Universidade de Brasília
Departamento de Ciência da Computação

Aula 14
Implementação RISC-V
Multiciclo – Unidade Operativa

RISC-V 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
  - add demora 4 ciclos
  - beq demora 3 ciclos

RISC-V Multiciclo

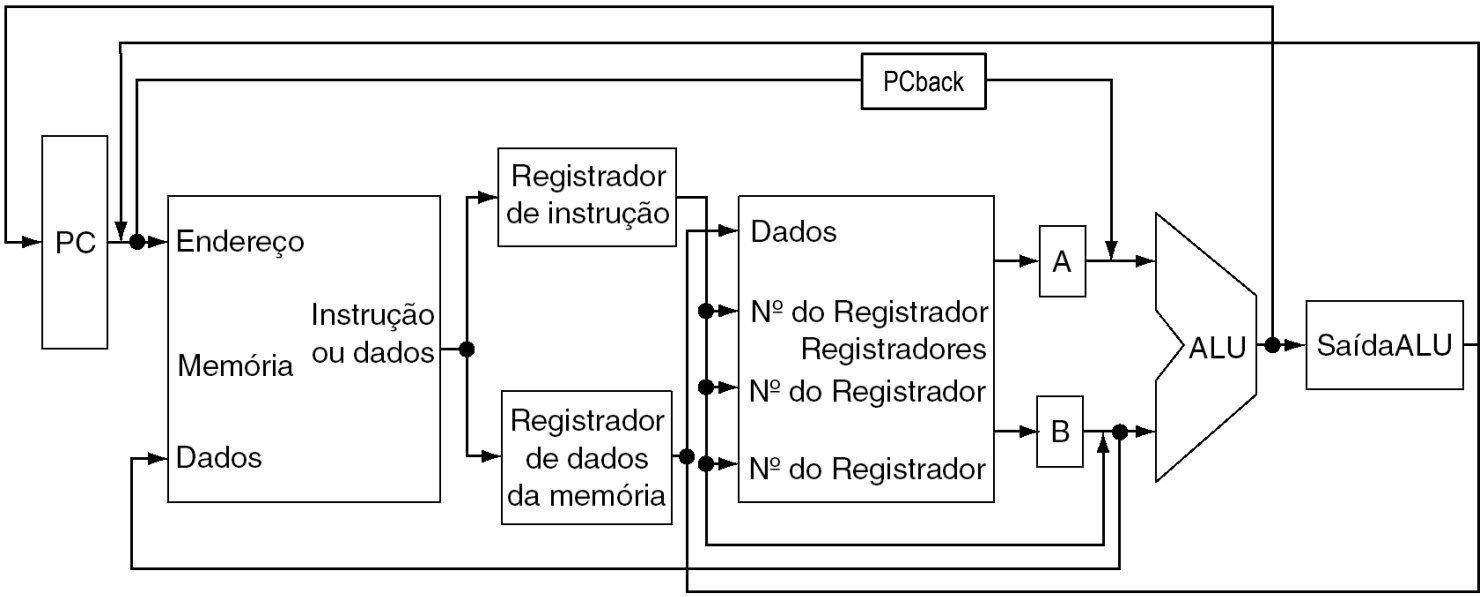
- Ciclo dimensionado de acordo com a etapa mais demorada
- Vamos reutilizar unidades funcionais
  - ULA usada também para calcular endereço e incrementar o PC
  - Memória usada para armazenar instruções E dados
- A organização da parte operativa pode ser reestruturada em função destas características
- 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

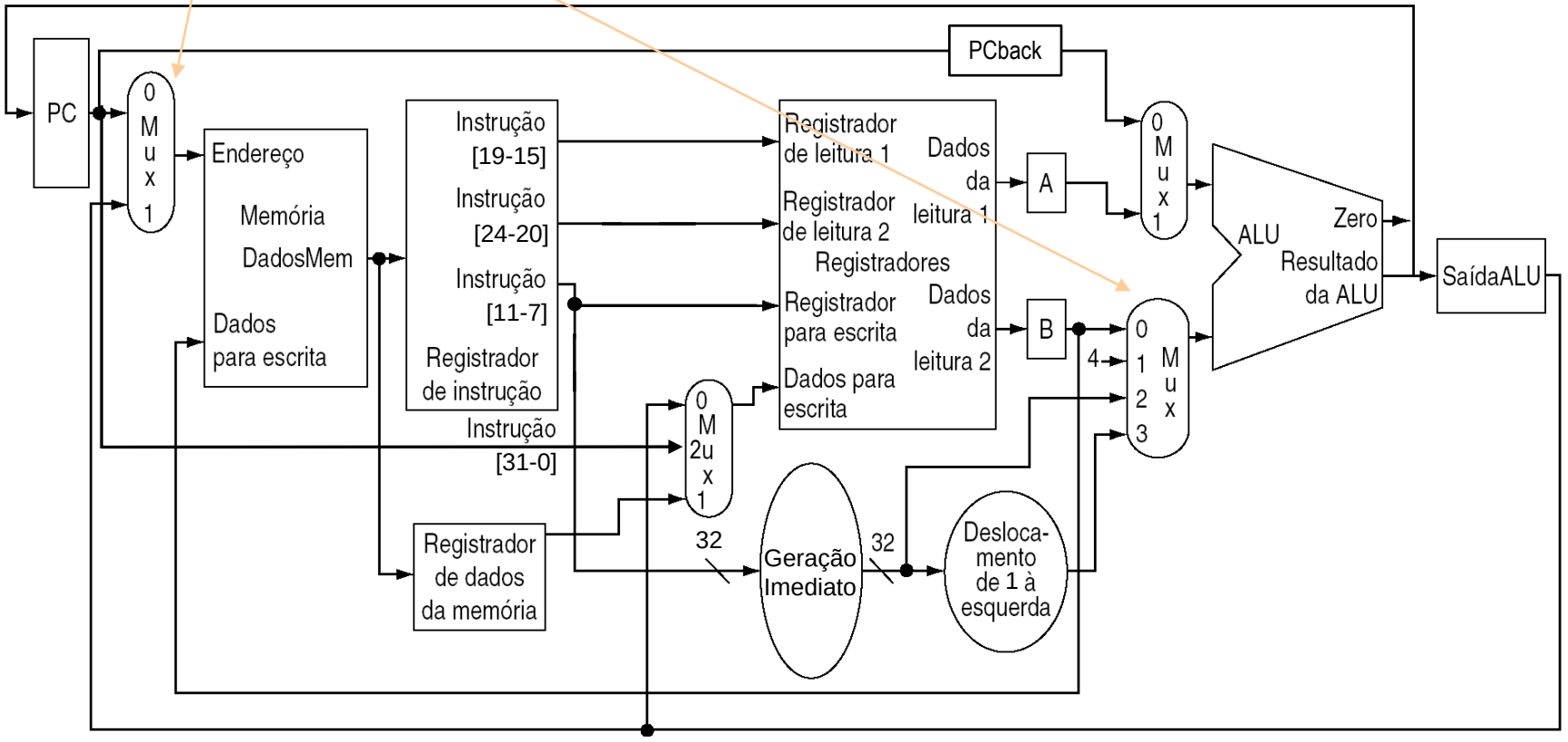
- Visão Abstrata:
  - Uma única ULA
  - Uma única Unidade de Memória
  - Registradores Auxiliares após unidades funcionais
    - onde colocá-los? Unidade combinacional de 1 ciclo e onde necessitar ser salvo. No nosso caso:



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)

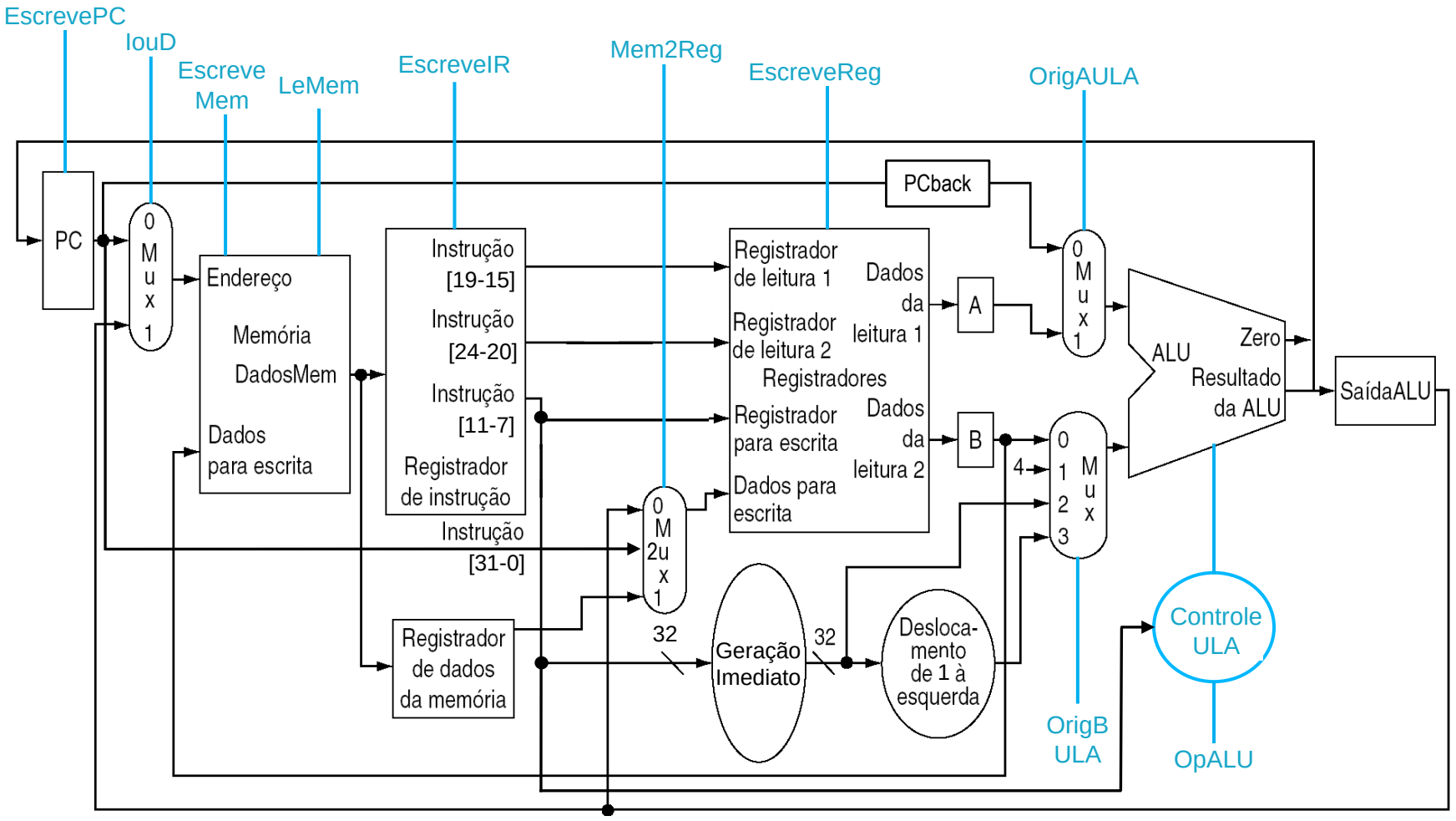
Unidade Operativa Multiciclo

- 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



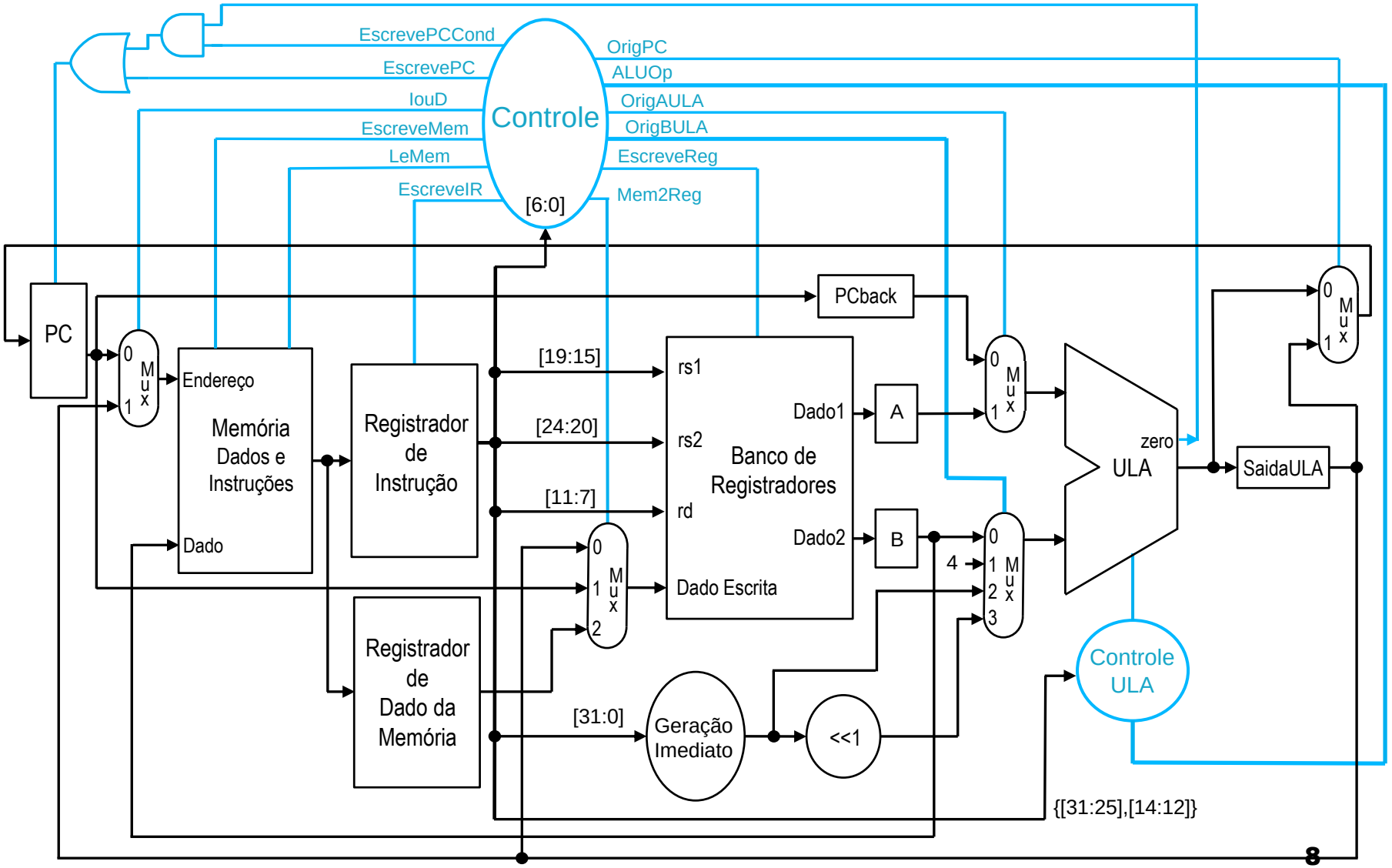
Caminho de Dados Multiciclo

- Definição das Linhas de Controle



Caminho de Dados Multiciclo

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



Ações dos Sinais de Controle

Sinal	Valor	Efeito
EscreveReg	0	Não permite a escrita no Banco de Registradores
	1	O Registrador de Destino é escrito com o Dado de Escrita
OpALU	00	Soma
	01	Subtração
	10	Os campos {Funct7,Funct3} definem a operação
OrigAULA	0	Primeiro argumento da ULA é o registrador PC
	1	Primeiro argumento da ULA é o registrador A
OrigBULA	00	Segundo argumento da ULA é o registrador B
	01	Segundo argumento da ULA é o número 4
	10	Segundo argumento da ULA é o Imediato
	11	Segundo argumento da ULA é o Imediato deslocado 1 bit
LeMem	0	Desabilita a leitura da Memória
	1	Habilita a leitura da Memória
EscreveMem	0	Desabilita a escrita na Memória
	1	Habilita a escrita na Memória
Mem2Reg	00	O valor escrito no Banco de Registradores vem da SaídaALU
	01	O valor escrito no Banco de Registradores vem do PC
	10	O valor escrito no Banco de Registradores vem do MDR
IouD	0	O PC é usado para endereçar a memória
	1	O registrador SaídaALU é usado para endereçar a memória
EscreveIR	0	Desabilita a escrita no IR
	1	Habilita a escrita no IR
EscrevePC	0	Desabilita a escrita incondicional no PC
	1	Habilita a escrita incondicional no PC
EscrevePCCond	0	Desabilita a escrita condicional no PC
	1	Habilita a escrita condicional no PC
OrigPC	0	O valor escrito no PC é o resultado da ULA
	1	O valor escrito no PC é o registrador SaídaULA

Controle Multiciclo

- Exemplo: Instrução Lógico-Aritmética

Poderia ser pensada na seguinte pseudoinstrução em Verilog para o Uniciclo:

Reg[Memory[PC][11:7]] <= Reg[Memory[PC][19:15]] ***op*** *Reg[Memory[PC][24:20]];*

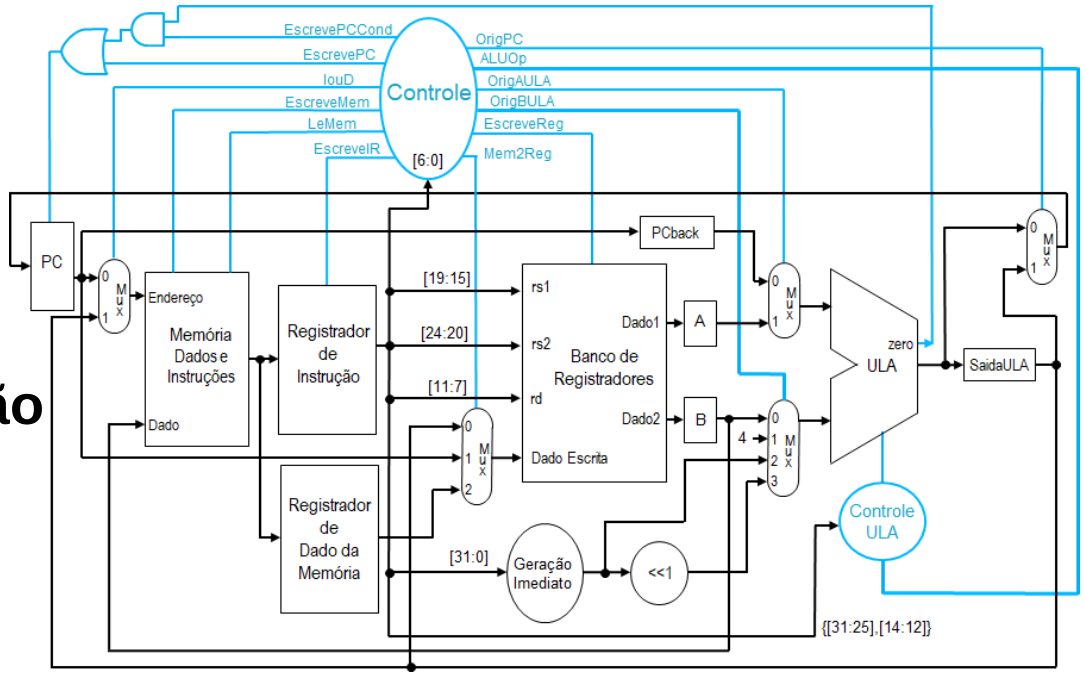
No multiciclo é decomposta nas seguintes etapas usando “variáveis locais”:

```
IR <= Memory[PC];              //Busca da Instrução
A <= Reg[IR[19:15]];           //Leitura dos Registradores
B <= Reg[IR[24:20]];
SaidaALU <= A op B;            //Cálculo com a ULA
Reg[IR[11:7]] = SaidaALU;      //Escrita do Resultado
PC <= PC+4;                    //Próxima instrução
```

Controle Multiciclo: 5 Etapas

Considerando (por ciclo):

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



1. Etapa de Busca da Instrução

IR <= Memoria[PC];

PCback <= PC;

PC <= PC+4;

Execução: Ler a Instrução da memória

- Ativar os sinais de controle LeMem e EscreveIR
- Definir PC como origem do endereço: Colocar IouD em 0
- Incrementar PC+4: OrigAULA em 0, OrigBULA 01, e ALUOp 00 (soma).
- Armazenar o endereço de instrução incrementado em PC: OrigPC 0 e ativar EscrevePC

Controle Multicliclo...

2. Etapa de Decodificação da Instrução e busca de operandos

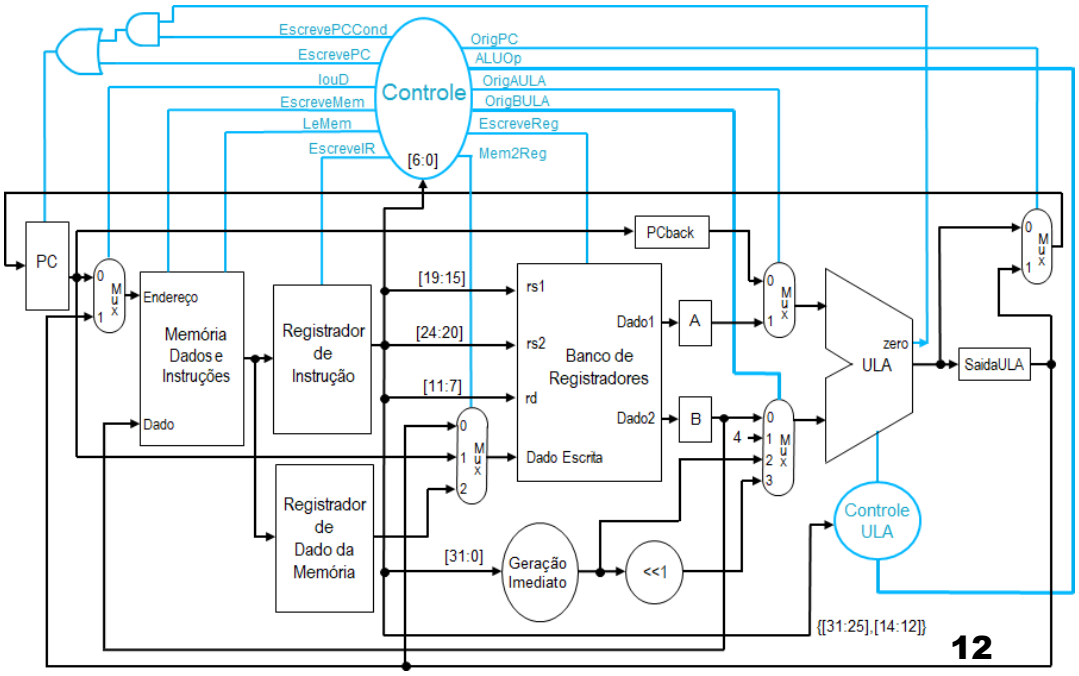
A <= Reg[IR[19:15]];

B <= Reg[IR[24:20]];

SaidaALU <= PCback + (Imediato<<1);

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

- Acessar o banco de registradores, ler *rs1* e *rs2* e armazenar em A e B.
- Cálculo prévio do endereço de desvio e coloca em SaidaALU: OrigAULA em 0 (PCback para ALU), OrigBULA em 11 (Imediato<<1) e ALUOp 00 (soma).



Controle Multiciclo ...

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

Referência à memória (lw,sw):

SaidaULA = A + imediato

Execução: Calcular o endereço

- OrigAULA em 1 (A), OrigBULA em 10 (imediato), ALUOp 00 (soma)

Tipo-R:

SaidaULA <= A op B

Execução: Operar com os valores A e B

- OrigALU em 1 (A), OrigBULA em 00 (B), ALUOp 10 (operação definida pelo campos {funct7,funct3})

beq:

if (A==B) PC <= SaidaULA

Execução: o endereço de desvio já está pronto.

- OrigAULA em 1 (A), OrigBALU em 00 (B), ALUOp 01 (subtração)
- Colocar OrigPC em 1 (SaidaULA)
- Ativar sinal EscrevePCCond

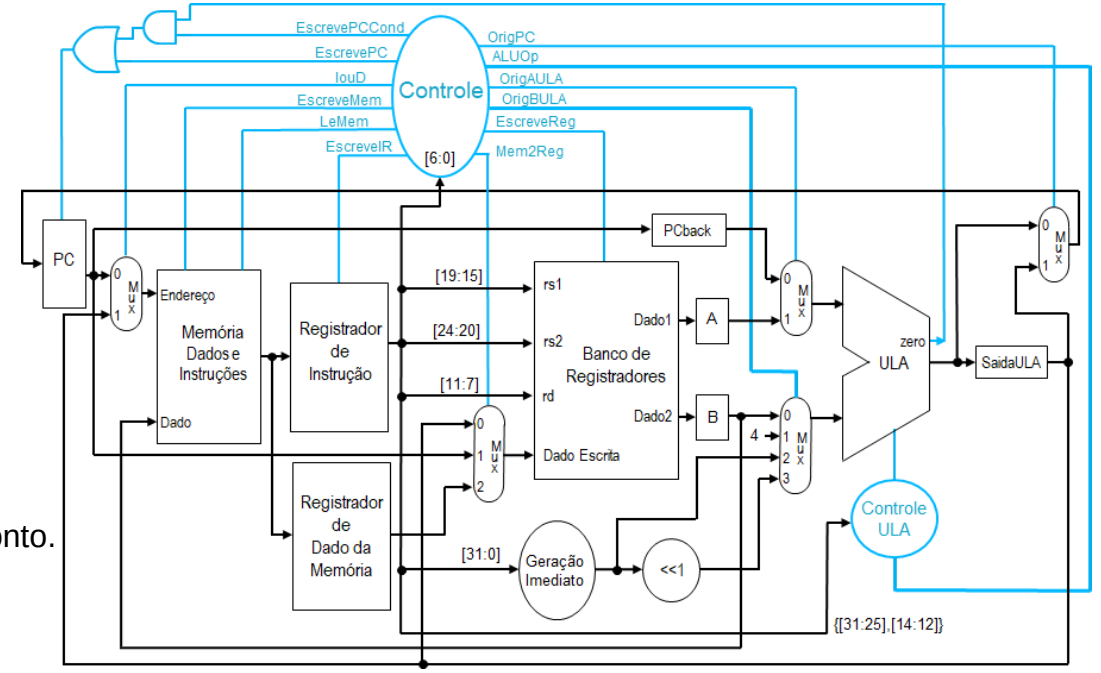
jal:

R[rd] <= PC+4 (já calculado em PC)

PC <= SaidaULA

Execução: salvar rd e endereço de desvio já pronto.

- Colocar OrigPC em 1 (SaidaULA)
- Mem2Reg em 01 (PC)
- Ativar EscrevePC e EscreveReg



Controle Multiciclo ...

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

Referência à memória (lw, sw):

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

Memória[SaidaULA] <= B; # para store

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

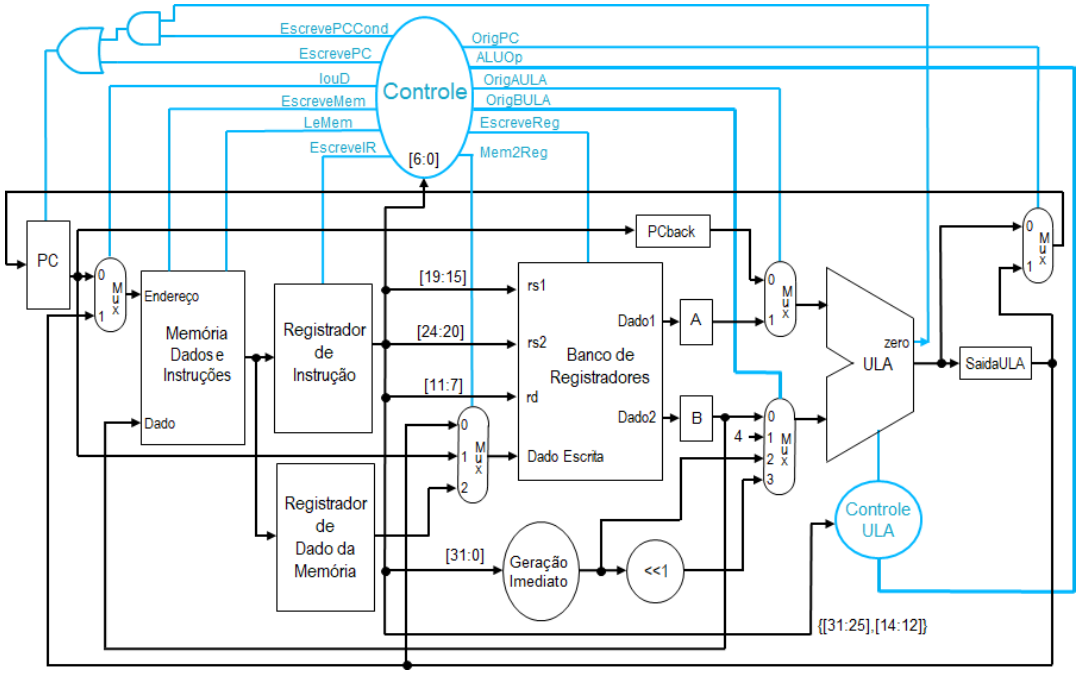
- IouD colocado em 1 (Endereço da ULA)
- LeMem ou EscreveMem deve ser ativado

Instrução Tipo-R:

Reg[IR[11:7]] <= SaidaULA

Execução: Escrever o resultado no rd

- Mem2Reg em 00 (SaidaULA)
- Ativar EscreveReg



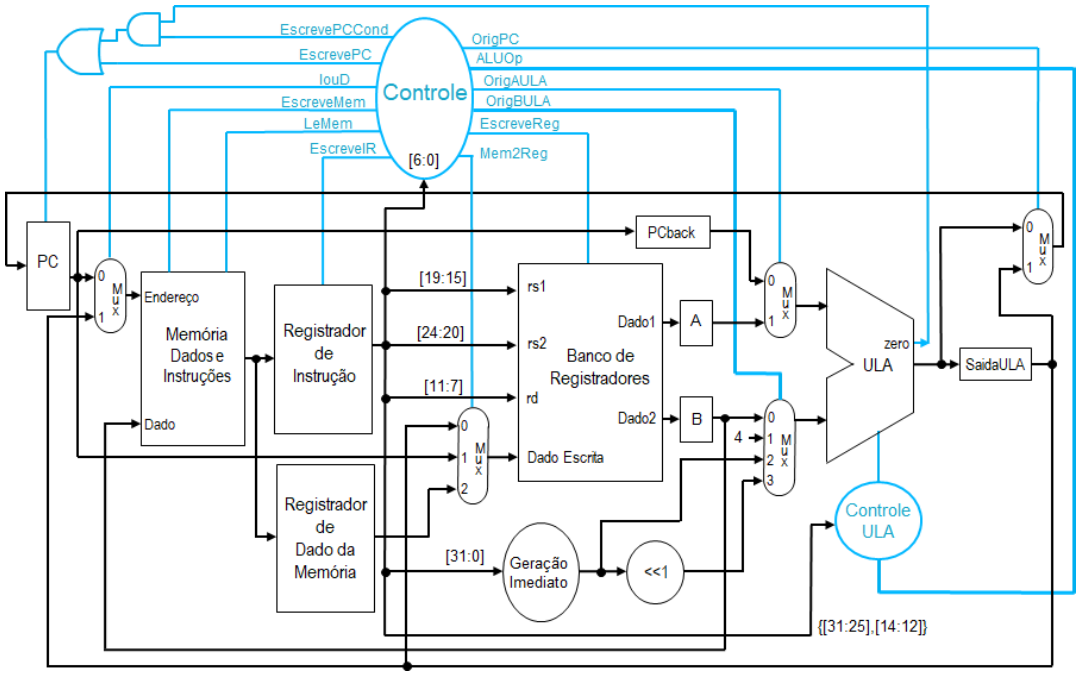
Controle Multiciclo ...

5. Conclusão da Leitura da Memória (lw)

Reg[IR[11:7]] <= MDR;

Execução: Escrever o dado do MDR no banco de registradores

- Colocar Mem2Reg em 10 (Dado da memória)
- Ativar o EscreveReg



Resumo Controle Multiciclo

Etapa	Tipo-R	Acesso à Memória	Desvios Condicionais	Desvios Incondicionais
Busca da Instrução	IR<=Mem[PC] PCback<=PC PC<=PC+4			
Decodificação, Leitura dos registradores	A<=Reg[IR[19:15]] B<=Reg[IR[24:20]] SaidaULA<=PC+imm<<1			
Execução, cálculo do endereço	SaidaULA<=A *op* B	SaidaULA<=A+imm	Se (A==B) PC<=SaidaULA	Reg[IR[11:7]]<=PC+4 PC<=SaidaULA
Acesso à memória, conclusão tipo-R	Reg[IR[11:7]]<=SaidaULA	Load: MDR<=Mem[SaidaULA] Store: Mem[SaidaULA]<=B		
Conclusão lw		Load: Reg[IR[11:7]]<=MDR		

Ex.: Considerando um workload de 25% loads, 10% stores, 11% branches, 2% jals e 52% operações Tipo-R. Qual a CPI média implementada?

CPI = 0.25×5 + 0.1×4 + 0.52×4 + 0.11×3 + 0.02×3 = 4.12

Lembrando que no pior caso: CPI=5 !