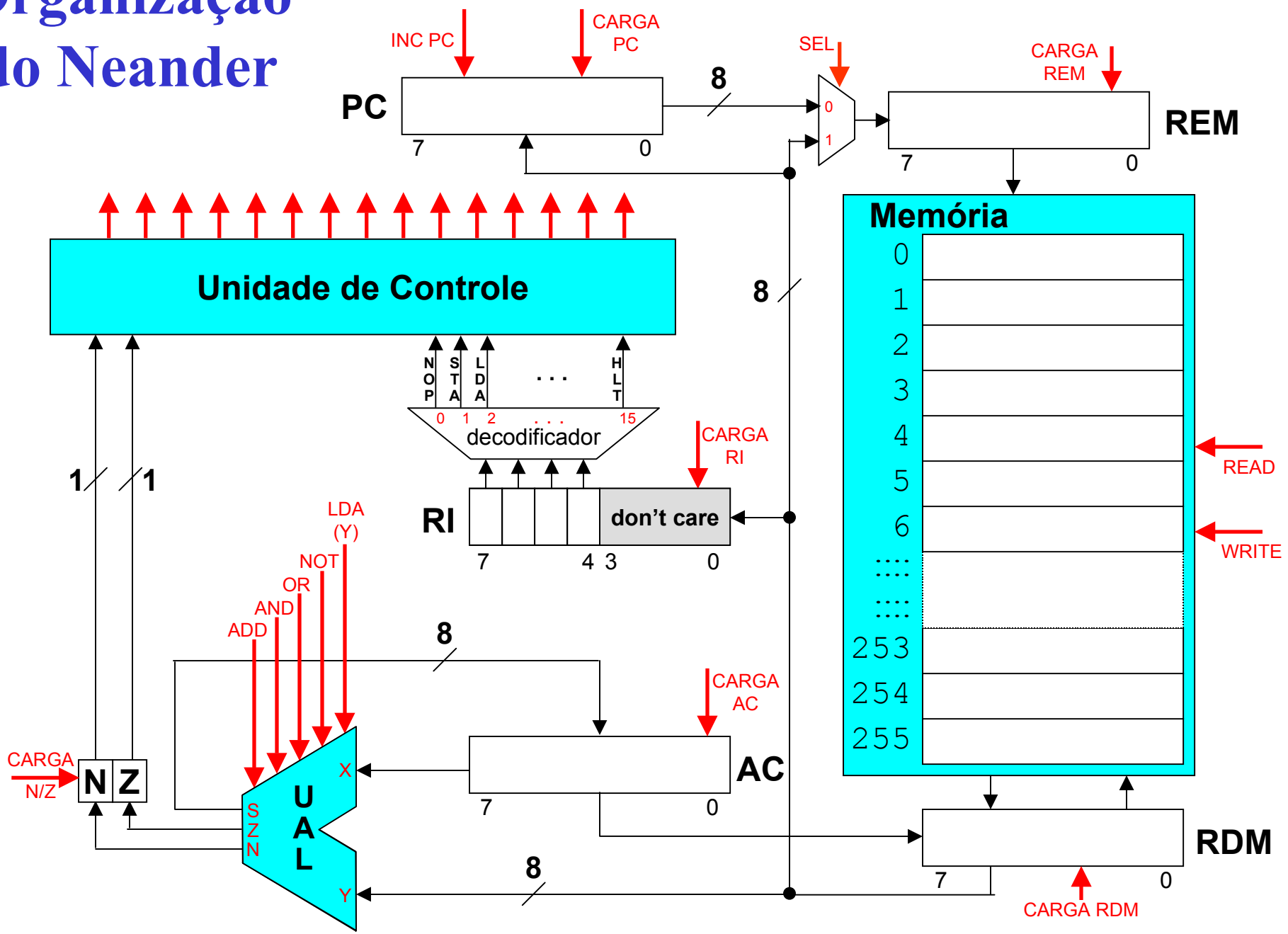
Organização do Neander



Sinais de controle do Neander para cada operação de transferência (micro-operação)

Transferência	Sinais de controle
REM ← PC	sel=0, carga REM
RDM ← MEM(REM)	Read
PC ← PC + 1	incrementa PC
RI ← RDM	carga RI
REM ← RDM	sel =1, carga REM
RDM ← AC	carga RDM
MEM(REM) ← RDM	Write
AC ← RDM; Atualiza N e Z	UAL(Y), carga AC, carga NZ
AC ← AC + RDM; Atualiza N e Z	UAL(ADD), carga AC, carga NZ
AC ← AC or RDM; Atualiza N e Z	UAL(OR), carga AC, carga NZ
AC ← AC and RDM; Atualiza N e Z	UAL(AND), carga AC, carga NZ
AC ← not(AC); Atualiza N e Z	UAL(NOT), carga AC, carga NZ
PC ← RDM	carga PC
Parar o processamento	Halt

Sinais de controle para instruções do Neander

ciclo	STA	LDA	ADD	OR	AND	NOT
t0	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM
t1	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC
t2	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI
t3	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	UAL(NOT), carga AC, carga NZ, goto t0
t4	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	
t5	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	sel=1, carga REM	
t6	carga RDM	Read	Read	Read	Read	
t7	Write, goto t0	UAL(Y), carga AC, carga NZ, goto t0	UAL(ADD), carga AC, carga NZ, goto t0	UAL(OR), carga AC, carga NZ, goto t0	UAL(AND), carga AC, carga NZ, goto t0	

Sinais de controle para instruções do Neander (continuação)

Ciclo	JMP	JN, N=1	JN, N=0	JZ, Z=1	JZ, Z=0	NOP	HLT
t0	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM	sel=0, carga REM
t1	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC	Read, incrementa PC
t2	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI	carga RI
t3	sel=0, carga REM	sel=0, carga REM	incrementa PC, goto t0	sel=0, carga REM	incrementa PC, goto t0	goto t0	Halt
t4	Read	Read		Read			
t5	carga PC, goto t0	carga PC, goto t0		carga PC, goto t0			
t6							
t7							

Expressões booleanas dos sinais de controle

carga REM = t0 + t3.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z +
t5.(STA+LDA+ADD+OR+AND)

incrementa PC = t1 + t4.(STA+LDA+ADD+OR+AND) + t3.(JN.N' + JZ.Z')

carga RI = t2

sel = t5.(STA+LDA+ADD+OR+AND)

carga RDM = t6.STA

Read = t1 + t4.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) +
t6.(LDA+ADD+OR+AND)

Write = t7.STA

UAL(Y) = t7.LDA

UAL(ADD) = t7.ADD

UAL(OR) = t7.OR

UAL(AND) = t7.AND

UAL(NOT) = t3.NOT

carga AC = t7.(LDA+ADD+OR+AND) + t3.NOT

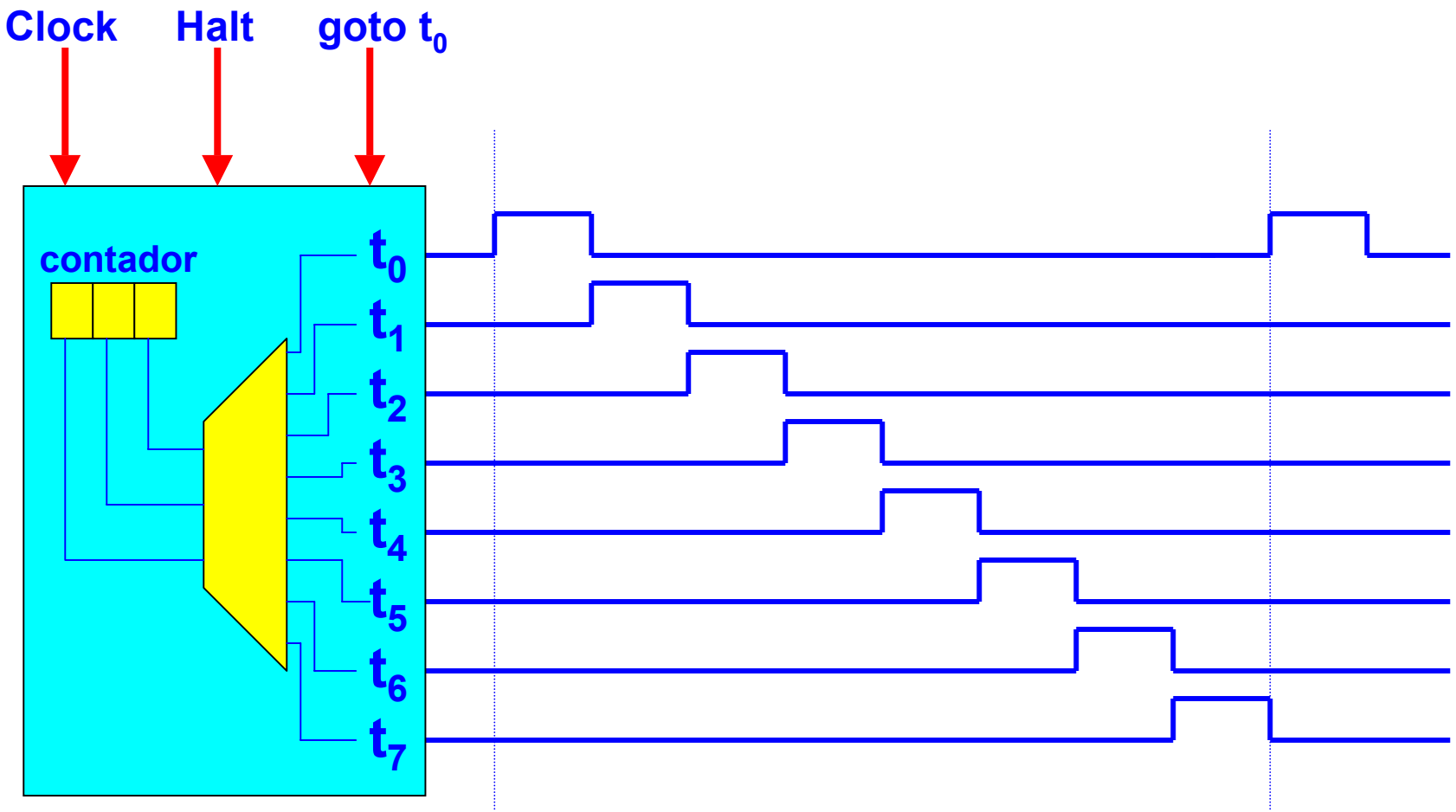
carga NZ = t7.(LDA+ADD+OR+AND) + t3.NOT = carga AC

carga PC = t5.(JMP+JN.N+JZ.Z)

goto t0 = t7.(STA+LDA+ADD+OR+AND) + t3.(NOP+NOT+JN.N'+JZ.Z') +
t5.(JMP+JN.N+JZ.Z)

Circuito de temporização dos ciclos de UCP

(alternativa 1: usando um contador de 3 bits)



Circuito com contador

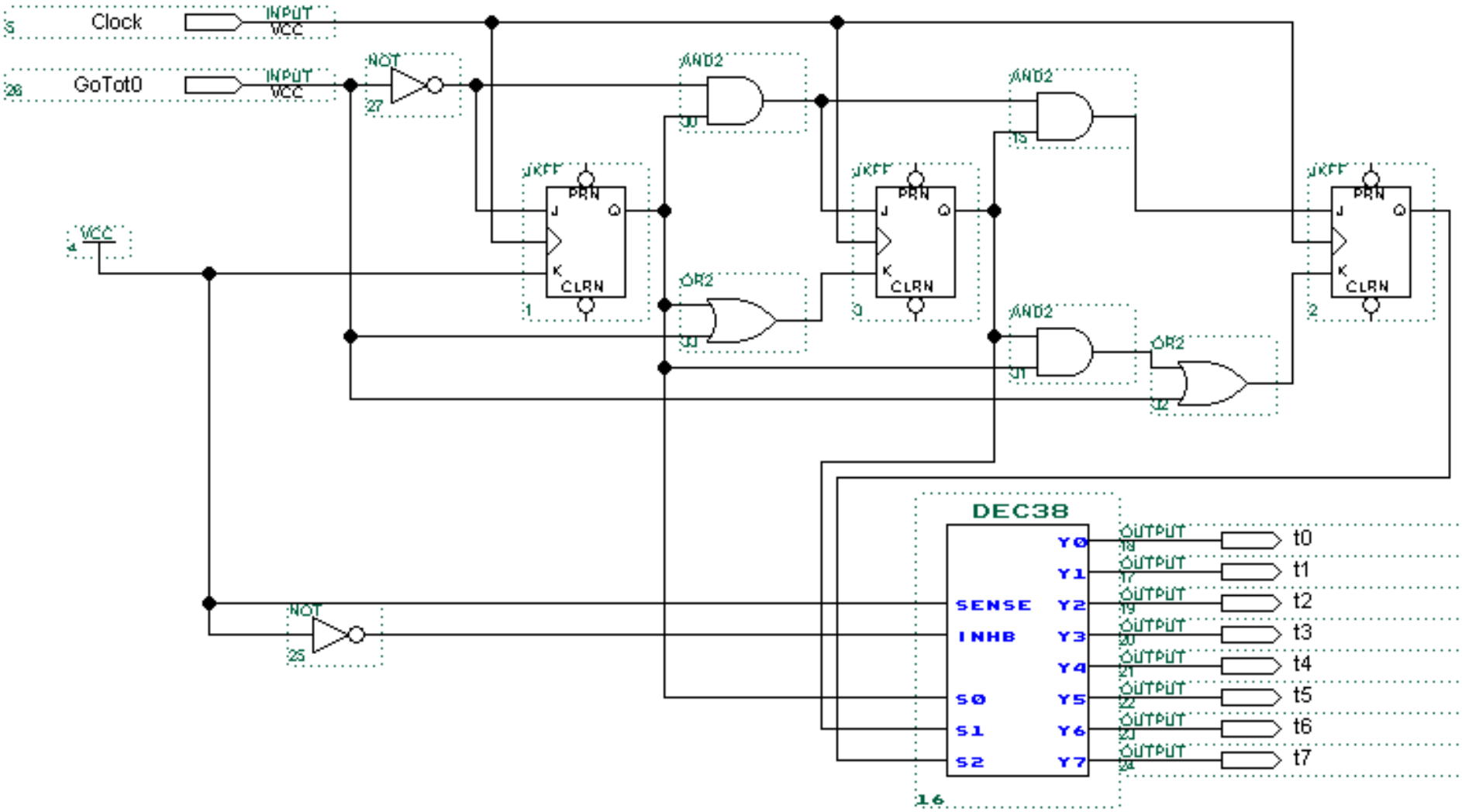
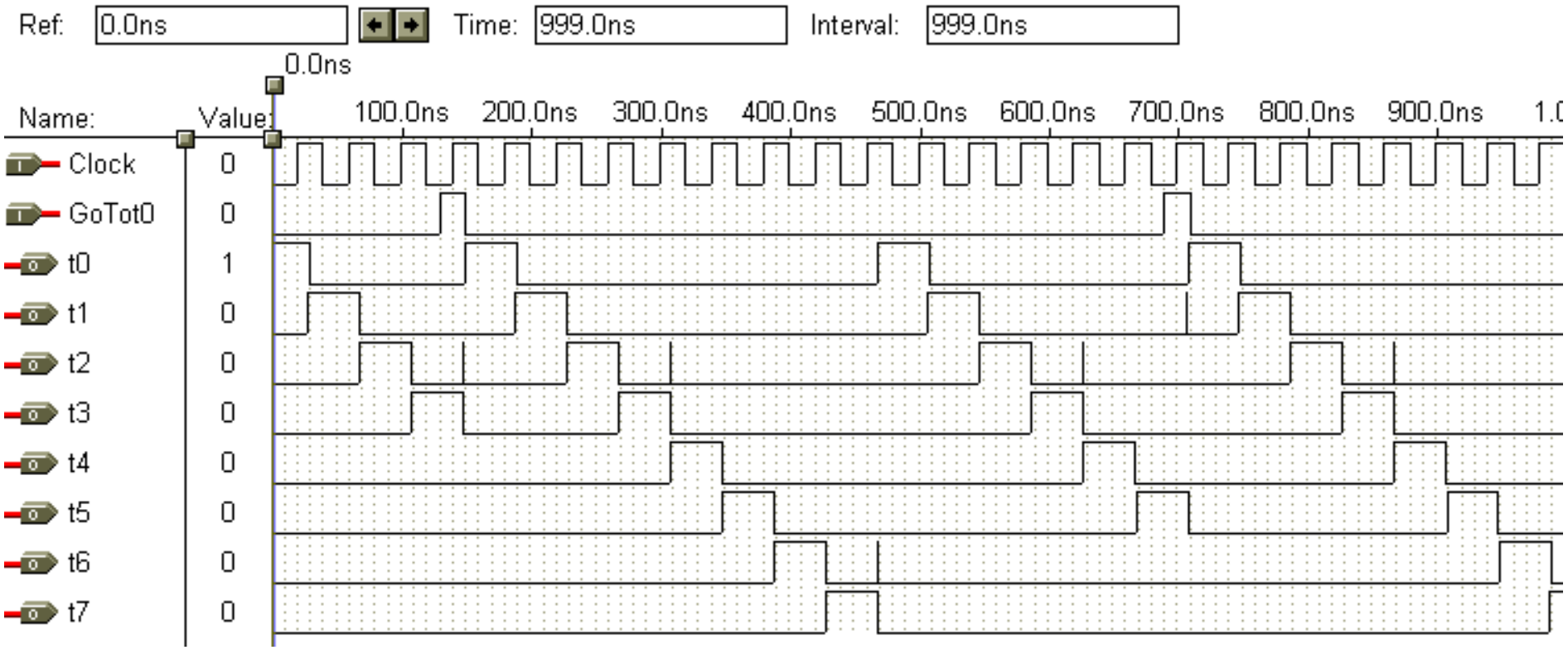
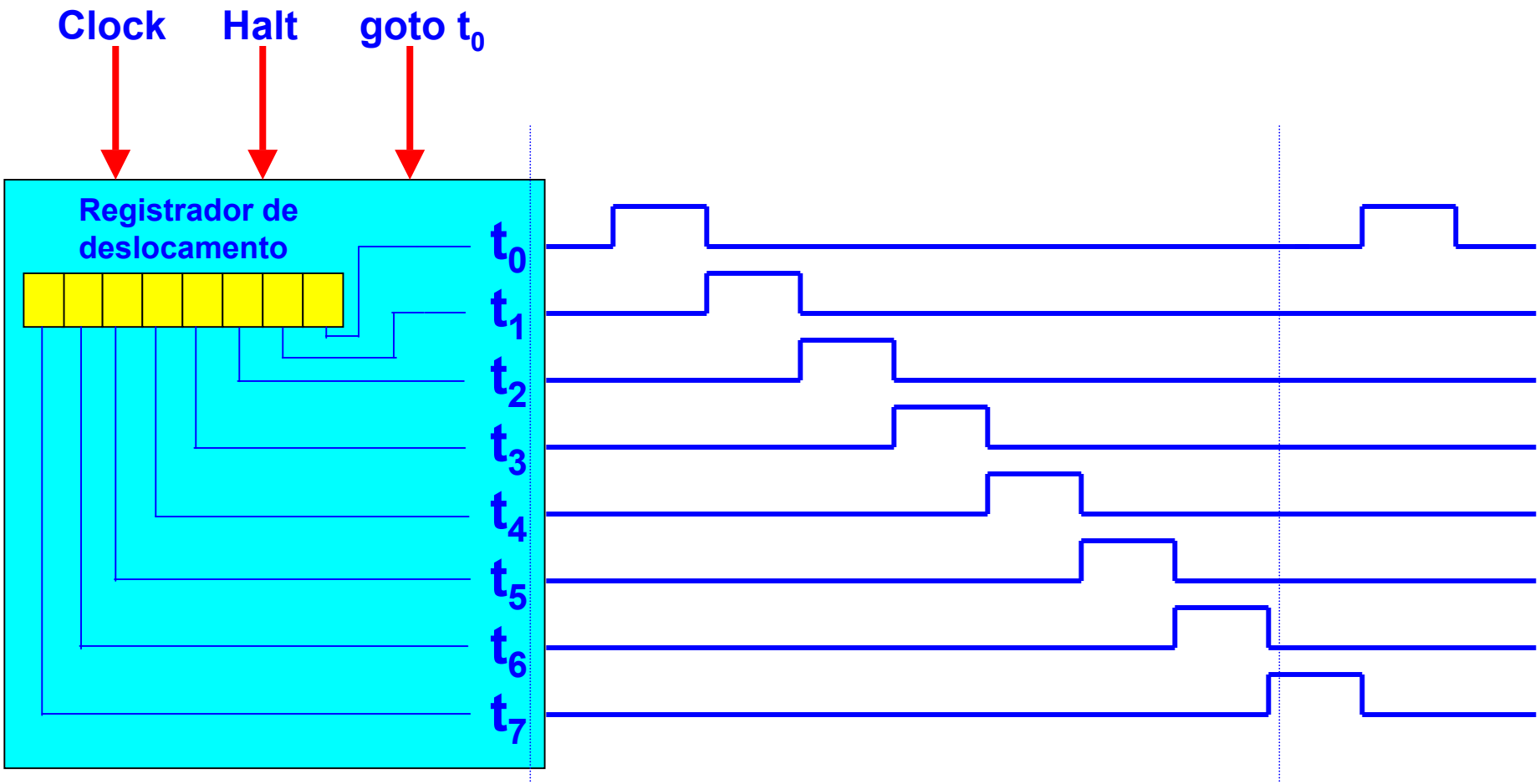


Diagrama de tempos com contador



Circuito de temporização dos ciclos de UCP

(alternativa 2: usando um registrador de deslocamento de 8 bits)



Circuito com registrador de deslocamento

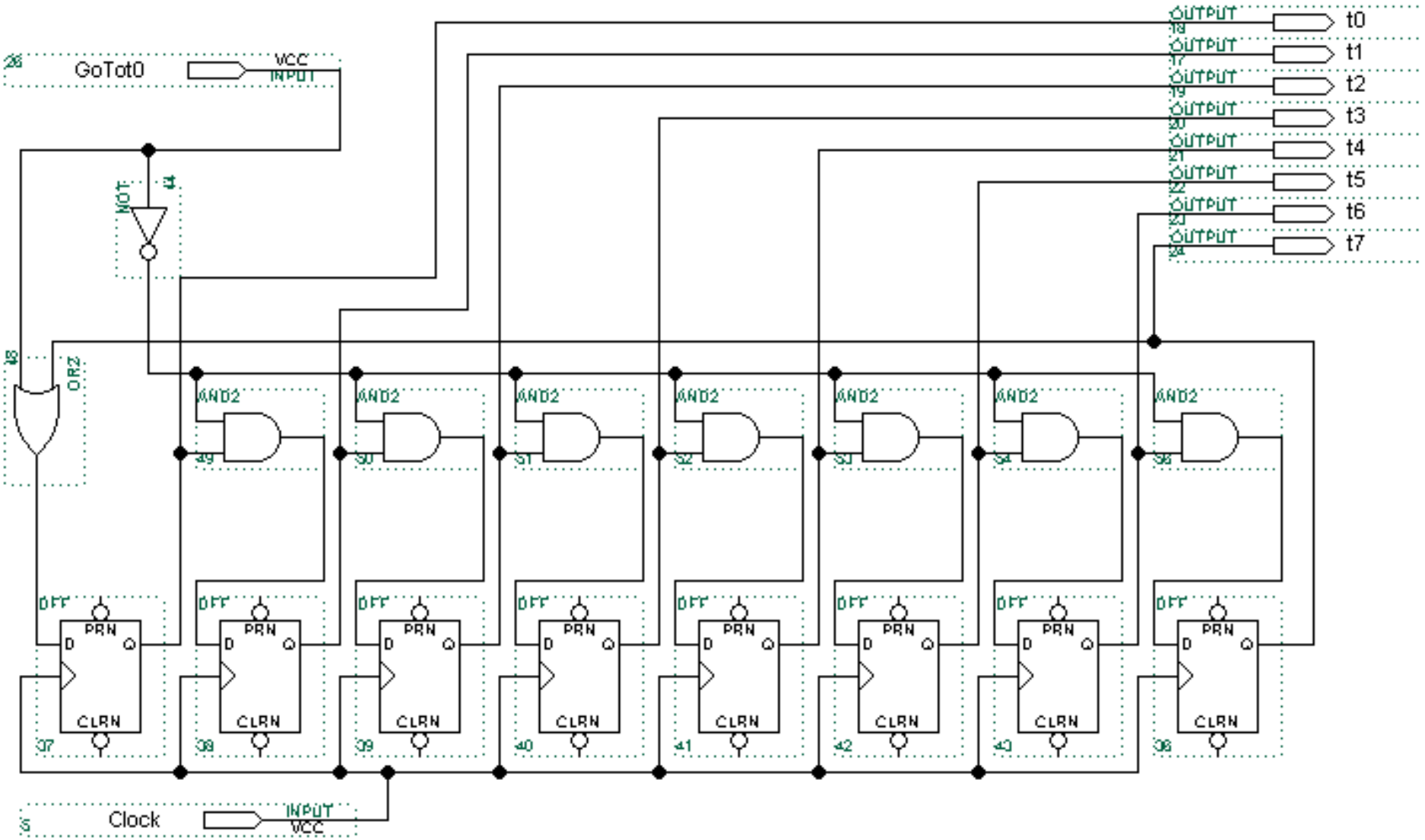
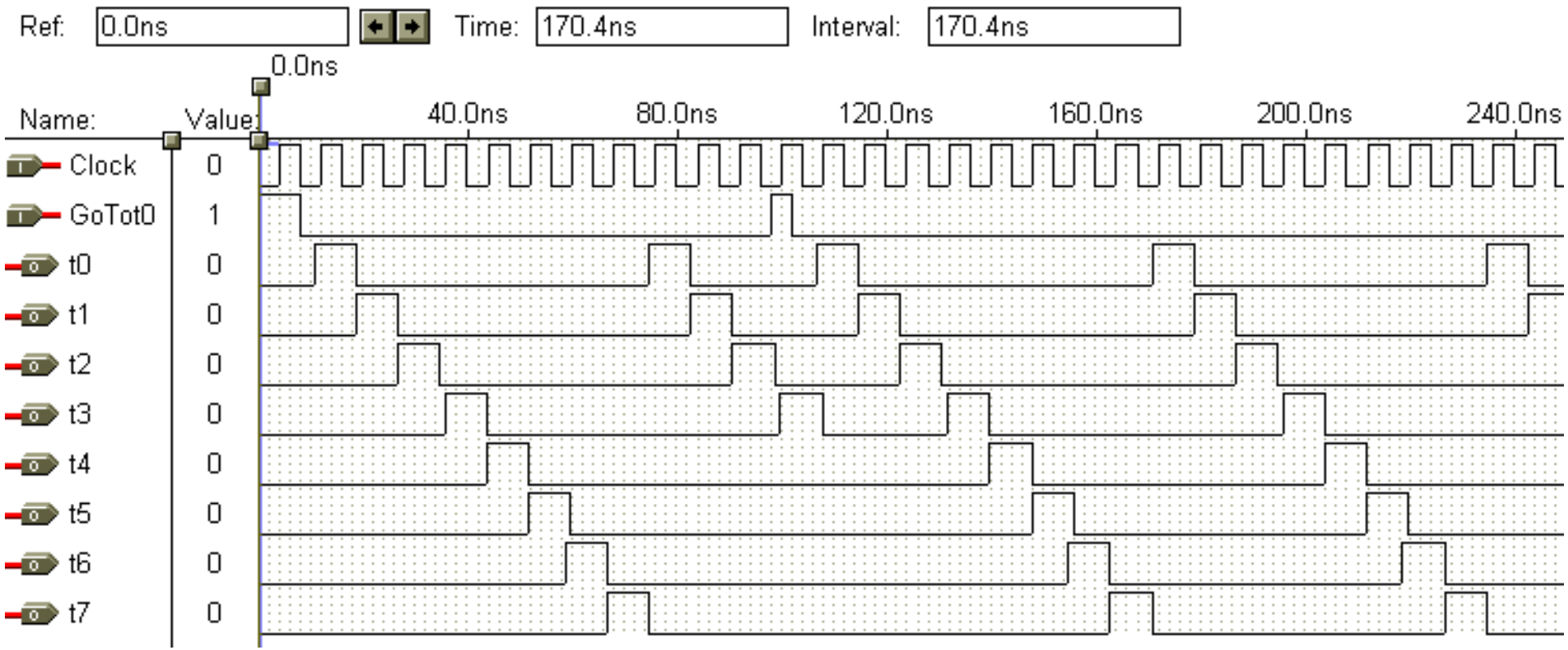
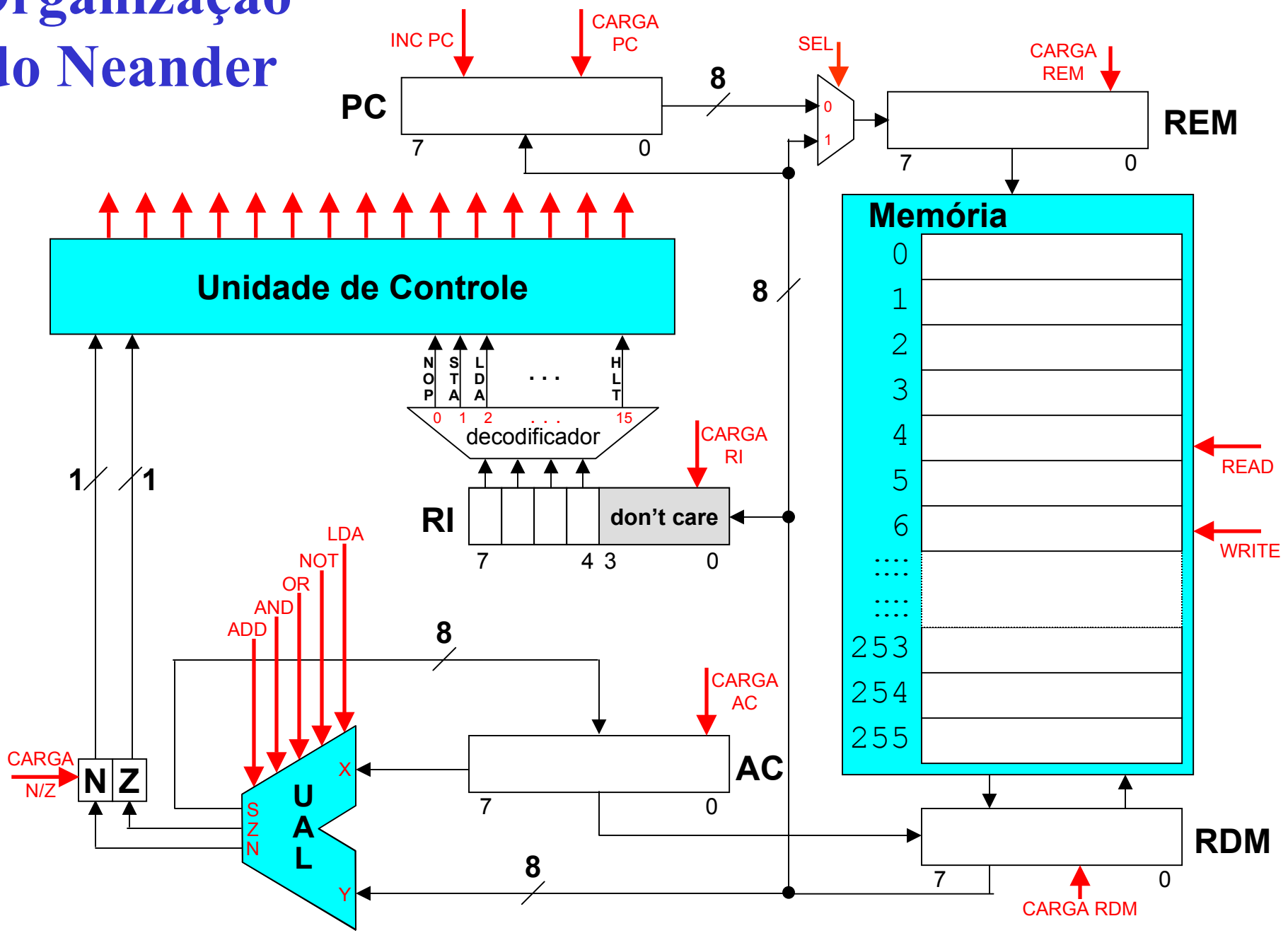


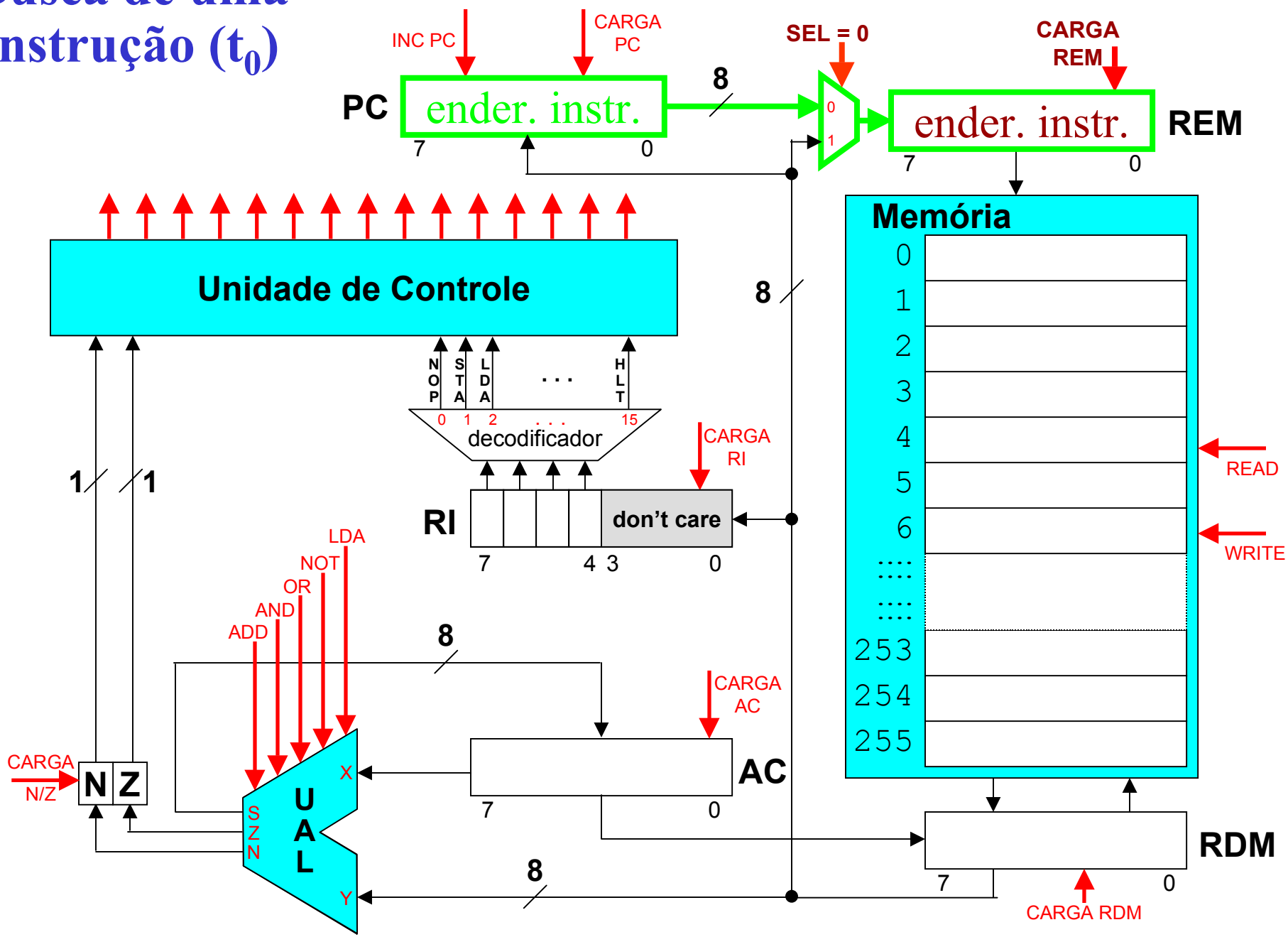
Diagrama de tempos com registrador de deslocamento



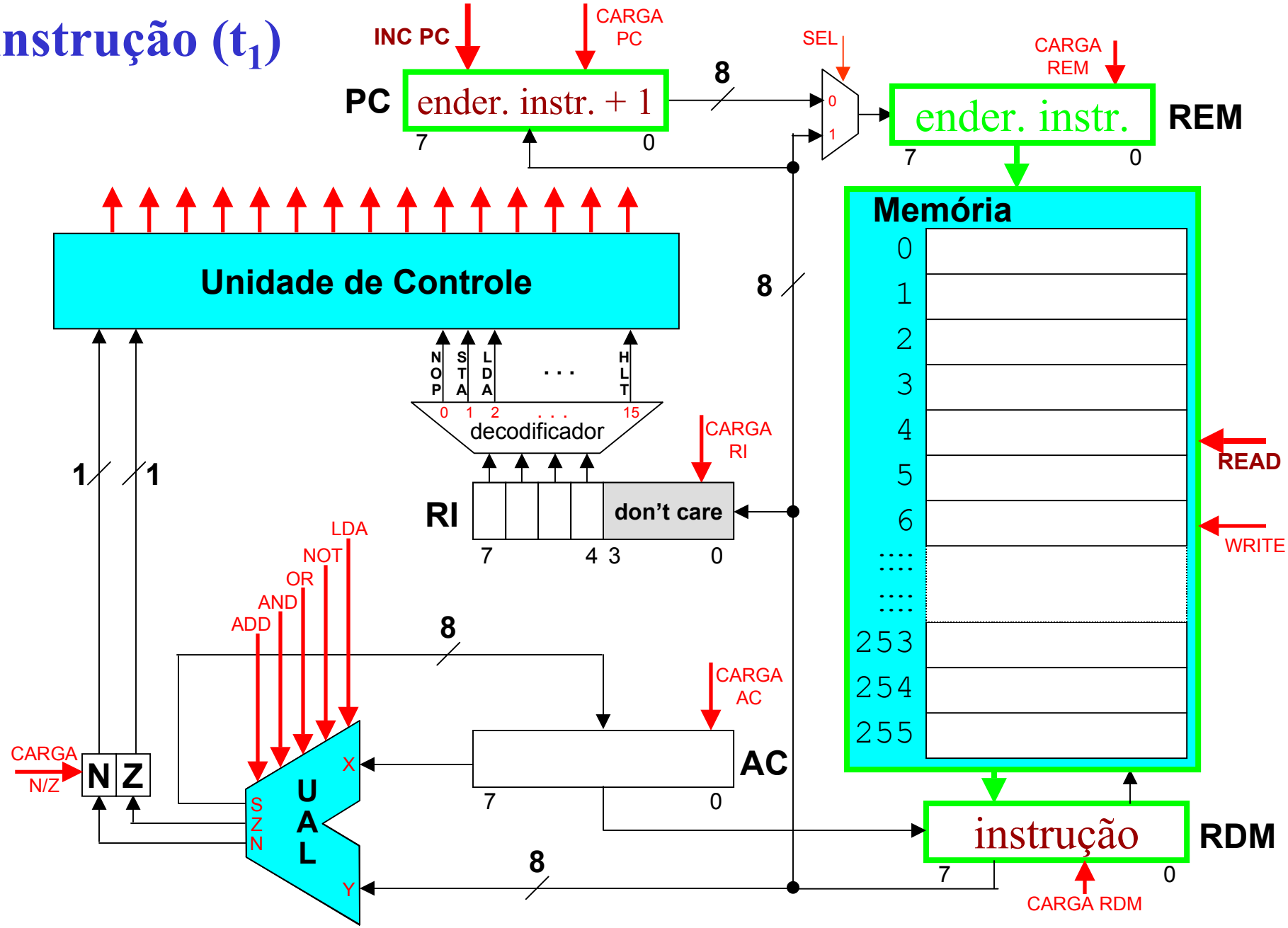
Organização do Neander



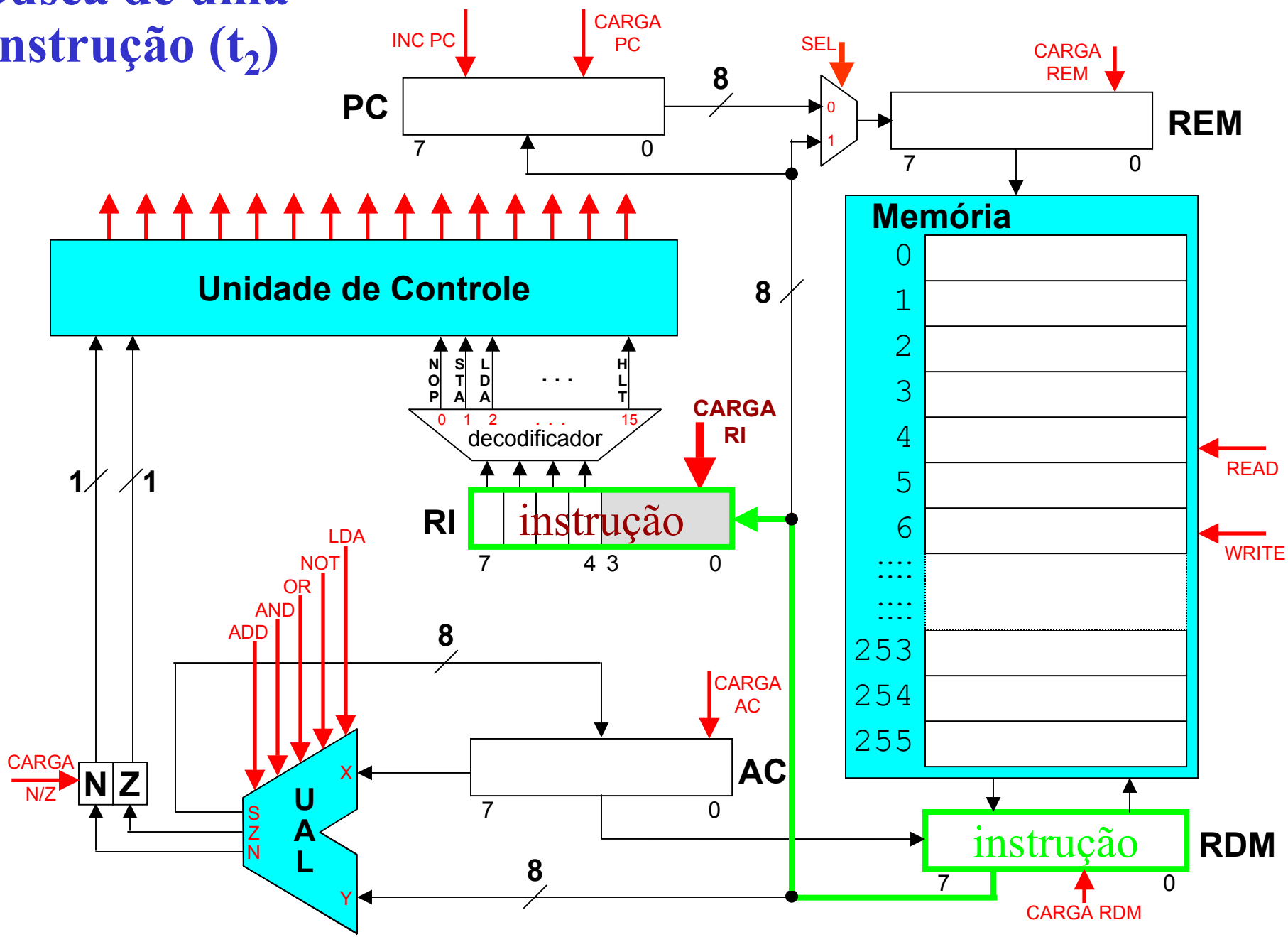
Busca de uma instrução (t_0)



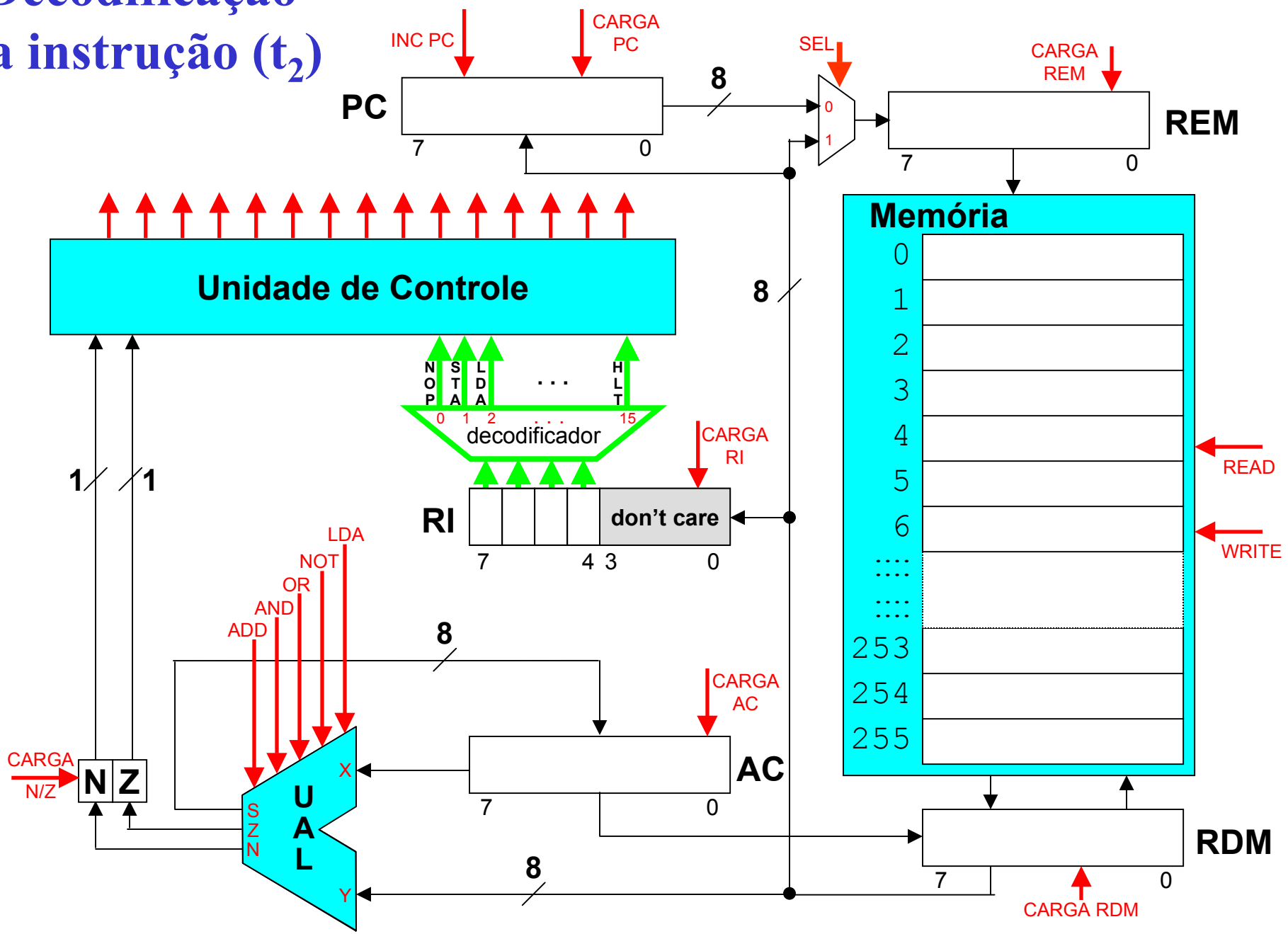
Busca de uma instrução (t_1)



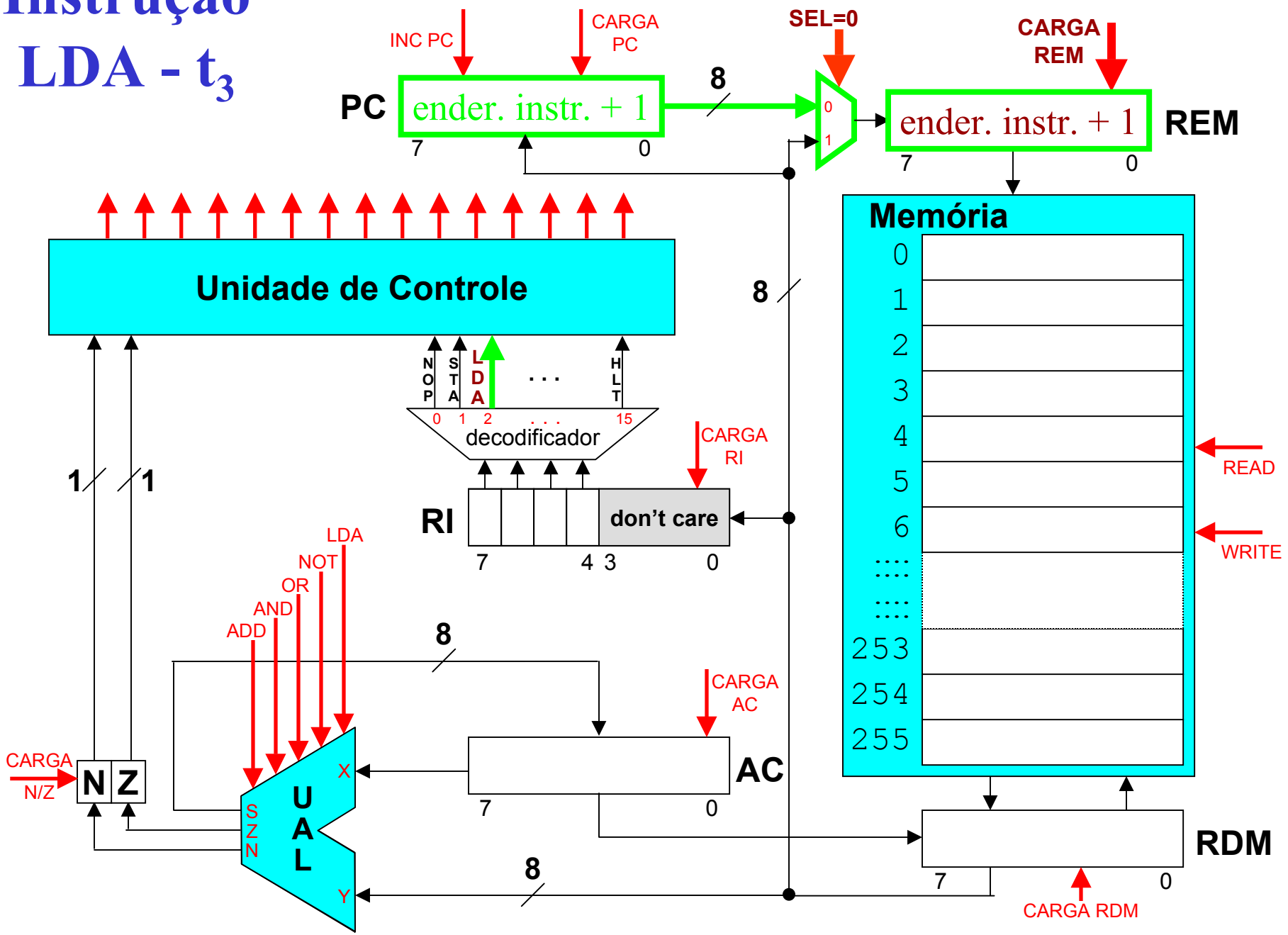
Busca de uma instrução (t_2)



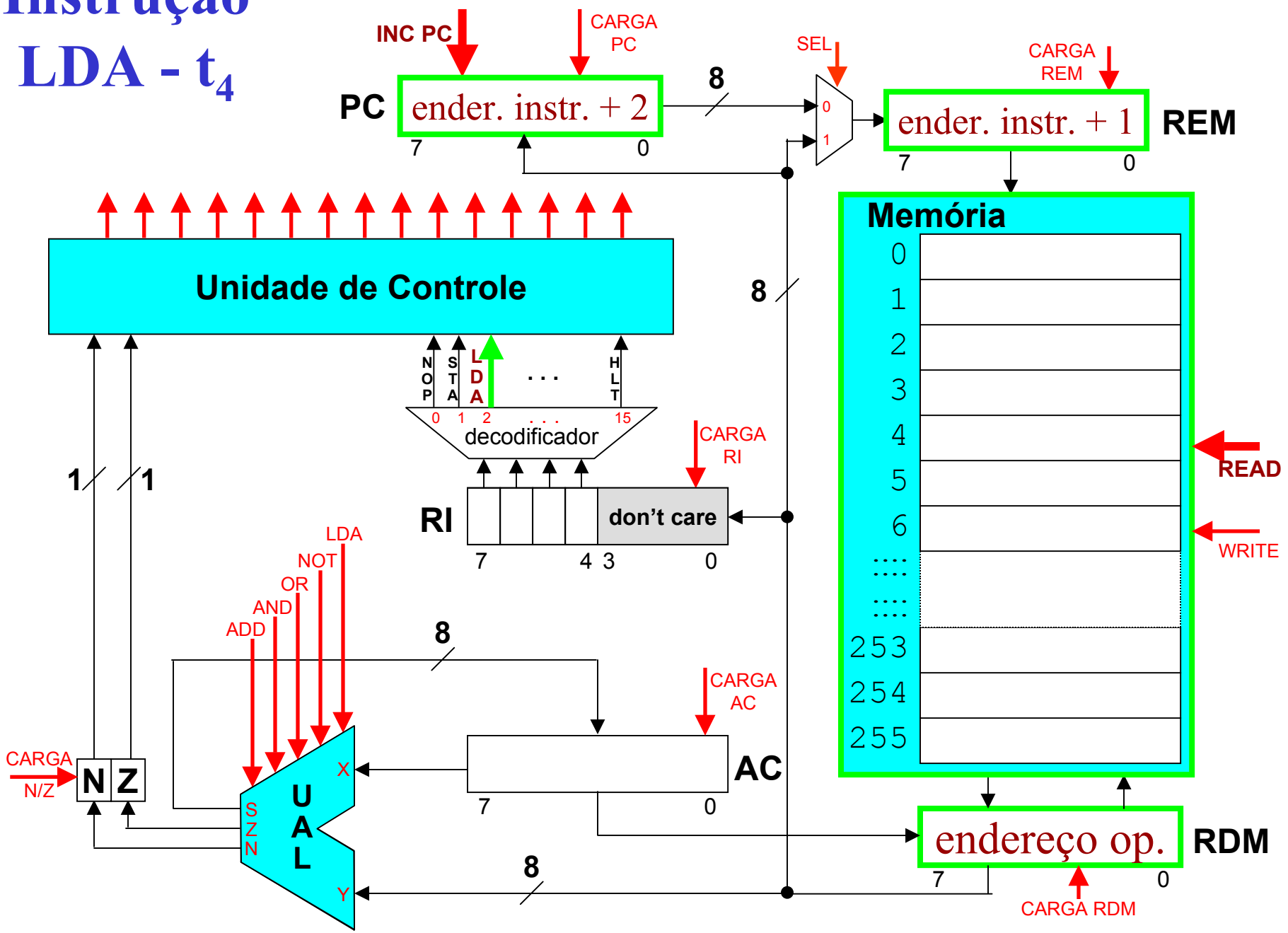
Decodificação da instrução (t_2)



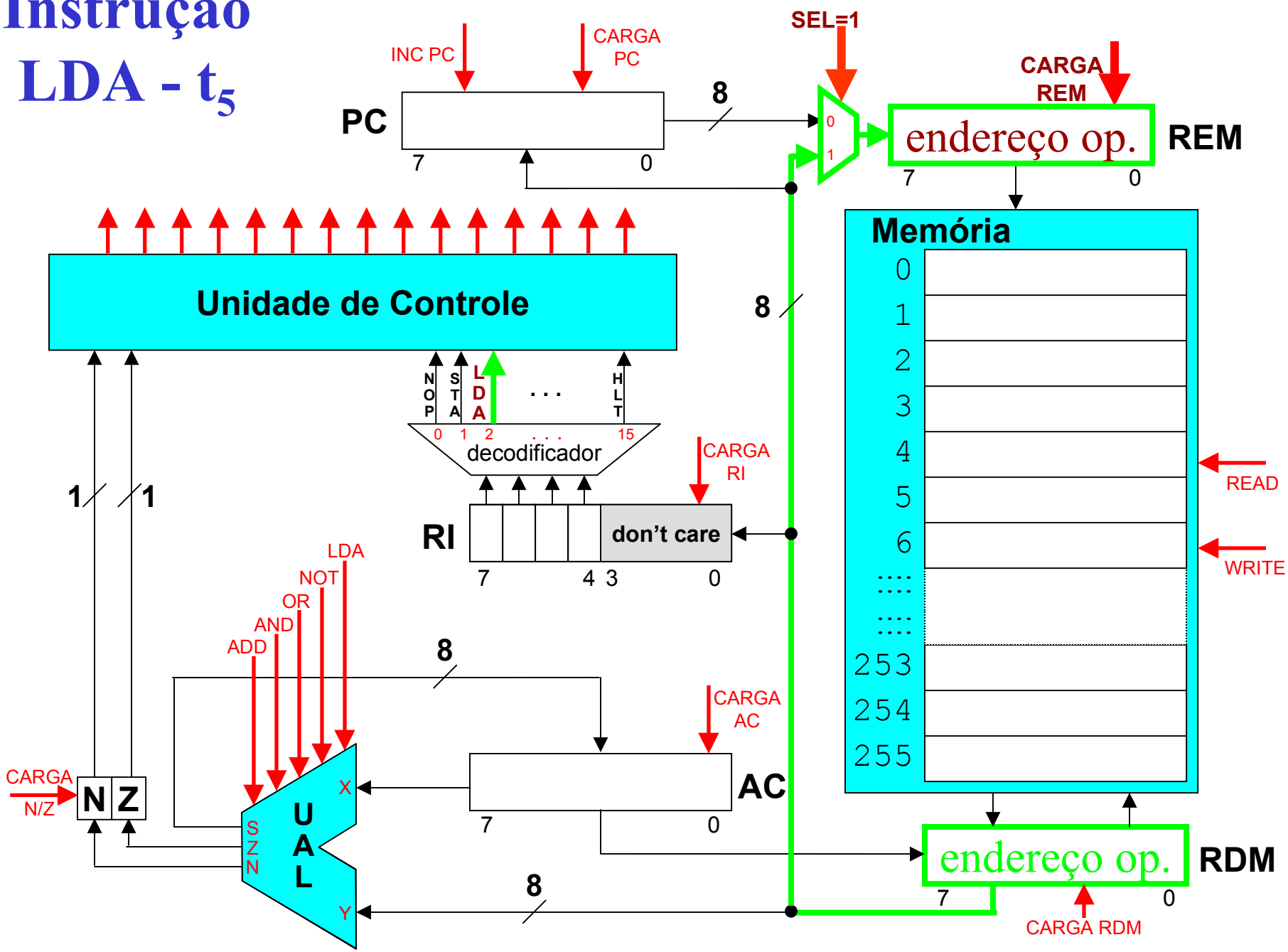
Instrução LDA - t_3



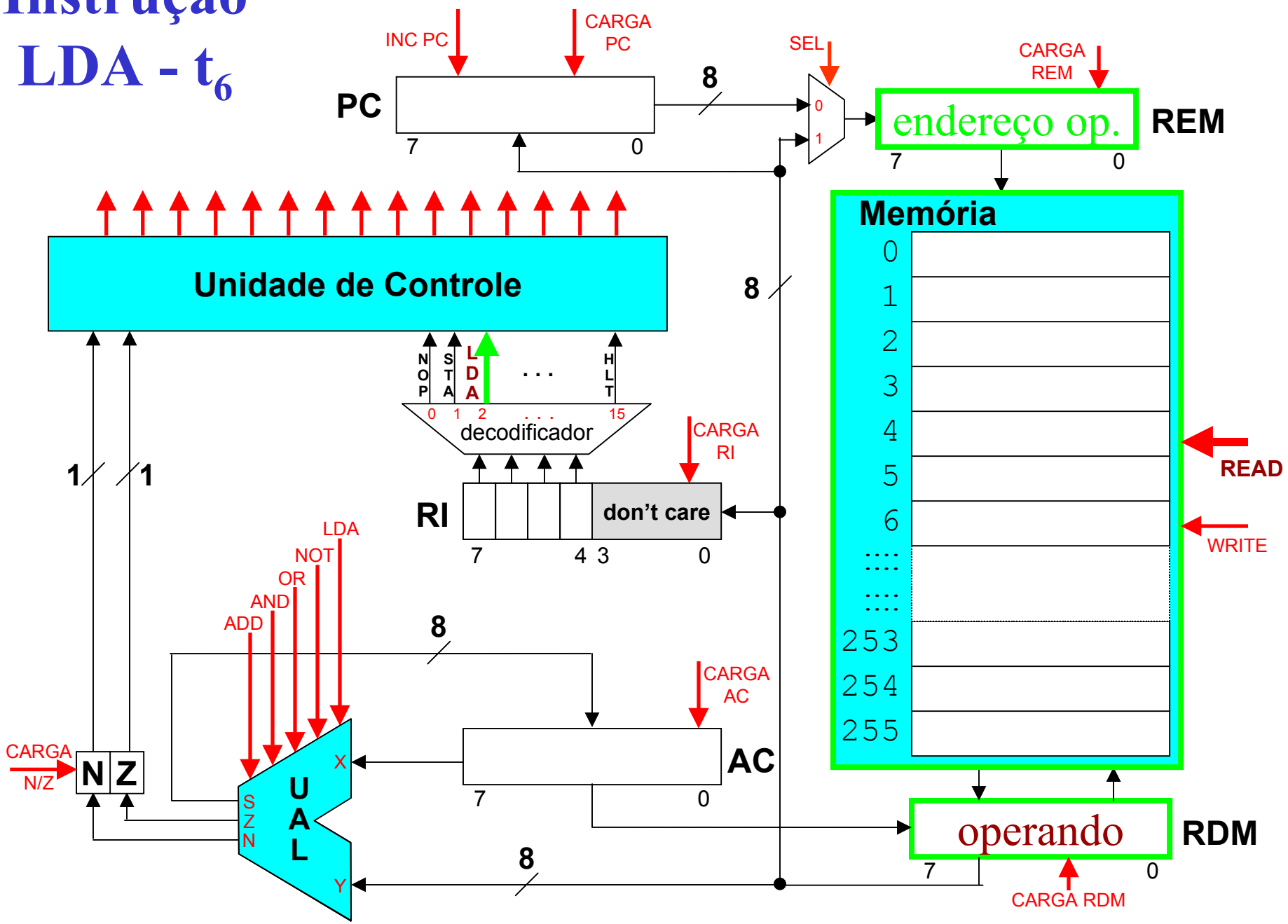
Instrução LDA - t_4



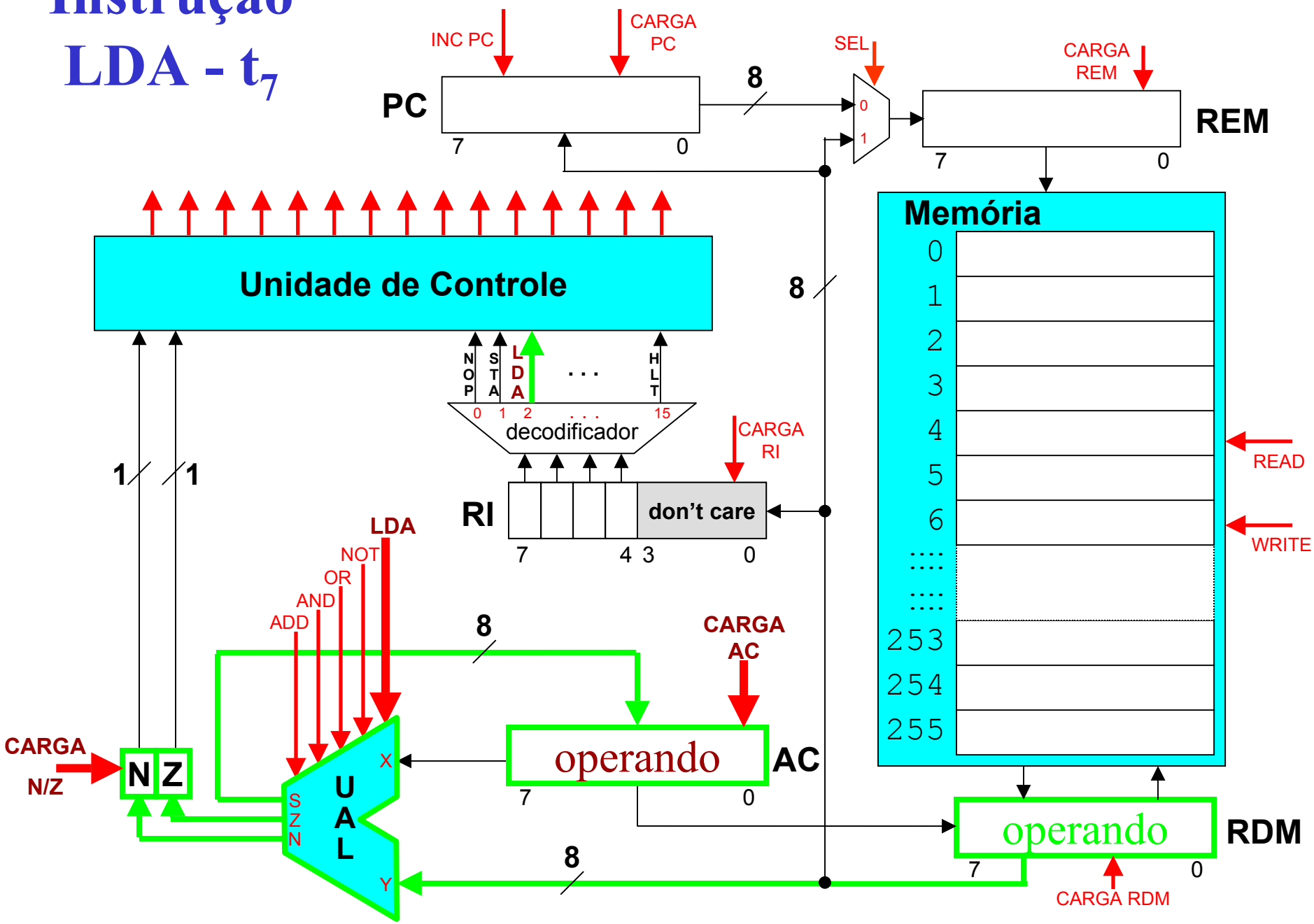
Instrução LDA - t_5



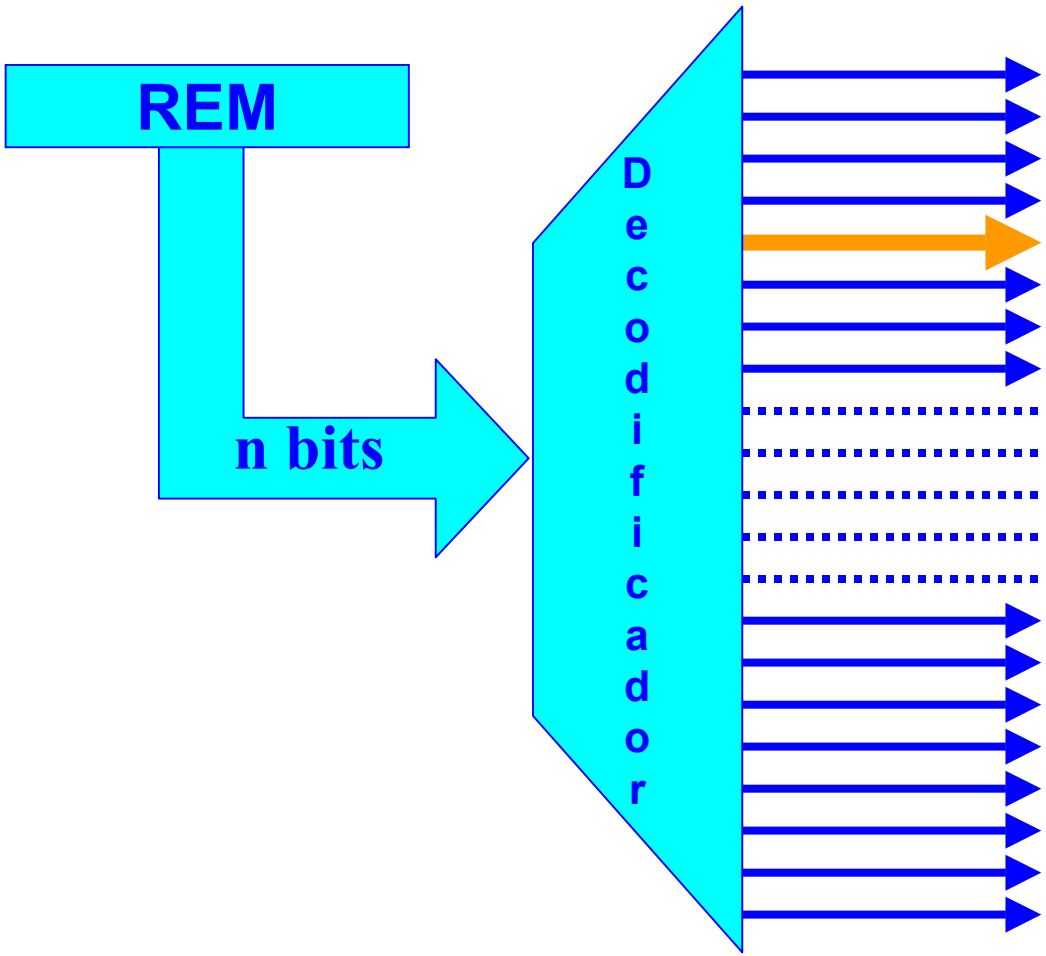
Instrução LDA - t_6



Instrução LDA - t_7



Memória com “seleção linear”

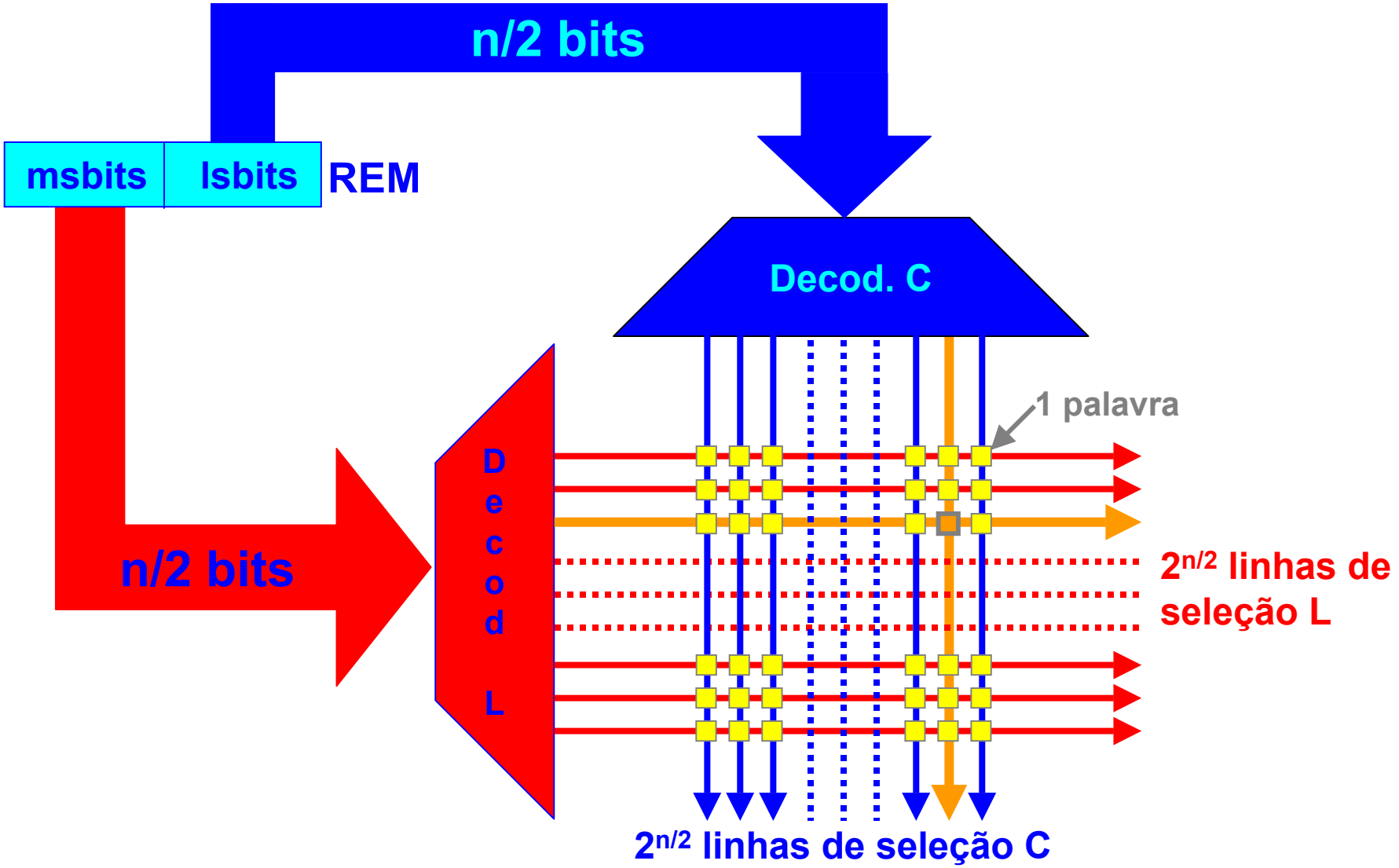


2^n linhas de seleção

(uma por palavra)

REM	8 bits	16 bits	20 bits	24 bits	32 bits
Linhas	256	65.536	1.048.576	16.777.216	4.294.967.296

Memória com "seleção matricial"

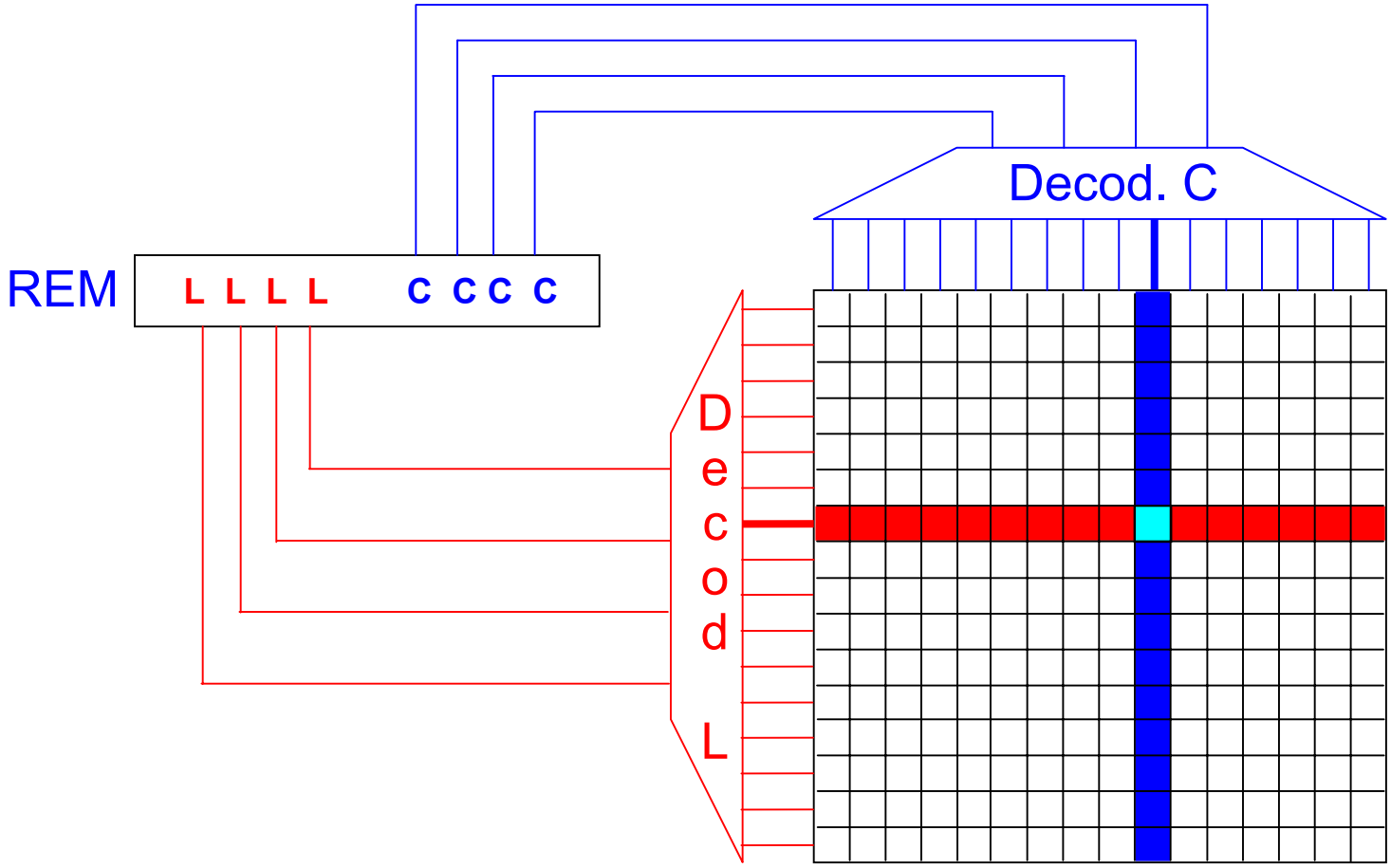


REM	8 bits	16 bits	20 bits	24 bits	32 bits
Linhas	32	512	2.048	8.192	131.072

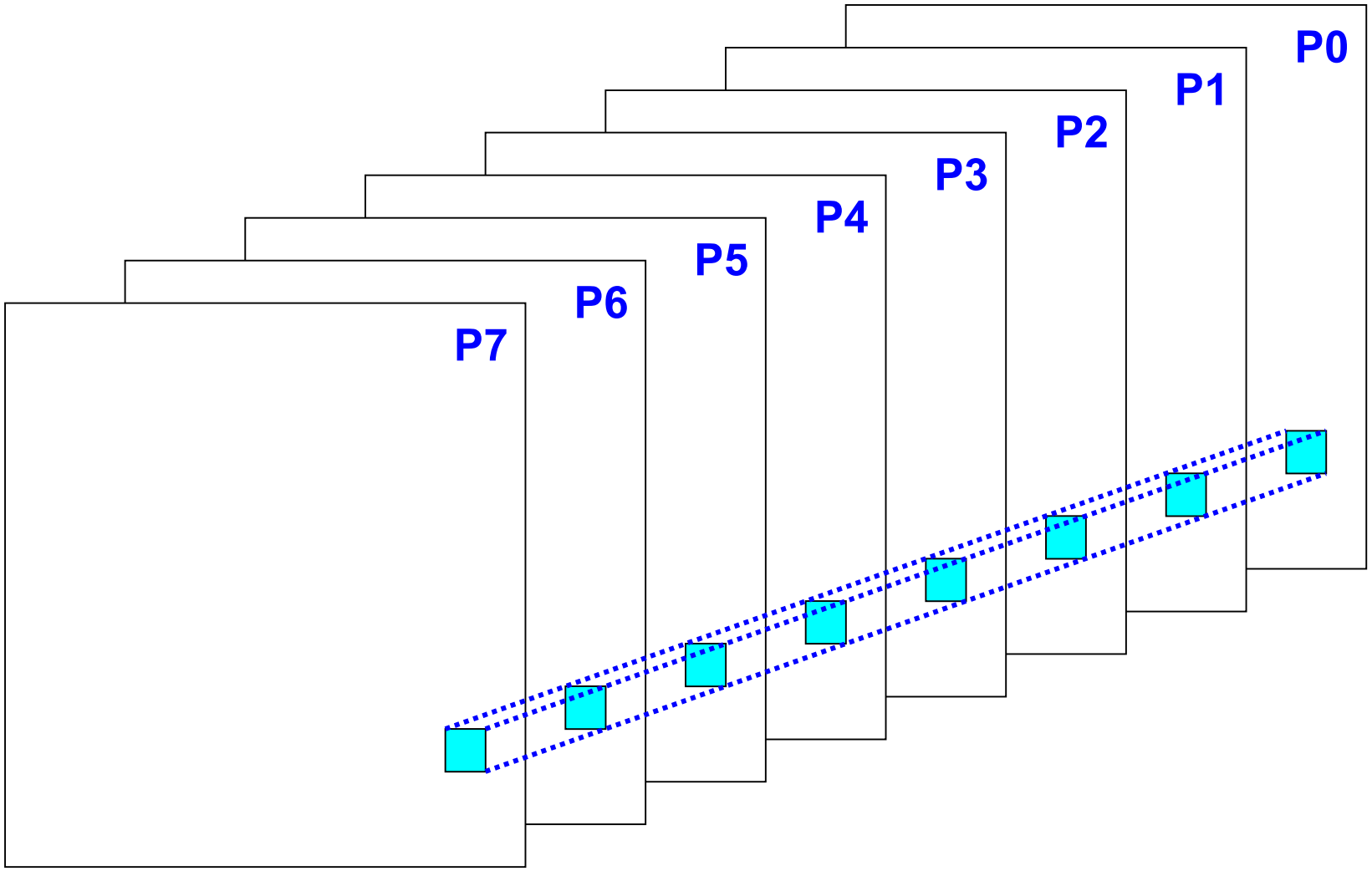
Organização de um “plano” de memória

CCCC LLLL	0000	0001	0010	0011	0100	0101	0110	0111	1000	**1001**	1010	1011	1100	1101	1110	1111
0000	b000	b016	b032	b048	b064	b080	b096	b112	b128	**b144**	b160	b176	b192	b208	b224	b240
0001	b001	b017	b033	b049	b065	b081	b097	b113	b129	**b145**	b161	b177	b193	b209	b225	b241
0010	b002	b018	b034	b050	b066	b082	b098	b114	b130	**b146**	b162	b178	b184	b210	b226	b242
0011	b003	b019	b035	b051	b067	b083	b099	b115	b131	**b147**	b163	b179	b195	b211	b227	b243
0100	b004	b020	b036	b052	b068	b084	b100	b116	b132	**b148**	b164	b180	b196	b212	b228	b244
0101	b005	b021	b037	b053	b069	b085	b101	b117	b133	**b149**	b165	b181	b197	b213	b229	b245
0110	**b006**	**b022**	**b038**	**b054**	**b070**	**b086**	**b102**	**b118**	**b134**	**b150**	**b166**	**b182**	**b198**	**b214**	**b230**	**b246**
0111	b007	b023	b039	b055	b071	b087	b103	b119	b135	**b151**	b167	b183	b199	b215	b231	b247
1000	b008	b024	b040	b056	b072	b088	b104	b120	b136	**b152**	b168	b184	b200	b216	b232	b248
1001	b009	b025	b041	b057	b073	b089	b105	b121	b137	b153	b169	b185	b201	b217	b233	b249
1010	b010	b026	b042	b058	b074	b090	b106	b122	b138	**b154**	b170	b186	b202	b218	b234	b250
1011	b011	b027	b043	b059	b075	b091	b107	b123	b139	**b155**	b171	b187	b203	b219	b235	b251
1100	b012	b028	b044	b060	b076	b092	b108	b124	b140	**b156**	b172	b188	b204	b220	b236	b252
1101	b013	b029	b045	b061	b077	b093	b109	b125	b141	**b157**	b173	b189	b205	b221	b237	b253
1110	b014	b030	b046	b062	b078	b094	b110	b126	b142	**b158**	b174	b190	b206	b222	b238	b254
1111	b015	b031	b047	b063	b079	b095	b111	b127	b143	**b159**	b175	b191	b207	b223	b239	b255

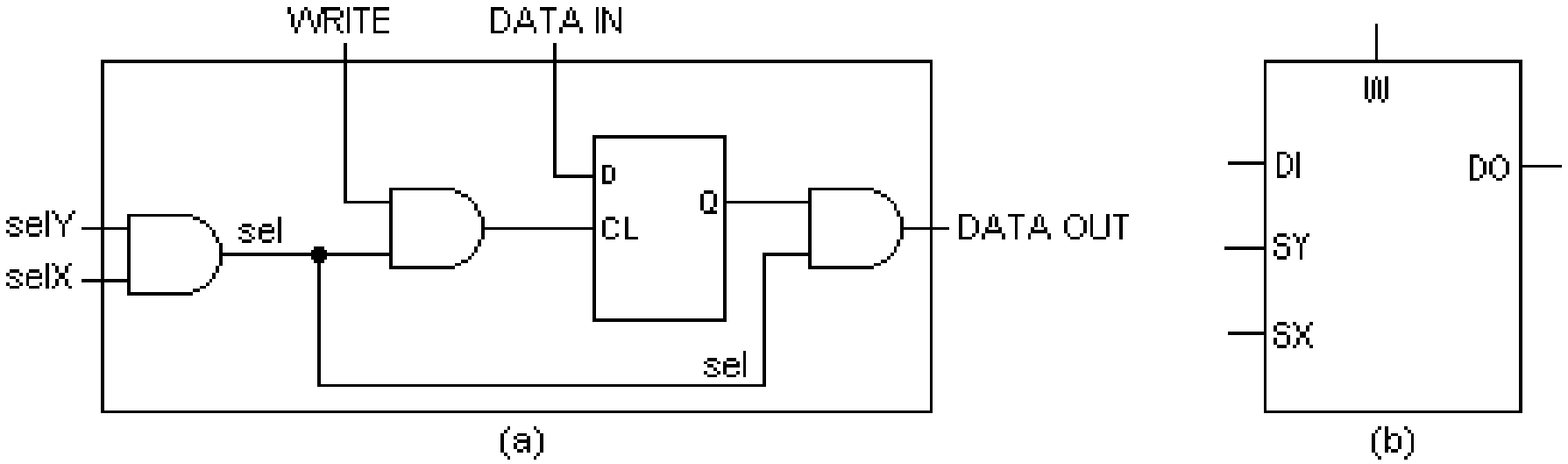
Decodificação do endereço contido no REM



Bits de uma palavra nos “planos” de memória



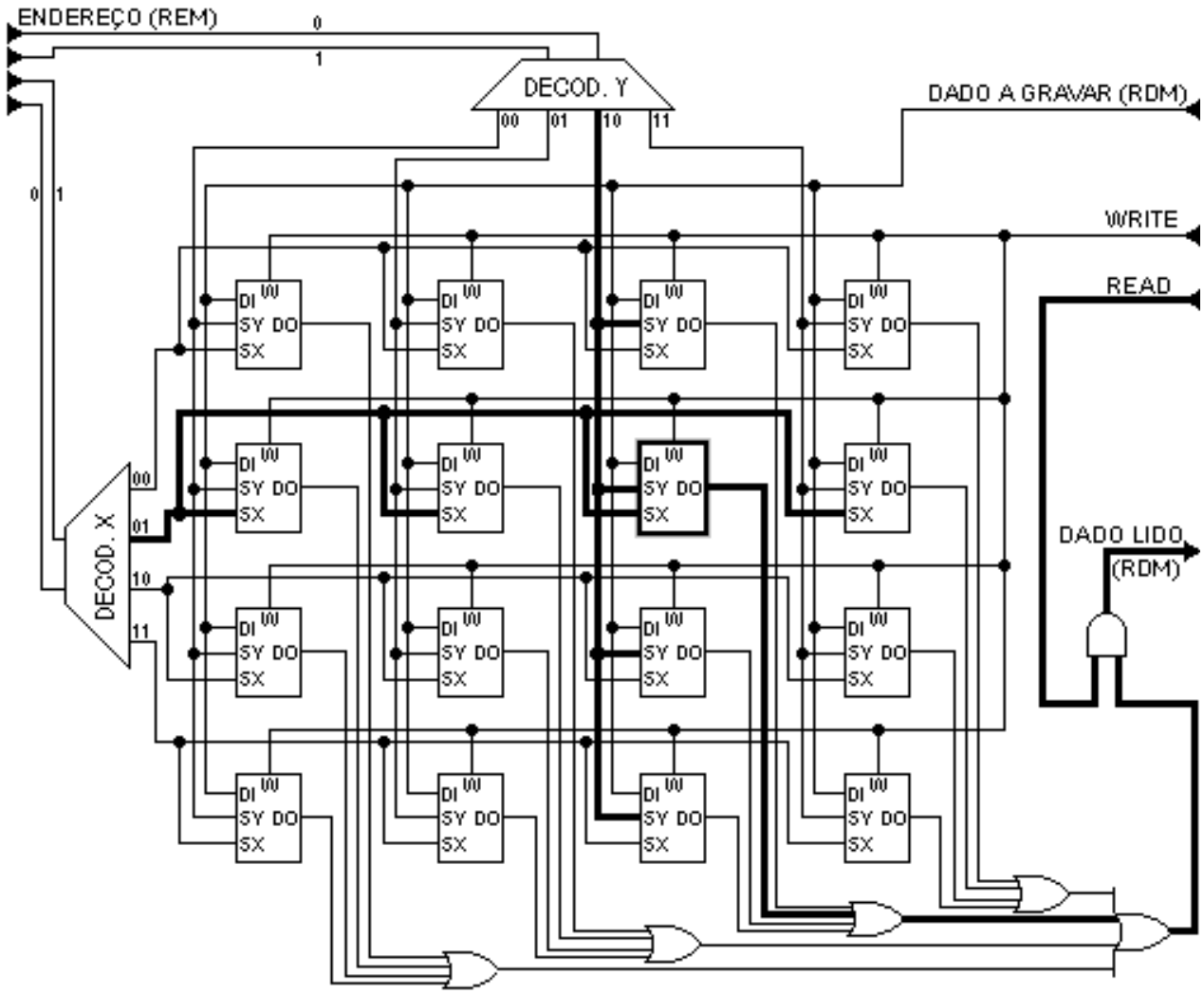
Célula de memória do Neander



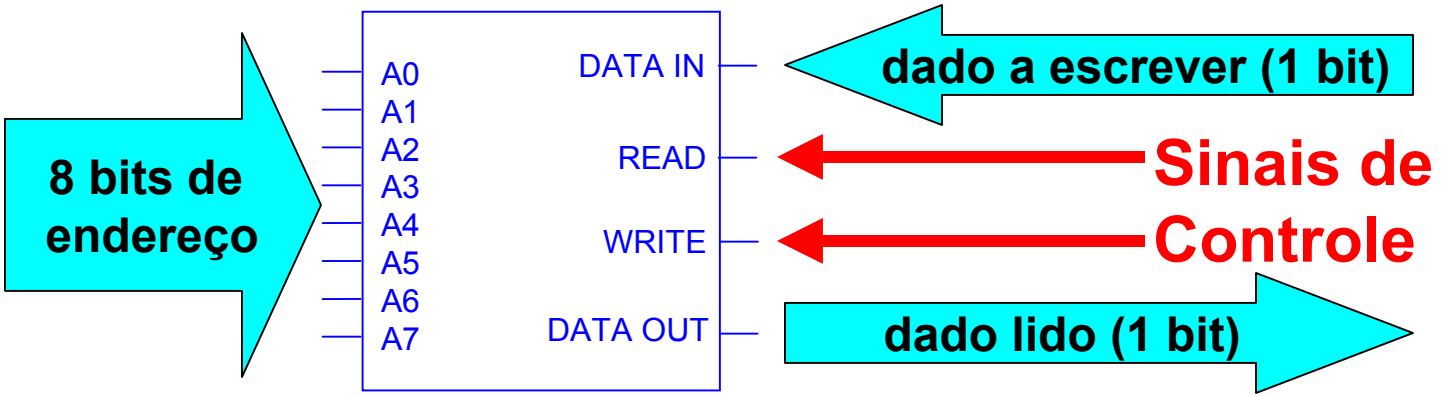
Plano de memória construído com as células

REM

0	1	1	0

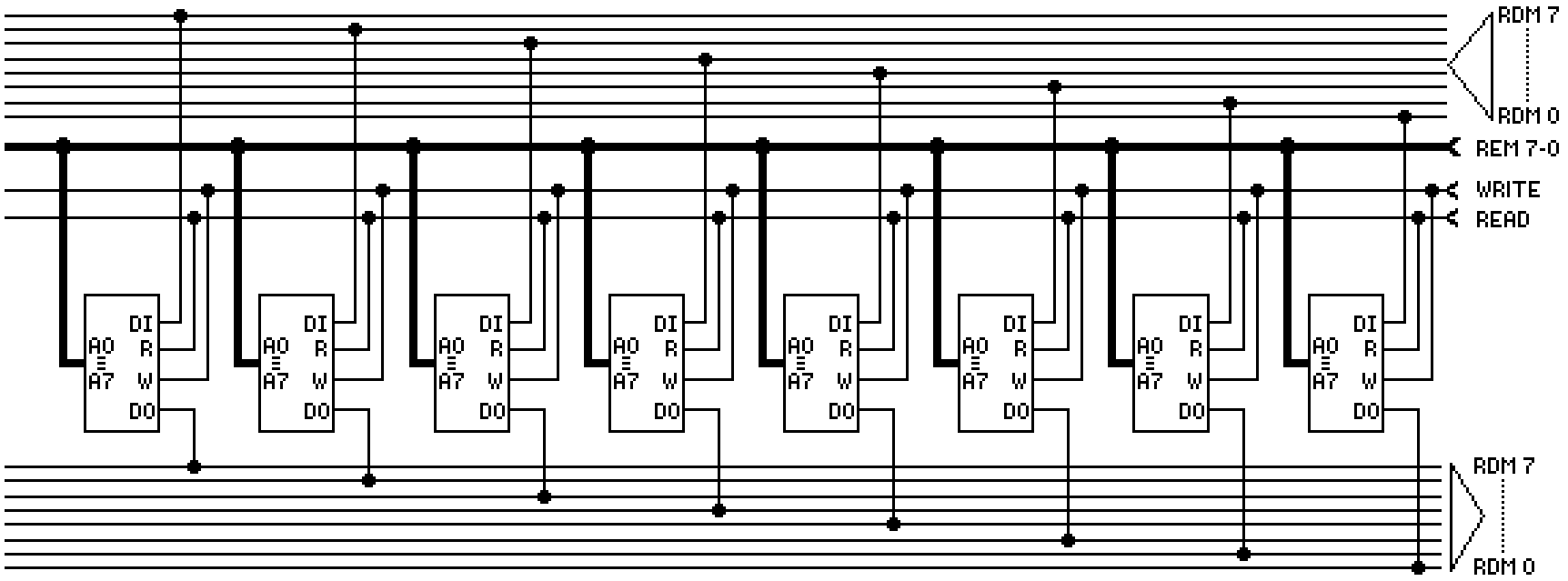


“CI” de memória para o Neander
(memória RAM de 256 bits x 1)

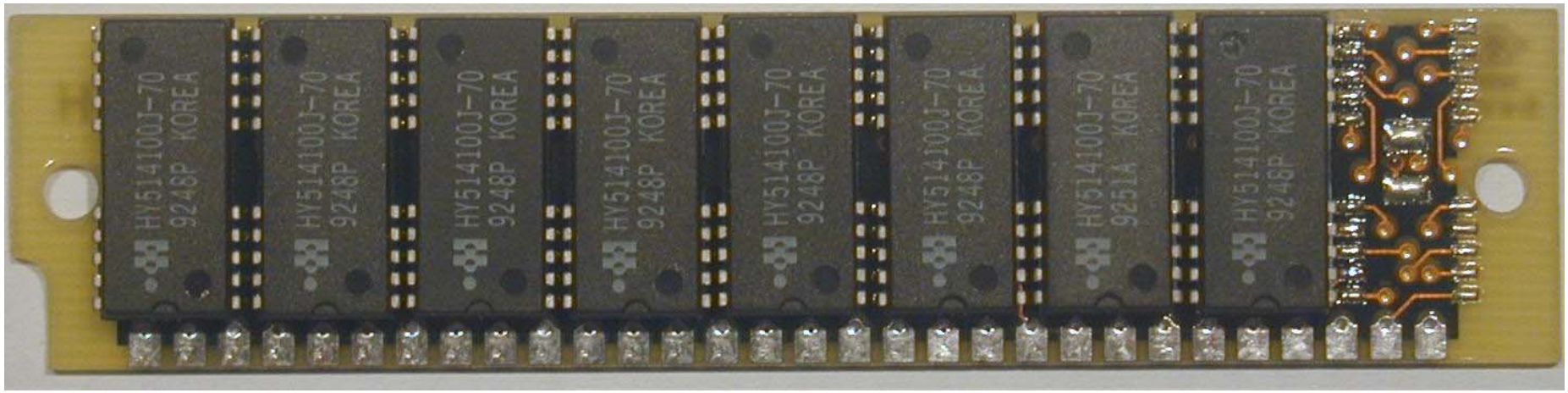


Nota histórica: o primeiro circuito integrado de memória RAM produzido em escala comercial foi o Intel 1103, em 1970, com 1 kbit x 1 (1024 bits).

Memória do Neander completa,
com 8 CIs de 256 bits x 1

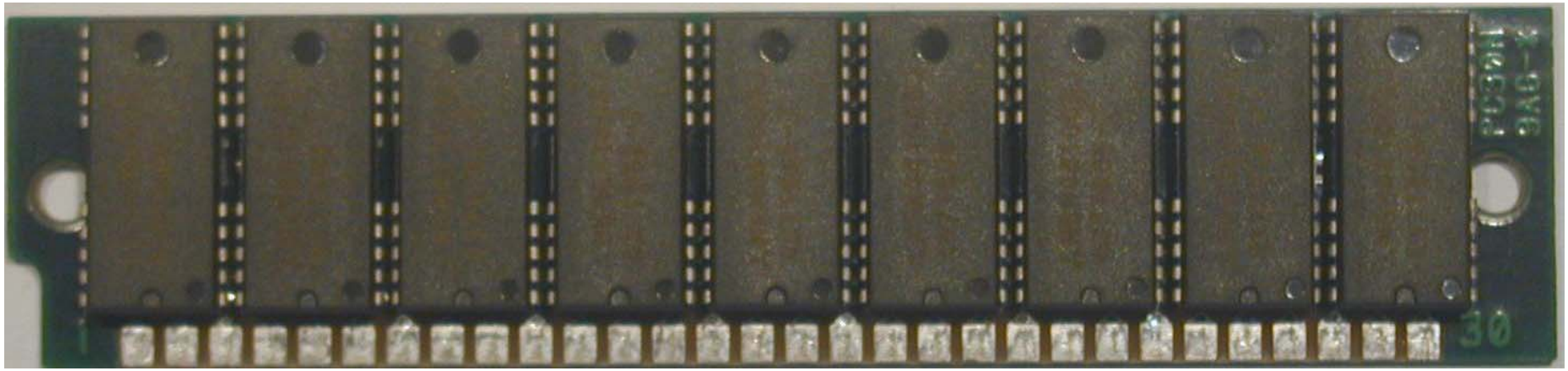


Módulo SIMM* de memória do Neander (sem bit de paridade)



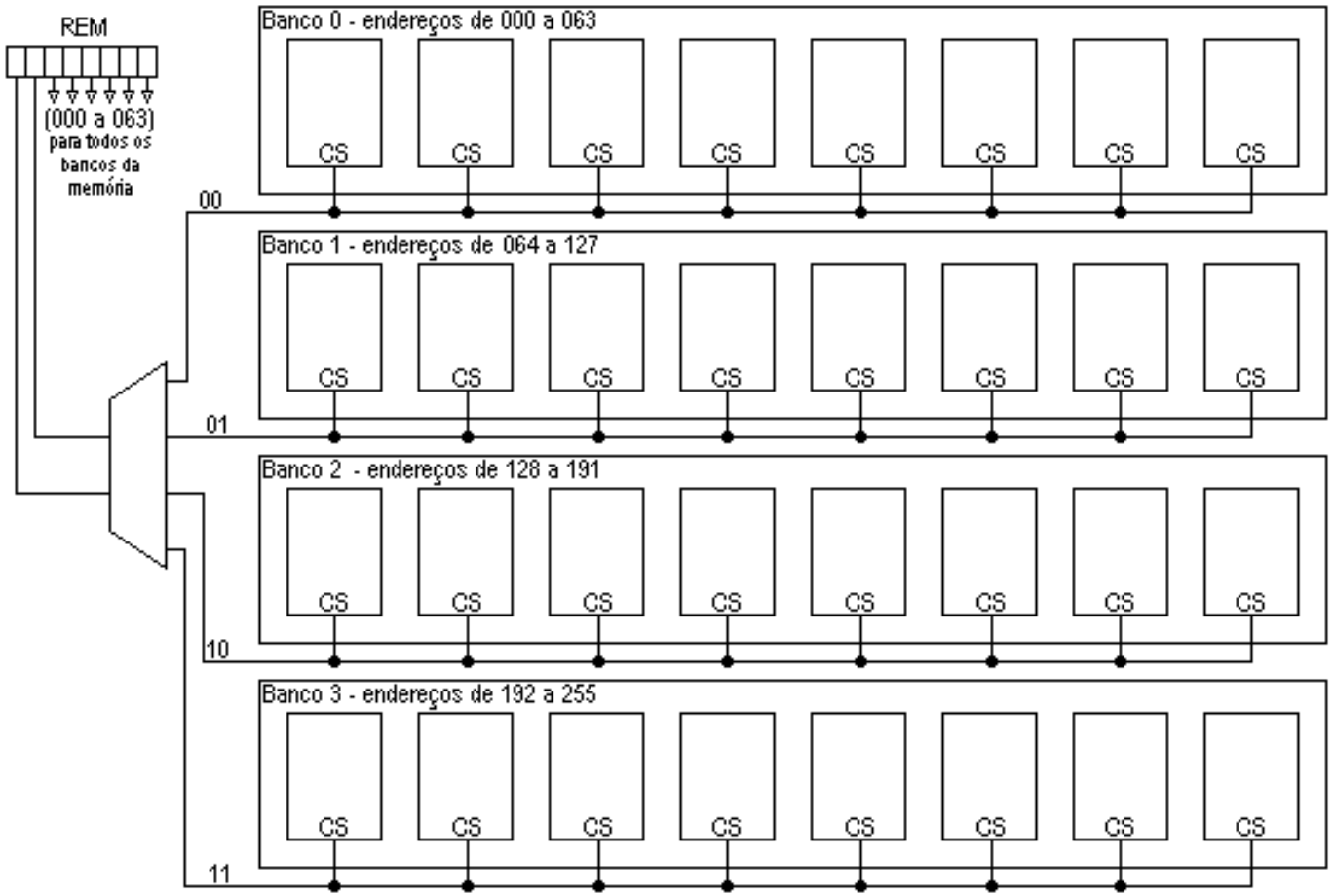
(*) Single Inline Memory Module ;-)

Módulo SIMM* de memória do Neander (com bit de paridade)

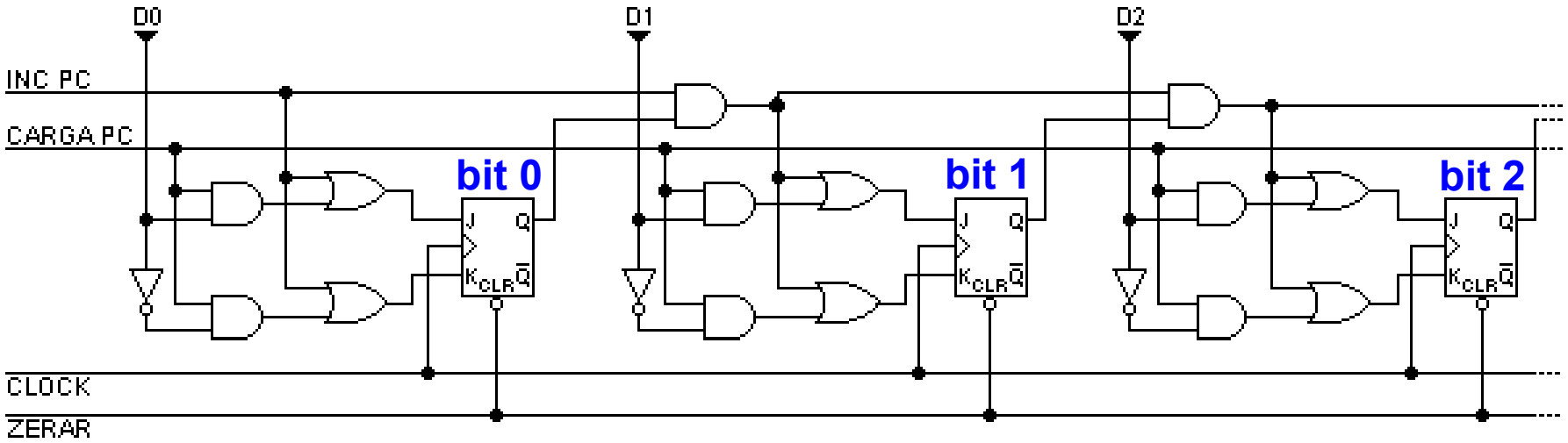


(*) Single Inline Memory Module ;-)

Memória do Neander com 4 “bancos” (usando CIs de 64 bits x 1)



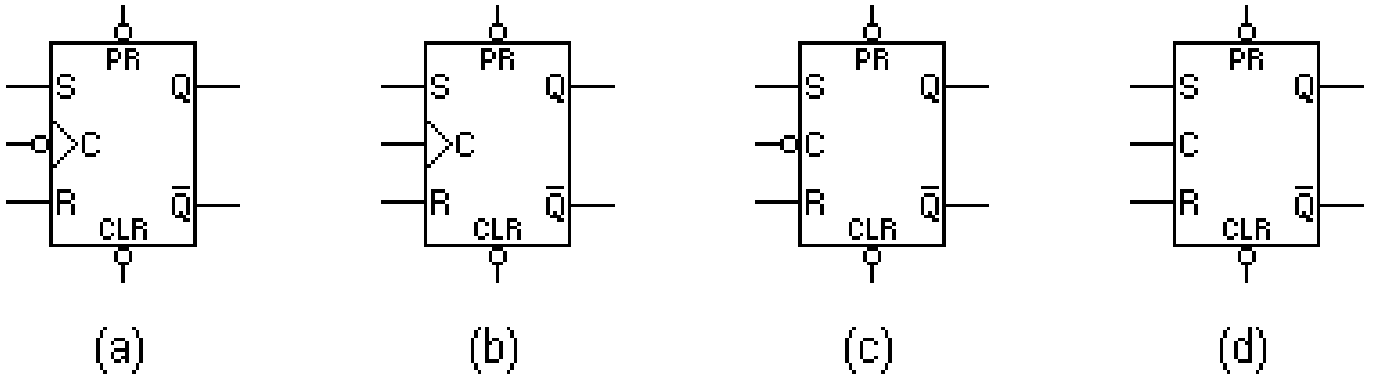
Contador de Programa do Neander (usando flip-flops tipo JK)



J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q_t'

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	1
0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...

Notação gráfica para indicar o tipo de resposta de flip-flops (FFs tipo RS no exemplo)



(a) flip-flop com resposta à borda negativa do sinal de controle
(b) flip-flop com resposta à borda positiva do sinal de controle
(c) latch com resposta ao nível zero do sinal de controle
(d) latch com resposta ao nível um do sinal de controle
PR = “preset”, ou “DC Set”
CLR = “clear” ou “DC Reset”