

Gliederung

Systemprogrammierung

Rechnerorganisation: Schichtenstruktur

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

14. Mai 2014

1 Semantische Lücke

- Fallstudie

2 Mehrebenenmaschinen

- Maschinenhierarchie
- Maschinen und Prozessoren
- Entvirtualisierung

3 Zusammenfassung

Verschiedenheit zwischen Quell- und Zielsprache

Faustregel: $\left\{ \begin{array}{l} \text{Quellsprache} \rightarrow \text{höheres} \\ \text{Zielsprache} \rightarrow \text{niedrigeres} \end{array} \right\} \text{Abstraktionsniveau}$

Semantische Lücke (engl. *semantic gap*, [6])

The difference between the complex operations performed by high-level constructs and the simple ones provided by computer instruction sets. It was in an attempt to try to close this gap that computer architects designed increasingly complex instruction set computers.

- Kluft zwischen gedanklich Gemeintem und sprachlich Geäußertem

Problemraum



(Mathematik)

Lösungsraum



(Informatik)

- „gedanklich gemeint“ ist ein Verfahren aus der linearen Algebra
- „sprachlich geäußert“ auf verschiedenen Ebenen der **Abstraktion**

Verschiedenheit zwischen Quell- und Zielsprache (Forts.)

Ebene der **Modellsprache** „Lineare Algebra“ ~ 1 Produktsummenformel

- welches Problem behandelt wird, ist (nahezu) offensichtlich
- eine semantische Lücke ist eigentlich nicht vorhanden

Ebene der **Programmiersprache** C ~ 5 Komplexschritte

- welches Problem behandelt wird, ist (für Experten) noch erkennbar
- die semantische Lücke ist vergleichsweise klein

Ebene der **Assembliersprache** ASM $\sim 35+n$ Elementarschritte

- welches Problem behandelt wird, ist (eigentlich) nicht erkennbar
- die semantische Lücke ist vergleichsweise sehr groß

Ebene der **Maschinensprache** x86 ~ 97 Bytes Programmtext

- welches Problem behandelt wird, ist überhaupt nicht mehr erkennbar
- die semantische Lücke ist (nahezu) unendlich groß

Gliederung

1 Semantische Lücke

- Fallstudie

2 Mehrebenenmaschinen

- Maschinenhierarchie
- Maschinen und Prozessoren
- Entvirtualisierung

3 Zusammenfassung

Zwischenzusammenfassung

$$C_{i,j} = \sum_k A_{ik} \cdot B_{kj} \xrightarrow{?} 5589E5 \dots 5F5DC3$$

Die Diskrepanz zwischen der vom Menschen skizzierten Lösung des Problems und dem dazu korrespondierenden, von einem Prozessor ausführbaren Maschinenprogramm ist beträchtlich.

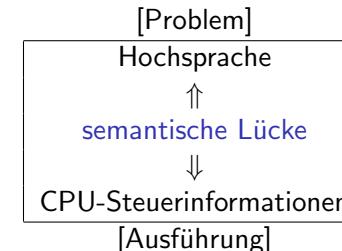
Abstraktion half, sich auf das Wesentliche konzentrieren zu können

- eine **virtuelle Maschine** zur Matrizenmultiplikation entstand
- die schrittweise abgebildet wurde auf die **reale Maschine „x86“**

Umwandlung ins ausführbare Programm meint viele Arbeitsgänge:

① Vorverarbeitung des Quellmoduls	cpp(1)
② Kompilierung der vorverarbeiteten Quelle	gcc(1)
③ Assemblierung des (zwischenzeitlich) erzeugten Aggregats	as(1)
④ Bindung assemblierter Objektmodule zum Lademodul	ld(1)

Aufgabenstellung \mapsto Programmlösung



Breite der semantischen Lücke variiert:

- bei gleich bleibendem Problem mit der Plattform (dem System)
- bei gleich bleibender Plattform mit dem Problem (der Anwendung)

Lückenschluss ist ganzheitlich zu sehen

Semantische Lücke schrittweise schließen

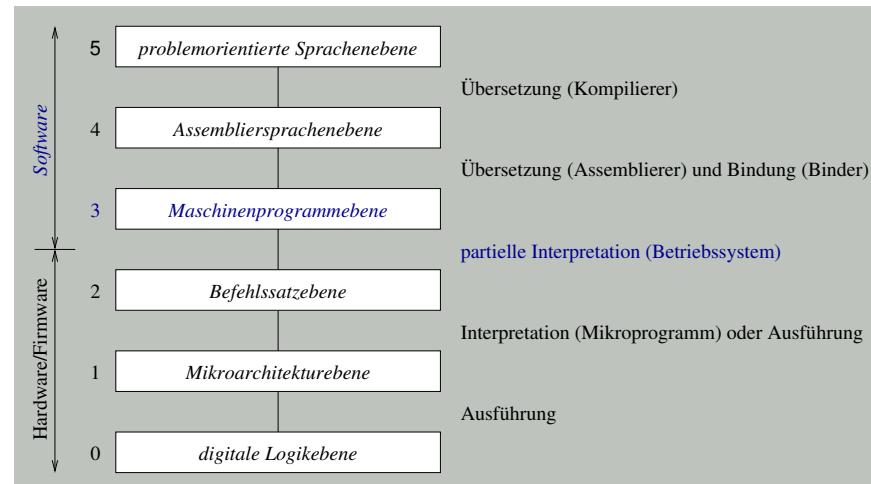
- durch hierachisch angeordnete **virtuelle Maschinen** Programmlösungen auf die reale Maschine abbilden [4]
- Prinzip *divide et impera* („teile und herrsche“): einen „Gegner“ in leichter besiegbare „Untergruppen“ aufspalten

Hierarchie virtueller Maschinen [5, S. 3]

Interpretation und **Übersetzung** (Komplizierung, Assemblierung):

Ebene		
n	virtuelle Maschine M_n mit Maschinensprache S_n	Programme in S_n werden von einem auf einer tieferen Maschine laufenden Interpreter gedeutet oder in Programme tieferer Maschinen übersetzt
:	:	:
2	virtuelle Maschine M_2 mit Maschinensprache S_2	Programme in S_2 werden von einem auf M_1 bzw. M_0 laufenden Interpreter gedeutet oder nach S_1 bzw. S_0 übersetzt
1	virtuelle Maschine M_1 mit Maschinensprache S_1	Programme in S_1 werden von einem auf M_0 laufenden Interpreter gedeutet oder nach S_0 übersetzt
0	reale Maschine M_0 mit Maschinensprache S_0	Programme in S_0 werden direkt von der Hardware ausgeführt

Hardware/Software-Hierarchie von Rechensystemen [4]



Programme sorgen für die Maschinenabbildung

Komplizierer (engl. *compiler*) und Interpretierer (engl. *interpreter*)

Komplizierer lat. (Zusammenträger)

- ein **Softwarereprocessor**, transformiert Programme einer *Quellsprache* in semantisch äquivalente Programme einer *Zielsprache*
 - {Ada, C, C++, Eiffel, Modula, Fortran, Pascal, ...} \mapsto Assembler
 - aber ebenso: C++ \mapsto C \mapsto Assembler

Interpretierer lat. (Ausleger, Erklärer, Deuter)

- ein in Hard-, Firm- oder Software realisierter **Prozessor**, führt Programme einer bestimmten Quellsprache „direkt“ aus
 - z.B. Basic, Perl, C, sh(1)
- ggf. **Vorübersetzung** durch einen Komplizierer, um Programme in eine für die Interpretation günstigere Repräsentation zu bringen
 - z.B. Pascal P-Code, Java Bytecode, x86-Befehle

Softwaremaschine: Komplizierer/Assemblierer (Binder)

problemorientierte Programmiersprachenebene

[AuD/PFP]

- bietet „höhere Programmiersprachen“ zur abstrakten, problemorientierten Formulierung von Programmlösungen
- Programme setzen sich zusammen aus Konstrukten zur Selektion und Iteration, zur Formulierung von Sequenzen, Blockstrukturen, Prozeduren, zur Beschreibung von elementaren und abstrakten Datentypen und (getypten) Operatoren

Assembliersprachenebene (symbolischer Maschinenkode)

[GRA]

- bietet „niedere Programmiersprachen“ zur konkreten, CPU-spezifischen Formulierung von Programmlösungen
- Programme bestehen aus Pseudobefehle, mnemonisch ausgelegte Maschinenbefehle (ISA), symbolisch bezeichnete Operanden (Speicheradressen, Register) und Adressierungsarten

Abbildung durch Übersetzung

Ebene 5 \mapsto Ebene 4 (Kompilierung)

- Ebene 5-Befehle „1:N“ in Ebene 4-Befehle übersetzen
 - ein Hochsprachenbefehl als Sequenz von Assembliersprachenbefehlen
 - eine **semantisch äquivalente Befehlsfolge** generieren
- im Zuge der Transformation ggf. Optimierungsstufen durchlaufen

Ebene 4 \mapsto Ebene 3 (Assemblierung und Binden)

- Ebene 4-Befehle „1:1“ in Ebene 3-Befehle übersetzen
 - ein **Quellmodul** in ein **Objektmodul** umwandeln
 - mit **Bibliotheken** zum Maschinenprogramm zusammenbinden
- symbolischen Maschinenkode („Mnemoniks“) auflösen
 - in binären Maschinenkode umwandeln

Übersetzung und Interpretation durch Prozessoren

Ebene 5 \leadsto Kompilierer

- Interpretation von Konstrukten/Anweisungen einer „Hochsprache“

Ebene 4 \leadsto Assemblierer und Binder

- Interpretation von Anweisungen einer Assembliersprache

Ebene 3 \leadsto Betriebssystem

- Interpretation von Systemaufrufen und sensitiven Ebene 2-Befehlen
 - Ausführung von Ebene 3-Programmen (durch Teilinterpretation)

Ebene 2 \leadsto Zentraleinheit (CPU)

- Interpretation von Instruktionen (an die ALU, FPU, MMU, ...)
 - Ausführung von Ebene 2-Programmen

Abbildung durch Interpretation

Ebene 3 \mapsto Ebene 2 (Teilinterpretation, auch *partielle Interpretation*)

- Ebene 3-Befehle typ- und zustandsabhängig verarbeiten:
 - (a) als Folgen von Ebene 2-Befehlen ausführen
 - Systemaufrufe annehmen und befolgen
 - (synchrone/asynchrone) Programmunterbrechungen behandeln
 - sensitive Ebene 2-Befehle emulieren
 - (b) „1:1“ auf Ebene 2-Befehle abbilden (nach unten „durchreichen“)
 - ein Ebene 3-Befehl aktiviert ggf. ein Ebene 2-Programm

Ebene 2 \mapsto Ebene 1 (Interpretation)

- Ebene 2-Befehle als Folgen von Ebene 1-Befehlen ausführen
 - Abruf- und Ausführungszyklus (engl. *fetch-execute-cycle*) der CPU
- ein Ebene 2-Befehl löst Ebene 1-Steueranweisungen aus

Zeitpunkte der Abbildungsvorgänge

Bezogen auf das jeweils zu interpretierende/übersetzende Programm

vor Laufzeit (Ebene 5 \mapsto Ebene 3) \leadsto statisch

- Vorverarbeitung (engl. *preprocessing*)
- Vorübersetzung (engl. *precompilation*)
- Übersetzung: Kompilierung, Assemblierung
- Binden (engl. *static linking*)

zur Laufzeit (Ebene 5 \mapsto Ebene 1) \leadsto dynamisch

- bedarfsorientierte Übersetzung (engl. *just in time compilation*)
- Binden (engl. *dynamic linking*)
- bindendes Laden (engl. *linking loading, dynamic loading*)
- Teilinterpretation
- Interpretation

Betriebssysteme entvirtualisieren zur Laufzeit

- dynamisches Binden, bindendes Laden, Teilinterpretation

Gliederung

1 Semantische Lücke

- Fallstudie

2 Mehrebenenmaschinen

- Maschinenhierarchie
- Maschinen und Prozessoren
- Entvirtualisierung

3 Zusammenfassung

Resümee

Strukturierte Organisation von Rechensystemen

• semantische Lücke

- „Kluft zwischen gedanklich Gemeintem und sprachlich Geäußertem“
- schrittweise schließen: kleine Schritte zum großen Ziel...

• Hierarchie virtueller (auch: abstrakter) Maschinen

- Ebene i ; \mapsto Ebene $i-1$ durch Programme, für $i > 1$
- Maschinenprogrammebene (Ebene 3) \mapsto Betriebssystem

• Entvirtualisierung

- gedachte, unwirkliche, scheinbare Maschinen konkretisieren
- vor oder zur Programmlaufzeit stattfindende Arbeitsgänge
 - Übersetzung und Interpretation in den verschiedensten Facetten

Literaturverzeichnis

- [1] CHAMBERLAIN, S. ; TAYLOR, I. L.:
Using ld: The GNU Linker.
Boston, MA, USA: Free Software Foundation, Inc., 2003
- [2] ELSNER, D. ; FENLASON, J. :
Using as: The GNU Assembler.
Boston, MA, USA: Free Software Foundation, Inc., Jan. 1994
- [3] RITCHIE, D. M.:
/ You are not expected to understand this. */*.
<http://cm.bell-labs.com/cm/cs/who/dmr/odd.html>, 1975
- [4] TANENBAUM, A. S.:
Multilevel Machines.
In: *Structured Computer Organization*[5], Kapitel 7, S. 344–386
- [5] TANENBAUM, A. S.:
Structured Computer Organization.
Prentice-Hall, Inc., 1979. –
443 S. –
ISBN 0-130-95990-1
- [6] <http://www.hyperdictionary.com/computing/semantic+gap>