

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO



Guilherme Campos Hazan

Uma Especificação de Máquina de Registradores para Java

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Roberto Ierusalimsky



Guilherme Campos Hazan

Uma Especificação de Máquina de Registradores para Java

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Roberto Ierusalimschy

Orien-
tador

Departamento de Informática – PUC-
Rio

Profa. Noemi de La Rocque Rodriguez

Departamento de Informática – PUC-
Rio

Prof. Renato Fontoura de Gusmão Cerqueira

Departamento de Informática – PUC-
Rio

Prof. José Eugenio Leal

Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de janeiro, 09 de abril de 2007

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Guilherme Campos Hazan

Bacharel em Ciência da Computação pela Universidade Federal do Rio de Janeiro (UFRJ). Começou a programar em 1984. Iniciou em Java em 1998, trabalhando para a Quality Software, onde criou applets de geração de diversos tipos de gráficos, além de uma applet de CAD (2o lugar no Concurso Anual de Comércio Eletrônico da Siemens de 2002). Palestrante em diversas conferências: Comdex Sucesu 2002, One Day Java 2002, Just Java 2003, Java In Rio 2003, Congresso Catarinense de Software Livre 2003, dentre outros. Guilherme é o criador e da máquina virtual SuperWaba. Professor de Java para PDAs na pós-graduação da CESUMAR.

Ficha Catalográfica

Hazan, Guilherme Campos

Uma especificação de máquina de registradores para Java / Guilherme Campos Hazan; orientador: Roberto Ierusalimschy. – 2007.

72 f. ; 30 cm

Dissertação (Mestrado em Informática)–Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Inclui bibliografia

1. Informática – Teses. 2. Java. 3. Bytecode. 4. Compilador. 5. Máquina virtual. 6. Registrador. 7. Dispositivos móveis. I. Ierusalimschy, Roberto. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

Incluí referências bibliográficas.

Java, bytecode, compilador, máquina virtual, registrador, pilha, desempenho, otimização, pda, dispositivos móveis

Gostaria de dedicar essa dissertação à Verinha, minha mãe, que faleceu em Fevereiro de 2005. Ela, como Diretora da Escola de Musica da Universidade Federal de Minas Gerais, e Diretora Artística da Fundação Clovis Salgado, foi um exemplo para mim, pelo seu profissionalismo reconhecido pelos colegas de trabalho e capacidade de tornar um sucesso os trabalhos que fez pela grande dedicação e, principalmente, pela paixão com que tratava a sua profissão.

Agradecimentos

Gostaria de agradecer ao meu orientador, Roberto, pelo seu desprendimento em me orientar em um trabalho diferente do que costuma realizar, na plataforma Lua. Isso causava surpresa em todas as pessoas com quem eu conversava sobre o assunto da dissertação e sobre quem era o orientador. Elas indagavam: "... mas não é sobre Lua?".

Gostaria de agradecer também ao Prof. Arndt von Staa, cuja matéria sobre Testes e Métricas de Software foi de fundamental importância para auxiliar a correta implementação da máquina virtual.

Resumo

Hazan, Guilherme Campos; Ierusalimschy, Roberto. **Uma Especificação de Máquina de Registradores para Java**. PUC-Rio, 2007. 72p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A linguagem Java foi definida tendo como foco a portabilidade. O código gerado pela compilação é interpretado por uma máquina virtual, e não diretamente pelo processador destino, como um programa em C. Este código intermediário, também conhecido como bytecode, é a chave da portabilidade de Java. Os Bytecodes Java usam uma pilha para manipular os operandos das instruções. O uso de pilha tem suas vantagens e desvantagens. Dentre as vantagens, podemos citar a simplicidade da implementação do compilador e da máquina virtual. A principal desvantagem é a redução na velocidade de execução dos programas, devido à necessidade de se mover os operandos para a pilha e retirar dela o resultado, gerando um aumento no número de instruções que devem ser processadas. Diversos estudos indicam que máquinas virtuais baseadas em registradores podem ser mais rápidas que as baseadas em pilha. Decidimos criar uma nova especificação de bytecodes, específicos para máquinas virtuais baseadas em registradores. Esperamos com isso obter um aumento no desempenho das aplicações.

Palavras-chave

Java; bytecode; compilador; máquina virtual; registrador; pilha; desempenho; otimização; pda; dispositivos móveis

Abstract

Hazan, Guilherme Campos; Ierusalimschy, Roberto. **A Specification for a Java Register-Based Machine**. PUC-Rio, 2007. 72p. MSc Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The Java language was created with a focus on portability. The code generated by the compiler is interpreted by a virtual machine, and not directly by the target processor, like programs written in C. This intermediate code, also known as bytecode, is the key to Java's portability. The Java Bytecodes use a stack to manipulate the instruction operands. The use of stack has its pros and cons. Among the advantages, we can cite the simplicity of implementation of the compiler and virtual machine. On the other hand, there is a speed reduction in the program's execution, due to the need to move the operands to and from the stack, and retrieve results from it, increasing the number of instructions that are processed. Much study has been done that indicating that register-based virtual machines can be faster than the ones based on stacks. Based on this, we decided to create a new bytecode specification, proper for a virtual machine based on registers. By doing this, we hope to obtain an increase in an application's performance.

Keywords

Java; bytecode; compiler; virtual machine; register; stack; performance; optimization; pda; mobile devices

Sumário

1 Introdução	13
2 Referencial Teórico	15
2.1. Tamanho do código armazenado	17
2.2. Desempenho dos programas	18
2.3. Máquinas virtuais baseadas em registradores	20
2.4. Conversão de pilha para registradores	21
3 Nova Arquitetura de Bytecodes	23
3.1. Separação dos registradores em grupos	23
3.2. Quantidade de registradores	24
3.3. Operações com campos	25
3.4. Ortogonalidade das operações	25
3.5. Otimização da tabela de constantes	27
3.6. Arquitetura de palavra das instruções	27
3.7. Formato das instruções	28
3.8. Escolha das instruções	28
3.9. Listagem das instruções	37
4 Avaliação do Código Gerado	46
4.1. Descrição dos testes	46
4.2. Desempenho obtido nas plataformas	50
4.3. Tamanho do código gerado	54
5 Conclusão	56
6 Referências Bibliográficas	59
7 Apêndice	61
7.1. Lista dos bytecodes Java	62
7.2. Mapeamento entre os bytecodes Java e as novas instruções	63

Lista de tabelas

Tabela 1: Categorias dos bytecodes Java	16
Tabela 2: Tipos Java	16
Tabela 3: Percentagem da frequência dinâmica para grupos de Bytecodes	19
Tabela 4: Bytecodes necessários para somar 1 a um campo de instância	19
Tabela 5: Percentagem do número de parâmetros usados em métodos	24
Tabela 6: Percentagem do número de locais usados em métodos	24
Tabela 7: Programas usados durante o teste e resumo dos resultados	32
Tabela 8: Instruções aritméticas e lógicas dos programas Java	33
Tabela 9: Instruções aritméticas e lógicas do J2SE 5.0	33
Tabela 10: Operandos das instruções aritméticas e lógicas para Java	34
Tabela 11: Operandos das instruções aritméticas e lógicas para J2SE 5.0	34
Tabela 12: Instruções de desvio condicional para Java	35
Tabela 13: Instruções de desvio condicional para J2SE 5.0	36
Tabela 14: Operandos das instruções de desvio condicional para Java	36
Tabela 15: Operandos das instruções de desvio condicional para J2SE 5.0	37
Tabela 16: Número de locais por método para programas Java	37
Tabela 17: Número de locais por método para J2SE 5.0	37
Tabela 18: Operandos usados nas instruções	40
Tabela 19: Instruções de movimentação	41
Tabela 20: Instruções aritméticas e lógicas	42
Tabela 21: Instruções de desvio condicional	43
Tabela 22: Instruções de chamada de método	44
Tabela 23: Instruções de conversão entre tipos de dados	44
Tabela 24: Instruções para retorno de método	44
Tabela 25: Instruções não categorizadas	45
Tabela 26: Partes que diferem entre as plataformas	48
Tabela 27: Partes comuns às plataformas	48
Tabela 28: Código fonte dos testes realizados no Pentium 4-M.	49
Tabela 29: Tempos em milisegundos dos testes realizados no Pentium 4-M	53
Tabela 30: Número de instruções dentro do laço	53
Tabela 31: Tempos em milisegundos dos testes realizados no Dell Axim X3	53
Tabela 32: Bytecodes Java	62

Lista de figuras

Figura 1: Composição de um arquivo class	18
Figura 2: Tabela de constantes	18
Figura 3: Comparação dos tipos primitivos em relação ao <code>int</code> .	26
Figura 4: Código usado para medir o tempo de execução	26
Figura 5: Gráfico comparativo com os tempos obtidos no Pentium 4-M	52
Figura 6: Gráfico comparativo com os tempos obtidos no Dell Axim X3	52
Figura 7: Tamanho do código gerado pela compilação do teste	55
Figura 8: Composição do arquivo gerado pela compilação do teste	55
Figura 9: Tamanho do código gerado pela compilação dos protótipos	55