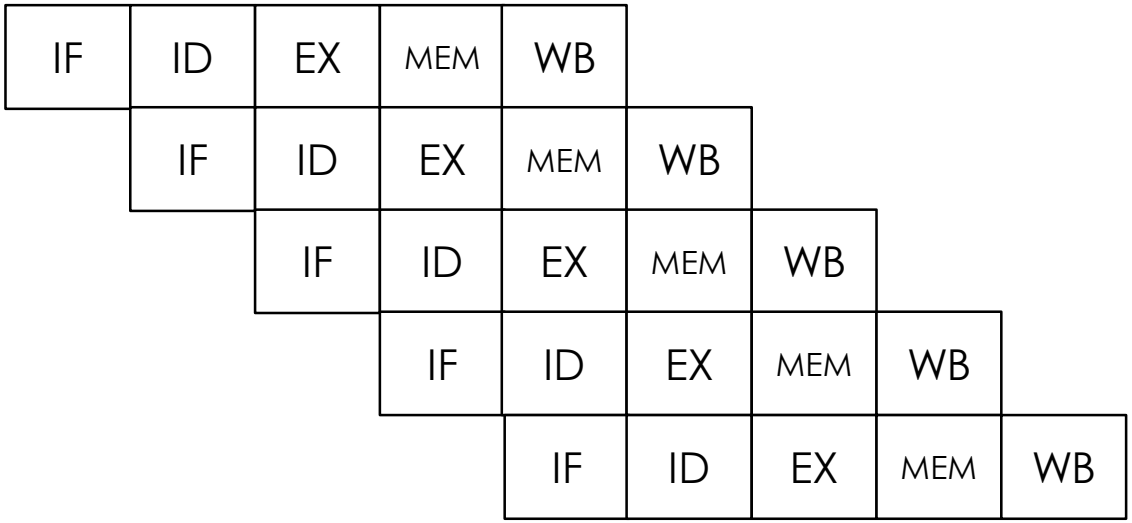
Paralelismo em Nível de Instruções (ILP)

Prof Gustavo Girão

Mais paralelismo? Onde e quando?

- ILP !!
  - Superpipelines
  - Superescalares
  - VLIW
- O objetivo é sempre diminuir o CPI abaixo de 1
- Conflitos de dados e de recursos são um problema
- Tudo tem limite!

Superpipelines



Superpipelines

10 instruções ao mesmo tempo

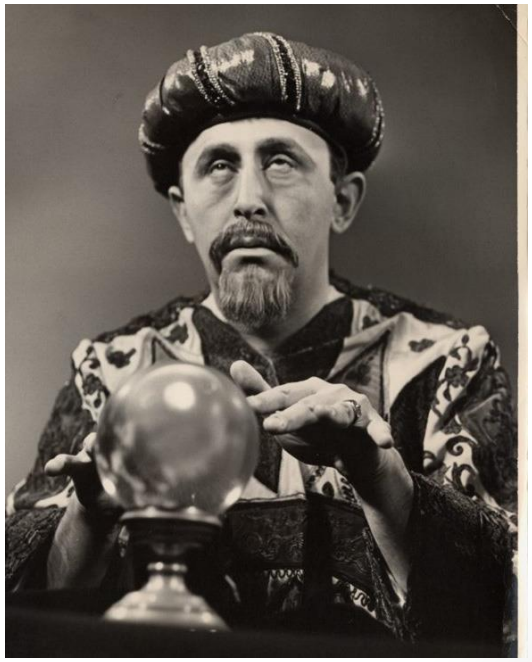
20 instruções ao mesmo tempo

Conceitos avançados

- Podemos obter CPI < 1?
  - Conceito de IPC – Instruções por ciclo
  - Como fazer isso?
    - ✧Executando mais de uma instrução (despacho)
- Despacho de instruções
  - Estático
    - ✧Superescalar
  - Dinâmico
    - ✧VLIW
- Ações
  - Definir quais instruções serão despachadas
  - Garantir corretude

Especulação

- Para aumentar o IPC é preciso saber o grau de dependência das instruções
- As vezes também é preciso adivinhar o fluxo de instruções
  - Computação Esotérica!
- Pode ser feita pelo hardware ou Software
- O problema é quando erramos
  - Mecanismos de recuperação
  - Exceções disparadas
  - Flush do pipeline
  - Perda de tempo!

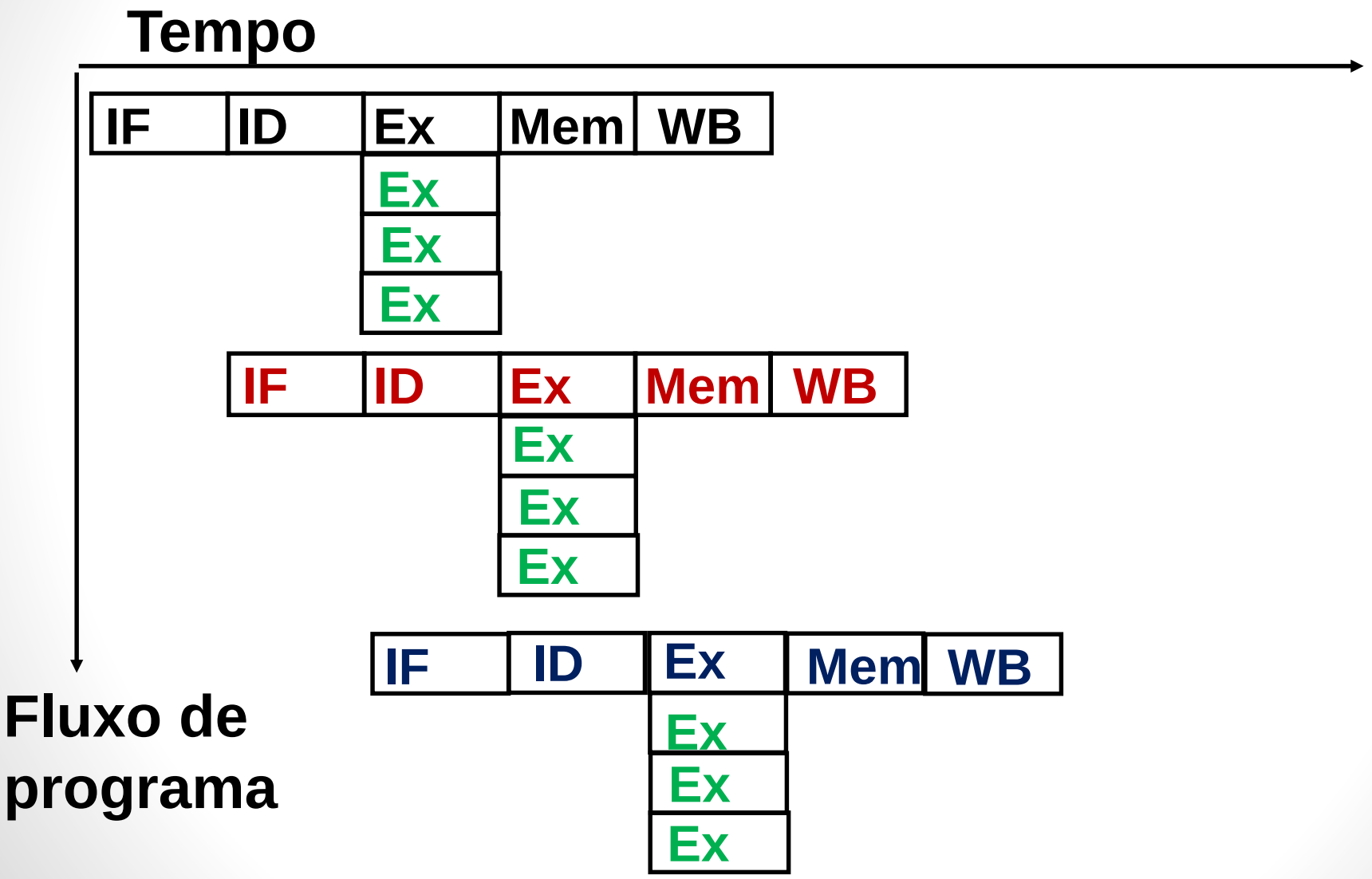


Execução fora de ordem

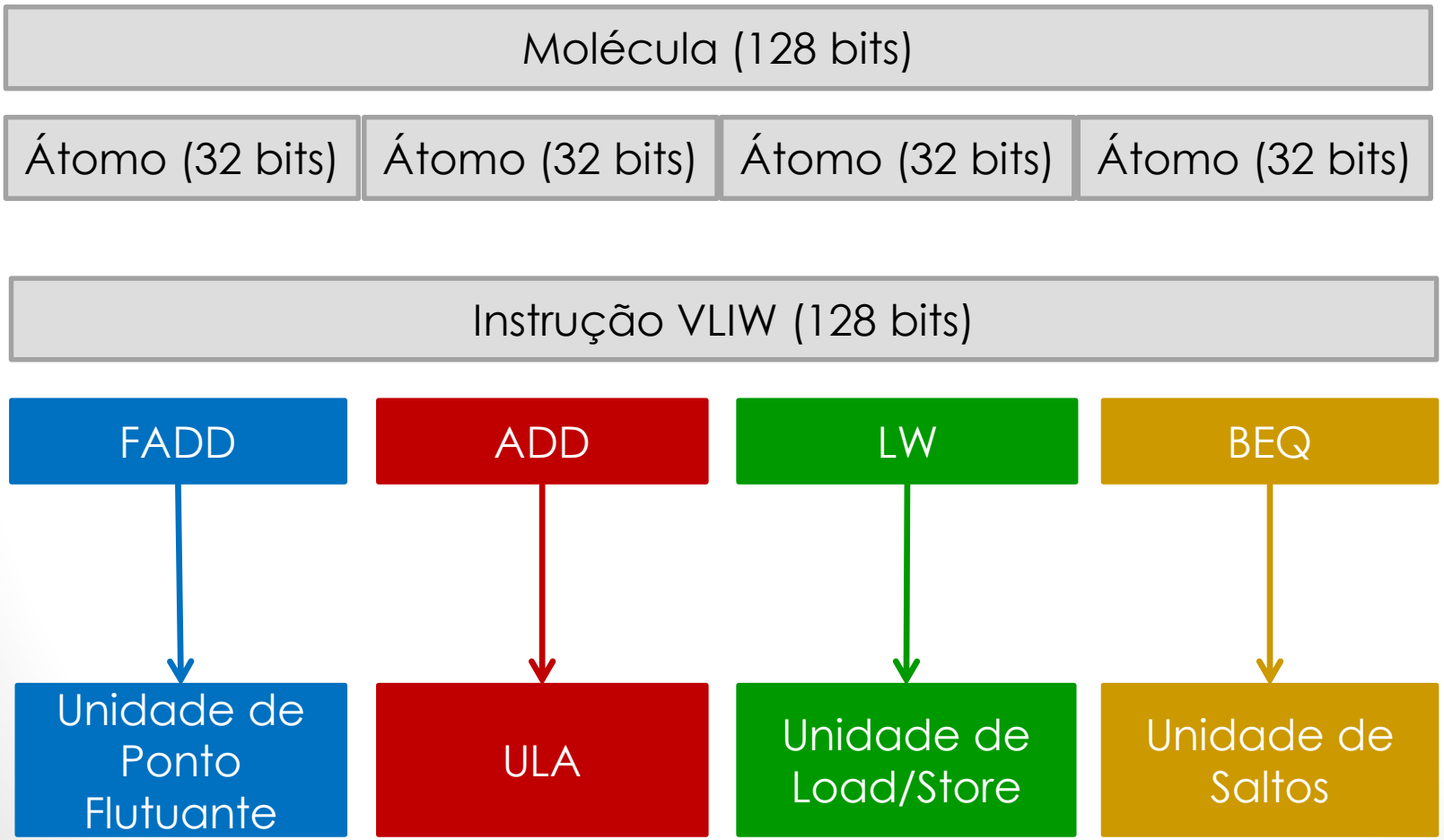
- Uma das soluções para evitar (ou amenizar) stalls no pipeline é o reordenamento de instruções
- Entretanto, podemos tentar executar instruções fora de ordem a qualquer momento
  - Para amenizar atrasos em função de conflitos estruturais, por exemplo

Despacho Estático

Very Large Instruction Word



Very Large Instruction Word (VLIW



Processadores VLIW

- Deu inicio à "programação horizontal"
  - Instruções muito "largas" utilizadas para gerar sinais de controle diretos
- Exemplos de processadores VLIW comerciais
  - Intel i860 RISC (dual mode: **scalar** and **VLIW**)
  - Intel I-64 (EPIC: **Itanium** and **Itanium 2**)
  - Transmeta **Crusoe**
  - Lucent/Motorola **StarCore**
  - ADI **TigerSHARC**
  - Infineon (Siemens) **Carmel**

Processadores VLIW

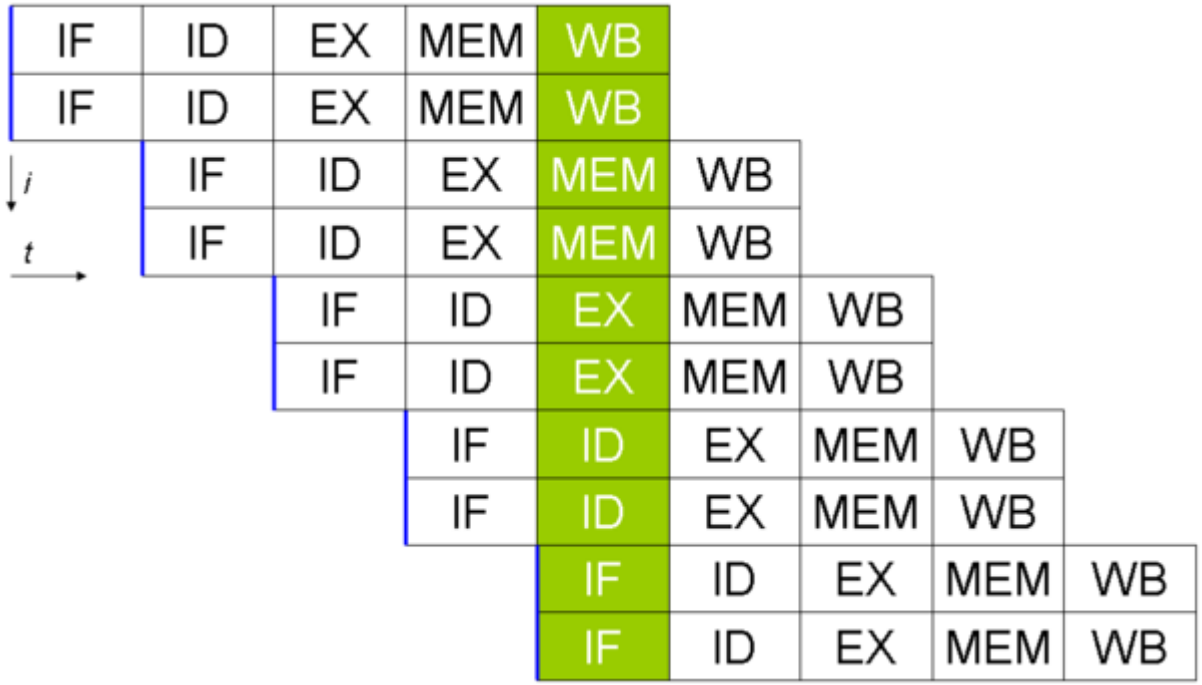
- São processadores de despacho múltiplo estático
  - O compilador decide quais instruções serão despachadas e executadas simultaneamente
  - **Pacote de despacho**

- VLIWs tem:
  - Múltiplas unidades funcionais
  - Banco de registradores com múltiplas portas
  - Barramento de instruções muito largo

Despacho Dinâmico

Princípios da super-escalaridade

- Várias unidades de execução
- Várias instruções completadas simultaneamente em cada ciclo de relógio
- **Hardware** é responsável pela extração de paralelismo



Princípios da super-escalaridade

- Processadores comuns conseguem IPC = 2
  - Técnicas avançadas podem chegar a IPC = 6
- Problemas com a execução simultânea de instruções
  - Dependências **estruturais**
    - memória
  - dependências de **dados**
    - verdadeiras
    - falsas - anti-dependências, dependências de saída
  - dependências de **controle** (desvios)

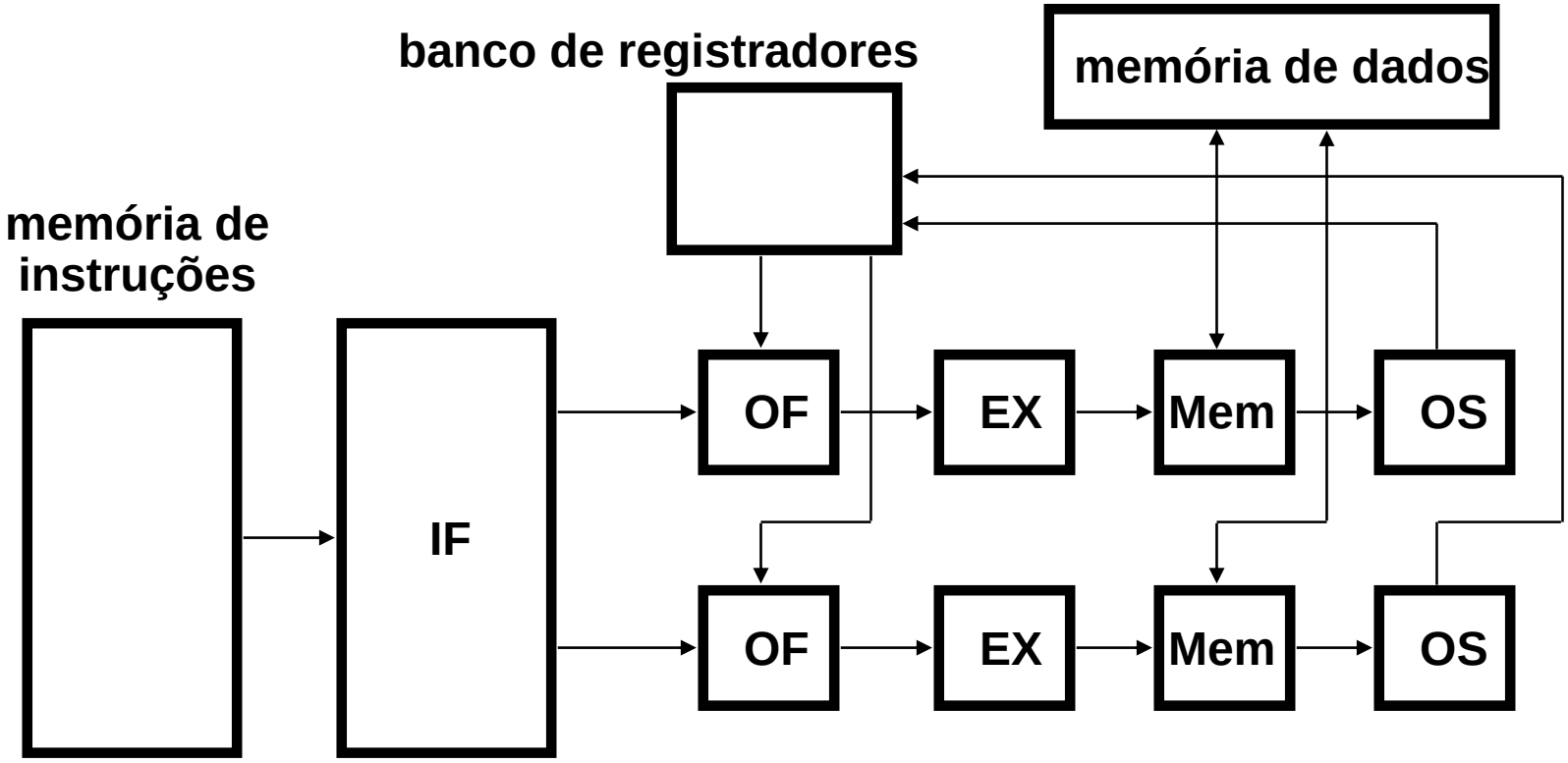
Princípios da super-escalaridade

- Término das instruções pode não seguir a seqüência estabelecida no programa

- Processador com capacidade de "look-ahead"
  - se há conflito que impede execução da instrução atual, processador
    - examina instruções além do ponto atual do programa
    - procura instruções que sejam independentes
    - executa estas instruções

- Possibilidade de execução **fora de ordem**
  - cuidado para manter a corretude dos resultados do programa

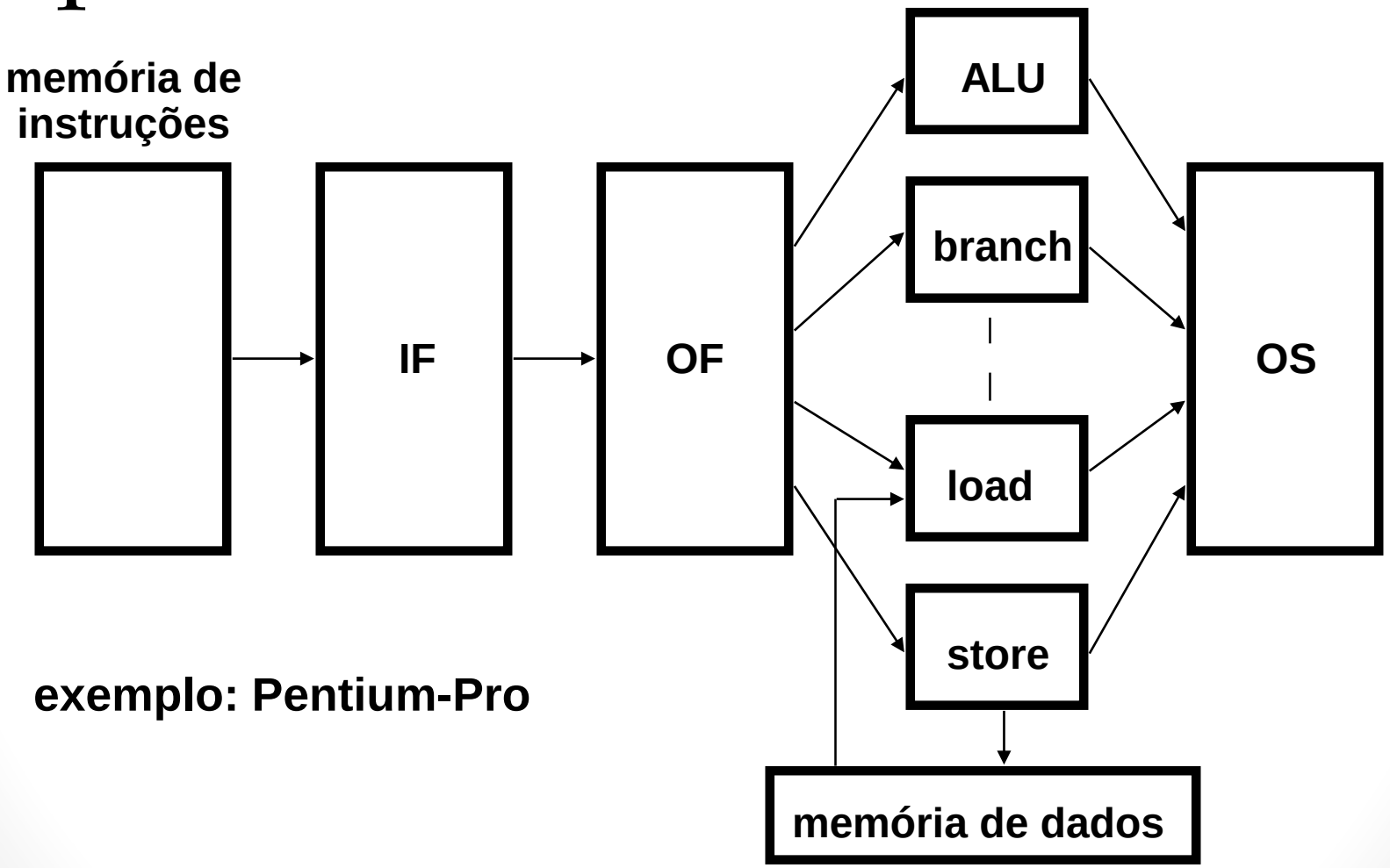
Processador com 2 pipelines



exemplo: Pentium I

cache de instruções precisa fornecer dobro de instruções por ciclo!!

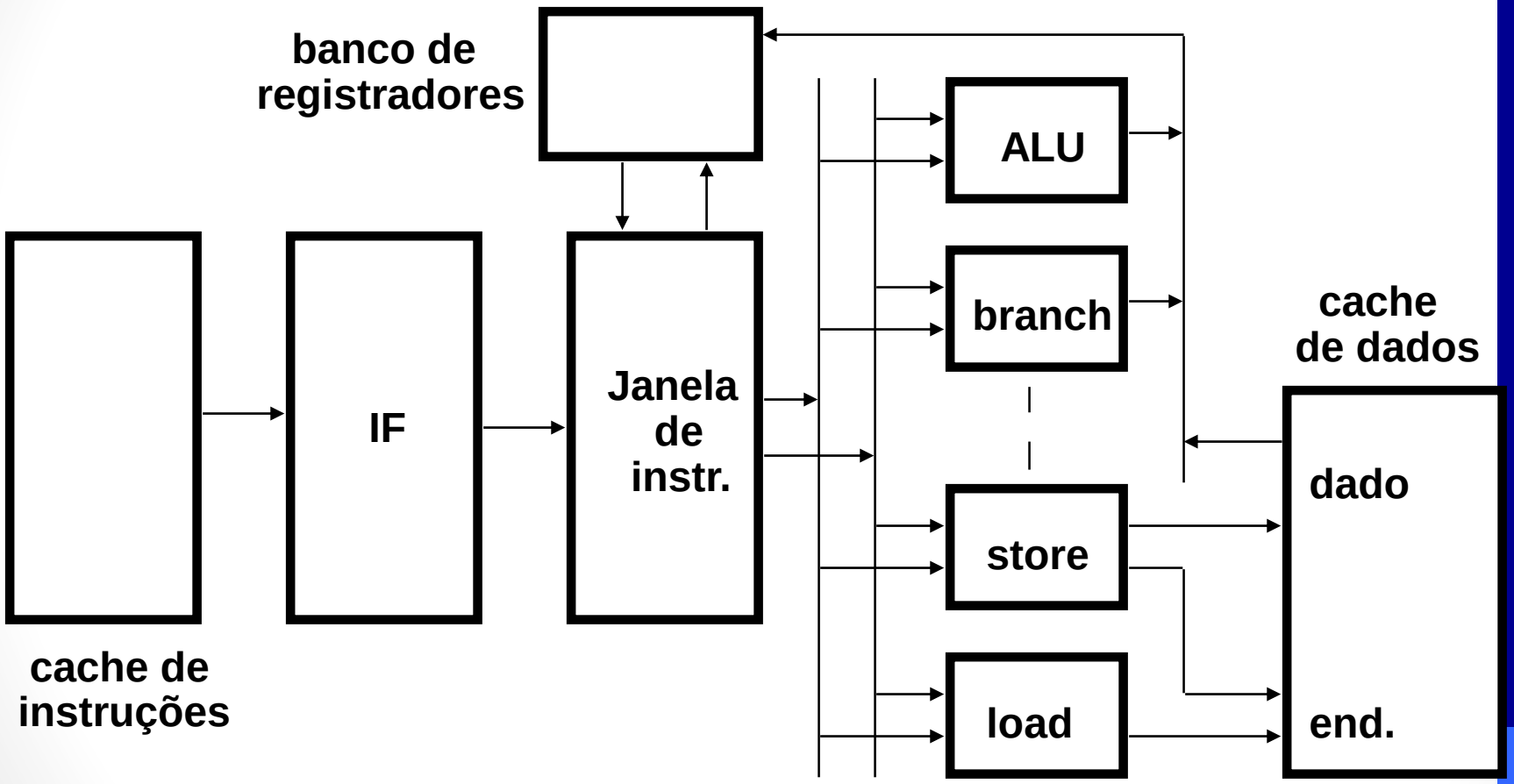
Unidades de execução especializadas



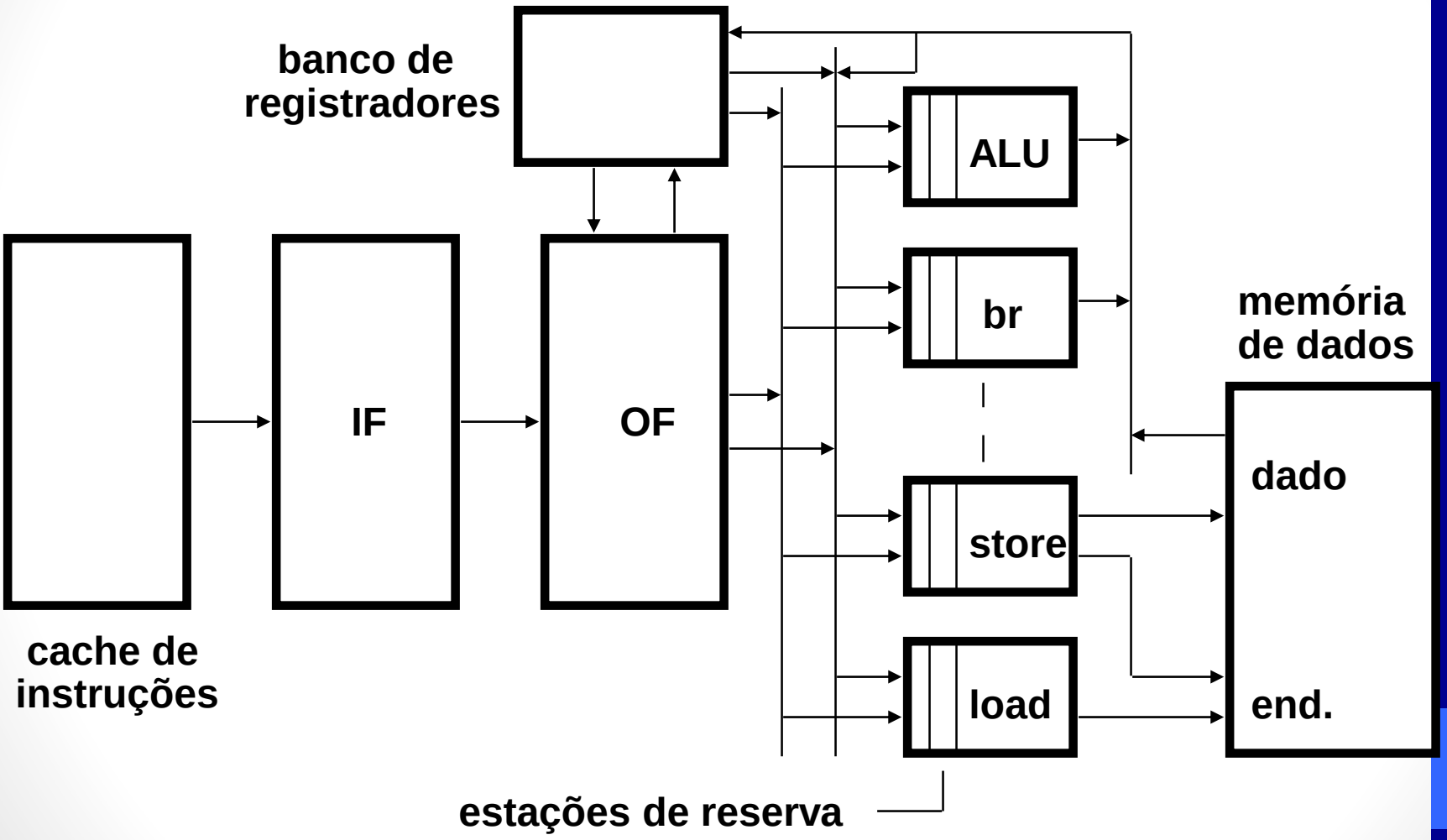
exemplo: Pentium-Pro

Outras técnicas mais refinadas

Janela de instruções centralizada



Janela de instruções distribuída



Renomeação de registradores

- Objetivo: resolver Antidependências e dependências de saída.
  - Poderia ser resolvido com mais registradores
- Exemplo
  - ADD R1, R2, R3 ; R1 = R2 + R3
  - ADD R2, R4, 1 ; R2 = R4 + 1 antidependência em R2
  - ADD R1, R5, R6 ; R1 = R5 + R6 dependência de saída em R1
- Utilizando 2 outros registradores R7 e R8 pode-se eliminar as dependências falsas
  - ADD R1, R2, R3 ; R1 = R2 + R3
  - ADD R7, R4, 1 ; R7 = R4 + 1
  - ADD R8, R5, R6 ; R8 = R5 + R6

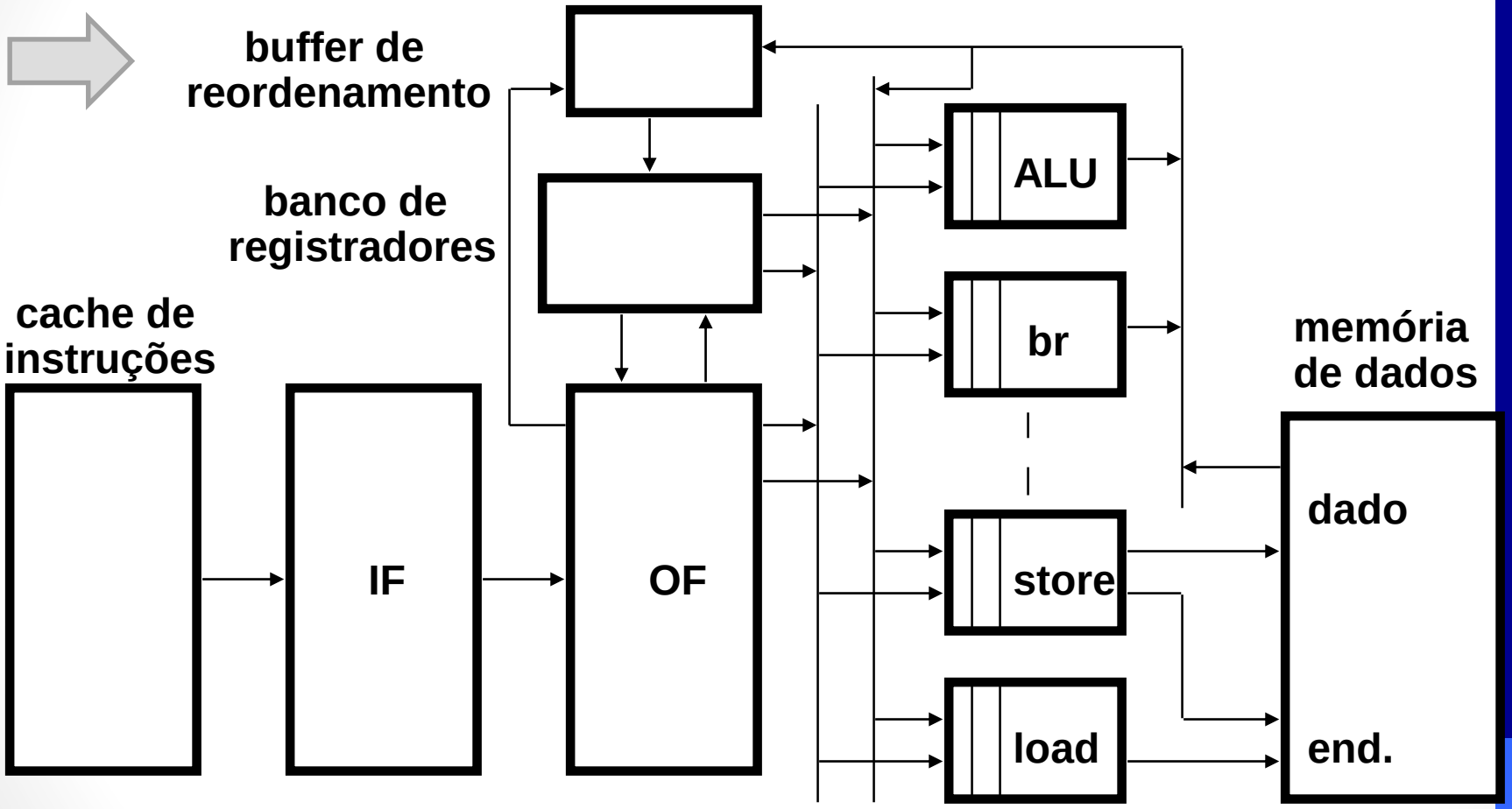
Renomeação de registradores

- Não é possível criar número ilimitado de registradores
- Arquitetura deve manter compatibilidade quanto aos registradores visíveis para o programador
- Solução
  - utilizar banco de registradores interno, bem maior do que o banco visível
  - renomear registradores temporariamente
  - cada registrador visível que é escrito numa instrução é renomeado para um registrador interno escolhido dinamicamente

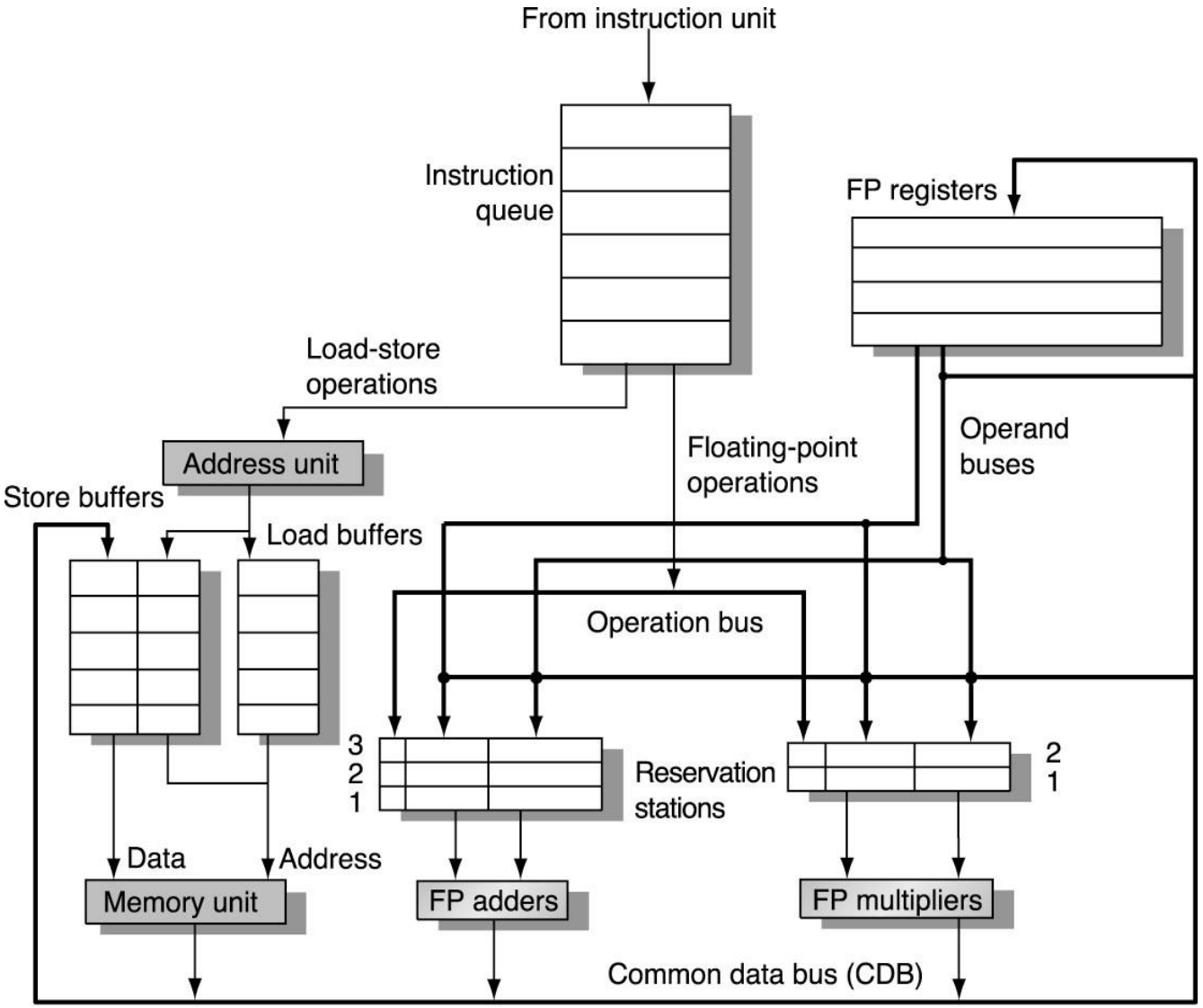
Renomeação de registradores

- No exemplo anterior, supondo registradores internos Ra, Rb, Rc, Rd, Re, Rf, Rg
  - ADD Ra, Rb, Rc
  - ADD Rd, Ra, 1
  - ADD Re, Rf, Rg
- Antidependência e dependência de saída foram eliminadas

Buffer de reordenamento



Microarchitecture of an ILP-based CPU (Power PC)



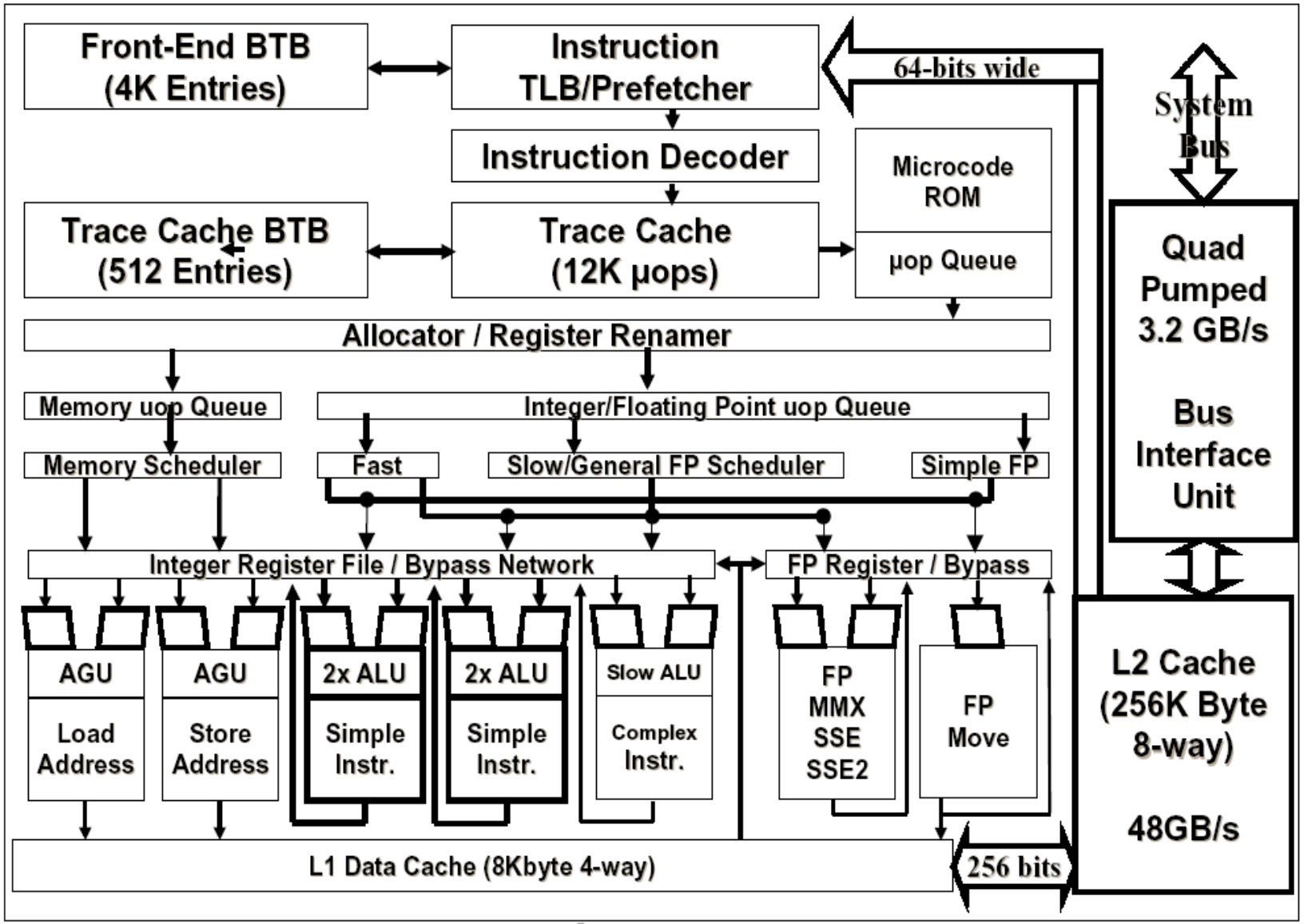


Figure 4: Pentium® 4 processor microarchitecture

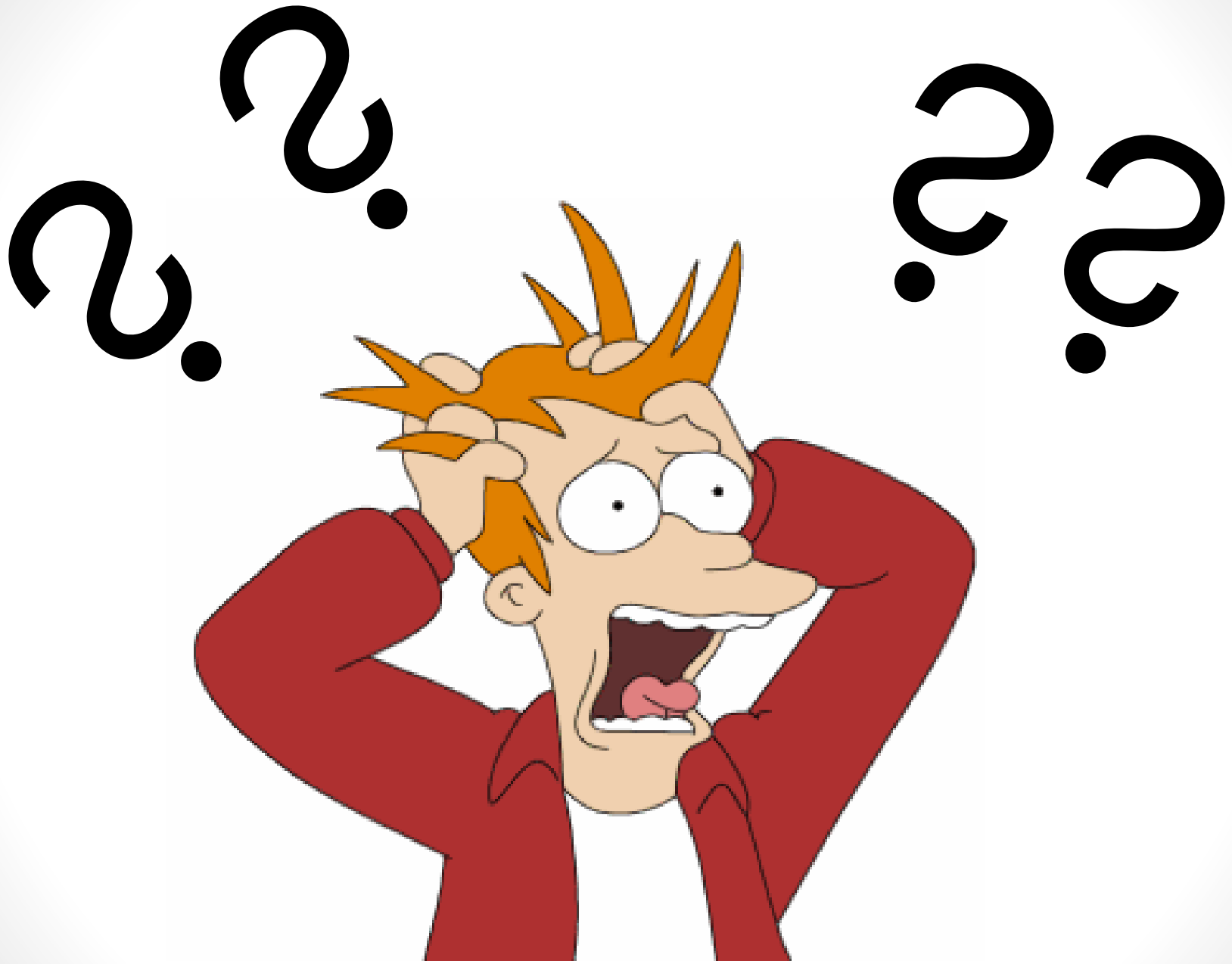
VLIW vs Superescalar

- Complexidade do hardware
  - VLIW é mais simples e mais fácil de escalar (decodificação)
  - Superescalares precisam encontrar paralelismo em tempo de execução

- Programação e complexidade do compilador
  - VLIW é mais complexo
    - Compiladores precisam encontrar todo o paralelismo
    - Lock step: conflitos podem fazer com que outras instruções fiquem paradas
  - Superescalares utilizam compilação comum

VLIW vs Superescalar

- Compatibilidade de software
  - VLIW precisa de recompilação
- Espaço em memória
  - VLIW precisa de maior largura de banda de instruções
  - Técnicas para evitar conflitos podem ser despendiosas
    - NOPs são um desperdício
    - Loop unrolling utiliza uma quantidade ainda maior de espaço em memória



Bibliografia

- Patterson e Hennessy

  Organização e projeto de computadores – A interface Hardware/Software

  CAPÍTULO 6 – Melhorando o desempenho com Pipelining

- William Stalling

  Arquitetura e Organização de Computadores

  CAPÍTULO 12 – Sessão 12.4

Próxima aula

- Memórias Cache