Universidade Federal do Rio Grande do Sul
Instituto de Informática

Organização de Computadores
Aula 07

Bloco de controle multi-ciclo
Projeto com FSM – Máquina de Estados Finitos

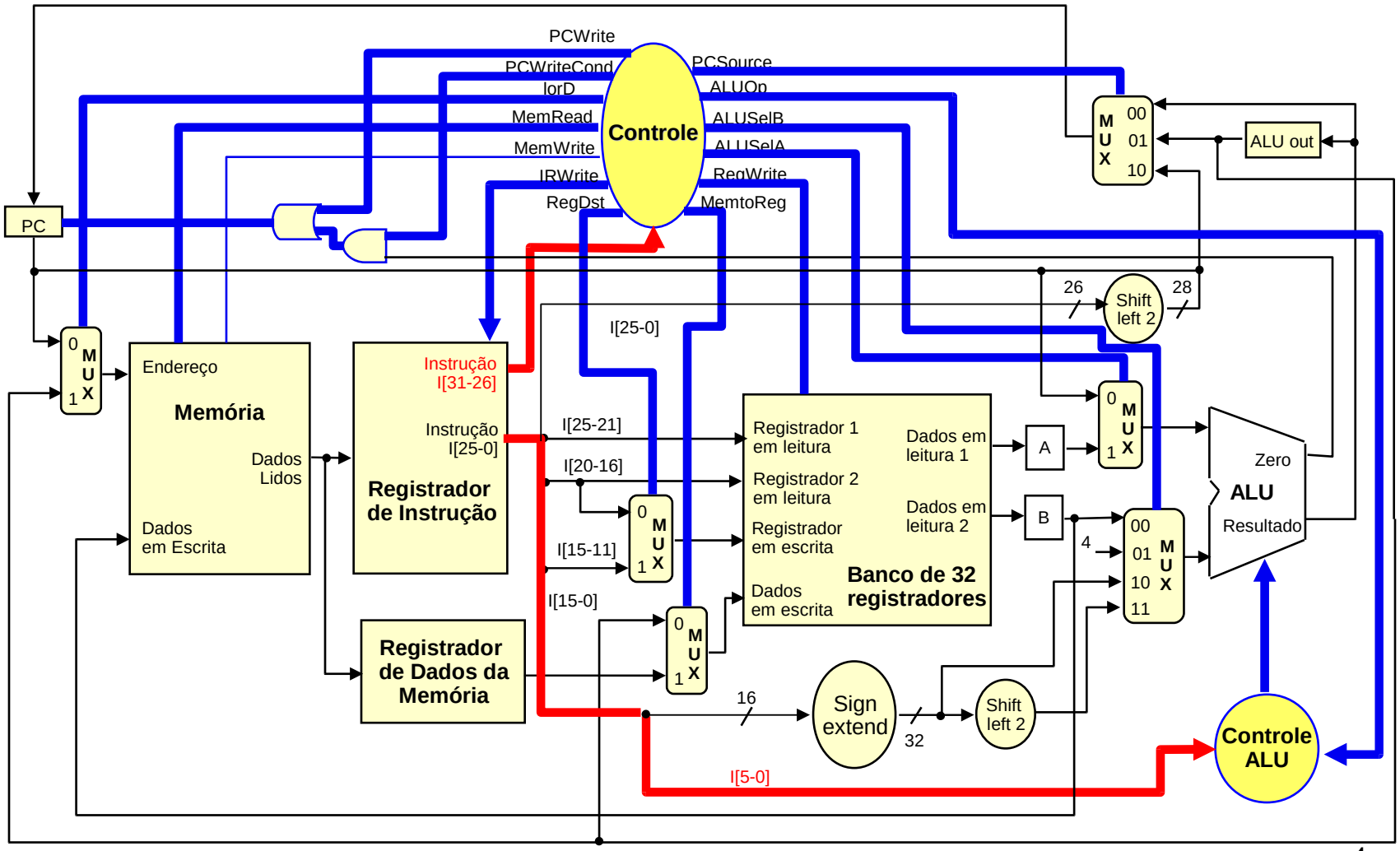
Introdução

- **Na aula passada foi visto o caminho de dados para uma unidade multiciclo, com a determinação dos sinais de controle necessários para ativá-los,**
- **Agora veremos a Implementação da Unidade de Controle Multiciclo,**
- **Na unidade de controle Mono-ciclo foi empregado Tabelas de Verdade ,**
- **No Multiciclo é um pouco mais complicado pois as instruções são executadas por passos, ciclos,**
- **Vamos ver duas técnicas:**
  - **Baseado numa Máquina de Estados Finitos - FSM (nesta aula),**
  - **Microprogramação (próxima aula).**

Bloco de controle – projeto com FSM

1. Sinais de controle
2. Máquina de estados finitos
    - Busca e decodificação de instruções
    - Instruções de referência à memória
    - Instruções de tipo R
    - Instrução de desvio condicional (branch)
    - Instrução de desvio incondicional (jump)
3. Implementando a FSM

1. Sinais de controle



Sinais de controle de 1 bit

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador-destino no banco de registradores vem do campo rt	O número do registrador-destino no banco de registradores vem do campo rd.
EscReg	Nenhum	O registrador de propósito geral selecionado pelo número do registrador de escrita é atualizado com o valor da entrada Dado de Escrita.
UALFonteA	O primeiro operando da UAL é o PC	O primeiro operando da UAL vem do registrador A.
LerMem	Nenhum	O conteúdo da memória no endereço especificado na entrada Endereço é colocado na saída Dado de Saída.
EscMem	Nenhum	O conteúdo da memória no endereço especificado na entrada Endereço é substituído pelo valor na entrada Dado a ser Escrito.
MemParaReg	O valor colocado na entrada Dado a ser Escrito do banco de registradores vem de UALOut.	O valor na entrada Dado a ser Escrito do banco de registradores vem do MDR.
IouD	O PC é usado para fornecer o endereço da unidade de memória.	UALSaída é usada para fornecer o endereço para a unidade de memória.
IREsc	Nenhum	A saída da unidade de memória é escrita no IR.
PCEsc	Nenhum	O PC é atualizado. A fonte é controlada pelo sinal FontePC.
PCEscCond	Nenhum	O PC é atualizado se a saída Zero da UAL também estiver ativa.

Sinais de Controle de 2 bits

Nome do sinal	Valor	Efeito
UALOp	00	A UAL efetua uma operação de soma.
	01	A UAL efetua uma operação de subtração.
	10	O campo funct (função) da instrução determina a operação da UAL.
UALFonteB	00	A segunda entrada da UAL vem do registrador B.
	01	A segunda entrada da UAL é a constante 4.
	10	A segunda entrada da UAL é a extensão do sinal dos 16 bits menos significativos do IR.
	11	A segunda entrada da UAL é a extensão do sinal dos 16 bits menos significativos do IR deslocados 2 bits à esquerda.
FontePC	00	A saída da UAL (PC + 4) é enviada ao PC para atualizar seu valor.
	01	O conteúdo de UALSaída (o endereço-alvo do desvio condicional) é enviado ao PC para atualizar seu valor.
	10	O endereço-alvo do desvio incondicional (IR[25–0]), deslocado 2 bits à esquerda e concatenado com PC + 4[31–28] é enviado ao PC para atualizar seu valor.

Sinais de controle

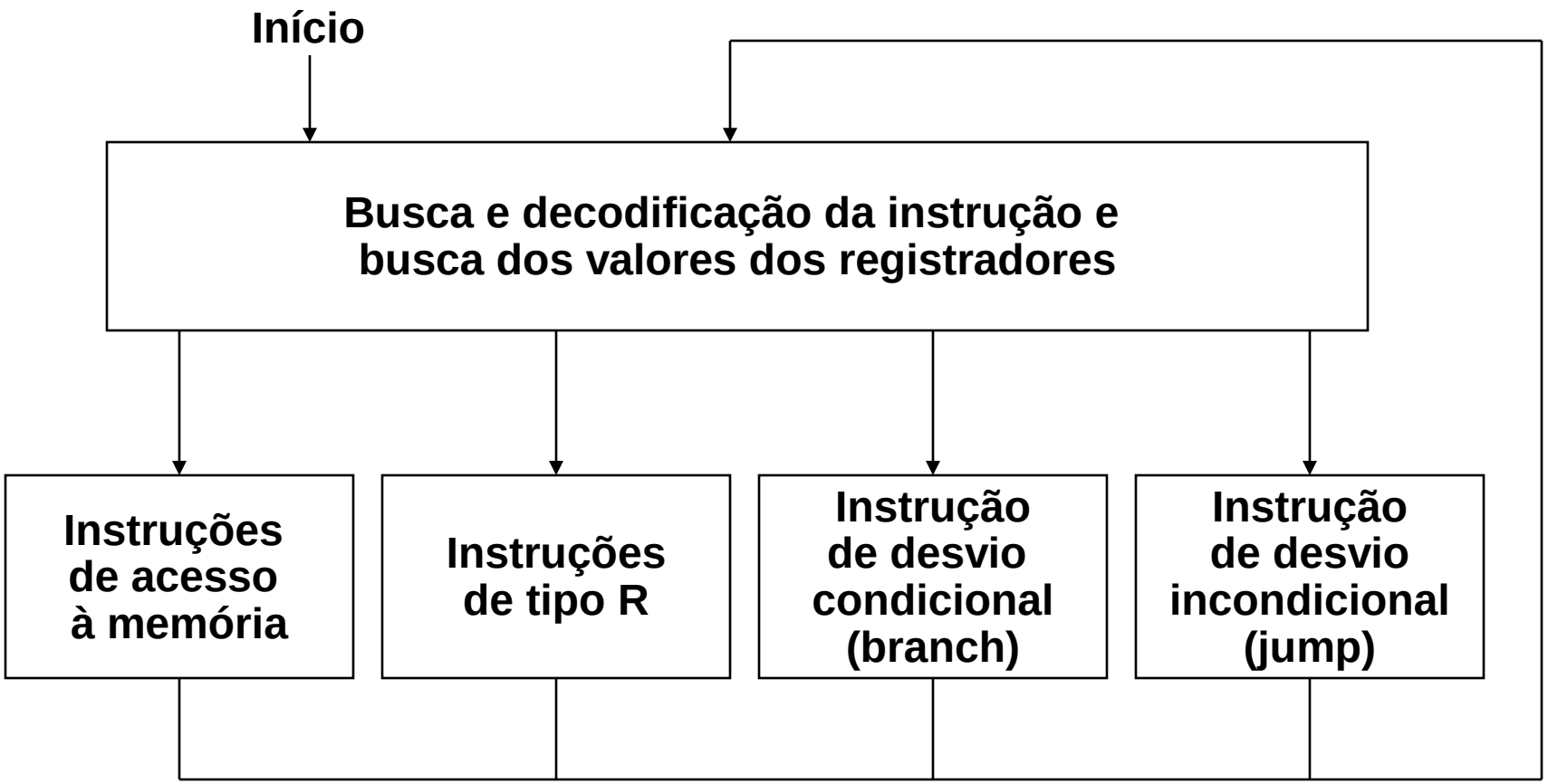
MemRead = 1	**Ler conteúdo da memória**
MemWrite = 1	**Escrever conteúdo na memória**
ALUSelA = 0	**PC é primeiro operando para ALU**
ALUSelA = 1	**Registrador A é primeiro operando para ALU**
ALUSelB = 00	**Registrador B é segundo operando para ALU**
ALUSelB = 01	**Constante = 4 é segundo operando para ALU**
ALUSelB = 10	**Deslocamento (estendido p/ 32 bits) é segundo operando para ALU**
ALUSelB = 11	**Deslocamento (estendido p/ 32 bits e deslocado 2 bits p/ esq.) é 2º operando p/ ALU**
RegDst = 0	**Registrador a ser escrito é especificado pelo campo rt**
RegDst = 1	**Registrador a ser escrito é especificado pelo campo rd**
RegWrite = 1	**Escrever conteúdo em registrador**
MemtoReg = 0	**Valor a ser escrito em registrador vem do registrador da saída da ALU (ALUout)**
MemtoReg = 1	**Valor a ser escrito em registrador vem do registrador de dados da memória (MDR)**
IorD = 0	**Endereço em memória vem do PC**
IorD = 1	**Endereço em memória vem da ALU**
IRWrite = 1	**Armazenar instrução no IR**

Sinais de controle para o PC

PCWrite = 1	**PC é carregado; fonte é controlada por PCSource**
PCWriteCond = 1	**PC é carregado se saída Zero da ALU está ligada**

PCSource	**00**	**Saída da ALU é enviada para o PC**
	01	**Conteúdo do registrador ALUout é enviado para o PC**
	10	**Endereço de destino do Jump enviado para o PC**

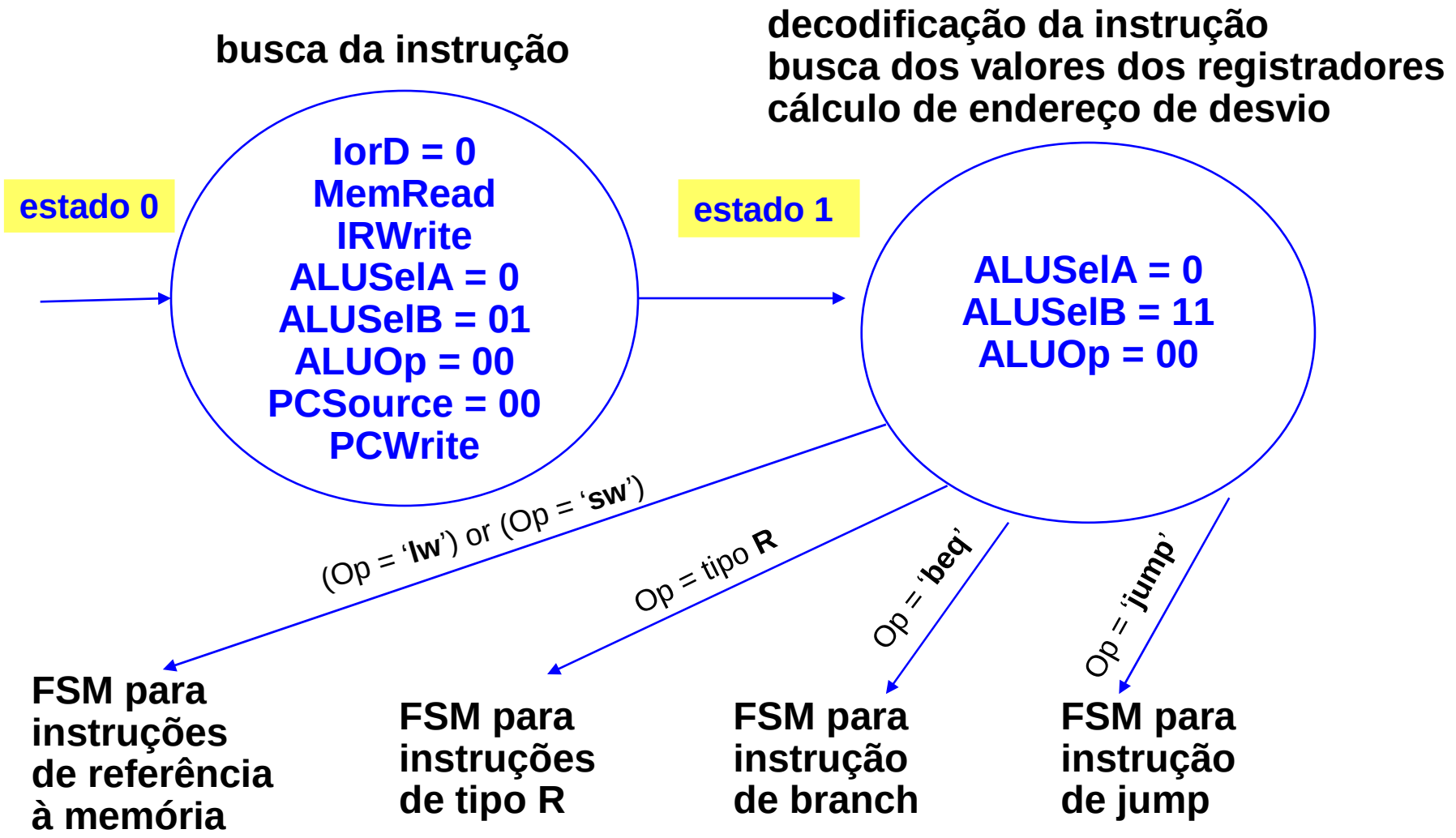
2. Máquina de estados finitos - FSM



Estado 0
Busca e Decodificação de Instruções

Estado 1
Busca dos Valores dos Registradores

FSM: busca e decodificação de instruções

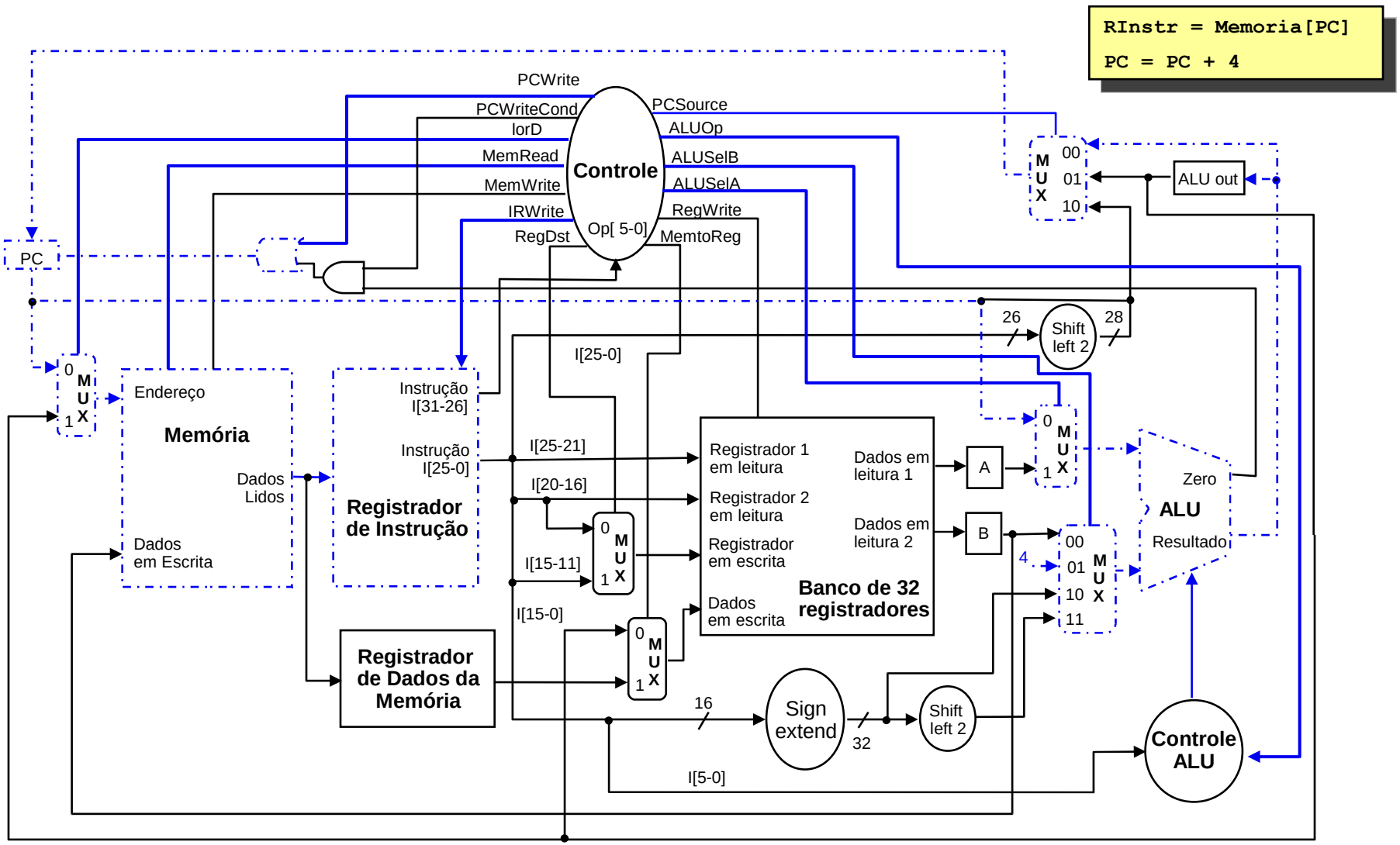


FSM: busca e decodificação de instruções

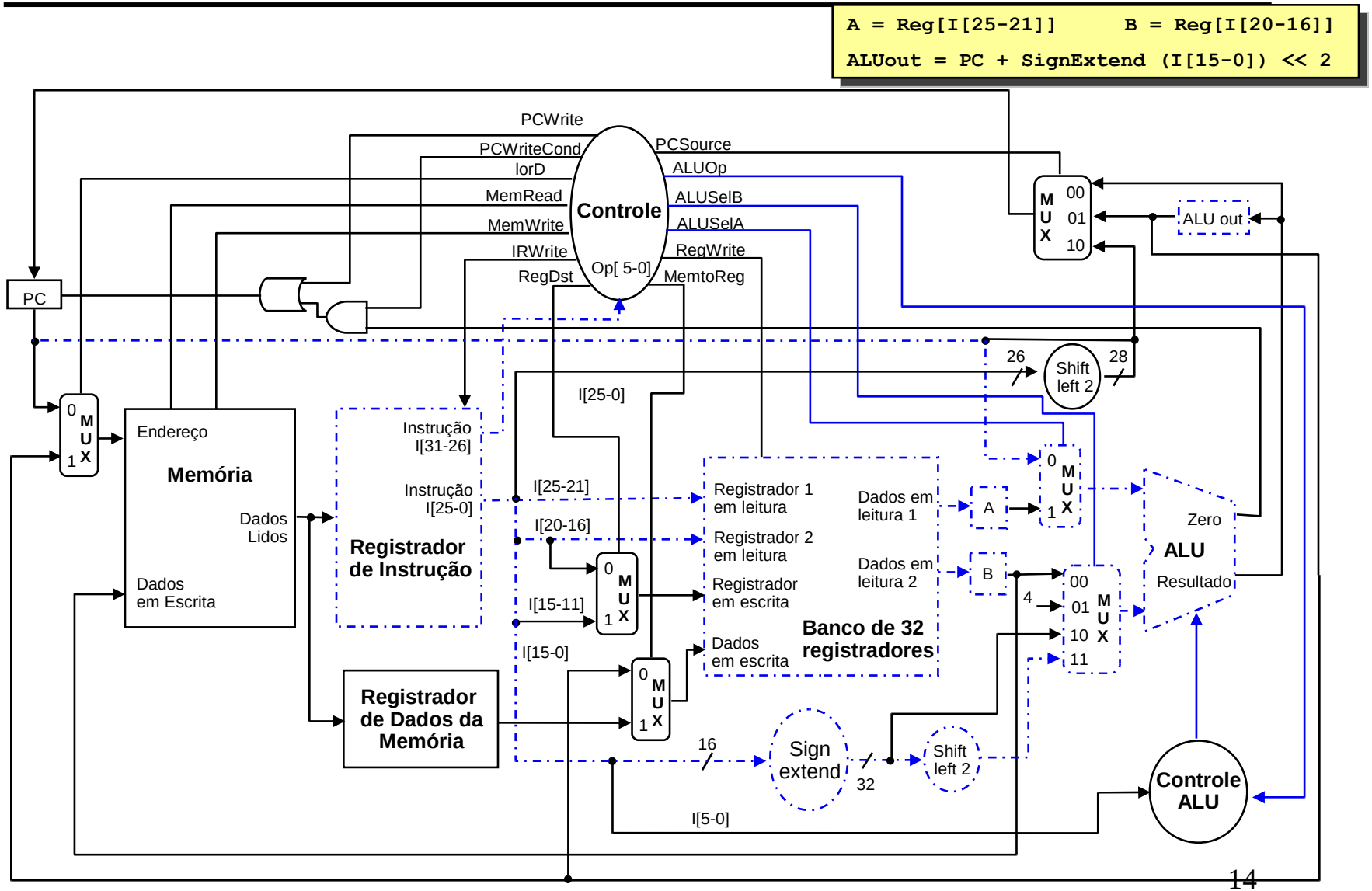
- **estado 0 – busca da instrução**
  - **IorD = 0:** **PC é fonte de endereço para a memória**
  - **MemRead:** **ler instrução da memória**
  - **IRWrite:** **escrever instrução no IR**
  - **ALUSelA = 0:** **PC é primeiro operando para ALU**
  - **ALUSelB = 01:** **constante 4 é segundo operando para ALU**
  - **ALUOp = 00:** **ALU soma PC + 4**
  - **PCSource = 00:** **Saída da ALU é enviada para o PC**
  - **PCWrite:** **PC é escrito com PC + 4**

- **estado 1 – decodificação da instrução, busca dos valores dos registradores, cálculo do endereço de desvio condicional (branch)**
  - **ALUSelA = 0:** **PC é primeiro operando para ALU**
  - **ALUSelB = 10:** **deslocamento é segundo operando para ALU**
  - **ALUOp = 00:** **ALU soma PC + deslocamento**

Estado 0 - Busca de instrução

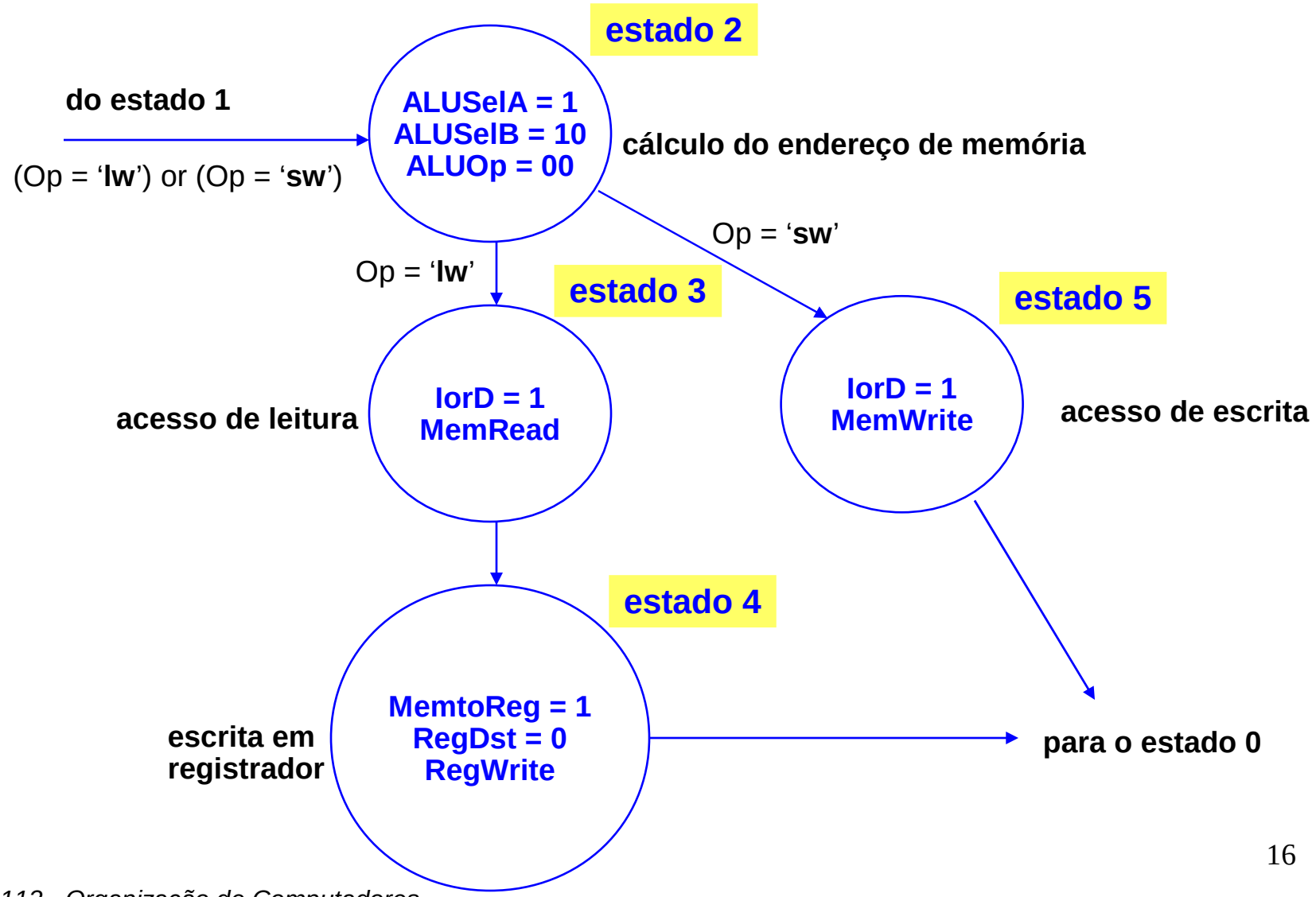


Estado 1 - Decodificação de instrução (+ leitura de registradores, + cálculo de end. branch)



Estados 2, 3, 4 e 5
Instruções de Referência à Memória

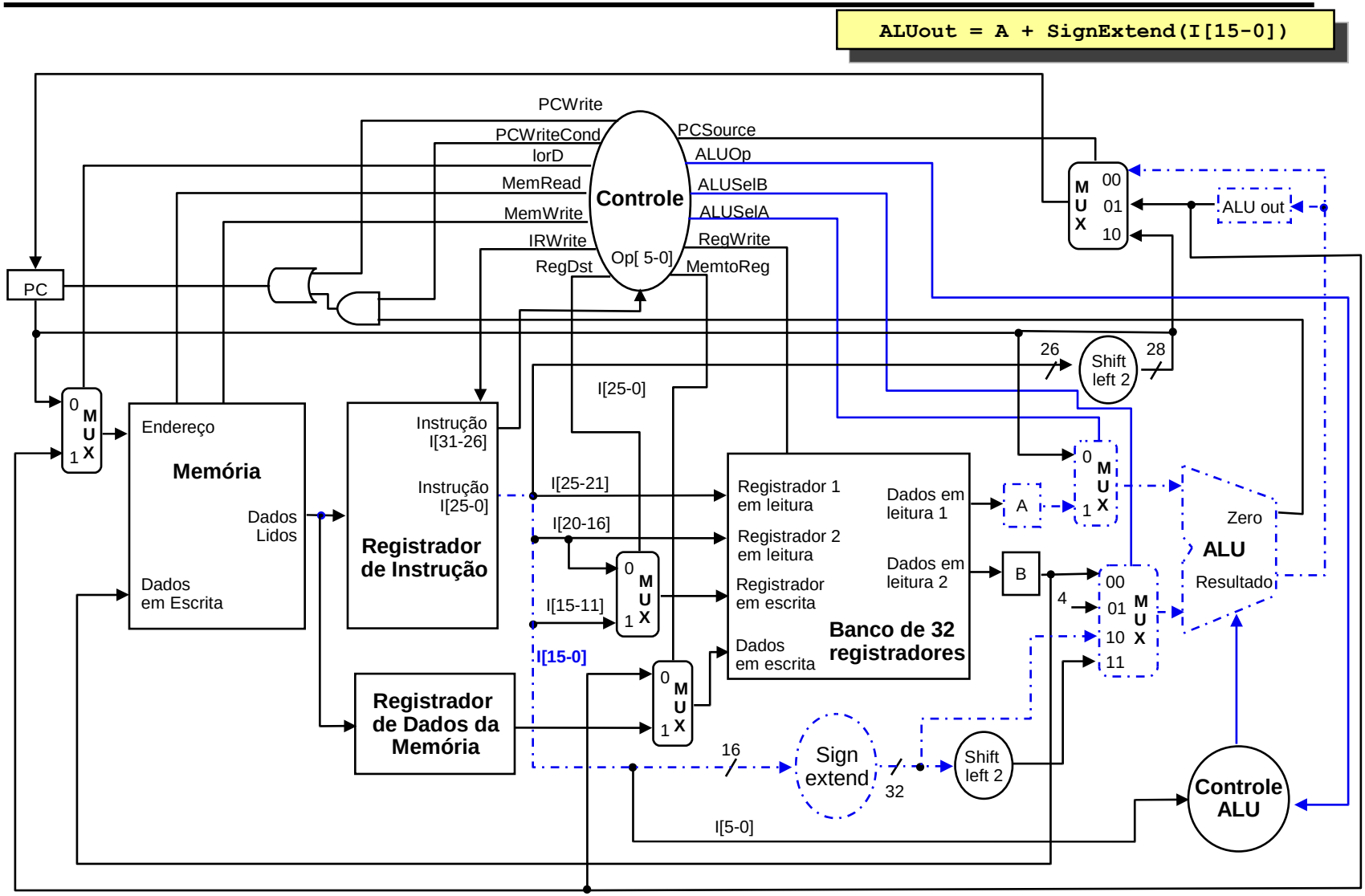
FSM: instruções de referência à memória



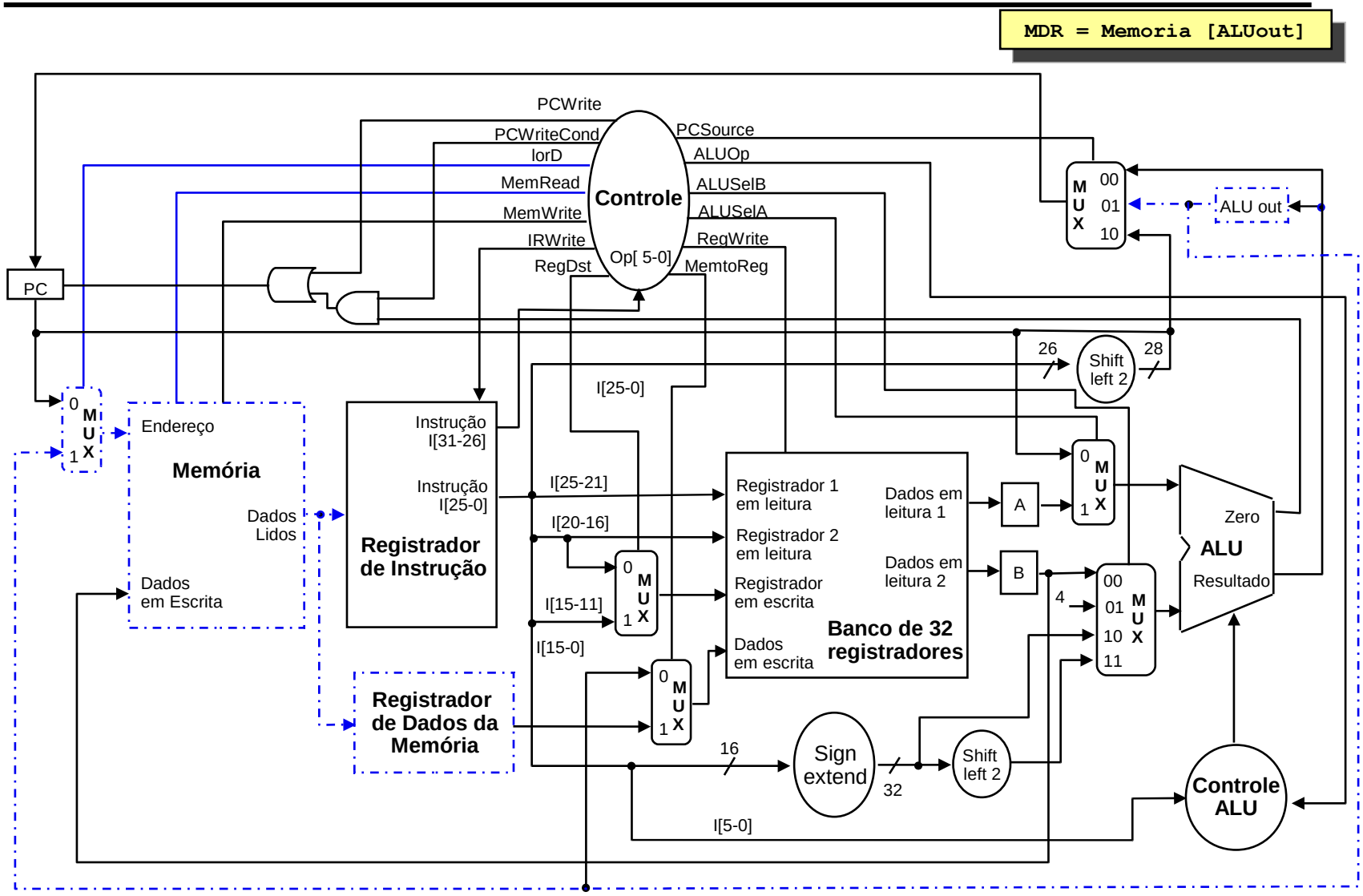
FSM: instruções de referência à memória

- **estado 2 – cálculo do endereço efetivo**
  - **ALUSelA = 1: registrador base é primeiro operando para ALU**
  - **ALUSelB = 10: deslocamento é segundo operando para ALU**
  - **ALUOp = 00: ALU soma base + deslocamento**
- **estado 3 – acesso de leitura na memória**
  - **IorD = 1: endereço de memória vem da ALU**
  - **MemRead: leitura na memória**
- **estado 4 – escrita em registrador**
  - **MemtoReg = 1: valor para registrador vem da memória**
  - **RegDst = 0: registrador a ser escrito especificado pelo campo rd**
  - **RegWrite: escrita em registrador**
- **estado 5 – acesso de escrita na memória**
  - **IorD = 1: endereço de memória vem da ALU**
  - **MemWrite: escrita na memória**

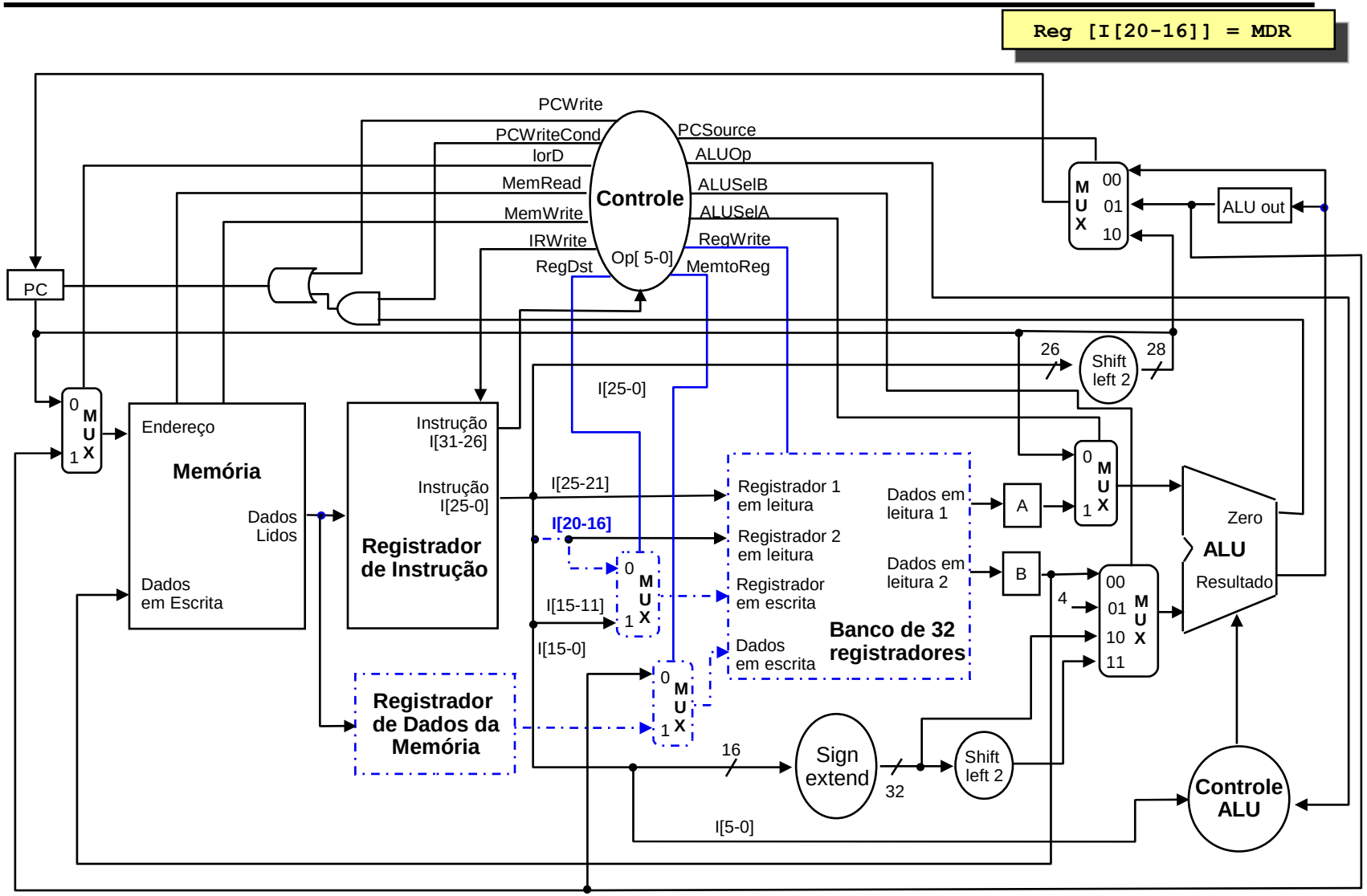
Estado 2 - cálculo do endereço efetivo



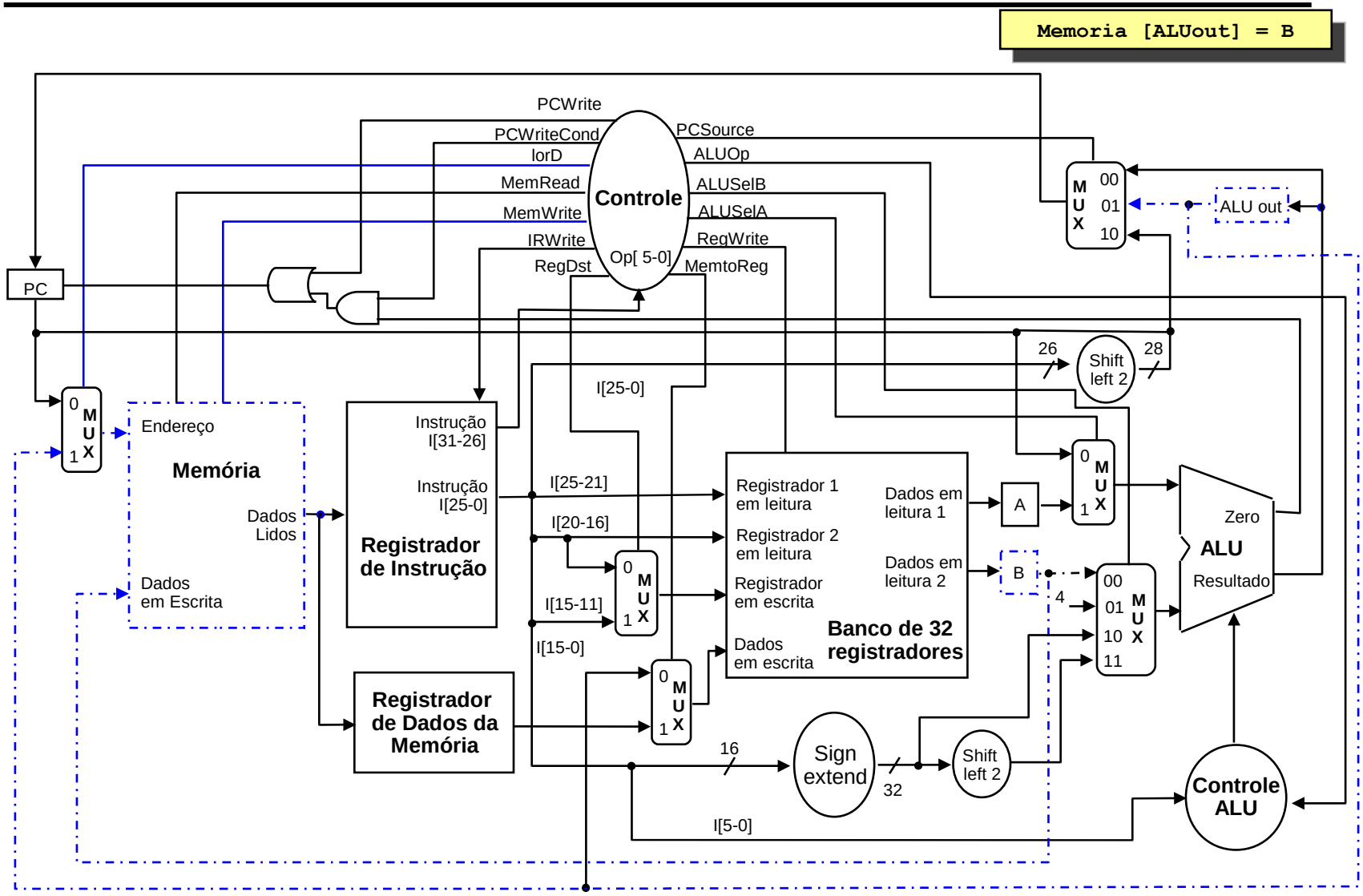
Estado 3 – acesso de leitura na memória (load)



Estado 4 – escrita em registrador (load)

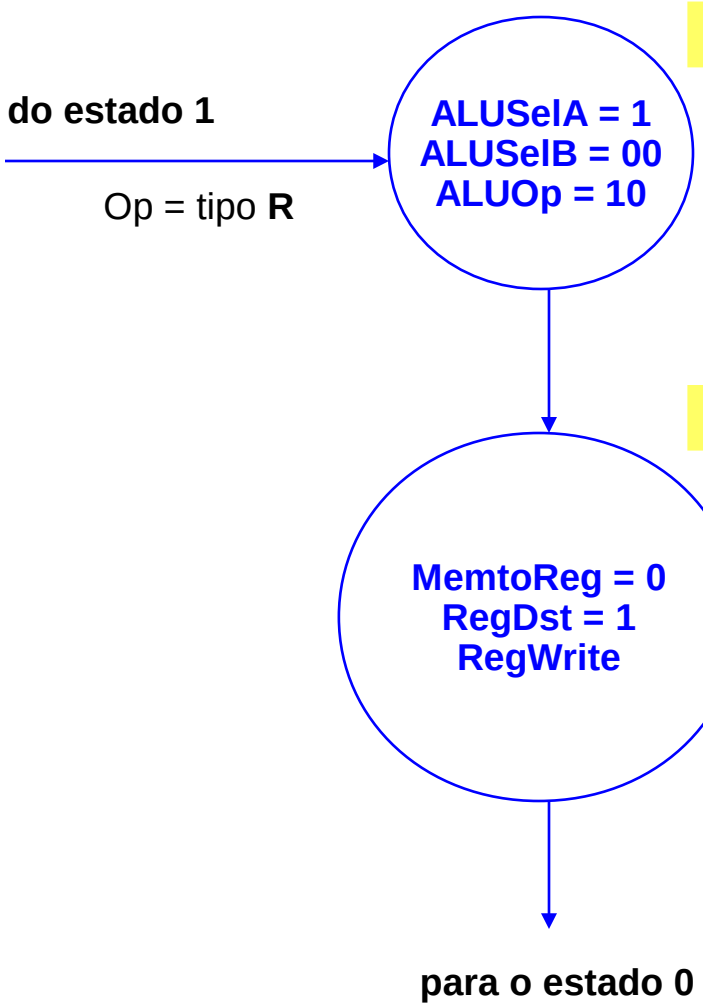


Estado 5 – acesso de escrita na memória (store)



Estados 6 e 7
Instruções de tipo R

FSM: instruções de tipo R



estado 6

execução da operação na ALU

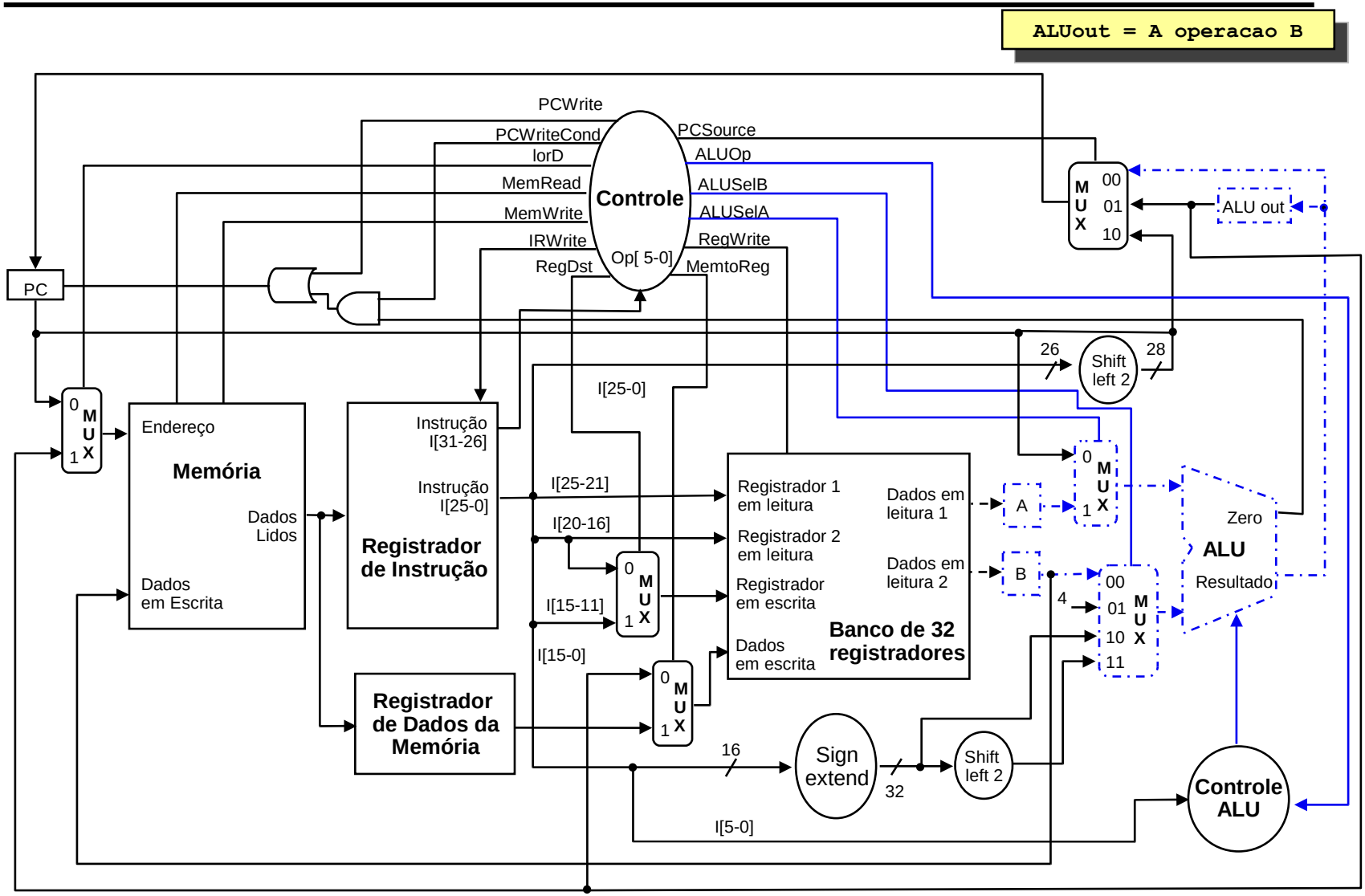
- **ALUSelA = 1: 1º operando vem do registrador A**
- **ALUSelB = 00: 2º operando vem do registrador B**
- **ALUOp = 10: operação vem do campo 'fnct'**

estado 7

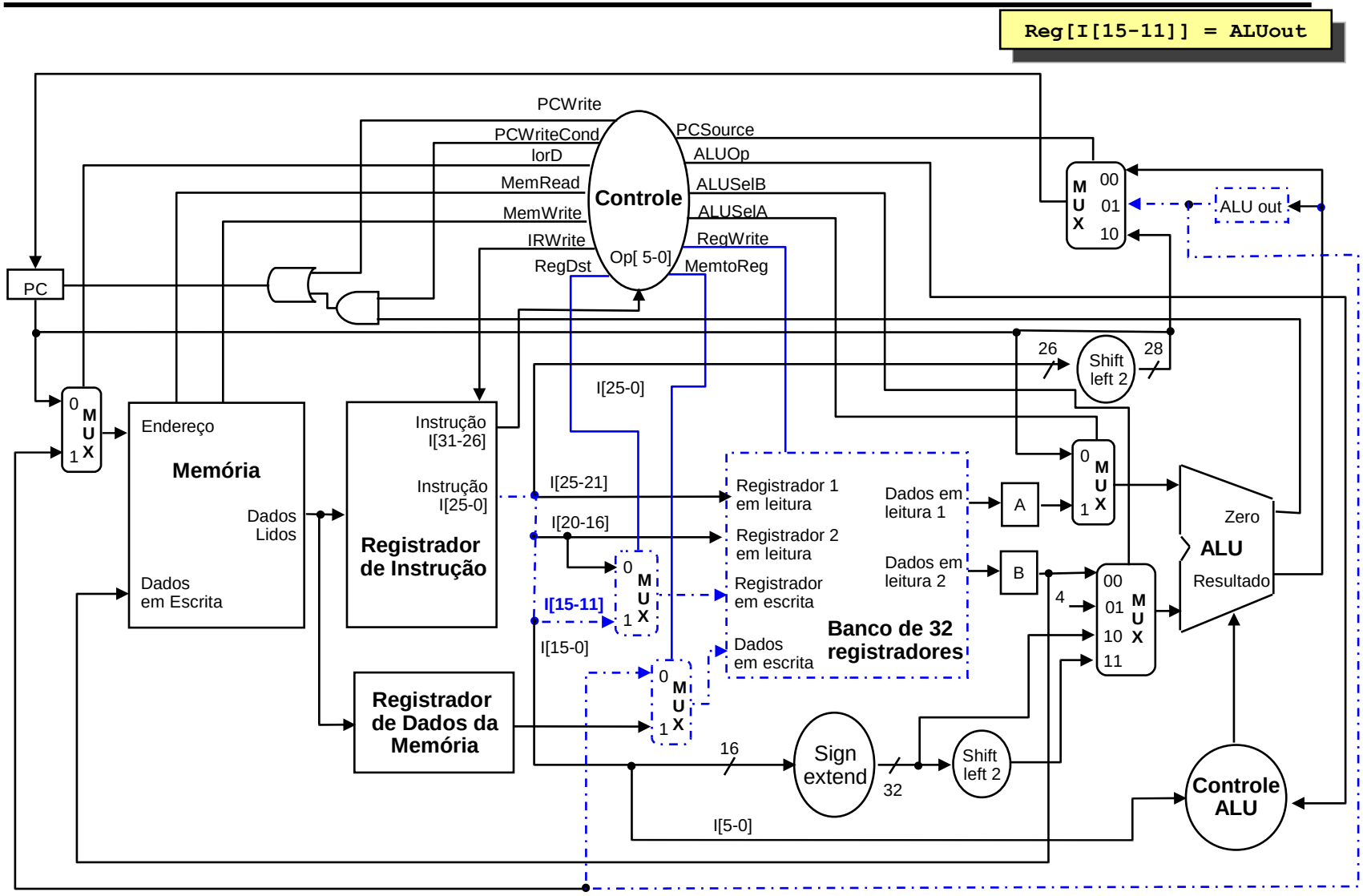
escrita em registrador

- **MemtoReg = 0: valor para reg. vem de ALUout**
- **RegDst = 1: reg. especificado pelo campo 'rd'**
- **RegWrite: escreve em registrador**

Estado 6 - execução da operação na ALU

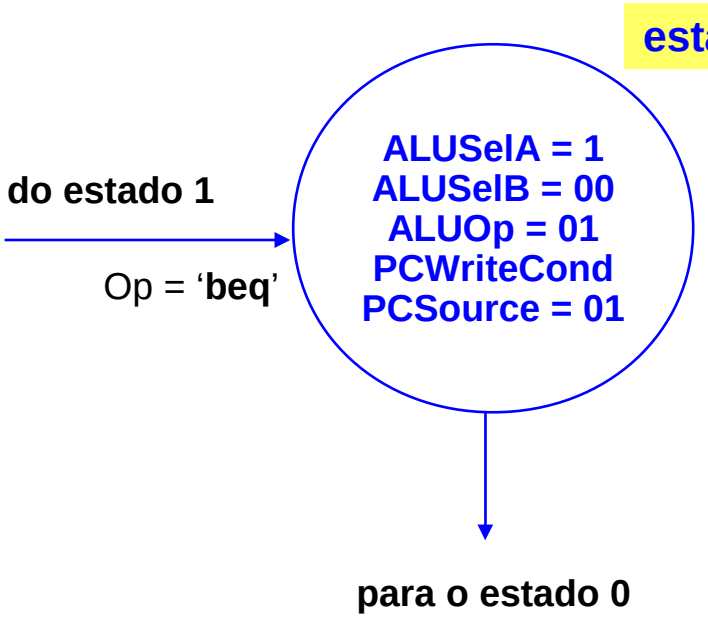


Estado 7 - escrita em registrador



Estado 8
Instrução de Desvio Condicional

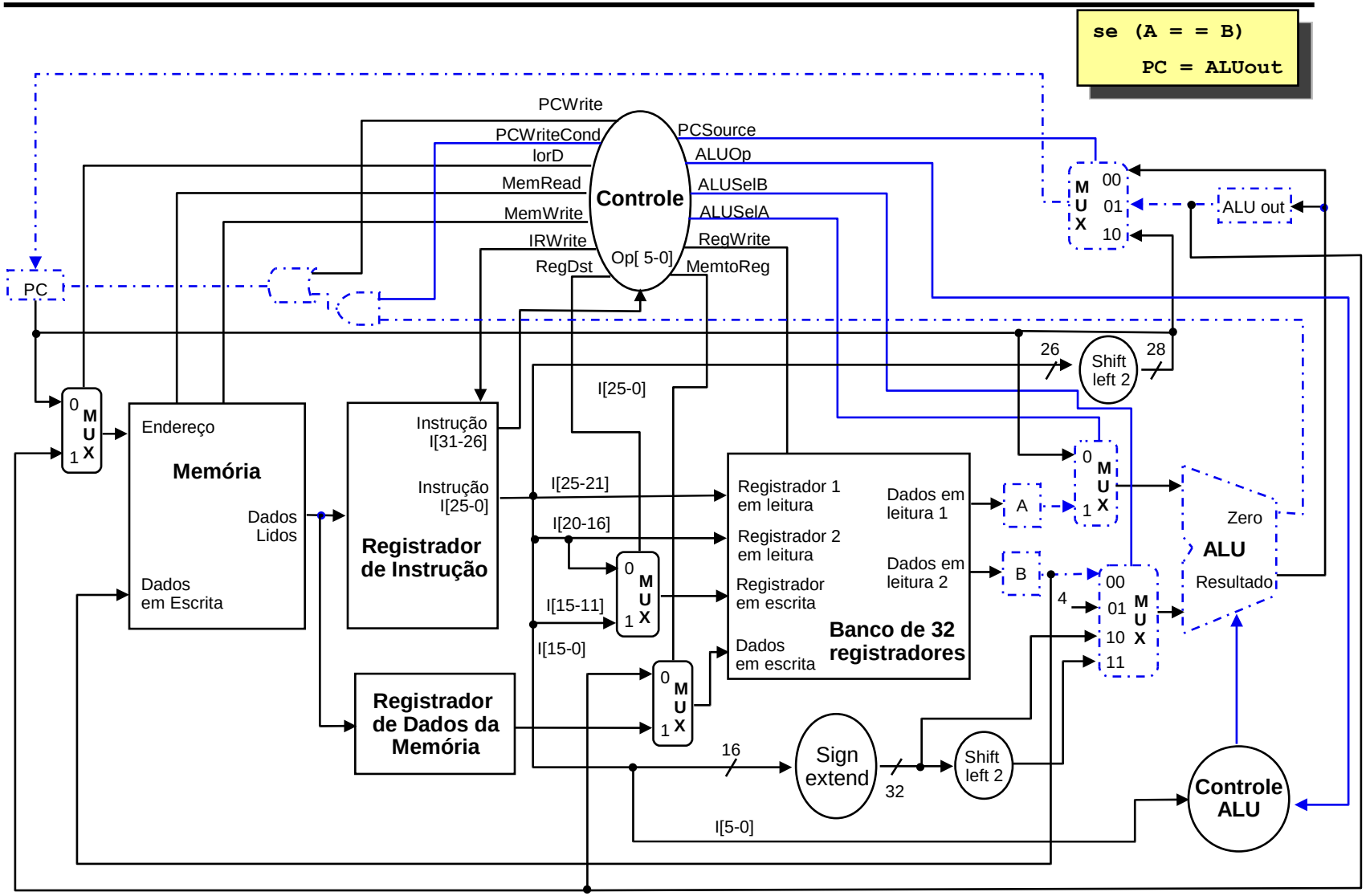
FSM: instrução de desvio condicional (branch)



término do desvio condicional (branch)

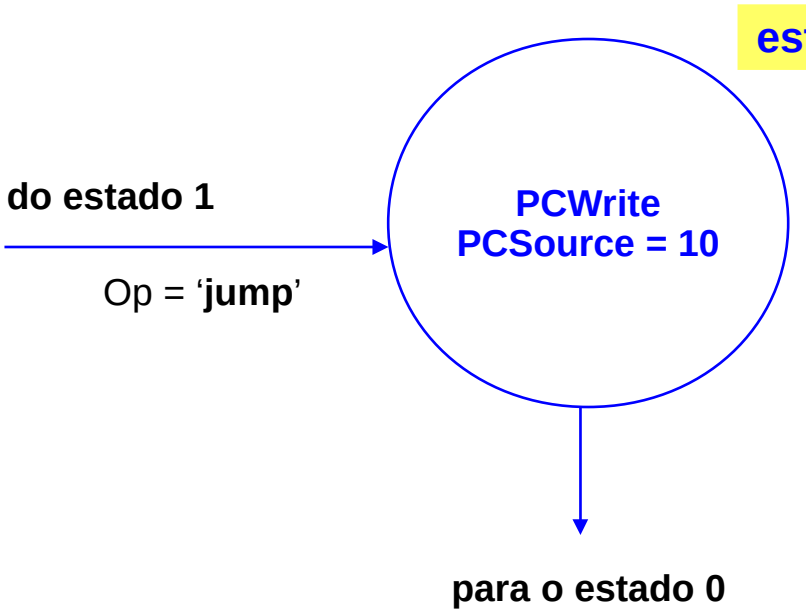
- **ALUSelA = 1:** 1º operando vem do registrador A
- **ALUSelB = 00:** 2º operando vem do registrador B
- **ALUOp = 01:** ALU executa subtração
- **PCWriteCond:** PC escrito se saída Zero = 1
- **PCSource = 01:** PC recebe valor do registrador ALUout

Estado 8 - término do desvio condicional



Estado 9
Instrução de Desvio Incondicional

FSM: instrução de desvio incondicional (jump)

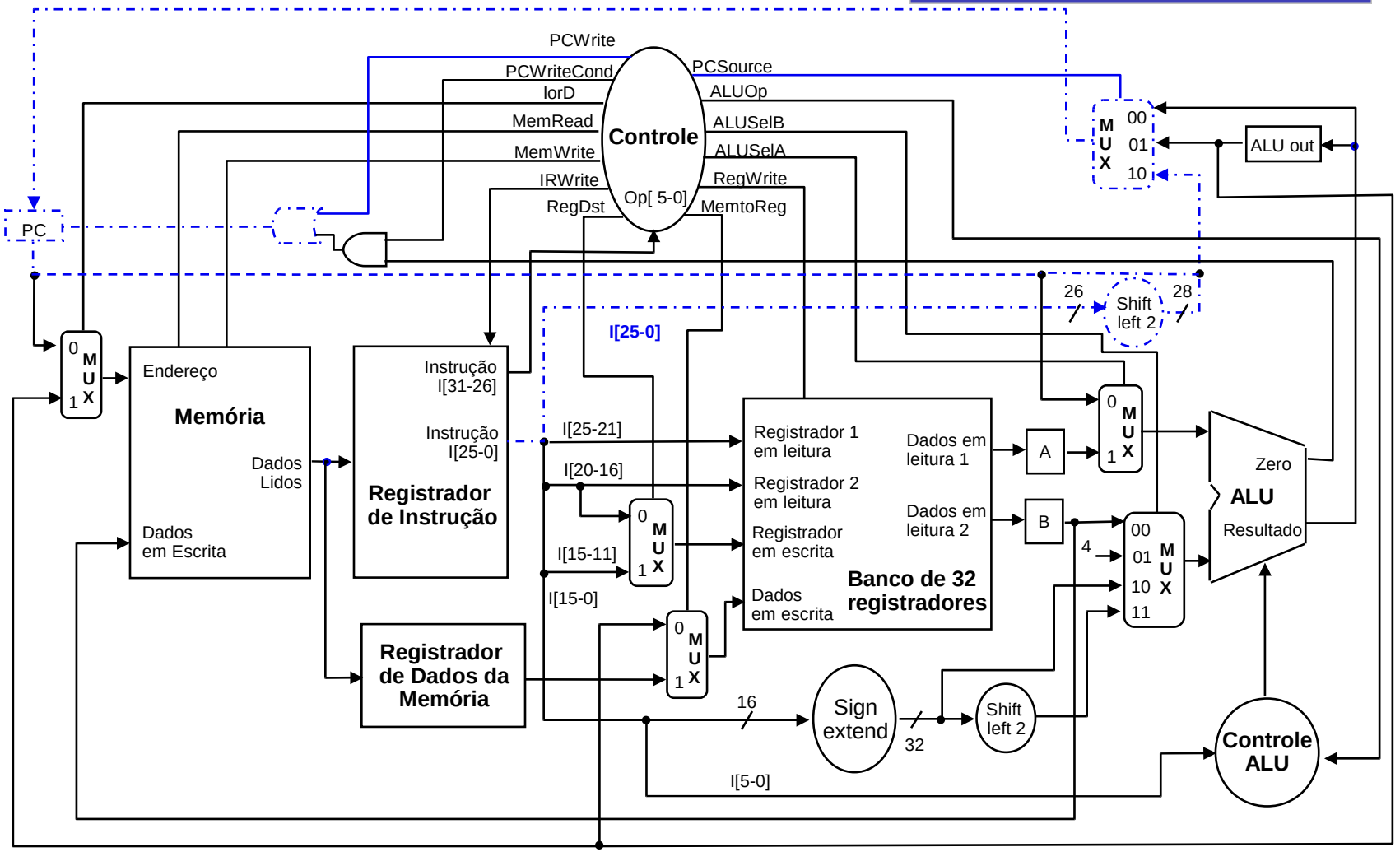


término do desvio incondicional (jump)

- **PCWrite:** **PC escrito**
- **PCSource = 10:** **PC recebe valor do endereço de desvio**

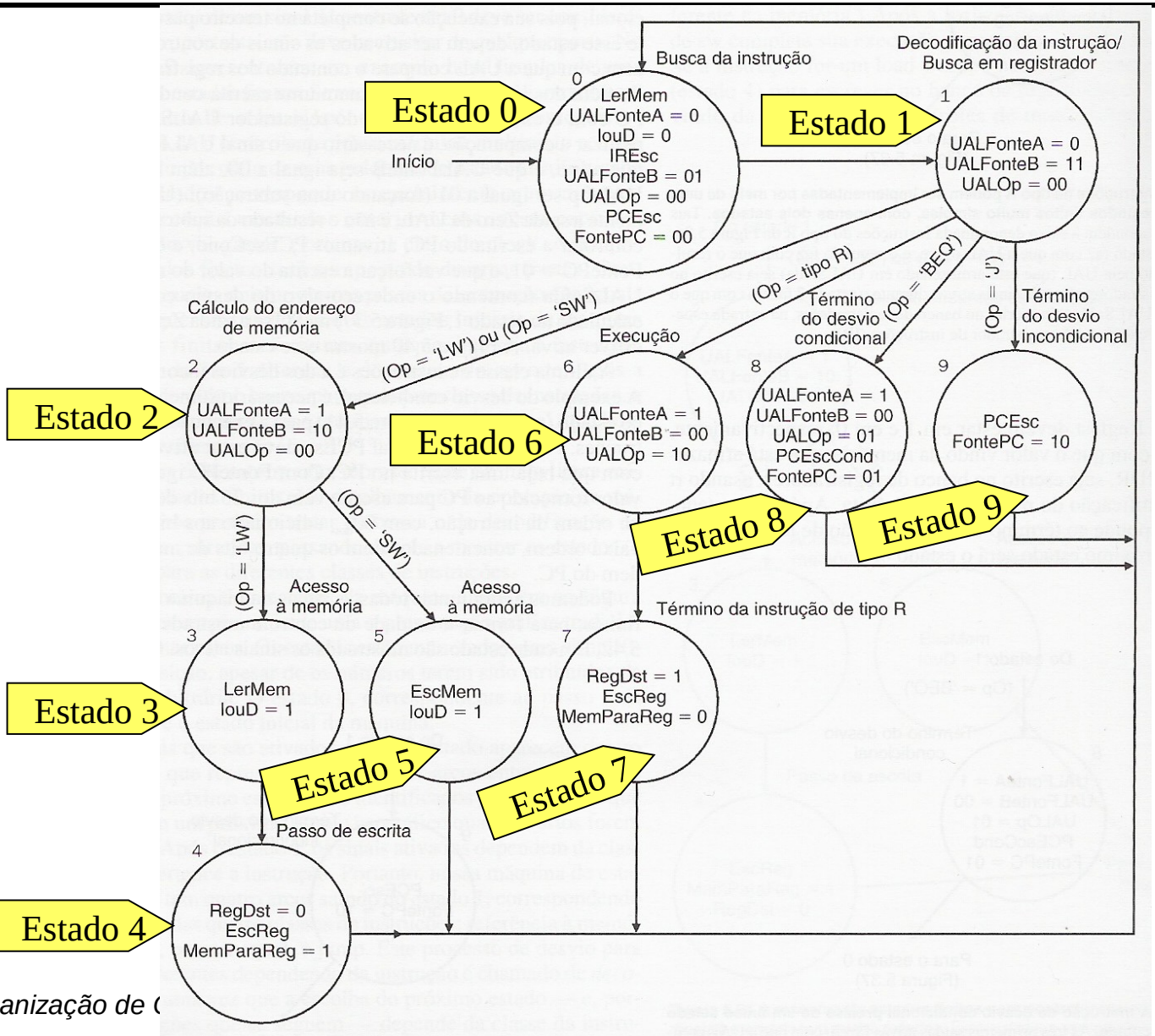
Estado 9 - término do desvio incondicional

```
PC = PC[31-28] || (I[25-0]<<2)
concatenação
```



Máquina de Estados Finitos com todos Estados

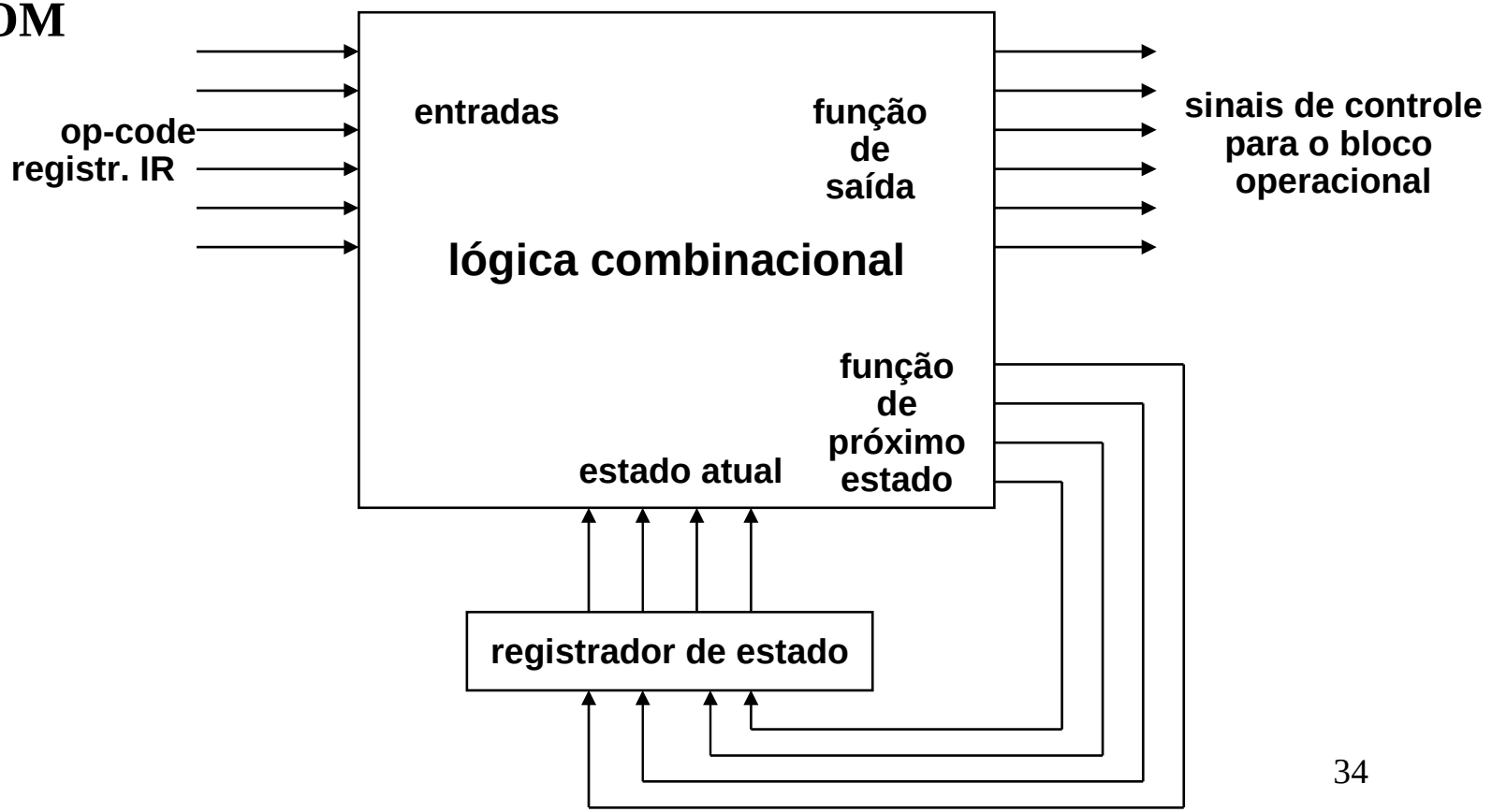
Máquina de Estados Finitos com todos Estados



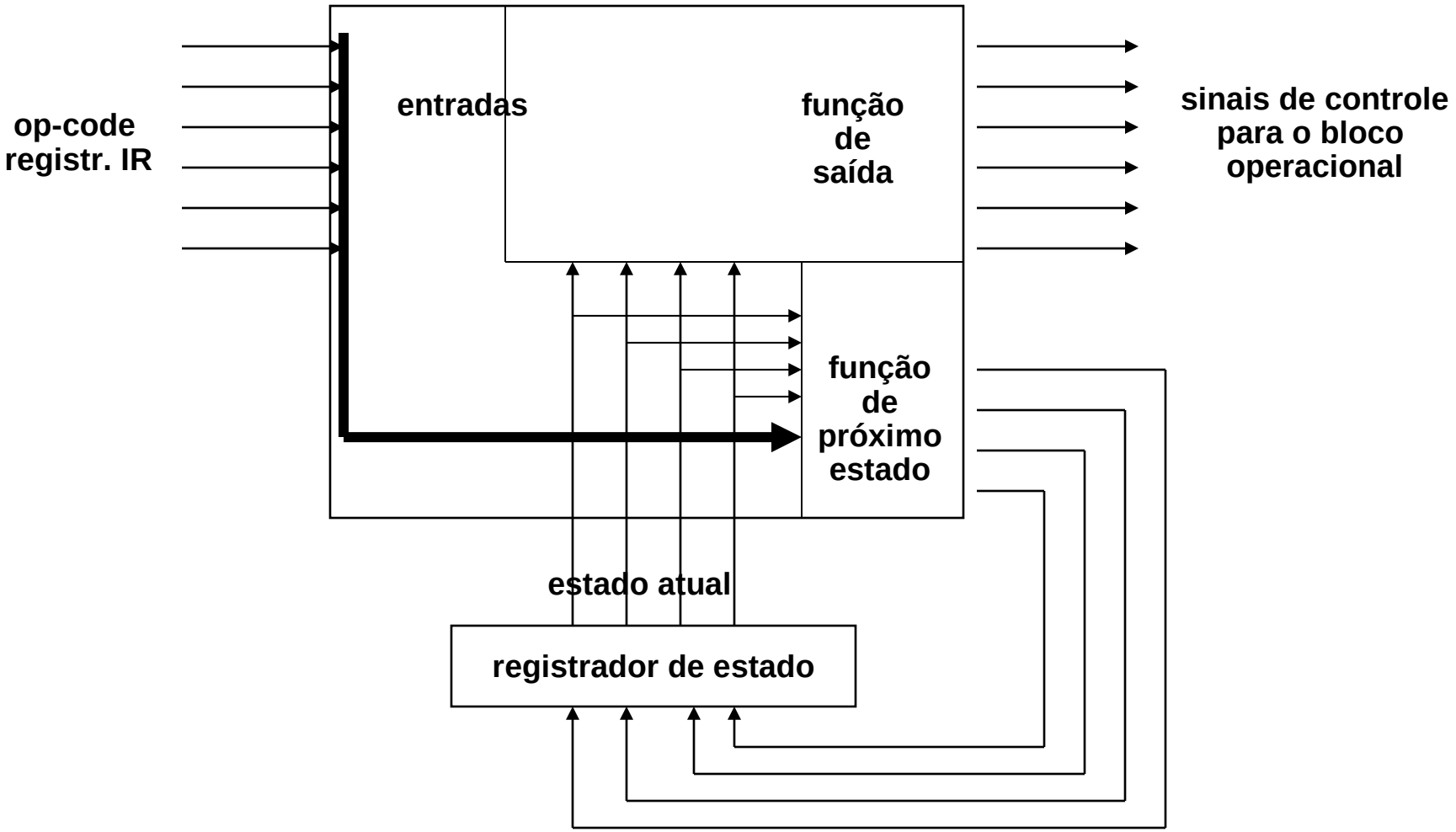
3. Implementando a FSM

A Lógica de Controle pode ser implementada empregando:

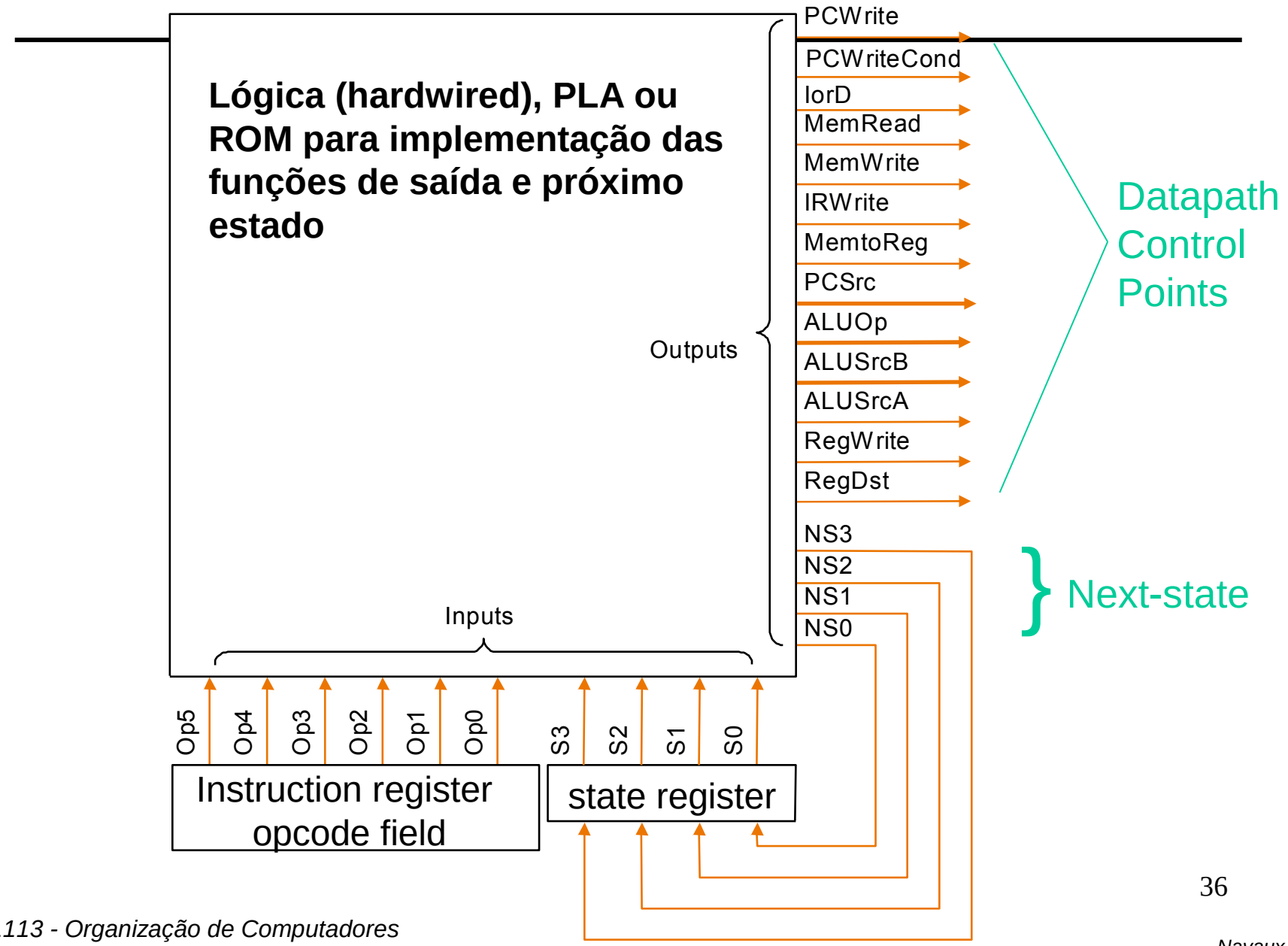
- **Portas Lógicas**
- **PLA**
- **ROM**



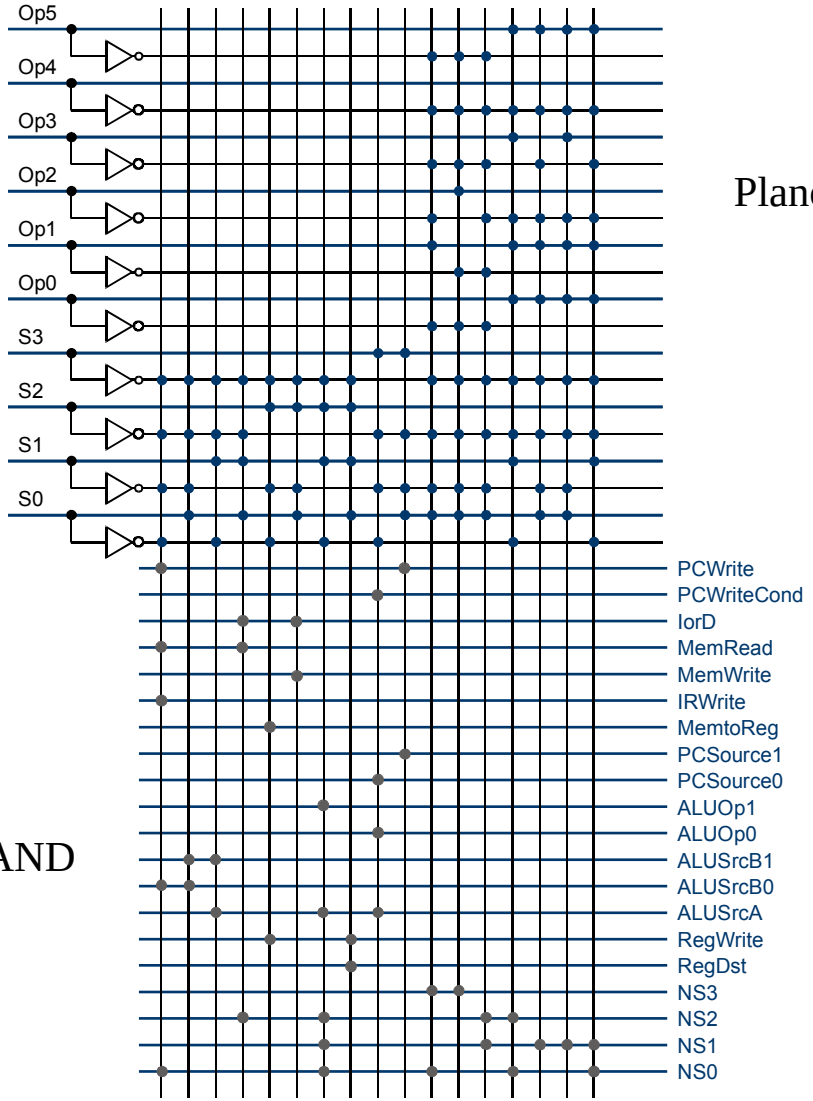
Implementando a FSM



Implementando o Controle da FSM



Implementando com PLA



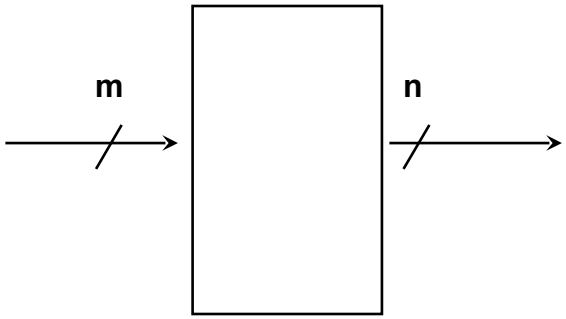
Plano OR

Piano AND

Implemantação com ROM

- **ROM = "Read Only Memory"**
  - **valores das posições de memória são fixados antes do tempo**

- **A ROM pode ser empregada para implementar a tabela de verdades**
  - **se o endereço é m-bits, podemos endereçar 2m entradas na ROM**
  - **as saídas são os bits de dados que apontam para o endereço.**
    m é a "altura", e n é o "largura" igual ao número de saídas.



FIM