Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD0121 – Arquitetura de Computadores

Caminho de Dados
Monociclo x Multiciclo x Pipeline

Prof. Gustavo Girão

Plano de Aula

- Conhecer o fluxo de execução de três técnicas
  - Monociclo
  - Multiciclo
  - Pipeline
- Entender as implicações de cada técnica no desempenho
- Discutir as soluções para melhoria de desempenho

Organização do Processador

- Etapas de execução
  1. Busca da Instrução
  2. Decodificação da Instrução
  3. Execução
  4. Acesso à Memória
  5. Gravar resultado

- Principais componentes
  - ULA
  - Controle
  - Banco de registradores
  - Memória
  - PC

Sobre as etapas de execução

- Todas as instruções passam pelas cinco etapas

```
add  $t0, $t1, $t2
lw   $t0, 64($t1)
j    LACO
beq  $t0, $t1, SQRT
```

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- OPERAÇÕES LÓGICAS E ARITMÉTICAS

add $t0, $t1, $t2

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

O que acontece nessa etapa?

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

O que acontece nessa etapa?

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

E nesssa etapa?

Sobre as etapas de execução

- ACESSO À MEMÓRIA

lw $t0, 64($t1)

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- **Todas as instruções passam pelas cinco etapas**

```
add  $t0, $t1, $t2
lw   $t0, 64($t1)
j    LACO
beq  $t0, $t1, SQRT
```

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

Sobre as etapas de execução

- **Todas as instruções passam pelas cinco etapas**

```
add  $t0, $t1, $t2
lw   $t0, 64($t1)
j    LACO
beq  $t0, $t1, SQRT
```

1. Busca da Instrução
2. Decodific. da Instrução
3. Execução
4. Acesso à Memória
5. Gravar resultado

CAMINHO DE DADOS MONOCICLO

Monociclo

Cada instrução leva um ciclo para executar

Monociclo

Cada instrução leva um ciclo para executar

Independente de quais operações a instrução precisa realizar

Independente de qual é a instrução

Cada instrução leva um ciclo para executar

Ciclo da Instrução no Monociclo

- **Monociclo**
  - Toda instrução começa sua execução em uma transição ativa do sinal do *clock* e completa a execução na próxima transição ativa
  - Mesmo tempo para todas as instruções

COMO DEFINIR O TEMPO DA INSTRUÇÃO?

Ciclo da Instrução no Monociclo

- Exemplo:
  - Operações lógicas/aritméticas: 0,5ns
  - Acesso à memória: 2ns
  - Controle: 1ns

- Qual o tempo da instrução do monociclo, considerando os atrasos acima?

```
add     $t1, $t2, $t3
lw      $s1, 64($s2)
j       LACO
beq     $s0, $s4, FIM
```

Ciclo da Instrução no Monociclo

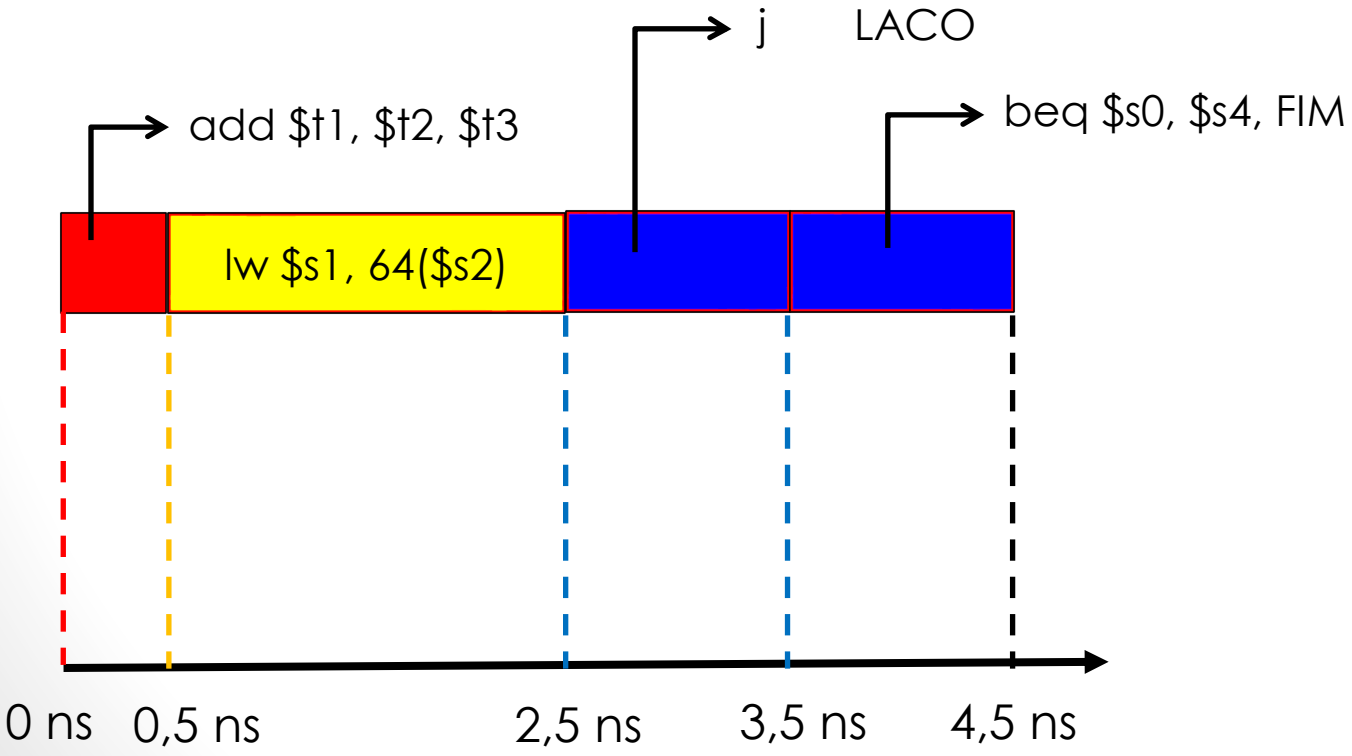
- Exemplo:
  - Operações lógicas/aritméticas: 0,5ns
  - Acesso à memória: 2ns
  - Controle: 1ns

- Qual o tempo da instrução do monociclo, considerando os atrasos acima?


```
add     $t1, $t2, $t3
lw      $s1, 64($s2)
j       LACO
beq     $s0, $s4, FIM
```

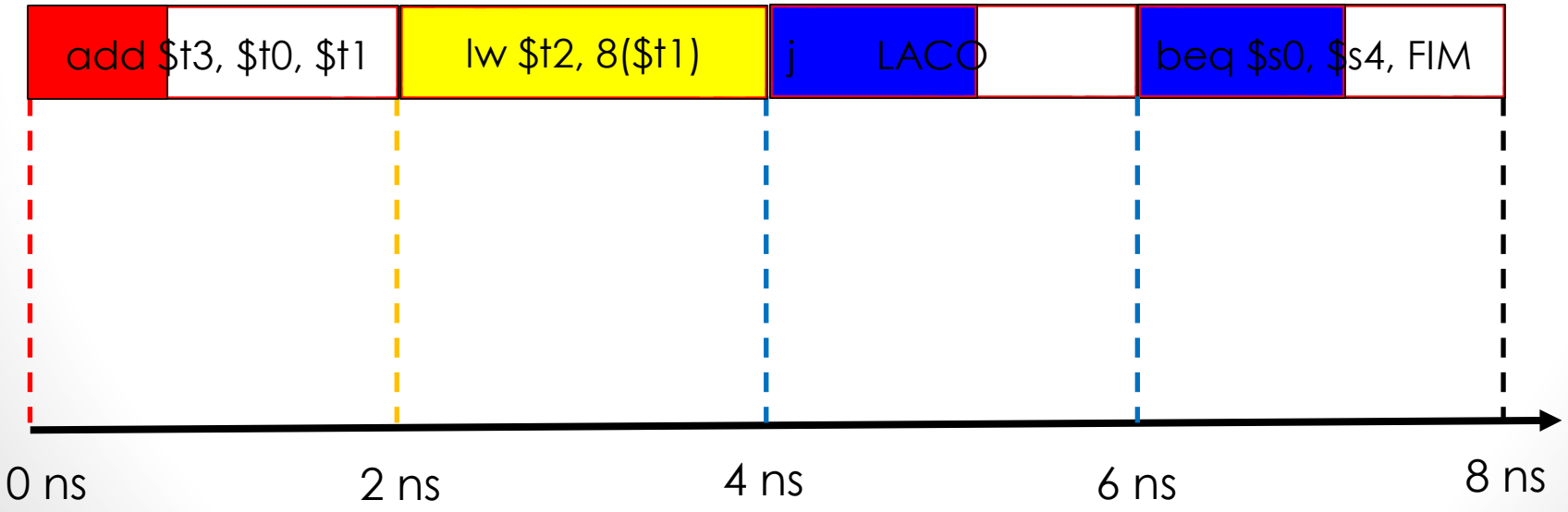
Ciclo da Instrução no Monociclo

- Como seria?
- As instruções, efetivamente levariam tempos diferentes:



Ciclo da Instrução no Monociclo

- Porém, No monociclo **toda** instrução precisa executar em um ciclo.
- Logo, o tamanho deste ciclo precisa de tal tamanho que possa comportar as necessidades de **todas** as instruções



QUAIS AS DESVANTAGENS DO MONOCICLO?

Implementação do Ciclo Único

Se houver uma instrução mais complicada, como ponto flutuante???

Implementação do Ciclo Único

Se houver uma instrução mais complicada, como ponto flutuante???

Não adianta melhorar os casos mais comuns!!!

Implementação do Ciclo Único

O que acontece com os componentes que não estão sendo usados no momento?

Implementação do Ciclo Único

O que acontece com os componentes que não estão sendo usados no momento?

Muitos componentes ociosos

QUAIS AS VANTAGENS DO MONOCICLO?

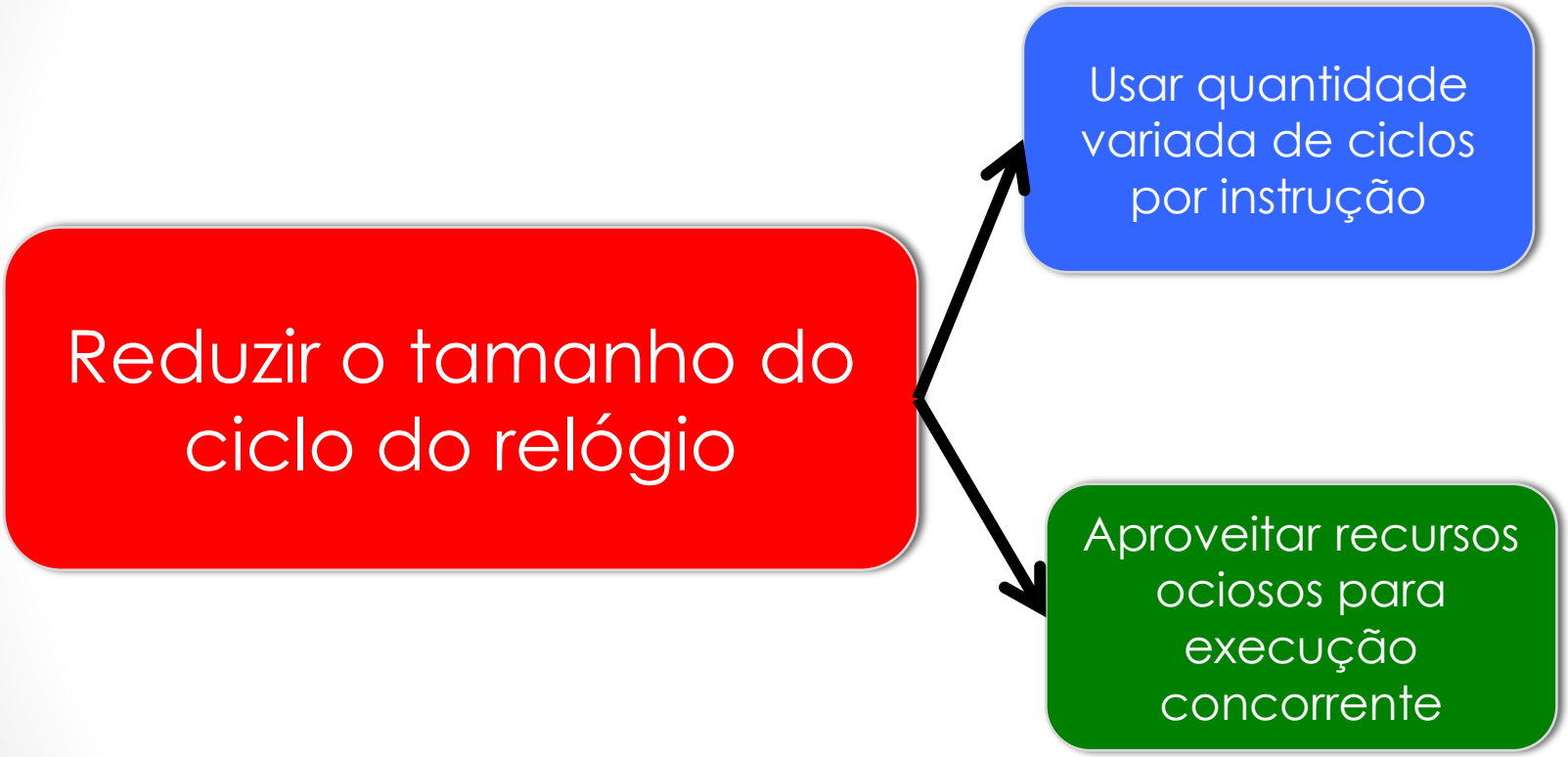
Implementação do Ciclo Único

Simplicidade

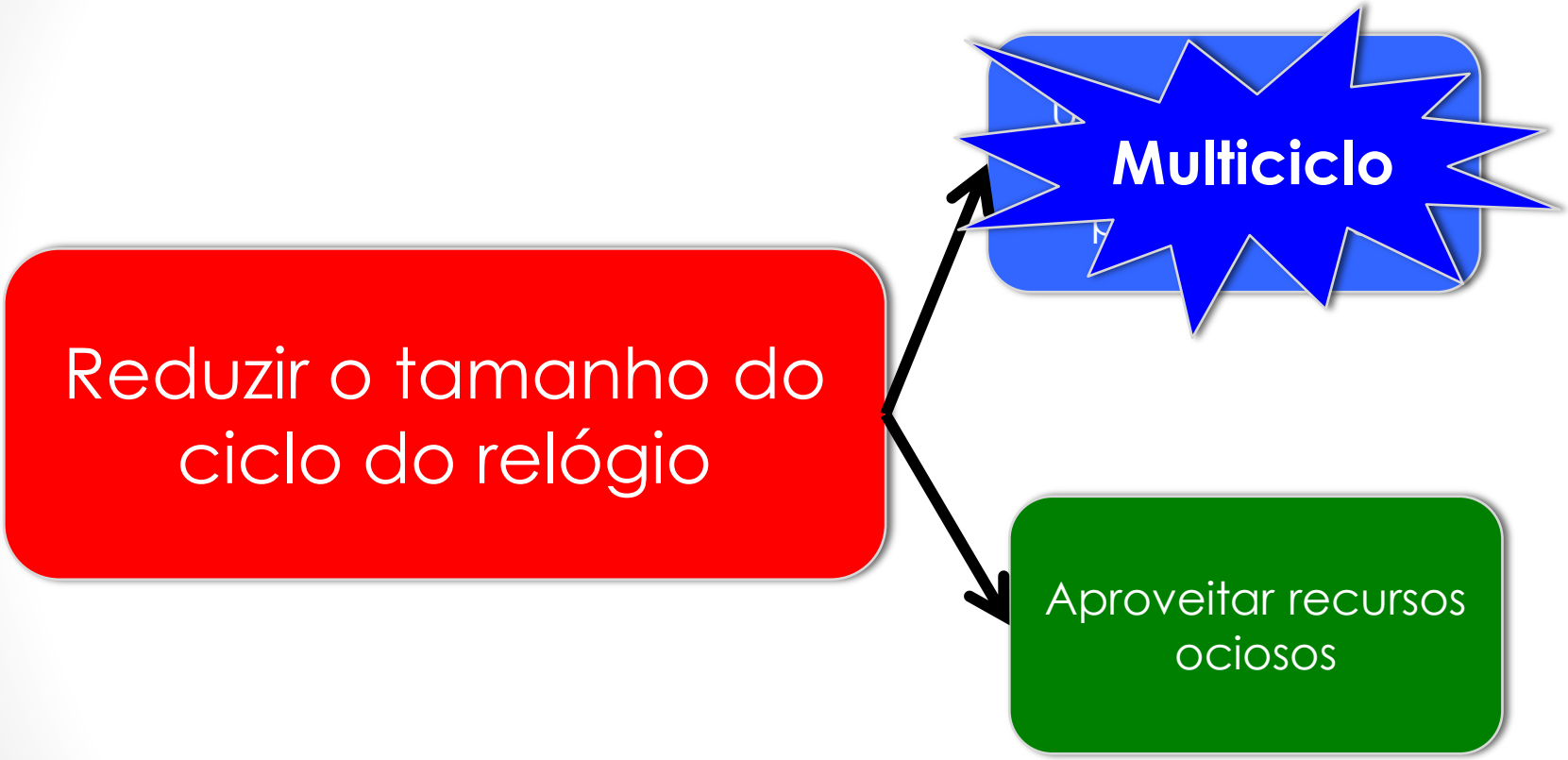
Implementação do Ciclo Único

Previsibilidade

Soluções para melhorar desempenho



Soluções para melhorar desempenho



Soluções para melhorar desempenho

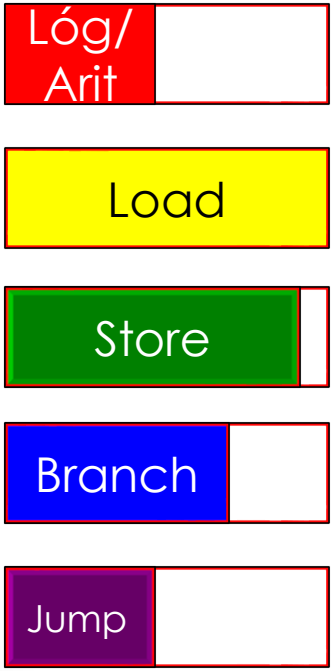
- Reduzir o tamanho do ciclo do relógio
  - Cada ciclo passa a ser um estágio de execução da instrução
  1. Busca da Instrução
  2. Decodificação da Instrução
  3. Execução
  4. Acesso à Memória
  5. Gravar resultado
  - Usa o tempo do estágio mais lento para definir o tamanho do ciclo

Solução 1 - Multiciclo

- Vários ciclos por instrução
- Cada instrução pode ser executada num número diferente de ciclos.

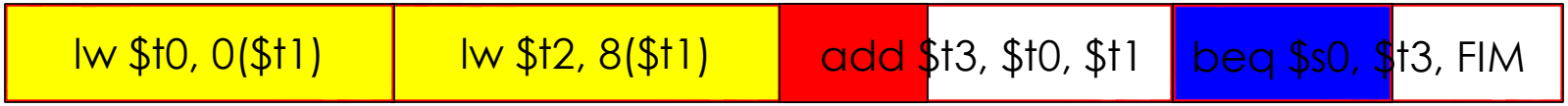
Solução 1 - Multiciclo

Monociclo

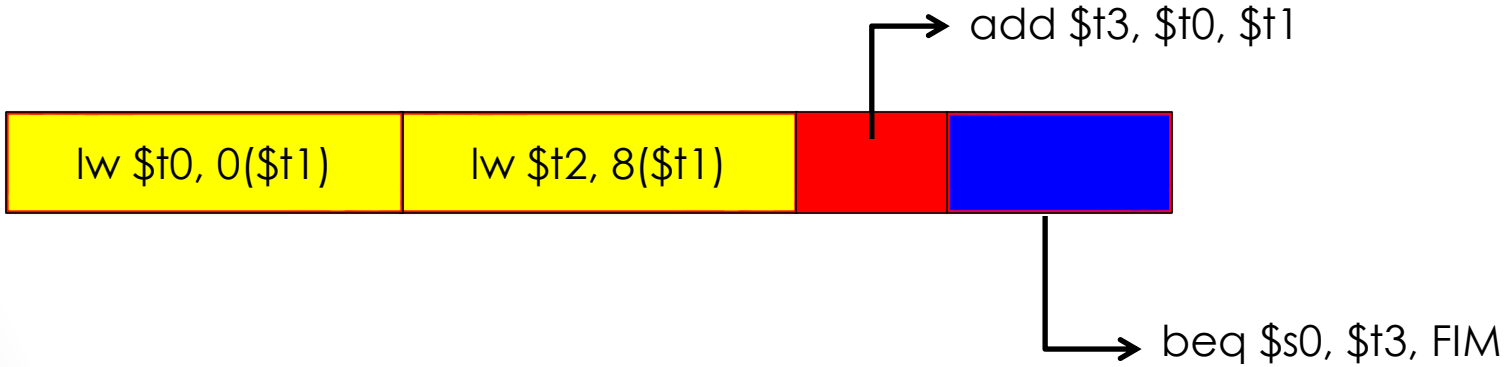


Solução 1 - Multiciclo

Monociclo

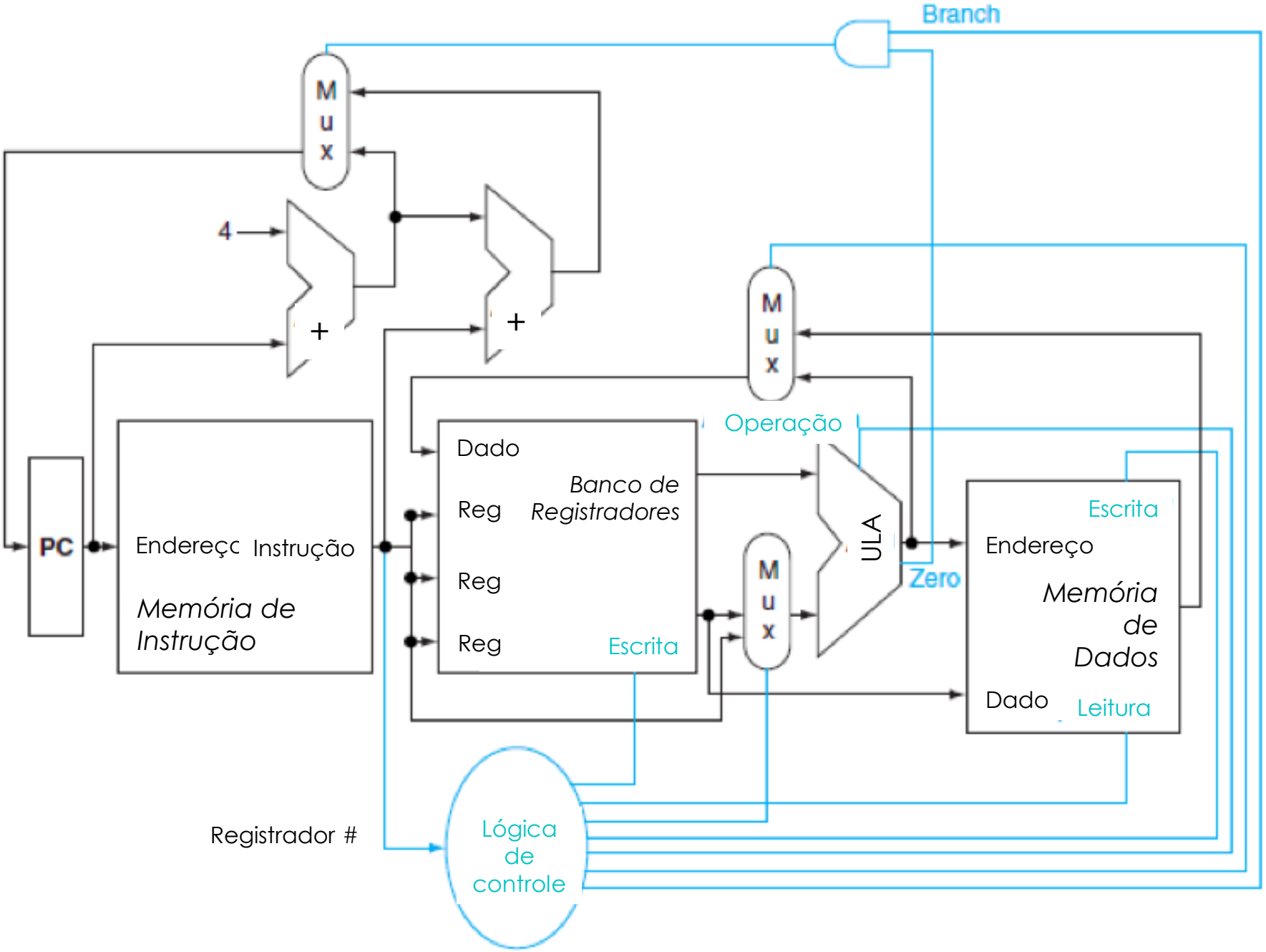


Multiciclo

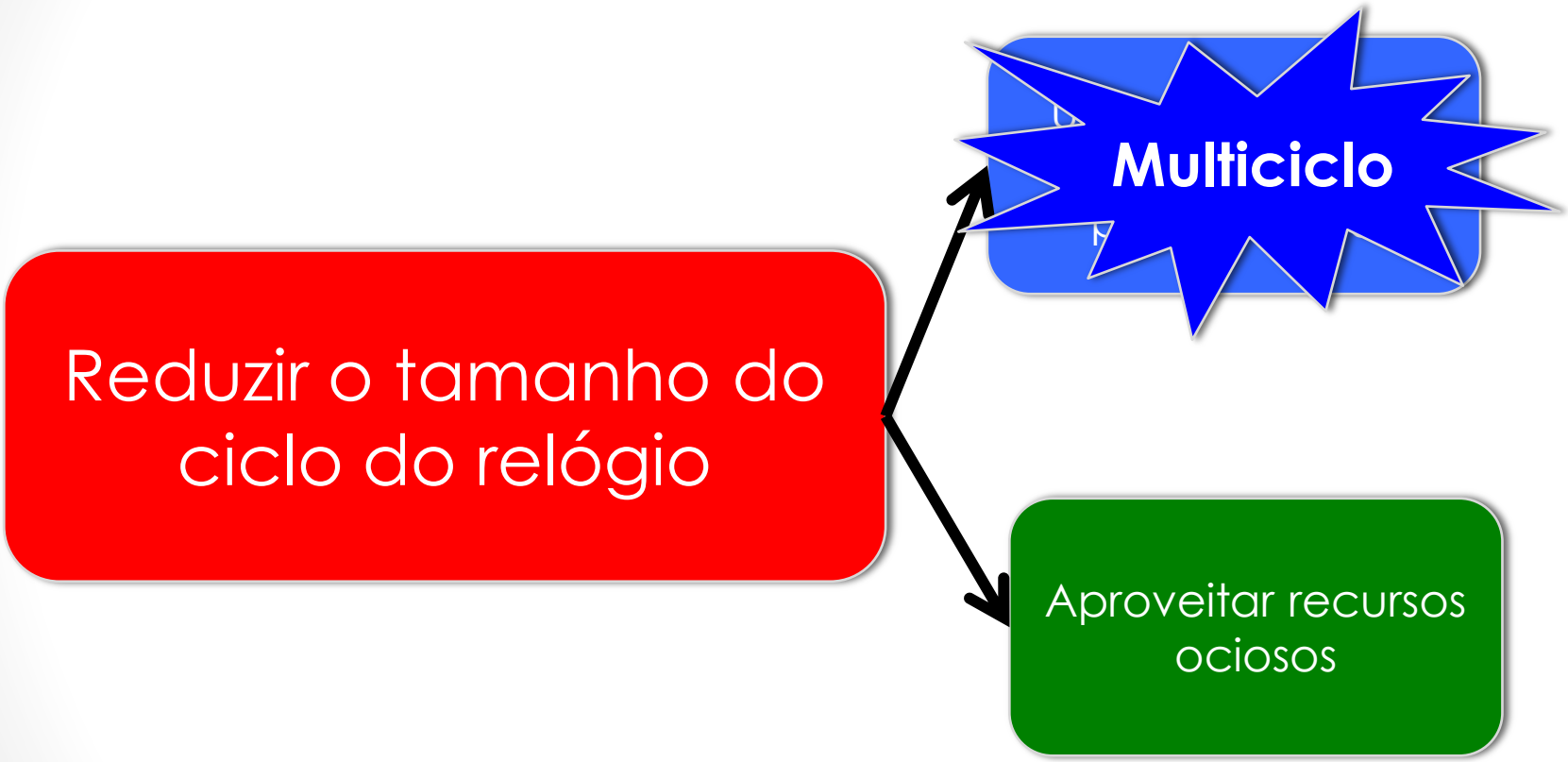


DESVANTAGEM DO MULTICICLO????

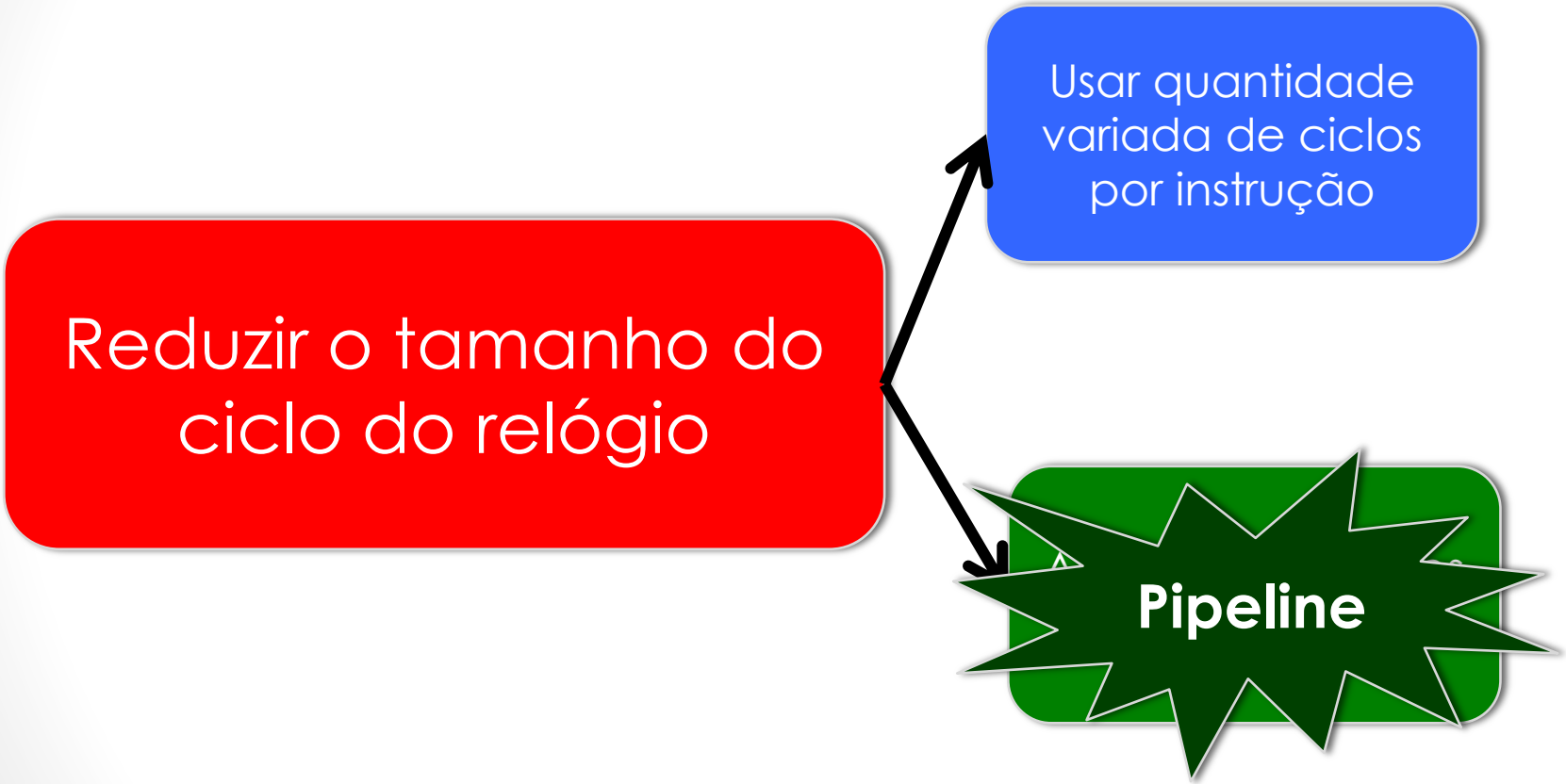
MUITOS COMPONENTES OCIOSOS EM CADA ESTÁGIO


Branch
M
u
x
4
+
+
M
u
x
Operação
PC
Endereço
Instrução
Memória de Instrução
Dado
Reg
Reg
Reg
Banco de Registradores
Escrita
M
u
x
ULA
Zero
Escrita
Endereço
Memória de Dados
Dado
Leitura
Registrador #
Lógica de controle

Soluções para melhorar desempenho



Soluções para melhorar desempenho

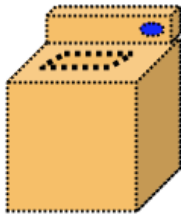


Solução 2 - Pipeline

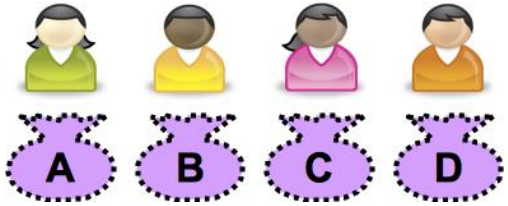
- Aproveitar componentes ociosos para execução concorrente de diversas instruções

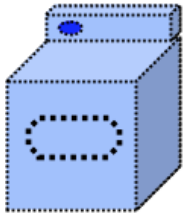
Analogia

- Exemplo de uma lavanderia (Hennessy & Patterson)
  - 4 pessoas (A, B, C, D) têm sacolas de roupa para lavar, secar e dobrar.



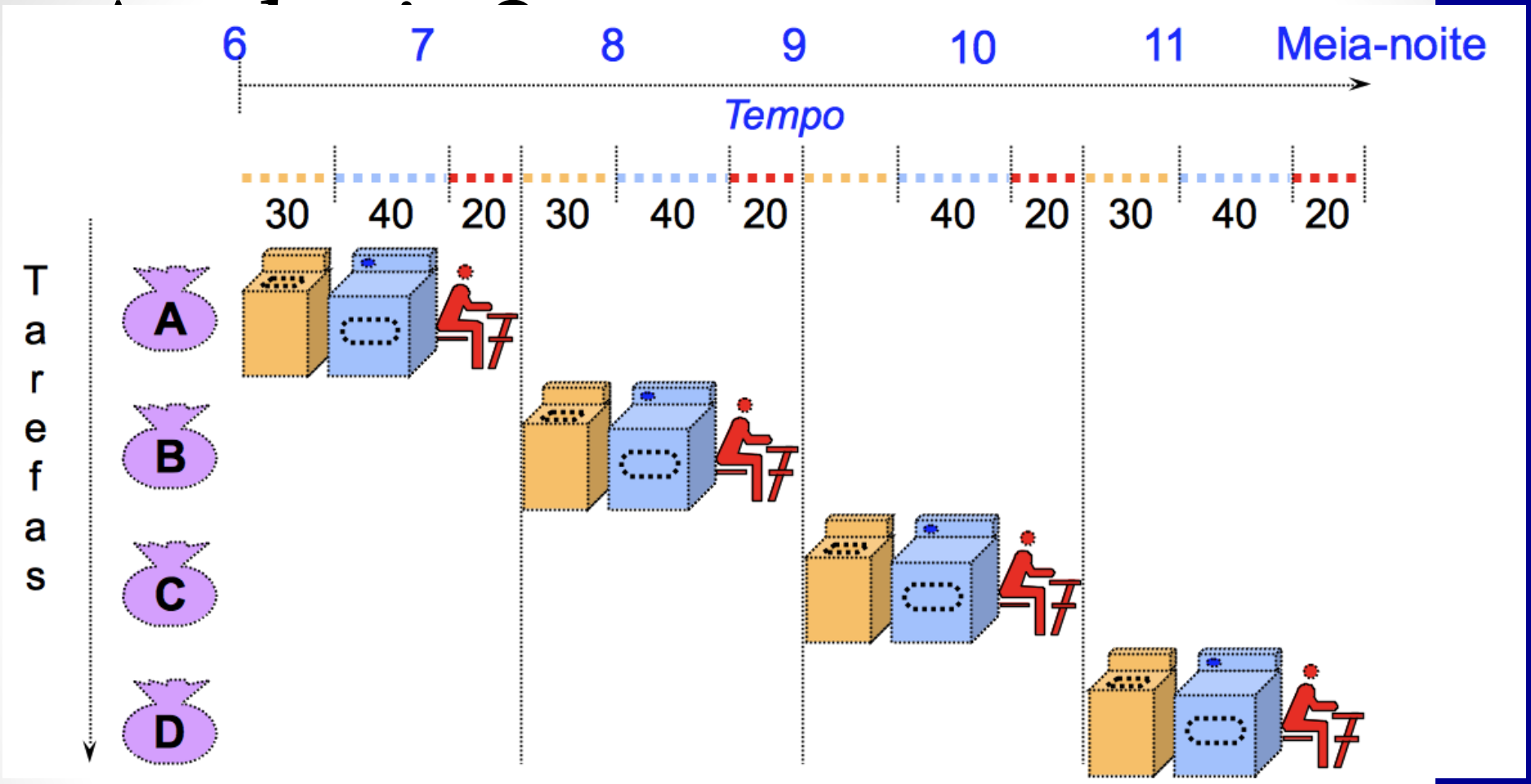
Lavar leva 30 minutos





Secar leva 40 minutos

Dobrar leva 20 minutos

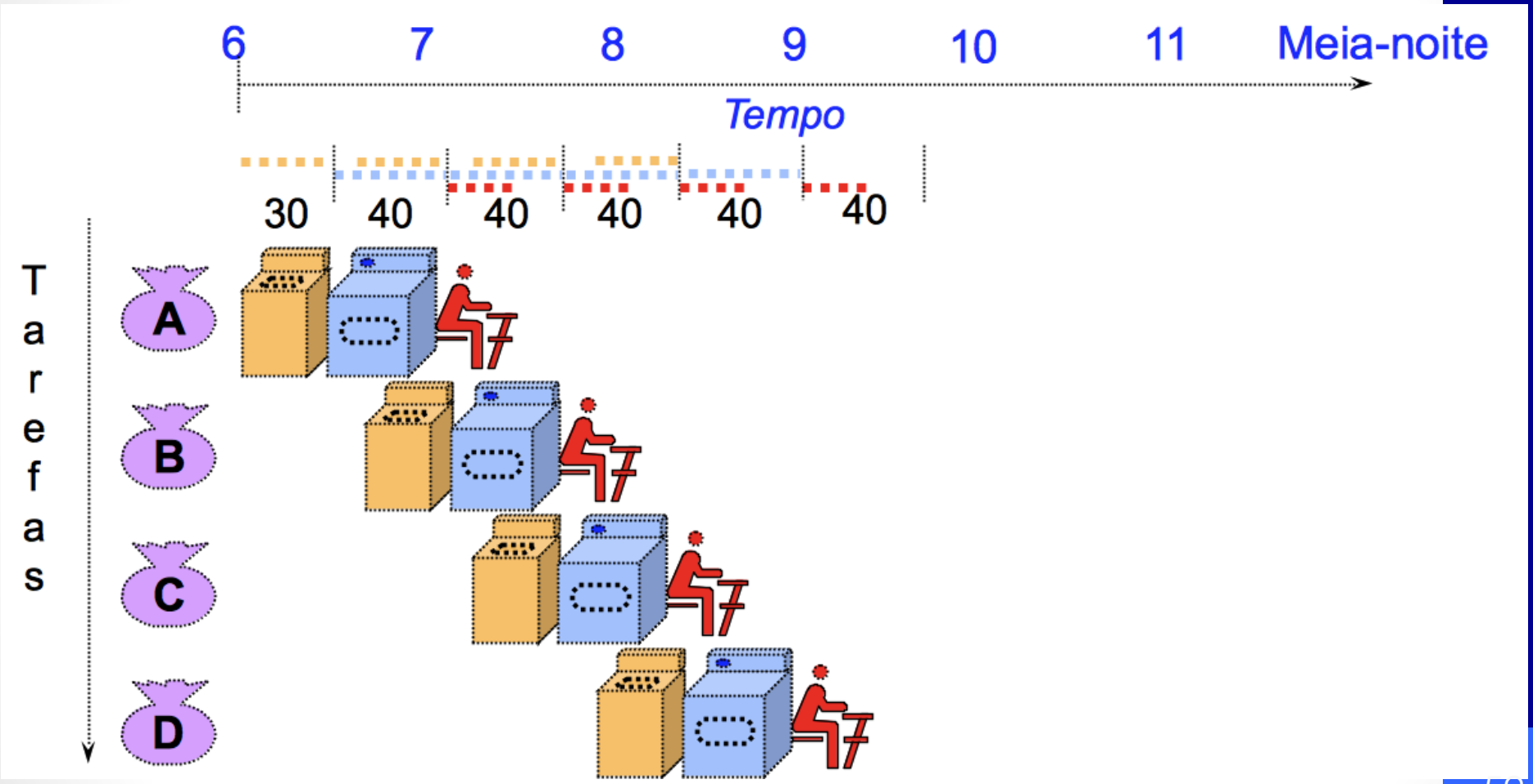


- Lavanderia sequencial leva **6 horas** para terminar

Analogia

Considerando o uso de equipamentos ociosos para adiantar as tarefas

Analogia



- Lavanderia com pipeline leva **3 horas e 50 minutos**

Pipeline

- Latência: tempo que UMA sacola de roupa leva para ser lavada
  - No exemplo da lavanderia, a latência é de ~120 minutos (110 minutos se considerarmos a primeira lavagem)

- Taxa de vazão (*throughput*): uma sacola por unidade de tempo
  - **Somente quando o pipeline está cheio**

Pipeline

- Objetivo
  - Aumento de desempenho
- Como?
  - N tarefas executadas concorrentemente, uma em cada estágio

Definição

- Pipeline é uma técnica de implementação de processadores que permite a **sobreposição temporal** de diversas fases de execução de instruções.

- Ou seja, o hardware **processa mais de uma instrução** de cada vez, sem esperar que uma instrução termine para começar outra.

Características

- Pipeline não melhora a latência de uma única tarefa, mas melhora a vazão (*throughput*) de todo o trabalho;
- Tempo de execução de uma tarefa é o mesmo, com ou sem pipeline;
- Ganho começa a existir a partir da segunda tarefa;
- Taxa de inserção de tarefas é limitada pela tarefa mais lenta;

Características

- Pipeline explora paralelismo entre as instruções em um fluxo de instruções sequenciais;

- É uma técnica invisível ao programador, ao contrário de técnicas de multiprocessadores.

Exemplo – Cinco estágios de pipeline

PIPELINE CLÁSSICO

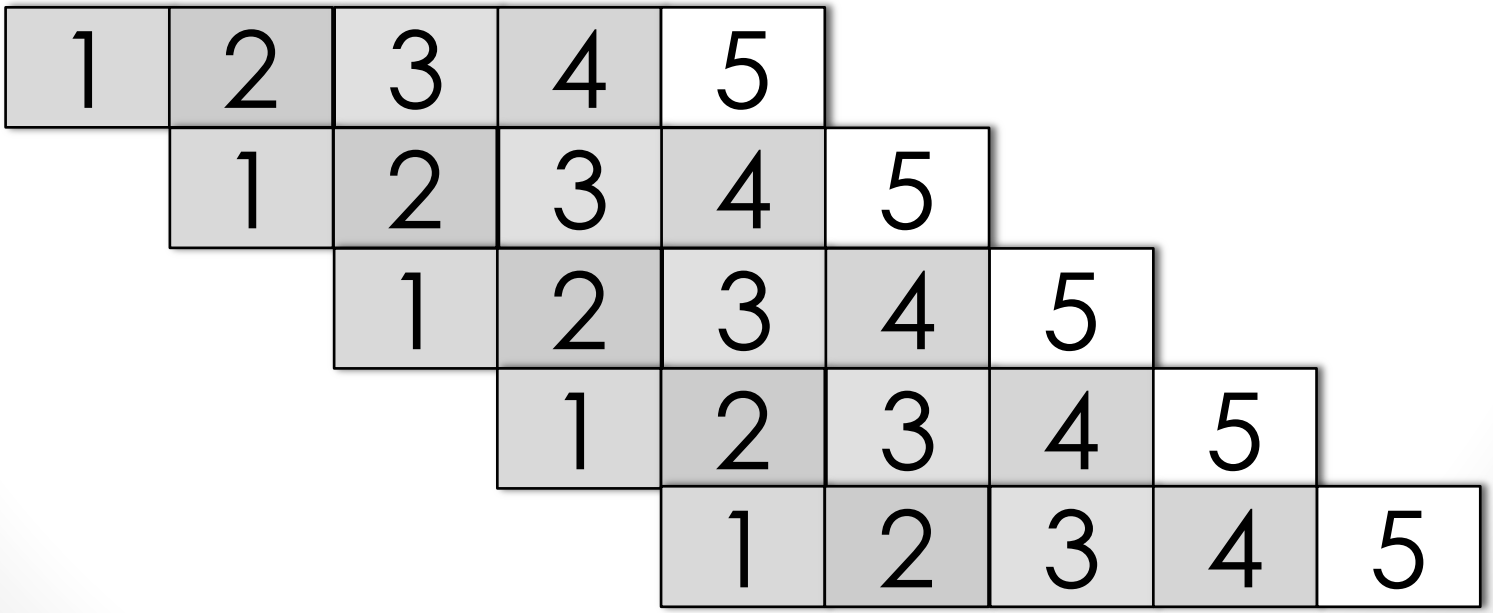
[1] Busca da instrução

[2] Decodificação da instrução

[3] Execução

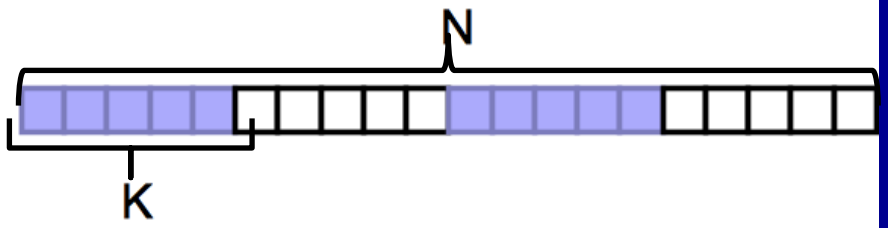
[4] Acesso à memória

[5] Armazenamento do resultado



Desempenho

- Existe um tempo inicial até que o pipeline "encha"
- Cada tarefa leva o mesmo tempo, com ou sem pipeline
- Média de tempo por tarefa é, no entanto, dividida por N tarefas
- Supondo
  - N instruções
  - K estágios de duração T



- Tempo de execução SEM pipeline

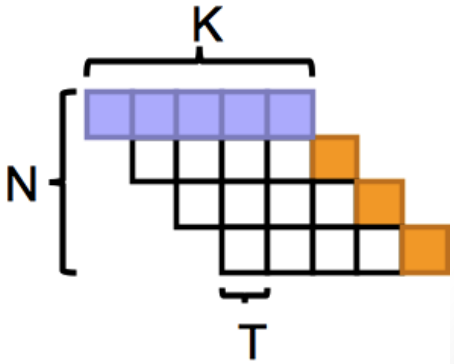
  N x K x T

- Tempo de execução COM pipeline

  [K + (N-1)] x T

  instr1 = K ciclos

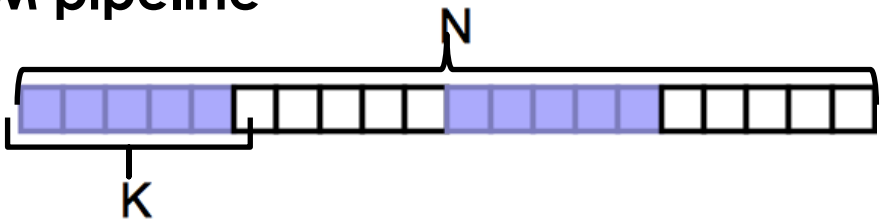
  demais instrs. = n-1 ciclos



Desempenho

- Supondo
  - N instruções
  - K estágios de duração T
- **Tempo de execução SEM pipeline**

  N x K x T

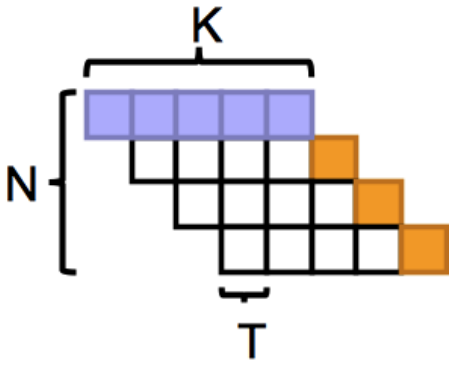


- **Tempo de execução COM pipeline**

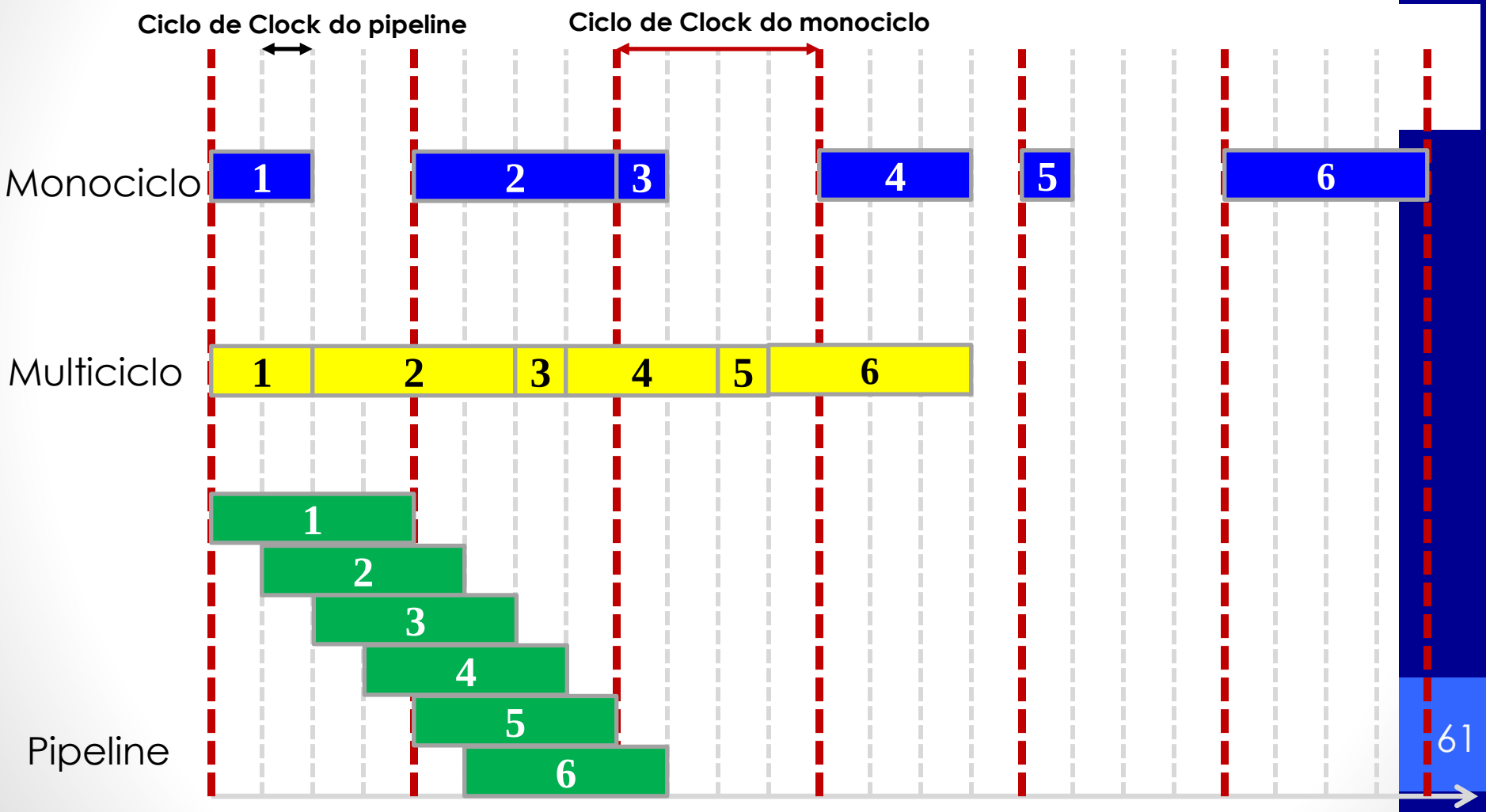
  [K + (N-1)] x T

  instr1 = K ciclos

  demais instrs. = n-1 ciclos



COMPARAÇÃO


Ciclo de Clock do pipeline
Ciclo de Clock do monociclo
Monociclo
1
2
3
4
5
6
Multiciclo
1
2
3
4
5
6
1
2
3
4
5
6
Pipeline

Bibliografia

- Patterson e Hennessy

  Organização e projeto de computadores – A interface Hardware/Software

O Processador: Caminho de Dados e Controle

Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD0121 – Arquitetura de Computadores

Caminho de Dados
Monociclo x Multiciclo x Pipeline

Prof. Gustavo Girão