



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



# Sistemas Digitais

INE 5406

## Aula 13-T

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

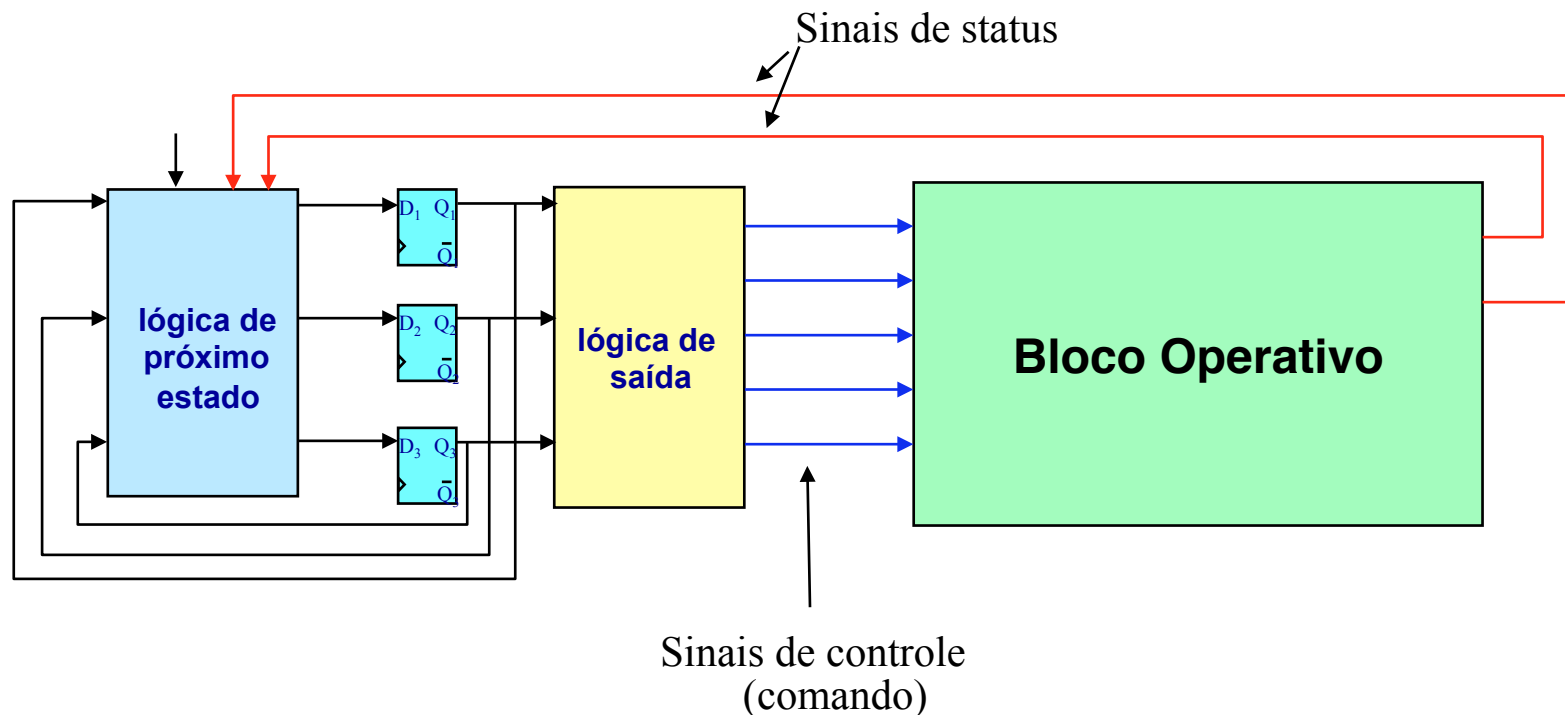
**Prof. José Luís Güntzel**  
guntzel@inf.ufsc.br

[www.inf.ufsc.br/~guntzel/ine5406/ine5406.html](http://www.inf.ufsc.br/~guntzel/ine5406/ine5406.html)

# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ **Análise de *Timing***

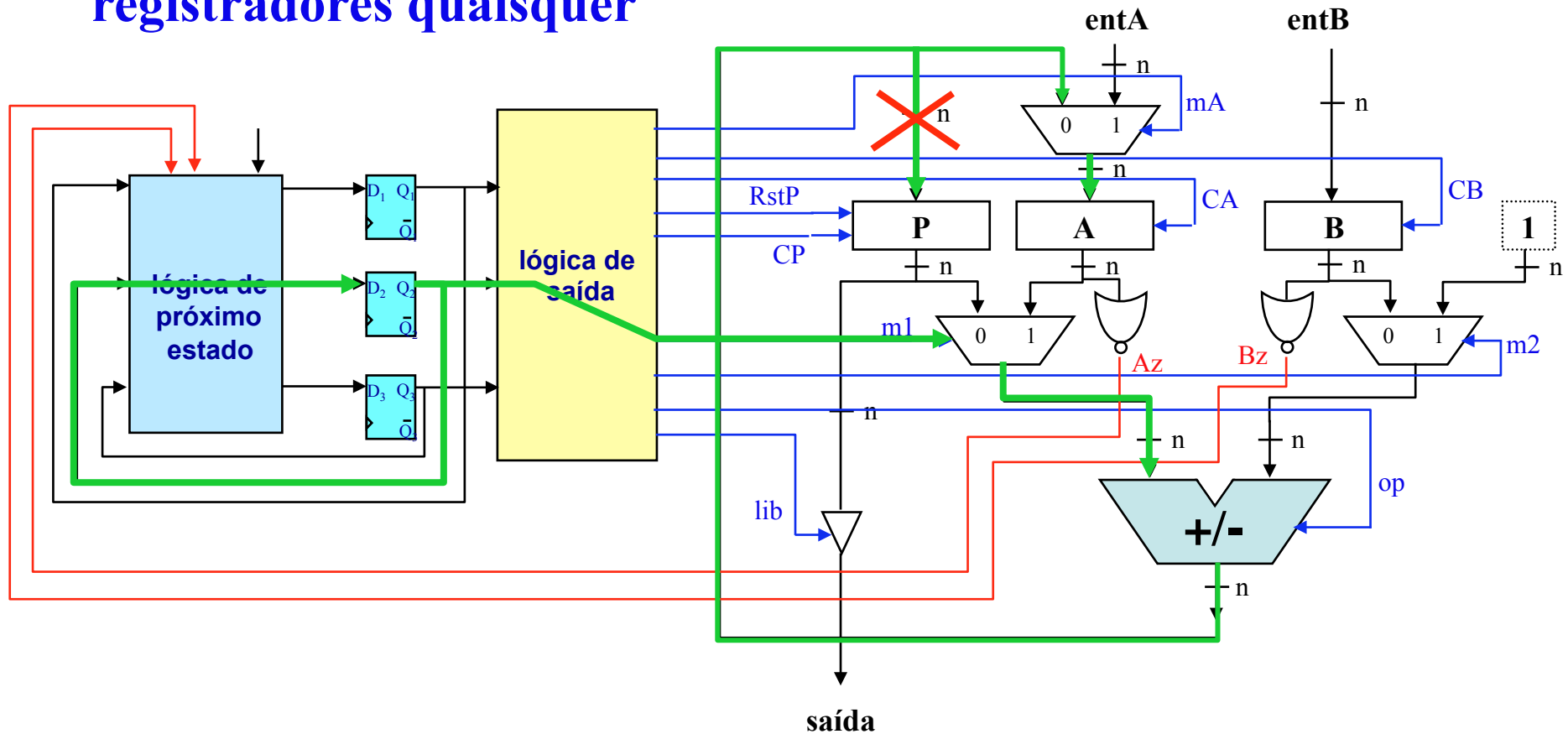
**Objetivo: encontrar o caminho mais longo entre dois registradores quaisquer (considerando BC & BO)**



# 4. Projeto de Sistemas Digitais no Nível RT

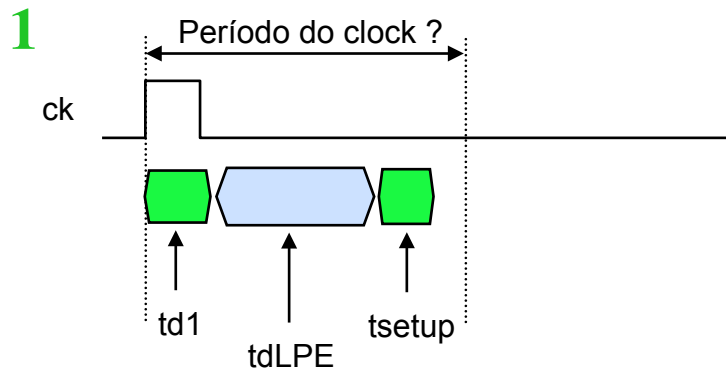
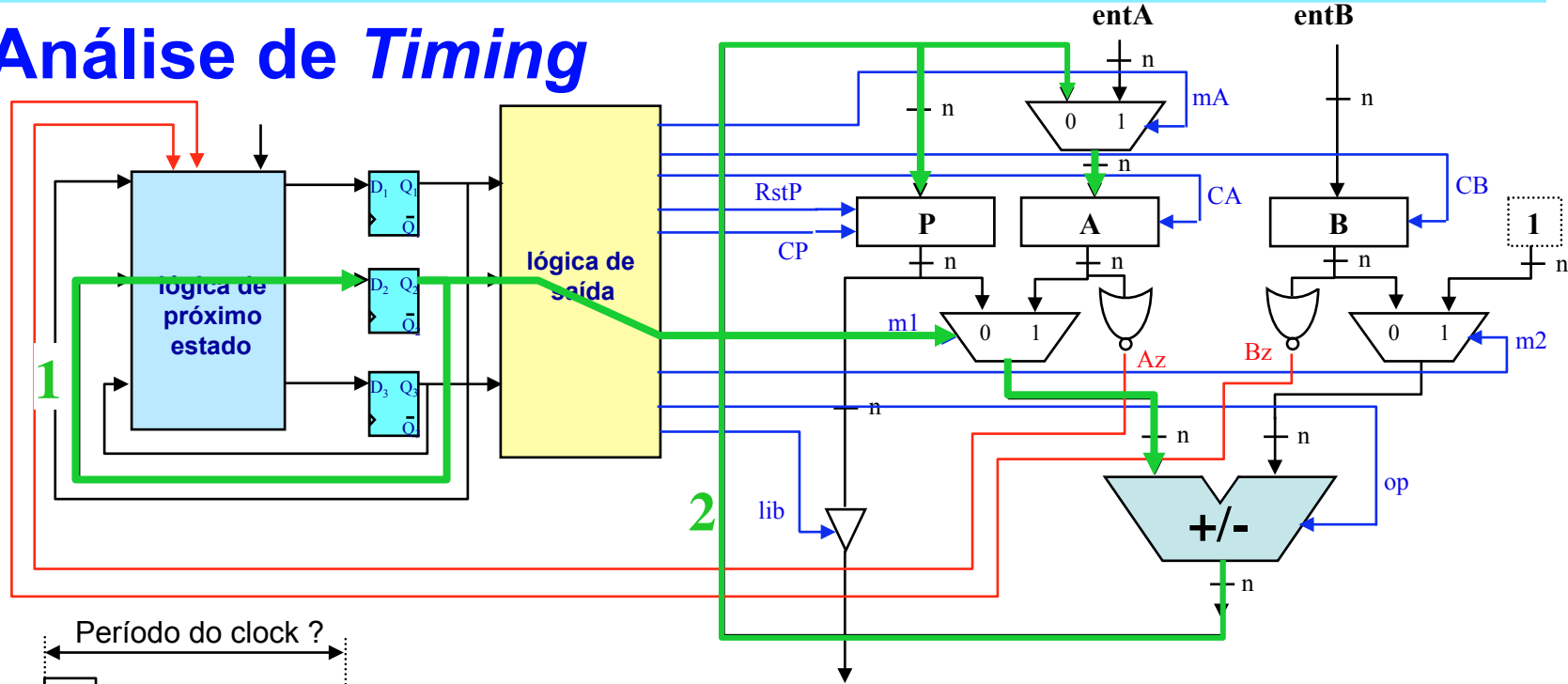
## ► Análise de *Timing*

Objetivo: encontrar o caminho mais longo entre dois registradores quaisquer



# 4. Projeto de Sistemas Digitais no Nível RT

## Análise de *Timing*



$td1 = \max \{ tdRegEst, thold \}$ , onde:

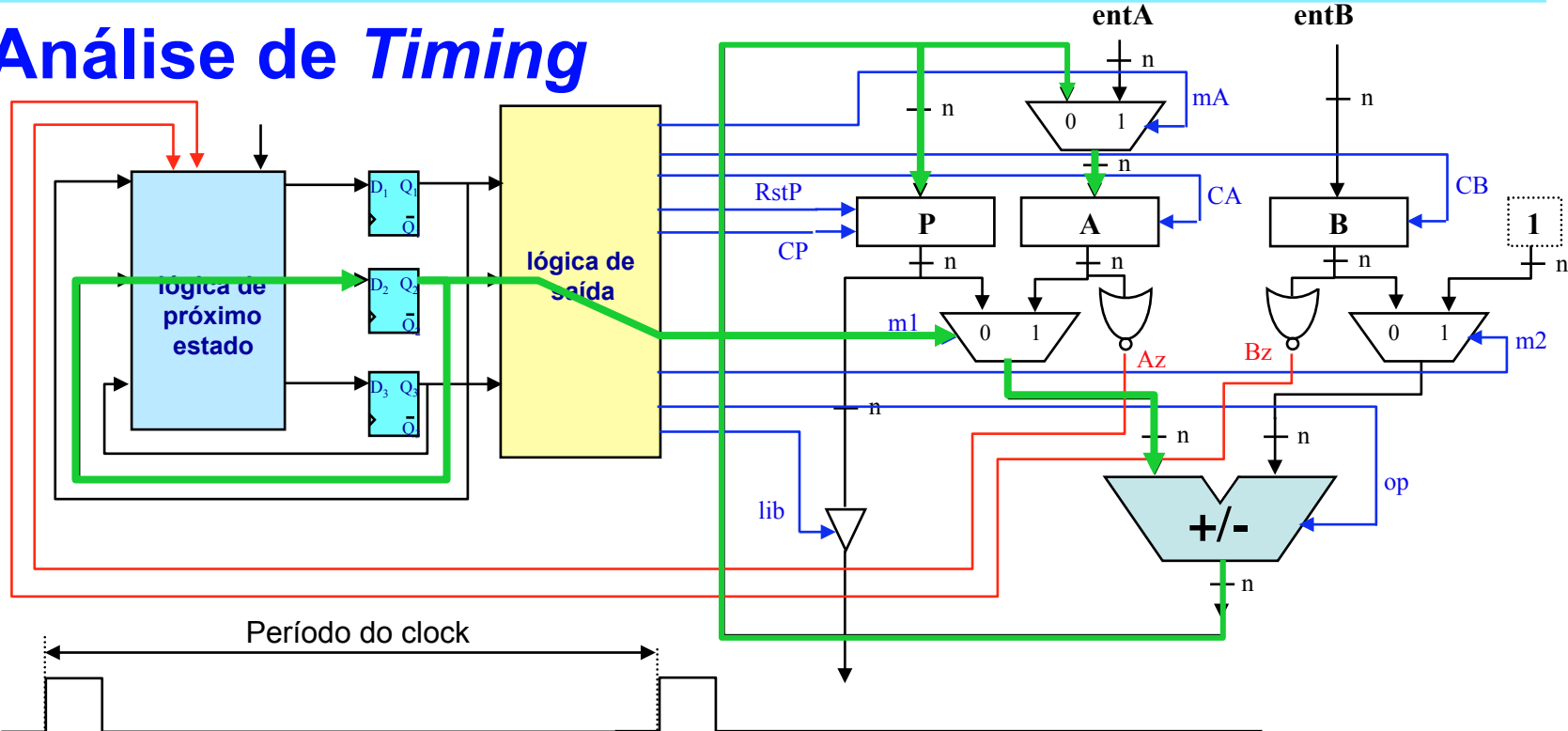
- $tdRegEst$  = atraso de propagação do reg. de estado
- $thold$  = tempo de manutenção (hold) do reg. de estado

$tdLPE$  = atraso e propagação da lógica de próximo estado

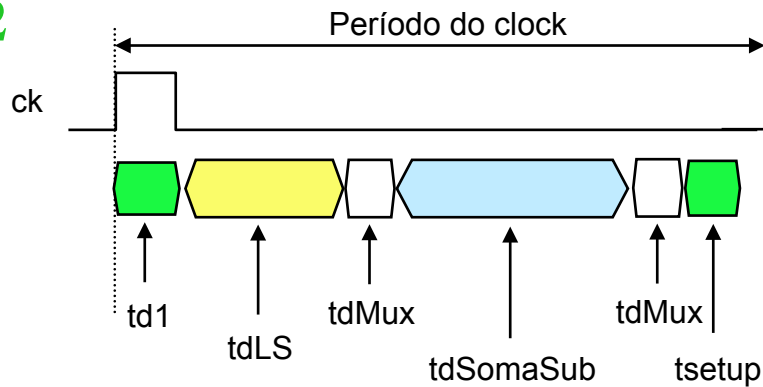
$tdSU$  = tempo de *setup* do registrador de estado

# 4. Projeto de Sistemas Digitais no Nível RT

## Análise de *Timing*



2



$$td1 = \max \{ tdRegEst, thold \}$$

$tdLS$  = atraso e propagação da lógica de saída

$tdMux$  = atraso de propagação de um mux 2:1

$tdSomaSub$  = atraso de propagação do somador-subtrator

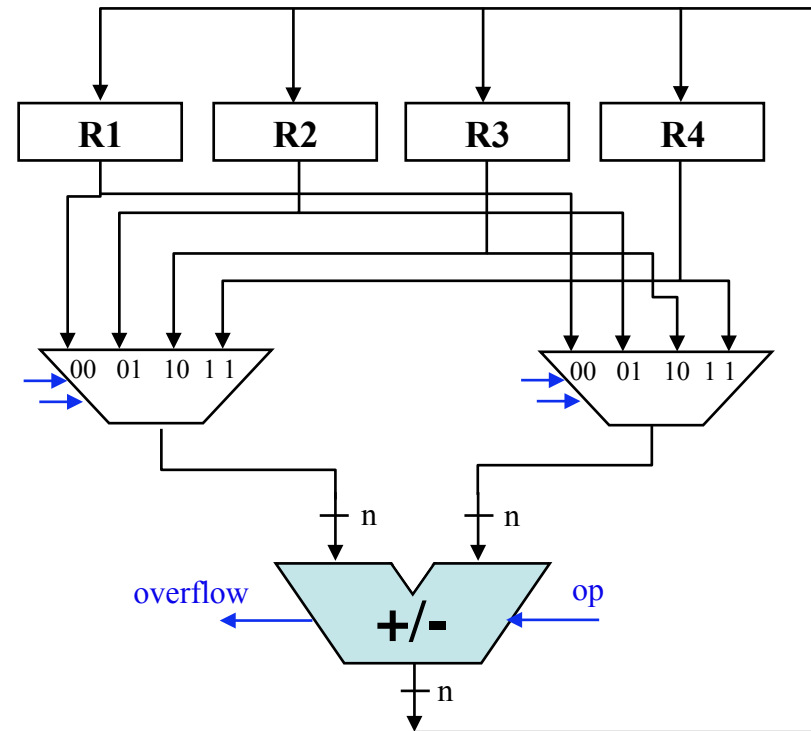
$tdMux$  = atraso de propagação de um mux 2:1

$tsetup$  = 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

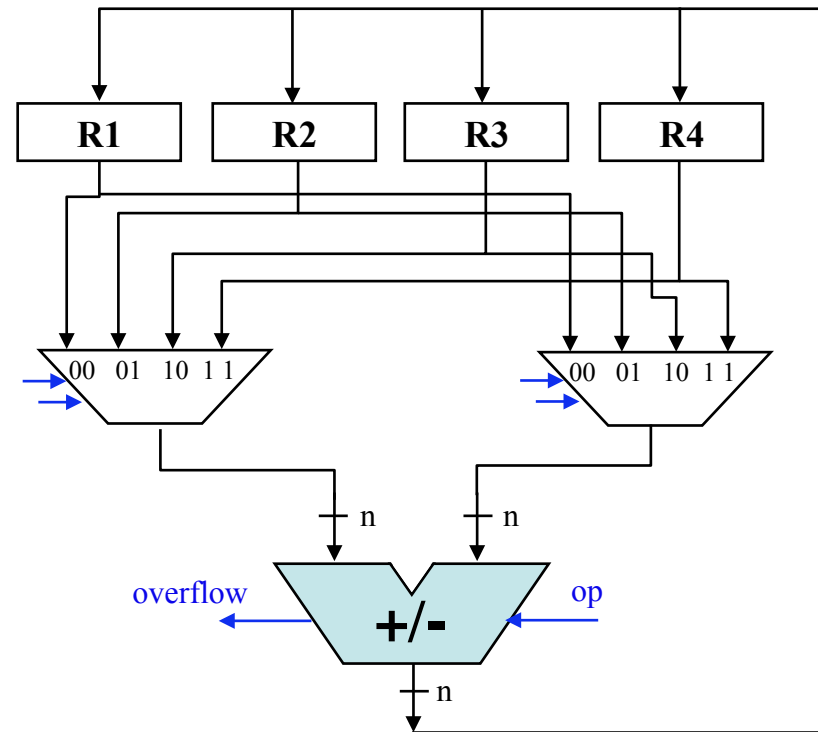


# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ 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)

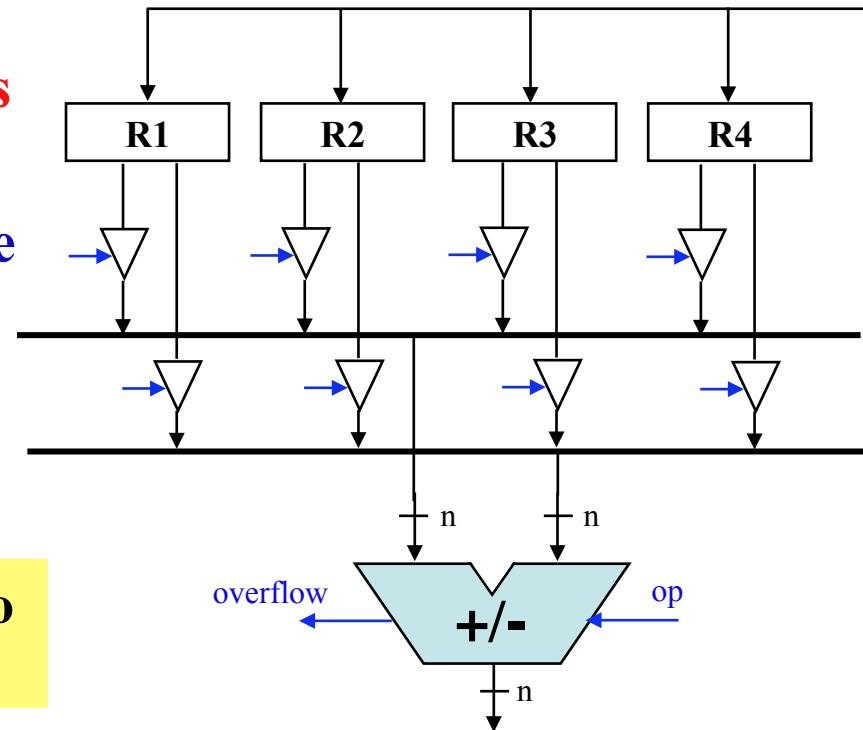


# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ 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!



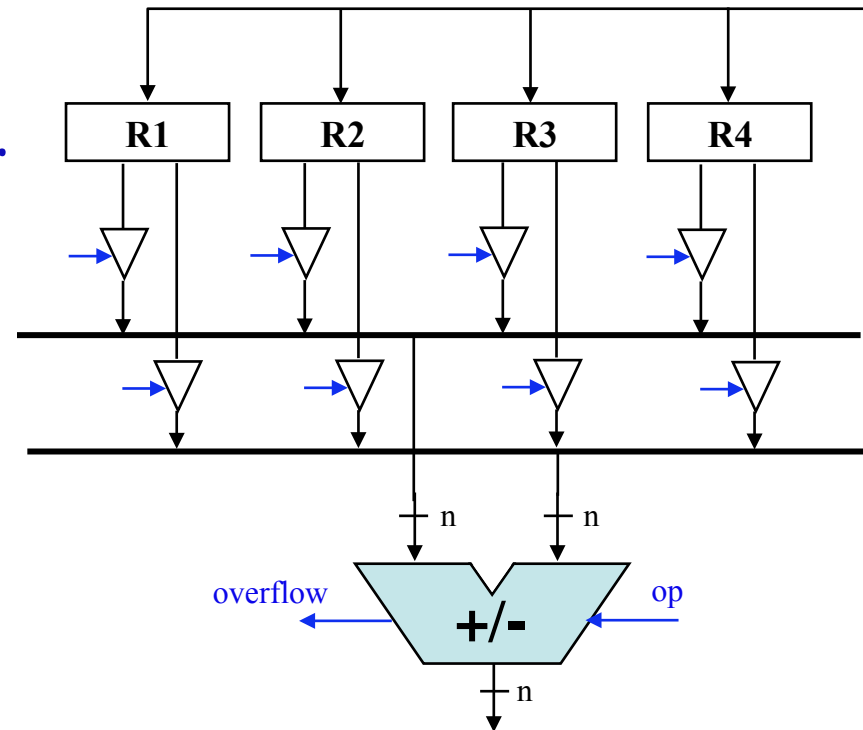


# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ 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

---

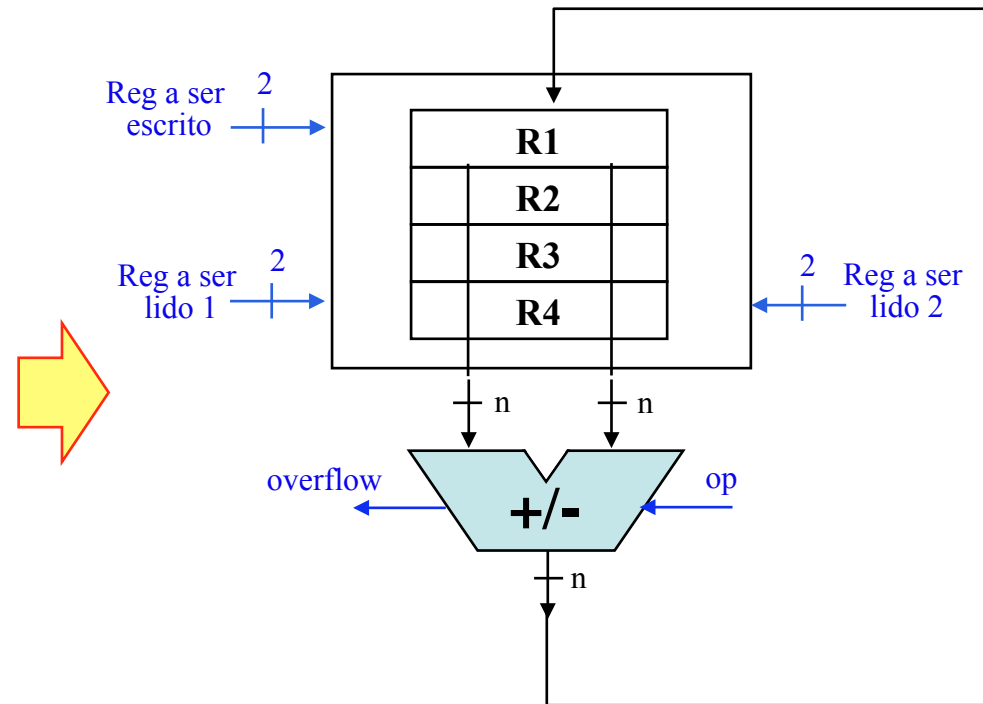
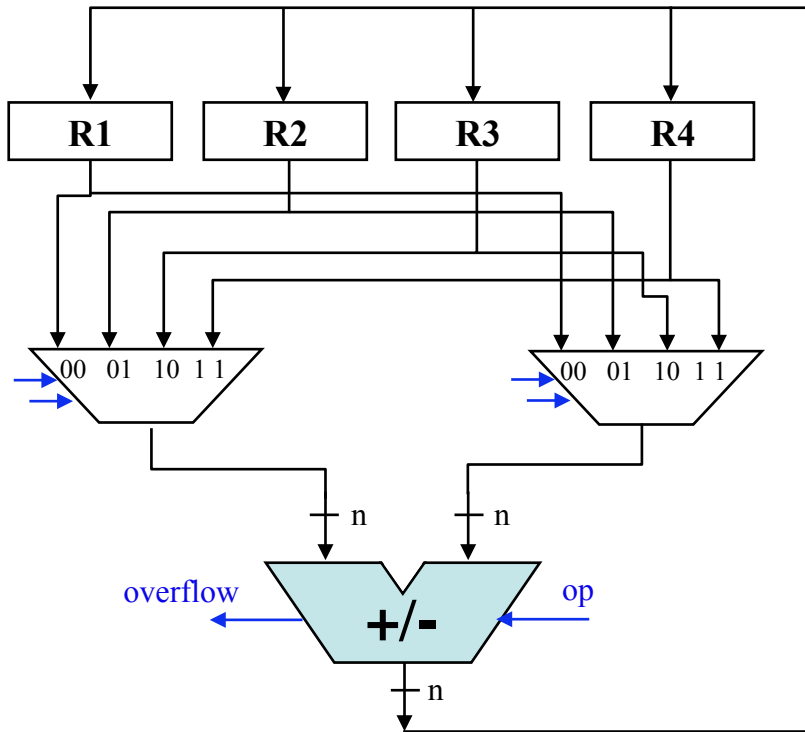
## ▶ Registradores x Banco de Registradores

- Se houver uma quantidade grande de registradores ( $\geq 4$ )
- 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:

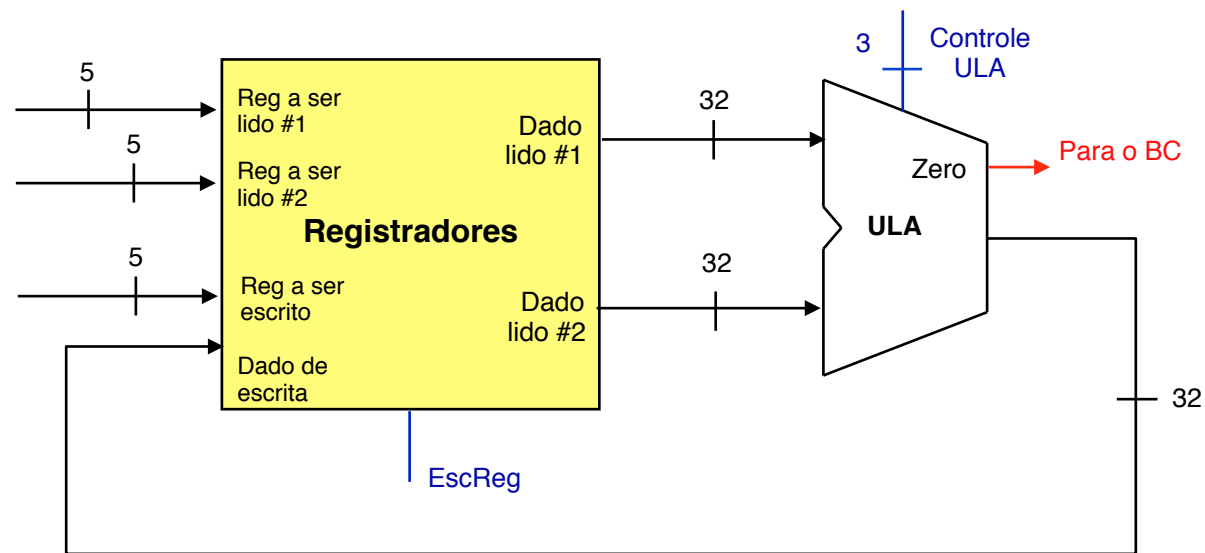


- 4 registradores
- 2 “portas” de leitura e 1 “porta” de escrita
- 2 bits de endereço/ por “porta”

# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ Registradores x Banco de Registradores

### Exemplo 2: Banco de registradores de um microprocessador

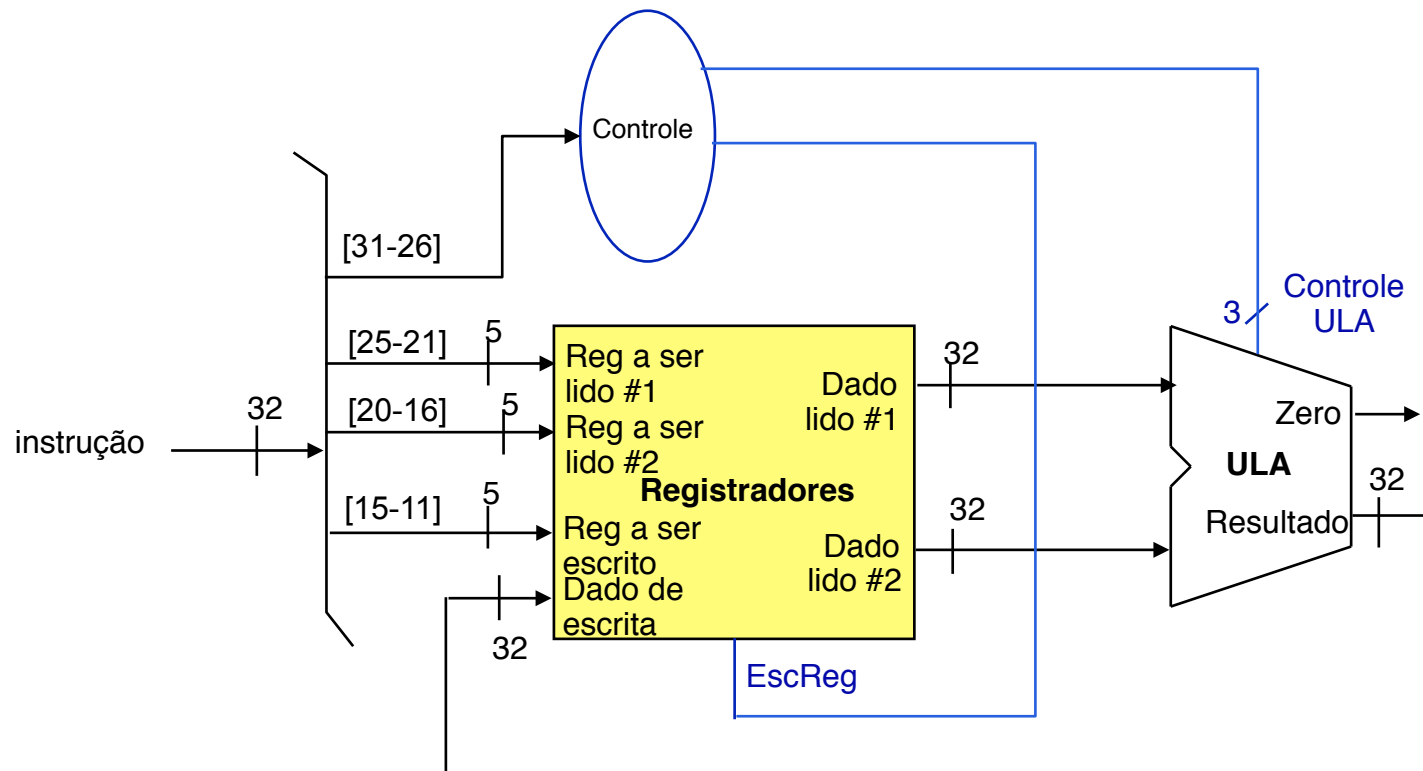


- 2 “portas” de leitura e 1 “porta” de escrita (há um sinal p/ habilitar escrita)
- Quantos registradores há neste banco de registradores?
- Qual o comprimento (ou tamanho) dos dados?

# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ Registradores x Banco de Registradores

### Exemplo 2: Banco de registradores de um microprocessador

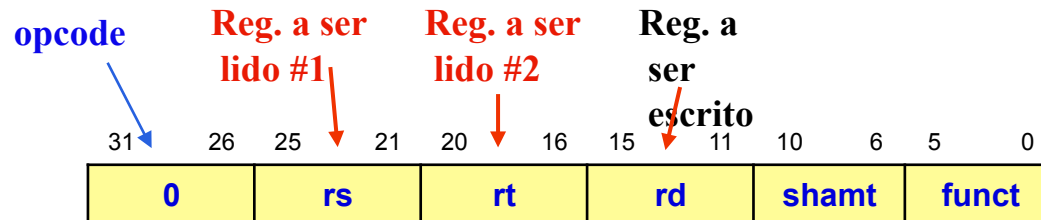


# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ Exemplos de Instruções (o Processador MIPS)

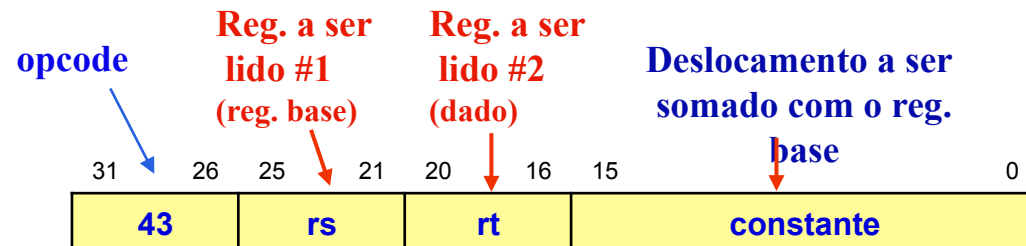
### Instrução tipo R:

```
add $s1,$s2, $s3  
( $\$s1 \leftarrow \$s2 + \$s3$ )
```



### Instrução store word :

```
sw $s1, constante($s2)  
( $\text{Mem}[\$s2 + \text{constante}] \leftarrow \$s1$ )
```



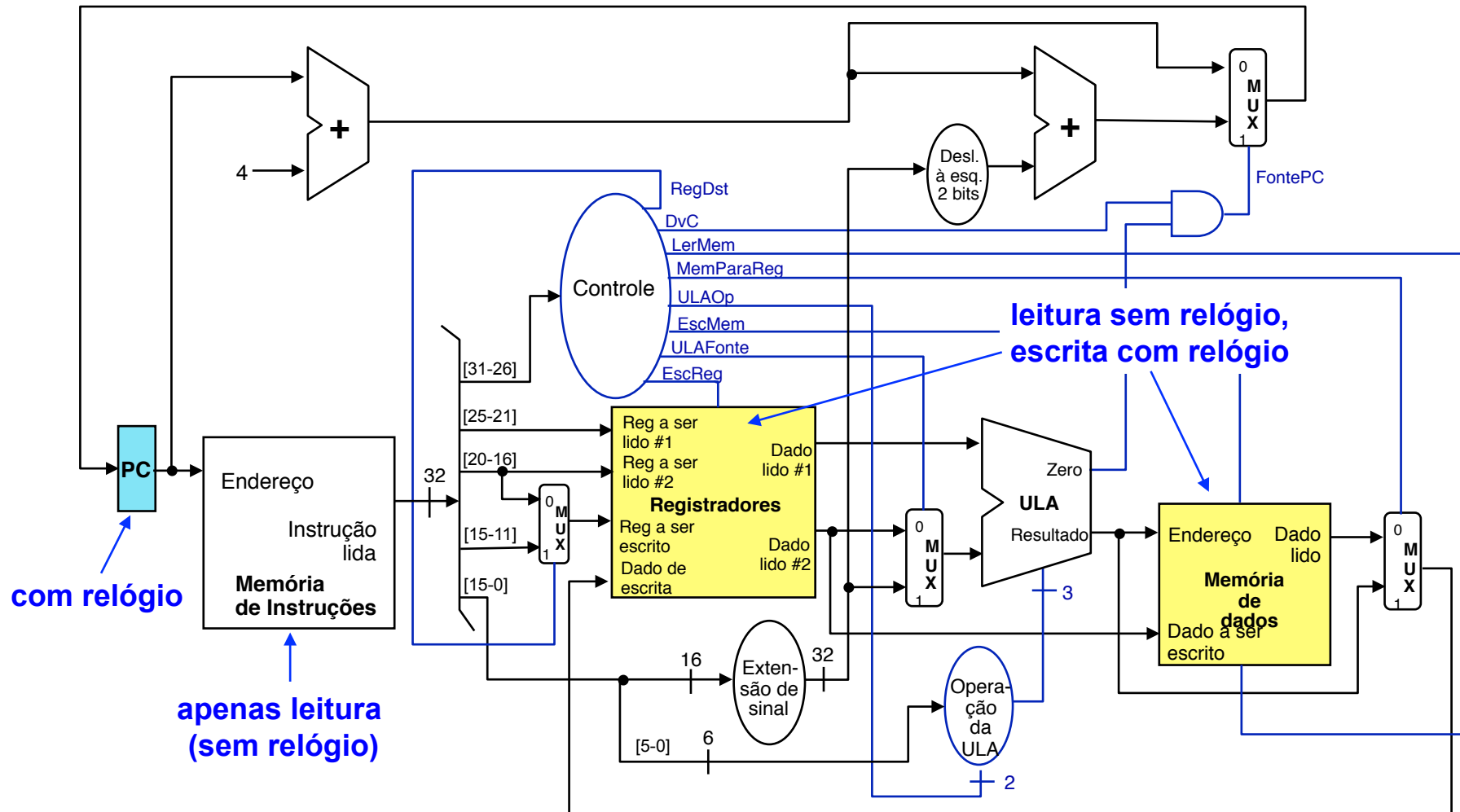
### Instrução load word:

```
lw $s1, constante($s2)  
( $\$s1 \leftarrow \text{Mem}[\$s2 + \text{constante}]$ )
```



# 4. Projeto de Sistemas Digitais no Nível RT

## ▶ O Processador MIPS: BO + BC



## 4. Projeto de Sistemas Digitais no Nível RT

---

### ▶ 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:

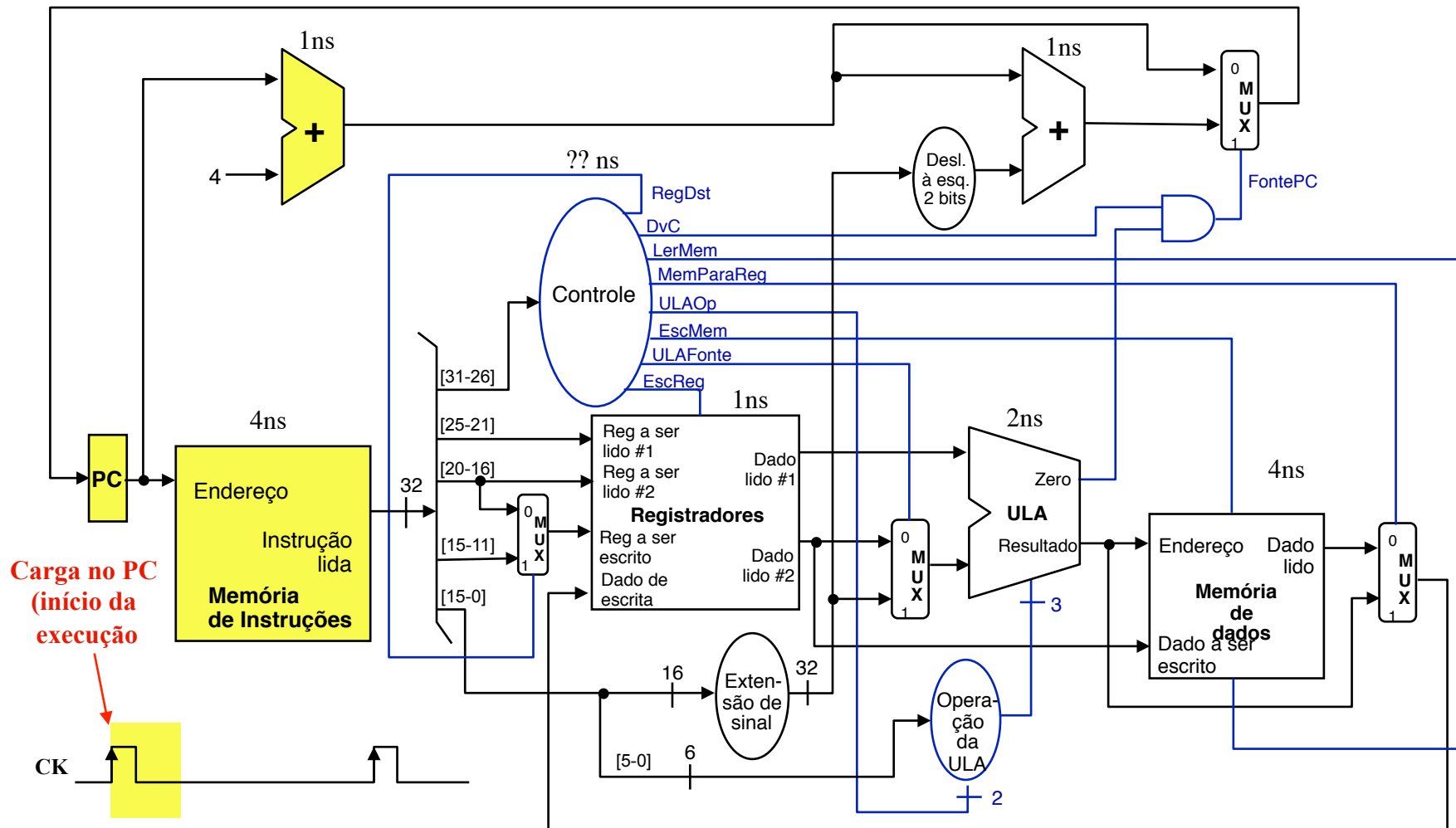
1. Busca da instrução (na memória de instruções) e incremento do PC
2. Leitura de dois registradores (no caso, \$t2 e \$t3, ou Rs e Rt) e geração dos sinais de controle para o resto do bloco operativo (decodificação da instrução)
3. Operação na ULA
4. Escrita (do resultado da operação realizada na ULA)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

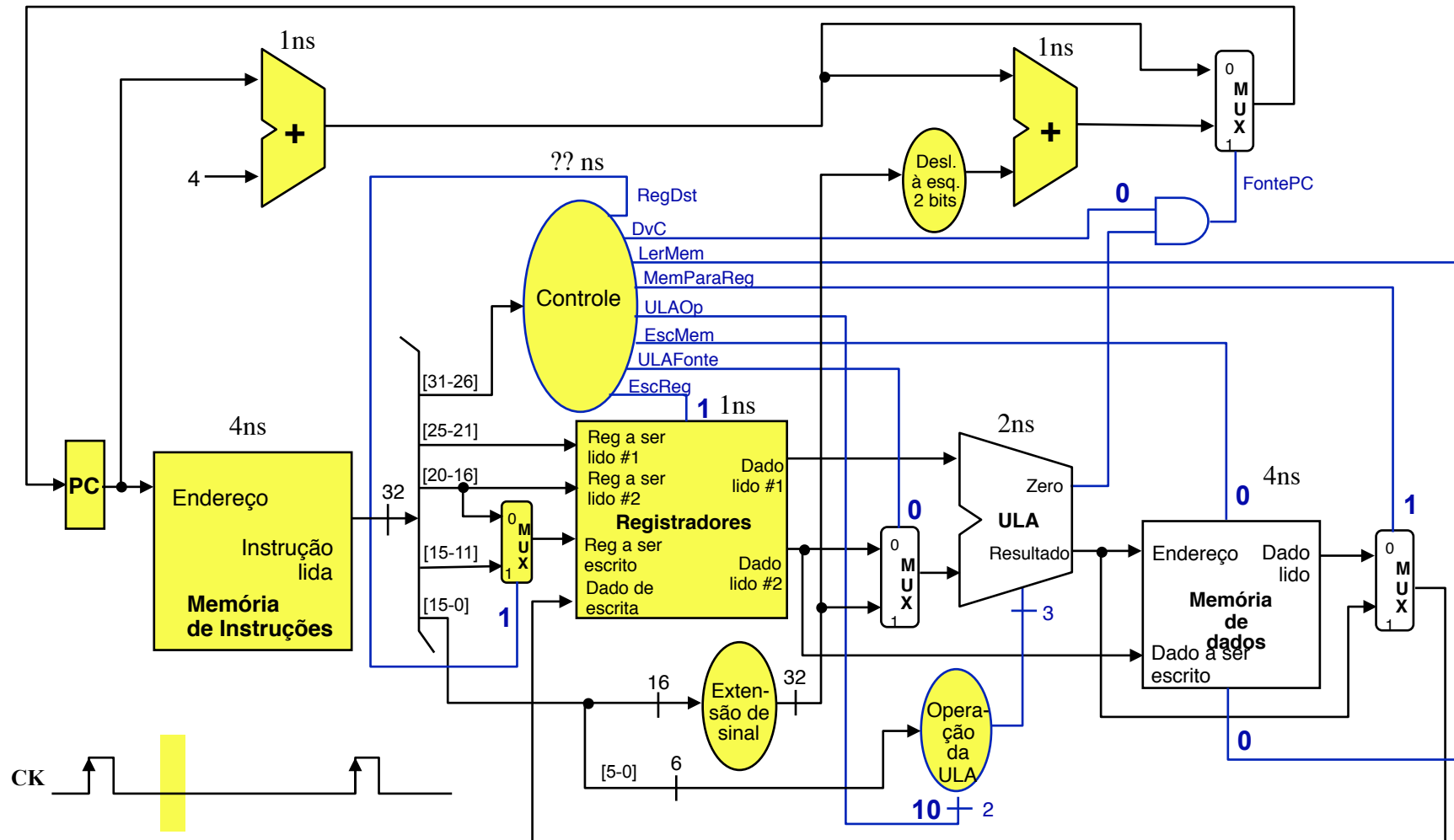
## ► Instrução Tipo R: busca da instrução e cálculo de PC+4



Carga no PC  
(início da  
execução)

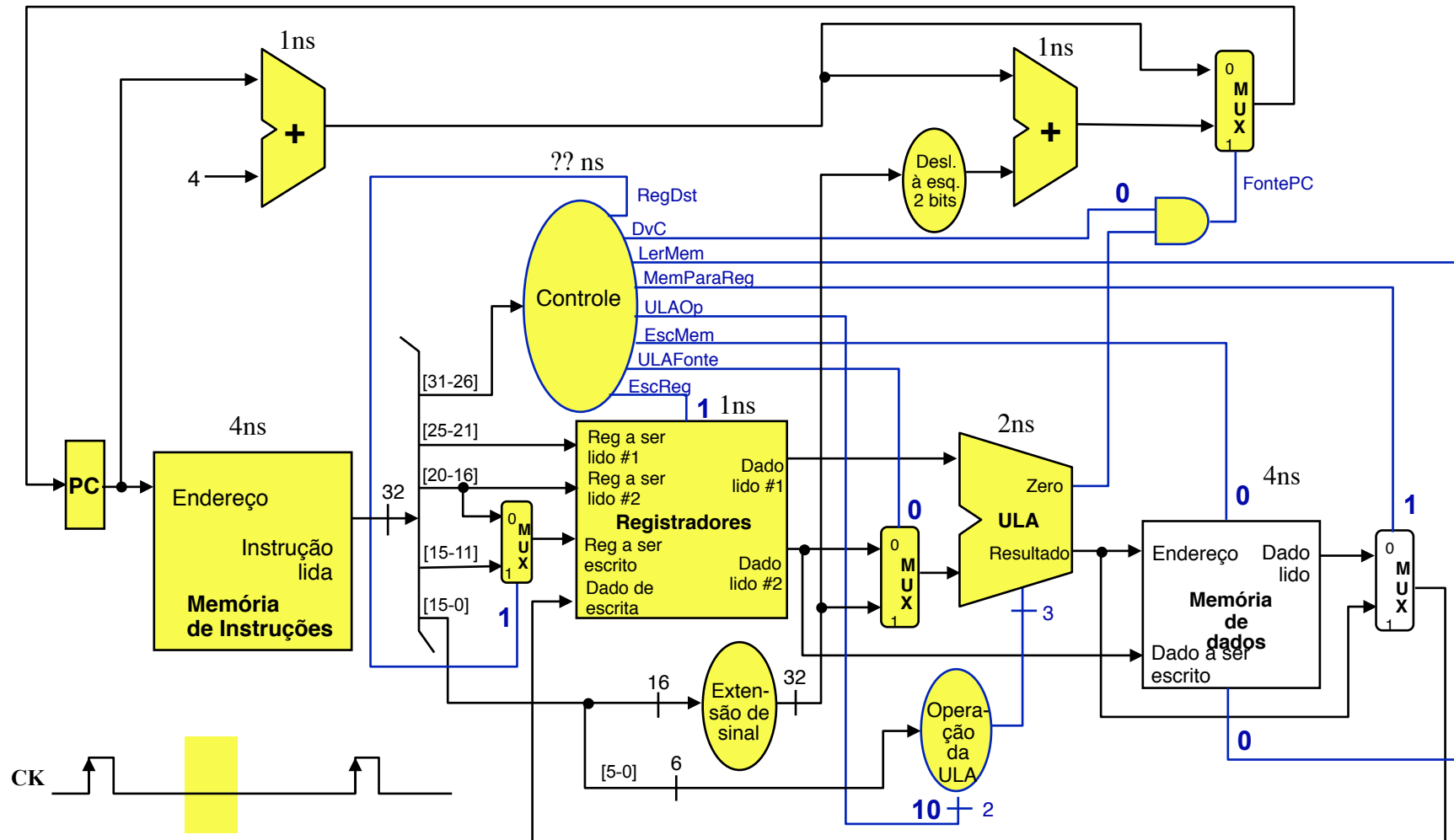
# 4. Projeto de Sistemas Digitais no Nível RT

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



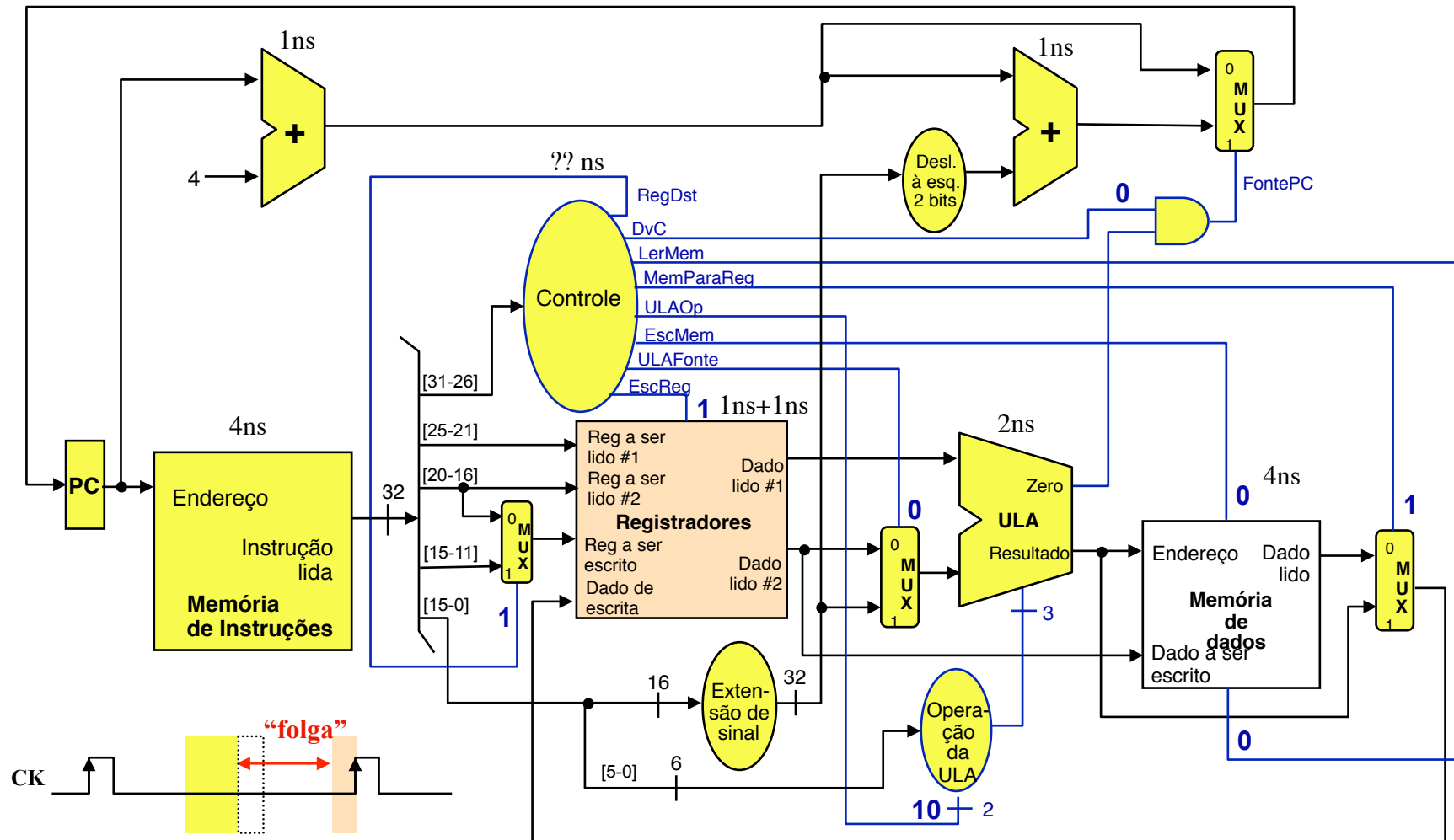
# 4. Projeto de Sistemas Digitais no Nível RT

## ► 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

