# Organização de Computadores I
# DCC006

## Aula 8 – O Processador – Pipeline

**Prof. Omar Paranaiba Vilela Neto**

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

# Introdução

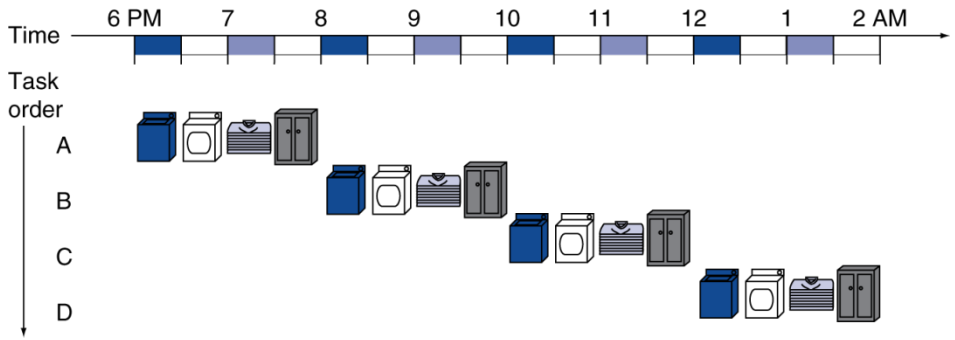
- Fatores de performance da CPU
  - Número de Instruções
    - Determinado pelo ISA e compilador
  - CPI e tempo de Ciclo
    - Determinado pela CPU hardware
- Nós vamos examinar duas implementações RISC-V + Um MIPS extra
  - Uma versão simplificada
  - Uma versão mais realista - Pipeline
- Subconjunto simples, mostra muitos aspectos
  - Referência à Memória: `ld`, `sd`
  - Aritmética/lógica: `add`, `sub`, `and`, `or`
  - Transferência de Controle: `beq`

# Questões de Desempenho

- O maior atraso determina o período de clock
  - Caminho crítico: Instrução load
  - Instrução de memória → banco de registradores → ALU → memória de dados → banco de registradores
- Não é factível variar o período para diferentes instruções
- Viola princípio de design
  - Fazer o caso mais comum mais rápido

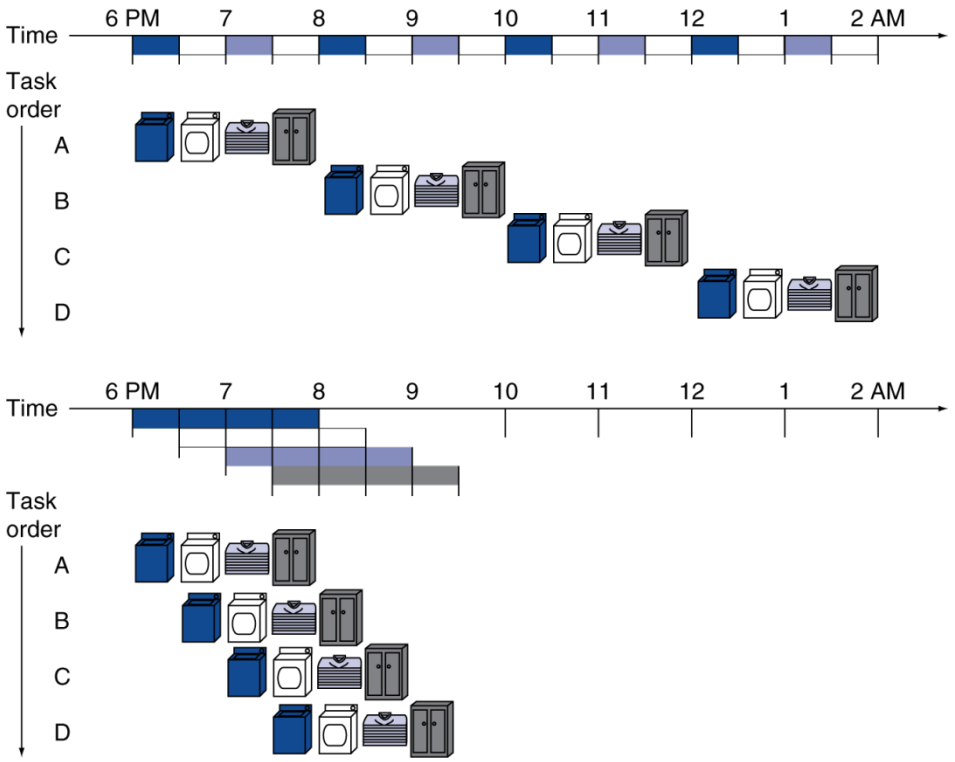
# Pipelining Analogia

- Lavanderia sequencial:



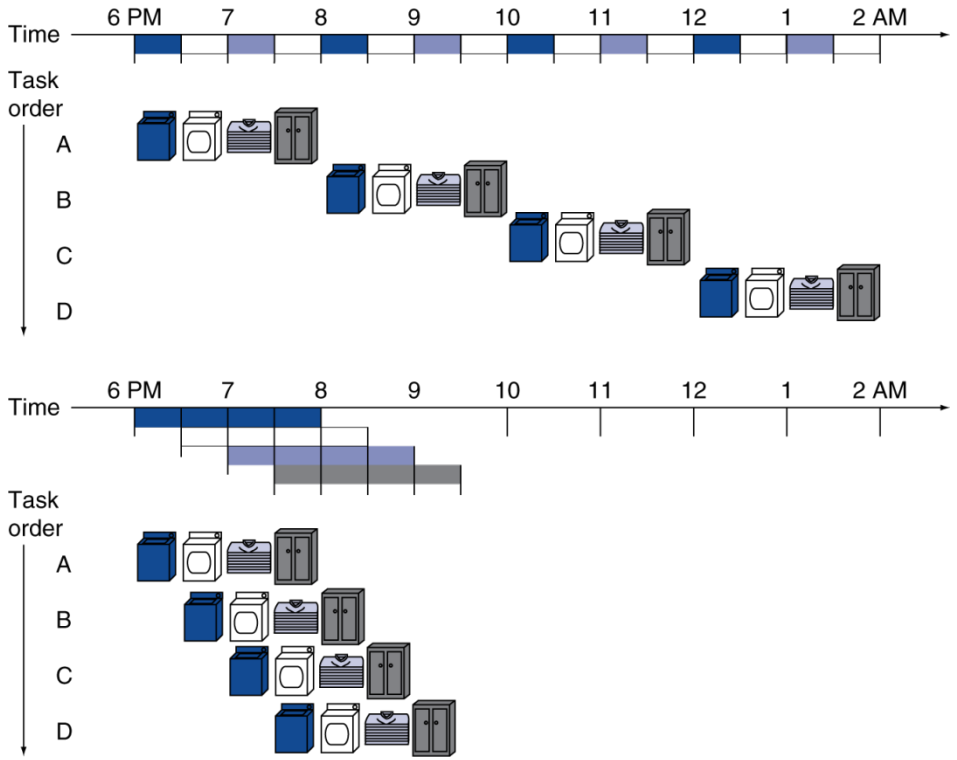
# Pipelining Analogia

- Lavanderia Pipelined:
  - Paralelismo melhor desempenho



# Pipelining Analogia

- Lavanderia Pipelined:
  - Paralelismo melhor desempenho




- Quatro cargas:
  - Aceleração = 8/3,5 = 2,3
- Cargas infinitas:
  - Aceleração = 2n/(0,5n + 1,5) ≈ 4 = número de estágios

# RISC-V Pipeline

- Cinco estágios, um passo por estágio

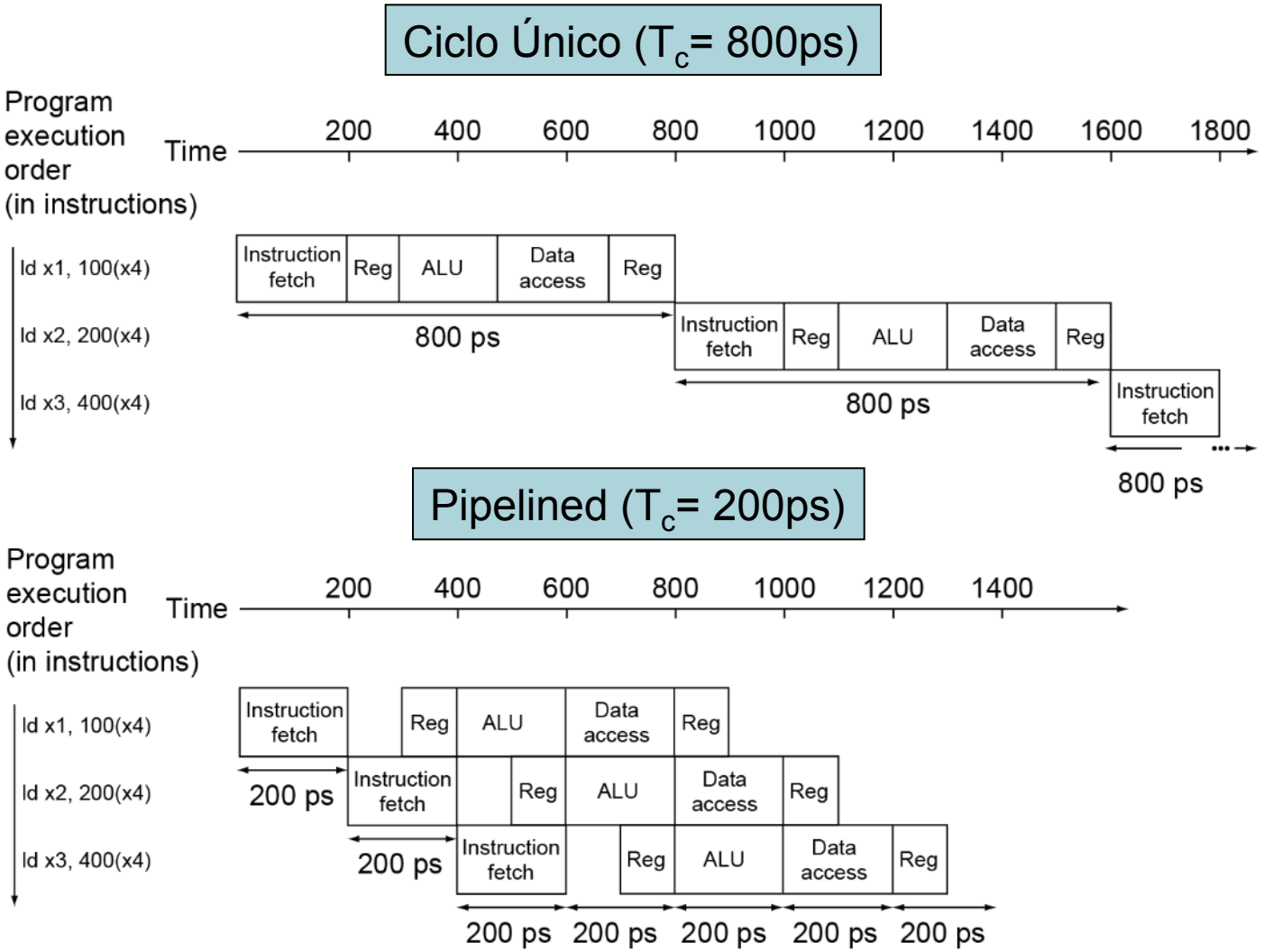
1. IF: Instruction fetch from memory
2. ID: Instruction decode & register read
3. EX: Execute operation or calculate address
4. MEM: Access memory operand
5. WB: Write result back to register

# Pipeline Desempenho

- Assuma que os tempos dos estágios são:
  - 100ps para ler e escrever registradores
  - 200ps para outros estágios
- Comparar o datapath pipeline com o datapath ciclo único

| Instr | Instr fetch | Register read | ALU op | Memory access | Register write | Total time |
|---|---|---|---|---|---|---|
| ld | 200ps | 100 ps | 200ps | 200ps | 100 ps | 800ps |
| sd | 200ps | 100 ps | 200ps | 200ps | | 700ps |
| R-format | 200ps | 100 ps | 200ps | | 100 ps | 600ps |
| beq | 200ps | 100 ps | 200ps | | | 500ps |

# Pipeline Desempenho



# Aceleração do Pipeline

- Se todos os estágios são balanceados
  - i.e., todos levam o mesmo tempo
  - $\text{Tempo\_Instruções}_{\text{pipelined}} = \frac{\text{Tempo\_Instruções}_{\text{nonpipelined}}}{\text{Número de estágios}}$
- Se não balanceado, aceleração menor
- Aceleração devido aumento da VAZÃO
  - Latência (tempo de cada instrução) não diminui

# Pipelining & ISA Design

- RISC-V ISA designed para pipelining
  - Todas instruções de 32-bits
    - Fácil de buscar e decodificar em um ciclo
    - ex. x86: instruções de 1- a 17-byte
  - Poucos formatos de instrução e regulares
    - Fácil de buscar registradores e decodificar em um ciclo
  - Endereçamento Load/store
    - Calcula endereço no 3º. estágio e acessa memória no 4º. estágio.


Problems

# Hazards (Perigos)

- Situações que não permitem começar a próxima instrução no próximo ciclo
- Hazards Estrutural
  - Um recurso está ocupado
- Hazard de Dado
  - Precisa esperar que instrução anterior complete a leitura/escrita de dado
- Hazard de Controle
  - Decisão de ação de controle depende de instrução anterior

# Hazards Estrutural

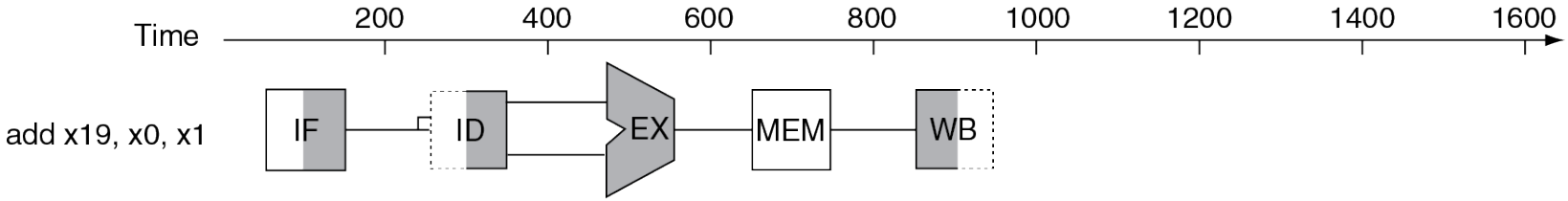
- Conflito no uso de um recurso
- No RISC-V pipeline com uma memória
  - Load/store requer acesso a dado
  - Instruction fetch tem que *parar* naquele ciclo
    - Pode causar uma "bolha" no pipeline
- Logo, pipelined datapaths deve separar as memórias de dados e instruções

# Hazards de Dados

- Uma instrução depende da finalização do dado de uma instrução anterior
  - `add  x19, x0, x1`
    `sub  x2, x19, x3`

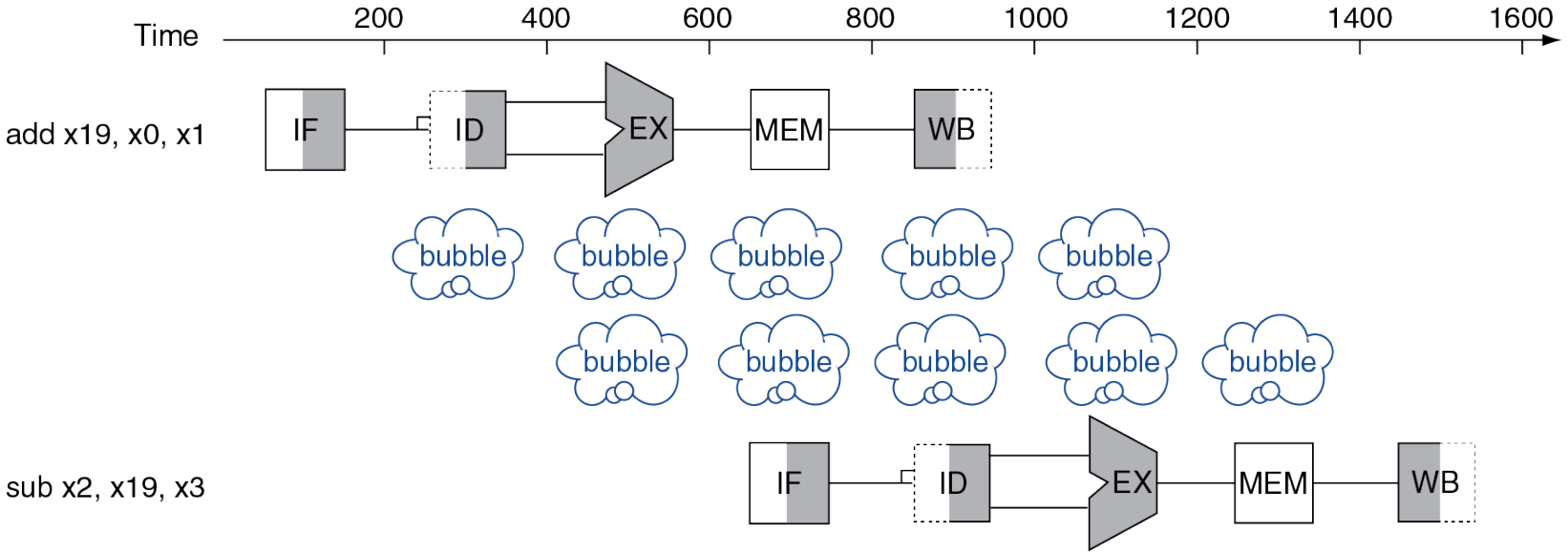
# Hazards de Dados

- Uma instrução depende da finalização do dado de uma instrução anterior
  - `add  x19, x0, x1`
    `sub  x2, x19, x3`



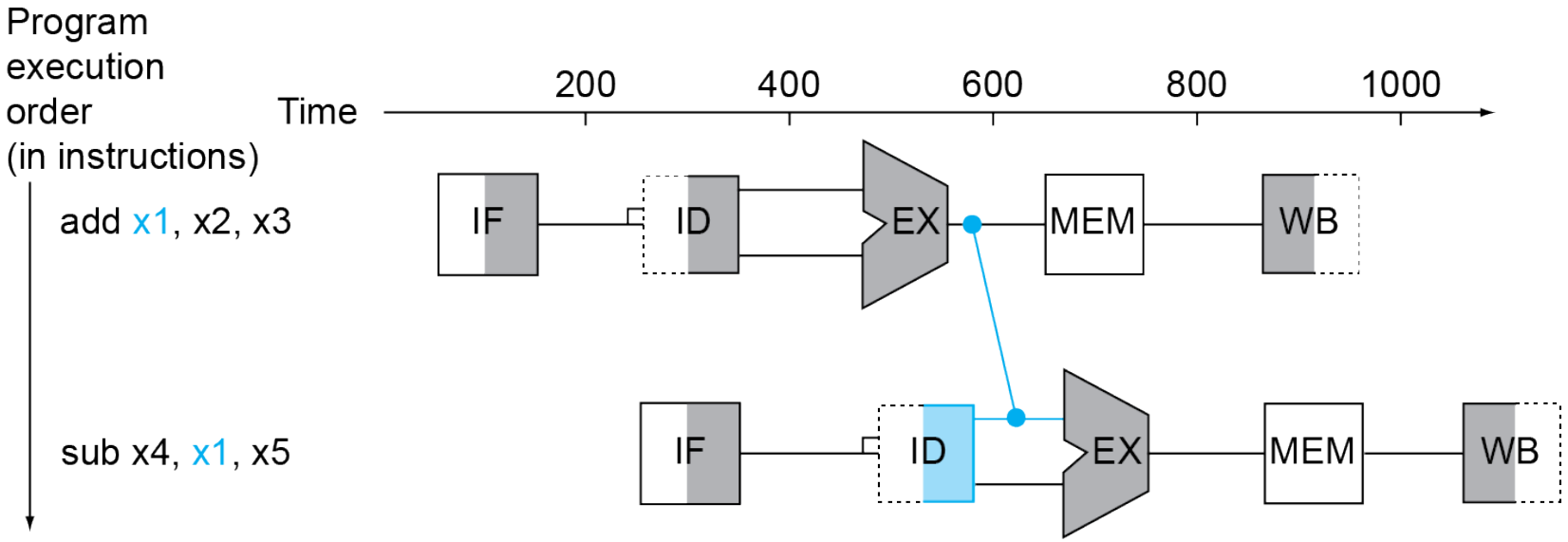
# Encaminhamento(Forwarding)

- Use o resultado tão logo calculado
  - Não espere guardar no registrador
  - Requer novas conexões no datapath



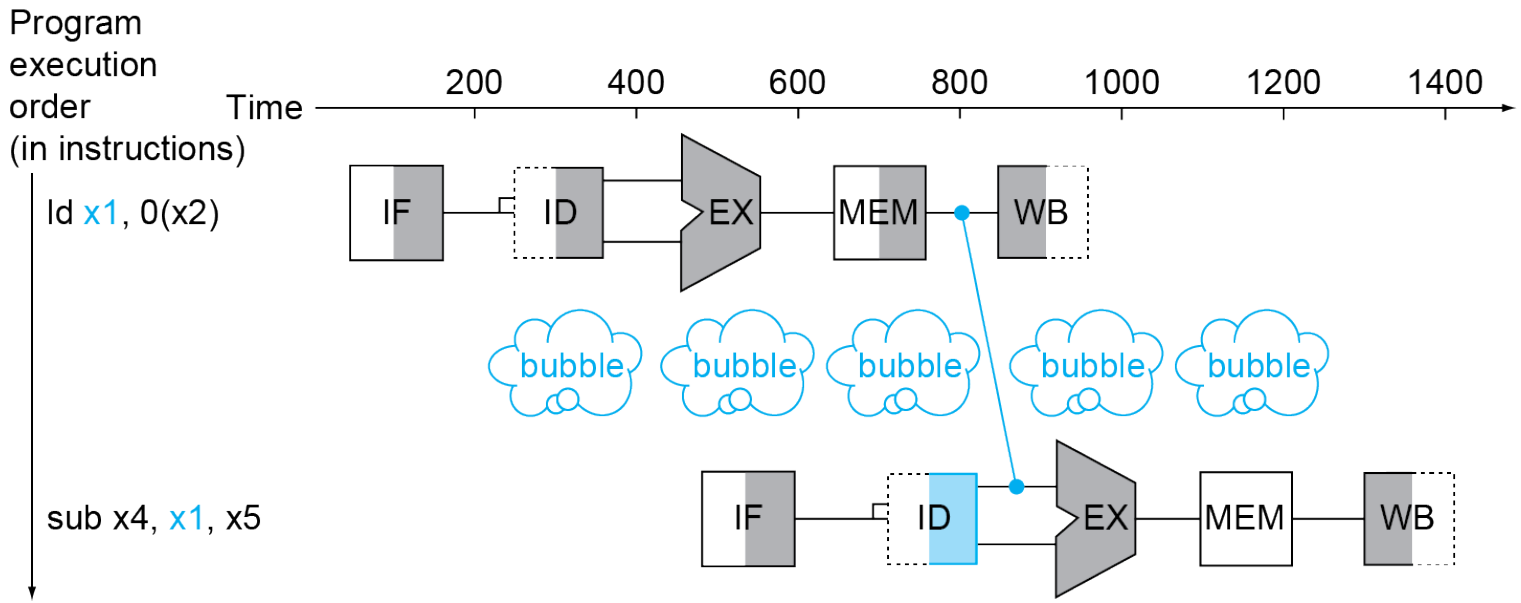
# Encaminhamento(Forwarding)

- Use o resultado tão logo calculado
  - Não espere guardar no registrador
  - Requer novas conexões no datapath



# Hazards de Dados - Load

- Nem sempre dá para evitar *stall* por encaminhamento
  - Quando o valor não é computado quando necessário

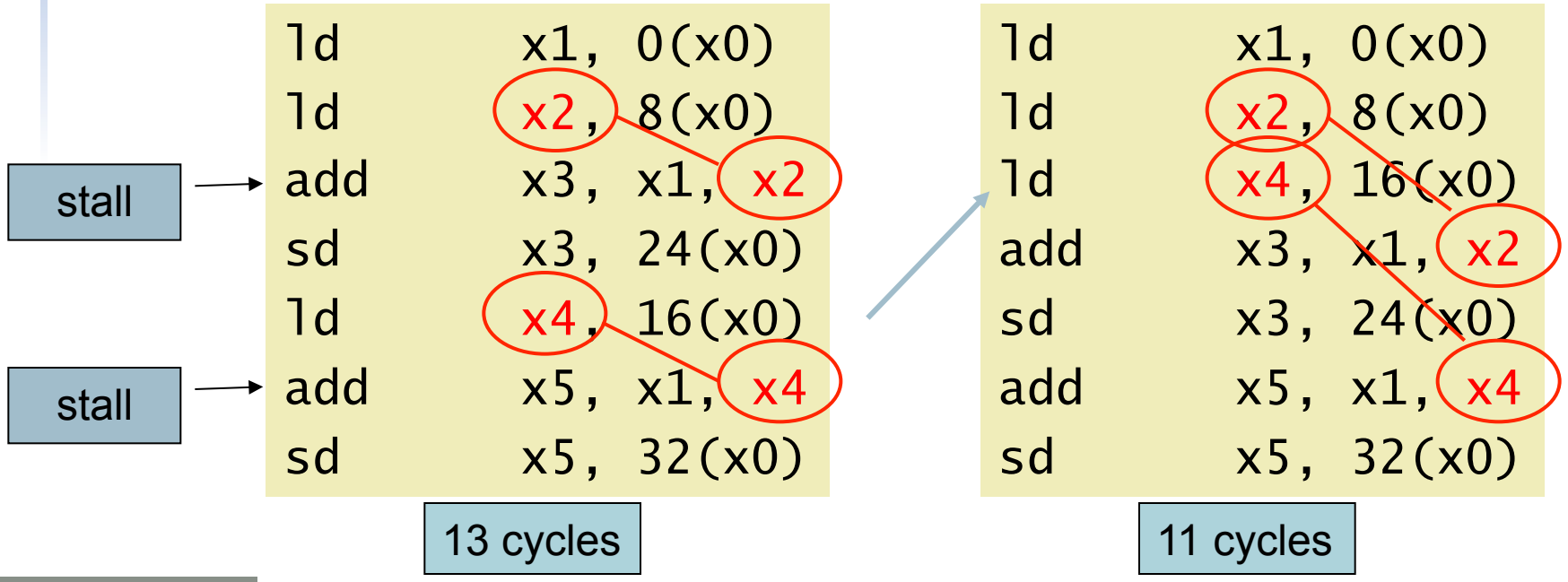


# Reescalonamento de Código evita *Stalls*

- Reordene o código para evitar uso do resultado do load na próxima instrução
- Código para `a = b + e; c = b + f;`

# Reescalonamento de Código evita *Stalls*

- Reordene o código para evitar uso do resultado do load na próxima instrução
- Código para `a = b + e; c = b + f;`



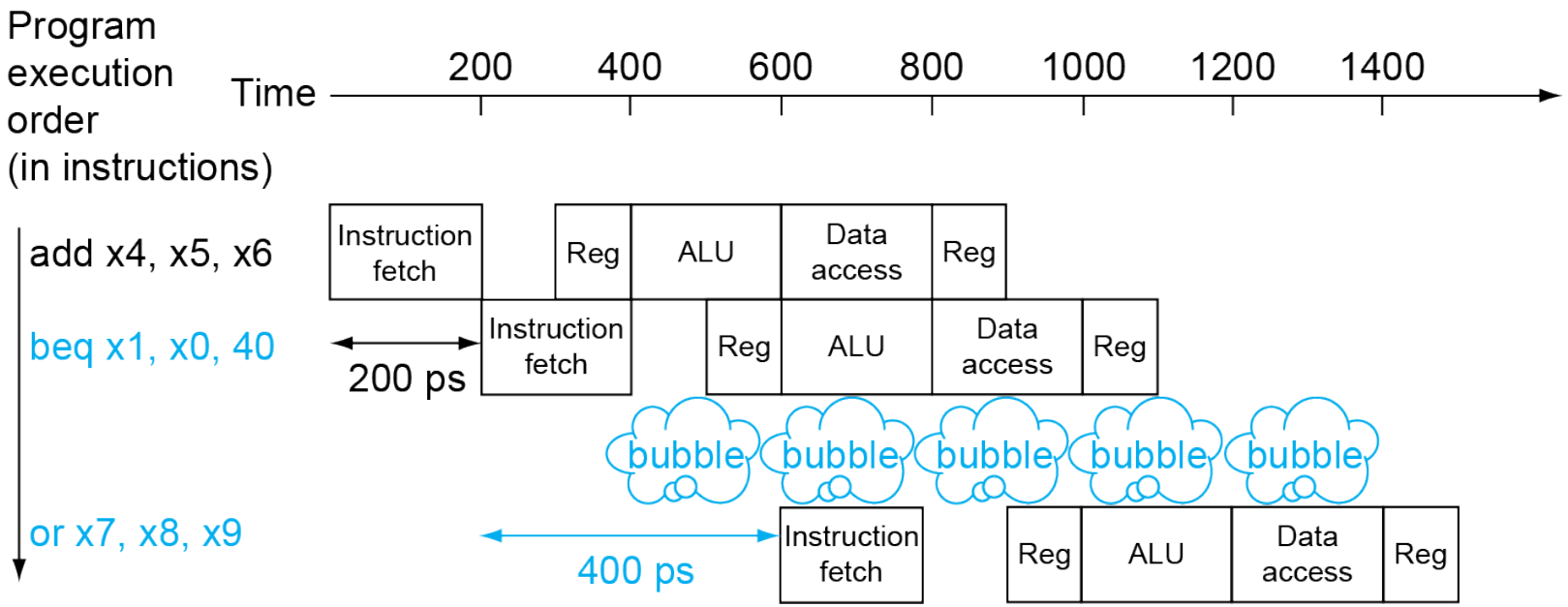
# Hazards de Controle

- Branch determina o fluxo de controle
  - Busca de nova instrução depende de decisão do branch
  - Pipeline não pode sempre buscar correto
    - Continua funcionando no estágio ID do branch

- No RISC-V pipeline
  - Necessita comparar registradores e computar destino cedo no pipeline
  - Adiciona hardware para fazer isso no estágio ID

# Stall no Branch

- Espere até que a saída do branch seja determinada para buscar instrução



# Previsão de Branch

- Pipelines mais longos não podem determinar resultado cedo
  - Penalidade de *Stall* começa a ser inaceitável
- Prever o resultado do branch
  - *Stall* só se prever errado

- No RISC-V pipeline
  - Pode prever sempre não tomado
  - Busca instrução depois do branch, sem atraso

# Previsões mais realistas

- Previsão estática
  - Baseado no comportamento típico
  - Examplo: loop e if branches
    - Prevê tomado quando para trás
    - Prevê não tomado para frente
- Previsão dinâmica
  - Hardware calcula o que aconteceu
    - ex., Guarda a história recente do branch
  - Assume que o comportamento se mantém
    - Quando erra: stall e corrige histórico

# Resumo Pipeline

**The BIG Picture**

- Pipelining melhora o desempenho aumentando a vazão
  - Executa múltiplas instruções em paralelo
  - Cada instrução tem a mesma latência
- Sujeito a hazards
  - Estrutural, Dados e de Controle
- O design do conjunto de instruções afeta a complexidade de implementar o pipeline