Arquitetura de Computadores

Estrutura interna de um processador de 8 bits

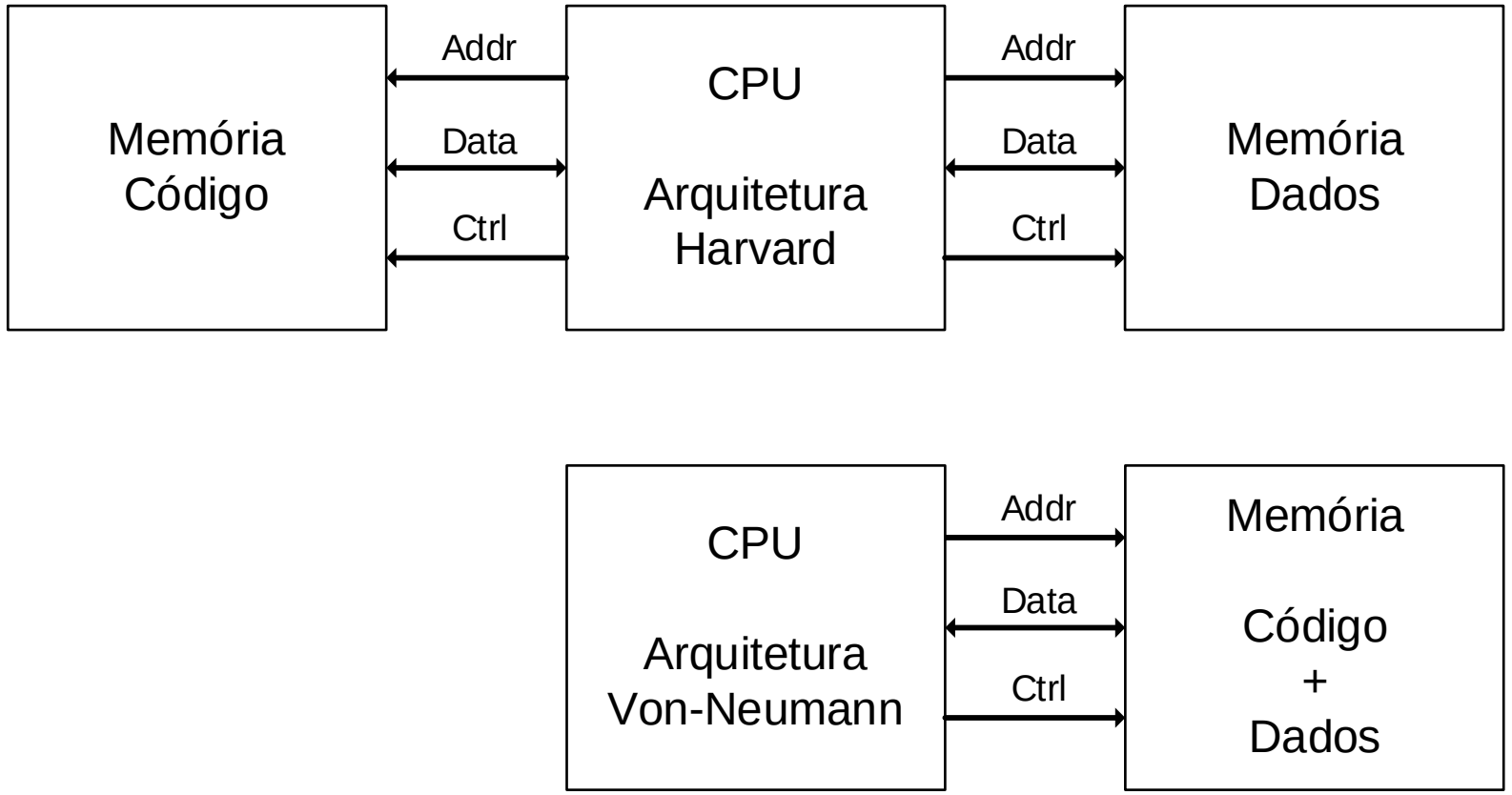
Arquitetura de ciclo múltiplo – Arquitetura Von-Neumann

Bib: A – secção 7.4

João Pedro Patriarca (jpatri@cc.isel.ipl.pt), Gabinete F.0.23 do edifício F

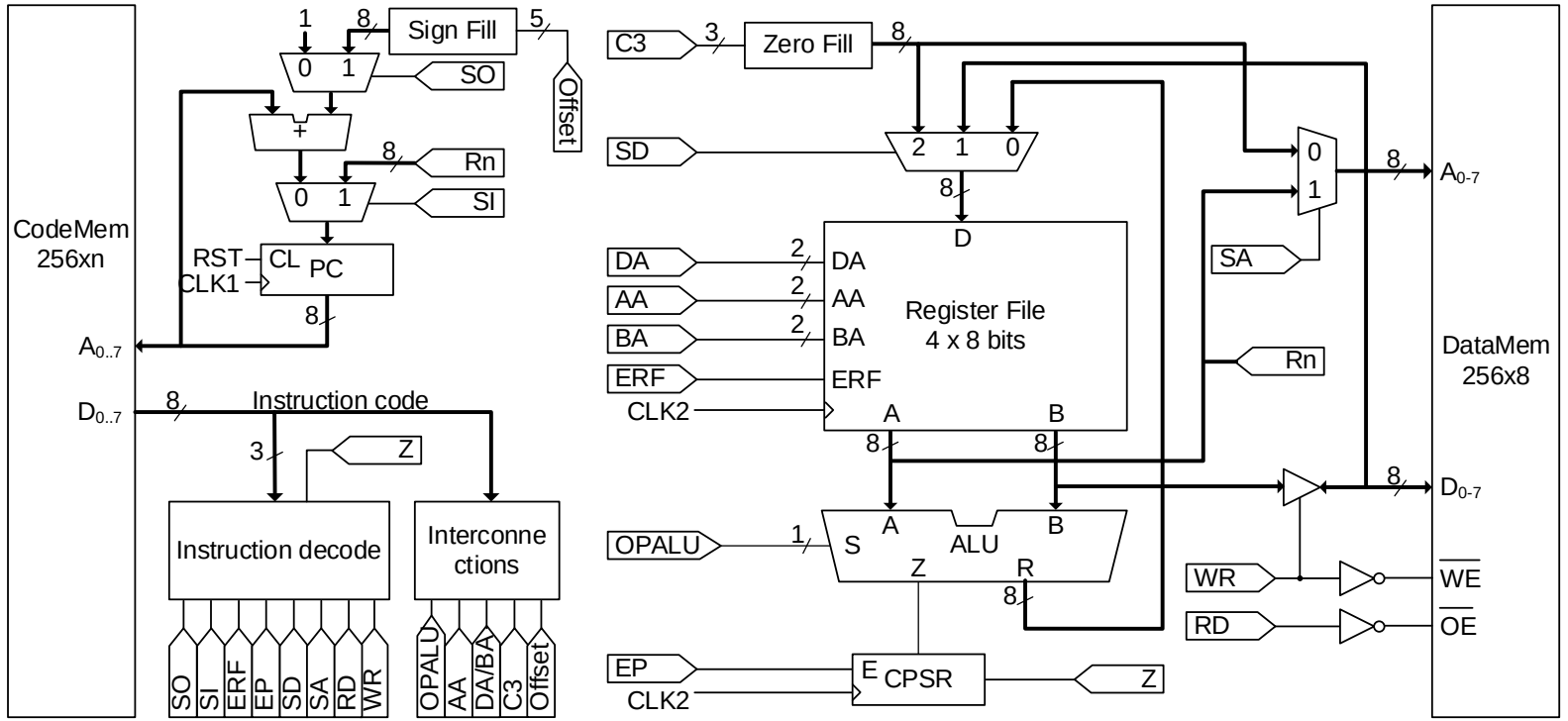
ISEL, ADEETC, LEIC

Arquitetura Harvard versus arquitetura Von-Neumann

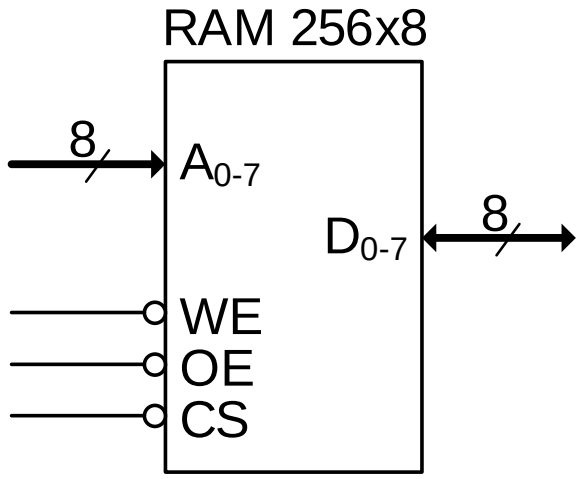


Arquitetura de Harvard

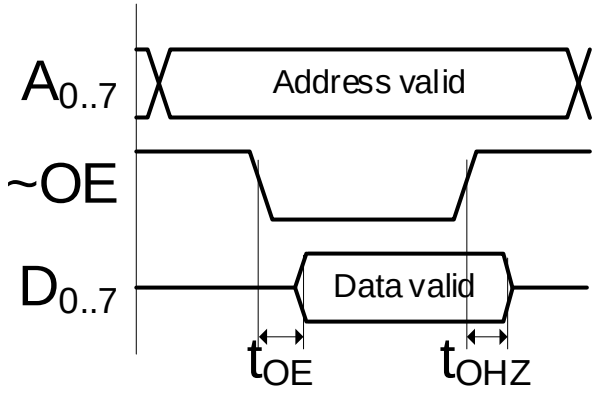
- Espaço de endereçamento de dados e de código distinto
- Sistema de controlo combinatório
- Uma instrução executa num ciclo de *clock* – arquitetura de ciclo único
- Exige uma RAM com tempos de acesso ideais (observar o sistema no Logisim)



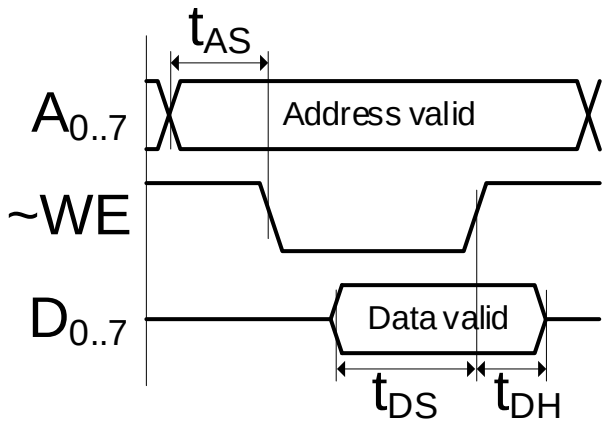
Tempos de acesso a uma memória



Ciclo de leitura

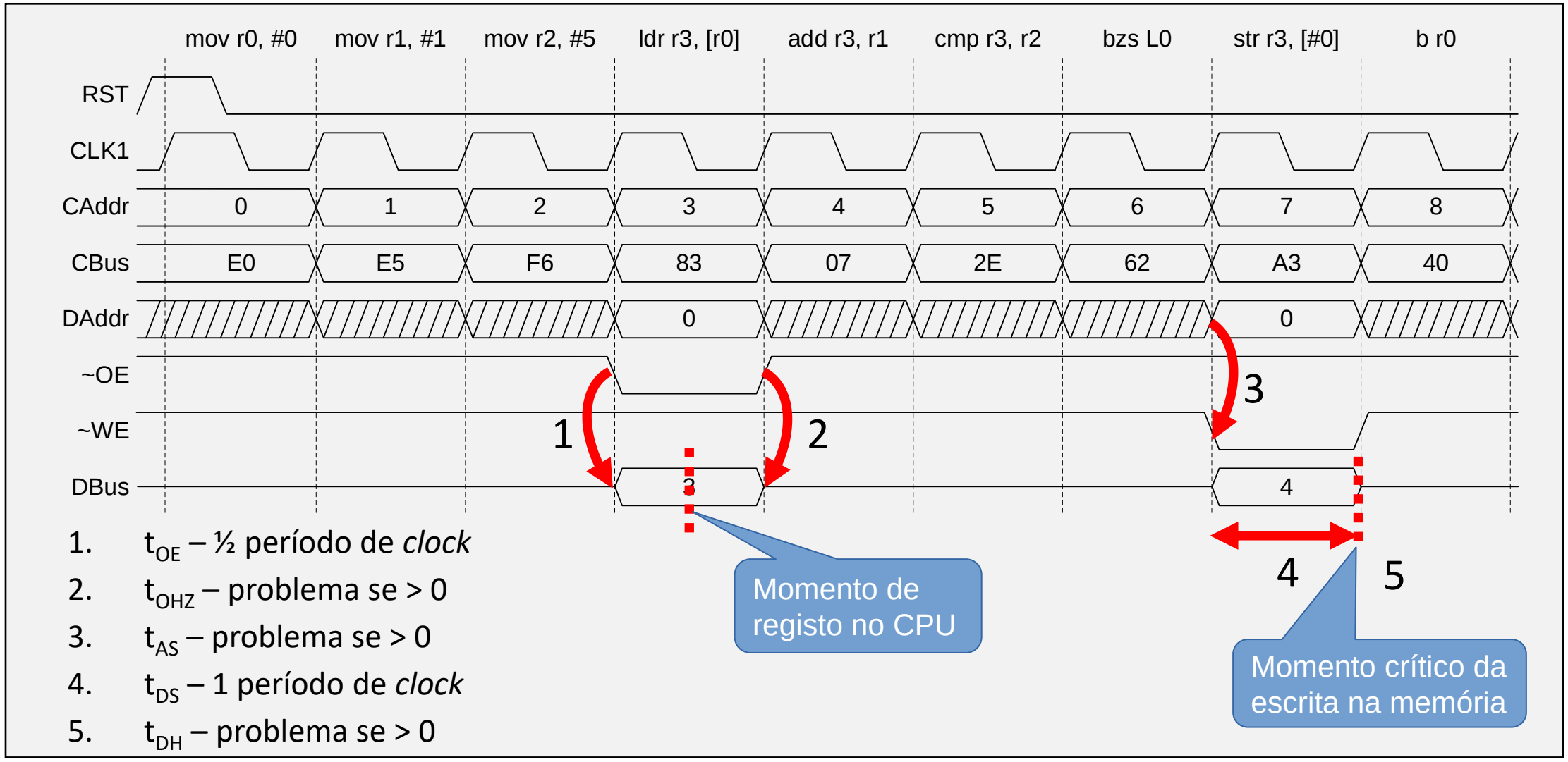


Ciclo de escrita



- t_{OE} – *Output enable to output valid*
- t_{OHZ} – *Output disable to High-Z*
- t_{AS} – *Address setup time*
- t_{DS} – *Data setup time*
- t_{DH} – *Data hold time*

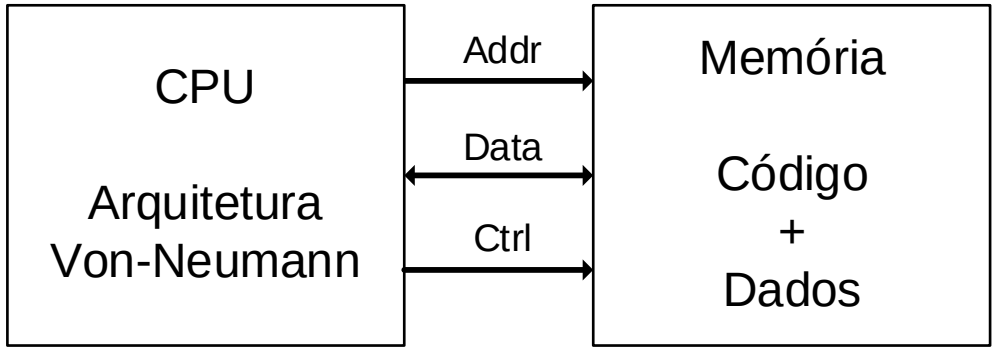
Relação entre tempos de acesso a memória e tempos proporcionados por CPU de ciclo único



1. t_{OE} – ½ período de *clock*
2. t_{OHZ} – problema se > 0
3. t_{AS} – problema se > 0
4. t_{DS} – 1 período de *clock*
5. t_{DH} – problema se > 0

Processador de ciclo múltiplo – arquitetura Von-Neumann

- Ciclo máquina – conjunto de passos para executar uma instrução:
  - *Fetch*: carregamento do código da instrução para registo interno
  - *Decode*: descodificação da instrução
  - *Execute*: ação que concretiza a instrução (escrita em registo interno ou escrita em memória)
  - *Store*: armazenamento do resultado (por simplificação, inclui-se no passo execute)
- Arquitetura Von-Neumann – espaço de endereçamento de código e de dados comum
  - Mesmo barramento de dados de acesso à memória para transferência de código e dados
  - Sistema de controlo sequencial
  - Necessários vários ciclos de *clock* para executar uma instrução
  - Instruções LDR e STR com maior número de ciclos de *clock* por incluírem dois acessos a memória



Estrutura interna da arquitetura Von-Neumann

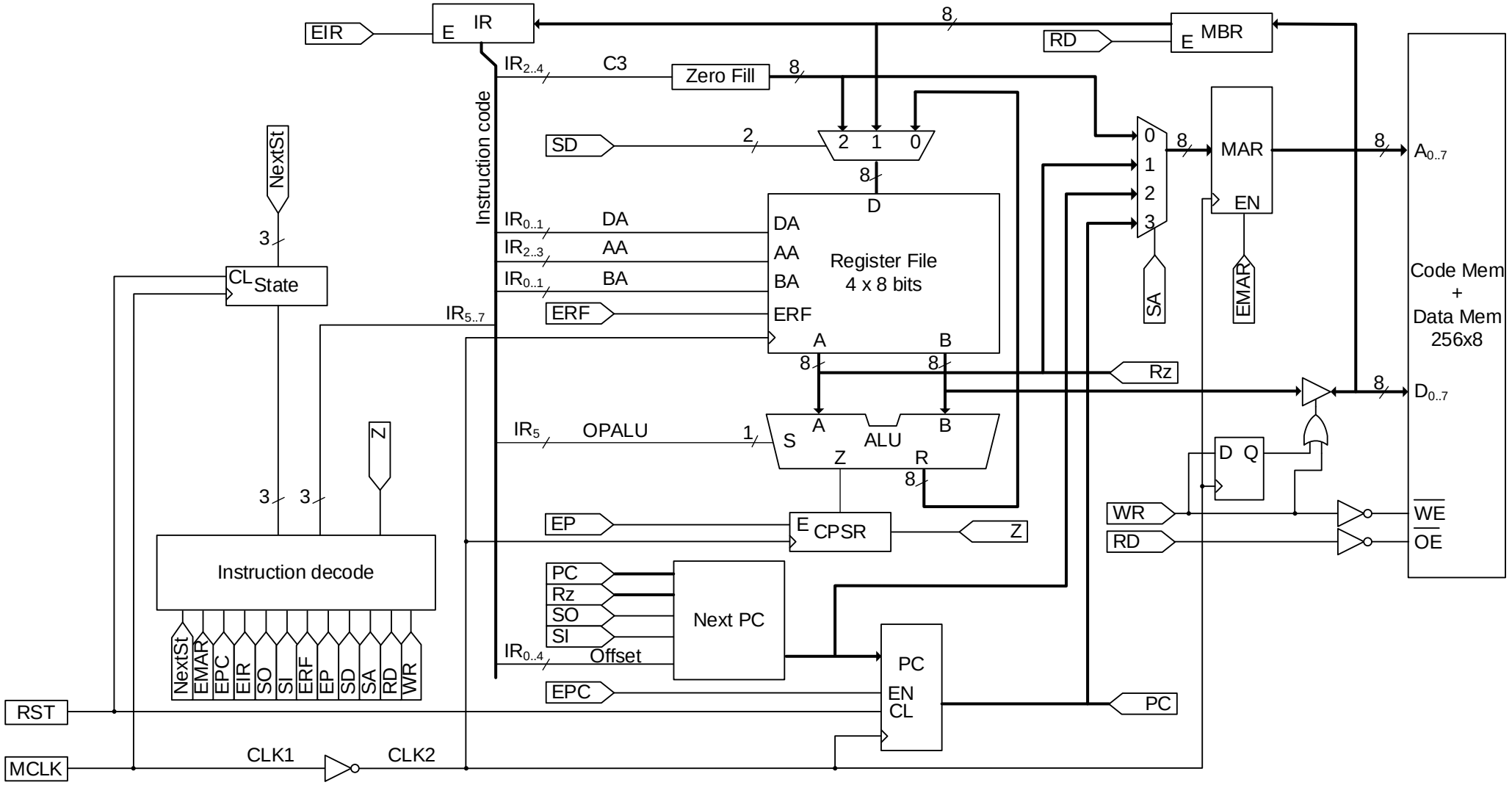


Diagrama de blocos – PC

- Mesmo CLK que restantes módulos funcionais
- Estabelecimento em simultâneo do valor do PC e do próximo endereço (entrada 2 do Mux)
  - Otimização na execução dos Branches

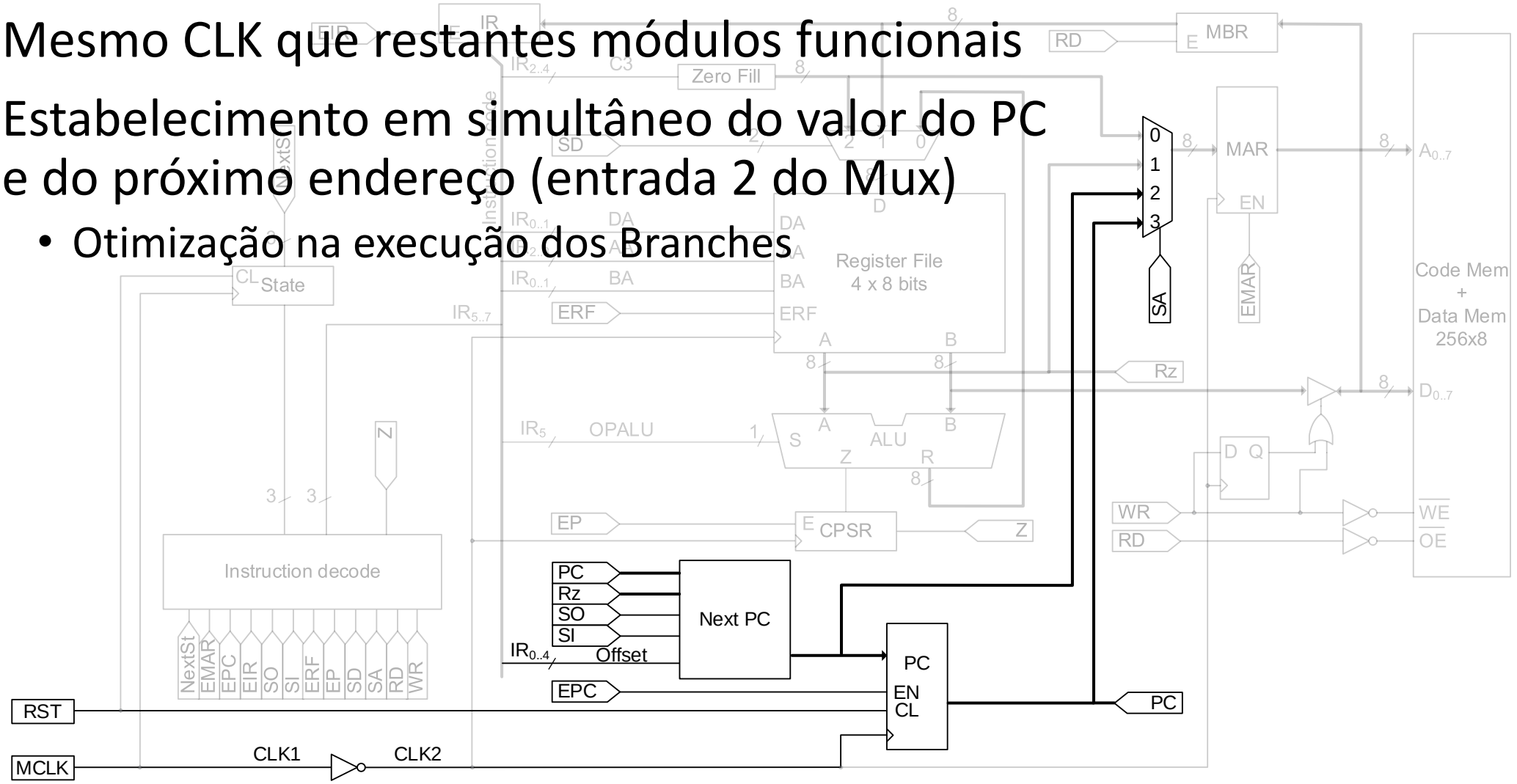


Diagrama de blocos – IR, MBR, MAR e WR

- Registo IR
  - Mantém código da instrução durante todo o ciclo máquina
- Registo MBR (*Memory Buffer Register*)
  - Isola barramento de dados da memória dos componentes internos do CPU
  - *Hold-time* igual a 0
  - Natureza *latch* para permitir um tempo de acesso lento à memória
- Registo MAR (*Memory Address Register*)
  - Mantém endereço durante acesso a memória
- Flip-flop D e OR no WR
  - *Hold-time* de ½ período de *clock* no acesso de escrita à memória

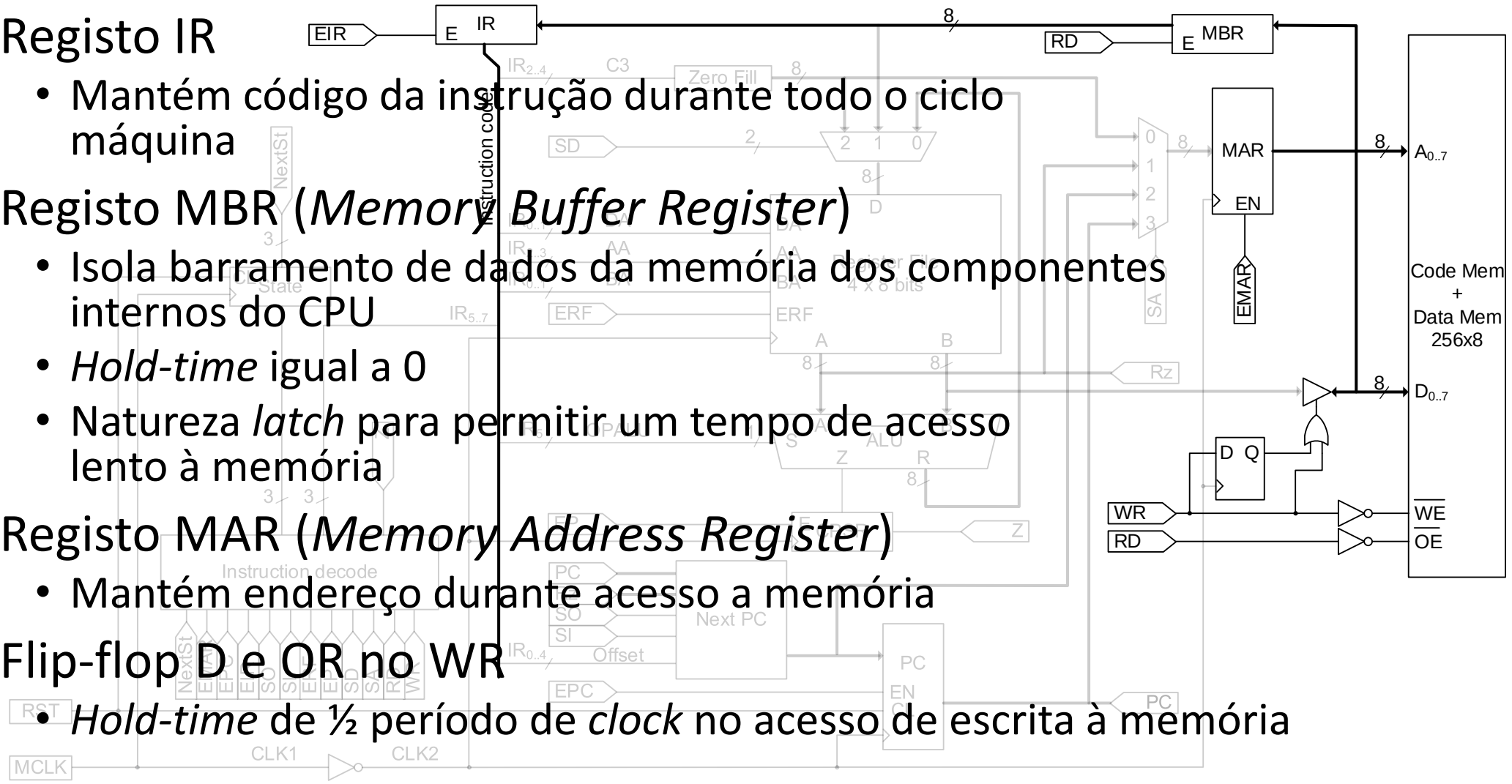
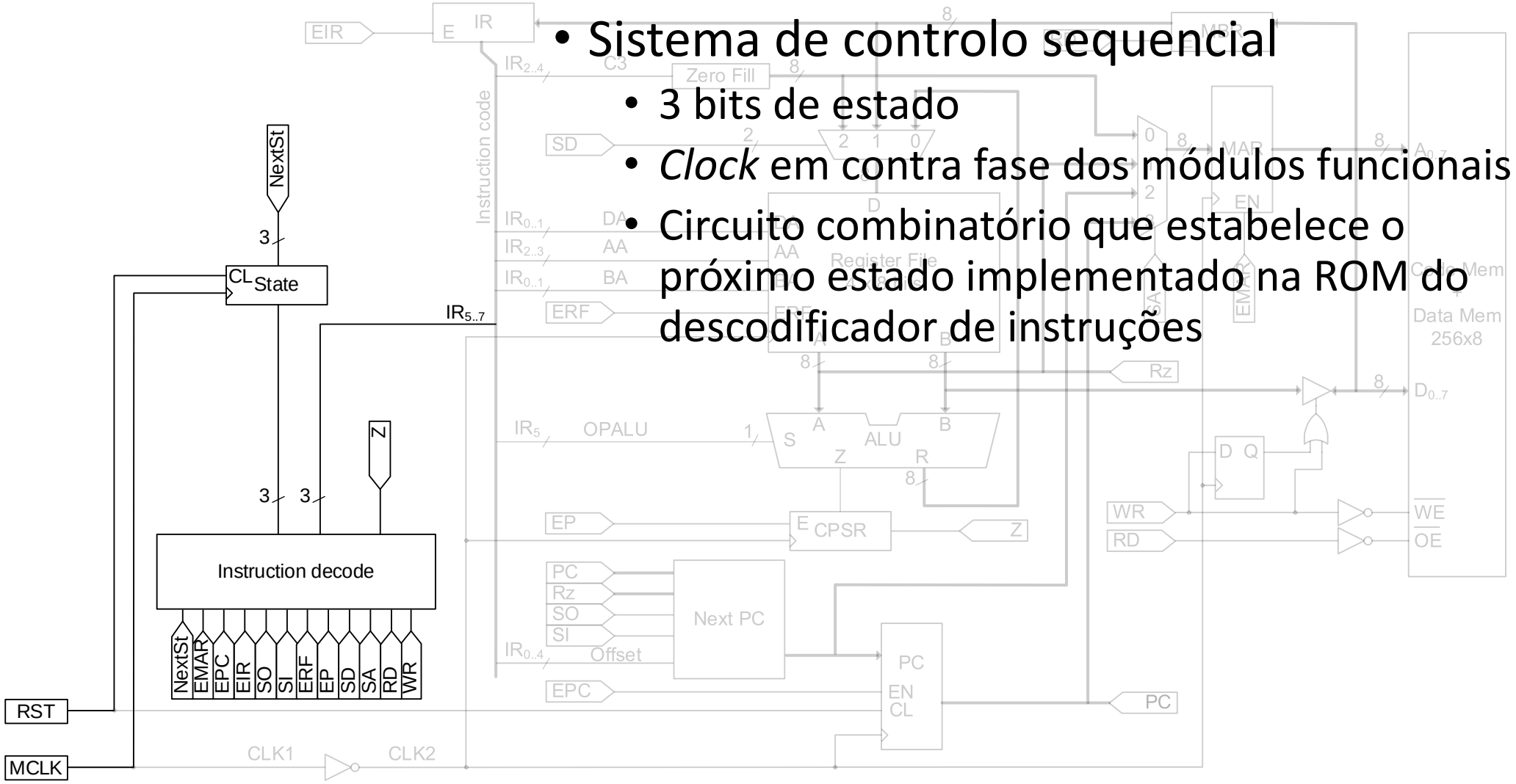
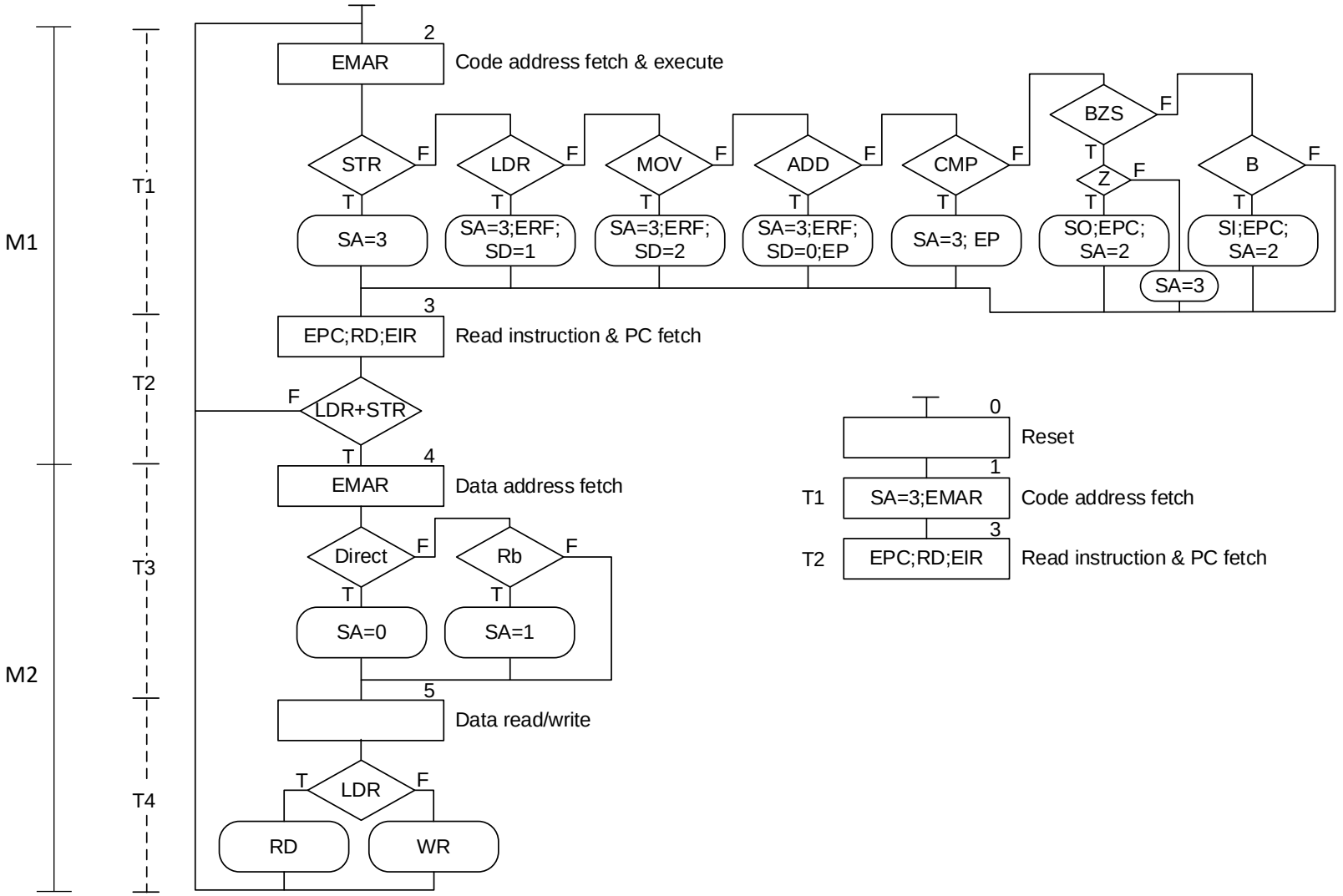


Diagrama de blocos – Sistema de controlo



- Sistema de controlo sequencial
  - 3 bits de estado
  - *Clock* em contra fase dos módulos funcionais
  - Circuito combinatório que estabelece o próximo estado implementado na ROM do descodificador de instruções

Sistema de controlo – ASM



Programação da ROM com micro-código (1/2)

	S2	S1	S0	IC7	IC6	IC5	Z	NS2	NS1	NS0	EMAR	EPC	EIR	SO	SI	ERF	EP	SD1	SD0	SA1	SA0	RD	WR
	A6	A5	A4	A3	A2	A1	A0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Reset	0	0	0	-	-	-	-	0	0	1	0	0	0	-	-	0	0	-	-	-	-	0	0
Code Fecth	0	0	1	-	-	-	-	0	1	1	1	0	0	-	-	0	0	-	-	1	1	0	0
add rd, rn	0	1	0	0	0	0	-	0	1	1	1	0	0	-	-	1	1	0	0	1	1	0	0
cmp rn, rm	0	1	0	0	0	1	-	0	1	1	1	0	0	-	-	0	1	-	-	1	1	0	0
b rn	0	1	0	0	1	0	-	0	1	1	1	1	0	-	1	0	0	-	-	1	0	0	0
bzs label	0	1	0	0	1	1	0	0	1	1	1	0	0	-	-	0	0	-	-	1	1	0	0
bzs label	0	1	0	0	1	1	1	0	1	1	1	1	0	1	0	0	0	-	-	1	0	0	0
ldr rd, [rn]	0	1	0	1	0	0	-	0	1	1	1	0	0	-	-	1	0	0	1	1	1	1	0
str rs, [#imm3]	0	1	0	1	0	1	-	0	1	1	1	0	0	-	-	0	0	-	-	1	1	0	1
NOP (No Operation)	0	1	0	1	1	0	-	0	1	1	1	0	0	-	-	0	0	-	-	1	1	0	0
mov rd, #imm3	0	1	0	1	1	1	-	0	1	1	1	0	0	-	-	1	0	1	0	1	1	0	0

Programação da ROM com micro-código (2/2)

	S2	S1	S0	IC7	IC6	IC5	Z	NS2	NS1	NS0	EMAR	EPC	EIR	SO	SI	ERF	EP	SD1	SD0	SA1	SA0	RD	WR
	A6	A5	A4	A3	A2	A1	A0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Rd Inst & PC Fetch	0	1	1	0	-	-	-	0	1	0	0	1	1	0	0	0	0	-	-	-	-	1	0
Rd Inst & PC Fetch	0	1	1	1	0	-	-	1	0	0	0	1	1	0	0	0	0	-	-	-	-	1	0
Rd Inst & PC Fetch	0	1	1	1	1	-	-	0	1	0	0	1	1	0	0	0	0	-	-	-	-	1	0
Data Addr Fetch	1	0	0	0	-	-	-	1	0	1	1	0	0	-	-	0	0	-	-	-	-	0	0
Data Addr Fecth	1	0	0	1	0	0	-	1	0	1	1	0	0	-	-	0	0	-	-	0	1	0	0
Data Addr Fecth	1	0	0	1	0	1	-	1	0	1	1	0	0	-	-	0	0	-	-	0	0	0	0
Data Addr Fecth	1	0	0	1	1	-	-	1	0	1	1	0	0	-	-	0	0	-	-	-	-	0	0
Data Read/Write	1	0	1	-	-	0	-	0	1	0	0	0	0	-	-	0	0	-	-	-	-	1	0
Data Read/Write	1	0	1	-	-	1	-	0	1	0	0	0	0	-	-	0	0	-	-	-	-	0	1
NOP (No Operation)	1	1	-	-	-	-	-	0	1	0	0	0	0	-	-	0	0	-	-	-	-	0	0

Programa exemplo

```
ADDR BC Instruction
0000 E0 mov r0, #0
0001 E5 mov r1, #1
0002 F6 mov r2, #5
0003 83 ldr r3, [r0]
0004 07 add r3, r1
0005 2E cmp r3, r2
0006 62 bzs L0
0007 A3 str r3, [#0]
0008 40 L0: b   r0
```

	7	6	5	4	3	2	1	0
1. add rd, rn	0	0	0	-	rn		rd	
2. b rn	0	1	0	-	rn		-	-
3. bzs label	0	1	1	label				
4. cmp rn, rm	0	0	1	-	rn		rm	
5. ldr rd, [rn]	1	0	0	-	rn		rd	
6. mov rd, #imm3	1	1	1	imm3			rd	
7. str rs, [#imm3]	1	0	1	imm3			rs	

Diagrama temporal produzido pela execução do programa exemplo 1/2

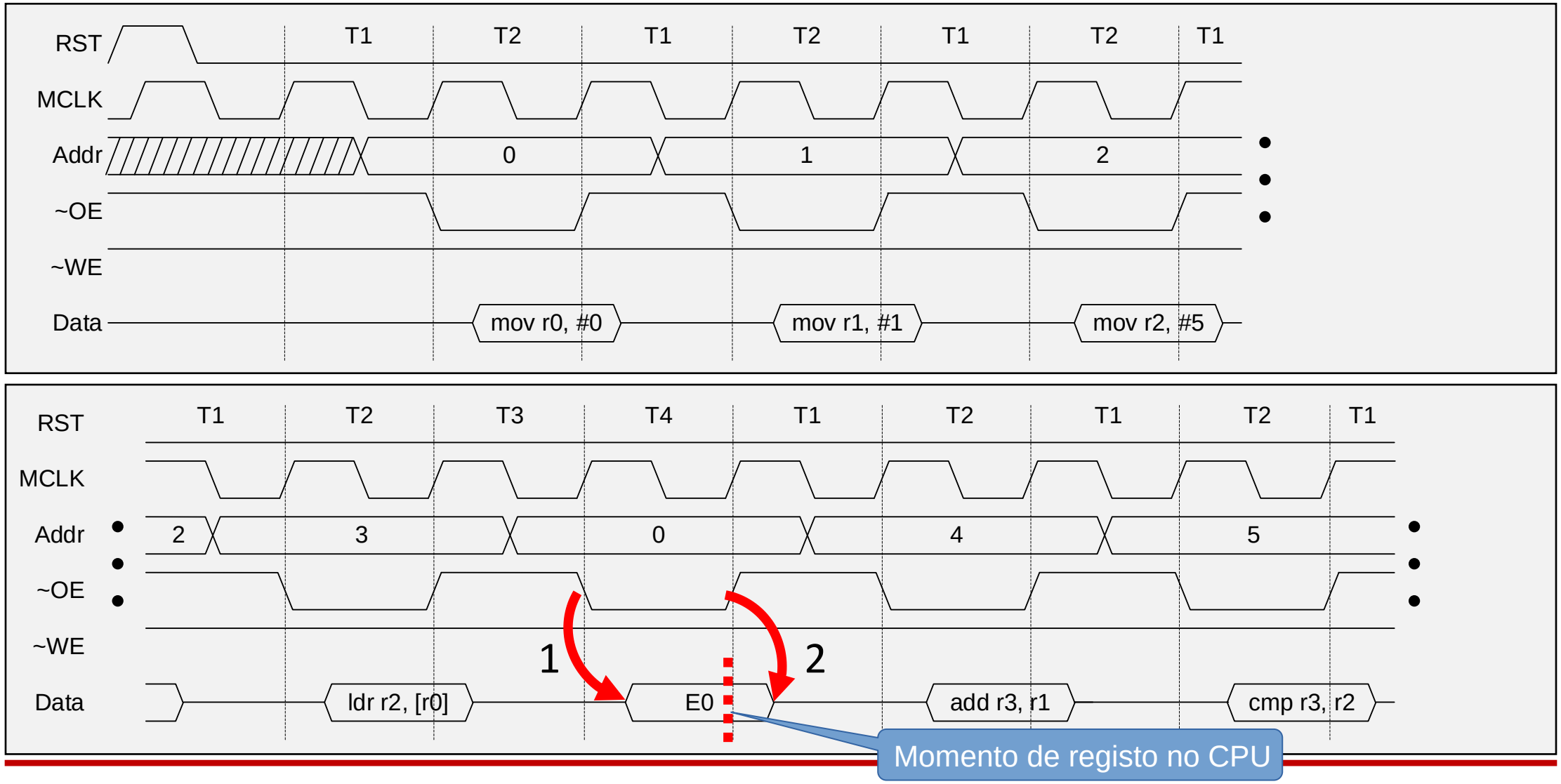
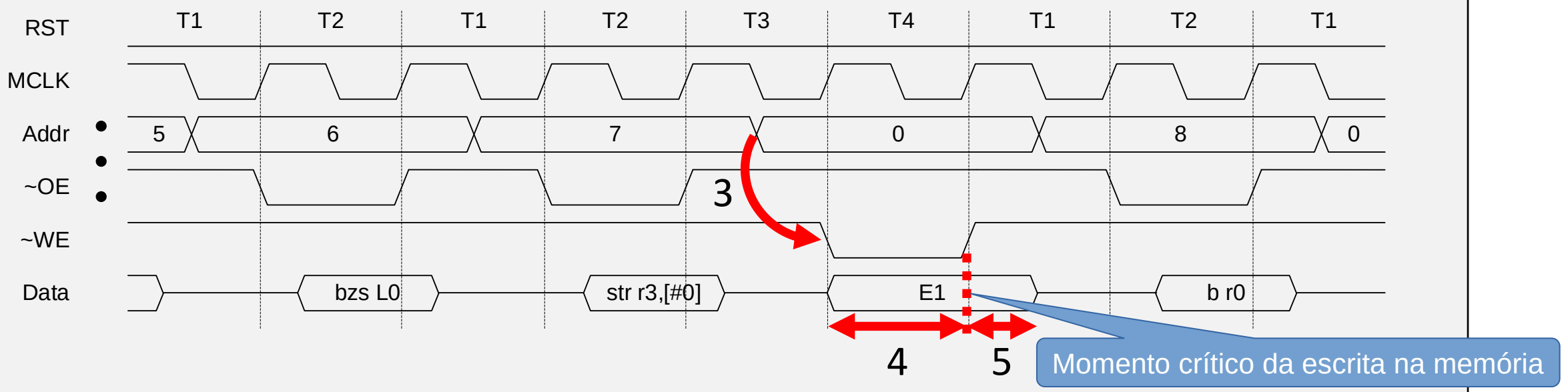


Diagrama temporal produzido pela execução do programa exemplo 2/2



1. t_{OE} – o CPU proporciona 1 ciclo de *clock* para a memória estabilizar os dados no barramento de dados
2. t_{OHZ} – o CPU proporciona 1 ciclo de *clock* para a memória libertar o barramento de dados
3. t_{AS} – o CPU proporciona ½ ciclo de *clock* de endereços estáveis antes de ativar o sinal WR
4. t_{DS} – o CPU proporciona 1 ciclo de *clock* de dados estáveis antes de desativar o sinal WR
5. t_{DH} – o CPU proporciona ½ ciclo de *clock* de dados estáveis após a desativação do sinal WR