Arquitetura de Computadores

Pipeline: Motivação e Conceitos

Desempenho de uma arquitetura

- Capítulos 4 e 6 - *Organização e Projeto de Computadores: A interface Hardware/Software.* Patterson, D.A.; Hennessy, J.L. 3ª edição, Ed. Campus.
- Outra opção:
  - *Computer Architecture: A Quantitative Approach.* J.L. Hennessy & D. A. Patterson.

Ciclos...

- Um determinado programa exigirá:
  - um determinado número de instruções (instruções de máquina);
  - um determinado número de ciclos;
  - um determinado número de segundos.
- Relacionando essas quantidades:
  - tempo de ciclo (segundos por ciclo);
  - velocidade de clock (ciclos por segundo);
  - CPI (ciclos por instrução) — uma aplicação com excessivo uso de ponto flutuante pode ter uma CPI mais alta.

Desempenho

- O desempenho é determinado pelo tempo de execução.
- Qualquer uma das outras variáveis igualam o desempenho?
  - número de ciclos para executar o programa?
  - número de instruções no programa?
  - número de ciclos por segundo?
  - número médio de ciclos por instrução?
  - número médio de instruções por segundo?

Desempenho

- **Armadilha comum**: pensar que uma das variáveis é indicadora do desempenho, quando na realidade não é.

Exemplo de CPI

- Suponha que tenhamos duas implementações da mesma arquitetura do conjunto de instruções.
- Para um determinado programa:
  - máquina A tem um tempo de ciclo de clock de 250 ps e uma CPI de 2,0;
  - máquina B tem um tempo de ciclo de clock de 500 ps e uma CPI de 1,2.

Exemplo de CPI

- Que máquina é mais rápida para esse programa e o quanto?
- *Se duas máquinas possuem a mesma arquitetura do conjunto de instruções, qual de nossas quantidades (por exemplo, velocidade de clock, CPI, tempo de execução, número de instruções) será sempre idêntica?*

Princípios de Projeto

- **Make the common case fast!** ou “Faça o caso comum rápido”!
- Por exemplo, considere adição de dois números.
  - *Overflow* é raro.
  - Então, otimize o projeto para adição de dois números sem *overflow*!

Lei de Amdahl

- Lei fundamental para quantificar otimizações.
- Diz que: “o ganho de desempenho ao usar um modo de execução mais rápido é limitado pela fração de tempo que este modo rápido pode ser utilizado”.

Lei de Amdahl

- *Speedup* indica quão mais rápido uma tarefa será executada em uma máquina com melhorias quando comparada com a máquina original.

$$Speedup = \frac{T_{execução\,da\,tarefa\,toda\,sem\,melhora}}{T_{execução\,da\,tarefa\,toda\,com\,melhora}}$$

Lei de Amdahl - Definições

- Fração do tempo de computação original que pode ser convertida para aproveitar a melhoria:
  - Ex.: se 20 segundos de um programa que demora 60 segundos para ser executado podem utilizar a melhoria, então:

$$Fraction_{enhanced} = 20/60 \leqslant 1$$

Lei de Amdahl – Definições (2)

- Melhora ganha pelo modo de execução otimizado, i.e., quão mais rápido uma tarefa seria executada se o modo otimizado fosse usado na tarefa toda:
  - Ex.: se a execução original demora 5 segundos, e a otimizada demora 2 segundos, então:

$$Speedup_{enhanced} = 5/2 > 1$$

Lei de Amdahl – Definições (3)

- Assim:

$$Speedup_{overall} = \frac{TempoExecução_{antigo}}{TempoExecução_{novo}}$$

$$Speedup_{overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Lei de Amdahl – Exemplo

- Considere uma melhoria que roda 10 vezes mais rápido do que a máquina original, mas só pode ser utilizada 40% do tempo. Qual o Speedup?
  - Fraction_en = 0,4 e Speedup_en = 10

$$Speedup_{overall} = \frac{1}{(1-0,40)+\frac{0,40}{10}} = \frac{1}{0,604} = 1,66$$

Lei de Amdahl

- Serve de guia para:
  - entender quanto uma otimização pode melhorar o desempenho;
  - e determinar como distribuir recursos para melhorar custo/desempenho.
- Também pode ser usada para comparar 2 alternativas de projeto (próximo exemplo).

Lei de Amdahl – Exemplo 2

- Implementação de FP (floating-point).
- Raiz quadrada -> 20% do tempo de execução de um benchmark crítico.
- 2 alternativas:
  - hardware para raiz quadrada em FP -> speedup = 10;
  - todas instruções de FP 2 vezes mais rápido (50% do tempo de execução).

Lei de Amdahl – Exemplo 2

- Qual alternativa é melhor?

$$Speedup_{RQ} = \frac{1}{(1-0,20)+\frac{0,20}{10}} = \frac{1}{0,82} = 1,22$$

$$Speedup_{FP} = \frac{1}{(1-0,50)+\frac{0,50}{2}} = \frac{1}{0,75} = 1,33$$

Resumindo...

- O desempenho é específico a um determinado programa!
- Dada uma arquitetura, os aumentos de desempenho vêm de:
  - aumentos na velocidade de clock (sem efeitos de CPI adversos);
  - melhorias na organização do processador que diminuem a CPI;
  - melhorias no compilador que diminuem a CPI e/ou a contagem de instruções;
  - escolhas de algoritmo/linguagem que afetam a contagem de instruções.

Como otimizar a máquina?

- Capítulo 11.4 – Stallings.
- Capítulo 6 – Patterson & Hennessy.

Prefetch- Busca Antecipada

- Busca (fetch) de instrução acessa memória principal.
- Execução não acessa memória principal normalmente.
- Então, pode buscar a próxima instrução durante a execução da instrução atual!
- Isto é “instruction prefetch” (busca antecipada de instrução).

Prefetch (2)

- Pipeline de 2 estágios!
- Melhora o desempenho!
- Em condições ideais:
  - Dobra número de instruções executadas!

Prefetch (3)

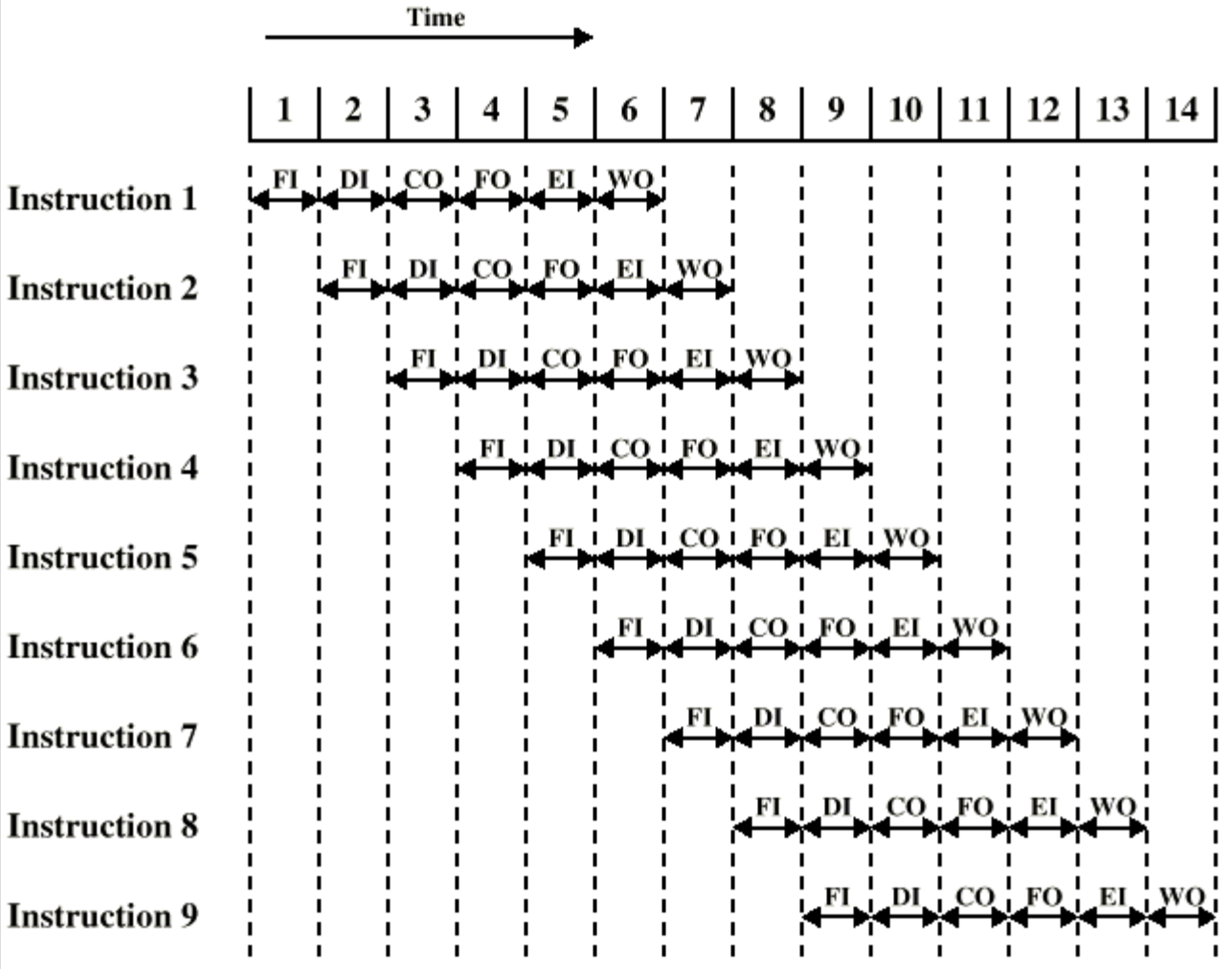
- Na prática, não dobra...
  - Tempo de execução > T de busca:
    - Execução também envolve leitura e armazenamento de operandos.
  - Desvio condicional precisa esperar a execução da instrução para determinar próxima instrução!

- Adicionar mais estágios para melhorar desempenho.

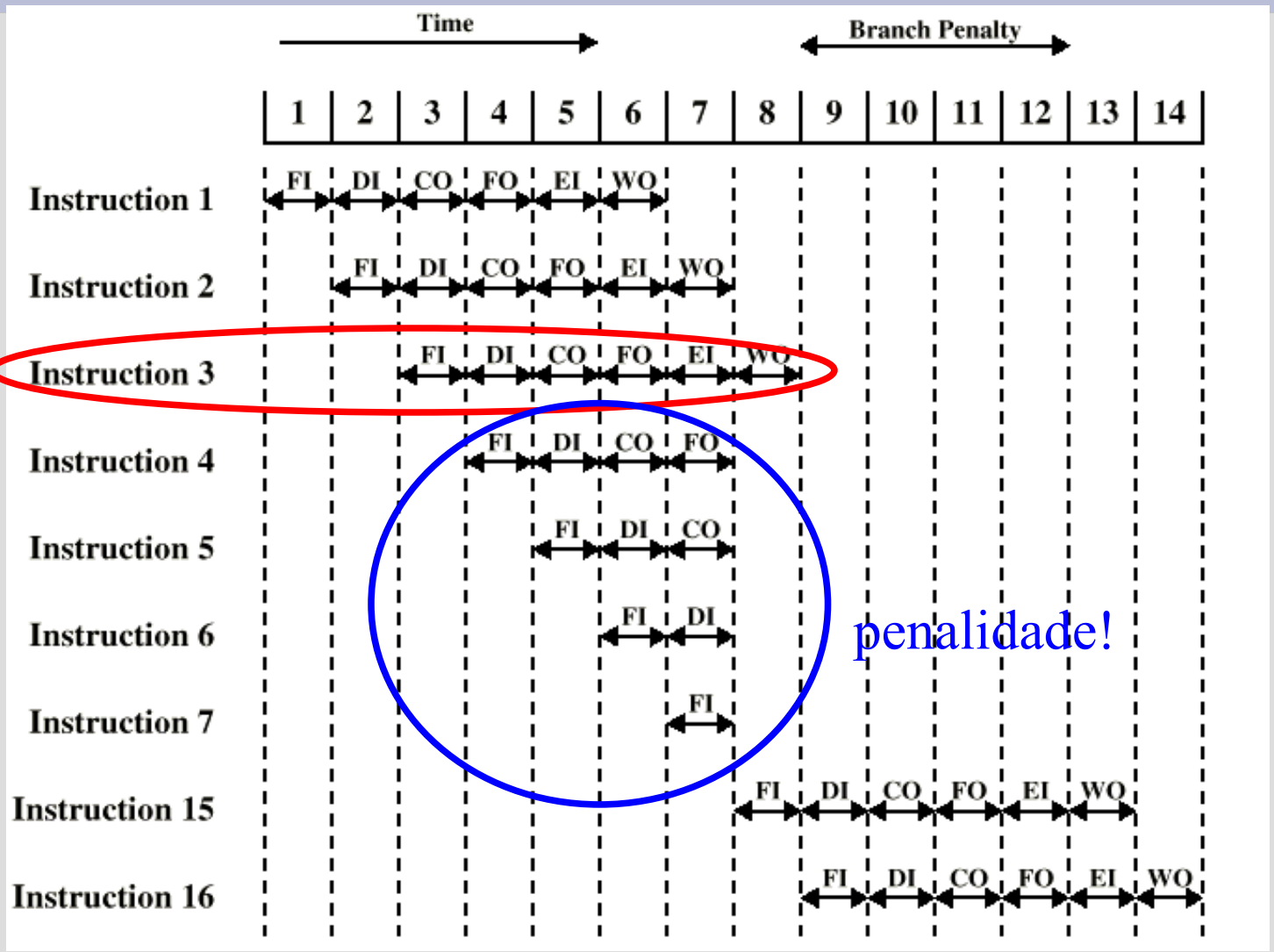
Pipelining

- Múltiplos estágios:
  - Busca de instrução (FI);
  - Decodifica instrução (DI);
  - Calcula operandos (i.e. EAs) (CO);
  - Busca operandos (FO);
  - Executa instruções (EI);
  - Escreve resultados (WO).
- Sobrepor estas operações!

Temporização de Pipeline



Branch (desvio) em Pipeline



Custos de Pipeline

- *Overhead* da movimentação de dados entre estágios do pipeline:
  - armazenamento temporário.
- Lógica de controle mais complexa:
  - dependências entre acessos à memória e registradores;
  - controle de estágios.

Desempenho

- Seja:
  - t = atraso máximo de estágio;
  - k = número máximo de estágios;
  - d = atraso máximo para propagação de sinais entre estágios.
  - n = número de instruções executadas;
  - T = tempo para executar *n* instruções.

$$Speedup_k = \frac{T_1}{T_k} = \frac{n*k*t}{[k+(n-1)]*t} = \frac{n*k}{k+(n-1)}$$

Desempenho (2)

$$Speedup_k = \frac{T_1}{T_k} = \frac{n*k*t}{[k+(n-1)]*t} = \frac{n*k}{k+(n-1)}$$

- No limite, fator de aceleração = k.
- Na prática, ganho diminuído por conta do *overhead*!
- Número típico de estágios: 6 a 9.

Resolvendo Desvios

- Diversas abordagens:
  - múltiplos fluxos;
  - busca antecipada da instrução-alvo do desvio;
  - memória de laço de repetição (*loop buffer*);
  - previsão de desvios;
  - atraso do desvio (d*elayed branch*).

(1) Múltiplos fluxos

- Usar dois pipelines:
  - Buscar as instruções de cada caminho do desvio em um pipeline separado;
  - Usar o pipeline correto.
- Leva a problemas de uso dos registradores e barramentos.
- Múltiplos desvios levam a necessidade de criar mais pipelines!
- Usado por IBM 370/168 e 3033.

(2) Busca antecipada da instrução-alvo do desvio

- Instrução-alvo do desvio é buscado (prefetched), assim como as instruções seguintes.
- Mantém a instrução-alvo em registrador até que a instrução desvio seja executada.
- Usado pelo IBM 360/91.

(3) Memória de laço de repetição

- Memória muito rápida.
- Estágio de busca do pipeline a controla/mantém.
- Verificar buffer antes de buscar na memória.
- Bom para loops pequenos ou jumps (if-then e if-then-else).
- Usado pelo CRAY-1.

(4) Previsão de Desvios (1)

- Prever que nunca ocorrerá (never taken):
  - sempre busca a próxima instrução;
  - 68020 & VAX 11/780.
- Prever que sempre ocorrerá (always taken):
  - sempre busca a instrução alvo.
- Previsão baseda no Opcode:
  - algumas instruções são mais prováveis de resultar em saltos do que outras;
  - até 75% de sucesso.

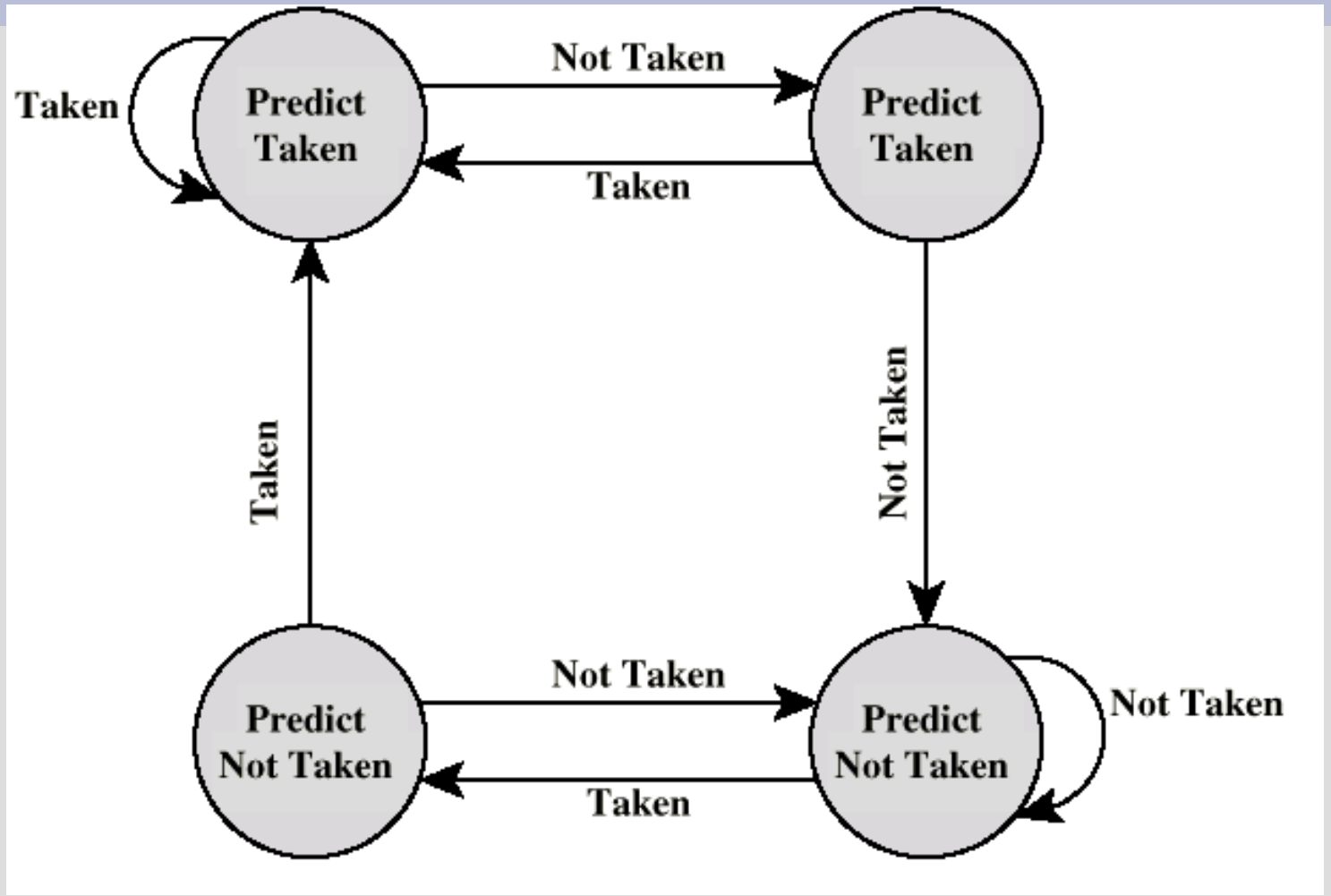
Desvios...

- Análise de comportamento de programas:
  - Desvios condicionais são tomados em mais de 50% das vezes.
- Então, se o custo da busca antecipada de instruções for o mesmo, melhor estratégia.

(4) Previsão de Desvios (2)

- Previsão baseada em chaves de desvio tomado e não tomado:
  - baseado no histórico (1 bit);
  - dinâmico.
- Previsão baseada em tabela de histórico de desvios:
  - bom para loops;
  - armazenada em cache;
  - dinâmico.

Previsão de Desvios baseado em Histórico - Diagrama de Estado



(5) Atraso do Desvio

- Reorganizar as instruções de um programa;
- não executar o salto até que seja necessário;
- mais no Cap. 12...

Próximo...

- Arquiteturas CISC x RISC.
- Juntando tudo:
  - arquitetura RISC:
    - projeto de ULA e Unidade de Controle;
    - re-organizando o projeto para uso de Pipeline.