

# O Processador (CPU)

## Implementação *Multi-Cycle*

António de Brito Ferrari

ferrari@ua.pt

### Ciclo de Instrução

1. Instruction Fetch – PC endereça a memória;  
leitura do código de instrução;  
Atualização do PC  
 $IR \leftarrow \text{MEM}[PC]$  ;  
 $PC \leftarrow PC + 4$  (instruções de Branch e Jump – outro valor para o PC)
2. Instruction Execution – a operação especificada no código de instrução é executada

## 2. Instruction Execution

2.1 – Descodificação da instrução e Leitura dos Registos –***Decode & Read Registers***

2.2 – Execução da operação (R-type ou I-type immed.)

OU cálculo do endereço de memória (Load, Store)

OU cálculo da condição de branch (BEQ)

***Execute***

2.3 – Acesso à memória (Load, Store)

OU Escrita do resultado (R-type ou I-type immed.)

***Memory access***

2.4 – Escrita do conteúdo da posição de memória (Load)

***Write Back***

***Multi-Cycle:***

**Cada passo executado num ciclo de relógio**

## Quantos ciclos de relógio?

- Maximizar desempenho = minimizar tempo de ciclo
- Minimizar tempo de ciclo => tempos de execução das operações em cada um dos ciclos serem iguais (ideal)
  - Cada uma dos passos de execução deve envolver componentes com tempos de execução das operações o mais próximos possível

## Quais as unidades funcionais envolvidas em cada ciclo?

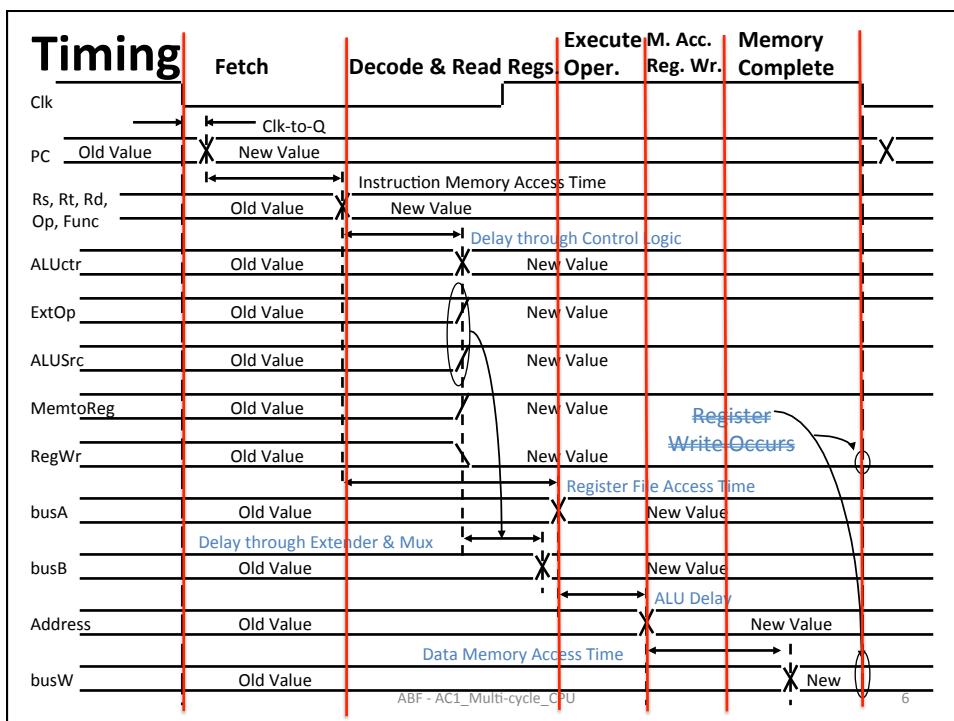
1. Fetch:
  - Memória (Read); ALU (Add)
2. Decode:
  - Unidade de Control: geração dos sinais de control
  - Datapath: Register File (Read)
3. Execute:
  - ALU (Op)
4. Memory access:
  - Memória (Read)
5. Write Back: Register File (Write)

➤ Tempo de ciclo mínimo (máxima frequência de relógio):

$$t_{\text{MemAccess}} = t_{\text{RegAccess}} = t_{\text{ALU}}$$

ABF - AC1\_Multi-cycle\_CPU

5



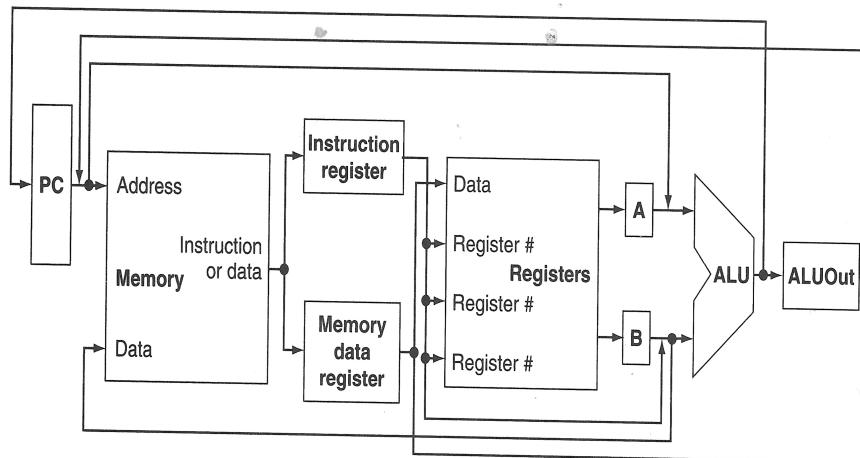
## Vantagens Multicycle

- Permite que uma unidade funcional seja usada mais do que uma vez, desde que em ciclos de relógio diferentes, durante a execução da instrução
- Permite que diferentes instruções sejam executadas num número de ciclos de relógio diferente, não ficando assim o tempo de execução de todas as instruções dependente do tempo de execução da instrução mais lenta (*lw*)

ABF - AC1\_Multi-cycle\_CPU

7

## Datapath Multicycle: visão geral



ABF - AC1\_Multi-cycle\_CPU

8

## Datapath: Multicycle vs. Single Cycle

- Uma memória única para instruções e dados
- Uma única ALU em vez de uma ALU e 2 somadores (na *Fetch Unit single cycle*)
- Um ou mais registos colocados à saída de cada uma das principais unidades funcionais para que os valores possam ser usados no ciclo de relógio seguinte:
  - **Instruction Register** e **Data Register** à saída da memória
  - Registos **A** e **B** à saída do banco de registos
  - Registo **ALUout** à saída da ALU
- Mais / maiores Multiplexers

## 1. Multicycle: Datapath

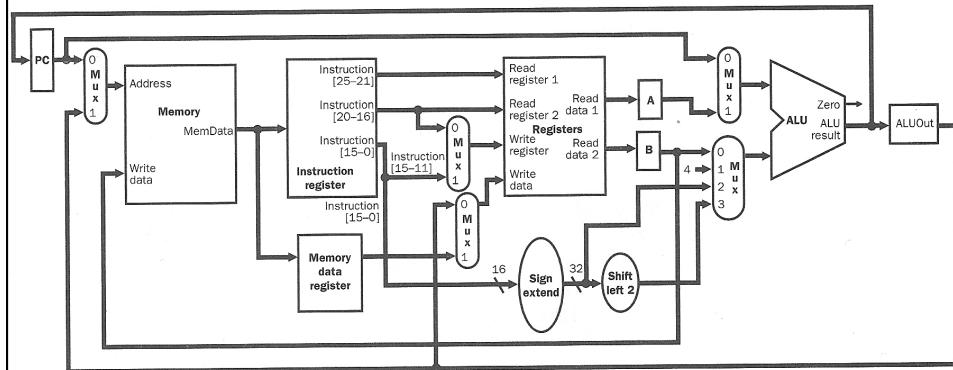
## As Fases de Execução

1. Instruction Fetch – leitura do código de instrução; Atualização do PC  
 $IR \leftarrow MEM[PC]$ ;  $PC \leftarrow PC + 4$
2. Descodificação da instrução e Leitura dos Registros  
 $A \leftarrow Reg[IR[25:21]]$ ;  $B \leftarrow Reg[IR[20:16]]$ ;  $ALUOut \leftarrow PC + (\text{sign-extend } (IR[15:0] \ll 2))$
3. R-type ou I-type immed.: Execução da operação  
 $ALUOut \leftarrow A \text{ op } B$   
**Load, Store:** cálculo do endereço de memória  
 $ALUOut \leftarrow A + (\text{sign-extend } (IR[15:0]))$   
**BEQ:** cálculo da condição de branch – execução terminada  
 $\text{if } (A == B) PC \leftarrow ALUOut$
4. R-type ou I-type immed.: Escrita do resultado – execução terminada  
 $Reg[IR[15:11]] \leftarrow ALUOut$   
**Load, Store:** Acesso à memória; **Store:** execução terminada  
Load:  $MDR \leftarrow Mem[ALUOut]$       Store:  $Mem[ALUOut] \leftarrow B$
5. Load: Escrita no registo do conteúdo da posição de memória  
 $Reg[IR[20:16]] \leftarrow MDR$

ABF - AC1\_Multi-cycle\_CPU

11

## Multi-Cycle Datapath



**Novo:** Memória única (instruções + dados)

Instruction Register (IR) – contém o código de Instrução durante os 5 ciclos

Memory Data Register (MDR) – contém o dado lido/a ser escrito da memória na execução

ALU para além de executar a operação especificada pela instrução calcula novo valor do PC

ALUOut – armazena o resultado da operação executada pela ALU no 3º ciclo (2.2)

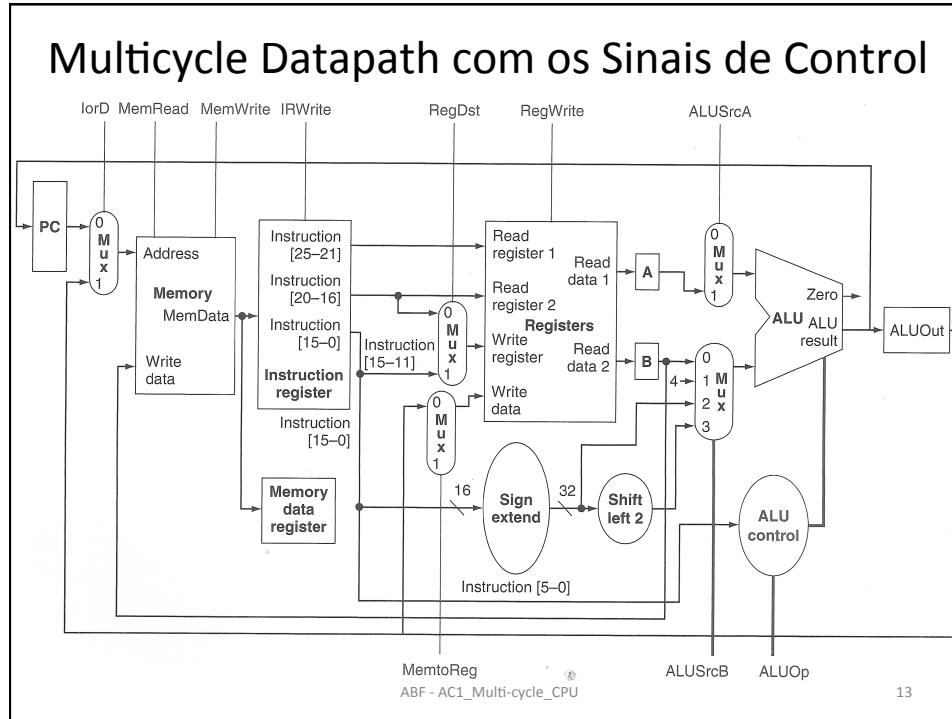
MUX 2:1 (PC ou Rs) na entrada A da ALU; MUX 4:1 na entrada B da ALU

MUX 2:1 (Address Bus) seleciona endereço de memória:

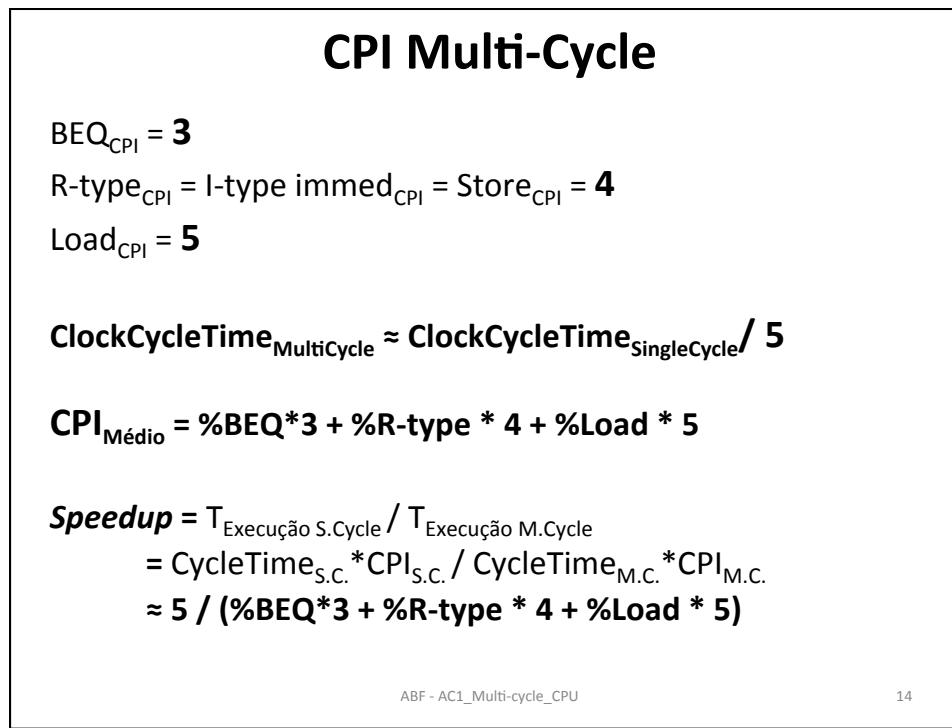
1º ciclo (Fetch da Instrução): Addr. <= PC;      4º ciclo (Ld, St): Addr. <= ALUOut

ABF - AC1\_Multi-cycle\_CPU

12



13



ABF - AC1\_Multi-cycle\_CPU

14

## 2. Multi Cycle: Control

ABF - AC1\_Multi-cycle\_CPU

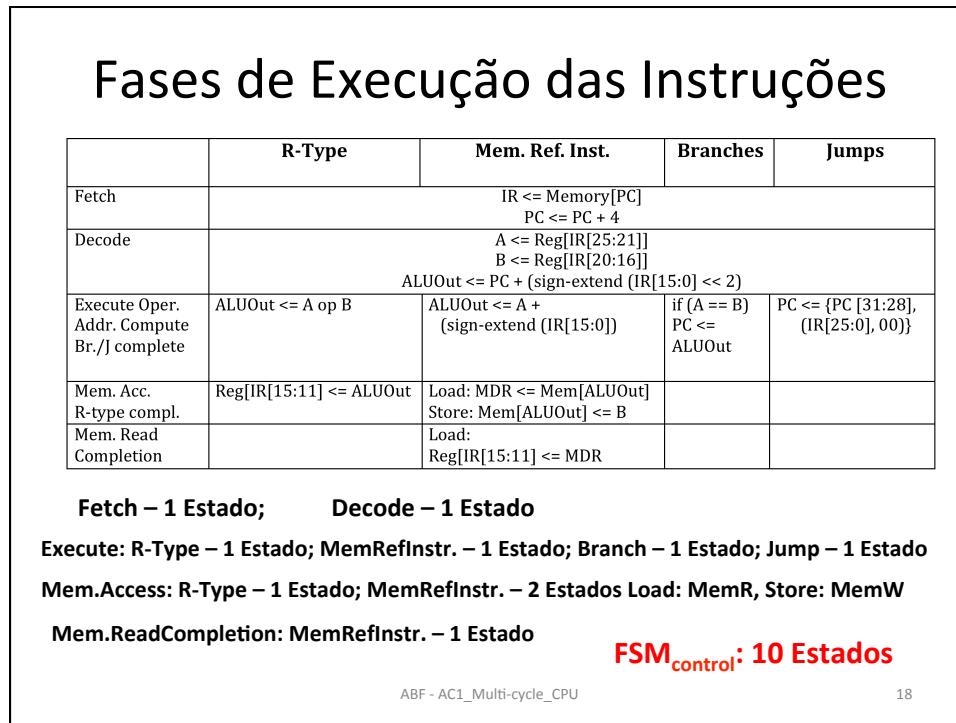
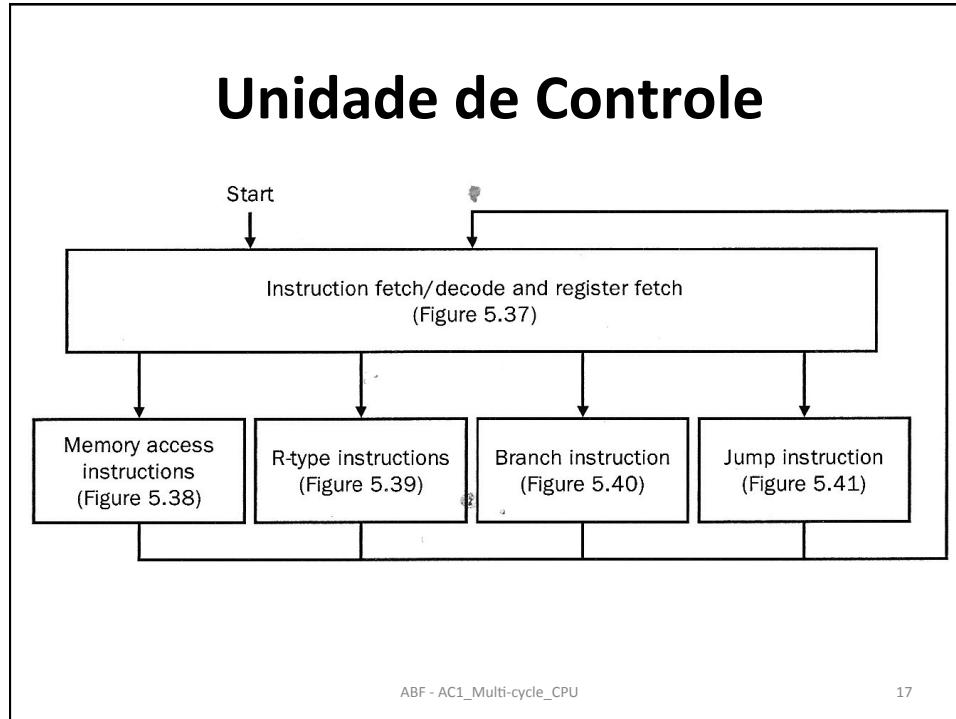
15

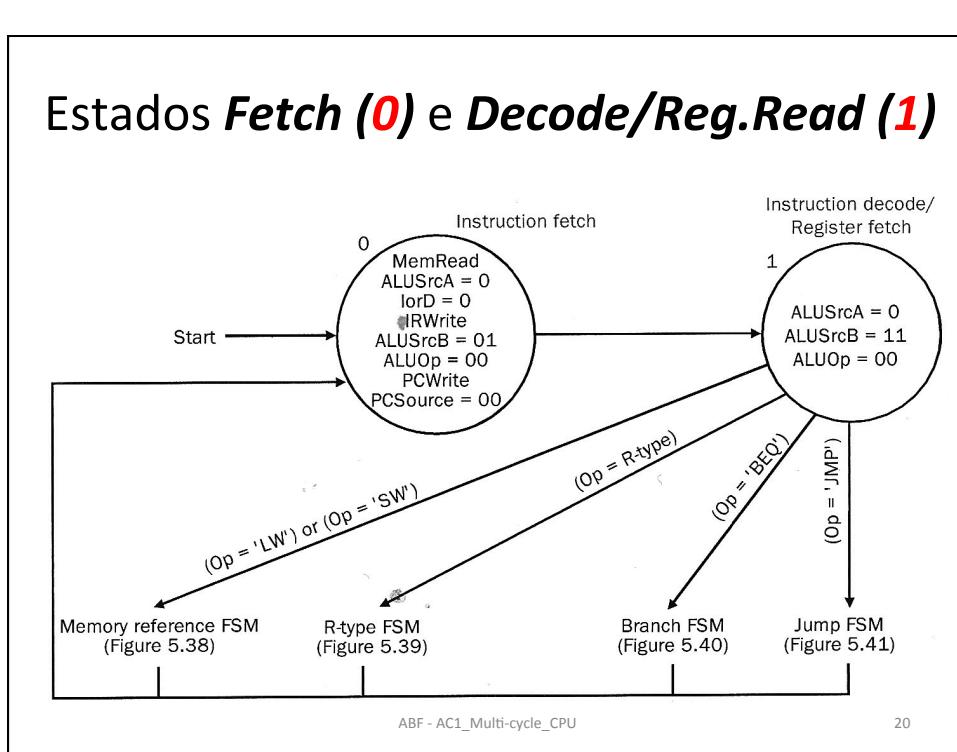
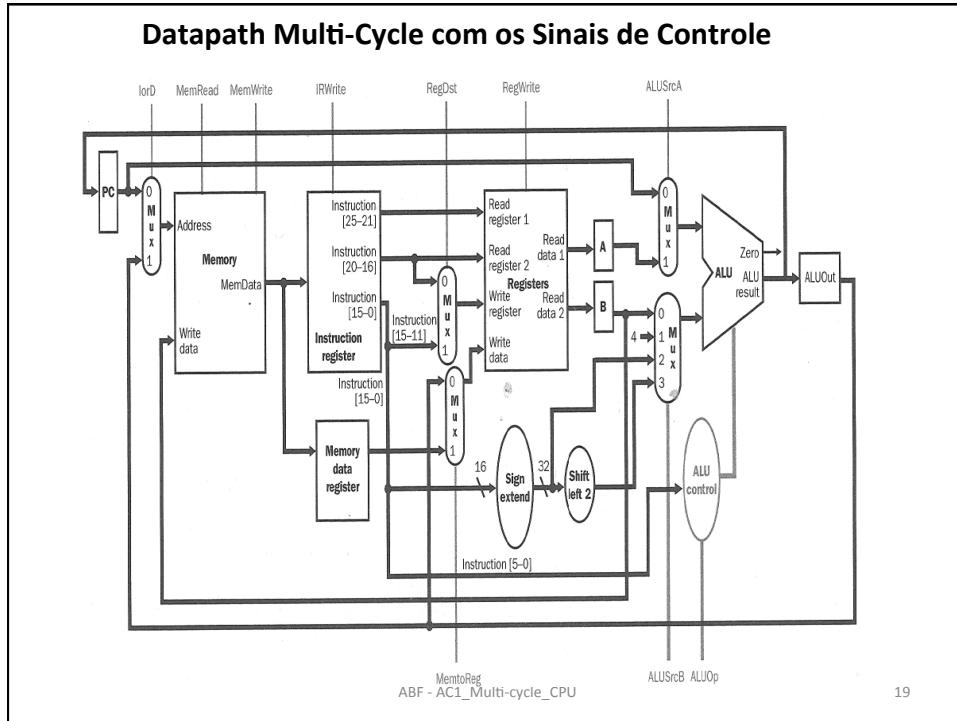
### Unidade de Controle

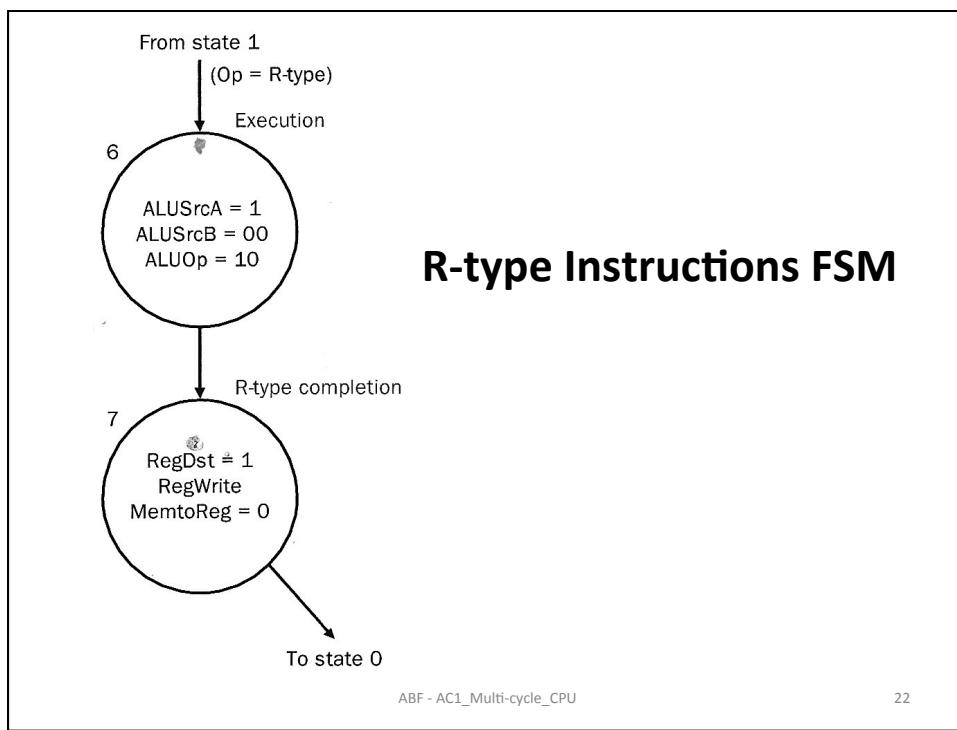
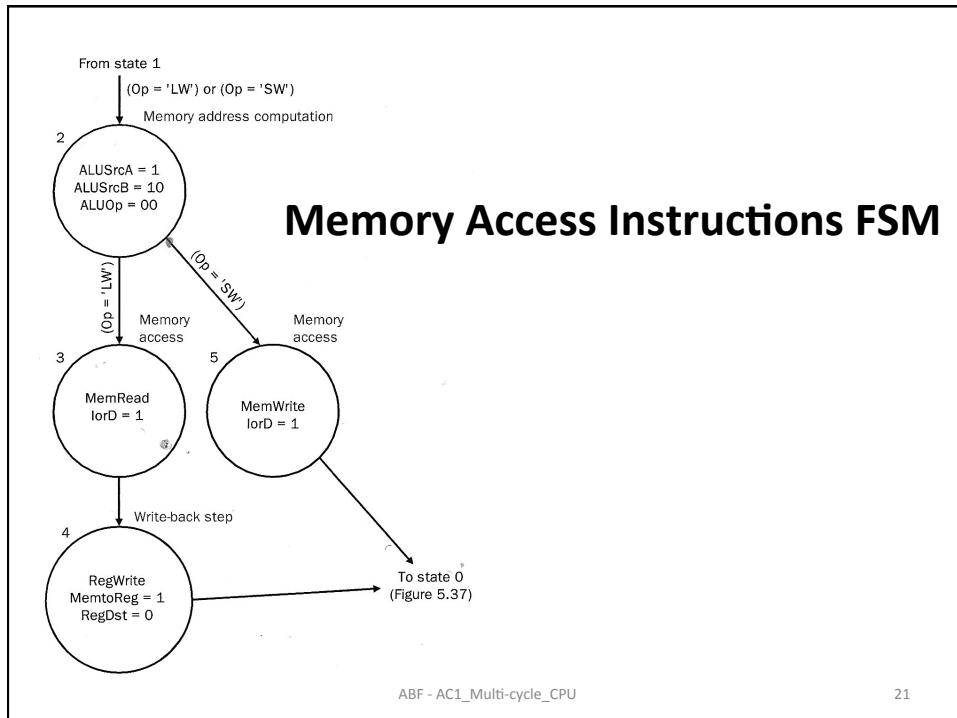
- Single-Cycle: circuito combinatório
- Multi-Cycle: Máquina de Estados (FSM)
  - **Ciclo corresponde a tantos Estados quantas as diferenças das operações a executar para cada categoria de instruções**
  - Em cada estado a unidade de controle gera os sinais que controlam as operações do datapath a executar nesse estado

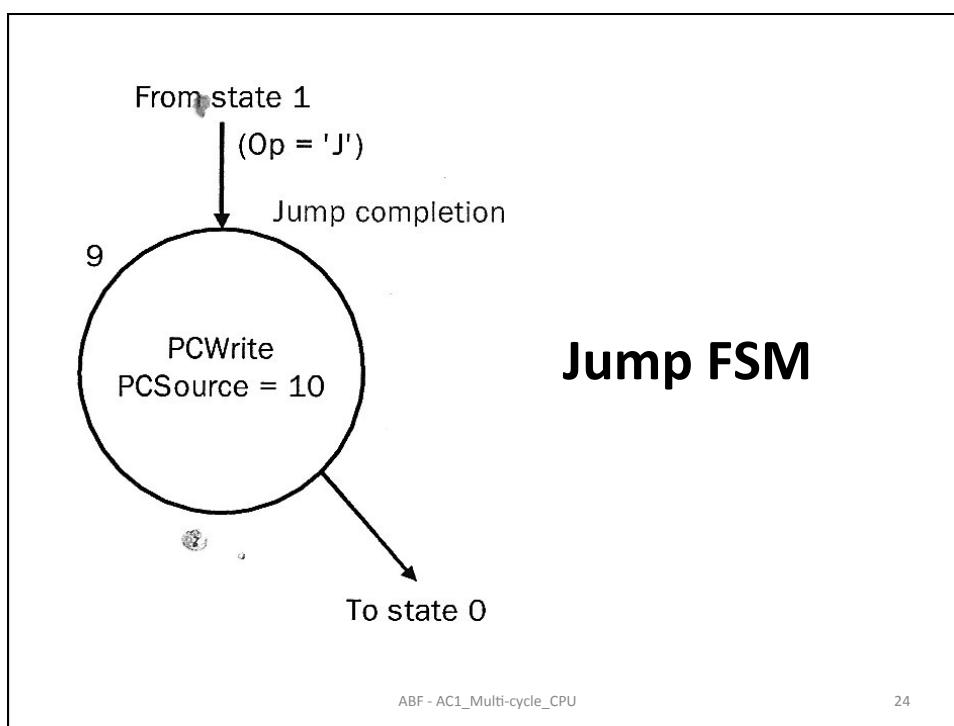
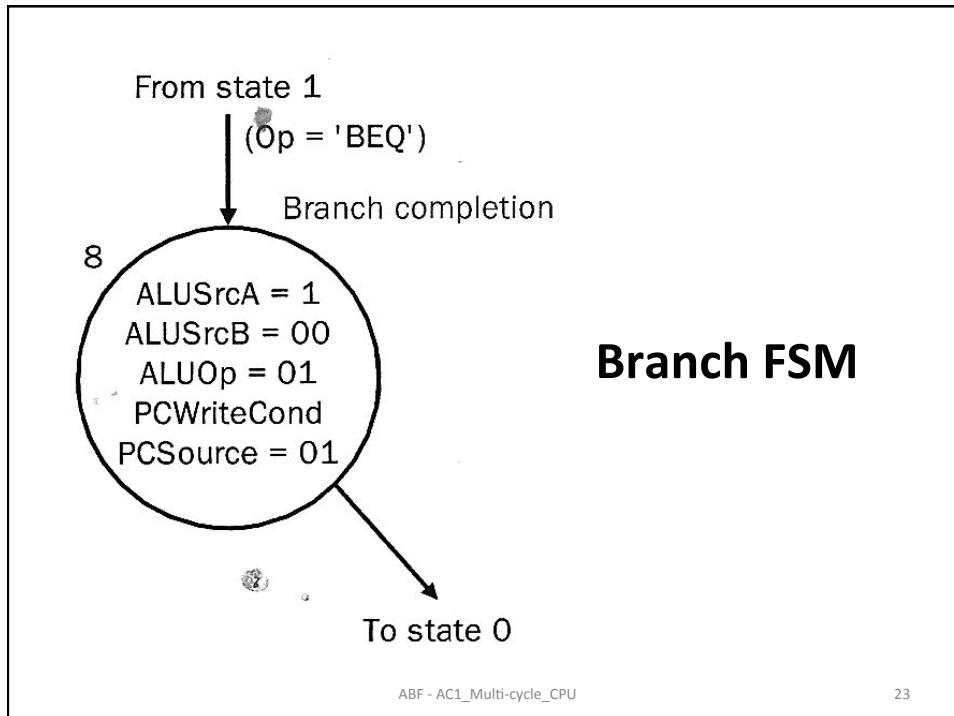
ABF - AC1\_Multi-cycle\_CPU

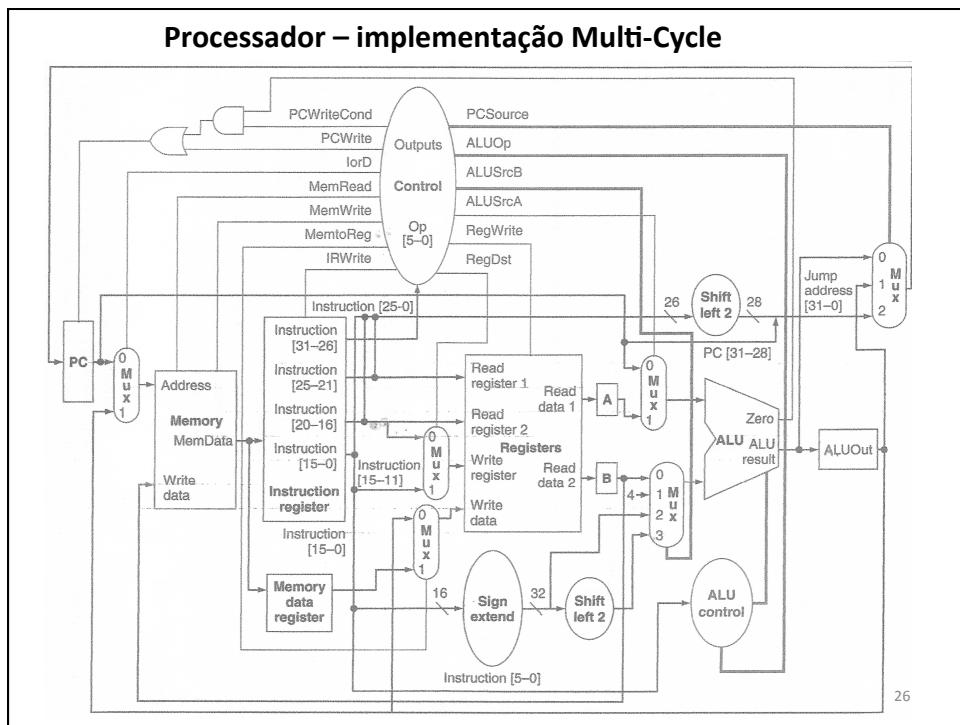
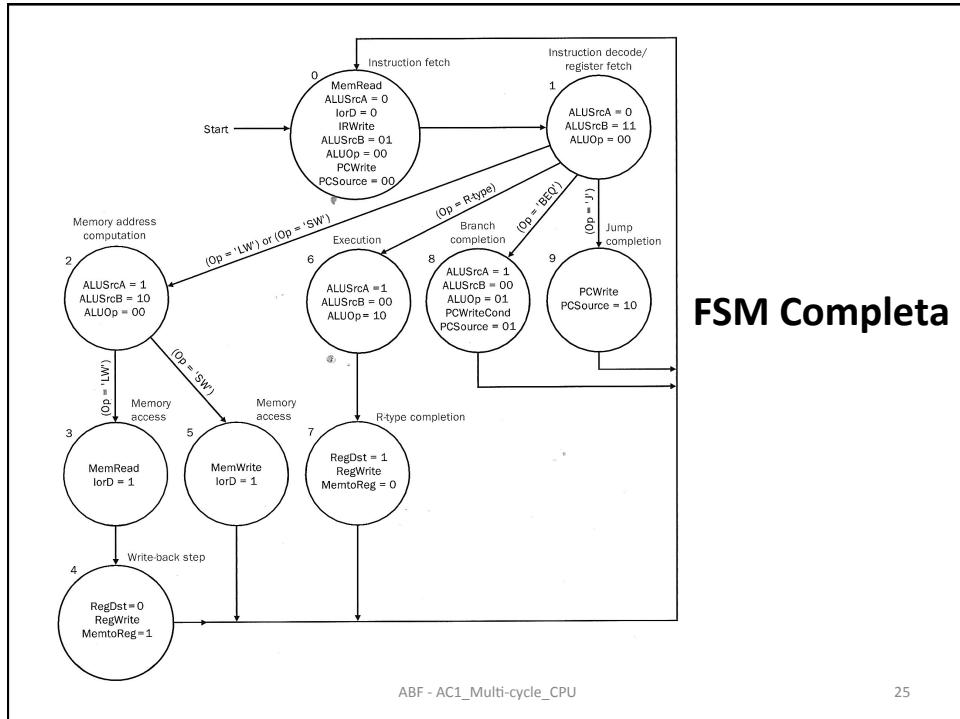
16









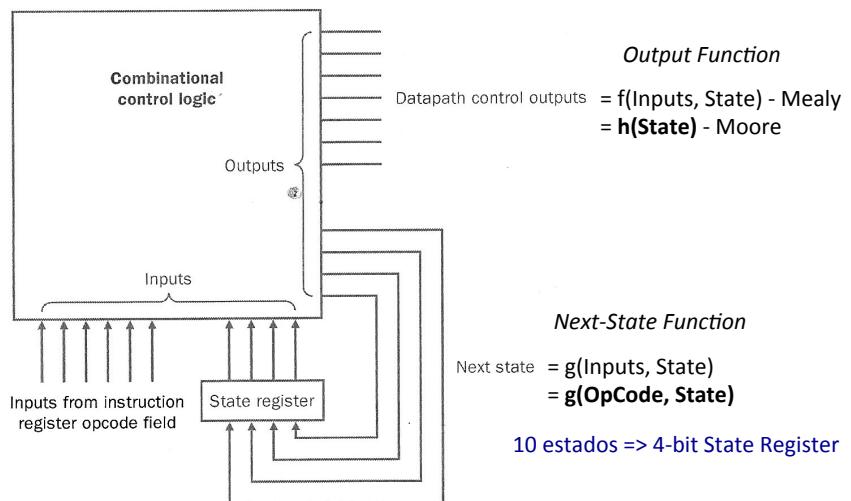


### 3. Síntese da Unidade de Controle (FSM)

ABF - AC1\_Multi-cycle\_CPU

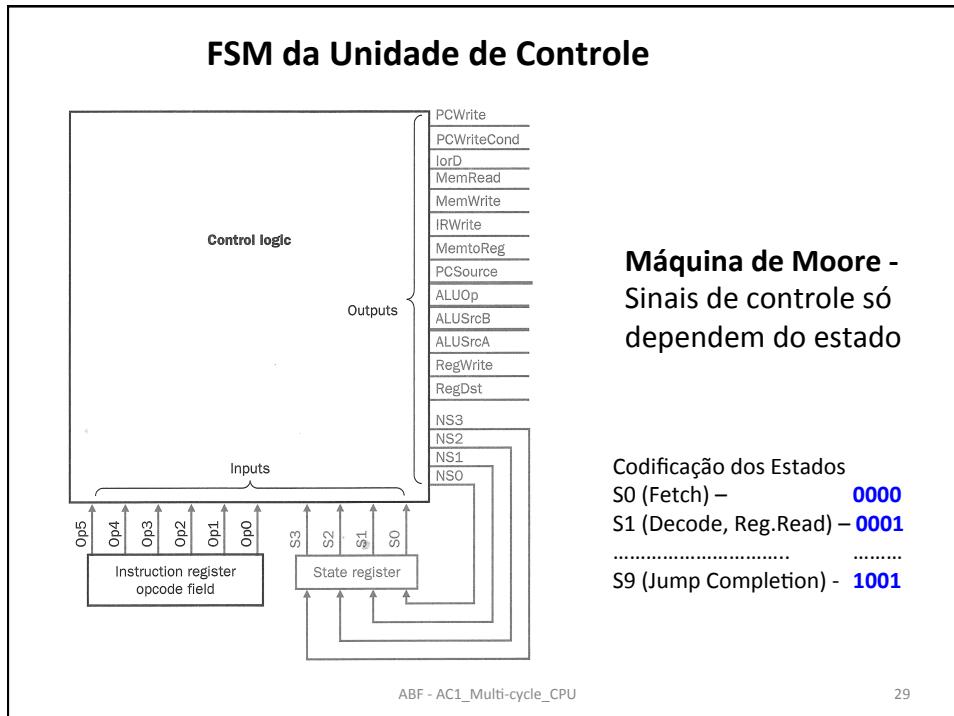
27

### FSM –esquema geral



ABF - AC1\_Multi-cycle\_CPU

28



Output	Current states	Op
PCWrite	state0 + state9	
PCWriteCond	state8	
IorD	state3 + state5	
MemRead	state0 + state3	
MemWrite	state5	
IRWrite	state0	
MemtoReg	state4	
PCSrc1	state9	
PCSrc0	state8	
ALUOp1	state6	
ALUOp0	state8	
ALUSrcB1	state1 + state2	
ALUSrcB0	state0 + state1	
ALUSrcA	state2 + state6 + state8	
RegWrite	state4 + state7	
RegDst	state7	
NextState0	state4 + state5 + state7 + state8 + state9	
NextState1	state0	
NextState2	state1	(Op = 'lw') + (Op = 'sw')
NextState3	state2	(Op = 'lw')
NextState4	state3	
NextState5	state2	(Op = 'sw')
NextState6	state1	(Op = 'R-type')
NextState7	state6	
NextState8	state1	(Op = 'beq')
NextState9	state1	(Op = 'jmp')

30

## Unidade de Control – Output Function

Outputs	Input values ( <b>S[3-0]</b> )									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
PCWrite	1	0	0	0	0	0	0	0	0	1
PCWriteCond	0	0	0	0	0	0	0	0	1	0
IorD	0	0	0	1	0	1	0	0	0	0
MemRead	1	0	0	1	0	0	0	0	0	0
MemWrite	0	0	0	0	0	1	0	0	0	0
IRWrite	1	0	0	0	0	0	0	0	0	0
MemtoReg	0	0	0	0	1	0	0	0	0	0
PCSource1	0	0	0	0	0	0	0	0	0	1
PCSource0	0	0	0	0	0	0	0	0	1	0
ALUOp1	0	0	0	0	0	0	1	0	0	0
ALUOp0	0	0	0	0	0	0	0	0	1	0
ALUSrcB1	0	1	1	0	0	0	0	0	0	0
ALUSrcB0	1	1	0	0	0	0	0	0	0	0
ALUSrcA	0	0	1	0	0	0	1	0	1	0
RegWrite	0	0	0	0	1	0	0	1	0	0
RegDst	0	0	0	0	0	0	0	1	0	0

ABF - AC1\_Multi-cycle\_CPU

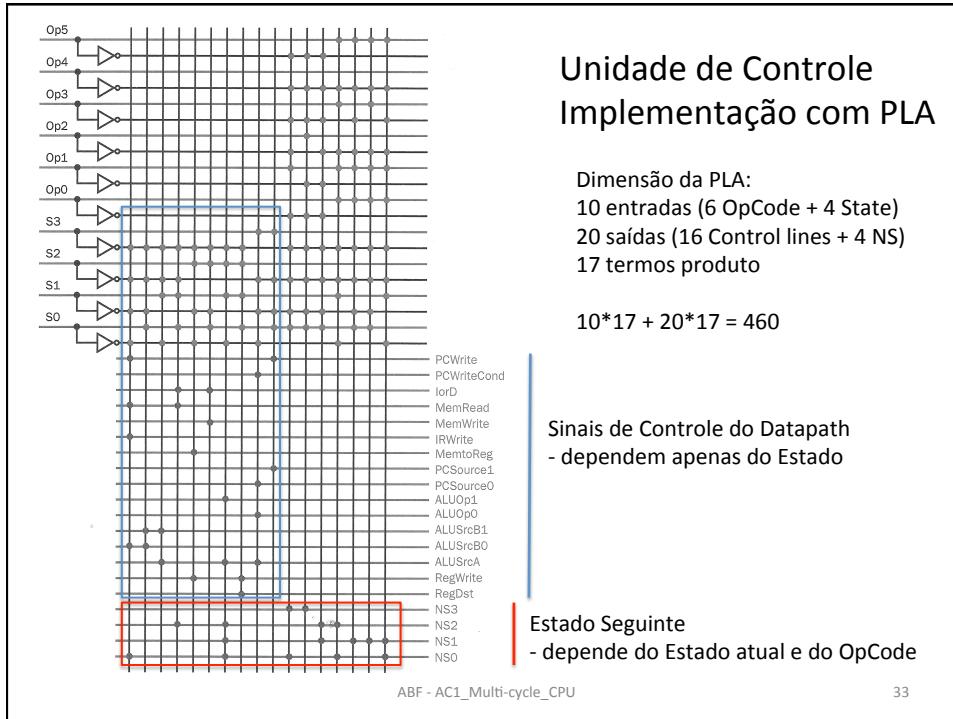
31

## Unidade de Control – Next State Function

Current state <b>S[3-0]</b>	Op [5-0]					
	<b>000000</b> (R-format)	<b>000010</b> (jmp)	<b>000100</b> (beq)	<b>100011</b> (lw)	<b>101011</b> (sw)	<b>Any other value</b>
0000	0001	0001	0001	0001	0001	0001
0001	0110	1001	1000	0010	0010	illegal
0010	XXXX	XXXX	XXXX	0011	0101	illegal
0011	0100	0100	0100	0100	0100	illegal
0100	0000	0000	0000	0000	0000	illegal
0101	0000	0000	0000	0000	0000	illegal
0110	0111	0111	0111	0111	0111	illegal
0111	0000	0000	0000	0000	0000	illegal
1000	0000	0000	0000	0000	0000	illegal
1001	0000	0000	0000	0000	0000	illegal

ABF - AC1\_Multi-cycle\_CPU

32



## Microprogramação

- Implementação alternativa: **unidade de controle microprogramada**
- Ideia-base: numa implementação multi-ciclo em cada ciclo de execução de uma instrução o datapath executa um conjunto de operações determinado pelos valores dos sinais de controle nesse ciclo – o datapath executa uma **(micro)instrução**
- A execução de uma instrução corresponde à execução de uma sequência de microinstruções, i.e. à execução de um **(micro)programa**

Aula seguinte:

## **MICROPROGRAMAÇÃO**