Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Graduação em Ciências da Computação

Sistemas Digitais

INE 5406

Aula 12-T

4. Projeto de Sistemas Digitais no Nível RT. Análise de *Timing* de um SD, Barramentos x Multiplexadores, Registradores x Banco de Registradores.

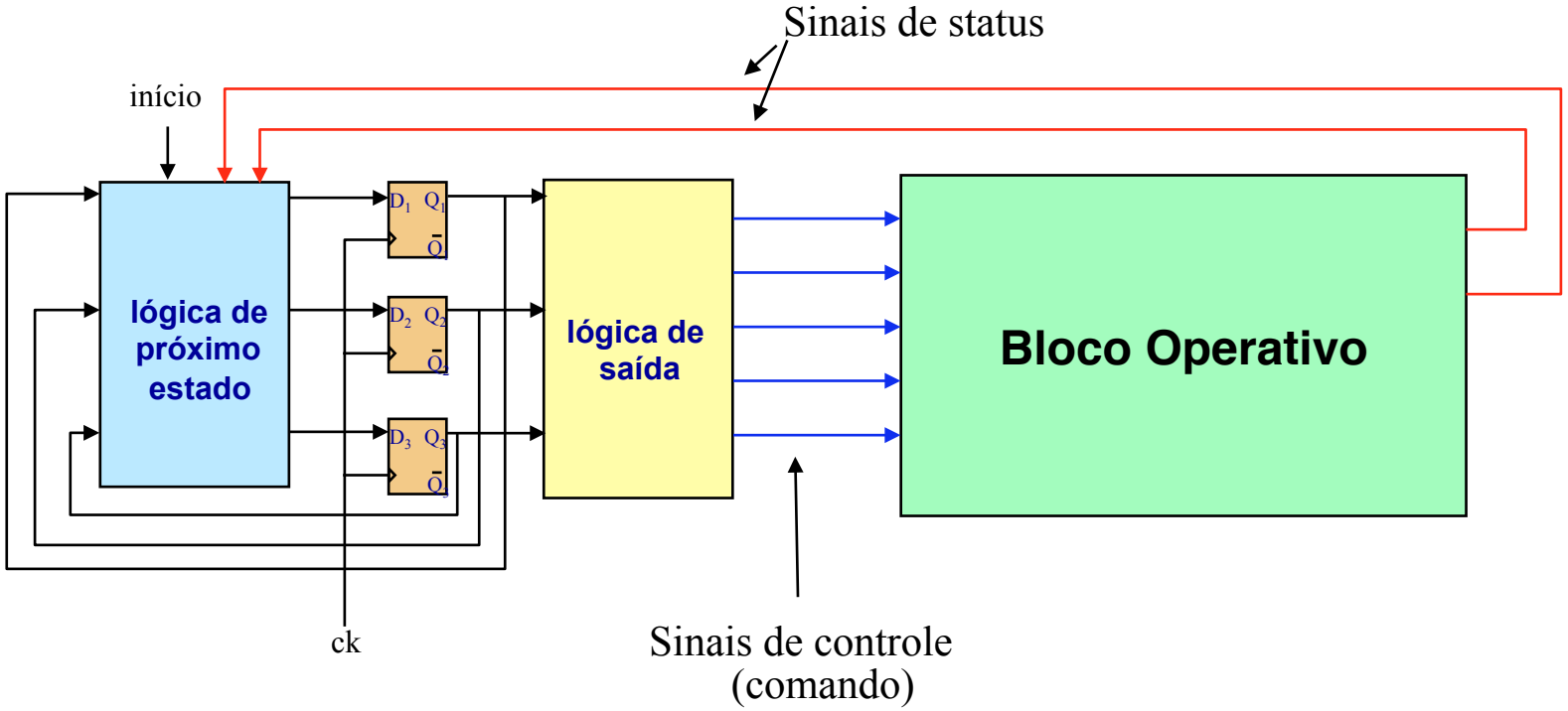
Prof. José Luís Güntzel
guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing*

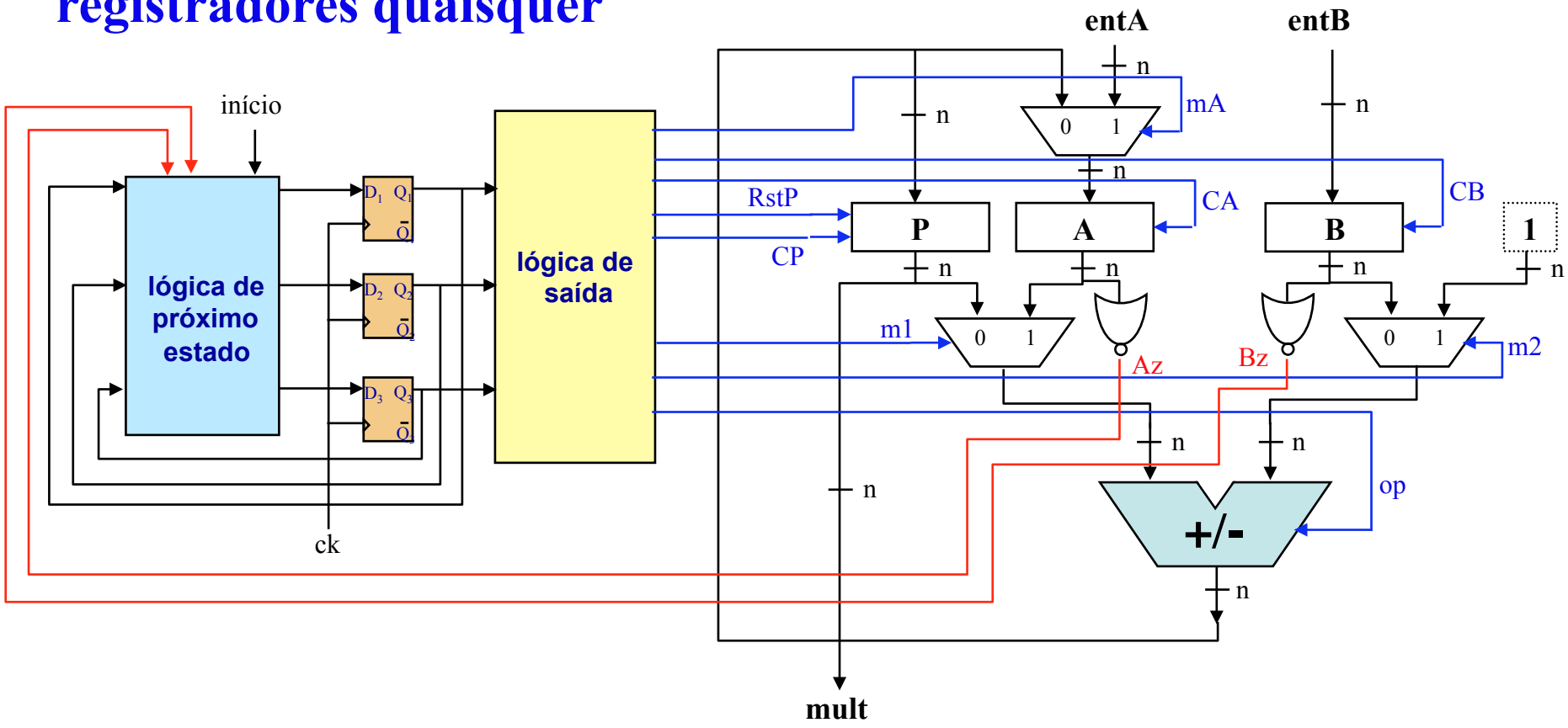
Objetivo: encontrar o caminho mais longo entre dois registradores quaisquer (considerando BC & BO)



4. Projeto de Sistemas Digitais no Nível RT

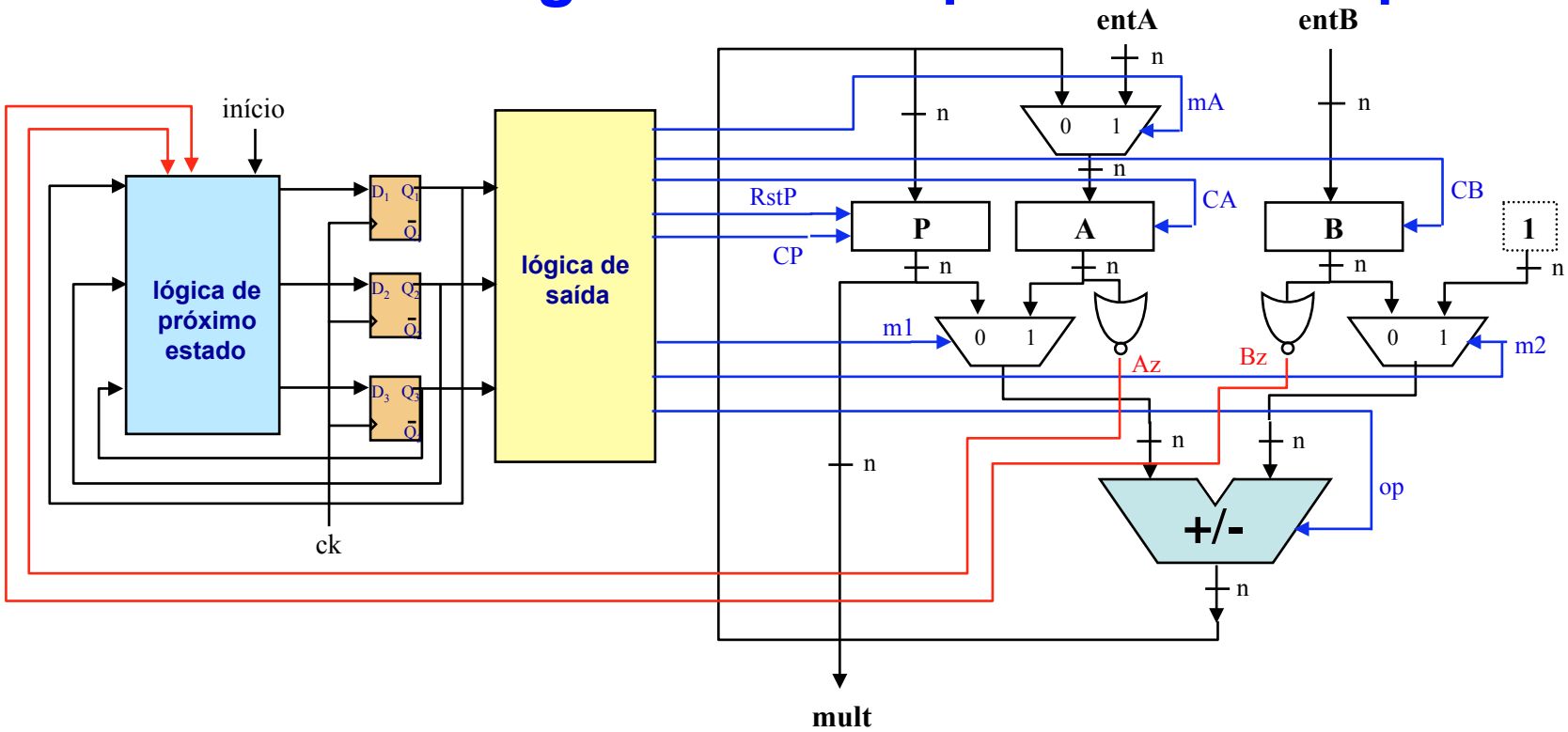
Análise de *Timing*

Objetivo: encontrar o caminho mais longo entre dois registradores quaisquer



4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing*: Premissas para este exemplo



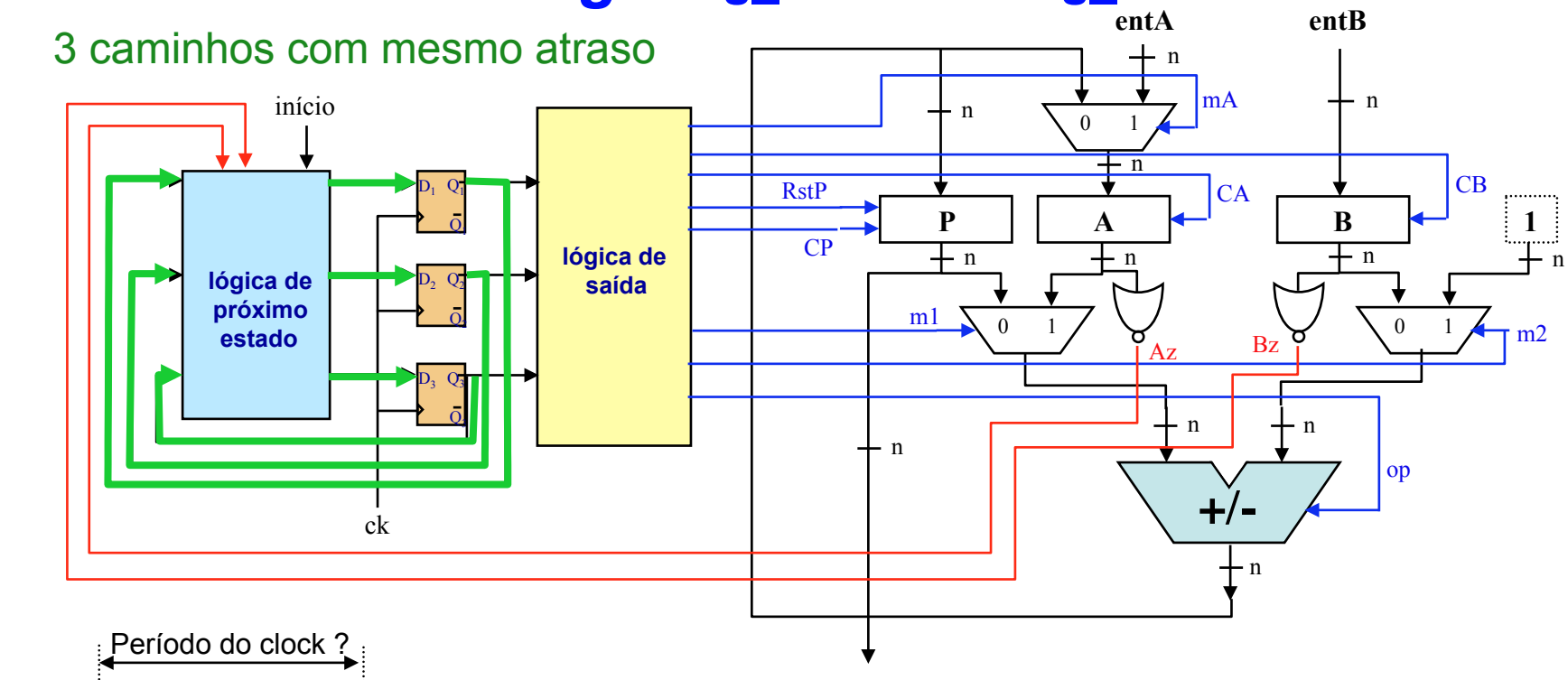
Supor que todos os registradores (incluso o de estados):

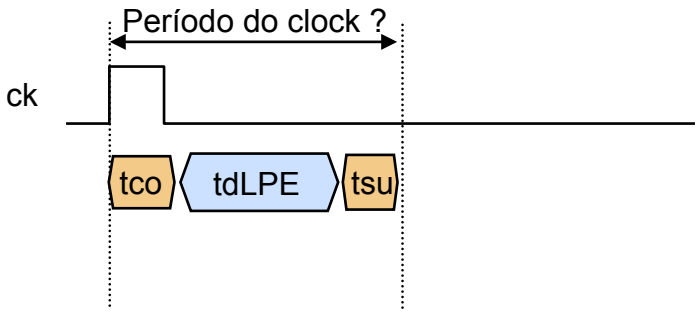
- São disparados pela borda de subida de ck
- Apresentam mesmas características temporais, i.e., tco, tsu e th

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing*: Reg_estado ➔Reg_estado

3 caminhos com mesmo atraso





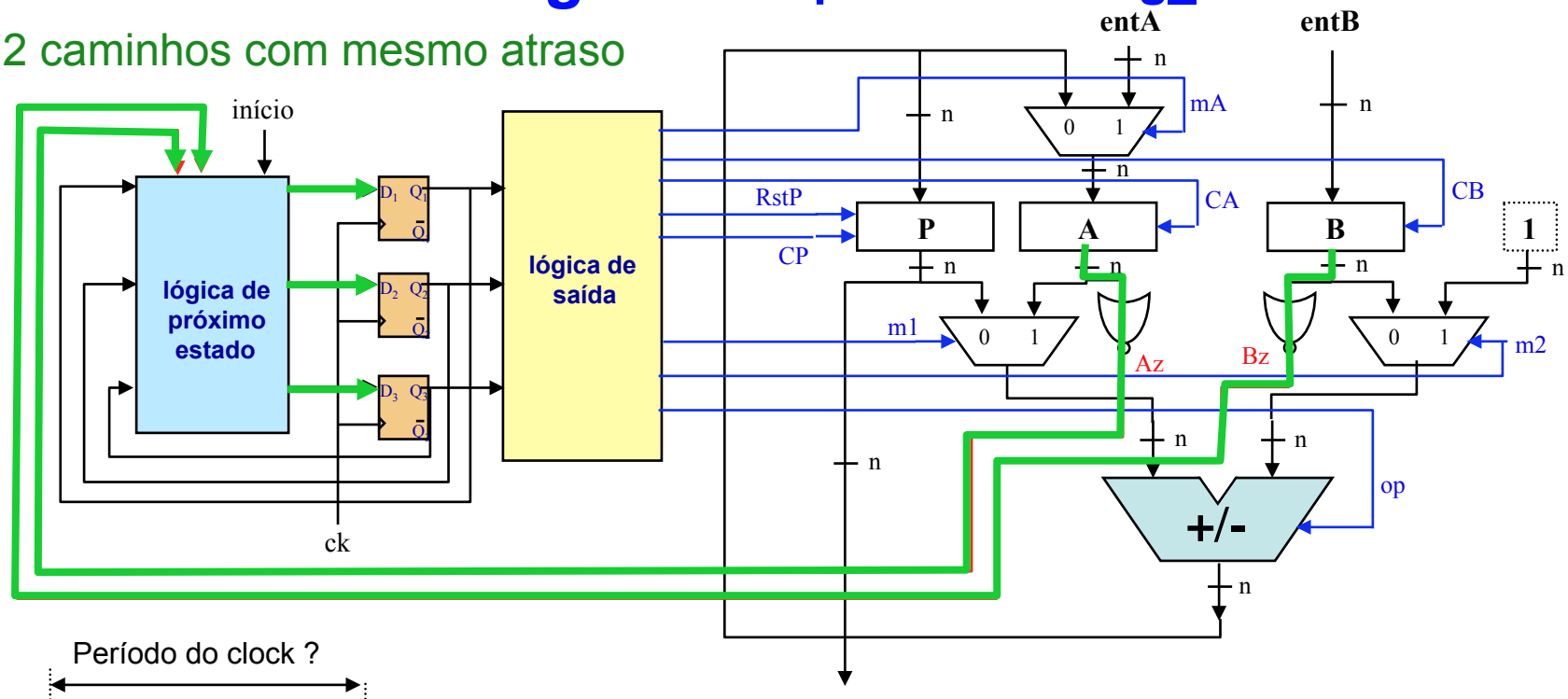
tco e tsu do registrador de estado

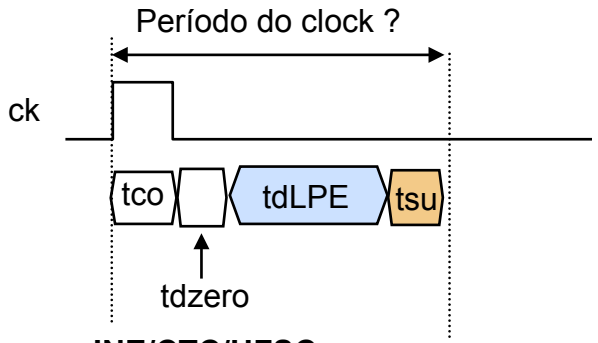
tdLPE = atraso (max) de propagação da lógica de próximo estado

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing:* Bloco operativo ➔ Reg_estado

2 caminhos com mesmo atraso





tco = tco do registrador A ou B

tdzero = atraso (max) de propagação de um testador de zero

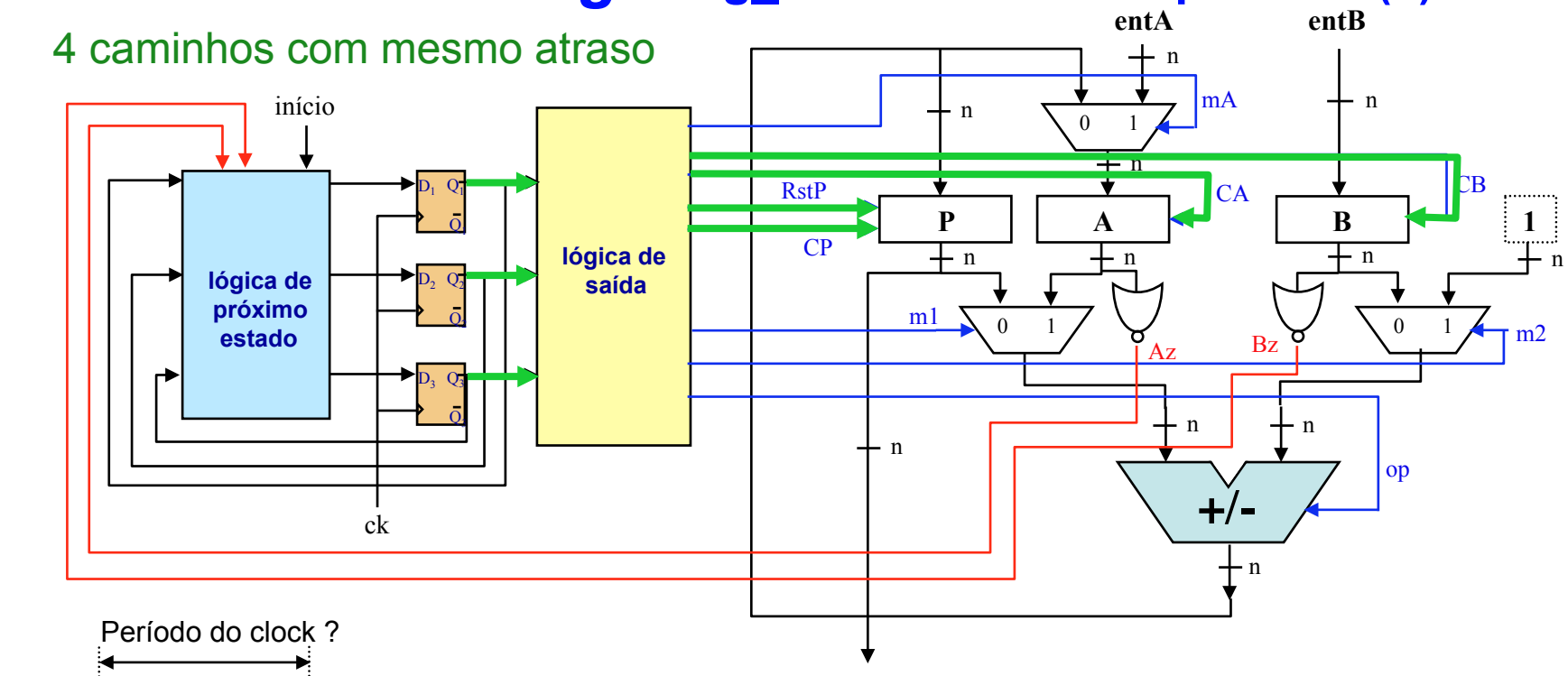
tdLPE = atraso (max) de propagação da lógica de próximo estado

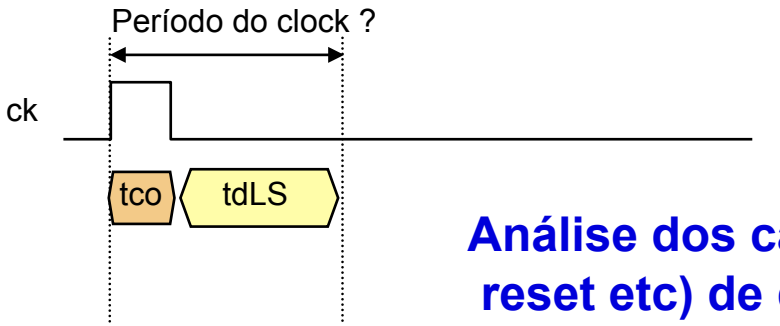
tsu = tempo de *setup* do registrador de estado

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing:* Reg_estado ➔Bloco operativo (1)

4 caminhos com mesmo atraso





tco = tco do registrador de estado

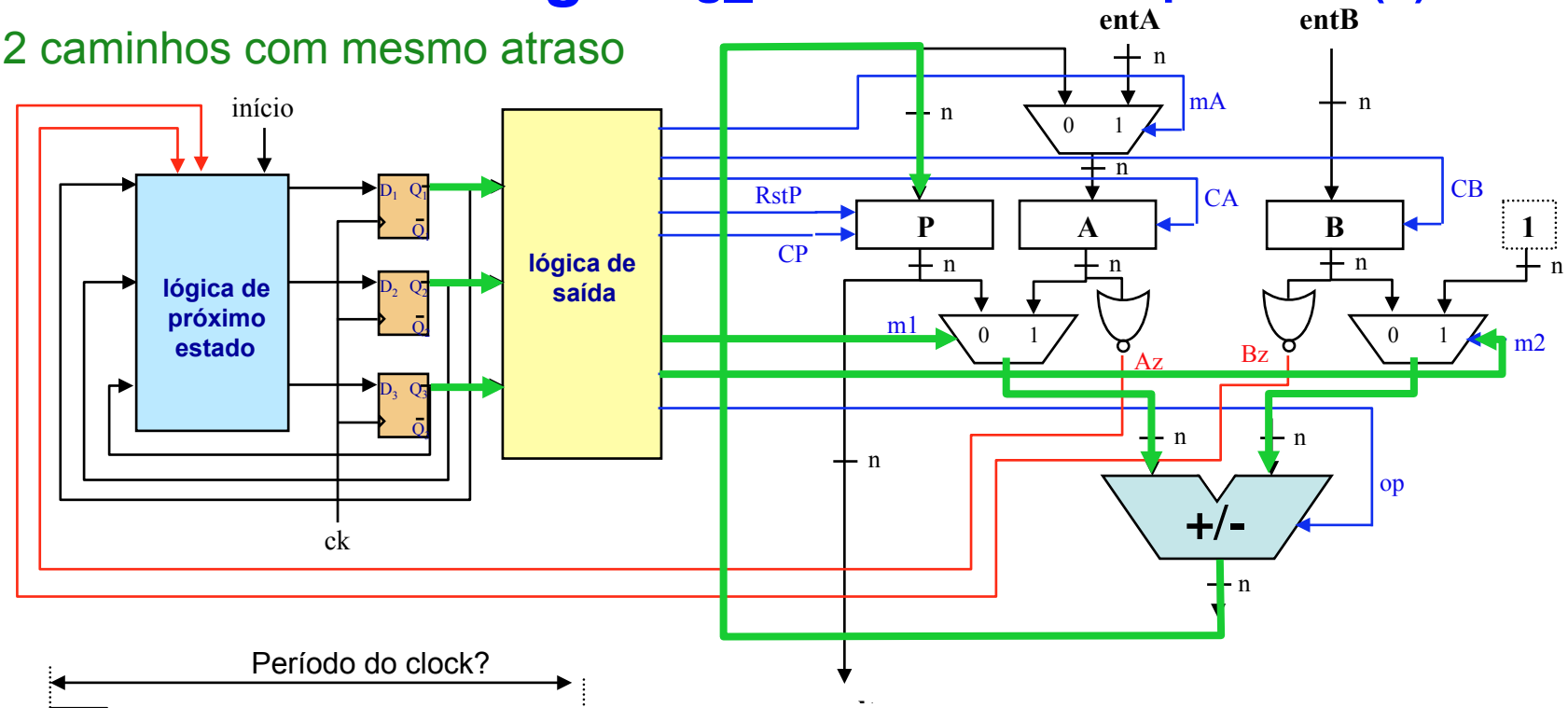
tdLS = atraso (max) de propagação da lógica saída

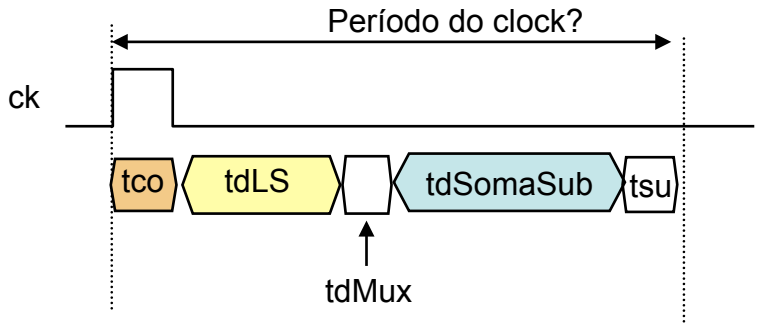
Análise dos caminhos que terminam no controle (carga, reset etc) de qualquer registrador do bloco operativo

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing:* Reg_estado ➔Bloco operativo (2)

2 caminhos com mesmo atraso



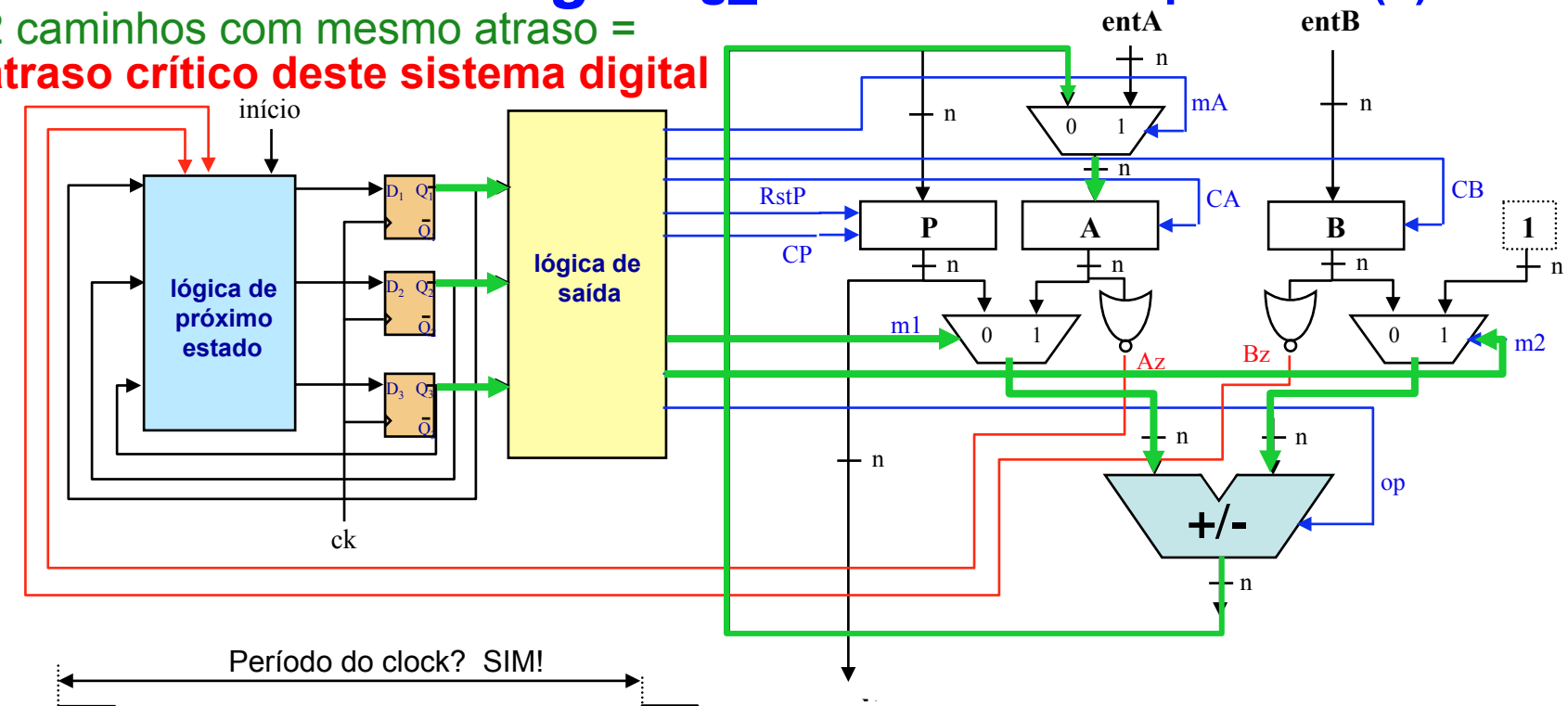


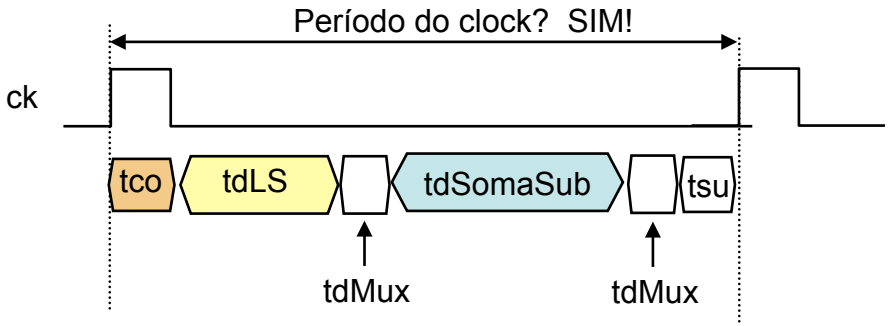
tco = tco do registrador de estado
tdLS = atraso (max) de propagação da lógica saída
tdSomaSub = atraso de propagação do somador-subtrator
tsu = tempo de *setup* do registrador P

4. Projeto de Sistemas Digitais no Nível RT

Análise de *Timing:* Reg_estado ➔Bloco operativo (3)

2 caminhos com mesmo atraso =
atraso crítico deste sistema digital



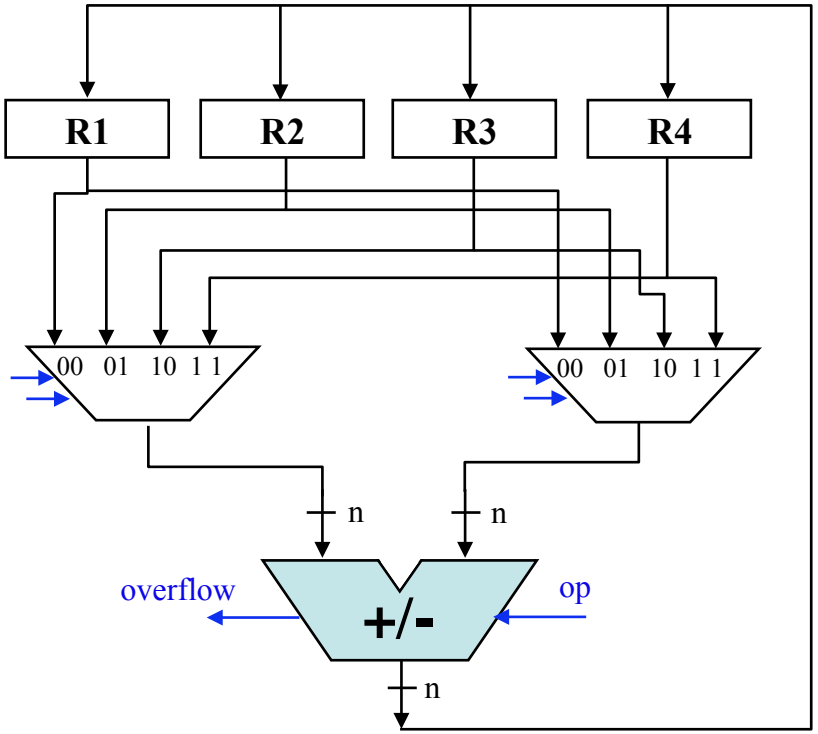


tco = tco do registrador de estado
tdLS = atraso (max) de propagação da lógica saída
tdSomaSub = atraso de propagação do somador-subtrator
tdMux = atraso (max) de propagação de um mux 2:1
tsu = tempo de *setup* do registrador A

4. Projeto de Sistemas Digitais no Nível RT

BO com Multiplexadores

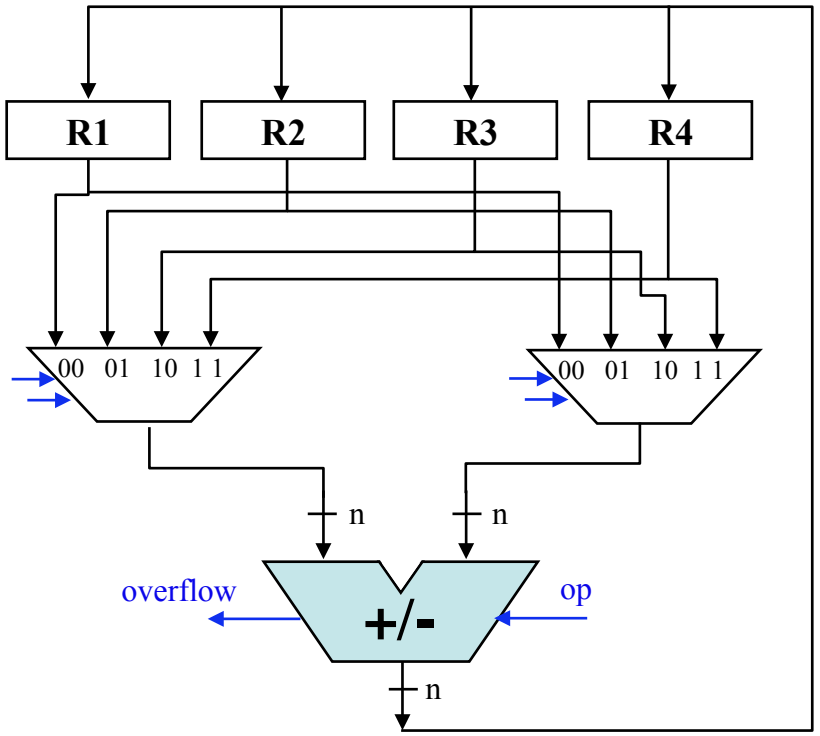
4 sinais para controlar o acesso à UF



Barramentos x Multiplexadores

Porém:

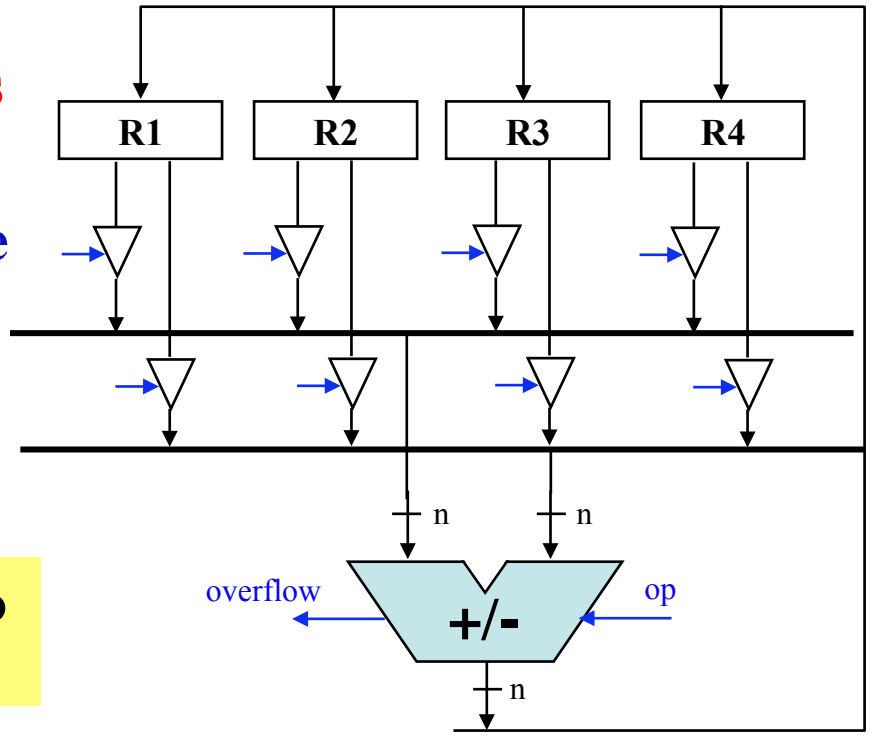
- **Se qualquer registrador pode ser fonte ou destino de dados para ambas as entradas da UF**
- **E se somente dois registradores servem de operandos para a UF (a cada ciclo de relógio)**



4. Projeto de Sistemas Digitais no Nível RT

Barramentos x Multiplexadores

- **Então, é melhor usar barramentos (um por entrada da UF)!**
- **Usar chaves tri-state, pois somente um registrador pode escrever no barramento, por vez (i.e., por ciclo de relógio)**

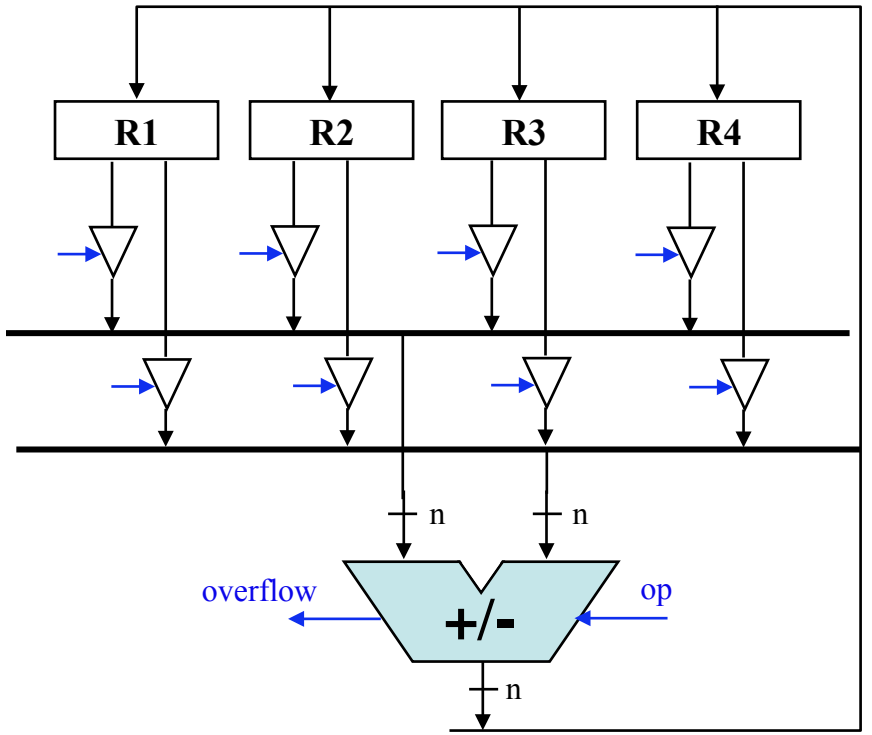


8 sinais para controlar o acesso à UF!

Barramentos x Multiplexadores

- **Porém, somente um registrador pode escrever no barramento, por vez (i.e., por ciclo de relógio)**
- **Logo, usar *tri-state***

8 sinais de controle



4. Projeto de Sistemas Digitais no Nível RT

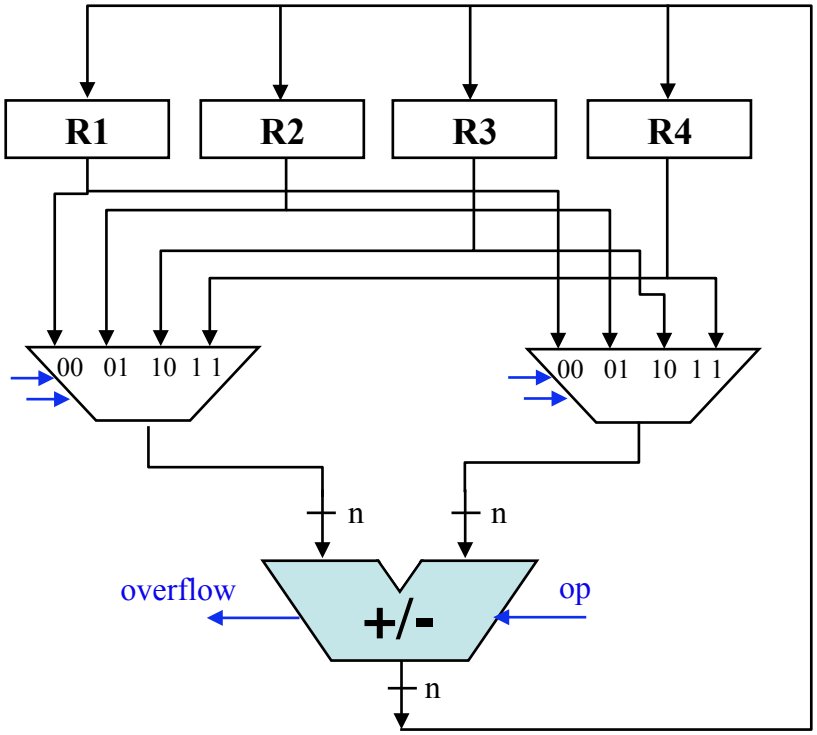
Registradores x Banco de Registradores

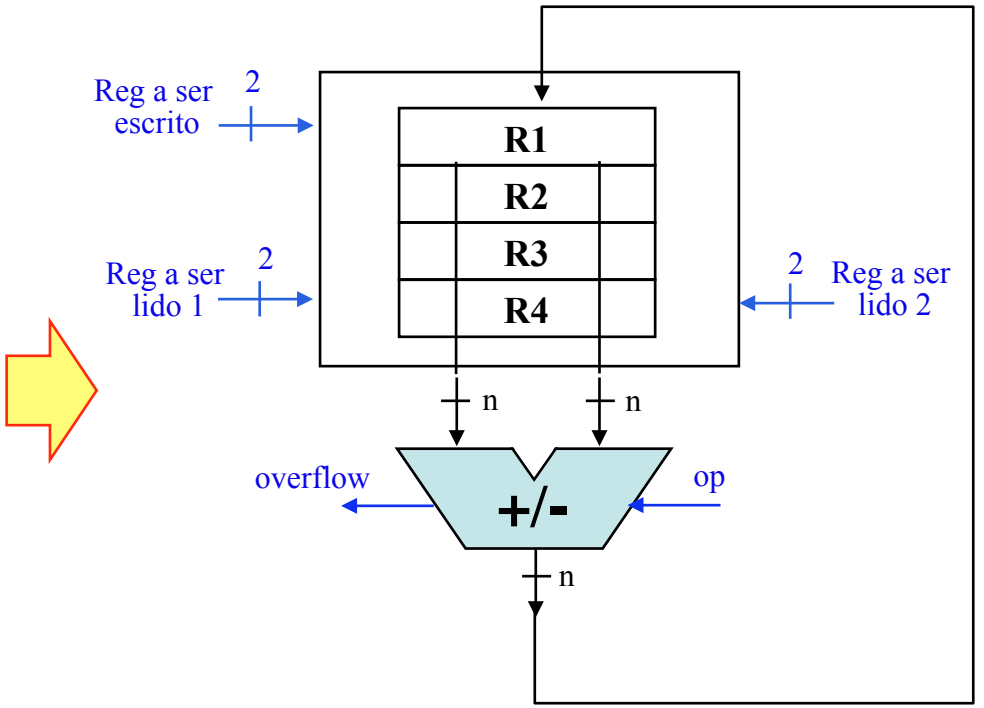
- **Se houver uma quantidade grande de registradores (>= 4)**
- **Se somente um registrador está conectado a cada entrada da UF, por vez (i.e., por ciclo de relógio)**
- **Então, é possível reduzir custo da rede de interconexão agrupando os registradores em um "banco de registradores"**

4. Projeto de Sistemas Digitais no Nível RT

Registradores x Banco de Registradores

Exemplo 1:

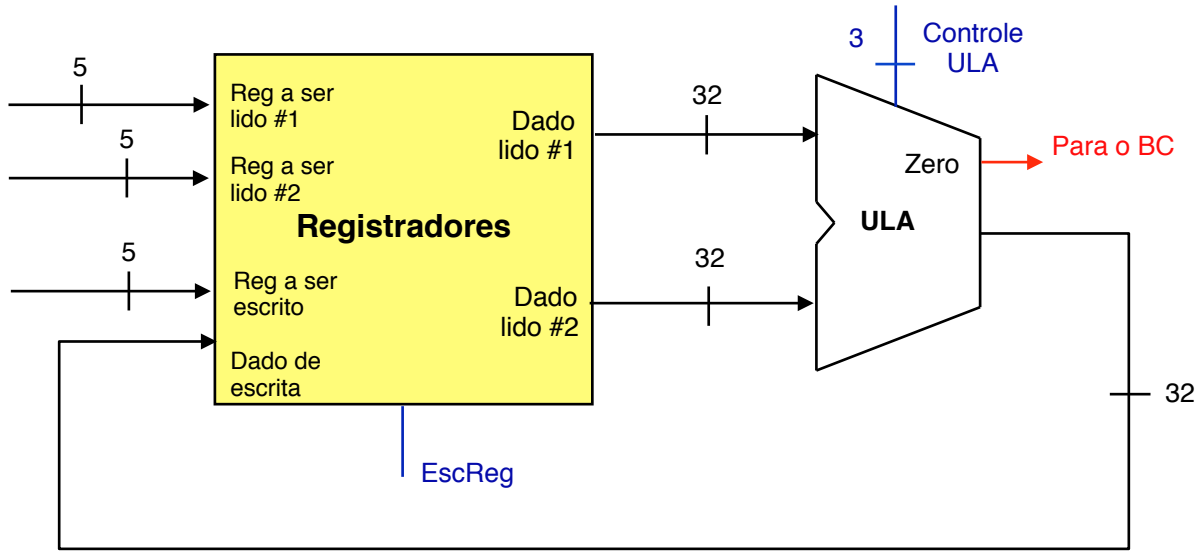




- 4 registradores
- 2 “portas” de leitura e 1 “porta” de escrita
- 2 bits de endereço/ por “porta”

Registradores x Banco de Registradores

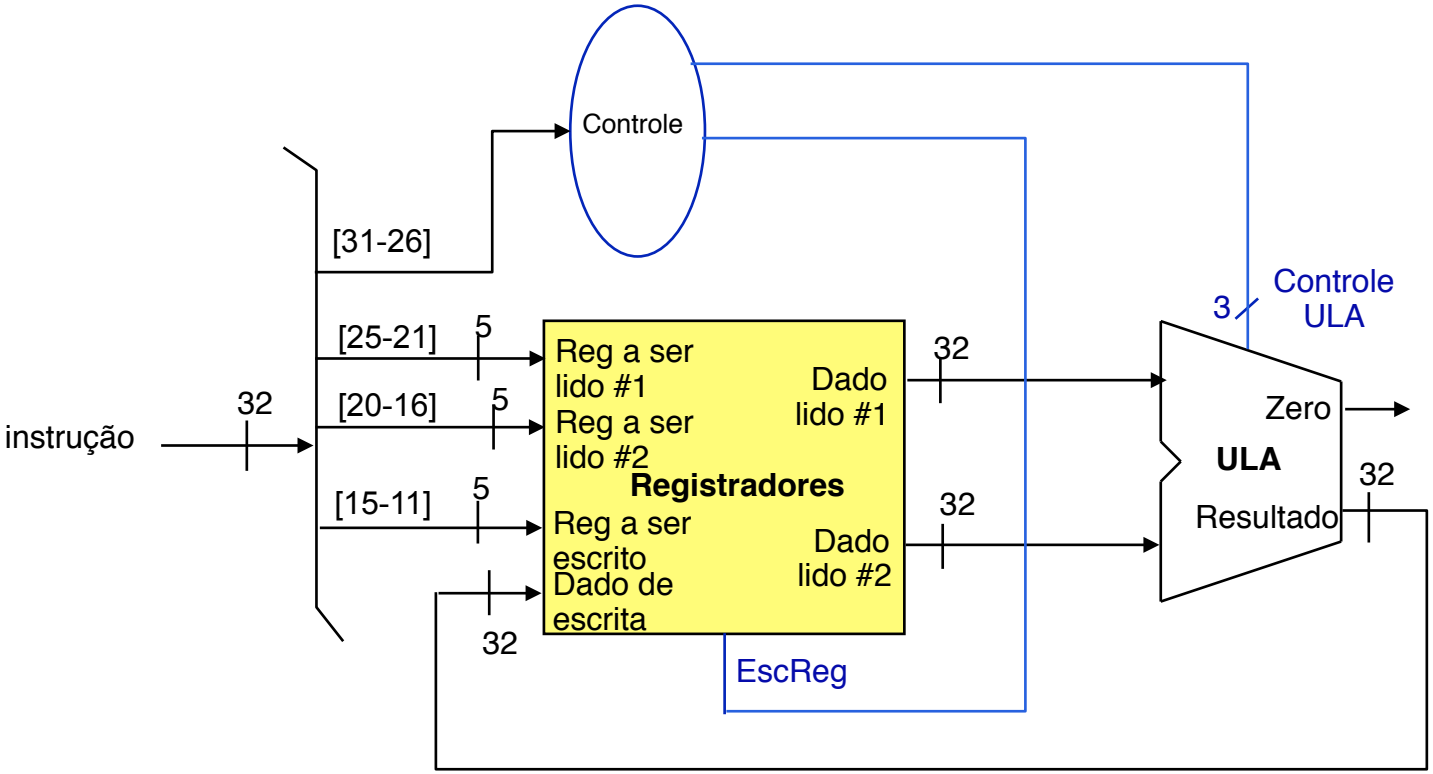
Exemplo 2: Banco de registradores de um microprocessador



- 2 “portas” de leitura e 1 “porta” de escrita (há um sinal p/ habilitar escrita)
- Quantos registradores há neste banco de registradores?
- Qual o comprimento (ou tamanho) dos dados?

Registradores x Banco de Registradores

Exemplo 2: Banco de registradores de um microprocessador



Instruções: o Processador MIPS

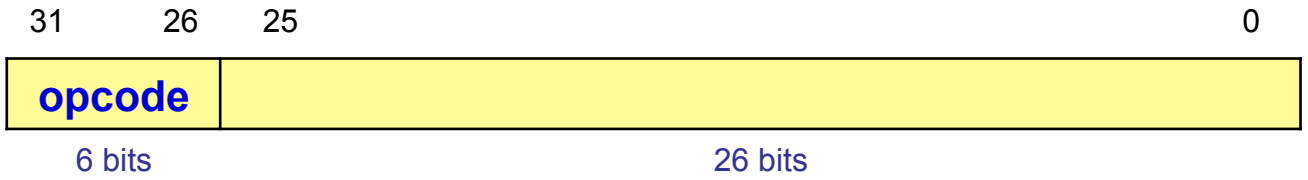
Referência

PATTERSON, David A.; HENNESSY, John L. "Computer Organization and Design: The Hardware/Software Interface", 3rd edition, Morgan Kaufmann Publishers, San Francisco, California, USA, 2007.

Edições mais antigas também servem!

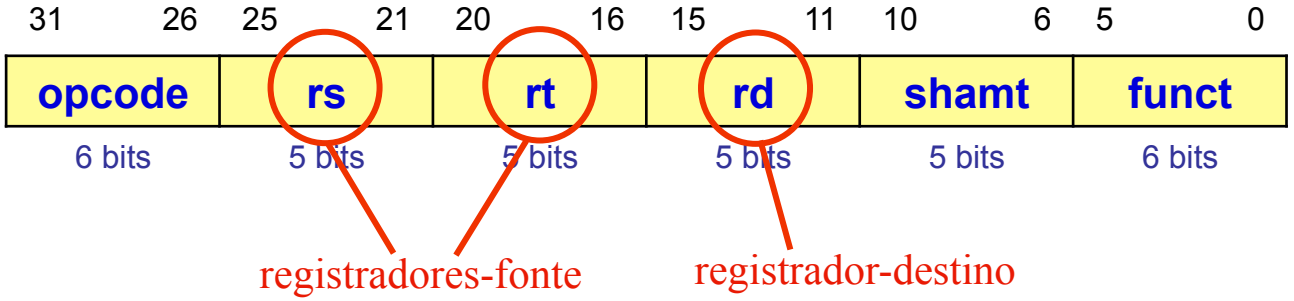
Instruções: o Processador MIPS

- todas as instruções têm 32 bits
- todas têm opcode de 6 bits
- o modo de endereçamento é codificado juntamente com o opcode



Instruções formato R: add, sub, or, and

- opcode = 0
- "funct" define a operação a ser feita pela ALU
- "shamt" (shift amount) é usado em instruções de deslocamento

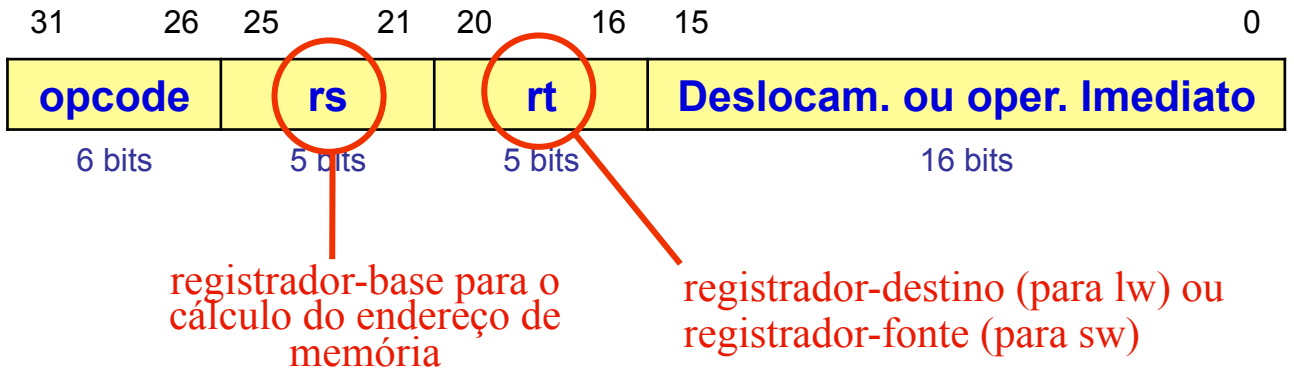


Simbólico (exemplo): `add $s1,$s2, $s3` (`$s1 ← $s2 + $s3`)

4. Projeto de Sistemas Digitais no Nível RT

Instruções formato I: load word (ld) e store word (sw)

- **load word (lw): opcode = 35**
- **store word (sw): opcode = 43**



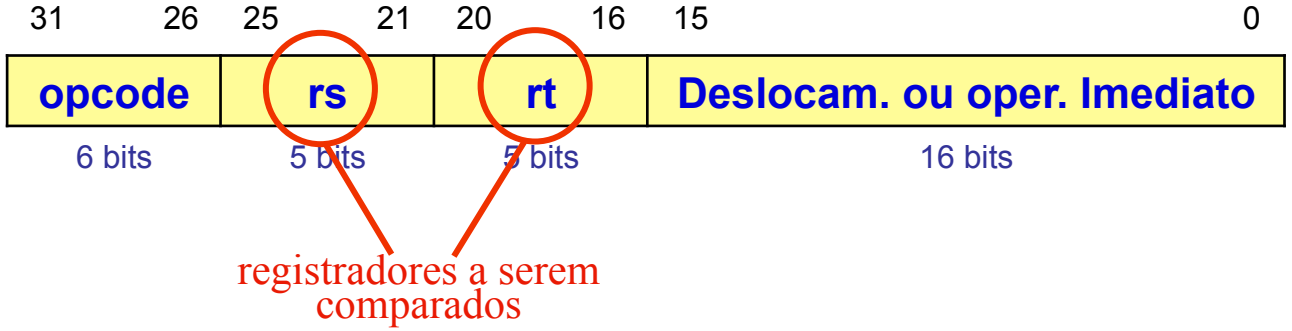
Simbólico

```
lw $s1, deslocam($s2)   ($s1 ← Mem[$s2 + deslocam] )
sw $s1, deslocam($s2)   (Mem[$s2 + deslocam] ← $s1 )
```

Instrução formato I: Desvio Condicional

beq: branch on equal

- **Opcode = 4**
- **Campo deslocamento usado para calcular o endereço-alvo**
- **Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4**



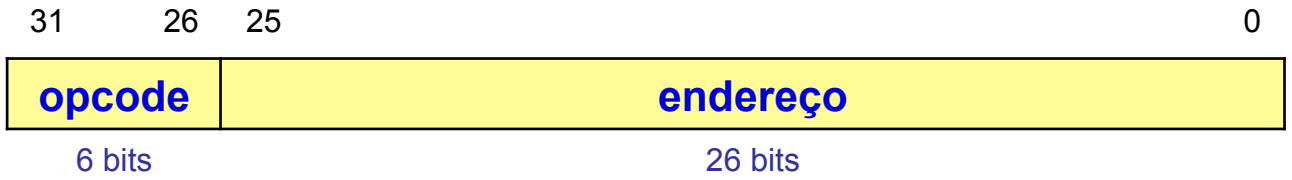
Simbólico

```
beq $s1, $s2, deslocam   ( if ($s1== $s2) then PC←PC+4+deslocam)
```

Instrução formato J: Desvio Incondicional

j: jump

- **Opcode = 2**
- **Campo deslocamento usado para calcular o endereço-alvo**
- **Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4**

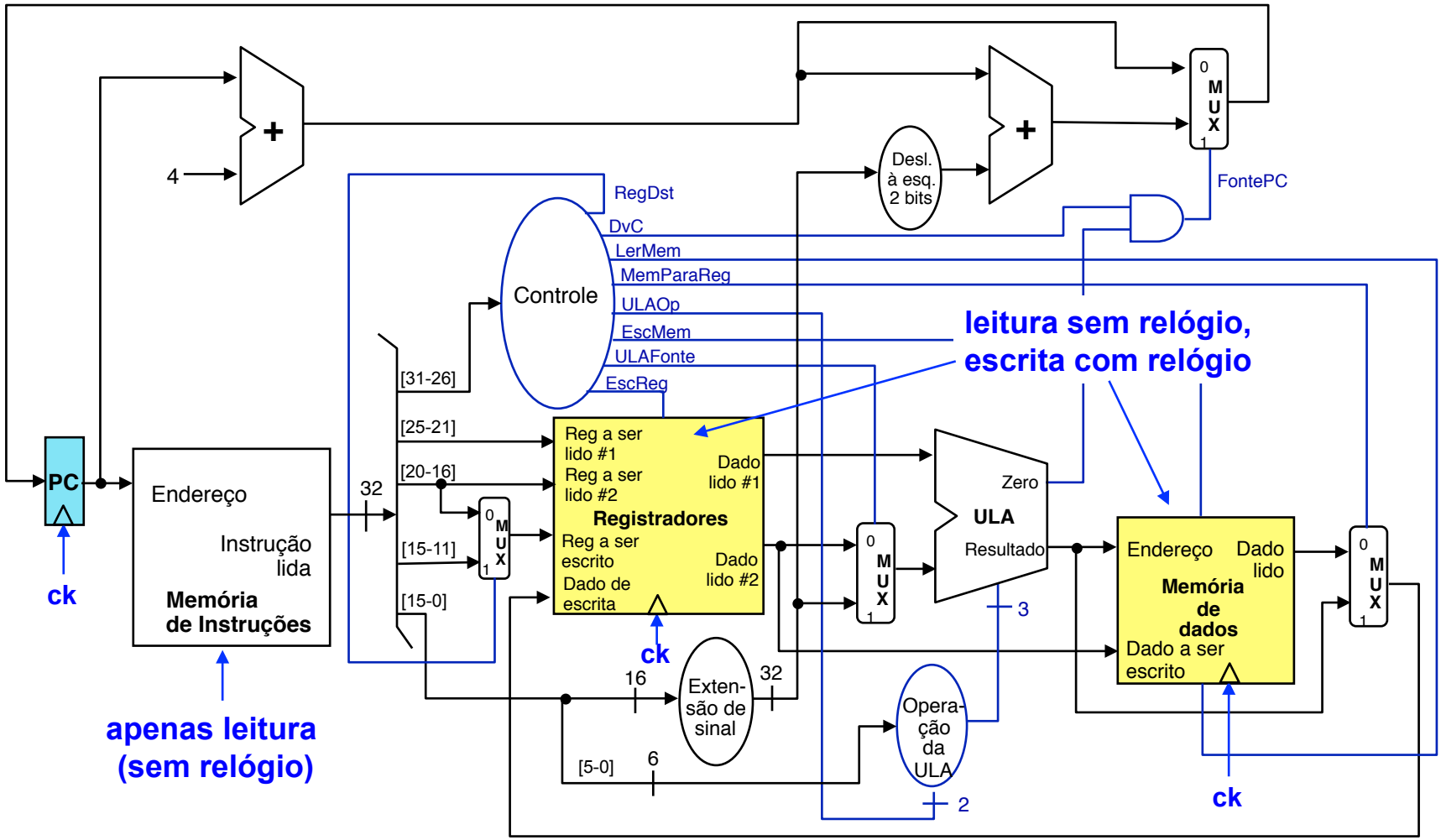


Simbólico

```
j endereço   ( PC←endereço )
```

4. Projeto de Sistemas Digitais no Nível RT

O Processador MIPS (Monociclo): BO + BC



Execução de uma Instrução Tipo R

Seja uma instrução tipo R, como por exemplo add $t1, $t2, $t3**:**

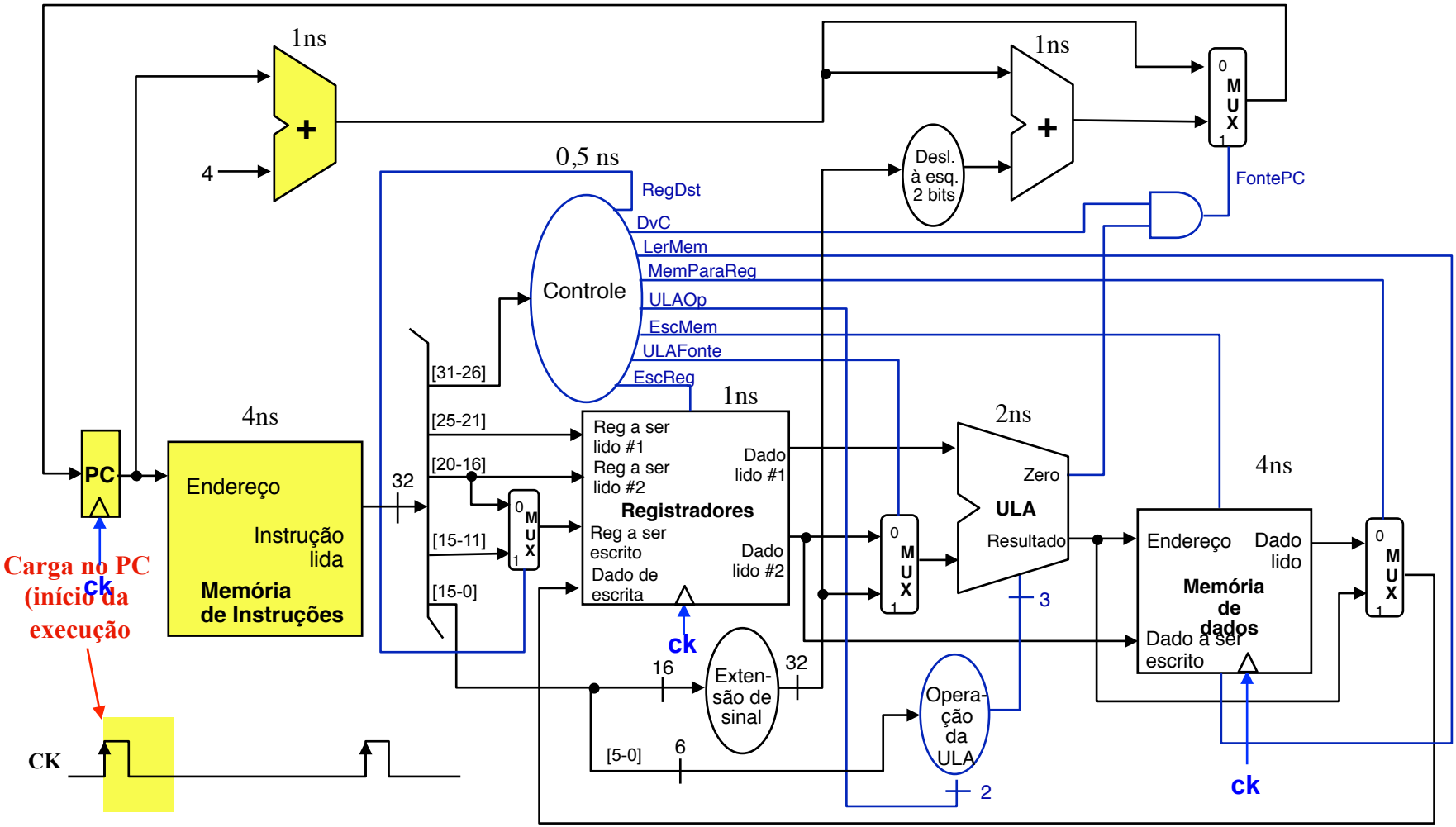
Podemos imaginar que esta instrução é executada em 4 etapas:

1. **Busca da instrução (na memória de instruções) e incremento do PC**
2. **Leitura de dois registradores (no caso, $t2 e $t3, ou Rs e Rt) e geração dos sinais de controle para o resto do bloco operativo (decodificação da instrução)**
3. **Operação na ULA**
4. **Escrita (do resultado da operação realizada na ULA) no registrador destino ($t1 ou Rd)**

Como estes passos ocorrem dentro do mesmo ciclo de relógio (regime monociclo), a ordem real irá depender do atraso de cada componente.

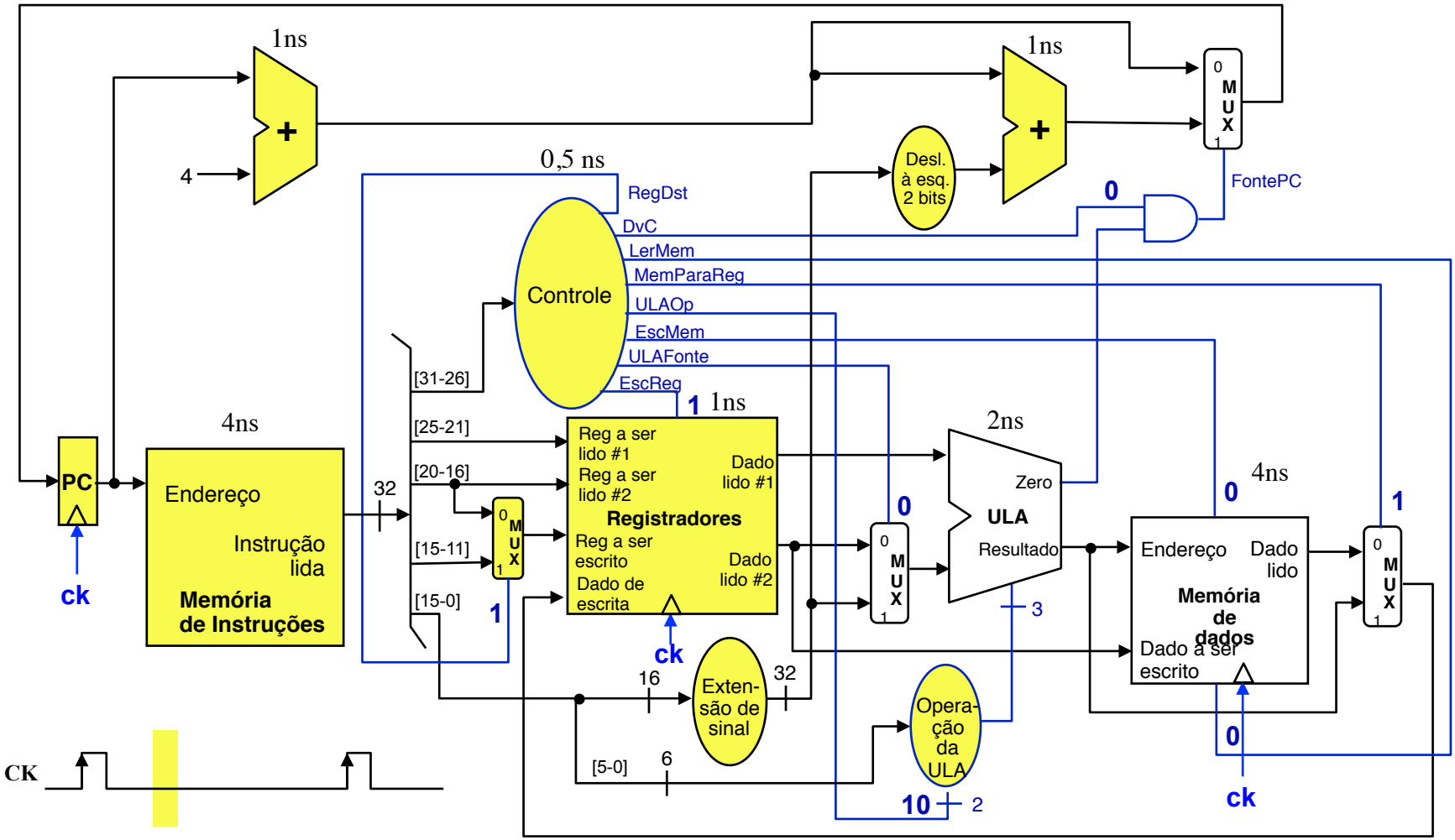
4. Projeto de Sistemas Digitais no Nível RT

Instrução Tipo R: busca da instrução e cálculo de PC+4



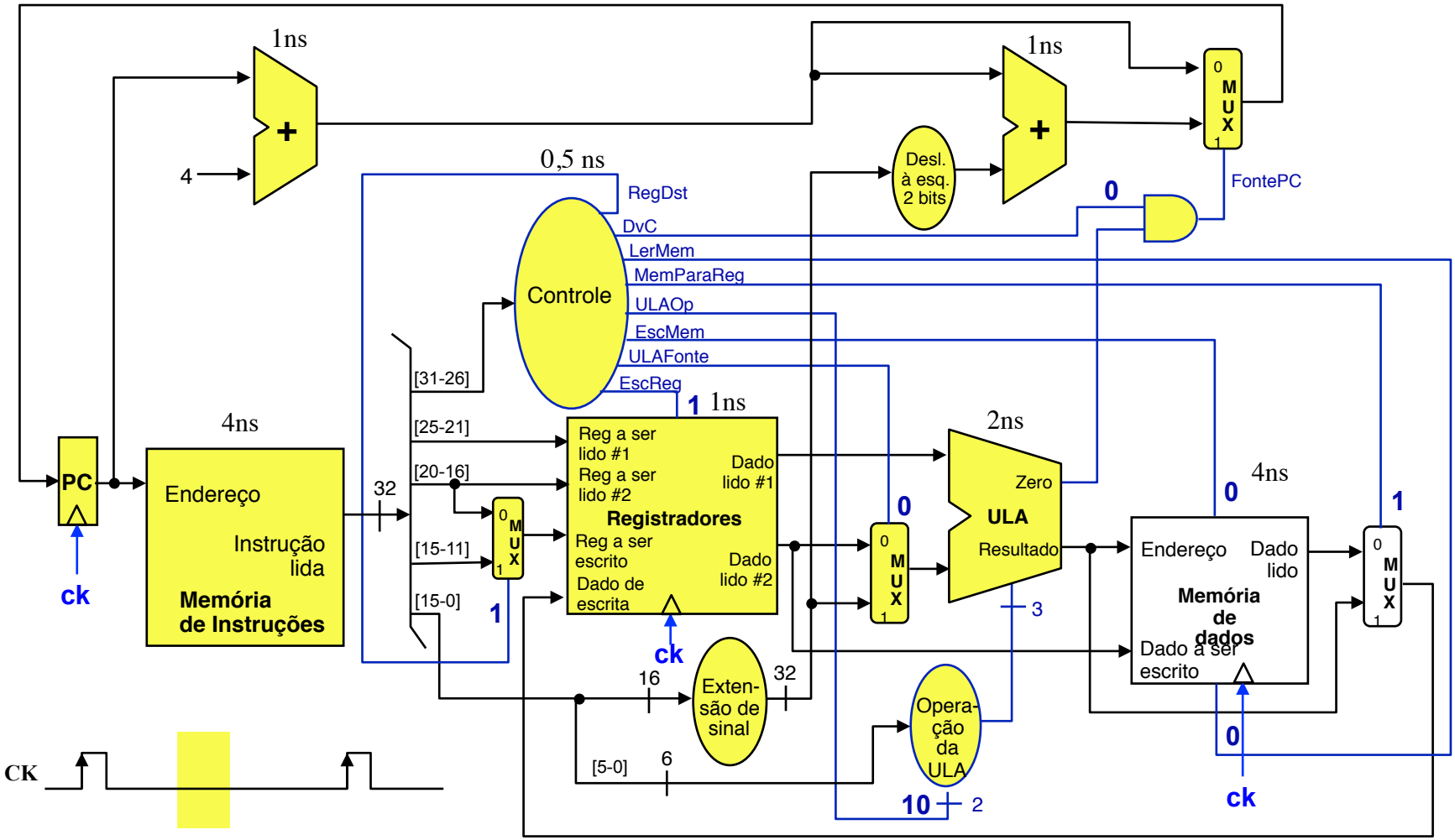
4. Projeto de Sistemas Digitais no Nível RT

Instrução Tipo R: leit. de Rs e Rt e geração sinais de controle



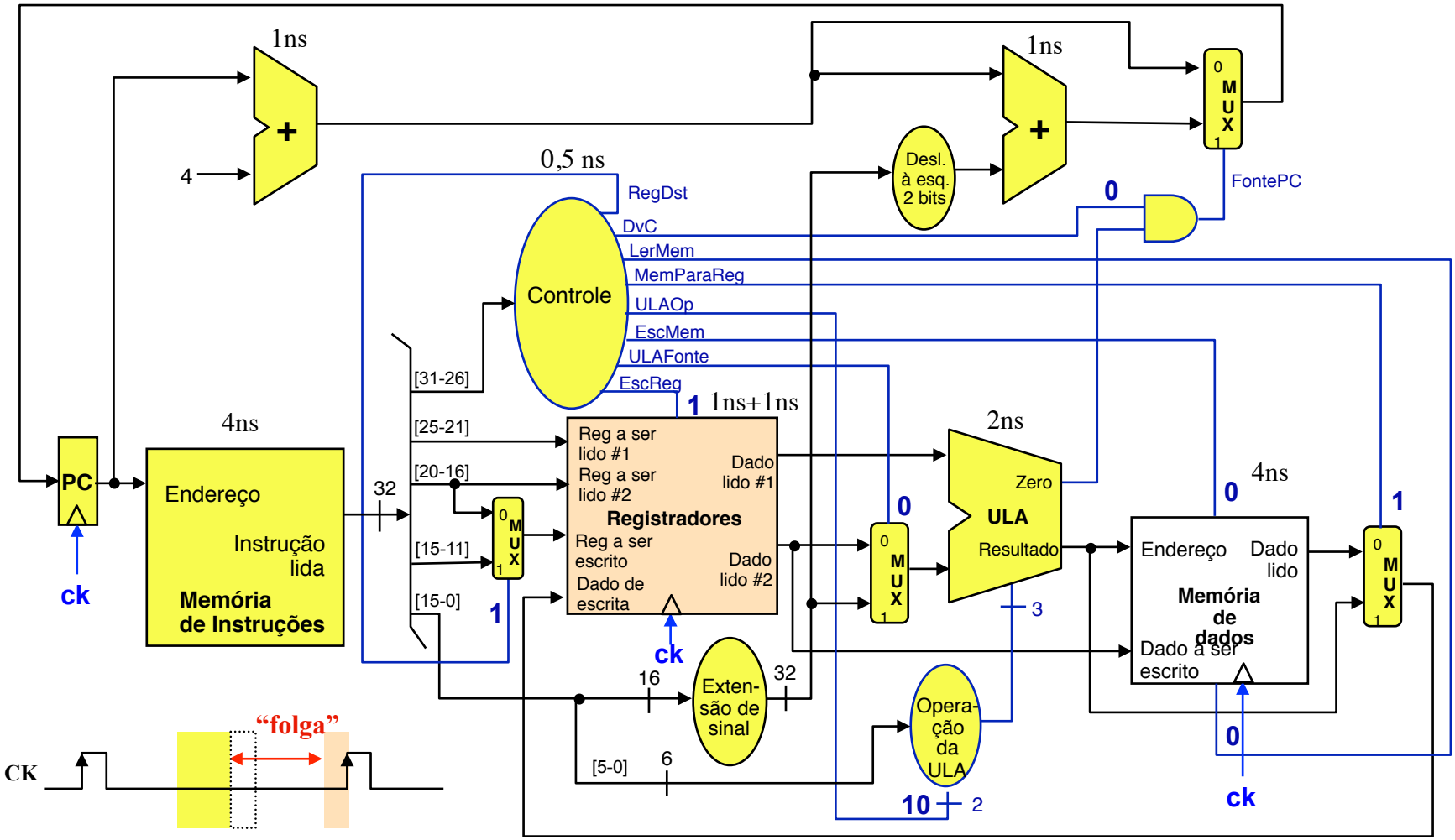
4. Projeto de Sistemas Digitais no Nível RT

Instrução Tipo R: operação na ULA (depende de "funct")



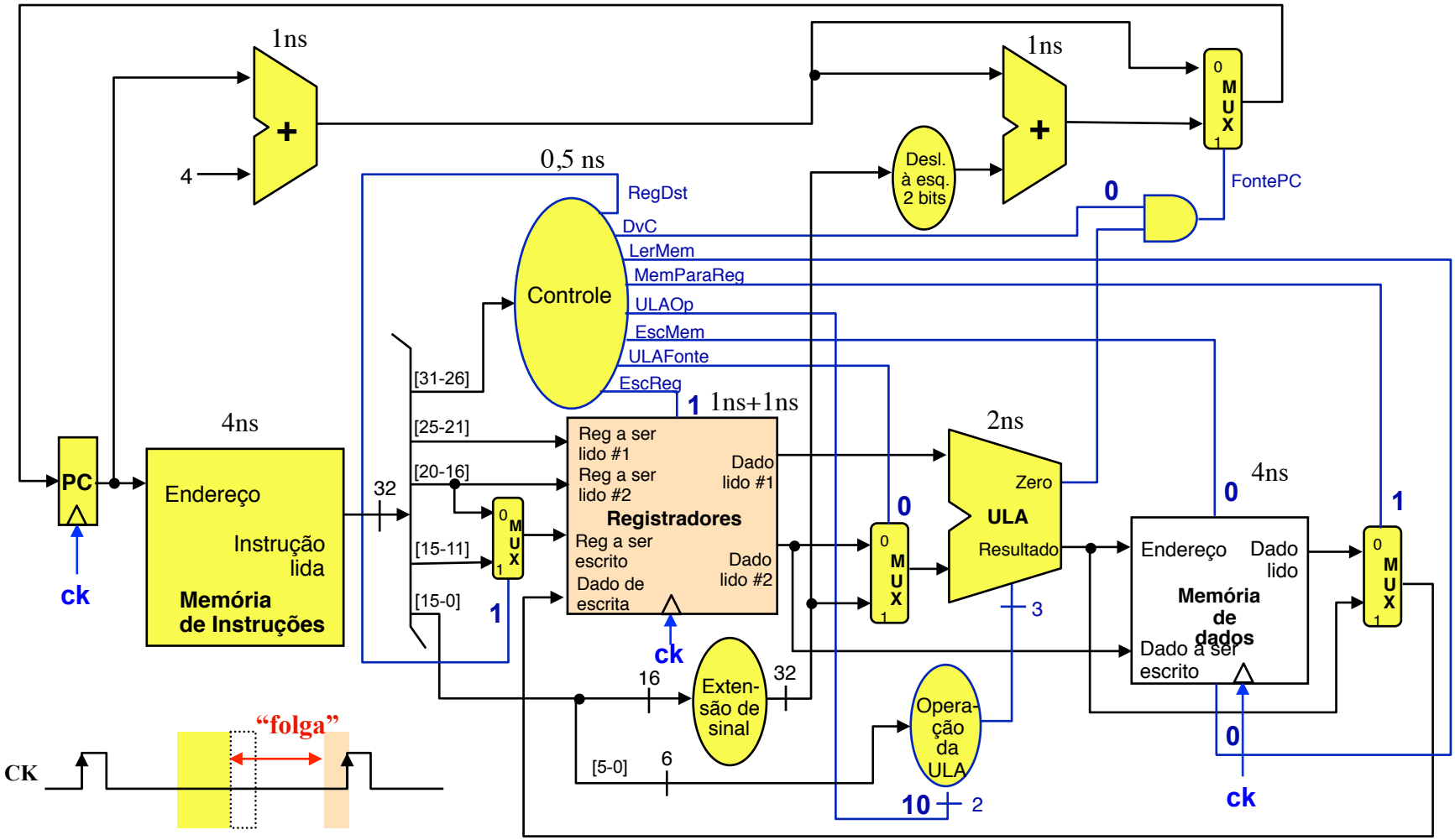
4. Projeto de Sistemas Digitais no Nível RT

Instrução Tipo R: escrita no registrador-destino



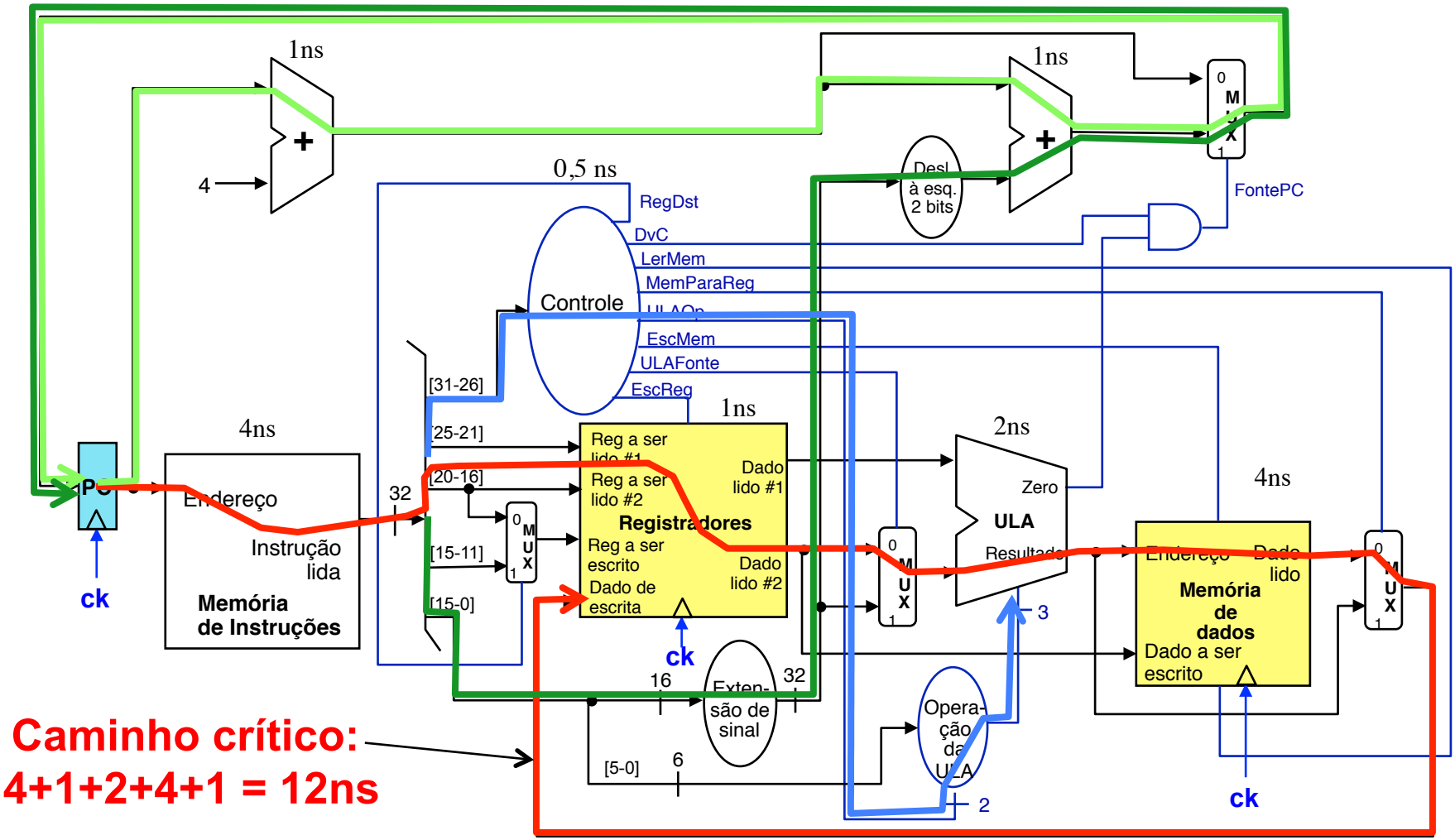
4. Projeto de Sistemas Digitais no Nível RT

Instrução Tipo R: escrita no registrador-destino

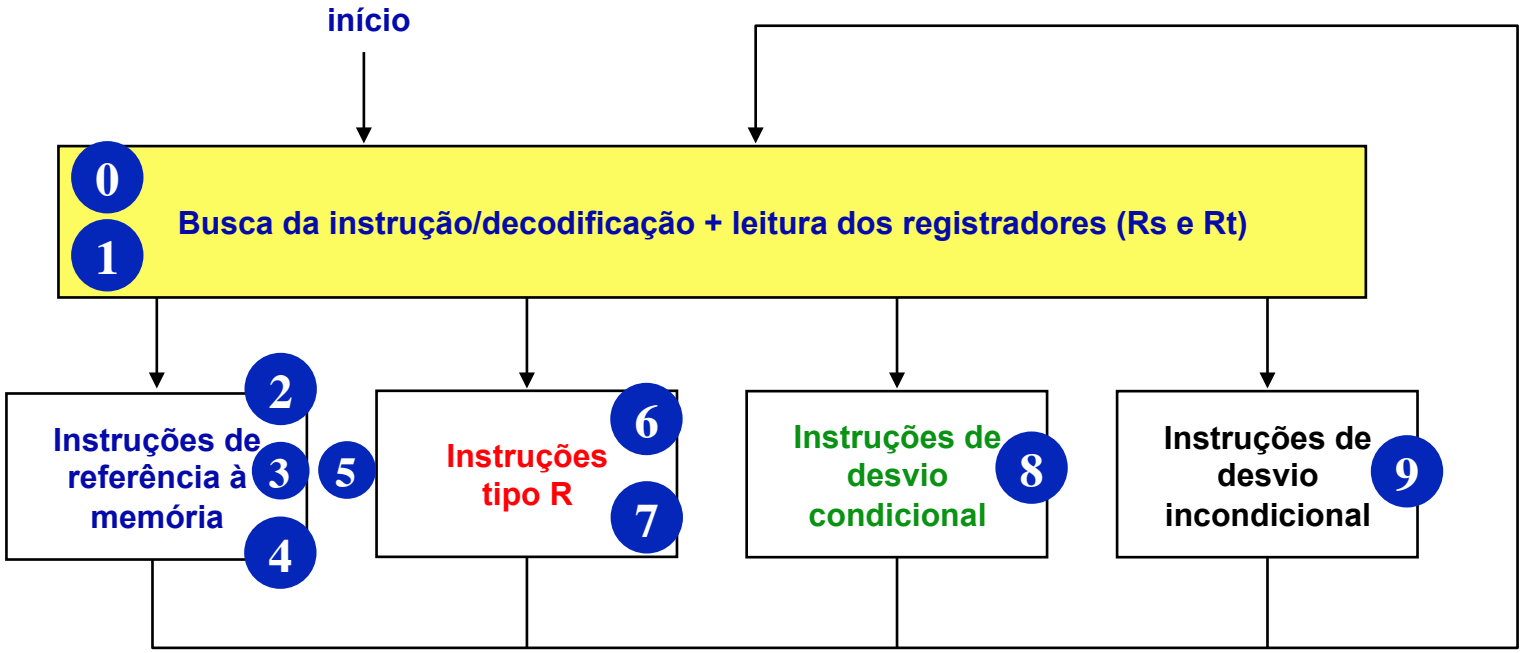


4. Projeto de Sistemas Digitais no Nível RT

O Processador MIPS: BO + BC



Implementação Multiciclo



Cada caixa pode representar um ou mais passos e portanto, pode executar em um ou mais ciclos de relógio (dependendo da instrução).

4. Projeto de Sistemas Digitais no Nível RT

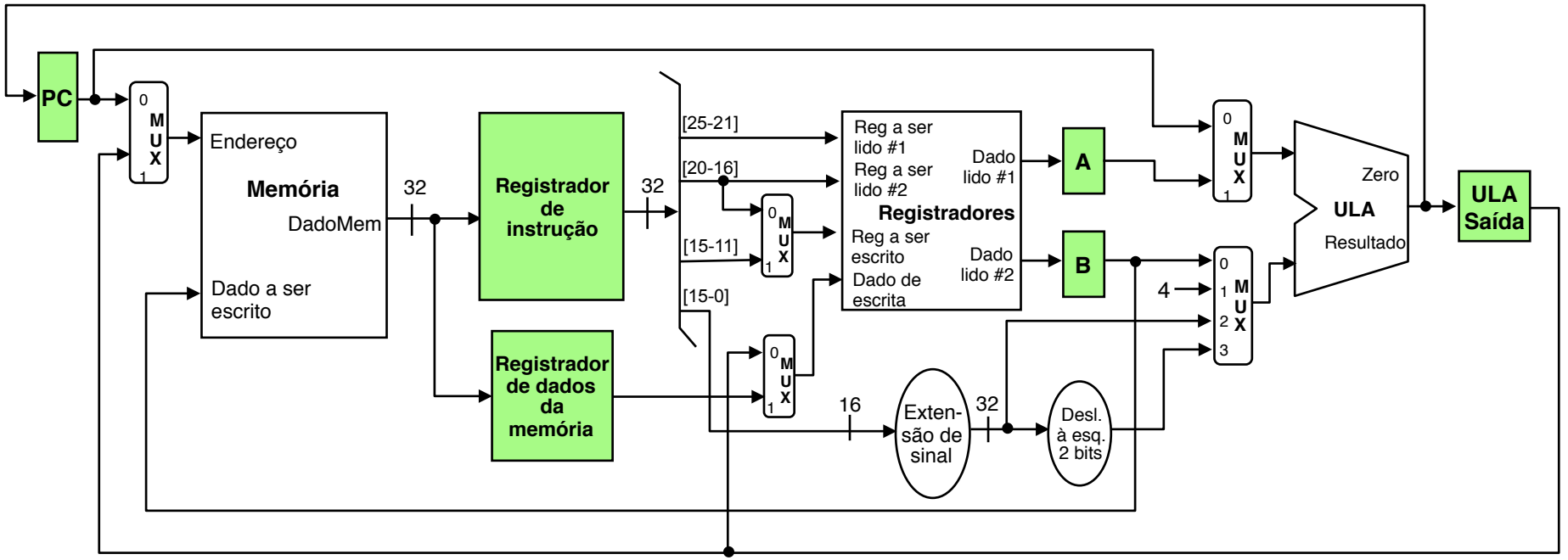
Resumo dos Passos Necessários para Cada Instrução

Nome do passo	Instrução tipo R	Instrução lw	Instrução sw	Instrução beq	Instrução j
Busca da instrução	RI = Mem[PC] PC = PC + 4				
Decodificação da instrução & leitura dos registradores Rs e Rt & cálculo do endereço de desvio (cond.)	A = Reg [RI[25-21]] B = Reg [RI[20-16]] ULASaída = PC + (extensão de sinal(RI[15-0]) <<2)				
Execução, cálculo do endereço de acesso à memória, término de uma instrução branch/jump	ULAOp = A op	ULASaída = A + extensão de s (RI[15-0])		Se (A == B) e PC = ULASaída	PC = PC[(RI[25-0] <<
Término de uma instrução store word ou de tipo R	Reg [RI[15-11] ULASaída	RDM = Mem [ULASaída]	Mem [ULASaí = B		
Término de uma instrução load word		Reg[RI[20-1 = RDM			
Número de passos	4	5	4	3	3

4. Projeto de Sistemas Digitais no Nível RT

Bloco Operativo Multiciclo

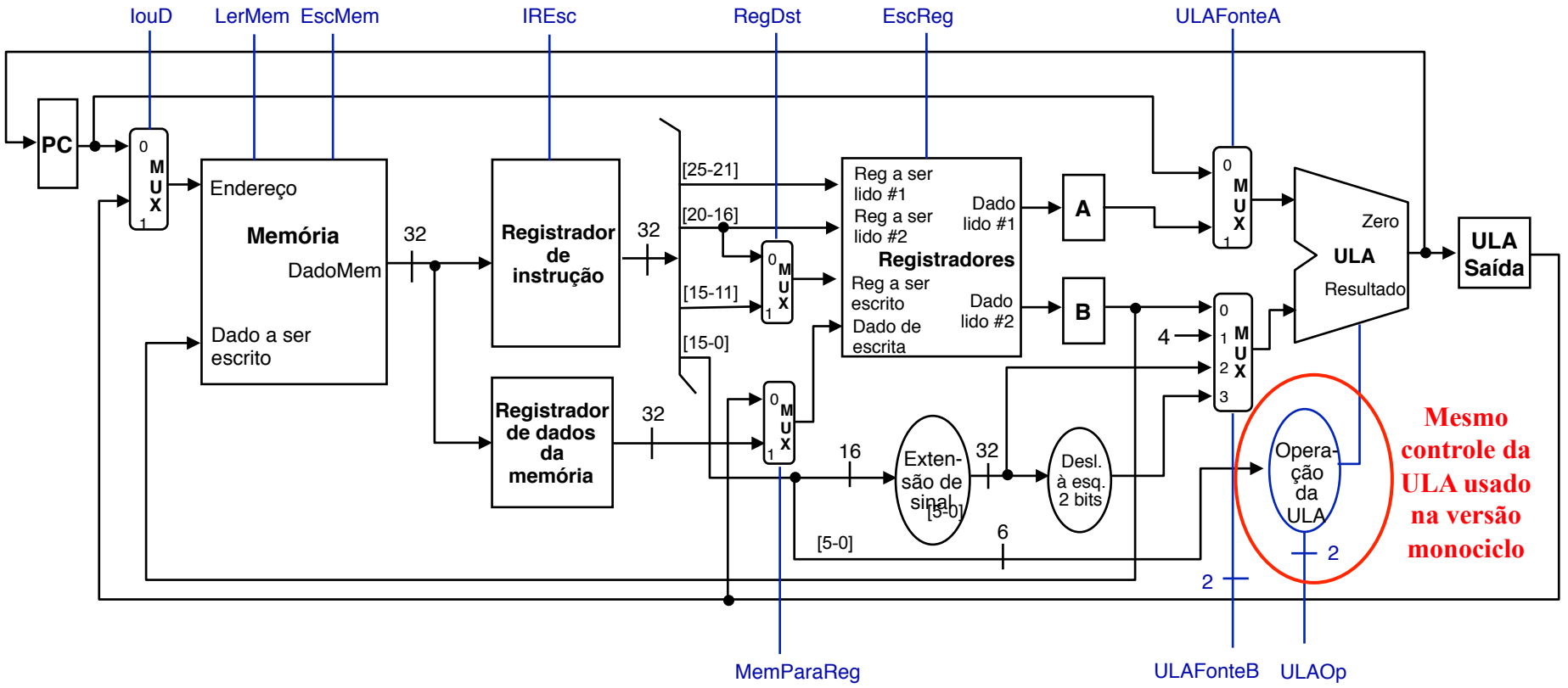
Registradores Não-Visíveis ao Programador



4. Projeto de Sistemas Digitais no Nível RT

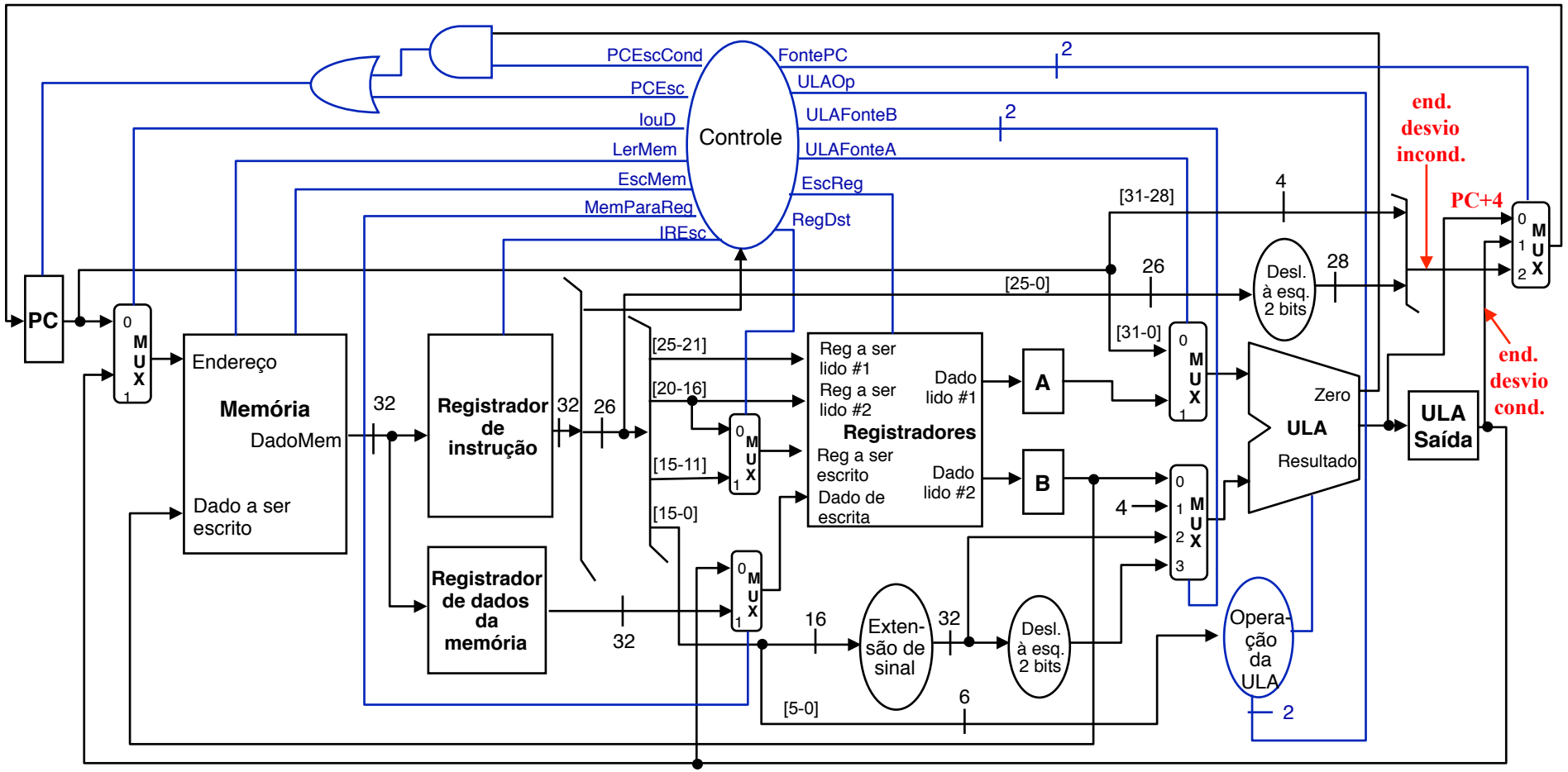
Bloco Operativo Multiciclo

Os Sinais de Controle



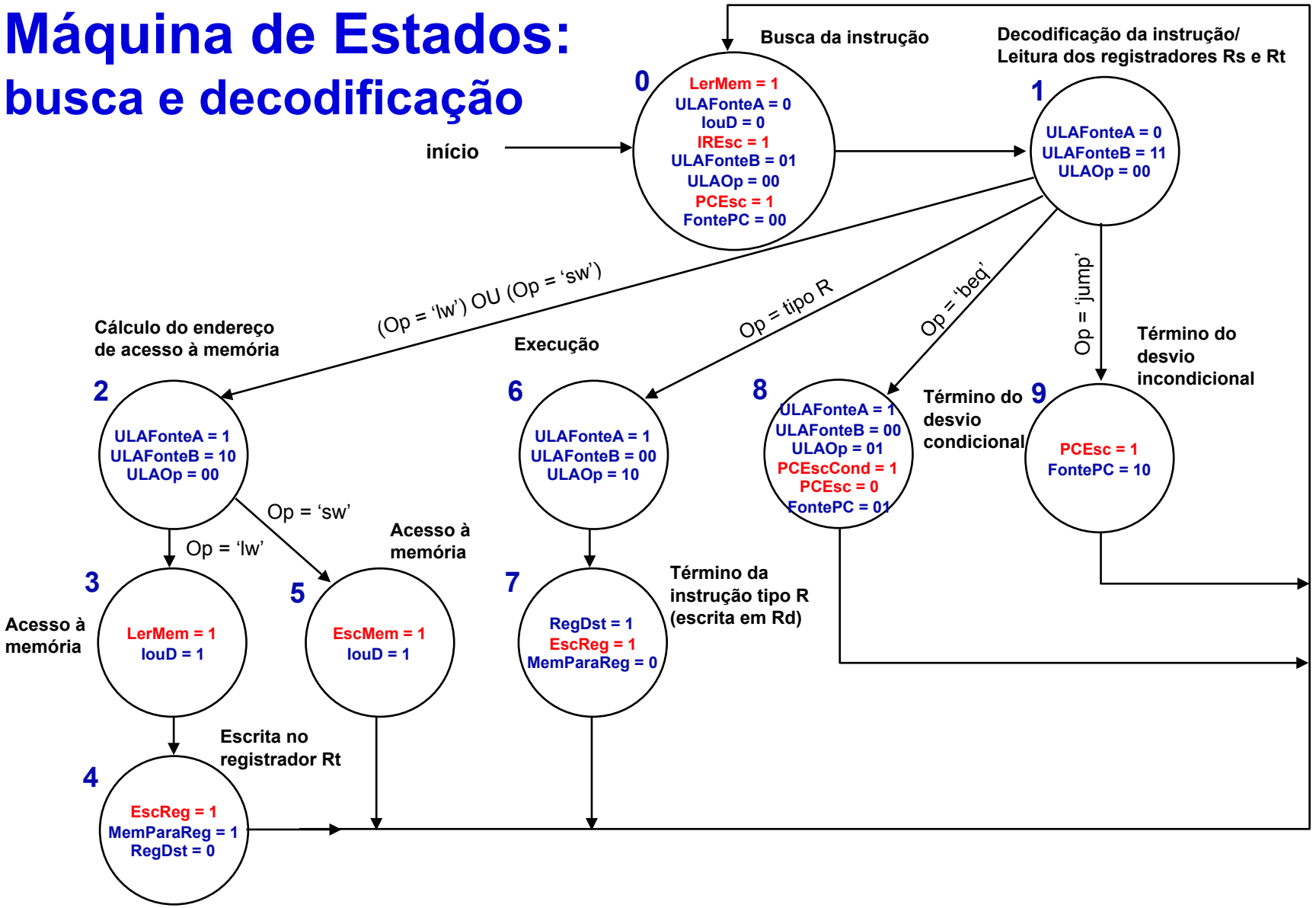
4. Projeto de Sistemas Digitais no Nível RT

Sinais de Controle do Bloco Operativo Multiciclo



4. Projeto de Sistemas Digitais no Nível RT

Máquina de Estados:
busca e decodificação



4. Projeto de Sistemas Digitais no Nível RT

Caminhos Críticos (para Estimar o Relógio)