IPC
Instructo Paper Computer

OPERATOR'S MANUAL

a real programmable paper computer

INSTRUCTO / McGraw-Hill

No. 460
(No. 470 package of 10 Computers)

Dedications and Acknowledgments

This work is dedicated to a small group of people who are in many ways responsible for its creation.

To my Mom and Dad, who somehow managed to guide me through the younger years of my life, and made sure that I at least finished high school at a time when I cared less.

To my dear wife, Helene, whose limitless capacity to absorb written material was not only instrumental in my coming to Lower Moreland Middle School, but in my locating Instructo/McGraw-Hill as a publisher for this work.

To my children, Freddy and Diane, who often wonder what Dad is writing about most of the time.

To the administrators of Lower Moreland Middle School, in Huntingdon Valley, Pennsylvania, who have afforded me the degree of academic freedom necessary to create a work such as this, and to Dr. Barbara Flexer, a colleague of mine, who inspired me through our many mathematical discussions, and who did a great deal of proofreading on some of the early manuscripts.

OPERATOR'S MANUAL

Written and Developed by
Fred C. Matt

INSTRUCTO / McGraw-Hill
Paoli, Pennsylvania 19301

ERRATA

The operation of the Paper Computer is not affected by the incorrect page numbers printed in your Operator's Manual.
The correct page numbers are circled below.

Page 3

Program instructions, using the Mnemonic List, will direct the operator through various exercises which will result in meaningful information being printed on one of the Output strips. For more detailed step by step directions, see pages 9 and 10 in this book.

3. *Cut five 1½" strips from an 8½" x 11" blank sheet of paper or use the strips provided on page 15. Insert one each in the slots of* **Input B, Register A, Register B, Output A, Output B.** *(See diagram.)*
4. *Attach a* **Program** *(Programs are located on pages 17 to 24) and a* **Main Storage Unit** *sheet (page 13) to the computer by using four paper clips as shown below. (The Program page must be folded in half lengthwise.)*

Page 9

RUNNING A PRACTICE PROGRAM — STEP BY STEP

To learn how your computer operates, follow the directions below, checking off each one as you complete it. Before you begin, cut out or reproduce (if multiple copies are desired), the **Mnemonic List** (pages 11 and 12). The **Mnemonic List** should be kept with the computer for easy reference.

____ 1. Cut out or reproduce page 17. Fold it in half and clip Practice Program A into the **Program Storage Unit** of your computer. Read the directions at the top of the program and return here to check off this step before you continue.

Page 10

____ 25. Run Practice Program B. If you have trouble doing Program B, check the Flow Chart on page 8, or the TO RUN IPC directions on this page.

Page 17

Practice Program A

Practice Program A gives you an opportunity to see just how easy it is to operate your paper computer. Fold this page down the middle so only Practice Program A shows, and place it in the Program Storage Unit of your computer. Then, go back to where you were on page 9.

CONTENTS

Project Editor/Manager — Eileen F. Moyer

Editor — Carole F. Charters

Graphic Designer — Carole Smith

ISBN 07-530136-9

INTRODUCTION TO INSTRUCTO PAPER COMPUTER (IPC)

Welcome to what may be your first experience with computers and computer programming. What you are about to learn will be challenging and rewarding. If you learn it well, and if you really understand how to program and operate IPC, then you will be ready to start learning about the real electronic machines that control so much of the world we live in. Their operation is not so different from that of your Paper Computer. Their main advantage is that they can do millions of operations each second with very few, if any, errors.

In the world of the future most people will be computer literate. It will be important for people to know how to operate and use computers, because computers will affect all aspects of our lives. The future is yours, and this computer is a first step toward computer literacy.

OVERVIEW OF IPC

IPC has been designed to give students an educational tool with which they can use basic skills to learn how computers function.

This manual will be your guide to understanding how IPC operates. The components of the manual include Programs, Mnemonic List, detailed step by step operating procedures and flow chart giving a simplified explanation.

Also included are Main Storage Unit sheet, Register tapes and information about electronic computers. All pages may be reproduced for class distribution, or cut out for use by an individual.

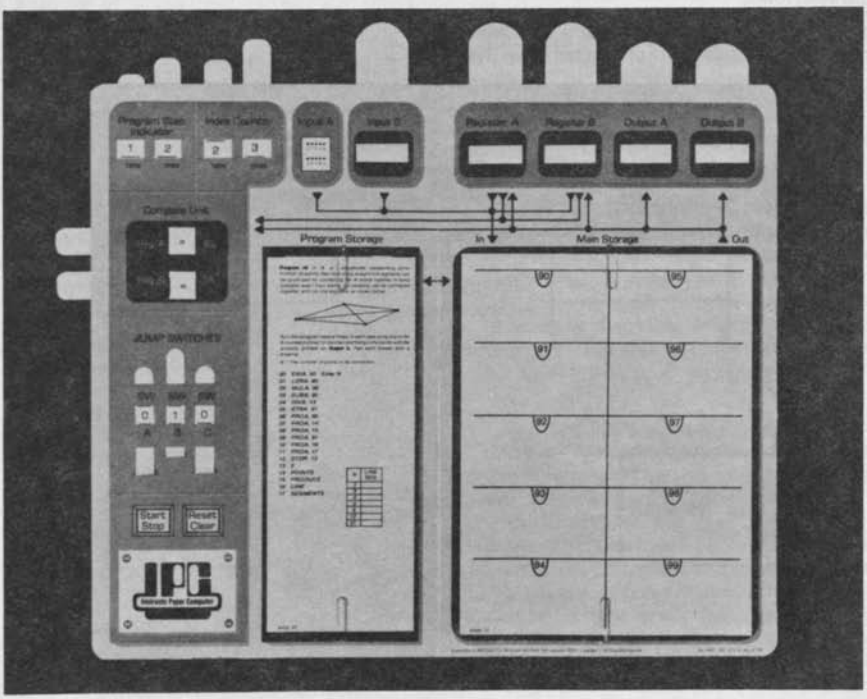
Beginners will need the Mnemonic List to operate the computer. As the operator becomes more familiar with IPC, the Mnemonic List will become less necessary.

Program instructions, using the Mnemonic List, will direct the operator through various exercises which will result in meaningful information being printed on one of the Output strips. For more detailed step by step directions, see pages 7 and 8 in this book.

To assemble IPC follow directions below.

ASSEMBLY DIRECTIONS FOR IPC

1. *Carefully separate the nine slides which are attached to your computer.*
2. *Place the slides in the computer as shown in the diagram below. Bend up the bottoms of the Jump Switch slides.*
3. *Cut five 1½" strips from an 8½" x 11" blank sheet of paper or use the strips provided on page 13. Insert one each in the slots of* **Input B, Register A, Register B, Output A, Output B.** *(See diagram.)*
4. *Attach a* **Program** *(Programs are located on pages 15 to 22) and a* **Main Storage Unit** *sheet (page 11) to the computer by using four paper clips as shown below. (The Program page must be folded in half lengthwise.)*



ABOUT ELECTRONIC COMPUTERS AND IPC

Your computer can do almost anything a real electronic computer can do, but it works using pencil and paper instead of electricity and integrated electronic circuits. The following four pages will tell you how the components and operations of real computers compare to IPC.

Complicated devices, like automobiles, televisions, rockets and computers, are made from many separate units, all working together. In an automobile, for example, the fuel system brings gasoline to the carburetor; the carburetor then produces a mixture of air and gasoline which the motor burns for fuel; the motor turns to produce mechanical power; and the transmission decides how this power is used to turn the wheels of the car. A car also has a suspension system, a cooling system and an electrical system, to name just a few. An automobile is made up of many separate systems; each having its own special job which contributes to the smooth operation of the machine.

When a machine is made up of many separate units, it is said to have a **modular** *design. Most electronic computers have a modular design, as they consist of many separate units, all working together through a complex system of interconnecting wires. Some of these wires are part of the control and timing system which tells each unit exactly when to do its job. Other wires are part of the power system, for they carry electrical energy to the electronic components in the various units. Still other wires are grouped together into a system called* **data busses.**

Data busses *carry alphabetic and numeric information between the units of a computer just like roads carry automobiles between towns.*

Your **Instructo Paper Computer** has a modular design, but only the data busses are shown between the units. The other interconnecting wires are left to your imagination. Figure A is a simple diagram which describes the modular design of IPC.

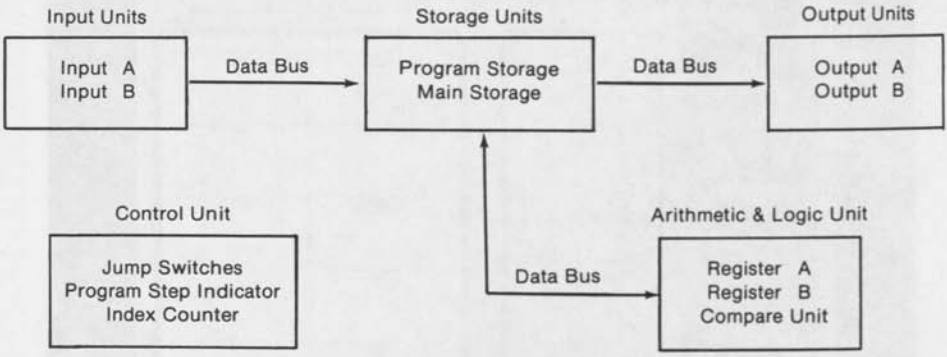
Look at your computer and notice that the Inputs, Outputs, Registers, Compare Unit and Storage Units are connected by a network of lines. These lines are the data busses, and the seven dots on them show where two lines connect. When two lines cross but do not connect, there is no dot at the intersection.

Many data busses are like one way streets, allowing information to flow in only one direction. On your computer these directions are shown by arrows at the ends of the busses. Notice, for instance, that information can only flow into the Main Storage Unit from Input A, Input B, Register A or Register B, as these are the only four units connected to the Input of the Main Storage Unit. Notice, too, that data cannot go from any Input to the Compare Unit because there is no connecting data bus between these units. Data cannot go from an Output to a Register either, because it would flow against the arrow under the Output. The pattern of data flow for your computer is shown in the chart below. Verify the flow pattern by following the data busses and arrows between the units.

Data Flow

From Input A or Input B	→	To Storage Units
From Register A or Register B	→	To Storage Units or Compare Unit
From Storage Units	→	To Register A, Register B, Compare Unit, Output A or Output B

Figure A



Storage Units *The storage units of a large computer are actually a collection of millions of identical storage cells or locations. Each storage location has the ability to store and remember a small amount of alphabetic or numeric information. Every storage location has its own numeric* **address** *by which it can be identified. Information travels to and from the storage units of a computer through the data busses. The entire process is generally governed by the computer program.*

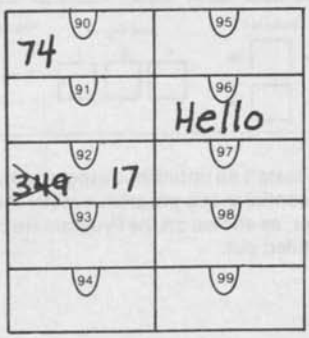
Program Storage Unit Your computer has 100 storage locations, numbered from 00 to 99. The first 90 locations, numbered from 00 to 89, make up what is called the Program Storage Unit. It is located on the left side of your Paper Computer, and it looks like the figure below when a program is in place.

Sample Program

```
00  LDRA, 31
01  STRA, 90
02  ADDA, 42
03  MULA, 37
04  LDRB, 42
05  JJC1, 13
06  DIVB, 90
07  STRB, 92
```

Notice that storage location 00 contains program instruction step LDRA, 31. Storage location 01 contains the instruction STRA, 90. Storage location 02 has the instruction ADDA, 42, and so on.

Main Storage Unit On the right side of your computer there are ten rectangular boxes numbered from 90 to 99. These boxes, or storage locations, are shown below. They make up what is called the Main Storage Unit of IPC. You can store information in any of these ten Main Storage locations by writing the information in the appropriate box after crossing out any old information that might have been there. Notice that Main Storage location 90 contains the number 74, and location 96 contains the word H E L L O. Main Storage location 92 had contained the number 349, but that was crossed out when the number 17 was stored at a later time.



Inputs *Computers are made to solve problems, but they must be given enough information to know exactly what the problem is and how to solve it. An input device provides a way of entering information into the storage units of a computer. It is similar to the way your eyes and ears feed information into your brain. The zero through nine buttons of a small calculator act as its input devices, while large computers rely on special typewriters, magnetic tape and punched card readers, etc. Information may be entered into some computers by talking to them.*

Your computer has two Inputs. Input A works like the zero through nine buttons on a small calculator, while Input B operates like the magnetic tape reader of the larger machines.

Outputs *Once a computer has solved a problem, it must have a way to communicate the answer. It does so by using an output device which provides a means of getting information out of the storage units. The outputs of a calculator are the small lights where numeric answers are displayed. Your voice is an output for your brain. High speed printers and television displays are common output devices on large computers. Many computers can actually talk quite well in several languages.*

Your computer has two Outputs. They are called Output A and Output B. IPC will give you a message and some information by instructing you to write certain words and numbers on the strips of paper showing through the windows of the Outputs. When your computer stops running a program, you can pull up the Output strips and read the message, obtaining the answer to the problem the computer solved.

Program Step Indicator *A computer program is actually a step by step set of instructions which tells the computer exactly what to do. The program is created by a person called a programmer.*

In your computer, the program is stored in the Program Storage Unit, starting at storage location 00 and ending somewhere before storage location 90, depending upon how long the program is. When the computer operates, it follows the program one step at a time doing a specific task at each step. The Program Step Indicator tells exactly what step of the program the computer is doing at any given time.

Registers *During the operation of any computer there are countless arithmetic operations going on inside, and these operations almost always involve the movement of data from one unit to another. Information moving out of the storage units, for instance, can travel over the data busses and be loaded into small storage devices called registers in the arithmetic and logic unit of the machine. Once in the register, the data can be added to, subtracted from, compared to, etc., another number in a specified storage location. When the arithmetic operations are complete, the results are put into the register and sent over the data busses to be stored in the storage units for future use.*

IPC has two registers. They are Register A and Register B, and either of them can be loaded with the information contained in any specified storage location. To load information into a register from a storage location, determine what data is in that storage location and write the same information on the paper strip of the register, pulling up the paper strip to clear any previously written data. To store information from the register into a storage location, first determine what data is in the register, then write the same information in the specified storage location, erasing or crossing out any other data that might have been there.

Compare Unit *In the operation of a computer it is frequently necessary to determine whether or not some number is greater than (>) , equal to (=), or less than (<) some other number.*

The Compare Unit of your computer is designed to do this. When a number in either Register A or Register B is compared to a number in some designated storage location (SS), the result will show on the slides of the Compare Unit. This result can then be used in a decision-making process that can change the number in the Program Step Indicator and alter the sequence in which the computer follows its program.

Jump Switches Three Jump Switches are located on the left side of your computer and are lettered A, B and C. Each of the switches can be set to the number 0 or 1. It is easy to see from the chart below that exactly eight different settings of the three Jump Switches are possible.

Jump Switches

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

To understand the use of the Jump Switches you must remember that a computer simply follows a set of instructions called the computer program, one step at a time, in order. Remember also that the Program Step Indicator keeps track of these instructions and tells which one is being done at any given time. With a special program instruction, however, the computer can test the setting of the Jump Switches and change the number in the Program Step Indicator accordingly.

Certain programs, then, can be controlled to do different things depending upon the setting of the Jump Switches. For example, one program may be used to calculate the area of a triangle if the Jump Switches are set one way, while a different setting of the Jump Switches can cause the same program to calculate the area of a rectangle. In this way, it is possible to make one program do at least eight different things by setting the Jump Switches in different ways.

Index Counter *The ability of a computer to search sequentially through a list of data, or to use a table of numbers in order, is referred to as indexing. Only the more advanced machines have this ability.*

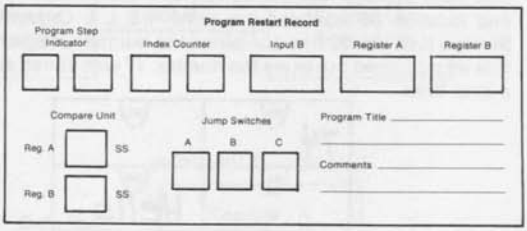
Such a feature is built into your computer in the form of an Index Counter which can be incremented, decremented, and tested for zero by special program instructions. Many repetitive or sequential processes can easily be done using the Index Counter.

Start/Stop Switch *Pushing the Start/Stop Switch of a computer causes it to start following the program instructions located in the Program Storage Unit. The program steps are completed one at a time at the locations shown by the Program Step Indicator.*

Pushing the Start/Stop Switch while a computer is running causes it to complete the program instruction step in progress, and then to stop.

If you want to stop your computer in the middle of a program and continue at a later time, follow the directions below.

1. Complete the program instruction step you are doing.
2. Set the Program Step Indicator to the next program step.
3. Record the necessary information on a Program Restart Record like the one shown below.



To restart an unfinished program, start your computer and continue the program exactly where you stopped earlier, as shown on the Program Restart Record which you filled out.

Reset/Clear Switch *Most computers have a method of resetting the machine and clearing out old data before running a new program.*

When you reset and clear IPC you should do the following five things:

1. Set the Program Step Indicator to 00.
2. Set the Index Counter to 00.
3. Set Registers A and B to 0.
4. Set the Compare Unit slides to = .
5. Clear Output A and Output B.

Operating IPC The computer program should be placed in the Program Storage Unit before your computer is turned on.

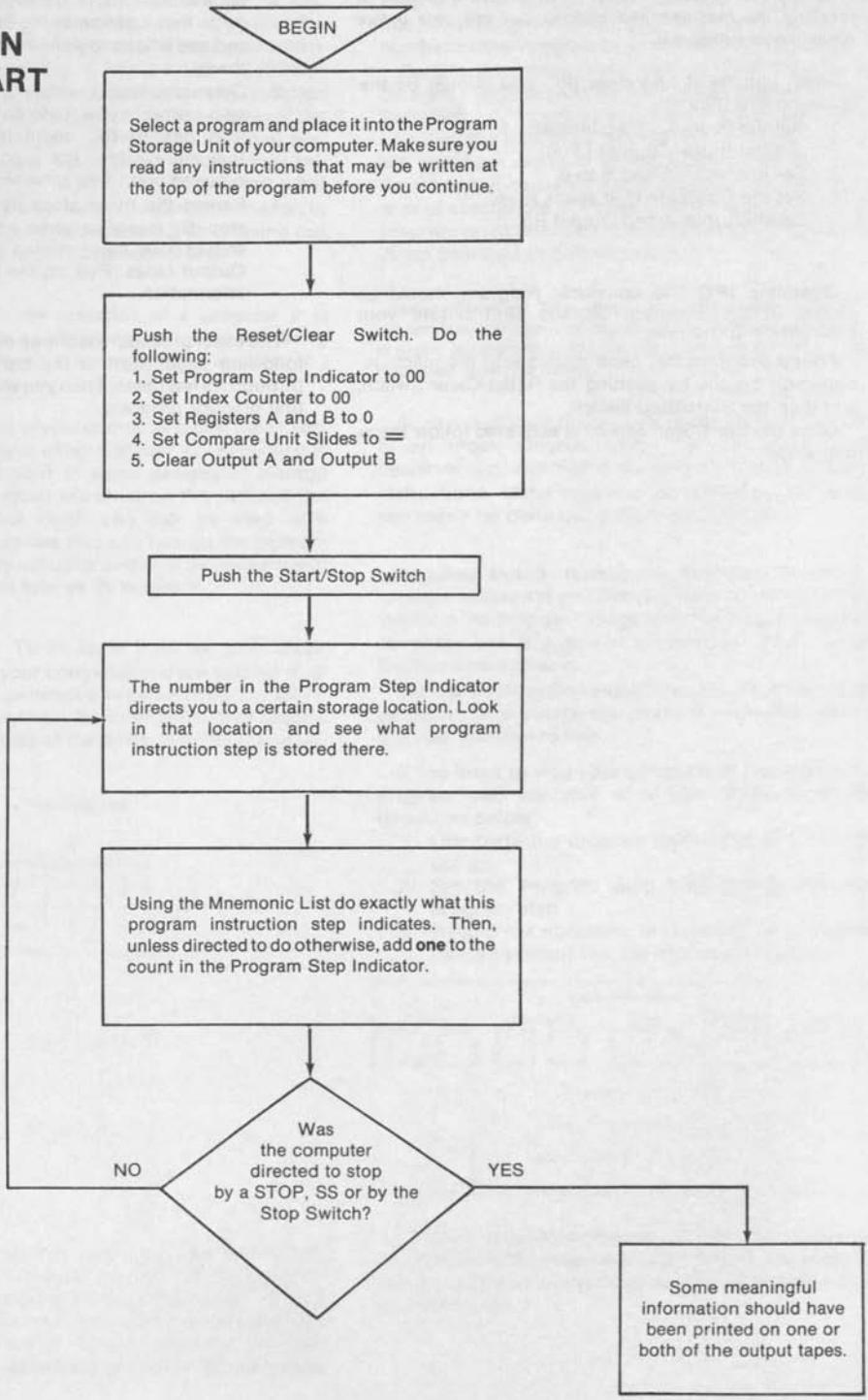
After a program has been loaded into the machine, operation begins by pushing the Reset/Clear Switch, and then the Start/Stop Switch.

Once the Start/Stop Switch is activated follow these four steps:

1. Look at the number in the Program Step Indicator, go to that location in the Program Storage Unit, and see what program instruction step is stored there.
2. Determine exactly what that program instruction step instructs you to do and do it.
3. Add **one** to the count in the Program Step Indicator, unless the program instruction step required you to do otherwise.
4. Repeat the three steps above until directed to stop. By that time some meaningful information should have been printed on one or both of the Output tapes. Pull up the tape(s) and read the information.

The exact procedure is shown on the flow chart on the following page. Start at the top of the chart and go through it a few times. Then you will be ready to run your first practice program.

IPC OPERATION FLOW CHART



RUNNING A PRACTICE PROGRAM — STEP BY STEP

To learn how your computer operates, follow the directions below, checking off each one as you complete it. Before you begin, cut out or reproduce (if multiple copies are desired), the **Mnemonic List** (pages 9 and 10). The **Mnemonic List** should be kept with the computer for easy reference.

____ 1. Cut out or reproduce page 15. Fold it in half and clip Practice Program A into the **Program Storage Unit** of your computer. Read the directions at the top of the program and return here to check off this step before you continue.

____ 2. Before the computer is used it must be reset and cleared of old data. Make sure the **Program Step Indicator** and **Index Counter** are set to 00, both of the **Compare Unit** slides read =, **Registers A** and **B** are set at 0, and **Outputs A** and **B** are clear.

____ 3. Push the Start/Stop Switch to begin the operation of your computer.

____ 4. Remember that when a computer operates it simply follows a set of instructions, one at a time, in some order. This set of instructions is called the computer program, and on your computer it is located in the **Program Storage Unit** starting at location 00, and possibly using the entire storage unit up to location 89. Notice that Practice Program A, now in your computer, occupies only twenty-three storage locations, since it starts at location 00 and ends at location 22.

____ 5. You have started your computer and you should notice that the **Program Step Indicator** contains the number 00. This tells you to look in location 00 of the **Program Storage Unit** where you should find the program instruction step LDRA, 13. Verify this for yourself.

____ 6. Now you must find out exactly what LDRA, 13 means and do what is required to the units of the computer. To do this, look up LDR*, SS on the **Mnemonic List.** The * stands for an **A** or a **B** and the **SS** represents any storage location address from 00 to 99. In LDRA, 13, then, the * stands for an **A,** and the **SS** equals 13. The entire instruction reads, "Load Register A with the contents of storage location 13." Storage location 13, on this program, contains a 4.

____ 7. You should have written a 4 in **Register A.** Since the program instruction step did not require you to alter the reading in the **Program Step Indicator,** simply add one to its count, making the new reading 01.

____ 8. Repeat the procedure. Look at the number in the **Program Step Indicator,** and go to that storage location to see what program instruction step is there. Since the **Program Step Indicator** now reads 01, go to storage location 01 and see what is there.

____ 9. Location 01 contains the program instruction step MULA, 14. Look up MUL*, SS on the **Mnemonic List** and do what is required. Keep in mind that * means either **A** or **B,** and **SS** is any storage location number from 00 to 99. For MULA, 14, the * equals **A** and the **SS** is 14. This instruction reads, "Multiply the number in Register A by the number in storage location 14." There is a 6 in storage location 14.

____ 10. Since the 4 that had been in **Register A** was multiplied by the 6 that is in storage location 14, the product 24 should now be in **Register A.** The program instruction step did not tell you to change the number in the **Program Step Indicator,** so add one to its count, making the new reading 02.

____ 11. The **Program Step Indicator** now reads 02. Look at storage location 02 where you should find the program instruction step SUBA, 15.

____ 12. Look up SUB*, SS and figure out what you must do. If you are correct, there should be a zero in **Register A** when this program instruction step is finished.

____ 13. Like the other program instruction steps you've done so far, SUB*, SS does nothing to the count in the **Program Step Indicator,** so just increase the count by one before you continue.

____ 14. The **Program Step Indicator** should now read 03. Look in storage location 03 where you should find JAZE, 09.

____ 15. Every computer instruction that starts with the letter J is called a "jump instruction." Each jump instruction has the ability to alter the number in the **Program Step Indicator,** causing the computer to jump over a group of program instruction steps. Since you are now working with the instruction in location 03, look up J*ZE, SS and follow the requirements given before returning to check off this direction.

____ 16. Since there was a zero in **Register A** when you did JAZE, 09, the **Program Step Indicator** should now read 09. Go to location 09 and verify that PROA, 17 is stored there.

____ 17. Look up PRO*, SS. Do what is required, and finish this program instruction step by adding one to the count in the **Program Step Indicator.**

____ 18. The **Program Step Indicator** now reads 10, so look in storage location 10 where you should find PROA, 18. Do what is required and finish the program instruction step by adding one to the count in the **Program Step Indicator.**

____ 19. The **Program Step Indicator** now reads 11, where you should find another PRO*, SS type instruction. Do what is required and finish the program instruction step by adding one to the count in the **Program Step Indicator.**

____ 20. The **Program Step Indicator** now reads 12. Look in storage location 12, determine what is to be done, and do it.

(over)

____ 21. The **Program Step Indicator** should still read 12, but the computer was directed to stop operating. There should be meaningful information on the output tape. Pull up the tape. The message on **Output A** should read as follows:

HELLO
I-M-A
COMPUTER

____ 22. In the program you just ran, 4 and 6 were multiplied together and 24 was subtracted from the product, leaving a zero in **Register A.** The program instruction step, JAZE, 09 (jump to 09 if there is a zero in **Register A**), caused the program to jump to 09 because there was a zero in **Register A,** and the message "Hello I-M-A Computer" printed on **Output A.** The portion of the program that was jumped over was never done.

Run Practice Program A again. This time, make an error in the arithmetic. Notice the difference in the output message.

____ 23. Much of the power of a computer lies in its ability to make decisions. In IPC these decisions are made by the jump instructions. IPC is able to check the numbers in its registers and make a decision. To indicate a computational error in Practice Program A, the error message, "you goofed, try again" was printed.

____ 24. Load Practice Program B into the **Program Storage Unit** of your computer. You should find the program on the same page that is now in your computer.

____ 25. Run Practice Program B. If you have trouble doing Program B, check the Flow Chart on page 6, or the TO RUN IPC directions on this page.

TO RUN IPC

1. Select and load a program into the **Program Storage Unit.**
2. Reset and clear the computer.
3. Start the computer.
4. Look in the storage location indicated by the **Program Step Indicator** to see what program instruction step is stored there.
5. Determine what the program instruction step tells you to do and do it.
6. Add one to the count in the **Program Step Indicator** unless told to do otherwise.
7. Repeat the steps above until directed to stop.

MNEMONIC LIST
(with code explanations)

mnemonic ni-män-ik adj. Intended to assist memory

The Program Directions for IPC are written in a mnemonic code to make the information easy to remember.

SS is a placeholder for any storage location number from 00 to 99.

Example: Program reads — Mnemonic Code reads

00 ENIA, 90 — 00 ENIA, SS

When **IN** is used in place of the **SS** number it means that the **SS** number should be taken from the Index Counter.
e.g. 00 JUMP, IN means jump to the storage location shown in the Index Counter.

* is a placeholder for A or B or C or I as designated in Program Directions.

Example: Program reads — Mnemonic Code reads

00 ENIA, 90 — 00 ENI*, SS

is a placeholder for 0 or 1 on the Jump Switches as designated in Program Directions.

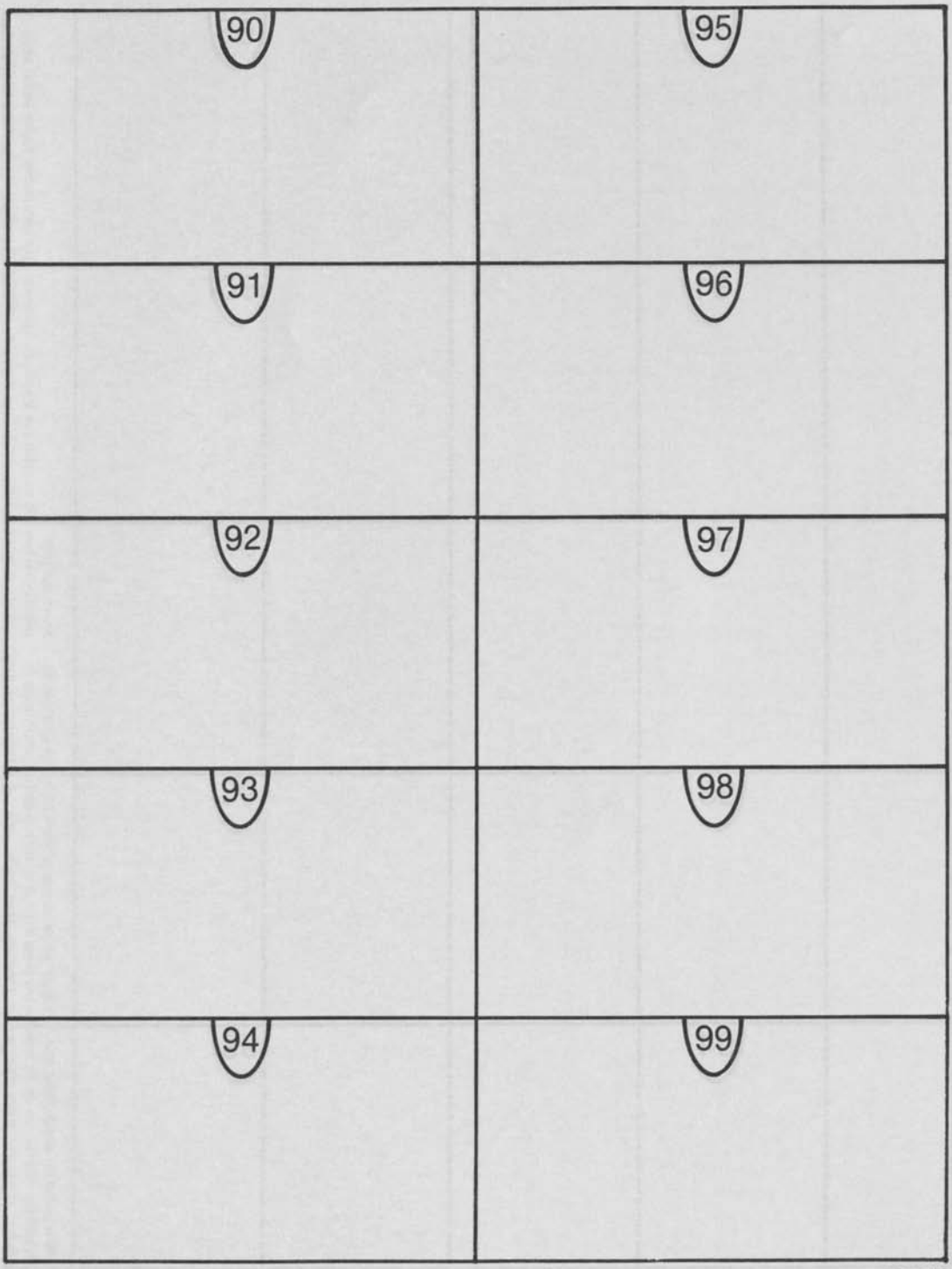
Example: Program reads — Mnemonic Code reads

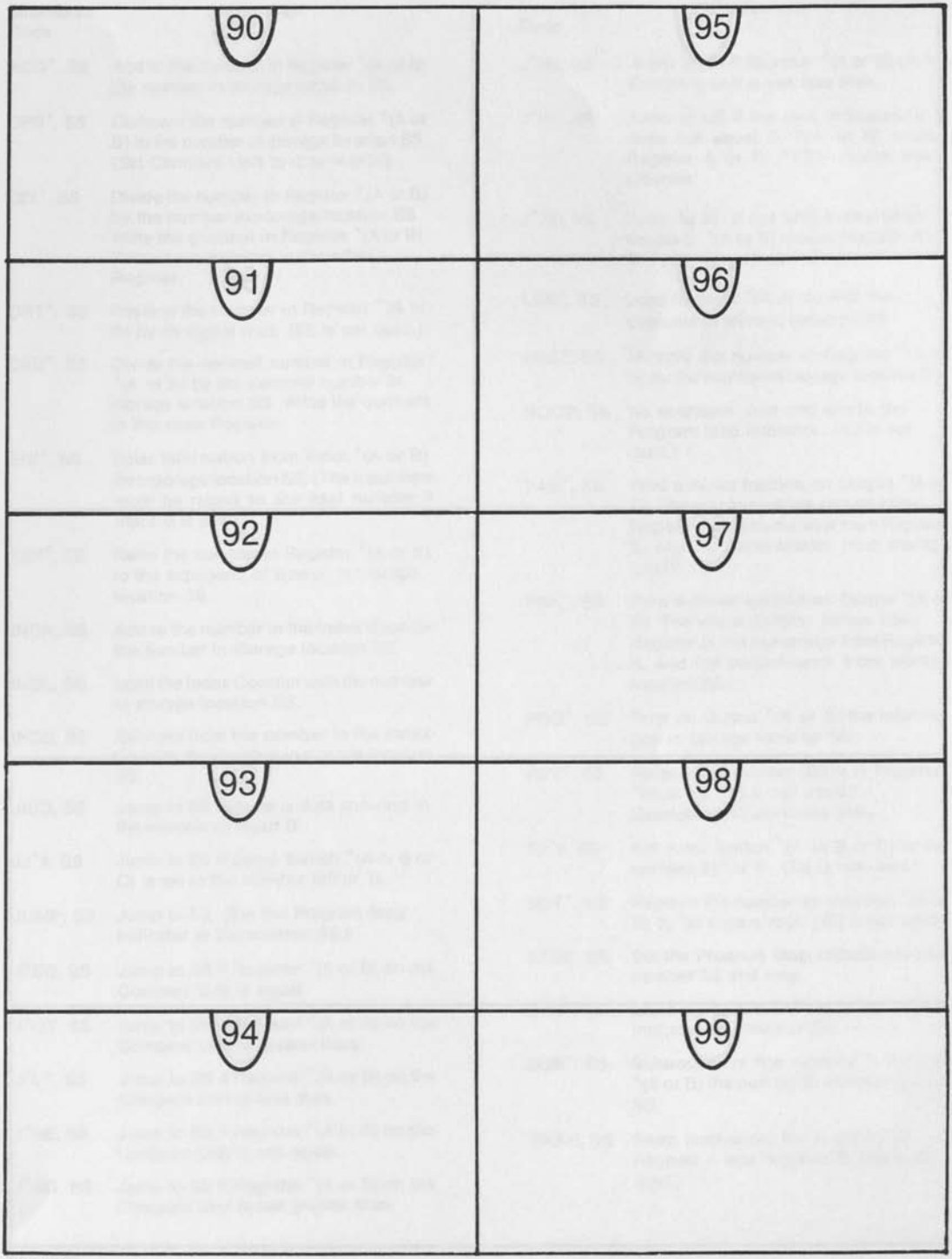
00 JJB1, 18 — 00 JJ*#, SS

When a program requires you to do an **Enter, Load, Print** or **Store** instruction, the information should be written by hand into the designated unit.

(over)

Mnemonic Code	Explanation
ADD*, SS	Add to the number in Register *(A or B) the number in storage location SS.
CPR*, SS	Compare the number in Register *(A or B) to the number in storage location SS. (Set Compare Unit to < or = or >.)
DIV*, SS	Divide the number in Register *(A or B) by the number in storage location SS. Write the quotient in Register *(A or B). Write the remainder in the other Register.
DRT*, SS	Replace the number in Register *(A or B) by its digital root. (SS is not used.)
DVD*, SS	Divide the decimal number in Register *(A or B) by the decimal number in storage location SS. Write the quotient in the same Register.
ENI*, SS	Enter information from Input *(A or B) into storage location SS. (The Input tape must be raised to the next number if Input B is used.)
EXP*, SS	Raise the number in Register *(A or B) to the exponent, or power, in storage location SS.
INDA, SS	Add to the number in the Index Counter the number in storage location SS.
INDL, SS	Load the Index Counter with the number in storage location SS.
INDS, SS	Subtract from the number in the Index Counter the number in storage location SS.
JIBD, SS	Jump to SS if there is data showing in the window of Input B.
JJ*#, SS	Jump to SS if Jump Switch *(A or B or C) is set to the number #(0 or 1).
JUMP, SS	Jump to SS. (Set the Program Step Indicator to the number SS.)
J*EQ, SS	Jump to SS if Register *(A or B) on the Compare Unit is **equal.**
J*GT, SS	Jump to SS if Register *(A or B) on the Compare Unit is **greater than.**
J*LT, SS	Jump to SS if Register *(A or B) on the Compare Unit is **less than.**
J*NE, SS	Jump to SS if Register *(A or B) on the Compare Unit is **not equal.**
J*NG, SS	Jump to SS if Register *(A or B) on the Compare Unit is **not greater than.**
J*NL, SS	Jump to SS if Register *(A or B) on the Compare Unit is **not less than.**
J*NZ, SS	Jump to SS if the unit indicated by * does **not equal** 0. *(A or B) means Register A or B; *(I) means Index Counter.
J*ZE, SS	Jump to SS if the unit indicated by * **equals** 0. *(A or B) means Register A or B; *(I) means Index Counter.
LDR*, SS	Load Register *(A or B) with the contents of storage location SS.
MUL*, SS	Multiply the number in Register *(A or B) by the number in storage location SS.
NOOP, SS	No operation. Just add one to the Program Step Indicator. (SS is not used.)
PAB*, SS	Print a mixed fraction on Output *(A or B). The whole number comes from Register A, the numerator from Register B, and the denominator from storage location SS.
PBA*, SS	Print a mixed fraction on Output *(A or B). The whole number comes from Register B, the numerator from Register A, and the denominator from storage location SS.
PRO*, SS	Print on Output *(A or B) the information in storage location SS.
REV*, SS	Reverse the number digits in Register *(A or B). (SS is not used.) Example: (543 becomes 345.)
SJ*#, SS	Set Jump Switch *(A or B or C) to the number #(0 or 1). (SS is not used.)
SQT*, SS	Replace the number in Register *(A or B) by its square root. (SS is not used.)
STOP, SS	Set the Program Step Indicator to the number SS and stop.
STR*, SS	Store the contents of Register *(A or B) into storage location SS.
SUB*, SS	Subtract from the number in Register *(A or B) the number in storage location SS.
SWAP, SS	Swap (exchange) the numbers in Register A and Register B. (SS is not used.)


90
95
91
96
92
97
93
98
94
99


90
95
91
96
92
97
93
98
94
99

INPUT, OUTPUT AND REGISTER TAPES—This page may be reproduced for classroom use, or the strips may be cut from a blank sheet of paper. Round ends of paper strips for easier insertion into slots. Write on the tapes, as you are directed by the computer program, through the Input, Output and Register windows. Pull up the tapes from the top to clear the windows. Replace with new paper strips when both sides are full.

Practice Program A

Practice Program A gives you an opportunity to see just how easy it is to operate your paper computer. Fold this page down the middle so only Practice Program A shows, and place it in the Program Storage Unit of your computer. Then, go back to where you were on page 7.

Storage Location (SS)

```
00  LDRA, 13
01  MULA, 14
02  SUBA, 15
03  JAZE, 09
04  PROA, 20
05  PROA, 16
06  PROA, 21
07  PROA, 19
08  STOP, 08
09  PROA, 17
10  PROA, 18
11  PROA, 22
12  STOP, 12
13  4
14  6
15  24
16  GOOFED
17  HELLO
18  I-M-A
19  AGAIN
20  YOU
21  TRY
22  COMPUTER
```

Practice Program B

Write down a secret number from 000 to 999. Then form a new number by mixing up your secret number. Follow these six steps. The number 726 will be used as an example to show how each of the steps below is used.

1. Double the hundreds digit. (726) 2 x 7 = 14
2. Add three to the results. (726) 14 + 3 = 17
3. Multiply that answer by five. (726) 5 × 17 = 85
4. Add on the tens digit of your original secret number. (726) 85 + 2 = 87
5. Multiply this by ten. (726) 10 × 87 = 870
6. Finish forming this new number by adding on the ones digit of your original secret number.
   (726) 870 + 6 = 876
   The new number is 876.

Pick your own secret number and mix it up according to the six steps above. Then, by running the program below and entering only your new number and your age, the paper computer will be able to tell you the original secret number that you named.

Storage Location (SS)

```
00  ENIA, 93   Enter new number.
01  LDRB, 93
02  ENIA, 97   Enter your age.
03  SUBB, 97
04  LDRA, 13
05  SUBA, 97
06  STRA, 97
07  SUBB, 97
08  STRB, 93
09  PROB, 93
10  PROB, 14
11  PROB, 15
12  STOP, 12
13  150
14  IS YOUR
15  NUMBER
```

Program #1 This is a simple practice exercise. If done correctly, **Output A** should read as follows:

TESTING
1
2
3
THE END

If a mistake is made in the arithmetic or the comparing, then the program will jump to an error routine and print the word ERROR on **Output B.**

```
00 PROA, 21
01 LDRA, 20
02 STRA, 93
03 PROA, 93
04 LDRB, 20
05 ADDB, 20
06 STRB, 91
07 PROA, 91
08 STRA, 94
09 ADDB, 94
10 STRB, 94
11 PROA, 94
12 CPRA, 91
13 JALT, 16
14 PROB, 22
15 STOP, 15
16 CPRB, 91
17 JBGT, 19
18 JUMP, 14
19 JUMP, 24
20 1
21 TESTING
22 ERROR
23 THE END
24 PROA, 23
25 STOP, 25
```

Program #2 This program will print several important dates from the history of the United States. When finished, check your dates.

```
00 LDRA, 38
01 MULA, 39
02 STRA, 90
03 PROA, 36
04 PROA, 90
05 ADDA, 40
06 STRA, 91
07 PROA, 33
08 PROA, 91
09 ADDA, 41
10 STRA, 92
11 PROA, 34
12 PROA, 92
13 LDRB, 42
14 MULB, 39
15 STRB, 94
16 ADDB, 92
17 STRB, 97
18 PROA, 37
19 PROA, 97
20 ADDB, 94
21 STRB, 99
22 PROA, 32
23 PROA, 99
24 LDRA, 38
25 MULA, 42
26 SUBA, 90
27 SUBA, 39
28 STRA, 97
29 PROA, 35
30 PROA, 97
31 STOP, 31
32 GOLD RUSH
33 PHILADELPHIA
34 INDEPENDENCE
35 CIVIL WAR
36 COLUMBUS
37 BRITISH WAR
38 373
39 4
40 190
41 94
42 9
```

Program #3 This is a program testing your ability to add, subtract, multiply and divide. If you make no errors, the word CORRECT will print on **Output A.** If you make an error, such as forgetting to indicate the remainder after dividing, the computer will print the word ERROR on **Output A.** In that case, reset the computer and run the program again.

```
00  LDRA, 19
01  ADDA, 20
02  LDRB, 21
03  STRA, 98
04  SUBB, 22
05  CPRB, 98
06  JBNE, 13
07  MULB, 19
08  DIVB, 20
09  STRB, 91
10  ADDA, 91
11  CPRA, 23
12  JAEQ, 15
13  PROA, 26
14  STOP, 14
15  PROA, 25
16  PROA, 24
17  PROA, 27
18  STOP, 18
19  47
20  13
21  183
22  123
23  228
24  U-R-A
25  CORRECT
26  ERROR
27  GENIUS
```

Program #4 Pick one of the faces below and answer the following questions by setting a **Jump Switch** to 1 for YES, or 0 for NO.

Jump Switch A: Does it have a beard?
Jump Switch B: Does it have teeth?
Jump Switch C: Does it have hair?

Run the program and the computer will tell you which face you chose.

```
00  JJBO, 07
01  JJC1, 14
02  JJAO, 05
03  PROA, 11
04  JUMP, 18
05  PROA, 12
06  JJB1, 18
07  JJCO, 22
08  JJAO, 17
09  PROA, 32
10  JUMP, 18
11  WEIRDO
12  IKE
13  DROOP
14  JJA1, 29
15  PROA, 13
16  JUMP, 18
17  PROA, 25
18  PROA, 20
19  STOP, 19
20  IS THE ONE
21  GRORF
22  JJAO, 27
23  PROA, 26
24  JUMP, 18
25  FLIPI
26  GRUNGE
27  PROA, 21
28  JUMP, 18
29  PROA, 31
30  JJC1, 18
31  RALPH
32  BONE
```

Program #5 This is a program showing how a person is able to enter information into the storage units of a computer. It works exactly like the 0 to 9 buttons on a small calculator. Program step 00 asks you to enter any number, N, from **Input A,** and to place that number into storage position 90.

The computer will then calculate the first three nonzero multiples of the number N and print them on **Output B.** If you enter a 5, for instance, **Output B** will print the numbers 5, 10, 15.

```
00  ENIA, 90   Enter N
01  ADDA, 90
02  STRA, 91
03  PROB, 91
04  ADDB, 08
05  CPRB, 09
06  JBLT, 01
07  STOP, 07
08  1
09  3
```

Program #6 Computers often seem able to read your mind. For this program write down any number from 0 to 109. Then divide your number first by 10, and then by 11, to determine what the remainders are in each case. By using only these remainders (let's call them R10 and R11), the computer can tell you the number you thought of.

R 10 = The remainder when your number is divided by 10.

R 11 = The remainder when your number is divided by 11.

```
00  ENIA, 90   Enter R10
01  ENIA, 91   Enter R11
02  LDRA, 90
03  MULA, 14
04  LDRB, 91
05  MULB, 15
06  STRB, 90
07  ADDA, 90
08  DIVA, 16
09  STRB, 99
10  PROB, 17
11  PROB, 18
12  PROB, 99
13  STOP, 13
14  11
15  100
16  110
17  YOUR
18  NUMBER IS
```

Program #7 This program will show you one of the uses for **Inputs A** and **B** by calculating the average of a set of numbers entered one at a time from **Input B.** To do this, first list the numbers to be averaged on a 1½ by 11 inch strip of paper. Then install the strip through the slots of **Input B** so that the first number of the list shows through the window. When you run the program, step 00 will require you to enter the number of numbers listed on the **Input B** strip. Afterwards, the average will print on **Output A.** Use this program to average the number sets below.

N = The number of numbers listed on the **Input B** strip.

```
00  ENIA, 97   Enter N
01  ENIB, 90
02  ADDA, 90
03  ADDB, 13
04  CPRB, 97
05  JBLT, 01
06  DIVA, 97
07  PROA, 12
08  CPRB, 14
09  JBEQ, 15
10  PABA, 97
11  STOP, 11
12  AVERAGE IS
13  1
14  0
15  STRA, 99
16  PROA, 99
17  STOP, 17
```

Determine the average of each number set shown below by writing each list on an **Input B** strip and running the program.

(8, 6, 33, 5) _____

(18, 31, 40, 77, 64, 83) _____

(31, 92, 47, 37, 15, 83, 26) _____

(142, 301, 195, 247, 290) _____

Program #8 The commutative and the associative laws of multiplication can be used to show that a group of numbers can be multiplied together in any order to produce the same answer. Three different numbers, for instance, can be arranged and multiplied in six ways as shown:

1 x 2 x 3	1 x 3 x 2
2 x 1 x 3	2 x 3 x 1
3 x 1 x 2	3 x 2 x 1

This program is designed to determine how many possible ways there are to arrange and multiply a set of (N) different numbers together. Run the program several times, in each case using an N-number from the chart shown below. Check your answers on scrap paper and fill in the blank boxes of the chart.

N = The number of numbers in the set.

```
00  ENIA, 92   Enter N
01  LDRA, 17
02  STRA, 90
03  ADDA, 17
04  LDRB, 18
05  MULA, 90
06  CPRB, 92
07  JBEQ, 11
08  ADDB, 17
09  STRB, 90
10  JUMP, 05
11  STRA, 91
12  PROA, 91
13  PROA, 19
14  PROA, 92
15  PROA, 20
16  STOP, 16
17  1
18  2
19  WAYS TO X
20  NUMBERS
```

N	2	3	4	5
WAYS TO X				

Program #9 If N is a placeholder representing some number of points, then how many straight line segments can be produced by connecting the N points together in every possible way? Four points, for instance, can be connected together with six line segments as shown below.

Run this program several times, in each case using one of the N-numbers shown in the chart and filling in the blanks with the answers printed on **Output A.** Test each answer with a drawing.

N = The number of points to be connected.

```
00  ENIA, 90   Enter N
01  LDRA, 90
02  MULA, 90
03  SUBA, 90
04  DIVA, 13
05  STRA, 91
06  PROA, 90
07  PROA, 14
08  PROA, 15
09  PROA, 91
10  PROA, 16
11  PROA, 17
12  STOP, 12
13  2
14  POINTS
15  PRODUCE
16  LINE
17  SEGMENTS
```

N	LINE SEG.
2	
3	
4	
5	
6	
12	
37	

Program #10 If N is a symbol used to represent some counting number, and if the number is entered into storage position 90, then this program will calculate the sum of all the numbers from 1 to N.

N = Any counting number

Sum = 1 + 2 + 3 + 4 ... + N

For example, if N = 7, then:

Sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28

Run this program several times, in each case using one of the N-numbers shown in the chart and filling in the blanks with the answers printed on **Output A.** Check your sums with a calculator.

```
00  ENIA, 90   Enter N
01  LDRA, 90
02  MULA, 90
03  ADDA, 90
04  DIVA, 13
05  STRA, 91
06  PROA, 11
07  PROA, 90
08  PROA, 12
09  PROA, 91
10  STOP, 10
11  SUM OF 1st
12  NUMBERS IS
13  2
```

N	SUM
4	
8	
12	
23	
40	
73	
100	

Program #11 Program 10 calculated the sum of all the numbers from one up to any other number N. This program is more powerful, for if N1 and N2 are used to represent any two counting numbers, then the sum of all the numbers from N1 to N2 can be found. For example, if N1 = 7 and N2 = 13, then:

Sum = 7 + 8 + 9 + 10 + 11 + 12 + 13 = 70

Run this program several times to answer the problems shown below.

```
00  ENIA, 95   Enter N1
01  ENIA, 96   Enter N2
02  LDRA, 95
03  CPRA, 96
04  JAGT, 25
05  MULA, 95
06  STRA, 97
07  LDRA, 96
08  MULA, 96
09  SUBA, 97
10  ADDA, 95
11  ADDA, 96
12  DIVA, 24
13  STRA, 97
14  PROA, 21
15  PROA, 95
16  PROA, 22
17  PROA, 96
18  PROA, 23
19  PROA, 97
20  STOP, 20
21  SUM FROM
22  TO
23  EQUALS
24  2
25  LDRB, 96
26  STRA, 96
27  STRB, 95
28  LDRA, 95
29  JUMP, 05
```

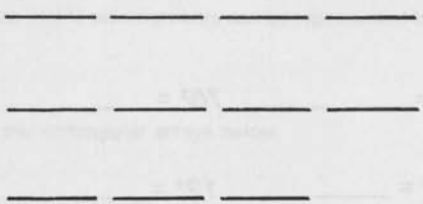
```
    11
    12
    13
    14
    15
  +17
    26
    27
    28
     .
     .
     .
  +41
    63
    64
    65
     .
     .
     .
  +99
```

42 + 41 + 40 + 39 . . . + 30 = ____

Program #12 Mathematicians often study special lists of numbers, searching for patterns which might be useful in explaining the many phenomena in the world around us. The famous **Fibonacci** number sequence, for instance, closely describes how some trees add on new branches, how pine cones grow, and how certain characteristics are passed along from parents to their children. This program will print a list of all the Fibonacci numbers under one hundred. After running the program, transfer the numbers from **Output A** to the spaces provided below. There are hundreds of patterns in these numbers. Can you discover a few for yourself?

```
00  PROA, 16
01  PROA, 17
02  LDRA, 14
03  STRA, 95
04  STRA, 96
05  PROA, 95
06  LDRB, 96
07  CPRB, 15
08  JBGT, 13
09  ADDA, 95
10  STRA, 96
11  STRB, 95
12  JUMP, 05
13  STOP, 13
14  1
15  100
16  FIBONACCI
17  NUMBERS
```

Write the first eleven Fibonacci numbers below.



Program #13 The **Index Counter** of your computer can be incremented, decremented, or loaded with any number, all under control of the program. The **Index Counter** can also be tested to see if it is zero or not; this is done with a special jump instruction. The **Index Counter** is used in this program to compute the value of N^e.

N = Any number
e = Any whole number exponent of **N**
Example: If **N** = 5 and **e** = 3, then:
$5^3 = 5 \times 5 \times 5 = 125$

Run this program several times, in each case entering the values of **N** and **e** from one of the examples shown below. Fill in the blanks with the answers printed on **Output A.** Check your answers.

```
00  ENIA, 96   Enter N
01  ENIA, 97   Enter e
02  INDL, 97
03  LDRA, 15
04  JIZE, 08
05  INDS, 15
06  MULA, 96
07  JUMP, 04
08  STRA, 90
09  PROA, 96
10  PROA, 16
11  PROA, 97
12  PROA, 17
13  PROA, 90
14  STOP, 14
15  1
16  EXPONENT
17  EQUALS
```

53^2 = _____ 76^3 = _____

491^0 = _____ 12^4 = _____

Program #14 Notice how the **Index Counter** is used in this program to determine if a number is prime. The **Index Counter** will allow the number to be divided by certain prime numbers in storage, one at a time, in order. By doing so this program can test any number less than 1369 for being prime or composite. Use your computer to test each N-number in the chart and tell if it is prime or composite by putting a **P** or a **C** in each of the blank spaces. Then check your answers with a prime number list.

N = Any number less than 1369.

```
00  ENIA, 90   Enter N
01  INDL, 33
02  LDRA, 90
03  DIVA, IN
04  CPRB, 19
05  JBEQ, 14
06  INDA, 20
07  LDRA, IN
08  MULA, IN
09  CPRA, 90
10  JANG, 02
11  PROA, 90
12  PROA, 18
13  STOP, 13
14  PROA, 90
15  PROA, 17
16  STOP, 16
17  IS COMPOSITE
18  IS PRIME
19  0
20  1
21  2
22  3
23  5
24  7
25  11
26  13
27  17
28  19
29  23
30  29
31  31
32  37
33  21
```

N	P/C
11	
18	
29	
39	
91	
201	
211	
319	

How to Write a Computer Program for IPC

Creating a computer program can be a challenging and satisfying experience, and it is hoped that you will share these feelings as you follow the development of the two sample computer programs below.

Super Simple Sample Program Number One

Step #1 Describe the problem you wish to program in terms that you understand.

Example: Given any whole number N, a program will be created to add together the first N even numbers. For example, N=3 implies that the sum of the first three even numbers is desired. Since 2 is the first even number, 4 is the second, and 6 is the third, the required sum is 12: (2 + 4 + 6 = 12).

Step #2 Describe the problem in mathematical terms. This usually involves finding a formula or an algorithm that can become a step by step procedure leading to the solution of the problem. If you already have a formula and simply want to write a program for it, go directly to step #3. If you would like to follow the derivation of a formula, continue below.

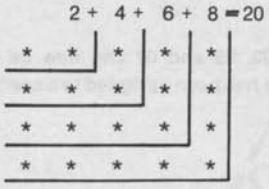
Example continued: The first even number is 2. The sum of the first two even numbers is 6: (2+4 = 6). The sum of the first three even numbers of 12: (2 + 4 + 6 = 12), and so on. Shown in chart form, some interesting patterns appear which lead to the desired formula.

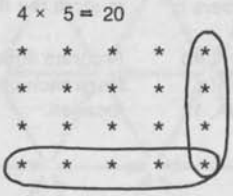
N	Nth even number	Sum of the first N even numbers	An easy pattern for the sums
1	2	2	2 = 1 x 2
2	4	2 + 4 = 6	6 = 2 x 3
3	6	2 + 4 + 6 = 12	12 = 3 x 4
4	8	2 + 4 + 6 + 8 = 20	20 = 4 x 5
5	10	2 + 4 + 6 + 8 + 10 = 30	30 = 5 x 6
N	2N	2 + 4 + 6 ... + 2 N = ?	Sum = N (N + 1)

Mathematically, the patterns in the chart show that the sum of the first N even numbers can be represented by the following formula:

Sum = N (N + 1)

The validity of the formula is demonstrated by the partitions of the rectangular arrays below.





Step #3 Assemble the step by step algorithm, or the formula, into a set of mnemonic program instruction steps which precisely describes the operations to be done. When completed, this list will become the computer program.

Example continued: In this example, a list of mnemonic program instruction steps must be created to describe the formula

Sum of the first N even numbers = N (N + 1).

A method to enter the number N, a way to print the answer and a way to stop the computer when the job is completed must also be included.

00	ENIA, 95	Computer programs always start at storage location 00. In this case, the first step will allow you to enter the number N into storage location 95.
01	LDRA, 95	This loads the number N into Register A.
02	ADDA, ??	The quantity (N + 1) will be formed by adding 1 to the number N already in Register A. The 1 will be stored somewhere in the Program Storage Unit, but the exact location will not be determined until the program length is known. Until that time, two question marks will be used to represent the unknown storage location.
03	MULA, 95	Multiply the contents of Register A by the number N. This completes the formula N (N + 1).
04	STRA, 99	Store the quantity N (N + 1) so it can be printed on an output.
05	PROA, ??	Program steps 05 to 08 will print a message like the sample at the right. The ?? of steps 05 and 07 will locate the words "Sum of first" and "even numbers is", but the exact locations will be unknown until the length of the program is determined. The "3" in the message is printed by step 06, and is really the number N, which was entered into storage location 95 at the beginning. The "12" in the message is the answer that was stored in location 99 by program instruction step 04.
06	PROA, 95	
07	PROA, ??	
08	PROA, 99	
09	STOP, 09	Remember that the computer will continue to run unless told to stop.
10	1	Three data entries were used for this program, and each of them must be assigned a place in the Program Storage Unit. Since the working part of this program ends at 09, locations 10, 11 and 12 can be used to store the three entries.
11	Sum of first	
12	even numbers is	
02	ADDA, 10	Program instruction steps 02, 05 and 07 can now be completed, since each of the data entries has been assigned to a specific storage location.
05	PROA, 11	
07	PROA, 12	

Sum of first
3
even numbers is
12

Step #4 Write the complete program and some simple directions on its use.

If N is a whole number, then this program will calculate the sum of the first N even numbers.
Example: N = 5 will determine the sum of the first five even numbers. (2 + 4 + 6 + 8 + 10 = 30)

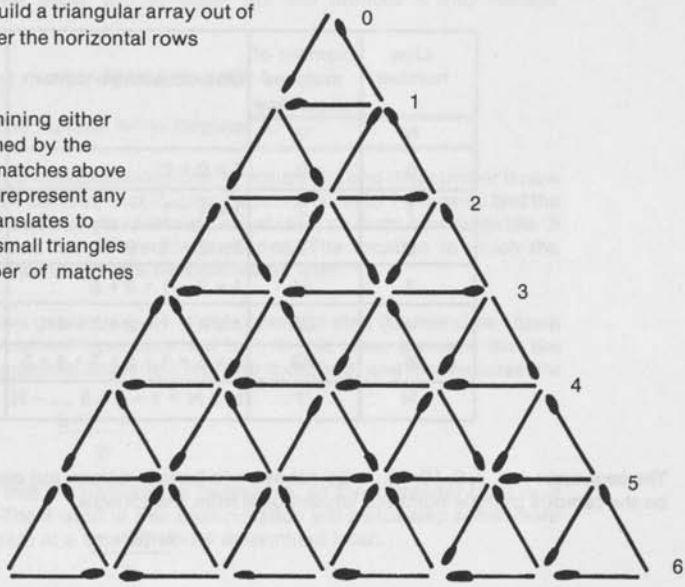
```
00  ENIA, 95   (Enter N)
01  LDRA, 95
02  ADDA, 10
03  MULA, 95
04  STRA, 99
05  PROA, 11
06  PROA, 95
07  PROA, 12
08  PROA, 99
09  STOP, 09
10  1
11  Sum of first
12  even numbers is
```

Sample Program Number Two

The progression of the following program, also shown step by step, will show you how to develop and write a more involved computer program.

Step #1 Describe the problem you wish to program in terms that you understand.

Example: Suppose one were to build a triangular array out of matchsticks and number the horizontal rows as shown.



Let the problem be that of determining either the number of little triangles formed by the matches, or the actual number of matches above any given line. If N is allowed to represent any line number, then the problem translates to determining either the number of small triangles above the Nth line, or the number of matches above the Nth line.

Step #2 Describe the problem in mathematical terms. This usually amounts to finding a formula or an algorithm that can become a step by step procedure leading to the solution of the problem. If you already have a formula and simply want to write a program for it, then go directly to step #3. If you would like to follow the derivation of a formula, continue below.

Example continued: Consider the number of small triangles the matches form above the Nth line of the array. There are none above line 0, one above line 1, four above line 2, etc. Often, when relationships are studied in chart form, patterns appear which are otherwise obscured.

Line number	Number of triangles above line	One observable pattern	Another pattern
N	T		
1	1	1	1 x 1
2	4	1 + 3	2 x 2
3	9	1 + 3 + 5	3 x 3
4	16	1 + 3 + 5 + 7	4 x 4
5	25	1 + 3 + 5 + 7 + 9	5 x 5
N	?	1 + 3 + 5 + 7 ... + (2N-1)	N x N

Mathematically, the number of small triangles (T) above line N equals N^2.

$$T = N \times N \text{ or } T = N^2$$

The number of matches above the Nth line can be determined in much the same way, although it is a bit more involved. Notice that there are no matches above line 0, two above line 1, seven above line 2, and so on. Again, certain patterns appear as shown in the following chart.

Line number	Number of matches above line	One observable pattern	Another pattern
N	M		
0	0	0 x 0 + 0	0 x 0 + 0
1	2	1 x 1 + 1	1 x 1 + 1
2	7	2 x 2 + 1 + 2	2 x 2 + 3
3	15	3 x 3 + 1 + 2 + 3	3 x 3 + 6
4	26	4 x 4 + 1 + 2 + 3 + 4	4 x 4 + 10
5	40	5 x 5 + 1 + 2 + 3 + 4 + 5	5 x 5 + 15
N	?	N x N + 1 + 2 + 3 ... + N	N x N + ?

The sequence 0, 1, 3, 6, 10, 15... may not seem to have a pattern, but close examination will show these numbers to be the famous triangle numbers which come from the formula

$$\frac{N\,(N+1)}{2}$$

This formula will replace the question mark in the lower right hand section of the second chart on the previous page. With this substitution, the chart shows that

$$M = (N \times N) + \frac{N\,(N+1)}{2} .$$

Or, $M = N^2 + \frac{N^2 + N}{2} = \frac{2N^2}{2} + \frac{N^2 + N}{2} = \frac{2N^2 + N^2 + N}{2}$.

Finally it becomes clear that the number of matches, M, above the Nth line is represented by

$$M = \frac{3N^2 + N}{2} .$$

Step #3 Assemble the step by step algorithm or the formula into a set of mnemonic program instruction steps which precisely describes the operations to be done. When completed, this list will become the computer program.

Example continued: In this example we wish to create a list of mnemonic program instruction steps to completely describe the two formulas below.

Number of triangles above the Nth line = N^2

Number of matches above the Nth line = $\frac{3N^2 + N}{2}$

Furthermore, a method must be provided to enter the number N into the computer, and to print the answer on the output when finished. The computer must also be told to stop at the appropriate time. Let's start.

00	ENIA, 90	Computer programs always start in storage location 00. In this case, the first step will allow you to enter the line number N into storage location 90.
01	LDRA, 90	This puts the number N into Register A.
02	MULA, 90	This forms the number N^2 in Register A.
03	JJA1, ??	The number of triangles above line N equals N^2, and that number is now in Register A. Let's say that if Jump Switch A is set to 1 it means: **find the number of small triangles formed.** In that case, an output message like "9 triangles above line 3" would be printed. The location to which the program will jump, ??, will be determined later.
04	MULA, ??	If the program gets to step 04, it didn't jump in step 03; therefore, Jump Switch A must not have been set to 1. In this case, it means that the number of matches above the Nth line is desired, and that requires the formula $\frac{3N^2 + N}{2}$. Remember that N^2 is already in Register A, so the computer must next form $3N^2$. The 3 used in this multiplication will be located somewhere in the program at a location to be determined later.

05 ADDA, 90 — Register A will contain $3N^2 + N$ when this step is done.

06 DIVA, ?? — Register A now contains $3N^2 + N$ which must be divided by 2 to complete the formula describing the number of matches above the Nth line. The 2 used for the division will be stored at a location which will be determined later.

07 STRA, 91 — Register A now has the answer which must be stored and printed on the output.

08 PROA, 91
09 PROA, ??
10 PROA, ??
11 PROA, 90

Suppose the message to be printed looks like the sample at the right. Remember that the "15" in this case is the answer that was stored in storage location 91 in step 07, while the "3" is the line number N which was entered into storage location 90 at the start. The ?? in step 09 represents the location of the word "matches," while the ?? in step 10 locates the words "above line."

15
matches
above line
3

12 STOP, 12 — Remember that the computer will continue to run unless told to stop.

03 JJA1, 13 — Recall that step 03 said JJA1, ?? because the location ?? was not known before. Since the first part of the program ends at location 12, the second part can start at 13, so complete step 03 to read JJA1, 13.

13 STRA, 91
14 PROA, 91
15 PROA, ??

The program can only get to step 13 by jumping from step 03. If this happens the number of *triangles* is required and an output like the one at the right is appropriate. The ?? in step 15 will locate the word "triangles."

9
triangles
above line
3

16 JUMP, 10 — Go back and look at steps 10, 11 and 12. These three steps will complete the output message, so let's jump back to 10 and use them again.

17 2
18 3
19 matches
20 triangles
21 above line

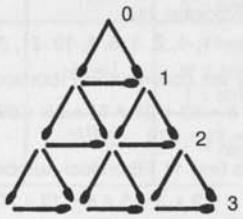
Five data entries were used in this program, and each of them must be assigned a place in the Program Storage Unit. The assignments at the left represent one possibility.

04 MULA, 18
06 DIVA, 17
09 PROA, 19
10 PROA, 21
15 PROA, 20

After the data entries have been assigned to some storage location, steps 04, 06, 09, 10 and 15 can be completed.

Step #4 Write the complete program, along with some simple directions on its use.

If N represents a line number in a triangular array of matches as shown at the right, then this program will calculate either the number of small triangles formed above the Nth line if Jump Switch A is set to 1, or the number of matches above the Nth line if Jump Switch A is set to 0.




```
00  ENIA, 90      Enter  N
01  LDRA,90
02  MULA, 90
03  JJA1, 13
04  MULA, 18
05  ADDA, 90
06  DIVA, 17
07  STRA, 91
08  PROA, 91
09  PROA, 19
10  PROA, 21
11  PROA, 90
12  STOP, 12
13  STRA, 91
14  PROA, 91
15  PROA, 20
16  JUMP, 10
17  2
18  3
19  matches
20  triangles
21  above line
```

Suggestions for Future Paper Computer Programs

1. One could spend a lifetime working with the seemingly endless properties of the Fibonacci numbers which were partly produced by Program #12.

   1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987 ...

   a. The sum of any ten consecutive Fibonacci numbers equals eleven times the seventh number of the list.

      Example: 5 + 8 + 13 + 21 + 34 + 55 + 89 + 144 + 233 + 377 = 979
      89 × 11 = 979

   b. The sum of the first N Fibonacci numbers is one less than the (N + 2)th Fibonacci number.

      Example: 1 + 1 + 2 + 3 + 5 + 8 + 13 = 33
      13 is the seventh Fibonacci number, so the sum must be one less than the (7 + 2)th Fibonacci number which is 34. 33 = 34 - 1 as predicted.

   c. The square of any Fibonacci number is one more or less than the product of the numbers next to it in the sequence.

      Examples:

3, 5, 8	5 × 5 = 25	3 × 8 = 24
5, 8, 13	8 × 8 = 64	5 × *13* = 65
21, 34, 55	34 × 34 = 1156	21 × 55 = 1155

   d. Every 3rd Fibonacci number is evenly divisible by 2.
      Every 4th Fibonacci number is evenly divisible by 3.
      Every 5th Fibonacci number is evenly divisible by 5.
      Every 6th Fibonacci number is evenly divisible by 8.
      Every 7th Fibonacci number is evenly divisible by 13, and so on.

   e. The number of clockwise and counter clockwise spirals in a pine cone are usually consecutive Fibonacci numbers: (3, 5), (5, 8), (8, 13).

   Lucas, and other generalized Fibonacci type number sequences, have similar interesting properties. Write a program to calculate the general Fibonacci sequence described by

   a, b, (a + b), (a + 2b), (2a + 3b), (3a + 5b), (5a + 8b) ...

   where a and b are any two numbers entered into the Paper Computer on the first few program steps. The program will be similar to Program #12 in this book. The number sequence thus produced on the output tape of the Paper Computer will make an interesting and stimulating topic for a class lesson.

   Example: If a = 2 and b = 4, the sequence produced would be
   2, 4, 6, 10, 16, 26, 42, 68, 110, 178 ...

   Just to maintain your interest, you might notice that the sum of the first N term of this sequence is four less than the (N + 2)th term.

   Example: 2 + 4 + 6 + 10 + 16 + 26 + 42 = 106
   42 is the sixth term of the sequence; 110 is the eighth.
   110 - 4 = 106

2. Write a program to calculate the diameter, circumference and area of a circle by having an ENIA, SS instruction step for entering only the radius of the circle.

   diameter = 2r circumference = $2\pi r$ area = πr^2

3. Write a program to calculate the area or the perimeter of a rectangle, depending upon the setting of a jump switch. Two ENIA, SS instruction steps will be needed: one to enter the length of the rectangle, the other to enter the width. If the area is calculated, a multiplication must be performed. If the perimeter is needed, the width and length must be added together and doubled. The idea can be expanded to include both areas and perimeters of triangles, trapezoids, regular polygons, etc.

4. Write a program to calculate the volume or surface area of a common solid using any of the well known formulas.

Solid	Surface Area	Volume	Terms
Cube	$6\,h^2$	h^3	h = height
Rectangular Prism	$2\,(hw + hl + wl)$	$h\,w\,l$	h = height w = width l = length
Sphere	$4\pi r^2$	$\frac{4\pi r^3}{3}$	r = radius

5. When the difference between any two consecutive numbers of a sequence is the same, the sequence is said to be an arithmetic sequence.

   3, 7, 11, 15, 19, 23, 27 is an arithmetic sequence.

   F = 3 means that the first term is 3.
   L = 27 means that the last term is 27.
   d = 4 means that the arithmetic difference between the terms is 4.
   n = 7 means that only 7 terms of the sequence have been written, or are under consideration.

   The following formulas apply to arithmetic sequences, and each can be built into an interesting program for the Paper Computer.

   $$L = F + (n - 1)d$$

   Example: Seven terms of the arithmetic sequence above are shown. The first term is 3, and the difference is 4. Therefore, the last term must be

   $L = 3 + (7 - 1)4 = 3 + 6 \times 4 = 3 + 24 = 27$.

   The sum, S, of the terms of an arithmetic sequence is given by

   $$S = \frac{n\,(F + L)}{2} \quad \text{or} \quad \frac{n\,(2F + d(n - 1))}{2}.$$

   Example: Seven terms of the arithmetic sequence above are shown. The first term is 3, and the difference is 4. The sum of these terms must, therefore, be

   $$S = \frac{7\,(2 \times 3 + 4\,(7 - 1))}{2} = \frac{7\,(6 + 24)}{2} = \frac{7 \times 30}{2} = \frac{210}{2} = 105.$$

   Or, using the first formula,

   $$S = \frac{7\,(3 + 27)}{2} = \frac{7 \times 30}{2} = \frac{210}{2} = 105.$$

   Check it for yourself: $3 + 7 + 11 + 15 + 19 + 23 + 27 = 105$.

6. The number of surfaces (F), the number of edges (E), and the number of vertex points (V) of any simple solid (those with no holes in them) are related by Euler's function: $V + F = E + 2$. This formula makes an interesting computer program by calculating any one of the three variables F, E or V after entering the other two. For example, a simple cube has six surfaces and eight vertex points, so it must have

   $$E = V + F - 2 = 8 + 6 - 2 = 14 - 2 = 12 \text{ edges.}$$

A tent in the shape of a square pyramid will have four sides and a bottom (F = 5) and eight seams where it is sewn together (E = 8). It must, therefore, have

$$V = E + 2 - F = 8 + 2 - 5 = 10 - 5 = 5$$ vertex points where the sides meet.

The four vertex points on the bottom are for the tent pegs; the other vertex point at the top is for the center tent pole.

7. Write a program to separate the first ten or so numbers into primes or composites. Print the prime numbers on Output A, and the composites on Output B.

8. Write a program to keep track of one person's bank account. Let the input of a positive number represent a deposit, while negative inputted numbers represent withdrawals. Check for the condition where a customer tries to withdraw more money than is in the account. Make the program calculate and add interest to the balance when run with Jump Switch A set to 1.

9. Create a conversion program which converts miles, yards or feet to inches, depending upon the setting of the Jump Switches. For instance, if Jump Switch B = 1, it could mean that the number entered at Input A represents a quantity in yards which requires a multiplication by 36 to convert it to inches. A similar program can be written for temperature conversion between centigrade and Fahrenheit, between weights for ounces, pounds, tons, etc., and between many other types of units. A program to convert English and metric systems of measurements will require multiplication by decimal numbers.

10. The average cost of driving an automobile one mile, and the average miles per gallon obtained by that automobile are given by the formulas below.

$$\text{Cost per mile} = \frac{\$}{M_2 - M_1} \qquad \text{Miles per gallon} = \frac{M_2 - M_1}{g}$$

where, M_1 = the mileage reading when you fill the car's fuel tank

M_2 = the mileage reading when you fill it up the next time

$\$$ = the cost of the second fill up

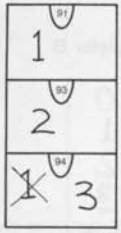
g = the number of gallons of fuel used at the second filling.

The formulas above not only make an interesting program, but they make a very useful and practical one.

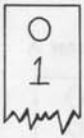
ANSWERS TO PROGRAMS

Program #1

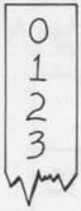
Storage



Register A



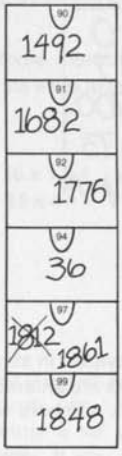
Register B



Output A

Program #2

Storage



Register A

Register B

Output A

Program #3

Storage

Register A

Register B

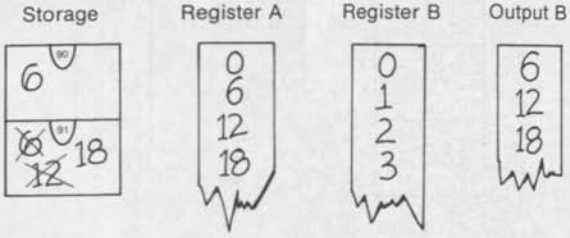
Output A

Program #4 As an example for Program #4, suppose that the Jump Switches are set as shown below. If so, the readings of the Program Step Indicator and the message of Output A are shown at the right.

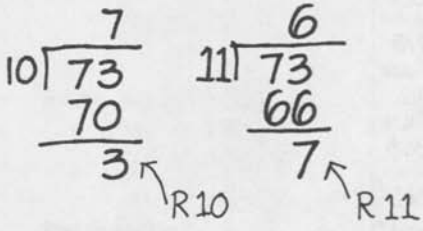
Jump Switch A = 1
Jump Switch B = 0
Jump Switch C = 1

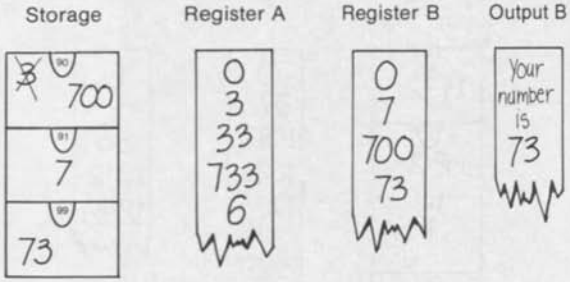
Program Step Indicator	Output A
00	
01	Ralph is the ace
14	
29	
30	
18	
19	

Program #5 As an example for Program #5, suppose you enter a 6. The units of your Paper Computer would appear as shown at the right.



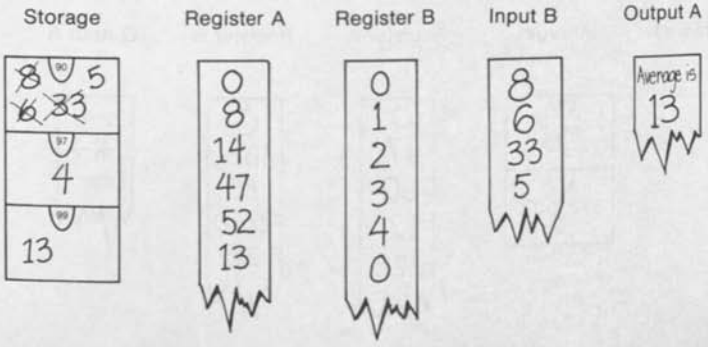
Program #6 As an example for Program #6, the number 73 will be used as the secret number. The calculations for R10 and R11 for this number are shown below. The units of the Paper Computer are shown at the right.



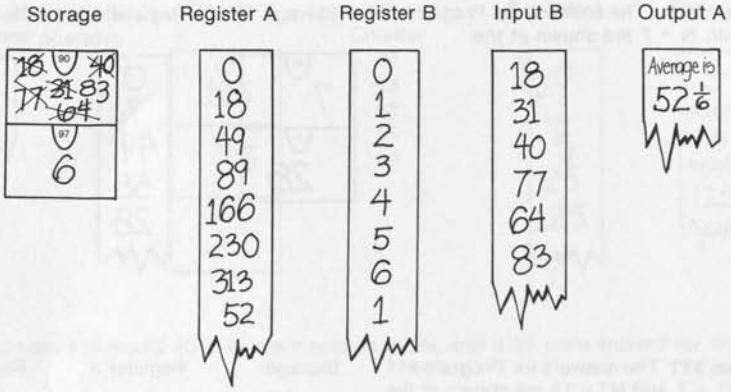


Program #7 Program #7 operates differently depending upon whether or not the division in step 06 has a zero remainder. The first set of answers below shows what will happen when four numbers are entered which produce a zero remainder. The second set of answers shows the results when entering six numbers which produce a nonzero remainder.

For N = 4:



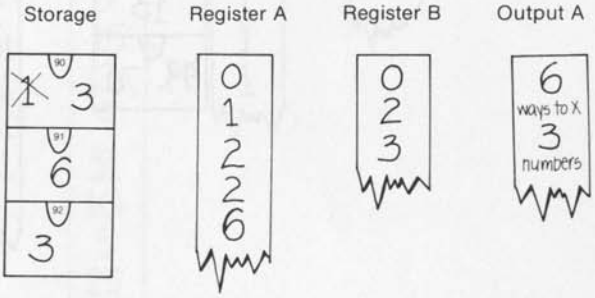
For N = 6:



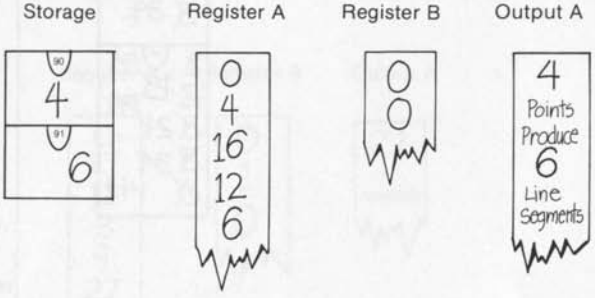
Program #8 This program asks you to use the commutative and associative laws to mix up a group of numbers in as many ways as possible. Your Paper Computer will tell you how many ways there are. The answers for a set of three numbers (N = 3) are given below.

Suppose the three numbers were 7, 16, 4. Then the six ways to multiply them together are:

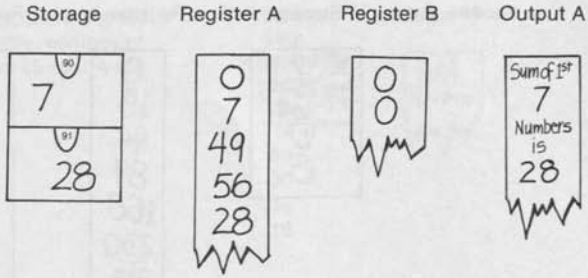
7 x 16 x 4	16 x 7 x 4	4 x 7 x 16
7 x 4 x 16	16 x 4 x 7	4 x16 x 7



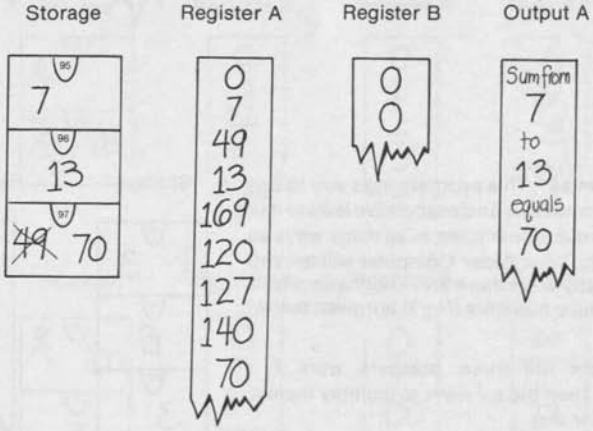
Program #9 This program will tell you how many line segments can be produced by connecting together a set of N points in every possible way. If you try to demonstrate this by drawing a set of points and connecting them, it is easiest to count the line segments if you make sure that no more than two points are on the same line segment. The answers for N = 4 are given at the right.



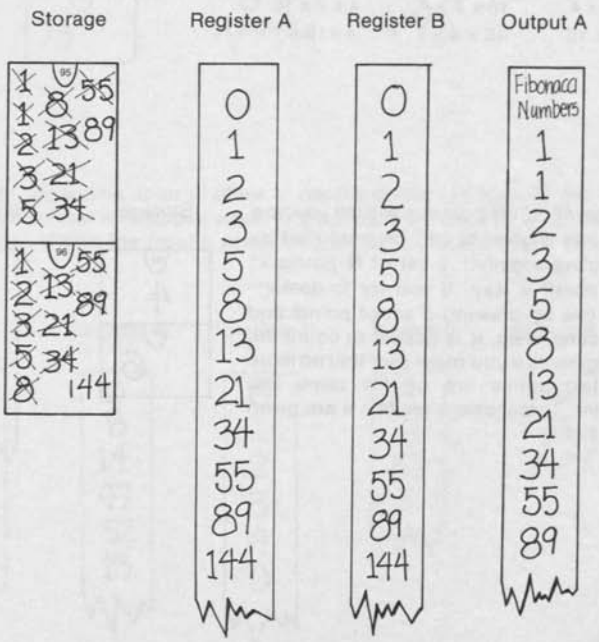
Program #10 The answers for Program #10 with N = 7 are shown at the right.



Program #11 The answers for Program #11 with N1 = 7 and N2 = 13 are shown at the right.

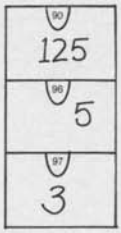


Program #12 The Fibonacci numbers as produced by Program #12 are shown at the right.



Program #13 The answers at the right will show you how the Index Counter operates to calculate 5 to the exponent 3.

Storage

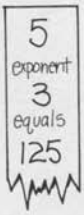


Index Counter

00
03
02
01
00

Register A

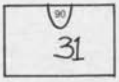
Output A



Program #14 Here is what Program #14 would do if 31 were entered for N, and if 27 were entered for N.

For N = 31:

Storage

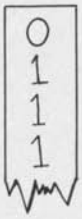


Index Counter

00
21
22
23
24

Register A

Register B



Output A

For N = 27:

Storage

Index Counter

00
21
22

Register A

Register B

Output A

Photo by Gary Rowe

Fred C. Matt is presently a mathematics teacher and Chairman of the Math Department of Lower Moreland Middle School in Huntingdon Valley, Pennsylvania.

He received an Associate in Technology Degree in Electronics from the Technical Institute of Temple University, a B.A. in Physics from Temple University, an M.A. in Mathematics from The Pennsylvania State University and has done postgraduate work in Mathematics Education and Management.

In addition, Mr. Matt has an extensive background in computer technology and engineering, and has been an industrial instructor of computer physics and electronics.

He has lectured numerous groups on teaching academically talented students in mathematics.

In his spare time, Mr Matt plays the guitar, goes backpacking and camping, is a leader with the Boy and Girl Scouts, travels to locations important in U.S. history and constructs numerous machines and equipment from recycled found materials.

ISBN 07-530136-9