PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA

DEPARTAMENTO: *Fundamentos da Computação*
CURSO: *Ciência da Computação*
DISCIPLINA: *Organização e Arquitetura de Computadores II*
CÓDIGO: *4613H* **CRÉDITOS:** *04* **CARGA HORÁRIA:** *60 horas-aula*
VALIDADE: *a partir de 2006/I*
REQUISITOS:
Pré-Requisito: *4613F-04 Organização e Arquitetura de Computadores I*
Substituta: *46185-04 Arquitetura de Computadores I*

OBJETIVOS:

O cumprimento da disciplina busca dar ao aluno, ao final do semestre, condições de:

1. Utilizar mecanismos de avaliação de desempenho de arquiteturas de computadores;
2. Comparar arquiteturas de computadores sob o ponto de vista do desempenho, mediante emprego de padrões quantitativos de avaliação;
3. Identificar as estruturas fundamentais de conjuntos de instruções e linguagens de montagem de arquiteturas RISC, sendo capaz de empregar estes conceitos em programação destas arquiteturas;
4. Reconhecer as relações fundamentais existentes entre o hardware e o software em arquiteturas de computadores modernas em suas abstrações mais relevantes, organização do hardware, linguagem de montagem e linguagem de alto nível;
5. Identificar a relação entre linguagens de programação de alto nível e as estruturas de hardware em arquiteturas de computadores;
6. Utilizar e ter noções de como implementar programas básicos empregados na tradução e execução de programas escritos em linguagem de montagem, tais como montadores, ligadores e carregadores;
7. Compreender como os elementos da arquitetura e/ou da linguagem de máquina são efetivamente usados na execução de um programa escrito em linguagem de alto nível e como as variações na organização destes elementos influenciam a implementação de programas básicos tais como compiladores;
8. Identificar as principais formas de elaborar organizações de computadores que seguem o paradigma RISC: implementação monociclo, multiciclo e pipeline;
9. Dominar as técnicas básicas de projeto, controle e implementação de *pipelines* simples e superescalares em arquiteturas de computadores modernas.

EMENTA:

Relações entre linguagem de montagem e linguagem de alto nível. Arquitetura do conjunto de instruções: tipos, formatos, modos de endereçamento. Medidas de desempenho. *Pipeline*/Superescalar.

UNIDADE: 01 **Nº DE HORAS EM PERCENTUAL: 10%**
CONTEÚDO: Avaliação de Desempenho de Arquiteturas de Computadores

1. Introdução
   1.1. Definição de desempenho
   1.2. Tempo de resposta e vazão
   1.3. Desempenho relativo
2. Como medir desempenho
3. Relação entre métricas de desempenho
4. *Benchmarks*: programas para medir desempenho

PROCEDIMENTOS E RECURSOS:
Esta unidade é muito bem explorada no Capítulo 2 de Patterson [1-2] e no Capítulo 1 de Hennessy [3], que devem servir de base para o estudo. A idéia é prover os alunos com ferramental de base para avaliar arquiteturas do ponto de vista de desempenho. Definições de equações de cálculo quantitativo de desempenho temporal da execução de programas devem ser salientadas. Exercícios devem ser realizados pelos alunos, visando fixar os principais conceitos e unidades, tais como ciclos de relógio por instrução (CPI), instruções por segundo (IPS) e instruções de ponto flutuante por segundo (FLOPS). Os exercícios constantes nas bibliografias supra citadas formam bom material de apoio. Recomendam-se, além destes, exercícios extra-classe de pesquisa na Internet por dados sobre a caracterização de desempenho de arquiteturas conhecidas e comparações entre arquiteturas usando os conceitos e medidas aprendidos nesta unidade.

UNIDADE: 02 **Nº DE HORAS EM PERCENTUAL: 15%**
CONTEÚDO: Estudo de uma Arquitetura RISC
2.1 Definição da arquitetura
   2.1.1 Registradores acessíveis ao programador em linguagem de montagem
   2.1.2 Conjunto de instruções
   2.1.3 Formatos de instrução
   2.1.4 Modos de endereçamento
   2.1.5 Linguagem de montagem
2.2 Programação em linguagem de montagem na arquitetura

PROCEDIMENTOS E RECURSOS:
Esta unidade faz a ligação com disciplina anterior. A última unidade da disciplina anterior introduz conceitos para processadores RISC, diferenciando-os de processadores CISC. O objetivo desta unidade é revisar rapidamente a definição da arquitetura RISC, passando em seguida para a parte mais relevante, que consiste na prática de programação em linguagem de montagem da arquitetura.

UNIDADE: 03 **Nº DE HORAS EM PERCENTUAL: 25%**
CONTEÚDO: A Relação entre Linguagem de Alto Nível e Arquitetura

3.1 Linguagem de montagem versus linguagem de alto nível
3.2 Visão geral de sistemas de apoio à programação e execução de programas
- **3.2.1** Compiladores, montadores, ligadores, carregadores
- **3.2.2** Outros sistemas de apoio: processadores de macro e interpretadores

3.3 Modos de endereçamento
3.4 Avaliação de expressões complexas, lógicas e aritméticas
3.5 Implementação de estruturas de controle de fluxo e laços
3.6 Formas de acesso a estruturas de dados
- **3.6.1** Inteiros e caracteres
- **3.6.2** Vetores e matrizes
- **3.6.3** Ponteiros

PROCEDIMENTOS E RECURSOS:
Esta unidade tem por objetivo apresentar a relação existente entre a estrutura de linguagens de alto nível e a(s) respectiva(s) estrutura(s) em linguagem de montagem. Devem ser mostrados diversos exemplos de programação equivalentes nas duas linguagens. Deve-se também realizar exercícios de fixação, para que os alunos apreendam como se pode automatizar o processo de tradução de linguagens de alto nível para linguagem de montagem. Como exercício adicional, sugere-se a produção e o estudo de código em linguagem de montagem a partir do emprego de compiladores reais de linguagens de alto nível tais como gcc para o MIPS. A unidade deve iniciar com um estudo de algumas das principais ferramentas empregadas na tradução automatizada de código fonte em linguagem de montagem para código objeto. Deve-se também abordar ferramentas que automatizam o processo de transformação deste código objeto em um programa executável e sua carga em memória para posterior execução, com ênfase no estudo do processo de montagem do código objeto. Deve-se também enfatizar a conexão entre o passo de montagem e o anterior, de compilação do programa em linguagem de alto nível. Atenção deve ser dada para a existência de outros tipos de sistemas freqüentemente usados em conjunção com as ferramentas mais exploradas na Unidade, tais como processadores de macro e interpretadores. O texto clássico de Calingaert [4] deve ser citado e pode ser usado pelo ministrante, mas seu uso pelos alunos deve ser excluído devido à dificuldade que estes teriam em separar os conceitos ultrapassados daqueles que ainda fazem sentido no contexto atual de programação de sistemas.

UNIDADE: 04 **Nº DE HORAS EM PERCENTUAL: 20%**

CONTEÚDO: Organizações RISC sem Paralelismo a Nível de Instrução

4.1 Construção do bloco de dados.

4.1.1 Versão monociclo

4.1.2 Versão multiciclo

4.2 Construção do bloco de controle

4.2.1 Bloco de controle para versão monociclo

4.2.2 Bloco de controle para versão multiciclo

PROCEDIMENTOS E RECURSOS:

Esta unidade prepara a seguinte, que explora organizações que suportam a execução paralela de instruções e a as implicações desta característica na arquitetura. Aqui, deve-se familiarizar o aluno com a organização de um processador capaz de executar o conjunto de instruções de uma máquina RISC típica, tal como o MIPS. De uma organização que executa cada instrução possível em um ciclo, deve-se evoluir para uma organização multiciclo, salientando os inconvenientes da versão monociclo do ponto de vista prático. A versão multiciclo deve ser apresentada avançando a importância de pensar em viabilizar a paralelização da execução de instruções do processador, bem como os problemas que isto pode vir a gerar para o controle da organização.

UNIDADE: 05 **Nº DE HORAS EM PERCENTUAL: 30%**

CONTEÚDO: Organizações RISC com Paralelismo a Nível de Instrução

5.1 Conceitos básicos de pipeline

5.2 Vantagens de *pipelining* – vazão

5.3 Inconvenientes de *pipelining* – *hazards* (RAW, WAR, WAW)

5.4 Solução de problemas em *pipelines* – bolhas, adiantamento, predição de desvios

5.5 Modelo bloco de dados – bloco de controle e *pipelines*

5.6 Organização do bloco de dados para *pipelining*

5.7 Estágios

5.8 Representação gráfica

5.9 Controle de *pipelines*

5.10 Além de *pipelines* – organizações *superpipeline* e superescalares

PROCEDIMENTOS E RECURSOS:

Nesta Unidade, estuda-se o conceito de paralelismo a nível de instrução (em inglês, *Instruction Level Parallelism*, ou ILP) e como a organização do processador pode dar suporte ao conceito. A ênfase é na compreensão de *pipelines*. A abordagem aqui deve ser eminentemente baseada em estudos de caso tais como o processador MIPS, descrito no Capítulo 6 do livro de Patterson e Hennessy [1-2]. As vantagens e inconvenientes de usar técnicas de *pipelining* devem ser exploradas e o fato de praticamente qualquer processador moderno conter alguma forma de *pipeline* deve ser salientado, para evidenciar a importância de estudá-los. Recursos tais como o simulador SPIM devem ser empregados, como forma de ajudar na apreensão prática do

conceitos de ILP. Deve estar implícito que os conceitos de arquitetura de computadores vistos inicialmente na disciplina anterior (instruções, registradores, modos de endereçamento, programação em linguagem de montagem) devem ser revisitados à luz da exploração de um estudo de caso de processador com *pipelines*. Cabe uma comparação entre técnicas de construção, tradução e execução de programas em arquiteturas com e sem *pipelines*.

AVALIAÇÃO:

$$G1 = 0,3*P1 + 0,3*P2 + 0,2*T1 + 0,2*T2$$

Onde:
P1 – Prova compreendendo os conteúdos das unidades 1,2 e 3
P2 – Prova compreendendo os conteúdos das unidades 4 e 5
T1, T2 – Trabalhos práticos

BIBLIOGRAFIA:

- **BÁSICA:**

1. PATTERSON, D. A. & HENNESSY, J. L. "Organização e Projeto de Computadores - A interface Hardware/Software" - editora LTC, 2000, 551p.

- **COMPLEMENTAR:**

2. Patterson, D. A. & Hennessy, J. L. Computer organization and design: the hardware/software interface. Morgan Kaufmann Publishers, Inc. San Mateo, CA, 964p. 2nd Edition, 1998.

3. Hennessy, J. L. & Patterson, D. A. Computer architecture: a quantitative approach. Morgan Kaufmann Publishers, Inc. San Francisco, CA, 998p. 2nd Edition, 1996

4. Calingaert, P. Assemblers, compilers and program translation. Computer Science Press. 240p. 1979.

5. Rafiquzzaman, M. Microprocessors and Microcomputer-based System Design, CRC Press, Boca Raton, FL, 776p. 1995.

6. Mano, M. M. Computer System Architecture. Prentice-Hall, Englewood Cliffs, NJ, 525p. 1993.

7. Blaauw, G. A. & Brooks, Jr. F. P. Computer architecture: concepts and evolution. Addison-Wesley Longman, Inc. Reading, MA. 1213p. 1997.

Emitido em: 01/03/06 - 14:19	Carimbo e Assinatura da Unidade:
Página 5 de 6	

8. Kain, R. Y. Advanced computer architecture: a systems design approach. Prentice Hall, Englewood Cliffs, NJ, 907p. 1996.
9. Stallings, W. Computer organization and architecture: designing for performance. Prentice Hall, Upper Saddle River, NJ, 682p. 4th Edition, 1996.
10. Zargham, M. R. Computer architecture: single and parallel systems. Prentice Hall, Englewood Cliffs, NJ, 471p.1996.

- **SOFTWARE DE APOIO**

1. Simuladores de microprocessadores de domínio público, tais como:
   - SPIM – Simulador do processador MIPS
   - Simulador para a família de microcontroladores Intel MCS@51
   - Simulador para o microprocessador Motorola 68000.
   - Simulador para microprocessadores da família MCS@86 da Intel.