[illegible] 104
Ed. 1

SOME ADVANTAGES OF ZEBRA PROGRAMMING

ZEBRA SIMPLE CODE

Many of the instructions in the Simple Code are easy to remember, for example:

A50	means add what is in location 50 into the arithmetic unit.
S51	means subtract what is in location 51 from the arithmetic unit.
D52	means divide what is in the arithmetic unit by what is in location 52.
T53	means transfer what is in the arithmetic unit into location 53.

Some other instructions are:—

XI	Jump to an instruction which is labelled with Q1. The label Q1 is written immediately before the instruction which it is required to label. 99 such labels may be used.
E2	If the contents of the accumulator are negative, proceed to the next instruction, otherwise jump to an instruction which is labelled with Q2.
Z1	This calls in a subroutine to extract a square root. There are also subroutines for finding many trigonometrical and hyperbolic functions, logarithms, etc.

Relative addresses can also be used. This facility is useful when it is required to repeat the same series of operations a number of times but with different variables each time. When this occurs it may be possible to employ the Simple Code counting routine; for instance

$\left.\begin{array}{l}+024\\ \text{AR5}\\ +1\end{array}\right\}$ means set the count limit to 24, add what is in location 5 into the arithmetic unit, add 1 to the address in the instruction AR5, reduce the count limit by 1 and repeat until the count limit becomes zero.

The effect of this program is to add into the arithmetic unit successively the contents of locations (5, 6 . . . 28). Other instructions can be included inside such a loop; for instance, it would be possible to include an instruction to print the result after each addition has been done.

Programs for including loops within loops are a little more complex but still short. To transpose a matrix requires a program of only 12 instructions. The inversion of a 10 x 10 matrix takes about 2 minutes and of a 30 x 30 matrix about one hour.

When the Simple Code is used all numbers are represented inside the machine in decimal floating-point form, but numbers can be fed into and printed out from the machine in the common everyday form.
The Simple Code also has many other facilities useful to the mathematician, research scientist, linear programmer and others. This is so because it is interpreted inside the machine into the extremely powerful, flexible and unique Zebra Normal Code.

ZEBRA NORMAL CODE

The Normal Code is based on a novel idea. Single letters perform particular basic operations such as add and subtract; but there are 15 such letters (called function digits). These give the programmer the possibility of specifying thousands of different instructions. It is therefore possible to instruct the machine to add, transfer, modify, test, etc., all at the same time, thus making the effective speed of calculation in Zebra greater than the intrinsic electronic speed would suggest.

An address part must appear in a written instruction in order to indicate the location where the necessary number or instruction is stored.
Sometimes the programmer finds it more convenient to have more than one address in an instruction and to be able to refer to instructions which are not stored sequentially.

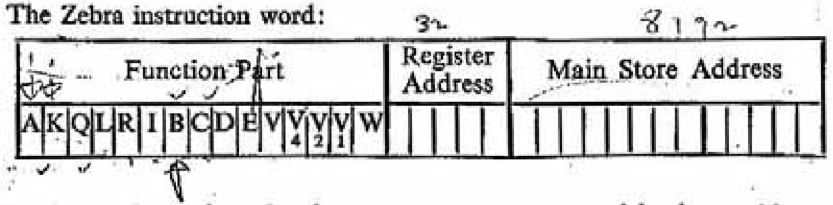
It is possible in some machines to have a program instruction of the form: Operation part; address n, address n′ where the address n′ tells the machine where the next instruction can be found. A machine which operates with this type of instruction is called a 1 + 1 address machine. In other machines, two addresses may be used to indicate the contents of two locations which one may wish to operate upon. In this case the machine is said to be a 2-address machine.

There is a further way in which an address may be used in an instruction. This is when the address refers to a location which contains a special constant and the programmer desires to modify an address by this number. This is called an order modification register (location) sometimes also called, for historical reasons, a 'B' register or 'B' line modification register. With an address of this nature modification is performed automatically.
Zebra can be used as a 1 + 1 address, a 2-address, or a 1 + B modifying address machine, any of 12 immediate access registers being used for order modification.

AQBC 100

A Brief Examination of the Zebra Normal Code instruction Word

The Zebra instruction word:



An instruction written by the programmer is converted in the machine to a 33 binary digit word. Each code letter written is represented by 1 and each code letter omitted is represented by 0. The addresses, written in decimal form, are converted to binary form. The three parts of the instruction word are made up as follows:

(1) The Main Store address consists of 13 binary digits, which allow any one of 8192 locations to be selected.

(2) The 5 digits forming the register address are decoded to give 32 possible selections. There are twelve true registers, each having a word length of 33 binary digits: the other addresses select accumulators, useful constants, etc., and some are used to operate input and output mechanisms. These latter addresses refer to what are called pseudo-registers.

(3) The unusual characteristic of the Zebra instruction word is the size of the function part and the way in which it is used. There are 15 binary digits in this function part and most of these are not decoded at all as is the usual practice. For when the instruction is staticised in the control unit before being executed, each function digit, being a 0 or a 1, operates a switch and causes an elementary operation within the computer. These are called function digits and are completely independent and individual.

Function Digits

The function digits are represented by letters: learning the normal code therefore necessitates the memorising of the functions represented by each of the 15 letters.

With the exception of A, the programmer may write the letters representing the function digits in any order. A must be written first, and if A is not required X must be written instead. This enables the machine to separate one instruction from another when reading a list of instructions from the input tape.

A and X are significant for another reason, and lead to instructions being classified as A instructions or X instructions. An A instruction does not specify the location of the next instruction, so the machines takes its next instruction sequentially. An X instruction specifies the location of the next instruction, and can therefore be used as a jump instruction to another part of the program such as a subroutine.

The interconnection of the four basic parts of the machine, and the direction of flow of information between them, is controlled by the digits A, K, D and E.

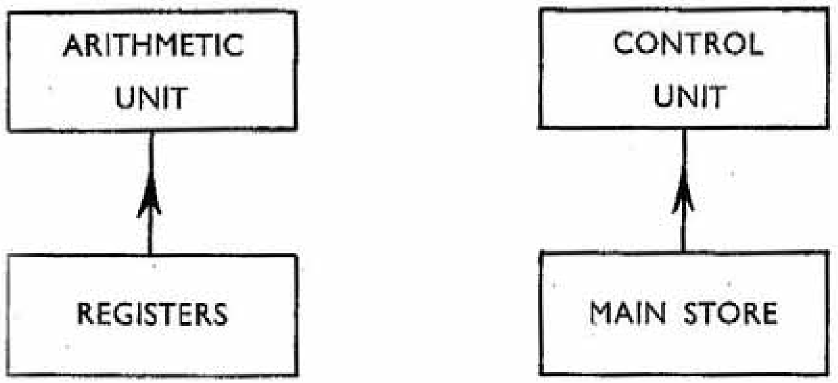
With an A instruction (i.e. A = 1) the main store is connected to the arithmetic unit. The direction of the flow of information between these two units is controlled by another digit, the D digit, as explained later.

With an X instruction (i.e. A = 0) the main store is connected to the control unit, so that the next instruction is set up in the control unit while the previous instruction is being executed.

Next consider the K digit. When K = 1 the registers are connected to the control unit. When K = 0 the registers are connected to the arithmetic unit. In each case the direction of flow of information is controlled by the E digit, as explained later.

The combined action of the A and K digits gives rise to four possible combinations. These are:—

(1) An X instruction (i.e. A = 0) with K = 0 means that a main store location is connected to the control unit, and a register is connected to the arithmetic unit.

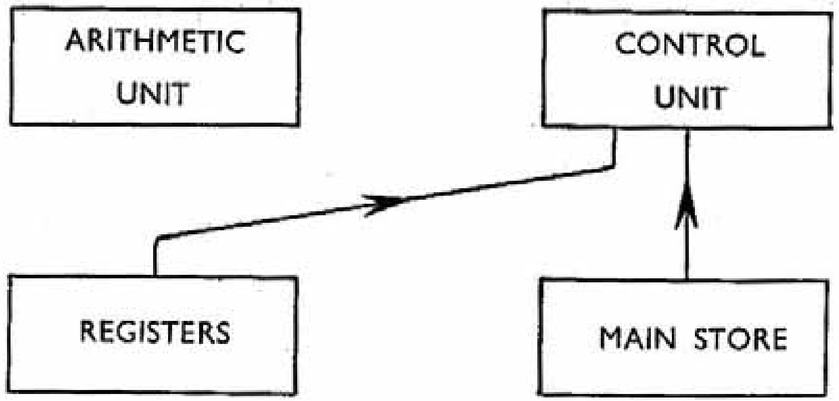


For example:

X100. 10	means "take the next instruction from main store location 100, and add the contents of register 10 to what is already in the arithmetic unit."

In this combination Zebra acts as a 1 + 1 address machine.

(2) An XK instruction means that a main store location and a register are connected to the control unit.

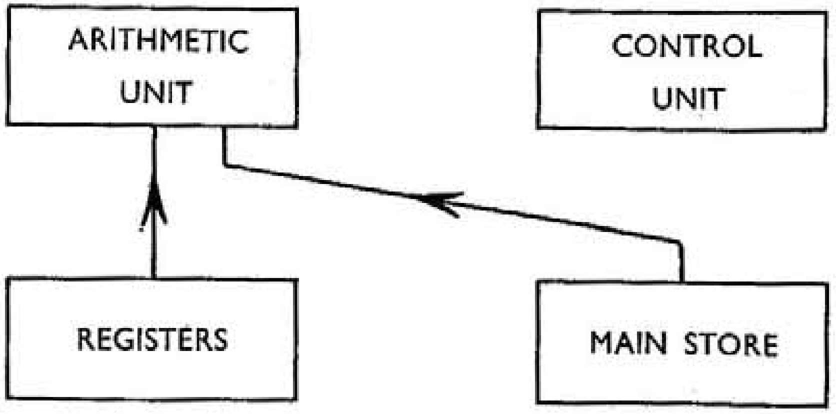


For example:

XK100.10 (which can also be written X100K10) means "take the next instruction from store location 100 and modify it by the contents of register 10."

In this combination, and in (4) below, Zebra uses the register for 'order modification'.

(3) An A instruction with K = 0, means that a main store location and a register are connected to the arithmetic unit.

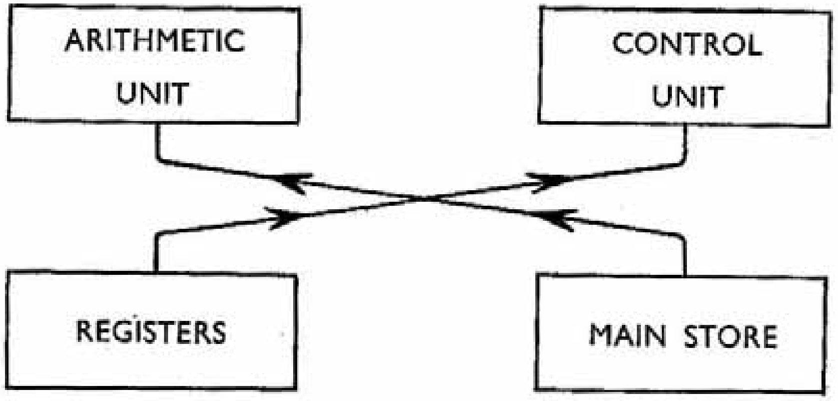


For example:

A100.10 means "add the contents of main store location 100 together with the contents of register 10 to what is already in the arithmetic unit."

In this combination Zebra acts as a 2-address machine.

(4) An A instruction where K is also equal to 1 means that a main store location is connected to the arithmetic unit, and a register to the control unit.



For example:

A100K10 means "add the contents of main store location 100 to what is already in the arithmetic unit, and modify an instruction held in a special register in the control unit by the contents of register 10."

So far only the A and K digits have been examined, but already the power and flexibility of the Zebra normal programming code is becoming evident. The action of the other function digits will now be briefly indicated.

When D is included in an instruction (i.e. D = 1), writing into instead of reading from a main store location occurs. Similarly, with E = 1 writing into a register occurs. For example:

A100D10 means "copy the contents of the arithmetic unit into main store location 100 and then add the contents of register 10 to the arithmetic unit".

X100KE10 means "take the next instruction from main store location 100 and write the existing contents of the control unit into register 10".

In the arithmetic unit, where the actual calculations take place, there are two accumulators called 'A' and 'B'. When the function digit B is absent from a written instruction, the 'A' accumulator is specified; when B is present (i.e. when B = 1) the 'B' accumulator is specified.

When Q is present, a '1' is automatically added into the least significant position of the 'B' accumulator without it being specified by the B digit. The I digit causes subtraction to occur; the L and R digits perform the functions of left and right shifting the accumulators, thereby effectively multiplying or dividing by two. When the C digit is present, the accumulator specified by the B digit is cleared.

The V digits are test digits, and are decoded to test certain significant digits of the numbers contained in the accumulators.

FURTHER PROGRAMMING FEATURES

Zebra is a stored program machine, which means that programs are generally stored before calculations are performed and before the execution of such calculations takes place within the machine. An interpretive program, which is normally stored, interprets instructions written in the Normal Code. By use of other interpretive programs instructions written in other codes may be accepted; in fact Zebra could be made to accept programs written in codes for other machines.

The Simple Code has been devised specially for the user who will not be primarily concerned with maximum speed of operation, such as the Normal Code can provide; indeed, by means of this code a comparatively unskilled programmer can instruct the machine. Not only is the Simple Code much easier to use than the Normal Code, but it has the added advantage that all calculations are performed in a floating-point form. Both codes may be used in conjunction with each other.

Very often problems of an almost identical nature need to be calculated; for such repetitive work a program will only have to be written once, and stored permanently, being modified where necessary. On the other hand a series of non-repetitive problems may arise where totally different programs are needed, and these may be used once and seldom afterwards. For this latter type of work, programs written in Simple Code are preferable—even though the calculation may take longer to perform, since program preparation time would be much shorter.

9550