The Character Set

EDSAC's instruction set consists of 16 fixed-length orders, each 17 bits long. The leftmost 5 bits encode the operation code, followed by a 10-bit address field, and a 2-bit flag to indicate whether the order is an arithmetic or logical operation, respectively. The address field is divided into two 5-bit halves, allowing the machine to operate on 10-bit operands or 20-bit addresses.

The Character Set

EDSAC's character set consists of 40 characters, represented by 5-bit codes. The characters are composed of a digit (0-9), a letter (A-Z), a space, or a sign (+,-). Each character is encoded using one of the 40 codes. The first 9 characters are used for punctuation, and the remaining 31 characters are used for letters, digits, and spaces. The codes are used in pairs to represent two-length codes, such as the digits 0-9.

The 1949 Instruction Set

EDSAC's instructions are executed in a cycle of about 600 microseconds. They are divided into three parts: the address field, the operation code, and the flag. The address field contains the memory address of the operand, and the operation code specifies the type of operation to be performed. The flag indicates whether the operation is arithmetic or logical.

The numerical values in the accumulator and multiplier registers are normally thought of as signed binary fractions, but integer operations could also be done. For example, the order YI could be interpreted as adding the product of the 17-bit integer in N[31] and the 17-bit integer in N[0] and adding the result into bits 0 to 32 of the ARC. With a suitable shift, the integer result can be placed in the senior 17 bits of N[0] ready for storing in memory.

The Initial Orders

The four glass panels on your right contain 20 segments of 5-track paper tape. Reading from left to right and from top to bottom, the first five segments correspond to the initial orders, and the remaining 15 to a program to compute squares. The glass panels contain arrows so a contracted version of the panels are given below. The initial orders were written by David Wheeler in May 1949 to load and enter a paper tape representation of a program.

When EDSAC was started, these initial orders were placed in memory locations 0 to 50 by a mechanism involving unspecifed paper tape processor before execution started from location 0. The glass panels give a paper tape representation of these orders even though no such paper tape ever existed. The following is an annotated listing of this program.

The instruction at location 0 does nothing useful, but the instruction at 1 loads the multiplier register R with a 17-bit pattern 0//0000000000000000 which is also 0 shifted left 11 places. The instruction at 12 loads the multiplier with the constant 25, used as a junk register when the instruction at 15 clears the multiplier. The first real instruction of the program is in location 2, which adds one to its address field, so the next time it is executed it will update the next instruction.

The main assembly loop starts at 6, having locations m[0] to m[5] available as variables and constants in the program. They are used as follows:

- m[0] uses include holding the first character of an order.
- m[1] used to hold the address field of the current order.
- m[2] used as a pointer register for the current instruction.
- m[3] used as a register that contains the address of the order at 11-places ARC.
- m[4] the constant 2 used to operate on the address field.
- m[5] the constant 10 used to check for the end of address digits.

The order at 25 is of the form Trx, initially T0S. It is used to start an order at location 0. This instruction is modified by the code in locations 26 to 28 which add a new order field, so the next time it is executed it will update location 25. Location 26 is the first order to be loaded and must be of the form Trx where m is the address of the last line of the program. It is used by the code in locations 29 and 30 which compare it with the current tension of Trx in 28. If holding is not yet complete execution jumps to 21, otherwise it full through to 31. Note that the instruction at 31 will do no damage, since it just writes a value to the first location following the loaded program. The first real instruction of the program is m[32].

The Character Set

The character set consists of 40 characters, represented by 5-bit codes. The characters are composed of a digit (0-9), a letter (A-Z), a space, or a sign (+,-). Each character is encoded using one of the 40 codes. The first 9 characters are used for punctuation, and the remaining 31 characters are used for letters, digits, and spaces. The codes are used in pairs to represent two-length codes, such as the digits 0-9.

The 1949 Instruction Set

EDSAC's instructions in 1949 were simple and were executed at a rate of about 600 microseconds. They were as follows:

A: Add
S: Sub
R: Shift
C: Conditional branch
X: Extend
Y: Create
Z: Stop

The numerical values in the accumulator and multiplier registers are normally thought of as signed binary fractions, but integer operations could also be done. For example, the order YI could be interpreted as adding the product of the 17-bit integer in N[31] and the 17-bit integer in N[0] and adding the result into bits 0 to 32 of the ARC. With a suitable shift, the integer result can be placed in the senior 17 bits of N[0] ready for storing in memory.

EDSAC's Character Set

The character set consists of 40 characters, represented by 5-bit codes. The characters are composed of a digit (0-9), a letter (A-Z), a space, or a sign (+,-). Each character is encoded using one of the 40 codes. The first 9 characters are used for punctuation, and the remaining 31 characters are used for letters, digits, and spaces. The codes are used in pairs to represent two-length codes, such as the digits 0-9.
The following is an annotated listing of the program:

The Squares Program
This program, written by Maurice Wilkes in June 1949, computes the following table of squares and differences of the numbers 1 to 100.

<table>
<thead>
<tr>
<th>Order</th>
<th>Order</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Holds c, x</td>
</tr>
<tr>
<td>00011</td>
<td>00011111110</td>
<td>goto 110</td>
<td>Order to place in m[52]</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 1</td>
<td>Order to place in m[53]</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 2</td>
<td>Order to place in m[54]</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 3</td>
<td>Order to place in m[55]</td>
</tr>
<tr>
<td>00110</td>
<td>00110110110</td>
<td>T129S m[129]=4, ABC=0</td>
<td>Start of main loop</td>
</tr>
<tr>
<td>00110</td>
<td>00110110110</td>
<td>E48S wr(m[48])</td>
<td>Print a space</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>goto 0</td>
<td>Write a space</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
<tr>
<td>00000</td>
<td>00000000000</td>
<td>data 0</td>
<td>Write a digit</td>
</tr>
</tbody>
</table>

The Green Door
The green door on your left was the Corn Exchange Street entrance to the Mathematical Laboratory where EDSAC was built. By convention, the brain plaque on this door holds the engraved names of those retired members of the Laboratory who used the door in its original location.

Links
http://www.cl.cam.ac.uk/U0CCL/misc/EDSAC99
This links to the Martin Campbell-Kelly’s excellent EDSAC simulator and related documents.
http://www.cl.cam.ac.uk/U0CCL/misc/EDSAC09
This links to pages relating to the celebration, held in Cambridge in April 1999, of the 50th anniversary of the EDSAC 1 Computer.
http://www.cl.cam.ac.uk/~m/e/edsc.html
This links to a shell based EDSAC simulator that runs on Pentium based Unix systems. It was designed to be educational, but it has interactive debugging allowing single step execution, debugging of breakpoints and convenient inspection and setting of memory and register values. It can be used to explore the execution of the programs described in this poster. This simulator appears as a demonstration program in the Congolese BCLP system (http://www.cl.cam.ac.uk/~m/e/BCLP.html).

The corrected tape segments etched on the Tea Room glass panels: