TECHNISCHE UNIVERSITÄT DRESDEN
Martin Stiller, Fakultät Informatik, Institut für Technische Informatik

# LLVA: Eine virtuelle Befehlssatzarchitektur

Hauptseminar Technische Informatik, 02.05.2012

# Gliederung

1 Virtual Instruction Set Computers (VISC)

2 Virtual Instruction Set Architecture

   i. Aufbau und Eigenschaften

   ii. Anforderungen und Ziele

3 Low Level Virtual ISA (LLVA)

4 Übersetzungsstrategie

5 Einsatzgebiete

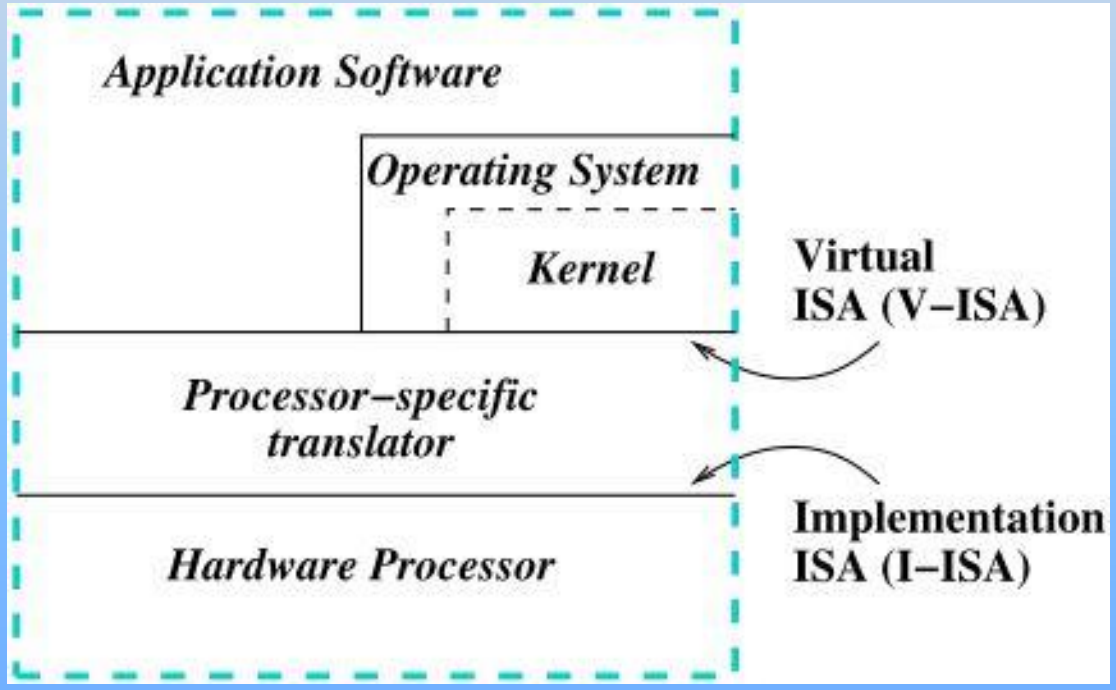
6 Ausblick

# 1 Virtual Instruction Set Computers (VISC)

- Programm- und Betriebssystemebene:
  - Virtual ISA (V-ISA)

- Hardware-Implementierung (Codesigned Virtual Machine):
  - Implementation ISA (I-ISA) eines Prozessors
  - Software Translation Layer (hardwarespezifisch)

# 1 Virtual Instruction Set Computers (VISC)




Entnommen aus [1]

# 1 Virtual Instruction Set Computers (VISC)

- Vorteile :
  - V-ISA beinhaltet maschinenunabhängige Programminformationen
  - I-ISA und Translator stellen hardwarespezifische Implementierungen zur Verfügung
  - angepasste Mikroarchitektur zur Unterstützung und Kontrolle der Übersetzung
- Nachteile:
  - Geschwindigkeitseinbußen durch Übersetzung

# 2 Virtual Instruction Set Architecture (VISA)

- Aufbau:
  - RISC-ähnliche Befehlsstruktur
  - sprachunabhängige Typisierung
  - Steuer- und Datenflussinformationen (Static Single Assignment)

# 2 Virtual Instruction Set Architecture (VISA)

- Eigenschaften:
  - Informationen für Compiler-Analyse und Transformation
  - verringertes Abstraktionsniveau für gute Abbildbarkeit auf Hardware
  - Unterstützung von Betriebssystemanweisungen
  - Definition von Ausnahmebehandlungen
  - Umgang mit selbstveränderlichem Code
  - Optimierungsmöglichkeiten zur Lauf- / Installationszeit und Offline-Translation

# 2 Virtual Instruction Set Architecture (VISA)

- Anforderungen:
  - Spezifikation einer Schnittstelle für V-ISA (V-ABI)
  - einfache Austauschbarkeit von I-ISA und Translator
  - Unterstützung verschiedener Betriebssysteme und Systemarchitekturen

# 2 Virtual Instruction Set Architecture (VISA)

- Ziele:
  - Direktabbildung einfacher low-level Operationen auf wenige Maschinenbefehle
  - Vereinfachung der Programmanalyse
  - Kompatibilität innerhalb einer Prozessorfamilie
  - Nutzung der high-level Informationen für Analysen (Optimierung, Scheduling und Registerallokation)
  - Sprachunabhängigkeit
  - Unterstützung von Betriebssystemen mit V-ABI

# 3 Low Level Virtual ISA (LLVA)

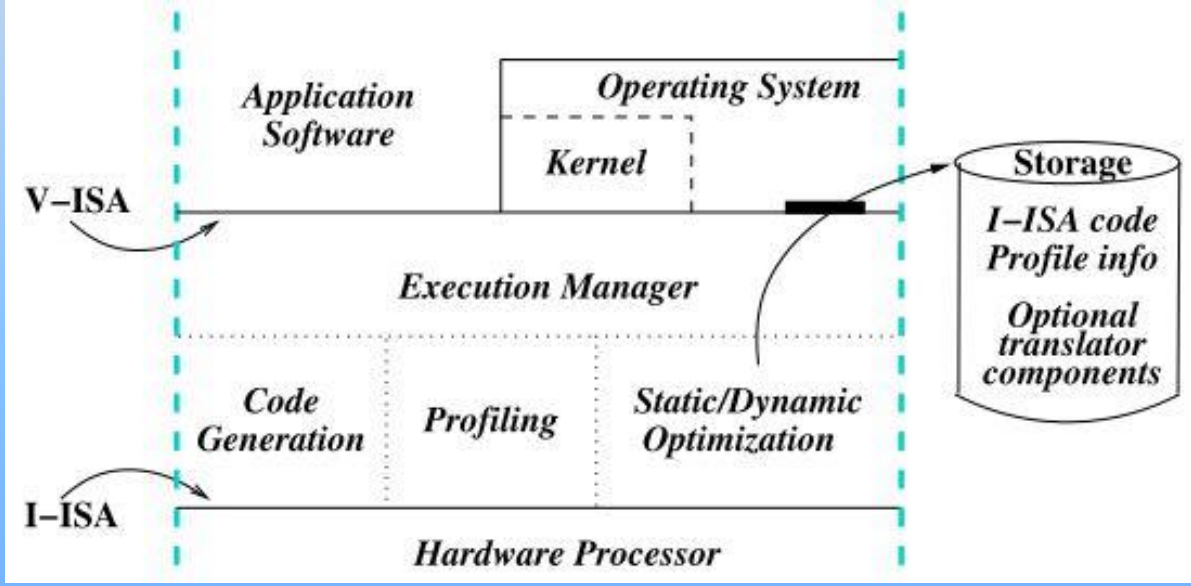
- typisierte Register
- 3-Adress-Maschine (32-bit-Format)
- Speicher: Stack, Heap, globaler Speicher (eindeutig allokierbar)
- Load-/Store-Architektur
- 28 Instruktionen (orthogonal)
- Erzeugung eines Steuerflussgraphen
- Primitive Datentypen (ubyte, uint, float, double, …)
- 4 abgeleitete Datentypen (pointer, array, structure, function)

# 4 Übersetzungsstrategie

- Ziele:
  - Reduzierung der Online-Translation
  - Ausnutzen  der gewonnen Optimierungsmöglichkeiten

- Low Level Execution Environment (LLEE):
  - koordiniert den Übersetzungsprozess
  - ermöglicht Offline-Translation

# 4 Übersetzungsstrategie

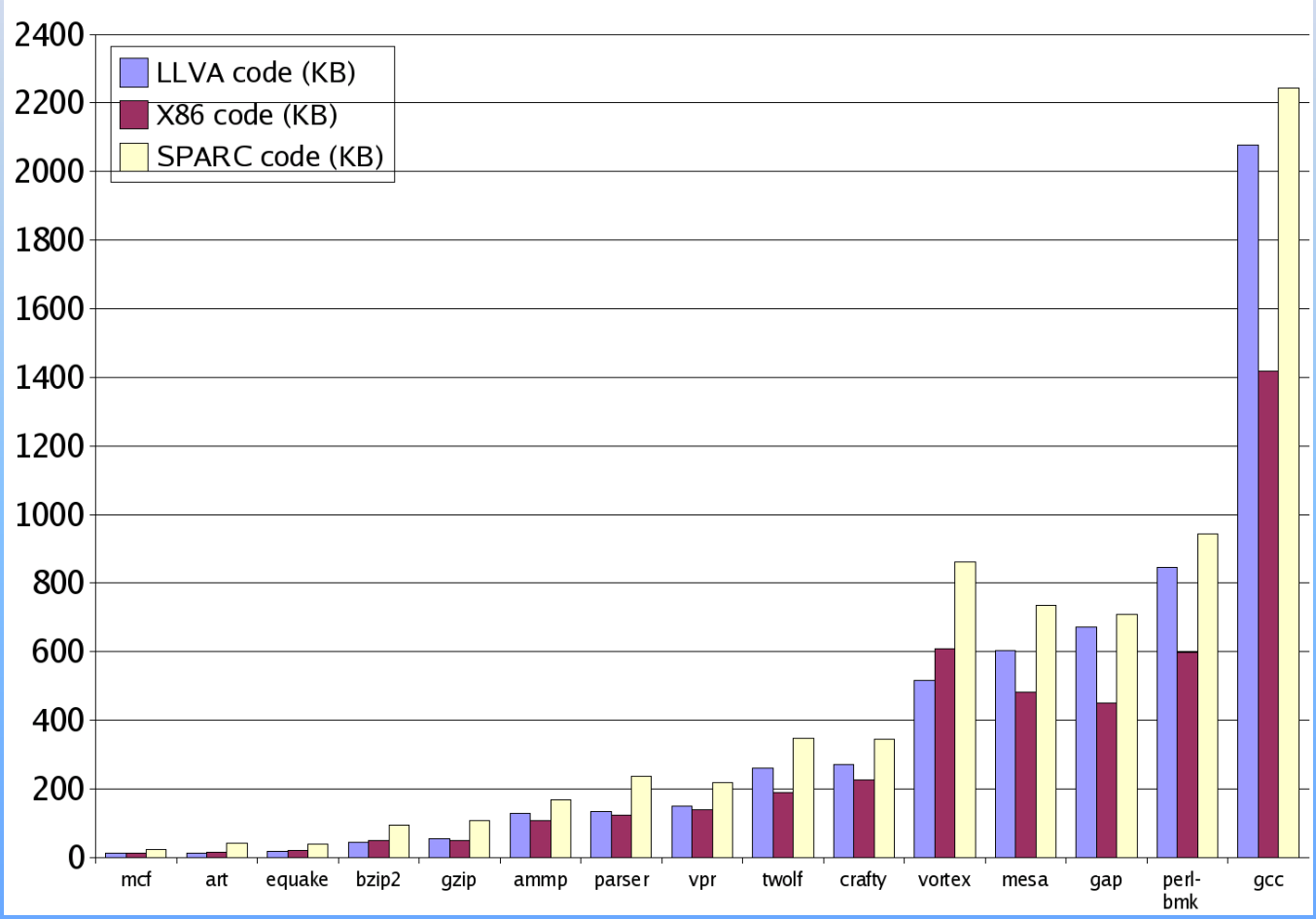



Entnommen aus [1]

# 5 Einsatzgebiete

- Low Level Virtual Machine (LLVM) Compiler Framework:
  - Front-Ends für z.B. C/C++ und Fortran
  - Back-Ends für z.B. x86, x86-64, PowerPC, PowerPC-64, ARM, SPARC

# 5 Einsatzgebiete




Entnommen aus [2]

# 6 Ausblick

- Hardwarebeschleunigung für Translator in Prozessor
- Erweiterung um Parallelisierungsmöglichkeiten
- Einbeziehung bestehender HLVM (z.B. JVM)
- Integration in Betriebssystem

# Quellen


- [1] LLVA: A Low-level Virtual Instruction Set Architecture, Vikram Adve, Chris Lattner, Michael Brukman, Anand Shukla, Brian Gaeke
- [2] LLVA Präsentation zu [1]
- [3] www.llvm.org/features.htm