Capítulo 6

Um Enfoque Interdisciplinar no Ensino de Arquitetura de Computadores

Cesar Albenes Zeferino, André Luis Alice Raabe,
Paulo Viniccius Vieira e Maicon Carlos Pereira

Abstract

This chapter presents the results of an experiment involving different courses of a Bachelor in Computer Science related to the use of a simple processors family designed to assist in learning concepts of computer architecture. The family of processors is called BIP (Basic Instruction-set Processor) and has three versions, named BIP I, BIP II and BIP III, with incremental resources and complexity. These different versions were used for teaching concepts related to Introductory Programming, Digital Circuits, Computer Architecture and Organization and Compilers. The chapter details the architecture of the BIP family of processors and discusses the results of its use in each of the courses mentioned.

Resumo

Este capítulo apresenta os resultados de um experimento envolvendo diferentes disciplinas de um curso de graduação em Ciência da Computação relacionadas ao uso de uma família de processadores simplificados, a qual foi concebida para auxiliar na aprendizagem de conceitos de arquitetura de computadores. A família de processadores foi denominada BIP (Basic Instruction-set Processor) e possui três versões, denominadas BIP I, BIP II e BIP III, com complexidades e recursos incrementais. As diferentes versões tornaram-se um recurso didático utilizado com enfoques diferentes nas disciplinas de Algoritmos e Programação, Circuitos Digitais, Arquitetura e Organização de Computadores e Compiladores. O capítulo detalha a arquitetura da família de processadores BIP e discute os resultados de sua utilização em cada uma das disciplinas mencionadas.

6.1 Introdução

A compreensão do funcionamento da arquitetura do computador e de seu processador possui importância central na formação dos alunos dos cursos de graduação em Ciência da Computação. Além de fornecer os conhecimentos básicos para possibilitar a inserção do aluno no contexto da pesquisa e do desenvolvimento de hardware, essa compreensão auxilia o entendimento da necessidade e do papel do software básico e fornece subsídios fundamentais para a aprendizagem e para a compreensão da lógica de programação.

Considerando esse último aspecto, é fato amplamente conhecido que alunos apresentam dificuldades na aprendizagem de conceitos de algoritmos e programação, em especial no primeiro ano do curso [Carbone e Kasboll 1998][Menezes e Nobre 2002][Pimentel *et al*. 2003][Good e Brna 2004][Khalife 2006]. Essa dificuldade está relacionada, entre outros aspectos, à ausência de afinidade com o raciocínio lógico formal que é o fundamento para a capacidade de abstração dos alunos. Nesse sentido, o estudo da arquitetura do computador cria a possibilidade de estabelecer relações dos conceitos de programação com aspectos concretos do hardware, reduzindo assim a necessidade de abstração.

Nesse contexto, identifica-se uma falta de sincronia nos currículos tradicionais em que o estudo da arquitetura e da organização do computador ocorre depois do ensino da programação. Para contornar esse problema, em muitos cursos, costuma-se apresentar algumas noções básicas de arquitetura e organização de computadores aos alunos em disciplinas que fornecem uma introdução geral à Computação, tipicamente no primeiro ano do curso.

No entanto, essa abordagem normalmente apresenta dois problemas: (*i*) a falta de uma articulação adequada entre os professores das disciplinas introdutórias a fim de estabelecer uma sincronia na apresentação dos conteúdos para que se possa beneficiar a aprendizagem de programação; e (*ii*) a limitação dos modelos utilizados para a apresentação dos conceitos básicos de arquitetura.

Buscando abordar esses problemas, uma pesquisa foi realizada pelos autores deste capítulo na Universidade do Vale do Itajaí (UNIVALI) a fim de desenvolver e disponibilizar uma série de processadores com um conjunto de instruções mínimo para auxiliar o aprendizado de conceitos de arquitetura e organização de computadores de modo incremental [Morandi, Raabe e Zeferino 2006][Morandi *et al.* 2006]. Dessa forma, os processadores desenvolvidos serviram de referência para a apresentação dos conceitos básicos necessários ao melhor entendimento das abstrações utilizadas nas disciplinas da área de Algoritmos e Programação.

A iniciativa de desenvolver uma família de processadores acabou permitindo que fosse dado um enfoque interdisciplinar aos conceitos de arquitetura e organização de computadores, vislumbrando seus desdobramentos em diferentes disciplinas correlatas, a saber: Computação Básica, Circuitos Digitais, Arquitetura e Organização de Computadores e Compiladores. Essa característica tornou-se um diferencial desta abordagem, uma vez que, muito frequentemente, os alunos relatam que não conseguem compreender plenamente as relações entre as disciplinas, e, nesse sentido, a abordagem utilizada fornece uma contribuição significativa.

Este capítulo busca enfatizar:

1. As diretrizes para elaboração da família de processadores proposta;
2. De que forma os conceitos foram trabalhados de maneira incremental nas diferentes disciplinas correlatas discutindo pontos positivos, negativos e dificuldades;
3. As ferramentas computacionais utilizadas e construídas para apoiar a aplicação da abordagem; e
4. Os resultados do ponto de vista didático.

Acredita-se que, por ser uma proposta que difere da abordagem tradicional de ensino na área, possua uma contribuição relevante para a comunidade interessada no ensino de Computação como um todo.

O restante deste capítulo está organizado em três seções. A Seção 2 apresenta a família de processadores BIP (Basic Instruction-Set Processor) fazendo uma análise introdutória dos processadores existentes para o ensino de Arquitetura de Computadores e fundamentando a necessidade de concepção da Família BIP. Na sequência, a seção está dividida em quatro subseções que descrevem as características gerais da Família BIP e a arquitetura e a organização de cada versão do processador. A Seção 3 detalha as estratégias, conteúdos, organização didática e ferramentas utilizadas para condução de atividades de aprendizagem com a família de processadores BIP. Apresenta-se uma descrição geral de como foi viabilizada a integração interdisciplinar e a visão dos autores sobre interdisciplinaridade em Computação. Cada uma das disciplinas envolvidas é descrita em uma subseção, a saber: (*i*) Algoritmos e Programação; (*ii*) Computação Básica; (*iii*) Circuitos Digitais; (*iv*) Arquitetura e Organização de Computadores; e (*v*) Compiladores, discutindo as formas como a arquitetura do BIP foi aplicada para apoiar o aprendizado e a consolidação de conceitos nessas disciplinas. Concluindo, a seção de considerações finais resgata os principais pontos da proposta e discute as direções futuras da pesquisa realizada.

6.2 Família de Processadores BIP

A escolha de modelos de processadores para o ensino de conceitos de arquitetura e organização de computadores é alvo de estudos frequentes pelos educadores da área, como, por exemplo, no trabalho apresentado por Clements (1999) que discute aspectos que devem ser levados em consideração na escolha de modelos de processadores a serem aplicados no ensino de graduação. Enquanto alguns autores e professores optam por utilizar modelos hipotéticos de processadores, outros adotam processadores reais e comerciais como referência para estudos de caso.

Para as fases iniciais de um curso de graduação, a seleção de processadores para o ensino concorrente da lógica de programação e de conceitos de arquitetura de computadores deve facilitar o estabelecimento de relações entre as abstrações lógicas necessárias à programação e à implementação dessas abstrações em hardware. Porém, os modelos de processadores tipicamente utilizados por professores de disciplinas introdutórias são abstratos demais e não permitem estabelecer essas relações. Uma alternativa seria utilizar modelos de processadores mais detalhados, como aqueles adotados nas disciplinas específicas da área de Arquitetura de Computadores

(*e.g.* MIPS, x86,...). Porém, esses processadores são demasiadamente complexos para serem aplicados em disciplinas do primeiro ano, e poucos são os livros-texto da área que os descrevem propiciando uma integração entre a arquitetura do processador e a programação em alto nível.

Uma exceção é o livro "Organização e Projeto de Computadores", de Patterson e Hennessy (2005), no qual os autores descrevem o processador MIPS com ênfase na exploração da interface entre o hardware e software, permitindo a interligação dos conceitos apresentados com aqueles estudados nas disciplinas de programação. Essa abordagem favorece a interdisciplinaridade e a relação entre os conceitos mais abstratos da programação de alto nível e a realização física do processador. No entanto, o pouco embasamento dos alunos nas fases iniciais torna inadequado o uso de processadores com o grau de complexidade do MIPS, e alternativas mais simples podem e devem ser buscadas.

Nesse contexto, deve-se buscar uma arquitetura simplificada que permita estabelecer uma relação entre as necessidades dos alunos que estão começando a programar e as representações em hardware correspondentes. É necessário realizar a identificação das principais fontes de incompreensão para os estudantes de modo a prover formas de minimizar suas dificuldades. Por exemplo, podem ser citadas algumas relações importantes entre a programação de alto nível e a sua implementação no hardware, sob a forma de conceitos de arquitetura e organização de computadores. Entre essas relações, destacam-se:

- Declaração de variável e alocação de memória;
- Constantes e operandos imediatos;
- Atribuição de variáveis e sua correspondência com as operações de acesso à memória; e
- Operações aritméticas e sua execução no hardware.

Nesse sentido, discussões continuadas entre professores das áreas de Algoritmos e Programação e de Arquitetura de Computadores na Universidade do Vale do Itajaí (UNIVALI) levaram à concepção de uma arquitetura simplificada de processador tendo duas diretrizes principais de projeto:

1. Facilitar o aprendizado da arquitetura e o entendimento dos conceitos por ela ilustrados por alunos da primeira fase do curso de Ciência da Computação da UNIVALI; e
2. Viabilizar e facilitar o uso da arquitetura em disciplinas mais avançadas, promovendo uma integração interdisciplinar.

Para minimizar a complexidade do processador, foi adotada uma arquitetura orientada a acumulador com características derivadas da arquitetura dos microcontroladores PIC® da Microchip (2003). No entanto, algumas escolhas foram feitas no sentido de conferir uma regularidade arquitetural maior do que a desses microcontroladores, nos quais as palavras de dado e de instrução têm tamanhos diferentes e a largura do campo de código de operação pode variar para as diversas classes de instrução. Essas escolhas foram baseadas na assertiva apresentada por Patterson e Hennessy (2005) de que quanto mais regular for a arquitetura de um processador, mais fácil será a sua implementação. Além

disso, entende-se que a regularidade também favorece o entendimento da arquitetura do processador.

Disso, foi realizada uma especificação arquitetural de uma família de processadores simplificados denominada BIP (Basic Instruction-set Processor) contendo três modelos de processador, cada um com suporte incremental ao entendimento de conceitos de Algoritmos e Programação e oferecendo recursos adicionais para seu uso em outras disciplinas, em uma abordagem interdisciplinar:

- BIP I: inclui apenas instruções de aritmética e de acesso à memória de dados, tendo como foco o suporte ao entendimento de conceitos como níveis de linguagem, constantes, variáveis, representação de dados e de instruções, conjuntos de instruções, programação em linguagem de montagem e geração de código em linguagem de máquina;
- BIP II: acrescenta instruções de desvio, com foco na inclusão de suporte aos conceitos de estruturas de controle para desvios condicionais e incondicionais e laços de repetição; e
- BIP III: acrescenta instruções de lógica, focando na inclusão de suporte a operações de lógica bit-a-bit.

A seguir, é apresentada a especificação da família BIP, identificando-se seus atributos arquiteturais e explicando o porquê de cada escolha tomada.

6.2.1 Atributos Arquiteturais da Família BIP

Com base nas diretrizes de projeto discutidas previamente, foram definidos os seguintes atributos para a arquitetura da Família BIP:

1. Palavras de instrução e de dados: Os tamanhos das palavras de instrução e de dado foram fixados em um mesmo valor (16 bits) para viabilizar implementações baseadas em uma única memória para o armazenamento de instruções e de dados (arquitetura de von Neumann) e em memórias separadas (arquitetura Harvard);
2. Registradores: A arquitetura prevê quatro registradores: PC (Program Counter), IR (Instruction Register), STATUS e ACC, sendo que o uso de alguns desses registradores varia conforme a implementação do processador. Por exemplo, numa implementação monociclo com arquitetura Harvard o PC aponta para a instrução corrente e o IR não é necessário, pois a saída da memória de instruções mantém a instrução a ser executada durante todo o ciclo da instrução. Já numa implementação multiciclo, o IR deve ser usado para armazenar a instrução corrente, pois o PC é atualizado durante o ciclo de instrução para apontar para a próxima instrução a ser executada. O registrador STATUS inclui *flags* referentes à execução da última instrução: Z (Zero), N (Negative) e C (Carry). Finalmente, o registrador ACC (o acumulador) armazena o resultado da última operação aritmética;
3. Modelos de execução: Pelo fato de ter um único registrador de propósito geral (o ACC), foi considerado que as operacões aritméticas e lógicas poderiam processar operandos oriundos da memória de dados (o que também ocorre nos

microcontroladores PIC®). Dessa forma, são suportados os modelos de execução Registrador-Registrador e Registrador-Memória;

4. Formato de instrução: Foi definido um único formato de representação para todo o conjunto de instruções. Esse formato, ilustrado na Figura 6.1, é composto por dois campos: um código de operação de 5 bits e um operando explícito de 11 bits;

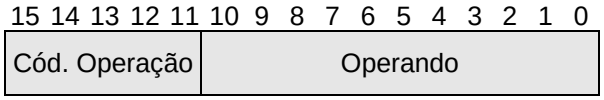

Figura 6.1. Formato de instrução

5. Espaços de endereçamento: O espaço de endereçamento é limitado a 2048 posições devido à largura do campo de operando da instrução (11 bits). A arquitetura utiliza entrada-e-saída (E/S) mapeada em memória, sendo que o acesso à E/S é realizado usando as mesmas instruções de acesso à memória. A organização dos espaços de endereçamento pode ser explorada de diferentes maneiras. Pode ser definido um espaço de endereçamento único para instruções, dados e E/S, conforme ilustrado na Figura 6.2.a. Porém, se o processador utilizar uma organização de memória tipo Harvard, podem ser definidos dois espaços de endereçamento separados: um para instruções e outro para dados e E/S (como é ilustrado na Figura 6.2.b). Essa abordagem permite implementar programas com até 2K instruções e 2K posições de dados e de E/S;

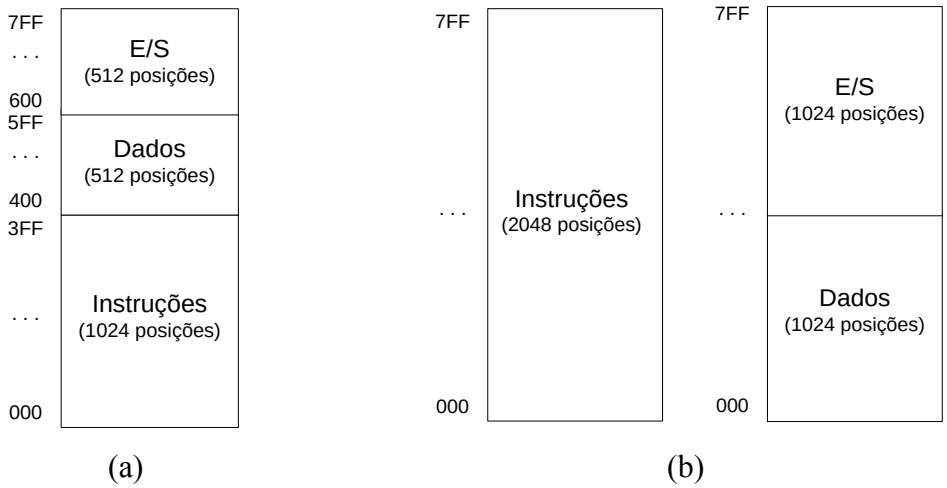

Figura 6.2. Alternativas de organização do espaço de endereçamento: (a) unificado; (b) dividido

6. Modos de endereçamento: Foram definidos dois modos de endereçamento: imediato e direto. O modo imediato (ilustrado na Figura 6.3) é utilizado para operações envolvendo um registrador (o ACC ou o PC) e o operando da instrução (neste caso uma constante de 11 bits com o sinal representado em complento de dois). O modo de endereçamento direto é utilizado para operações entre o acumulador e uma posição do espaço de endereçamento de memória apontada pelo operando da instrução, como ilustrado na Figura 6.4.

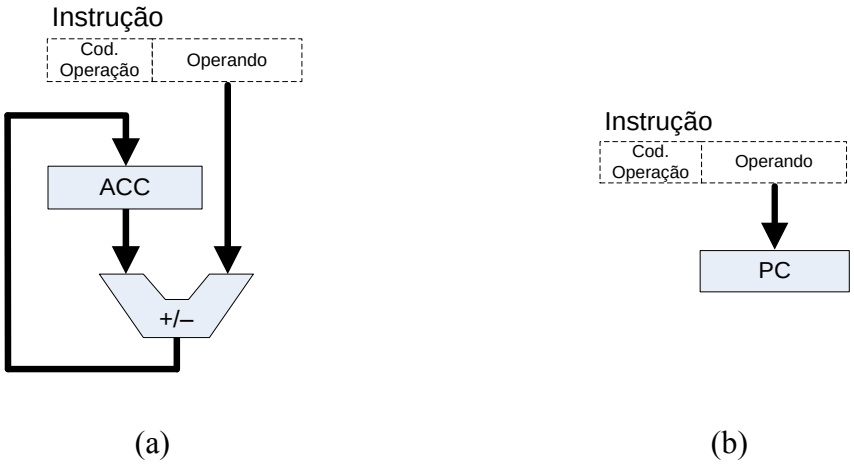

Figura 6.3. Modo de endereçamento imediato: (a) operação com o ACC; (b) operação com o PC

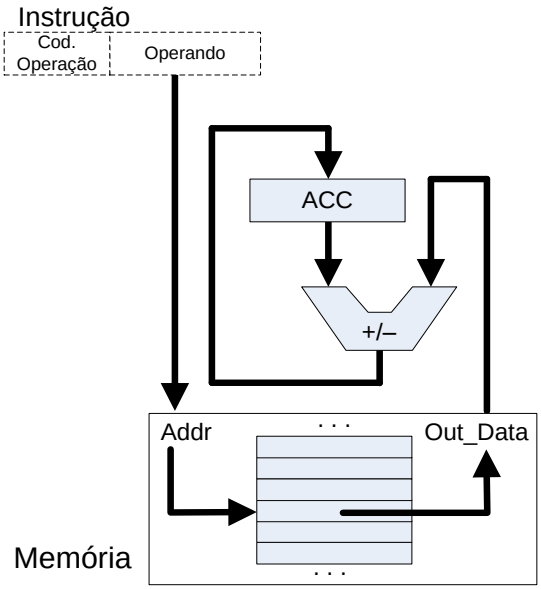

Figura 6.4. Modo de endereçamento direto

6.2.2 Arquitetura e Organização do BIP I

O conjunto de instruções do BIP I é formado por oito instruções, incluindo instruções de controle (HLT), armazenamento em memória (STO), carga no acumulador (LD e LDI) e de aritmética (ADD, ADDI, SUB e SUBI), as quais são descritas na Tabela 6.1, a seguir.

Tabela 6.1. Conjunto de instruções do BIP I

Código da Operação	Instrução		Operação	Classe
00000	HLT		Paralisa a execução do programa	Controle
00001	STO	operando	Memória[operando]← ACC	Armazenamento
00010	LD	operando	ACC ← Memória[operando]	Carga
00011	LDI	operando	ACC ← operando	Carga
00100	ADD	operando	ACC ← ACC + Memória[operando]	Aritmética
00101	ADDI	operando	ACC ← ACC + operando	Aritmética
00110	SUB	operando	ACC ← ACC – Memória[operando]	Aritmética
00111	SUBI	operando	ACC ← ACC – operando	Aritmética

A instrução HLT (*halt*) tem a função de desabilitar a atualização do PC, paralisando a execução do programa. Nas demais instruções, o PC é incrementado em uma unidade.

A instrução STO (*store*) realiza a transferência do conteúdo do registrador ACC para uma posição do espaço de endereçamento de memória (Memória[operando]). Quanto às instruções de carga, a instrução LD (*load*) realiza uma operação de transferência de uma posição do espaço de endereçamento de memória para o acumulador, enquanto que a instrução LDI (*load immediate*) carrega uma constante (o operando) no acumulador.

Com relação às instruções aritméticas, são disponibilizadas instruções de soma e de subtração entre o acumulador e uma posição do espaço de endereçamento de memória e entre o acumulador e uma constante. São elas: ADD (*add*), SUB (*subtract*), ADDI (*add immediate*) e SUBI (*subtract immediate*). Embora a instrução SUBI possa ser dispensada pelo uso do ADDI com um operando negativo, como é feito no MIPS [Patterson e Hennessy 2005], optou-se por disponibilizar a instrução SUBI para conferir uma maior facilidade ao aprendizado.

Quanto aos modos de endereçamento, as instruções LDI, ADDI e SUBI utilizam o modo imediato, enquanto que as instruções STO, LD, ADD e SUB utilizam o modo direto. Em todas essas instruções, o acumulador é utilizado como um operando implícito, atuando como operando fonte e/ou destino das operações realizadas.

Na Tabela 6.2, são apresentados alguns exemplos de uso da linguagem de montagem do BIP I para a implementação de abstrações representadas em linguagens de alto nível. Como pode ser observado, pelo seu conjunto de instruções, o BIP I consiste basicamente de uma calculadora programável que realiza operações de soma e subtração com variáveis e constantes. No entanto, apesar de limitado, esse conjunto de instruções permite ilustrar várias relações entre as abstrações estudadas nas disciplinas da área de Algoritmos e Programação e sua representação no nível arquitetural do processador, conforme será discutido posteriormente.

Tabela 6.2. Uso da linguagem de montagem do BIP I

Abstração	Código de alto nível	Código na linguagem de montagem
Atribuição de uma constante	`A = 10;`	`LDI 10 ;ACC ← 10` `STO A ;A ← ACC`
Atribuição de uma variável	`A = B;`	`LD B ;ACC ← B` `STO A ;A ← ACC`

Comando com uma operação aritmética	`A = A + 1;`	`LD A ;ACC ← A`
		`ADDI 1 ;ACC ← ACC + 1`
		`STO A ;A ← ACC`

Comando com múltiplas operações aritméticas	`A = A + B – 3;`	`LD A ;ACC ← A`
		`ADD B ;ACC ← ACC + B`
		`SUBI 3 ;ACC ← ACC – 3`
		`STO A ;A ← ACC`

Na Figura 6.5, são apresentadas duas alternativas de organização para o BIP I. A primeira é uma organização monociclo baseada em uma arquitetura Harvard, com memórias separadas para armazenar instruções e dados (Figura 6.5.a). Já a segunda é uma organização multiciclo baseada em uma arquitetura de von Neumann, com uma memória unificada (Figura 6.5.b). Os diagramas apresentados abstraem alguns aspectos de implementação, como o uso de multiplexadores, a largura dos canais e os sinais de controle (os quais serão detalhados posteriormente).

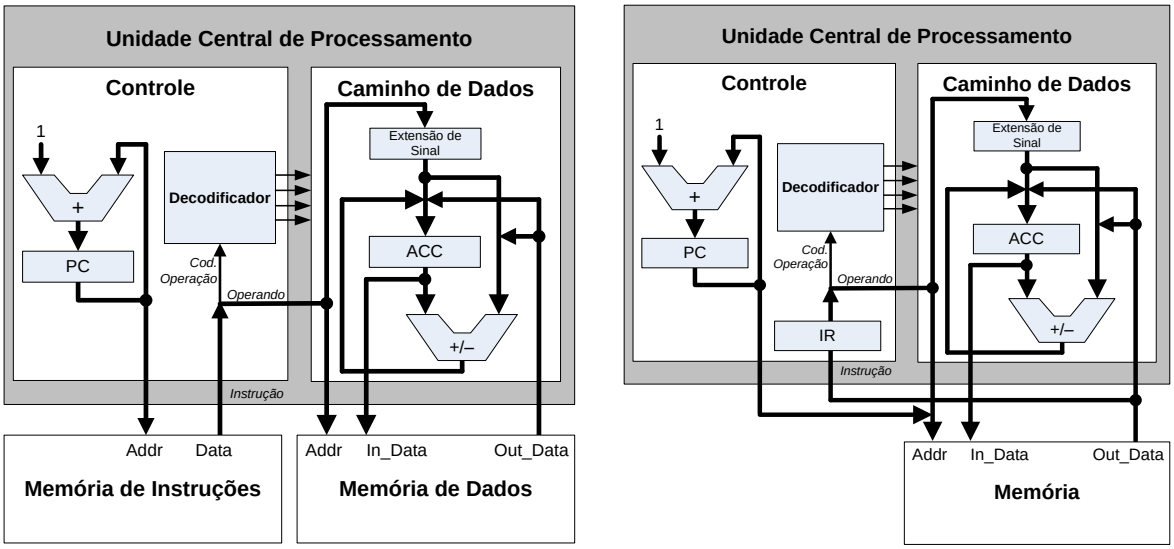

Figura 6.5. Organizações alternativas: (a) Harvard monociclo; (b) von Neumann multiciclo

A Unidade Central de Processamento (UCP) do BIP I é estruturada em dois blocos: o Controle e o Caminho de Dados. O Controle inclui o registrador PC, um somador para atualizar o valor do PC e o decodificador que gera os sinais de controle necessários para a execução de cada instrução. O Caminho de Dados, por sua vez, inclui o registrador

ACC, uma unidade aritmética de soma e de subtração e um circuito para estender o sinal do operando (de 11 para 16 bits). A principal diferença entre a UCP das duas organizações está no uso do registrador IR no Controle. A organização de Harvard dispensa esse registrador pois a memória de instrução mantém em sua saída a instrução corrente. Já a organização de von Neumann precisa do IR pois, após a busca da instrução, a memória pode ser utilizada com fonte ou destino de alguma operação. As duas organizações não incluem o registrador STATUS no Caminho de Dados, pois o conjunto de instruções do BIP I não possui instruções que analisem esse registrador.

Além das organizações ilustradas, outras variações podem ser implementadas. Por exemplo, adicionando o registrador IR à organização Harvard e com as devidas modificações no bloco de decodificação, pode ser construída uma organização com um *pipeline* de dois estágios sobrepondo a busca e a execução de duas instruções subsequentes, assim como é feito nos microcontroladores PIC® da Microchip (2003).

A Figura 6.6 apresenta um detalhamento da organização Harvard monociclo. Nela, podem ser observados os multiplexadores de seleção do Caminho de Dados, os sinais de controle gerados pelo Decodificador e a largura dos canais que interligam os componentes. Os circuitos utilizados na implementação da UCP são aqueles tipicamente estudados em disciplinas de Eletrônica Digital (decodificador, registrador, somador, somador/subtrator, multiplexador e memória). A exceção é bloco de extensão de sinal que converte o canal de 11 bits do operando para a largura da palavra do Caminho de Dados (16 bits). Essa conversão é feita pela atribuição do bit sinal do operando (bit 10) aos bits 11-15 da interface de saída do bloco.

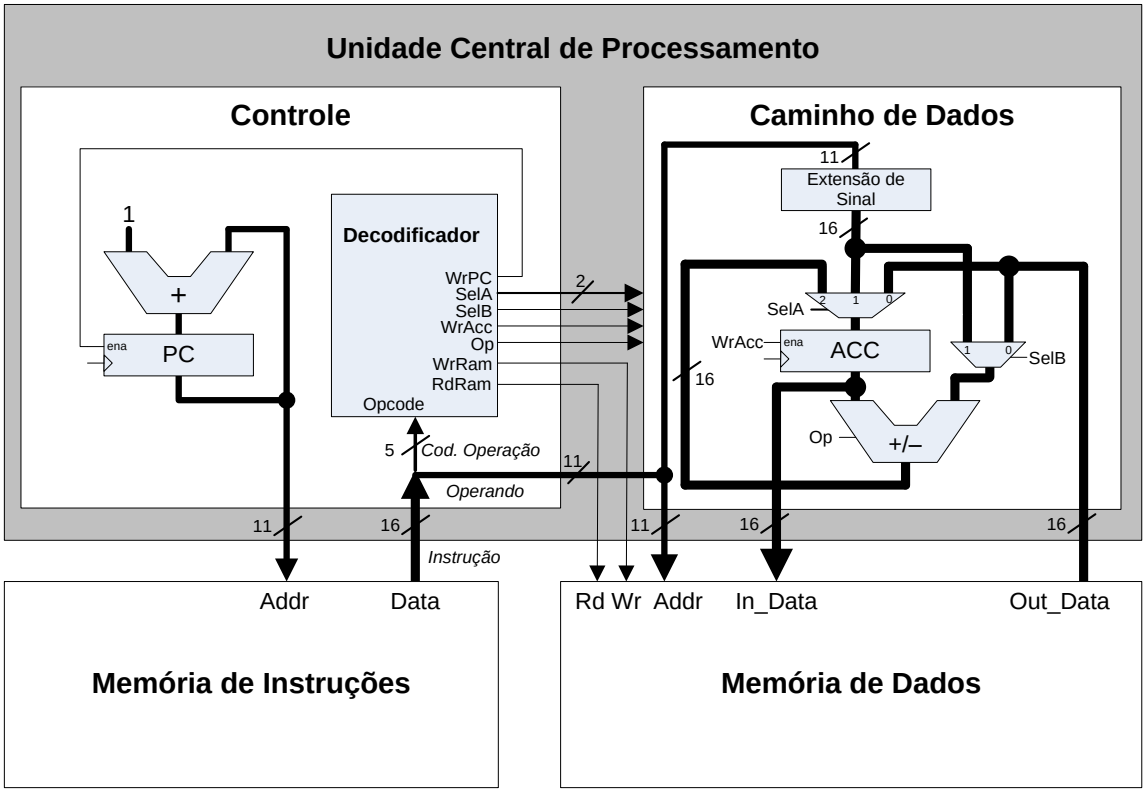

Figura 6.6. Organização monociclo do BIP I baseada em uma arquitetura tipo Harvard

Vale destacar que as organizações ilustradas não incluem uma interface de E/S. No entanto, por utilizar entrada-e-saída mapeada em memória, periféricos e dispositivos de E/S podem ser facilmente acrescentados conectando-os ao barramento da memória de dados.

6.2.3 Arquitetura e Organização do BIP II

O BIP II é uma extensão do BIP I e possui as mesmas características arquiteturais. Ele foi especificado para suportar estruturas de controle usadas em desvios e em laços de repetição. Para tal, o conjunto de instruções do BIP I foi estendido incluindo seis instruções de desvio condicional: BEQ (*branch on equal*), BNE (*branch on not equal*), BGT (*branch on greater than*), BGE (*branch on greater than or equal*), BLT (*branch on less than*) e BLE (*branch on less than or equal*). Também foi adicionada a instrução de desvio incondicional JMP (*jump*). Essas novas instruções são descritas na Tabela 6.3.

Tabela 6.3. Conjunto de instruções estendido do BIP II

Código da Operação	Instrução	Operação	Classe
01000	BEQ operando	Se (STATUS.Z=1) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01001	BNE operando	Se (STATUS.Z=0) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01010	BGT operando	Se (STATUS.Z=0) e (STATUS.N=0) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01011	BGE operando	Se (STATUS.N=0) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01100	BLT operando	Se (STATUS.N=1) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01101	BLE operando	Se (STATUS.Z=1) ou (STATUS.N=1) então PC ← operando Se não PC ← PC + 1	Desvio condicional
01110	JMP operando	PC ← operando	Desvio incondicional

Conforme visto na Tabela 6.3, para suportar as instruções de comparação e desvio condicional, foi necessário incluir o registrador STATUS, em especial os *flags* booleanos Z (*zero*) e N (*negative*). Qualquer instrução de comparação e desvio condicional deve ser precedida por uma instrução de subtração (SUB ou SUBI). Dependendo do resultado dessa operação, os *flags* Z (Zero) e N (Negative) do registrador STATUS são definidos em 0 (FALSE) ou em 1 (TRUE). As instruções de desvio condicional então verificam o valor desses *flags* para determinar se o desvio deve ser tomado ou não, conforme o tipo de comparação associado. Essa estratégia se deve à limitação do formato de instrução, o qual suporta um único operando explícito. Esse operando é usado para codificar o endereço de desvio, enquanto que os valores comparados são processados pela instrução de subtração.

As instruções de desvio utilizam o modo de endereçamento imediato quando carregam o endereço de desvio no PC, o qual é considerado um operando implícito.

As tabelas a seguir apresentam exemplos de uso do conjunto de instruções do BIP II para implementar as estruturas de controle. Nas tabelas, utiliza-se o termo "*Bloco i*" para designar um segmento de código que representa uma sequência de instruções que executa alguma tarefa, a qual é irrelevante para o contexto do exemplo.

Tabela 6.4. Uso da linguagem de montagem do BIP II (Parte I)

Abstração	**Código de alto nível**	**Código na linguagem de montagem**

Teste de condição do tipo `if-then`	`if (A==B) {`
	`  // Bloco 1`
	`}`
	`// Bloco 2`
		`  LD  A  ;ACC ← A`
		`  SUB  B  ;ACC ← ACC - B`
		`  BNE  L1`
		`  ;Bloco 1`
		`L1:`
		`  ;Bloco 2`

Teste de condição do tipo `if-then-else`	`if (A==B) {`
	`  // Bloco 1`
	`} else {`
	`  // Bloco 2`
	`}`
	`// Bloco 3`
		`  LD  A  ;ACC ← A`
		`  SUB  B  ;ACC ← ACC - B`
		`  BNE  L1`
		`  ;Bloco 1`
		`  JMP  L2`
		`L1:`
		`  ;Bloco 2`
		`L2:`
		`  ;Bloco 3`

Laço de repetição do tipo `while`	`i = 0;`
	`while (i<10) {`
	`  // Bloco 1`
	`  i++;`
	`}`
	`// Bloco 2`
		`  LDI  0  ;ACC ← 0`
		`  STO  I  ;I  ← ACC`
		`L1:`
		`  SUBI 10  ;ACC ← ACC - 10`
		`  BGE  L2`
		`  ;Bloco 1`
		`  LD  I  ;ACC ← I`
		`  ADDI 1  ;ACC ← ACC + 1`
		`  STO  I  ;I  ← ACC`
		`  JMP  L1`
		`L2:`
		`  ;Bloco 2`

Tabela 6.5. Uso da linguagem de montagem do BIP II (Parte II)

Abstração	Código de alto nível	Código na linguagem de montagem

Laço de repetição do tipo `do-while`	`i = 0;`
	`do {`
	`  // Bloco 1`
	`  i++;`
	`} while (i<10)`
	`// Bloco 2`
		`  LDI  0  ;ACC ← 0`
		`  STO  I  ;I  ← ACC`
		`L1:`
		`  ;Bloco 1`
		`  LD  I  ;ACC ← I`
		`  ADDI 1  ;ACC ← ACC + 1`
		`  STO  I  ;I  ← ACC`
		`  SUBI 10  ;ACC ← ACC - 10`
		`  BLT  L1`
		`  ;Bloco 2`

Laço de repetição do tipo `for`	`for(i=0;i<10;i++){`
	`  // Bloco 1`
	`}`
	`// Bloco 2`
		Implementação em linguagem de montagem idêntica à implementação do laço de repetição do tipo `while`

A Figura 6.7 apresenta a organização Harvard monociclo do processador BIP II. Em relação à organização do processador BIP I, ilustrada previamente na Figura 6.6, essa organização inclui os circuitos necessários à implementação das instruções de desvio: (*i*) um multiplexador na entrada do PC, para permitir a carga do valor do operando; e (*ii*) o registrador STATUS, com os *flags* N e Z.

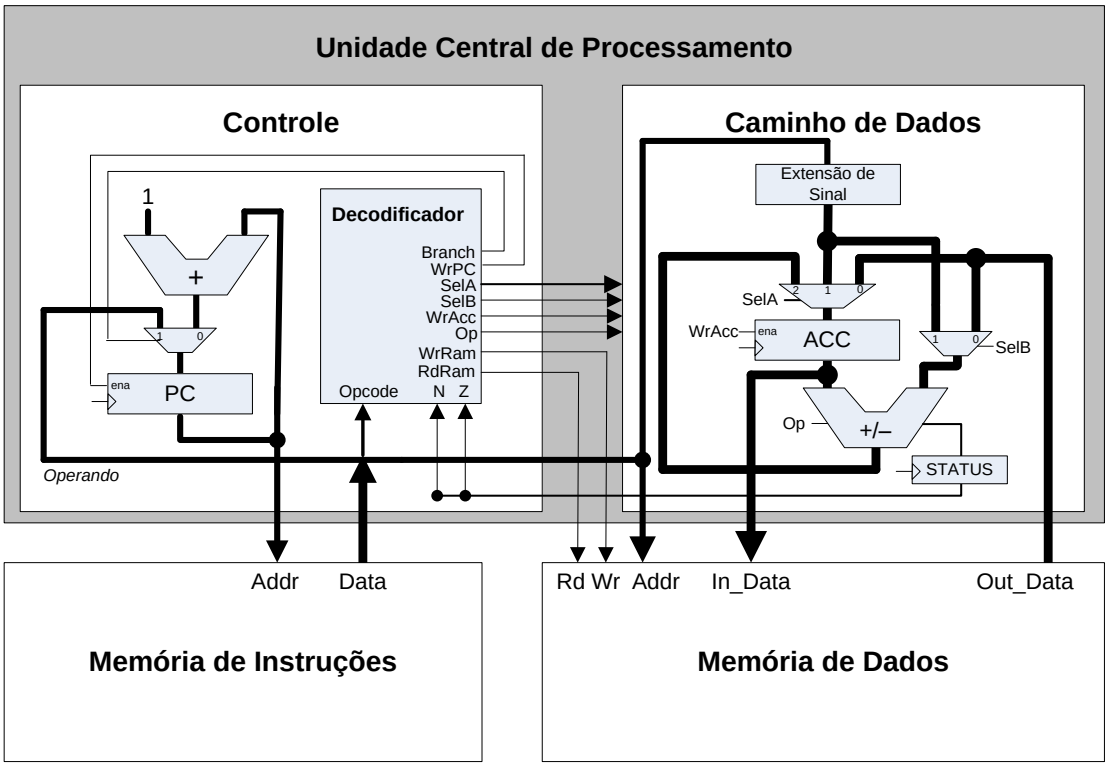

Figura 6.7. Organização monociclo do BIP II baseada em uma arquitetura tipo Harvard

6.2.4 Arquitetura e Organização do BIP III

O BIP III estende o conjunto de instruções do BIP II adicionando nove instruções de lógica bit-a-bit, as quais são descritas na Tabela 6.6.

Tabela 6.6. Conjunto de instruções estendido do BIP III

Código da Operação	Instrução	Operação	Classe
01111	NOT	ACC ← NOT(ACC)	Lógica
10000	AND operando	ACC ← ACC AND Memória[operando]	Lógica
10001	ANDI operando	ACC ← ACC AND operando	Lógica
10010	OR operando	ACC ← ACC OR Memória[operando]	Lógica
10011	ORI operando	ACC ← ACC OR operando	Lógica
10100	XOR operando	ACC ← ACC XOR Memória[operando]	Lógica
10101	XORI operando	ACC ← ACC XOR operando	Lógica
10110	SLL operando	ACC ← ACC << operando	Lógica
10111	SRL operando	ACC ← ACC >> operando	Lógica

A organização do BIP III é similar à organização do BIP II. As poucas modificações necessárias referem-se à substituição do somador-subtrator do caminho de dados por uma unidade funcional integrando uma UAL (Unidade Aritmética Lógica) e *barrel shifters* para realizar os deslocamentos de bits à direita e à esquerda. Além disso, o Decodificador deve ser atualizado para identificar as novas instruções. A Tabela 6.7, logo a seguir, apresenta a codificação definida para o sinal *Op* que comanda a Unidade Funcional, o qual passa a ter 3 bits de largura no BIP III.

Tabela 6.7. Operações suportadas pela unidade funcional do BIP III

Op	Operação	Instruções
000	Soma	ADD e ADDI
001	Subtração	SUB e SUBI
010	Função lógica E	AND e ANDI
011	Função lógica OU	OR e ORI
100	Função lógica XOR	XOR e XORI
101	Função lógica NOT	NOT
110	Deslocamento lógico para a esquerda	SLL
111	Deslocamento lógico para a direita	SRL

6.3 Utilização do BIP no Ensino

As iniciativas de criação de processadores com objetivo didático normalmente buscam apoiar as disciplinas diretamente envolvidas, como Arquitetura e Organização de Computadores. A criação da família de processadores BIP buscou transcender esse enfoque, induzindo um leque de possibilidades de exploração dos conceitos envolvidos em diferentes disciplinas da Ciência da Computação em uma abordagem interdisciplinar.

As três versões do processador BIP permitiram trabalhar níveis de complexidade diferentes e com isso envolver disciplinas do primeiro, segundo, terceiro e sétimo semestres do curso de Ciência da Computação da Universidade do Vale do Itajaí (UNIVALI), além de envolver trabalhos de conclusão de curso.

Sabe-se que existem variações na matriz curricular e na nomenclatura de disciplinas nos cursos de Ciência da Computação de instituições distintas. Tendo isso em vista, a Tabela 6.8 apresenta um extrato da organização curricular utilizada no curso de Ciência da Computação da UNIVALI. Foram acrescidas informações sobre as ementas das disciplinas em que os conceitos do BIP foram trabalhos a fim de permitir ao leitor estabelecer uma relação destas com as disciplinas correlatas em outras instituições que possuam o curso de Ciência da Computação.

Tabela 6.8. Extrato da matriz curricular do curso de Ciência da Computação da Universidade do Vale do Itajaí – UNIVALI

1º Período		
Disciplina	**C/H**	**Ementa**
Algoritmos e Programação	120	Conceitos preliminares. Manipulação de dados. Estruturas de controle de fluxo. Tipos compostos de dados. Modularização de algoritmos. Programação em linguagem C.
Computação Básica	60	Histórico da computação. Sistemas de numeração. O hardware do computador. O software do computador. Funcionamento básico do computador.
2º Período		
Disciplina	**C/H**	**Ementa**
Circuitos Digitais	60	Circuitos com portas lógicas. Simplificação de circuitos. Circuitos combinacionais. Circuitos sequenciais.
3º Período		
Disciplina	**C/H**	**Ementa**
Arquitetura e Organização de Computadores I	60	Introdução à arquitetura e organização de computadores. Conjunto de instruções e programação em linguagem de montagem. Representação de dados e aritmética binária. Organização de processadores. Processamento em Pipeline.
7º Período		
Disciplina	**C/H**	**Ementa**
Compiladores	60	Visão geral de um compilador. Análise léxica. Análise sintática. Análise semântica. Geração de código. Tópicos complementares em compiladores.

A seguir são relatadas as experiências dos autores na utilização dos conceitos relacionados ao uso do BIP nas referidas disciplinas.

6.3.1 Algoritmos e Programação

A crença inicial que motivou o enfoque interdisciplinar na concepção da família de processadores BIP foi de que utilizar um modelo simplificado de computador auxilia a melhorar a compreensão de conceitos relacionados à aprendizagem inicial de programação. Entende-se que, ao reduzir o grau de abstração envolvido com a aprendizagem de conceito ligados ao uso de memória, operações aritméticas, desvios e laços, é possível facilitar a aprendizagem dos alunos que em geral apresentam problemas em lidar com abstrações.

As relações entre as abstrações estudadas nas disciplinas da área de Programação e sua representação no nível arquitetural do processador são listadas na Tabela 6.9.

Tabela 6.9. Relações entre conceitos de Programação e de Arquitetura suportadas pelo BIP I

Conceitos de Programação	Conceitos de Arquitetura de Computadores
Variável	Posição na memória
Constante	Operando imediato na instrução
Atribuição	Acesso à memória para leitura e/ou escrita de/em uma posição
Operação aritmética	Utilização de unidade de soma/subtração
Comandos com múltiplas operações	Uso de uma instrução para cada operação realizada
Desempenho dos programas	Número de instruções na linguagem de montagem
Papel do compilador	Tradução da linguagem de alto nível para a linguagem de montagem

A abordagem utilizada anteriormente à existência do BIP resumia-se a uma breve explicação sobre o funcionamento de um computador, ilustrando os componentes básicos da arquitetura de Von Neumann e em seguida realizava-se a introdução de conceitos como váriáveis, atribuições e operações de entrada-e-saída.

Com a introdução do BIP I, a explicação dos conceitos básicos de arquitetura de computadores tornou-se mais detalhada e os conceitos apresentados não restringiram-se apenas à disciplina de Algoritmos e Programação. Foram mostrados exemplos de programas em português estruturado (Portugol) e os correspondentes na linguagem de montagem. O Portugol vem sendo adotado nesta disciplina como uma linguagem de programação simplificada visando reduzir a complexidade na criação dos primeiros programas eliminando a dificuldade que alguns alunos apresentam com o idioma inglês e com detalhes específicos da interface dos compiladores e ambientes de programação comerciais [Hostins e Raabe, 2007].

Posteriormente, foram realizados exercícios com os alunos sobre a construção de programas em linguagem de montagem. A seguir, são apresentados exemplos de exercícios com as respostas esperadas em destaque:

Exemplo de Exercício 1

Dado o trecho de código a seguir, escrito na linguagem de montagem do BIP, comente cada linha do código identificando a operação realizada:

```
LDI   0     ; ACC ← 0
ADDI  1     ; ACC ← ACC + 1
ADD   B     ; ACC ← ACC + B
STO   A     ; A   ← ACC
```

Resposta esperada

Exemplo de Exercício 2

Dado o trecho de código a seguir, escrito em linguagem de alto nível, escreva o código equivalente na linguagem de montagem do BIP, identificando, nos comentários, a operação realizada por cada comando na linguagem de montagem:

Código em linguagem de alto nível

```
X = 0;
Y = 2;
X = X + Y;
```

Código na linguagem de montagem do BIP:

```
LDI   0     ; ACC ← 0
STO   X     ; X   ← ACC
LDI   2     ; ACC ← 02
STO   Y     ; Y   ← ACC
ADD   X     ; ACC ← ACC + X
STO   X     ; X   ← ACC
```

Resposta esperada

Exemplo de Exercício 3

Dado o trecho de código a seguir, escrito na linguagem de montagem do BIP, comente cada linha de código e obtenha o código equivalente em linguagem de alto nível:

Código na linguagem de montagem do BIP:

```
LD    B     ; ACC ← B
ADDI  1     ; ACC ← ACC + 1
SUB   C     ; ACC ← ACC - C
STO   A     ; A   ← ACC
```

Código em linguagem de alto nível:

```
A = B + 1 - C;
```

Resposta esperada

Após a apresentação da arquitetura do BIP I e a aplicação de exercícios similares aos exemplificados acima, os principais aspectos observados com relação à aprendizagem dos alunos foram:

- Os alunos demonstraram alguma dificuldade inicial em compreender as instruções de linguagem de montagem, mas rapidamente passaram a adquirir fluência na resolução dos exercícios;
- A compreensão da existência do acumulador tornou a operação de atribuição mais clara. O que acontecia com o resultado intermediário da operação A+1 em A ← A + 1 antes ficava obscuro;
- Um problema recorrente em vários semestres, em que alguns alunos declaravam contantes inteiras como se fossem variáveis (Ex: inteiro 5) ou ainda realizavam operações de atribuição para constantes (5 ← 3 + 2), não ocorreu nos semestres em que foi usada a abodagem com o BIP I;
- Como a explicação sobre o BIP I reuniu turmas de períodos diferentes (Algoritmos e Programação do 1º período e Arquitetura e Organização de Computadores I do 3º período) ficou claro aos alunos do primeiro período que os conceitos seriam importantes também para a sequência do curso.

A aplicação do processador BIP I permitiu confirmar a efetividade da abordagem proposta, uma vez que os alunos puderam consolidar os conceitos estudados na disciplina da área de Programação.

No entanto, pôde-se evidenciar que o conjunto de instruções limitado do BIP I restringiu o seu uso por não suportar a implementação de estruturas de controle (desvios condicional e incondicional) no nível arquitetural. Essa limitação, prevista no início do projeto, já foi contornada com a disponibilização da arquitetura do BIP II. Porém esse processador ainda não foi utilizado nesta disciplina. Nas próximas turmas, pretende-se aplicar o BIP I inicialmente para apresentar os conceitos introdutórios e, após o seu entendimento, utilizar o BIP II para ampliar a abrangência dos conceitos abordados.

6.3.2 Computação Básica

Esta disciplina tem por objetivo apresentar uma visão geral da Ciência da Computação aos alunos ingressantes no curso, com ênfase no estudo dos princípios de funcionamento dos sistemas computacionais.

Nesse contexto, o BIP I passou a ser utilizado como processador de referência para ilustrar os conceitos de arquitetura e de organização de computadores, permitindo um melhor entendimento por parte dos alunos a respeito dos atributos arquiteturais de um processador e da diferença entre os conceitos de arquitetura e organização.

No estudo da arquitetura, foram apresentados os principais conceitos relacionados, utilizando o BIP I para exemplificar os diferentes atributos arquiteturais, como, por exemplo, o tamanho da palavra de dados, o tamanho da palavra de instrução, tipos de dados, formato de instrução, modos de endereçamento, registradores e conjunto de instrução. Uma vez feita a explanação desses conceitos e o estudo da arquitetura do BIP I, foram apresentadas as relações entre comandos de alto nível e comandos na

linguagem de montagem do BIP I. Após, os alunos realizaram exercícios de fixação similares aos ilustrados na Sub-seção 6.3.1.

Na sequencia, para reduzir o nível de abstração, foram mostrados os códigos binário equivalentes de alguns programas-exemplo, o que permitiu ilustrar o conceito de linguagem de máquina, processo de geração de código executável e o papel do compilador e do montador nesse processo.

Com relação ao estudo de organização de computadores, o BIP I foi utilizado para ilustrar o funcionamento de um processador. O grau de abstração adotado nesse estudo foi definido considerando que os alunos do primeiro período não possuem conhecimentos sobre circuitos digitais. Inicialmente, foram explicadas as funcionalidades dos blocos construtivos do BIP I, sem preocupação com a sua estrutura interna. Em seguida, foi mostrada a construção do processador para a execução de cada instrução e, por fim, foi apresentado um modelo simplificado da organização do BIP I (ilustrado na Figura 6.8). Foram realizadas atividades de fixação em que os alunos foram orientados a destacar os componentes e os canais do processador utilizados na execução de cada instrução, conforme ilustrado na Figura 6.9.

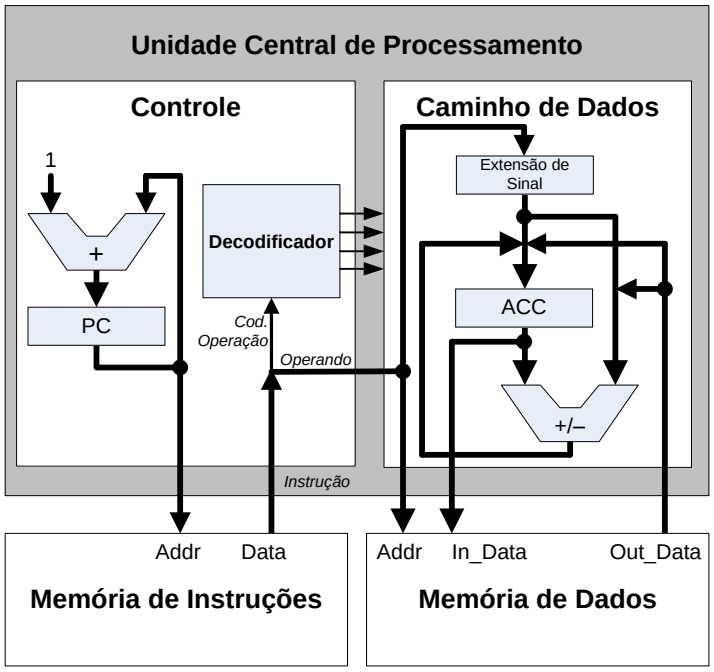

Figura 6.8. Organização Harvard monociclo do BIP I (visão simplificada)

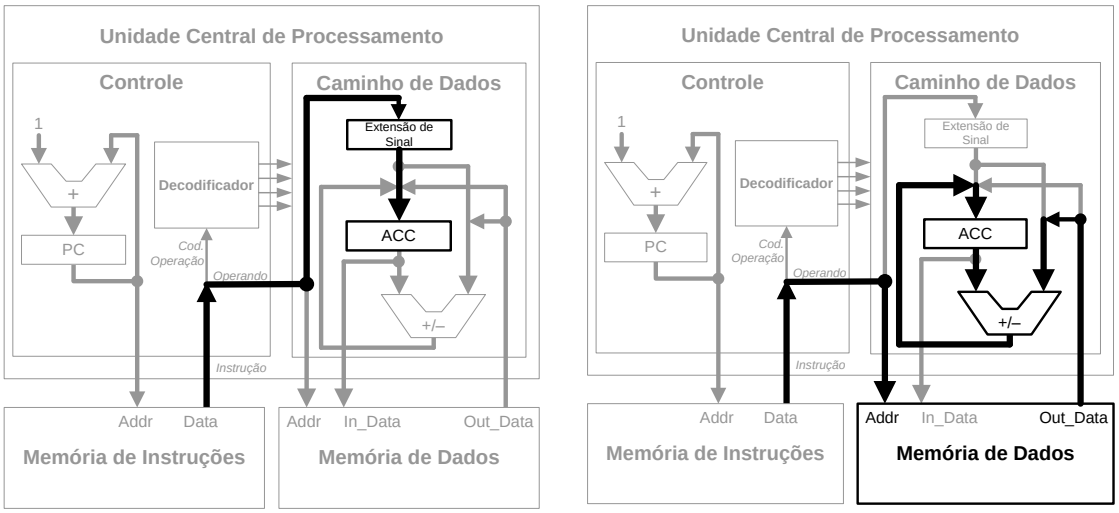

Figura 6.9. Exercício sobre a organização do BIP I: (a) execução da instrução LDI; (b) execução da instrução ADD

A partir dos exercícios de análise da execução das instruções na organização do BIP I, os alunos demonstraram entendimento básico a respeito da operação do processador e dos diferentes níveis de abstração: arquitetural e organizacional.

6.3.3 Circuitos Digitais

Esta disciplina tem por objetivo caracterizar e aplicar os fundamentos da Eletrônica Digital para aplicações em análise de circuitos utilizados em um computador. Nela, o aluno aprende as técnicas necessárias para implementar os blocos construtivos básicos de um processador, como, por exemplo, registradores, somadores, decodificadores e multiplexadores, entre outros.

Nesta disciplina, o BIP I foi utilizado para exemplificar o uso desses blocos construtivos para a construção de um sistema digital do tipo processador programável. Como a disciplina é baseada em ferramentas de captura de esquemático e de simulação lógica, foi aplicada uma atividade na qual os alunos deviam implementar o BIP I utilizando essas ferramentas e analisar o funcionamento do processador através de diagramas de forma de onda produzidos pelo simulador.

Diversos alunos já haviam estudado o BIP I nas disciplinas de Computação Básica e/ou de Algoritmos e Programação, o que facilitou a realização da atividade. No entanto, mesmo para aqueles que não haviam tido a oportunidade de estudar o BIP I, a compreensão acerca da sua arquitetura e da sua organização também foi facilitada pela simplicidade do processador.

O processador permitiu que os alunos vislumbrassem a aplicação dos conceitos estudados na disciplina aproximando-a da disciplina seguinte no curso (Arquitetura e Organização de Computadores). Ao mesmo tempo, essa atividade teve o efeito de um projeto integrador, incentivando os alunos a resgatar conceitos já estudados em outras disciplinas.

6.3.4 Arquitetura e Organização de Computadores I

Esta disciplina tem por objetivo geral ampliar a visão sobre arquitetura e organização de computadores, com vistas à programação na linguagem de montagem de um processador, identificando aspectos do projeto da sua organização. A disciplina utiliza o livro "Organização e Projeto de Computadores" de Patterson e Hennessy (2005) como livro-texto e, portanto, adota o processador MIPS como arquitetura de referência ao longo do semestre.

No entanto, no momento em que os conceitos de arquitetura e de organização começam a ser tratados, os processadores BIP são utilizados como referência inicial para ilustrar esses conceitos e resgatar o aprendizado obtido em disciplinas anteriores do Curso. Posteriormente, ao se realizar o estudo do MIPS, é feita uma análise comparativa das características arquiteturais e organizacionais dos dois processadores. Procura-se também discutir as diretrizes de projeto do BIP, demonstrando que o foco na simplicidade do processador lhe impõe limitações quando comparado a um processador cujo projeto é focado no desempenho, como o MIPS.

O papel do BIP nesta disciplina é diferente do seu papel nas disciplinas das fases iniciais. A sua contribuição está em permitir um aumento incremental da complexidade dos temas abordados ao longo do semestre. Como a arquitetura do MIPS não foi especificada visando o aprendizado, mas sim a facilidade de implementação do processador e o seu desempenho, iniciar o estudo dos conceitos de arquitetura e de organização a partir do BIP tende a facilitar o entendimento inicial desses conceitos.

6.3.5 Compiladores

A utilização da Família BIP na disciplina de compiladores proporcionou uma alternativa para demonstrar todas as fases de construção do compilador seguindo a abordagem sugerida por Lins (2000), na qual inicia-se definindo a arquitetura da linguagem alvo para então se definir os aspectos sintáticos e léxicos da linguagem de alto nível.

O conjunto de instruções do BIP mostrou-se bastante simples, permitindo a geração direta do código através de ações semânticas sem passar por linguagens intermediárias, conforme ilustra a Tabela 6.10. Essa característica possibilitou a substituição da linguagem intermediária que vinha sendo utilizada até então (Linguagem da Máquina Virtual de Pilha), viabilizando à maioria dos alunos concluir a etapa de geração de código (o que raramente ocorria em semestres anteriores).

Para ilustrar melhor a simplicidade do processo de geração de código, na Tabela 6.10, é demonstrada a geração de um desvio condicional para a linguagem Portugol. Foi utilizada a ferramenta ANTLR3 (Parr, 2009) para apoiar a geração do analisador léxico e sintático.

Tabela 6.10. Ações Semânticas para a geração de desvios condicionais

Portugol	**Ações Semânticas**	***Assembly***

`Se (x > 8) entao`
`  x <- 0`
`fimse`
	1. cmd_se: SE expRelacional
	2. {
	3. add_instrucao("LD", resultado_Esq);
	4. add_instrucao("SUB",resultado_Dir);
	5. add_instrucao(opRelacional,"ENDIF1”);
	6. rotulos.Push("ENDIF1");
	7. }
	8. ENTAO lista_cmdo FIMSE
	9. {
	10. add_instrucao(rotulos.Pop());
	11. }
	12. ;
		`  LD  x`
		`  STO $100`
		`  LDI 8`
		`  STO $101`
		`  LD  $100`
		`  SUB $101`
		`  BLE ENDIF1`
		`  LDI 0`
		`  STO x`
		`ENDIF1:`

Para a realização do desvio condicional, inicialmente, são resolvidas as expressões dos lados esquerdo e direito do operador relacional. Os resultados dessas expressões são armazenados em variáveis temporárias e, posteriormente, o resultado da expressão do lado direito é subtraído do valor resultante da expressão do lado esquerdo do operador (linhas 3 e 4). As instruções de desvio do BIP baseiam-se na execução prévia de uma instrução de subtração entre os valores comparados e na análise do registrador STATUS que armazena informações sobre o resultado da última operação. De acordo com o valor resultante dessa subtração, a instrução de desvio condicional determina se o desvio deve ser tomado ou não.

A instrução de desvio é gerada de acordo com o operador relacional utilizado (linha 4). O nome do rótulo é colocado em uma pilha (linha 5) para utilização ao final da expressão (linha 9).

O método `add_instrucao` (linha 10) adiciona a instrução em uma estrutura de dados que será utilizada para a geração de código e posterior uso na simulação do programa.

Alguns trabalhos iniciados na disciplina tiveram continuidade na forma de trabalhos de conclusão de curso. Uma das ferramentas desenvolvidas (descrita a seguir) consistiu de um simulador que será utilizado futuramente na disciplina.

6.3.6 Trabalhos de Conclusão de Curso

Dois trabalhos de conclusão de curso relacionados ao BIP já foram realizados no Curso de Ciência da Computação da UNIVALI. Em um deles, foi desenvolvido um microcontrolador baseado na arquitetura BIP (o μBIP). Em outro, foi desenvolvida uma IDE (Integrated Development Environment) denominada BipIde que inclui simuladores do BIP I e do BIP II. Esses trabalhos são relatados resumidamente a seguir.

6.3.6.1 μBIP

Visando estender o uso do BIP para disciplinas avançadas nas áreas de projeto de sistemas digitais e de sistemas embarcados, foi desenvolvido um microcontrolador baseado na arquitetura BIP, o qual foi denominado μBIP (lê-se *microbip*)

[Pereira 2008][Pereira e Zeferino 2008]. Esse microcontrolador estende as características dos processadores BIP, agregando novas instruções, periféricos e funcionalidades típicas de microcontroladores.

Para a especificação deste microcontrolador, foi feita uma pesquisa junto a professores e engenheiros de desenvolvimento de hardware de empresas de base tecnológica. Essa consulta permitiu definir as características de uma primeira versão do microcontrolador e estabelecer um *roadmap* para futuras versões.

Para essa primeira versão, foram selecionadas e implementadas as seguintes características:

- Suporte a endereçamento de vetores;
- Suporte a chamadas de procedimentos;
- Suporte a interrupções;
- Portas de E/S bidirecionais; e
- Temporizador programável com *prescaler*.

Para tal, foi necessário incluir hardware adicional no processador, como um registrador para endereçamento indireto de vetores (INDR), uma pilha em hardware para salvamento de contexto e um controlador de interrupções, além dos periféricos selecionados. A pilha em hardware é usada para salvar o contexto do programa nas chamadas de procedimentos e no atendimento de uma interrupção. Utilizando-se uma abordagem similar à adotada nos microcontroladores PIC® da Microchip (2003), apenas o valor de PC+1 é salvo na pilha, cujo topo (ToS – Top-of-Stack) é apontado pelo registrador SP (Stack Pointer).

A Figura 6.10 ilustra a organização do μBIP. Nessa figura, podem ser observados alguns dos novos componentes de hardware, sendo que os periféricos estão representados pelo bloco SFR (Special Function Registers).

O conjunto de instruções do BIP III foi estendido para oferecer o suporte arquitetural para uso desses recursos. As novas instruções, descritas na Tabela 6.11, incluem: (*i*) duas instruções para manipulação de vetor baseadas no modo de endereçamento indireto – STOV (*store vector*) e LDV (*load vector*); (*ii*) uma instrução de chamada de procedimentos (CALL); e (*iii*) duas instruções de retorno de procedimento (RETURN e RETINT), sendo que RETINT (*return from interrupt*) serve para realizar o retorno de rotinas de atendimento a interrupções.

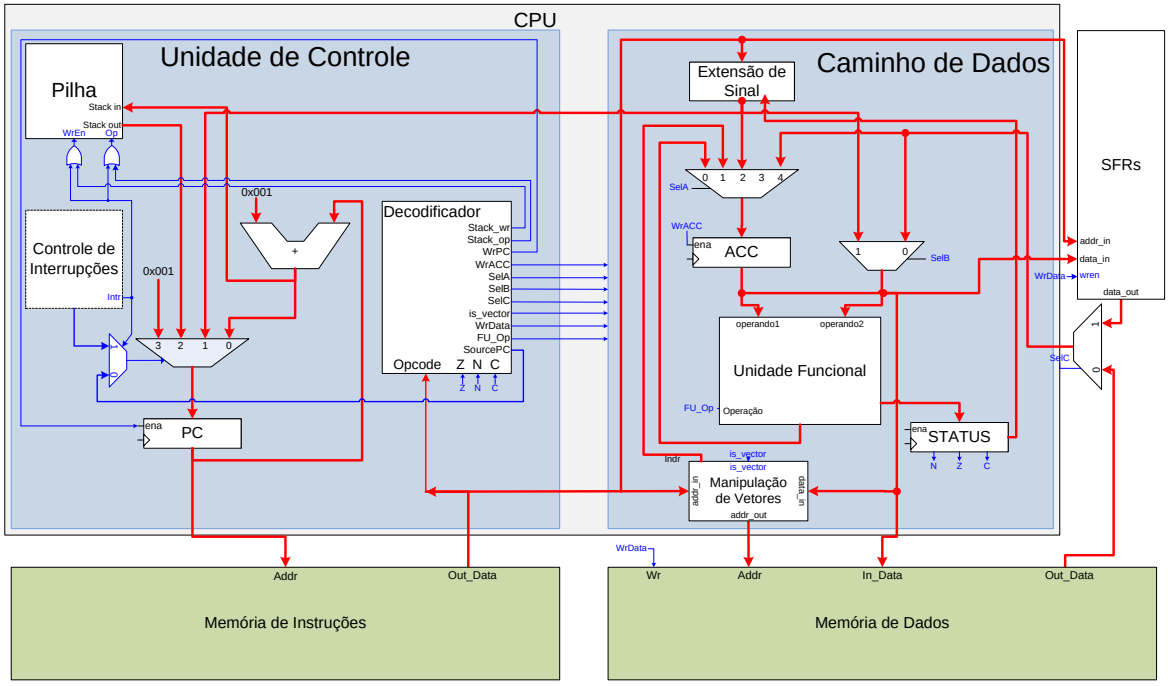

Figura 6.10. Organização do µBIP

Tabela 6.11. Conjunto de instruções estendido pelo µBIP

Código da Operação	Instrução	Operação	Classe
11000	STOV operando	Memória[operando + INDR] ← ACC	Manipulação de vetor
11001	LDV operando	ACC ← Memória[operando + INDR]	Manipulação de vetor
11010	RETURN	PC ← ToS	Suporte a procedimentos
11011	RETINT	PC ← ToS	Suporte a procedimentos
11100	CALL operando	PC ← operando ToS ← PC+1	Suporte a procedimentos

O µBIP foi desenvolvido utilizando-se o ArchC [The ArchC Team 2007] como suporte para especificação arquitetural e geração de código. O ArchC é uma linguagem de descrição de arquitetura (Architecture Description Language, ADL) baseada no SystemC e que foi desenvolvida pelo Instituto de Computação da Universidade de Campinas (IC-UNICAMP). O ArchC permite descrever a arquitetura do processador e a sua hierarquia de memória, gerando, automaticamente, ferramentas como um simulador de conjunto de instruções, um montador, um ligador e um depurador, as quais foram geradas para o µBIP. A especificação arquitetural foi validada com base em testes unitários das instruções e na execução de aplicações especificadas no testbench do Dalton Project [The Dalton Project Team 2001].

Além das ferramentas geradas com o auxílio do ArchC, no âmbito do trabalho de conclusão de curso no qual o μBIP foi desenvolvido, também foi implementado um modelo do microcontrolador na linguagem de descrição de hardware VHDL, o qual foi sintetizado e prototipado em FPGA (Field Programmable Gate Array). O protótipo físico foi validado executando-se as aplicações do Dalton Project.

O uso do BIP como referência para este trabalho mostrou que, apesar de suas limitações arquiteturais, definidas de modo a favorecer o aprendizado, a arquitetura do BIP pode ser estendida para incluir funcionalidades tipicamente encontradas em processadores comerciais.

O μBIP ainda não foi aplicado em atividades de ensino. Por enquanto, estão sendo realizadas atividades de desenvolvimento no sentido de integrar periféricos de comunicação serial ao microcontrolador. No futuro, pretende-se utilizar a experiência obtida nesses desenvolvimentos para elaborar roteiros de projetos a serem realizados em disciplinas avançadas na área de projeto de sistemas computacionais, conforme já mencionado.

6.3.6.2 BipIde

Neste trabalho de conclusão de curso [Viera 2009][Vieira, Raabe e Zeferino 2010], foi implementada uma IDE para desenvolvimento de pequenos algoritmos em Portugol e sua conversão na linguagem de montagem do BIP. A IDE inclui um simulador da arquitetura e da organização do BIP II que possibilita aos alunos testarem seus programas, acompanhando as ações realizadas no nível da organização do processador. A Figura 6.11 (mostrada na próxima página) ilustra a interface do BipIde, cujos recursos são descritos a seguir.

1. Simulação: apresenta os botões que permitem ao usuário controlar a simulação do programa e verificar sua execução passo-a-passo, incluindo as ações a seguir:
    - Simular: inicia a simulação de um programa;
    - Pausar: interrompe temporariamente a simulação do programa;
    - Parar: interrompe a simulação do programa;
    - Continuar: continua a simulação do programa até o seu final;
    - Repetir: repete a última instrução simulada; e
    - Próximo: simula a próxima instrução do programa.
2. Velocidade: através desta opção, o usuário pode controlar a velocidade com que ocorre a simulação do programa;
3. Simulador de instruções: o usuário pode simular uma instrução específica do processador, sem a necessidade de escrever um programa completo. Para isso, poderá ser digitado o valor do operando no campo específico e pressionado o botão correspondente à instrução que deverá ser simulada;
4. Portugol: exibe o programa Portugol que está sendo simulado. Neste editor, a linha do programa que está sendo simulada aparece destacada;

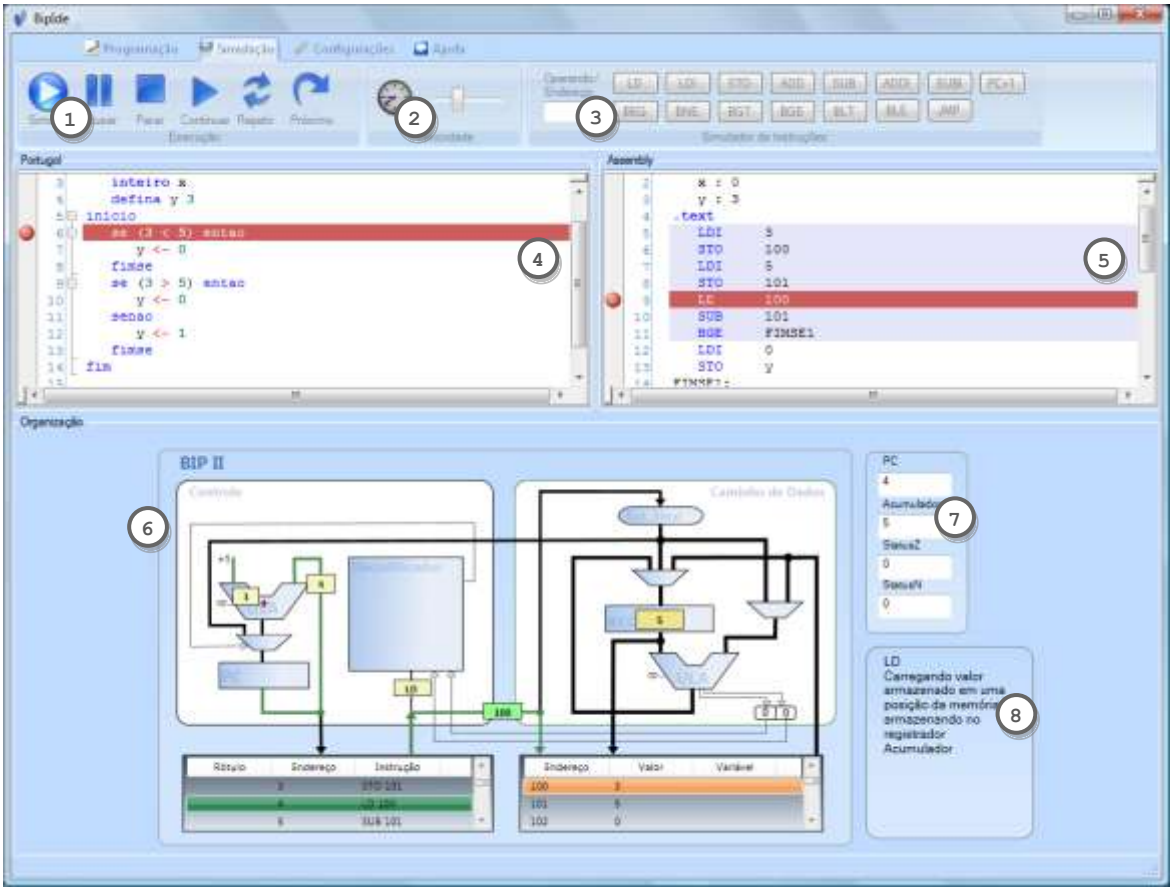

Figura 6.11. Interface do BipIde

5. Assembly: exibe o código *assembly* gerado pelo compilador. Neste editor, a linha do programa que está sendo simulada aparece destacada. É destacado também o bloco de instruções *assembly* que correspondem à linha do código em alto nível destacada na janela Portugol, permitindo ao usuário verificar quais instruções *assembly* foram geradas para executar uma instrução Portugol. No programa da Figura 6.11, foram geradas sete linhas de código *assembly*, para executar a instrução Portugol correspondente em destaque;
6. Organização do processador: exibe a imagem da organização do processador (BIP I ou BIP II), sobre a qual são feitas animações que representam a execução das instruções no processador. A fim de facilitar o entendimento do simulador, os seus principais componentes são nomeados e, com dois cliques sobre eles, é exibida uma janela informativa a respeito do componente;
7. Registradores: exibe os valores dos registradores do processador durante a simulação do programa; e
8. Descrição: exibe o nome da instrução simulada e uma descrição da ação ocorrida, a fim de auxiliar o usuário a compreender o que está ocorrendo no processador.

A ferramenta será utilizada para apoiar a apresentação do BIP I e BIP II na disciplina Algoritmos e Programação do primeiro período e também na disciplina Compiladores para fornecer um ambiente de simulação para o código gerado pelos compiladores desenvolvidos pelos alunos.

6.4 Considerações finais

A abordagem apresentada neste capítulo enfatizou uma visão multi e interdisciplinar do ensino de conceitos de arquitetura de computadores, sendo que o desenvolvimento da família de processadores levou em conta as necessidades das diferentes disciplinas envolvidas. Essa característica configura-se como principal diferencial desta proposta.

As atividades de aprendizagem já conduzidas apontaram diversos aspectos positivos. Os principais são:

1. Uma integração mais natural entre os conceitos de programação, circuitos digitais, arquitetura de computadores e compiladores;
2. A possibilidade de utilizar um mesmo modelo de computação durante boa parte do percurso do aluno na graduação;
3. A redução de problemas de aprendizagem relacionados com as abstrações envolvidas nos conceitos introdutórios de programação, tais como: variáveis, constantes e atribuições;
4. Uma ilustração melhor fundamentada aos alunos dos períodos iniciais da relação entre os desvios, laços, a linguagem de montagem e o funcionamento do processador;
5. Facilidade na consolidação de conceitos de arquitetura e organização de computadores pelo uso de uma arquitetura de referência em diferentes disciplinas;
6. Consolidação dos conceitos de circuitos digitais pelo uso do BIP como projeto integrador ao final da disciplina;
7. Ao utilizar o conjunto de instruções do BIP, a etapa de geração de código dos trabalhos de compiladores foi concluída pela maioria dos alunos;
8. Após a experiência no uso do BIP, iniciou-se um processo de reflexão acerca do melhor encadeamento dos conteúdos entre as disciplinas envolvidas. Como resultado dessa reflexão, foram identificados aspectos desejáveis que serão buscados em trabalhos futuros:
    - Rever a organização curricular do curso buscando reduzir a distância entre a disciplina Arquitetura de Organização de Computadores e a disciplina Compiladores;
    - Promover o uso dos processadores BIP em outras disciplinas, como, por exemplo, a disciplina de Sistemas Operacionais; e
    - Conduzir experimentos controlados a fim de obter evidências empíricas sobre a redução dos problemas de aprendizagem associados à necessidade de abstração.

6.5 Agradecimentos

Os autores agradecem ao CNPq pelo apoio concedido para realização da pesquisa via Edital Universal 2008, Processo No 477820/2008-5, e à UNIVALI pelo contínuo suporte às atividades desta pesquisa.

6.6 Referências

Carbone, A. e Kaasboll, J. (1998) "A survey of methods used to evaluate computer science teaching". In: Proceedings of the 3rd Conference on Teaching of Computing, Dublin, p. 41-45.

Clements, A. (1999) "Selecting a processor for teaching computer architecture", Microprocessor and Microsystems, v.23, n.5, p. 281-290.

Good, J. e Brna, P. (2004) "Program comprehension and authentic measurement: a scheme for analyzing descriptions of programs", Journal of Human-Computer Studies, v.61, n.2, p. 169-185.

Hostins, H. e Raabe, A. L. A. "Auxiliando a aprendizagem de algoritmos com a ferramenta Webportugol". In: XIV Workshop de Educação em Computação – XXVII Congresso da SBC, 2007, Rio de Janeiro. Anais do XXVII Congresso da SBC, 2007. v. 1. p. 96-105.

Khalife, J. T. (2006) "Threshold for the introduction of Programming: Providing Learners with a Simple Computer Model" In: Proceedings of the 28th International Conference on Information Technology Interfaces, p. 71-76.

Lins, R. D. (2000) "Uma Proposta de Plano Pedagógico para a matéria de Compiladores". In: II Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática, Congresso Anual da SBC.

Menezes, C. S. e Nobre, A. M. (2002) "Um ambiente cooperativo para apoio a cursos de introdução a programação". In: Workshop de Educação em Computação, Anais do XXII Congresso da Sociedade Brasileira de Computação. Porto Alegre: SBC.

Microchip (2003) PIC16F62X Data Sheet: FLASH-Based 8-Bit CMOS Microcontroller. Arizona.

Morandi, D., Raabe, A. L. A. e Zeferino, C. A. (2006) "Processadores para ensino de conceitos básicos de Arquitetura de Computadores" In: Anais do 1º Workshop sobre Educação em Arquitetura de Computadores, p. 17-24.

Morandi, D., Pereira, M. C., Raabe, A. L. A. e Zeferino, C. A. (2006) "Um processador básico para o ensino de conceitos de arquitetura e organização de computadores". Revista Hífen, v. 30, n. 58, p. 73-80.

Parr, T. (2009) ANTLR v3 Documentation. Disponível em: <http://www.antlr.org>.

Patterson, D. A. e Hennessy, J. L. (2005) Organização e projeto de computadores: a interface hardware/software, São Paulo, Campus.

Pereira, M. C. e Zeferino, C. A. (2008) “uBIP: a simplified microcontroller architecture for education in embedded systems design. In: Proceedings of the IP Based

Electronic System Conference & Exhibition - IP 08, Grenoble : Design and Reuse, p. 193-197.

Pereira, M. C. (2008) BIP: Microcontrolador Básico para o Ensino de Sistemas Embarcados. Trabalho de Conclusão de Curso, Ciência da Computação, Universidade do Vale do Itajaí.

Pimentel, E. P., Franca, V. F., Noronha, R. V. e Omar, N. (2003) "Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores". In: Workshop de Informática na Escola, Anais do XXIII Congresso da Sociedade Brasileira de Computação. Porto Alegre: SBC.

The ArchC Team (2007) The ArchC project home page, Disponível em: <http://www.archc.org >.

The Dalton Project Team (2001) The UCR Dalton Project, University of California, Riverside. Disponível em: < http://www.cs.ucr.edu/~dalton/ >.

Vieira, P. V. (2009) BipIde: Ambiente de Desenvolvimento Integrado para a Arquitetura dos Processadores BIP. Trabalho de Conclusão de Curso, Ciência da Computação, Universidade do Vale do Itajaí.

Vieira, P. V., Raabe, A. L. A. e Zeferino, C. A. (2010) Bipide: ambiente de desenvolvimento integrado para a arquitetura dos processadores BIP. Revista Brasileira de Informática na Educação, v. 18, p. 32-43.

Weber, R. F. (2004) Fundamentos de arquitetura de computadores, Porto Alegre, Sagra Luzzatto.