Science Fair

MICROCOMPUTER TRAINER

Cat. No. 28-260

Requires six "AA" batteries

For Ages 12 to Adult

TABLE OF CONTENTS

INTRODUCTION

On the kitchen table, next to a stack of bills and a pot of coffee, sits a computer. A colorful pie chart divided into uneven sections of the household budget, covers the screen. The printer, alongside the computer, whirs away as your mother awaits the printout.

In the den, down the hall, your father is busily sending a memo via telephone on his portable briefcase-size computer.

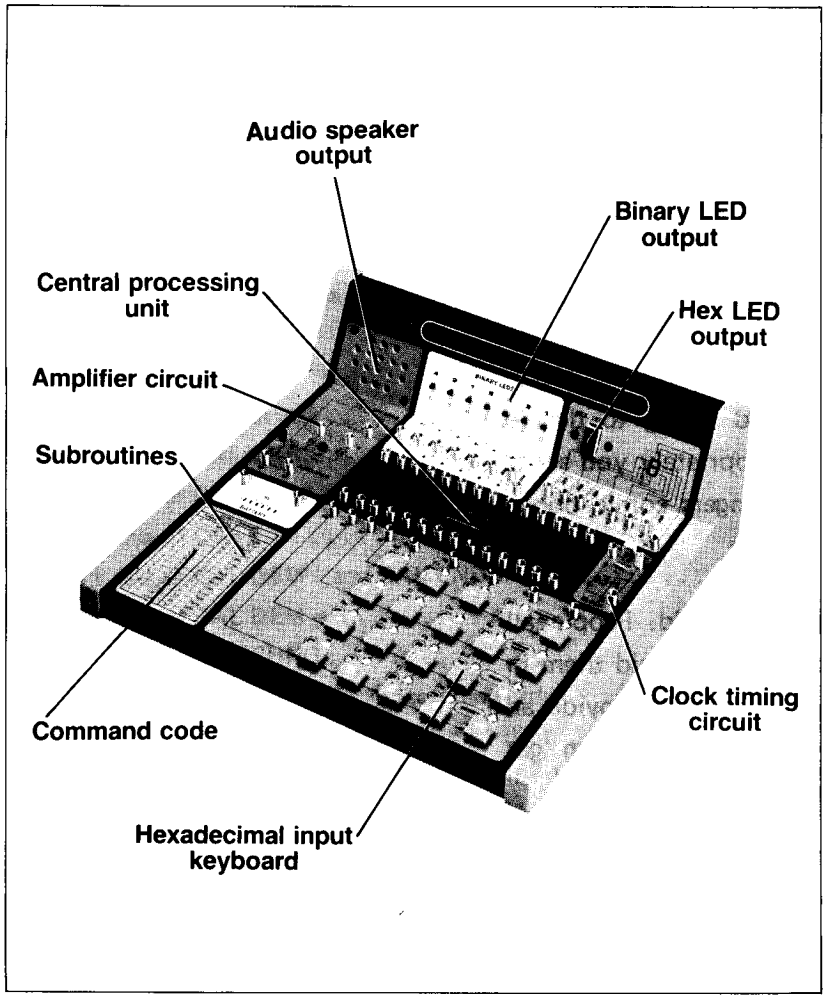
And you are standing in the doorway of your bedroom, carrying your Microcomputer Trainer Kit, and trying to decide where would be the best spot for it. Appropriately you set it on your desk next to a stack of Buck Rogers comic books and a pile of video arcade machine tokens.

Yours is the typical Computer Age household. People all over the world are using home computers to 1) aid them in business matters, 2) organize the home, and 3) provide an enjoyable diversion. And everyone in the family can get involved.

So now it's your turn. You have your Microcomputer Kit and you're ready to tell it to do something. (Remember a computer doesn't think; it only does what you program it to do.)

But first let's get to know your Computer.

Features and Controls



1) BINARY LEDs
These seven Light Emitting Diodes (LEDs) light up in different combinations to correspond to the binary numbering system. The binary numbering system uses only two digits, 1 and 0. Computers carry out all operations in binary.

2) HEX. LED
This LED displays the characters 0-9 and A-F. "Hex" is short for hexadecimal, a numbering system based on 16 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F).

3) 0-9, A-F Keys
These keys let you enter hex numbers into the computer.

4) ADDRESS (ADRS) SET Key
The memory of the microcomputer is divided into "addresses." Each address holds one character of data or information. When you press two number/letter keys and then ADRS SET, the binary LEDs corresponding to those two numbers will light. The HEX. LED will show the contents of that address.

5) INCREMENT (INCR) Key
This key loads data into the memory address and tells the computer to display the contents of the next address.

6) RUN Key
This key executes a program.

7) RESET Key
To reset and display the contents of address 00 on the HEX. LED, press this key.

The Microprocessor

The heart of this Microcomputer Trainer Kit is its microprocessor or Central Processing Unit, a tiny package located in the center of the kit. It measures about one inch by nearly half an inch and is made up of tens of thousands of transistors. Think of the microprocessor as the control center or "brains" of your computer.

Most computers, including your trainer kit, are made up of a microprocessor, some input and output circuits, and some memory.

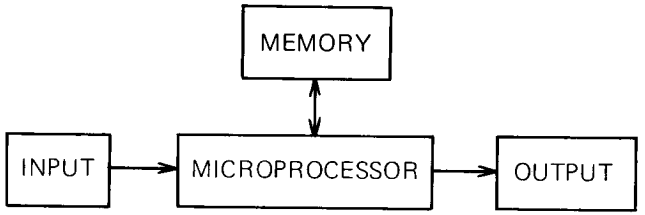


Figure 1

The memory sections of your computer consists of two parts —the RAM and the ROM. RAM (for random access memory) is the memory area where you may "write" and "read" information. The information can be anything from numbers and words to programs and instructions. RAM memory is lost whenever you turn power off. For this reason RAM is also called "volatile" memory, meaning not-permanent.

ROM memory (or read only memory), on the other hand, is permanently written into the computer. There is no possible way to alter this information. Your Microcomputer Trainer Kit contains various game programs and monitoring programs (that help you to read, write, and excute other programs). The information in the ROM is written in at the time of manufacture. Nothing you do (short of melting down your computer) can erase information from the ROM area.

Inside your microprocessor are a variety of subsections and registers. (A register is a temporary memory.) All around it are inputs. Information comes in through these inputs and is stored or processed in the subsections. Sometimes information is output. Figure 2 shows your micro chip in detail. Each input and subsection is explained below.

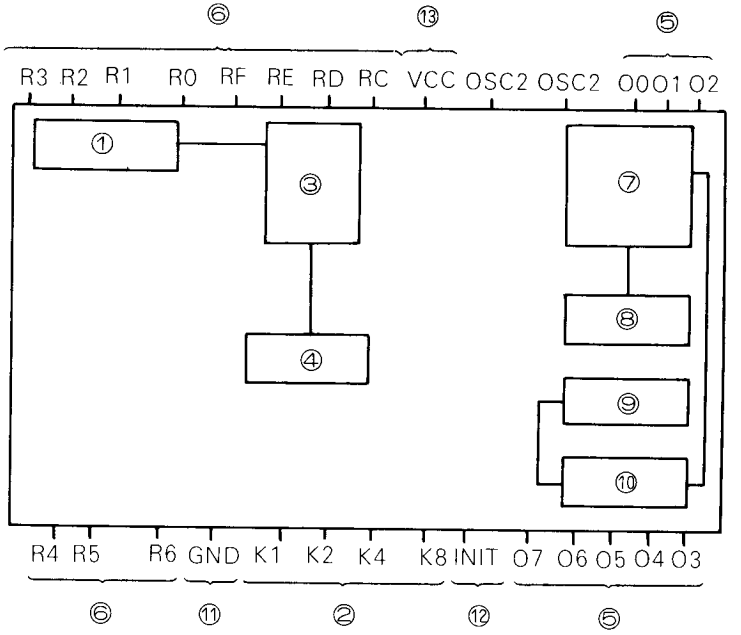


Figure 2

1) Program counter — a circuit that keeps track of the execution of steps in a program.

2) K input (4-bit) — a circuit that reads the key input

3) 2,048 word ROM (8 bit/word) — the read-only memory where permanent data (2,048 characters or group of characters) is stored

4) Command decoder — a circuit that determines the meaning of a set of signals and carries out the command

5) O outputs (8 bit) — circuits that control some key inputs and the lighting of the HEX. LEDs

6) R outputs (11 bit) — circuits that control some key inputs, the lighting of the binary LEDs, and the speaker signal

7) 128 word RAM (4 bit/word) — the read/write memory in which data can be erased or changed

8) X register — a temporary memory

9) Y register — a temporary memory

10) Accumulator register — a holding memory from which data is fetched and stored, i.e. part of an arithmetic operation and the end result of the operation can be stored here.

11) GND input — connects to ground

12) INIT input — a circuit that controls the starting of a program

13) Vcc — a circuit that connects to the battery terminal

Once you have your kit wired, you can see more clearly how the inputs link the various sections of your computer with the chip.

WIRING

Install 6 AA batteries into the battery compartment on the back of your kit. Observe correct polarity as shown in figure 3. To avoid running down the batteries, always remove them when you are not using your kit.
If the display becomes erratic or weak, replace all the batteries.

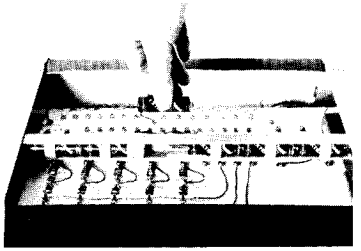


Figure 3

Your kit comes supplied with enough wires of different lengths to make all connections. Use the shorter wires to reach terminals that are closer together and the longer wires for more distant connections.

To make connections, bend the spring terminals and slide the exposed wire tip (or lead) inside the spring with long-nose pliers as shown in figure 4. Double-check all connections.

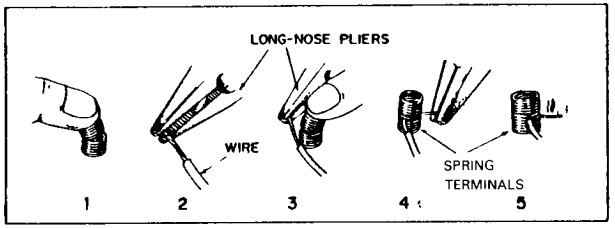


Figure 4

Binary LEDs Lighting Check

Connect one end of a wire to the negative terminal of the battery (54) and the other end to the ground terminal of the LEDs (2) as shown in figure 5. Attach a lead to the positive battery terminal (53) and with the other lead end touch each of the binary LED terminals—(10), (9), (8), (7), (6), (5), and (4). The LEDs should light in turn.

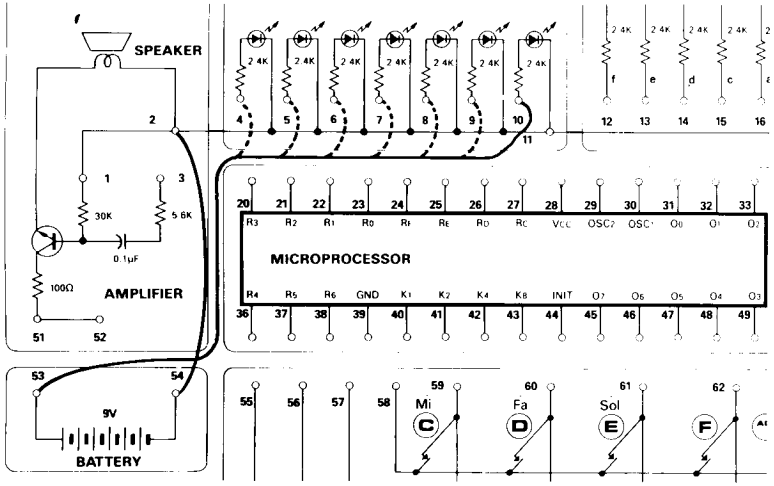


Figure 5

HEX. LED Lighting Check

Leave the negative battery terminal (54) connected to the ground terminal of the LEDs (2). Connect another lead to the positive battery terminal (53) and touch the other end of the lead to each of the HEX. LED terminals—(18), (17), (16), (15), (14), (13), and (12). (See figure 6.) Each time you touch a terminal a different segment of the HEX. LED should light.

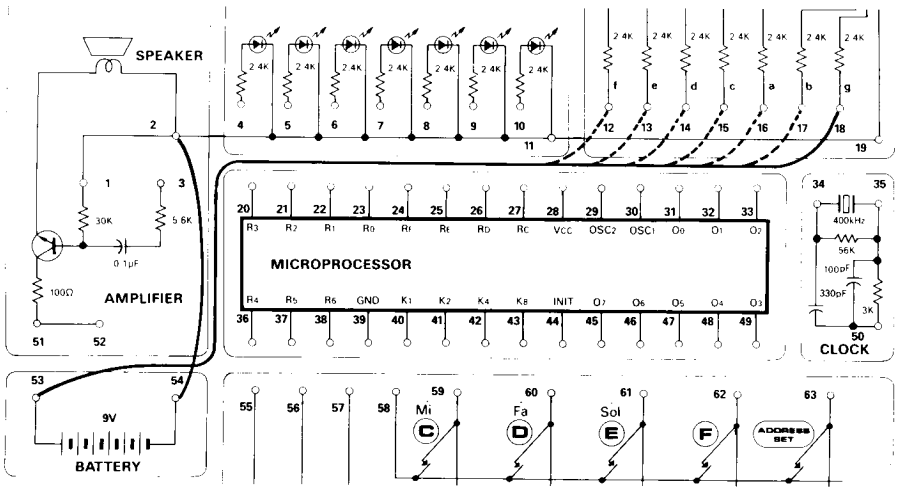


Figure 6

CPU Check

Binary LEDs

Now we'll check the functions of the central processing unit with the binary LEDs and the keyboard. Make these connections – see figure 7:

(53)–(28)	(1)–(39)	(30)–(34)	(29)–(35)
(19)–(50)	(45)–(63)	(11)–(44)	(42)–(57)
(10)–(23)	(9)–(22)	(8)–(21)	(7)–(20)
(6)–(36)	(5)–(37)	(4)–(38)	(40)–(55)
(2)–(54)			

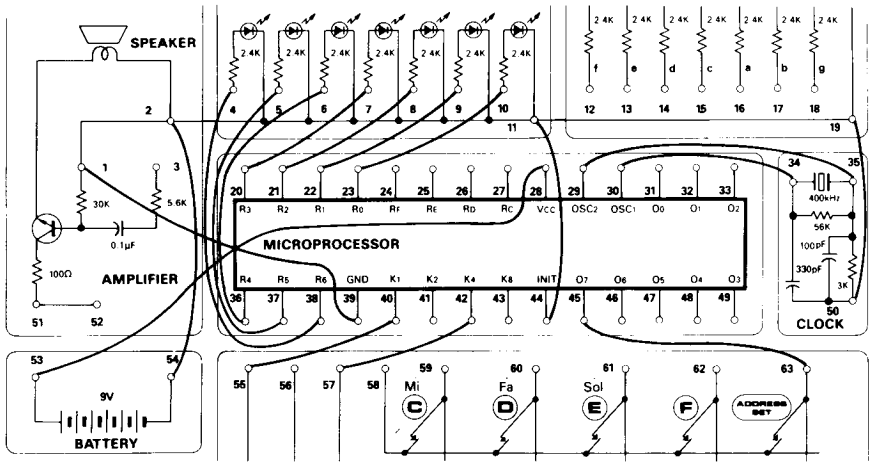
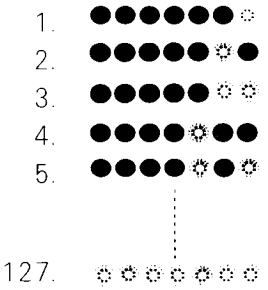


Figure 7

When you have completed these connections, press the RESET key – all the LEDs will turn off. Then press INCR; each time you press INCR a different combination of binary LEDs light as shown below.



This binary sequence is the computers method of counting. (More on the binary system later.)

HEX. LEDs

Now check the CPU functions controlling the HEX. LED and the keyboard. See figure 8 and make the following additional connections. (Dotted lines show existing connections.)

(18)–(33)	(17)–(32)	(16)–(31)	(15)–(49)
(14)–(48)	(13)–(47)	(12)–(46)	(27)–(59)
(26)–(60)	(25)–(61)	(24)–(62)	(43)–(58)
(41)–(56)			

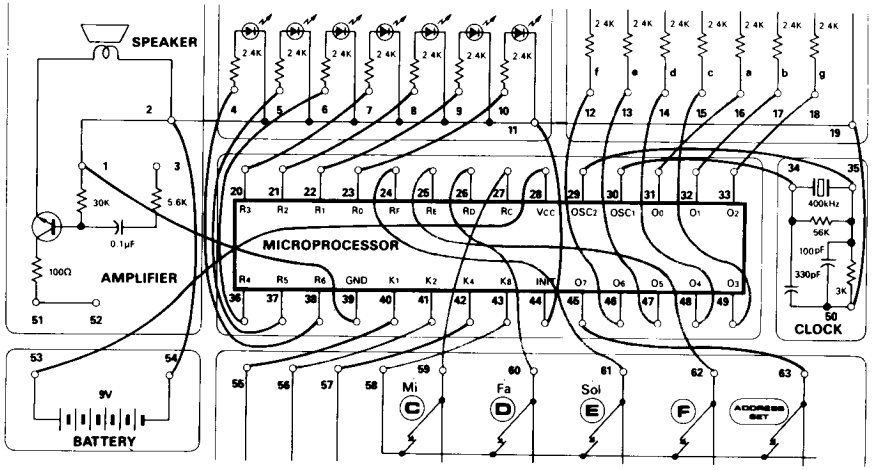
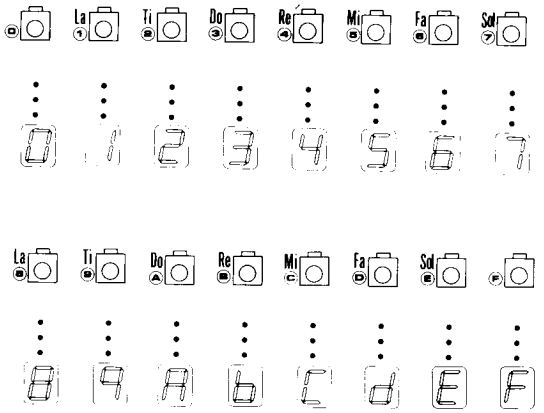


Figure 8

Press the 0-F keys and check that the appropriate HEX. LED segments light:



Speaker Amp

Finally, connect the CPU to the speaker amp with these additional connections – see figure 9:

(51)–(53) (3)–(20)

Repeat the key operations for Binary LED CPU check and HEX. LED CPU check (INCR and 0-F keys). You should hear a beep every time you press a key.

Once these connections are made, there is no need to make changes for any of the programs. You may want to disconnect the wire connected to terminal 54 when you're not using your kit. This wire functions as the on/off switch.

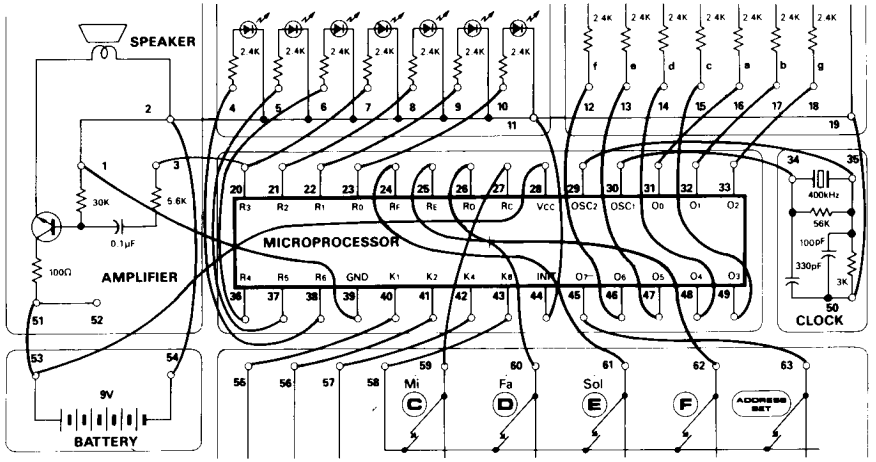


Figure 9

LET'S COUNT IN COMPUTER!

The Binary System

People think and talk with words – machines think and talk with numbers. Using numbers involves counting and to most of us counting means decimal counting – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Decimal counting is a system based on 10 digits; that is why we refer to it as a base ten numbering system. But this is only one among a variety of counting systems.

Computers count best in binary because there are only two digits (bits) to remember – 0, and 1; this is a base two numbering system. This system adapts well to computers because the two digits – 1 and 0 – can represent positive and neutral voltage respectively to the computer. Think of it as a switch – when the switch is on (1 position) a positive voltage is present; when it's off (0 position) there's no voltage – it's neutral. By combining a series of 1s and 0s, a computer can represent any number and perform a variety of operations on them.

In decimal, and in binary, the position of each digit gives it a particular value. That value depends on the base of the numbering system. In base ten, the columns from right to left correspond to 1, 10, 100, 1000..... The number 179 consists of:

hundreds	1×10^2
tens	7×10^1
ones	9×10^0

The value of each position increases as the base number (10) is increased:

$\ldots 10^5 \quad 10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0$

This process is the same for any other numbering system; the only thing that changes is the base number. In binary, we have

a two base numbering system. So the value of each position is increased by increasing the base number 2:

$\ldots 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

While the digit values in decimal look like this:

100,000 10,000 1,000 100 10 1

those in binary look like this:

32 16 8 4 2 1

Binary to Decimal Conversion

With the information given in the previous section, it is relatively easy to convert numbers from binary into decimal. Consider the following binary number:

1101

There are two steps for converting 1101 to its decimal equivalent. First, multiply each of the binary digits by its column value:

$1 \times 2^0 = 1 \quad 0 \times 2^1 = 0 \quad 1 \times 2^2 = 4 \quad 1 \times 2^3 = 8$

Next, add together all the products. The number you get by adding 8, 4, 0, and 1 is 13—the equivalent of binary 1101. 1101 in binary = 13 in decimal.

In binary counting, we can never count more than 1 without carrying over to the next column. For example, the binary number corresponding to decimal 2 is a one in the twos column and a 0 in the units column. (See the following chart.) Decimal 5 is a 1 in the fours column and a 1 in the units.

Use this chart to help you convert binary into decimal.

BINARY LED	BINARY No.	DECIMAL No.
●●●●●●●	0000000	0
●●●●●●○	0000001	1
●●●●●○●	0000010	2
●●●●●○○	0000011	3
●●●●○●●	0000100	4
●●●●○●○	0000101	5
●●●●○○●	0000110	6
●●●●○○○	0000111	7
●●●○●●●	0001000	8
●●●○●●○	0001001	9
●●●○●○●	0001010	10
●●●○●○○	0001011	11
●●●○○●●	0001100	12
●●●○○●○	0001101	13
●●●○○○●	0001110	14
●●●○○○○	0001111	15
●●○●●●●	0010000	16
●●○●●●○	0010001	17

In binary you may see a 7-character number like this:

1 (64s) 1 (32s) 1 (16s) 1 (8s) 1 (4s) 1 (2s) 1 (1s)

All 7 LEDs light for this binary number, which corresponds to decimal 127. (64+32+16+8+4+2+1=127)

Adding Binary Numbers

Now that you've learned a new way to count, it's time to learn the new math that goes with it. Binary addition is really simpler than it sounds. Just remember these rules:

1. 0 + 0 = 0
2. 0 + 1 = 1
3. 1 + 0 = 1
4. 1 + 1 = 0, carry 1 = 10
5. 1 + 1 + 1 = 10 + 1 = 11

Try working these sample problems:

A.	B.	C.
1001	101	10010
+ 100	+ 11	+ 1111
?	?	?

Answers: A. 1101 B. 1000 C. 100001

Subtracting In Binary

Most computers like the simple way out. They like two-digit number systems instead of nine-digit. And they prefer learning one arithmetic method to two, or three, or four. So when you tell most computers to subtract binary numbers, they subtract – by adding. Sound complicated? It's not!

The method used in "subtracting" is called Two's Complement Notation. The binary number to be subtracted is complemented or inverted by changing all the 1s to 0s and all the 0s to 1s. Then the two numbers are added. Add a 1 to the result and ignore any carry over and you have your answer!

For example:

14	=	1110	=	1110
– 9		–1001		+0110
5				10100
				+ 1
				101

= 5 (Remember that the carried 1 is ignored.)

That's not so bad, is it?

Now you try working a few:

A.	B.	C.
1100	1111	1000
–1010	–1011	–0111
?	?	?

Answers: A. 0010, B. 0100, C. 0001

The Hexadecimal System

Decimal to Hex Conversion

The second counting system your computer uses is hexadecimal. The "hex" system counts to a base 16. The first 10 numbers in this system look very much like the decimal system, but the next 6 are represented by letters – A, B, C, D, E, F. (See the chart below.)

Hexadecimal numbers are two digit combinations of numbers and/or letters. The hex system can be conveniently used as a shortened version of binary.

We can count up to 15(F) in hexadecimal before carrying over. Hex numbers/letters and their corresponding decimal numbers are listed in the chart below.

DECIMAL	BINARY	HEXADECIMAL
0	0000	00
1	0001	01
2	0010	02
3	0011	03
4	0100	04
5	0101	05
6	0110	06
7	0111	07
8	1000	08
9	1001	09
10	1010	0A
11	1011	0B
12	1100	0C
13	1101	0D
14	1110	0E
15	1111	0F
16	10000	10

Figure 10

To convert a decimal number into hex, first convert it into binary. Do this by dividing by 2, writing down the remainder, dividing that quotient by 2, write down the remainder, and so on until the final quotient is 1 or 0. Then reverse the order of the remainders and voila! your binary number.

For example: 39 decimal = ? binary

	quotient	remainder
39 ÷ 2 =	19	1
19 ÷ 2 =	9	1
9 ÷ 2 =	4	1
4 ÷ 2 =	2	0
2 ÷ 2 =	1	0
		1

(This 1 is the final quotient which also becomes the final remainder).

The remainders in reverse order = 100111, which is 39 in binary. (You can check this by reversing the process – binary to decimal – which you have already learned.)

Now let's take our binary number and divide it into 4-bit sections: 0010 0111 (you can always add 0s to the beginning of the number to make it 4 bits.)

Your new number is easy to convert into hex. Just use the chart in figure 10. 0010 or 2 = 02 in hex and 0111 or 7 = 07 in hex. Drop the 0s and you have the hex number 27 (two seven not twenty-seven).

Check this by converting your new hex number (27) back to decimal. This is very easy to do. Just multiply the first number (2) by 16 and add the second number (7) to the total.

$$16 \times 2 = 32 \qquad 32 + 7 = 39$$

It works!

Consider these numbers: 17, 60, 127.
17 = 10001 in binary = 0001 0001 = 11 in hex.
60 = 111100 in binary = 0011 1100 = 3C in hex.
127 = 1111111 in binary = 0111 1111 = 7F in hex.

Practice converting decimal numbers into binary and hex. Before you know it, you'll have the hang of it!

Hexadecimal Arithmetic

Hex numbers can be added, subtracted, multiplied, and divided. The easiest way to perform this math is to convert the hex numbers to decimal, do the arithmetic, and then convert your answer back to hex. We've already learned the conversion processes, but it won't hurt to go over them. Let's try a few hex problems:

1. 1A + 13 = ? 1A = (16 x 1) + A (or 10) = 16 + 10 = 26
   13 = (16 x 1) + 3 = 16 + 3 = 19

   26 + 19 = 45
   Convert 45 into binary by dividing by 2.

	quotient	remainder
45 ÷ 2 =	22	1
22 ÷ 2 =	11	0
11 ÷ 2 =	5	1
5 ÷ 2 =	2	1
2 ÷ 2 =	1	0
		1
		(final quotient)

   The binary number is 101101 or:
   divided into 4-bits: 0010 1101
   and converted to hex: 2 D

   So, 1A + 13 = 2D

2. 36 – 14 = ? 36 = (16 x 3) + 6 = 48 + 6 = 54
   14 = (16 x 1) + 4 = 16 + 4 = 20

   54 – 20 = 34
   Now convert 34 to binary by dividing by 2.

	quotient	remainder
34 ÷ 2 =	17	0
17 ÷ 2 =	8	1
8 ÷ 2 =	4	0
4 ÷ 2 =	2	0
2 ÷ 2 =	1	0
		1
		(final quotient)

Your binary number is 100010 or:
divided into 4-bits: 0010 0010
and converted to hex: 2 2

So, 36 – 14 = 22

3. 12 x 6 = ? 12 = (16 x 1) + 2 = 16 + 2 = 18
   6 = 06

   18 x 6 = 108

	quotient	remainder
108 ÷ 2 =	54	0
54 ÷ 2 =	27	0
27 ÷ 2 =	13	1
13 ÷ 2 =	6	1
6 ÷ 2 =	3	0
3 ÷ 2 =	1	1
		1
		(final quotient)

   1101100 = 0110 1100 = 6C

   So, 12 x 6 = 6C

4. 22 ÷ 3 = ? 22 = (16 x 2) + 2 = 32 + 2 = 34
   3 = 03

   34 ÷ 3 = 11 remainder 1
   11 = B in hex

   So, 22 ÷ 3 = B remainder 1

☆ MICRO GAMES

Seven Games to play now follow:

No.1 Electronic Organ
No.2 Automatic Tune-playing
No.3 Musical Guessing Game
No.4 Rat Catching
No.5 Tennis Game
No.6 Timer
No.7 Morse Code

Play these games to get some practice with the keyboard.

These game programs are stored permanently in the Micro-computer Trainer ROM memory; they cannot be "lost" when the batteries are taken out.

Later you will be learning how to load programs into the micro and how to run them; later still you will be learning how to write programs yourself.

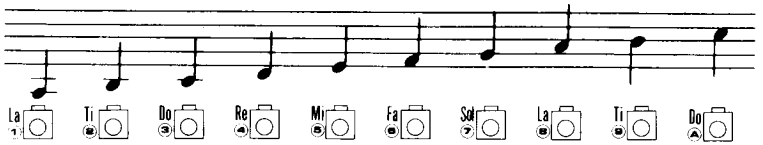
Gradually you will become very familiar with ideas which may seem difficult at the moment. Don't worry! Keep trying and at the end, if you stick at it, you will be a computer expert.

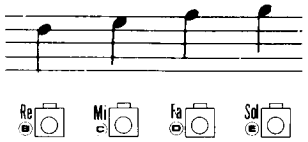
No.1 Electronic Organ

Let's begin with a program that converts the micro into a musical keyboard. Each key 1-9 and A-E plays a different note.

Press RESET, 9, RUN in that order.

That will start the Electronic Organ program. Now play the scale below by pressing the keys shown:





* The sound stops as soon as the key is released.

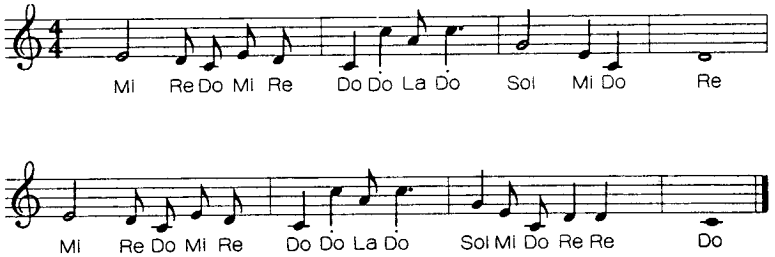
Learn the musical notes and then try to play a tune with which you are familiar.

The following are the Sol-Fa names for each note:

Key							
	1	La	6	Fa	B	Re
	2	Ti	7	Sol	C	Mi
	3	Do	8	La	D	Fa
	4	Re	9	Ti	E	Sol
	5	Mi	A	Do	O,F	No Sound

No.2 Automatic Tunes

SWANEE RIVER



A KEYING IN THE TUNE

Press RESET. Now press the note key indicated followed by the INCR key. Key in the entire song—from address 00 to 08. If you make a mistake or have to start over, press RESET before beginning.

When you press the note key, the letter or number of that key is displayed. When you press INCR or RESET, any character might be displayed (it doesn't matter which one), so a question mark appears in the program next to INCR and RESET.

ADDRESS/KEY		binary LEDs	HEX. LED
	RESET		?
00	Fa	displays	6
	INCR		?
01	Mi	〃	5
	INCR	〃	?
02	Sol	displays	7
	INCR	〃	?
03	Re	〃	4
	INCR	〃	?
04	La	〃	1
	INCR	〃	?
05	Do	〃	3
	INCR	〃	?
06	La	〃	1
	INCR	〃	?
07		〃	F
	INCR	〃	?
08		〃	0
	INCR	〃	?

You have entered the notes ♩ ♪ ♪

This is the meaning of each character:

address	binary LEDs	data
00	●●●●●●●	6 ····· Tempo (speed)
01	●●●●●●☼	5 ····· Note
02	●●●●●☼●	7 ····· Length
03	●●●●●☼☼	4 ····· Note
04	●●●●☼●●	1 ····· Length
05	●●●●☼●☼	3 ····· Note
06	●●●●☼☼●	1 ····· Length
07	●●●●☼☼☼	F ····· End of tune
08	●●●☼●●●	0 ····· Repeat

Notice the lighting of the binary LEDs when you enter a key. Can you tell what it all means from what you've learned about the binary system? When you press the key entry for address 01, the binary LED for this address lights (●●●●●●☼).

0000001

When you follow this entry with the INCR key, the binary LEDs for the next address (02) light (●●●●●☼●)

0000010.

And so on ... This is a convenient way to check where you are in the program just in case you lose your place. For example, if a 3 shows on the HEX. LED and the binary LEDs show ●●●●☼●☼ (0000101), then you will be entering 3 into address 05 as soon as you press INCREMENT. The lighting of the binary LEDs tells you which address you are about to enter information into.

The last number/letter pressed before INCREMENT will be the number entered. You can correct a wrong entry by pressing the right key before you press INCREMENT. If you have already pressed INCREMENT, you will have to correct your mistake by another method described under "ERROR CORRECTION."

B CHECKING THE DATA

It is essential that you always check what has been keyed in, otherwise you may have some surprises! Press the keys shown in the next list and check that at each step the HEX. LED shows the same character as the list.

		binary LEDs	HEX. LED
RESET	displays ·····	●●●●●●●	6
INCREMENT	"	●●●●●●☼	5
INCREMENT	"	●●●●●☼●	7
INCREMENT	"	●●●●●☼☼	4
INCREMENT	"	●●●●☼●●	1
INCREMENT	"	●●●●☼●☼	3
INCREMENT	"	●●●●☼☼●	1
INCREMENT	"	●●●●☼☼☼	F
INCREMENT	"	●●●☼●●●	0

Each time the INCR key is pressed, the next address number is displayed in BINARY on the binary LEDs and the contents of that address are displayed on the HEX. LED.

* If any data is wrong, press RESET and key in the sequence again.

C STARTING AUTOMATIC PLAY

When you have checked that the data is correct start the program by pressing RESET, A, RUN in that order. You will hear three notes repeated over and over again. Can you work out how to stop it yourself?

Now the whole tune may be keyed in by following the chart below. Press the data number followed by INCREMENT. Notice which characters are notes and which are lengths.

address	binary LEDs	data	
00	●●●●●●○	6 – – –Tempo	
01	●●●●●○○	5 – – –Note	Mi
02	●●●●○●○	7 – – –Length	
03	●●●●○○○	4 – – –Note	Re
04	●●●○●●○	1 – – –Length	
05	●●●○●○○	3 – – –Note	Do
06	●●●○○●○	1 – – –Length	
07	●●●○○○○	5 – – –Note	Mi
08	●●○●●●○	1 – – –Length	
09	●●○●●○○	4 – – –Note	Re
0A	●●○●○●○	1 – – –Length	
0B	●●○●○○○	3 – – –Note	Do
0C	●●○○●●○	3 – – –Length	
0D	●●○○●○○	A – – –Note	Do
0E	●●○○○●○	3 – – –Length	
0F	●●○○○○○	8 – – –Note	La
10	●○●●●●○	1 – – –Length	
11	●○●●●○○	A – – –Note	Do
12	●○●●○●○	5 – – –Length	
13	●○●●○○○	7 – – –Note	Sol
14	●○●○●●○	7 – – –Length	
15	●○●○●○○	5 – – –Note	Mi
16	●○●○○●○	3 – – –Length	
17	●○●○○○○	3 – – –Note	Do
18	●○○●●●○	3 – – –Length	
19	●●○○●●○	4 – – –Note	Re
1A	●●○○●○●	F – – –Length	
1B	●●○○●○○	5 – – –Note	Mi
1C	●●○○○●●	7 – – –Length	
1D	●●○○○●○	4 – – –Note	Re
1E	●●○○○○●	1 – – –Length	
1F	●●○○○○○	3 – – –Note	Do
20	●○●●●●●	1 – – –Length	
21	●○●●●●○	5 – – –Note	Mi
22	●○●●●○●	1 – – –Length	
23	●○●●●○○	4 – – –Note	Re
24	●○●●○●●	1 – – –Length	
25	●○●●○●○	3 – – –Note	Do
26	●○●●○○●	3 – – –Length	
27	●○●●○○○	A – – –Note	Do
28	●○●○●●●	3 – – –Length	
29	●○●○●●○	8 – – –Note	La
2A	●○●○●○●	1 – – –Length	
2B	●○●○●○○	A – – –Note	Do
2C	●○●○○●●	5 – – –Length	
2D	●○●○○●○	7 – – –Note	Sol
2E	●○●○○○●	3 – – –Length	
2F	●○●○○○○	5 – – –Note	Mi
30	●○○●●●●	1 – – –Length	
31	●○○●●●○	3 – – –Note	Do
32	●○○●●○●	1 – – –Length	
33	●○○●●○○	4 – – –Note	Re
34	●○○●○●●	3 – – –Length	
35	●○○●○●○	4 – – –Note	Re
36	●○○●○○●	3 – – –Length	
37	●○○●○○○	3 – – –Note	Do
38	●○○○●●●	F – – –Length	
39	●○○○●●○	F – – –End of tune	
3A	●○○○●○●	0 – – –Repeat play	

D ERROR CORRECTION

If you have made an error near the beginning of the program, it is easiest to just press RESET and start over. But if you have already keyed in much of the program and you enter the wrong number into an address, you can correct just the address where the mistake is. For example, if you have entered a 5 in address 23, but it should be a 4, use the following procedure to put in the correct number:

RESET	should display	?
Ti	"	2 — hex number of address where incorrect figure is stored
Do	"	3 — hex number of address where incorrect figure is stored
ADDRESS SET	"	5
Re	"	4 — correct figure 4 written into address 23
INCREMENT	"	? — correct figure 4 written into address 23

You can make this correction while you are keying in the program (and then continue entering the rest of the program data) or you can correct your mistake after you have keyed in the entire program.

If you accidentally press INCR twice after entering a value, you will have entered the same value in two succeeding addresses. For example you may enter 5 into address 01 and 02 when you only wanted it in 01. So just skip address 02, return to it at the end of the program, and use the error correction method above to put the correct value (the one you skipped) into 02 (which will automatically erase the incorrect value—5).

You cannot delete any extra addresses you might accidentally enter, so be extra careful, especially in long programs!

E STOPPING & RESTARTING THE TUNE

To stop the tune, press RESET and hold it down until the music stops. To restart, press A, RUN as before.

F CHANGING THE SPEED

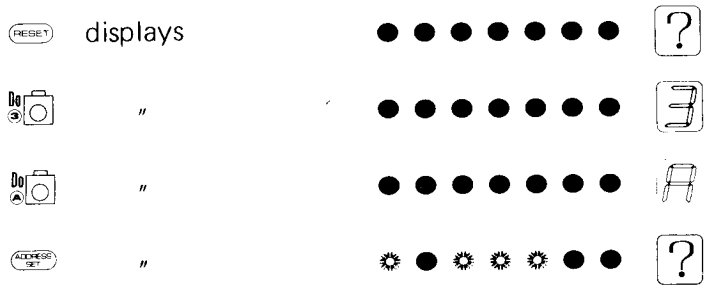
The tune may be played faster or slower by changing address 00. To make it slower, replace 6 in address 00 with 7, 8, 9, A, B, C, D, E or F (F is slowest). To make it faster, replace 6 with 5, 4, 3, 2, 1 or 0 (0 is fastest).

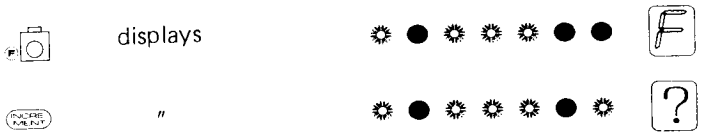
Try changing address 00 to 3 by pressing RESET, 3, INCR. When you run the program now it will sound faster.

Try putting other numbers into address 00 to get a whole range of different speeds.

G STOPPING AFTER ONE PERFORMANCE

To do this change the last character of the tune to F. This is at address 3A; at first you put 0 there but now change it as follows:





Then start the program by pressing RESET, A, RUN. The tune will play once only this time.

Note Codes

TEMPO – codes 0-F (fast) 0 (----------) F (slow)
A REST – code 0
LENGTH OF NOTES/RESTS – see codes in next table
END OF TUNE – F followed by 0 to repeat
END OF TUNE – F followed by F for a single play

H LENGTHS OF NOTES AND RESTS

In the following table, the second column shows the length of the note and the third column shows the code to use to control the length:

note/rest	length in 1/16	code
1/16 note 1/16 rest	1	0
1/8 note 1/8 rest	2	1
dotted 1/8 note	3	2
1/4 note 1/4 rest	4	3
dotted 1/4 note	6	5
1/2 note 1/2 rest	8	7
dotted 1/2 note (𝅗𝅥.)	12	B
whole note (O) whole rest (▬)	16	F

I WRITING IN YOUR OWN TUNE

You should now be able to code a tune of your own for this program. REMEMBER THESE RULES:

1) The tune must not go further than address 5F.

2) The tempo mark goes in address 00.

3) Codes for notes and rests go in addresses with odd numbers.

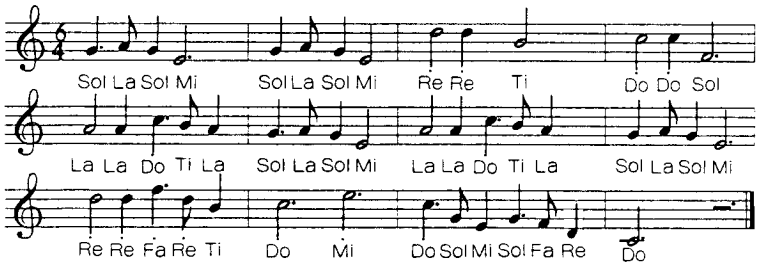
4) Codes for the length of the previous note or rest go in addresses with even numbers.

5) End the tune with F0 (repeat) or FF (stop after one play). You must leave 2 addresses free at the end for these codes.

On the next pages are more tunes. This is followed by a list of all the codes to be keyed in to make them play. Before you look at the list of codes, write down on a separate piece of paper what you think the codes should be, just by looking at the tune.

It will be very good practice for you.

SILENT NIGHT



Data for this tune

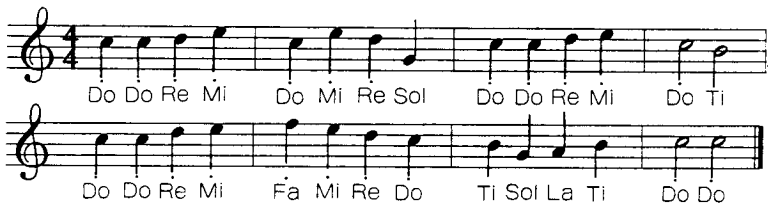
address	binary LEDs	data	
00	●●●●●●●	RESET A – – –Tempo	
01	●●●●●●○	7 – – –Note	Sol
02	●●●●●○●	5 – – –Length	
03	●●●●○○○	8 – – –Note	La
04	●●●○●●●	1 – – –Length	
05	●●●○●○○	7 – – –Note	Sol
06	●●●○○○●	3 – – –Length	
07	●●●○○○○	5 – – –Note	Mi
08	●●○●●●●	B – – –Length	
09	●●○●●○○	7 – – –Note	Sol
0A	●●○●○●●	5 – – –Length	
0B	●●○●○○○	8 – – –Note	La
0C	●●○○●●●	1 – – –Length	
0D	●●○○●○○	7 – – –Note	Sol
0E	●●○○○○●	3 – – –Length	
0F	●●○○○○○	5 – – –Note	Mi
10	●○●●●●●	B – – –Length	
11	●●○●●●○	B – – –Note	Re
12	●●○●●○●	7 – – –Length	
13	●●○●●○○	B – – –Note	Re
14	●●○●○●●	3 – – –Length	
15	●●○●○●○	9 – – –Note	Ti
16	●●○●○○●	B – – –Length	
17	●●○●○○○	A – – –Note	Do
18	●●○○●●●	7 – – –Length	
19	●●○○●●○	A – – –Note	Do
1A	●●○○●○●	3 – – –Length	
1B	●●○○●○○	7 – – –Note	Sol
1C	●●○○○●●	B – – –Length	
1D	●●○○○●○	8 – – –Note	La
1E	●●○○○○●	7 – – –Length	
1F	●●○○○○○	8 – – –Note	La
20	●○●●●●●	3 – – –Length	
21	●○●●●●○	A – – –Note	Do
22	●○●●●○●	5 – – –Length	
23	●○●●●○○	9 – – –Note	Ti
24	●○●●○●●	1 – – –Length	
25	●○●●○●○	8 – – –Note	La
26	●○●●○○●	3 – – –Length	
27	●○●●○○○	7 – – –Note	Sol
28	●○●○●●●	5 – – –Length	
29	●○●○●●○	8 – – –Note	La
2A	●○●○●○●	1 – – –Length	
2B	●○●○●○○	7 – – –Note	Sol
2C	●○●○○●●	3 – – –Length	
2D	●○●○○●○	5 – – –Note	Mi
2E	●○●○○○●	B – – –Length	
2F	●○●○○○○	8 – – –Note	La
30	●○○●●●●	7 – – –Length	

address	binary LEDs	data	
31	●○○●●●○	8 – – –Note 3 – – –Length	La
32	●○○●●○●		
33	●○○●●○○	A – – –Note 5 – – –Length	Do
34	●○○●○●●		
35	●○○●○●○	9 – – –Note 1 – – –Length	Ti
36	●○○●○○●		
37	●○○●○○○	8 – – –Note 3 – – –Length	La
38	●○○○●●●		
39	●○○○●●○	7 – – –Note 5 – – –Length	Sol
3A	●○○○●○●		
3B	●○○○●○○	8 – – –Note 1 – – –Length	La
3C	●○○○○●●		
3D	●○○○○●○	7 – – –Note 3 – – –Length	Sol
3E	●○○○○○●		
3F	●○○○○○○	5 – – –Note B – – –Length	Mi
40	○●●●●●●		
41	○●●●●●○	B – – –Note 7 – – –Length	Re
42	○●●●●○●		
43	○●●●●○○	B – – –Note 3 – – –Length	Re
44	○●●●○●●		
45	○●●●○●○	D – – –Note 5 – – –Length	Fa
46	○●●●○○●		
47	○●●●○○○	B – – –Note 1 – – –Length	Re
48	○●●○●●●		
49	○●●○●●○	9 – – –Note 3 – – –Length	Ti
4A	○●●○●○●		
4B	○●●○●○○	A – – –Note B – – –Length	Do
4C	○●●○○●●		
4D	○●●○○●○	C – – –Note B – – –Length	Mi
4E	○●●○○○●		
4F	○●●○○○○	A – – –Note 5 – – –Length	Do
50	○●○●●●●		
51	○●○●●●○	7 – – –Note 1 – – –Length	Sol
52	○●○●●○●		
53	○●○●●○○	5 – – –Note 3 – – –Length	Mi
54	○●○●○●●		
55	○●○●○●○	7 – – –Note 5 – – –Length	Sol
56	○●○●○○●		
57	○●○●○○○	6 – – –Note 1 – – –Length	Fa
58	○●○○●●●		
59	○●○○●●○	4 – – –Note 3 – – –Length	Re
5A	○●○○●○●		
5B	○●○○●○○	3 – – –Note B – – –Length	Do
5C	○●○○○●●		
5D	○●○○○●○	F – – –End of tune	
5E	○●○○○○●	0 – – –Repeat play	

* Addresses 5D & 5E above show the codes needed to repeat the tune.

Did you get it right? Now key in all the data exactly as you did for the first tune.

YANKEE DOODLE



Data for this tune

address	binary LEDs	data
00	●●●●●●●	RESET 3– – –Tempo
01	●●●●●●○	A– – –Note 3– – –Length → Do
02	●●●●●○●	A– – –Note 3– – –Length → Do
03	●●●●●○○	B– – –Note 3– – –Length → Re
04	●●●●○●●	C– – –Note 3– – –Length → Mi
05	●●●●○●○	A– – –Note 3– – –Length → Do
06	●●●●○○●	B– – –Note 3– – –Length → Re
07	●●●●○○○	C– – –Note 3– – –Length → Mi
08	●●●○●●●	A– – –Note 3– – –Length → Do
09	●●●○●●○	A– – –Note 3– – –Length → Do
0A	●●●○●○●	C– – –Note 3– – –Length → Mi
0B	●●●○●○○	B– – –Note 3– – –Length → Re
0C	●●●○○●●	A– – –Note 3– – –Length → Do
0D	●●●○○●○	9– – –Note 3– – –Length → Ti
0E	●●●○○○●	7– – –Note 3– – –Length → Sol
0F	●●●○○○○	A– – –Note 3– – –Length → Do
10	●●○●●●●	A– – –Note 3– – –Length → Do
11	●●○●●●○	A– – –Note 3– – –Length → Do
12	●●○●●○●	A– – –Note 3– – –Length → Do
13	●●○●●○○	B– – –Note 3– – –Length → Re
14	●●○●○●●	C– – –Note 3– – –Length → Mi
15	●●○●○●○	D– – –Note 3– – –Length → Fa
16	●●○●○○●	C– – –Note 3– – –Length → Mi
17	●●○●○○○	B– – –Note 3– – –Length → Re
18	●●○○●●●	A– – –Note 3– – –Length → Do
19	●●○○●●○	9– – –Note 3– – –Length → Ti
1A	●●○○●○●	7– – –Note 3– – –Length → Sol
1B	●●○○●○○	A– – –Note 7– – –Length → Do
1C	●●○○○●●	9– – –Note 7– – –Length → Ti
1D	●●○○○●○	A– – –Note 3– – –Length → Do
1E	●●○○○○●	A– – –Note 3– – –Length → Do
1F	●●○○○○○	A– – –Note 3– – –Length → Do
20	●○●●●●●	B– – –Note 3– – –Length → Re
21	●○●●●●○	C– – –Note 3– – –Length → Mi
22	●○●●●○●	D– – –Note 3– – –Length → Fa
23	●○●●●○○	C– – –Note 3– – –Length → Mi
24	●○●●○●●	B– – –Note 3– – –Length → Re
25	●○●●○●○	A– – –Note 3– – –Length → Do
26	●○●●○○●	9– – –Note 3– – –Length → Ti
27	●○●●○○○	7– – –Note 3– – –Length → Sol
28	●○●○●●●	
29	●○●○●●○	
2A	●○●○●○●	
2B	●○●○●○○	
2C	●○●○○●●	
2D	●○●○○●○	
2E	●○●○○○●	
2F	●○●○○○○	
30	●○○●●●●	

address	binary LEDs	data	
31	●○○●●●○	8– – –Note	La
32	●○○●●○●	3– – –Length	
33	●○○●●○○	9– – –Note	Ti
34	●○○●○●●	3– – –Length	
35	●○○●○●○	A– – –Note	Do
36	●○○●○○●	7– – –Length	
37	●○○●○○○	A– – –Note	Do
38	●○○○●●●	7– – –Length	
39	●○○○●●○	F– – –End of tune	
3A	●○○○●○●	F– – –Single play	

* Addresses 39 & 3A above show the codes needed to play once only.

Did you get it right? Now key in all the data exactly as you did for the first tune.

No.3 Musical Guessing Game

In this game, the micro plays two or more musical notes and you have to repeat those notes by pressing the keys corresponding to those notes.
Before you try this game, make sure that you understand the previous games 1 and 2, and that you recognize the notes.

A STARTING THE GAME

Press RESET.
Press B, then RUN.
The game has now started.
The micro will now play two notes; the first will always be Do, which is the sound that you heard in game 1 when you pressed 3.

B THE FIRST SOUND IS ALWAYS DO

If you press 3, you will again hear Do.
Now you must press the key (1-9, A-E) that you think will produce the same sound as the second note you heard.
If you press the right one, you will hear the second sound repeated and another sound added to it. And you score 1 point.
If you are wrong, the error signal will sound and you score nothing.
Press RUN and try again.

C IF YOU GET THE SECOND SOUND RIGHT .....

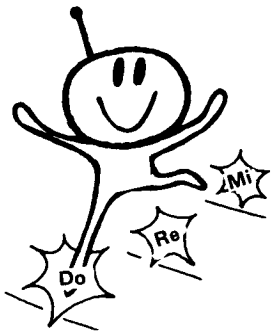
The micro will repeat the Do and the second sound and then play a third sound. You must now press the Do key, the key for the second sound, and then guess the key to press for the third sound.
If you are right again, your score goes up to 2 and the micro adds another note.
If you are wrong, your total score (1) is displayed on the HEX. LED and you will hear the error sound.

D A NEW NOTE IS ADDED EACH TIME YOU ARE RIGHT

The better you are at guessing the notes the better your memory will have to be. The micro plays all the notes you have already guessed, including the Do at the start, plus one more note. Each time that you remember all of the previous notes and the new one correctly, your score increases by 1.
If you get 10 right in a row, you will hear a special end-of-game sound and A will be displayed on the HEX. LED.
"A" ??? What does A stand for when you are counting in hex? Yes, it stands for decimal 10 – your winning score.

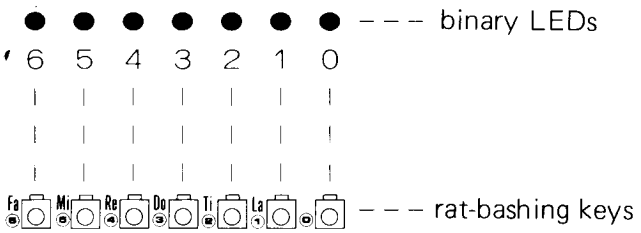
HOW DID YOU DO?

Keep practicing the games and trying to understand the principles of the micro.

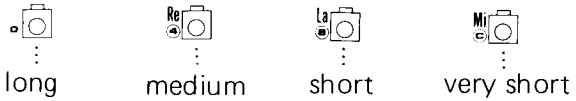


No.4 "Rat Bashing"

In this game, the seven binary LEDs are "rat holes." When one of the LEDs lights up it means that a "rat" is poking up his head. You have to "hit" him by pressing the number key below the LED corresponding to that "hole."
A total of 10 rats will appear during one game. At the end of the game the HEX. LED will show your score out of 10.

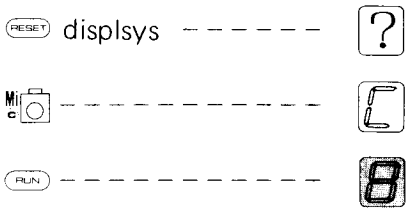


You can control the interval between the appearance of rats by pressing 0, 4, 8 or C key after RUN.



A STARTING THE GAME

Press RESET, C, RUN.

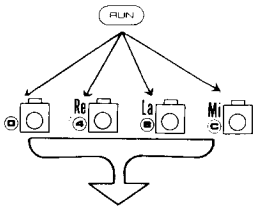


Now press 0, 4, 8, or C.

A rat appears at one of the binary LEDs. You must press the number key corresponding to the number beneath the LED to score a point. A "beep" sounds if you are successful. The game ends after 10 rats have appeared. You will hear a special sound and your score will be displayed on the HEX. LED. If you score 10, A will be displayed.

B TO PLAY AGAIN

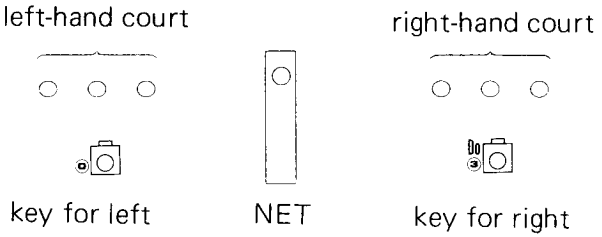
To play again press RUN, then a speed code (0, 4, 8, C) and the game will begin again.



Pressing one of these keys after RUN starts the game.

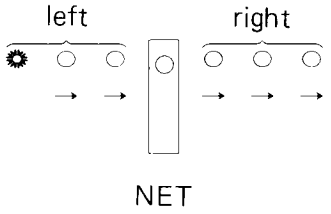
No.5 Tennis Game

In this game, a lighted binary LED represents the tennis ball and two number keys are the rackets. The game is for two people.



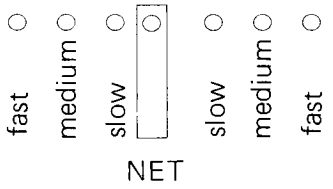
The player in the left-hand court uses 0 as their racket. The player in the right-hand court uses 3. To hit the ball, press your key, but do not hold it down.

When the game starts, the left-hand LED will be lit meaning the left-hand player serves first.



Pressing 0 now causes the "ball" to move towards the right-hand court. The player to the right presses 3 to hit it back. Each time a player makes a successful shot, there is a beep. When a mistake is made, you hear the error sound.

The speed of the ball depends upon where it is when it is hit.



The opponent scores a point in the following cases:

- the key is pressed when the ball is not in the court
- the key is kept pressed down
- the key is pressed too late to stop the ball from hitting the back of the court

The ball stops at the end of the court in which a mistake occurred. The player with the ball in his court now serves. Not like normal tennis, is it!

When one of the players has scored 7 points the game ends with a special sound. The two "courts" are used to display each player's score.

Left has scored 4+2+1 = 7
Right has scored only 2+1 = 3
So the left-hand player has won the game 7:3.

A STARTING THE GAME

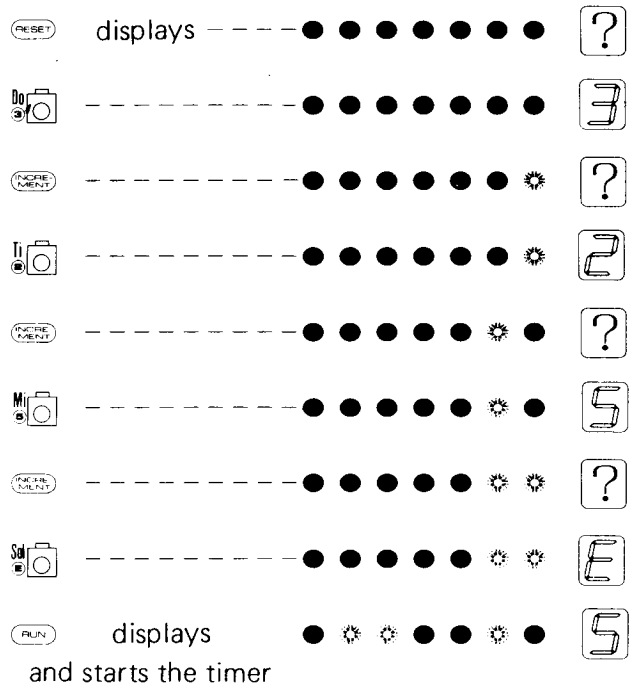
Press RESET, D, RUN in that order.

B TO START AGAIN, press RUN.

No.6 Timer

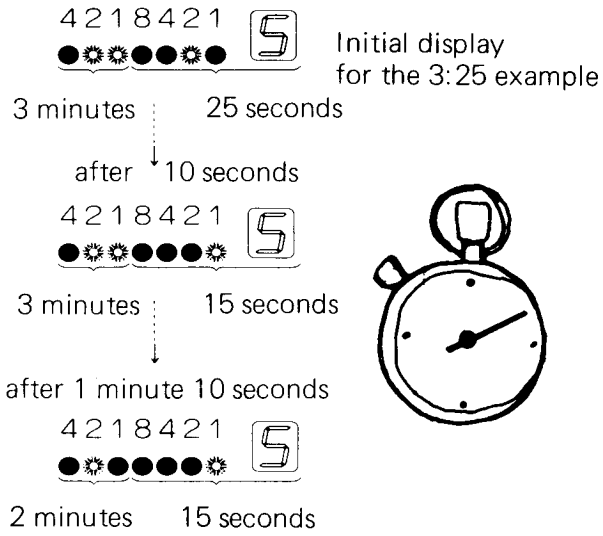
This is a program to instruct the micro to make an alarm sound after a chosen period of time up to 7 minutes and 59 seconds.

To set the alarm for 3 minutes and 25 seconds.:



When RUN is pressed the timer starts to count down with one tick sounding per second.

A HOW TO READ THE DISPLAY



The LEDs show the time remaining. When no LEDs light,

there is no time left and you hear the end sound.

B TO RESET THE TIMER, enter a new time following the example for 3 minutes and 25 seconds.

No.7 Morse Code

You can use computer codes to represent Morse codes and the micro will play back your message.

INTERNATIONAL MORSE CODE:

A	·–	1	·————		
B	–···	2	··–––		
C	–·–·	3	···––		
D	–··	4	····–		
E	·	5	·····		
F	··–·	6	–····		
G	––·	7	––···		
H	····	8	–––··		
I	··	9	––––·		
J	·–––	0	–––––		
K	–·–	Attention		·–·–·–·	
L	·–··	Query (?)		··––··	
M	––	Comma (,)		––··––	
N	–·	Period (.)		·–·–·–	
O	–––	Error		·····	
P	·––·	Received		·–·	
Q	––·–	End of message		·–·–·	
R	·–·				
S	···				
T	–				
U	··–				
V	···–				
W	·––				
X	–··–				
Y	–·––				
Z	––··				

Computer Codes for Morse					
	code		code		code
··	0	character gap & ·	8	7-unit gap	A
·–	1	〃 〃 –	9	10-unit gap	B
–·	4	· and word gap	3	end	E
––	5	– 〃 〃	7	repeat	F
· & character gap	2	word gap and ·	C		
– 〃 〃	6	〃 〃 –	D		

When Morse code is being transmitted a dot takes one unit of time. Between each character there should be a gap of three units. Between each word there should be a gap of 7 units.

Take the word TOM as an example.
This is converted as follows:

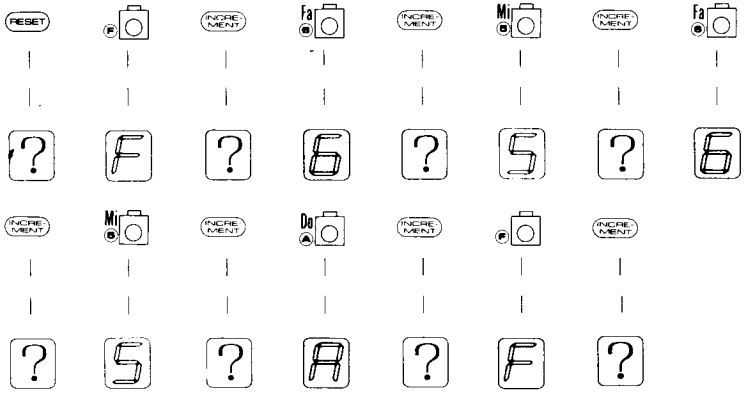
T O M

– (char·gap) ––– (char·gap) –– (word gap) = Morse code

6 5 6 5 A = Computer code

A KEYING IN THE DATA

Start off with RESET, F, INCR, then continue as shown in the following diagram:



B CHECKING THE DATA

Press RESET, INCR, INCR, ... After each key stroke check the HEX. LED to ensure that it contains the following data:

If the HEX. LED does not show these characters in order you will need to make some corrections.

Start the program by pressing RESET, F, RUN. You will then hear the Morse code for TOM.

The program can be stopped by pressing RESET. Restart with F, RUN.

The speed of transmission can be changed by using different codes: 0 - - - - - - - - - 7, 8 - - - - - - - - - F

← faster slower →

The speed code is the first character to be entered when keying in the message.

To change the speed of transmission, the contents of address 00 must be changed.

To change the speed to 5, press RESET. The HEX. LED should now display F. Change it to 5 by pressing 5, INCR. Then press RESET, F, RUN to hear the Morse code being transmitted at a faster speed.

These games and experiments have familiarized you with your computer.

All the games are RUN under the control of PROGRAMS. These are sets of commands which have been precisely put together and stored in the computer. The rest of this manual will show you HOW to put commands together. By the end you will know how to turn the Microcomputer Trainer into an electronic organ or a rat-trap game or ..... you may have lots of your own ideas.

The commands that send messages to your microcomputer consist of combinations of letters. They are called machine codes. A chart with these codes is shown in the bottom left corner of your computer. We will divide these commands into several groups and explain them in the next few sections.

Group 1 Commands

Let's begin with a group of commands that control the A and B registers. Remember, a register is a temporary memory address that stores one character at a time.

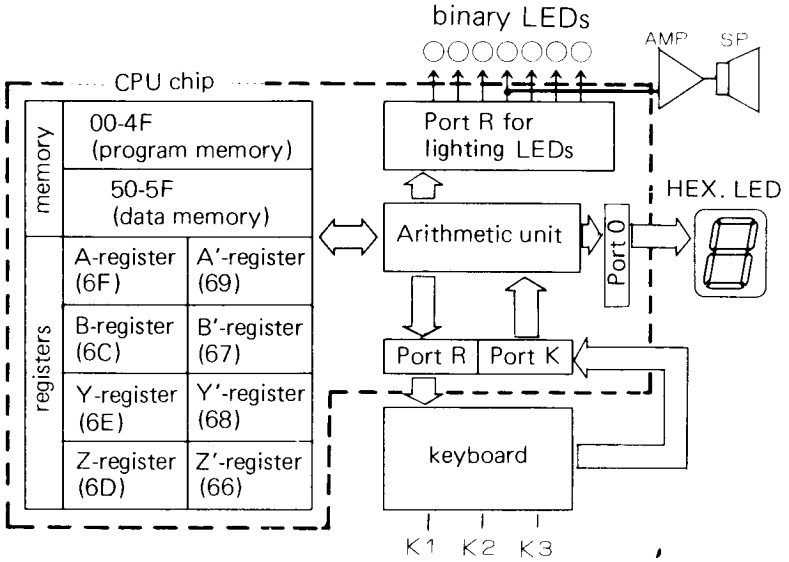
The commands we will discuss are: KA, AO, CH, TIA, AIA, and JUMP.

The A register (Ar) is the most important register. It is used to light the HEX. LED, to receive data from the keyboard, and as a work area for calculations. The B register (Br) helps Ar by temporarily storing the contents of Ar.

The diagram below shows the organization of the Micro-computer Trainer.

A "port" is any connection between the micro and its links with the outside world, for example, the LEDs.

Perhaps the diagram will not mean too much to you now. Refer back to it from time to time.



No.8 Use of TIA and AO to turn on the HEX. LED

TIA and AO

The TIA command stands for "Transfer Into A." It moves data into the A register.

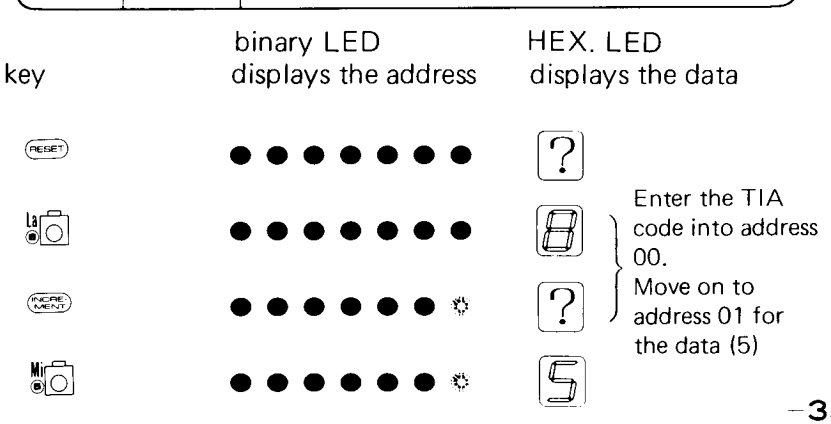
The AO command displays the contents of Ar on the HEX. LED

The diagram below shows "5" is moved to the A-register by the TIA command.

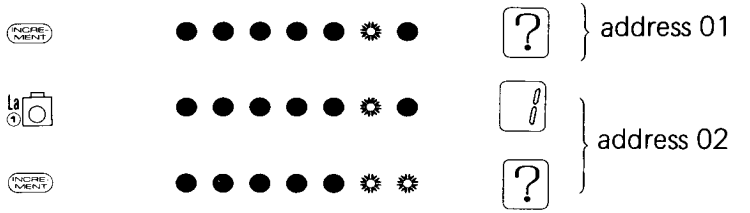
A) Key in the following program to see how these commands work:

PROGRAM EXAMPLE – to display 5 on the HEX. LED

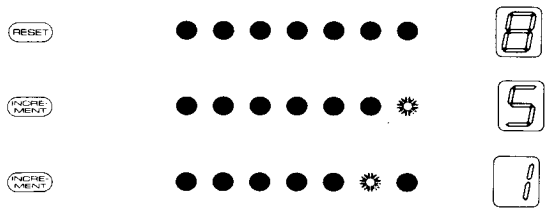
address	command	machine code	
00	TIA	8	Move 5 into Ar (the A-register)
01	⟨5⟩	5	Contents of Ar is output to port 0 to turn on HEX. LED.
02	AO	1	



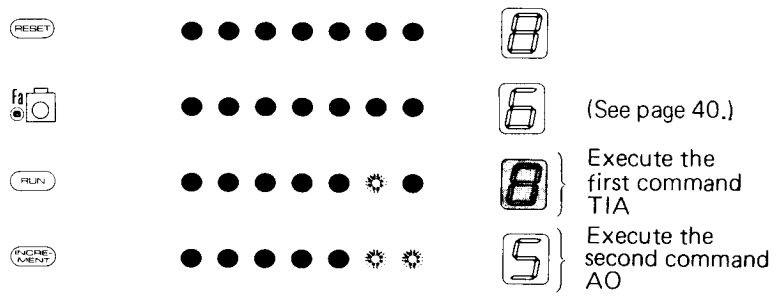
key



B) After writing in the codes, read them out and check them as follows:



C) Execute the commands like this:



When you pressed the INCR key, 5 was displayed on the HEX. LED – so it worked!

The HEX. LED is shown like this when nothing is displayed.

The HEX. LED is shown like this when anything can be displayed.

D) To change the 5 command at address 01 to the 2 command, press the following keys:

Now execute the new program:

Now  is displayed when you press INCR.

*** TIA (T ransfer Into A)**

The TIA command takes up two characters of machine code. When it is executed, the second character is moved to Ar.

command ····→ T I A
data ····→ ⟨5⟩

Contents of Ar

0	1	0	1
8	4	2	1

5

*** AO (move contents of Ar to HEX. LED port 0)**

AO displays the contents of Ar on the HEX. LED. Programs often contain ROUTINES like the one you have just tried, using TIA to move a character to Ar and AO to display it.

Contents of Ar

0	1	0	1

The contents of Ar are not changed by this command.

No.9 Use of CH and JUMP to display 0 & 1 alternately

CH and JUMP

When you need to display two numbers in turn on the HEX. LED, a single register is not enough. In the following program you will use CH which means to exCHange the contents of the A and B registers or to move what's in A to B to make room in A for another number.

With the JUMP command, you can interrupt the sequence of commands and "jump" back (or forward) to another address.

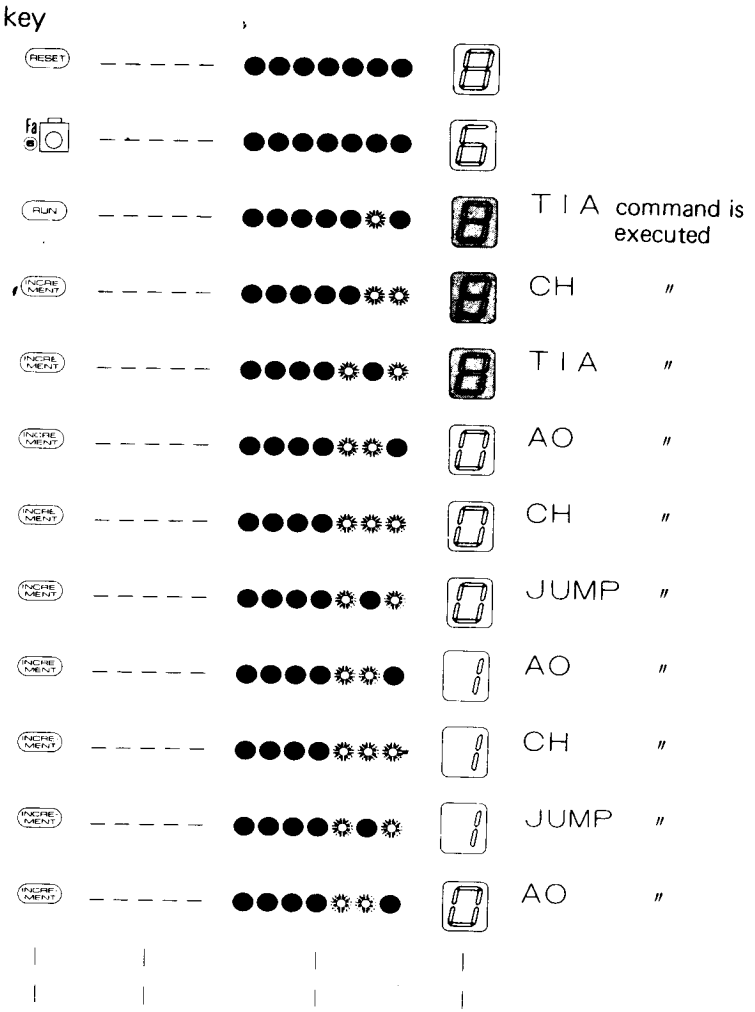
PROGRAM EXAMPLE – DISPLAY 0 and 1 in turn

address	command	machine code	
00	T I A	8	Move 1 to Ar
01	⟨1⟩	1	
02	CH	2	Move the 1 in Ar to Br
03	T I A	8	Move 0 to Ar
04	⟨0⟩	0	
05	AO	1	····· Display contents of Ar
06	CH	2	····· Swap contents of Ar/Br
07	JUMP	F	Return to address 05 to repeat the LOOP
08	⟨0⟩	0	
09	⟨5⟩	5	

A) Enter the above machine codes into addresses 00-09.

B) Check the program and correct it if necessary.

C) Run the program as follows:



And so on ..... each time you press INCR three times the CH, JUMP and AO commands are repeated and the number displayed changes from 0 to 1 to 0 .....

Try to understand clearly what the CH and JUMP commands are doing. CH is swapping 0 and 1 back and forth between Ar and Br. JUMP is forever making the program go back to the AO command at address 05.

- **CH (exCHange a and b registers)**

The CH command exchanges the contents of Ar with those of Br. Then the contents of Ar are moved to Br and the contents of Br are moved to Ar.

▲ BEFORE EXECUTING CH

Ar is assumed to contain 1 (put there by TIA)

Contents of Br are unknown — no character put into Br.

Ar	Br
1	unknown

▲ AFTER EXECUTING CH

Ar	Br
unknown	1

Now 0 can be put into Ar using TIA.

0	1

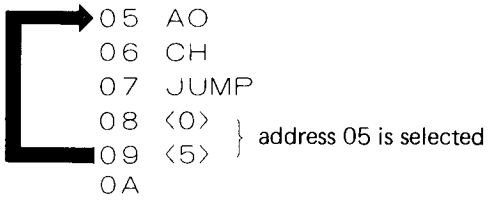
▲ IF CH IS EXECUTED AGAIN the result will be:

Ar	Br
1	0

The contents of Ar and Br have been exchanged once again.

- **JUMP**

When the micro has executed most commands it goes on to execute the command in the next memory address. JUMP is different – it interrupts the sequence of commands by telling the micro which address to JUMP to next.
For example:

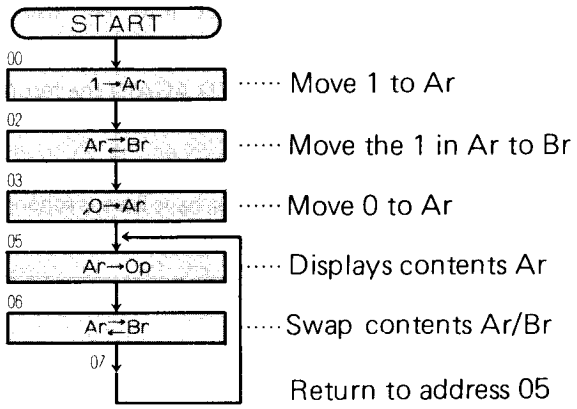


The JUMP command occupies three addresses. The second and third contain the number of the address to be JUMPed to. The above program will never proceed to 0A because JUMP always returns it to 05.
The heavy black arrow above indicates a program LOOP.

A program can be presented in many different ways – as a group of machine codes, in the form of a FLOWCHART or by means of a collection of descriptive statements. Each method is illustrated for program 9.
By following the arrows in a flowchart you can often see more clearly what is going on.

address	command
00	TIA
01	⟨1⟩
02	CH
03	TIA
04	⟨0⟩
05	AO
06	CH
07	JUMP
08	⟨0⟩
09	⟨5⟩

FLOWCHART



No.10 Use of KA to transfer data from keyboard to display

KA is the command used to transfer the key-value (O-F) from the keyboard to the A-register (Ar).

In this program, when a key is pressed the corresponding character is moved to Ar. As long as no key is pressed, the program JUMPS back to address 00. When a key is pressed, the JUMP command is not executed and the program goes through to address 04, where Ar is displayed on the HEX. LED.

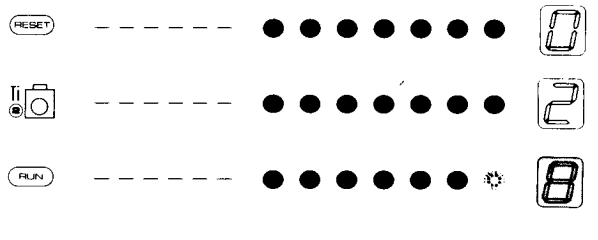
PROGRAM

address	command	machine code	
00	KA	O	If no key is struck at (address) 00, the program JUMPS back to 00 to check again.
01	JUMP	F	
02	⟨0⟩	O	
03	⟨0⟩	O	
04	AO	1	Contents of Ar on HEX. LED
05	JUMP	F	The program JUMPS back to 00 to wait for another key to be pressed.
06	⟨0⟩	O	
07	⟨0⟩	O	

A) Key in the program and check it.

B) Execute the program you have entered as follows:

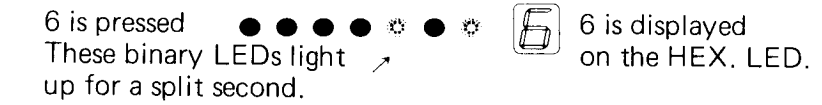
key



When the RUN key is pressed, the binary LEDs have the value 1, which represents the address 01. The program does not move to 04 until a key is pressed.

When a key is pressed, the binary LEDs flash on and off very quickly. You will probably not see more than a flicker before they settle back to 1 (address 01).

Try pressing 6 and see what happens.

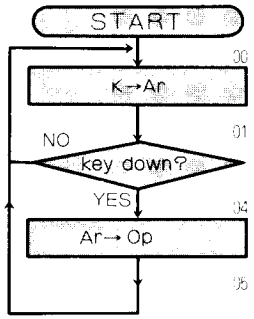


When a key is pressed, the program moves on to the next command, which is at address 04, instead of returning to 00, and the number corresponding to the key pressed is displayed. When the number has been displayed, the program returns to 00 to wait for another key to be pressed.
The binary LEDs then point to 01 again.

command symbols

KA
JUMP
〈0〉
〈0〉
AO
JUMP
〈0〉
〈0〉

FLOWCHART



key value is moved to Ar.

If no key input, program returns to KA at 00.
If key input: value of Ar is displayed.

Program returns to beginning.

- **KA**

The KA command moves the value of a number key (0-F) into Ar. If a key is not pressed when the program encounters the KA, a FLAG is set to 1. This tells the program to execute the next step which is the JUMP command.

If a key is pressed when the program executes the KA command, the FLAG is set to 0, meaning to ignore the JUMP command and look for the next command.

- **JUMP – HOW IT WORKS (2)**

When KA is executed, the FLAG can be set to either 1 or 0. If it is 1, a JUMP command coming after it will be executed. If it is 0, a JUMP command coming after it will be ignored. In program 10, as long as no key is pressed the FLAG will be 1 and the program will return to 00. When a key is pressed the FLAG becomes 0 and the program passes the JUMP as if it was not there.

The JUMP at 05 is always executed because AO always sets the FLAG to 1. KA, M+, M–, AIA, AIY, CIA, CIY, CAL, SIFT are all commands which set the FLAG to 0 or 1 depending on the circumstances.

- **STEP MODE & RUN MODE – HOW THEY WORK (1)**

In programs 8 and 9, programs were executed by pressing RESET, 6, RUN, INCR, INCR.... This is called STEP MODE. In program 10 you executed the program by pressing RESET, 2, RUN. This is called RUN MODE.

In STEP MODE, when RUN is pressed the first command is executed and each time INCR is pressed another command is executed. In RUN MODE when you press RUN all the commands in the program are rapidly executed one after the other (unless the program stops at KA).

STEP MODE is selected by entering 6, RUN; RUN MODE is selected by entering 2, RUN. Which one you will be instructed to use depends upon the program.

There will be more information given about MODES after the next program.

No.11 Use of AIA to add numbers together

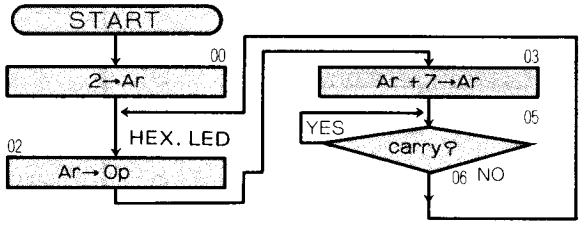
You have already learned how important the A-register is. You are now going to use the AIA command to add a number to Ar and store the answer in Ar.

> • **AIA COMMAND (Add Into Ar)**
> AIA takes up two lines of code; it is a TWO-WORD command. The second line contains a number which is added to Ar and the answer is stored in Ar.

PROGRAM

address	command	machine code	
00	TIA	8	} move 2 to Ar.
01	⟨2⟩	2	
02	AO	1	... contents (2) of Ar displayed
03	AIA	9	} 7 is added to Ar and the answer moved to Ar.
04	⟨7⟩	7	
05	JUMP	F	} If there is a CARRY (answer more than F) program stops.
06	⟨0⟩	0	
07	⟨5⟩	5	
08	JUMP	F	} If there is no CARRY (answer between 0 & F) program returns to 02 to repeat addition.
09	⟨0⟩	0	
0A	⟨2⟩	2	

FLOWCHART



A) Key in the progaram.

B) Check the program and correct it if necessary.

C) Run the program.

Execute the program by pressing the following keys:

KEY

RESET	● ● ● ● ● ● ●	8
Fa	● ● ● ● ● ● ●	6
RUN	● ● ● ● ● ○ ●	8 TIA is executed.
INCREMENT	● ● ● ● ● ○ ○	2 AO is executed.
INCREMENT	● ● ● ● ○ ● ○	2 AIA is executed.
INCREMENT	● ● ● ○ ● ● ●	2 JUMP not executed.
INCREMENT	● ● ● ● ● ○ ●	2 JUMP is executed.
INCREMENT	● ● ● ● ● ○ ○	9 AO is executed.
INCREMENT	● ● ● ● ○ ● ○	9 AIA is executed.
INCREMENT	● ● ● ● ○ ● ○	9 JUMP is executed.
INCREMENT	● ● ● ● ○ ● ○	9 JUMP is executed.

In this program the FLAG will be set to 0 or 1 by AIA, depending upon whether there is a CARRY. Now we will look again at HEX addition.

At 03, we add 2 + 7 = 9 in Ar:

Ar	AIA data	Ar
2	+7	→ 9

no CARRY; FLAG = 0

But next time (at 03) we added 7 & 9:

```
  9
+ 7
----
 10
```

..... and there is a CARRY (so FLAG =1) because in hex counting you cannot exceed 15 (F) without carrying over. So the program ends.

- **STEP MODE AND RUN MODE (2)**

STEP and RUN MODE have been explained under program 10.

a) In STEP MODE there are two methods of execution:

1) Addresses are displayed and key strokes sound. Press 6, RUN. The first command is executed and the binary LEDs display the address of the next command. Press INCR. The next command is executed and the binary LEDs display the address of the next command.

Go on pressing INCR and you can follow the course of the program right through to the end (if it has an end). Each time you press RUN or INCR you will hear a sound.

2) Addresses are not displayed and there is no sound when keys are pressed. Press 5, RUN. The first command is executed and the program stops. Each time you press INCR another command is executed. Addresses are not displayed on the binary LEDs so that these can be used to display other data.

b) RUN MODE also has two methods of execution.

1) Addresses are displayed and RUN key makes a sound. Press 2, RUN. Commands are executed in rapid succession. This is how you run program 10.

In program 10 a key might not be pressed between addresses 00 and 01, so the binary LEDs show 00 and 01 in turn. But the change from one to another is so rapid that the display seems to be stationary at 01.

A sound is generated when RUN is pressed but not when a number key is pressed.

2) Addresses are not displayed and no key-stroke sound is generated. Press 1, RUN. Commands will be executed without interruption but no address will be displayed.

Review this section carefully and make sure that you understand the work up to this point.

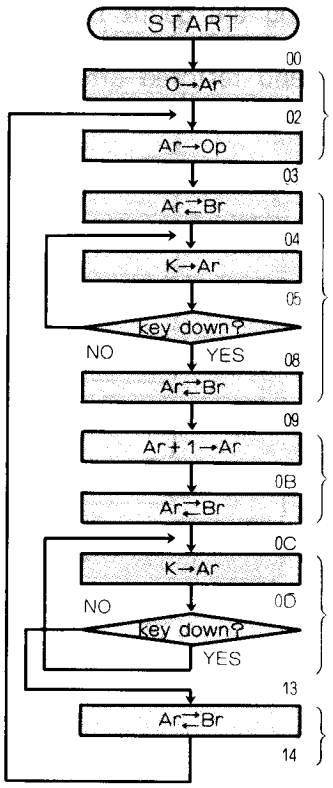
No.12 Display Hex Numbers in ascending order

Programs 8-11 showed how the first group of commands is used (KA, AO, CH, TIA, AIA, JUMP). The next few programs are going to show you how to apply them to the kind of tasks that computers are doing all the time, such as counting – in hex, binary, and decimal.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨1⟩	1
0B	CH	2
0C	KA	0
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨3⟩	3
10	JUMP	F
11	⟨0⟩	0
12	⟨C⟩	C
13	CH	2
14	JUMP	F
15	⟨0⟩	0
16	⟨2⟩	2

FLOWCHART



NOTE:
The series of two-digit numbers at the right hand side of the flowchart correspond to the addresses in the program.

1)
0 is displayed at address 02. Then each time the program returns to 02 by means of the JUMP command at address 14 the number displayed increases by 1 because of the AIA command at 09.

2)
The contents of Ar are stored in Br because any value entered at the KA command is going to replace whatever was in Ar before.
If no key is pressed, the program JUMPS back to 04 for another "look". If a key is pressed the value stored in Ar is swapped with Br.

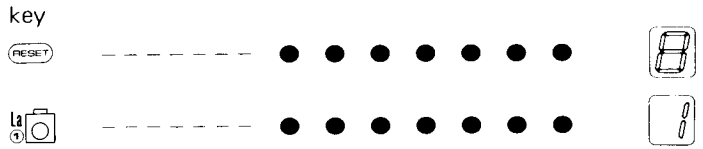
3)
The total so far, which is now held in Ar once again, is increased by 1 and the result is stored in Br again and on and on.....

4)
As long as the key pressed at the previous KA command is held down, this bit of the program keeps returning for another look. Compare the action taken at address 0D-12 with the action at 05-07.

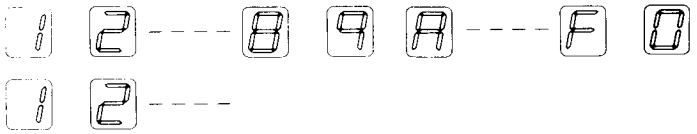
5)
When the key has been released, the accumulated total is restored to Ar and the program returns to 02 to display the contents of Ar.

A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.



When RUN is pressed, 0 will be displayed; then each time any of the keys 0-F is pressed and released the display will be increased by 1, as shown below:



This program may seem a bit difficult if this is all new to you. But keep at it!

By the time that you have reached program no. 21, the way that this program (no.12) works will seem quite familiar and straightforward. What is more, you will be able to use parts of it in your own programs later on.

ADDITIONAL NOTES ABOUT PROGRAM 12

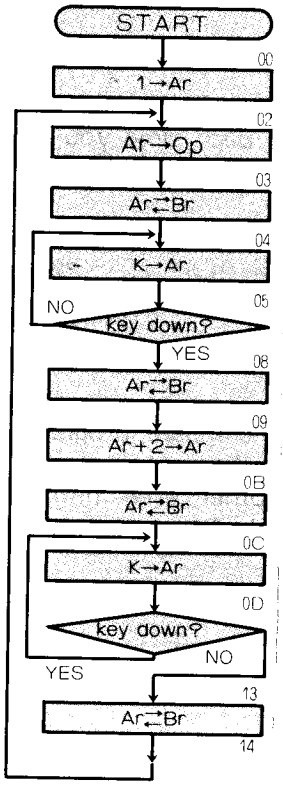
Notice that the CH command is used no less than four times. This is because any key pressed at a KA command causes the previous contents of Ar to be overwritten. As Ar is also used for increasing the count by 1, each time that there is any chance that Ar may be overwritten by a KA, the count must be stored and restored afterwards; and since there are two KAs in the program, each of them needs two CHs to achieve this.

No.13 Display Odd Hex Numbers in ascending order

PROGRAM

address	command	machine code
00	TIA	8
01	⟨1⟩	1
02	A0	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨2⟩	2
0B	CH	2
0C	KA	0
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨3⟩	3
10	JUMP	F
11	⟨0⟩	0
12	⟨C⟩	C
13	CH	2
14	JUMP	F
15	⟨0⟩	0
16	⟨2⟩	2

FLOWCHART



Move 1 to Ar to be displayed first.

Display Ar on HEX. LED.

Store Ar in Br to avoid over-writing.

If key input — go to next step.

No key input—wait.

Return stored count to Ar from Br.

Add 2 to count (alternate numbers displayed)

Store count in Br again.

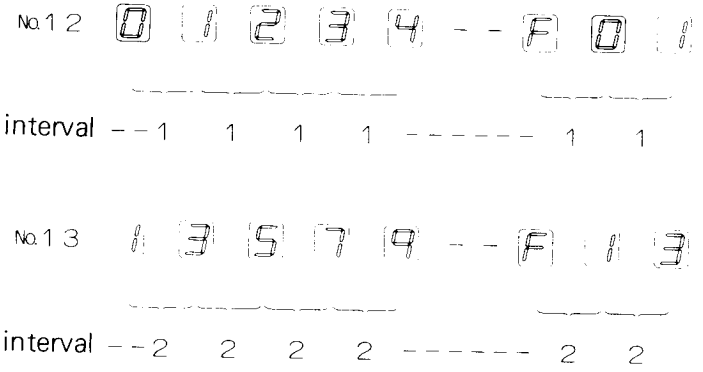
If key still pressed—wait. If key released—go to 13.

Restore count to Ar again. Go back to 02.

A) Key in the program and check it.

B) Execute the program by pressing RESET, 1, RUN.
When RUN is pressed [1] is displayed. Each time you press and release a key, odd numbers will be displayed in the following order:

Compare this flowchart and program with the flowchart and program in No.12. You will find that the only differences are that TIA puts 1 instead of 0 into Ar at 01 and AIA adds 2 instead of 1 to Ar at 0A. Look at the diagrams below to see the differences put in another way.

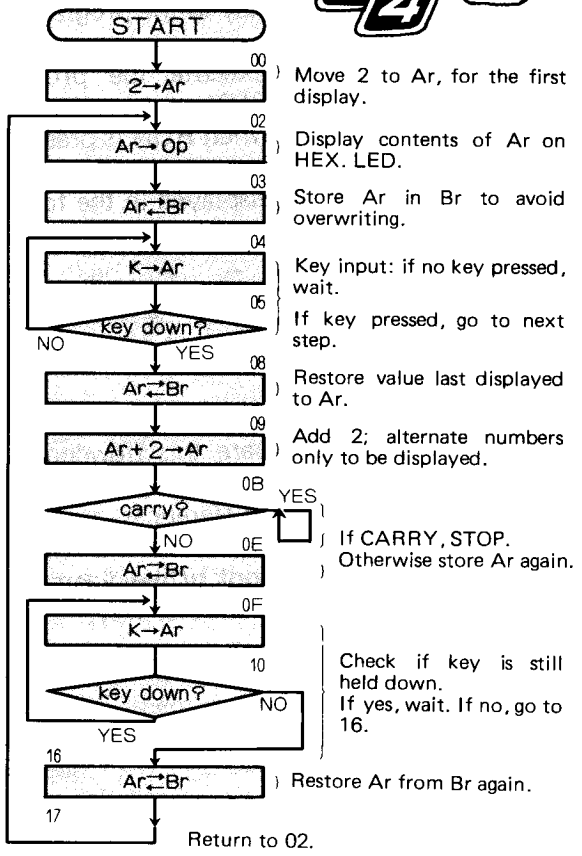


No.14 Display Even Hex Numbers in ascending order, once only

PROGRAM

address	command	machine code
00	TIA	8
01	⟨2⟩	2
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨2⟩	2
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨B⟩	B
0E	CH	2
0F	KA	0
10	JUMP	F
11	⟨1⟩	1
12	⟨6⟩	6
13	JUMP	F
14	⟨0⟩	0
15	⟨F⟩	F
16	CH	2
17	JUMP	F
18	⟨0⟩	0
19	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Execution—press RESET, 1, RUN to start.

When RUN is pressed, the HEX. LED will display 2. Each time you press and release a number key the display increases by 2, like this:

When the display shows E, the program will stop.

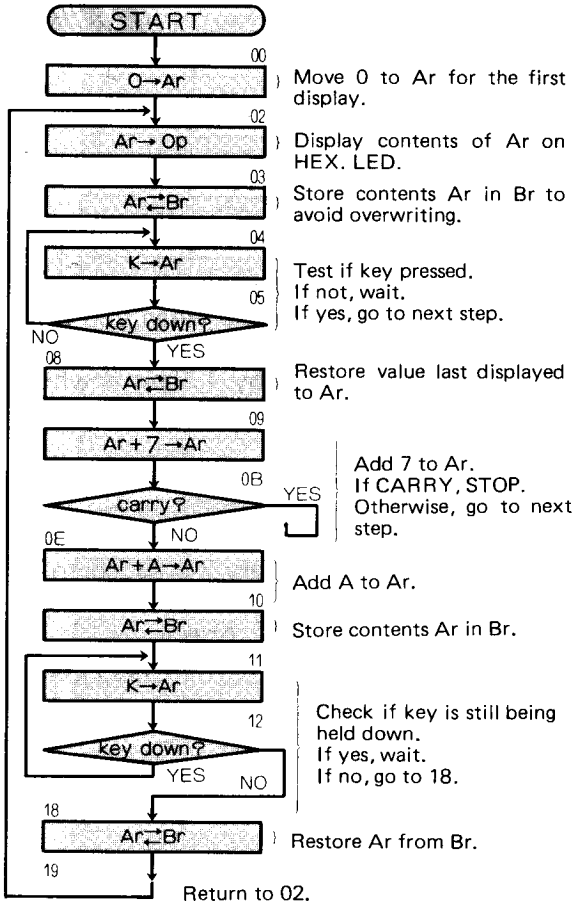
Now compare program 13 with program 14. Apart from the fact that one displays odd numbers and the other even numbers, the main difference is that No.14 stops after it reaches E.

No.15 Display Decimal Numbers in ascending order, once only

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨7⟩	7
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨B⟩	B
0E	AIA	9
0F	⟨A⟩	A
10	CH	2
11	KA	0
12	JUMP	F
13	⟨1⟩	1
14	⟨8⟩	8
15	JUMP	F
16	⟨1⟩	1
17	⟨1⟩	1
18	CH	2
19	JUMP	F
1A	⟨0⟩	0
1B	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Execution—press RESET, 1, RUN to start.

When RUN is pressed the HEX. LED will display 0. Each time you press a number key the display will increase by 1, until you reach 9. Then the program stops.

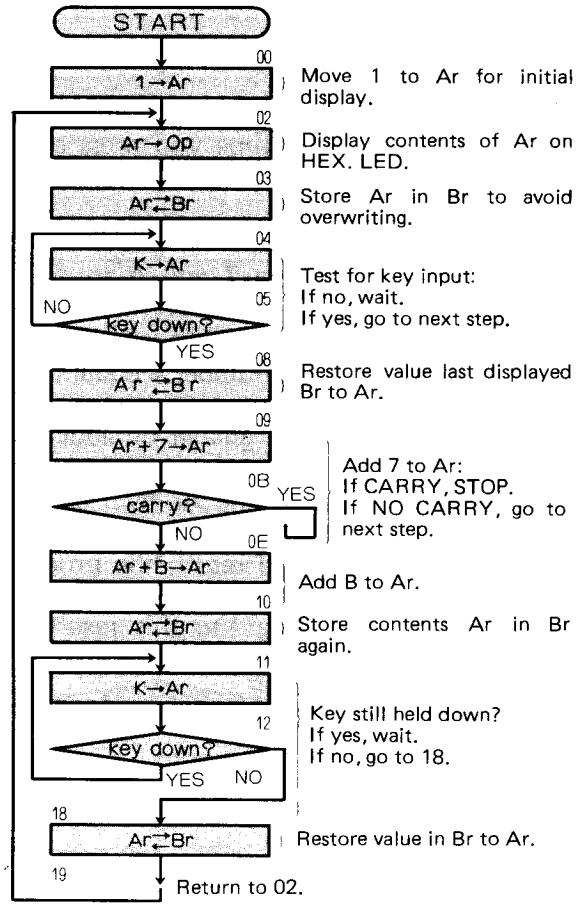
Adding 7 and A to Ar? It's a clever binary trick! Refer to page 14 where we explained calculating in binary. Adding 7 to A number results in a carry if the number is equal to or greater than 9. At address 09, the program checks to see if the number in Ar is 9 or greater by adding 7 and checking for a carry. The purpose of this program is to increase the number in the Ar by 1 each time. So when we add 7 at address 09, we're adding 6 too many. To get the desired result, the program deducts 6 (see page 14). 6 in binary is 0110 inverting 1 and 0 we get 1001. Adding 1 to this result, we get 1010, 10 in decimal, A in hexadecimal. This is the reason we add A at 0E.

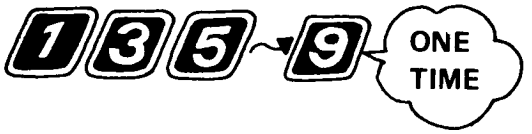
No.16 Display Odd Decimal Numbers, once only

PROGRAM

address	command	machine code
00	TIA	8
01	⟨1⟩	1
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨7⟩	7
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨B⟩	B
0E	AIA	9
0F	⟨B⟩	B
10	CH	2
11	KA	0
12	JUMP	F
13	⟨1⟩	1
14	⟨8⟩	8
15	JUMP	F
16	⟨1⟩	1
17	⟨1⟩	1
18	CH	2
19	JUMP	F
1A	⟨0⟩	0
1B	⟨2⟩	2

FLOWCHART





A) Key in the program and check it.

B) Execution — press RESET, 1, RUN to start.

The program displays the odd decimal numbers 1, 3, 5, 7, 9 when you press a key. It is very like No.15 except that the count increases by 2 each time instead of by 1. This is why B is added to Ar at 0E, instead of A. (B is 1 more than A).

Consider again what happens when binary numbers are added. In No.15 we saw that binary 5 + binary 7 = 1100, with no carry. Here is what happens when binary 7 is added to binary 9:

Ar = 9 . . . . . 1001 in binary
Ar + 7 0111
(1) 0000 There is a carry!

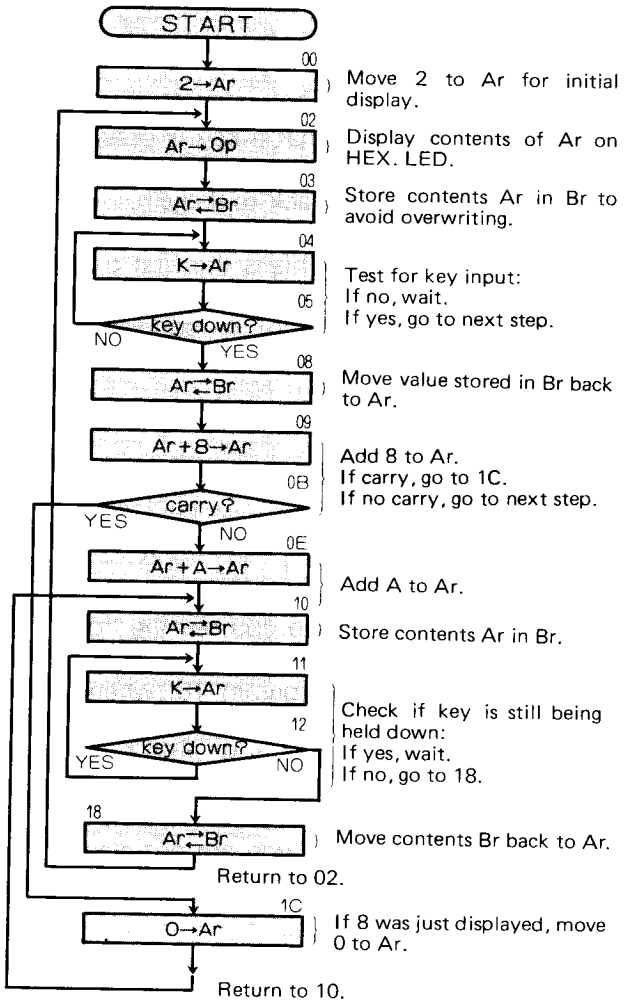
Because there is a carry, the FLAG is set to 1, and the JUMP is executed.

No.17 Repeated display of Even Decimal Numbers, in ascending order

PROGRAM

address	command	machine code
00	TIA	8
01	⟨2⟩	2
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨8⟩	8
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨C⟩	C
0E	AIA	9
0F	⟨A⟩	A
10	CH	2
11	KA	0
12	JUMP	F
13	⟨1⟩	1
14	⟨8⟩	8
15	JUMP	F
16	⟨1⟩	1
17	⟨1⟩	1
18	CH	2
19	JUMP	F
1A	⟨0⟩	0
1B	⟨2⟩	2
1C	TIA	8
1D	⟨0⟩	0
1E	JUMP	F
1F	⟨1⟩	1
20	⟨0⟩	0

FLOWCHART



A) Key in the program and check it.

B) Execution – press RESET, 1, RUN to start.

Each time you press a key, even numbers are displayed in ascending order.
The new feature in this program is what happens at 0B if there is a carry. Instead of stopping, the program JUMPS to 1C where Ar is set to 0 for the next display command.

What is happening ?

We start with 2 in Ar and this is increased by 2 each time we go through the loop. When the display reaches 8, what happens is this:–

Ar ..... 1000 (binary 8)
Ar + 8 1000
(1) 0000 There is a carry.

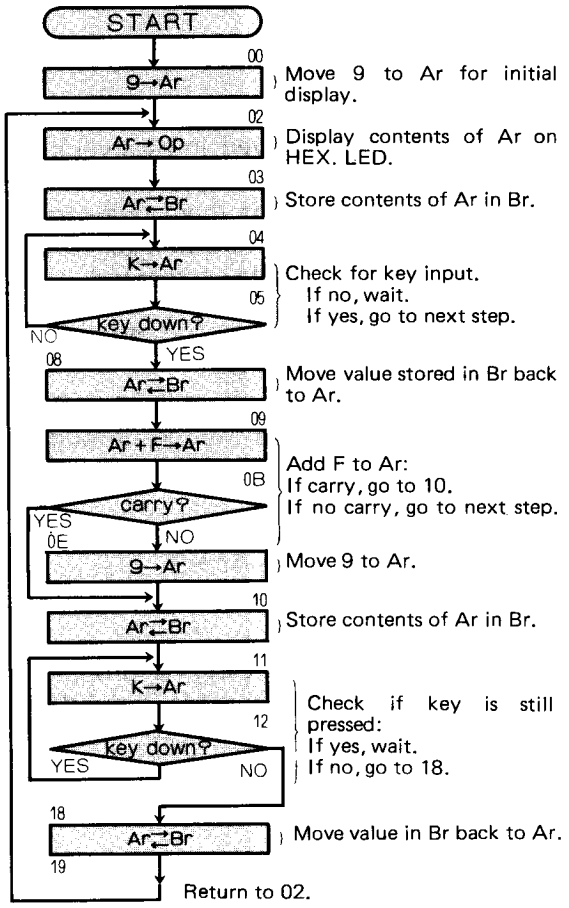
The carry is dropped and the next number to be displayed is 0 and the JUMP to 1C is the best way to get 0 into Ar. Notice that the program then JUMPS back to 10 to check that the key has been released.

No.18 Repeated display of Decimal Numbers in descending order

PROGRAM

address	command	machine code
00	TIA	8
01	⟨9⟩	9
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨F⟩	F
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨0⟩	0
0E	TIA	8
0F	⟨9⟩	9
10	CH	2
11	KA	0
12	JUMP	F
13	⟨1⟩	1
14	⟨8⟩	8
15	JUMP	F
16	⟨1⟩	1
17	⟨1⟩	1
18	CH	2
19	JUMP	F
1A	⟨0⟩	0
1B	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Execution — press RESET, 1, RUN to start.

When RUN is pressed, 9 is displayed. Then each time that a key is pressed and released the displayed value is reduced by 1. When the display reaches 0, it goes back to 9 again.

This is the first program to include binary deduction. It may seem strange that if F is added to Ar the value of Ar is reduced by 1. But remember, the program drops any carry over.

Here are some examples:—

	1....0001	6....0110	0....0000
	+F....1111	+F....1111	+F....1111
=	(1) 0000	(1) 0101	(0) 1111

Do you see the "odd one out"? It is 0 + F because that is the only one where there is no carry. In program 18, when there is no carry at 0B this is because the last value displayed was 0. So we want to put in 9, into Ar, ready for the next display, and this is exactly what happens at 0E.

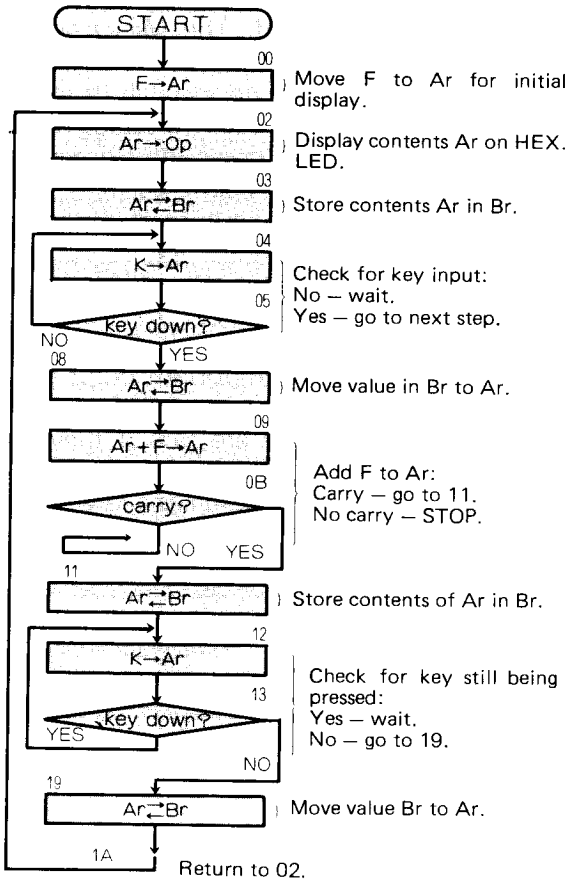
If you are not clear about any points refer to earlier parts of the manual.

No.19 Display Hex Numbers in descending order, once only

PROGRAM

address	command	machine code
00	TIA	8
01	⟨F⟩	F
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨F⟩	F
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨1⟩	1
0E	JUMP	F
0F	⟨0⟩	0
10	⟨E⟩	E
11	CH	2
12	KA	0
13	JUMP	F
14	⟨1⟩	1
15	⟨9⟩	9
16	JUMP	F
17	⟨1⟩	1
18	⟨2⟩	2
19	CH	2
1A	JUMP	F
1B	⟨0⟩	0
1C	⟨2⟩	2

FLOWCHART



Compare this program and flowchart with No.12. The difference is that in No.12 Ar starts with 0, and 1 is added at 0A, and in No.19 Ar starts with F and 1 is deducted at 0A, as explained in program No.18.

Here is how this program works:—

	binary	hex
F + F	1111	F
	+ 1111	F
	11110	E (F-1)

The carry is dropped and the result is E.

E + F	1110	E
	+ 1111	F
	11101	D (E-1)

The carry is dropped and the result is D.

. . . and so on.

F is continually added to the preceding result stored in the Ar and the total is always one less than before.

The only sum in binary not to give a carry is 0 + F. For all other sums there is a carry and the FLAG is set to 1 and the JUMP at 0B is executed :

0 + F	0000	0
	+ 1111	F
	1111	F

There is no carry so the program ends.

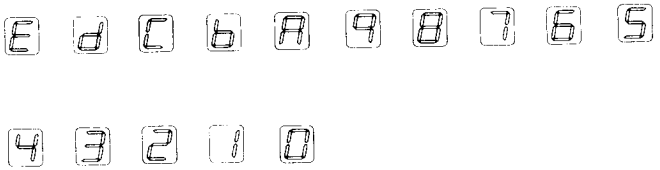
This is another difference between No.12 and No.19: in No.19 the countdown stops when it reaches 0.

A) Key in the program and check it.

B) Execution – Press RESET, 1, RUN.

When RUN is pressed, F is displayed. Each time that a key is pressed, the value displayed is reduced by 1 :

Example:



After 0 is displayed, no further change takes place; the program ends.

The explanations given for the flowchart to program No.12 are very important. The logic used there is also used with very little change in programs 13-19 and will continue to be used throughout this manual. Logic refers to the key input routines.

You must be clear on the following points.

First, the importance of the KA command in all of these programs.

The purpose of the first KA command is to allow a key to be pressed and that value to be stored in a register.

The purpose of the next KA commands in these programs is to check that the key which was pressed at the first KA has been released. The Microcomputer Trainer works very fast. If the second KA is left out, when a key is pressed at 04, the program will go through its loop and return to 04 again before the key has been released. Since the FLAG = 0 as long as a key is held down, the program will be able to increase or decrease the count again – and the loop may be repeated many times like this before the key is finally released.

Remember this when you write your own programs.

The second important point here is that you must not forget to SAVE the contents of Ar before executing a KA command. The contents are safe if they are stored in Br, using the CH command. But if you forget, then the previous contents of Ar will be lost as soon as a key is pressed. And don't forget to put it back into Ar when the key input routine is complete.

Refer back to program No.12, if you're still unsure.

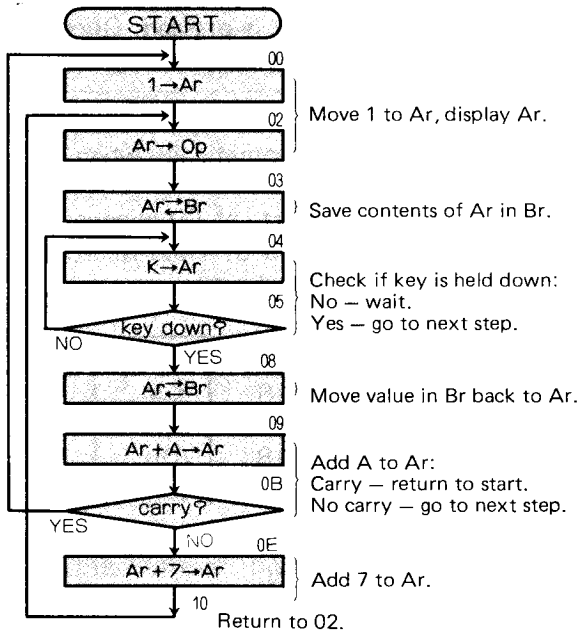
No.20 Electronic Dice — stops when key is released

This program displays only the numbers you find on dice (1-6) in rapid succession when you press a key. Release the key to display the number you have "rolled."

PROGRAM

address	command	machine code
00	TIA	8
01	⟨1⟩	1
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	CH	2
09	AIA	9
0A	⟨A⟩	A
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨0⟩	0
0E	AIA	9
0F	⟨7⟩	7
10	JUMP	F
11	⟨0⟩	0
12	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

When RUN is pressed the HEX. LED displays 1. If a number key is pressed the HEX. LED changes very rapidly and stops only when the key is released. It starts again when a number key is pressed again.

There is some new binary arithmetic demonstrated here. We want to restrict the display to the numbers you expect to find on a dice, i.e. 1—6. When the program has counted to 6, the A added at that point gives a carry in the answer and this causes the program to start again from the beginning. If there is no carry, then 7 is added and this increases Ar by 1. (Check program No.15 again where A and 7 are used to add 1 to Ar, but the other way around).

Try adding binary A yourself to binary 6 to check that the answer has a carry.

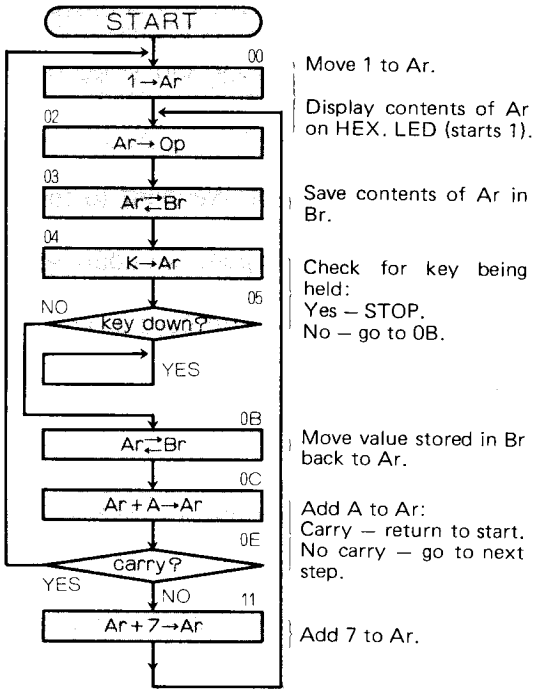
No.21 Electronic Dice – stops when key is pressed

In this program, the electronic dice stop "rolling" when you press a key.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨1⟩	1
02	AO	1
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨B⟩	B
08	JUMP	F
09	⟨0⟩	0
0A	⟨8⟩	8
0B	CH	2
0C	AIA	9
0D	⟨A⟩	A
0E	JUMP	F
0F	⟨0⟩	0
10	⟨0⟩	0
11	AIA	9
12	⟨7⟩	7
13	JUMP	F
14	⟨0⟩	0
15	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Press RUN.

When RUN is pressed, the display will change very rapidly. It will stop changing as soon as a number key is pressed. Once it has stopped it can be restarted only by pressing RESET, 1, RUN.

Let us try to change this program, to make it start automatically when the key is released.

First, at 09 change the 0 to 1 and at 0A change the 8 to 6. Then at 16 enter the following code in the usual way:

address	command	key
16	KA	0
17	JUMP	F
18	(0)	0
19	(0)	0

address	command	key
1A	JUMP	F
1B	(1)	1
1C	(6)	6

Check it and run the program again. Press RESET, 1, RUN.

(2) Group 2 Commands

The next group of commands are AM, MA, M+, M−, TIY, AIY, and CAL TIMR. All of them except CAL TIMR involve the use of the Y-register and the user MEMORY.
The word MEMORY is often used to describe a group of addresses. In program 22, data will be stored 'in MEMORY'. Examine the diagram at the end of "Group 1 Commands". The memory addresses 50-5F have been set aside for data storage. It is these addresses which the Y-register (Yr) uses:

Value of Yr	Address...*	Value of Yr	Address...*
0	50	8	58
1	51	9	59
2	52	A	5A
3	53	B	5B
4	54	C	5C
5	55	D	5D
6	56	E	5E
7	57	F	5F

* This column indicates the address used or "pointed to" by each particular Y-register value.

Addresses 50-5F are similar to the addresses previously used. But with the help of Yr, data can be stored in them or read from them, so they have a special usefulness.
Yr can also be used as an extra register and for many of the same purposes as Ar.

CAL COMMAND – (1). CAL is short for CALL.

It is always followed by a subroutine – such as TIMR. A SUBROUTINE is a group of instructions that do a particular job. TIMR counts fractions of seconds. CAL calls subroutines, executes them, and then the program continues.

TIMR is one of 16 subroutines used by the Microcomputer Trainer. They are discussed in group 3 commands.

CH COMMAND (2)

CH was first discussed under Program No. 9. It was used to swap Ar/Br contents.
But it can also exchange the contents of the Y and Z-registers:

Ar	Br	Watch out because both swaps take place simultaneously.
Yr	Zr	

No.22 Use of TIY, AIY and AM to store data in memory

TIY COMMAND (Transfer Into Yr)

This command uses two characters of machine code. The value of the second is loaded into Yr.
In a flowchart it is written: n → Yr, where n = 0-F.

AM COMMAND (Transfer contents of Ar to Memory)

This command moves the contents of Ar into the address (50-5F) pointed to by Yr. (Yr is set by using TIY).
In a flowchart it is written: Ar → M.

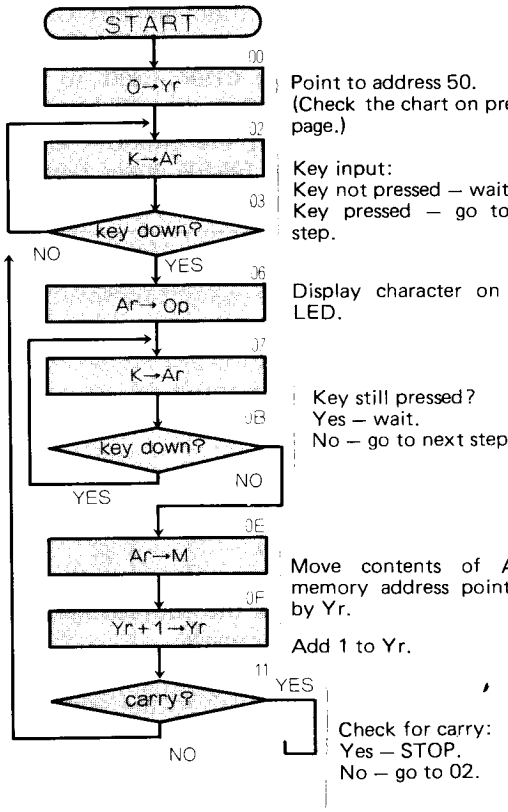
AIY COMMAND (Add Into Yr)

This command uses two characters of machine code. The value of the second is added into Yr and the result stored in Yr.
It is written: Yr + n → Yr, where n = 0-F.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	KA	0
03	JUMP	F
04	⟨0⟩	0
05	⟨2⟩	2
06	AO	1
07	KA	0
08	JUMP	F
09	⟨0⟩	0
0A	⟨E⟩	E
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨7⟩	7
0E	AM	4
0F	AIY	B
10	⟨1⟩	1
11	JUMP	F
12	⟨1⟩	1
13	⟨1⟩	1
14	JUMP	F
15	⟨0⟩	0
16	⟨2⟩	2

FLOWCHART



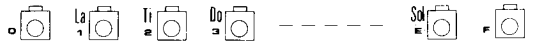
This program is another example of two KA commands being used in different ways. At the first KA, the program waits if a key has not been pressed; at the second, it waits if a key has not been released.
The program ends when data has been stored at all 16 addresses 50-5F.

A) Key in the program and check it.

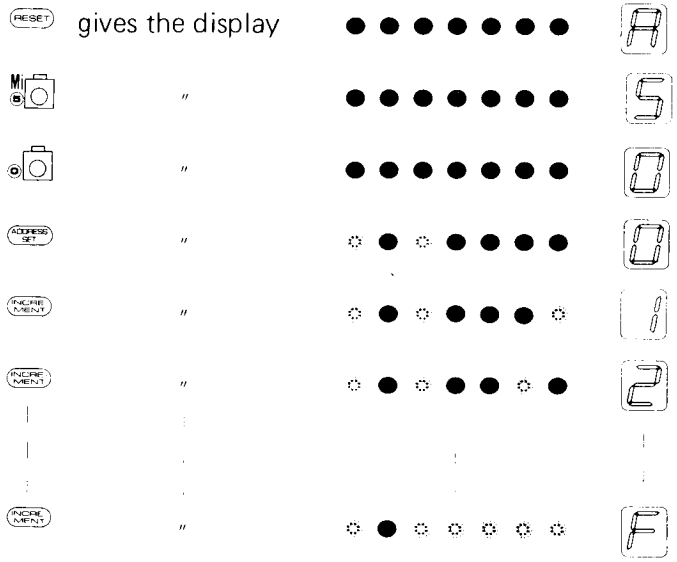
B) Start the program by pressing RESET, 1, RUN.

C) Press 16 number keys to store the values in memory.

For example, press keys 0-F in order as follows:



D) To read the contents of addresses 50-5F, press these keys:



So it worked! The program started with 0 in Yr. This caused the first character you keyed in to be stored at address 50. Then 1 was added to Yr so that your next key value was stored at 51 ..... and so on.

No.23 Use of MA to display memory contents

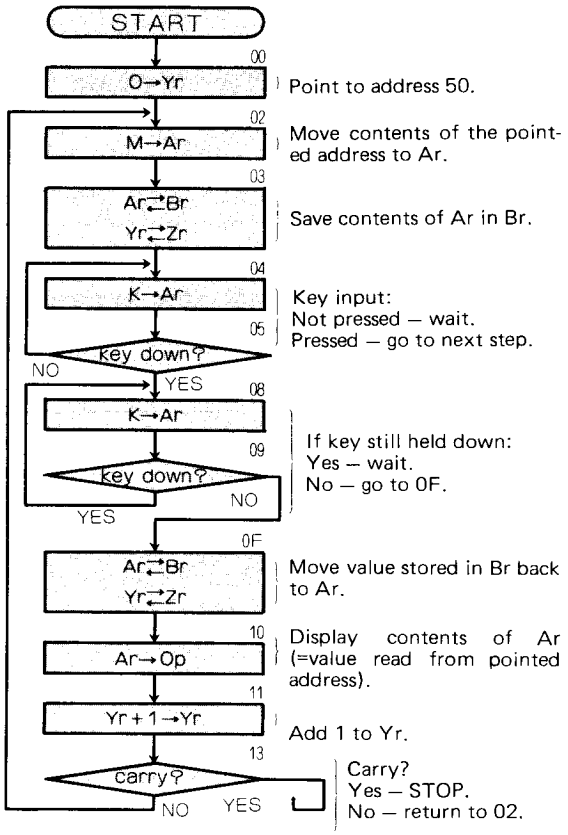
This program displays the contents of 50-5F in turn.

> MA COMMAND (Transfer contents of Memory to Ar)
>
> This command reads the value of the address pointed to by Yr and moves it to Ar.
> In a flowchart it is written: M → Ar.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	CH	2
04	KA	0
05	JUMP	F
06	⟨0⟩	0
07	⟨4⟩	4
08	KA	0
09	JUMP	F
0A	⟨0⟩	0
0B	⟨F⟩	F
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨8⟩	8
0F	CH	2
10	AO	1
11	AIY	B
12	⟨1⟩	1
13	JUMP	F
14	⟨1⟩	1
15	⟨3⟩	3
16	JUMP	F
17	⟨0⟩	0
18	⟨2⟩	2

FLOWCHART



A) Key in the program and check it.

B) Load any 16 key values (0-F) into 50-5F. i.e.,

C) Press RESET, 1, RUN to start. Then press any number key to display the contents of each address 50-5F in turn once only.

No.24 Use of M + to add to displayed numbers

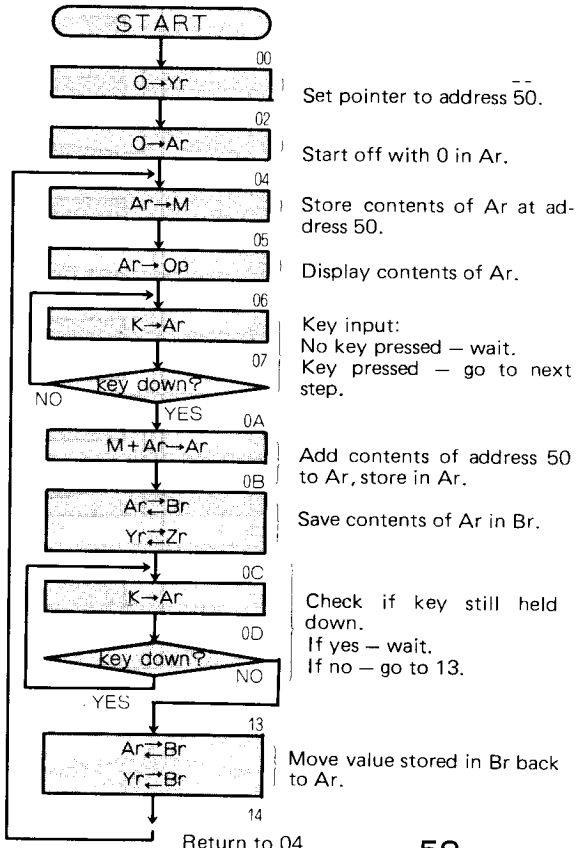
M+ COMMAND

This command adds the value of the address pointed to by Yr to Ar and stores the answer in Ar.
In a flowchart it is written: M + Ar → Ar.
If the addition generates a carry, the FLAG is set to 1.
If no FLAG is generated, the FLAG is set to 0.
When a JUMP or CAL command directly follows the M+ command, it will be executed if the FLAG is set to 1, but not if it is set to 0.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	AO	1
06	KA	0
07	JUMP	F
08	⟨0⟩	0
09	⟨6⟩	6
0A	M+	6
0B	CH	2
0C	KA	0
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨3⟩	3
10	JUMP	F
11	⟨0⟩	0
12	⟨C⟩	C
13	CH	2
14	JUMP	F
15	⟨0⟩	0
16	⟨4⟩	4

FLOWCHART



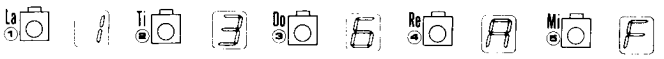
This program displays a figure on the HEX. LED and adds it to any value keyed in.

A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

C) The display starts off with 0.

Key in any value 0-F and it will be added to the total which will be displayed. If you key in 1, 2, 3, 4, 5 in order, the results will look like this:



When the sum is larger than F(15), the carry is dropped.

For example:

```
          1000 ( 8)
        + 1010 (10)
        ----------
drop carry 1 0010 ( 2)
```

2 is displayed.

No.25 Use of M − to reduce a displayed value

In No. 24 M+ was used to demonstrate addition. Here M− is used for subtraction.

M− COMMAND

In a flowchart M− is written: M−Ar → Ar.
This command subtracts the contents of Ar from the contents of the memory address pointed to by Yr and puts the answer in Ar.

If the value of Ar is greater than the value of the pointed address, the subtraction cannot be done. This condition is called OVERFLOW. The OVERFLOW condition sets the FLAG to 1. Otherwise M− sets the FLAG to 0.
M− also affects JUMP and CAL in the same way as M+. Refer back to Program No. 24.

A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

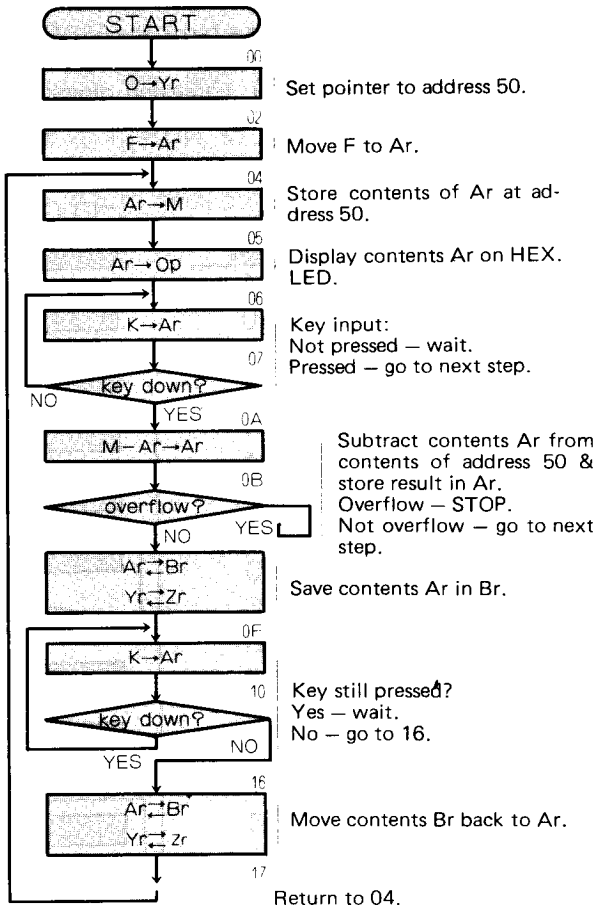
C) The initial display shows F.

F will be reduced by the value of the keys pressed until the display reaches 0. The program also stops if the result is less than 0.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨F⟩	F
04	AM	4
05	AO	1
06	KA	0
07	JUMP	F
08	⟨0⟩	0
09	⟨6⟩	6
0A	M-	7
0B	JUMP	F
0C	⟨0⟩	0
0D	⟨B⟩	B
0E	CH	2
0F	KA	0
10	JUMP	F
11	⟨1⟩	1
12	⟨6⟩	6
13	JUMP	F
14	⟨0⟩	0
15	⟨F⟩	F
16	CH	2
17	JUMP	F
18	⟨0⟩	0
19	⟨4⟩	4

FLOWCHART



No.26 Use of CAL TIMR

CAL TIMR COMMAND (CALI TIMeR)

When this command is executed a pause occurs. The length of the pause is determined by the value of Ar. This is the formula:

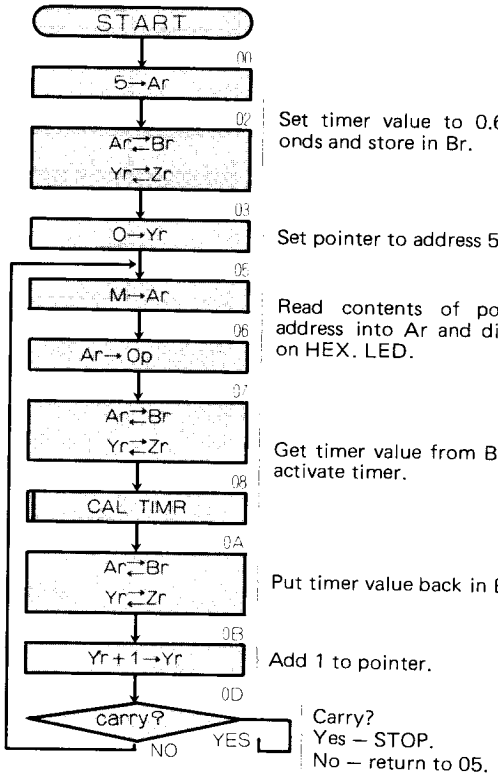
$$\text{seconds} = \frac{\text{contents of Ar} + 1}{10}$$

In program No. 26 the length of the pause is (5+1) ÷ 10 = 0.6 seconds.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	CH	2
03	TIY	A
04	⟨0⟩	0
05	MA	5
06	AO	1
07	CH	2
08	CAL	E
09	TIMR	C
0A	CH	2
0B	AIY	B
0C	⟨1⟩	1
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨D⟩	D
10	JUMP	F
11	⟨0⟩	0
12	⟨5⟩	5

FLOWCHART



Set timer value to 0.6 seconds and store in Br.

Set pointer to address 50.

Read contents of pointed address into Ar and display on HEX. LED.

Get timer value from Br and activate timer.

Put timer value back in Br.

Add 1 to pointer.

Carry?
Yes – STOP.
No – return to 05.

A) Key in the program and check it.

B) Key values (0-F) into addresses 50-5F

Example:

C) Press RESET, 1, RUN to start.

The HEX. LED will display the characters you have loaded into 50-5F addresses at roughly half-second intervals.
TIMR is a very important subroutine. Programs which display figures, like this program, need to pause because of the speed at which the Microcomputer Trainer works. You don't always want to make the operator press a key to display a new figure.

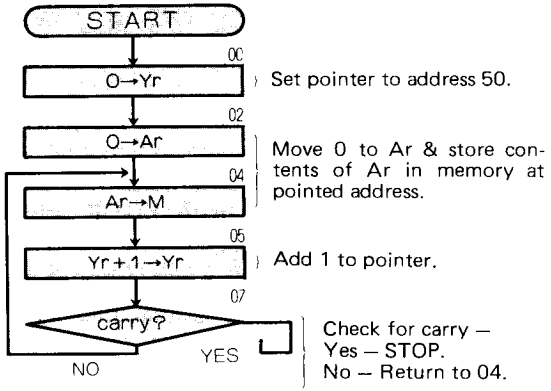
No.27 Load Zero into Memory

This is a simple loop which starts with 0 in Ar and puts it into each of the addresses 50-5F in turn.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	AIY	B
06	⟨1⟩	1
07	JUMP	F
08	⟨0⟩	0
09	⟨7⟩	7
0A	JUMP	F
0B	⟨0⟩	0
0C	⟨4⟩	4

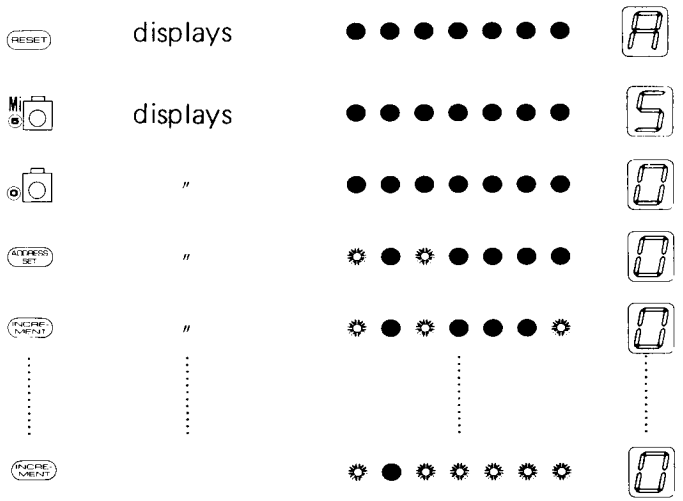
FLOWCHART



A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

C) Check that 0 has been loaded at all addresses:



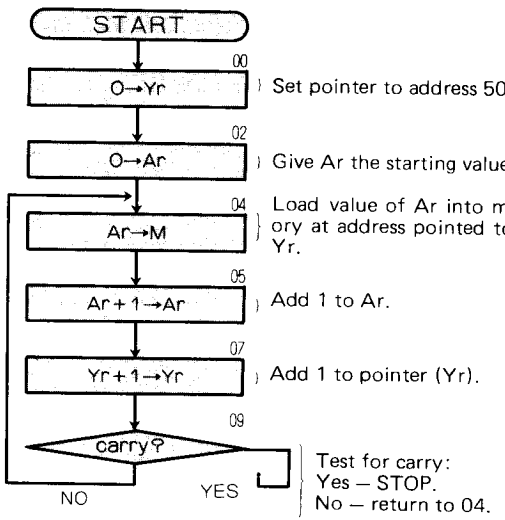
No.28 Load 0-F into Memory

This program loads 0 at address 50, 1 at 51, 2 at 52, ... F at 5F.

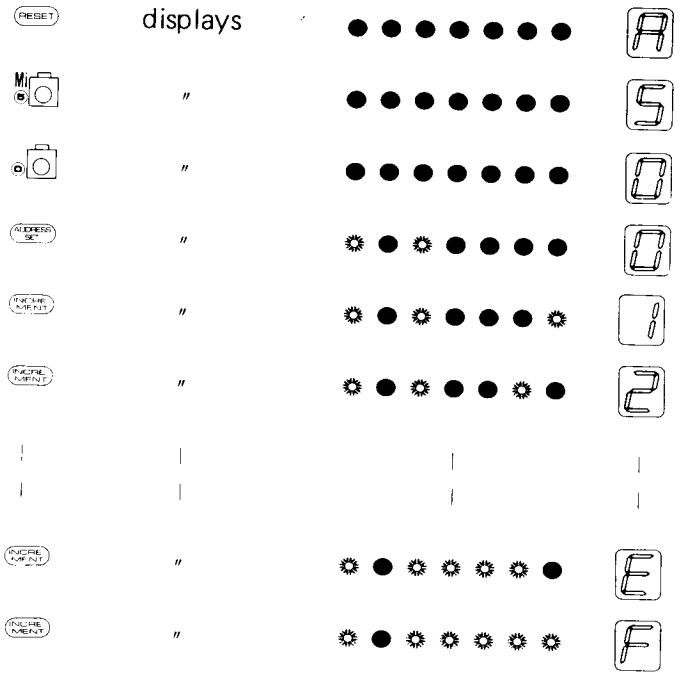
PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	AIA	9
06	⟨1⟩	1
07	AIY	B
08	⟨1⟩	1
09	JUMP	F
0A	⟨0⟩	0
0B	⟨9⟩	9
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4

FLOWCHART



Press:



A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

C) Read out addresses 50-5F like this:

Program No. 28 is the same as 27 except that here Ar is increased by 1 each time the loop is executed.

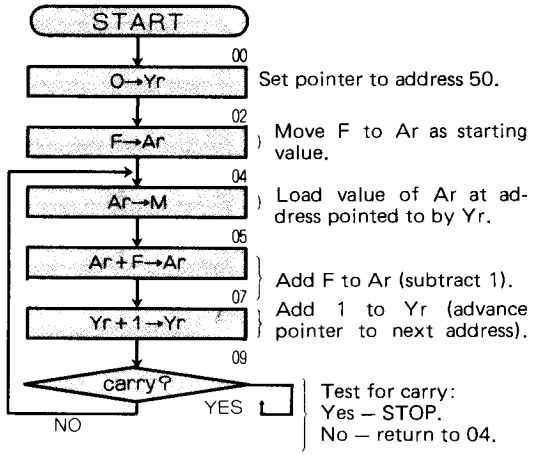
No.29 Load F-0 into Memory

This program is similar to the previous one. Here, F is loaded at 50, E at 51, ... 0 at 5F.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨F⟩	F
04	AM	4
05	AIA	9
06	⟨F⟩	F
07	AIY	B
08	⟨1⟩	1
09	JUMP	F
0A	⟨0⟩	0
0B	⟨9⟩	9
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4

FLOWCHART

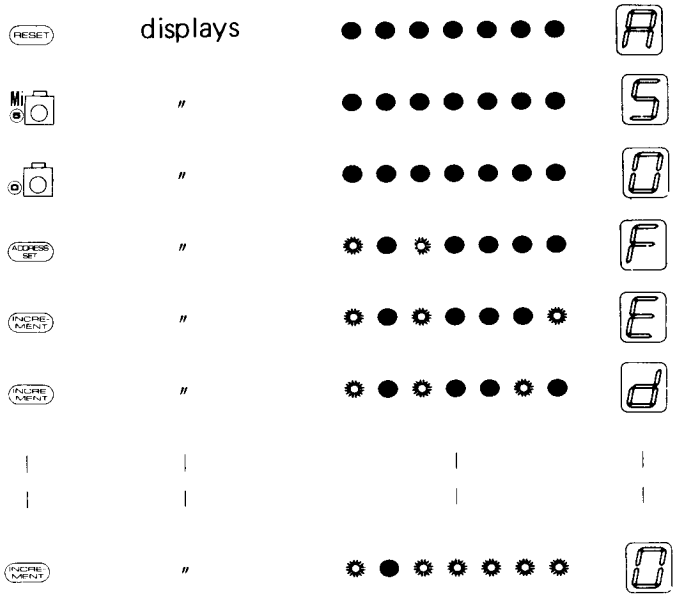


A) Key in the program and check it.

B) Start the program by pressing RESET, 1, RUN.

C) Read out addresses 50-5F like this:

Press:



This time we started with F in Ar and added F to Ar each time through the loop. Do you remember why this reduces Ar by 1? If you don't, then look again at the explanations for programs 18 and 19.

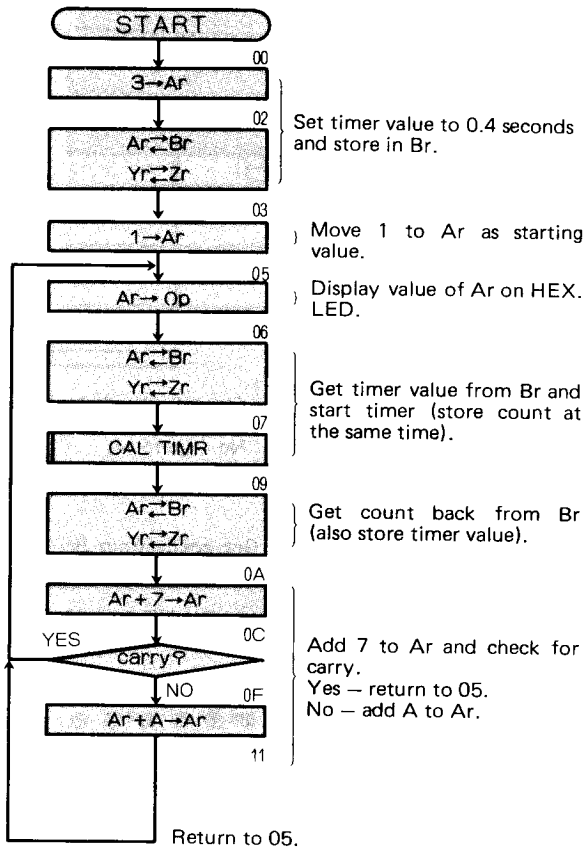
No.30 Decimal Counting in Ascending Order

This program displays decimal numbers 0, 1, 2, ... 9, 0, 1 etc. at intervals of 0.4 seconds.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨3⟩	3
02	CH	2
03	TIA	8
04	⟨1⟩	1
05	AO	1
06	CH	2
07	CAL	E
08	TIMR	C
09	CH	2
0A	AIA	9
0B	⟨7⟩	7
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨5⟩	5
0F	AIA	9
10	⟨A⟩	A
11	JUMP	F
12	⟨0⟩	0
13	⟨5⟩	5

FLOWCHART



Set timer value to 0.4 seconds and store in Br.

Move 1 to Ar as starting value.

Display value of Ar on HEX. LED.

Get timer value from Br and start timer (store count at the same time).

Get count back from Br (also store timer value).

Add 7 to Ar and check for carry.
Yes – return to 05.
No – add A to Ar.

A) Key in the program and check it.

B) To start, press RESET, 1, RUN.

C) This is how the display on the HEX. LED should appear:

Prior to this program, when 7 has been added to Ar and this has resulted in a carry, we have taken special action. But in this program we go straight back to the start of the loop as though everything was perfect. It is really, but that is by accident. Try adding 7 to binary 9:

$$\begin{aligned} 9 &\ldots\ldots\ldots 1001 \\ +\,7 &\ldots\ldots\ldots 0111 \\ &= (1)0000 \end{aligned}$$

Although there was a carry, the important point is that Ar now has 0 in it – and that is the decimal number we want after 9.

Adding 7 to any of the numbers 0 – 8 does NOT give a carry, so the program adds A to Ar – refer to Program No.15 to review this process.

No.31 Decimal Counting in Descending Order

This program counts in decimal, in descending order continuously until you stop it.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨3⟩	3
02	TIY	A
03	⟨0⟩	0
04	AM	4
05	CH	2
06	TIA	8
07	⟨9⟩	9
08	CH	2
09	KA	0
0A	JUMP	F
0B	⟨0⟩	0
0C	⟨9⟩	9
0D	KA	0
0E	JUMP	F
0F	⟨1⟩	1
10	⟨4⟩	4
11	JUMP	F
12	⟨0⟩	0
13	⟨D⟩	D
14	CH	2
15	AO	1
16	CH	2
17	MA	5
18	CAL	E
19	TIMR	C
1A	AM	4
1B	CH	2
1C	AIA	9

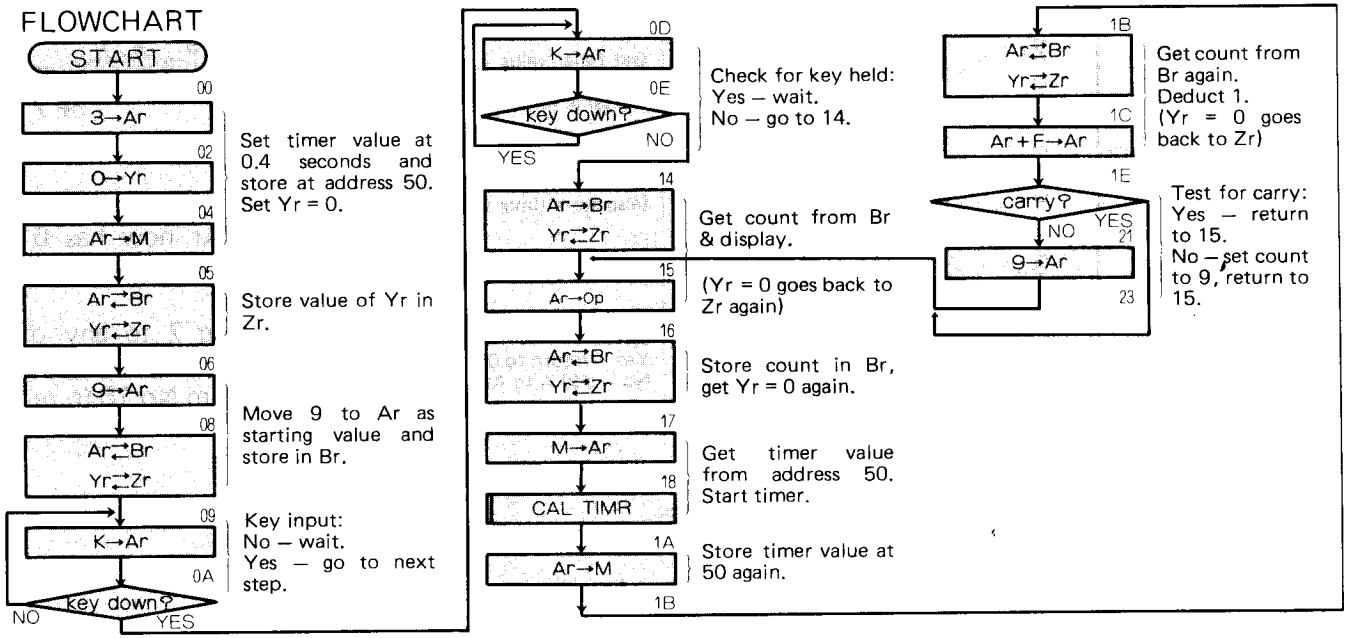
address	command	machine code
1D	⟨F⟩	F
1E	JUMP	F
1F	⟨1⟩	1
20	⟨5⟩	5
21	TIA	8
22	⟨9⟩	9
23	JUMP	F
24	⟨1⟩	1
25	⟨5⟩	5

A) Key in the program and check it.

B) To start press RESET, 1, RUN.

C) Press and release any number key to start the display:

FLOWCHART



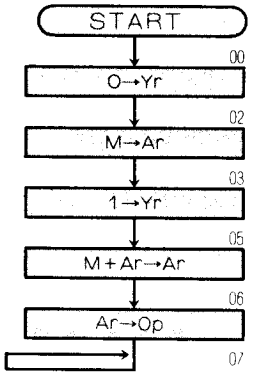
No.32 Hex Addition (1) — One Digit + One Digit = One Digit

This program takes a figure from 50, another from 51, adds them together and displays the result on the HEX. LED.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	TIY	A
04	⟨1⟩	1
05	M+	6
06	AO	1
07	JUMP	F
08	⟨0⟩	0
09	⟨7⟩	7

FLOWCHART



Move contents of address 50 to Ar.

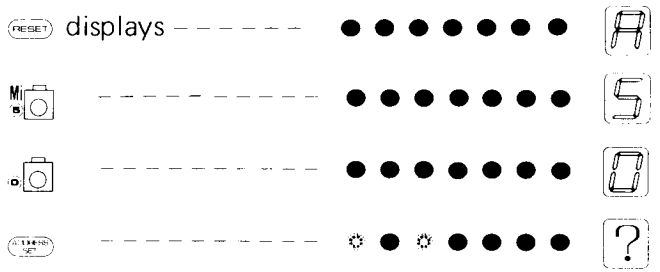
Add contents of address 51 to Ar. Put result in Ar.

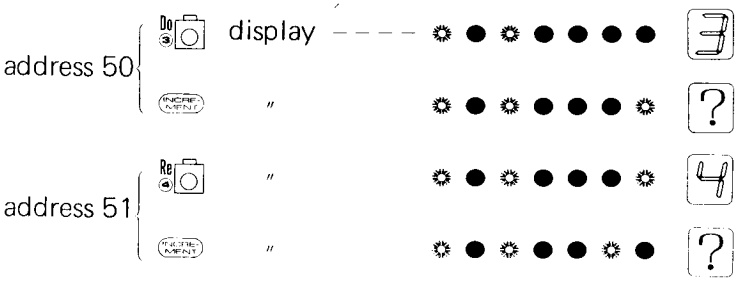
Display result (contents of Ar) on HEX. LED.

A) Key in the program and check it.

B) Before running the program, load into 50/51 the values to be added.

For example, put 3 in 50 and 4 in 51 like this:





C) Start the program by pressing RESET, 1, RUN.

D) The answer is displayed on the HEX. LED.

3 + 4 = 7

address 50 51

If the result is a 2-digit number, this program only shows the last digit of the sum on the HEX. LED.

Example:

9 + 9 = 12

Display shows 2

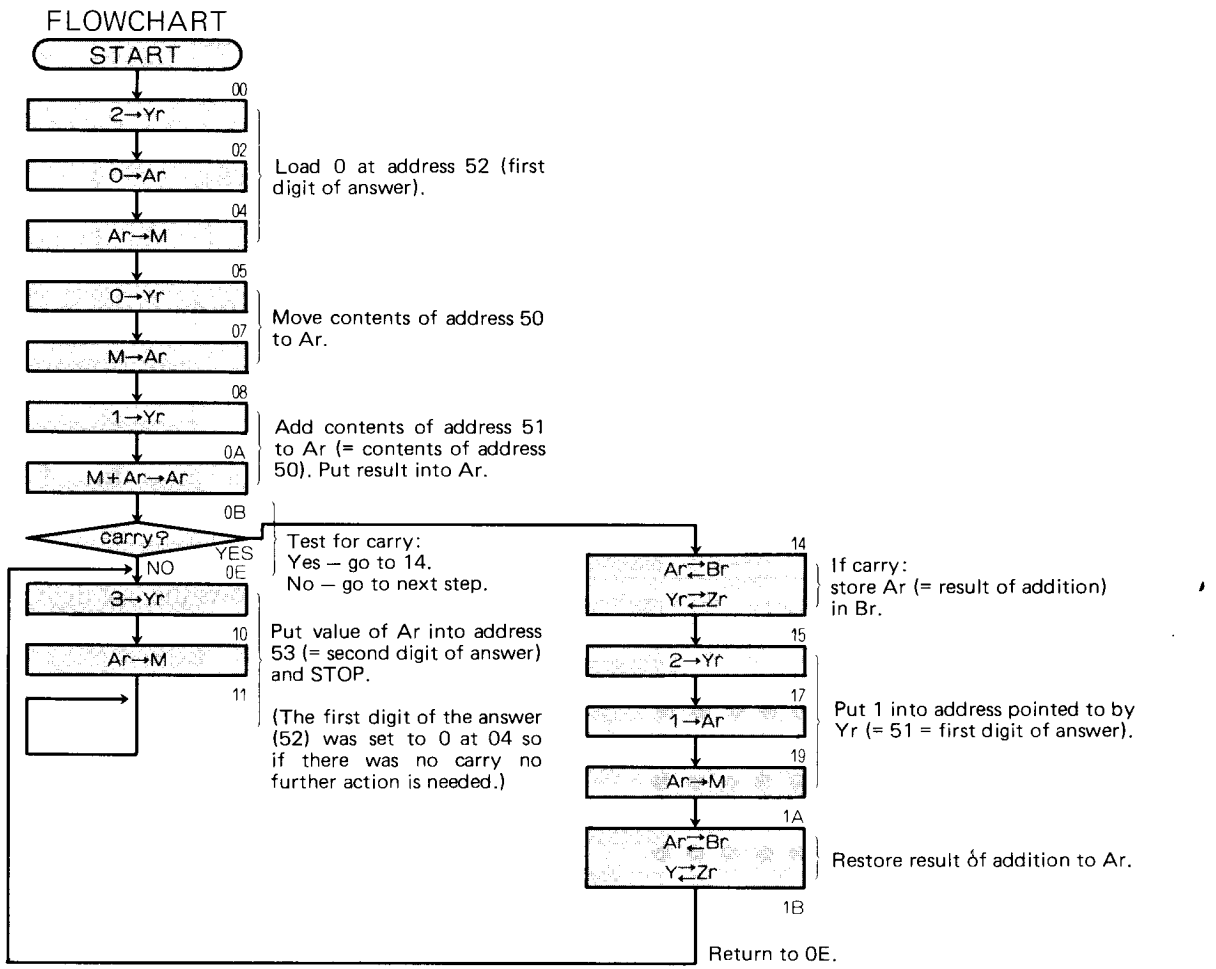
No.33 Hex Addition (2) — One Digit + One Digit = Two Digits

In Program No. 32 we displayed a 1-digit sum. In 33 we are being more ambitious and will produce a 2-digit sum, which makes the programming a bit more complicated. The first digit of the answer is stored at 52 and the second at 53.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	MA	5
08	TIY	A
09	⟨1⟩	1
0A	M+	6
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨4⟩	4
0E	TIY	A
0F	⟨3⟩	3
10	AM	4
11	JUMP	F
12	⟨1⟩	1
13	⟨1⟩	1
14	CH	2
15	TIY	A
16	⟨2⟩	2
17	TIA	8
18	⟨1⟩	1
19	AM	4
1A	CH	2
1B	JUMP	F
1C	⟨0⟩	0
1D	⟨E⟩	E

FLOWCHART

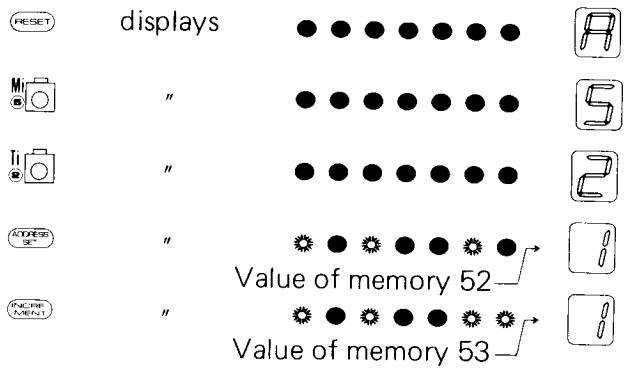


A) Key in the program and check it.

B) Load into 50 & 51 the figures to be added:
For example, 8 + 9.

C) Start the program by pressing RESET, 1, RUN.

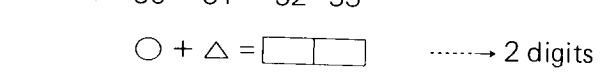
D) The program puts the answer into addresses 52 & 53 which should be recalled as shown below. The answer is displayed one digit at a time. It is a hex number: — 1 (first digit) 1 (second digit). (11 hex = 17 decimal = 8 + 9).



Is it so complicated? If the addition at 08 gives a carry, a 1 is carried over into the first digit of the answer. In the example given, the addition leaves 1 in Ar.

1 in Ar is the correct answer for the last digit despite the carry, so it is stored in address 53 at 0E just as it would have been had there been no carry.

Here is a MEMORY MAP for this program:



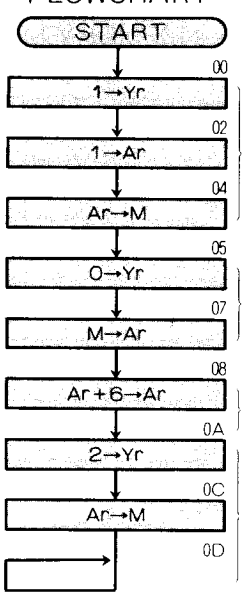
No.34 Hex to Decimal Conversion (A – F)

This program converts hex numbers to decimal. At this moment we are confining ourselves to A-F.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨1⟩	1
02	TIA	8
03	⟨1⟩	1
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	MA	5
08	AIA	9
09	⟨6⟩	6
0A	TIY	A
0B	⟨2⟩	2
0C	AM	4
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨D⟩	D

FLOWCHART



Load 1 into 51 (= first digit of answer – will always be 1 for A-F).

Get a hex number (A-F) from address 50.

Add 6 to Ar (convert to decimal).

Load contents of Ar at 52 (= second digit of answer). (First digit was dealt with at 00-04.)

A) Key in the program and check it.

B) Key into 50 the number to be converted:

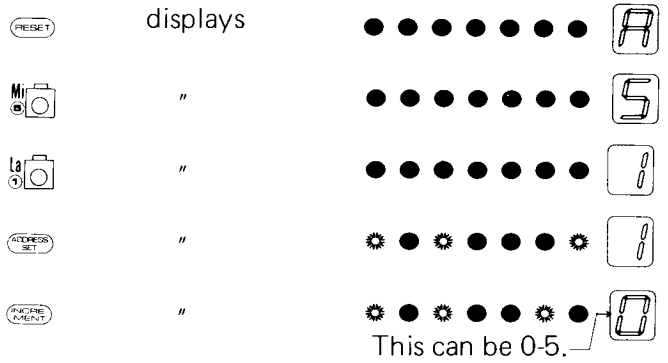
Example:

RESET Mi 0 ADDRESS SET Do A INCREMENT

C) Start the program by pressing RESET, 1, RUN.

D) Check that the number has been converted, as follows:

RESET displays



Mi " La 1 " ADDRESS SET " INCREMENT "

This can be 0-5.

In this case it is 0 because A = 10 in decimal.

The value at address 50 is increased by 6. If you try adding binary 6 to binary F the answer is 5 with a carry – but the carry is ignored in this program. As you know from previous programs, you reduce 6 to 5 by adding F.

address 50 51 52

A~F ↘2-digit decimal answer

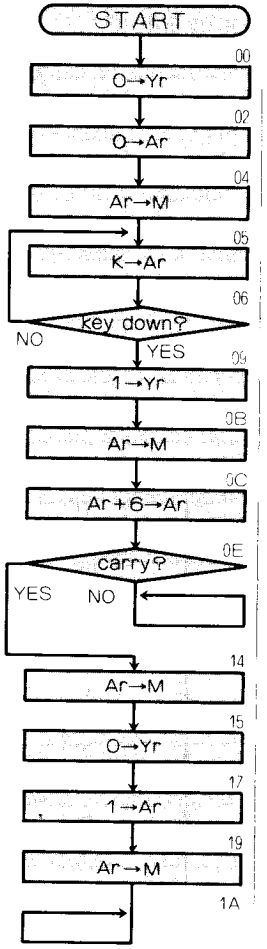
No.35 Hex to Decimal Conversion (0 – F)

This is an extension of Program No. 34. This time we allow any number (0-F) to be keyed in. The program has to distinguish between 0-9 input and A-F input, in order to decide whether the first digit of the answer should be 0 or 1. The answer goes into 50/51.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	KA	0
06	JUMP	F
07	⟨0⟩	0
08	⟨5⟩	5
09	TIY	A
0A	⟨1⟩	1
0B	AM	4
0C	AIA	9
0D	⟨6⟩	6
0E	JUMP	F
0F	⟨1⟩	1
10	⟨4⟩	4
11	JUMP	F
12	⟨1⟩	1
13	⟨1⟩	1
14	AM	4
15	TIY	A
16	⟨0⟩	0
17	TIA	8
18	⟨1⟩	1
19	AM	4
1A	JUMP	F
1B	⟨1⟩	1
1C	⟨A⟩	A

FLOWCHART



Load 0 at address 50 (= first digit of answer).

Key input:
No key pressed – wait.
Key pressed – go to next step.

Move keyed in value to address 51 (= second digit of answer).

Add 6 to Ar, test for carry:
No – STOP (input = 0-9).
Yes – go to next step (input = A-F).

If carry – move contents of Ar to address 51 (Yr = 1 at 09 above).

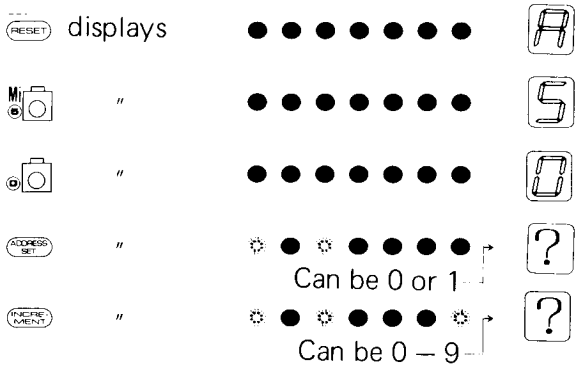
Move 1 to address 50 (= first digit of anwer) and STOP.

A) Key in the program and check it.

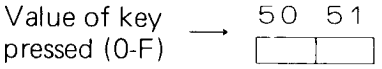
B) Press RESET, 2, RUN to start the program. (Note the mode.)

C) Press a key for the value you want to convert.

D) Check the answer like this:–



The test for a carry is at 0C-10. It distinguishes 0-9 from A-F. The program has two different ends, 11 and 1A. Since you started it with RESET, 2, RUN check the binary LEDs (at step C above) to see where it stopped – 11 if no carry, or 1A if carry.



Press RESET, 2, RUN each time you want to change the value you wish to convert.

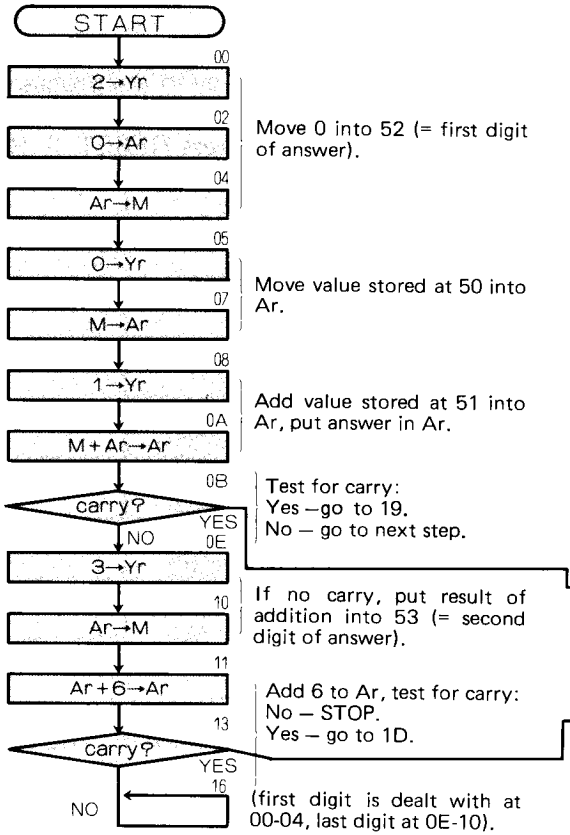
No.36 Decimal Addition — One Digit + One Digit = Two Digits

This program adds two decimal digits (0-9) (from 50/51) and puts the answer in 52/53.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	MA	5
08	TIY	A
09	⟨1⟩	1
0A	M+	6
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨9⟩	9
0E	TIY	A
0F	⟨3⟩	3
10	AM	4
11	AIA	9
12	⟨6⟩	6
13	JUMP	F
14	⟨1⟩	1
15	⟨D⟩	D
16	JUMP	F
17	⟨1⟩	1
18	⟨6⟩	6
19	AIA	9
1A	⟨6⟩	6
1B	TIY	A
1C	⟨3⟩	3
1D	AM	4
1E	TIY	A
1F	⟨2⟩	2
20	TIA	8
21	⟨1⟩	1
22	AM	4
23	JUMP	F
24	⟨2⟩	2
25	⟨3⟩	3

FLOWCHART



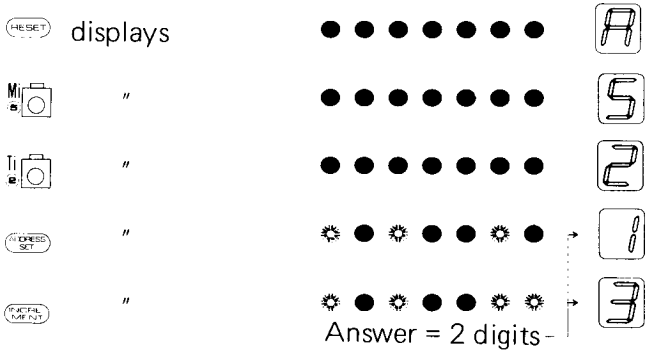
Study the logic. Carries of two different kinds can arise. At 0A it is a hex carry, that is, answer greater than F. At 12 it is a decimal carry that is, answer greater than 9. In both cases the first digit of the answer is 1.

A) Key in the program and check it.

B) Load two DECIMAL numbers at 50/51.
For example RESET Mi 7 ADDRESS SET Sd 6 INCRE MENT Fa INCRE MENT

7 + 6 = ?

C) Press RESET, 2, RUN to start the program.



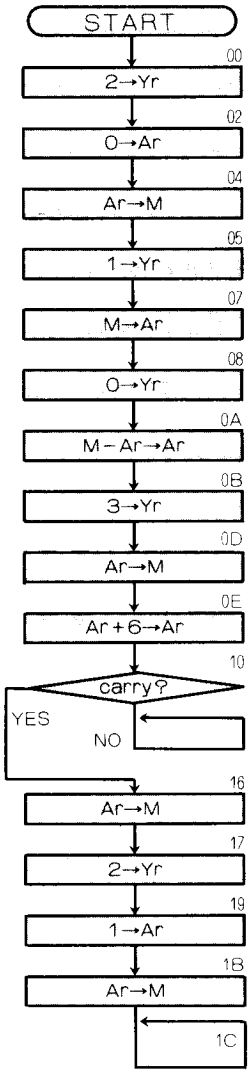
No.37 Hex Subtraction with Decimal Conversion — One Digit Minus One Digit

One hex number (51) is subtracted from another (50) and the answer is decimal-converted and put in 52/53.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨1⟩	1
07	MA	5
08	TIY	A
09	⟨0⟩	0
0A	M−	7
0B	TIY	A
0C	⟨3⟩	3
0D	AM	4
0E	AIA	9
0F	⟨6⟩	6
10	JUMP	F
11	⟨1⟩	1
12	⟨6⟩	6
13	JUMP	F
14	⟨1⟩	1
15	⟨3⟩	3
16	AM	4
17	TIY	A
18	⟨2⟩	2
19	TIA	8
1A	⟨1⟩	1
1B	AM	4
1C	JUMP	F
1D	⟨1⟩	1
1E	⟨C⟩	C

FLOWCHART



Move 0 to 52 (= first digit of answer).

Move contents of 51 to Ar.

Subtract contents of Ar from 50; put answer in Ar.

Put answer in 53.

Add 6 to Ar, test for carry: If No — STOP (first digit dealt with at 00-04, second at 08-0D), If Yes — go to 16.

If carry — put answer in 53 (already pointed to at 0B).

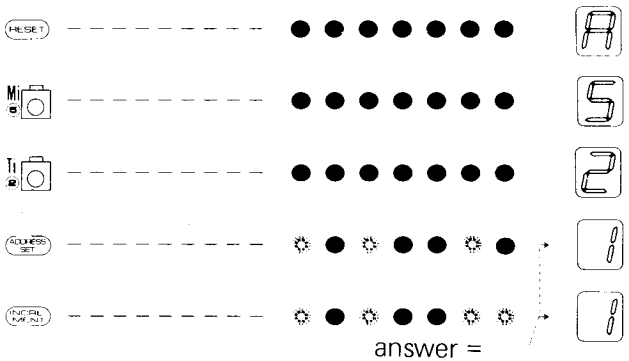
Put 1 in 52 (= first digit of answer; last digit dealt with at 16).

A) Key in the program and check it.

B) Load hex numbers into 50 and 51.
(50 MUST be greater than 51)

Example:

C) Press RESET, 2, RUN to start the program.
Read out answers like this:—



Decimal conversion is made at 0E-12 where 6 is added to Ar and the answer tested for carries.
Here is the memory map:—

address 50 51 52 53

○ − △ = □□

hex — The answer is a 2-digit decimal number.

No.38 Hex Addition with Decimal Conversion— One Digit + One Digit = Two Digits

This program adds two hex numbers together and converts the answer to decimal.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	MA	5
08	TIY	A
09	⟨1⟩	1
0A	M+	6
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨9⟩	9
0E	TIY	A
0F	⟨3⟩	3
10	AM	4
11	AIA	9
12	⟨6⟩	6
13	JUMP	F
14	⟨2⟩	2
15	⟨0⟩	0
16	JUMP	F
17	⟨1⟩	1
18	⟨6⟩	6
19	TIY	A
1A	⟨3⟩	3
1B	AIA	9
1C	⟨6⟩	6
1D	JUMP	F
1E	⟨3⟩	3
1F	⟨7⟩	7
20	AM	4
21	AIA	9
22	⟨6⟩	6
23	JUMP	F
24	⟨2⟩	2
25	⟨E⟩	E
26	TIY	A

address	command	machine code
27	⟨2⟩	2
28	TIA	8
29	⟨1⟩	1
2A	AM	4
2B	JUMP	F
2C	⟨2⟩	2
2D	⟨B⟩	B
2E	AM	4
2F	TIY	A
30	⟨2⟩	2
31	TIA	8
32	⟨2⟩	2
33	AM	4
34	JUMP	F
35	⟨3⟩	3
36	⟨4⟩	4
37	AIA	9
38	⟨6⟩	6
39	AM	4
3A	AIA	9
3B	⟨6⟩	6
3C	JUMP	F
3D	⟨4⟩	4
3E	⟨2⟩	2
3F	JUMP	F
40	⟨2⟩	2
41	⟨F⟩	F
42	AM	4
43	TIY	A
44	⟨2⟩	2
45	TIA	8
46	⟨3⟩	3
47	AM	4
48	JUMP	F
49	⟨4⟩	4
4A	⟨8⟩	8

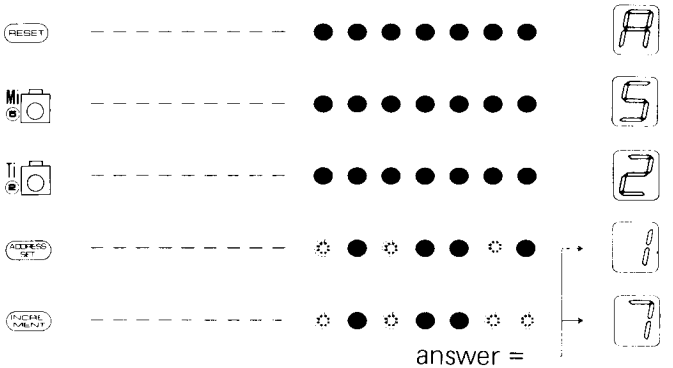
HEX TO DECIMAL CONVERSION

hex	→	decimal	hex	→	decimal
0~9	→	0~9	14~19	→	20~25
A~F	→	10~15	1A~1D	→	26~29
10~13	→	16~19	1E	→	30

A) Key in the program and check it.

B) Load into 50/51 the two numbers to be added. For example, try putting 8 in 50 and 9 in 51.

C) Press RESET, 2, RUN to start the program.
Now press the keys:—



◆ WATCH THIS ONE! Examine the flowchart very carefully.

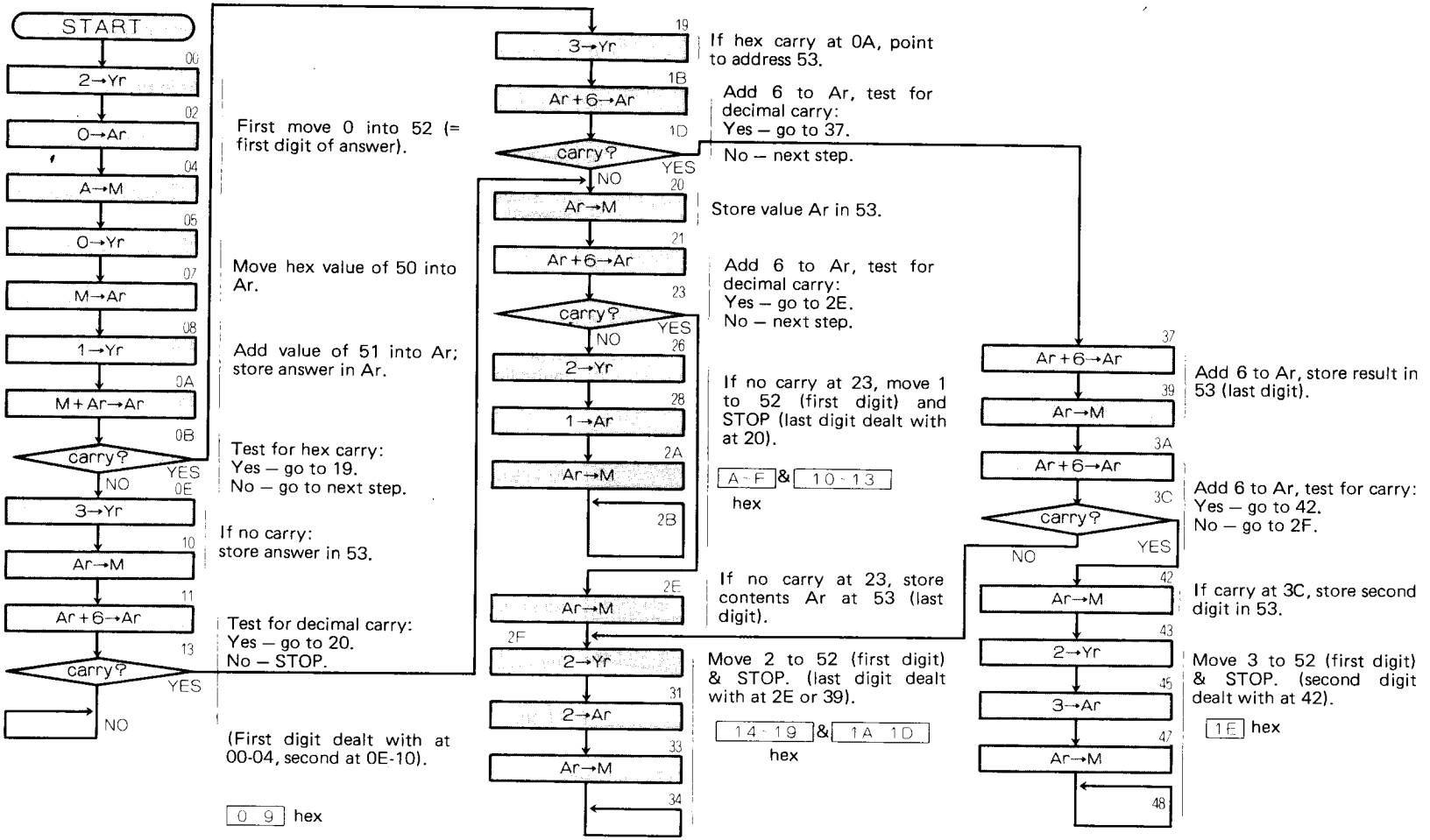
address 50 51 52 53

○ + △ = □□

hex The answer is a decimal number.

[ ] The numbers in the boxes are the answers in hex that are dealt with by that particular part of the program.

FLOWCHART



*** Program No. 38 is the most difficult, so far. Turn to the Appendix for some examples to help you to understand it.

No.39 Hex Multiplication — One Digit × One Digit = Two Digits

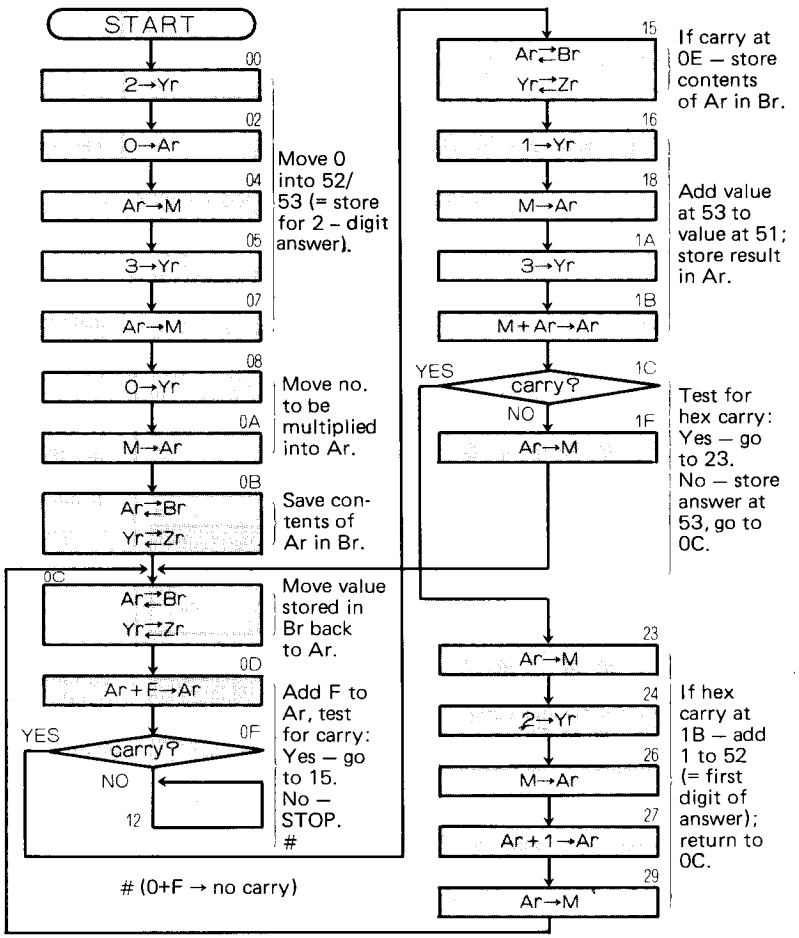
Computers multiply by repeated addition. For example, the answer to 5 x 6 can be obtained by adding 6 five times.
This program adds the contents of 51 the number of times contained in 50 and puts the answer in 52/53.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨3⟩	3
07	AM	4
08	TIY	A
09	⟨0⟩	0
0A	MA	5
0B	CH	2
0C	CH	2
0D	AIA	9
0E	⟨F⟩	F
0F	JUMP	F
10	⟨1⟩	1
11	⟨5⟩	5
12	JUMP	F
13	⟨1⟩	1
14	⟨2⟩	2
15	CH	2
16	TIY	A
17	⟨1⟩	1
18	MA	5
19	TIY	A
1A	⟨3⟩	3
1B	M+	6
1C	JUMP	F
1D	⟨2⟩	2
1E	⟨3⟩	3
1F	AM	4
20	JUMP	F
21	⟨0⟩	0
22	⟨C⟩	C
23	AM	4
24	TIY	A
25	⟨2⟩	2

address	command	machine code
26	MA	5
27	AIA	9
28	⟨1⟩	1
29	AM	4
2A	JUMP	F
2B	⟨0⟩	0
2C	⟨C⟩	C

FLOWCHART



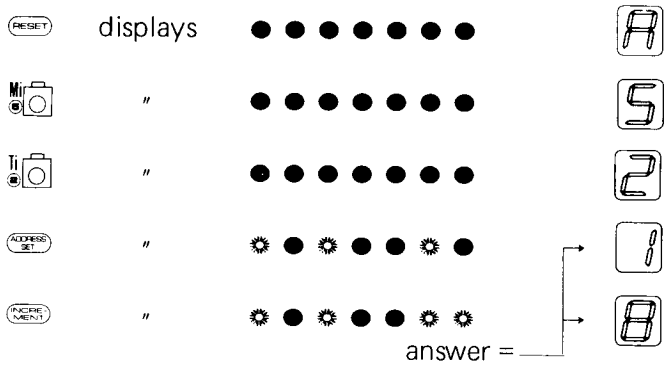
A) Key in the program and check it.

B) Load a hex number into both 50 and 51; the answer will be in 52/53.
Example:

$3 \times 8 =$ ☐

C) Press RESET, 2, RUN to start the program.

D) Get the answer displayed in hex:



Remember — Your answer will be in HEX. 18 hex = 24 decimal.

◆ The program first clears 52 and 53 (sets them to zero). The contents of 51 are added to 53 (with carries to 52 if needed) and at the same time the value of 50 is reduced by 1. The program stops when the value of 50 reaches 0.

address 50 51 52 53

○ × △ = ☐☐
↘ Two digits

For example:

	50	51	52	53
Step 1	3	8	0	0
Step 2	2	8	0	0 + 8 = 8
Step 3	1	8	1	0 (8 + 8 = 10; carry to 52)
Step 4	0	8	1	8 (10 + 8 = 18; carry to 52)

Program stops.

No.40 Decimal Multiplication

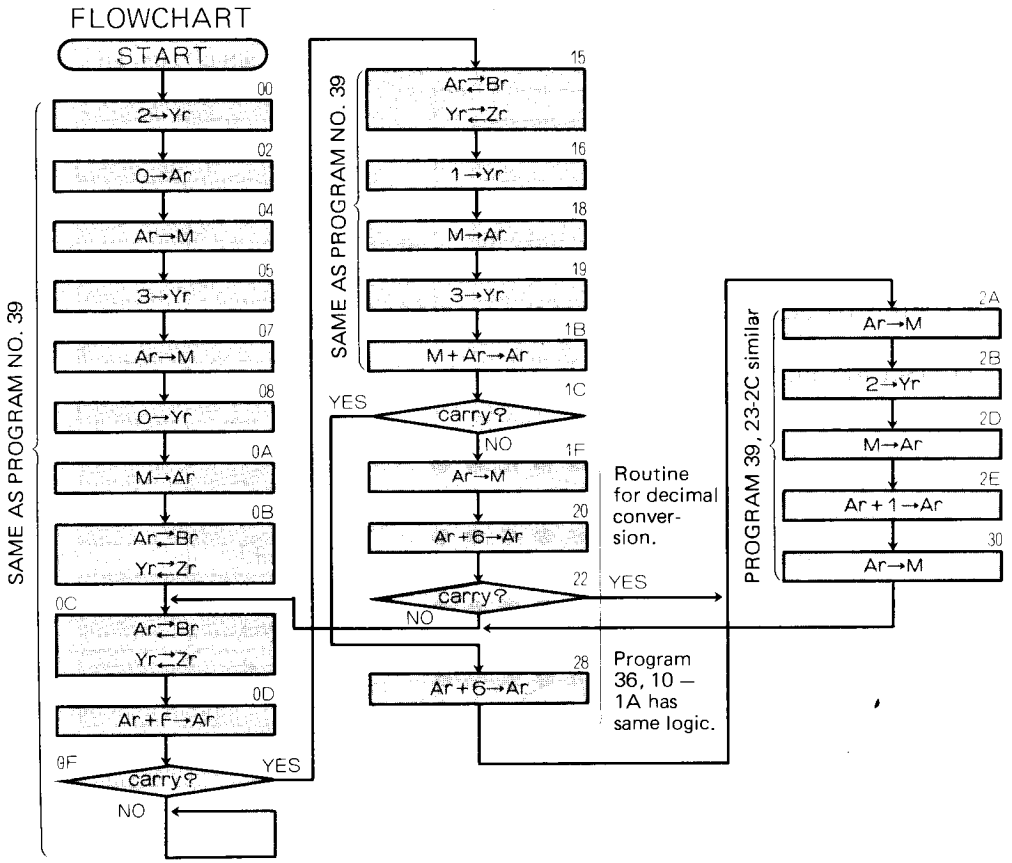
This program multiplies the value of 50 by the value of 51, like program 39. But here the result is converted to decimal before storing in 52/53.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨3⟩	3
07	AM	4
08	TIY	A
09	⟨0⟩	0
0A	MA	5
0B	CH	2
0C	CH	2
0D	AIA	9
0E	⟨F⟩	F
0F	JUMP	F
10	⟨1⟩	1
11	⟨5⟩	5
12	JUMP	F
13	⟨1⟩	1
14	⟨2⟩	2
15	CH	2
16	TIY	A
17	⟨1⟩	1
18	MA	5
19	TIY	A
1A	⟨3⟩	3
1B	M+	6

address	command	machine code
1C	JUMP	F
1D	⟨2⟩	2
1E	⟨8⟩	8
1F	AM	4
20	AIA	9
21	⟨6⟩	6
22	JUMP	F
23	⟨2⟩	2
24	⟨A⟩	A
25	JUMP	F
26	⟨0⟩	0
27	⟨C⟩	C
28	AIA	9
29	⟨6⟩	6
2A	AM	4
2B	TIY	A
2C	⟨2⟩	2
2D	MA	5
2E	AIA	9
2F	⟨1⟩	1
30	AM	4
31	JUMP	F
32	⟨0⟩	0
33	⟨C⟩	C

FLOWCHART



A) Key in the program and check it.

B) Load DECIMAL figures into 50/51.
Try 4 x 6 for example:—

C) Press RESET, 2, RUN to start the program.

D) 

This is the answer you should get on the HEX. LED when you press the keys shown.

Almost all of the program consists of sections taken out of previous programs (36, 39) — see remarks within flowchart.

Address 50 Address 51 Address 52 Address 53

The program contains a routine for decimal conversion which comes from program 36, 10-1A.

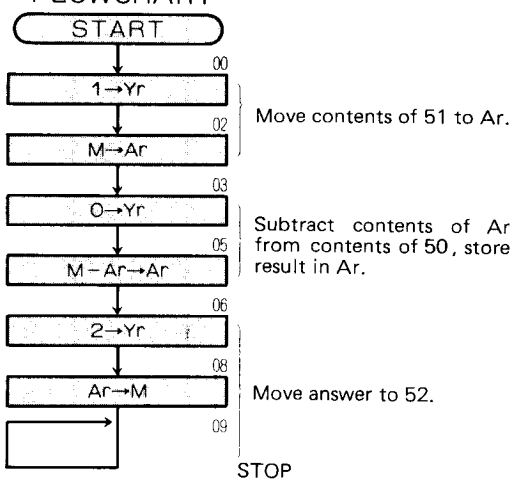
No.41 Decimal Subtraction — One Digit Minus One Digit

In this program the decimal contents of 51 are subtracted from 50 and the answer put into 52.

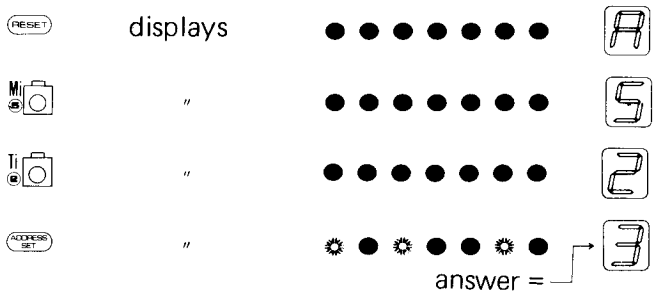
PROGRAM

address	command	machine code
00	TIY	A
01	⟨1⟩	1
02	MA	5
03	TIY	A
04	⟨0⟩	0
05	M−	7
06	TIY	A
07	⟨2⟩	2
08	AM	4
09	JUMP	F
0A	⟨0⟩	0
0B	⟨9⟩	9

FLOWCHART



D) The answer can be read from 52:—



address 50 51 52

○ − △ = □

A) Key in the program and check it.

B) Put DECIMAL figures into 50/51. 50 MUST be larger.

Example: 9 − 6 = □

C) Press RESET, 2, RUN to start the program.

No.42 Division — One Digit Divided by One Digit

Division can be performed by repeated subtractions. To obtain the answer to 8 ÷ 2 we can subtract 2 from 8, four times. If there is a "remainder", this is discarded. In this program the divisor (51) is subtracted from the dividend (50) repeatedly until the overflow occurs, i.e. the divisor will not go into the dividend any more.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨2⟩	2
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	MA	5
08	CH	2
09	TIY	A
0A	⟨1⟩	1
0B	MA	5
0C	TIY	A
0D	⟨0⟩	0
0E	M−	7
0F	JUMP	F
10	⟨1⟩	1
11	⟨C⟩	C
12	AM	4
13	TIY	A
14	⟨2⟩	2
15	MA	5
16	AIA	9
17	⟨1⟩	1
18	AM	4
19	JUMP	F
1A	⟨0⟩	0
1B	⟨9⟩	9
1C	CH	2
1D	AM	4
1E	JUMP	F
1F	⟨1⟩	1
20	⟨E⟩	E

A) Key in the program and check it.

B) Load figures into 50/51, 51 MUST NOT be zero and 50 must be greater than 51.

C) Press RESET, 2, RUN to start the program.

D) RESET displays A

Mi " 5

Ti " 2

ADDRESS SET " 3

answer = ↗

address 50 51 52

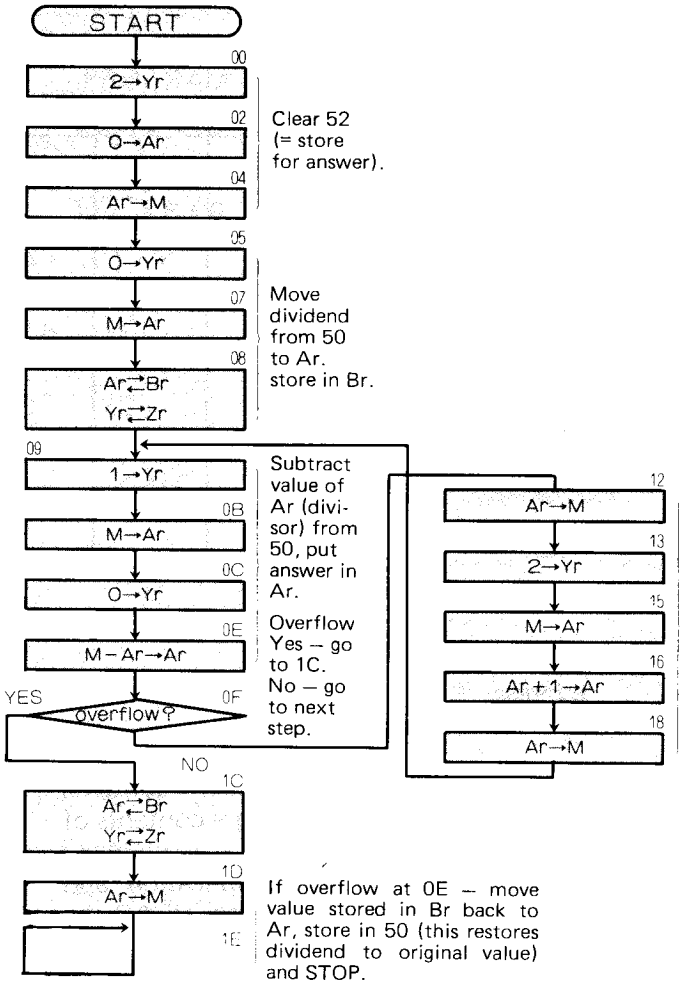
○ ÷ △ = □

The DIVIDEND is the number to be divided = contents of 50. The DIVISOR is doing the dividing = contents of 51.

You have probably noticed that multiplication and division take quite a long time — you can HEAR the micro working. This is because the program has to go through the loop several times.

If the division is not possible — i.e. the divisor is larger than the dividend — the HEX. LED will show "0."

FLOWCHART



Clear 52 (= store for answer).

Move dividend from 50 to Ar. store in Br.

Subtract value of Ar (divisor) from 50, put answer in Ar.

Overflow Yes – go to 1C. No – go to next step.

After a successful subtraction (no overflow) add 1 to 52, return to 09 (52 counts number of successful divisions).

If overflow at 0E – move value stored in Br back to Ar, store in 50 (this restores dividend to original value) and STOP.

(3) Group 3 Commands

There are nine instructions in this group.

They are: CY, CIA, CIY, CAL RSTO, CAL SETR, CAL RSTR, CAL CMPL, CAL DSPR, CAL SHTS.

Each one will be introduced and explained by means of programs.

machine code 1st digit	2nd	command name	machine code 1st digit	2nd	command name
E	0	CAL RSTO	E	8	CAL ERRS
E	1	CAL SETR	E	9	CAL SHTS
E	2	CAL RSTR	E	A	CAL LONS
E	3	NOT USED	E	B	CAL SUND
E	4	CAL CMPL	E	C	CAL TIMR
E	5	CAL CHNG	E	D	CAL DSPR
E	6	CAL SIFT	E	E	CAL DEM-
E	7	CAL ENDS	E	F	CAL DEM+

The CAL command is executed when the FLAG is 1, but not when it is 0 — just like JUMP.

CIY COMMAND

This command compares the value of Yr with the value following the command. If the two values are the same, the FLAG is set to 0. If they are different the FLAG is set to 1. There is an example of what it looks like on the next page.

CAL SETR COMMAND

This command lights up the binary LED pointed to by the value of Yr. Yr is loaded with the required value before CAL SETR is executed. Yr may have any of the values 0-6 (the binary LEDs are numbered 0-6 below each LED).

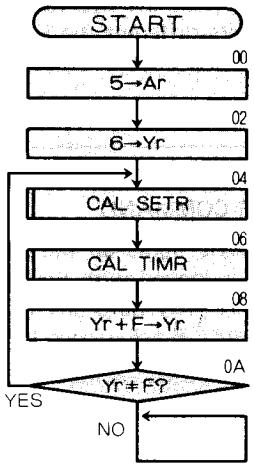
No.43 Use of CIY and CAL SETR

In this program, binary LEDs are turned on from left to right and remain lighted.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨6⟩	6
04	CAL	E
05	SETR	1
06	CAL	E
07	TIMR	C
08	AIY	B
09	⟨F⟩	F
0A	CIY	D
0B	⟨F⟩	F
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4
0F	JUMP	F
10	⟨0⟩	0
11	⟨F⟩	F

FLOWCHART



Set the timer value to 0.6 seconds.

Point to the LH binary LED (6).

Light the binary LED pointed to by Yr.

Start the timer.

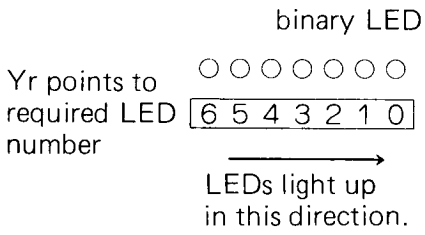
Deduct 1 from Yr*.

Test if Yr ≠ F.
If No, STOP (0+F=F; LED 0 has just been lighted).
If Yes, return to 04.

* Deduct 1 from Yr (and so point to the next LED to the right).

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.



The CIY test is shown on the flowchart as:—

Yr ≠ n ? where n is any number (0 – F).

In other words, "is Yr not-equal to n ? ". If Yr is in fact equal to n, the answer is "no, Yr is NOT not-equal to n. It is EQUAL to n". This is called a double negative.

No.44 Use of CAL RSTR

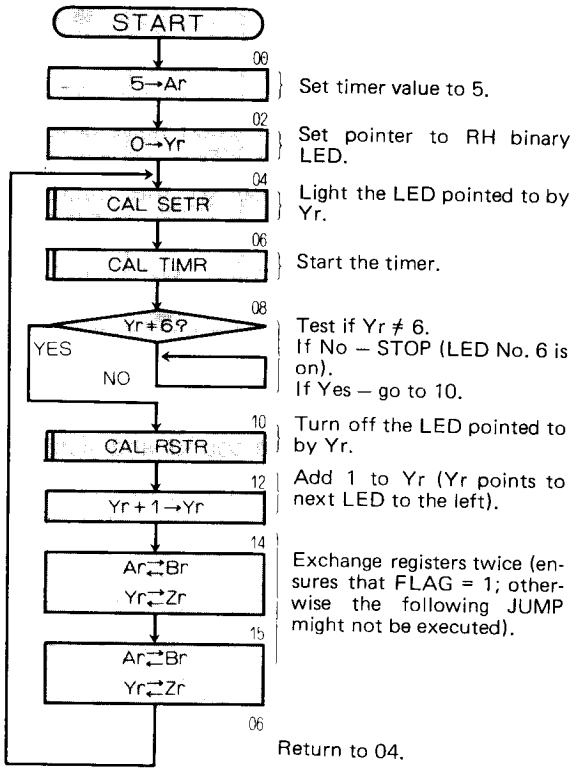
In this program, binary LEDs are turned on and off from right to left.

CAL RSTR COMMAND (CAL) ReSet port R)
This command calls in a subroutine which turns off the binary LED pointed to by Yr. It has exactly the opposite effect to CAL SETR, which was explained under program 43.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨0⟩	0
04	[CAL	E
05	[SETR	1
06	[CAL	E
07	[TIMR	C
08	CIY	D
09	⟨6⟩	6
0A	JUMP	F
0B	⟨1⟩	1
0C	⟨0⟩	0
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨D⟩	D
10	[CAL	E
11	[RSTR	2
12	AIY	B
13	⟨1⟩	1
14	CH	2
15	CH	2
16	JUMP	F
17	⟨0⟩	0
18	⟨4⟩	4

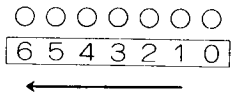
FLOWCHART



This program lights up each binary LED in turn, starting at the right (0). Only one light is on at a time. At the end, LED 6 is left on.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.



direction of LED lighting

No.45 Use of CAL SHTS

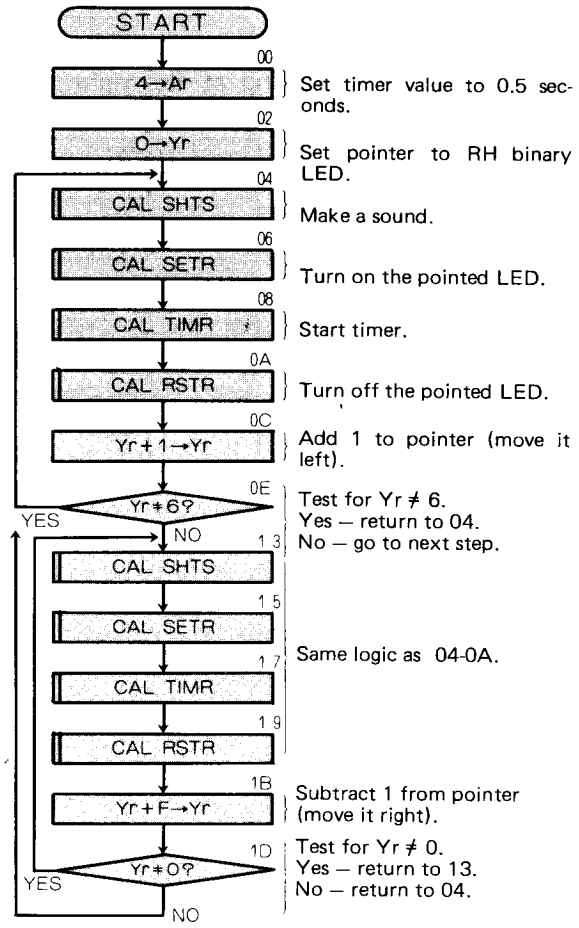
Binary LEDs are lighted one at a time in both directions (with sound effect).

CAL SHTS COMMAND (CALl SHorT Sound)
Each time this command is executed a short sound is generated through port 3. You will learn more about this command later on. Let's call it a BLIP.

PROGRAM

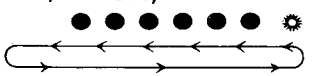
address	command	machine code
00	TIA	8
01	⟨4⟩	4
02	TIY	A
03	⟨0⟩	0
04	⌈CAL	E
05	⌊SHTS	9
06	⌈CAL	E
07	⌊SETR	1
08	⌈CAL	E
09	⌊TIMR	C
0A	⌈CAL	E
0B	⌊RSTR	2
0C	AIY	B
0D	⟨1⟩	1
0E	CIY	D
0F	⟨6⟩	6
10	JUMP	F
11	⟨0⟩	0
12	⟨4⟩	4
13	⌈CAL	E
14	⌊SHTS	9
15	⌈CAL	E
16	⌊SETR	1
17	⌈CAL	E
18	⌊TIMR	C
19	⌈CAL	E
1A	⌊RSTR	2
1B	AIY	B
1C	⟨F⟩	F
1D	CIY	D
1E	⟨0⟩	0
1F	JUMP	F
20	⟨1⟩	1
21	⟨3⟩	3
22	JUMP	F
23	⟨0⟩	0
24	⟨4⟩	4

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1 RUN, to start the program.



Before each light is turned on, a short sound is generated. This sound also turns on binary LED 3, because in the micro the sound is generated via the same port as LED 3.

This light goes on when the sound is generated.

The sound and lighting continue until you stop the program by pressing RESET.

No.46 Use of CAL RSTO

This program turns the HEX. LED on and off.

CAL RSTO COMMAND (CALI ReSeT port 0)
This command turns off the HEX. LED. It is not dependent on the values in Ar or Yr.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨1⟩	1
02	MA	5
03	TIY	A
04	⟨0⟩	0
05	M−	7
06	JUMP	F
07	⟨0⟩	0
08	⟨D⟩	D
09	AO	1
0A	JUMP	F
0B	⟨0⟩	0
0C	⟨A⟩	A
0D	TIY	A
0E	⟨0⟩	0
0F	TIA	8
10	⟨E⟩	E
11	AO	1
12	AIY	B
13	⟨1⟩	1
14	TIA	8
15	⟨2⟩	2
16	⌜CAL	E
17	⌞TIMR	C
18	⌜CAL	E
19	⌞RSTO	0
1A	CIY	D
1B	⟨A⟩	A

address	command	machine code
1C	JUMP	F
1D	⟨2⟩	2
1E	⟨2⟩	2
1F	JUMP	F
20	⟨1⟩	1
21	⟨F⟩	F
22	⌜CAL	E
23	⌞TIMR	C
24	JUMP	F
25	⟨0⟩	0
26	⟨F⟩	F

A) Key in the program and check it.

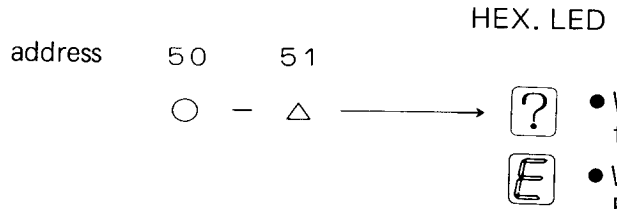
B) Load hex data into 50 and 51. To cause an error condition, make the contents of 51 greater than those of 50.

Example: 8−9=?

C) Press RESET, 1, RUN to start. E will flash on and off.

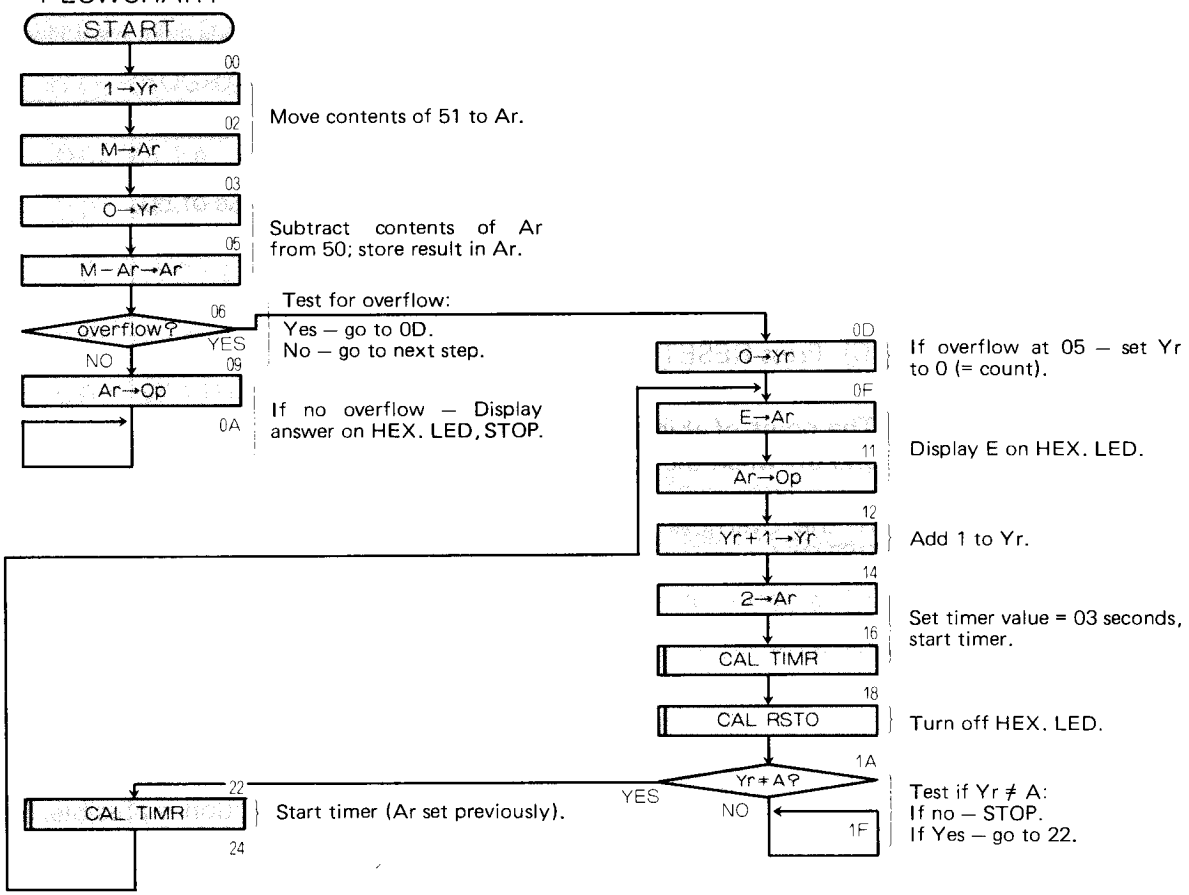
The point of this exercise is to make the HEX. LED flash on and off when the subtraction command causes an overflow.

If the overflow occurs, E flashes on (AO) and off. (CAL RSTO) ten times.



- When subtraction is possible, the answer is displayed.
- When subtraction is not possible, E flashes on and off.

FLOWCHART


START
00
1→Yr
02
M→Ar
03
0→Yr
05
M−Ar→Ar
06
overflow?
YES
NO
09
Ar→Op
0A
Move contents of 51 to Ar.
Subtract contents of Ar from 50; store result in Ar.
Test for overflow:
Yes — go to 0D.
No — go to next step.
If no overflow — Display answer on HEX. LED, STOP.
0D
0→Yr
0F
E→Ar
11
Ar→Op
12
Yr+1→Yr
14
2→Ar
16
CAL TIMR
18
CAL RSTO
1A
Yr≠A?
YES
NO
1F
22
CAL TIMR
24
If overflow at 05 — set Yr to 0 (= count).
Display E on HEX. LED.
Add 1 to Yr.
Set timer value = 03 seconds, start timer.
Turn off HEX. LED.
Test if Yr ≠ A:
If no — STOP.
If Yes — go to 22.
Start timer (Ar set previously).

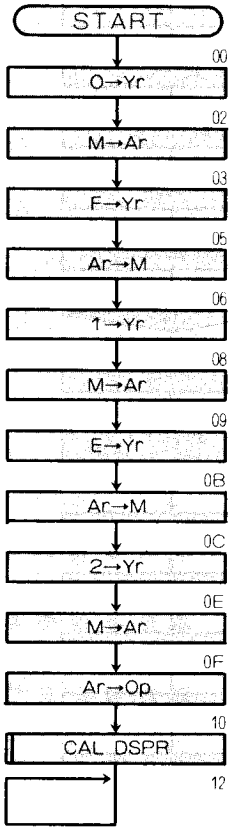
No.47 Use of CAL DSPR

In later programs we will need to display 3-digit results. One digit can be displayed on the HEX. LED; two others can be displayed in binary on the binary LEDs by using CAL DSPR.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	TIY	A
04	⟨F⟩	F
05	AM	4
06	TIY	A
07	⟨1⟩	1
08	MA	5
09	TIY	A
0A	⟨E⟩	E
0B	AM	4
0C	TIY	A
0D	⟨2⟩	2
0E	MA	5
0F	AO	1
10	⌜CAL	E
11	⌞DSPR	D
12	JUMP	F
13	⟨1⟩	1
14	⟨2⟩	2

FLOWCHART



Move contents of Ar to 50.

Move contents of Ar to 5F, in order to display it in the LH part of the binary LEDs.

Move contents of 51 to Ar.

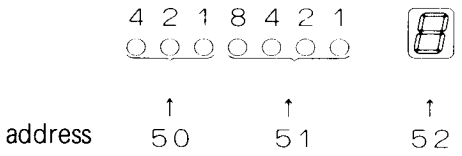
Move contents of Ar to 5E, in order to display it in the RH part of the binary LEDs.

Move contents of 52 into Ar.

Display contents of Ar on HEX. LED.

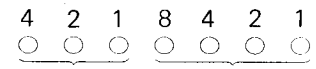
Translate contents of 5E and 5F into binary, display them on binary LEDs:— 5E on right, 5F on left, then STOP.

binary LED HEX. LED



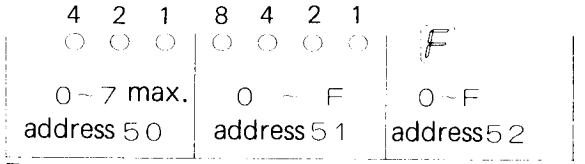
CAL DSPR (CALl DiSPlay on port R)

This command displays the contents of addresses 5E and 5F in binary on the binary LEDs. 5F is displayed on the first three LEDs, 5E on the next four LEDs.



A) Key in the program and check it.

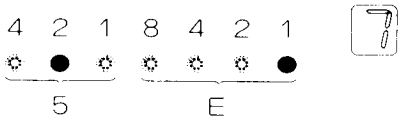
B) Load hex numbers into 50/51/52. The program will be displayed as below.



Example:

address. 50 ⟨5⟩ 、 51 ⟨E⟩ 、 52 ⟨7⟩

C) Press RESET, 1, RUN to start the program.



This is the third CALL routine we have tried that affects the binary LEDs. The other two are CAL SETR and CAL RSTR.

No.48 Use of CAL CMPL

This program sorts two numbers into ascending order.

CAL CMPL COMMAND (CALI CoMPLement)

This command replaces the value in Ar with its complement. The COMPLEMENT of a number is the answer you get if you subtract the number from F. NUMBER + COMPLEMENT = F. The following are numbers with their complements:

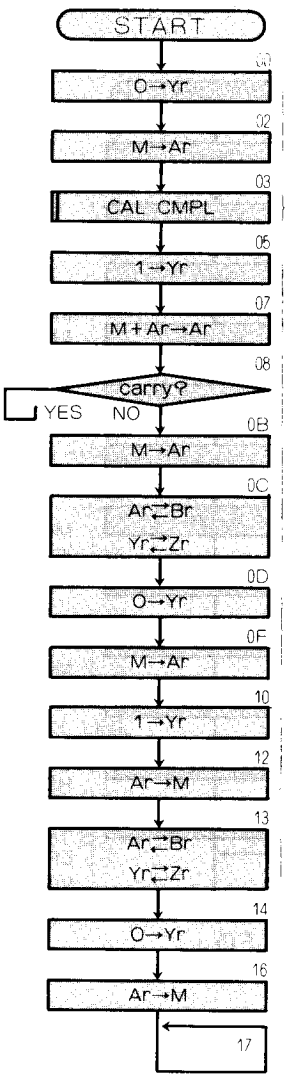
Ar before CMPL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ar after CMPL	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	⌜CAL	E
04	⌞CMPL	4
05	TIY	A
06	⟨1⟩	1
07	M+	6
08	JUMP	F
09	⟨0⟩	0
0A	⟨8⟩	8
0B	MA	5
0C	CH	2
0D	TIY	A
0E	⟨0⟩	0
0F	MA	5
10	TIY	A

address	command	machine code
11	⟨1⟩	1
12	AM	4
13	CH	2
14	TIY	A
15	⟨0⟩	0
16	AM	4
17	JUMP	F
18	⟨1⟩	1
19	⟨7⟩	7

FLOWCHART



Move contents of 50 into Ar.

Move complement of Ar into Ar.

Add contents of 51 to Ar. put result in Ar.

Test for carry:
If carry — STOP.*
If no carry — go to next step.

Store contents of 51 in Br.

Move contents of 50 to Ar.

Move contents of Ar (50) to 51.

Recall old contents of 51 from Br.

Move old contents of 51 to 50.

* If carry — STOP (complement of Ar (50) is greater than complement of value in 51; so value of 50 must be less than value of 51).

This program is our first attempt at a SORT. It is based on the principle that if the complement of a number A is greater than the complement of a number B then A must be smaller than B.
This program checks the complements of the numbers you enter into 50/51 to see which number is greater. Then it displays the smaller number first.

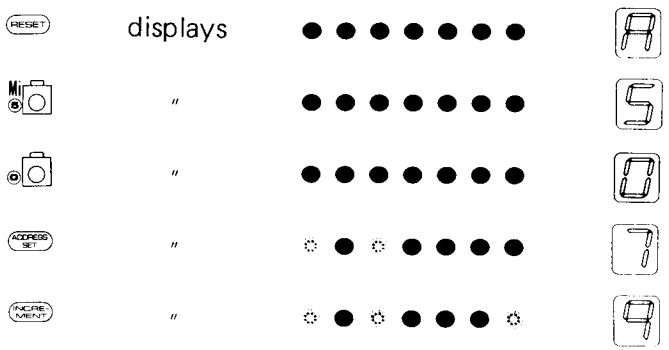
A) Key in the program and check it.

B) Load two numbers into 50 and 51.

Example: 9 in 50, 7 in 51, like this:—

C) Press RESET, 2, RUN to start the program.

D) Read addresses 50 and 51 to check that the smaller number now comes first:



Notes:

No.49 Use of CIA

This is an unusual kind of program because we load into memory not only the data, but also the commands.

memory address

50 → any number 0-9	52 → any number 0-9
51 → B (minus)	53 → E (=)

If the number in 52 is greater than that in 50, E will flash on and off on the HEX. LED. The same applies if the values in 51 and 53 are not B and E. Otherwise, the answer is stored in 54.

CIA COMMAND (Compare with Ar)

This command compares the contents of Ar with the value following the CIA command. If they are equal, the FLAG is set to 0. If they are not equal, the FLAG is set to 1.
The CAL and JUMP commands are used immediately after CIA. Whether they are executed or not depends on the setting of the FLAG – 1 = execute, 0 = don't execute.

	command word	machine code
command	CIA	C
data	⟨E⟩	E

A) Key in the program and check it.

B) Load some data into 50/51/52/53 for example, 9, B, 5, E:–

9 B(–) 5 E(=) answer

C) Press RESET, 2, RUN to start. The program displays: 4

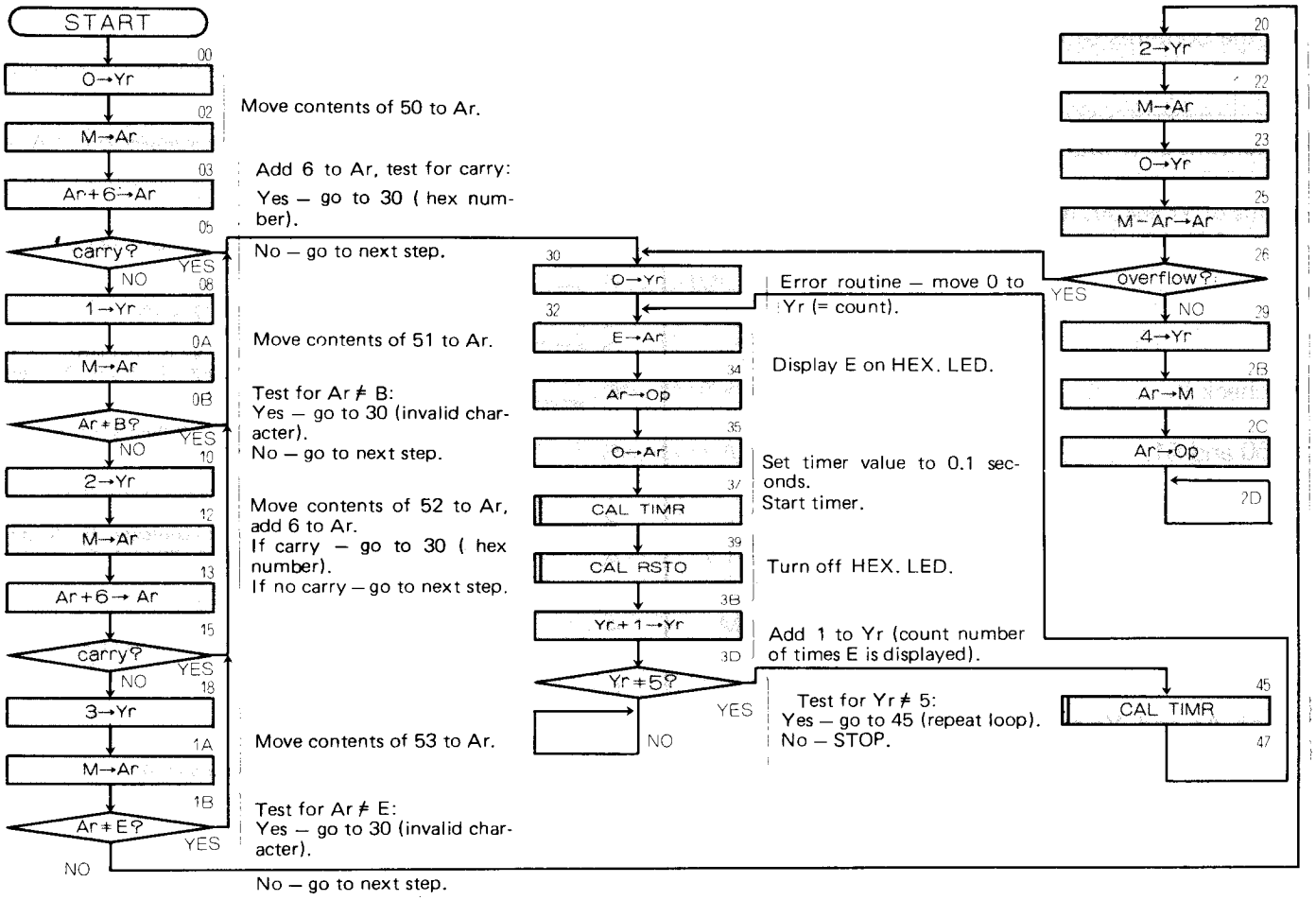
D) The answer 4 should show on the HEX. LED.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	AIA	9
04	⟨6⟩	6
05	JUMP	F
06	⟨3⟩	3
07	⟨0⟩	0
08	TIY	A
09	⟨1⟩	1
0A	MA	5
0B	CIA	C
0C	⟨B⟩	B
0D	JUMP	F
0E	⟨3⟩	3
0F	⟨0⟩	0
10	TIY	A
11	⟨2⟩	2
12	MA	5
13	AIA	9
14	⟨6⟩	6
15	JUMP	F
16	⟨3⟩	3
17	⟨0⟩	0
18	TIY	A
19	⟨3⟩	3
1A	MA	5
1B	CIA	C
1C	⟨E⟩	E
1D	JUMP	F
1E	⟨3⟩	3
1F	⟨0⟩	0
20	TIY	A
21	⟨2⟩	2
22	MA	5
23	TIY	A
24	⟨0⟩	0

address	command	machine code
25	M–	7
26	JUMP	F
27	⟨3⟩	3
28	⟨0⟩	0
29	TIY	A
2A	⟨4⟩	4
2B	AM	4
2C	AO	1
2D	JUMP	F
2E	⟨2⟩	2
2F	⟨D⟩	D
30	TIY	A
31	⟨0⟩	0
32	TIA	8
33	⟨E⟩	E
34	AO	1
35	TIA	8
36	⟨0⟩	0
37	⌈CAL	E
38	⌊TIMR	C
39	⌈CAL	E
3A	⌊RSTO	0
3B	AIY	B
3C	⟨1⟩	1
3D	CIY	D
3E	⟨5⟩	5
3F	JUMP	F
40	⟨4⟩	4
41	⟨5⟩	5
42	JUMP	F
43	⟨4⟩	4
44	⟨2⟩	2
45	⌈CAL	E
46	⌊TIMR	C
47	JUMP	F
48	⟨3⟩	3
49	⟨2⟩	2

FLOWCHART


START
00
0→Yr
02
M→Ar
03
Ar+6→Ar
05
carry?
YES
NO
08
1→Yr
0A
M→Ar
0B
Ar≠B?
YES
NO
10
2→Yr
12
M→Ar
13
Ar+6→Ar
15
carry?
YES
NO
18
3→Yr
1A
M→Ar
1B
Ar≠E?
YES
NO
Move contents of 50 to Ar.
Add 6 to Ar, test for carry:
Yes – go to 30 (hex number).
No – go to next step.
Move contents of 51 to Ar.
Test for Ar ≠ B:
Yes – go to 30 (invalid character).
No – go to next step.
Move contents of 52 to Ar, add 6 to Ar.
If carry – go to 30 (hex number).
If no carry – go to next step.
Move contents of 53 to Ar.
Test for Ar ≠ E:
Yes – go to 30 (invalid character).
No – go to next step.
30
0→Yr
32
E→Ar
34
Ar→Op
35
0→Ar
37
CAL TIMR
39
CAL RSTO
3B
Yr+1→Yr
3D
Yr≠5?
YES
NO
Error routine – move 0 to Yr (= count).
Display E on HEX. LED.
Set timer value to 0.1 seconds.
Start timer.
Turn off HEX. LED.
Add 1 to Yr (count number of times E is displayed).
Test for Yr ≠ 5:
Yes – go to 45 (repeat loop).
No – STOP.
20
2→Yr
22
M→Ar
23
0→Yr
25
M–Ar→Ar
26
overflow?
YES
NO
29
4→Yr
2B
Ar→M
2C
Ar→Op
2D
45
CAL TIMR
47
Move contents of 52 to Ar.
Subtract contents of Ar from 50; put result in Ar.
Test for overflow:
Yes – go to 30 (deduction not possible).
No – go to next step.
If answer 0.K., move to 54 and display on HEX. LED.
Start the timer.
Return to 30.

No.50 Use of CY

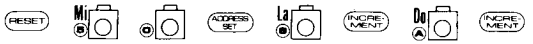
This program compares relative sizes of data in memory and shows the address of memory with the greater number on the binary LEDs and the contents of that address on the HEX. LED.

CY COMMAND (exChange contents of Yr with Ar)

Flowchart symbol: Ar⇄Yr This command is used to exchange the contents of Yr and Ar. It is useful for transferring the contents of Yr into memory, as in program number 50.

A) Key in the program and check it.

B) Load two numbers into 50 and 51.
Example: 8 into 50, A into 51.



C) Press RESET, 1, RUN to start the program.

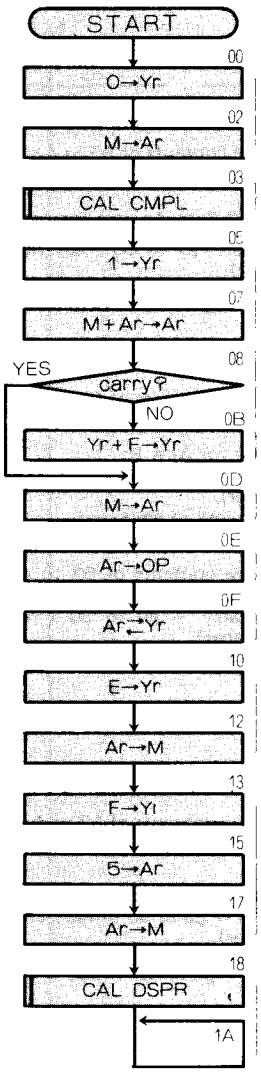
Answer should be:— ☼ ● ☼ ● ● ● ☼

CY causes the contents of Ar and Yr to be exchanged.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	⌜CAL	E
04	⌞CMPL	4
05	TIY	A
06	⟨1⟩	1
07	M+	6
08	JUMP	F
09	⟨0⟩	0
0A	⟨D⟩	D
0B	AIY	B
0C	⟨F⟩	F
0D	MA	5
0E	AO	1
0F	CY	3
10	TIY	A
11	⟨E⟩	E
12	AM	4
13	TIY	A
14	⟨F⟩	F
15	TIA	8
16	⟨5⟩	5
17	AM	4
18	⌜CAL	E
19	⌞DSPR	D
1A	JUMP	F
1B	⟨1⟩	1
1C	⟨A⟩	A

FLOWCHART



Move contents of 50 to Ar.

Move complement of Ar to Ar.

Add contents of 51 to Ar (complement of 50); put result in Ar.

Test for carry:
Yes — go to 0D. *
No — go to next step.

Deduct 1 from Yr sets pointer to 50.

Move contents of pointed memory to Ar.

Display contents of Ar.

Enter contents of Yr into Ar (= pointer to address with larger data).

Move contents of Ar to 5E.

Move 5 to 5F.

Move contents of 5F to higher binary LEDs; move value in 5E to lower binary LEDs, and STOP.

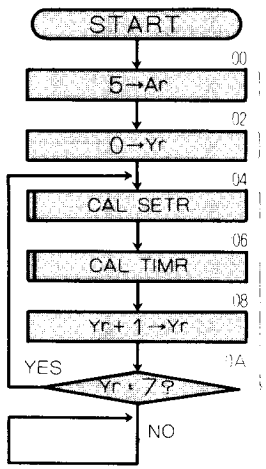
No.51 Turn on Binary LEDs from right to left

This program is similar to no. 43, but it turns on the LEDs in the opposite direction.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨0⟩	0
04	┌CAL	E
05	└SETR	1
06	┌CAL	E
07	└TIMR	C
08	AIY	B
09	⟨1⟩	1
0A	CIY	D
0B	⟨7⟩	7
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4
0F	JUMP	F
10	⟨0⟩	0
11	⟨F⟩	F

FLOWCHART



Set timer value to 0.6 seconds.

Set pointer to RH binary LED.

Turn on the LED pointed to by Yr.

Start timer, add 1 to Yr.

Test if Yr ≠ 7:
Yes – return to 04.
No – STOP.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

All the binary LEDs will be turned on one after the other. (It's nice to do something easy for a change!)

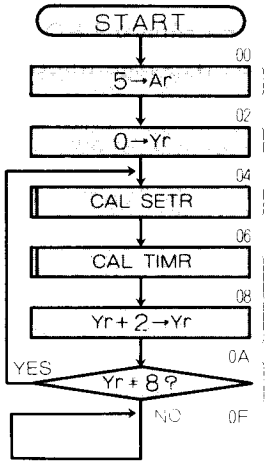
No.52 Turn on Alternate Binary LEDs from right to left

The LEDs light in alternating sequence from right to left.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨0⟩	0
04	⌜CAL	E
05	⌞SETR	1
06	⌜CAL	E
07	⌞TIMR	C
08	AIY	B
09	⟨2⟩	2
0A	CIY	D
0B	⟨8⟩	8
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4
0F	JUMP	F
10	⟨0⟩	0
11	⟨F⟩	F

FLOWCHART



Set timer value to 0.6 seconds.

Set pointer to RH LED.

Turn on pointed LED.

Start timer, add 2 to pointer.

Test for Yr ≠ 8:
Yes – return to 04 and repeat loop.
No – STOP.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

To light only alternate LEDs, the code at 09 is changed to 2.

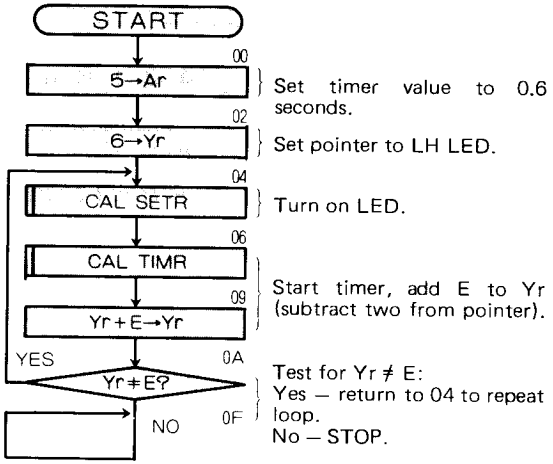
No.53 Turn on Alternate Binary LEDs from left to right

The LEDs light from left to right.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03,	⟨6⟩	6
04	⌜CAL	E
05	⌞SETR	1
06	⌜CAL	E
07	⌞TIMR	C
08	AIY	B
09	⟨E⟩	E
0A	CIY	D
0B	⟨E⟩	E
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨4⟩	4
0F	JUMP	F
10	⟨0⟩	0
11	⟨F⟩	F

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

The only difference in this program is that the pointer (Yr) is reduced by 2 each time round, instead of being increased by 2 as in program 52 and the pointer is set to begin at the left-hand LED.

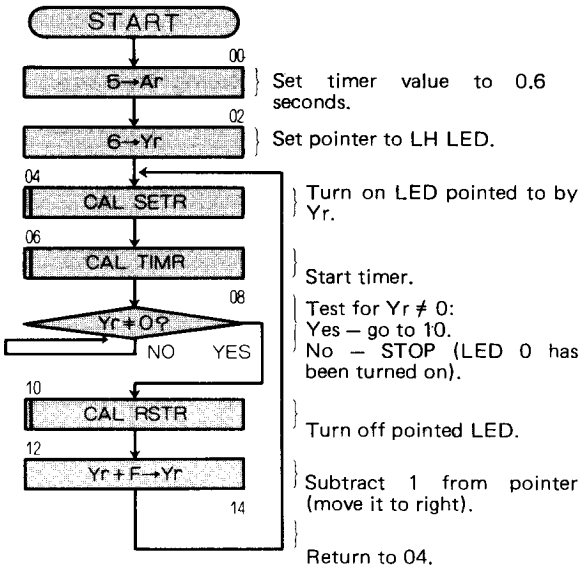
No.54 Turn on Binary LEDs One at a time from left to right

This program lights the LEDs in the opposite direction of program 44.

PROGRAM

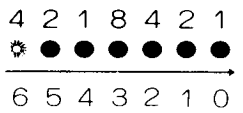
address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨6⟩	6
04	CAL	E
05	SETR	1
06	CAL	E
07	TIMR	C
08	CIY	D
09	⟨0⟩	0
0A	JUMP	F
0B	⟨1⟩	1
0C	⟨0⟩	0
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨D⟩	D
10	CAL	E
11	RSTR	2
12	AIY	B
13	⟨F⟩	F
14	JUMP	F
15	⟨0⟩	0
16	⟨4⟩	4

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.



LEDs are turned on and off one after another beginning at the left.

A small change has made a big difference!

No.55 Turn on Binary LEDs One at a time in both directions (1)

The LEDs light at varying speeds until count reaches 16.
The new feature in this program is that the rate of display slows down as the program runs.

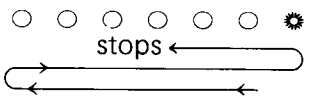
PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨0⟩	0
04	[CAL	E
05	[SETR	1
06	[CAL	E
07	[TIMR	C
08	[CAL	E
09	[RSTR	2
0A	AIA	9
0B	⟨1⟩	1
0C	JUMP	F
0D	⟨0⟩	0
0E	⟨C⟩	C
0F	AIY	B
10	⟨1⟩	1
11	CIY	D
12	⟨6⟩	6
13	JUMP	F
14	⟨0⟩	0
15	⟨4⟩	4
16	[CAL	E
17	[SETR	1
18	[CAL	E
19	[TIMR	C
1A	[CAL	E
1B	[RSTR	2
1C	AIA	9
1D	⟨1⟩	1
1E	JUMP	F
1F	⟨1⟩	1
20	⟨E⟩	E
21	AIY	B
22	⟨F⟩	F
23	CIY	D
24	⟨0⟩	0
25	JUMP	F

address	command	machine code
26	⟨1⟩	1
27	⟨6⟩	6
28	JUMP	F
29	⟨0⟩	0
2A	⟨4⟩	4

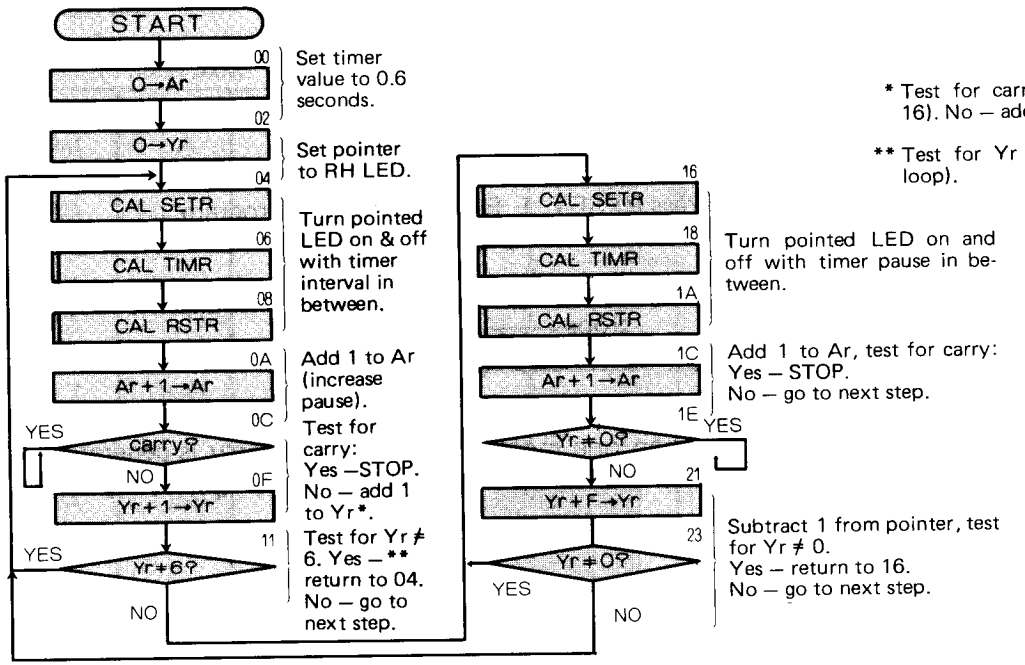
A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.



The program stops when the timer constant (Ar) has reached F — when one more is added to it, a carry is generated. The direction of movement changes twice during the program.

FLOWCHART



* Test for carry: Yes – STOP (Ar has reached 16). No – add 1 to Yr (increase pointer value).

** Test for Yr ≠ 6. Yes – return to 04 (repeat loop).

No.56 Turn on Binary LEDs One at a time in both directions (2)

Number of lighted LED is displayed on the HEX. LED.

PROGRAM

address	command	machine code	address	command	machine code
00	TIA	8	17	⟨0⟩	0
01	⟨5⟩	5	18	⟨7⟩	7
02	CH	2	19	⌜CAL	E
03	TIY	A	1A	⌞SETR	1
04	⟨0⟩	0	1B	AO	1
05	TIA	8	1C	CH	2
06	⟨0⟩	0	1D	⌜CAL	E
07	⌜CAL	E	1E	⌞TIMR	C
08	⌞SETR	1	1F	CH	2
09	AO	1	20	⌜CAL	E
0A	CH	2	21	⌞RSTR	2
0B	⌜CAL	E	22	AIA	9
0C	⌞TIMR	C	23	⟨F⟩	F
0D	CH	2	24	AIY	B
0E	⌜CAL	E	25	⟨F⟩	F
0F	⌞RSTR	2	26	CIY	D
10	AIA	9	27	⟨0⟩	0
11	⟨1⟩	1	28	JUMP	F
12	AIY	B	29	⟨1⟩	1
13	⟨1⟩	1	2A	⟨9⟩	9
14	CIY	D	2B	JUMP	F
15	⟨6⟩	6	2C	⟨0⟩	0
16	JUMP	F	2D	⟨7⟩	7

This program displays on the HEX. LED the number (0-6) of the binary LED currently turned on.

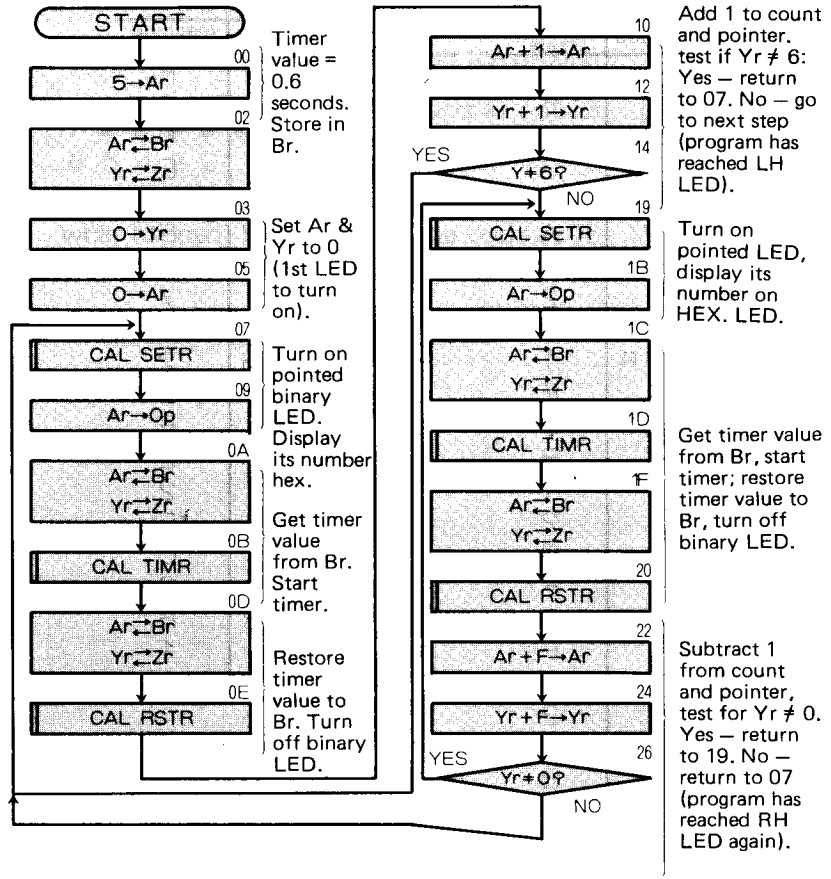
A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

[Ar→Op] This command displays on the HEX. LED the number reached on the binary LEDs.

To stop the program, press RESET.

FLOWCHART



No.57 Turn on Binary LEDs in both directions at once, starting the center

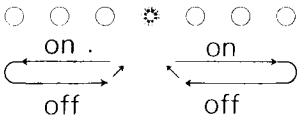
PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨3⟩	3
04	CH	2
05	TIY	A
06	⟨3⟩	3
07	CH	2
08	⌈CAL	E
09	⌊SETR	1
0A	⌈CAL	E
0B	⌊TIMR	C
0C	AIY	B
0D	⟨1⟩	1
0E	CH	2
0F	AIY	B
10	⟨F⟩	F
11	CIY	D
12	⟨F⟩	F
13	JUMP	F
14	⟨2⟩	2
15	⟨3⟩	3
16	AIY	B
17	⟨1⟩	1
18	CH	2
19	AIY	B
1A	⟨F⟩	F
1B	CIY	D
1C	⟨2⟩	2
1D	JUMP	F
1E	⟨2⟩	2
1F	⟨8⟩	8

address	command	machine code
20	JUMP	F
21	⟨0⟩	0
22	⟨0⟩	0
23	⌈CAL	E
24	⌊SETR	1
25	JUMP	F
26	⟨0⟩	0
27	⟨7⟩	7
28	⌈CAL	E
29	⌊RSTR	2
2A	CH	2
2B	⌈CAL	E
2C	⌊RSTR	2
2D	CH	2
2E	⌈CAL	E
2F	⌊TIMR	C
30	CH	2
31	JUMP	F
32	⟨1⟩	1
33	⟨6⟩	6

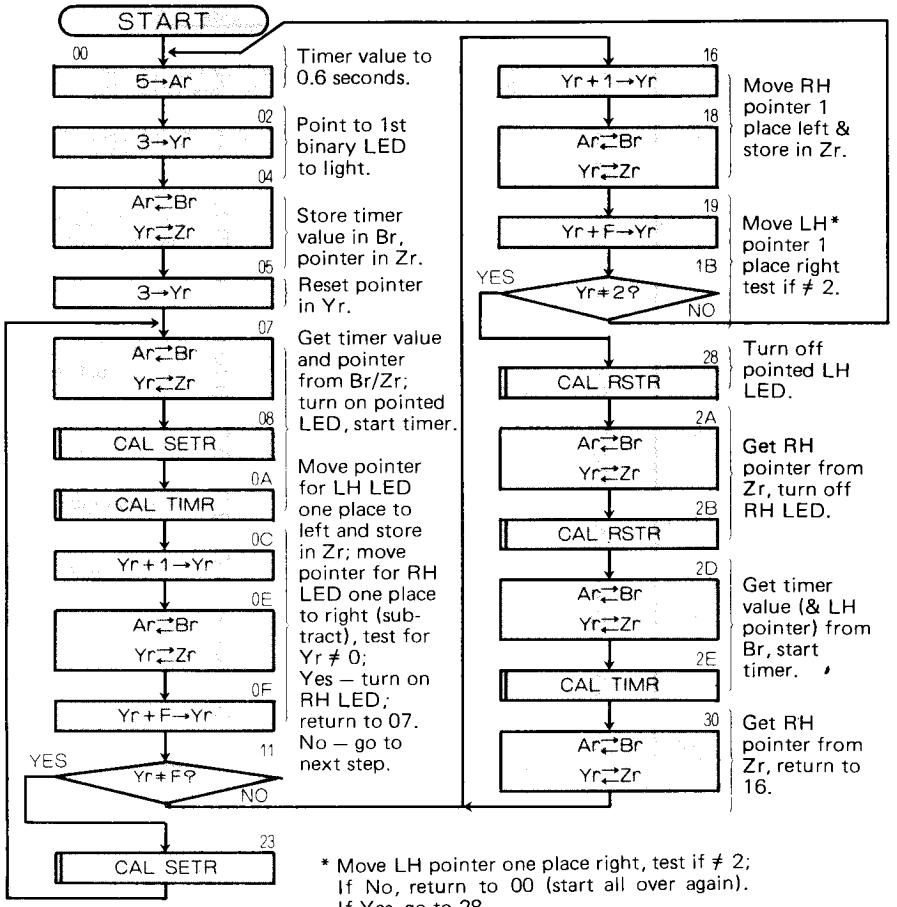
A) Key in the program and check it.

B) Press RESET, 1, RUN to start.
The LEDs turn on and off in this sequence.



The lighting continues until you press RESET.

FLOWCHART



* Move LH pointer one place right, test if ≠ 2;
If No, return to 00 (start all over again).
If Yes, go to 28.

If you can keep track of all that swapping you are doing very well!

No.58 Turn on Binary LEDs in both directions at once, starting at the outside

This program gives a display which looks like a neon sign — it shrinks and expands.

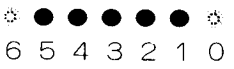
PROGRAM

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	TIY	A
03	⟨0⟩	0
04	CH	2
05	TIY	A
06	⟨6⟩	6
07	CAL	E
08	SETR	1
09	CH	2
0A	CAL	E
0B	SETR	1
0C	CAL	E
0D	TIMR	C
0E	AIY	B
0F	⟨1⟩	1
10	CH	2
11	AIY	B
12	⟨F⟩	F
13	CIY	D
14	⟨2⟩	2
15	JUMP	F
16	⟨0⟩	0
17	⟨7⟩	7
18	AIY	B

address	command	machine code
19	⟨1⟩	1
1A	CH	2
1B	AIY	B
1C	⟨F⟩	F
1D	CIY	D
1E	⟨F⟩	F
1F	JUMP	F
20	⟨2⟩	2
21	⟨5⟩	5
22	JUMP	F
23	⟨0⟩	0
24	⟨0⟩	0
25	CAL	E
26	RSTR	2
27	CH	2
28	CAL	E
29	RSTR	2
2A	CH	2
2B	CAL	E
2C	TIMR	C
2D	CH	2
2E	JUMP	F
2F	⟨1⟩	1
30	⟨8⟩	8

A) Key in the program and check it.

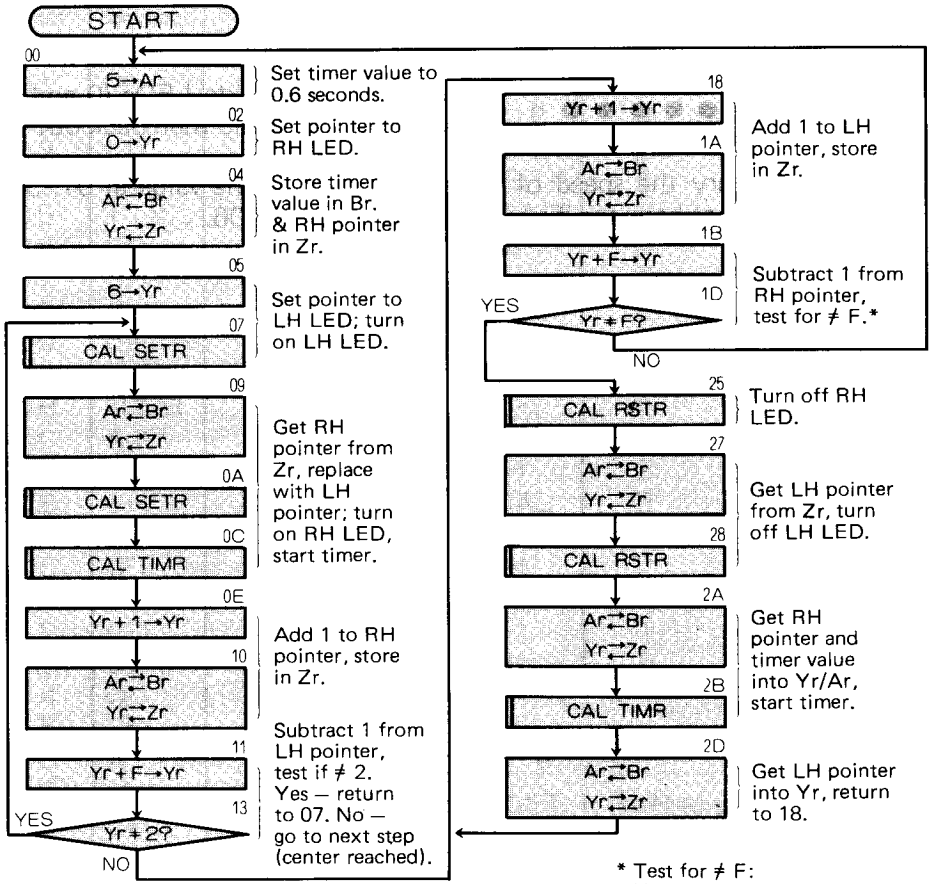
B) Press RESET, 1, RUN to start the program.



display starts at outside LEDs (0 and 6) and works inwards and then back out to 0 and 6.

Vary the speed of change by altering the value at 01. See what happens if you change the value at 06.

FLOWCHART



* Test for ≠ F:
No — go to 00 to start again.
Yes — go to next step.

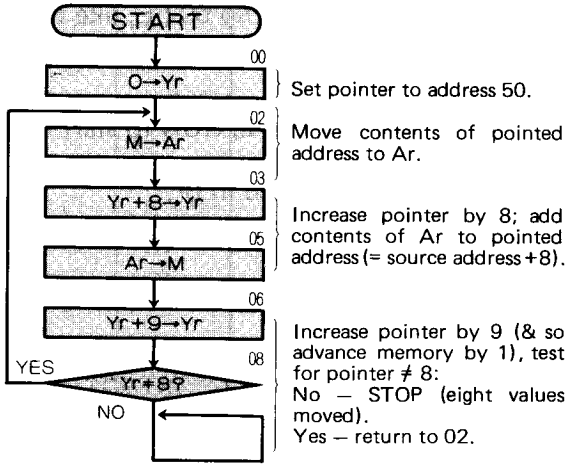
No.59 Transfer Contents of Addresses 50-57 to 58-5F

When you move a character from one location to another you TRANSFER it. Remember that the character is COPIED — when you move it from 50 to Ar, it remains in 50 but a copy is put into Ar. The Microcomputer Trainer has only a small amount of data storage, but this program illustrates a principle which is used in large computers too.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	AIY	B
04	⟨8⟩	8
05	AM	4
06	AIY	B
07	⟨9⟩	9
08	CIY	D
09	⟨8⟩	8
0A	JUMP	F
0B	⟨0⟩	0
0C	⟨2⟩	2
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨D⟩	D

FLOWCHART



Set pointer to address 50.

Move contents of pointed address to Ar.

Increase pointer by 8; add contents of Ar to pointed address (= source address + 8).

Increase pointer by 9 (& so advance memory by 1), test for pointer ≠ 8:
No — STOP (eight values moved).
Yes — return to 02.

A) Key in the program and check it.

B) Load some data into 50-57. To make sure that the program puts into 58-5F what you expected, load zero into 58-5F before running the program.

For example, you might put 0 in 50, 1 in 51,....., 7 in 57, like this:—

C) Press RESET, 2, RUN to start the program.

D) Check that your data has been moved into 58-5F, like this:—

The command at 03 (Yr + 8 → Yr) sets the pointer to an address 8 higher than the address from which a character has just been taken. The command at 06 (Yr + 9 → Yr) adds another 9 to the pointer — so each time round the loop, 17 is added, the carry is dropped, and this is what happens:

1 0001
+17 10001
= (10 0010 = 2 (decimal)

No.60 Count Frequency of Numbers less than 6 stored in memory

This program searches 50-5F. When it finds a number less than 6, it keeps track of it. Then it displays the number of times a number less than 6 was loaded into registers 50-5E.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	TIY	A
04	⟨0⟩	0
05	MA	5
06	AIA	9
07	⟨A⟩	A
08	JUMP	F
09	⟨0⟩	0
0A	⟨F⟩	F
0B	CH	2
0C	AIA	9
0D	⟨1⟩	1
0E	CH	2
0F	AIY	B
10	⟨1⟩	1
11	CIY	D
12	⟨F⟩	F
13	JUMP	F
14	⟨0⟩	0
15	⟨5⟩	5
16	CH	2
17	AO	1
18	JUMP	F
19	⟨1⟩	1
1A	⟨8⟩	8

A) Key in the program and check it.

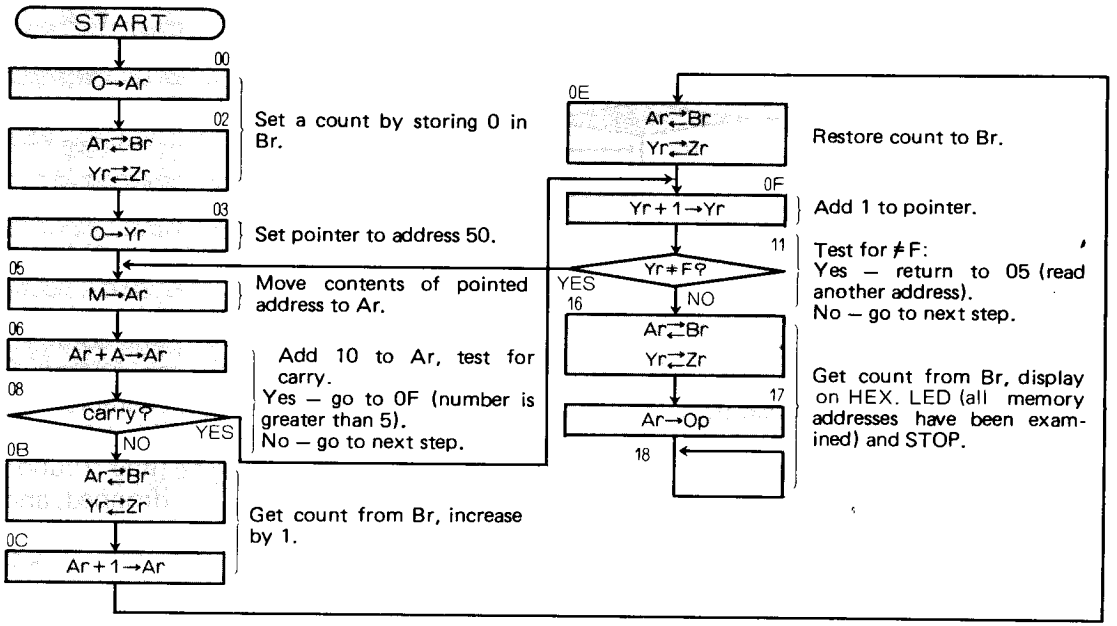
B) Load numbers into addresses 50-5E. For example, put 3 into 50-58, 7 into 59-5E, like this:—

C) Press RESET, 2, RUN to start.

Answer displayed

The count is stored in Br.

FLOWCHART



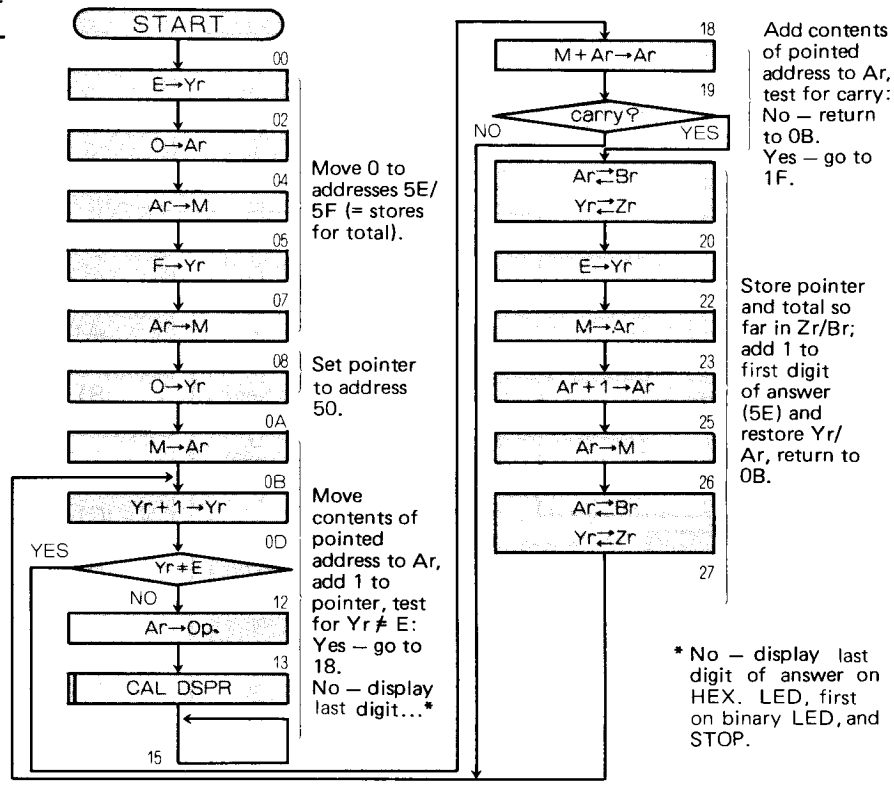
No.61 Accumulate and Display Total Contents of 50-5D

This program adds together 14 numbers (50-5D).
The program carries out a hex addition of the numbers at addresses 50-5D. The last digit of the answer is displayed on the HEX. LED, the first digit at RH side of binary LEDs.

PROGRAM

address	command	machine code	address	command	machine code
00	TIY	A	15	JUMP	F
01	⟨E⟩	E	16	⟨1⟩	1
02	TIA	8	17	⟨5⟩	5
03	⟨0⟩	0	18	M+	6
04	AM	4	19	JUMP	F
05	TIY	A	1A	⟨1⟩	1
06	⟨F⟩	F	1B	⟨F⟩	F
07	AM	4	1C	JUMP	F
08	TIY	A	1D	⟨0⟩	0
09	⟨0⟩	0	1E	⟨B⟩	B
0A	MA	5	1F	CH	2
0B	AIY	B	20	TIY	A
0C	⟨1⟩	1	21	⟨E⟩	E
0D	CIY	D	22	MA	5
0E	⟨E⟩	E	23	AIA	9
0F	JUMP	F	24	⟨1⟩	1
10	⟨1⟩	1	25	AM	4
11	⟨8⟩	8	26	CH	2
12	AO	1	27	JUMP	F
13	┌CAL	E	28	⟨0⟩	0
14	└DSPR	D	29	⟨B⟩	B

FLOWCHART



A) Key in the program and check it.

B) Load some numbers into addresses 50-5D.

Example: 50~55⟨2⟩ 56~59⟨1⟩ 5A~5D⟨3⟩
2+2+2+2+2+2+1+1+1+1+3+3+3=1C(28)

RESET MI ADDRESS SET TI INCRE MENT TI ---------- DO INCRE MENT

C) Press RESET, 1, RUN to start.

4 2 1 8 4 2 1

● ● ● ● ● ● ✲

The hex answer is 1C (= decimal 28)

No.62 Display the Average of Numbers held in memory

Work out average of 14 numbers held in 50-5D.
An average is calculated by adding together a group of numbers and then dividing the result by the number of items in the group. In this program, any remainder is discarded.

PROGRAM ①

address	command	machine code
00	T I Y	A
01	⟨E⟩	E
02	T I A	8
03	⟨0⟩	0
04	AM	4
05	T I Y	A
06	⟨F⟩	F
07	AM	4
08	T I Y	A
09	⟨0⟩	0
0A	MA	5
0B	A I Y	B
0C	⟨1⟩	1
0D	C I Y	D
0E	⟨E⟩	E
0F	JUMP	F
10	⟨3⟩	3
11	⟨B⟩	B
12	CH	2
13	T I A	8
14	⟨0⟩	0
15	CH	2
16	T I Y	A
17	⟨F⟩	F
18	AM	4
19	T I A	8
1A	⟨E⟩	E
1B	M−	7
1C	JUMP	F
1D	⟨2⟩	2
1E	⟨7⟩	7

②

address	command	machine code
1F	AM	4
20	CH	2
21	A I A	9
22	⟨1⟩	1
23	CH	2
24	JUMP	F
25	⟨1⟩	1
26	⟨9⟩	9
27	AM	4
28	T I Y	A
29	⟨E⟩	E
2A	MA	5
2B	A I A	9
2C	⟨F⟩	F
2D	JUMP	F
2E	⟨3⟩	3
2F	⟨5⟩	5
30	CH	2
31	AO	1
32	JUMP	F
33	⟨3⟩	3
34	⟨2⟩	2
35	AM	4
36	T I Y	A
37	⟨F⟩	F
38	JUMP	F
39	⟨2⟩	2
3A	⟨0⟩	0
3B	M+	6
3C	JUMP	F
3D	⟨4⟩	4

③

address	command	machine code
3E	⟨2⟩	2
3F	JUMP	F
40	⟨0⟩	0
41	⟨B⟩	B
42	CH	2
43	T I Y	A
44	⟨E⟩	E
45	MA	5

④

address	command	machine code
46	A I A	9
47	⟨1⟩	1
48	AM	4
49	CH	2
4A	JUMP	F
4B	⟨0⟩	0
4C	⟨B⟩	B

A) Key in the program and check it.

B) Load some numbers into 50-5D.
Example: 50~54⟨4⟩ 55~59⟨6⟩ 5A~5D⟨1⟩

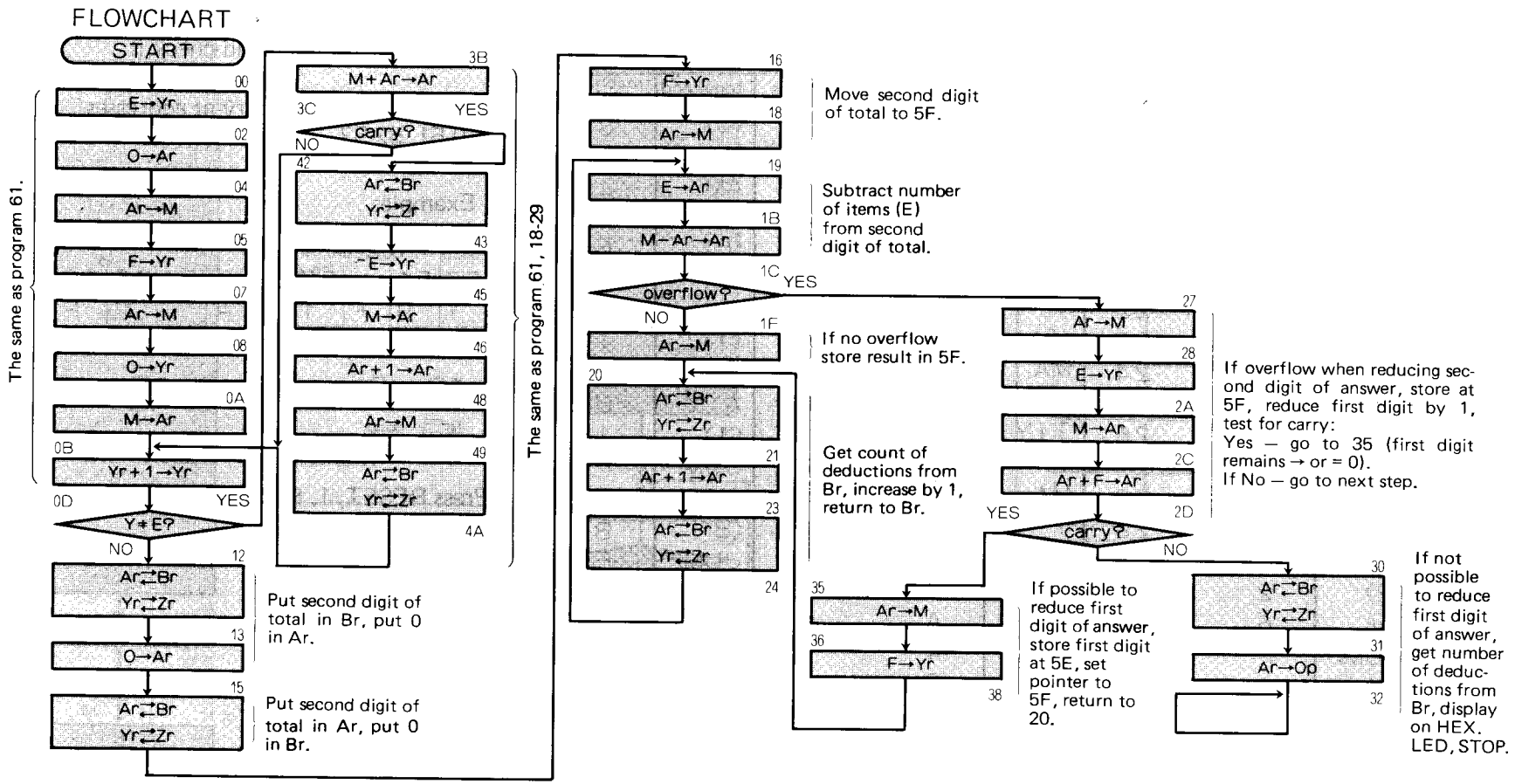
C) Press RESET, 2, RUN to start the program.

D) Answer is 3 sum total ÷ number of items = Average
(4+4+4+4+4+6+6+6+6+6+1+1+1+1)÷14≒3

E) Try changing the data in 50-5D.
Remember that remainders are ignored.

Division is carried out by repeated deductions. The number of deductions is stored in Br and displayed on the HEX. LED.
A program this long has a very complicated flowchart; treat it like a jigsaw puzzle and fit all the pieces together in your mind.

This is a long program. Don't worry if you cannot understand it at first. Come back to it later.


FLOWCHART
START
00 E→Yr
02 0→Ar
04 Ar→M
05 F→Yr
07 Ar→M
08 0→Yr
0A M→Ar
0B Yr+1→Yr
0D Y=E?
YES
NO
12 Ar⇄Br Yr⇄Zr
13 0→Ar
15 Ar⇄Br Yr⇄Zr
Put second digit of total in Br, put 0 in Ar.
Put second digit of total in Ar, put 0 in Br.
The same as program 61.
3B M+Ar→Ar
3C carry?
YES
NO
42 Ar⇄Br Yr⇄Zr
43 -E→Yr
45 M→Ar
46 Ar+1→Ar
48 Ar→M
49 Ar⇄Br Yr⇄Zr
4A
The same as program 61, 18-29
16 F→Yr
18 Ar→M
19 E→Ar
1B M-Ar→Ar
1C overflow?
YES
NO
1F Ar→M
20 Ar⇄Br Yr⇄Zr
21 Ar+1→Ar
23 Ar⇄Br Yr⇄Zr
24
Move second digit of total to 5F.
Subtract number of items (E) from second digit of total.
If no overflow store result in 5F.
Get count of deductions from Br, increase by 1, return to Br.
27 Ar→M
28 E→Yr
2A M→Ar
2C Ar+F→Ar
2D carry?
YES
NO
If overflow when reducing second digit of answer, store at 5F, reduce first digit by 1, test for carry:
Yes — go to 35 (first digit remains → or = 0).
If No — go to next step.
35 Ar→M
36 F→Yr
38
If possible to reduce first digit of answer, store first digit at 5E, set pointer to 5F, return to 20.
30 Ar⇄Br Yr⇄Zr
31 Ar→Op
32
If not possible to reduce first digit of answer, get number of deductions from Br, display on HEX. LED, STOP.

No.63 Hex Addition: 2 Digits + 2 Digits

Add two hex numbers to get a hex answer.

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨4⟩	4
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨5⟩	5
07	AM	4
08	TIY	A
09	⟨6⟩	6
0A	AM	4
0B	TIY	A
0C	⟨1⟩	1
0D	MA	5
0E	AIY	B
0F	⟨2⟩	2
10	M+	6
11	JUMP	F
12	⟨3⟩	3
13	⟨7⟩	7
14	AIY	B
15	⟨3⟩	3
16	M+	6
17	JUMP	F
18	⟨3⟩	3
19	⟨A⟩	A
1A	AM	4
1B	AIY	B

②

address	command	machine code
1C	⟨A⟩	A
1D	CIY	D
1E	⟨F⟩	F
1F	JUMP	F
20	⟨0⟩	0
21	⟨D⟩	D
22	TIY	A
23	⟨6⟩	6
24	MA	5
25	AO	1
26	TIY	A
27	⟨5⟩	5
28	MA	5
29	TIY	A
2A	⟨E⟩	E
2B	AM	4
2C	TIY	A
2D	⟨4⟩	4
2E	MA	5
2F	TIY	A
30	⟨F⟩	F
31	AM	4
32	[CAL	E
33	[DSPR	D
34	JUMP	F
35	⟨3⟩	3
36	⟨4⟩	4
37	AIY	B

③

address	command	machine code
38	⟨3⟩	3
39	M+	6
3A	AM	4
3B	AIY	B
3C	⟨F⟩	F
3D	MA	5
3E	AIA	9
3F	⟨1⟩	1
40	AM	4
41	AIY	B
42	⟨1⟩	1
43	JUMP	F
44	⟨1⟩	1
45	⟨B⟩	B
46	JUMP	F
47	⟨1⟩	1
48	⟨B⟩	B

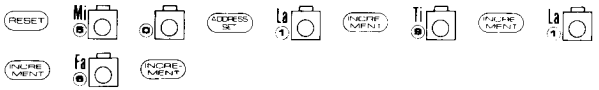
A) Key in the program and check it.

B) Load data into memory at addresses 50-51, 52-53.

Example:

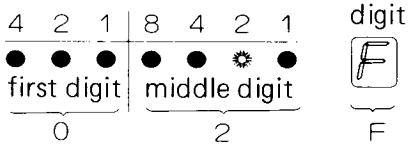
50~51⟨19⟩ 52~53⟨16⟩

50	51		52	53		54	55	56
1	9	+	1	6	=		2	F
HEX		+	HEX		=	HEX		



C) Press RESET, 1, RUN to start the program.

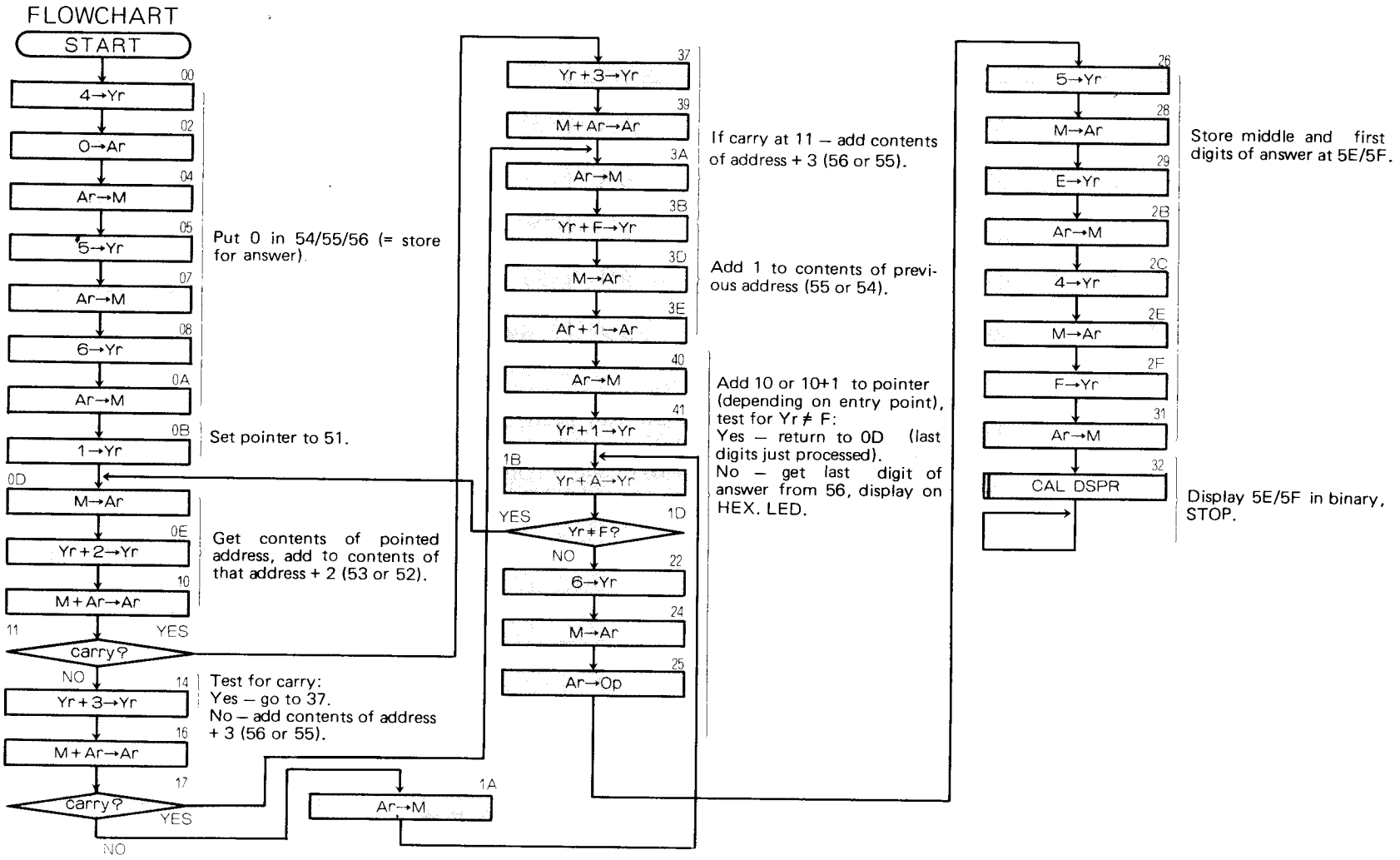
D) First digit



last digit

Note — the answer is in hex.

The displayed answer is 3 digits and is displayed using both halves of the binary LED and the HEX. LED. The first and middle digits are displayed in binary.


FLOWCHART
START
00
4→Yr
02
0→Ar
04
Ar→M
05
5→Yr
07
Ar→M
08
6→Yr
0A
Ar→M
0B
1→Yr
Put 0 in 54/55/56 (= store for answer).
Set pointer to 51.
0D
M→Ar
0E
Yr+2→Yr
10
M+Ar→Ar
11
YES
carry?
NO
14
Yr+3→Yr
16
M+Ar→Ar
17
carry?
YES
NO
Get contents of pointed address, add to contents of that address + 2 (53 or 52).
Test for carry:
Yes – go to 37.
No – add contents of address + 3 (56 or 55).
1A
Ar→M
37
Yr+3→Yr
39
M+Ar→Ar
3A
Ar→M
3B
Yr+F→Yr
3D
M→Ar
3E
Ar+1→Ar
40
Ar→M
41
Yr+1→Yr
1B
Yr+A→Yr
YES
1D
Yr≠F?
NO
22
6→Yr
24
M→Ar
25
Ar→Op
If carry at 11 – add contents of address + 3 (56 or 55).
Add 1 to contents of previous address (55 or 54).
Add 10 or 10+1 to pointer (depending on entry point), test for Yr ≠ F:
Yes – return to 0D (last digits just processed).
No – get last digit of answer from 56, display on HEX. LED.
26
5→Yr
28
M→Ar
29
E→Yr
2B
Ar→M
2C
4→Yr
2E
M→Ar
2F
F→Yr
31
Ar→M
32
CAL DSPR
Store middle and first digits of answer at 5E/5F.
Display 5E/5F in binary, STOP.

No.64 Hex Subtraction: 2 Digits − 2 Digits

Displays E on HEX. LED if answer would be minus. Remember, these are hex numbers.

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	TIY	A
04	⟨4⟩	4
05	AM	4
06	TIY	A
07	⟨1⟩	1
08	MA	5
09	TIY	A
0A	⟨5⟩	5
0B	AM	4
0C	AIY	B
0D	⟨E⟩	E
0E	MA	5
0F	AIY	B
10	⟨2⟩	2
11	M−	7
12	JUMP	F
13	⟨2⟩	2
14	⟨E⟩	E
15	AM	4
16	AIY	B
17	⟨D⟩	D
18	CIY	D
19	⟨1⟩	1
1A	JUMP	F
1B	⟨0⟩	0
1C	⟨E⟩	E

②

address	command	machine code
1D	TIY	A
1E	⟨E⟩	E
1F	AM	4
20	TIY	A
21	⟨F⟩	F
22	TIA	8
23	⟨0⟩	0
24	AM	4
25	[CAL	E
26	[DSPR	D
27	TIY	A
28	⟨5⟩	5
29	MA	5
2A	AO	1
2B	JUMP	F
2C	⟨2⟩	2
2D	⟨B⟩	B
2E	AM	4
2F	AIY	B
30	⟨F⟩	F
31	CIY	D
32	⟨4⟩	4
33	JUMP	F
34	⟨3⟩	3
35	⟨C⟩	C
36	MA	5
37	AIA	9
38	⟨F⟩	F
39	JUMP	F

③

address	command	machine code
3A	⟨0⟩	0
3B	⟨B⟩	B
3C	TIA	8
3D	⟨E⟩	E
3E	AO	1
3F	TIA	8
40	⟨2⟩	2
41	[CAL	E
42	[TIMR	C

④

address	command	machine code
43	[CAL	E
44	[RSTO	0
45	[CAL	E
46	[TIMR	C
47	JUMP	F
48	⟨3⟩	3
49	⟨C⟩	C

A) Key in the program and check it.

B) Load a number into 50/51, another into 52/53, for example load the numbers 32 and 19.

50	51		52	53		54	55
3	2	−	1	9	=	1	9

Note — these are hex numbers.

C) Press RESET, 1, RUN to start the program.

D)

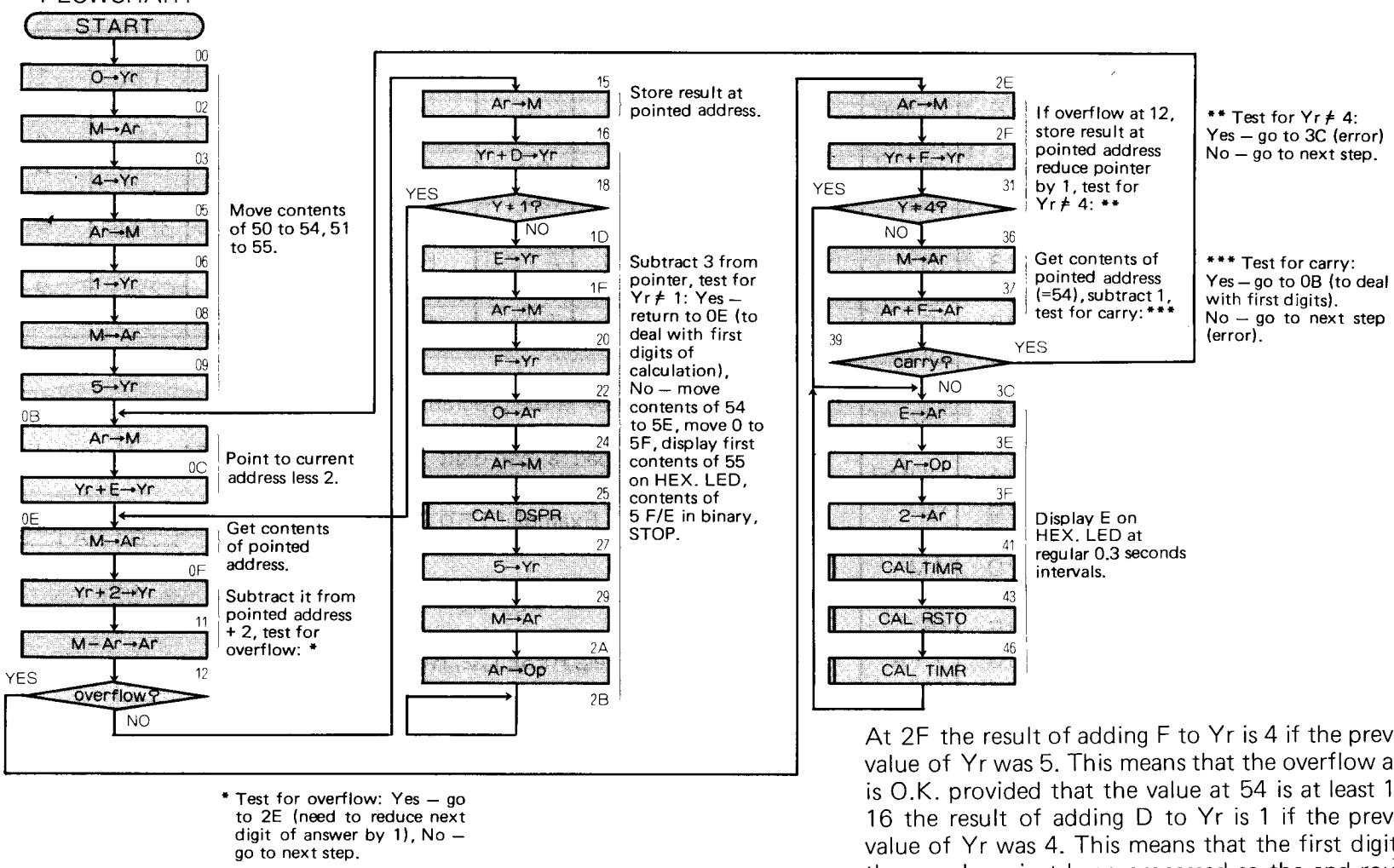
4 2 1	8 4 2 1
● ● ●	● ● ● ○

Note — the answer is in hex.

If the answer is negative, E is displayed on the HEX. LED.

All three displaying positions are used to display the result. But since in this program the answer is never greater than 2 digits, the first digit is always set to 0.

FLOWCHART



** Test for Yr ≠ 4: Yes — go to 3C (error) No — go to next step.

*** Test for carry: Yes — go to 0B (to deal with first digits). No — go to next step (error).

* Test for overflow: Yes — go to 2E (need to reduce next digit of answer by 1), No — go to next step.

At 2F the result of adding F to Yr is 4 if the previous value of Yr was 5. This means that the overflow at 12 is O.K. provided that the value at 54 is at least 1. At 16 the result of adding D to Yr is 1 if the previous value of Yr was 4. This means that the first digits of the sum have just been processed so the end routine should now be executed.

No.65 Hex Multiplication: 2 Digits × 1 Digit

Three-digit result is stored at addresses 53-55.

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨5⟩	5
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	AIY	B
06	⟨F⟩	F
07	CIY	D
08	⟨2⟩	2
09	JUMP	F
0A	⟨0⟩	0
0B	⟨4⟩	4
0C	MA	5
0D	CIA	C
0E	⟨0⟩	0
0F	JUMP	F
10	⟨1⟩	1
11	⟨5⟩	5
12	JUMP	F
13	⟨1⟩	1
14	⟨2⟩	2
15	AIA	9
16	⟨F⟩	F
17	CH	2
18	TIY	A
19	⟨1⟩	1
1A	MA	5
1B	AIY	B
1C	⟨4⟩	4

②

address	command	machine code
1D	M+	6
1E	JUMP	F
1F	⟨2⟩	2
20	⟨D⟩	D
21	AM	4
22	AIY	B
23	⟨B⟩	B
24	CIY	D
25	⟨F⟩	F
26	JUMP	F
27	⟨1⟩	1
28	⟨A⟩	A
29	CH	2
2A	JUMP	F
2B	⟨0⟩	0
2C	⟨D⟩	D
2D	AM	4
2E	AIY	B
2F	⟨F⟩	F
30	TIA	8
31	⟨1⟩	1
32	M+	6
33	JUMP	F
34	⟨3⟩	3
35	⟨F⟩	F
36	AM	4
37	AIY	B
38	⟨1⟩	1
39	JUMP	F

③

address	command	machine code
3A	⟨2⟩	2
3B	⟨2⟩	2
3C	JUMP	F
3D	⟨2⟩	2
3E	⟨2⟩	2
3F	AM	4
40	AIY	B
41	⟨F⟩	F
42	TIA	8

④

address	command	machine code
43	⟨1⟩	1
44	M+	6
45	AM	4
46	TIY	A
47	⟨5⟩	5
48	JUMP	F
49	⟨2⟩	2
4A	⟨2⟩	2

A) Key in the program and check it.

B) Load a two-digit multiplicand in 50/51 and a one-digit multiplier in 52.

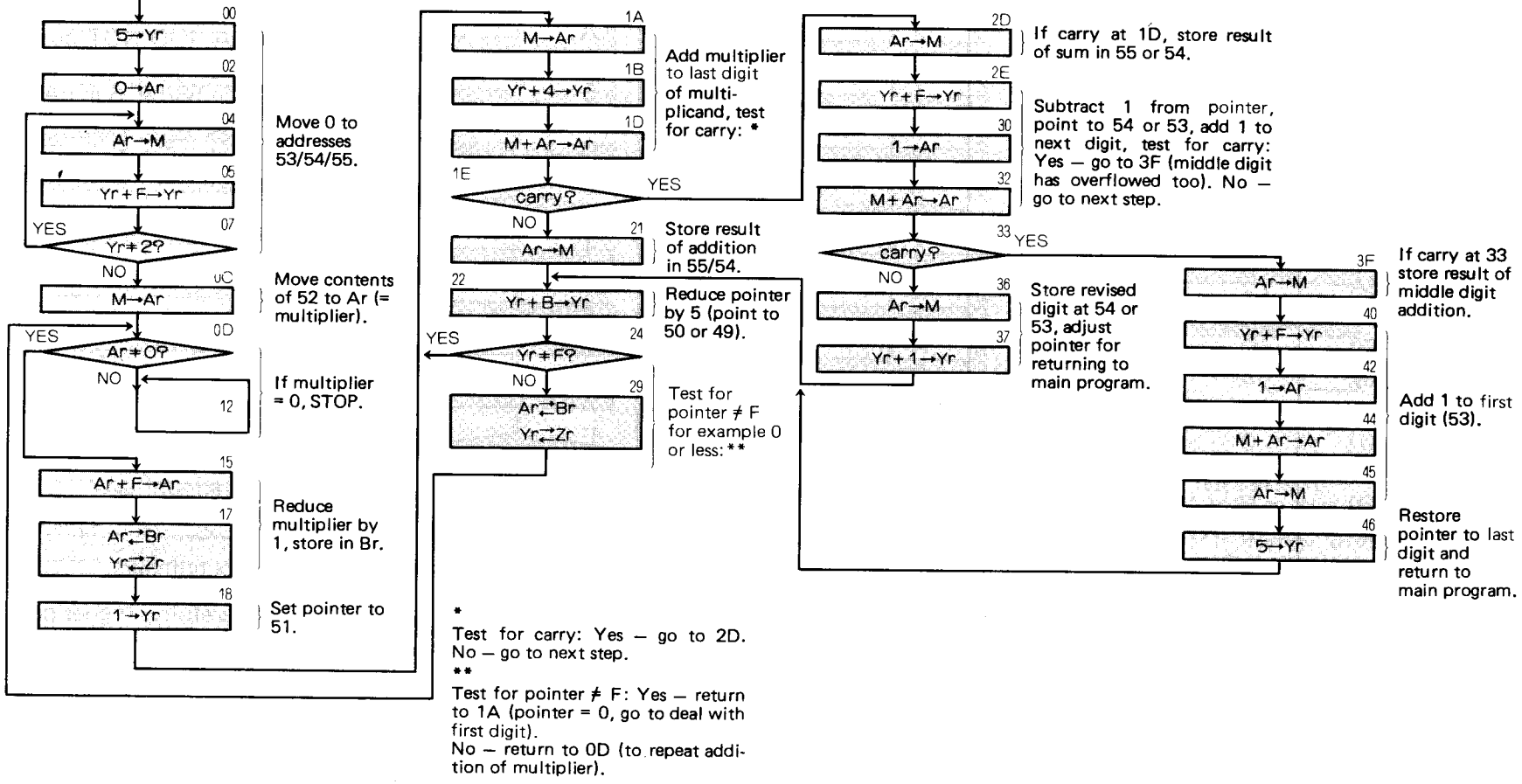
Example: | 50 | 51 | (1 | 2) × | 52 | (9) = | 53 | 54 | 55 | (0 | A | 2)

C) Press RESET, 2, RUN to start the program.

D) 

Note — the answer is in hex.

FLOWCHART



*
Test for carry: Yes – go to 2D.
No – go to next step.

**
Test for pointer ≠ F: Yes – return to 1A (pointer = 0, go to deal with first digit).
No – return to 0D (to repeat addition of multiplier).

No.66 Hex Division: 2 Digits ÷ 1 Digit

The answer is displayed in binary on binary LEDs; the remainder on the HEX. LED.

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨8⟩	8
02	MA	5
03	TIY	A
04	⟨C⟩	C
05	AM	4
06	TIY	A
07	⟨9⟩	9
08	MA	5
09	TIY	A
0A	⟨D⟩	D
0B	AM	4
0C	TIA	8
0D	⟨0⟩	0
0E	TIY	A
0F	⟨B⟩	B
10	AM	4
11	CH	2
12	TIY	A
13	⟨A⟩	A
14	MA	5
15	TIY	A
16	⟨D⟩	D
17	M−	7
18	JUMP	F
19	⟨2⟩	2
1A	⟨F⟩	F
1B	AM	4

②

address	command	machine code
1C	CH	2
1D	AIA	9
1E	⟨1⟩	1
1F	JUMP	F
20	⟨2⟩	2
21	⟨5⟩	5
22	JUMP	F
23	⟨1⟩	1
24	⟨1⟩	1
25	CH	2
26	TIY	A
27	⟨B⟩	B
28	MA	5
29	AIA	9
2A	⟨1⟩	1
2B	AM	4
2C	JUMP	F
2D	⟨1⟩	1
2E	⟨2⟩	2
2F	AM	4
30	TIY	A
31	⟨C⟩	C
32	MA	5
33	AIA	9
34	⟨F⟩	F
35	JUMP	F
36	⟨1⟩	1
37	⟨B⟩	B

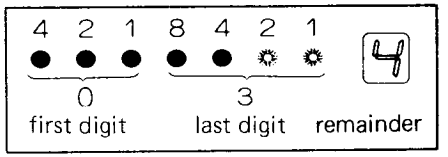
③

address	command	machine code
38	TIY	A
39	⟨A⟩	A
3A	MA	5
3B	TIY	A
3C	⟨D⟩	D
3D	M+	6
3E	AM	4
3F	AO	1
40	CH	2
41	TIY	A
42	⟨C⟩	C
43	AM	4
44	TIY	A
45	⟨E⟩	E
46	AM	4
47	TIY	A
48	⟨B⟩	B
49	MA	5
4A	TIY	A
4B	⟨F⟩	F
4C	AM	4
4D	⌜CAL	E
4E	⌞DSPR	D
4F	JUMP	F
50	⟨4⟩	4
51	⟨F⟩	F

A) Key in the program and check it.

B) Load the number to be divided at 58/59, the divisor at 5A. (DO NOT put 0 in 5A)
Example: Put 16 in 58/59, 6 in 5A.

C) Press RESET, 1, RUN to start.



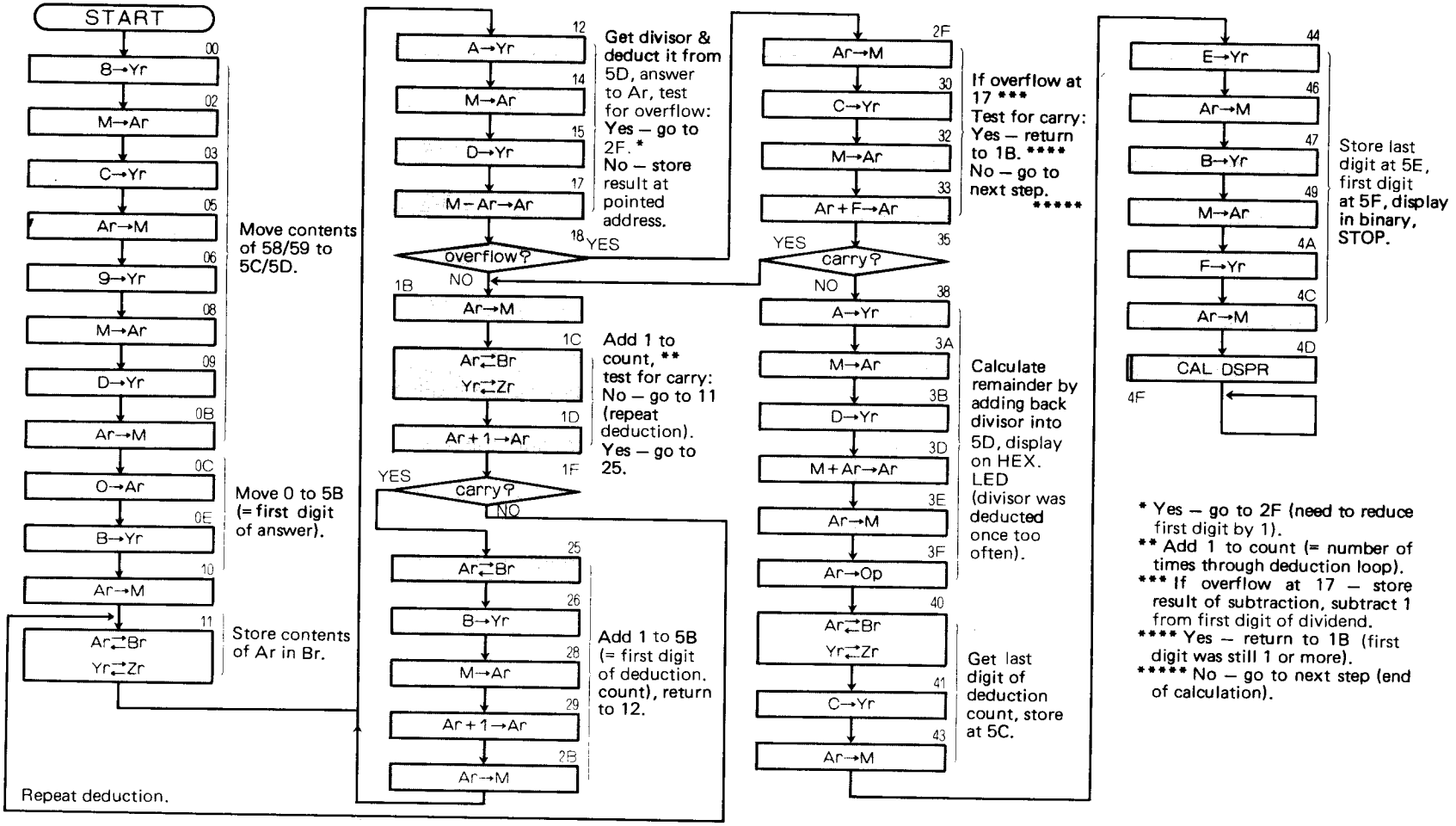
58	59		5A		5B	5C	
1	6	÷	6	=	0	3	hex!

4 remainder

Remember, these are hex numbers.

Look at this program carefully. It goes past the upper limit of 4F! But this is all right — the program does not use addresses 50/51 for storage. Remember this: if there is a large program you can use 50-5F for it provided that the program does not need any memory for data storage apart from the registers.

FLOWCHART



* Yes – go to 2F (need to reduce first digit by 1).
** Add 1 to count (= number of times through deduction loop).
*** If overflow at 17 – store result of subtraction, subtract 1 from first digit of dividend.
**** Yes – return to 1B (first digit was still 1 or more).
***** No – go to next step (end of calculation).

(4) Group 4 Commands

There are nine commands in group 4. They are CAL INPT, CAL CHNG, CAL SIFT, CAL ENDS, CAL ERRS, CAL LONS, CAL SUND, CAL DEM –, CAL DEM +. As usual they will be introduced and explained one by one.

This is the last group of commands. All of the commands that you are learning are useful and important and make it possible to do interesting things with your Microcomputer Trainer.

CAL SUND COMMAND (CALI SoUND)

This command generates a musical sound that varies according to the contents of Ar. The range of notes is the same as the range used in earlier experiments 1 and 2.

Table of musical values:

Contents of Ar sol-fah	Contents of Ar sol-fah
0 ····· no sound	8 ····· la
1 ····· la	9 ····· ti
2 ····· ti	A ····· do
3 ····· do	B ····· re
4 ····· re	C ····· mi
5 ····· mi	D ····· fa
6 ····· fa	E ····· sol
7 ····· sol	F ····· no sound

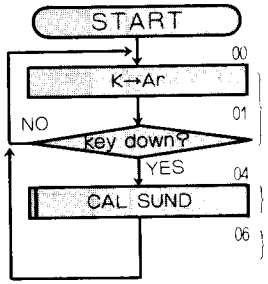
No.67 Use of CAL SUND

Basic program for generating organ notes

PROGRAM

address	command	machine code
00	KA	0
01	JUMP	F
02	⟨0⟩	0
03	⟨0⟩	0
04	CAL	E
05	SUND	B
06	JUMP	F
07	⟨0⟩	0
08	⟨0⟩	0

FLOWCHART



Test for key pressed: No – wait, Yes – go to next step.

Play note corresponding to key pressed.

Return to 00 and repeat.

This is the program for a very simple organ.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

C) Press 1-E keys to play notes of different pitches.

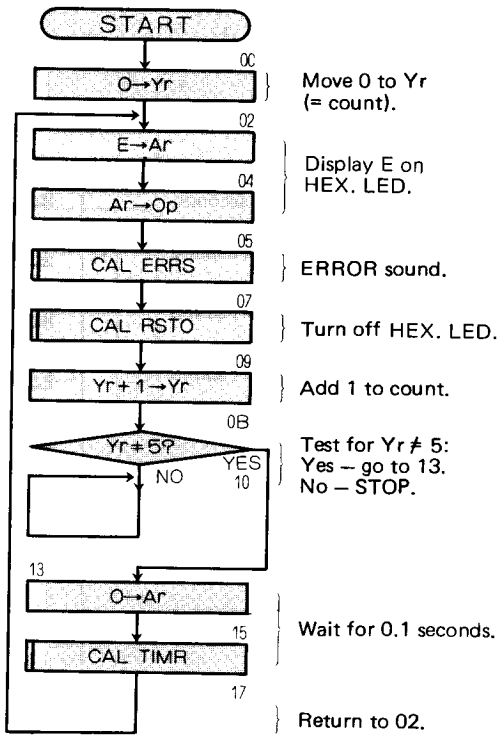
No.68 Use of CAL ERRS

CAL ERRS COMMAND (CALI ERRor Sound)
This command makes a nasty noise. It is used when an error occurs in a program.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨E⟩	E
04	AO	1
05	⌜CAL	E
06	⌞ERRS	8
07	⌜CAL	E
08	⌞RSTO	0
09	AIY	B
0A	⟨1⟩	1
0B	CIY	D
0C	⟨5⟩	5
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨3⟩	3
10	JUMP	F
11	⟨1⟩	1
12	⟨0⟩	0
13	TIA	8
14	⟨0⟩	0
15	⌜CAL	E
16	⌞TIMR	C
17	JUMP	F
18	⟨0⟩	0
19	⟨2⟩	2

FLOWCHART



Move 0 to Yr (= count).

Display E on HEX. LED.

ERROR sound.

Turn off HEX. LED.

Add 1 to count.

Test for Yr ≠ 5:
Yes – go to 13.
No – STOP.

Wait for 0.1 seconds.

Return to 02.

This program makes the ERROR sound and displays E on the HEX. LED five times.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

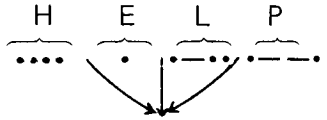
No.69 Use of CAL LONS

This program is a simple way of generating code signals.

CAL LONS COMMAND (CALl LONg Sound)
This command generates a long sound (compared with CAL SHTS, program 45).

Some of the programs in this manual use Morse Code. Here we are writing codes into a short program which then produces the signals.

This program repeats the Morse code for HELP.



Space between characters

PROGRAM

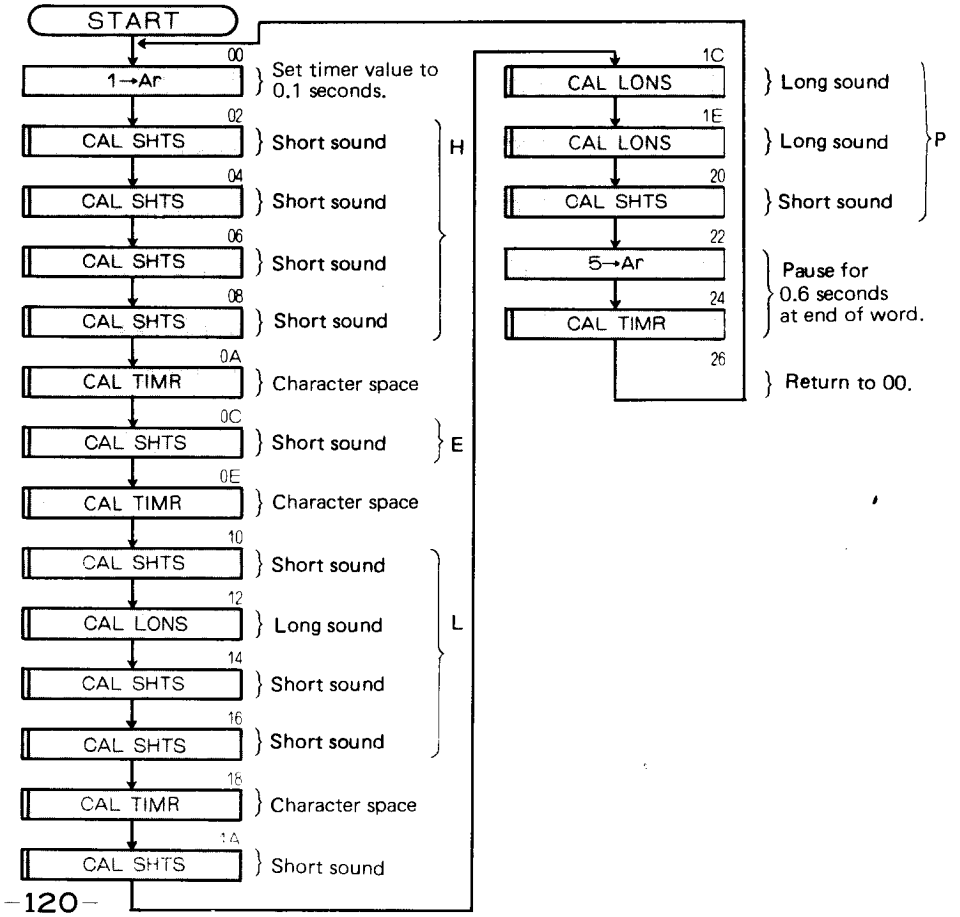
address	command	machine code
00	T I A	8
01	⟨1⟩	1
02	⌈CAL	E
03	⌊SHTS	9
04	⌈CAL	E
05	⌊SHTS	9
06	⌈CAL	E
07	⌊SHTS	9
08	⌈CAL	E
09	⌊SHTS	9
0A	⌈CAL	E
0B	⌊TIMR	C
0C	⌈CAL	E
0D	⌊SHTS	9
0E	⌈CAL	E
0F	⌊TIMR	C
10	⌈CAL	E
11	⌊SHTS	9
12	⌈CAL	E
13	⌊LONS	A
14	⌈CAL	E
15	⌊SHTS	9

address	command	machine code
16	⌈CAL	E
17	⌊SHTS	9
18	⌈CAL	E
19	⌊TIMR	C
1A	⌈CAL	E
1B	⌊SHTS	9
1C	⌈CAL	E
1D	⌊LONS	A
1E	⌈CAL	E
1F	⌊LONS	A
20	⌈CAL	E
21	⌊SHTS	9
22	T I A	8
23	⟨5⟩	5
24	⌈CAL	E
25	⌊TIMR	C
26	JUMP	F
27	⟨0⟩	0
28	⟨0⟩	0

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

FLOWCHART



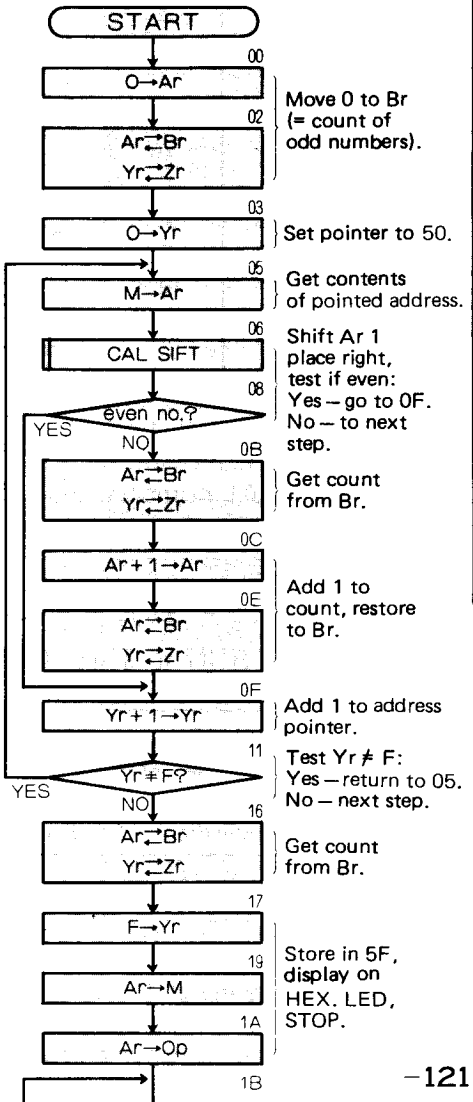
No.70 Use of CAL SIFT

This program uses CAL SIFT to count the odd numbers in 50-5E.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	TIY	A
04	⟨0⟩	0
05	MA	5
06	CAL	E
07	SIFT	6
08	JUMP	F
09	⟨0⟩	0
0A	⟨F⟩	F
0B	CH	2
0C	AIA	9
0D	⟨1⟩	1
0E	CH	2
0F	AIY	B
10	⟨1⟩	1
11	CIY	D
12	⟨F⟩	F
13	JUMP	F
14	⟨0⟩	0
15	⟨5⟩	5
16	CH	2
17	TIY	A
18	⟨F⟩	F
19	AM	4
1A	AO	1
1B	JUMP	F
1C	⟨1⟩	1
1D	⟨B⟩	B

FLOWCHART



CAL SIFT COMMAND (CALI ShIFT)

This command shifts the binary contents of Ar one place to the right. The command sets the FLAG to 0 or 1, values shown below:

BEFORE CAL SIFT		AFTER CAL SIFT	executing
hex No.	A-register	A-register	FLAG setting
0	0000	0000	1
1	0001	0000	0
2	0010	0001	1
3	0011	0001	0
4	0100	0010	1
5	0101	0010	0
6	0110	0011	1
7	0111	0011	0
8	1000	0100	1
9	1001	0100	0
A	1010	0101	1
B	1011	0101	0
C	1100	0110	1
D	1101	0110	0
E	1110	0111	1
F	1111	0111	0

1-digit shift right

When the CAL SIFT command is executed the FLAG is set to 1 if the number shifted is even or 0 if the number shifted is odd.

A) Key in the program and check it.

B) Load data into 50-5E.

Example:

50	51	52	53	54	55	·····	59	5A	5B	5C	5D	5E
0	1	2	3	4	5	·····	9	A	B	C	D	E

C) Press RESET, 2, RUN to start.
There are 7 odd numbers listed in this program.

display 7 on HEX. LED

No.71 Use of CAL DEM+ and CAL ENDS

Decimal addition of Two digits + Two digits

CAL DEM+ COMMAND (CALI DEcimal conversion of M+ result)

This command adds together the decimal contents of Ar and the pointed address to give a decimal answer and stores that answer at the pointed address. If there is a carry, 1 is added to the pointed address less 1. After the command has been executed the pointer is left pointing one address below the pointed address. (If the number to be added is in 54, the answer is put in 54 and the pointer is reduced by 3. If there is a carry, one is added to 53.)

CAL ENDS COMMAND (CALI END Sound)

This command is used to generate a sound to indicate that a game or calculation has ended. It sounds nicer than ERRS sound!

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨1⟩	1
02	MA	5
03	AIY	B
04	⟨5⟩	5
05	AM	4
06	AIY	B
07	⟨A⟩	A
08	CIY	D
09	⟨F⟩	F
0A	JUMP	F
0B	⟨0⟩	0
0C	⟨2⟩	2
0D	TIA	8
0E	⟨0⟩	0

②

address	command	machine code
0F	TIY	A
10	⟨4⟩	4
11	AM	4
12	TIY	A
13	⟨3⟩	3
14	MA	5
15	AIY	B
16	⟨3⟩	3
17	CH	2
18	CH	2
19	⌜CAL	E
1A	⌞DEM+	F
1B	AIY	B
1C	⟨D⟩	D
1D	CIY	D

③

address	command	machine code
1E	⟨1⟩	1
1F	JUMP	F
20	⟨1⟩	1
21	⟨4⟩	4
22	⌜CAL	E
23	⌞ENDS	7
24	TIY	A
25	⟨4⟩	4
26	MA	5
27	TIY	A
28	⟨F⟩	F
29	AM	4
2A	TIY	A
2B	⟨5⟩	5

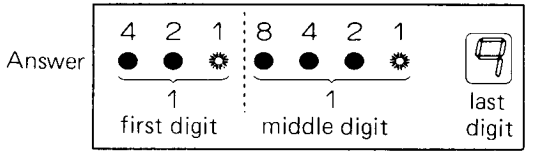
④

address	command	machine code
2C	MA	5
2D	TIY	A
2E	⟨E⟩	E
2F	AM	4
30	TIY	A
31	⟨6⟩	6
32	MA	5
33	AO	1
34	⌜CAL	E
35	⌞DSPR	D
36	JUMP	F
37	⟨3⟩	3
38	⟨6⟩	6

A) Key in the program and check it.

B) Load the DECIMAL numbers to be added into 50-53.
Example: 50⟨6⟩ 51⟨3⟩ 52⟨5⟩ 53⟨6⟩
63+56=?

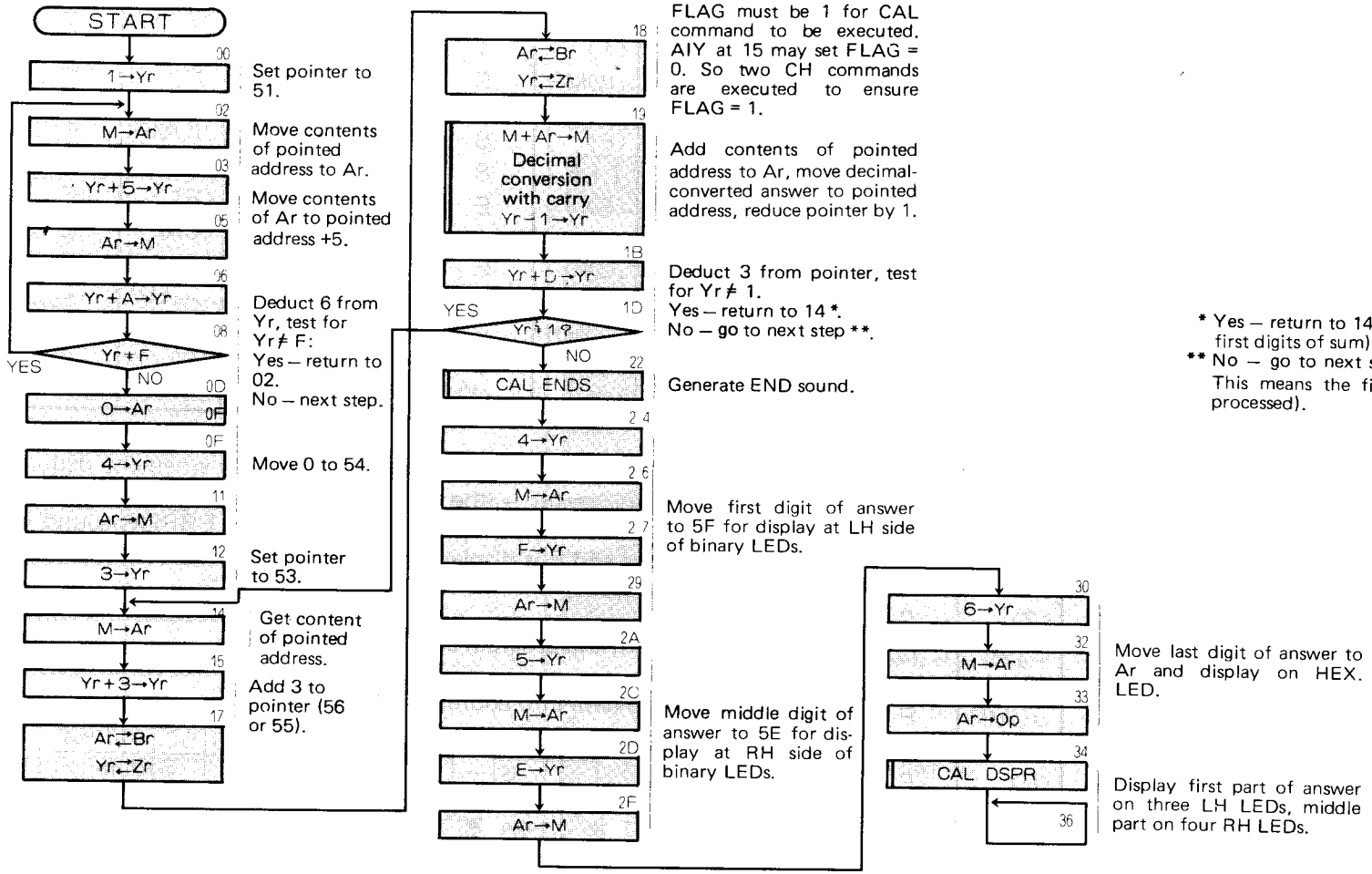
C) Press RESET, 1, RUN to start. The program then displays the answer and makes the END sound.



address	50	51		52	53		54	55	56
	6	3	+	5	6	=	1	1	9

Decimal sums may be easier to understand than hex sums, but converting hex to decimal uses up a lot of memory (see programs 35/36). The CAL DEM+ routine is provided to save that space in the micro.

FLOWCHART



Set pointer to 51.

Move contents of pointed address to Ar.

Move contents of Ar to pointed address +5.

Deduct 6 from Yr, test for Yr≠ F:
Yes – return to 02.
No – next step.

Move 0 to 54.

Set pointer to 53.

Get content of pointed address.

Add 3 to pointer (56 or 55).

FLAG must be 1 for CAL command to be executed. AIY at 15 may set FLAG = 0. So two CH commands are executed to ensure FLAG = 1.

Add contents of pointed address to Ar, move decimal-converted answer to pointed address, reduce pointer by 1.

Deduct 3 from pointer, test for Yr≠ 1.
Yes – return to 14 *.
No – go to next step **.

Generate END sound.

Move first digit of answer to 5F for display at LH side of binary LEDs.

Move middle digit of answer to 5E for display at RH side of binary LEDs.

Move last digit of answer to Ar and display on HEX. LED.

Display first part of answer on three LH LEDs, middle part on four RH LEDs.

* Yes – return to 14 (to deal with first digits of sum).
** No – go to next step (4-3 = 1. This means the first digits are processed).

No.72 Use of CAL DEM−

Decimal deduction − Two digits minus Two digits

CAL DEM-COMMAND (CALl DEcimal conversion of M− result)

This command deducts the contents of Ar from the pointed address & stores the answer at the pointed address. The answer is decimal-converted. The value of the pointer is reduced by 1.***

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨A⟩	A
04	AM	4
05	AIY	B
06	⟨1⟩	1
07	CIY	D
08	⟨0⟩	0
09	JUMP	F
0A	⟨0⟩	0
0B	⟨4⟩	4
0C	TIY	A
0D	⟨6⟩	6
0E	MA	5
0F	TIY	A
10	⟨B⟩	B
11	CH	2
12	TIY	A
13	⟨7⟩	7
14	MA	5
15	TIY	A
16	⟨C⟩	C
17	AM	4
18	AIY	B

②

address	command	machine code
19	⟨D⟩	D
1A	MA	5
1B	AIY	B
1C	⟨3⟩	3
1D	CH	2
1E	CH	2
1F	[CAL	E
20	[DEM−	E
21	MA	5
22	CIY	D
23	⟨B⟩	B
24	JUMP	F
25	⟨3⟩	3
26	⟨2⟩	2
27	CIA	C
28	⟨0⟩	0
29	JUMP	F
2A	⟨3⟩	3
2B	⟨F⟩	F
2C	CH	2
2D	AM	4
2E	CH	2
2F	JUMP	F
30	⟨1⟩	1
31	⟨8⟩	8

③

address	command	machine code
32	CIA	C
33	⟨1⟩	1
34	JUMP	F
35	⟨4⟩	4
36	⟨6⟩	6
37	TIA	8
38	⟨E⟩	E
39	AO	1
3A	[CAL	E
3B	[ERRS	8
3C	JUMP	F
3D	⟨3⟩	3
3E	⟨C⟩	C

④

address	command	machine code
3F	CH	2
40	AIA	9
41	⟨F⟩	F
42	AM	4
43	JUMP	F
44	⟨2⟩	2
45	⟨E⟩	E
46	[CAL	E
47	[ENDS	7
48	JUMP	F
49	⟨4⟩	4
4A	⟨8⟩	8

In this program an END sound is generated when the deduction is successful − the answer is stored at addresses 5A-5C. If it is not successful (answer is minus) the ERROR sound is generated and E is displayed on the HEX. LED.

A) Key in the program and check it.

B) Load data into addresses 56/57, 58/59.

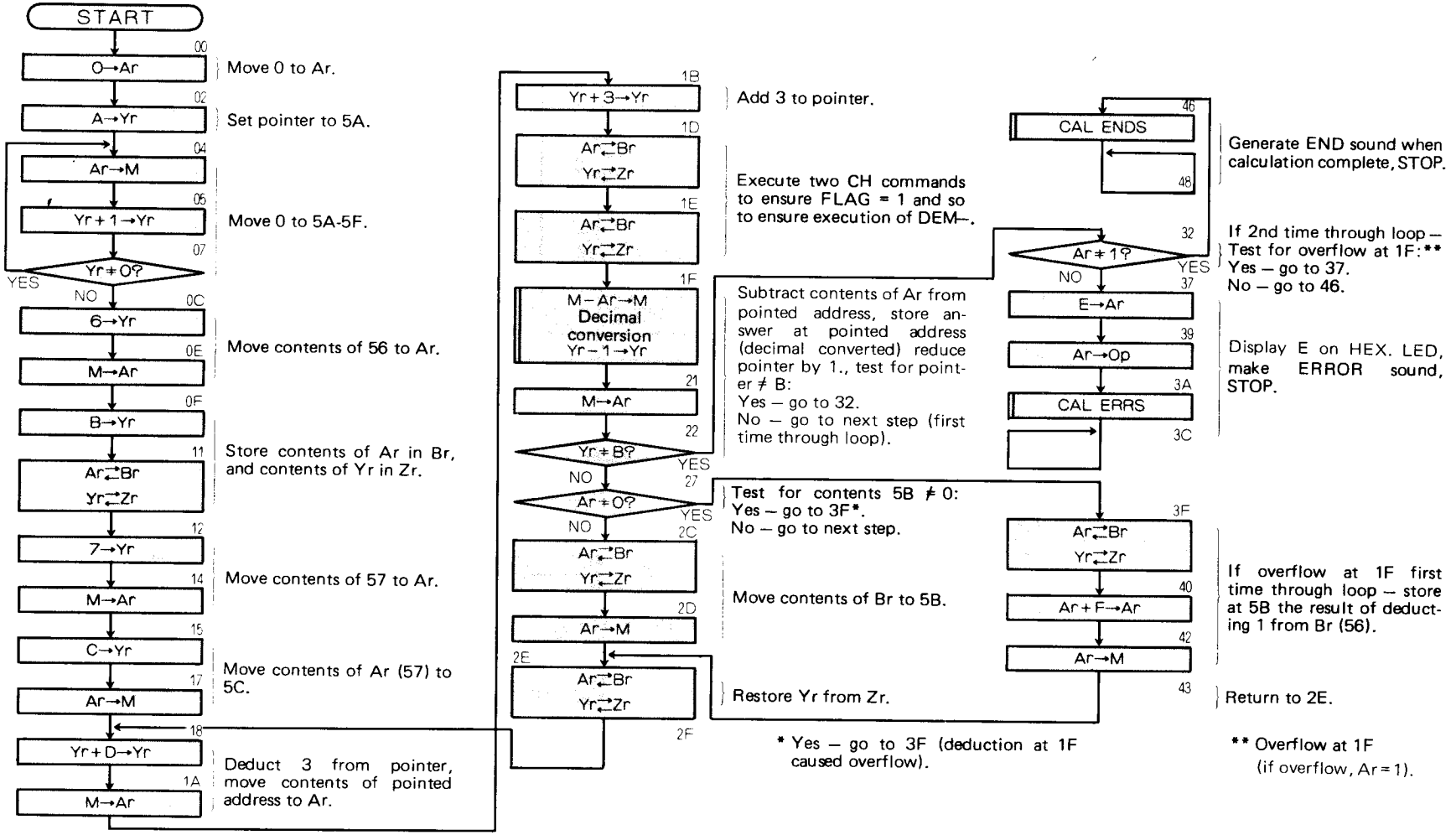
Example:

56	57		58	59		5A	5B	5C	address
7	6	−	4	8	=	0	2	8	data

C) Press RESET, 1, RUN to start. When you hear the END or ERROR sound the program has finished.
Now key in:

D) 

FLOWCHART



*** The value of the pointer is reduced by 1. If there is an overflow, one is moved to the lower address.

No.73 Use of CAL CHNG

This program sorts 15 numbers (in 50-5E) into descending order.

CAL CHNG COMMAND (CALl CHaNGe registers)

This command swaps the contents of Ar, Br, Yr & Zr with the contents of complement registers Ar', Br', Yr', Zr'. These registers can only be used by means of CAL CHNG. They are useful when extra addresses are required.

PROGRAM ①

address	command	machine code
00	T I Y	A
01	⟨F⟩	F
02	T I A	8
03	⟨2⟩	2
04	AM	4
05	┌CAL	E
06	└CHNG	5
07	T I Y	A
08	⟨0⟩	0
09	MA	5
0A	CH	2
0B	T I Y	A
0C	⟨0⟩	0
0D	MA	5
0E	┌CAL	E
0F	└CMPL	4
10	A I Y	B
11	⟨1⟩	1
12	M+	6
13	JUMP	F
14	⟨2⟩	2
15	⟨8⟩	8
16	MA	5
17	CH	2
18	A I Y	B

②

address	command	machine code
19	⟨1⟩	1
1A	CH	2
1B	CH	2
1C	┌CAL	E
1D	└CHNG	5
1E	A I A	9
1F	⟨1⟩	1
20	JUMP	F
21	⟨3⟩	3
22	⟨6⟩	6
23	┌CAL	E
24	└CHNG	5
25	JUMP	F
26	⟨0⟩	0
27	⟨D⟩	D
28	MA	5
29	A I Y	B
2A	⟨F⟩	F
2B	AM	4
2C	A I Y	B
2D	⟨1⟩	1
2E	CH	2
2F	A I Y	B
30	⟨1⟩	1
31	AM	4

③

address	command	machine code
32	CH	2
33	JUMP	F
34	⟨1⟩	1
35	⟨C⟩	C
36	T I Y	A
37	⟨F⟩	F
38	MA	5
39	A I A	9
3A	⟨1⟩	1
3B	JUMP	F

④

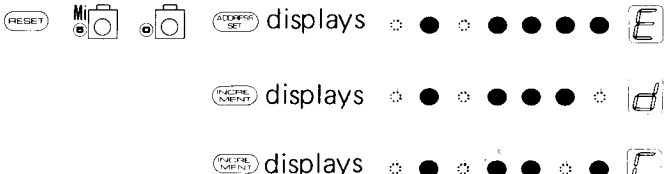
address	command	machine code
3C	⟨4⟩	4
3D	⟨2⟩	2
3E	AM	4
3F	JUMP	F
40	⟨0⟩	0
41	⟨5⟩	5
42	┌CAL	E
43	└ENDS	7
44	JUMP	F
45	⟨4⟩	4
46	⟨4⟩	4

A) Key in the program and check it.

B) Load some numbers into addresses 50-5E.
Example:

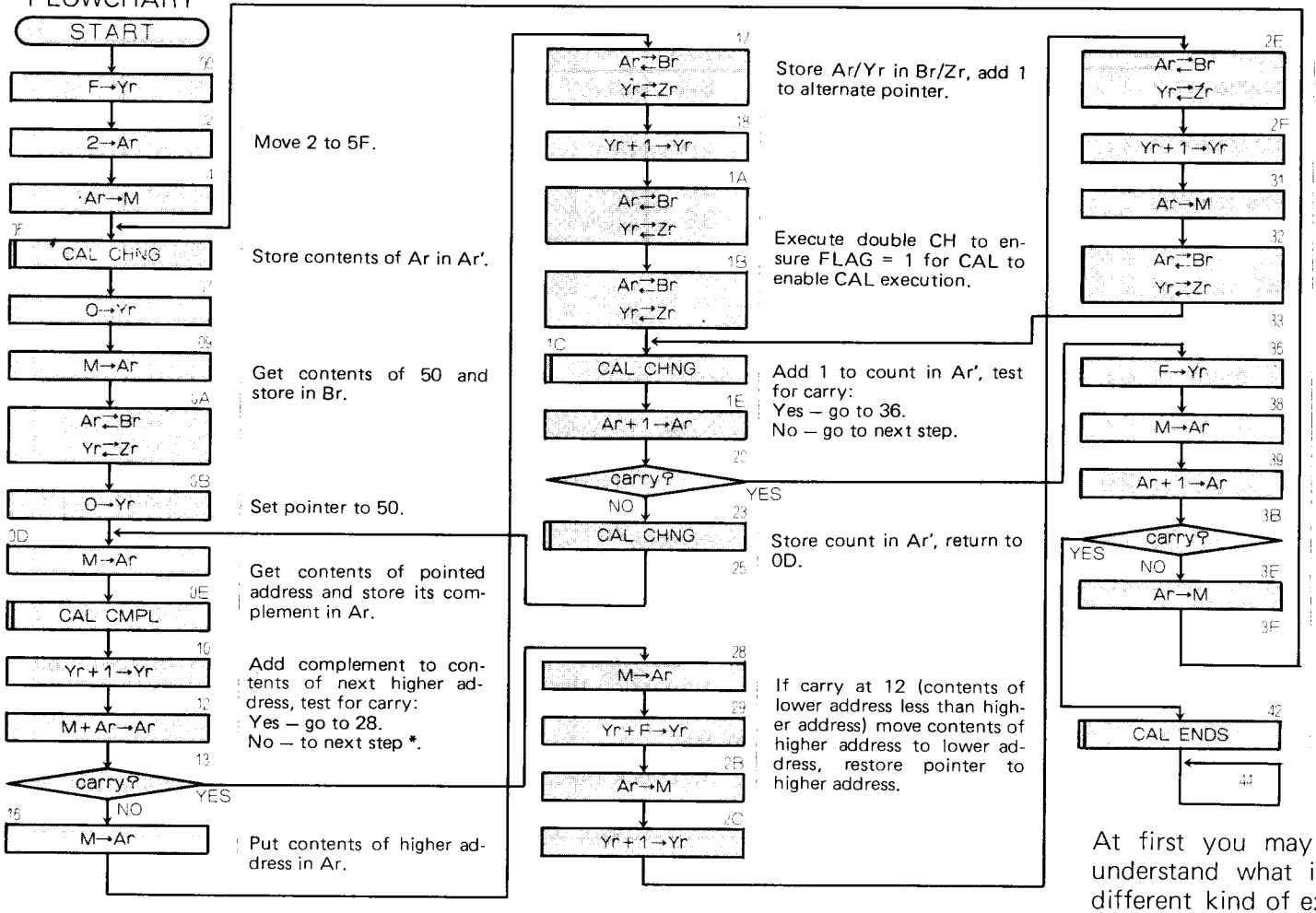
50	51	52	53	54	55	·····	59	5A	5B	5C	5D	5E
0	1	2	3	4	5	·····	9	A	B	C	D	E

C) Press RESET, 2, RUN to start. You will hear the END sound when the sort is complete.

D) 

. . . and so on.

FLOWCHART



Move 2 to 5F.

Store contents of Ar in Ar'.

Get contents of 50 and store in Br.

Set pointer to 50.

Get contents of pointed address and store its complement in Ar.

Add complement to contents of next higher address, test for carry:
Yes – go to 28.
No – to next step *.

Put contents of higher address in Ar.

Store Ar/Yr in Br/Zr, add 1 to alternate pointer.

Execute double CH to ensure FLAG = 1 for CAL to enable CAL execution.

Add 1 to count in Ar', test for carry:
Yes – go to 36.
No – go to next step.

Store count in Ar', return to 0D.

If carry at 12 (contents of lower address less than higher address) move contents of higher address to lower address, restore pointer to higher address.

Get pointer to lower address, and its previous value, from Zr/Br.

Bring pointer up-to-date, store previous contents of lower address in higher address, store once more in Zr/Br.

If carry at 20 (a number has been sorted into the right place) get count from 5F, add 1, test for carry:
Yes – go to 42 (end of sort).
No – put count back in 5F, return to 05.

Generate END sound. STOP.

* No – go to next step (Contents of lower address are greater).

At first you may find it difficult to understand what is happening here. A different kind of explanation is given in the appendix at the end of the manual.

No.74 Sort Contents of 50-5E in ascending order

The contents of 50-5E are sorted and then displayed in ascending order.

PROGRAM ①

address	command	machine code
00	T I Y	A
01	⟨F⟩	F
02	T I A	8
03	⟨2⟩	2
04	AM	4
05	┌CAL	E
06	└CHNG	5
07	T I Y	A
08	⟨0⟩	0
09	MA	5
0A	CH	2
0B	T I Y	A
0C	⟨0⟩	0
0D	MA	5
0E	┌CAL	E
0F	└CMPL	4
10	A I Y	B
11	⟨1⟩	1
12	M+	6
13	JUMP	F
14	⟨2⟩	2
15	⟨D⟩	D
16	MA	5
17	A I Y	B
18	⟨F⟩	F
19	AM	4
1A	A I Y	B
1B	⟨1⟩	1
1C	CH	2

②

address	command	machine code
1D	A I Y	B
1E	⟨1⟩	1
1F	AM	4
20	CH	2
21	┌CAL	E
22	└CHNG	5
23	A I A	9
24	⟨1⟩	1
25	JUMP	F
26	⟨3⟩	3
27	⟨5⟩	5
28	┌CAL	E
29	└CHNG	5
2A	JUMP	F
2B	⟨0⟩	0
2C	⟨D⟩	D
2D	MA	5
2E	CH	2
2F	A I Y	B
30	⟨1⟩	1
31	CH	2
32	JUMP	F
33	⟨2⟩	2
34	⟨0⟩	0
35	T I Y	A
36	⟨F⟩	F
37	MA	5
38	A I A	9
39	⟨1⟩	1

③

address	command	machine code
3A	JUMP	F
3B	⟨4⟩	4
3C	⟨1⟩	1
3D	AM	4
3E	JUMP	F
3F	⟨0⟩	0

④

address	command	machine code
40	⟨5⟩	5
41	┌CAL	E
42	└ENDS	7
43	JUMP	F
44	⟨4⟩	4
45	⟨3⟩	3

A) Key in the program and check it.

B) Load some numbers at addresses 50-5E.

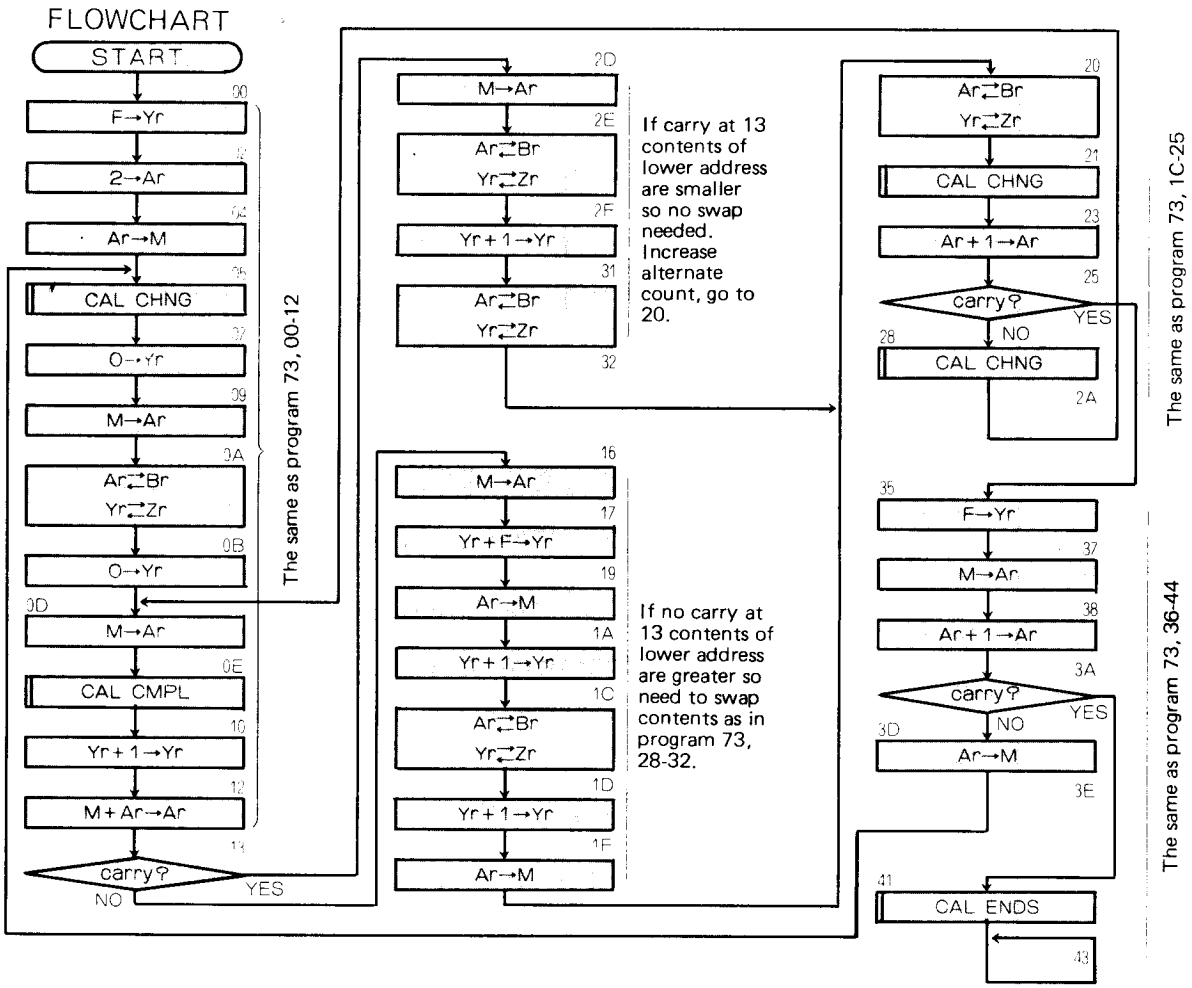
Example:

50	51	52	53	54	- - - -	5B	5C	5D	5E
8	1	C	0	A		5	1	B	6

C) Press RESET, 2, RUN to start. The END sound will tell you when the sort is complete.

D) When you hear the END sound, read out addresses 50-5E to check that the sort has been completed correctly.

Each time you press (INCR) you should see a number on the HEX. LED which is equal to or larger than the previous one. In this program both CH and CHNG are used. This means that both Br and Ar' are used as working memories.



Compare this program with no. 73.

In 73 the data has to be sorted into descending order; if there is a carry at 12, the neighboring memory contents are swapped. If there is no carry at 12, the lower address already contains a larger number so no swap is necessary.

But in 74 the tests work the other way around; if there is a carry at 12, no swap takes place. If there is no carry, a swap is necessary.

The same differences are reflected on the flowchart. Addresses 16-1A in no. 73 do the same job as 2D-31 in 74, and addresses 28-32 of 73 have the same effect as 16-20 in 74. Only the positions are changed.

In the appendix is a further reminder concerning complements.

No.75 Decimal Multiplication: 2 Digits × 1 Digit

Multiply decimal numbers to get a decimal result up to 799.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨3⟩	3
04	AM	4
05	TIY	A
06	⟨4⟩	4
07	AM	4
08	TIY	A
09	⟨5⟩	5
0A	AM	4
0B	TIY	A
0C	⟨2⟩	2
0D	MA	5
0E	CH	2
0F	TIY	A
10	⟨1⟩	1
11	MA	5
12	AIY	B
13	⟨4⟩	4
14	CH	2
15	CH	2
16	⌜CAL	E
17	⌞DEM+	F
18	AIY	B
19	⟨C⟩	C
1A	CIY	D
1B	⟨F⟩	F
1C	JUMP	F

②

address	command	machine code
1D	⟨1⟩	1
1E	⟨1⟩	1
1F	CH	2
20	AIA	9
21	⟨F⟩	F
22	CIA	C
23	⟨0⟩	0
24	JUMP	F
25	⟨0⟩	0
26	⟨E⟩	E
27	TIY	A
28	⟨3⟩	3
29	MA	5
2A	AIA	9
2B	⟨8⟩	8
2C	JUMP	F
2D	⟨4⟩	4
2E	⟨2⟩	2
2F	MA	5
30	TIY	A
31	⟨F⟩	F
32	AM	4
33	TIY	A
34	⟨4⟩	4
35	MA	5
36	TIY	A
37	⟨E⟩	E
38	AM	4
39	⌜CAL	E

③

address	command	machine code
3A	⌞DSPR	D
3B	TIY	A
3C	⟨5⟩	5
3D	MA	5
3E	AO	1
3F	JUMP	F
40	⟨3⟩	3
41	⟨F⟩	F

④

address	command	machine code
42	TIA	8
43	⟨C⟩	C
44	AO	1
45	⌜CAL	E
46	⌞ERRS	8
47	JUMP	F
48	⟨4⟩	4
49	⟨7⟩	7

The answer is stored at 53-55. If the answer is 799 or less, all three digits are displayed on the hex and binary LEDs.
If the answer is 800 or more, it is stored in memory but not displayed. Instead, C is displayed on the HEX. LED and the ERROR sound is heard. This is because the largest first digit that can be displayed is 7 (binary 111).

A) Key in the program and check it.

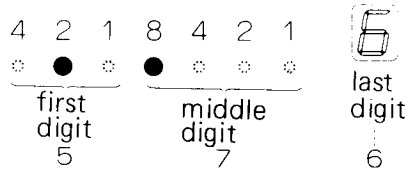
B) Load DECIMAL numbers in 50/51/52.

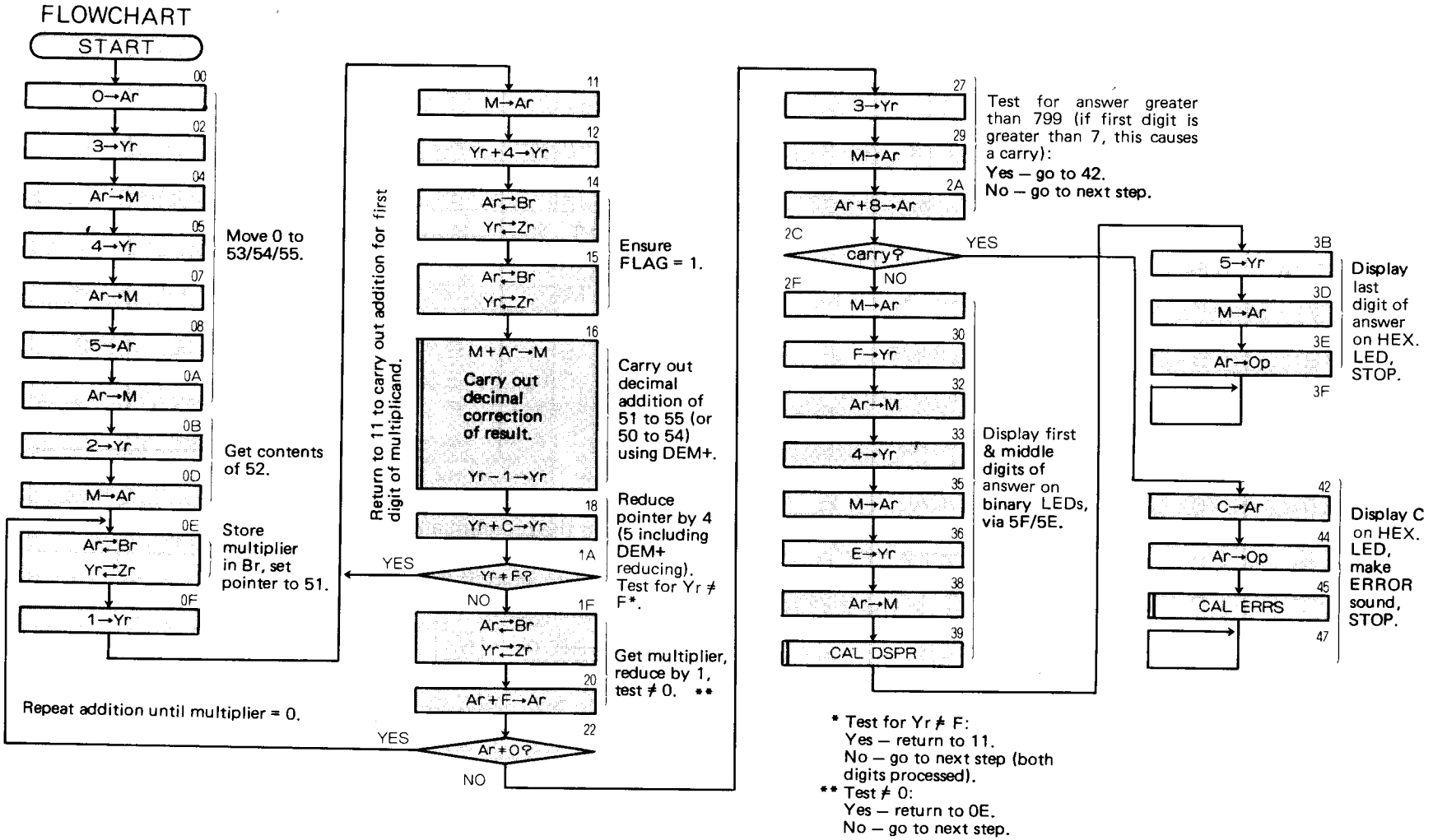
address 50 51 × 52 = 53 54 55

7 2 × 8 = 5 7 6

C) Press RESET, 1, RUN to start the program.

D) Answer is:




FLOWCHART
START
00
0→Ar
02
3→Yr
04
Ar→M
05
4→Yr
07
Ar→M
08
5→Ar
0A
Ar→M
0B
2→Yr
0D
M→Ar
0E
Ar⇄Br
Yr⇄Zr
0F
1→Yr
Move 0 to 53/54/55.
Get contents of 52.
Store multiplier in Br, set pointer to 51.
Repeat addition until multiplier = 0.
Return to 11 to carry out addition for first digit of multiplicand.
11
M→Ar
12
Yr+4→Yr
14
Ar⇄Br
Yr⇄Zr
15
Ar⇄Br
Yr⇄Zr
16
M+Ar→M
Carry out decimal correction of result.
Yr−1→Yr
18
Yr+C→Yr
1A
YES
Yr≠F?
NO
1F
Ar⇄Br
Yr⇄Zr
20
Ar+F→Ar
22
YES
Ar≠0?
NO
Ensure FLAG = 1.
Carry out decimal addition of 51 to 55 (or 50 to 54) using DEM+.
Reduce pointer by 4 (5 including DEM+ reducing). Test for Yr ≠ F*.
Get multiplier, reduce by 1, test ≠ 0. **
27
3→Yr
29
M→Ar
2A
Ar+8→Ar
2C
carry?
YES
NO
2F
M→Ar
30
F→Yr
32
Ar→M
33
4→Yr
35
M→Ar
36
E→Yr
38
Ar→M
39
CAL DSPR
Test for answer greater than 799 (if first digit is greater than 7, this causes a carry):
Yes – go to 42.
No – go to next step.
Display first & middle digits of answer on binary LEDs, via 5F/5E.
3B
5→Yr
3D
M→Ar
3E
Ar→Op
3F
Display last digit of answer on HEX. LED, STOP.
42
C→Ar
44
Ar→Op
45
CAL ERRS
47
Display C on HEX. LED, make ERROR sound, STOP.
* Test for Yr ≠ F:
Yes – return to 11.
No – go to next step (both digits processed).
** Test ≠ 0:
Yes – return to 0E.
No – go to next step.

No.76 Decimal Division using DEM− : 2 Digit ÷ 1 Digit

PROGRAM ①

address	command	machine code
00	T I Y	A
01	⟨C⟩	C
02	T I A	8
03	⟨1⟩	1
04	⌜CAL	E
05	⌞CHNG	5
06	T I A	8
07	⟨0⟩	0
08	T I Y	A
09	⟨B⟩	B
0A	AM	4
0B	A I Y	B
0C	⟨1⟩	1
0D	C I Y	D
0E	⟨0⟩	0
0F	JUMP	F
10	⟨0⟩	0
11	⟨A⟩	A
12	T I Y	A
13	⟨8⟩	8
14	MA	5
15	CH	2
16	T I Y	A
17	⟨9⟩	9
18	MA	5
19	T I Y	A
1A	⟨E⟩	E
1B	AM	4
1C	T I Y	A

②

address	command	machine code
1D	⟨A⟩	A
1E	MA	5
1F	T I Y	A
20	⟨E⟩	E
21	⌜CAL	E
22	⌞DEM−	E
23	MA	5
24	C I A	C
25	⟨0⟩	0
26	JUMP	F
27	⟨3⟩	3
28	⟨6⟩	6
29	⌜CAL	E
2A	⌞CHNG	5
2B	⌜CAL	E
2C	⌞DEM+	F
2D	A I Y	B
2E	⟨1⟩	1
2F	CH	2
30	CH	2
31	⌜CAL	E
32	⌞CHNG	5
33	JUMP	F
34	⟨1⟩	1
35	⟨C⟩	C
36	T I A	8
37	⟨0⟩	0
38	AM	4
39	CH	2

③

address	command	machine code
3A	C I A	C
3B	⟨0⟩	0
3C	JUMP	F
3D	⟨4⟩	4
3E	⟨E⟩	E
3F	T I Y	A
40	⟨A⟩	A
41	MA	5
42	T I Y	A
43	⟨E⟩	E
44	⌜CAL	E
45	⌞DEM+	F
46	T I A	8

④

address	command	machine code
47	⟨0⟩	0
48	AM	4
49	⌜CAL	E
4A	⌞ENDS	7
4B	JUMP	F
4C	⟨4⟩	4
4D	⟨B⟩	B
4E	A I A	9
4F	⟨F⟩	F
50	CH	2
51	JUMP	F
52	⟨2⟩	2
53	⟨9⟩	9

A) Key in the program and check it.

B) Load data at addresses 58/59/5A.

Example:

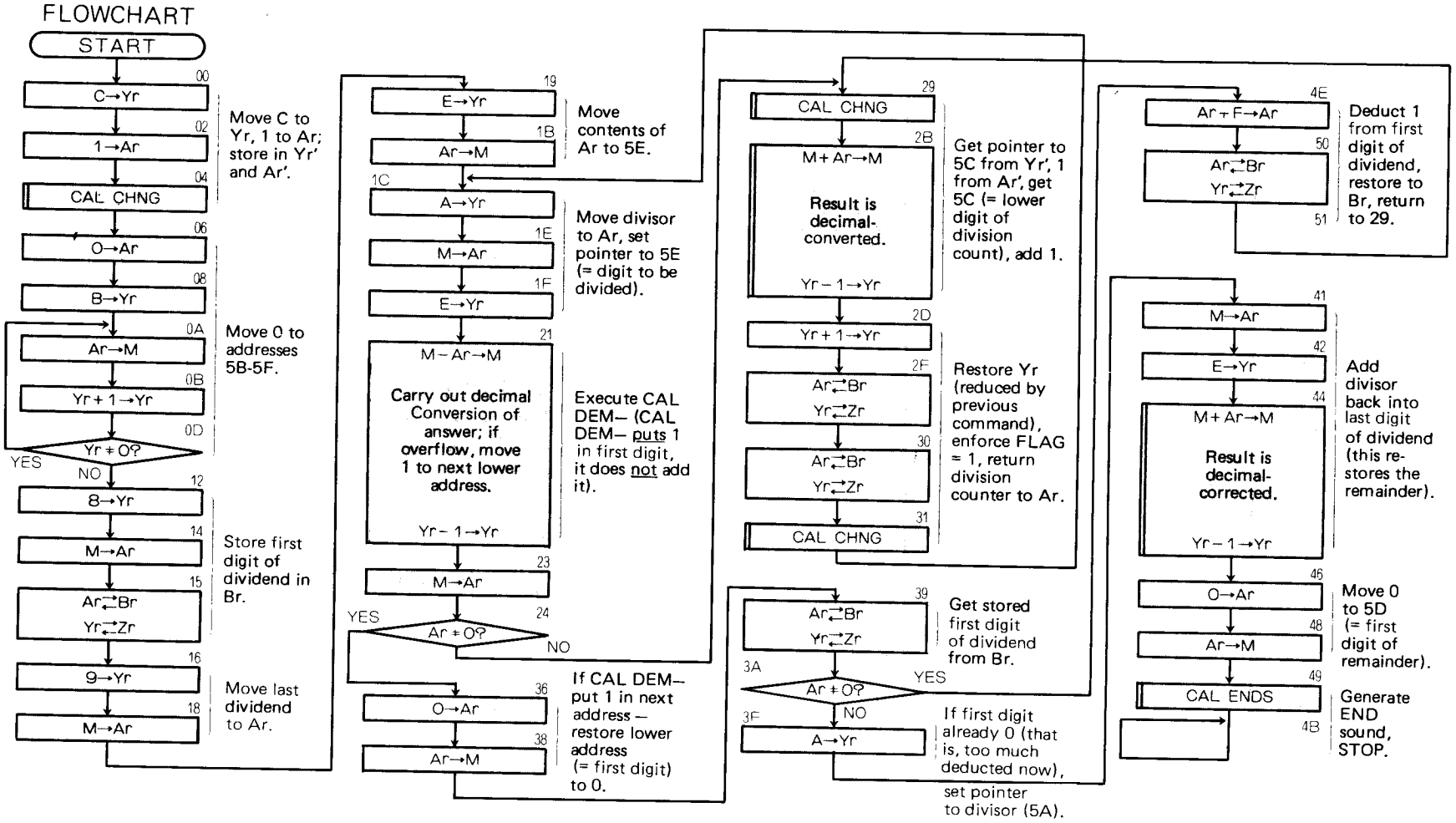
58	59		5A		5B	5C		5D	5E	
7	5	÷	8	=	0	9	decimal answer	0	3	address decimal remainder

C) Press RESET, 1, RUN to start the program. When you hear the END sound press:—

D) 

answer remainder

Do not try to divide a number by 0. It does not work. 5D will always have 0 in it at the end of the program.


FLOWCHART
START
00
C→Yr
02
1→Ar
04
CAL CHNG
06
0→Ar
08
B→Yr
0A
Ar→M
0B
Yr+1→Yr
0D
Yr≠0?
YES
NO
12
8→Yr
14
M→Ar
15
Ar⇄Br
Yr⇄Zr
16
9→Yr
18
M→Ar
Move C to Yr, 1 to Ar; store in Yr' and Ar'.
Move 0 to addresses 5B-5F.
Store first digit of dividend in Br.
Move last dividend to Ar.
19
E→Yr
1B
Ar→M
1C
A→Yr
1E
M→Ar
1F
E→Yr
21
M−Ar→M
Carry out decimal Conversion of answer; if overflow, move 1 to next lower address.
Yr−1→Yr
23
M→Ar
24
Ar≠0?
YES
NO
36
0→Ar
38
Ar→M
Move contents of Ar to 5E.
Move divisor to Ar, set pointer to 5E (= digit to be divided).
Execute CAL DEM− (CAL DEM− puts 1 in first digit, it does not add it).
If CAL DEM− put 1 in next address − restore lower address (= first digit) to 0.
29
CAL CHNG
2B
M+Ar→M
Result is decimal-converted.
Yr−1→Yr
2D
Yr+1→Yr
2F
Ar⇄Br
Yr⇄Zr
30
Ar⇄Br
Yr⇄Zr
31
CAL CHNG
Get pointer to 5C from Yr', 1 from Ar', get 5C (= lower digit of division count), add 1.
Restore Yr (reduced by previous command), enforce FLAG = 1, return division counter to Ar.
39
Ar⇄Br
Yr⇄Zr
3A
Ar≠0?
YES
NO
3F
A→Yr
Get stored first digit of dividend from Br.
If first digit already 0 (that is, too much deducted now), set pointer to divisor (5A).
4E
Ar−F→Ar
50
Ar⇄Br
Yr⇄Zr
51
Deduct 1 from first digit of dividend, restore to Br, return to 29.
41
M→Ar
42
E→Yr
44
M+Ar→M
Result is decimal-corrected.
Yr−1→Yr
46
0→Ar
48
Ar→M
49
CAL ENDS
4B
Add divisor back into last digit of dividend (this restores the remainder).
Move 0 to 5D (= first digit of remainder).
Generate END sound, STOP.

No.77 Add Together Decimal Values in Memory

With this program, various decimal numbers can be added.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨C⟩	C
04	AM	4
05	AIY	B
06	⟨1⟩	1
07	CIY	D
08	⟨0⟩	0
09	JUMP	F
0A	⟨0⟩	0
0B	⟨4⟩	4
0C	TIY	A
0D	⟨C⟩	C
0E	MA	5
0F	CY	3
10	MA	5
11	TIY	A
12	⟨F⟩	F
13	⌜CAL	E
14	⌞DEM+	F
15	TIY	A
16	⟨C⟩	C
17	MA	5
18	AIA	9
19	⟨1⟩	1
1A	AM	4
1B	CIA	C
1C	⟨C⟩	C

address	command	machine code
1D	JUMP	F
1E	⟨0⟩	0
1F	⟨F⟩	F
20	TIY	A
21	⟨F⟩	F
22	MA	5
23	AO	1
24	CH	2
25	TIY	A
26	⟨D⟩	D
27	MA	5
28	TIY	A
29	⟨F⟩	F
2A	AM	4
2B	⌜CAL	E
2C	⌞DSPR	D
2D	CH	2
2E	AM	4
2F	JUMP	F
30	⟨2⟩	2
31	⟨F⟩	F

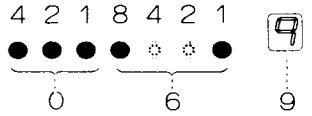
A) Key in the program and check it.

B) Load data (0-9) into addresses 50-5B.

Example:

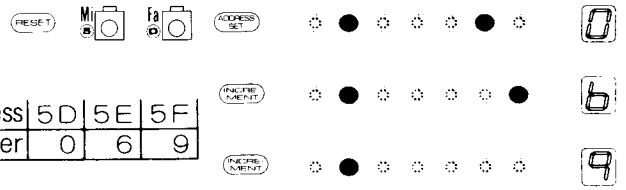
50	51	52	53	54	55	56	57	58	59	5A	5B	address
7	4	3	9	5	8	3	4	8	9	7	2	data

C) Press RESET, 1, RUN to start the program. The answer will then be displayed like this:—



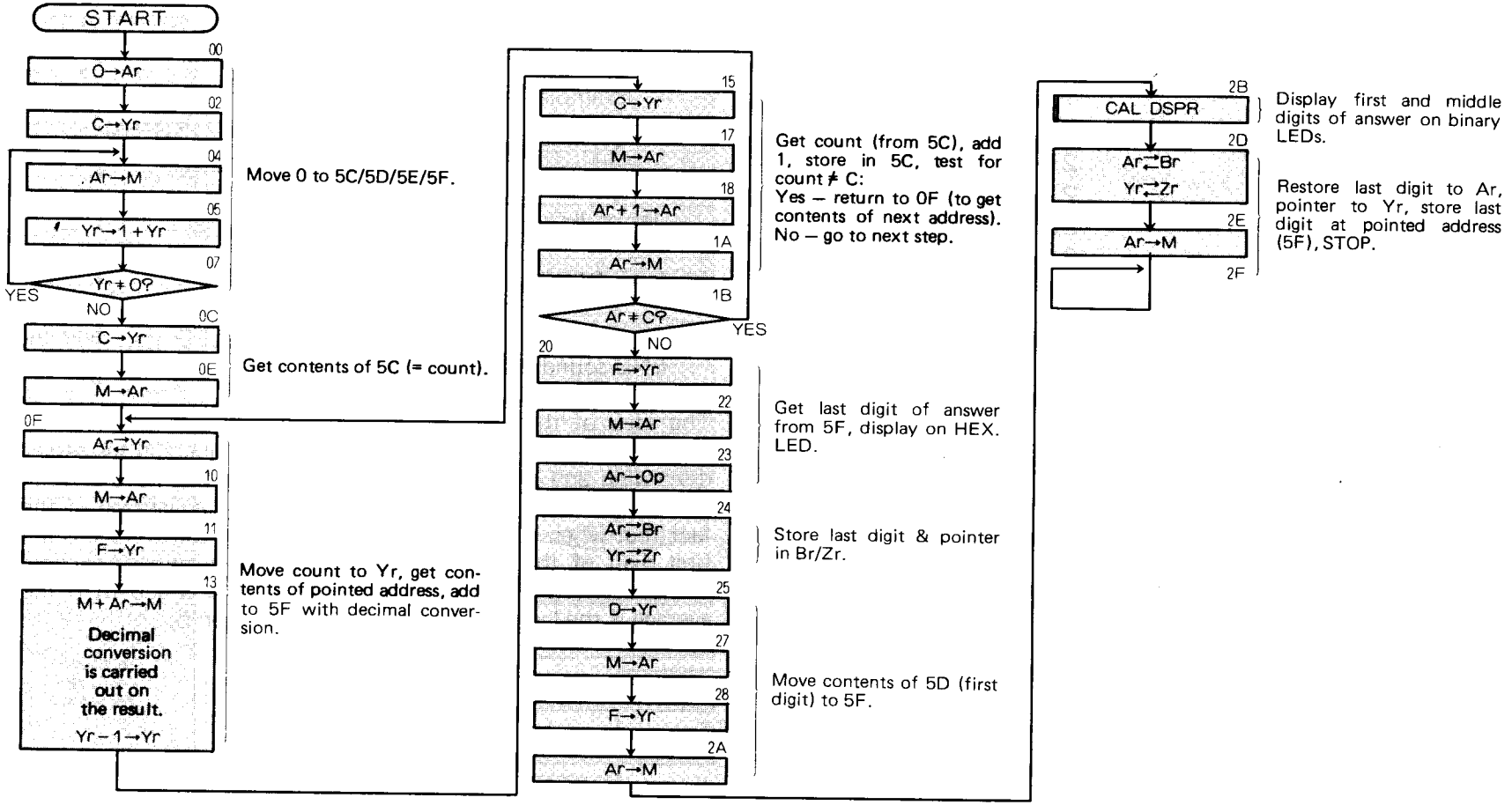
ANSWER 69

D) Then read out the value from memory:—



address	5D	5E	5F
answer	0	6	9

FLOWCHART


START
00
0→Ar
02
C→Yr
04
Ar→M
05
Yr→1+Yr
07
Yr+0?
YES
NO
0C
C→Yr
0E
M→Ar
0F
Ar⇄Yr
10
M→Ar
11
F→Yr
13
M+Ar→M
Decimal conversion is carried out on the result.
Yr−1→Yr
Move 0 to 5C/5D/5E/5F.
Get contents of 5C (= count).
Move count to Yr, get contents of pointed address, add to 5F with decimal conversion.
15
C→Yr
17
M→Ar
18
Ar+1→Ar
1A
Ar→M
1B
Ar+C?
YES
NO
20
F→Yr
22
M→Ar
23
Ar→Op
24
Ar⇄Br
Yr⇄Zr
25
0→Yr
27
M→Ar
28
F→Yr
2A
Ar→M
Get count (from 5C), add 1, store in 5C, test for count ≠ C:
Yes — return to 0F (to get contents of next address).
No — go to next step.
Get last digit of answer from 5F, display on HEX. LED.
Store last digit & pointer in Br/Zr.
Move contents of 5D (first digit) to 5F.
2B
CAL DSPR
2D
Ar⇄Br
Yr⇄Zr
2E
Ar→M
2F
Display first and middle digits of answer on binary LEDs.
Restore last digit to Ar, pointer to Yr, store last digit at pointed address (5F), STOP.

No.78 Calculate Decimal Averages in Memory

This program figures the average of 10 numbers.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨C⟩	C
04	AM	4
05	TIY	A
06	⟨B⟩	B
07	AM	4
08	TIY	A
09	⟨A⟩	A
0A	AM	4
0B	CY	3
0C	MA	5
0D	TIY	A
0E	⟨C⟩	C
0F	⌜CAL	E
10	⌞DEM+	F
11	TIY	A
12	⟨A⟩	A
13	MA	5
14	AIA	9
15	⟨1⟩	1
16	AM	4
17	CIA	C
18	⟨A⟩	A
19	JUMP	F
1A	⟨0⟩	0
1B	⟨B⟩	B
1C	TIY	A

address	command	machine code
1D	⟨C⟩	C
1E	MA	5
1F	AIA	9
20	⟨B⟩	B
21	JUMP	F
22	⟨2⟩	2
23	⟨D⟩	D
24	TIY	A
25	⟨B⟩	B
26	MA	5
27	AO	1
28	⌜CAL	E
29	⌞ENDS	7
2A	JUMP	F
2B	⟨2⟩	2
2C	⟨A⟩	A
2D	TIY	A
2E	⟨B⟩	B
2F	MA	5
30	AIA	9
31	⟨1⟩	1
32	CH	2
33	CH	2
34	JUMP	F
35	⟨2⟩	2
36	⟨7⟩	7

A) Key in the program and check it.

B) Load decimal numbers into 50/59.

Example:

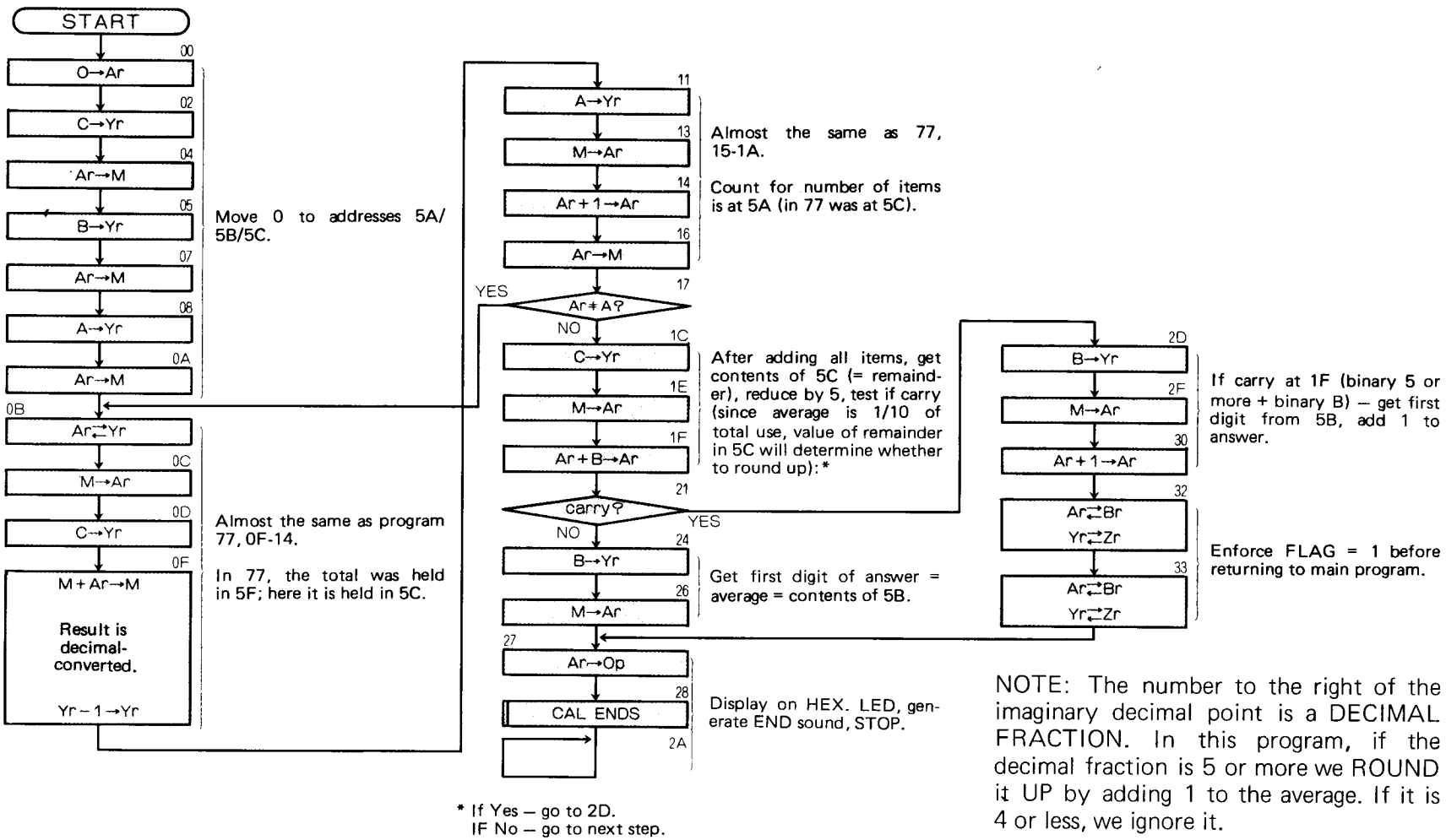
50	51	52	53	54	55	56	57	58	59
3	4	7	1	9	3	5	6	8	2

C) Press RESET, 1, RUN to start the program.

D) The answer is displayed on the HEX. LED 5 and the END sound is generated.

The idea of this program is to obtain a total of a series of numbers and divide it by the number of numbers in the series, discarding any remainder. However, we do not have enough room to do all that; so this program takes advantage of the fact that when the number of numbers is 10 the average is the first digit of the total value — we put an imaginary decimal point between the first and last digits. One is added to the average if the last digit equals or is greater than 5.

FLOWCHART



* If Yes — go to 2D.
IF No — go to next step.

NOTE: The number to the right of the imaginary decimal point is a DECIMAL FRACTION. In this program, if the decimal fraction is 5 or more we ROUND it UP by adding 1 to the average. If it is 4 or less, we ignore it.

No.79 Transmit Morse Code from Data stored in Memory

This program reads codes from 50-5F, converts them into Morse, and generates Morse code sounds.
Refer to program 7 for International Morse Code symbols.

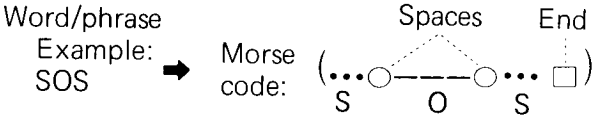
PROGRAM

address	command	machine code	address	command	machine code
00	TIY	A	21	CH	2
01	⟨0⟩	0	22	TIA	8
02	MA	5	23	⟨6⟩	6
03	CIA	C	24	⌜CAL	E
04	⟨F⟩	F	25	⌞TIMR	C
05	JUMP	F	26	CH	2
06	⟨0⟩	0	27	AIY	B
07	⟨F⟩	F	28	⟨1⟩	1
08	TIA	8	29	MA	5
09	⟨9⟩	9	2A	JUMP	F
0A	⌜CAL	E	2B	⟨0⟩	0
0B	⌞TIMR	C	2C	⟨3⟩	3
0C	JUMP	F	2D	⌜CAL	E
0D	⟨0⟩	0	2E	⌞SIFT	6
0E	⟨0⟩	0	2F	JUMP	F
0F	CIA	C	30	⟨3⟩	3
10	⟨E⟩	E	31	⟨D⟩	D
11	JUMP	F	32	CH	2
12	⟨1⟩	1	33	TIA	8
13	⟨7⟩	7	34	⟨2⟩	2
14	JUMP	F	35	JUMP	F
15	⟨1⟩	1	36	⟨2⟩	2
16	⟨4⟩	4	37	⟨4⟩	4
17	⌜CAL	E	38	⌜CAL	E
18	⌞SIFT	6	39	⌞LONS	A
19	JUMP	F	3A	JUMP	F
1A	⟨2⟩	2	3B	⟨2⟩	2
1B	⟨D⟩	D	3C	⟨7⟩	7
1C	⌜CAL	E	3D	⌜CAL	E
1D	⌞SIFT	6	3E	⌞SHTS	9
1E	JUMP	F	3F	JUMP	F
1F	⟨3⟩	3	40	⟨2⟩	2
20	⟨8⟩	8	41	⟨7⟩	7

code	morse
0	→ short sound
1	→ long sound
2	→ space between characters
3	→ space between words
E	→ end
F	→ repeat

↘ These codes are stored at 50-5F.

Word/phrase
Example:
SOS ➡ Morse code:



data ⬇

Example:

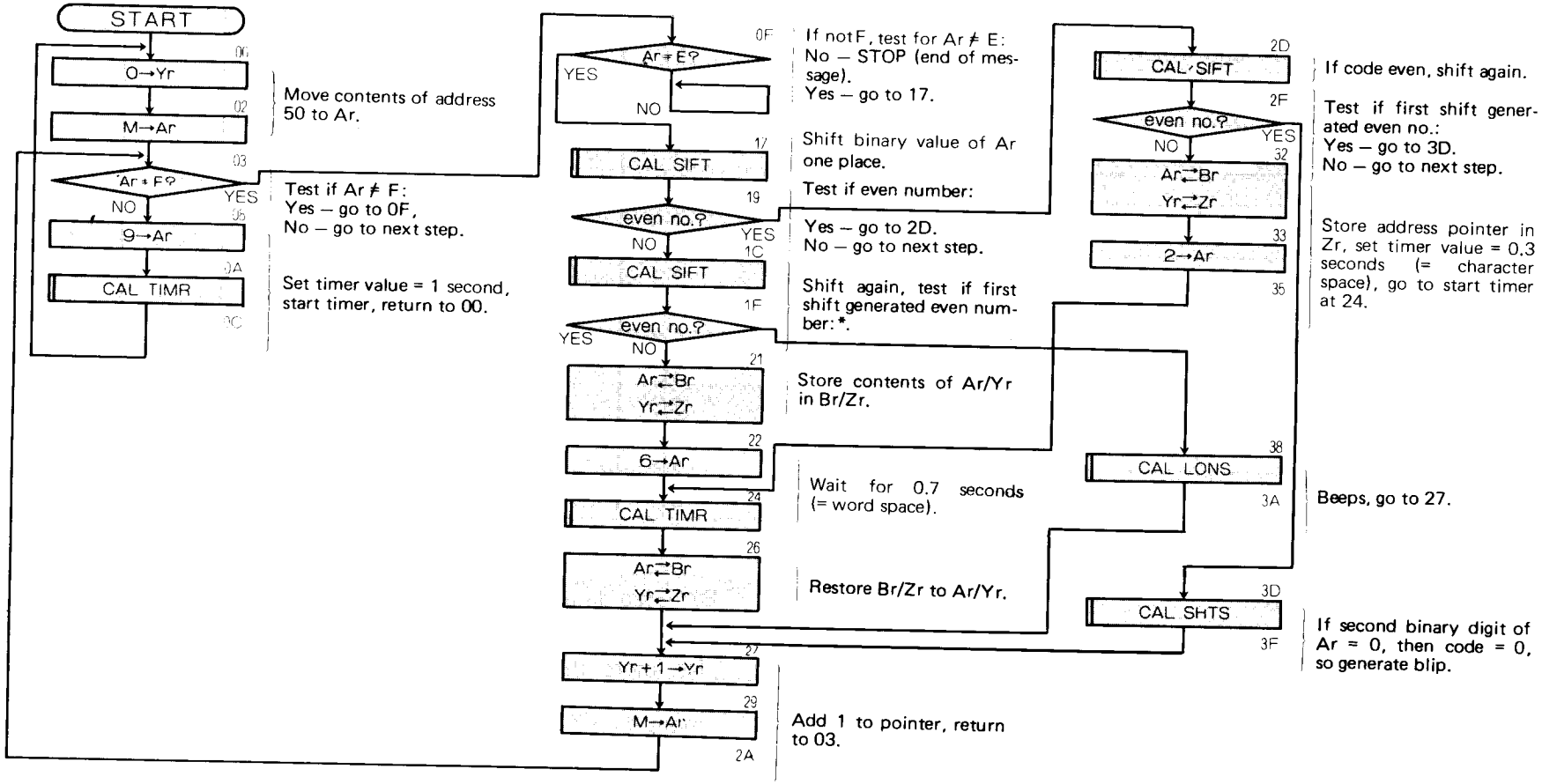
Address	50	51	52	53	54	55	56	57	58	59	5A	5B
Data	0	0	0	2	1	1	1	2	0	0	0	E

You will hear the Morse code for S O S once when the program is run.

A) Load the program and Morse code and check it.

B) Press RESET, 1, RUN to start the program.

FLOWCHART



* Test if first shift generated even number:
Yes – go to 38 (code in memory was 1).
No – go to next step.

No.80 Countdown timer (Max. 7 mins. 59 secs.)

Before running this program, load seconds at addresses 51/52, and minutes at 50. When the program is run, the display shows the period of time left and generates the END sound when the time has expired.

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	TIY	A
04	⟨F⟩	F
05	AM	4
06	CH	2
07	TIY	A
08	⟨1⟩	1
09	MA	5
0A	TIY	A
0B	⟨E⟩	E
0C	AM	4
0D	CAL	E
0E	CHNG	5
0F	TIA	8
10	⟨9⟩	9
11	CH	2
12	TIY	A
13	⟨2⟩	2
14	MA	5
15	AO	1
16	CAL	E
17	DSPR	D
18	CH	2

②

address	command	machine code
19	CAL	E
1A	TIMR	C
1B	CH	2
1C	CIA	C
1D	⟨0⟩	0
1E	JUMP	F
1F	⟨3⟩	3
20	⟨6⟩	6
21	CAL	E
22	CHNG	5
23	CIA	C
24	⟨0⟩	0
25	JUMP	F
26	⟨3⟩	3
27	⟨C⟩	C
28	CH	2
29	CIA	C
2A	⟨0⟩	0
2B	JUMP	F
2C	⟨4⟩	4
2D	⟨6⟩	6
2E	AO	1
2F	CAL	E
30	DSPR	D

③

address	command	machine code
31	CAL	E
32	ENDS	7
33	JUMP	F
34	⟨3⟩	3
35	⟨3⟩	3
36	AIA	9
37	⟨F⟩	F
38	AO	1
39	JUMP	F
3A	⟨1⟩	1
3B	⟨6⟩	6
3C	AIA	9
3D	⟨F⟩	F
3E	AM	4
3F	CAL	E
40	CHNG	5

④

address	command	machine code
41	TIA	8
42	⟨9⟩	9
43	JUMP	F
44	⟨1⟩	1
45	⟨5⟩	5
46	AIA	9
47	⟨F⟩	F
48	AM	4
49	CH	2
4A	TIA	8
4B	⟨5⟩	5
4C	JUMP	F
4D	⟨3⟩	3
4E	⟨E⟩	E

A) Key in the program and check it.

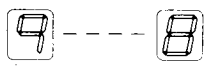
B) Load data into addresses 50/51/52.

Example: 

No more than 7 minutes and 59 seconds can be stored because the first digit on the binary LEDs cannot display more than 7 (111).

C) Press RESET, 1, RUN to start the program.

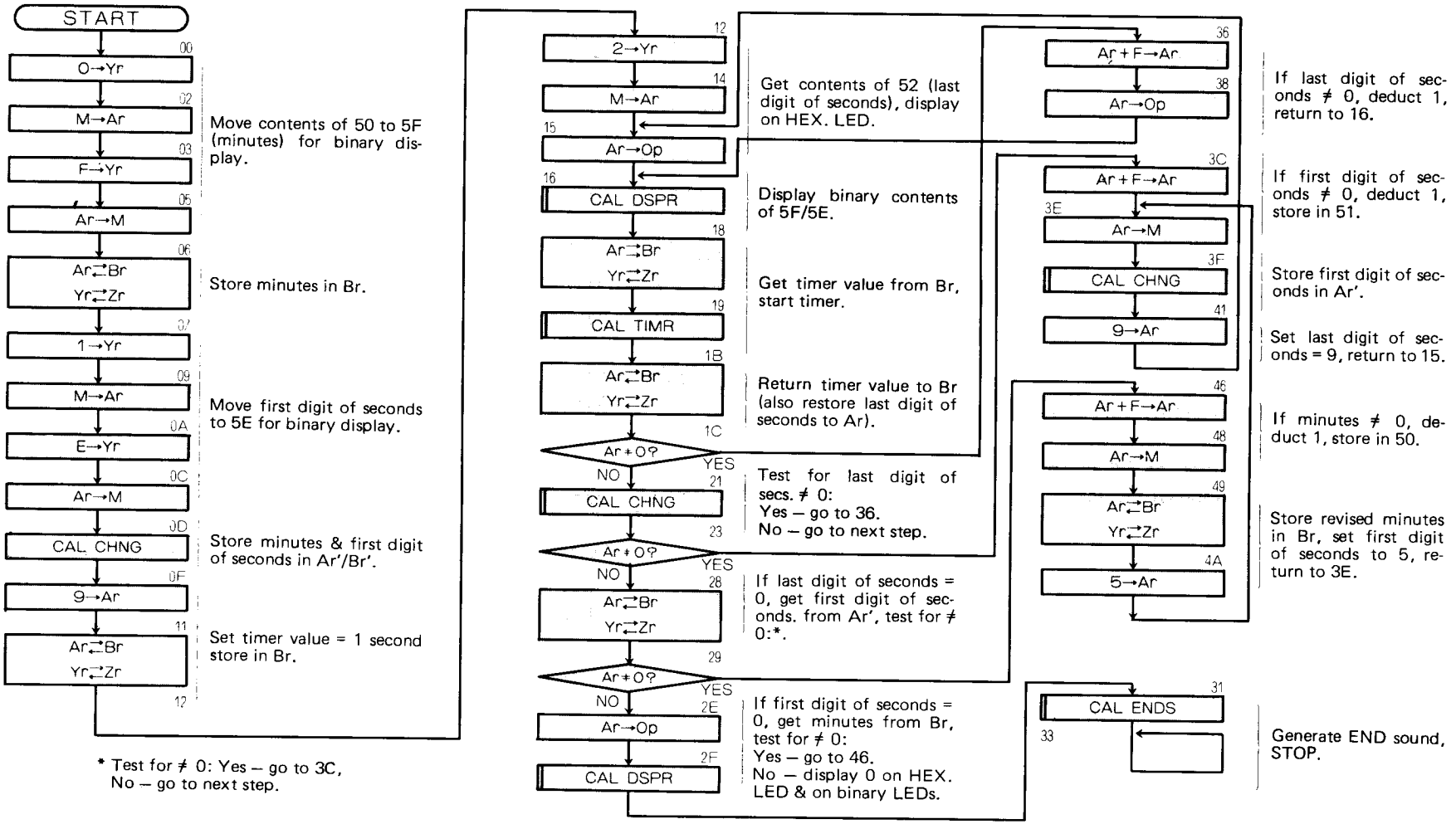
7 minutes 3 ---------- 9 seconds



Decreases once per seconds

The time value is reduced second by second from the starting time set under B above until zero is reached.

FLOWCHART


START
00
0→Yr
02
M→Ar
03
F→Yr
05
Ar→M
06
Ar⇄Br
Yr⇄Zr
07
1→Yr
09
M→Ar
0A
E→Yr
0C
Ar→M
0D
CAL CHNG
0F
9→Ar
11
Ar⇄Br
Yr⇄Zr
12
Move contents of 50 to 5F (minutes) for binary display.
Store minutes in Br.
Move first digit of seconds to 5E for binary display.
Store minutes & first digit of seconds in Ar'/Br'.
Set timer value = 1 second store in Br.
* Test for ≠ 0: Yes – go to 3C, No – go to next step.
12
2→Yr
14
M→Ar
15
Ar→Op
16
CAL DSPR
18
Ar⇄Br
Yr⇄Zr
19
CAL TIMR
1B
Ar⇄Br
Yr⇄Zr
1C
Ar≠0?
YES
NO
21
CAL CHNG
23
Ar≠0?
YES
NO
28
Ar⇄Br
Yr⇄Zr
29
Ar≠0?
YES
NO
2E
Ar→Op
2F
CAL DSPR
Get contents of 52 (last digit of seconds), display on HEX. LED.
Display binary contents of 5F/5E.
Get timer value from Br, start timer.
Return timer value to Br (also restore last digit of seconds to Ar).
Test for last digit of secs. ≠ 0:
Yes – go to 36.
No – go to next step.
If last digit of seconds = 0, get first digit of seconds. from Ar', test for ≠ 0:*.
If first digit of seconds = 0, get minutes from Br, test for ≠ 0:
Yes – go to 46.
No – display 0 on HEX. LED & on binary LEDs.
36
Ar+F→Ar
38
Ar→Op
3C
Ar+F→Ar
3E
Ar→M
3F
CAL CHNG
41
9→Ar
46
Ar+F→Ar
48
Ar→M
49
Ar⇄Br
Yr⇄Zr
4A
5→Ar
31
CAL ENDS
33
If last digit of seconds ≠ 0, deduct 1, return to 16.
If first digit of seconds ≠ 0, deduct 1, store in 51.
Store first digit of seconds in Ar'.
Set last digit of seconds = 9, return to 15.
If minutes ≠ 0, deduct 1, store in 50.
Store revised minutes in Br, set first digit of seconds to 5, return to 3E.
Generate END sound, STOP.

No.81 Turn on Binary LEDs with accompanying musical notes

CAL SUND is used to generate notes.

CAL SUND COMMAND (CALI SoUND)

This command plays a note at a pitch that is determined by the contents of Ar. All notes are the same length. When a note is played, binary LED 3 is turned on.

Value of Ar	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Note	silent		ti		re		fa		la		do		mi		sol	
		la		do		mi		sol		ti		re		fa		silent

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

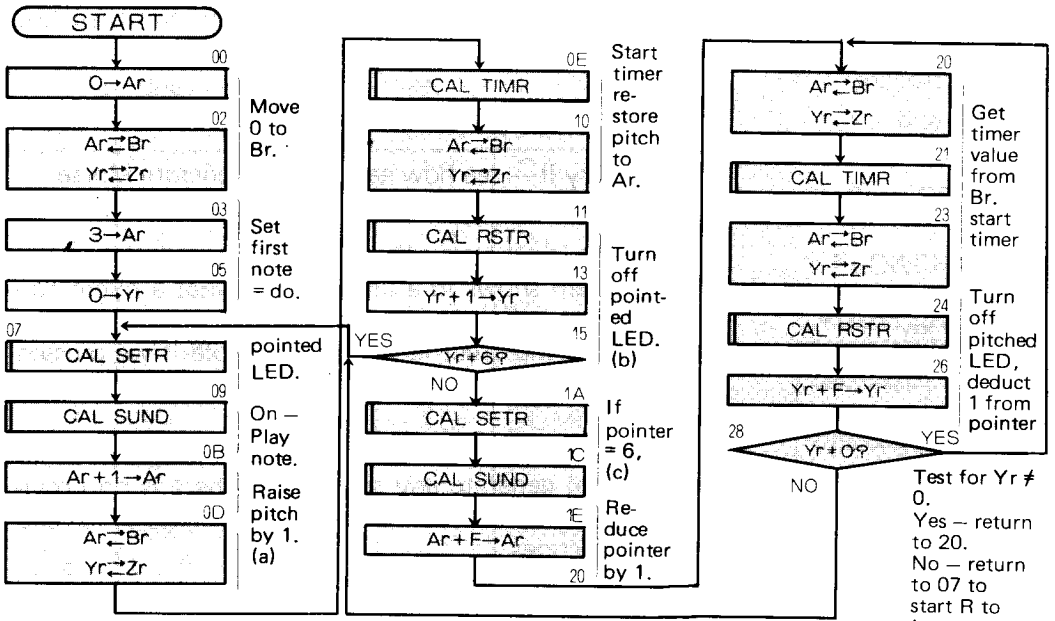
C) If the starting pitch at 04 is changed, the program will play a different range of notes.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	TIA	8
04	⟨3⟩	3
05	TIY	A
06	⟨0⟩	0
07	⌈CAL	E
08	⌊SETR	1
09	⌈CAL	E
0A	⌊SUND	B
0B	AIA	9
0C	⟨1⟩	1
0D	CH	2
0E	⌈CAL	E
0F	⌊TIMR	C
10	CH	2
11	⌈CAL	E
12	⌊RSTR	2
13	AIY	B
14	⟨1⟩	1
15	CIY	D
16	⟨6⟩	6
17	JUMP	F
18	⟨0⟩	0
19	⟨7⟩	7
1A	⌈CAL	E
1B	⌊SETR	1

address	command	machine code
1C	⌈CAL	E
1D	⌊SUND	B
1E	AIA	9
1F	⟨F⟩	F
20	CH	2
21	⌈CAL	E
22	⌊TIMR	C
23	CH	2
24	⌈CAL	E
25	⌊RSTR	2
26	AIY	B
27	⟨F⟩	F
28	CIY	D
29	⟨0⟩	0
2A	JUMP	F
2B	⟨1⟩	1
2C	⟨A⟩	A
2D	JUMP	F
2E	⟨0⟩	0
2F	⟨7⟩	7

FLOWCHART



(a) Store pitch in Br, get timer value.
(b) Add 1 to pointer.
(c) If pointer = 6, turn on LED, play note.

No.82 Morse Code Input Controller

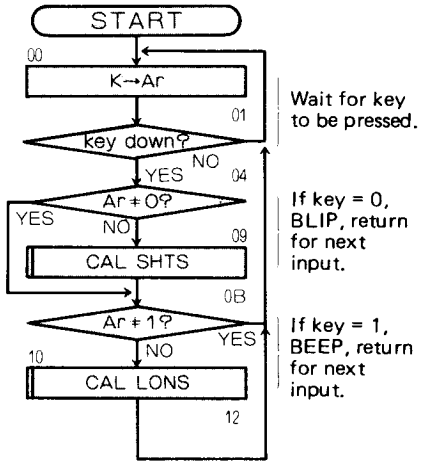
This program ensures that valid Morse signals are generated by enforcing the use of two keys only, 0 for a short sound, 1 for a long sound.

Short sound: ⊙◯ Long sound: 1◯

PROGRAM

address	command	machine code
00	KA	0
01	JUMP	F
02	⟨0⟩	0
03	⟨0⟩	0
04	CIA	C
05	⟨0⟩	0
06	JUMP	F
07	⟨0⟩	0
08	⟨B⟩	B
09	⌜CAL	E
0A	⌞SHTS	9
0B	CIA	C
0C	⟨1⟩	1
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨0⟩	0
10	⌜CAL	E
11	⌞LONS	A
12	JUMP	F
13	⟨0⟩	0
14	⟨0⟩	0

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

C) Now press either ⊙◯= short sound or 1◯= long sound

Try it — see how easy it is to generate Morse.

Morse code input controller

This program shows in a limited way what a micro can do to reduce the chances of operator error (meaning pressing the incorrect key — for example if we pressed (2) when we meant to press (1)).

Only 0 and 1 keys are relevant in this program; other keys do not generate any sound. But there is no provision for automatic character and word spaces or for end and repeat codes.

Examine the list of codes in program no. 79 again. Perhaps you could improve program no. 82 by making provisions for the extra codes. Also consider waiting for a key to be released before looking for the next one; is that a good idea that you could incorporate into this program? Can you think of any other ideas that would help to make Morse transmission as easy and accurate as possible?

No.83 Metronome

A metronome gives a steady beat at different speeds and in different rhythms. In this program, the contents of 50 determine the rhythm and 51, the speed.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	CIA	C
04	⟨0⟩	0
05	JUMP	F
06	⟨0⟩	0
07	⟨B⟩	B
08	JUMP	F
09	⟨0⟩	0
0A	⟨0⟩	0
0B	CIA	C
0C	⟨1⟩	1
0D	JUMP	F
0E	⟨2⟩	2
0F	⟨3⟩	3
10	CH	2
11	CAL	E
12	SHTS	9
13	TIY	A
14	⟨1⟩	1
15	MA	5

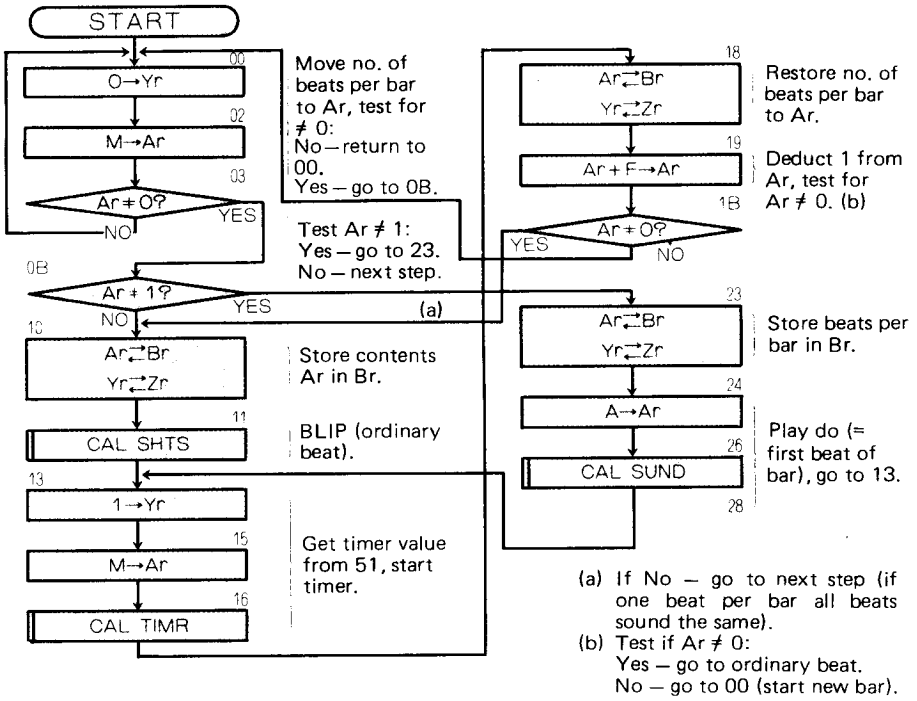
address	command	machine code
16	CAL	E
17	TIMR	C
18	CH	2
19	AIA	9
1A	⟨F⟩	F
1B	CIA	C
1C	⟨0⟩	0
1D	JUMP	F
1E	⟨1⟩	1
1F	⟨0⟩	0
20	JUMP	F
21	⟨0⟩	0
22	⟨0⟩	0
23	CH	2
24	TIA	8
25	⟨A⟩	A
26	CAL	E
27	SUND	B
28	JUMP	F
29	⟨1⟩	1
2A	⟨3⟩	3

A) Key in the program and check it.

B) Load data into 50 (beats per bar) and into 51 (speed):

C) Press RESET, 1, RUN to start the program.

FLOWCHART



No.84 Guessing Game: Odd/Even Numbers

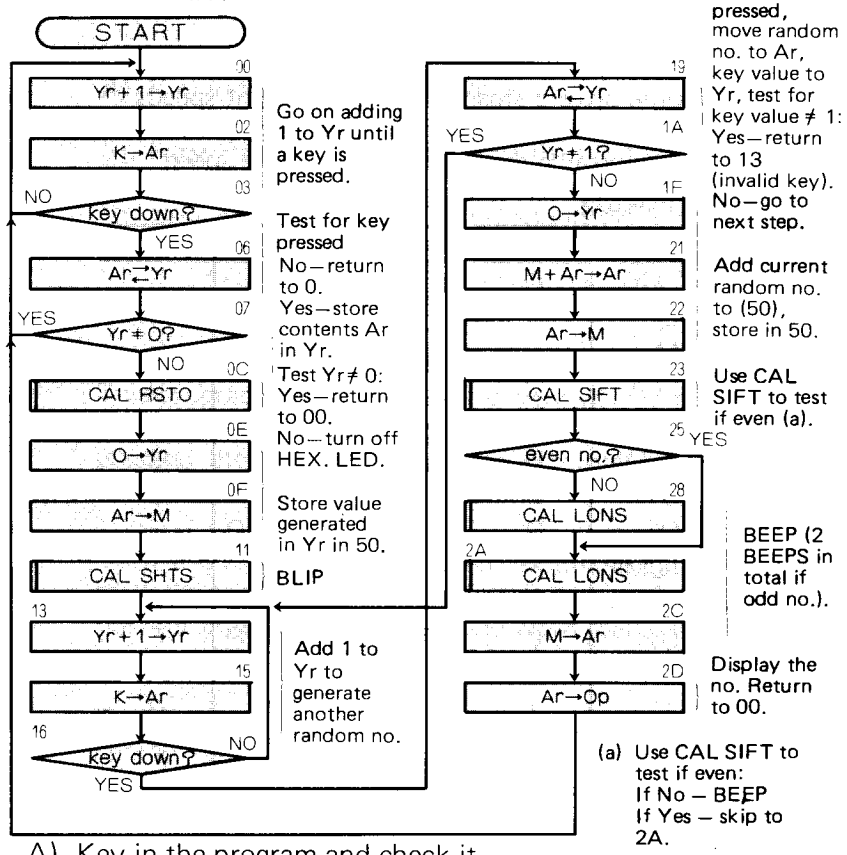
In this program the (0) and (1) keys are pressed in turn to select two random numbers. If the sum of these numbers is an odd number you will hear two long notes; if even, one long note.

PROGRAM

address	command	machine code
00	A I Y	B
01	⟨1⟩	1
02	KA	0
03	JUMP	F
04	⟨0⟩	0
05	⟨0⟩	0
06	CY	3
07	C I Y	D
08	⟨0⟩	0
09	JUMP	F
0A	⟨0⟩	0
0B	⟨0⟩	0
0C	⌜CAL	E
0D	⌞RSTO	0
0E	T I Y	A
0F	⟨0⟩	0
10	AM	4
11	⌜CAL	E
12	⌞SHTS	9
13	A I Y	B
14	⟨1⟩	1
15	KA	0
16	JUMP	F
17	⟨1⟩	1
18	⟨3⟩	3

address	command	machine code
19	CY	3
1A	C I Y	D
1B	⟨1⟩	1
1C	JUMP	F
1D	⟨1⟩	1
1E	⟨3⟩	3
1F	T I Y	A
20	⟨0⟩	0
21	M+	6
22	AM	4
23	⌜CAL	E
24	⌞SIFT	6
25	JUMP	F
26	⟨2⟩	2
27	⟨A⟩	A
28	⌜CAL	E
29	⌞LONS	A
2A	⌜CAL	E
2B	⌞LONS	A
2C	MA	5
2D	AO	1
2E	JUMP	F
2F	⟨0⟩	0
30	⟨0⟩	0

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1, RUN to start. Program will wait for a key to be pressed.

C) Press (0) to get the first number; a sound is generated. Press (1) to get the second random number. If the sum of the two numbers is even you will hear just one long sound. If the sum is odd you will hear two long sounds. The game may be repeated as often as you like by pressing (0) and (1) in turn.
Is the figure displayed on the HEX. LED supposed to be the sum—? Do you ever know the identity of the random numbers?

No.85 Guessing Game: Large or Small?

In this program you have to guess whether a number in memory is greater than or less than 8. If you think it is greater you must press (1); if smaller you press (0). If you are correct you will hear the END sound; otherwise the ERROR sound will be heard.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	MA	5
03	AIA	9
04	⟨8⟩	8
05	JUMP	F
06	⟨2⟩	2
07	⟨1⟩	1
08	KA	0
09	JUMP	F
0A	⟨0⟩	0
0B	⟨8⟩	8
0C	CIA	C
0D	⟨0⟩	0
0E	JUMP	F
0F	⟨1⟩	1
10	⟨C⟩	C
11	⌜CAL	E
12	⌞ENDS	7
13	AIY	B
14	⟨1⟩	1
15	KA	0
16	JUMP	F

address	command	machine code
17	⟨0⟩	0
18	⟨2⟩	2
19	JUMP	F
1A	⟨1⟩	1
1B	⟨5⟩	5
1C	⌜CAL	E
1D	⌞ERRS	8
1E	JUMP	F
1F	⟨1⟩	1
20	⟨3⟩	3
21	KA	0
22	JUMP	F
23	⟨2⟩	2
24	⟨1⟩	1
25	CIA	C
26	⟨1⟩	1
27	JUMP	F
28	⟨1⟩	1
29	⟨C⟩	C
2A	JUMP	F
2B	⟨1⟩	1
2C	⟨1⟩	1

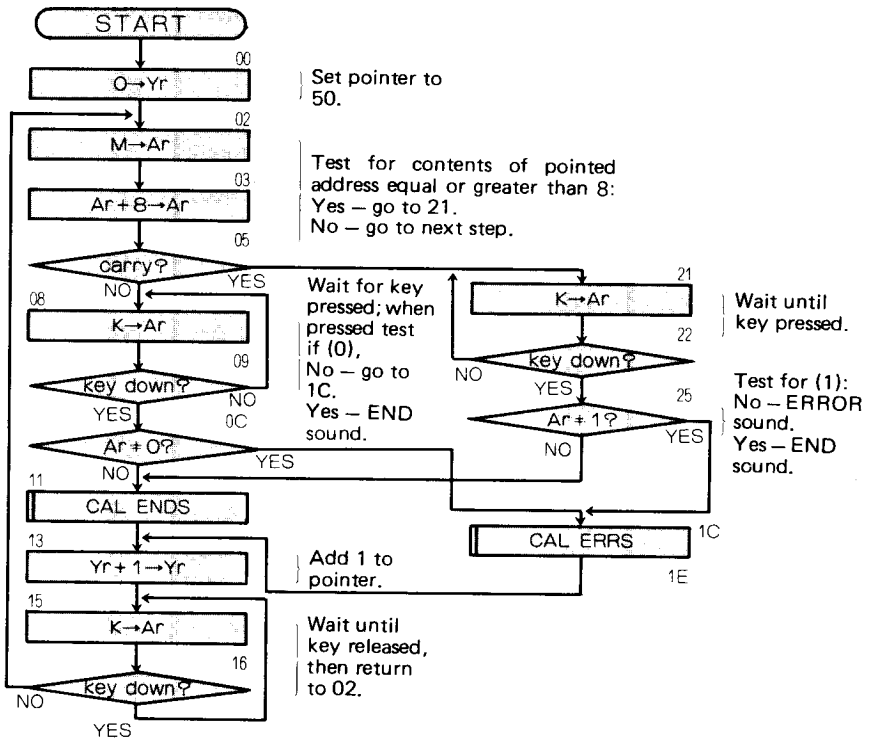
A) Key in the program and check it.

B) Load various numbers into memory at 50-5F.

C) Press RESET, 1, RUN to start the program.

D) Now press either 1 (if you think the number in memory is 8 or greater than 8), or press 0 if less than 8. The number you are guessing is the number in the memory currently pointed to. When you have guessed (correctly or incorrectly), the program points to the next address and you can try again. When you have tried to guess what is in 5F the program points to 50 once more. Make the game more difficult by asking somebody else to load the numbers for you!

FLOWCHART



No.86 Guess the Number Game

One of the addresses 50-5F has 0 in it — this is the trap. When the game starts and you press a key, the program stops at a random address. If the sum of the contents of that address and the key that you pressed equals F you win and hear the END sound. If it does not, the program stops at the address pointed to by the previous sum. If you land in the TRAP (the address containing 0), you lose the game and hear ERROR.

PROGRAM

address	command	machine code
00	A I Y	B
01	⟨1⟩	1
02	KA	0
03	JUMP	F
04	⟨0⟩	0
05	⟨0⟩	0
06	KA	0
07	JUMP	F
08	⟨0⟩	0
09	⟨6⟩	6
0A	KA	0
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨1⟩	1
0E	JUMP	F
0F	⟨0⟩	0
10	⟨A⟩	A
11	⌜CAL	E
12	⌞SHTS	9
13	M+	6
14	AO	1
15	CY	3
16	C I Y	D

address	command	machine code
17	⟨F⟩	F
18	JUMP	F
19	⟨2⟩	2
1A	⟨0⟩	0
1B	⌜CAL	E
1C	⌞ENDS	7
1D	JUMP	F
1E	⟨1⟩	1
1F	⟨D⟩	D
20	MA	5
21	C I A	C
22	⟨0⟩	0
23	JUMP	F
24	⟨0⟩	0
25	⟨6⟩	6
26	⌜CAL	E
27	⌞ERRS	8
28	JUMP	F
29	⟨2⟩	2
2A	⟨8⟩	8

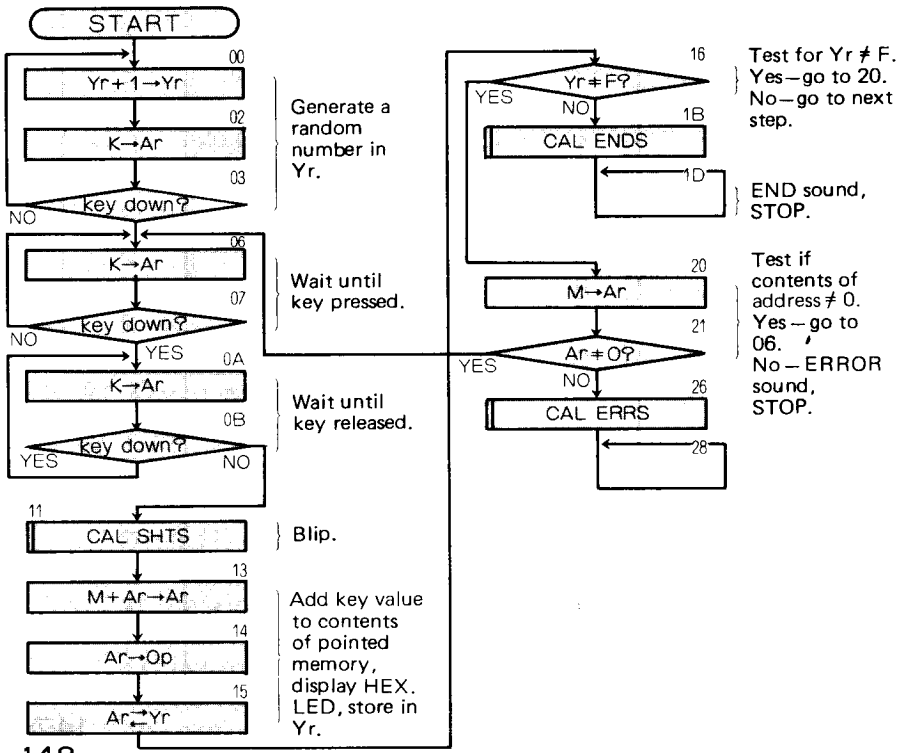
A) Key in the program and check it.

B) Load various numbers into 50-5F including a 0 in one of them, for example:

C) Press RESET, 1, RUN to start. At first the HEX. LED will be off, but as soon as any key is pressed an address selected at random will be displayed. Now press keys until you hit the winning combination — or fall into the trap.

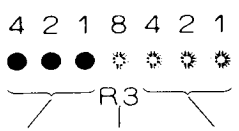
D) If the sum is F, you hear the END sound. If the address has 0 in it, you hear the error sound.

FLOWCHART



No.87 Reflex Tester

This game is for two players. The middle binary LED blinks on and off. When the LED goes off, each player must try to press their key first before it comes back on (use 0 and 3 keys). If a player is successful, he scores a point. The binary LEDs light to keep score. The first person to score 7 (all three binary LEDs lit) wins.
But be careful! If you press your key at the wrong time (while the middle LED is still lit), then you lose whatever points you have scored. Your binary LEDs that were lit will go off.



points
7 points when all your LEDs light.

Score for (0), light, score for (3)

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.
A sound is generated each time a player presses 0 or 3. When the game is over, the sound continues until you press RESET.

C) To play again, press RESET, 1, RUN.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	TIY	A
03	⟨F⟩	F
04	AM	4
05	TIY	A
06	⟨E⟩	E
07	AM	4
08	TIY	A
09	⟨3⟩	3
0A	AM	4
0B	AIY	B
0C	⟨F⟩	F
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨E⟩	E
10	CAL	E
11	DSPR	D
12	CAL	E
13	TIMR	C

②

address	command	machine code
14	TIY	A
15	⟨3⟩	3
16	MA	5
17	CAL	E
18	CMPL	4
19	AM	4
1A	CAL	E
1B	SIFT	6
1C	CAL	E
1D	SETR	1
1E	TIA	8
1F	⟨0⟩	0
20	CAL	E
21	TIMR	C
22	KA	0
23	JUMP	F
24	⟨0⟩	0
25	⟨B⟩	B
26	TIY	A
27	⟨3⟩	3

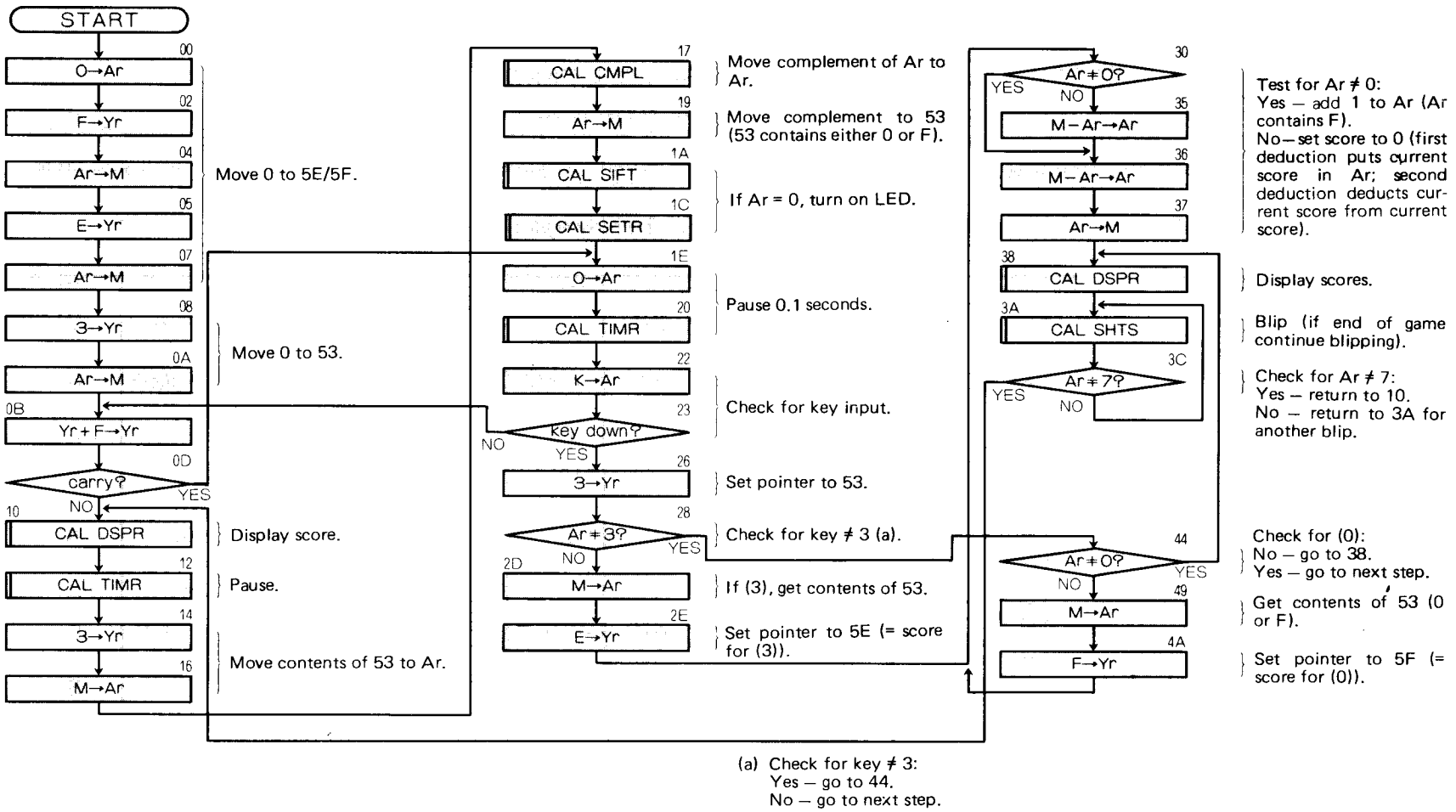
③

address	command	machine code
28	CIA	C
29	⟨3⟩	3
2A	JUMP	F
2B	⟨4⟩	4
2C	⟨4⟩	4
2D	MA	5
2E	TIY	A
2F	⟨E⟩	E
30	CIA	C
31	⟨0⟩	0
32	JUMP	F
33	⟨3⟩	3
34	⟨6⟩	6
35	M−	7
36	M−	7
37	AM	4
38	CAL	E
39	DSPR	D
3A	CAL	E
3B	SHTS	9

④

address	command	machine code
3C	CIA	C
3D	⟨7⟩	7
3E	JUMP	F
3F	⟨1⟩	1
40	⟨0⟩	0
41	JUMP	F
42	⟨3⟩	3
43	⟨A⟩	A
44	CIA	C
45	⟨0⟩	0
46	JUMP	F
47	⟨3⟩	3
48	⟨8⟩	8
49	MA	5
4A	TIY	A
4B	⟨F⟩	F
4C	JUMP	F
4D	⟨3⟩	3
4E	⟨0⟩	0

FLOWCHART



(a) Check for key ≠ 3:
Yes – go to 44.
No – go to next step.

No.88 "Blackjack" Card Game

You may imagine yourself in a casino as you play this program because it is a variation of the popular card game called Blackjack. The first key you press selects your "blackjack" or end result card (which you must guess). It can be any number 0-F. The next key you press chooses your first "hit" card which is displayed on the HEX. LED.

The idea in this version of Blackjack is to keep drawing cards until the total value is equal to or exceeds your "blackjack" card without exceeding the limit—F. Each time you draw a card (press a key), a new total is displayed. When you think your total is equal to or greater than your "blackjack" card, press 0 to end the game. If you are correct, you will hear the END sound. If not (or if you have exceeded the F limit), you will hear the ERROR sound.

PROGRAM ①

address	command	machine code
00	A I Y	B
01	⟨1⟩	1
02	KA	0
03	JUMP	F
04	⟨0⟩	0
05	⟨0⟩	0
06	CY	3
07	T I Y	A
08	⟨0⟩	0
09	AM	4
0A	⌜CAL	E
0B	⌞SHTS	9
0C	KA	0
0D	JUMP	F
0E	⟨1⟩	1
0F	⟨3⟩	3
10	JUMP	F

②

address	command	machine code
11	⟨0⟩	0
12	⟨C⟩	C
13	A I Y	B
14	⟨1⟩	1
15	KA	0
16	JUMP	F
17	⟨1⟩	1
18	⟨3⟩	3
19	CY	3
1A	T I Y	A
1B	⟨1⟩	1
1C	AM	4
1D	AO	1
1E	⌜CAL	E
1F	⌞SHTS	9
20	KA	0
21	JUMP	F

③

address	command	machine code
22	⟨2⟩	2
23	⟨7⟩	7
24	JUMP	F
25	⟨2⟩	2
26	⟨0⟩	0
27	A I Y	B
28	⟨1⟩	1
29	KA	0
2A	JUMP	F
2B	⟨2⟩	2
2C	⟨7⟩	7
2D	C I A	C
2E	⟨0⟩	0
2F	JUMP	F
30	⟨3⟩	3
31	⟨F⟩	F
32	CY	3
33	MA	5
34	T I Y	A
35	⟨1⟩	1
36	M−	7
37	JUMP	F
38	⟨4⟩	4

④

address	command	machine code
39	⟨A⟩	A
3A	⌜CAL	E
3B	⌞ENDS	7
3C	JUMP	F
3D	⟨3⟩	3
3E	⟨C⟩	C
3F	CY	3
40	T I Y	A
41	⟨1⟩	1
42	M+	6
43	JUMP	F
44	⟨4⟩	4
45	⟨A⟩	A
46	AM	4
47	JUMP	F
48	⟨1⟩	1
49	⟨D⟩	D
4A	⌜CAL	E
4B	⌞ERRS	8
4C	JUMP	F
4D	⟨4⟩	4
4E	⟨C⟩	C

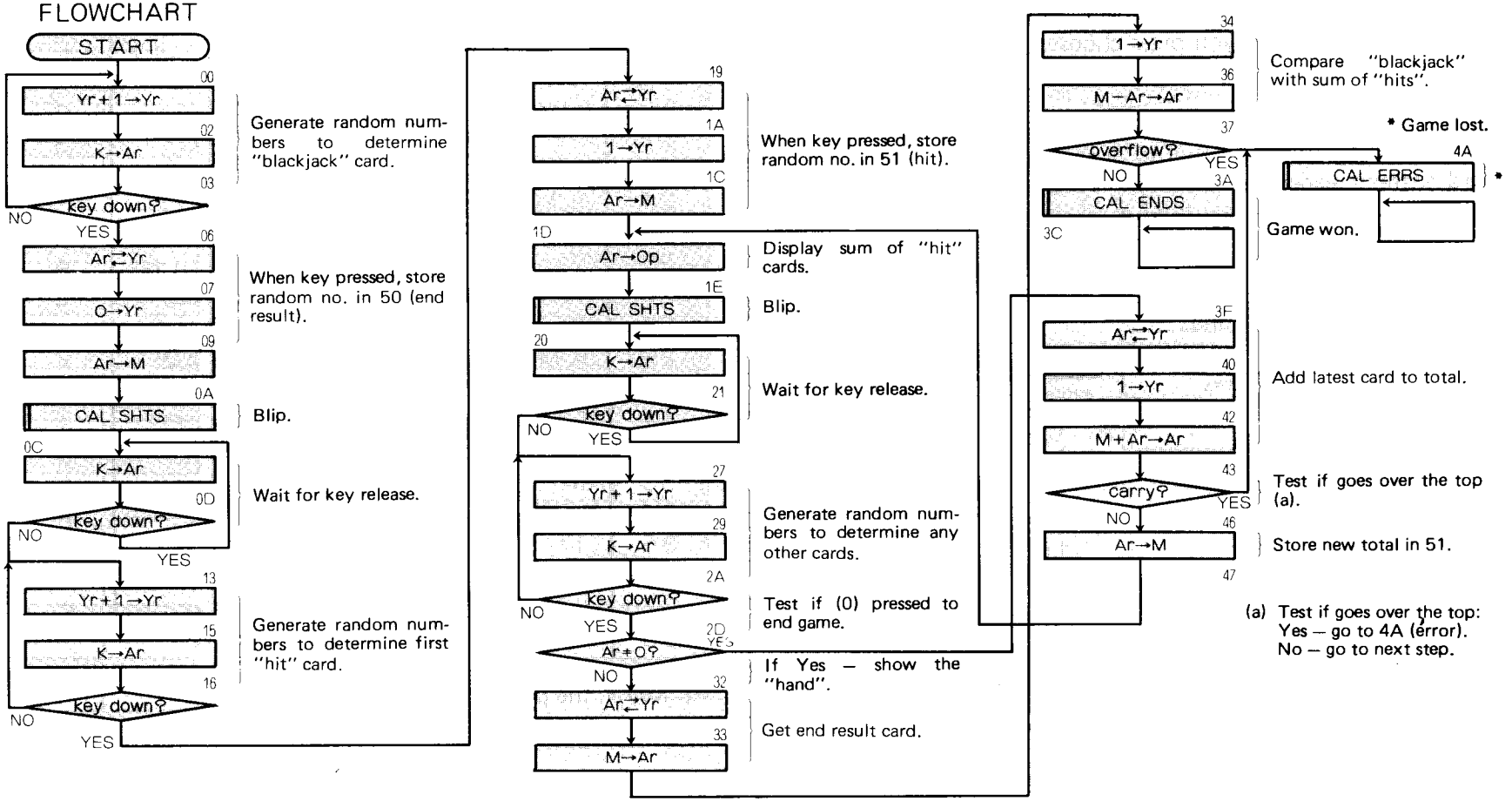
A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

C) The first key pressed "draws" the "blackjack" card (no display). The second key pressed draws your first card which displays on the HEX. LED. The value of the end result card is stored in 50; the sum of the "hit" cards is stored in 51.

D) Press RESET, 1, RUN to play again.

FLOWCHART


START
00
Yr+1→Yr
02
K→Ar
03
key down?
NO
YES
06
Ar⇄Yr
07
0→Yr
09
Ar→M
0A
CAL SHTS
0C
K→Ar
0D
key down?
NO
YES
13
Yr+1→Yr
15
K→Ar
16
key down?
NO
YES
Generate random numbers to determine "blackjack" card.
When key pressed, store random no. in 50 (end result).
Blip.
Wait for key release.
Generate random numbers to determine first "hit" card.
19
Ar⇄Yr
1A
1→Yr
1C
Ar→M
1D
Ar→Op
1E
CAL SHTS
20
K→Ar
21
key down?
NO
YES
27
Yr+1→Yr
29
K→Ar
2A
key down?
NO
YES
2D
Ar≠0?
YES
NO
32
Ar⇄Yr
33
M→Ar
When key pressed, store random no. in 51 (hit).
Display sum of "hit" cards.
Blip.
Wait for key release.
Generate random numbers to determine any other cards.
Test if (0) pressed to end game.
If Yes — show the "hand".
Get end result card.
34
1→Yr
36
M−Ar→Ar
37
overflow?
YES
NO
3A
CAL ENDS
3C
3F
Ar⇄Yr
40
1→Yr
42
M+Ar→Ar
43
carry?
YES
NO
46
Ar→M
47
4A
CAL ERRS
Compare "blackjack" with sum of "hits".
* Game lost.
Game won.
Add latest card to total.
Test if goes over the top (a).
Store new total in 51.
(a) Test if goes over the top:
Yes — go to 4A (error).
No — go to next step.

No.89 "Make-a-Match" Game

In this game, when a key is pressed the contents of the corresponding address is displayed. You then press another key and if the second memory has the same contents you win. Otherwise you lose. It's a game to test your patience!

PROGRAM

address	command	machine code
00	KA	0
01	JUMP	F
02	⟨0⟩	0
03	⟨0⟩	0
04	⌜CAL	E
05	⌞SHTS	9
06	CY	3
07	MA	5
08	AO	1
09	CY	3
0A	KA	0
0B	JUMP	F
0C	⟨1⟩	1
0D	⟨1⟩	1
0E	JUMP	F
0F	⟨0⟩	0
10	⟨A⟩	A
11	KA	0
12	JUMP	F
13	⟨1⟩	1
14	⟨1⟩	1
15	CY	3
16	M−	7
17	C I A	C
18	⟨0⟩	0

address	command	machine code
19	JUMP	F
1A	⟨2⟩	2
1B	⟨7⟩	7
1C	⌜CAL	E
1D	⌞ENDS	7
1E	MA	5
1F	AO	1
20	KA	0
21	JUMP	F
22	⟨0⟩	0
23	⟨0⟩	0
24	JUMP	F
25	⟨2⟩	2
26	⟨0⟩	0
27	⌜CAL	E
28	⌞ERRS	8
29	JUMP	F
2A	⟨1⟩	1
2B	⟨E⟩	E

A) Key in the program and check it.

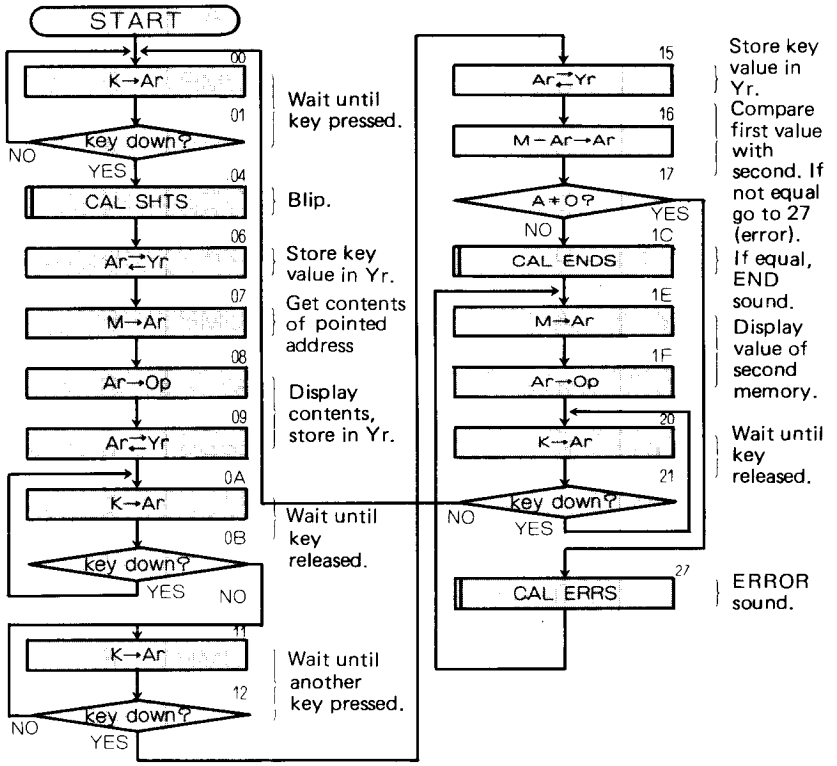
B) Load some numbers into 50-5F (some of them must be the same!)

Example: 50⟨3⟩ 51⟨1⟩ 52⟨3⟩ 53⟨2⟩ −−−−−−−

C) Press RESET, 1, RUN to start. Then each time one of the keys 0-F is pressed, the memory contents will be displayed.

D) First key—memory value is displayed.
Second key (different than first)—memory value is compared with first:
if equal, you win (END sound).
if not equal, you lose (ERROR sound).

FLOWCHART



No.90 Guess a Random Number Game

The first key pressed in this game selects a random number. You then try to guess which number to add to the random number to give the answer 0. Keep pressing keys until you come to the right (or wrong) number. The game ends when you succeed. But there is a trap — if the sum of the two numbers is 8 you hear the ERROR sound. The number of attempts made is displayed on the HEX. LED.

A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

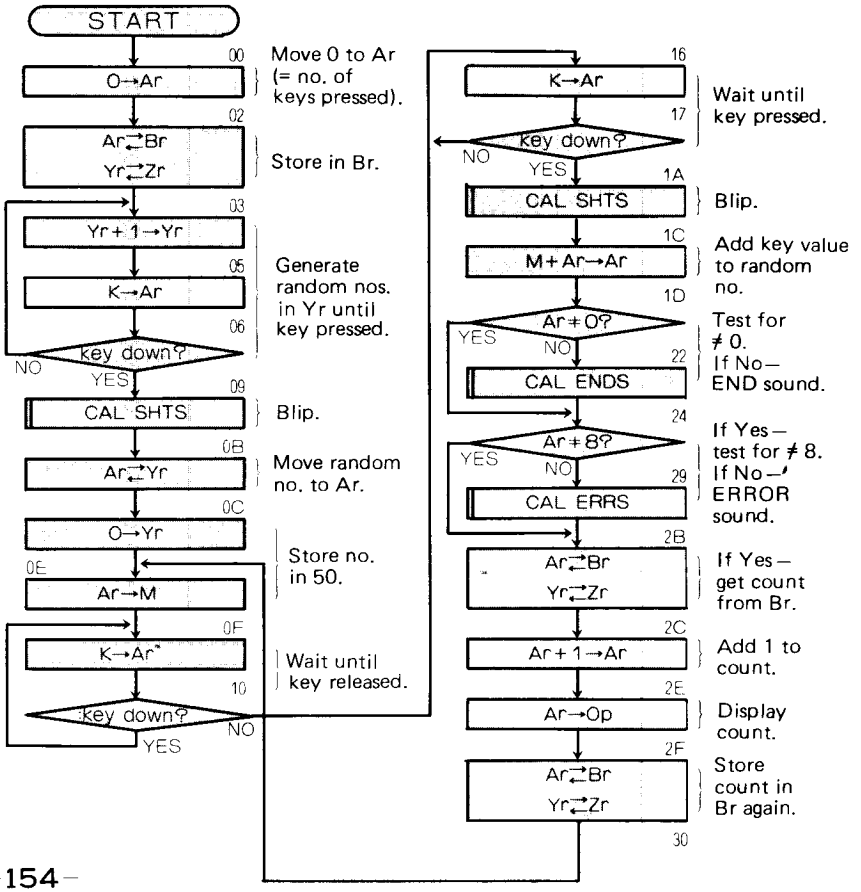
C) The game begins when any number key is pressed.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	AIY	B
04	⟨1⟩	1
05	KA	0
06	JUMP	F
07	⟨0⟩	0
08	⟨3⟩	3
09	⌈CAL	E
0A	⌊SHTS	9
0B	CY	3
0C	TIY	A
0D	⟨0⟩	0
0E	AM	4
0F	KA	0
10	JUMP	F
11	⟨1⟩	1
12	⟨6⟩	6
13	JUMP	F
14	⟨0⟩	0
15	⟨F⟩	F
16	KA	0
17	JUMP	F
18	⟨1⟩	1
19	⟨6⟩	6

address	command	machine code
1A	⌈CAL	E
1B	⌊SHTS	9
1C	M+	6
1D	CIA	C
1E	⟨0⟩	0
1F	JUMP	F
20	⟨2⟩	2
21	⟨4⟩	4
22	⌈CAL	E
23	⌊ENDS	7
24	CIA	C
25	⟨8⟩	8
26	JUMP	F
27	⟨2⟩	2
28	⟨B⟩	B
29	⌈CAL	E
2A	⌊ERRS	8
2B	CH	2
2C	AIA	9
2D	⟨1⟩	1
2E	AO	1
2F	CH	2
30	JUMP	F
31	⟨0⟩	0
32	⟨E⟩	E

FLOWCHART



No.91 Guess the Number in 50 Game

In this game you try to guess the number stored at address 50. If you are correct, you hear the END sound. If your guess is within two numbers of the correct answer, you will hear two beeps. If it is a wrong guess, you will hear just a blip. The HEX. LED shows the number of attempts made.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨1⟩	1
02	TIA	8
03	⟨0⟩	0
04	AM	4
05	TIY	A
06	⟨0⟩	0
07	KA	0
08	JUMP	F
09	⟨0⟩	0
0A	⟨5⟩	5
0B	M−	7
0C	CIA	C
0D	⟨0⟩	0
0E	JUMP	F
0F	⟨1⟩	1
10	⟨6⟩	6
11	⌈CAL	E
12	⌊ENDS	7
13	JUMP	F
14	⟨1⟩	1
15	⟨3⟩	3
16	⌈CAL	E
17	⌊SHTS	9
18	AIA	9
19	⟨D⟩	D
1A	JUMP	F

address	command	machine code
1B	⟨1⟩	1
1C	⟨F⟩	F
1D	⌈CAL	E
1E	⌊LONS	A
1F	AIA	9
20	⟨3⟩	3
21	AIA	9
22	⟨2⟩	2
23	⌈CAL	E
24	⌊LONS	A
25	TIY	A
26	⟨1⟩	1
27	MA	5
28	AIA	9
29	⟨1⟩	1
2A	AM	4
2B	AO	1
2C	KA	0
2D	JUMP	F
2E	⟨0⟩	0
2F	⟨5⟩	5
30	JUMP	F
31	⟨2⟩	2
32	⟨C⟩	C

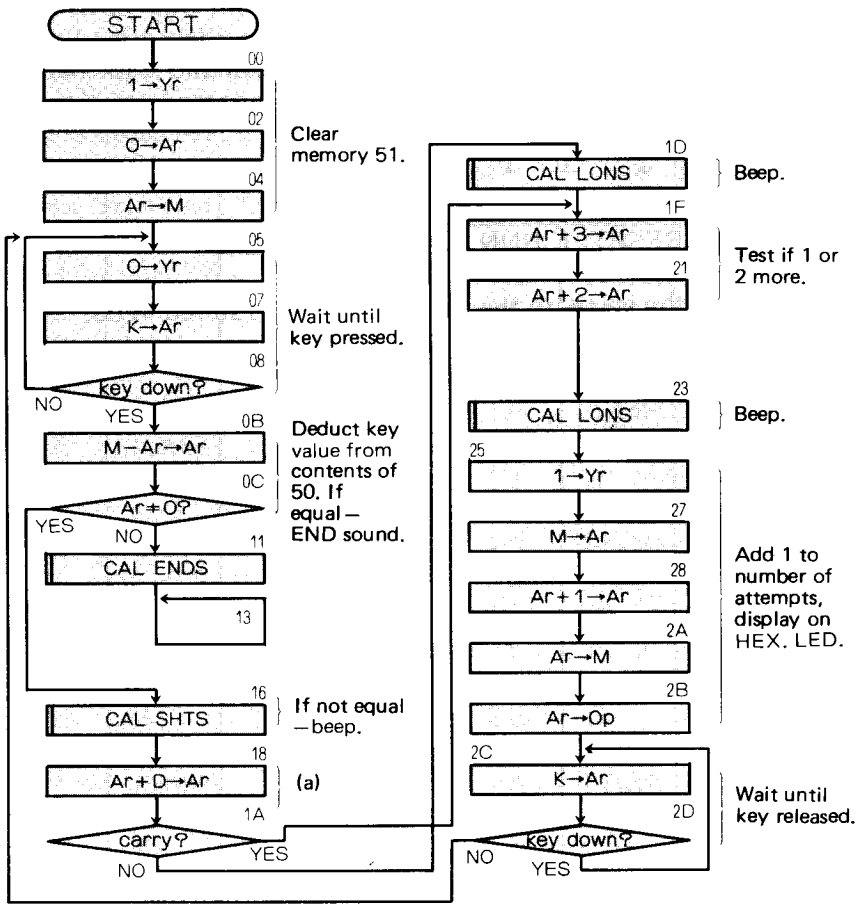
A) Key in the program and check it.

B) Load a number into address 50, for example 5 (or better still get someone else to do it for you):

C) Press RESET, 1, RUN to start the program.

D) Press the key with the value you think is stored in 50.

FLOWCHART



(a) Test if 1 or 2 less.

No.92 Sharpshooter Game

The target light starts at binary LED 6 and moves towards LED 0. You have to press the key corresponding to the LED that is turned on at that particular moment. If you hit it in time, you win. If not, the target continues moving and when it gets to the end (0), you hear the ERROR sound.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨6⟩	6
02	TIY	A
03	⟨0⟩	0
04	AM	4
05	MA	5
06	CY	3
07	⌜CAL	E
08	⌞SETR	1
09	KA	0
0A	KA	0
0B	KA	0
0C	KA	0
0D	KA	0
0E	KA	0
0F	JUMP	F
10	⟨2⟩	2
11	⟨3⟩	3
12	TIY	A
13	⟨0⟩	0
14	M−	7
15	CIA	C
16	⟨0⟩	0
17	JUMP	F

②

address	command	machine code
18	⟨2⟩	2
19	⟨A⟩	A
1A	⌜CAL	E
1B	⌞ENDS	7
1C	MA	5
1D	CY	3
1E	⌜CAL	E
1F	⌞SETR	1
20	JUMP	F
21	⟨2⟩	2
22	⟨0⟩	0
23	AIY	B
24	⟨F⟩	F
25	JUMP	F
26	⟨0⟩	0
27	⟨9⟩	9
28	TIY	A
29	⟨0⟩	0
2A	MA	5
2B	CY	3
2C	⌜CAL	E
2D	⌞RSTR	2
2E	KA	0
2F	JUMP	F

③

address	command	machine code
30	⟨3⟩	3
31	⟨5⟩	5
32	JUMP	F
33	⟨2⟩	2
34	⟨E⟩	E
35	CY	3
36	AIA	9
37	⟨F⟩	F

④

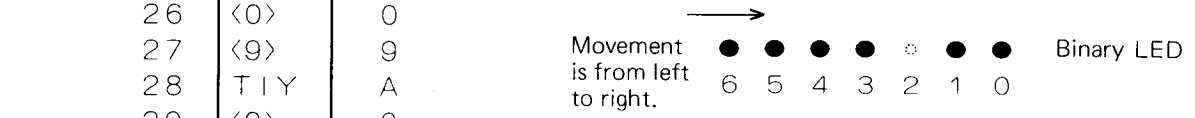
address	command	machine code
38	JUMP	F
39	⟨0⟩	0
3A	⟨2⟩	2
3B	⌜CAL	E
3C	⌞ERRS	8
3D	JUMP	F
3E	⟨3⟩	3
3F	⟨D⟩	D

A) Key in the program and check it.

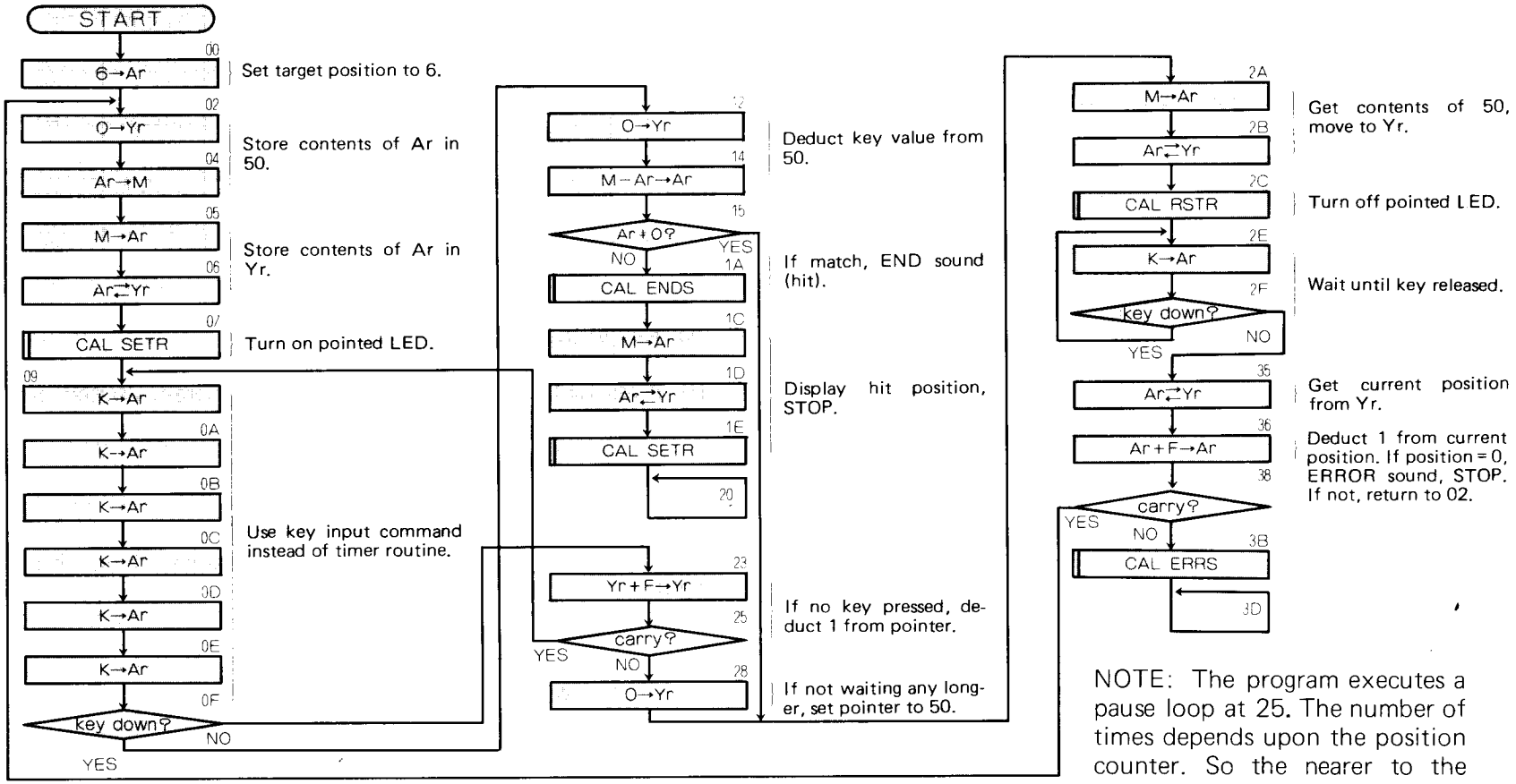
B) Press RESET, 1, RUN to start the program. Watch out! The target appears very quickly.

C) Press the number of the lighted LED – fast!

This game may be played with more than one person. Score points according to where you hit the target. If you hit it right at the start you score 6 points, so if you are fast you score more.



FLOWCHART



NOTE: The program executes a pause loop at 25. The number of times depends upon the position counter. So the nearer to the right that the target gets, the smaller the count and the faster the movement.

No.93 Speed Counting in hex

The object of this program is to add two hex numbers together to equal 0. One number is displayed on the HEX. LED for only a short time. You must quickly figure out what number added to the displayed number will equal 0 (or (1)0000 = 16— the carry is dropped). Then press that key while the original number is still displayed. If you press and hold down the correct key in time, you score one point. When all 16 memory contents have been displayed you will hear the END sound and your score is displayed.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	TIY	A
04	⟨F⟩	F
05	KA	0
06	JUMP	F
07	⟨0⟩	0
08	⟨C⟩	C
09	JUMP	F
0A	⟨3⟩	3
0B	⟨E⟩	E
0C	MA	5
0D	AO	1
0E	TIA	8
0F	⟨F⟩	F
10	[CAL	E
11	[TIMR	C
12	KA	0
13	JUMP	F
14	⟨2⟩	2
15	⟨E⟩	E
16	M+	6
17	CIA	C

②

address	command	machine code
18	⟨0⟩	0
19	JUMP	F
1A	⟨3⟩	3
1B	⟨9⟩	9
1C	CH	2
1D	AIA	9
1E	⟨1⟩	1
1F	CH	2
20	[CAL	E
21	[LONS	A
22	AIY	B
23	⟨F⟩	F
24	JUMP	F
25	⟨0⟩	0
26	⟨5⟩	5
27	CH	2
28	AO	1
29	[CAL	E
2A	[ENDS	7
2B	JUMP	F
2C	⟨2⟩	2
2D	⟨B⟩	B
2E	TIA	8
2F	⟨0⟩	0

③

address	command	machine code
30	[CAL	E
31	[TIMR	C
32	KA	0
33	JUMP	F
34	⟨3⟩	3
35	⟨9⟩	9
36	JUMP	F
37	⟨1⟩	1
38	⟨6⟩	6
39	[CAL	E
3A	[SHTS	9

④

address	command	machine code
3B	JUMP	F
3C	⟨2⟩	2
3D	⟨2⟩	2
3E	[CAL	E
3F	[RSTO	0
40	JUMP	F
41	⟨0⟩	0
42	⟨5⟩	5

A) Key in the program and check it.

B) Load various numbers into 50-5F.

Example:

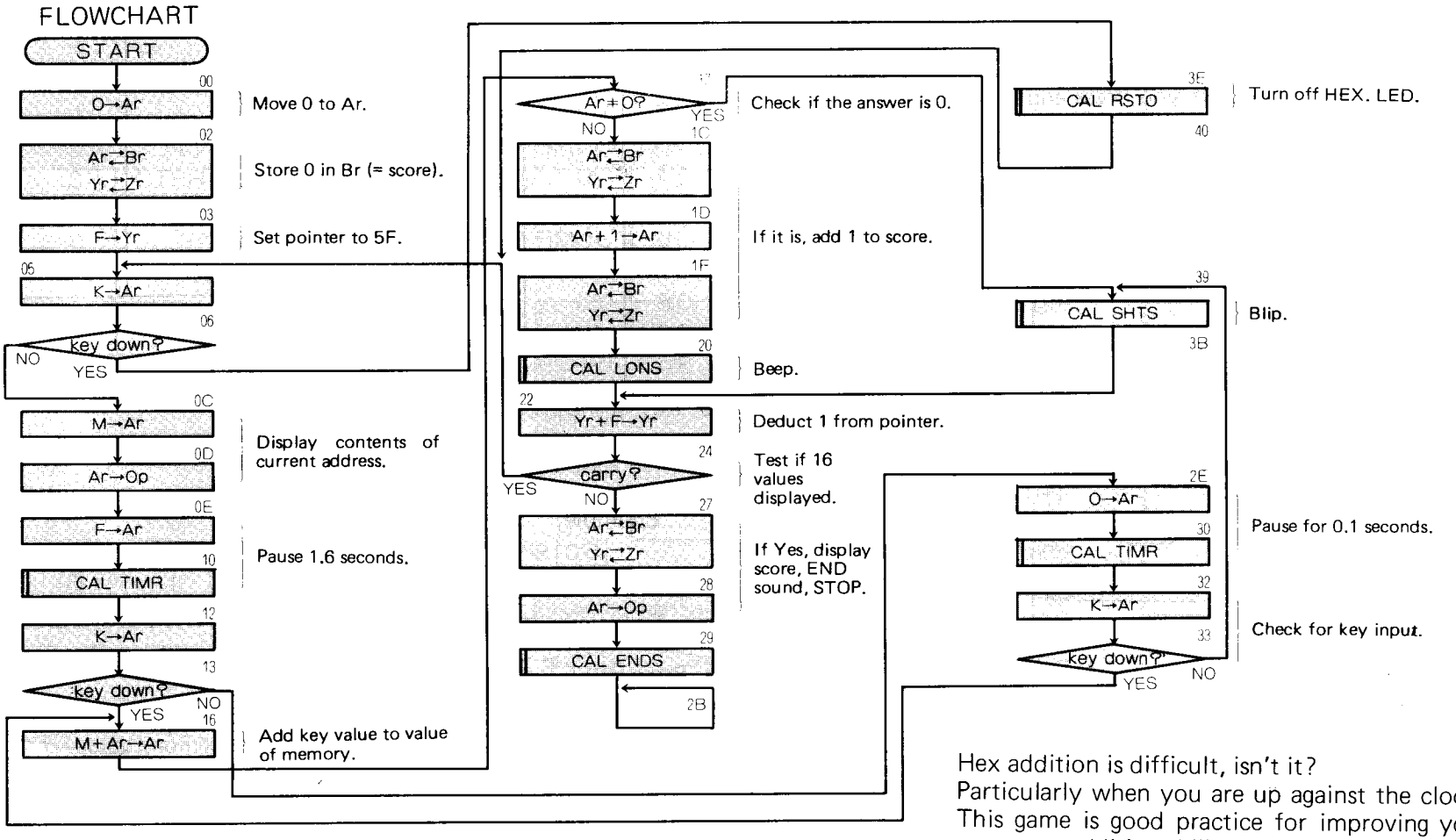
50	51	52	53	54	55	.....	5B	5C	5D	5E	5F
9	C	4	A	F	3		E	1	8	B	6

C) Press RESET, 1, RUN to start the program.

D) If a number key is pressed while a memory value is being displayed, the key value is added to the memory value. If the answer is binary (1)0000, i.e. there was a carry which left 0 in the register, you score one point.

NOTE: For your key value to be recognized, you must hold the key down until the HEX. LED is turned off. There is a blip sound each time you get the answer wrong, a beep if correct.

FLOWCHART



Hex addition is difficult, isn't it?
Particularly when you are up against the clock!
This game is good practice for improving your computer addition skills.

No.94 Gunfight Game

This game is for two players. When you press (RUN) the middle binary LED (R3) will light up after a short pause. The two players use (4) and (7) as "triggers". The player who shoots first is the winner.

Display if (4) wins: ○ ● ● ● ● ● ●
Display if (7) wins: ● ● ● ● ● ● ○

If you press a key before the middle LED comes on, you are "trigger happy" and you will hear the ERROR sound.

If (4) fires too soon: ● ○ ● ● ● ● ●
If (7) fires too soon: ● ● ● ● ● ○ ●

Note: Due to the nature of the KA command, (7) has a slight advantage.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨F⟩	F
02	[CAL	E
03	[TIMR	C
04	CH	2
05	KA	0
06	JUMP	F
07	⟨0⟩	0
08	⟨C⟩	C
09	JUMP	F
0A	⟨3⟩	3
0B	⟨6⟩	6
0C	TIA	8

②

address	command	machine code
0D	⟨3⟩	3
0E	[CAL	E
0F	[TIMR	C
10	CH	2
11	AIA	9
12	⟨F⟩	F
13	JUMP	F
14	⟨0⟩	0
15	⟨4⟩	4
16	TIY	À
17	⟨3⟩	3
18	[CAL	E
19	[SETR	1

③

address	command	machine code
1A	TIY	A
1B	⟨6⟩	6
1C	KA	0
1D	JUMP	F
1E	⟨1⟩	1
1F	⟨C⟩	C
20	CIA	C
21	⟨4⟩	4
22	JUMP	F
23	⟨2⟩	2
24	⟨8⟩	8
25	JUMP	F
26	⟨2⟩	2
27	⟨F⟩	F
28	CIA	C
29	⟨7⟩	7
2A	JUMP	F
2B	⟨1⟩	1
2C	⟨C⟩	C
2D	TIY	A
2E	⟨0⟩	0
2F	[CAL	E
30	[SETR	1
31	[CAL	E
32	[ENDS	7
33	JUMP	F

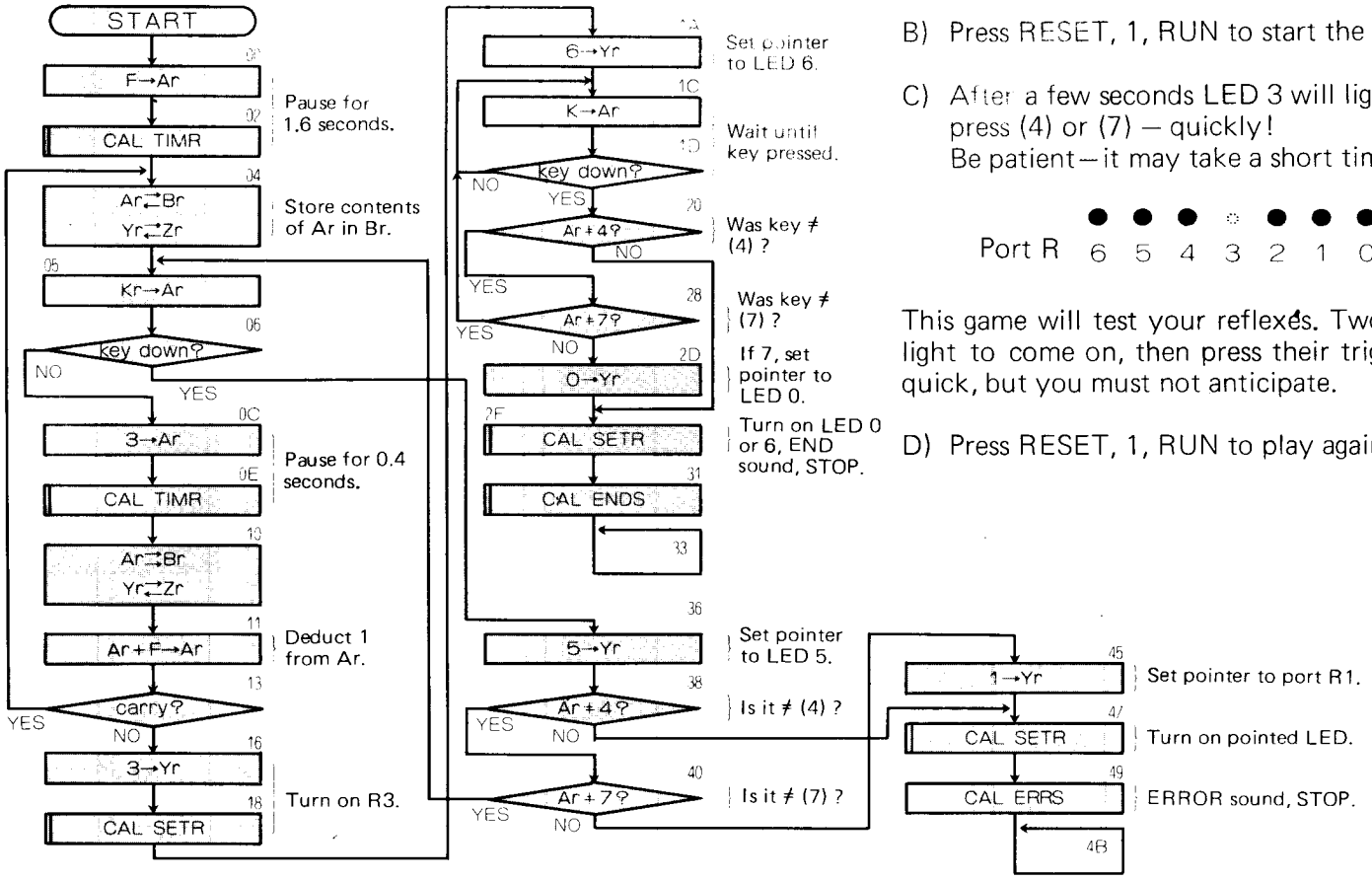
④

address	command	machine code
34	⟨3⟩	3
35	⟨3⟩	3
36	TIY	A
37	⟨5⟩	5
38	CIA	C
39	⟨4⟩	4
3A	JUMP	F
3B	⟨4⟩	4
3C	⟨0⟩	0
3D	JUMP	F
3E	⟨4⟩	4
3F	⟨7⟩	7
40	CIA	C
41	⟨7⟩	7
42	JUMP	F
43	⟨0⟩	0
44	⟨5⟩	5
45	TIY	A
46	⟨1⟩	1
47	[CAL	E
48	[SETR	1
49	[CAL	E
4A	[ERRS	8
4B	JUMP	F
4C	⟨4⟩	4
4D	⟨B⟩	B

NOTE: "port R1" is another way of referring to binary LED 1.

● ● ● ● ● ● ●←PORT R
R6 R5 R4 R3 R2 R1 R0

FLOWCHART



A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

C) After a few seconds LED 3 will light up and you have to press (4) or (7) – quickly!
Be patient – it may take a short time for R3 to come on.

This game will test your reflexes. Two players wait for the light to come on, then press their triggers. You have to be quick, but you must not anticipate.

D) Press RESET, 1, RUN to play again.

No.95 "Slot Machine" Game

This program provides a simple slot machine game. The display is divided into 3 parts, as shown in the diagram.

(a) (b) (c)

At (a) only 3 binary digits are displayed; if the number displayed there is larger than 7 you have to guess what it is. If the three numbers in the three sections match perfectly you will hear two END sounds. If (a) matches (b), or (b) matches (c), you will hear one END sound.
When the game starts the display changes at high speed. First press (1) to freeze (a), then press (2) to freeze (b), then press (3) to freeze (c).

PROGRAM ①

address	command	machine code
00	TIY	A
01	⟨D⟩	D
02	MA	5
03	AIA	9
04	⟨1⟩	1
05	AM	4
06	AIY	B
07	⟨1⟩	1
08	AO	1
09	⌜CAL	E
0A	⌞DSPR	D
0B	CIY	D
0C	⟨0⟩	0
0D	JUMP	F

②

address	command	machine code
0E	⟨0⟩	0
0F	⟨2⟩	2
10	CY	3
11	KA	0
12	CIA	C
13	⟨1⟩	1
14	JUMP	F
15	⟨0⟩	0
16	⟨0⟩	0
17	TIY	A
18	⟨E⟩	E
19	MA	5
1A	AIA	9
1B	⟨1⟩	1

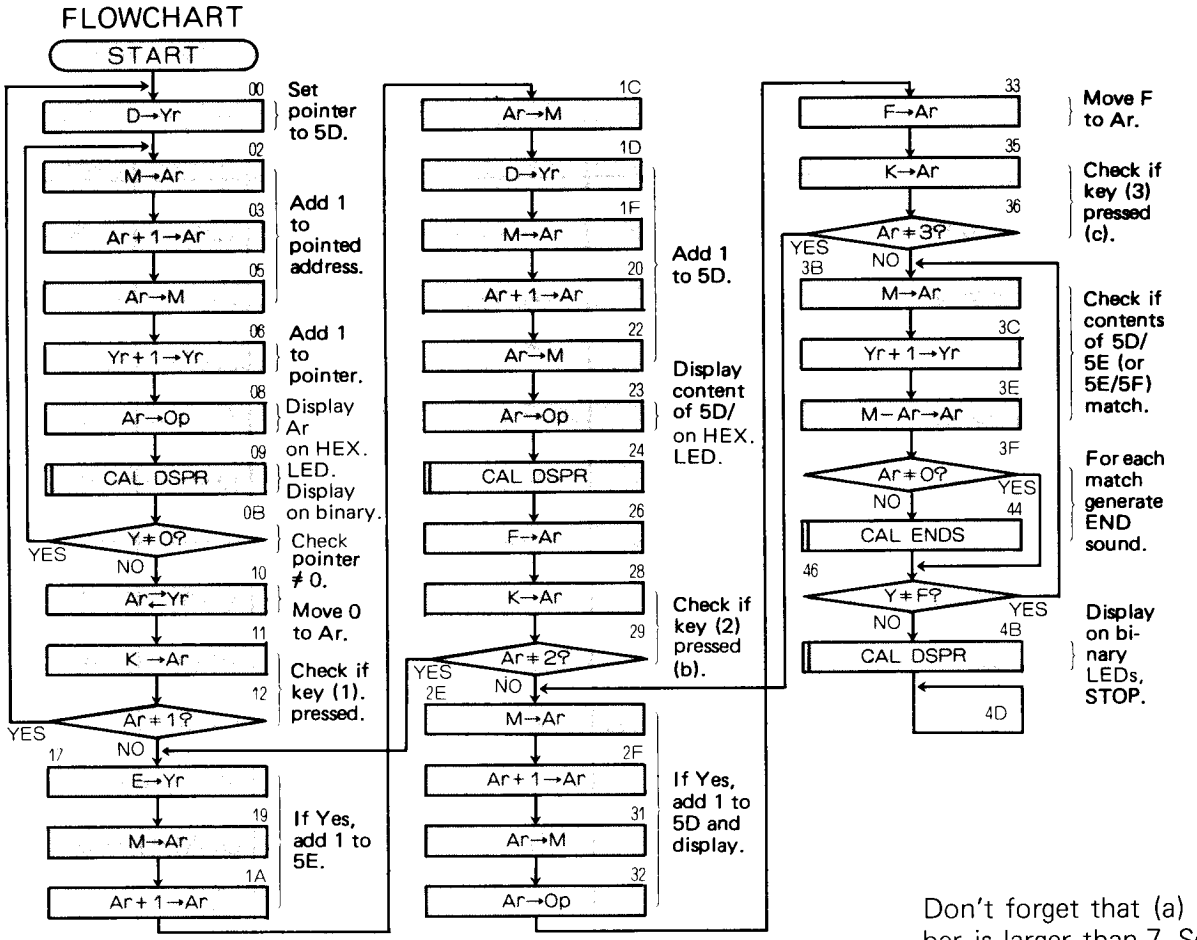
③

address	command	machine code
1C	AM	4
1D	TIY	A
1E	⟨D⟩	D
1F	MA	5
20	AIA	9
21	⟨1⟩	1
22	AM	4
23	AO	1
24	⌜CAL	E
25	⌞DSPR	D
26	TIA	8
27	⟨F⟩	F
28	KA	0
29	CIA	C
2A	⟨2⟩	2
2B	JUMP	F
2C	⟨1⟩	1
2D	⟨7⟩	7
2E	MA	5
2F	AIA	9
30	⟨1⟩	1
31	AM	4
32	AO	1
33	TIA	8
34	⟨F⟩	F
35	KA	0

④

address	command	machine code
36	CIA	C
37	⟨3⟩	3
38	JUMP	F
39	⟨2⟩	2
3A	⟨E⟩	E
3B	MA	5
3C	AIY	B
3D	⟨1⟩	1
3E	M−	7
3F	CIA	C
40	⟨0⟩	0
41	JUMP	F
42	⟨4⟩	4
43	⟨6⟩	6
44	⌜CAL	E
45	⌞ENDS	7
46	CIY	D
47	⟨F⟩	F
48	JUMP	F
49	⟨3⟩	3
4A	⟨B⟩	B
4B	⌜CAL	E
4C	⌞DSPR	D
4D	JUMP	F
4E	⟨4⟩	4
4F	⟨D⟩	D

FLOWCHART

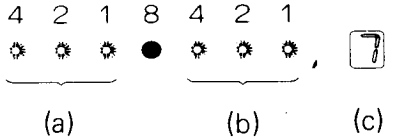


A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.
Press 1, 2, 3 to freeze the a, b, c positions.

C) If you are not lucky at first, try again by pressing RESET, 1, RUN.

This "slot machine" behaves very much like those that you see at amusement arcades. But you can only play for fun—the Microcomputer Trainer has not been programmed to give cash prizes.



If (a), (b) and (c) all match you will hear the END sound twice.

Don't forget that (a) may have an imaginary digit if the number is larger than 7. So if you think you have a match between (a) and (b) but the END sound is not generated, it is because the number in (a) is larger than 7.

No.96 Memory Tester

Before running the program, load numbers into addresses 50-5E and try remember what is there. When the program starts, enter the key value that you think matches the value in each address, starting at 50. If you make a mistake, you have lost; if you get them all correct you will hear the END sound. In both cases your score will be displayed.

PROGRAM

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	CH	2
03	TIY	A
04	⟨0⟩	0
05	KA	0
06	JUMP	F
07	⟨0⟩	0
08	⟨5⟩	5
09	KA	0
0A	JUMP	F
0B	⟨1⟩	1
0C	⟨0⟩	0
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨9⟩	9
10	⌜CAL	E
11	⌞CMPL	4
12	M+	6
13	CIA	C
14	⟨F⟩	F
15	JUMP	F
16	⟨2⟩	2
17	⟨E⟩	E
18	MA	5
19	AO	1

address	command	machine code
1A	⌜CAL	E
1B	⌞SHTS	9
1C	CH	2
1D	AIA	9
1E	⟨1⟩	1
1F	CH	2
20	AIY	B
21	⟨1⟩	1
22	CIY	D
23	⟨F⟩	F
24	JUMP	F
25	⟨0⟩	0
26	⟨5⟩	5
27	⌜CAL	E
28	⌞ENDS	7
29	CH	2
2A	AO	1
2B	JUMP	F
2C	⟨2⟩	2
2D	⟨B⟩	B
2E	⌜CAL	E
2F	⌞ERRS	8
30	JUMP	F
31	⟨2⟩	2
32	⟨9⟩	9

A) Key in the program and check it.

B) Load some numbers into 50-5E.

Example:

50	51	52	53	54	55	56	57	
3	4	5	6	7	8	9	A	

C) Press RESET, 1, RUN to start the program.

D) Press a number key for each address. If you choose the correct number, the value will be displayed and you score 1. Continue pressing keys until you make a mistake or hear the END sound.

Example:

Top score = F

The score is displayed at the end, or when you make a mistake.

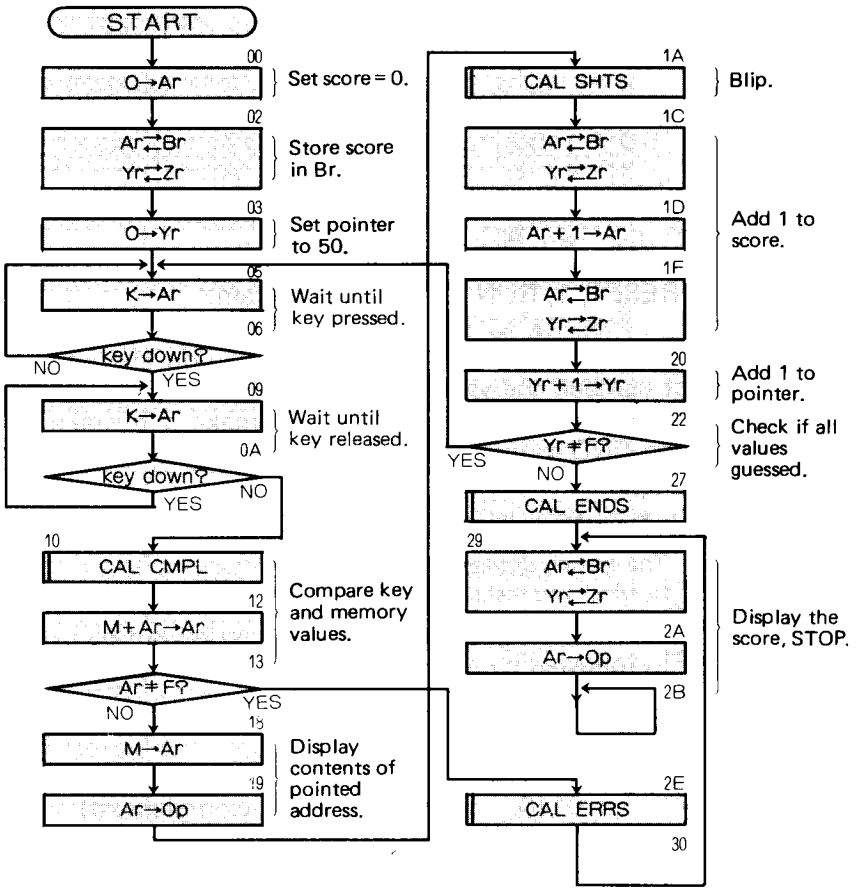
Notice that in this program, deduction is carried out by using CAL CMPL, followed by an addition,

Example:

	Method 1	Method 2
6	0110	0110 → 1001 (complement)
−6	−0110	+0110
	= 0000	= 1111

In method 2, the values are the same if the answer is F.

FLOWCHART



Notes:

No.97 Store Random Numbers (0-9) in memory

This program stores a random number (0-9) in each of the addresses 50-5F.

PROGRAM

address	command	machine code
00	TIY	A
01,	⟨0⟩	0
02	TIA	8
03	⟨F⟩	F
04	AIA	9
05	⟨1⟩	1
06	CH	2
07	KA	0
08	JUMP	F
09	⟨1⟩	1
0A	⟨5⟩	5
0B	CH	2
0C	AO	1
0D	CH	2
0E	KA	0
0F	JUMP	F
10	⟨1⟩	1
11	⟨E⟩	E
12	JUMP	F
13	⟨0⟩	0
14	⟨E⟩	E
15	CH	2
16	CIA	C
17	⟨9⟩	9
18	JUMP	F

address	command	machine code
19	⟨0⟩	0
1A	⟨4⟩	4
1B	JUMP	F
1C	⟨0⟩	0
1D	⟨2⟩	2
1E	CH	2
1F	AM	4
20	AIY	B
21	⟨1⟩	1
22	CIY	D
23	⟨0⟩	0
24	JUMP	F
25	⟨1⟩	1
26	⟨6⟩	6
27	⌈CAL	E
28	⌊ENDS	7
29	JUMP	F
2A	⟨2⟩	2
2B	⟨9⟩	9

A) Key in the program and check it.

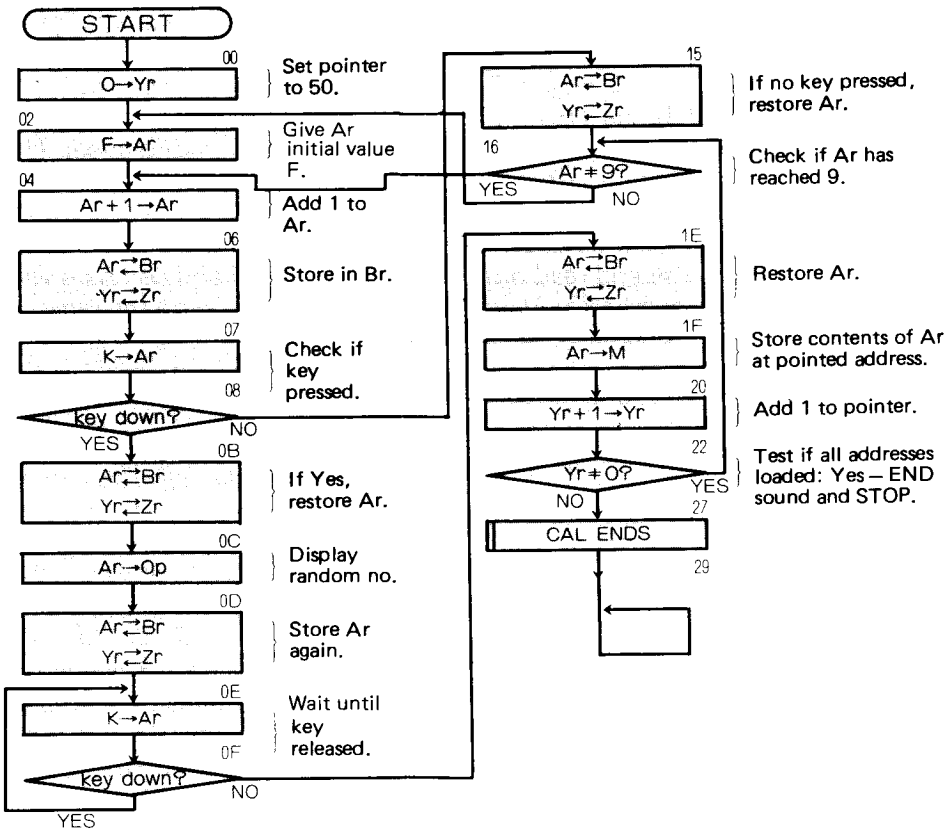
B) Press RESET, 1, RUN to start the program.

C) Watch the HEX. LED while you press any number keys. Each time you press a key a number is displayed. This number is stored at the pointed address. Write these numbers down in sequence.

D) At the end read out the addresses 50-5F and compare with the numbers you wrote down. They should be the same.

The command (F → Ar) sets the initial value for generating random numbers. When 1 is added, Ar contains 0. The range of numbers is limited to 9 at address 16, where a test sends the program back to address 02 if 9 has been reached. In program 99 you will find that the range is extended to 1-E when you want to generate random musical sounds.

FLOWCHART


START
00
0→Yr
Set pointer to 50.
02
F→Ar
Give Ar initial value F.
04
Ar+1→Ar
Add 1 to Ar.
06
Ar⇄Br
Yr⇄Zr
Store in Br.
07
K→Ar
Check if key pressed.
08
key down?
NO
YES
0B
Ar⇄Br
Yr⇄Zr
If Yes, restore Ar.
0C
Ar→Op
Display random no.
0D
Ar⇄Br
Yr⇄Zr
Store Ar again.
0E
K→Ar
Wait until key released.
0F
key down?
NO
YES
15
Ar⇄Br
Yr⇄Zr
If no key pressed, restore Ar.
16
Ar≠9?
YES
NO
Check if Ar has reached 9.
1E
Ar⇄Br
Yr⇄Zr
Restore Ar.
1F
Ar→M
Store contents of Ar at pointed address.
20
Yr+1→Yr
Add 1 to pointer.
22
Yr≠0?
NO
YES
Test if all addresses loaded: Yes – END sound and STOP.
27
CAL ENDS
29

No.98 Guessing Musical Notes

Find out just how musical you are.

In this game you will store notes, (1-E), in 50-5E. When they are played back, you guess the note in turn and press the corresponding key. If you get them all correct you will hear the END sound. When you make a mistake you will hear the ERROR sound and the program will stop. In both cases your score will be displayed. Start over until you can guess them all.

Remembering the sequence of the notes will help you get a little farther each time you play. When you have guessed all the notes you stored, enter them in a different sequence and play again.

The characters 0 and F do not generate notes.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨5⟩	5
02	[CAL	E
03	[CHNG	5
04	TIA	8
05	⟨0⟩	0
06	CH	2
07	TIA	A
08	⟨0⟩	0
09	MA	5
0A	[CAL	E
0B	[SUND	B
0C	KA	0
0D	JUMP	F
0E	⟨0⟩	0
0F	⟨C⟩	C

②

address	command	machine code
10	KA	0
11	JUMP	F
12	⟨1⟩	1
13	⟨7⟩	7
14	JUMP	F
15	⟨1⟩	1
16	⟨0⟩	0
17	[CAL	E
18	[SUND	B
19	[CAL	E
1A	[CMPL	4
1B	M−	6
1C	CIA	C
1D	⟨F⟩	F
1E	JUMP	F
1F	⟨3⟩	3

③

address	command	machine code
20	⟨9⟩	9
21	CH	2
22	AIA	9
23	⟨1⟩	1
24	CH	2
25	[CAL	E
26	[CHNG	5
27	[CAL	E
28	[TIMR	C
29	[CAL	E
2A	[CHNG	5
2B	AIY	B
2C	⟨1⟩	1
2D	CIY	D
2E	⟨F⟩	F
2F	JUMP	F

④

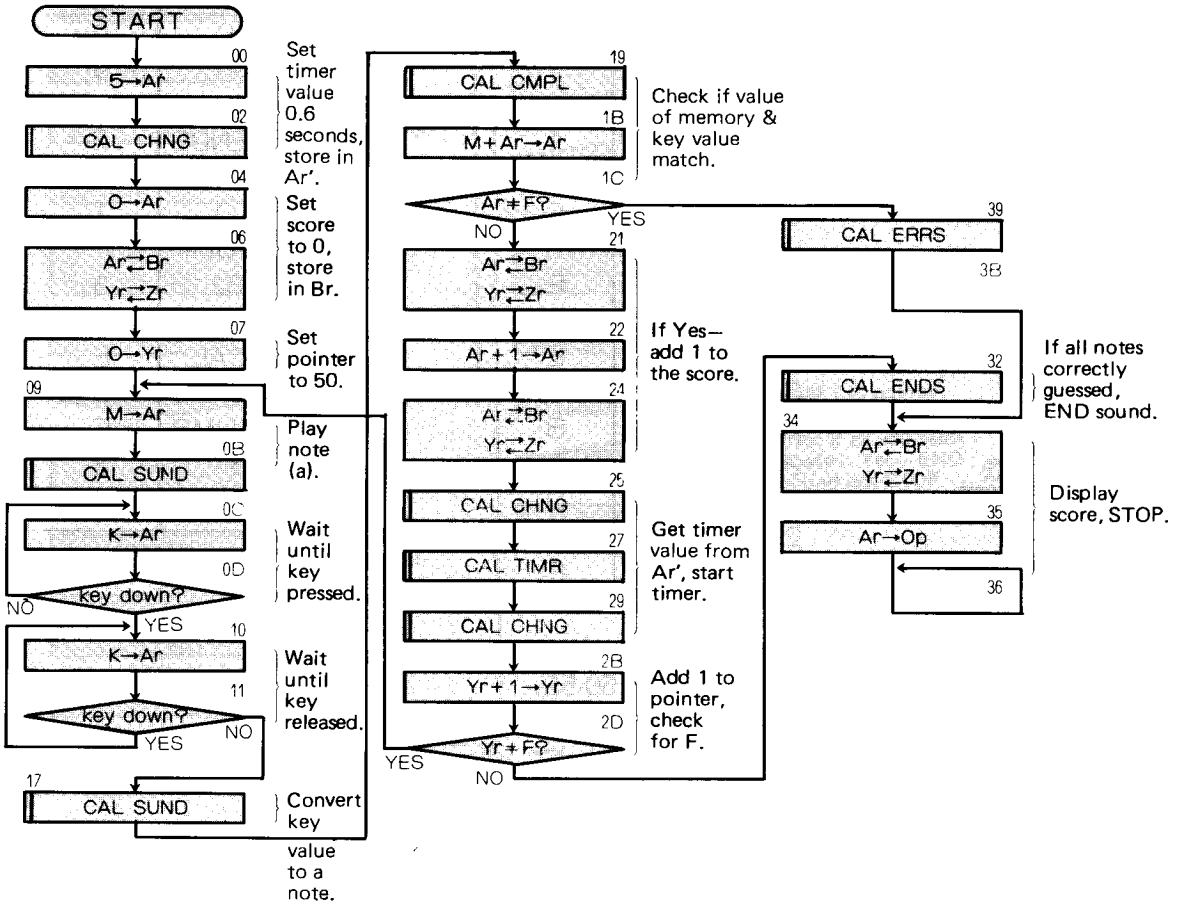
address	command	machine code
30	⟨0⟩	0
31	⟨9⟩	9
32	[CAL	E
33	[ENDS	7
34	CH	2
35	AO	1
36	JUMP	F
37	⟨3⟩	3
38	⟨6⟩	6
39	[CAL	E
3A	[ERRS	8
3B	JUMP	F
3C	⟨3⟩	3
3D	⟨4⟩	4

A) Load codes for various notes (1-E) in 50-5E.
Example:

B) Key in the program and check it.

C) Press RESET, 1, RUN to start. The first note will be played. Press the key that you think corresponds to the note played.

FLOWCHART



(a) Play note for code stored at pointed address.

No.99 Store Random Numbers for Musical Notes in Memory

With some of the games that you have programmed you know in advance what is stored in the memory. That gives you a rather unfair advantage in guessing. The purpose of this program is to show you how to store random numbers in memory (1-E) so that NOBODY knows in advance what is in there.

PROGRAM

address	command	machine code
00	TIY	A
01	⟨0⟩	0
02	TIA	8
03	⟨0⟩	0
04	AIA	9
05	⟨1⟩	1
06	CH	2
07	KA	0
08	JUMP	F
09	⟨1⟩	1
0A	⟨5⟩	5
0B	CH	2
0C	AO	1
0D	CH	2
0E	KA	0
0F	JUMP	F
10	⟨1⟩	1
11	⟨E⟩	E
12	JUMP	F
13	⟨0⟩	0
14	⟨E⟩	E
15	CH	2

address	command	machine code
16	CIA	C
17	⟨E⟩	E
18	JUMP	F
19	⟨0⟩	0
1A	⟨4⟩	4
1B	JUMP	F
1C	⟨0⟩	0
1D	⟨2⟩	2
1E	CH	2
1F	AM	4
20	AIY	B
21	⟨1⟩	1
22	CIY	D
23	⟨0⟩	0
24	JUMP	F
25	⟨1⟩	1
26	⟨6⟩	6
27	⌜CAL	E
28	⌞ENDS	7
29	JUMP	F
2A	⟨2⟩	2
2B	⟨9⟩	9

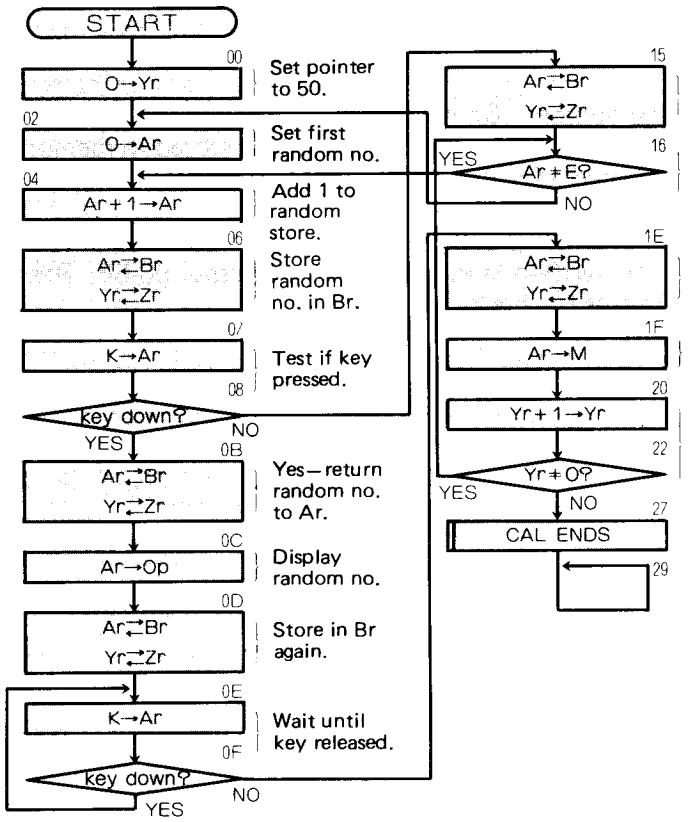
A) Key in the program and check it.

B) Press RESET, 1, RUN to start the program.

C) Press any number 16 times. Each time that you press a key a random number will be displayed and stored in memory at the pointed address.

D) When the END sound is heard, random numbers (1-E) have been written into all of the addresses 50-5E. Read the addresses to check.

FLOWCHART


START
00
0→Yr
Set pointer to 50.
02
0→Ar
Set first random no.
04
Ar+1→Ar
Add 1 to random store.
06
Ar⇄Br
Yr⇄Zr
Store random no. in Br.
07
K→Ar
Test if key pressed.
08
key down?
NO
YES
0B
Ar⇄Br
Yr⇄Zr
Yes—return random no. to Ar.
0C
Ar→Op
Display random no.
0D
Ar⇄Br
Yr⇄Zr
Store in Br again.
0E
K→Ar
Wait until key released.
0F
key down?
NO
YES
15
Ar⇄Br
Yr⇄Zr
Restore random no. to Ar.
16
YES
Ar≠E?
NO
Check if random no. has reached E. If Yes, return to set count to 0.
1E
Ar⇄Br
Yr⇄Zr
Restore random number to Ar.
1F
Ar→M
Move random number to pointed address.
20
Yr+1→Yr
Add 1 to pointer, test if it has gone past F, that is, is 0 again. If yes, END sound and STOP.
22
Yr≠0?
YES
NO
27
CAL ENDS
29

No.100 "Rat Bashing" Game

For this game any number in the range 0-6 is loaded into the addresses 50-5E. When the game starts the "rats" come out of their holes (binary LEDs) specified by the codes you have loaded, starting at 5E. When you see a "rat" (the lighted LED), press the corresponding key (0-6) to bash it. Each time a "rat" appears the pointer is reduced by 1, and when the "rat" at address 50 has emerged the game ends.

PROGRAM ①

address	command	machine code
00	TIA	8
01	⟨0⟩	0
02	⌜CAL	E
03	⌞CHNG	5
04	TIY	A
05	⟨E⟩	E
06	MA	5
07	CY	3
08	⌜CAL	E
09	⌞SETR	1
0A	CY	3
0B	CH	2
0C	TIY	A
0D	⟨F⟩	F
0E	KA	0
0F	JUMF	F
10	⟨3⟩	3
11	⟨9⟩	9
12	CH	2
13	KA	0
14	JUMP	F
15	⟨3⟩	3
16	⟨5⟩	5
17	M−	7

②

address	command	machine code
18	CIA	C
19	⟨0⟩	0
1A	JUMP	F
1B	⟨2⟩	2
1C	⟨6⟩	6
1D	⌜CAL	E
1E	⌞SHTS	9
1F	⌜CAL	E
20	⌞CHNG	5
21	AIA	9
22	⟨1⟩	1
23	AO	1
24	⌜CAL	E
25	⌞CHNG	5
26	MA	5
27	CY	3
28	⌜CAL	E
29	⌞RSTR	2
2A	CY	3
2B	AIY	B
2C	⟨F⟩	F
2D	JUMP	F
2E	⟨4⟩	4
2F	⟨6⟩	6

③

address	command	machine code
30	⌜CAL	E
31	⌞ENDS	7
32	JUMP	F
33	⟨3⟩	3
34	⟨2⟩	2
35	CH	2
36	JUMP	F
37	⟨0⟩	0
38	⟨E⟩	E
39	TIA	8
3A	⟨0⟩	0
3B	⌜CAL	E
3C	⌞TIMR	C
3D	AIY	B
3E	⟨F⟩	F

④

address	command	machine code
3F	JUMP	F
40	⟨0⟩	0
41	⟨E⟩	E
42	CH	2
43	JUMP	F
44	⟨2⟩	2
45	⟨6⟩	6
46	KA	0
47	JUMP	F
48	⟨0⟩	0
49	⟨6⟩	6
4B	JUMP	F
4B	⟨4⟩	4
4C	⟨6⟩	6

A) Key in the program and check it.

B) Load various numbers (0-6) at addresses 50-5E.

Example:

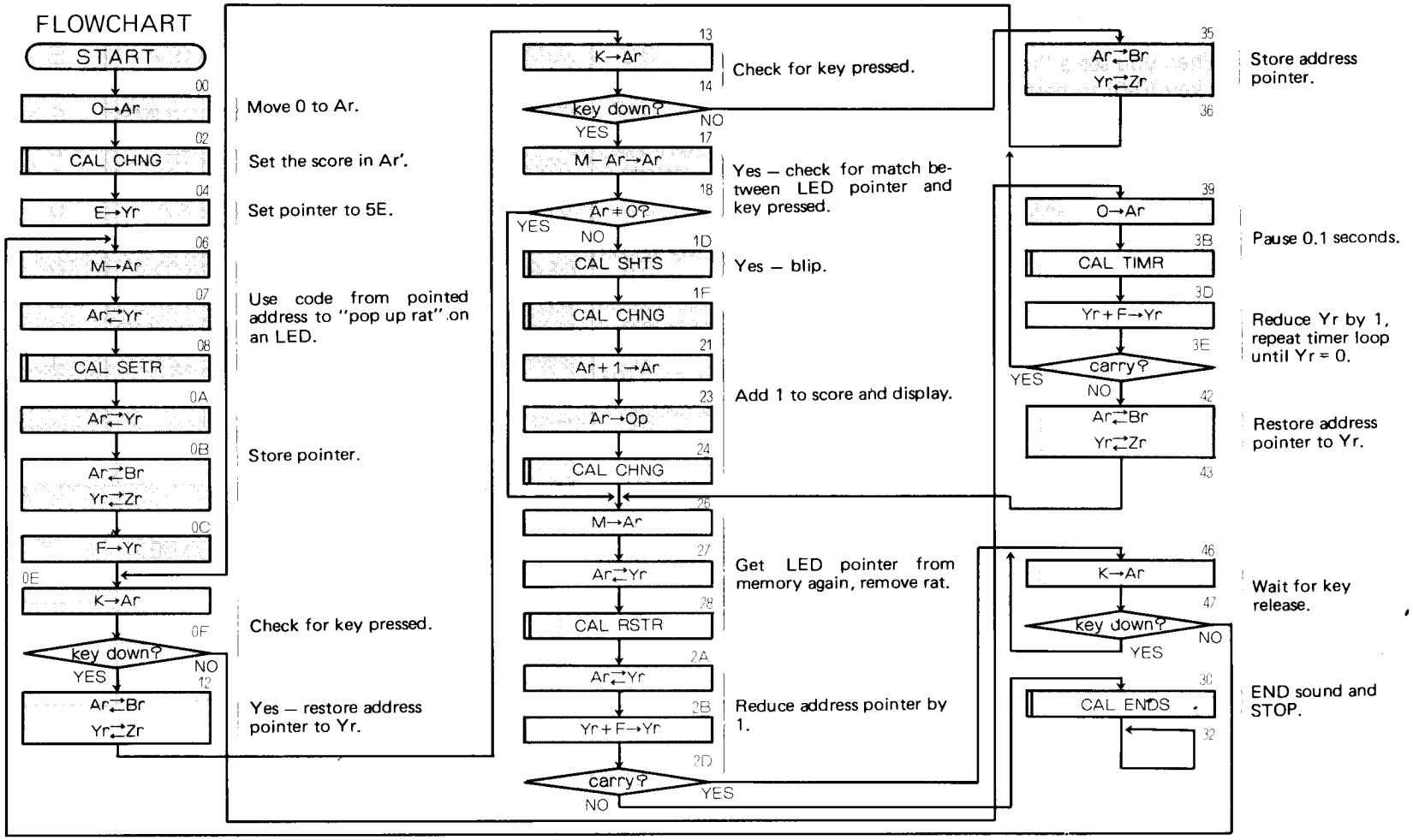
50	51	52	53	54	55	56
5	1	6	2	5	0	4

C) Press RESET, 1, RUN to start. Then watch the binary LEDs.

? Score is displayed

Press one of the keys (0) to (6), corresponding to the LED that is lit, while it is on.

If you are able to "catch all the rats" by pressing all the correct keys, your score will be F.


FLOWCHART
START
00
0→Ar
Move 0 to Ar.
02
CAL CHNG
Set the score in Ar'.
04
E→Yr
Set pointer to 5E.
06
M→Ar
07
Ar⇄Yr
Use code from pointed address to "pop up rat" on an LED.
08
CAL SETR
0A
Ar⇄Yr
0B
Ar⇄Br
Yr⇄Zr
Store pointer.
0C
F→Yr
0E
K→Ar
0F
key down?
YES
NO
12
Ar⇄Br
Yr⇄Zr
Yes – restore address pointer to Yr.
Check for key pressed.
13
K→Ar
Check for key pressed.
14
key down?
YES
NO
17
M−Ar→Ar
18
Ar=0?
YES
NO
Yes – check for match between LED pointer and key pressed.
1D
CAL SHTS
Yes – blip.
1F
CAL CHNG
21
Ar+1→Ar
23
Ar→Op
24
CAL CHNG
Add 1 to score and display.
26
M→Ar
27
Ar⇄Yr
28
CAL RSTR
2A
Ar⇄Yr
2B
Yr+F→Yr
2D
carry?
YES
NO
Get LED pointer from memory again, remove rat.
Reduce address pointer by 1.
35
Ar⇄Br
Yr⇄Zr
36
Store address pointer.
39
0→Ar
3B
CAL TIMR
3D
Yr+F→Yr
3E
carry?
YES
NO
Pause 0.1 seconds.
Reduce Yr by 1, repeat timer loop until Yr = 0.
42
Ar⇄Br
Yr⇄Zr
43
Restore address pointer to Yr.
46
K→Ar
47
key down?
YES
NO
Wait for key release.
30
CAL ENDS
32
END sound and STOP.

●Appendix

CONTINUED FROM PAGE 75:

Some examples to help you to understand program No. 38. Numbers are in hex except where stated:

CARRY AT? :—	0B	13	1D	23	3C
5+4 = 9	No	—	—	—	—
5+5 = A	No	A+6 = 10	—	No	—
		Yes			
	binary				
9+7 = 10 =	(1)0000 = 0		0+6 = 6	6+6 = C	
	Yes	—	No	No	—
A+A = 14 =	(1)0100 = 4		4+6 = A	A+6 = 10	
	Yes	—	No	Yes	—
F+E = 1D =	(1)1101 = D		D+6 = 13		3+C = F
	Yes	—	Yes	—	No
F+F = 1E =	(1)1110 = E		E+6 = 14		4+C = 0
	Yes	—	Yes	—	Yes

CONTINUED FROM PAGE 129:

A reminder about complements:

1 is less than 2. The program checks this by comparing the complement of 1 with the complement of 2. Remember that a number plus its complement always adds up to F (15 decimal). The complement of 1 is 14. The program now takes this complement and adds it to 2; this answer generates a carry. (14 + 2 = 16 decimal or 10 hex, which is greater than F.) Because the result of this addition generates a carry, then the complement involved is greater than the complement of 2. The smaller the number, the larger its complement, therefore, the number complemented (1) is smaller than 2.

CONTINUED FROM PAGE 127:

A different kind of explanation concerning program 73:

Suppose that in 50-5E there is the following data:— 0, 1, 2, 3, 4, ... , D, E. You require this program to reverse the contents so that they look like this:— E, D, C, B . . . . . , 1, 0.

To do this the program uses two loops. The first, or INNER, loop, uses a count that is stored in Ar'. In this loop each adjacent pair of numbers is examined to see which is larger (the technique used is illustrated and explained in program 48). If the first of the pair is smaller, the two are swapped round and then the first pair is examined. In our example the process will look like this:—

Step 1 0, 1, 2 . . . . . , D, E
0 is less than 1, so we get 1, 0, 2 . . . . . , D, E

Step 2 1, 0, 2 . . . . . , D, E
0 is less than 2, so we get 1, 2, 0 . . . . . , D, E

and so on,

until after step 14 we get 1, 2, 3 . . . . . , E, 0

That is the end of the first set of executions of the INNER loop. One is now added to the count for the OUTER loop, which is stored in 5F, and this outer loop is repeated until its count reaches 0 (carry by adding 1 to F). The second set of executions of the inner loop will work like this:—

Step 1 1, 2, 3, ... E, 0 -------- 2, 1, 3, ... E, 0
Step 2 2, 1, 3, ... E, 0 -------- 2, 3, 1, ... E, 0
Step 14 2, 3, 4, ... 1, 0 -------- 2, 3, 4, ... 1, 0

So gradually the numbers are re-arranged with the sorted values appearing at the lower end. In our example, each time the outer loop is executed there is one less physical sort to do, because more and more of the pairs are already in the right order. If you run the program with (2), (RUN), you may be able to hear it getting faster and faster as it goes through the loop each time — each loop is doing less work each time.

Program Title

address	command	machine code
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
0A		
0B		
0C		
0D		
0E		
0F		

address	command	machine code
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
1A		
1B		
1C		
1D		
1E		
1F		

address	command	machine code
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
2A		
2B		
2C		
2D		
2E		
2F		

address	command	machine code
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
3A		
3B		
3C		
3D		
3E		
3F		

address	command	machine code
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		
4A		
4B		
4C		
4D		
4E		
4F		

Program Title

address	command	machine code
00		
01		
02		
03		
04		
05		
06		
07		
08		
09		
0A		
0B		
0C		
0D		
0E		
0F		

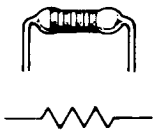
address	command	machine code
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
1A		
1B		
1C		
1D		
1E		
1F		

address	command	machine code
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
2A		
2B		
2C		
2D		
2E		
2F		

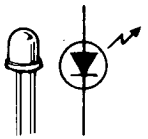
address	command	machine code
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
3A		
3B		
3C		
3D		
3E		
3F		

address	command	machine code
40		
41		
42		
43		
44		
45		
46		
47		
48		
49		
4A		
4B		
4C		
4D		
4E		
4F		

ELECTRICAL PARTS

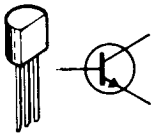


RESISTORS: Resistors are the brown tubular objects with colored bands around them. They are called resistors because they resist the flow of electricity through them. The amount of strength a resistor has to resist the electricity, is measured in units called OHMS. Each registor in your kit has its strength (in ohms) listed near it. The K after some of the numbers stands for thousands. So the "strongest" resistor in the kit has 56K or 56,000 ohms of resistance.



LED: Look at the display area of the kit. You will see seven diodes in a row, and one LED shaped as figure 8. The HEX. LED actually consists of 8 LEDs. Any desired number and most of the letters can be displayed with 7 LEDs. The 8th LED is for decimal point. LED stands for **light emitting diode**. The LED does the same thing as any other diode, except that it gives off light when the electricity passes through it (in the right direction). LEDs last longer and use less electricity than regular light bulbs.

CAPACITORS: Capacitors store and release electricity as a circuit needs it. Their ability to store electricity is measured in units called FARADS, but since a farad is a very large amount, most capacitors are rated in **micro-farads** (μF) and **pico-farads** (pF). A micro-farads is one millionth of a farad and a pico-fared is one millionth of a micro-farad. The capacitors used in this kit are ceramic disc type and are rated in both micro- (μF) and pico-farads (pF).



TRANSISTOR: Transistors have three connections (instead of two like the other parts you have seen). In your kit, the transistor acts as an amplifier to make things louder.

PARTS LIST

Note: Most of these parts are already connected to the Computer. This parts list will serve to remind you of what parts make up your kit.

Description	Description
Battery Holder for 6 AA Cells Battery Terminal, "+" Battery Terminal Spring, "−" Capacitors: 100 pF, ceramic disc type 330 pF, ceramic disc type 0.1 μF, ceramic disc type Frame for Panel Board, Left Frame for Panel Board, Right Key Contact (20) Key Switch with Key Top (20) Large Integrated Circuit, TMS1100 Light Emitting Diodes for Binary Digit Display (7) Light Emitting Diode for Hexadecimal Digit Display Nuts 3mm (46) Panel Board PCB for LED PCB for TMS1100	Resistors: 100 ohm 2.4 K (14) 3 K 5.6 K 30 K 56 K Resonator, 400 kHz, KBR400B Screws, 3 x 6 mm (20) Screws, 3 x 8 mm, black (6) Speaker, 57mm, 8 ohm Speaker Holders (2) Spring Terminals (63) Transistor 2SC 2320 NPN, silicon (or 2SC945 or 2SC828 or 2SC536 or 2SC1815) Wires Yellow 10 cm (26) Black 18 cm (5) Red 30 cm (1)

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U.K.
91 KURRAJONG AVENUE MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

6A4

Printed in Japan