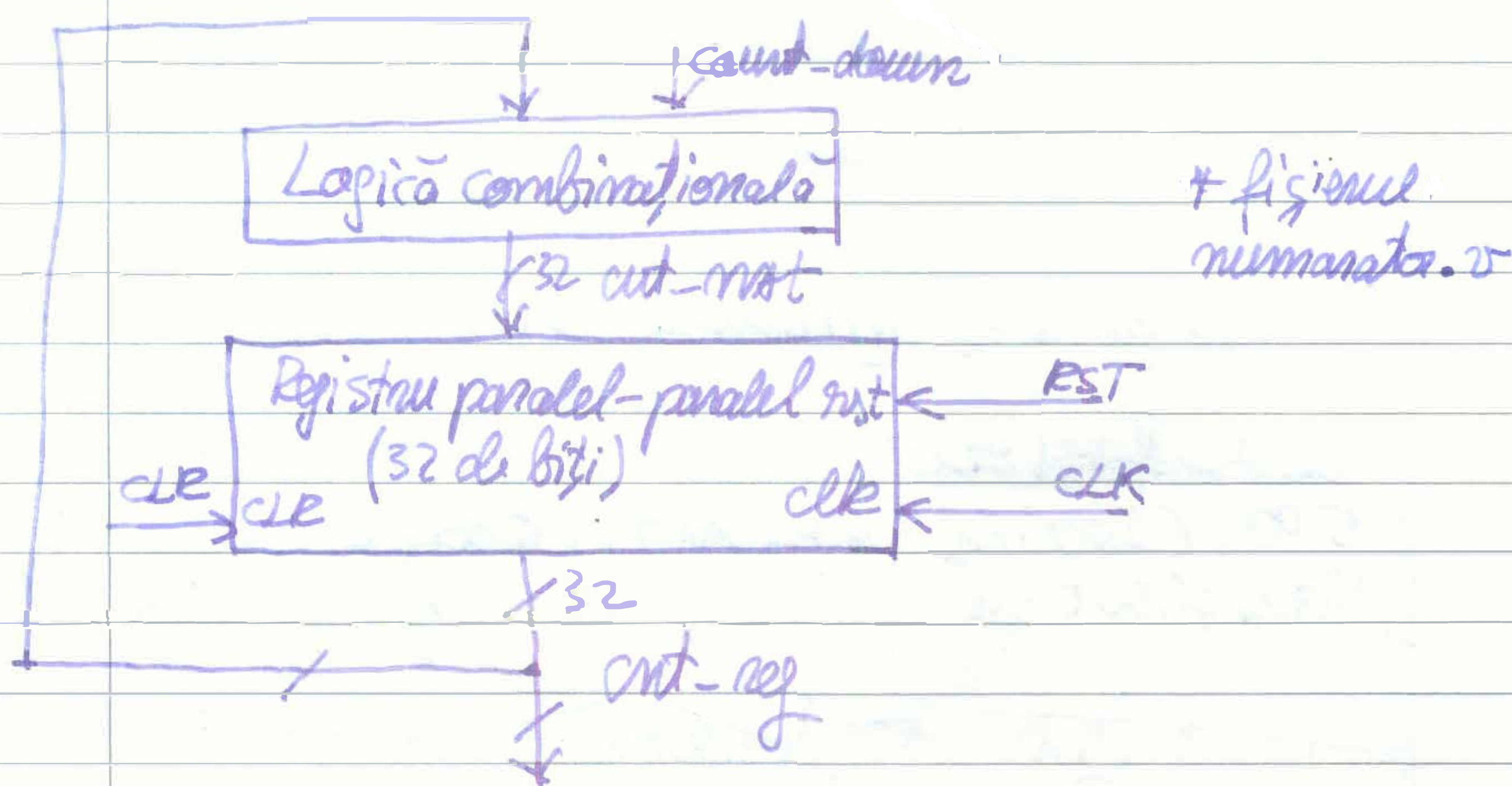


# Assignment 1

Vom folosi:

→ un numărator proiectat conform principiilor RTL:

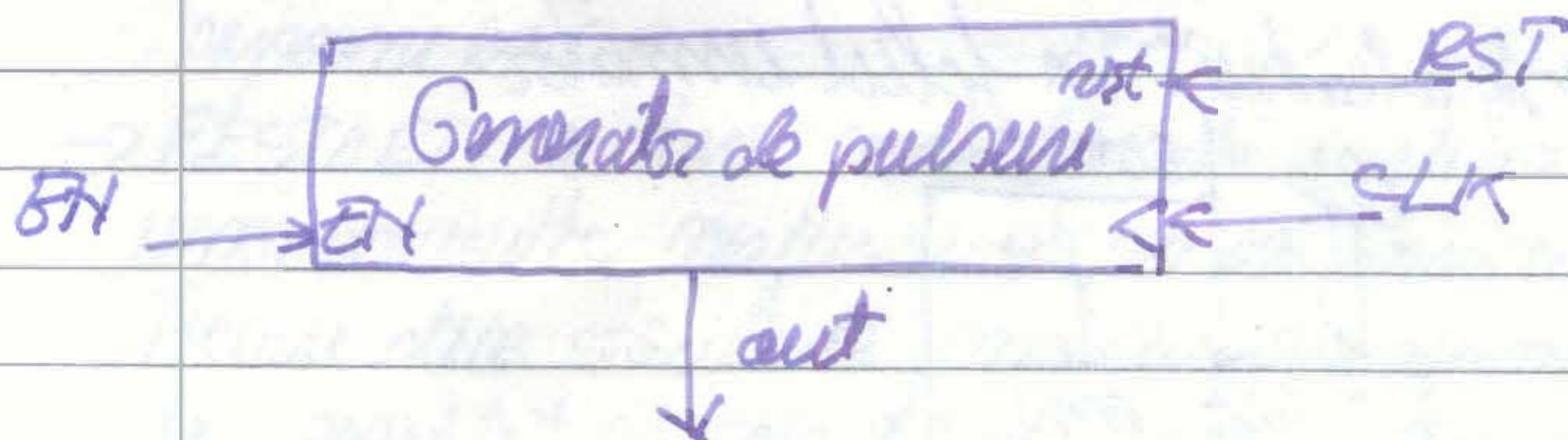


Descriere: → count-down: când este activ numără descrescător, iar când este inactiv numără crescător (activ pe  $\overline{1}$ )  
→ RST: Semnal de reset asincron (activ pe  $\overline{1}$ )  
→ CLR: Semnal de clear sincron (activ pe  $\overline{0}$ )  
→ CLK: Semnal de tact  
→ registru este construit din 32 de bițiabile  
sincrone, comută la frontul crescător  
al semnalului de tact  
→ În cazul nostru numărătorul are ca valoare de  
preîncărcare sau valoare limită superioară 2500 (numărul 2500 de pași)

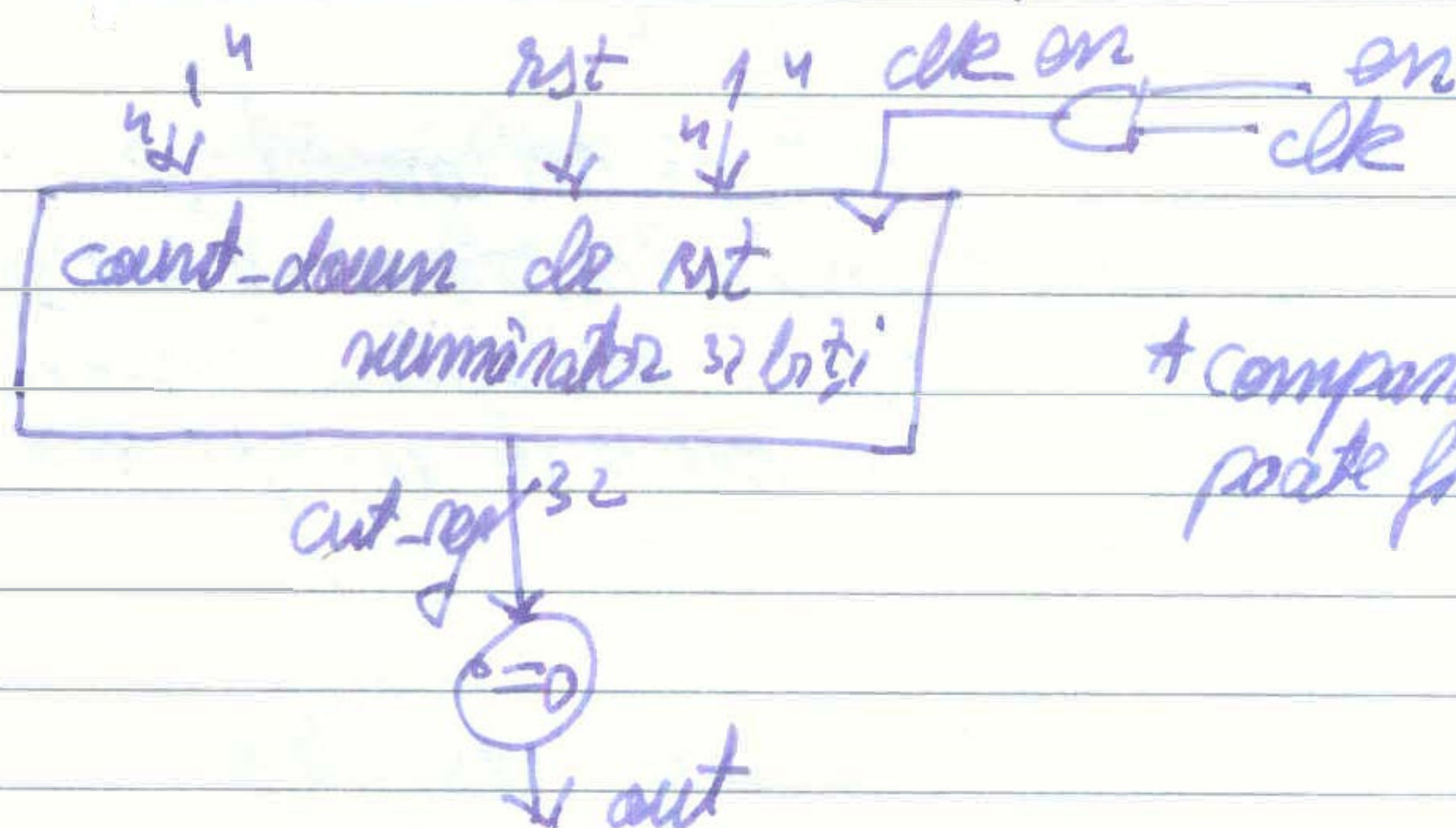


→ un generator de pulsuri, care este construit folosind circuitul anterior (numărătorul)

\* fisierul:  
generatores



Descriere: → semnal de tact (CLK): oclon pe frontul cresterii  
→ reset sincron (RST): activ în '0'  
→ semnal de activare (EN): activ în '1'

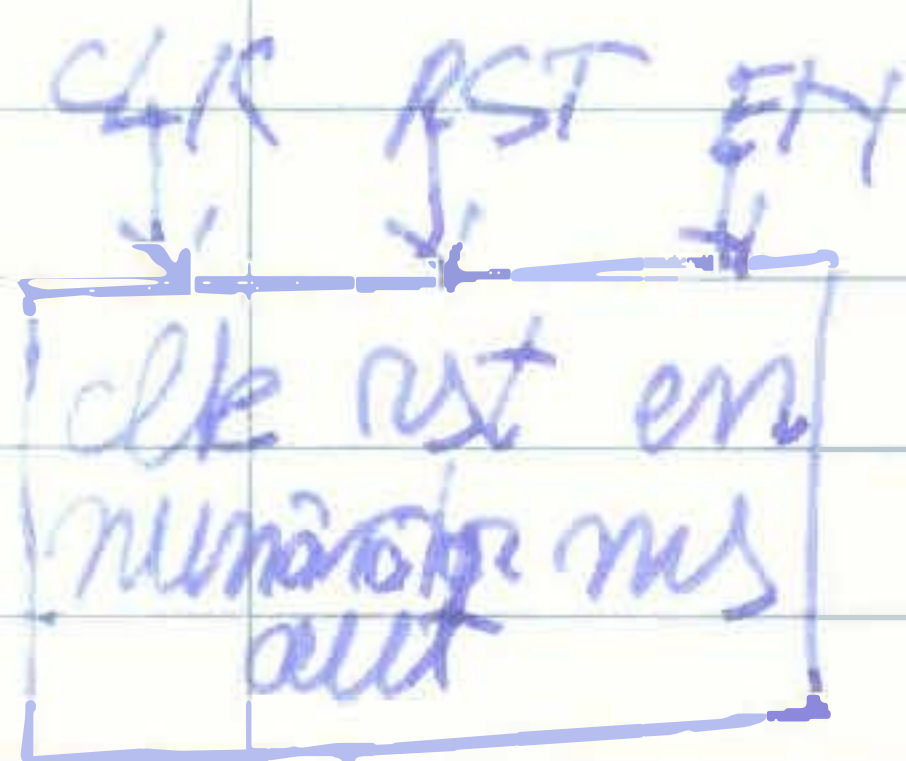


\* comparatorul ⑤  
poate fi o poarta NOR.

→ semnalul de tact are frecvența de  $f = 2,5 \text{ MHz}$ , iar cum numărătorul numără 2500 de poziții → generatorul de pulsuri generează un puls la  $\frac{2500}{2,5 \cdot 10^6} = 10^{-3} \text{ s} = 1 \text{ ms}$ .



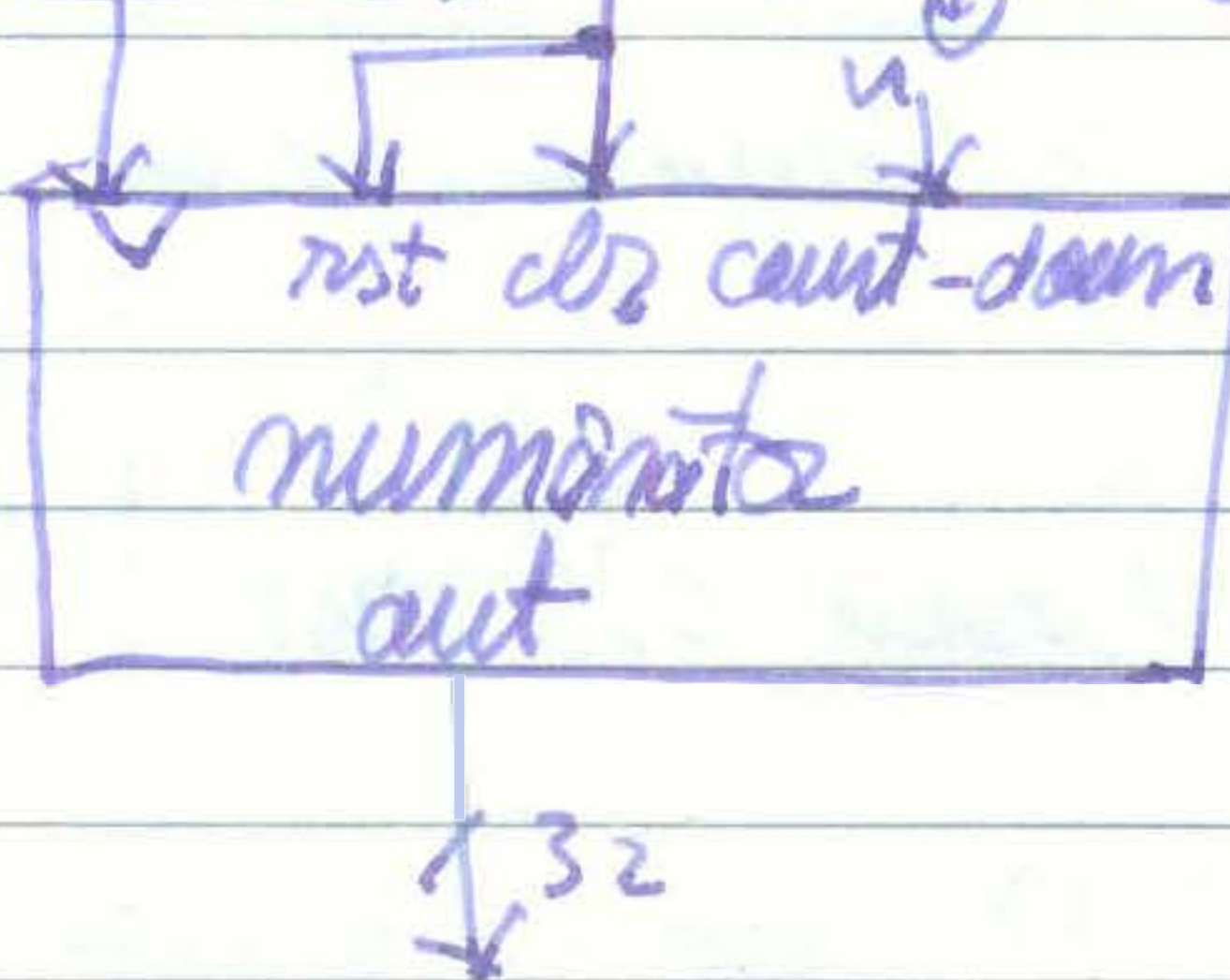
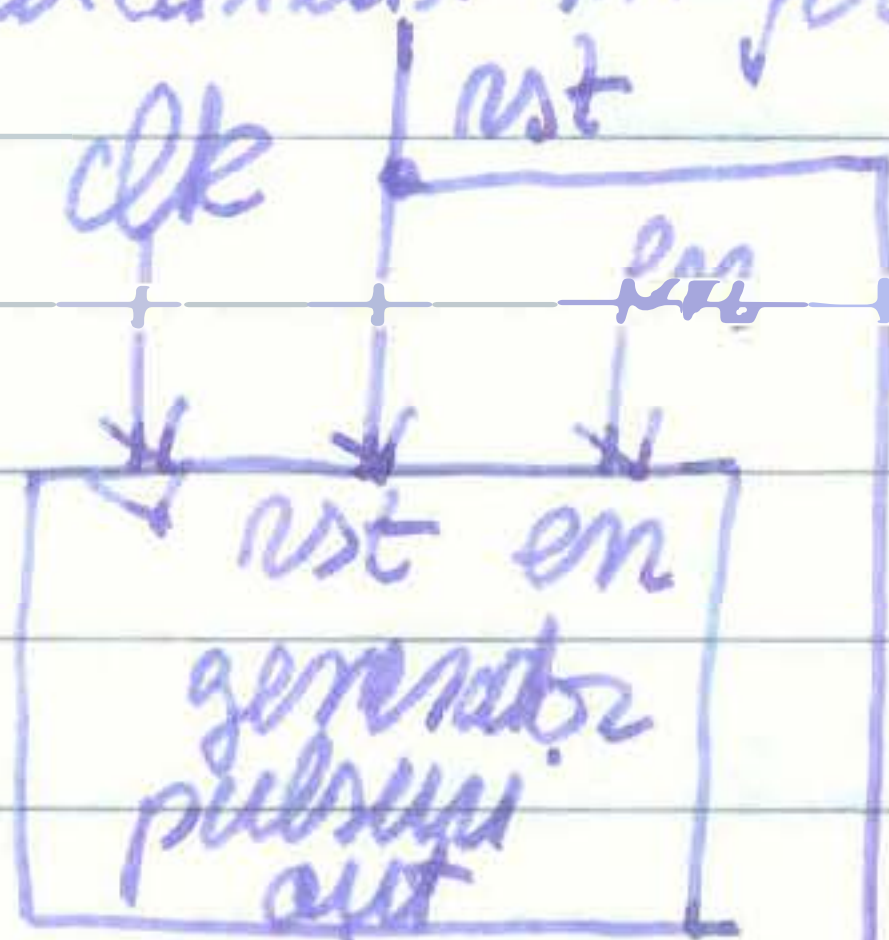
→ un numărator de milisecunde alcătuit din cele două circuite descrise anterior: un numărator, respectiv un generator de pulsuri (care va juca rolul generatorului semnalului de test cu perioadă de  $T=1ms$ )



32  
aut

\* fișierul: ms.v

este alcătuit în felul următor:



numără  
crescător

→ circuitului anterior îi vom adăuga un convertor BCD, acesta va avea ieșiri de 4 biți (reprezentare BCD) pentru cifrele: mii, sute, zeci și unități. Presupunem că nu vom număra pe mai mult de 10 biți (milisecunde).  
\* fișierul: bcd.v

→ circuitul anterior poate fi comandat de un automat Moore (în cazul nostru) cu următoarele stări:  
START, PAUSE, STOP, RESTART, având intrările:  
nop, start, pause, stop.  
(00) (01) (10) (11)



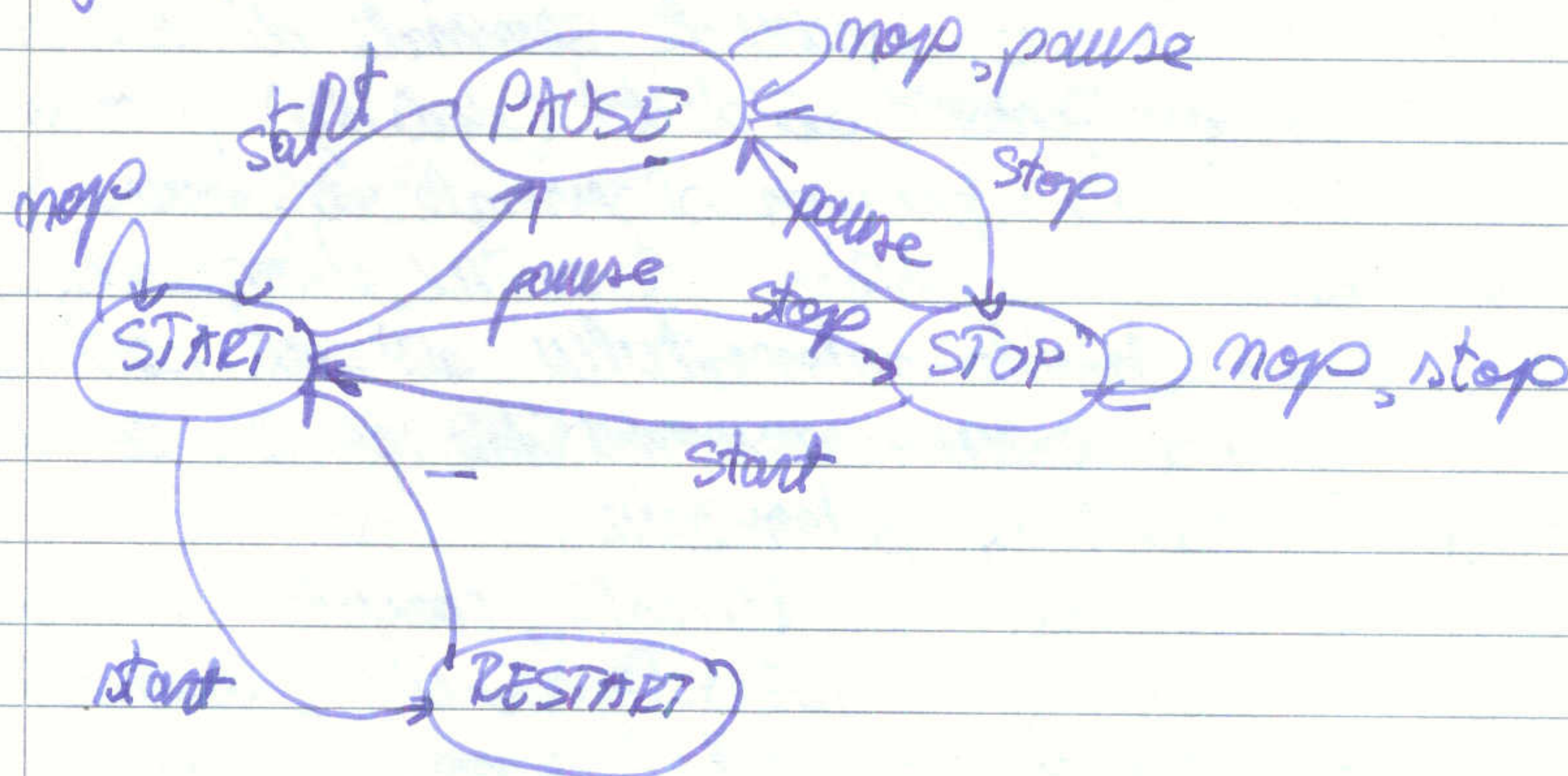
Stările produc un anumit impact "asupra circuitului":

- START: este declanșată numărarea
- PAUSE: este oprită numărarea, dar valoarea de la care va reporni este cea la care a rămas
- STOP: este oprită numărarea, dar valoarea de la care va reporni este 0
- RESTART: o stare confecționată "artificial" pentru oprimarea unor anumite măsurări (nu în prezenta inițial)

Fiecare din stările de mai sus va acționa asupra semnalelor de "enable" și "reset" ale numărătorului de milisecunde.

	enable	reset	
→ START:	activat	deactivat	* fisierul automat.v
→ PAUSE:	deactivat	deactivat	
→ STOP:	x	deactivat	
→ RESTART:	x	deactivat	

Graful automatului Simpson:



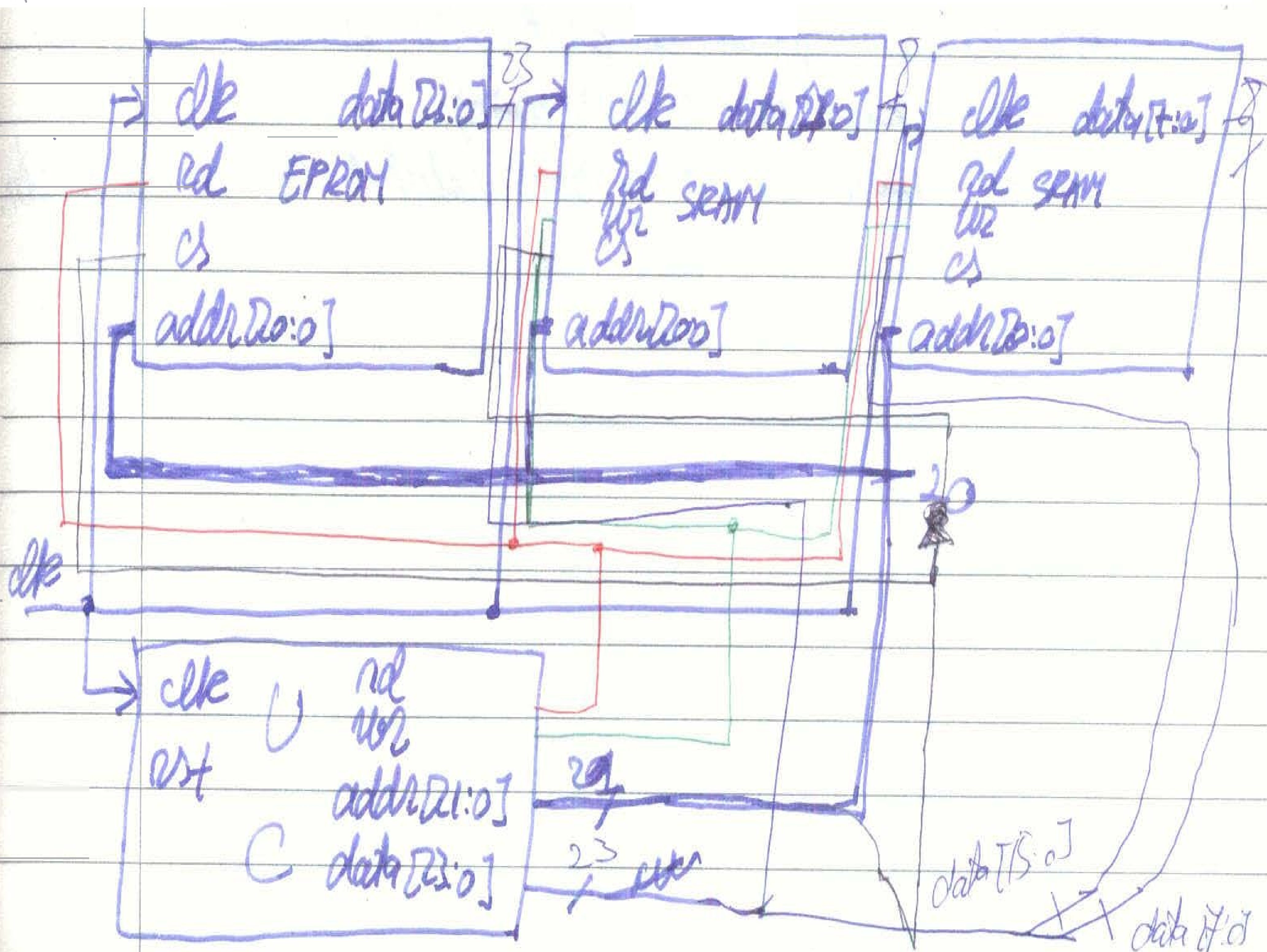
Am folosit starea RESTART pentru cazul în care "reapetăm" "start" din starea de START (reîncepă numărarea, însă de la 0).

Top module-ul pentru întreaga pseudounitate automată-geometrică în fisierul: unitatecontrol.v.



- avem două tipuri de memorie: EPROM în carel nostru va stoca instrucțiunile și va putea fi doar citită, existând în fișorul eprom.v (un ciclu inițial de decodare) și SRAM (care va conține datele, implementarea fiind în fișorul sram.v).
- EPROM: are semnale de tact, read, dup select address și data, mai concret:  $\overline{st}$  activă la frontul oricărui semnalului de tact operativ, read: memoria situează pe magistrala de date conținutul de la adresa de memorie address, altfel o stă în impedanță ridicată, cuvântul are dimensiunea de 24 de biți iar dimensiunea memoriei este de  $2^{21}$  (are 21 de biți de adresă).
- SRAM: are semnal de tact, al cărui front comandă decodarea operațiilor, semnal de read care atunci când este activ face ca memoria să situeze pe magistrala de date (de doar 8 biți aici) cuvântul stocat la adresa address (pe 21 de biți), iar în cazul în care read este inactiv, dacă write este activ, magistrala de date devine în impedanță ridicată pentru a putea primi date pe aceasta, care vor fi scrise la adresa de memorie address.
- unitate centrală care dispune de semnale de read, write, tact, reset asincron, magistrală de date (24 de biți) și magistrală de adresă (22 de biți), precum și semnale de comunicare cu top module-ul care conține automatul Moore, alu-in pentru obținerea inhibiției automatului, alu-out pentru obținerea primirii inhibiției aceluși automat, dar și alu-ck pentru semnalul de tact al automatului.
- decodificarea memoriilor este trivială, completă, LSB-ul de pe magistrala de adresă va intra în CS-urile memoriilor, Când A0 este activă va fi selectată memoria de cod (EPROM), în rest ea are date (vom folosi 2 în paralel, întrucât intrarea automatului este pe 16 biți - 4 cifre BCD - iar magistrala de date a fiecărei memorii SRAM este de doar 8 biți).
- unitatea centrală este de tip von Neumann (non-pipeline) având 5 stadii, care vor fi descrise mai amănunțit ulterior.







## Stagiile unității centrale:

IF:  $AR \leftarrow PC + 1$ ; pentru decodificare codului  $1^n$

$WR \leftarrow 0$ ; deactivăm scrierea, în cazul în care a fost activată

ID:  $PC \leftarrow PC + 1$ ; incrementăm număratorul de program

$RD \leftarrow 1$ ; prețitim o citire

DT:  $DR \leftarrow \text{data-bus}$ ; preluăm datele

Ex:  $\text{alu-clk} \leftarrow 1$ ; activăm tactul pentru automat

$IR \leftarrow DR[23:2]$ ; instrucțiunea reprezintă primii 24 biți

; de date, restul fiind rezervat

; pentru adresa la care să fie depozitat rezultatul în cazul

; instrucțiunilor de start sau „pause”

$RS \leftarrow 0$ ; nu vom mai citi

WB: if ( $IR = \text{pause}$  sau  $IR = \text{stop}$ )

$AR \leftarrow DR[21:0]$ ; preluăm adresa din START la care să

$DR \leftarrow \text{alu-out}$ ; scriem rezultatul ALU (automatului)

$WR \leftarrow 1$ ; vom activa citirea

$\text{alu-clk} \leftarrow 0$ ; s-a încheiat un ciclu al automatului

→ ciclurile de mai sus se repetă, fiecare ciclu durând o perioadă de tact

→ am folosit registrele: PC pe 24 de biți (număratorul de program), AR pe 22 de biți (registru de adrese),

DR pe 24 de biți (registru de date), IR (registru de instrucțiuni),

semnale de RD (read) și WR (write).

→ alu-out putem spune că joacă rolul de registru acumulator.