

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?

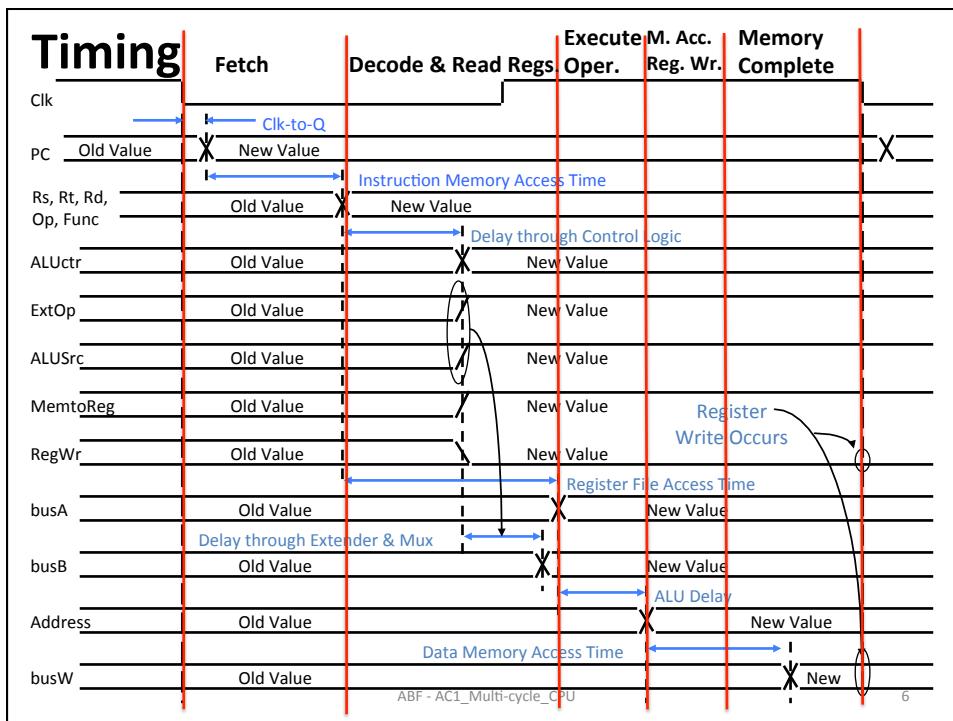
1. Fetch:
 - Memória (Read); ALU (Add)
2. Decode:
 - Unidade de Control: geração dos sinais de control
 - Datapath: Register File (Read)
3. Execute:
 - ALU (Op)
4. Memory access:
 - Memória (Read)
5. Write Back: Register File (Write)

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

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

ABF - AC1_Multi-cycle_CPU

5



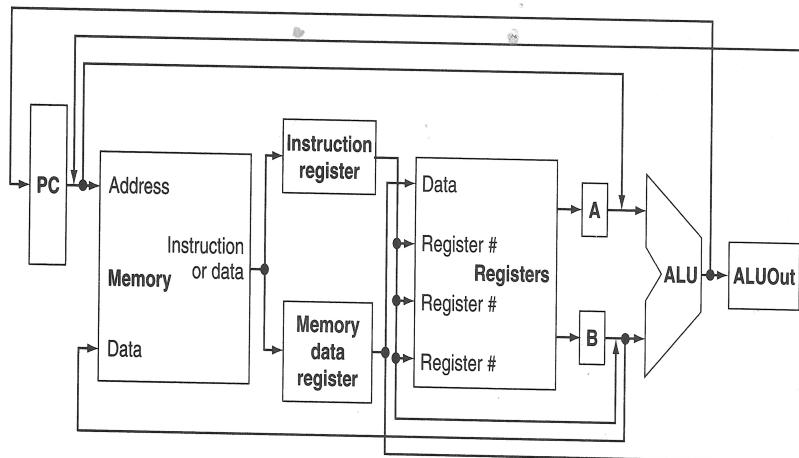
Vantagens Multicycle

- 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



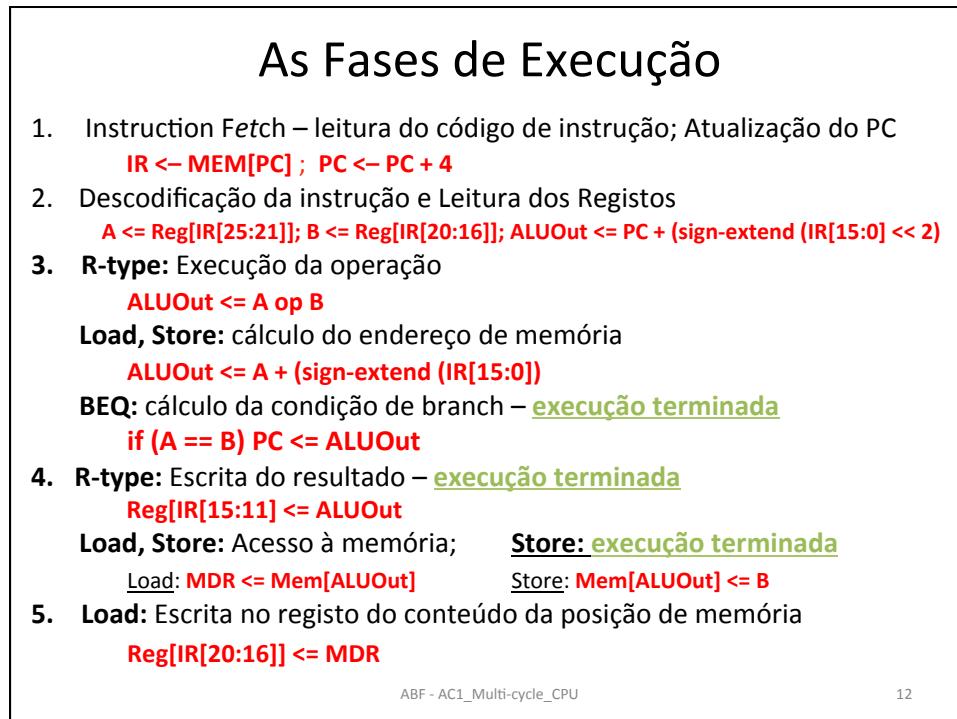
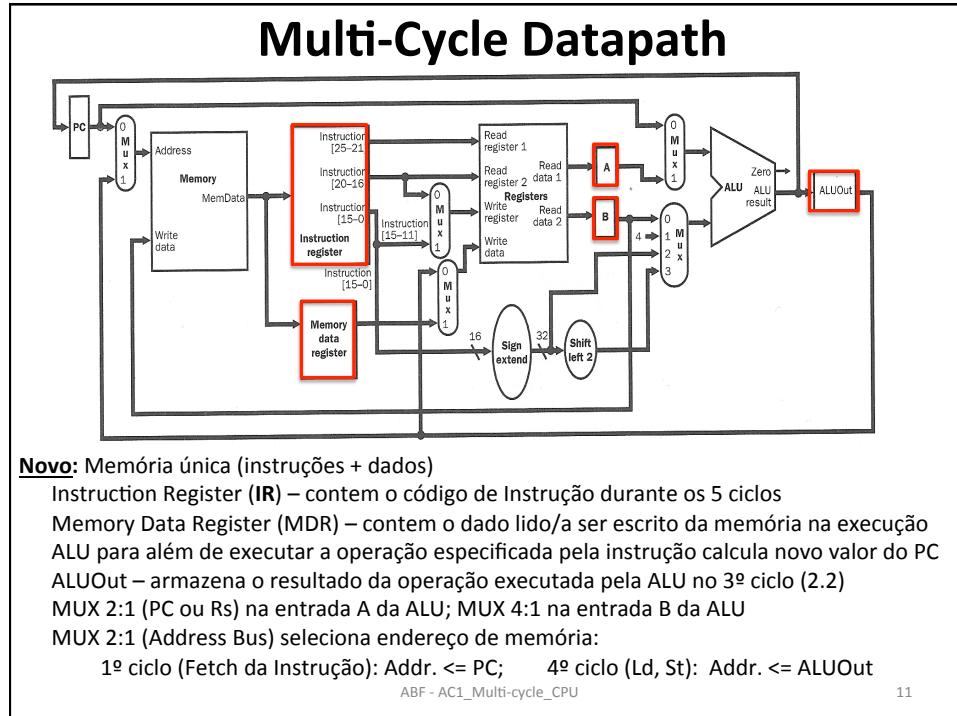
ABF - AC1_Multi-cycle_CPU

8

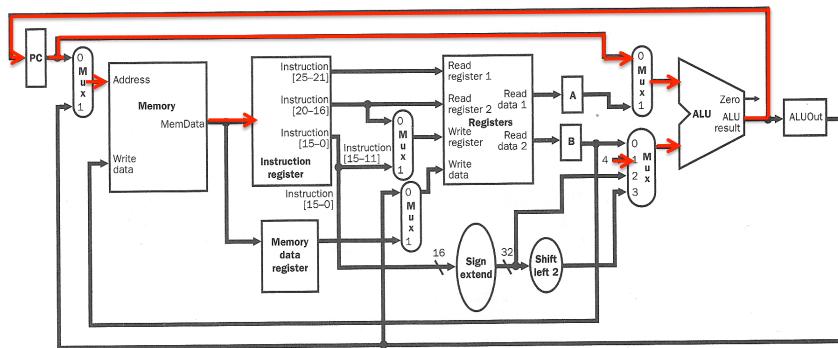
Datapath: Multicycle vs. Single Cycle

- 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



1º ciclo: Instruction Fetch



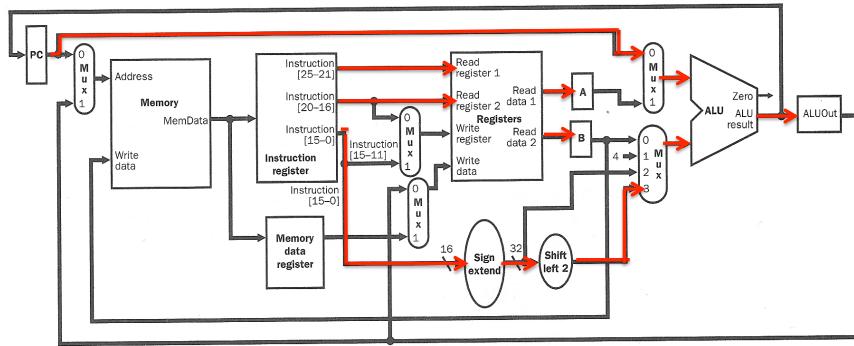
Program Counter endereça a memória
Entradas da ALU: PC e 4 Operação: soma
MemData = Código da instrução a executar

No fim do 1º ciclo:
IR = Código da instrução a executar
PC = PC + 4

ABF - AC1_Multi-cycle_CPU

13

2º ciclo: Descodificação, Leitura dos Registros, Cálculo do endereço-alvo



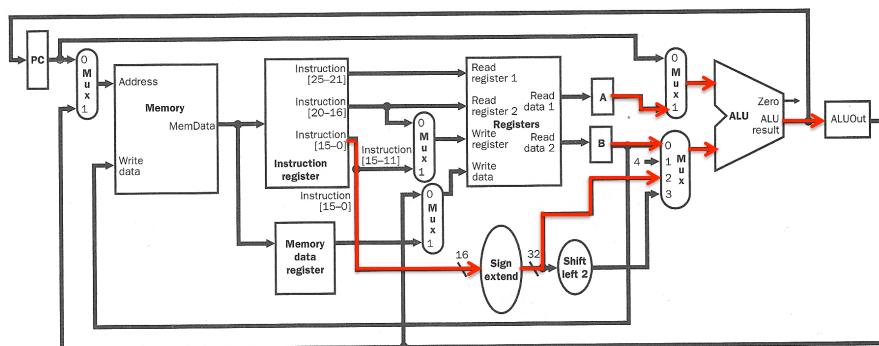
No fim do ciclo:

**A <= Reg[IR[25:21]] B <= Reg[IR[20:16]];
ALUOut <= PC + (sign-extend (IR[15:0] << 2))**

ABF - AC1_Multi-cycle_CPU

14

3º ciclo: R-type / Immediate Op.

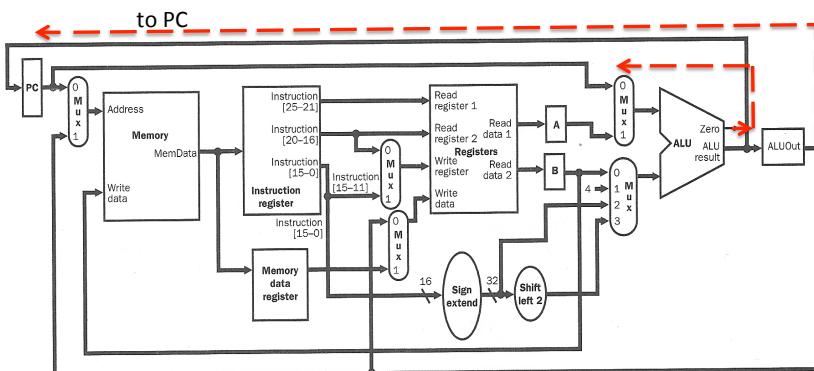


ALUOut <= A op B ou ALUOut <= A op (sign-extend (IR[15:0]))

ABF - AC1_Multi-cycle_CPU

15

3º Ciclo: Branch on Equal



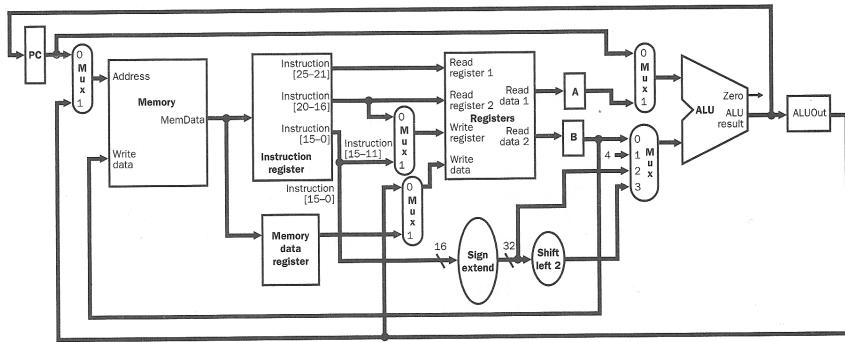
if (Zero) PC <= ALUOut

execução terminada

ABF - AC1_Multi-cycle_CPU

16

3º ciclo: Load, Store

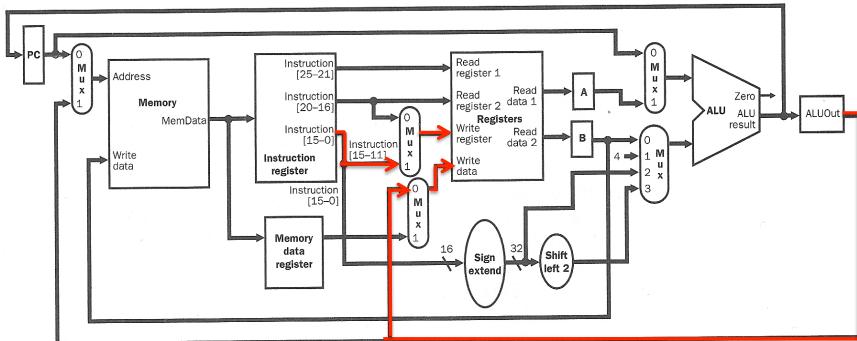


ALUOut <= A op (sign-extend (IR[15:0]))

ABF - AC1_Multi-cycle_CPU

17

4º ciclo: R-Type



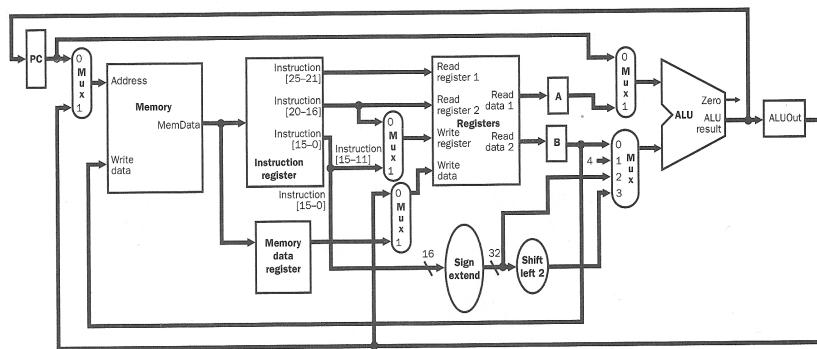
Reg[IR[15:11]] <= ALUOut

execução terminada

ABF - AC1_Multi-cycle_CPU

18

4º ciclo:Load

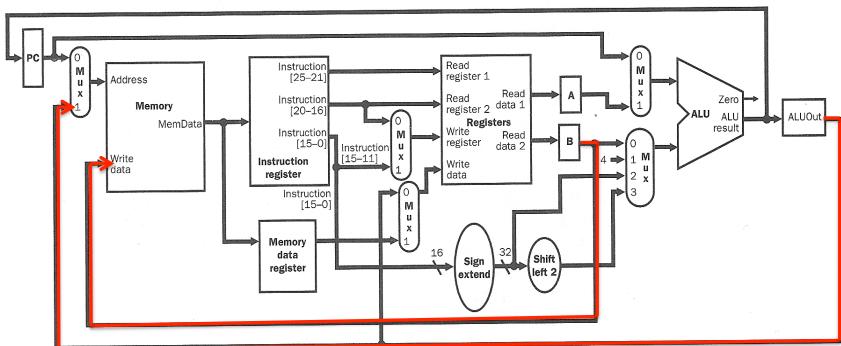


MDR <= Mem[ALUOut]

ABF - AC1_Multi-cycle_CPU

19

4º ciclo:Store



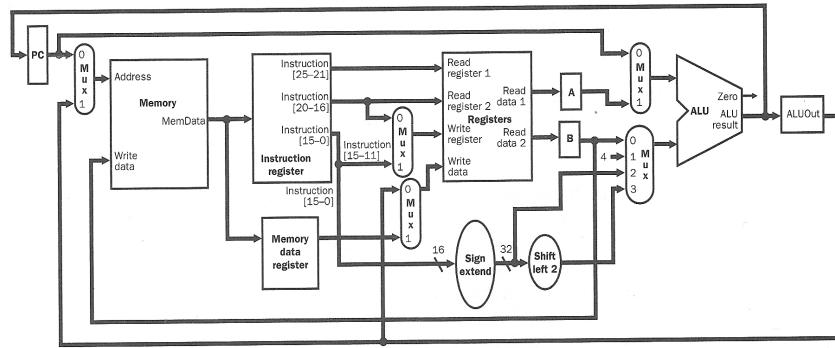
Mem[ALUOut] <= B

execução terminada

ABF - AC1_Multi-cycle_CPU

20

5º ciclo: Load



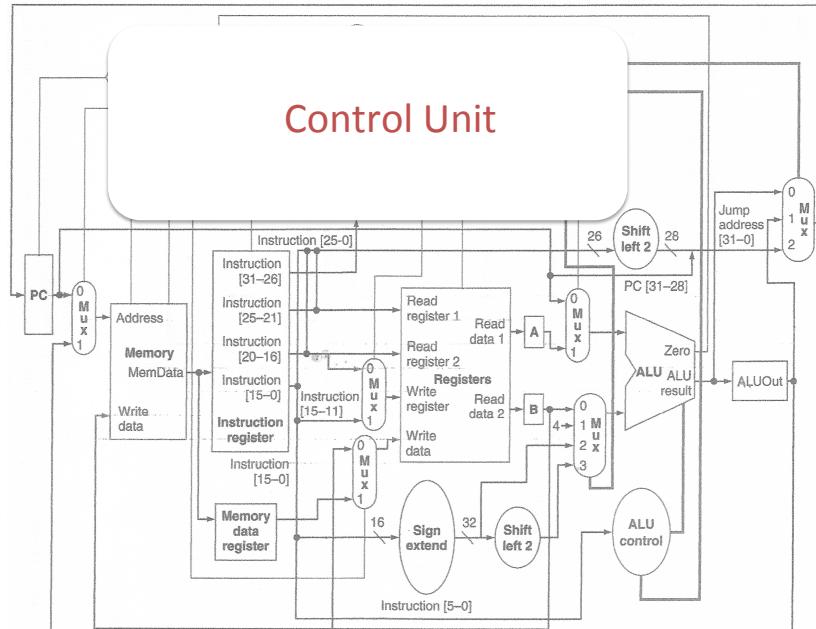
Reg[IR[20:16 <= MDR

execução terminada

ABF - AC1_Multi-cycle_CPU

21

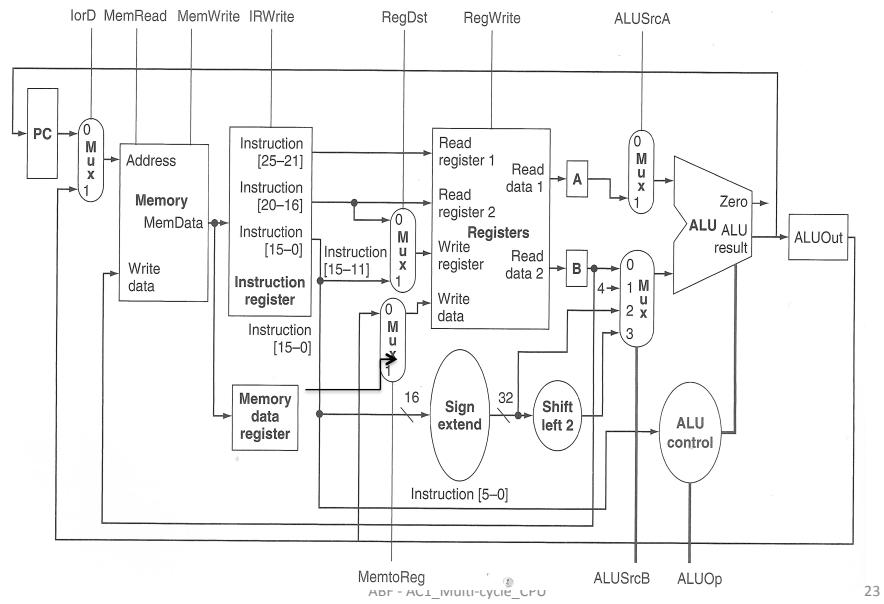
Control Unit



ABF - AC1_Multi-cycle_CPU

22

Multicycle Datapath com os Sinais de Control



23

CPI Multi-Cycle

$$\text{BEQ}_{\text{CPI}} = 3$$

$$\text{R-type}_{\text{CPI}} = \text{l-type immed}_{\text{CPI}} = \text{Store}_{\text{CPI}} = 4$$

$$\text{Load}_{\text{CPI}} = 5$$

$$\text{ClockCycleTime}_{\text{MultiCycle}} \approx \text{ClockCycleTime}_{\text{SingleCycle}} / 5$$

$$\text{CPI}_{\text{Médio}} = \% \text{BEQ} * 3 + \% \text{R-type} * 4 + \% \text{Load} * 5$$

$$\begin{aligned} \text{Speedup} &= T_{\text{Execução S.Cycle}} / T_{\text{Execução M.Cycle}} \\ &= \text{CycleTime}_{\text{S.C.}} * \text{CPI}_{\text{S.C.}} / \text{CycleTime}_{\text{M.C.}} * \text{CPI}_{\text{M.C.}} \\ &\approx 5 / (\% \text{BEQ} * 3 + \% \text{R-type} * 4 + \% \text{Load} * 5) \end{aligned}$$

ABF - AC1_Multi-cycle_CPU

24