Implementando Suporte a Novas Linguagens de Programação e outros Idiomas no Ambiente de Desenvolvimento Integrado Bipide

Nereu P. de Oliveira Junior[1], André L. A. Raabe[1, 2], Cesar A. Zeferino[1, 2]
[1]Bacharelado em Ciência da Computação
[2]Programa de Pós-Graduação em Computação
Universidade do Vale do Itajaí – Univali
{nereu, raabe, zeferino}@univali.br

Resumo

A ferramenta Bipide foi desenvolvida para auxiliar na redução da abstração envolvida em conceitos fundamentais de lógica de programação. Ela possibilita a programação de pequenas aplicações em Portugol, a geração de código na linguagem de montagem da família de processadores BIP e a simulação das aplicações ilustrando conceitos de arquitetura e organização de computadores. Este artigo apresenta melhorias que foram feitas nessa ferramenta. Para ampliar o seu escopo de uso em disciplinas de cursos de graduação na área de Computação, foi integrado um compilador C. Para facilitar sua adoção em outros países, foi acrescentado o suporte a idiomas estrangeiros. Além disso, foram realizadas correções e melhorias no simulador da ferramenta para implementar o suporte ao microcontrolador da família BIP, o µBIP.

1. Introdução

Um problema encontrado em muitos cursos de graduação em Computação é que as disciplinas da área de Algoritmos e Programação são ministradas antes das disciplinas da área de Arquitetura e Organização de Computadores. Isso faz com que alguns conceitos de programação como, por exemplo, constantes, variáveis, vetores, níveis de linguagem entre outros, não sejam completamente assimilados pelos estudantes devido ao alto nível de abstração adotado nas disciplinas de Programação [1].

Para aprimorar o relacionamento entre os conceitos das disciplinas de Algoritmos, Programação e Arquitetura e Organização de Computadores, foi especificada uma família de processadores com arquitetura simples, denominada BIP (Basic Instruction-set Processor) e foi desenvolvido um ambiente integrado para programação e simulação desses processadores, denominado Bipide (BIP Integrated Development Environment).

O Projeto BIP busca ser referência para o ensino de conceitos da área de Arquitetura e Organização de Computadores para os alunos de fases iniciais de cursos de graduação em Computação. O projeto visa também prover a base necessária para a compreensão das abstrações adotadas nas disciplinas que ensinam Algoritmos e Programação [1].

A primeira geração do processador BIP (BIP I) oferece o suporte necessário para ilustrar os conceitos de instrução, declaração de variáveis e constantes, execução de operações aritméticas e de atribuição de valores. Permite também ilustrar o papel do compilador e o conceito de desempenho. O BIP II estende a arquitetura do BIP I acrescentando o suporte necessário para ilustrar a execução de desvios e laços de repetição [2]. No BIP III, foi adicionado o suporte a operações de lógica bit-a-bit [3]. Já o BIP IV permite realizar operações de entrada e saída e chamadas de procedimentos. Também no BIP IV foram incluídas instruções dedicadas à manipulação de vetores [3]. A família BIP inclui ainda o microcontrolador µBIP [4] (lê-se micro BIP), o qual adiciona suporte a interrupções para comunicação com periféricos, típicas em microcontroladores.

A Versão 3.0 do Bipide [3] permite o desenvolvimento, a execução e a simulação de programas escritos na linguagem Portugol, relacionando esses programas à arquitetura dos processadores BIP. A interface do Bipide é composta de: (*i*) um editor de textos para escrita dos programas; (*ii*) um compilador que realiza a tradução do código fonte em Portugol para a linguagem de montagem dos processadores BIP; (*iii*) um simulador que ilustra a execução do programa no processador; e (*iv*) um módulo de ajuda com informações sobre os processadores BIP e o ambiente Bipide [5].

Apesar das diversas funcionalidades da Versão 3.0, foram identificadas duas oportunidades de melhoria

importantes no Bipide: (*i*) a implementação de um compilador para a linguagem C que viabilize seu uso em mais disciplinas; e (*ii*) a adição de um suporte multi-idioma, de modo a possibilitar sua adoção em instituições de ensino de outros países.

Neste artigo, é apresentada a Versão 4.0 do Bipide, a qual foi desenvolvida para promover as melhorias supracitadas com a inclusão de um compilador C e de soluções para suporte multi-idioma proporcionando que novos idiomas sejam facilmente acrescentados à ferramenta. Essa nova versão inclui também o microcontrolador µBIP (não suportado na Versão 3.0).

O restante do artigo é organizado da seguinte forma. A Seção 2 apresenta detalhes sobre o desenvolvimento da nova versão do Bipide. A Seção 3 apresenta aspectos referentes a resultados de validação. A Seção 4 apresenta as considerações finais.

2. Desenvolvimento

2.1. Suporte à novos compiladores

No projeto da integração do Compilador C com o Bipide, definiu-se uma representação intemediária comum para que novos compiladores pudessem ser acrescentados no futuro. O requisito fundamental é que qualquer novo compilador deve gerar uma Árvore Sintática Abstrata (ASA), seguindo o formato do Portugol Núcleo [7]. Dessa forma, todas as funcionalidades desenvolvidas para linguagem Portugol (verificação semântica, geração de código *assembly*, otimização e interpretação) seguirão funcionando para cada novo compilador acoplado. Além disso, aprimoramentos e novas funcionalidades que forem realizadas com base na ASA do Portugol Núcleo funcionarão para todos compiladores acoplados.

Foi definido, também, um arquivo de configuração para cada novo compilador acoplado. O Bipide carrega esse arquivo para exibir na sua inteface gráfica as informações das linguagens que suporta. O arquivo de configuração armazena informações como: (*i*) nome da linguagem de programação; (*ii*) caminho do executável do compilador (*iii*) extensão do arquivo; (*iv*) caminho para o salientador de sintaxe da linguagem; (*v*) trecho de código modelo (usado como base inicial nos programas); e (*vi*) exemplos de programas escritos na linguagem.

2.2. O compilador C

No desenvolvimento do compilador C, foram utilizadas as seguintes ferramentas: (*i*) ANTLR, para a criação da gramática; (*ii*) Visual Studio, para a criação do pré-compilador; e (*iii*) NetBeans, para implementação das ações semânticas para geração da ASA.

A gramática foi definida de forma a incorporar todas as palavras reservadas da linguagem C mencionadas a seguir, as quais são pertencentes ao padrão ANSI:

- *Palavras reservadas*: As palavras reservadas selecionadas para compor o conjunto de instruções do compilador C foram baseadas nas instruções do padrão ANSI e nas instruções suportadas pela ASA do Portugol Núcleo. As instruções que se enquadram neste critério de seleção são: `if`, `else`, `switch`, `case`, `default`, `goto`, `for`, `do`, `while`, `break`, `continue` e `return`;
- *Operadores*: Operadores aritméticos (soma, subtração), operadores relacionais e operadores lógicos (AND, OR, NOT). Os operadores de multiplicação e de divisão não foram implementados pois não têm suporte na arquitetura do BIP;
- *Tipos de dados*: São suportados os tipos de dado inteiro e vazio. O primeiro é utilizado apenas em assinaturas de métodos para identificar que uma função não possui retorno. O tipo inteiro pode ser utilizado em declarações de variáveis (escalares e vetores) e na assinatura de métodos para indicar que a função possui um retorno do tipo inteiro. As variáveis do tipo inteiro podem ser expressas nas bases decimal, hexadecimal e binária;
- *Entrada e Saída*: Os métodos de entrada e saída da linguagem não fazem parte da gramática em si. Porém, tais métodos foram previamente definidos e mapeados no analisador semântico para evitar a necessidade de implementar suporte a bibliotecas. Sendo assim `scanf` (entrada) e `printf` (saída) são procedimentos que foram adicionados a uma lista de exceção no analisador semântico para que um erro não seja gerado quando o varredor da linguagem se deparar com algum desses métodos. Posteriormente, o gerador de código identifica os métodos de entrada e/ou saída para realizar o tratamento adequado com cada um deles.

2.3. Suporte multi-idioma

Da mesma forma que o suporte a novos compiladores, o suporte multi-idioma foi implementado de forma dinâmica para possibilitar o acoplamento de novos idiomas.

O suporte a outros idiomas já havia sido parcialmente implementado na Versão 3.0. Porém, as mensagens de erro e de sucesso, assim como o menu de ajuda, eram apresentadas somente em português, independentemente do idioma selecionado na ferramenta. Outra característica da Versão 3.0 era que as traduções dos idiomas estavam embutidas no código fonte, tornando assim a tradução da ferramenta restrita a quem possuía o código fonte.

Na Versão 4.0, o suporte multi-idioma foi desenvolvido de forma corrigir e contornar essas limitações. Para que isso fosse possível, foram utilizados arquivos no formato CSS (Cascading Style Sheets), HTML (HyperText Markup Language), JavaScript), XAML (eXtensible Application Markup Language) e XML. O modo com que esses arquivos foram empregados para constituir o suporte multi-idioma da ferramenta é descrito a seguir.

2.3.1. Interface de usuário. Para oferecer o suporte multi-idioma na interface de usuário, foi utilizada a técnica de dicionário de recursos. O dicionário de recursos é implementado em formato XAML e todo atributo do arquivo contém um conteúdo e uma chave por onde a busca do conteúdo é realizada. Nessa situação, os conteúdos tratam das traduções referentes aos componentes da interface como, por exemplo, botões, menus e abas, entre outros.

2.3.2. Dicas de função. Assim como os componentes da interface, as dicas de função foram implementadas utilizando-se do dicionário de recursos. Apesar de se empregar da mesma técnica que os componentes de interface, foi tomada a decisão de separar as dicas de função em outro arquivo, tendo em vista que os contextos de utilização dos dois arquivos são diferentes e a visualização da estrutura dos arquivos como um todo fica melhor.

2.3.3. Ajuda. A técnica utilizada para permitir com que a ajuda do sistema pudesse oferecer suporte multi-idioma foi diferente da técnica de dicionários de recursos empregada no sistema. A Ajuda foi desenvolvida utilizando-se de arquivos nos formatos HTML, JavaScript, CSS e XML, nos quais todo o conteúdo e recurso visual utilizados na ajuda são construídos por meio do HTML, JavaScript e CSS. O XML é utilizado para definir a hierarquia com a qual os arquivos HTML serão exibidos para o usuário.

2.3.4. Mensagens de erro e sucesso. Para exibir as mensagens de acordo com o idioma selecionado, foi necessário incluir as mensagens em arquivos XAML, um para cada idioma suportado. Os arquivos contendo as mensagens de erro e sucesso funcionam da mesma maneira que os dicionários de recurso citados anteriormente.

2.3.5. Simulador. Outra funcionalidade do Bipide que sofreu alterações em decorrência da implementação do suporte multi-idioma, foi o simulador. Para que fosse possível a alteração de idioma de forma dinâmica, foi necessário implementar novas funcionalidades e realizar alterações no projeto do simulador. Com as alterações realizadas, foi possível transferir ao simulador o dicionário de recursos que o mesmo deve utilizar, de acordo com o idioma selecionado.

2.3.6. Mapeamento geral. Para que seja possível utilizar todo o conteúdo referente a idiomas citado anteriormente, esse conteúdo precisa estar centralizado e mapeado para que as opções de idioma possam ser apresentadas aos usuários, possibilitando assim que os usuários possam selecionar o idioma que mais atende às suas necessidades. Devido a esses fatores, foi criado um XML de mapeamento geral que é lido pela aplicação e transformado em um componente visual que permite realizar a seleção do idioma desejado.

2.4. Suporte ao µBIP

Para que nova versão do Bipide pudesse suportar o microcontrolador µBIP algumas alterações e melhorias foram necessárias, como a inclusão de um pré-compilador e de extensões no simulador.

2.4.1. O pré-compilador

Para que fosse possível aproveitar ao máximo todo o potencial do µBIP, foi necessário suportar o acesso direto a registradores e permitir a inclusão de código *assembly* no código de alto nível por meio de diretivas específicas. Para isso, foi implementado um pré-compilador da linguagem C.

As diretivas utilizadas para o pré-compilador foram definidas de modo a facilitar o reaproveitamento de código e permitir com que bibliotecas de função sejam escritas para realizar operações comuns de acesso e configuração de registradores.

Para as operações que o µBIP precisa realizar, foram definidas as seguintes diretivas: (*i*) `#include`, que permite referenciar uma biblioteca e utilizar suas funções; (*ii*) `#define`, que permite definir constantes e macros; e (*iii*) `#asm` e `#endasm`, que delimitam um trecho de código em *assembly* incorporado ao código fonte em C.

2.4.2. Alterações no simulador

Para que o simulador do Bipide pudesse suportar o µBIP, foi incluído o bloco do controlador de interrupções, o qual é responsável por desviar o fluxo de execução do programa para o endereço da rotina de tratamento (0x001) quando um sinal de interrupção é ativado. Além disso, No painel direito do simulador, onde é disponibilizado acesso aos registradores do processador, foram adicionados os registradores de propósito específico do µBIP, a saber: `$port0_dir`, `$port1_dir`, `$tmr0_config`, `$tmr0_value`, `$int_config`, `$int_status` e `$mcu_config`.

3. Resultados

Para validação do compilador C, foi aplicado um *testbench* composto de 12 programas que foram utilizados na validação do Bipide 3.0 [3]. A esse *testbench*, foram acrescidos 13 programas para testar as novas funcionalidades. A Tabela 1 apresenta um exemplo de programa e o método de validação utilizado. A tabela mostra o código fonte de entrada, o código *assembly* gerado pelo compilador C e o estado esperado dos registradores e da memória ao final da execução do programa. O exemplo ilustra um procedimento de multiplicação de inteiros baseada em somas sucessivas, o qual é chamado pela função principal. O resultado é impresso na porta de saída.

Tabela 1. Exemplo de programa de validação

Código fonte C	Código ASM	PC	ACC	$out_port

`int mul(int a, int c)`
`{`
`  int i;`
`  int res = 0;`
 	
`  for (i=1;i<=c;i++)`
`   res = res + a;`
 	
`  return res;`
`}`
 	
`int main()`
`{`
`  int j = 2;`
`  int k = 3;`
 	
`  k = mul(k,j);`
 	
`  printf(k);`
`}`
	`.data`
	`  mul_a : 0`
	`  mul_c : 0`
	`  mul_i : 0`
	`  mul_res: 0`
	`  j : 2`
	`  k : 3`
	`.text`
	`  JMP _MAIN`
	`_MUL:`
	`  LDI 1`
	`  STO mul_i`
	`PARA1:`
	`  LD mul_i`
	`  SUB mul_c`
	`  BGT FIMPARA1`
	`  LD mul_res`
	`  ADD mul_a`
	`  STO mul_res`
	`  LD mul_i`
	`  ADDI 1`
	`  STO mul_i`
	`  JMP PARA1`
	`FIMPARA1:`
	`  LD mul_res`
	`  RETURN 0`
	`_MAIN:`
	`  LD k`
	`  STO mul_a`
	`  LD j`
	`  STO mul_c`
	`  CALL _MUL`
	`  STO k`
	`  STO $out_port`
	`  HLT 0`
		22	6	6

Os testes possibilitaram identificar erros de codificação que foram corrigidos. Ao final do processo de validação, todos os 25 problemas executados apresentaram os resultados esperados.

4. Conclusões

Esse artigo apresentou as alterações realizadas na ferramenta Bipide para ampliar o seu escopo e sua abrangência de utilização. Com isso, espera-se que a ferramenta possa ser utilizada em um maior número de disciplinas de cursos da área de Computação. Também se espera viabilizar a adoção da ferramenta em outros países.

A versão atual (4.0), disponível em http://bipide.com.br suporta os idiomas português e inglês e possui compladores para Portugol e C. Como trabalhos futuros, pretende-se avaliar o uso dessa nova versão em disciplinas das áreas de Algoritmos e Programação e de Sistemas Operacionais, realizar otimizações no compilador C e suportar outros idiomas e compiladores.

Referências

[1] D. Morandi, M. C. Pereira, A. L. A. Raabe, C. A. Zeferino, "Um Processador Básico para o Ensino de Conceitos de Arquitetura e Organização de Computadores", *Hífen*, Uruguaiana, v. 30, p. 73-80, 2006.

[2] D. Morandi, A. L. A. Raabe, C. A. Zeferino, "Processadores para Ensino de Conceitos Básicos de Arquitetura de Computadores", *1º Workshop de Educação em Arquitetura de Computadores*, Ouro Preto, 2006, p. 17-24.

[3] P. R. Rech, P. V. Vieira, C. A. Zeferino, A. L. A. Raabe, "BIP IV: Especificação e Suporte na Ferramenta Bipide", *6º Workshop de Educação em Arquitetura de Computadores*, Vitória, 2011.

[4] M. C. Pereira, C. A. Zeferino, "uBIP: a Simplified Microcontroller Architecture for Education in Embedded Systems Design", *IP-based Electronic System Conference & Exhibition*, Grenoble, 2008, p. 193-197.

[5] P. Vieira, A. L. A. Raabe, C. A. Zeferino, "Bipide: Ambiente de Desenvolvimento Integrado para a Arquitetura dos Processadores BIP", *Revista Brasileira de Informática na Educação*, v. 18, n. 1, p. 32-43, 2010.

[6] P. Mannes, *Integração do PortugolCore com o Bipide*, Trabalho de Conclusão de Curso (Graduação em Ciência da Computação), Universidade do Vale do Itajaí, Itajaí, 2013.

[7] L. F. Noschang, E. A. De Jesus, F. Pelz, A. L. A. Raabe, "Portugol Studio: uma IDE para Iniciantes em Programação", *22º Workshop sobre Educação em Informática*, Brasília, 2014, p. 535-545.