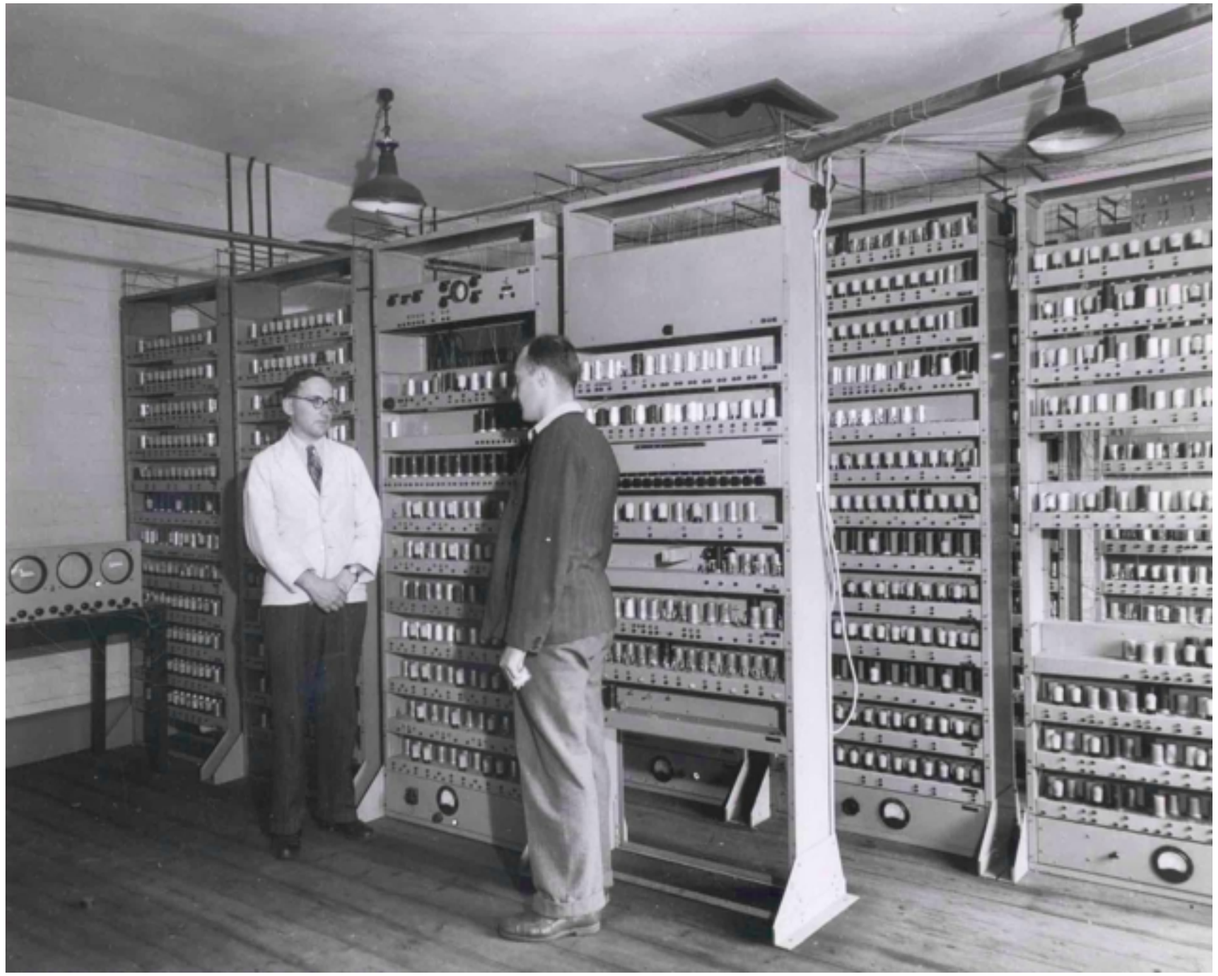
FizzBuzz EDSAC

Olve Maudal



A 5 minute lightning talk at ACCU 2015, April 24, Bristol, UK

https://youtu.be/x-vS0WcJyNM

The EDSAC 1951 film
abridged version

Commentary by
M. V. Wilkes

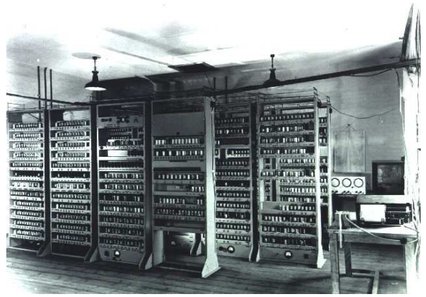
EDSAC Initial Orders and Squares Program

Martin Richards

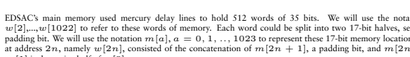
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

EDSAC

EDSAC (Electronic Delay Storage Automatic Computer), pictured below, was the world's first stored-program computer to operate a regular computing service. Maurice Wilkes lead the team responsible for its design and construction. It ran its first program successfully on May 6, 1949.

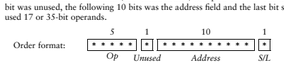


EDSAC's main memory used mercury delay lines to hold 512 words of 35 bits. We will use the notation: $w[0]$, $w[2]$,..,$w[1022]$ to refer to these words of memory. Each word could be split into two 17-bit halves, separated by a padding bit. We will use the notation $m[a]$, $a = 0, 1, .., 1023$ to represent these 17-bit memory locations. The word at address $2n$, namely $w[2n]$, consisted of the concatenation of $m[2n+1]$, a padding bit, and $m[2n]$. Note that $m[1]$ is the senior half of $w[0]$.



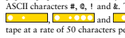
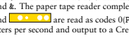
The machine had two central registers visible to the user: the 71-bit accumulator and the 35-bit multiplier register. We will use the notation `ABC` to represent the whole accumulator, and `A` and `AB` to represent its senior 17 and 35 bits, respectively. We will use `RS` to represent the whole multiplier register and `R` to represent its senior 17 bits. The leftmost bit of each register was the sign bit and the remaining bits form a binary fraction.

EDSAC's machine instructions (also called orders) occupied 17 bits. The leftmost 5 bits was the operation code, the next bit was unused, the following 10 bits was the address field and the last bit specified (where appropriate) whether the order used 17 or 35-bit operands.



Orders were punched on paper tape and consisted of: a character that directly gave the 5-bit operation code, followed by zero or more decimal digits giving the address, and terminated by `S` or `L` specifying the operand length bit. For example, `R16S` assembled to `00100 0 0000010000 0` and `T11L` to `00101 0 0000001011 1`. Note that the characters `R` and `T` had codes 4 and 5, respectively.

The Character Set

EDSAC used 5-bit integers (0 to 31) to represent characters using two shifts: letters and figures. In letter shift the codes 0 to 31 respectively represented: `P`, `Q`, `W`, `E`, `R`, `T`, `Y`, `U`, `I`, `O`, `J`, *figs*, `S`, `Z`, `K`, *lets*, *null*, `F`, *cr*, `D`, *sp*, `H`, `N`, `M`, *lf*, `L`, `X`, `G`, `A`, `B`, `C` and `V`. In figure shift the encoding was as follows: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ?, *figs*, *, +, (, *lets*, *null*, $, *cr*, ;, *sp*, £, ,, ., */*,), /, #, -, ?, : and =. In these tables, *figs*, *cr*, *sp* and *lf* denote figure shift, carriage return, space and line feed, and on the paper tape perforator their keys were labelled π, θ, ϕ and Δ, respectively. In this document, these codes correspond to the ASCII characters `#`, `@`, ! and `&`. The paper tape reader complemented the high order bit of each 5-bit character, so the rows  and  are read as codes 0(`P`), 7(`U`) and 27(`G`), respectively. The machine could read paper tape at a rate of 50 characters per second and output to a Creed teleprinter at nearly 7 characters per second.

The 1949 Instruction set

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

`A`*n*`S`:	`A` += $m[n]$	`A`*n*`L`:	`AB` += $w[n]$
`S`*n*`S`:	`A` -= $m[n]$	`S`*n*`L`:	`AB` -= $w[n]$
`H`*n*`S`:	`R` += $m[n]$	`H`*n*`L`:	`RS` += $w[n]$
`V`*n*`S`:	`AB` += $m[n]$ * `R`	`V`*n*`L`:	`ABC` += $w[n]$ * `RS`
`N`*n*`S`:	`AB` -= $m[n]$ * `R`	`N`*n*`L`:	`ABC` -= $w[n]$ * `RS`
`T`*n*`S`:	$m[n]$ = `A`; `ABC` = 0	`T`*n*`L`:	$w[n]$ = `AB`; `ABC` = 0
`U`*n*`S`:	$m[n]$ = `A`	`U`*n*`L`:	$w[n]$ = `AB`
`C`*n*`S`:	`AB` += $m[n]$ & `R`	`C`*n*`L`:	`ABC` += $w[n]$ & `RS`

`R`*n*`S`, `R`*n*`L`: Shift `ABC` right arithmetically by the number of places corresponding to the position of the least significant one in the shift instruction. For example, `R0L`, `R1S`, `R16S` and `R0S` shift by 1, 2, 6 and 15 places, respectively.

`L`*n*`S`, `L`*n*`L`: Shift `ABC` left arithmetically by the number of places corresponding to the position of the least significant one in the shift instruction. For example, `L0L`, `L1S`, `L16S`, `L64S` and `L0S` shift by 1, 2, 6, 8 and 13 places, respectively.

`E`*n*`S`: if `A` >= 0 goto n

`G`*n*`S`: if `A` < 0 goto n

`I`*n*`S`: Place the next paper tape character in the least significant 5 bits of $m[n]$.

`O`*n*`S`: Output the character in the most significant 5 bits of $m[n]$.

`F`*n*`S`: Verify the last character output.

`X`*n*`S`: No operation.

`Y`*n*`S`: Add a one to bit position 35 of `ABC`, counting the sign bit as bit zero. This effectively rounds `ABC` up to 34 fractional bits.

`Z`*n*`S`: Stop the machine and ring a bell.

The numerical values in the accumulator and multiplier registers are normally thought of as signed binary fractions, but integer operations could also be done easily. For example, the order `V1S` can be interpreted as adding the product of the 17-bit signed integer in `m[1]` and to the 17-bit integer in `RS` and adding the result into bits 0 to 32 of the `ABC`. With a suitable shift, the integer result can be placed in the senior 17 bits of `A` ready for storing in memory.

Initial Orders

The four glass panels on your right contain 20 segments of 5 track paper tape. Reading from right to left 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 errors so a corrected 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 30 by a mechanism involving uniselectors before execution stared 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.

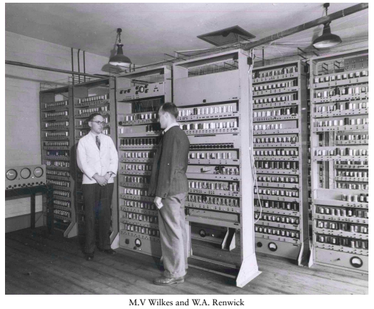
Order bit pattern	Loc	Order	Meaning	Comment
00101 0 0000000000 0	0:	`T0S`	$m[0]$=`A`; `ABC`=0	
10101 0 0000000010 0	1:	`H2S`	`R`=$m[2]$	Put 10<<11 in `R`
00101 0 0000000000 0	2:	`T0S`	$m[0]$=`A`; `ABC`=0	
00011 0 0000000110 0	3:	`E6S`	`goto 6`	Jump to main loop
00000 0 0000000001 0	4:	`P1S`	`data 2`	The constant 2
00000 0 0000000101 0	5:	`P5S`	`data 10`	The constant 10
00101 0 0000000000 0	6:	`T0S`	$m[0]$=`A`; `ABC`=0	Start of the main loop
01000 0 0000000000 0	7:	`I0S`	$m[0]$=`rdch()`	Get operation code
11100 0 0000000000 0	8:	`A0S`	`A`+=$m[0]$	
00100 0 0000010000 0	9:	`R16S`	`ABC`>>=6	Shift and store it
00101 0 0000000000 1	10:	`T0L`	$w[0]$=`AB`; `ABC`=0	so that it becomes the senior 5 bits of $m[0]$
				$m[1]$ is now zero
01000 0 0000000010 0	11:	`I2S`	$m[2]$=`rdch()`	Put next ch in $m[2]$
11100 0 0000000010 0	12:	`A2S`	`A`+=$m[2]$	Put ch in `A`
01100 0 0000000101 0	13:	`S5S`	`A`-=$m[5]$	`A`=ch-10
00011 0 0000010101 0	14:	`E21S`	`if A>=0 goto 21`	Jump to 21, if ch>=10
00101 0 0000000011 0	15:	`T3S`	$m[3]$=`A`; `ABC`=0	Clear `A`, $m[3]$ is junk
11111 0 0000000001 0	16:	`V1S`	`AB`+=$m[1]$*`R`	`A` = $m[1]$*(10<<11)
11001 0 0000001000 0	17:	`L8S`	`A`<<=5	Shift 5 more places
11100 0 0000000010 0	18:	`A2S`	`A`+=$m[2]$	Add the new digit
00101 0 0000000001 0	19:	`T1S`	$m[1]$=`A`; `ABC`=0	Store back in $m[1]$
00011 0 0000001011 0	20:	`E11S`	`goto 11`	Repeat from 11
				`A`=2, if ch='S' (=12)
				`A`=15, if ch='L' (=25)
00100 0 0000000100 0	21:	`R4S`	`ABC`>>=4	lenbit=0, if ch='S'
				lenbit=1, if ch='L'
11100 0 0000000001 0	22:	`A1S`	`A`+=$m[1]$	Add in the address
11001 0 0000000000 1	23:	`L0L`	`ABC`<<=1	Shift to correct position
11100 0 0000000000 0	24:	`A0S`	`A`+=$m[0]$	Add in the operation field
00101 0 0000011111 0	25:	`T31S`	$m[31]$= `A`; `ABC`=0	Store the order
				in next location
11100 0 0000011001 0	26:	`A25S`	`A`+=$m[25]$	Increment the address
				field of $m[25]$
11100 0 0000000100 0	27:	`A4S`	`A`+=$m[4]$	$m[4]$ holds 2
00111 0 0000011001 0	28:	`U25S`	$m[25]$=`A`	Update $m[25]$
01100 0 0000011111 0	29:	`S31S`	`A`-=$m[31]$	Jump to 6, if there are
11011 0 0000000110 0	30:	`G6S`	`if A<0 goto 6`	more orders to load

The instruction at location 0 does nothing useful, but the instruction at 1 loads the multiplier register R with a 17-bit pattern 00101000000000000 which is also 10 shifted left 11 places. The instruction instruction at 2 (`T0S`) assembles into exactly this bit pattern, so is used both as data and as an instruction to clear $m[0]$. The instruction at 3 skips to location 6 over the instructions at 4 and 5 that assemble as the 17-bit constants 2 and 10, respectively.

The main assembly loop starts at 6, leaving 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,
$m[2]$	initially 001010...0 as discussed above but also
	used for characters other than the first of an order,
$m[3]$	used as a junk register when the instruction at 15 clears `ABC`,
$m[4]$	the constant 2 used at 27 to add one to an address field,
$m[5]$	the constant 10 used to check for the end of address digits.

The order at 25 is of the form `T`*n*`S`, initially `T31S`. It is used to store an order at location n. This instruction is modified by the code in locations 26 to 28 which adds one to its address field, so the next time it is executed it will update the next location. Location 31 is the first order to be loaded and must be of the form `T`*n*`S` where $n-1$ is the address of last instruction of the program. It is used by the code in locations 29 and 30 which compares it with the current version of `T`*n*`S` in 25. If loading is not yet complete execution jumps to 11, otherwise it fall 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 in $m[32]$.



M.V Wilkes and W.A. Renwick

The Squares Program

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

1	1	1
2	4	3
3	9	5
....	.....	.....
98	9604	195
99	9801	197
100	10000	199

The following is an annotated listing of the program.

Order bit pattern	Loc	Order	Meaning	Comment
00101 0 0001111011 0	31:	`T123S`	`m`[123]=`A`; `ABC`=0	The required first word
00011 0 0001010100 0	32:	`E84S`	`goto 84`	Jump to start
00000 0 0000000000 0	33:	`P0S`	`data 0`	For the next decimal digit
00000 0 0000000000 0	34:	`P0S`	`data 0`	For the current power of ten
00100 1 1100010000 0	35:	`P10000S`	`data 10000<<1`	The table of 16-bit
00000 0 1111101000 0	36:	`P1000S`	`data 1000<<1`	powers of ten
00000 0 0001100100 0	37:	`P100S`	`data 100<<1`	
00000 0 0000001010 0	38:	`P10S`	`data 10<<1`	
00000 0 0000000001 0	39:	`P1S`	`data 1<<1`	
00001 0 0000000000 0	40:	`Q0S`	`data 1<<12`	00001 in MS 5 bits,
				used to form digits
01011 0 0000000000 0	41:	`#S`	`data 11<<12`	Figure shift character
11100 0 0000101000 0	42:	`A40S`		End limit for values
				placed in $m[52]$
10100 0 0000000000 0	43:	`!S`	`data 20<<12`	Space character
11000 0 0000000000 0	44:	`&S`	`data 24<<12`	Line feed character
10010 0 0000000000 0	45:	`@S`	`data 18<<12`	Carriage return character
01001 0 0000101011 0	46:	`D43S`	`wr`(`m`[43])	Write a space
01001 0 0000100001 0	47:	`D33S`	`wr`(`m`[33])	Write a digit
00000 0 0000000000 0	48:	`P0S`	`data 0`	The number to print
11100 0 0000101110 0	49:	`A46S`	`A`+=`m`[46]	Print subroutine entry point
00101 0 0001000001 0	50:	`T65S`	`m`[65]=`A`; `ABC`=0	Put D43S in $m[65]$
00101 0 0010000001 0	51:	`T129S`	`m`[129]=`A`; `ABC`=0	Clear `A`
11100 0 0000100011 0	52:	`A35S`	`A`+=`m`[35]	`A` is next power of ten,
				$m[52]$ cycles through
				`A35S`, `A36S`, `A37S`,
				`A38S` and `A39S`
00101 0 0000100010 0	53:	`T34S`	`m`[34]=`A`; `ABC`=0	Store it in $m[34]$
00011 0 0000111101 0	54:	`E61S`	`goto 61`	
00101 0 0000110000 0	55:	`T48S`	`m`[48]=`A`; `ABC`=0	Store value to be printed
11100 0 0000101111 0	56:	`A47S`	`A`+=`m`[47]	Store instruction D33S
00101 0 0001000001 0	57:	`T65S`	`m`[65]=`A`; `ABC`=0	in $m[65]$
11100 0 0000100001 0	58:	`A33S`	`A`+=`m`[33]	Increment the decimal digit
11100 0 0000101000 0	59:	`A40S`	`A`+=`m`[40]	held in the MS 5 bits
00101 0 0000100001 0	60:	`T33S`	`m`[33]=`A`; `ABC`=0	of $m[33]$
11100 0 0000110000 0	61:	`A48S`	`A`+=`m`[48]; `ABC`=0	Get value to print
11100 0 0000100010 0	62:	`S34S`	`A`-=`m`[34]	Subtract a power of 10
00011 0 0000110111 0	63:	`E55S`	`if A>=0 goto 55`	Repeat, if positive
11100 0 0000100010 0	64:	`A34S`	`A`+=`m`[34]	Add back the power of 10
00000 0 0000000000 0	65:	`P0S`	`data 0`	This is replaced by either
				`D43S` to write a space, or
				`D33S` to write a digit
00101 0 0000110000 0	66:	`T48S`	`m`[48]=`A`; `ABC`=0	Set the value to print
00101 0 0000100001 0	67:	`T33S`	`m`[33]=`A`; `ABC`=0	Set digit to 0
11100 0 0000110100 0	68:	`A52S`	`A`+=`m`[52]	Increment the address field
11100 0 0000000100 0	69:	`A4S`	`A`+=`m`[4]	of the instruction
00111 0 0000110100 0	70:	`U52S`	`m`[52]=`A`	in $m[52]$
01100 0 0000010010 0	71:	`S42S`	`A`-=`m`[42]	Compare with `A40S` and
11011 0 0000110011 0	72:	`G51S`	`if A<0 goto 51`	Repeat, if more digits
11100 0 0001110101 0	73:	`A117S`	`A`+=`m`[117]	Put `A35S` back
00101 0 0000110100 0	74:	`T52S`	`m`[52]=`A`; `ABC`=0	in $m[52]$
00000 0 0000000000 0	75:	`P0S`	`data 0`	To hold the return jump
				instruction which is
				`E95S`, `E110S` or `E118S`
00000 0 0000000000 0	76:	`P0S`	`data 0`	Holds x
00000 0 0000000000 0	77:	`P0S`	`data 0`	Holds x^2
00000 0 0000000000 0	78:	`P0S`	`data 0`	Holds previous x^2
00000 0 0000000000 0	79:	`P0S`	`data 0`	Holds Δx^2
00011 0 0001101110 0	80:	`E110S`	`goto 110`	Order to place in $m[52]$
00011 0 0001110110 0	81:	`E118S`	`goto 118`	Order to place in $m[52]$
00000 0 0001100100 0	82:	`P100S`	`data 100<<1`	End limit for x
00011 0 0001011111 0	83:	`E95S`	`goto 95`	Order to place in $m[52]$
01001 0 0000101001 0	84:	`D41S`	`wr`($m[41]$)	Write figure shift
00101 0 0010000001 0	85:	`T129S`	`m`[129]=`A`; `ABC`=0	Start of main loop
01001 0 0000101100 0	86:	`D44S`	`wr`($m[44]$)	Write line feed
01001 0 0000101101 0	87:	`D45S`	`wr`($m[45]$)	Write carriage return
11100 0 0001001100 0	88:	`A76S`	`A`+=$m[76]$; `ABC`=0	Get x
11100 0 0000000100 0	89:	`A4S`	`A`+=$m[4]$	Increment it
00111 0 0001001100 0	90:	`U76S`	$m[76]$=`A`	and store it back in x
00101 0 0000110000 0	91:	`T48S`	$m[48]$=`A`; `ABC`=0	Put it also in $m[48]$
				for printing
11100 0 0001010011 0	92:	`A83S`	`A`+=$m[83]$	Put return jump `E95S`
00101 0 0001001011 0	93:	`T75S`	$m[75]$=`A`; `ABC`=0	into $m[75]$
00011 0 0000110001 0	94:	`E49S`	`goto 49`	Enter the print subroutine
01001 0 0000101011 0	95:	`D43S`	`wr`($m[43]$)	Write a space
01001 0 0000101011 0	96:	`D43S`	`wr`($m[43]$)	Write a space
10101 0 0001001100 0	97:	`H76S`	`R`=$m[76]$	Multiply x by
11111 0 0001001100 0	98:	`V76S`	`ABC`+=$m[76]$*`RS`	itself and
11001 0 0000100000 0	99:	`L64S`	`ABC`<<8	re-position
11001 0 0000100000 0	100:	`L32S`	`ABC`<<7	the result
00111 0 0001001101 0	101:	`U77S`	$m[77]$=`A`	Store in location for x^2
01100 0 0001001110 0	102:	`S78S`	`A`-=$m[78]$	Subtract the previous value
00111 0 0001001111 0	103:	`T79S`	$m[79]$=`A`; `ABC`=0	and store the new Δx^2
11100 0 0001001101 0	104:	`A77S`	`A`+=$m[77]$	Update variable holding
00111 0 0001001110 0	105:	`U78S`	$m[78]$=`A`	the previous x^2
00101 0 0000110000 0	106:	`T48S`	$m[48]$=`A`; `ABC`=0	Put x^2
				in $m[48]$ for printing
11100 0 0001010000 0	107:	`A80S`	`A`+=$m[80]$	Put return jump `E110S`
00101 0 0001001011 0	108:	`T75S`	$m[75]$=`A`; `ABC`=0	into $m[75]$
00011 0 0000110001 0	109:	`E49S`	`goto 49`	Enter the print subroutine
01001 0 0000101011 0	110:	`D43S`	`wr`($m[43]$)	Write a space
01001 0 0000101011 0	111:	`D43S`	`wr`($m[43]$)	Write a space
11100 0 0001001111 0	112:	`A79S`	`A`+=$m[79]$	Get Δx^2
00101 0 0000110000 0	113:	`T48S`	$m[48]$=`A`; `ABC`=0	Put it in $m[48]$ for printing
11100 0 0001010001 0	114:	`A81S`	`A`+=$m[81]$	Put return jump `E118S`
00101 0 0001001011 0	115:	`T75S`	$m[75]$=`A`; `ABC`=0	into $m[75]$
00011 0 0000110001 0	116:	`E49S`	`goto 49`	Enter the print subroutine
11100 0 0000100011 0	117:	`A35S`	`A`+=$m[35]$	Order to place in $m[52]$
11100 0 0001001100 0	118:	`A76S`	`A`+=$m[76]$	Get x
01100 0 0001010010 0	119:	`S82S`	`A`-=$m[82]$	Subtract the end limit (=100)
11011 0 0001010101 0	120:	`G85S`	`if A<0 goto 85`	Repeat, if more to do
01001 0 0000101001 0	121:	`D41S`	`wr`($m[41]$)	Write figure shift
01101 0 0000000000 0	122:	`Z0S`	`Stop`	Stop the machine

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 brass plaque on this door holds the engraved names of those retired members of the Laboratory who used the door in its original location.

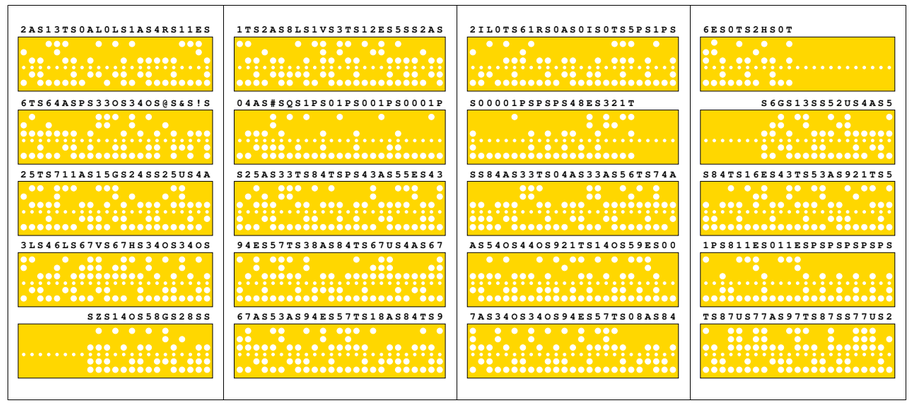
Links

`http://www.dcs.warwick.ac.uk/~edsac/`
This links to Martin Campbell-Kelly's excellent EDSAC simulator and related documents.

`http://www.cl.cam.ac.uk/UOCCL/misc/EDSAC99`
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/~mr/Edsac.html`
This links to a shell based EDSAC simulator that runs on Pentium based Linux systems. It was designed to be educational having a built-in interactive debugger allowing single step execution, the setting 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 also appears as a demonstration program in the Cintcode BCPL system (`http://www.cl.cam.ac.uk/~mr/BCPL.html`).

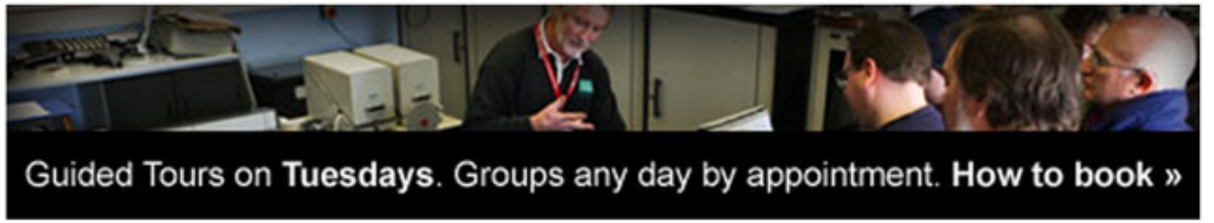
`http://www.cl.cam.ac.uk/~mr/edsacposter.pdf`
This is a PDF version of this poster on two A4 pages.

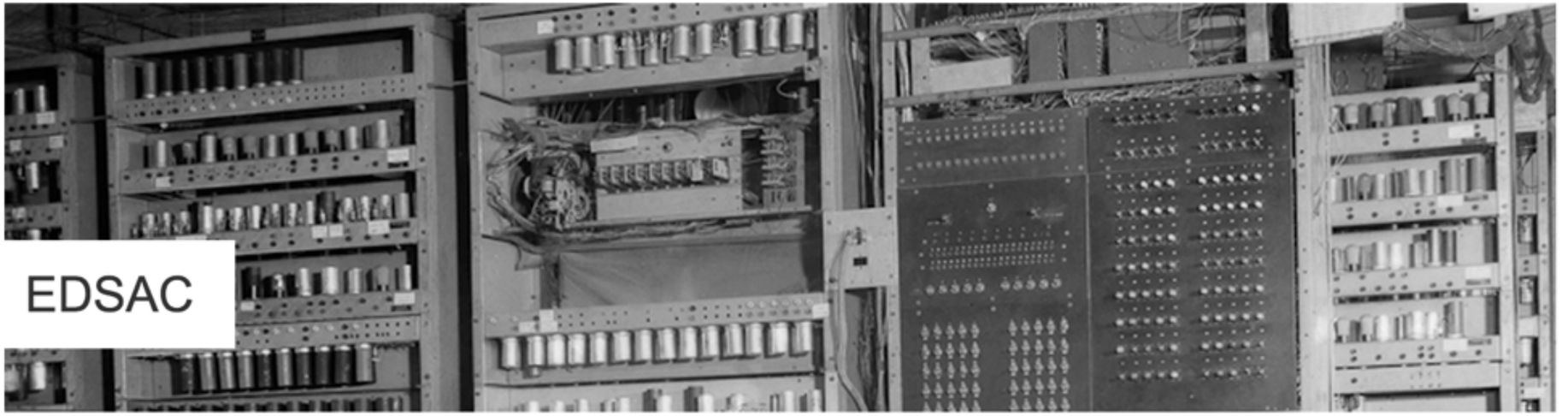


The corrected tape segments etched on the Tea Room glass panels

http://www.cl.cam.ac.uk/~mr10/edsacposter.pdf

```
AnS     Acc += Mem[n]
SnS     Acc -= Mem[n]
EnS     if Acc >= 0 goto n
GnS     if Acc < 0 goto n
LnS     leftshift
OnS     output
TnS     Mem[n] = Acc; Acc = 0
UnS     Mem[n] = Acc
XnS     No operation
ZnS     Stop the machine and ring a bell
```





EDSAC

The EDSAC Replica Project aims to reconstruct one of the most important early British digital computers.

Designed in 1947 by a team lead by Maurice Wilkes, the original EDSAC computer operated for almost 10 years, starting from its first successful program run on 6th May 1949, at the Cambridge University Mathematical Laboratory.

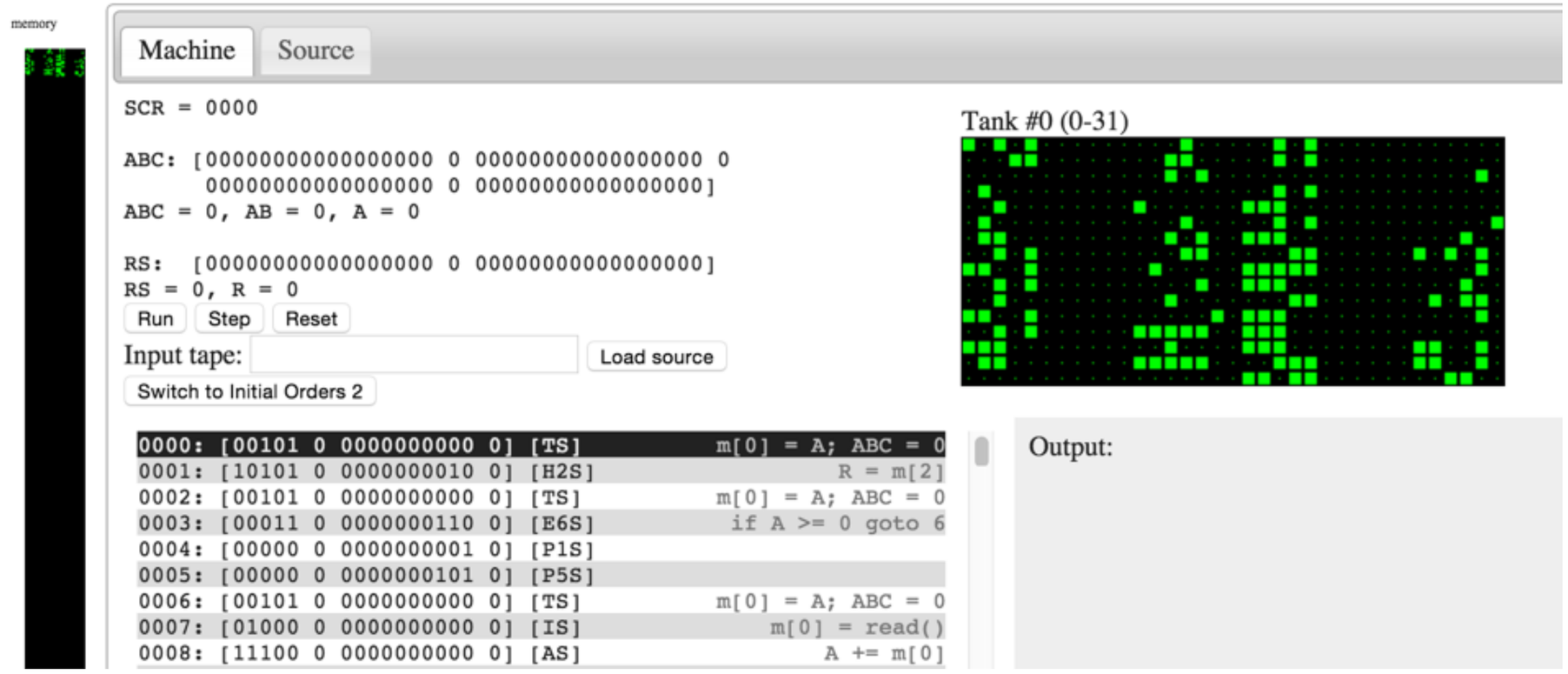
We are now initially targeting to have a working reconstruction of EDSAC as it was in 1951 (when it was in everyday service at the University of Cambridge Mathematical Laboratory) operational by late 2015.

The EDSAC Replica Project is a registered charity and is affiliated to the UK's Computer Conservation Society.

Please help the next stage of the EDSAC Project

http://www.tnmoc.org/special-projects/edsac

http://nhiro.org/learn_language/EDSAC-on-browser.html



“Hi” on the EDSAC / Initial Orders 1

```
T44S            31          T _end+1        mark end of program
E38S            32          E _start        jump to beginning of program
*S              33  lshift  *               letter shift
HS              34  _H      H               letter H
IS              35  _I      I               letter I
&S              36  lf      &               LF - line feed character
@S              37  cr      @               CR - carriage return character
O33S            38  _start  O lshift        prepare for printing lettersn
O34S            39          O _H            print H
O35S            40          O _I            print I
O36S            41          O lf            print lf
O37S            42          O cr            print cr
ZS              43  _end    Z               end of program
```

T44SE38S*SHSIS&S@SO33SO34SO35SO36SO37SZS

“FizzBuzz” on the EDSAC / Initial Orders 1

written in a “primitive” 1949-like style
by Olve Maudal, Monday, April 20, 2015

I pretended I was a student, who had won a **single** chance to run my program on this precious computer.

The program did actually ran on the very first attempt!

“FizzBuzz” on the EDSAC / Initial Orders 1

```
T123S   31              T L_end         mark end of program
E60S    32              E L_start       jump to the beginning of program
#S      33 _FS          #               figure shift
*S      34 _LS          *               letter shift
&S      35 _LF          &               linefeed character
@S      36 _CR          @               carriage return character
P100S   37 _100         P 100           constant 100
P10S    38 _10          P 10            constant 10
P5S     39 _5           P 5             constant 5
P3S     40 _3           P 3             constant 3
P1S     41 _1           P 1             constant 1
QS      42 _'1'         Q               constant figure 1
PS      43 _'0'         P               constant figure 0
BS      44 _B           B               constant letter B
FS      45 _F           F               constant letter F
IS      46 _I           I               constant letter I
US      47 _U           U               constant letter U
ZS      48 _Z           Z               constant letter Z
PS      49 _dummy       P               used to flush and reset the accumulator
P1S     50 _cnt         P 1             counter, current number to be considered, will be increased
PS      51 _num         P               number to be printed, negative if counter is mod 3 or mod 5
PS      52 _d           P               digit to be printed
O34S    53 L_next       O _LS           output LS, prepare for printing letters
O35S    54              O _LF           output LF, linefeed
O36S    55              O _CR           output CR, carriage return
T49S    56              T _dummy        reset Acc
A50S    57              A _cnt          load Acc with _cnt
A41S    58              A _1            increase Acc
T50S    59              T _cnt          store Acc into _cnt, reset Acc
A50S    60 L_start      A _cnt          load Acc with _cnt (we know that Acc initially is 0)
U51S    61              U _num          tentatively set number to be printed
S40S    62 L_tryFizz    S _3            subtract 3
E62S    63              E L_tryFizz     loop until Acc < 0
A40S    64              A _3            add 3, restore previous value
S41S    65              S _1            subtract 1, to check if Acc was 0
E73S    66              E L_notFizz     jump if Acc was not 0, ie number was not divisable by 3
T51S    67              T _num          set _num to negative value, flag that no value should be printed
O34S    68              O _LS           prepare printing letters
O45S    69              O _F            output F
O46S    70              O _I            output I
O48S    71              O _Z            output Z
O48S    72              O _Z            output Z
T49S    73 L_notFizz    T _dummy        reset Acc
A50S    74              A _cnt          load Acc with _cnt
S39S    75 L_Buzz       S _5            subtract 5
E75S    76              E L_Buzz        loop until Acc < 0
A39S    77              A _5            add 5, restore previous value
S41S    78              S _1            subtract 1, to check if Acc was 0
E86S    79              E L_notBuzz     jump if Acc was not 0, ie number was not divisable by 5
T51S    80              T _num          set _num to negative value, flag that no value should be printed
O34S    81              O _LS           prepare printing letters
O44S    82              O _B            output B
O47S    83              O _U            output U
O48S    84              O _Z            output Z
O48S    85              O _Z            output Z
T49S    86 L_notBuzz    T _dummy        reset Acc
A51S    87              A _num          load _num to check number to be printed
G53S    88              G L_next        goto next iteration if _num is negative
O33S    89 L_printNum   O _FS           prepare for printing numbers
T49S    90              T _dummy        reset Acc
A50S    91              A _cnt          load counter
S37S    92              S _100          subtract 100, check if we should stop
G98S    93              G L_not100      jump if not 100 yet
O42S    94              O _'1'          output 1
O43S    95              O _'0'          output 0
O43S    96              O _'0'          output 0
ZS      97              Z               end the program
T49S    98 L_not100     T _dummy        reset Acc
T52S    99              T _d            reset digit
A50S   100              A _cnt          load counter
S38S   101 L_count10s   S _10          subtract 10
G109S  102              G L_print10s    goto print 10s if Acc < 0
T51S   103              T _num          store number
A52S   104              A _d            load digit
A41S   105              A _1            increase digit
T52S   106              T _d            store digit
A51S   107              A _num          load number
E101S  108              E L_count10s    loop unconditionally
T49S   109 L_print10s   T _dummy        reset Acc
A52S   110              A _d            load digit
S41S   111              S _1            decrease digit by 1
G117S  112              G L_1           if negative (digit was 0), skip printing of tens digits
A41S   113              A _1            restore digit, by increasing with 1
L512S  114              L 2^(11-2)      Acc << 11, create a printable figure
T52S   115              T _d            save printable figure
O52S   116              O _d            print figure / digit
T49S   117 L_1:         T _dummy        reset Acc
A51S   118              A _num          load number
L512S  119              L 2^(11-2)      Acc << 11, create a printable figure
T52S   120              T _d            save printable figure
O52S   121              O _d            print figure / digit
E53S   122              E L_next        unconditional jump
XS     123 L_end        X
```


```
T123S   31              T L_end         mark end of program
E60S    32              E L_start       jump to the beginning of program
#S      33 _FS          #               figure shift
*S      34 _LS          *               letter shift
&S      35 _LF          &               linefeed character
@S      36 _CR          @               carriage return character
P100S   37 _100         P 100           constant 100
P10S    38 _10          P 10            constant 10
P5S     39 _5           P 5             constant 5
P3S     40 _3           P 3             constant 3
P1S     41 _1           P 1             constant 1
QS      42 _'1'         Q               constant figure 1
PS      43 _'0'         P               constant figure 0
BS      44 _B           B               constant letter B
FS      45 _F           F               constant letter F
IS      46 _I           I               constant letter I
US      47 _U           U               constant letter U
ZS      48 _Z           Z               constant letter Z
PS      49 _dummy       P               used to flush and reset the accumulator
P1S     50 _cnt         P 1             counter, current number to be considered, will be increased
PS      51 _num         P               number to be printed, negative if counter is mod 3 or mod 5
PS      52 _d           P               digit to be printed
```

“FizzBuzz” on the EDSAC / Initial Orders 1

```
T123S   31              T L_end         mark end of program
E60S    32              E L_start       jump to the beginning of program
#S      33 _FS          #               figure shift
*S      34 _LS          *               letter shift
&S      35 _LF          &               linefeed character
@S      36 _CR          @               carriage return character
P100S   37 _100         P 100           constant 100
P10S    38 _10          P 10            constant 10
P5S     39 _5           P 5             constant 5
P3S     40 _3           P 3             constant 3
P1S     41 _1           P 1             constant 1
QS      42 _'1'         Q               constant figure 1
PS      43 _'0'         P               constant figure 0
BS      44 _B           B               constant letter B
FS      45 _F           F               constant letter F
IS      46 _I           I               constant letter I
US      47 _U           U               constant letter U
ZS      48 _Z           Z               constant letter Z
PS      49 _dummy       P               used to flush and reset the accumulator
P1S     50 _cnt         P 1             counter, current number to be considered, will be increased
PS      51 _num         P               number to be printed, negative if counter is mod 3 or mod 5
PS      52 _d           P               digit to be printed
O34S    53 L_next       O _LS           output LS, prepare for printing letters
O35S    54              O _LF           output LF, linefeed
O36S    55              O _CR           output CR, carriage return
T49S    56              T _dummy        reset Acc
A50S    57              A _cnt          load Acc with _cnt
A41S    58              A _1            increase Acc
T50S    59              T _cnt          store Acc into _cnt, reset Acc
A50S    60 L_start      A _cnt          load Acc with _cnt (we know that Acc initially is 0)
U51S    61              U _num          tentatively set number to be printed
S40S    62 L_tryFizz    S _3            subtract 3
E62S    63              E L_tryFizz     loop until Acc < 0
A40S    64              A _3            add 3, restore previous value
S41S    65              S _1            subtract 1, to check if Acc was 0
E73S    66              E L_notFizz     jump if Acc was not 0, ie number was not divisable by 3
T51S    67              T _num          set _num to negative value, flag that no value should be printed
O34S    68              O _LS           prepare printing letters
O45S    69              O _F            output F
O46S    70              O _I            output I
O48S    71              O _Z            output Z
O48S    72              O _Z            output Z
T49S    73 L_notFizz    T _dummy        reset Acc
A50S    74              A _cnt          load Acc with _cnt
S39S    75 L_Buzz       S _5            subtract 5
E75S    76              E L_Buzz        loop until Acc < 0
A39S    77              A _5            add 5, restore previous value
S41S    78              S _1            subtract 1, to check if Acc was 0
E86S    79              E L_notBuzz     jump if Acc was not 0, ie number was not divisable by 5
T51S    80              T _num          set _num to negative value, flag that no value should be printed
O34S    81              O _LS           prepare printing letters
O44S    82              O _B            output B
O47S    83              O _U            output U
O48S    84              O _Z            output Z
O48S    85              O _Z            output Z
T49S    86 L_notBuzz    T _dummy        reset Acc
A51S    87              A _num          load _num to check number to be printed
G53S    88              G L_next        goto next iteration if _num is negative
O33S    89 L_printNum   O _FS           prepare for printing numbers
T49S    90              T _dummy        reset Acc
A50S    91              A _cnt          load counter
S37S    92              S _100          subtract 100, check if we should stop
G98S    93              G L_not100      jump if not 100 yet
O42S    94              O _'1'          output 1
O43S    95              O _'0'          output 0
O43S    96              O _'0'          output 0
ZS      97              Z               end the program
T49S    98 L_not100     T _dummy        reset Acc
T52S    99              T _d            reset digit
A50S   100              A _cnt          load counter
S38S   101 L_count10s   S _10          subtract 10
G109S  102              G L_print10s    goto print 10s if Acc < 0
T51S   103              T _num          store number
A52S   104              A _d            load digit
A41S   105              A _1            increase digit
T52S   106              T _d            store digit
A51S   107              A _num          load number
E101S  108              E L_count10s    loop unconditionally
T49S   109 L_print10s   T _dummy        reset Acc
A52S   110              A _d            load digit
S41S   111              S _1            decrease digit by 1
G117S  112              G L_1           if negative (digit was 0), skip printing of tens digits
A41S   113              A _1            restore digit, by increasing with 1
L512S  114              L 2^(11-2)      Acc << 11, create a printable figure
T52S   115              T _d            save printable figure
O52S   116              O _d            print figure / digit
T49S   117 L_1:         T _dummy        reset Acc
A51S   118              A _num          load number
L512S  119              L 2^(11-2)      Acc << 11, create a printable figure
T52S   120              T _d            save printable figure
O52S   121              O _d            print figure / digit
E53S   122              E L_next        unconditional jump
XS     123 L_end        X
```

```
O34S    53 L_next       O _LS           output LS, prepare for printing letters
O35S    54              O _LF           output LF, linefeed
O36S    55              O _CR           output CR, carriage return
T49S    56              T _dummy        reset Acc
A50S    57              A _cnt          load Acc with _cnt
A41S    58              A _1            increase Acc
T50S    59              T _cnt          store Acc into _cnt, reset Acc
A50S    60 L_start      A _cnt          load Acc with _cnt (we know that Acc initially is 0)
U51S    61              U _num          tentatively set number to be printed
S40S    62 L_tryFizz    S _3            subtract 3
E62S    63              E L_tryFizz     loop until Acc < 0
A40S    64              A _3            add 3, restore previous value
S41S    65              S _1            subtract 1, to check if Acc was 0
E73S    66              E L_notFizz     jump if Acc was not 0, ie number was not divisable by 3
T51S    67              T _num          set _num to negative value, flag that no value should be printed
O34S    68              O _LS           prepare printing letters
O45S    69              O _F            output F
O46S    70              O _I            output I
O48S    71              O _Z            output Z
O48S    72              O _Z            output Z
```

“FizzBuzz” on the EDSAC / Initial Orders 1

```
T123S   31              T L_end         mark end of program
E60S    32              E L_start       jump to the beginning of program
#S      33 _FS          #               figure shift
*S      34 _LS          *               letter shift
&S      35 _LF          &               linefeed character
@S      36 _CR          @               carriage return character
P100S   37 _100         P 100           constant 100
P10S    38 _10          P 10            constant 10
P5S     39 _5           P 5             constant 5
P3S     40 _3           P 3             constant 3
P1S     41 _1           P 1             constant 1
QS      42 _'1'         Q               constant figure 1
PS      43 _'0'         P               constant figure 0
BS      44 _B           B               constant letter B
FS      45 _F           F               constant letter F
IS      46 _I           I               constant letter I
US      47 _U           U               constant letter U
ZS      48 _Z           Z               constant letter Z
PS      49 _dummy       P               used to flush and reset the accumulator
P1S     50 _cnt         P 1             counter, current number to be considered, will be increased
PS      51 _num         P               number to be printed, negative if counter is mod 3 or mod 5
PS      52 _d           P               digit to be printed
O34S    53 L_next       O _LS           output LS, prepare for printing letters
O35S    54              O _LF           output LF, linefeed
O36S    55              O _CR           output CR, carriage return
T49S    56              T _dummy        reset Acc
A50S    57              A _cnt          load Acc with _cnt
A41S    58              A _1            increase Acc
T50S    59              T _cnt          store Acc into _cnt, reset Acc
A50S    60 L_start      A _cnt          load Acc with _cnt (we know that Acc initially is 0)
U51S    61              U _num          tentatively set number to be printed
S40S    62 L_tryFizz    S _3            subtract 3
E62S    63              E L_tryFizz     loop until Acc < 0
A40S    64              A _3            add 3, restore previous value
S41S    65              S _1            subtract 1, to check if Acc was 0
E73S    66              E L_notFizz     jump if Acc was not 0, ie number was not divisable by 3
T51S    67              T _num          set _num to negative value, flag that no value should be printed
O34S    68              O _LS           prepare printing letters
O45S    69              O _F            output F
O46S    70              O _I            output I
O48S    71              O _Z            output Z
O48S    72              O _Z            output Z
T49S    73 L_notFizz    T _dummy        reset Acc
A50S    74              A _cnt          load Acc with _cnt
S39S    75 L_Buzz       S _5            subtract 5
E75S    76              E L_Buzz        loop until Acc < 0
A39S    77              A _5            add 5, restore previous value
S41S    78              S _1            subtract 1, to check if Acc was 0
E86S    79              E L_notBuzz     jump if Acc was not 0, ie number was not divisable by 5
T51S    80              T _num          set _num to negative value, flag that no value should be printed
O34S    81              O _LS           prepare printing letters
O44S    82              O _B            output B
O47S    83              O _U            output U
O48S    84              O _Z            output Z
O48S    85              O _Z            output Z
T49S    86 L_notBuzz    T _dummy        reset Acc
A51S    87              A _num          load _num to check number to be printed
G53S    88              G L_next        goto next iteration if _num is negative
O33S    89 L_printNum   O _FS           prepare for printing numbers
T49S    90              T _dummy        reset Acc
A50S    91              A _cnt          load counter
S37S    92              S _100          subtract 100, check if we should stop
G98S    93              G L_not100      jump if not 100 yet
O42S    94              O _'1'          output 1
O43S    95              O _'0'          output 0
O43S    96              O _'0'          output 0
ZS      97              Z               end the program
T49S    98 L_not100     T _dummy        reset Acc
T52S    99              T _d            reset digit
A50S   100              A _cnt          load counter
S38S   101 L_count10s   S _10          subtract 10
G109S  102              G L_print10s    goto print 10s if Acc < 0
T51S   103              T _num          store number
A52S   104              A _d            load digit
A41S   105              A _1            increase digit
T52S   106              T _d            store digit
A51S   107              A _num          load number
E101S  108              E L_count10s    loop unconditionally
T49S   109 L_print10s   T _dummy        reset Acc
A52S   110              A _d            load digit
S41S   111              S _1            decrease digit by 1
G117S  112              G L_1           if negative (digit was 0), skip printing of tens digits
A41S   113              A _1            restore digit, by increasing with 1
L512S  114              L 2^(11-2)      Acc << 11, create a printable figure
T52S   115              T _d            save printable figure
O52S   116              O _d            print figure / digit
T49S   117 L_1:         T _dummy        reset Acc
A51S   118              A _num          load number
L512S  119              L 2^(11-2)      Acc << 11, create a printable figure
T52S   120              T _d            save printable figure
O52S   121              O _d            print figure / digit
E53S   122              E L_next        unconditional jump
XS     123 L_end        X
```

```
T49S    73 L_notFizz    T _dummy        reset Acc
A50S    74              A _cnt          load Acc with _cnt
S39S    75 L_Buzz       S _5            subtract 5
E75S    76              E L_Buzz        loop until Acc < 0
A39S    77              A _5            add 5, restore previous value
S41S    78              S _1            subtract 1, to check if Acc was 0
E86S    79              E L_notBuzz     jump if Acc was not 0, ie number was not divisable by 5
T51S    80              T _num          set _num to negative value, flag that no value should be printed
O34S    81              O _LS           prepare printing letters
O44S    82              O _B            output B
O47S    83              O _U            output U
O48S    84              O _Z            output Z
O48S    85              O _Z            output Z
T49S    86 L_notBuzz    T _dummy        reset Acc
A51S    87              A _num          load _num to check number to be printed
G53S    88              G L_next        goto next iteration if _num is negative
O33S    89 L_printNum   O _FS           prepare for printing numbers
T49S    90              T _dummy        reset Acc
A50S    91              A _cnt          load counter
S37S    92              S _100          subtract 100, check if we should stop
G98S    93              G L_not100      jump if not 100 yet
O42S    94              O _'1'          output 1
O43S    95              O _'0'          output 0
O43S    96              O _'0'          output 0
ZS      97              Z               end the program
```

“FizzBuzz” on the EDSAC / Initial Orders 1

```
T123S   31              T L_end         mark end of program
E60S    32              E L_start       jump to the beginning of program
#S      33 _FS          #               figure shift
*S      34 _LS          *               letter shift
&S      35 _LF          &               linefeed character
@S      36 _CR          @               carriage return character
P100S   37 _100         P 100           constant 100
P10S    38 _10          P 10            constant 10
P5S     39 _5           P 5             constant 5
P3S     40 _3           P 3             constant 3
P1S     41 _1           P 1             constant 1
QS      42 _'1'         Q               constant figure 1
PS      43 _'0'         P               constant figure 0
BS      44 _B           B               constant letter B
FS      45 _F           F               constant letter F
IS      46 _I           I               constant letter I
US      47 _U           U               constant letter U
ZS      48 _Z           Z               constant letter Z
PS      49 _dummy       P               used to flush and reset the accumulator
P1S     50 _cnt         P 1             counter, current number to be considered, will be increased
PS      51 _num         P               number to be printed, negative if counter is mod 3 or mod 5
PS      52 _d           P               digit to be printed
O34S    53 L_next       O _LS           output LS, prepare for printing letters
O35S    54              O _LF           output LF, linefeed
O36S    55              O _CR           output CR, carriage return
T49S    56              T _dummy        reset Acc
A50S    57              A _cnt          load Acc with _cnt
A41S    58              A _1            increase Acc
T50S    59              T _cnt          store Acc into _cnt, reset Acc
A50S    60 L_start      A _cnt          load Acc with _cnt (we know that Acc initially is 0)
U51S    61              U _num          tentatively set number to be printed
S40S    62 L_tryFizz    S _3            subtract 3
E62S    63              E L_tryFizz     loop until Acc < 0
A40S    64              A _3            add 3, restore previous value
S41S    65              S _1            subtract 1, to check if Acc was 0
E73S    66              E L_notFizz     jump if Acc was not 0, ie number was not divisable by 3
T51S    67              T _num          set _num to negative value, flag that no value should be printed
O34S    68              O _LS           prepare printing letters
O45S    69              O _F            output F
O46S    70              O _I            output I
O48S    71              O _Z            output Z
O48S    72              O _Z            output Z
T49S    73 L_notFizz    T _dummy        reset Acc
A50S    74              A _cnt          load Acc with _cnt
S39S    75 L_Buzz       S _5            subtract 5
E75S    76              E L_Buzz        loop until Acc < 0
A39S    77              A _5            add 5, restore previous value
S41S    78              S _1            subtract 1, to check if Acc was 0
E86S    79              E L_notBuzz     jump if Acc was not 0, ie number was not divisable by 5
T51S    80              T _num          set _num to negative value, flag that no value should be printed
O34S    81              O _LS           prepare printing letters
O44S    82              O _B            output B
O47S    83              O _U            output U
O48S    84              O _Z            output Z
O48S    85              O _Z            output Z
T49S    86 L_notBuzz    T _dummy        reset Acc
A51S    87              A _num          load _num to check number to be printed
G53S    88              G L_next        goto next iteration if _num is negative
O33S    89 L_printNum   O _FS           prepare for printing numbers
T49S    90              T _dummy        reset Acc
A50S    91              A _cnt          load counter
S37S    92              S _100          subtract 100, check if we should stop
G98S    93              G L_not100      jump if not 100 yet
O42S    94              O _'1'          output 1
O43S    95              O _'0'          output 0
O43S    96              O _'0'          output 0
ZS      97              Z               end the program
T49S    98 L_not100     T _dummy        reset Acc
T52S    99              T _d            reset digit
A50S   100              A _cnt          load counter
S38S   101 L_count10s   S _10          subtract 10
G109S  102              G L_print10s    goto print 10s if Acc < 0
T51S   103              T _num          store number
A52S   104              A _d            load digit
A41S   105              A _1            increase digit
T52S   106              T _d            store digit
A51S   107              A _num          load number
E101S  108              E L_count10s    loop unconditionally
T49S   109 L_print10s   T _dummy        reset Acc
A52S   110              A _d            load digit
S41S   111              S _1            decrease digit by 1
G117S  112              G L_1           if negative (digit was 0), skip printing of tens digits
A41S   113              A _1            restore digit, by increasing with 1
L512S  114              L 2^(11-2)      Acc << 11, create a printable figure
T52S   115              T _d            save printable figure
O52S   116              O _d            print figure / digit
T49S   117 L_1:         T _dummy        reset Acc
A51S   118              A _num          load number
L512S  119              L 2^(11-2)      Acc << 11, create a printable figure
T52S   120              T _d            save printable figure
O52S   121              O _d            print figure / digit
E53S   122              E L_next        unconditional jump
XS     123 L_end        X
```

```
T49S    98 L_not100     T _dummy        reset Acc
T52S    99              T _d            reset digit
A50S   100              A _cnt          load counter
S38S   101 L_count10s   S _10          subtract 10
G109S  102              G L_print10s    goto print 10s if Acc < 0
T51S   103              T _num          store number
A52S   104              A _d            load digit
A41S   105              A _1            increase digit
T52S   106              T _d            store digit
A51S   107              A _num          load number
E101S  108              E L_count10s    loop unconditionally
T49S   109 L_print10s   T _dummy        reset Acc
A52S   110              A _d            load digit
S41S   111              S _1            decrease digit by 1
G117S  112              G L_1           if negative (digit was 0), skip printing of tens digits
A41S   113              A _1            restore digit, by increasing with 1
L512S  114              L 2^(11-2)      Acc << 11, create a printable figure
T52S   115              T _d            save printable figure
O52S   116              O _d            print figure / digit
T49S   117 L_1:         T _dummy        reset Acc
A51S   118              A _num          load number
L512S  119              L 2^(11-2)      Acc << 11, create a printable figure
T52S   120              T _d            save printable figure
O52S   121              O _d            print figure / digit
E53S   122              E L_next        unconditional jump
XS     123 L_end        X
```

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SG98SO42SO43S
O43SZST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

!

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SG98SO42SO43S
O43SZST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

“FizzBuzz” on the EDSAC / Initial Orders 1

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SG98SO42SO43S
O43SZST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

There is a small bug in the program. Did you notice?

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SG98SO42SO43S
O43SZST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SG98SO42SO43S
O43SZST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Here is a quick and dirty fix!

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SA41SG98SZSO4
3SO43ST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SA41SG98SZSO4
3SO43ST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html

“FizzBuzz” on the EDSAC / Initial Orders I

```
T123SE60S#S*S&S@SP100SP10SP5SP3SP1SQSPSBSFSISU
SZSPSP1SPSPSO34SO35SO36ST49SA50SA41ST50SA50SU5
1SS40SE62SA40SS41SE73ST51SO34SO45SO46SO48SO48S
T49SA50SS39SE75SA39SS41SE86ST51SO34SO44SO47SO4
8SO48ST49SA51SG53SO33ST49SA50SS37SA41SG98SZSO4
3SO43ST49ST52SA50SS38SG109ST51SA52SA41ST52SA51
SE101ST49SA52SS41SG117SA41SL512ST52SO52ST49SA5
1SL512ST52SO52SE53SXS
```

Enjoy!

Try this program on NISHIO Hirokazu’s EDSAC Simulator
http://nhiro.org/learn_language/repos/EDSAC-on-browser/index.html