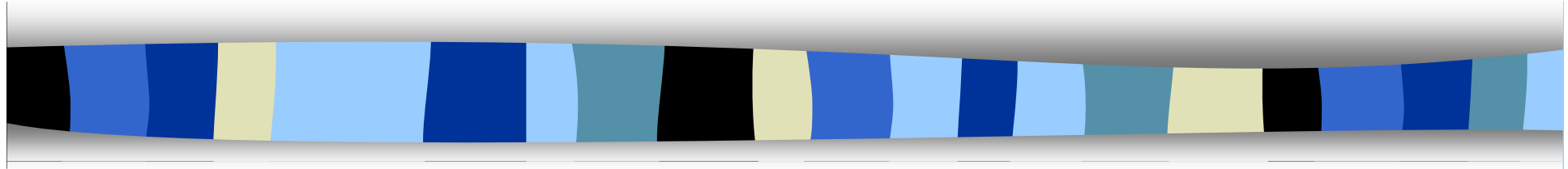


Universidade Federal do Amazonas
Faculdade de Tecnologia
Programa de Pós-graduação em Engenharia Elétrica



Arquiteturas de Software



Lucas Cordeiro

lucascordeiro@ufam.edu.br

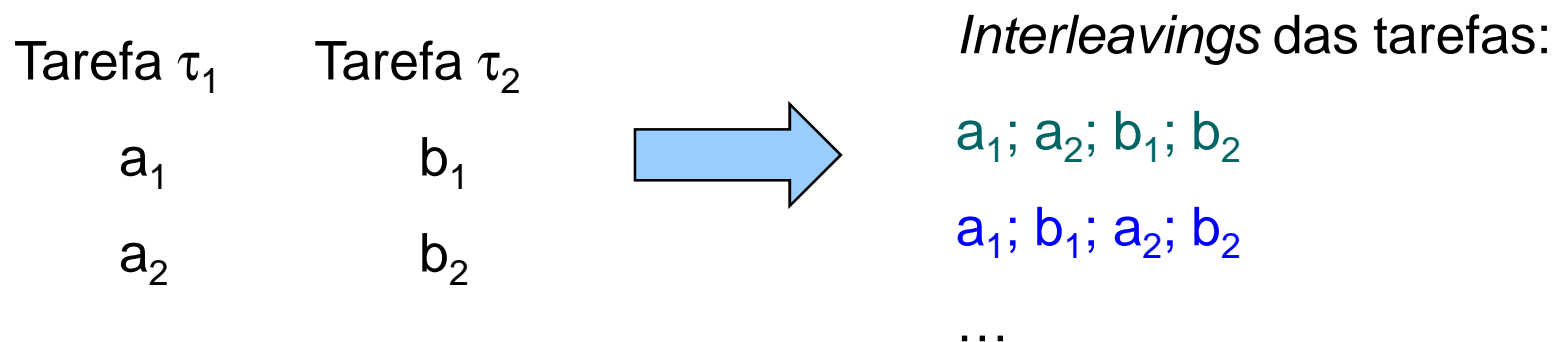
Notas de Aula



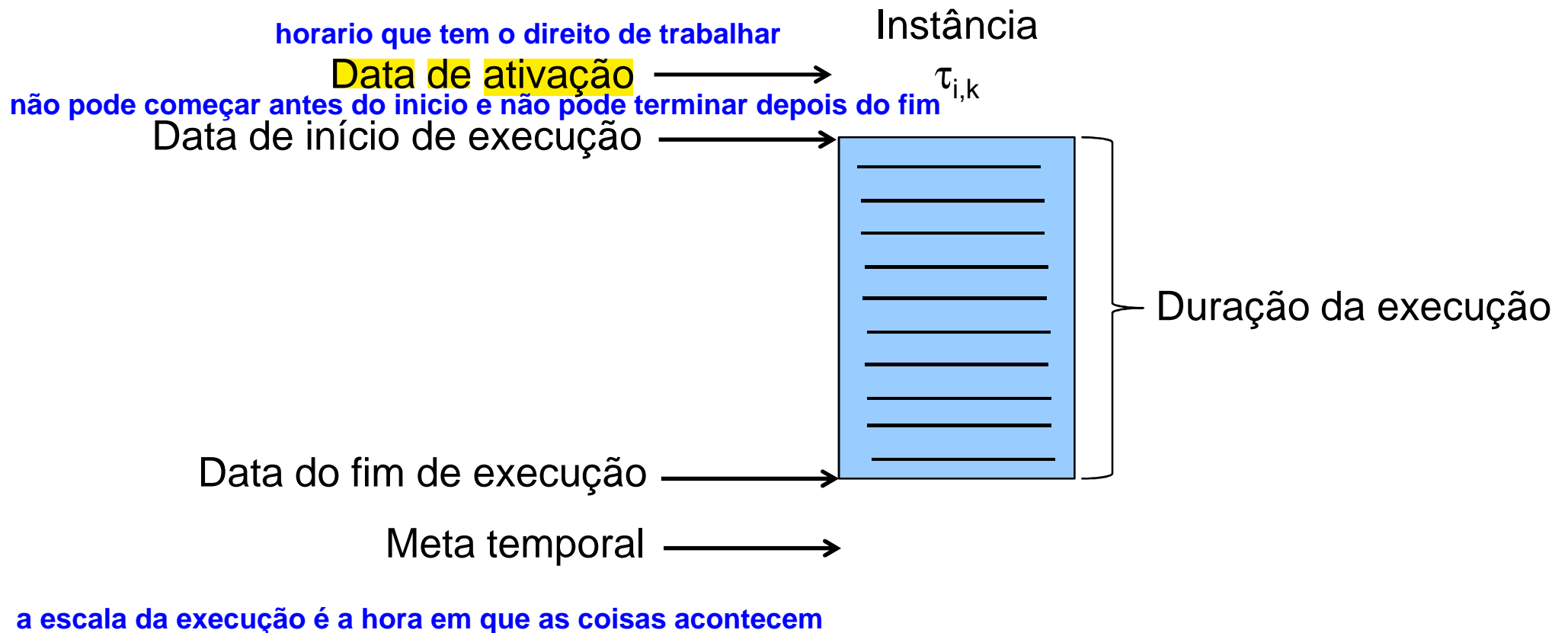
- Estes slides são baseados no livro do Prof. Alan Shaw e nas notas de aula Prof. Francisco Vasques.
<http://www.fe.up.pt/~vasques>
- Os slides estão disponíveis em:
<http://users.ecs.soton.ac.uk/lcc08r/disciplinas/str/>

Conceitos e Definições (1)

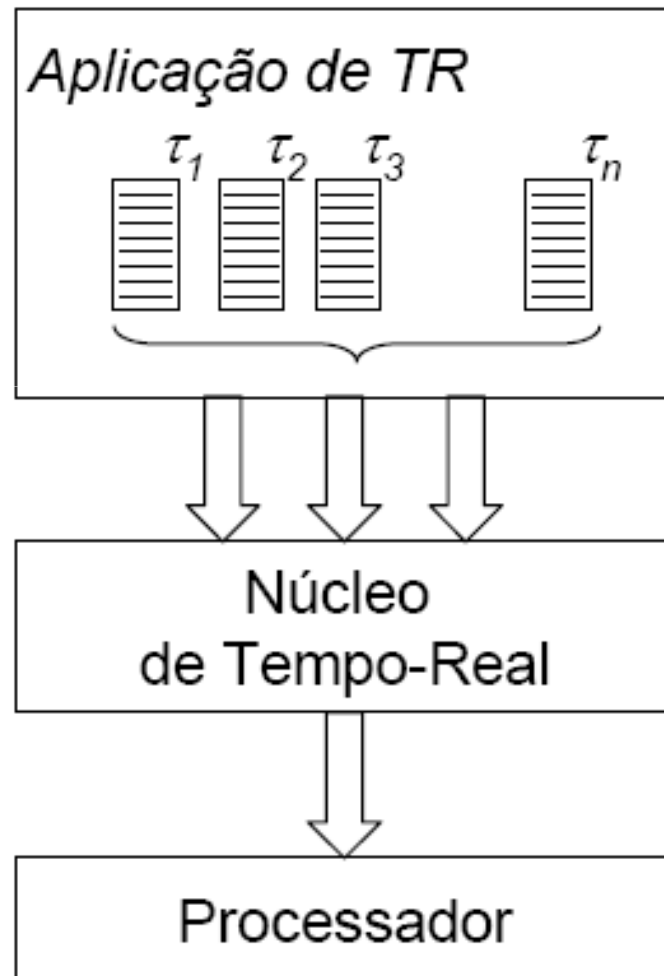
- Programas em Sistemas de Tempo-Real
 - ... constituídos por *tarefas*
- Conceito de Tarefa: *cada pedaço do código teria uma tarefa (uma espécie de thread de Tempo real)*
 - Tarefa (τ_i): Segmento de código de software que deverá ser executado múltiplas vezes com diferentes dados de entrada
 - Uma tarefa é caracterizada por ter uma execução com *características temporais próprias*
 - Instância (“job”) ($\tau_{i,k}$) é a execução k da tarefa τ_i



Conceitos e Definições (2)

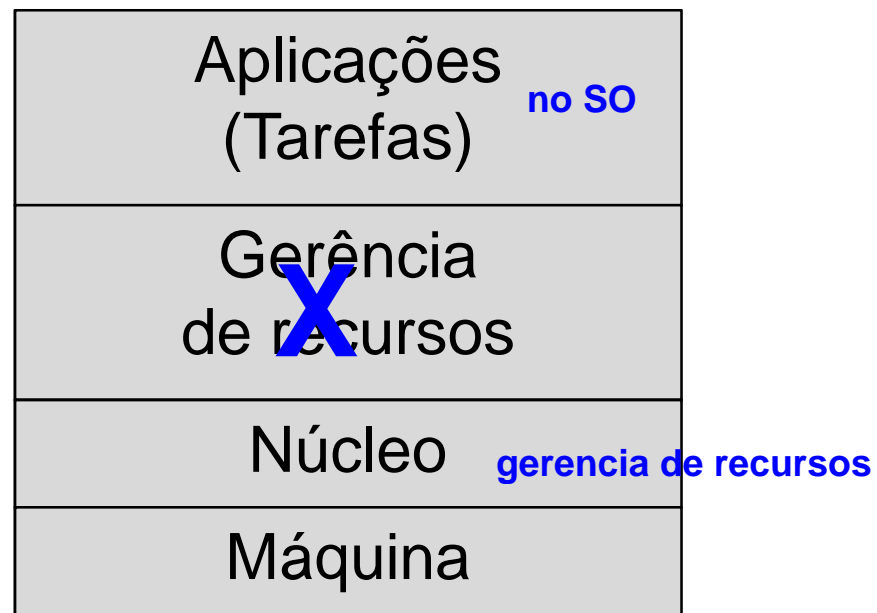


Núcleo de Tempo-Real (1)



- O núcleo de TR implementa:
 - ilusão de concorrência tempo compartilhado
 - métodos de sincronização p/garantir o uso eficiente dos recursos compartilhados e garantir a integridade
 - chaveamento e escalonamento
 - serviço de medição de tempo precisa ter uma granularidade e precisão para sincronização c/soft
 - tratamento de interrupções
- Gerência de recursos:
 - gerência de memória
 - controle de dispositivo de ES
 - alocação de recursos

Núcleo de Tempo-Real (2)



Núcleo de Tempo-Real (3)



- Objetivo: **transparência** para a aplicação de tempo-real do processador e dos seus mecanismos de interface
- Permitir a execução de múltiplas tarefas num ambiente **pseudo-paralelo**, garantindo o respeito das metas temporais associadas a cada uma das tarefas
- Pseudo-paralelismo: O processador executa *sucessivamente múltiplas tarefas*
 - Se as diferentes tarefas são executadas sequencialmente: **escalonamento não preemptivo**
 - Se uma tarefa em curso de execução pode ser interrompida por uma outra tarefa: **escalonamento preemptivo**

Núcleo de Tempo-Real (4)

- Porquê utilizar um núcleo de TR?
 - Desenvolvimento mais rápido das aplicações (**ganho de produtividade**), visto o compartilhamento do processador ser transparente
 - Utilização de espaço **otimizado de memória**, com *menor custo em termos de tempo de execução*
- Critérios para a seleção de um núcleo de tempo-real
 1. Hardware-alvo suportado
 2. Linguagens de programação suportadas (p.e. C, C++)
 3. Dimensão de memória ocupada (“*footprint*”)

Núcleo de Tempo-Real (5)



4. Serviços fornecidos pelo núcleo

- **Escalonamento:** adaptado para o suporte de aplicações de tempo-real?
- **Sincronização entre tarefas:** Que tipo de gestão de filas (FIFO, por prioridades)? Que mecanismos para gerir a inversão de prioridades?
- **Gestão de memória:** estática ou dinâmica?
- **Gestão do tempo:** ativação de tarefas periódicas? Qualidade das temporizações (granularidade, precisão)? **granularidade = precisão**
- Interruptibilidade do núcleo? Quais os serviços mínimos que têm que ser incluídos numa versão reduzida do núcleo?

5. Componentes de software fornecidos além do núcleo

- Pilhas de protocolos, base de dados de tempo-real, serviços Web? **acesso a rede, tempo de relógio**

Núcleo de Tempo-Real (6)

6. Desempenho (“*performance*”):

- A existência de análises efetuadas pelos fabricantes devem ser examinados com base nos seguintes elementos:
 - Qual a plataforma para a qual foram efetuadas?
 - Em que condições de medida?
 - Tempos médios, máximos ou mínimos?
 - Presença de interrupções?

7. Existência de **drivers** para os periféricos

8. Qualidade do suporte técnico, para futura evolução para novas arquiteturas **ex. trabalhar num software de código aberto p/windows;**

9. Natureza do produto (código fonte ou binário?)

limitado se tivervos acesso a partes limitadas do código. que impacta no preço final do produto

10. Custo (preço por licença?)

11. Certificação para a área de aplicação pretendida?

obs.: sist. Op c/versões diferentes p/diferentes aplicações, como p/so p/veículo e SO p/avião ...

Núcleo de Tempo-Real (7)

■ Objetivo de um Núcleo (“*Kernel*”) de *Tempo-Real*

- O elemento fundamental de um núcleo (“*Kernel*”) de *tempo-real* é o **escalonador**, que tem como função permitir a execução de múltiplas tarefas num ambiente pseudo-paralelo, garantindo o respeito das suas metas temporais

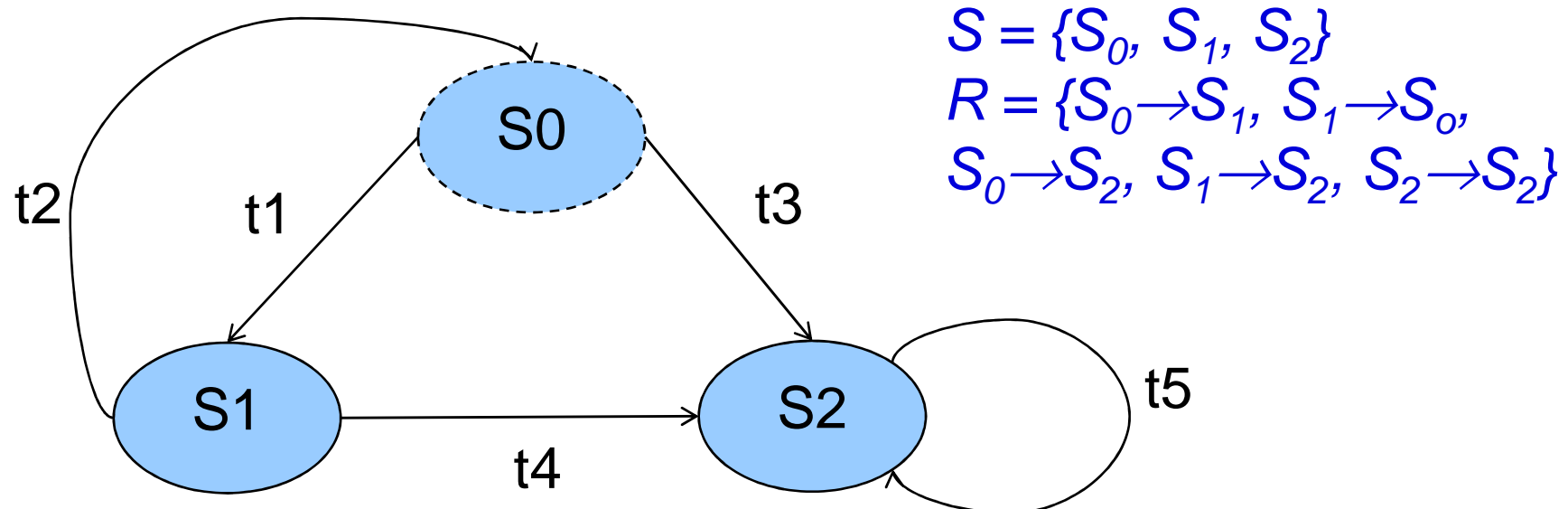
■ Escalonamento (“*Scheduling*”) de Tempo-Real

escalonamento injusto e consequentemente menos eficiente

- O escalonamento de tempo-real pretende atribuir de uma forma ótima ao processador às tarefas que os pretendem utilizar, garantindo o respeito das metas temporais (“*deadlines*”) associadas à execução de cada uma das tarefas
há várias formas de fazer o escalonamento

Estados de uma Tarefa (1)

- **Definição:** Um sistema de transição de estados, denotado por M , é definido por uma tripla (S, R, S_0) onde S representa o conjunto de estados, $R \subseteq S \times S$ representa o conjunto de transições e $S_0 \subseteq S$ representa o conjunto de estados iniciais (i.e., cada elemento de S_0 é também elemento de S)

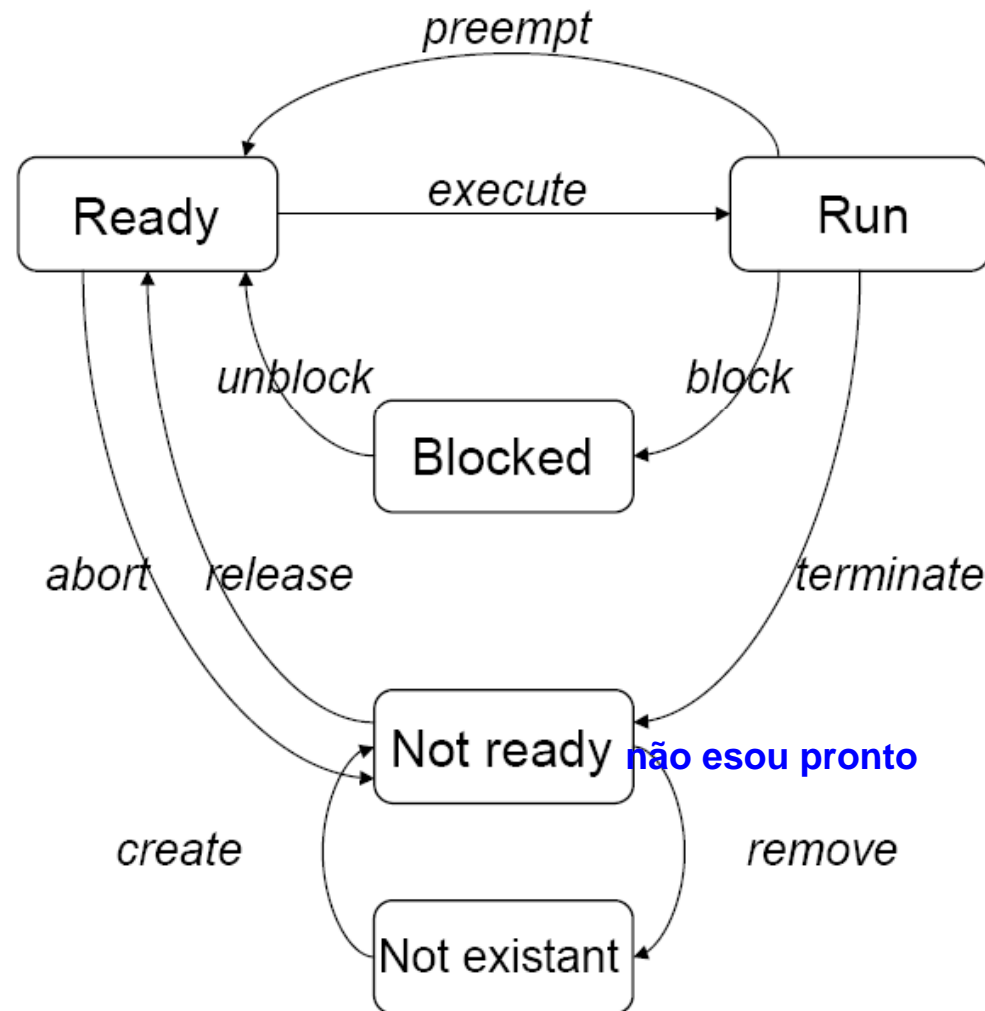


Estados de uma Tarefa (2)

máquina de estados de um STR típico

- Durante a sua execução, uma tarefa de uma aplicação de tempo-real pode estar num, e num só, dos seguintes estados

bloqueada : entrada e saída, sleep, ...



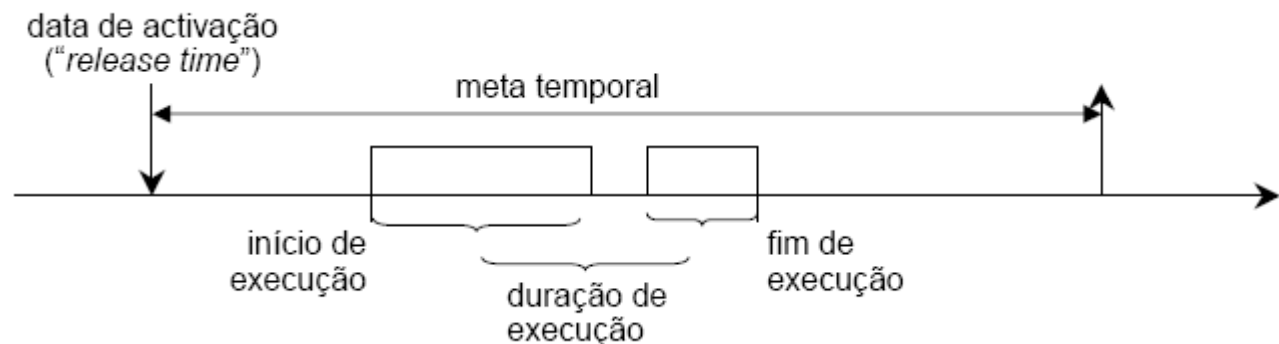
Escalonamento de Tempo Real

comumente teremos mais tarefas q processadores daí precisamos organizar os escalonadores

- Um algoritmo de escalonamento de tempo-real deve:
 - Selecionar entre as **tarefas pendentes** (“ready”), qual a que deve ser executada
 - Verificar se deve **interromper a tarefa** que está a ser executada (“*preemption*”), para permitir a execução de uma tarefa de maior prioridade
 - Verificar se as **regras de execução** impõem o bloqueio (“*blocking*”) da tarefa que está a ser executada
 - **Otimizar a execução** do conjunto de tarefas (tipicamente existem mais tarefas sendo executadas do que processadores disponíveis).

Modelo de Tarefas (1)

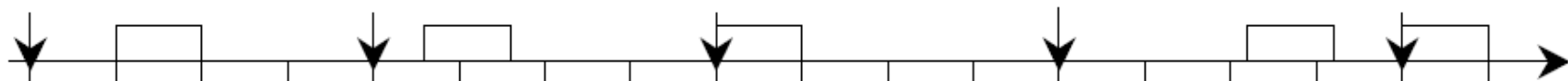
- Parâmetros fundamentais de um modelo de tarefas, que permita descrever de uma forma adequada as *propriedades temporais de um sistema de tempo-real*:
 - *Periodicidade de ativação de cada tarefa*: P_i
 - *Máxima duração de execução de cada tarefa (sem preempção)*: C_i
 - *Meta temporal (“deadline”) associada à execução de cada tarefa*: D_i
 - *Tempo de ativação da instância k da tarefa τ_i* : $r_{i,k}$
 - *Tempo de início de execução da instância k da tarefa τ_i* : $s_{i,k}$
 - *Tempo de fim de execução da instância k da tarefa τ_i* : $f_{i,k}$



Modelo de Tarefas (2)

- **Tarefas periódicas** são ativada de forma **regular** entre **intervalos fixos** de tempo (monitoramento sistemático, *polling*)
 - ler a temperatura de um líquido a cada 50 milisegundos
- **Tarefas esporádicas** são dirigidos por **eventos**; eles são ativados por um sinal externo ou uma mudança de alguma relação (reagir a um evento indicando uma falha de alguma peça do equipamento) *ex.: ativar um culler* *são mais difíceis de tratar pois não há uma regularidade*
 - Tratar as notificações de chegada através do sistema de comunicações

o problema é deixar a CPU esperando pelo evento, faz-se isso p/dar garantia porém reduz eficiência dos sistema



Executivos cíclicos (1)

é a primeira(primórdios) abordagem de STR

não há escalonadores

- Historicamente é o enfoque mais usado para organizar software de tempo-real
- Só tarefas periódicos
 - Tradução de tarefas esporádicas em periódicas
 - Assumir tempo de execução negligenciável
- Objetivo: tarefas satisfazendo restrições de *prazos-limite*
- Solução: predefinir, explicitamente, e antes da execução, uma intercalação (escala) de execução de tarefas que produza um escalonamento viável de execução

não há escalonadores, o escalonamento é feito a mão, p/isso quebra-se as tarefas em tarefas menores

VANTAGENS: muito simples implementar

DESVANTAGENS: MUITO Rígida se houver alguma alteração pode-se invalidar e será necessário refazer a escala

Executivos cíclicos (2)



- Um programa **executor cíclico** (despachante) controla a execução das tarefas de acordo com esse escalonamento de pré-execução (*pre-runtime scheduling*)
- Vantagens:
 - Simples
 - Eficiente
 - Altamente previsível
 - Menos *overheads*
- Desvantagens:
 - Muito baixo nível
 - Inflexível
 - Difícil projetar, alterar e manter os sistemas

Definições e propriedades (1)

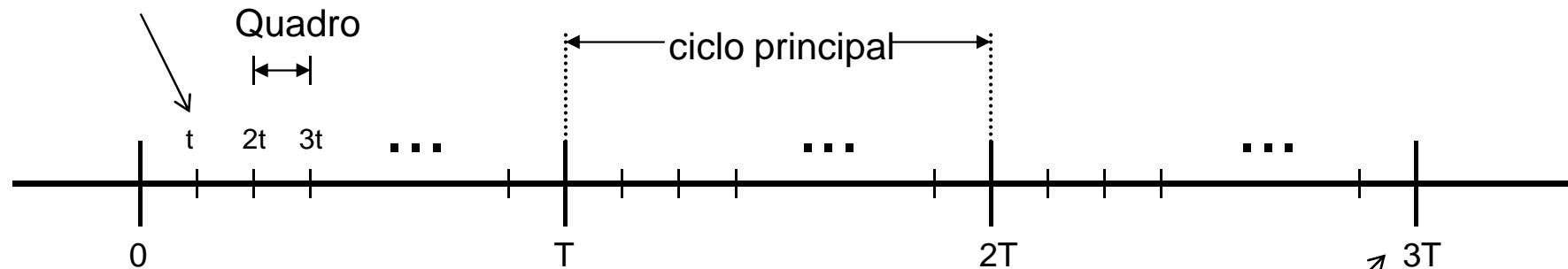
- Código do programa S para um processo periódico é quebrado em uma seqüência (ou blocos)
 - $S = S_1; S_2; \dots; S_n$
 - cada S_i poderia ser uma chamada de sub-rotina, uma seqüência de comandos sem desvios ou um laço
- Cada bloco é não-preemptível e tem um WCET (*worst-case execution time*)
- Escalonamento principal: alocação de blocos tal que *prazos-limite* e períodos sejam satisfeitos
- É cíclico porque pode ser repetido
- É usual que o tempo de ciclo principal (TCP) ou *macro-ciclo* seja o MMC entre os períodos de todos os processos

Definições e propriedades (2)

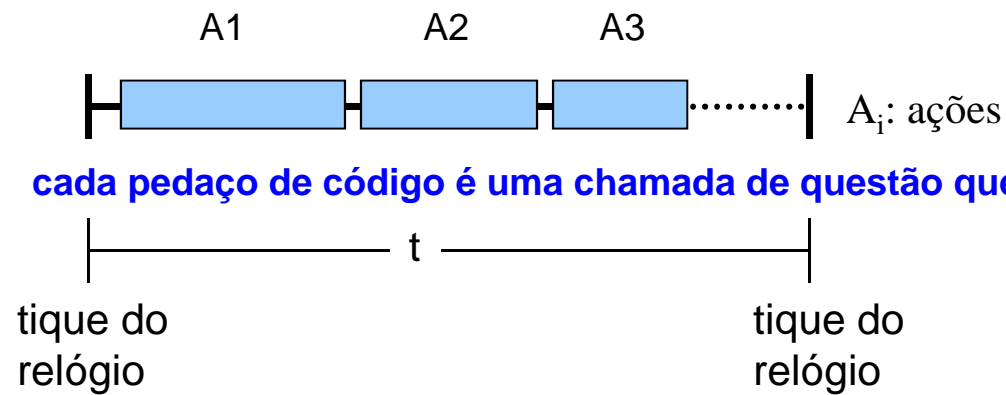
- Um escalonamento principal é dividido em ciclos secundários (ou quadros)
- O tempo do quadro é chamado de tempo de ciclo secundário (**tcs**) ou *micro-ciclo*
- Blocos são alocados nos quadros
- Controle de tempo é definido e imposto apenas nos limites de quadro, através de eventos de tique de relógio

Definições e propriedades (3)

Tempo de ciclo secundário



(a) Ciclos principais



cada pedaço de código é uma chamada de questão que é chamada no executivo cíclico

(b) Quadros

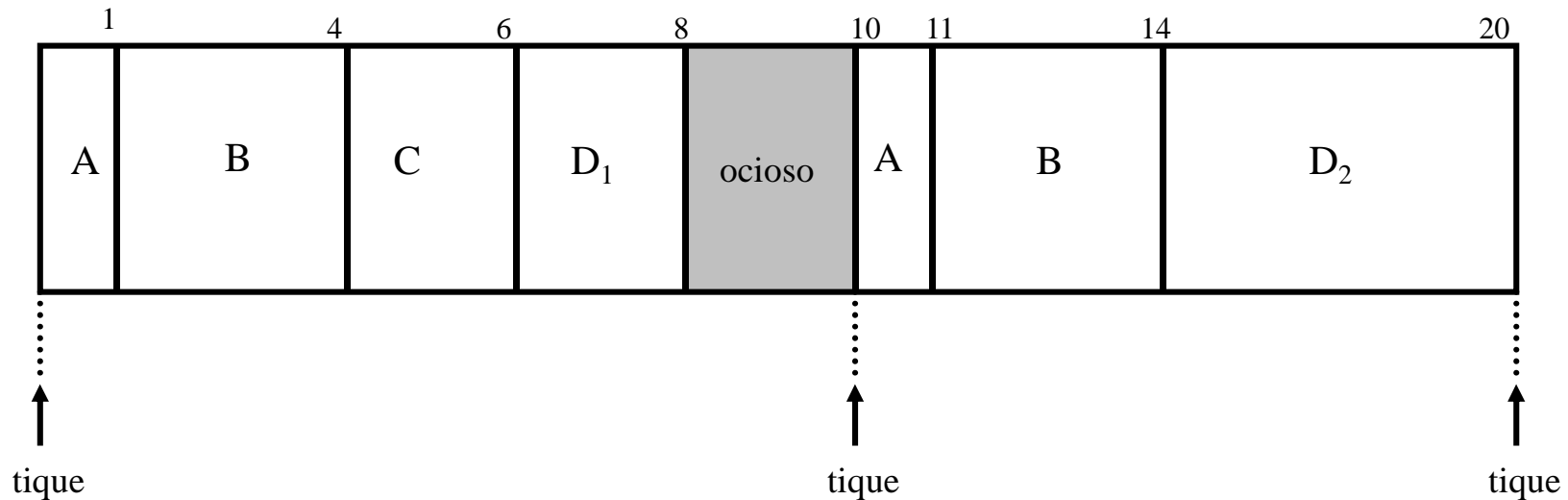
Exemplo

- 1) Considere 4 processos periódicos (c,d,p) :
 - $A=(1,10,10)$; $B=(3,10,10)$; $C=(2,20,20)$; e $D=(8,20,20)$
 - A, B e C são fatias únicas
 - D é dividido em 2 fatias D1 ($c=2$) e D2 ($c=6$)
 - TCP = 20 (o escalonamento inteiro pode ser repetido a cada 20 unidades de tempo)
 - Vamos assumir que tcs vale 10
- a) Escreva uma escala (diagrama temporal ou de *Gantt*) para este conjunto de processos
- b) Escreva um programa executor cíclico (ou despachante) para essa escala

Diagrama temporal do exemplo

quanto mais tempo ocioso + simples encontrar a escala

quanto mais divisões(Blocos) tiver maior o espaço de busca de soluções, mais soluções possíveis porém mais difícil(+ trabalhoso) encontrar a solução



```
ti=now()i
A()i
tf=now()i
D=Ca-()tf-ti
sleep (D)
```

Código do Executor Cíclico do Exemplo

```
tcs: constant := 10; --tempo de ciclo secundário
prox_momento : time := Clock + tcs; --próximo momento
de tique, clock retorna hora atual
Num_Quadro: integer := 1;
loop delay_until prox_momento; --bloqueio da tarefa
  Num_Quadro := (Num_Quadro++) mod 2;
  case Num_Quadro is
    when 0 => A; B; C; D1;
    when 1 => A; B; D2;
  end case;
  prox_momento := prox_momento + tcs;
  if Clock > prox_momento
    then Trata_Quadro_excedido;
  end if;
end loop;
```


Considerações sobre TCP e tcs

- $TCP = mmc(p_1, \dots, p_n)$ – é o menor tempo repetível que as tarefas podem executar ao menos uma vez mmc do período
- TCP é múltiplo de tcs
- tcs maior ou igual que o maior tempo de computação do maior bloco
- O tcs tem que ser menor ou igual que o menor *deadline*
- Relacionamento entre tcs , período e *deadline*:

$$tcs + (tcs - MDC(tcs, p_i)) \leq d_i.$$

Derivação de TCP e tcs

- $A=(1,14,14)$; $B=(2,20,20)$; e $C=(3,22,22)$
- Qual seria o valor de TCP e tcs?
- $TCP = MMC(14, 20, 22) = 1540$
- $tcs \leq 14$ e $tcs \geq 3$
- Candidatos de 3 até 14
- Quais os divisíveis por 1540?
 - 4, 5, 7, 10, 11 e 14

Derivação de TCP e tcs

- Aceitando somente os que satisfazem o relacionamento fica {4, 5, 7}
- Testando o 10 ($tcs + (tcs - MDC(tcs, p_i)) \leq d_i$)
 - $10 + (10 - MDC(10, 14)) \leq 14$
 - $10 + (10 - 2) \leq 14$
 - $18 \leq 14$ ✗
- Testando o 4
 - $4 + (4 - MDC(4, 14)) \leq 14 \therefore 6 \leq 14$ ✓
 - $4 + (4 - MDC(4, 20)) \leq 20 \therefore 4 \leq 20$ ✓
 - $4 + (4 - MDC(4, 22)) \leq 22 \therefore 6 \leq 22$ ✓

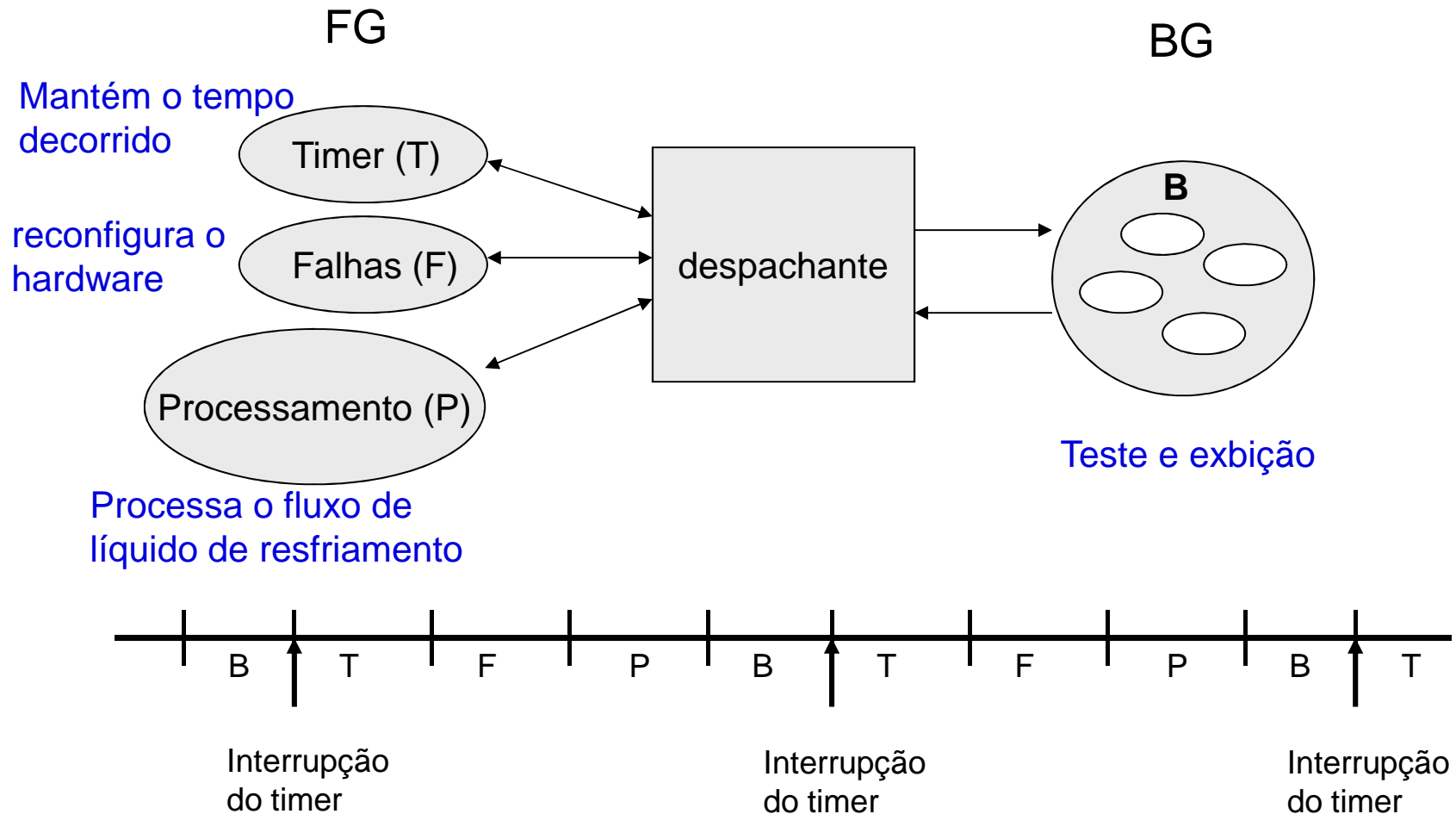
Organizações FG e BG



- Pode-se considerar mais simples que o EC
- Sistema consiste de 2 conjuntos de tarefas:
 - *Foreground* (FG): alta prioridade, críticos no tempo, não preemptíveis
 - *Background* (BG): baixa prioridade, processos que não são de tempo-real ou *soft* (brandos)
 - Processos em BG podem ser interrompidos por processos em FG
- Processos FG são executados de acordo com uma escala prévia
- Quando há tempo livre, processos em BG são despachados.
- Um *timer* interrompe o processo corrente (de background) sinalizando o início de um novo ciclo

Exemplo: monitor de usina nuclear

Sistema opera com redundância quádrupla e eleição sobre os resultados



p/ verificar se é possível ($V = C/p$)
 $2/9 + 4/12 + 1/15 = (40+60+12)/180$

Exercício

tempo máximo

- Um sistema tem três tarefas periódicas $A=(2, 9, 9)$, $B=(4, 12, 12)$ e $C=(1, 15, 15)$.

- Qual é o TCP para um escalonamento cíclico EC para este sistema? $\text{tcp} \rightarrow \text{tempo de ciclo principal} = \text{precisamos do mmc do período valores destacados em amarelo}$

Resp.: 180

- ~~Assumindo que os blocos sejam idênticos em relação ao código completo para todas as tarefas, derive os valores possíveis para o tcs~~
- Tome uma das respostas de (b) e construa um escalonamento principal para as três tarefas
- Substitua C por $C' = (5, 15, 15)$. Usando os valores tcs computados em (b), mostre que um escalonamento EC não pode ser construído

resp.C

A	B	C	A	B	C	A
2	4	1	2	4	1	2
7	9	11	12	16	12	