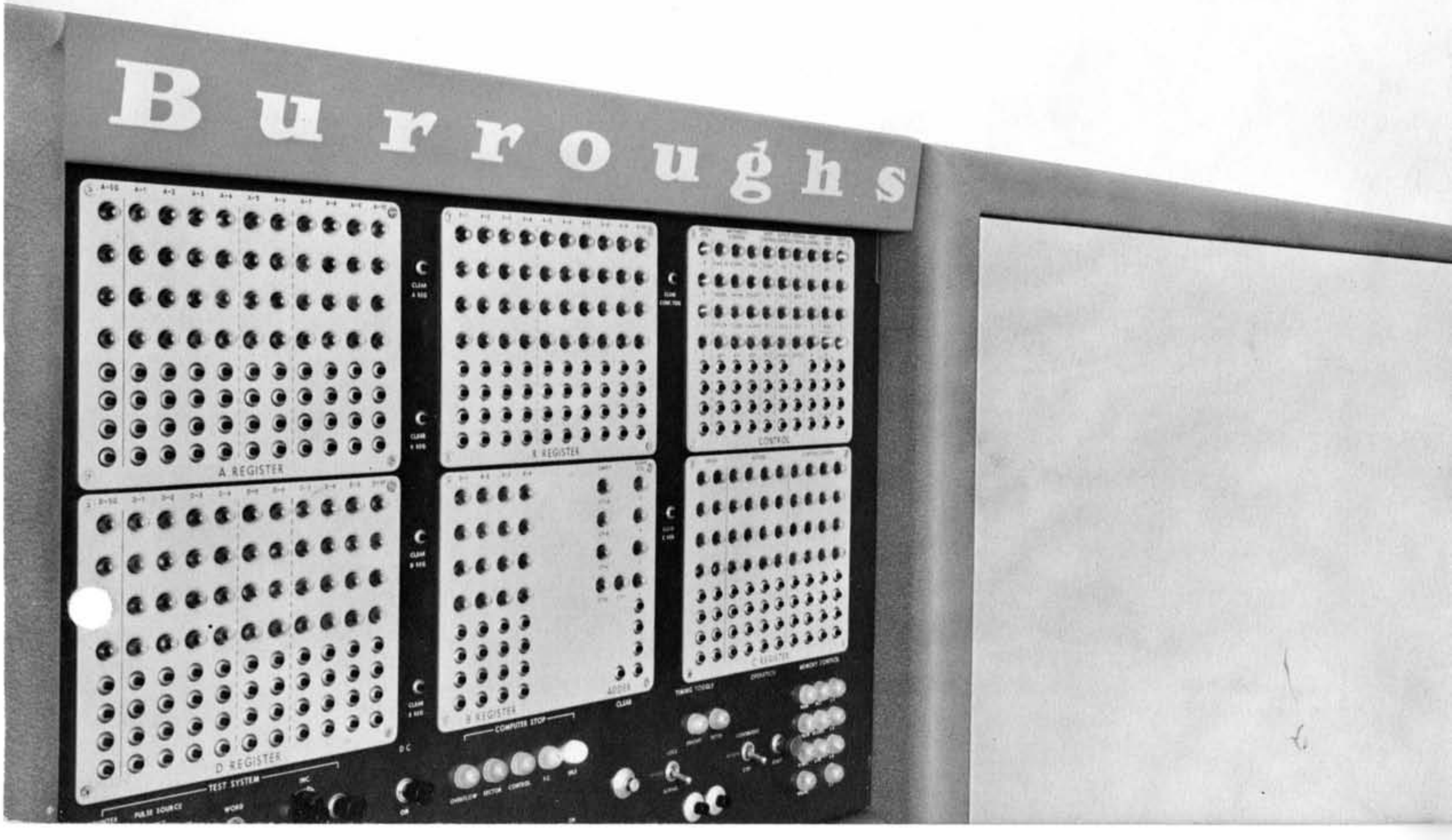
BULLETIN 3021

# Burroughs 205

ELECTRONIC DATA-PROCESSING SYSTEMS

# HANDBOOK

## central computer



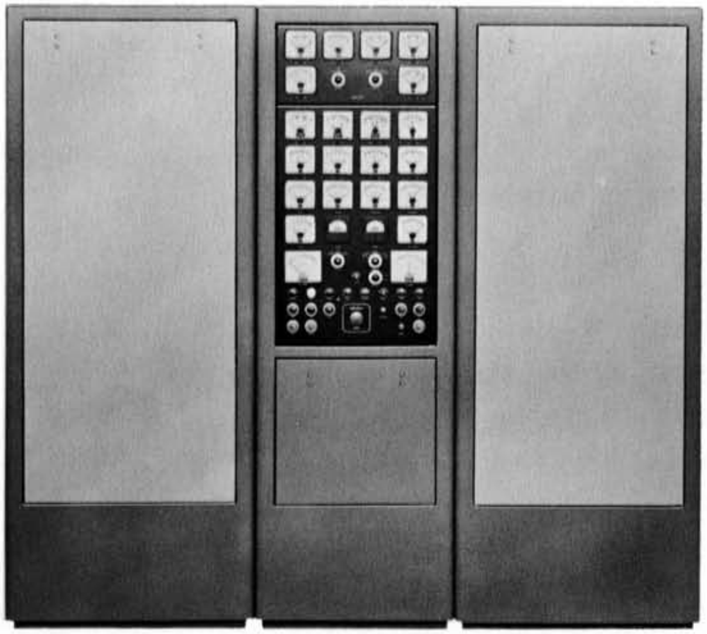
This handbook supersedes and replaces previous editions of Bulletin 3010, Summary Instruction List, and Bulletin 3040A (Programming and Coding Manual). Symbols and nomenclature used to designate instructions conform to the revised standard practice adopted in March, 1956.

First printing March, 1956

Revised February, 1958

# TABLE OF CONTENTS

C CABINET A CABINET B CABINET



AUXILIARY POWER SUPPLY UNIT POWER CONTROL UNIT BASIC POWER SUPPLY UNIT

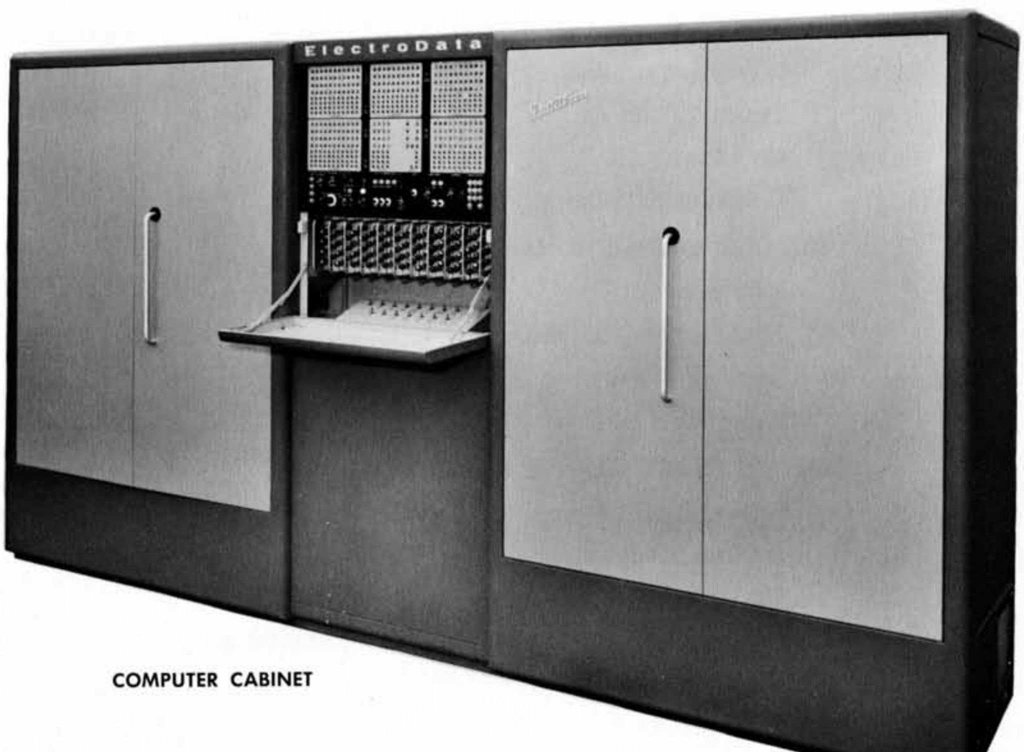



Figure 1 Magnetic Amplifier Power Supply – Computer Cabinet

# BURROUGHS 205 ELECTRONIC DATA-PROCESSING SYSTEMS

## GENERAL

Electronic data-processing systems have five components – input, storage (working and auxiliary), arithmetic, control, and output. This handbook describes the characteristics and explains the use of the Burroughs Electronic Data-Processing System Model 205. This unit, which consists of the Computer Cabinet, the Power Control, and Magnetic Amplifier Power Supply (Figure 1), contains the working storage, arithmetic, and control components of a complete system.

## BURROUGHS 205 COMPUTER

The computer is a general purpose, internally programmed, decimal, electronic computer with magnetic drum storage. It is the heart, or central controlling and processing unit, of an electronic data-processing system which accomplishes the functions of:

1. Accepting data directly from punched cards, punched tape, magnetic tape, keyboard – employing input units singly or in multiple.
2. Selecting from magnetic tape files the historical or reference records necessary to process data.
3. Processing data – comparing, computing, analyzing, sorting, classifying as required – in obedience to a series of instructions which have previously been stored in the system (stored program).
4. Bringing up to date the historical or reference records maintained on magnetic tape, and returning the up-dated records to magnetic tape.
5. Transmitting required information directly into punched cards, punched tape, magnetic tape, printed documents, visual indications – employing output units singly or in multiple.

As a result of its ability to control data-processing systems of wide scope, and because of its economical and reliable operation, the Burroughs 205 has been applied effectively to a wide range of commercial, manufacturing, scientific and engineering problems.

In speed of computer operation, the Burroughs 205 is classed below the very large-scale electronic data processors – and considerably above card-programmed computers, other externally programmed computers, and the small, stored program computers.

In capacity and data-processing capability, the Burroughs 205 (as the central unit in a system) approaches large-scale systems in power and ability to produce an effective and economical flow of work.

## COMPONENTS OF THE BURROUGHS 205

The Computer Cabinet contains the arithmetic and control units (see Figure 1). The center section contains the magnetic drum working storage and the Control Panel. Switches, indicators, and displays required by the operator are mounted on this panel.

The Magnetic Electronic Power Supply is a compact, electronically controlled power system which provides all voltages necessary for the Burroughs 205 computing system. Many features are included, such as complete metering circuits, metering display of all voltages, and dependable regulation under varying load conditions and input line variations.

The power system shown in Figure 1 is composed of three consoles: the Basic Power Control Unit (A cabinet); the Basic Power Supply Unit (B cabinet); and the Auxiliary Power Supply Unit (C cabinet). All Burroughs 205 Systems require the Basic Power Supply Unit and Power Control Unit. Only systems that include Cardatron require the additional Auxiliary Power Supply Unit.

# OPERATING CHARACTERISTICS OF THE DATATRON

## HOW INFORMATION IS REPRESENTED

Information is represented in the Burroughs 205 as fixed length numbers, each of which contains ten decimal digits. Each ten digit number is preceded by an additional digit (Figure 2) which
– represents the algebraic sign of the number, or
– is sometimes used to control machine operation, or
– is an arbitrary zero having no special significance.

Each of these 11 digit units of information, called a **word,** may represent numerical data, alphabetic data, or a mixture (alphanumeric data), or an instruction which the Burroughs 205 is to obey. For example:

| | | |
|---|---|---|
| 0 4259 64 4955 | can represent | + 4 259 644 955 |
| 0 4259 64 4955 | can represent | B R U I N |
| 0 4259 64 4955 | can represent | "Clear the A Register. Add the contents of storage cell 4955." |

By changing two digits in the sample word, an alphanumeric word can be represented:

0 4259 **86** 4955 represents Part Number B R 6 I N

The interpretation of the word is controlled by the programmer as he arranges the instructions and data for input to the computer.

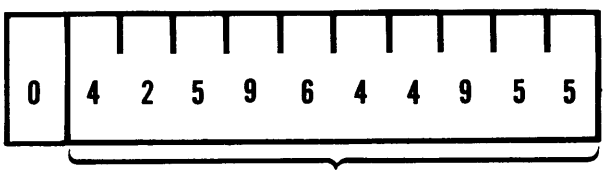



Figure 2

The 11 digit word is treated as a unit by the Burroughs 205. It is stored as a unit, and it is manipulated as a unit. However, if it is necessary to break up a word into smaller units of information, or to combine words into longer records, this can be done by placing the proper series of instructions in the Burroughs 205.

## HOW INFORMATION IS STORED

Over 4000 words of information are stored on the surface of a large-capacity magnetic drum which revolves at 3570 revolutions per minute. This unusual storage capacity makes possible
– adequate reference to data,
– adequate facility for classification of data,
– convenient use of long programs,
– convenient insertion of temporary programs for "spot" analysis,
– improved internal sorting techniques, and
– a reduction, in many cases, in the number of times the same data must be fed through the central data processor to secure the desired results.

Once placed on the drum, information is retained (even if the power is turned off) until it is "erased" by writing new information on the drum over the old information.

Only the digits **zero** and **one** are represented on the surface of the magnetic drum – and this representation is made by magnetizing a small spot on the drum for each digit. All **zero** spots are magnetized in the same direction of polarity, and all **one** spots are magnetized alike in the opposite direction. Four such spots (called bits of information or binary digits) are used to represent one decimal digit. In this scheme of representation (binary-coded decimal), one bit of information is assigned the value **1**, the second bit is assigned the value **2**, the third bit is assigned the value **4**, and the fourth bit is assigned the value **8**. Decimal digits are represented according to the following table:

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **8 Bit** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **4 Bit** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| **2 Bit** | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **1 Bit** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Write heads and read heads are mounted on the magnetic drum casing (Figure 3). As the drum cylinder revolves inside the casing, the surface of the drum passes these heads. The function of each write head is to place

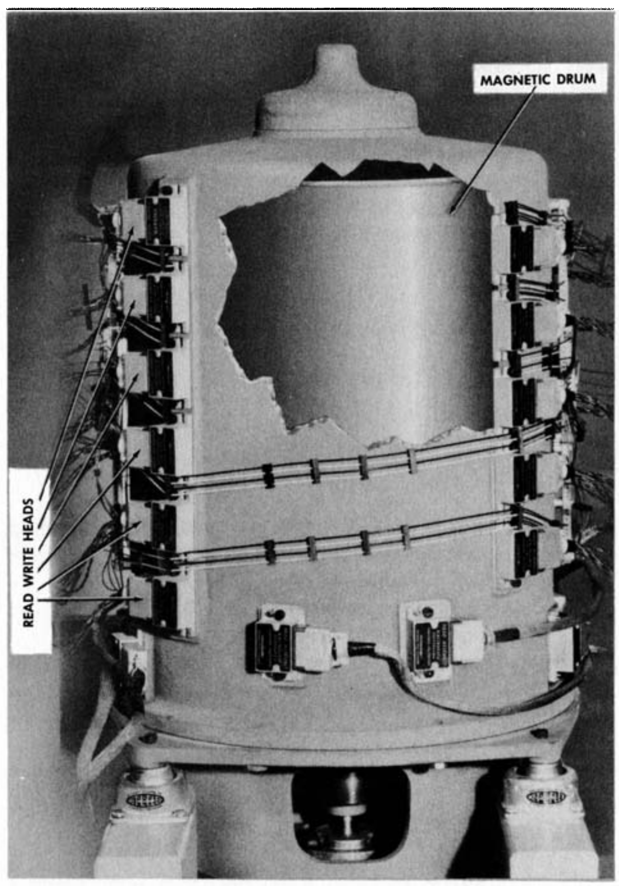



Figure 3 Magnetic Drum Assembly

information on the surface of the drum by magnetizing four spots at a time according to the code tabulated above. The function of each read head is to interpret the pattern of magnetic spots on the surface of the drum, four bits of information at a time, thus making the information available for use.

## LOCATION OF INFORMATION ON THE MAGNETIC DRUM

A space on the drum large enough to write the contents of exactly one word is called a **storage cell**. Storage cells are arranged in bands which extend around the circumference of the magnetic drum. Each band consists of four tracks of magnetized spots (Figure 4), making possible the use of the binary-coded decimal scheme of representing digits. Four **zeros**, one in each of the tracks, separate each word from its adjoining words. Associated with each band is a read head and a write head, or a combination read-write head.

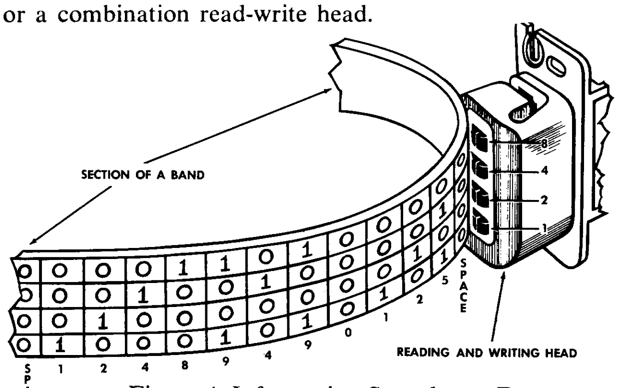



Figure 4 Information Stored on a Drum

Each storage cell on the magnetic drum has its own **address**, a four digit number which identifies the cell and specifies its location. The top 20 bands on the magnetic drum (Figure 5) each contain 200 words, a total of 4000 storage cells being located in the portion of the magnetic drum called **main storage**. The addresses of these cells are the numbers 0000 through 3999.

The bottom four bands on the drum are reserved for high-speed storage. Only 20 different words are stored in each of these quick-access bands, but each word is repeated ten times at equal intervals around the drum. This provides the basis for faster access to each word in the band—as described below.

The bands can be addressed from 4000 through 4999, 5000 through 5999, 6000 through 6999, and 7000 through 7999. Because each word is repeated ten times, the word in 4000 will be repeated in 4020, 4040, 4060, 4080; however, the word can be addressed without regard to the second address digit. Thus the word in 4000 could be addressed as 4100, 4520, 4760, 4980, etc.; cell 5569 will contain the same word as cell 5009; cell 6738 will contain the same word as cell 6018, etc.

## OPERATION OF QUICK-ACCESS BANDS

Each main storage band has associated with it **one combination** read-write head (Figure 6). A word stored in a main storage cell passes the read-write head only once in every revolution of the magnetic drum. A word stored in a main storage cell is available for use, then, once in every revolution of the drum. The access time (or waiting time) for this word can vary from zero to 0.017 seconds (17 milliseconds). The average access time for the word is 8.5 milliseconds, the time for a half-revolution of the drum.

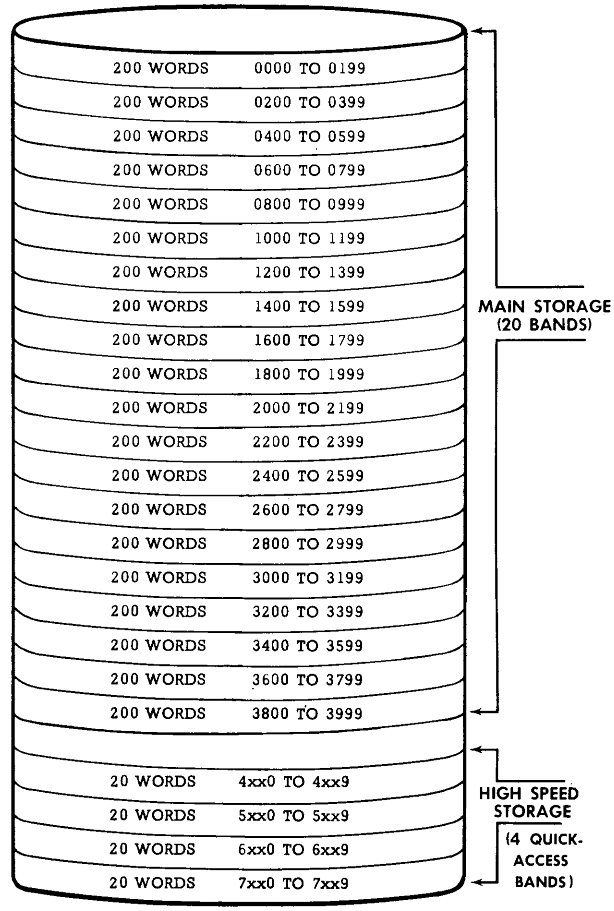



Figure 5 Location of Information on Magnetic Drum

Each quick access band has a **separate** read head and, 20 words distant from this head along the drum circumference, a **separate** write head (Figure 7). Since a complete band around the magnetic drum contains 200 words, these two heads are one-tenth of the drum circumference apart.

As each word passes under the read head, it is **always** immediately rewritten twenty words back along the drum circumference. If a block of 20 words is placed in a quick access band this continual process of reading and writing will duplicate the 20 words in **ten locations** around the drum—in the first revolution of the magnetic drum following the transfer of information into the band.

A word stored in one of the cells of a quick access band is available for use once in every one-tenth of a drum revolution, or **ten times in every revolution.** In effect, the quick access bands supply data and instructions at the same rate as if the magnetic drum were revolving at 35,700 revolutions per minute. The access time for a word in a band can vary from zero to 1.7 milliseconds. The average access time for a word stored in a band is 0.85 milliseconds.

In most applications, Burroughs 205 instructions are transferred from main storage into the quick access bands before the execution of the instructions. Similarly, data and intermediate results are normally stored in the quick access bands, or transferred from main storage into the quick access bands.

To accomplish the necessary manipulation of information, block transfer instructions are used. These instructions move 20 words at a time from main storage to band, or from band to main storage, at the rate of 1.7 milliseconds per block of 20 words. This is the amount of time required for 20 words to pass by a read head. The actual transfer of each digit is almost instantaneous.

Words transferred from main storage to a band remain (in unaltered form) in main storage, facilitating the process of making memo entries in records. Words transferred from a band to main storage remain (in unaltered form) in the band.

Burroughs 205 programs are written to maintain a continuous flow of data and instructions through the bands. Thus, the Burroughs 205 maintains the high rate of processing associated with optimal (or minimal access) programming, but retains the reliability inherent in a conservative speed of drum revolution.

## ELECTRONIC REGISTERS

On the magnetic drum, each decimal digit is represented by a combination of four magnetized spots, each spot being an indicator of either **zero** or **one**. This method of representing information has proved to be extremely reliable.

An electronic circuit, called a **flip-flop**, can also represent **zero** or **one** by being in one of two possible states – either "low" or "high." Several registers, or storage cells with zero access time, use the **flip-flop** circuit to store information. In these registers, each decimal digit is represented by four flip-flops. Just as in the case of the magnetized spots on the magnetic drum, relative values are assigned to each flip-flop. The first flip-flop is assigned the value **1**, the second flip-flop is assigned the value **2**, the third flip-flop is assigned the value **4**, and the fourth flip-flop is assigned the value **8**. Decimal digits are represented in electronic registers according to the table of combinations used to represent decimal digits on the drum. (Refer to How Information is Stored.)

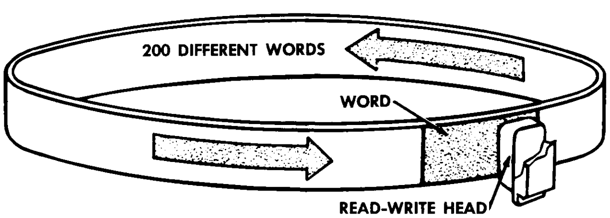


Figure 6 Access to Word Stored in Main Storage Band

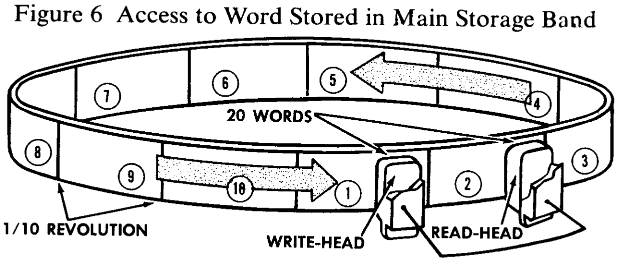


Figure 7 Access to Word Stored in Quick Access Band

## ARITHMETIC REGISTERS

Three electronic registers are used to contain numbers involved in computation and data processing (Figure 8).

**A Register** holds an 11 digit word. This register is an accumulator in which the results of all arithmetic operations appear.

**R Register** holds ten decimal digits. This register is primarily an extension of the A Register. However, multiplication and division are the only arithmetic operations which affect the R Register.

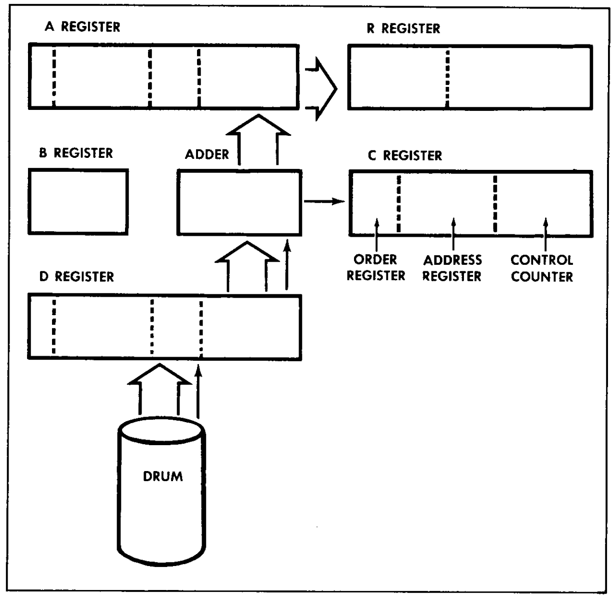


Figure 8 Arithmetic Registers

**D Register** holds an 11 digit word which cannot be manipulated by the programmer. Words entering the A, R and C registers from an input medium, or from main storage, first pass through this register.

In an arithmetic operation one of the numbers involved is always in the A Register, or in the combined A Register and R Register. The second number involved is always transferred from the drum into the D Register.

## INSTRUCTION FORMAT

A Burroughs 205 instruction is made up of three parts (Figure 9):

(a) the four digit address – which designates the location of the storage cell referred to during execution of the instruction;

(b) The two digit order – which designates the specific operation to be performed;

(c) the four control digits – which designate variations in the execution of the instruction

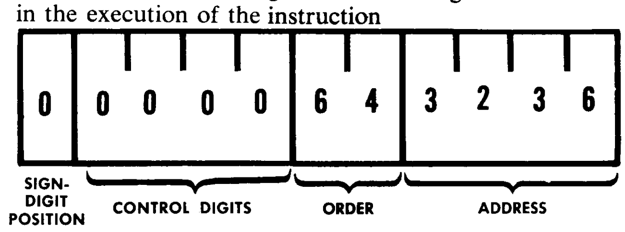


Figure 9 Instruction Format

## C REGISTER

**C Register** receives each instruction from the magnetic drum through the D Register (Figure 10). The function of this register is to start the operation of the control component of the computer.

The C Register is composed of three sub-registers (reading from left to right):

**Order Register** holds the two digits which designate the specific operation to be performed.

**Address Register** holds the four digits which designate the location of the storage cell referred to during execution of the instruction. The contents of the Order Register and the Address Register, together, are the same as the six right hand digits of the instruction word as it appears in the D Register and on the magnetic drum.

**Control Counter** holds the four digits which specify the address of the next instruction which will be executed – after the completion of the operation specified in the Order Register and the Address Register.

## OPERATION SEQUENCE

In normal, continuous operation, instructions are executed in the order in which they are stored on the magnetic drum. Thus, if instructions are stored in storage cells 1000, 1001, and 1002, the instruction stored in cell 1001 will be executed after the instruction stored in cell 1000 and the instruction stored in cell 1002 will be executed after the instruction stored in cell 1001.

The Control Counter counts up 1 after each instruction comes into the C Register so that the next instruction will be read from the next cell. In the preceding example, when the instruction stored in cell 1000 is being executed, the Control Counter will read 1001. When the instruction stored in cell 1001 is being executed, the Control Counter will read 1002 (Figure 10).

To change this normal method of sequential operation, change of control instructions are used. These instructions may be used to alter the sequence of instruction execution arbitrarily – in which case they are **unconditional changes of control.** A similar series of instructions may be used to alter the sequence of instruction execution only in response to the presence of a machine condition (see **Overflow**, below). These **conditional changes of control** are used for decision-making or branching.

Instead of allowing the Control Counter to count up 1, the change of control instructions insert their address digits into the Control Counter, and thus specify the next instruction to be executed.

## TIMING CYCLE

The timing cycle of the Burroughs 205 has two distinct phases:

In the **fetch phase** of the timing cycle (Figure 11), the instruction word located in the storage cell specified in the Control Counter is brought from the magnetic drum, through the D Register, through the Adder, to the C Register.

In the **execute phase** of the timing cycle (Figure 12), the data word specified in the instruction just fetched is brought from the magnetic drum, through the D Register, through the Adder (where an arithmetic operation takes place) to the A Register.

The fetch phase and the execute phase alternate as the timing cycle repeats.

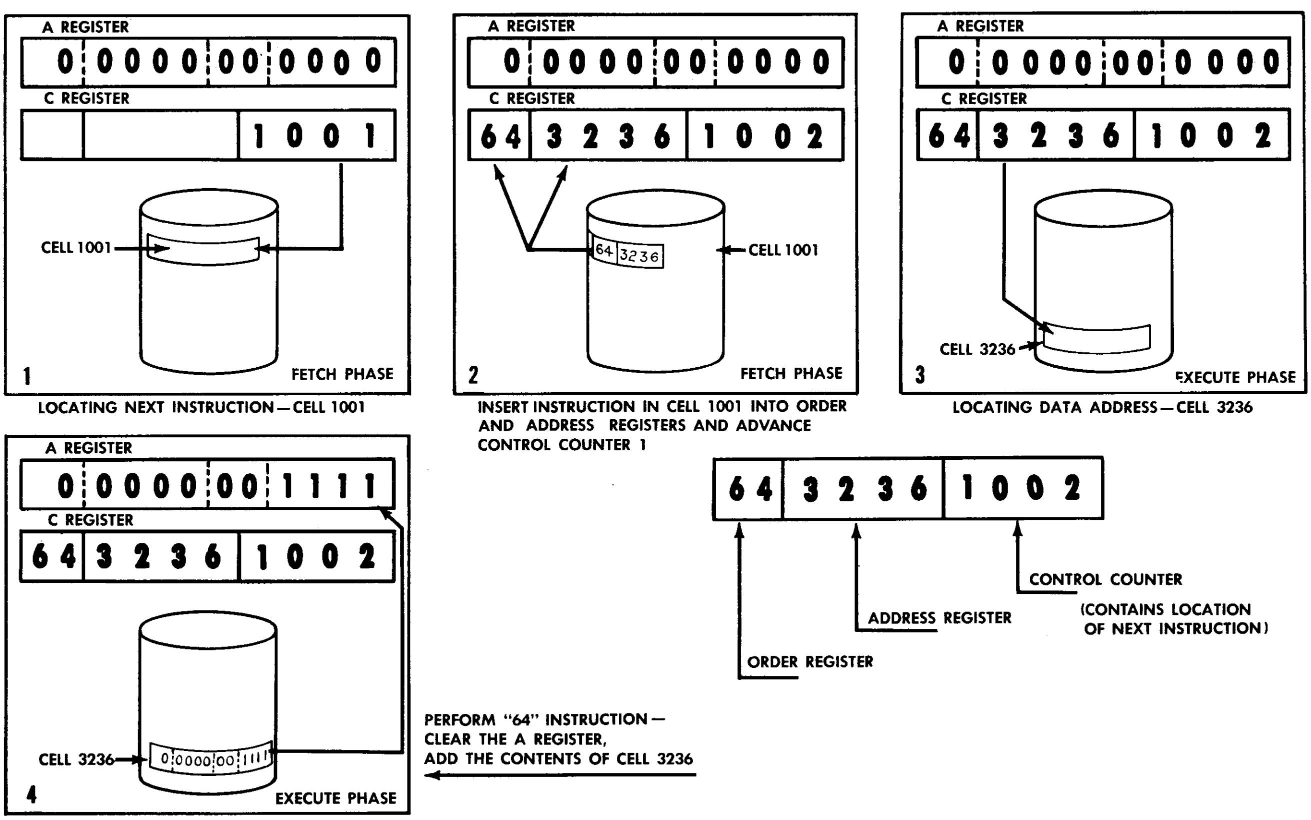



Figure 10 Action of Control Counter

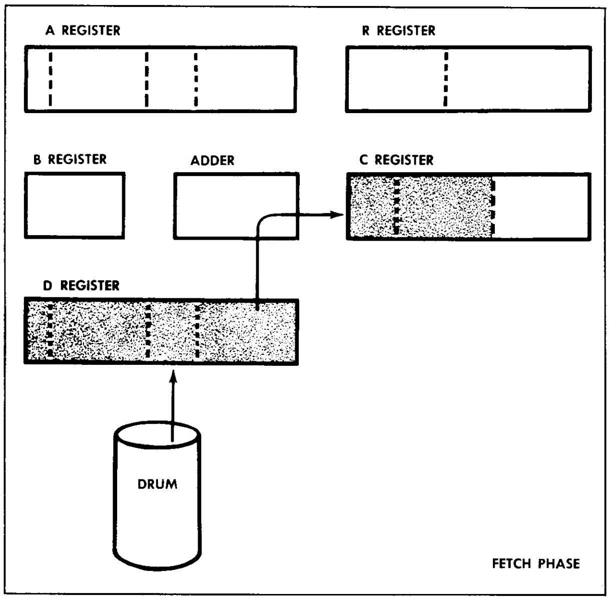



Figure 11 Timing Cycle

## B REGISTER

The **B Register** holds any four decimal digits from 0000 to 9999. These digits can be added to the address digits of an **instruction word** as the instruction goes through the Adder to the C Register (Figure 13).

The addition of the contents of the B Register to an instruction (address modification) is signaled by the first control digit of the instruction word, when the word reaches the D Register. If the digit is 1, the contents of the B Register are added. If the control digit of the instruction word is 0, the contents of the B Register are not added (see Figure 11).

The contents of the B Register can be increased by **one**, or decreased by **one**, during the execution of a series of instructions. When the series of instructions is repeated many times, the B Register can serve, in this case, as a tallying device.

## DECIMAL POINT

Inside the computer, a decimal point is considered to be fixed at the left of each ten digit word stored on the magnetic drum or in the electronic registers.

The eleventh digit, at the left of the decimal point, represents the algebraic sign of numerical data (**zero** for **plus** and **one** for **minus**), or (in the case of an instruction word) is sometimes used to control machine operation, or (in the case of alphabetic or alphanumeric data) is an arbitrary zero having no special significance.

Outside the computer, the decimal point may be located in its proper position (by programming) regardless of its internal position. For example:

| **Internally** | | **Externally** |
|---|---|---|
| 0 1621 00 0000 | | |
| 0 0001 62 1000 | may represent | 16.21 |
| 0 0000 00 1621 | | |

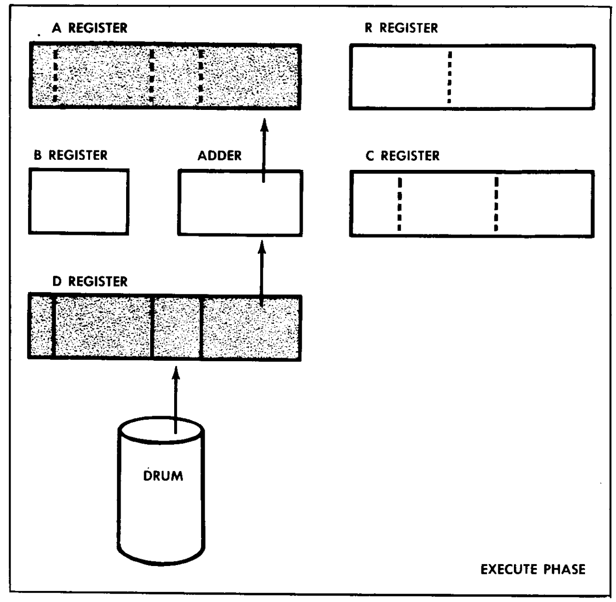



Figure 12 Timing Cycle

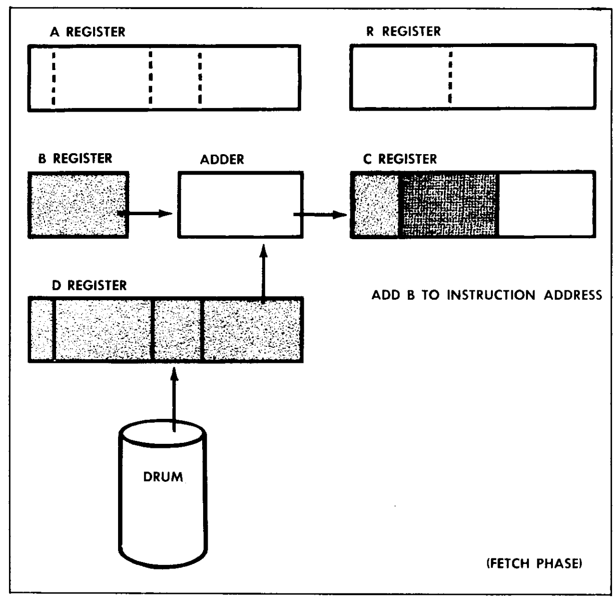



Figure 13 B Register Modification

## OVERFLOW

Whenever the execution of an instruction produces a result which is too large to be inserted in the A Register, an **overflow condition** is set up in the computer. This condition will cause the computer to stop, unless a conditional change of control instruction follows the instruction that caused the overflow condition (previously discussed in the section, **Operation Sequence**). The presence of the overflow condition is determined as follows:

Indication to computer—Overflow flip-flop is in a "high" state.

Indication to operator – Overflow light is ON.

**EXAMPLE 1**

| Actual Arithmetic | Computer Arithmetic |
| --- | --- |
| 0.9000 00 0000 | 0 9000 00 0000 |
| +0.8000 00 0000 | 0 8000 00 0000 |
| 1.7000 00 0000* | 0 7000 00 0000** |
| *Carry produces number to the left of the decimal point. | **Overflow indicator ON. Zero to the left of decimal point position represents plus sign. The resulting 'carry one' is lost. |

**EXAMPLE 2**

**Actual Arithmetic**

$$0.3000\ 00\ 0000 \overline{)0.9000\ 00\ 0000} = 3.0000\ 00\ 0000^{*}$$

*Division of larger number by a smaller number produces a whole number or overflow to the left of the machine decimal point.

**Computer Arithmetic**

$$0\ 3000\ 00\ 0000 \overline{)0\ 9000\ 00\ 0000} = 0\ 0000\ 00\ 0000^{**}$$

**Overflow indicates ON. Zero to the left of decimal point position represents plus sign.

The overflow condition **may** follow the arithmetic manipulation of the contents of the A Register. Overflow **always** follows the test for and detection of a difference between the algebraic sign of the A Register and the sign of a number brought from a storage cell for comparison.

When the possible appearance of an overflow is anticipated, a conditional change of control instruction is inserted in the program to allow the program to branch (take one of two possible alternate paths). When an unanticipated overflow occurs (a programming error) the computer stops.

## CHECKING FACILITIES

When the computer automatically stops upon the appearance of an unanticipated overflow, an alarm light is turned on and computation is stopped by a forbidden combination (binary-coded decimal digits 10 through 15) in the A, B, D, or R Registers, the Address Register, the Control Counter, or the Shift Counter. Inspection of the register contents as indicated on the Control Panel indicates the failure.

An alarm will stop machine operation if the storage cell counter does not contain 0 at the start of each drum revolution. This check prevents information from being recorded on or read from incorrect locations on the drum.

An audible alarm indicates excessive rise in exhaust air temperature in the computer cabinet and after a preset interval up to 15 minutes, dc voltages will be shut off if the temperature stays at or above a predetermined level.

# COMPUTER INSTRUCTIONS

This section defines the Burroughs 205 instructions available to the programmer and illustrates their use.

## ARITHMETIC

**Instructions for Addition and Subtraction.**

Addition and subtraction instructions affect the A Register, but not the R Register.

The series of instructions below illustrates the use of the add and subtract instructions and the effect that each instruction has on the A Register and the R Register.

Assume that:

1. Storage cell 1000 contains the number 0 2222 22 2222.
2. Storage cell 1001 contains the number 1 3333 33 3333.
3. The A Register contains the number 1 9874 53 1234.
4. The R Register contains the number 0000 560000.
5. Insert a 7 on Keyboard for Digit Add.

**CAD** — **CLEAR, ADD**
**000p 64 xxxx**
Clear the A Register. Add the contents of **xxxx**.

**AD** — **ADD**
**000p 74 xxxx**
Add the contents of **xxxx** to the contents of the A Register.

**ADA** — **ADD ABSOLUTE**
**000p 76 xxxx**
Add the absolute value of the contents of **xxxx** to the contents of the A Register.

**CADA** — **CLEAR, ADD ABSOLUTE**
**000p 66 xxxx**
Clear the A Register. Add the absolute value of the contents of **xxxx.**

**CSU** — **CLEAR, SUBTRACT**
**000p 65 xxxx**
Clear the A Register. Subtract the contents of **xxxx.**

**SU** — **SUBTRACT**
**000p 75 xxxx**
Subtract the contents of **xxxx** from the contents of the A Register.

**CSUA** — **CLEAR, SUBTRACT ABSOLUTE**
**000p 67 xxxx**
Clear the A Register. Subtract the absolute value of the contents of **xxxx.**

**SUA** — **SUBTRACT ABSOLUTE**
**000p 77 xxxx**
Subtract the absolute value of the contents of **xxxx** from the contents of the A Register.

**DAD** — **DIGIT ADD**
**0000 10 0000**
Stop machine operation. Add the next digit read (from manual keyboard or paper tape reader) to the least significant position of the A Register.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 9874 53 1234 | |
| CAD | 1000 | 0 2222 22 2222 | 0000 560000 |
| AD | 1001 | 1 1111 11 1111 | 0000 560000 |
| ADA | 1001* | 0 2222 22 2222 | 0000 560000 |
| CADA | 1001* | 0 3333 33 3333 | 0000 560000 |
| CSU | 1000 | 1 2222 22 2222 | 0000 560000 |
| SU | 1001 | 0 1111 11 1111 | 0000 560000 |
| CSUA | 1001* | 1 3333 33 3333 | 0000 560000 |
| SUA | 1001* | 1 6666 66 6666 | 0000 560000 |
| DAD | 0000 | 1 6666 66 6659 | 0000 560000 |

*In addition and subtraction of absolute numbers, the number is treated as a positive number, regardless of its sign.

The condition of overflow in AD, ADA SU, SUA is possible and will appear as follows:

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 9874 53 1234 | 0000 560000 |
| *SU | 1000 | 1 2096 75 3456 | 0000 560000 |
| SUA | 1001 | 1 5430 08 6789 | 0000 560000 |
| CADA | 1001 | 0 3333 33 3333 | 0000 560000 |
| ADA | 1001 | 0 6666 66 6666 | 0000 560000 |
| ADA | 1001 | 0 9999 99 9999 | 0000 560000 |
| *ADA | 1001 | 0 3333 33 3332 | 0000 560000 |

*Overflow indicator ON.

**Addition and Subtraction instructions can be used in:** posting, accumulating receipts, debiting and crediting accounts and, in general, updating records.

PROBLEM: A store has four sections. Following each day's business the owner wants to know net receipts. Each section reports total receipts and amount of sales commissions.

TO FIND: Net Receipts

ASSUME: Information from sections located in storage cells:

| | | |
|---|---|---|
| 1000 | 0 0000 01 9432 | (Section 1 – Sales – 194.32) |
| 1001 | 0 0000 00 3886 | (Section 1 – Commissions – 38.86) |
| 1002 | 0 0000 01 5203 | (Section 2 – Sales – 152.03) |
| 1003 | 0 0000 00 3040 | (Section 2 – Commissions – 30.40) |
| 1004 | 0 0000 00 9367 | (Section 3 – Sales – 93.67) |
| 1005 | 0 0000 00 1873 | (Section 3 – Commissions – 18.73) |
| 1006 | 0 0000 01 0152 | (Section 4 – Sales – 101.52) |
| 1007 | 0 0000 00 2030 | (Section 4 – Commissions – 20.30) |

SOLUTION:

| Location | | S | Control Digits | Operation | | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | | | No | Alpha | | |
| | | | | | | | |
| 000 0 | 0 | | | | CAD | 1000 | A = 0∧0000 01 94.32 |
| 000 1 | 1 | | | | SU | 1001 | A = 0∧0000 01 55.46 |
| 000 2 | 2 | | | | AD | 1002 | A = 0∧0000 03 07.49 |
| 000 3 | 3 | | | | SU | 1003 | A = 0∧0000 02 77.09 |
| 000 4 | 4 | | | | AD | 1004 | A = 0∧0000 03 70.76 |
| 000 5 | 5 | | | | SU | 1005 | A = 0∧0000 03 52.03 |
| 000 6 | 6 | | | | AD | 1006 | A = 0∧0000 04 53.55 |
| 000 7 | 7 | | | | SU | 1007 | A = 0∧0000 04 33.25 |

**ANSWER: Located in the A Register represents $433.25**

**∧ = machine decimal point**

**• = programmer's decimal point**

**Instructions for Multiplication and Division:**

Multiplication and division instructions affect both the A Register and the R Register.

The series of instructions below illustrates the use of instructions for multiplication and division, and the effect that each instruction has on the A Register and the R Register.

## ▼ MULTIPLICATION

**M** **MULTIPLY**
**000p 60 xxxx**

Multiply the contents of **xxxx** by the contents of the A Register. Insert the twenty digit product in the A Register and the R Register. The ten most significant digits are in the A Register. The ten least significant digits appear in the R Register. SC ← 9

**MRO** **MULTIPLY, ROUND**
**000p 70 xxxx**

Multiply the contents of **xxxx** by the contents of the A Register. Round the product to ten digits. Clear the R Register.

During the execution of the Multiply instruction, the R Register is cleared to permit the insertion of the least significant ten digits of the product.

The A Register will contain the proper algebraic sign of the product. SC ← 9

## ▼ DIVISION

**DIV** **DIVIDE**
**000p 61 xxxx**

Divide the twenty digit contents of the A Register and the R Register by the contents of **xxxx.**

(a) If Overflow indicator ON, clear the A Register and the R Register. SC ← 15

(b) If Overflow indicator OFF, insert the quotient in the A Register, and insert the undivided remainder (if any) in the R Register. SC ← 9

In division, the divisor must be greater than the portion of the dividend in the A Register. If the dividend is greater than or equal to the divisor, the quotient will exceed the capacity of the A and R Register and an overflow will occur. If the dividend is contained in the A Register, then the R Register must be cleared before dividing.

Assume that:

1. Storage cell 1000 contains the number 0 2222 22 2222.
2. Storage cell 1001 contains the number 1 3333 33 3333.
3. The A Register contains the number 0 9999 99 9999.
4. The R Register contains the number 9999 999999.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 0 9999 99 9999 | 9999 999999 |
| CAD | 1000 | 0 2222 22 2222 | 9999 999999 |
| M | 1001 | 1 0740 74 0740 | 5925 925926 |
| CAD | 1000 | 0 2222 22 2222 | 5925 925926 |
| MRO | 1001 | 1 0740 74 0741 | 0000 000000 |
| CAD | 1000 | 0 2222 22 2222 | 0000 000000 |
| DIV | 1001 | 1 6666 66 6666 | 2222 222222 |

Overflow is impossible in multiplication.

Example of overflow in division:

| Program | | A Register | R Register |
|---|---|---|---|
| | | 0 9822 70 9243 | 0000 000000 |
| CAD | 1001 | 1 3333 33 3333 | 0000 000000 |
| *DIV | 1000 | 0 0000 00 0000 | 0000 000000 |

***Overflow indicator ON.**

Multiplication and Division instructions can be used in: **determining rates, payroll extension, billing, tax computation,** and general engineering computations.

PROBLEM: A store owner wants to take advantage of a close-out sale to purchase 2,250 items at $10.00 each. There will be a shipping cost of $380.00. There is a $909.00 discount if the purchase is made on an 18 month contract at 6.5% interest. The store owner wants to know what his monthly interest payments will be.

TO FIND: Monthly interest payments.

ASSUME: Information for purchase located in storage cells:

| | | |
|---|---|---|
| 1000 | 0 0002 25 0000 | (Quantity) |
| 1001 | 0 1000 00 0000 | (Unit Price) |
| 1002 | 0 0000 00 3800 | (Shipping Cost) |
| 1003 | 0 0000 00 9090 | (Discount) |
| 1004 | 0 6500 00 0000 | (Interest Rate) |
| 1005 | 0 0000 18 0000 | (Number of Payments) |

SOLUTION:

| Location Main | Location Loop | S | Control Digits | Operation No | Operation Alpha | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 0000 | 0 | | | | CAD | 1000 | A= 0∧0002 25 0.000 |
| 0001 | 1 | | | | M | 1001 | A= 0∧0000 22 500.0 |
| 0002 | 2 | | | | AD | 1002 | A= 0∧0000 22 880.0 |
| 0003 | 3 | | | | SU | 1003 | A= 0∧0000 21 971.0 |
| 0004 | 4 | | | | MRO | 1004 | A= 0∧0000 14 28.12 |
| 0005 | 5 | | | | DIV | 1005 | A= 0∧79.34 00 0000 |

**ANSWER: Located in the A Register represents $79.34**

**∧ = machine decimal point**

**• = programmer's decimal point**

# MANIPULATION AND TRANSFER OF INFORMATION

**SL** **SHIFT LEFT**
**000p 14 00nn**
Shift the contents of the A Register and the R Register **nn** places left. The **nn** digits shifted out of the left end of the A Register re-enter the right end of the R Register in the same order. The sign does not move.

**SR** **SHIFT RIGHT**
**000p 13 00nn**
Shift the contents of the A Register and the R Register **nn** places right. The **nn** digits shifted out of the right end of the R Register are lost, and **nn** zeros enter the left end of the A Register. The sign does not move. The maximum value for **nn** is 19.

Overflow can not occur on shifting instructions

**CR** **CLEAR R**
**000p 33 0000**
Clear the R Register.

**RO** **ROUND**
**000p 23 0000**
Round the twenty digit contents of the A Register and the R Register to ten digits. Clear the R Register.

**STOP** **STOP**
**000p 08 0000**
Stop machine operation.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 0 1234 56 7891 | 2345 678912 |
| SL | 0004 | 0 5678 91 2345 | 6789 121234 |
| CR | 0000 | 0 5678 91 2345 | 0000 000000 |
| SR | 0012 | 0 0000 00 0000 | 0056 789123 |
| SL | 0006 | 0 0000 00 5678 | 9123 000000 |
| RO | 0000 | 0 0000 00 5679 | 0000 000000 |
| STOP | 0000 | 0 0000 00 5679 | 0000 000000 |

The operation of the computer stops, but no information is lost. Operation is resumed at the next program step when the START button is pressed.

**ST** **STORE**
**000p 12 xxxx**
Store the contents of the A Register in **xxxx.**

**STC** **STORE, CLEAR**
**000p 02 xxxx**
Store the contents of the A Register in **xxxx**. Clear the A Register.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 0 7133 21 9821 | 4792 014910 |
| CAD | 1000 | 0 2222 22 2222 | 4792 014910 |
| ST | 1004 | 0 2222 22 2222 | 4792 014910 |
| AD | 1001 | 1 1111 11 1111 | 4792 014910 |
| STC | 1005 | 0 0000 00 0000 | 4792 014910 |

Memory cell 1004 will contain 0 2222 22 2222.
Memory cell 1005 will contain 1 1111 11 1111.

Manipulation instructions are provided to facilitate the effective use of arithmetic instructions during operation.

PROBLEM: A store wants to take advantage of a close-out sale to purchase 2,250 items at $10.00 each. There will be a shipping cost of $380.00. There is a $909.00 discount if the purchase is made on an 18-month contract at 6.5% interest. The store owner wants to know what his monthly principal and interest payments will be.

TO FIND: Monthly payments—principal and interest.

ASSUME: Information for purchase located in storage cells:

| | | |
|---|---|---|
| 1000 | 0 0000 00 2250 | (Quantity) |
| 1001 | 0 0000 00 1000 | (Unit Price) |
| 1002 | 0 0000 03 8000 | (Shipping Cost) |
| 1003 | 0 0000 09 0900 | (Discount) |
| 1004 | 0 0000 00 0065 | (Interest Rate) |
| 1005 | 0 0000 00 0018 | (Number of Payments) |

SOLUTION:

| Location Main | Location Loop | S | Control Digits | Operation No | Operation Alpha | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 0000 | 0 | | | | CR | 0000 | R = 0000 000000 |
| 0001 | 1 | | | | CAD | 1000 | A = 0‸0000 00 2250 |
| 0002 | 2 | | | | M | 1001 | A=ZERO, R= 0002 250000 |
| 0003 | 3 | | | | SL | 0010 | A= 0‸0002 25 00.00 |
| 0004 | 4 | | | | AD | 1002 | A= 0‸0002 28 80.00 |
| 0005 | 5 | | | | SU | 1003 | A= 0‸0002 19 71.00 |
| 0006 | 6 | | | | ST | 1006 | A= 0‸0002 19 71.00 |
| 0007 | 7 | | | | M | 1004 | A=ZERO R=0142 8.11500 |
| 0008 | 8 | | | | SL | 0007 | A= 0‸0000 14 28.11 R=5000 000000 |
| 0009 | 9 | | | | RO | 0000 | A=0‸0000 14 28.12, R=ZERO |
| 0010 | 0 | | | | SR | 0010 | A=ZERO, R=0000 1428.12 |
| 0011 | 1 | | | | DIV | 1005 | A=0‸0000 00 79.34, R=ZERO |
| 0012 | 2 | | | | STC | 1007 | A=ZERO STORING MONTHLY INTEREST |
| 0013 | 3 | | | | CAD | 1006 | A=0‸0002 19 71.00 |
| 0014 | 4 | | | | SR | 0010 | A=ZERO, R=0002 1971.00 |
| 0015 | 5 | | | | DIV | 1005 | A=0‸0000 12 20.61, R=0000 000002 |
| 0016 | 6 | | | | AD | 1007 | A=0‸0000 12 99.95 |
| 0017 | 7 | | | | STC | 1008 | A=ZERO, STORING 17 MONTHLY PAYMENTS. |
| 0018 | 8 | | | | SL | 0010 | A= 0‸0000 00 0002 |
| 0019 | 9 | | | | AD | 1008 | A= 0‸0000 12 99.97 |
| 0020 | | | | | STC | 1009 | A=ZERO STORING LAST PAYMENT |

ANSWER: Located in memory cells 1008 (17 monthly payments), and 1009 (last payment).

**BT4** — **BLOCK TO BAND 4**

**000p 34 xxxx**

Block transfer the contents of twenty consecutive main storage cells, beginning with **xxxx**, to the 4000 quick access band. Use **BT5 (35)** for the 5000 band, **BT6 (36)** for the 6000 band, and **BT7 (37)** for the 7000 band.

**BF4** — **BLOCK FROM BAND 4**

**000p 24 xxxx**

Block transfer the contents of the 4000 quick access band to twenty consecutive main storage cells, beginning with **xxxx**. Use **BF5 (25)** for the 5000 band, **BF6 (26)** for the 6000 band, and **BF7 (27)** for the 7000 band.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2179 430198 |
| $BT_4$ | 1000 | 1 6214 91 2721 | 2179 430198 |
| $BT_5$ | 1020 | 1 6214 91 2721 | 2179 430198 |
| $BF_4$ | 2660 | 1 6214 91 2721 | 2179 430198 |
| $BF_5$ | 2680 | 1 6214 91 2721 | 2179 430198 |

The contents of 1000-1019, band 4, and 2660-2679 are alike.

The contents of 1020-1039, band 5, and 2680-2699 are alike.

In blocking to a quick access band, main storage is unchanged and the previous contents of that band are completely erased.

In blocking from a quick access band to main storage, 20 words in main storage are erased and 20 new words are written. The quick access band remains unchanged.

**EX** — **EXTRACT**

**000p 63 xxxx**

Extract from the contents of the A Register by changing each digit in the A Register (including sign) to zero if the digit in the corresponding position in **xxxx** is zero. The digit in the A Register remains unchanged if the digit in the corresponding position in **xxxx** is one.

Assume cell 1000 = 1 1011 01 1101

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2179 430198 |
| EX | 1000 | 1 6014 01 2701 | 2179 430198 |

**CIRA** — **CIRCULATE A**

**000p 01 00nn**

Shift the contents (including sign) of the A Register **nn + 1** places left. The digits shifted out of the left end of the A Register re-enter the right end of the A Register in the same order.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2179 430198 |
| CIRA | 0006 | 2 7211 62 1491 | 2179 430198 |

**UA** — **UNIT ADJUST**

**000p 06 0000**

Increase by one the most significant position of the A Register if the digit in this position is even.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2179 430198 |
| UA | 0000 | 1 7214 91 2721 | 2179 430198 |
| UA | 0000 | 1 7214 91 2721 | 2179 430198 |

When the digit is odd, there is no change. The sign of the A Register is immaterial.

The block transfer instructions enable the programmer to place data in the quick access bands and thereby shorten operation time. The manipulation instructions presented on the opposite page are excellent for editing and separating parts of a word.

PROBLEM: A warehouse maintains a file of supplies. The data is filed in the following code:

O uuvw wx xyyy

where: *uu* is the Supplier code
*v* is the Color code
*ww* is the Warehouse Classification code
*xx* is the Assembly code
*yyy* is the Detail code

It has been determined that the Warehouse Classification code should be revised to contain three digits and that the color code is not necessary. The current files shall be assigned the Warehouse Classification code of lww (100 plus the current code number).

TO FIND: Revised file. (Only one file will be revised in the example program below.)

ASSUME: Example file for revision is located in cell 1000. The program is located in band 7.

SOLUTION:

| Location Main | Location Loop | S | Control Digits | Operation No | Operation Alpha | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 3000 | 7000 | | | | BT4 | 1000 | BLOCK CONTENTS OF CELL 1000 (FILE) TO 4000 |
| 3001 | 7001 | | | | CAD | 4000 | A= 0. u uvw wx xyyy. |
| 3002 | 7002 | | | | EX | 7009 | A= 0. u u0w wx xyyy. |
| 3003 | 7003 | | | | CIRA | 0001 | A= u. 0ww x xy yy0u. |
| 3004 | 7004 | | | | UA | 0000 | A = u. 1ww x xy yy0u. |
| 3005 | 7005 | | | | CIRA | 0008 | A= 0. uu1w wx xyyy. |
| 3006 | 7006 | | | | STC | 4000 | |
| 3007 | 7007 | | | | BF4 | 1000 | BLOCK REVISED FILE IN CELL 4000 TO CELL 1000 |
| 3008 | 7008 | | | | STOP | 0000 | |
| 3009 | 7009 | 1 | 11 01 | 11 | | 1111 | EXTRACT CONSTANT. |

## DECISION MAKING AND BRANCHING

**CC** **CHANGE CONDITIONALLY**
**000p 28 xxxx**
Overflow indicator ON: Change control to **xxxx.** Reset Overflow.
Overflow indicator OFF: Control continues in sequence.

**CCB** **CHANGE CONDITIONALLY, BLOCK**
**000p 38 xxxx**
Overflow indicator ON: Block transfer the contents of twenty consecutive main storage cells, beginning with **xxxx,** to the 7000 band. Change control to **70xx.** Reset Overflow.
Overflow indicator OFF: Control continues in sequence.

**CCR** **CHANGE CONDITIONALLY, RECORD**
**000p 29 xxxx**
Overflow indicator ON: Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the instruction next in sequence. Change control to xxxx. Reset Overflow.
Overflow indicator OFF: Control continues in sequence.

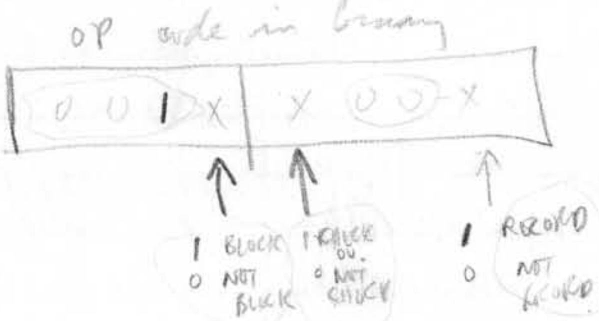
**CCBR** **CHANGE CONDITIONALLY, BLOCK, RECORD**
**000p 39 xxxx**
Overflow indicator ON: Block transfer the contents of twenty consecutive main storage cells, beginning with **xxxx,** to the 7000 band. Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the instruction next in sequence. Change control to 70xx. Reset Overflow.
Overflow indicator OFF: Control continues in sequence.

**CU** **CHANGE UNCONDITIONALLY**
**000p 20 xxxx**
Change control to **xxxx.**

**CUB** **CHANGE UNCONDITIONALLY, BLOCK**
**000p 30 xxxx**
Block transfer the contents of twenty consecutive main storage cells, beginning with **xxxx,** to the 7000 band. Change control to **70xx.**

**CUR** **CHANGE UNCONDITIONALLY, RECORD**
**000p 21 xxxx**
Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the instruction next in sequence. Change control to **xxxx.**

**CUBR** **CHANGE UNCONDITIONALLY, BLOCK, RECORD**
**000p 31 xxxx**
Block transfer the contents of twenty consecutive main storage cells, beginning with **xxxx,** to the 7000 band. Clear the R Register. Store in the four most significant positions of the R Register the address (as contained in the Control Counter) of the instruction next in sequence. Change control to **70xx.**



Change of control instructions enable the programmer to allow for possible exceptions during the execution of the program.

PROBLEM: A new car agency requests each salesman to report sales, in dollars, twice a month, and the number of demonstrations for customers made once a month. The sales for the first half of the month are located in storage cells 1000–1999. The sales for the last half of the month are located in storage cells 2000–2999. The number of demonstrations are located in storage cells 3000-3999. (All of the above are arranged according to salesman number; i.e., salesman no. 9, reported the information in 1009, 2009, 3009.)

TO FIND: Total dollar sales per salesman during the month, per customer demonstration.

ASSUME: For example purposes, use only salesman no. 10. (1010, 2010, 3010 cell nos.)

SOLUTION:

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0800 | 0800 | | | | CUB | 0801 | PLACE PROGRAM IN LOOP 7. |
| 0801 | 7001 | | | | BT4 | 1000 | FIRST HALF MONTH SALES. → BAND 4 |
| 0802 | 7002 | | | | BT5 | 2000 | SECOND HALF MONTH SALES → BAND 5 |
| 0803 | 7003 | | | | BT6 | 3000 | DEMONSTRATIONS. → BAND 6 |
| 0804 | 7004 | | | | CR | 0000 | |
| 0805 | 7005 | | | | CAD | 4010 | } FIND TOTAL SALES |
| 0806 | 7006 | | | | AD | 5010 | } |
| 0807 | 7007 | | | | CC | 7015 | CHECK FOR OVERFLOW |
| 0808 | 7008 | | | | ST | 4010 | |
| 0809 | 7009 | | | | DIV | 6010 | SALES ÷ DEMONSTRATIONS |
| 0810 | 7010 | | | | CCB | * | DIVIDEND ADJUSTMENT. |
| 0811 | 7011 | | | | STC | 6010 | $/SALESMAN/DEMONSTRATION. |
| 0812 | 7012 | | | | BF4 | 1000 | } ANSWERS TO MAIN STORAGE. |
| 0813 | 7013 | | | | BF6 | 3000 | } |
| 0814 | 7014 | | | | STOP | 0000 | OR CONTINUE TO NEXT SALESMAN |
| 0815 | 7015 | | | | SR | 0001 | } |
| 0816 | 7016 | | | | UA | 0000 | } ADDITION ADJUSTMENT. |
| 0817 | 7017 | | | | CU | 7008 | } |
| 8 | 8 | | | | | | |
| 9 | 9 | | | | | | |
| | | | | | | | |

Additional Remarks * AT THE END OF THE DIVIDEND ADJUSTMENT PROGRAM, THERE WILL BE A CUB 0811 INSTRUCTION TO RE-ENTER ABOVE PROGRAM.

**NOR** **NORMALIZE (CHANGE ON ZERO)**
**000p 15 xxxx**

**(a) If the content of the A Register is not zero, shift the twenty digits in the A Register and the R Register left until the most significant position in the A Register is not zero. The sign does not move. Record the number of shifts in the Special Counter.**

**(b) If the content of the A Register is zero, shift the contents of the R Register left into the A Register, clear the R Register, and change control to xxxx. The sign does not move.** SC ← 10

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 0006 21 4912 | 2179 430198 |
| NOR | 3172 | 1 6214 91 2217 | 9430 198000 |

Special Counter is 3. Control continues sequentially.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 0000 00 0000 | 2179 430198 |
| NOR | 3172 | 1 2179 43 0198 | 0000 000000 |

Special Counter is 10.
Control transfers to cell 3172.

**CNZ** **CHANGE ON NON-ZERO**
**000p 04 xxxx**

Test the contents of the A Register (not the sign) for zero.
(a) If the A Register setting is zero, set the sign of the A Register to zero and continue control in sequence.
(b) If the A Register setting is not zero, change control to **xxxx.**

A non-zero number in the A Register is unaltered.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2179 430198 |
| CNZ | 3172 | 1 6214 91 2721 | 2179 430198 |

Control transfers to cell 3172.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 1 0000 00 0000 | 2179 430198 |
| CNZ | 3172 | 0 0000 00 0000 | 2179 430198 |

Control continues sequentially.

**ADSC** **ADD SPECIAL COUNTER**
**000p 16 0000**

**Add the contents of the Special Counter to the least significant position of the A Register.**

The condition of Overflow is possible with this instruction if the A Register is at least 9999 99 9991. If the Special Counter is 10, the machine will stop on "forbidden combination" without performing the addition.

**SUSC** **SUBTRACT SPECIAL COUNTER**
**000p 17 0000**

**Subtract the contents of the Special Counter from the least significant position of the A Register.**

(See Add Special Counter.)

Assume Special Counter is 2.

| Program | | A Register | R Register |
|---|---|---|---|
| | | 0 0000 00 0050 | 2179 430198 |
| ADSC | 0000 | 0 0000 00 0052 | 2179 430198 |
| SUSC | 0000 | 0 0000 00 0050 | 2179 430198 |

**OSGD** **OVERFLOW ON SIGN DIFFERENCE**
**000p 73 xxxx**

**If the sign of the A Register differs from the sign of xxxx, Overflow indicator ON.**

The contents of the A Register is not affected. This instruction must *always* be followed by a Conditional Change instruction. or else what

how long does SC stay around?? change

Assume cell 3710 is positive.
Assume cell 3720 is negative.

| Cell | Program | A Register | R Register |
|---|---|---|---|
| | | 0 6214 91 2721 | 2179 430198 |
| 0000 | OSGD3710 | 0 6214 91 2721 | 2179 430198 |
| 0001 | CC 0004 | 0 6214 91 2721 | 2179 430198 |
| 0002 | OSGD3720 | 0 6214 91 2721 | 2179 430198 |
| 0003 | CC 0005 | 0 6214 91 2721 | 2179 430198 |
| 0004 | STOP 0000 | 0 6214 91 2721 | 2179 430198 |
| 0005 | **If the sign of the A register is negative (−), control transfers to cell 0005.** | | |

**If the sign of the A register is positive (+), control continues to cell 0004.**

Decision making instructions are used to determine the nature of a number. As a result of this determination the next steps are performed, either in normal sequence or by branching to another part of the program.

PROBLEM: An inventory count of various parts is located in storage cells 1000–1999. If a count is 100 or less, that part should be re-ordered.

TO FIND: The nature of the inventory count for each part, and determine whether or not it is necessary to re-order.

NOTE: Two methods of solution are shown.

ASSUME: The inventory count located in cell 1000 shall be used as an example.

SOLUTION 1:

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0640 | 7000 | | | | BT4 | 1000 | BLK. INVENTORY TO 4000 BAND |
| 0641 | 7001 | | | | BT6 | ---- | RE-ORDER PROGRAM. |
| 0642 | 7002 | | | | CAD | 4000 | FIRST INVENTORY COUNT. |
| 0643 | 7003 | | | | SU | 7009 | BALANCE TO MAINTAIN. |
| 0644 | 7004 | | | | CNZ | 7006 | CHECK AMT. INVENTORY |
| 0645 | 7005 | | | | CU | 6000 | RE-ORDER PROGRAM. : |
| 0646 | 7006 | | | | OSGD | 7006 | ← OWN LOCATION ← ← slow? |
| 0647 | 7007 | | | | CC | 6000 | RE-ORDER PROGRAM. |
| 0648 | 7008 | | | | STOP | 0000 | OR CONTINUE TO NEXT COUNT. |
| 0649 | 7009 | 0 | 0000 | 00 | | 0100 | CONSTANT. |

SOLUTION 2:

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0640 | 7000 | | | | BT4 | 1000 | BLK. INVENT. TO 4000 BAND |
| 0641 | 7001 | | | | BT6 | ---- | RE-ORDER PROGRAM |
| 0642 | 7002 | | | | CAD | 4000 | FIRST INVENTORY COUNT |
| 0643 | 7003 | | | | NOR | 6000 | RE-ORDER PROGRAM, COUNT=0 |
| 0644 | 7004 | | | | CAD | 7012 | |
| 0645 | 7005 | | | | SUSC | 0000 | |
| 0646 | 7006 | | | | OSGD | 7006 | CHECK IF COUNT UNDER 100 |
| 0647 | 7007 | | | | CC | 6000 | RE-ORDER PROGRAM, COUNT UNDER 100 |
| 0648 | 7008 | | | | CAD | 4000 | FIRST INVENTORY COUNT |
| 0649 | 7009 | | | | SU | 7013 | BALANCE TO MAINTAIN |
| 0650 | 7010 | | | | NOR | 6000 | RE-ORDER PROGRAM, COUNT=100 |
| 0651 | 7011 | | | | STOP | 0000 | OR CONTINUE TO NEXT COUNT |
| 0652 | 7012 | 0 | 0000 | 00 | | 0008 | CONSTANT |
| 0653 | 7013 | 0 | 0000 | 00 | | 0100 | CONSTANT |
| 4 | 4 | | | | | | |
| 5 | 5 | | | | | | |
| 6 | 6 | | | | | | |
| 7 | 7 | | | | | | |
| 8 | 8 | | | | | | |
| 9 | 9 | | | | | | |
| | | | | | | | |

## USING THE B REGISTER

Assume cell 3170 contains 0 1279 42 0019.

**SB** — **SET B**

**000p 72 xxxx**

Set the B Register to the value of the four least significant positions of **xxxx.**

| Program | | A Register | B Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 2174 |
| SB | 3170 | 1 6214 91 2721 | 0019 |

**IB** — **INCREASE B**

**000p 32 0000**

Add one to the contents of the B Register.

| Program | | A Register | B Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 0019 |
| IB | 0000 | 1 6214 91 2721 | 0020 |

**BA** — **B TO A**

**000p 11 0000**

Clear the A Register. Add the contents of the B Register.

| Program | | A Register | B Register |
|---|---|---|---|
| | | 1 6214 91 2721 | 0020 |
| BA | 0000 | 0 0000 00 0020 | 0020 |

**DB** — **DECREASE B**

**000p 22 xxxx**

Subtract one from the contents of the B Register.
(a) If the new B Register setting is 9999 (0000 − 1), control continues in squence.
(b) If the new B Register setting is not 9999, change control to **xxxx.**

| Cell | Program | | A Register | B Register |
|---|---|---|---|---|
| | | | 1 6214 91 2721 | 0001 |
| 1000 | DB | 1004 | 1 6214 91 2721 | 0000 |

Control will transfer to cell 1004.

| Cell | Program | | A Register | B Register |
|---|---|---|---|---|
| | | | 1 6214 91 2721 | 0000 |
| 1000 | DB | 1004 | 1 6214 91 2721 | 9999 |
| 1001 | STOP 0000 | | | |

The B Register is one of the most valuable tools available to the programmer. Use of the instructions for the B Register is as varied as the problems in the business world. The use of the instructions is shown below. For a more refined use of the B Register, refer to the discussions of various techniques on the following pages.

PROBLEM: A manufacturing concern has an hourly payroll of 1000 employees. A daily computation of hours and earnings is required by employee number. Daily computations are accumulated for the weekly payroll.

TO FIND: Accumulate hours and amount by employee number. Current daily totals are readily available if required. Totals are accumulated for weekly payroll processing.

ASSUME: Employee numbers range from 0000 through 0999. In the cell corresponding to the employee number is the current day's record for that employee. That is, in cell 0000 will be found the following information for employee 0000: his daily hours worked, and his wage rate. This information is stored in the following manner: xxx (hours), yyy(rate), 0000.

The accumulated hours and pay for each employee will be stored in cells 1000 through 1999, with the total for employee 0000 located in cell 1000, that of employee 0001 in cell 1001, etc. This information is stored in the following manner: xxxx (hours), yyyyyy(amount of pay).

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0 | 6000 | | | | SB | 6017 | |
| 1 | 6001 | 1 | | | CAD | (0000) | CURRENT HOURS AND RATE |
| 2 | 6002 | | | | EX | 6018 | EXTRACT AND STORE |
| 3 | 6003 | | | | STC | 2000 | RATE |
| 4 | 6004 | 1 | | | CAD | (0000) | CURRENT HOURS AND RATE |
| 5 | 6005 | | | | EX | 6019 | EXTRACT AND STORE |
| 6 | 6006 | | | | ST | 2001 | HOURS |
| 7 | 6007 | | | | SR | 0002 | SCALE |
| 8 | 6008 | | | | MRO | 2000 | CURRENT AMOUNT |
| 9 | 6009 | | | | STC | 2000 | ↳ STORE |
| 0 | 6010 | | | | CAD | 2001 | CURRENT HOURS |
| 1 | 6011 | | | | SR | 0001 | SCALE TO ADD HOURS |
| 2 | 6012 | | | | AD | 2000 | ADD CURRENT AMOUNT |
| 3 | 6013 | 1 | | | ST | (0000) | STORE CURRENT HRS. AND AMOUNT |
| 4 | 6014 | 1 | | | AD | (1000) | ACCUMULATED HRS. AND AMOUNT |
| 5 | 6015 | 1 | | | STC | (1000) | ↳ STORE |
| 6 | 6016 | | | | DB | 6001 | TALLY |
| 7 | 6017 | | | | STOP | 0999 | MODIFICATION CONSTANT |
| 8 | 6018 | | 0001 | 11 | | 0000 | CONSTANT FOR EXTRACTING RATE |
| 9 | 6019 | | 1110 | 00 | | 0000 | CONSTANT FOR EXTRACTING HOURS |

## SCALING

Scaling is an important part of a program. It is a problem for the programmer only, not the computer. The computer will always handle the data as ten digit numbers. The programmer, then, has the problem of keeping track of the movement of the decimal point.

In addition and subtraction, scaling must locate the digits in a number so that the decimal points are in line. For example, 52.9 plus 5.32 is written

```
  52.9
+  5.32
  -----
  58.22
```

No further scaling is necessary for addition or subtraction.

However, multiplication and division change the position of the decimal point. When two 10-digit numbers are multiplied, the product is a 20-digit number in the A Register and the R Register. In division, the dividend is a 20-digit number, the divisor is a 10-digit number, and the quotient is a 10-digit number appearing in the A Register.

(1) In multiplication, the number of places to the right of the decimal point in the product is the sum of the number of decimal places to the right of the decimal point in the multiplicand and the multiplier.

(2) In division, the number of places to the right of the decimal point in the quotient is the number of places to the right of the decimal point in the dividend (A and R) minus the number of decimal places to the right of the decimal point in the divisor.

Examples:

(1) 0 0002 25 0000 multiplied by 0 1000 00 0000 is
0 0000 22 5000 0000 000000
(A Register) (R Register)

The number represented by 0 0002 25 0000 is actually 2250.

The number represented by 0 1000 00 0000 is actually 10.

Therefore, the number of decimal places to the right of the decimal point is:

Mutliplicand 3
Multiplier 8
Product (A and R) 11

The product, therefore, represented by
0 0000 22 5000 0000 000000 is actually 22,500.

(2) 0 0000 14 2812 0000 000000 divided by 0 0000 18 0000 is 0 7934 00 0000.

The number represented by 0 0000 14 2812 0000 000000 is actually 1428.12.

The number represented by 0 0000 18 0000 is actually 18.

Therefore, the number of decimal places to the right of the decimal point is:

Dividend 12
Divisor 4
Quotient (A) 8

The quotient, therefore, represented by
0 7934 00 0000 is actually 79.34.

## ADDRESS MODIFICATION AND CYCLING

In using a stored program machine such as the Burroughs 205, it is often necessary to provide a means of modifying instructions in order that a small program may be used repeatedly, but with different data each time the calculation is performed. To accomplish this effect, it is required that the instruction addresses which refer to the data be changed on each cycle.

The Burroughs 205 provides two means for the programmer to accomplish address modification; i.e., B modification and programmed modification.

**B modification of instructions:**

Any instruction with a 1 in the first control digit (numerical minus sign) will have the contents of the B Register added to the address of the instruction before it is executed. Example: Sum 500 sales stored in cells 0000-0499.

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0 | 7000 | 0 | 0000 | | SB | 7004 | SET BREGISTER TO 0498 |
| 1 | 7001 | 0 | 0000 | | CAD | 0499 | CLEAR A REGISTER AND ADD LAST SALE |
| 2 | 7002 | 1 | 0000 | | AD | [0000] | SUM SALES |
| 3 | 7003 | 0 | 0000 | | DB | 7002 | TALLY |
| 4 | 7004 | 0 | 0000 | | STOP | 0498 | COMPUTER WILL STOP WHEN BREGISTER HAS COUNTED DOWN 499 TIMES |
| 5 | 5 | | | | | | |

In this case, the B Register serves two functions; i.e., automatic address modifier and tallying device, for which DB performs both functions.

**Programmed modification:**

Any instruction may be inserted into the A Register. It appears to the computer just as a number, and any other number may be added or subtracted, thereby modifying the instruction. Usually, only the instruction address is modified. Example: 500 sales are stored in 0000-0499 which are to be summed.

The instruction in cell 7001 was modified on each cycle to serve as a tally device and refer to the next data item.

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0 | 7000 | 0 | 0000 | | CAD | 0000 | FIRST SALE |
| 1 | 7001 | 0 | 5010 | | AD | [0001] | ADD NEXT SALE |
| 2 | 7002 | 0 | 0000 | | STC | 7010 | STORE PARTIAL SUM TEMPORARILY |
| 3 | 7003 | 0 | 0000 | | CAD | 7001 | } MODIFY ADD COMMAND |
| 4 | 7004 | 0 | 0000 | | AD | 7009 | } |
| 5 | 7005 | 0 | 0000 | | CC | 7011 | TEST FOR COMPLETION OF PROBLEM |
| 6 | 7006 | 0 | 0000 | | STC | 7001 | MODIFIED ADD COMMAND |
| 7 | 7007 | 0 | 0000 | | CAD | 7010 | PARTIAL SUM |
| 8 | 7008 | 0 | 0000 | | CU | 7001 | REPEAT CYCLE |
| 9 | 7009 | 0 | 0010 | 00 | | 0001 | TALLY AND MODIFYING CONSTANT |
| 0 | 7010 | 0 | 0000 | 00 | | 0000 | TEMPORARY-STORAGE CELL |
| 1 | 7011 | 0 | 0000 | | STOP | 0000 | PROBLEM COMPLETED |

## USE OF THE QUICK ACCESS BANDS

**General:**

The sole purpose of the quick access bands is to increase the speed of computer operation. Normally, all instructions are executed from one of these bands, and, if possible, the data groups used in a section of the program are stored in any of the bands. Only under difficult situations should the program sequence be shifted to main storage where the speed of operation is ten times as lengthy. The blocking instructions are provided to facilitate the use of the quick access bands.

## GENERAL RULES FOR USE OF THE QUICK ACCESS BANDS

**PROGRAMMING**: Experience shows that the 7000 band receives greatest use in programming because of the CUB instruction's dual function of blocking from main storage to the 7000 band and then transferring control to the first instruction of the 20 words blocked. Of course, the program will operate at the same speed when using any of the three remaining bands. Normally programs are broken into segments of 20 instructions or less such that discrete small groups of program steps are used in the bands as a unit. This does **not** mean that every twentieth step must be a CUB instruction but that a CUB or CU or any other transfer instruction must appear at least once in every 20 instructions as used in one band.

**DATA STORAGE**: For considerations of speed, it is desirable to store data and constants in the quick access bands along with the program. Generally, the program which has the least number of main storage instruction addresses will be the most rapid.

**ADDRESSING**: The accepted practice for designating quick access band instruction addresses is to keep them within the ranges 4000-4019, 5000-5019, 6000-6019, 7000-7019.

## DATA EDITING

Usually in business problems, arithmetic computation is not complex. The general case is the making of logical decisions and the preparation of data in proper format for these decisions. The Burroughs 205 provides five very convenient instructions for this purpose, viz., CIRA, EX, SL, SR, UA.

CIRA. This instruction provides a means of "opening" a word any number of spaces and inserting additional digits. CIRA is really a shift left instruction that applies to the A Register only.

Example: Expand a six-digit account number to eight digits, inserting 00 between the second and third digits.

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0 | 7000 | 0 | 0000 | | CAD | 7006 | A = 0. 0000 XX YYYY |
| 1 | 7001 | 0 | 0000 | | SR | 0004 | A = 0. 0000 00 00 XX<br>R = YYYY 000000 |
| 2 | 7002 | 0 | 0000 | | CIRA | 0001 | A = 0. 0000 00 XX00<br>R = YYYY 000000 |
| 3 | 7003 | 0 | 0000 | | SL | 0004 | A = 0. 00XX 00 YYYY<br>R = 0000 000000 |
| 4 | 7004 | 0 | 0000 | | STC | 7007 | |
| 5 | 7005 | 0 | 0000 | | STOP | 0000 | |
| 6 | 7006 | 0 | 0000 | XX | | YYYY | ACCOUNT NUMBER (OLD) |
| 7 | 7007 | 0 | 00XX | 00 | | YYYY | ACCOUNT NUMBER (NEW) |

EXTRACT. This instruction is of great value in breaking down coded numbers (part numbers, account numbers, policy numbers, etc.) such that certain logical decisions requiring a particular type of processing may be made easily.

Example: Check a nine digit part number to determine whether the seventh digit (warehouse class code) is a 7. If so, increase unit cost by 1%.

| Location | | | Control | Operation | | Operand | Remarks |
|---|---|---|---|---|---|---|---|
| Main | Loop | S | Digits | No | Alpha | Address | |
| | | | | | | | |
| 0 | 7000 | 0 | 0000 | | CAD | 7014 | A = 0.0XXX XX XXXY.·PART.* |
| 1 | 7001 | 0 | 0000 | | EX | 7011 | A = 0.000X 00 0000·WARE. CL.CODE |
| 2 | 7002 | 0 | 0000 | | SU | 7010 | } |
| 3 | 7003 | 0 | 0000 | | CNZ | 7009 | } CHECK FOR "7." |
| 4 | 7004 | 0 | 0000 | | CAD | 7012 | A = 0.0000 00 XXXX. UNIT COST |
| 5 | 7005 | 0 | 0000 | | SR | 0002 | A = 0.0000 00 00XX. 1%. |
| 6 | 7006 | 0 | 0000 | | RO | 0000 | |
| 7 | 7007 | 0 | 0000 | | AD | 7012 | A = 0.0000 00 XXXX NEW UNIT COST |
| 8 | 7008 | 0 | 0000 | | STC | 7013 | |
| 9 | 7009 | 0 | 0000 | | STOP | 0000 | |
| 0 | 7010 | 0 | 0007 | 00 | | 0000 | CONSTANT |
| 1 | 7011 | 0 | 0001 | 00 | | 0000 | CONSTANT |
| 2 | 7012 | 0 | 0000 | 00 | | XXXX | OLD UNIT COST |
| 3 | 7013 | 0 | 0000 | 00 | | XXXX | NEW UNIT COST |
| 4 | 7014 | 0 | 0XXX | XX | | XXXX | PART NO. |

## TABLE LOOK-UP

In general, table look-up with the Burroughs 205 is a simple and rapid machine process. Basically, the problem is this: a file exists in storage, and a particular item from the file is called for by each input item. If the file is lengthy, the search for the desired item may be time-consuming unless it is efficiently programmed.
There are several methods of performing the search:

1. Comparing item by item, starting at the head of the table. This is the slowest but simplest method.
2. Comparing by starting at the middle of the table and continuously eliminating half the remaining possibilities. This is Binary Search and is usually fairly rapid. This technique should be used when the third method is not applicable.
3. The most used method, a particularly efficient one with the Burroughs 205, is this: construct the file in such a way that the location of each item is correlated to the input number itself. If this can be done, the Burroughs 205 can use the input number (part number, employee number, etc.) to specify the address of the item in the table. Then the table entry can be called up directly, and no search for it need be performed.
   In the simplest case, the input number can directly identify the storage location in the table. An example of this is the following.

PROBLEM: A bank has 3000 checking accounts. It keeps the current balance for each account in a file in main storage. Transactions are to be entered in groups as they are processed. Each account is assigned a number from 0000 to 2999, and its current balance is kept in the corresponding storage location. Thus account number 1504 has its current balance recorded in location 1504.

100 transactions have been read into storage locations 3500-3599. Each is recorded in the following form:

± xxxxx0yyyy
where ± xxxxx = transaction
yyyy = account number

The table look-up in this example was done without any searching, because the account number could be used to directly indicate the proper location of the proper current balance.

In most practical cases, the identifying number (account, employee, or part number) does not directly correspond to a storage address. The storage location corresponding to an input item must be computed or looked up as a separate operation before the table look-up can be carried out. In some cases, an arithmetic computation can convert the item into the appropriate address. In others, it is necessary to use a "dictionary" to find the storage location corresponding to an input item. In either case, the program operating time is somewhat increased. However, this method will still be much faster than an item-by-item search of the main file or a binary search.

| Location Main | Location Loop | S | Control Digits | Operation No | Operation Alpha | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 0 | 7000 | | 9000 | | CAD | 3500 | A = xxxxx0yyyy |
| 1 | 7001 | | | | ST | 7013 | |
| 2 | 7002 | | | | SB | 7013 | B = yyyy |
| 3 | 7003 | | | | SR | 0005 | A = 00000xxxxx |
| 4 | 7004 | 1 | 0000 | | AD | 0000 | Add balance from file |
| 5 | 7005 | 1 | 0000 | | ST | 0000 | Store new balance in file |
| 6 | 7006 | | | | CAD | 7000 | |
| 7 | 7007 | | | | AD | 7012 | |
| 8 | 7008 | | | | CC | 7011 | modify and Tally order |
| 9 | 7009 | | | | STC | 7000 | in 7000 |
| 0 | 7010 | | | | CU | 7000 | |
| 1 | 7011 | | | | STOP | 0000 | |
| 2 | 7012 | 0 | 0010 | | 00 | 0001 | Constant to modify 7000 |

PROBLEM: A bank keeps current balances for checking accounts in a file in main storage. The account numbers vary from 00000 to 99999, but only 1500 accounts exist. As before, the problem is to post transactions in the current balance file.

In this case, there is no correspondence between account number and storage location. For example, the current balance for account number 14708 may be in storage location 1695, while 1696 contains the current balance for account number 35614. The file is maintained in account number sequence however. A dictionary must be provided which will relate the account number to the location where the current balance is stored.

The solution to this problem, as given below, is fairly complex. However, it shows the Burroughs 205 performing a practical operation at high efficiency, and demonstrates the power of this table look-up technique.

SOLUTION: Table #2 is composed of 1500 current balances stored in locations 1600-3099, and Table #1 is composed of 1500 account numbers stored in locations 0100-1599. With each account number is stored the address of its balance (Table #2) so that the entry has the following form.

| A Table #1 Entry | ± 0 dddd xxxxx<br>dddd = address of current balance in Table #2.<br>xxxxx = account number |
|---|---|

The table of account numbers, Table #1, is arranged in ascending sequence. A third table, Table #3, is constructed, which consists of 100 BT4 instructions referring to Table #1. The first two digits of the account number will correspond to a storage cell in Table #3. The instruction address of the BT4 instruction in that cell will refer to Table #1. This table is stored in 0000-0099, and makes possible a very rapid look-up of the account number, in the following way.

1. An input account number is in 5000, and the amount to be posted in 5001.
2. The first two digits of the account number call for the appropriate BT4 order from the Table #3 in 0000-0099.
3. A BT4 instructions is executed, bringing a block of 20 account numbers from Table #1 into band 4.
4. The 20 account numbers are searched by sequential comparison, and the desired account number is found. This is the only actual search necessary.
5. The address accompanying the matched account number is used to refer to the proper cell in the Table #2.
6. The posting is performed, and the operation is complete.

| Table #3 | Table #1 | Table #2 |
|---|---|---|
| Cells 0000-0099 | Cells 0100-1599 | Cells 1600-3099 |
| Table 1 BT4 addresses. | Account numbers and Table 2 addresses. | Current balances. |

| SECTION | NO. | SUBJECT | DATE | PAGE |
|---|---|---|---|---|
| PROGRAMMING | 2 | MEMORY SEARCH FEATURE OF THE 205 | 10/15/59 | 1 |

Two new commands have been added to the command list of the Burroughs 205 scheduled for delivery to the Eaton Manufacturing Company. These commands and their characteristics are as follows:

1. MEMORY SEARCH HIGH

MSH 0 0000 85 aaaa

A. Begin searching memory at aaaa for the absolute value of the number in the A register. If this number is found, insert it (including sign) into the A register, recording its address in the four most significant positions of the R register.

B. If the exact number is not present, but a higher number is found, insert the higher number (including sign) into the A register and record its address in the four most significant positions of the R register.

C. If neither the exact number nor a higher number is found, continue search through location 3999 after which set the overflow toggle to "ON".

2. MEMORY SEARCH EQUAL

MSE 0 0000 87 aaaa

A. Begin searching memory at aaaa for the absolute value of the number in the A register. If the number is found insert it (including sign) in the A register and record it's address in the four most significant positions of the R register.

B. If the exact number is not found after searching location 3999, set the overflow toggle to "ON".

3. SPECIAL CHARACTERISTICS

A. The operand addresses of both commands (aaaa) listed above Must be main memory locations (0000-3999).

B. Neither of the commands above places significance on sign position during search, either in the A register or in memory. All comparisons are made on absolute values.

3. SPECIAL CHARACTERISTICS

One word is compared in one word time, for a total of 340 milliseconds to search all 4000 locations of memory.

4. PROGRAMMING TIPS

A. If the file of key words is arranged in numerical sequence, the speed of memory searches can be greatly increased as follows: After each successfuk memory search, change the address portion of the MS command to equal the address of the mate found by the preceding memory search, thereby avoiding the progressive lengthening of MS operations.

B. If a mate or mate high for the key word does not exist in main memory, the overflow toggle will be set and the key word will remain in the A register at the end of the MS operation. The 28 command (which should follow every MS command) will cause a control change, which can be utilized to store the A register in an area of memory reserved for the accumulation of key words for which there are no mates in memory. The accumulated key words can be handled as special cases at the end of the program.

5. TECHNICAL DESCRIPTION

The memory search operation can be divided into four distinct phases plus the overflow condition.

A. Memory Scan - Compare A register with words in memory and check.

B. Interchange - Interchange contents of address register with contents of control counter.

C. Record - Transfer contents of control counter to first four positions of R register R1 through R4.

D. D to A - Clear A register and transfer contents of the D register (including sign) to the A register.

E. Overflow - Set overflow toggle on if a mate or a mate high is not found in main memory. Do not clear the key word from the A register.

| Location Main | Location Loop | S | Control Digits | Operation No | Operation Alpha | Operand Address | Remarks |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| 0 | 698 0 | | | | CAD | 5000 | GIVEN ACCOUNT NUMBER A=0. 0000 0XXYYY |
| 1 | 698 1 | | | | SR | 0003 | A=0.0000 00 00XX |
| 2 | 698 2 | | | | AD | 7001 | A=0.0000 64 00XX (CAD 00XX) |
| 3 | 698 3 | | | | STC | 6984 | |
| 4 | 698 4 | | | | [CAD | 00XX] | COMMAND MADE UP BY PRECEDING THREE COMMANDS |
| 5 | 698 5 | | | | STC | 6986 | (6986)=0.0000BT4 (0100-1599) |
| 6 | 698 6 | | | | [BT4 | ----] | |
| 7 | 698 7 | | | | SB | 7000 | SET (B) to 19 |
| 8 | 698 8 | | | | CAD | 4000 | A=0dddd XXXXX |
| 9 | 698 9 | | | | CR | 0000 | —0— |
| 0 | 699 0 | | | | SL | 0005 | A=0.XXXXX 00000 R=00000 0dddd |
| 1 | 699 1 | | | | CIRA | 0005 | A=0 00000 XXXXX TABLE LOOK UP |
| 2 | 699 2 | | | | SU | 5000 | |
| 3 | 699 3 | | | | CNZ | 6999 | |
| 4 | 699 4 | | | | SL | 0010 | A=0.0000 00 dddd |
| 5 | 699 5 | | | | AD | 7001 | A=0.0000 64 dddd |
| 6 | 699 6 | | | | ST | 7004 | |
| 7 | 699 7 | | | | CU | 7002 | |
| 8 | 699 8 | 0 | 0000 | 62 | | 0000 | CONSTANT |
| 9 | 699 9 | | | | DB | 6988 | TALLY |
| 0 | 700 0 | | | | STOP | 0019 | VALUE NOT IN TABLE ACCOUNT NUMBER INCORRECT |
| 1 | 700 1 | 0 | 0000 | 64 | | 0000 | CONSTANT 64 = CAD |
| 2 | 700 2 | | | | SU | 6998 | A=0.0000 02----; 02=STC |
| 3 | 700 3 | | | | STC | 7006 | |
| 4 | 700 4 | | | | [CAD | ---] | MADE UP BY COMMAND IN 6996 |
| 5 | 700 5 | | | | AD | 5001 | POST AMOUNT |
| 6 | 700 6 | | | | [STC | ----] | MADE UP BY COMMAND IN 7003 |
| 7 | 700 7 | | | | STOP | 0000 | |
| 8 | 8 | | | | | | |

# NOTES

# NOTES

```
777 FAST DRUM ZERO. USES 4000 LOOP. ZEROES 6000-6019 AND 0000-3999 IN 1.1 SEC.
666608104D40240000020402600006440350000224017100000260100000072402740810444016
6666081044403010000B60100100002600601000026002000000072403500000124035000000754034
66660002040160000000039000000000010000810441000000000224024100000260180100000260140
```

# NOTES

**BURROUGHS**

ELECTRONIC
DATA-
PROCESSING
SYSTEMS

Burroughs Corporation

ELECTRODATA DIVISION
PASADENA, CALIFORNIA