Praktisch-Angewandte Informatik

Prof. Joubert

WS 1995/96

Wilko Hein

Vorlesung

3 ..... Rechnersysteme

3.1 ..... Analogrechner
Vorteile: Hohe Geschwindigkeit
Nachteile: Ungenauigkeit, schwierig zu programmieren
Bsp: Rechenschieber

3.2 ..... Hybridrechner

3.3 ..... Digitalrechner
Vorteile: Leicht programmierbar, Zuverlässig, hohe Geschwindigkeit, großer Speicher, Ein-/Ausgabe, Kommunikation
Bsp: Digitalrechner

3.4 ..... Rechner mit gespeicherten Programmen
Speicher, Steuereinheit, Rechenwerk, Ein/Ausgabe

..... Harvard-Architektur
Getrennter Speicher für Daten und Befehle. Nur-Lesen und Lesen/Schreiben

..... Von-Neumann-Rechner
Gleicher Speicher für Daten und Befehle. Register: Befehlszähler, Stack-Zähler, Mehrzweckregister. Adressformate, Addressierungsarten.

3.5 ..... Rechengeschwindigkeit
Bessere Technologie (LSI -> VLSI)
Architektur
- Operationsprinzip
  - Datenflußrechner
- Struktur
  - Cache
  - Bus (Architektur, Bandbreite)
  - CISC -> RISC

3.6 ..... Mikroprozessoren

3.6.1 ..... Historische Entwicklung

3.6.2 ..... Architektur Von-Neumann-Rechner
Am Bus Mikroprozessor, Speicher, Platten, E/A

3.6.3 ..... Busse
Vorteil: Leichte Erweiterbarkeit
Daten-, Adress- und Steuerleitungen

4 ..... Rechnerarchitektur

4.1 ..... Einführung
Operationsprinzip
Hardwarestruktur

4.2 ..... Parallelrechnerarchitektur

4.3 ..... Rechnerarchitektur allgemein

4.3.1 ..... Operationsprinzip
Maschinen-Datentypen (Codierung der Informationen und Operationen)
V.N.R.: Semantische Lücke
Kontrollstruktur (Zeitliche Abfolge)

4.3.2 ..... Struktur
Prozessoren, Verbindungselemente, Speicher, Logik, Busse

4.4 ..... Von-Neumann-Architektur

4.4.1 ..... Operationsprinzip
Befehl holen, Befehlszähler erhöhen, Befehl decodieren und Adresse des Datums berechnen. Bitkette holen, Operation ausführen. Datum zurückschreiben
Von-Neumann-Flaschenhals: Holen, Abarbeiten. Streng sequentiell.
Jeweils genau ein Befehl oder ein Datum.

4.4.2 ..... Struktur
Am Bus liegen CPU, Arbeitsspeicher und E/A-Prozessoren

4.4.2.1 ..... Prozessor
Steuerwerk: Register, Microcode. Zustandsregister + Befehlszähler = Unterbrechungsvektor
Rechenwerk: Int / Float-Berechnung durch Arbeitsregister, ALU (Verknüpfungslogik), Status
Unterbrechungswerk: Zwischenspeicherung des Unterbrechungsvektors, Prioritätsermittlung

4.4.2.2 ..... Speicher
Speicheradreßregister, Datenkommunikationsregister

4.6 ..... Datenstrukturarchitektur
Nachteile von Von-Neumann: Semantische Lücke, keine Zugriffsbeschränkungen, physikalischer V-N-Flaschenhals: Bussystem nur zum holen und schreiben der Daten, intellektueller V-N-Flaschenhals: sequentielles Denken

4.7 ..... Datenflußarchitektur
Operationsprinzip: Impliziter Parallelismus
Datenabhängigkeitsanalyse; Datenfluß identisch mit Kontrollfluß. i1:(+ () 1 i3/1)

4.8 ..... RISC-Architektur

4.8.1 ..... Architekturprinzipien
Ein-Zyklus-Operationen (alle Befehle gleiche Ausführungszeit)
Load/Store-Architektur (Nur load/store auf Speicher, sonst nur Register)
Verzicht auf Microcode (nur festverdrahtete Instruktionen)
Einheitliches Befehlsformat
Wenige Befehle und Adressierungsarten

4.9 ..... Speicher (ortsadressierter Speicher)
sequentieller Zugriff
direkter RA
index-sequentieller
Mehr-Ebenen-System

4.10 ..... RAM/ROM

5 ..... Betriebssysteme
Systemsoftware, Anwendersoftware

5.1 ..... Einzelfunktionen des Betriebssystems
Bereitstellung von Hilfsmitteln (Mehr-Programm-Betrieb)
Bereitstellung der Ressourcen
Gerätetreiber

5.2.1.1 ..... Konstruktionsprinzip
Schichtenmodell, weiter abstrahieren

5.3 ..... Prozesse und deren Zustände

5.3.2 ..... Prozeßumschaltung und Mehrprogrammbetrieb

Bereit (Wartet auf Zuteilung des Prozessors)
Laufend (Ausführung erfolgt)
Blockiert (Wartet auf Ereignis, z.B. EA und nicht auf Prozessor)
Beendet

5.4.1 ..... Semaphoren-Konzept

Große Übung

0 ..... Inhalt

1 ..... Grundlagen

1.1 ..... Algorithmus

1.1.1 ..... Beispiel

1.1.2 ..... Charakterisierung des Begriffes „Algorithmus“

P. Schefe: Ein Algorithmus ist ein eindeutig bestimmtes Verfahren unter Verwendung von Grundoperationen über primitiven (gegebenen) Objekten.
Bauer/Goos: Ein Algorithmus ist eine präzise, d.h. in einer festgelegten Sprache abgefaßte, endliche Beschreibung eines allgemeinen Verfahrens unter Verwendung ausführbarer elementarer (Verarbeitungs-) Schritte.

Merkmale:
- Endlichkeit der Beschreibung
- Effektivität
- Determinismus
- Endlichkeit der Ausführung

Exakte Definition:
- Markov-Algorithmus
- Turing-Algorithmus
- while-Programme
- μ-rekursive Funktionen

1.2 ..... Mathematische Hilfsmittel

Zeitkomplexität, Speicherkomplexität, jeweils in Abhängigkeit von Fallgrößen.

1.2.1 ..... Definition „Ordnung“

$O(f) = \{ g \mid g: N_0 \rightarrow R_0^+,\ V(g) \subset V(f),\ $ Ex. n_0 natürlich, c pos. reell, $g(n) \leq c*f(n)$ alle $n \geq n_0\}$
$\Omega(f) = \{ g \mid g: N_0 \rightarrow R_0^+,\ V(g) \subset V(f),\ $ Ex. n_0 natürlich, c pos. reell, $g(n) \geq c*f(n)$ alle $n \geq n_0\}$
V(g) = Vorbereich von g

1.3 ..... Elementare Datenstrukturen

Felder, Verkettete Listen, Stapel, Schlangen, Bäume

1.3.1 ..... Felder

1.3.2 ..... Verkettete Listen

Vorwärts:
```
type T=record
  int : integer;
  next : ^T;
end;
```
2 Arrays:
```
int: array[1..n] of char
next: array[1..n] of integer;
```

1.3.3 ..... Stapel

Liste, bei der Einfügen und Löschen nur am Ende. Push and Pop.

1.3.4 ..... Schlangen

Einfügen vorne, Lesen hinten. Put and Get.

1.3.5 ..... Bäume

Preorder-Traversierung: Wurzel, links, rechts
Inorder-Traversierung: Links, Wurzel, rechts
Postorder-Traversierung: Links, rechts, Wurzel
Darstellung: Adjazenzmatrix, verkettete Liste

2. ..... Sortieralgorithmen

2.1 ..... Elementare Sortieralgorithmen

2.1.1 ..... Sortieren durch direktes Einfügen

Füge Element an richtiger Stelle in vorsortierte Liste (evtl. erster Teil der Liste) ein
$t \in O(n^2)$

2.1.2 ..... Sortieren durch direktes Auswählen

Suche kleinstes Element und tausche gegen erstes
$t \in O(n^2)$

2.1.3 ..... Sortieren durch direktes Austauschen (Bubblesort)

Durchlaufen und, falls nötig, mit Element von hinten vertauschen

2.2 ..... Rekursive Sortieralgorithmen

hard split - easy join
easy split - hard join

2.2.1 ..... Quicksort

hard-split - easy-join. C.A.R.Haare
Vergleichselement. Zeiger von links und rechts. Diese tauschen, wenn falsche Seite vom Vergleichselement. Bis Zeiger übereinander weg. Teillisten rekursiv sortieren, falls nicht einzelnes Element
$t \in O(n \log n)$
Günstig: Alle Elemente im Speicher. Schlecht: Vergleichselement ungünstig ($O(n^2)$), Verbesserung: Zufallsalgorithmus, für kleine Sequenzen anderen Algorithmus

2.2.2 ..... Heap-Sort

Aufbauphase: Baum erstellen, so daß jeder Knoten nicht kleiner als Nachfolger. Damit Wurzel (bis auf Gleichheit) größtes Element.

Sortierphase:
- Wurzel entfernen
- Diese vor Liste anfügen
- Baum wiederherstellen

Algorithmischer Aufbau: Knotenzahl berechnen, alle Blätter aus Liste füllen. Ebenenweise durchlaufen, weiteres Element eintragen und evtl absinken lassen.

Wiederherstellen: Wurzel entfernen, letztes Blatt unten rechts einsinken lassen.

Realisierung als Array: Vorgänger befindet sich auf (j div 2). Nachfolger auf 2*j, 2*j+1

2.2.3 ..... Mergesort

O(n log n)

2.3 ..... Bucket-Sort

Für Worte fixer Länge: Tabelle Alle Buchstaben * Alle Stellen. Beginnend mit letzem Buchstaben einsortieren. Dann den davorstehenden usw. Variable Länge: Worte nur mitsortieren, wenn Stelle auch im Wort drin

2.4 ..... Sortieren auf parallelem Rechnerm „Odd-even transposition"-Sort

Anzahl sortierender Elemente = Anzahl der Resourcen.

Benachbarte Prozessoren tauschen, falls nötig. Wechselweise links - rechts vergleichen

3 ..... Suchalgorithmen

3.1 ..... Sequentielles Suchen

Datei nicht notwendig sortiert. Abbruch bei EOF oder Gefunden.

O(n)

3.2 ..... Binäres Suchen

Sortiertes Feld. In zwei Teile und mit Trennelement vergleichen. Grenzen anpassen

3.3 ..... Hash-Verfahren

Hash-Funktion: Gute Streuung, Ausnutzung des kompletten Speichers, einfache Berechnung

a) h(k) = k mod m, m prim.

b) Zeichenkette: Bitweise XOR-Verknüpfung der Zeichen

Kollisionen: Verkettete Liste, dort sortiert. Lineares Austesten: In eine Richtung bis zum nächsten freien Platz ausweichen

3.4 ..... Binäre Suchbäume

Vorr.: Jedes Element nur genau einmal, Teilbäume geordnet.

Beim Löschen und zwei Teilbäume: Im linken das größte Element an diese Stelle setzen.

3.5 ..... B-Bäume

Ordnung n. Jeder Knoten höchstens 2*n Elemente; jeder Knoten außer Wurzel mindestens n Elemente. Knoten entweder Blatt oder einen Nachfolger mehr als Elemente. Alle Blätter auf gleicher Ebene