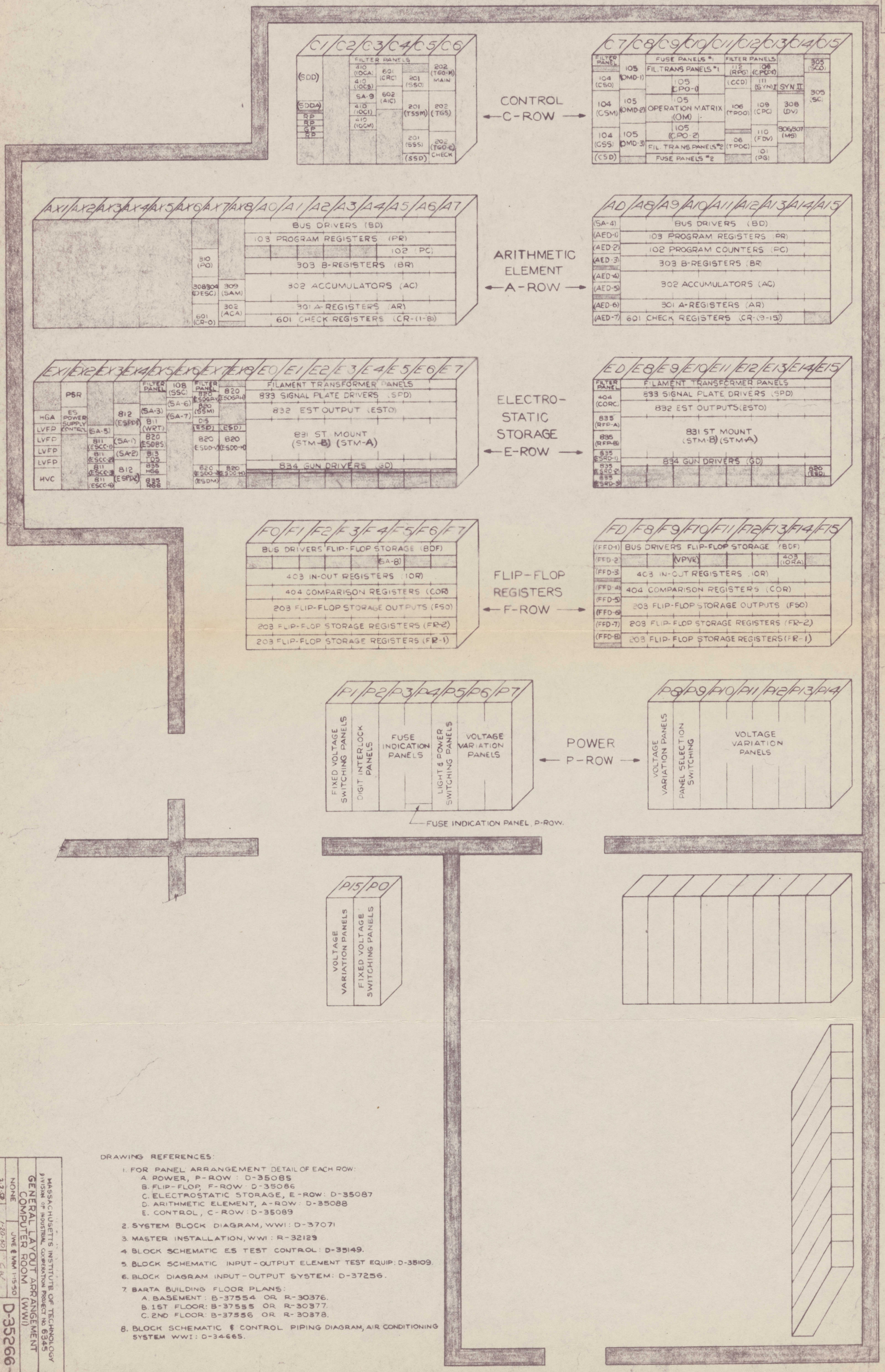


instruction



CONTROL DETERMINES WHETHER A GIVEN WORD REPRESENTS A NUMBER OR INSTRUCTION

D-35266-1



DRAWING REFERENCES:

1. FOR PANEL ARRANGEMENT DETAIL OF EACH ROW:
   A. POWER, P-ROW: D-35085
   B. FLIP-FLOP, F-ROW: D-35086
   C. ELECTROSTATIC STORAGE, E-ROW: D-35087
   D. ARITHMETIC ELEMENT, A-ROW: D-35088
   E. CONTROL, C-ROW: D-35089
2. SYSTEM BLOCK DIAGRAM, WWI: D-37071
3. MASTER INSTALLATION, WWI: R-32129
4. BLOCK SCHEMATIC ES TEST CONTROL: D-35149.
5. BLOCK SCHEMATIC INPUT-OUTPUT ELEMENT TEST EQUIP: D-35109.
6. BLOCK DIAGRAM INPUT-OUTPUT SYSTEM: D-37256.
7. BARTA BUILDING FLOOR PLANS:
   A. BASEMENT: B-37554 OR R-30376.
   B. 1ST FLOOR: B-37555 OR R-30377.
   C. 2ND FLOOR: B-37556 OR R-30378.
8. BLOCK SCHEMATIC & CONTROL PIPING DIAGRAM, AIR CONDITIONING SYSTEM WWI: D-34665.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DIVISION OF INDUSTRIAL COOPERATION PROJECT NO. 6345

GENERAL LAYOUT ARRANGEMENT COMPUTER ROOM (WWI)

NONE | JUNE & MM 1-15-50 | D-35266-1

2-2-50 | 1-20-50 | 2-10-50 | B-REDUCTION

Project Whirlwind
Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: PROJECT REPORTS OF CURRENT INTEREST: VI. MATHEMATICS

Date: December 1, 1949

No.	Title	No. of Pages	Date
A.	GENERAL OBJECTIVES		
M-61	Discussion of Numerical Methods	6	3-17-47
M-160	Mathematical Work of Project Whirlwind	7	11-12-47
B.	NUMERICAL METHODS		
	1. Solution of Algebraic Equations		
*E-148	The Solution of Systems of Linear Algebraic Equations by Successive Approximation	36	9-29-48
*M-206	Numerical Solution of Systems of Simultaneous Linear Algebraic Equations by Elimination Methods	17	1-8-48
	2. Solution of Differential Equations		
R-114	A Numerical Integration Method	10	2-24-47
*E-162	Extension of Runge-Kutta Method to Differential Equations of Order N, Where N is Greater than Four	10	11-15-48
*M-176	Methods of Numerical Integration of Ordinary Differential Equations Most Suitable for Use by WWI and WWII	22	12-9-47

* Of basic importance.

No.	Title	No. of Pages	Date
C-50		6	5-3-48
C-51		3	5-5-48
C-52		5	5-10-48
C-53	Runge-Kutta Method of Numerical Integration	7	5-12-48
C-54	(Project Whirlwind Seminar Series)	4	5-17-48
C-55		3	5-19-48
C-56		5	5-24-48
C-57		8	5-26-48
C-60	The Runge-Kutta Method for Solving Ordinary Differential Equations and its Variations	3	9-7-48
C-99	Runge-Kutta Method Applied to a Simplified Problem	6	3-6-49
	3. Solution of Integral Equations		
*E-143	Numerical Solution of Linear Integral Equations	48	9-16-48
C.	GENERAL AND THEORETICAL		
	1. Errors		
*E-147	The Error in the Runge-Kutta Method	4	9-27-48
*M-239	Order of Combination of Arithmetical Operations for Minimum Round-Off Error	4	9-23-48
	2. General Computation		
E-166	Simulation of Empirical Functions by Polynomials	27	12-3-48
E-194	Vertical Parabola through N Points	2	2-14-49
M-124	Location of Target from Combined Observations	5	10-21-47
	3. Review of the Theoretical Work of Others		
M-71	Discussion of Determinants of Large Order	4	5-1-47
M-78	Solution of Equations with Minimum Error	6	6-2-47
M-95	Report on Wiener's Ideas	4	8-11-47
M-524	Application of Newton's Method to Functional Equations	9	7-7-48
E-152	Review of Work of Kantorovich and Krylov: "Approximate Solution of Partial Differential Equations"	13	10-7-48

* Of basic importance.

No.	Title	No. of Pages	Date
	4. Thesis Work		
R-165	Convergence of the Gauss-Seidel Iterative Method	11	4-15-49
D.	BINARY NUMBERS		
R-90	The Binary System of Numbers	13	1-15-46
M-788	Suggestions for Mental or Manual Binary Conversion	3	7-1-49
M-792	Tables of Binary and Decimal Numbers	5	2-23-49

Project Whirlwind
Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: PROJECT REPORTS OF CURRENT INTEREST: VII. PROGRAMMING AND APPLICATION STUDIES

Date: December 1, 1949

No.	Title	No. of Pages	Date
A. Basic Description			
*E-235	Description of Whirlwind I Codes	16	5-6-49
*C-93	Notations for Coding	10	2-14-49
	A Short Guide To Coding	2	10-49
B. Examples			
	1. Example of Coding Procedure		
*C-94		11	2-14-49
	2. Solution of Simultaneous Equations by Iteration		
C-78		3	12-7-48
C-81		5	2-14-49
	3. Sorting Problems		
C-68		2	11-16-48
C-71		1	11-19-48
C-72		7	11-22-48

* Of basic importance

No.	Title	No. of Pages	Date
	4. Ship Control Problem		
C-75		2	12-2-48
C-79		2	12-7-48
C-80		3	12-10-48
C-82		1	12-15-48
C-84		3	12-23-48
C-85		2	12-29-48
C-86		4	1-6-49
C-87		2	1-12-49
C-88		5	1-19-49
C-89		2	1-21-49
C-90		4	1-28-49
	5. Ballistic Problem		
C-91		2	1-31-49
C-92		2	2-2-49
C-95		2	2-11-49
C-98		4	2-18-49
C-103		13	4-1-49
	C. Packaged Programs		
E-161	Code for the Solution of Simultaneous Equations by Elimination	32	11-4-48
E-170	Codes for the Evaluation of e^{-x} and ln x	9	1-18-49
C-70	Codes for Sine and Cosine	5	11-22-48
C-74	Code for the Square Root	3	12-3-48
C-77	Codes for the Square Root II	6	12-6-48
	D. Specialized Procedures		
	1. Double-Length Operations		
M-328	Double-Length Operations in WWI	11	3-31-48
M-549	Comment on the Two-Register Method	4	7-22-48
M-659	The Problem of Scale Factor as Applied to Non-Simulator Problems	12	10-20-48

No.	Title	No. of Pages	Date
F. Test and Display Programs (continued)			
E-297	Test Program No. 3: Counting in A-C	3	10-13-49
E-298	Test Program No. 4: Shifting In A-C	3	10-13-49
E-300	Display Program No. 1: Family of Parabolas and Powers of X	3	10-17-49
E-304	Display Program No. 2: Second Order Differential Equations	6	10-25-49
G. Application Studies			
R-170	Analysis and Design of Sampled-Data Control Systems	154	6-30-49
R-172	The Study of Non-Linear Servomechanisms With the Aid of an Automatic Digital Computer	163	9-26-49
M-768	Navy Logistics: - Note for Members of the ONR Committee Studying the Application of Computers	7	2-4-49

Air Traffic Control Project
Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: EQUIPMENT AND TECHNIQUES FOR INSERTING INFORMATION INTO WWI

To: Mathematics, Block Diagrams, and Input-Output Groups

From: C. W. Adams

Date: July 12, 1950

Abstract: The terminal equipment to handle the insertion of numerical data into Whirlwind I will consist of Eastman Kodak reader-recorder film units and Flexowriter typewriters, printers, tape readers and tape punches. Information will be typed in duplicate in a standard form using Flexowriter equipment. The computer will automatically convert Flexowriter-coded information into its proper form. It will also automatically select any desired routines or subroutines from a film library and put these routines into any desired storage location. Brief descriptions of the terminal equipment and a general outline of the techniques proposed for using the equipment are given in this note.

Table of Contents:

A. Introduction

Although several notes have been written in the past few years describing the Whirlwind order code and methods using that code, and although a number of coded programs have already been written, no specific techniques have heretofore been described for actually setting the computer up to perform a computation. Now that the computer has become a physical reality and the necessary terminal equipment has been obtained, however, it becomes necessary to describe the techniques and the equipment which are to be employed in preparing programs in a form usable by the computer.

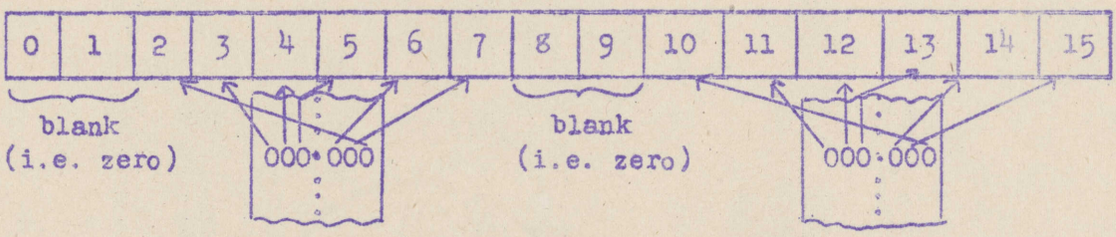
B. Brief Description of the Proposed Terminal Equipment

According to the present plans, input-output equipment will consist of (a) six Eastman Kodak reader-recorder units; (b) three complete Flexowriter units each consisting of a typewriter, a printer, a tape punch which can be actuated by the typewriter, and a tape reader which can control the printer; and (c) three cathode-ray display tubes, one of which is primarily intended for photographic recording of plotted results. The manner in which these units will be used is described below and is illustrated in the attached pictorial schematic (E-35714).

The Eastman Kodak reader-recorders are capable of reading or recording binary digital information on film at a rate of 2000 words per second. Each word can be 25 binary digits long. For checking purposes, the complement of each word is recorded or read concurrently with the recording or reading of the word (no extra time is required for handling and checking the complements). Only 16-digit words will be used in Whirlwind I, so that 9 digit spaces will be unused on the film. It should be emphasized that the reader-recorders can do only one job at a time -- i.e., that each particular unit is at any given time either a reader or a recorder but not both. Furthermore, a recorder works with unexposed and undeveloped film, a reader with exposed and developed film. The film produced by a recorder must be developed in an external unit at a rate of about 6 inches per minute (10 words per second). The movement of film from a recorder to the developer and thence to a reader is manual. Developing cannot begin until recording is complete; reading cannot begin until developing is complete.

The Flexowriter equipment punches or reads tape at a rate of about 8 lines per second. Each line of tape contains a feed hole and room for 7 punched holes; and each typewriter key corresponds

uniquely to a single 6-binary-digit (hole or no hole) character, the seventh space being left blank. The tape can be read as soon as it comes out of the punch. Two 6-digit characters are then put into a 16-digit coded word in Whirlwind I or on film, the remaining 4 digits of the word being always zero, in the following fashion:



Extra equipment will be provided so that each character typed on tape will be followed automatically by its complement for checking purposes. Punching or reading the complement on tape is, unlike in the film equipment, a separate operation. Two characters and their complements, and therefore 4 lines of tape, are required to make up one word. Furthermore, the coded information punched on tape must be converted eventually to pure binary form and each pure binary word will generally be made up from 8 typed characters (see page 5). Therefore, each binary word will require 8 characters and their complements, 16 characters in all, and will therefore be handled at a rate of about 1/2 word per second. Samples of a first tape (no complements), a second tape (with complements), and a film recorded from the second tape are shown in the attached photograph (A-35681).

The cathode-ray display tubes will have their deflection plates connected to decoders which will provide horizontal and vertical deflections proportional to the size of the binary numbers read into the decoders by the computer, Thus points can easily be plotted on the tube faces in Cartesian coordinates. Alternatively, horizontal or rotating time-base sweeps may be provided to permit Cartesian or polar plots in which a single computed coordinate is plotted against time (this relies on a constant time interval occurring between successive points). The decoders, and consequently the display tubes, will very likely be able to handle binary words faster than the computer can provide them. In fact, the present temporary display system, working with the computer using 32 registers of very-rapid-access test storage, displays up to 50,000 words per second.

C. Brief Description of the Proposed Library Film

In addition to the equipment just described, there will be a library film on which are stored in pure binary form an initial program and a set, or library, of standard routines and subroutines.

The initial program is the program, written once and for all, which is to be read into the computer initially in order to direct the computer:

(1) To read-in the desired main program word by word from film or from tape;

(2) To convert the words from Flexowriter-coded form to pure binary;

(3) To store each converted word in its assigned location;

(4) To determine, from specific information given with the main program, the subroutines which are needed and the storage locations in which each is to be located;

(5) To select the proper subroutines from the library film and store each in its assigned storage location;

(6) To adapt each subroutine to its assigned storage location; (i.e. each subroutine must originally have been written assuming that it is to be stored starting at some definite storage address, whereas in each application it will be actually stored starting at some other register and must therefore be adapted).

(7) To start execution of main program.

The standard routines are programs which are used frequently enough to warrant their being stored on the library film for easy reference when needed, thus eliminating the need for any hand operation in preparing the computer for a routine program. For instance, an accounting or inventory problem may be done once each week with new data, or an oil well or ballistic problem may be performed each time a new situation has given rise to new data. Such programs will be put on the library film in pure binary form and will then be available once and for all.

The standard subroutines are programs written to perform frequently-used computations, such as the evaluation of sin x or log x. Subroutines, unlike routines, do not stand by themselves but are intended to be components of a larger program (cf. E-329: Techniques for Using Standard Automatic Subroutines). These routines and subroutines are to be stored in pure binary form in numerical order on a roll of film, the number of each of the routines and subroutines being assigned

in the order in which it is written. The distinction between a standard routine and a standard subroutine is logically important, but in practice the two can be handled in identical fashion so that the distinction is of no practical importance.

D. Procedure by which a Coded Program can be Prepared for WWI.

In order to actually prepare the computer to solve a problem, a coded program for the particular problem must be written out in full, leaving enough space for the standard subroutines desired. The address of the storage register in which each word (order or number) to be stored must be clearly indicated according to the convention that the words are stored consecutively unless otherwise indicated. Thus the first word is to be stored in some explicitly indicated storage register, and the succeeding words are to be stored consecutively (except that, at any time, a new address given explicitly will indicate that the next word is to be stored at the new address and succeeding words consecutively from there). If a standard routine or subroutine has been used in the program and is to be taken automatically from film, its library number is simply indicated at the place in the program at which it is to be stored. If more than one subroutine is desired, the library numbers of each are given. These subroutines will then be automatically selected from the library film by the computer (under the direction of the initial program) and will be stored in consecutive storage registers, starting at the indicated place in the program.

Each item in the program -- each order, number, storage address, and subroutine number -- is typed out as a single line in a columnar form on a Flexowriter typewriter. The following conventions must be observed:

(1) An order is written as a conventional pair of small letters, a single space, and four decimal digits (e.g. ca 1234 or sl 0015).

(2) A number is written as either a + or a -, a single space, a decimal point, and four decimal digits (e.g. + .1234 or - .9870).

(3) A new storage address is written as a capital letter A followed by four decimal digits (e.g. A0793).

(4) A desired subroutine number is written as a capital letter S followed by four decimal digits (e.g. S0093).

(5) Each item is written in a single, separate line, thus requiring a carriage return at the end of each item.

(6) These four different items can be intermixed at will except that all orders are to be given before any subroutine number is given.

(7) The end of the program is indicated by typing a line of 7 virgules (i.e. ///////).

(8) No other symbol or machine function is to be used except those permitted above and the exact form given must be adhered to, including spaces, decimal points, zeros, and plus signs.

For checking purposes, the manuscript form of the program is copied twice on two different Flexowriter typewriters by two different typists. The first typing is performed in a straightforward way, with the standard Flexowriter procedure for nullifying typographical errors being followed where necessary. The tape prepared by the first typing is put in a tape reader and each character on the first tape is automatically compared with the character typed by the second typist. As long as the first typing and the check typing agree (overlooking automatically any errors that were correctly nullified in the first typing), the desired character is punched on a second tape (or in special cases it can be recorded on film or read directly into WWI), while if the two fail to agree, an alarm is sounded and nothing is recorded. In the event of an alarm, the typist must determine the source of error, correct it, and proceed. Thus, in the second typing, the procedure for nullifying typographical errors is unnecessary and is replaced by the new procedure of simply retyping if the second typing caused the error or of overriding the first tape if the error was in that tape. On the second tape, each character is followed automatically by its complement. This complement is used for checking the actual punching and subsequent reading of the tape. The checked tape can then be recorded on film (or in special cases read directly into the computer).

Briefly then, the transfer from the typewriter to the computer is usually accomplished by first punching duplicate tapes and then preparing a film, although in special cases the transfer can be performed directly, or via film or via tape alone. The intermediate processes are intended to serve as buffers of increasing speed between the low typing speed and the high computer speed. Generally, the computer's time will be too valuable to permit it to take much

information directly from a typewriter because the efficiency would be much less than 1%. By the same token, reading a self-checking punched tape is nearly as slow as good typing and so the tape-to-computer link is also usually inefficient. The advantage of the intermediate use of tape is that it is cheap, fairly reliable, sturdy, easily handled and easily read, so that it is ideal for use in a manual process which may require correction of many errors. Then the routine, automatic transfer from tape to film can be accomplished at the convenience of the faster, more expensive recorder with no unnecessary delays for a typist to correct her mistakes. A film can be prepared from the tape whenever a recorder is available, and the recorded film can then be read into the computer at the proper time, reliably and rapidly.

E. Procedure by which WWI can be Prepared for a New Program

The computer can be actually started up by the following steps. A film kept in one reader, the library reader, is so arranged that when the reader is started, the first recorded words encountered on the film by the reader will be the start of the initial program, recorded in pure binary form. The computer is cleared by means of a pushbutton, and toggle switches are set to select the library reader and to indicate the address of the storage register in which the first word taken from the library film is to be stored. Then the computer and the reader are started simultaneously by a push-button. Since the control switch of the computer is initially clear and since the cleared position (00000) designates the operation ri, the computer performs operation ri which reads in the pure binary words of the initial program one at a time as they come from the library film and transfers them into consecutive storage registers, starting at the register indicated by the toggle switches and continuing until storage is full -- i.e. until a word has been put in register 255,511,1023 or 2047 depending on the storage capacity available in the computer. The storage register into which the first word is put is so selected that the last word of the initial program falls into the last available storage register (255,511,1023 or 2047). When that last word has been read in, the computer automatically stops the library reader and starts to perform the initial program which has just been read in.

In addition to the library reader there is a second, or input, reader in which is the film, prepared from the Flexowriter tape, containing the coded characters corresponding to the typewritten form of the desired program. The film is divided into blocks of four words corresponding to one line of typewritten copy, with enough space between blocks to allow the film to be stopped between blocks without coasting into the next block. Directed by the initial program, the computer starts the input reader, takes in one block of

coded characters, stops the input reader, translates the block just read in, and then continues to take in one block at a time, starting and stopping the input reader once for each block. If the information in the block turns out to be an address, this new address is translated and is stored in the register which is assigned to keep track of the address at which the next word is to be stored. If the block turns out to be an order or a number, it is properly translated into pure binary and is stored in the register designated by the address indication. Then the address indication is increased by one in preparation for the next word.

The first block that turns out to be a request for a subroutine initiates a small program that starts the library reader and reads in a subroutine-selection routine, storing it in registers formerly used by the initial program for translating orders. This selection routine then allows the library reader to run and counts, checks and discards the words read in by that reader until the desired subroutine is reached. After the desired subroutine has been read into the computer, the library reader is stopped. The address section of each order in the subroutine is examined and is changed if necessary to adapt the subroutine to the storage location now assigned to it.

To facilitate the performance of the procedure just described, one prepares the library film according to the following conventions. The standard subroutines are all numbered and stored consecutively on the library film in pure binary form in blocks -- one block for each routine, so that these blocks are of varying length rather than being always just four words long. The first word of each block is the block number and hence the subroutine number. The second word in each block gives the total number of words (orders and numbers) in that subroutine and the third word gives the number of orders in that subroutine. Each and every subroutine is written originally (before adaptation by the computer) as if it is to be stored beginning in storage register 1024. The address sections of orders referring to other registers within the subroutine are then no less than 1024, while the address sections of orders like sl, sr, rf and rs are always kept less than 1024. The orders are all at the beginning, the numbers all at the end of each subroutine. Knowing how many orders there are, and that the orders come first, the computer can tell whether it is dealing with an order or a number. By examining the address section of each order and changing only those that are not less than 1024, the computer can properly adapt the subroutine to its assigned location in storage. It can then store the adapted subroutine (both orders and numbers) in the proper registers, start the input reader

again, and continue with the next block of Flexowriter characters. More requests for subroutines can be handled in the same way, but no more orders can be translated because the order-translating portion of the initial program has been replaced by the subroutine selection routine. The purpose of replacing the one by the other is simply to reduce the amount of storage required by the initial program at any one time.

F. Conclusion

Much of the equipment described in this note is still being developed and tested, so that the system illustrated in the attached pictorial schematic will not come completely into being for at least a year. In the meantime, a direct tape-to-computer reader with checking, shifting and control performed by the computer, will be used as a primary source of input. As soon as film equipment passes the testing stage, film will be prepared from tape, again making use of the computer for checking, shifting and control. This interim system, which will be described in a forthcoming note, will require new programs for handling the direct type read-in, but the techniques described in section D will be essentially no different in the interim than in the final system.

Techniques for manual control (i.e. stopping the computer at a designed place, examining contents of specified electrostatic storage registers, and inserting new orders or parameters) will be necessary to give the programmer control over the computer after it has been started on a program. These techniques will probably involve the use of the toggle-switches and flip-flops comprising test storage, and they will be described in a note as soon as they are worked out in full.

The techniques described in this note are to be considered as proposals subject to considerable modification.

Signed: C. W. Adams
C. W. Adams

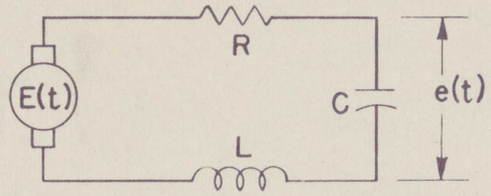
Approved: RRE
R. R. Everett

CWA:bm

Attached: A-35681
E-35714

WHIRLWIND I DISPLAY

OF TYPICAL DRIVING FUNCTIONS AND RESPONSES



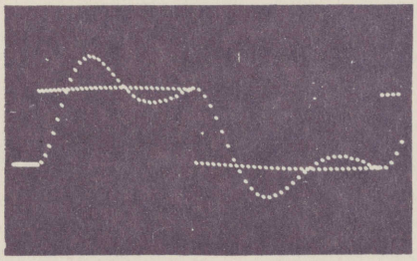
$$\frac{d^2e}{dt^2} + 2Z\omega\frac{de}{dt} + \omega^2 e = \omega^2 E(t)$$

$Z = \frac{R}{2}\sqrt{\frac{C}{L}}$ = DAMPING RATIO $\qquad \omega = \frac{1}{\sqrt{LC}}$ = FREQUENCY

h = INTERVAL BETWEEN POINTS

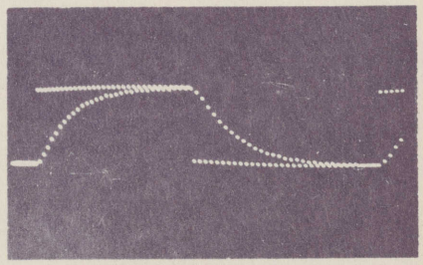
PLOTS OF E(t) AND e(t) AGAINST TIME

CALCULATED USING DISPLAY PROGRAM NUMBER IV



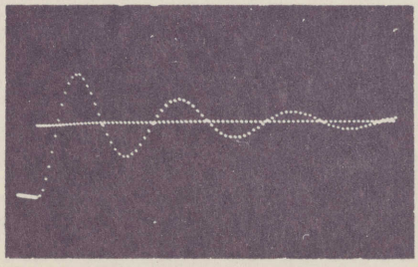
E(t) = SQUARE WAVE

$Z = \frac{1}{4}$ $\qquad \omega h = \frac{1}{4}$



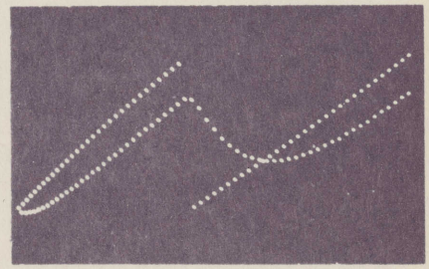
E(t) = SQUARE WAVE

$Z = 1.0$ $\qquad \omega h = \frac{1}{4}$



E(t) = STEP FUNCTION

$Z = \frac{1}{8}$ $\qquad \omega h = \frac{1}{4}$

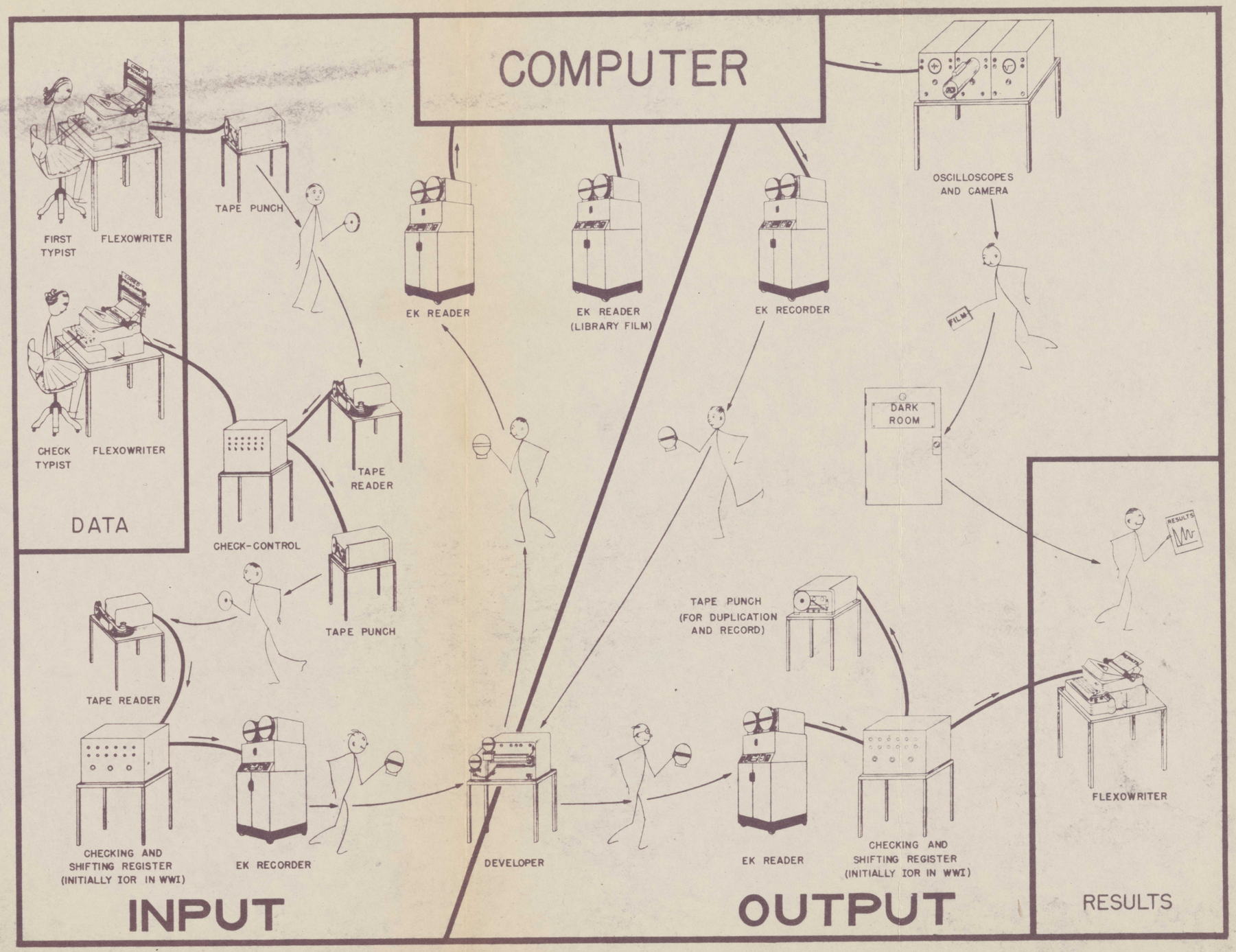


E(t) = SAWTOOTH WAVE

$Z = 1.0$ $\qquad \omega h = \frac{1}{4}$

A-35128

E-35714



GENERAL-PURPOSE TERMINAL EQUIPMENT

Project Whirlwind
Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: TECHNIQUES FOR USING STANDARD AUTOMATIC SUBROUTINES

To: Mathematics Group

From: C. W. Adams

Date: February 10, 1950

Abstract: Standard automatic subroutines are programs for the evaluation of frequently needed functions or for the performance of routine computational chores. Such subroutines, which are intended primarily for use as a part of a larger main program, will be kept on permanent file in the slow-speed memory of a computer and will be inserted into high-speed memory along with any program of which they are to be a part. A subroutine, if it is to be referred to from several different parts of a main program, must be prepared for each use by the insertion of new addresses into some if its orders.

In the present note, various preparation techniques are discussed. Most promising is a deferred preparation, in which the only preparatory order needed in the main program is a transfer of control order, all of the necessary changes in the subroutine being performed by orders in the subroutine.

In the Whirlwind computer the address of the register next after the register containing a transfer of control is stored automatically in the A register by the transfer of control (sp) order. This register, next after the one containing the sp order, is a prearranged location characteristic of one and only one sp order, and as such this register and the ones following it can be used to store any addresses which must be available to the subroutine in making the necessary preparations. That is, the correct return address and the addresses of registers which are to contain all necessary extra addresses can all be deduced within the subroutine with no assistance from the main program.

Introduction

The simplest computer imaginable would need to have only a few basic abilities -- in fact, the ability to (1) sense and (2) complement a selected binary digit would be sufficient. Unfortunately, many repeated

applications of these basic operations would be required to perform a single complete addition since the addition of each pair of digits and each carry would have to be ordered separately; so the program for any sizable problem would be unwieldy at best. In designing a computer, therefore, one builds a control and an arithmetic element of at least enough complexity to be able to perform many of the elementary arithmetic operations in response to single orders in the program. The number of elementary operations so built in may influence the speed, effective memory capacity and convenience in use of the computer, but it in no way limits the capability of the machine. For just as addition can be programmed using only a single-digit sense-and-complement order, any of the more complicated numerical operations can be programmed using the simpler built-in operations.

The Whirlwind computer has built into it the abilities to add, subtract, multiply, divide, find magnitude, and point off (find the characteristic of the $\log_2 x$ for any given x), using numbers within the prescribed range. Other operations, such as the evaluation of $\sqrt{x}$, $\log x$, e^x, $\sin x$, etc., are not built in. The line between included and excluded operations is drawn, as it is in all such cases, by a compromise based on such considerations as the flexibility, convenience, speed, economy of construction, simplicity, and ease of maintenance desired; these considerations hinge in turn on the intended purpose of the machine. The decision is made less difficult by the comparative ease with which the non-automatized operations can be synthesized from the elementary ones and made to seem almost automatic.

Purpose

The intention of this report is to combine and bring up to date under one cover the pertinent information on the means of accomplishing programmed automatization of the non-elementary but nonetheless common numerical operations. In the present report, a working knowledge of coding for the Whirlwind computer is presupposed. The Appendix contains a summary guide to coding and a brief but exact description of the Whirlwind order code as of January 1950.

Standard Automatic Subroutines

Programs that are commonly used as part of a larger program are called subroutines. Because the subroutines described in this report will be so arranged that they seem to be built in, these subroutines are called automatic. Only one or at most a few of the many possible variants of the subroutines for each function will be kept permanently available (in the case of sin x, for example, different routines would be provided at least for (1) x in revolutions, (2) x in radians, (3) x in radians times some scale factor.) Such subroutines will then be called standard

automatic subroutines for the evaluation of the given functions. In this note, techniques for preparing to use the subroutines are discussed at length.

Standard automatic subroutines will be kept on permanent file on the slow-speed film storage of the Whirlwind computer and each will be inserted into high-speed storage along with any program in which the particular subroutine is needed. Some indication of the necessary subroutines and of the addresses of the storage registers to be occupied by each subroutine will be given as part of each main program. That is, in writing each program, the programmer will be able to select for use in his program any of the standard automatic subroutines and he will be able to designate the storage registers in which each of the selected subroutines is to be stored, subject only to the condition that his assignments are compatible with the length of each subroutine so that there are enough consecutive storage registers available for each of the subroutines.

According to the present plans for Whirlwind I, all of the standard automatic subroutines will have been written, converted to binary form, and stored in some order on one (or more if necessary) roll of film called a library film. Each subroutine will have been written under the assumption that it is to be stored beginning at register #1024. All the necessary constants, except possibly some universal constants like 1/2, will be included as a part of the subroutine, stored in registers immediately following the last order of the subroutine. The purpose of writing each subroutine starting at register 1024 is to permit easy discrimination by the computer between those orders whose address sections refer to other parts of the subroutine and those orders whose addresses are irrelevant or refer to something else (such as a number of shifts, an address of an external device, or an address of some universal constant, all of which would normally be less than 1024).

The actual insertion of a program into the computer will probably be done with the aid of three preliminary routines, all of which will presumably be stored at the beginning of the library film and will be put into the high-speed storage of the computer by means of the ri operation. The main program to be performed will have been typed out in some normal fashion on an automatic typewriter which at the same time prepares a perforated tape. (Flexowriter equipment will be used for this purpose.) Each character (i.e., each of the 50 keys and controls on the typewriter) has a six-digit binary representation on the tape and this binary-coded information can be translated in the computer to the proper binary form and stored in the proper registers by means of a suitable conversion program. The typewritten form of each new program will also have, probably at the end, some indication of the subroutines needed and the addresses assigned to each. The conversion routine, the first of the three preliminary

routines, will then turn this information, properly translated, over to the second preliminary routine, a library selection routine, which will select the desired subroutines from the library film. The third subroutine (the adaptation routine) will take each subroutine and change the addresses as necessary to adapt the subroutine to its assigned place in storage. These preliminary routines will be discussed in a subsequent note.

A library of subroutines is actually being built up, starting with the common function evaluations, most of which have already been prepared in preliminary form and published (cf. E-170, C-70, C-77 for e^x and log x, sin x and cos x, and $\sqrt{x}$ respectively). A note which will contain revised and "final" forms of these and other subroutines is also forthcoming.

Classification of Subroutines

Standard automatic subroutines can be classified according to the amount of information which must be exchanged between the subroutine and the main program each time the subroutine is used. Obviously, every subroutine must be supplied with the proper return address -- the storage address of the next order in the main program to which control is to be returned at the completion of the subroutine. Aside from the return address, many subroutines such as those for the evaluation of x, log x, e^x, sin x, etc., need only to be given the value of x and need only to supply the value of the desired function. Since the quantity x and the resulting function of x will in most cases occupy only a single register each, the simplest procedure, apparently, is for the main program to put the quantity x into the Accumulator (AC) just before transferring control to the subroutine and for the subroutine to put the resulting function of x into AC just before returning control to the main program. Thus in this case no storage address, other than the return address, needs to be exchanged. Subroutines of this type, requiring the exchange of no addresses, will be said to be zero-address subroutines.

Some subroutines require the exchange of some number other than the quantity x and the result. For example, the quantity x or its result may be double-length -- i.e., require two registers to accommodate it because of its magnitude or its precision or both. (Frequently, however, a two-register result such as a number and a scale factor will be obtained from a zero-address subroutine since the result can easily be stored with the number in AC and the scale factor in some predetermined register, chosen once and for all.) Or, as another example, a subroutine intended to shift the contents of AC and BR left without roundoff must be supplied with the number of shifts to be performed. In such cases it is necessary for the main program to supply an address, over and above the return address, to

the subroutine -- that address being either the address at which some necessary quantity will be found or at which some result is to be stored. Subroutines of this type, requiring that one address be exchanged, will be called one-address subroutines.

Similarly, other subroutines require the exchange of two, three or more orders. Thus, finding a quotient plus a remainder requires two addresses, one at which the divisor is stored and one at which the remainder is to be stored, with the dividend being put into AC and the quotient appearing in AC. The double-length arithmetic operations (addition, subtraction, multiplication, division) generally require three addresses, one for each of two operands and one for the result, since the three quantities involved are each double-length and cannot be stored in AC. In the case of double-length numbers, three pairs of addresses, six in all, are actually required, but it is assumed that the two halves of a double length number are stored in consecutive registers so that the second address of a pair can always be deduced from the first address and is not therefore a separate piece of information. As before, the number of addresses, exclusive of the return address, characterize the subroutine so that the quotient-and-remainder program is two-address while the double length operations are three-address subroutines.

Automatization of Zero-Address Subroutines

Fundamentally a subroutine is a set of orders which is used in several different parts of a main program but which is only to be put in one place in storage. The problem of automatizing the subroutine is just the problem of how to permit the subroutine to be effectively inserted into the main program by unconditional transfers of control from the main program to the subroutine and back again. By definition, the zero-address subroutines require the exchange of no address except the return address.

Suppose, for example, that the programmer has requested that a subroutine for the calculation of the square root of a number be stored starting in register 938. Suppose further that, at the completion of the main program order stored in register 619, the quantity x is in AC and that the square root of x is wanted. Then the order stored in register 620 might be sp 938 which would transfer control to the start of the square root subroutine. At the end of the square root subroutine is another sp order, which in this case should be sp 621, to return control to the proper point in the main program. This address 621, the return address, must be supplied from somewhere. It obviously cannot be simply written in once and for all, for the subroutine will probably be referred to from several different places in the main program and the return address will differ in each case.

One way to supply the return address would be to store the return address, which is known in each case, in some predetermined register. Then in the main program, before the sp 938 order, one could clear and add that address and transfer it, using the td operation, to the register containing the sp order at the end of the subroutine. This procedure requires at least two extra orders and one extra register of constant storage for each place in which the subroutine is to be inserted into the program. In addition it presupposes anticipation by the programmer who must be sure that the ca and td sequence is inserted in the main program before the quantity x is formed in AC, it being assumed that x is to be in AC when control is transferred to the subroutine.

	Reg.	Content	
main program	606)	ca 833	anticipated preparation
	607)	td 964	
	620)	sp 938	transfer of control
constants	833)	ri 621	stored constant (operation ri is used because its coded representation is 00000)
subroutines	938)	sf 965	desired subroutine
	964)	sp --	return of control
	965)	---	

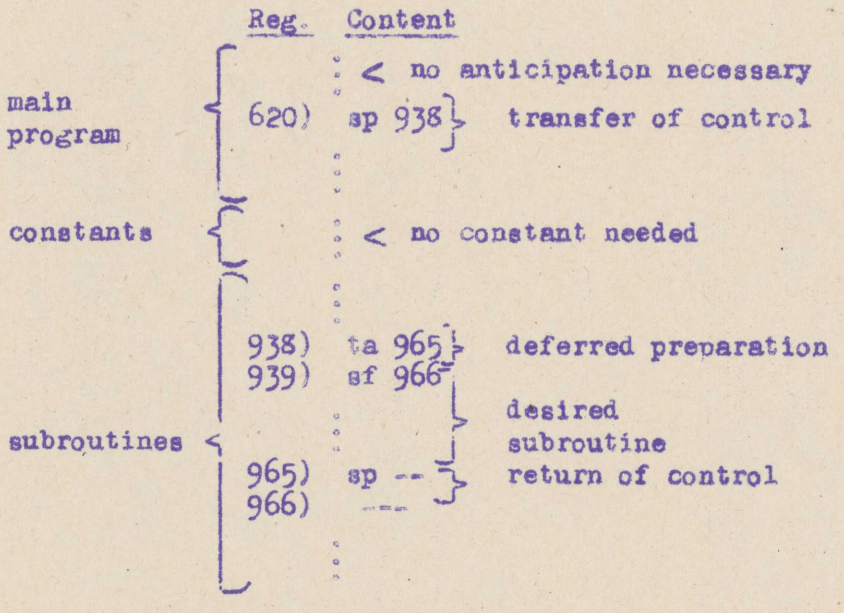
ANTICIPATED PREPARATION FOR
A ZERO-ADDRESS SUBROUTINE

This anticipated preparation of the return address is possible in almost any computer. But in Whirlwind I, the addition of one simple feature to the sp operation and the addition of one new operation, ta, has removed the need for the clumsy procedure just described. The function of these two orders is as follows:

sp x - Transfer control unconditionally to register x. Assuming that the sp x order is stored in register y, store the quantity y+1 (which is obtained from the program counter before control is transferred) in the last 11 digit positions of AR.

ta x - Transfer the last 11 digits from AR to the last 11 digit positions of register x, leaving the first 5 digits unchanged.

Thus in the example of the previous paragraph, the return address 621 would be stored in AR, without affecting the contents of AC, by the sp 938 order which was stored in register 620. Then the first order in the subroutine would be a ta order which would transfer the return address to the last order, the sp order, at the end of the subroutine. The intervening orders of the subroutine would determine the square root of the number in AC and leave the result in AC. In this way the red tape involved in using the zero-address subroutines is reduced to practically nothing. The square root of the number is found by simply inserting the order sp 938 into the program whenever the square root of the contents of AC is wanted. Using the zero-address subroutines requires no more thought or effort than using a single operation built into the computer.



DEFERRED PREPARATION FOR
A ZERO-ADDRESS SUBROUTINE

Automatization of One-Address Subroutines

A one-address subroutine can be automatized in much the same way as a zero-address subroutine. The only difference is, of course, that some storage address in addition to the return address must be made available to the subroutine from the main program. Consider, for example, the subroutine for shifting left without rounding off. It is assumed that the number to be shifted is in AC and BR when control is transferred to the subroutine and that the shifted result is to be left in AC and BR at the end of the subroutine.

The subroutine itself could be

Reg.	Content	
718)	ts 751	store content of AC
719)	sl 15	store content of BR
720)	ts 752	
721)	ca 751	shift and store the original content of AC
722)	sl n	
723)	ts 751	
724)	ca 752	shift content of BR
725)	mh 753+n	
726)	ad 751	add in shifted content of AC
727)	sp --	return of control
751)	---	
752)	---	
753)	2^{-15}	
	⋮	
753+n)	2^{n-15}	
	⋮	
767)	2^{-1}	

EXAMPLE: A SUB-ROUTINE TO SHIFT LEFT WITHOUT ROUNDOFF

Actually, such a subroutine would be valid only for n in the range $0 \leq n \leq 14$ and consequently the standard automatic subroutine might well be slightly more complicated and more general. The subroutine just given is, however, satisfactory as an example of various preparation techniques.

The most obvious technique is anticipated preparation of the extra address, using deferred preparation of the return address. In the example given, the number of shifts, n, must be inserted in the sl n order in register 722 and the address 753+n must be inserted in the mh 753+n order in register 725. Note that although two different addresses are needed, one may readily be deduced from the other and consequently the subroutine is classed as a one-address subroutine. The return address must be inserted in register 727, but this can be handled as in the zero-address technique. The result is

	Reg.	Content	
main program	223)	ca 690	anticipated preparation of the extra address (es)
	224)	td 722	
	225)	ad 728	
	226)	td 725	
	240)	sp 717	transfer of control
constants	690)	ri n	stored constant address
subroutines	717)	ta 727	deferred preparation of return address
	718)	ts 751	desired shift-left subroutine
	719)	sl 15	
	720)	ts 752	
	721)	ca 751	
	722)	sl --	
	723)	ts 751	
	724)	ca 752	
	725)	mh --	
	726)	ad 751	
	727)	sp --	return of control
	728)	ri 753	constant address needed to prepare address for the mh order

ANTICIPATED PREPARATION OF
A ONE-ADDRESS SUBROUTINE

A second technique is the use of a deferred preparation of the extra address as well as of the return address. This can be accomplished by storing the extra address in some prearranged location so that the preparation orders can be put into the subroutine, thereby obviating the duplication of the preparation orders in the main program. It is not satisfactory to simply decide that the address should be stored in some one register, such as in 690 as it was in the example; for since the address will presumably be different each time the subroutine is used, the main program would still be required to transfer the address to some given location such as 690. (Although in the case at hand, in which the quantity n is needed in two different orders, some economy of orders could indeed be made by letting the main program put only the first address into the subroutine and letting orders in the subroutine form the second address from the first.)

In the deferred preparation technique to be described, the extra address is stored in a location so chosen that each reference to the subroutine from the main program designates uniquely its own prearranged location. Thus far in this discussion, the return address has been the address of the register next after the register which contains the transfer of control (the sp order). However, this register, next after the one containing the sp order, is a prearranged location characteristic of one and only one transfer of control point, and as such this register could be used to store the necessary extra address. The correct return address would then be the address of the register second after the transfer of control point.

	Reg.	Content	
main program			no anticipation necessary
	240)	sp 717	transfer of control
	241)	ri n	the necessary address, stored in a dummy order
constants			no constants needed, except those directly associated with the subroutine
subroutines	717)	ta 723	preparation for deferred preparation of extra addresses
	718)	ta 733	partial deferred preparation of the return address
	719)	ts 751	storing the contents of AC and BR (start of the shift-left subroutine)
	720)	sl 15	
	721)	ts 752	
	722)	ao 733	completion of deferred preparation of the return address
	723)	ca --	deferred preparation of the extra address (es)
	724)	td 728	
	725)	ad 734	
	726)	td 732	

(continued on next page)

	Reg.	Content	
subroutines (continued)	727)	ca 751	completion of the desired shift-left subroutine
	728)	sl --	
	729)	ts 751	
	730)	ca 752	
	731)	mh --	
	732)	ad 751	
	733)	sp --	return of control
	734)	ri 753	constant address needed to prepare address for the mh order

DEFERRED PREPARATION OF A ONE-ADDRESS SUBROUTINE

Automatization of Several-Address Subroutines

The generalization to several addresses of the preparatory techniques just described for one-address subroutines is almost trivial and little need be said. The anticipated preparation works in exactly the same fashion except that more than one address must be stored in constant storage and must then be transferred by orders in the main program to the subroutine. The deferred preparation is likewise almost unchanged; the several addresses are simply stored as dummy orders in consecutive registers immediately following the sp order in the main program, and the return address is simply the address of the register next after all of the several dummy orders.

History of the Discontinued "Automatic Subprogram" Operations

One of the most important set of three-address subroutines for a computer with comparatively short register length is the double-length operation subroutines. Some time ago (cf. M-111, pps. 8-10, dated October 6, 1947) it was proposed that a set of five special operations be incorporated into the design of the Whirlwind computer primarily to facilitate work with double-length numbers. The first of these operations, designated by as, would (1) transfer the contents of AC bodily into BR and (2) transfer the as order itself into AC. Three of the operations were logically identical; these operations, designated by ax, ay, az, would (1) transfer the ax (or ay or az) order itself into AR, (2) transfer the return address from the program counter to register 2047, and (3) transfer control unconditionally to some preselected -- i.e., wired in -- storage address, three different addresses being selected, one by ax, one by ay, one by az. A fifth operation, logically almost equivalent to the present ta operation but designated by ro, permitted the contents of AR to be read into AC.

The intended function of these operations is best illustrated by an example. Suppose a double-length addition subroutine is to be used and that the augend is stored in registers 618 and 619, the addend in registers 712 and 713, and the sum is to be stored in registers 832 and 833. Then the preparatory orders would be

```
as 618
as 712
ax 832
```

The first order would put the address 618 into AC; the second order would shift the address 618 into BR and put the address 712 into AC; the third order would put the address 832 into AR, would transfer control to a preselected position (x), and would store the return address in register 2047. Then the subroutine for double-length addition, stored beginning in register x, would proceed to unstack the various addresses stored in AC, AR and BR and supply them where needed in the subroutine, would deduce from the given addresses the second address of each pair (e.g., 619 = 618 + 1), would transfer the return address from register 2047 to the return of control (sp order) at the end of the subroutine, and would then perform the double-length addition.

The so-called "automatic subprogram" operations were eliminated from the Whirlwind I order code (cf. E-235, May 6, 1949). The reason for mentioning them here is threefold. First, these operations were referred to in a number of notes on programming techniques written in 1948 and it seems advisable to take cognizance of them for the benefit of anyone who has already or may yet encounter references to them in the literature of Project Whirlwind. Second, these operations point out at least one way in which special operations can be built into the machine to facilitate particular applications. Third, the method used is fundamentally a good one and provides a standard for comparison with other techniques. The reason that the automatic subprogram operations were dropped was simply that their value did not justify their existence when compared to the possible value of other special built-in operations. It should be noticed that there would be almost no gain, in fact less than none in some cases, in using the automatic subprogram operations in preparing for zero- or one-address subroutines because (1) only three locations (x, y and z) and consequently only three different subroutines can be used and (2) the use of the arithmetic element in storing addresses is obviated by the fact that in many cases the numbers themselves can be stored in AC even more effectively than their addresses.

Evaluation of the Preparation Techniques

There are several criteria by which a programming technique should be evaluated. The most important of these criteria can be summed up in the form of four questions.

(1) Is the technique easy to learn and to use?
(2) Does the technique reduce coding and manual preparation time?
(3) Does the technique reduce the storage capacity needed to solve the problem?
(4) Does the technique reduce the computing time needed to solve the problem?

In evaluating the three methods discussed in this report, one might prepare a table of comments on their relative merits. Of course, the use of special orders is not a technique of any practical importance in the Whirlwind or any other computer at present, since the orders do not exist; but it is well to keep the possibility in mind. It should also be noted that deferred preparation of a return address is only possible in a computer such as Whirlwind having the appropriate orders (sp and ta) in its code. But deferred preparation of extra addresses is possible in any digital computer, even if the order code requires use of anticipated preparation of the return address, for once the return address is available to the subroutine, the addresses of registers containing the extra addresses are also available.

EVALUATION OF PREPARATION TECHNIQUES

criterion	anticipated preparation	deferred preparation	use of special "automatic subprogram" orders
ease in learning and applying	quite easy to learn since no special technique of coding is needed; cumbersome in use	requires special knowledge, but once learned is easy to use	requires some special knowledge, but once learned is probably the easiest to use. Lacks generality since it cannot be applied to many subroutines.
reduction in coding and manual preparation time	wasteful and cumbersome	quite efficient since only the essential addresses need be inserted (including the address of the desired subroutine)	most efficient for double-length numbers, since the address of commonly used subroutines does not need to be specified by the main program, but has no advantage over deferred preparation in many applications
reduction in required storage capacity			
reduction in the required computing time (actually, use of subroutines necessarily increases computing time compared to not using subroutines at all)	uses two orders to prepare the return address, which is poor; uses two orders to prepare each distinct extra address, which is good.	uses one order to prepare the return address, which is good; uses four orders to prepare each distinct extra address, which is poor.	uses two orders to prepare the return address, which is poor; uses two, plus (to get an address out of BR), orders to prepare each distinct extra address, which is fair.

Conclusions

The deferred preparation technique will be used in connection with all standard automatic subroutines for Whirlwind I. By this method, only one order is needed to bring about the evaluation of a common function, and only one extra register is needed for each extra address which is to be supplied.

Use of the various preparation techniques is shown in the following example, where the program is not an example of efficient coding but merely illustrates the thoughtless, brute force way in which results can be obtained.

Required to evaluate $(e^x \sin y - e^y \sin x)2^8$

where it is known that $0 > x > -1$
$0 > y > -1$
$2^{-8} > e^x \sin y - e^y \sin x > -2^8$

Suppose the following subroutines are available:

Evaluation of e^x for $-1 < x \leq 0$, first order stored in register A

Evaluation of sin x for $-1 < x < 1$, first order stored in register B

Double-length subtraction, first order stored in register C

A subroutine to take a double-length number, shift it left n times and put it back in the same pair of registers, first order stored in register D

Other registers are assigned as follows:

register X contains x

register Y contains y

registers T1, T2, etc. are consecutive registers available for temporary storage

The program then is: (asterisks indicate use of a standard subroutine)

```
    ca X
*   sp A      } find and store e^x
    ts T1
    ca Y
*   sp B      } find sin y
```

```
    mh T1
    ts T1
    sl 15      } form and store 30-digit product e^x sin y
    ts T2
    ca Y
*   sp A       } find and store e^y
    ts T3
    ca X
*   sp B       } find sin x
    mh T3
    ts T3
    sl 15      } form and store 30-digit product e^y sin x
    ts T4
*   sp C       subtract the second product from the first
    ri T1      } address of minuend
    ri T3      } address of subtrahend
    ri T1      } address of difference
*   sp D       shift the result left 8 times
    ri 8       } number of shifts required
    ri T1      } address of operand
```

Thus it is seen that such operations as the evaluation of e^x or sin x and subtracting or shifting double-length numbers can be programmed almost as easily as if they were built in to the computer, using deferred preparation of standard automatic subroutines.

Signed C W Adams
C. W. Adams

Approved R R Everett
R. R. Everett

CWA/lfu/aec

Attached: A Short Guide to Coding

Project Whirlwind
Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: DISPLAY PROGRAM NUMBER IV: NON-HOMOGENEOUS SECOND ORDER DIFFERENTIAL EQUATION

To: N. H. Taylor, G. C. Sumner, H. L. Ziegler, G. Cooper

From: C. W. Adams

Date: January 30, 1950

Abstract: A program is presented which solves the equation $\frac{d^2e}{dt^2} + 2\zeta\omega\frac{de}{dt} + \omega^2 e = E(t$ for E(t) a square wave or sawtooth wave. Method used is a linear extrapolation-integration procedure. The program is tailored to the 32 registers of test storage available in Whirlwind I.

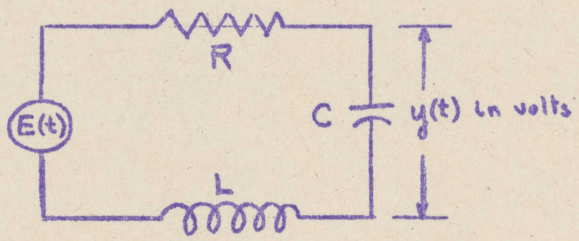
The Problem

Display Program Number IV is a modification of Display Program Number II using a less refined approximation in the numerical solution in order to permit a driving function to be generated and introduced while still keeping the program within the limits of the 32 registers of test storage.*

The equation solved (approximately) in Display Program Number IV is the non-homogeneous differential equation:

$$\frac{d^2y}{dt^2} + 2\zeta\omega\frac{dy}{dt} + \omega^2 y = \omega^2 E(t)$$

This equation describes (among other things) the response of a series RLC circuit to a driving voltage E(t).



* See the description of Display and Storage at the end of this note - Appendix I

The constants zeta (ζ) and omega (ω) are respectively the damping ratio and the natural frequency of the RLC circuit,

$$\zeta = \frac{R}{2}\sqrt{\frac{C}{L}} \qquad \omega = \frac{1}{\sqrt{LC}}$$

A numerical formula (i.e. a set of difference equations) to approximate the differential equation is obtained in a manner analogous to that described in E-304, Display Program Number II, using one less step than is taken there.

Derivation of the Formula

The equation to be solved is

$$\frac{d^2y}{dt^2} = \omega^2 E - 2\zeta\omega \frac{dy}{dt} - \omega^2 y$$

We let $\frac{dy}{dt} = \omega v$

so that $\frac{d^2y}{dt^2} = \omega \frac{dv}{dt} = \omega^2 E - 2\zeta\omega^2 v - \omega^2 y$

and letting $E - 2\zeta v - y = P$ for convenience

we get $\frac{dv}{dt} = \omega P$

Thus the original second order equation reduces to the system

$$\frac{dy}{dt} = \omega v; \quad \frac{dv}{dt} = \omega P$$

Now let $y_n = y(nh)$ be the value of y when $t = nh$, where h is a small increment of time, and

similarly let $v_n = v(nh)$; $E_n = E(nh)$; $P_n = E_n - 2\zeta v_n - y_n$

First approximation: $v_{n+1}^{(1)} = v_n$; $y_{n+1}^{(1)} = y_n$; $E_{n+1}^{(1)} = E_n$;

and so $P_{n+1}^{(1)} = P_n$

Second approximation: (integration)

$$v_{n+1}^{(2)} = v_n + \omega \int_{nh}^{nh+h} P\,dt = v_n + \omega \frac{P_n + P_{n+1}^{(1)}}{2} h$$

$$y_{n+1}^{(2)} = y_n + \omega \int_{nh}^{nh+h} v\,dt = y_n + \omega \frac{v_n + v_{n+1}^{(1)}}{2} h$$

$$E_{n+1}^{(2)} = E_n$$

and so

$$P_{n+1}^{(2)} = E_n - 2\zeta v_{n+1}^{(2)} - y_{n+1}^{(2)}$$

Final approximation: (integration)

$$v_{n+1} = v_n + \omega \frac{P_n + P_{n+1}^{(2)}}{2} h$$

$$y_{n+1} = y_n + \omega \frac{v_n + v_{n+1}}{2} h$$

Making the necessary substitutions from the second and first approximations into the final approximation, one obtains the difference equations

$$v_{n+1} = v_n + \omega h \left(P_n - \omega h \zeta P_n - \frac{1}{2} \omega h\, v_n\right)$$

$$y_{n+1} = y_n + \frac{1}{2} \omega h (v_n + v_{n+1})$$

It is important to notice that ω and h always appear as the combination ωh, never separately. Since ω is the natural frequency of oscillation of the circuit, in radians per second, and h is the increment in time between two successive points, in seconds per point, the product ωh is expressed in radians per point.

$\frac{2\pi}{\omega h}$ is then the number of points per period. By increasing the value of ωh one obtains fewer points in each period. This may be thought of as keeping the increment fixed and increasing the frequency or as keeping the frequency fixed and increasing the increment; both have

the same effect and cannot be separated.

Outline of the Program

The program consists of two separate parts, (1) the actual numerical solution of the differential equation for $t = h, 2h, 3h, \ldots, Nh$, and (2) the generation of the function $E(t)$ for $t = h, 2h, 3h, \ldots, Nh$, where N is the total number of points to be displayed. It is impractical to store the N separate values of $E(t)$, so these values must be generated; furthermore, since the numerical solution requires most of the 32 registers of test storage, only a few orders may be used in generating the function. Among the functions which may be generated easily are square waves and sawtooth waves. These possibilities are discussed in detail later.

The Numerical Solution Program

First, let us simply assume that the value of $E(t)$ is generated by some short program, so that on the nth cycle through the program the value of $E(nh)$ is calculated by this short program when it is needed. Assume further that at the end of the nth cycle the new value v_{n+1} is left in AC and that the program starts by storing this value in a FF register. Just before the start of each new cycle, the FF registers (except the one which contains y_n and one which is used by the function-generating program) are reset by the program counter and carry (overflow) to some quantity specified by manually-operated reset switches. The first six registers are assigned as follows:

Reg. #

0 (FF) contains the damping ratio ζ which is changed only by changing the reset switches of this register. The only reason for using a FF register is to permit changing ζ from the test control center rather than at toggle-switch storage.

1 (FF) contains the quantity y_n; reset to y_0 (initial value determined by reset switches) at the beginning of each sweep.

2 (FF)
3 (toggle switch) } reserved for use by the function-generating program.

4 (FF) reset at the beginning of each cycle to the first order of the cycle (ts 4); used during the program to store the quantity v_n.

5 (FF) reset at the beginning of each cycle to the second order of the cycle (mr 0); used during the program to store various quantities temporarily.

Register 4 contains the first order of each cycle and that order transfers the contents of AC, namely v_n, into the same register. The purpose of this is to save one storage register by making the FF register serve double duty. At the end of each cycle the old value of v_{n-1} has been added to the increment to obtain the new value v_n, and, while v_n is in AC, register 4 is not being used for anything else and hence can be reset and used for an order. Once the order has been taken from storage, register 4 is again free and can again be used to store the quantity v_n. A similar use is made of register 5. The initial value of v (i.e. the value assigned to v at the beginning of each sweep) is always zero, since at the start v is assumed to be in AC and AC is cleared (along with most of the rest of the computer) at the start of each sweep. The trick of using the FF registers twice is only possible because of the reset controls peculiar to the FF registers and is then only possible when the contents of the register is immaterial at the beginning of each cycle.

In order to simplify the program, ωh is chosen to be 2^{-3}. Since $\frac{2\pi}{\omega h}$ is the number of points per period, choosing $\omega h = 2^{-3}$ corresponds to choosing to plot $16\pi \simeq 50$ points for each period of the natural frequency of the solution. Actually, minor changes in the program can be made to make $\omega h = 2^{-n}$ for several different values of n, but ωh cannot easily be given values other than a single power of 2. This difficulty in varying ωh (and v_0) is the relatively small price that is paid in addition to the slightly reduced accuracy to gain enough storage to permit introduction of a driving function.

The quantity $\frac{z}{}$ is stored with a scale factor of 2^{-3} so that $\frac{z}{}$ can be given any value less in magnitude than $2^3 = 8$. No scale factor is associated with y, v or the function E(t).

Coded Program for the Numerical Solution, in detail

Order #	Register #	
1 (ts 4)	4	stores the new value of v_n in reg. 4 (at the start of each new sweep of the scope, the program starts here with zero in AC).
2 (mr 0)	5	multiplies v_n by $\frac{1}{3} \times 2^{-3}$
3 (ts 5)	6	stores $\frac{1}{3} v_n \times 2^{-3} = 2\frac{1}{3} v_n \times 2^{-4}$ in reg. 5
4	7	these registers reserved for function-generating program which leaves E_n in AC.
.	.	
.	.	
.	.	
.	.	
11 (su 1)	14	subtracts y_n from E_n
12 (sr 4)	15	multiplies by 2^{-4}, leaving $(E_n - y_n)2^{-4}$ in AC
13 (su 5)	16	subtracts $2\frac{1}{3} v_n \times 2^{-4}$ from $(E_n - y_n)\ 2^{-4}$
14 (ts 5)	17	stores $(E_n - 2\frac{1}{3} v_n - y_n)\ 2^{-4} = P_n \times 2^{-4}$ temporarily in reg. 5.
15 (mr 0)	18	multiplies $P_n \times 2^{-4}$ by $\frac{1}{3} \times 2^{-3}$, leaving $\frac{1}{3} P_n \times 2^{-3} \times 2^{-4} = \omega h \frac{1}{3} P_n \times 2^{-4}$
16 (su 5)	19	subtracts $P_n \times 2^{-4}$ from $\omega h \frac{1}{3} P_n \times 2^{-4}$
17 (ts 5)	20	stores $-(P_n - \omega h \frac{1}{3} P_n)\ 2^{-4} = -\omega h(P_n - \omega h \frac{1}{3} P_n)\ 2^{-1}$ temporarily in reg. 5
18 (cs 4)	21	puts $-v_n$ into AC
19 (sr 8)	22	multiplies by 2^{-8}, leaving $-v_n \times 2^{-8} = -\frac{1}{2}(\omega h)^2 v_n \times 2^{-1}$ in AC.
20 (su 5)	23	subtracts $-\omega h(P_n - \omega h \frac{1}{3} P_n)2^{-1}$ from $-\frac{1}{2}(\omega h)^2 v_n \times 2^{-1}$
21 (ts 5)	24	stores $\omega h(P_n - \omega h \frac{1}{3} P_n - \frac{1}{2}\omega h v_n)\ 2^{-1}$ in reg. 5.
22 (ad 4)	25	adds v_n to $\frac{1}{2}\omega h(P_n - \omega h\ P_n - \frac{1}{2}\omega h v_n)$
23 (sr 3)	26	multiplies $\frac{1}{2}\left\{v_n + \left[v_n + \omega h(P_n - \omega h\ P_n - \frac{1}{2}\omega h v_n)\right]\right\} = \frac{1}{2}(v_n + v_{n+1})$ by 2^{-3}

Order #	Register #	
24 (ad 1)	27	adds y_n to $\frac{1}{2}(v_n + v_{n+1}) \times 2^{-3} = \omega h \frac{v_n + v_{n+1}}{2}$
25 (ad 1)	28	displays and stores $y_n + \omega h \frac{v_n + v_{n+1}}{2} = y_{n+1}$ in reg. 1.
26 (ca 5)	29	puts $\omega h(P_n - \omega h \not{7} P_n - \frac{1}{2} \omega h v_n) 2^{-1}$ in AC
27 (sl 1)	30	multiplies by 2.
28 (ad 4)	31	adds v_n to $\omega h (P_n - \omega h \not{7} P_n - \frac{1}{2} \omega h v_n)$, forming v_{n+1}

(the program counter end carry now resets FF registers 4 and 5 and resets PC to 4, so that the next order is taken from register 4 and the cycle is repeated)

Function-Generating Program

In the Whirlwind I Display photographs (A-35128, attached), are shown three different functions, a square wave, a sawtooth wave, and a step function. All of these functions are easily generated using the following scheme.

Once each cycle, an adjustable constant (c), stored in register 3, is added to the contents of register 2. As a result the contents of register 2 increase as a straight line with a slope determined by the adjustable constant. Then, by shifting the contents left a few digits (k), some of the left end digits are discarded. This results in chopping the straight line up into pieces and making it start over at zero with each new piece. Technically, the result after n cycles is the residue $nc2^k \pmod 1$. In short, the result is a sawtooth wave such as is shown in the photograph.

By singling out the leftmost digit of the sawtooth wave, one can generate a square wave, for the leftmost digit taken alone gives 0 for the first half of the sawtooth and $\frac{1}{2}$ for the second half. The leftmost digit can be singled out by following the sl k order used for the sawtooth by sr 15 and then sl 14. The sr 15 order shifts the number into BR and rounds off, leaving 2^{-15} in AC if the leftmost digit was 1 and 0 in AC otherwise. The sl 14 order then shifts the singled-out digit back into the proper place.

A step function can be made up by (1) generating a square wave of very long period and (2) shifting the time base to the left until the step becomes near the origin. The time base can be shifted easily by resetting FF register 2 to contain some positive number m. This shifts the wave shape to the left because the function-generating program starts out, when actually $t = 0$, with m in register 2 and therefore behaves at the beginning as if $t = \frac{m}{n}$. Similarly, resetting register 2 to contain a negative number shifts the wave shape to the right.

Many variations of these functions can easily be generated. Only the square and saw tooth waves are given in the attached program. It is important to notice that the frequency of the waves is varied by adjusting the constant which is added into register 2 on each cycle. The amplitude can usually be varied only by factors of 2 by changing the final shift order. The od order which is used to actually display the driving function is arranged to transfer into a toggle-switch register. The reason for this is that (1) actually the driving function does not need to be stored and (2) there is no FF register available in which to store it. Transferring into a toggle-switch register generates a transfer check alarm. Thus an alarm is normal for this program and must be suppressed by turning the alarm switches off. By changing the od to sp, the alarm can be averted but the driving function is no longer displayed.

signed C. W. Adams
C. W. Adams

approved R. R. Everett
R. R. Everett

CWA/lfu
attached: Appendix I - Display and Storage
Display Program Number IV
A-35128 - Whirlwind I Display

Appendix I

Display

The present temporary display system consists of a cathode-ray oscilloscope connected to a decoder with a capacity of eight binary digits. The decoder produces a voltage with an amplitude corresponding to the sign and magnitude of a binary number and the voltage is used to provide a vertical deflection for the scope. The horizontal deflection is provided by the sawtooth sweep generator within the scope itself. This horizontal sweep generator controls the computer so that when the sweep starts the computer is started and when the sweep stops the computer is stopped, cleared, reset, and ready to be started again by the next sweep. The FF storage registers may be reset or not depending on the setting of some switches.

Once started, the computer proceeds on its own and whenever a number in the accumulator is to be plotted, a special instruction called cd - display - which is written into the program puts the new value into the decoder (thus setting the vertical deflection) and intensifies the scope beam momentarily, thus plotting a spot on the scope face. Since the time required by the computer to obtain each new value is about constant and since the scope is approximately linear, the horizontal distance between spots on the scope face is nearly constant. The frequency of the horizontal sweep is adjusted so that the length of each sweep corresponds to the time required by the computer for the solution of the problem at hand.

Storage

The storage of the computer (i.e. the capacity of the machine to accept and remember numbers and instructions) is at present restricted to the 32 test storage registers, each 16 binary digits long. Of these 32 storage registers, 5 are composed of flip-flops and are therefore erasable and may be used for storing new information obtained by the computer during automatic operation. The remaining 27 registers are composed of toggle switches which must be individually set by hand and cannot be erased or altered by the computer in automatic operation. All numbers and all instructions are stored in these 32 storage registers. New information is usually introduced by means of toggle switches which control the resetting of the 5 flip-flop registers. These switches do not reset the registers directly; they merely determine the numbers to which the registers will be reset when a reset pulse occurs. A reset pulse can be generated by an end carry (overflow) from the program counter and/or by a pulse (called SD end carry) generated at the end of each sweep of the display oscilloscope.

Display Program Number IV

```
 0 (FF) reset to z
 1 (FF) reset to y0
 2 (FF) reset shifts driving
 3 *    function right or left
 4 (FF) reset to ts 4
 5 (FF) reset to mr 0
 6 ts 5
 7 *
 8 *
 9 *
10 *
11 *
12 *
13 *
14 su 1
15 sr 4   sr 3 for wh = 2^-2
          sr 5 for wh = 2^-4
16 su 5
17 ts 5
18 mr 0
19 su 5
20 ts 5
21 os 4
22 sr 8   sr 6 for wh = 2^-2
          sr 10 for wh = 2^-4
23 su 5
24 ts 5
25 ad 4
26 sr 3   sr 2 for wh = 2^-2
          sr 4 for wh = 2^-4
27 ad 1
28 qd 1
29 ca 5
30 sl 1
31 ad 4
```

* Driving Function #1.

```
 3  2^-12 (this register determines
           frequency of wave)
 7  ca 2
 8  ad 3
 9  ts 2
10  sl 6
11  sr 15
12  sl 14 (or sl 13 if reduced amplitude
           is desired)
13  qd 14 (or sp 14 if driving function
           is not to be displayed)
```

* Driving Function #2

```
 3  2^-12 (this register determines
           frequency of wave)
 7  ca 2
 8  ad 3
 9  ts 2
10  sl 6
11  sr 7
12  sl 6 (or sl 5 if reduced amplitude
           is desired)
13  qd 14 (or sp 14 if driving
           function is not to be displayed)
```

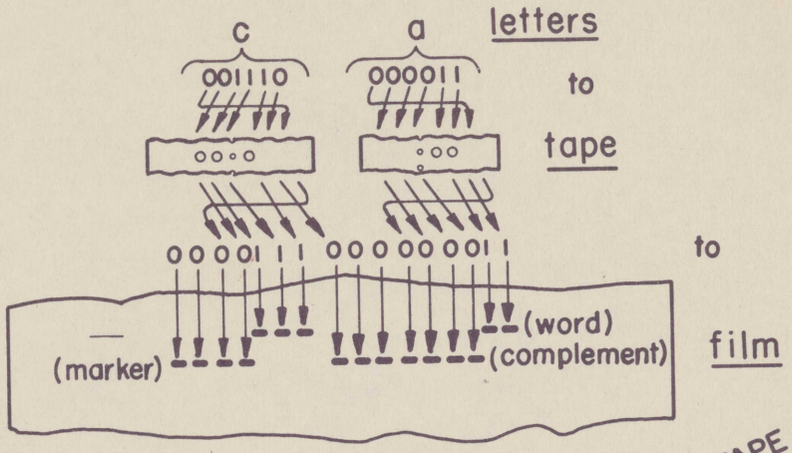
Turn alarm switches off
PC reset to 4
PC must be reset by PC end carry (pulse stand. 8-1a set to give pos. pulse, max. ampl.)
PC must be reset by SD end carry (pulse stand. 8-1b set to give pos. pulse, max. ampl.)
FF registers 0, 4, 5 must be reset by PC end carry (Coder 8-2, channels 1,4,5 on, positive)
FF registers 1, 2 must not be reset by PC end carry (Coder 8-2, channels 2,3 off)
FF registers 0,1,2,4,5 must be reset by SD end carry (Coder 8-3, all channels on, positive)

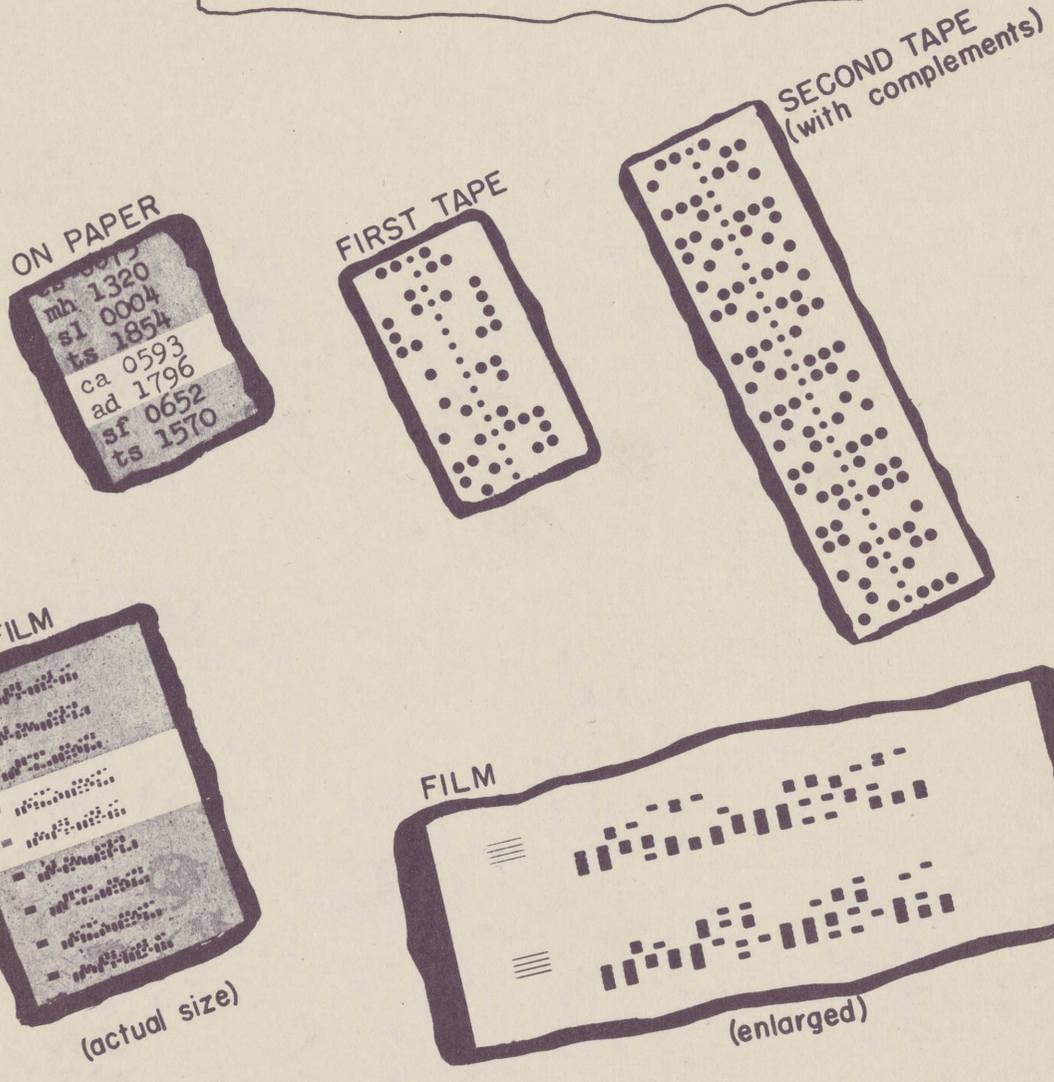
Note 1: Changing from #1 to #2 requires changing only digit 12 of registers 11 and 12.

Note 2: The gd order in register 13 will give a transfer check alarm in normal operation.

Note 3: Program as given is for $\omega h = 2^{-3}$

Note 4: If it is desired to vary the driving function frequency and not the damping ratio ζ, switch the FF-TS switches and cables to make reg. 0 toggle switch and reg. 3 flip flop.





TAPE and FILM for WHIRLWIND I INPUT

A-35681

Author C.W.ADAMS(Ed.)

Shelf No. one 7.311 17

Title R-193 WHIRLWIND Selected descriptive ~~material~~

Edition UNCLASSIFIED

Year 1950

Date Borrowed	SIGN HERE	Returned

LC 33 (OVER) F.E.B. †

7.311
17

UNCLASSIFIED

one copy
unnumbered

SERVOMECHANISMS LABORATORY LIBRARY BUILDING 32, M. I. T.

SIGN THIS CARD AND LEAVE IT upon the desk.

NO BOOK may be taken from the room until it has been REGISTERED in this manner.

RETURN this book to the DESK as soon as you have finished with it.

ANOTHER READER may need it.

LC44-5M-4-50