# Aula 8 - Implementação de tarefas em sistemas pequenos

## Executivo cíclico

# Sumário

- Executivo Cíclico
- Tratadores de Interrupções
- Laço Principal com Tratadores de Interrupções
- Microkernel Simples

# Introdução

- Tempo de resposta de uma tarefa de tempo real está diretamente associado com a forma adotada para implementá-la
  - Depende da **organização dos fluxos de execução**
  - Do **design do software do sistema**
- Implementação de tarefas em sistemas computacionais simples pode ter as seguintes arquiteturas de software:
  - Executivo cíclico
  - Laço principal com interrupções
  - Microkernel simples

# Executivo Cíclico 1/19

- Os sistemas mais simples podem ser construídos de tal forma que existe apenas um único fluxo de controle no sistema
- Todo o sistema consiste de um grande laço que sempre é repetido periodicamente
- Tipo de solução chamada de **Executivo Cíclico** (***Cyclic Executive***)
- Trata-se de **escalonamento dirigido por tempo (*clock-driven scheduling*)**

# Executivo Cíclico 2/19

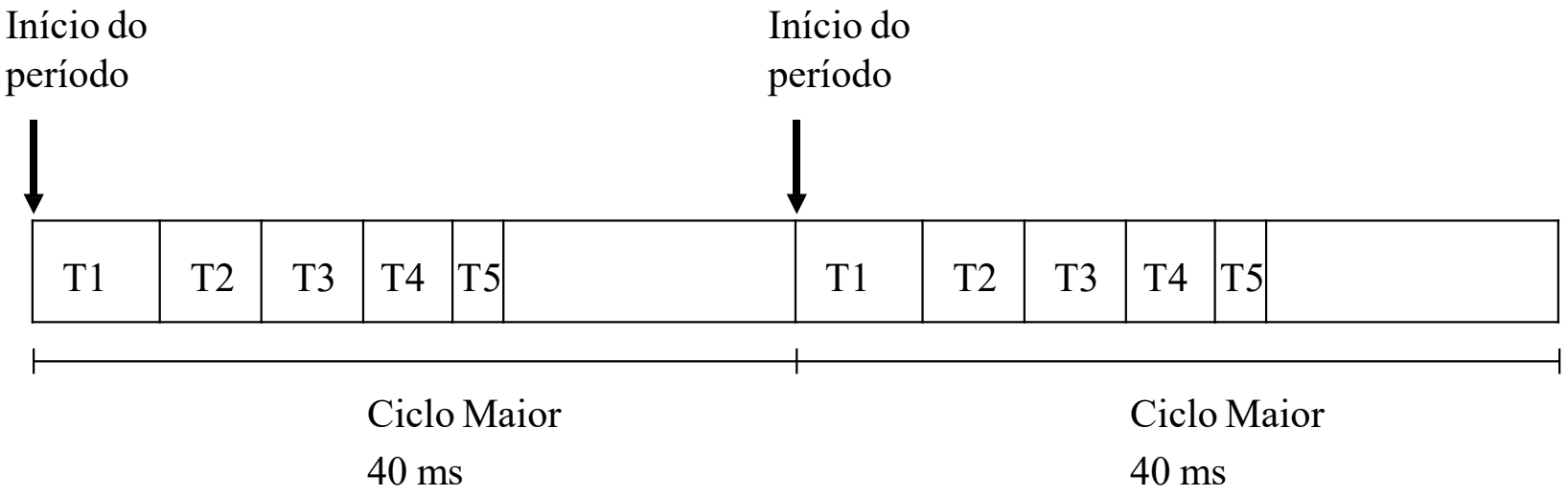
- O período de repetição do laço controlado através de um temporizador em hardware (*timer*)

```
CicloMaior = 40 ms
While( true ) {
      Espera_próximo_ciclo_maior_iniciar( );
      funcao_tarefa_1( );
      funcao_tarefa_2( );
      funcao_tarefa_3( );
      funcao_tarefa_4( );
      funcao_tarefa_5( );
}
```

# Executivo Cíclico 3/19

- Exemplo com 5 tarefas, cada uma mapeada para uma função
- O tempo de execução de cada tarefa pode ser diferente
- Porém todas elas executam com o mesmo período
  - 40ms no caso do exemplo
- É necessário garantir que a soma dos tempos de execução no pior caso de todas as tarefas seja menor que 40ms
- O período de tempo no qual todas as execuções se repetem é chamado de **Ciclo Maior** (***major cycle***)

# Executivo Cíclico 4/19



- Tarefas podem ter períodos diferentes
- Por exemplo, a tarefa que executa a estratégia de controle de um motor elétrico precisará executar com maior frequência (menor período) do que a tarefa que atualiza o display
- Suponha sistema com 5 tarefas, mas cujos períodos e tempos de execução no pior caso sejam:

| Tarefa τi | Período Pi | Tempo de computação no pior caso Ci |
|---|---|---|
| τ1 | 20 | 8 |
| τ2 | 20 | 7 |
| τ3 | 40 | 4 |
| τ4 | 40 | 3 |
| τ5 | 80 | 2 |

## Executivo Cíclico 6/19

- Períodos diferentes podem ser acomodados no executivo cíclico
- Através da divisão do ciclo maior em um número inteiro de ciclos menores
- Dentro de cada **Ciclo Menor** (***minor cycle***) apenas algumas tarefas executam
- A cada ciclo maior tudo se repete

# Executivo Cíclico 7/19

- O código abaixo utiliza 4 ciclos menores dentro do ciclo maior

```
CicloMenor = 20 ms
While( true ) {
     Espera_próximo_ciclo_menor_iniciar( );
     funcao_tarefa_1( );
     funcao_tarefa_2( );
     funcao_tarefa_3( );
     Espera_próximo_ciclo_menor_iniciar( );
     funcao_tarefa_1( );
     funcao_tarefa_2( );
     funcao_tarefa_4( );
     funcao_tarefa_5( );
     Espera_próximo_ciclo_menor_iniciar( );
     funcao_tarefa_1( );
     funcao_tarefa_2( );
     funcao_tarefa_3( );
     Espera_próximo_ciclo_menor_iniciar( );
     funcao_tarefa_1( );
     funcao_tarefa_2( );
     funcao_tarefa_4( );
}
```

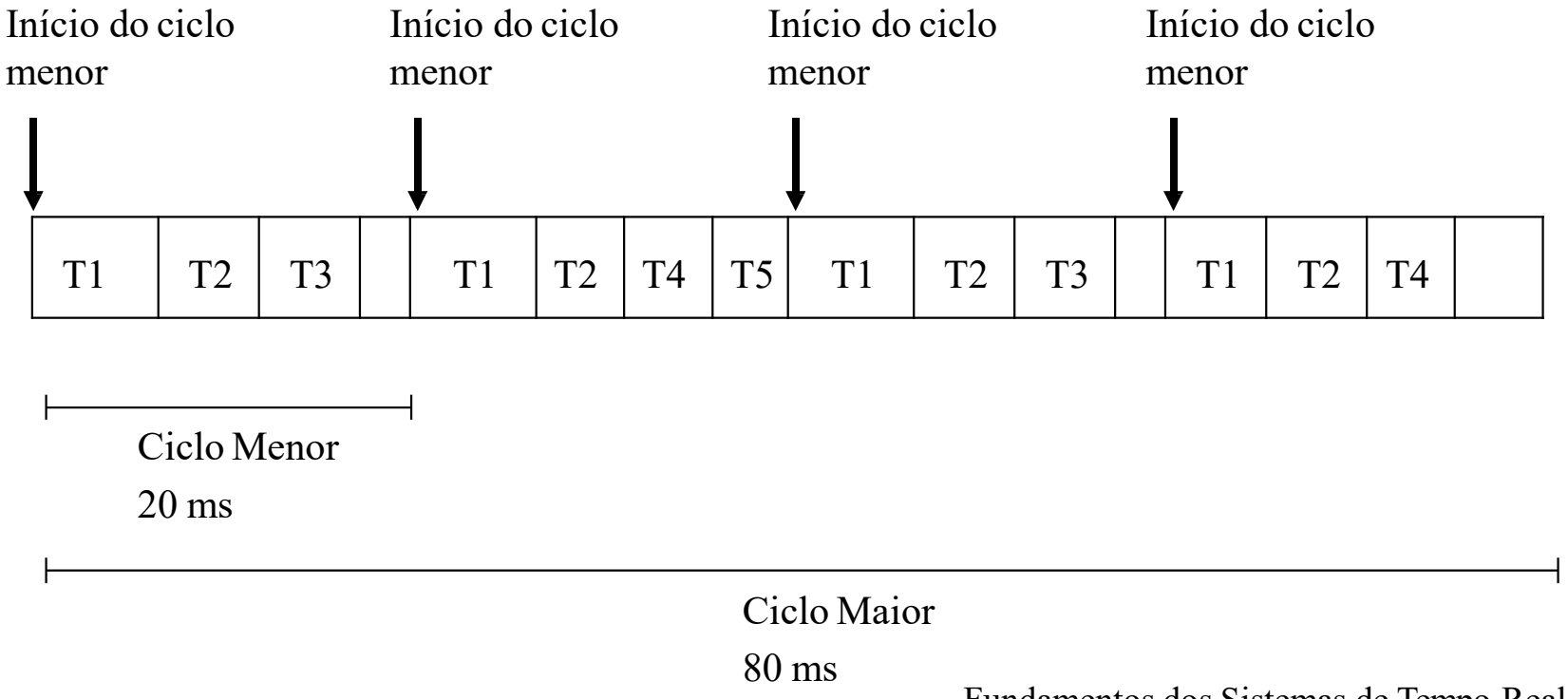
# Executivo Cíclico 8/19

- O período de cada tarefa é respeitado
- A tarefa $\tau 1$ é executada em todos os ciclos menores
  - Executada uma vez a cada 20ms, seu período
- A tarefa $\tau 5$ é executada em apenas um dos ciclos menores
  - Apenas uma vez a cada ciclo maior, pois seu período é 80ms
- Tipicamente o valor do ciclo maior corresponde ao **mínimo múltiplo comum** dos períodos das tarefas
  - **Hiperperíodo (*Hyperperiod*)**
- Uma duração conveniente para o ciclo menor é **o máximo divisor comum** dos períodos das tarefas
  - Ciclo menor (20ms) foi escolhido desta forma
  - Entretanto, esta não é a única solução possível

# Executivo Cíclico 9/19

- A soma dos tempos de execução no pior caso das tarefas contidas em cada ciclo menor não pode ultrapassar 20ms
- A computação em cada ciclo menor totaliza 19ms e 18ms

| Tarefa τi | Período Pi | Tempo de computação no pior caso Ci |
|---|---|---|
| τ1 | 20 | 8 |
| τ2 | 20 | 7 |
| τ3 | 40 | 4 |
| τ4 | 40 | 3 |
| τ5 | 80 | 2 |



## Executivo Cíclico 10/19

- Uma tarefa com tempo de computação grande pode dificultar esta divisão
- Neste caso, pode ser necessário adaptar as tarefas
- Uma tarefa com período Pk e tempo de computação Ck talvez possa ser dividida em duas partes
- Duas novas tarefas com períodos Pk e
  tempo de computação Ck/2 cada uma delas
- Este tipo de divisão facilita a organização dos ciclos menores

## Executivo Cíclico 11/19

- A função *Espera_próximo_ciclo_menor_iniciar( )* representa a espera pelo final do ciclo menor corrente
- Pode usar um laço que fica constantemente lendo um relógio de tempo real no hardware
- Ou usar uma interrupção de temporizador em hardware (*timer*) previamente programado
- De qualquer forma, o tempo gasto nesta função deve ser somado ao tempo total de execução do ciclo menor

## Executivo Cíclico 12/19

- É usual sobrar algum tempo de computação em cada ciclo menor
- No caso do exemplo anterior, temos sobras de 1ms no primeiro e terceiro ciclos menores, e uma sobra de 2ms no quarto ciclo menor
- Este tempo pode ser aproveitado para a execução de tarefas que não são de tempo real e aproveitam o tempo restante
- Tempos previstos no executivo cíclico são para o pior caso
- No caso típico as tarefas executam em menos tempo
  - Sobra ao final de cada ciclo menor será ampliada com este tempo adicional
- Todo tempo de processamento reservado para uma tarefa porém não utilizado por ela é chamado de **tempo ganho** (***gain time***)
- Pode ser usado por outras tarefas

## Executivo Cíclico 13/19

- Caso o tempo alocado para o ciclo menor tenha terminado, mas ainda existe código de tarefa para executar, é dito que ocorreu um ***overrun*** do ciclo menor
- Uma abordagem simples é continuar a execução das tarefas
  - Na esperança de que as folgas existentes nos ciclos menores em sequência recomponham a corretude temporal do sistema
- No caso de sistemas críticos, a ocorrência do *overrun* representa uma grave **falta temporal** (***timing fault***)
- Algum tipo de tratamento de falta será necessário
  - Sinalizar a ocorrência da falta no painel do equipamento
  - Enviar mensagem para o fabricante
  - Reinicializar o equipamento
  - Mudar para um modo de segurança onde ações físicas perigosas do equipamento são suspensas

# Executivo Cíclico 14/19

- Planejamento da execução das tarefas em ciclos menores e ciclo maior pode ser feita manualmente no caso de sistemas pequenos e simples
- Como a escala de execução é construída em tempo de projeto, algoritmos mais complexos também podem ser usados
- Exemplo: meta-heurísticas tais como Busca Tabu, Algoritmos Genéticos e Recozimento Simulado (*Simulated Annealing*)

- Podem existir objetivos secundários
  - Distribuir as folgas uniformemente entre os ciclos menores favorece o tratamento de *overrun*
  - Minimizar a variação do intervalo de tempo entre términos consecutivos de uma mesma tarefa

# Executivo Cíclico 15/19

- A maior vantagem do executivo cíclico é a sua simplicidade
- As tarefas são implementadas como simples funções
- Facilitando a gerência do processador em sistemas pequenos e simples
  - Sistema operacional não é usado devido às limitações do hardware

- Fácil verificar se todas as tarefas cumprem os seus respectivos requisitos temporais
- Basta inspecionar a escala de execução gerada

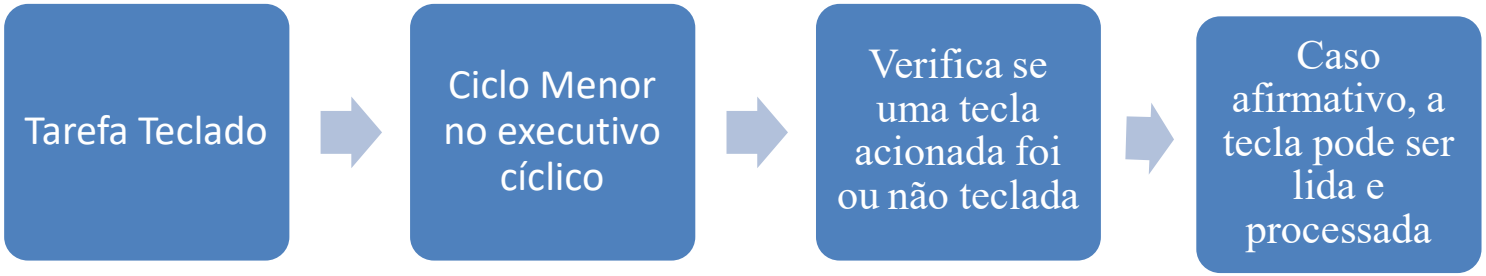
## Executivo Cíclico 16/19

- Executivo cíclico apresenta diversas limitações
- Períodos das tarefas devem ser múltiplos do tempo de ciclo menor
- É difícil incorporar tarefas com períodos longos
- Tarefas que atendem a situações de emergência e precisam executar imediatamente não são facilmente posicionadas na escala de execução

- O executivo cíclico funciona bem quando todas as tarefas possuem um deadline relativo igual ao período
- Tarefas esporádicas, ou tarefas periódicas com deadline relativo menor que o período, são difíceis de serem acomodadas
- Quando feito, geram ociosidade no sistema

# Executivo Cíclico 17/19

- A programação das tarefas deve ser muito cuidadosa, pois nenhuma tarefa pode ficar bloqueada esperando por algum evento
- Considere o exemplo de um teclado.




- Não existe algo como uma função *scanf()* que a tarefa possa chamar e ficar bloqueada até que algo seja teclado

# Executivo Cíclico 18/19

- Quando uma tarefa espera pela realização de uma operação de entrada ou saída a mesma é chamada de **entrada e saída síncrona**
- É desta forma que a maioria dos programas são construídos
- Forma mais simples de programar, mais legível, menos sujeita a erros

- No caso do executivo cíclico é necessário empregar operações de **entrada e saída assíncronas**
  - Tarefa não fica bloqueada esperando por elas
- Para saber se uma dada operação de entrada ou saída comandada antes foi concluída, é necessário ler alguma variável ou registrador
- Programação com entrada e saída assíncrona é mais difícil
  - Mais sujeita a erros
  - Manutenção mais custosa

# Executivo Cíclico 19/19

- Executivo cíclico puro não inclui tratadores de interrupções
  - No máximo um tratador de interrupção para o temporizador no hardware que sinaliza o início de um ciclo menor
- Ações urgentes não podem ser realizadas por tratadores de interrupções
- Todos os sensores precisam ser amostrados por tarefas, como previsto na escala de execução

- Resposta aos eventos externos urgentes bem mais lenta do que seria com tratadores de interrupções

# Exercícios

| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |

Mímimo múltiplo comum dos períodos = 100

Ciclo maior = 100 ms

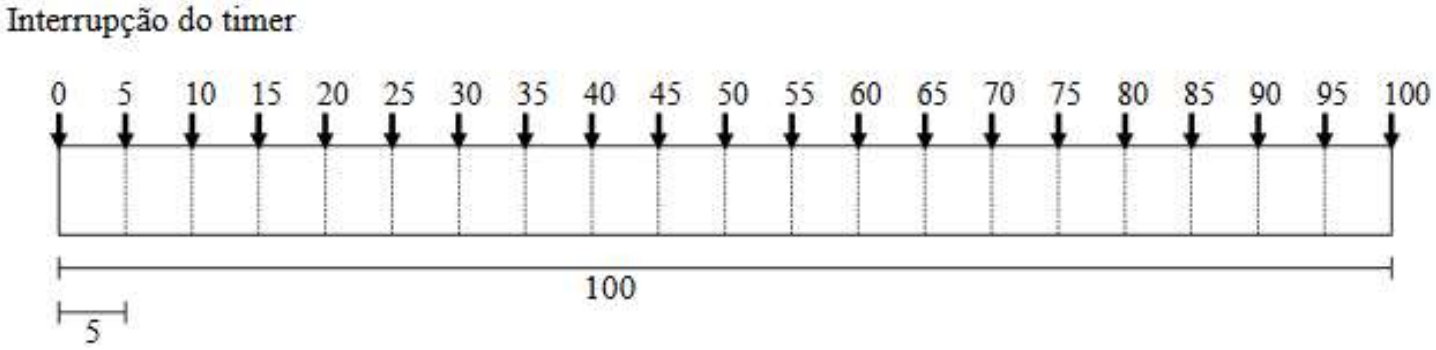
Máximo divisor comum dos períodos = 5

Ciclo menor = 5 ms

Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

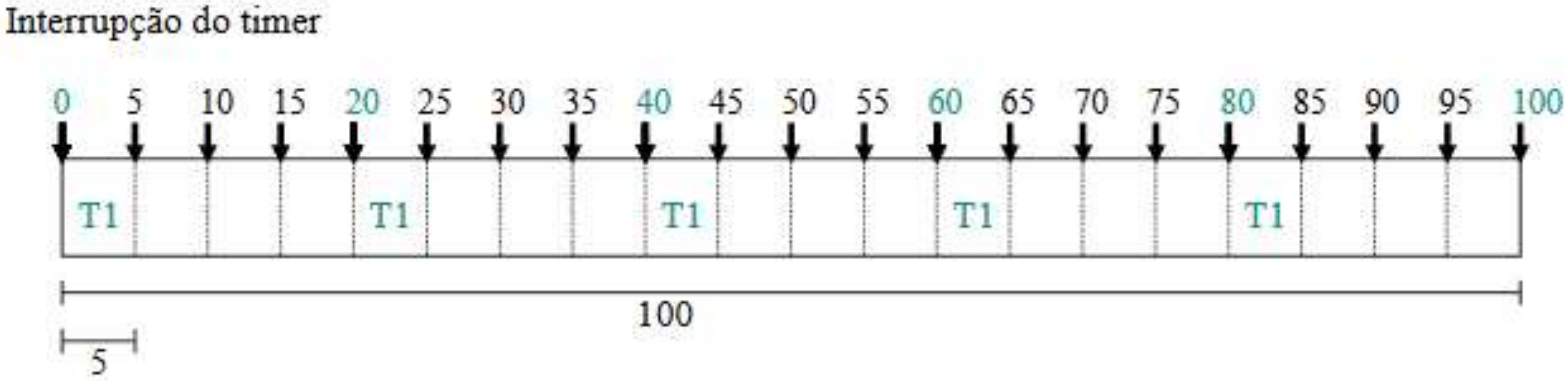
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

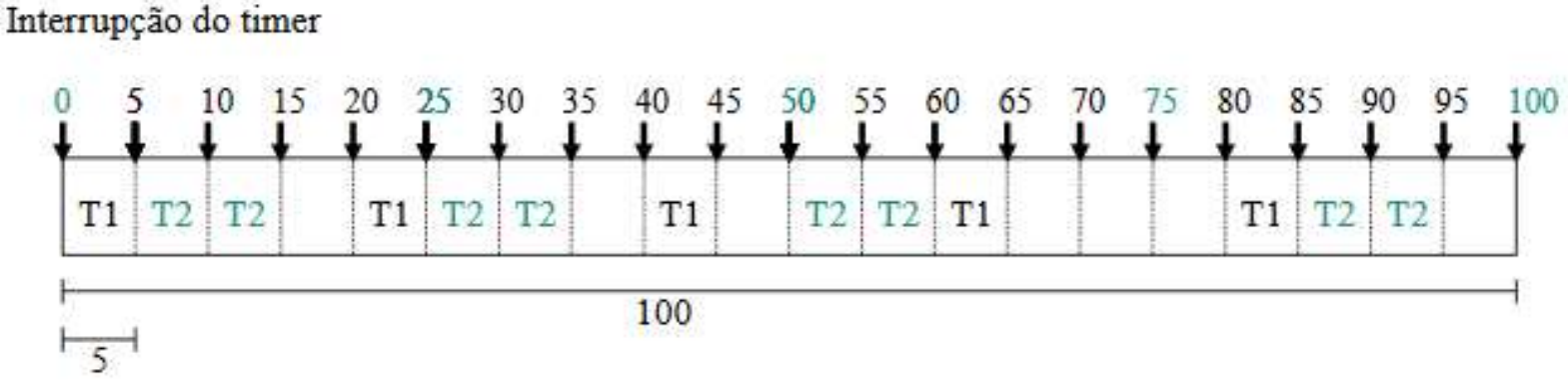
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

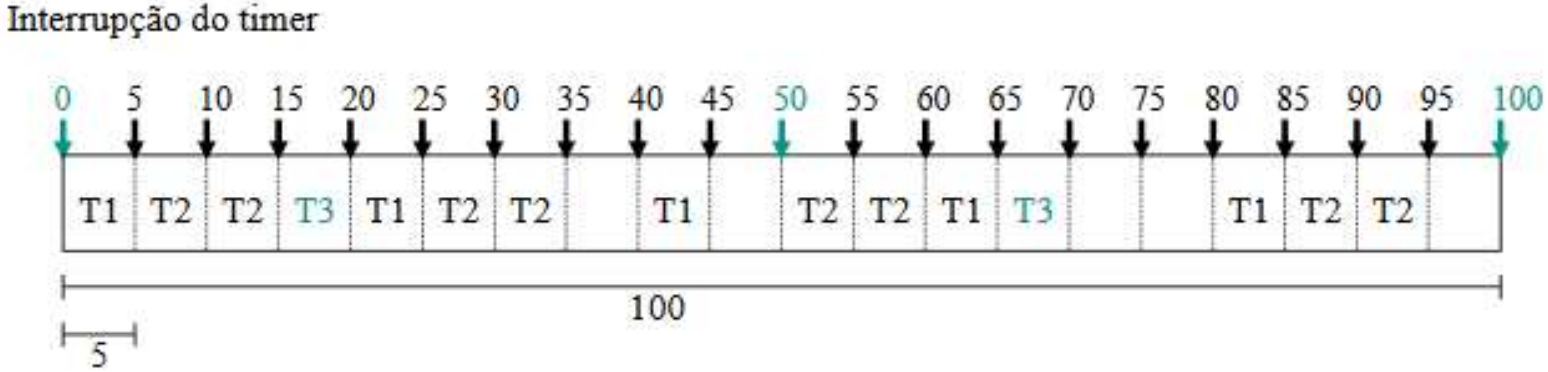
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

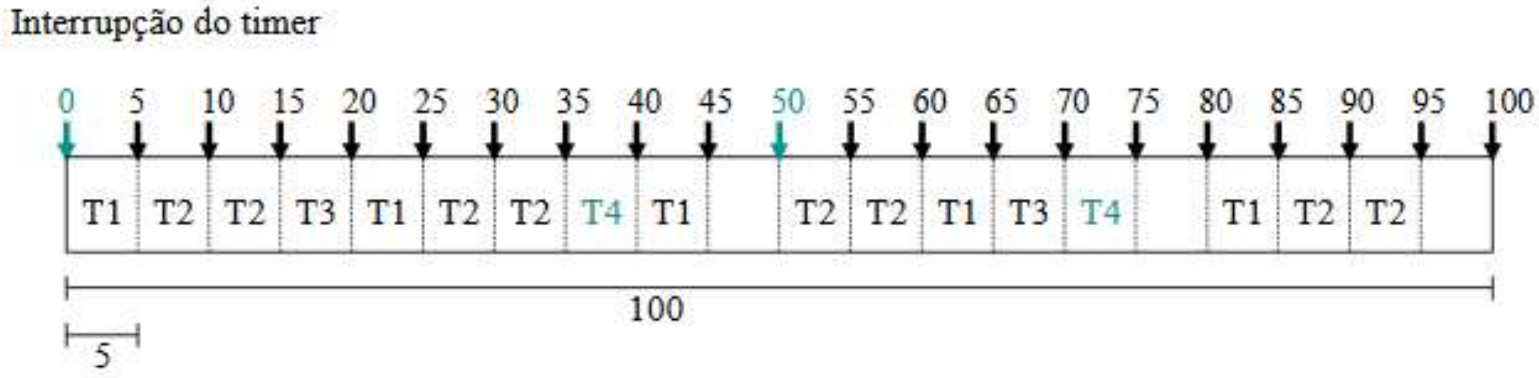
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

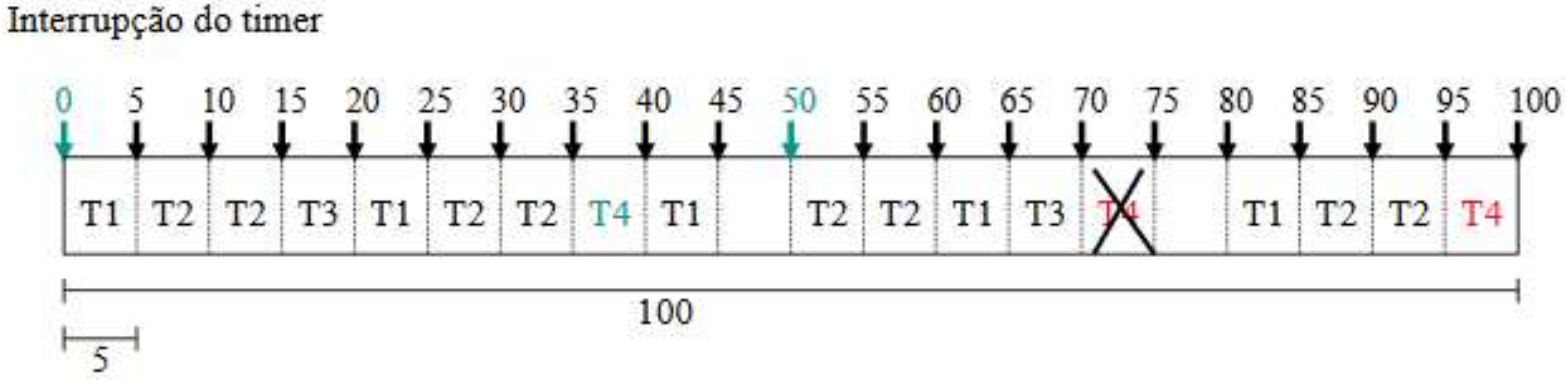
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

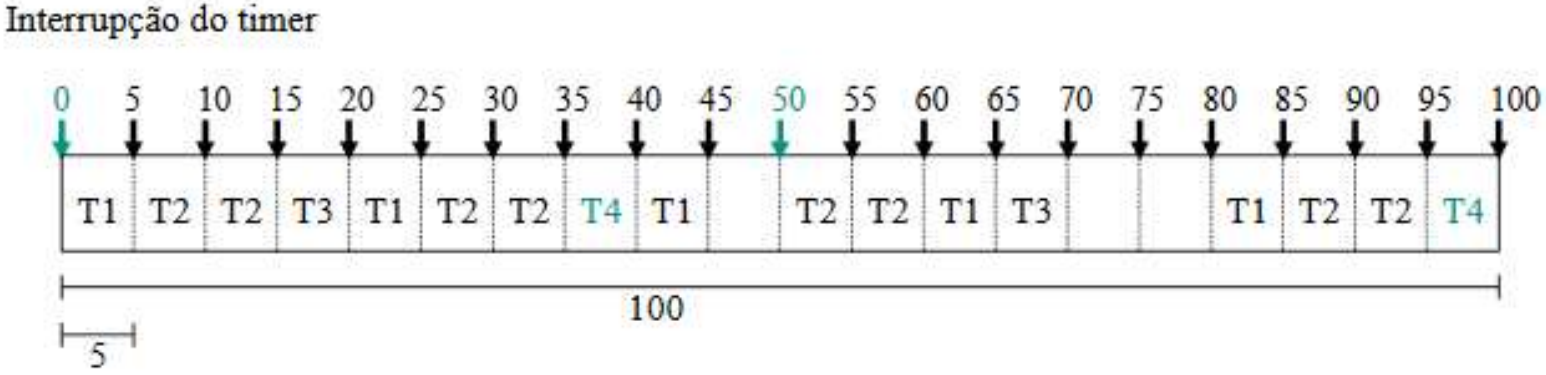
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

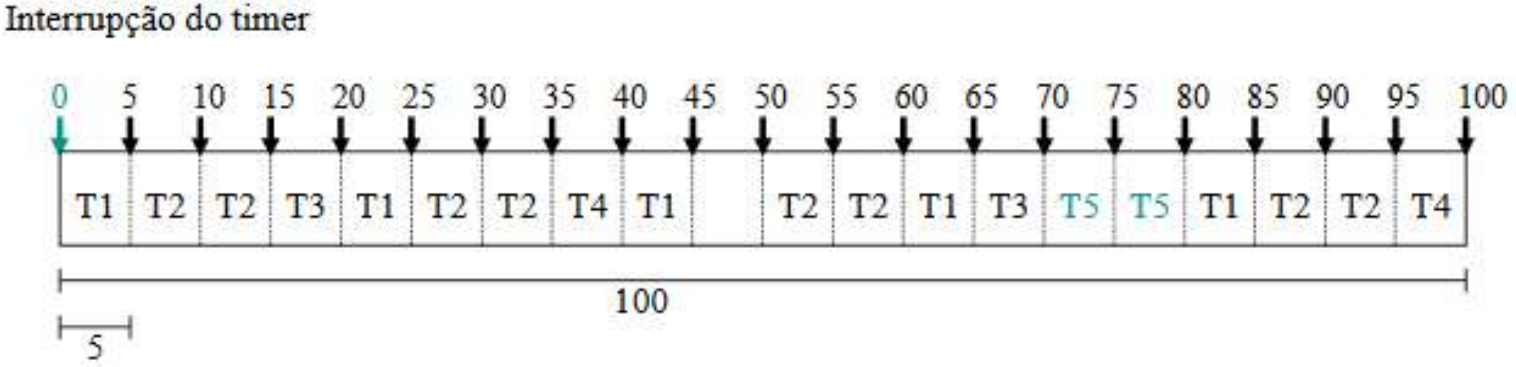
| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

# Exercícios

| Tarefa Ti | Período Pi | Tempo de computação Ci |
|---|---|---|
| T1 | 20 ms | 5 ms |
| T2 | 25 ms | 8 ms |
| T3 | 50 ms | 5 ms |
| T4 | 50 ms | 4 ms |
| T5 | 100 ms | 10 ms |




Fonte:http://www.romulosilvadeoliveira.eng.br/discipli/cad-tr1/Tr6x1-ExecutivoCiclico-2010b.pdf

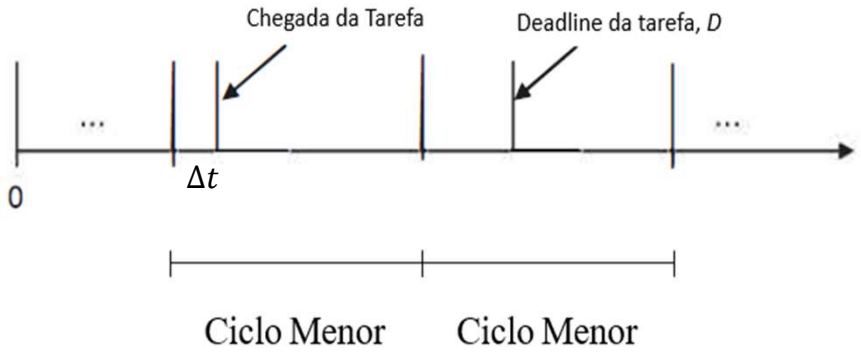
# Executivos Cíclicos – Premissas mais precisas

- Premissas mais precisas para determinar o Ciclo Menor (CM)

  *Dado uma tarefa* $T_i = (P_i, D_i, C_i)$

  - Premissa 1
    - $CM \geq \max\{C_i\}$ $(maior\ tempo\ de\ computação\ entre\ as\ tarefas)$
  - Premissa 2
    - O CM deve dividir o *ciclo maior obtido pelo mmc dos períodos*
  - *Premissa 3*
    - $CM \leq P_i$
  - *Premissa 4*
    - $2CM - \Delta t \leq D_i$
    - $2CM - MDC(P_i, CM) \leq D_i,$

    $onde\ MDC$ é $o\ máximo\ divisor\ comum.$



# Executivos Cíclicos – Premissas mais precisas

- *Exemplo: Considere um sistema de três tarefas periódicas:*
  - $T_1 = (4, 4, 1), T_2 = (5, 5, 1), T_3 = (10, 10, 2)$
- *Solução:*
- *Premissa 1*
  - $CM \geq 2$
- *Premissas 2 e 3*
  - *Pelo mmc, o ciclo maior é 20*
  - *Os tamanhos possíveis para CM considerando também a premissa 3*
    - *2, 4, 5 e 10*
- *Premissa 4*

| | $CM = 2$ | | $CM = 4$ | | $CM = 5$ | | $CM = 10$ | |
|---|---|---|---|---|---|---|---|---|
| | $2CM - MDC(P_i, CM) \leq D_i$ | $D_i$ | $2CM - MDC(P_i, CM) \leq D_i$ | $D_i$ | $2CM - MDC(P_i, CM) \leq D_i$ | $D_i$ | $2CM - MDC(P_i, CM) \leq D_i$ | $D_i$ |
| T1 | 2*2-MDC(4,2)=4-2=2 | 4 | 2*4-MDC(4,4)=8-4=2 | 4 | 2*5-MDC(4,5)=10-1=9 | 4 | 2*10-MDC(4,2)=20-2=18 | 4 |
| T2 | 2*2-MDC(5,2)=4-1=3 | 5 | 2*4-MDC(5,4)=8-1=7 | 5 | | | | |
| T3 | 2*2-MDC(10,2)=4-2=2 | 10 | | | | | | |