Reimagining EDSAC in Open Source

Mary Bennett
Overview
Predecessors
Building EDSAC
EDSAC
EDSAC’s Memory
EDSAC Architecture

- **CONTROL**
  - Sequence Control
  - Order Tank

- **ALU**
  - Acc
  - Multiplier
  - Multiplicand

- **STORE**
  - 1024 words of 18 bits

- **INPUT** → **STORE** → **ALU** → **OUTPUT**
EDSAC Data

(a) short integer

(b) long integer

(c) short fraction

(d) long fraction
EDSAC Instructions

A \( n \)  Add value at storage location \( n \) to the accumulator
S \( n \)  Subtract value at storage location \( n \) from the accumulator
...
Y      Round the accumulator to 34 bits
Z      Stop the machine

Total of 18 instructions
Program Input
Job Queue
Teleprinter
Bootstrap
## EDSAC Initial Orders

<table>
<thead>
<tr>
<th>Order bit pattern</th>
<th>Loc</th>
<th>Order</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101 0 000000000 0</td>
<td>0</td>
<td>TOS</td>
<td>m[n][0] = A; ABC = 0</td>
<td>Put 10&lt;&lt;11 in R</td>
</tr>
<tr>
<td>10101 0 000000010 0</td>
<td>1</td>
<td>H2S</td>
<td>k = m[n][2]</td>
<td></td>
</tr>
<tr>
<td>00101 0 000000000 0</td>
<td>2</td>
<td>TOS</td>
<td>m[n][0] = A; ABC = 0</td>
<td>Jump to main loop</td>
</tr>
<tr>
<td>00011 0 000000110 1</td>
<td>3</td>
<td>ESS</td>
<td>goto 6</td>
<td></td>
</tr>
<tr>
<td>00000 0 000000001 0</td>
<td>4</td>
<td>PIS</td>
<td>data 2</td>
<td>The constant 2</td>
</tr>
<tr>
<td>00000 0 000000101 0</td>
<td>5</td>
<td>PSS</td>
<td>data 10</td>
<td>The constant 10</td>
</tr>
<tr>
<td>00101 0 000000000 0</td>
<td>6</td>
<td>TOS</td>
<td>m[n][0] = A; ABC = 0</td>
<td>Start of the main loop</td>
</tr>
<tr>
<td>01000 0 000000000 0</td>
<td>7</td>
<td>IOS</td>
<td>m[n][0] = rdxch()</td>
<td>Get operation code</td>
</tr>
<tr>
<td>00011 0 000000000 0</td>
<td>8</td>
<td>AOS</td>
<td>k = m[n][0]</td>
<td>Put s in A</td>
</tr>
<tr>
<td>00100 0 000000010 0</td>
<td>9</td>
<td>R1S</td>
<td>ABC = &gt; = 0</td>
<td>Shift and store it</td>
</tr>
<tr>
<td>00101 0 000000000 0</td>
<td>10</td>
<td>TOL</td>
<td>m[n][0] = A; ABC = 0</td>
<td>So that it becomes the</td>
</tr>
<tr>
<td>01000 0 000000010 0</td>
<td>11</td>
<td>I2S</td>
<td>m[n][2] = rdxch()</td>
<td>the senior 3 bits of m[n][0]</td>
</tr>
<tr>
<td>11110 0 000000010 0</td>
<td>12</td>
<td>A2S</td>
<td>A = m[n][2]</td>
<td>m[1] is now zero</td>
</tr>
<tr>
<td>01100 0 000000010 0</td>
<td>13</td>
<td>SSS</td>
<td>k = m[n][0]</td>
<td>Put next ch in m[n][2]</td>
</tr>
<tr>
<td>00111 0 000000101 1</td>
<td>14</td>
<td>E2IS</td>
<td>if A = 0 goto 21</td>
<td>Put ch in A</td>
</tr>
<tr>
<td>00101 0 000000011 0</td>
<td>15</td>
<td>T3S</td>
<td>m[n][3] = A; ABC = 0</td>
<td>A = m[n][2] + 10&lt;&lt;11</td>
</tr>
<tr>
<td>11111 0 000000001 0</td>
<td>16</td>
<td>V1S</td>
<td>ABC = m[n][1] + R</td>
<td>A = m[n][3] + 10&lt;&lt;11</td>
</tr>
<tr>
<td>11100 0 000000100 0</td>
<td>17</td>
<td>L2S</td>
<td>A &lt;&lt; = 0</td>
<td>Shift 5 more places</td>
</tr>
<tr>
<td>11100 0 000000010 0</td>
<td>18</td>
<td>A2S</td>
<td>A = m[n][2]</td>
<td>Shift 5 more places</td>
</tr>
<tr>
<td>00101 0 000000000 0</td>
<td>19</td>
<td>T1S</td>
<td>m[n][1] = A; ABC = 0</td>
<td>Add the new digit</td>
</tr>
<tr>
<td>00011 0 000000101 1</td>
<td>20</td>
<td>E11S</td>
<td>goto 11</td>
<td>Store back in m[n][1]</td>
</tr>
<tr>
<td>00100 0 000000010 0</td>
<td>21</td>
<td>R4S</td>
<td>ABC = &gt; = 4</td>
<td>Repeat from 11</td>
</tr>
<tr>
<td>11100 0 000000001 0</td>
<td>22</td>
<td>A1S</td>
<td>A = m[n][1]</td>
<td>A = 2, if ch = ‘G’ (+12)</td>
</tr>
<tr>
<td>11000 0 000000000 0</td>
<td>23</td>
<td>L0L</td>
<td>ABC = &gt; = 1</td>
<td>A = 1, if ch = ‘L’ (+26)</td>
</tr>
<tr>
<td>11100 0 000000000 0</td>
<td>24</td>
<td>AOS</td>
<td>k = m[n][0]</td>
<td>k = 0, if ch = ‘L’</td>
</tr>
<tr>
<td>00101 0 000001111 0</td>
<td>25</td>
<td>T3IS</td>
<td>m[n][1] = A; ABC = 0</td>
<td>k = 1, if ch = ‘L’</td>
</tr>
<tr>
<td>11100 0 000001100 0</td>
<td>26</td>
<td>A2SS</td>
<td>A = m[n][25]</td>
<td>Store the order in next location</td>
</tr>
<tr>
<td>11100 0 000000100 0</td>
<td>27</td>
<td>A4S</td>
<td>m[n][4] = A</td>
<td>Increment the address</td>
</tr>
<tr>
<td>00111 0 000001101 0</td>
<td>28</td>
<td>U2SS</td>
<td>m[n][25] = A</td>
<td>field of m[n][25]</td>
</tr>
<tr>
<td>01100 0 000001111 0</td>
<td>29</td>
<td>S3IS</td>
<td>A = m[n][31]</td>
<td>m[n][4] holds 2</td>
</tr>
<tr>
<td>11001 0 000000010 0</td>
<td>30</td>
<td>GSS</td>
<td>if A &lt; 0 goto 6</td>
<td>Update m[n][25]</td>
</tr>
<tr>
<td>00100 0 000001111 0</td>
<td>31</td>
<td>A1S</td>
<td>A = m[n][1]</td>
<td>Jump to 6, if there are</td>
</tr>
</tbody>
</table>

**EDSAC Initial Orders and Squares Program**, Martin Richards, University of Cambridge Computer Laboratory
## Wheeler Jumps

<table>
<thead>
<tr>
<th>Location</th>
<th>Order</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p - 1$</td>
<td>$T0S$</td>
<td>Move acc to loc 0 and zero acc</td>
</tr>
<tr>
<td>$p$</td>
<td>$ApS$</td>
<td>Store current loc in acc</td>
</tr>
<tr>
<td>$p + 1$</td>
<td>$EqS$</td>
<td>Jump to loc $q$ if acc positive</td>
</tr>
<tr>
<td>$q$</td>
<td>$A3S$</td>
<td>Loc 3 contains $E2S$, creates $Ep+2S$</td>
</tr>
<tr>
<td>$q + 1$</td>
<td>$Tq+rS$</td>
<td>Overwrite link order</td>
</tr>
<tr>
<td>$q + r - 1$</td>
<td>$T0S$</td>
<td>Zero accumulator</td>
</tr>
<tr>
<td>$q + r$</td>
<td>$E2S$</td>
<td>Link order which will be modified</td>
</tr>
</tbody>
</table>

### Main program

### Prologue

### Epilogue
THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER

With special reference to the EDSAC and the use of a library of subroutines

by

MAURICE V. WILKES
Director of the Mathematical Laboratory of the University of Cambridge

DAVID J. WHEELER
and

STANLEY GILL

1951
Famous Programs

$$180(2^{127} - 1)^2 + 1$$
Popular Appreciation

The “brain” [computer] may one day come down to our level [of the common people] and help with our income-tax and book-keeping calculations. But this is speculation and there is no sign of it so far.
Reimagined EDSAC
Field Programmable Gate Array (FPGA)
EDSAC Verilog
Reimagined Paper Tape
Reimagined Paper Tape Reader

- UART
- LDR signals
- Motor, LDRs & LEDs are in here
- Motor ON/OFF
- Forward/Reverse
- Arduino
- Tape comes out here
- Tape goes in here
Sensor Design
Sensor Circuit

5V

22kΩ
to analogue input

0V
Raw Sensor Data
Calculating Rate of Signal Change

\[
slope = \frac{dx}{dt}
\]
Calculating Rate of Signal Change

\[ \text{slope} = \frac{dx}{dt} \]

\[ \text{slope} = \frac{\Delta x}{\Delta t} \]
Calculating Rate of Signal Change

\[ \text{slope} = \frac{dx}{dt} \]

\[ \text{slope}_n = \frac{x_n - x_{n-1}}{t_n - t_{n-1}} \]
Calculating Rate of Signal Change

\[
\text{slope} = \frac{dx}{dt}
\]

\[
\text{slope} = \frac{\Delta x}{\Delta t}
\]

\[
\text{slope}_n = \frac{x_n - x_{n-1}}{t_n - t_{n-1}}
\]

\[
\text{slope}'_n = x_n - x_{n-1}
\]
Exponential Smoothing
Exponential Smoothing

\[ S_t = 0.2 \cdot x_t + 0.8 \cdot S_{t-1} \]
Exponential Smoothing

\[ S_t = 0.2 \cdot x_t + 0.8 \cdot S_{t-1} \]
Sprocket
Paper Tape Reader Output
Paper Tape Reader Demo

/get/ttyUSB0

Get ready
Go
T0S
H2S
T0S
E6S
P1S
P5S
T0S
I0S
A0S
R16S
T0L
I2S
A2S
S5S

Autoscroll

No line ending

115200 baud
Mercury Delay Line
Delay Line Fluids
Delay Line Fluids
Delay Line Fluids
Delay Line Fluids
Delay Line Fluids
Reimagined Delay Line

- Arduino Cuttlefish
- Speaker
- Lots of wire
- Microphone
- Acoustic Foam
Mirrors

- Tube
- Mirror
- Microphone
- Speaker
Mirrors

- Tube
- Mirror
- Microphone
- Speaker
Mirrors

- Mirror
- Tube
- Microphone
- Speaker
Mirrors

- Tube
- Microphone
- Mirror
- Speaker
“The Actual Solution”

- Arduino Cuttlefish
- Speaker
- Lots of wire
- Microphone
- Acoustic Foam
Microphone Circuit

- Mic
- 5V
- C1 0.1μF
- R1 (value varies)
- 10KΩ R2
- LM386
- C2 10μF
- C4 100μF
- C3 10μF
- C5 0.047μF
- 10Ω R3

(plugs into analog terminal A0 on the arduino)
(plugs into GND terminal of arduino)
Microphone Circuit
Testing the Microphone Circuit
Speaker Circuit
Summary
Thank You
Contact

• Make your own EDSAC peripherals:
  • github.com/embecosm/edsac-peripherals

• Hatim Kanchwala Verilog EDSAC:
  • github.com/librecores/gsoc-museum-edsac

• Chip Hack repository:
  • github.com/embecosm/chiphack

• More information on the myStorm:
  • mystorm.uk
Questions?

www.embecosm.com