Performance em Sistemas Ciberfísicos

Aula 4 – Pipelining

Frank Coelho de Alcantara – 2023 -1

Uma História Antiga

O IBM Stretch, também conhecido como IBM 7030, introduzido em 1961. Foi a primeira arquitetura e a implementar os conceitos de Pipelining.

O objetivo era ganhar tempo de processamento, tornando o sistema mais eficiente.

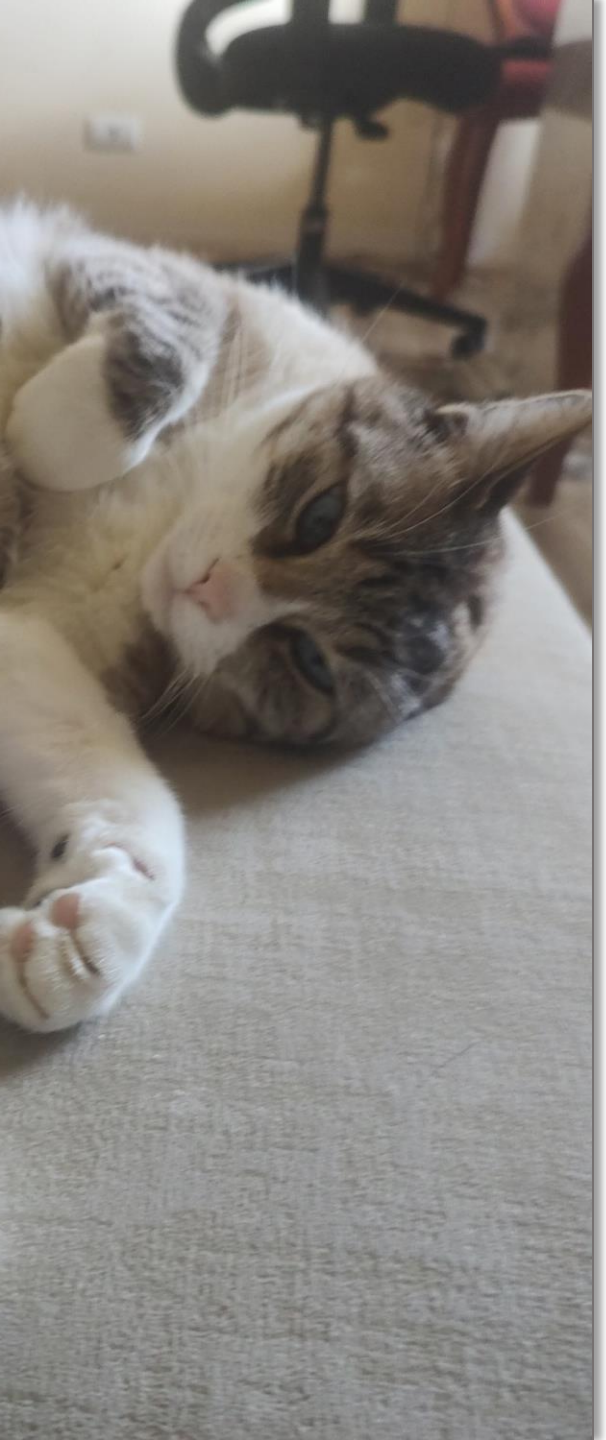
Uma História Antiga

Na arquitetura CDC 6600, projetada por Seymour Cray e lançada em 1964, o pipelining realmente ganhou destaque. Foi o supercomputador mais rápido de sua época e fez uso extensivo de pipelining para obter um desempenho extraordinário para a época!

Uma História Antiga

A metáfora "pipelining" foi introduzida Michael J. Flynn em "*Computer Architecture: Pipelined and Parallel Processor Design*“ de 1966.

Desde então, tem sido utilizada e aprimorada. Hoje não se imagina um unidade de processamento sem pipeline.

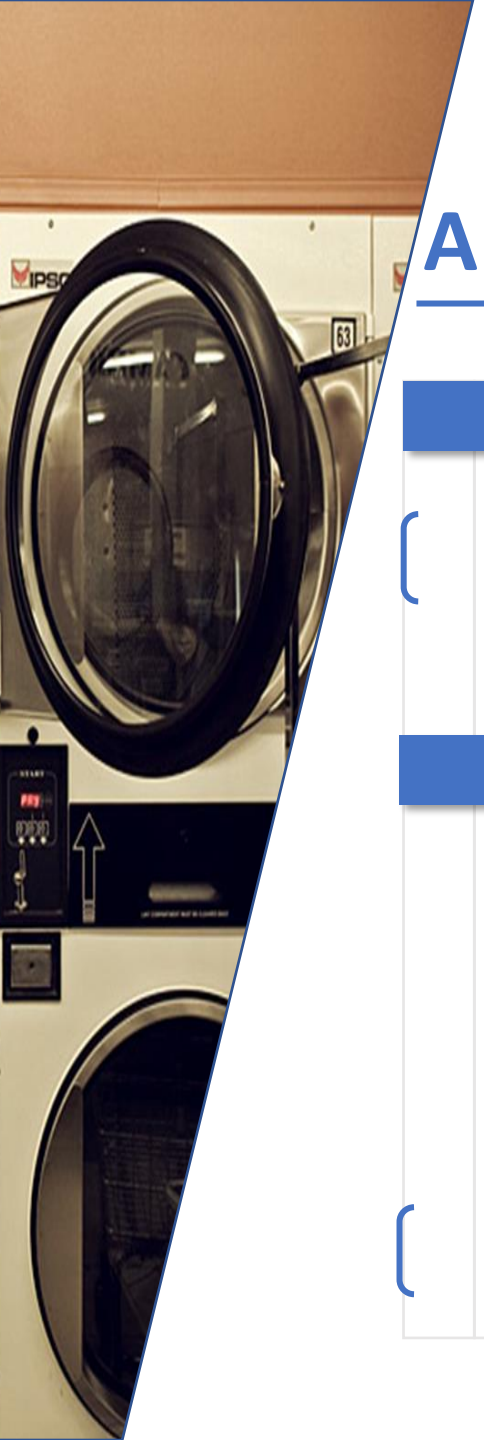


Exercício 1 – Para Entender o Problema

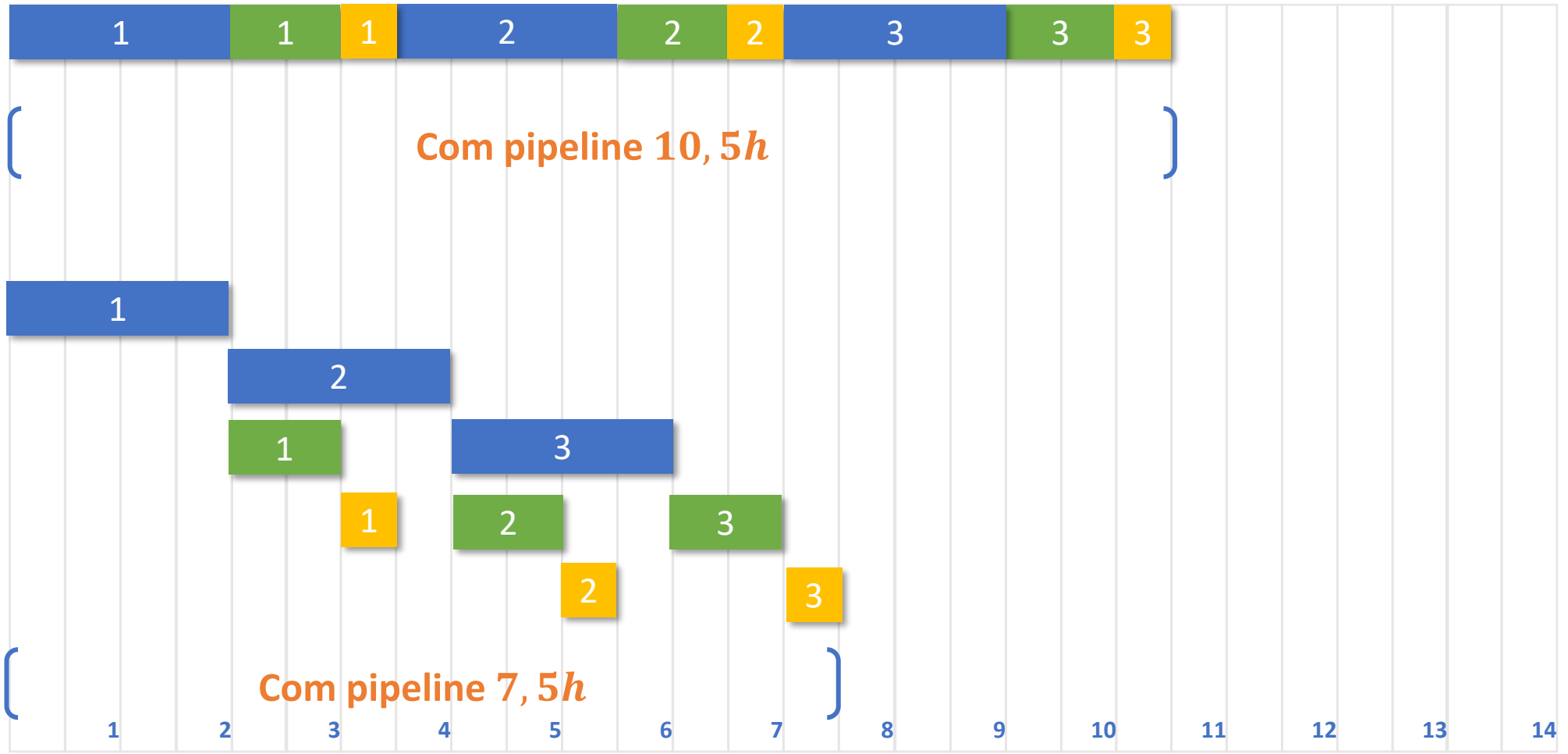
"Errar é humano. Mas para fazer algo realmente estúpido você precisa de um computador."
Paul R. Ehrlich

A Inevitável lavanderia

Você dispõe de um conjunto com três máquinas: uma de lavar, uma de secar e uma de passar, todas as máquinas permitem a mesma carga. Sabendo que você leva 2h para lavar, 1h para secar e 0,5h para passar. Quanto tempo levará para lavar, passar e secar 3 cargas de roupas sem pipeline? Com pipeline?



A Inevitável lavanderia



Lições Aprendidas

- O Pipelining não afeta a *latência* de uma tarefa específica. Ajuda na redução do tempo de trabalho (*throughput*) de toda carga.
- O menor tempo de carga será limitado pelo estágio mais lento.
- Teremos múltiplas tarefas sendo executadas ao mesmo tempo
- Sempre é possível acelerar o processo incluindo estágios extras.
- O tempo necessário para começar (*fill*) e terminar (*drain*) reduz a velocidade.

Eu falei *Throughput*?

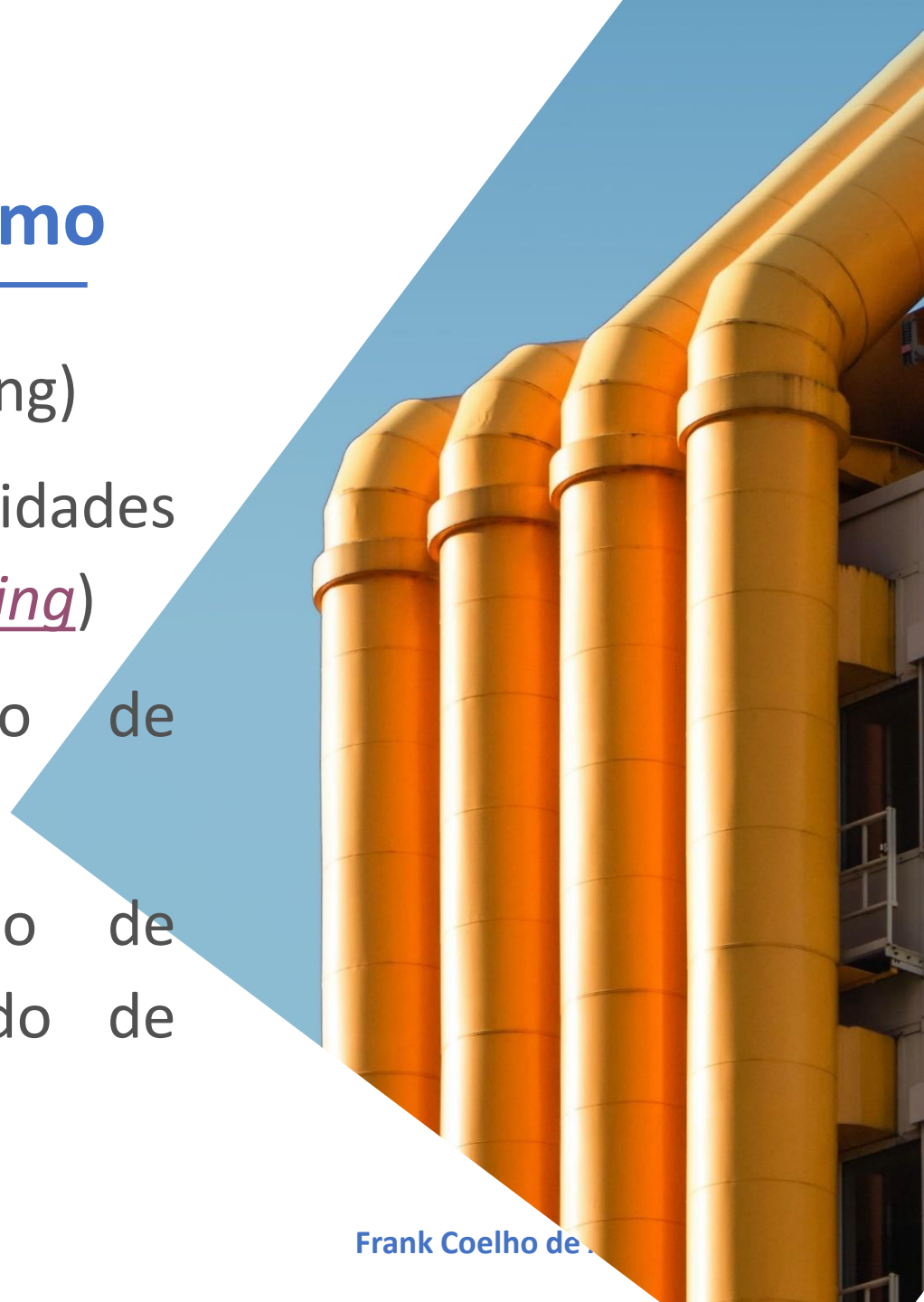
- Latência (tempo): tempo gasto para terminar uma tarefa.
- *Throughput* (*bandwidth*): número de tarefas realizadas em um dado intervalo de tempo.
- Exemplo: pessoas entre lugares distantes 10Km
  - Carro: capacidade = 5, velocidade = 60 km/h
  - Ônibus: capacidade = 60, velocidade = 20 km/h
  - Latência: carro = 10 min, ônibus = 30 min
  - *Throughput*: carro = 15 PPH (contar a volta), ônibus = 60 PPH

Na Vida Real - TPS

O TPS – *Toyota Productions System.* É uma metodologia de processos e controle desenvolvida pela Toyota para otimizar o processo de fabricação de veículos. Entre outras coisas, este sistema incluiu o conceito de pipelining para atingir os objetivos do *just-in-time.*

Pipelining é uma forma de Paralelismo

- **ILP** – *instruction level parallelism* (pipelining)
- **DLP** – *data level parallelism* (unidades aritméticas usando pipelining, *loop unrolling*)
- **TLP** – *thread-level processing* (uso de processadores/cores múltiplos)
- **RLP** – *request-level processing* (uso de processadores/cores múltiplos acrescido de apoio do Sistema Operacional)



Pipeline – Didática em 4 Fases

1. Busca de instruções (Instruction Fetch, IF)
2. Decodificação de instruções (Instruction Decode, ID)
3. Execução (Execution, EX)
4. Armazenamento de resultados (Write Back, WB)

Pipeline – Didática Primeira Fase

Busca de instruções (Instruction Fetch, IF): Nessa etapa, a CPU busca a instrução a ser executada da memória. O endereço da instrução é armazenado no contador de programa (Program Counter, PC), que é incrementado após cada busca para apontar para a próxima instrução.

Pipeline – Didática Segunda Fase

Decodificação de instruções (Instruction Decode, ID): Aqui, a instrução buscada é decodificada para determinar a operação a ser realizada e identificar os operandos envolvidos. Além disso, os registradores de origem são lidos e, se necessário, os valores são buscados da memória.

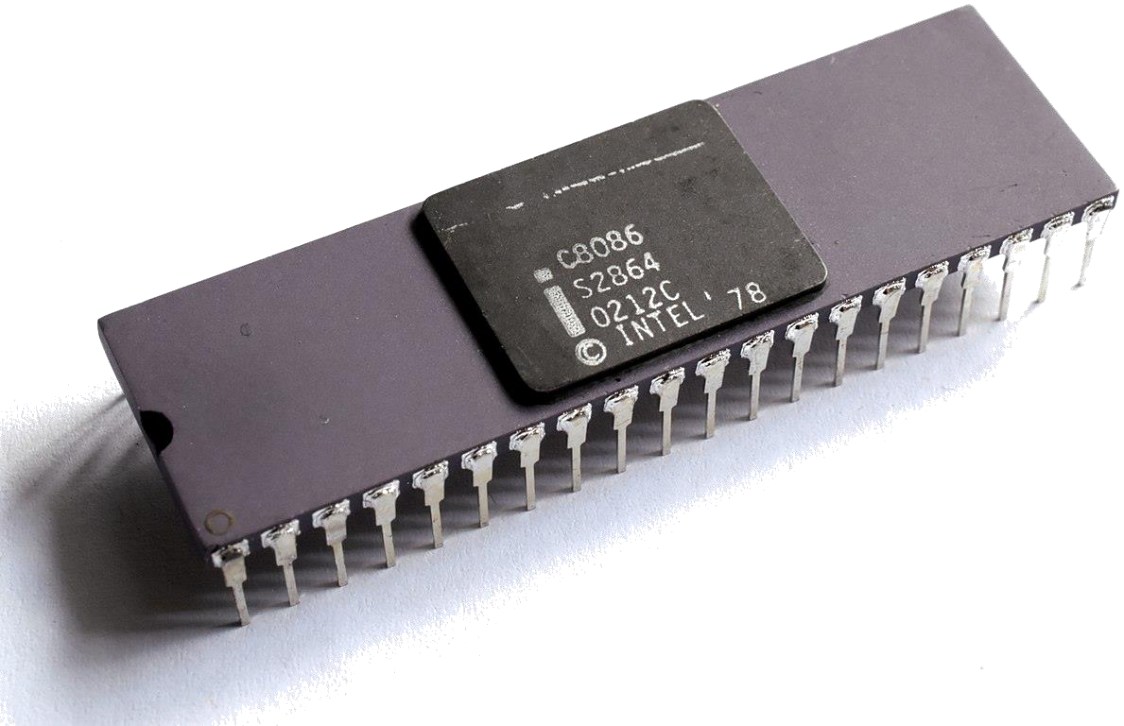
Pipeline – Didática Terceira Fase

Execução (Execution, EX): Nessa etapa, a operação especificada pela instrução é realizada. Isso pode incluir operações aritméticas, lógicas ou de deslocamento, entre outras. A Unidade de Execução (Execution Unit, EU) é responsável por executar a operação.

Pipeline – Didática Terceira Fase

Armazenamento de resultados (Write Back, WB): Por fim, os resultados da operação executada são armazenados nos registradores de destino ou na memória, conforme necessário.

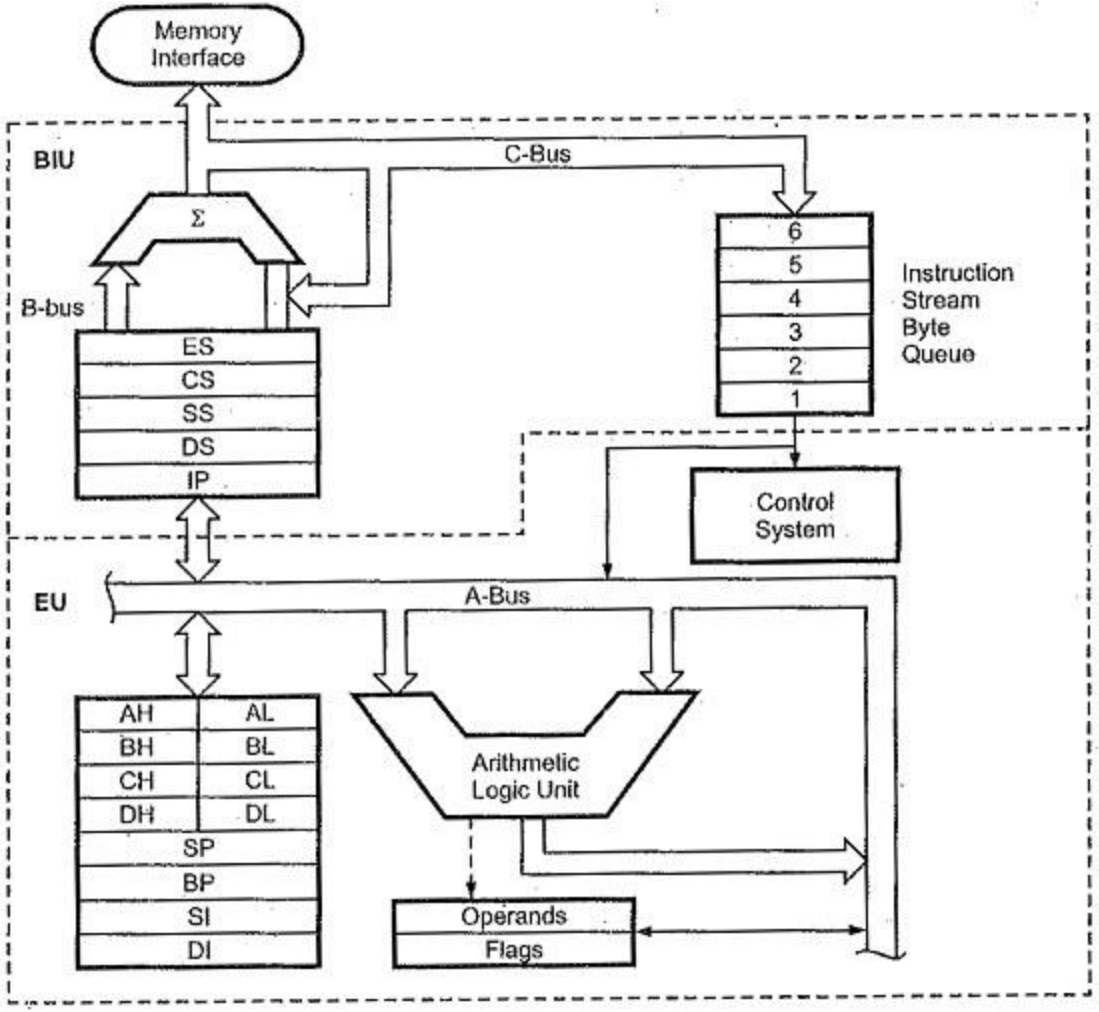
Intel 8086 – Um chip pioneiro



Um dos primeiros dispositivos a usar alguma coisa parecida com pipelining (1978)

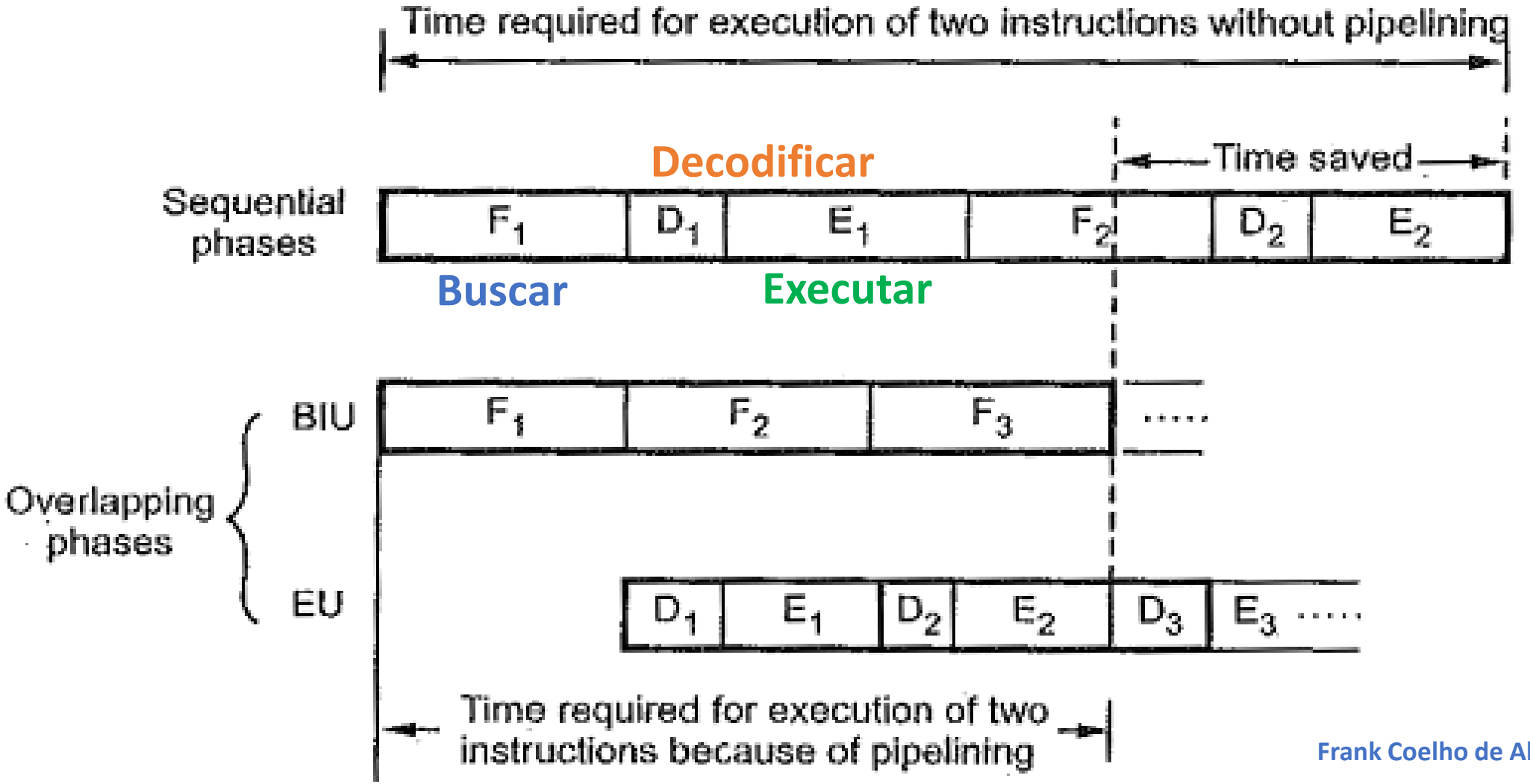
1. Bus Interface Unit;
2. Execution Unit.

Intel 8086 – Dividido em Duas Fases de Processamento



- Determina o endereço da memória ou I/o.
- Recupera a instrução da memória.
- Lê e escreve dados na memória ou I/O.
- Gerência a Lista de instruções.

Intel 8086 – Onde está o paralelismo



A Evolução da Intel

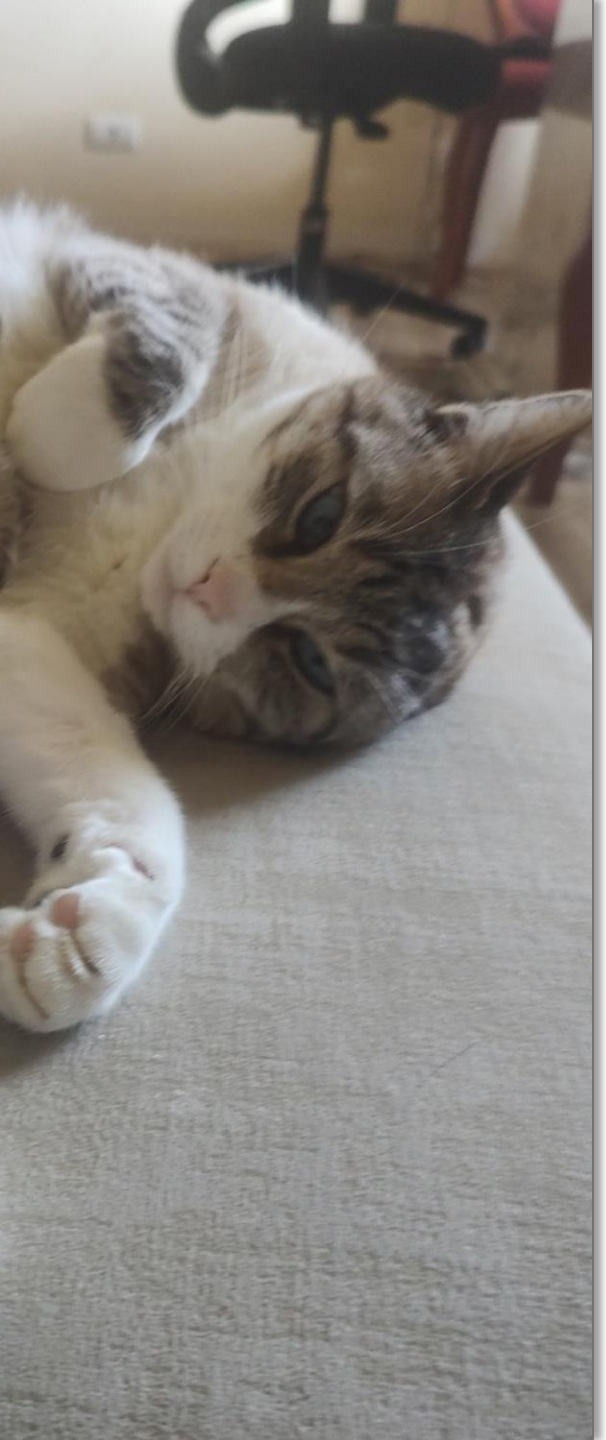
- Pentium Pro (em 1995): pipeline de 5 estágios.
- Pentium II (em 1997): superescalares, chegando a mais de 10 estágios de pipeline.
- Intel Core (1ª geração, em 2006) adotaram pipeline de 14-16 estágios, continuando até a 10ª geração (lançada em 2017).
- 11ª geração Intel Core (Tiger Lake, em 2020) adotou abordagem de pipeline híbrido, com mistura de pipeline de profundidade fixa e variável.

A Evolução da AMD

- AMD K5 (em 1996): arquitetura de pipeline de 5 estágios.
- AMD Athlon (em 1999): pipeline mais profunda, com 10 a 12 estágios de pipeline.
- AMD Athlon 64 (em 2003): pipeline mais longa, entre 15 e 17 estágios de pipeline.
- Essa abordagem continuou nas gerações subsequentes, como a série FX (lançada em 2011) e Ryzen (em 2017).

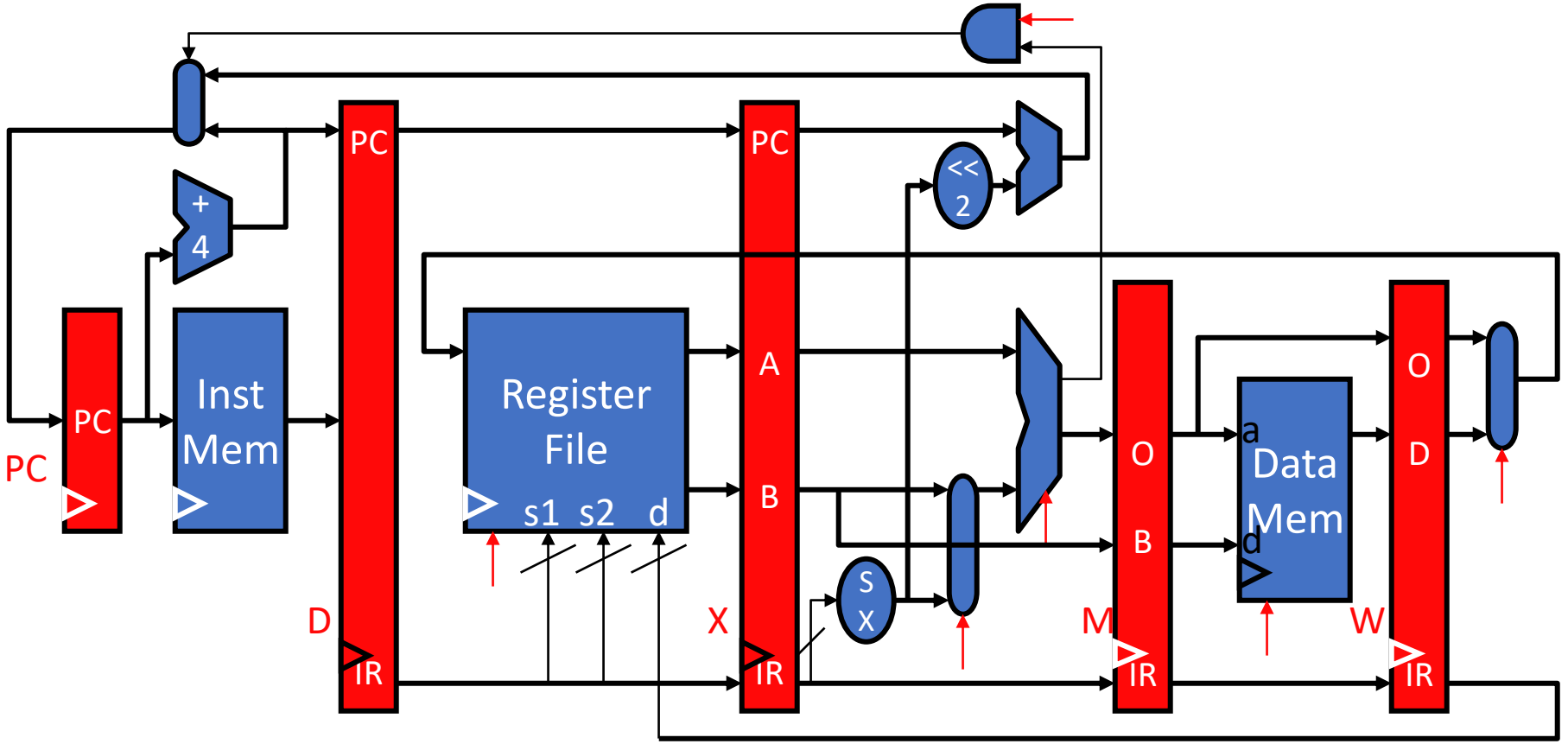
A Evolução da ARM

- A primeira geração, como o ARM1 (1985), não utilizava pipeline. ARM2 e ARM3 (1986 e 1989): pipeline de 3 estágios.
- ARM7TDMI (1994) e o ARM9 (1997): 5 estágios de pipeline foi aumentada para 5 estágios. ARM11 (2002) possui um pipeline de 8 estágios.
- A partir do Cortex-A8 (2005): pipeline da ARM tornou-se variando entre 13 e 16 estágios, dependendo da implementação. Cortex-A72 (2015) e o Cortex-A77 (2019) chegam a 20 estágios.

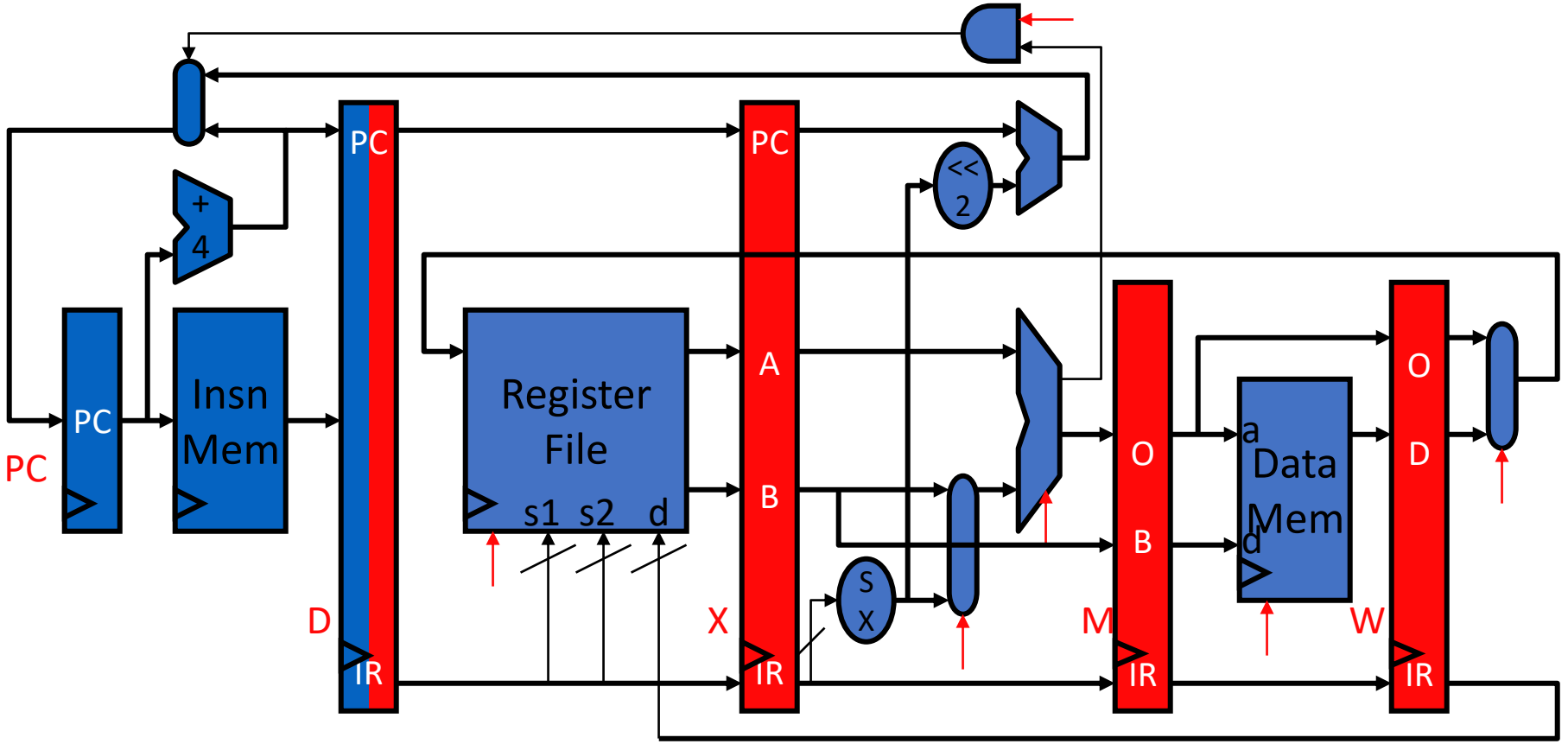


Visualizando em 5 estágios

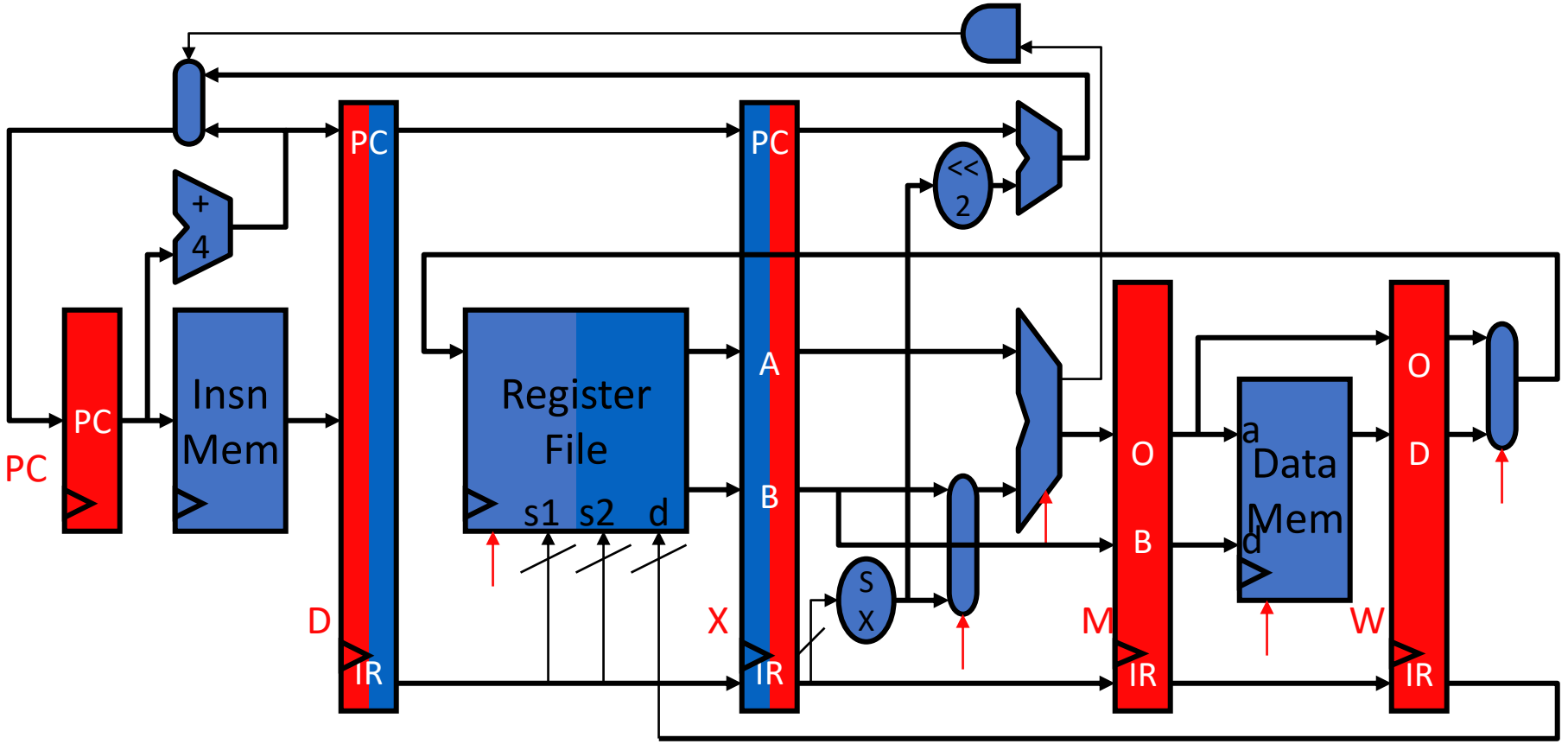
Um pipeline didático, levemente parecido com o MIPS



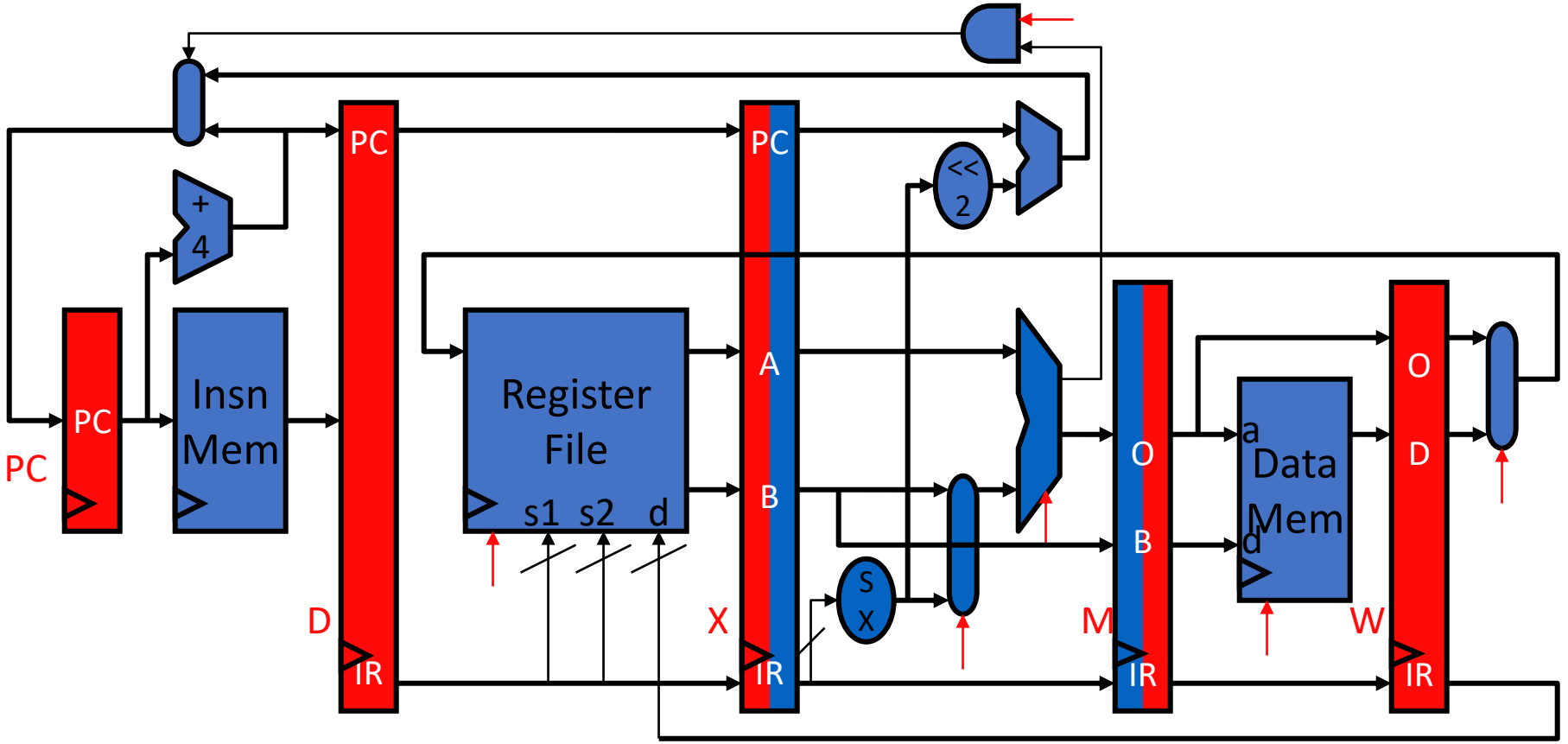
Frank Coelho de Alcantara – 2023-1



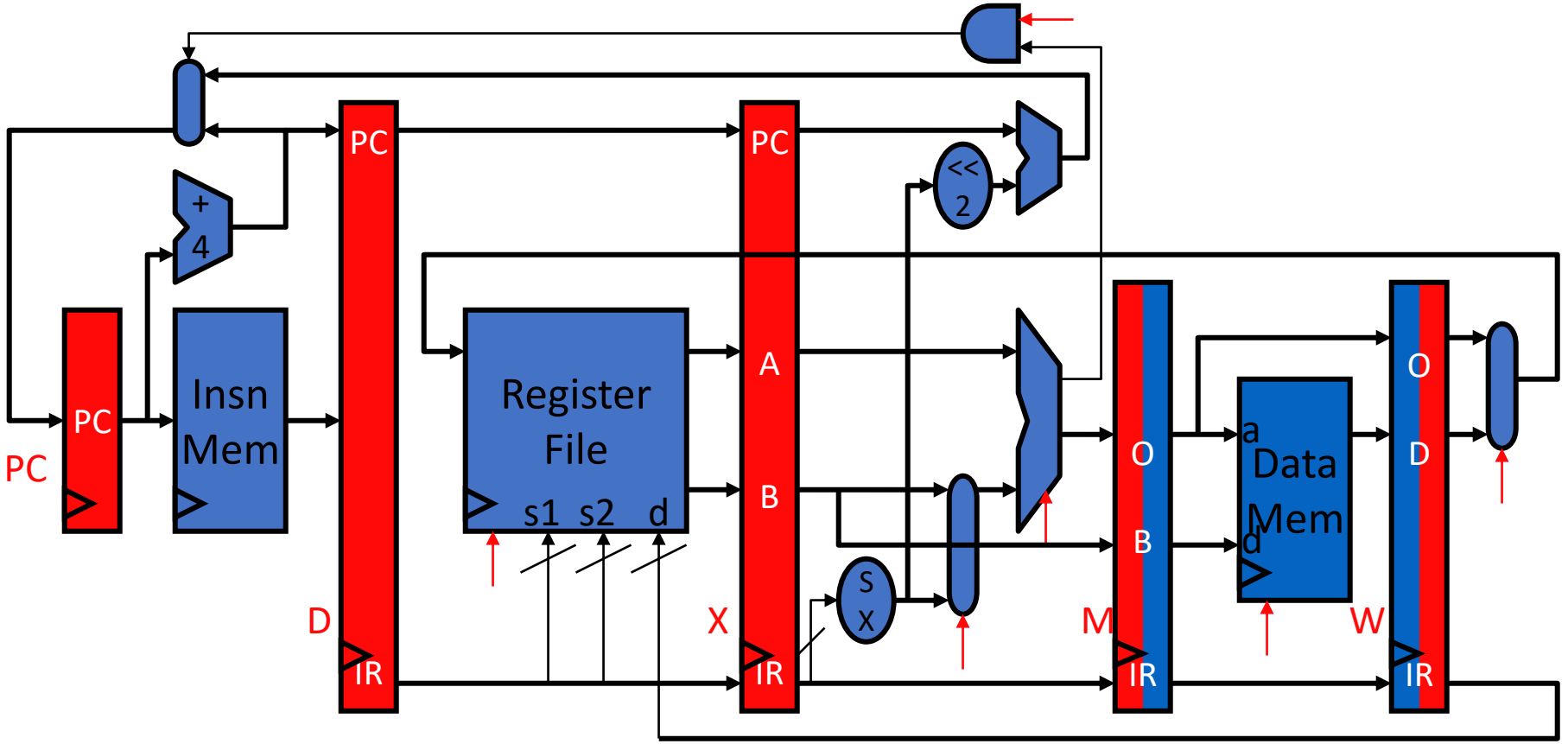
Fetch (Busca de Instrução): o processador busca a próxima instrução a ser executada na memória principal (RAM) e a coloca no cache de instruções (uma memória especializada para armazenar as instruções a serem executadas). O processador também incrementa o valor do contador de programa (*program counter*) para apontar para a próxima instrução na memória.



Decode (Decodificação de Instrução): o processador decodifica a instrução buscada no estágio anterior para determinar qual operação deve ser executada e quais operandos serão usados. O processador também identifica quais registradores serão usados na instrução e habilita as funções necessárias para a execução da operação.

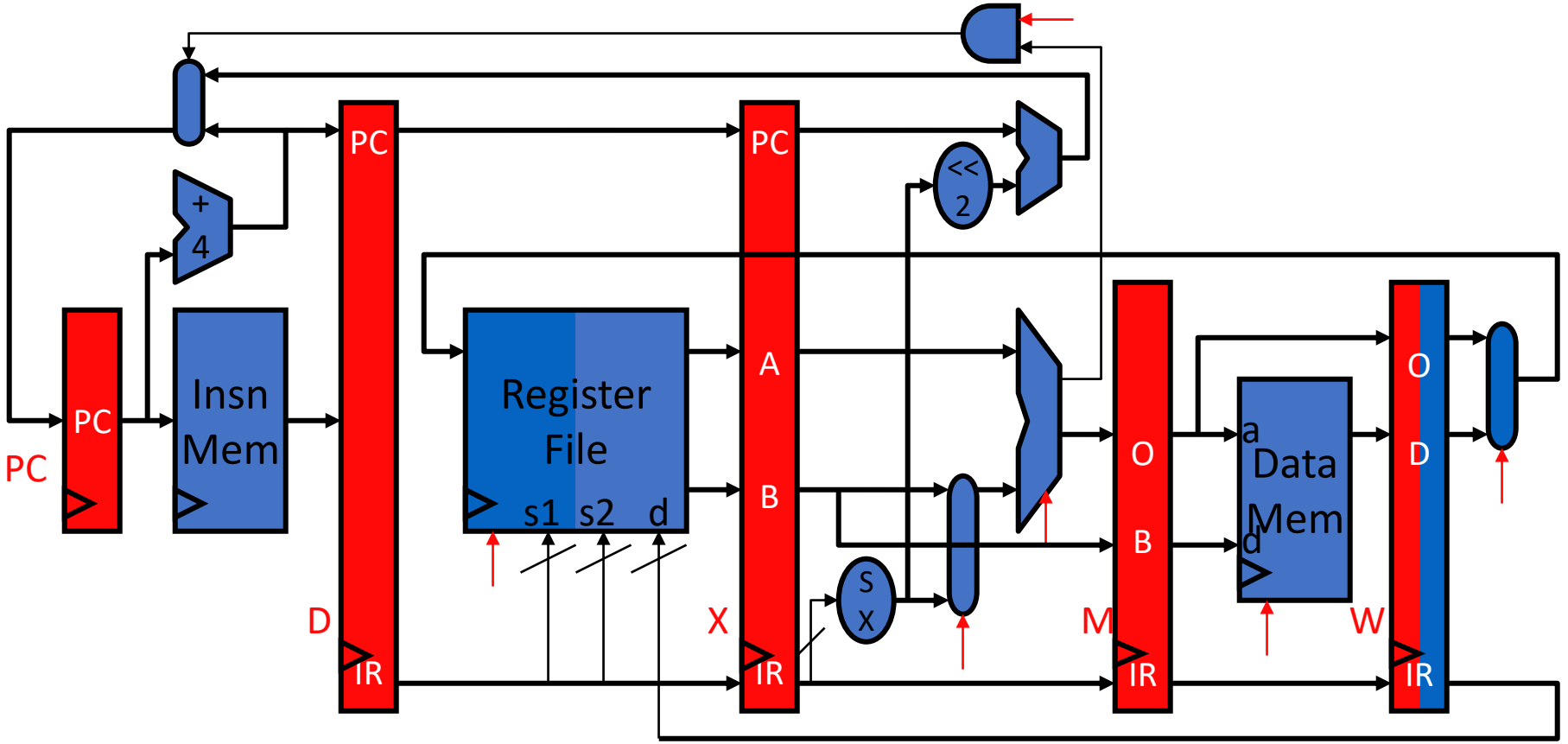


Execute (Execução): o processador executa a operação especificada pela instrução. Os operandos necessários para a operação são recolhidos dos registradores ou da memória, conforme seja necessário para a instrução. A operação é realizada e o resultado é armazenado temporariamente em um registrador interno.

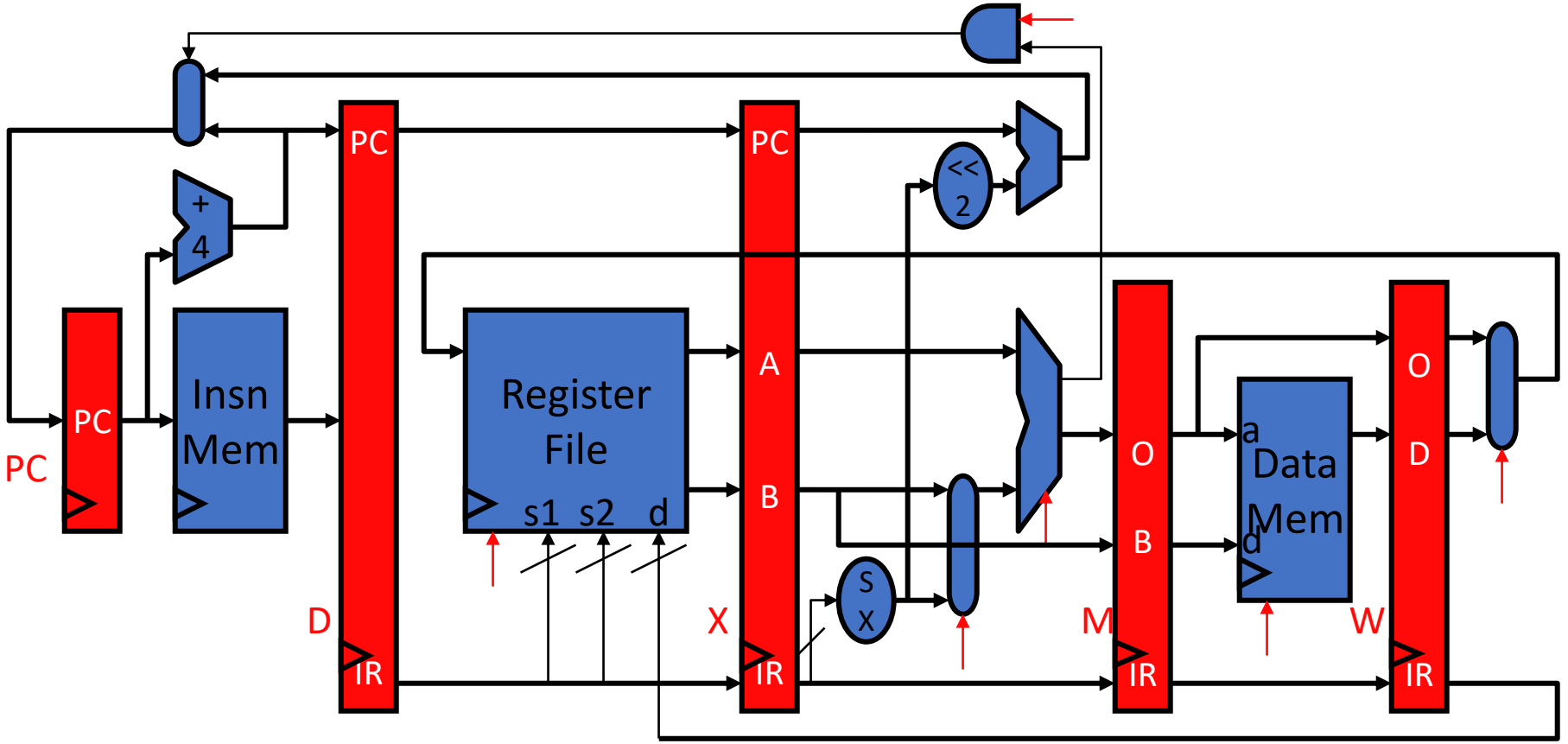


Memory (Acesso à Memória): o processador acessa a memória principal para buscar ou armazenar dados. Isso é necessário para instruções que envolvem acesso à memória, como leitura ou gravação de dados em um endereço de memória específico. Se a instrução não envolver acesso à memória, este estágio será ignorado.

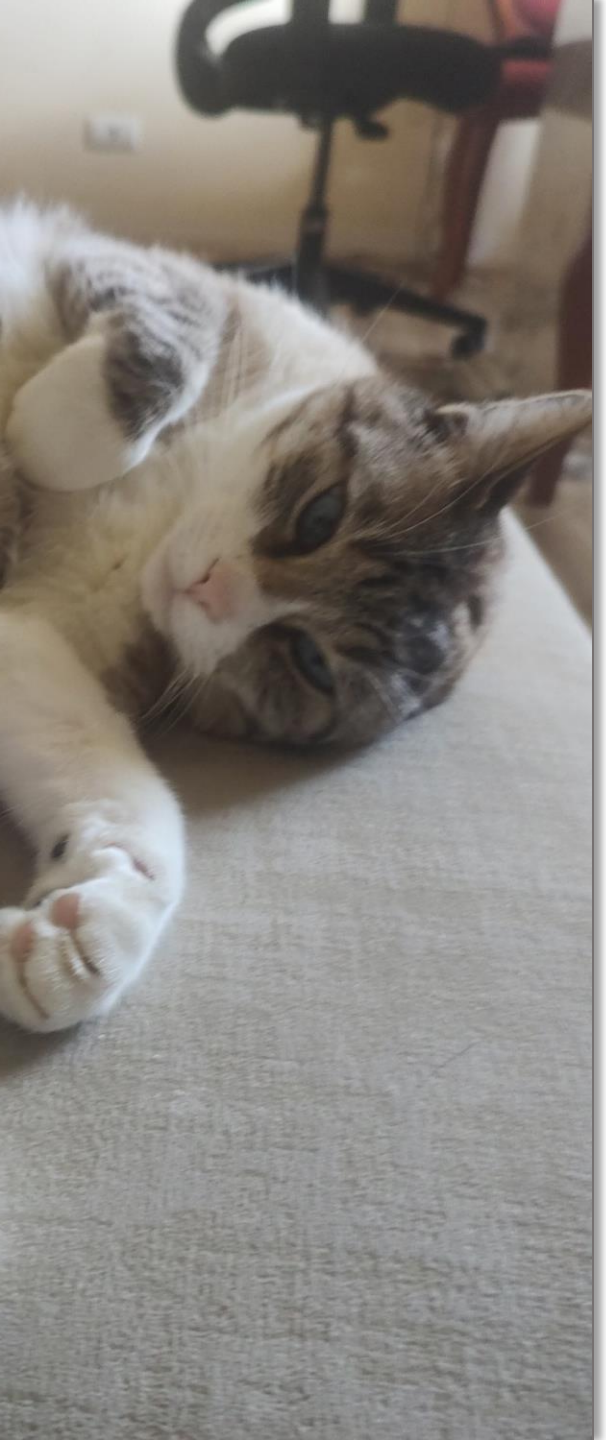
Frank Coelho de Alcantara – 2023-1



Writeback (Escrita de volta): o processador escreve o resultado da operação no registrador final ou na memória, conforme a instrução. Se a instrução envolver uma operação de leitura da memória, o resultado será armazenado no registrador interno. Se a instrução envolver uma operação de gravação na memória, o resultado será escrito na memória no endereço especificado pela instrução. Se a instrução envolver uma operação aritmética ou lógica, o resultado será escrito no registrador especificado na própria instrução.



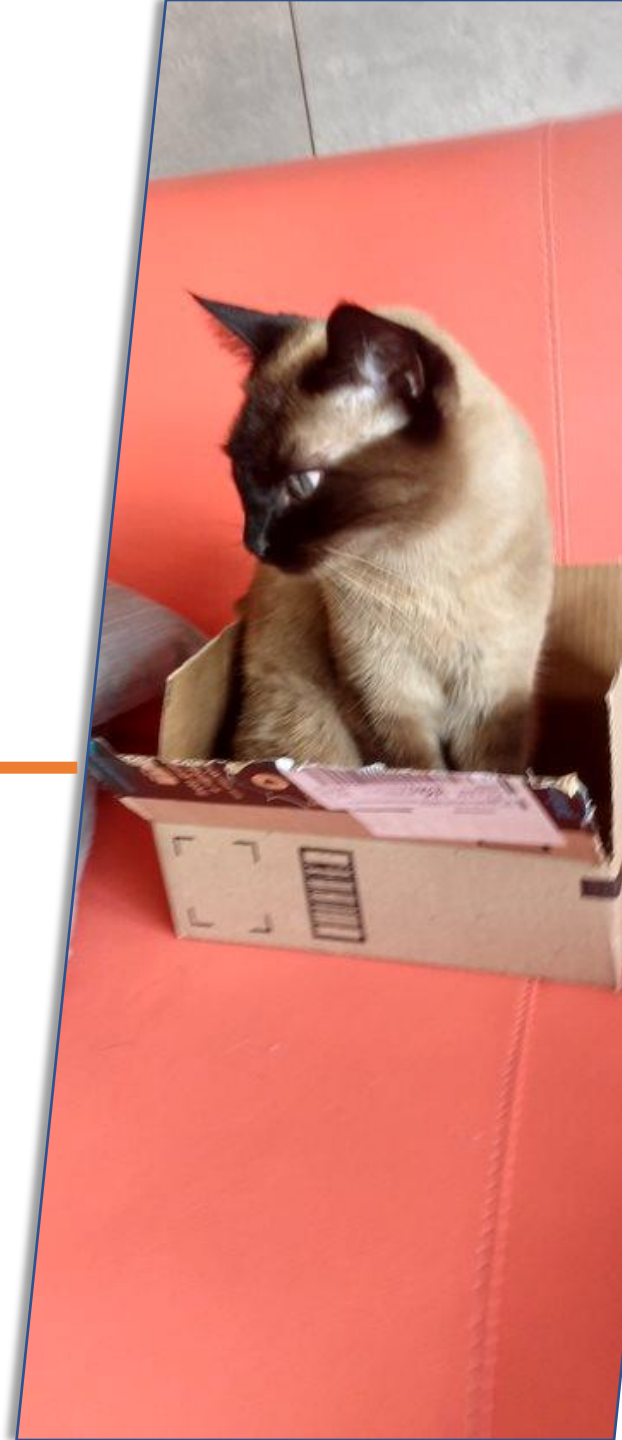
Ciclos de clock	1	2	3	4	5	6	7	8	9
Instrução 1	F	D	**X**	M	W				
Instrução 2		F	D	**X**	M	W			
Instrução 3			F	D	**X**	M	W		



MIPS Online Pipelining Simulator

Clique aqui!

O que é Performance?



Definindo Performance

Performance, ou eficiência, é ambígua e significa coisas diferentes para pessoas diferentes e até mesmo coisas diferentes para a mesma pessoa em momentos diferentes.

A definição depende do objetivo, da pessoa, da situação e do momento.

Formas de Análise

- Resposta ou Tempo de Execução:
  - O tempo entre o início e o fim de uma tarefa;
  - Mede a percepção do usuário sobre a tarefa;
- *Throughput*:
  - Total de tarefas realizadas em um determinado tempo;
  - Relevante em sistemas de gande demanda.
- Consumo:
  - Quantidade de energia consumida ou tempo de bateria.

Formas de Análise

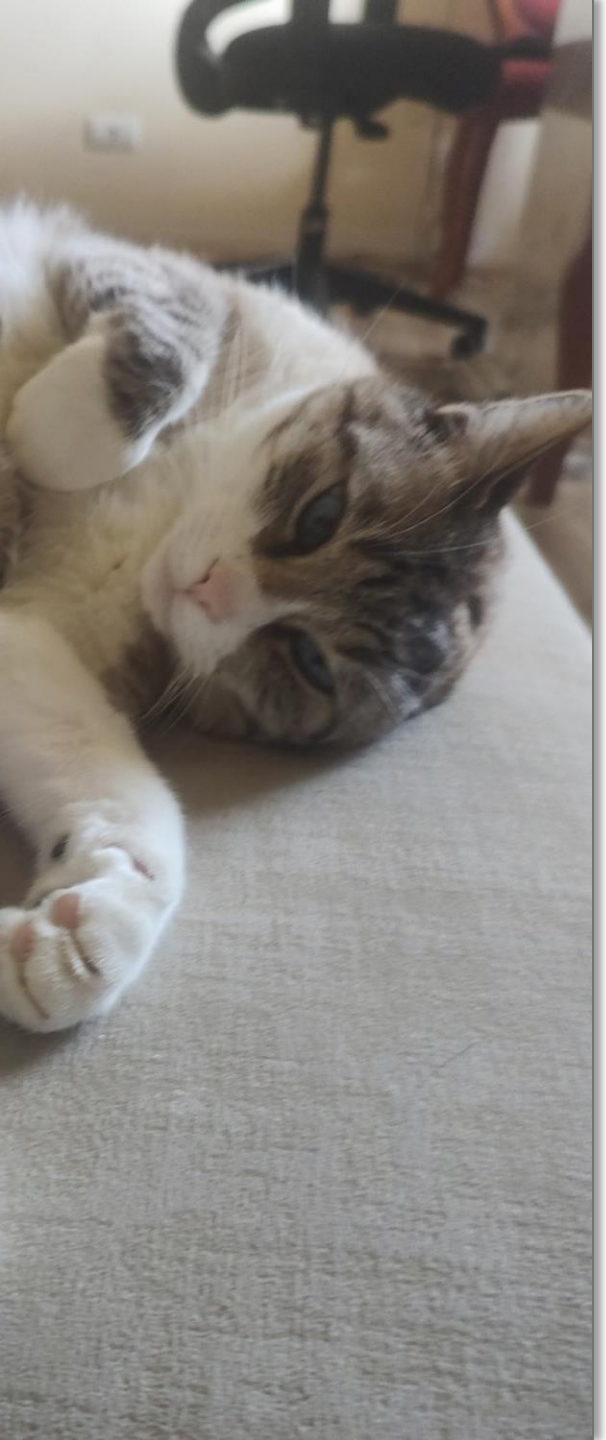
- Resposta ou Tempo de Execução:
  - O tempo entre o início e o fim de uma tarefa;
  - Mede a percepção do usuário sobre a tarefa;
- *Throughput*:
  - Total de tarefas realizadas em um determinado tempo;
  - Relevante em sistemas de gande demanda.
- Consumo:
  - Quantidade de energia consumida ou tempo de bateria.

Um Pouco de Matemática

- Maximizar a performance geralmente implica em minimizar o tempo de execução.
- A eficiência está relacionada com o tempo de execução, ou espera.

$$Performance = \frac{1}{Tempo\ de\ Execução}$$

$$Perf = \frac{1}{Tempex}$$



Exercício 2 – Testando dois Ambientes

Comparando a Performance Entre Dois Ambientes

Uma startup de tecnologia financeira está desenvolvendo uma nova aplicação de criptografia que envolve cálculos relacionados a números primos. A equipe de desenvolvimento está considerando a utilização do Google Colab e do Repl.it como ambientes de desenvolvimento e teste. No entanto, a equipe precisa determinar qual ambiente oferece o melhor desempenho para o caso de uso específico da empresa.

Comparando a Performance Entre Dois Ambientes

- Escreva uma função, *eh_primo* que recebe um número inteiro como entrada e retorna *True.*
- Escreva uma função, *calcular_primos* que recebe um número inteiro *n* como entrada e retorna uma lista com os primeiros *n* números primos.
- Utilize a função *calcular_primos* para calcular os primeiros 1000 números primos e meça o tempo necessário para realizar esse cálculo no Google Colab e no Repl.it.

Obrigado!

Frank Coelho de Alcantara 2023-1