Ensino de Arquitetura de Computadores com Enfoque na Interface Hardware/Software

Roberto A Hexsel & Renato Carmo
Departamento de Informática
Universidade Federal do Paraná
CEP 81531-990 — Curitiba, PR, Brasil
{roberto,renato}@inf.ufpr.br

Resumo

Apresentamos[1] *e discutimos o ensino de Arquitetura de Computadores com enfoque na Interface Hardware/Software. Caracterizamos este enfoque, apresentamos uma proposta de conteúdo para um curso de graduação em Computação na linha de formação de Arquitetura de Computadores e discutimos a implementação de tal proposta no Bacharelado em Ciência da Computação da Universidade Federal do Paraná.*

1. Introdução

O núcleo da formação de um profissional de Computação contém uma linha de formação em *Sistemas de Computação* (ou simplesmente *Sistemas*), onde se encaixam disciplinas como *Sistemas Operacionais*, *Arquitetura de Computadores* etc.

Podemos destacar duas abordagens possíveis na formação em *Sistemas*, que chamaremos aqui de *enfoques*. O primeiro, que poderíamos chamar de "enfoque formativo", é aquele que se propõe dar ao aluno uma base sólida o suficiente para permitir o projeto, desenvolvimento e a avaliação de sistemas computacionais relativamente complexos. O segundo, que poderíamos chamar de "enfoque informativo", é aquele em que pretende-se apenas informar o aluno sobre a infraestrutura necessária para a execução de programas em computadores.

O enfoque formativo, por sua vez, pode assumir diferentes perfis, de acordo com o objetivo a que se propõe o curso de graduação como um todo. Em linhas gerais, podemos distinguir duas opções. Por um lado, o caso de cursos de engenharia e assemelhados, em que o objetivo é formar projetistas de sistemas, com forte embasamento em hardware. Do outro lado estaria o caso tipificado pelos cursos de graduação em Ciência da Computação, nos quais a formação do aluno se caracteriza por uma forte ênfase em software, tanto em programação quanto em análise e projetos de sistemas. Neste último caso, o objetivo da formação em *Sistemas* não é o de formar projetistas de hardware, mas sim de capacitar programadores a usar eficaz e eficientemente o hardware. Este tipo de capacitação vem sendo objeto de demanda crescente por parte do mercado de trabalho, à medida que se expande o mercado de eletrônica embarcada e embutida.

Esta última abordagem do ensino da linha de *Sistemas* é o que chamamos aqui de *"enfoque na Interface Hardware/Software"*. Defendemos a conveniência de tal abordagem em cursos de Computação porque a experiência tem mostrado aos autores que tal enfoque cumpre o objetivo de formar programadores capacitados a lidar eficientemente com sistemas digitais. Ao colocar o foco do estudo na interface Hardware/Software, este enfoque permite ao estudante associar com relativa facilidade o comportamento abstrato dos programas que escreve com sua execução em dispositivos concretos.

Não por acaso, esta é exatamente a abordagem adotada por *Hennessy* e *Patterson* nos textos que se impuseram nos últimos anos como de referência obrigatória no assunto, *Computer Organization & Design: The Hardware/Software Interface* [10] e *Computer Architecture: A Quantitative Approach* [4].

Este artigo está organizado da seguinte forma. O enfoque na interface Hardware/Software como opção didática é discutido Seção 2. A Seção 3 discute os conteúdos cuja abordagem consideramos necessária para uma sólida formação do aluno de Computação

[1] Apresentado no Workshop sobre Educação em Arquitetura de Computadores – WEAC 2006, realizado em conjunto com SBAC-PAD 2006 e WSCAD 2006, Ouro Preto, 17out06.

na linha de *Sistemas*. Nossa experiência com relação à implementação de tal proposta no *Bacharelado em Ciência da Computação* da *Universidade Federal do Paraná* ao longo dos últimos dez anos é apresentada na Seção 4. Concluímos o texto na Seção 5. Os programas e a bibliografia das disciplinas mencionados na Seção 4 estão listados no Apêndice A.

2 O Enfoque na Interface Hardware–Software como Opção Didática

Quando se considera um sistema digital que seja programável segundo o modelo de Von Neumann, a *interface* entre hardware e software é definida por dois componentes: (i) o conjunto de instruções do processador, que inclui os registradores visíveis ao programador, e (ii) os mecanismos de acesso aos recursos controlados pelo sistema operacional, tais como a unidade de gerenciamento de memória, temporizadores etc.

Na prática, a organização de um processador depende fundamentalmente do conjunto de instruções que ele executa, ao passo que a implementação da microarquitetura depende de vários de fatores tecnológicos e econômicos. Numa determinada faixa de preço, estes fatores tecnológicos e econômicos influenciam a complexidade da organização (segmentação, escalaridade), mas são a facilidade de programação e a eficiência da execução do código o que determina o sucesso de uma particular implementação.

Do ponto de vista do projetista do processador, o conjunto de instruções pode ser visto como uma *"Application Programming Interface"* (*API*) a implementar. Do mesmo modo, do ponto de vista do programador, o conjunto de instruções pode ser visto como uma *API* implementada. Colocar o foco do estudo nessa *API*, encarada como especificação abstrata, é o que chamamos de *"Ensino com Enfoque na Interface Hardware/Software"*. Tal enfoque estabelece uma simetria entre as visões do projetista e do programador que é de grande utilidade do ponto de vista didático. Isto é especialmente verdade no caso do aluno de Computação, que nas diferentes disciplinas do seu currículo é colocado alternadamente tanto na situação de implementador como na de usuário de diferentes *API*s.

Uma alternativa ao enfoque defendido aqui é o que poderíamos chamar de *"enfoque casuístico"*, como é o caso de textos como Tanembaum [16] ou Stallings [15]. Nestes textos, são apresentados tópicos de hardware tais como o projeto do circuito de dados e da hierarquia de memória, sem que o conjunto de instruções a ser implementado tenha sido estudado com a profundidade necessária para que o aluno seja conduzido a relacionar as características do conjunto de instruções às técnicas de implementação mais adequadas.

Uma característica comum neste tipo de texto é a tentativa de descrever a maioria das arquiteturas em uso – são apresentados vários conjuntos de instruções – sem a devida atenção às razões para as diferenças entre as variações apresentadas. Para muitos alunos, a variedade de arquiteturas com diferentes possibilidades de implementação acaba por causar confusão porque a apresentação tende a ser superficial em função da grande quantidade e variedade de casos a estudar. Este enfoque acaba por resultar mais informativo do que formativo.

Acreditamos que seja mais adequado a um curso introdutório o estudo de um menor número de tópicos discutidos em maior profundidade, com as relações entre eles claramente estabelecidas. Numa segunda disciplina, depois que os fundamentos tenham sido solidamente apreendidos, o estudo comparativo de diversas arquiteturas e organizações torna-se muito mais proveitoso.

3 Conteúdos das Disciplinas

A maioria das disciplinas do início de um curso de Computação compreende conceitos que são completa novidade para o aluno. A apreensão destes conceitos é facilitada quando são apresentados de forma adequadamente simplificada, sem que as complexidades inerentes ao assunto sejam ignoradas. Os textos de *Hennessy* e *Patterson* mencionados anteriormente representam uma das melhores combinações de simplicidade e cobertura. Neles, a arquitetura *MIPS* é usada como objeto de estudo, o que é particularmente interessante por ser uma arquitetura simples e elegante, sem as idiossincrasias encontradas em processadores como os da família *x86*, por exemplo.

Como discutido na Seção 2, a premissa fundamental do *enfoque na interface Hardware/Software* é que o conjunto de instruções de um processador define uma *API* tanto para o seu projetista quanto para o programador. Por conta disso, é conveniente separar os conteúdos em dois conjuntos que vamos chamar de *conteúdos de hardware* e *conteúdos de software*. A Figura 1 mostra um esquema simplificado dos conteúdos indicando a ordem em que devem ser apresentados. As setas indicam as principais dependências entre os dois conjuntos de conteúdos.

Alguns conhecimentos preliminares são necessários a ambos os conjuntos, tais como representação binária, suas diferentes codificações e rudimentos de Álgebra de Boole. Após os conhecimentos preliminares, os conteúdos devem observar a uma certa seqüência.

Pelo lado dos *conteúdos de software*, a programação

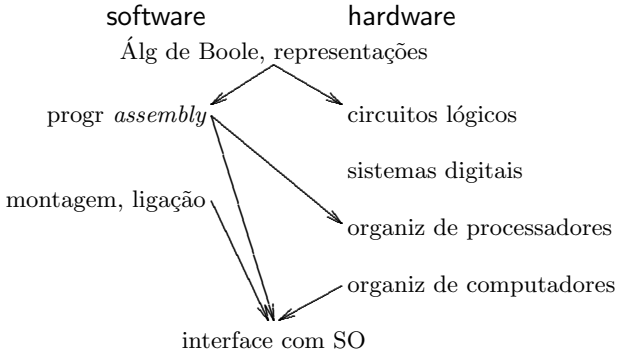

Figura 1. Conteúdos necessários e suas dependências

em *assembly* apresenta a *API* ao aluno e o exercita na escrita de programas com baixo nível de abstração. A seguir vem o estudo de montadores, ligadores e carregadores. Neste ponto, é apresentada uma versão primitiva de um compilador e seu respectivo sistema de suporte à execução de programas ("run-time system"), bem como as bibliotecas de sistema. Estes últimos, por sua vez, constituem uma parte da interface com o sistema operacional.

Pelo lado dos *conteúdos de hardware*, a seqüência inicia com lógica combinacional e seqüencial, quando as portas lógicas e "flip-flops" são apresentados. Estes elementos são então usados na construção de sistemas digitais relativamente complexos tais como unidades de lógica e aritmética, e sistemas baseados em micro-controladores. A partir daí é possível apresentar a organização de um processador como um circuito de dados ("datapath") que interliga vários sub-sistemas, todos obedecendo aos comandos de um circuito controlador. A organização de um computador, por sua vez, também pode ser mostrada como um circuito de dados que interliga vários componentes. O sub-sistema de entrada e saída, em particular, permite apresentar outra parte da interface com o sistema operacional.

Observando a Figura 1, note que, por exemplo, programação em *assembly* tem como efeito colateral o estudo da *API* do processador, cuja implementação é o objeto de estudo da organização de processadores.

4 A Experiência no BCC — UFPR

A Figura 2 mostra as disciplinas da linha de *Sistemas* do *Bacharelado em Ciência da Computação* da *Universidade Federal do Paraná*[2] que implementam os conteúdos discutidos na Seção 3, ordenadas por semestre do curso. As disciplinas mostradas em itálico e entre parênteses tem dependências fracas com as outras no conjunto. No lado direito de cada coluna colocamos nomes abreviados de cada disciplina, para melhor fluência do texto.

No currículo do BCC da UFPR não há nenhuma disciplina de Física e nem de Eletrônica. Isso influencia as disciplinas da área de Sistemas de três maneiras importantes: (i) o nível de abstração com que o projeto de circuitos digitais pode ser abordado – o comportamento de portas CMOS é modelado e apresentado a partir da analogia com um sistema hidráulico porque o aluno não tem a noção de capacitância; (ii) por conta do nível de abstração, o aluno tem dificuldade em lidar com a noção de tempo de propagação de sinais e com o fato de que os eventos não são instantâneos, já que sua percepção como usuário é de que todas as operações internas ao computador são instantâneas; e (iii) o aluno não passa pelo treinamento de resolver baterias de problemas de Física, como é típico em cursos de Engenharia. Juntos, estes três fatores condicionam a apresentação dos conteúdos, os trabalhos práticos, e o encadeamento dos conteúdos nas disciplinas.

A apresentação dos *conteúdos de software* inicia em *Máquinas* que abarca Álgebra de Boole, as várias formas de representação da informação, e programação em *assembly*. A programação em *assembly* tem a finalidade adicional de prover uma alternativa para o aluno que encontra dificuldade na programação em linguagens de alto nível, assunto de outra disciplina do primeiro período. *SwBásico* amplia o conteúdo de *Máquinas* através da investigação das interfaces entre processador, montadores, ligadores e ambiente de execução. Nesta disciplina a programação em *assembly* explora em profundidade os mecanismos de suporte a funções e de alocação de memória em *heap* e pilha. O conteúdo de *Sistemas Operacionais* depende de *Arquitetura* para a compreensão dos conceitos de processo, memória virtual e E/S.

A apresentação dos *conteúdos de hardware* inicia em *Circuitos* que abarca lógica combinacional e seqüencial, apresentando o material como uma linguagem para descrever algoritmos em hardware. *Projetos* amplia este escopo dando ênfase à implementação, no nível das portas lógicas, de um microprocessador simples. O projeto é fornecido ao aluno como um diagrama em blocos do processador, com uma especificação mais ou menos detalhada dos componentes, bem como uma definição da semântica das instruções. Sua tarefa é implementá-lo num simulador. *Arquitetura* revisita a *API* cobrindo o projeto de processadores segmentados (*pipeline*), hierarquia de memória e dispositivos de entrada e saída.

[2] A grade curricular do BCC pode ser vista em `http://www.inf.ufpr.br/bcc/0_Curso/Grade_Curricular/index.html` .

sem.	Disciplina (sw)	*abrev.*	Disciplina (hw)	*abrev.*
1	Máquinas Programáveis	*Máquinas*		
2			Circuitos Lógicos	*Circuitos*
3	Software Básico	*SwBásico*	Projetos Digitais e Microprocessadores	*Projetos*
4			Organização e Arquitetura de Computadores	*Arquitetura*
5	Sistemas Operacionais	*SO*	*(Redes de Computadores I)*	
6	*(Construção de Compiladores)*		Tópicos em Arquitetura de Computadores	*Tópicos*

Figura 2. Disciplinas da Linha de Sistemas

Em *Tópicos* (disciplina optativa) estuda-se com mais profundidade o conteúdo de *Arquitetura*, abrangendo processadores super-escalares e versões detalhadas do projeto de hierarquia de memória. *Redes I* depende dos conteúdos de E/S apresentados em *Arquitetura*.

O encadeamento dos conteúdos faz que o nível de abstração seja crescente ao longo do tempo. Por exemplo, nos *conteúdos de hardware* circuitos simples são construídos no segundo semestre em *Circuitos*. Circuitos relativamente complexos são construídos no terceiro em *Projetos*. Finalmente, processadores complexos são modelados no quarto semestre em *Arquitetura*. O aluno interessado tem ainda a oportunidade de avaliar o desempenho de um processador complexo em *Tópicos*, que é a disciplina optativa da linha de *Sistemas* ofertada no curso.

Tipicamente em *Circuitos* são implementados dois projetos, uma unidade de lógica e aritmética de 8 ou 16 bits e um multiplicador seqüencial (8×8 bits). Estes componentes farão parte do microprocessador que será construído no semestre seguinte em *Projetos*. Este microprocessador (chamado *Mico* [5]) por sua vez, é uma versão simplificada do *MIPS R3000*, que é o objeto de estudo de *Arquitetura*. Existe, portanto, um forte encadeamento entre conteúdos e trabalhos práticos, ao mesmo tempo em que se expõe o aluno às noções de abstração, encapsulamento e re-utilização de componentes e sub-sistemas.

Os trabalhos práticos de *Circuitos* e *Projetos* são implementados no simulador digital *TKgate* [3] que permite "construir" circuitos sofisticados a partir da edição do esquemático e efetuar simulações funcionais e de temporização através de um analisador lógico. Considerando que não existem disciplinas de Eletrônica no curso, esta abordagem tem sido bastante eficaz para desenvolver e exercitar a capacidade de construir dispositivos complexos a partir de componentes simples, bem como a capacidade de abstração. Os trabalhos práticos de *Arquitetura* são implementados com a linguagem de descrição de arquiteturas *ArchC* [11, 2]. *ArchC* permite a descrição de um conjunto de instruções em *C++*, pela especificação da sintaxe das instruções, essencialmente definindo-se a linguagem *assembly* do processador a ser modelado. A partir do conjunto de instruções podem ser codificados simuladores mais ou menos detalhados.

Com relação ao nível de abstração nos trabalhos práticos, nas disciplinas de *Circuitos* e *Projetos* os circuitos são "implementados" no nível de portas lógicas, e são analisados no nível de sinais lógicos e tempo de propagação. Tanto a implementação quanto a análise são efetuadas de maneira concreta. Nas disciplinas de *Arquitetura* e *Tópicos* (optativa), o aluno emprega o *ArchC*, que permite trabalhar de forma mais ou menos abstrata, dependendo do nível de detalhe do simulador, que pode ser puramente funcional ou simular detalhadamente a execução de instruções num processador segmentado com circuitos de adiantamento e previsão de desvios, por exemplo.

5 Conclusão

Apresentamos e discutimos o ensino de Arquitetura de Computadores com enfoque na Interface Hardware/Software. Caracterizamos este enfoque, apresentamos uma proposta de conteúdos e de métodos para a linha de formação de Arquitetura de Computadores que julgamos adequada a um curso de graduação em Computação. Este enfoque se propõe oferecer ao aluno uma base sólida o suficiente para permitir o projeto, desenvolvimento e a avaliação de sistemas computacionais relativamente complexos, direcionado a formar programadores capazes a usar eficaz e eficientemente o hardware. Discutimos a implementação de tal proposta no Bacharelado em Ciência da Computação da Universidade Federal do Paraná.

Agradecimentos Os autores agradecem a colaboração dos colegas que tem contribuído para o desenvolvimento da linha de Sistemas no BCC: Armando L N Delgado, Bruno Muller Jr, Cristina D Murta, Elias P Duarte Jr, Heinz A Niederheitmann Jr, Heraldo M F Madeira, Luis Allan Kunzle, Luis Carlos E de Bona, Luiz Carlos P Albini, Nelson Suga e Wagner N Zola.

A Organização dos Conteúdos

Os números entre parênteses são uma sugestão de seqüência e extensão da abordagem de cada tópico. Um semestre dura 16 semanas, em média, e por isso os programas contém 15 itens.

Máquinas Programáveis

Objetivo Fornecer ao aluno conhecimentos básicos sobre sistemas computacionais. Fornecer conhecimento elementar sobre a organização de um computador e o relacionamento entre a máquina, sua linguagem de baixo nível, e as linguagens de alto nível usadas para programá-la. Sedimentar a compreensão, através da programação em linguagem de máquina, dos conceitos básicos de programação: variáveis, atribuição, decisão, iteração, funções.
Enfoque As primeiras 5 semanas devem prover ao aluno uma série de conhecimentos básicos que lhe serão úteis. Estes incluem as diferentes formas de representação e de codificação da informação. A segunda parte do curso deve ser dedicada ao ensino de programação "em concreto" e a fazer a ligação entre um "programa" e a máquina que o executa. Ao final do semestre deve ser discutida (de forma superficial e através de exemplos simples) a tradução de um programa em linguagem da alto nível para a linguagem de baixo nível. Os alunos escrevem alguns (4-6) programas simples em linguagem de máquina.
Programa (1) Organização de um computador: processador, memória, periféricos; (2) funcionamento do sistema operacional, compilador e aplicativos; (3) representação de dados, sistemas de numeração, conversão de bases; (4) representação digital de dados: tipos de dados, armazenamento; (5) Álgebra de Boole, funções lógicas básicas; (6) definição de linguagem de programação; (7) modelo do processador, registradores, operadores, instruções; (8) linguagem de máquina; instruções de lógica e aritmética; (9) linguagem de máquina; controle de fluxo; (10) linguagem de máquina; acesso à memória; (11,12) linguagem de máquina; modos de endereçamento; (13) implementação de construções de alto-nível em linguagem de máquina; (14) linguagem de máquina; suporte a subrotinas; (15) exemplos de programas em linguagem de máquina.
Bibliografia Hennessy & Patterson, *Computer Organization & Design* (capítulo 2) [10].
Ferramenta spim, xspim.

Software Básico

Objetivo Fornecer ao aluno conhecimentos sobre sistemas de suporte à execução de programas. Isso se dá através do estudo, projeto e programação de programas controladores de dispositivos, sistemas de interrupções, e sistemas operacionais para microprocessadores e sistemas embarcados. Além disso, é necessário o estudo montadores e de seus sistemas auxiliares, como ligador e carregador.
Enfoque No início do semestre deve-se expandir o material visto em Máquinas Programáveis, aprofundando-se o estudo da programação em linguagem de máquina, com ênfase no suporte a sub-rotinas, e na interface de chamadas de sistema, servindo como ligação com os conteúdos de Sistemas Operacionais e Compiladores. Nos trabalhos práticos, os alunos devem escrever um ou mais programas que façam uso intensivo deste material. O professor deve tomar extremo cuidado para que os alunos não se habituem à *programação por tentativa e erro*.
Programa (1) Sistema de suporte a execução de programas: compiladores, ligadores, carregadores, bibliotecas; (2) estrutura de um montador: codificação simbólica; (3-4) programa montador: geração de código de máquina; uso de macros; (5) segmentação e ligação de programas; (6-8) bibliotecas de funções; construção e uso; (9-13) programa carregador; mapas de memória; segmentos de código e dados; (14-15) sistema de interrupções.
Bibliografia Levine, *Linkers & Loaders* [9]; Hennessy & Patterson, *Computer Organization & Design* (cap. 2, apên. A) [10].
Ferramentas gcc, gpp, gdb, as, ld.

Circuitos Lógicos

Objetivo Fornecer ao aluno sólida base em Sistemas Digitais. Capacitá-lo a analisar e determinar o comportamento de sistemas digitais simples tais como unidade de lógica e aritmética e circuitos para multiplicação. Capacitá-lo a projetar sistemas digitais simples.
Enfoque Apresentar Circuitos Lógicos como uma linguagem para descrever algoritmos a serem implementados em hardware. Demonstrar as técnicas de projeto dos blocos básicos que são usados para construir computadores completos. O relacionamento entre o hardware construído e seu uso em processadores (ULA e multiplicador) deve ser apontado sempre que possível, para colocar o conteúdo em um contexto mais amplo. Enfatizar que os projetos devem ser concebidos e executados *no papel* e que o simulador deve ser usado somente como uma ferramenta de verificação e teste dos

componentes, para evitar que os alunos o usem para *projetar por tentativa e erro*.
Programa (1) Revisão de Álgebra de Boole; circuitos de chaves; (2) funções lógicas e sua implementação; (3) simplificação de funções e Mapa de Karnaugh; (4) circuitos combinacionais: codificadores e decodificadores; (5) circuitos combinacionais: multiplexadores e demultiplexadores; (6) tempo de propagação de sinais; corridas, riscos; (7) somador rápido; Unidade de Lógica e Aritmética; (8) propagação de sinais através de circuitos; básculas SR, JK; (9) memória: flip-flops D, T, JK; (10) temporização e circuitos de relógio; (11) memória: flip-flops e registradores; (12) contadores simples; (13) circuitos seqüenciais: contadores; (14) circuitos seqüenciais: controladores e seqüenciadores; (15) multiplicador seqüencial.
Bibliografia Tocci, Widmer, *Sistemas Digitais* [17]; Hennessy & Patterson, *Computer Organization & Design* (capítulo 3) [10]; Katz, *Contemporary Logic Design* [7].
Ferramenta TKgate.

Projetos Digitais e Microprocessadores

Objetivo Capacitar o aluno a conceber, projetar e implementar Sistemas Digitais de média complexidade (processador RISC de 8-16 bits). Fornecer os rudimentos de projeto de sistemas baseados em microprocessadores.
Enfoque A ênfase nesta disciplina é na implementação de um computador idealizado a partir da sua linguagem de programação e dos componentes projetados em Circuitos Lógicos. O professor deve fornecer a arquitetura e a organização completas do computador e os alunos devem projetar e implementar os circuitos necessários. O trabalho prático é o trabalho mais extenso que os alunos enfrentam no curso (de 50 a 100 horas) e portanto deve-se enfatizar a necessidade de planejar e organizar as tarefas, especialmente a fase de testes. Tipicamente, o programa usado para o teste da implementação é recursivo (cálculo do fatorial ou Série de Fibonacci).
Programa (1) implementação em CMOS; fabricação de circuitos CMOS; (2) projetos combinacionais complexos; (3) projetos seqüenciais complexos; (4) circuitos de memória RAM; (5) organização de um microcomputador; interligação entre os componentes; (6) linguagem de máquina: instruções, formatos e operações; (7) linguagem de máquina: controle de fluxo, funções, recursão; (8) linguagem de máquina e sua implementação no processador; (9) circuito de dados; (10) subsistema de memória, RAM, ROM, endereçamento; (11) subsistema de entrada/saída, endereçamento, barramentos de E/S; (12) projeto do controlador; micro-programação; (13) interrupções; projeto de barramentos; (14) interfaces paralela e serial; (15) interfaces analógico-digitais; contadores e temporizadores;
Bibliografia Hexsel, *Sistemas Digitais e Microprocessadores* [6]. Katz, *Contemporary Logic Design* [7]. Hennessy & Patterson, *Computer Organization & Design* (caps. 4 e 5) [10];
Ferramenta TKgate.

Organiz. e Arquitetura de Computadores

Objetivo Fornecer ao aluno conhecimentos básicos sobre Arquitetura de Computadores e sobre as técnicas de projeto de máquinas. Capacitá-lo a avaliar e comparar diferentes arquiteturas. Capacitá-lo a apontar e diagnosticar problemas relacionados ao desempenho de sistemas ou sub-sistemas, bem como a prescrever soluções para otimizar a utilização e o desempenho de sistemas computacionais.
Enfoque Apresentar o conjunto de instruções como uma API a ser implementada. Apresentar os componentes de um computador e relacionar a implementação com a execução de programas, e o impacto de diferentes técnicas de projeto e construção no desempenho global do sistema. Discutir as diferenças no projeto de computadores para as diferentes aplicações (estação de trabalho, sistema embarcado). O trabalho prático deve ser usado para demonstrar o uso de diferentes níveis de abstração, e relacionar custo de implementação com o ganho de desempenho que pode ser obtido.
Programa (1) linguagem de máquina do MIPS; conjunto de instruções; (2) instruções de acesso à memória, suporte à funções; (3) aritmética de ponto fixo e flutuante: soma, multiplicação e divisão; (4) implementação com ciclo longo; (5) implementação multiciclos; (6) avaliação de desempenho, métricas, formas de comparação; (7) circuito de dados segmentado, controle da segmentação; (8) melhorias na segmentação, dependências de dados e de controle; (9) super-segmentação, super-escalaridade, arquitetura VLIW; (10) hierarquias de memória, memória cache, técnicas de projeto; (11) memória virtual e física; (12) implementações de tabela de páginas, cache de mapeamentos, TLB; (13) hierarquias de memória: caches, TLB, memória principal e discos; (14) dispositivos de entrada e saída; programação, interrupções; DMA; (15) exemplos de arquiteturas de alto desempenho da última geração;
Bibliografia Hennessy & Patterson, *Computer Organization & Design* [10]; Stallings, *Computer Organization and Architecture: Designing for Performance* [15].
Ferramenta ArchC.

Sistemas Operacionais

Objetivo Fornecer ao aluno conhecimentos básicos sobre Sistemas Operacionais e sobre as técnicas básicas de projeto destes. Capacitá-lo a avaliar e comparar diferentes Sistemas. Capacitá-lo a apontar e diagnosticar problemas relacionados ao desempenho de sistemas ou sub-sistemas, bem como a prescrever soluções para otimizar o uso e desempenho de sistemas computacionais.
Enfoque Identificar os conceitos abstratos (processo, ambiente de execução, memória virtual) com as interfaces de hardware que permitem a sua implementação. Os trabalhos práticos devem envolver a programação de sistemas concorrentes para que os alunos percebam as dificuldades inerente à programação concorrente.
Programa (1) Introdução; dispositivos de hardware e de software; (2) organização de um sistema operacional; serviços; (3) processos; (4) representação de processos; escalonador; processos leves; (5) comunicação entre processos; (6) sincronização entre processos; alocação de recursos; impasses; (7) gerenciamento de memória; localidade; conjuntos de trabalho; (8) paginação; (9) segmentação; (10) gerenciamento de entrada e saída; tipos de dispositivos; (11) sistema de arquivos; (12) operações em arquivos; memória secundária; (13) segurança e proteção; (14) sistemas distribuídos; sistemas de arquivos distribuídos; (15) chamadas de procedimentos remotos.
Bibliografia Silberschatz, Galvin, Gagne, *Operating System Concepts* [12]; Stallings, *Operating Systems - Internals and Design Principles* [14].

Redes de Computadores I

Objetivo Fornecer ao aluno conhecimentos básicos sobre Redes de Dados e sobre as técnicas básicas e tecnologias empregadas em redes de dados e redes locais. Capacitá-lo a avaliar e comparar diferentes sistemas de comunicação. Capacitá-lo a apontar e diagnosticar problemas relacionados ao desempenho de sistemas ou sub-sistemas, bem como a prescrever soluções para otimizar o uso e desempenho de sistemas de comunicação.
Programa (1) Princípios de redes de computadores; Modelo ISO/OSI; (2) conceitos teóricos sobre sinais, séries de Fourier, capacidade do canal; (3) codificação digital/digital: NRZ, bipolar, *bi-phase*, *scrambling*, *spread spectrum*; (4) codificação digital-analógica: ASK, FSK, PSK, QPSK, QAM, TCM; (5) digitalização (PCM/DM), modem 56Kbps (PCM/TCM); (6) Camada de Enlace: serviços ofertados e tarefas realizadas; (7) Camada de Enlace: enquadramento, detecção de erros; (8) Camada de Enlace: controle de fluxo; correção de erros; (9) exemplo de protocolos: HDLC, PPP; (10) sub-camada de acesso ao meio: CSMA (11) sub-camada de acesso ao meio: MACA, TDMA e CDMA; (12) tecnologia 802.x; pontes; VLANs; Wi-Fi; DQDB 802.6; (13) redes de comutação de circuito e de pacotes; (14) encaminhamento de datagramas; circuitos virtuais; comutação de células; (15) tecnologia de rede: linhas alugadas, *frame-relay*, ATM, Gigabit Ethernet.
Bibliografia Stallings, *Data and Computer Communications* [13].

Construção de Compiladores

Objetivo Implementação de um compilador para subconjunto da linguagem Pascal.
Programa (1) Visão histórica, processadores de linguagens, tipos de tradutores; arquitetura de um compilador; (2) análise léxica; (3) especificação de um analisador léxico, Lex; (4) análise sintática; (4) gramáticas para linguagens de programação; tipos de análise sintática; (5) função do analisador sintático; gramáticas; (6) análise sintática descendente; linguagens LL(1); (7) construção de analisadores sintáticos recursivos descendentes; (8) análise sintática ascendente; método LR; linguagens LR(1); método SLR(1); (9) Yacc/Bison; (10) análise semântica; (11) tipos e escopo; (12) tradução dirigida por sintaxe; gramáticas de atributos; (13) geração de código.
Bibliografia Aho, Sethi, Ullman, *Compilers – Principles, and Tools* [1]; Kowaltowski, *Implementação de Linguagens de Programação* [8].

Referências

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1995. ISBN 0-201-10194-7.

[2] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araújo, C. Araújo, and E. Barros. The ArchC architecture description language and tools. *Inter'l Journal of Parallel Programming*, 33(5), October 2005.

[3] J. P. Hansen. TKgate Home Page, June 2006. `http://www.tkgate.org`.

[4] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 3rd edition, 2003. ISBN 1-55860-724-2.

[5] R. A. Hexsel. Especificação do Mico-V9. Especificação, Depto de Informática, UFPR, maio 2004. `http://www.inf.ufpr.br/roberto/mico9.pdf`.

[6] R. A. Hexsel. Sistemas digitais e microprocessadores. Tutorial, Depto de Informática, UFPR, março 2006. `http://www.inf.ufpr.br/roberto/microprocessadores.pdf`.

[7] R. H. Katz. *Contemporary Logic Design*. Benjamin-Cummings, 1994. ISBN 080532703-7.

[8] T. Kowaltowski. *Implementação de Linguagens de Programação*. Ed. Guanabara Dois, 1983.

[9] J. R. Levine. *Linkers & Loaders*. Morgan Kaufmann, 2000. ISBN 1-55860-496-0.

[10] D. A. Patterson and J. L. Hennessy. *Computer Organization & Design: The Hardware/Software Interface*. Morgan Kaufmann, 3rd edition, 2004. ISBN 1-55860-604-1.

[11] S. Rigo, R. Azevedo, and G. Araujo. The ArchC architecture description language. Technical report, IC - Unicamp, junho 2003. `http://www.archc.org`.

[12] A. Silberschatz, P. Galvin, and G. Gagne. *Operating Systems Concepts*. John Wiley, 6th edition, 2003. ISBN 047141743-2.

[13] W. Stallings. *Data and Computer Communications*. Prentice-Hall, 7th edition, 2004. ISBN 013100681-9.

[14] W. Stallings. *Operating Systems: Internals and Design Principles*. Prentice-Hall, 5th edition, 2005. ISBN 013147954-7.

[15] W. Stallings. *Computer Organization and Architecture: Designing for Performance*. Prentice-Hall, 7th edition, 2006. ISBN 013185644-8.

[16] A. S. Tanenbaum. *Organização Estruturada de Computadores*. Livros Técnicos e Científicos, 4th edition, 1999. ISBN 85-216-1253-2.

[17] R. J. Tocci and N. S. Wiedmer. *Sistemas Digitais*. Pearson, 2003. ISBN 8587918206.