Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

Caminho de Dados e Controle (datapath and control)

Prof. Luciano L. Caimi
lcaimi@uffs.edu.br

Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Temporização

1. SDs Assíncronos:
   - ✓ Não possuem um sinal de relógio para prover o cadenciamento das operações
   - ✓ As operações são vistas como eventos encadeados (ou independentes)
   - ✓ Possui um custo maior em termos de recursos, pois é necessário implementar protocolos de comunicação entre os blocos do sistema
   - ✓ São tolerantes a variações na temporização
   - ✓ São mais difíceis de projetar (difícil de garantir o funcionamento, após a fabricação)

Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Temporização

2. SDs Síncronos:

- ✓ Utilizam um sinal de relógio para prover o cadenciamento das operações
- ✓ O funcionamento é quebrado em passos denominados operações
- ✓ Cada operação geralmente leva um ciclo de relógio para ser realizada, mas há esquemas alternativos (ex.: chaining, pipelining)
- ✓ Em um ciclo de relógio uma ou mais operações podem ser realizadas simultaneamente
- ✓ As técnicas de projeto existentes permitem abstrair-se detalhes do comportamento

Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Aplicação

1. Sistemas Digitais de Aplicação Específica (ASICs)

2. Sistemas Digitais de Propósito Geral

3. Sistemas Digitais Programáveis para uma Classe de Aplicações

Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Aplicação

1. Sistemas Digitais de Aplicação Específica (ASICs)
   - ✓ Realizam somente um algoritmo (ou parte de um algoritmo)
   - ✓ Oferecem pouca ou nenhuma programabilidade (i.e., alteração da funcionalidade)
   - ✓ O projeto é feito de modo a otimizar a execução do algoritmo implementado, visando a aplicação específica (máximo desempenho com o mínimo custo e eventualmente, mínimo consumo de energia)
   - ✓ Exemplos: codificadores/decodificadores de imagens estáticas (jpeg etc) ou dinâmicas (mpeg, H.264/AVC etc)

Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Aplicação

2. Sistemas Digitais de Propósito Geral:

- ✓ Podem ser programados para executar (virtualmente) qualquer algoritmo
- ✓ Para tanto, são projetados para realizar um conjunto de instruções
- ✓ São otimizados para realizar o conjunto de instruções para o qual são projetados (e não um algoritmo ou uma classe de algoritmo)
- ✓ Exemplos: microprocessadores (CPUs)

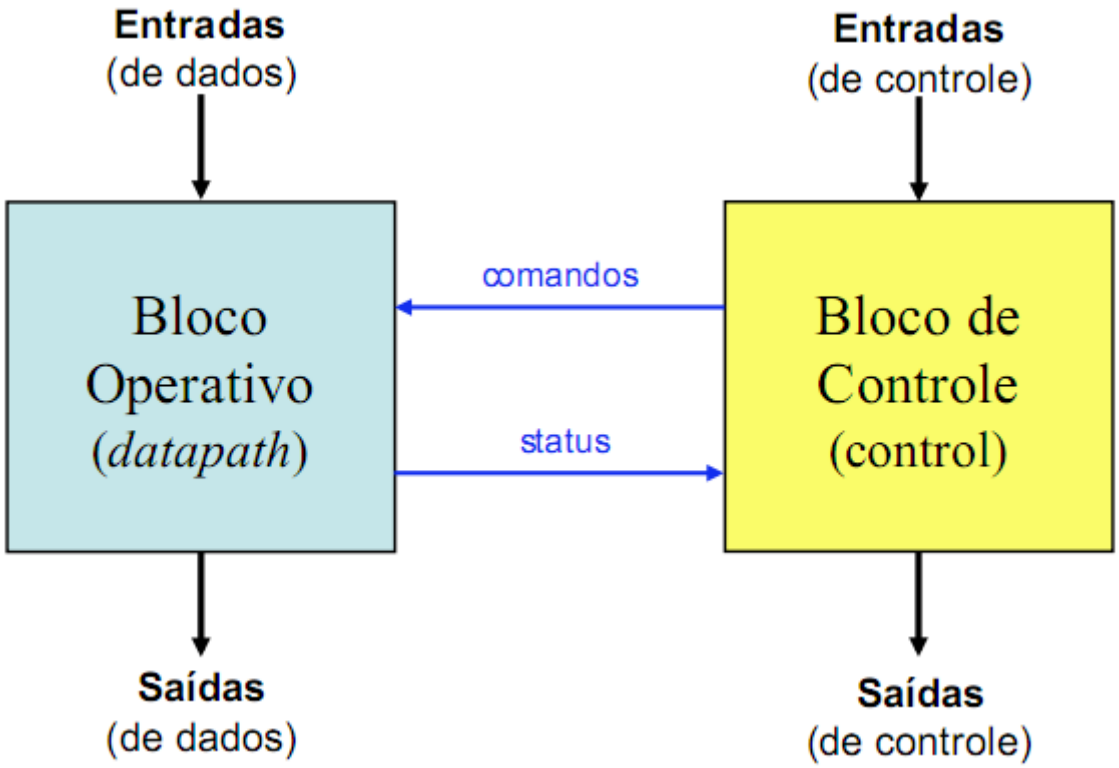
Tipos e Caract. dos Sistemas Digitais

Classificação Segundo a Aplicação

3. Sistemas Digitais Programáveis para uma Classe de Aplicações:
   - ✓ Podem ser programados para executar uma função ou um algoritmo pertencente a uma determinada classe.
   - ✓ São projetados e otimizados para realizar um conjunto de instruções apropriado à classe de problema a que se destinam
   - ✓ Exemplos: microcontroladores, DSPs (*Digital Signal Processors*) e GPUs (*Graphic Processing Units*)

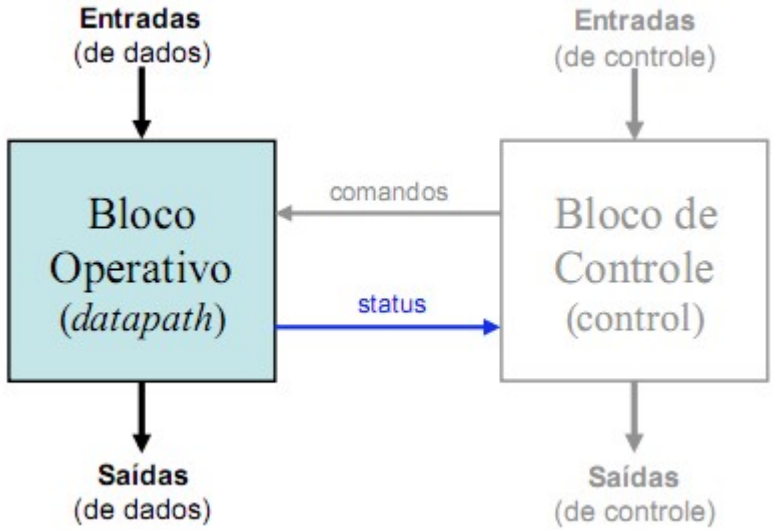
Projeto de Sistemas Digitais

Modelo Bloco Operativo / Controle



Projeto de Sistemas Digitais

Modelo Bloco Operativo / Controle

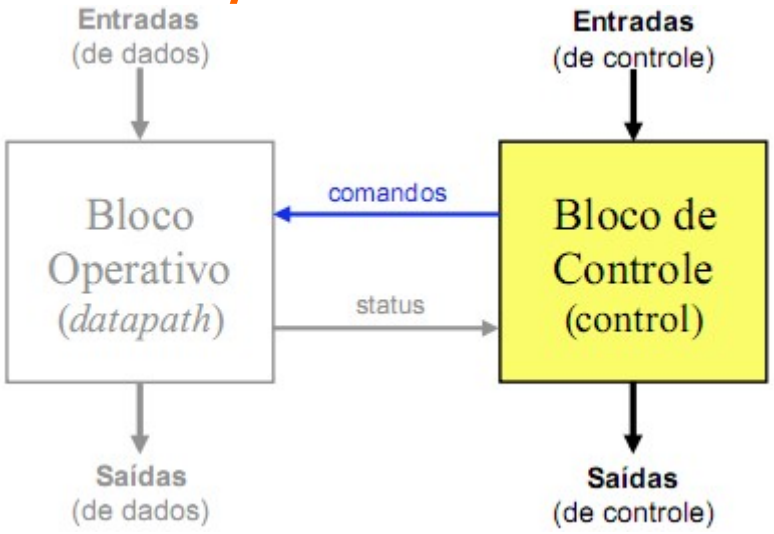


Bloco Operativo:

- ✓ Realiza transformações sobre dados, geralmente provenientes do ambiente externo
- ✓ As transformações são realizadas em um ou mais passos, cada passo demorando um ciclo de relógio
- ✓ Gera sinais de “status” que são usados pelo Bloco de Controle para definir a sequência de operações a serem realizadas (às vezes são chamados de “flags”)

Projeto de Sistemas Digitais

Modelo Bloco Operativo / Controle

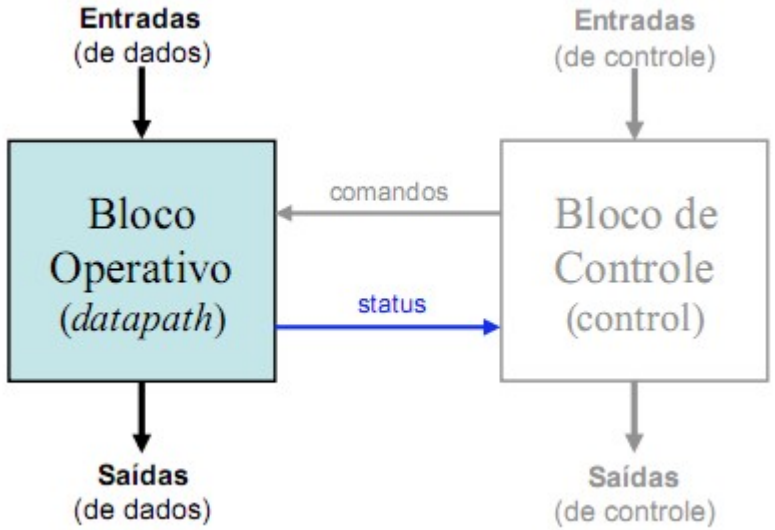


Bloco de Controle:

- ✓ Gera comandos, que são sinais de controle na ordem necessária p/ que o B.O. realize os passos desejados
- ✓ Recebe sinais de controle do ambiente externo, podendo ser desde um simples “iniciar” até um código de operação (“opcode”, dos processadores)
- ✓ Pode gerar uma ou mais saídas de controle para se comunicar com outros sistemas digitais (p. ex.: “done”, “bus request”, “ack”)

Projeto de Sistemas Digitais

Componentes no nível RT

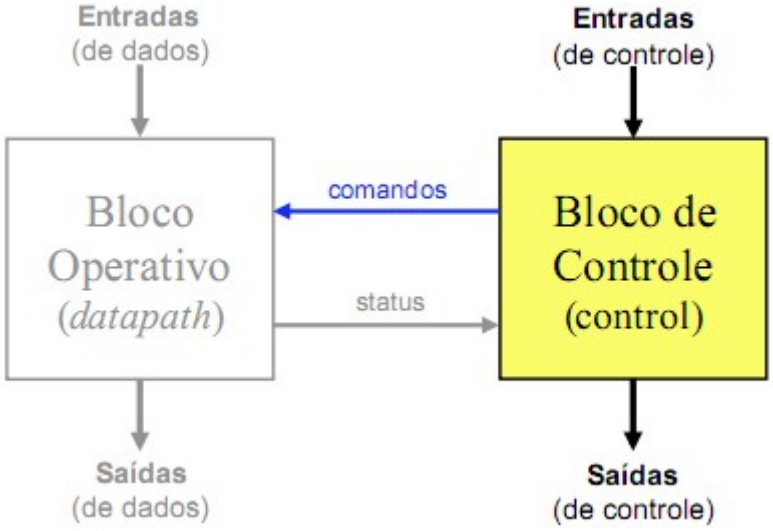


Bloco Operativo:

- ✓ Unidades Funcionais (UFs). Exemplos: somadores, subtratores, deslocadores, multiplicadores, UFs combinadas (somadores/subtratores, ULAs)
- ✓ Elementos de armazenamento: registradores, banco de registradores (vários registradores, mas com limitação de portas de entrada/saída), memórias RAM
- ✓ Rede de interconexão: fios, multiplexadores, barramentos + buffers tri-state

Projeto de Sistemas Digitais

Componentes no nível RT

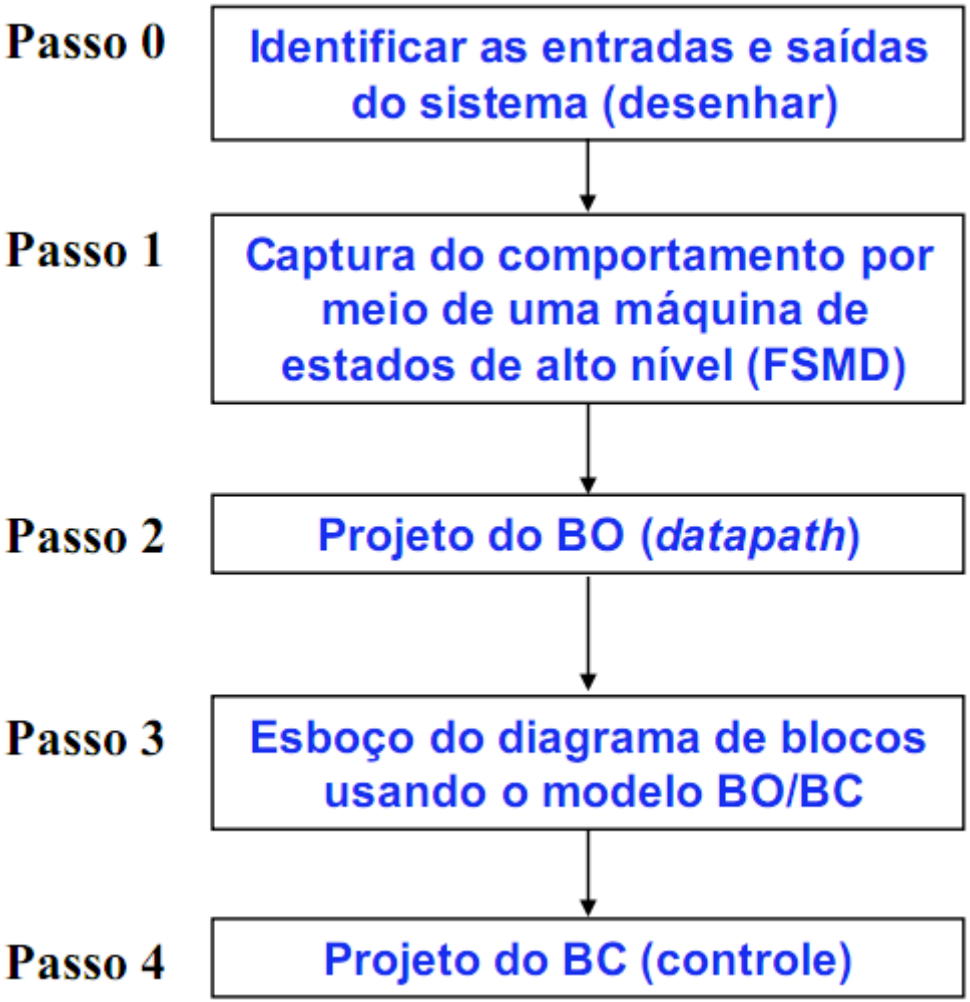


Bloco de Controle:

- Implementado por uma ou mais FSMs usando um dos seguintes métodos:
  - Hardwired: registrador de estados + circuitos lógicos ou
  - Usando ROM: registrador de estados + circuito comb. + um ou mais blocos ROM ou
  - Microprogramada: registrador-contador + circuito comb. + um ou mais blocos ROM (subcaso da anterior...)

Projeto de Sistemas Digitais

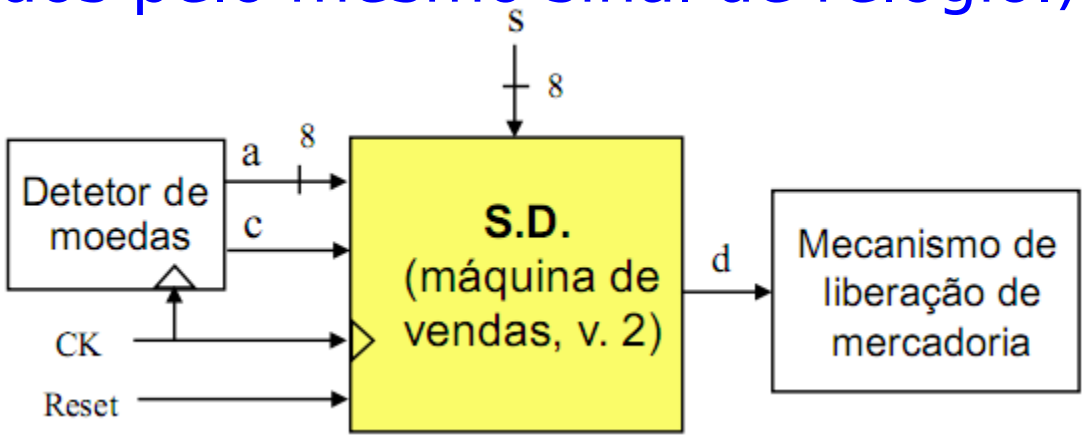
Modelo de Projeto



Projetando um Sistema Digital

Exemplo: máquina de vendas

Necessita-se de uma máquina de vendas capaz de gerenciar a venda de (apenas) um tipo de mercadoria, doravante denominada “item”. Esta máquina possui um detetor de moedas que provê um sinal de 1 bit chamado “c”, o qual vale “1” durante um ciclo de relógio quando uma moeda é detetada. (O controle da máquina de vendas e o seu detetor de moedas são sincronizados pelo mesmo sinal de relógio.)

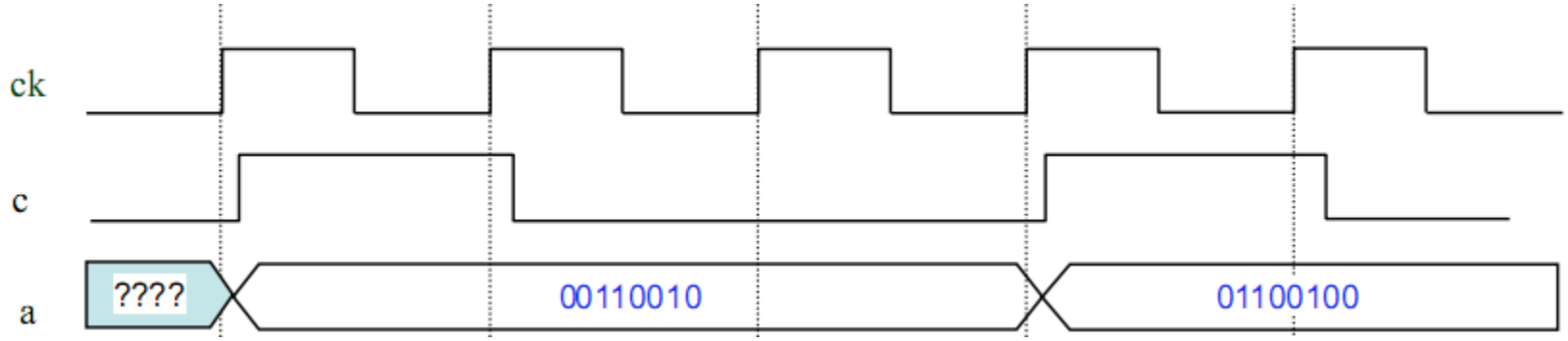


Projetando um Sistema Digital

Exemplo: máquina de vendas

O detector também possui um registrador de 8 bits no qual ele armazena o valor da moeda inserida, em centavos. Este registrador é carregado somente quando o sinal “c” sobe. A saída deste registrador é entrada pra máquina de vendas, sendo chamada “a”.

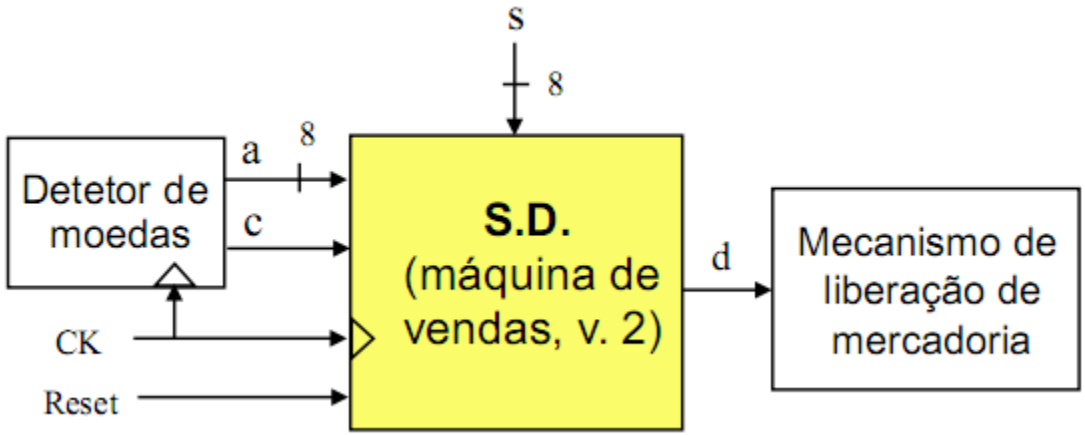
Exemplo de sincronismo dos sinais “c” e “a” (detecção de uma moeda de 50 centavos e depois, de uma moeda de 1 real, supondo um relógio lento...)



Projetando um Sistema Digital

Exemplo: máquina de vendas

A máquina possui ainda uma entrada de 8 bits denominada “s”, pela qual o proprietário pode definir o preço (unitário) da mercadoria. (Assuma que o valor correspondente ao preço permanece estável em “s” durante a operação normal.)

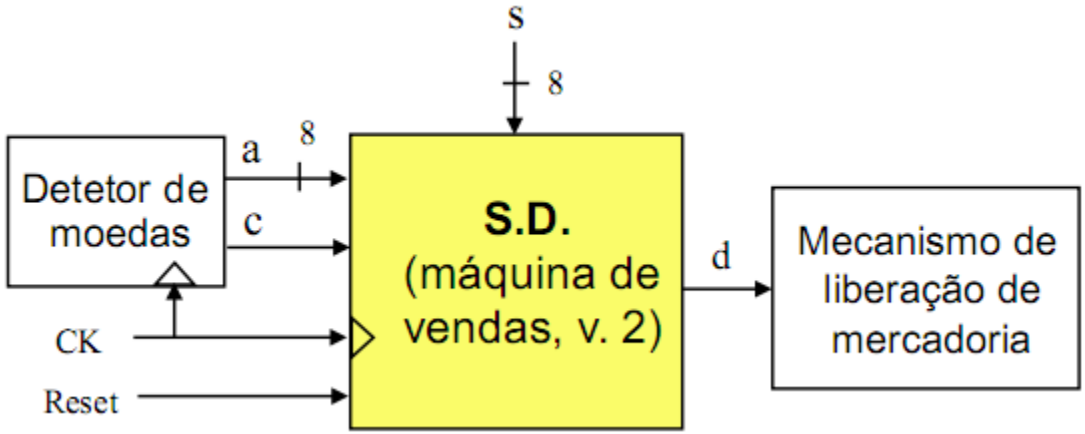


Projetando um Sistema Digital

Exemplo: máquina de vendas

Uma vez que o sistema digital identifica moedas cujo valor seja igual ou maior que o preço do item, ele "seta" o valor do sinal de saída "d" (que tem um bit) durante um ciclo de relógio, causando a liberação de um item.

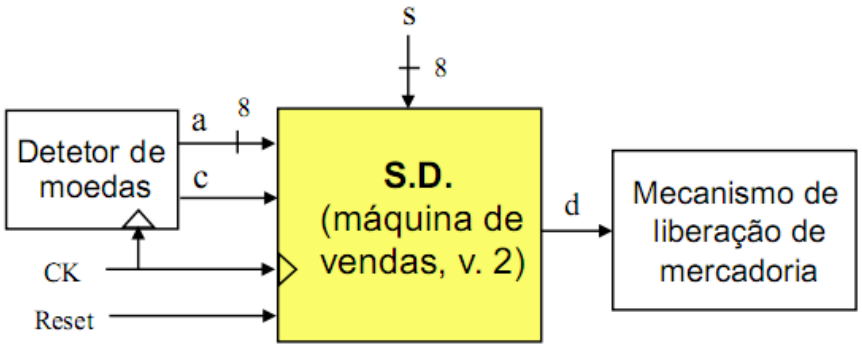
O Sistema não fornece troco.

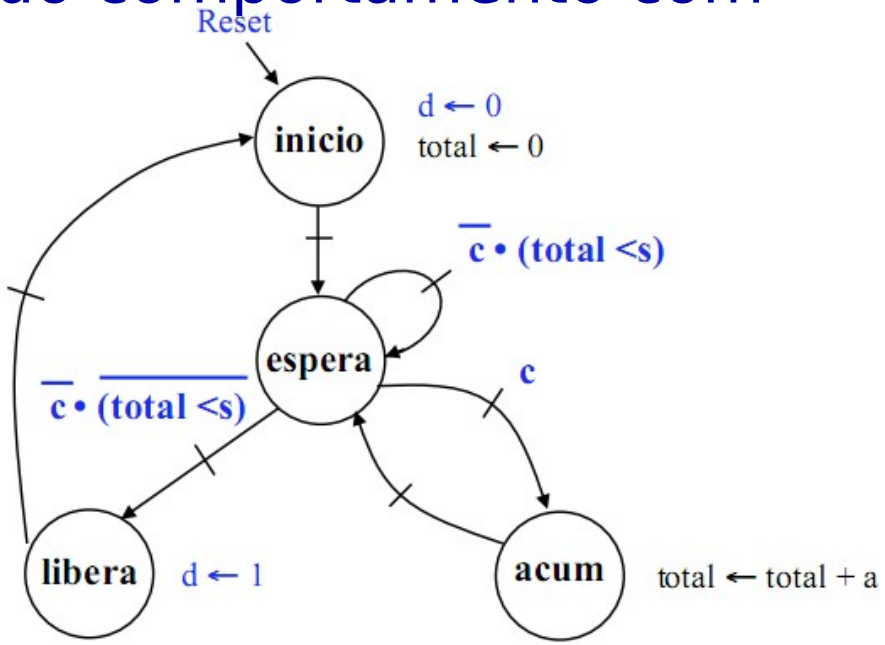


Projeto de Sistemas Digitais

Projetando um Sistema Digital

Exemplo 1: Passo 1 (captura do comportamento com FSMD)





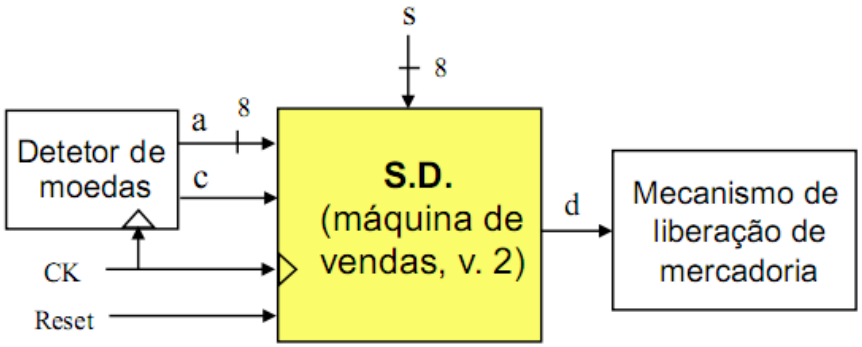
Pergunta:
Em termos práticos, existe alguma limitação quanto à duração do período do relógio?

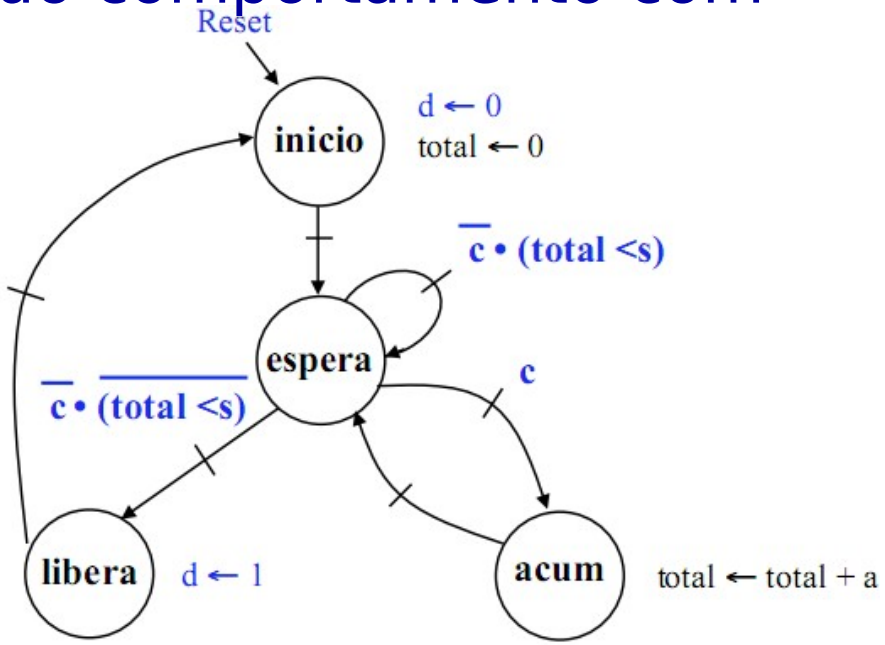
OBS:

- Condições para troca de estados em azul (junto às arestas), atribuições de controle também em azul (junto aos estados)
- Atribuições e operações com dados em preto, junto aos respectivos estados

Projetando um Sistema Digital

Exemplo 1: Passo 1 (captura do comportamento com FSMD)





Entradas: c (1 bit), a (8 bits), s (8 bits)

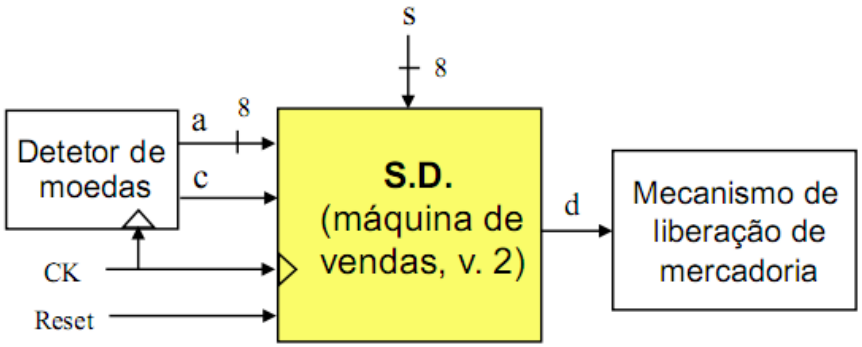
Saídas: d (1 bit)

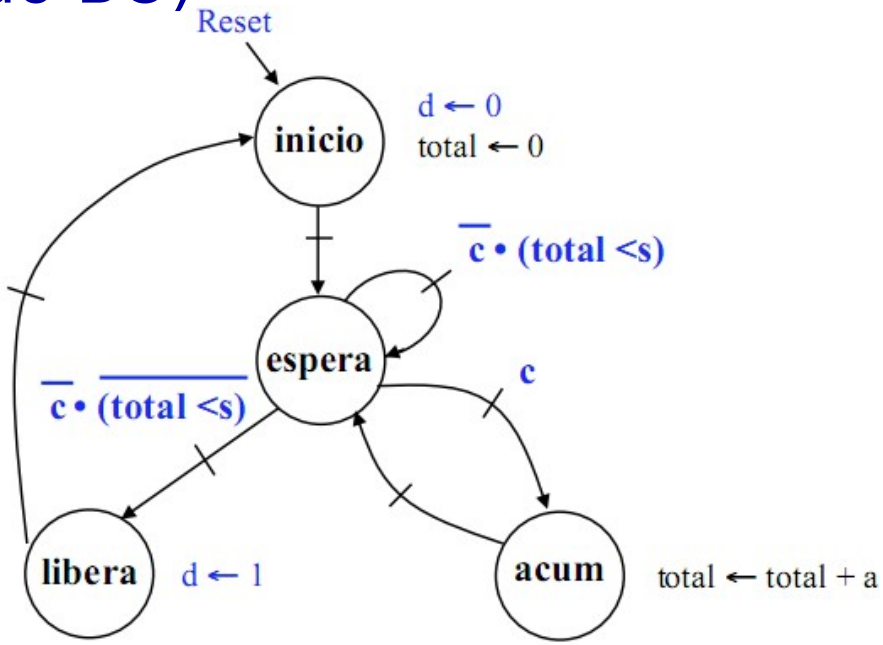
Variáveis internas ou locais: total (8 bits)

OBS: em primeira aproximação, pode-se assumir que cada variável interna é mapeada para um registrador.

Projetando um Sistema Digital

Exemplo 1: Passo 2 (projeto do BO)





Questões:

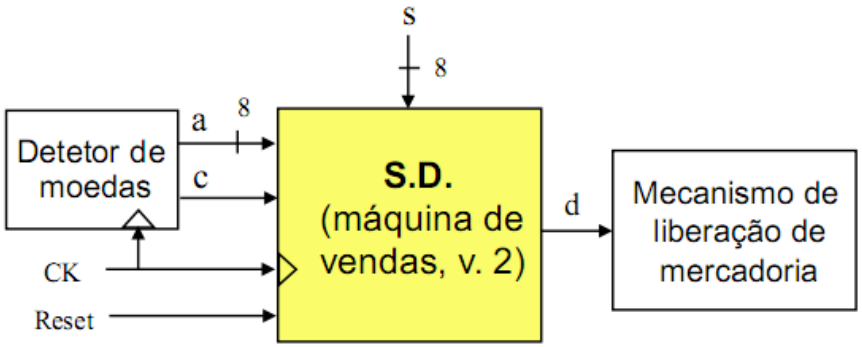
Quais variáveis são usadas para armazenar dados?

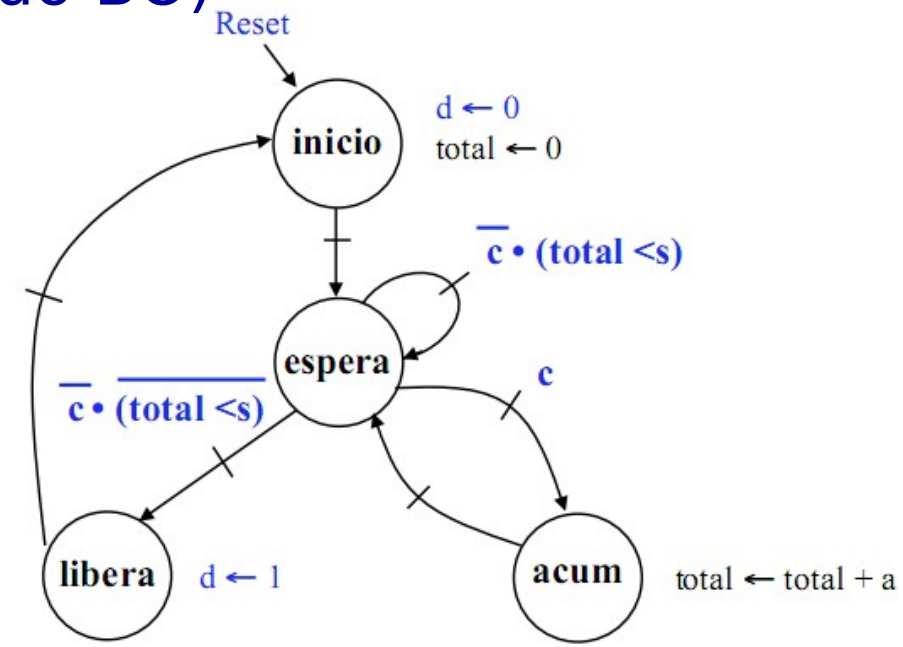
Apenas uma variável: “total” (note que “a” é uma entrada)

Logo, teremos um registrador denominado “total” para esta variável

Projetando um Sistema Digital

Exemplo 1: Passo 2 (projeto do BO)





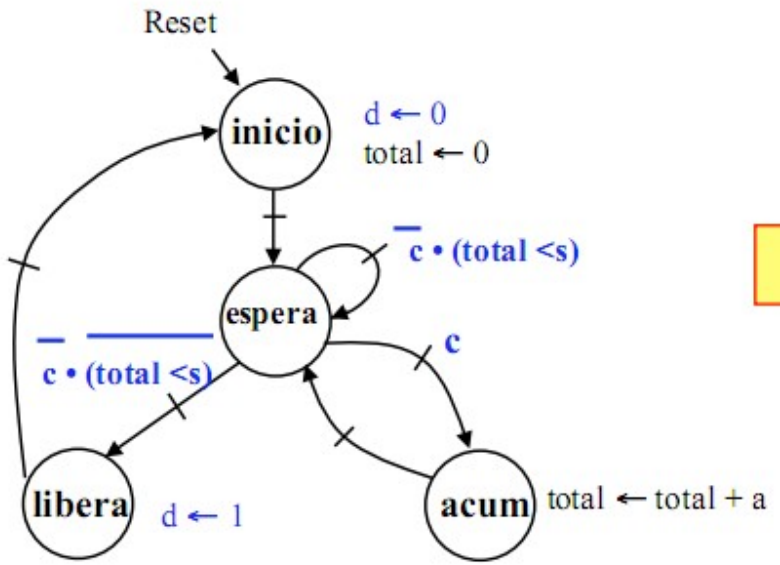
Questões:

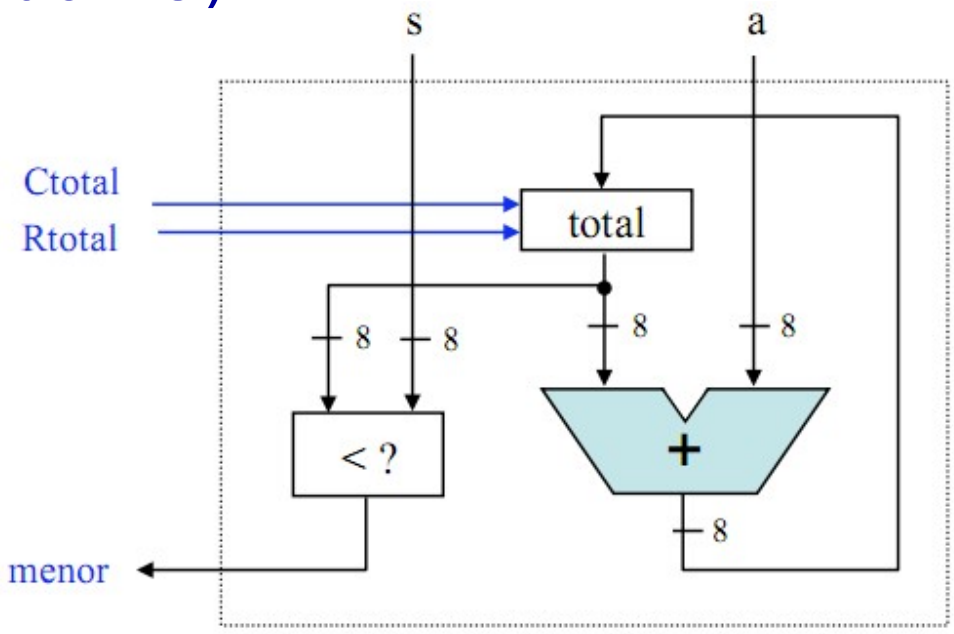
Quais operações são realizadas sobre dados (incluindo-se as condições)?

- Uma adição para números de 8 bits (total ← total + a)
- Uma comparação entre “total” e “s”, ambos com 8 bits.

Projetando um Sistema Digital

Exemplo 1: Passo 2 (projeto do BO)





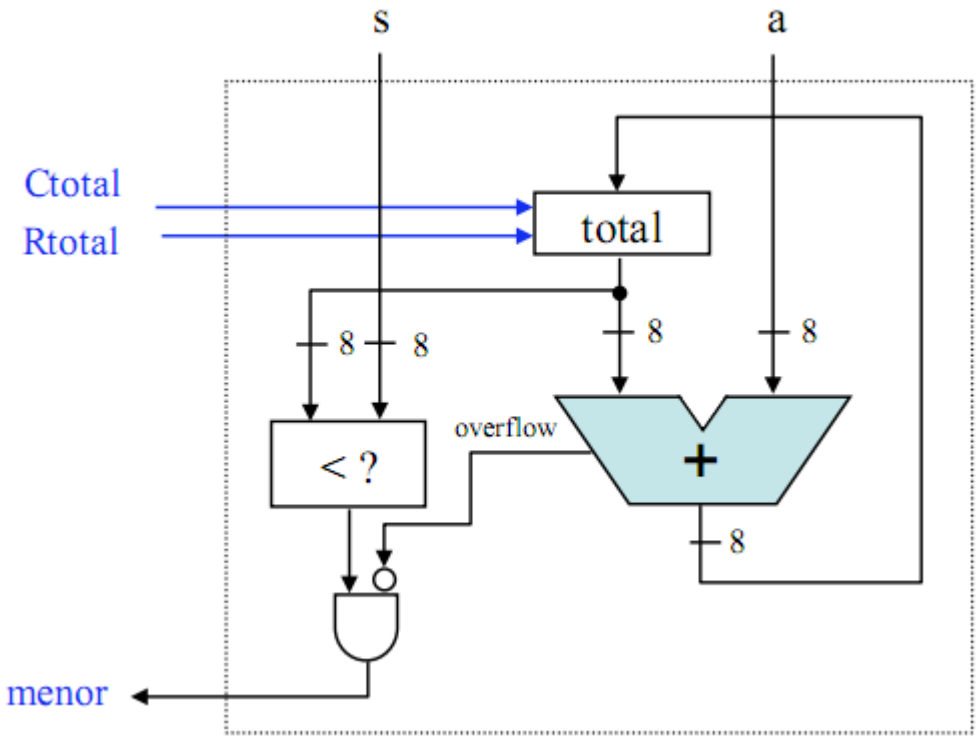
Convenção:

Todos os registradores são cadenciados pelo sinal de relógio (ck).

Entretanto, para simplificar o desenho, o sinal de relógio está omitido.

Projetando um Sistema Digital

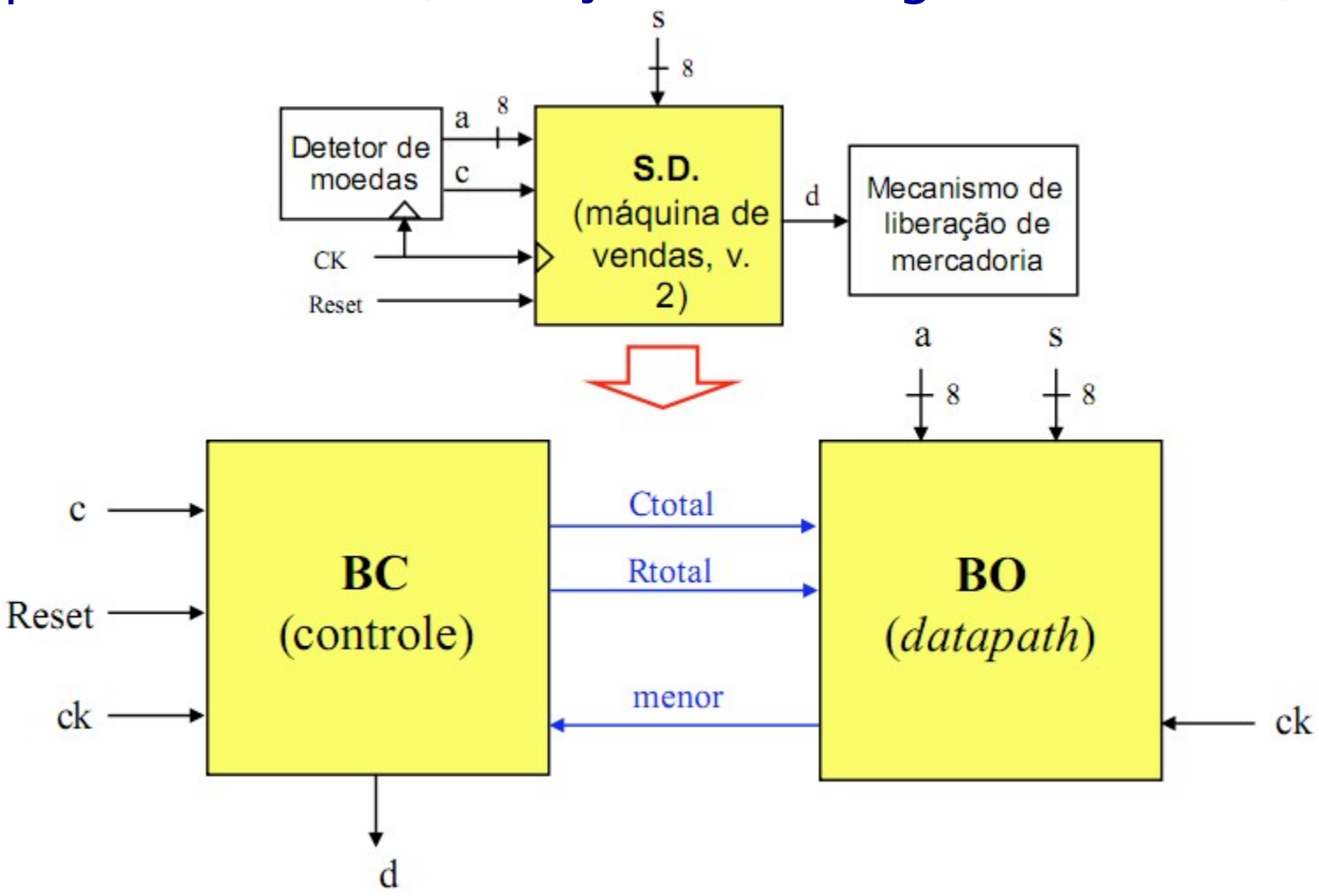
Exemplo 1: Passo 2 (projeto do BO)



Prevendo a possibilidade de ocorrência de overflow.

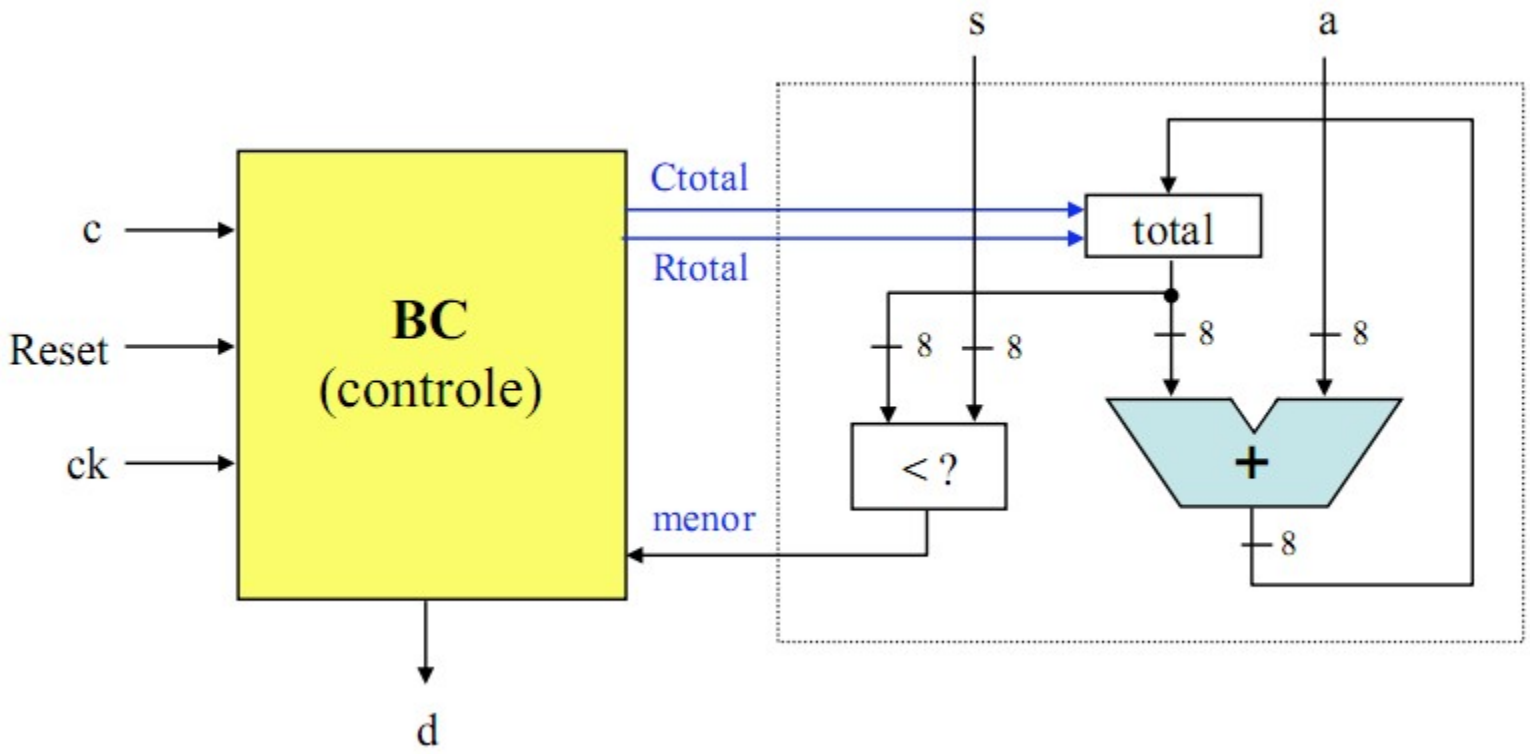
Projetando um Sistema Digital

Exemplo 1: Passo 3 (Esboçando o diagrama BO/BC)



Projetando um Sistema Digital

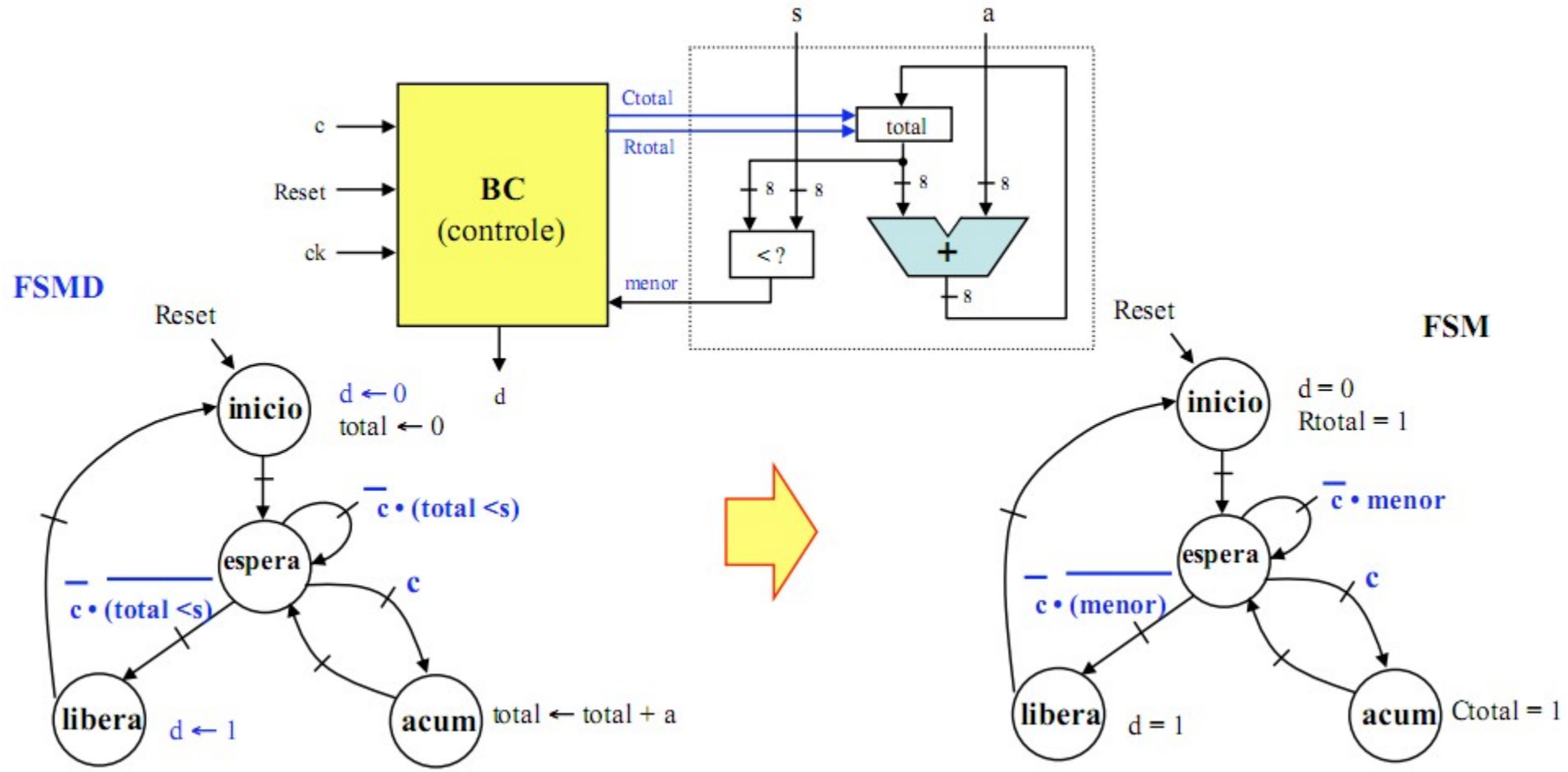
Exemplo 1: Passo 3 (diagrama BO/BC mais detalhado...)



Projeto de Sistemas Digitais

Projetando um Sistema Digital

Exemplo 1: Passo 4 (Derivando a FSM a partir do BO e da FSMD)



Projetando um Sistema Digital

Exemplo 1: Passo 4 (Projeto do BC)

FSM

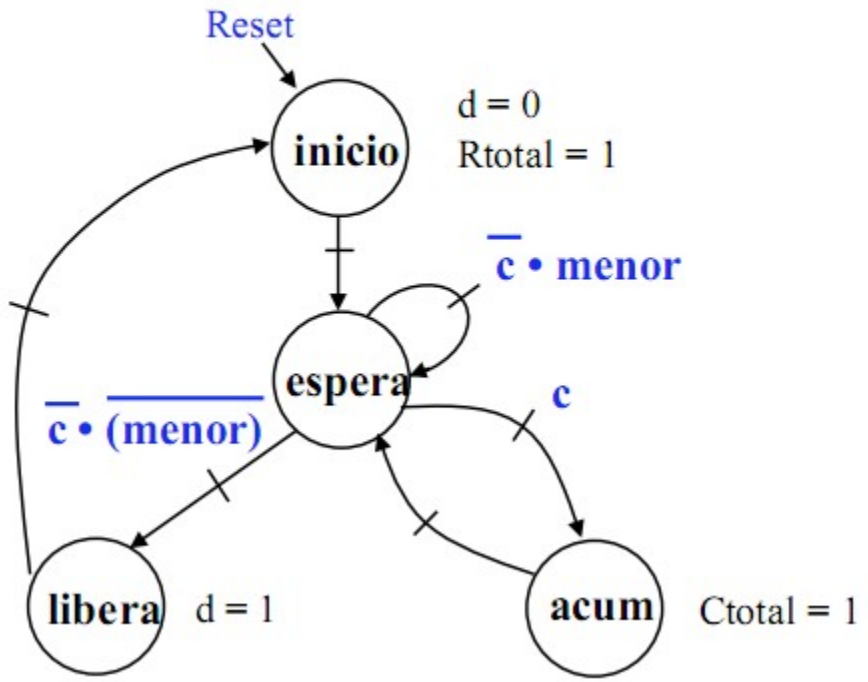


Tabela de Transição de Estados

Estado atual	c	menor	Próximo estado
início	X	X	espera
espera	0	0	libera
espera	0	1	espera
espera	1	X	acum
acum	X	X	espera
libera	X	X	início

Projetando um Sistema Digital

Exemplo 1: Passo 4 (Projeto do BC)

FSM

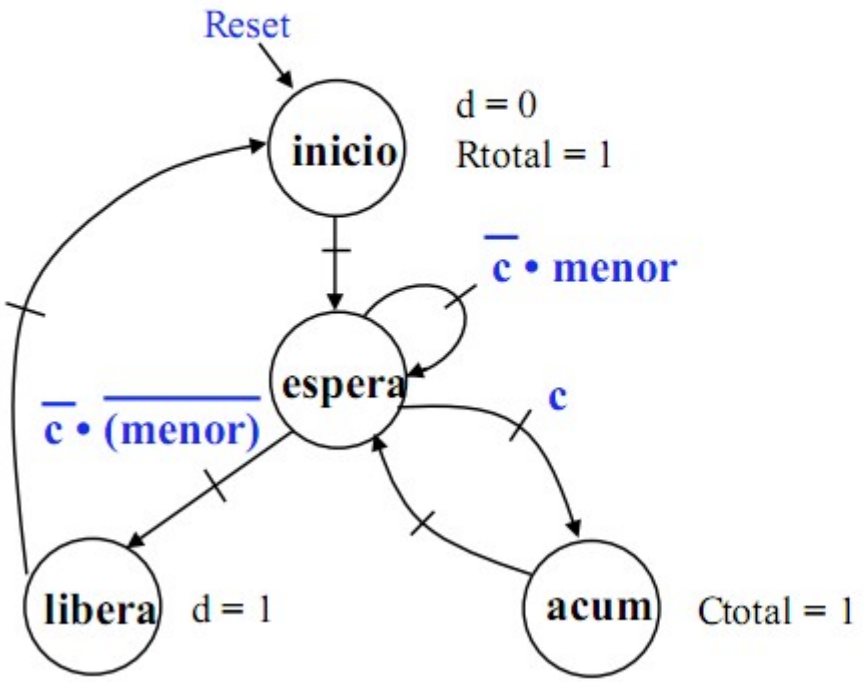
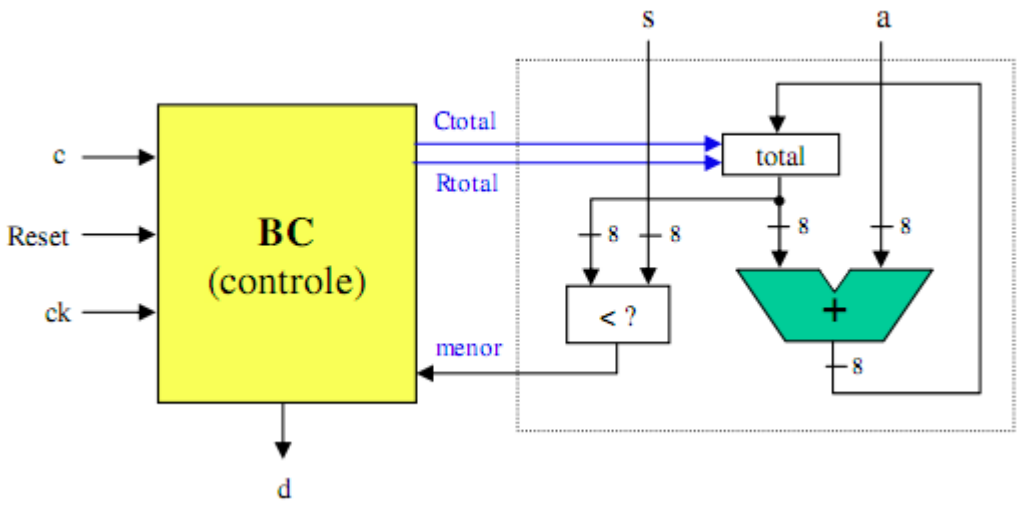


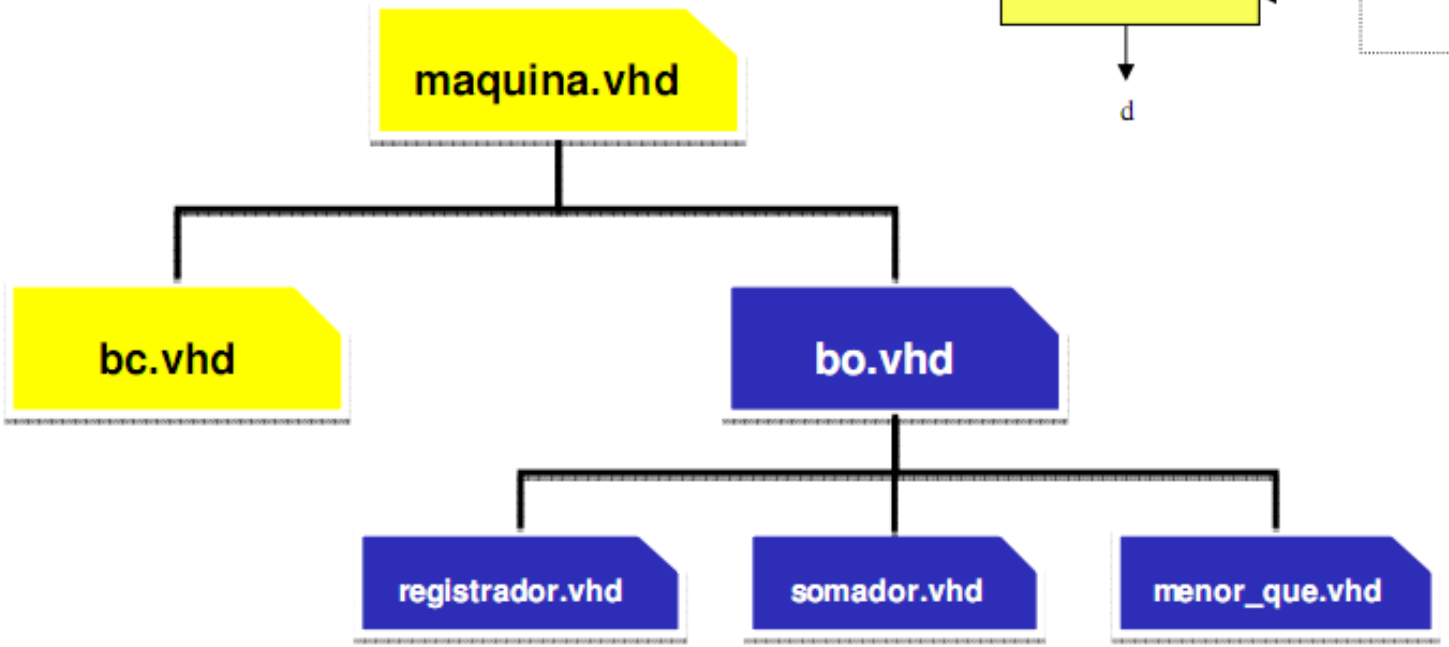
Tabela de Saídas

Estado	Rtotal	Ctotal	d
início	1	0	0
espera	0	0	0
acum	0	1	0
libera	0	0	1

Projeto de Sistemas Digitais

Hierarquia de arquivos VHDL

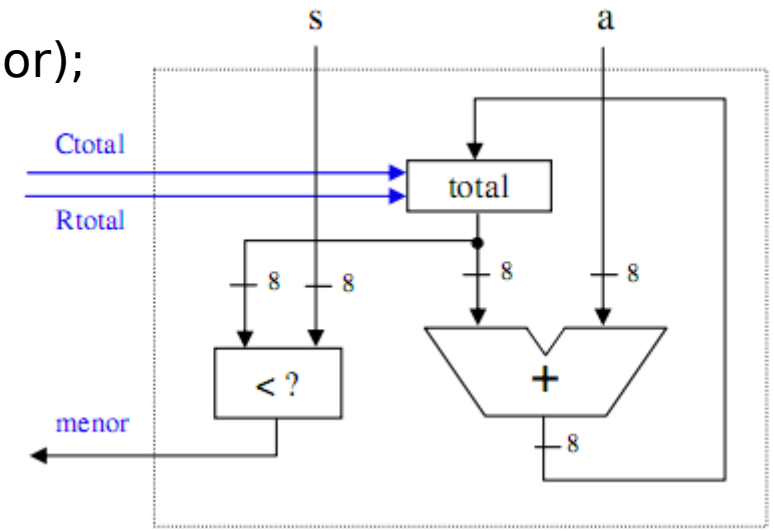




Projeto de Sistemas Digitais

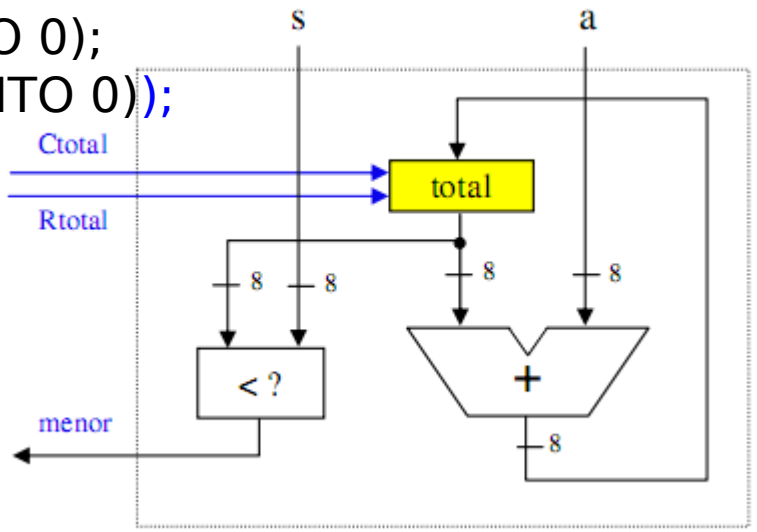
Bloco Operacional

```
ENTITY bo IS
  PORT (clk, Ctotal, Rtotal : IN STD_LOGIC;
        s, a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        menor : OUT STD_LOGIC);
END bo;
ARCHITECTURE estrutura OF bo IS
   -- components
   SIGNAL soma, total: STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
      reg: registrador PORT MAP (clk, Ctotal, Rtotal, soma, total);
      som: somador PORT MAP (total, a, soma);
      men: menor_que PORT MAP (total, s, menor);
END estrutura;
```



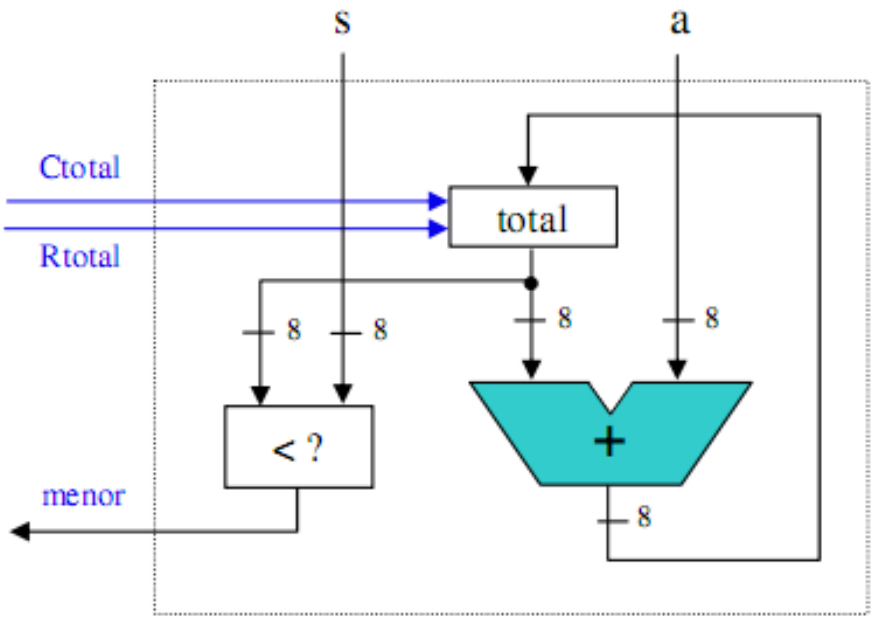
Registrador

```
ENTITY registrador IS
    PORT (clk, carga, reset : IN STD_LOGIC;
          d : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
          q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END registrador;
ARCHITECTURE estrutura OF registrador IS
BEGIN
   PROCESS(clk, reset)
   BEGIN
      IF(reset = '1') THEN
         q <= "00000000";
      ELSIF(clk'EVENT AND clk = '1' AND carga = '1') THEN
         q <= d;
     END IF;
   END PROCESS;
END estrutura;
```



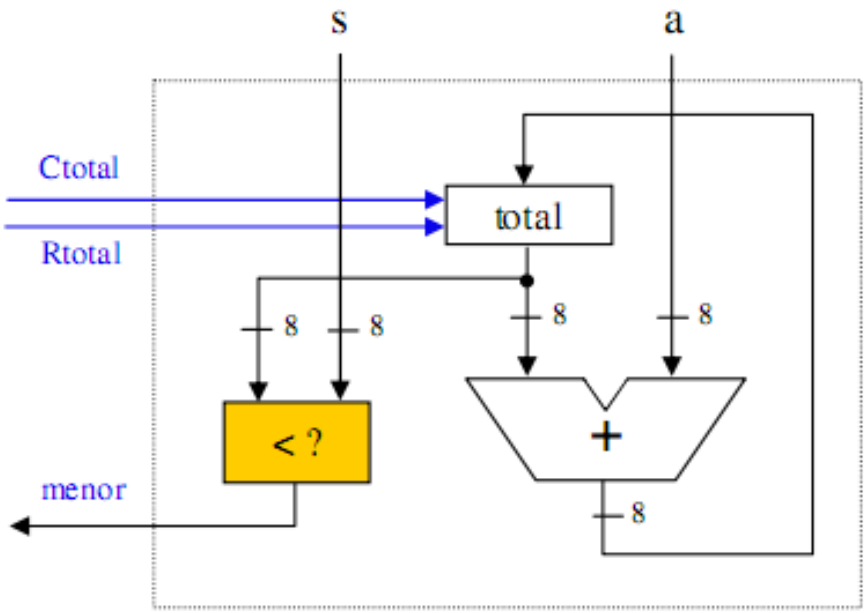
Somador

```
ENTITY somador IS
PORT ( a, b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
       s : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END somador;

ARCHITECTURE estrutura OF somador IS
BEGIN
   s <= a + b;
END estrutura;
```



Comparador

```
ENTITY menor_que IS
    PORT (a, b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
          menor : OUT STD_LOGIC);
END menor_que;

ARCHITECTURE estrutura OF menor_que IS
BEGIN
   menor <= '1' WHEN A < B ELSE '0';
END estrutura;
```



Projeto de Sistemas Digitais

Bloco de Controle

```
ARCHITECTURE estrutura OF bc IS
   TYPE state_type IS (INICIO, ESPERA, ACUM, LIBERA);
   SIGNAL state: state_type;
BEGIN
   -- Logica de proximo estado (e registrador de estado)

   PROCESS (clk, Reset)
   BEGIN
         -- completar
   END PROCESS;

   -- Logica de saida
   PROCESS (state)
   BEGIN
         -- completar
   END PROCESS;
END estrutura;
```

Estado atual	c	menor	Próximo estado
início	X	X	espera
espera	0	0	libera
espera	0	1	espera
espera	1	X	acum
acum	X	X	espera
libera	X	X	início

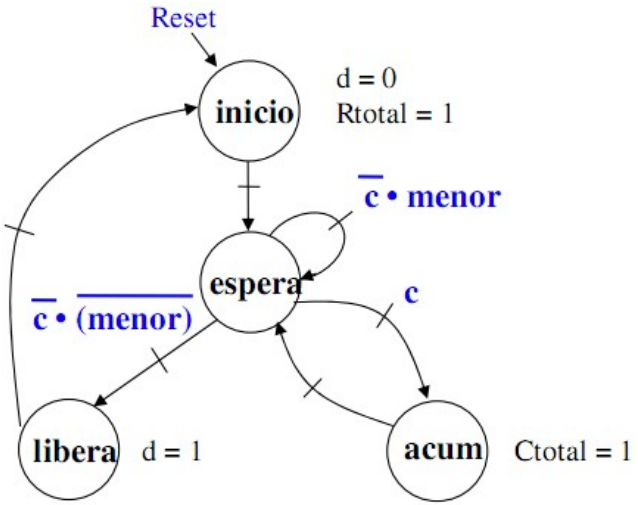
Estado	Rtotal	Ctotal	d
início	1	0	0
espera	0	0	0
acum	0	1	0
libera	0	0	1

Projeto de Sistemas Digitais

Bloco de Controle

```
PROCESS (clk, Reset)
BEGIN
      if(Reset = '1') THEN  state <= INICIO;
      ELSIF (clk'EVENT AND clk = '1') THEN
        CASE state IS
              WHEN INICIO =>
                    state <= ESPERA;
              WHEN ESPERA =>
                    IF (c = '1') THEN
                          state <= ACUM;
                    ELSIF (menor = '0') THEN
                          state <= LIBERA;
                    ELSE
                          state <= ESPERA;
                    END IF;
              WHEN ACUM =>
                    state <= ESPERA;
              WHEN LIBERA =>
                    state <= INICIO;
      END CASE;
END IF;
```

Estado atual	c	menor	Próximo estado
início	X	X	espera
espera	0	0	libera
espera	0	1	espera
espera	1	X	acum
acum	X	X	espera
libera	X	X	início

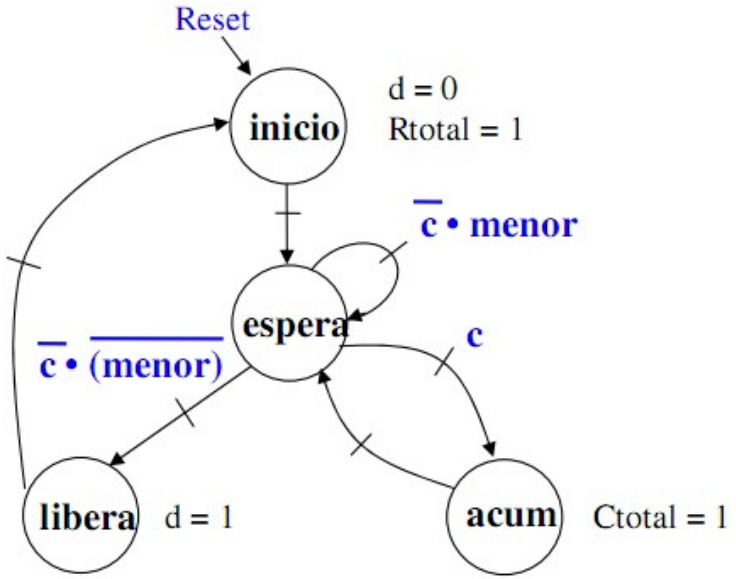


Projeto de Sistemas Digitais

Bloco de Controle

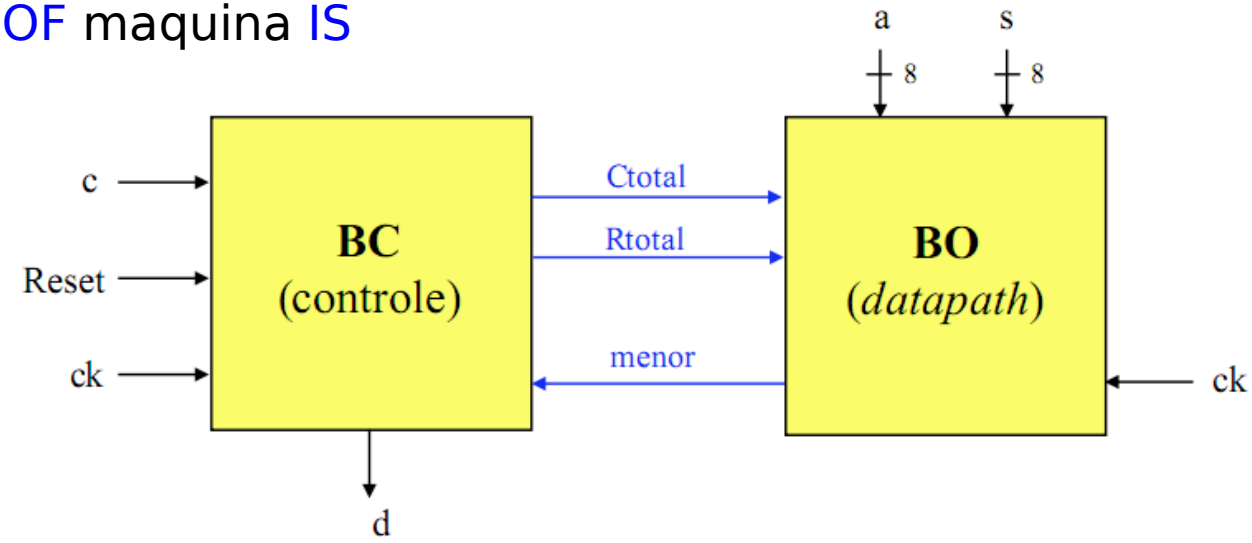
```
PROCESS (state)
BEGIN
      CASE state IS
            WHEN INICIO =>
                  d <= '0';
                  Rtotal <= '1';
                  Ctotal <= '0';
            WHEN ESPERA =>
                  d <= '0';
                  Rtotal <= '0';
                  Ctotal <= '0';
            WHEN ACUM =>
                  d <= '0';
                  Rtotal <= '0';
                  Ctotal <= '1';
            WHEN LIBERA =>
                  d <= '1';
                  Rtotal <= '0';
                  Ctotal <= '0';
      END CASE;
END PROCESS;
```

Estado	Rtotal	Ctotal	d
início	1	0	0
espera	0	0	0
acum	0	1	0
libera	0	0	1



Bloco de Controle

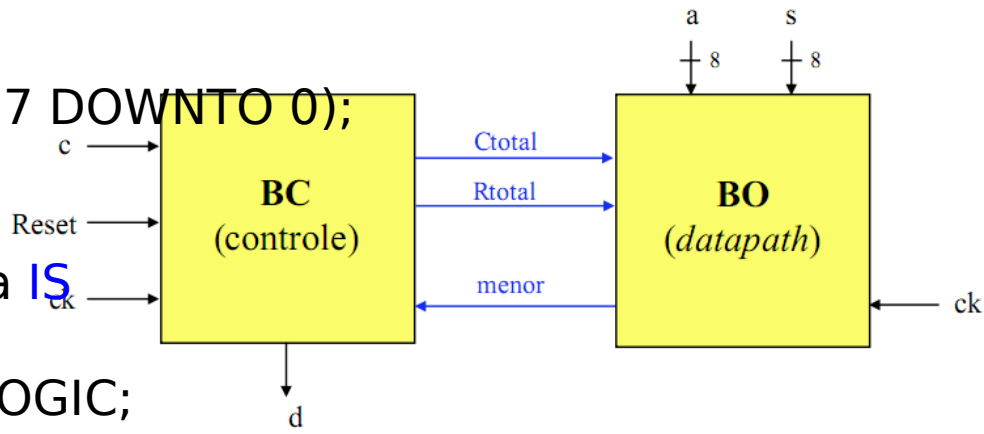
```
ENTITY maquina IS
   PORT (Reset, ck, c : IN STD_LOGIC;
         s, a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         d : OUT STD_LOGIC);
END maquina;
ARCHITECTURE estrutura OF maquina IS

      COMPONENT bo IS
         -- completar
      END COMPONENT;

      COMPONENT bc IS
         -- completar
      END COMPONENT;

     SIGNAL -- completar
BEGIN
      -- completar
END estrutura;
```



Projeto de Sistemas Digitais

Bloco de Controle

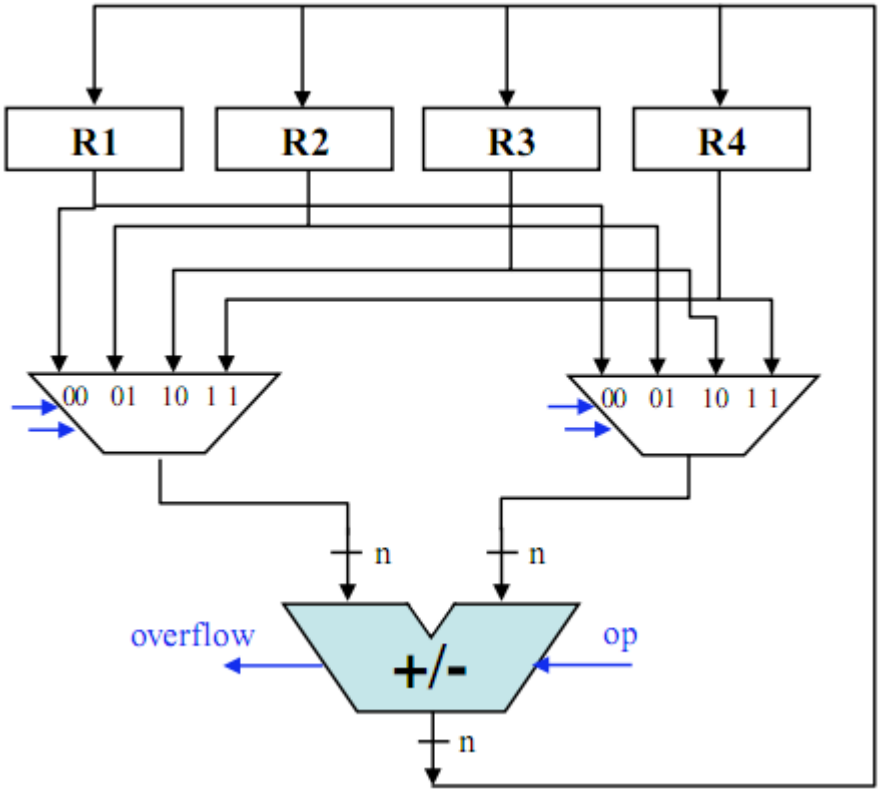
```
ENTITY maquina IS
   PORT (Reset, ck, c : IN STD_LOGIC;
         s, a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         d : OUT STD_LOGIC);
END maquina;
ARCHITECTURE estrutura OF maquina IS
   COMPONENT bo IS
   PORT (clk, Ctotal, Rtotal : IN STD_LOGIC;
         s, a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
         menor : OUT STD_LOGIC);
   END COMPONENT;
   COMPONENT bc IS
   PORT (Reset, clk, c, menor : IN STD_LOGIC;
          d, Ctotal, Rtotal : OUT STD_LOGIC);
   END COMPONENT;

   SIGNAL Ctotal, Rtotal, menor : STD_LOGIC;
BEGIN
   b_operativo: bo PORT MAP (clk, Ctotal, Rtotal, s, a, menor);
   b_controle: bc PORT MAP (Reset, clk, c, menor, d, Ctotal, Rtotal);
END estrutura;
```



Projeto de Sistemas Digitais

BO com Multiplexadores

São necessários 4 sinais para controlar o acesso a Unidade Funcional (UF)

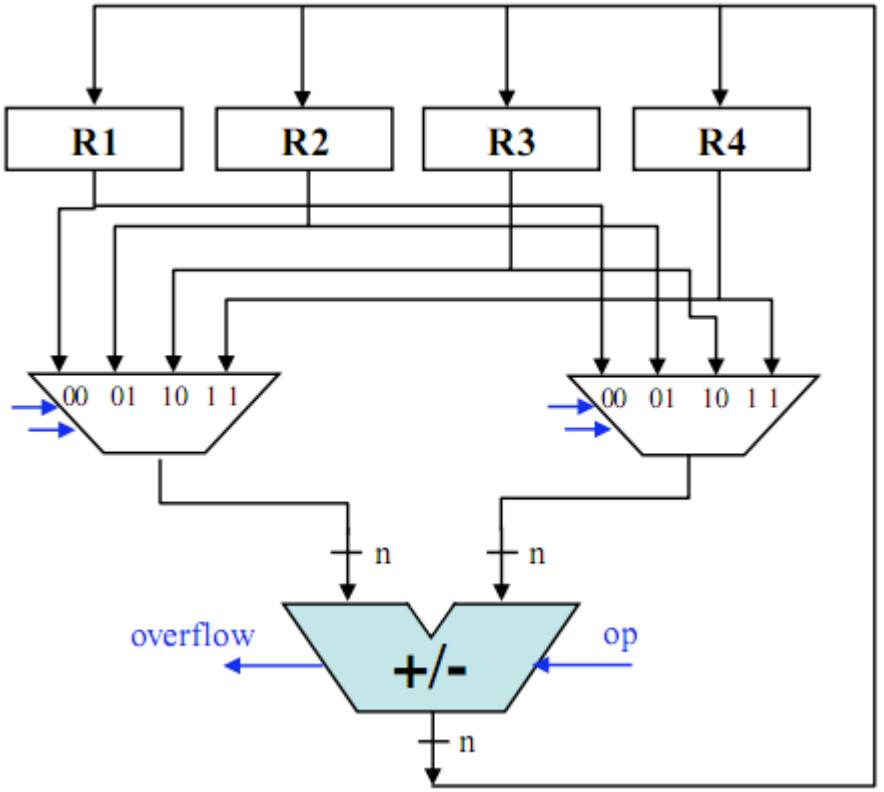


BO com Multiplexadores

Porém:

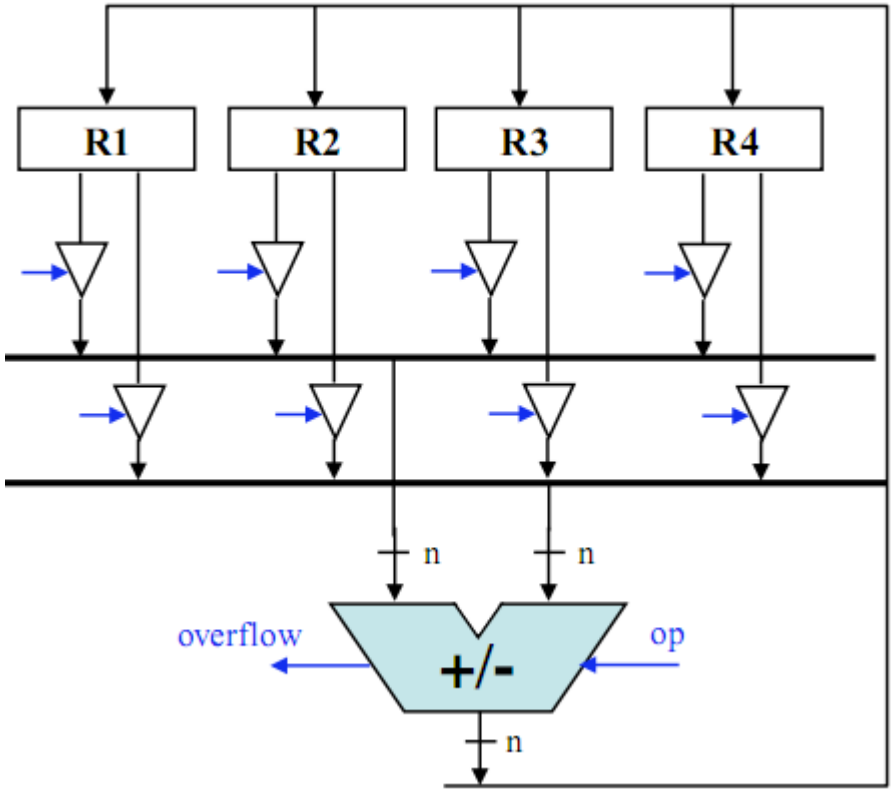
- Se qualquer registrador pode ser fonte ou destino de dados para ambas as entradas da UF

- E se somente dois registradores servem de operandos para a UF (a cada ciclo de relógio)



BO com Dispositivos Tri-State

- 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)

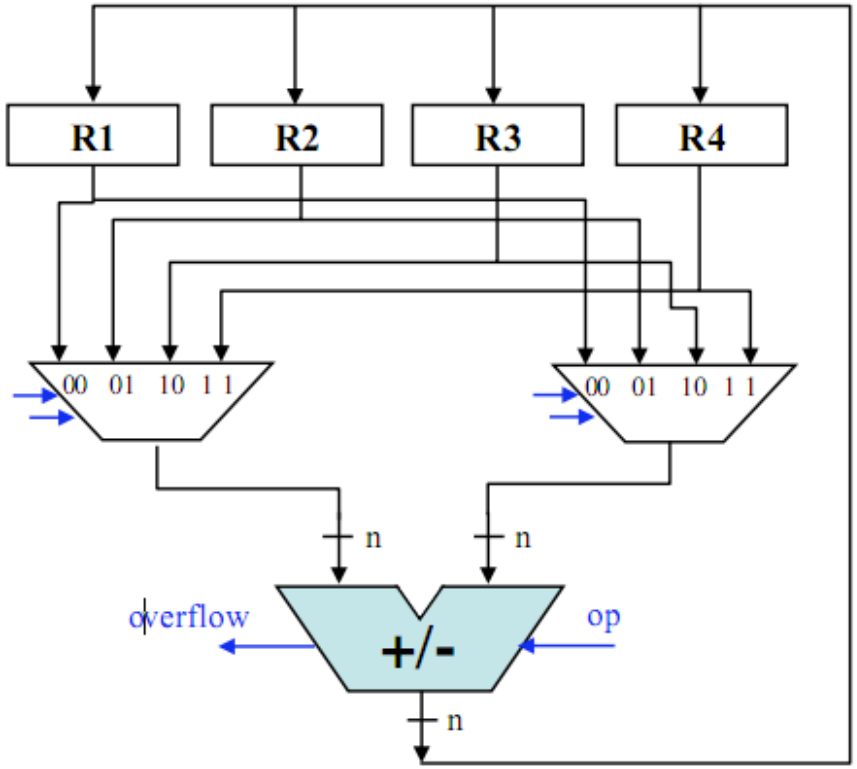


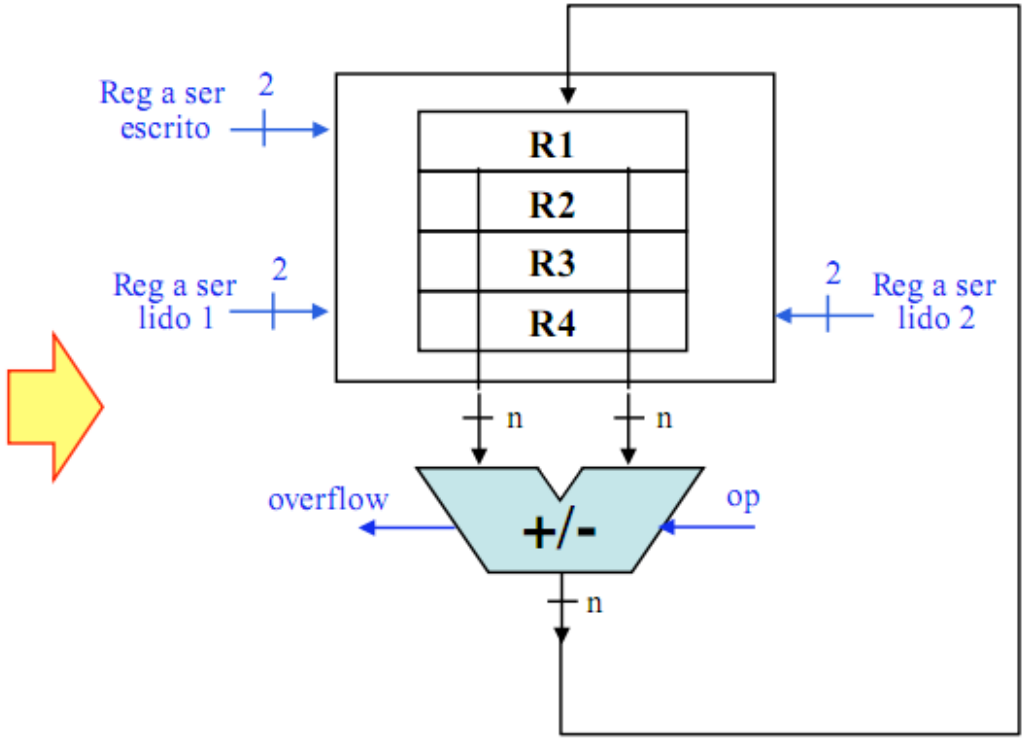
Registradores x Banco de Registradores

- 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”

Projeto de Sistemas Digitais

Registradores x Banco de Registradores

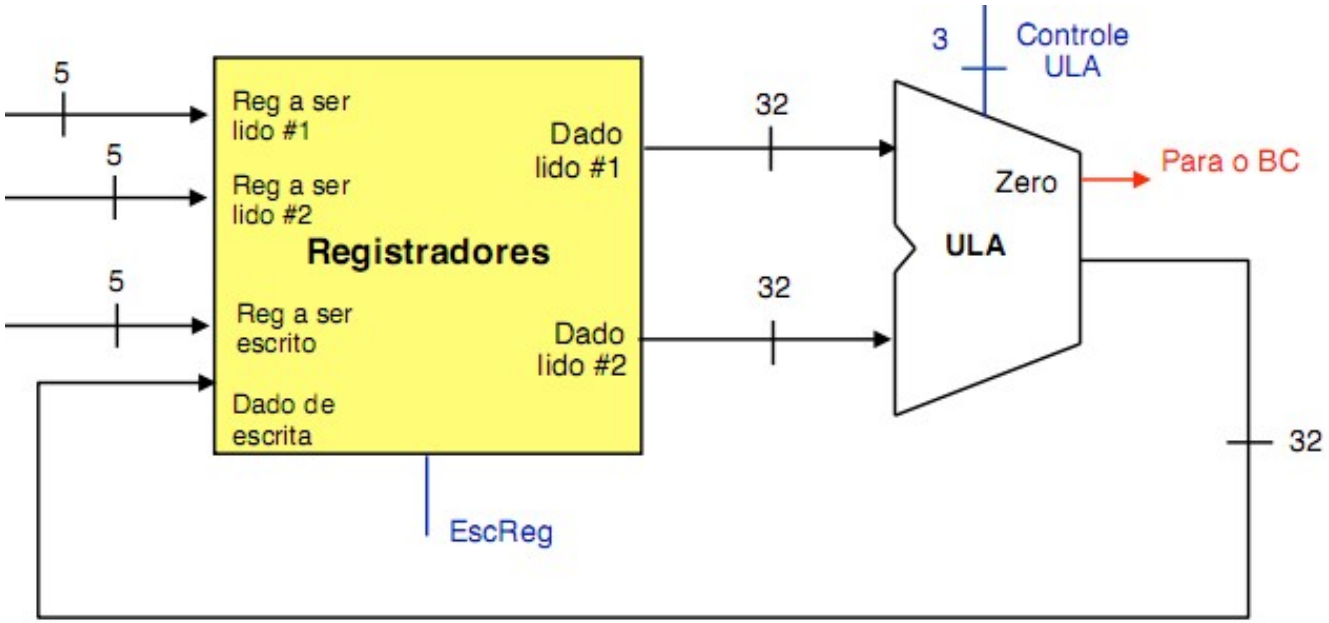




- 4 registradores
- 2 “portas” de leitura e 1 “porta” de escrita
- 2 bits de endereço por “porta”

Banco de Registradores

Exemplo:Banco de registradores de um microprocessador



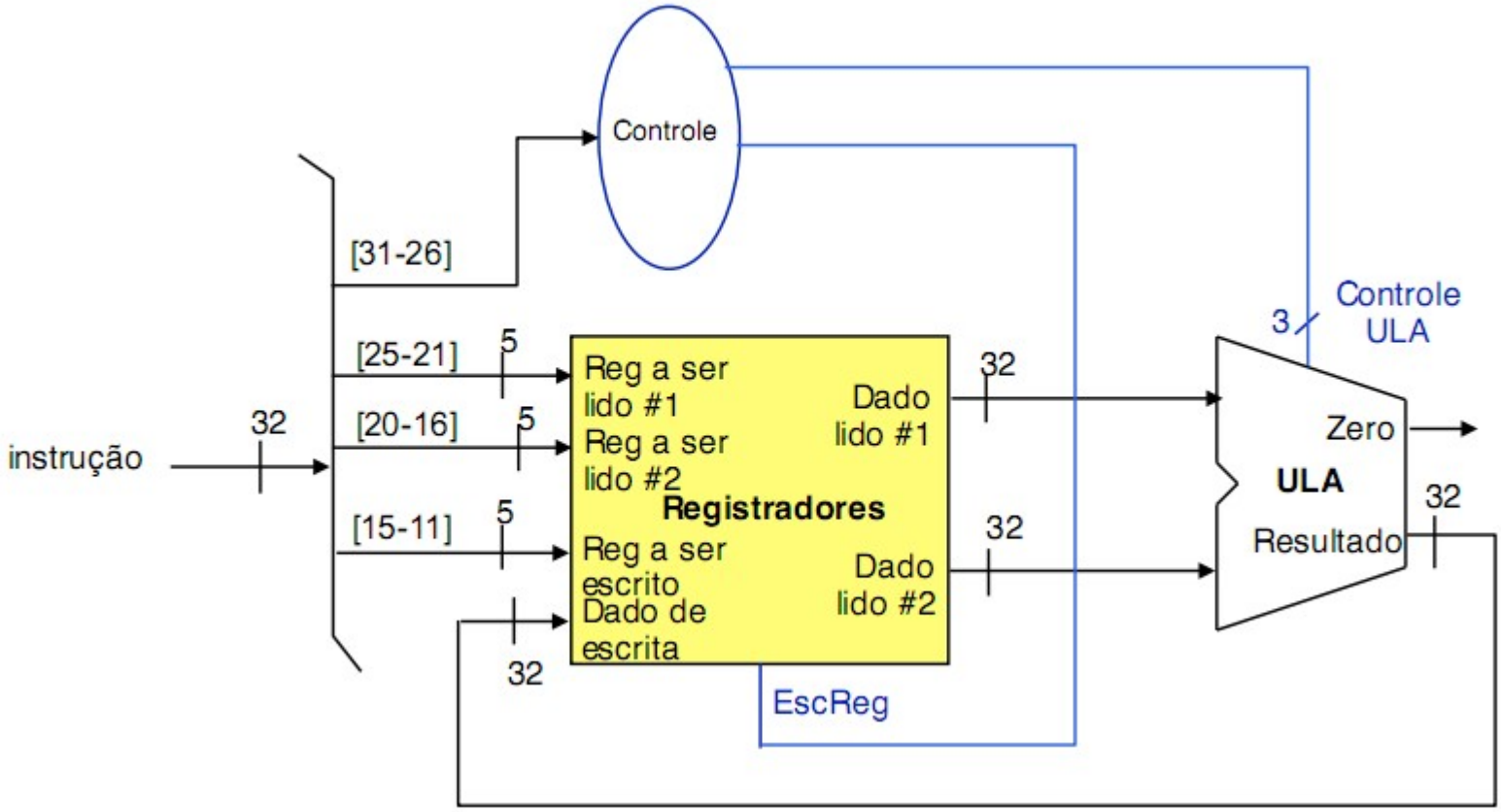
- 2 “portas” de leitura e 1 “porta” de escrita (há um sinal p/ habilitar escrita)

- Quantos registradores há neste banco de registradores?
- Qual o comprimento (ou tamanho) dos dados?

Projeto de Sistemas Digitais

Banco de Registradores

Exemplo:Banco de registradores de um microprocessador



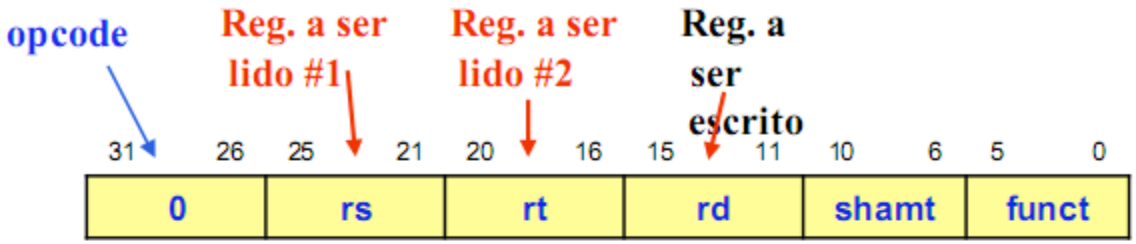
Projeto de Sistemas Digitais

Conjunto de Instruções

Exemplo de instruções (microprocessador MIPS)

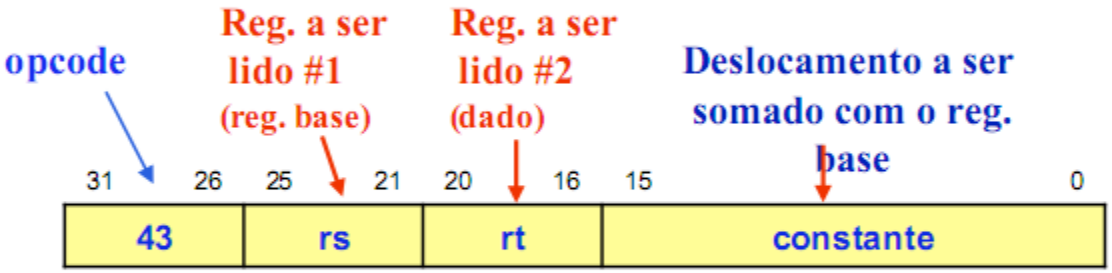
Instrução tipo R:

```
add $s1,$s2, $s3
($s1 ← $s2 + $s3)
```



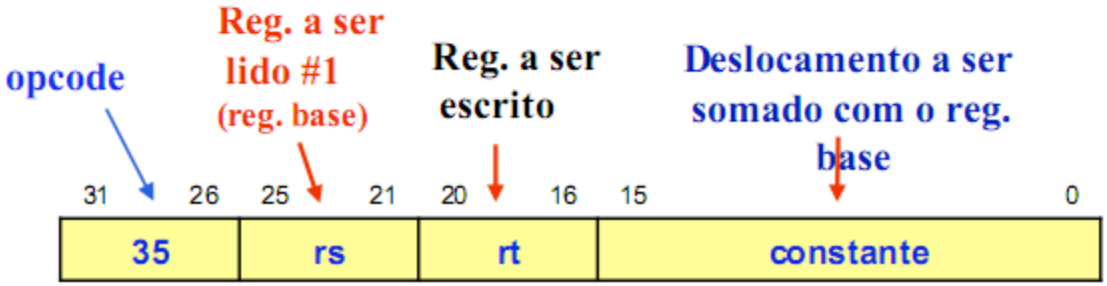
Instrução store word :

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



Instrução load word:

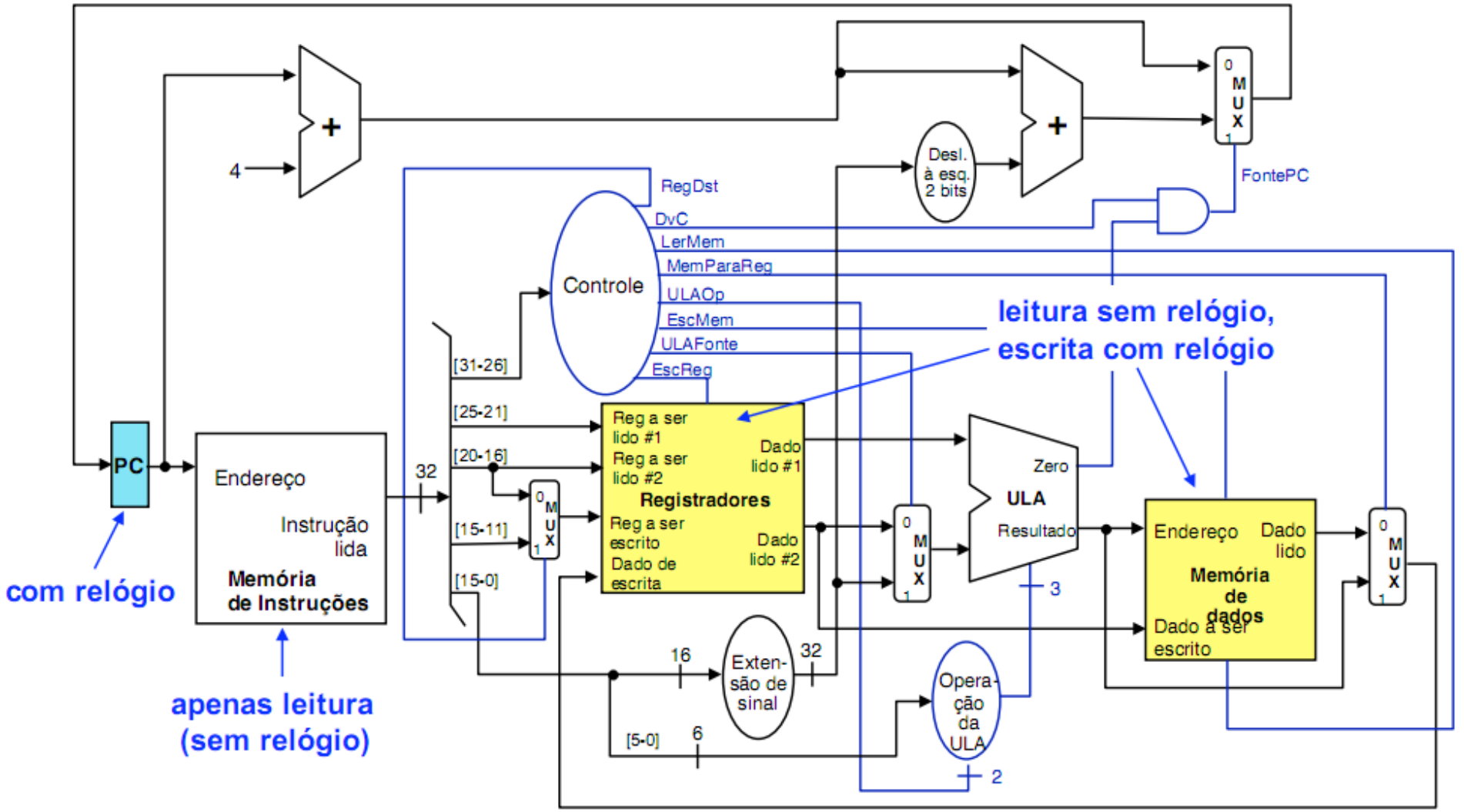
```
lw $s1, constante($s2)
($s1 ← Mem[$s2 + constante] )
```



Projeto de Sistemas Digitais

O Processador MIPS: BO + BC

Exemplo de instruções (microprocessador MIPS)



Projeto de Sistemas Digitais

Execução de uma Instrução Tipo R

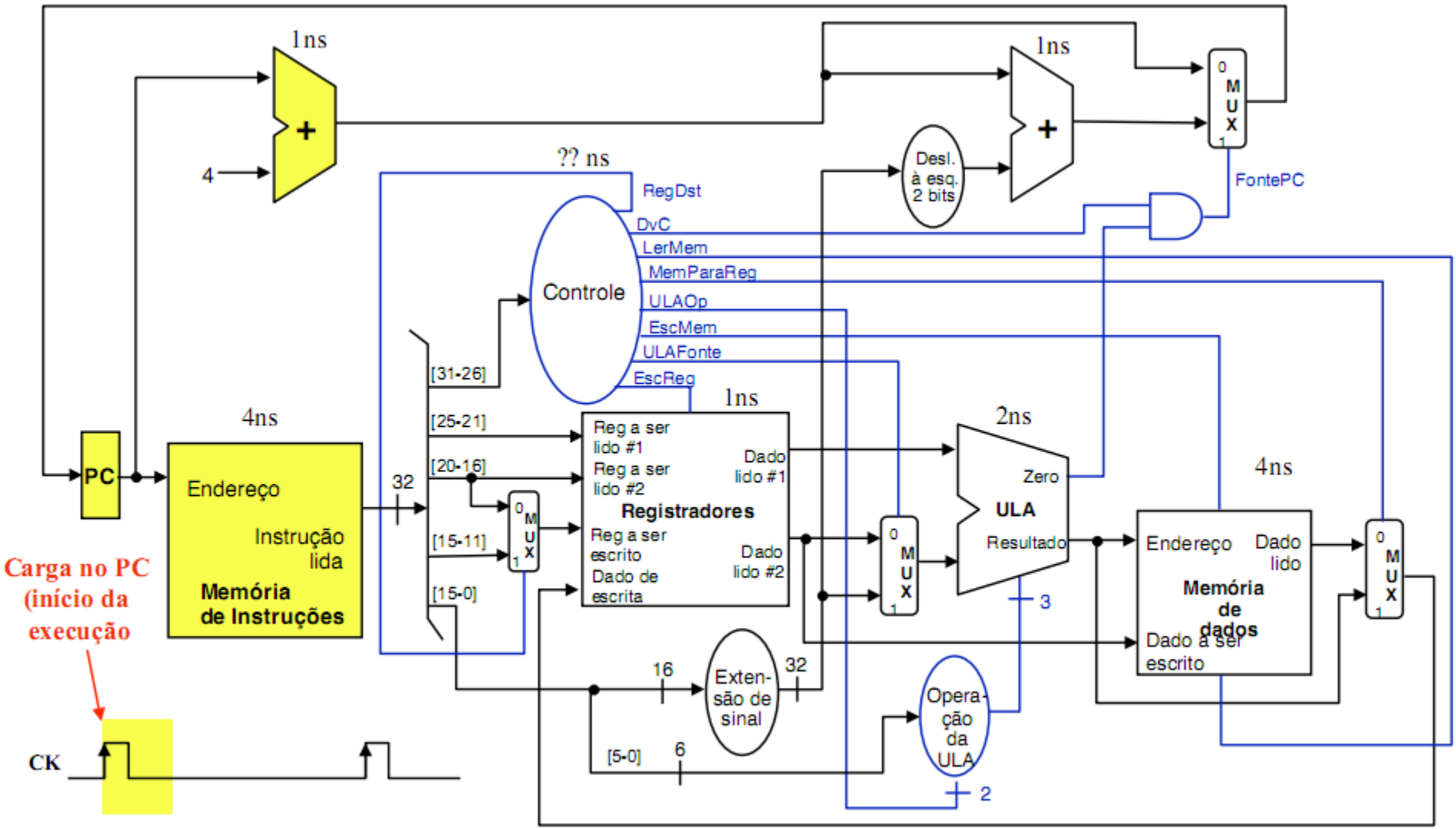
Seja uma instrução tipo R -> add $t1, $t2, $t3:
Esta instrução pode ser executada em 4 etapas:

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.

Instrução tipo R: busca da instrução

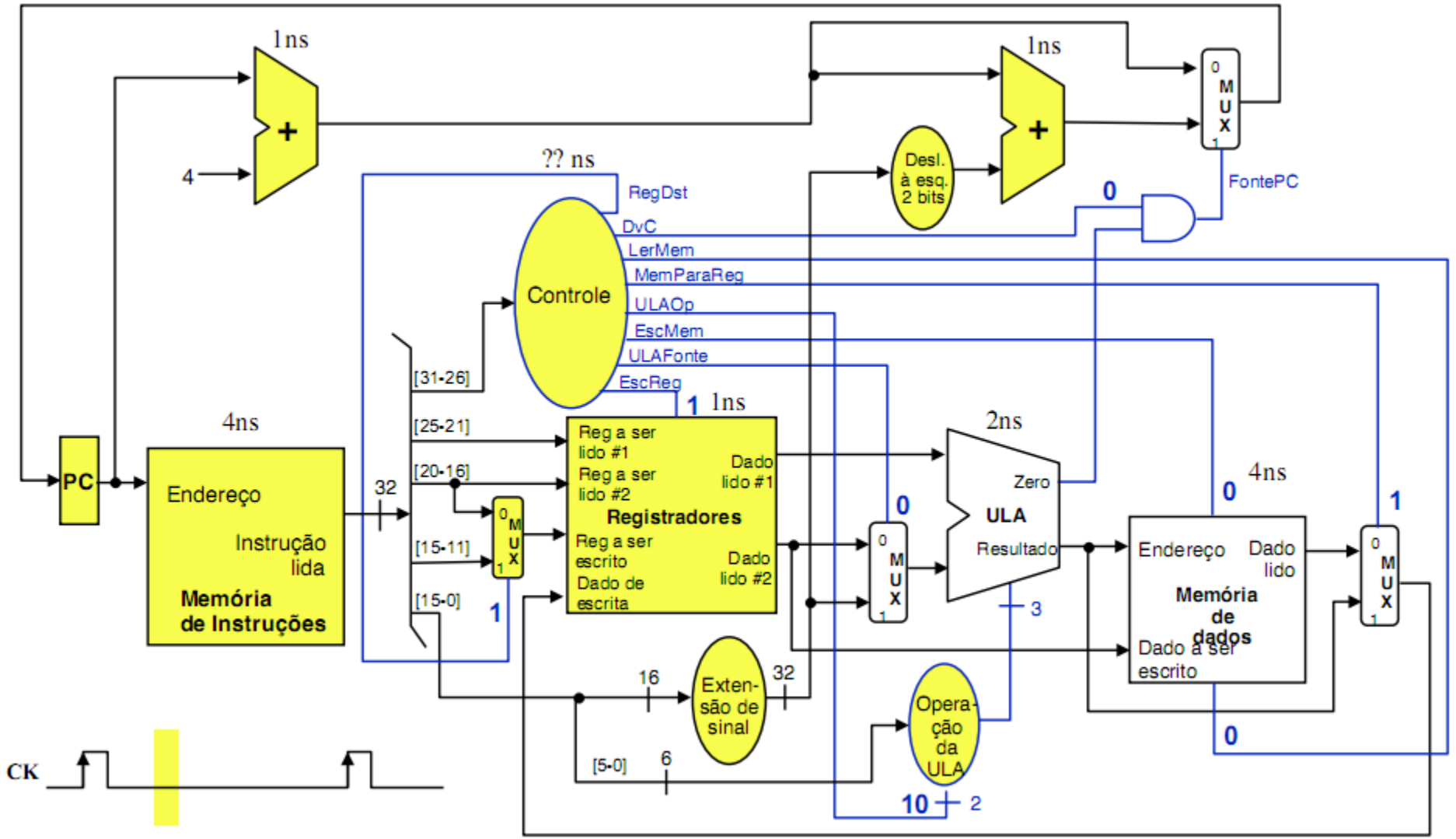
busca da instrução e cálculo de PC+4



Projeto de Sistemas Digitais

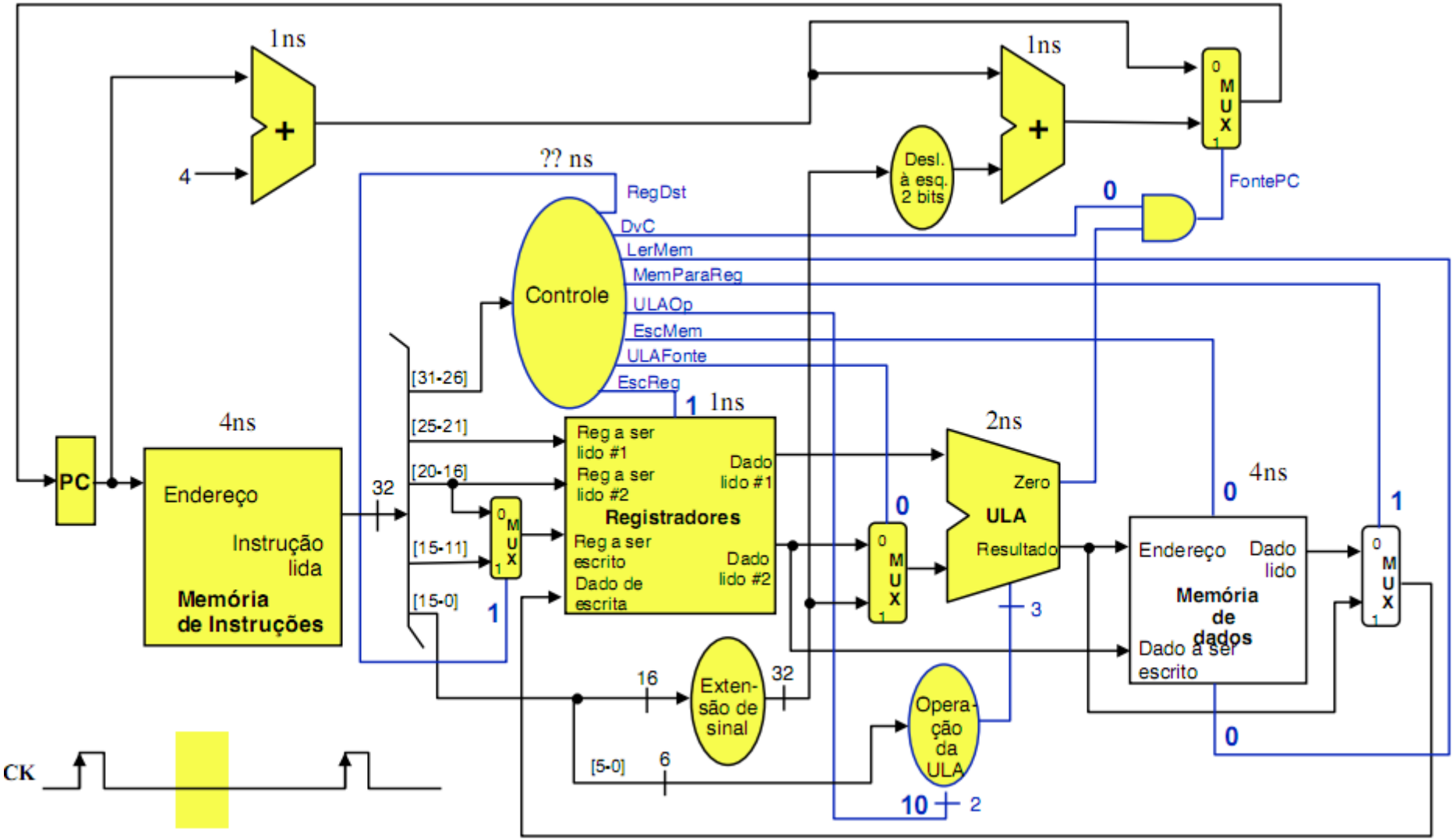
Instrução tipo R: busca da instrução

leitura de Rs e Rt e geração sinais de controle



Instrução tipo R: busca da instrução

operação na ULA (depende de “funct”)



Projeto de Sistemas Digitais

Instrução tipo R: busca da instrução

escrita no registrador-destino