

## Sequência de Fibonacci

```

1.      MOV     R4, #10           ; elements number (1 - n)
2.      MOV     R0, #1000        ; memory offset
3.      MOV     R2, #1           ; tmp variable for
4.      MOV     R1, #1           ; init fibb. sequence
5.
6.      STR     R1, [R0]          ; storage first number serie
7.  FOR
8.      CMP     R2, R4            ; compare
9.      BEQ     ENDFOR           ; if(R2 == R4) break;
10.
11.     CMP     R2, #1            ; compare
12.     BNE     ELSE             ; to else if(R2 != 1)
13.     STR     R1, [R0, #4]      ; storage second number
14.     B       ENDFIF           ; finalized conditional
15.
16. ELSE
17.     SUBS     R3, R2, #2        ; indice = i - 2
18.     LDR     R1, [R0, R3, LSL #2] ; R1 = array[i-2]
19.     SUBS     R3, R2, #1        ; indice = i - 1
20.     LDR     R3, [R0, R3, LSL #2] ; R3 = array[i - 1]
21.     ADD     R12, R1, R3        ; add R1 + R3
22.     STR     R12, [R0, R2, LSL #2] ; storage value
23. ENDFIF
24.     ADD     R2, R2, #1        ; i = i + 1
25.     B       FOR              ; repeat loop
26. ENDFOR

```

<https://pastebin.com/FJg3Px6>

The screenshot displays an ARM emulator interface with the following components:

- Assembly Code (Left Panel):** Shows the Fibonacci sequence assembly code, including instructions like `MOV R4, #10`, `STR R1, [R0]`, `CMP R2, R4`, `BEQ ENDFOR`, `SUBS R3, R2, #2`, `LDR R1, [R0, R3, LSL #2]`, `ADD R12, R1, R3`, and `STR R12, [R0, R2, LSL #2]`.
- View Memory Contents (Center Panel):** A window showing memory addresses from `0x100` to `0x40C` with corresponding byte and word values. The word values are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.
- Register Values (Right Panel):** A table showing the current values of registers R0 through R15. R0 is `0x3E8`, R1 is `0x15`, R2 is `0xA`, R3 is `0x22`, R4 is `0xA`, R5 is `0x0`, R6 is `0x0`, R7 is `0x0`, R8 is `0x0`, R9 is `0x0`, R10 is `0x0`, R11 is `0x0`, R12 is `0x37`, R13 is `0xFF000000`, LR is `0x0`, and PC is `0x0`.
- Emulation Status (Top Bar):** Shows "Emulation Complete" and "Line Issues: 26, 0".
- Execution Controls (Top Right):** Buttons for "Execute", "Reset", "Step Backwards", and "Step Forwards".
- Bottom Panel:** Displays "Clock Cycles" and "Current Instruction: 0 Total: 175".

PG

```
1.      MOV    R0, #0x100 ; 20161ceca70326
2.      ORR    R0, R0, #0x46
3.
4.      MOV    R1, #1      ; increment for
5.      MOV    R4, #10     ; elements number
6.      MOV    R2, #1000   ; offset memory
7.
8.      STR    R0, [R2]    ; storage first number serie
9.
10.
11. FOR
12.      CMP    R1, R4
13.      BEQ    ENDFOR
14.
15.      LSL    R0, R0, #1
16.      STR    R0, [R2, R1, LSL #2]
17.
18.
19.      ADD    R1, R1, #1
20.      B      FOR
21. ENDFOR
```

<https://pastebin.com/PfUFJepg>

Professor não coloquei no código a função LDR, no exemplo acima não ia ser muito útil a utilização do mesmo.

The screenshot displays an ARM assembler emulator interface. The main window shows assembly code with line numbers 1 through 22. The code includes instructions like MOV, ORR, STR, CMP, BEQ, LSL, and B. A 'View Memory Contents' dialog box is open, showing a table of memory addresses and values. The table has columns for Word Address, Byte 3, Byte 2, Byte 1, Byte 0, and Word Value. The memory map on the right shows registers R0 through R13, LR, and PC, each with a value and format (Dec, Bin, Hex). The status bar at the bottom indicates 'Emulation Complete' and 'Clock Cycles'.

Word Address	Byte 3	Byte 2	Byte 1	Byte 0	Word Value
0x3E8	0x0	0x0	0x1	0x46	326
0x3EC	0x0	0x0	0x2	0x8C	652
0x3F0	0x0	0x0	0x5	0x18	1304
0x3F4	0x0	0x0	0xA	0x30	2608
0x3F8	0x0	0x0	0x14	0x60	5216
0x3FC	0x0	0x0	0x28	0xC0	10432
0x400	0x0	0x0	0x51	0x80	20864
0x404	0x0	0x0	0xA3	0x0	41728
0x408	0x0	0x1	0x46	0x0	83456
0x40C	0x0	0x2	0x8C	0x0	166912

Register	Value	Dec	Bin	Hex
R0	0x28C00			
R1	0xA			
R2	0x3E8			
R3	0x0			
R4	0xA			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			
R9	0x0			
R10	0x0			
R11	0x0			
R12	0x0			
R13	0xFF000000			
LR	0x0			
PC	0x3C			