Implementation of LIFE (in 32 bit)

The unicellular organisms of Life live, die, and are reborn on a two-dimensional grid, according to simple rules. The future state of a cell is simply influenced by its current state and that of its neighbors. Each cell has exactly eight "neighbors," which are eight other adjacent cells: two vertically, two horizontally, and four at the corners.
To determine the future of a cell, just count how many of its neighbors are alive, and see what state the cell is currently in. 1) If it is alive and has two or three living neighbors, it remains alive; otherwise, it dies. 2) If it is dead and has exactly three living neighbors, it is reborn; otherwise, it remains dead.
Therefore, we can state that a cell with fewer than two living neighbors dies from isolation, and if it has more than three, it dies from overpopulation.

On the TI-66, it is possible to implement LIFE by using the PC-200 printer with the limitations of the case. The method used is to represent the cellular grid through 4 numbers (< 1) with 8 decimal digits that can take the values 0 or 1. This way, we obtain the representation of a grid of 4 lines by 8 columns in which a 0 (zero) indicates a dead cell and a 1 (one) indicates a living cell.

Prog LIFE

Before running the program ([A])we set the initial state of the 4 x 8 grid (32 cells) represented by the temporary registers.
For example:

```
r7  = .01010001
r8  = .10000010
r9  = .10000100
r10 = .00100010
```

The following auxiliary registers are also initialized
r4 = 1, which represents the column counter [1…8]
r6 = 0, which is used to count the evolutionary ERAs (before extinction or a stall situation).
Also, it is necessary to SET [2nd][FIX]8 so that all 8 digits are always displayed (including the final zeros).
In the absence of a printer, set FLAG 00. Doing so tells the program to stop at each cycle (ERA). In that case, an 8-digit decimal number is displayed, representing the state of the grid. In particular, each digit indicates the number of living cells present in the column it represents.
For example, the number:

0,02003000	**indicates that in the 2nd column there are 2 living cells and in the 5th column there are 3 living cells.**

This representation somehow allows us to follow the evolution of the cellular automaton even if we do not have a printer.

At the beginning of the ERA, the temporary registers are saved in the state registers (r0, r1, r2, r3) which remain unchanged throughout the evolutionary cycle of the current ERA.

In case of 0.00000000, we stop because this indicates the occurrence of one of two events:

1. **EXTINCTION: in this case, all cells are dead, and the temporary registers are zeroed.**

2. STALL: in this case, the values of the state registers are respectively equal to the relative temporary registers (the grid values no longer change and remain equal to those of the previous ERA).

"At the beginning of the ERA, the current state of the cells is printed (if the printer is present), and the grid registers are zeroed. It is also checked that we are not in the presence of a STALL."
"If I=1 (start of ERA), print the state registers r0, r1, r2, r3
"after loading them with the values contained in the temporary registers r7, r8, r9, "r10 which are zeroed.

```
 0.  [LBL]
 1.  A
 2.  [RCL]
 3.  00
 4.  -
 5.  [RCL]         "r7 → r0
 6.  07
 7.  [STO]
 8.  00
 9.  [2nd][Prt]
10.  +
11.  [RCL]
12.  01
13.  -
14.  [RCL]         "r8 → r1
15.  08
16.  [STO]
17.  01
18.  [2nd][Prt]
19.  +
20.  [RCL]
21.  02
22.  -
23.  [RCL]         "r9 → r2
24.  09
25.  [STO]
26.  02
27.  [2nd][Prt]
28.  +
29.  [RCL]
30.  03
31.  -
32.  [RCL]         "r10 → r3
33.  10
34.  [STO]
35.  03
36.  [2nd][Prt]
37.  [2nd][Adv]
38.  =
39.  [x=t]
40.  C             "→ STALL, I stop (the cellular automaton does not vary anymore)
                   "and 0.00000000 is displayed
41.  0             "0 → r7, r8, r9, r10
42.  [STO]
43.  07
44.  [STO]
45.  08
46.  [STO]
47.  09
48.  [STO]
```

```
   49. 10
   50. [OP]
   51. 26            "Zero temporary registers and increment number of ERAs in r6
"Calculate the number of living cells around the cell under examination
   52. [LBL]
   53. B
   54. [RCL]
   55. 00
   56. C'            "→ r0[I] = INT(Frac(INT(r0*10^(I))÷2)+0.5)
   57. [STO]
   58. 11            "r0[I]
   59. [STO]
   60. 18            "Nr1[I]
   61. [STO]
   62. 12            "Nr3[I]
   63. [RCL]
   64. 00
   65. D'            "→ r0[I+1]=INT(Frac(INT(r0*10^(I+1))÷2)+0.5)(se I=8→r0[0])
   66. [STO]
   67. 17            "Nr0[I]
   68. [SUM]
   69. 18            "Nr1[I]
   70. [SUM]
   71. 12            "Nr3[I]
   72. [RCL]
   73. 00
   74. E'            "→ r0[I-1]=INT(Frac(INT(r0*10^(I-1))÷2)+0.5)(se I=1 →r0[8])
   75. [SUM]
   76. 17            "Nr0[I]
   77. [SUM]
   78. 18            "Nr1[I]
   79. [SUM]
   80. 12            "Nr3[I]

   81. [RCL]
   82. 01
   83. C'            "→ r1[I] = INT(Frac(INT(r1*10^(I))÷2)+0.5)
   84. [STO]
   85. 14            "r1[I]
   86. [SUM]
   87. 17            "Nr0[I]
   88. [STO]
   89. 19            "Nr2[I]
   90. [RCL]
   91. 01
   92. D'            "→ r1[I+1] = INT(Frac(INT(r1*10^(I+1))÷2)+0.5)(se I=8 →r1[0])
   93. [SUM]
   94. 17            "Nr0[I]
   95. [SUM]
   96. 18            "Nr1[I]
   97. [SUM]
   98. 19            "Nr2[I]
   99. [RCL]
   100.   01
   101.   E'          "→ r1[I-1]=INT(Frac(INT(r1*10^(I-1))÷2)+0.5)(se I=1 →r1[8])
   102.   [SUM]
   103.   17          "Nr0[I]
   104.   [SUM]
   105.   18          "Nr1[I]
```

```
106.   [SUM]
107.   19         "Nr2[I]

108.   [RCL]
109.   02
110.   C'         "→ r2[I] = INT(Frac(INT(r2*10^(I))÷2)+0.5)
111.   [STO]
112.   15         "r2[I]
113.   [SUM]
114.   18         "Nr1[I]
115.   [SUM]
116.   12         "Nr3[I]
117.   [RCL]
118.   02
119.   D'         "→ r2[I+1] = INT(Frac(INT(r2*10^(I+1))÷2)+0.5)(se I=8 →r1[0])
120.   [SUM]
121.   18         "Nr1[I]
122.   [SUM]
123.   19         "Nr2[I]
124.   [SUM]
125.   12         "Nr3[I]
126.   [RCL]
127.   02
128.   E'         "→ r2[I-1]=INT(Frac(INT(r2*10^(I-1))÷2)+0.5)(se I=1 →r2[8])
129.   [SUM]
130.   18         "Nr1[I]
131.   [SUM]
132.   19         "Nr2[I]
133.   [SUM]
134.   12         "Nr3[I]

135.   [RCL]
136.   03
137.   C'         "→ r3[I] = INT(Frac(INT(r3*10^(I))÷2)+0.5)
138.   [STO]
139.   16         "r3[I]
140.   [SUM]
141.   17         "Nr0[I]
142.   [SUM]
143.   19         "Nr2[I]
144.   [RCL]
145.   03
146.   D'         "→ r3[I+1] = INT(Frac(INT(r3*10^(I+1))÷2)+0.5)(se I=8 →r3[0])
147.   [SUM]
148.   17         "Nr0[I]
149.   [SUM]
150.   19         "Nr2[I]
151.   [SUM]
152.   12         "Nr3[I]
153.   [RCL]
154.   03
155.   E'         "→ r3[I-1]=INT(Frac(INT(r3*10^(I-1))÷2)+0.5)(se I=1 →r3[8])
156.   [SUM]
157.   17         "Nr0[I]
158.   [SUM]
159.   19         "Nr2[I]
160.   [SUM]
161.   12         "Nr3[I]
```

```
"Update the cell state based on the number of living cells surrounding it
    162.    [RCL]       "r0[I]  → cell under examination
    163.    11
    164.    [STO]
    165.    05          "rX[I]"
    166.    [RCL]
    167.    07          "r7 → cells of the row"
    168.    [STO]
    169.    13          "rX"
    170.    [RCL]
    171.    17          "Nr0[I] → number of living cells around r0[I]"
    172.    A'
    173.    [RCL]
    174.    13
    175.    [STO]
    176.    07          "Update in r7 the state of the cell r7[I]

    177.    [RCL]
    178.    14          "r1[I] → cell under examination"
    179.    [STO]
    180.    05
    181.    [RCL]
    182.    08          "r8 → state of row cells"
    183.    [STO]
    184.    13
    185.    [RCL]
    186.    18          "Nr1[I] → number of living cells around r1[I]
    187.    A'
    188.    [RCL]
    189.    13
    190.    [STO]
    191.    08          "Update in r8 the state of the cell r8[I]

    192.    [RCL]
    193.    15          "r2[I]
    194.    [STO]
    195.    05
    196.    [RCL]
    197.    09          "r9
    198.    [STO]
    199.    13
    200.    [RCL]
    201.    19          "Nr2[I] → number of living cells around r2[I]"
    202.    A'
    203.    [RCL]
    204.    13
    205.    [STO]
    206.    09          "Update in r9 the state of the cell r9[I]

    207.    [RCL]
    208.    16          "r3[I]
    209.    [STO]
    210.    05
    211.    [RCL]
    212.    10          "r10
    213.    [STO]
    214.    13
    215.    [RCL]
    216.    12          "Nr3[I] → number of living cells around r3[I]
```

```
   217.   A'
   218.   [RCL]
   219.   13
   220.   [STO]
   221.   10         "Update in r10 the state of the cell r10[I]

"Loop until the state of all cells is updated
   222.   [OP]
   223.   24         "I=I+1 incremento il contatore"
   224.   8
   225.   -
   226.   [RCL]
   227.   04
   228.   =
   229.   [2nd][x>=t]
   230.   B          "if I is less than 9, LOOP"

"End of ERA: In case of extinction, STOP
   231.   1
   232.   [STO]
   233.   04         "I = 1 initialize the counter"

   234.   [RCL]
   235.   07
   236.   +
   237.   [RCL]
   238.   08
   239.   +
   240.   [RCL]
   241.   09
   242.   +
   243.   [RCL]
   244.   10
   245.   =
   246.   [x=t]      "if all cells are zero → EXTINCTION → STOP"
   247.   C


   248.   [IfF]      "If we have set FLAG 00 of PRINTER NOT PRESENT, display
   249.   00         "the SUM state of the grid and stop. → [R/S] to continue
   250.   C          "with a new ERA.
   251.   A          "Otherwise, the cycle restarts automatically, printing the
                     "grid and incrementing the number of ERAs

"*** ROUTINES ***"
   252.   [LBL]      "Update the value of the row
   253.   A'         "→ rX = rX + rX[I]*10^(-I)
   254.   -
   255.   3
   256.   )
   257.   [x=t]      "=>if NrX[I]=3 → rX[I]=1
   258.   02
   259.   67
   260.   +
   261.   1
   262.   )
   263.   [x=t]      "=>if NrX[I]=2 → rX[I] does not change
   264.   02
   265.   70
```

```
266.   RTN]        "=>if NrX[I]<>2 → rX[I]=0
267.   1
268.   [STO]
269.   05
270.   [RCL]
271.   04
272.   [+-]        "rX = rX + rX[I]*10^(-I)
273.   [INV]
274.   [log]
275.   *
276.   [RCL]
277.   05
278.   )
279.   [SUM]
280.   13
281.   [RTN]

282.   [LBL]
283.   B'          "INT(Frac(INT(X)÷2)+0.5)
284.   )
285.   [2nd][Intg]
286.   /
287.   2
288.   )
289.   [INV]
290.   [2nd][Intg]
291.   +
292.   .
293.   5
294.   )
295.   [2nd][Intg]
296.   [RTN]

297.   [LBL]
298.   C'          " => rX[I] = INT(Frac(INT(rX*10^(I))÷2)+0.5)
299.   *
300.   [RCL]
301.   04
302.   [INV]
303.   [log]
304.   B'
305.   [RTN]

306.   [LBL]
307.   D'          "=> rX[I+1]=INT(Frac(INT(rX*10^(I+1))÷2)+0.5) (se I=8 -> rX[0])
308.   *
309.   (
310.   [RCL]
311.   04
312.   -
313.   8
314.   )
315.   (
316.   [x=t]
317.   03
318.   21
319.   +
320.   8
321.   +
322.   1
323.   )
324.   [INV]
325.   [log]
326.   B'
327.   [RTN]

328.   [LBL]
329.   E'        => rX[I-1]=INT(Frac(INT(rX*10^(I-1))÷2)+0.5)(se I=1 →rX[8])
330.   *
331.   (
332.   [RCL]
333.   04
334.   -
335.   1
336.   )
337.   INV
338.   [x=t]
339.   03
340.   42
341.   8
342.   [INV]
343.   [log]
344.   B'
345.   [RTN]

346.   [LBL]
347.   C
348.   [R/S]
349.   A
```

"Requires 20 registers for 349 program steps (351 max).

- It takes about 70 sec per iteration; this means that to complete an ERA it takes about 9 minutes!