Geração de Back-End LLVM para o Processador ρ-vex

Aluno(a): Richard Steffano Martins da Silva

Orientador: Prof. Dr. Ricardo Ribeiro dos Santos

Universidade Federal de Mato Grosso do Sul

LSCAD - Laboratório de Sistemas Computacionais de Alto Desempenho

16 de fevereiro de 2012

1 Antecedentes e Justificativa

Um dos aspectos fundamentais para que programas possam utilizar os recursos disponíveis em computadores de alto desempenho diz respeito à geração de código eficiente desses programas. Gerar código eficiente é um desafio confrontado a décadas por parte dos pesquisadores das áreas de compiladores e arquiteturas de processadores. Ademais, um outro desafio para os projetistas de processadores é prover ferramentas que possam gerar código para esses processadores. Nesse sentido, duas das etapas mais importantes no processo de geração de código de programas são o escalonamento de instruções e a alocação de registradores.

Diante do exposto, este projeto em nível de graduação volta-se para o projeto, implementação e avaliação de desempenho de infraestrutura de back-end junto à plataforma de compilação LLVM. Em particular, o foco do projeto está na implementação de etapas do back-end (geração de código), usando o compilador LLVM, para o processador reconfigurável ρ-vex. Antes da implementação efetiva desse back-end para o ρ-vex, pretende-se compreender detalhadamente os passos para geração de novos back-ends sobre LLVM e desenvolver um exemplo completo de back-end baseando-se no processador MIPS que utilizará também um novo algoritmo de escalonamento de instruções e alocação de registradores. O projeto possui vertente teórica (compiladores, arquitetura de computadores, otimizações de código, análise de algoritmos, etc.) mas ainda com forte ênfase prática (implementação de parsers, integração com o algoritmo de escalonamento e alocação de registradores, avaliação de desempenho, etc.).

2 Objetivos

Projetar, implementar e avaliar o desempenho de um back-end para geração de código sobre o processador ρ-vex utilizando o compilador LLVM.

Os objetivos específicos são:

1. Estudar o back-end da infraestrutura LLVM e os detalhes de implementação envolvidos na geração de novos back-ends.

2. Estudar a arquitetura VEX e a implementação ρ-vex.

3. Estudar detalhamente o funcionamento e, em especial, a etapa de geração de grafos base para o algoritmo de escalonamento e alocação de registradores baseado em isomorfismo de subgrafos.

4. Projetar e implementar o back-end para o processador ρ-vex junto ao LLVM
5. Implementar o procedimento de geração do grafo base sobre o processador ρ-vex junto ao LLVM
6. Avaliar o desempenho da implementação do back-end
7. Documentar os resultados obtidos através da escrita de artigos e relatórios

3 Metodologia

A metodologia para desenvolvimento deste plano de trabalho toma como base os objetivos estabelecidos na Seção 2.

1. Estudo do LLVM, arquitetura VEX e processador ρ-vex e o procedimento para geração do grafo base.
2. Implementação do back-end e procedimento para geração do grafo base baseado no processador ρ-vex.
3. Testes de funcionalidade e validação.
4. Avaliação de desempenho com benchmarks.
5. Escrita da monografia e artigos resultantes do trabalho.

4 Cronograma

A Tabela 1 detalha os prazos para execução de cada uma das atividades apresentada na Seção Metodologia.

Atividade	2012											
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1	x	x	x									
2			x	x	x	x						
3					x	x	x					
4							x	x	x			
5						x	x	x	x	x	x	

Tabela 1: Cronograma para execução do plano de trabalho

5 Resultados Esperados

- Integração do procedimento para geração do grafo base junto aos demais procedimentos do algoritmo para escalonamento e alocação de registradores no LLVM.
- Implementação do back-end para geração de código sobre o processador ρ-vex.
- Caracterização das características e desempenho do ρ-vex sobre a infraestrutura LLVM.
- Escrita de artigo com resultados obtidos durante o projeto.

As referências bibliográficas apresentadas a seguir, embora não citadas no texto, servem como base para o início das leituras necessárias para o entendimento do projeto.

Referências

[1] R. Sethi A. V. Aho, M. S. Lam and J. D. Ullman. *Compilers - Principles, Techniques, & Tools*. Addison Wesley, 2 edition, 2007.

[2] Andrew W. Appel. *Modern Compiler Implementation in Java*. Cambridge University Press, 1998.

[3] Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

[4] Rajeev Motwani, Krishna V. Palem, Vivek Sarkar, and Salem Reyen. Combining register allocation and instruction scheduling. Technical report, Stanford, CA, USA, 1995.

[5] R. Santos, R. Azevedo, and G. Araujo. Instruction scheduling based on subgraph isomorphism for a high performance computer processor. *jucs*, 14(21):3465–3480, 2008. http://www.jucs.org/jucs$_1$4$_2$1/$instruction_s chedulin g_b ase d_o n$.

[6] Ricardo Santos, Kariston Avila, Marco Naka, Rodolfo Azevedo, and Amaury Junior. An instruction scheduling and register allocation algorithm for constrained processor architectures. 2010.

[7] L. Santos Silva. Algoritmos para escalonamento de instrucoes e alocacao de registradores na infraestrutura llvm. Technical report, Faculdade de Computacao - Universidade Federal de Mato Grosso do Sul, Maio 2011. Qualificacao de Mestrado.

[8] The LLVM Team. Documentation for the llvm system. [on-line], May 2010. http://llvm.org/releases/2.8/docs/index.html.