ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Apresentação

Profª. Fabiana F F Peres

Apoio: Camile Bordini

Ementa

- Estudo de conceitos de arquitetura de computadores
- Estudo das arquiteturas RISC e CISC de computadores, incluindo o recurso de pipeline
- Avaliação de desempenho de computadores

Ementa

- Estudo das tecnologias de memória: memória cache, memória virtual
- Organização de memórias
- Interface entre processador e periféricos: barramentos, dispositivos de entrada e saída
- Arquiteturas paralelas

Conteúdo programático

- Introdução
- Desempenho/ Performance
- Arquitetura de Computadores
- Modelo de Arquitetura RISC
- Pipeline
- Microprogramação
- Memória
- Interface Processador/Periféricos
- Arquiteturas Paralelas

Porque estudar Organização e Arquitetura de Computadores?

- “Os profissionais de computação não devem encarar o computador apenas como uma caixa preta que executa programas por mágica”

- “A Arquitetura e Organização de Computadores baseia-se em na compreensão sobre o **ambiente de hardware** no qual toda a computação é baseada, e na interface que é fornecida para **camadas de software superiores**”

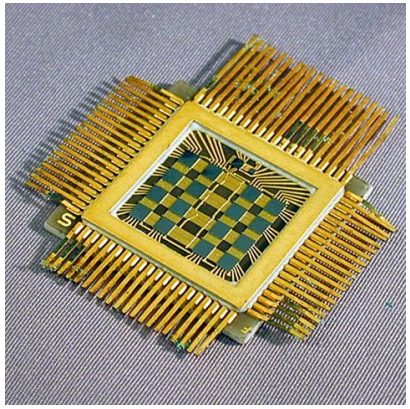
- “Os alunos devem adquirir uma compreensão dos **componentes de um sistema de computador**, e em particular, o desafio de aproveitar o **paralelismo** para sustentar melhorias de desempenho agora e no futuro”

Computer Science Curricula, 2013 – IEEE, pg. 62

Alguns dilemas

- Os bons programadores sempre se preocuparam com o desempenho de seus programas

- **Décadas 1960 e 1970**: uma grande limitação no desempenho era o tamanho da memória do computador

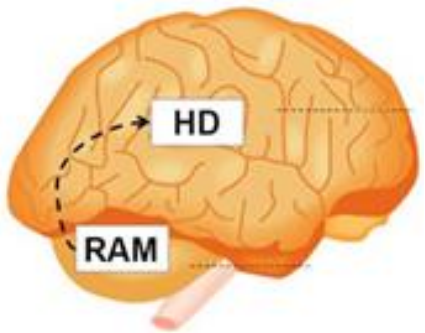


  - Programadores seguiam a regra: minimizar o espaço ocupado na memória

Alguns dilemas

- **Últimas décadas**: os avanços na arquitetura e nas tecnologias de memórias reduziram drasticamente a importância do tamanho da memória na maioria das aplicações (com exceção dos sistemas embutidos)

- Agora programadores precisam entender:
  - A **natureza paralela dos processadores**
  - A **natureza hierárquica das memórias**
  - Eficiência em **consumo de energia**



Questões

Como os programas escritos em uma linguagem de alto nível, como C ou Java, são traduzidos para a linguagem do hardware?

E como o hardware executa o programa resultante?

Qual é a interface entre o software e o hardware e como o software instrui o hardware a executar as funções necessárias?

Questões

O que determina o desempenho de um programa e como um programador pode melhorar o desempenho?

Quais técnicas podem ser usadas pelos projetistas de hardware para melhorar o desempenho? E a eficiência energética?

Quais são as razões e as consequências da mudança do processamento sequencial para o processamento paralelo?

Questões

“Sem entender as respostas a essas perguntas, melhorar o desempenho do seu programa em um computador moderno ou avaliar quais recursos podem tornar um computador melhor do que outro será um processo complexo de tentativa e erro, em vez de um procedimento científico conduzido por discernimento e análise”.

Patterson & Hennessy, pg. 49

Desde o primeiro computador comercial em 1951, que grandes ideias os arquitetos de computador criaram para estabelecer as bases da computação moderna?

Grandes ideias sobre arquitetura de computadores

1. **Lei de Moore** (1965, Gordon Moore – um dos fundadores da Intel)

   – Os recursos do circuito integrado dobram a cada 18 a 24 meses

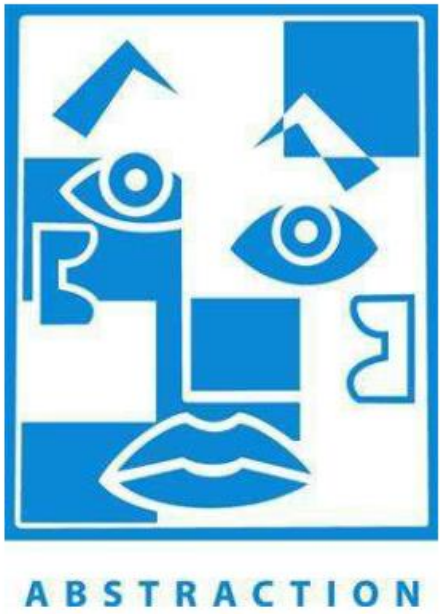
   – Projetistas de computador precisam antecipar onde estará a tecnologia quando um projeto de computador terminar, e não quando ele começar

Grandes ideias sobre arquitetura de computadores

2. **Técnica de produtividade importante tanto para o *hardware* quanto para o *software*:**

   – O uso de **abstrações** para representar um projeto de computador em diferentes níveis

   – Detalhes de nível mais baixo serão ocultados, para oferecer um modelo mais simples nos níveis mais altos



Grandes ideias sobre arquitetura de computadores

3. **Torne o caso comum veloz**

   – Melhor tornar o caso comum veloz do que otimizar o caso raro, em termos de desempenho

   – O caso comum normalmente é mais simples do que o caso raro

   – No entanto, é necessário saber qual é o caso comum – possível apenas com experimentação e medição

Grandes ideias sobre arquitetura de computadores

4. **Desempenho através de paralelismo**

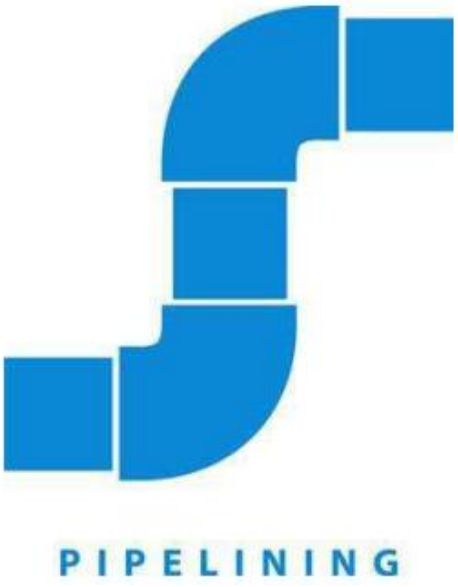
   – Desde o nascimento da computação, os arquitetos de computador têm oferecido projetos que geram mais desempenho realizando operações em paralelo

   – Veremos vários exemplos de paralelismo

Grandes ideias sobre arquitetura de computadores

5. **Desempenho através de pipelining**

   – Um padrão de paralelismo em particular é tão prevalecente na arquitetura de computação que merece seu próprio nome: **pipelining**



Grandes ideias sobre arquitetura de computadores

6. **Desempenho através de predição**

   – "*Ás vezes pode ser melhor pedir perdão do que permissão*"

   – Em certos casos, pode ser mais rápido prever e agir do que esperar até saber o correto - supondo que o mecanismo para se recuperar de um erro não seja tão dispendioso e sua predição seja relativamente precisa

Grandes ideias sobre arquitetura de computadores

7. Hierarquia de memórias

– Os programadores desejam que a memória seja rápida, grande e barata

– É possível resolver esse conflito com uma **hierarquia de memórias**
  - **No topo**: mais rápida, menor e mais cara
  - **Na base**: mais lenta, maior e mais barata

Grandes ideias sobre arquitetura de computadores

8. **Confiabilidade através de redundância**

   – Os computadores não apenas precisam ser rápidos, eles precisam ser **estáveis**

   – Como qualquer dispositivo pode falhar, é importante os sistemas serem estáveis, incluindo **componentes redundantes**, que podem assumir o controle quando uma falha ocorre

Para pensar

O número de processadores embutidos vendidos a cada ano supera, e muito, o número de processadores para PC e até mesmo pós-PC.

Você pode confirmar ou negar isso com base em sua própria experiência?

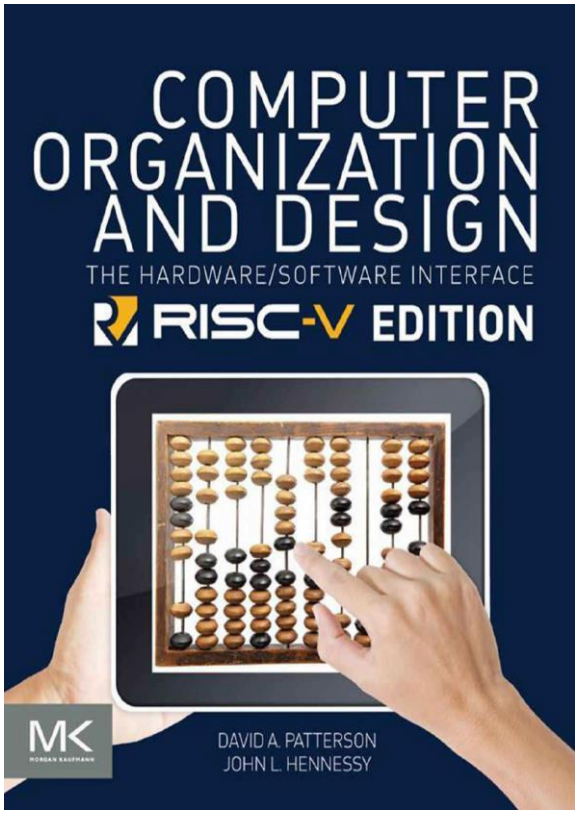
Para pensar

Tanto o **software** quanto o **hardware** podem afetar o desempenho de um programa. Você pode pensar em exemplo nos quais cada um dos fatores a seguir é o responsável pelo gargalo no desempenho?

- O algoritmo escolhido
- A linguagem de programação ou compilador
- O sistema operacional
- O processador

Bibliografia Básica

- Patterson, David A. & Hennessy, John L. "**Computer Organization and Design - The Hardware/Software Interface RISC V Edition**" Morgan Kaufmann Publishers Inc., San Francisco, CA, 2018**.**



Bibliografia Básica

- Patterson, David A. & Hennessy, John L. "**Computer Organization & Design: the hardware/software interface"**. 3nd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, 2005.

- Tanenbaum, A S **"Organização Estruturada de Computadores"** – Prentice Hall do Brasil 5ª edição, 2006.

Bibliografia Complementar

- Patterson, david & Hennessy, John L. "**Computer Architecture a Quantitative Approach**". 2nd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1996.

- Patterson, david & Hennessy, John L. "**Computer Organization & Design: the hardware/software interface**". 2nd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, 1998.

- Patterson, david & Hennessy, John L. "**Computer Organization & Design: the hardware/software interface**". 4nd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, 2012.

Bibliografia Complementar

- Stallings, William. “**Computer Organization and Architecture – Designing for Performance**”. 8° ed. Prentice Hall, Inc., New Jersey, 2010.

- Tanenbaum, A S “**Organização Estruturada de Computadores**” – Prentice Hall do Brasil 3o edição, 1990.

- Weber, Raul Fernando. “**Arquitetura de Computadores Pessoais**”. Sagra Luzzatto 1 edição, 2000.

- RISC-V Guide. https://mark.theis.site/riscv/