

ElectroData Division
Burroughs Corporation

THE OPERATIONAL CHARACTERISTICS
OF THE DATATRON 220
ELECTRONIC, DATA-PROCESSING SYSTEM

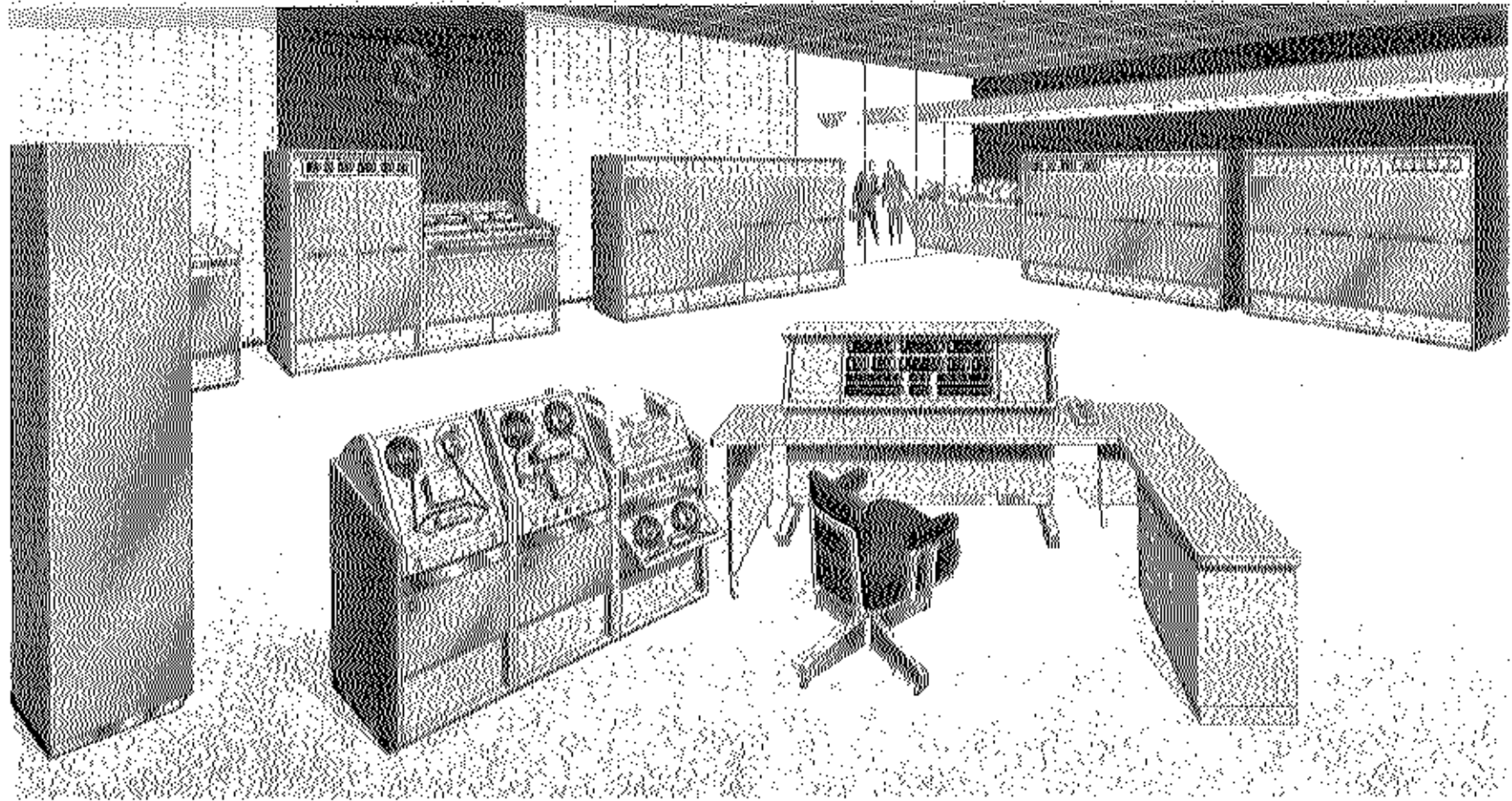
PRELIMINARY EDITION, SECOND REVISION
CUSTOMER EDUCATION GROUP
PUBLICATIONS AND TRAINING

Copyright © 1957
Burroughs Corporation
Printed in U.S.A.

The information contained herein
is subject to change without
notice to the holder of this
volume.

9/1/57

Cardatron DataFile Magnetic DataFile Digital Storage Core Storage Power
 (multiple (multiple Tape (reel) Computer Control (5000 words) Supply
 tape) tape)



Paper Tape Paper Tape Supervisory Control
 Punch Reader Printer Console
 with
 Mechanical
 Reader

THE DATATRON 220 ELECTRONIC DATA PROCESSING SYSTEM

PREFACE

Preface

This volume in the ElectroData Data-Processing Library, the Operational Characteristics of the DATATRON 220, is intended to be the basic reference for the professional programmer. Not only is it written in the language of the programmer, but this volume abounds in the system details without which the professional cannot satisfactorily perform his job: in a sense, this volume is primarily a translation of the flow diagrams, logical equations, and circuit diagrams which are the ultimate source of knowledge concerning the DATATRON 220 System.

The organization of this book is intended to answer the needs of the professional. For example, it is in loose-leaf form, not only to facilitate changes in its contents, but also to allow the book to lie flat on a desk for ease of use. The pagination, also, was designed to permit easy addition to, deletion from, and/or change of the contents. The various sections of the book are set apart by colored index tabs for easier reference. The format of the pages was chosen to permit rapid reading as well as to provide space for exegetic notes which the owner of the volume might deem useful (such notes, it is hoped, would be communicated to the editors).

One other aspect of this book's organization is worth noting: parts of the technical information appear repeatedly in different sections, frequently in exactly the same wording. This repetition and identity of style was deliberate: the function of this technique is to permit the reader to have in front of him, when he needs it, the bulk of important information; the identity of style permits instant recognition of this basic information. Cross references

were reduced in this manner, but not eliminated entirely. Besides, there are certain basic notions which are defined once, and for which the reader must hold himself responsible. In this effort he will be assisted by glossaries of terms and symbols which appear as appendices to this volume.

A final explanatory note is in order: this book is not intended as a "programming manual" in the sense in which that term is ordinarily used. This is not an apology: it is a statement of fact; this book is a manual for programmers. The novice will find in the ElectroData Data-Processing Library other volumes whose purpose it is to introduce him to programming and the art of coding a stored-program digital computer.

On the other hand, the person who comes to the DATATRON 220 with experience in other stored-program computing systems will find in this volume sufficient information to provide him with a comprehensive knowledge of the DATATRON 220 System. This knowledge can be acquired without the formality of classroom attendance (but, of course, actual experience on a machine would assist in its assimilation), especially if this volume is supplemented by the Handbook of Programming and Coding Techniques for the DATATRON 220. Operational Characteristics was designed especially to serve the purposes of the experienced programmer.

A separate Handbook of Operating Procedures for the DATATRON 220 contains information required for actual operation of the system.

It is unlikely that this volume is error-free. The editors will be grateful for communications which describe the errors which remain.

ADVICE TO THE READER

Advice to the Reader

The purpose of this book is to provide a description of the operational characteristics of the DATATRON 220 Electronic Data-Processing System; the book is intended to be a manual for programmers not a programming manual.

If the reader is to take advantage of the contents of the book, he ought to be familiar with its organization. In Section I, Introduction, the reader will find a general description of the System and its characteristics. Section VIII contains the appendices, and therein the reader will find such useful information as summaries of operations, glossaries of terms and symbols, as well as other useful tools of the trade.

Each of the remaining sections of the book is devoted to a description of some component of the system. Each section begins with an exposition of the characteristics of the component and a description of the place of the component in the system.

For example, Section II is concerned with the DATATRON 220 Digital Computer. In the beginning of the section the reader will find a physical description of the Computer; a description of the arithmetic and control units; the representation of information; the B register; the format of instruction words; the operation cycle; information flow; etc.

The operation cycle of the DATATRON 220 is divided into two parts, a Fetch Phase and an Execute Phase. The Fetch Phase is the same for every operation, and is described, with flow charts, once and for all in the beginning of Section II. In order to complete the description of every operation, all that is required are the details of the Execute Phase:

the second part of each section is a detailed description, including abbreviated flow charts, of every Execute Phase.

Typically, the pages corresponding to the Execute Phase of every operation will list the operation name, code, abbreviation, execution time, instruction format (a definition of every field in every instruction word appears also), a description of the operation in summary and flow chart form, a list of exceptional condition, pertinent remarks, and a tabulation of the status of control and arithmetic registers after the operation is complete. If examples are required to make clearer the text, they are included.

It is clear from the foregoing that the introductory material in each section ought to be read before the descriptions of the Execute Phase. In fact, the reader is cautioned that in the writing of the Execute Phase a knowledge of the introductory material was assumed.

The reader is reminded that Appendix 4 is a Glossary of Terms and Appendix 5 is a Glossary of Symbols: he should find both of these useful in helping him to find his way through the book.

TABLE OF CONTENTS

Preface	i
Advice to the Reader.....	iii
Table of Contents.....	v

Section I Introduction

Introduction.....	I-Intro-1
-------------------	-----------

Section II The Computer

Introduction.....	II-Intro-1
Representation of Information.....	II-Intro-1
Word Format.....	II-Intro-3
Numeric Information.....	II-Intro-3
Fixed-Point Numbers.....	II-Intro-3
Floating Point Numbers.....	II-Intro-5
Alphabetic or Alphanumeric Information.....	II-Intro-7
Instruction Words.....	II-Intro-7
Partial-Word Fields.....	II-Intro-9
Registers.....	II-Intro-10
Information Flow.....	II-Intro-13
The Operation Cycle.....	II-Intro-14
The Fetch Phase.....	II-Intro-14
The Execute Phase.....	II-Intro-18
Input Flow.....	II-Intro-19
Output Flow.....	II-Intro-20
Exceptional Conditions.....	II-Intro-22
The Execute Phase.....	II-Intro-25
HALT.....	II-00-2
NO OPERATION.....	II-01-2
CLEAR ADD.....	II-10-2
CLEAR ADD ABSOLUTE.....	II-10-2
CLEAR SUBTRACT.....	II-11-2
CLEAR SUBTRACT ABSOLUTE.....	II-11-2
ADD.....	II-12-2
ADD ABSOLUTE.....	II-12-2
SUBTRACT.....	II-13-2
SUBTRACT ABSOLUTE.....	II-13-2
MULTIPLY.....	II-14-2
DIVIDE.....	II-15-2
ROUND.....	II-16-2
EXTRACT.....	II-17-2
COMPARE FIELD A.....	II-18-2
COMPARE FIELD R.....	II-18-2
ADD TO LOCATION.....	II-19-2
INCREASE B, BRANCH.....	II-20-2
DECREASE B, BRANCH.....	II-21-2
FLOATING ADD.....	II-22-2
FLOATING ADD ABSOLUTE.....	II-22-2
FLOATING SUBTRACT.....	II-23-2
FLOATING SUBTRACT ABSOLUTE.....	II-23-2
FLOATING MULTIPLY.....	II-24-2
FLOATING DIVIDE.....	II-25-2

INCREASE FIELD LOCATION.....	II-26-2
DECREASE FIELD LOCATION.....	II-27-2
DECREASE FIELD LOCATION, LOAD B.....	II-28-2
RECORD TRANSFER.....	II-29-2
BRANCH, UNCONDITIONALLY.....	II-30-2
BRANCH, OVERFLOW.....	II-31-2
BRANCH, REPEAT.....	II-32-2
BRANCH, SIGN A.....	II-33-2
BRANCH, COMPARISON HIGH.....	II-34-2
BRANCH, COMPARISON LOW.....	II-34-2
BRANCH, COMPARISON EQUAL.....	II-35-2
BRANCH, COMPARISON UNEQUAL.....	II-35-2
BRANCH, FIELD A.....	II-36-2
BRANCH, FIELD R.....	II-37-2
STORE A.....	II-40-2
STORE R.....	II-40-2
STORE B.....	II-40-2
LOAD R.....	II-41-2
LOAD B.....	II-42-2
LOAD B COMPLEMENT.....	II-42-2
LOAD SIGN A.....	II-43-2
STORE P.....	II-44-2
CLEAR A.....	II-45-2
CLEAR R.....	II-45-2
CLEAR B.....	II-45-2
CLEAR LOCATION.....	II-46-2
SHIFT RIGHT A.....	II-48-2
SHIFT RIGHT A AND R.....	II-48-2
SHIFT RIGHT A WITH SIGN.....	II-48-2
SHIFT LEFT A.....	II-49-2
SHIFT LEFT A AND R.....	II-49-2
SHIFT LEFT A WITH SIGN.....	II-49-2

Section III The Control Console

Introduction.....	III-Intro-1
PROGRAM CONTROL SWITCHES.....	III-Intro-1
The Keyboard.....	III-Intro-2
The Supervisory Printer.....	III-Intro-2
The Interval Timer.....	III-Intro-3
The Execute Phase.....	
KEYBOARD ADD.....	III-08-2
SUPERVISORY PRINT-OUT.....	III-09-2
BRANCH, CONTROL SWITCH.....	III-38-2

Section IV The Magnetic-Tape System

Introduction.....	IV-Intro-1
The Magnetic-Tape Storage Unit.....	IV-Intro-2
The Datafile.....	IV-Intro-3

TABLE OF CONTENTS

Tape Format.....	IV-Intro-4
The Editing Process.....	IV-Intro-9
Writing on Newly-Edited Tape.....	IV-Intro-12
Search.....	IV-Intro-18
Scan.....	IV-Intro-19
Reading.....	IV-Intro-21
Overwriting.....	IV-Intro-23
Positioning.....	IV-Intro-23
Interrogation.....	IV-Intro-24
Input Sign Control.....	IV-Intro-25
Parity Checking.....	IV-Intro-25
Digit-count/Word-count Checking.....	IV-Intro-28
Timing.....	IV-Intro-28
Information Flow.....	IV-Intro-33
Input Flow.....	IV-Intro-33
Output Flow.....	IV-Intro-36
Use of DATATRON 205 Magnetic Tape.....	IV-Intro-36
Exceptional Conditions.....	IV-Intro-38
The Execute Phase.....	IV-Intro-39
MAGNETIC-TAPE SEARCH.....	IV-50-2
MAGNETIC-TAPE FIELD SEARCH.....	IV-50-2
MAGNETIC-TAPE LANE SELECT.....	IV-50-2
MAGNETIC-TAPE REWIND.....	IV-50-2
MAGNETIC-TAPE REWIND, DE-ACTIVATE.....	IV-50-2
MAGNETIC-TAPE SCAN.....	IV-51-2
MAGNETIC-TAPE FIELD SCAN.....	IV-51-2
MAGNETIC-TAPE READ.....	IV-52-2
MAGNETIC-TAPE READ, RECORD.....	IV-53-2
MAGNETIC-TAPE INITIAL WRITE.....	IV-54-2
MAGNETIC-TAPE INITIAL WRITE, RECORD.....	IV-55-2
MAGNETIC-TAPE OVERWRITE.....	IV-56-2
MAGNETIC-TAPE OVERWRITE, RECORD.....	IV-57-2
MAGNETIC-TAPE POSITION FORWARD.....	IV-58-2
MAGNETIC-TAPE POSITION BACKWARD.....	IV-58-2
MAGNETIC-TAPE POSITION AT END OF INFORMATION.....	IV-58-2
MAGNETIC-TAPE INTERROGATE, BRANCH.....	IV-59-2
MAGNETIC-TAPE INTERROGATE END OF TAPE, BRANCH.....	IV-59-2

Section V The Paper-Tape System

Introduction.....	V-Intro-1
The Photo-Electric Reader.....	V-Intro-1
Word Format.....	V-Intro-3
Instructions.....	V-Intro-4
The Paper-Tape Punch.....	V-Intro-6
The Printer.....	V-Intro-7
The Mechanical Reader.....	V-Intro-7
Information Flow.....	V-Intro-7
Input Flow.....	V-Intro-7
Output Flow.....	V-Intro-10
Exceptional Conditions.....	V-Intro-10
The Execute Phase.....	
PAPER-TAPE READ.....	V-03-2
PAPER-TAPE READ, BRANCH.....	V-04-2

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

PAPER-TAPE READ, INVERSE FORMAT..... V-05-2
PAPER TAPE WRITE V-06-2
PAPER TAPE WRITE INTERROGATE, BRANCH..... V-07-2

Section VI The Cardatron

Introduction..... VI-Intro-1
Input
The Input Unit..... VI-Intro-2
Information Transfer..... VI-Intro-3
 Part 1: From Card to Buffer..... VI-Intro-4
 Part 2: From Buffer to Data Processor..... VI-Intro-7
Designation of Format Bands..... VI-Intro-7
Format Band Selection..... VI-Intro-9
Editing Control Stream Digits..... VI-Intro-14
The Construction of Editing Control Streams
 for Input..... VI-Intro-15
Instructions..... VI-Intro-27
Output
The Output Unit..... VI-Intro-29
Information Transfer..... VI-Intro-30
 Part 1: From Data Processor to Buffer..... VI-Intro-30
 Part 2: From Buffer to Card-Handling Machine VI-Intro-31
Designation of Format Bands..... VI-Intro-35
Format Band Selection..... VI-Intro-35
Editing Control Stream Digits..... VI-Intro-36
The Construction of Editing Control Streams
 for Output..... VI-Intro-40
Instructions..... VI-Intro-52
Information Flow..... VI-Intro-53
 Input Flow..... VI-Intro-54
 Output Flow..... VI-Intro-56
Exceptional Conditions..... VI-Intro-59
The Execute Phase..... VI-Intro-60
 CARD READ..... VI-60-2
 CARD WRITE..... VI-61-2
 CARD READ, FORMAT LOAD..... VI-62-2
 CARD WRITE, FORMAT LOAD..... VI-63-2
 CARD READ INTERROGATE, BRANCH..... VI-64-2
 CARD WRITE INTERROGATE, BRANCH..... VI-65-2

Section VIII Appendices

Appendix 1. A Summary of Operations in Operation-Code
 Order..... A1-2
Appendix 2. A Summary of Operations in Alphabetic Order.. A2-2
Appendix 3. Alphanumeric Codes and Their Representation.. A3-1
Appendix 4. A Glossary of Terms..... A4-1
Appendix 5. A Glossary of Symbols..... A5-1
Appendix 6. A Summary of Execution Times..... A6-1

INTRODUCTION

The DATATRON 220 is a general-purpose, stored-program, sequentially-controlled, automatic, decimal, digital, computer system. The system, organized on a building-block basis, permits the inclusion of the following units, in addition to the DATATRON 220 Digital Computer:

1. The Control Console, which is actually a necessary adjunct to the Computer, is equipped with operating controls and indicators for the entire system. Associated with the Control Console is a decimal keyboard for (supervisory) input and a character-at-a-time printer for (supervisory) output.
2. Magnetic-core working-storage, available in increments of 1,000 words, ranges from 2,000 to 10,000 44-bit words. The 8,4,2,1 binary code is used to code the ten-decimal-digit-plus-sign-digit-position word. Each 5,000 words of storage is housed in its own package.
3. Auxiliary storage is provided by magnetic-tape facilities. Up to ten Magnetic-Tape Storage Units or Datafiles, in any combination, may be included in the DATATRON 220 System. Each Magnetic-Tape Storage Unit can store from 875,000 to 1,376,000 words; each Datafile has capacity for from 3,125,000 to 4,880,000 words. A block stored on magnetic tape may vary in length from ten to 100 words.
4. As many as ten photo-electric paper-tape readers may be included in the DATATRON 220 System.

5. As many as ten paper-tape punches may be included in the System. For any of these paper-tape output stations, a (supervisory) printer may be substituted.

6. The Cardatron provides facilities for on-line punched-card input and output and line-at-a-time printed output. As many as seven input-output devices may be attached to the Cardatron, thus becoming a part of the DATATRON 220 System. The unique buffering system of the Cardatron permits simultaneous operation of its several input-output devices independently of Computer control. Automatic editing and translation of alphanumeric information to and from the DATATRON code is provided by the Cardatron.

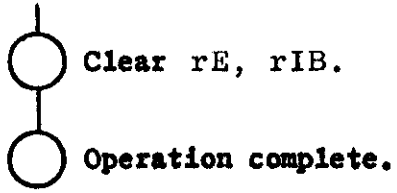
7. With the DataPrinter, the DATATRON 220 System has high-speed, line-at-a-time printing facilities. The tape-driven DataPrinter may be used on line -- in which case as many as ten of them may be included in the System -- or off line -- in which case an arbitrary number of DataPrinters can be used. The DataPrinter has its own buffer storage and comprehensive plugboard editing facilities.

Description of operation:

Summary:

Perform no operation: at the time the operation code is sensed at the beginning of the Execute Phase, the execution is complete; the computer proceeds to fetch the next instruction.

Flow chart:



Exceptional conditions: NONE.

Remarks:

1. Although the address field is not relevant to the execution of this instruction, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.

Register status:

Register name	Contents after execution of NOP.												
A	Unchanged												
R	"												
D	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> </tr> </table>	+	i	i	i	i	0	1	i	i	i	i	
+	i	i	i	i	0	1	i	i	i	i			
B	Unchanged												
P	$(rP)_b + 1$												
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">]</td> </tr> </table>	i	i	i	i	0	1	B	[i	i	i]
i	i	i	i	0	1	B	[i	i	i]		
E	Cleared												

Introduction

The DATATRON 220 is a general-purpose, stored-program, sequentially-controlled, series-parallel, automatic, electronic, digital computer which employs a single-address order code. The basic order code of the DATATRON 220 Digital Computer admits of 39 orders, but variations in the structure bring the number of distinct operations to 61 (this does not include orders referring to input-output equipment, which will be enumerated in the sections in which the peripheral equipment is described). Control and arithmetic units as well as power supplies and working storage are housed separately.

The control console is regarded as being an integral part of the computer; ~~although its details~~^{however,} will be discussed in a separate section (Section III).

Depending on the size of magnetic-core working-storage in the system, one or two units are required to house that storage.

Representation of Information

The basic unit of information in the DATATRON 220 is the 8, 4, 2, 1-binary-coded decimal digit. An aggregate of eleven digit positions is called a word. More accurately, the word should be described as ten digits plus sign, since it is not intended that the sign-digit position in a word shall be used in the same unrestricted fashion as the other ten digits. The digit positions in a word are identified as shown in Figure II-Intro-1.

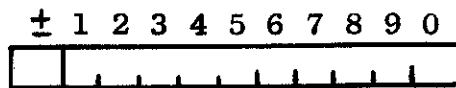


Figure II-Intro-1. Digit-Position Identification
in the Computer Word

The logical structure of the DATATRON 220 is based on the fixed-length ten-digit-plus-sign word described above. Certain operations, however, permit manipulation of partial-word fields,¹ in a manner to be described below.

Depending on the manner in which the word is referred to, its contents may be regarded as numeric, alphabetic or mixed numeric and alphabetic (otherwise, alphanumeric), or an instruction. For example, 0 5941 52 4149 may represent the number + 5941524149, the noun RAJAH, ~~the part-number RA2AH~~, or an instruction which causes information to be transferred from magnetic tape to computer storage.

Each word in magnetic-core storage is identified by a unique four-decimal-digit number called the address of the word. The word itself is said to be stored in the location whose address identifies it.

Although the DATATRON 220 uses only the ten decimal digits, each decade of four bits in the 8, 4, 2, 1 code might have been used to count to 15. The occurrence - for any reason - in the low-order digit-position

¹ A partial-word field is any set of contiguous digit-positions within a computer word; for example, digit positions 5, 6, and 7; or ±, 1, 2, 3, and 4; or ±.

of certain of the control registers of a configuration corresponding to any one of the decimal numbers from ten to 15, is detected automatically. The detection of such a configuration results in a digit-check ALARM STOP:¹ the computer stops and an ALARM indicator light - on the control console - labelled DIGIT CHECK comes on.

Word Format

1. Numeric information

a. Fixed-point numbers

Each word of ten digit-positions plus sign-digit position may represent a number in the range -9999 99 9999 to +9999 99 9999. It is conventional to regard the so-called machine decimal-point as being located between the sign and the high-order numeric digit; hence, it is customary to assert that fixed-point machine-numbers are restricted to the range $-1 < n < +1$.

If the result of an arithmetic operation yields a number outside the range specified above, arithmetic overflow is said to have occurred. The occurrence of arithmetic overflow causes the OVERFLOW Indicator to be turned on. The status of the OVERFLOW Indicator may be interrogated

¹ Optionally, an audible alarm signal may be caused to sound whenever the computer stops on detecting an exceptional condition. All exceptional conditions will be described in context below.

by the BRANCH, OVERFLOW instruction.

The OVERFLOW Indicator must be interrogated by the instruction immediately following the instruction which turns it on. If the OVERFLOW Indicator is not turned off an overflow ALARM STOP will occur: the computer stops with the OVERFLOW Indicator light "on."

Whenever it is necessary to distinguish the machine decimal-point from a problem decimal-point, the symbol \wedge will be used to specify the location of the machine decimal-point. A dot - the conventional symbol - will represent the problem decimal-point.

The convention for representing signs is the following: "0" in the sign-digit position means the number is positive (or, at least, non-negative, i.e., greater than or equal to zero); "1" in the sign-digit position means the number is negative (or, at least, non-positive, i.e., less than or equal to zero). An immediate consequence of this convention is that zero may be signed either positive or negative.¹

¹ When an arithmetic operation yields a zero result the sign of the result can be predicted. See the Remarks section as well as the flow chart in the description of the appropriate operation.

Normally, the sign-digit position of a word which is numeric, and on which arithmetic operations are performed, is not different from 0 or 1. If arithmetic operands do have sign digits which differ from 0 or 1, the three high-order bits (8, 4, and 2) in the sign digit will be set to zero before arithmetic begins; in the result word, the three high-order bits will be zero.

b. Floating-point numbers

Each word of ten digit-positions plus sign-digit position may represent a number in exponential or scientific notation; we call such a number a floating-point number. Digit-positions 1 and 2 hold the coded exponent - including its sign. A floating-point number with exponent greater than ^{or equal to} -50 and less than 0 ^{or equal to} is coded in the range 00 to 50, that is, the actual exponent is increased by 50; a floating-point number with exponent greater than 0 but less than 50 is coded in the range 51 to 99: again, the actual exponent is increased by 50.

Digit positions 3 through 0 hold the so-called mantissa of the floating-point number; the sign of the mantissa is the sign of the computer word. The convention for representing signs of mantissas - and, hence, of floating-point numbers - is the same as the convention for fixed-point numbers. The decimal point of a floating-point number is regarded as being located between digit

positions 2 and 3 of the floating-point word, as a result of which it is customary to assert that floating-point machine-numbers, n , are restricted to the range $10^{-50} < n < 10^{+49}$.

Usually, floating-point numbers appear in normalized form; that is, the exponent and mantissa of a number are adjusted so that the mantissa contains no high-order zeros. Of course, in adjusting the mantissa, it may happen that the exponent will fall outside the permissible range, 00 through 99:

- i. If the result of an arithmetic operation causes a floating-point exponent to exceed 99, (exponent) overflow is said to have occurred, and the OVERFLOW Indicator is set "on."
- ii. If the result of an arithmetic operation causes a floating-point exponent to be smaller than 00, (exponent) underflow is said to have occurred, and the arithmetic register(s) which was (were) to contain the result is (are) cleared.

The result of every floating-point arithmetic operation is normalized.

The following examples illustrate floating-point representation:

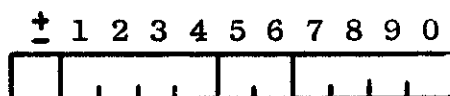
<u>Number</u>	<u>Floating-point Representation</u>
+ 737.0 69 0000	+ 5373 70 6900
+ .0004 88 7900	+ 4748 87 9000
- 2344 9.1 8672	- 5523 44 9186
-.8000 00 0236	- 5080 00 0002

2. Alphabetic or alphanumeric information

An alphabetic or alphanumeric character is represented by two numeric digits in the DATATRON 220 code. The complete DATATRON 220 code is shown in Appendix A3. Each word, therefore, has a capacity for five alphanumeric characters; the sign-digit position of the word is used for a "flag" which indicates that the contents of the word are coded alphanumerically. The flag used is the digit 2.

3. Instruction words

A DATATRON 220 instruction-word may be regarded as composed of four parts, called the address, operation code, control, and sign-digit parts, which are illustrated below.



a. Address part

The four low-order digit-positions in the instruction word are called the address part of the instruction because, in many instructions, they name the address of

a location whose contents are used during the execution of the instruction; otherwise, the address part of the word is used to specify some quantity which is not an address; or the address part may not be relevant to the execution of the instruction.

Any part of an instruction word which is irrelevant to its execution may be coded in an arbitrary fashion.

b. Operation-code part

Digit-positions 5 and 6 in an instruction word are used to specify the numeric operation code of the instruction to be executed.

c. Control part

Digit-positions 1 through 4 are used for a variety of purposes, among which may be listed the specification of partial-word boundaries, the designation of input and/or output units, the enumeration of tallies, etc., each of which specifies some control over the manner in which the instruction will be executed.

d. Sign-digit part

The sign-digit position of an instruction is used to specify whether the address part of the instruction is to be modified - in a manner which will be specified below - by the contents of the B Register as the instruction is brought from storage to the control register: if

the sign digit is an odd integer, B-register address-modification will occur; if the sign digit is an even integer, it will not.

On input from paper tape or punched cards the sign-digit position of an instruction may contain a flag which directs the instruction to the control register and disconnects the input device from the computer.

Additionally, the four-bit of the sign digit is used to specify that a partial-word key is desired and, hence, that MAGNETIC-TAPE FIELD SEARCH or MAGNETIC-TAPE ^{FIELD SCAN} ~~CATEGORY FIELD SEARCH~~ is to be executed.

Partial-word fields

Several instructions allow operations to be performed on partial-word fields. It is necessary to identify the low-order digit-position (i.e., the beginning) and the number of digits (i.e., the length) of the partial-word field. Digit-positions 1 and 2 of the instruction word are used for this purpose: digit-position 1 specifies the location of the low-order digit-position; digit-position 2 specifies the number of digits in the partial-word field. Thus, for example, if these two digit positions contain 04, the four low-order digit-positions of the referenced word are singled out; if these two digit-positions contain 62, digit-positions 5 and 6 are singled out of the referenced word.

The letter "s" is used to symbolize the low-order digit-position, the letter "L" the length, of a partial-word field. In order to describe completely the location of a partial-word field, we also require the address of the word in which the partial-word field is located. The complete address of a partial-word field is symbolized thus: aaaa:sL. In accordance with the usual convention (aaaa:sL) specifies the contents of the partial-word field. Thus, for example, (1000:04) is the address part of the word in location 1000; and (1426:00) specifies all but the sign digit of the word in location 1426.

Registers

In the control and arithmetic sections of the computer are several registers of interest to the programmer. The specific role of each of these registers is described in detail as each operation is described below.

1. The A register is a ten-digit-plus-sign-digit-position register. Its primary function is to store one of the operands as well as the result of an arithmetic operation, although it serves other purposes as well. It is frequently called the accumulator. The A register will be designated as rA, for short.
2. The R register is a ten-digit-plus-sign-digit-position register. It is primarily an extension of the A register. It will be designated as rR.

3. The D register is a ten-digit-plus-sign-digit-position register. The D register is primarily an intermediate buffer used during the transfer of information; in particular, the D register buffers all input to the computer. It will be designated as rD.
4. The B register is a four-digit-position register. Its primary purpose is to provide for automatic address-modification in a manner which will be specified below; it is also used for tallying. The B register will be designated as rB.
5. The C register is ten digits long. It is used to contain ~~the instruction being executed,~~ It is convenient to regard the C register as being divided into three parts:
 - a. The four high-order digit-positions (1, 2, 3, and 4) of the C register contain what are called control digits.
 - b. Digit-positions 5 and 6 always contain the operation code.
 - c. The four low-order digit-positions (7, 8, 9, and 0) of the C register contain the address part of the instruction.

The C register is frequently called the control register. It will be designated as rC.

6. The P register is a four-digit-position register. The P register controls the sequential operation of the computer: it contains the address of the location from which the next instruction will be selected for execution. It is frequently called the program register. The P register will be designated as rP.
Four other registers, not in the control and arithmetic units, are worthy of note.
7. The IB register is a ten-digit-plus-sign-digit-position register in the storage control unit. It is used as a buffer between core storage and the control and arithmetic units. The IB register is frequently called the information buffer register. It will be designated as rIB.
8. The E register is a four-digit-position register in the storage control unit. It is used for control purposes: the E register will always contain the address of a location to which access is being made under computer or manual control. This register will be designated as rE.
9. The CD register is a ten-digit^{plus-sign}-position register in Cardatron control unit 2. It is used for control purposes: the CD register will always contain a copy of the Cardatron instruction which is being executed. The CD register will be designated as rCD.

10. The T register is a ten-digit-position register in the magnetic-tape control unit. It is used for control purposes while magnetic-tape instructions are being executed. The T register will be designated as rT.

The contents of the A, R, D, B, P, C, and E registers are displayed in 8, 4, 2, 1-binary-coded form on the control console with facilities for changing the contents of any one of them. This procedure is described in The Handbook of Operating Procedures for the DATATRON 220.

Information flow

The flow of information between core storage and the registers, between registers, between input-output equipment and the registers is parallel-serial. For example, all four bits of every digit are transferred in parallel; all 44 bits of a word are transferred between the information buffer register and core storage in parallel; but information is transferred between the information buffer register and the D register serially by digit. In the flow charts which follow this will be noted in detail.

There are ^{three}~~four~~ different types of information flow:

1. Operation cycle
 - a. Fetch Phase
 - b. Execute Phase
2. Input
3. Output

1. The operation cycle

The cycle of computer operation is divided into two parts, the first of which is called the Fetch Phase, the second the Execute Phase. Each of these two names is descriptive of computer operation for the duration of that part of the operation cycle: instructions are brought to the control unit during the Fetch Phase; they are executed during the Execute Phase. In normal operation the Fetch and Execute Phases follow each other alternately.

a. The Fetch Phase

Flow charts for the Fetch Phase are shown in Figures II-Intro-2 and II-Intro-3. A verbal description of this flow follows immediately:

1. At the beginning of the Fetch Phase, the contents of the P register are transferred in parallel to the E register.
2. The contents of the location whose address is in the E register are transferred in parallel to the information buffer register. In this register the sign-digit position of the word is examined to determine whether B-register address-modification is intended.

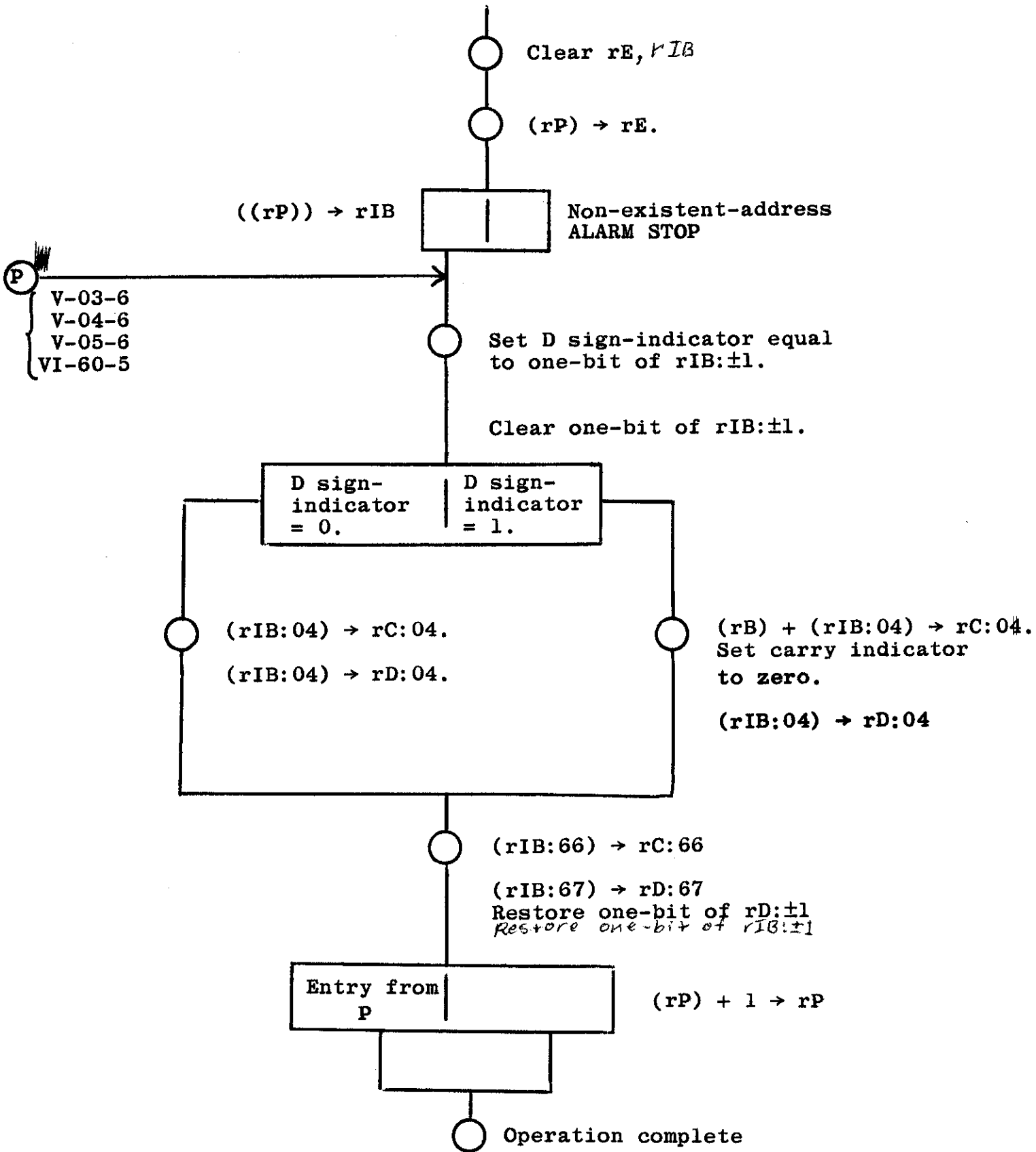


Figure II-Intro-2. The Fetch Phase

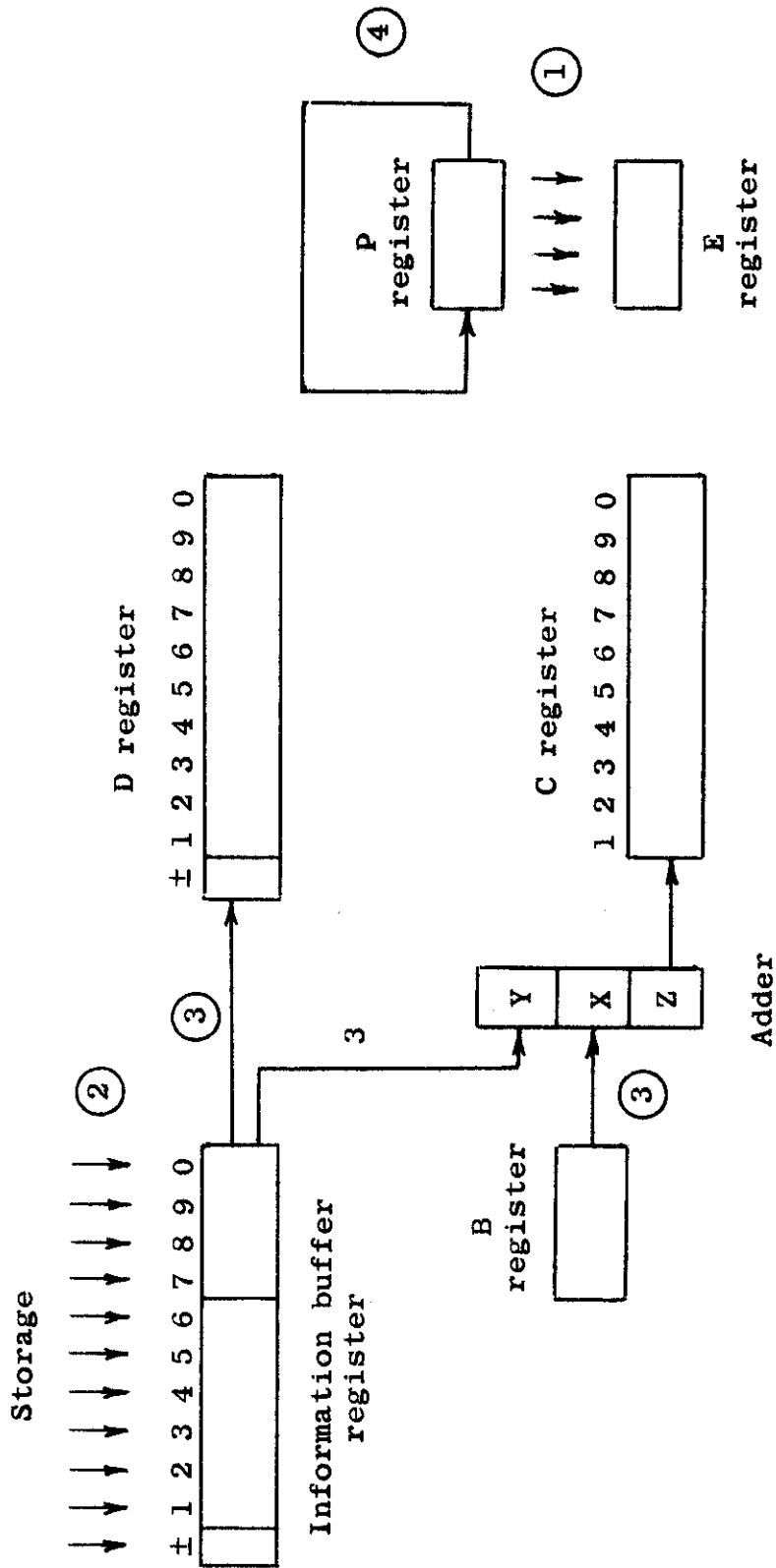


Figure II-Intro-3. The Fetch Phase

If the one-bit of the sign digit is equal to 1 (in which case we may write $(rIB:\pm 1)/1=1$), it is set to zero.

3. If the sign digit in rIB was an odd integer, B-register address-modification will occur as the ten low-order digits of the instruction word pass serially through the adder and into the C register. This transfer of information takes place in two parts: the address part of the word goes first; immediately after this transfer is completed the carry indicator in the adder is set to zero, so that, in case a five-digit sum was generated, there will be no overflow into the operation-code part of the instruction word. The second part of this transfer takes (rIB:66) into the C register.

If the sign digit in rIB was an even integer, no B-register address-modification will occur as the instruction is transferred to the C register.

Simultaneously with the transfer to the C register, the instruction word, including the (possibly-modified) sign digit, is transferred to the D register, but without address

modification, even if it is specified. When the transfer to the D register has been completed the one-bit of the sign digit will be restored in the sign-digit position in the D register. The word in the D register is, therefore, an exact copy of the instruction as it is represented in storage; it is used for checking purposes.

4. Finally, the contents of the P register are counted up 1, unless the entry to the Fetch Phase was made as a result of interrupting a paper-tape or punched-card input instruction, in which case the counting up is inhibited. At the conclusion of the Fetch Phase the P register will contain the address of the location from which will come the next instruction selected for execution, if control continues in sequence.

The time for execution of the Fetch Phase is uniformly 90 microseconds.

b. The Execute Phase

During the Execute Phase, the instruction in the C register is executed. The nature of the Execute Phase and the time required to complete it are functions of the operation code, in particular, as well

as the other digits in the C register. The detailed description of the Execute Phase for each operation comprises the bulk of this volume (see, for example, II-00-2).

2. Input flow

Input information pulses to the computer may be received from a photo-electric reader (paper tape), the Cardatron (punched cards), the keyboard (manual)^{Magnetic-Tape Storage Unit,}, or Datafile (magnetic tape). Figure II-Intro-4 shows the information flow.

1. The address part of the C register is transferred in parallel to the E register to provide the address of the location in which will be stored the information which is destined for storage. The manner in which the contents of the E register are counted up for successive input words varies with the instruction being executed: on the one hand, (rC:04) may be counted up for each input word, after which it is transferred to rE; on the other hand, (rE) may be counted up. (See the flow chart in the description of the appropriate Execute Phase for details.)
2. The information is sent first to the D register where it is assembled as a word, digit by digit.

3. After each word is assembled it is transferred serially to the information buffer register (except in the case of manual cycles (see Handbook of Operating Procedures for details) and KEYBOARD ADD (which see, pages III-08-2, ff.)).
 - a. If the word is to go to storage, B-register address-modification will occur if it is specified.
 - b. If the word is to go to the C register, B-register address-modification is postponed until the entry to the Fetch Phase is made.
4.
 - a. If the word is to go to storage, the contents of the information buffer register are transferred, in parallel, to the location whose address is in the E register.
 - b. If the word is to go to the C register, where it will be interpreted as a new instruction, entry to the Fetch Phase is made at connector P . (See Figure II-Intro-2.)

3. Output flow

Figure II-Intro-5 shows the output flow.

1. The address part of the C register is transferred in parallel to the E register to provide the address of

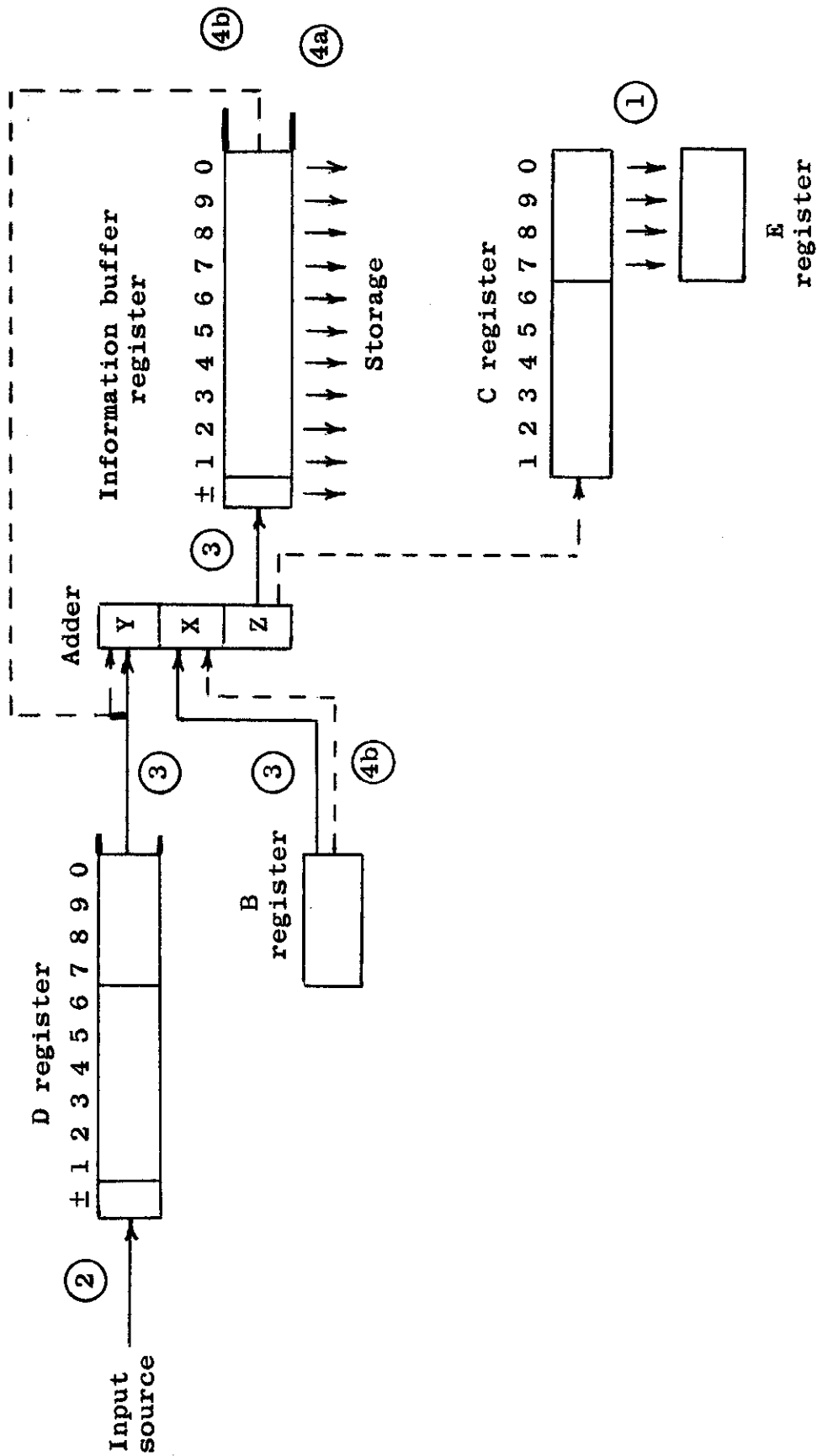


Figure II-Intro-4. Input flow

the location from which will be taken the first word destined for output. The manner in which the contents of the E register are counted up for successive output words varies with the instruction being executed: on the one hand (rC:04) may be counted up for each output word, after which it is transferred to rE; on the other hand, (rE) may be counted up. (See the flow chart in the description of the appropriate Execute Phase for details.)

2. The information is first transferred in parallel from storage to the information buffer register.
3. The contents of the information buffer register are then transferred serially to the D register.
4. The contents of the D register then pass serially through the adder to the output device.

Exceptional conditions

1. The OVERFLOW Indicator may be turned on during the execution of several instructions (a list of them may be found on page II-31-3). The instruction which turns the OVERFLOW Indicator on must be followed immediately by a BRANCH, OVERFLOW instruction which will turn it off. If the OVERFLOW Indicator is not turned off, an overflow ALARM STOP will occur.

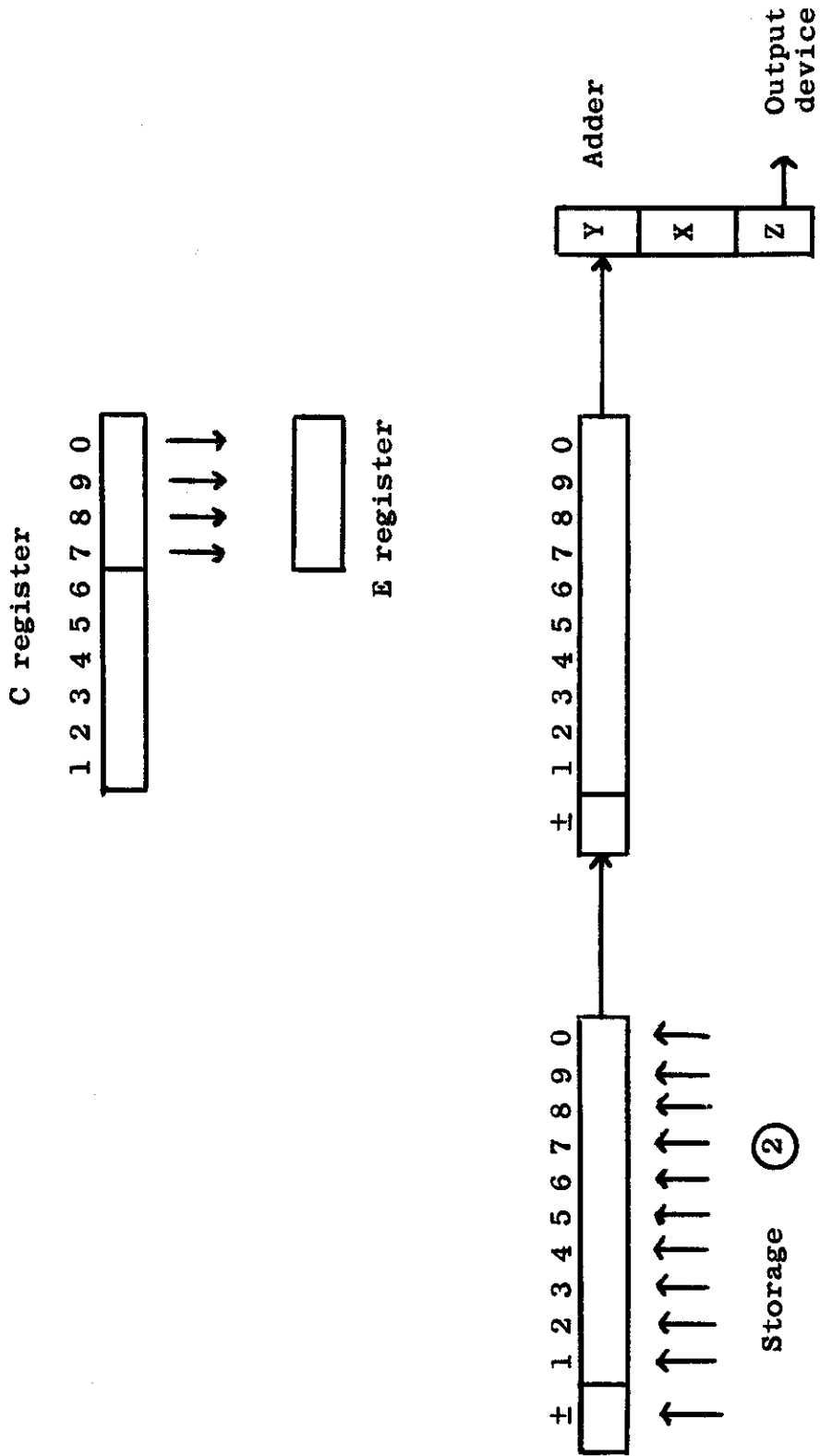


Figure II-Intro-5. Output flow.

2. In those operations which can manipulate partial-word fields, one must have specified sL , the partial-word boundaries. If $s > L+1$, $s \neq 0$, the partial-word will extend beyond the sign-digit position of the word. In this case field overflow is said to have occurred. This condition is detected by the computer and results in a field-overflow ALARM STOP.
3. If the low-order digit position of the IB, A, R, D, or B register or either input to the adder contains a configuration equivalent to one of the decimal numbers 10 through 15, it will be sensed as an error. A digit-check ALARM STOP will occur.
4. If an attempt is made to have access to a location not in the storage package in the system, a nonexistent-address ALARM STOP will occur. For example, suppose the system has 3000 words of storage. An attempt to have access to location 4500 will produce the ALARM STOP.
5. If a nonexistent operation code is sensed in the operation-code part of the C register, a nonexistent-operation-code ALARM STOP will occur.
6. If the COMPARISON Indicator is off when it is interrogated (by a BRANCH COMPARISON instruction; see pages II-35-2, ff.), a no-comparison ALARM STOP will occur.

The Execute Phase

The remainder of Section II is devoted to descriptions of the Execute Phase of each operation with which the computer is concerned exclusively.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CLEAR, ADD

CLEAR, ADD ABSOLUTE

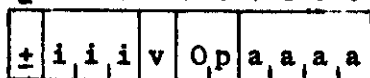
Operation code: 10

Abbreviation: CAD

CAA

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Time (μs):

fetch: 90

execute: 95

total: 185

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: CLEAR, ADD will be executed.

v = 1: CLEAR, ADD ABSOLUTE will be executed.

Op: operation code.

aaaa: address of base of location of augend.

Description of operation:

Summary:

v = 0: (B [aaaa]) replace (rA).

v = 1: |(B [aaaa])| replaces (rA).

Flow chart:

See page II-10-4.

Exceptional conditions:

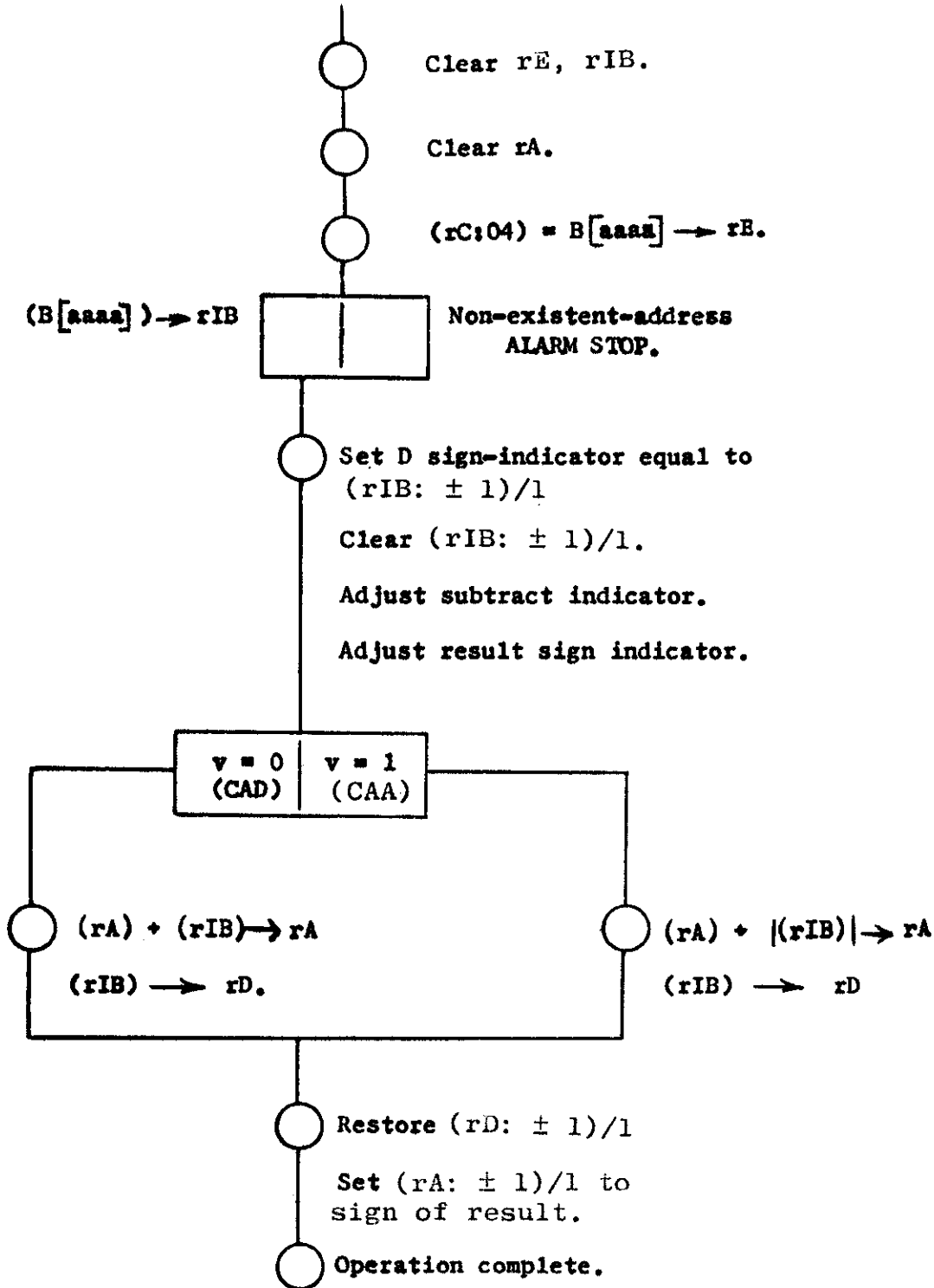
1. Non-existent-address ALARM STOP.

Remarks:

1. The CLEAR, ADD variation will be selected for execution if $v \neq 1$.

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of CAD	Contents after execution of CAA														
A	(B [aaaa])	(B [aaaa])														
R	Unchanged	Unchanged														
D	(B [aaaa])	(B [aaaa])														
B	Unchanged	Unchanged														
P	(rP) _b + 1	(rP) _b + 1														
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>0</td><td>1</td><td>0</td><td>B [aaaa]</td></tr></table>	i	i	i	0	1	0	B [aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>1</td><td>1</td><td>0</td><td>B [aaaa]</td></tr></table>	i	i	i	1	1	0	B [aaaa]
i	i	i	0	1	0	B [aaaa]										
i	i	i	1	1	0	B [aaaa]										
E	B [aaaa]	B [aaaa]														

Register name	Contents if non-existent-address ALARM STOP occurs.											
A	Cleared											
R	Unchanged											
D	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>+</td><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>0</td><td>a</td><td>a</td><td>a</td><td>a</td></tr></table>	+	i	i	i	v	1	0	a	a	a	a
+	i	i	i	v	1	0	a	a	a	a		
B	Unchanged											
P	(rP) _b + 1											
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>0</td><td>B [aaaa]</td></tr></table>	i	i	i	v	1	0	B [aaaa]				
i	i	i	v	1	0	B [aaaa]						
E	B [aaaa]											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 11

Operation name: CLEAR, SUBTRACT

Abbreviation: CSU

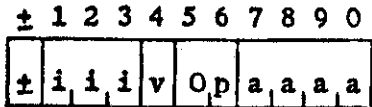
CLEAR, SUBTRACT ABSOLUTE

CSA

Time (μs):

Instruction format:

fetch: 90
 execute: 95
 total: 185



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: CLEAR, SUBTRACT will be executed.

v = 1: CLEAR, SUBTRACT ABSOLUTE will be executed.

Op: operation code.

aaaa: address of base of location of subtrahend.

Description of operation:

Summary:

v = 0: - (B[aaaa]) replaces (rA).

v = 1: - |(B[aaaa])| replaces (rA).

Flow chart:

See page II-11-4.

Exceptional conditions:

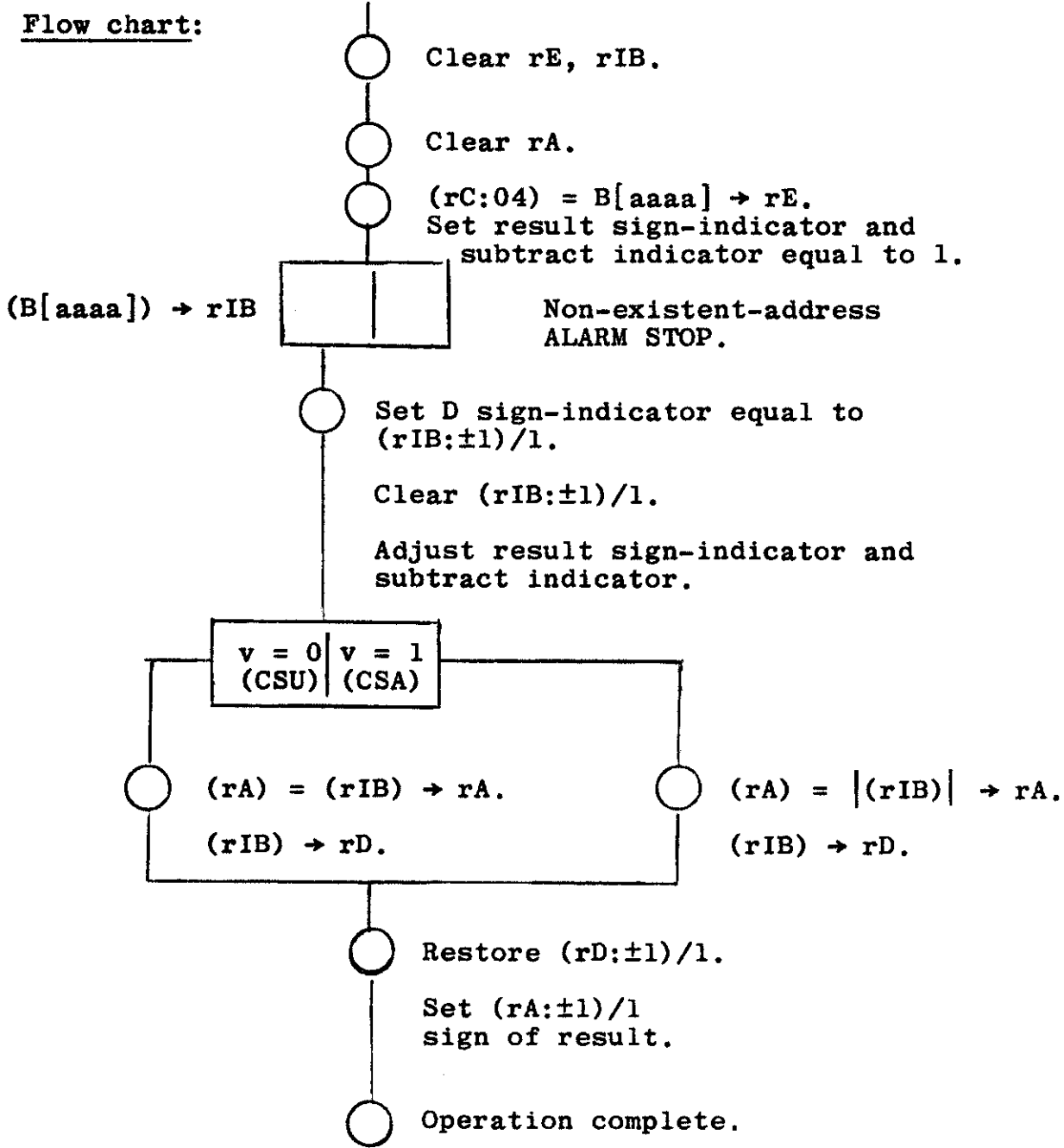
1. Non-existent-address ALARM STOP.

Remarks:

1. The CLEAR SUBTRACT variation will be executed if v ≠ 1.

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of CSU.	Contents after execution of CSA.																
A	$-(B[aaaa])$	$-(B[aaaa])$																
R	Unchanged	Unchanged																
D	$(B[aaaa])$	$(B[aaaa])$																
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>0</td><td>1</td><td>1</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	0	1	1	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>1</td><td>1</td><td>1</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	1	1	1	B	[aaaa]
i	i	i	0	1	1	B	[aaaa]											
i	i	i	1	1	1	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents if non-existent-address ALARM STOP occurs.											
A	Cleared											
R	Unchanged											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>1</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	+	i	i	i	v	1	1	a	a	a	a
+	i	i	i	v	1	1	a	a	a	a		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>1</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	v	1	1	B	[aaaa]			
i	i	i	v	1	1	B	[aaaa]					
E	$B[aaaa]$											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: ADD

ADD ABSOLUTE

Operation code: 12

Abbreviation: ADD

ADA

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: ADD will be executed.

v = 1: ADD ABSOLUTE will be executed.

Op: operation code

aaaa: address of base of location of augend.

Time (μs):

Unnecessary to de-complement sum:

fetch: 90
 executes: 95
 total: 185

Necessary to de-complement sum:

fetch: 90
 executes: 155
 total: 245

Description of operation:Summary:

v = 0: $(rA) + (B[aaaa]) \rightarrow rA.$

v = 1: $(rA) + |(B[aaaa])| \rightarrow rA.$

Flow chart:

See page II-12-4.

Exceptional conditions:

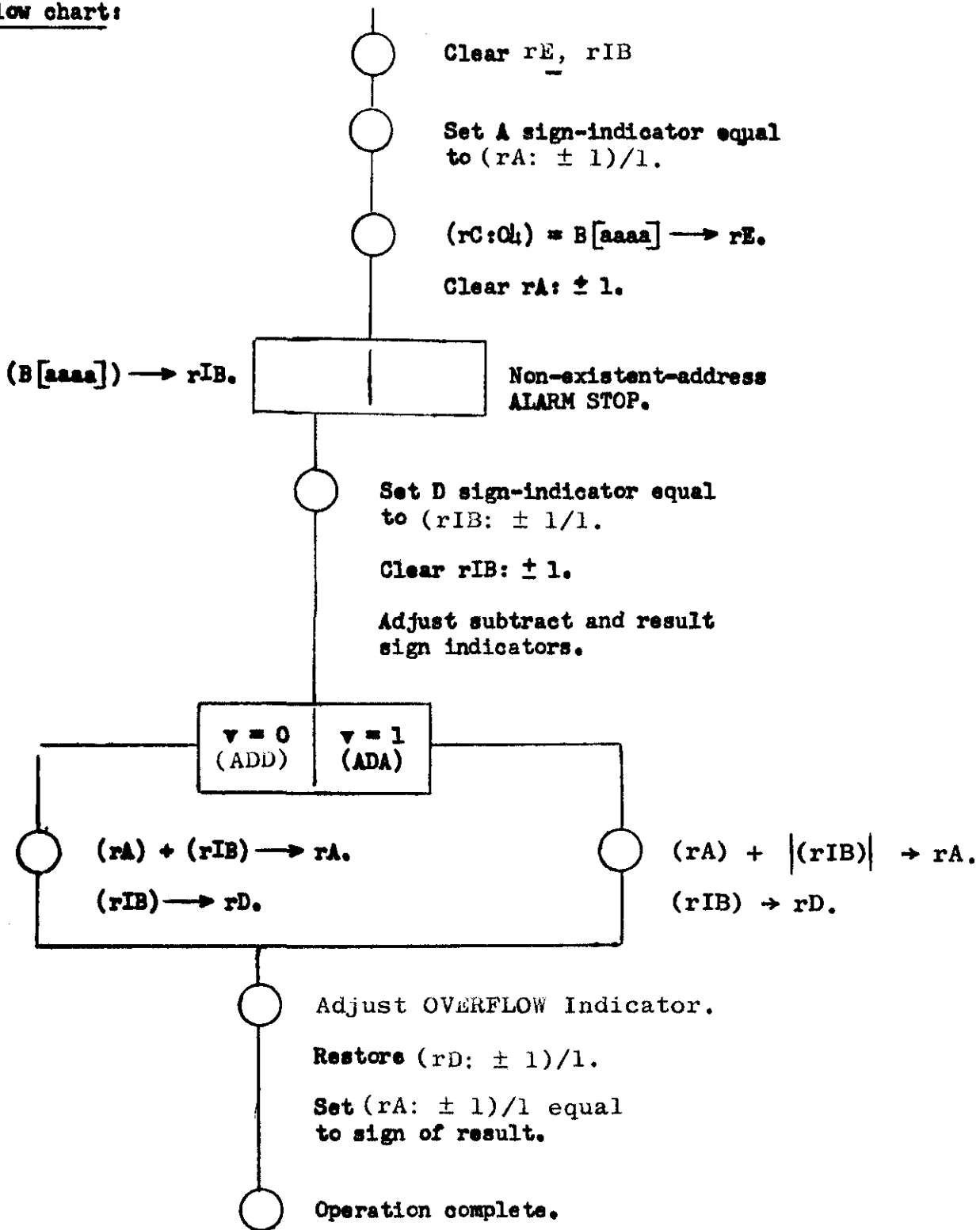
1. Non-existent-address ALARM STOP.

Remarks:

1. The ADD variation will be executed if $v \neq 1$.
2. The execution of both of these instructions can cause arithmetic overflow, in which case the OVERFLOW Indicator is set "on".
3. If the result of an ADD or ADA instruction is zero, the sign of the result is the same as the one-bit of the sign digit in the A register before execution of the instruction.

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of ADD.	Contents after execution of ADA.														
A	$(rA)_b + (B[aaaa])$	$(rA)_b + (B[aaaa])$														
R	Unchanged	Unchanged														
D	$(B[aaaa])$	$(B[aaaa])$														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>0</td><td>1</td><td>2</td><td>B[aaaa]</td></tr></table>	i	i	i	0	1	2	B[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>1</td><td>1</td><td>2</td><td>B[aaaa]</td></tr></table>	i	i	i	1	1	2	B[aaaa]
i	i	i	0	1	2	B[aaaa]										
i	i	i	1	1	2	B[aaaa]										
E	$B[aaaa]$	$B[aaaa]$														

Register name	Contents if non-existent-address ALARM STOP occurs.											
A	$(rA:\pm 1) = 0; (rA:00)_a = (rA:00)_b$											
R	Unchanged											
D	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>+</td><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>2</td><td>a</td><td>a</td><td>a</td><td>a</td></tr></table>	+	i	i	i	v	1	2	a	a	a	a
+	i	i	i	v	1	2	a	a	a	a		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>2</td><td>B[aaaa]</td></tr></table>	i	i	i	v	1	2	B[aaaa]				
i	i	i	v	1	2	B[aaaa]						
E	$B[aaaa]$											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: SUBTRACT

SUBTRACT ABSOLUTE

Operation code: 13

Abbreviation: SUB

SUA

Time (μ s):

Instruction format:

+ 1 2 3 4 5 6 7 8 9 0



Unnecessary to de-complement difference:

fetch: 90
 execute: 95
 total: 185

Definitions:

+: if + is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii: not relevant to the execution of these instruction.

v: variation designator:

v = 0: SUBTRACT will be executed.

v = 1: SUBTRACT ABSOLUTE will be executed.

Op: operation code.

aaaa: address of base of location of subtrahend.

Necessary to de-complement difference:

fetch: 90
 execute: 155
 total: 245

Description of operation:

Summary:

$v = 0: (rA) - (B[aaaa]) \rightarrow rA.$

$v = 1. (rA) - |(B[aaaa])| \rightarrow rA.$

Flow chart:

See page II-13-4

Exceptional conditions:

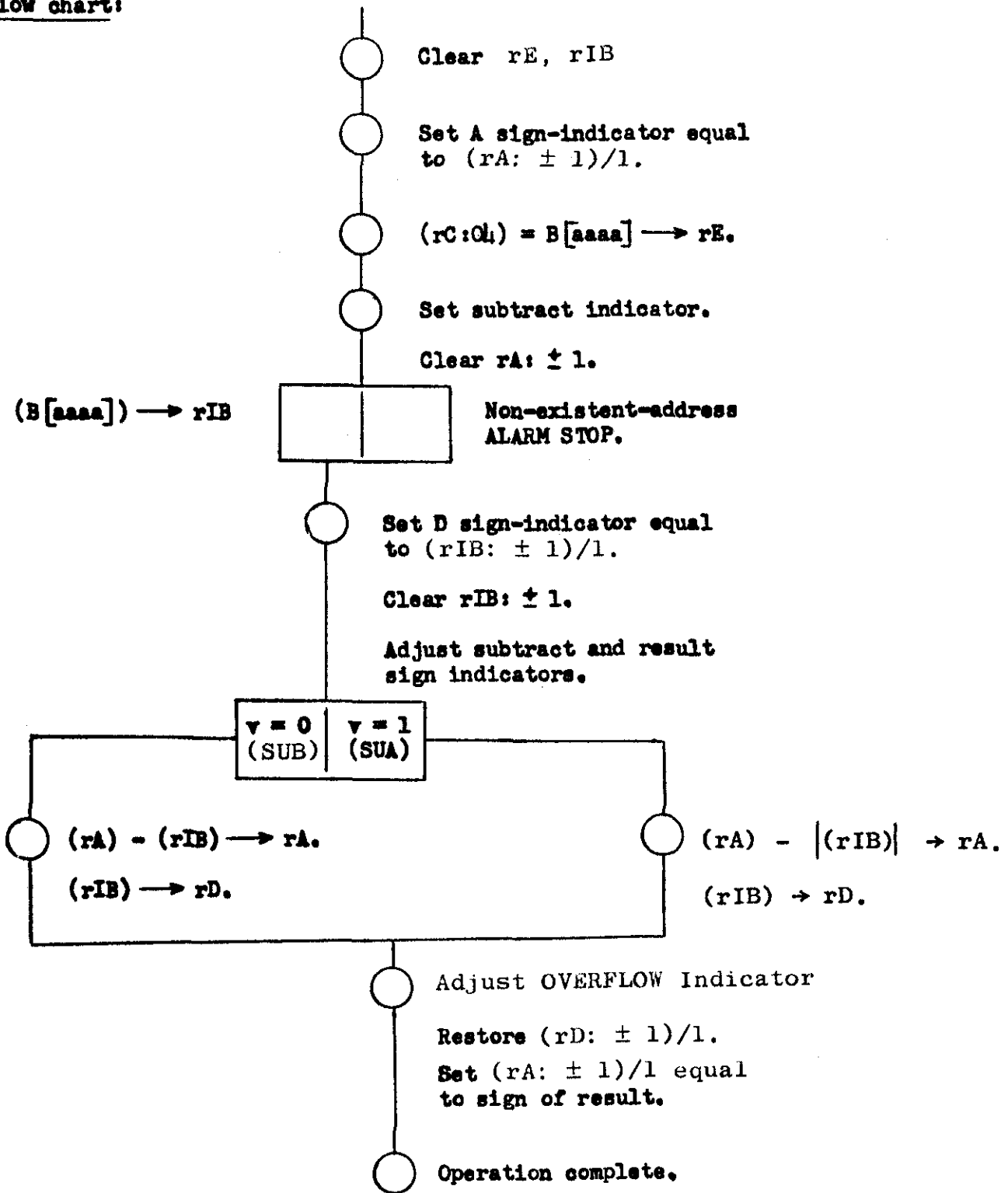
1. Non-existent-address ALARM STOP.

Remarks:

1. The SUBTRACT variation will be executed if $v \neq 1$.
2. The execution of both of these instructions can cause arithmetic flow, in which case the OVERFLOW Indicator is set "on".
3. If the result of a SUB or SUA instruction is zero, the sign of the result is the same as the one-bit of the sign digit in the A register before execution of the instruction.

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of SUB	Contents after execution of SUA														
A	$(rA)_b - (B[aaaa])$	$(rA)_b - (B[aaaa]) $														
R	Unchanged	Unchanged														
D	$(B[aaaa])$	$(B[aaaa])$														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>0</td><td>1</td><td>3</td><td>B[aaaa]</td> </tr> </table>	i	i	i	0	1	3	B[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>1</td><td>1</td><td>3</td><td>B[aaaa]</td> </tr> </table>	i	i	i	1	1	3	B[aaaa]
i	i	i	0	1	3	B[aaaa]										
i	i	i	1	1	3	B[aaaa]										
E	B [aaaa]	B [aaaa]														

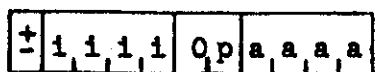
Register name	Contents if non-existent-address ALARM STOP occurs.											
A	$(rA: \pm 1)_a = 0; (rA:00)_a = (rA:00)_b$											
R	Unchanged											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>\pm</td><td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>3</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	\pm	i	i	i	v	1	3	a	a	a	a
\pm	i	i	i	v	1	3	a	a	a	a		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>1</td><td>3</td><td>B [aaaa]</td> </tr> </table>	i	i	i	v	1	3	B [aaaa]				
i	i	i	v	1	3	B [aaaa]						
E	B [aaaa]											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: MULTIPLY

Instruction format:

‡ 1 2 3 4 5 6 7 8 9 0



• Definitions:

- ‡: if ‡ is odd, B-register address-modification will occur; otherwise, there will be no such modification.

- iiii: not relevant to the execution of this instruction.

- Op: operation code.

- aaaa: address of base of location of multiplier.

Operation code: 14

Abbreviation: MUL

Time (μs):

Minimum: (rA) = †0000 00 0000
 fetch: 90
 execute: 140
 total: 230

Maximum: (rA) = †6666 66 6666
 fetch: 90
 execute: 3340
 total: 3430

Average: (rA) = †0123 45 6789
 fetch: 90
 execute: 1980
 total: 2070

Description of operation:

Summary:

The twenty-digit-long algebraic product, the contents of B[aaaa] multiplied by the contents of the A register, is generated. The ten low-order digits of the product replace the contents of the R register; the ten high-order digits of the product replace the contents of the A register. The sign of the product is inserted in the sign-digit position in both the A and R registers.

Flow chart:

See page II-14-4.

Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

1. Execution time for MULTIPLY-exclusive of fetch time-is a function of the magnitude of the multiplier, $(rA)_b$. It may be calculated from the following formula:

$$T = 90 + 5 \sum_{k=0}^9 M_k \text{ } \mu\text{s.}$$

where

$$M_k = 1 \quad \text{if } (rA:k1) = 0,$$

$$M_k = 13[(rA:k1)] + 1 \quad \text{if } 1 \leq (rA:k1) \leq 5,$$

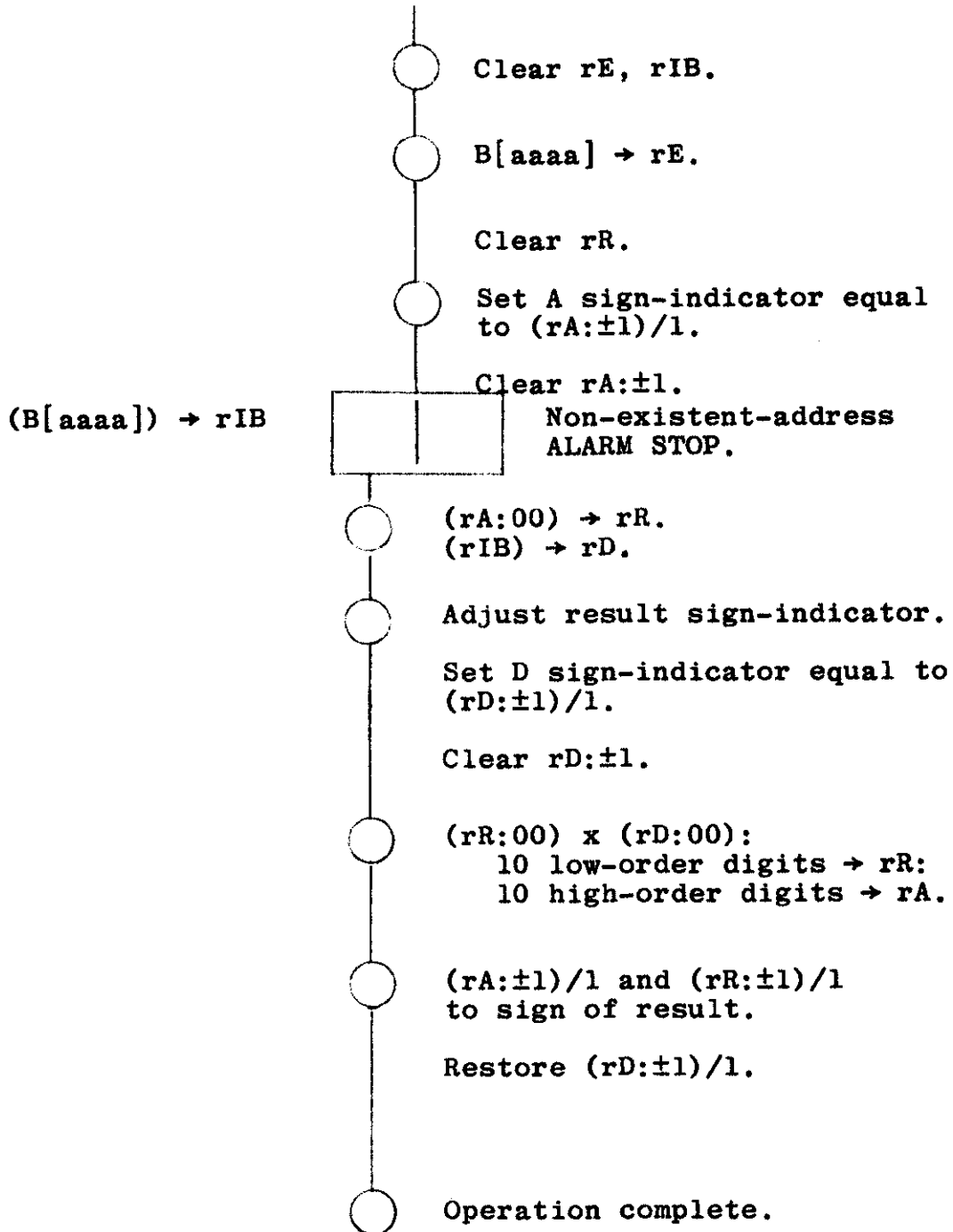
$$M_k = 13[11 - (rA:k1)] \quad \text{if } 6 \leq (rA:k1) \leq 9.$$

Hence, the following table:

$(rA:k1)$	0	1	2	3	4	5	6	7	8	9
M_k	1	14	27	40	54	66	65	52	39	26

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of MUL.	Contents if non-existent-address ALARM STOP occurs.																
A	high-order digits of product.	$(rA:\pm 1)_a = 0$; $(rA:00)_a = (rA:00)_b$																
R	low-order digits of product. $(rR:\pm 1)_a = (rA:\pm 1)_a$.	Cleared																
D*	$(B[aaaa])$	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>±</td> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>1</td> <td>4</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> </tr> </table>	±	i	i	i	i	1	4	a	a	a	a					
±	i	i	i	i	1	4	a	a	a	a								
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>1</td> <td>4</td> <td>B</td> <td>[aaaa]</td> </tr> </table>	i	i	i	i	1	4	B	[aaaa]	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>1</td> <td>4</td> <td>B</td> <td>[aaaa]</td> </tr> </table>	i	i	i	i	1	4	B	[aaaa]
i	i	i	i	1	4	B	[aaaa]											
i	i	i	i	1	4	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

$$*(rD:\pm 1)/2 = (rD:\pm 1)/4 = (rD:\pm 1)/8 = 0;$$

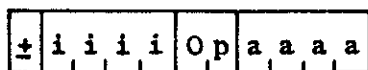
$$(rD:\pm 1)/1 = (B[aaaa]:\pm 1)/1;$$

$$(rD:00) = (B[aaaa]:00).$$

Operation name: DIVIDE

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

aaaa: address of base of location of divisor.

Operation code: 15

Abbreviation: DIV

Time (µs):

Minimum: OVERFLOW Indicator is set "on:"

fetch: 90
 execute: 95
 total: 185

Maximum: quotient =
 + 9090 90 9090:

fetch: 90
 execute: 6595
 total: 6685

Average: quotient =
 + 5555 55 5555:

fetch: 90
 execute: 3895
 total: 3985

Description of operation:

Summary:

The dividend is the twenty-digit number whose high-order digits are the contents of the A register and whose low-order digits are the contents of the R register. The sign of the A register is taken to be the sign of the dividend; the sign of the R register is not relevant.

The divisor is the contents of B[aaaa].

Before the arithmetic process is begun, the absolute value of the divisor is compared with the absolute value of the contents of the A register. Then:

1. If the absolute value of the divisor is greater than the absolute value of the contents of the A register, the process of division begins. The process terminates when a ten-digit quotient has been generated: the quotient, with sign, replaces the contents of the A register. The remainder replaces the contents of the R register; the sign of the remainder is the same as the sign of the dividend. On the other hand,

2. If the absolute value of the divisor is less than or equal to the absolute value of the contents of the A register, the OVERFLOW Indicator will be set "on", and the execution of the instruction terminates, leaving unaltered the contents of the A and R register.

Flow chart:

See page II-15-4.

Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

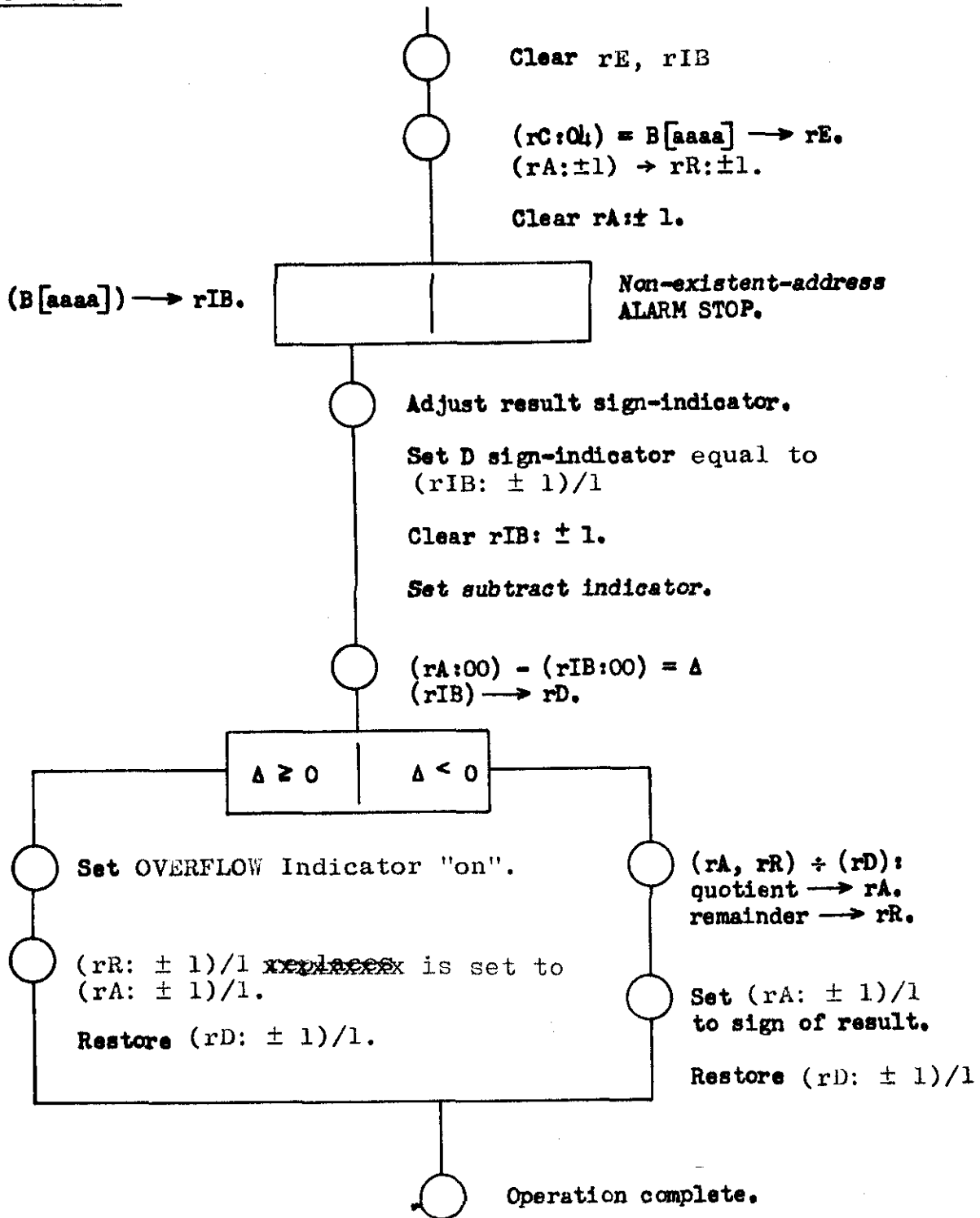
1, Ex

1. Execution time for DIVIDE-exclusive of fetch time-is a function of the magnitude of the quotient $(rA)_a$, if overflow does not occur. It may be calculated from the following formula:

$$\begin{aligned}
 T &= 3895 + 60 \sum_{k=0}^4 [(rA:2k+1,1) - (rA:2k,0)] \text{ us.} \\
 &= 3895 + 60 [(rA:11) - (rA:21) + \\
 &\quad (rA:31) - (rA:41) + \\
 &\quad (rA:51) - (rA:61) + \\
 &\quad (rA:71) - (rA:81)].
 \end{aligned}$$

Description of operation:

Flow chart:



Register status:

Register name	Contents after execution of DIV.	Contents if the OVERFLOW Indicator is set "on."																
A	Quotient	one-bit $(rA:\pm 1)_a = \text{one-bit } (rA:\pm 1)_b$ other bits $(rA:\pm 1)_a = 0$																
R	Remainder	$(rA:00)_a = (rA:00)_b$ $(rR:\pm 1)_a = (rA:\pm 1)_b$ $(rR:00)_a = (rR:00)_b$																
D	$(B[aaaa])^*$	$(B[aaaa])^*$																
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>i</td><td>1</td><td>5</td><td>B</td><td>[aaaa]</td></tr></table>	i	i	i	i	1	5	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>i</td><td>1</td><td>5</td><td>B</td><td>[aaaa]</td></tr></table>	i	i	i	i	1	5	B	[aaaa]
i	i	i	i	1	5	B	[aaaa]											
i	i	i	i	1	5	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents if non-existent-address ALARM STOP occurs.											
A	$(rA:\pm 1) = 0; (rA:00)_a = (rA:00)_b$											
R	$(rR:\pm 1) = (rA:\pm 1)_b$ $(rR:00)_a = (rR:00)_b$											
D	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>±</td><td>i</td><td>i</td><td>i</td><td>i</td><td>1</td><td>5</td><td>a</td><td>a</td><td>a</td><td>a</td></tr></table>	±	i	i	i	i	1	5	a	a	a	a
±	i	i	i	i	1	5	a	a	a	a		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>i</td><td>i</td><td>i</td><td>i</td><td>1</td><td>5</td><td>B</td><td>[aaaa]</td></tr></table>	i	i	i	i	1	5	B	[aaaa]			
i	i	i	i	1	5	B	[aaaa]					
E	$B[aaaa]$											

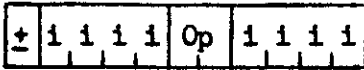
$*(rD:\pm 1)/2 = (rD:\pm 1)/4 = (rD:\pm 1)/8 = 0$
 $(rD:\pm 1)/1 = (B[aaaa] : \pm 1)/1;$
 $(rD:00) = (B[aaaa] : 00).$

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: ROUND

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

Operation code: 16

Abbreviation: RND

Time (us):

(rR:ll) < 5:
 fetch: 90
 execute: 15
 total: 105

(rR:ll) ≥ 5:
 fetch: 90
 execute: 70
 total: 160

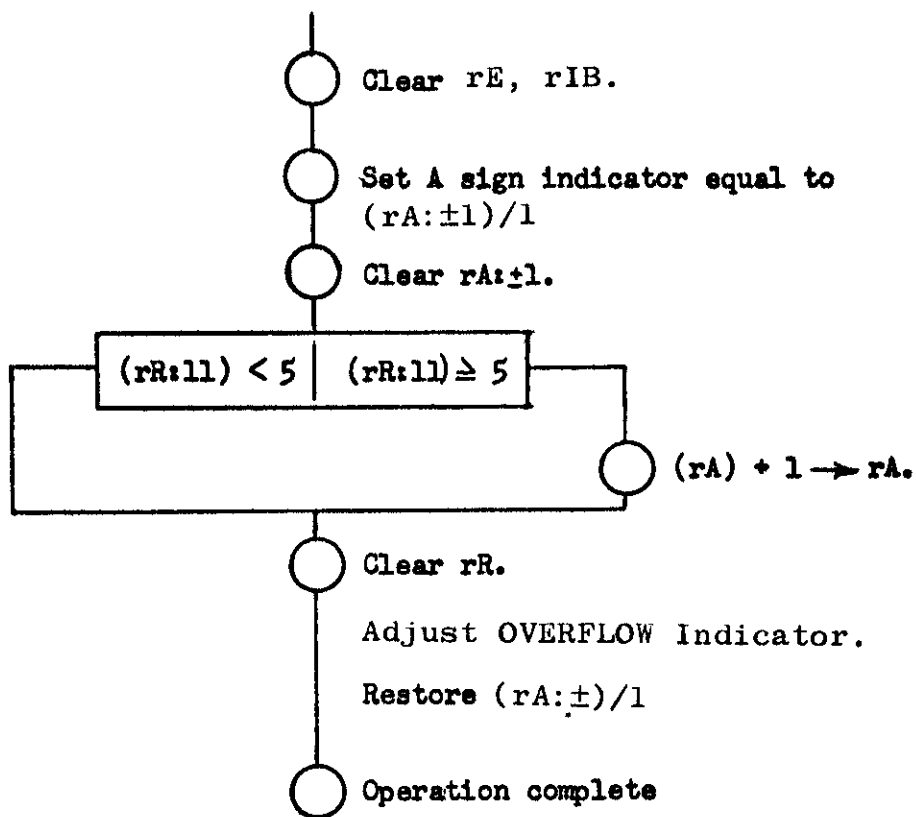
Description of operation:

Summary:

If $(rR:11) < 5$, clear rR.

If $(rR:11) \geq 5$, $|(rA)|$ is increased by + 0000 00 0001. Then rR is cleared.

Flow chart:



Exceptional conditions: NONE.

Remarks:

1. The execution of this instruction can cause arithmetic overflow, in which case the OVERFLOW Indicator is set to "on".

2. Although the address field is not relevant to the execution of this instruction, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.

Register status:

Register name	Contents after execution of RND.						
A	$(rA)_b$, rounded						
R	cleared.						
D	<table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">±</td> <td style="text-align: center;">1 1 1 1</td> <td style="text-align: center;">1 6</td> <td style="text-align: center;">1 1 1 1</td> </tr> </table>	±	1 1 1 1	1 6	1 1 1 1		
±	1 1 1 1	1 6	1 1 1 1				
B	Unchanged						
P	$(rP)_b + 1$						
C	<table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">1 1 1 1</td> <td style="text-align: center;">1 6</td> <td style="text-align: center;">B [1111]</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </table>	1 1 1 1	1 6	B [1111]			
1 1 1 1	1 6	B [1111]					
E	Cleared						

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: EXTRACT

Operation code: 17

Abbreviation: EXT

Instruction format:

Time (μ s)

± 1 2 3 4 5 6 7 8 9 0

±	1	2	3	4	5	6	7	8	9	0
±	i	i	i	i	Op	a	a	a	a	

fetch: 90

execute: $\frac{145}{235}$

total: 235

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

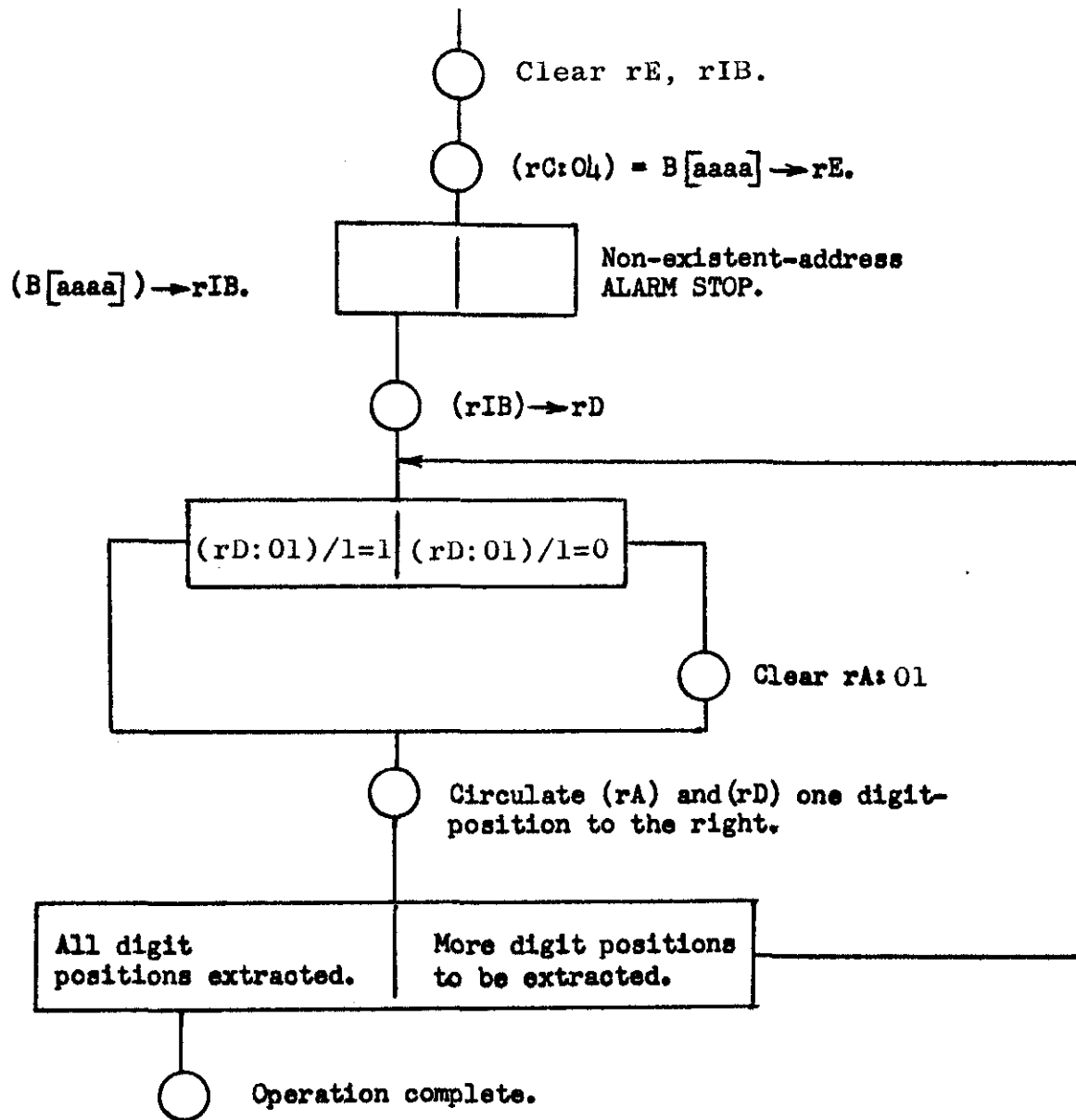
aaaa: address of base of location of extractor.

Description of operations:

Summary:

Wherever the extractor — which is the contents of B [aaaa] — has an even digit, the corresponding digit in rA is replaced by 0; wherever the extractor has an odd digit, the corresponding digit in rA is not altered.

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

Register status:

Register name	Contents after execution of EXT.	Contents if non-existent-address ALARM STOP occurs.														
A	See description of operation.	Unchanged.														
R	Unchanged.	Unchanged.														
D	(B[aaaa])	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	1	1	1	1	1	7	a	a	a	a			
±	1	1	1	1	1	7	a	a	a	a						
B	Unchanged.	Unchanged.														
P	(rP) _b + 1	(rP) _b + 1														
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	1	1	1	1	1	7	B[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	1	1	1	1	1	7	B[aaaa]
1	1	1	1	1	7	B[aaaa]										
1	1	1	1	1	7	B[aaaa]										
E	B[aaaa]	B[aaaa]														

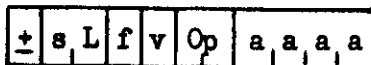
OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: COMPARE FIELD A
 COMPARE FIELD R

Operation code: 18
Abbreviation: CFA
 CFR

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Time (µs):

fetch: 90
 execute: 150
 total: 240

Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

- s: partial-word designator:
 - f = 0: s is not relevant.
 - f = 1: s designates the position, within the word, of the low-order digit of each partial-word operand.

- L: partial-word designator:
 - f = 0: L is not relevant.
 - f = 1: L specifies the number of digits in each partial-word operand.

- f: field designator:
 - f = 0: entire words will be used as operands.
 - f = 1: the contents of the partial-word fields defined by digits s and L will be used as operands.

- v: variation designator:
 - v = 0: COMPARE FIELD A will be executed.
 - v = 1: COMPARE FIELD R will be executed.

- Op: operation code.

- aaaa: address of base of location of comparator.

Description of operation:

Summary:

v = 0: COMPARE FIELD A will be executed.

Compare the contents of the specified field in the A register with the corresponding field in B[aaaa]. According as the contents of the specified part of the A register are greater than, equal to, or less than, the corresponding part of B[aaaa], set the COMPARISON Indicator to HIGH, EQUAL, or LOW.

v = 1: COMPARE FIELD R will be executed.

Except that the R register is used, this variation is identical with the one specified by v = 0.

Flow chart:

See page II-18-4.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP.

Remarks:

1. The COMPARE FIELD A variation will be executed if $v \neq 1$.

2. If f is an even integer, this has the same effect as f = 0; if f is an odd integer, this has the same effect as f = 1.

3a. If the sign-digit position of a word is not included in the field specified as operand, then the comparison may be considered to be made with respect to the absolute value of each operand. On the other hand,

3b. If the sign-digit position of a word is included in the field specified as operand, then the comparison is algebraic, in which case the contents of the sign-digit position are regarded as having the following order: 3 < 2 < 1 < 0 < 7 < 6 < 5 < 4 < 8 < 9. Hence, 3 9999 99 9999 < ... < 3 0000 00 0000 < 2 9999 99 9999 < ... < 2 0000 00 0000 < 1 9999 99 9999 < ... < 1 0000 00 0000 < 0 9999 99 9999 < 7 0000 00 0000 < ... < 7 9999 99 9999 < 6 0000 00 0000 < ... < 6 9999 99 9999 < 5 0000 00 0000 < ... < 5 9999 99 9999 < 4 0000 00 0000 < ... < 4 9999 99 9999 < 8 0000 00 0000 < ... < 8 9999 99 9999 < 9 0000 00 0000 < ... < 9 9999 99 9999.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Remarks (continued):

3c. One consequence of the ordering described above is that the sign digit must never be included in a comparison field if that field is alphanumeric; otherwise, the inverse of the natural alphabetic order will be determined.

3d. A second consequence of the ordering described above is that + 0000 00 0000 > - 0000 00 0000.

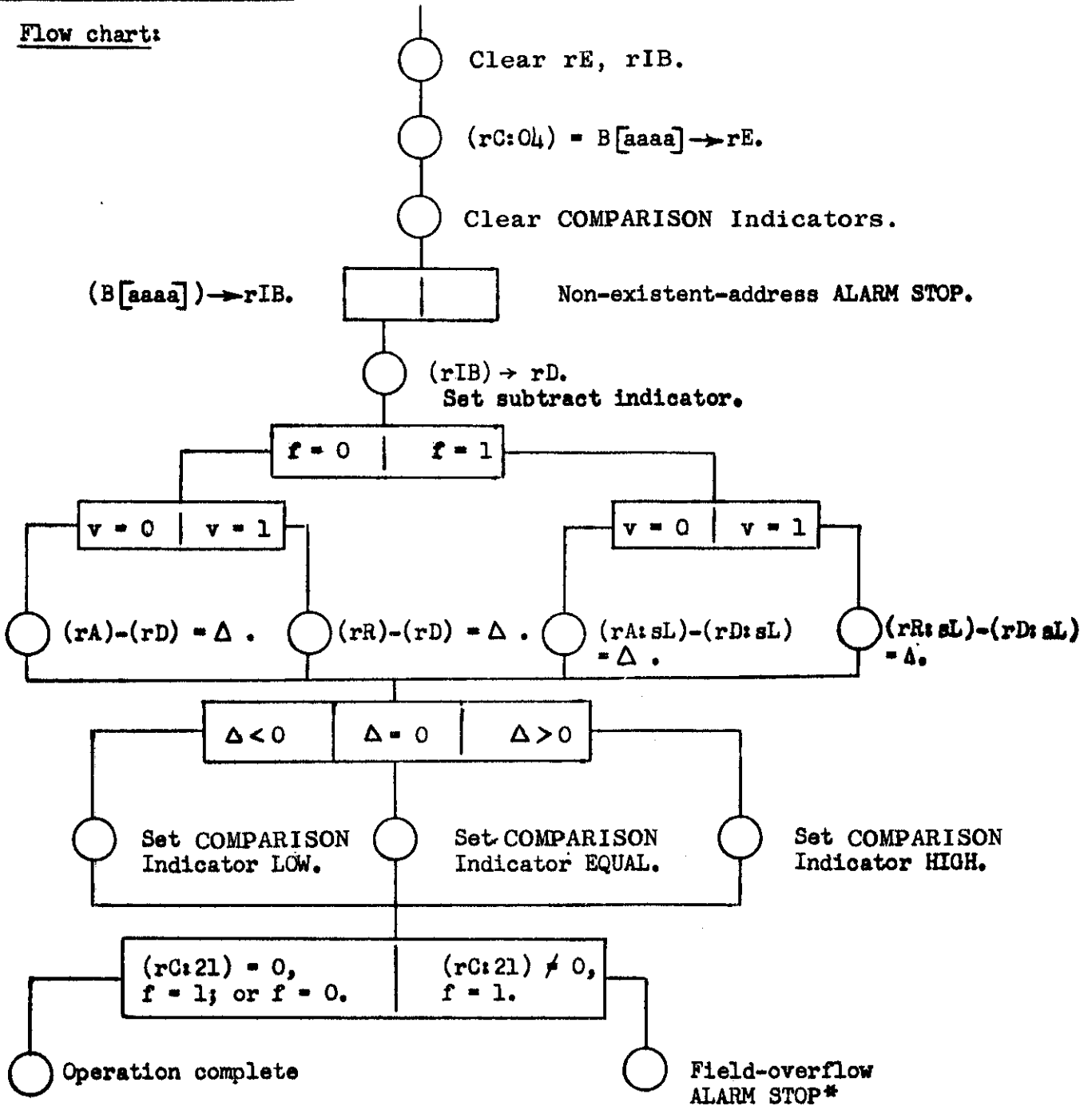
4. The order relationships of the content of the sign-digit position were determined as follows: the principal requirement is that negative numbers shall precede positive numbers; hence, 1 0. A second requirement--sorting--is that alphanumeric information shall precede numeric; hence, 2 1 0. The remaining order relationships are just a by-product of the way in which the required ordering of 2, 1, and 0 is achieved.

The technique used is known as the "threes complement" method: the 1-bit and the 2-bit of the sign digit are complemented if, and only if, the 8-digit is 0. The results are displayed in the following table:

Decimal Digit in Sign Position	Binary Representation	Threes Complemented	Decimal Equivalent (= order)
3	0011	0000	0
2	0010	0001	1
1	0001	0010	2
0	0000	0011	3
7	0111	0100	4
6	0110	0101	5
5	0101	0110	6
4	0100	0111	7
8	1000	1000	8
9	1001	1001	9

Description of operation:

Flow chart:



*See NOTE top of page II-18-6.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

NOTE: If $f = 1$, as each digit of the partial-word difference is generated, L (i.e., $(rC:21)$) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the comparison $(rC:21)$ will be different from zero, and field overflow will be detected.

Register status:

Register name	Contents after execution of either CFA or CFR.	Contents if non-existent-address ALARM STOP occurs.																
A	Unchanged.	Unchanged.																
R	Unchanged.	Unchanged.																
D	$(B[aaaa])$	<table border="1" style="display: inline-table;"> <tr> <td>±</td><td>s</td><td>L</td><td>f</td><td>v</td><td>1</td><td>8</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	f	v	1	8	a	a	a	a					
±	s	L	f	v	1	8	a	a	a	a								
B	Unchanged.	Unchanged.																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>0</td><td>f</td><td>v</td><td>1</td><td>8</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	0	f	v	1	8	B	[aaaa]	<table border="1" style="display: inline-table;"> <tr> <td>s</td><td>L</td><td>f</td><td>v</td><td>1</td><td>8</td><td>B</td><td>[aaaa]</td> </tr> </table>	s	L	f	v	1	8	B	[aaaa]
0	0	f	v	1	8	B	[aaaa]											
s	L	f	v	1	8	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents after field-overflow ALARM STOP occurs.								
A	Unchanged.								
R	Unchanged.								
D	$(B[aaaa])$								
B	Unchanged.								
P	$(rP)_b + 1$								
C	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>*</td><td>f</td><td>v</td><td>1</td><td>8</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	*	f	v	1	8	B	[aaaa]
0	*	f	v	1	8	B	[aaaa]		
E	$B[aaaa]$								

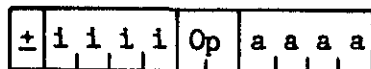
* L-s-1

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: ADD TO LOCATION

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iiii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location of augend.

Operation code: 19

Abbreviation: ADL

Time (µs):

Unnecessary to de-complement sum:

fetch: 90
 execute: 165
 total: 255

Necessary to de-complement sum:

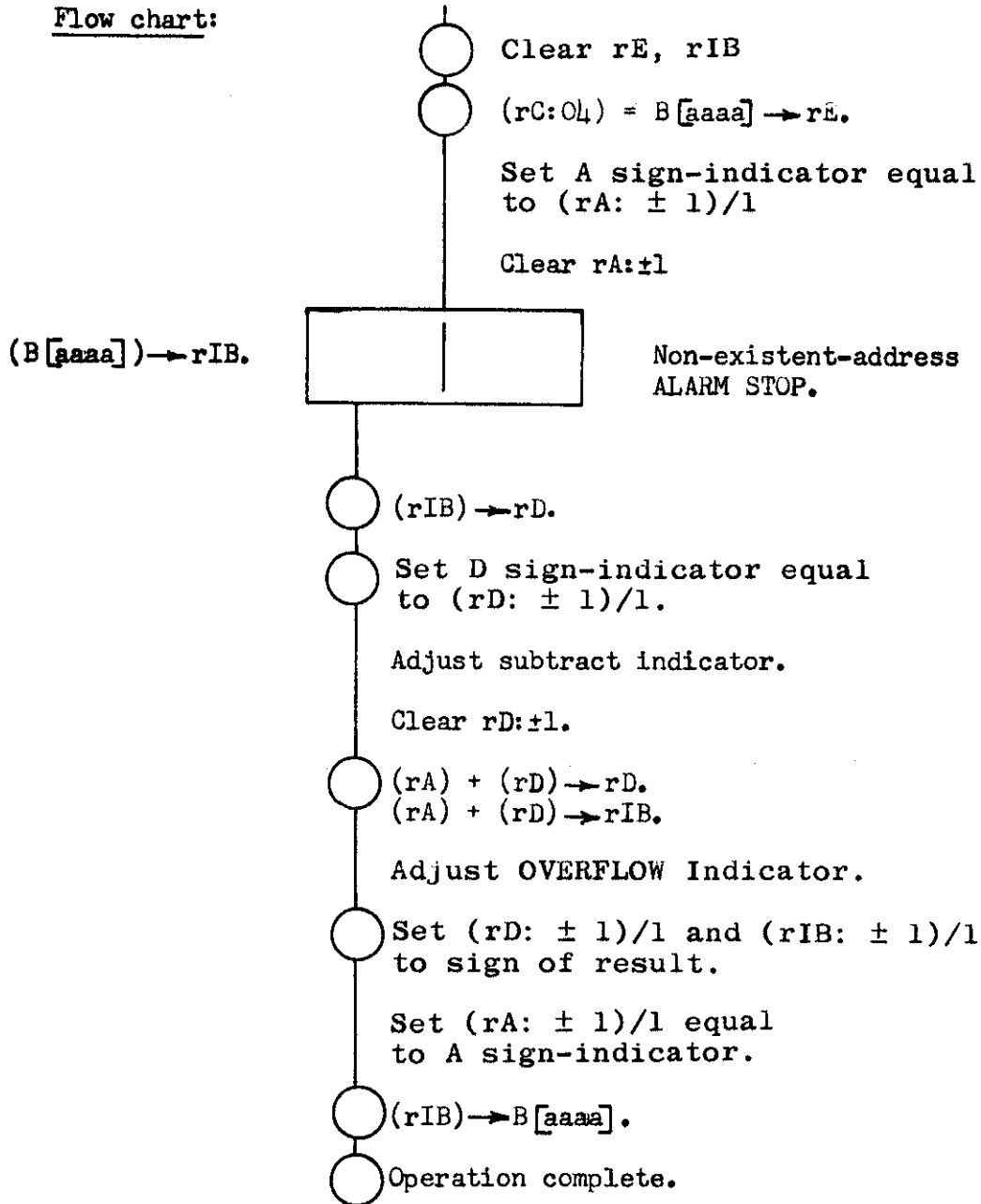
fetch: 90
 execute: 225
 total: 315

Description of operation:

Summary:

$(rA) + (B[aaaa]) \rightarrow B[aaaa].$

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

1. The execution of this instruction can cause arithmetic overflow, in which case the OVERFLOW Indicator is set to "on".

Register status:

Register name	Contents after execution of ADL.	Contents if non-existent-address ALARM STOP occurs.																										
A	Unchanged.*	$(rA:\pm 1) = 0, (rA:00)_a = (rA:00)_b$																										
R	Unchanged.	Unchanged.																										
D	$(rA)_b + (B[aaaa])$.	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">±</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">9</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	±	1	1	1	1	1	9	a	a	a	a															
±	1	1	1	1	1	9	a	a	a	a																		
B	Unchanged.	Unchanged.																										
P	$(rP)_b + 1$.	$(rP)_b + 1$.																										
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">9</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">]</td> </tr> </table>	1	1	1	1	1	9	B	[a	a	a	a]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">9</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">]</td> </tr> </table>	1	1	1	1	1	9	B	[a	a	a	a]
1	1	1	1	1	9	B	[a	a	a	a]																
1	1	1	1	1	9	B	[a	a	a	a]																
E	$B[aaaa]$	$B[aaaa]$																										

*Except that $(rA:\pm 1)/2=(rA:\pm 1)/4=(rA:\pm 1)/8=0$

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: INCREASE B, BRANCH

Operation code: 20

Abbreviation: IBB

Instruction format:

Time (μ s)

\pm 1 2 3 4 5 6 7 8 9 0

No branch:

\pm	1	2	3	4	5	6	7	8	9	0
\pm	n	n	n	n	Op	a	a	a	a	

fetch: 90
 execute: 35
 total: 125

Definitions:

Branch:

\pm if \pm is odd, B-register address-modification will occur; otherwise, there will be no modification.

fetch: 90
 execute: 55
 total: 145

nnnn: modifier for the contents of the B register.

Op: operation code.

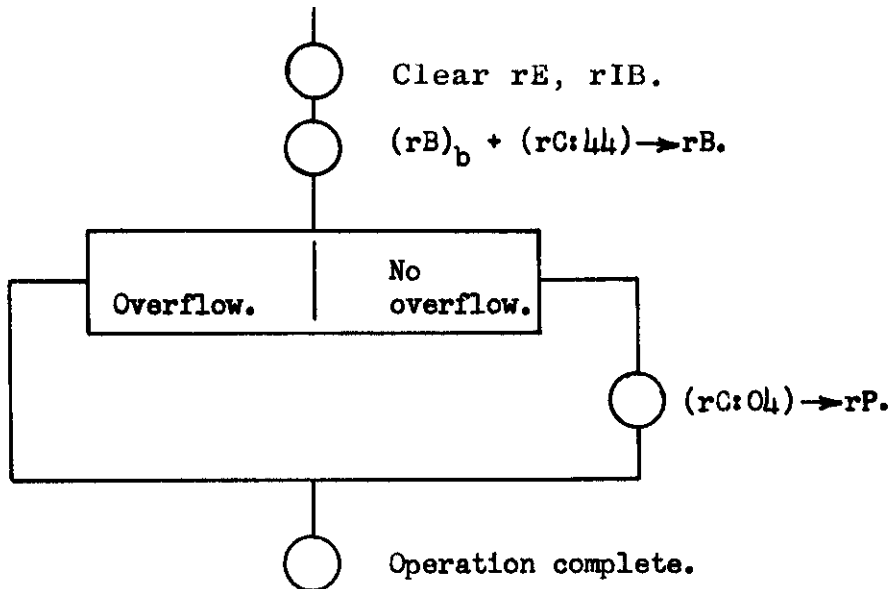
aaaa: address of base of location containing alternate instruction.

Description of operation:

Summary:

Increase (rB) by nnnn. If no overflow occurs, set (rP) = B[aaaa] (i.e., prepare to branch to location containing alternate instruction); if overflow occurs, control continues in sequence. Overflow of the B register does not set the OVERFLOW Indicator on.

Flow chart:



Exceptional conditions: NONE

Remarks:

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status:

Register name	Contents after execution of IBB.													
A	Unchanged.													
R	"													
D	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">2</td> <td style="text-align: center;">0</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	n	n	n	n	2	0	a	a	a	a		
+	n	n	n	n	2	0	a	a	a	a				
B	low-order digits of sum, $S = (rB)_b + nnnn.$													
P	$(rP)_b + 1$, if $S > 9999$ (overflow) $B[\bar{a}aaa]$, if $S \leq 9999$ (no overflow)													
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">2</td> <td style="text-align: center;">0</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">]</td> </tr> </table>	n	n	n	n	2	0	B	[a	a	a	a]
n	n	n	n	2	0	B	[a	a	a	a]		
E	Cleared.													

Examples:

1. $(rP)_b = 0412$
 $(rB)_b = 9984$
 $(rC) = 0001\ 20\ 0396$

 $(rB)_a = 9985$
 $(rP)_a = 0396$ (Branch; no overflow)

2. $(rP)_b = 0412$
 $(rB)_b = 9997$
 $(rC) = 0005\ 20\ 0396$

 $(rB)_a = 0002$
 $(rP)_a = 0413$ (No branch; overflow)

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: DECREASE B, BRANCH

Operation code: 21

Abbreviation: DBB

Instruction format:

Time (μ s):

‡ 1 2 3 4 5 6 7 8 9 0

‡	n	n	n	n	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

No branch:
 fetch: 90
 execute: 35
 total: 125

Definitions:

Branch:
 fetch: 90
 execute: 55
 total: 145

‡: if ‡ is odd, B-register address-modification will occur; otherwise there will be no such modification.

nnnn: modifier for the contents of the B register.

Op: operation code.

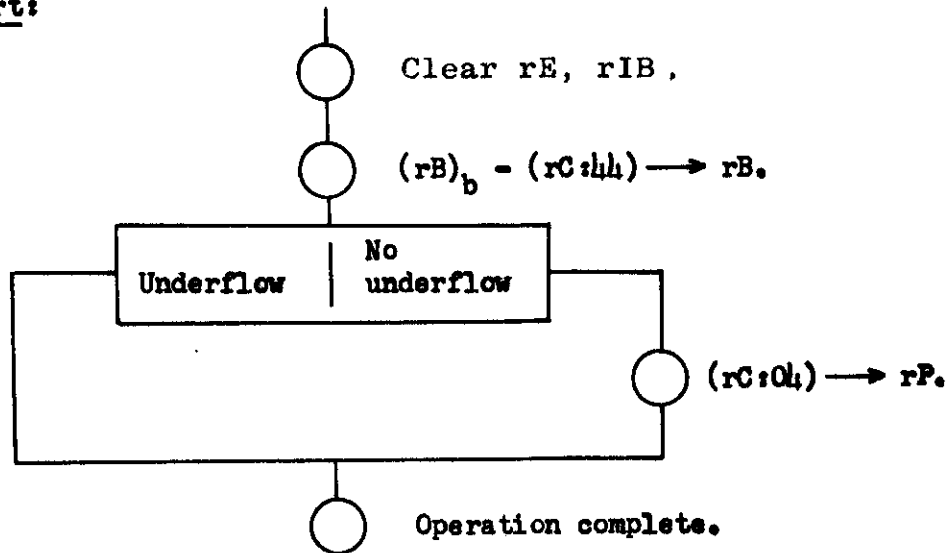
aaaa: address of base of location containing alternate instruction.

Description of operation:

Summary:

Decrease (rB) by mnnn. If no underflow occurs set (rP) = B[aaaa] (i.e., prepare to branch to location containing alternate instruction); if underflow occurs, control continues in sequence.

Flow chart:



NOTE: When underflow does occur, the complement of the algebraic difference $(rB)_p - (rC:hh)$, will appear in rB.

Exceptional conditions: none.

Remarks:

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status:

Register name	Contents after execution of DBB.											
A	Unchanged.											
R	"											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">2</td> <td style="text-align: center;">1</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	n	n	n	n	2	1	a	a	a	a
+	n	n	n	n	2	1	a	a	a	a		
B	low-order digits of D = $ (rB)_b - nnnn $											
P	$(rP)_b + 1$, if underflow. B[aaaa], if no underflow.											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">n</td> <td style="text-align: center;">2</td> <td style="text-align: center;">1</td> <td style="text-align: center;">B[aaaa]</td> </tr> </table>	n	n	n	n	2	1	B[aaaa]				
n	n	n	n	2	1	B[aaaa]						
E	Cleared											

Examples:

1. $(rP)_b = 0412$
 $(rB)_b = 0006$
 $(rC) = 0002\ 20\ 0396$

 $(rB)_a = 0004$
 $(rP)_a = 0396$ (Branch; no underflow.)

2. $(rP)_b = 0412$
 $(rB)_b = 0002$
 $(rC) = 0005\ 20\ 0396$

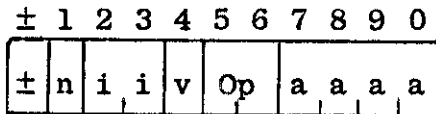
 $(rB)_a = 9997$
 $(rP)_a = 0413$ (No branch; underflow.)

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: FLOATING ADD
 FLOATING ADD ABSOLUTE

Operation code: 22
Abbreviation: FAD
 FAA

Instruction format:



Time (μs) (continued):

Sum = 0:
 fetch: 90
 execute: 190
 total: 280

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise there will be no such modification.

n: normalizing limiter.

ii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: FLOATING ADD will be executed.

v = 1: FLOATING ADD ABSOLUTE will be executed.

Op: operation code.

aaaa: address of base of location of augend.

Sum ≠ 0:
 no de-complement:

minimum:
 fetch: 90
 execute: 125
 total: 215

maximum:
 fetch: 90
 execute: 270
 total: 360

de-complement:

minimum:
 fetch: 90
 execute: 190
 total: 280

Time (μs):

Underflow:

Minimum:

fetch: 90
 execute: 235
 total: 325

Maximum:

fetch: 90
 execute: 320
 total: 410

maximum:
 fetch: 90
 execute: 325
 total: 415

Description of operation:Summary:

Both variations treat the operands like floating-point numbers:

$$\underline{v = 0}: (rA) + (B[aaaa]) \rightarrow rA.$$

$$\underline{v = 1}: (rA) + |(B[aaaa])| \rightarrow rA.$$

Let m be the number of digit positions through which the sum is shifted to obtain the sum in normalized form. If $m > n$, the Data Processor will halt at the end of the Execute Phase. $n = 8, 9, \text{ or } 0$ (where 0 means 10) is meaningless and irrelevant.

Flow chart:

See page II-22-5.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Normalization-limit ALARM STOP.

Remarks:

1. The FLOATING ADD variation will be selected for execution if $v \neq 1$.

2. The execution of these instructions can cause arithmetic overflow, in which case the OVERFLOW Indicator will be turned on. Arithmetic overflow occurs when the machine-coded exponent exceeds 99.

3. If the result of a FAD or FAA instruction is zero, the A register--which will contain the sum--is cleared. Hence, a zero sum has the form + 00 0000 0000, that is, 0×10^{-50} .

4. In adjusting the exponents so that addition can take place, low-order digits of one of the operands--the one whose exponent is smaller--are lost. Rounding does not occur; that is, the highest-order digit discarded is not examined to determine whether the low-order digit retained should be increased by 1.

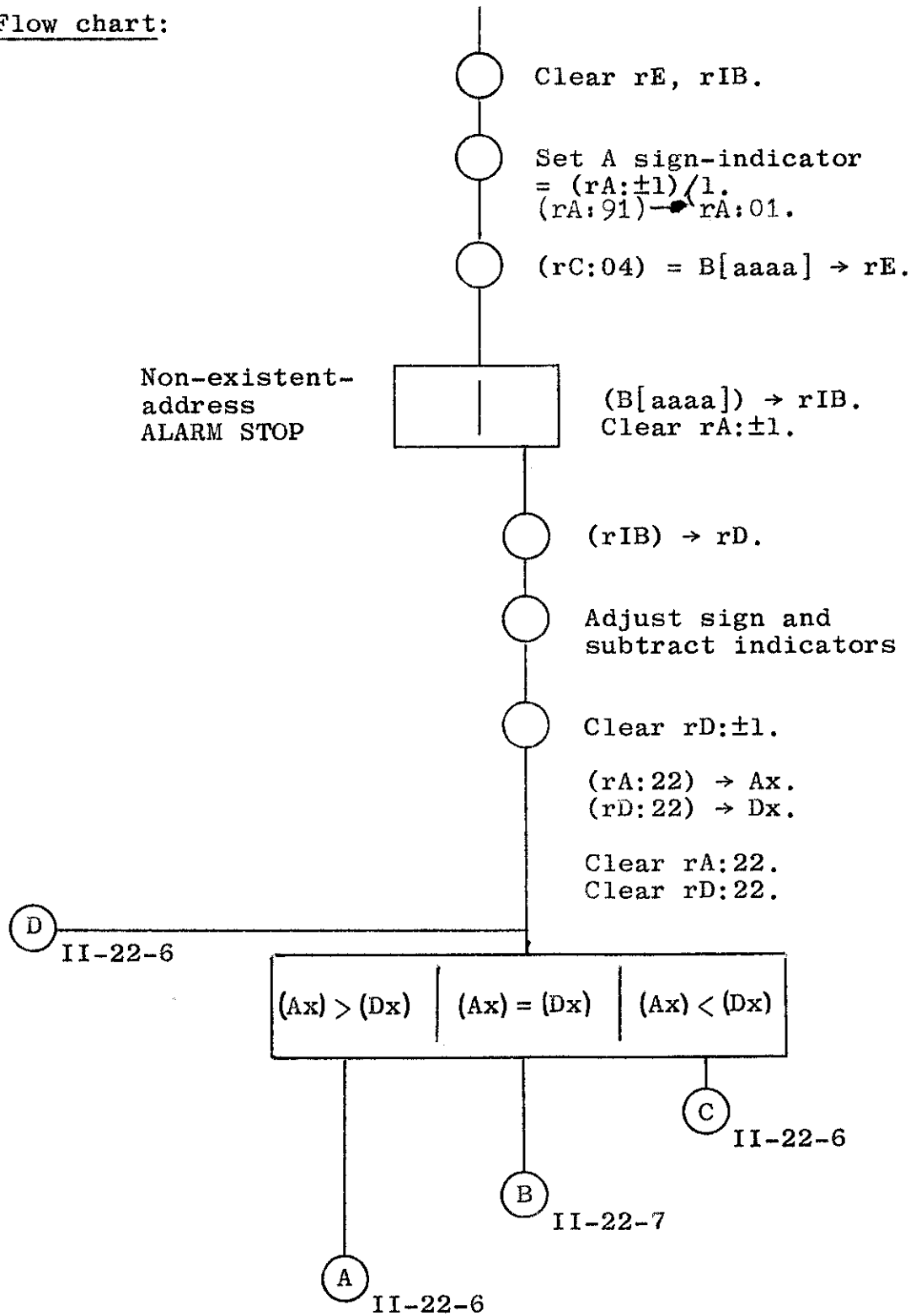
5. During the normalization process the number of digit positions, m , through which the sum is to be shifted is determined. If $m > n$, $n \neq 8, 9, \text{ or } 0$, where n is the normalizing limiter, then $(rC:11)_a = 10 - (m - n)$; if $m = n$, $(rC:11)_a = 0$. If $n = 8, 9, \text{ or } 0$, $(rC:11)_a = 0$.

After normalization, if $(rC:11) \neq 0$, the Single Step Toggle (SST) will be turned on. Whenever SST is turned on, the Data Processor will stop at the end of the cycle--Fetch Phase or Execute Phase--and wait for a signal to resume (automatic) operation.

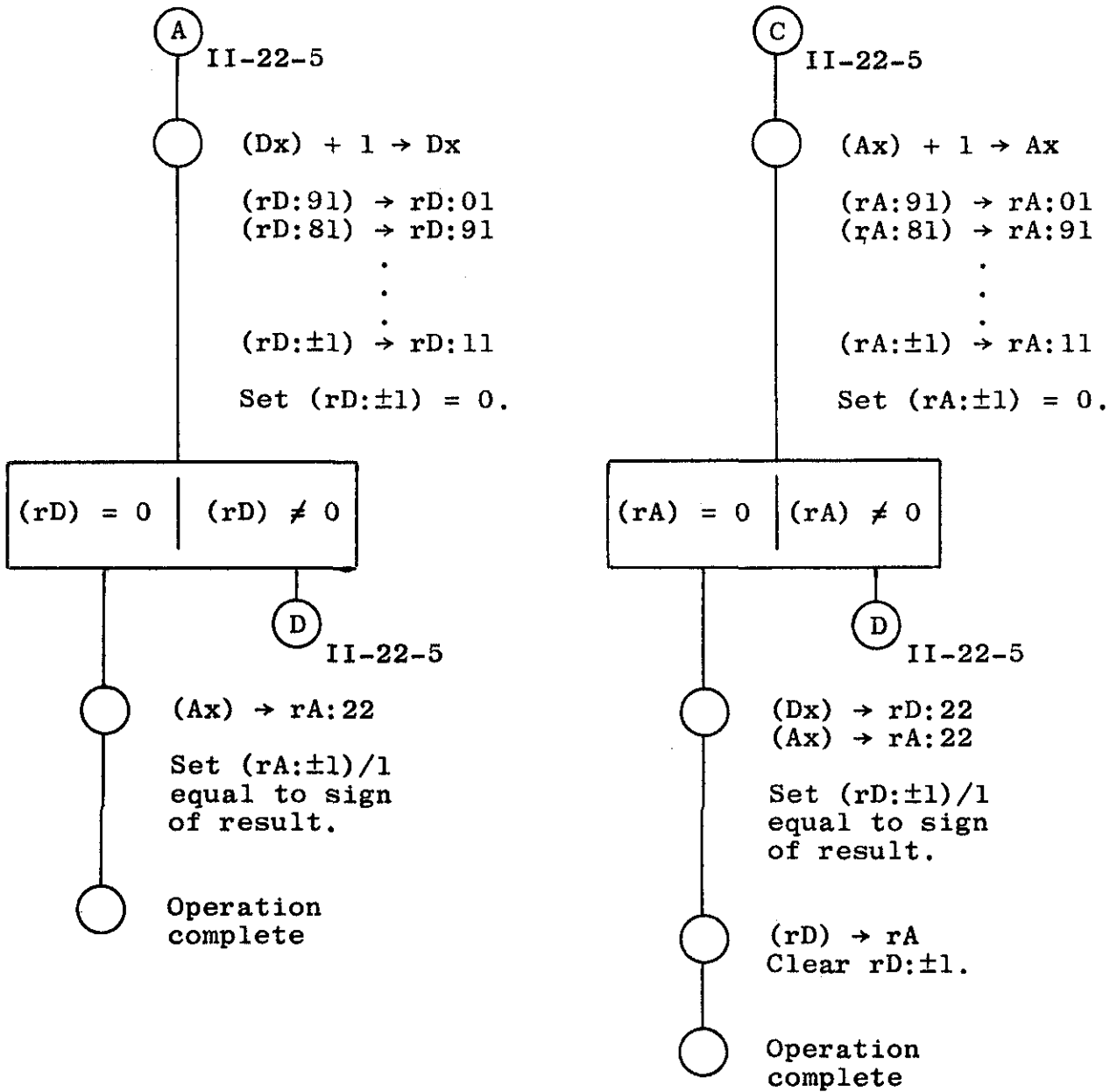
THE DATA PROCESSOR

Description of operation:

Flow chart:

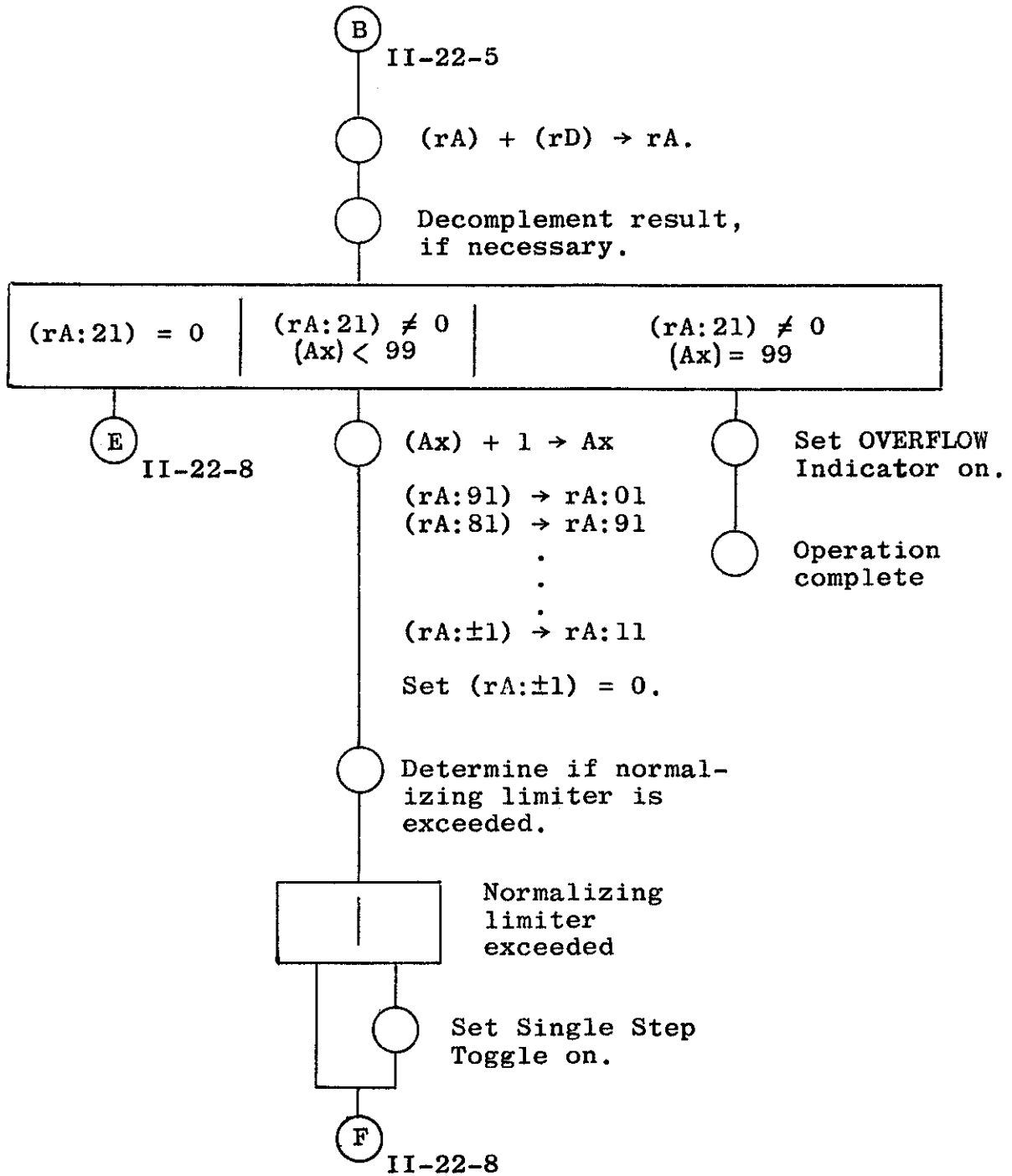


Flow chart (continued):

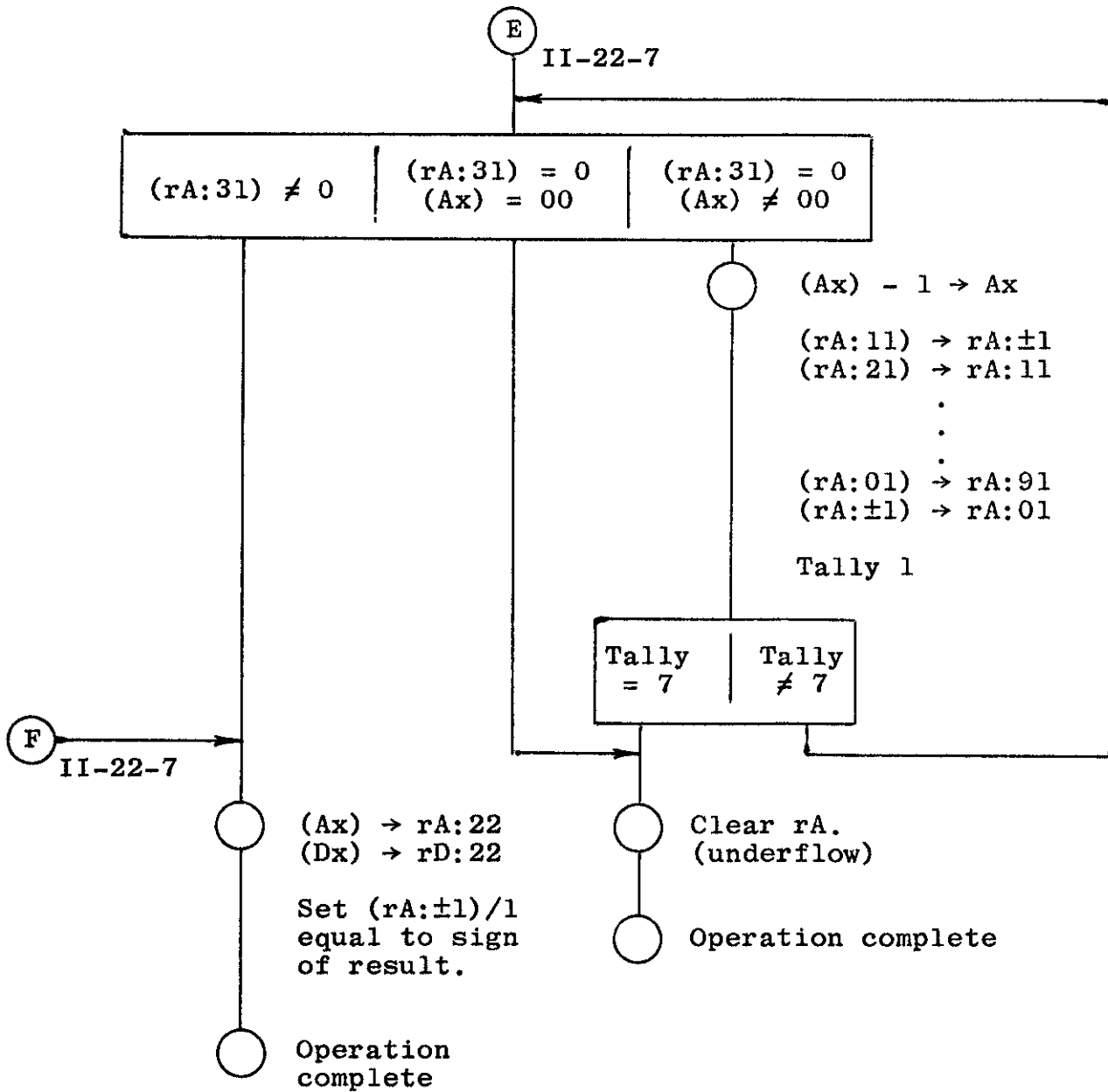


THE DATA PROCESSOR

Flow chart (continued):



Flow chart (continued):



THE DATA PROCESSOR

Register status:

Register name	Contents after execution of FAD.	Contents after execution of FAA.																								
A	$(rA)_b + (B[aaaa])$	$(rA)_b + (B[aaaa])$																								
R	Unchanged	Unchanged																								
D	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0	0	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0															
0	1	0	0	0	0	0	0	0	0	0	0															
B	Unchanged	Unchanged																								
P	$(rP)_b + 1$	$(rP)_b + 1$																								
C	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>i</td><td>i</td><td>0</td><td>2</td><td>2</td><td>B[aaaa]</td> </tr> </table>	0	i	i	0	2	2	B[aaaa]	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>i</td><td>i</td><td>1</td><td>2</td><td>2</td><td>B[aaaa]</td> </tr> </table>	0	i	i	1	2	2	B[aaaa]										
0	i	i	0	2	2	B[aaaa]																				
0	i	i	1	2	2	B[aaaa]																				
E	B[aaaa]	B[aaaa]																								

Register name	Contents if non-existent-address ALARM STOP occurs.	Contents if OVERFLOW Indicator is on.														
A	Unchanged	Unnormalized sum without exponent.														
R	Unchanged	Unchanged														
D	<table border="1" style="display: inline-table;"> <tr> <td>±</td><td>n</td><td>i</td><td>i</td><td>v</td><td>2</td><td>2</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	n	i	i	v	2	2	a	a	a	a	Cleared			
±	n	i	i	v	2	2	a	a	a	a						
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table;"> <tr> <td>n</td><td>i</td><td>i</td><td>v</td><td>2</td><td>2</td><td>B[aaaa]</td> </tr> </table>	n	i	i	v	2	2	B[aaaa]	<table border="1" style="display: inline-table;"> <tr> <td>n</td><td>i</td><td>i</td><td>v</td><td>2</td><td>2</td><td>B[aaaa]</td> </tr> </table>	n	i	i	v	2	2	B[aaaa]
n	i	i	v	2	2	B[aaaa]										
n	i	i	v	2	2	B[aaaa]										
E	B[aaaa]	B[aaaa]														

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status (continued):

Register name	Contents if normalization-limit ALARM STOP occurs.											
A	Normalized sum											
R	Unchanged											
D	<table border="1" style="margin-left: 20px;"> <tr> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">1</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> <td style="width: 15px; text-align: center;">0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="margin-left: 20px;"> <tr> <td style="width: 15px; text-align: center;">*</td> <td style="width: 15px; text-align: center;">i</td> <td style="width: 15px; text-align: center;">i</td> <td style="width: 15px; text-align: center;">v</td> <td style="width: 15px; text-align: center;">2</td> <td style="width: 15px; text-align: center;">2</td> <td style="width: 15px; text-align: center;">B[aaaa]</td> </tr> </table>	*	i	i	v	2	2	B[aaaa]				
*	i	i	v	2	2	B[aaaa]						
E	B[aaaa]											

*10 - (m - n).

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: FLOATING SUBTRACT
 FLOATING SUBTRACT ABSOLUTE

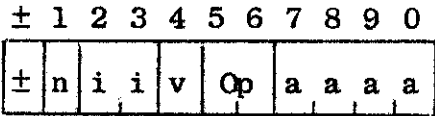
Operation code: 23

Abbreviation: FSU

FSA

Instruction format:

Time (μs) (continued):



Sum = 0

fetch: 90
 execute: 190
 total: 280

Definitions:

Sum ≠ 0:

±: if ± is odd, B-register address-modification will occur; otherwise there will be no such modification.

no de-complement:

n: normalizing limiter.

minimum:

ii: not relevant to the execution of these instructions.

fetch: 90
 execute: 125
 total: 215

v: variation designator:

maximum:

v = 0: FLOATING SUBTRACT will be executed.

fetch: 90
 execute: 270
 total: 360

v = 1: FLOATING SUBTRACT ABSOLUTE will be executed.

de-complement:

Op: operation code.

minimum:

aaaa: address of base of location of subtrahend.

fetch: 90
 execute: 190
 total: 280

Time (μs)

maximum:

Underflow:

minimum:

fetch: 90
 execute: 235
 total: 325

fetch: 90
 execute: 325
 total: 415

maximum:

fetch: 90
 execute: 320
 total: 410

Description of operation:Summary:

Both variations treat the operands like floating-point numbers:

$$\underline{v = 0}: (rA) - (B[aaaa]) \rightarrow rA.$$

$$\underline{v = 1}: (rA) - |(B[aaaa])| \rightarrow rA.$$

Let m be the number of digit positions through which the difference is shifted to obtain the difference in normalized form. If $m > n$, the Data Processor will halt at the end of the Execute Phase. $n = 8, 9, \text{ or } 0$ (where 0 means 10) is meaningless and irrelevant.

Flow chart:

See page II-23-5.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Normalization-limit ALARM STOP.

Remarks:

1. The FLOATING SUBTRACT variation will be selected for execution if $v \neq 1$.

2. The execution of these instructions can cause arithmetic overflow, in which case the OVERFLOW Indicator will be turned on. Arithmetic overflow occurs when the machine-coded exponent exceeds 99.

3. If the result of a FSU or FSA instruction is zero, the A register--which will contain the difference--is cleared. Hence, a zero difference has the form + 00 0000 0000, that is, 0×10^{-50} .

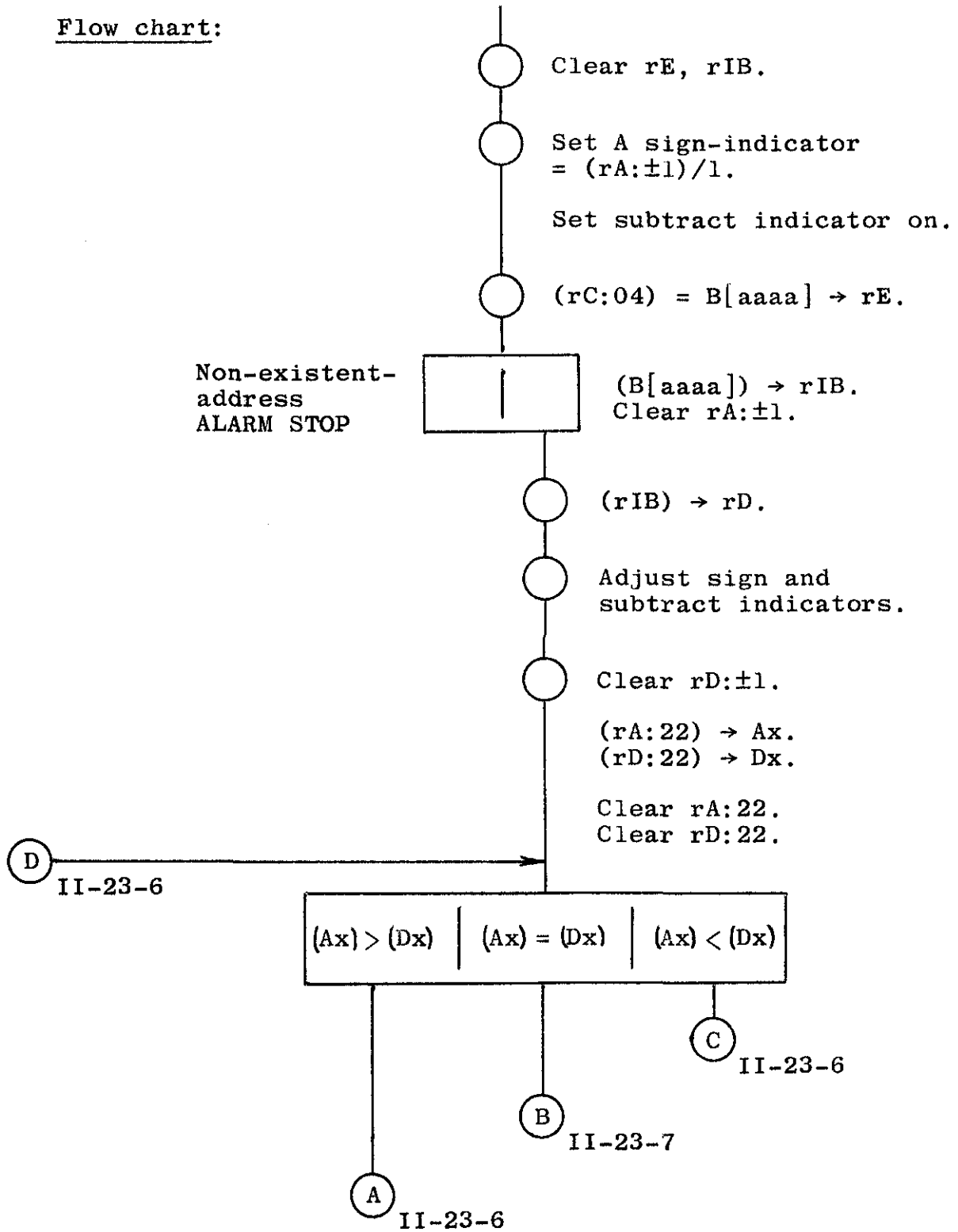
4. In adjusting the exponents so that subtraction can take place, low-order digits of one of the operands--the one whose exponent is smaller--are lost. Rounding does not occur; that is, the highest-order digit lost is not examined to determine whether the low-order digit retained should be increased by 1.

5. During the normalization process the number of digit positions, m , through which the difference is to be shifted is determined. If $m > n$, $n \neq 8, 9, \text{ or } 0$, where n is the normalizing limiter, then $(rC:11)_a = 10 - (m - n)$; if $m = n$, $(rC:11)_a = 0$. If $n = 8, 9, \text{ or } 0$, $(rC:11)_a = 0$.

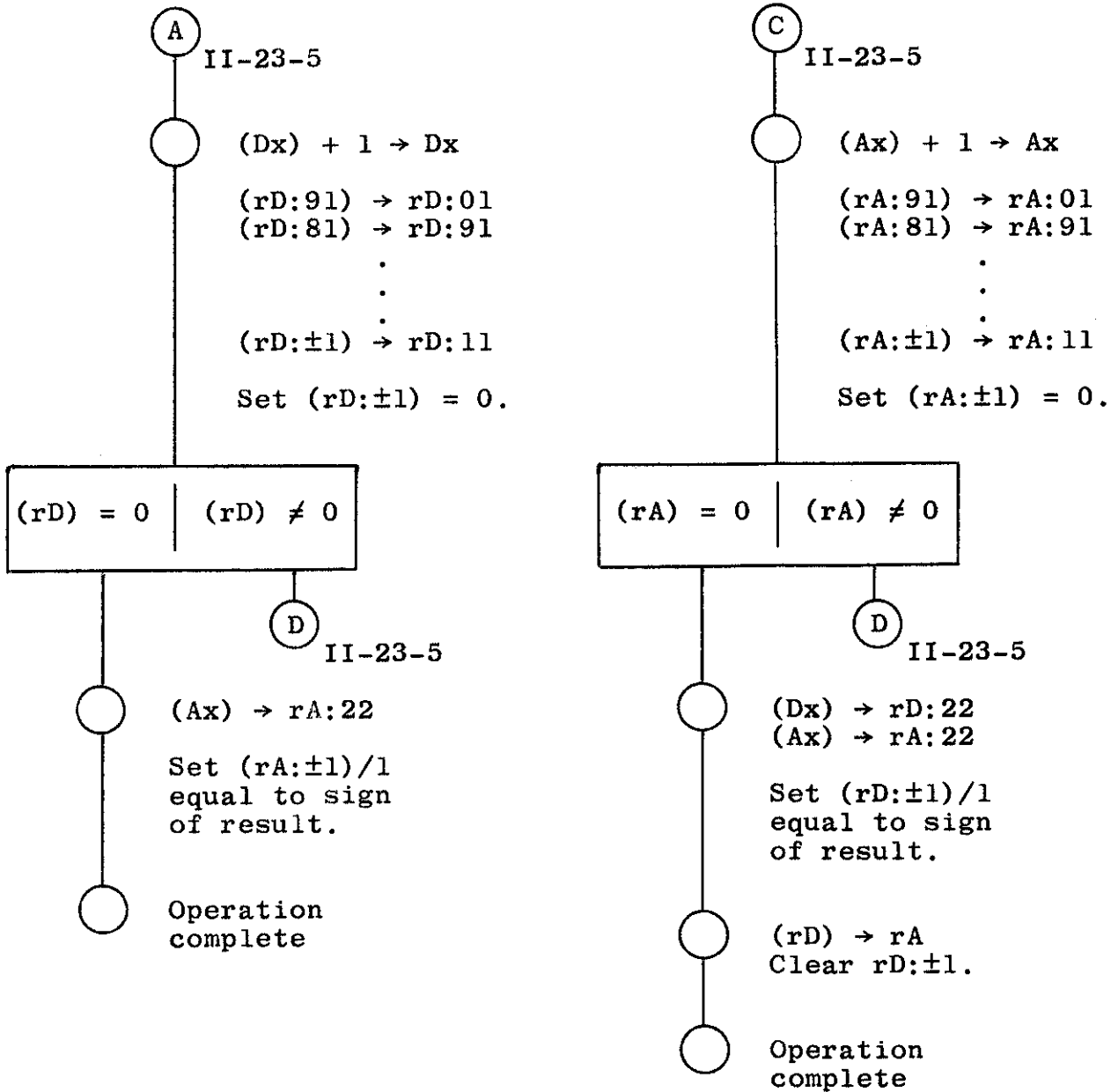
After normalization, if $(rC:11) \neq 0$, the Single Step Toggle (SST) will be turned on. Whenever SST is turned on, the Data Processor will stop at the end of the cycle--Fetch Phase or Execute Phase--and wait for a signal to resume (automatic) operation.

Description of operation:

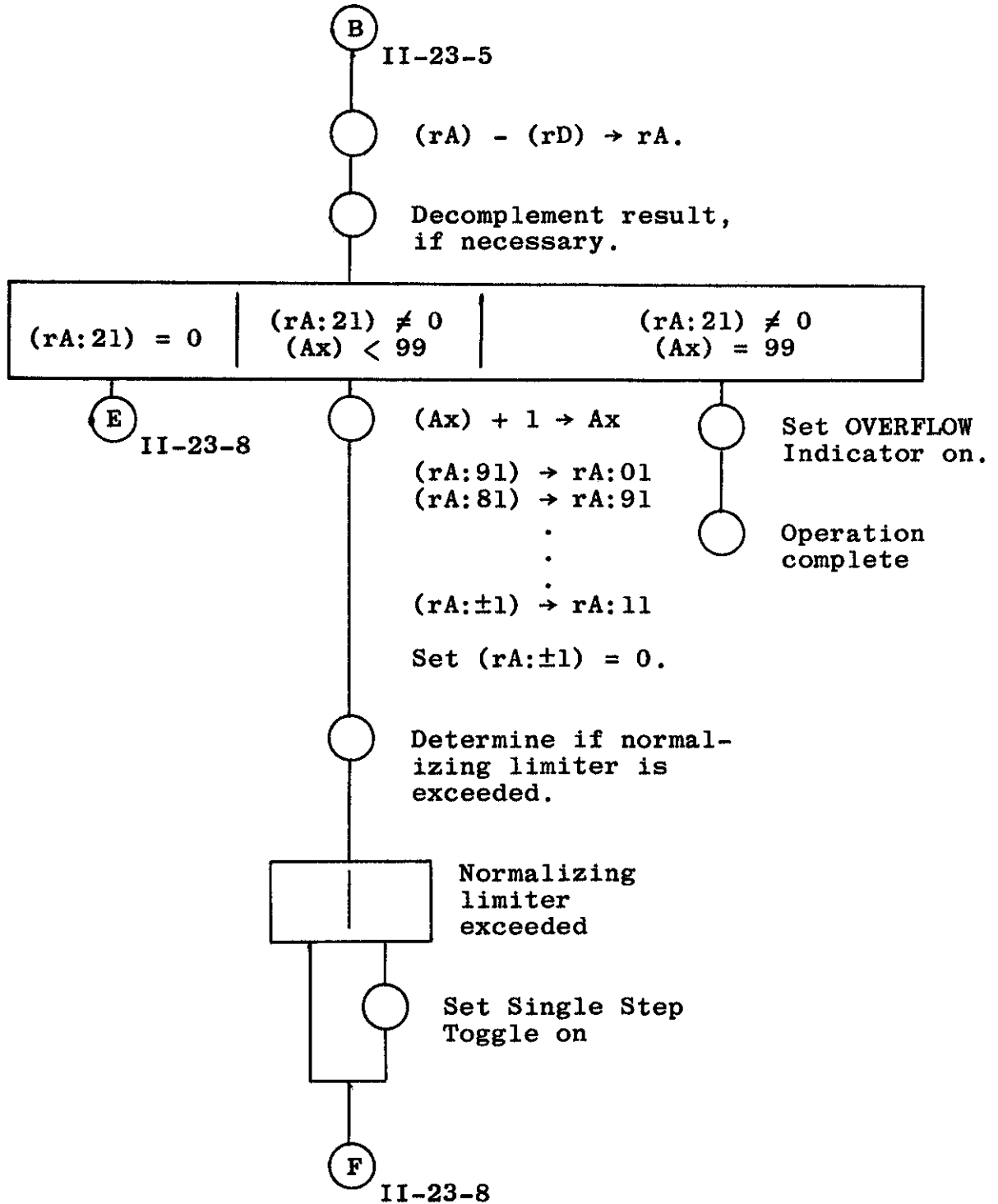
Flow chart:



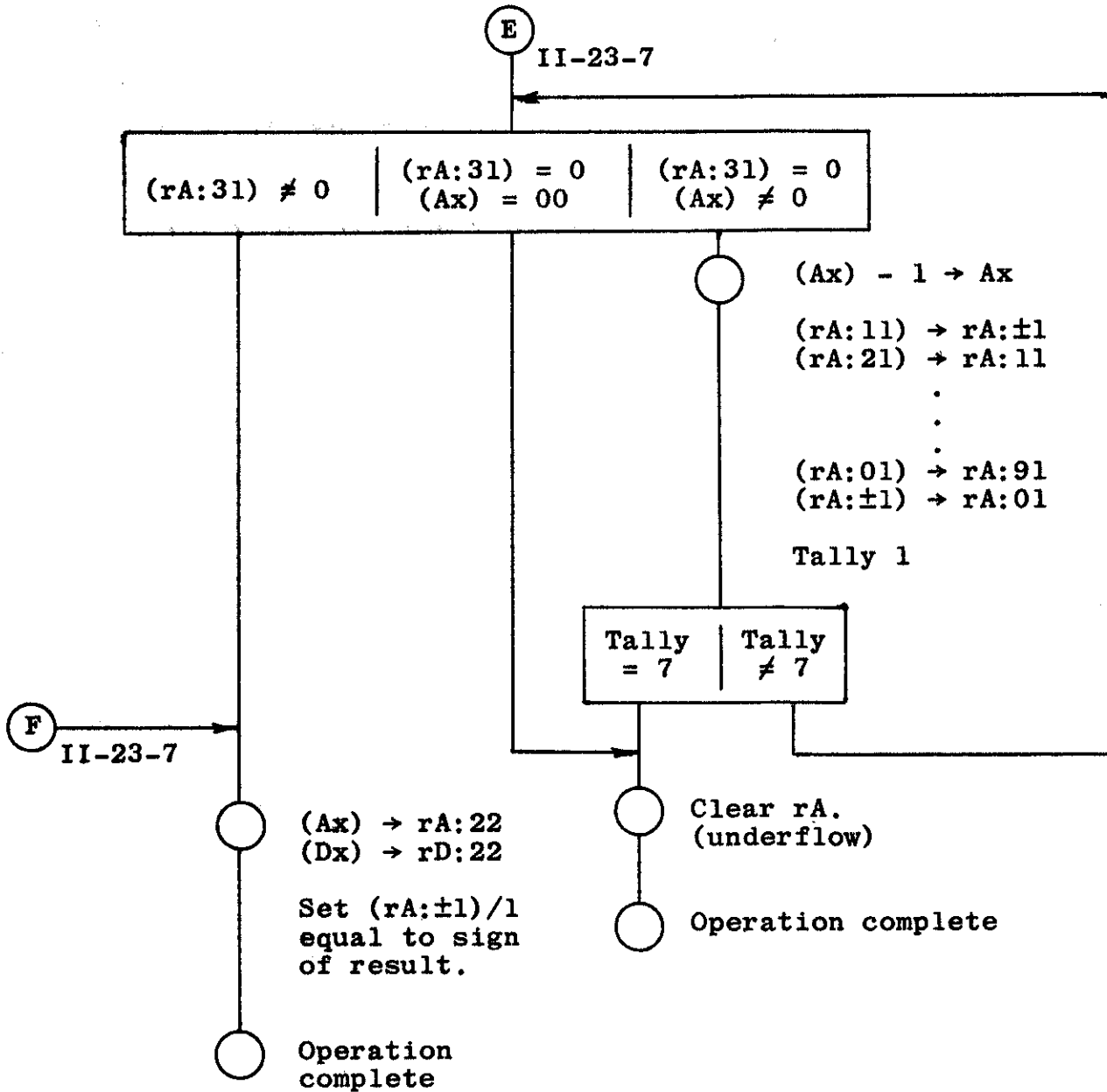
Flow chart (continued):



Flow chart (continued):



Flow chart (continued):



THE DATA PROCESSOR

Register status:

Register name	Contents after execution of FSU.	Contents after execution of FSA.																						
A	$(rA)_b - (B[aaaa])$	$(rA)_b - (B[aaaa]) $																						
R	Unchanged	Unchanged																						
D	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td> <td style="width: 10%;">1</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td> <td style="width: 10%;">1</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> <td style="width: 10%;">0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0														
0	1	0	0	0	0	0	0	0	0	0														
B P	Unchanged $(rP)_b + 1$	Unchanged $(rP)_b + 1$																						
C	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td> <td style="width: 10%;">i</td> <td style="width: 10%;">i</td> <td style="width: 10%;">0</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">B[aaaa]</td> </tr> </table>	0	i	i	0	2	3	B[aaaa]	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td> <td style="width: 10%;">i</td> <td style="width: 10%;">i</td> <td style="width: 10%;">1</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">B[aaaa]</td> </tr> </table>	0	i	i	1	2	3	B[aaaa]								
0	i	i	0	2	3	B[aaaa]																		
0	i	i	1	2	3	B[aaaa]																		
E	B[aaaa]	B[aaaa]																						

Register name	Contents if non-existent-address ALARM STOP occurs.	Contents if OVERFLOW Indicator is on.														
A	Unchanged	Unnormalized difference without exponent.														
R	Unchanged	Unchanged														
D	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">±</td> <td style="width: 10%;">n</td> <td style="width: 10%;">i</td> <td style="width: 10%;">i</td> <td style="width: 10%;">v</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">a</td> <td style="width: 10%;">a</td> <td style="width: 10%;">a</td> <td style="width: 10%;">a</td> </tr> </table>	±	n	i	i	v	2	3	a	a	a	a	Cleared			
±	n	i	i	v	2	3	a	a	a	a						
B P	Unchanged $(rP)_b + 1$	Unchanged $(rP)_b + 1$														
C	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">n</td> <td style="width: 10%;">i</td> <td style="width: 10%;">i</td> <td style="width: 10%;">v</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">B[aaaa]</td> </tr> </table>	n	i	i	v	2	3	B[aaaa]	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">n</td> <td style="width: 10%;">i</td> <td style="width: 10%;">i</td> <td style="width: 10%;">v</td> <td style="width: 10%;">2</td> <td style="width: 10%;">3</td> <td style="width: 10%;">B[aaaa]</td> </tr> </table>	n	i	i	v	2	3	B[aaaa]
n	i	i	v	2	3	B[aaaa]										
n	i	i	v	2	3	B[aaaa]										
E	B[aaaa]	B[aaaa]														

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status (continued):

Register name	Contents if normalization limit ALARM STOP occurs.											
A	Normalized difference											
R	Unchanged											
D	<table border="1" style="margin-left: 20px;"> <tr> <td style="width: 15px;">0</td> <td style="width: 15px;">1</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> <td style="width: 15px;">0</td> </tr> </table>	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0		
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="margin-left: 20px;"> <tr> <td style="width: 15px;">*</td> <td style="width: 15px;">i</td> <td style="width: 15px;">i</td> <td style="width: 15px;">v</td> <td style="width: 15px;">2</td> <td style="width: 15px;">3</td> <td style="width: 15px;">B[aaaa]</td> </tr> </table>	*	i	i	v	2	3	B[aaaa]				
*	i	i	v	2	3	B[aaaa]						
E	B[aaaa]											

* $10 - (m - n)$.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

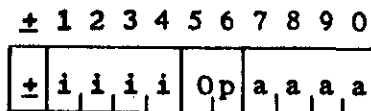
Operation name: FLOATING MULTIPLY

Operation code: 24

Abbreviation: FMU

Instruction format:

Time (μ s):



Product = 0:

fetch:	90
execute:	85
total:	175

Definitions:

Overflow:

fetch:	90
execute:	165
total:	255

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

Underflow:

Minimum:

fetch:	90
execute:	165
total:	255

iiii: not relevant to the execution of this instruction.

Maximum:

fetch:	90
execute:	2825
total:	2915

Op: operation code.

aaaa: address of base of location of multiplicand.

Product \neq 0:

Average:

fetch:	90
execute:	1475
total:	1565

Maximum:

fetch:	90
execute:	2825
total:	2915

Description of operation:

Summary:

This instruction treats the operands like floating-point numbers:

The floating-point product, $(rA) \times (B[aaaa])$, is generated. The two-digit exponent and the eight high-order digits of the product's mantissa replace the contents of the A register. The seven or eight low-order digits of the product's mantissa are inserted in the high-order end of the R register with the three or two low-order digit positions of the R register set to zero. The sign of the product is inserted in the sign-digit position in both the A and R registers.

Flow chart:

See page II-24-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. "Spurious exponent overflow" can occur in case exponent arithmetic--which takes place before the mantissas are multiplied--yields 100 for the adjusted sum of the exponents of (rA) and $(B[aaaa])$: if the adjusted sum of these exponents is 100, the OVERFLOW Indicator is turned on, even though normalization of the product might yield a valid exponent, namely, 99, and the Execute Phase is terminated.

For example, suppose $(rA) = 0\ 75\ 1000\ 0000 = (aaaa)$; the instruction is $0\ 0000\ FMU\ aaaa$. The execution of this instruction should yield $(rA) = 0\ 99\ 1000\ 0000$. Instead, the OVERFLOW Indicator is turned on.

Remarks:

1. Exponent overflow can occur, in which case the OVERFLOW Indicator is turned on.
2. Exponent underflow causes both the A and R registers to be cleared, but is not otherwise indicated.
3. In case the mantissa of either operand is not normalized--in which case at least one of $(rA:31)$ and $(rD:31)$ is 0--it is assumed that the product will be zero. The operation is terminated as soon as this condition is determined to exist--85 μ s are required--with the A and R registers cleared.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

4. In normalizing the product, the A and R registers are shifted together one or two places to the right (see flow chart). Hence, the R register will contain either seven or eight low-order digits of the product's mantissa. It is not possible to predict which will be the case unless one knows the magnitudes of the multiplier and multiplicand.

Register status:

Register name	Contents after execution of FMU.	Contents if non-existent-address ALARM STOP occurs.														
A	Exponent and eight high-order digits of product.	Unchanged														
R	Low order digits of product as described.	Unchanged														
D	$(B[aaaa])^*$	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">4</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	i	i	i	i	2	4	a	a	a	a			
+	i	i	i	i	2	4	a	a	a	a						
B P	Unchanged $(rP)_b + 1$	Unchanged $(rP)_b + 1$														
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">4</td> <td style="text-align: center;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	4	B[aaaa]	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">4</td> <td style="text-align: center;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	4	B[aaaa]
i	i	i	i	2	4	B[aaaa]										
i	i	i	i	2	4	B[aaaa]										
E	$B[aaaa]$	$B[aaaa]$														

* However, $(rD:\pm 1) = 0$; that is, the sign digit of the multiplicand may not be restored.

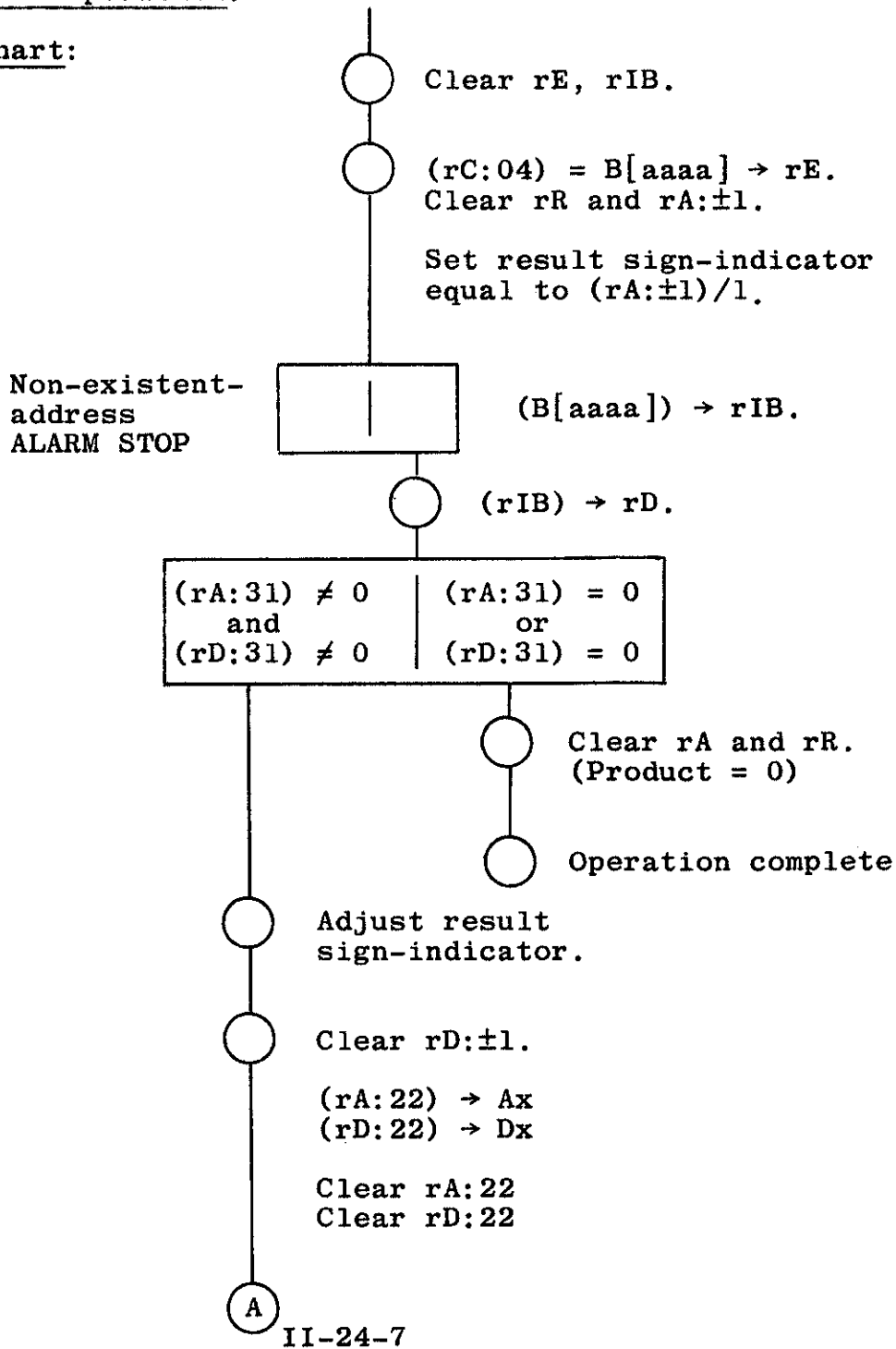
THE DATA PROCESSOR

Register status (continued):

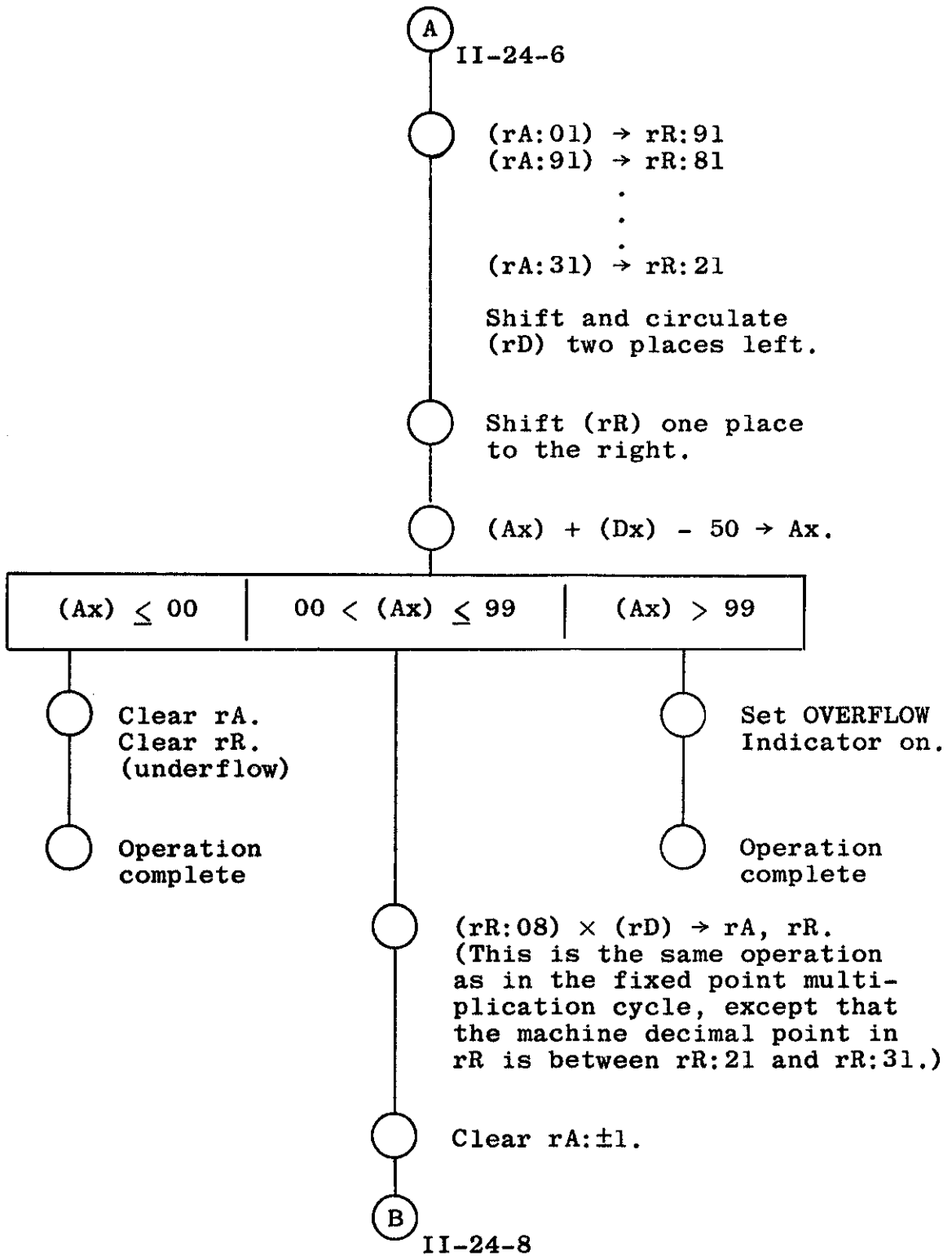
Register name	Contents if (exponent) overflow occurs during exponent arithmetic.	Contents if (exponent) underflow occurs.																												
A R	Cleared (rR:23) = 000; (rR:08) _a = (rR:08) _b .	Cleared Cleared																												
D	(rD:11) = 0; (rD:88) _a = (rD:08) _b ; (rD:02) = 00.	(rD:23) = 000; (rD:86) _a = (rD:06) _b ; (rD:02) = 00.																												
B P	Unchanged (rP) _b + 1	Unchanged (rP) _b + 1																												
C	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B[aaaa]</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </table>	i	i	i	i	2	4	B[aaaa]								<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B[aaaa]</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </table>	i	i	i	i	2	4	B[aaaa]							
i	i	i	i	2	4	B[aaaa]																								
i	i	i	i	2	4	B[aaaa]																								
E	B[aaaa]	B[aaaa]																												

Description of operation:

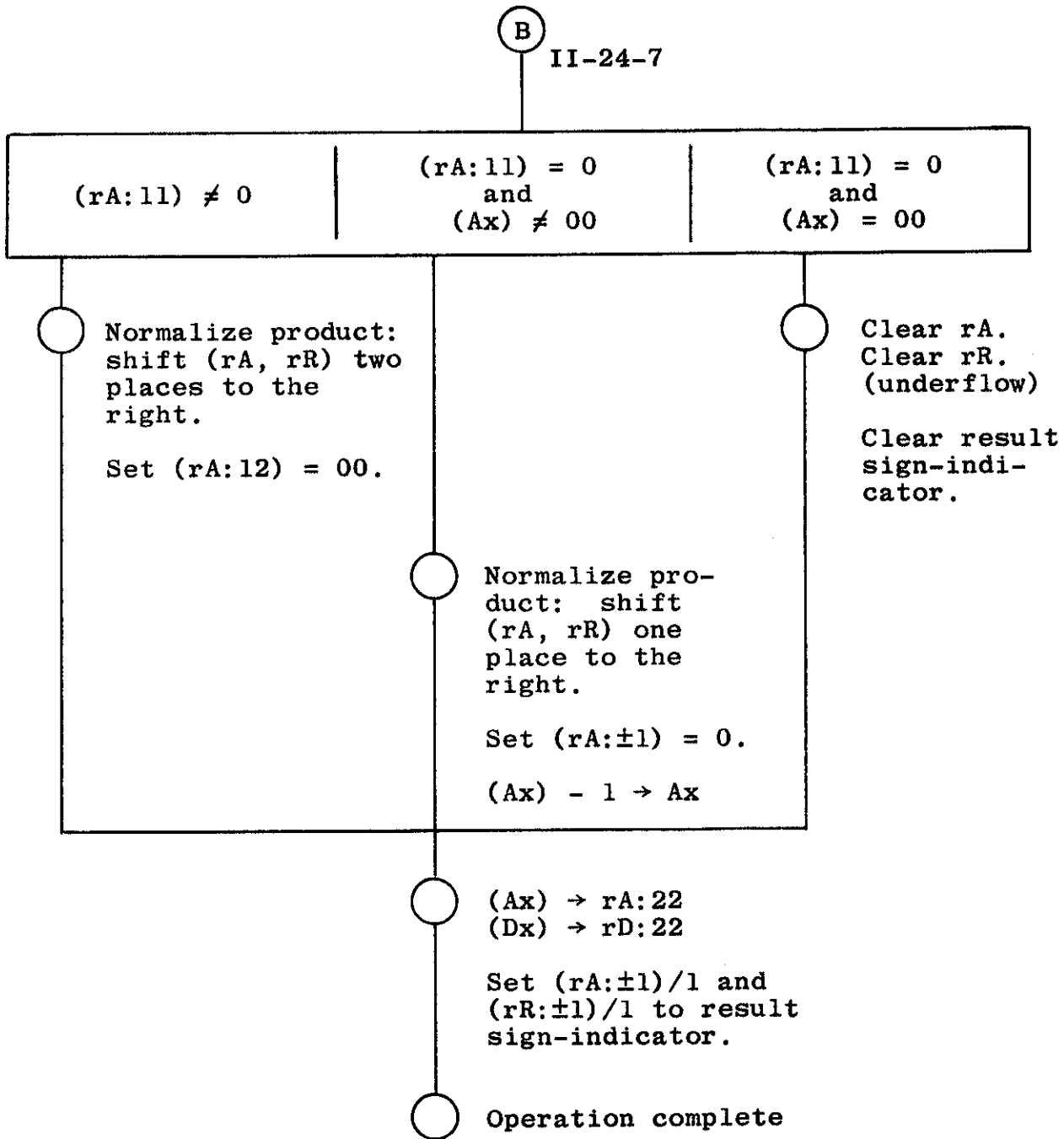
Flow chart:



Flow chart (continued):



Flow chart (continued):



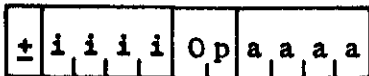
Operation name: FLOATING DIVIDE

Operation code: 25

Abbreviation: FDV

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

aaaa: address of base of location of divisor.

Time (μs):

Quotient or divisor = 0:

fetch: 90
 execute: 85
 total: 175

Underflow:

fetch: 90
 execute: 170
 total: 260

Overflow:

Minimum:

fetch: 90
 execute: 170
 total: 260

Maximum:

fetch: 90
 execute: 6685
 total: 6775

Quotient ≠ 0.

Average:

fetch: 90
 execute: 3985
 total: 4075

Maximum:

fetch: 90
 execute: 6685
 total: 6775

Description of operation:

Summary:

This instruction treats the operands like floating-point numbers:

Dividend: Exponent is (rA:22); high-order digits of mantissa are (rA:08), low-order digits of mantissa are (rR:88).

Divisor: (B[aaaa]).

Quotient: exponent is (rA:22); high-order digits of mantissa are (rA:08); low-order digit(s) of mantissa is (are) (rR:11) [(rR:22)].

Remainder: the low-order digit positions of the R register.

Flow chart:

See page II-25-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. "Spurious exponent underflow" can occur in case exponent arithmetic--which takes place before the mantissas are divided--yields the equivalent of -01 for the adjusted difference of the exponents of (rA) and (B[aaaa]): if the adjusted difference of these exponents is -01, the A and R registers are cleared, and the Execute Phase is terminated, even though normalization of the quotient might yield a valid exponent, namely, 00.

For example, suppose (rA) = 0 00 1000 0000 and (aaaa) = 0 51 1000 0000; the instruction is 0 0000 FDV aaaa. The execution of this instruction should yield (rA) = 0 00 1000 0000 (since the divisor is unity). Instead, the A and R registers are cleared.

Remarks:

1. Exponent overflow can occur, in which case the OVERFLOW Indicator is turned on.
2. Exponent underflow causes both the A and R registers to be cleared, but is not otherwise indicated.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

3. In case the mantissa of the dividend is not normalized--in which case (rA:31) = 0--but the mantissa of the divisor is normalized--in which case (rD:31) ≠ 0--it is assumed that the quotient will be zero. The operation is terminated as soon as this condition is determined to exist--85 μs are required--with the A and R registers cleared.

4. In case the divisor is not normalized--in which case (rD:31) = 0--it is assumed that the divisor is zero: the operation is terminated as soon as this condition is determined to exist--85 μs are required--with the OVERFLOW Indicator turned on.

Register status:

Register name	Contents after execution of FDV.	Contents after non-existent-address ALARM STOP.														
A	Quotient	Unchanged														
R	"Remainder"	Unchanged														
D	(rD:23) = 000; (rD:08) = (B[aaaa]:08)	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">5</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	i	i	i	i	2	5	a	a	a	a			
+	i	i	i	i	2	5	a	a	a	a						
B	Unchanged	Unchanged														
P	(rP) _b + 1	(rP) _b + 1														
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">5</td> <td style="text-align: center;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	5	B[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">2</td> <td style="text-align: center;">5</td> <td style="text-align: center;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	5	B[aaaa]
i	i	i	i	2	5	B[aaaa]										
i	i	i	i	2	5	B[aaaa]										
E	B[aaaa]	B[aaaa]														

THE DATA PROCESSOR

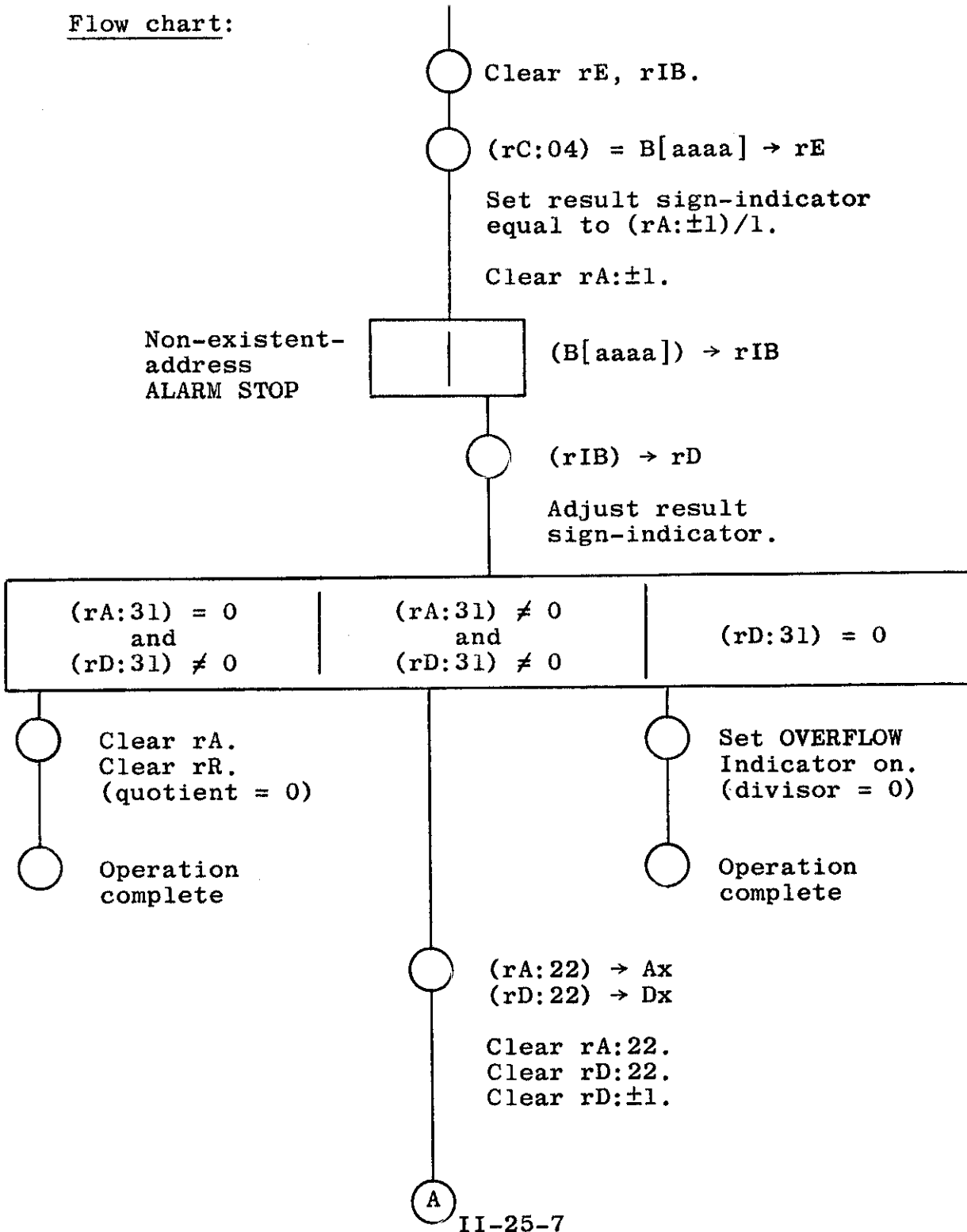
Register status (continued):

Register name	Contents if underflow occurs after exponent arithmetic.	Contents if dividend = 0, divisor \neq 0.														
A	Cleared	Cleared														
R	Unchanged	Unchanged														
D	(rD:23) = 000 (rD:08) = (B[aaaa]:08)	(B[aaaa])														
B	Unchanged	Unchanged														
P	(rP) _b + 1	(rP) _b + 1														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td><td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	5	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td><td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	5	B[aaaa]
i	i	i	i	2	5	B[aaaa]										
i	i	i	i	2	5	B[aaaa]										
E	B[aaaa]	B[aaaa]														

Register name	Contents if divisor = 0.							
A	(rA:±1) = 0 (rA:00) _a = (rA:00) _b							
R	Unchanged							
D	B[aaaa]							
B	Unchanged							
P	(rP) _b + 1							
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td><td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td><td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	i	i	i	i	2	5	B[aaaa]
i	i	i	i	2	5	B[aaaa]		
E	B[aaaa]							

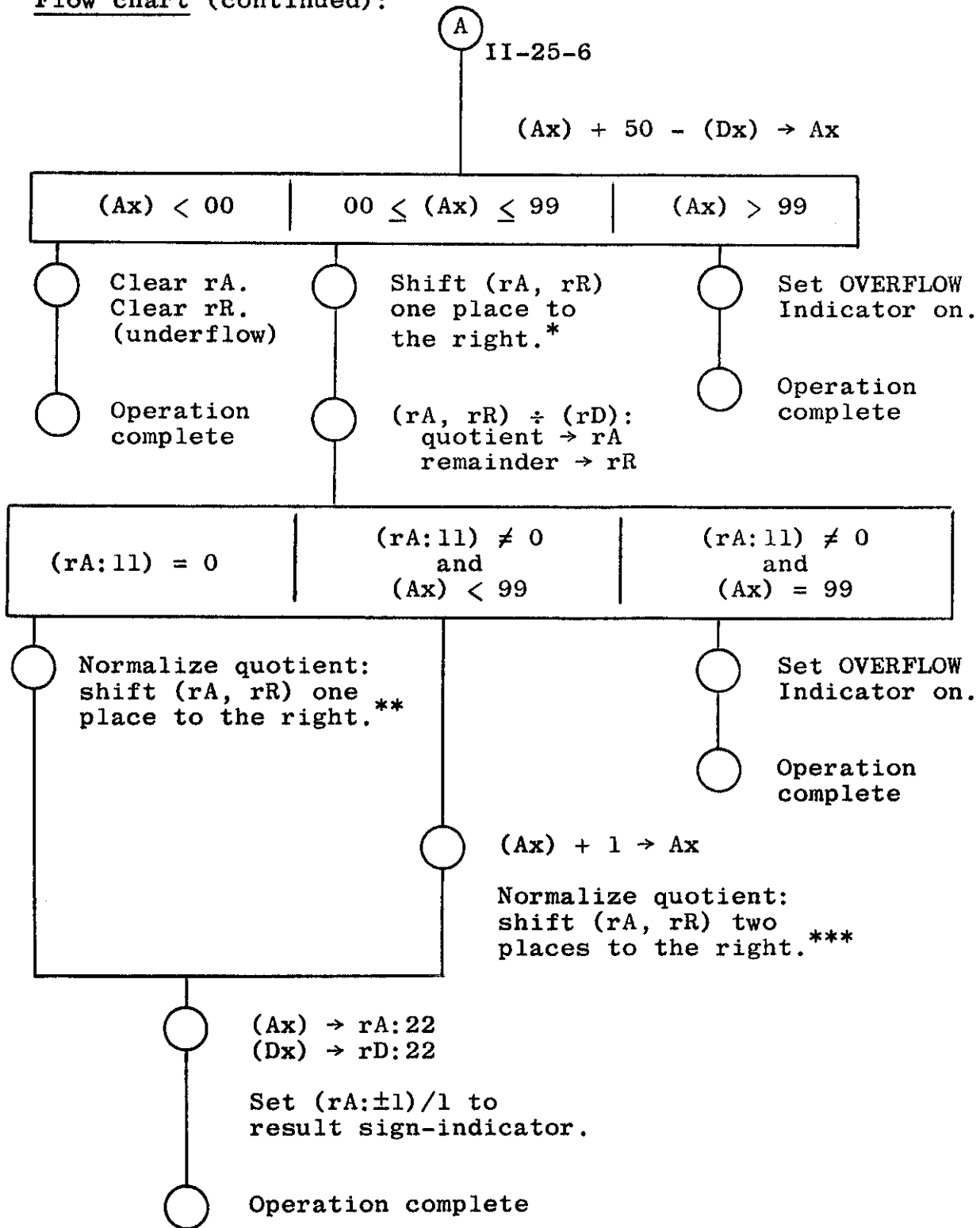
Description of operation:

Flow chart:



II-25-7

Flow chart (continued):



* Exponent adjustment is made at the end of the operation if required.

** Then (rR:11) is the low-order digit of the quotient.

*** Then (rR:22) are the two low-order digits of the quotient.

Operation name: INCREASE FIELD LOCATION

Operation Code: 26

Abbreviation: IFL

Instruction format:

Time (μ s):

fetch: 90
 execute: 160
 total: 250

\pm	1	2	3	4	5	6	7	8	9	0
\pm	s	L	n	n	Op	a	a	a	a	

Definitions:

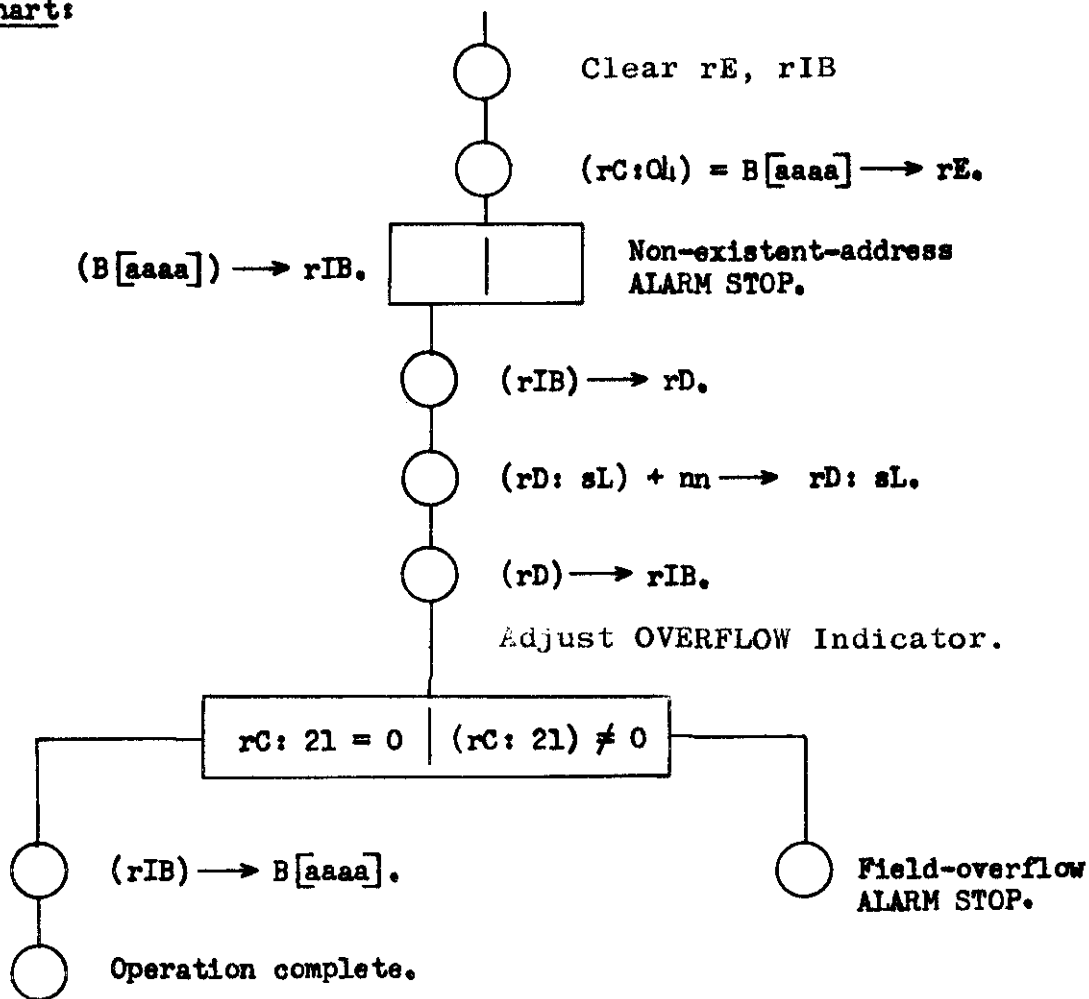
- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- s: partial-word designator:
 s designates the position, within the word, of the low-order digit of the partial-word operand.
- L: partial-word designator:
 L specifies the number of digits in the partial-word operand.
- nn: modifier for the partial-word operand.
- Op: operation code.
- aaaa: address of base of location of partial-word addend.

Description of operation:

Summary:

Increase $(B[aaaa]:sL)$ by nn . If overflow occurs, set the OVERFLOW Indicator to "on".

Flow chart:



Note: as each digit of the partial-word sum is generated, L (i.e., $(rC:21)$) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the addition $(rC:21)$ will be different from zero, and field overflow will be detected. See Example 6, page II-26-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP.

Remarks:

1. If the sign-digit position of (B [aaaa]) is included in the partial-word field specified by sL, it does not have sign significance: the sign-digit position has numeric significance. See Examples 4,5, and 6, page II-26-6.

Register status:

Register name	Contents after execution of IFL,	Contents if non-existent-address ALARM STOP occurs.																
A	Unchanged	Unchanged																
R	"	"																
D	$(B[aaaa])_a$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>6</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	n	n	2	6	a	a	a	a					
±	s	L	n	n	2	6	a	a	a	a								
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>n</td><td>n</td><td>2</td><td>6</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	0	n	n	2	6	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>6</td><td>B</td><td>[aaaa]</td> </tr> </table>	s	L	n	n	2	6	B	[aaaa]
0	0	n	n	2	6	B	[aaaa]											
s	L	n	n	2	6	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents if field-overflow ALARM STOP occurs.								
A	Unchanged								
R	"								
D	$(B[aaaa])_b$, as modified.								
B	Unchanged								
P	$(rP)_b + 1$								
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>*</td><td>n</td><td>n</td><td>2</td><td>6</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	*	n	n	2	6	B	[aaaa]
0	*	n	n	2	6	B	[aaaa]		
E	$B[aaaa]$								

* L-S-1

Examples:

1. (rC) = 0202 26 0396
 (0396)_b = + 0000 00 0012
 (rD)_a = (0396)_a = + 0000 00 0014

2. (rC) = 6314 26 0416
 (0416)_b = - 2973 43 9216
 (rD)_a = (0416)_a = - 2973 57 9216

3. (rC) = 6220 26 0534
 (0534)_b = + 0002 90 2400
 (rD)_a = (0534)_a = + 0002 10 2400

OVERFLOW Indicator is set "on".

4. (rC) = 3412 26 0600
 (0600)_b = 1 2490 00 9000
 (rD)_a = (0600)_a = 1 2610 00 9000

5. (rC) = 1232 26 0657
 (0657)_b = 9 5236 47 8888
 (rD)_a = (0657)_a = 2 7236 47 8888

OVERFLOW Indicator is set "on".

6. (rC) = 2530 26 0900
 (0900)_b = 0 8429 90 5432
 (rD)_a = 1 1429 90 5432

Field-overflow ALARM STOP.

(rC)_a = 0230 26 0900

Examples (continued):

$$\begin{aligned}
 7. \quad (rC) &= 0104 \ 26 \ 0963 \\
 (0963)_b &= - \ 1123 \ 87 \ 000\underline{3} \\
 (rD)a = (0963)a &= - \ 1123 \ 87 \ 000\underline{7}
 \end{aligned}$$

$$\begin{aligned}
 8. \quad (rC) &= 6122 \ 26 \ 2440 \\
 (2440)_b &= + \ 0000 \ 8\underline{1} \ 0000 \\
 (rD)a = (2440)a &= + \ 0000 \ 8\underline{3} \ 0000
 \end{aligned}$$

$$\begin{aligned}
 9. \quad (rC) &= 4115 \ 26 \ 4000 \\
 (4000)_b &= - \ 112\underline{5} \ 99 \ 9999 \\
 (rD)a = (4000)a &= - \ 112\underline{0} \ 99 \ 9999
 \end{aligned}$$

OVERFLOW Indicator is set "on".

Operation code: 27

Operation name: DECREASE FIELD LOCATION

Abbreviation: DFL

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

\pm	s	L	n	n	Op	a	a	a	a
-------	---	---	---	---	----	---	---	---	---

fetch: 90

execute: 160

total: 250

Definitions:

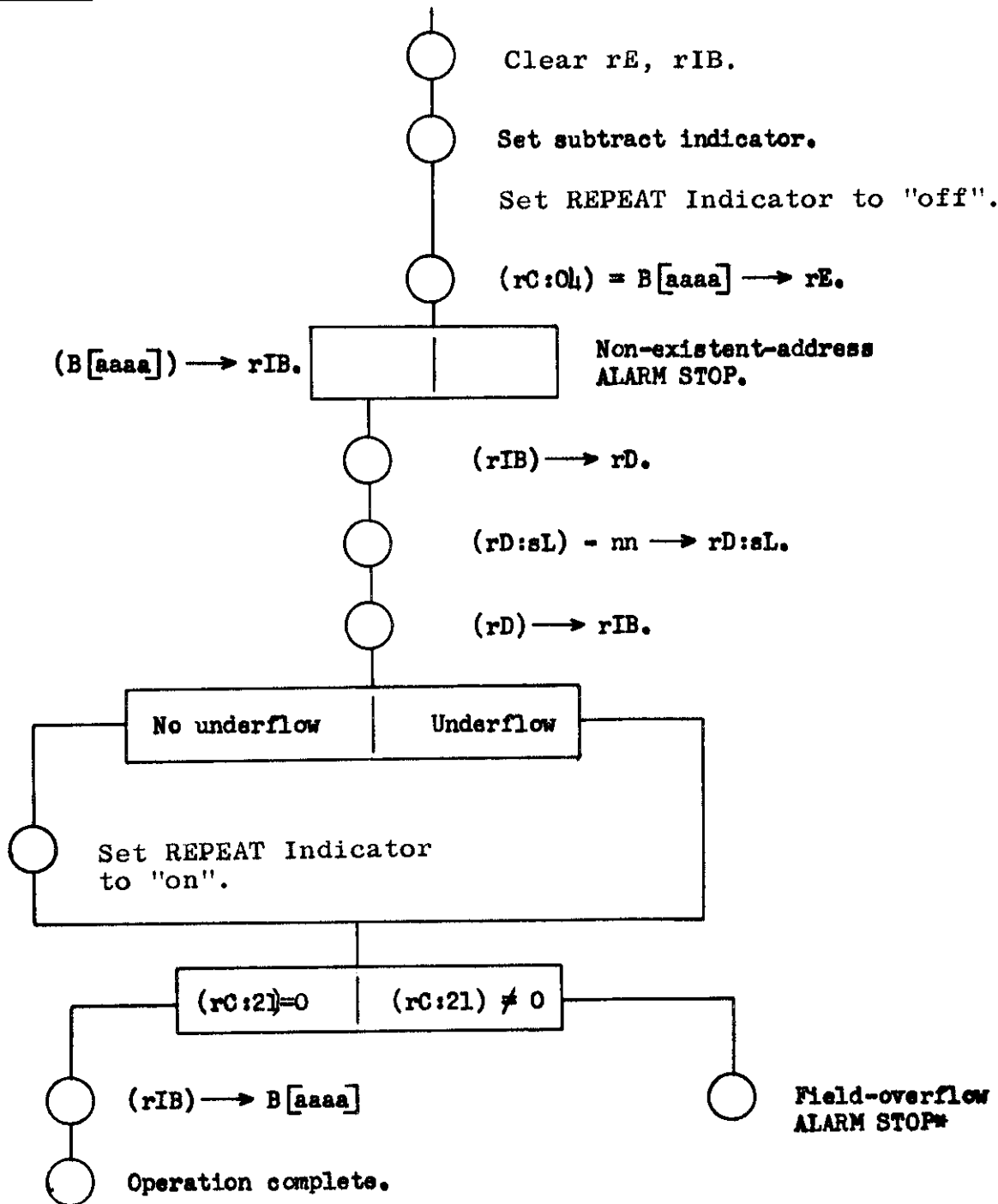
- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- s: partial-word designator:
 - s designates the position, within the word, of the low-order digit of the partial-word operand.
- L: partial word designator:
 - L specifies the number of digits in the partial word operand.
- nn: modifier for the partial-word operand.
- Op: operation code.
- aaaa: address of base of location of partial-word minuend.

Description of operation:

Summary:

Decrease (B[aaaa]: sL) by nn. If underflow occurs, set the REPEAT Indicator to "off"; if not, set the REPEAT Indicator to "on".

Flow chart:



* See NOTE, top of page II-27-4.

NOTE: as each digit of the partial-word difference is generated, L (i.e., (rC:21)) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the subtraction (rC:21) will be different from zero, and field overflow will be detected. See Examples 6, page II-27-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP.

Remarks:

1. If the sign-digit position of (B [aaaa]) is included in the partial-word field specified by sL, it does not have sign significance: the sign-digit position has numeric significance. See Examples 5 and 6, page II-27-6.
2. It is not necessary to follow this instruction immediately with a BRANCH, REPEAT instruction. See page II-32-2.

THE COMPUTER

Register status:

Register name	Contents after execution of DFL.	Contents if non-existent-address ALARM STOP occurs.																
A	Unchanged	Unchanged																
R	Unchanged	Unchanged																
D	$(B[aaaa])_a$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>7</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	n	n	2	7	a	a	a	a					
±	s	L	n	n	2	7	a	a	a	a								
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>n</td><td>n</td><td>2</td><td>7</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	0	n	n	2	7	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>7</td><td>B</td><td>[aaaa]</td> </tr> </table>	s	L	n	n	2	7	B	[aaaa]
0	0	n	n	2	7	B	[aaaa]											
s	L	n	n	2	7	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents if field-overflow ALARM STOP occurs.								
A	Unchanged								
R	Unchanged								
D	$(B[aaaa])_b$, as modified								
B	Unchanged								
P	$(rP)_b + 1$								
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>*</td><td>n</td><td>n</td><td>2</td><td>7</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	*	n	n	2	7	B	[aaaa]
0	*	n	n	2	7	B	[aaaa]		
E	$B[aaaa]$								

* L-s-1

Examples:

1. (rC) = 0202 27 0396

(0396)_b = + 0000 00 0012

(rD)_a = (0396)_a = + 0000 00 0010

REPEAT Indicator is set "on".

2. (rC) = 6314 27 0416

(0416)_b = - 2973 43 9216

(rD)_a = (0416)_a = - 2973 29 9216

REPEAT Indicator is set "on".

3. (rC) = 6220 27 0534

(0534)_b = + 0002 10 2400

(rD)_a = (0534)_a = + 0002 90 2400

REPEAT Indicator is set "off".

4. (rC) = 3412 27 0600

(0600)_b = 1 2490 00 9000

(rD)_a = (0600)_a = 1 2370 00 9000

REPEAT Indicator is set "on".

5. (rC) = 1232 27 0657

(0657)_b = 0 5236 47 8888

(rD)_a = (0657)_a = 7 3236 47 8888

REPEAT Indicator is set "off".

Examples (continued):

6. (rC) = 2530 27 0900

(0900)_b = 0 8429 90 5432

(rD) = 0 5429 90 5432

REPEAT Indicator is set "on".
Field-overflow ALARM STOP.

(rC)_a = 0230 27 0900

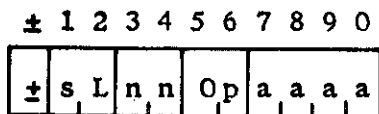
Operation code: 28

Operation name: DECREASE FIELD LOCATION, LOAD B

Abbreviation: DLB

Instruction format:

Time (μs):



fetch: 90
 execute: 160
 total: 250

Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

- s: partial-word designator:
 s designates the location, within the word, of the low-order digit of the partial-word operand.

- L: partial-word designator:
 L specifies the number of digits in the partial-word operand.

- nn: modifier for the partial-word operand.

- Op: operation code.

- aaaa: address of base of location of partial-word operand.

Description of operation:

Summary:

Execute DFL, and, in addition, load rB with the modified partial-word field. That is, decrease (B[aaaa]:sL) by nn. If underflow occurs, set the REPEAT Indicator to "off"; if not, set the REPEAT Indicator to "on". Then load rB with the modified partial-word field.

Flow chart:

See page II-28-4.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP.

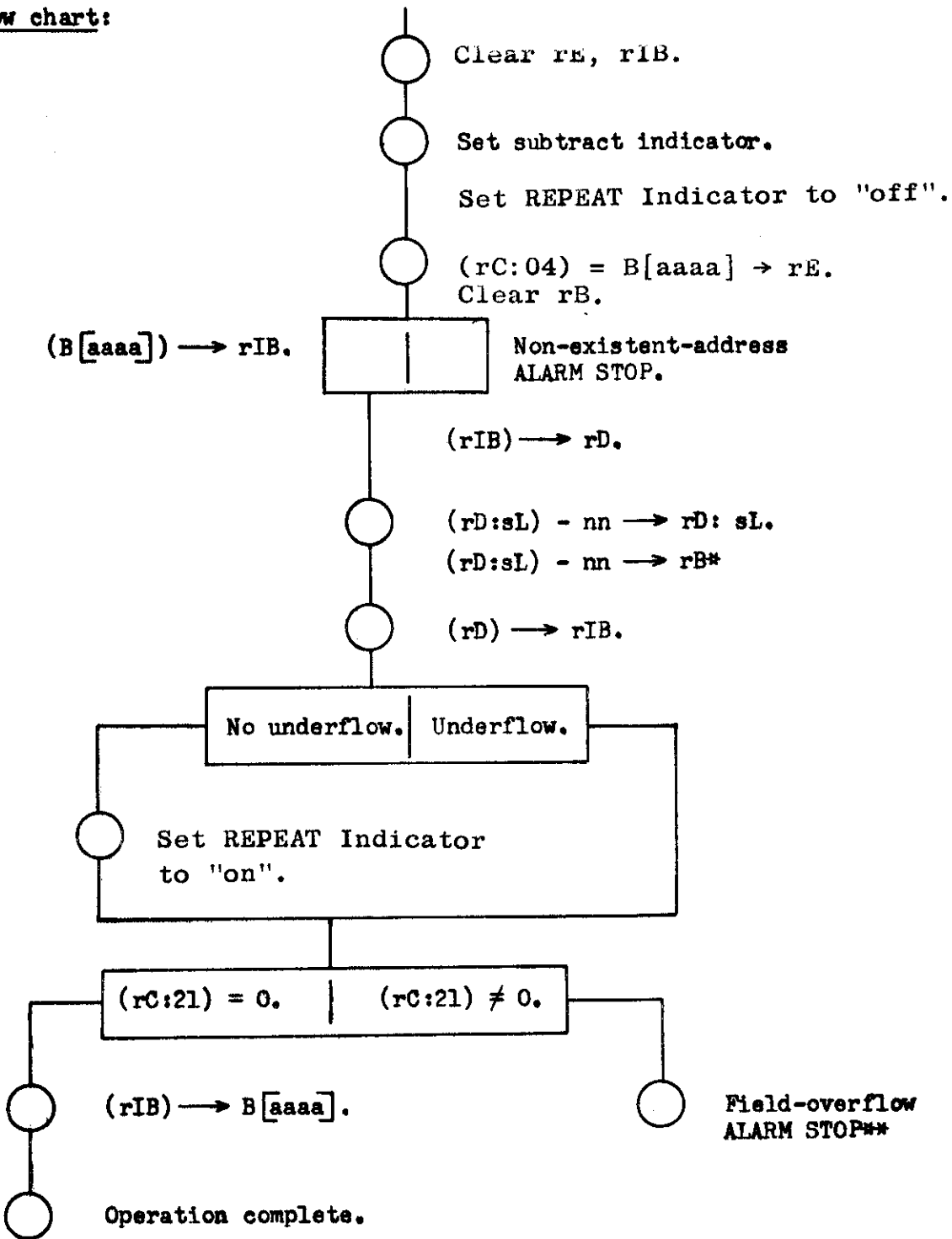
Remarks:

1. If the sign-digit position of (B[aaaa]) is included in the partial-word field specified by sL, it does not have sign significance: the sign-digit position has numeric significance. See Examples 5 and 6, page II-27-6.

2. The Examples for DFL will suffice to explain a large part of DLB. The reader is referred to page II-27-6. Light is shed on the mechanism of the "load B" part of the operation by the Examples on page II-28-6.

Description of operation:

Flow chart:



* If $L < 4$, the L high-order digits of $(rD:sL) - nn$ replace the L high-order digits of rB ; if $L = 4$, the four high-order digits of $(rD:sL) - nn$ replace all of rB .

** See Note at top of next page.

NOTE: As each digit of the partial-word difference is generated, L (i.e., (rC:21)) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the subtraction (rC:21) will be different from zero, and field overflow will be detected. See Examples 5 and 6, pages II-28-7.

Register status:

Register name	Contents after execution of DLB.	Contents if non-existent-address ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	"	"														
D	$(B[aaaa])_a$	<table border="1" style="display: inline-table;"> <tr> <td>±</td><td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>8</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	n	n	2	8	a	a	a	a			
±	s	L	n	n	2	8	a	a	a	a						
B	See description of operation.	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>0</td><td>n</td><td>n</td><td>2</td><td>8</td><td>B[aaaa]</td> </tr> </table>	0	0	n	n	2	8	B[aaaa]	<table border="1" style="display: inline-table;"> <tr> <td>s</td><td>L</td><td>n</td><td>n</td><td>2</td><td>8</td><td>B[aaaa]</td> </tr> </table>	s	L	n	n	2	8	B[aaaa]
0	0	n	n	2	8	B[aaaa]										
s	L	n	n	2	8	B[aaaa]										
E	$B[aaaa]$	$B[aaaa]$														

Register name	Contents if field-overflow ALARM STOP occurs.							
A	Unchanged							
R	Unchanged							
D	$(B[aaaa])_b$, as modified.							
B	See description of operation.							
P	$(rP)_b + 1$							
C	<table border="1" style="display: inline-table;"> <tr> <td>0</td><td>*</td><td>n</td><td>n</td><td>2</td><td>8</td><td>B[aaaa]</td> </tr> </table>	0	*	n	n	2	8	B[aaaa]
0	*	n	n	2	8	B[aaaa]		
E	$B[aaaa]$							

* L-s-1

Examples:

NOTE: See also Examples, page II-27-6, which elucidate DFL.

1. (rC) = 0402 28 0496
- (0496)_b = + 1223 49 0148
- (rD)_a = (0496)_a = + 1223 49 0146
- (rB)_a = 0146

REPEAT Indicator is set "on."

2. (rC) = 8205 28 0516
- (0516)_b = - 3946 25 2014
- (rD)_a = (0516)_a = - 3946 25 1514
- (rB)_a = 1500

REPEAT Indicator is set "on."

3. (rC) = 3310 28 0634
- (0634)_b = + 0050 40 2222
- (rD)_a = (0634)_a = + 9950 40 2222
- (rB)_a = 9950

REPEAT Indicator is set "off."

4. (rC) = 6650 28 0700
- (0700)_b = - 1255 00 9753
- (rD)_a = (0700)_a = - 1254 50 9753
- (rB)_a = 1254

REPEAT Indicator is set "on."

Examples (continued):

5. (rC) = 2420 28 0790
 (0790)_b = 1 3540 44 2345
 (rD)_a = 1 1540 44 2345
 (rB)_a = 1 150

REPEAT Indicator is set "on".
 Field-overflow ALARM STOP.

6. (rC) = 1310 28 2040
 (2040)_b = 0 5998 74 0000
 (rD)_a = 9 5998 74 0000
 (rB)_a = 9 500

REPEAT Indicator is set "off".
 Field-overflow ALARM STOP.

Operation name: RECORD TRANSFER

Operation code: 29

Abbreviation: RTF

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

\pm	i	n	n	i	Op	a	a	a	a
-------	---	---	---	---	----	---	---	---	---

fetch: 90

execute: $\frac{50 + 60nn}{}$

total: $140 + 60nn$

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

i: not relevant to the execution of this instruction.

nn: specifies the number of words to be relocated: if nn = 00, 100 words will be relocated.

Op: operation code.

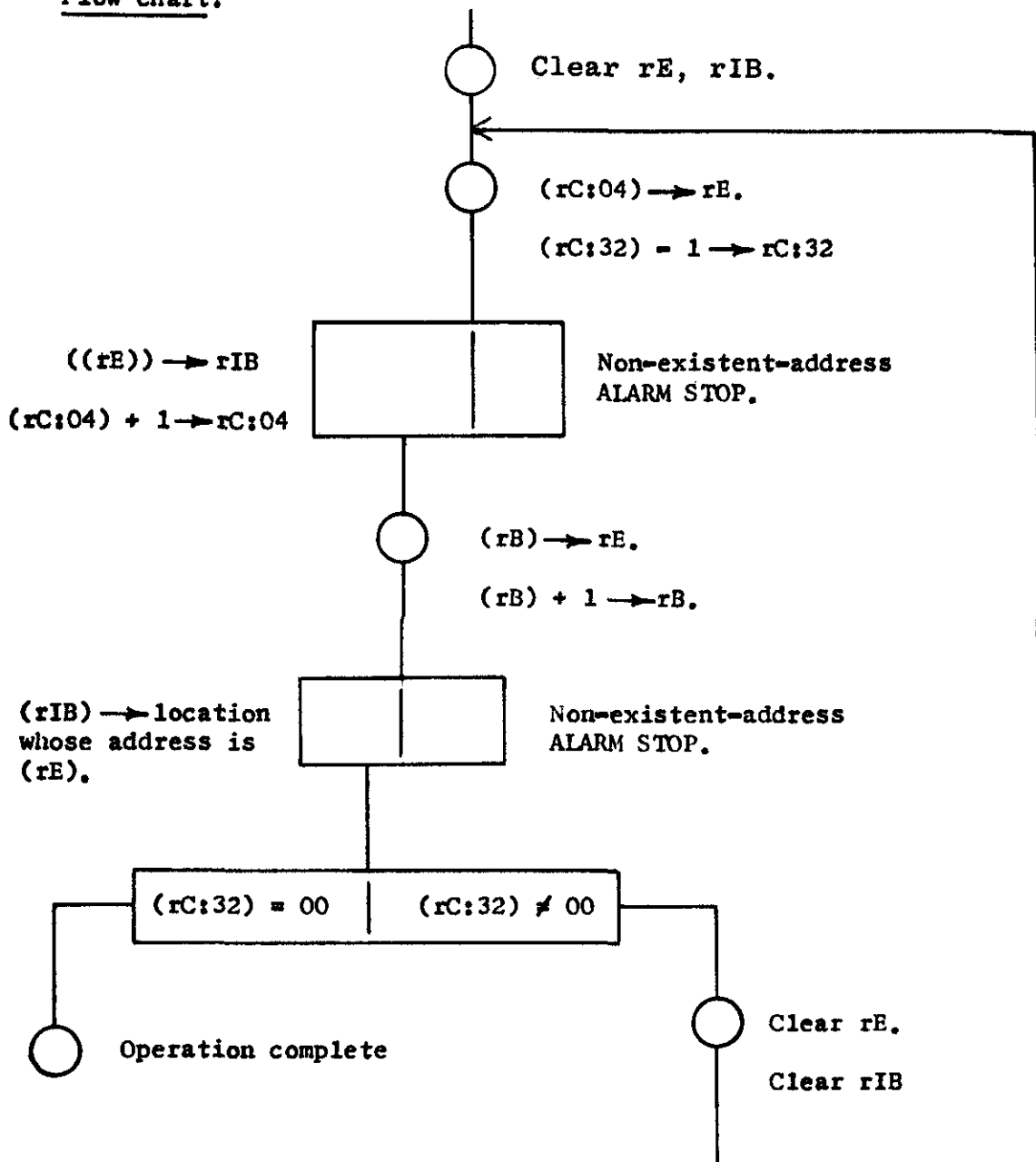
aaaa: address of base of location of first word to be relocated.

Description of operation:

Summary:

Relocate the contents of nn consecutively-addressed locations, beginning with the one whose address is $B[aaaa]$. Transfer the specified words, in succession, and one at a time, to the nn consecutively-addressed locations, beginning with the one whose address is in the B register.

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

1. After the execution of a RECORD TRANSFER operation, the B register will contain the sum of the address of the last location filled plus 1, that is, the address of "the next location to be filled."

Register status:

Register name	Contents after execution of RTF.	Contents after non-existent-address ALARM STOP.																						
A	Unchanged	Unchanged																						
R	"	"																						
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>i</td><td>n</td><td>n</td><td>i</td><td>2</td><td>9</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	i	n	n	i	2	9	a	a	a	a	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>i</td><td>n</td><td>n</td><td>i</td><td>2</td><td>9</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	i	n	n	i	2	9	a	a	a	a
±	i	n	n	i	2	9	a	a	a	a														
±	i	n	n	i	2	9	a	a	a	a														
B	$(rB)_b + nn^*$	Indeterminate***																						
P	$(rP)_b + 1$	$(rP)_b + 1$																						
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>0</td><td>0</td><td>i</td><td>2</td><td>9</td><td>B</td><td>[aaaa]</td><td>+ nn**</td> </tr> </table>	i	0	0	i	2	9	B	[aaaa]	+ nn**	Indeterminate***													
i	0	0	i	2	9	B	[aaaa]	+ nn**																
E	$(rB)_b + nn - 1^*$	Indeterminate***																						

Note: if $nn = 00$,
 $*(rB)_b + nn = (rB)_b + 100$
 $(rB)_b + nn - 1 = (rB)_b + 99$
 $**B[aaaa] + nn = B[aaaa] + 100$

***See description of operation

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 00

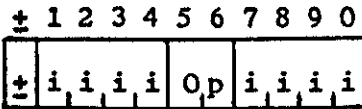
Operation name: HALT

Abbreviation: HLT

Time (μ s):

Instruction format:

fetch: 90
execute: 10
total: 100



Definitions:

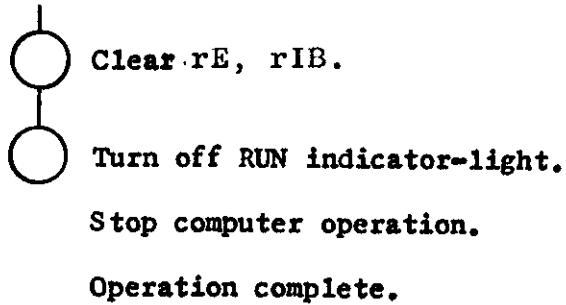
- \pm : if \pm is odd, B-register address modification will occur; otherwise, there will be no such modification.
- iiii: not relevant to the execution of this instruction.
- Op: operation code.

Description of operation:

Summary:

The computer stops, ready to fetch the next instruction in sequence. The RUN indicator-light is off.

Flow chart:



Exceptional conditions: NONE.

Remarks:

1. Although the address field is not used for addressing purposes, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.

2. If the computer senses an unassigned operation code at the beginning of the Execute Phase, an ALARM STOP will occur: the computer will stop with the COMPUTER PROGRAM alarm-indicator light on.

Register status:

Register name	Contents after execution of HALT.													
A	Unchanged													
R	"													
D	<table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> </tr> </table>	+	i	i	i	i	0	0	i	i	i	i		
+	i	i	i	i	0	0	i	i	i	i				
B	Unchanged													
P	$(rP)_b + 1$													
C	<table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">]</td> </tr> </table>	i	i	i	i	0	0	B	[i	i	i	i]
i	i	i	i	0	0	B	[i	i	i	i]		
E	Cleared													

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 01

Operation name: NO OPERATION

Abbreviation: NOP

Time (μ s):

Instruction format:

fetch: 90
execute: 10
total: 100

\pm 1 2 3 4 5 6 7 8 9 0



Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

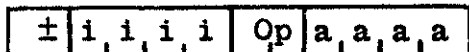
Operation name: BRANCH, UNCONDITIONALLY

Operation code: 30

Instruction format:

Abbreviation: BUN

± 1 2 3 4 5 6 7 8 9 0



Time (μs):

fetch: 90
 execute: 35
 total: 125

Definitions:

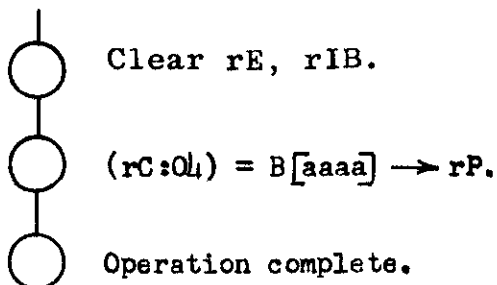
- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iiii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location of next instruction.

Description of operation:

Summary:

Transfer control to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa].

Flow chart:



Exceptional conditions: NONE.

Remarks:

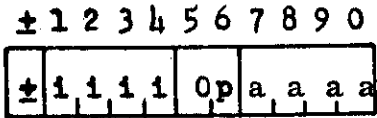
Register status:

Register name	Contents after execution of BUN													
A	Unchanged													
R	"													
D	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px;">+</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	+	1	1	1	1	3	0	a	a	a	a		
+	1	1	1	1	3	0	a	a	a	a				
B	Unchanged													
P	B [aaaa]													
C	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">]</td> </tr> </table>	1	1	1	1	3	0	B	[a	a	a	a]
1	1	1	1	3	0	B	[a	a	a	a]		
E	Cleared													

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: BRANCH, OVERFLOW

Instruction format:



Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

- iiii: not relevant to the execution of this instruction.

- Op: operation code.

- aaaa: address of base of location of alternate instruction.

Operation code: 31

Abbreviation: BOF

Time (µs):

No branch:

fetch:	90
execute:	<u>15</u>
total:	105

Branch:

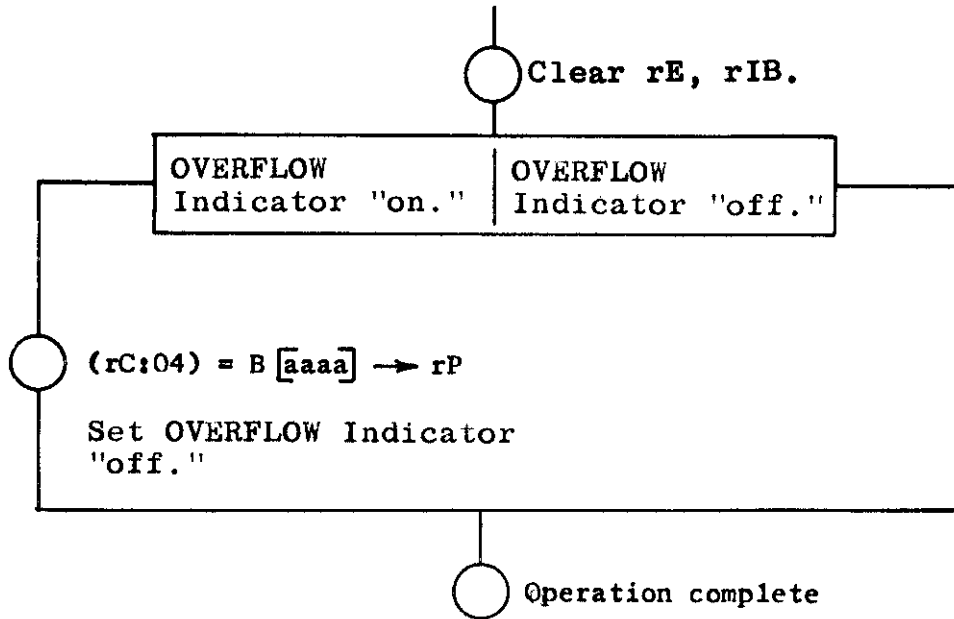
fetch:	90
execute:	<u>35</u>
total:	125

Description of operation:

Summary:

If the OVERFLOW Indicator is "on", transfer control to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. If the OVERFLOW Indicator is "off", control continues in sequence.

Flow chart:



Exceptional conditions: NONE.

Remarks:

- The OVERFLOW Indicator may be set "on" by the following operations.

Code	Name	Code	Name
08	KEYBOARD ADD	22	FLOATING ADD
12	ADD	22	FLOATING ADD ABSOLUTE
12	ADD ABSOLUTE	23	FLOATING SUBTRACT
13	SUBTRACT	23	FLOATING SUBTRACT ABSOLUTE
13	SUBTRACT ABSOLUTE	24	FLOATING MULTIPLY
15	DIVIDE	25	FLOATING DIVIDE
16	ROUND	26	INCREASE FIELD LOCATION
19	ADD TO LOCATION		

Register status:

Register name	Contents after execution of BOF.																										
A	Unchanged																										
R	"																										
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>3</td> <td>1</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> </tr> <tr> <td></td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>	+	1	1	1	1	3	1	a	a	a	a															
+	1	1	1	1	3	1	a	a	a	a																	
B	Unchanged																										
P	B[aaaa], if OVERFLOW Indicator "on." (rP) _b + 1, if OVERFLOW Indicator "off."																										
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>3</td> <td>1</td> <td>B</td> <td>[</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> <td>]</td> </tr> <tr> <td></td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>	1	1	1	1	3	1	B	[a	a	a	a]													
1	1	1	1	3	1	B	[a	a	a	a]															
E	Cleared																										

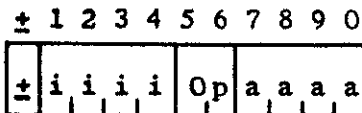
Operation name: BRANCH, REPEAT

Operation code: 32

Abbreviation: BRP

Instruction format:

Time (μ s):



No branch:

fetch: 90
 execute: 15
 total: 105

Definitions:

Branch:

\pm : if \pm is odd, B-register address-modification will occur; otherwise there will be no such modification.

fetch: 90
 execute: 35
 total: 125

iiii: not relevant to the execution of this instruction.

Op: operation code.

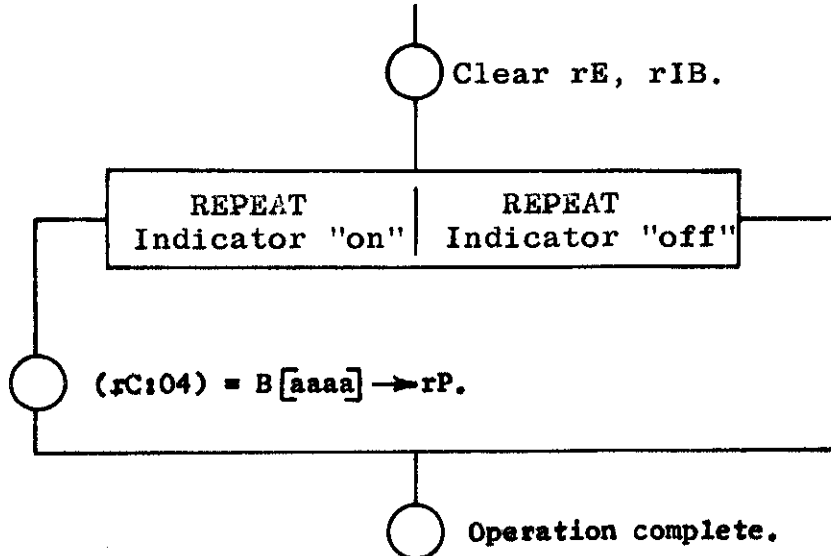
aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

If the REPEAT Indicator is "on", transfer control to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. If the REPEAT Indicator is "off", control continues in sequence.

Flow chart:



Exceptional conditions: NONE

Remarks:

1. The state of the REPEAT Indicator is not disturbed by the execution of a BRANCH, REPEAT instruction.
2. The REPEAT Indicator may be set "on" by the following operations:

Code	Name
27	DECREASE FIELD LOCATION
28	DECREASE FIELD LOCATION, LOAD B

Register status:

Register name	Contents after execution of BRP.											
A	Unchanged											
R	"											
D	<table border="1" style="margin-left: 20px;"> <tr> <td>+</td> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>3</td> <td>2</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> </tr> </table>	+	i	i	i	i	3	2	a	a	a	a
+	i	i	i	i	3	2	a	a	a	a		
B	Unchanged											
P	B[aaaa], if REPEAT Indicator "on". (rP) _b + 1, if REPEAT Indicator "off".											
C	<table border="1" style="margin-left: 20px;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>3</td> <td>2</td> <td>B[aaaa]</td> </tr> </table>	i	i	i	i	3	2	B[aaaa]				
i	i	i	i	3	2	B[aaaa]						
E	Cleared											

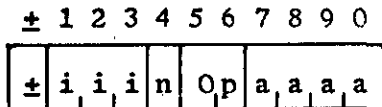
Operation name: BRANCH, SIGN A

Operation code: 33

Abbreviation: BSA

Instruction format:

Time (μ s):



No branch:

fetch:	90
execute:	<u>80</u>
total:	170

Definitions:

Branch:

- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iii: not relevant to the execution of this instruction.
- n: comparison digit.
- Op: operation code.
- aaaa: address of base of location of alternate instruction.

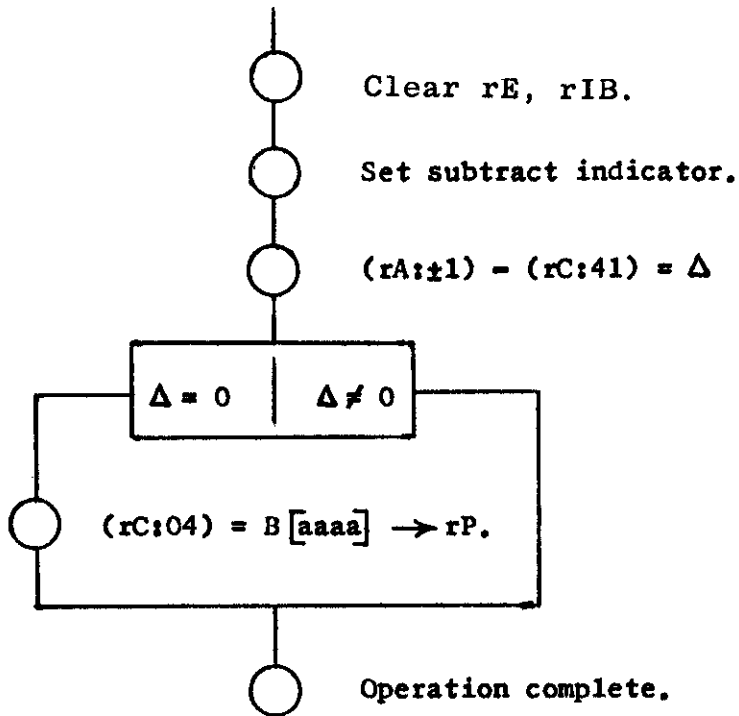
fetch:	90
execute:	<u>100</u>
total:	190

Description of operation:

Summary:

If $(rA:\pm 1) = n$, transfer control to location $B[aaaa]$, i.e., prepare to take the next instruction from $B[aaaa]$. If $(rA:\pm 1) \neq n$, control continues in sequence.

Flow chart:



Exceptional conditions: NONE.

Remarks:

Register status:

Register name	Contents after execution of BSA.											
A	Unchanged											
R	"											
D	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 10px;">±</td> <td style="width: 10px;">i</td> <td style="width: 10px;">i</td> <td style="width: 10px;">i</td> <td style="width: 10px;">n</td> <td style="width: 10px;">3</td> <td style="width: 10px;">3</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> </tr> </table>	±	i	i	i	n	3	3	a	a	a	a
±	i	i	i	n	3	3	a	a	a	a		
B	Unchanged											
P	$B[aaaa]$, if $(rA_{\pm 1}) = n$. $(rP)_b + 1$, if $(rA_{\pm 1}) \neq n$.											
C	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 10px;">i</td> <td style="width: 10px;">i</td> <td style="width: 10px;">i</td> <td style="width: 10px;">n</td> <td style="width: 10px;">3</td> <td style="width: 10px;">3</td> <td style="width: 10px;">B</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> <td style="width: 10px;">a</td> </tr> </table>	i	i	i	n	3	3	B	a	a	a	a
i	i	i	n	3	3	B	a	a	a	a		
E	Cleared											

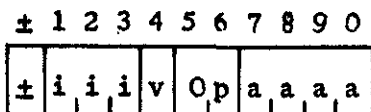
OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 34

Operation name: BRANCH, COMPARISON HIGH
 BRANCH, COMPARISON LOW

Abbreviation: BCH
 BCL

Instruction format:



Time (μs):

No branch:

fetch: 90
 execute: 15
 total: 105

Branch:

fetch: 90
 execute: 35
 total: 125

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: BRANCH, COMPARISON HIGH will be executed.

v = 1: BRANCH, COMPARISON LOW will be executed.

aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

v = 0: BRANCH, COMPARISON HIGH will be executed.

If the COMPARISON Indicator is HIGH, transfer control to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. If the COMPARISON Indicator is LOW or EQUAL, control continues in sequence.

v = 1: BRANCH, COMPARISON LOW will be executed.

If the COMPARISON Indicator is LOW, transfer control to location B[aaaa]. If the COMPARISON Indicator is HIGH or EQUAL, control continues in sequence.

Flow chart:

See page II-34-4.

Exceptional conditions:

1. No-comparison ALARM STOP.

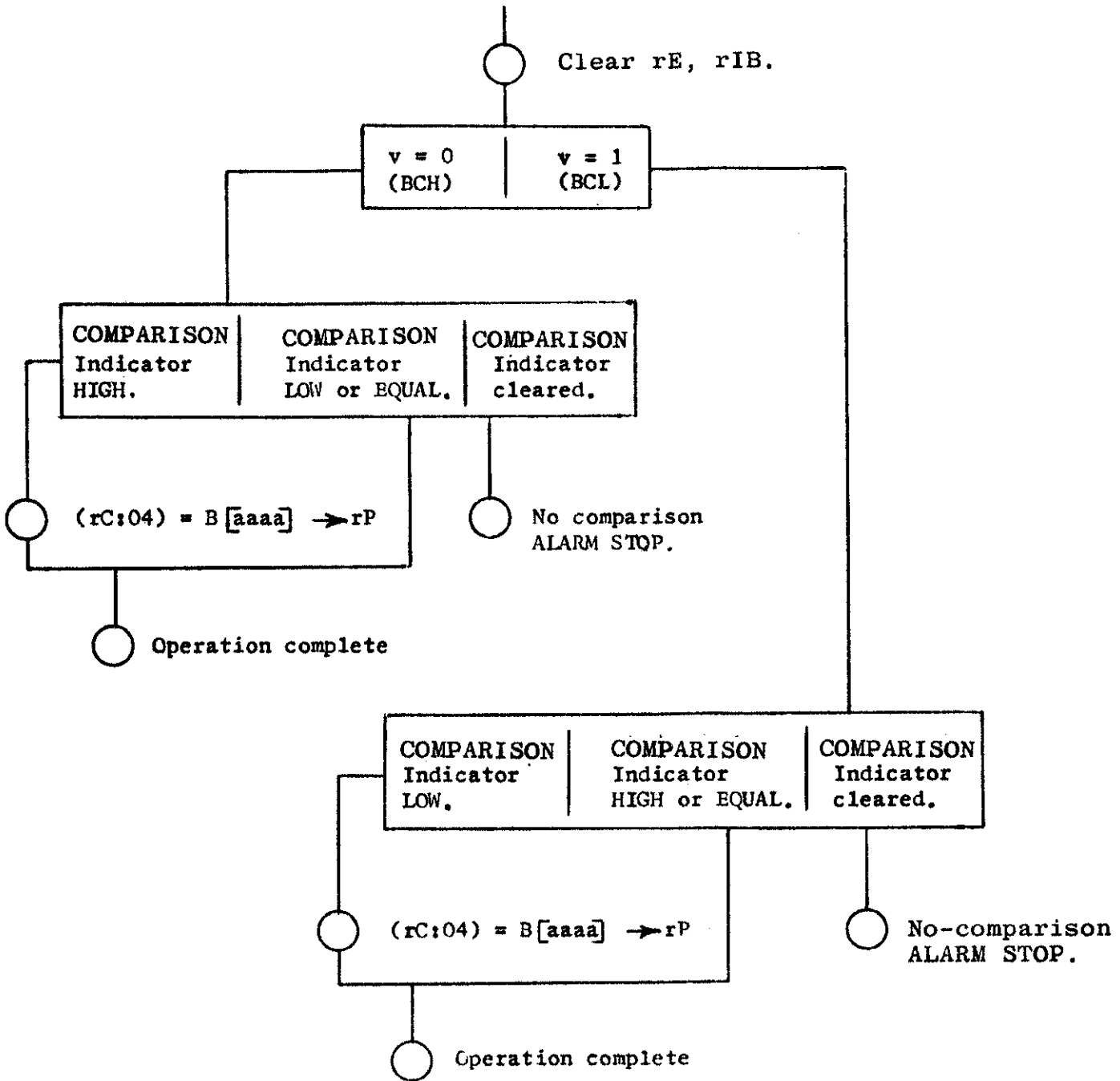
Remarks:

1. The BRANCH, COMPARISON HIGH variation will be executed if $v \neq 1$.
2. The state of the COMPARISON Indicator is not disturbed by the execution of these instructions.
3. The COMPARISON Indicator is set by the following operations:

Code	Name
18	COMPARE FIELD A
18	COMPARE FIELD R

Description of operation:

Flow chart:



THE COMPUTER

Register status:

Register name	Contents after execution if branching occurs.	Contents after execution if branching does not occur.																										
A	Unchanged	Unchanged																										
R	"	"																										
D	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	i	i	i	v	3	4	a	a	a	a	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	i	i	i	v	3	4	a	a	a	a				
±	i	i	i	v	3	4	a	a	a	a																		
±	i	i	i	v	3	4	a	a	a	a																		
B	Unchanged	Unchanged																										
P	B [aaaa]	(rP) _b + 1																										
C	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	v	3	4	B	[a	a	a	a]	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	v	3	4	B	[a	a	a	a]
i	i	i	v	3	4	B	[a	a	a	a]																
i	i	i	v	3	4	B	[a	a	a	a]																
E	Cleared	Cleared																										

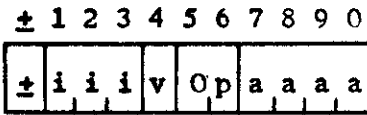
Register name	Contents if no-comparison ALARM STOP occurs.													
A	Unchanged													
R	"													
D	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	i	i	i	v	3	4	a	a	a	a		
±	i	i	i	v	3	4	a	a	a	a				
B	Unchanged													
P	(rP) _b + 1													
C	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	v	3	4	B	[a	a	a	a]
i	i	i	v	3	4	B	[a	a	a	a]		
E	Cleared													

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: BRANCH, COMPARISON EQUAL
 BRANCH, COMPARISON UNEQUAL

Operation code: 35
Abbreviation: BCE
 BCU

Instruction format:



Time (µs):

No branch:
 fetch: 90
 execute: 15
 total: 105

Definitions:

Branch:
 fetch: 90
 execute: 35
 total: 125

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iii: not relevant to the execution of these instructions.
- v: variation designator:
 - v = 0: BRANCH, COMPARISON EQUAL will be executed.
 - v = 1: BRANCH, COMPARISON UNEQUAL will be executed.
- Op: operation code.
- aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

v = 0: BRANCH, COMPARISON EQUAL will be executed.

If the COMPARISON Indicator is EQUAL, transfer control to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. If the COMPARISON Indicator is HIGH or LOW, control continues in sequence.

v = 1: BRANCH, COMPARISON UNEQUAL will be executed.

If the COMPARISON Indicator is HIGH or LOW, transfer control to location B[aaaa]. If the COMPARISON Indicator is EQUAL, control continues in sequence.

Flow chart:

See page II-35-4.

Exceptional conditions:

1. No-comparison ALARM STOP.

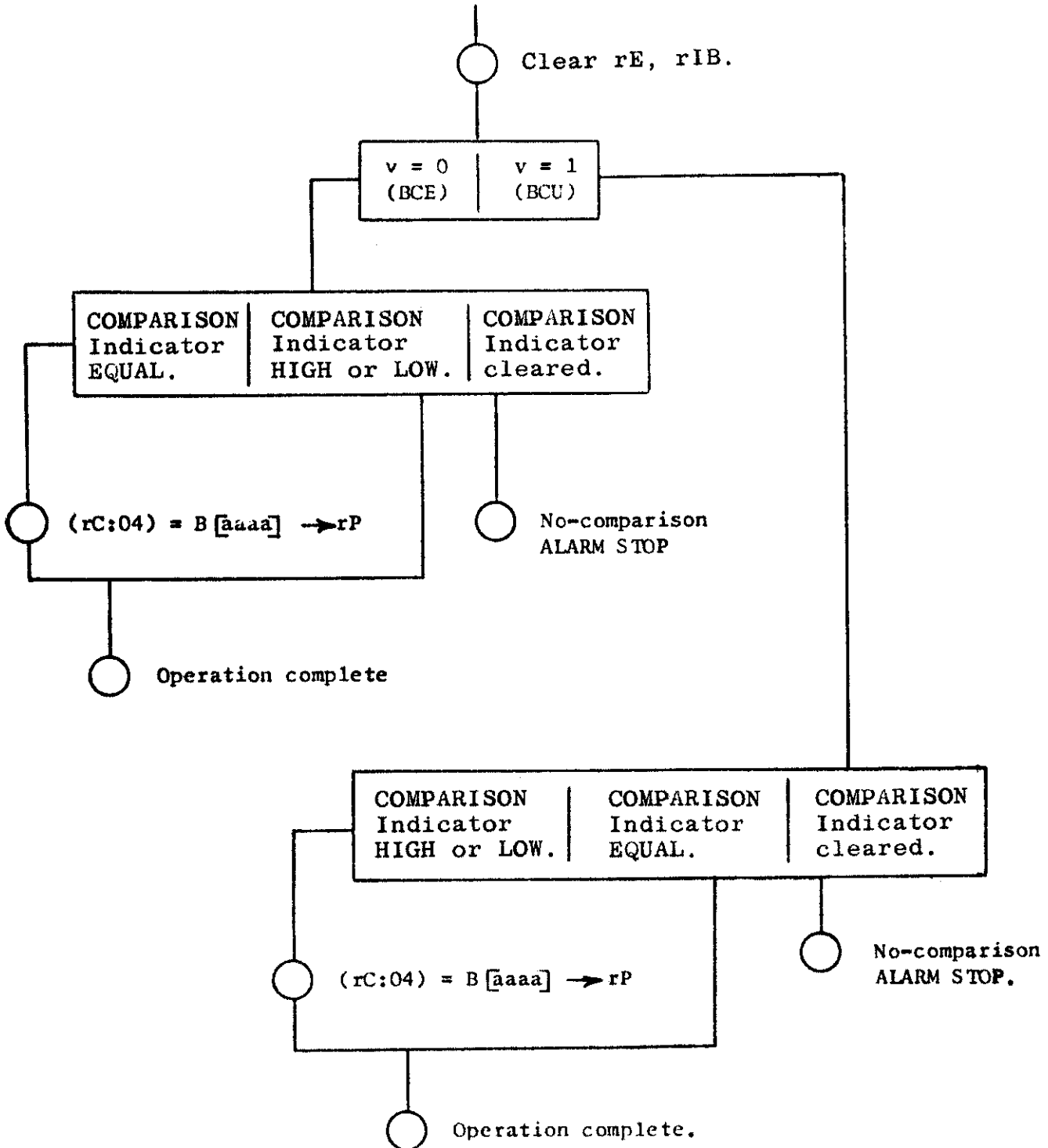
Remarks:

1. The BRANCH, COMPARISON EQUAL variation will be executed if $v \neq 1$.
2. The state of the COMPARISON Indicator is not disturbed by the execution of these instructions.
3. The COMPARISON Indicator is set by the following operations:

Code	Name
18	COMPARE FIELD A
18	COMPARE FIELD R

Description of operation:

Flow chart:



THE COMPUTER

Register status:

Register name	Contents after execution if branching occurs.	Contents after execution if branching does not occur.																						
A	Unchanged	Unchanged																						
R	"	"																						
D	<table border="1" style="margin-left: 20px;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	i	i	i	v	3	5	a	a	a	a	<table border="1" style="margin-left: 20px;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	i	i	i	v	3	5	a	a	a	a
±	i	i	i	v	3	5	a	a	a	a														
±	i	i	i	v	3	5	a	a	a	a														
B	Unchanged	Unchanged																						
P	B[aaaa]	(rP) _b + 1																						
C	<table border="1" style="margin-left: 20px;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	v	3	5	B	[aaaa]	<table border="1" style="margin-left: 20px;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	v	3	5	B	[aaaa]						
i	i	i	v	3	5	B	[aaaa]																	
i	i	i	v	3	5	B	[aaaa]																	
E	Cleared	Cleared																						

Register name	Contents if no-comparison ALARM STOP occurs.											
A	Unchanged											
R	"											
D	<table border="1" style="margin-left: 20px;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	i	i	i	v	3	5	a	a	a	a
±	i	i	i	v	3	5	a	a	a	a		
B	Unchanged											
P	(rP) _b + 1											
C	<table border="1" style="margin-left: 20px;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>3</td><td>5</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	v	3	5	B	[aaaa]			
i	i	i	v	3	5	B	[aaaa]					
E	Cleared											

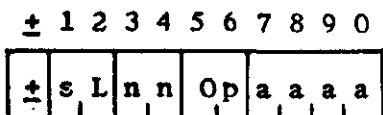
Operation name: BRANCH, FIELD A

Operation code: 36

Abbreviation: BFA

Instruction format:

Time (μ s):



No branch:

fetch:	90
execute:	<u>75</u>
total:	165

Definitions:

Branch:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

fetch:	90
execute:	<u>95</u>
total:	185

s: partial-word designator:

s designates the position, within the word, of the low-order digit of the partial-word operand.

L: partial-word designator:

L specifies the number of digits in the partial-word operand.

nn: basis for comparator.

Op: operation code.

aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

Beginning with the low order digit of (rA:SI), successively higher-order digits are compared alternately with the low-order and high-order digit of nn. If equality obtains for every digit position compared, transfer control to location B[aaaa], i.e., take the next instruction from B[aaaa]. If inequality obtains for any digit position, control continues in sequence.

Flow chart:

See page II-36-4.

Exceptional conditions:

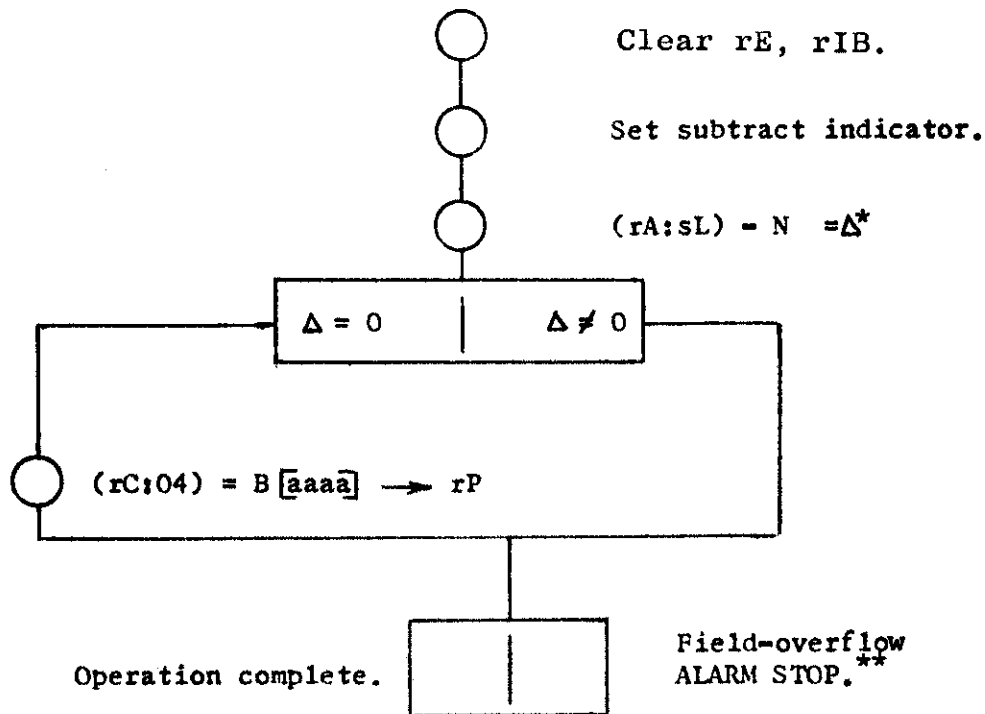
1. Field-overflow ALARM STOP.

Remarks:

1. Except for the difference in registers, BRANCH, FIELD A and BRANCH FIELD R are identical in operation. See pages II-37-2, ff.

Description of operation:

Flow chart:



* The subtrahend, N , is an L -digit number, constructed with the two-digit number, nn , as a basis: the low-order digit of N is the low-order digit of nn ; the next-higher-order digit of N is the high-order digit of nn ; the next-higher-order digit of N is the low-order digit of nn ; and so forth, successively higher-order digits of N being, alternately the high- and low-order digits of nn .

** As each digit of N is generated, L (i.e., $(rC:21)$) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the comparison $(rC:21)$ is different from zero, and field overflow is detected.

THE COMPUTER

Register status:

Register name	Contents after execution of BFA.	Contents if field-overflow ALARM STOP occurs.																						
A	Unchanged	Unchanged																						
R	"	"																						
D	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>±</td><td>s</td><td>L</td><td>n</td><td>n</td><td>3</td><td>6</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	n	n	3	6	a	a	a	a	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>±</td><td>s</td><td>L</td><td>n</td><td>n</td><td>3</td><td>6</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	n	n	3	6	a	a	a	a
±	s	L	n	n	3	6	a	a	a	a														
±	s	L	n	n	3	6	a	a	a	a														
B	Unchanged	Unchanged																						
P	B[aaaa], if branch. (rP) _b + 1, if no branch.	B[āaaa], if branch would have occurred. (rP) _b + 1, if no branch would have occurred.																						
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0</td><td>0</td><td>n</td><td>n</td><td>3</td><td>6</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	0	n	n	3	6	B	[aaaa]	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0</td><td>*</td><td>n</td><td>n</td><td>3</td><td>6</td><td>B</td><td>[āaaa]</td> </tr> </table>	0	*	n	n	3	6	B	[āaaa]						
0	0	n	n	3	6	B	[aaaa]																	
0	*	n	n	3	6	B	[āaaa]																	
E	Cleared	Cleared																						

*L-S-1

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: BRANCH, FIELD R

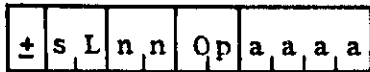
Operation code: 37

Abbreviation: BFR

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0



No branch:

fetch:	90
execute:	<u>75</u>
total:	165

Definitions:

Branch:

\pm : if \pm is odd, B-register address, modification will occur; otherwise, there will be no such modification.

fetch:	90
execute:	<u>95</u>
total:	185

s: partial-word designator.

s designates the position, within the word, of the low-order digit of the partial-word operand.

L: partial-word designator.

L specifies the number of digits in the partial-word operand.

nn: basis for comparator.

Op: operation code.

aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

Beginning with the low-order digit of (rR:sL), successively higher-order digits are compared alternately with the low-order and high-order digit of mn. If equality obtains for every digit position compared, transfer control to B [aaaa], i.e., take the next instruction from B [aaaa]. If inequality obtains for any digit position, control continues in sequence.

Flow chart:

See page II-37-4.

Exceptional conditions:

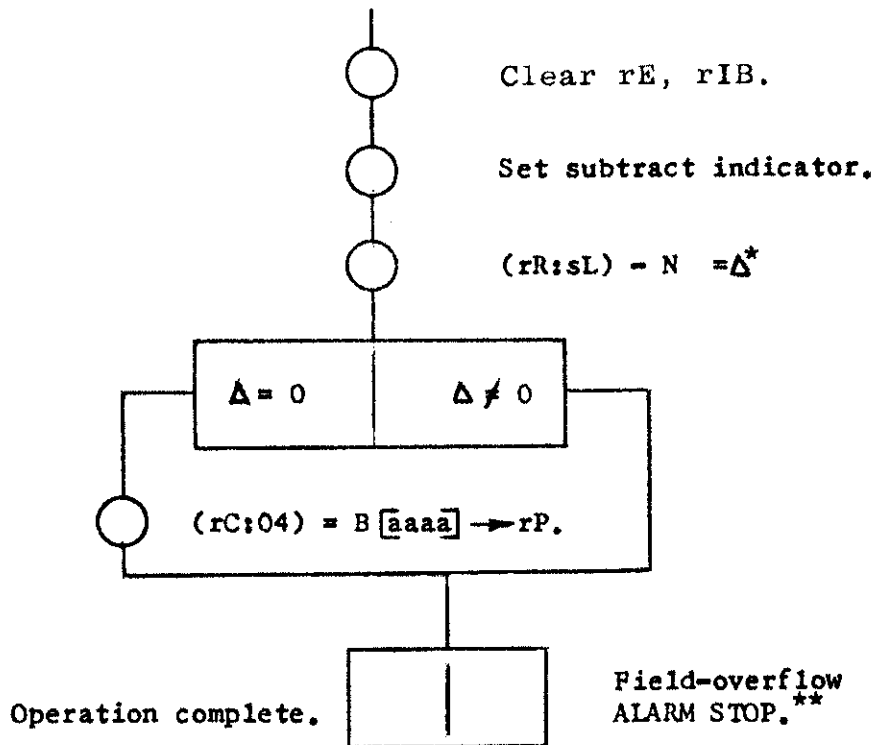
1. Field-overflow ALARM STOP.

Remarks:

1. Except for the difference in registers, BRANCH, FIELD R and BRANCH, FIELD A are identical in operation. See pages II-36-2, ff.

Description of operation:

Flow chart:



* The subtrahend, N , is an L - digit number, constructed with the two-digit number, nn , as a basis: the low-order digit of N is the low-order digit of nn ; the next-higher-order digit of N is the high-order digit of nn ; the next-higher-order digit of N is the low-order digit of nn ; the so forth, successively higher-order digits of N being, alternately, the high- and low-order digits of nn .

** As each digit of N is generated, L (i.e., $(rC:21)$) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the comparison, $(rC:21)$ is different from zero, and field overflow is detected.

THE COMPUTER

Description of operation:

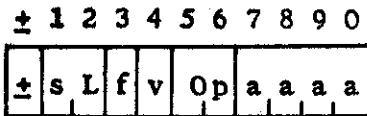
Register name	Contents after execution of BFR	Contents if field-overflow ALARM STOP occurs.																						
A	Unchanged	Unchanged																						
R	"	"																						
D	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">s</td> <td style="padding: 2px;">L</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	s	L	n	n	3	7	a	a	a	a	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">s</td> <td style="padding: 2px;">L</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> <td style="padding: 2px;">a</td> </tr> </table>	±	s	L	n	n	3	7	a	a	a	a
±	s	L	n	n	3	7	a	a	a	a														
±	s	L	n	n	3	7	a	a	a	a														
B	Unchanged	Unchanged																						
P	B [aaaa], if branch. (rP) _b + 1, if no branch.	B [aaaa], if branch would have occurred. (rP) _b + 1, if no branch would have occurred.																						
C	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[aaaa]</td> </tr> </table>	0	0	n	n	3	7	B	[aaaa]	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;">0</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[aaaa]</td> </tr> </table>	0	*	n	n	3	7	B	[aaaa]						
0	0	n	n	3	7	B	[aaaa]																	
0	*	n	n	3	7	B	[aaaa]																	
E	Cleared	Cleared *L-s-1																						

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: STORE A
 STORE R
 STORE B

Operation code: 40
Abbreviation: STA
 STR
 STB

Instruction format:



Time (μs):

f = 0:
 fetch: 90
 execute: 95
 total: 185

f = 1:
 fetch: 90
 execute: 105
 total: 195

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

s: partial-word designator:

f = 0: s is not relevant.

f = 1: s designates the position within the word, of the low-order digit of each partial-word operand.

L: partial-word designator:

f = 0: L is not relevant.

f = 1: L specifies the number of digits in each partial-word operand.

f: partial-word designator:

f = 0: entire words will be used as operands.

f = 1: the contents of the partial-word fields defined by sL will be used as operands.

v: variation designator:

v = 0: STORE A will be executed.

v = 1: STORE R will be executed.

v = 2: STORE B will be executed.

Op: operation code.

aaaa: address of base of location in
which the selected field will be
stored.

Description of operation:

Summary:

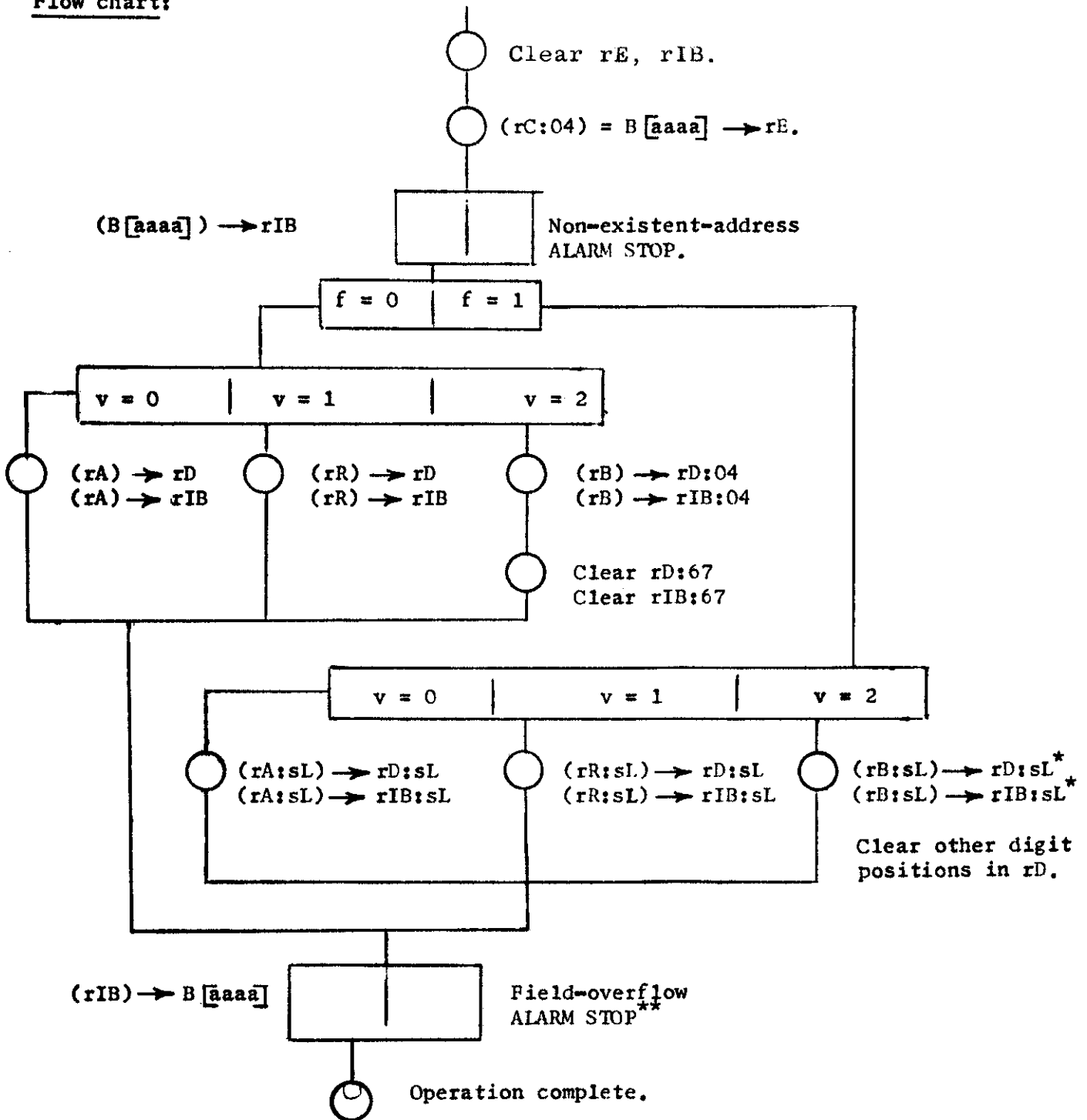
Store the specified field of the designated register in the corresponding field location in B[aaaa].

Flow chart:

See page II-40-4.

Description of operation:

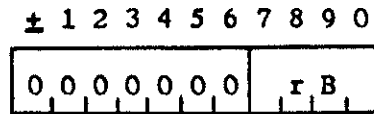
Flow chart:



* See Note 1, page II-40-5

** See Note 2, page II-40-5

Note 1. The B register is regarded as if it were 11 digits long as indicated:



Note 2. As each digit is transferred from the designated register to the D register, L (i.e., (rC:21)) is counted down. If, at the start, $L > s + 1$, $s \neq 0$, then at the end of the transfer (rC:21) will be different from zero and field overflow will be detected.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP.

Remarks:

1. The STORE A variation will be executed if $v \neq 1$ or 2.
2. $f = 2, 4, 6$, or 8 has the same effect as $f = 0$; $f = 3, 5, 7$, or 9 has the same effect as $f = 1$.

Register status:

Register name	Contents after execution of all instructions	Contents if non-existent-address ALARM STOP occurs.																
A	Unchanged	Unchanged																
R	"	"																
D	$(B[aaaa])_a$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>s</td><td>L</td><td>f</td><td>v</td><td>4</td><td>0</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	±	s	L	f	v	4	0	a	a	a	a					
±	s	L	f	v	4	0	a	a	a	a								
B	Unchanged	Unchanged																
P	$(rP)_b + 1$	$(rP)_b + 1$																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>0</td><td>f</td><td>v</td><td>4</td><td>0</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	0	f	v	4	0	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>s</td><td>L</td><td>f</td><td>v</td><td>4</td><td>0</td><td>B</td><td>[aaaa]</td> </tr> </table>	s	L	f	v	4	0	B	[aaaa]
0	0	f	v	4	0	B	[aaaa]											
s	L	f	v	4	0	B	[aaaa]											
E	$B[aaaa]$	$B[aaaa]$																

Register name	Contents if field-overflow ALARM STOP occurs.								
A	Unchanged								
R	"								
D	See flow chart.								
B	Unchanged								
P	$(rP)_b + 1$								
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0</td><td>*</td><td>f</td><td>v</td><td>4</td><td>0</td><td>B</td><td>[aaaa]</td> </tr> </table>	0	*	f	v	4	0	B	[aaaa]
0	*	f	v	4	0	B	[aaaa]		
E	$B[aaaa]$								

* L-s-1

Examples:

1. (rC) = 0412 40 1000
 (1000)_b = 0 1234 56 7890
 (rB) = 1234
 (rD)_a = 0 0000 00 1234
 (1000)_a = 0 1234 56 1234

2. (rC) = 0002 40 1000
 (1000)_b = 0 1234 56 7890
 (rB) = 1234
 (rD)_a = 0 0000 00 1234
 (1000)_a = 0 0000 00 1234

3. (rC) = 8412 40 2000
 (2000)_b = 0 1234 56 7890
 (rB) = 2345
 (rD)_a = 0 0000 00 2300
 (2000)_a = 0 1234 00 2390

4. (rC) = 5312 40 3000
 (3000)_b = 0 1234 56 7890
 (rB) = 1357
 (rD)_a = 0 0000 00 0000
 (3000)_a = 0 1200 06 7890

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: LOAD R

Operation code: 41

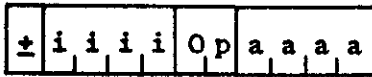
Abbreviation: LDR

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

fetch: 90



execute: 85

total: 175

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

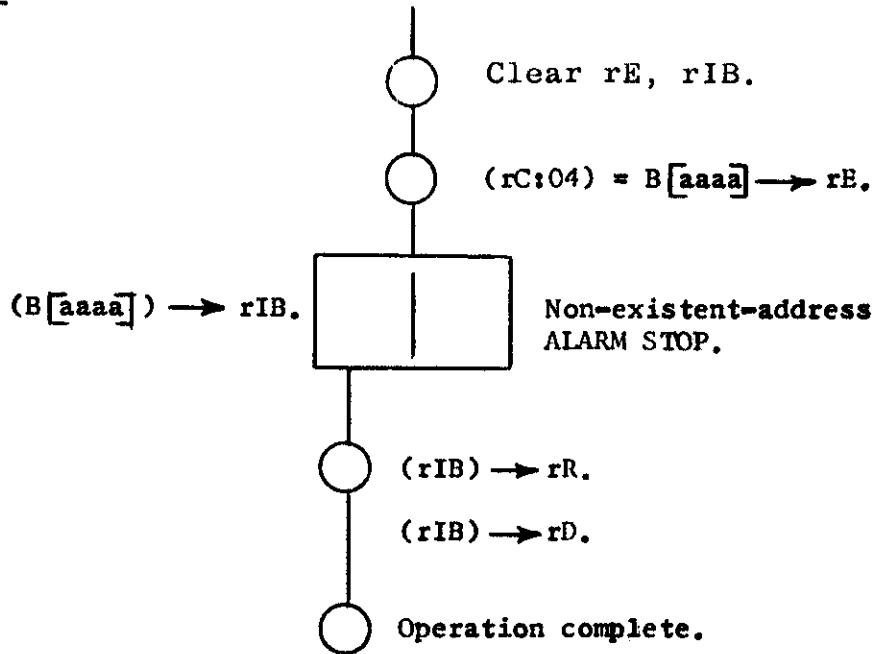
aaaa: address of base of location of operand.

Description of operation:

Summary:

Replace (rR) by (B[aaaa]).

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

Register status:

Register name	Contents after execution of LR.	Contents if non-existent-address ALARM STOP occurs.																
A	Unchanged	Unchanged																
R	(B [aaaa])	"																
D	(B [aaaa])	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>4</td> <td>1</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> </tr> </table>	+	i	i	i	i	4	1	a	a	a	a					
+	i	i	i	i	4	1	a	a	a	a								
B	Unchanged (rP) _b + 1	Unchanged (rP) _b + 1																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>4</td> <td>1</td> <td>B</td> <td>[aaaa]</td> </tr> </table>	i	i	i	i	4	1	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>i</td> <td>4</td> <td>1</td> <td>B</td> <td>[aaaa]</td> </tr> </table>	i	i	i	i	4	1	B	[aaaa]
i	i	i	i	4	1	B	[aaaa]											
i	i	i	i	4	1	B	[aaaa]											
E	B [aaaa]	B [aaaa]																

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 42

Operation name: LOAD B

Abbreviation: LDB

LOAD B, COMPLEMENT

LBC

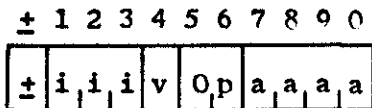
Time (μs):

fetch: 90

execute: 90

total: 180

Instruction format:



Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: LOAD B will be executed.

v = 1: LOAD B, COMPLEMENT will be executed.

Op: operation code.

aaaa: address of base of location of operand.

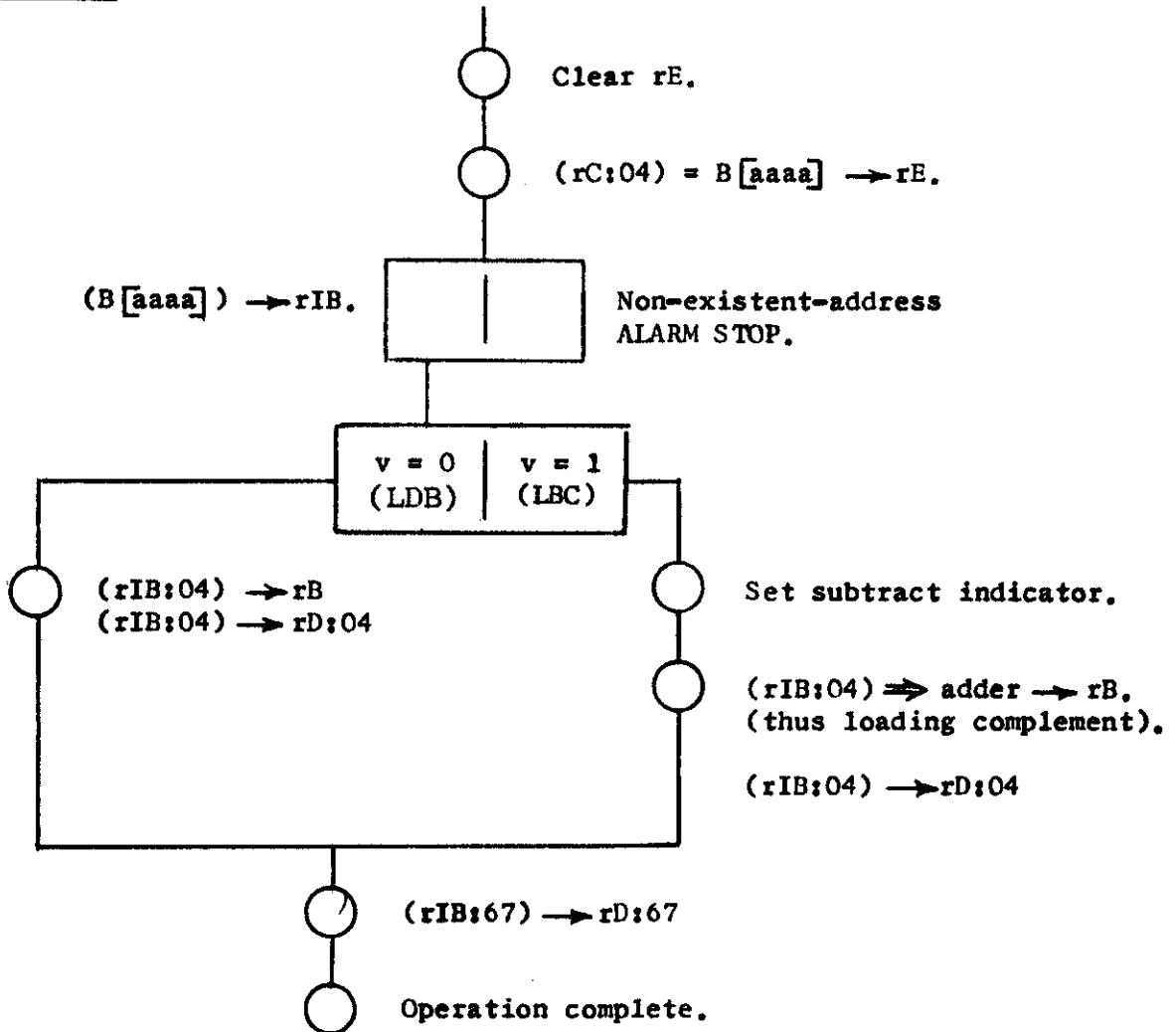
Description of operation:

Summary:

v = 0: (B[aaaa]:04) replace (rB).

v = 1: the 10's complement of (B[aaaa]) replaces (rB).

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

1. The LOAD B variation will be executed if $v \neq 1$.

Register status:

Register name	Contents after execution of LDB	Contents after execution of LBC.																
A	Unchanged	Unchanged																
R	"	"																
D	(B [aaaa])	(B [aaaa])																
B	(B [aaaa] :04)	10's complement of (B [aaaa] :04)																
P	(rP) _b + 1	(rP) _b + 1																
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>0</td><td>4</td><td>2</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	0	4	2	B	[aaaa]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>1</td><td>4</td><td>2</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	1	4	2	B	[aaaa]
i	i	i	0	4	2	B	[aaaa]											
i	i	i	1	4	2	B	[aaaa]											
E	B [aaaa]	B [aaaf]																

Register name	Contents if non-existent-address ALARM STOP occurs.											
A	Unchanged											
R	"											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td><td>i</td><td>i</td><td>i</td><td>v</td><td>4</td><td>2</td><td>a</td><td>a</td><td>a</td><td>a</td> </tr> </table>	+	i	i	i	v	4	2	a	a	a	a
+	i	i	i	v	4	2	a	a	a	a		
B	Unchanged											
P	(rP) _b + 1											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>v</td><td>4</td><td>2</td><td>B</td><td>[aaaa]</td> </tr> </table>	i	i	i	v	4	2	B	[aaaa]			
i	i	i	v	4	2	B	[aaaa]					
E	B [aaaa]											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: LOAD SIGN A

Operation code: 43

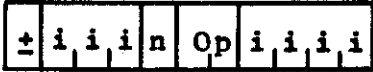
Abbreviation: LSA

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

fetch: 90



execute: 15

total: 105

Definitions:

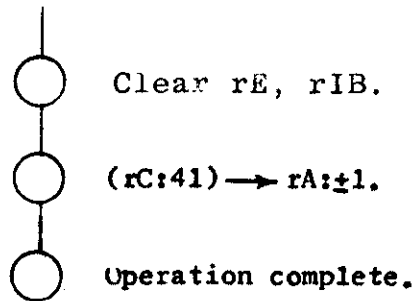
- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iii: not relevant to the execution of this instruction.
- iiii: instruction.
- n: modifier for sign-digit position of A register.
- Op: operation code.

Description of operation:

Summary:

Replace the contents of the sign-digit position of the A register by n.

Flow chart:



Exceptional conditions: NONE

Remarks:

1. Although the address-field is not used for addressing purposes, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.

Register status:

Register name	Contents after execution of LSA.																						
A	$(rA:\pm 1) = n$ $(rA:00)_a = (rA:00)_b$																						
R	Unchanged																						
D	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td>\pm</td> <td>i</td> <td>i</td> <td>i</td> <td>n</td> <td>4</td> <td>3</td> <td>i</td> <td>i</td> <td>i</td> <td>i</td> </tr> <tr> <td></td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>	\pm	i	i	i	n	4	3	i	i	i	i											
\pm	i	i	i	n	4	3	i	i	i	i													
B	Unchanged																						
P	$(rP)_b + 1$																						
C	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td>i</td> <td>i</td> <td>i</td> <td>n</td> <td>4</td> <td>3</td> <td>B</td> <td>iiii </td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table>	i	i	i	n	4	3	B	iiii														
i	i	i	n	4	3	B	iiii																
E	Cleared																						

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 44

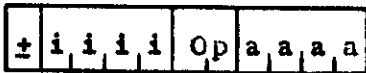
Operation name: STORE P

Abbreviation: STP

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0



fetch: 90

execute: 95

total: 185

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

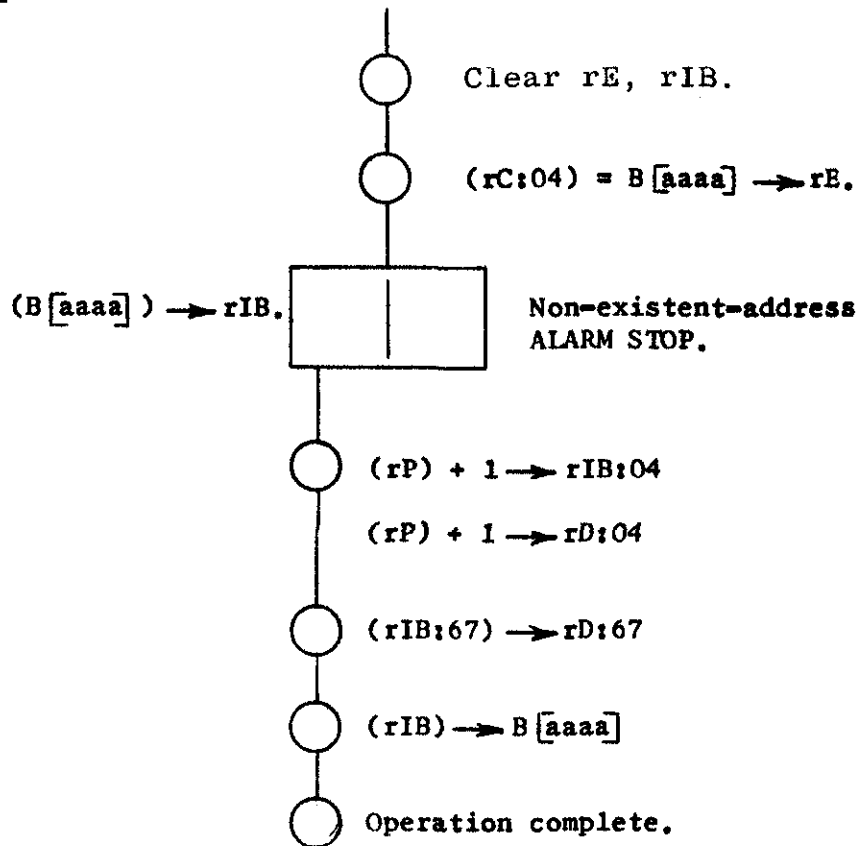
aaaa: address of base of location in which will be stored the augmented content of the P register.

Description of operation:

Summary:

$(rP) + 1 \rightarrow B[aaaa]:04.$

Flow chart:



Exceptional conditions:

1. Non-existent-address ALARM STOP.

Remarks:

Register status:

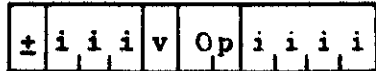
Register name	Contents after execution of STP.	Contents if non-existent-address ALARM STOP occurs.																										
A	Unchanged	Unchanged																										
R	"	"																										
D	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: 1px solid black; padding: 2px;">±</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> </tr> </table> </div>	±	i	i	i	i	4	4	a	a	a	a	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: 1px solid black; padding: 2px;">±</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> </tr> </table> </div>	±	i	i	i	i	4	4	a	a	a	a				
±	i	i	i	i	4	4	a	a	a	a																		
±	i	i	i	i	4	4	a	a	a	a																		
B	Unchanged	Unchanged																										
P	$(rP)_b + 1$	$(rP)_b + 1$																										
C	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">B</td> <td style="border: 1px solid black; padding: 2px;">[</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">]</td> </tr> </table> </div>	i	i	i	i	4	4	B	[a	a	a	a]	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> <table style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">i</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">B</td> <td style="border: 1px solid black; padding: 2px;">[</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">a</td> <td style="border: 1px solid black; padding: 2px;">]</td> </tr> </table> </div>	i	i	i	i	4	4	B	[a	a	a	a]
i	i	i	i	4	4	B	[a	a	a	a]																
i	i	i	i	4	4	B	[a	a	a	a]																
E	B [aaaa]	B [aaaa]																										

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

<u>Operation name:</u> CLEAR A	<u>Operation code:</u> 45
CLEAR R	<u>Abbreviation:</u> CLA
CLEAR B	CLR
	CLB

Instruction format:

± 1 2 3 4 5 6 7 8 9 0



Time (µs):

fetch: 90
 execute: 10
 total: 100

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii, not relevant to the execution of
 iii; these instructions.

v: variation designator: if

(rC:41)/1 = 1*: CLEAR A will be executed;
 and/or

(rC:41)/2 = 1*: CLEAR R will be executed;
 and/or

(rC:41)/4 = 1*: CLEAR B will be executed.

Op: operation code.

* (rC:41)/1 is the one-bit of v;
 (rC:41)/2 is the two-bit of v;
 and (rC:41)/4 the four-bit of v.

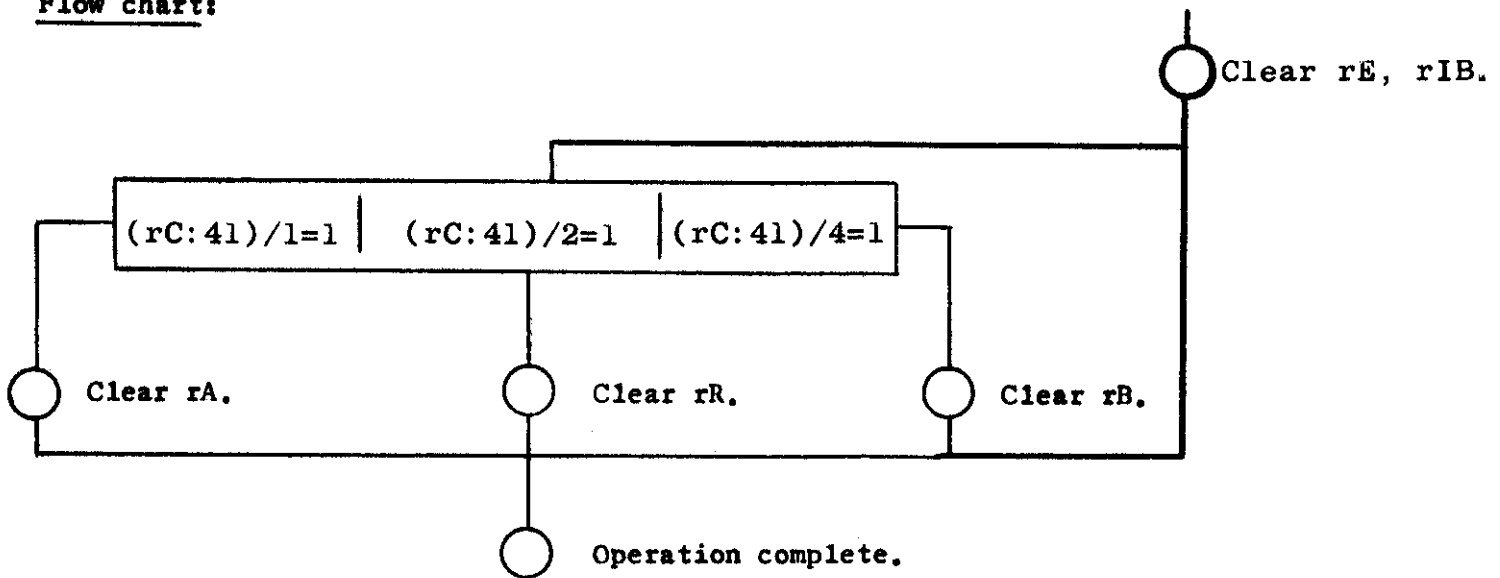
THE COMPUTER

Description of operation:

Summary:

v	Clear register	Abbreviation
0	None	
1	A	CLA
2	R	CLR
3	A, R	CAR
4	B	CLB
5	A, B	CAB
6	R, B	CRB
7	A, R, B	CLT
8	None	
9	A	

Flow chart:



Exceptional conditions: NONE.

Remarks:

1. If $v = 0$ or if $v = 8$, no registers will be cleared: the effect is the same as that of a NO OPERATION instruction.

2. Although the address field is not used for addressing purposes, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however..

Register status:

Register name	Contents after execution of CLA	Contents after execution of CLR																						
A	Cleared	Unchanged																						
R	Unchanged	Cleared																						
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>i</td><td>i</td><td>i</td> <td>1</td><td>4</td><td>5</td> <td>i</td><td>i</td><td>i</td><td>i</td> </tr> </table>	+	i	i	i	1	4	5	i	i	i	i	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>i</td><td>i</td><td>i</td> <td>2</td><td>4</td><td>5</td> <td>i</td><td>i</td><td>i</td><td>i</td> </tr> </table>	+	i	i	i	2	4	5	i	i	i	i
+	i	i	i	1	4	5	i	i	i	i														
+	i	i	i	2	4	5	i	i	i	i														
B	Unchanged	Unchanged																						
P	$(rP)_b + 1$	$(rP)_b + 1$																						
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td> <td>1</td><td>4</td><td>5</td> <td>B</td> <td>[i][i][i][i]</td> </tr> </table>	i	i	i	1	4	5	B	[i][i][i][i]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td> <td>2</td><td>4</td><td>5</td> <td>B</td> <td>[i][i][i][i]</td> </tr> </table>	i	i	i	2	4	5	B	[i][i][i][i]						
i	i	i	1	4	5	B	[i][i][i][i]																	
i	i	i	2	4	5	B	[i][i][i][i]																	
E	Cleared	Cleared																						

Register name	Contents after execution of CLB											
A	Unchanged											
R	"											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>i</td><td>i</td><td>i</td> <td>4</td><td>4</td><td>5</td> <td>i</td><td>i</td><td>i</td><td>i</td> </tr> </table>	+	i	i	i	4	4	5	i	i	i	i
+	i	i	i	4	4	5	i	i	i	i		
B	Cleared											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td> <td>4</td><td>4</td><td>5</td> <td>B</td> <td>[i][i][i][i]</td> </tr> </table>	i	i	i	4	4	5	B	[i][i][i][i]			
i	i	i	4	4	5	B	[i][i][i][i]					
E	Cleared											

Note: if, for example, $v = 6$, then both rR and rB will be cleared. The status of the other registers is as indicated.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CLEAR LOCATION

Instruction format:

\pm 1 2 3 4 5 6 7 8 9 0



Operation code: 46

Abbreviation: CLL

Time (μ s):

fetch: 90

execute: 25

total: 115

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

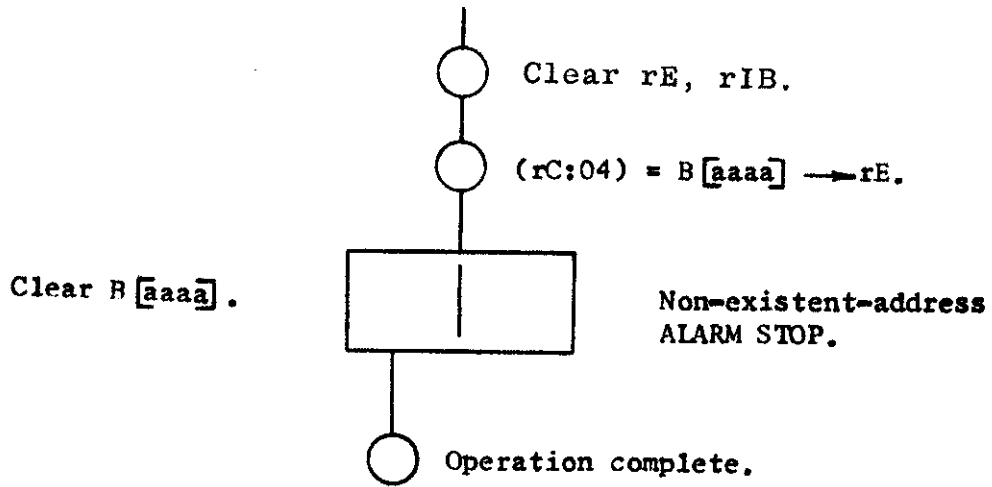
aaaa: address of base of location to be cleared.

Description of operation:

Summary:

Clear the contents of location B [aaaa].

Flow chart:



Exceptional conditions: NONE

Remarks:

Register status:

Register name	Contents after execution of CLL	Contents if non-existent-address ALARM STOP occurs.																										
A	Unchanged	Unchanged																										
R	"	"																										
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">4</td> <td style="text-align: center;">6</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	i	i	i	i	4	6	a	a	a	a	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">4</td> <td style="text-align: center;">6</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	i	i	i	i	4	6	a	a	a	a				
+	i	i	i	i	4	6	a	a	a	a																		
+	i	i	i	i	4	6	a	a	a	a																		
B	Unchanged	Unchanged																										
P	$(rP)_b + 1$	$(rP)_b + 1$																										
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">4</td> <td style="text-align: center;">6</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">]</td> </tr> </table>	i	i	i	i	4	6	B	[a	a	a	a]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">4</td> <td style="text-align: center;">6</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">]</td> </tr> </table>	i	i	i	i	4	6	B	[a	a	a	a]
i	i	i	i	4	6	B	[a	a	a	a]																
i	i	i	i	4	6	B	[a	a	a	a]																
E	B [aaaa]	B [aaaa]																										

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 48

Operation name: SHIFT RIGHT A

Abbreviation: SRA

SHIFT RIGHT A AND R

SRT

SHIFT RIGHT A WITH SIGN

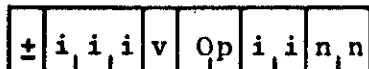
SRS

Time (μ s):

Instruction format:

fetch: 90
 execute: $20 + 5$ per digit
 total: $110 + 5$ per digit

\pm 1 2 3 4 5 6 7 8 9 0



Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iii, not relevant to the execution of
 ii: these instructions.

v: variation designator:

v = 0: SHIFT RIGHT A will be executed.

v = 1: SHIFT RIGHT A AND R will be executed.

v = 2: SHIFT RIGHT A WITH SIGN will be executed.

nn: specifies the number of digit positions (00 to 19) through which the operand will be shifted.

Description of operation:

Summary:

v = 0: SHIFT RIGHT A will be executed.

The contents of the A register, excluding the contents of the sign-digit position, are shifted to the right through the number of digit positions specified by nn. Digits shifted out of the low-order end of the A register are lost; as each digit is shifted out of the A register, a "0" is entered into digit-position 1 in the A register.

v = 1: SHIFT RIGHT A AND R will be executed.

The contents of the A and R registers, excluding the contents of the sign-digit positions of both registers, but regarded as one twenty-digit-long number, are shifted to the right through the number of digit positions specified by nn. Digits shifted out of the low-order end of the R register are lost; as each digit is shifted out of the R register, a "0" is entered into digit-position 1 in the A register.

Although the contents of the sign-digit position of neither register is shifted during the execution of this instruction, the contents of the sign-digit position of the R register are replaced by the contents of the sign-digit position of the A register; the sign digit of the A register is not altered.

v = 2: SHIFT RIGHT A WITH SIGN

The contents of the A register, including the contents of the sign-digit position, are shifted to the right through the number of digit positions specified by nn. Digits shifted out of the low-order end of the A register are lost; as each digit is shifted out of the A register, a "0" is entered into the sign-digit position of the A register.

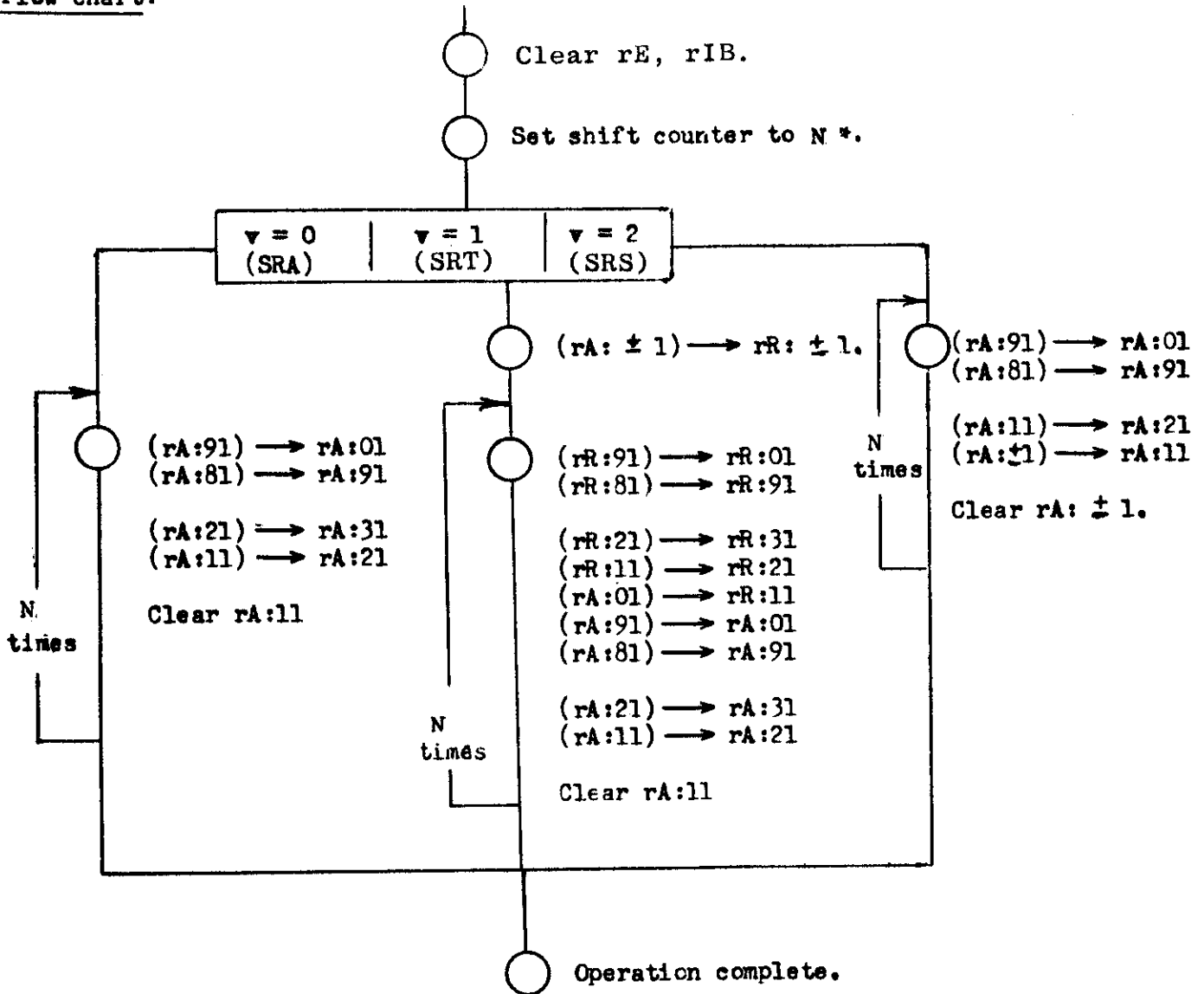
Flow chart:

See page II-48-4.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Description of operation:

Flow chart:



*N is the least non-negative remainder obtained on dividing nn by 20.

Exceptional conditions: NONE.

Remarks:

1. The SHIFT RIGHT A variation will be executed if $v \neq 1$ or 2.
2. Although the address field is not used for addressing purposes, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.
3. The number of digit positions through which the contents of the register(s) will be shifted is always less than or equal to 19, regardless of the value of nn . The number, N , of digit positions actually shifted is the least non-negative remainder obtained on dividing nn by 20.

Register status:

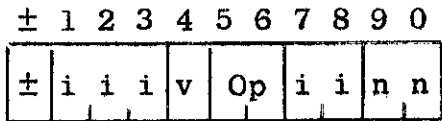
Register name	Contents after execution of SRA.	Contents after execution of SRT.																						
A	(rA:±1) is unchanged (rA:00) shifted as specified	(rA:±1) is unchanged (rA:00) shifted as specified																						
R	Unchanged	(rR:±1) _a = (rA:±1) _b (rR:00) _b , shifted as specified.																						
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>0</td><td>4</td><td>8</td><td>i</td><td>i</td><td>n</td><td>n</td> </tr> </table>	±	i	i	i	0	4	8	i	i	n	n	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>1</td><td>4</td><td>8</td><td>i</td><td>i</td><td>n</td><td>n</td> </tr> </table>	±	i	i	i	1	4	8	i	i	n	n
±	i	i	i	0	4	8	i	i	n	n														
±	i	i	i	1	4	8	i	i	n	n														
B	Unchanged	Unchanged																						
P	(rP) _b + 1	(rP) _b + 1																						
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>0</td><td>4</td><td>8</td><td>B</td><td>[iinn]</td> </tr> </table>	i	i	i	0	4	8	B	[iinn]	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>1</td><td>4</td><td>8</td><td>B</td><td>[iinn]</td> </tr> </table>	i	i	i	1	4	8	B	[iinn]						
i	i	i	0	4	8	B	[iinn]																	
i	i	i	1	4	8	B	[iinn]																	
E	Cleared	Cleared																						

Register name	Contents after execution of SRS.											
A	(rA), shifted as specified.											
R	Unchanged											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>±</td><td>i</td><td>i</td><td>i</td><td>2</td><td>4</td><td>8</td><td>i</td><td>i</td><td>n</td><td>n</td> </tr> </table>	±	i	i	i	2	4	8	i	i	n	n
±	i	i	i	2	4	8	i	i	n	n		
B	Unchanged											
P	(rP) _b + 1											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>i</td><td>i</td><td>i</td><td>2</td><td>4</td><td>8</td><td>B</td><td>[iinn]</td> </tr> </table>	i	i	i	2	4	8	B	[iinn]			
i	i	i	2	4	8	B	[iinn]					
E	Cleared											

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

<u>Operation name:</u> SHIFT LEFT A	<u>Operation code:</u> 49
SHIFT LEFT A AND R	<u>Abbreviation:</u> SLA
SHIFT LEFT A WITH SIGN	SLT
	SLS

Instruction format:



Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- iii, not relevant to the execution of these instructions.
- ii: these instructions.
- v: variation designator:
 - v = 0: SHIFT LEFT A will be executed.
 - v = 1: SHIFT LEFT A AND R will be executed.
 - v = 2: SHIFT LEFT A WITH SIGN will be executed.
- Op: operation code
- nn: specifies the number of digit positions through which the operand will be shifted.

Time (µs):

SLA, SLS:

fetch:	90
execute:	70 - 5N ₁ *
total:	160 - 5N ₁

SLT:

fetch:	90
execute:	120 - 5N ₂ *
total:	210 - 5N ₂

*See Remarks 3 and 4.

Description of operation:

Summary:

v = 0: SHIFT LEFT A will be executed.

The contents of the A register, excluding the content of the sign digit position, are shifted to the left through the number of digit positions specified by nn. This is a circulating shift, that is, as each digit is shifted out of digit position 1 in the A register it is entered into the low-order digit position of the A register.

v = 1: SHIFT LEFT A AND R will be executed.

The contents of the A and R registers, excluding the contents of the sign digit positions of both registers, but regarded as one twenty-digit-long number, are shifted to the left through the number of digit positions specified by nn. This is a circulating shift, that is, as each digit is shifted out of digit position 1 in the A register it is entered into the low-order digit position in the R register.

Although the contents of the sign-digit position of neither register is shifted during the execution of this instruction, the content of the sign digit position of the A register is replaced by the content of the sign digit position of the R register; the sign digit of the R register is not altered.

v = 2: SHIFT LEFT A WITH SIGN will be executed:

The contents of the A register, including the content of the sign digit position, are shifted to the left through the number of digit positions specified by nn. This is a circulating shift, that is, as each digit is shifted out of the sign digit position in the A register it is entered into the low-order digit position in the A register.

Flow chart:

See page II-49- .

Exceptional conditions: NONE

Remarks:

1. The SHIFT LEFT A variation will be selected for execution if $v \neq 1$ or 2.
2. Although the address field is not used for addressing purposes, B-register address-modification will occur if it is specified. *No non-existent-address ALARM STOP can occur, however.*
3. The Data Processor is capable only of shifting the contents of the A and R registers to the right. In order to shift to the left, therefore, it is necessary for the machine to simulate left shifts by executing circulating right shifts. This is of no other importance to the programmer/coder than that he knows how to determine execution times: for all practical purposes, if the instruction is SLS 0003, the contents of the A register will be shifted and circulated three places to the left. In this particular case, as will be explained below, the number of digit positions through which the contents of the A register is shifted is eight, of which only seven enter into timing considerations. Examination of the flow chart may help to make clearer the description below.

The counter which determines how many digit positions are to be shifted is called the Digit Counter. The Digit Counter is capable of counting only to 20. Thus, for example, if the Digit Counter is set to 13, only seven pulses will be provided to accomplish shifting, one pulse each time the contents of the specified register(s) are shifted to the right. When the Digit Counter reaches 20, no more pulses are provided.

Suppose now that the instruction is SLA 00nn. The Data Processor first determines N_1 , where $N_1 \equiv nn, \text{ modulo } 10$ (that is, N_1 is the least non-negative remainder obtained on dividing nn by 10). The Digit Counter is then set to $10 + N_1$. Hence, the number of digit positions actually shifted is $20 - (10 + N_1) = 10 - N_1$.

Example 1. SLA 0003. $N_1 = 3$. The Digit Counter is set to 13. The contents of rA:00 will be shifted left, and circulated, three places by shifting right, and circulating, seven places.

Digit Counter	(rA)
13	± 1234 56 7890
14	± 0123 45 6789
15	± 9012 34 5678
16	± 8901 23 4567
17	± 7890 12 3456
18	± 6789 01 2345
19	± 5678 90 1234
20	± 4567 89 0123

THE DATA PROCESSOR

Example 2. SLA 0049. $N_1 = 9$. The Digit Counter is set to 19. The contents of rA:00 will be shifted left and circulated nine places by shifting right, and circulating, one place.

Digit Counter	(rA)
19	± 1234 56 7890
20	± 0123 45 6789

Suppose the instruction is SLS 00nn. As in the case of SLA, N_1 is determined. Also, because the operation is SLS, the Data Processor automatically shifts and circulates the contents of the A register one place to the right while it is determining N_1 . The Digit Counter is set to $10 + N_1$, so that the number of additional shifts is $20 - (10 + N_1) = 10 - N_1$. In this way the proper kind of modulo-11 shift is achieved.

Example 3. SLS 0007. $N_1 = 7$. The Digit Counter is set to 17.

Digit Counter	(rA)
Original	± 1234 56 7890
17	0 ±123 45 6789
18	9 0±12 34 5678
19	8 90±1 23 4567
20	7 890± 12 3456

Because the SHIFT LEFT A AND R instruction handles sets of 20 digits, it provides a modulo-20 shift. The Data Processor first determines N_2 , where $N_2 \equiv nn$, modulo 20. The Digit Counter is then set to N_2 , so that the number of digit positions actually shifted is $20 - N_2$.

Example 4. SLT 0018. $N_2 = 18$.

Digit Counter	(rA)	(rR)
18	± 1234 56 7890	A BCDE FG HIJK
19	A K123 45 6789	A OBCD EF GHIJ
20	A JK12 34 5678	A 90BC DE FGHI

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

4. The table below summarizes the times required to execute the various shifting instructions. Fetch time is included. As is noted in the headings, the entries in the table were computed using the following formulas:

$$\text{SLA, SLS: } t = 160 - 5N_1,$$

$$\text{SLT: } t = 210 - 5N_2.$$

N_1 and N_2 were defined in Remark 3.

nn	Time (μs)	
	SLA, SLS ($160 - 5N_1$)	SLT ($210 - 5N_2$)
00	160	210
01	155	205
02	150	200
03	145	195
04	140	190
05	135	185
06	130	180
07	125	175
08	120	170
09	115	165
10	160	160
11		155
12		150
13		145
14		140
15		135
16		130
17		125
18		120
19		115

THE DATA PROCESSOR

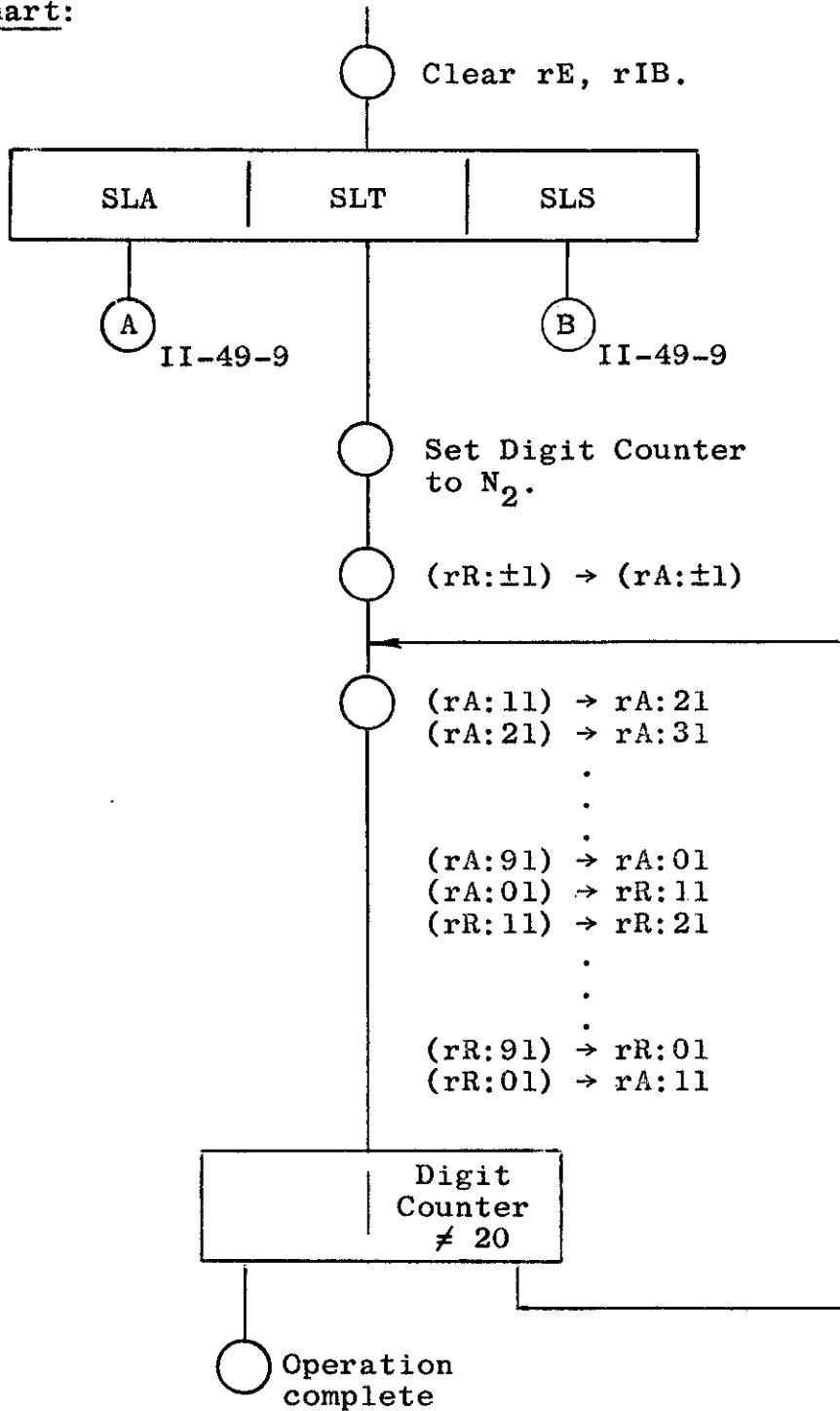
Register status:

Register name	Contents after execution of SLA.	Contents after execution of SLT.																										
A	(rA:±1) _a = (rA:±1) _b (rA:00) shifted as specified.	(rA:±1) _a = (rR:±1) _b (rA:00, rR:00) shifted as specified.																										
R	Unchanged	(rR:±1) _a = (rR:±1) _b (rA:00, rR:00) shifted as specified.																										
D	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> </tr> </table>	±	i	i	i	0	4	9	i	i	n	n	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> </tr> </table>	±	i	i	i	1	4	9	i	i	n	n				
±	i	i	i	0	4	9	i	i	n	n																		
±	i	i	i	1	4	9	i	i	n	n																		
B P	Unchanged (rP) _b + 1	Unchanged (rP) _b + 1																										
C	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	0	4	9	B	[i	i	n	n]	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	1	4	9	B	[i	i	n	n]
i	i	i	0	4	9	B	[i	i	n	n]																
i	i	i	1	4	9	B	[i	i	n	n]																
E	Cleared	Cleared																										

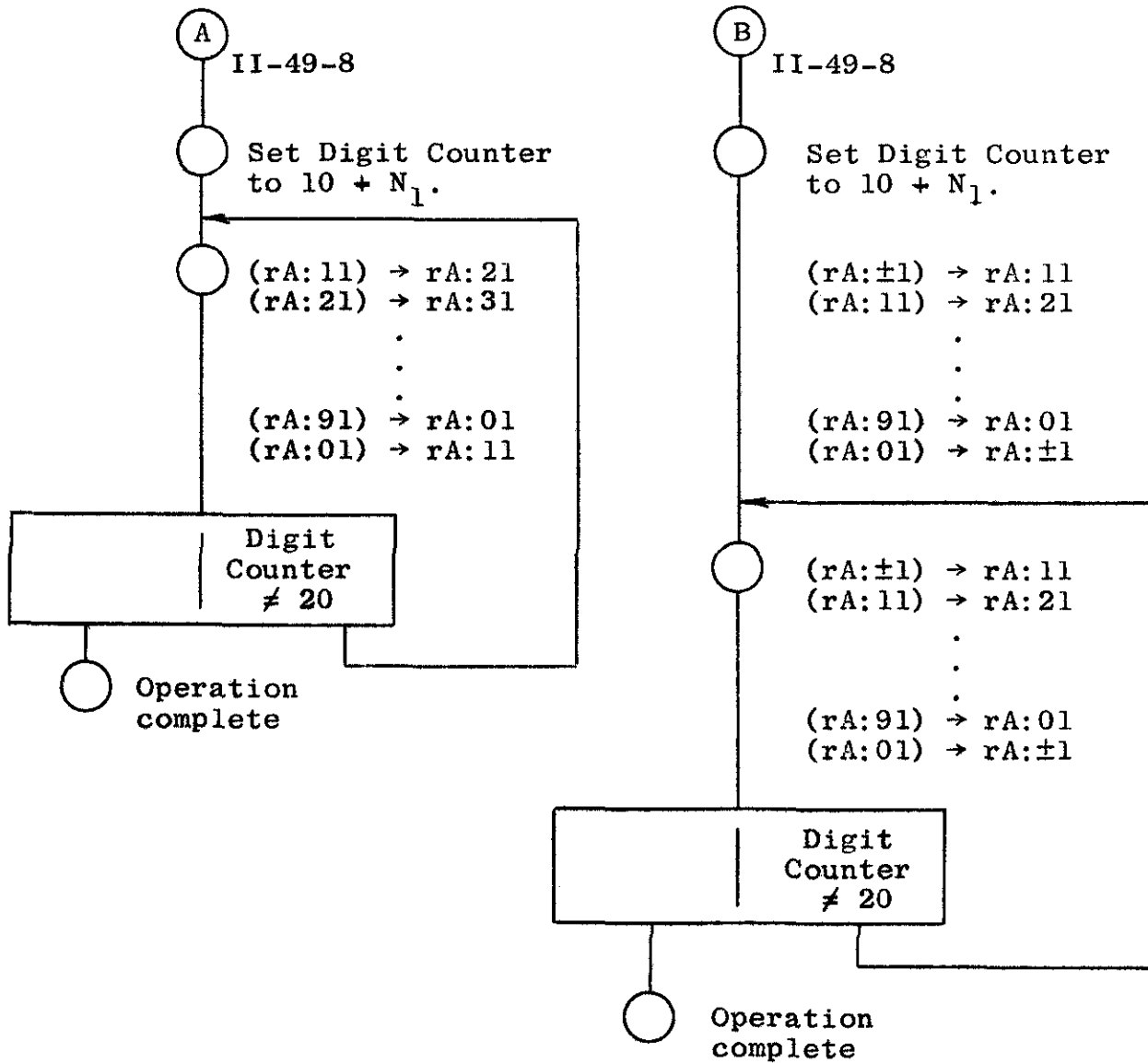
Register name	Contents after execution of SLS.													
A	(rA) shifted as specified.													
R	Unchanged													
D	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">±</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> </tr> </table>	±	i	i	i	2	4	9	i	i	n	n		
±	i	i	i	2	4	9	i	i	n	n				
B P	Unchanged (rP) _b + 1													
C	<table border="1" style="margin: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">9</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">n</td> <td style="padding: 2px;">]</td> </tr> </table>	i	i	i	2	4	9	B	[i	i	n	n]
i	i	i	2	4	9	B	[i	i	n	n]		
E	Cleared													

Description of operation:

Flow chart:



Flow chart (continued):



Introduction

The control console of the DATATRON 220 Electronic Data-Processing System is the control center for the system. By means of neon lamps, indicators, push-button indicators, and organ switches, the status of the System is indicated, and supervisory control over its operation is provided.

The control console is divided into three parts, of which only the central section is of concern to the system operator and/or programmer. The two side panels, mounted behind doors which normally are closed, are for the maintenance or test engineer.

This volume is not concerned with operating procedures and techniques. The reader is referred to the Handbook of Operating Procedures for the DATATRON 220 for such information. Our concern in this section is with three items: the ten PROGRAM CONTROL SWITCHES, the numeric keyboard, and the supervisory printer.

PROGRAM CONTROL SWITCHES

The control console is equipped with ten clear-plastic organ switches. Each switch has a light behind it: the light is on when the switch is on. The status of a switch may be interrogated by the BRANCH CONTROL SWITCH instruction. Although more than one switch may be on at any one time, only one switch may be interrogated with any one instruction.

Flexible manual-control facilities are provided by these switches.

The Keyboard

A numeric keyboard is an integral part of the control console. This keyboard may be activated under program control by use of the KEYBOARD ADD instruction (which see, Page III-08-2).

When the computer is not in RUN status the keyboard may be activated by depressing the KEYBOARD switch.

The Supervisory Printer

Associated with the control console is a character-at-a-time alphanumeric printer. The printer, operating at a rate of 10 characters per second, is capable of printing both numeric and alphabetic information.

Controls are provided to suppress the printing of high-order zeros as well as the translation of words specified as alphanumeric (an alphanumeric word is signaled by the presence of a 2 in the sign-digit position).

The following format controls are provided:

1. The right-hand margin may be set where required to provide an automatic carriage return. Carriage return always includes a single line feed.

THE CONTROL CONSOLE

2. A three-position switch permits the choice of "space," "tab," or "carriage return" action when the end-of-word is sensed.

3. Vertical form-control is provided so that fan-fold paper can feed past the tear line to a predetermined position or positions. These vertical tabs may be set at 2.75-, 5.5-, 8.25-, and 11-inch intervals.

The Supervisory Printer can print all the decimal digits and alphabetic characters; in addition, the following characters can be printed: -, \$, &, *, H, ., ', /, #, %, @, and "space." The computer codes for these characters are shown in Appendix A3.

The Supervisory Printer is the same printer described in Section V, The Paper-Tape System.

The Interval Timer

Attached to the Control Console is a five-digit-position counter which can count to 9999.9 seconds by tenths of a second. The counter may be reset manually to zero.

The Interval Timer is connected, electrically, to the RUN Indicator: when the RUN Indicator is "on," the timer will count.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

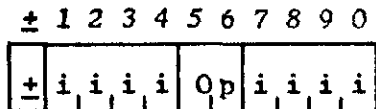
Operation code: 08

Operation name: KEYBOARD ADD

Abbreviation: KAD

Instruction format:

Time (μ s):



fetch: 90
 execute: manual
 total: manual

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

iiii: not relevant to the execution of this instruction.

Op: operation code.

Description of operation:

Summary:

The D register is cleared, following which the computer waits, prepared to accept information entered on the console keyboard. Each digit selected on the keyboard enters the low-order digit position of the D register, shifting the previous contents, including the contents of the sign-digit position, one place to the left; digits shifted out of the sign-digit position are lost. When the ADD key on the keyboard is depressed, the sum of the contents of the A register and the D register is generated. This sum replaces the contents of the A register.

Automatic control is resumed when the ADD key is depressed.

Flow chart:

See page III-08-4.

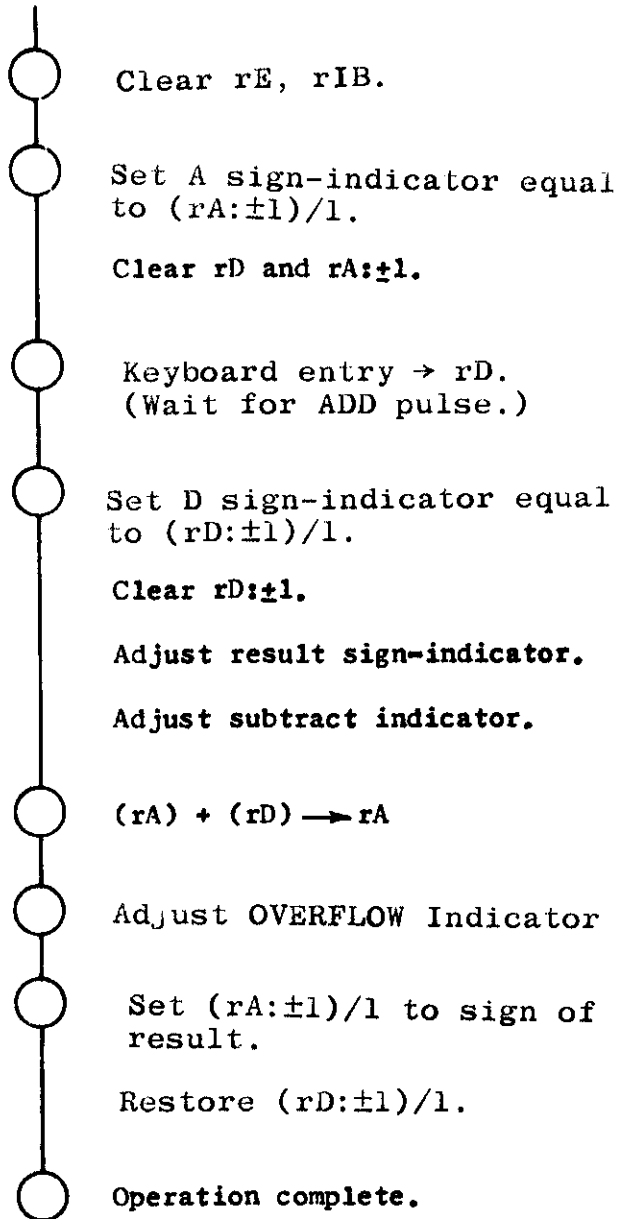
Exceptional conditions: NONE.

Remarks:

1. Although the address-field is not used for addressing purposes, B-register address-modification will occur if it is specified. No non-existent-address ALARM STOP can occur, however.
2. The execution of this instruction can cause arithmetic overflow, in which case the OVERFLOW Indicator is set "on."
3. If $(rA) + (rD) = 0$, the sign of this result is the sign that was in the A register before execution.

Description of operation:

Flow chart:



THE CONTROL CONSOLE

Register status:

Register name	Contents after execution of KAD.																										
A	$(rA)_b + \text{number keyed into rD}$																										
R	Unchanged																										
D	number keyed in																										
B	Unchanged																										
P	$(rP)_b + 1$																										
C	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">8</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">]</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> <td style="text-align: center;"> </td> </tr> </table>	i	i	i	i	0	8	B	[i	i	i	i]													
i	i	i	i	0	8	B	[i	i	i	i]															
E	Cleared																										

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 09

Operation name: SUPERVISORY PRINT-OUT

Abbreviation: SPO

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

fetch: 90

execute: printer speed

total: printer speed

\pm	d	n	n	v	Op	a	a	a	a
-------	---	---	---	---	----	---	---	---	---

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise there will be no such modification.

d: v = 0: d is not relevant to the execution of this instruction.

v = 1: d specifies the number of digit positions to the right of a decimal point provided by the Data Processor. d = 0 means 10.

nn: specifies the number of words to be printed. nn = 00 means 100.

v: variation designator:

v = 1: the Data Processor will insert a decimal point in the place specified by d.

v = 0: no decimal point insertion.

Op: operation code.

aaaa: address of base of location of first word to be printed.

Description of operation:

Summary:

Print, on the Supervisory Printer, nn words from consecutively addressed locations, beginning with (B[aaaa]). The next word is taken from B[aaaa] + 1. And so forth.

If v = 1, d specifies the number of digit positions to the right of a Data-Processor-inserted decimal point.

The abilities to control zero suppression and/or to print the contents of a location exactly as a word appears in core storage--for example, to prevent the translation of an alphanumeric word--are controlled by four switches, two of them on the Control Console, and two of them on the Supervisory Printer. The function of these switches is described below (see Remark 2).

Flow chart:

See page III-09-10.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Input-output ALARM STOP.
3. Not-ready ALARM STOP.

Remarks:

1. $v \neq 1$ has the same effect as $v = 0$.
2. There are four two-position switches which govern the form of output, two of them on the left-hand maintenance panel of the Control Console, and two of them on the Supervisory Printer:
 - A. On the Control Console
 1. HOLD PZT TO ZERO
 2. PUNCH SUPPRESS--LEADING ZEROS
 - B. On the Supervisory Printer
 1. MAP MEMORY--NORMAL
 2. ZERO SUPPRESS--NORMAL

Normally the two switches on the Control Console are used by the maintenance engineer, but because the system operator does have access to them, it is worth describing the effects they can produce. It will be convenient to describe the effects produced by these switches separately from the effects produced by the switches on the Supervisory Printer. What we are concerned with is the following:

A. There are four switch-position combinations of the HOLD PZT TO ZERO and PUNCH SUPPRESS--LEADING ZEROS switches. We need to know the effect of each combination on output from the Data Processor. This "regulated output" will be regarded as input to the Supervisory Printer for, in fact, that is what it is.

B. There are four switch-position combinations of the MAP MEMORY--NORMAL and ZERO SUPPRESS--NORMAL switches. We need to know the effect of each combination on input to the Supervisory Printer.

In what follows it will be convenient to use the following abbreviations:

- A. "PZT = 1" means "HOLD PZT TO ZERO is on".
- B. "PZT = 0" means "HOLD PZT TO ZERO is off".
- C. "PSLZ = 1" means "PUNCH SUPPRESS--LEADING ZEROS is on".
- D. "PSLZ = 0" means "PUNCH SUPPRESS--LEADING ZEROS is off".
- E. "MM = 1" means "MAP MEMORY--NORMAL is in MAP MEMORY position".
- F. "MM = 0" means "MAP MEMORY--NORMAL is in NORMAL position".
- G. "ZS = 1" means "ZERO SUPPRESS--NORMAL is in ZERO SUPPRESS position".
- H. "ZS = 0" means "ZERO SUPPRESS--NORMAL is in NORMAL position".

In the examples which will be cited below, it will be assumed that the SPO instruction calls for a word from the location whose address is L.

It should be noted that the normal setting of the switches on the Control Console is PZT = 0 and PSLZ = 0. If the reader wishes to do so, therefore, he may examine only Cases 4, 5, 6 and 7 below.

Case 1. PZT = 1 and PSLZ = 1.

A. If (L;±1) = 2--which flags (L) as an alphanumeric word--the word will not be recognized and hence not translated as an alphanumeric word because PZT = 1. The regulated output is an 11-digit (numeric) word whose sign digit is 2.

THE CONTROL CONSOLE

B. Because PSLZ = 1, the specification of decimal point insertion by the SPO instruction will be overridden: no decimal point will be emitted by the Data Processor. If $(L:\pm 1) = 0$, and

1. If $(L:00) = 0000\ 00\ 0000$, the regulated output is 00eow, where "eow" means "end-of-word" symbol. The leftmost 0 is the sign digit of L. The rightmost 0 is a substitute for $(L:00)$; or

2. If $(L:k1) \neq 0$, $k \neq 0$; and if $(L:j1) = 0$, $j = 1, 2, \dots, k - 1$; then the regulated output is 0 (for $(L:\pm 1)$) followed by $(L:k1)$ followed by the remaining digits of the word. An eow symbol terminates the regulated output.

Example 1. $(L) = 0\ 0041\ 42\ 4344$. $(L:j1) = 0$ for $j = 1, 2$. $(L:k1) \neq 0$ for $k = 3$. The regulated output is 041424344eow.

C. Hence, if $(L:\pm 1) \neq 0$, the regulated output is (L) .

Case 2. PZT = 1 and PSLZ = 0.

A. Same as Case 1, A.

B. Because PSLZ = 0, the code for a decimal point will be emitted by the Data Processor if it is specified by the SPO instruction ($v = 1$). There will be no suppression of 0's.

Example 2. $(L) = 0\ 0041\ 42\ 4344$. $v = 1$, $d = 3$. The regulated output is 00041424.344eow.

Example 3. $(L) = 2\ 0041\ 42\ 4344$. $v = 1$, $d = 4$. The regulated output is 2004142.4344eow.

Case 3. PZT = 0 and PSLZ = 1.

A. Because PZT = 0, if $(L:\pm 1) = 2$, the word will be recognized and translated as an alphanumeric word.

Example 4. $(L) = 2\ 0041\ 42\ 4344$. The regulated output is 2spABCDEow, where "sp" means "space".

B. Same as Case 1, B. (See also Example 1.)

Example 5. (L) = 1 0041 42 4344. The regulated output is 10041424344eow.

Example 6. (L) = 3 0041 42 4344. The regulated output is 30041424344eow.

Case 4. PZT = 0 and PSLZ = 0.

A. Same as Case 3, A.

B. Same as Case 2, B.

In Examples 7 through 11, let $v = 1$ and $d = 3$ in the SPO instruction.

Example 7. (L) = 0 0041 42 4344. The regulated output is 00041424.344eow.

Example 8. (L) = 1 0041 42 4344. The regulated output is 10041424.344eow.

Example 9. (L) = 2 0041 42 4344. The regulated output is 2spABCDEow.

Example 10. (L) = 3 0041 42 4344. The regulated output is 30041424.344eow.

Example 11. (L) = 0 0000 00 0000. The regulated output is 00000000.000eow.

Now we need to consider what happens to the regulated output from the Data Processor. From the point of view of the Supervisory Printer, regulated output from the Data Processor is input to the Supervisory Printer. In what follows the word "input" refers to such regulated output; the word "output" means "what is printed by the Supervisory Printer."

First, recall that the Supervisory Printer is equipped with a three-position FORMAT switch. The position of the switch determines what action will be caused when an end-of-word symbol is recognized: TAB, SPACE, or CARRIAGE RETURN.

Case 5. MM = 1.

A. If MM = 1, it does not matter whether ZS = 1 or ZS = 0.

B. Print the regulated output exactly as it is presented for input. The eow symbol causes the action specified by the position of the FORMAT switch.

THE CONTROL CONSOLE

(See Examples 7 through 11 for what has been described as normal regulated output, that is, regulated output when HOLD PZT TO ZERO and PUNCH SUPPRESS--LEADING ZEROS switches are both off.)

Case 6. MM = 0 and ZS = 0.

A. If the sign digit of the input word is odd, a - (minus) will be printed in place of the sign digit. Then all the remaining digits of the input word are printed, and eow action occurs.

Example 12. v = 0. The input word is 30041424344eow. The output is -0041424344, followed by eow action.

Example 13. v = 1, d = 3. The input word is 10041424.344eow. The output is -0041424.344, followed by eow action.

B. If the sign digit of the input word is 2, this signals the Supervisory Printer that the word is alphanumeric. The 2 is not printed nor is any character--e.g., space--substituted for it. The remaining characters--if PZT = 0--or digits--if PZT = 1--are printed just as they are presented. There is no eow action, because the word is flagged as alphanumeric.

Example 14. The input word is 20041424344eow. The output is 0041424344.

Example 15. The input word is 2spABCDEow. The output is spABCD.

C. If the sign digit of the input word is even, but $\neq 2$, a "space" is "printed" in place of the sign digit. Then all the remaining digits of the input word are printed and eow action occurs.

Example 16. The input word is 40041424344eow. The output is sp0041424344, followed by eow action.

Case 7. MM = 0 and ZS = 1.

A. Suppress all leading 0's, including sign, in the input word. Spaces are substituted for suppressed 0's. Print the remaining part of the input word, beginning with the first non-zero digit. Eow action occurs.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Example 17. The input word is 00041424344eow. The output is spspsp41424344, followed by eow action.

Example 18. $v = 0$, $d = 9$. The input word is 00.041424344eow. The output is spsp.041424344, followed by eow action.

B. Same as Case 6, B. (See also Examples 14 and 15.)

C. If the sign digit of the input word is even, but $\neq 0$ or 2, a "space" is "printed" in place of the sign digit. Because the sign digit is different from 0, it is recognized as such for control of zero suppression. The remainder of the input word is printed as it is presented. Eow action occurs.

Example 19. The input word is 40041424344eow. The output is sp0041424344, followed by eow action.

THE CONTROL CONSOLE

Register status:

Register name	Contents after execution of SPO.											
A	Unchanged											
R	Unchanged											
D	$(B[aaaa] + nn - 1)^*$											
B	Unchanged											
P	$(rP)_b + 1$											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 20px;">d</td> <td style="width: 20px;">0</td> <td style="width: 20px;">0</td> <td style="width: 20px;">v</td> <td style="width: 20px;">0</td> <td style="width: 20px;">9</td> <td style="width: 20px;">$B[aaaa]$</td> <td style="width: 20px;">$+$</td> <td style="width: 20px;">nn</td> <td style="width: 20px;">$-$</td> <td style="width: 20px;">1^*</td> </tr> </table>	d	0	0	v	0	9	$B[aaaa]$	$+$	nn	$-$	1^*
d	0	0	v	0	9	$B[aaaa]$	$+$	nn	$-$	1^*		
E	$B[aaaa] + nn - 1^*$											

Register name	Contents if non-existent-address ALARM STOP occurs.										
A	Unchanged										
R	Unchanged										
D	Last word printed										
B	Unchanged										
P	$(rP)_b + 1$										
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 20px;">d</td> <td style="width: 20px;">*</td> <td style="width: 20px;">*</td> <td style="width: 20px;">v</td> <td style="width: 20px;">0</td> <td style="width: 20px;">9</td> <td style="width: 20px;">*</td> <td style="width: 20px;">*</td> <td style="width: 20px;">*</td> <td style="width: 20px;">*</td> </tr> </table>	d	*	*	v	0	9	*	*	*	*
d	*	*	v	0	9	*	*	*	*		
E	****										

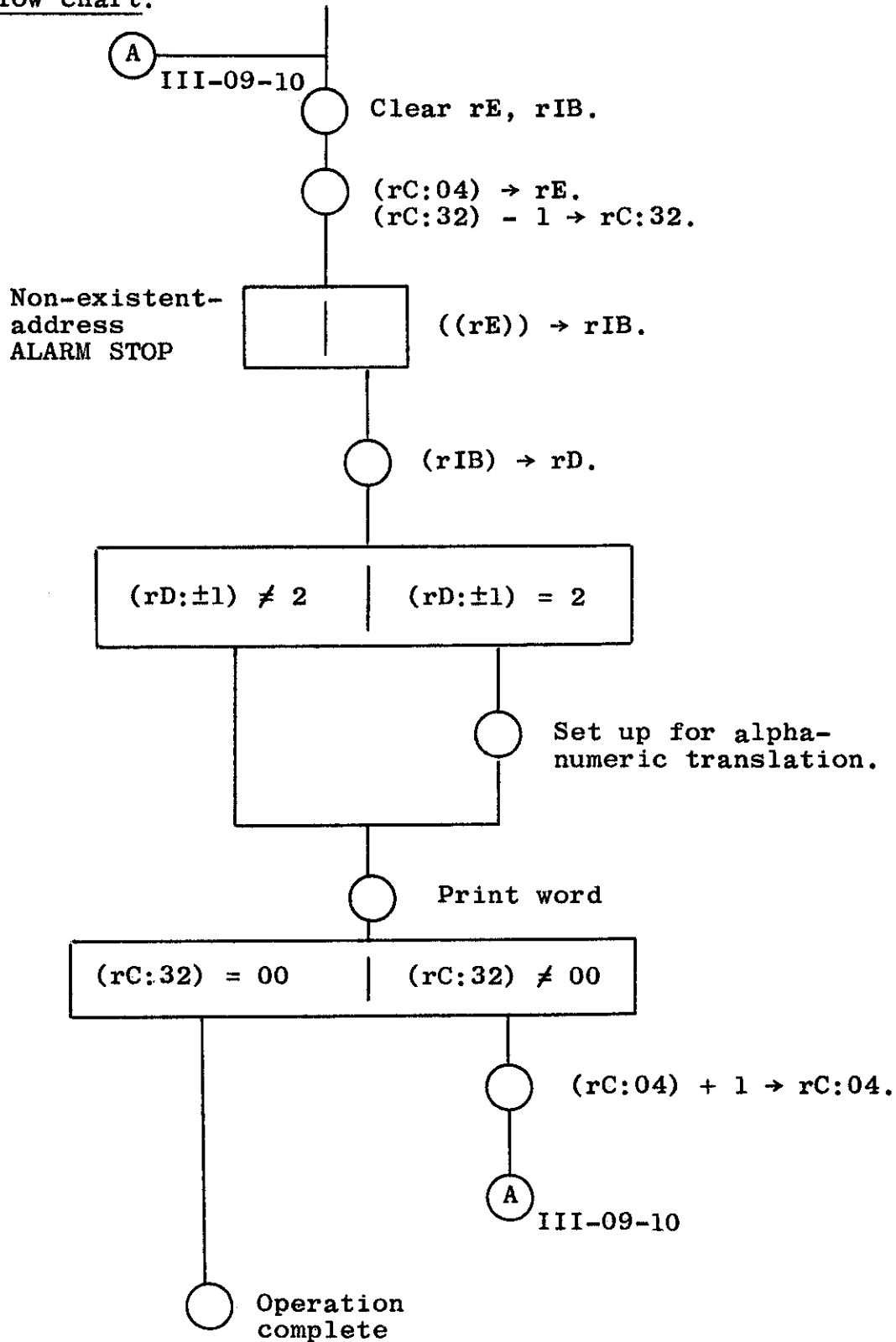
*If $nn = 00$, $B[aaaa] + nn = B[aaaa] + 100$.

**See flow chart.

****Address of location causing ALARM STOP.

Description of operation:

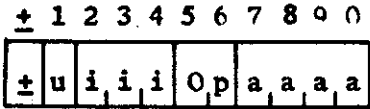
Flow chart:



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: BRANCH, CONTROL SWITCH

Instruction format:



Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates PROGRAM CONTROL SWITCH to be tested.
- iii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location of alternate instruction.

Operation code: 38

Abbreviation: BCS

Time (ps):

No branch:

fetch: 90
 execute: 15
 total: 105

Branch:

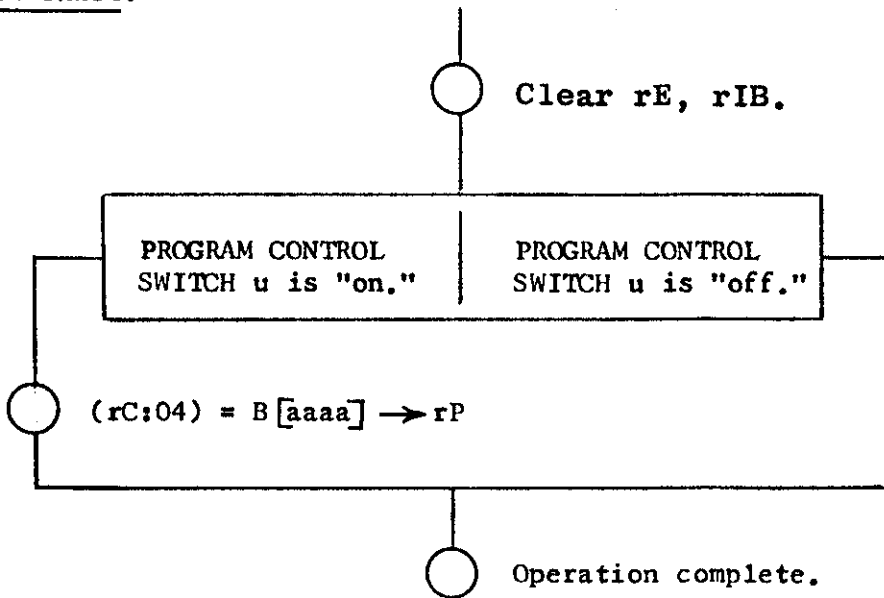
fetch: 90
 execute: 35
 total: 125

Description of operation:

Summary:

If PROGRAM CONTROL SWITCH u is "on," branch to location B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. If SWITCH u is "off," control continues in sequence.

Flow chart:



Exceptional conditions: NONE

Remarks:

Register status:

Register name	Contents after execution of BCS.											
A R	Unchanged "											
D	<table border="1" style="display: inline-table;"> <tr> <td style="text-align: center;">+</td> <td style="text-align: center;">u</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">3</td> <td style="text-align: center;">8</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> <td style="text-align: center;">a</td> </tr> </table>	+	u	i	i	i	3	8	a	a	a	a
+	u	i	i	i	3	8	a	a	a	a		
B P	Unchanged B[aaaa], if SWITCH u is "on." (rP) _b + 1, if SWITCH u is "off."											
C	<table border="1" style="display: inline-table;"> <tr> <td style="text-align: center;">u</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">i</td> <td style="text-align: center;">3</td> <td style="text-align: center;">8</td> <td style="text-align: center;">B</td> <td style="text-align: center;">[aaaa]</td> </tr> </table>	u	i	i	i	3	8	B	[aaaa]			
u	i	i	i	3	8	B	[aaaa]					
E	Cleared											

Introduction

Auxiliary-storage capacity is provided through the medium of magnetic tape. The components of the magnetic-tape system are a Magnetic-Tape Control Unit, a Magnetic-Tape Storage Unit, and a Datafile.

Magnetic tape is "edited" prior to its initial use to insure a flawless writing medium. It may be re-edited at any time if that is desired.

Additional protection of information recorded on magnetic tape is afforded by the use of a ^{File Protect Ring}~~write ring~~: it is possible to write or edit only on reels -- used by the tape storage unit -- which are NOT equipped with a ^{FILE PROTECT RING}~~write ring~~. On the Datafile there is a "write lever" for each piece of tape; writing can occur only when the lever is in the write position.

Information is recorded on tape in blocks whose lengths may vary from ten to one hundred words; no two adjacent blocks need to be the same length. From one to ten blocks may be written or read with one instruction.

The first data word of a block is regarded as its address. It is possible to search¹ for a block with a given address. Once initiated, the searching operation is carried out independently of Computer control.

1 By "searching" is meant the ability to locate in the tape file a specified block without the necessity of passing a substantial part of the "file" -- i.e., the blocks which are written on tape -- or all of it through the Computer in order to identify the desired block. The search operation will be defined more explicitly in context below.

Any one of the first ten words may identify a category. It is possible to "scan" for blocks belonging to a specified category. Once initiated, the scanning operation is carried out independently of the Computer.

(Search and scan are executed in distinct and different manners. The nature of the execution of each will be described in detail below.)

Information is transferred between the Computer and magnetic tape at a nominal rate of 25,000 digits per second.

The Magnetic-Tape Storage Unit

The Magnetic-Tape Storage Unit handles reels of magnetic tape, each reel containing up to 3500 feet of tape; the reels are 10 1/2 inches in diameter. Although the number of tape storage units in a system cannot exceed ten, the number of reels of tape which can be used is limited only by the magnitude of the task to be accomplished.

Recording on magnetic tape is accomplished by a read-write mechanism (called the "read-write head" or the "head") over which tape is moved as it is wound on or unwound from the feed and take-up reels. Actually, there are two heads on each unit, fixed in position with respect to each other, to permit the recording of two parallel lanes of information on each piece of tape. The lanes are distinguished as the "even" and "odd" lanes, or lanes 0 and 1, respectively.

The tape drive mechanism is capable of moving tape in either of two directions, from feed reel to take-up reel---the "forward" direction---or from take-up reel to feed reel---the "backward" direction. The direction in which tape will be moved is a function of the operation being performed; it will be specified as each operation is described.

The Datafile

The Datafile unit holds fifty lengths of magnetic tape, each 250 feet in length, and each hanging freely in its own static-free bin. The bins are parallel to each other, so that the tapes are parallel to one another along their lengths.

Recording on any tape in a Datafile is identical with recording on a tape storage unit. The head mechanism, however, is on a traverse rod on which it is free to move: under program control the head may be positioned to perform a specified operation on any particular one of the 100 lanes in a unit. Each lane in a Datafile is distinguished uniquely by a two-digit number, in order, from 00 through 99. If we consider the tapes as numbered from 1 through 50, in order, from left to right, then tape number k contains lanes number $2k - 2$ and $2k - 1$.

Logically, the Datafile and tape storage units are indistinguishable to the Magnetic-Tape Control Unit. For this reason, in the description which follows the text will not distinguish between the units unless some characteristic---capacity, for example---requires mention of the difference.

Tape Format

Magnetic tape used in a DATATRON 220 system has provision for recording two lanes of information, parallel to one another, along the length of each tape. In each lane six channels are recorded: channels one through four are used to record decimal-digital information in the 8, 4, 2, 1 - binary-code; channel five is used to provide an odd-parity bit; and channel six is used for control purposes.

In each lane the recording density is approximately 208 digits per linear inch of tape. Tape is transported at the rate of 120 inches per second. The nominal transfer rate, therefore, is approximately 25,000 digits per second.

The physical beginning and end of a piece of tape are marked by reflective strips. The reflective strip is a piece of pressure-sensitive tape which is affixed manually near each end of the tape. It is not possible to drive tape past these reflective strips.

The magnetic beginning-of-tape and end-of-tape are recorded during the editing process. During the editing process magnetic beginning-of-tape is overwritten by flaw markers to insure that information cannot be recorded too near the physical beginning of the tape. The magnetic end-of-tape area is sufficiently long to permit both the completion of any (initial) writing operation in progress when it is encountered as well as the (initial) writing of necessary end-of-tape control blocks before physical end-of-tape is encountered.

A block (of information) on magnetic tape is defined as the information which is recorded between two inter-block gaps (except for the last block, which is followed by blank¹ tape in case all of the tape is not used, or by the magnetic end-of-tape). Associated with each block is a "preface word." The preface word contains the two-digit preface which specifies how many words are contained in the block with which the preface is associated. The manner of reading or writing the preface will be described below.

The minimum block size is ten words; the maximum block size is 100 words. An attempt to write a block of information whose length is less than ten words will result in an improper-block-length ALARM STOP. The format of the writing instructions precludes specification of a block length in excess of 100 words.

A typical k-word-long block layout is shown in figure IV - Intro - 1. From the figure we obtain the information that a block of k words occupies space for $80 + 12k$ digits, including the space for one inter-block gap. However, only $11k$ digits represent data useful to a program. Table IV - Intro - 1 shows how the percentage of useful information varies with block size.

1 "Blank" tape is tape on which no information is recorded. The subject will be discussed at length, in context, below.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

	<u>CHARACTER COUNT</u>	
	<u>Identification</u>	<u>Information and Timing</u>
Inter-block gap	_____ 39	
Preface "word"	_____ 8	
Preface gap	_____ 9	
Timing character	_____ 1	
first data word (address)	_____ 11	
Address gap	_____ 3	
Timing character	_____ 1	
Second data word	_____ 11	
Timing character	_____ 1	
Third data word	_____ 11	
	_____ k-4 *	
	_____ 11(k-4)**	
Timing character	_____ 1	
k th data word	_____ 11	
End-of-block character	_____ 1	
Erase gap	_____ 20	
	_____ 80	_____ 12k

* Timing characters for words 4 through k-1.
 ** Digits for words 4 through k-1.

Figure IV - Intro - 1. Block format
(not to scale)

THE MAGNETIC-TAPE SYSTEM

Words per Block (k)	Useful Data (11k)	Total Digit Count (80 + 12k)	% of Useful Data
10	110	200	55.0
20	220	320	68.8
30	330	440	75.0
40	440	560	78.7
50	550	680	80.9
60	660	800	82.5
70	770	920	83.7
80	880	1040	84.6
90	990	1160	85.3
100	1100	1280	86.0

Table IV-Intro-1. Percentage of useful data in a block

Since approximately 208 digits can be recorded per linear inch of tape, the number of digit spaces per 100 feet of tape in one lane is:

$$Q_{100} \approx 208 \frac{\text{digits}}{\text{inch}} \times 12 \frac{\text{inches}}{\text{foot}} \times 100 \text{ feet}$$

$$\approx 250,000 \text{ digit spaces}$$

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Table IV - Intro - 2 shows the number of blocks and the number of words of useful data which can be recorded in 100 feet of tape in one lane.

Block size (k words)	Number of blocks $[250,000/(80+12k)]=B$	Number of words of useful data ($W = B \times k$)
10	1,250.00	12,500
20	781.25	15,625
30	568.18	17,045
40	446.43	17,857
50	367.65	18,383
60	312.50	18,750
70	271.74	19,022
80	240.38	19,230
90	215.52	19,397
100	195.31	19,531

Table IV-Intro-2. Number of blocks and words of useful data recorded in 100 feet of tape in one lane.

Table IV-Intro-3 shows how the capacity of a tape storage unit containing a 3500-foot length of tape varies with block size.

Block size (k words)	Per lane		Per tape	
	Blocks	Words	Blocks	Words
10	43,750	437,500	87,500	875,000
20	27,344	546,880	54,688	1,093,760
30	19,886	596,580	39,772	1,193,160
40	15,625	625,000	31,250	1,250,000
50	12,868	643,400	25,736	1,286,800
60	10,938	656,280	21,876	1,312,560
70	9,511	665,770	19,022	1,331,540
80	8,413	673,040	18,826	1,346,080
90	7,543	678,870	15,086	1,357,740
100	6,836	683,600	13,672	1,367,200

Table IV-Intro-3. Magnetic-Tape Storage Unit capacity

THE MAGNETIC-TAPE SYSTEM

Table IV-Intro-4 shows how the capacity of a Datafile unit varies with block size. Each of the 50 tapes in such a unit is 250 feet long.

Block size (k words)	Per Lane		Per Tape		Per Unit	
	Blocks	Words	Blocks	Words	Blocks	Words
10	3,125	31,250	6,250	62,500	312,500	3,125,000
20	1,953	39,060	3,906	78,120	195,300	3,906,000
30	1,420	42,840	2,840	85,680	142,000	4,284,000
40	1,116	44,640	2,232	89,280	111,600	4,464,000
50	919	45,950	1,838	91,900	91,900	4,595,000
60	781	46,860	1,562	93,720	78,100	4,686,000
70	679	47,530	1,358	95,060	67,900	4,753,000
80	601	48,080	1,202	96,160	60,100	4,808,000
90	539	48,510	1,078	97,020	53,900	4,851,000
100	488	48,800	976	97,600	48,800	4,880,000

Table IV-Intro-4. Datafile Capacity

The Editing Process

The editing procedure consists of an examination of magnetic tape to determine whether there are any imperfections---i.e., flaws---on it, and the subsequent "deletion" of the flaws by preceding them with flaw markers. A flaw marker consists of "zeros" in channels one through four, "ones" in channels five and six (thus, 11 0000). A sufficient number of flaw markers are recorded preceding a flaw so that any block which is being written when the flaw-marked area is encountered can be completely written before the actual flaw is encountered. Before the next block is written, the remainder of the flaw-marked area and the flaw are skipped.

Any magnetic-tape unit can be provided with the ability to edit tape by the inclusion in it of the "edit packages." It is mandatory, of course, that at least one unit have the ability to edit tape.

The procedure for editing tape in a Datafile is essentially the same as that for the tape storage unit. It is necessary to distinguish two cases, however: "brand new" tape, and tape which has been used (i.e., which is to be re-edited).

If the tape has been used, it should bear a label---that is, an identification block---which can be verified, under program control, as identifying a tape which it is safe to edit. After verification, necessary monitoring information and/or directions to the operator can be printed on the Supervisory Printer, following which the editing procedure can be initiated.

If the tape is brand new, however, it will not bear a label for interrogation. But it is possible to distinguish tape which has no information written on it: if a MAGNETIC-TAPE READ (MRD) instruction is issued to a tape which is blank, tape motion will be started and continued in an effort to find written information; thus, tape will be moved to the end, unless there is manual intervention.

In addition, brand new tape may not bear the reflective strips which are used to define physical beginning and end. It is necessary to affix the strips if they are not present. Let us assume that this has been accomplished.¹

Suppose also that appropriate monitoring information and/or directions to the operator are printed on the Supervisory Printer in this case too.

1 A procedure for doing this is described in the Handbook of Operating Procedures.

The initiation of the editing procedure begins with the release by the operator of several interlocks whose purposes include verification that the tape unit is in write status as well as protection against accidental editing. The precise sequence of steps to initiate editing is described in the Handbook of Operating Procedures. The release of the interlocks causes control to be transferred to the unit on which editing is to take place: both the Computer and the Magnetic-Tape Control Unit are released for other use when the interlocks are released. That is to say, once initiated, editing is carried out independently of both the Magnetic-Tape Control Unit and the Computer.

When the EDIT button is depressed, tape movement starts in the forward direction. After reaching the physical end-of-tape marker, the direction of motion is reversed. Tape movement ceases--- and the editing process has been completed---when the physical beginning-of-tape is reached. During these two passes the following are accomplished:

1. At the beginning of the forward pass, magnetic beginning-of-tape (10 1111) is written in both lanes simultaneously.
2. During the remainder of the forward pass, the tape is "erased."
3. At the beginning of the backward pass, magnetic end-of-tape (10 1111) is written in both lanes simultaneously.
4. During the remainder of the backward pass, whenever a flaw is discovered in either lane, flaw

markers are written simultaneously in both lanes.

The magnetic beginning-of-tape appears to be covered with flaws: it is overwritten with flaw markers.

Of course, a tape label should be written after the editing process so that the tape may be identified, under program control, as one which has been edited and which is ready for initial writing.

Writing on Newly-Edited Tape

The establishment of format on newly-edited tape is accomplished by execution of MAGNETIC-TAPE INITIAL WRITE (MIW) or MAGNETIC-TAPE INITIAL WRITE, RECORD (MIR) instructions. These instructions write the preface associated with each block as well as the block itself. MIW is used to write from one to ten blocks, all of which contain the same number of words; MIR is used to write from one to ten blocks, no two of which need to contain the same number of words.

Two kinds of blocks are used to provide terminal-condition control: one is called an "end-of-tape" block, the other a "control" block. Except for its contents, each of these blocks is completely defined by the manner in which it is regarded during the execution of instructions. Table IV-Intro-5a defines the end-of-tape block; Table IV-Intro-5b defines the control block. The reader is cautioned that some of the defining conditions will become (more) meaningful only after he has a grasp of the manner in which instructions are executed.

The end-of-tape block is a one-word block---i.e., its preface is 01---which occupies the space of a ten-word block when it is written on tape. An end-of-tape block is the only block which can

be written if the preface specified is less than ten; unless the preface is 01---when it is specified as less than ten---an improper-block-length ALARM STOP will occur.

The first word of the end-of-tape block is the only word of the block which contains information relevant to the program (the rest of the block written on tape is an "erase block" which automatically is supplied by the tape control unit). The format of the relevant word---it is called the control word---is the following: \pm ii aaaa bbbb. The sign digit may be used to specify B-register address-modification of bbbb when the control word is read; ii are not relevant; aaaa is the address of the location in which will be stored information enabling the program to determine the circumstances in which the end-of-tape block is encountered (see Table IV-Intro-5a), and B[bbbb] is the address of an alternate instruction, that is, the address of the location to which control is transferred when the block is encountered (again, see Table IV-Intro-5a for details).

A control block is a uniquely identified block, namely any block---with the exception of an end-of-tape block---with a 7 in the sign-digit position of the address word.¹ End-of-tape blocks cannot also be used as control blocks: if an end-of-tape block has a 7 in the sign-digit position of the control word, the 7 will not be recognized as designating a control block.

1 As will be seen below, the sign-digit position of the address word cannot be included in the search key; that is, the contents of the sign-digit position cannot be used as part of the address of a block. The sign-digit is also excluded from scan key.

Operation	Description of effect on encountering end-of-tape block
<p>MAGNETIC-TAPE INITIAL WRITE (MIW)</p> <p>MAGNETIC-TAPE INITIAL WRITE, RECORD (MIR)</p>	<p>Not applicable, since tape is newly edited.</p>
<p>MAGNETIC-TAPE OVERWRITE (MOW)</p> <p>MAGNETIC-TAPE OVERWRITE, RECORD (MOR)</p>	<p>If prefaces match, end-of-tape block is overwritten just like any other block.</p> <p>If prefaces do not match:</p> <ol style="list-style-type: none"> 1. Terminate the operation; 2. Position tape so "next" block can be written; 3. Store (rP) in aaaa:04 and (rC:04) in aaaa:64 and 4. Transfer control to B[bbbb] (i.e., B[bbbb] → rP).
<p>MAGNETIC-TAPE READ (MRD)</p> <p>MAGNETIC-TAPE READ, RECORD (MRR)</p>	<ol style="list-style-type: none"> 1a. If MRD, no words are transferred to storage; 1b. If MRR, the preface is transferred to storage; 2. Terminate the operation; 3. Position tape so "next" block can be read; 4. Store (rP) in aaaa:04 and (rC:04) in aaaa:64; and 5. Transfer control to B[bbbb] (i.e., B[bbbb] → rP).

Table IV-Intro-5a. Defining the End-of-Tape Block

THE MAGNETIC-TAPE SYSTEM

Operation	Description of effect on encountering end-of-tape block
MAGNETIC-TAPE SEARCH (MTS) MAGNETIC-TAPE FIELD SEARCH (MFS)	<ol style="list-style-type: none"> 1. When tape is moving forward: <ol style="list-style-type: none"> a. Terminate the operation; and b. Position tape so end-of-tape block can be read. 2. When tape is moving backward the end-of-tape block is treated like a normal block.
MAGNETIC-TAPE SCAN (MTC) MAGNETIC-TAPE FIELD SCAN (MFC)	<ol style="list-style-type: none"> 1. Terminate the operation; and 2. Position tape so end-of-tape block can be read.
MAGNETIC-TAPE POSITION FORWARD (MPF) MAGNETIC-TAPE POSITION BACKWARD (MPB) MAGNETIC-TAPE POSITION AT END OF INFORMATION (MPE)	Treated like a normal block.
MAGNETIC-TAPE REWIND (MRW) MAGNETIC-TAPE REWIND, DEACTIVATE (MDA)	Rewind ignores all information written on tape.

Table IV-Intro-5a (concluded). Defining the End-of-Tape Block

The last word of a control block is a control word which has the same format as the control word in an end-of-tape block. The control word in a control block serves the same function as the control word in an end-of-tape block (see Table IV-Intro-5b for details).

When a MIW or MIR instruction is issued, the writing of the first preface does not begin until an amount of blank tape equivalent to inter-block gap is passed. The writing of each subsequent preface is subject to the same restriction: after the end-of-block character and erase gap are written---or sensed, if another MIW or MIR instruction is issued---writing is inhibited until an amount of tape equivalent to inter-block gap is passed. In this manner, writing in the flaw-marked area at the beginning of tape is prevented; also, writing over a flaw is prevented, since a flaw-marked area preceding an imperfection is long enough to permit the completion of the writing of any maximum-length block if the flaw-marked area is encountered while writing is in process.

The magnetic-tape End-of-Tape Indicator is set "on" at the completion of the operation if magnetic end-of-tape is encountered during the execution of a MIW or MIR instruction. The writing operation will be completed: during the editing process the amount of tape marked as magnetic end-of-tape is more than sufficient to permit the completion of any instruction which may be in progress at the time magnetic end-of-tape is encountered. In fact, the area marked as magnetic end-of-tape is long enough to permit the writing of control information at the end of every tape, as well.

THE MAGNETIC-TAPE SYSTEM

Operation	Description of effect on encountering control block
MAGNETIC-TAPE INITIAL WRITE (MIW) MAGNETIC-TAPE INITIAL WRITE, RECORD (MIR)	Not applicable, since tape is newly edited.
MAGNETIC-TAPE OVERWRITE (MOW) MAGNETIC-TAPE OVERWRITE, RECORD (MOR)	Treated like a normal block.
MAGNETIC-TAPE READ (MRD) MAGNETIC-TAPE READ, RECORD (MRR)	<ol style="list-style-type: none"> 1. All words but the control word---the last word---go to storage; 2. Terminate the operation; 3. Position tape to read the next block; 4. Store (rP) in aaaa:04 and (rC:04) in aaaa:64; and 5. Transfer control to B[bbbb] (i.e., B[bbbb] → rP).
MAGNETIC-TAPE SEARCH (MTS) MAGNETIC-TAPE FIELD SEARCH (MFS)	Treated like a normal block.
MAGNETIC-TAPE SCAN (MTC) MAGNETIC-TAPE FIELD SCAN (MFC)	<ol style="list-style-type: none"> 1. Terminate the operation; and 2. Position tape so control block can be read.
MAGNETIC-TAPE POSITION FORWARD (MPF) MAGNETIC-TAPE POSITION BACKWARD (MPB) MAGNETIC-TAPE POSITION AT END OF INFORMATION (MPE)	Treated like a normal block.
MAGNETIC-TAPE REWIND (MRW) MAGNETIC-TAPE REWIND, DEACTIVATE (MDA)	Rewind ignores all information written on tape.

Table IV - Intro - 5b. Defining the Control Block

Every MIW and MIR instruction must be followed by a MAGNETIC-TAPE INTERROGATE END-OF-TAPE, BRANCH (MIE) instruction to determine whether magnetic end-of-tape was encountered by the last MIW or MIR instruction. If magnetic end-of-tape was encountered, a marker must be written to identify the end of information. A procedure for establishing a control marker, that is, for bounding a file, is described in the Handbook of Programming and Coding Techniques.

Search

MAGNETIC-TAPE SEARCH (MTS) is an instruction which provides the ability to locate in a tape file a specified block. The block specified is the one whose first word-- the "address" of the block -- is identical with a specified search key. The sign digit is excluded from the address, that is, from the search key. Once initiated, searching is done under direction of the Magnetic-Tape Control Unit, independently of the Computer.

The searching operation is executed as follows: tape is moved first in the forward direction until a block is encountered whose address is greater than or equal to the search key.¹ The direction of tape motion is then reversed; tape moves backward until one of two events occurs:

1. A block is found whose address is identical with the search key. In this case the operation terminates with tape positioned so that the head can read the block whose address is identical with the search key.

1 In case an end-of-tape block is encountered during a searching operation, the operation is modified as indicated in Table IV Intro-5a.

2. A block is found whose address is less than the search key. In this case, the direction of tape motion is again reversed, and tape is moved in the forward direction. Tape movement stops with the head near the end of the block whose address is less than the search key. The head is in position to read the next block, that is, the first block whose address is greater than the search key.

It is apparent from this last statement that a searching operation may terminate with the tape positioned so the head can read a block whose address is different from the search key. Hence, it is necessary to verify, under program control, that the block sought has been found.

It should be noted, also, that the nature of the searching operation imposes one important requirement on the structure of a file: the blocks which are written in any lane must be arranged in order of increasing addresses, from beginning to end, if the searching operation is to be used. Techniques for partitioning a lane so that it may contain a part or all of more than one file must also order each file within partitions.

It is possible to execute a search instruction using a partial-word search key. The definition of the partial-word boundaries, sL, must be pre-set in the two high-order digit positions in the B-register (that is, sL = (rB:82)). The instruction is MAGNETIC-TAPE FIELD SEARCH (MFS).

Scan

MAGNETIC-TAPE SCAN (MTC) is an instruction which provides the ability to locate blocks having a code which categorizes the blocks

as belonging to a certain class; for example, from an accounts-receivable file we may wish to examine the record of every account which is delinquent.

The category code of a block may be in any one of the first ten words of the block. The sign-digit position of a word is excluded from the category code.

The scanning operation is executed as follows: tape is moved in the forward direction in an attempt to locate a block whose category code is identical with the scan key. The operation will terminate when either of two events occurs:

1. A block is found whose category code is identical with the scan key. The operation terminates with the head in position to read the block whose category code is identical with the scan key. Or,
2. A control block or end-of-tape block is encountered, in which case the operation terminates with the head in position to read the control or end-of-tape block.

Note that a scanning operation can terminate without having located a sought-for block. Hence, it is necessary to verify, under program control, that a sought-for block has been found.

It is possible to execute a scanning operation using a partial-word scan key. The definition of the partial-word boundaries, sL, must be pre-set in the two high-order digit positions in the B register (that is, sL=(rB:82)). The instruction is MAGNETIC-TAPE FIELD SCAN (MFC).

Reading

It is possible to read from one to ten blocks with one instruction. If these blocks are all the same length, a MAGNETIC-TAPE READ (MRD) instruction is executed. The first word of the first block read---the address of that block---is written in the location whose address is specified in the MRD instruction. Succeeding words of the first, and all following blocks, are written into consecutively-addressed locations following the location whose address is specified in the instruction.

If the blocks to be read are not all the same length, or---what is the same thing---if their lengths are not known to the program in advance--- a MAGNETIC-TAPE READ, RECORD (MRR) instruction is executed. This instruction causes the preface---which specifies how many words are contained in the block---associated with each block to be read and written into storage immediately preceding the first word of the associated block. Suppose, for example, that the instruction orders three blocks to be read, that the address specified in the instruction is 1001, and that the three blocks are, in order, 23, 37, and 17 words long. Figure IV - Intro - 2 shows the allocation of storage after completion of the instruction.

Note that the preface occupies the two high-order digit positions of the location in which it is stored.

Note also that Figure IV-Intro-2 shows the allocation of storage required for the execution of a MAGNETIC-TAPE INITIAL WRITE, RECORD instruction.

LOCATION	CONTENTS
1001	0 2300 00 0000 (preface of first block)
1002	Address word of first block
.	.
.	.
1024	23rd word of first block
1025	0 3700 00 0000 (preface of second block)
1026	Address word of second block
.	.
.	.
1062	37th word of second block
1063	0 1700 00 0000 (preface of third block)
1064	Address word of third block
.	.
.	.
1080	17th word of third block

Figure IV-Intro-2. Example of allocation of storage on execution of MRR.

It is possible for a reading instruction to be terminated without having read as many blocks as are specified in the instruction. This event may occur in case a control or end-of-tape block is encountered during the operation. The consequences of encountering a control block or end-of-tape block during a reading operation are specified in Table IV-Intro-5b and Table IV-Intro-5a, respectively.

Overwriting

The magnetic-tape system of the DATATRON 220 features the ability to modify information already recorded on tape. That is, the contents of any block may be altered, as required, without the need to copy the entire tape. This ability is provided by two instructions, MAGNETIC-TAPE OVERWRITE (MOW) and MAGNETIC-TAPE OVERWRITE, RECORD (MOR).

The execution of these instructions differs from the execution of the corresponding MIW and MIR instructions only in these essential respects:

1. It is not possible to execute MOW or MOR on newly-edited tape. If an attempt is made to overwrite on newly-edited tape, the tape will be moved to the physical end-of-tape marker, unless there is manual intervention.
2. Before overwriting begins, the preface written on tape is compared with the preface specified by the instruction. If the prefaces are identical, the block on tape will be overwritten; otherwise, a preface-mismatch ALARM STOP will occur, unless the mismatch occurs with an end-of-tape block (see Table IV-Intro-5a).

The contents of a control block or end-of-tape block may be modified using the MOW or MOR instructions.

Positioning

MAGNETIC-TAPE POSITION FORWARD (MPF) and MAGNETIC-TAPE POSITION BACKWARD (MPB) permit tape to be moved forward or backward, respectively, passing over from one to ten blocks. Once initiated, these

instructions are executed under direction of the Magnetic-Tape Control Unit, independently of the Computer.

MAGNETIC-TAPE POSITION AT END OF INFORMATION (MPE) moves tape, positioning it so that the read-write head is near the end of the last block written (that is, near the end of information) ready to (initial) write a next block.

MAGNETIC-TAPE REWIND (MRW) and MAGNETIC-TAPE REWIND, DEACTIVATE (MDA) allow tape to be rewound at the direction of the program. Once initiated, these instructions are executed under direction of the Magnetic-Tape Storage Unit or Datafile, independently of both the Computer and the Magnetic-Tape Control Unit. The MDA instruction differs from MRW in one essential respect: MDA causes interlocks to be set; these interlocks must be released manually by the system operator before the tape unit specified can be used by the program. If an instruction refers to a unit which is inoperative because these interlocks have not been released, a not-ready ALARM STOP will occur.

MAGNETIC-TAPE LANE SELECT (MLS) allows for the selection of a read-write head in a specified lane without any tape motion. Its principal purpose is in connection with the writing of control blocks so that they occupy, logically, the same relative position in both lanes of the same tape when control blocks are used to mark end-of-file or end-of-tape.

Interrogation

MAGNETIC-TAPE INTERROGATE, BRANCH (MIB) permits the program to determine if a designated tape unit is ready to be used. If the unit queried is ready, control is transferred.

Input Sign Control

MAGNETIC-TAPE READ and MAGNETIC-TAPE READ, RECORD may be coded to permit B-register address-modification of designated words read from magnetic tape.

B-register address modification can occur only if $(rC:41)/8=1$. The word read from tape is examined in the D register to see if B-register address modification is designated:

1. If $(rD:\dagger 1)/8=1$, B-register address modification will occur; at the same time, $(rD:\dagger 1)/8$ is set to zero.
2. If $(rD:\dagger 1)/8=0$, no modification of the word read from tape will occur.

There is one exception to the above: if B-register address modification of a control word (in a control or end-of-tape block) is intended, the sign digit of the control word should be odd. Then, when the control word is read, B-register address modification will occur no matter how the instruction is coded.

Parity Checking

As noted above, every digit is written on magnetic tape with an odd-parity bit. A parity error can be detected, however, only during the portion of an operation in which the read-write head is in read status. For example, the head is in read status during an entire searching operation.

Generally speaking, the detection of parity errors falls into two classes:

1. The detection of errors in the part of the block which is relevant to the instruction being executed. (For example, the block address when searching; the entire block when reading.) We call this a Class I error.
2. The detection of errors in a part of the block which is not relevant to the instruction being executed. (For example, all other words but the block address when searching.) We call this a Class II error.

In the DATATRON 220, only parity errors which fall into the first class are detected. The detection of Class II errors is ignored.

Table IV-Intro-6 defines the Class I errors.

In conclusion, two statements can be made about parity errors:

1. The detection of a parity error has priority over any other kind of error which might occur simultaneously. For example, suppose a parity error is detected in a preface during a MAGNETIC-TAPE OVERWRITE instruction: the parity error has priority - i.e., takes precedence in the demand for activity - over the preface mismatch.
2. The initial detection of a parity error results, automatically, in two attempts to repeat the operation during which the error is detected in order to eliminate the detected error (if it is due to a dust particle, say). For example, if the error is detected during MAGNETIC-TAPE READ, two additional attempts will be made to read the block. In case the parity error cannot be eliminated, a parity-error ALARM STOP will occur. Tape movement will

Operation	Location of error	Remarks
MAGNETIC-TAPE SEARCH MAGNETIC-TAPE FIELD SEARCH	1. Address 2. Preface of end-of-tape block	1. Correct address, but appears to be a. incorrect b. correct (if the parity bit is "dropped"). 2. Incorrect address, but appears to be a. correct b. incorrect. 3. End-of-tape block must be recognized.
MAGNETIC-TAPE SCAN MAGNETIC-TAPE FIELD SCAN	1. Category code. 2. Control block marker-digit in address of block. 3. Preface of end-of-tape block.	1. Correct category code, but appears to be a. incorrect b. correct (if the parity bit is "dropped"). 2. Incorrect category code, but appears to be a. correct b. incorrect. 3. Control block must be recognized. 4. End-of-tape block must be recognized.
MAGNETIC-TAPE READ MAGNETIC-TAPE READ, RECORD	1. Entire block. 2. Preface. 3. Control block marker-digit in address of block.	1. Control block must be recognized. 2. End-of-tape block must be recognized.
MAGNETIC-TAPE OVER-WRITE MAGNETIC-TAPE OVER-WRITE, RECORD	1. Preface.	1. Correct preface, but appears to be a. incorrect. b. correct (if parity bit is dropped) 2. Incorrect preface, but appears to be a. correct b. incorrect. 3. Parity has priority over preface mismatch.

Table IV - Intro - 6. Definition of Class I Parity Errors

stop with the head positioned to read the block in which the error was detected.

The block in question will have been read from tape and written in core storage.

Digit-count/Word-count Checking

Preceding every word written on tape is a timing character. During reading operations this character is used by the Magnetic-Tape Control Unit to verify that every word read is 11 digits long. That is, a "digit count" check is made.

During reading operations the preface associated with the block being read is used by the Magnetic-Tape Control Unit to verify that all of the data words in the block have been read. That is, a "word count" check is made.

If, during a reading operation, the digit count and/or the word count fail to check, two additional attempts will be made to read the block in question. These re-trials are made automatically under direction of the Magnetic-Tape Control Unit. If the digit count or the word count fails to check in both of these attempts to read the block, a digit-count/word-count ALARM STOP will occur, with the read-write head in position to read the block in which the error occurred.

The questionable block will have been read from tape and written in core storage.

Timing

Normally, when processing in a lane (specifically excluding the situation after execution of MAGNETIC-TAPE LANE SELECT), tape stops with the head positioned not more than five words (=60 digits) from and in front of the end-of-block mark. After a MAGNETIC-TAPE

READ instruction, for example, tape is repositioned so that the head is located, as just described, in the last block read, prepared to read the next block. Since there are 20 digit-positions in the erase gap and 39 digit-positions in the inter-block gap, it is required to pass over 119 digit-positions before the beginning of the next block is encountered.

DEFINITION: Acceleration time is the time required to elapse before the signal read from tape reaches maximum amplitude. Acceleration time is less than 3 milliseconds (0.003 seconds).

DEFINITION: Start time is the time required to get from rest to the next beginning-of-block mark, that is, to pass over 119 digit-positions, starting from rest.

In accelerating to the normal speed of 120 inches per second, approximately 75 digit-positions are passed. This is accomplished in 0.003 seconds. The remaining approximately 44 digit-positions which must be passed before the beginning-of-block mark is found require slightly less than 0.002 seconds to traverse. A conservative estimate of start time, then, is 0.005 seconds (5 milliseconds).

An operation which terminates with the tape in motion in the forward direction requires that tape be moved so as to place the read-write head near the end of the last block operated on. This tape movement is called "turn-around".

DEFINITION: Turn-around time is the time required

1. To stop the forward motion of the tape;
2. To reverse the direction of the motion of

tape; and

3. To stop the backward motion of tape with the head positioned in the block and not more than 60 digit-positions in front of the end-of-block mark.

A conservative estimate of turn-around time is 0.005 seconds (5 milliseconds).

Because it is possible to establish files in so many different formats, it is not possible to provide an exhaustive listing of the time required to perform any of the many operations which are possible. Sufficient information is provided in this section to permit any necessary time estimates to be made.

The tables which follow illustrate some representative samples of time estimates for execution of MAGNETIC-TAPE READ or MAGNETIC-TAPE WRITE.

Block size (words)	Start time (ms)	Reading or Writing time (ms)	Total time (ms)
10	5	6.4	11.4
20	5	11.2	16.2
30	5	16.0	21.0
40	5	20.8	25.8
50	5	25.6	30.6
60	5	30.4	35.4
70	5	35.2	40.2
80	5	40.0	45.0
90	5	44.8	49.8
100	5	49.6	54.6

Table IV-Intro-7. Time required to read or write one block (starting from rest).

THE MAGNETIC-TAPE SYSTEM

Block size (words)	Start time (ms)	Reading or Writing time (ms)	Total time (ms)
10	5	38.4	43.4
20	5	62.4	67.4
30	5	86.4	91.4
40	5	110.4	115.4
50	5	134.4	139.4
60	5	158.4	163.4
70	5	182.4	187.4
80	5	206.4	211.4
90	5	230.4	235.4
100	5	254.4	259.4

Table IV-Intro-8. Time required to read or write five blocks (starting from rest).

Block size (words)	Start time (ms)	Reading or Writing time (ms)	Total time (ms)
10	5	78.4	83.4
20	5	126.4	131.4
30	5	174.4	179.4
40	5	222.4	227.4
50	5	270.4	275.4
60	5	318.4	323.4
70	5	366.4	371.4
80	5	414.4	419.4
90	5	462.4	467.4
100	5	510.4	515.4

Table IV - Intro - 9. Time required to read or write ten blocks (starting from rest).

The reader will note that none of the tables include turn-around time, although that time may have to be given consideration when making timing estimates. Generally, however, turn-around time as well as repositioning time -- if the job requires overwriting the blocks read -- can be masked, in substantial part, by the time required to process the blocks read.

The entries in the preceding three tables were computed on the basis of the following considerations: the time, $T_{R,W}$ required to read or write n blocks, each k words long, starting with the tape at rest, is composed of the following elements:

1. Start time, t_s . $t_s = 5$ ms.
2. Time required to pass the space occupied by (the remainder of) the first block, t_1 . $t_1 = 0.04(41 + 12k)$ ms.
3. Time required to pass the space occupied by the remaining $n-1$ blocks, including the inter-block gap preceding each of the remaining $n-1$ blocks, t_f . $t_f = 0.04(n-1)(80 + 12k)$ ms.

Hence,

$$\begin{aligned} T_{R,W} &= t_s + t_1 + t_f \\ &= 5 + 0.04(41 + 12k) + 0.04(n-1)(80 + 12k) \end{aligned}$$

$$T_{R,W} = 3.44 + 3.2n + 0.48 nk \text{ ms.}$$

t_f provides a measure of the time required to read or write $n-1$ blocks, each k words long, when the tape is travelling at full speed. Hence, one may compute the time, t , required to read or write m blocks, each containing k words, with the tape moving at full speed, using the formula

$$t = m (3.2 + 0.48k) \text{ ms.}$$

The time, T_P , required to position backward (that is, to execute a MPB instruction) may be deduced on the basis of the following considerations: the time required to position backward n blocks, each k words long, starting with the tape at rest, is composed of the following elements:

1. Start time, t_s . $t_s = 5$ ms.

2. Time required to pass the space occupied by the remainder of the first block, not including the inter-block gap which precedes this block τ_1 . Recall that the head is positioned initially approximately 60 digit-positions from the end-of-block character.

$$\tau_1 = 0.04(80 + 12k - 119 - 60 - 20 - 39) \text{ ms.}$$

3. Time required to pass the space occupied by the remaining $n-1$ blocks, including the inter-block gap following each of the remaining $n-1$ blocks,

$$\tau_f. \tau_f = 0.04(n-1)(80 + 12k) \text{ ms.}$$

4. Stop time. This is the same as start time.

Hence,

$$\begin{aligned} T_p &= t_s + \tau_1 + \tau_f + t_s \\ &= 5 + 0.04(12k - 158) + 0.04(n-1)(80 + 12k) + 5 \\ T_p &= 3.2 n + 0.48 kn + 0.5 \text{ ms.} \end{aligned}$$

Information Flow

The general nature of information flow during input and output was discussed in Section II. In some details, however, the flow of information during execution of magnetic-tape operations is different from that indicated in Section II. For this reason, the flow of information is described again below.

Input flow

Input flow is shown in Figure IV-Intro-7.

1. (rD:00) -- the D register contains an image of the instruction brought from storage during the Fetch Phase -- are transferred to the T register in the Magnetic-Tape Control Unit.

2. The address part of the C register -- which specifies the location in storage where the first word read from tape will be stored -- is copied into the four high-order digit-positions of the C register, that is, (rC:04) → rC:44. This address is preserved in case a parity-error is detected or the digit-count/word-count check fails and an attempt is made -- under machine control -- to re-read the block in which the error is detected: the Computer must have the address of the location into which is to be written the first word of the block read from tape.

3. The address part of the C register -- which specifies the location in storage where the next word read from magnetic tape is to be stored -- is transferred to the E register. At the same time, the address part of the C register is counted up 1.

4. The next word on tape is read and transferred to the D register.

5. The contents of the D register are then transferred to the IB register, with or without B-register modification as specified and indicated.

6. The contents of the IB register are transferred to storage.

If all of the words of the block have not been read, there is a return to step 3 for the next word.

If all of the blocks, as specified by the instruction, have not been read, there is a return to step 2 to prepare for reading the next block.

Such modifications in the flow as are required when a control or end-of-tape block is encountered will be explained in the description of the Execute Phase of the appropriate instruction.

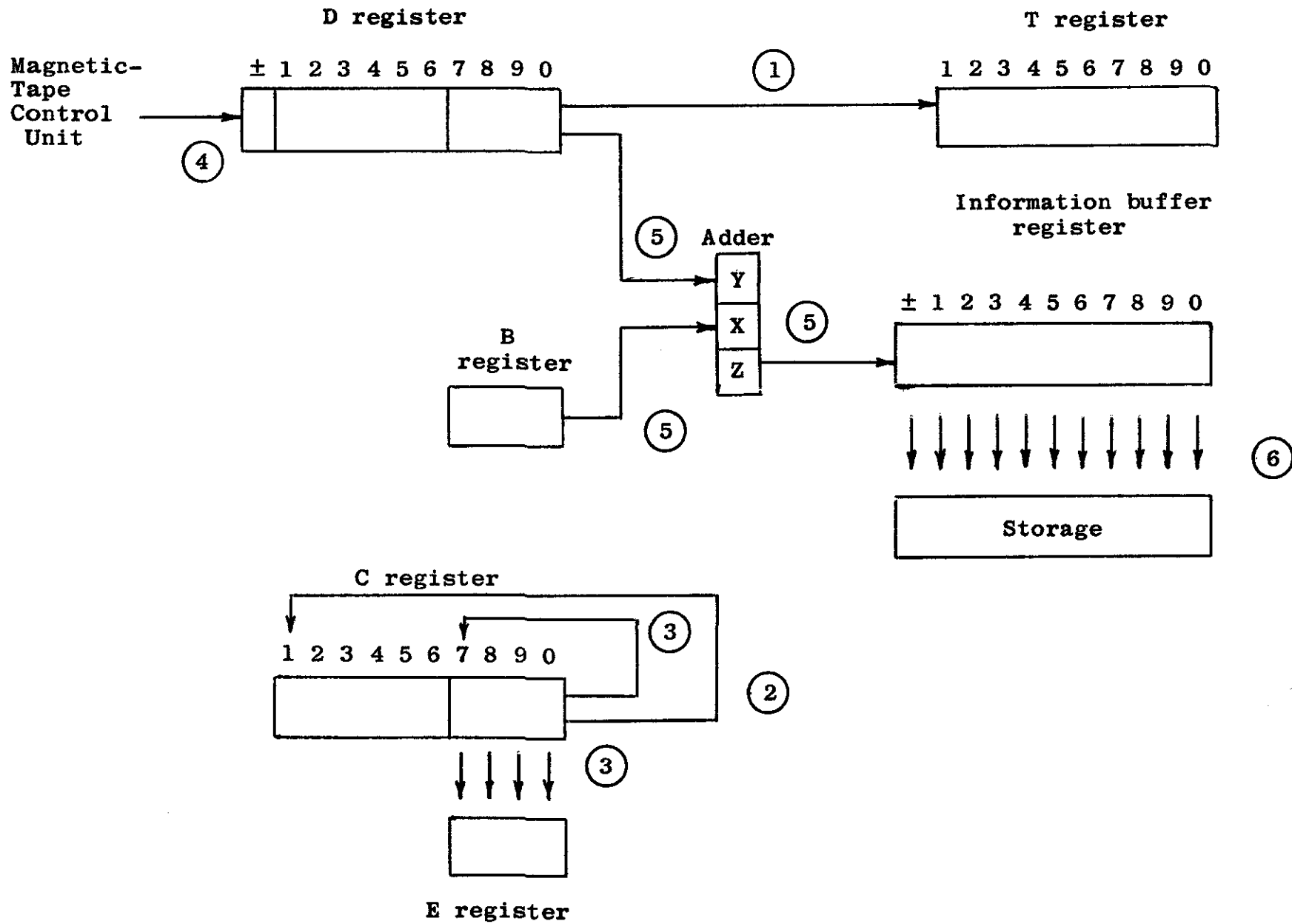


Figure IV-Intro-7. Input flow

Output flow

Output flow is shown in Figure IV - Intro - 8.

1. (rD:00) -- the D register contains an image of the instruction brought from storage -- are transferred to the T register.

2. The address part of the C register -- which specifies the location in storage from which will be taken the first word to be written on tape -- is copied into the four high-order digit positions of the C register, that is, (rC:04) → rC:44.

3. The address part of the C register -- which specifies the location in storage from which will be taken the next word to be written on tape -- is transferred to the E register. At the same time, the address part of the C register is counted up 1.

4. The word is transferred from storage to the IB register.

5. The contents of the IB register pass through the adder to the Magnetic-Tape Control Unit and thence to the designated tape unit, with the low-order digit of the word written first.

If all of the words of the block have not been written, there is a return to step 3 for the next word.

If all of the blocks, as specified by the instruction, have not been written, there is a return to step 2 to prepare for writing the next block.

Use of DATATRON 205 Magnetic Tape

It is possible to read from magnetic tape which has been prepared by a DATATRON 205 system.

The reader may recall that a word is written on DATATRON 205 magnetic tape with sign digit first, low-order digit last. DATATRON 220 tape, on the other hand, is written with low-order digit first, sign digit last.

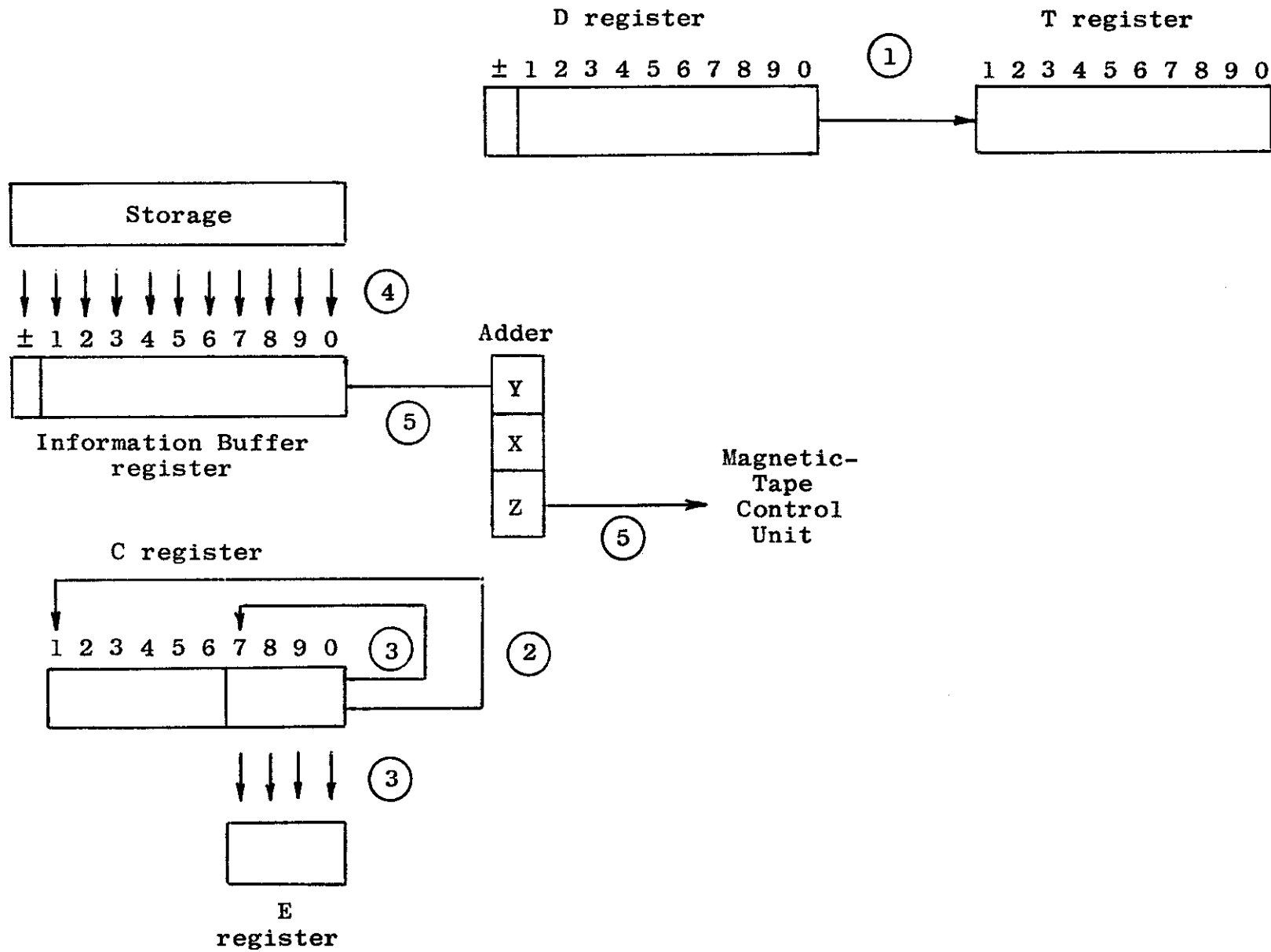


Figure IV-Intro-8. Output flow

It is necessary, therefore, to invert the order of the digits in a word which may be written on DATATRON 205 tape before the word can be processed in the DATATRON 220. This may be accomplished in the DATATRON 205, before the word is written, or it may be done in the DATATRON 220, after the word has been read. A procedure to accomplish this inversion in the DATATRON 220 can be found in the Handbook of Programming and Coding Techniques.

Exceptional Conditions

Following is a list summarizing conditions which can occur and are detected by the DATATRON 220. In addition to those enumerated below, a non-existent-address ALARM STOP and field-overflow ALARM STOP -- both of which were defined in Section II -- can occur.

1. A parity-error ALARM STOP will occur whenever a failure in parity is detected, and attempts to eliminate the error have not been successful. Parity errors are defined by Table-IV-Intro-6

2. A digit-count/word count ALARM STOP will occur during a reading operation if either the digit count (per word) or the word count (per block) fails to check.

3. An improper-block-length ALARM STOP will occur if an instruction specifies the writing of a block whose length is 2, 3, 4, 5, 6, 7, 8, or 9 words.

4. MOW and MOR begin by comparing the preface already on tape with the preface specified by the instruction. If the two prefaces are not identical, and the block on tape is not an end-of-tape block, a preface-mismatch ALARM STOP will occur.

5. If an attempt is made to write on a unit which is in not-write status, a not-write-status ALARM STOP will occur.

6. Suppose a tape is positioned at physical end-of-tape after execution of a MIW or MIR instruction. If the next instruction referring to that unit is one which initially moves tape in the forward direction, an end-of-tape ALARM STOP will occur.

If physical end-of-tape is sensed during the execution of MTS, MFS, MTC, MFC, or MPF instructions, an "operation complete" signal will not be generated. Manual intervention is required before processing can proceed.

If physical beginning-of-tape is sensed during the execution of MTS, MFS, or MPB instructions, an "operation complete" signal will not be generated. Manual intervention is required before processing can proceed.

7. If the unit referred to by an instruction is inoperative, a not-ready ALARM STOP will occur.

Execute Phase

The remainder of Section IV is devoted to a description of the Execute Phase of all the instructions which refer to magnetic tape.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 50

Operation name:

MAGNETIC-TAPE SEARCH
 MAGNETIC-TAPE FIELD SEARCH
 MAGNETIC-TAPE LANE SELECT
 MAGNETIC-TAPE REWIND
 MAGNETIC-TAPE REWIND, DE-ACTIVATE

Abbreviation:

MTS
 MFS
 MLS
 MRW
 MDA

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	h	h	v	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Computer time (μs):

MTS, MFS:
 fetch: 90
 execute: 160
 total: 250

Definitions:

MRW, MDA:
 fetch: 90
 execute: 110
 total: 200

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

± = 4 or 5: if v = 0, MAGNETIC-TAPE FIELD SEARCH will be executed.

MLS:
 fetch: 90
 execute:
 total:

u: designates tape unit.

hh: if u is a tape storage unit, lane 0 or 1 is selected according as hh is an even or odd number.

if u is a Datafile, lane hh is selected.

v: variation designator.

v = 0: ± = 0 or 1: MAGNETIC-TAPE SEARCH will be executed.

± = 4 or 5: MAGNETIC-TAPE FIELD SEARCH will be executed.

v = 4: MAGNETIC-TAPE LANE SELECT will be executed.

THE MAGNETIC-TAPE SYSTEM

v = 8: MAGNETIC-TAPE REWIND
will be executed.

v = 9: MAGNETIC-TAPE REWIND, DE-ACTIVATE
will be executed.

Op: Operation code

aaaa: v=0: address of base of location of
search key.

v≠0: aaaa is not relevant to the
execution of these instructions.

Description of operation:

Summary:

v = 0, ± ≠ 4 or 5: MAGNETIC-TAPE SEARCH will be executed.

Search on unit u, in the lane specified by hh, for the block whose address is identical with (B[aaaa]:00), the search key. When the block sought has been found, the operation will terminate with the read-write head positioned to read the desired block.

After MAGNETIC-TAPE SEARCH is initiated, it is executed under control of the Magnetic-Tape Control Unit, independently of the Computer.

v=0, ± = 4 or 5: MAGNETIC-TAPE FIELD SEARCH will be
executed.

Search on unit u, in the lane specified by hh, for the block, the corresponding part of whose address is identical with (B[aaaa]:sL), the partial-word search key. The partial-word boundaries are defined by (rB:82)=sL. When the block sought has been found, the operation will terminate with the read-write head positioned to read the desired block.

After MAGNETIC-TAPE FIELD SEARCH is initiated, it is executed under control of the Magnetic-Tape Control Unit,

independently of the Computer.

v = 4: MAGNETIC-TAPE LANE SELECT will be executed.

On unit u, select the read-write head in the lane specified by hh. There is no tape movement.

v = 9: MAGNETIC-TAPE REWIND, DE-ACTIVATE will be executed.

Rewind unit u to the physical beginning-of-tape marker. At completion of the rewind, select the read-write head in the lane specified by hh. Then, de-activate the unit.

After MAGNETIC-TAPE REWIND, DE-ACTIVATE is initiated, it is executed under control of the magnetic-tape unit, independently of both the Computer and Magnetic-Tape Control Unit.

v = 8: MAGNETIC-TAPE REWIND will be executed.

Except that the unit is not de-activated, this variation is the same as that executed when v = 9.

Flow chart:

See page IV-50-9

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP. The test for field overflow is made before initiating the searching operation: if field overflow is detected, there will be no tape movement.
3. Parity-error ALARM STOP.
4. End-of-tape ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. The MAGNETIC-TAPE SEARCH or MAGNETIC-TAPE FIELD SEARCH variation will be selected for execution if $v = 0, 1, 2,$ or 3 . Which of MTS or MFS will be selected is determined by the sign digit of the instruction.
2. The MAGNETIC-TAPE LANE SELECT variation will be selected for execution if $v = 4, 5, 6,$ or 7 .
3. The nature of the searching operation is defined as follows: Tape movement starts in the forward direction and continues in that direction until one of two events occurs:
 - a. A block is encountered whose address is greater than or equal to the search key; or
 - b. An end-of-tape block is encountered. When an end-of-tape block is encountered, the operation is terminated with the head in position to read the end-of-tape block.

When (a) occurs, the direction of tape motion is reversed. Tape continues to move in the backward direction until one of two events occurs:

- c. A block is encountered whose address is equal to the search key. In this case, the operation terminates with the head positioned to read the block whose address is identical with the search key.
- d. A block is encountered whose address is less than the search key.

When (d) occurs, the direction of tape motion is again reversed. Tape moves in the forward direction until the read-write head is near the end of the block whose address is less than the search key: tape movement stops with the head in position to read the block following the one whose address is less than the search key. That is, tape movement ceases with the head in position to read the first block whose address is greater than the search key.

4. As a result of the definition of the searching operation, it can happen that an instruction to search will fail to find the sought-for block. It is necessary, therefore, to verify under program control that an instruction to search has in fact found the desired block.

5. As a result of the definition of the searching operation, an order structure is imposed on a file - or that part of it which is in one lane - if it is one in which searching is to be done: the file must be set up so that if the block whose address is k follows the block whose address is j , then k must be greater than j .

6. If the address of every block on a tape is greater than the search key, the reflective strip which marks the physical beginning of tape will be encountered during the backward pass after the first block has been passed. Tape movement - and, hence, searching - stops when the reflective strip is sensed. No "operation complete" signal

is generated, however; so, the Magnetic-Tape Control Unit will appear to be busy if a MIB instruction is issued or if an attempt is made to use tape (by attempting to execute a MRD instruction, for example). Manual intervention is required in order to proceed with processing.

In a properly-established file this event will not occur.

7. If the address of every block on tape is less than the search key, and if the tape is not bounded by an end-of-tape block, the reflective strip which marks the physical end of tape will be encountered during the forward pass after the last block has been passed. Tape movement - and, hence, searching - stops when the reflective strip is sensed. No "operation complete" signal is generated, however; so, the Magnetic-Tape Control Unit will appear to be busy if a MIB instruction is issued or if an attempt is made to use tape (by attempting to execute a MRD instruction, for example). Manual intervention is required in order to proceed with processing.

In a properly-established file this event will not occur.

8. If the Magnetic-Tape Control Unit is directing a searching operation at the time another magnetic-tape instruction is issued - with the exception of MIB and MIE instructions - the Computer will wait for an "operation complete" signal from the Magnetic-Tape Control Unit before attempting to complete the execution of the instruction.

9. If a tape unit is rewinding - executing a MRW or MDA instruction - at the time another magnetic-tape instruction is issued - with the exception of MIB and MIE instructions - the Computer will wait for an "operation complete" signal from the referenced tape unit before attempting to complete the execution of

the instruction. If the rewinding instruction is MDA, manual intervention is required before continuing.

10. Blank tape - that is, tape on which no information is recorded - is treated like inter-block gap.

11. The sign digit cannot be included in a partial-word scan key. If it is, a field-overflow ALARM STOP will occur.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 51

Operation name:

MAGNETIC-TAPE SCAN

MAGNETIC-TAPE FIELD SCAN

Abbreviation:

MTC

MFC

Computer time (μ s):

fetch: 90
execute: 160
total: 250

Instruction format:

\pm 1 2 3 4 5 6 7 8 9 0

\pm	u	h	h	k	Op	a	a	a	a
-------	---	---	---	---	----	---	---	---	---

Definitions:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

$\pm = 4$ or 5 : MAGNETIC-TAPE FIELD SCAN will be executed.

u: designates magnetic-tape unit.

hh: if u is a tape storage unit, lane 0 or 1 is selected according as hh is an even or odd number;

if u is a Datafile, lane hh is selected.

k: specifies in which word of a block the category code will be found. If $k = 0$, the tenth word of the block contains the category code.

Op: operation code.

aaaa: address of base of location of scan key.

Description of operation:

Summary:

$\pm \neq 4$ or 5: MAGNETIC-TAPE SCAN will be executed.

Scan on unit u , in the lane specified by hh , for the block whose category code in the k -th word is identical with $(B[aaaa]:00)$, the scan key. When the block sought has been found, the operation will terminate with the head positioned to read the sought-for block.

After MAGNETIC-TAPE SCAN is initiated, it is executed under control of the Magnetic-Tape Control Unit, independently of the Computer.

$\pm = 4$ or 5: MAGNETIC-TAPE FIELD SCAN will be executed.

Scan on unit u , in the lane specified by hh , for the block the corresponding part of whose category code in the k -th word is identical with $(B[aaaa]:sL)$, the scan key. The partial-word boundaries are defined by $(rB:82) = sL$. When the block sought has been found, the operation will terminate with the head positioned to read the sought-for block.

After MAGNETIC-TAPE FIELD SCAN is initiated, it is executed under control of the Magnetic-Tape Control Unit, independently of the Computer.

Flow chart:

See page IV-51-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Field-overflow ALARM STOP. The test for field overflow is made before initiating the searching operation: if field overflow is detected, there will be no tape movement.
3. Parity-error ALARM STOP.
4. End-of-tape ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. The nature of the scanning operation - which is different from MTS or MFS (which see, page IV-50-5) - is defined as follows: tape movement starts in the forward direction and continues in that direction until one of two events occurs:

- a. A block is encountered whose category code is identical with the scan key; or
- b. A control block or end-of-tape block is encountered.

In either case, the operation terminates with the head positioned to read the block which caused the operation to terminate.

2. Note, therefore, that both MTC and MFC can terminate without having found a sought-for block. It is necessary, therefore, to verify under program control that a sought-for block has been found.

3. These two operations do not require an ordered file.

4. If there is no block on tape whose category code is identical with the scan key, and if the tape is not bounded by a control block or end-of-tape block, the reflective strip which marks the physical end-of-tape will be encountered after the last block has been passed. Tape movement - and, hence, scanning - stops when the reflective strip

THE MAGNETIC-TAPE SYSTEM

is sensed. No "operation complete" signal is generated, however; so the Magnetic-Tape Control Unit will appear to be busy if a MIB instruction is issued or if an attempt is made to use tape (by attempting to execute a MRD instruction, for example). Manual intervention is required in order to proceed with processing.

In a properly-established file, this event will not occur.

5. If the Magnetic-Tape Control Unit is directing a scanning operation at the time another magnetic-tape instruction is issued - with the exception of MIB and MIE instructions - the Computer will wait for an "operation complete" signal from the Magnetic-Tape Control Unit before attempting to complete the execution of the instruction.

6. Blank tape - that is, tape on which no information is recorded - is treated like inter-block gap.

7. The sign digit cannot be included in a partial-word scan key. If it is, a field-overflow ALARM STOP will occur.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 52

Abbreviation:

MRD

Time (μ s):

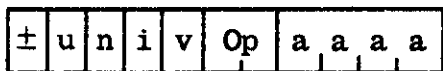
fetch: 90
execute: see discussions of timing, pages IV-Intro-28, ff.

Operation name:

MAGNETIC-TAPE READ

Instruction format:

\pm 1 2 3 4 5 6 7 8 9 0



Definitions:

- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

- u: designates magnetic-tape unit.

- n: specifies the number of blocks to be read. n = 0 means ten blocks.

- i: not relevant to the execution of this instruction.

- v: variation designator
(rC:41)/8 = 1: B-register address-modification of designated input will occur.

- Op: operation code.

- aaaa: address of base of location in which is written the first word read from magnetic tape.

Description of operation:

Summary:

Read n blocks from unit u . The lane read from is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words read from tape are written into consecutively-addressed storage locations beginning with $B[aaaa]$.

Words read from tape are assembled in the D register before being sent to storage: if $(rC:41)/8 = 1$ and $(rD:\pm 1)/8 = 1$, the word will be B-register address-modified as it goes from the D register to storage; at the same time, $(rD:\pm 1)/8$ is set to 0.

A control block is recognized as such: if a control block is encountered during the execution of MRD, it will be sensed, and the following events will occur:

1. Each word of the control block, except the last, will be read and written in the storage location designated.
2. The last word of the block read from tape - the control word - remains in the Computer's control section to provide information with which to terminate the operation. The control word has the following format:

$\pm ii\ aaaa\ bbbb$

- a. The contents of the P register are preserved: $(rP) \rightarrow aaaa:04$. This enables the program to determine the location of the instruction during whose execution the control block was encountered.

- b. The contents of the address part of the C register are preserved: (rC:04) → aaaa:64. This enables the program to determine the location of the last word read from magnetic tape.
- c. The address of the location of the next instruction selected for execution - an entry to the control routine - is transferred to the P register: B[bbbb] → rP.

After reading the control block, MRD is terminated.

An end-of-tape block is recognized as such: if an end-of-tape block is encountered during the execution of MRD, it will be sensed, and the following events will occur:

The control word is read from tape but remains in the Computer's control section to provide information with which to terminate the operation. The control word has the same format as the control word in a control block. The transfer of the remainder of the end-of-tape block from tape to core storage is inhibited. The events described under 2(a), 2 (b), and 2(c), above, occur (with obvious changes in language: "end-of-tape" is substituted for "control").

After reading the end-of-tape block, MRD is terminated.

After termination of MRD, tape is positioned so that the head can read the block following the one which caused the operation to terminate.

Flow chart:

See page IV-52-7.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Parity-error ALARM STOP.
3. End-of-tape ALARM STOP.
4. Digit-count/word-count ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. A description of block layout can be found in the Introduction to this section; see also the description of MAGNETIC-TAPE INITIAL WRITE - pp. IV-54-2, ff. - where the assignment of storage locations is noted, as well.
2. This instruction does not read the preface word associated with each block. If the preface must be read, a MAGNETIC-TAPE READ, RECORD instruction - see pp. IV-53-2, ff. - must be executed.
3. Note that a MRD instruction may terminate without having read n blocks as specified.
4. Blank tape - that is, tape on which no information is recorded - is treated like inter-block gap.
5. If an end-of-tape block is also identified as a control block - by having a 7 in the sign-digit position of the control word - the block will invariably be recognized as end-of-tape. When the end-of-tape block is read, however, B-register address-modification of the control word will occur because its sign digit is odd.
6. During the execution of MRD, the timing character preceding each word written on tape is used to verify that 11 digits are read for each word; the preface and the end-of-block character

are used to verify that the correct number of words has been read. Should there be a failure in either the digit count or the word count during MRD, two additional attempts to eliminate the failure will be made, automatically, under the direction of the Magnetic-Tape Control Unit. If neither of the additional attempts succeeds, a digit-count/word-count ALARM STOP will occur with the tape positioned so that the read-write head can read the block in which the failure occurred.

7. After a parity-error ALARM STOP, the read-write head is in position to read the block in which the error was detected. The contents of the block will have been read from tape and written in storage, however.

Operation code: 53

Operation name:

MAGNETIC-TAPE READ, RECORD

Abbreviation:

MRR

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	n	i	v	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Time (μs):

fetch: 90
 execute: See discussion of timing, pages IV-Intro-28, ff.

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise there will be no such modification.

u: designates magnetic-tape unit.

n: specifies the number of blocks to be read. n = 0 means 10 blocks.

i: not relevant to the execution of this instruction.

v: variation designator.

(rC:41)/8 = 1: B-register address-modification of designated input will occur.

Op: operation code.

aaaa: address of base of location in which will be written the preface associated with the first block read from tape.

Description of operation:

Summary:

Read n blocks from unit u . The lane read from is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words read from tape are written into consecutively-addressed storage locations, the preface associated with the first block going into $B[aaaa]$.

Words read from tape are assembled in the D register before being sent to storage: if $(rC:41)/8 = 1$, and $(rD:\pm 1)/8 = 1$, the word will be B-register address-modified as it goes from the D register to storage; at the same time, $(rD:\pm 1)/8$ is set to 0.

A control block is recognized as such: if a control block is encountered during the execution of MRR, it will be sensed, and the following events will occur:

1. The preface and each word of the control block, except the last, will be read and written in the storage location designated.
2. The last word of the control block read from tape - the control word - remains in the Computer's control section to provide information with which to terminate the operation. The control word has the following format: $\pm ii\ aaaa\ bbbb$
 - a. The contents of the P register are preserved: $(rP) \rightarrow aaaa:04$. This enables the program to determine the location of the instruction during whose execution the control block was encountered.
 - b. The contents of the address part of the C register are preserved: $(rC:04) \rightarrow aaaa:64$. This enables

the program to determine the location of the last word read from magnetic tape.

- c. The address of the location of the next instruction selected for execution - an entry to the control routine - is transferred to the P register:
B[bbbb] → rP.

After reading the control block MRR is terminated.

An end-of-tape block is recognized as such: if an end-of-tape block is encountered during the execution of MRR, it will be sensed, and the following events will occur:

1. The preface is read from tape and written in core storage in the location designated.
2. The control word is read from tape but remains in the Computer's control section to provide information with which to terminate the operation. This control word has the same format as the control word in a control block. The transfer of the remainder of the end-of-tape block from tape to core storage is inhibited. The events described under 2 (a), 2 (b), and 2 (c), above, occur (with obvious changes in language: "end-of-tape" is substituted for "control").

After reading the end-of-tape block, MRR is terminated.

After termination of MRR, tape is positioned so that the head can read the block following the one which caused the operation to terminate.

Flow chart:

See page IV-53-7.

THE MAGNETIC-TAPE SYSTEM

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Parity-error ALARM STOP.
3. End-of-tape ALARM STOP.
4. Digit-count/word-count ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. During MRR, the preface associated with each block is read from tape and written into core storage. Otherwise, MRR and MRD are executed in identical fashion.

2. Suppose that a preface is kk and the preface is written in the location whose address is L . Then the first data word of the block will be in $L + 1$ and the last data word in $L + kk$.

For example, suppose MRR says, "read three blocks; preface of first block to be written in 0984." Suppose that the first two blocks are 25 and 30 words long, respectively; the third block is a control block and is 40 words long. The allocation of storage is as shown:

Location	Contents	Remarks
0984	0 2500 00 0000	} Preface of 1st block
0985	1st word	
.	.	} first block
.	.	
.	.	
1009	25th word	} Preface of 2nd block
1010	0 3000 00 0000	
1011	1st word	} second block
.	.	
.	.	
1040	30th word	} Preface and control-block marker
1041	7 4000 00 0000	
1042	1st word	} Control block
.	.	
.	.	
1080	39th word	

3. Note that a MRR instruction may terminate without having read n blocks as specified.

4. Blank tape - that is, tape on which no information is recorded - is treated as if it were inter-block gap.

5. If an end-of-tape block is also identified as a control block - by having a 7 in the sign-digit position of the control word - the block will invariably be recognized as end-of-tape. When the end-of-tape block is read, however, B-register address-modification of the control word will occur because its sign digit is odd.

6. During the execution of MRR, the timing character preceding each word written on tape is used to verify that 11 digits are read for each word; the preface and the end-of-block character are used to verify that the correct number of words has been read. Should there be a failure in either the digit count, or the word count during MRR, two additional attempts to eliminate the failure will be made, automatically, under the direction of the Magnetic-Tape Control Unit. If neither of the additional attempts succeeds, a digit-count/word-count ALARM STOP will occur with the tape positioned so that the read-write head can read the block in which the failure occurred.

7. After a parity-error ALARM STOP, the read-write head is in position to read the block in which the error was detected. The contents of the block will have been read from tape and written in storage, however.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 54

Operation name:

MAGNETIC-TAPE INITIAL WRITE

Abbreviation:

MIW

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	n	k	k	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Time (μs):

fetch: 90

execute: See discussion
of timing,
pp. IV-Intro-
28, ff.

Definitions:

±: if ± is odd, B-register address-
modification will occur; otherwise,
there will be no such modification.

u: designates magnetic-tape unit.

n: specifies number of blocks to be
written. n = 0 means ten blocks.

kk: specifies number of words in each
of the n blocks. kk = 00 means
100 words.

Op: operation code.

aaaa: address of base of location from
which is read the first word of
the first block to be written.

Description of operation:

Summary:

Write - on newly-edited tape - on unit u; write n blocks; each block is kk words long: the minimum-length block is ten words; the maximum-length block is 100 words. The lane written on is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words written on tape are taken from consecutively-addressed locations beginning with B[aaaa].

This instruction causes the preface to be written immediately preceding the first word of the block with which the preface is associated.

If magnetic end-of-tape is encountered during the execution of MIW, the magnetic End-of-Tape Indicator will be set "on" at the completion of the operation.

Flow chart:

See page IV-54-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Improper-block-length ALARM STOP.
3. Not-write-status ALARM STOP.
4. End-of-tape ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. MIW recognizes magnetic end-of-tape. Therefore, every MIW instruction must be followed by a MIE instruction to sense the status

of the Magnetic End-of-Tape Indicator. The MIE instruction provides the branch to the end-of-tape control procedure (see page IV-59-2, ff.).

2. After termination of a MIW instruction during which magnetic end-of-tape was encountered, tape is positioned at physical end. This fact is sensed by the MIE instruction which follows every MIW instruction. After execution of the MIE instruction, if the next instruction referring to the tape unit is one which causes tape movement, it must be one which causes tape to move initially in the backward direction; otherwise an ALARM STOP will occur. (Admissible instructions are MRW, MDA, MPB.)

Normally, after encountering magnetic end-of-tape, terminal control blocks must be written to bound the file just written. Procedures for accomplishing this are described in The Handbook of Programming and Coding Techniques.

3. MIW is one of two magnetic-tape operations which recognize magnetic end-of-tape. The other operation is MAGNETIC-TAPE INITIAL WRITE, RECORD.

4. It is necessary to verify, under program control, that newly-edited tape is being used (see discussion of magnetic-tape labelling procedures in Handbook of Programming and Coding Techniques). If unedited tape is used with MIW - or MIR - there is no guarantee that the tape will be readable, or if it is, that the information read will be meaningful.

5. If the unit referred to by this instruction is in not-write status - by virtue of lacking a write ring, for example - an ALARM STOP will occur.

THE MAGNETIC-TAPE SYSTEM

6. To illustrate the allocation of storage, suppose a MIW instruction specifies that three blocks are to be written, each 18 words long; the first word of the first block is in location 5001.

Location	Contents	Remarks
5001	1st word	} First block
.	.	
5018	18th word	
5019	1st word	} Second block
.	.	
5036	18th word	
5037	1st word	} Third block
.	.	
5054	18th word	

7. The format of a block as it appears on tape is shown on page IV-Intro-6.

Operation code: 55

Operation name:

MAGNETIC-TAPE INITIAL WRITE,
RECORD

Abbreviation:

MIR

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	n	i	i	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Time (μs):

fetch: 90
execute: See discussion
of timing pages
IV-Intro-28, ff.

Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates magnetic-tape unit.
- n: specifies the number of blocks to be written. n = 0 means ten blocks.
- ii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location from which is read the first word to be written on tape. This word contains the preface.

Description of operation:

Summary:

Write -- on newly-edited tape -- on unit u; write n blocks. The lane written on is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words written on tape are taken from consecutively-addressed locations beginning with B[aaaa].

The first word written is a preface word: (B[aaaa]:22) are interpreted as the preface. This preface specifies how many words in the consecutively-addressed locations following B[aaaa] are in the first block. If more than one block is to be written, the first word following the last word of the previous block written will be interpreted as containing the preface of the next block to be written.

The minimum-length block is ten words; the maximum, 100 words.

If magnetic end-of-tape is encountered during the execution of MIR, the magnetic End-of-Tape Indicator will be set "on." The instruction will be completed, however.

Flow chart:

See page IV-55-6.

Exceptional conditions:

1. Non-existent-address ALARM STOP
2. Improper-block-length ALARM STOP
3. Not-write-status ALARM STOP
4. End-of-tape ALARM STOP
5. Not-ready ALARM STOP

Remarks:

1. MIR recognizes magnetic end-of-tape. Therefore, every MIR instruction must be followed by a MIE instruction to sense the status of the magnetic End-of-Tape Indicator. The MIE instruction provides the branch to the end-of-tape control procedure (see pages IV-59-2, ff.).

2. After termination of a MIR instruction during which magnetic end-of-tape was encountered, tape is positioned at physical end. This fact is sensed by the MIE instruction which follows every MIR instruction. After execution of the MIE instruction, if the next instruction referring to the tape unit is one which causes tape movement, it must be one which causes tape to move initially in the backward direction; otherwise, an ALARM STOP will occur. (Admissible instructions are MRW, MDA, MPB.)

Normally, after encountering magnetic end-of-tape, terminal control blocks must be written to bound the file just written. Procedures for accomplishing this are described in the Handbook of Programming and Coding Techniques.

3. Instruction formats for MIW and MIR are very much alike: the formats differ in digit positions 3 and 4 which are used to specify the preface in MIW and are noted as "not relevant" for MIR. However, even in a MIR instruction, digit positions 3 and 4 must not contain 02, 03, 04, 05, 06, 07, 08, or 09: if one of these configurations does appear, an improper-block-length ALARM STOP will occur, with the read-write head positioned at its starting position.

4. MIR is one of two magnetic-tape operations which recognize the magnetic end-of-tape. The other operation is MAGNETIC-TAPE INITIAL WRITE.

THE MAGNETIC-TAPE SYSTEM

5. It is necessary to verify, under program control, that newly-edited tape is being used (see discussion of magnetic-tape labelling procedures in Handbook of Programming and Coding Techniques). If unedited tape is used with MIR -- or MIW -- there is no guarantee that the tape will be readable, or if it is, that the information read will be meaningful.

6. If the unit referred to by this instruction is in not-write status -- by virtue of lacking a write ring, for example -- an ALARM STOP will occur.

7. To illustrate the allocation of storage, suppose a MIR instruction specifies that three blocks are to be written, 15, 27 and 12 words long, respectively, in that order; the preface associated with the first block is in location 5001.

Location	Contents	Remarks
5001	0 1500 00 0000	Preface of first block
5002	1st word	} First block
⋮	⋮	
5016	15th word	
5017	0 2700 00 0000	Preface of second block
5018	1st word	} Second block
⋮	⋮	
5044	27th word	
5045	0 1200 00 0000	Preface of third block
5046	1st word	} Third block
⋮	⋮	
5057	12th word	

8. The format of a block as it appears on tape is shown on page IV-Intro-6

Operation code: 56

Operation name:

Abbreviation:

MAGNETIC-TAPE OVERWRITE

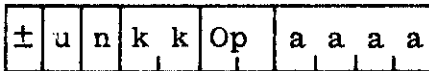
MOW

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

fetch: 90



execute: See discussion of timing, pages IV-Intro-28 ff.

Definitions:

- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates magnetic-tape unit.
- n: specifies the number of blocks to be written. n = 0 means ten blocks.
- kk: specifies number of words in each of the n blocks. kk = 00 means 100 words.
- Op: operation code
- aaaa: address of base of location from which is read the first word of the first block to be written.

Description of operation:

Summary:

Write n blocks on unit u ; each block contains kk words; the lane written on is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words written on tape are taken from consecutively-addressed locations beginning with $B[aaaa]$.

Before each block is overwritten, its preface (already on tape) is compared with kk : the block is written if and only if the preface and kk are identical; otherwise - except when the block on tape is an end-of-tape block - a preface-mismatch ALARM STOP occurs, with the read-write head positioned to read the block causing the ALARM STOP.

If a preface mismatch occurs with an end-of-tape block on tape, the ALARM STOP will not occur. Instead, the end-of-tape block is sensed, the control word is read from tape into the Computer control section, and terminating procedures are begun:¹

1. The contents of the P register are preserved: $(rP) \rightarrow aaaa:04$. This enables the program to determine the location of the instruction during whose execution the end-of-tape block was encountered.
2. The contents of the address part of the C register are preserved: $(rC:04) \rightarrow aaaa:64$. This enables the program to determine the location in core storage of the last word written on tape.

¹Recall that the format of the control word is $\pm ii aaaa bbbb$.

3. The address of the location of the next instruction selected for execution - an entry to the end-of-tape routine - is transferred to the P register: B[bbbb] → rP.

After reading the end-of-tape block, MOW is terminated.

After termination of MOW, tape is positioned so that the head can write the block following the one which caused termination.

Flow Chart

See page IV-56-5

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Preface-mismatch ALARM STOP.
3. Parity-error ALARM STOP.
4. Not-write-status ALARM STOP.
5. End-of-tape ALARM STOP.
6. Not-ready ALARM STOP.

Remarks:

1. If the unit referred to by this instruction is in not-write status - by virtue of lacking a write ring, for example - an ALARM STOP will occur.

2. The allocation of storage is as depicted for MIW.

3. Control blocks are not recognized as such during the execution of a MOW instruction.

4. Blank tape - that is, tape on which no information is recorded - is treated like inter-block gap.

5. Note that a MOW instruction may terminate without having written n blocks as specified.

Operation code: 57

Operation name:

MAGNETIC-TAPE OVERWRITE,
RECORD

Abbreviation:

MOR

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	n	i	i	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Time (μs):

fetch: 90
 execute: See discussion
 of timing,
 pages
 IV-Intro-28, ff.

Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates magnetic-tape unit.
- n: specifies the number of blocks to be written. n = 0 means ten blocks.
- ii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location from which is read the first word to be written on tape. This word contains the preface.

Description of operation:

Summary:

Write n blocks on unit u ; the lane written on is that specified by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC. Words written on tape are read from consecutively-addressed storage locations beginning with $B[aaaa]$, except as noted below.

In storage the preface associated with each block is in the location which immediately precedes the location of the first word of the block (see description of storage allocation under MIR, page IV-55-5. For example, $(B[aaaa]:22)$ is the preface associated with the block whose first word is in $B[aaaa] + 1$. Before each block is overwritten, its preface (already on tape) is compared with the preface read from storage: the block is overwritten only if the two prefaces are identical; otherwise - except when the block on tape is an end-of-tape block - a preface-mismatch ALARM STOP occurs, with the read-write head positioned to read the block causing the ALARM STOP.

If a preface mismatch occurs with an end-of-tape block on tape, the ALARM STOP will not occur. Instead, the end-of-tape block is sensed, the control word is read from tape into the Computer control section, and terminating procedures are begun:¹

1. The contents of the P register are preserved:
 $(rP) \rightarrow aaaa:04$. This enables the program to determine the location of the instruction during whose

¹Recall that the format of the control word is $\pm ii aaaa bbbb$.

execution the end-of-tape block was encountered.

2. The contents of the address part of the C register are preserved: (rC:04) → aaaa:64. This enables the program to determine the location in core storage of the last word written on tape.
3. The address of the location of the next instruction selected for execution - an entry to the end-of-tape routine - is transferred to the P register:
B[bbbb] → rP.

After reading the end-of-tape block, MOR is terminated.

After termination of MOR, tape is positioned so that the head can write the block following the one which caused termination.

Flow chart:

See page IV-57-6

Exceptional conditions:

1. Non-existent-address ALARM STOP
2. Preface-mismatch ALARM STOP
3. Parity-error ALARM STOP
4. Not-write-status ALARM STOP
5. End-of-tape ALARM STOP
6. Not-ready ALARM STOP

Remarks:

1. Instruction formats for MOW and MOR are very much alike: the formats differ in digit positions 3 and 4 which are used to specify the preface in MOW and are noted as "not relevant" for MOR. However, even in a MOR instruction, digit positions 3 and 4 must not contain 02, 03, 04, 05, 06, 07, 08, or 09: if one of these configurations does appear, an improper-block-length ALARM STOP will occur, with the read-write head being positioned at its starting position.

2. If the unit referred to by this instruction is in not-write status - by virtue of lacking a write ring, for example - an ALARM STOP will occur.

3. The allocation of storage is as depicted for MIR.

4. Control blocks are not recognized as such during the execution of a MOR instruction.

5. Blank tape - that is, tape on which no information is recorded - is treated like interblock gap.

6. Note that a MOR instruction may terminate without having written n blocks as specified.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 58

Operation name:

MAGNETIC-TAPE POSITION FORWARD

MAGNETIC-TAPE POSITION BACKWARD

MAGNETIC-TAPE POSITION AT END
OF INFORMATION

Abbreviation:

MPF

MPB

MPE

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	n	i	v	Op	i	i	i	i
---	---	---	---	---	----	---	---	---	---

Definitions:

±: if ± is odd, B register address-
modification will occur; other-
wise, there will be no such
modification.

u: designates magnetic-tape unit.

n: specifies the number of blocks
to be passed. n = 0 means ten
blocks.

i, not relevant to the execution of
iiii: these instructions.

v: variation designator.

v = 0: MAGNETIC-TAPE POSITION
FORWARD will be executed.

v = 1: MAGNETIC-TAPE POSITION
BACKWARD will be executed.

v = 2: MAGNETIC-TAPE POSITION
AT END OF INFORMATION will
be executed.

Op: operation code.

Computer time (μs):

fetch: 90
execute:
total:

Also see discussion of
timing, pages IV-Intro-
28, ff.

Description of operation:

Summary:

v = 0: MAGNETIC-TAPE POSITION FORWARD will be executed.

Move tape on unit u in the forward direction until n blocks have been passed. The lane in which counting is done is the lane referred to by the last instruction referring to a lane: MTS, MFS, MRW, MDA, MLS, MTC, MFC.

Control blocks and end-of-tape blocks are counted and passed without recognition of their control function.

After completion of MPF, the tape is positioned so that the read-write head can read or overwrite the block following the last block passed.

v = 1: MAGNETIC-TAPE POSITION BACKWARD will be executed.

Except that tape is moved in the opposite direction, this variation is identical with that specified by v = 0.

After completion of MPB, the tape is positioned to permit reading the last block passed.

v = 2: MAGNETIC-TAPE POSITION AT END OF INFORMATION will be executed.

Tape movement begins in the forward direction, and continues until a blank area longer than inter-block gap is sensed, at which time the direction of tape movement is reversed. Tape is then moved in the backward direction until a block is sensed, at which time tape movement stops with the head in position to (initial) write a block following the one which caused the operation to terminate.

Flow chart:

See page IV-58-5

Exceptional conditions:

1. End-of-tape ALARM STOP
2. Not-ready ALARM STOP

Remarks:

1. If $v \neq 1$ or 2, the MAGNETIC-TAPE POSITION FORWARD variation will be executed.

2. Both MPB and MPF treat blank tape -- that is, tape on which no information is recorded -- like inter-block gap. MPE, on the other hand, recognizes blank tape uniquely as described above.

3. For purposes of counting blocks during execution of a MPF or MPB instruction, preface words are used to indicate that a block is being passed. Thus, for example, when six preface words have been sensed, the Magnetic-Tape Control Unit is "aware" that six blocks have been passed.

4. If the reflective strip which marks the physical end-of-tape is encountered during the execution of a MPF instruction, no "operation complete" signal will be generated. So, the Magnetic-Tape Control Unit will appear to be busy if a MIB instruction is issued or if an attempt is made to use the tape unit (by attempting to execute a MRD instruction, for example). Manual intervention is required in order to proceed with processing.

Similar remarks apply to the situation which results when the reflective strip which marks the physical beginning-of-tape is encountered during the execution of a MPB instruction.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 59

Operation name:

MAGNETIC-TAPE INTERROGATE,
BRANCH

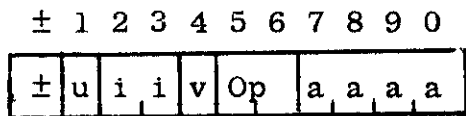
MAGNETIC-TAPE INTERROGATE
END OF TAPE, BRANCH

Abbreviation:

MIB

MIE

Instruction format:



Time (μs):

No branch:
 fetch: 90
 execute: 15
 total: 105

Definitions:

Branch:
 fetch: 90
 execute: 35
 total: 125

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

u: designates magnetic-tape unit.

ii: not relevant to the execution of these instructions.

v: variation designator:

v = 0: MAGNETIC-TAPE INTERROGATE, BRANCH will be executed.

v = 1: MAGNETIC-TAPE INTERROGATE END OF TAPE, BRANCH will be executed.

Op: operation code.

aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

v = 0: MAGNETIC-TAPE INTERROGATE, BRANCH will be executed.

If unit u is ready, transfer control to B[aaaa], i.e., prepare to take the next instruction from B[aaaa]; otherwise, control continues in sequence.

v = 1: MAGNETIC-TAPE INTERROGATE END-OF-TAPE, BRANCH will be executed.

If the magnetic End-of-Tape Indicator is "on", transfer control to B[aaaa]; otherwise, control continues in sequence.

Flow chart:

See page IV-59-4

Exceptional conditions:

1. Not-ready ALARM STOP

Remarks:

1. If $v \neq 1$, the MAGNETIC-TAPE INTERROGATE, BRANCH variation will be executed.

2. Because operations independent of both tape-control unit and Computer (i.e., MRW, MDA) as well as operations independent only of the Computer (e.g., MTS) can be in process, it is necessary to distinguish four cases for MIB. These are shown in the table below.

Status of tape - control unit	Status of tape - storage unit	Branch?
Ready	Ready	Yes
Ready	Not ready	No
Not ready	Ready	No
Not ready	Not ready	No

Introduction

Paper-tape input and output facilities are provided by means of photo-electric readers and punches. Interchangeable with the latter are character-at-time printers (physically and functionally these printers are identical with the supervisory printer). Each of the printers is capable of being driven off line by a mechanical reader which may be attached to it.

The Photo-electric Reader

The paper-tape reader used by the DATATRON 220 reads seven-channel single-frame paper. The code is shown in Appendix 3. The code is of the odd-parity type: provision is made for the detection of parity errors; in case a parity error is detected a parity-error ALARM STOP will occur.

When reading in the numeric mode, the appearance of a code for a non-numeric character -- the letter "A", for example -- will be detected as an error and an inadmissible-character ALARM STOP will occur.

The photo-electric reader utilizes two sizes of single-flange plastic reels: a five-and-one-half-inch reel which holds 350 feet of tape, and a seven-inch reel which holds 700 feet of tape. Tape is moved at the rate of 100 inches per second. Since information is punched ten characters per inch, the transfer rate is 1000 characters per second.

Start time is less than five milliseconds. The reader will stop on the stop character. That is, the reader will stop with the last character read still under the reading head.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

As many as ten photo-electric readers may be included in a DATATRON 220 system. Selection of a particular reader is under program control. In case the reader called for is inoperative, a not-ready ALARM STOP will occur.

Word format

Provision is made in the order structure for reading information with the sign digit punched first -- this is normal format -- as well as information with the sign digit punched last -- this is inverse format.

A word, as it is punched on paper tape, is defined to be the information bounded by two end-of-word characters (except for the first word, which does not have an end-of-word character preceding it). As each character is read from paper tape it is transferred to rD:01, shifting the contents of the D register one place to the left and causing the content of rD:±1 to be lost, i.e., the sign digit is shifted out of the D register.

In normal format, the sign digit is the first character of the word on paper tape; in inverse format, the sign digit is the character immediately preceding the end-of-word character.

Alphanumeric words are distinguished both on paper tape and in the computer by having the symbol "2" punched as the sign digit. The recognition of this character causes automatic translation of the single-frame alphanumeric characters in the word into the two-decimal-digit DATATRON code.¹ The sign digit is read and translated as a "2", (the digit 2 being used internally by the DATATRON to flag alphanumeric words.)

When reading inverse format, however, alphanumeric translation is not possible, because the sign digit is the last character sensed as the word is read. After the word in inverse

¹ Of course, when alphanumeric words are read and translated, the contents of the D register are shifted twice for each character read, once when the "zone" part of the character is sent to rD:01 from the decoding matrix, and a second time when the "numeric" part is sent to rD:01.

format has been read, the contents of the D register are permuted so that the sign of the word is in its proper position. The sign digit is then checked; if it is 2 -- indicating that the word is supposed to be alphanumeric-- an alphanumeric-sign ALARM STOP will occur.

The appearance of any numeric digit in the sign-digit position of a word on paper tape will cause the word to be translated numerically. Certain of the numeric digits, namely 6, 7, 8, and 9, in the sign-digit position are used for control purposes. Generally speaking, a 6 or 7 indicates that the word is to go to the C register, instead of to storage, without or with B register address-modification, respectively; an 8 or 9 in the sign-digit position indicates that the word is to be B register address-modified as it goes to storage. It is possible, it should be noted, to read from paper tape a word with any digit in the sign position, and to store that word exactly as it appeared on tape.

The reader is referred to the Execute Phase description of each input instruction for an explicit and detailed specification of sign-digit control. Except as noted in those specifications, the sign digits are read and translated exactly as they are punched in tape.

Instructions

There are two instructions for reading information in normal format. The first of these, PAPER-TAPE READ (see pages V-03-2,ff.), specifies that a fixed number of words (from 1 to 100) shall be read. However, the instruction may be coded to permit overriding of this specification: when so coded, the instruction says, in effect, "Read nn words, unless a control word is encountered, in which case terminate the reading operation."

The second instruction for reading information in normal format, PAPER-TAPE READ, BRANCH (see pages V-04-2, ff.), causes information to be read until a control word is encountered. The control word causes the reading operation to terminate.

There is one instruction for reading information in inverse format, namely, PAPER-TAPE READ, INVERSE FORMAT (see pages V-05-2, ff.). This instruction specifies that a fixed number of words - from one to 100 - shall be read unless a control word is encountered. The control word causes the reading operation to terminate. With this instruction it is not possible to read into storage a word with a 2, 6, or 7 in the sign-digit position. (The reader is reminded that a 2 in the sign-digit position causes an alphanumeric-sign ALARM STOP.)

The control word mentioned in the preceding paragraphs is a word flagged with a 6 or 7 in the sign-digit position. The detection of such a word terminates the instruction being executed, disconnects the reader, and causes preparations to be made for transferring the word to the C register for execution. The reading operation stops with the control word in the information buffer register, after which control is transferred to the Fetch Phase circuits. During the Fetch Phase -- which see, pages II-Intro-14 -- the control word is taken from the IB register to the C register -- without B register address-modification if the sign digit is 6, with such modification if the sign digit is 7 -- where it is regarded as an instruction. This is to say that a control word must be an instruction.

The Paper-Tape Punch

The paper-tape punch used by the DATATRON 220 punches seven-channel single-frame tape at the rate of 60 characters per second. That is to say, it punches tape which can be used for input to a DATATRON 220 System. However, it punches tape in normal format, automatically supplying the end-of-word character after the last character has been punched.

As many as ten paper-tape punches may be included in a DATATRON 220 system. Selection of a particular punch is under program control. In case the punch called for is inoperative, a not-ready ALARM STOP will occur.

Alphanumeric words - which are flagged internally with a 2 in the sign-digit position - are translated automatically to the single-frame code for punching in tape. The digit "2" - which also identifies the word in tape as alphanumeric - is punched as a "2".

Provision exists for the suppression of alphanumeric translation, in which case the contents of the word are punched as 11 decimal digits.

There is also provision for the suppression of high-order zeros. This feature is provided by the SUPPRESS LEADING ZEROS switch on the Control Console. When the SUPPRESS LEADING ZEROS switch is "on", the punching of zeros preceding the first non-zero digit in the word will be suppressed. This means that no zeros will be suppressed if the sign digit is different from 0.

The Printer

In place of any paper-tape punch there may be substituted a character-at-a-time printer. Naturally, this printer is logically equivalent to the punch which it replaces. The printer is physically and functionally identical with the supervisory printer described in Section III.

The reader is referred to Section III for a description of the printer.

The Mechanical Reader

Any printer - including the supervisory printer - in a DATATRON 220 System may have attached to it a mechanical reader which is capable of reading DATATRON 220 paper tape. The reader provides off-line facilities for printing the contents of paper tape at the printer's rate of 10 characters per second.

Except for its ability to check for parity errors, the mechanical reader and its circuits are logically equivalent to the computer's circuits when the printer is driven on line.

Information flow

The general nature of information flow during input and output was discussed in Section II. In some details, however, the actual flow of information is different from that indicated in Section II. For this reason, the flow of information is described again below.

Input flow

Input flow is shown in figure V-Intro-1.

1. The address part of the C register - which specifies the location in storage where the next word read from paper tape

is to be stored - is transferred to the E register. At the same time, the address part of the C register is counted up 1.

2. The word is read from paper tape through the translator and into the D register.

3. a. If the instruction is PRI, the contents of the D register are permuted so that the sign is in its normal position.

b. The contents of the D register are then sent to the information buffer register, with or without B register address modification as specified and indicated, after which the D register is cleared.

4. a. If the word is to go to storage, the contents of the information buffer register are stored in the location whose address is in the E register.

b. If the word is to go to the C register, the reader is disconnected, and preparations are made to enter the Fetch Phase, during which the contents of the information buffer register are transferred, with or without B register address modification as indicated, to the C register, where the word is regarded as an instruction.

If the instruction has not been terminated, there is a return to step 1 for the next word.

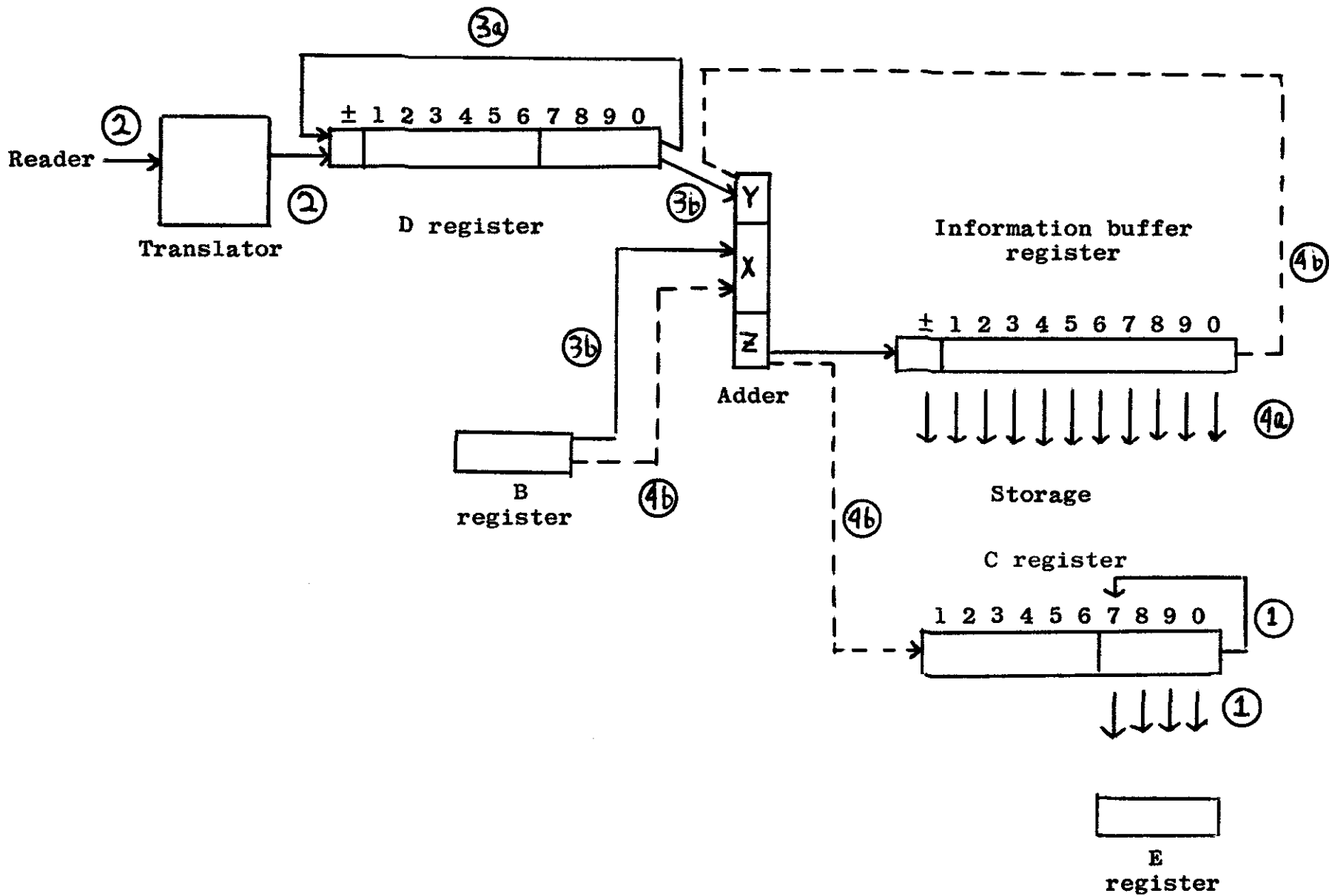


Figure V-Intro-1. Input flow

Output flow

Output flow is shown in figure V-Intro-2.

1. The address part of the C register — which specifies the location in storage from which the next word to be written will be taken — is transferred to the E register. At the same time, the address part of the C register is counted up 1.
2. The word is transferred from storage to the information buffer register.
3. The contents of the information buffer register are transferred to the D register.
4. a. The contents of the D register are permuted so that the (rD:±1) appear in rD:01.
b. The contents of rD:01 pass through the adder to the translator and thence to the printer or punch.
4a. and 4b. are repeated until the entire word has been printed or punched.

If the instruction has not terminated, there is a return to step 1 for the next word.

Exceptional Conditions

Following is a list defining exceptional conditions which can occur and are detected. In addition to those enumerated below, a non-existent-address ALARM STOP — which was defined in Section II — can occur.

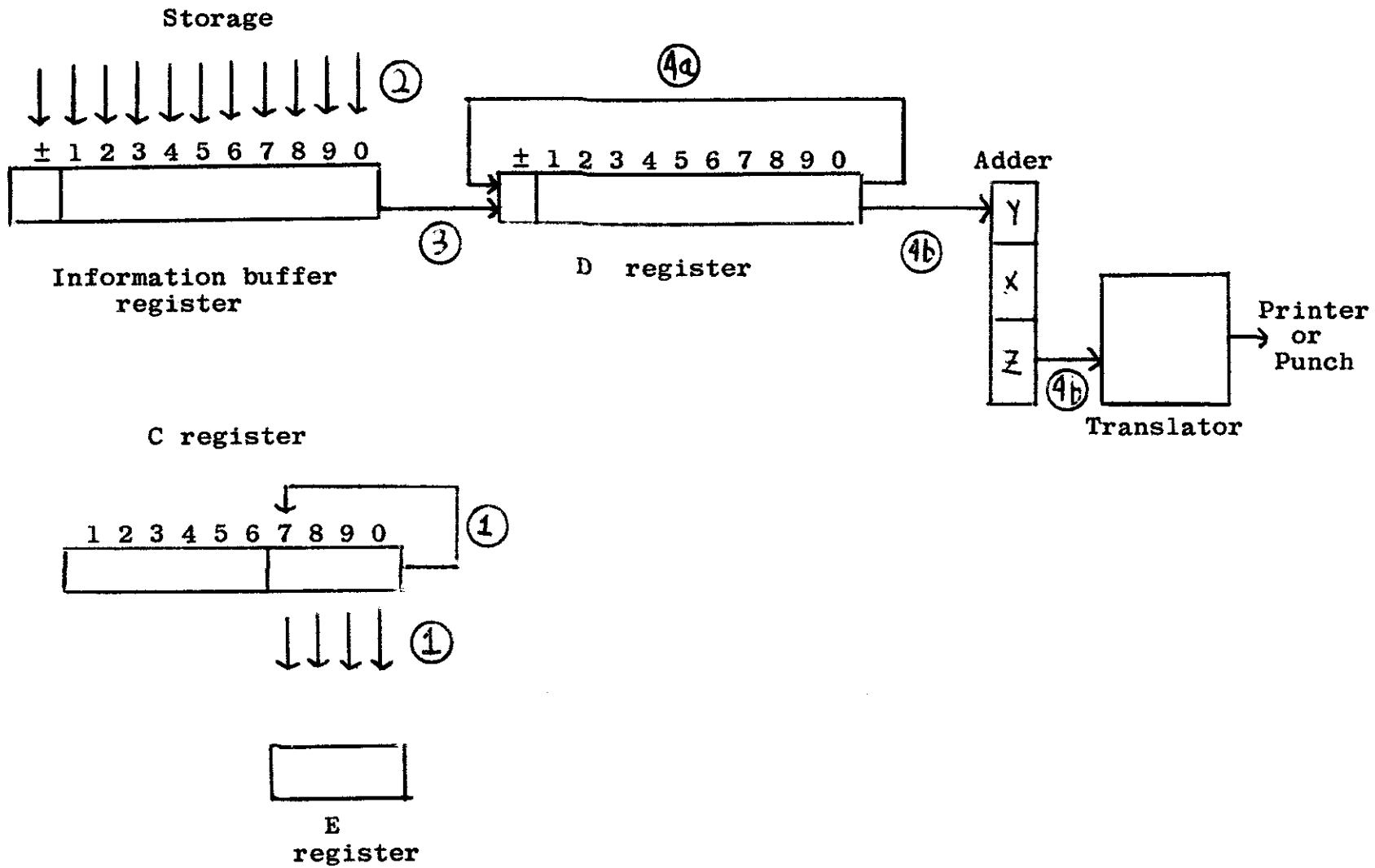


Figure V-Intro-2. Output flow

1. The seven-channel code uses an odd-parity check. Whenever a character is sensed which has an even number of channels punched, the fact is signaled by a parity-error ALARM STOP.

2. The detection of a code for a non-numeric character when reading in the numeric mode is signaled by an inadmissible-character ALARM STOP.

3. When executing PAPER-TAPE READ, INVERSE FORMAT, the appearance in the sign-digit position of a "2" is detected and an alphanumeric-sign ALARM STOP occurs.

4. If the input or output unit designated by the instruction is inoperative, a not-ready ALARM STOP will occur.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: PAPER-TAPE READ

Operation code: 03

Abbreviation: PRD

Instruction format:

Time (μ s):

+ 1 2 3 4 5 6 7 8 9 0

+	u	n	n	v	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

fetch: 90
 execute: $\frac{\text{reader speed}}{\text{reader speed}}$
 total: $\frac{\text{reader speed}}{\text{reader speed}}$

Definitions:

- ±:** if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u:** designates paper-tape reader unit.
- nn:** specifies the number of words to be read. nn = 00 means 100 words.
- v:** variation designator:
 - v = 1: specified word goes to C register for execution; reader is turned off.
 - v = 8, 9: specified input will be B-register address-modified.
- Op:** operation code.
- aaaa:** address of base of location into which is written first word read from paper tape.

Description of operation:

Summary:

v ≠ 1:

Read nm words from unit μ into consecutively-addressed locations beginning with B[aaaa].

v = 1:

Read nm words, or until a word with sign digit equal to 6 or 7 is encountered, from unit u into consecutively-addressed locations beginning with B[aaaa]. An input word with sign digit equal to 6 or 7 goes to the C register, without or with B-register address-modification, respectively, for immediate execution; the reader is turned off.

Input sign-control:

If the one-bit of v is equal to 1:

- a. If the input sign digit is 6, the word goes to rC for immediate execution; the reader is turned off.
- b. If the input sign digit is 7, the word goes to rC, with B-register address-modification, for immediate execution; the reader is turned off.

If the eight-bit of v is equal to 1:

- a. If the input sign digit is 8, the word is B-register address-modified; the sign goes into the specified storage location as 0.
- b. If the input sign digit is 9, the word is B-register address-modified; the sign goes into the specified storage location as 1.

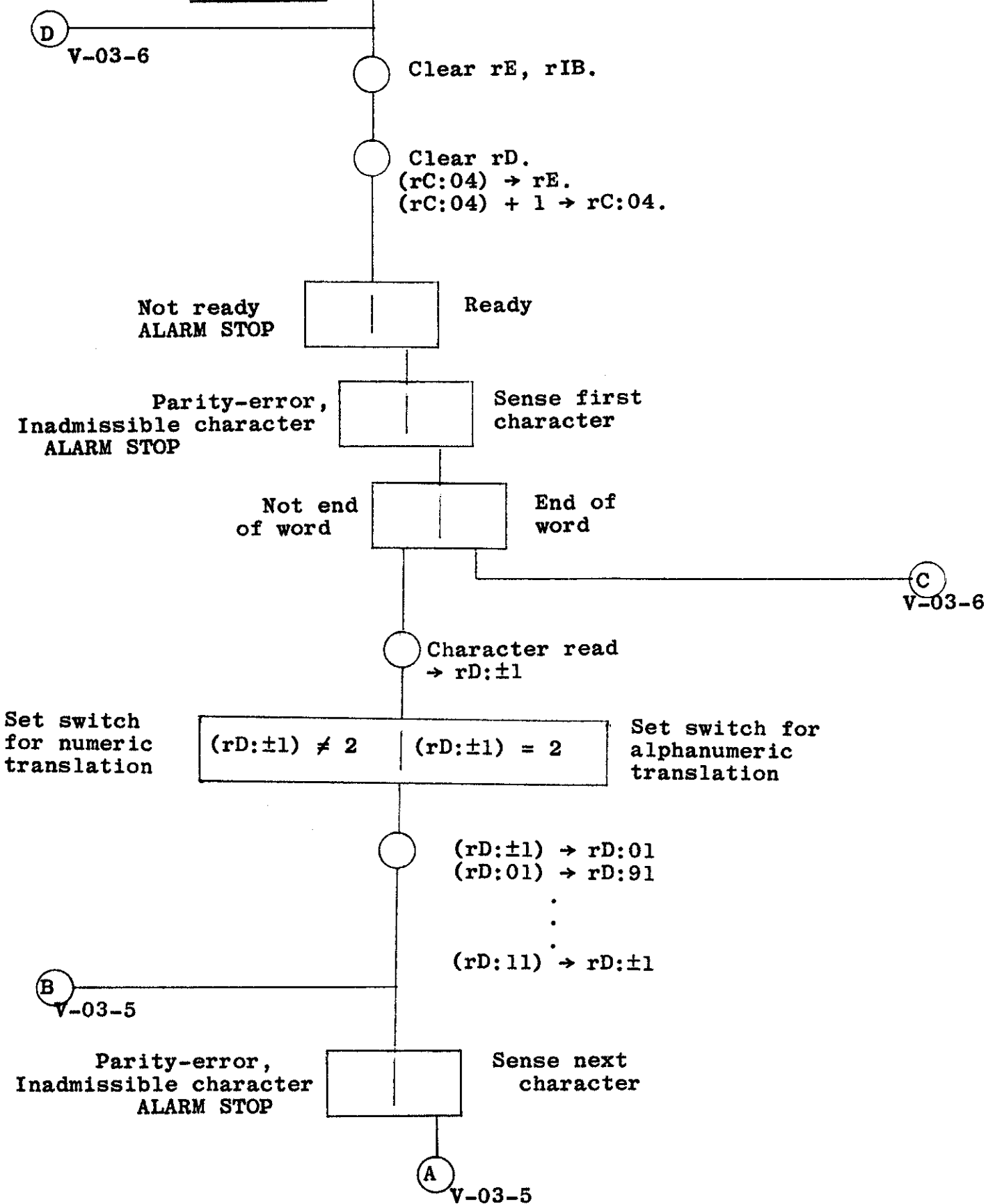
Flow chart:

See page V-03-4.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

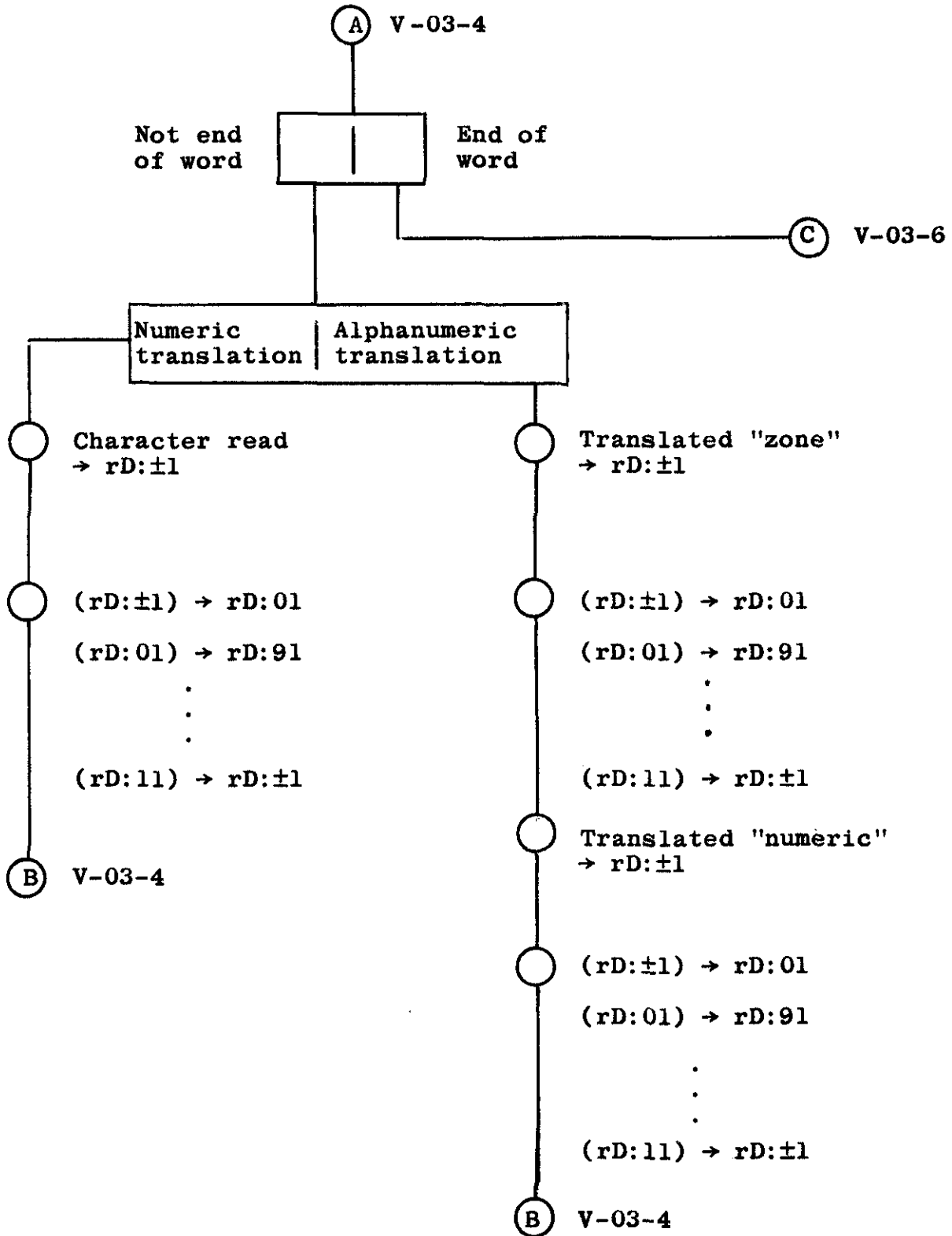
Description of Operation:

Flow Chart:



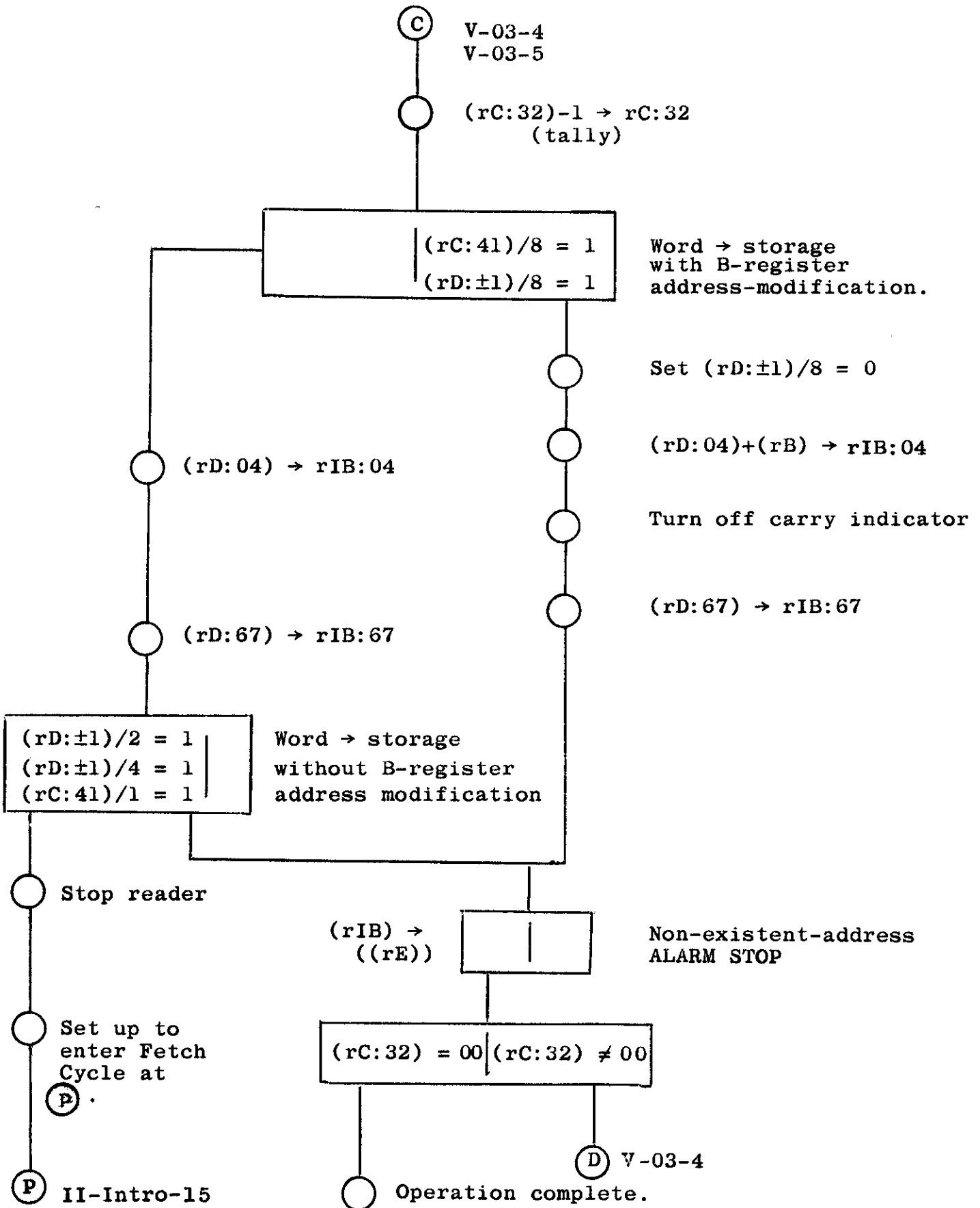
THE PAPER-TAPE SYSTEM

Flow chart (continued):



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Flow chart (continued):



Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Parity-error ALARM STOP.
3. Inadmissible-character ALARM STOP.
4. Not-ready ALARM STOP.

Remarks:

- 1.

Register status:

Register name	Contents after execution of PRD; no override of nn.	Contents after execution with override of nn.																				
A	Unchanged	Unchanged																				
R	"	"																				
D	Last word read	Last word read.																				
B	Unchanged	Unchanged																				
P	$(rP)_b + 1$	$(rP)_b + 1$																				
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[aaaa]</td> <td style="padding: 2px;">+</td> <td style="padding: 2px;">nn*</td> </tr> </table>	u	0	0	v	0	3	B	[aaaa]	+	nn*	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> </tr> </table>	u	*	*	v	0	3	*	*	*	*
u	0	0	v	0	3	B	[aaaa]	+	nn*													
u	*	*	v	0	3	*	*	*	*													
B	$B[aaaa] + nn - 1^*$	***																				
	* If nn=00, add 100.	**, ***, ****; See description of operation.																				

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register name	Contents if non-existent-address ALARM STOP occurs
---------------	----------------------------------------------------

A	Unchanged										
R	"										
D	Last word read.										
B	Unchanged										
P	$(rP)_b + 1$										
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> </tr> </table>	u	*	*	v	0	3	*	*	*	*
u	*	*	v	0	3	*	*	*	*		
E	address causing ALARM STOP.										

** nn minus number of words read.

**** Sum: address causing ALARM STOP + 1

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 04

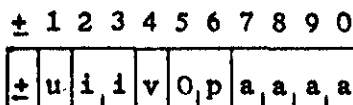
Abbreviation: PRB

Time (μ s):

fetch: 90
 execute: reader speed
 total: reader speed

Operation name: PAPER-TAPE READ, BRANCH

Instruction format:



Definitions:

- ±: if ± is odd, B-register address modification will occur; otherwise, there will be no such modification.
- u: designates paper-tape reader unit.
- ii: not relevant to the execution of this instruction.
- v: variation designator:
 - v = 8, 9: specified input will be B-register address-modified.
 - v = 0: no B-register address modification of input.
- Op: operation code.
- aaaa: address of base of location into which is written first word read from paper tape.

Description of operation:

Summary:

Read from unit u, into consecutively-addressed locations beginning with B[aaaa], until a word with sign-digit equal to 6 or 7 is encountered. An input word with the sign digit equal to 6 or 7 goes to the C register, without or with B-register address-modification, respectively, for immediate execution; the reader is turned off.

Input sign-control:

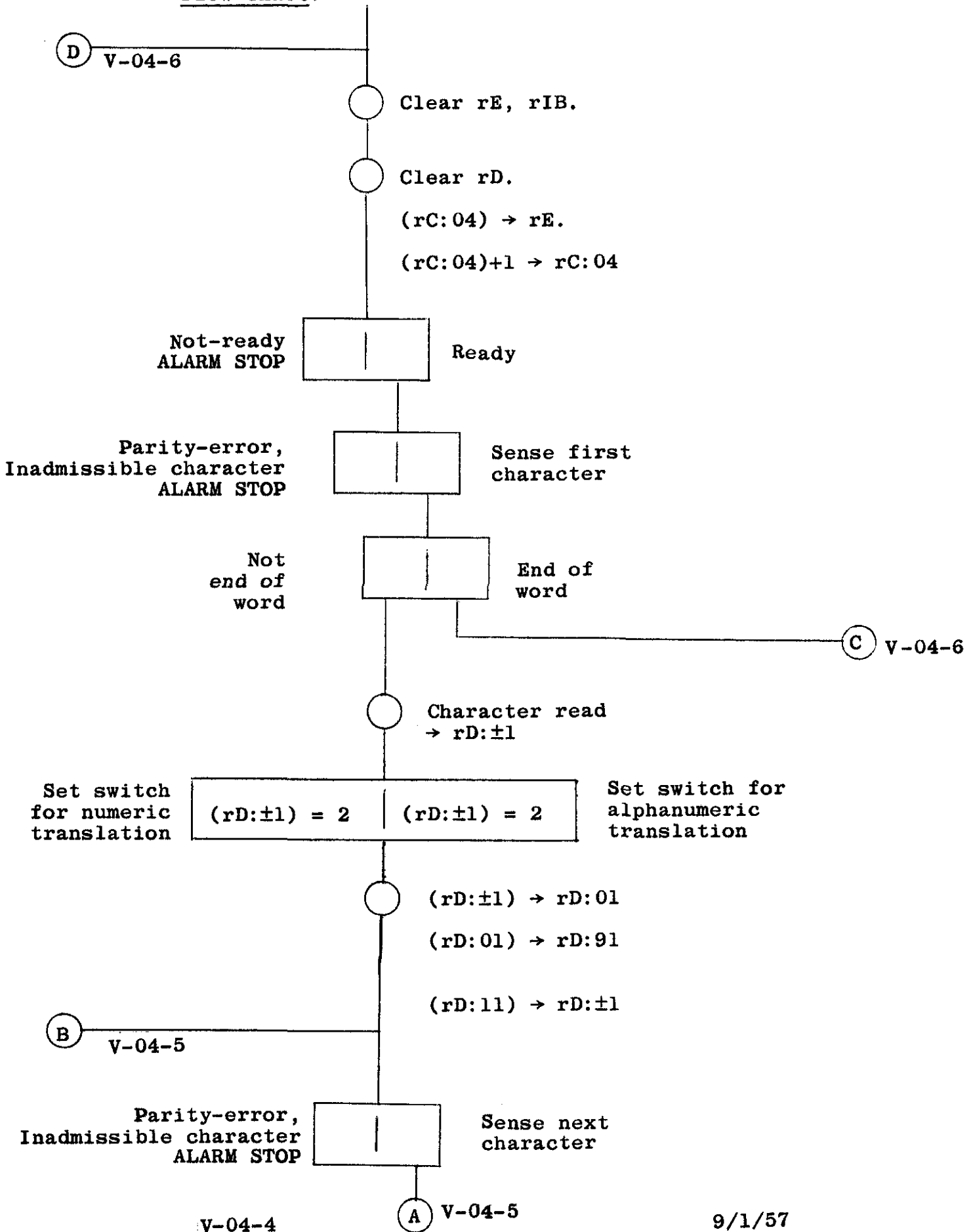
- a. If the input sign digit is 6, the word goes to rC for immediate execution; the reader is turned off.
- b. If the input sign digit is 7, the word goes to rC, with B-register address modification, for immediate execution; the reader is turned off.
- c. If the eight-bit of v is equal to 1:
 1. If the input sign digit is 8, the word is B-register address-modified; the sign goes into the specified storage location as 0.
 2. If the input sign digit is 9, the word is B-register address-modified; the sign goes into the specified storage location as 1.

Flow chart:

See page V-04-4.

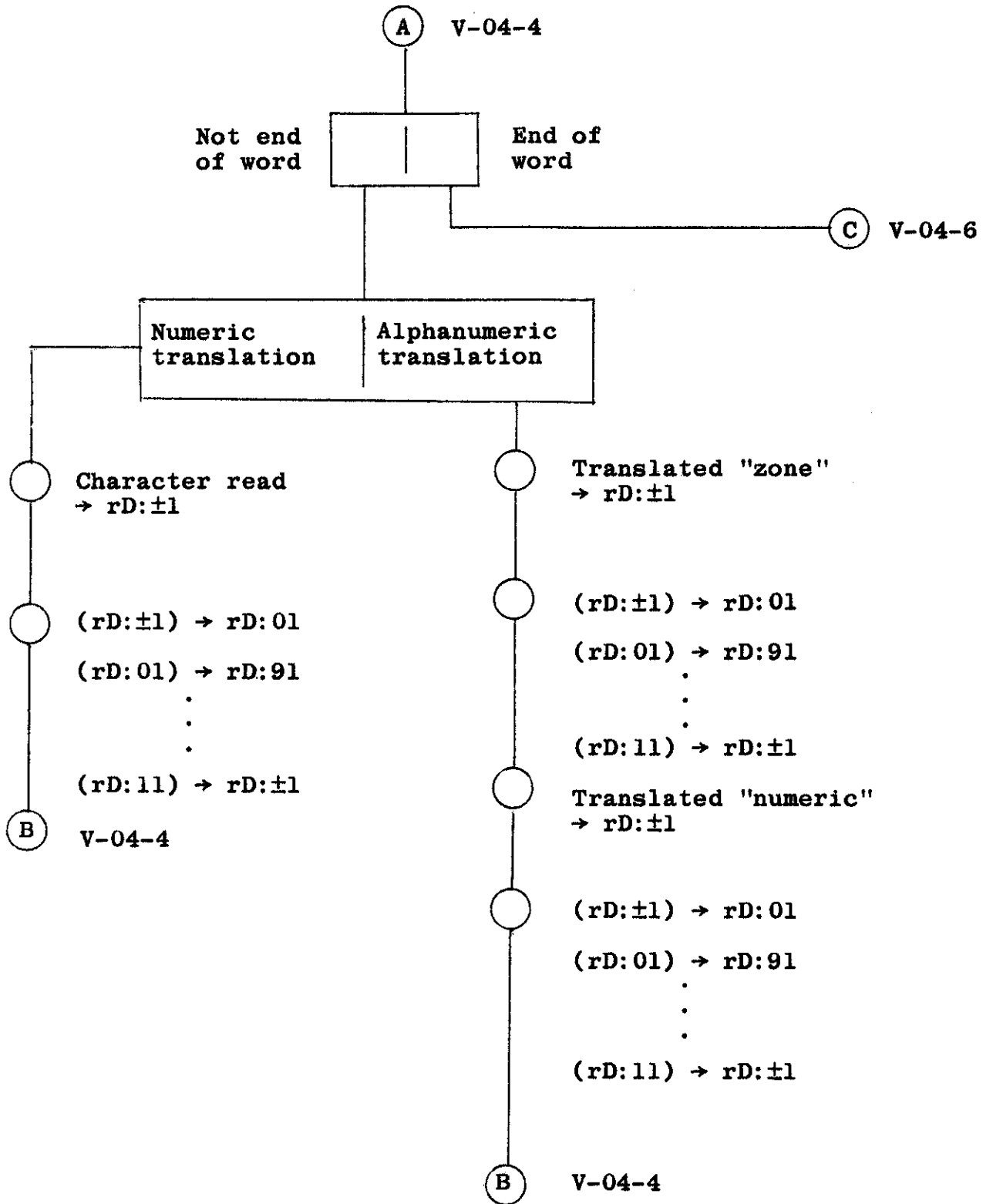
Description of Operation:

Flow chart:

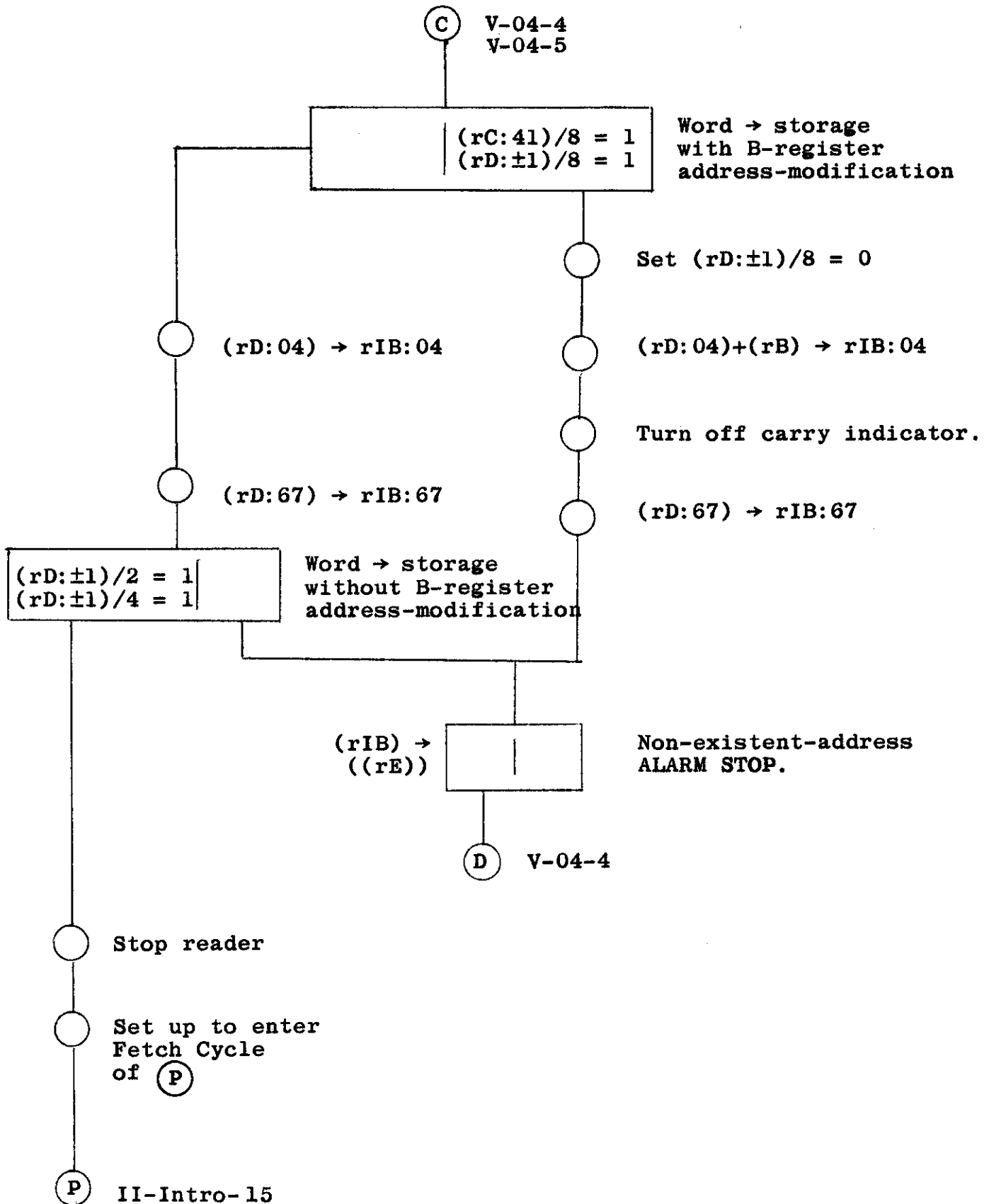


THE PAPER-TAPE SYSTEM

Flow chart (continued):



Flow chart (continued):



THE PAPER-TAPE SYSTEM

Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Parity-error ALARM STOP.
3. Inadmissible-character ALARM STOP.
4. Not-ready ALARM STOP.

Remarks:

Register status:

Register name	Contents after execution of PRB	Contents after non-existent-address ALARM STOP.																				
A	Unchanged	Unchanged																				
R	"	"																				
D	Last word read.	Last word read																				
B	Unchanged	Unchanged																				
P	$(rP)_b + 1$	$(rP)_b + 1$																				
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td>u</td><td>i</td><td>i</td><td>v</td><td>0</td><td>4</td><td>*</td><td>*</td><td>*</td><td>*</td> </tr> </table>	u	i	i	v	0	4	*	*	*	*	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td>u</td><td>i</td><td>i</td><td>v</td><td>0</td><td>4</td><td>*</td><td>*</td><td>*</td><td>*</td> </tr> </table>	u	i	i	v	0	4	*	*	*	*
u	i	i	v	0	4	*	*	*	*													
u	i	i	v	0	4	*	*	*	*													
E	Address of last location filled.	Address causing ALARM STOP.																				
	**** Sum: address of last location filled + 1.	**** Sum: address causing ALARM STOP + 1.																				

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation code: 05

Operation name: PAPER-TAPE READ, INVERSE
FORMAT

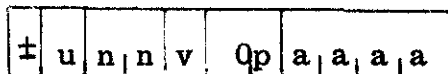
Abbreviation: PRI

Time (μ s):

Instruction format:

fetch: 90
execute: reader speed
total: reader speed

\pm 1 2 3 4 5 6 7 8 9 0



Definitions:

- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates paper-tape reader unit.
- nn: specifies the number of words to be read. nn=00 means 100 words.
- v: variation designator:
 - (rc:41)/8=1: specifies input going to storage will be B-register address-modified.
- Op: operation code.
- aaaa: address of base of location into which is written first word read from paper tape.
- v: v=8 or 9: designated input will be B-register address-modified.

Description of operation:

Summary:

Read nn words, or until a word with sign digit equal to 6 or 7 is encountered, from unit u into consecutively-addressed locations beginning with B [aaaa]. An input word with sign digit equal to 6 or 7 goes to the C register, without or with B-register address-modification, respectively, for immediate execution; the reader is turned off.

Input sign-control:

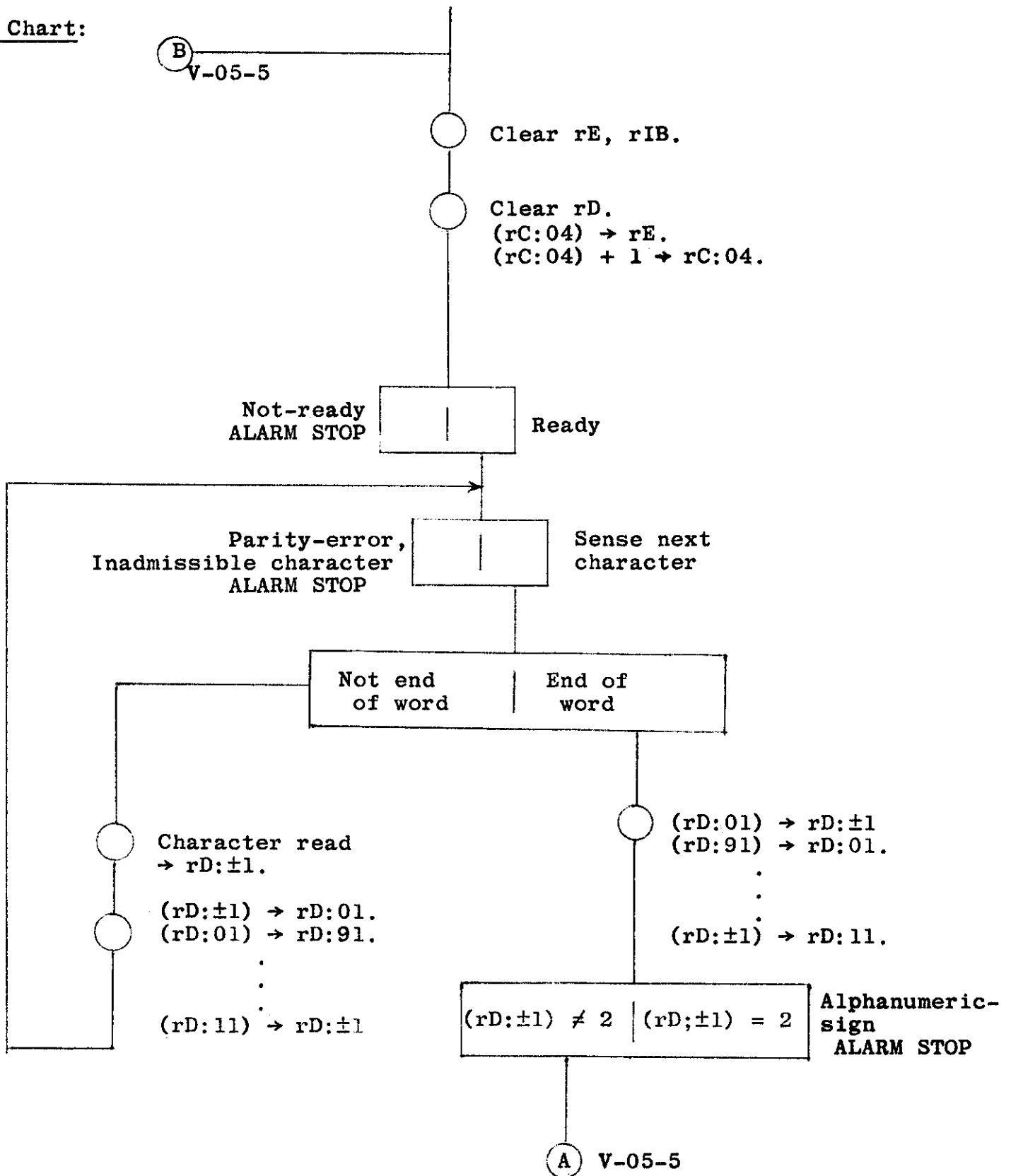
- a. If the input sign digit is 6, the word goes to rC for immediate execution; the reader is turned off.
- b. If the input sign digit is 7, the word goes to rC, with B-register address-modification, for immediate execution; the reader is turned off.
- c. If the eight-bit of v is equal to 1:
 1. If the input sign digit is 8, the word is B-register address-modified; the sign goes into the specified storage location as 0.
 2. If the input sign digit is 9, the word is B-register address-modified; the sign goes into the specified storage location as 1.

Flow chart:

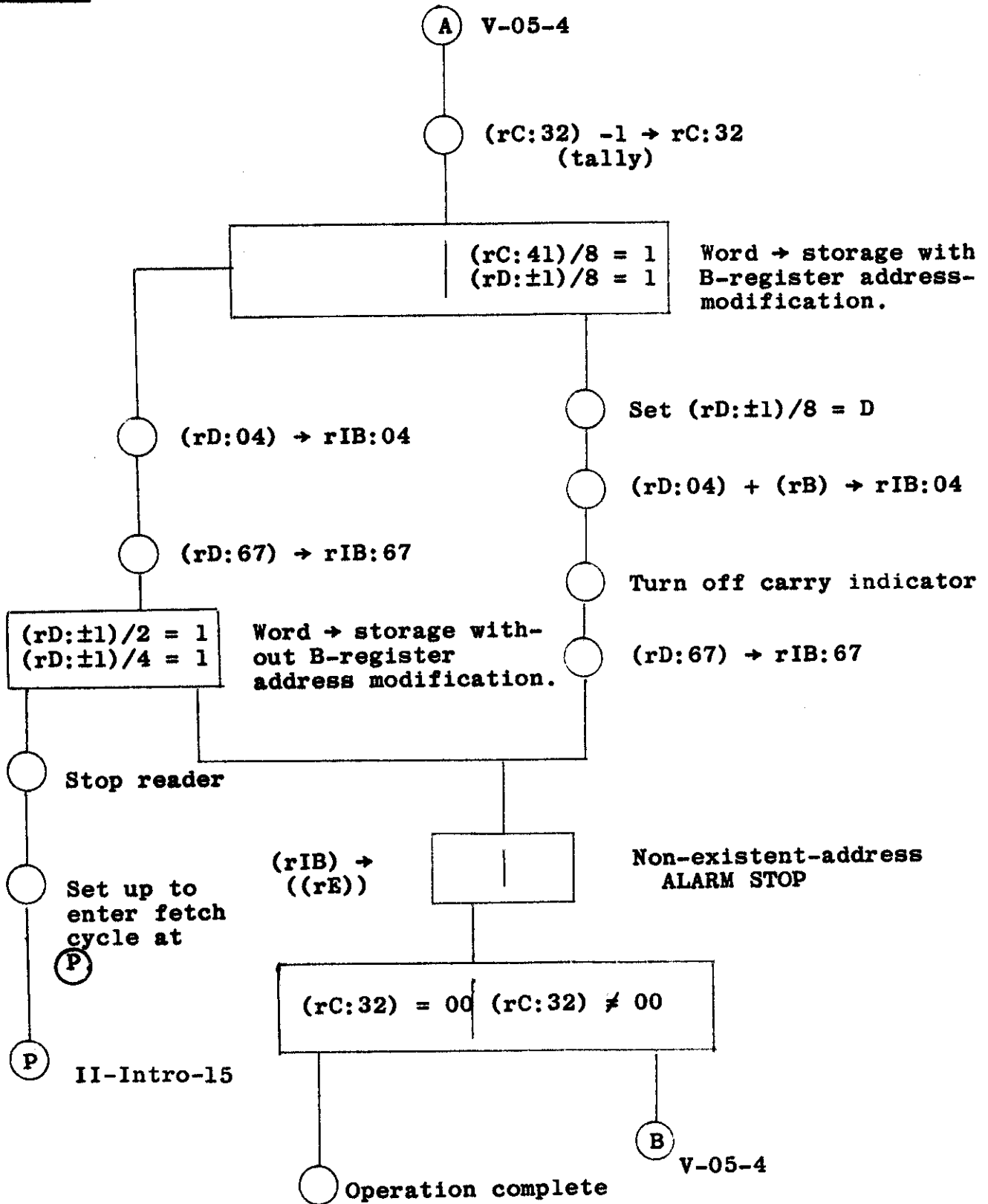
See page V-05-4.

Description of operation:

Flow Chart:



Flow Chart (Continued):



Exceptional conditions:

1. Non-existent-address ALARM STOP.
2. Parity-error ALARM STOP.
3. Inadmissible-character ALARM STOP
4. Alphanumeric-sign ALARM STOP.
5. Not-ready ALARM STOP.

Remarks:

1. Because the sign digit is read last, alphanumeric translation is not possible.

THE PAPER-TAPE SYSTEM

Register status:

Register name	Contents after execution of PRI, no override of nn.	Contents after execution with override of nn.																		
A	Unchanged	Unchanged																		
R	"	"																		
D	Last word read	Last word read																		
B	Unchanged	Unchanged																		
P	$(rP)_b + 1$	$(rP)_b + 1$																		
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> <td style="padding: 2px;">+ nn*</td> </tr> </table>	u	0	0	v	0	5	B[aaaa]	+ nn*	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> </tr> </table>	u	*	*	v	0	5	*	*	*	*
u	0	0	v	0	5	B[aaaa]	+ nn*													
u	*	*	v	0	5	*	*	*	*											
E	$B[aaaa] + nn - 1*$	***																		

* If nn=00, add 100.

, *, **** See description of operation.

Register name	Contents if non-existent-address ALARM STOP occurs.										
A	Unchanged										
R	"										
D	Last word read										
B	Unchanged										
P	$(rP)_b + 1$										
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">v</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> </tr> </table>	u	*	*	v	0	5	*	*	*	*
u	*	*	v	0	5	*	*	*	*		
E	Address causing ALARM STOP										

** nn minus number of words read.
**** Sum: address causing ALARM STOP + 1.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

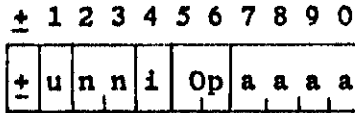
Operation name: PAPER-TAPE WRITE

Operation code: 06

Abbreviation: PWR

Instruction format:

Time (μ s):



fetch: 90
 execute: punch or printer speed
 total: punch or printer speed

Definitions:

- +**: if + is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u**: designates paper-tape punch unit.
- nn**: specifies number of words to be punched. nn=00 means 100 words.
- i**: not relevant to the execution of this instruction.
- Op**: operation code
- aaaa**: address of base of location from which is read first word to be punched

Description of operation:

Summary:

Write nn words on unit u, taking the words from consecutively-addressed locations beginning with B[aaaa].

Flow Chart:

See page V-06-4.

Exceptional Conditions:

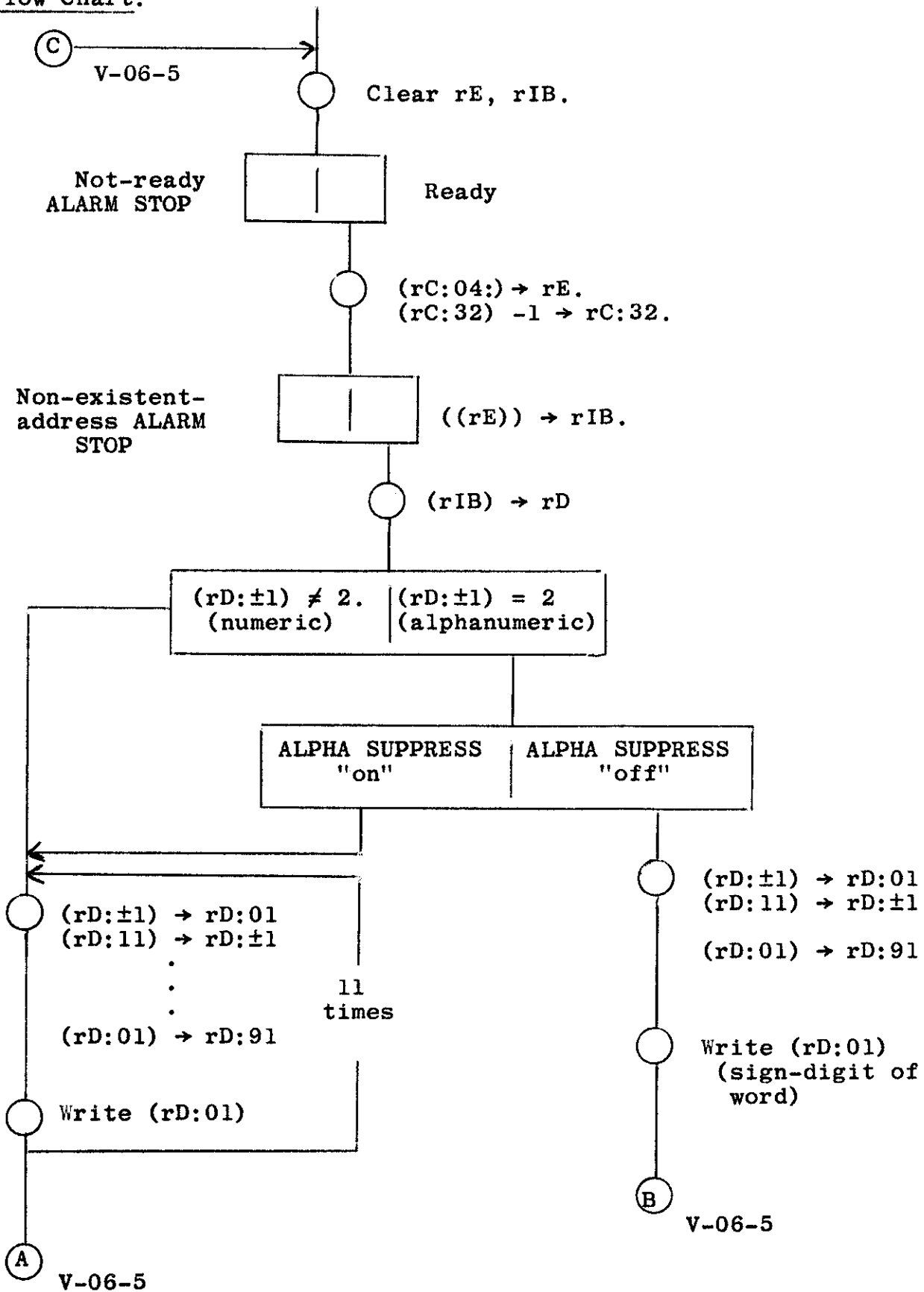
1. Non-existent-address ALARM STOP.
2. Not-ready alarm stop.

Remarks:

1. Alphanumeric translation may be suppressed by setting the ALPHA SUPPRESS switch on the writing unit to "on."
2. The writing of leading zeros may be suppressed by setting the ZERO SUPPRESS switch on the writing unit to "on."

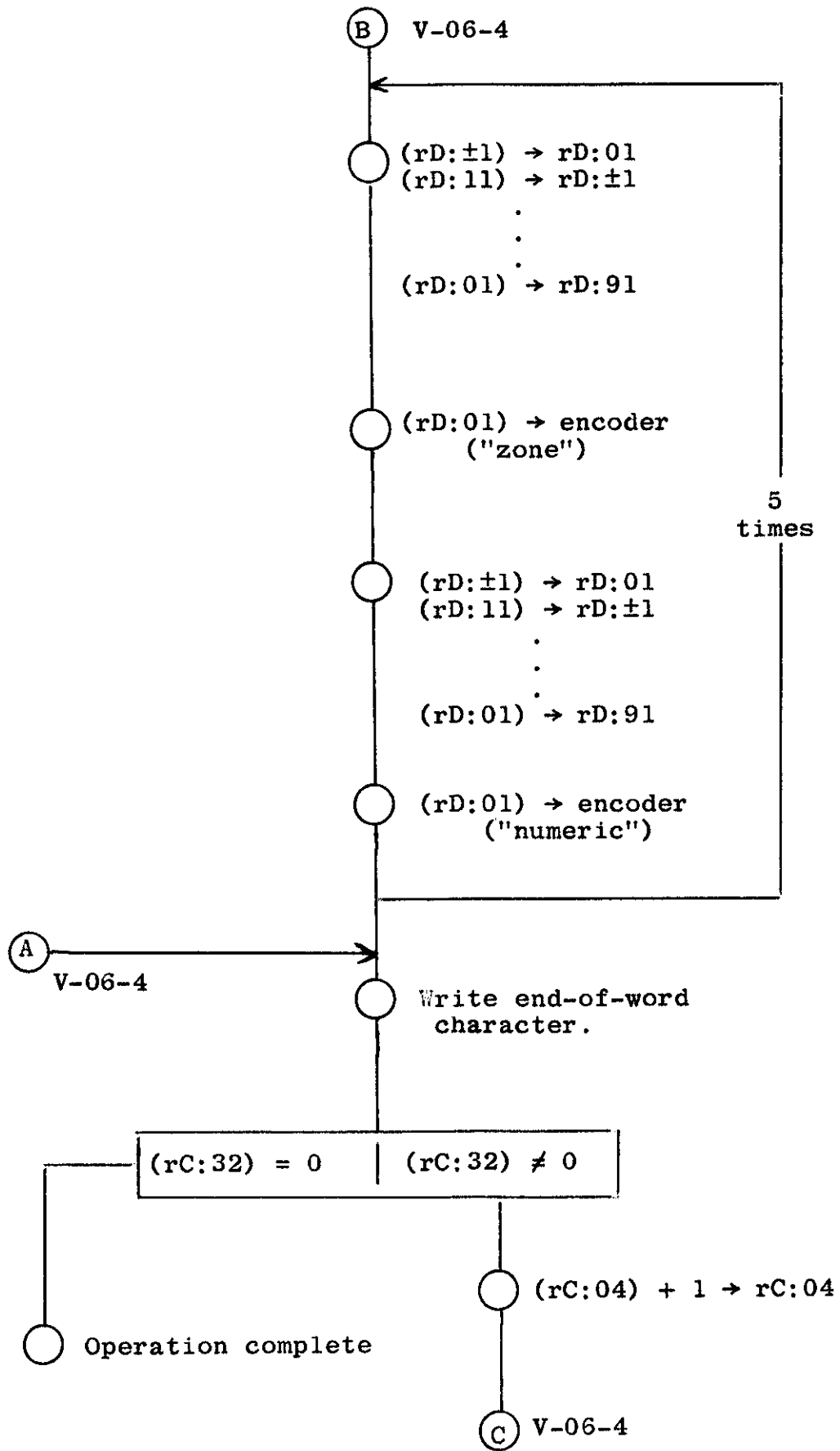
OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Flow Chart:



THE PAPER-TAPE SYSTEM

Flow Chart (Continued):



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status:

Register name	Contents after execution of PWR.	Contents if non-existent-address ALARM STOP occurs.																				
A	Unchanged	Unchanged																				
R	"	"																				
D	Last word punched	Last word punched																				
B	Unchanged	Unchanged																				
P	$(rP)_b + 1$	$(rP)_b + 1$																				
C	<table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">B</td> <td style="padding: 2px;">[aaaa]</td> <td style="padding: 2px;">+</td> <td style="padding: 2px;">nn*</td> </tr> </table>	u	0	0	i	0	6	B	[aaaa]	+	nn*	<table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> <td style="padding: 2px;">*</td> </tr> </table>	u	*	*	i	0	6	*	*	*	*
u	0	0	i	0	6	B	[aaaa]	+	nn*													
u	*	*	i	0	6	*	*	*	*													
E	$B[aaaa] + nn - 1^*$	Address causing ALARM STOP.																				

* if nn=00, add 100.

** nn minus number of words punched

**** Sum: address causing ALARM STOP + 1.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: PAPER-TAPE WRITE,
INTERROGATE, BRANCH

Operation code: 07

Abbreviation: PWI

Time (μ s):

Instruction format:

+ 1 2 3 4 5 6 7 8 9 0

+	u	i	i	i	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

No branch:

fetch:	90
execute:	<u>15</u>
total:	105

Branch:

fetch:	90
execute:	<u>35</u>
total:	125

Definitions:

+: if + is odd, B-register address-modification will occur; otherwise, there will be no such modification.

u: designates paper-tape punch unit.

iii: not relevant to the execution of this instruction.

Op: operation code.

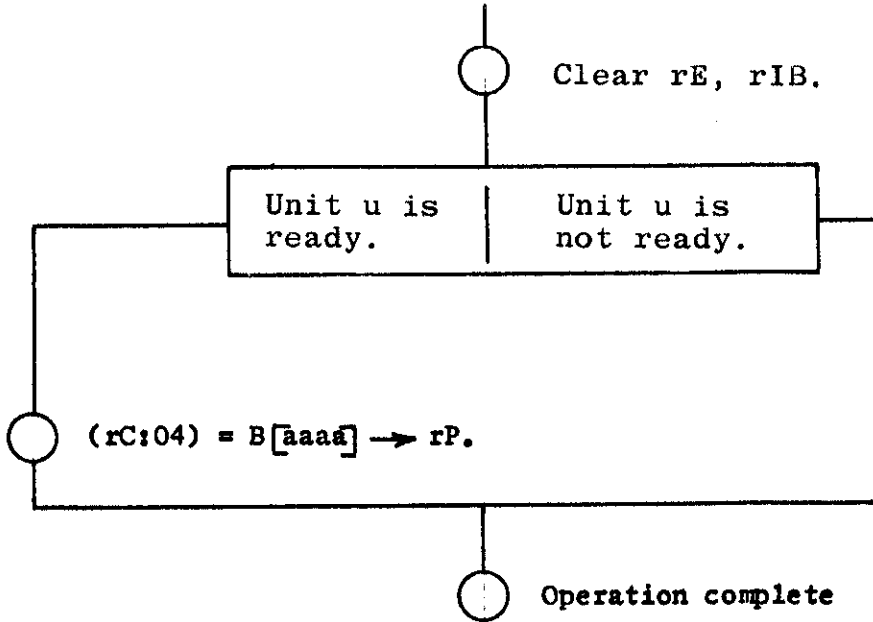
aaaa: address of base of location of alternate instruction.

Description of operation:

Summary:

If unit u is ready, transfer control to B[aaaa], i.e., prepare to take the next instruction from B[aaaa]. Otherwise, control continues in sequence.

Flow chart:



Exceptional conditions:

Remarks:

Register status:

Register name	Contents after execution of PWI											
A	Unchanged											
R	"											
D	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>+</td> <td>u</td> <td>i</td> <td>i</td> <td>i</td> <td>0</td> <td>7</td> <td>a</td> <td>a</td> <td>a</td> <td>a</td> </tr> </table>	+	u	i	i	i	0	7	a	a	a	a
+	u	i	i	i	0	7	a	a	a	a		
B	Unchanged											
P	$(rP)_b + 1$, if no branch B [aaaa], if branch											
C	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>u</td> <td>i</td> <td>i</td> <td>i</td> <td>0</td> <td>7</td> <td>B</td> <td>[aaaa]</td> </tr> </table>	u	i	i	i	0	7	B	[aaaa]			
u	i	i	i	0	7	B	[aaaa]					
E	Cleared											

INTRODUCTION

The Cardatron is a device by means of which relatively slow card-handling machines and the high-speed DATATRON 220 Data Processor are coupled. This purpose is accomplished by interposing a buffer between the card-handling mechanism and the Data Processor. The Data Processor communicates only with the buffer; the card-handling mechanism communicates only with the buffer. It is the purpose of this section to describe this communication system, in which any combination of up to seven input and output card-handling machines may be used.

Independently of Data Processor control, each card-handling machine can communicate with the buffer in the Input or Output Unit to which it is attached; hence, the several card-handling machines may be operating simultaneously. Whereas the information transfer rate of the IBM 089, for example, is not more than approximately 320 alphanumeric characters per second, the information transfer rate of the buffer--when it is communicating with the Data Processor--is approximately 44,000 digits per second, including access time. The Cardatron clock rate is actually 115,000 cycles per second.

The second distinctive feature of the Cardatron is its editing ability, which complements the editing and format control facilities of the control panel on the card-handling machine. On the input side, for example, each card can select one from among six different editing modes--five of them under program control--which not only edit the contents of the card in a variety of different ways but also automatically translate information into Data Processor code;

the contents of a card may be expanded to occupy selected portions of as many as 28 Data Processor locations.

Figure VI-Intro-1 shows the organization of a Cardatron system as well as the flow of information.

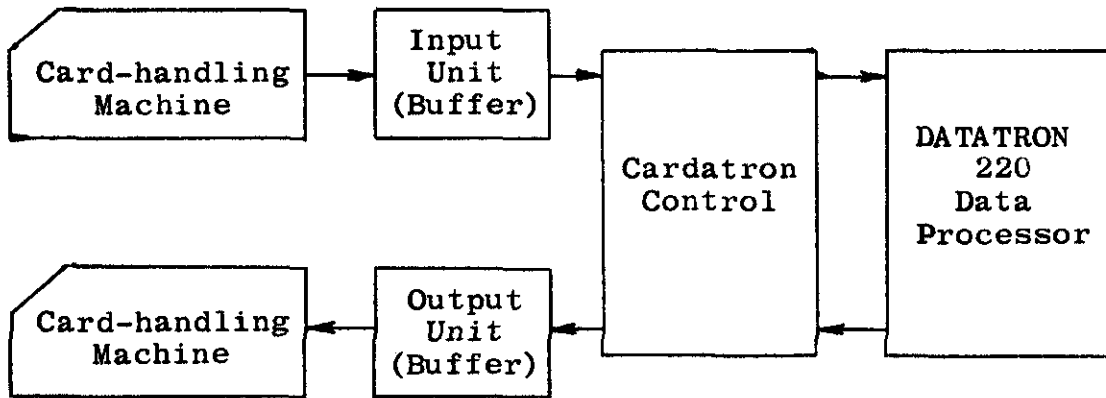


Figure VI-Intro-1

Block diagram of Cardatron system organization.

INPUT

THE INPUT UNIT

The Input Unit is made up of two essential parts, control circuitry and a magnetic drum (the buffer) which has on it six bands for storing information. One band--called the information band--is used to store information read from a card. The remaining bands--called format bands--are used to store editing control streams, that is, the sequence of digits used to control the editing of information on the information band.

Each of the bands on the buffer has provision for holding 315 digits. However, in the format bands as well as in the information

band, seven of these digit positions are not available for either editing control stream digits or the storage of information. Hence, under format band control the contents of a card may be expanded to a maximum of 308 digits, that is, to 28 Data Processor words.

Corresponding to each digit position in the information band is a digit position in each of the five format bands. Each of these digit positions is identified uniquely and in order by one of the numbers from 1 to 315. Digit position 1 is regarded as the origin of the band. Whereas each digit position in the information band consists of a full four-bit decade--and hence is capable of storing any one of the decimal digits--each digit position in a format band is comprised of only two bit positions: it is possible to write only digits 0, 1, 2, and 3 in a format band.*

INFORMATION TRANSFER

The transfer of information from a card to the Data Processor takes place in two parts. During the first part, information is transferred from the card to the information band. This is accomplished under control of an Input Unit, independently of the Data Processor. During the second part, the contents of the information band are transferred to the Data Processor by the execution of a CARD READ instruction. After the contents of the information band have been transferred to the Data Processor, the Input Unit

*If an attempt is made to load--by executing a CARD READ, FORMAT LOAD instruction--as a format band digit a decimal digit greater than or equal to 4, the four bit or the eight bit will be ignored.

automatically directs the transfer of the contents of the next card--if any--to the buffer unless directed by the CARD READ instruction not to read the next card; in this case we say that Reload Lockout has been imposed.

PART 1: FROM CARD TO BUFFER (See Figure VI-Intro-2.)

A part of the editing process occurs when the contents of a card are transferred to the buffer. For example, 0's are inserted. In addition, part of the conversion from card code to Data Processor code is accomplished; that is, information is stored on the information band in what is called Cardatron code. (Because the programmer can never have direct access to the information in the buffer except by transferring it to the Data Processor, in which case it will appear in Data Processor code, we have chosen not to specify the Cardatron code.)

The conversion of information from parallel presentation (card code) to serial presentation (Cardatron code) takes place when the contents of the input Core Shift Register are transferred to the information band. The input Core Shift Register is comprised of 80 magnetic cores which are connected to the 80 TO CARDATRON hubs on the control panel of the card reader. When a card is read the card reader scans the card row by row, sensing all punches in one row of 80 columns simultaneously. The punches appearing under the read brushes complete circuits to the cores of the Core Shift Register, thus setting them. If, for example, a punch occurs in column 72 of the card being read, the core

THE CARDATRON SYSTEM

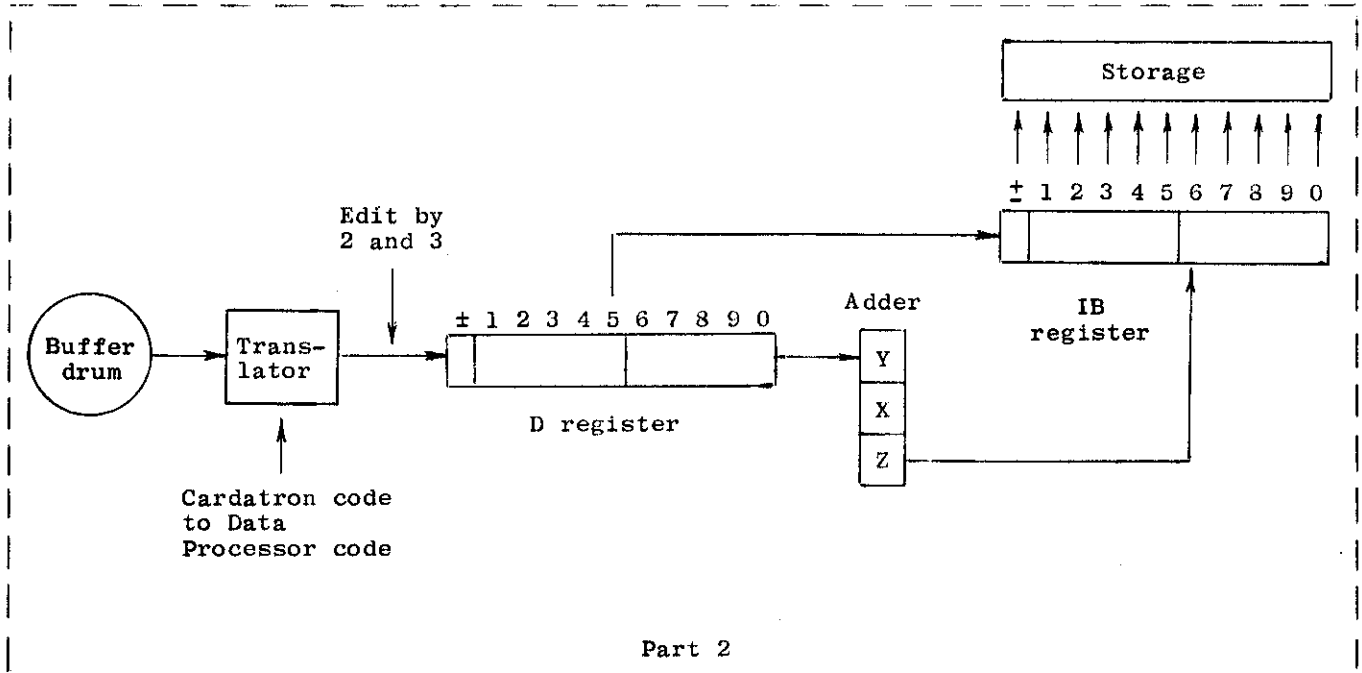
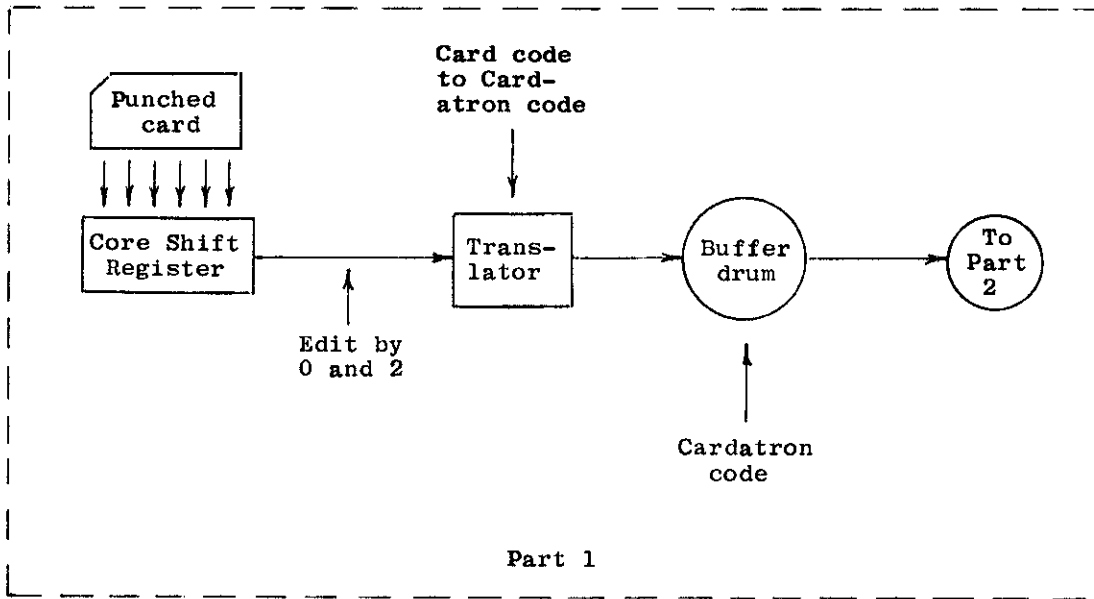


Figure VI-Intro-2. Input Information Path

corresponding to column 72 will be set. Thus, the Core Shift Register duplicates the configuration of punches in the given card row.

While the card reader is preparing to read the next row, the contents of the Core Shift Register are transferred to the information band. The Core Shift Register shifts 80 positions to the right in synchronism with the rotation of the buffer drum. The information that was in the Core Shift Register is recorded on the information band in the positions specified by the contents of the format band which was selected by the card whose contents are being transferred.

The translation from card code to Cardatron code occurs as the information passes from the Core Shift Register to the information band. The translation function is controlled by a binary-coded decade called the Row Counter (it is located in the Input Unit). The Row Counter gives a numeric value to the set positions of the Core Shift Register in the following way: as each successive row of the card is read, the Row Counter counts in synchronism with the digit emitter of the card machine to indicate which row is being read. By sensing the row being read, the Row Counter is able to establish the value of information in that row. For example, when the 4-row of a card is read the contents of the Core Shift Register specify the relative locations of all punches in the row, and the Row Counter indicates that the value of the set positions is 4.

PART 2: FROM BUFFER TO DATA PROCESSOR (See Figure VI-Intro-2.)

The execution of a CARD READ instruction causes the contents of the information band to be transferred to a set of consecutively-addressed locations in core storage. As the transfer of information is accomplished, the final editing--for example, the deletion of digits--and conversion to Data Processor code is completed. As soon as the transfer of information has been completed, the Input Unit automatically attempts to read the next card, unless the CARD READ instruction which was just executed directs it not to do so by imposing Reload Lockout.

If an attempt is made to execute a CARD READ instruction before the transfer of information from card to buffer has been completed, the Data Processor will wait until that transfer is completed; the execution of the CARD READ instruction will then proceed. The status of an Input Unit may be ascertained by executing a CARD READ INTERROGATE, BRANCH instruction.

DESIGNATION OF FORMAT BANDS

Format bands are the means by which it is determined what to do with the contents of each card that appears at the reading station of the card-handling machine. Five of the format bands are under program control; that is, the entire contents of a designated format band may be changed by executing a CARD READ, FORMAT LOAD instruction. (The execution of such an instruction transfers to a designated format band from core storage a specified set of 315 editing control stream digits.) Each of these five format bands is iden-

tified uniquely by one of the digits 1, 2, 3, 4, 5.

In addition to the five format bands mentioned above, there are three special format bands. Their identification and purpose are described below:

FORMAT BAND 6: NUMERIC FORMAT

The selection of format band 6 causes the information being transferred to be regarded as if it came from a card having the following unique format:

1. All overpunches are deleted.
2. The card has exactly eight fields, namely, those card columns wired to the following sets of TO CARDATRON hubs: 1 through 3, 4 through 14, 15 through 25, 26 through 36, 37 through 47, 48 through 58, 59 through 69, and 70 through 80.

The numeric part of the columns in each of these fields, with the exception of the first, is translated into an 11-digit Data Processor word, the sign coming from the lowest numbered TO CARDATRON hub. To the numeric part of the three-column field wired to TO CARDATRON hubs 1 through 3 are added eight high-order 0's to make a complete Data Processor word. An additional five words of zeros are supplied automatically; they are supplied as if they came from fields located to the left of column 1 and were wired to (the non-existent) TO CARDATRON hubs preceding hub 1.

FORMAT BAND 7: "REJECT" FORMAT

The selection of format band 7 causes that card to be rejected. That is, the next card automatically is fed into the reading station, unless Format Lockout is also called out by this card (by selecting also format band 8).

Although the selection of format band 7 causes a card to be rejected, its contents are transferred to the information band (the contents are edited as if numeric format had been selected). But the selection of format band 7 sets the Row Counter so that Part 2 of the information transfer is bypassed, thus allowing the next card to be fed.

FORMAT BAND 8: "FORMAT LOCKOUT"

The selection of format band 8 inhibits the reading of the card following the card which selected format band 8, that is, the contents of the following card are not transferred to the buffer until Format Lockout is removed. Format Lockout is removed by the CARD READ instruction which first refers to the Input Unit following the imposition of Format Lockout.

Format band 8 must be selected in conjunction with some one of the other format bands. If only format band 8 is selected by a card, a no-format ALARM STOP will occur.

FORMAT BAND SELECTION

The desired format band(s) usually is (are) selected by a punch (punches) in the card being read. The chosen punches

are wired to the appropriate hubs on the control panel of the card-handling machine.

There are eight FORMAT BAND SELECT hubs on the control panels of the IBM 087, 089 and 523 (see Figures VI-Intro-3, VI-Intro-4 and VI-Intro-5, respectively). Except as noted below, any punch in a card can be used to select a format band.

When a collator is being used as a card reader, the FORMAT BAND SELECT hubs are wired from primary feed. The selection of format bands by cards in the secondary feed is accomplished by wiring to one or more of four SEC(ONDARY) FORMAT SELECT (SFS) hubs. These four hubs comprise a binary-coded decade. A selected set of up to four card columns is wired to provide pulses to the SFS hubs, thus setting the FORMAT SELECT toggles in the Input Unit.

Only 9-punches can be used for the selection of a format band by cards in the secondary feed of a collator. Suppose, for example, that columns 1 through 4 are reserved for this purpose; suppose also that column 1 is wired to SFS 1, column 2 to SFS 2, column 3 to SFS 4 and column 4 to SFS 8. Then, if columns 1 and 3 contain 9-punches, format band 5 will be selected; if columns 1, 2 and 4 contain 9-punches, format band 3 and format band 8 (Format Lockout) will be selected.

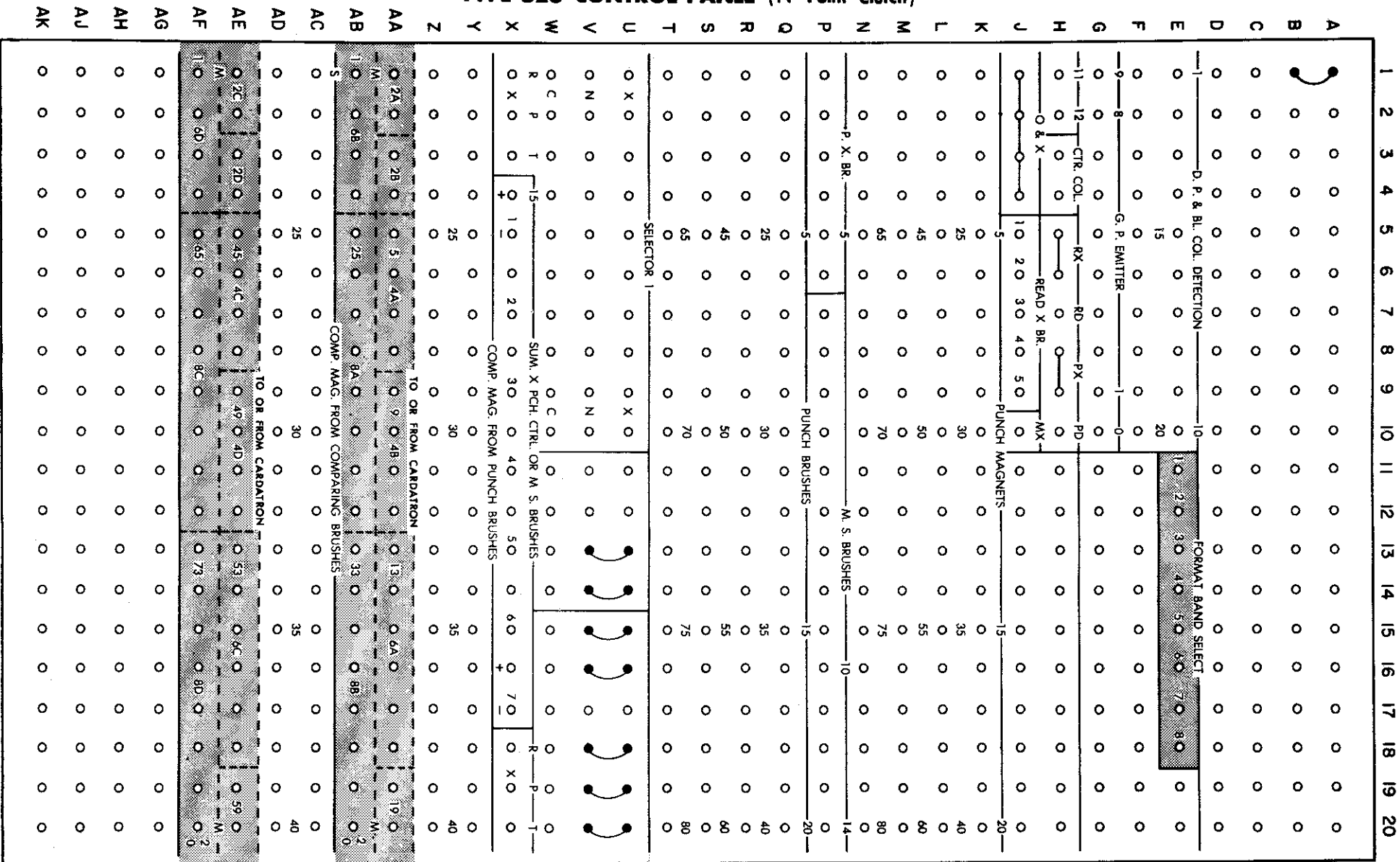
An alternative method of selecting format from secondary feed is the following: if only cards with one kind of format will be in the secondary feed, the SEC. FORMAT SELECT hubs may be jack-plugged. The control panel is then wired to provide impulses

TYPE 087 CONTROL PANEL

(Not available at time of printing.)

BURROUGHS CARDATRON

TYPE 523 CONTROL PANEL (14-Point Clutch) *



VI-Intro-13

NAME _____ DEPT. _____

USE _____ NO. _____

*PORTIONS OF THIS CONTROL PANEL DIAGRAM REPRINTED BY PERMISSION OF INTERNATIONAL BUSINESS MACHINES CORPORATION

Figure VI-Intro-5

whenever a card is fed from secondary feed.

Although 9-punches must be used to select format bands by cards in the secondary feed of a collator, they cannot be used to select format bands by cards in the primary feed; nor can 9-punches be used to select format bands if the 523 is the card reader: the pulse emitted at 9-time is used to set the cores in the Core Shift Register. If 9-punches are used, it will appear that no selection of format band was made, and a no-format ALARM STOP will occur.

EDITING CONTROL STREAM DIGITS

The four digits used to edit information and the function of each digit are described below:

- 0: Insert a 0 in the corresponding digit position in the information band.
- 1: Transfer the numeric or zone punch which corresponds to this digit position.

The distinction between numeric and zone times is made by noting those digit positions in the format band which contain 1's and 3's. Counting from format band digit position 1, the first 1 or 3 sensed edits the numeric part of the card column which corresponds to TO CARDATRON hub 80; the next 1 or 3 edits the zone part of the same card column. The next 1 or 3 edits the numeric part of the card column which corresponds to TO CARDATRON hub 79. And so forth.

- 2: Insert a 2 in the corresponding digit position in the information band.

This editing control stream digit permits the insertion--in the sign-digit position of a word--of the flag which identifies it as an alphanumeric word.

- 3: Delete the numeric or zone punch which corresponds to this digit position.

It is possible to use an overpunch for the sign of a numeric field. However, care must be exercised in the construction of editing control streams to insure that the overpunch is translated and transferred at the digit time which corresponds to sign time in the Data Processor. This is easily accomplished by counting the number of digits transferred to the Data Processor: sign time corresponds to the transfer of every eleventh digit from the information band to the Data Processor.

THE CONSTRUCTION OF EDITING CONTROL STREAMS FOR INPUT

The preparation for use of an input editing control stream can be considered to take place in five stages:

1. Determination of the editing requirements.
This is accomplished when the card format is known and after processing requirements have been stated.

2. Construction of the editing control stream to meet the editing requirements. The use of a CARDATRON FORMAT BAND CODING FORM (see Figure VI-Intro-6) facilitates their construction measurably.
3. Preparation of the input medium which is to contain the editing control stream.
4. Reading the editing control stream into core storage.
5. Writing the editing control stream on the desired format band.

This section and the one which follows will describe these processes in the context of an illustrated example.

Suppose it is desired to edit a card the format of which is shown at the top of Figure VI-Intro-6.* Processing requirements demand that the contents of the card be stored in locations 2001 through 2011 (location 2011 is to be cleared) as shown in the section of the CODING FORM labeled STORAGE DISPLAY. (Scaling is indicated by inserted 0's.)

The contents of each location and the corresponding card field from which the data are taken are tabulated below:

*The letter codes over the column designations (shown at the top of the CODING FORM) have the following meaning: D means "delete this column"; N means "this column is numeric"; and A means "this column is alphabetic." A glossary of abbreviations appears on the reverse side of the FORMAT BAND CODING FORM.

ELECTRODATA DIVISION

TITLE: Example: input

Card format

#1	#2	#3	#4	#5	#6	#7
Contract to one word. Sign in column 2.	Delete	Sign over col. 33	Sign over col. 37	Sign over col. 42	No alpha. flag	Set alpha. flag
D N N N N D N N N D D D D D N N N		A A N N N N N N N N	N N N N N N N N	N N N N A N A N	A A A A A A A A	A A A A A N
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26 27 28	29 30 31 32 33 34 35 36 37 38 39 40 41	42 43 44 45 46 47 48 49	50 51 52 53 54 55 56 57 58 59 60		

Card column

PAPER-TAPE PREPARATION											
#	±	1	2	3	4	5	6	7	8	9	0
1				316	315	314	313	312	311	310	309
		0									0
2	3	308	307	306	305	304	303	302	301	300	299
											3
3	3	297	296	295	294	293	292	291	290	289	288
											3
4	3	286	285	284	283	282	281	280	279	278	277
											3
5	3	275	274	273	272	271	270	269	268	267	266
											3
6	3	264	263	262	261	260	259	258	257	256	255
											3
7	3	253	252	251	250	249	248	247	246	245	244
											3
8	3	242	241	240	239	238	237	236	235	234	233
											3
9	3	231	230	229	228	227	226	225	224	223	222
											3
10	3	220	219	218	217	216	215	214	213	212	211
											3
11	3	209	208	207	206	205	204	203	202	201	200
											3 3 3
12	3	198	197	196	195	194	193	192	191	190	189
											3 1 3 1 3 1 3 1 3
13	3	187	186	185	184	183	182	181	180	179	178
											3 3 3 3 3
14	3	176	175	174	173	172	171	170	169	168	167
											3 1 3 1 3
15	1	165	164	163	162	161	160	159	158	157	156
											3
16	3	154	153	152	151	150	149	148	147	146	145
											3 0
17	1	143	142	141	140	139	138	137	136	135	134
											3 1 3
18	1	132	131	130	129	128	127	126	125	124	123
											0
19	3	121	120	119	118	117	116	115	114	113	112
											0 0 0 0 1
20	3	110	109	108	107	106	105	104	103	102	101
											1 0 1
21	1	99	98	97	96	95	94	93	92	91	90
											1 1 1 1 1 1 1
22	3	88	87	86	85	84	83	82	81	80	79
											1
23	1	77	76	75	74	73	72	71	70	69	68
											1
24	1	66	65	64	63	62	61	60	59	58	57
											3 1 3 1 3 1 3
25	1	55	54	53	52	51	50	49	48	47	46
											3 1 3 1 3 1 3 1 3
26	3	44	43	42	41	40	39	38	37	36	35
											0
27	0	33	32	31	30	29	28	27	26	25	24
											3 1 3 1 3 1 3
28	1	22	21	20	19	18	17	16	15	14	13
											3 1 3 1 3 1 3 1
29	0	11	10	9	8	7	6	5	4	3	2
											0

PUNCHED-CARD PREPARATION																																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33		
				0	2	0	0	0	6	0	1	1	2	9																				
					</																													

BAND NO: _____ TITLE: _____
 UNIT NO: _____ ROUTINE: _____
 INPUT: _____ CODER: _____
 OUTPUT: _____ DATE: _____

REMARKS: _____

INPUT:

FORMAT BAND DIGITS	EDITING FUNCTION	CORE REGISTER SHIFTS	INFORMATION TRANSFER	D. IB REGISTERS SHIFT
0	INSERT 0	NO	NO	YES
1	TRANSFER DIGIT	YES	YES	YES
2	INSERT 2	NO	NO	YES
3	DELETE DIGIT	YES	NO	NO

TYPE OF CARD COLUMN	ABBR	FORMAT BAND DIGITS	EDITED RESULTS IN DATATRON WORD
ALPHABETIC OR SPECIAL CHARACTER	A	11	TRANSLATED INTO TWO-DIGIT DATATRON CODE.
	D	33	CHARACTER DELETED.
NUMERIC	N	31	TRANSLATED INTO SINGLE DATATRON DIGIT.
	A	11	TRANSLATED INTO TWO-DIGIT DATATRON CODE; DIGIT PRECEDED BY INSERTED 8.
	D	33	DIGIT DELETED.
BLANK	A	11	TRANSLATED AS 00.
	D	33	COLUMN DELETED.
NONE		0	0 INSERTED; NO EFFECT ON CARD CONTENTS.
		2	2 INSERTED; NO EFFECT ON CARD CONTENTS.

NOTES:

1. Always start the active segment of the editing control stream in digit position 1 on the format band.
2. The inactive segment follows the active segment and contains 3's, except as noted below.
3. Seven 0's are needed following the active segment to shift the last word out of the D register and into storage. These 0's may appear anywhere in the inactive segment (usually they should occupy digit positions 309 through 315).

OUTPUT:

FORMAT BAND DIGITS	EDITING FUNCTION	D. IB REGISTERS SHIFT	INFORMATION TRANSFER	CORE REGISTER SHIFTS
0	INSERT BLANK	NO	NO	YES
1	TRANSFER ALPHA	YES	YES	YES
2	TRANSFER NUMERIC	YES	YES	YES
3	DELETE DIGIT	YES	YES	NO

TYPE OF OUTPUT	ABBR	FORMAT BAND DIGITS	PUNCHED OR PRINTED RESULTS
ALPHABETIC OR SPECIAL CHARACTER	A	11	TWO-DIGIT DATATRON CODE TRANSLATED AS ALPHABETIC OR SPECIAL CHARACTER.
		33	TWO DATATRON DIGITS DELETED.
NUMERIC	N	2	ONE DIGIT TRANSLATED NUMERICALLY.
	SN	01	1 THROUGH 9 TRANSLATED NUMERICALLY; 0 TRANSLATED AS BLANK.
	SZ	10	OVERPUNCH SIGN WITH NO DIGIT UNDERPUNCH.
	A	11	OVERPUNCH SIGN WITH DIGIT UNDERPUNCH.
	D	3	ONE DATATRON DIGIT DELETED.
BLANK	B	00	BLANK COLUMN PRODUCED.

NOTES:

1. Always start the active segment of the editing control stream with digit position 1 corresponding to the numeric part of column 120. Always write a format band as if the output contained 120 printing or punching positions.
2. The inactive segment follows the active segment and contains 3's.
3. The editing control stream digit pair 01 does not always produce the same effect as the digit 2.

Figure VI-Intro-6

THE CARDATRON SYSTEM

Card Columns	Data Processor Location
1	(Deleted)
2 - 18	2001
19 - 28	(Deleted)
29 - 32	2002
33 - 36	2003
37 - 41	2004
42 - 49	2005
50 - 54	2006
55 - 59	2007
60 - 70	2008
(Inserted 0's)	2009
71 - 80	2010
(Inserted 0's)	2011

It is assumed that 80-80 control panel wiring is used, that is, wiring is from PRIMARY READ or SECONDARY READ hub k to TO CARD-ATRON hub k (a collator is being used as a card reader). In the description which follows, references are to the section of the FORMAT BAND CODING FORM labeled PUNCHED-CARD PREPARATION.

(A word of caution is in order here: this example would be more readily understood if the reader knew both how the format band was loaded and how the contents of the information band were transferred to core storage; in turn, those mechanisms are easier to understand if one knows how editing control streams are constructed. It is recommended, therefore, that the reader accept on faith the contents of this section during its first reading; after reading the narrative description of CARD READ and CARD READ, FORMAT LOAD which follow, a second reading--if it is necessary--will make this description of the construction of editing control streams much more palatable.)

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

#10										
Sign over col. 71										
N	N	N	N	N	N	N	N	N	N	N
63	64	65	66	67	68	69	70	71	72	73
74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95

45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80		
#10 27											28											#11						29									
37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
0	0	0	0	0	0	1	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	0	0	0	0	0	0	0	0	0	0	0	0

S T O R A G E													D I	
LOC.	#	1	2	3	4	5	6	7	8	9	0	REMARKS	LOC.	
2010	10	±	N	N	N	N	N	N	N	N	N			
2011	11	0	0	0	0	0	0	0	0	0	0	Insert 0's		

The first 11 digit positions on the format band--format band digit positions 1 through 11--are coded with 0's. These 11 0's yield the word of 0's which is the content of location 2011.

The contents of column 80 are then translated. A 1 in digit position 12 of the format band translates the numeric part of the column numerically, as specified. When the contents of the information band are transferred to core storage, this digit will occupy 2010:01, the low-order digit position of the word in location 2010. A 3 in the next digit position of the format band, digit position 13, deletes the zone part of column 80.

Next, the contents of column 79 are translated. A 1 in digit

position 14 of the format band translates the numeric part of the column. When the contents of the information band are transferred to core storage, this digit will occupy 2010:91. A 3 in digit position 15 deletes the zone part of column 79.

In similar fashion, the contents of columns 78, 77, ..., 72 are translated, each by the digit pair 31: the contents of column 78 are translated by the digit pair in format band digit positions 17 and 16; the contents of column 77 are translated by the digit pair in positions 19 and 18; and so forth. The digits will occupy 2010:81, 2010:71, ..., 2010:21, respectively, when the contents of the information band are transferred to core storage.

The numeric part of column 71 is translated by the digit 1 in format band digit position 30; this digit will occupy 2010:11. The zone part of column 71, which contains the overpunched sign of the numeric field, is translated by the 1 in format band digit position 31. Because this translated zone punch is transferred to the Data Processor at sign time--the translated digit will occupy the sign-digit position of the word in location 2010--it is recognized uniquely as a sign overpunch and translated as 0 or 1 (+ or -, respectively) in accordance with the usual conventions for overpunching signs.*

The 0's in format band digit positions 32 through 42 yield the 11 0's which are the contents of location 2009 (see illustration on next page).

*An X punch for -, a Y punch or no overpunch for +.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

														#8																								
														Sign in col. 60																								
														AAANNNNNNNNNNNNNNNNNNN																								
71	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75																				

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	
											#8 24					25										#9 26					#10							
68	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30		
						3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0

S T O R A G E																			D I	
LOC.	#	±	1	2	3	4	5	6	7	8	9	0	REMARKS						LOC.	
2008	8	±	N	N	N	N	N	N	N	N	N	N								
2009	9	0	0	0	0	0	0	0	0	0	0	0	Insert 0's							

Next there are 11 digit pairs, 31, to translate numerically the 11-column field in columns 60 through 70. These 11 translated digits will form the word in location 2008. Note that the sign appears as a numeric punch in column 60.

											#7																								
											Set alpha. flag																								
											AAAAAAAAAANN																								
51	52	53	54	55	56	57	58	59	60	61																									

											#7 23																		
80	79	78	77	76	75	74	73	72	71	70	69	68	67																
					2	1	1	1	1	1	1	1	1																

13	14	15	16	17	18	19	20	21																						
								#8 24																						
									66	65	64	63	62	61	60	5														
									1	1																				

THE CARDATRON SYSTEM

S T O R A G E											D I			
LOC.	5	4	1	2	3	4	5	6	7	8	9	0	REMARKS	LOC.
2	0	7	7	2	A	A	A	A	A	A	A	A	Insert 2	

The next field to be translated is designated as alpha-numeric. The digit pair 11 in the format band is used to translate the contents of a column into the two-digit Data Processor code (see Appendix A3 for this code).* In particular, the digit pair in format band digit positions 66 and 65 translates column 59; the two-digit Data Processor code will occupy 2007:02. The digit pair in positions 68 and 67 translates column 58; the two-digit Data Processor code will occupy 2007:82. Similarly, the contents of column 57 will be translated into the two-digit code occupying 2007:62; the contents of column 56 will be translated into the two-digit code occupying 2007:42; and the contents of column 55 will be translated into the two-digit code occupying 2007:22. The digit 2 in format band digit position 75 causes a 2 to be inserted into 2007:±1, the sign-digit position of the word in location 2007.

Suppose, for example, that column 59 is punched numerically with a 4 (that is, there is no overpunch in this column). Then the 1 in format band digit position 65 would translate the digit punch numerically, and the translated digit--a 4--would occupy

*Note that in the STORAGE DISPLAY section of the CODING FORM two Data Processor digit positions--both designated as A--correspond to one card column which is also designated as A.

2007:01. The 1 in format band digit position 66, translating the zone part of the column, would recognize both no overpunch as well as the fact that the zone part is not being transferred at Data Processor sign time. Hence an 8 would be inserted in 2007:91. The two-digit (alphanumeric) Data Processor code for 4 is 84; and this is what would appear in 2007:02.

Note the quite different effect produced by the editing digit 1 when it is used to translate the zone part of a column at sign time. That effect is described in the paragraph above, relating to the transfer of the contents of column 71 into 2010:±1.

The reader now ought to be able to verify the coding which translates the designated card fields into words 6 through 2, occupying core storage locations 2006 through 2002, respectively. Note especially the insertion of scaling 0's in words 6, 4, 3 and 2. (Please refer to Figure VI-Intro-6.)

Columns 28 through 19 are deleted by the ten digit pairs, 33, occupying format band digit positions 145 through 164.

Again, it is left as an exercise for the reader to verify that the coding which translates columns 18 through 1 into the word occupying location 2001 is the set of digits in format band digit positions 165 through 200 in Figure VI-Intro-6.

The part of the format band occupying format band digit positions 1 through 200 is called the active segment of the format band; it is always comprised of those format band digits which lie between and

include format band digit position 1 and either:

- a. the format band digit in the position which refers to the last digit transferred to core storage if that is an inserted digit; or
- b. the format band digit in the position which refers to the zone part of the column wired to TO CARDATRON hub 1, if no inserted digits follow that reference.

The part of the format band which follows the active segment is called the inactive segment; it always contains 3's, except for seven 0's, which are required to shift the last word transferred from the information band into core storage. It is customary to have these 0's in format band digit positions 309 through 315. The contents of the remainder of the format band word designated 1 are irrelevant, that is, may be coded arbitrarily since they are not written on the format band, there being no room for them.

In the lower right corner of the obverse side of the FORMAT BAND CODING FORM (see also the next page) will be found a form for checking the digit count in the active segment of an input format band. This check does not guarantee that the encoding is without error, but it does help in detecting clerical errors when constructing editing control streams.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

DIGIT CHECK FOR ACTIVE SEGMENT

INPUT	OUTPUT
NUMBER OF 0's = <u>39</u>	NUMBER OF 0's = _____
NUMBER OF 1's = <u>81</u>	NUMBER OF 1's = _____
NUMBER OF 2's = <u>1</u>	NUMBER OF 2's = _____
NUMBER OF 3's = <u>79</u>	NUMBER OF 3's = _____
SUM* = 200	SUM* = _____
$\frac{0's + 1's + 2's}{11} = \text{WORDS PER CARD}$	$\frac{0's + 1's + 2 \times 2's}{2} = \text{COLUMNS PRINTED OR PUNCHED}$
$\frac{39 + 81 + 1}{11} = 11$	_____ = _____
$\frac{1's + 3's}{2} = \text{CARD COLUMNS}$	$\frac{1's + 2's + 3's}{11} = \text{DATATRON WORDS}$
$\frac{81 + 79}{2} = 80$	_____ = _____

*If the active segment starts in format band digit position 1, the sum of 0's, 1's, 2's, and 3's should be identical with the format band digit position containing the last digit of the active segment.

Note that the **FORMAT BAND CODING FORM** has provisions for laying out editing control streams for punching in paper tape or cards. In the section labeled **PUNCHED-CARD PREPARATION**, columns 4 through 14 of each card are reserved for a bootstrap instruction if the cards containing the editing control stream are not being loaded under program control;* the bootstrap instruction will read the next card. The set of instructions shown in Figure VI-Intro-6 will cause the 29 words of the editing control stream shown to be stored in locations 1101 through 1129; word 29 will be in location 1129; word 28 will be in location 1128; and so forth. Each set of 11 contiguous format digits comprises one word whose identification

*From this remark the reader will infer--correctly--that input sign control is effective. The subject is discussed in detail on page VI-60-3, ff.

number is as shown on the FORMAT BAND CODING FORM.

The instruction 6 Op aaaa in the fifth card is used as a bootstrap to continue. It may be, for example, another CARD READ instruction if there are additional cards to be read, or it may be a BRANCH UNCONDITIONALLY instruction if it is desired to turn over control to an already-loaded program.

Column 1 is used for selecting numeric format.

The section of the CODING FORM labeled PAPER-TAPE PREPARATION permits the editing control stream to be laid out for paper-tape input. The numbering of the words in this section corresponds to the numbering in the section labeled PUNCHED-CARD PREPARATION. When preparing paper tape, one starts punching with word number 1 because the PRD or PRB instruction loads core storage in order of sequentially ascending addresses; as will be seen below, information from cards is loaded in order of sequentially descending addresses.

INSTRUCTIONS

CARD READ (CRD) is an instruction that transfers the contents of the information band on a designated Input Unit to core storage. The format band which is used to edit this information is the one selected by the card whose contents are on the information band. Up to 28 words are transferred to sequentially-addressed storage locations. The first word transferred is stored in the location specified in the address part of the CARD READ instruction. The remaining words are

stored in order of sequentially descending addresses.

As shown in Figure VI-Intro-6, the contents of the information band were transferred to locations 2011 through 2001 by a CARD READ instruction with 2011 as its address.

CARD READ, FORMAT LOAD (CRF) is an instruction that provides the ability to load the editing control stream digits onto the buffer drum of an Input Unit. It transfers these digits--stored in 29 consecutively-addressed core storage locations--to the designated format band of a designated Input Unit. The contents of the location specified by the address part of the CARD READ, FORMAT LOAD instruction will occupy the first 11 digit positions on the format band. Suppose the instruction were CRF 1129. Then (1129:01) goes to format band digit position 1; (1129:02) goes to format band digit position 2; ...; and (1129:11) goes to format band digit position 11. The word in location 1128 will occupy format band digit positions 12 through 22; and so forth. Finally, (1101:08) will occupy the last eight digit positions on the format band.

As shown in Figure VI-Intro-6, the editing control stream digits were stored in locations 1101 through 1129 by the bootstrap instructions in columns 4 through 14. A CRF 1129 instruction transferred 315 of these digits to the desired format band as described above.

(The reader will note in the following sections on output that the CARD READ, FORMAT LOAD instruction performs the same type

of function on input that the CARD WRITE and CARD WRITE, FORMAT LOAD instructions do on output: information is transferred from core storage of the Data Processor to a buffer drum.

CARD READ INTERROGATE, BRANCH (CRI) is an instruction which permits the program to determine if a designated Input Unit is ready to be used. If the unit queried is ready, control is transferred, that is, the contents of the P register are replaced by the address part of the CRI instruction; otherwise, control continues in sequence.

OUTPUT

THE OUTPUT UNIT

The Output Unit, like the Input Unit, is made up of two essential parts, control circuitry and a magnetic drum (the buffer). The drum has on it six bands for storing information. One band--called the information band--is used to store information transferred from the Data Processor to the card-handling machine. The remaining five bands--called format bands--are used to contain editing control streams. Each of the six bands on the buffer has provision for holding 316 digits (whereas there are only 315 digit positions on the bands of an Input Unit).

Corresponding to each digit position in the information band is a digit position in each of the five format bands. Each of these digit positions is identified uniquely and in

order by one of the numbers from 1 to 316. Digit position 1 is regarded as the origin of a band. Whereas each digit position in the information band consists of a full four-bit decade--and hence is capable of storing any one of the decimal digits--each digit position in a format band is comprised of only two positions: it is possible to write only digits 0, 1, 2 and 3 in a format band. As was the case with an input format band, the four bit or the eight bit will be ignored if an attempt is made to write as a format band digit a decimal digit greater than or equal to four.

INFORMATION TRANSFER

The transfer of information from the Data Processor to a card-handling machine takes place in two parts. During the first part, information is transferred from the Data Processor to the information band when a CARD WRITE instruction is executed. During the second part, the contents of the information band are transferred to a card-handling machine. This is accomplished under control of an Output Unit, independently of the Data Processor. After information has been transferred from the Data Processor to the information band, execution of the program is resumed automatically.

PART 1: FROM DATA PROCESSOR TO BUFFER (See Figure VI-Intro-7.)

Each CARD WRITE instruction specifies the format band which is to edit the stream of information to be transferred from the Data Processor.

Up to 316 digits from 29 words can be transferred from sequentially-addressed storage locations of the Data Processor. The first digit transferred is taken from the storage location specified by the address part of the CARD WRITE instruction. The remaining digits are transferred from core storage locations in order of sequentially descending addresses.

The editing control stream on the format band selected by the CARD WRITE instruction performs a part of the editing process when the information is transferred to the information band: 0's are inserted. In addition, part of the conversion from Data Processor code to card code is accomplished; that is, information is stored in the information band in Cardatron code.

If an attempt is made to execute another CARD WRITE instruction before the transfer of information from the buffer to the card-handling machine has been completed--that is, before completion of Part 2 of the information transfer associated with the preceding CARD WRITE instruction--the Data Processor will wait until the Output Unit is in a state of readiness. The status of an Output Unit may be ascertained by executing a CARD WRITE INTERROGATE, BRANCH instruction.

PART 2: FROM BUFFER TO CARD-HANDLING MACHINE (See Figure VI-Intro-7.)

After the information band has been loaded by the execution of a CARD WRITE instruction, its contents are transferred to the

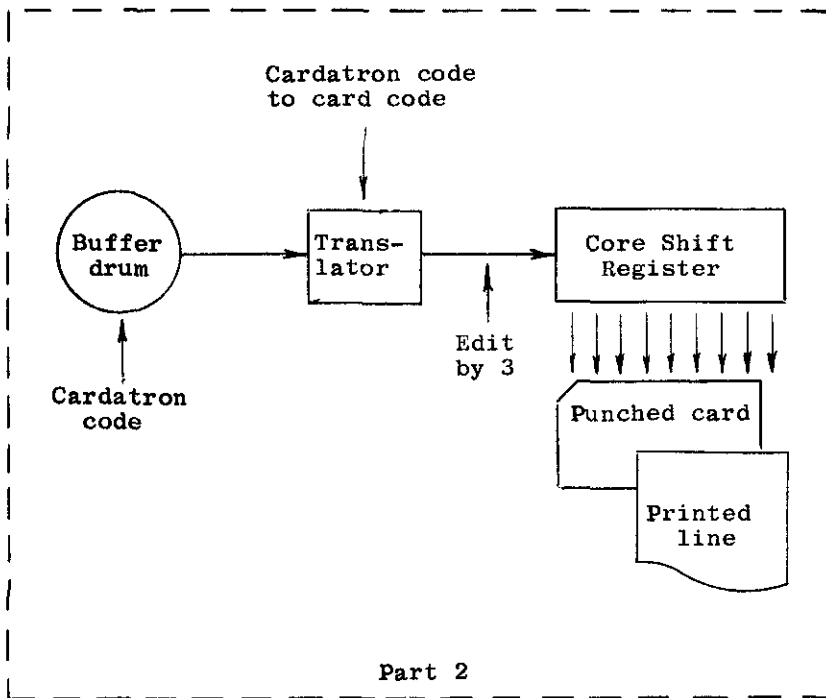
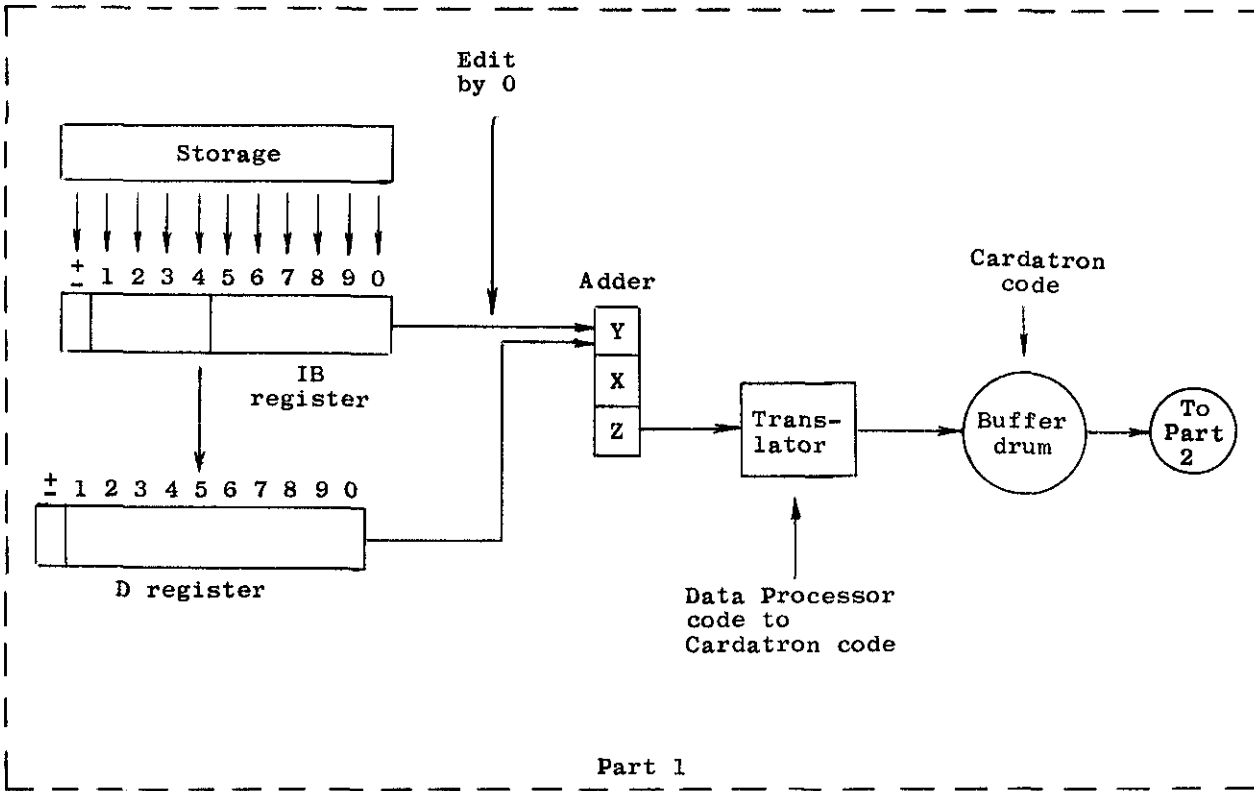


Figure VI-Intro-7. Output Information Path

card-handling machine. As the transfer of information is accomplished, the final editing--the deletion of unwanted digits and conversion to card code--is completed.

The purpose of Part 2 of the output cycle is to present information from the buffer to the card-handling machine in the proper sequence for printing or punching. The conversion of information from serial presentation (Cardatron code) to parallel presentation (card code) takes place when the contents of the output Core Shift Register are transferred to the card-handling machine.

As each digit is transferred from the information band it is set into a decade of comparison toggles. When the toggles are set, their value is compared with the value of the Output Unit's Row Counter setting; at the same time the corresponding editing control stream digit in the format band is examined. Three cases need to be considered:

1. Each time there is agreement between the Row Counter and the comparison toggles the input position--the first core--of the Core Shift Register* is set and the entire

*The output Core Shift Register is comprised of 120 magnetic cores that are connected to the 120 FROM CARDATRON hubs on the control panel of a line printer; or, 80 of the cores are connected to the 80 FROM CARDATRON hubs on the control panel of a card punch. If the Output Unit is connected to a punch, cores numbered 1 through 80 in the Core Shift Register will have been cable-connected to the punch.

Core Shift Register is shifted one place to the right, except when the editing control stream digit is a 3.

2. When the editing control stream digit is a 3, the corresponding digit in the information band is deleted by the simple expedient of not setting and not shifting the Core Shift Register.
3. When the Row Counter setting and the comparison toggles do not agree, the Core Shift Register is shifted one place to the right without setting the input position, except when the editing control stream digit is a 3. In the latter case, the effect is as described in 2.

Thus, for each Row Counter setting the information band is scanned for digits corresponding in value to the Row Counter setting: the configuration of such digits in the information band is represented by the set positions in the Core Shift Register, with the exception of digits that have been deleted. At this time a digit impulse is emitted from the card-handling machine. The purpose of this impulse is to establish synchronism between the Output Unit and the card-handling machine so that information corresponding to the set positions of the Core Shift Register can be transferred to the card-handling machine. The value of this information is determined by the digit time of the card-handling machine's cycle during which this transfer occurs:

digit time and the Row Counter count together, of course.

Each position in the Core Shift Register is connected to and thereby controls--by means of control panel wiring--the energizing of a corresponding punch or print magnet in the card-handling machine. If the machine is a punch, all the positions in the card that are to be punched in a given row will be punched at the time the information in the cores is transferred.

If the card-handling machine is a printer, each of the bits of information in the Core Shift Register sets the corresponding type wheel each time the digit pulse is emitted. When all of the information has been transferred from the information band, the line printer will print one line.

DESIGNATION OF FORMAT BANDS

Each of the five format bands on the buffer of an Output Unit is identified uniquely by one of the digits from 1 to 5. No special format bands are used by the Output Unit. All of the format bands are under program control; that is, the entire contents of a designated format band may be changed by executing a CARD WRITE, FORMAT LOAD instruction.

FORMAT BAND SELECTION

The desired format band is designated by a digit in the CARD WRITE or CARD WRITE, FORMAT LOAD instruction.

EDITING CONTROL STREAM DIGITS

The four digits used to edit information and the function of each digit are described below:

- 0: Insert a 0 in the corresponding digit position in the information band.
- 1: Transfer the Data Processor digit which corresponds to this digit position with numeric or zone significance.

The distinction between numeric and zone significance is made by noting those digit positions in the format band which contain pairs of 1's, pairs of 0's, or 0's and 1's in pairs, that is, 11, 00, 01, or 10. Counting from format band digit position 1, the first 1 or 0--of the pair of digits--sensed when the Row Counter is at numeric time corresponds to a selected FROM CARDATRON hub on the control panel of the card-handling machine; the next 1 or 0 sensed when the Row Counter is at zone time corresponds to the same control panel hub. The next 1 or 0 corresponds to its selected control panel hub during numeric time; the next 1 or 0 corresponds to the same control panel hub during zone time. And so forth.

- 2: Transfer the Data Processor digit which corresponds to this digit position with numeric significance.

The digit 2 interprets the digit being transferred as a numeric digit, forcing a blank for the corresponding zone part. Thus, the first 2--counting from format band digit position 1--corresponds to the assigned control panel hub during numeric time; at zone time a blank is forced for that control panel hub. The next 2 corresponds to its assigned control panel hub during numeric time; at zone time, a blank is forced for the same control panel hub. And so forth.

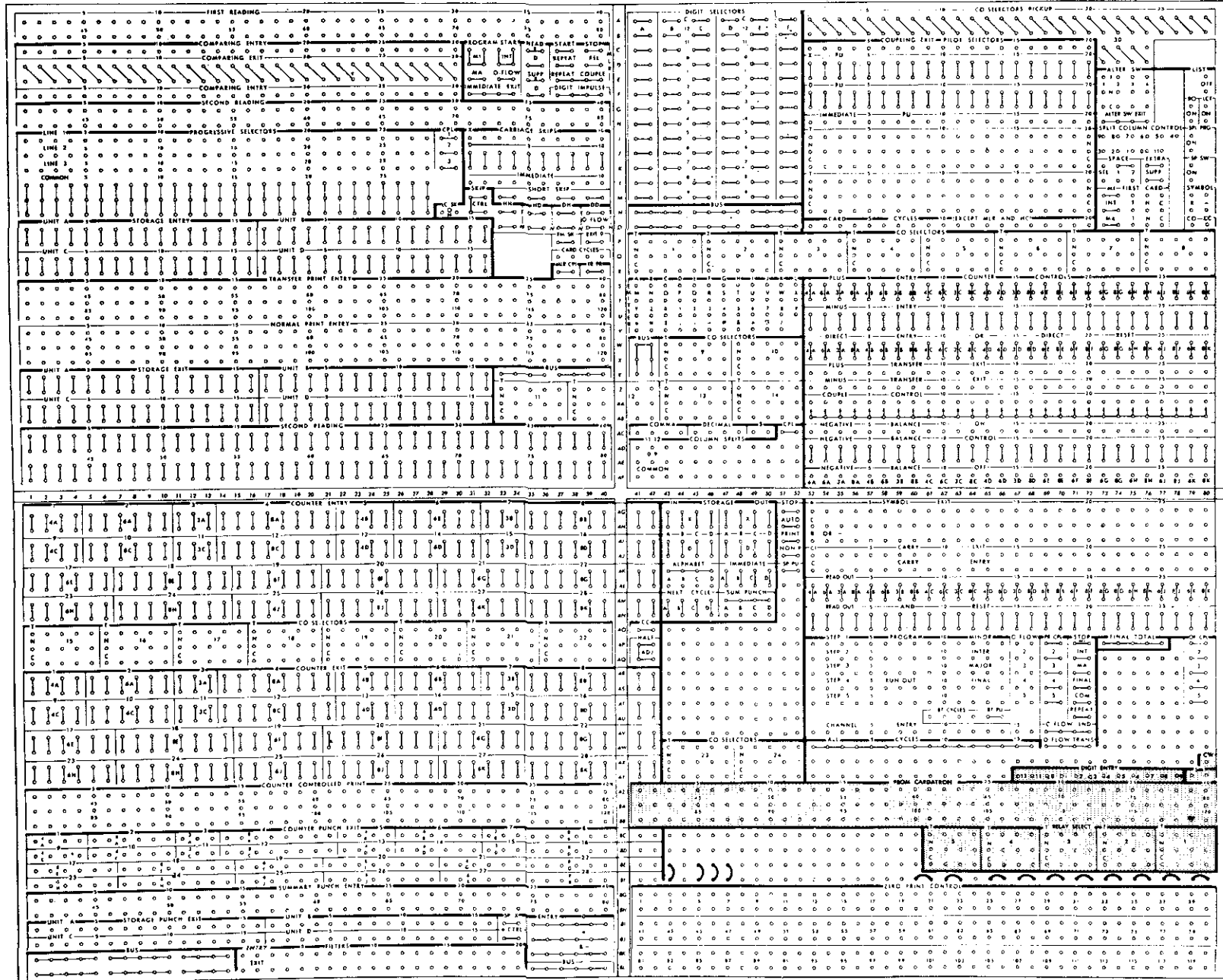
The digit 2 must occupy a position in the format band which corresponds to numeric time in the sense described above; otherwise, an invalid translation will occur.

- 3: Delete the Data Processor digit which corresponds to this digit position.

It is possible to use an overpunch for the sign of a numeric field. Either of the two conventions for overpunching signs may be selected by proper specification in the CARD WRITE instruction. However, care must be exercised in the construction of the editing control stream to insure that the overpunch is translated and transferred at the digit time which corresponds to sign time in the Data Processor. If the digit 1 that transfers the sign as an overpunch does not occur at sign time, an invalid translation will result.

BURROUGHS CARDATRON

TYPE 407 CONTROL PANEL*



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Figure VI-Intro-8A

VI-Intro-38

7/1/58

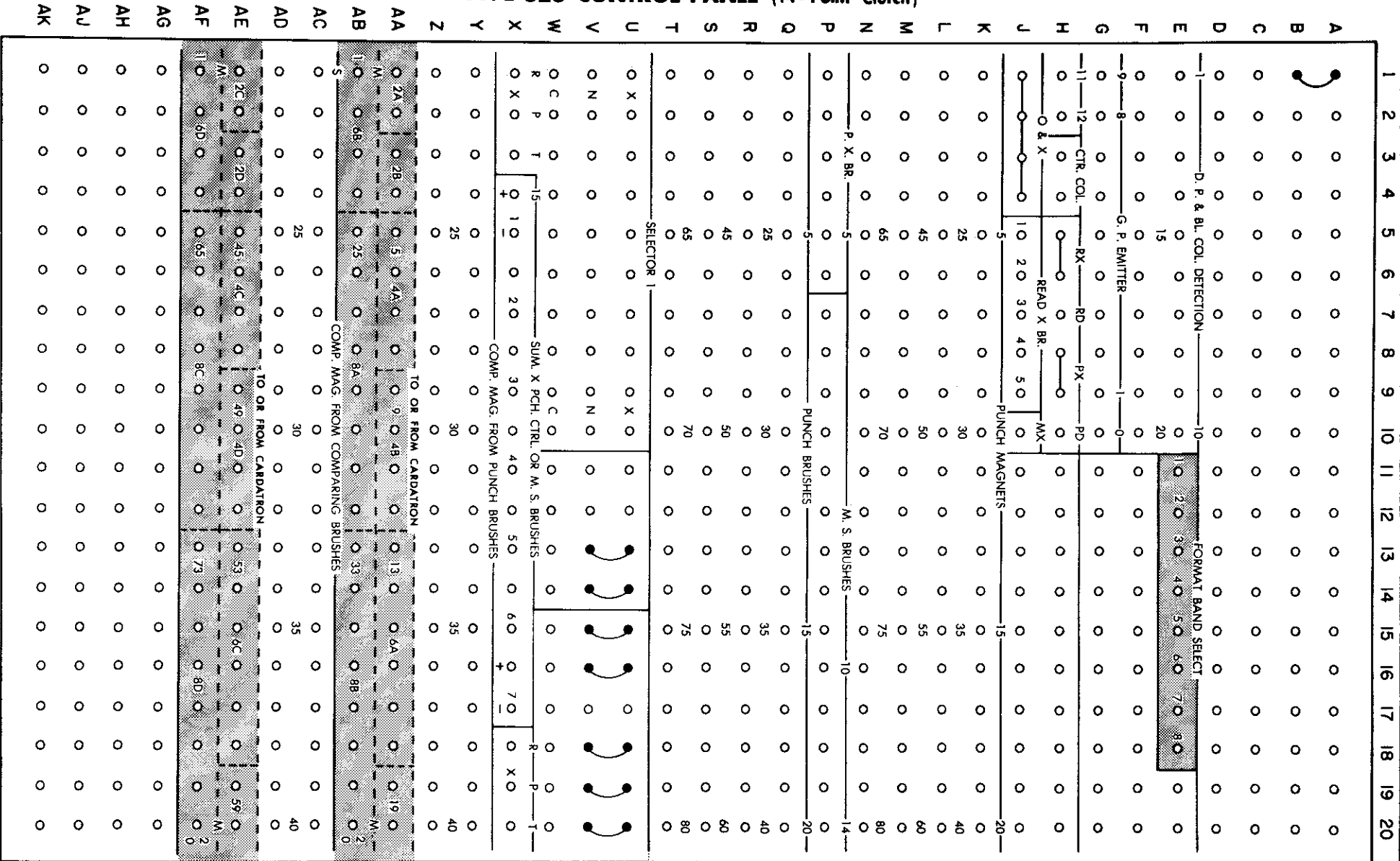
NAME _____ DEPT. _____

USE _____ NO. _____

*PORTIONS OF THIS CONTROL PANEL DIAGRAM REPRINTED BY PERMISSION OF INTERNATIONAL BUSINESS MACHINES CORPORATION

BURROUGHS CARDATRON

TYPE 523 CONTROL PANEL (14-Point Clutch) *



NAME _____ DEPT. _____

USE _____ NO. _____

*PORTIONS OF THIS CONTROL PANEL DIAGRAM REPRINTED BY PERMISSION OF INTERNATIONAL BUSINESS MACHINES CORPORATION

Figure VI-Intro-8B

THE CONSTRUCTION OF EDITING CONTROL STREAMS FOR OUTPUT

The preparation for use of an output editing control stream also takes place in five stages:

1. Determination of the editing requirements.
This is accomplished when the format of the document or card is known and after processing requirements have been stated.
2. Construction of the editing control stream to meet the editing requirements.
3. Preparation of the input medium which is to contain the output editing control stream.
4. Reading the editing control stream into core storage.
5. Writing the editing control stream on the desired format band.

This section and the one which follows will describe these processes in the context of an illustrated example.

Suppose it is desired to edit a line of print the format of which is shown at the top of Figure VI-Intro-9. Processing requirements demand that information for the line to be printed be stored in locations 2017 through 2026, as shown in the section of the FORMAT BAND CODING FORM labeled STORAGE DISPLAY.

The contents of each printing position and the corresponding Data Processor location from which the data are taken are tabulated below:

BAND NO: _____ TITLE: _____
 UNIT NO: _____ ROUTINE: _____
 INPUT: _____ CODER: _____
 OUTPUT: _____ DATE: _____

REMARKS: _____

INPUT:

FORMAT BAND DIGITS	EDITING FUNCTION	CORE REGISTER SHIFTS	INFORMATION TRANSFER	D. IB REGISTERS SHIFT
0	INSERT 0	NO	NO	YES
1	TRANSFER DIGIT	YES	YES	YES
2	INSERT 2	NO	NO	YES
3	DELETE DIGIT	YES	NO	NO

TYPE OF CARD COLUMN	ABBR	FORMAT BAND DIGITS	EDITED RESULTS IN DATATRON WORD
ALPHABETIC OR SPECIAL CHARACTER	A	11	TRANSLATED INTO TWO-DIGIT DATATRON CODE.
	D	33	CHARACTER DELETED.
NUMERIC	N	31	TRANSLATED INTO SINGLE DATATRON DIGIT.
	A	11	TRANSLATED INTO TWO-DIGIT DATATRON CODE; DIGIT PRECEDED BY INSERTED 8.
	D	33	DIGIT DELETED.
BLANK	A	11	TRANSLATED AS 00.
	D	33	COLUMN DELETED.
NONE		0	0 INSERTED; NO EFFECT ON CARD CONTENTS.
		2	2 INSERTED; NO EFFECT ON CARD CONTENTS.

NOTES:

1. Always start the active segment of the editing control stream in digit position 1 on the format band.
2. The inactive segment follows the active segment and contains 3's, except as noted below.
3. Seven 0's are needed following the active segment to shift the last word out of the D register and into storage. These 0's may appear anywhere in the inactive segment (usually they should occupy digit positions 309 through 315).

OUTPUT:

FORMAT BAND DIGITS	EDITING FUNCTION	D. IB REGISTERS SHIFT	INFORMATION TRANSFER	CORE REGISTER SHIFTS
0	INSERT BLANK	NO	NO	YES
1	TRANSFER ALPHA	YES	YES	YES
2	TRANSFER NUMERIC	YES	YES	YES
3	DELETE DIGIT	YES	YES	NO

TYPE OF OUTPUT	ABBR	FORMAT BAND DIGITS	PUNCHED OR PRINTED RESULTS
ALPHABETIC OR SPECIAL CHARACTER	A	11	TWO-DIGIT DATATRON CODE TRANSLATED AS ALPHABETIC OR SPECIAL CHARACTER.
		33	TWO DATATRON DIGITS DELETED.
NUMERIC	N	2	ONE DIGIT TRANSLATED NUMERICALLY.
	SN	01	1 THROUGH 9 TRANSLATED NUMERICALLY; 0 TRANSLATED AS BLANK.
	SZ	10	OVERPUNCH SIGN WITH NO DIGIT UNDERPUNCH.
	A	11	OVERPUNCH SIGN WITH DIGIT UNDERPUNCH.
	D	3	ONE DATATRON DIGIT DELETED.
BLANK	B	00	BLANK COLUMN PRODUCED.

NOTES:

1. Always start the active segment of the editing control stream with digit position 1 corresponding to the numeric part of column 120. Always write a format band as if the output contained 120 printing or punching positions.
2. The inactive segment follows the active segment and contains 3's.
3. The editing control stream digit pair 01 does not always produce the same effect as the digit 2.

Figure VI-Intro-9

THE CARDATRON SYSTEM

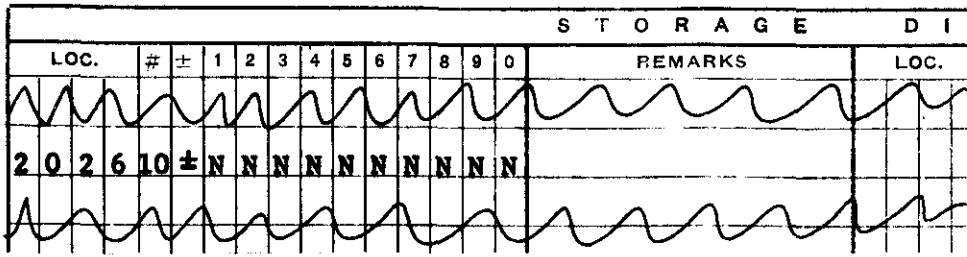
Printing Position	Data Processor Location
1 - 8	Blanks inserted by the editing control stream.
9 - 19	2017
20 - 29	Blanks inserted by the editing control stream.
(Deleted)	2018
30 - 33	2019
34 - 38	Blanks inserted by the editing control stream.
39 - 43	2020
44 - 48	Blanks inserted by the editing control stream.
49 - 54	2021
55 - 59	Blanks inserted by the editing control stream.
60 - 68	2022
69 - 73	Blanks inserted by the editing control stream.
74 - 78	2023
79 - 83	Blanks inserted by the editing control stream.
84 - 88	2024
89 - 93	Blanks inserted by the editing control stream.
94 - 104	2025
105 - 109	Blanks inserted by the editing control stream.
110 - 120	2026

It is assumed that 120-120 control panel wiring is used, that is, wiring is from FROM CARDATRON hub k to NORMAL PRINT ENTRY hub k. In the description which follows, references are to the section of the CODING FORM labeled PUNCHED-CARD PREPARATION.

		#10															
		Blank								Numeric							
										S Z							
3	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
3	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
												28				#10				29				
25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
			0	0	0	0	0	0	0	0	0	0	1	0	2	2	2	2	2	2	2	2	2	2



The first digit to be translated is the digit occupying 2026:01, the low-order digit of the word in location 2026. A 2 in digit position 1 of the format band translates this digit as a number; that is, the digit is treated as having a numeric part and a blank zone part. When the contents of the information band are transferred to the line printer, this number will occupy print position 120, the rightmost digit of the line of print.

Next, the digit occupying 2026:91 is translated. A 2 in digit position 2 of the format band translates this digit as a number. When the contents of the information band are transferred to the line printer, this number will occupy print position 119.

In similar fashion, the contents of 2026:81, 2026:71, ..., 2026:11 are each translated by a digit 2: the contents of 2026:81 are translated by the digit in format band digit position 3; the contents of 2026:71 by the digit in position 4; and so forth. The digits will occupy print positions 118, 117, ..., 111, respectively.

The pair of digits 0 and 1 in format band digit positions 11 and 12 translate the sign digit--(2026:±1)--with zone significance only. Because this pair of digits corresponds to sign time in the Data Processor, the sign will be printed with zone significance: the 0 in digit position 11 forces a blank during numeric time and the 1 translates the sign digit at zone time. Since the sign digit is transferred with zone significance, a negative sign is printed as a minus (-) sign rather than as a 1. A positive sign digit will be "printed" as a blank--rather than a 0--if the Suppress-12 mode is specified by the CARD WRITE instruction which uses this format band. If the Suppress-12 mode is not specified by the CARD WRITE instruction, an ampersand (&) will be printed (assuming the 407 has a Type A print wheel). The sign digit will occupy print position 110.

If the card-handling machine were a punch instead of a line printer, the sign digit would appear as an overpunch. A negative sign digit (1) would be an 11-punch. If the Suppress-12 mode were specified by the CARD WRITE instruction, a positive sign digit (0) would leave the 12-row blank; if the Suppress-12 mode were not specified, a positive sign digit would produce a 12-punch.

The 0's in format band digit positions 13 through 22 insert a blank field corresponding to print positions 109 through 105. Two 0's are required for each blank position, one for numeric time, and one for zone time at each of the control panel hubs.

THE CARDATRON SYSTEM

23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
				#8				25										
58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40
				3	1	1	1	1	1	1	1	1	1	1				

S T O R A G E													D I	
LOC.	#	±	1	2	3	4	5	6	7	8	9	0	REMARKS	LOC.
													[WAVEFORM]	
20	24	82	A	A	A	A	A	A	A	A	A	A		
													[WAVEFORM]	

The next word to be translated (the contents of location 2024) is designated as alphanumeric. The digit pair 11 in the format band is used to translate each pair of digits from two-digit Data Processor code to a single alphanumeric character.* In particular, the digits in format band digit positions 44 and 45 translate the contents of 2024:02; the resulting single alphanumeric character will occupy print position 88. The digit pair in format band digit positions 46 and 47 translate the contents of 2024:82; the resulting single alphanumeric character will occupy print position 87. Similarly, the contents of 2024:62 will be translated into the single alphanumeric character occupying print position 86. The contents of 2024:42 will be translated into the single alphanumeric character occupying print position 85. The contents of 2024:22 will be translated into the single alphanumeric character occupying print

*Note that in the STORAGE DISPLAY section of the FORMAT BAND CODING FORM two Data Processor digit positions--both designated as A== correspond to one alphanumeric character in a printing position, which is also designated as A.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

position 84. The digit 3 in format band digit position 54 causes the 2 in 2024:±1 to be deleted.

The reader should now be able to verify the coding which inserts a blank field into print positions 83 through 79; translates the word in location 2023 (designated as word number 7) alphabetically-- the alphanumeric characters occupying print positions 78 through 74; and inserts a blank field into print positions 73 through 69. (Refer to Figure VI-Intro-9.)

#6																		
Alphabetic and numeric S S S S Z N N N N A N A N																		
54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72

#6										21										22									
4	103	102	101	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83								
				1	0	0	1	0	1	0	1	2	1	1	2	1	1	2											

S T O R A G E													D	
LOC.	#	±	1	2	3	4	5	6	7	8	9	0	REMARKS	LOC.
20226	±		N	N	N	N	A	A	N	A	A	N		

The word to be translated next is designated as alphanumeric. The digit 2--in format band digit position 86--translates the digit in 2022:01 numerically; the decimal digit will occupy print position 68. The digit pair 11 in positions 87 and 88 of the format band translates the digits in 2022:92 into the single alphanumeric character occupying print position 67. The 2 in format band digit

position 89 translates the digit in 2022:71 numerically; the decimal digit will occupy print position 66. The digit pair 11 in format band digit positions 90 and 91 translates the pair of digits in 2022:62 as a single alphanumeric character occupying print position 65. The 2 in format band digit position 92 translates the digit in 2022:41 numerically; the decimal digit will occupy print position 64.

The digit pair 01 in format band digit positions 94 and 93 transfers the digit in 2022:31 with numeric significance: if the digit is different from 0, the non-zero digit will be printed in print position 63; if the digit is 0, a blank will be "printed" in print position 63. The digit pairs 01 in format band digit positions 96 and 95, and 98 and 97, will produce the same effect on the contents of 2022:22.

The pair of digits 10 in format band digit positions 100 and 99 translate the sign digit (2022:±1) with zone significance only. The effect is the same as that described for (2026:±1).

If the card-handling machine were a punch, the digit pair 01 in the format band would produce an underpunch 1 through 9 in the card if the digit in 2022:31, for example, were different from 0; if the digit were 0, a blank would appear in the card.

Again, it is left as an exercise for the reader to verify that the editing control stream digits occupying format band digit positions 101 through 222 will yield the line of print whose format is shown in Figure VI-Intro-9.

The part of the format band occupying format band digit positions 1 through 222 is called the active segment of the format band. The active segment of a format band is always comprised of those editing control stream digits which lie between and include format band digit position 1 and either:

- a. the editing control stream digit in the position which refers to the last digit transferred to the card-handling machine if that is an inserted digit; or
- b. the editing control stream digit in the position which corresponds to zone time at FROM CARDATRON hub 1, if no inserted digits follow that reference.

The part of the format band which follows the active segment is called the inactive segment. It always contains 3's. Since format band digit position 316 contains the last editing control stream digit, the contents of the remainder of the format band word designated 1 are irrelevant. These three positions may be coded arbitrarily, since they will not be written on the format band, there being no room for them.

It is recommended that output editing control streams be constructed as if 120 print or punch positions were available for use. That is, format band digit position 1 should always correspond to print or punch position 120. This means that the active segment of an editing control stream for a punch will always have 0's in format band digit positions 1 through 80, corresponding to the

THE CARDATRON SYSTEM

(non-existent) punch positions (i.e., card columns) 120 through 81, respectively.

The section of the CODING FORM labeled DIGIT CHECK FOR ACTIVE SEGMENT may be used to help in detecting clerical errors when constructing editing control streams for an output format band.

DIGIT CHECK FOR ACTIVE SEGMENT

INPUT	OUTPUT
NUMBER OF 0's = _____	NUMBER OF 0's = <u>112</u>
NUMBER OF 1's = _____	NUMBER OF 1's = <u>34</u>
NUMBER OF 2's = _____	NUMBER OF 2's = <u>47</u>
NUMBER OF 3's = _____	NUMBER OF 3's = <u>29</u>
SUM* = _____	SUM* = <u>222</u>
$\frac{0's + 1's + 2's}{11} = \text{WORDS PER CARD}$	$\frac{0's + 1's + 2 \times 2's}{2} = \text{COLUMNS PRINTED OR PUNCHED}$
_____ =	$\frac{112 + 34 + 94}{2} = 120$
$\frac{1's + 3's}{2} = \text{CARD COLUMNS}$	$\frac{1's + 2's + 3's}{11} = \text{DATATRON WORDS}$
_____ =	$\frac{34 + 47 + 29}{11} = 10$
*If the active segment starts in format band digit position 1, the sum of 0's, 1's, 2's, and 3's should be identical with the format band digit position containing the last digit of the active segment.	

In the section of the FORMAT BAND CODING FORM labeled PUNCHED-CARD PREPARATION columns 4 through 14 of each card are reserved for a bootstrap instruction which will read the next card. The set of instructions shown will cause the 29 words of the editing control stream to be stored in locations 1221 through 1249: word 29 will be in location 1249, word 28 in location 1248, and so forth.

INSTRUCTIONS

CARD WRITE (CWR) is an instruction that transfers up to 316 digits from sequentially-addressed storage locations in the Data Processor to the information band of a designated Output Unit. The instruction also designates the format band whose contents are to edit the information transferred. In addition, it must be specified whether the Suppress-12 mode is elected. The first digits transferred are taken from the storage location specified by the address part of the CARD WRITE instruction. The remaining digits are taken from storage locations in order of sequentially-descending addresses.

Referring to Figure VI-Intro-9, the contents of storage locations 2017 through 2026--as shown in the STORAGE DISPLAY section--will be transferred to the information band of the designated Output Unit by a CARD WRITE instruction with 2026 as its address part. Because this instruction can transfer up to 316 digits, the contents of storage locations 2016 through 1998 will also be transferred to the information band. The information contained in these 19 locations is deleted when the contents of the information band are transferred to the output Core Shift Register.

CARD WRITE, FORMAT LOAD (CWF) is an instruction that provides the ability to load the editing control stream digits onto the buffer drum of an Output Unit. It transfers 316 digits--stored in 29 consecutively-addressed storage locations--to the designated format band of a designated Output Unit. The contents of the

location specified by the address part of the CARD WRITE, FORMAT LOAD instruction will occupy the first 11 digit positions on the format band. Suppose the instruction were CWF 1249. Then (1249:01) goes to format band digit position 1; (1249:91) goes to format band digit position 2; ...; and (1249:±1) goes to format band digit position 11. The word in location 1248 will occupy format band digit positions 12 through 22. And so forth. Finally, (1221:09) will occupy the last nine digit positions on the format band.

Referring again to Figure VI-Intro-9, the editing control stream digits were stored in locations 1221 through 1249 by the bootstrap instructions in columns 4 through 14. A CWF 1249 instruction transferred 316 of these digits to the desired format band as described above.

CARD WRITE INTERROGATE, BRANCH (CWI) is an instruction that permits the program to determine if a designated Output Unit is ready to be used. If the unit queried is ready, control is transferred, that is, the contents of the P register are replaced by the address part of the CWI instruction; otherwise, control continues in sequence.

INFORMATION FLOW

The general nature of information flow during input and output was discussed in Section II. In many details, however, the flow of information during the execution of instructions referring to Cardatron is different from that indicated in Section II. For this reason

the flow of information is presented again here. The description which follows supplements the description of the transfer of information presented previously in this section.

INPUT FLOW

Input flow is shown in Figure VI-Intro-10.

1. (rD)--the D register contains an image of the instruction brought from storage during the Fetch Phase--are transferred to the CD register in the Cardatron Control Unit.

2. The address part of the C register (rC:04), which specifies the location in storage where the first word transferred from the information band is to be stored, is transferred to the E register.

At this time the clock in the Data Processor is turned off, the Cardatron clock being the source of pulses which govern the operation of both Cardatron and Data Processor during Cardatron operations.

3. The first word is transferred from the information band to the D register.

4a. The contents of the D register are then transferred, two digits at a time, to the IB register, with or without B-register address modification as specified and indicated.

4b. Each time the D register is shifted to the right to force the transfer of the next pair of digits, the next digit from the information band enters rD:±1.

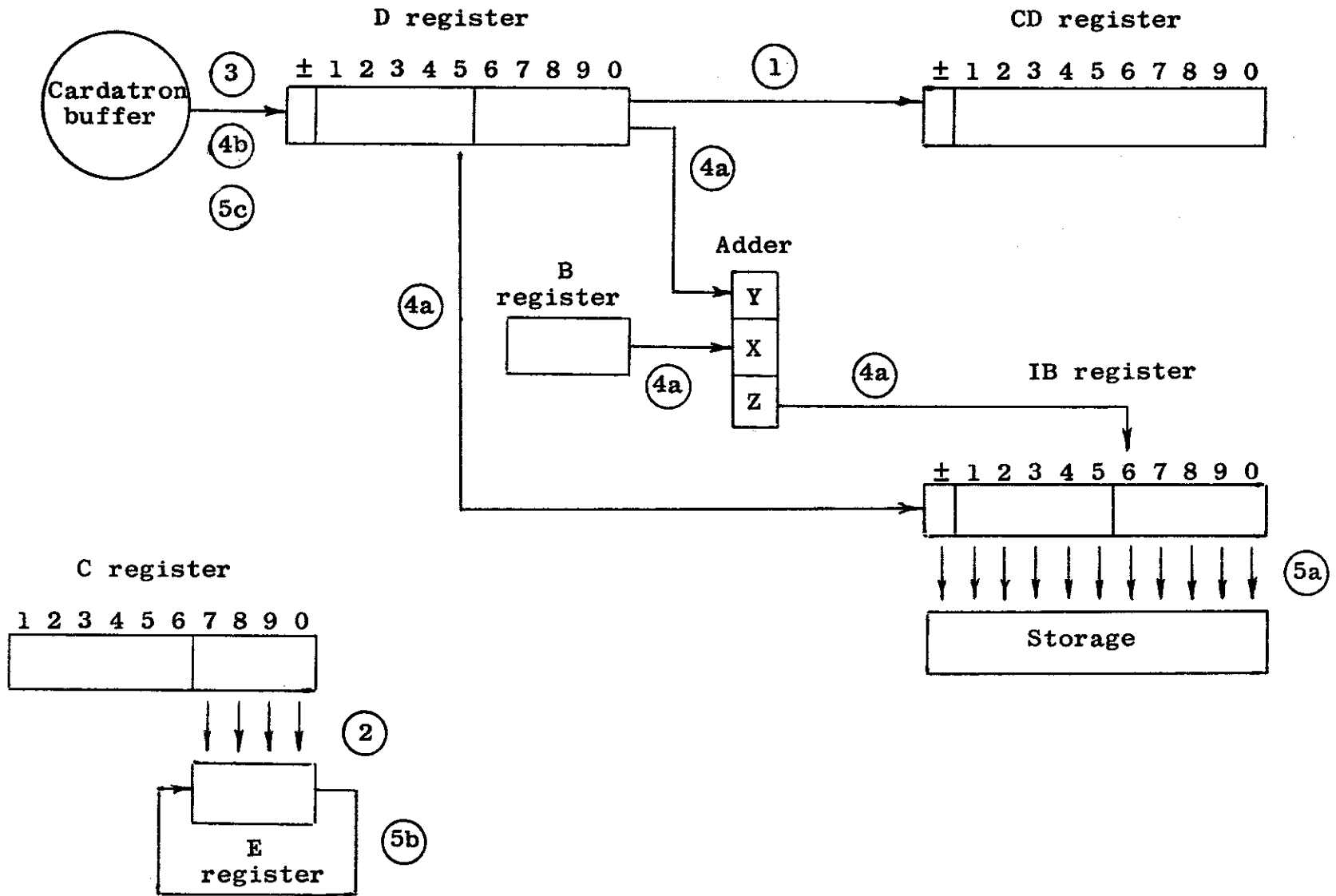


Figure VI-Intro-10. Input flow

5a. As soon as the IB register is filled its contents are transferred to the location whose address is the contents of the E register.

5b. Then the contents of the E register are decreased by 1.

5c. While the contents of the IB register are being transferred to storage and the E register is being counted down, the D register is being filled with the remainder of the next word transferred from the information band.

If all of the information has not been transferred from the information band, as is determined by the editing control stream governing the transfer and the possible occurrence of a control word read from the card--that is, a word whose sign digit is a 6 or 7--there is a return to step 4.

The explicit detail of steps 4 and 5 will be found in the flow chart which is a part of the description of the Execute Phase of the CARD READ instruction.

The CARD READ, FORMAT LOAD instruction is essentially an output instruction: information is transferred from core storage to the buffer during its execution. Thus the flow of information during execution of a CRF instruction is shown below.

OUTPUT FLOW

Output flow is shown in Figure VI-Intro-11.

1. (rD) are transferred to the CD register.

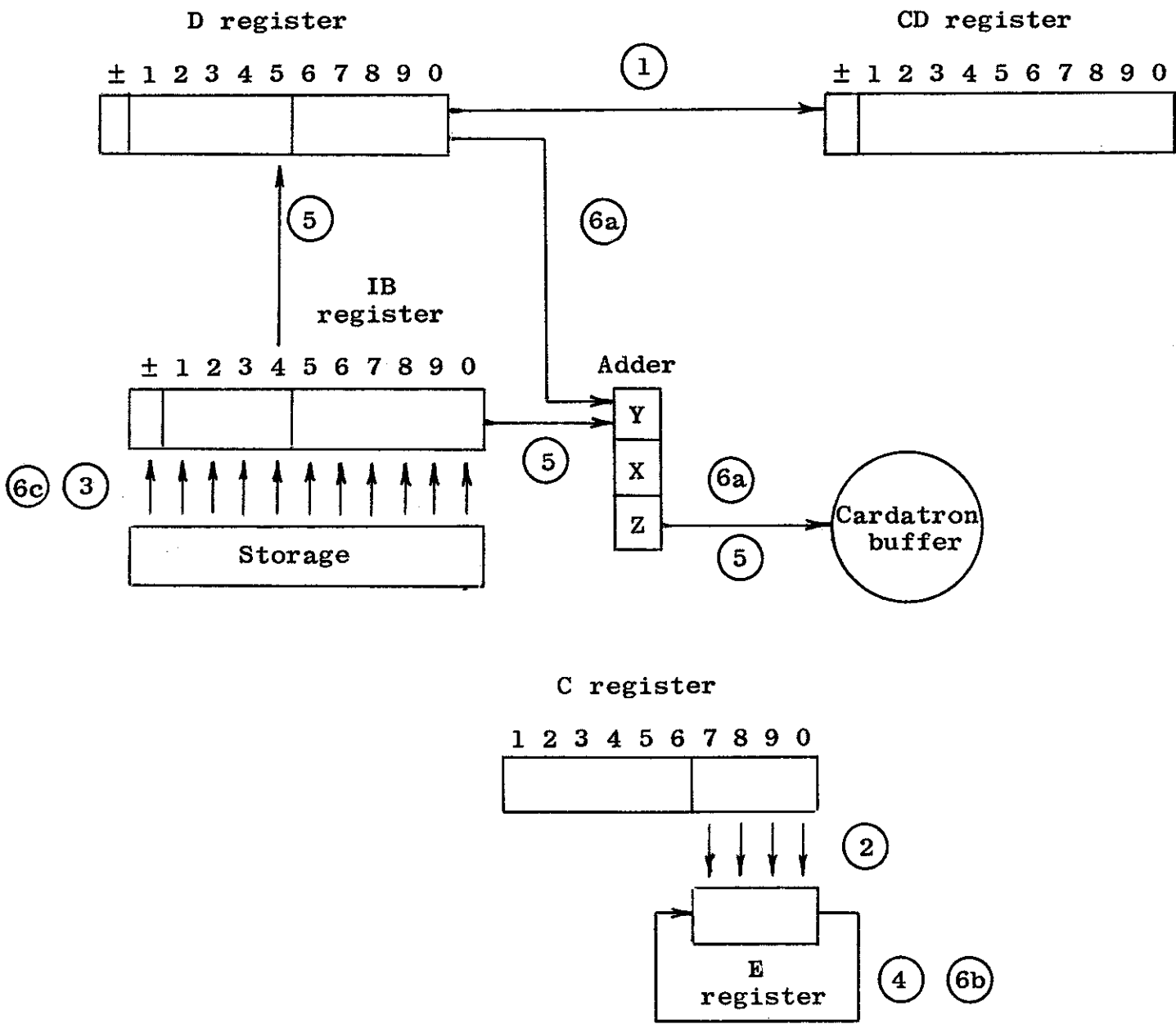


Figure VI-Intro-11. Output flow

2. The address part of the C register (rC:04), which specifies the location in storage from which is taken the first digit to be transferred to the buffer, is transferred to the E register.

At this time the clock in the Data Processor is turned off, the Cardatron clock being the source of pulses which govern the operation of both Cardatron and Data Processor during Cardatron operations.

3. The contents of the location whose address is in the E register are transferred to the IB register.

4. The contents of the E register are decreased by 1.

5. The contents of the IB register are transferred, two digits at a time, one digit to the D register and one digit to the adder. As each successive digit enters the adder the previous contents are transferred to the buffer.

Each time a digit is transferred from the IB register to the D register, the contents of the D register are shifted one place to the right.

6a. After the sixth digit is transferred from the IB register to the adder, the sign digit and the four high-order digits which were transferred to the D register are transferred from rD to the adder and thence to the buffer.

6b. While the transfer of information from the D register is occurring, the E register is counted down by 1.

6c. The next word ((rE)) is transferred to rIB.

If all of the information has not been transferred to the buffer, there is a return to step 5.

The explicit detail of steps 5 and 6 will be found in the flow chart which is a part of the description of the Execute Phase of each of the instructions, CARD WRITE; CARD WRITE, FORMAT LOAD; and CARD READ, FORMAT LOAD.

EXCEPTIONAL CONDITIONS

1. A non-existent-address ALARM STOP will occur if an instruction refers to a location not in the storage package in the system. The effects of this condition during input are different from those during output. See the Execute Phase description of the relevant instructions for complete details.
2. A digit-check ALARM STOP can occur.
3. If the Input Unit or Output Unit designation specified by an instruction is not one which has actually been assigned, that is, if there is no unit having the specified designation, a non-existent-Input-Unit or non-existent-Output-Unit ALARM STOP will occur. The Data Processor will stop at the beginning of the Execute Phase.
4. A no-format ALARM STOP will occur if no format band is selected when a card is being read. No such alarm can occur on output.

EXECUTE PHASE

The remainder of Section VI is devoted to a description of the Execute Phase of all instructions which refer to the Cardatron.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD READ

Operation code: 60

Instruction format:

Abbreviation: CRD

±	1	2	3	4	5	6	7	8	9	0
±	u	i	v	r	Op	a	a	a	a	

Time (μs):

Minimum:

fetch: 90
 execute: 5765
 total: 5855

Average:

fetch: 90
 execute: 7180
 total: 7270

Maximum:

fetch: 90
 execute: 9020
 total: 9110

Definitions:

±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.

u: designates Input Unit.

i: not relevant to the execution of this instruction.

v: variation designator:

v = 0: control words will be recognized as such.

v = 1: control words will not be recognized as such.

r: r = 8: designated input will be B-register address-modified.

r = 0: no B-register address-modification of input.

r = 9: designated input will be B-register address-modified; Reload Lockout will be imposed.

r = 1: no B-register address-modification of input; Reload Lockout will be imposed.

Op: operation code

aaaa: address of base of location into which is written first word transferred from the information band.

Description of operation:

Summary:

Transfer the contents of the information band from Input Unit u to core storage. The first 11 digits transferred from the information band are stored in location B[aaaa]; the next 11 digits transferred are stored in location B[aaaa] - 1; the next 11 digits transferred are stored in location B[aaaa] - 2; and so forth. The editing control stream used to edit the contents of the information band is the one on the format band which was selected by the card whose contents are on the information band.

If $r = 1$ or 9 , Reload Lockout will be imposed, that is, the transfer to the information band of the contents of the next card in the card reader will be inhibited.

If $r = 0$ or 8 , Reload Lockout will be released, if it had been imposed.

If Format Lockout was imposed by the card whose contents are being transferred by this CRD instruction, the execution of this CRD instruction will release Format Lockout, leaving Reload Lockout set. Thus, the contents of the card following the one which imposed Format Lockout will not be automatically transferred to the information band after completion of this CRD instruction. Another Cardatron cycle--the execution of either a CRD or CRF instruction--is required to release Reload Lockout.

Input sign control is effected as described in the table on the following page.

Flow chart:

See page VI-60-9.

Sign Digit	Where Sensed	r = (rC:41)	v = (rC:31)	Effect
0	rD	Any	Any	None. (Sign digit stored as 0.)
1	rD	Any	Any	None. (Sign digit stored as 1.)
2	rD	Any	Any	None. (Sign digit stored as 2.)
3	rD	Any	Any	None. (Sign digit stored as 3.)
4	rD	Any	Any	None. (Sign digit stored as 4.)
5	rD	Any	Any	None. (Sign digit stored as 5.)
6	rIB	Any	0	This is a control word: terminate the transfer of information from the information band; release the Input Unit to read the next card; enter the Fetch Phase at connector P (see page II-Intro-15). The control word is then treated as an instruction (because the sign digit is even, there will be no B-register address modification).
			1	None. (Sign digit stored as 6.)
7	rIB	Any	0	Except that B-register address modification will occur during the Fetch Phase--because the sign digit is odd--the effect is the same as that when the sign digit is 6 and v = 0.
			1	None. (Sign digit stored as 7.)
8	rD	≠ 8, ≠ 9	Any	None. (Sign digit stored as 8.)
		8, 9	Any	B-register address modification occurs; the sign digit is stored as 0.
9	rD	≠ 8, ≠ 9	Any	None. (Sign digit stored as 9.)
		8, 9	Any	B-register address modification occurs; the sign digit is stored as 1.

Cardatron Input Sign Control

VI-60-4

Replaces 9/1/57
7/1/58

Exceptional conditions:

1. Non-existent-address ALARM STOP. (See Remark 8, below.)
2. No-format ALARM STOP.
3. Non-existent-Input-Unit ALARM STOP.
4. Digit-check ALARM STOP. (See Remark 9, below.)

Remarks:

1. $r = 3, 5, \text{ or } 7$ has the same effect as $r = 1$; $r = 2, 4, \text{ or } 6$ has the same effect as $r = 0$.

2. $v = 2, 4, 6, \text{ or } 8$ has the same effect as $v = 0$; $v = 3, 5, 7, \text{ or } 9$ has the same effect as $v = 1$.

3. The inactive segment of a format band must contain seven 0's. (Usually these 0's will occupy format band digit positions 309 through 315.) If the inactive segment does not contain these seven 0's, the last word transferred from the information band will not be written in core storage: it will be lost to the program.

4. If Format Lockout is imposed, it is done by selecting format band 8. Format band 8 must be selected in conjunction with the selection of some other format band: if it is not, a no-format ALARM STOP will occur.

5. Once Reload Lockout is imposed, it can be removed only by specification in a CRD or CRF instruction: removal of Reload Lockout is achieved by making r an even digit.

6. At the beginning of the Execute Phase of a CRD instruction $B[aaaa] = (rC:04)$ is transferred to the E register. The address of the core storage location into which will be written each successive word transferred from the information band is obtained by decreasing by 1 the contents of the E register after each word is stored.

Since the E register counts modulo the size of storage, care must be exercised in the choice of $B[aaaa]$. For example, suppose the Data Processor has 4000 words of core storage; suppose also that $B[aaaa]$ is chosen so that some word is transferred to location 0000. After this word is stored, 1 will be subtracted from 0000, the contents of the E register, modulo 4000. The E register will then contain 3999.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

7. The selection of numeric format--format band 6--allows a maximum of 13 numeric words to be transferred from the information band to core storage when the next CRD instruction is executed: all overpunches will be deleted.

Suppose, for example, that the instruction is 0 1000 CRD 1013; suppose also that the control panel is wired 80-80. The execution of the specified CRD instruction will load storage as displayed in the table below:

Location	Contents of numeric part of card column stored in digit position										
	±	1	2	3	4	5	6	7	8	9	0
1013	70	71	72	73	74	75	76	77	78	79	80
1012	59	60	61	62	63	64	65	66	67	68	69
1011	48	49	50	51	52	53	54	55	56	57	58
1010	37	38	39	40	41	42	43	44	45	46	47
1009	26	27	28	29	30	31	32	33	34	35	36
1008	15	16	17	18	19	20	21	22	23	24	25
1007	4	5	6	7	8	9	10	11	12	13	14
1006	←	inserted 0's						→	1	2	3
1005	←	inserted 0's						→			
1004	←	inserted 0's						→			
1003	←	inserted 0's						→			
1002	←	inserted 0's						→			
1001	←	inserted 0's						→			

If it is desired to prevent the insertion of zeros as shown above, it is necessary to supply the card with a word having a 6 or 7 in the sign-digit position. Suppose, for example, that it is desired to transfer only four words from a card which has selected numeric format; suppose also that the control panel is wired 80-80. Then these four words must occupy the fields comprised of columns 70 - 80, 59 - 69, 48 - 58, 37- 47, inclusive. The field comprised of columns 26 - 36, inclusive, must be an instruction, and must have a 6 or 7 in the sign-digit position (column 26). Let the CRD instruction be as specified above. The execution of that instruction will load storage as displayed below:

Location	Contents of numeric part of card column stored in digit position										
	±	1	2	3	4	5	6	7	8	9	0
1013	70	71	72	73	74	75	76	77	78	79	80
1012	59	60	61	62	63	64	65	66	67	68	69
1011	48	49	50	51	52	53	54	55	56	57	58
1010	37	38	39	40	41	42	43	44	45	46	47

No other locations in storage will be affected.

8. Once the transfer of information from the information band to the Data Processor has begun, it can terminate only when all of the information has been transferred to the D register. Examination of the flow chart will disclose that the information transfer is already under way when a non-existent-address ALARM STOP can occur (it occurs before the first word is written in core storage). Hence, none of the words will be stored; in other words, the information is "lost," that is, the card whose contents were on the information band must be re-read.

9. A digit-check ALARM STOP can occur after the execution of a CRD instruction only as a result of detecting the impermissible configuration when the attempt is made to shift the digit out of rD:01. Examination of the flow chart will disclose that it is possible to write in core storage a word with an impermissible configuration in some digit position. This can happen because:

- a. The contents of the D register are transferred to the IB register in two parts, the high-order part of the word being split off with rD:51. Hence, if the impermissible configuration first occurs in any digit position in rD:56 before the transfer to rIB of (rD) is begun, it will be transferred to rIB; and
- b. Since the contents of the IB register are transferred to storage in parallel, no check is made in rIB for the impermissible configuration.

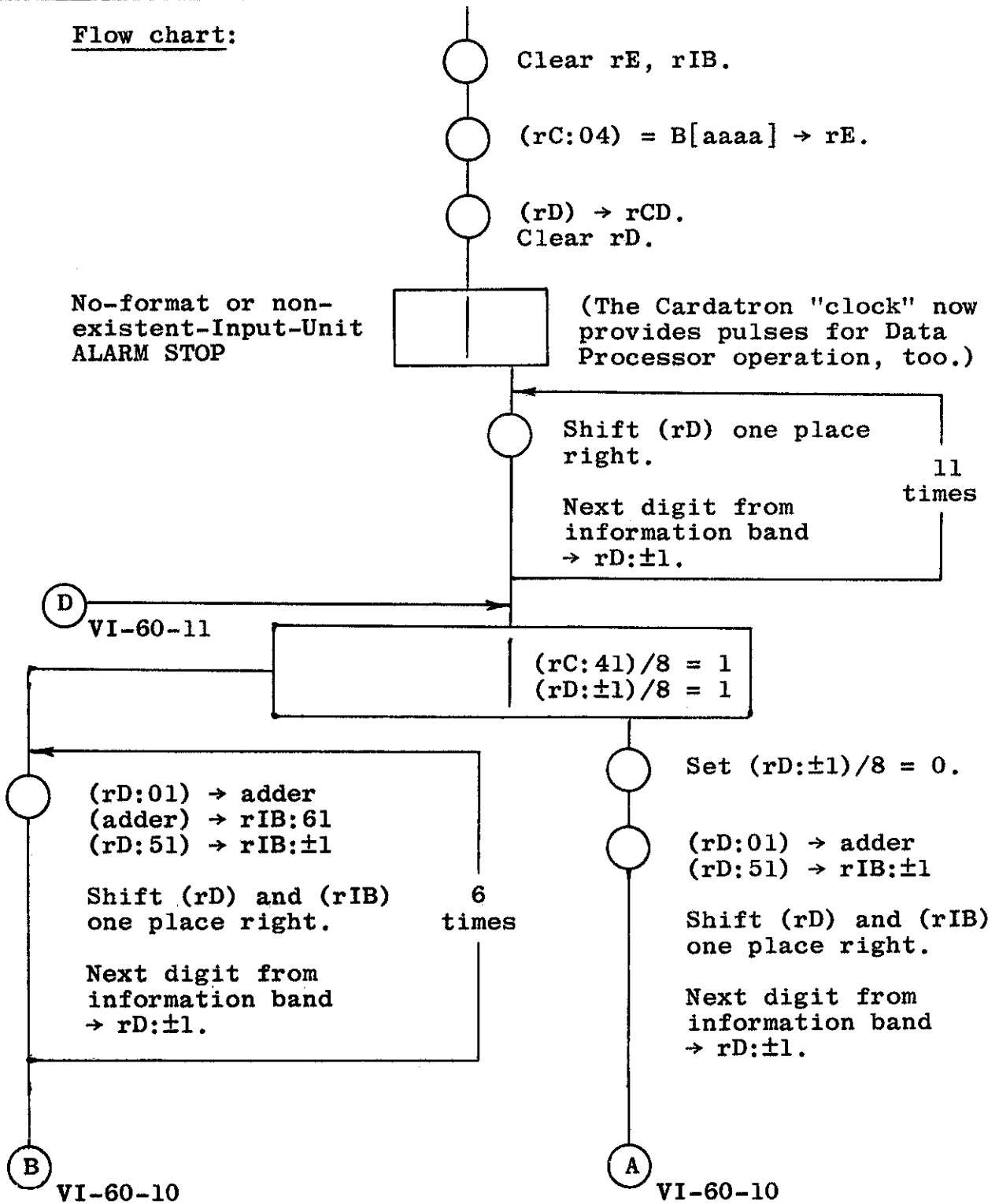
Further examination of the flow chart discloses that the contents of the D register are shifted to the right as each additional digit is transferred from the information band to the D register. Some digits are transferred to rD--whose contents are shifted to the right as each digit is entered in rD:±1--while the word in rIB is being written in storage: the transfer to storage is completed before the D register is completely filled with the next word from the information band.

Again it should be noted that once the transfer of information to the Data Processor has begun, it can terminate only when all of the information has been transferred to rD. Thus, digits from the information band will "pile up" in rD:±1 immediately after the impermissible configuration is detected. In no case can the ALARM STOP be prevented from occurring. In any event:

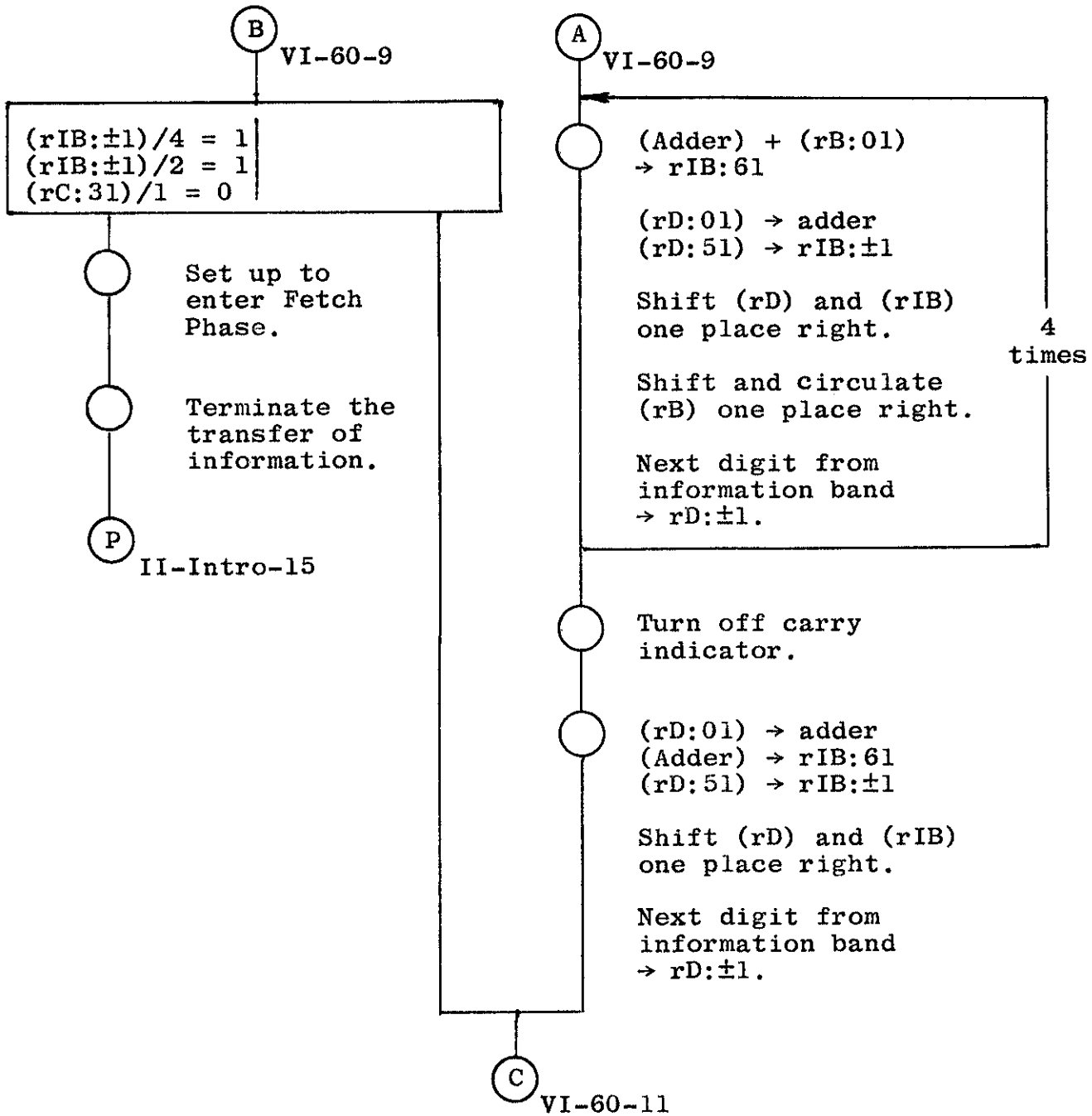
- a. It is not possible to predict with certainty what will be in the D register;
- b. The card whose contents were on the information band must be re-read; and
- c. The E register will provide a clue to the location in storage of the word with the impermissible configuration. Suppose the word is in aaaa. Then $(rE) = aaaa - 1$.

Description of operation:

Flow chart:

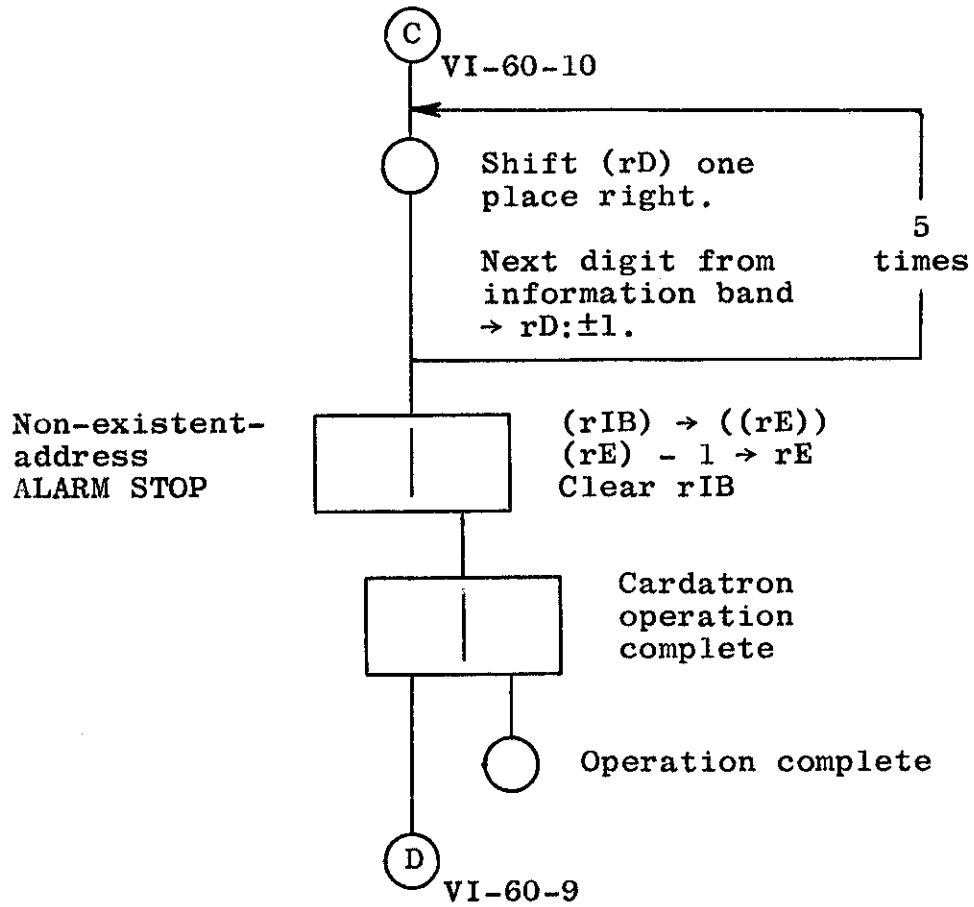


Flow chart (continued):



THE CARDATRON SYSTEM

Flow chart (continued):



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Register status:

Register name	Contents after execution of CRD.	Contents if non-existent-address ALARM STOP occurs														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	*	*														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; text-align: center;">u</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">r</td> <td style="border: 1px solid black; text-align: center;">6</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">B[aaaa]</td> </tr> </table>	u	i	i	r	6	0	B[aaaa]	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; text-align: center;">u</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">r</td> <td style="border: 1px solid black; text-align: center;">6</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">B[aaaa]</td> </tr> </table>	u	i	i	r	6	0	B[aaaa]
u	i	i	r	6	0	B[aaaa]										
u	i	i	r	6	0	B[aaaa]										
E	Address of last location filled - 1.	B[aaaa]														

* (rD:67) = 0000000; (rD:04) are the four high-order digits of the last word transferred from the buffer.

Register name	Contents if no-format or non-existent-Input-Unit ALARM STOP occurs.	Contents if digit-check ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	**														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; text-align: center;">u</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">r</td> <td style="border: 1px solid black; text-align: center;">6</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">B[aaaa]</td> </tr> </table>	u	i	i	r	6	0	B[aaaa]	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; text-align: center;">u</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">i</td> <td style="border: 1px solid black; text-align: center;">r</td> <td style="border: 1px solid black; text-align: center;">6</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">B[aaaa]</td> </tr> </table>	u	i	i	r	6	0	B[aaaa]
u	i	i	r	6	0	B[aaaa]										
u	i	i	r	6	0	B[aaaa]										
E	B[aaaa]	Address of last location filled - 1.														

** Depends on when the impermissible configuration is detected. See flow chart.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD WRITE

Instruction format:

± 1 2 3 4 5 6 7 8 9 0

±	u	i	c	f	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Definitions:

- ±: if ± is odd, B-register address modification will occur; otherwise, there will be no such modification.
- u: designates Output Unit.
- i: not relevant to the execution of this instruction
- c: specifies which of the T RELAYS is/are selected for additional control of the line printer.
- f: specifies which format band is used to edit the output; in addition, if
 - f is even: the Suppress-12 mode of printing or punching is selected.
 - f is odd: the Suppress-12 mode is not selected.
- Op: operation code.
- aaaa: address of base of location from which is taken the first digit transferred to the information band.

Operation code: 61

Abbreviation: CWR

Time (μs):

Minimum:

fetch:	90
execute:	8545
total:	8635

Average:

fetch:	90
execute:	9960
total:	10050

Maximum:

fetch:	90
execute:	11940
total:	12030

Description of operation:

Summary:

Transfer to the information band of Output Unit u the contents of up to 29 core storage locations as follows: the contents of B[aaaa] are transferred first; (B[aaaa] - 1) are transferred next; ...; (B[aaaa] - 27) are transferred next; (B[aaaa] - 28:08) are transferred last. The number of digits transferred is determined by the editing control stream on the format band which is specified by f.

The format band whose contents are used to edit the information transferred to the information band is specified in the table below:

f	Format Band	Suppress-12 Mode?
0	1	Yes
1	1	No
2	2	Yes
3	2	No
4	3	Yes
5	3	No
6	4	Yes
7	4	No
8	5	Yes
9	5	No

The Suppress-12 mode has to do with overpunching the sign digits of numeric words, that is, words whose signs are either 0 or 1, in accordance with either of the two conventions for overpunching signs. The Suppress-12 mode also has to do with printing the sign digits of numeric words (in the tables below it is assumed that the printer is a 407 with Type A print wheels).

Two cases need to be considered:

Case 1: the sign digit is translated by the editing control stream digit pair 10. In this case the sign digit occupies a column or print position by itself.

Suppress-12 Mode?	$\pm = 0$		$\pm = 1$	
	Punch	Print	Punch	Print
Yes	blank	blank	11	-
No	12	&	11	-

Case 2: the digit preceding the sign digit and the sign digit are translated by the editing control stream digit pair 11. In this case the sign digit shares a column or print position with the digit which precedes it.

Suppress-12 Mode?	$\pm = 0$		$\pm = 1$	
	Punch	Print	Punch	Print
Yes	Blank	a	11	b
No	12	c	11	b

- a. The digit preceding the sign digit.
- b. The alphanumeric character whose code is 11-n, where n is the digit preceding the sign digit.
- c. The alphanumeric character whose code is 12-n, where n is the digit preceding the sign digit.

The digit c is used to transfer the T RELAYS whose points are available on the control panel of the printer. After the edited contents of the information band have been transferred to the printer, all of the relays will have been returned to their normal (N) position: they are returned to normal just prior to the transfer from the Core Shift Register of the last row of information. The selection of relays specified by c is shown in the following table:

c	Relays Selected
0	None
1	1
2	2
3	3
4	4
5	5
6	2 and 4
7	3 and 5
8	None
9	1

As soon as the transfer of information from core storage to the information band is completed, the Output Unit automatically directs the transfer of the contents of the information band to the card-handling machine.

Flow chart:

See page VI-61-7.

Exceptional conditions:

1. Non-existent-address ALARM STOP. (See Remark 2, below.)
2. Non-existent-Output-Unit ALARM STOP.
3. Digit-check ALARM STOP. (See Remark 3, below.)

Remarks:

1. At the beginning of the Execute Phase of a CWR instruction B[aaaa] = (rC:04) is transferred to the E register. The address of the core storage location from which is read each successive word transferred to the information band is obtained by decreasing by 1 the contents of the E register after each word is read.

Since the E register counts modulo the size of storage, care must be exercised in the choice of B[aaaa]. For example, suppose the Data Processor has 4000 words of core storage; suppose also that B[aaaa] < 0028: hence, some word will be read from location 0000. After this word is read, 1 will be subtracted from 0000, the contents of the E register, modulo 4000. The E register will then contain 3999.

2. Examination of the flow chart will disclose that control is vested in the Cardatron--by virtue of the fact that the Cardatron "clock" is providing pulses for Data Processor operation, too--when a non-existent address is detected. The detection of the non-existent address inhibits the transfer of any information in the Data Processor. But the Cardatron must complete its cycle of operation, even though no information is transferred from the Data Processor to the buffer. The ALARM STOP will occur after Part 1 of the Cardatron cycle is completed. Part 2 of the Cardatron cycle will also occur:

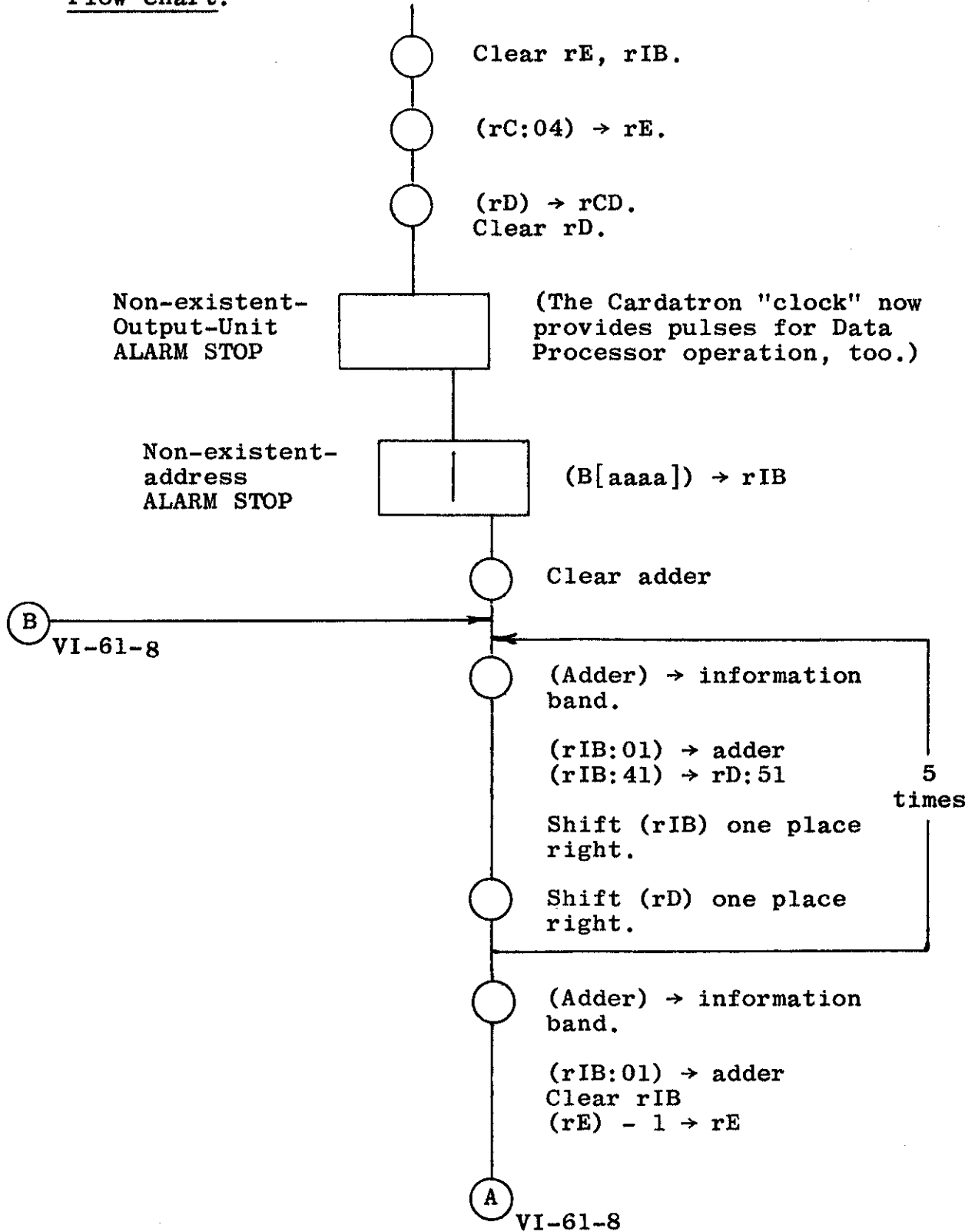
- a. If the Output Unit is connected to a punch, a card will be fed. The card will contain a mixture of blank columns and 0-punches, the configuration being determined by the contents of the format band selected by the CWR instruction.

- b. If the Output Unit is connected to a printer, a blank line will be "printed."

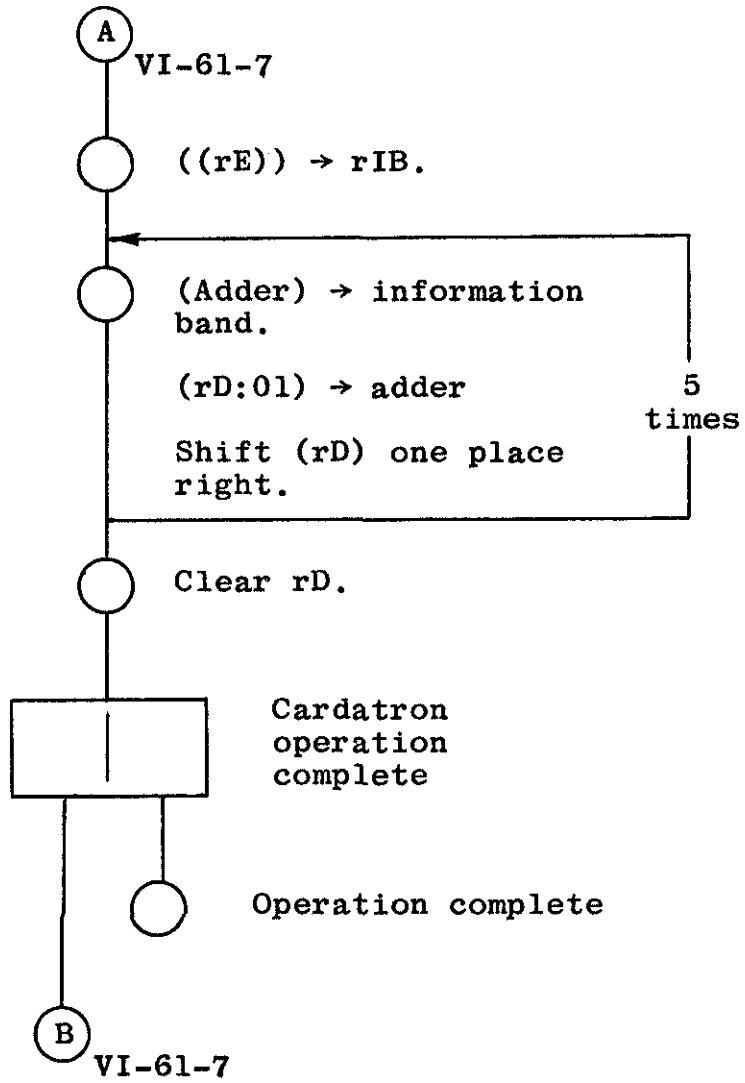
3. The detection of an impermissible configuration occurs when (rIB:01) and (rD:01) are shifted to the right. Such an event can occur only after control is vested in the Cardatron (see the flow chart). The detection of the impermissible configuration immediately stops the transfer of information within the Data Processor, and from the Data Processor to the buffer. But Cardatron must complete its cycle of operation, even though no more information is transferred to it. The digit-check ALARM STOP will occur after Part 1 of the Cardatron cycle is completed. Part 2 of the Cardatron cycle will also occur. The nature of the output depends both on the editing control stream and on how much information was transferred from the Data Processor to the buffer.

Description of operation:

Flow chart:



Flow chart (continued):



THE CARDATRON SYSTEM

Register status:

Register name	Contents after execution of CWR.	Contents if digit-check ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	*														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td><td style="padding: 2px;">i</td><td style="padding: 2px;">c</td><td style="padding: 2px;">f</td><td style="padding: 2px;">6</td><td style="padding: 2px;">l</td><td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	c	f	6	l	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td><td style="padding: 2px;">i</td><td style="padding: 2px;">c</td><td style="padding: 2px;">f</td><td style="padding: 2px;">6</td><td style="padding: 2px;">l</td><td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	c	f	6	l	B[aaaa]
u	i	c	f	6	l	B[aaaa]										
u	i	c	f	6	l	B[aaaa]										
E	$B[aaaa] - 29$	*														

*Depends on when the impermissible configuration is detected. See flow chart.

Register name	Contents if non-existent-address or non-existent-Output-Unit ALARM STOP occurs.							
A	Unchanged							
R	Unchanged							
D	Cleared							
B	Unchanged							
P	$(rP)_b + 1$							
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td><td style="padding: 2px;">i</td><td style="padding: 2px;">c</td><td style="padding: 2px;">f</td><td style="padding: 2px;">6</td><td style="padding: 2px;">l</td><td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	c	f	6	l	B[aaaa]
u	i	c	f	6	l	B[aaaa]		
E	$B[aaaa]$							

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD READ, FORMAT LOAD

Operation code: 62

Abbreviation: CRF

Instruction format:

Time (μ s):

\pm 1 2 3 4 5 6 7 8 9 0

\pm	u	i	i	f	Op	a	a	a	a
-------	---	---	---	---	----	---	---	---	---

Minimum:

fetch:	90
execute:	8545
total:	<u>8635</u>

Definitions:

Average:

\pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.

fetch:	90
execute:	9960
total	<u>10050</u>

u: designates Input Unit.

ii: not relevant to the execution of this instruction.

Maximum:

f: specifies which format band will be written.

fetch:	90
execute:	11940
total:	<u>12030</u>

Op: operation code.

aaaa: address of base of location from which is read the first word transferred to the format band specified by f.

Description of operation:

Summary:

Transfer to the format band specified by f--on Input Unit u--the contents of 29 core storage locations as follows: the contents of B[aaaa] occupy format band digit positions 1 through 11; (B[aaaa] - 1) occupy positions 12 through 22; ...; (B[aaaa] - 27) occupy positions 298 through 308; (B[aaaa] - 28:07) occupy positions 309 through 315.

The selection of the format band on which the editing control stream is written and the imposition or release of Reload Lockout are accomplished as described in the table below:

f	Format Band	Reload Lockout
0	1	Released
1	1	Imposed
2	2	Released
3	2	Imposed
4	3	Released
5	3	Imposed
6	4	Released
7	4	Imposed
8	5	Released
9	5	Imposed

Flow chart:

See page VI-62-5.

Exceptional conditions:

1. Non-existent-address ALARM STOP. (See Remark 3, below.)
2. Non-existent-Input-Unit ALARM STOP.
3. Digit-check ALARM STOP (See Remark 4, below.)

Remarks:

1. A format band digit position contains only two bits. Hence, a format band digit may be only one of 0, 1, 2, or 3. If an attempt is made to write as a format band digit some decimal digit greater than or equal to 4, the 4-bit or the 8-bit will be dropped when that decimal digit is transferred to the selected format band.

2. At the beginning of the Execute Phase of a CRF instruction B[aaaa] = (rC:04) is transferred to the E register. The address of

the core storage location from which is read each successive word transferred to the format band is obtained by decreasing by 1 the contents of the E register after each word is read.

Since the E register counts modulo the size of storage, care must be exercised in the choice of B[aaaa]. For example, suppose the Data Processor has 4000 words of core storage; suppose also that $B[aaaa] \leq 0028$; then some word will be read from location 0000. After this word is read, 1 will be subtracted from 0000, the contents of the E register, modulo 4000. The E register will then contain 3999.

3. Examination of the flow chart will disclose that control is vested in the Cardatron--by virtue of the fact that the Cardatron "clock" is providing pulses for Data Processor operation, too--when a non-existent address is detected. The detection of the non-existent address inhibits the transfer of information in the Data Processor. But the Cardatron must complete its cycle of operation, even though no information is transferred from the Data Processor to the buffer.

The ALARM STOP will occur after the Cardatron cycle is completed. It will appear that the designated format band had all 0's written on it.

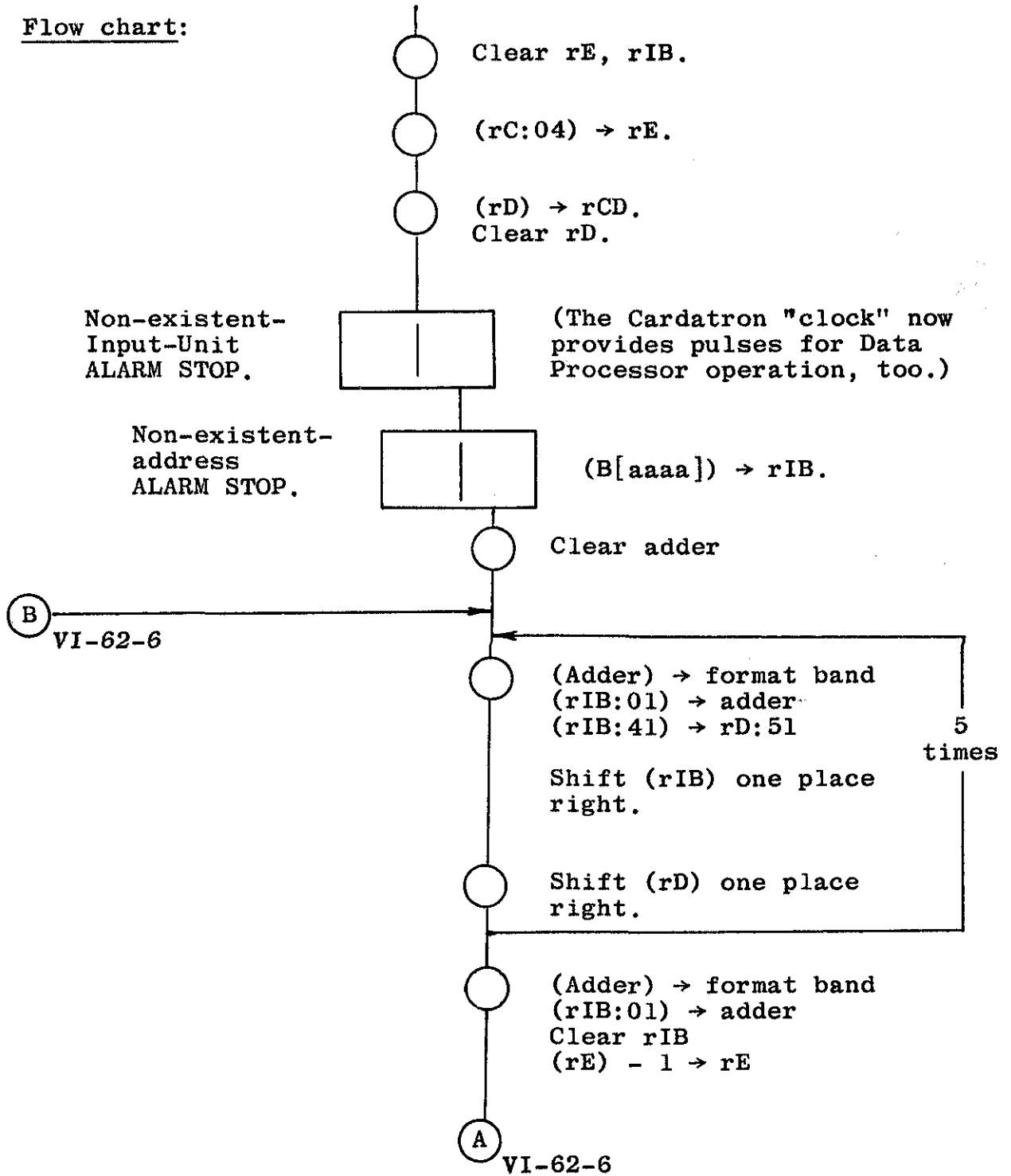
4. The detection of an impermissible configuration occurs when (rIB:01) and (rD:01) are shifted to the right. Such an event can occur only after control is vested in the Cardatron (see the flow chart). The detection of the impermissible configuration immediately stops the transfer of information within the Data Processor, and from the Data Processor to the buffer. But the Cardatron must complete its cycle of operation, even though no more information is transferred to it.

The digit-check ALARM STOP will occur after the Cardatron cycle is completed. It will appear that the designated format band contains 0's following the last digit transferred from the Data Processor.

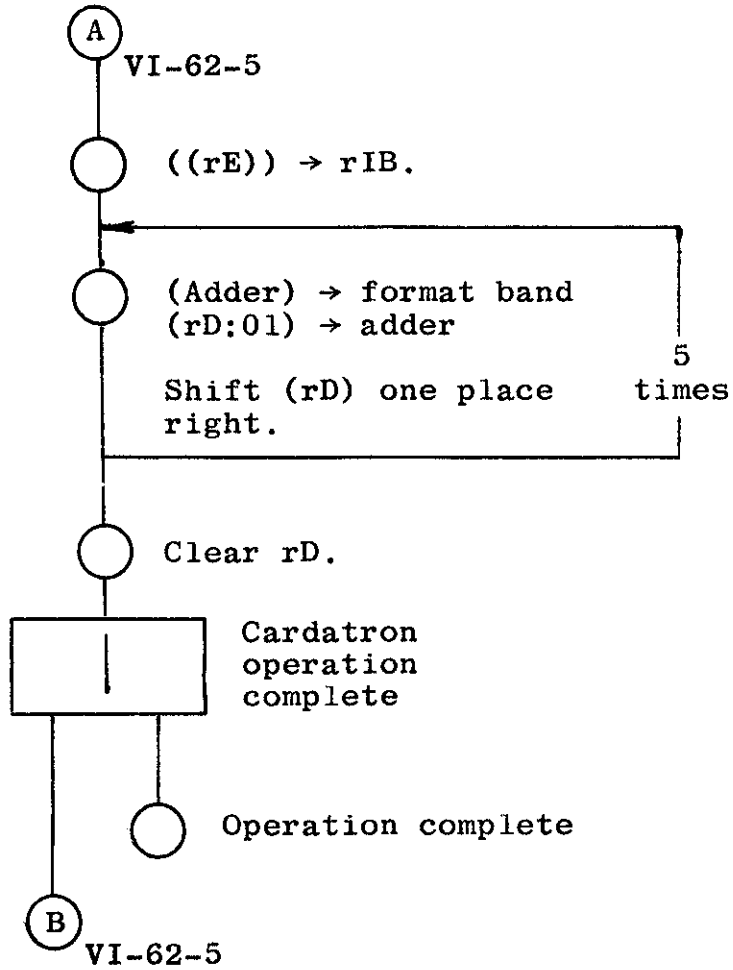
THE CARDATRON SYSTEM

Description of operation:

Flow chart:



Flow chart (continued):



THE CARDATRON SYSTEM

Register status:

Register name	Contents after execution of CRF.	Contents if digit-check ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	*														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	2	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	2	B[aaaa]
u	i	i	f	6	2	B[aaaa]										
u	i	i	f	6	2	B[aaaa]										
E	$B[aaaa] - 29$	*														

*Depends on when the impermissible configuration is detected. See flow chart.

Register name	Contents if non-existent-address or non-existent-Input-Unit ALARM STOP occurs.							
A	Unchanged							
R	Unchanged							
D	Cleared							
B	Unchanged							
P	$(rP)_b + 1$							
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	2	B[aaaa]
u	i	i	f	6	2	B[aaaa]		
E	$B[aaaa]$							

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD WRITE, FORMAT LOAD

Operation code: 63

Abbreviation: CWF

Instruction format:

Time (μ s):

\pm	1	2	3	4	5	6	7	8	9	0
\pm	u	i	i	f	Op	a	a	a	a	

Minimum:

fetch:	90
execute:	8545
total:	8635

Definitions:

Average:

- \pm : if \pm is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates Output Unit.
- ii: not relevant to the execution of this instruction.
- f: specifies which format band will be written.
- Op: operation code.
- aaaa: address of base of location from which is read the first word transferred to the format band specified by f.

fetch:	90
execute:	9960
total:	10050

Maximum:

fetch:	90
execute:	11940
total:	12030

Description of operation:

Summary:

Transfer to the format band specified by f--on Output Unit u--the contents of 29 core storage locations as follows: the contents of B[aaaa] occupy format band digit positions 1 through 11; (B[aaaa] - 1) occupy positions 12 through 22; ...; (B[aaaa] - 27) occupy positions 298 through 308; (B[aaaa] - 28:08) occupy positions 309 through 316.

The selection of the format band on which the editing control stream is written is accomplished as described in the table below:

f	Format Band
0,1	1
2,3	2
4,5	3
6,7	4
8,9	5

Flow chart:

See page VI-63-5.

Exceptional conditions:

1. Non-existent-address ALARM STOP. (See Remark 3, below.)
2. Non-existent-Output-Unit ALARM STOP.
3. Digit-check ALARM STOP. (See Remark 4, below.)

Remarks:

1. A format band digit position contains only two bits. Hence, a format band digit may be only one of 0, 1, 2, or 3. If an attempt is made to write as a format band digit some decimal digit greater than or equal to 4, the 4-bit or the 8-bit will be dropped when the decimal digit is transferred to the selected format band.

2. At the beginning of the Execute Phase of a CWF instruction B[aaaa] = (rC:04) is transferred to the E register. The address of the core storage location from which is read each successive word transferred to the format band is obtained by decreasing by 1 the contents of the E register after each word is read from storage.

Since the E register counts modulo the size of storage, care must be exercised in the choice of B[aaaa]. For example, suppose the Data Processor has 4000 words of core storage; suppose also that $B[aaaa] \leq 0028$: then some word will be read from location 0000. After this word is read, 1 will be subtracted from 0000, the contents of the E register, modulo 4000. The E register will then contain 3999.

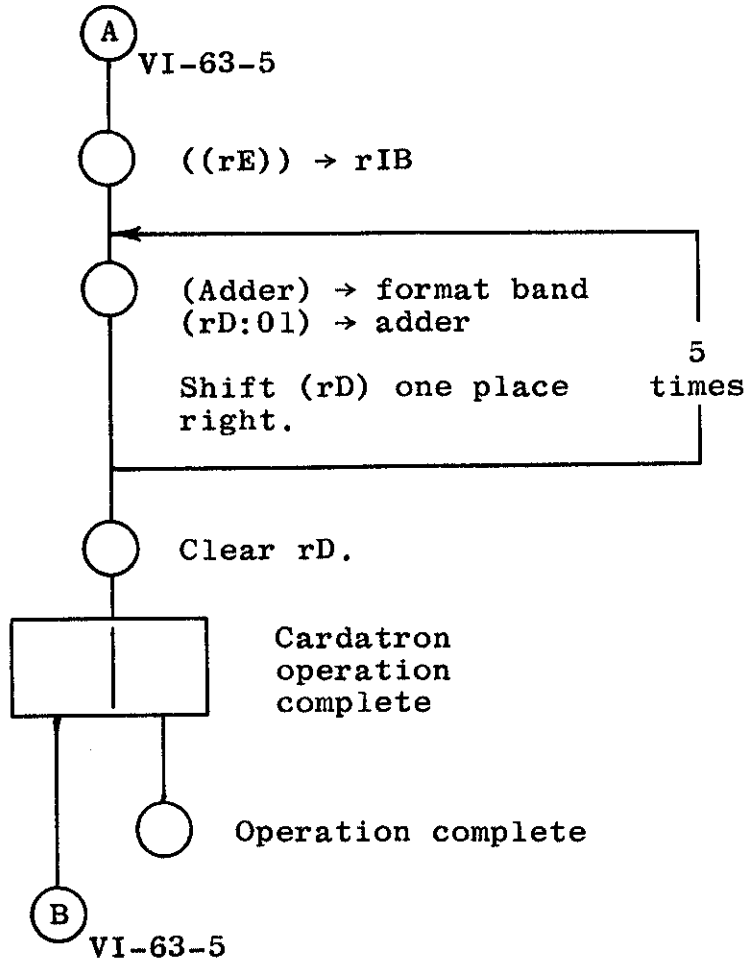
3. Examination of the flow chart will disclose that control is vested in the Cardatron--by virtue of the fact that the Cardatron "clock" is providing pulses for Data Processor operation, too--when a non-existent address is detected. The detection of the non-existent address inhibits the transfer of information in the Data Processor. But the Cardatron must complete its cycle, even though no information is transferred from the Data Processor to the buffer.

The ALARM STOP will occur after the Cardatron cycle is completed. It will appear that the designated format band had all 0's written on it.

4. The detection of an impermissible configuration occurs when (rIB:01) and (rD:01) are shifted to the right. Such an event can occur only after control is vested in the Cardatron (see the flow chart). The detection of the impermissible configuration immediately stops the transfer of information with the Data Processor, and from the Data Processor to the buffer. But Cardatron must complete its cycle of operation, even though no more information is transferred to the buffer.

The digit-check ALARM STOP will occur after the Cardatron cycle is completed. It will appear that the designated format band contains 0's following the last digit transferred from the Data Processor.

Flow chart (continued):



THE CARDATRON SYSTEM

Register status:

Register name	Contents after execution of CWF.	Contents if digit-check ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	*														
B	Unchanged	Unchanged														
P	$(rP)_b + 1$	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	3	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	3	B[aaaa]
u	i	i	f	6	3	B[aaaa]										
u	i	i	f	6	3	B[aaaa]										
E	$B[aaaa] - 29$	*														

*Depends on when the impermissible configuration is detected. See flow chart.

Register name	Contents if non-existent-address or non-existent-Output-Unit ALARM STOP occurs.							
A	Unchanged							
R	Unchanged							
D	Cleared							
B	Unchanged							
P	$(rP)_b + 1$							
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">f</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	f	6	3	B[aaaa]
u	i	i	f	6	3	B[aaaa]		
E	$B[aaaa]$							

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD READ INTERROGATE,
BRANCH

Operation code: 64

Abbreviation: CRI

Instruction format:

Time: (μs):

No branch:
Minimum:

fetch: 90
execute: 245
total: 335

± 1 2 3 4 5 6 7 8 9 0

±	u	i	i	i	Op	a	a	a	a
---	---	---	---	---	----	---	---	---	---

Average:

fetch: 90
execute: 265
total: 355

Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates Input Unit.
- iii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location of alternate instruction.

Maximum:

fetch: 90
execute: 285
total: 375

Branch:
Minimum:

fetch: 90
execute: 265
total: 355

Average:

fetch: 90
execute: 285
total: 375

Maximum:

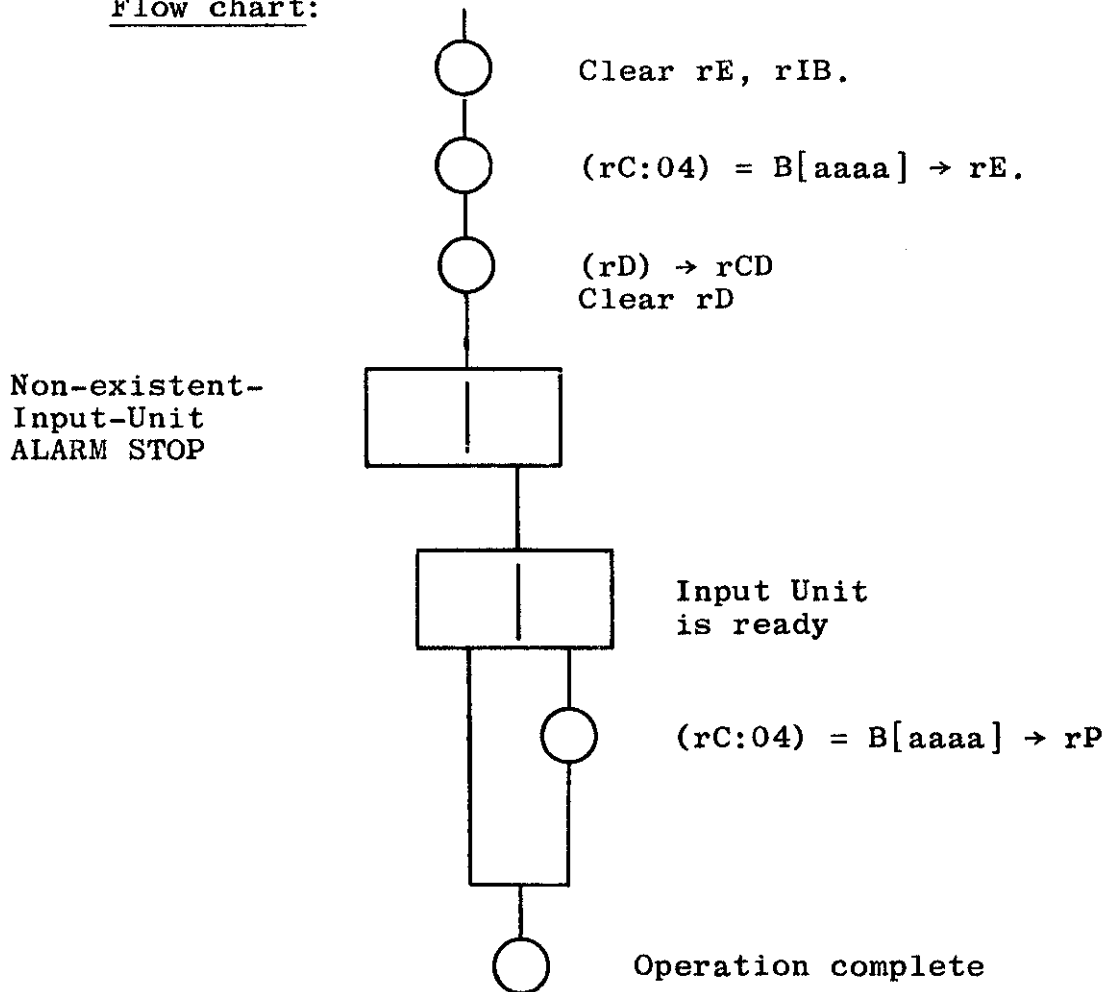
fetch: 90
execute: 305
total: 395

Description of operation:

Summary:

If Input Unit u is ready, transfer control to B[aaaa], that is, prepare to take the next instruction from B[aaaa]; otherwise, control continues in sequence.

Flow chart:



Exceptional conditions:

1. Non-existent-Input-Unit ALARM STOP.

Remarks:

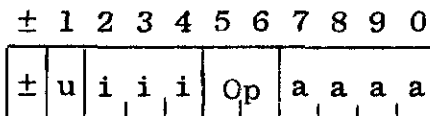
Register status:

Register name	Contents after execution of CRI.	Contents if non-existent-Input-Unit ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	Cleared														
B	Unchanged	Unchanged														
P	No branch: $(rP)_b + 1$ Branch: B[aaaa] ^b + 1	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	i	6	4	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	i	6	4	B[aaaa]
u	i	i	i	6	4	B[aaaa]										
u	i	i	i	6	4	B[aaaa]										
E	B[aaaa]	B[aaaa]														

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Operation name: CARD WRITE INTERROGATE,
BRANCH

Instruction format:



Definitions:

- ±: if ± is odd, B-register address-modification will occur; otherwise, there will be no such modification.
- u: designates Output Unit.
- iii: not relevant to the execution of this instruction.
- Op: operation code.
- aaaa: address of base of location of alternate instruction.

Operation code: 65

Abbreviation: CWI

Time (µs)

No branch:
Minimum:

fetch:	90
execute:	245
total:	335

Average:

fetch	90
execute:	265
total:	355

Maximum:

fetch:	90
execute:	285
total:	375

Branch:
Minimum:

fetch:	90
execute:	265
total:	355

Average:

fetch:	90
execute:	285
total:	375

Maximum:

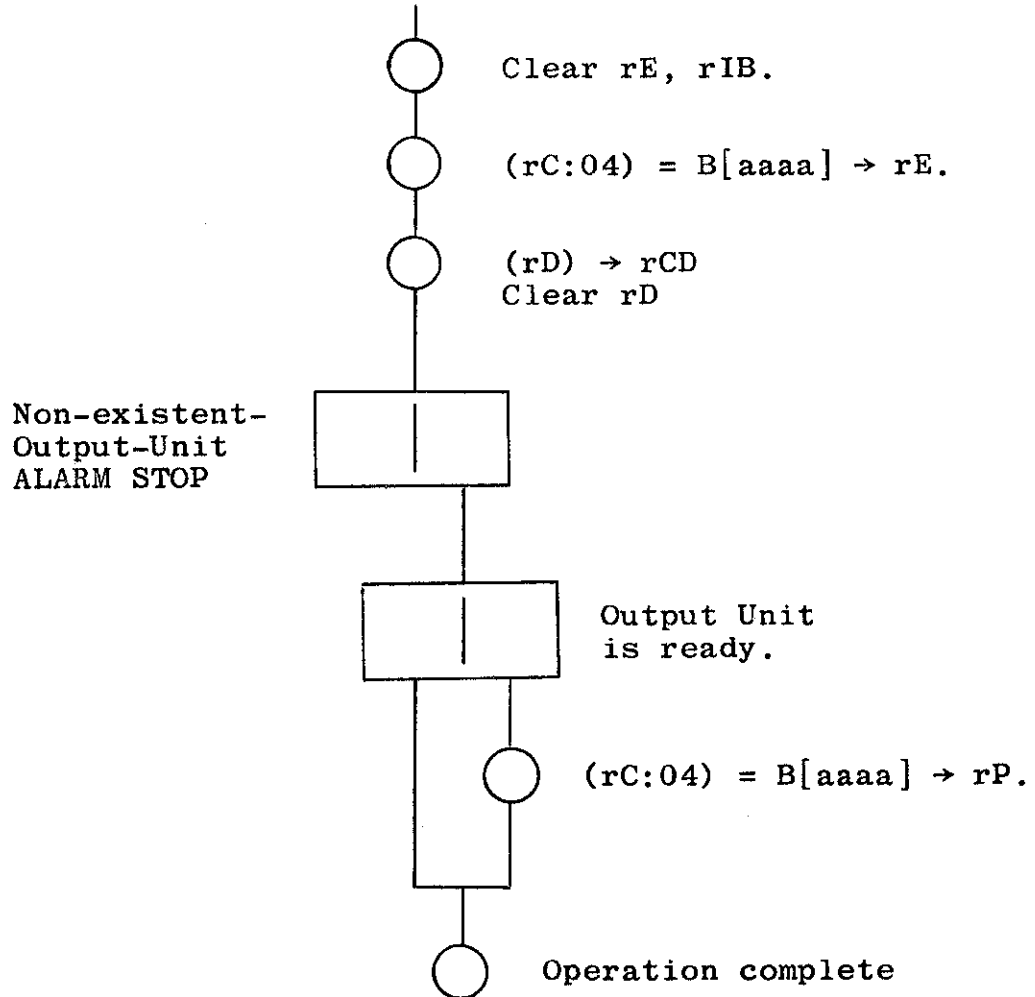
fetch:	90
execute:	305
total:	395

Description of operation:

Summary:

If Output Unit u is ready, transfer control to B[aaaa], that is, prepare to take the next instruction from B[aaaa]; otherwise, control continues in sequence.

Flow chart:



OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Exceptional conditions:

1. Non-existent-Output-Unit ALARM STOP

Remarks:

Register status:

Register name	Contents after execution of CWI.	Contents if non-existent-Output-Unit ALARM STOP occurs.														
A	Unchanged	Unchanged														
R	Unchanged	Unchanged														
D	Cleared	Cleared														
B	Unchanged	Unchanged														
P	No branch: $(rP)_b + 1$ Branch: B[aaaa]	$(rP)_b + 1$														
C	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	i	6	5	B[aaaa]	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">u</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">i</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">B[aaaa]</td> </tr> </table>	u	i	i	i	6	5	B[aaaa]
u	i	i	i	6	5	B[aaaa]										
u	i	i	i	6	5	B[aaaa]										
E	B[aaaa]	B[aaaa]														

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Appendix 1. A Summary of Operations in Operation-Code Order

Instruction Format				Operation		Reference
±	1234	56	7890	Abr	Name	
±	iiii	00	iiii	HLT	HALT	II-00
±	iiii	01	iiii	NOP	NO OPERATION	II-01
±	unnv	03	aaaa	PRD	PAPER-TAPE READ	V-03
±	uiiv	04	aaaa	PRB	PAPER-TAPE READ, BRANCH	V-04
±	unnv	05	aaaa	PRI	PAPER-TAPE READ, INVERSE FORMAT	V-05
±	unni	06	aaaa	PWR	PAPER-TAPE WRITE	V-06
±	uiii	07	aaaa	PWI	PAPER-TAPE WRITE, INTERROGATE BRANCH	V-07
±	iiii	08	iiii	KAD	KEYBOARD ADD	III-08
±	iiii 09	aaaa	aaaa	SPO	SUPERVISORY PRINT-OUT	III-09
±	iii0	10	aaaa	CAD	CLEAR ADD	II-10
±	iiil	10	aaaa	CAA	CLEAR ADD ABSOLUTE	II-10
±	iii0	11	aaaa	CSU	CLEAR SUBTRACT	II-11
±	iiil	11	aaaa	CSA	CLEAR SUBTRACT ABSOLUTE	II-11
±	iii0	12	aaaa	ADD	ADD	II-12
±	iiil	12	aaaa	ADA	ADD ABSOLUTE	II-12
±	iii0	13	aaaa	SUB	SUBTRACT	II-13
±	iiil	13	aaaa	SUA	SUBTRACT ABSOLUTE	II-13
±	iiii	14	aaaa	MUL	MULTIPLY	II-14
±	iiii	15	aaaa	DIV	DIVIDE	II-15
±	iiii	16	iiii	RND	ROUND	II-16
±	iiii	17	aaaa	EXT	EXTRACT	II-17
±	sLf0	18	aaaa	CFA	COMPARE FIELD A	II-18
±	sLf1	18	aaaa	CFR	COMPARE FIELD R	II-18
±	iiii	19	aaaa	ADL	ADD TO LOCATION	II-19
±	nnnn	20	aaaa	IBB	INCREASE B, BRANCH	II-20
±	nnnn	21	aaaa	DBB	DECREASE B, BRANCH	II-21
±	iiii 22	aaaa	aaaa	FAD	FLOATING ADD	II-22
±	iiil	22	aaaa	FAA	FLOATING ADD ABSOLUTE	II-22
±	iii0	23	aaaa	FSU	FLOATING SUBTRACT	II-23
±	iiil	23	aaaa	FSA	FLOATING SUBTRACT ABSOLUTE	II-23
±	iiii	24	aaaa	FMU	FLOATING MULTIPLY	II-24
±	iiii	25	aaaa	FDV	FLOATING DIVIDE	II-25
±	sLnn	26	aaaa	IFL	INCREASE FIELD LOCATION	II-26
±	sLnn	27	aaaa	DFL	DECREASE FIELD LOCATION	II-27
±	sLnn	28	aaaa	DLB	DECREASE FIELD LOCATION, LOAD B	II-28
±	inni	29	aaaa	RTF	RECORD TRANSFER	II-29
±	iiii	30	aaaa	BUN	BRANCH, UNCONDITIONALLY	II-30
±	iiii	31	aaaa	BOF	BRANCH, OVERFLOW	II-31
±	iiii	32	aaaa	BRP	BRANCH, REPEAT	II-32
±	iiin	33	aaaa	BSA	BRANCH, SIGN A	II-33
±	iii0	34	aaaa	BCH	BRANCH, COMPARISON HIGH	II-34
±	iiil	34	aaaa	BCL	BRANCH, COMPARISON LOW	II-34
±	iii0	35	aaaa	BCE	BRANCH, COMPARISON EQUAL	II-35
±	iiil	35	aaaa	BCU	BRANCH, COMPARISON UNEQUAL	II-35
±	sLnn	36	aaaa	BFA	BRANCH, FIELD A	II-36
±	sLnn	37	aaaa	BFR	BRANCH, FIELD R	II-37
±	uiii	38	aaaa	BCS	BRANCH, CONTROL SWITCH	III-38
±	sLf0	40	aaaa	STA	STORE A	II-40
±	sLf1	40	aaaa	STR	STORE R	II-40
±	sLf2	40	aaaa	STB	STORE B	II-40

A SUMMARY OF OPERATIONS IN OPERATION-CODE ORDER

Instruction Format				Operation		Reference
±	1234	56	7890	Abr	Name	
±	iiii	41	aaaa	LDR	LOAD R	II-41
±	iii0	42	aaaa	LDB	LOAD B	II-42
±	iiil	42	aaaa	LBC	LOAD B COMPLEMENT	II-42
±	iiin	43	iiii	LSA	LOAD SIGN A	II-43
±	iiii	44	aaaa	STP	STORE P	II-44
±	iiil	45	iiii	CLA	CLEAR A	II-45
±	iii2	45	iiii	CLR	CLEAR R	II-45
±	iii3	45	iiii	CAR	CLEAR A, R	II-45
±	iii4	45	iiii	CLB	CLEAR B	II-45
±	iii5	45	iiii	CAB	CLEAR A, B	II-45
±	iii6	45	iiii	CRB	CLEAR R, B	II-45
±	iii7	45	iiii	CLT	CLEAR A, R, B	II-45
±	iiii	46	aaaa	CLL	CLEAR LOCATION	II-46
±	iii0	48	iinn	SRA	SHIFT RIGHT A	II-48
±	iiil	48	iinn	SRT	SHIFT RIGHT A AND R	II-48
±	iii2	48	iinn	SRS	SHIFT RIGHT A WITH SIGN	II-48
±	iii0	49	iinn	SLA	SHIFT LEFT A	II-49
±	iiil	49	iinn	SLT	SHIFT LEFT A AND R	II-49
±	iii2	49	iinn	SLS	SHIFT LEFT A WITH SIGN	II-49
0	uhh0	50	aaaa	MTS	MAGNETIC-TAPE SEARCH	IV-50
4	uhh0	50	aaaa	MFS	MAGNETIC-TAPE FIELD SEARCH	IV-50
±	uhh4	50	iiii	MLS	MAGNETIC-TAPE LANE SELECT	IV-50
±	uhh8	50	iiii	MRW	MAGNETIC-TAPE REWIND	IV-50
±	uhh9	50	iiii	MDA	MAGNETIC-TAPE REWIND, DE-ACTIVATE	IV-50
0	uhhk	51	aaaa	MTC	MAGNETIC-TAPE SCAN	IV-51
4	uhhk	51	aaaa	MFC	MAGNETIC-TAPE FIELD SCAN	IV-51
±	univ	52	aaaa	MRD	MAGNETIC-TAPE READ	IV-52
±	univ	53	aaaa	MRR	MAGNETIC-TAPE READ, RECORD	IV-53
±	unkk	54	aaaa	MIW	MAGNETIC-TAPE INITIAL WRITE	IV-54
±	unii	55	aaaa	MIR	MAGNETIC-TAPE INITIAL WRITE, RECORD	IV-55
±	unkk	56	aaaa	MOW	MAGNETIC-TAPE OVERWRITE	IV-56
±	unii	57	aaaa	MOR	MAGNETIC-TAPE OVERWRITE, RECORD	IV-57
±	uni0	58	iiii	MPF	MAGNETIC-TAPE POSITION FORWARD	IV-58
±	unil	58	iiii	MPB	MAGNETIC-TAPE POSITION BACKWARD	IV-58
±	uni2	58	iiii	MPE	MAGNETIC-TAPE POSITION AT END OF INFORMATION	IV-58
±	uii0	59	aaaa	MIB	MAGNETIC-TAPE INTERROGATE, BRANCH	IV-59
±	uiil	59	aaaa	MIE	MAGNETIC-TAPE INTERROGATE END OF TAPE, BRANCH	IV-59
±	uiir	60	aaaa	CRD	CARD READ	VI-60
±	uicf	61	aaaa	CWR	CARD WRITE	VI-61
±	uiif	62	aaaa	CRF	CARD READ, FORMAT LOAD	VI-62
±	uiif	63	aaaa	CWF	CARD WRITE, FORMAT LOAD	VI-63
±	uiii	64	aaaa	CRI	CARD READ INTERROGATE, BRANCH	VI-64
±	uiii	65	aaaa	CWI	CARD WRITE INTERROGATE, BRANCH	VI-65

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Appendix 2. A Summary of Operations in Alphabetic Order

Instruction Format				Operation		Reference
±	1234	56	7890	Abr	Name	
±	iii0	12	aaaa	ADD	ADD	II-12
±	iiil	12	aaaa	ADA	ADD ABSOLUTE	II-12
±	iiii	19	aaaa	ADL	ADD TO LOCATION	II-19
±	iii0	35	aaaa	BCE	BRANCH, COMPARISON EQUAL	II-35
±	iii0	34	aaaa	BCH	BRANCH, COMPARISON HIGH	II-34
±	iiil	34	aaaa	BCL	BRANCH, COMPARISON LOW	II-34
±	iiil	35	aaaa	BCU	BRANCH, COMPARISON UNEQUAL	II-35
±	uiii	38	aaaa	BCS	BRANCH, CONTROL SWITCH	III-38
±	sLnn	36	aaaa	BFA	BRANCH, FIELD A	II-36
±	sLnn	37	aaaa	BFR	BRANCH, FIELD R	II-37
±	iiii	31	aaaa	BOF	BRANCH, OVERFLOW	II-31
±	iiii	32	aaaa	BRP	BRANCH, REPEAT	II-32
±	iiin	33	aaaa	BSA	BRANCH, SIGN A	II-33
±	iiii	30	aaaa	BUN	BRANCH, UNCONDITIONALLY	II-30
±	uuir	60	aaaa	CRD	CARD READ	VI-60
±	uiif	62	aaaa	CRF	CARD READ, FORMAT LOAD	VI-62
±	uiii	64	aaaa	CRI	CARD READ INTERROGATE, BRANCH	VI-64
±	uicf	61	aaaa	CWR	CARD WRITE	VI-61
±	uiif	63	aaaa	CWF	CARD WRITE, FORMAT LOAD	VI-63
±	uiii	65	aaaa	CWI	CARD WRITE INTERROGATE, BRANCH	VI-65
±	iiil	45	iiii	CLA	CLEAR A	II-45
±	iii5	45	iiii	CAB	CLEAR A, B	II-45
±	iii0	10	aaaa	CAD	CLEAR ADD	II-10
±	iiil	10	aaaa	CAA	CLEAR ADD ABSOLUTE	II-10
±	iii3	45	iiii	CAR	CLEAR A, R	II-45
±	iii7	45	iiii	CLT	CLEAR A, R, B	II-45
±	iii4	45	iiii	CLB	CLEAR B	II-45
±	iiii	46	aaaa	CLL	CLEAR LOCATION	II-46
±	iii2	45	iiii	CLR	CLEAR R	II-45
±	iii6	45	iiii	CRB	CLEAR R, B	II-45
±	iii0	11	aaaa	CSU	CLEAR SUBTRACT	II-11
±	iiil	11	aaaa	CSA	CLEAR SUBTRACT ABSOLUTE	II-11
±	sLf0	18	aaaa	CFA	COMPARE FIELD A	II-18
±	sLf1	18	aaaa	CFR	COMPARE FIELD R	II-18
±	nnnn	21	aaaa	DBB	DECREASE B, BRANCH	II-21
±	sLnn	27	aaaa	DFL	DECREASE FIELD LOCATION	II-27
±	sLnn	28	aaaa	DLB	DECREASE FIELD LOCATION, LOAD B	II-28
±	iiii	15	aaaa	DIV	DIVIDE	II-15
±	iiii	17	aaaa	EXT	EXTRACT	II-17
±	iii0	22	aaaa	FAD	FLOATING ADD	II-22
±	iiil	22	aaaa	FAA	FLOATING ADD ABSOLUTE	II-22
±	iiii	25	aaaa	FDV	FLOATING DIVIDE	II-25
±	iiii	24	aaaa	FMU	FLOATING MULTIPLY	II-24
±	iii0	23	aaaa	FSU	FLOATING SUBTRACT	II-23
±	iiil	23	aaaa	FSA	FLOATING SUBTRACT ABSOLUTE	II-23
±	nnnn	20	aaaa	IBB	INCREASE B, BRANCH	II-20
±	sLnn	26	aaaa	IFL	INCREASE FIELD LOCATION	II-26
±	iiii	00	iiii	HLT	HALT	II-00

A SUMMARY OF OPERATIONS IN ALPHABETIC ORDER

Instruction Format	Operation		Reference
	Abr	Name	
± 1234 56 7890			
± iiii 08 iiii	KAD	KEYBOARD ADD	III-08
± iii0 42 aaaa	LDB	LOAD B	II-42
± iiii 42 aaaa	LBC	LOAD B COMPLEMENT	II-42
± iiii 41 aaaa	LDR	LOAD R	II-41
± iiii 43 iiii	LSA	LOAD SIGN A	II-43
4 uhhk 51 aaaa	MFC	MAGNETIC-TAPE FIELD SCAN	IV-51
4 uhh0 50 aaaa	MFS	MAGNETIC-TAPE FIELD SEARCH	IV-50
± unkk 54 aaaa	MIW	MAGNETIC-TAPE INITIAL WRITE	IV-54
± unii 55 aaaa	MIR	MAGNETIC-TAPE INITIAL WRITE, RECORD	IV-55
± uiio 59 aaaa	MIB	MAGNETIC-TAPE INTERROGATE, BRANCH	IV-59
± uiil 59 aaaa	MIE	MAGNETIC-TAPE INTERROGATE END OF TAPE, BRANCH	IV-59
± uhh4 50 iiii	MLS	MAGNETIC-TAPE LANE SELECT	IV-50
± unkk 56 aaaa	MOW	MAGNETIC-TAPE OVERWRITE	IV-56
± unii 57 aaaa	MOR	MAGNETIC-TAPE OVERWRITE, RECORD	IV-57
± uii2 58 iiii	MPE	MAGNETIC-TAPE POSITION AT END OF INFORMATION	IV-58
± unil 58 iiii	MPB	MAGNETIC-TAPE POSITION BACKWARD	IV-58
± unio 58 iiii	MPF	MAGNETIC-TAPE POSITION FORWARD	IV-58
± univ 52 aaaa	MRD	MAGNETIC-TAPE READ	IV-52
± univ 53 aaaa	MRR	MAGNETIC-TAPE READ, RECORD	IV-53
± uhh8 50 iiii	MRW	MAGNETIC-TAPE REWIND	IV-50
± uhh9 50 iiii	MDA	MAGNETIC-TAPE REWIND, DE-ACTIVATE	IV-50
0 uhhk 51 aaaa	MTC	MAGNETIC-TAPE SCAN	IV-51
0 uhh0 50 aaaa	MTS	MAGNETIC-TAPE SEARCH	IV-50
± iiii 14 aaaa	MUL	MULTIPLY	II-14
± iiii 01 iiii	NOP	NO OPERATION	II-01
± unnv 03 aaaa	PRD	PAPER-TAPE READ	V-03
± uiiv 04 aaaa	PRB	PAPER-TAPE READ, BRANCH	V-04
± unnv 05 aaaa	PRI	PAPER-TAPE READ, INVERSE FORMAT	V-05
± unni 06 aaaa	PWR	PAPER-TAPE WRITE	V-06
± uiii 07 aaaa	PWI	PAPER-TAPE WRITE INTERROGATE, BRANCH	V-07
± inni 29 aaaa	RTF	RECORD TRANSFER	II-29
± iiii 16 iiii	RND	ROUND	II-16
± iii0 49 iinn	SLA	SHIFT LEFT A	II-49
± iiii 49 iinn	SLT	SHIFT LEFT A AND R	II-49
± iii2 49 iinn	SLS	SHIFT LEFT A WITH SIGN	II-49
± iii0 48 iinn	SRA	SHIFT RIGHT A	II-48
± iiii 48 iinn	SRT	SHIFT RIGHT A AND R	II-48
± iii2 48 iinn	SRS	SHIFT RIGHT A WITH SIGN	II-48
± sLf0 40 aaaa	STA	STORE A	II-40
± sLf2 40 aaaa	STB	STORE B	II-40
± iiii 44 aaaa	STP	STORE P	II-44
± sLf1 40 aaaa	STR	STORE R	II-40
± iii0 13 aaaa	SUB	SUBTRACT	II-13
± iiii 13 aaaa	SUA	SUBTRACT ABSOLUTE	II-13
± inri 09 aaaa	SPO	SUPERVISORY PRINT-OUT	III-09

innv

ALPHANUMERIC CODES AND THEIR REPRESENTATION

Appendix 3. Alphanumeric Codes and their Representation

Symbol		Computer and Magnetic Tape	Code							Cardatron Buffer	Punched Card
			Paper Tape								
DATATRON	046		X	0	Parity Check	8	4	2	1		
(space)	(space)	00			✓					0 - 0	(blank)
.	.	03	X	0		8		2	1	1 - 11	12 8-3
☐	☐	04	X	0	✓	8	4			1 - 12	12 8-4
&	&	10	X	0	✓					1 - 0	12
\$	\$	13	X		✓	8		2	1	2 - 11	11 8-3
*	*	14	X			8	4			2 - 12	11 8-4
-	-	20	X							2 - 0	11
/	/	21		0	✓				1	4 - 1	0 1
,	,	23		0	✓	8		2	1	4 - 11	0 8-3
%	%	24		0		8	4			4 - 12	0 8-4
#	#	33				8		2	1	0 - 11	8-3
@	@	34			✓	8	4			0 - 12	8-4
A	A	41	X	0					1	1 - 1	12 1
B	B	42	X	0				2		1 - 2	12 2
C	C	43	X		✓			2	1	1 - 3	12 3
D	D	44	X	0			4			1 - 4	12 4
E	E	45	X	0	✓		4		1	1 - 5	12 5
F	F	46	X	0	✓		4	2		1 - 6	12 6
G	G	47	X	0			4	2	1	1 - 7	12 7

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Symbol		Computer and Magnetic Tape	Code							Cardatron Buffer	Punched Card	
			Paper Tape Channel									
DATATRON	046		X	0	Parity Check	8	4	2	1			
H	H	48	X	0		8				1 - 8	12	8
I	I	49	X	0	✓	8			1	1 - 9	12	9
J	J	51	X		✓				1	2 - 1	11	1
K	K	52	X		✓				2	2 - 2	11	2
L	L	53	X						2	2 - 3	11	3
M	M	54	X		✓		4			2 - 4	11	4
N	N	55	X				4		1	2 - 5	11	5
O	O	56	X				4	2		2 - 6	11	6
P	P	57	X		✓		4	2	1	2 - 7	11	7
Q	Q	58	X		✓	8				2 - 8	11	8
R	R	59	X			8			1	2 - 9	11	9
S	S	62		0	✓				2	4 - 2	0	2
T	T	63		0					2	4 - 3	0	3
U	U	64		0	✓		4			4 - 4	0	4
V	V	65		0			4		1	4 - 5	0	5
W	W	66		0			4	2		4 - 6	0	6
X	X	67		0	✓		4	2	1	4 - 7	0	7
Y	Y	68		0	✓	8				4 - 8	0	8
Z	Z	69		0		8			1	4 - 9	0	9
0 (zero)	0 (zero)	80		0						4 - 0	0	0

ALPHANUMERIC CODES AND THEIR REPRESENTATION

Symbol		Computer and Magnetic Tape	Code							Cardatron Buffer	Punched Card
			Paper Tape Channel								
X	0		Parity Check	8	4	2	1				
DATATRON	046										
1	1	81							1	0 - 1	1
2	2	82						2		0 - 2	2
3	3	83			/			2	1	0 - 3	3
4	4	84					4			0 - 4	4
5	5	85			/		4		1	0 - 5	5
6	6	86			/		4	2		0 - 6	6
7	7	87					4	2	1	0 - 7	7
8	8	88				8				0 - 8	8
9	9	89			/	8			1	0 - 9	9
TAB	SKIP	26		0	/	8	4	2		0 - 14	0 8-6
CARRIAGE RETURN	CR	16	X		/	8	4	2		2 - 14	11 8-6
FORM OUT	PI 6	15	X		/	8	4		1	↑ Not assigned ↓	11 8-5
END-OF-WORD	PI 7	35				8	4		1		8-5
BLANK	SP 1	02	X	0	/	8		2			12 8-2
TAPE FEED	TAPE FEED	*	X	0	/	8	4	2	1		12 8-7
↑ Not assigned	END CARD 1	36				8	4	2			8-6
↓	END CARD 2	27		0		8	4	2	1	0 8-7	

* Not represented.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Symbol		Computer and Magnetic Tape	Code							Cardatron Buffer	Punched Card
DATATRON	046		Paper Tape Channel								
			X	0	Parity Check	8	4	2	1		
Not assigned ↑ ↓	CORRECT TAB	37			✓	8	4	2	1	Not assigned ↑ ↓	8-7
	ERROR	17	X			8	4	2	1		11 8-7
	PI 1	32			✓	8		2			8-2
	PI 2	12	X			8		2			11 8-2
	PI 3	22		0		8		2			0 8-2
	PI 4	25		0	✓	8	4		1		0 8-5
	PI 5	05	X	0		8	4		1		12 8-5
	SP 2	06	X	0		8	4	2			12 8-6

Appendix 4. A Glossary of Terms.

ABSOLUTE VALUE

The absolute value of a number, n , is the number which results from making the sign of n positive. For example, if $n = +6$, the absolute value of n is $+6$; if $n = -7$, the absolute value of n is $+7$. In symbols, we would write: $|+6| = +6$; $|-7| = +7$.

In general, then, $|n| = +n$, if n is greater than or equal to zero; and $|n| = -n$, if n is less than zero.

ADDRESS OF BASE OF LOCATION . . .

Because the address part of an instruction word can be augmented by the contents of the B register as the instruction is fetched from storage to the C register, that address frequently is not the address of the location referenced by the instruction which is executed. In such cases the address part of the instruction in storage is the base on which is constructed--by addition of the contents of the B register--the actual address used by the instruction which is executed. In this context it is convenient to call the address part of the stored instruction the "address of the base of the location of"

B-REGISTER ADDRESS-MODIFICATION

The B register and the address part of a word are four digits long. B-register address-modification is the addition of the contents of the B register and the address part of a word; only the four low-order digits of the sum are retained, however. Moreover, there can be no carry into digit-position 5 in the word which is modified.

FIELD

A field is a set of contiguous digit-positions.

PARTIAL-WORD FIELD

A field, all of whose digit positions lie in the same word.

Appendix 5. A Glossary of Symbols

- (aaaa): Contents of location whose address is aaaa.
- (aaaa)_a: The subscript "a" identifies the contents after execution.
- (aaaa)_b: The subscript "b" identifies the contents before execution.
- [aaaa]: The enclosure, in square brackets, of an address, for example, is an indication that the address may be or will be modified during program execution.
- <aaaa>: It is convenient to distinguish B-register address-modification from other types. Although the contents of the sign-digit position of an instruction word specify such modification, the sign-digit is not sufficiently distinctive on a handwritten coding sheet.
- B[aaaa]: This symbol is used only in this book and only in the following context: in the description of each operation it is necessary to indicate that B-register address-modification may or may not occur. The context is "... the contents of aaaa or the contents of the location whose address is aaaa augmented by the contents of the B-register, in case B-register address-modification were intended." This is a very clumsy sentence, and one which ought not to appear more than once, anywhere: in this paragraph is its only occurrence - anywhere. Throughout this book, B[aaaa] is a substitute for the sentence.

rA, etc.: A lower-case "r" will precede the names of the various registers in the control and arithmetic units in order to distinguish them. rA, for example, is the A register.

rA:sL: In order to specify a partial-word field, three parameters are required: the location of the word containing the field, and the two boundaries of the field. In the example shown, the location is the A register. s designates the position of the right-hand boundary, i.e., the low-order digit of the partial-word field. The left-hand boundary is defined by specifying that there are L digits in the partial-word field. For example, rA:04 specifies the address part of the A register; (rA:04) is the address itself, i.e., the four digits in the specified field. And rD:±1 is the sign-digit position in the D register.

(rA:±1)/1: This symbol specifies the one bit of the A register's sign digit. Similarly, (rA:±1)/2 means the two bit of the A register's sign digit. Etc.

± 1 2 3 4 5 6 7 8 9 0 : This diagram identifies the digit positions in a computer word. The digit position identified by the symbol "±" is called the sign-digit position.



The contents of digit positions 1, 2, 3, and 4 are sometimes called control digits; the contents of digit positions 5 and 6 comprise the operation code; and the remaining four digit-positions comprise the address part of the word, when these terms are meaningful.

A GLOSSARY OF SYMBOLS

→: "(rP) → rE" means "the contents of the P register go into (i.e., replace the contents of) the E register." Alternatively one may write "(rP) → (rE)" which is to be read "the contents of the P register replace the contents of the E register".


⇒ "(rB) ⇒ Adder → (rC:04)" means "the contents of the B register go through the adder and replace the contents of the address part of the C register."

|nnnn|: the absolute value of nnnn. See Appendix A4, A Glossary of Terms, for a definition of absolute value.

The following symbols are used in the flow charts, which elaborate on the execution of the DATATRON 220 operations:

Ⓐ II-24-5: This is connector A. If it is an output connector, the corresponding input connector will be found on page II-24-5; if it is an input connector, the corresponding output connector will be found on page II-24-5.

○ : This indicates action. All of the action associated with this symbol is specified beside the symbol. All of the action associated with the symbol occurs simultaneously.

 : This is the symbol for a branch point; it is used to indicate alternative courses of action.

A SUMMARY OF EXECUTION TIMES IN OPERATION-ABBREVIATION ORDER

Appendix 6

Abr	Op Code	Total time	(Fetch + Execute)	(μ s) / Remarks
ADA	12	185	/ No decompement	
		245	/ Decompement	
ADD	12	185	/ No decompement	
		245	/ Decompement	
ADL	19	255	/ No decompement	
		315	/ Decompement	
BCE	35	105	/ No branch	
		125	/ Branch	
BCH	34	105	/ No branch	
		125	/ Branch	
BCL	34	105	/ No branch	
		125	/ Branch	
BCS	38	105	/ No branch	
		125	/ Branch	
BCU	35	105	/ No branch	
		125	/ Branch	
BFA	36	165	/ No branch	
		185	/ Branch	
BFR	37	165	/ No branch	
		185	/ Branch	
BOF	31	105	/ No branch	
		125	/ Branch	
BRP	32	105	/ No branch	
		125	/ Branch	
BSA	33	175	/ No branch	
		195	/ Branch	
BUN	30	125		
CAA	10	185		
CAB	45	100		
CAD	10	185		
CAR	45	100		

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Abr	Op Code	Total time (Fetch + Execute) (μs) / Remarks	
CFA	18	240	
CFR	18	240	
CLA	45	100	
CLB	45	100	
CLL	46	115	
CLR	45	100	
CLT	45	100	
CRB	45	100	
CRD	60	5855	/ Minimum
		7270	/ Average
		9110	/ Maximum
CRF	62	8635	/ Minimum
		10050	/ Average
		12030	/ Maximum
CRI	64	335	/ Minimum, no branch
		355	/ Average, no branch
		375	/ Maximum, no branch
		355	/ Minimum, branch
		375	/ Average, branch
		395	/ Maximum, branch
CSA	11	185	
CSU	11	185	
CWF	63	8635	/ Minimum
		10050	/ Average
		12030	/ Maximum
CWI	65	335	/ Minimum, no branch
		355	/ Average, no branch
		375	/ Maximum, no branch
		355	/ Minimum, branch
		375	/ Average, branch
		395	/ Maximum, branch
CWR	61	8635	/ Minimum
		10050	/ Average
		12030	/ Maximum

A SUMMARY OF EXECUTION TIMES IN OPERATION-ABBREVIATION ORDER

Abr	Op Code	Total time (Fetch + Execute) (μs) / Remarks	
DBB	21	130	/ No branch
		150	/ Branch
DFL	27	250	
DIV	15		/ Quotient is (rA) _a
		180	/ Minimum, overflow
		1285	/ Minimum, (rA) _a = ± 0909 09 0909
		3985	/ Average, (rA) _a = ± 5555 55 5555
		6685	/ Maximum, (rA) _a = ± 9090 90 9090
/ Execution time--exclusive of fetch time--may be calculated with the aid of the following formula:			
$T = 3895 + 60 \sum_{k=0}^4 [(rA:2k+1,1)_a - (rA:2k,1)_a]$			
DLB	28	250	
EXT	17	235	
FAA	22	280	/ Sum = 0
		325	/ Minimum, underflow
		410	/ Maximum, underflow
		215	/ Sum ≠ 0, no decompement, minimum
		360	/ Sum ≠ 0, no decompement, maximum
		280	/ Sum ≠ 0, decompement, minimum
FAD	22	415	/ Sum ≠ 0, decompement, maximum
		280	/ Sum = 0
		325	/ Minimum, underflow
		410	/ Maximum, underflow
		215	/ Sum ≠ 0, no decompement, minimum
		360	/ Sum ≠ 0, no decompement, maximum
FDV	25	280	/ Sum ≠ 0, decompement, minimum
		415	/ Sum ≠ 0, decompement, maximum
		175	/ Quotient or divisor = 0
		260	/ Underflow
		175	/ Overflow, minimum
		6775	/ Overflow, maximum
4075	/ Quotient ≠ 0, average		
6775	/ Quotient ≠ 0, maximum		

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Abr	Op Code	Total time (Fetch + Execute) (μ s) / Remarks	
FMU	24	175	/ Product = 0
		255	/ Overflow
		255	/ Underflow, minimum
		2915	/ Underflow, maximum
		1735	/ Product \neq 0, average
		2915	/ Product \neq 0, maximum
FSA	23	280	/ Difference = 0
		325	/ Underflow, minimum
		410	/ Underflow, maximum
		215	/ Difference \neq 0, no decompement, minimum
		360	/ Difference \neq 0, no decompement, maximum
		280	/ Difference \neq 0, decompement, minimum
		415	/ Difference \neq 0, decompement, maximum
FSU	23	280	/ Difference = 0
		325	/ Underflow, minimum
		410	/ Underflow, maximum
		215	/ Difference \neq 0, no decompement, minimum
		360	/ Difference \neq 0, no decompement, maximum
		280	/ Difference \neq 0, decompement, minimum
		415	/ Difference \neq 0, decompement, maximum
HLT	00	100	
IBB	20	130	/ No branch
		150	/ Branch
IFL	26	250	
KAD	03		/ Manual operation
LBC	42	180	
LDB	42	180	
LDR	41	175	
LSA	43	105	
MDA	50	195	/ Data Processor time, unit ready
MFC	51	245	/ Data Processor time, unit ready
MFS	50	245	/ Data Processor time, unit ready
MIB	59	100	/ No branch, TCU not ready
		230	/ No branch, TCU ready, TSU not ready
		250	/ Branch
MIE	59	100	/ No branch, TCU not ready
		230	/ No branch, TCU ready, TSU not ready
		250	/ Branch

A SUMMARY OF EXECUTION TIMES IN OPERATION-ABBREVIATION ORDER

Abr	Op Code	Total time (Fetch + Execute) (μs) / Remarks																							
MIR	55	160	/ Fetch + setup; see Section IV for other details																						
MIW	54	160	/ Fetch + setup; see Section IV for other details																						
MLS	50	195	/ Data Processor time, unit ready																						
MOR	57	160	/ Fetch + setup; see Section IV for other details																						
MOW	56	160	/ Fetch + setup; see Section IV for other details																						
MPB	58	160	/ Fetch + setup; see Section IV for other details																						
MPE	58	160	/ Fetch + setup; see Section IV for other details																						
MPF	58	160	/ Fetch + setup; see Section IV for other details																						
MRD	52	160	/ Fetch + setup; see Section IV for other details																						
MRR	53	160	/ Fetch + setup; see Section IV for other details																						
MRW	50	195	/ Data Processor time, unit ready																						
MTC	51	245	/ Data Processor time, unit ready																						
MTS	50	245	/ Data Processor time, unit ready																						
MUL	14		/ Multiplier is (rA) _b																						
		230	/ Minimum, (rA) _b = ± 0000 00 0000																						
		2095	/ Average, (rA) _b = ± 0123 45 6789																						
		3480	/ Maximum, (rA) _b = ± 5555 55 5555																						
			/ Execution time--exclusive of fetch time--may be calculated with the aid of the following formula:																						
$T = 90 + 5 \sum_{k=0}^9 M_k,$																									
where M _k assumes the values shown in the table below:																									
<table border="1"> <thead> <tr> <th>(rA:k1)</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> </tr> </thead> <tbody> <tr> <td>M_k</td> <td>1</td> <td>14</td> <td>27</td> <td>40</td> <td>53</td> <td>66</td> <td>65</td> <td>52</td> <td>39</td> <td>26</td> </tr> </tbody> </table>				(rA:k1)	0	1	2	3	4	5	6	7	8	9	M _k	1	14	27	40	53	66	65	52	39	26
(rA:k1)	0	1	2	3	4	5	6	7	8	9															
M _k	1	14	27	40	53	66	65	52	39	26															
NOP	01	100																							
PRB	04		/ Photoreader speed, nominally 1000 characters per second																						
PRD	03		/ Photoreader speed, nominally 1000 characters per second																						

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Abr	Op Code	Total time (Fetch + Execute) (μ s) / Remarks	
PRI	05	/ Photoreader speed, nominally 1000 characters per second	
PWI	07	105	/ No branch
		125	/ Branch
PWR	06	/ Punch speed, nominally 60 characters per second / Printer speed, nominally 7 - 10 characters per second	
RND	16	105	/ (rR:11) < 5
		165	/ (rR:11) \geq 5
RTF	29	130 + 60(nn)	/ 01 \leq nn \leq 99
		6130	/ nn = 00
SLA	49	160 - 5N ₁	/ nn \equiv N ₁ , mod 10
SLS	49	160 - 5N ₁	/ nn \equiv N ₁ , mod 10
SLT	49	210 - 5N ₂	/ nn \equiv N ₂ , mod 20
SPC	09	/ Printer speed, nominally 7 - 10 characters per second	
SRA	48	110 + 5N	/ nn \equiv N, mod 20
SRS	48	110 + 5N	/ nn \equiv N, mod 20
SRT	48	110 + 5N	/ nn \equiv N, mod 20
STA	40	190	
STB	40	190	
STP	44	185	
STR	40	190	
SUA	13	185	/ No deocomplement
		245	/ Deocomplement
SUB	13	185	/ No deocomplement
		245	/ Deocomplement

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Errata Sheet 1

Page/Date	Location	Description
I-Intro-2; 9/1/57		This page is improperly identified as I-1-2.
II-Intro-1; 9/1/57	Line 13*	Should read "...the Computer; however,..."
II-Intro-2; 9/1/57	Line 13	Delete the clause "the part number RA2AH".
II-Intro-5; 9/1/57	Line 15	Should read "greater than or equal to -50 and less than or equal to 0..."
II-Intro-6; 9/1/57	Line 5	Should read " $10^{-51} < n < 10^{+49}$..."
II-Intro-9; 9/1/57	Line 12	Should read "FIELD SCAN is to be executed."
II-Intro-12; 9/1/57	Line 19	Should read "The CD register is a ten-digit-plus-sign-digit-position register..."
II-Intro-13; 9/1/57	Line 16	After the word "parallel;" insert the clause "all 16 bits of the P register are transferred to the E register in parallel;"
II-Intro-13; 9/1/57	Line 20	Instead of "four" read "three".
II-Intro-14; 9/1/57	Line 20	After the word "parallel" insert "from core storage."
II-Intro-15; 9/1/57	Line 2	Should read "Clear rE, rIB."
II-Intro-15; 9/1/57	Line 6	Delete the asterisk (*) above P
II-Intro-15; 9/1/57	Line 19	After "Restore one bit of rD:±1" insert "Restore one bit of rIB:±1"
II-Intro-19; 9/1/57	Line 9	After "keyboard (manual)" insert "Magnetic-Tape Storage Unit"

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Page/Date	Location	Description
II-Intro-23; 9/1/57		The register on the left should be labelled "IB register;" the register to the right of the IB register should be labelled "D register."
II-Intro-24; 9/1/57	Line 4*	Instead of "s > L + 1" read "L > s + 1".
II-Intro-24; 9/1/57	Line 9	Delete "or either input to the adder".
II-10-3; 9/1/57	Line 12	Add Remark 2: "CAD loads the A register with a word exactly as the word appears in storage."
II-13-3; 9/1/57	Line 5	Instead of "v = 1." read " <u>v = 1:</u> ".
II-13-3; 9/1/57	Line 12	Should read "...can cause arithmetic over-..."
II-14-4; 9/1/57	Line 5	Should read "(rC:04) = B[aaaa] → rE."
II-14-4; 9/1/57	Line 21	Should read "Set (rA:±1)/1 and..."
II-15-4; 9/1/57	Between line 6 and line 7	Insert "Set A sign-indicator equal to (rA:±1)/1."
II-15-4; 9/1/57	Lines 21 and 22	Reverse registers: rA replaces rR and rR replaces rA.
II-19-4; 9/1/57	Line 7	Add Remark 2: "The arithmetic operation of ADL is essentially the same as that of ADD."
II-20-3; 9/1/57		Add Remark 1: "The following statement embodies a method for determining whether branching will occur: if $(rB)_a \geq (rB)_b$, branching will occur: if $(rB)_a < (rB)_b$, control continues in sequence."

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Errata Sheet 1

Page/Date	Location	Description
II-21-3; 9/1/57		Add Remark 1: "The following statement embodies a method for determining whether branching will occur: if $(rB)_a \leq (rB)_b$, branching will occur: if $(rB)_a > (rB)_b$, control continues in sequence."
II-26-3; 9/1/57	Line 4*	Should read: "... If arithmetic overflow occurs - that is, if the result exceeds the capacity of the specified partial-word field - set the OVERFLOW Indicator to "on."
II-26-3; 9/1/57	Line 15	Instead of "rC:21 = 0" read "(rC:21) = 0."
II-26-7; 9/1/57		Add Example 10: $(rC) = 0400\ 26\ 3000$ $(3000)_b = -1233\ 00\ \underline{2914}$ $(rD)_a = (3000)_a = -1233\ 00\ \underline{2914}$
II-27-3; 9/1/57	Line 4	Should read: "... If underflow occurs - that is, if $(B[aaaa]:sL)_a > (B[aaaa]:sL)_b$ - set the..."
II-27-7; 9/1/57	Line 5	Should read " $(rD)_a = (0900)_a = \underline{0\ 5429\ 90\ 5432}$ "
II-27-7; 9/1/57		Add Example 7: $(rC) = 0400\ 27\ 0900$ $(0900)_b = +2345\ 67\ \underline{1212}$ $(rD)_a = (0900)_a = +2345\ 67\ \underline{1212}$ REPEAT Indicator is set "on."
II-33-4; 9/1/57	Line 11	The contents of the C register should be <code>iiin 33 B[aaaa]</code> .
II-40-5; 9/1/57		Add Remark 3: If, for example, the STORE A variation is being executed with $f = 1$ (partial-word selection), then $(rA:sL) \rightarrow (B[aaaa]:sL)$. The remainder of $(B[aaaa])$ is unaltered.

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Page/Date	Location	Description
II-41-4; 9/1/57	Line 4*	Instead of "LR" read "LDR".
V-Intro-4; 9/1/57	Line 7	Should read "The appearance of any digit, except 2, in the..."
V-03-3; 9/1/57	Line 6	Add the sentence: "If v = 8 or 9, designated input will be B-register address-modified."
V-03-3; 9/1/57	Line 14	Should read: "If v = 1:"
V-03-6; 9/1/57	Lower left side of page	The decision box should be as follows: <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;"> $(rD:\pm 1)/2 = 1$ $(rD:\pm 1)/4 = 1$ $(rC:41) = 1$ </div>
V-04-4; 9/1/57	Line 19	The left side of the decision box should be " $(rD:\pm 1) \neq 2$ "
A3-1; 9/1/57		The paper-tape code for "C" should show a punch in channel "0"

*Count every line of type, including heading.

Errata Sheet 2

Page/Date	Location	Description
		Wherever the term "computer" or "digital computer" appears throughout the text, replace it by the term "Data Processor."
II-Intro-2; 9/1/57	Line 12*	Should read: "0 5941 51 4148 may represent the number +5941 51 4148, the noun RAJAH,"
II-Intro-6; 9/1/57	Lines 8, 9	Delete the phrase "contains no high order zeros." Change the lines to read: "adjusted so that the highest-order digit of the mantissa is different from zero."
II-Intro-6; 9/1/57	Line 16	Delete the word "causes." After the word "operation" insert "would have caused."
II-Intro-8; 9/1/57	Following Line 18	Insert the following paragraph: The E register is used as a counter during the execution of certain Cardatron instructions. When it is so used it is capable of counting <u>only</u> modulo the number of words in storage. For example, if there are only 6000 words of core storage, the E register counts modulo 6000. (See Remark 5, page VI-60-4, for additional details.)
II-Intro-9; 9/1/57	Line 8	Delete the phrase "and disconnects the input device from the computer." Place a semicolon after "ter." Following the semicolon, insert the phrase "simultaneously, the input device receives a signal that the execution of the input instruction is completed."

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Page/Date	Location	Description
II-Intro-15; 9/1/57	Connector P	Change the reference to page VI-60-5; it should be page VI-60-10.
II-Intro-18; 9/1/57	Lines 8, 9*	Following the semicolon replace the word "is" by "may be." After the word "purposes" add "by the system operator or the maintenance engineer."
II-Intro-19; 9/1/57	Line 15	After the word "up" insert "or down."
II-Intro-19; 9/1/57	Line 19	Replace "up" by "down."
II-Intro-21; 9/1/57		The dashed connector from the Adder to the C register should be labeled (46).
II-Intro-22; 9/1/57	Line 4	After "up" insert "or down."
II-Intro-22; 9/1/57	Line 8	Replace "up" by "down."
II-00-3; 9/1/57	Line 5	After "is" insert "turned."
II-00-4; 9/1/57	Line 9	Delete the word "COMPUTER." After "PROGRAM" insert "CHECK."
II-10-3; 9/1/57	<u>Remarks:</u>	Add the following table as Remark 3:

(B[aaaa]:±1)	(rA:±1) _a	
	CAD	CAA
0	0	0
1	1	0
2	2	2
3	3	2
4	4	4
5	5	4
6	6	6
7	7	6
8	8	8
9	9	8

*Count every line of type, including heading.

Errata Sheet 2

Page/Date	Location	Description																																			
II-11-3; 9/1/57	Remarks:	Add the following table as Remark 2:																																			
		<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th rowspan="2" style="text-align: center;">(B[aaaa]: ±1)</th> <th colspan="2" style="text-align: center;">(rA:±1)a</th> </tr> <tr> <th style="text-align: center;">CSU</th> <th style="text-align: center;">CSA</th> </tr> </thead> <tbody> <tr><td style="text-align: center;">0</td><td style="text-align: center;">1</td><td style="text-align: center;">1</td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">0</td><td style="text-align: center;">1</td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;">3</td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">2</td><td style="text-align: center;">3</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;">5</td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">4</td><td style="text-align: center;">5</td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">7</td><td style="text-align: center;">7</td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">6</td><td style="text-align: center;">7</td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">9</td><td style="text-align: center;">9</td></tr> <tr><td style="text-align: center;">9</td><td style="text-align: center;">8</td><td style="text-align: center;">9</td></tr> </tbody> </table>	(B[aaaa]: ±1)	(rA:±1)a		CSU	CSA	0	1	1	1	0	1	2	3	3	3	2	3	4	5	5	5	4	5	6	7	7	7	6	7	8	9	9	9	8	9
(B[aaaa]: ±1)	(rA:±1)a																																				
	CSU	CSA																																			
0	1	1																																			
1	0	1																																			
2	3	3																																			
3	2	3																																			
4	5	5																																			
5	4	5																																			
6	7	7																																			
7	6	7																																			
8	9	9																																			
9	8	9																																			
II-11-4; 9/1/57	Line 17	On the left, the operation should read: (rA)-(rIB) → rA. On the right, the operation should read: (rA)- (rIB) →rA.																																			
II-12-4; 9/1/57	Line 12	Should read "to (rIB: ±1)/1."																																			
II-12-5; 9/1/57	Line 7	Place an asterisk after the contents of the D register in the columns for ADD and ADA. The asterisk refers to the following footnote which should be inserted below these two tables of Register Status: $*(rD:±1)/2=(rD:±1)/4=(rD:±1)/8=0;$ $(rD:±1)/1 = (B[aaaa]:±1)/1;$ $(rD:00) = (B[aaaa]:00).$																																			
II-13-5; 9/1/57	Line 7	Place an asterisk after the contents of the D register in the columns for SUB and SUA. The asterisk refers to the following footnote which should be inserted below these two tables of Register Status: $*(rD:±1)/2=(rD:±1)/4=(rD:±1)/8=0;$ $(rD:±1)/1 = (B[aaaa]:±1)/1;$ $(rD:00) = (B[aaaa]:00).$																																			

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Page/Date	Location	Description																						
II-14-3; 9/1/57	Remarks:	<p>Add Remark 1.</p> <p>1. Execution time for MULTIPLY-exclusive of fetch time-is a function of the magnitude of the multiplier, $(rA)_b$. It may be calculated from the following formula:</p> $T = 90 + 5 \sum_{k=0}^9 M_k \text{ } \mu\text{s,}$ <p>where</p> $M_k = 1 \quad \text{if } (rA:k1)=0,$ $M_k = 13 [(rA:k1)] + 1 \text{ if } 1 \leq (rA:k1) \leq 5,$ $M_k = 13 [11-(rA:k1)] \text{ if } 6 \leq (rA:k1) \leq 9.$ <p>Hence, the following table:</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">(rA:k1)</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">2</td> <td style="padding: 2px;">3</td> <td style="padding: 2px;">4</td> <td style="padding: 2px;">5</td> <td style="padding: 2px;">6</td> <td style="padding: 2px;">7</td> <td style="padding: 2px;">8</td> <td style="padding: 2px;">9</td> </tr> <tr> <td style="padding: 2px;">M_k</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">14</td> <td style="padding: 2px;">27</td> <td style="padding: 2px;">40</td> <td style="padding: 2px;">54</td> <td style="padding: 2px;">66</td> <td style="padding: 2px;">65</td> <td style="padding: 2px;">52</td> <td style="padding: 2px;">39</td> <td style="padding: 2px;">26</td> </tr> </table>	(rA:k1)	0	1	2	3	4	5	6	7	8	9	M _k	1	14	27	40	54	66	65	52	39	26
(rA:k1)	0	1	2	3	4	5	6	7	8	9														
M _k	1	14	27	40	54	66	65	52	39	26														
II-14-5; 9/1/57	Line 10	<p>Place an asterisk after the contents of the D register in the column for MUL. The asterisk refers to the following footnote which should be inserted below this table of Register Status:</p> $*(rD:\pm 1)/2 = (rD:\pm 1)/4 = (rD:\pm 1)/8 = 0;$ $(rD:\pm 1)/1 = (B[aaaa]:\pm 1)/1;$ $(rD:00) = (B[aaaa]:00).$																						
II-15-3; 9/1/57	Remarks:	<p>Add Remark 1.</p> <p>1. Execution time for DIVIDE-exclusive of fetch time-is a function of the magnitude of the quotient $(rA)_a$, if overflow does not occur. It may be calculated from the following formula:</p>																						

*Count every line of type, including heading.

Errata Sheet 2

Page/Date	Location	Description
,II-15-3; 9/1/57	Remarks:	Add Remark 1. (Continued) $T = 3895+60 \sum_{k=0}^4 [(rA:2k+1,1) - (rA:2k,0)] \mu s.$ $= 3895+60 [(rA:11) - (rA:21) + (rA:31) - (rA:41) + (rA:51) - (rA:61) + (rA:71) - (rA:81) + (rA:91) - (rA:01)] .$
, II-15-4; 9/1/57	Lines 21 and 22	The operation on the left should read "(rA:±1)/1 is set to (rR:±1)/1."
II-15-5: 9/1/57	Line 10	Place an asterisk after the contents of the D register in the columns for DIV and OVERFLOW. The asterisk refers to the following footnote which should be inserted below these tables of Register Status: $*(rD:±1)/2=(rD:±1)/4=(rD:±1)/8 = 0;$ $(rD:±1)/1 = (B[aaaa]:±1)/1;$ $(rD:00) = (B[aaaa]: 00).$
II-16-3; 9/1/57	Line 5	Replace "Than" by "Then."
II-16-3; 9/1/57	Line 13	The operation should read "[rA] + 0 0000 00 0001→rA."

*Count every line of type, including heading.

OPERATIONAL CHARACTERISTICS OF THE DATATRON 220

Page/Date	Location	Description
II-18-4; 9/1/57	Remarks:	Add Remark 4.

The order relationships of the content of the sign-digit position were determined as follows: the principal requirement is that negative numbers shall precede positive numbers; hence, $1 < 0$. A second requirement—sorting—is that alphanumeric information shall precede numeric; hence, $2 < 1 < 0$. The remaining order relationships are just a by-product of the way in which the required ordering of 2, 1, and 0 is achieved.

The technique used is known as the "threes complement" method: the 1-bit and the 2-bit of the sign digit are complemented if, and only if, the 8-bit is 0. The results are displayed in the following table:

Decimal Digit in Sign Position	Binary Representation	Threes Complemented	Decimal Equivalent (= order)
3	0011	0000	0
2	0010	0001	1
1	0001	0010	2
0	0000	0011	3
7	0111	0100	4
6	0110	0101	5
5	0101	0110	6
4	0100	0111	7
8	1000	1000	8
9	1001	1001	9

*Count every line of type, including heading.

Errata Sheet 2

Page/Date	Location	Description
II-19-4; 9/1/57	Line 11	Place an asterisk after the contents of the A register in the column for ADL. The asterisk refers to the following footnote which should be inserted following the table of Register Status: *Except that $(rA:\pm 1)/2 = (rA:\pm 1)/4 = (rA:\pm 1)/8 = 0$.
II-20-3; 9/1/57	Line 6	Add the following sentence: "Overflow of the B register does <u>not</u> set the OVERFLOW Indicator on."
II-22-5; 9/1/57	Line 6	On the right, the operation should be "(rA:91)→rA:01."
II-23-5; 9/1/57	Line 6	On the right, the operation should be "(rA:91)→rA:01."
II-27-3; 9/1/57	Line 19	The left half of the branch point operation should read "(rC:21) = 0."
III-Intro-3; 10/18/57	Line 8	Instead of "7·25-" read "8·25-."
IV-50-8; 9/1/57		Add Remark 11: The sign digit cannot be included in a partial-word search key. If it is, a field-overflow ALARM STOP will occur.
IV-51-5; 9/1/57		Add Remark 7: The sign digit cannot be included in a partial-word scan key. If it is, a field-overflow ALARM STOP will occur.
V-05-2; 9/1/57		Replace the definition of v by: <u>v = 8 or 9</u> : designated input will be B-register address-modified.

*Count every line of type, including heading.

Errata Sheet 2

Page/Date	Location	Description
A1-2; 9/1/57		Change instruction format for
A2-2; 9/1/57		SPO to ± dnnv 09 aaaa; FAD to
A2/3; 9/1/57		± nii0 22 aaaa; FAA to
		± niil 22 aaaa; FSU to
		± nii0 23 aaaa; FSA to
		± niil 23 aaaa.
A3-3; 9/1/57		Change the punched-card code
		for FORM OUT to 11 8-5.