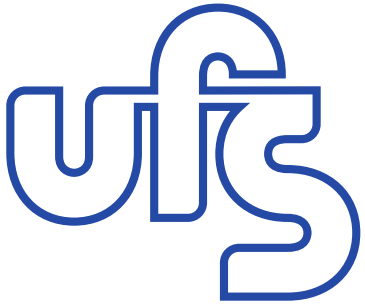
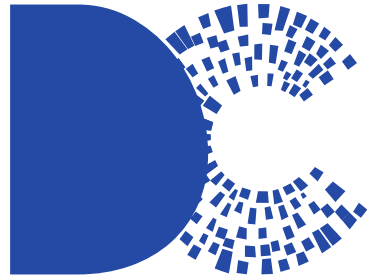


UNIVERSIDADE FEDERAL DE SERGIPE



DEPARTAMENTO DE COMPUTAÇÃO

Multiciclo e *pipeline*

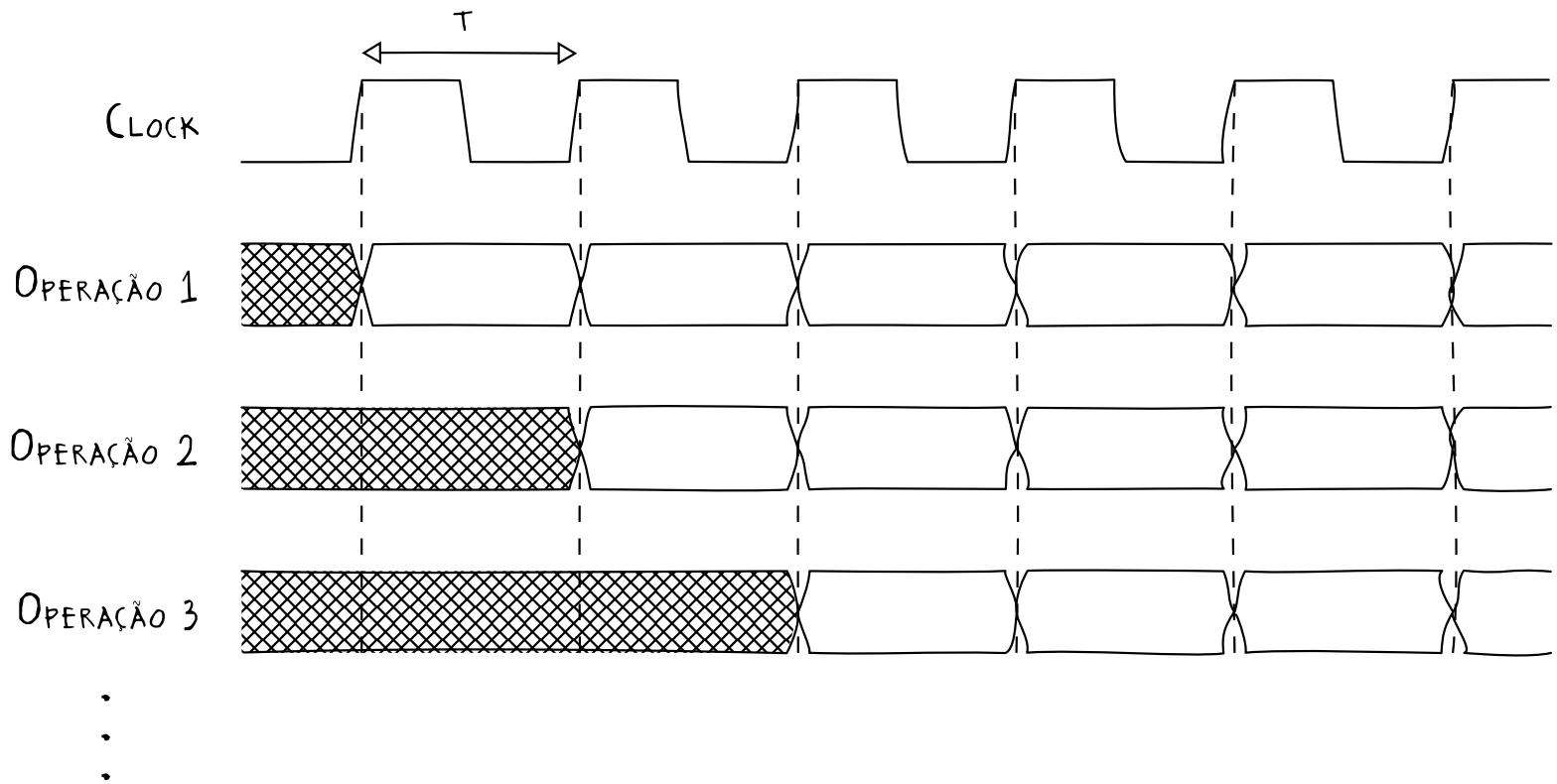
Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

Introdução

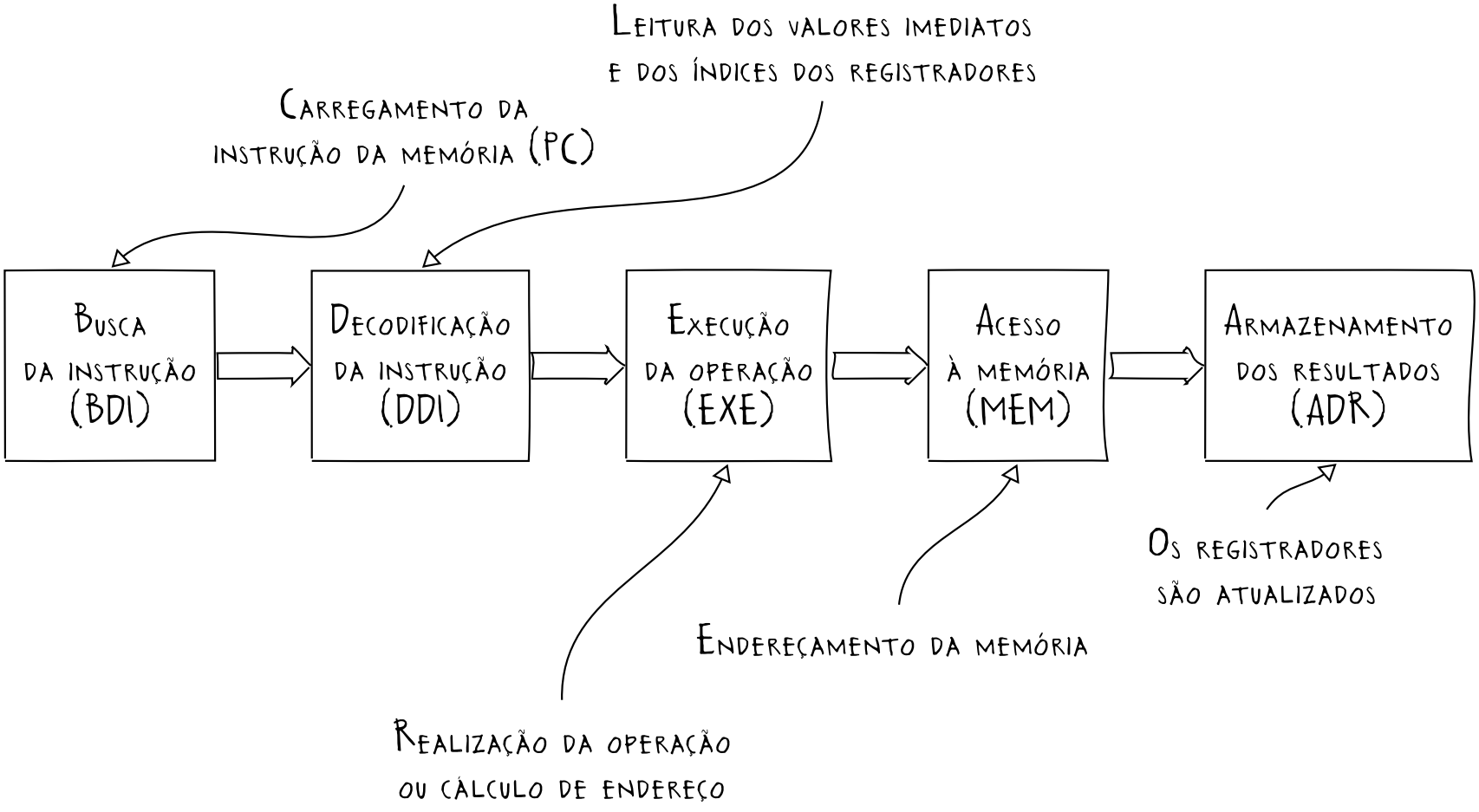
- A maioria dos sistemas computacionais é síncrono



$$f = 1 \div T$$

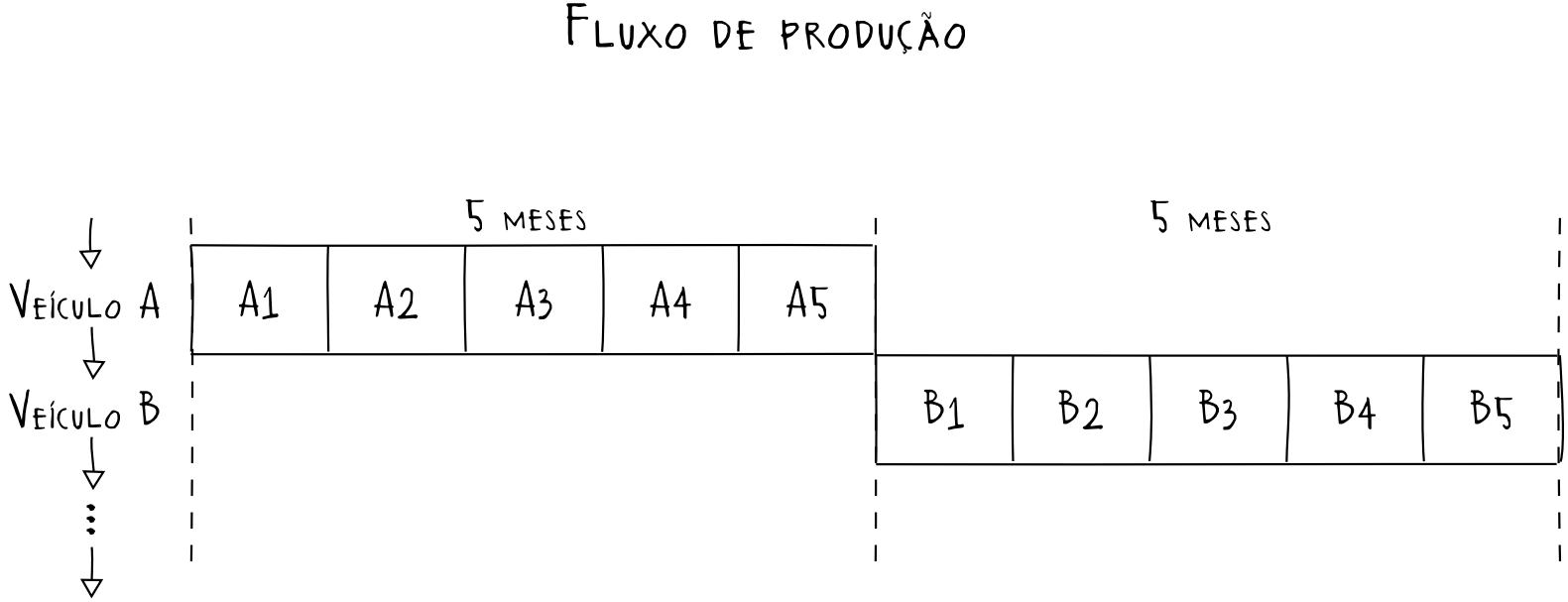
Introdução

- Etapas da execução de uma instrução



Introdução

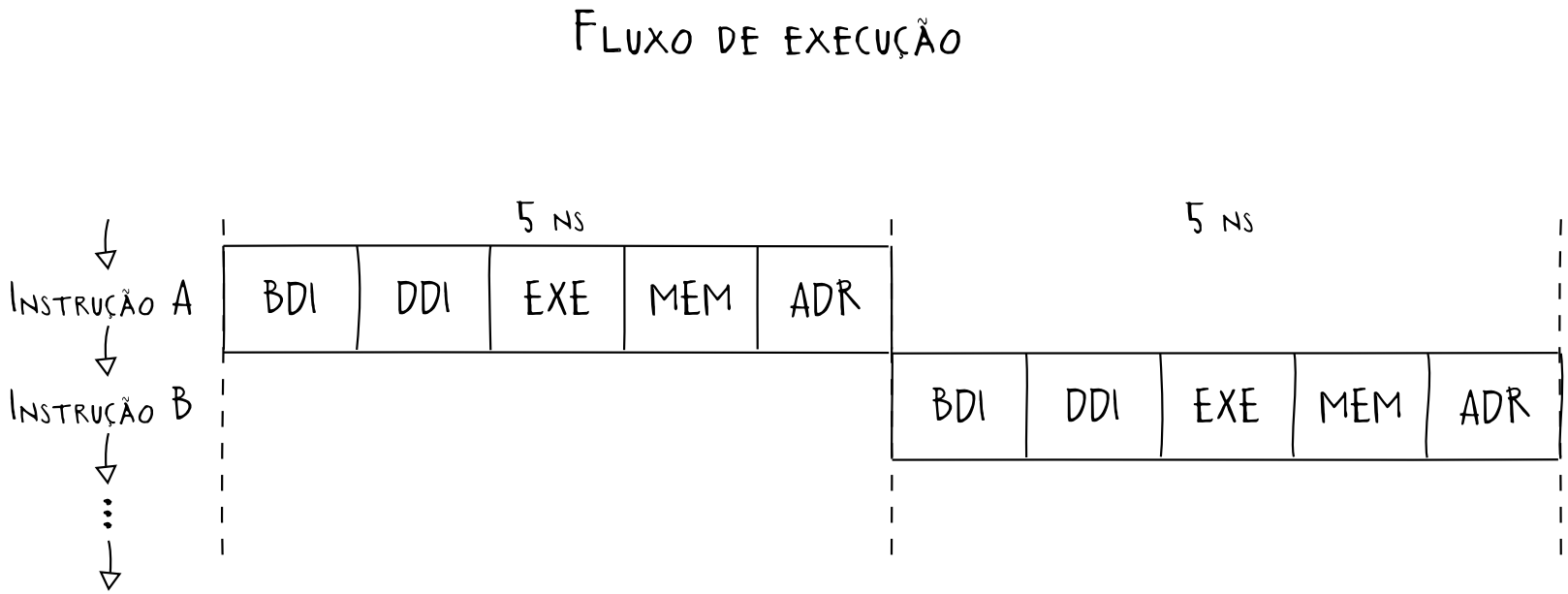
- Analogia com a produção de veículos em série



Todos os recursos da fábrica
são alocados durante
a produção de cada veículo

Introdução

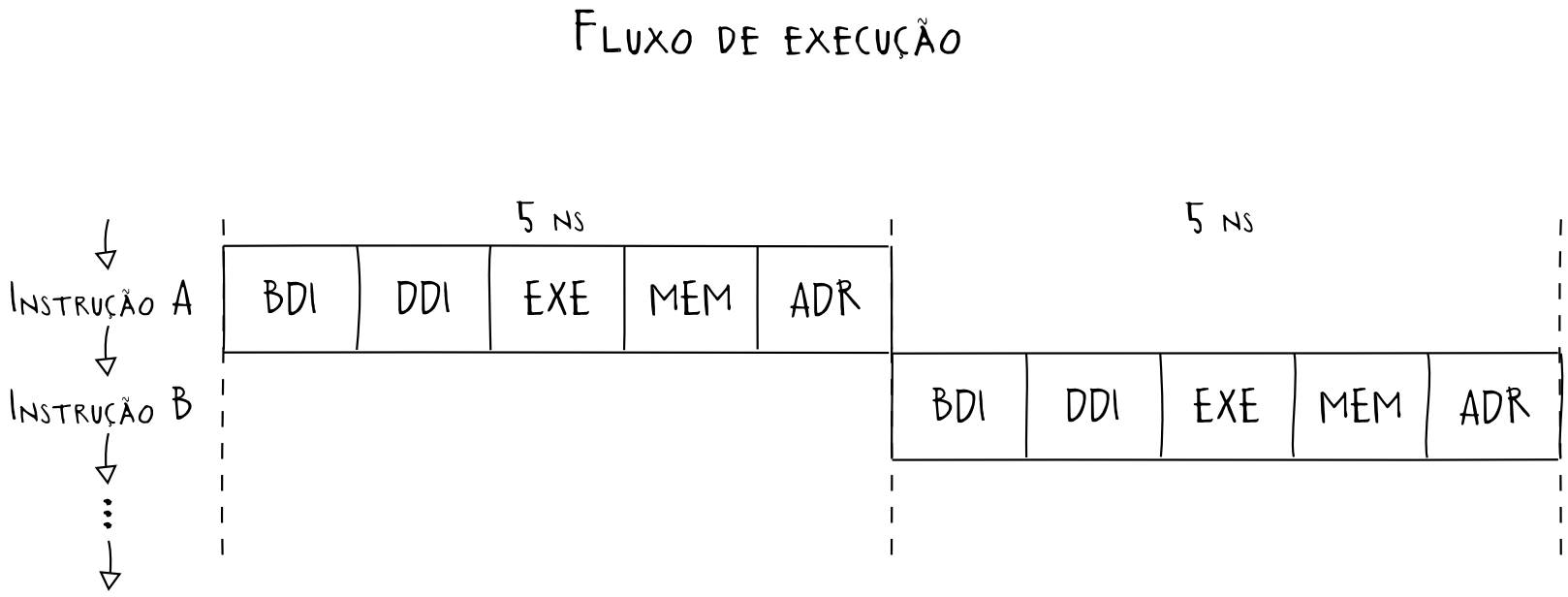
- Execução de uma instrução em ciclo único



A execução de cada instrução é sequencial
e ocupa todas as unidades de processamento

Introdução

- Execução de uma instrução em ciclo único



A execução de cada instrução é sequencial
e ocupa todas as unidades de processamento

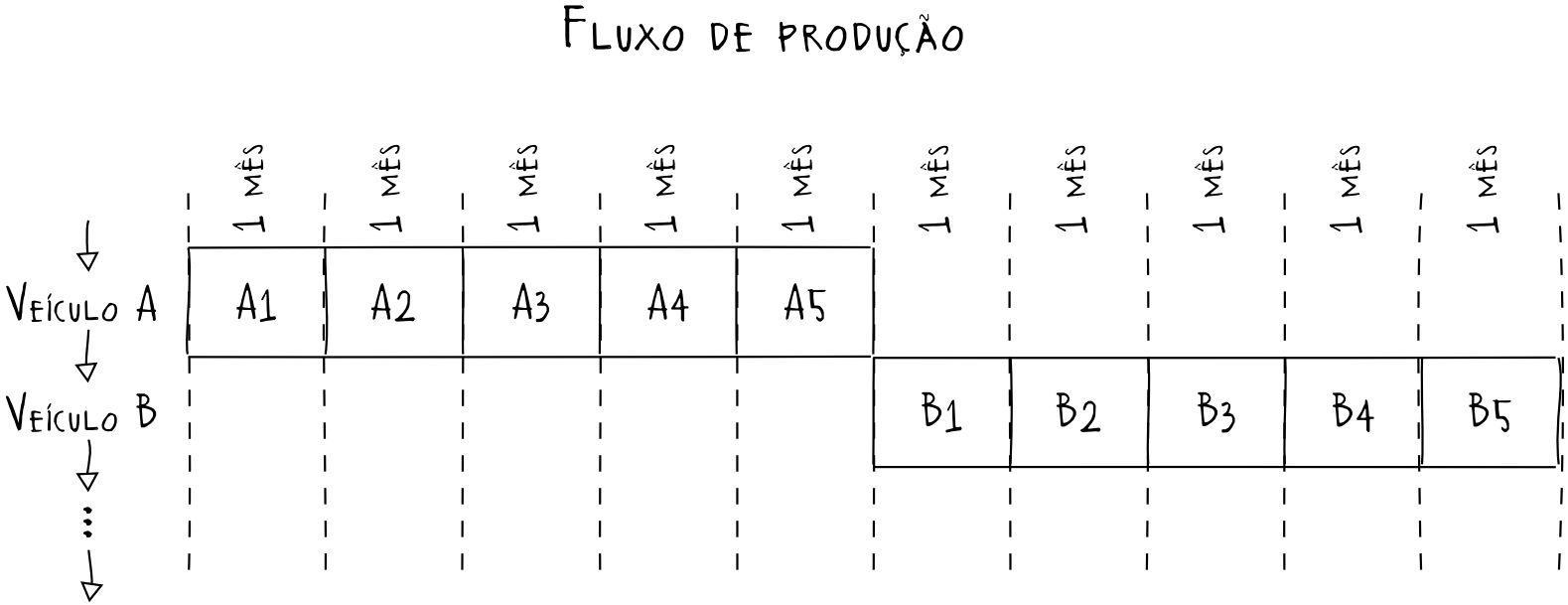
Somente uma etapa é executada por vez
(80% de ociosidade do hardware)

Introdução

- Projetos multiciclo e *pipeline*
  - As instruções são implementadas em múltiplos ciclos de relógio para reaproveitamento dos componentes de hardware entre as etapas de execução
    - **Multiciclo**: compartilhamento dos recursos de hardware em cada estágio (otimização do hardware)
    - ***Pipeline***: aumento do desempenho pela execução sobreposta das instruções (taxa de execução)

Multiciclo

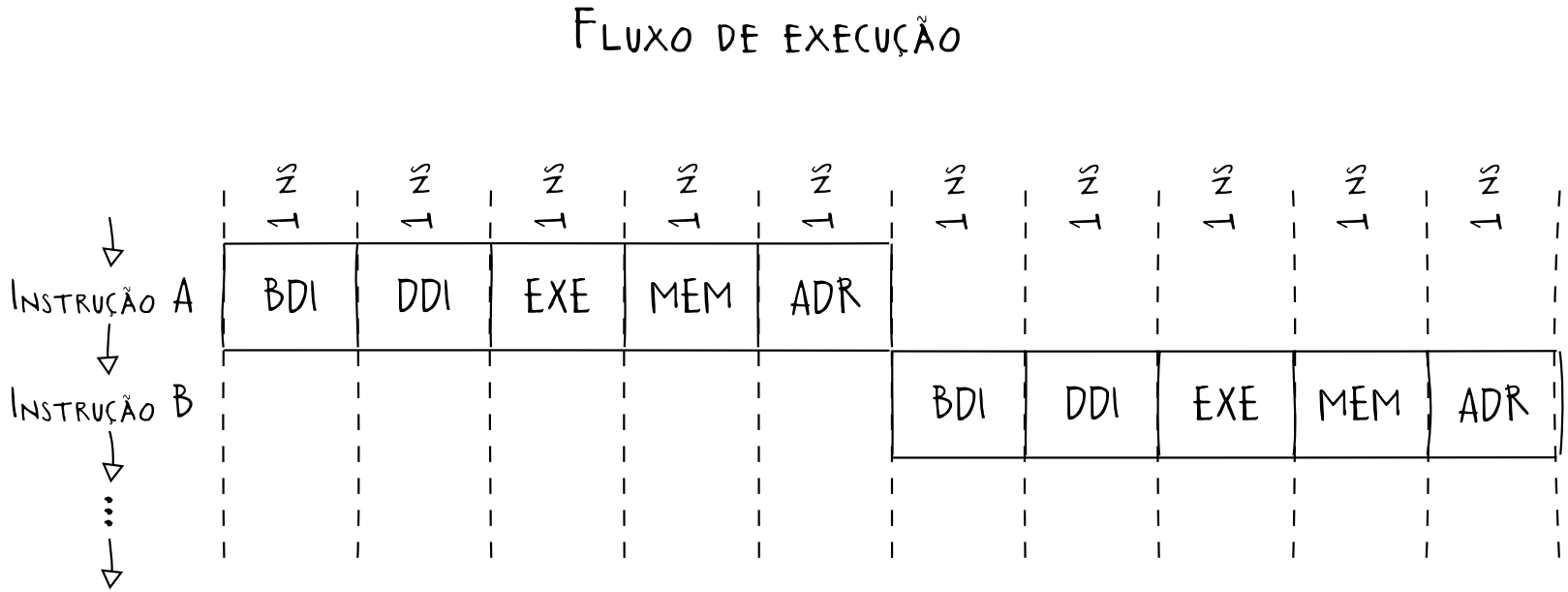
- Analogia com a produção de veículos em série



Cada etapa da produção possui um prazo
para ser realizada com acesso compartilhado
aos recursos da fábrica, como ferramentas e funcionários

Multiciclo

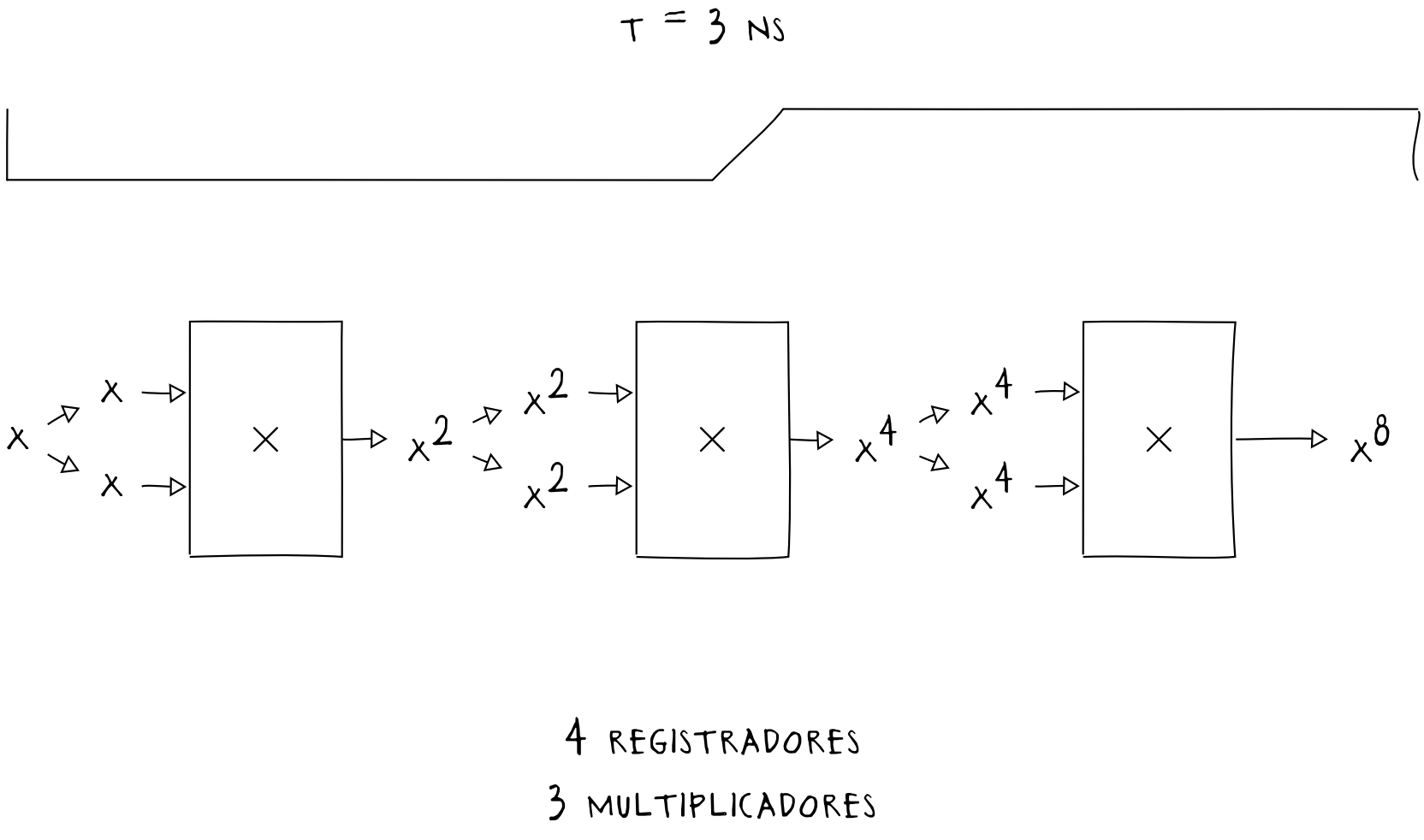
- Execução em multiciclo



Cada etapa de execução é executada em ciclos distintos, possibilitando o reaproveitamento dos recursos de hardware

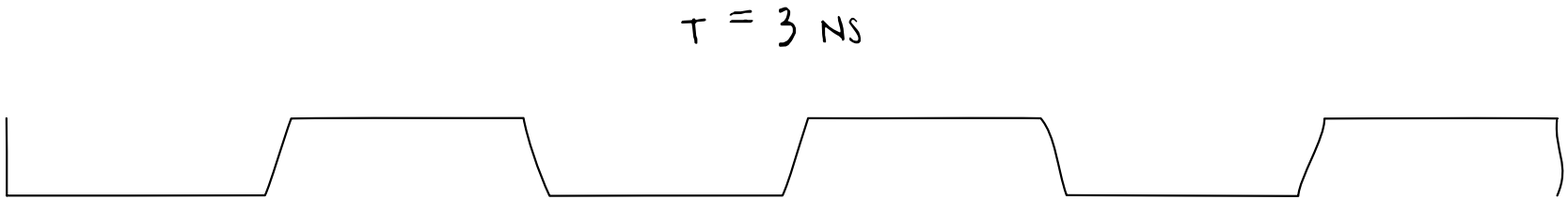
Multiciclo

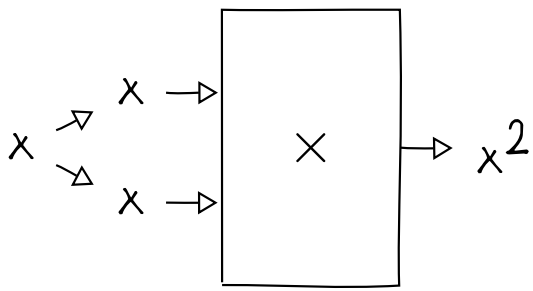
- Comparativo entre ciclo único e multiciclo
  - Instrução hipotética que calcula $f(x) = x^8$ (ciclo único)



Multiciclo

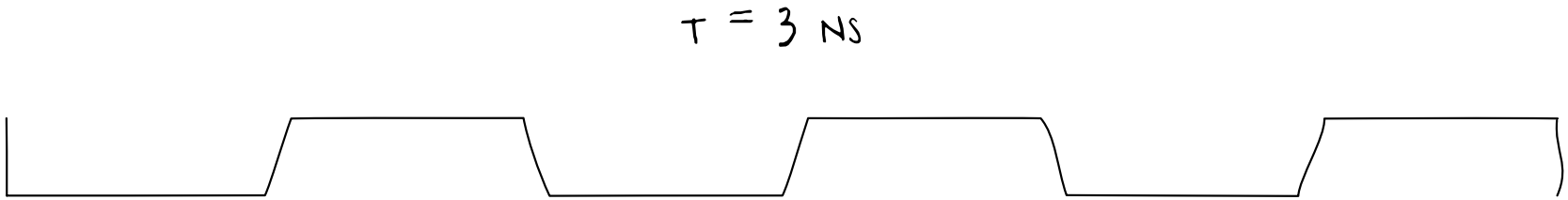
- Comparativo entre ciclo único e multiciclo
  - Instrução hipotética que calcula $f(x) = x^8$ (multiciclo)

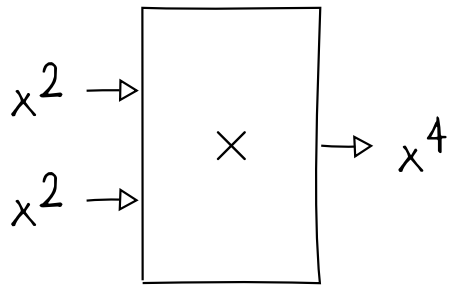




Multiciclo

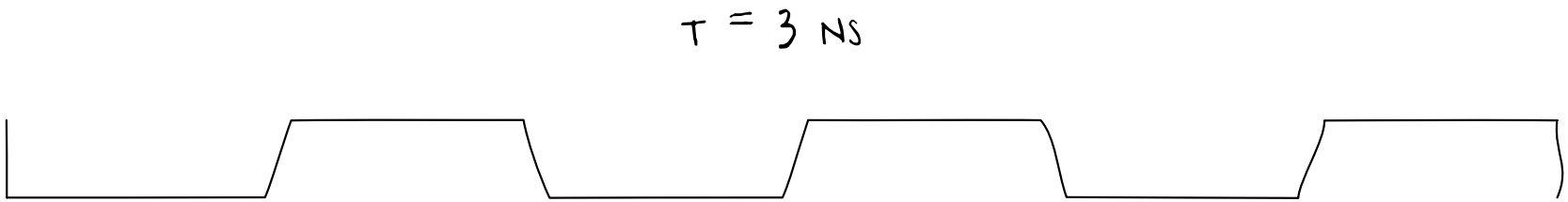
- Comparativo entre ciclo único e multiciclo
  - Instrução hipotética que calcula $f(x) = x^8$ (multiciclo)

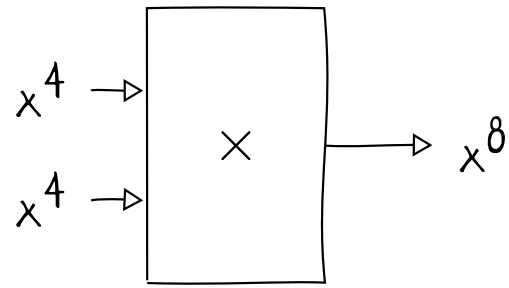




Multiciclo

- Comparativo entre ciclo único e multiciclo
  - Instrução hipotética que calcula $f(x) = x^8$ (multiciclo)





2 REGISTRADORES

1 MULTIPLICADOR

Multiciclo

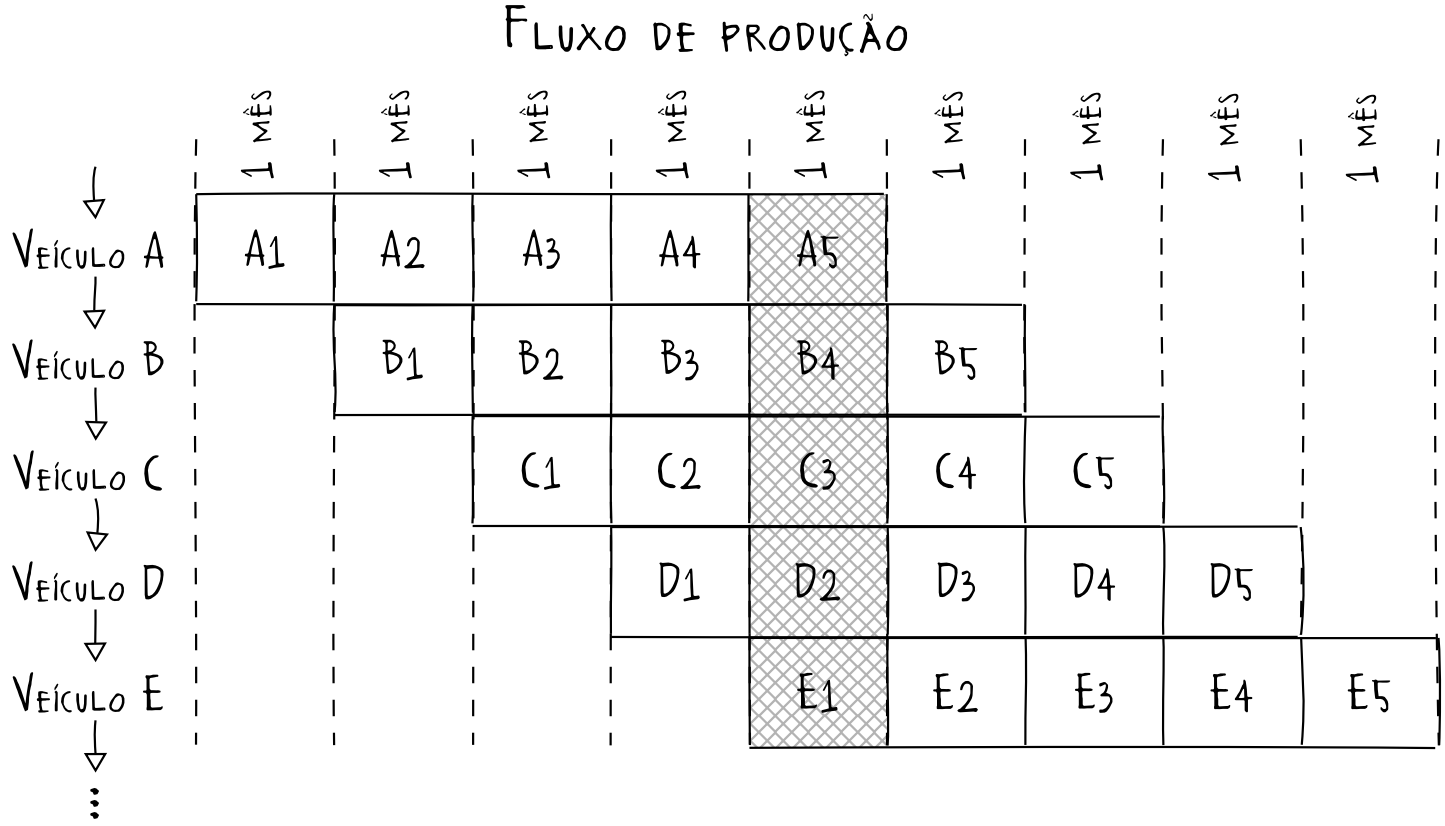
- Execução em multiciclo
  - Vantagens
    - ✓ Melhora aproveitamento dos recursos de hardware
    - ✓ Reduz a quantidade de operações por ciclo
    - ✓ Aumenta a frequência de relógio

Multiciclo

- Execução em multiciclo
  - Vantagens
    - ✓ Melhora aproveitamento dos recursos de hardware
    - ✓ Reduz a quantidade de operações por ciclo
    - ✓ Aumenta a frequência de relógio
  - Desvantagens
    - ✗ Não reduz o tempo para execução das instruções
    - ✗ Mantém a mesma taxa de execução do ciclo único

Pipeline

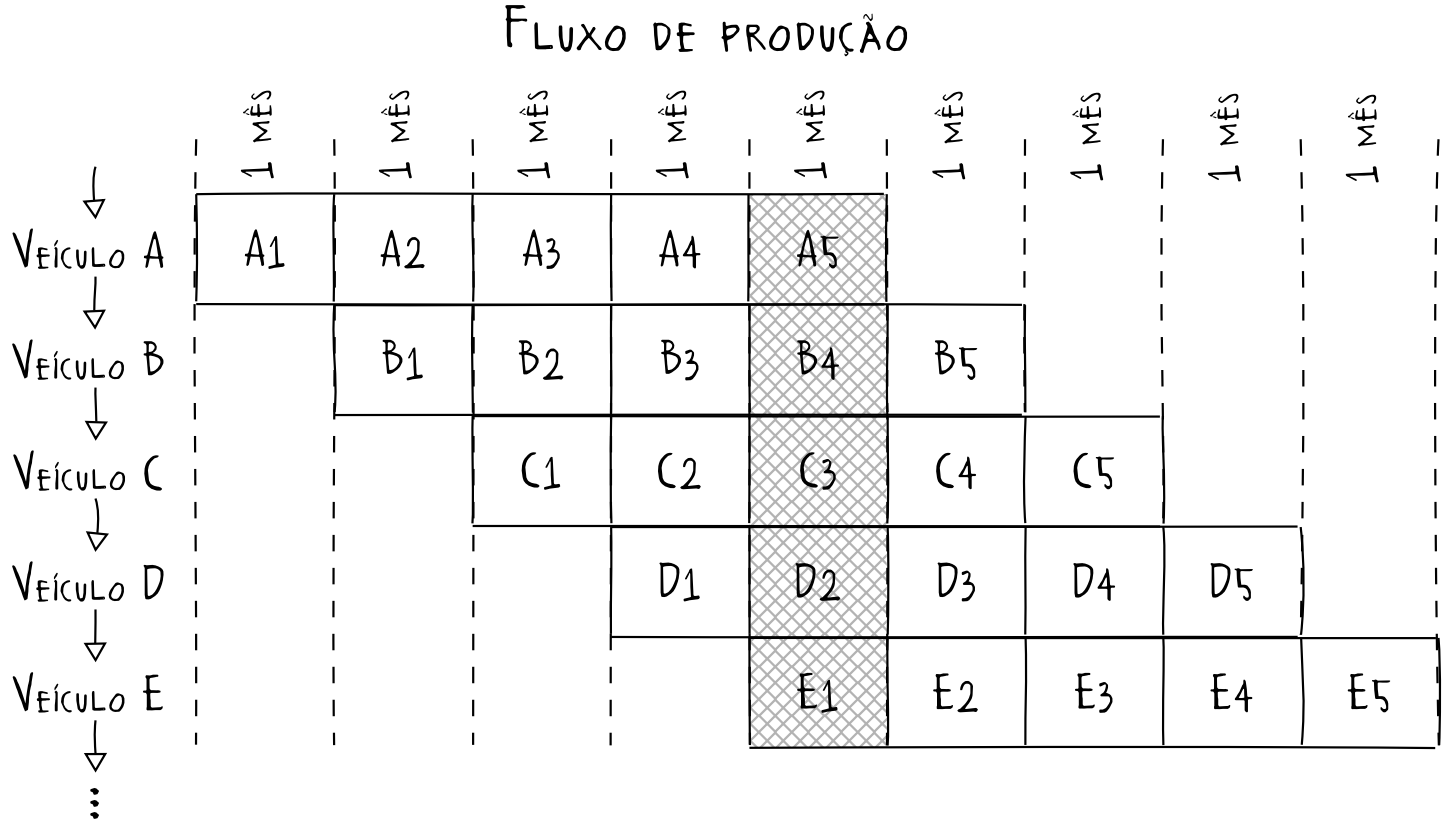
- Analogia com a produção de veículos em série



As etapas de produção são sobrepostas com diferentes veículos, executando concorrentemente em estágios distintos

Pipeline

- Analogia com a produção de veículos em série



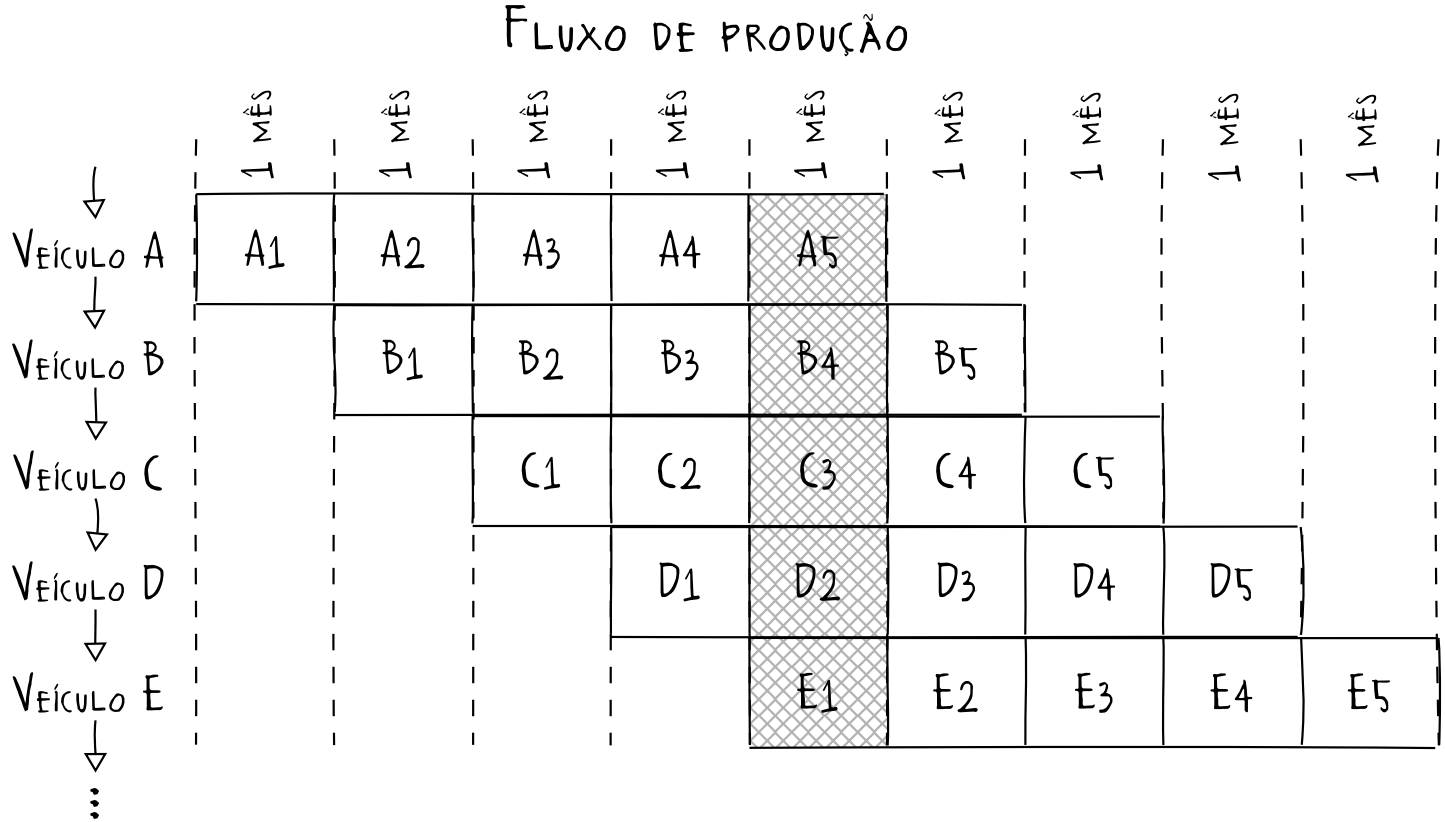
As etapas de produção são sobrepostas com diferentes veículos, executando concorrentemente em estágios distintos

↑

Aumento da taxa de execução e melhor aproveitamento do hardware

Pipeline

- Analogia com a produção de veículos em série

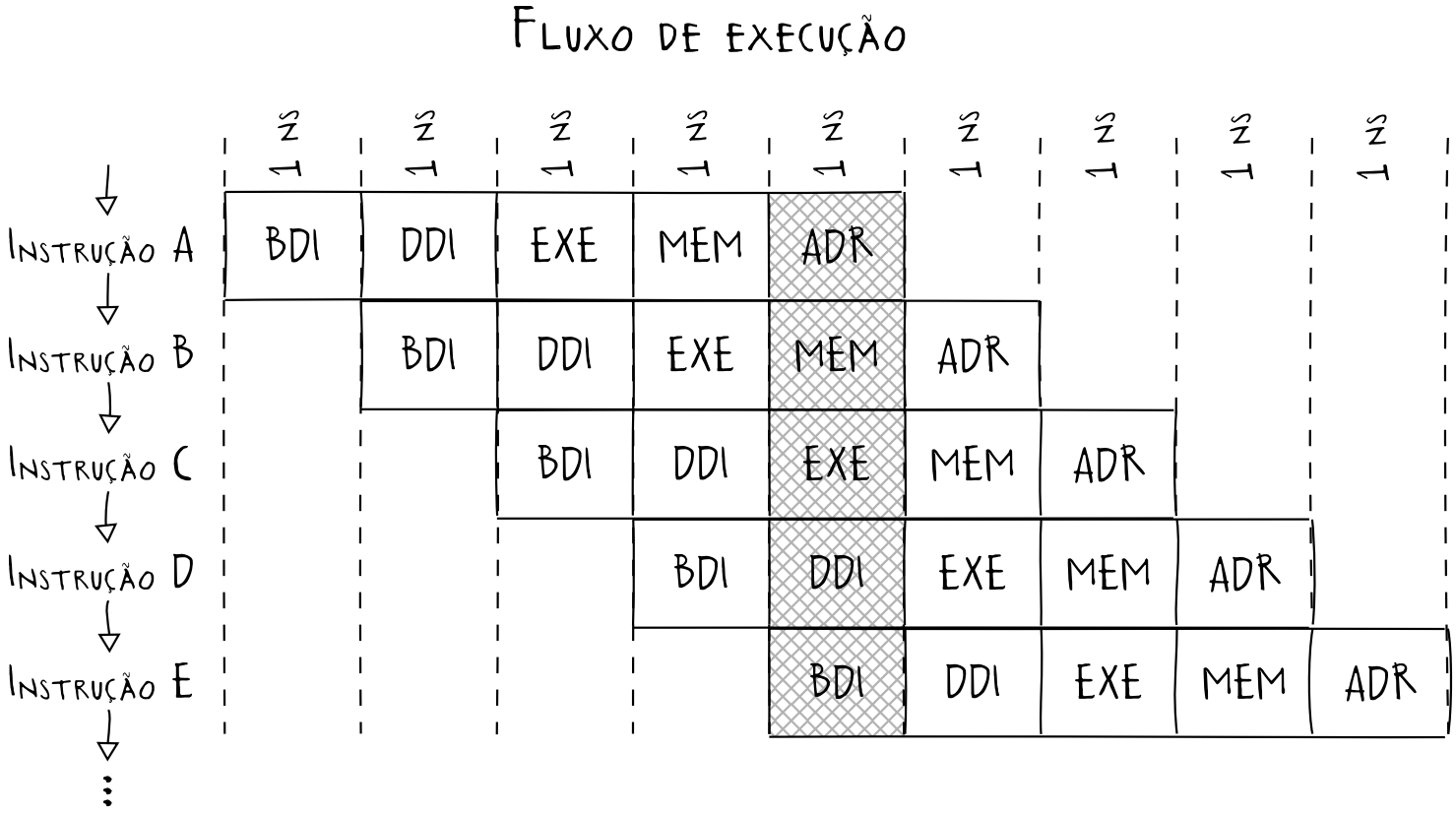


As etapas de produção são sobrepostas com diferentes veículos, executando concorrentemente em estágios distintos

Cada veículo é fabricado com o mesmo tempo e recursos, entretanto a taxa de produção pode ser aumentada em até 5 vezes

Pipeline

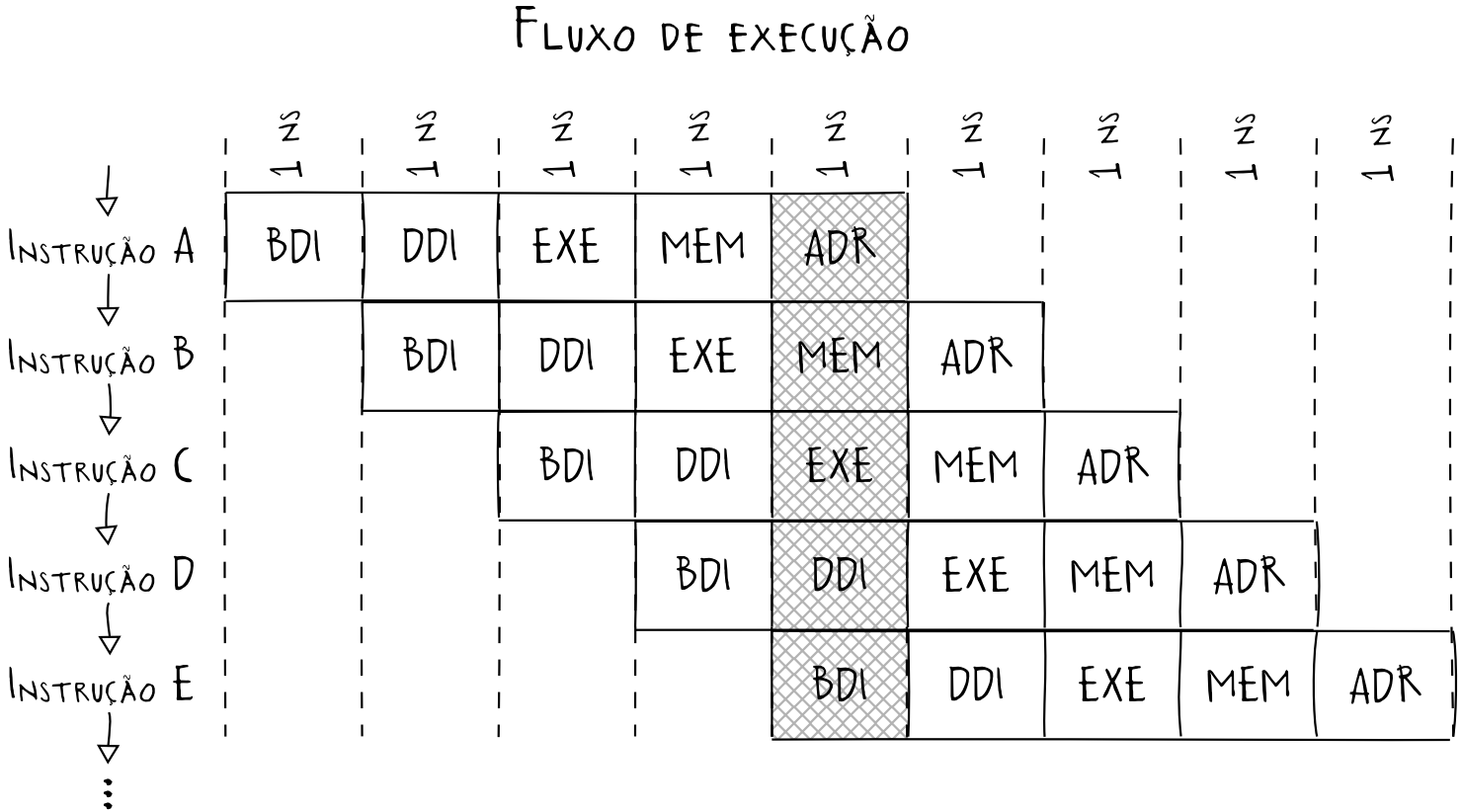
- Execução em *pipeline*



As instruções são executadas de forma concorrente e sobreposta nos estágios do pipeline

Pipeline

- Execução em *pipeline*

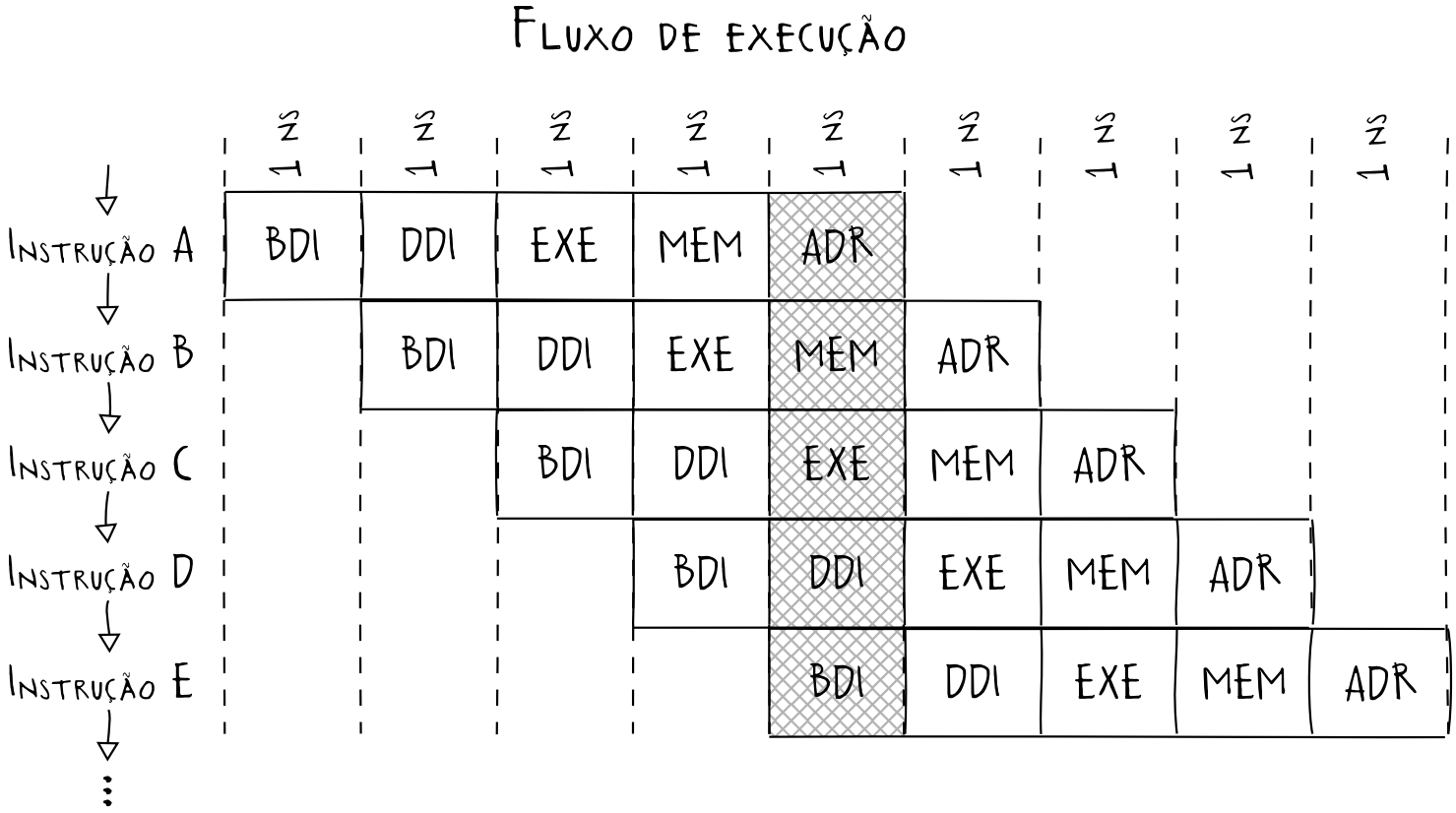


As instruções são executadas de forma concorrente e sobreposta nos estágios do pipeline

A taxa de execução é aumentada em até 5 vezes, com 1 instrução a cada 1 ns ao invés de 5 ns

Pipeline

- Execução em *pipeline*



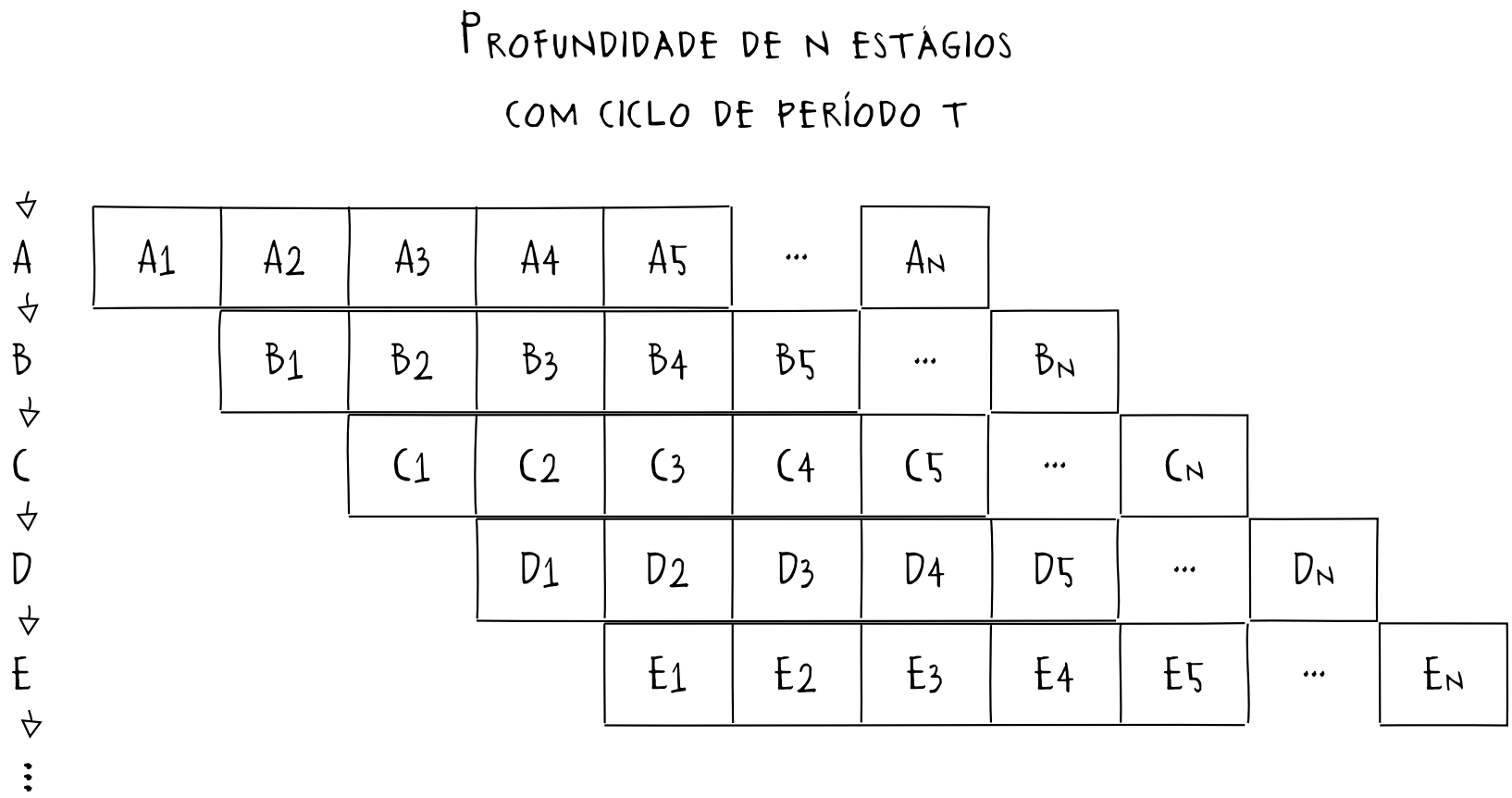
As instruções são executadas de forma concorrente e sobreposta nos estágios do pipeline

↑

Pipeline não é paralelismo!

Pipeline

- Projeto de *pipeline*



EM CONDIÇÕES IDEAIS, A QUANTIDADE E O TEMPO DE PROCESSAMENTO DOS ESTÁGIOS DEFINEM A TAXA DE EXECUÇÃO (N / T INSTRUÇÕES/NS)

Pipeline

- Projeto de *pipeline*
  - Aumento de desempenho do processador
    - Decorrente do aumento da taxa de execução, com melhor aproveitamento dos recursos de hardware
    - Por executarem sequencialmente, não existe redução do tempo execução individual das instruções

Pipeline

- Projeto de *pipeline*
  - Aumento de desempenho do processador
    - Decorrente do aumento da taxa de execução, com melhor aproveitamento dos recursos de hardware
    - Por executarem sequencialmente, não existe redução do tempo execução individual das instruções
  - Requisitos para arquitetura
    - Poucas variações nos formatos das instruções
    - As instruções devem ter o mesmo tamanho

Pipeline

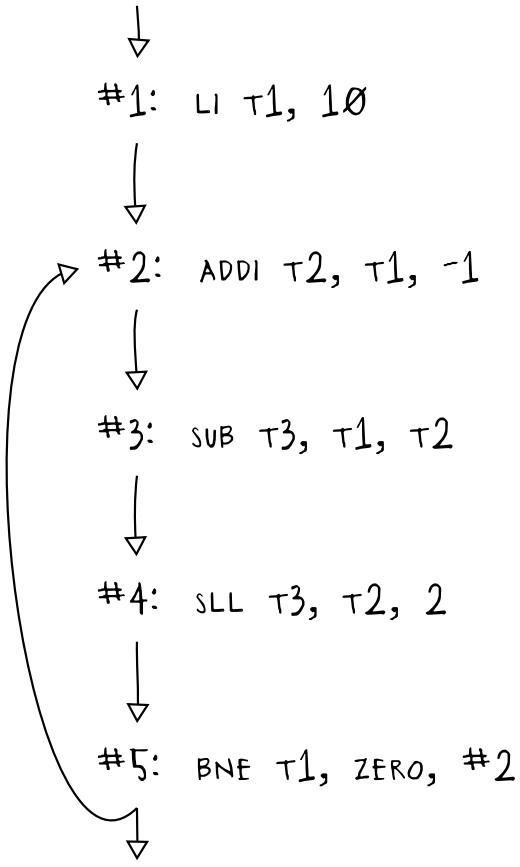
- Cenários de execução em *pipeline*
  - Ideal
    - O desempenho do processador é multiplicado pelo número de estágios utilizados no *pipeline*
    - Não existem conflitos na execução das instruções e a taxa de execução só depende da frequência de operação e da quantidade de estágios

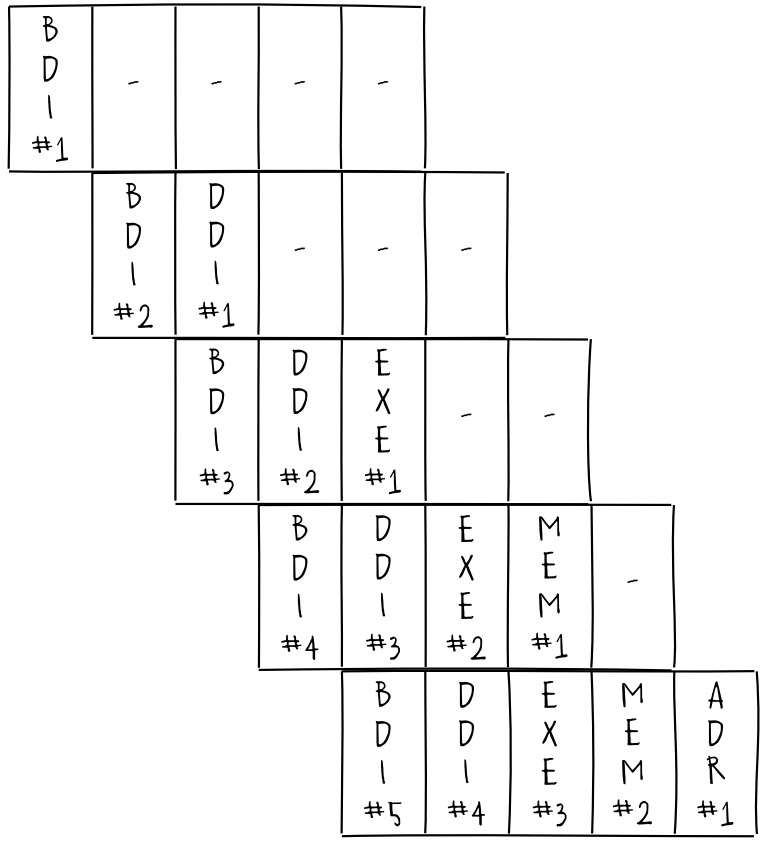
Pipeline

- Cenários de execução em *pipeline*
  - Ideal
    - O desempenho do processador é multiplicado pelo número de estágios utilizados no *pipeline*
    - Não existem conflitos na execução das instruções e a taxa de execução só depende da frequência de operação e da quantidade de estágios
  - Real
    - O desempenho é variável, sendo diretamente afetado pela sequência de instruções executadas
    - Como podem existir conflitos que precisam ser tratados, o desempenho do sistema é reduzido para manter o comportamento sequencial das instruções

Pipeline

- O que são conflitos de execução no *pipeline*?

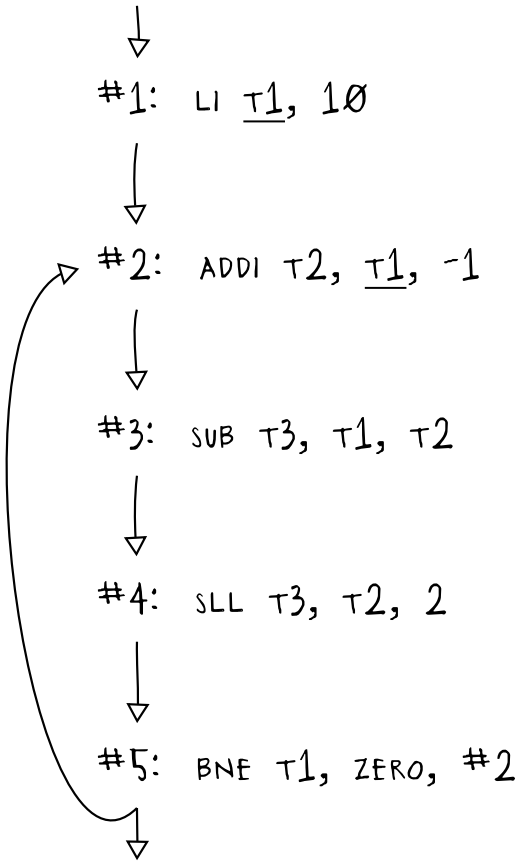


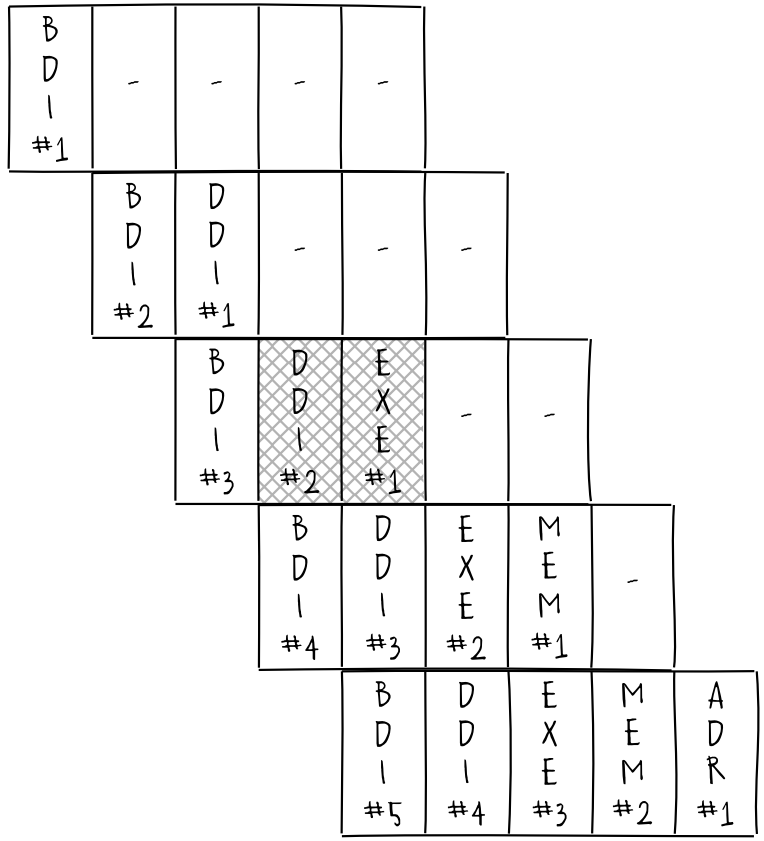


OS CONFLITOS OCORREM QUANDO UMA INSTRUÇÃO
NÃO PODE EXECUTAR NO PRÓXIMO ESTÁGIO
SEM ALTERAR O COMPORTAMENTO SEQUENCIAL ESPERADO

Pipeline

- O que são conflitos de execução no *pipeline*?

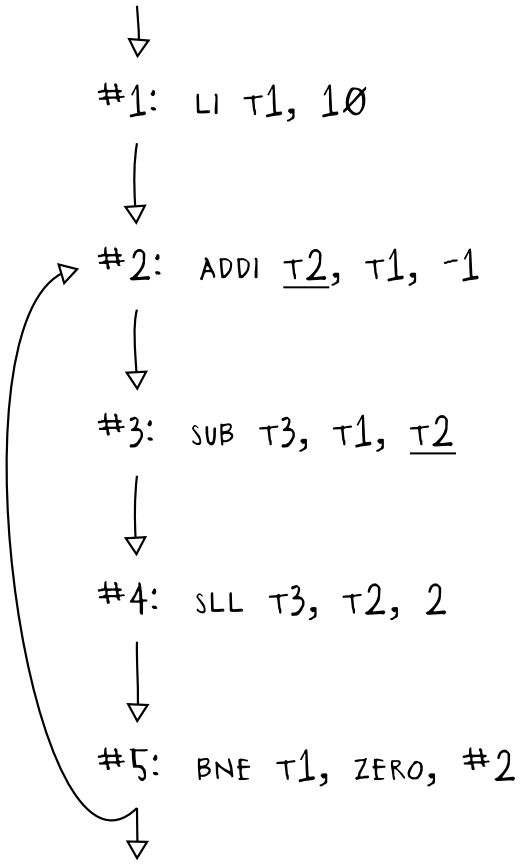


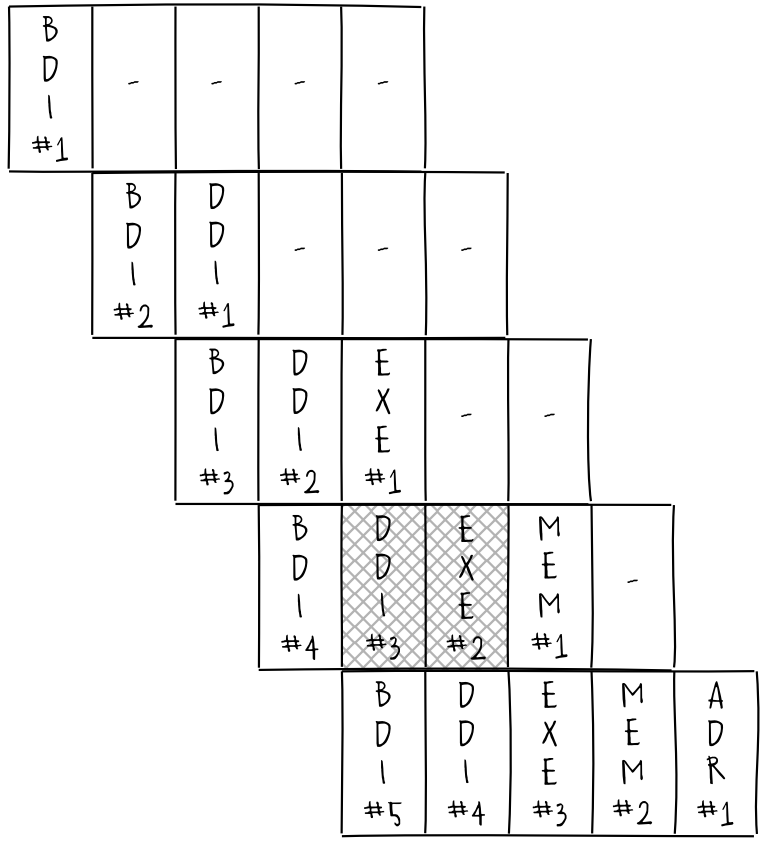


CONFLITO ENTRE #1 E #2: T1 = 10 <-> 0

Pipeline

- O que são conflitos de execução no *pipeline*?

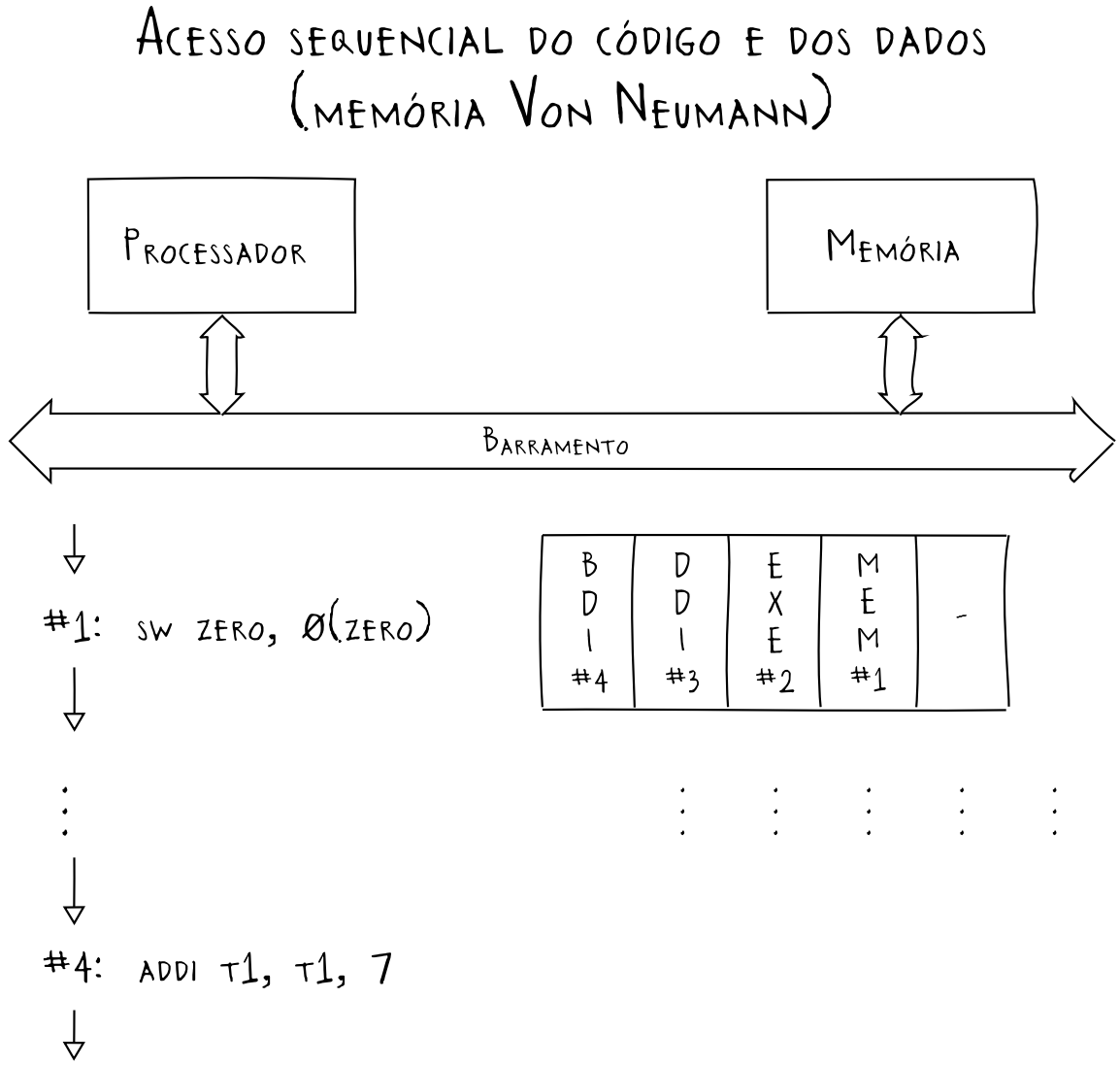




CONFLITO ENTRE #2 E #3: T2 = 9 <-> 0

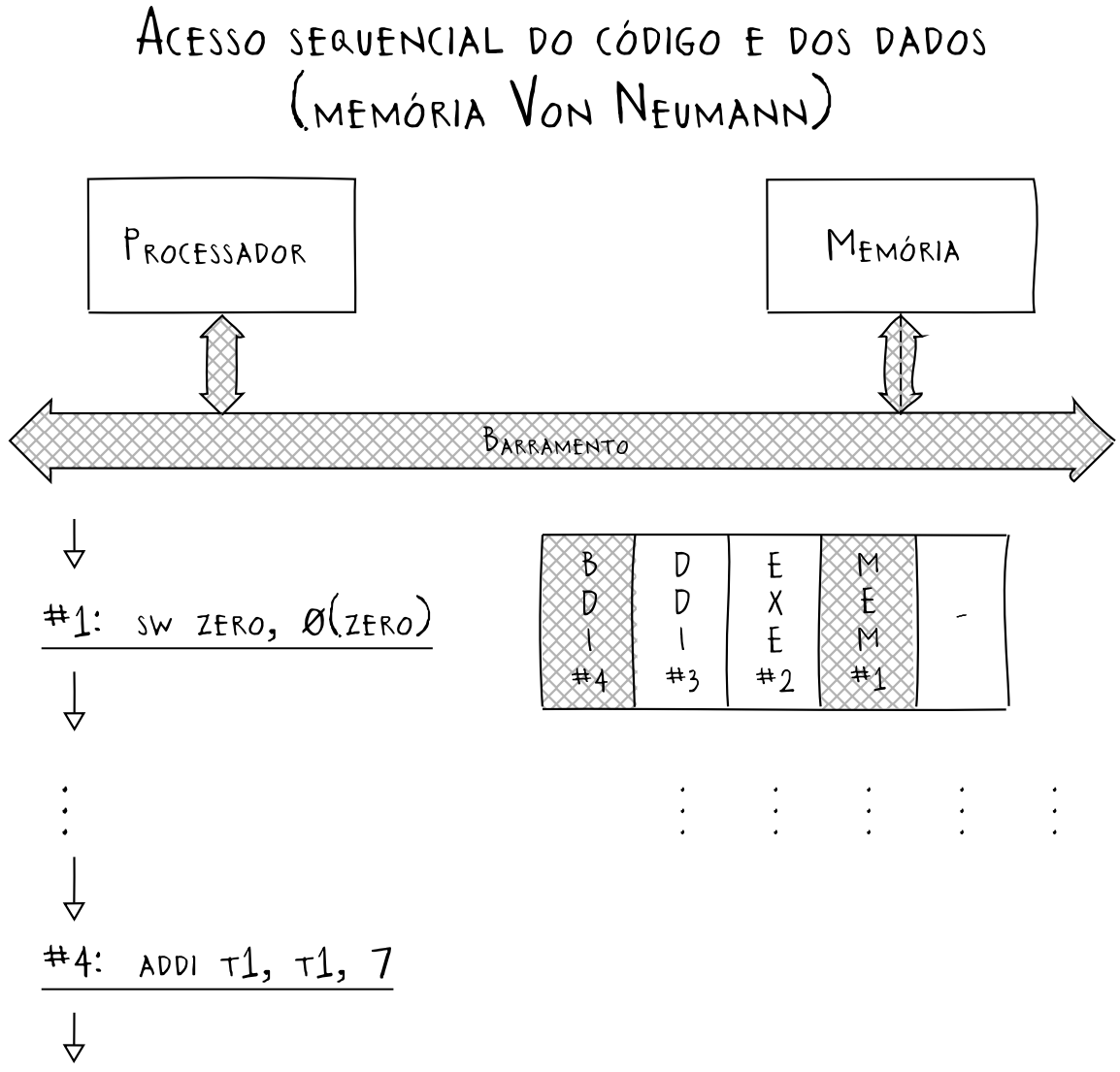
Conflito estrutural

- Limitações no projeto do sistema



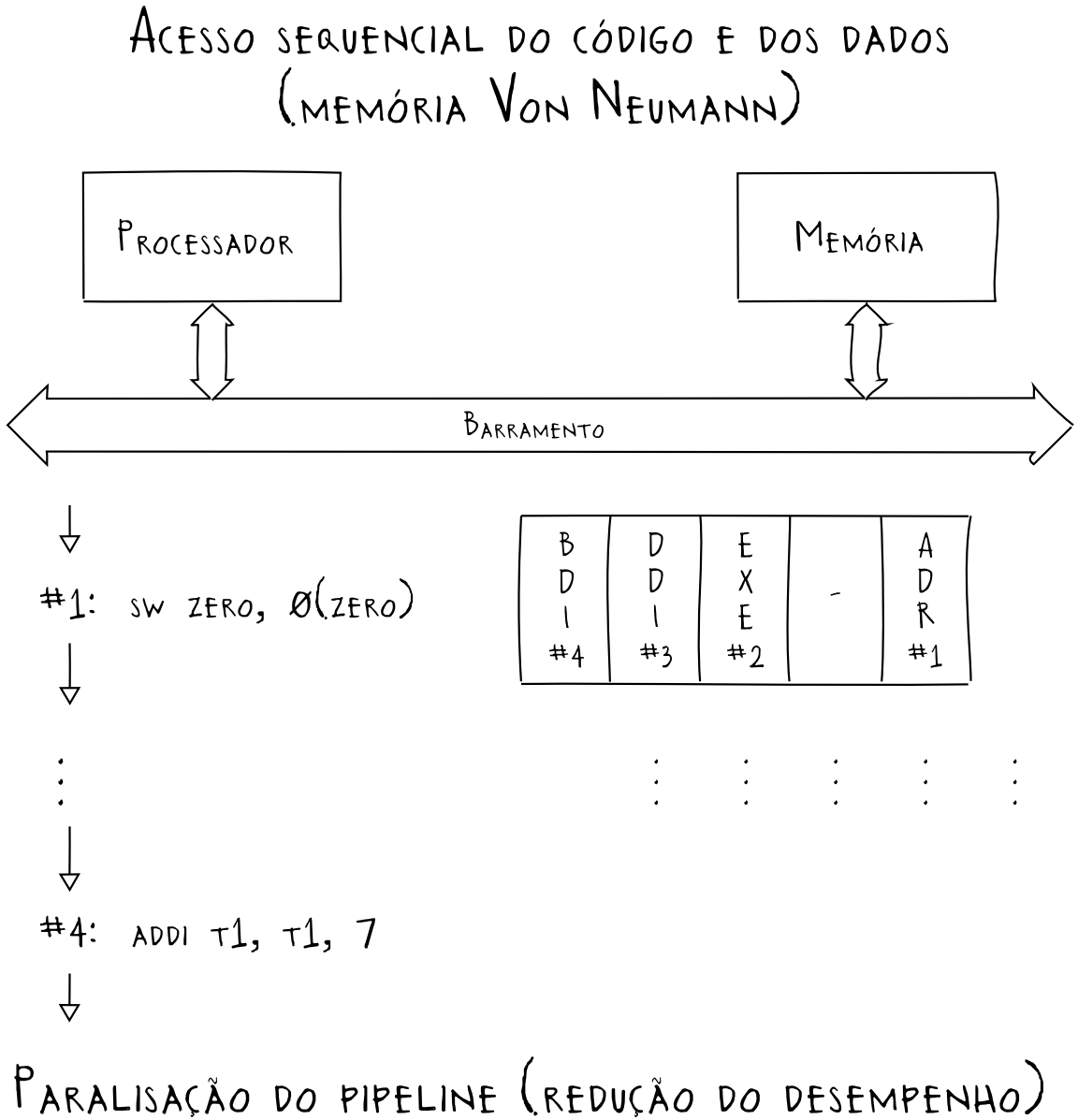
Conflito estrutural

- Limitações no projeto do sistema



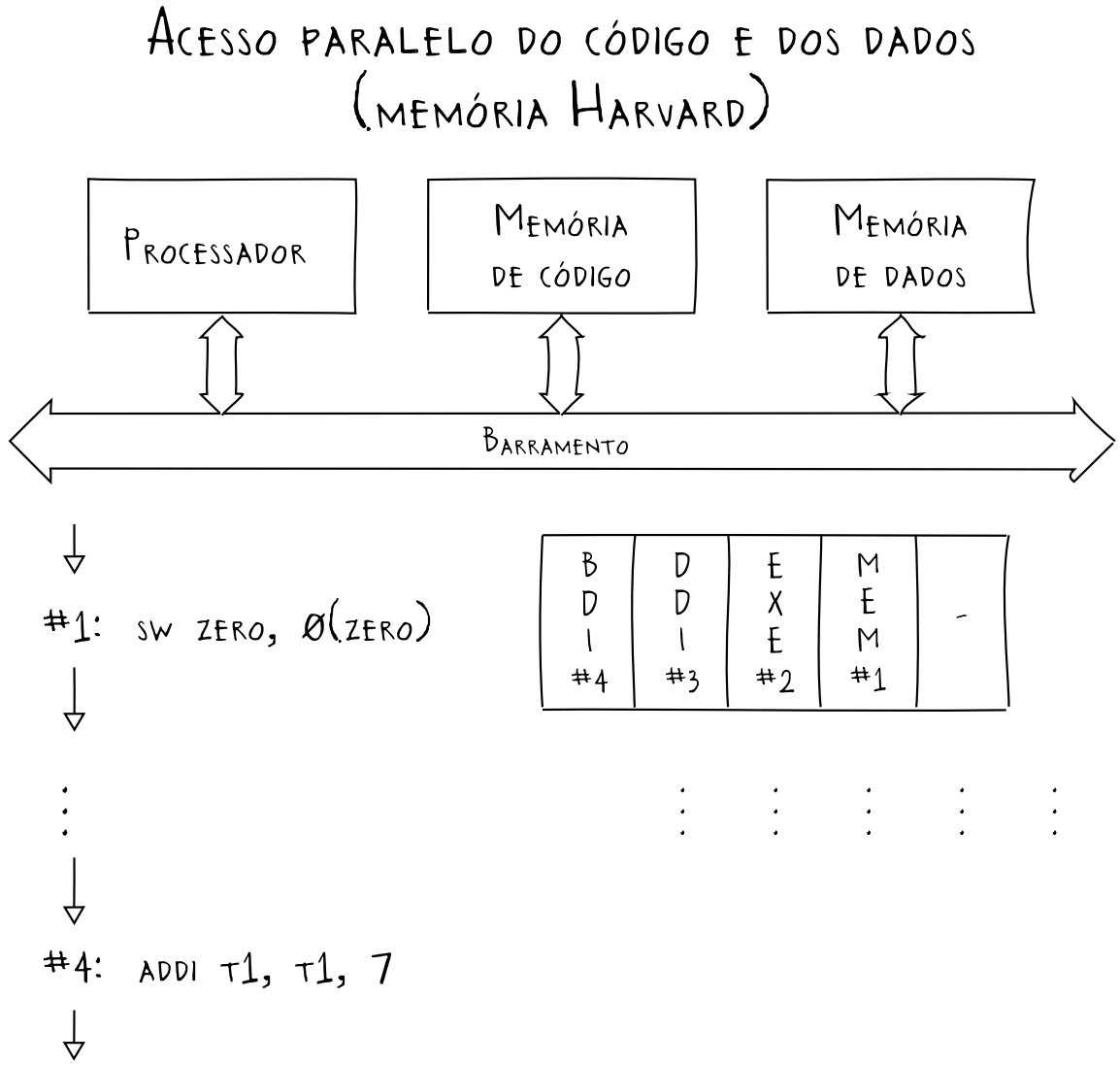
Conflito estrutural

- Limitações no projeto do sistema



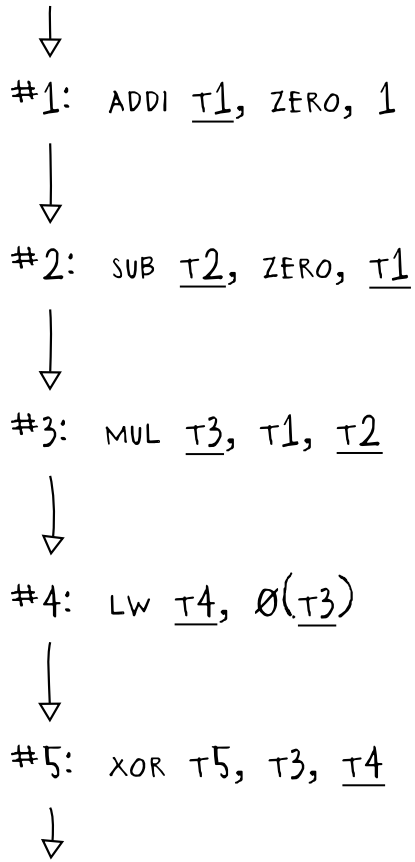
Conflito estrutural

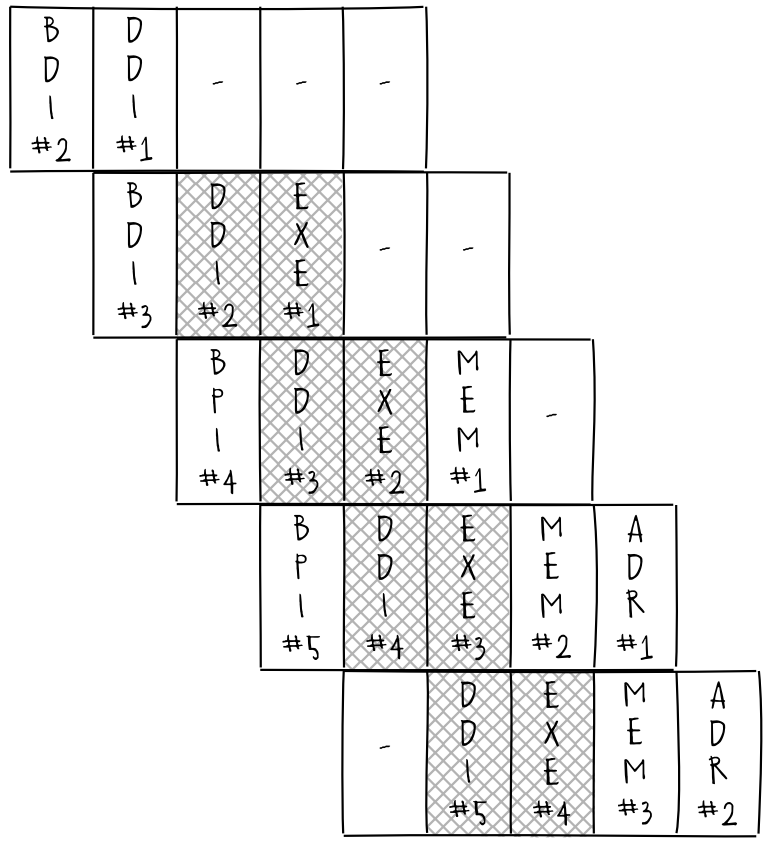
- Limitações no projeto do sistema



Conflito de dados

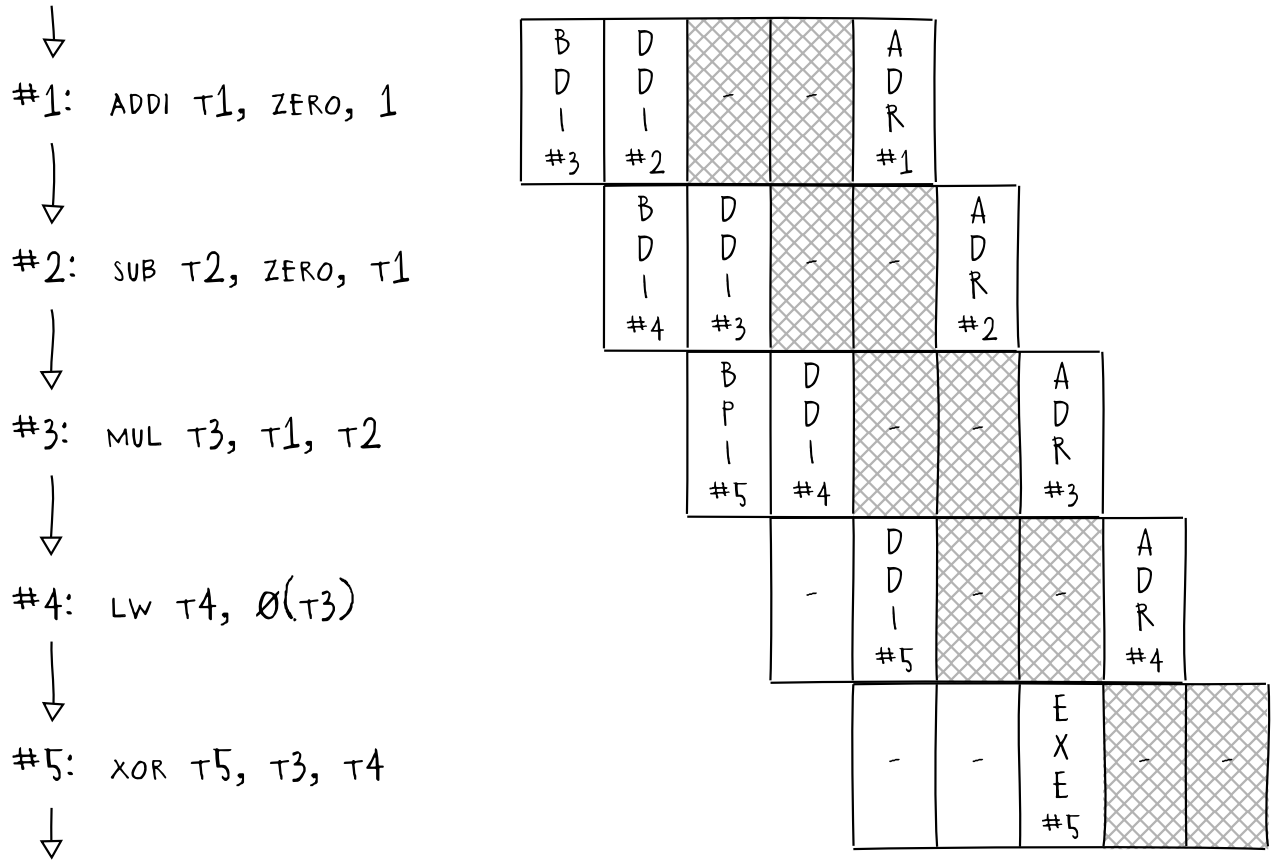
- Criado por uma dependência de dados entre instruções consecutivas executando no *pipeline*





Conflito de dados

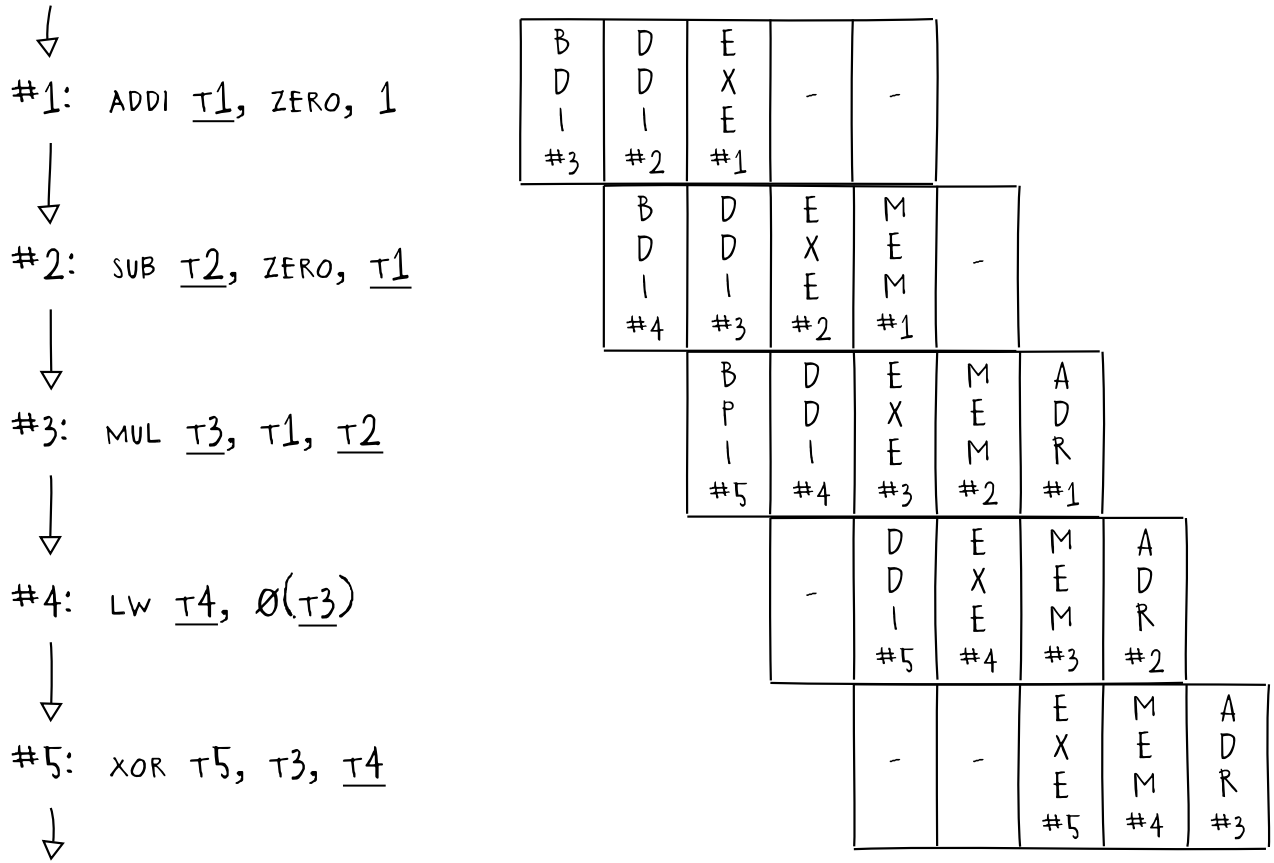
- Inserção de atrasos ou bolhas no *pipeline*



É UMA SOLUÇÃO SIMPLES, MAS INEFICIENTE

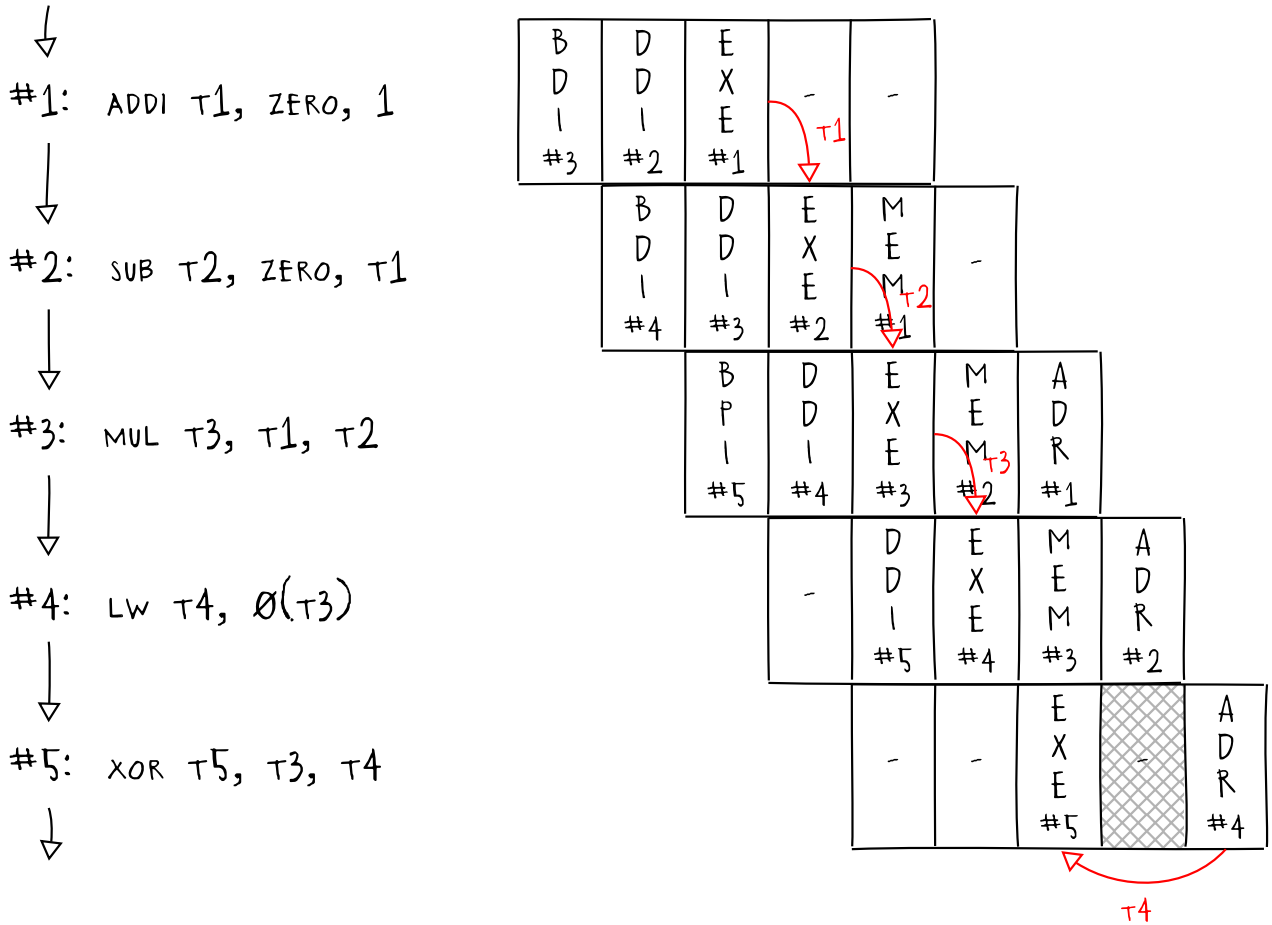
Conflito de dados

- Adiantamento de dados



Conflito de dados

- Adiantamento de dados

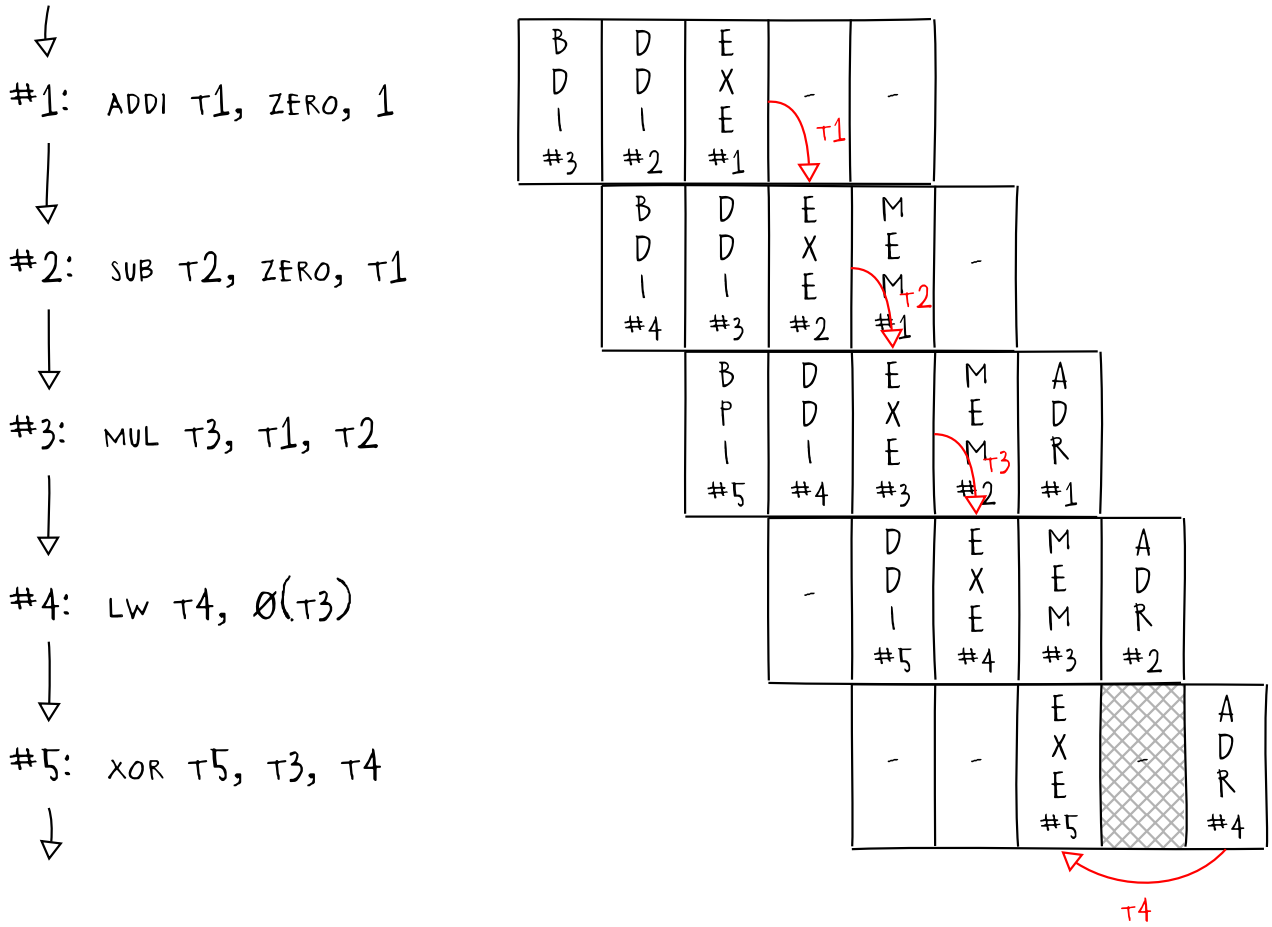


Reduz os atrasos na execução,

mas necessita de hardware dedicado

Conflito de dados

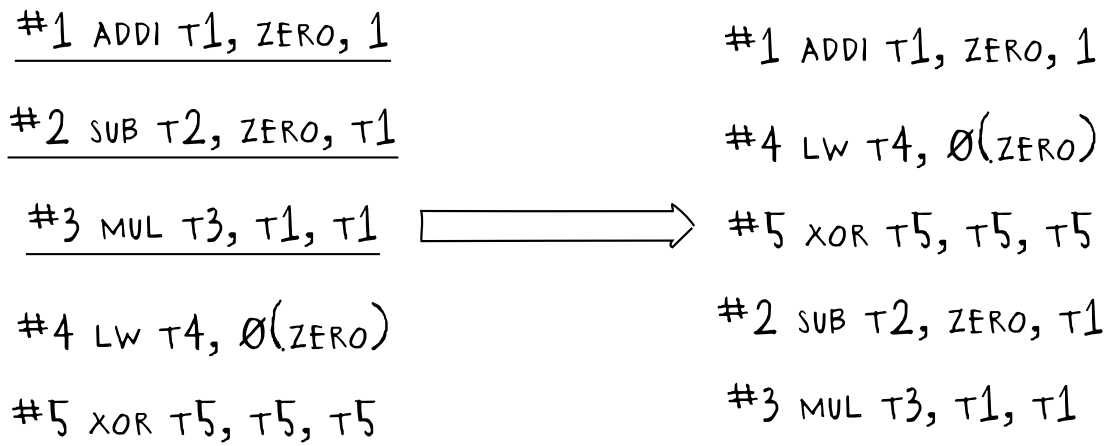
- Adiantamento de dados



SEM ACESSOS À MEMÓRIA (LOAD-STORE),
É EVITADA A INSERÇÃO DE BOLHAS NO PIPELINE

Conflito de dados

- Papel do compilador
  - A reorganização do código gerado pelo compilador pode eliminar a ocorrência dos conflitos de dados no *pipeline*



A REORGANIZAÇÃO DAS INSTRUÇÕES #1, #2 E #3 ELIMINA OS CONFLITOS SEM IMPACTO NO DESEMPENHO

Conflito de dados

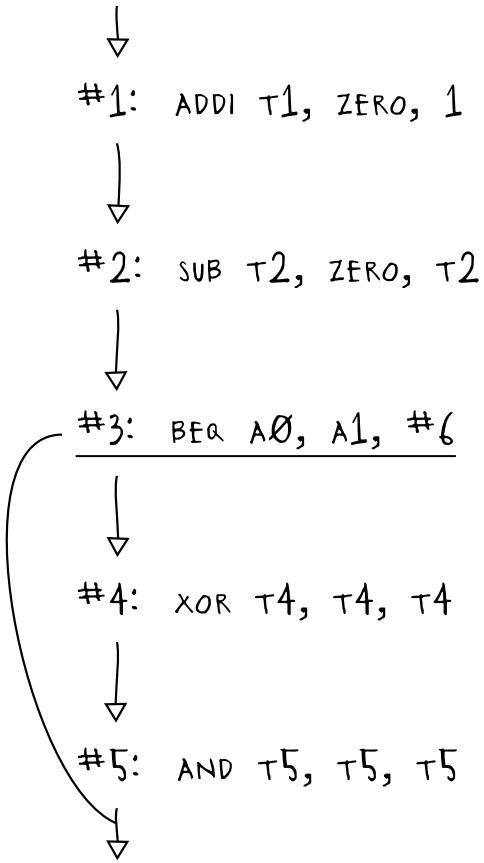
- Quem é responsável por tratar conflitos no *pipeline*?
  - Hardware
    - Durante a popularização do projeto em *pipeline* no final dos anos 70, os compiladores eram muito limitados
    - O projeto do hardware era mais complexo para detectar e tratar conflitos em tempo de execução
    - Em última instância, o processador precisa garantir o comportamento correto na execução do software

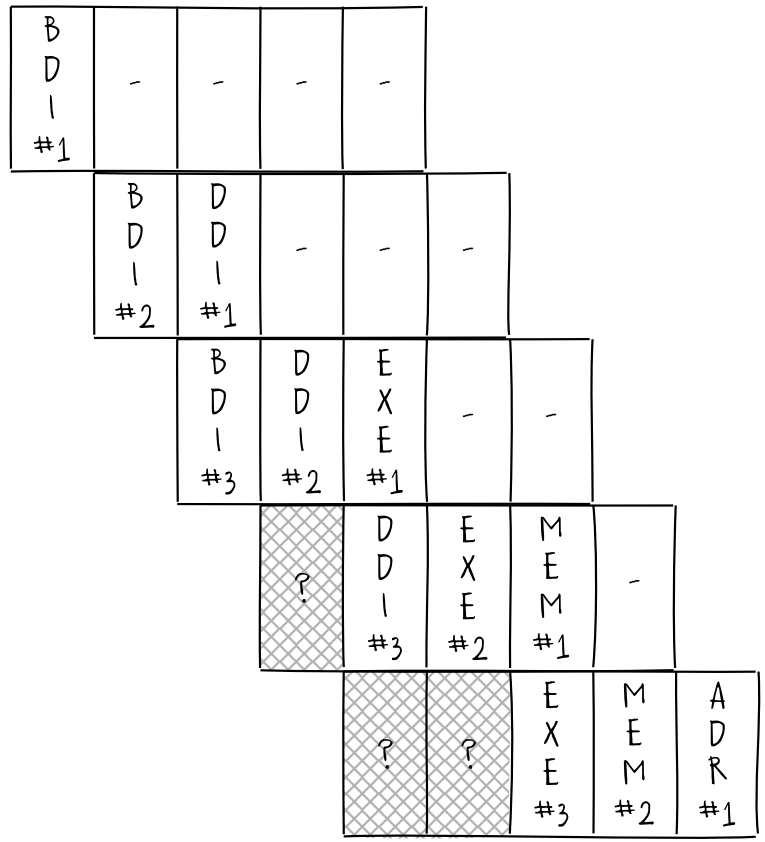
Conflito de dados

- Quem é responsável por tratar conflitos no *pipeline*?
  - Hardware
    - Durante a popularização do projeto em *pipeline* no final dos anos 70, os compiladores eram muito limitados
    - O projeto do hardware era mais complexo para detectar e tratar conflitos em tempo de execução
    - Em última instância, o processador precisa garantir o comportamento correto na execução do software
  - Software
    - Maior flexibilidade para otimizações e melhorias
    - Ferramentas e técnicas de compilação avançadas
    - Simplificação do projeto de processador, delegando para o compilador tarefas de tratamento de conflito

Conflito de controle

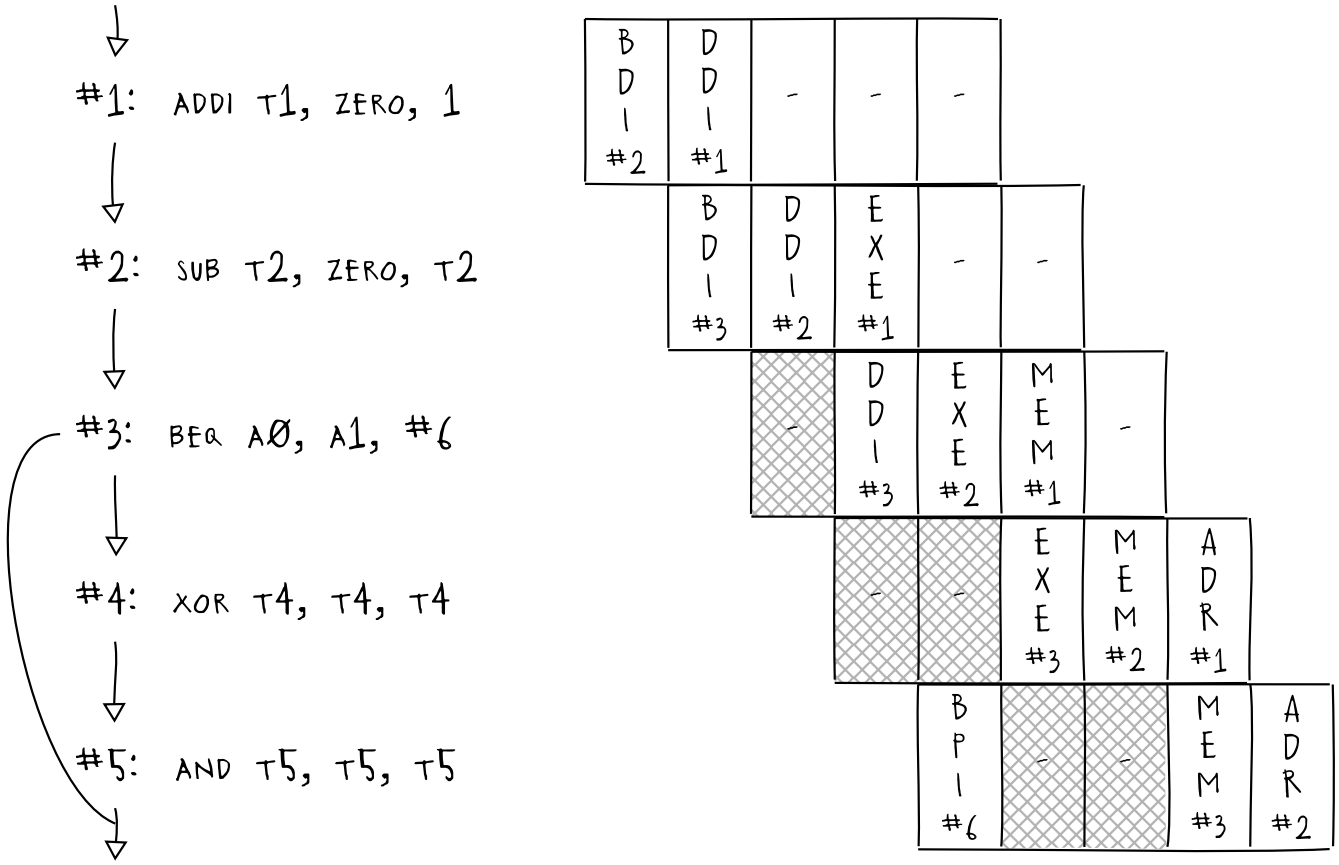
- É decorrente de instruções de controle de fluxo que ainda serão calculadas ou modificadas no *pipeline*





Conflito de controle

- Inserção de atrasos ou bolhas no *pipeline*



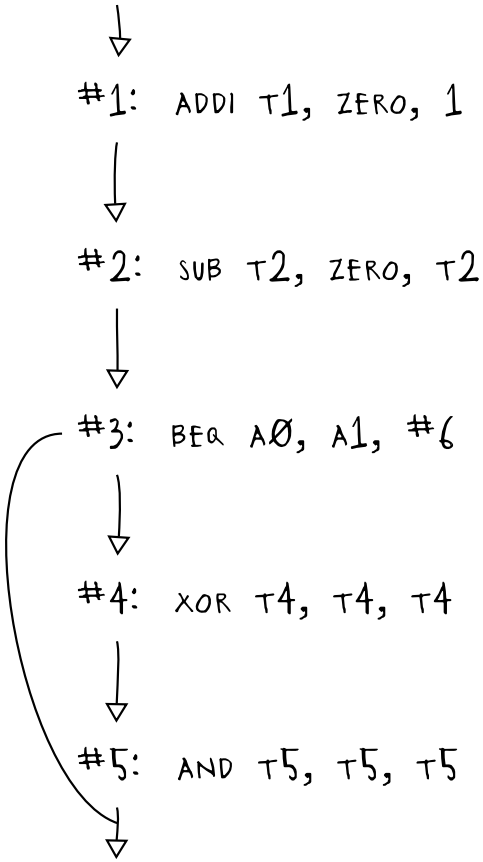
É UMA SOLUÇÃO SIMPLES, MAS INEFICIENTE

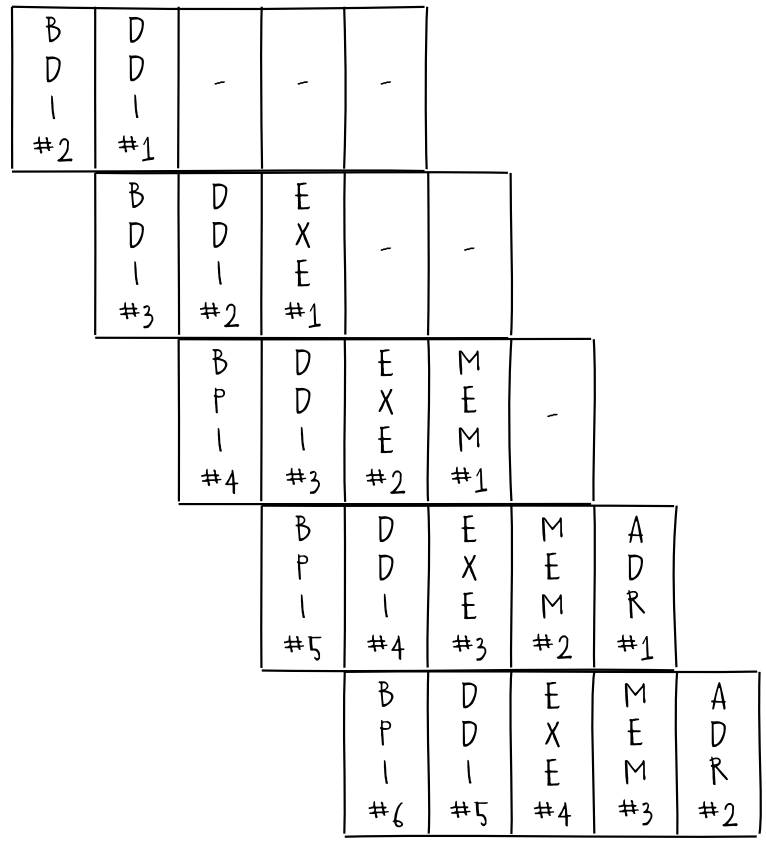
Conflito de controle

- Como prever de forma eficiente quais instruções serão executadas após o desvio no *pipeline*?
  - **Estaticamente**: assumir que o desvio não ocorre, sempre buscando sequencialmente as instruções
  - **Dinamicamente**: executar os desvios de forma especulativa, com base no histórico armazenado

Conflito de controle

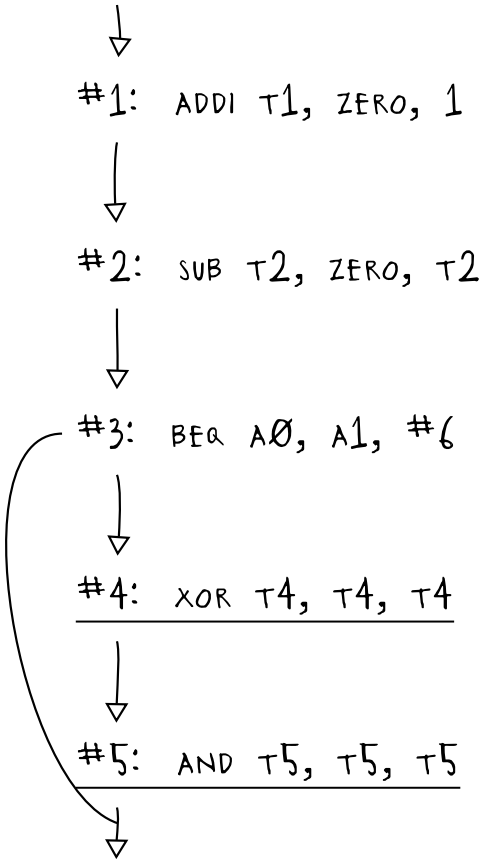
- Predição estática de desvio

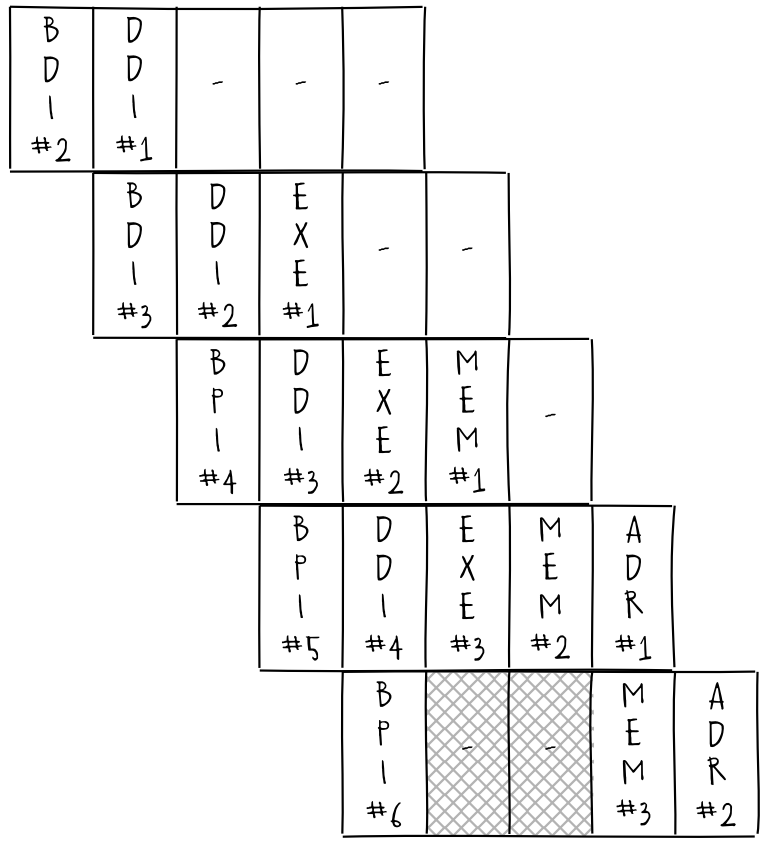




Conflito de controle

- Predição estática de desvio



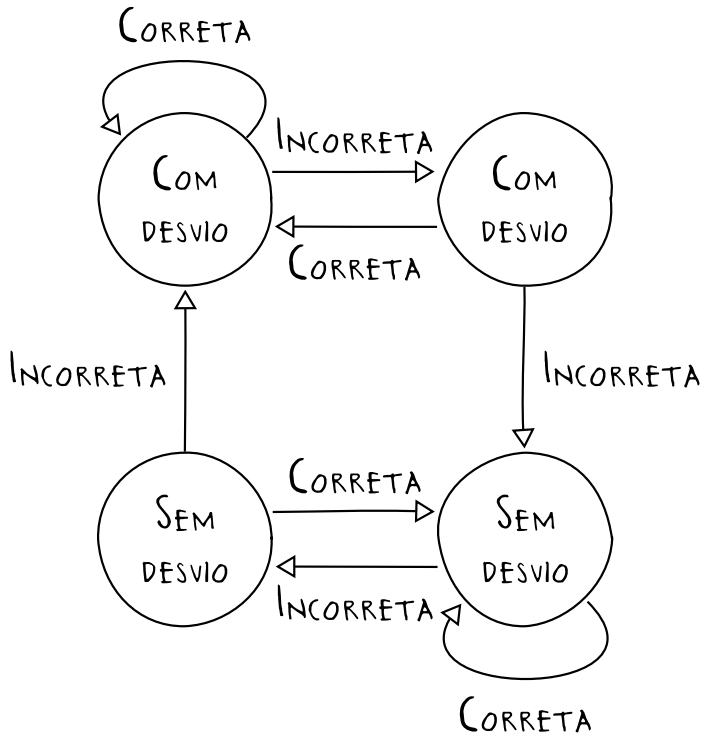


QUANDO O DESVIO OCORRE É EQUIVALENTE
À INSERÇÃO DE ATRASOS OU BOLHAS NO PIPELINE

Conflito de controle

- Predição dinâmica de desvio

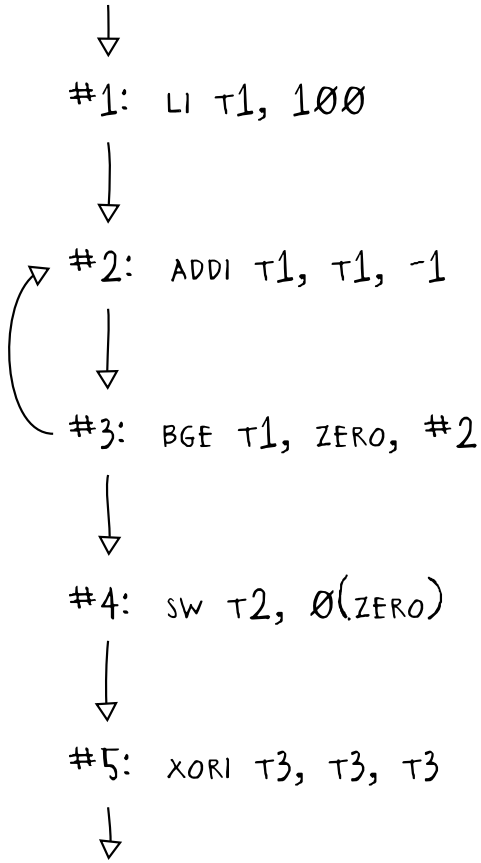
Código	Estimativa	Correta	Incorreta
00	Sem desvio	00	01
01	Sem desvio	00	10
10	Com desvio	10	11
11	Com desvio	10	00

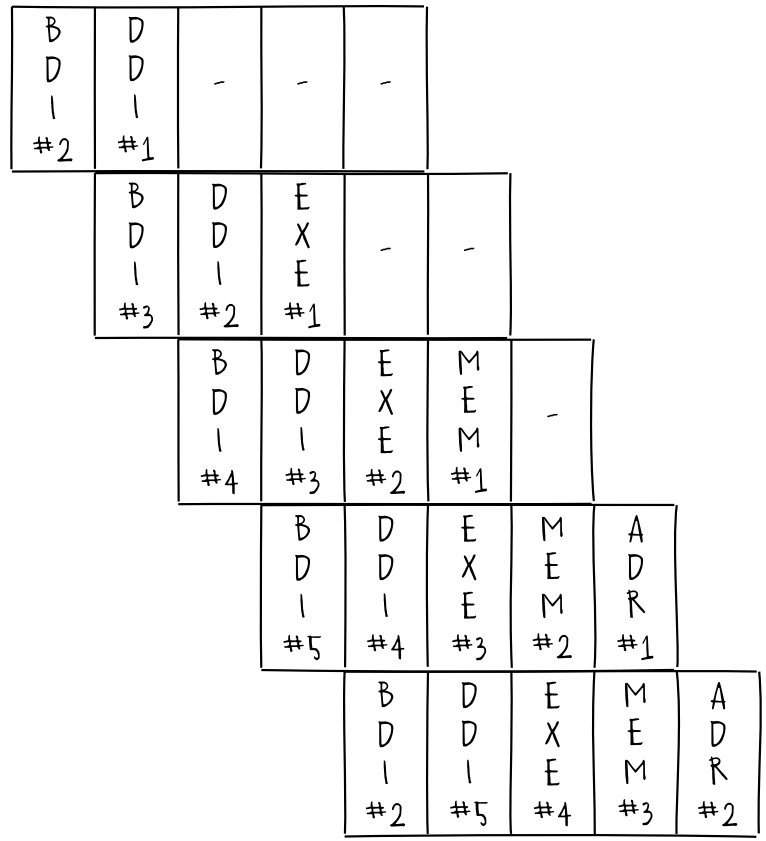


O histórico pode ser local, global ou combinado

Conflito de controle

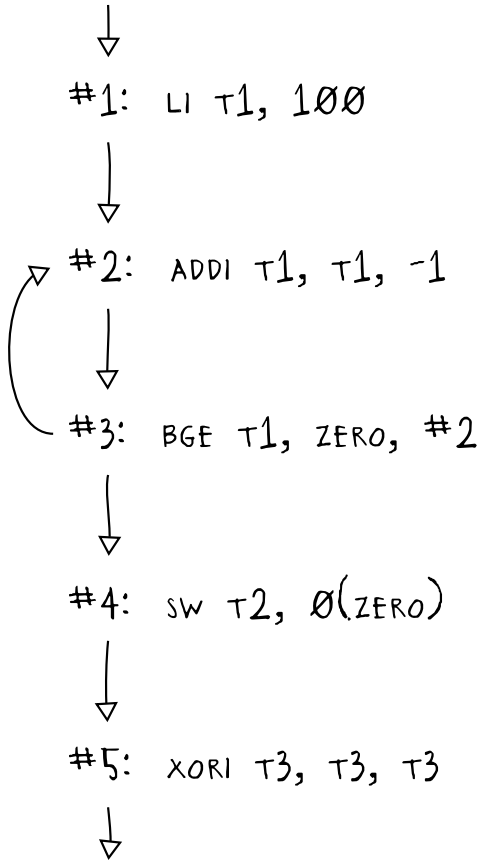
- Predição dinâmica de desvio

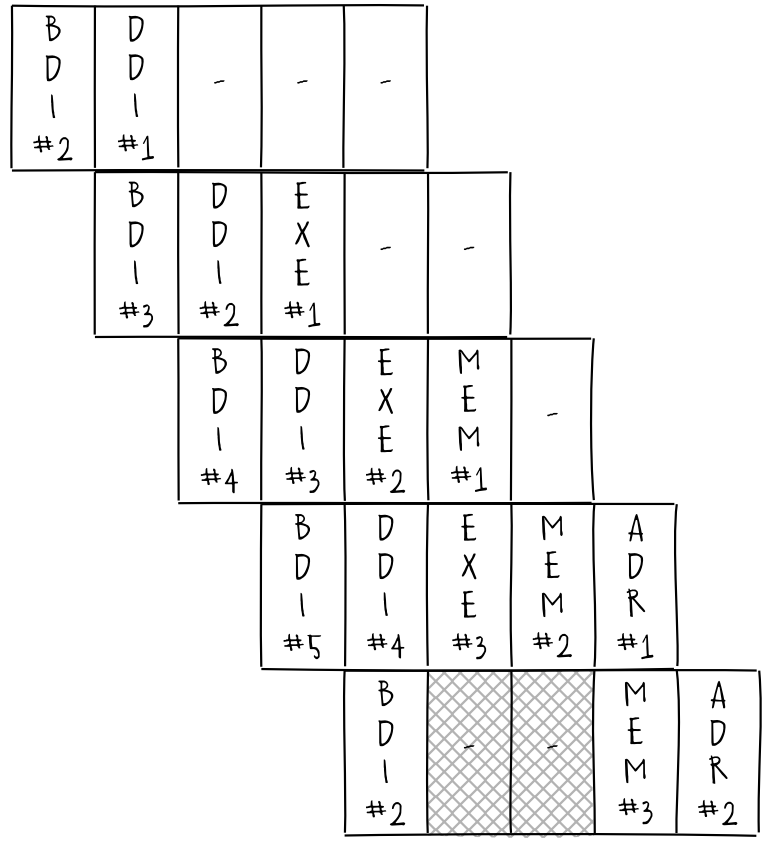




Conflito de controle

- Predição dinâmica de desvio

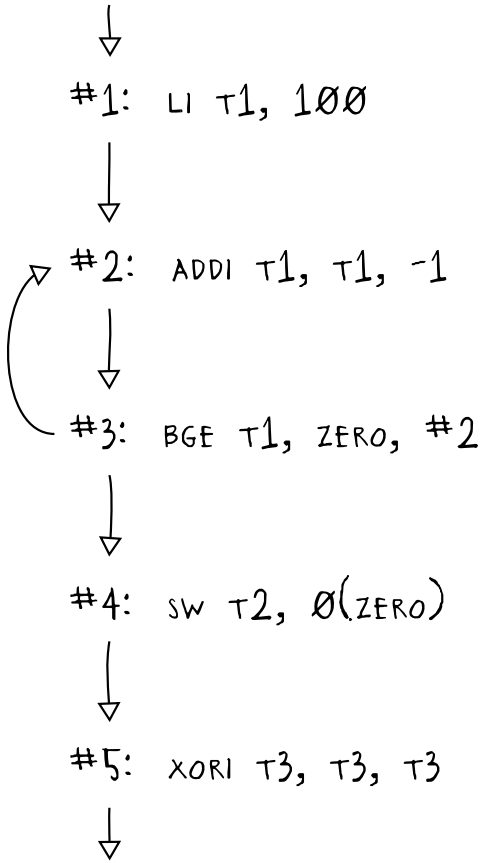


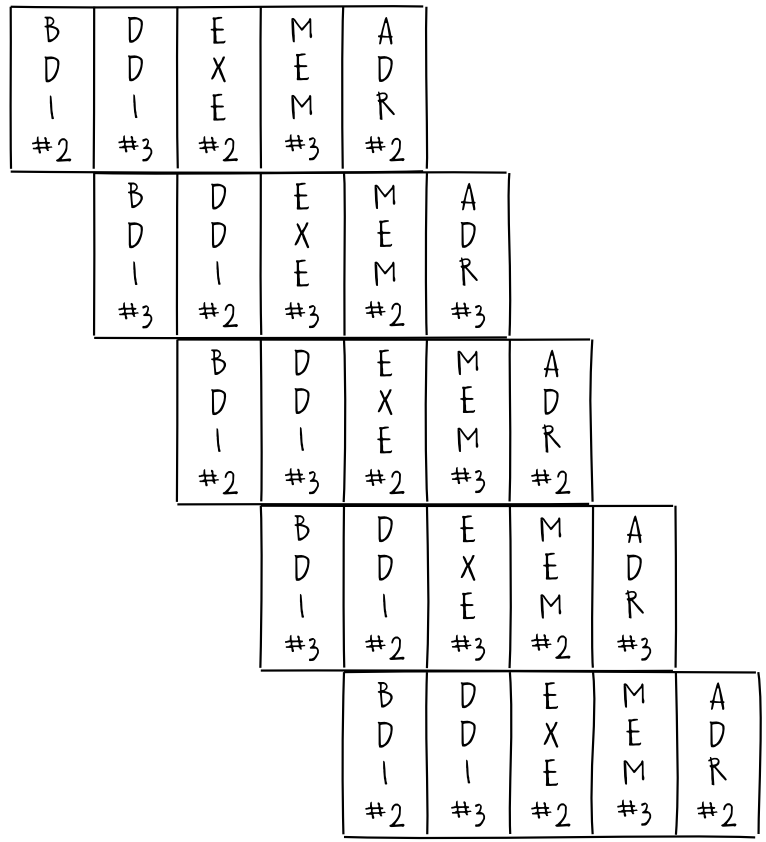


ITERAÇÕES 1 E 2 (PREDIÇÕES INCORRETAS)

Conflito de controle

- Predição dinâmica de desvio

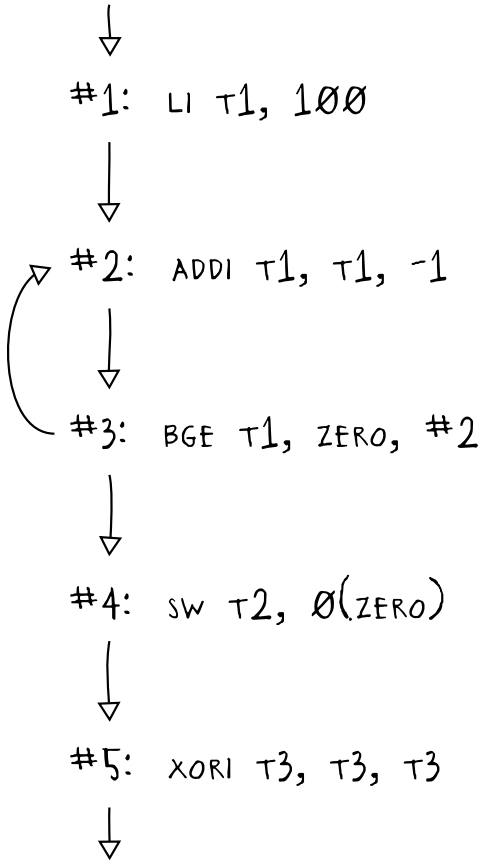


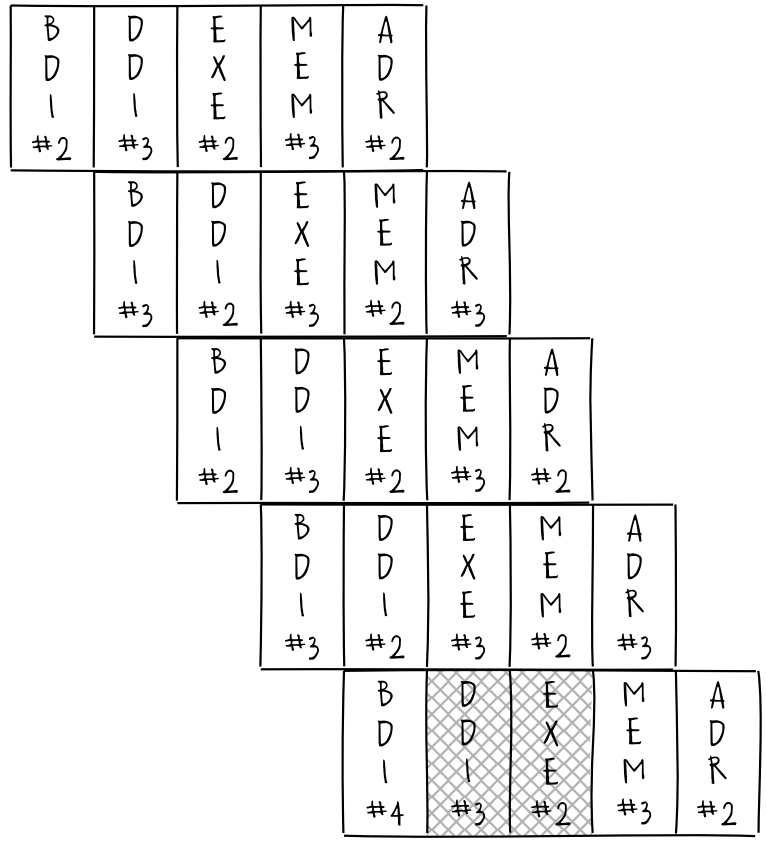


ITERAÇÕES 3 ATÉ 99 (PREDIÇÕES CORRETAS)

Conflito de controle

- Predição dinâmica de desvio

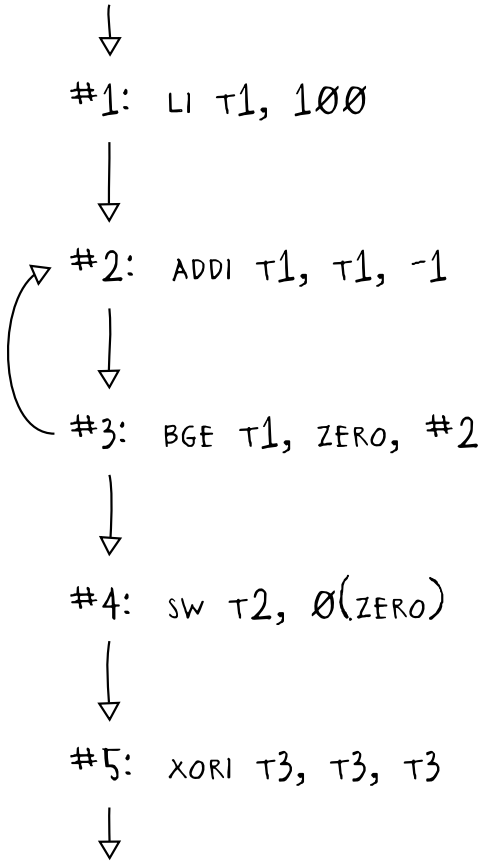


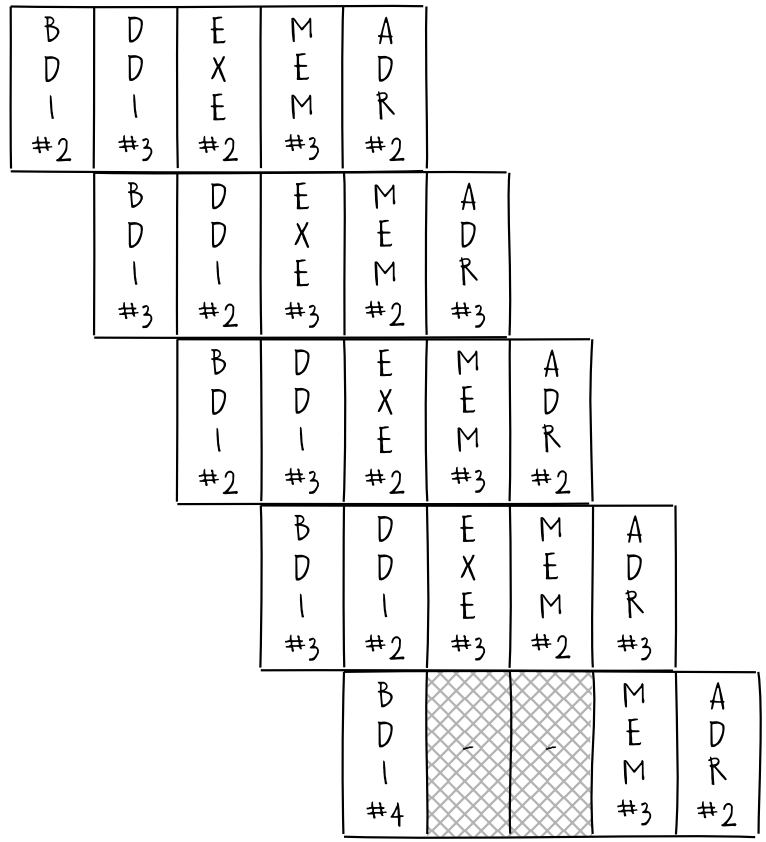


ITERAÇÃO 100 (PREDIÇÃO INCORRETA)

Conflito de controle

- Predição dinâmica de desvio

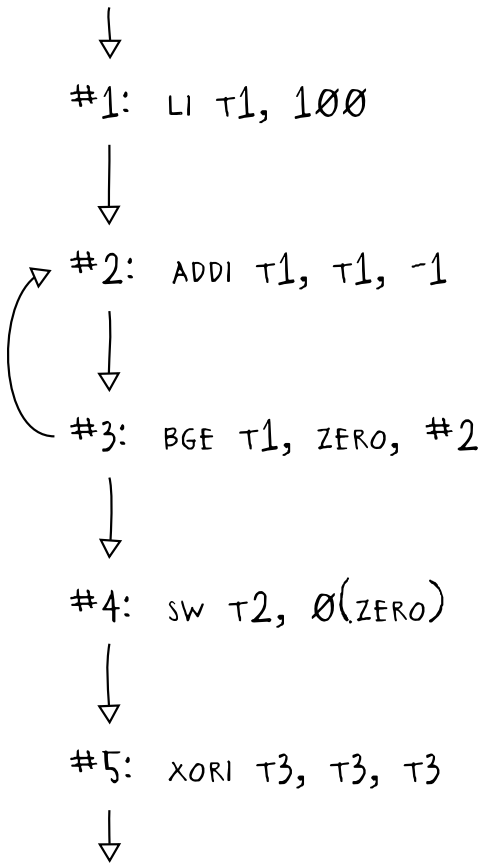


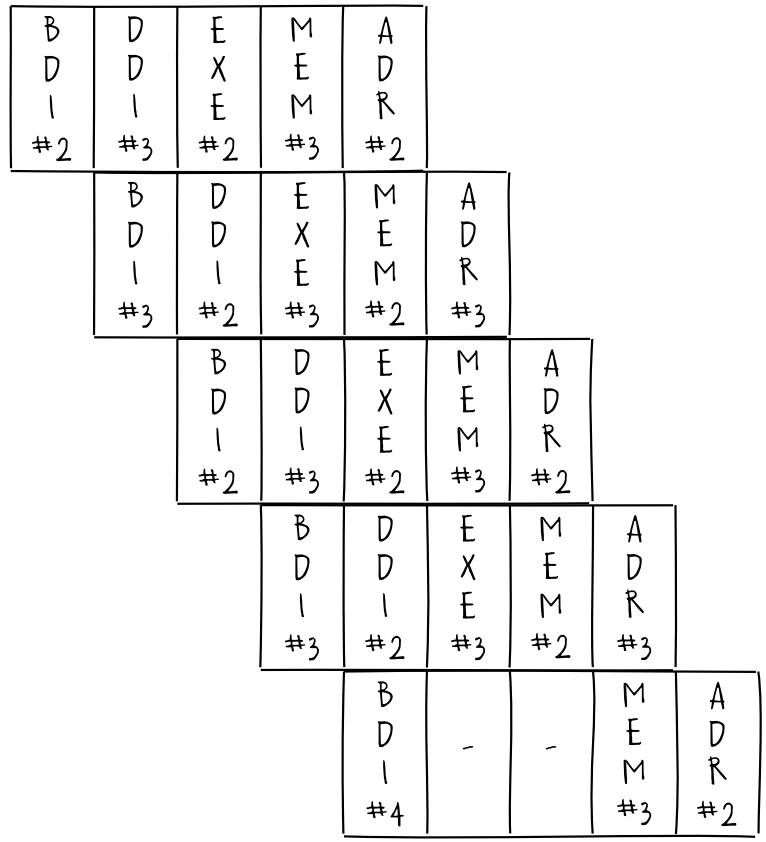


ITERAÇÃO 100 (PREDIÇÃO INCORRETA)

Conflito de controle

- Predição dinâmica de desvio





TAXA DE ACERTO DE 97%

Exercício

- Considerando um processador com *pipeline* de 5 estágios
  - Ilustre graficamente os estágios do *pipeline*
  - Execute o programa abaixo, detalhando como pode ser feito o tratamento dos diferentes conflitos
  - Calcule a melhoria de desempenho do projeto de *pipeline* com relação à versão multiciclo

```
1  # t1 = 0x80000000
2  addi t1, zero, 1
3  slli t1, t1, 31
4  # t2 = t1 + 1024
5  addi t2, t1, 1024
6  # Laço iterativo
7  loop:
8      # MEM[t1] = 0
9      sw zero, 0(t1)
10     # t1 = t1 + 4
11     addi t1, t1, 4
12     # t1 != t2 -> loop
13     bne t1, t2, loop
```