PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA

DEPARTAMENTO: Fundamentos da Computação
DISCIPLINA: *Arquitetura de Computadores I*
CÓDIGO: 46185 **CRÉDITOS:** 04 **CARGA HORÁRIA:** 60 horas-aula
REQUISITOS:

Pré-requisito:	4490R-04 - Circuitos Digitais (INF)
Requisito especial:	46183-04 - Organização de Computadores
Substituta:	46123-04 - 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. Dominar as técnicas básicas de projeto, controle e implementação de "pipelines" em arquiteturas de computadores modernas.
4. Dominar as principais técnicas vinculadas à execução de operações aritméticas em hardware, tanto para números naturais e inteiros quanto para números racionais.
5. 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.
6. Reconhecer as relações fundamentais existentes entre o hardware e o software em arquiteturas de computadores modernas.
7. Controlar a relação entre linguagens de programação de alto nível e as estruturas de hardware em arquiteturas de computadores.
8. 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 suas variações influenciam a implementação eficiente de compiladores.

EMENTA:

Revisão do modelo bloco de dados - bloco de controle. Dados e endereços. Instruções: tipos, formatos, modos de endereçamento. Acesso à memória: decodificação de endereços. Conjunto de instruções. Uma arquitetura típica. Linguagem de montagem. Princípios de programação em linguagem de montagem. Relações entre linguagens de montagem e linguagens de alto nível.

N° DA UNIDADE: 01 **N° HORAS AULA EM PERCENTUAL:** 15%
CONTEÚDO: Avaliação de Desempenho de Arquiteturas de Computadores

1.1 Introdução
- **1.1.1** Definição de desempenho
- **1.1.2** Tempo de resposta e "throughput"
- **1.1.3** Desempenho relativo

1.2 Como medir desempenho

1.3 Relação entre métricas de desempenho
1.4 “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], estes devem servir de base para o estudo. A idéia é prover os alunos com ferramental de base para avaliar arquiteturas dadas e comparar arquiteturas entre si, do ponto de vista de desempenho. Definições de equações de cálculo quantitativo de desempenho temporal da execução de programas devem ser salientados. Uma ampla gama de exercícios deve ser realizada pelos alunos, visando fixar os principais conceitos e unidades, tais como ciclos de relógio por instrução (CPI), instruções por segundo (IPS), instruções de ponto flutuante por segundo (FLOPS), etc. Os exercícios constantes nas bibliografias supra-citadas são formam bons conjuntos de material de apoio. Recomenda-se, além destes, passar 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.

N° DA UNIDADE: 02 **N° HORAS AULA EM PERCENTUAL:** 30%
CONTEÚDO: Pipelines – Estrutura Geral, Controle e Construção

2.1 Revisão de conceitos básicos de “pipeline”
- **2.1.1** Vantagens de “pipelining” – “throughput”
- **2.1.2** Inconvenientes de “pipelining” – “hazards”
- **2.1.3** Solução de problemas em “pipelines”
  - **2.1.3.1** Bolhas
- **2.1.4** Modelo bloco de dados – bloco de controle e “pipelines”

2.2 Construindo um bloco de dados para “pipelining”
- **2.2.1** Estágios
- **2.2.2** Representação gráfica

2.3 Controle de “pipelines”
- **2.3.1** Instruções: tipos, formatos e modos de endereçamento
- **2.3.2** Acesso a memória em arquiteturas “pipeline”
- **2.3.3** Efeitos de “pipelining” na linguagem de montagem
- **2.3.4** Efeitos de “pipelining” em linguagens de alto nível

2.4 “Hazards” de dados e “forwarding”
2.5 “Hazards” de dados e bolhas
2.6 “Hazards” devido a saltos
2.7 Exceções em “pipelines”

PROCEDIMENTOS E RECURSOS:

Nesta Unidade, estuda-se o conceito de “pipeline” em detalhe, aprofundando a introdução vista na disciplina de Organização de Computadores. 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. Aconselha-se que 50% da nota de Trabalho Prático provenha de um trabalho de implementação envolvendo “pipelines”, tal como a síntese e simulação de um processador “pipeline” simples em VHDL. Recursos tais como o simulador SPIM devem ser empregados como forma de ajudar na apreensão prática do conceito de “pipeline”. Aqui, está implícito que os conceitos de arquitetura de computadores vistos inicialmente

na disciplina de Organização de Computadores (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”.

N° DA UNIDADE: 03 **N° HORAS AULA EM PERCENTUAL:** 20%
CONTEÚDO: Aritmética Computacional Avançada

3.1 Aritmética básica para implementação em sistemas digitais
- **3.1.1** Aritmética de naturais – soma e multiplicação
- **3.1.2** Aritmética de inteiros – soma, subtração, multiplicação e divisão

3.2 Aritmética de ponto flutuante : o padrão IEEE-754
- **3.2.1** Introdução ao padrão IEEE-754
- **3.2.2** Operações aritméticas no padrão IEEE-754
- **3.2.3** Arredondamento e valores não-numéricos

PROCEDIMENTOS E RECURSOS:

Esta Unidade dedica-se a discutir técnicas de implementação em hardware de circuitos digitais que implementam as operações aritméticas (soma, subtração, multiplicação e divisão) sobre números naturais, inteiros e racionais. As representações de números devem ser inicialmente estudadas, seguindo-se após para a apresentação de sistemas digitais que as manipulam. O texto básico é o Capítulo 4 das referências [1] ou [2].

N° DA UNIDADE: 04 **N° HORAS AULA EM PERCENTUAL:** 15%
CONTEÚDO: Sistemas de Apoio à Programação e Execução de Programas

4.1 Visão geral de sistemas de apoio à programação e execução de programas
- **4.1.1** Código objeto
- **4.1.2** Código em linguagem de montagem
- **4.1.3** Desvantagens de linguagens de montagem

4.2 Montadores
- **4.2.1** Rótulos
- **4.2.2** Tipos de montadores
- **4.2.3** Estruturas de dados básicas: tabelas de instruções e tabelas de símbolos
- **4.2.4** Facilidades adicionais em linguagens de montagem – diretivas e pseudo-instruções

4.3 Ligadores
4.4 Carregadores
4.5 Outros sistemas de apoio: processadores de macro e interpretadores

PROCEDIMENTOS E RECURSOS:

Esta Unidade deve prover um estudo de algumas das principais ferramentas empregadas na tradução automatizada de código fonte em linguagem de montagem para código objeto. Além disso, deve-se abordar também ferramentas que automatizam o processo de transformação deste código objeto em um programa executável e sua carga em memória para posterior execução. A ênfase deve ser maior 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. Aconselha-se que os alunos realizem um trabalho prático que represente os 50% restantes da nota TP no escopo de montadores e/ou ligadores e/ou carregadores. Atenção deve ser dada para salientar 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 de separar os conceitos ultrapassados daqueles que ainda fazem sentido no contexto atual de programação de sistemas.

N° DA UNIDADE: 05 **N° HORAS AULA EM PERCENTUAL:** 20%
CONTEÚDO: A Relação entre Arquitetura e Programação em Linguagens de Alto Nível

5.1 Linguagem de montagem versus linguagem de alto nível
5.2 Modos de endereçamento
5.3 Avaliação de expressões complexas, lógicas e aritméticas
5.4 Implementação de estruturas de controle de fluxo e laços
5.5 Formas de acesso a estruturas de dados complexas (vetores, matrizes, registros e 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 em ambas as 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, preparando o terreno para a disciplina de Compiladores I, do semestre a seguir. Como exercícios adicionais, sugere-se a produção e o estudo de código em linguagem de montagem a partir do emprego de compiladores comerciais de linguagens de alto nível tais como C/C++ e Pascal.

AVALIAÇÃO:

$$G1 = (P1 + P2 + 2TP) / 4$$

Onde:

P1 – Abrange às Unidades 01, 02 e 03
P2 - Abrange às Unidades 03, 04 e 05
TP - Trabalho Prático
Obs: A nota de Trabalho Prático deve provir de uma ponderação das avaliações dos trabalhos práticos mencionados nos itens Procedimentos e Recursos das Unidade 02 e 03.

BIBLIOGRAFIA:

- **LIVRO(S) TEXTO**

1. Patterson, D. A. & Hennessy, J. L. Organização e projeto de computadores: a interface hardware/software. LTC – Livros Técnicos e Científicos S.A. Rio de Janeiro, RJ, 551p. Segunda Edição, 2000.

- **LIVRO(S) REFERENCIADO(S)**

1. 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.

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

3. Calingaert, P. Assemblers, compilers and program translation. Computer Science press. 240p. 1979.

- **OUTRAS REFERÊNCIAS**

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

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

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

4. Kain, R. Y. Advanced computer architecture: a systems design approach. Prentice Hall, Englewood Cliffs, NJ, 907p. 1996.

5. Stallings, W. Computer organization and architecture: designing for performance. Prentice Hall, Upper Saddle River, NJ, 682p. 4th Edition,1996.

6. Zargham, M. R. Computer architecture: single and parallel systems. Prentice Hall, Englewood Cliffs, NJ, 471p.1996.

- **SOFTWARE DE APOIO**

1. Active-HDL – Um simulador para a linguagem de descrição de hardware VHDL (50 licenças).

2. 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.