



PUC

MARIZA ANDRADE DA SILVA BIGONHA

OTIMIZAÇÃO DE CÓDIGO EM MÁQUINAS SUPERESCALARES

TESE DE DOUTORADO
VOLUME I

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 — CEP 22453

RIO DE JANEIRO — BRASIL

Mariza Andrade da Silva Bigonha

Otimização de Código em Máquinas Superescalares

Tese apresentada ao Departamento de Informática da PUC/RJ como parte dos requisitos para obtenção do título de Doutor em Informática: Ciência da Computação.

Orientador: José Lucas Mourão Rangel Netto.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 29 de abril de 1994.

Ao Roberto, Patrícia e Carolina.
Aos meus pais.
À tia Maria.
Ao tio Álvaro (em memória).

Agradecimentos

Agradeço ao Professor José Lucas Mourão Rangel Netto a orientação acadêmica, a amizade e o entusiasmo com que acolheu a idéia inicial deste trabalho, e por todas as vezes que o fiz madrugar na PUC para nossas reuniões.

Agradeço ao professor Daniel Schwabe a oportunidade a mim oferecida e a orientação inicial deste trabalho.

Agradeço aos membros da banca, professor Wilson de Pádua Paula Filho, professor Sérgio E. Rodrigues de Carvalho, professor Michael Anthony Stanton, Dr. Raul C. Baptista Martins, Dr. Marco Antonio Casanova e professor Júlio César S. P. Leite que prontamente aceitaram meu convite.

Agradeço a todos que me apoiaram de alguma forma na execução deste trabalho; especialmente ao José Jesus P. Alcázar o trabalho de resolver as questões burocráticas junto à PUC/RJ; ao Lucas M. do Amaral que me auxiliou durante a depuração de uma parte do sistema Marion; à Lucília C. de Figueiredo a leitura de parte desta tese e as sugestões apresentadas, ao professor Márcio Luiz B. de Carvalho a assessoria em $\text{\LaTeX}$, à Ruth M. F. de Souza, Fátima, Delma e Rosane T. L. Carvalho, do Departamento de Informática da PUC/RJ, pelo apoio administrativo.

Agradeço ao Departamento de Ciência da Computação da UFMG os recursos oferecidos e à CAPES, CNPq e FAPEMIG o apoio financeiro.

Agradeço à Regina Helena B. Cabral a amizade e o companheirismo de todas as horas.

Agradeço aos meus pais a confiança depositada em mim.

Agradeço às minhas filhas, Carolina e Patrícia, serem compreensivas, carinhosas e me permitirem estudar.

E acima de tudo, agradeço ao Roberto, que além de marido, amigo e companheiro, através de seu amor, apoiou-me pacientemente em todos os momentos da concepção desta tese. Sua competência, seriedade, sistemática de trabalho e sobretudo seu entusiasmo, servem-me de exemplo.

RESUMO

Arquiteturas mais modernas de computadores motivaram pesquisas por técnicas mais eficazes de implementação de compiladores capazes de gerar código de alta qualidade. As arquiteturas superescalares têm conjuntos reduzidos de instruções, cuja combinação para execução eficiente deve ser feita pelos "back-ends" dos compiladores usualmente de maior complexidade que seus correspondentes para arquiteturas CISC. Os objetivos desta tese são: (1) o estudo dos problemas relacionados com a geração de código para arquiteturas mais complexas, tais como as superescalares; (2) o estudo das características de sistemas geradores de código existentes, e das características de sistemas adequados para estas arquiteturas; (3) o estudo das características de linguagens formais de descrição de arquiteturas, apropriadas para uso com geradores de código. Entre os problemas estudados estão a interdependência da alocação de registradores e do escalonamento de instruções, e o desenvolvimento de estruturas de dados mais sofisticadas, capazes de permitir o uso de algoritmos propostos na literatura. O trabalho apresenta então o projeto de um sistema gerador de código adequados para estas arquiteturas e o projeto e a validação semântica de uma linguagem de descrição de arquiteturas apropriada para uso com o gerador de código projetado.

ABSTRACT

Modern computer architectures led to research looking for better techniques for effective implementation of compilers which must be able to produce high-quality code. In the special case of superscalar architectures we have reduced instruction sets, and instructions must be combined to warrant efficient execution by the compilers, which have more complex back-ends than their CISC counterparts. Objectives of this thesis are: (1) the study of the problems of the code generation for more complex architectures, such as the superscalar ones; (2) the study of the characteristics of existing code generator generators, specially those intended for these architectures; (3) the study of the characteristics of formal languages for the description of architectures, meant for use with code generator generators. Among these problems are the interdependence of register allocation and instruction scheduling, and the design of more complex data structures, allowing the use of well-established algorithms reported in the literature. We also present the project of a generator systems which produces code generator well-suited for the afore mentioned architectures, and the project and the semantic validation of an architecture description language adequate for use with the generator system.

Contents

I	Otimização de Código	1
1	O Problema da Otimização de Código	2
1.1	Introdução	2
1.2	Processo de Geração e Otimização de Código	5
1.2.1	Metodologia de Construção de Compiladores	5
1.2.2	Escalonamento de Instruções Redirecionável	8
1.3	Caracterização do Problema	10
1.3.1	Histórico do Trabalho	10
1.3.2	Levantamento dos Problemas	11
1.4	Trabalho Proposto	13
1.4.1	Principais Contribuições	14
1.4.2	Organização da Tese	14
2	Geração de Código para Arquiteturas Superescalares	17

2.1	Introdução	17
2.2	Processador Motorola MC88000	21
2.3	Processador IBM RISC/6000	25
2.4	Processador Intel i860	29
2.5	Processador MIPS R2000	33
2.6	Processador SPARC	35
2.6.1	Janela de Registradores	41
3	Linguagens para Descrição de Arquitetura de Computadores	49
3.1	Introdução	49
3.2	Revisão da Literatura	50
3.2.1	Primeira Família de Geradores de Código	50
3.2.2	Segunda Família de Geradores de Código	53
3.2.3	Terceira Família de Geradores de Código	53
3.2.4	Quarta Família de Geradores de Código	57
3.2.5	Outros Sistemas	58
3.3	Uma Linguagem para Descrição de Arquiteturas	59
3.3.1	Conjunto de instruções do processador	60
3.3.2	Descrição de Máquina	61
3.3.3	Exemplo de uma descrição de máquina	62
3.3.4	Comparação com outros trabalhos	65

3.4	Avaliação	73
3.5	Conclusão	76
4	Escalonamento de Instruções e Alocação de Registradores	79
4.1	Introdução	79
4.2	Estrutura de Dados	79
4.3	Algoritmos de Escalonamento	81
4.4	Alocação de Registradores	84
4.4.1	Método de Chaitin	85
4.5	Conclusão	86
5	O Sistema Marion	88
5.1	Introdução	88
5.2	Maril	89
5.2.1	Declarações	90
5.2.2	Definição da Máquina Virtual	91
5.2.3	Instruções	92
5.2.4	Mapeamento com a Linguagem Intermediária	94
5.3	Gerador de Geradores de Código	96
5.4	Gerador de Código	97
5.4.1	Seletor de Código	98

5.4.2	Alocação de Registradores em Marion	98
5.5	Escalonamento de Instruções em Marion	99
5.5.1	Classes e Relógios	101
5.5.2	Escalonamento Temporal	104
5.5.3	Escalonamento Temporal com Encadeamento	109
5.5.4	Escalonamento para o Intel i860	112
5.6	Conclusão	113
6	Integração da Alocação de Registradores e o Escalonamento de Instruções	120
6.1	Introdução	120
6.2	O Problema	121
6.3	Estratégias de Geração e Otimização de Código	124
6.3.1	Escalonamento <i>POSTPASS</i>	126
6.3.2	Escalonamento IPS	128
6.3.3	Geração de Código RASE	130
6.4	Análise Crítica das Estratégias POSTPASS, IPS e RASE . . .	137
6.4.1	Estratégia de Geração de Código	138
6.4.2	Tamanho do Conjunto de Registradores	138
6.4.3	Organização de Registradores	139
6.4.4	Latência das Operações	140

6.4.5	Latências das Instruções <i>load</i>	140
6.5	Conclusão	141
7	Generalização do Algoritmo de Alocação de Registradores e Escalonamento de Instruções	148
7.1	Introdução	148
7.2	Suposições Básicas	150
7.3	Alocação de Registradores para Máquinas Superescalares . .	151
7.3.1	Formalização do Modelo Proposto	153
7.3.2	Gerenciamento do Derramamento de Código	159
7.4	Pares de Registradores	161
7.5	Algoritmo para Alocação de Registradores	163
7.6	Conclusão	165
8	A Linguagem de Descrição de Arquitetura	168
8.1	Considerações Gerais	168
8.2	Características da Linguagem LDA	169
8.3	Notação	170
8.4	Símbolos Básicos de LDA	171
8.5	Sintaxe Concreta de LDA	173
8.6	Estrutura da LDA	176
8.7	Seção de Definição das Declarações	176

8.8	Seção de Definição da Máquina Alvo	181
8.9	Seção de Definição das Instruções	186
8.9.1	Instruções Comuns	186
8.9.2	Instruções Especiais	192
8.10	Escopo de Aplicação de LDA	197
8.10.1	Elementos Descritíveis em LDA	197
8.10.2	Elementos Não-descritíveis em LDA	200
9	O Sistema GGCO	202
9.1	Introdução	202
9.2	Compilador mcc	203
9.3	Estrutura do Módulo Gerador de Código	204
9.4	Interface entre o <i>Front-end</i> e o Gerador de Código	205
9.4.1	Linguagem Intermediária	206
9.5	Avaliação de GGCO	210
9.5.1	Modelos de Pilha de GGCO	210
9.5.2	Modelo de Pilha da Arquitetura SPARC	212
9.5.3	Pilha nos Registradores	219
9.6	Conclusão	220
10	Conclusões	223

10.1 Principais Contribuições	225
10.2 Propostas de Novas Direções	226
11 Bibliografia	229
A Definição do Processador SPARC	239
B Descrição de Máquina	270
B.1 MC68000	270
B.1.1 Endereçamento de Memória	271
B.1.2 Instruções	274
B.2 VAX-11	275
B.2.1 Instruction-set Processor	275
B.2.2 Registradores	276
B.2.3 Endereçamento de Memória	277
B.2.4 Modos de endereçamento	277
B.2.5 Instruções	279
B.3 PDP-11	280
B.3.1 Instruction-set Processor	280
B.3.2 Endereçamento de Memória	281
B.3.3 Modos de endereçamento	281
B.3.4 Instruções	283



PUC

MARIZA ANDRADE DA SILVA BIGONHA

OTIMIZAÇÃO DE CÓDIGO EM MÁQUINAS SUPERESCALARES

TESE DE DOUTORADO

VOLUME II

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 — CEP 22453

RIO DE JANEIRO — BRASIL

C	Arquivo MD.h	284
II	Formalização da Linguagem de Arquitetura	289
D	Estrutura da Definição Formal	290
D.1	Filosofia da Definição	290
D.2	Domínios Sintáticos de LDA	292
D.3	Domínios Semânticos	299
D.4	Funções Semânticas	307
D.5	Resposta Final	311
D.5.1	Tabela de Nomes	312
D.5.2	Tabela de Declarações	312
D.5.3	Tabela com as Convenções de Chamadas de Procedi- mento	314
D.5.4	Tabela de Recursos Utilizados pelas Instruções	315
D.5.5	Tabela de Associação dos Elementos a Relógios	318
D.5.6	Tabela de Classes	319
D.5.7	Tabela de Produções	320
D.5.8	Tabela de Instruções Glue	328
D.5.9	Tabela de Latências Auxiliares	328
D.5.10	Listas e Rotinas	330

List of Figures

2.1	Arquitetura do Sistema Motorola MC88100	23
2.2	Arquitetura do Sistema/ RS6000	26
2.3	Arquitetura do Sistema Intel i860	32
2.4	Exemplo de uma Operação de Adição no i860	33
2.5	Arquitetura SPARC	36
2.6	Pilha Circular de Janela de Registradores	46
2.7	Janela de Registradores	47
2.8	Sobreposição Parcial de Janelas de Registradores	47
2.9	Exemplo de três Procedimentos utilizando Janela de Reg- istradores	48
4.1	Exemplo de um Grafo de Dependências	82
5.1	Declarações em Maril	91
5.2	Máquina Virtual	92
5.3	Descrição de uma Instrução em Maril	93

5.4	Instruções em Maril	95
5.5	Exemplo de uma função de escape	96
5.6	FPU do i860 Modelado como uma Instrução Longa	102
5.7	Diretivas da Instrução de Multiplicação do i860	103
5.8	Parte do DAG de Código Modificado para Evitar Travamento no Escalonamento	107
5.9	Algoritmo 1: Protege Seqüências Temporais	115
5.10	Entradas Alternativas em Seqüências Temporais	116
5.11	Instruções para <i>pipeline</i> de Adição do i860	117
5.12	Exemplo de uma Árvore Temporal	118
5.13	Parte das Declarações de Elementos e Classes do i860	119
6.1	Alocação Registradores <i>antes</i> do Escalonamento Instruções	144
6.2	Dois Escalonamentos para <i>loads</i> e <i>adds</i> Independentes	145
6.3	Uso da Política de Coloração <i>Basic-block</i>	145
6.4	Arquitetura RASE	145
6.5	Grafo de Interferência na Abordagem de Chaitin	146
6.6	Grafo de Interferência na Abordagem de Bradlee	146
6.7	Latências das operações <i>longerop</i> e <i>shorterop</i>	147
7.1	Grafo de escalonamento.	152

7.2	Exemplo 2: (a) Código para um Trecho de Programa; (b) Grafo de Escalonamento para (a)	154
7.3	(a) Vértices no Conjunto E_t , (b) Grafo de Dependências Falsas de (a) e (c) Grafo de Interferência	155
7.4	(a) Grafo de Interferência Paralelizável do Exemplo 2 e (b) Possível Alocação de Registradores.	157
7.5	(a) Grafo de Interferência para o Exemplo 1.	158
8.1	Significado da Posição dos Caracteres em um Mnemônico. . .	181
9.1	Estrutura do Gerador de Gerador de Código Otimizado-GGCO203	
9.2	Estrutura do Compilador mcc	204
9.3	Estrutura do Gerador de Código.	205
9.4	Vértices (nodes) definindo os Operadores.	209
9.5	Pilha e Registro de Ativação.	211
9.6	Pilha do Usuário	213
9.7	Pilha de Execução	218
D.1	Mapeamento de lda em mdc.	291
D.2	Estrutura do Domínio Env	299
D.3	Estrutura da Lista Ntlist-s	305
D.4	Exemplo da Tabela Mnametab[]	312
D.5	Exemplo da Tabela Decl[s[]]	313

D.6 Exemplo da Tabela <code>PROCCONV Std-call</code>	315
D.7 Diretivas para as Instruções <code>div</code>	316
D.8 Exemplo da Tabela <code>RVECT Ruse[]</code> para as Instruções <code>div</code> , <code>ld</code> e <code>ldsc</code>	317
D.9 Exemplo da Tabela <code>ELENT Element[]</code>	318
D.10 Diretiva de <code>%class</code> usada na formação da Tabela <code>Class[]</code> . . .	319
D.11 Exemplo da Tabela <code>Class[]</code>	320
D.12 Exemplo da Especificação de uma Instrução <code>st</code>	321
D.13 Árvore da Instrução: <code>%instr st r, r, #uconst16 {m[\$2+\$3]=\$1}</code>	322
D.14 Exemplo de um trecho da Tabela <code>Prods[]</code>	323
D.15 Exemplo da Tabela <code>Prodindex[]</code>	324
D.16 Exemplo da Tabela <code>Ntindex[]</code>	325
D.17 Tabelas para o Reconhecimento de Padrões	326
D.18 Exemplo de Diretivas de Instruções <code>*func</code>	327
D.19 Declaração de Funções para as Diretivas <code>*sfunc</code>	327
D.20 Exemplo da Tabela <code>SFUNC Sfunc[3]</code>	327
D.21 Exemplo da Tabela <code>GLUE Glue[28]</code>	328
D.22 Exemplo das Funções geradas para as Diretivas <code>%aux</code>	329
D.23 Exemplo da Tabela <code>Aux-latency[7]</code>	329
D.24 Árvore Padrão para: <code>%instr add r, r, #uconst16</code>	334
D.25 Árvore Padrão para: <code>%glue r, r</code>	336

List of Tables

2.1	Modos de Endereçamento das Arquiteturas RISCs	18
2.2	Características Básicas das Arquiteturas RISCs	19
2.3	Latências das Operações no MC88000	24
2.4	Latências das Operações no i860	33
2.5	Latências das Operações no MIPS R2000	34
2.6	Latências das Operações na SPARC	41
2.7	Nomes Alternativos para os Registradores da Unidade de In- teiros	42
5.1	Declarações para o Relógio de Multiplicação	104
5.2	Significado da chave do Mnemônico “X”	112
9.1	Tipos dos Sufixos.	207

Part I

Otimização de Código