Capítulo 5

Parte Operativa e Parte de Controle do MIPS

Adaptado de Bruno Cunha e Carlos Llanos, IESB

Introdução

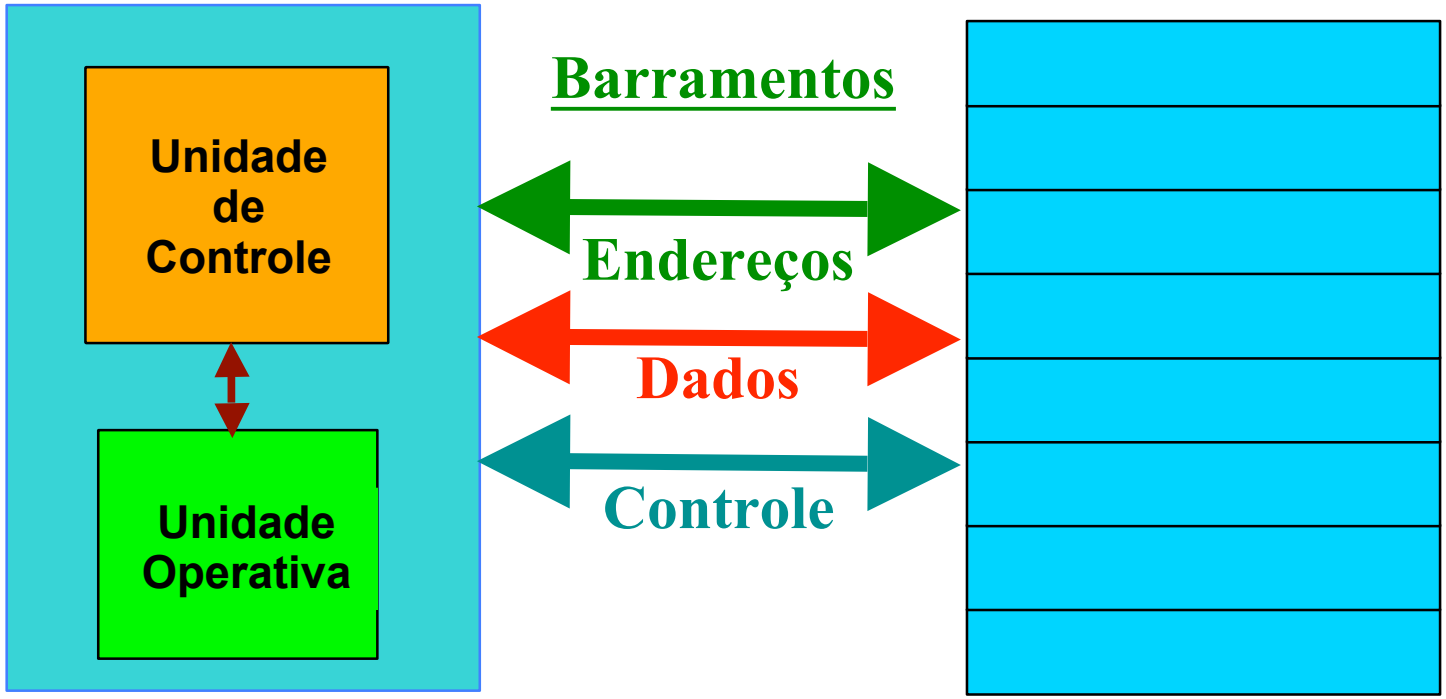
- O desempenho de uma máquina pode ser determinado por três fatores:
  - número de instruções executadas
  - período do clock (ou freqüência)
  - Número de ciclos por instrução (CPI)
- O compilador e a ISA (Instruction Set Arquitecture) determinam a quantidade de instruções a serem executadas por certo programa

Introdução

- Este capitulo aborda a implementação de um subconjunto das instruções do MIPS
- O interrelacionamento entre ISA e a implementação é exemplificado em dois projetos alternativos da parte operativa do processador
  - PO uniciclo
  - PO multiciclo

Arquitetura Von Neumann

- Estrutura básica de um processador



Unidade Operativa

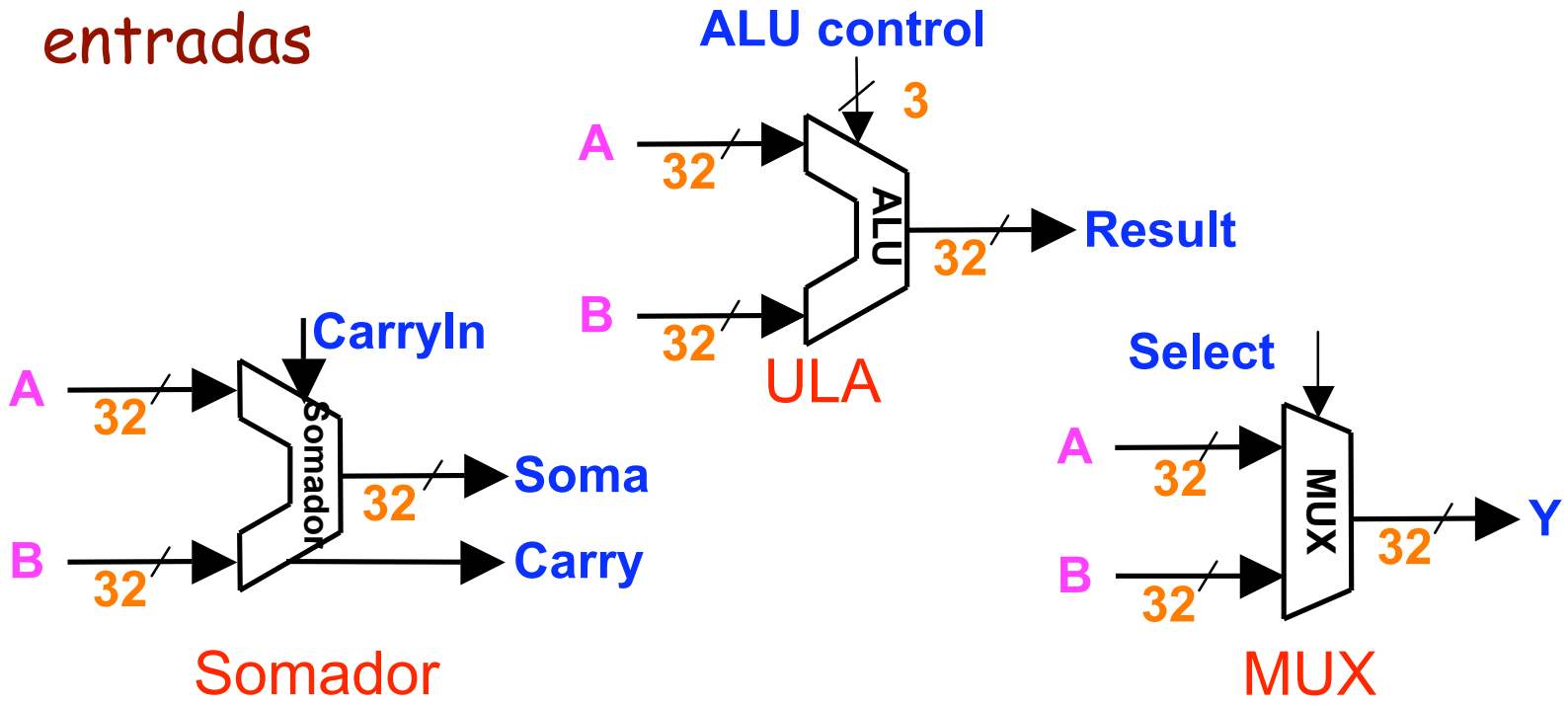
- Tambem chamada "Parte Operativa", "Via de Dados" ou, em inglês, "*Datapath*"
- É construída a partir dos seguintes componentes:
  - elementos de armazenamento (registradores, ffs)
  - operadores lógico-aritméticos
  - recursos de interconexão (barramentos, mux)

Unidade Operativa MIPS

- Será projetada para implementar o seguinte subconjunto de instruções do MIPS:
  - Instruções de referência a memória:
    - *load word* (lw) e *store word* (sw)
  - Instruções Aritméticas e lógicas:
    - *add*, *sub*, *and*, *or* e *slt*
  - Instruções de desvio de fluxo:
    - *equal* (beq), *jump* (j)

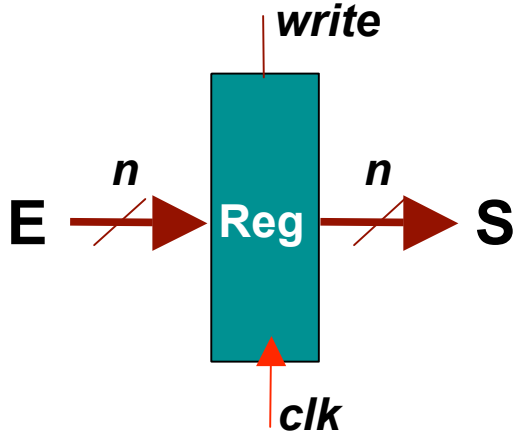
Componentes Combinacionais

- Componentes combinacionais definem o valor de suas saidas apenas em funcao dos valores presentes nas suas entradas



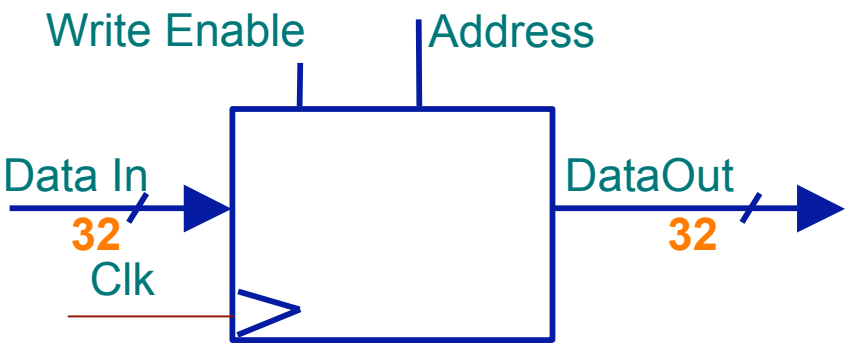
Componentes Sequenciais

- Componentes sequenciais tem um *estado*, que define seu valor em um dado instante de tempo
- Registrador:
  - Um conjunto de flip-flops tipo D (Registrador)
    - Com *n* bits de entrada e saída
    - entrada de habilitação de escrita (*write enable*)
  - Habilitação de escrita (*write enable*):
    - falso (0): O dado de saída não muda
    - verdade (1): o dado de entrada será carregado (saída = entrada)



Memória

- Memória (idealizada)
  - Um barramento de entrada: *Data In*
  - Um barramento de saída: *Data Out*



- Uma palavra é selecionada por:
  - Um endereço seleciona a palavra para ser colocada na saída (*Data out*)
  - Write Enable = 1: Permite que a palavra selecionada seja escrita com a informação na entrada (*Data in*)
- Entrada de Clock (*CLK*)
  - Sincroniza os processos de acesso à memória
  - **Geralmente, os processos são sincronizados pela borda de subida ou de descida do clock**

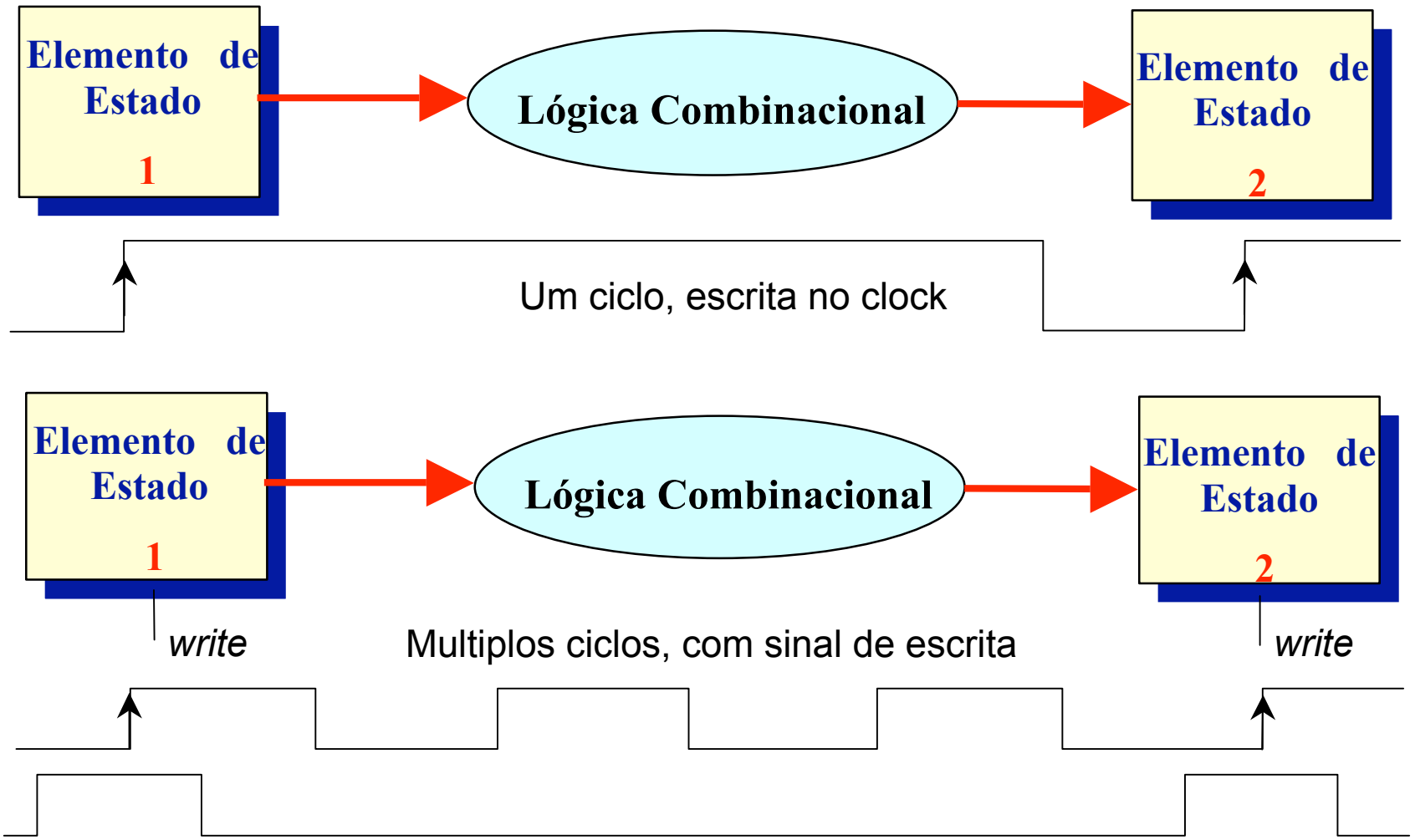
Estrategia de Temporizacao

- Uma metodologia de temporização define quando os sinais podem ser lidos e quando eles podem ser escritos
- É necessário evitar situações de conflito, por exemplo, querer ler uma palavra e escrevê-la simultaneamente
- Será adotada uma metodologia de temporização sensível às transições do sinal do clock
- Nesta metodologia, qualquer valor armazenado nos elementos de estado só pode ser atualizado durante a transição do sinal de relogio (clk)

Estrategia de Temporizacao

- Logica combinacional usualmente tem entradas e saidas conectadas a elementos de estado (sequenciais)
- O elemento de estado 2 so pode ser escrito depois de os dados estarem estaveis
  - atraso de propagacao no elemento de estado 1
  - atraso da logica combinacional
  - tempo de pre-carga (*setup*) no elemento de estado 2

Estrategia de Temporizacao

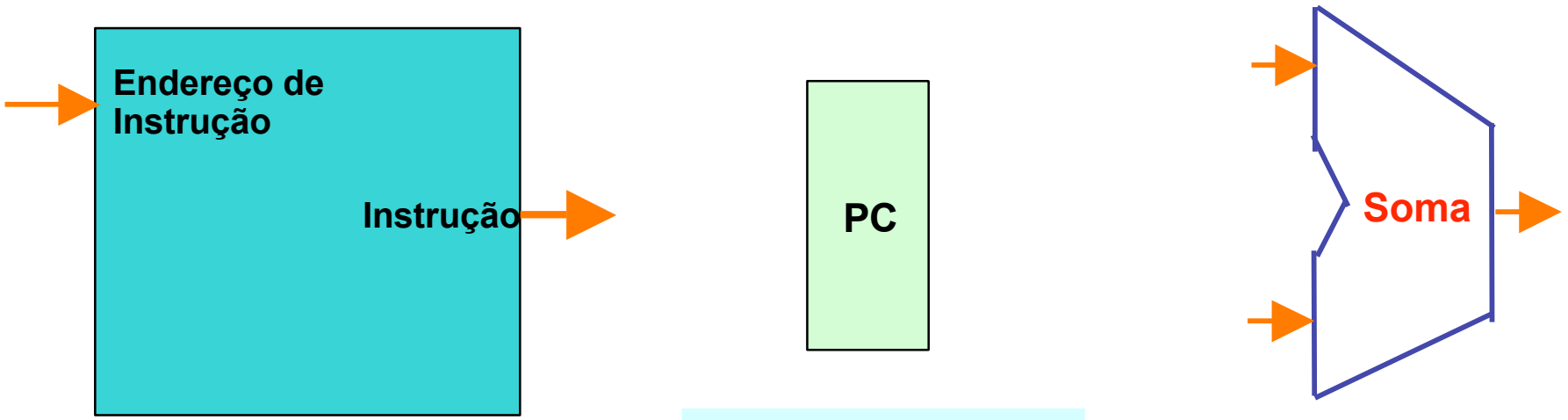


Criando a Unidade Operativa

- **Uma maneira de se começar o projeto de uma unidade operativa (*data path*) é examinar cada um dos componentes necessários para a execução de cada uma das classes de instruções do MIPS**
- Elementos necessários:
  - um lugar para armazenar as instruções (*memória de instruções*)
  - Um registrador para armazenar o endereço de instrução (*Program Counter – PC*)
  - Um contador para incrementar o PC, para compor o endereço da próxima instrução (soma 4 para endereçar próxima instrução)

Criando a Unidade Operativa ...

- Elementos Básicos



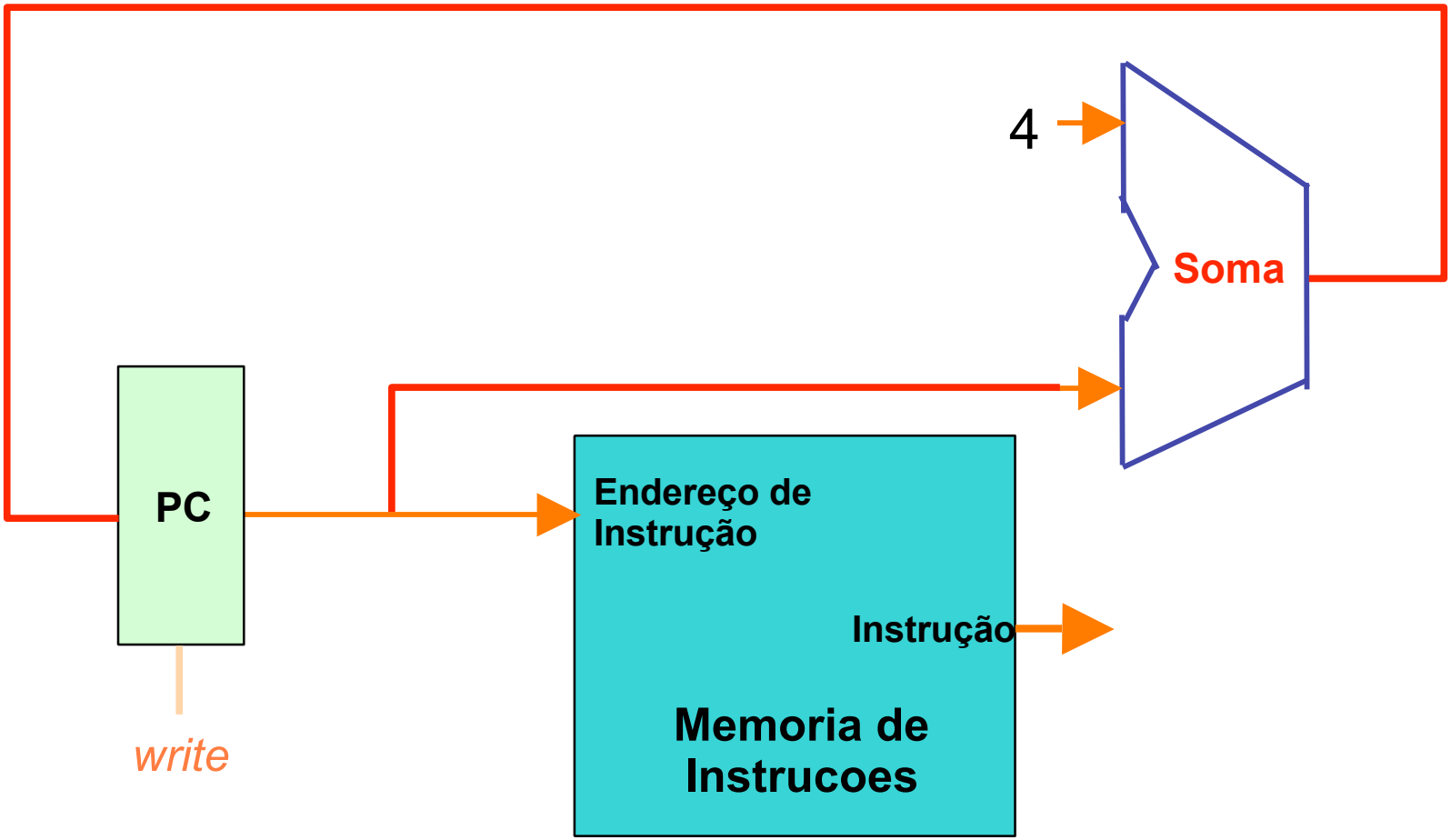
Memória de Instruções

Contador do programa

(*program counter*)

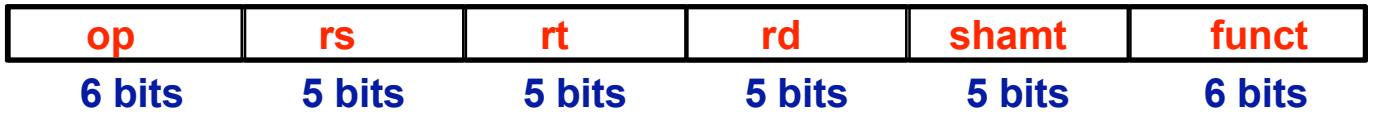
Somador

Busca de Instruções



Instruções Lógico-Aritméticas

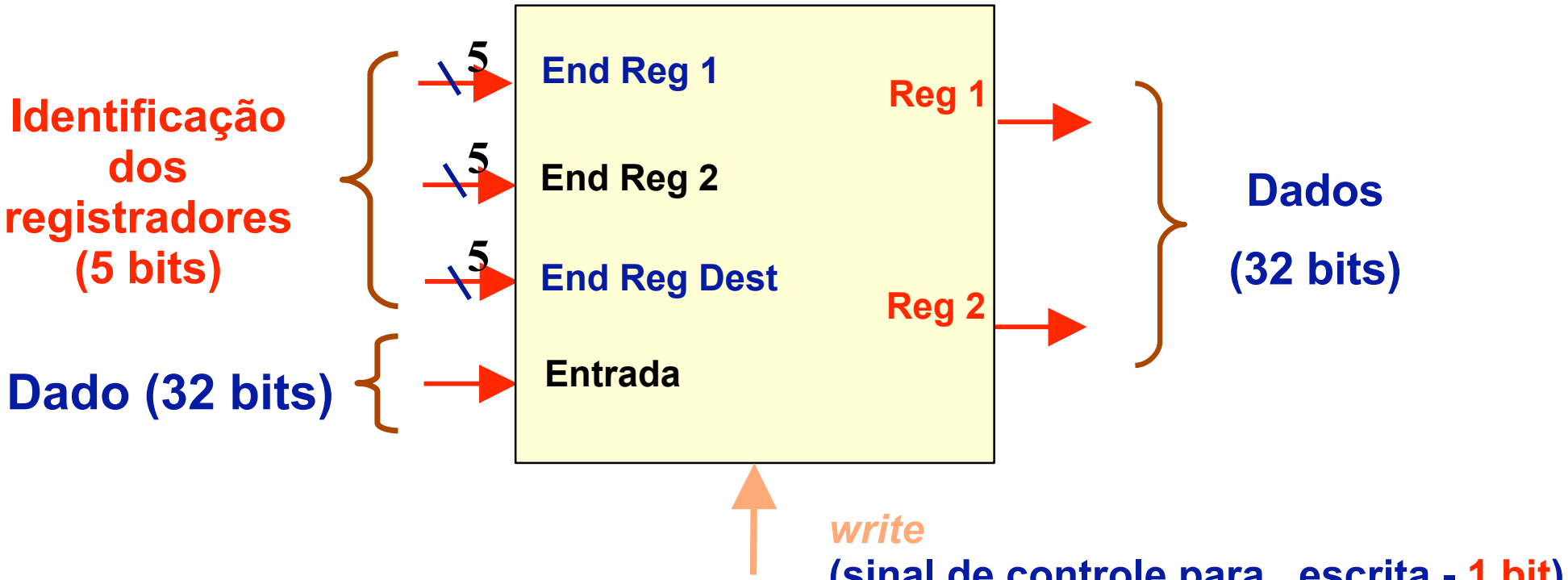
- Formato de uma instrução tipo R no MIPS:



- Semântica:
  - $rd <- op($rs, $rt)
- Estrutura de suporte: banco de registradores

Banco de Registradores

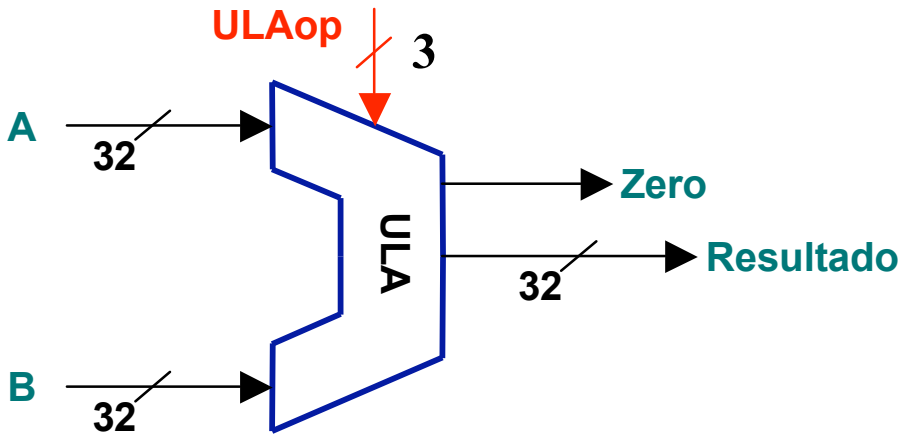
- Dupla porta: leitura de dois registradores ao mesmo tempo
- Sinal de controle para escrita - leitura não necessita controle



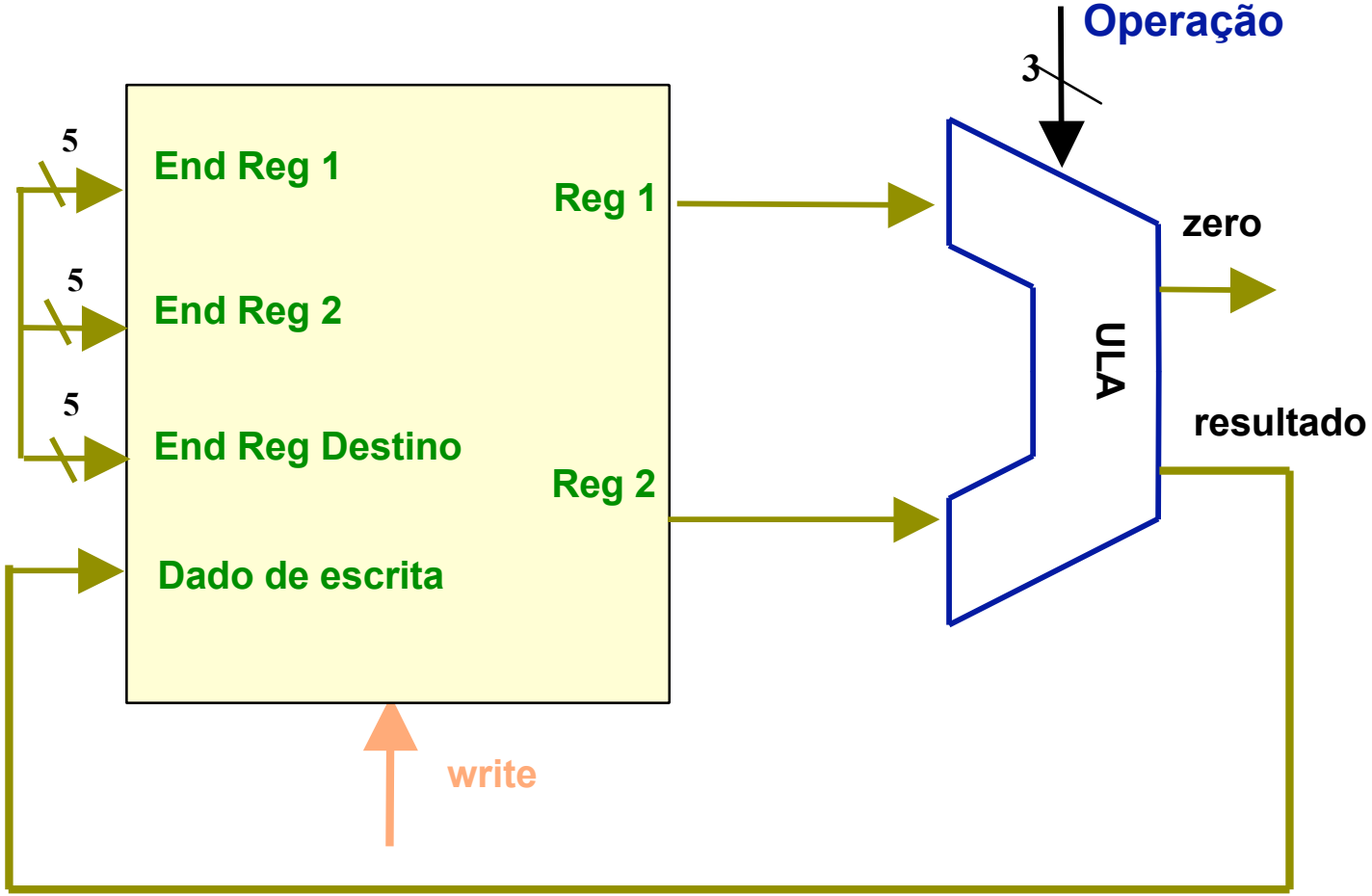
Projeto da ULA

- A ULA foi desenvolvida no capítulo anterior
- 3 bits de controle indicam a operação a ser realizada

ULAop	Função
000	and
001	or
010	add
110	sub
111	slt



Instruções Tipo R – unid. operativa



Instruções de Acesso a Memória

- Formato da instrução tipo I (lw/sw):

op	rs	rt	endereço
6 bits	5 bits	5 bits	16 bits

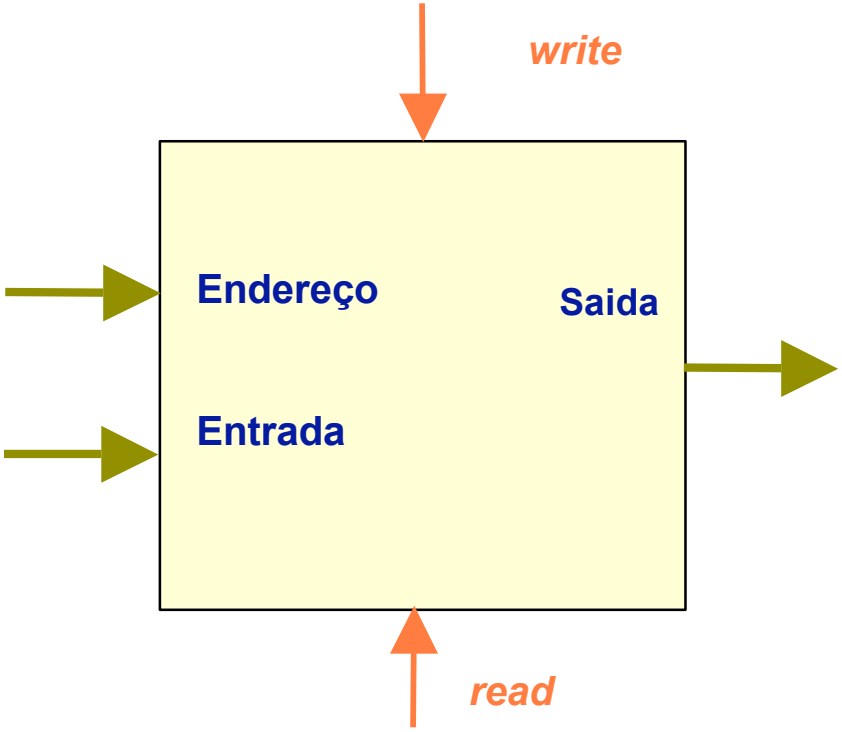
- Ex:
  - lw $8, 32($19)

35	19	8	32

  - end = $19 + 32

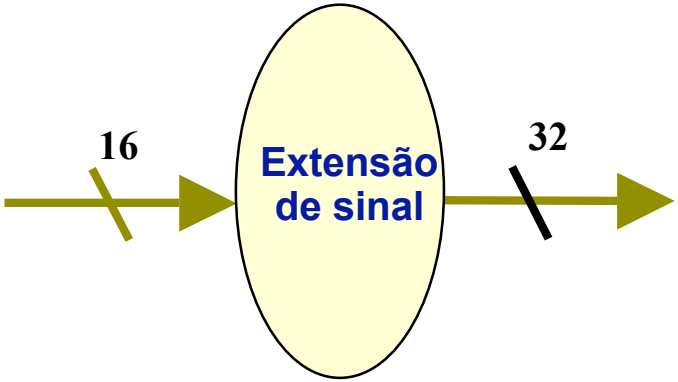
Memória

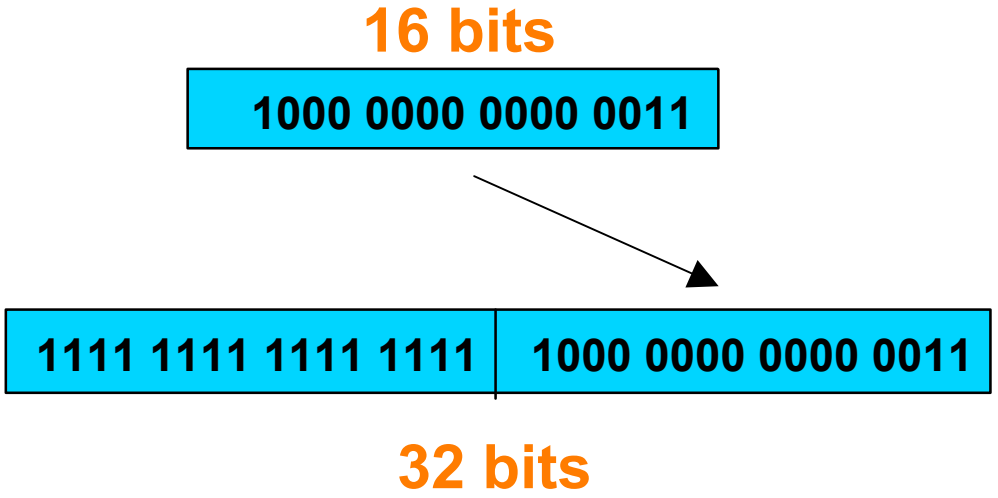
- Memória com um barramento de entrada independente do de saída
- Controle de escrita (*write*) e leitura (*read*)
- Barramento de endereços
- Um acesso de cada vez



Extensão de Sinal do Deslocamento

- Deslocamento na instrução deve ser extendido de 16 para 32 bits, mantendo-se o sinal
  - se for negativo, 16 bits superiores = 1
  - se for positivo, 16 bits superiores = 0

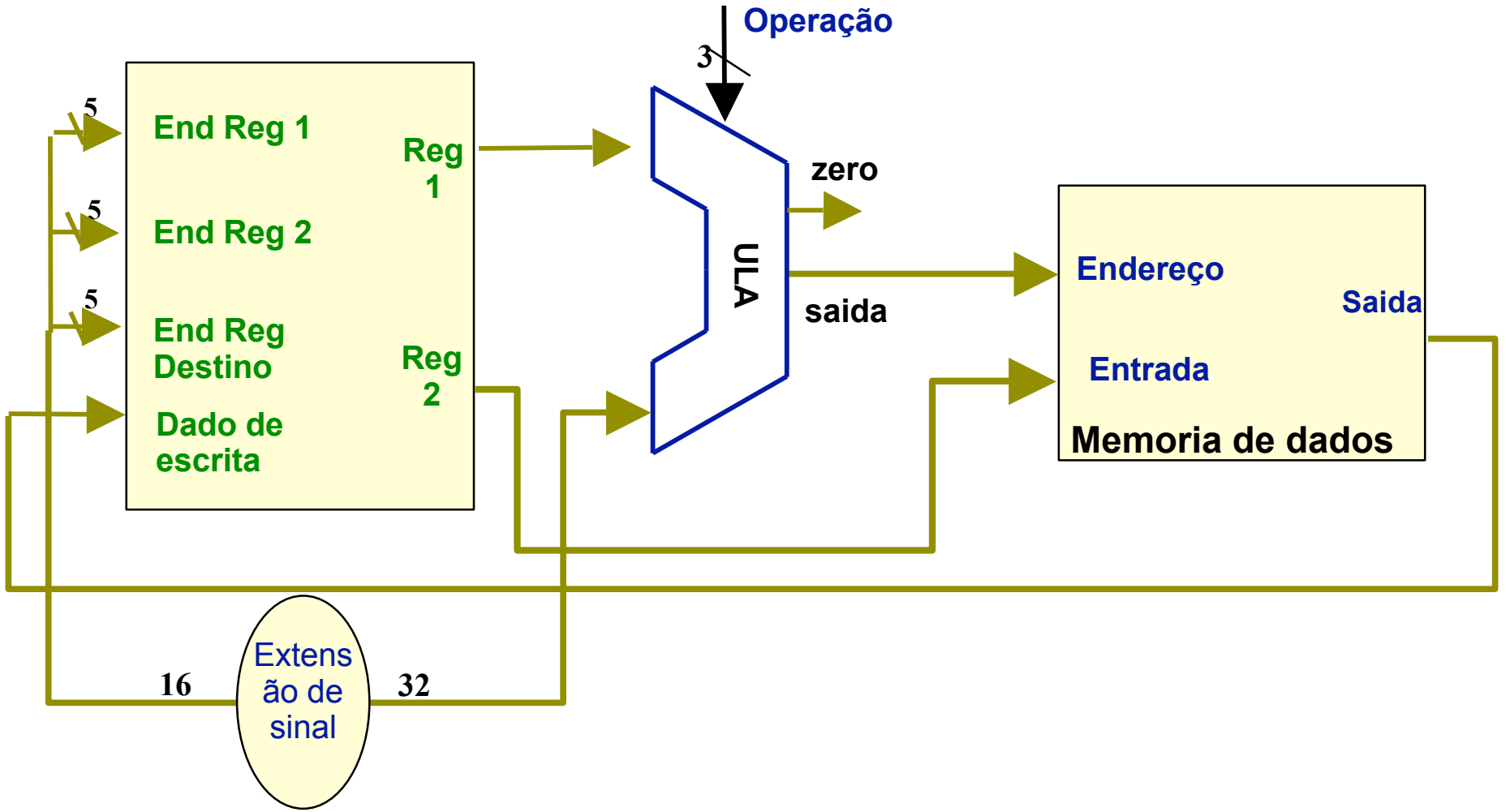




Acesso a Memória

- lw lê um dado da memória e escreve em um registrador
  - conexão entre a saída da memória e a entrada do banco de registradores
- sw lê um dado de um registrador e escreve na memória
  - ligação entre saida do banco de registradores e entrada de dados da memória
- endereço calculado através da ULA
  - saída da ULA ligada ao barramento de endereços da memória

Unidade Operativa lw/sw



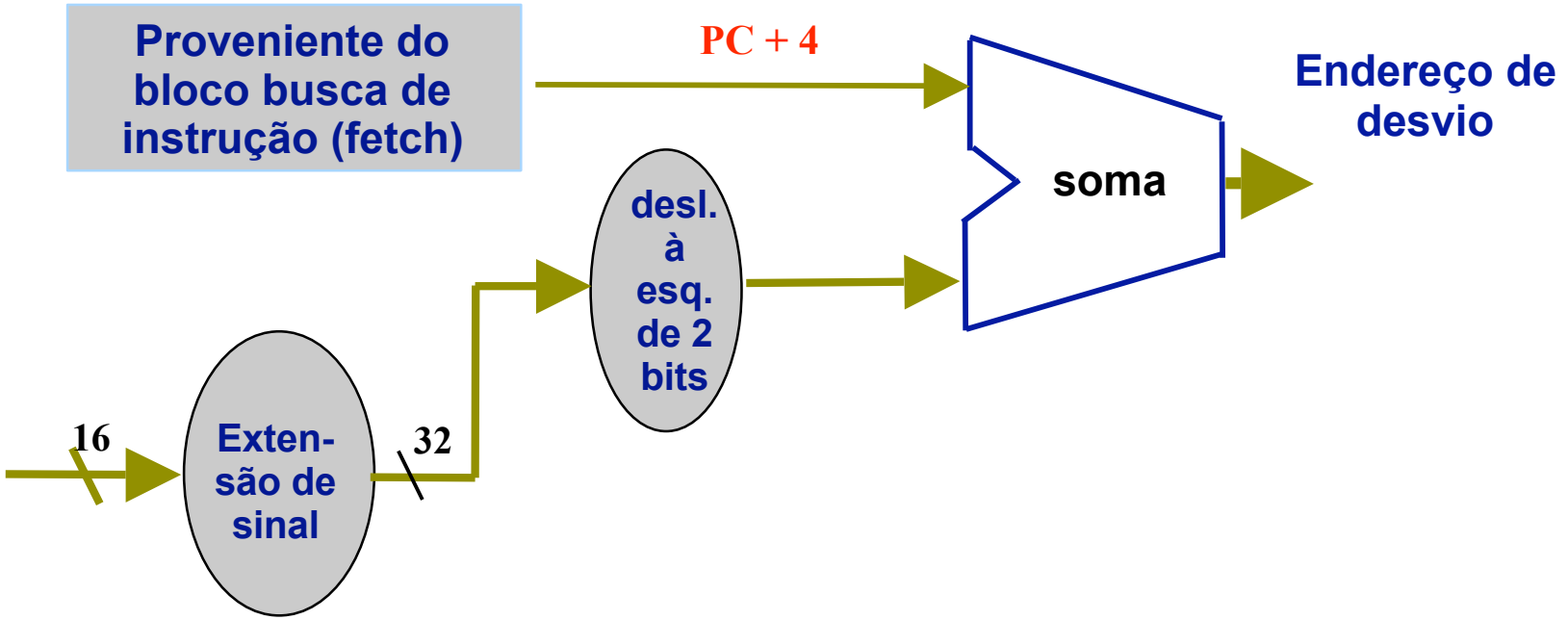
Instruções de Desvio

- beq $1, $2, desloc
  - compara dois registradores
  - soma desloc a PC+4 se $1 = $2
    - PC + 4 e o endereço da próxima instrução
  - no montador utiliza-se uma versão simplificada, com o rótulo do destino
    - beq $1, $2, Rótulo
    - neste caso, o montador calcula o deslocamento
  - desloc é um deslocamento de palavras, ou seja, cada unidade de desloc corresponde a 4 bytes

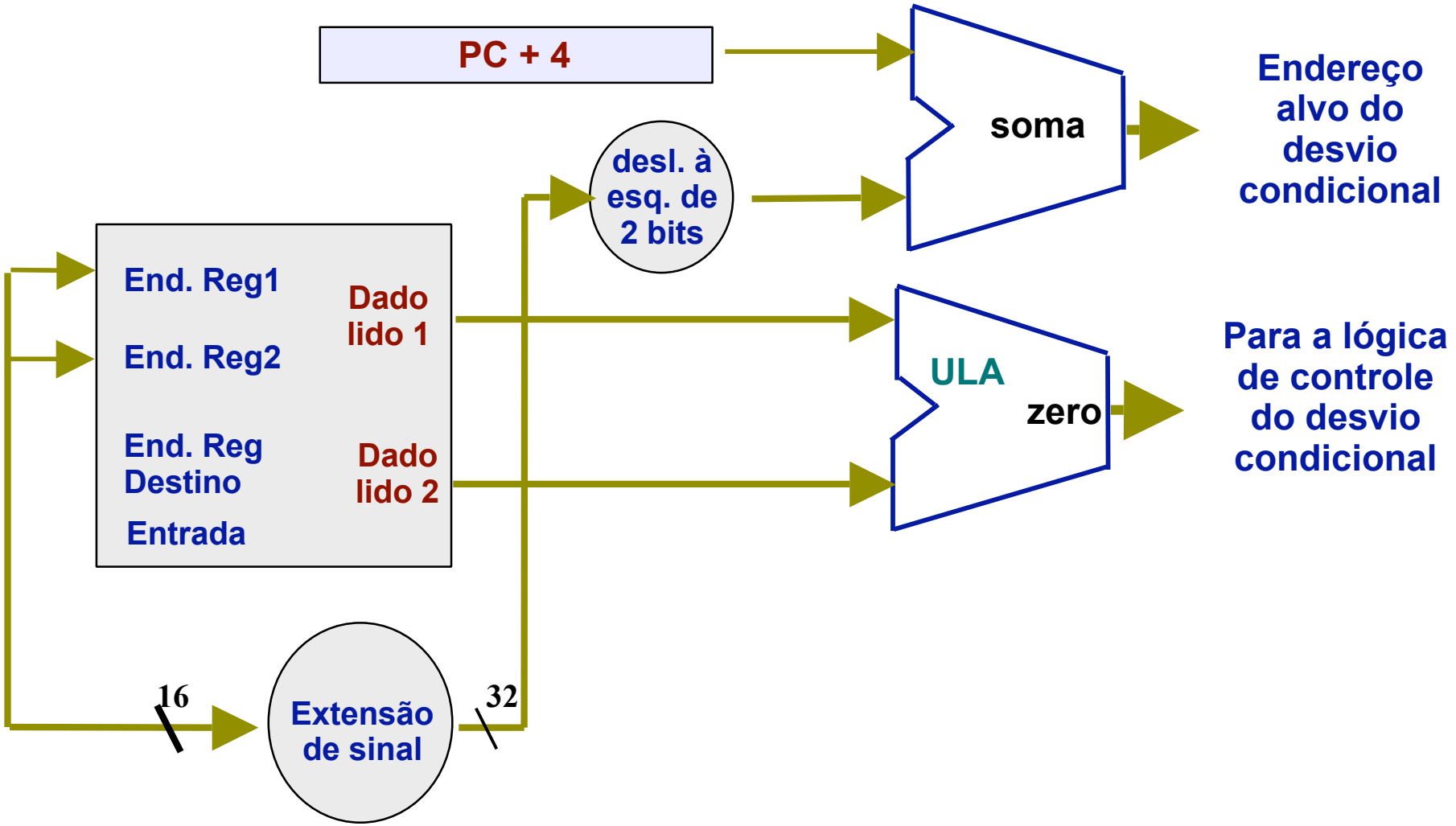
Instruções de Desvio

- A comparação entre os registradores é feita subtraindo-os na ULA e verificando se o resultado é zero
- O PC deve ser carregado com PC + 4 ou PC + 4 + desloc*4, de acordo com o resultado do teste
- A multiplicação de *desloc* por 4 é feita deslocando-se de 2 bits o seu valor

Cálculo do Endereço de Desvio



Circuito Cálculo Endereço Desvio



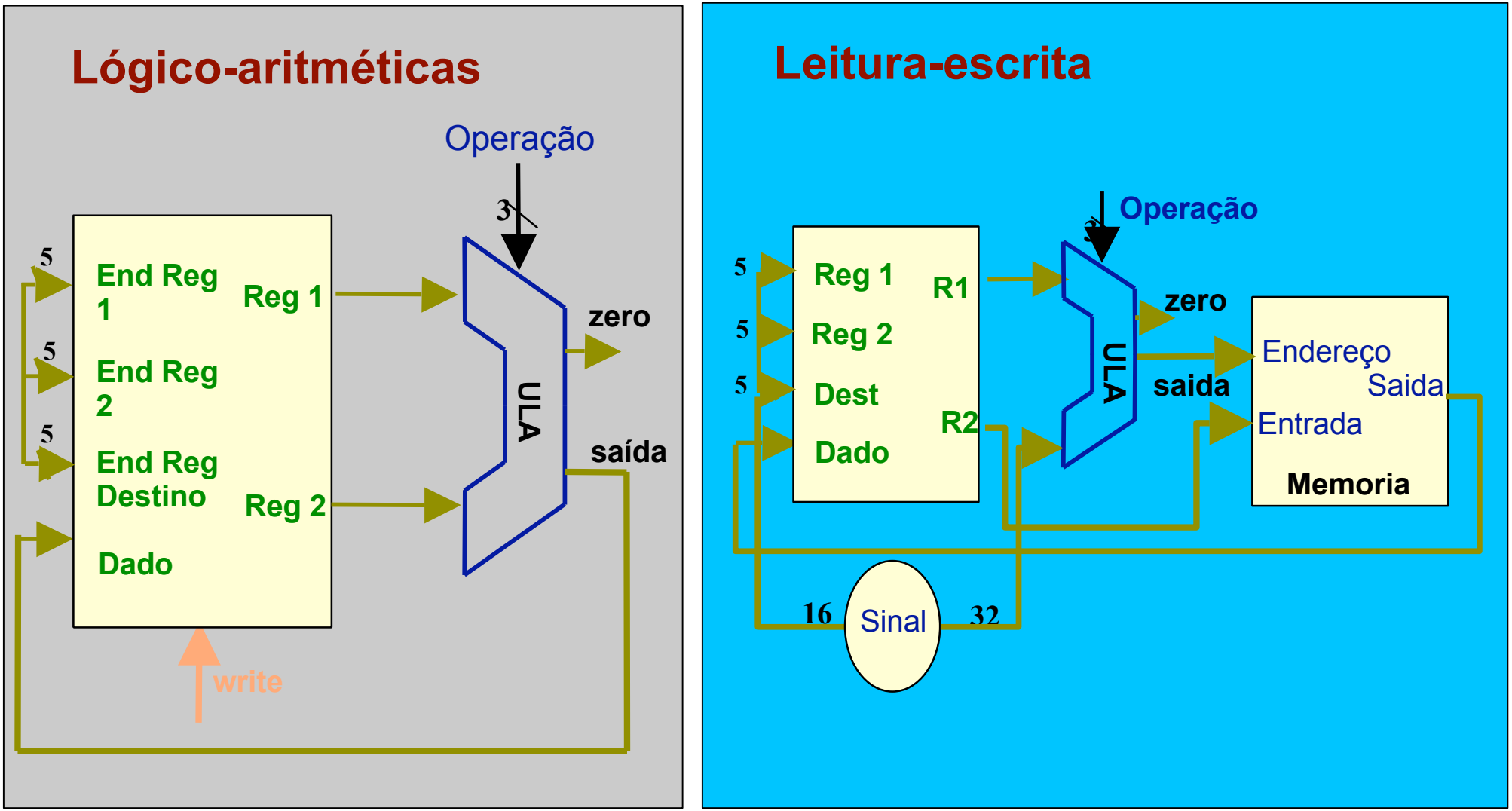
Observações

- Para realizar a comparação dos dois operandos precisamos usar o banco de registradores (os dois operandos estão lá)
- O cálculo do endereço de desvio foi incluído no circuito (já estudado)
- Na instrução não é preciso escrever no banco de registradores
- A comparação será feita pela ULA, subtraindo-se os registradores e utilizando a saída *zero* para verificar a igualdade

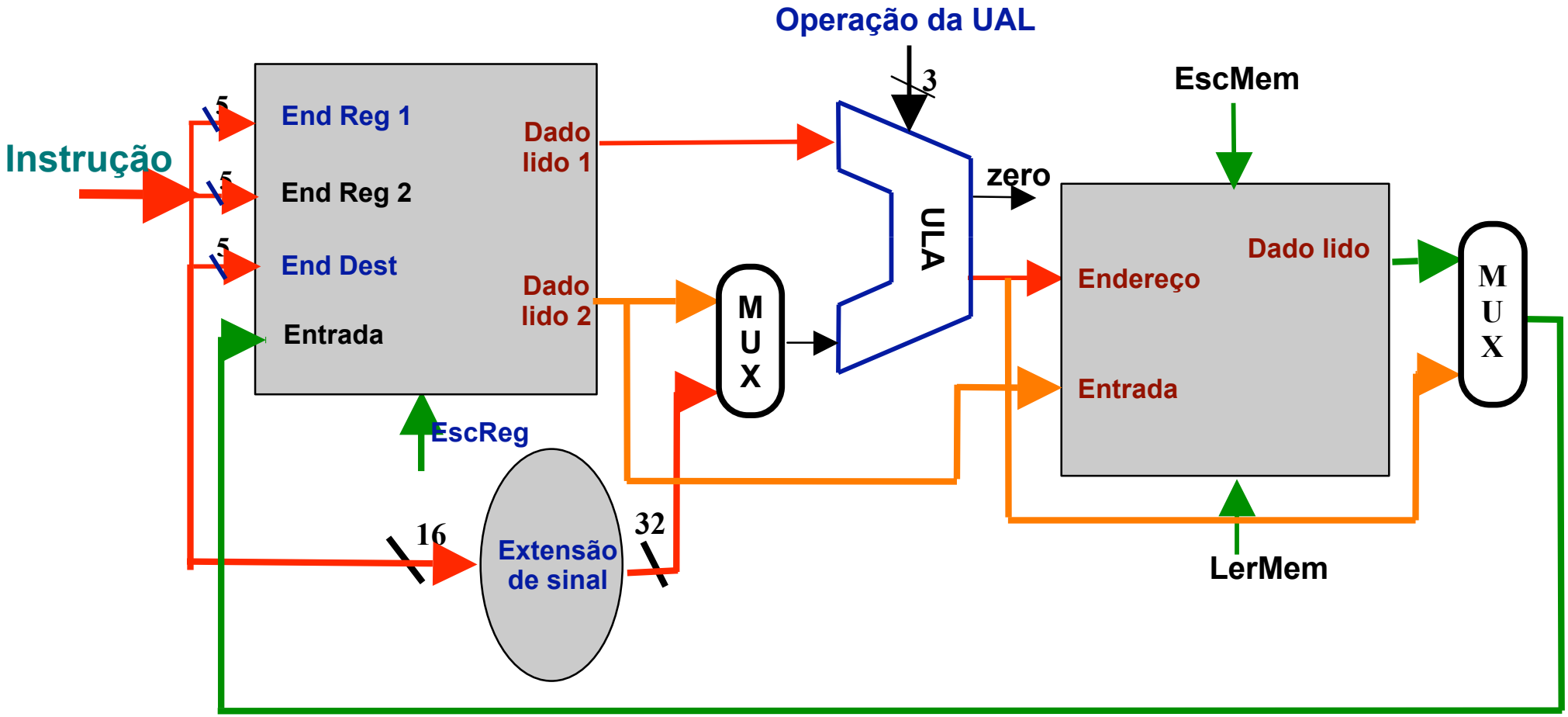
MIPS Uniciclo

- Foram desenvolvidas, na verdade, três tipos de unidades operativas:
  - uma para instruções no formato R (*add*, *sub*, etc.)
  - uma para instruções de no formato I (*load* e *store)*
  - uma para *instruções condicionais (formato I)*
- Na fase de projeto as vezes precisamos replicar recursos
- A via de dados mais simples deve-se propor executar as instruções num único período do clock
- Isto quer dizer que nenhum dos recursos pode ser usado mais de uma vez por instrução

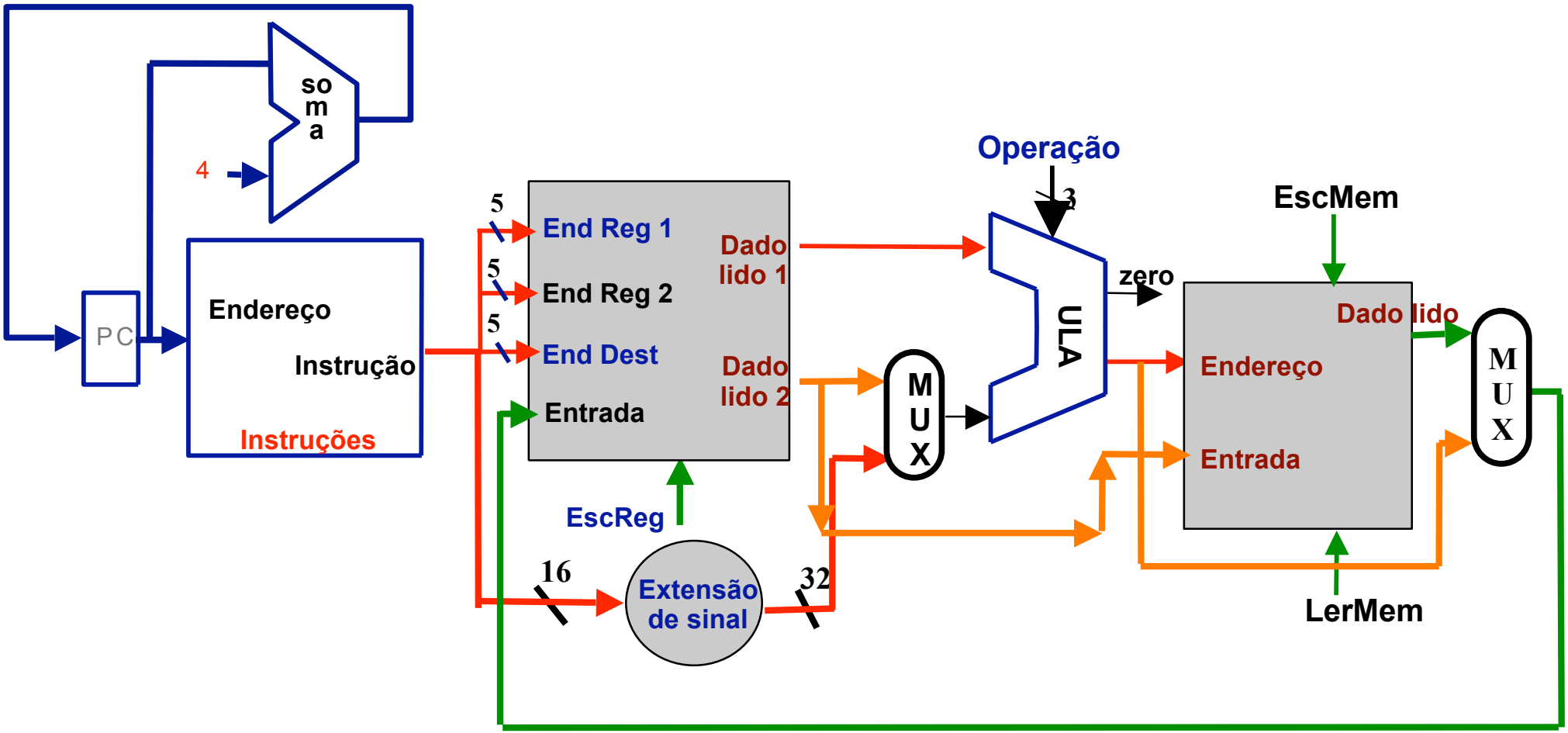
Instruções Lóg/Arit e lw/sw



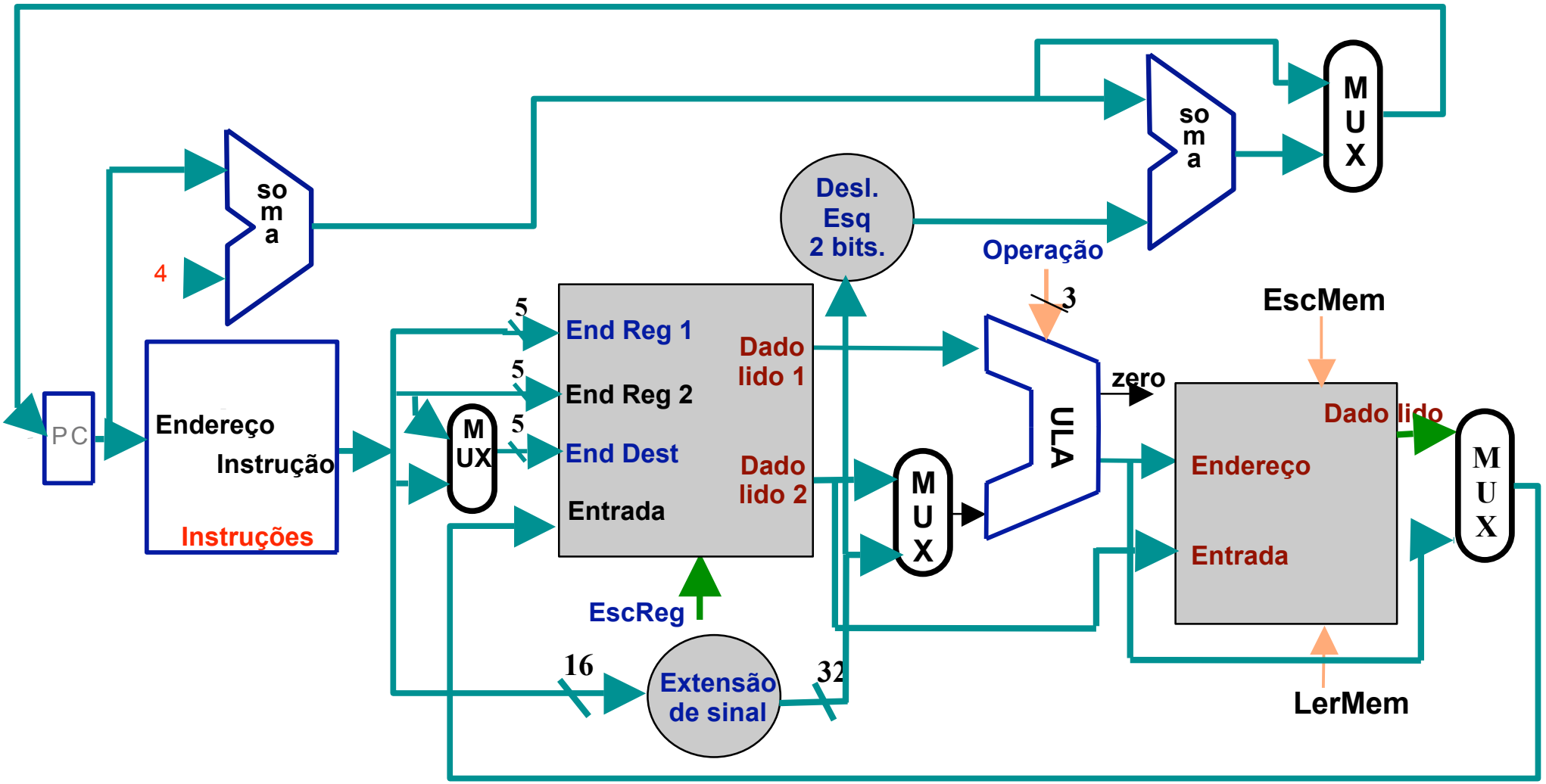
Combinando as Unidades



Acrescentando a Busca



Final


soma
4
soma
MUX
Desl.
Esq
2 bits.
Operação
3
EscMem
PC
Endereço
Instrução
Instruções
MUX
5
5
5
End Reg 1
End Reg 2
End Dest
Entrada
Dado
lido 1
Dado
lido 2
MUX
ULA
zero
Endereço
Entrada
Dado lido
MUX
EscReg
16
Extensão
de sinal
32
LerMem

Controle do MIPS

- Cada operação requer a utilização adequada dos recursos do MIPS
- Ex: *add $t2, $s1, $s2*
  - é necessário que os endereços dos operandos $s1 e $s2 sejam enviados ao banco de registradores
  - Da mesma maneira, o endereço do registrador destino ($t2) deverá ser informando ao banco de registradores
  - Uma vez que o banco de registradores disponibilize os valores de $s1 e $s2, estes deverão ser encaminhados à ULA
  - Posteriormente, o resultado deverá ser escrito no banco de registradores (registrador $t1)

Controle da ULA

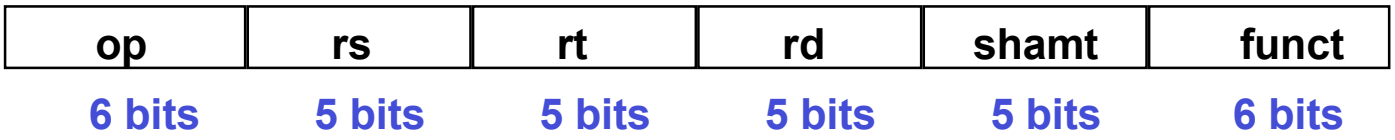
- As operações da ULA são controladas por um código de 3 bits:

ULAop	Função
000	and
001	or
010	add
110	sub
111	slt

- As instruções lw e sw utilizam a operação de soma da ULA
- As instruções do tipo R utilizam uma das 5 operações

Identificação da Operação

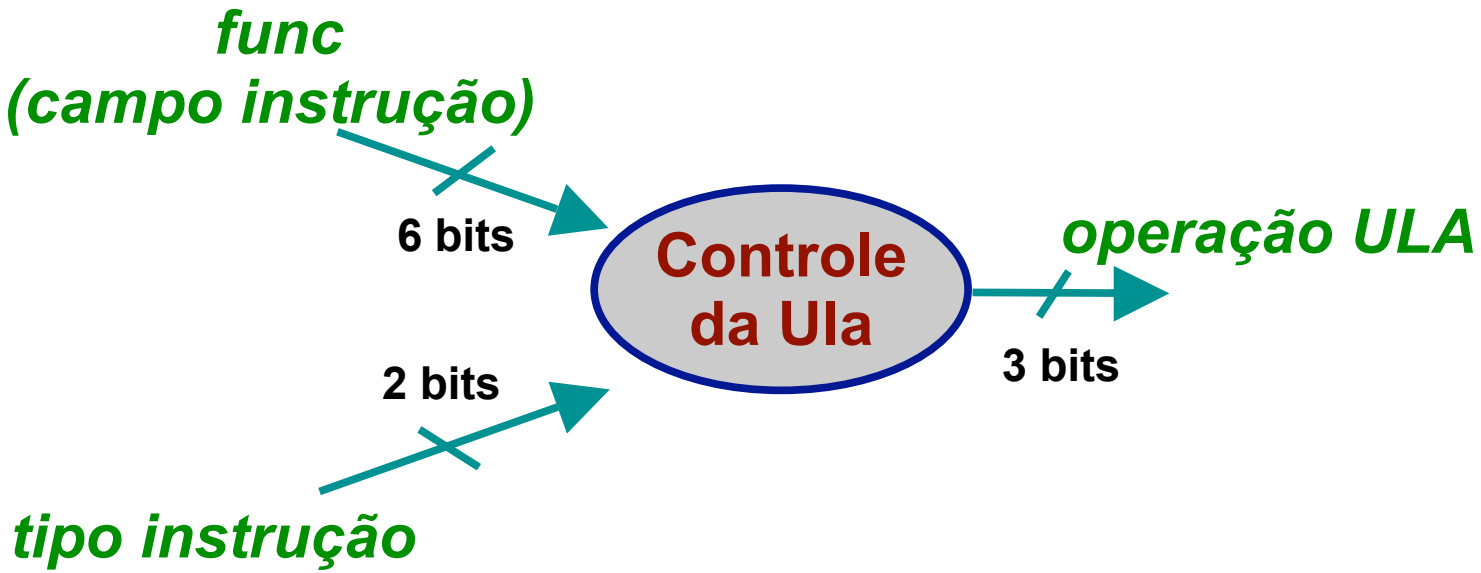
- O campo *funct*, de 6 bits (no formato R) indica que tipo de operação aritmética/lógica será executada:



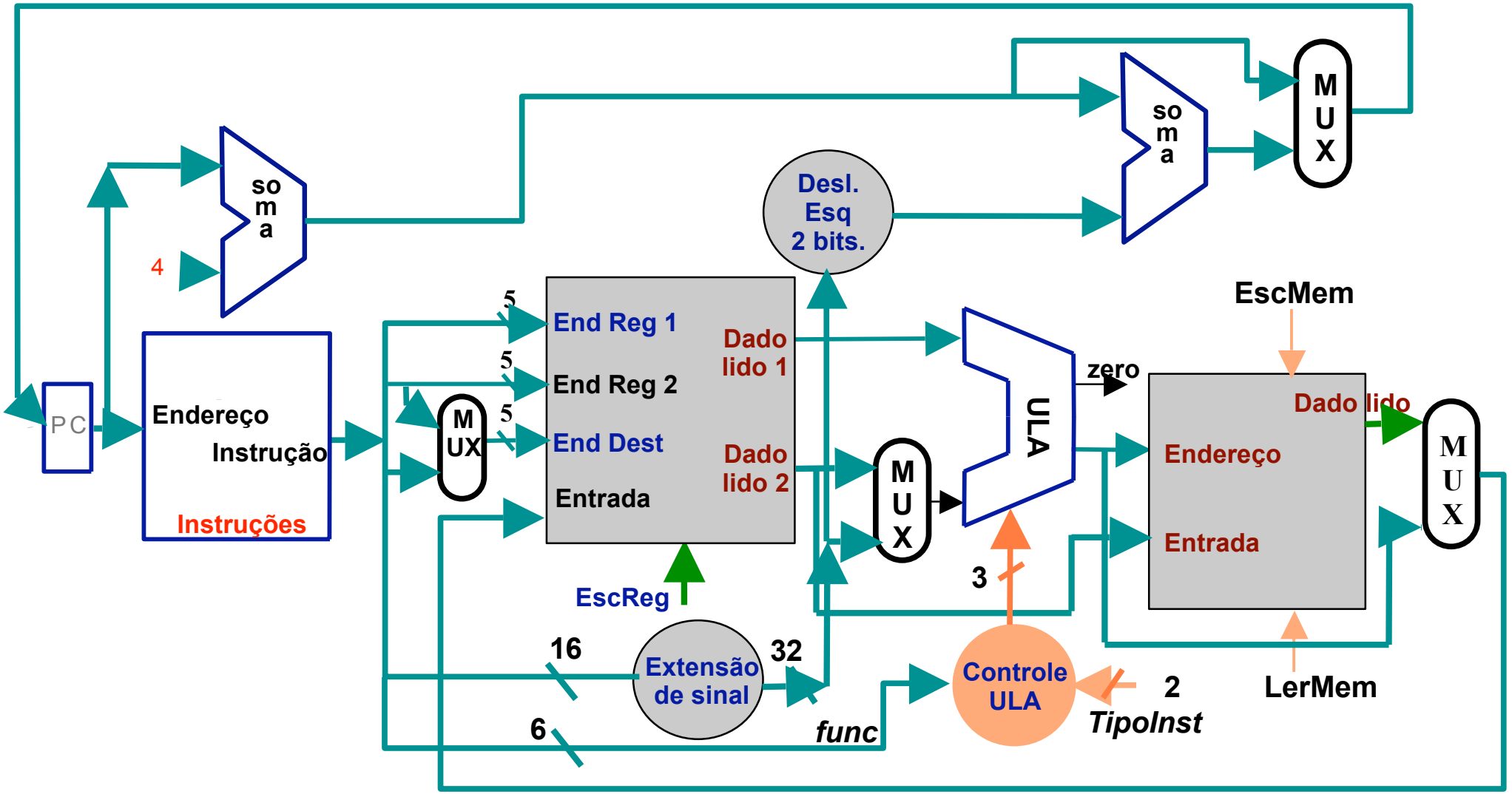
- Pode-se ainda utilizar 2 bits para identificar uma das instruções:
  - 00 acesso à memória (lw, sw)
  - 01 desvio (beq, bne)
  - 10 lógico-aritméticas (add, sub, ...)

Controle da ULA

- Lógica da unidade de controle encarregada de gerar os bits que determinam a operação da ULA



Lógica de Controle da ULA



Circuito de Controle

- Projeto do Circuito de controle da ULA
  - xxx indica irrelevâncias (don't cares)

Código de operação da instrução	ULAop	Operação da Instrução	Campo da Função	Operação desejada da ULA	Entrada de controle da ULA
LW	00	load word	xxxxxx	soma	010
SW	00	store word	xxxxxx	soma	010
Beq	01	branch equal	xxxxxx	subtração	110
Tipo R	10	add	100000	soma	010
Tipo R	10	subtract	100010	subtração	110
Tipo R	10	and	100100	and	000
Tipo R	10	or	100101	or	001
Tipo R	10	set less than	101010	set less than	111

Tabela Verdade Controle ULA

- Considerando apenas os códigos que definem a operação:

ULAop		Campo da Função						Operação da ULA
ULAop1	ULAop2	F5	F4	F3	F2	F1	F0	
0	0	x	x	x	x	x	x	010
x	1	x	x	x	x	x	x	110
1	x	x	x	0	0	0	0	010
1	x	x	x	0	0	1	0	110
1	x	x	x	0	1	0	0	000
1	x	x	x	0	1	0	1	001
1	x	x	x	1	0	1	0	111

Equações Lógicas

- Uma versão simplificada do projeto lógico:

ALU OP		Campo de Função					
ALU1	ALU0	F5	F4	F3	F2	F1	F0
x	1	x	x	x	x	x	x
1	x	x	x	x	x	1	x

- **Op2 = Alu0 + Alu1_F1**

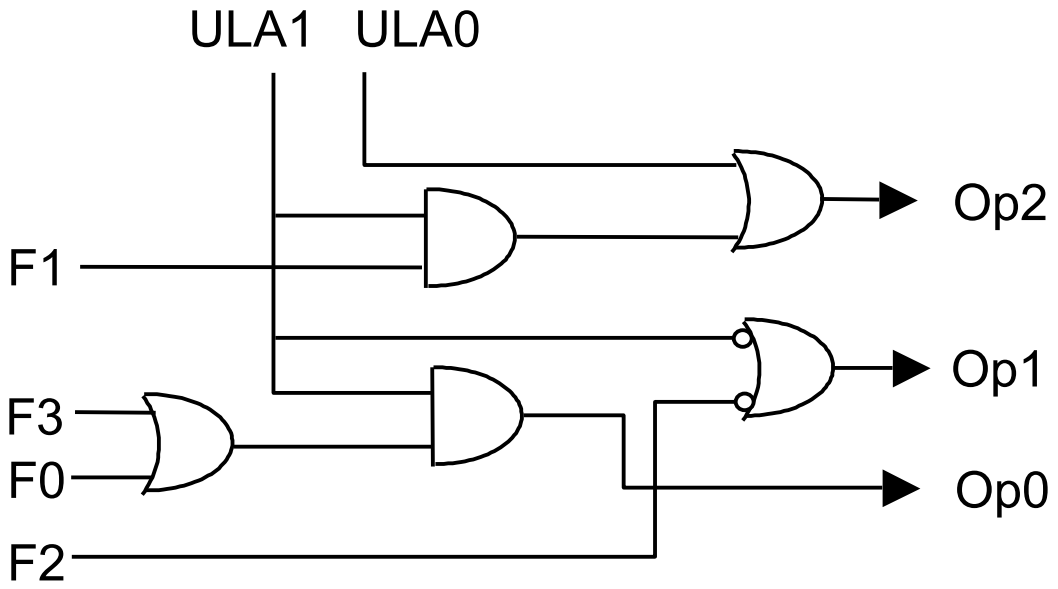
ALU OP		Campo de Função					
ALU1	ALU0	F5	F4	F3	F2	F1	F0
0	x	x	x	x	x	x	x
x	x	x	x	x	0	x	x

- **Op1 = $\overline{Alu0}$ + $\overline{Alu1}$**

Equações Lógicas ...

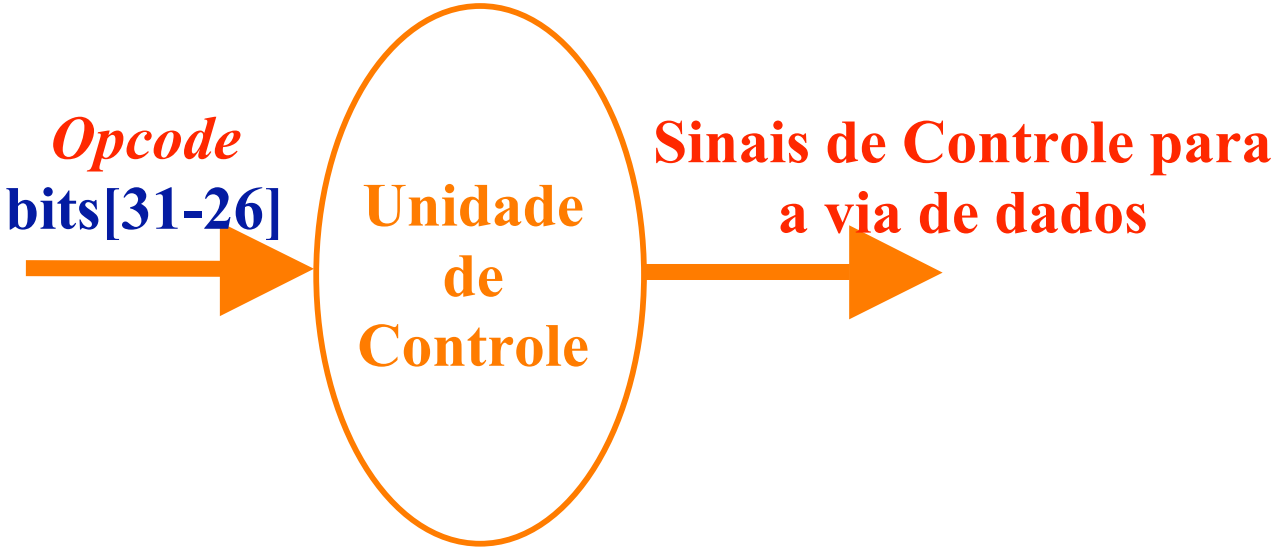
ALU OP		Campo de Função					
ALU1	ALU0	F5	F4	F3	F2	F1	F0
1	x	x	x	x	x	x	1
1	x	x	x	1	x	x	x

- **Op0 = Alu1(F3 + F0)**



Unidade de Controle Uniciclo

- A unidade de controle deve, a partir do código da instrução, fornecer os sinais que realizam as instruções na unidade operativa



Controle do Add

- Por exemplo, a execução da instrução
  - *add $t1, $s0, $s2*

  requer as seguintes tarefas:
  - encaminhar para a ULA o conteúdo dos registradores $s0 e $s2
  - indicar para a ULA que vai ser realizada uma operação da adição
  - Encaminhar o resultado para o registrado $t1

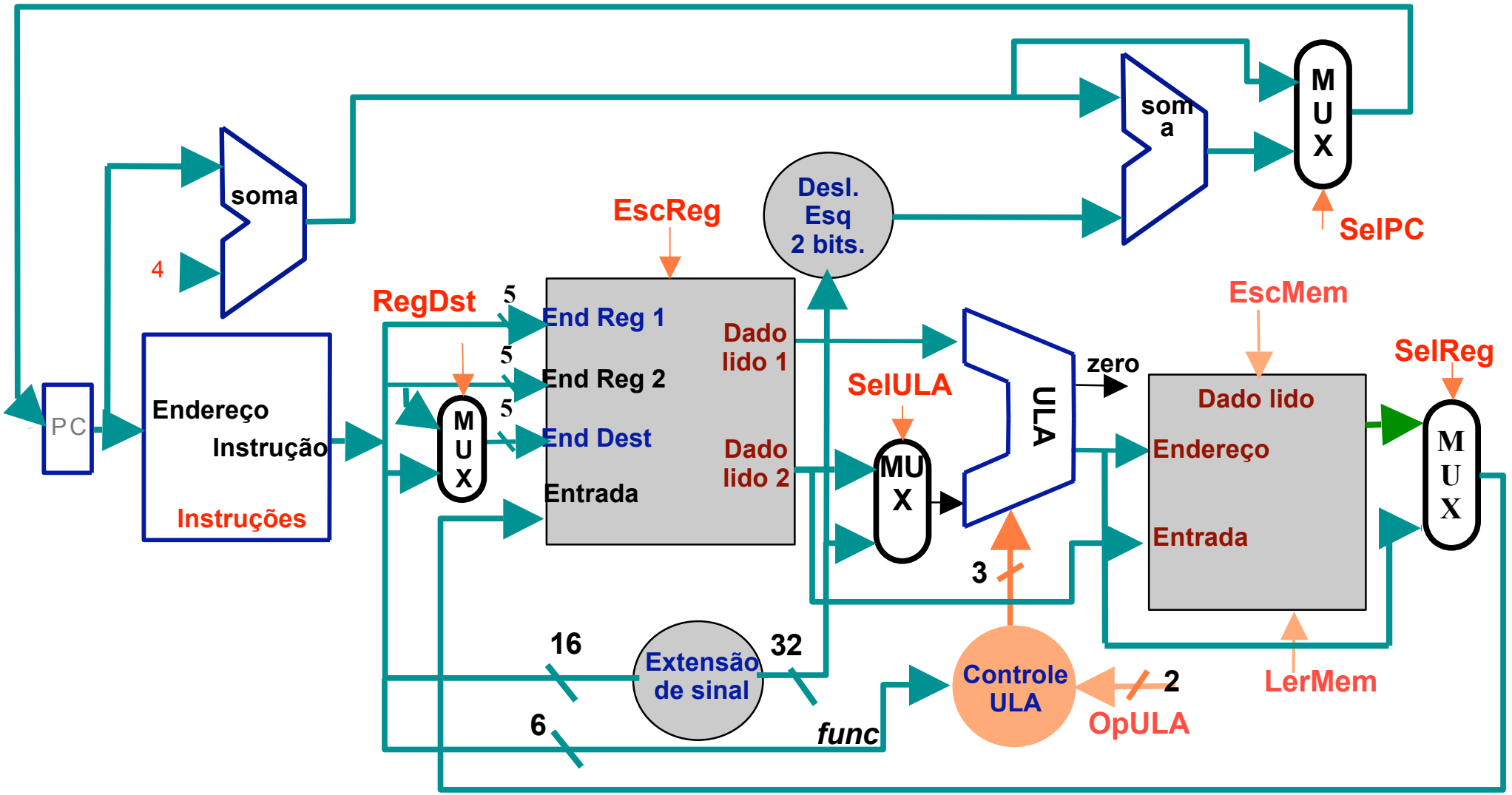
Entrada da Unidade de Controle

- as informações necessárias a execução de uma instrução são retiradas da própria instrução
  - O *opcode* (código de operação) sempre está nos bits [31-26]
  - Os 2 registradores a serem lidos (rs e rt) sempre estão nas posições [25-21] e [20-16] (para todos os formatos!!!)
  - O registrador base (rs) para as instruções lw e sw sempre está especificado nas posições [25-21]
  - Os 16 bits de deslocamento para as instruções beq, lw e sw estão sempre nas posições [15-0]
  - O registrador destino está em uma das duas posições
  - [20-16] para lw (registrador rt)
  - [15-11] para instruções aritméticas/lógicas (registrador rd)

Sinais de Controle

- A unidade de controle de prover:
  - sinais para os multiplexadores
  - sinais de leitura e escrita para as memórias
  - seleção da operação da ULA
  - controle do novo endereço a ser carregado no PC, para instruções de salto

Sinais de controle



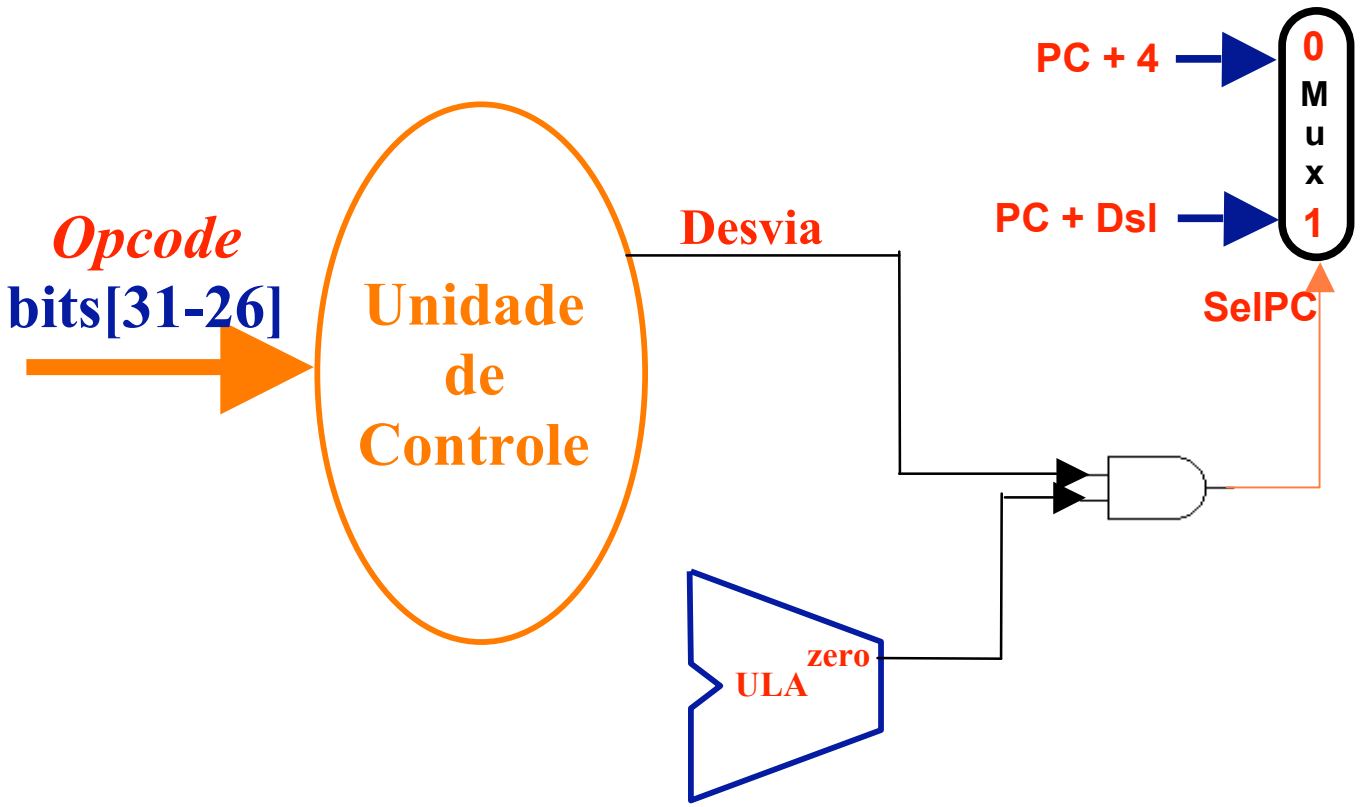
Instruções de Desvio

- A instrução de desvio condicional coloca:
  - PC+Desl ou (salto)
  - PC + 4 (instrução seguinte)

no contador de programa PC

- É necessário selecionar SelPC em função:
  - do código da instrução (beq, bne)
  - do resultado da comparação (sinal zero da ULA)

Controle de Desvio



Sinais de Controle

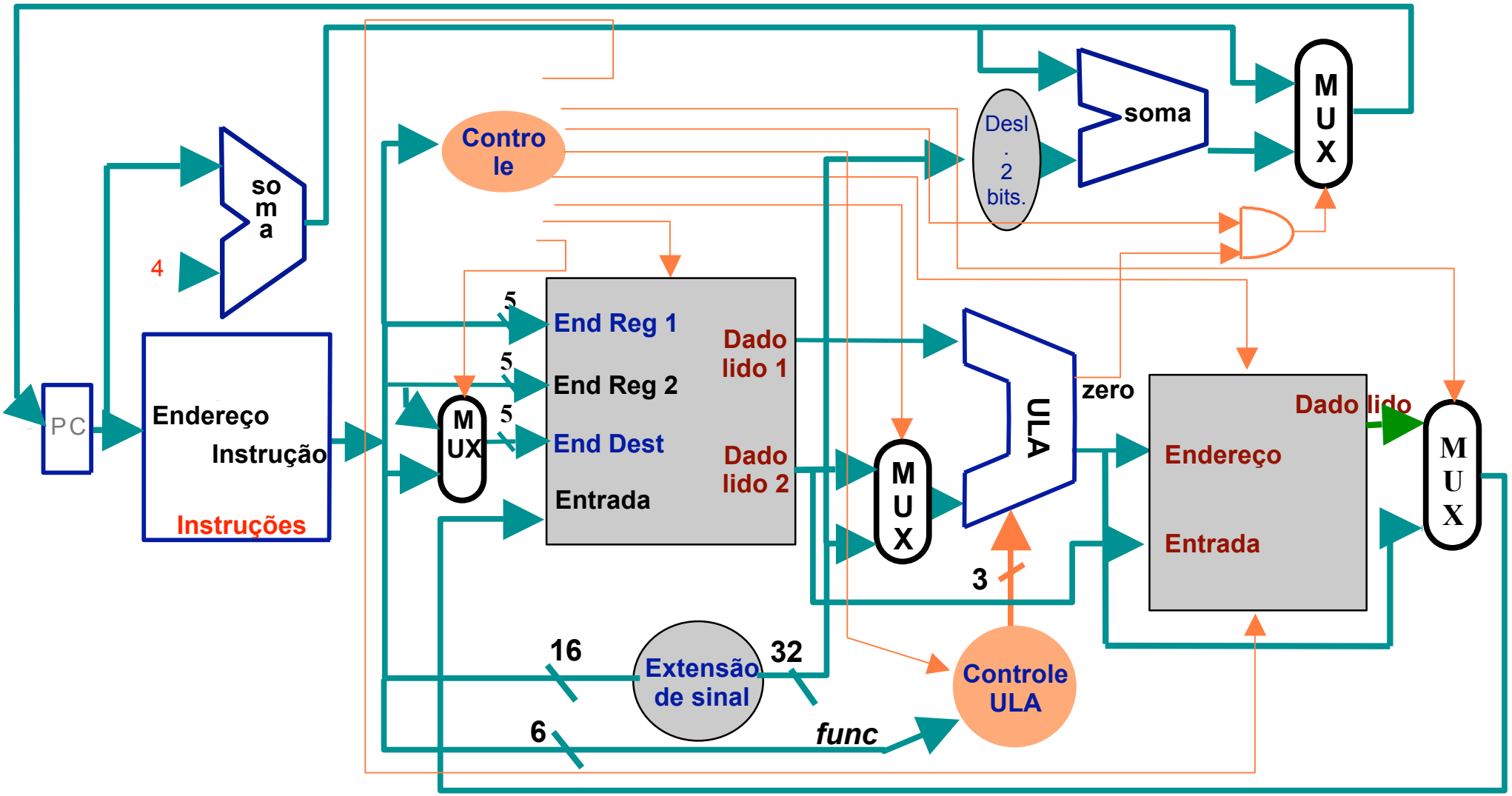
- LerMEM: dado da memória no endereço especificado é lido e colocado na saída
- EscMEM: conteúdo da memória endereçado recebe o dado
- SelULA: 0 – operando é a segunda saída do banco de registradores
  1 – operando é o deslocamento extendido
- RegDST: 0 – índice do registrador destino é o campo rt da instrução
  1 – índice do registrador destino é o campo rd da intrução
- EscREG: escreve dado no registrador indexado
- SelPC: 0 – PC recebe próximo endereço (PC+4)
  1 – PC recebe endereço de desvio
- SelREG: 0 – registrador recebe saída da ULA
  1 – registrador recebe saída da memória

Acionamento dos Sinais

- O acionamento é determinado diretamente pelos códigos das instruções

Instrução	RegDST	SelULA	SelREG	EscREG	LerMEM	EscMEM	Desvia	Ula1	Ula0
Tipo R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

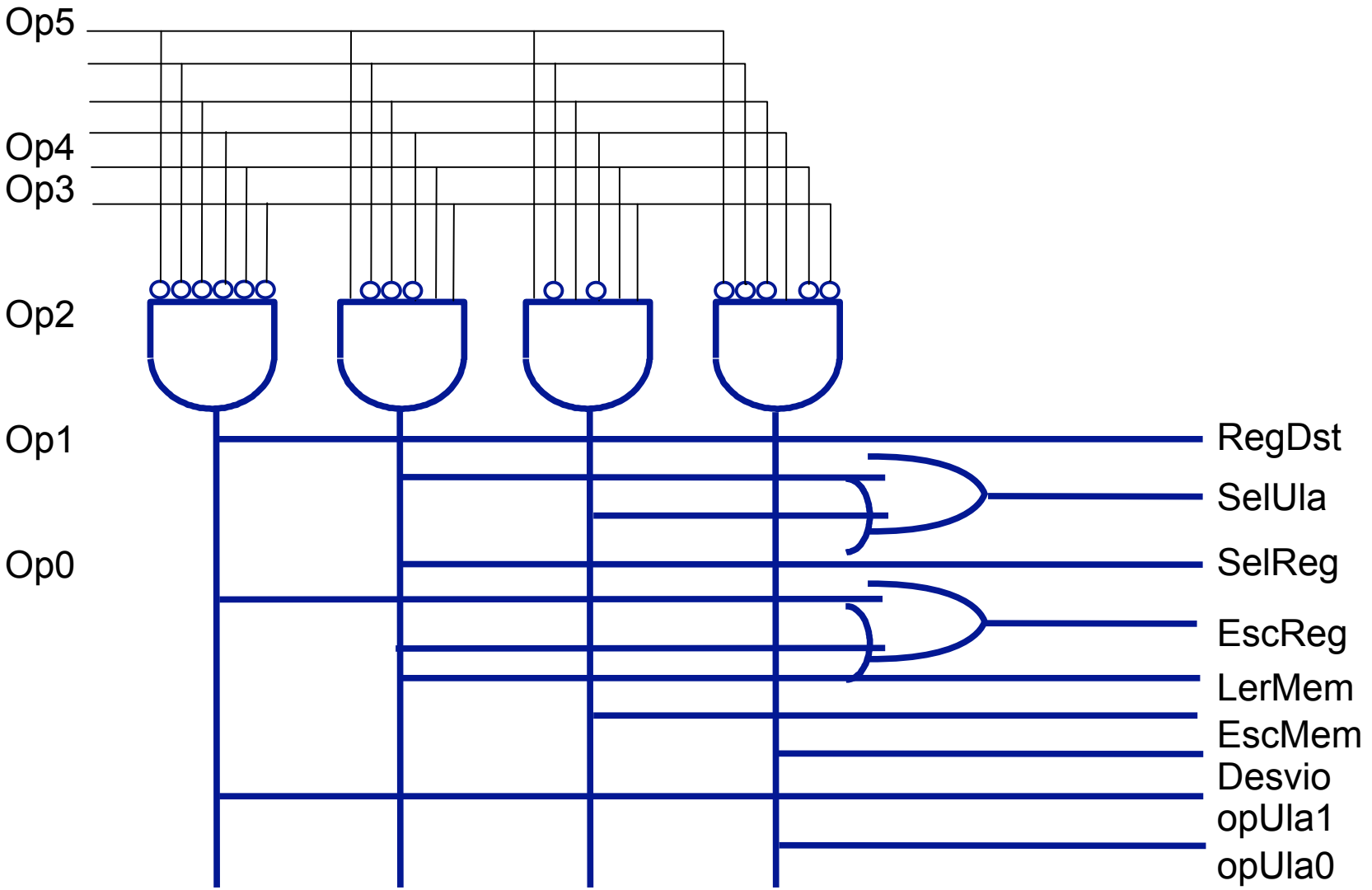
MIPS Uniciclo



Lógica de Controle

		Rformat	lw	sw	beq
Entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	1	1	0
	Op1	0	1	1	0
	Op0	0	1	1	0
Saídas	RegDst	0	1	1	0
	SelUla	0	1	1	0
	SelReg	0	1	1	0
	EscReg	0	1	1	0
	LerMem	0	1	1	0
	EscMem	0	1	1	0
	Desvio	0	1	1	0
	opUla1	0	1	1	0
	opUla0	0	1	1	0

Implementação em PLA



Exercício

- Estender a organização do MIPS para dar suporte a execução de JUMP, desvio incondicional
- O endereço de desvio é obtido por:
  - PC[31 – 28] # Instrução[25 – 0] # 00
  - onde # indica concatenação de bits

Problemas com MIPS Uniciclo

- Período do relógio determinado pelo caminho mais longo
  - instrução lw:
    - leitura da instrução
    - leitura do registrador de base, extensão de sinal
    - cálculo do endereço
    - leitura do dado da memória
    - escrita em registrador
- TODAS as instruções levam o mesmo tempo para executar

Exemplo

- Supondo os seguintes tempos de execução das unidades do MIPS:
  - Acesso a memória: 10 ns
  - ULA e somadores: 10 ns
  - Acesso ao banco de registradores: 5 ns
  - outros: 0 ns
  - **Quais os tempos de execução das instruções supondo uma implementação uniciclo e outra com ciclo variável, ou seja, duração do ciclo igual a duração da instrução?**

Exemplo ...

- Considerando a distribuição de instruções do benchmark gcc, a diferença de velocidade entre as implementações seria:
  - GCC: 22% lw, 11% sw, 49% tipo-R, 16% beq, 2% jump
  - Período uniciclo: 40 ns
  - Período ciclo variável:

    40*0.22 + 35*0.11 + 30*0.49 + 25*0.16 + 10*0.2 = 31.6 ns
  - Ganho: 40/31.6 = 1,27

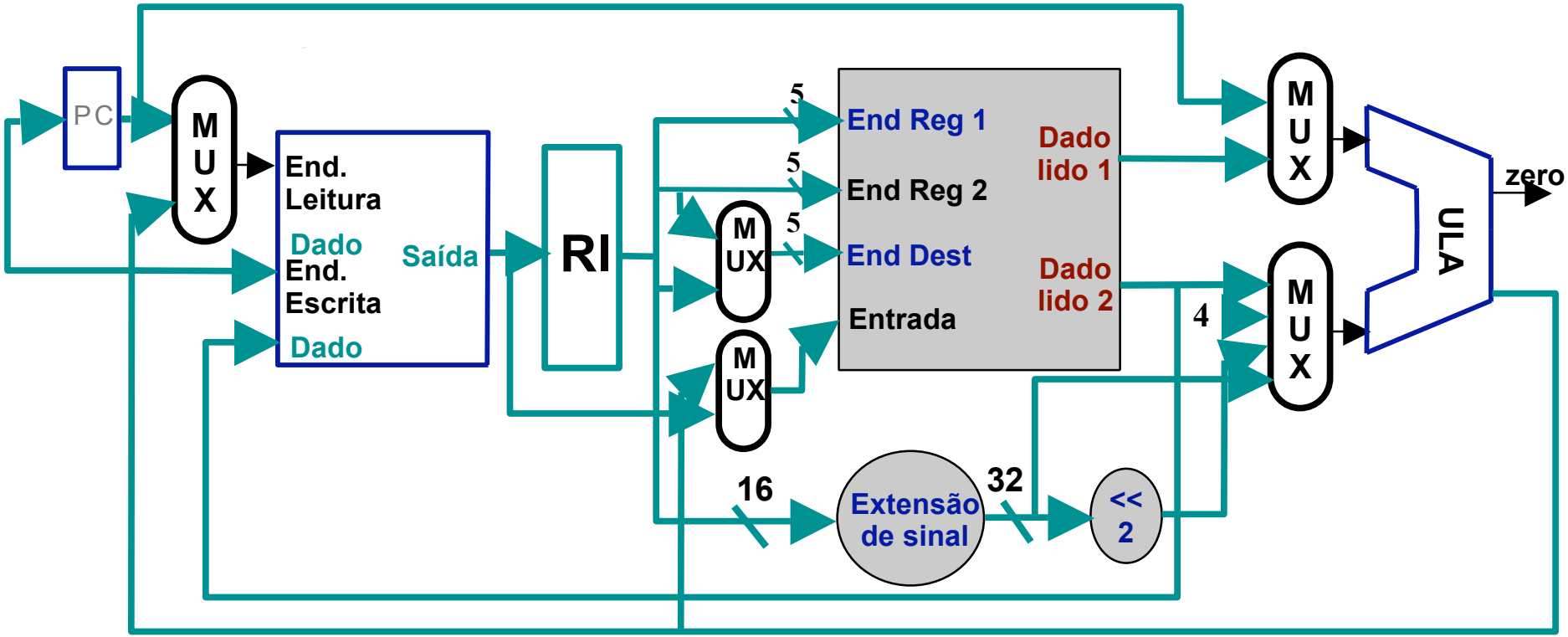
MIPS Multiciclo

- Idéia: cada fase de execução de uma instrução dura um ciclo de relógio
- Instruções podem ser executadas em um número diferente de ciclos
  - lw leva 5 ciclos
  - jump leva 1 ciclo
  - add leva 4 ciclos

MIPS Multiciclo

- Ciclo dimensionado de acordo com a fase mais demorada
- Unidades funcionais podem ser utilizadas para realizar mais de uma operação durante a execução de uma instrução
- A organização da parte operativa pode ser re-estruturada em função destas características

PO Multiciclo



PO Multiciclo x PO Uniciclo

- Apenas uma memória, com instruções e dados
- Incremento do PC, cálculo do endereço de desvio, operações lógico-aritméticas realizadas todas pela mesma unidade funcional
- Introdução e ampliação dos multiplexadores nas entradas da ULA
- Introdução do registrador de Instruções (RI), para armazenar a instrução lida

Controle Multiciclo

1. Busca da Instrução
    1. RI = Memoria[PC]
    2. PC = PC + 4
2. Decodificação e busca de operandos

    2.1 A = Reg[RI[25-21]]

    2.2 B = Reg[RI[20-16]]

    2.3 Rdesvio = PC + (ext-sinal(IR[15-0])) << 2

    - Rdesvio armazena o endereço de desvio pré-calculado de forma a liberar a ULA para outras operações
    - a utilização destes valores depende do tipo de instrução

Controle Multiciclo ...

3. Execução

   3.1 sULA = A + ext-sinal(IR[15-0]) (acesso a memória)

   3.2 sULA = A op B (tipo R)

   3.3 if (A == B) PC = Rdesvio (desvio condicional)

4. Acesso à memória ou escrita de registador

   4.1 Acesso à memória:

   saídaMem = Memória[saiULA] ou (load)

   Memória[saiULA] = B (store)

   4.2 Reg[IR[15-11]] = saiULA; (R-type)

Controle Multiciclo ...

5. Escrita da Memória

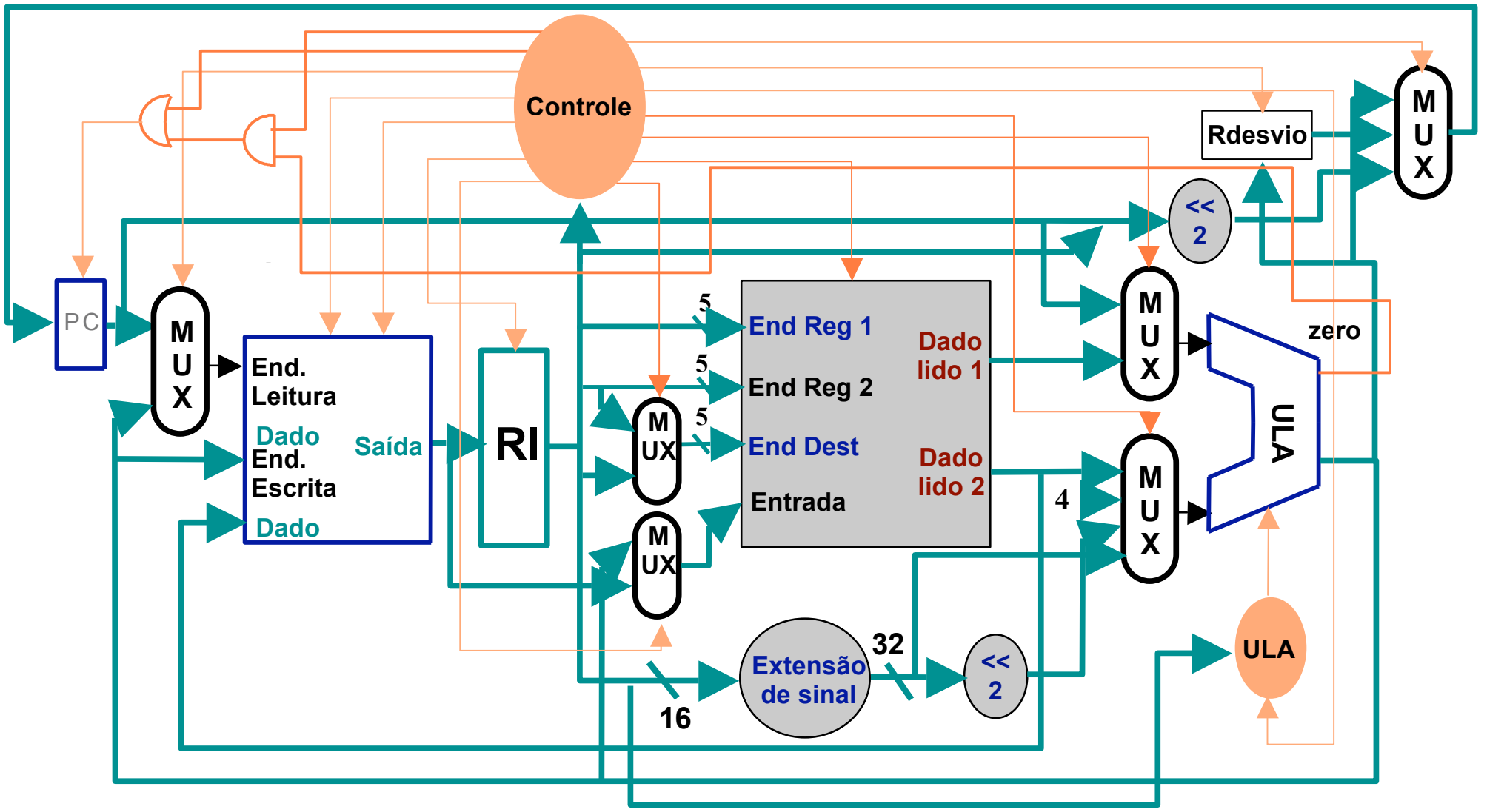
Reg[IR[20-16]] = saidaMemória;

Atualização do PC:

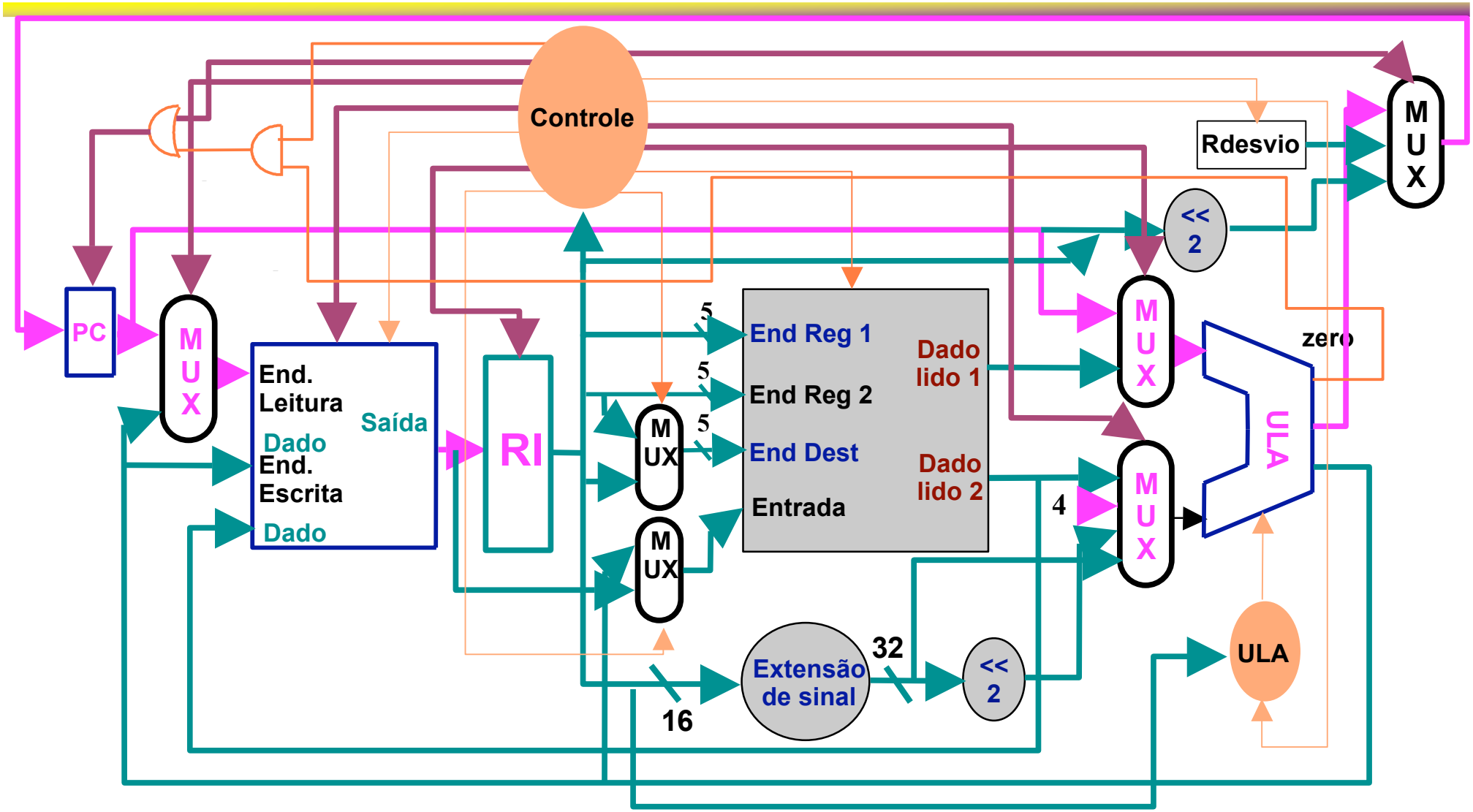
- saída da ULA: endereçamento sequencial
- Rdesvio: qdo um salto condicional é realizado
- Jump: concatenação do PC com bits do RI

- multiplexador na entrada do PC para selecionar uma destas entradas

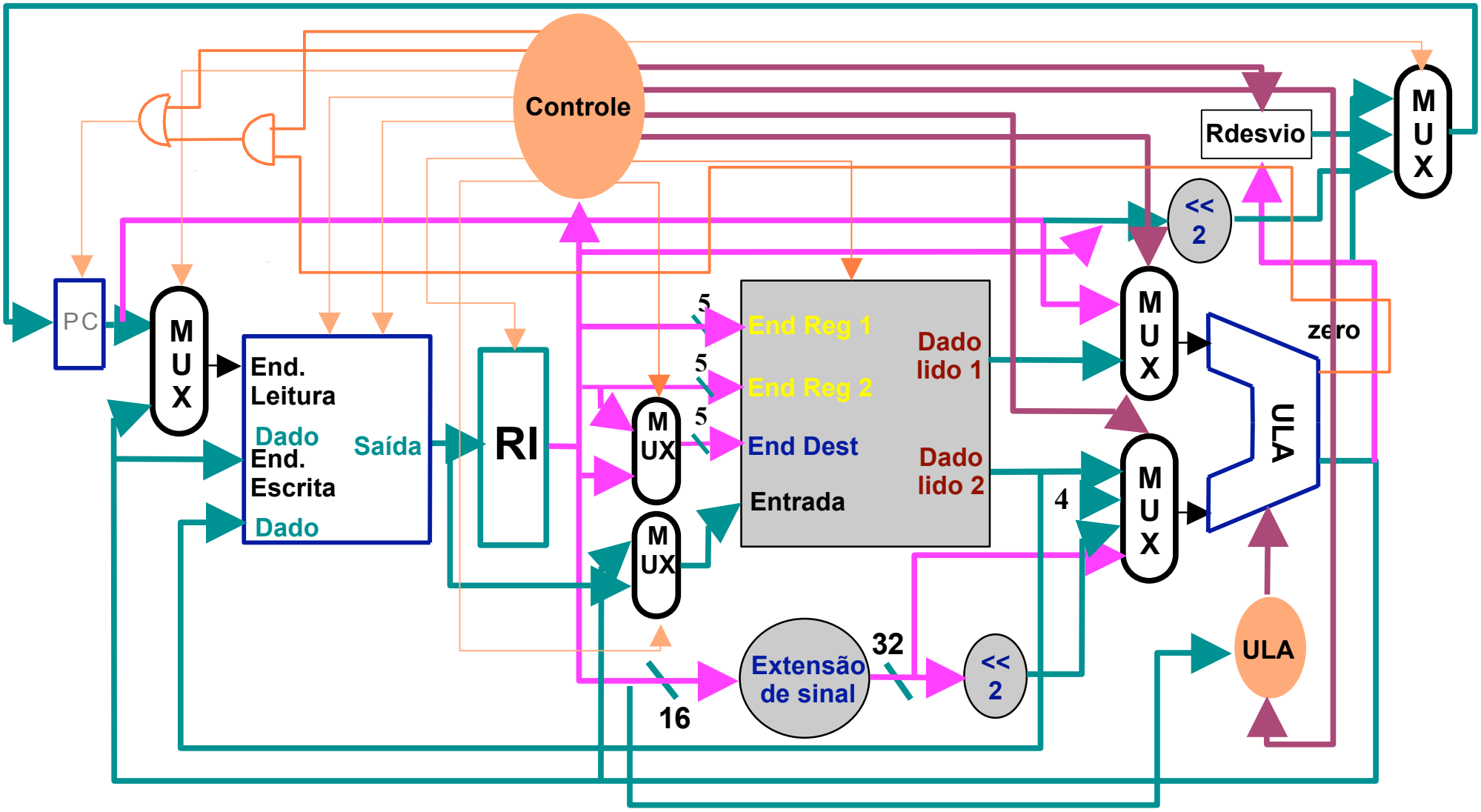
MIPS Multiciclo



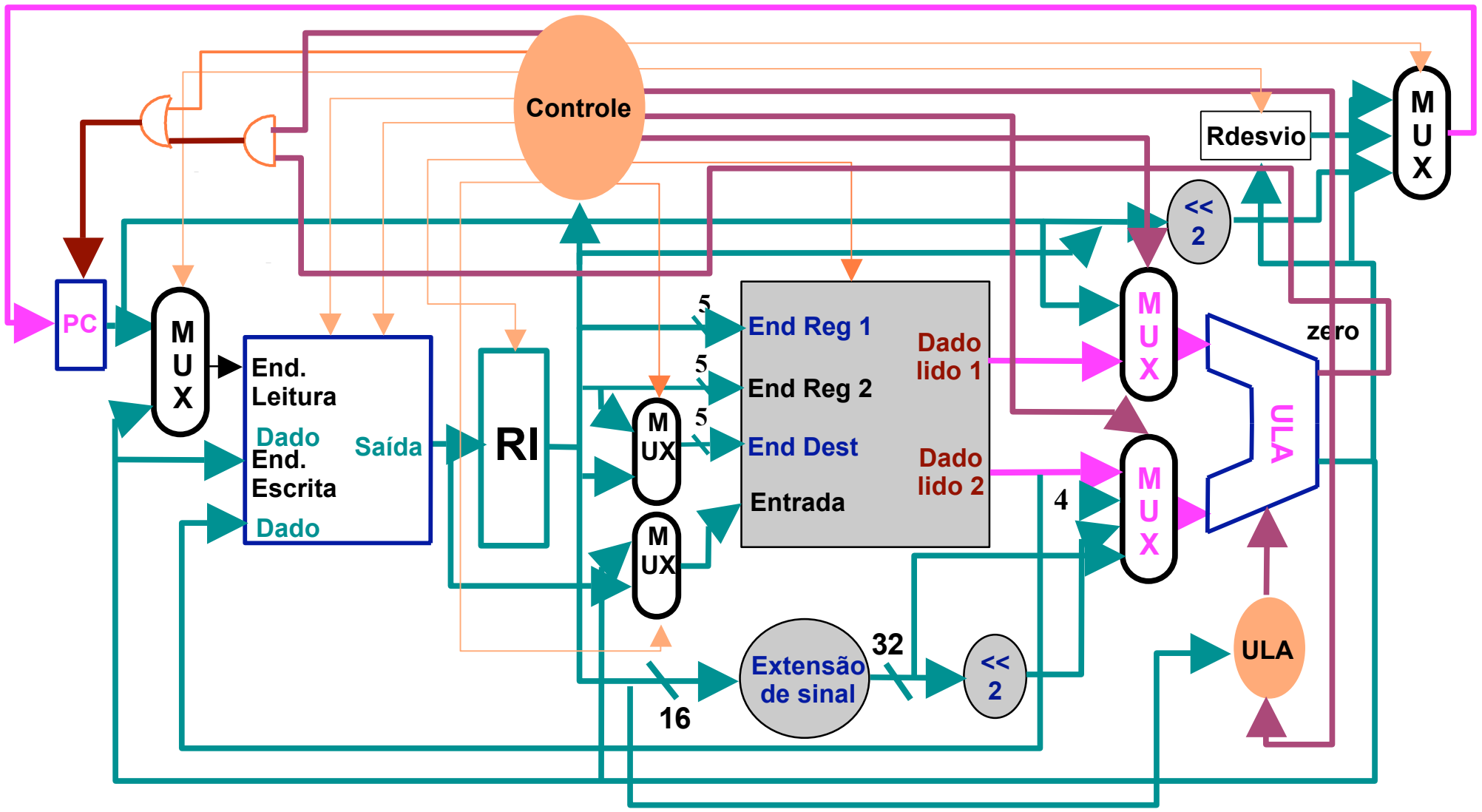
Fase 1: Busca


Controle
Rdesvio
MUX
<<
2
PC
MUX
End.
Leitura
Saída
Dado
End.
Escrita
Dado
RI
5
5
5
MUX
MUX
End Reg 1
End Reg 2
End Dest
Entrada
Dado
lido 1
Dado
lido 2
MUX
MUX
4
zero
ULA
ULA
Extensão
de sinal
32
16
<<
2

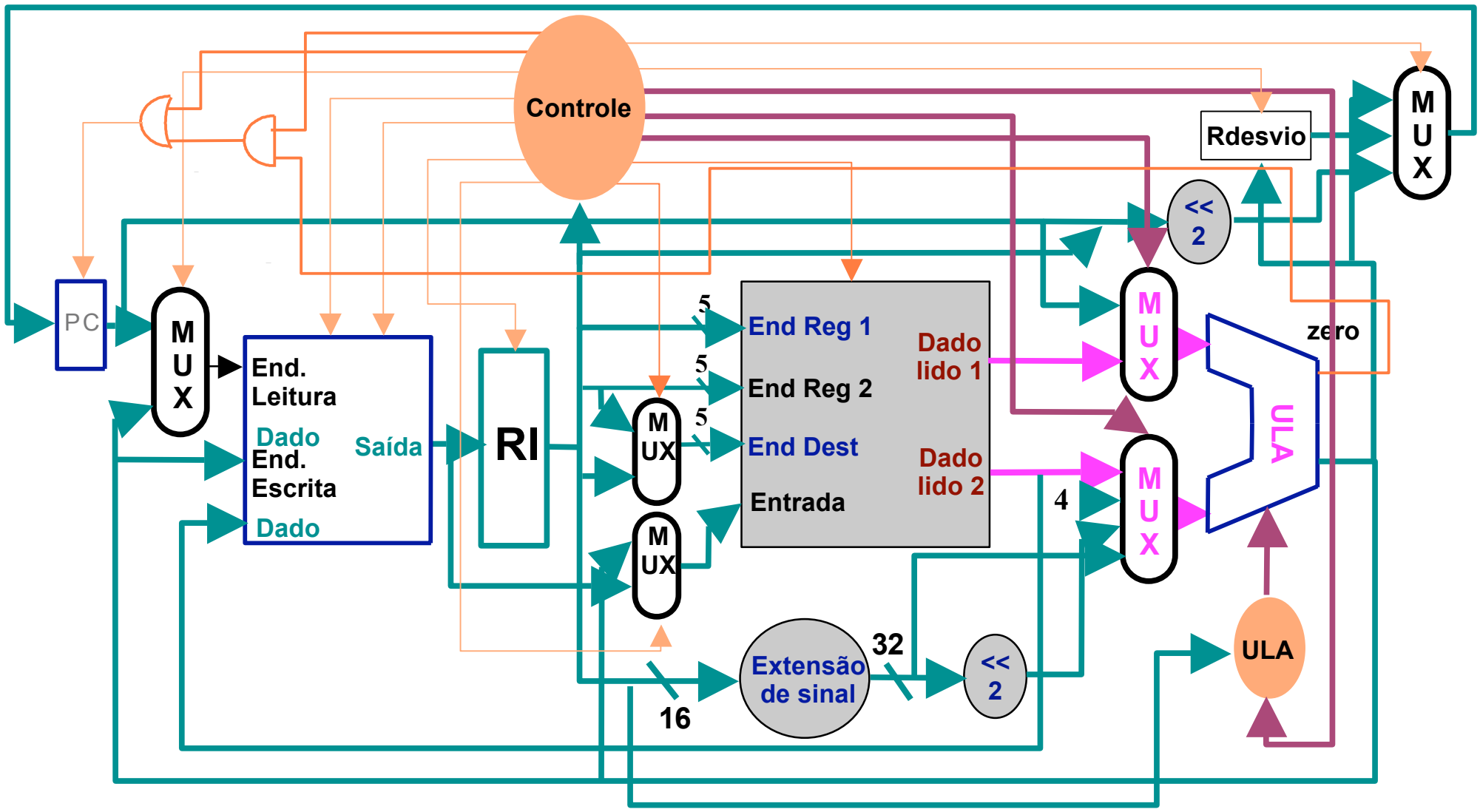
Fase 2: decodificação



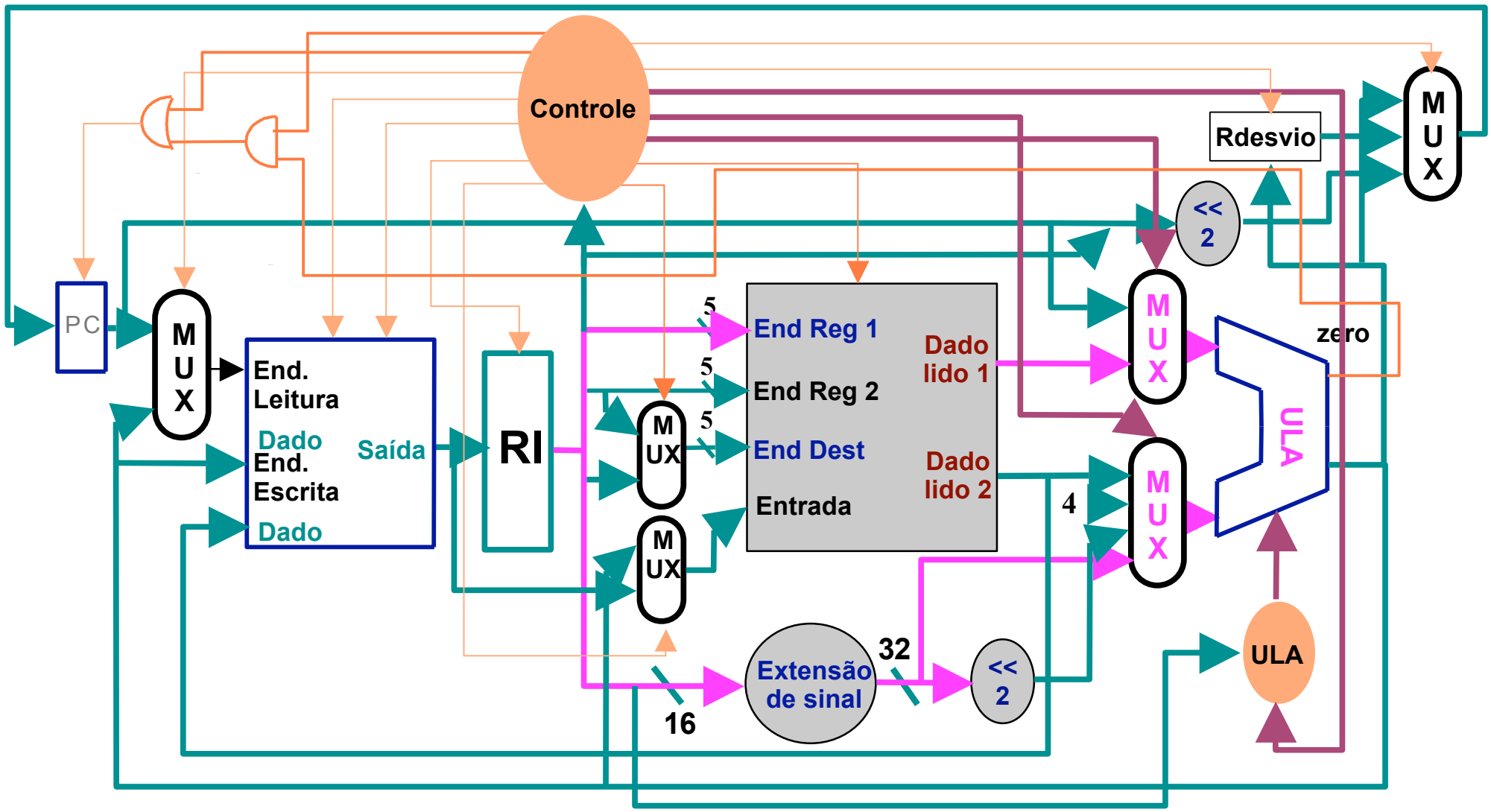
Fase 3: Execução Desvio



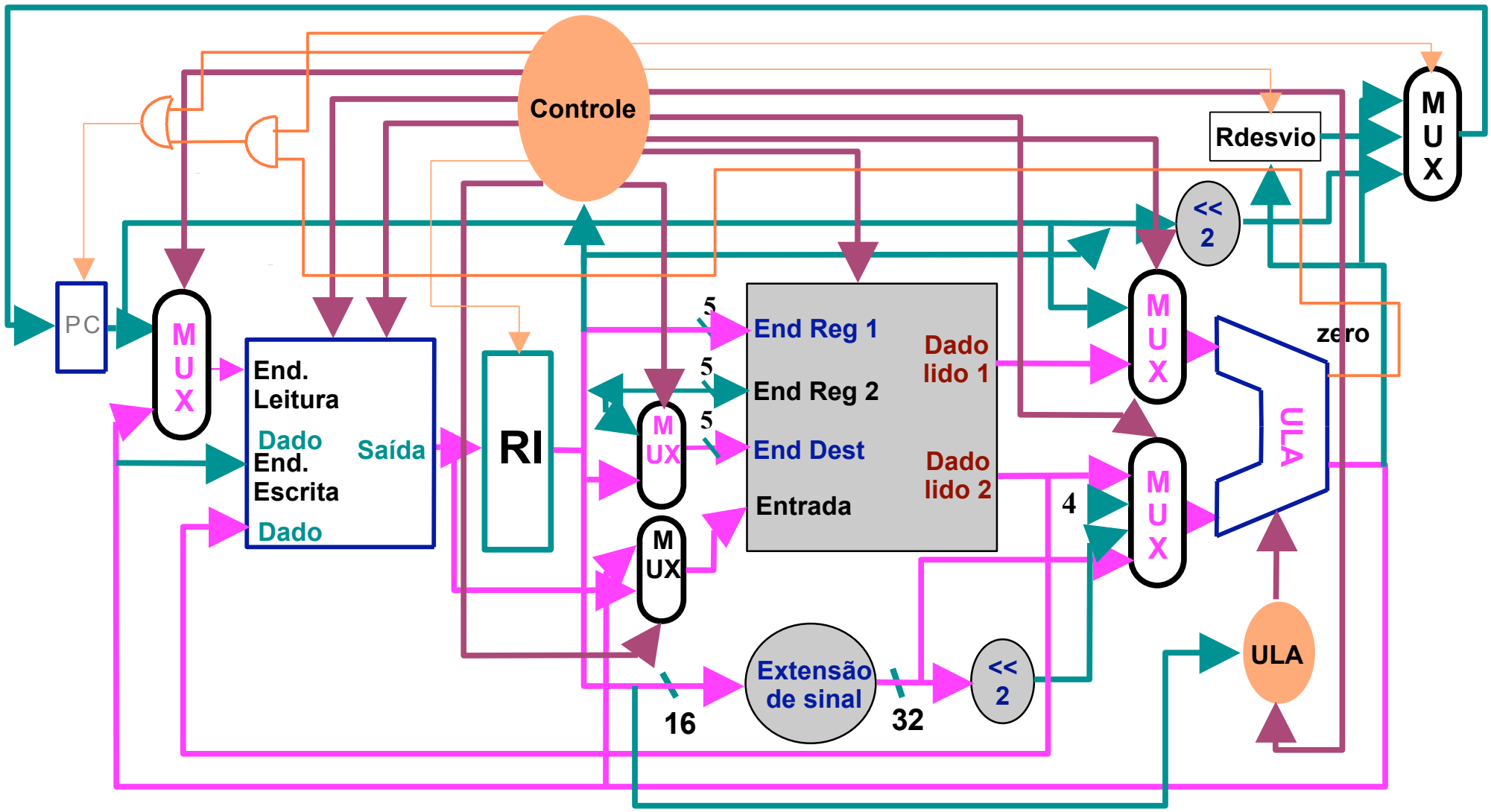
Fase 3: Execução Log-Aritmética



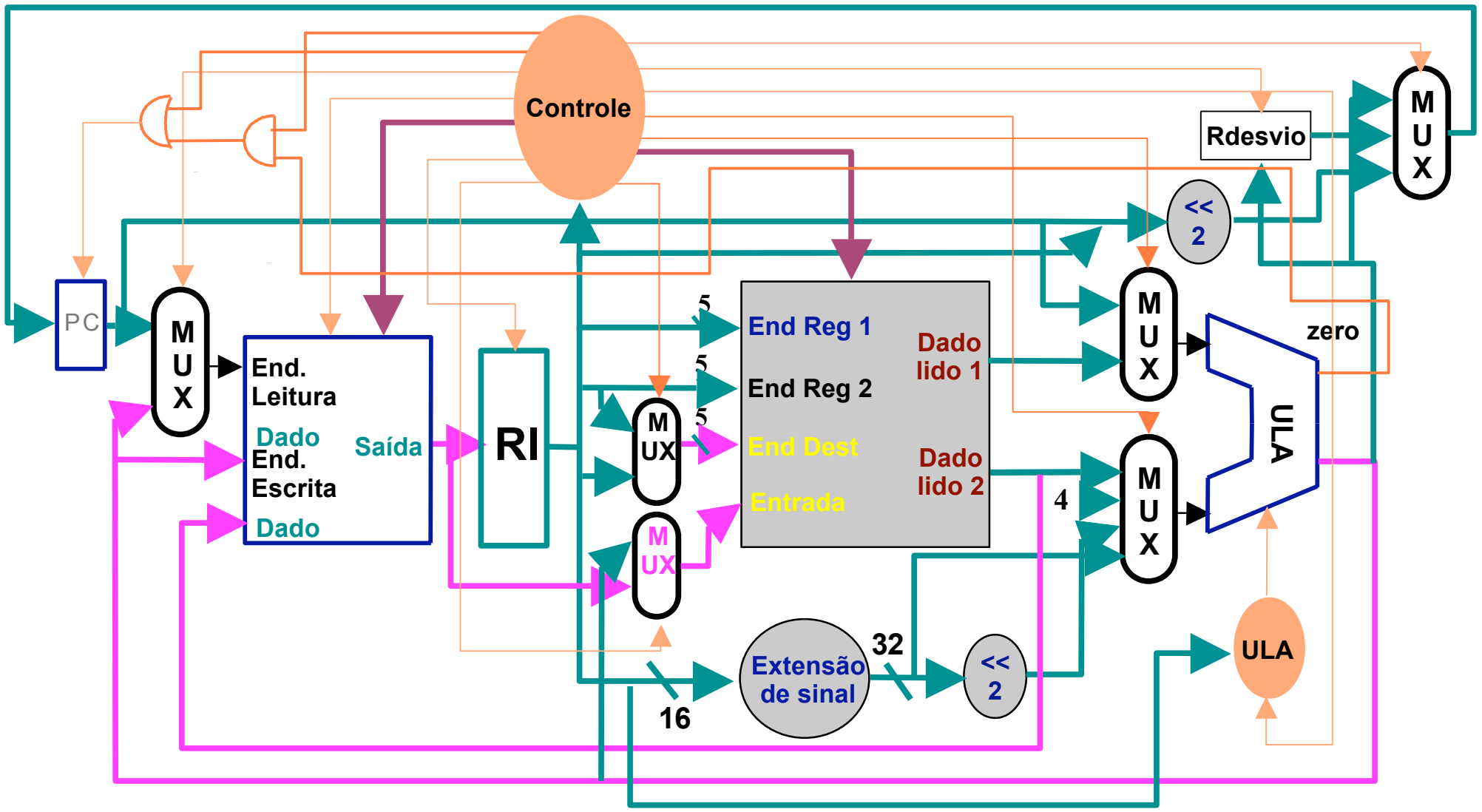
Fase 3: Execução Acesso Memória



Fase 4: Acesso Memória/Registrador

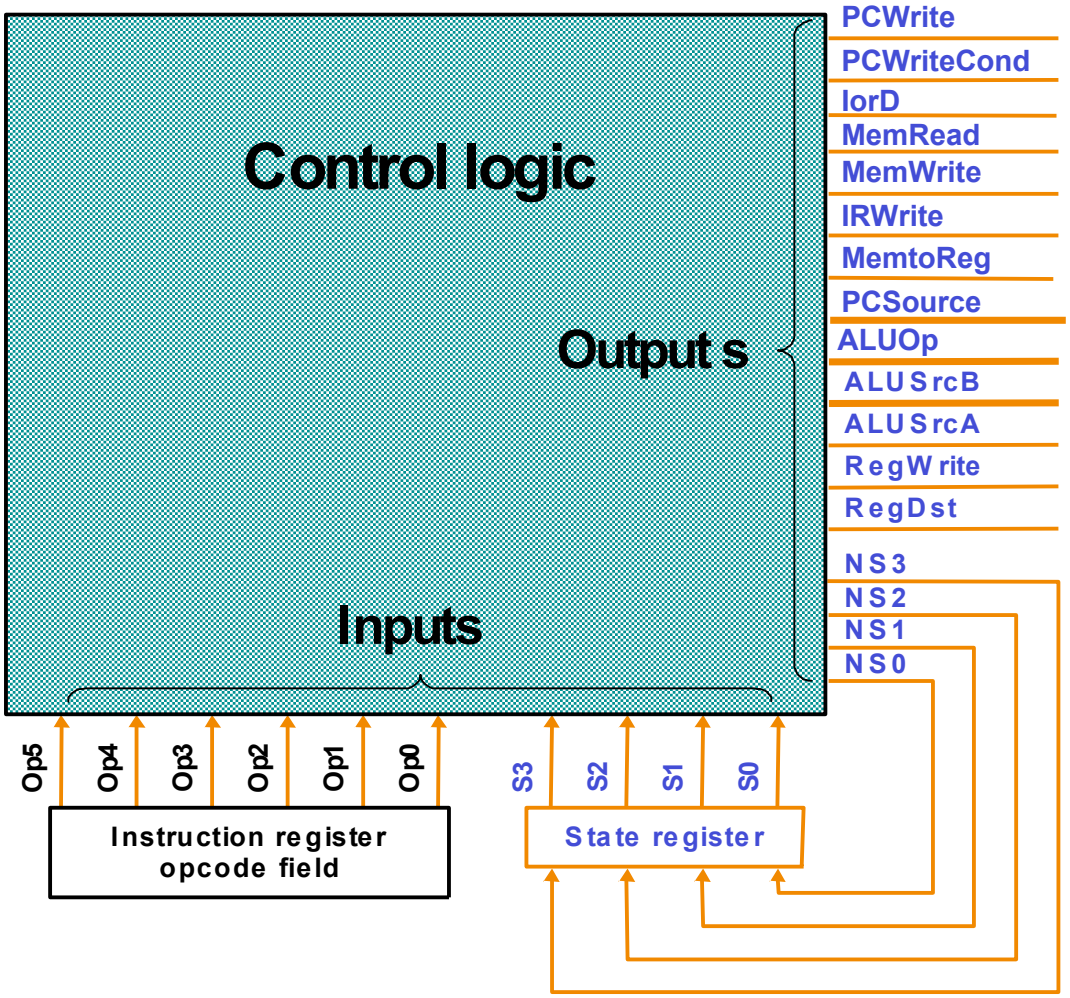


Fase 5: Escrita Registrador (lw)



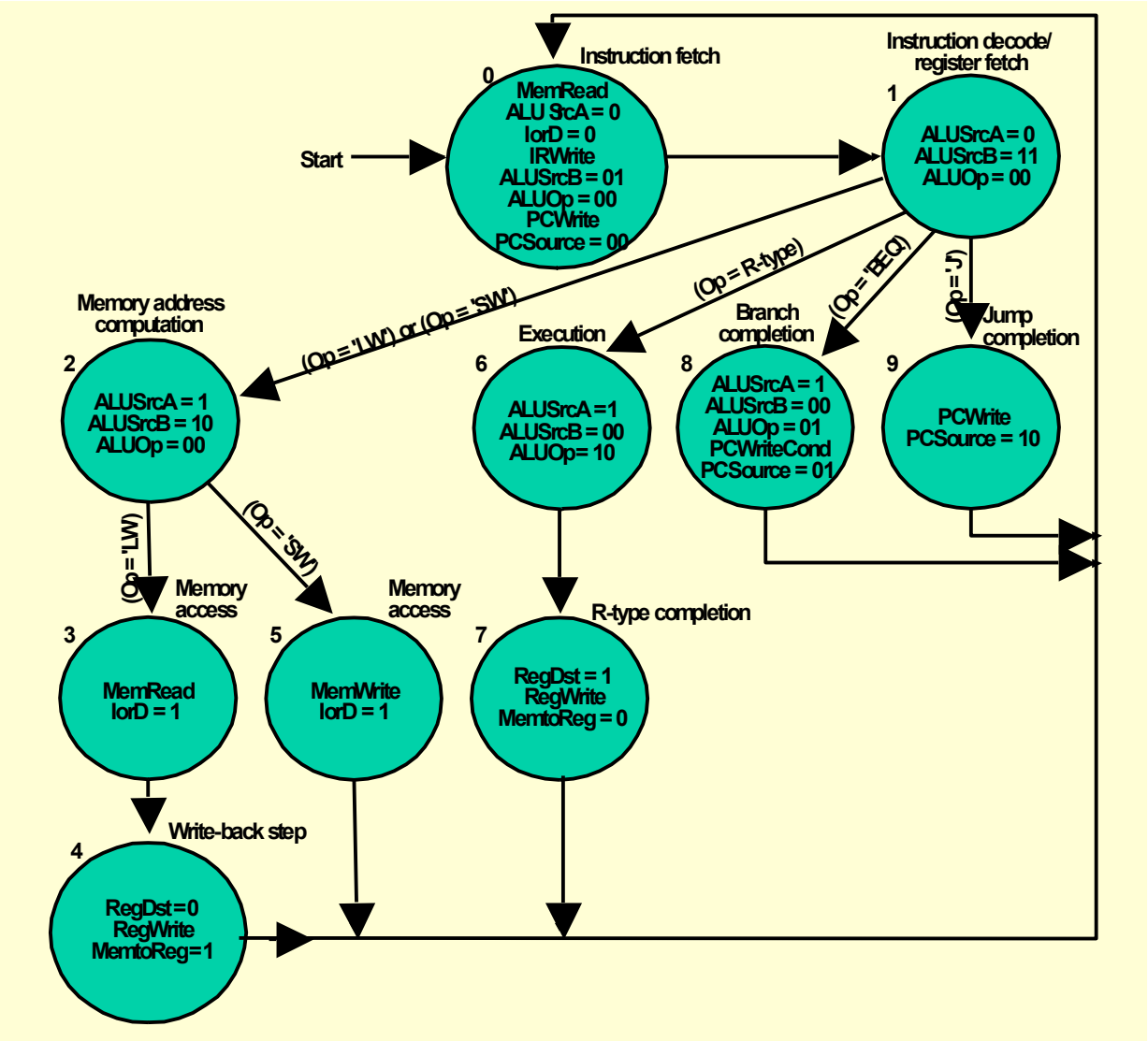
Controle com MEF

- estrutura típica de uma máquina de estados:
  - lógica de saída
  - lógica de transição
  - registrador de estado
  - entradas externas (código da instrução)



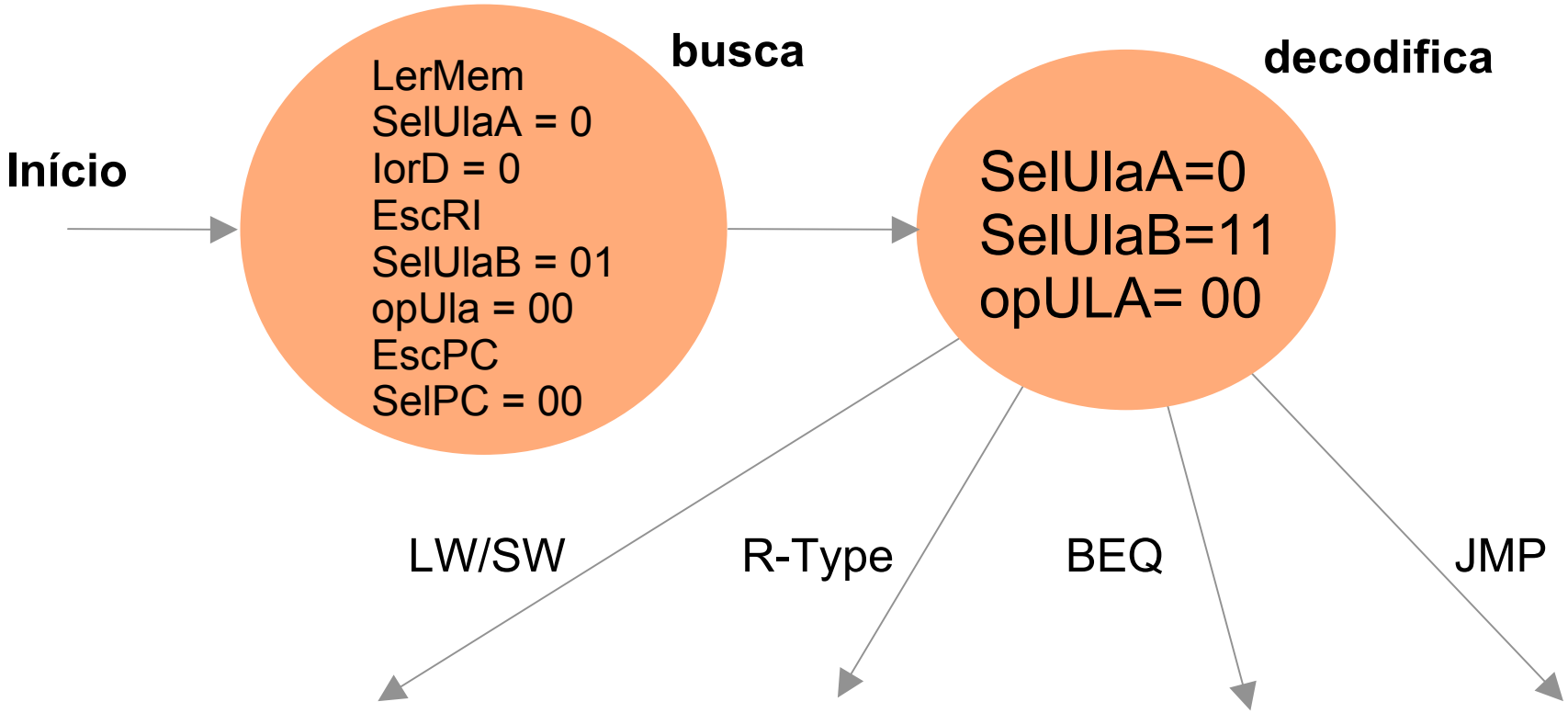
MEF: Diagrama de Estados

- Cada nó do diagrama representa um estado
- A transição entre estados é indicada por arcos
- As condições de disparo de uma transição são associadas aos arcos
- Cada estado corresponde a um ciclo de relógio

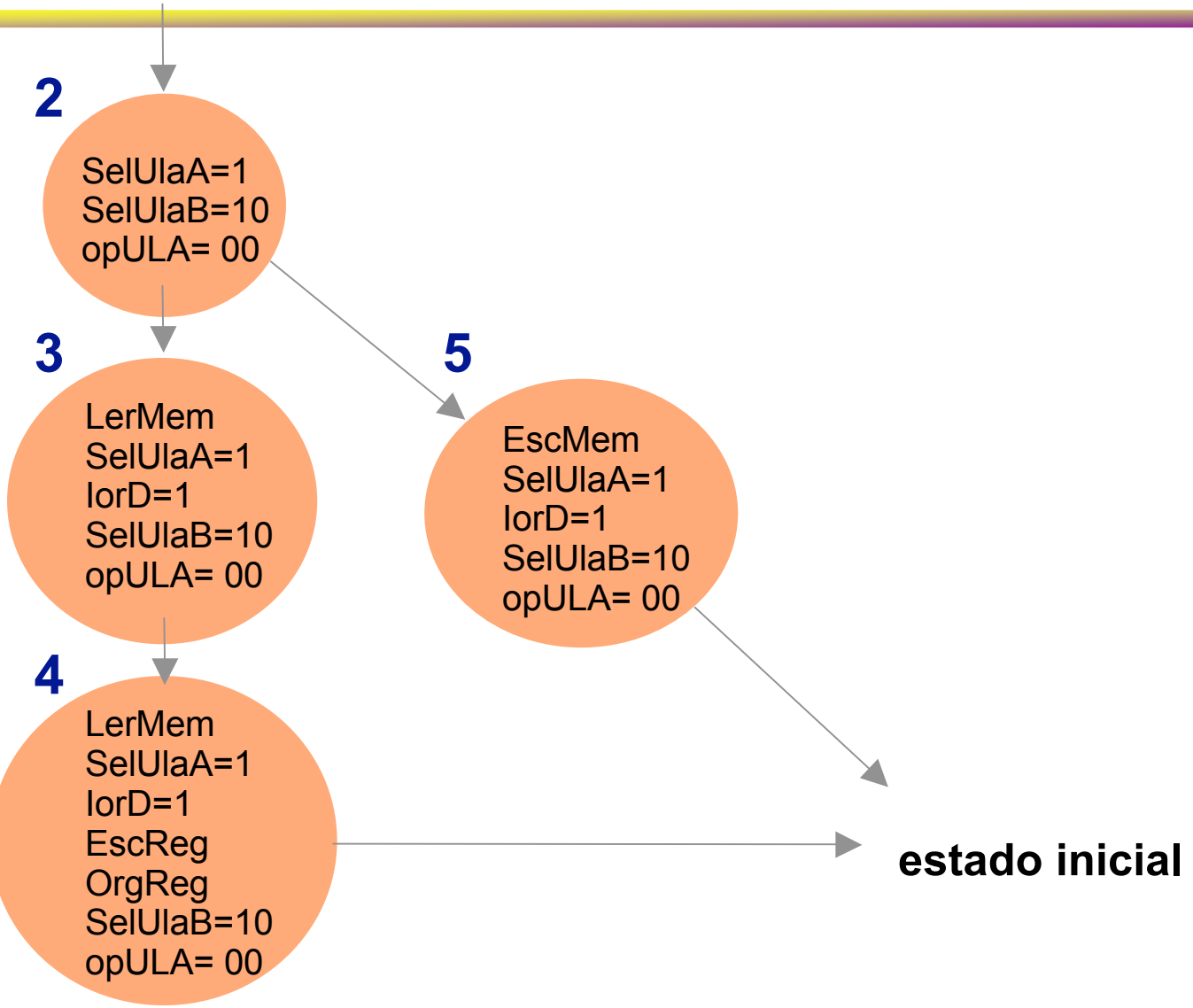


Detalhando a MEF (I)

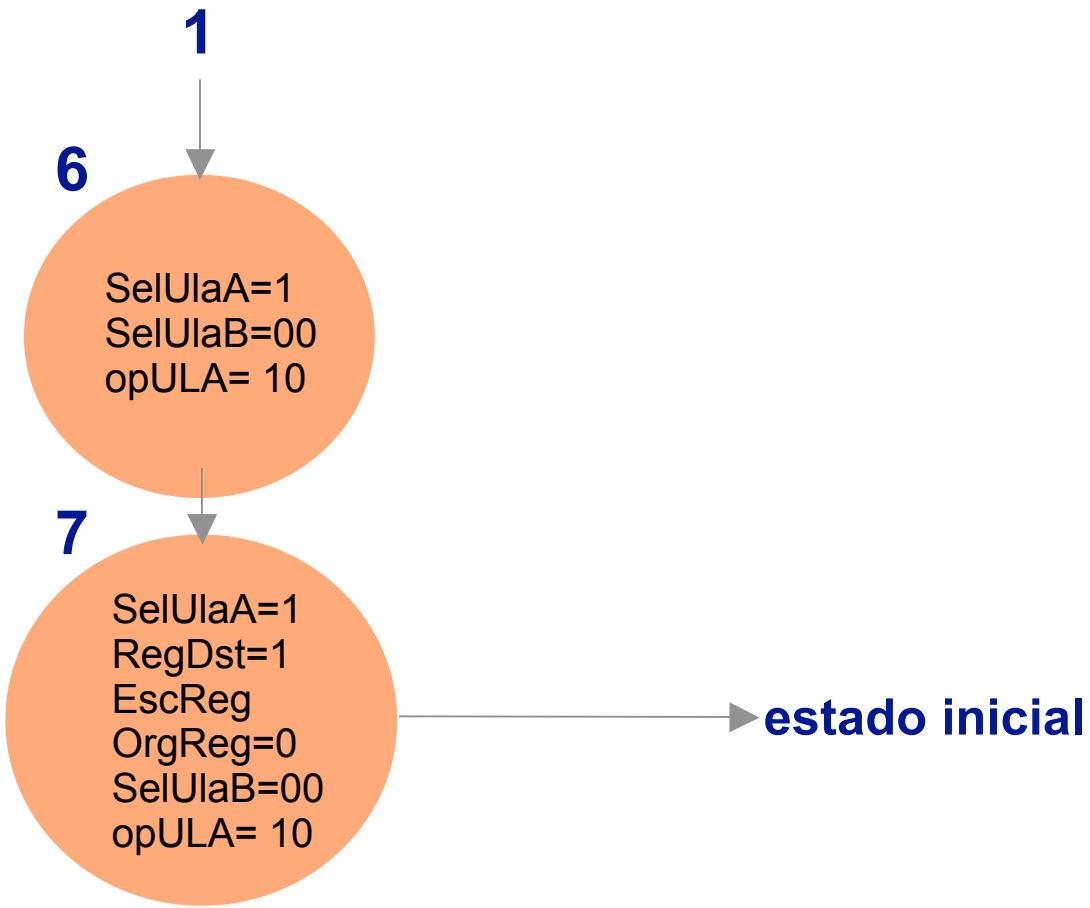
- Busca e decodificação



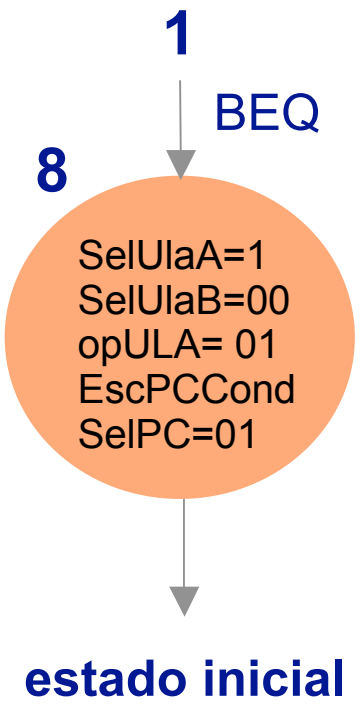
Detalhando a MEF (II)

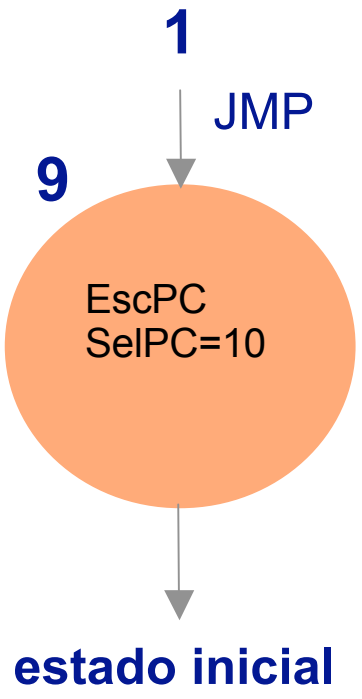


Detalhando a MEF (III)



Detalhando a MEF (IV)





Exemplo

- Considerando o exemplo do gcc, como fica o CPI do MIPS multiciclo ?
  - lw: 22% (5 ciclos)
  - sw: 11% (4 ciclos)
  - logarit: 49% (4 ciclos)
  - ramificações: 16% (3 ciclos)
  - saltos: 2% (3 ciclos)

Exemplo

- *CPI = 0.22*5 + 0.11*4 + 0.49*4 + 0.16*3 + 0.02*3 =*

  1.1 + 0.44 + 1.96 + 0.48 + 0.06 = 4.04

Microprogramação

- O projeto da parte de controle através de diagramas de transição de estados pode rápidamente se tornar inviável se o número de estados for muito grande
- MEF's de processadores complexos podem ter milhares de estados
- Uma alternativa para projeto é seguir um processo semelhante a programação

Microinstruções

- Uma microinstrução é definida pelos valores dos sinais de controle que atuam na unidade operativa durante um estado da MEF
- A execução de uma instrução do processador pode então ser realizada através de uma sequência de microinstruções
- O conjunto de microinstruções que implementa o controle de um processador é chamado de microprograma

Microprograma

- O sequenciamento das microinstruções é realizado de forma similar a de um programa normal
  - microinstruções são usualmente executadas em sequência -> correspondem a caminhos no diagrama de estados
  - em alguns casos, a sequência a ser seguida depende de informações externas (código da instrução, por exemplo). Nestes casos, são necessários mecanismos de desvio

Formato da Microinstrução

- A microinstrução é dividida em campos que atuam sobre conjuntos de elementos da unidade operativa
- Os campos são escolhidos de acordo com sua finalidade. O controle da ULA, por exemplo, seria associado a um campo
- O microprograma é usualmente implementado em ROM ou PLA, onde cada microinstrução tem seu próprio endereço

Microinstrução do MIPS

- cntrULA: especifica a operação realizada pela ULA no ciclo de relógio atual
- ula1: define o primeiro operando da ULA
- ula2: define o segundo operando da ULA
- dstULA: especifica o registrador a receber o resultado da ULA
- mem: indica se acesso é de leitura ou escrita e a origem do endereço
- memReg: especifica o registrador envolvido no acesso
- escPC: controla a escrita no PC
- proxMI: especifica como escolher a próxima microinstrução

Sequenciamento da Microinstruções

- Três alternativas:
  - incrementar o endereço da microinstrução atual
    - valor seq no campo proxMI
  - saltar para o início do microprograma (inicia execução de uma nova instrução do MIPS)
    - valor fetch no campo proxMI
  - desviar de acordo com informação de controle
    - valor despacha i no campo proxMI
    - o despacho de microinstruções é realizado com o auxílio de tabelas, implementadas com PLAs/ROMs

Valores dos Campos

- cntrULA
  - **soma**: ULA realiza uma soma
  - **func**: operação definida pelo campo func da instrução
  - **sub**: operação de subtração
- ula1
  - **PC**: usa o PC como primeiro operando
  - **rs**: usa o registrador rs do banco de registradores
- ula2
  - **4**: usar a constante 4 como segundo operando
  - **extend**: usar a saída da extensão de sinal
  - **extdsl**: usar a saída do deslocador de 2 bits
  - rt: usar o registrador rt do banco como 2º operando

Valores dos Campos

- dstULA
  - **Rdesvio**: saída da ULA é escrita no registrador de desvio
  - **rd**: saída é escrita em registrador
- Memoria
  - **lerPC**: usar PC para endereçar a memória
  - **lerULA**: usar a saída da ULA para endereçar a memória
  - **escULA**: escrever na memória usando ULA para endereçar
- memReg
  - **RI**: dado escrito no registrador de instruções
  - **escRT**: dado escrito no registrador indicado na instrução
  - **lerRT**: dado escrito na memória vem de registrador

Valores dos Campos

- escPC
  - **ULA**: saída da ULA é escrita no PC
  - **Rdesvio**: escreve Rdesvio no PC
  - **jump**: escreve o endereço de salto no PC
- prox
  - **seq**: escolhe a próxima instrução sequencialmente
  - **busca**: volta para a primeira microinstrução
  - **despacha i**: despachar a instrução utilizando a tabela i (1 ou 2)

Microprograma

Label	cntrULA	ula1	ula2	dstULA	Memória	memReg	escPC	prox
busca	soma	PC	4		lerPC	RI	ULA	seq
	soma	PC	extdsl	Rdesvio				despacho

- Primeira microinstrução:
  - cntrULA, ula1, ula2: calcula PC + 4
  - memória, memReg: busca instrução e coloca em RI
  - escPC: saída da ULA vai para o PC
  - seq: vai para a próxima microinstrução

Instruções lw e sw

Label	cntrULA	ula1	ula2	dstULA	Memória	memReg	escPC	prox
lwsw1	soma	rs	extend					despacho
lw2	soma	rs	extend		lerULA			seq
	soma	rs	extend		lerULA	escRT		busca
sw2	soma	rs	extend		escULA	lerRT		busca

- lwsw1:
  - calcula o endereço da memória, salta para lw2 ou sw2
- lw2:
  - lê a memória usando a saída da ULA como endereço, próxima
  - escreve no registrador, vai para a busca
- sw2:
  - escreve na memória, ULA com o endereço e o dado em rt, busca

Microprograma Completo

Label	cntrULA	ula1	ula2	dstULA	Memória	memReg	escPC	prox
busca	soma	PC	4		lerPC	RI	ULA	seq
	soma	PC		Rdesvio				despach1
lwsw1	soma	rs	extend					despach2
lw2	soma	rs	extend		lerULA			seq
	soma	rs	extend		lerULA	escRT		busca
sw2	soma	rs	extend		escULA	lerRT		busca
tipoR	func	rs	rt					seq
	func	rs	rt	rd				busca
beq1	sub	rs	rt				Rdesvio	busca
jump1							jump	busca

Exceções e Interrupções

- Exceções são mudanças no fluxo de execução devido a eventos inesperados gerados internamente ao processador
- Interrupções são mudanças no fluxo devido a eventos externos, tipicamente entrada e saída
- Exceções:
  - instrução inválida
  - overflow em operações aritmética
- Interrupções:
  - DMA
  - acesso ao barramento

Exceções e Interrupções

- A unidade de controle deve identificar a causa da exceção e armazená-la para posterior tratamento pelo sistema operacional
- Além disso, deve informar o endereço onde ocorreu a exceção
- MIPS utiliza dois registradores para isso:
  - EPC: endereço da instrução
  - Cause: indicação da causa da exceção