Módulo 5
Pipeline: Princípios Fundamentais

A duração de ciclo de relógio (Tcc) de um processador com uma organização encadeada é determinada pela latência da lógica combinatória do estágio mais demorado somada com a latência do registo que preserva os resultados de cada estágio.

$T_{estagio\_i}$ – latência da lógica combinatória do estágio i

$T_{registo}$ – latência dos registos

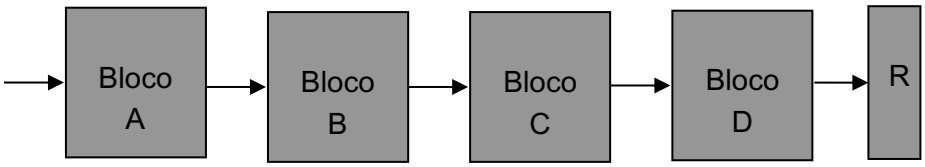
$T_{cc} = \max(T_{estagio}) + T_{registo}$

Assumindo que a instrução não é atrasada devido à ocorrência de anomalias, então:

- A **frequência do relógio** determina a taxa à qual o sistema pode mudar de estado. Se o CPI==1 então é igual ao débito de instruções, isto é, número de instruções executadas por unidade de tempo.
- **O tempo de execução de uma instrução** é o produto do número de estágios pelo período do relógio.

Exercício 1

Considere que a execução de instruções num processador pode ser decomposta em 4 blocos de igual duração (60 ps) conforme ilustrado na figura.

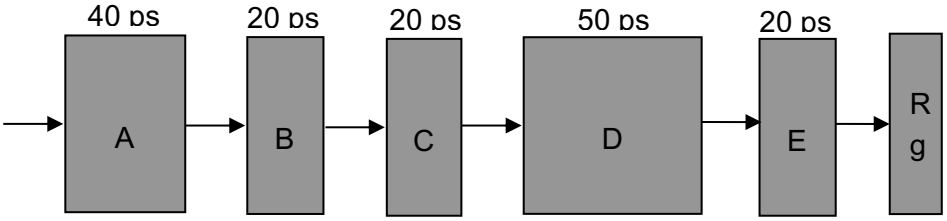


Sabendo que a latência dos registos é de 20 ps calcule o tempo de execução de uma instrução e a frequência máxima para uma organização de ciclo único (isto é, um *pipeline* degenerado num único estágio) e organizações com 2 e 4 estágios encadeados.

1 estágio (SEQ)		**2 estágios**		**4 estágios**	
Tcc		**Tcc**		**Tcc**	
Tinst		**Tinst**		**Tinst**	
f		***F***		***f***	

Exercício 2

Considere que a execução de instruções num processador pode ser decomposta em 5 blocos com a duração indicada na figura.



Sabendo que a latência dos registos é de 20 ps calcule:

i) Para uma organização encadeada com 2 estágios como devem ser agrupados os blocos para maximizar a frequência? Qual a frequência máxima do relógio possível para esta organização e o tempo de execução de cada instrução?

ii) Qual a máxima frequência que pode ser obtida e a quantos estágios corresponde.

Exercício 3

Pretende-se analisar o desempenho de um programa com 1000 instruções a executar nas organizações propostas abaixo.

Considere que a lógica combinatória de uma organização sequencial tem uma latência de 500 ps. Um bloco de registos tem uma latência de 20 ps. Considere também que a lógica que executa uma instrução pode ser dividida em qualquer ponto, permitindo sub-blocos com latências arbitrárias (exigindo-se apenas que a soma das latências de todos os sub-blocos combinatórios seja de 500 ps).

A partir das condições descritas acima pretende-se desenhar várias versões encadeadas, criando sub-blocos de lógica combinatória e acrescentando os registos necessários. Cada novo estágio de *pipeline* criado a partir da versão sequencial incorre em 2 custos:

1. tempo de registo e,
2. para este programa, 100 ciclos adicionais devido a dependências de dados e de controlo (causados por eventuais injecções de bolhas (*pipeline staling*)).

i) Para uma organização sequencial: qual a frequência máxima, qual o tempo de execução de uma instrução e qual o tempo de execução do programa?

ii) Para organizações com 2, 4 e 10 estágios calcule o tempo de execução deste programa.

iii) Não esquecendo nunca o custo associado ao *stalling* do *pipeline* , qual o número de estágios que minimiza o tempo de execução?
Sugestão: Preencha a tabela abaixo usando uma folha de cálculo.

#estágios	Tcc	#ciclos	Texec
1			
2			
4			
10			
13			
14			
15			
20			

Exercício 4

a) Considerando a sequência de código ao lado, identifique dependências de dados (RAW) e controlo.

b) Considere um processador com 4 estágios de *pipelining* (*Fetch, Decode, Execute* e *Writeback*) e que incorre um custo de 2 ciclos por cada dependência (*stalling*). Quantos ciclos do relógio São necessários para executar este programa?

c) Um processador idêntico mas com *data forwarding* (0 ciclos para dependências de dados) e previsão de saltos como **tomados** (0 ciclos para saltos bem previstos, 2 ciclos para erros de previsão) quantos ciclos do relógio necessita para executar este programa?

```
I1: mov $10,%eax
I2: cmpl $20, %eax
I3: jl I5
I4: add $20, %ecx
I5: add %eax, %ecx
```