Universidade de Brasília
Departamento de Ciência da Computação

Aula 14
Implementação MIPS
MIPS Multiciclo – Unidade Operativa

MIPS Multiciclo

- Ideia: cada etapa de execução de uma instrução dura um ciclo de relógio
- Instruções podem ser executadas em um número diferente de etapas (ciclos)
- Ex.:
  - lw demora 5 ciclos
  - jump demora 3 ciclos
  - add demora 4 ciclos

MIPS Multiciclo

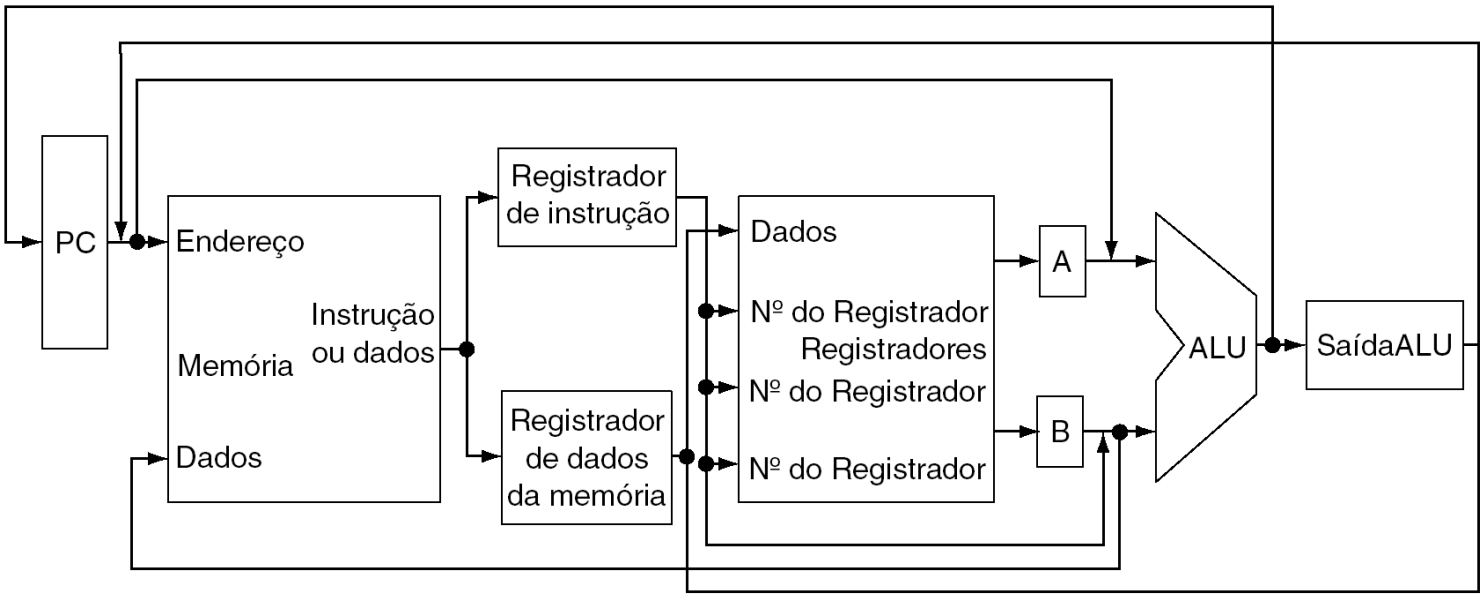
- Ciclo dimensionado de acordo com a etapa mais demorada
- Vamos reutilizar unidades funcionais
  - ULA usada também para calcular endereço e incrementar o PC
  - Memória usada para armazenar instrução E dados
- A organização da parte operativa pode ser re-estruturada em função destas características
- Sinais de controle não serão determinados diretamente pela instrução, e sim pela instrução+etapa
- Usaremos uma máquina de estado finito para controle

Unidade Operativa Multiciclo

- Dividir as instruções em etapas, cada uma durando um ciclo
  - – equilibrar a quantidade de trabalho a ser feito
  - – restringir cada ciclo para usar apenas uma vez cada unidade funcional
- No final de um ciclo
  - – armazenar os valores para uso em ciclos posteriores
  - – Logo, necessita introduzir registradores auxiliares “internos” adicionais

Unidade Operativa Multiciclo

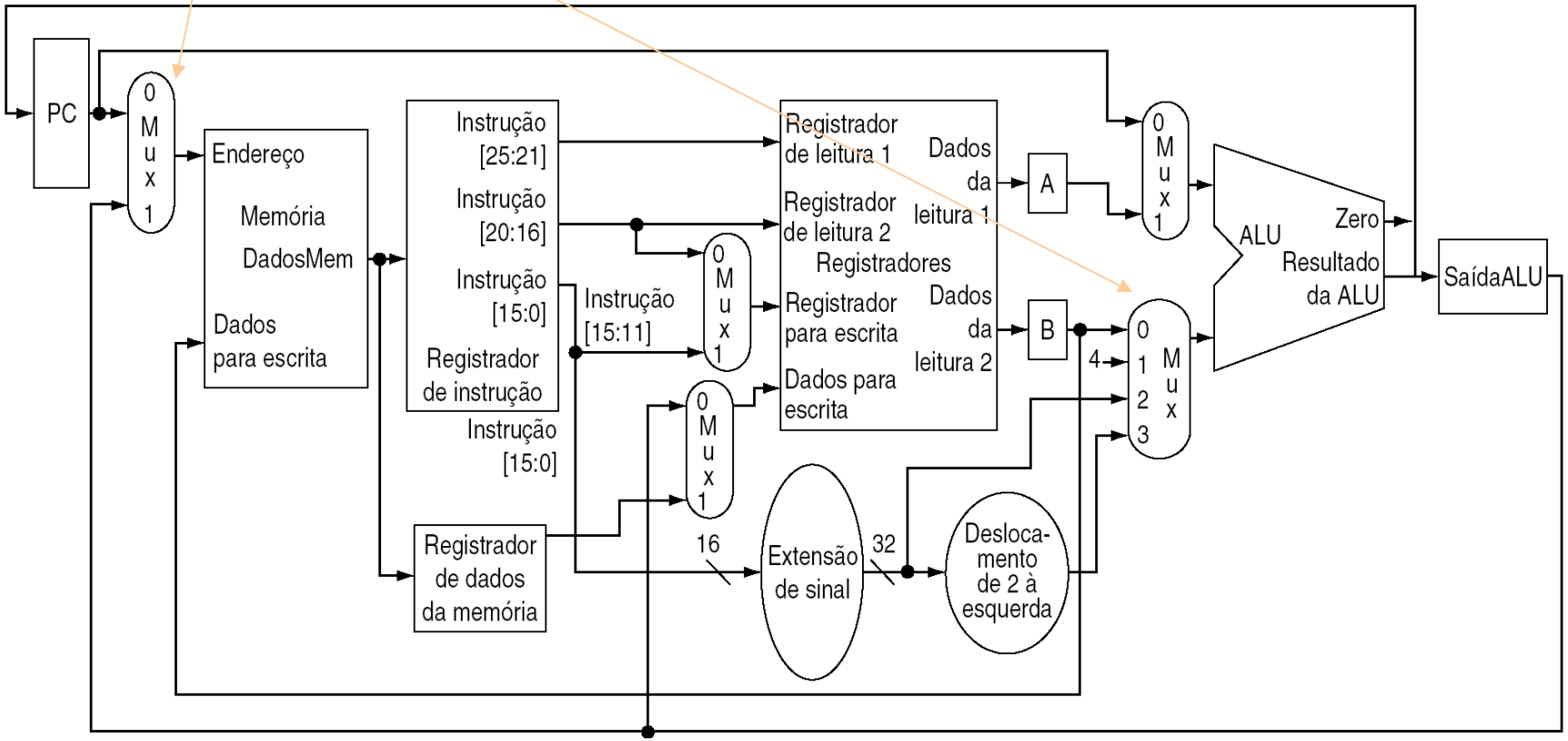
- Visão Abstrata:
  - Uma única ULA
  - Uma única Unidade de Memória
  - Registradores Auxiliares após unidades funcionais
    - onde colocá-los? Unidade combinacional de 1 ciclo e onde necessitar ser salvo. No nosso caso:



Obs.:
Dados utilizados pelas etapas subsequentes armazenados em registradores (elemento de estado invisível ao programador)
Dados utilizados pelas instruções subsequentes armazenados na Memória, Banco de Registradores (visível ao programador)

Unidade Operativa Multiciclo

- Detalhamento:
  - Todos os registradores auxiliares precisam manter o dado por 1 ciclo apenas (EXCEÇÃO IR)
    Logo não necessitam de pino de controle de escrita
  - Incluir e expandir Multiplexadores para poder reusar Unidades Funcionais



Caminho de Dados Multiciclo

- Definição das Linhas de Controle

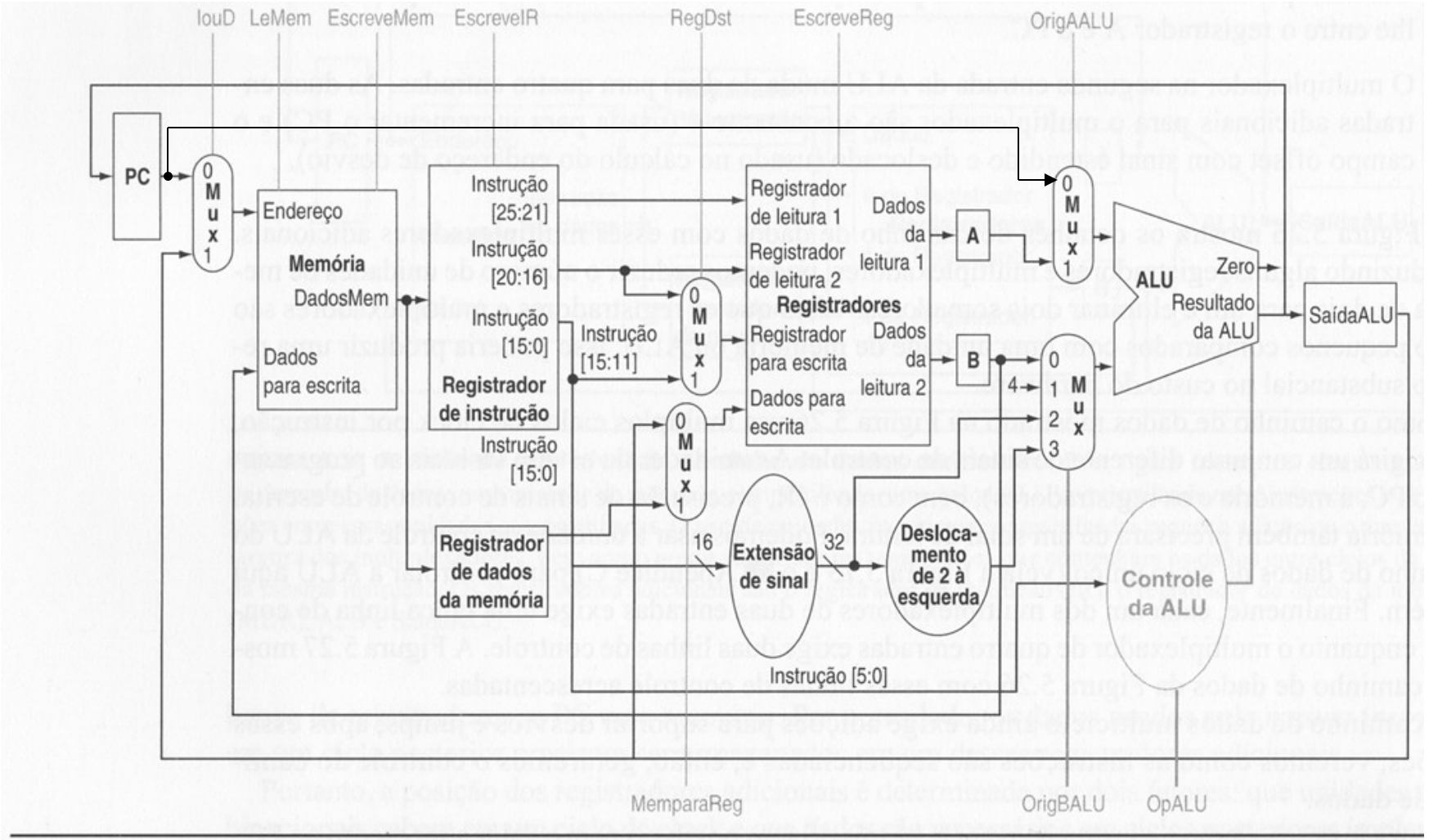
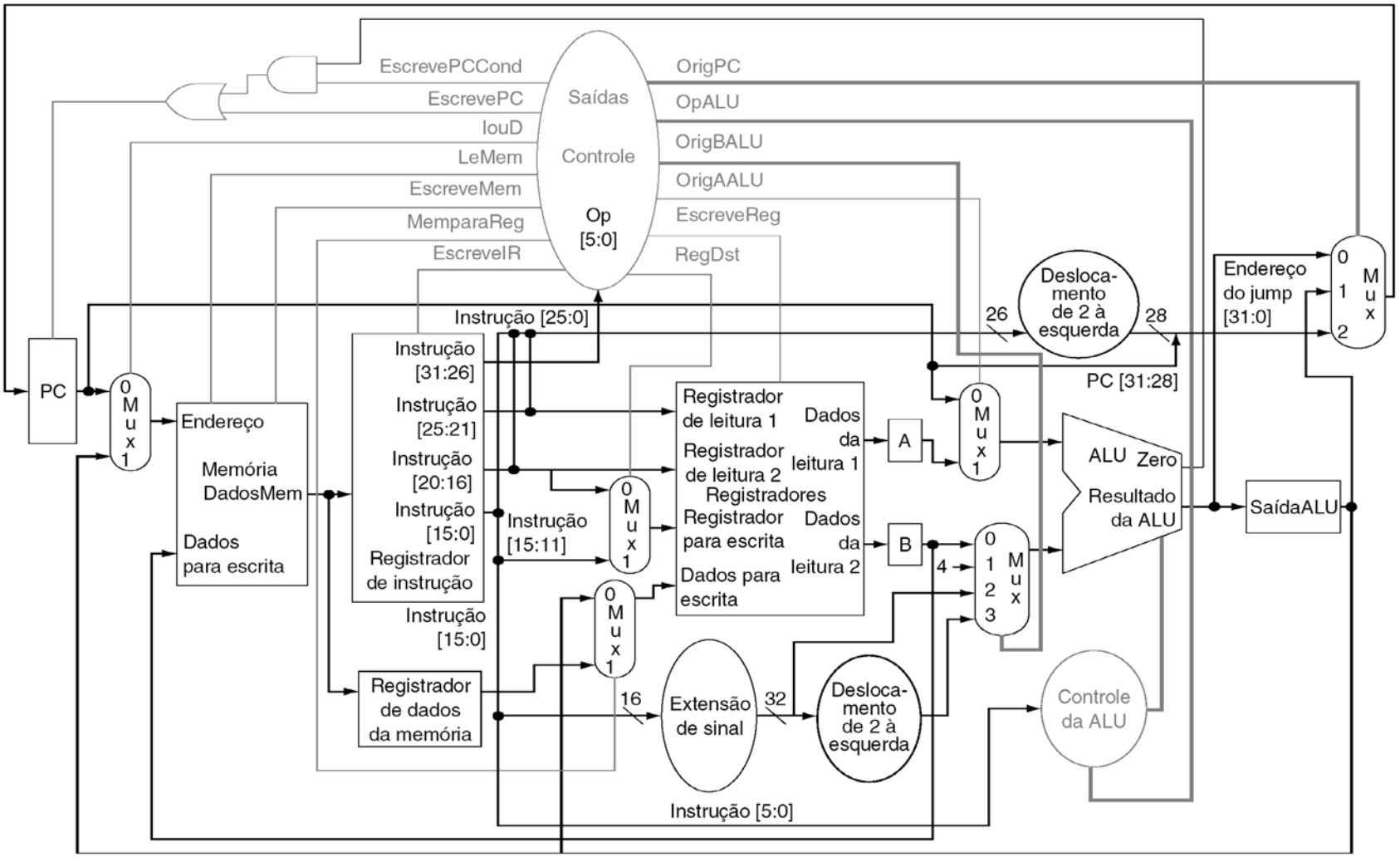


FIGURA 5.27 O caminho de dados multiciclo da Figura 5.26 com as linhas de controle indicadas.

Caminho de Dados Multiciclo

- Implementação do controle + controle do PC (PC+4, beq, jump)



Ações dos Sinais de Controle

Ações dos sinais de controle de 1 bit

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador de destino do banco de registradores para a entrada Registrador para escrita vem do campo rt.	O número do registrador de destino do banco de registradores para a entrada Registrador para escrita vem do campo rd.
EscreveReg	Nenhum.	O registrador de uso geral selecionado pelo número na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigAALU	O primeiro operando da ALU é o PC.	O primeiro operando da ALU vem do registrador A.
LeMem	Nenhum.	O conteúdo da memória no local especificado pela entrada Endereço é colocado na saída Dados da memória.
EscreveMem	Nenhum.	O conteúdo da memória no local especificado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado para a entrada Dados para escrita do banco de registradores vem de SaídaALU.	O valor enviado para a entrada Dados para escrita do banco de registradores vem do MDR.
IouD	O PC é usado para fornecer o endereço para a unidade de memória.	SaídaALU é usado para fornecer o endereço para a unidade de memória.
IR Write	Nenhum.	A saída da memória é escrita no IR.
EscrevePC	Nenhum.	O PC é escrito; a origem é controlada por OrigPC.
EscrevePCCond	Nenhum.	O PC é escrito se a saída Zero da ALU também estiver ativa.

Ações dos sinais de controle de 2 bits

Nome do sinal	Valor (binário)	Efeito
OpALU	00	A ALU realiza uma operação de adição.
	01	A ALU realiza uma operação de subtração.
	10	O campo funct da instrução determina a operação da ALU.
OrigBALU	00	A segunda entrada para a ALU vem do registrador B.
	01	A segunda entrada da ALU é a constante 4.
	10	A segunda entrada da ALU são os 16 bits menos significativos com sinal estendido do IR.
	11	A segunda entrada da ALU são os 16 bits menos significativos com sinal estendido do IR deslocados de 2 bits para a esquerda.
OrigPC	00	A saída da ALU (PC + 4) é enviada ao PC para escrita.
	01	O conteúdo de SaídaALU (o endereço de destino do desvio) é enviado ao PC para escrita.
	10	O endereço de destino do jump (IR[25:0]) deslocado de 2 bits para a esquerda e concatenado com PC + 4[31:28] é enviado ao PC para escrita.

Controle Multiciclo

- Exemplo: Instrução Tipo-R

poderia ser pensada na seguinte pseudo-instrução em linguagem de descrição de hardware Verilog:

Reg[Memory[PC][15:11]] <= Reg[Memory[PC][25:21]] op Reg[Memory[PC][20:16]] ;

Decomposta nas seguintes etapas usando “variáveis locais”:

IR <= Memory[PC];

A <= Reg[IR[25:21]];

B <= Reg[IR[20:16]];

saidaALU <= A op B;

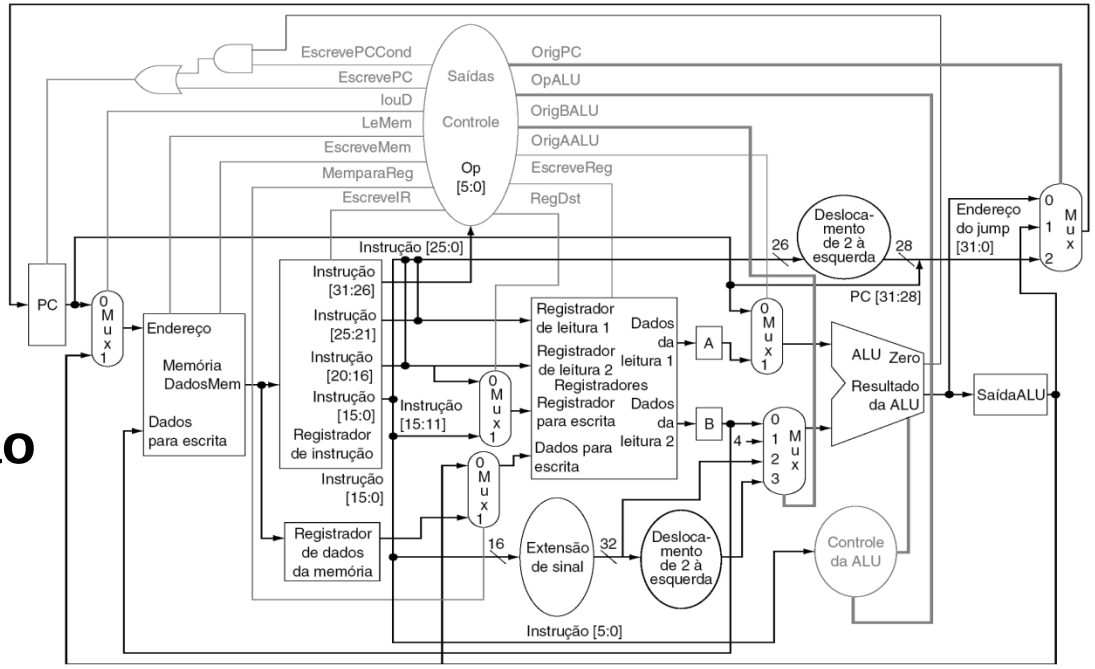
Reg[IR[15:11]] = saidaALU;

PC <= PC+4;

Controle Multiciclo: 5 Etapas

Considerando:

- 1 acesso à memória,
- 1 acesso ao banco de registradores
- 1 operação da ULA por ciclo.
- Projeto acionado por transição



1. Etapa de Busca da Instrução

IR <= Memoria[PC];

PC <= PC + 4;

Execução: Ler a Instrução da memória e incrementar PC

- Ativar os sinais de controle LeMem e Escreve IR
- Definir PC como origem do endereço: Colocar IouD em 0
- Incrementar PC+4: OrigAALU em 0 (PC para ALU), OrigBALU 01 (4 para ALU) e opALU 00 (soma).
- Armazenar o endereço de instrução incrementado em PC: origPC 00 e ativar EscrevePC

Controle Multicliclo...

2. Etapa de Decodificação da Instrução e busca de operandos

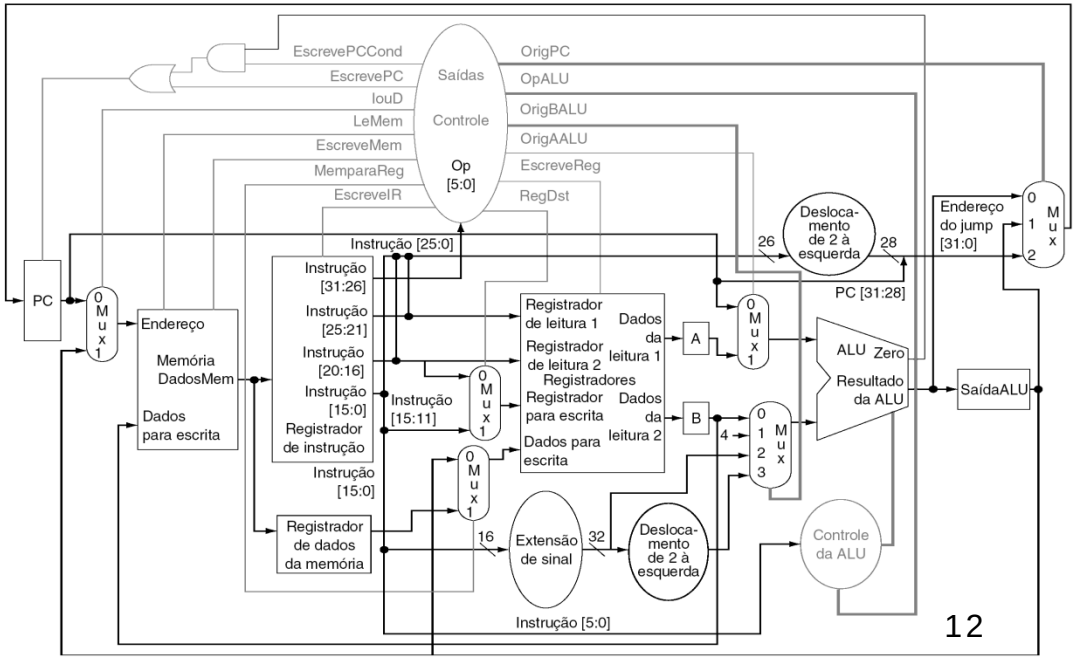
A <= Reg[IR[25:21]];

B <= Reg[IR[20:16]];

SaidaALU <= PC + (extensão-sinal(IR[15:0]) << 2);

Execução: Como não sabemos qual a instrução ainda, podemos fazer ações gerais que não causem problemas depois!

- Acessar o banco de registradores e ler *rs* e *rt* e armazenar em A e B.
- Cálculo prévio do endereço de desvio e coloca em SaidaALU: OrigAALU em 0 (PC para ALU), OrigBALU em 11 (offset ext sinal) e OpALU 00 (soma).



Controle Multiciclo ...

3. Etapa de Execução, cálculo do endereço de memória ou conclusão de desvio

Referência à memória (lw,sw):

SaidaULA = A + extensão-sinal(IR[15-0])

Execução: Calcular o endereço

- OrigAALU em 1 (A na ALU), OrigBALU em 10 (saída ext-sinal na ALU), opALU 00 (soma)

Instrução Tipo-R:

SaidaULA <= A op B

Execução: os valores de A e B estão prontos!

- OrigAALU em 1 (A na ALU), OrigBALU em 00 (B na ALU), opALU 10 (lógico-aritmética pelo campo funct)

beq:

if (A==B) PC <= SaidaALU

Execução: o endereço de desvio já está pronto.

- OrigAALU em 1 (A na ALU),

OrigBALU em 00 (B na ALU), opALU 01 (subtrai)

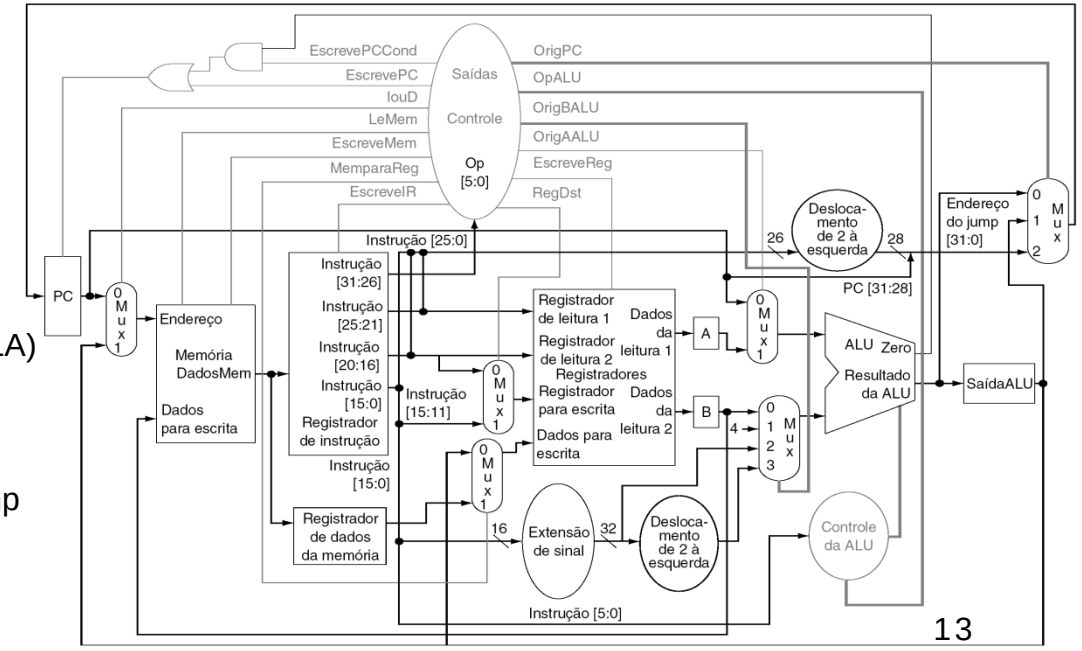
- Ativar sinal EscrevePCCond
- Colocar OrigPCem 01 (valor do PC vem da ULA)

Jump:

PC <= {PC[31:28],(IR[25:0]],2'b00)}

Execução: PC é substituído pelo endereço de Jump

- Colocar OriPC em 10
- Ativar EscrevePC



Controle Multiciclo ...

4. Etapa de acesso à memória ou conclusão de instrucão Tipo-R

Referência à memória (lw, sw):

MDR <= Memoria[SaidaALU]; #para load ou

Memória[saidaALU] <= B # para store

Execução: Em qualquer dos casos o endereço já está na SaidaALU

- IouD colocado em 1 (Endereço da ALU)
- LeMem ou EscreveMem deve ser ativado

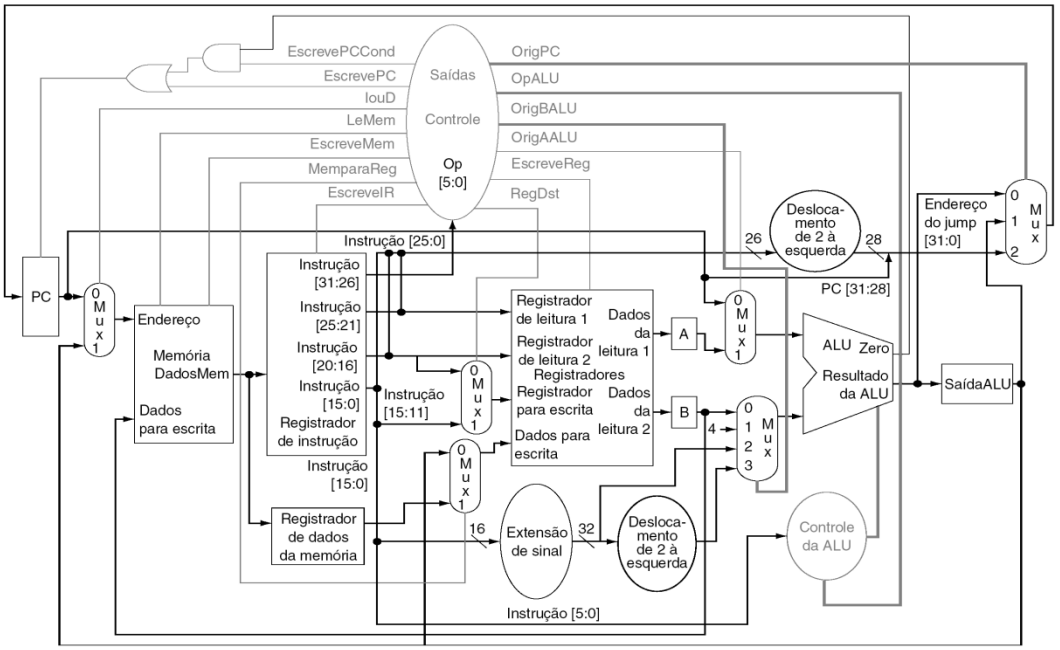
Instrução Tipo-R:

Reg[IR[15:11]] <= SaidaALU

Execução: Colocar o resultado no reg destino

- Colocar RegDST em 1 (usar rd como dest)
- Ativar EscreveReg
- Colocar MemparaReg em 0

(saída da ALU na entrada do banco de reg)



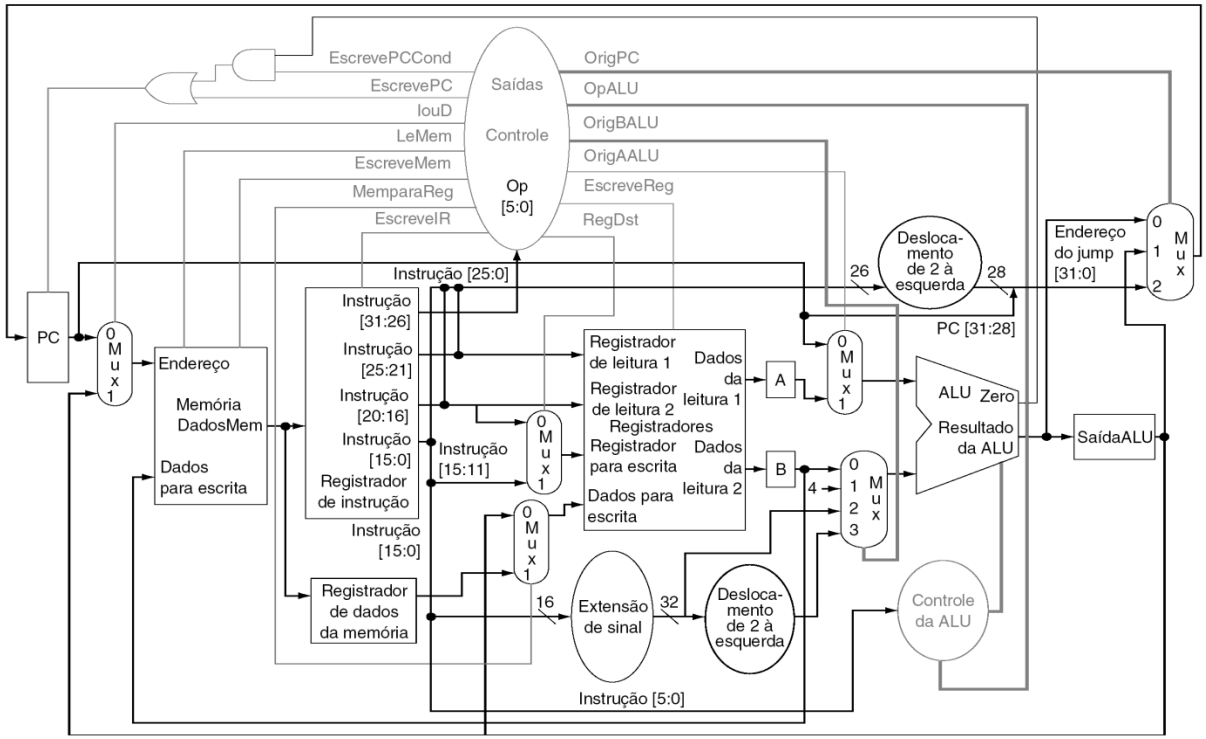
Controle Multiciclo ...

5. Conclusão da Leitura da Memória (lw)

Reg[IR[20:16]] <= MDR;

Execução: Escrever o dado do MDR no banco de reg

- Colocar MemparaReg em 1 (resultado da memória)
- Ativar o EscreveReg
- Colocar RegDst em 0 (campo rt é o reg destino)



Resumo Controle Multiciclo

Etapa	Ação para instruções tipo R	Ação para instruções de acesso à memória	Ação para desvios	Ação para jumps
Busca da instrução	IR <= Memória[PC] PC <= PC + 4			
Decodificação da instrução e busca dos registradores	A <= Reg[IR[25:21]] B <= Reg[IR[20:16]] SaídaALU <= PC + (estende-sinal (IR[15:0]) << 2)			
Execução, cálculo do endereço, conclusão do desvio/jump	SaídaALU <= A op B	SaídaALU <= A + estende-sinal (IR[15:0])	if (A == B) PC <= SaídaALU	PC <= {PC [31:28], (IR[25:0]],2'b00)}
Acesso à memória ou conclusão de instrução tipo R	Reg[IR[15:11]]<= SaídaALU	Load: MDR <= Memória[SaídaALU] ou Store: Memória [SaídaALU] <= B		
Conclusão da leitura da memória		Load: Reg[IR[20:16]] <= MDR		

FIGURA 5.30 Resumo das etapas realizadas para executar qualquer classe de instrução. As instruções levam de três a cinco etapas de execu-

Ex.: Considerando um workload de 25% loads, 10% stores, 11% branches, 2% jumps e 52% operações com ALU. Qual a CPI média implementada?

CPI=0.25x5+0.1x4+0.52x4+0.11x3+0.02x3= 4.12

Lembrando que no pior caso: CPI=5 !