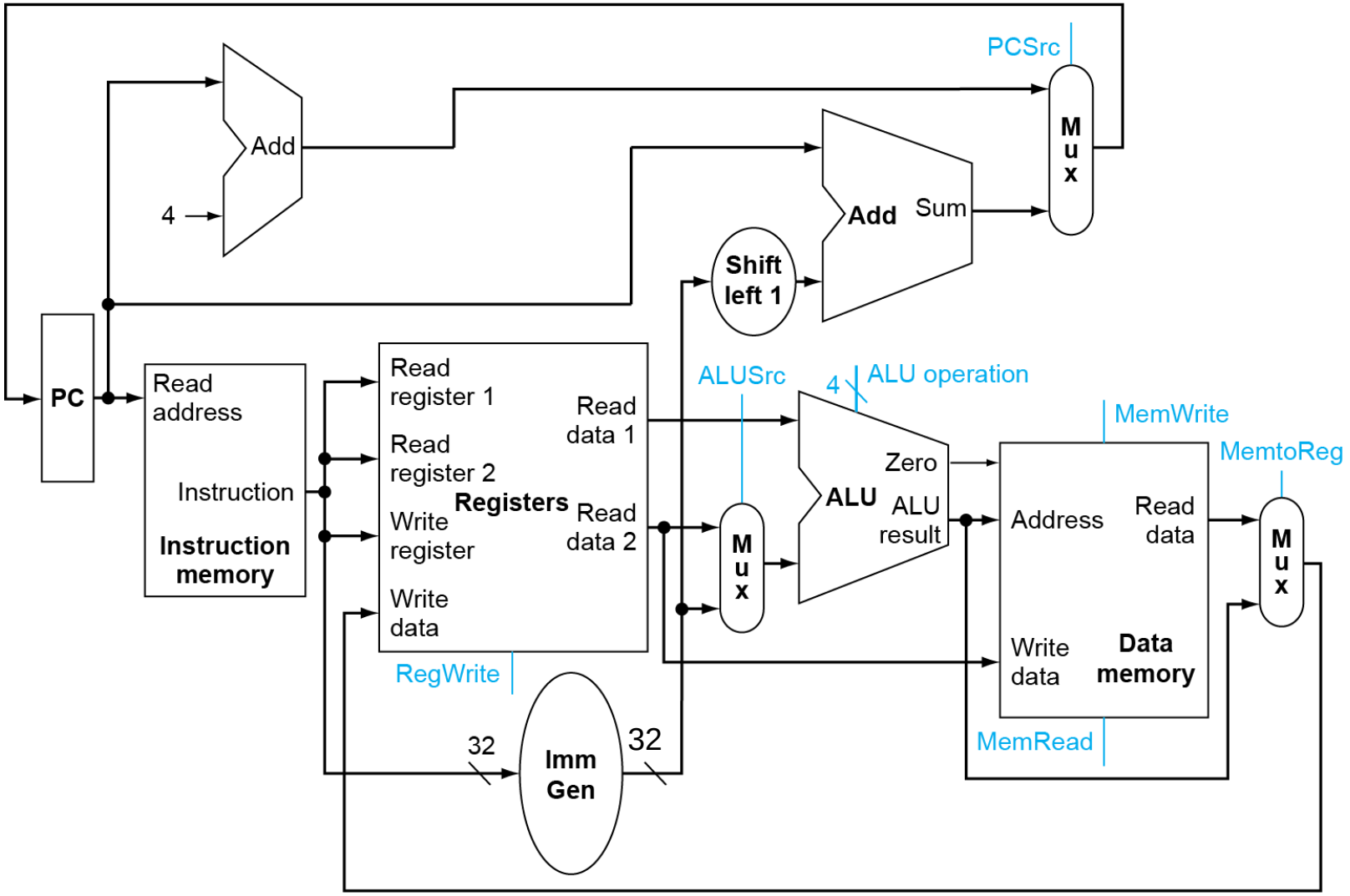
Universidade de Brasília
Departamento de Ciência da Computação

Aula 13
Implementação RISC-V
Uniciclo – Unidade de Controle

RISC-V Uniciclo

- Foram desenvolvidas, na verdade, três tipos de unidades operativas:
  - instruções de operações lógicas e aritméticas (*add*, *sub*, *and, or, slt*)
  - instruções de acesso a memória (*lw* e *sw)*
  - instruções de desvio condicionais (*beq)*
- Na fase de projeto as vezes precisamos replicar recursos.
- Na via de dados mais simples deve-se propor executar as instruções em um único período do *clock.*
- **Isto quer dizer que nenhum dos recursos pode ser usado mais de uma vez por instrução.**

Caminho de Dados UNICICLO



Bloco de Controle do RISC-V

- Cada operação requer a utilização adequada dos recursos
- Ex: *add t0, 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 (t0) 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 t0)

Exemplo de projeto de controle hierárquico: **Controle da ULA**

- As operações da ULA são controladas por um código de 4 bits:

Linhas de controle da ALU	Função
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than

- As instruções **lw** e **sw** utilizam a operação de **soma** da ULA
- As instruções **add,sub,and,or,slt** utilizam uma das **5 operações**
- A instrução **beq** utiliza a **subtração** da ULA

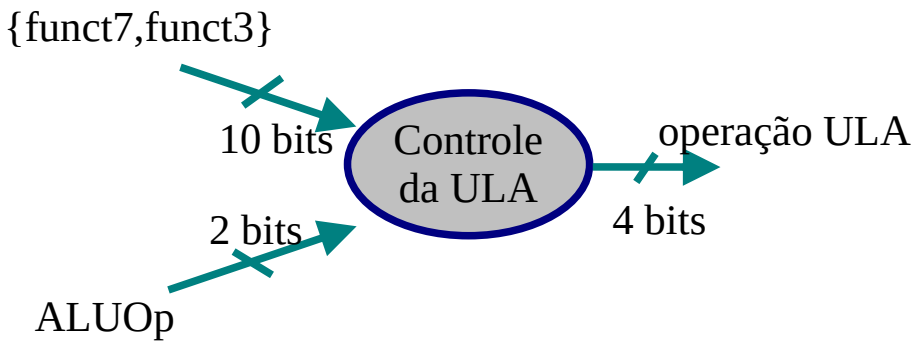
Identificação da Operação

- Os Opcodes definem as operações:

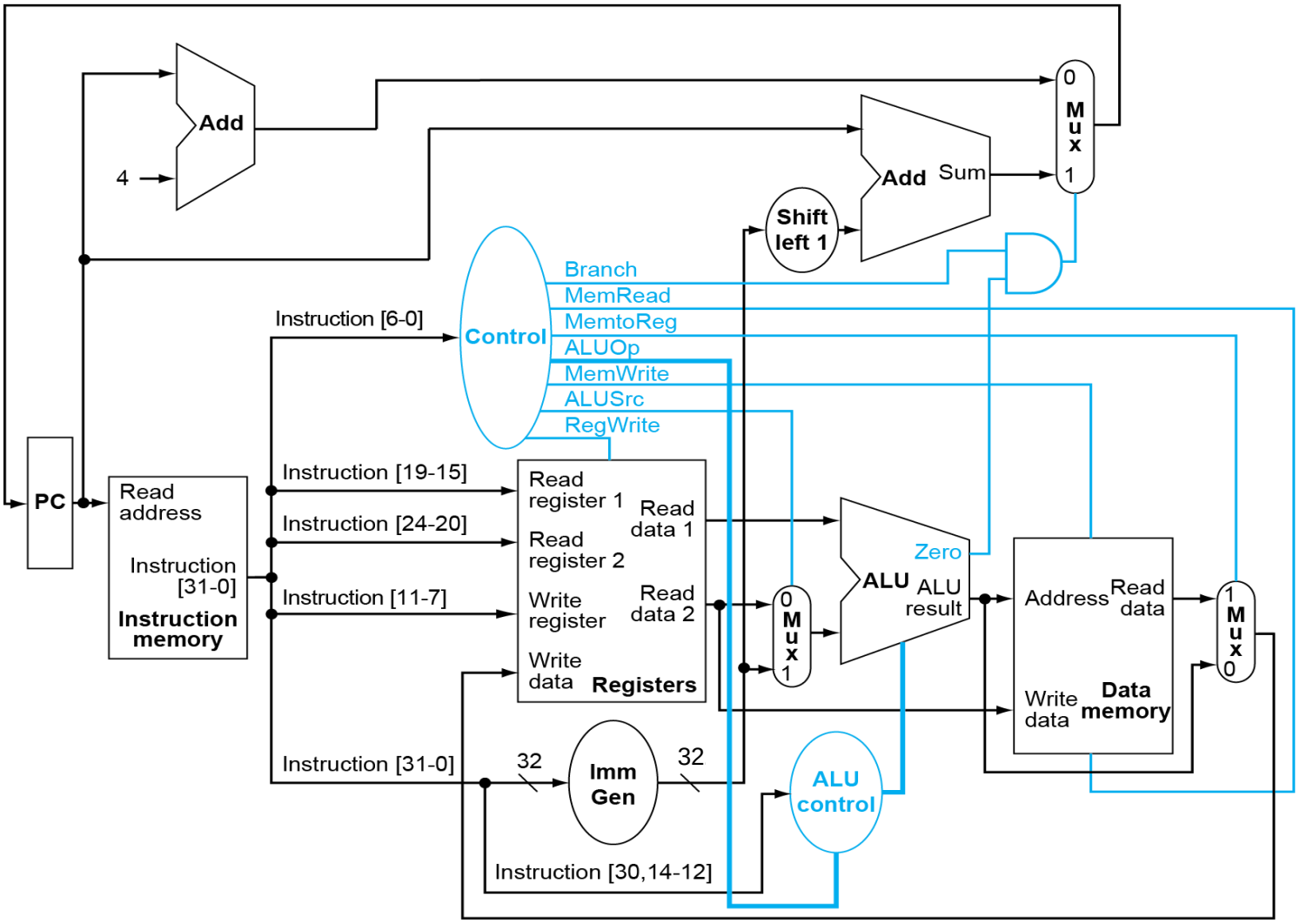
  lw(0x23), sw(0x23), beq(0x63), Tipo-R (0x33)

- No Tipo-R os campos *funct7 e funct3*, especificam o tipo de operação aritmética/lógica.

- Vamos definir 2 bits para identificar cada uma das categorias de instruções (ALUop):
  - 00 acesso à memória (lw, sw) [**add**]
  - 01 desvio (beq) [**sub**]
  - 10 lógico-aritméticas ([**add, sub, and, or, slt**])



Lógica de Controle da ULA



Circuito de Controle da ULA

- xxx indica irrelevâncias (*don't cares*)

OPcode	ALUOp	Operação	Funct7 Funct3	Operação da ULA	ALU Control
lw	00	Load	xxxxxxx xxx	add	0010
sw	00	Store	xxxxxxx xxx	add	0010
beq	01	Beq	xxxxxxx xxx	sub	0110
Tipo-R	10	Add	0000000 000	add	0010
Tipo-R	10	Sub	0100000 000	sub	0110
Tipo-R	10	And	0000000 111	and	0000
Tipo-R	10	Or	0000000 110	or	0001
Tipo-R	10	Slt	0000000 010	slt	0111

Controle da ULA

Com base nas definições anteriores podemos montar a Tabela Verdade do circuito que gera os 4 bits de controle da ULA

ALUOp		**Funct 7**							**Funct 3**			**ALU Control**
ALU Op1	ALU Op0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operação
0	0	x	x	x	x	x	x	x	x	x	x	0 0 1 0
0	1	x	x	x	x	x	x	x	x	x	x	0 1 1 0
1	0	0	0	0	0	0	0	0	0	0	0	0 0 1 0
1	0	0	1	0	0	0	0	0	0	0	0	0 1 1 0
1	0	0	0	0	0	0	0	0	1	1	1	0 0 0 0
1	0	0	0	0	0	0	0	0	1	1	0	0 0 0 1
1	0	0	0	0	0	0	0	0	0	1	0	0 1 1 1

Exercício: Projetar o circuito digital com portas lógicas

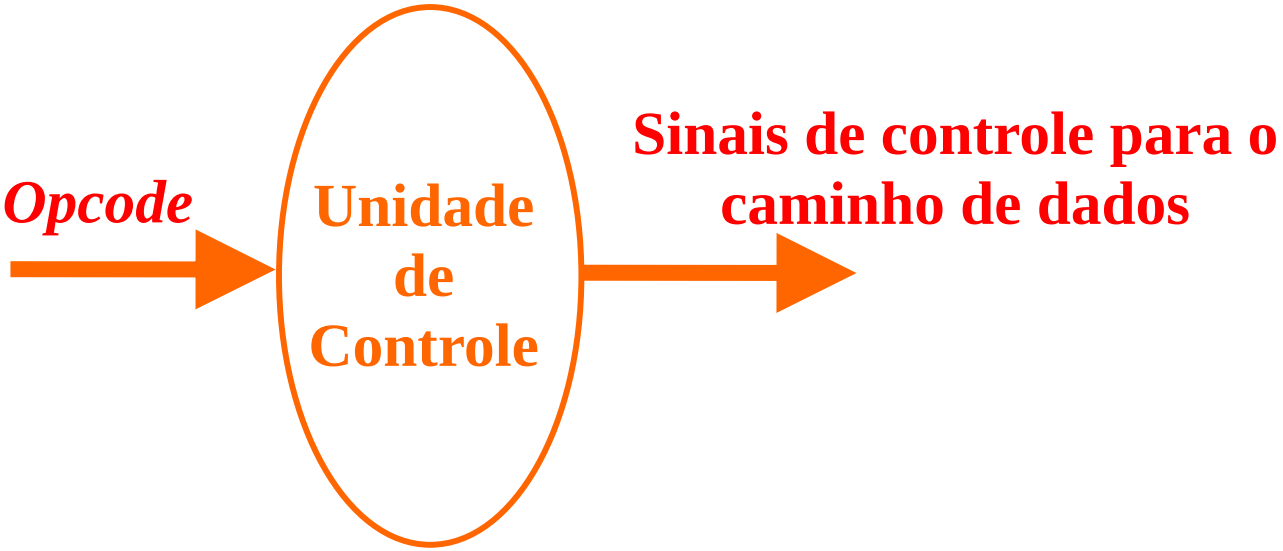
Controle da ULA

```
module ALUControl (
            input   [9:0] Funct10,
            input   [1:0] ALUOp,
            output [3:0] ALUCtrl
            );

always @(*)
  case (ALUOp)
       2'b00:  ALUCtrl  = OPADD;
       2'b01: ALUCtrl  = OPSUB;
       2'b10:
          case (Funct10)
             FUNADD:  ALUCtrl  = OPADD;
             FUNSUB: ALUCtrl  = OPSUB;
             FUNAND: ALUCtrl  = OPAND;
             FUNOR:   ALUCtrl  = OPOR;
             FUNSLT:  ALUCtrl  = OPSLT;
             default: ALUCtrl = 4'b0000;
          endcase
      default: ALUCtrl = 4'b0000;
  endcase

endmodule
```

```
parameter
            OPADD = 4'b0010,
            OPSUB = 4'b0110,
            OPAND = 4'b0000,
            OPOR   = 4'b0001,
            OPSLT  = 4'b0111,
            FUNADD = 10'b0000000_000,
            FUNSUB = 10'b0100000_000,
            FUNAND = 10'b0000000_111,
            FUNOR   = 10'b0000000_110,
            FUNSLT  = 10'b0000000_010;
```

Unidade de Controle Uniciclo

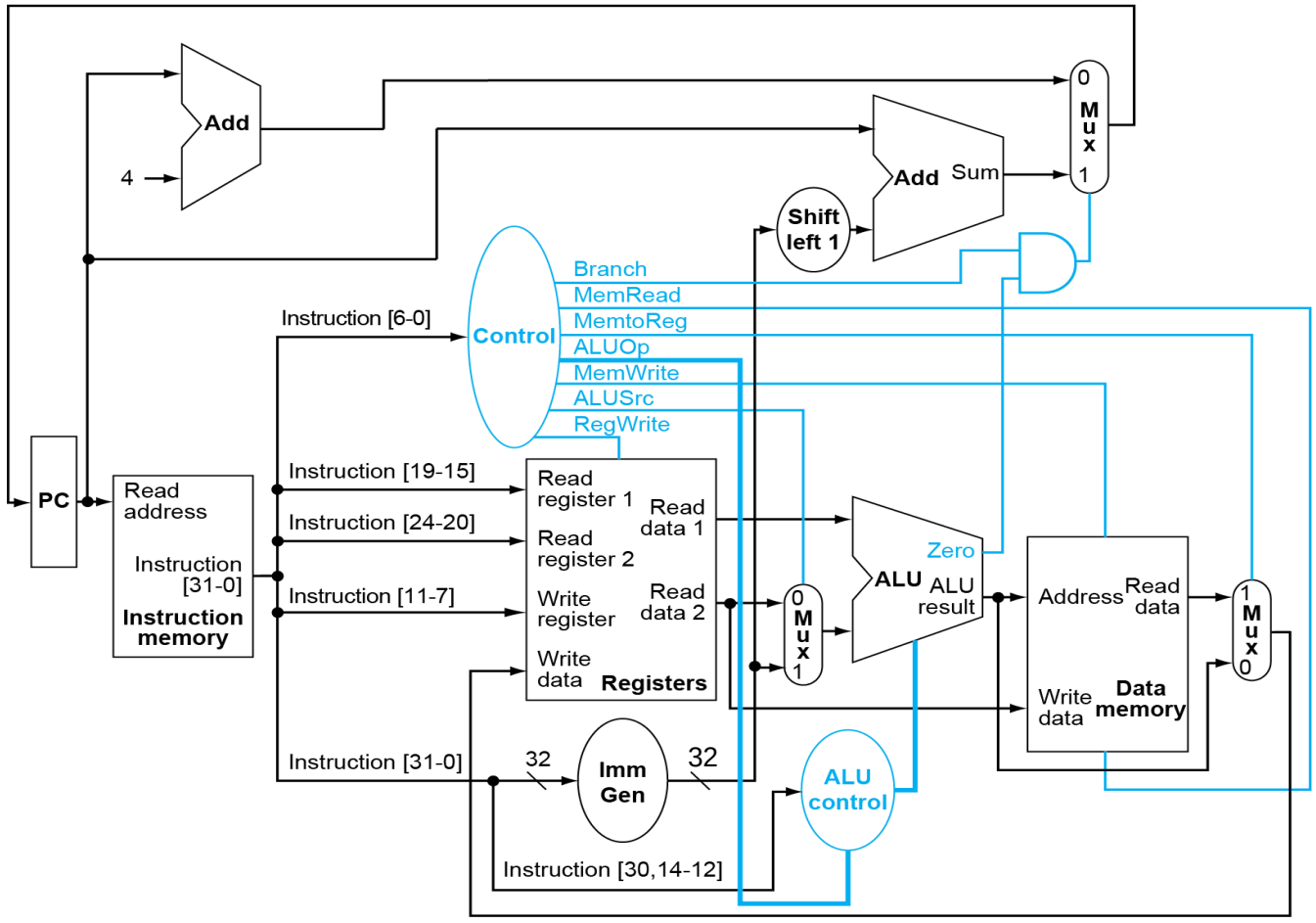
- A unidade de controle deve, a partir do código da instrução, fornecer os sinais que realizam as instruções na unidade operativa



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 Instr[6:0]
  - Os 2 registradores a serem lidos (rs1 e rs2) sempre estão nas posições Instr[19:15] e Instr[24:20] (tipo-R, beq e sw)
  - O registrador destino está sempre na posição Instr[11:7]
  - A formação do Imediato depende da instrução
- **A unidade de controle deve prover**
  - Controle dos multiplexadores
  - Controle de leitura e escrita da Memória de Dados
  - Controle de escrita no Banco de Registradores
  - Sinal de desvio condicional para seleção do próximo endereço
  - Controle ALUOp

Sinais de Controle



Instrução	ALUSrc	Mem2Reg	RegWrite	MemRead	MemWrite	Branch	ALUOp
Tipo-R	0	0	1	0	0	0	10
lw	1	1	1	1	0	0	00
sw	1	X	0	0	1	0	00
beq	0	X	0	0	0	1	01

Tabela Verdade do Controle

Instrução	ALUSrc	Mem2Reg	RegWrite	MemRead	MemWrite	Branch	ALUOp
Tipo-R 0110011	0	0	1	0	0	0	10
lw 0000011	1	1	1	1	0	0	00
sw 0100011	1	X	0	0	1	0	00
beq 1100011	0	X	0	0	0	1	01

- Lógica Combinacional Clássica – Tabela verdade, qual tamanho?
- ROM (*Read Only Memory*) – Qual o tamanho?
- PLA (*Programable Logic Array)* – Ferramentas automáticas

Exercícios

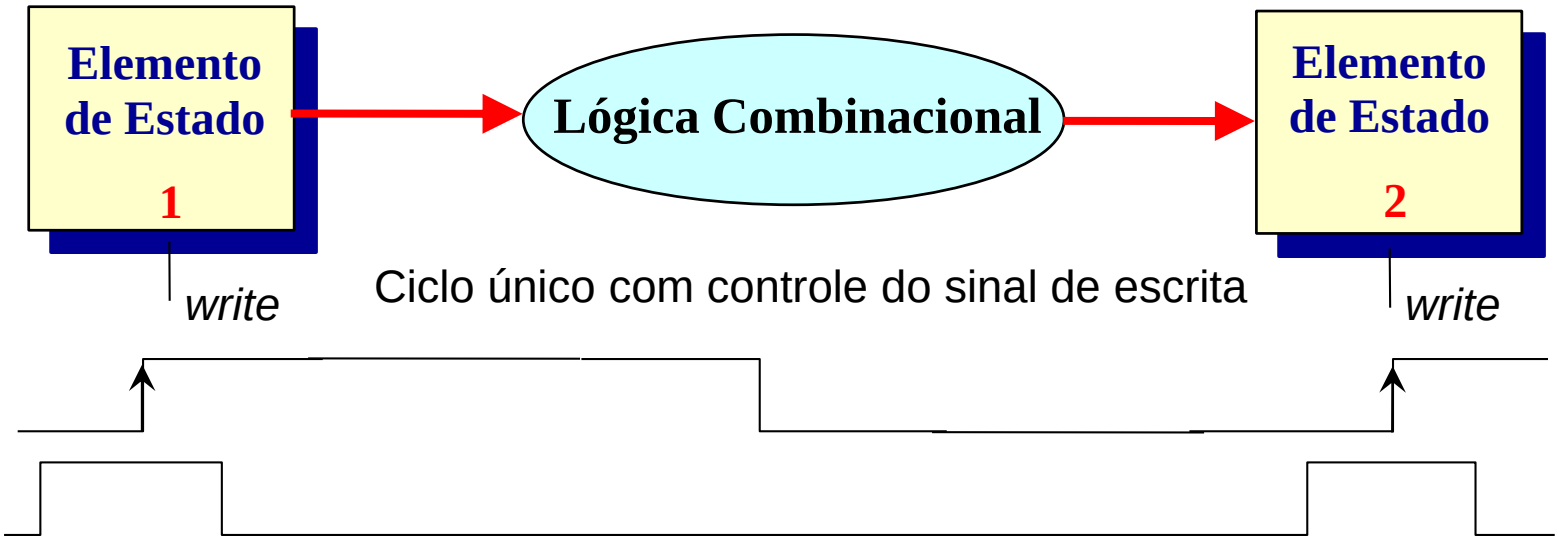
1) Analisar a instrução: add t1,t2,t3
2) Analisar a instrução: lw t1,offset(t2)
3) Analisar a instrução: beq t1,t2,desloc

4) Estender a organização do RISC-V para dar suporte à execução da instrução JAL, desvio incondicional e link

O endereço de desvio é obtido por: PC + (Imediato<<1)

Temporização

- Todas as tarefas executadas em 1 período do clock
- Elementos de estado alteram seu valor apenas na borda de subida do clock



Quais são os elementos de estado? Quando são escritos?

- PC não necessita controle de escrita (pq?)

Problemas da Implementação UNICICLO

- Período de *clock* determinado pelo caminho mais longo
  - **instrução lw: (imagina se tivesse operações em Ponto Flutuante!)**
    - leitura da instrução
    - leitura do registrador de base e geração do imediato
    - cálculo do endereço
    - leitura do dado da memória
    - escrita em registrador
- TODAS as instruções levam o mesmo tempo para executar: CPI=1

E ainda!

- Viola o princípio de tornar o caso comum rápido!
- Unidades funcionais duplicadas

Análise de Desempenho do Uniciclo

- Supondo os seguintes tempos de execução das unidades operativas:
  - Acesso a memória: 200ps
  - ULA e somadores: 100ps
  - Acesso ao banco de registradores (leitura ou escrita): 50ps
  - Multiplexadores, Unidades de Controle, acesso ao PC, Geração de Imediato e fios considerados sem atraso (!!!!)
- Qual das seguintes implementações seria mais rápida e quanto?

1) Implementação onde toda instrução é feita em 1 ciclo de *clock* de duração fixa.
2) Implementação onde toda instrução é feita em 1 ciclo de *clock* de duração somente o necessário (!!**exemplo ideal - didático**!!)

Para comparação consideremos um workload composto de:

25% lw, 10% sw, 45% Tipo-R, 15% beq, 5% jal

- Lembrando que:

Tempo Execução = Contagem de Instruções x CPI x Período de clock

Classe de instrução	Unidades funcionais usadas pela classe de instrução				
tipo R	Busca de instrução	Acesso a registrador	ALU	Acesso a registrador	
Load word	Busca de instrução	Acesso a registrador	ALU	Acesso à memória	Acesso a registrador
Store word	Busca de instrução	Acesso a registrador	ALU	Acesso à memória	
Branch	Busca de instrução	Acesso a registrador	ALU		
Jump and link	Busca de instrução		Somador		Acesso a registrador

Usando esses caminhos críticos, podemos calcular o tamanho exigido para cada classe de instrução:

Classe de instrução	Memória de instrução	Leitura de registrador	Operação ALU	Memória de dados	Escrita de registrador	Total
Tipo R	200	50	100		50	400ps
Load word	200	50	100	200	50	600ps
Store word	200	50	100	200		550ps
Branch	200	50	100			350ps
Jump and link	200		100		50	350ps

Tempo médio de execução de uma instrução:
Uniciclo fixo: 600ps
Uniciclo variável: 600x0.25+550x0.1+400x0.45+350x0.15+350x0.05=455ps

Fator de desempenho: n=600/455 = 1.32