A simple computer with nand gates and diodes

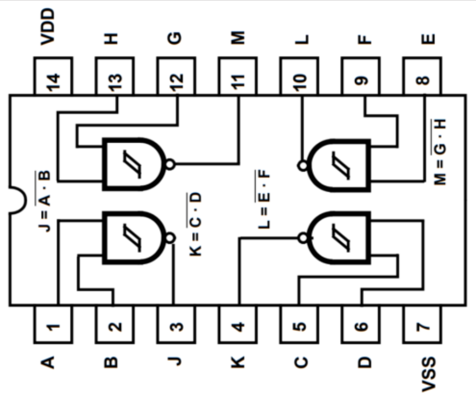
Features :

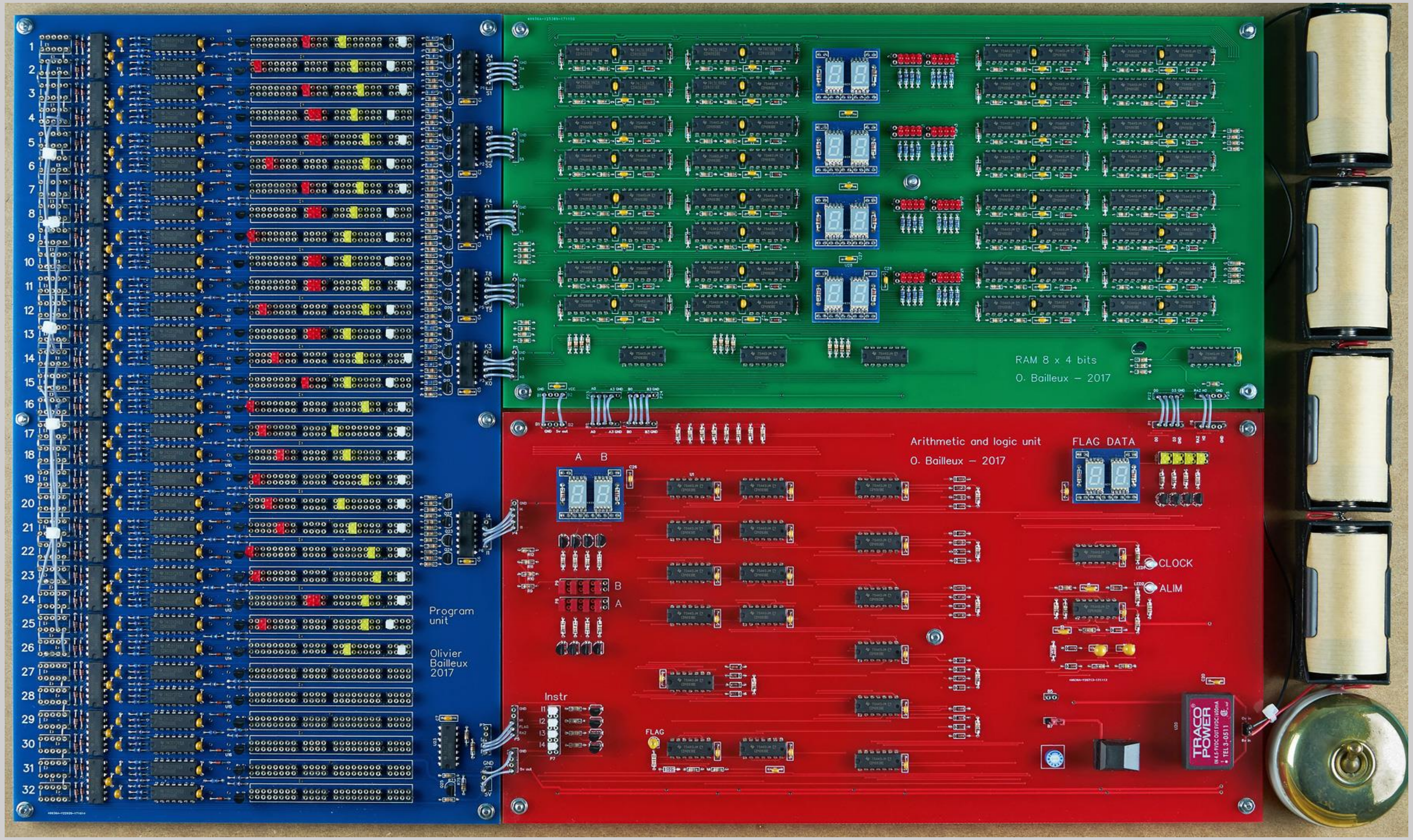
Harvard architecture

32 program lines

4 instructions (addition, nand, move, divide by 2)

8 x 4-bits registers = 32 bits memory

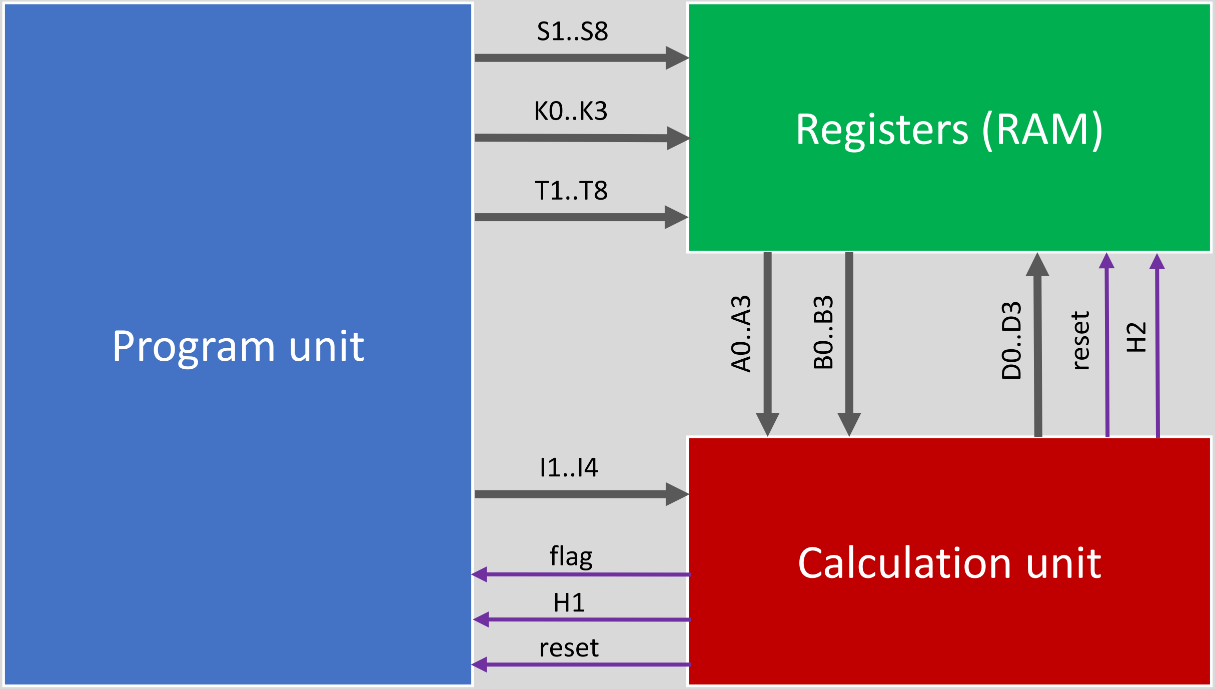

VDD
H
G
M
L
F
E
14
13
12
11
10
9
8
$J = \overline{A \cdot B}$
$K = \overline{C \cdot D}$
$L = \overline{E \cdot F}$
$M = \overline{G \cdot H}$
1
2
3
4
5
6
7
A
B
J
K
C
D
VSS



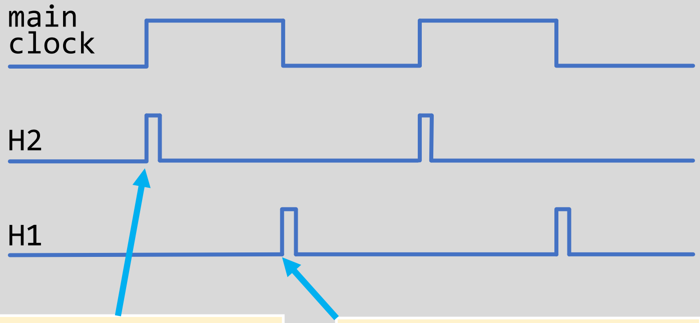
The only active components are **nand gates**, switching **diodes** and a few **transistors**.

(except for *optional* hexadecimal displays units that use microcontrollers to reduce their footprint)

Architecture and execution cycle



S1..S8	:	source register
K0..K3	:	source constant
T1..T8	:	target register
A0..A3	:	first operand
B0..B3	:	second operand
D0..D3	:	result
I1..I4	:	Instruction
H1	:	program clock
H2	:	write clock

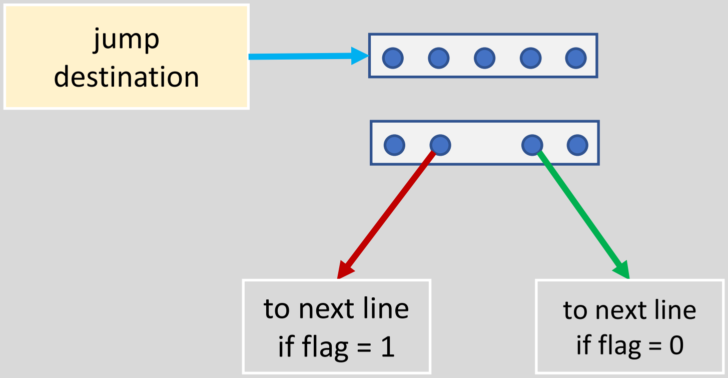


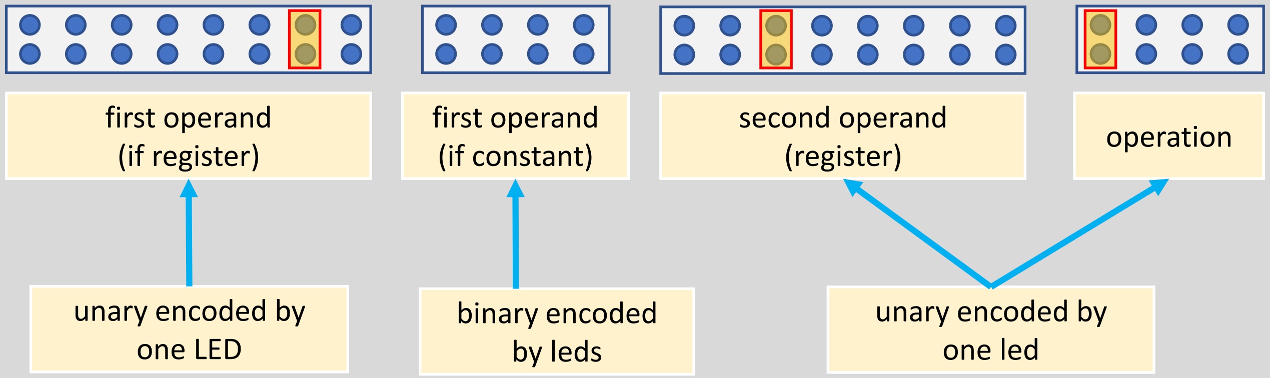
The calculation result is written into the target register

Jump to the next program line according to the flag value

Each program line, including an operation and a conditional jump, is executed in one clock cycle. In a first stage (main clock = 0), the calculation is done, then the H2 pulse writes the result into the target register. In a second stage (main clock = 1), The flag value is transmitted to the program unit, then the H1 pulse causes the jump to the next line, according to the flag value.

Detail of a program line




first operand (if register)
first operand (if constant)
second operand (register)
operation
unary encoded by one LED
binary encoded by leds
unary encoded by one led

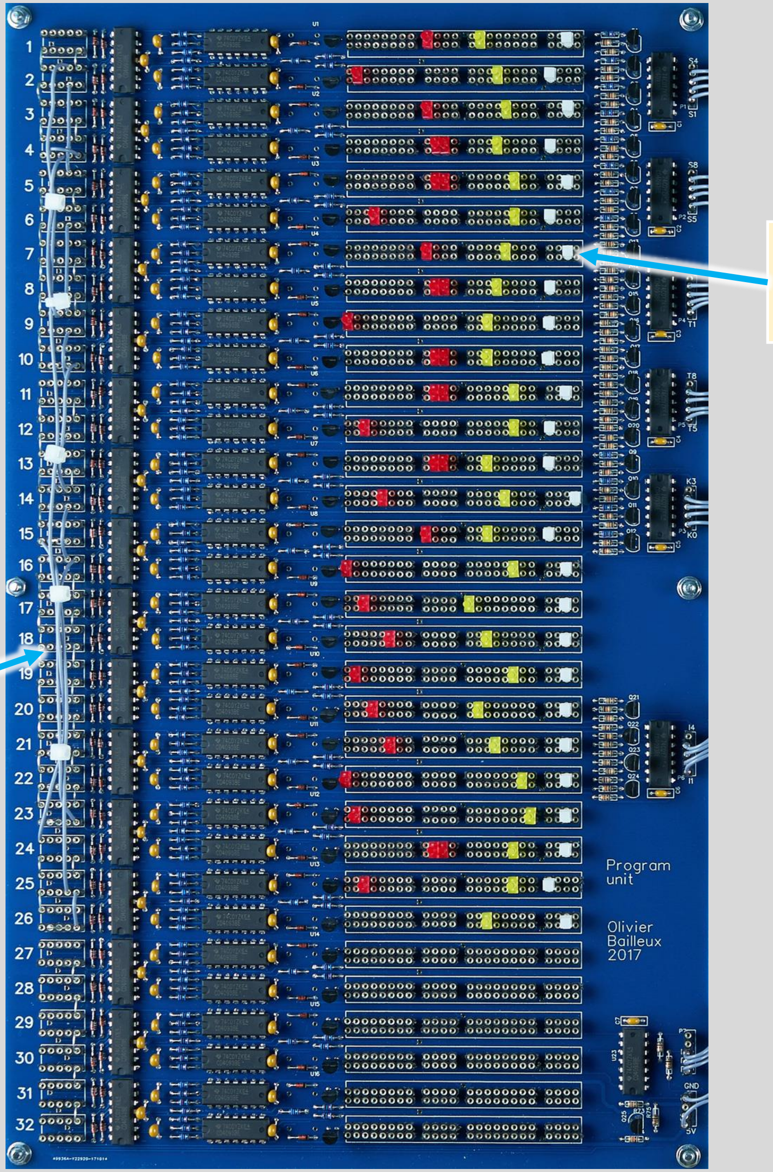
The program has at most 32 lines, each of them including an instruction and a conditional jump. The instruction is divided into three parts : the source operand (either a constant or a register number), the target operand (a register number), and the operation code.

After executing the instruction, the program continues by jumping to a new line according to the value of the flag.

The 4 following operations are available :

operation	code	result	flag
addition	1	`source + target -> target`	`carry`
nand	2	`!(source & target) -> target`	`0 if result = 0, else 1`
move	3	`source -> target`	`0`
div2	4	`source / 2 -> target`	`lowest bit`

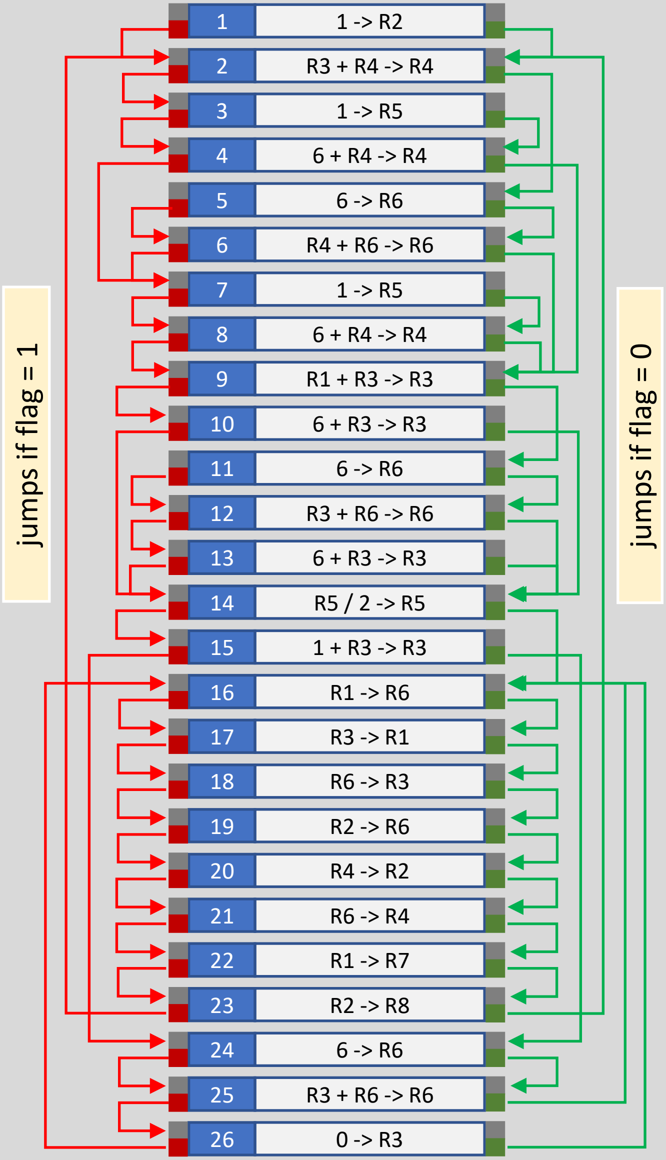
Program unit



jumps wiring

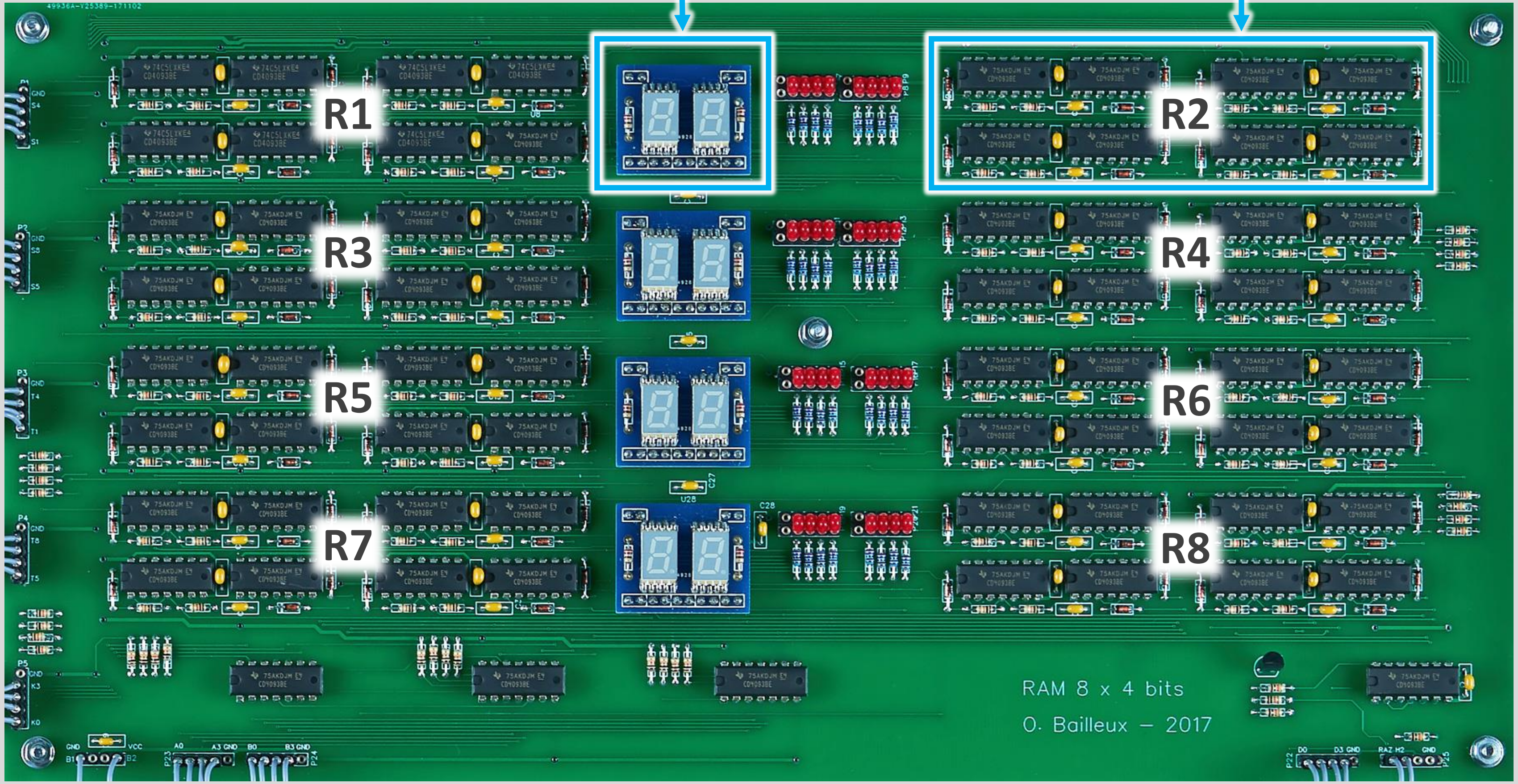
program line encoding

synthetic vue of the demonstration program



1	1 -> R2
2	R3 + R4 -> R4
3	1 -> R5
4	6 + R4 -> R4
5	6 -> R6
6	R4 + R6 -> R6
7	1 -> R5
8	6 + R4 -> R4
9	R1 + R3 -> R3
10	6 + R3 -> R3
11	6 -> R6
12	R3 + R6 -> R6
13	6 + R3 -> R3
14	R5 / 2 -> R5
15	1 + R3 -> R3
16	R1 -> R6
17	R3 -> R1
18	R6 -> R3
19	R2 -> R6
20	R4 -> R2
21	R6 -> R4
22	R1 -> R7
23	R2 -> R8
24	6 -> R6
25	R3 + R6 -> R6
26	0 -> R3

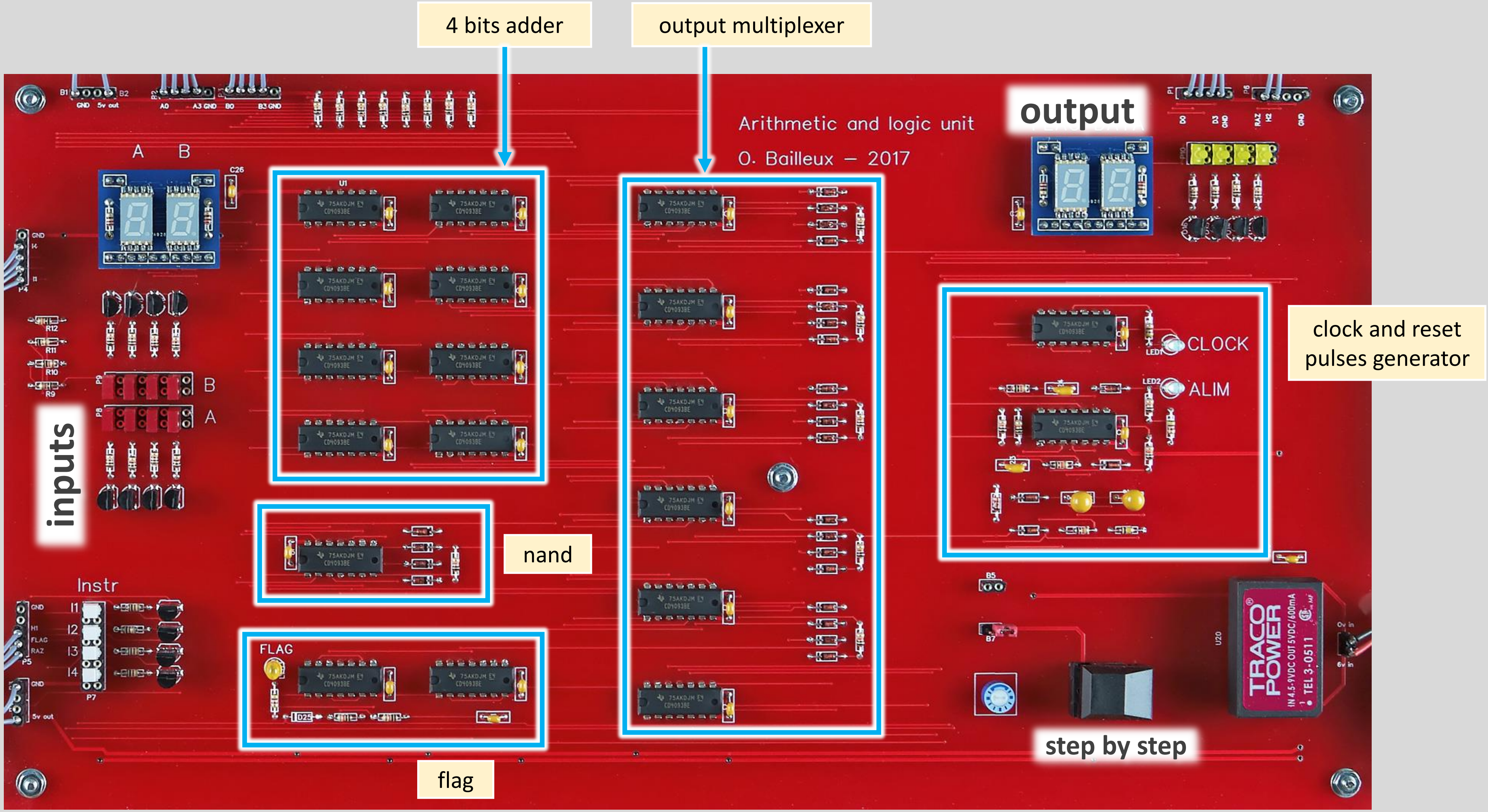
Memory unit



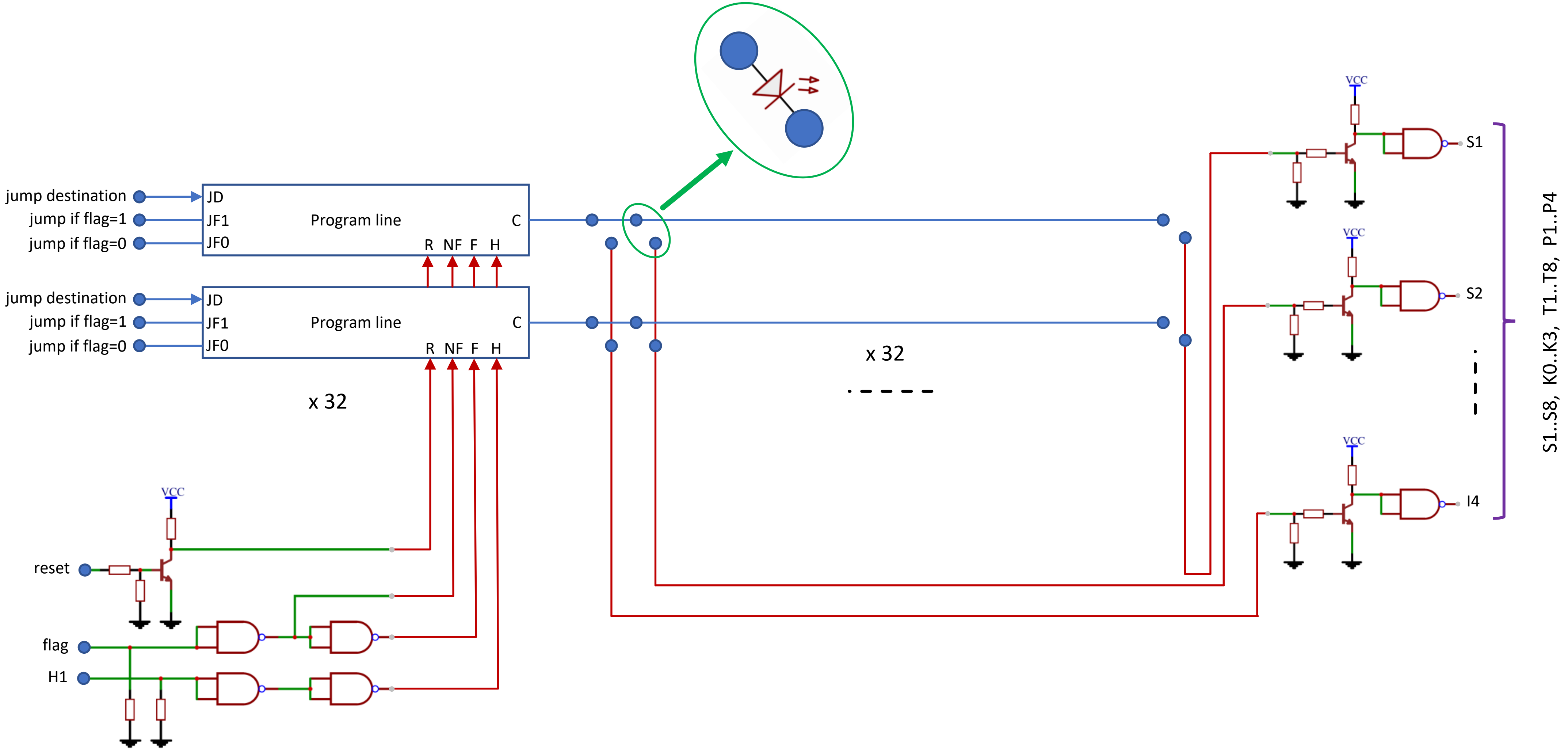
This display unit can be removed without affecting the operation.

4 bits register

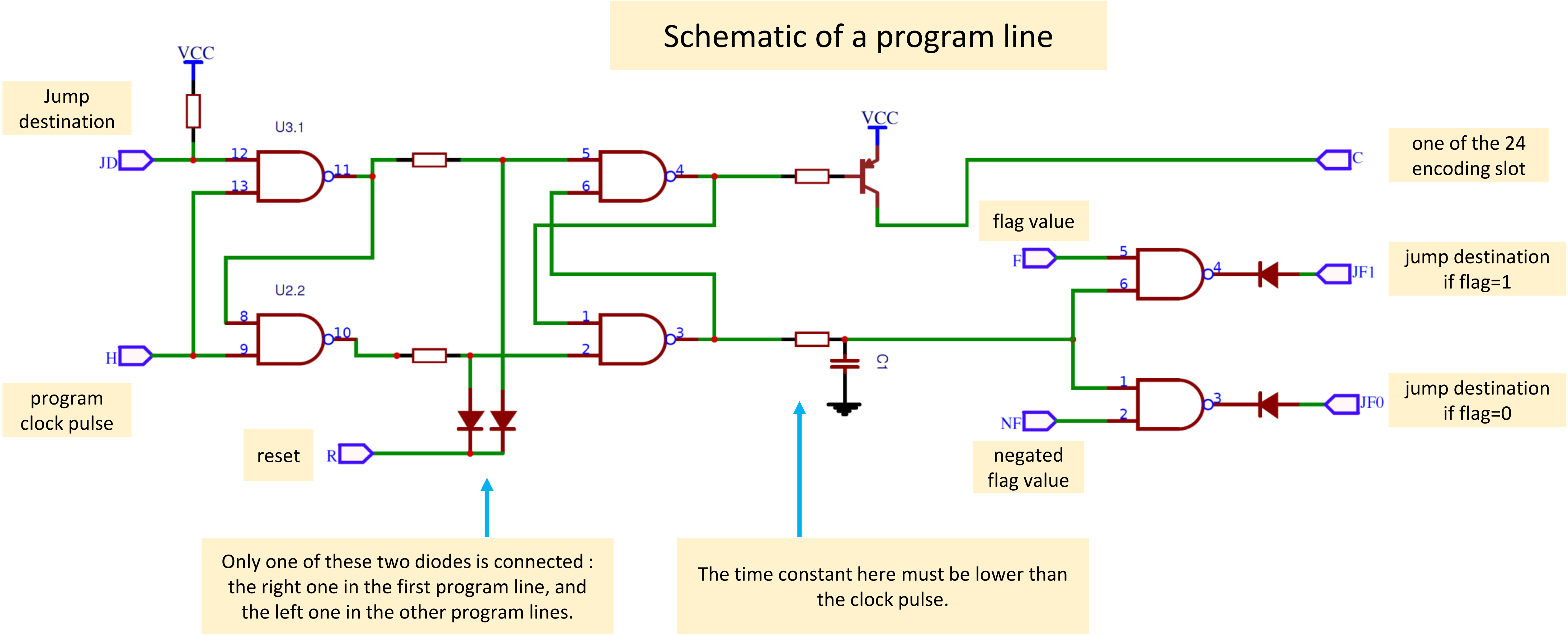
Program unit



The program unit : implementation



The program unit : implementation



Schematic of a program line

Jump destination

program clock pulse

reset

one of the 24 encoding slot

flag value

jump destination if flag=1

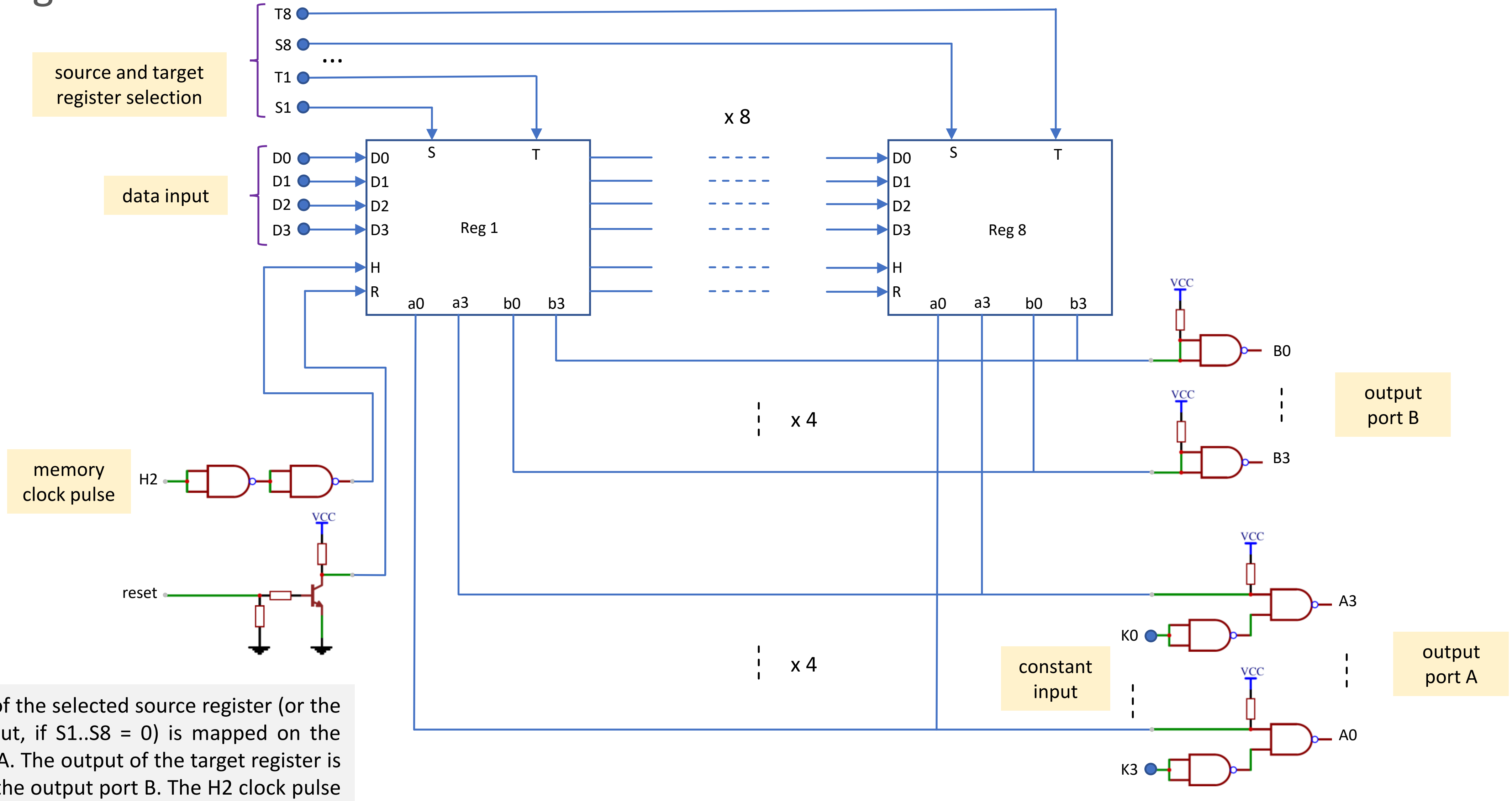
jump destination if flag=0

negated flag value

Only one of these two diodes is connected : the right one in the first program line, and the left one in the other program lines.

The time constant here must be lower than the clock pulse.

The registers


source and target register selection

data input

memory clock pulse

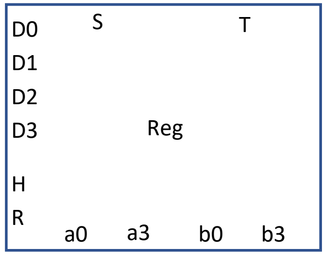
output port B

constant input

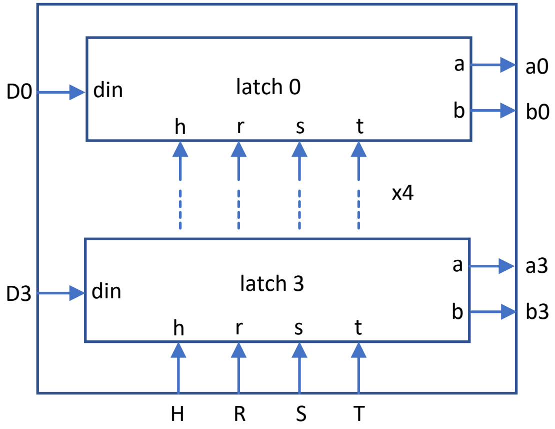
output port A

The output of the selected source register (or the constant input, if S1..S8 = 0) is mapped on the output port A. The output of the target register is mapped on the output port B. The H2 clock pulse stores the input data in the target register.

The registers

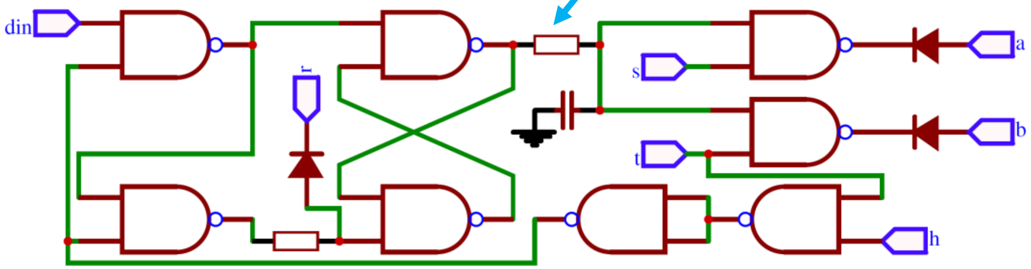


Each register includes 4 latches



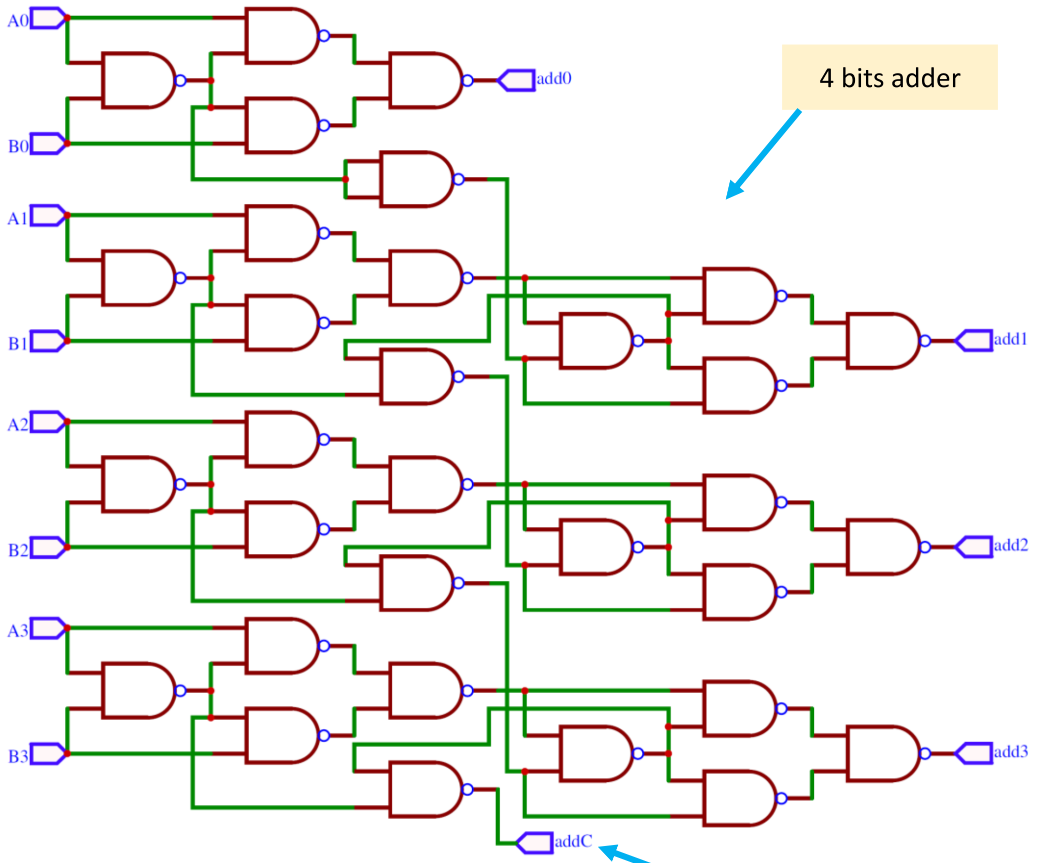
Each latch include 8 nand gates

The time constant here must be lower than the clock pulse.



Each of the 8 registers includes 4 latches, each of them including 8 nand gates. A latch is transparent when t = h = 1, else the last value of din is stored. The output a is pulled to 0 if s = 1 and the stored value is 1. The output b is pulled to 0 if t = 1 and the stored value is 1. The stored value is set to 0 when r = 0.

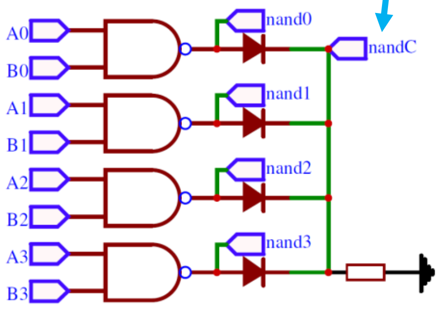
The calculation unit (add and nand operations)



4 bits adder

carry

The addC and nandC will be mapped to the flag, according to the current operation.

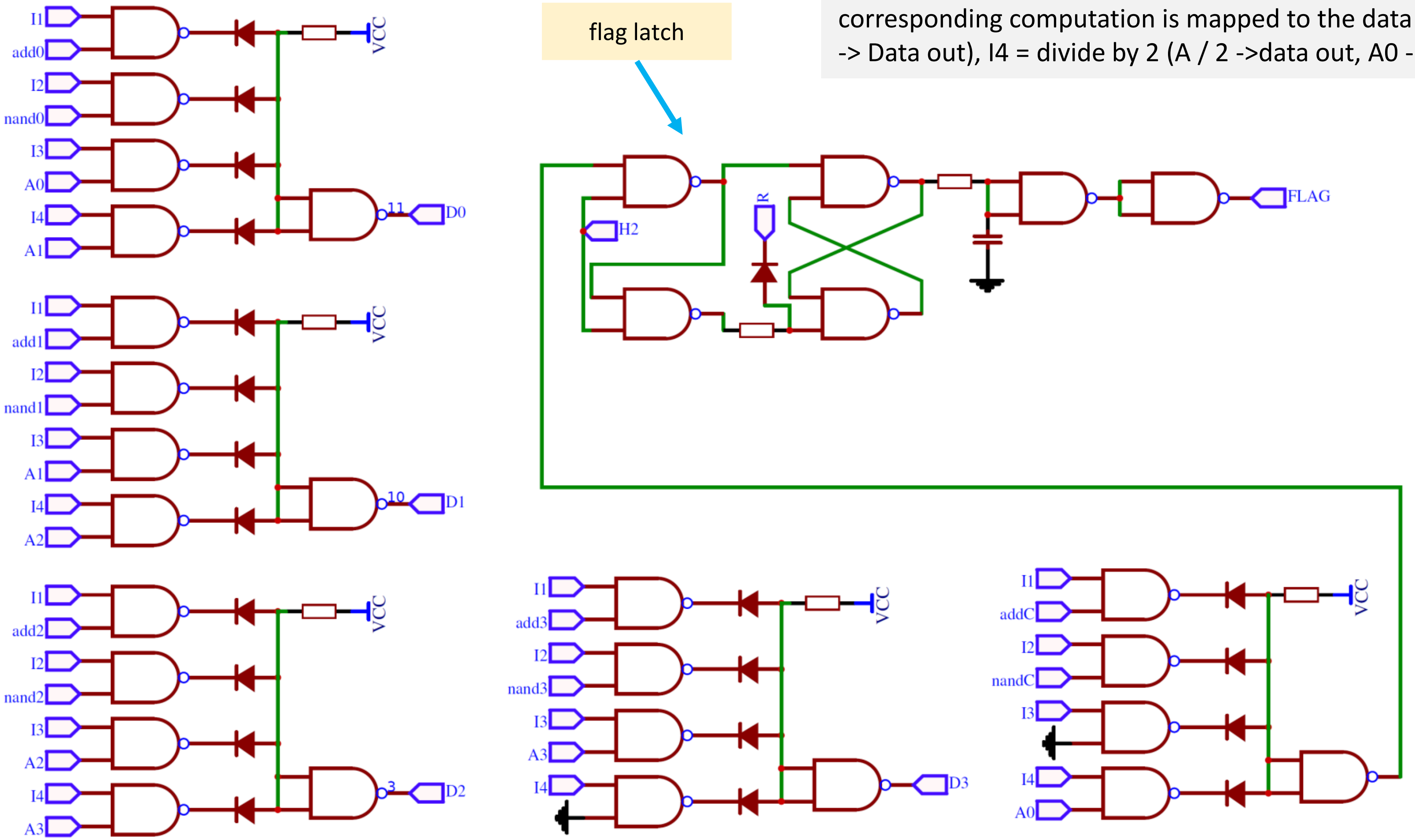


4 bits nand

This output = 0 if and only if the A nand B = 0000

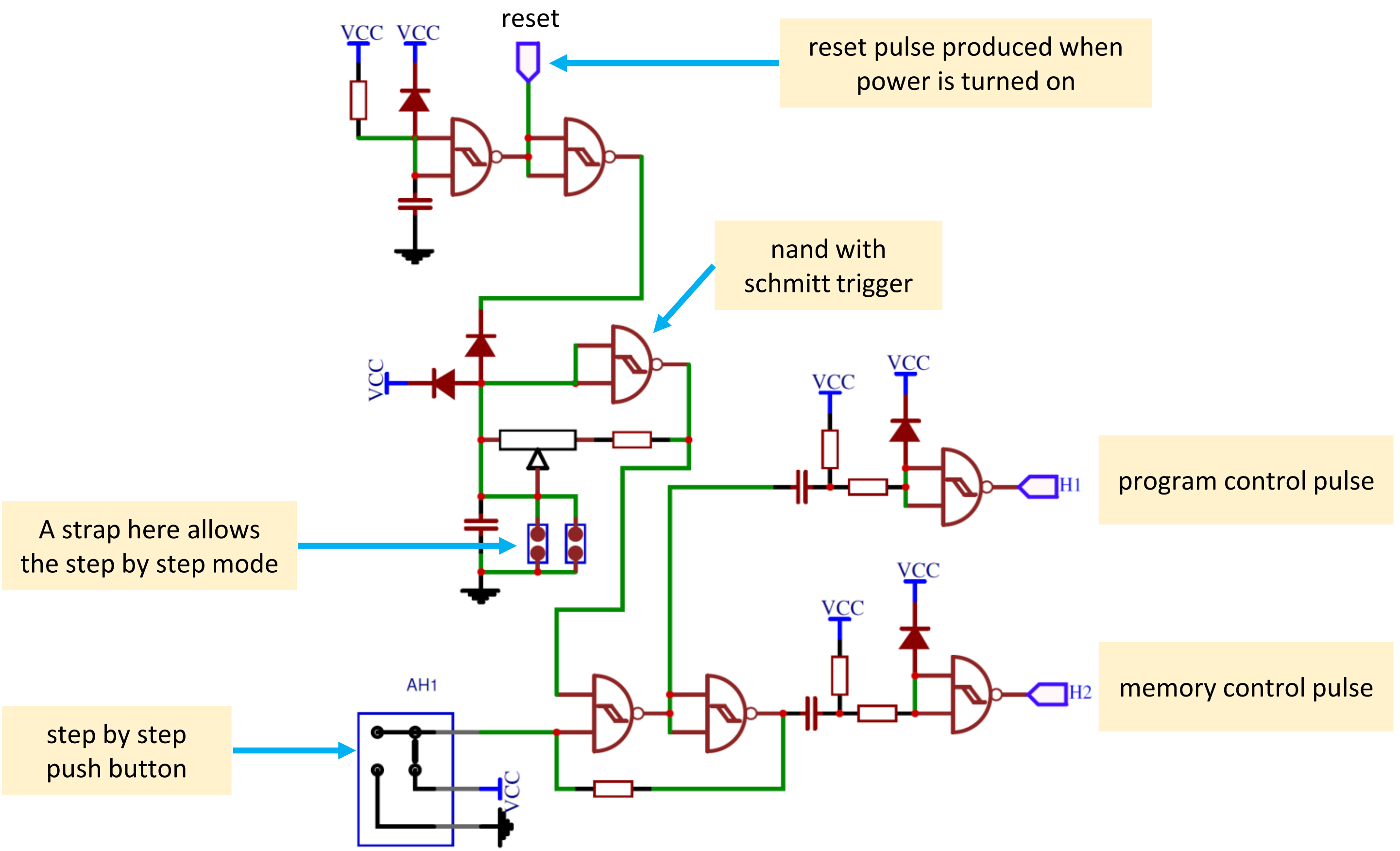
The calculation unit (results multiplexing)

According to the current operation, either I1, I2, I3 or I4 is set to 1. The result of the corresponding computation is mapped to the data output : I1 = addition, I2 = nand, I3 = move (A -> Data out), I4 = divide by 2 (A / 2 ->data out, A0 -> flag).



The flag is latched at the same time as the computation result is stored in the destination register (i.e., during the H2 pulse).

The calculation unit (Reset and clock pulses)



reset pulse produced when power is turned on

nand with schmitt trigger

program control pulse

A strap here allows the step by step mode

memory control pulse

step by step push button

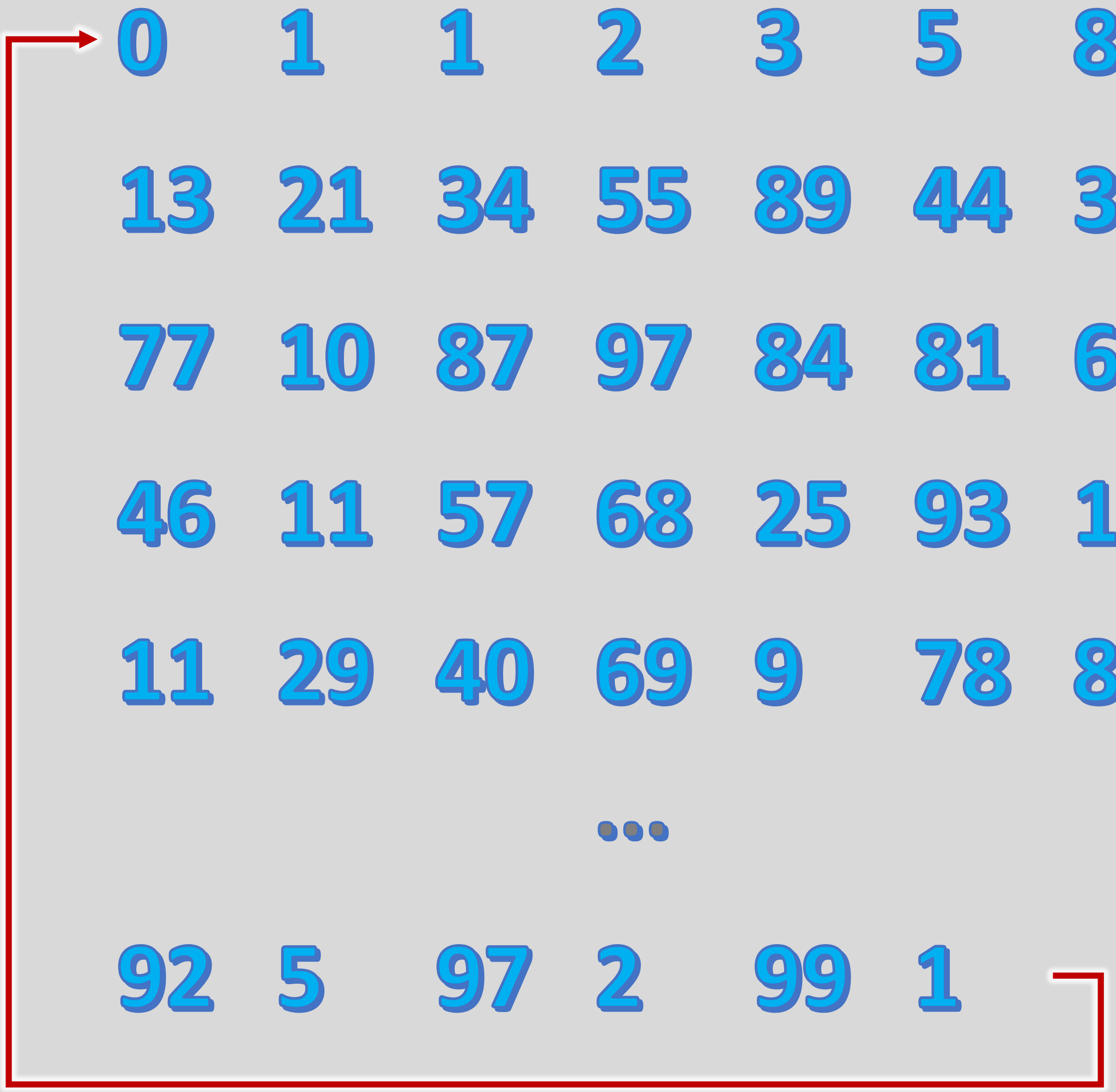
Demonstration program

The Fibonacci sequence modulo 100

$$\begin{cases} x_0 = 0 \\ x_1 = 1 \\ x_i = x_{i-1} + x_{i-2} \ \mathbf{mod}\ 100 \end{cases}$$

If the result exceeds 99, only the last two digits are retained.

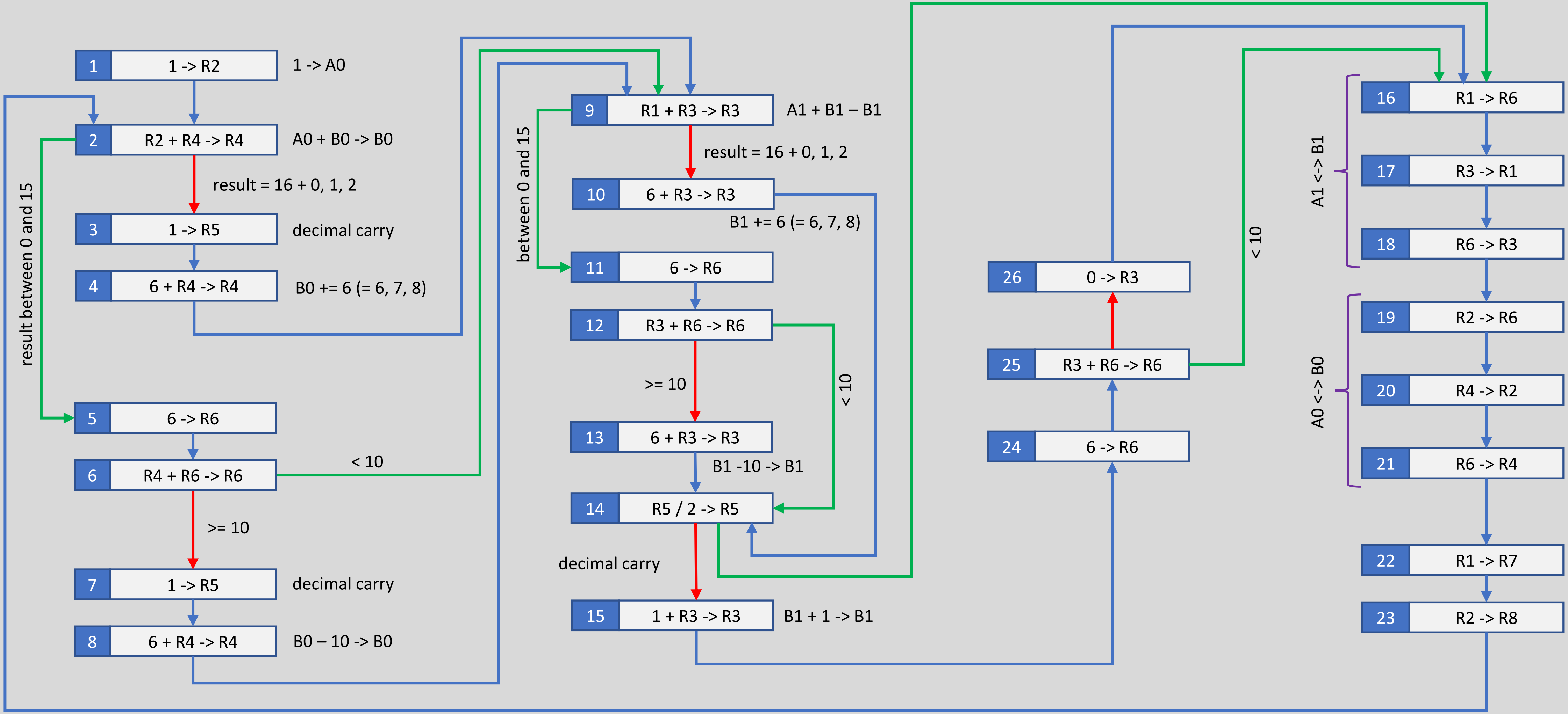
$$55 + 89 = 1\boxed{44}$$

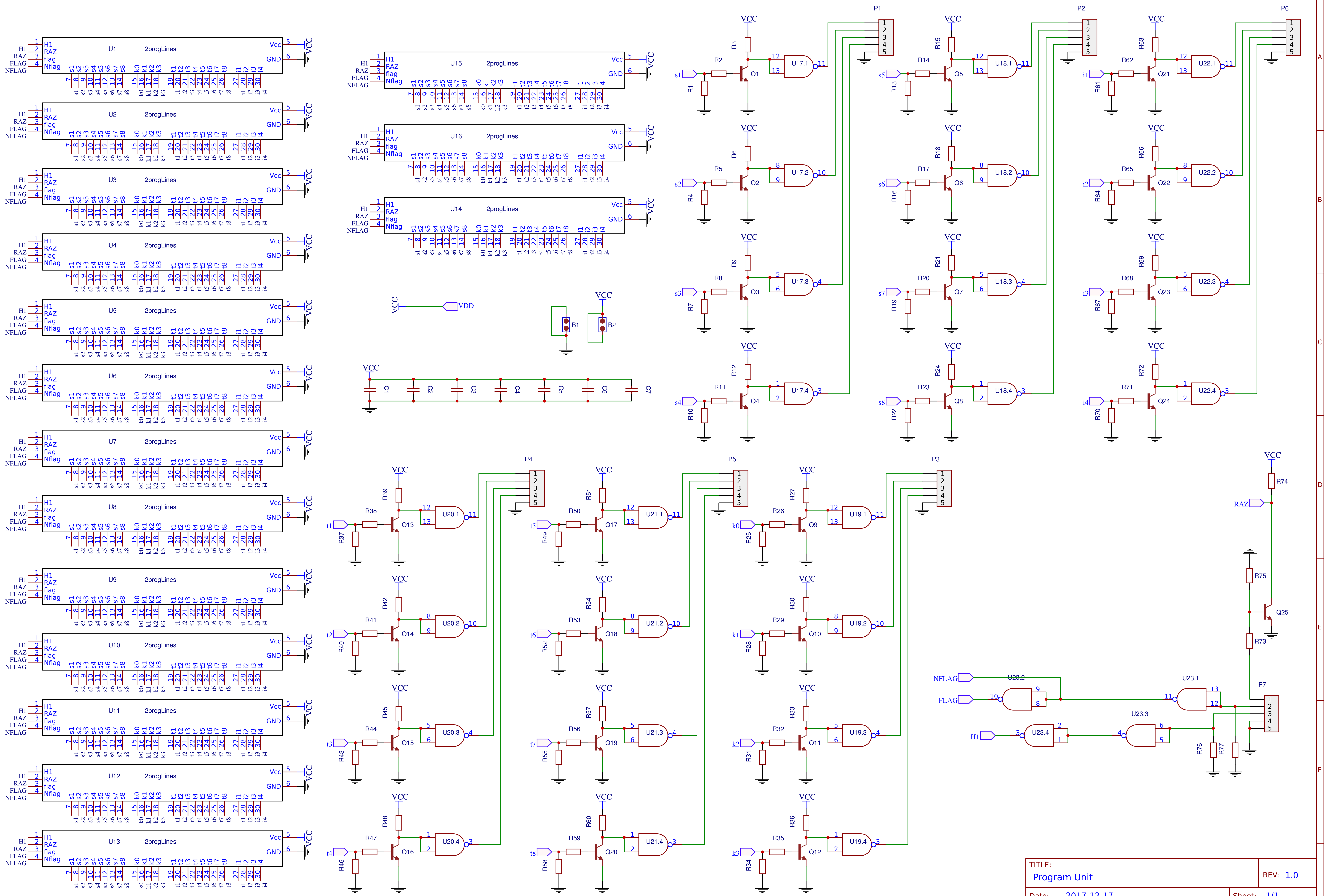


0	1	1	2	3	5	8
13	21	34	55	89	44	33
77	10	87	97	84	81	65
46	11	57	68	25	93	18
11	29	40	69	9	78	87
			...			
92	5	97	2	99	1	

Demonstration program

This program computes the Fibonacci sequence modulo 100 in binary-coded decimal. It consists of a sequence of 300 numbers (in 0..99) that repeats indefinitely. The current term is stored in R7 / R8.





TITLE:

Program Unit

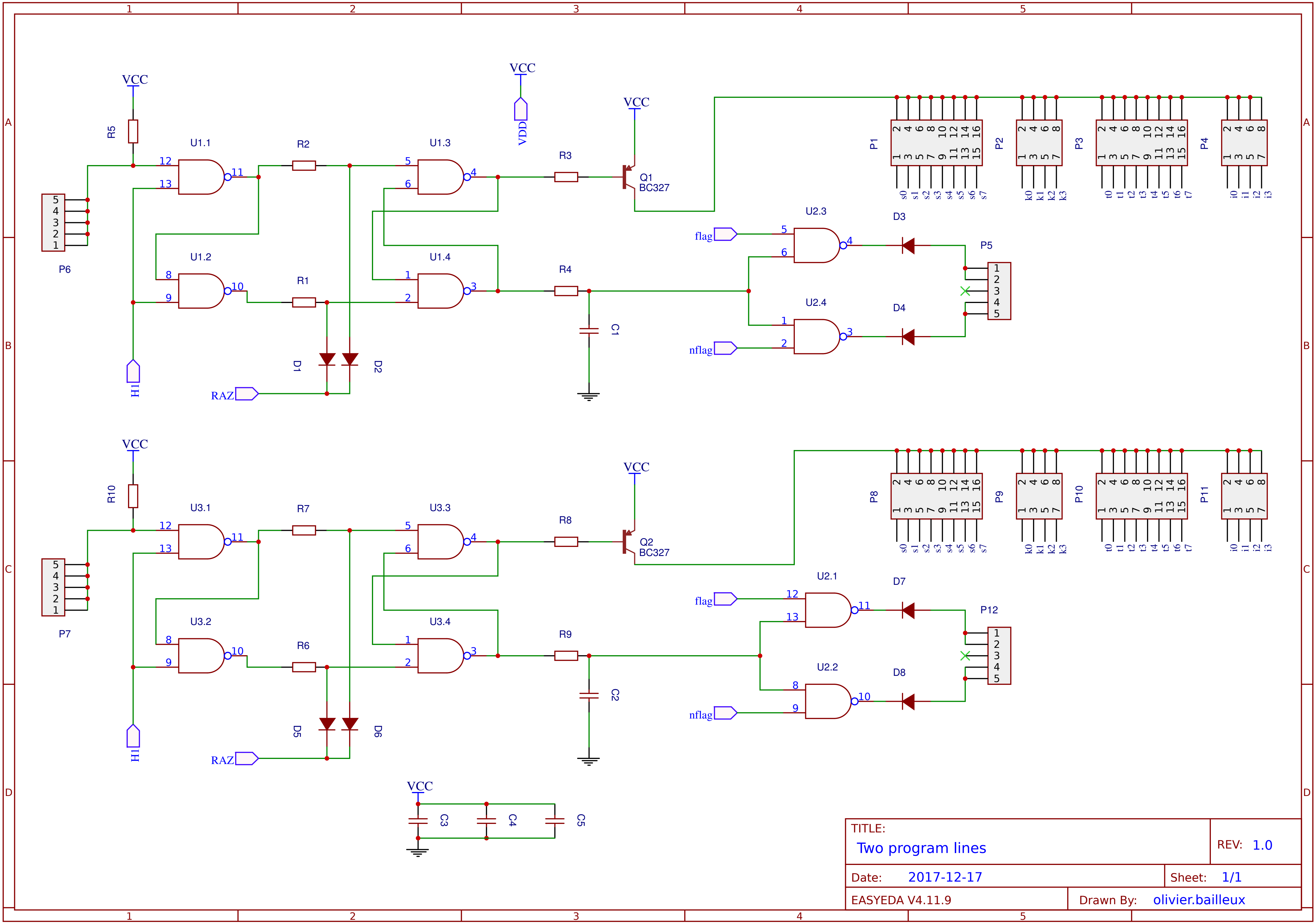
REV: 1.0

Date: 2017-12-17

Sheet: 1/1

EASYEDA V4.11.9

Drawn By: olivier.bailleux



TITLE:

Two program lines

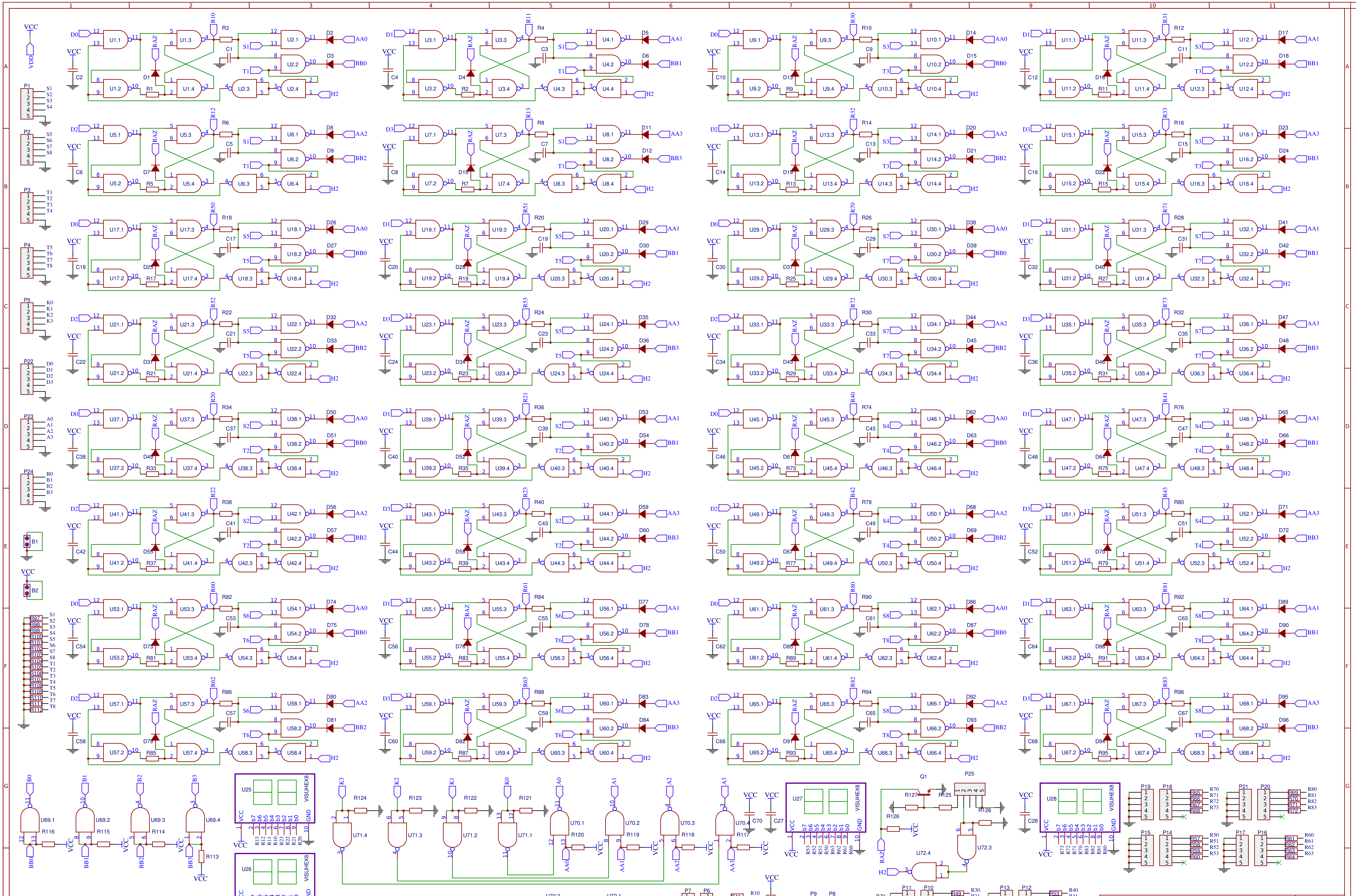
REV: 1.0

Date: 2017-12-17

Sheet: 1/1

EASYEDA V4.11.9

Drawn By: olivier.bailleux



TITLE:

Ram Module

REV: 1.0

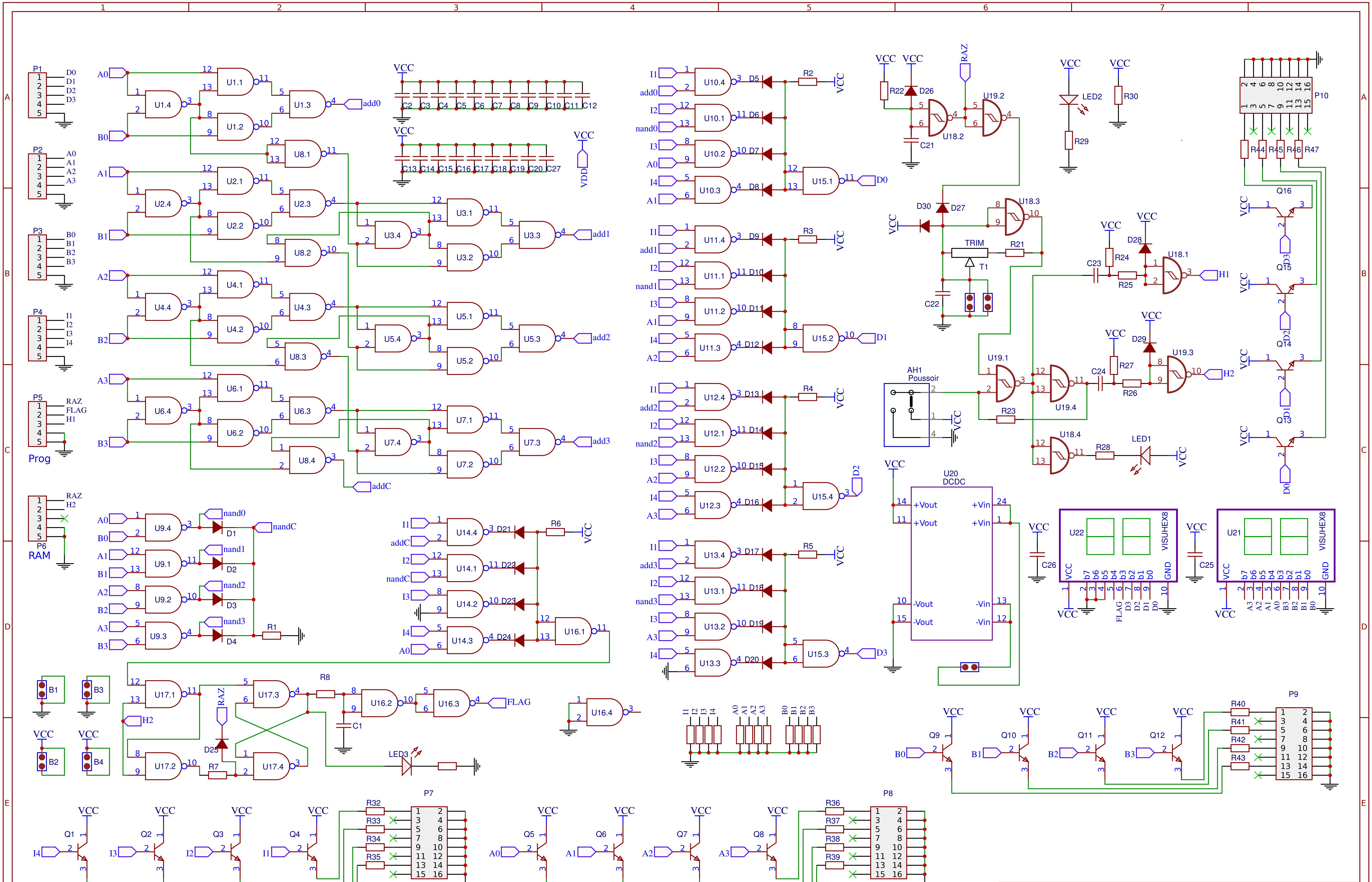
Date: 2017-12-17

Sheet: 1/1

EASYEDA V4.11.9

Drawn By: olivier.bailleux

TITLE:



TITLE:
Calculation unit

REV: 1.0

Date: 2017-12-17

Sheet: 1/1

EASYEDA V4.11.9

Drawn By: olivier.bailleux