

RESPOSTAS

unicycle.cpu

Primeiro Desafio:

Código a ser executado:

li \$s0, 10

li \$s1, 17

add \$s2, \$s1, \$s0

Esse código tem 3 linhas, portanto será utilizado 3 clocks para a sua execução completa porque essa é uma máquina monociclo. A primeira instrução começa no endereço **0**, a segunda no **4** e a terceira no **8**.

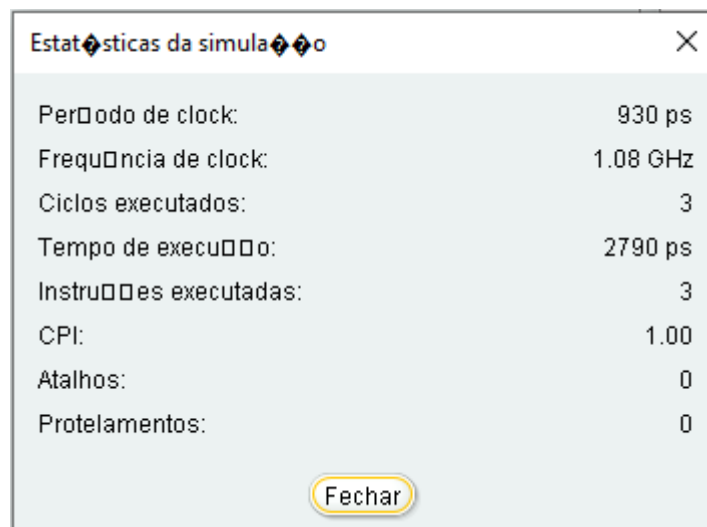
1. A primeira linha de comando registra na variável **\$s0** o valor **10**;
2. A segunda linha de comando registra na variável **\$s1** o valor **17**;
3. A terceira linha de comando registra na variável **\$s2** o valor de soma entre as variáveis **\$s0** e **\$s1**.

16: \$s0	10
17: \$s1	17
18: \$s2	27

Como essa é uma arquitetura **RISC**, tudo é gravado nos registradores da máquina. Nessa máquina monociclo, ela a cada clock de execução faz os 5 passos essenciais (Busca; Decodificação; Execução; Gravar dados na memória; Voltar ao estado inicial), ao mesmo tempo, ou seja, faz os 5 itens por clock.

Segundo Desafio:

Foram 3 ciclos de execução por justamente serem 3 linhas de comando de código.
O período do clock foi de **930 ps** e o tempo de execução **2790 ps**.



Período de clock:	930 ps
Frequência de clock:	1.08 GHz
Ciclos executados:	3
Tempo de execução:	2790 ps
Instruções executadas:	3
CPI:	1.00
Atalhos:	0
Protelamentos:	0

Fechar

Primeiro Desafio:

Código a ser executado é o mesmo:

```
li $s0, 10
```

```
li $s1, 17
```

```
add $s2, $s1, $s0
```

A primeira instrução começa no endereço **0**, a segunda no **4** e a terceira no **8**.
Como essa máquina é em pipeline, o código será executado de maneira paralela.

Nessa máquina, ela faz os 5 passos essenciais (*Busca; Decodificação; Execução; Gravar dados na memória; Voltar ao estado inicial*), de maneira paralelizada em sua execução, ou seja, ela faz paralelismo de código a nível de instrução.

1. O primeiro clock faz a **busca** da 1ª linha de código;
2. O segundo clock faz a **busca** da 2ª linha de código e **decodifica** 1ª linha de código;
3. O terceiro clock faz a **busca** da 3ª linha de código, **decodifica** 2ª linha de código e **executa** a 1ª linha de código registrando na variável **\$s0** o valor **10**;
4. O quarto clock **decodifica** 3ª linha de código, **executa** a 2ª linha de código registrando na variável **\$s1** o valor **17** e **grava** os dados da execução da 1ª linha de código;
5. O quinto clock **executa** a 3ª linha de código registrando na variável **\$s2** o valor de soma entre as variáveis **\$s0** e **\$s1**, **grava** os dados da execução da 2ª linha de código e **volta** ao estado inicial;
6. O sexto clock **grava** os dados da execução da 3ª linha de código e **volta** ao estado inicial;
7. O sétimo clock **volta** ao estado inicial, fechando assim o ciclo.

No final ficou gravado na variável **\$s0** o valor **10**, **\$s1** o valor **17** e na variável **\$s2** o valor resultante da soma **27**.

16: \$s0	10
17: \$s1	17
18: \$s2	27

Segundo Desafio:

Foram 7 ciclos de execução por justamente as linhas de comando de código serem executadas em paralelo.

O período do clock foi de **400 ps** que por sinal melhorou muito em relação a arquitetura monociclo que era de **930 ps**. E o tempo de execução foi de **2790 ps**, executando também 2 atalhos que a máquina achou necessário fazer.

Estatísticas da simulação		×
Período de clock:	400 ps	
Frequência de clock:	2.50 GHz	
Ciclos executados:	7	
Tempo de execução:	2800 ps	
Instruções executadas:	3	
CPI:	2.33	
Atalhos:	2	
Protelamentos:	0	
Fechar		