

Übung 10

Imperatives Programmieren

Maschinenmodell der RAM

Die Redcode RAM

Anwendungsdomäne

Beispiele

Implementierung



- Random Access Machine (RAM)
 - Maschine mit wahlfreiem Zugriff
- Bestandteile
 - Speicher
 - Behälter mit wahlfreiem Zugriff
 - Inhalt einer Speicherstelle:
Datum oder Befehl
 - Adresse: numerischer Zugriffsschlüssel
 - Programmzähler (PC)
 - Adresse des nächsten auszuführenden Befehls
 - Register
 - Besonders ausgezeichnete Speicherstellen (meist besonders schnell)
 - i.R. keine Indizierung möglich

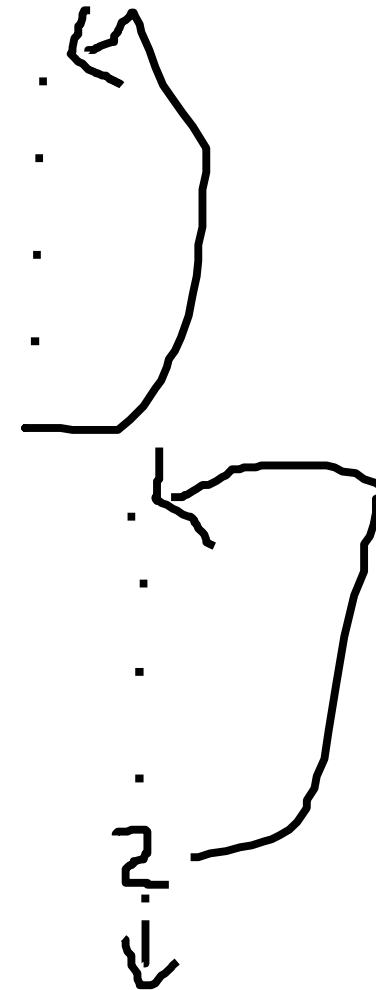
Adresse	Inhalt
0	MOV 0,1
1	DAT 0
2	DAT 0
...	...

PC
2

- Viele Operationen sind binär
 - Zwei Operanden
 - Ein Ergebnis
- Einadreßform: OPCODE OPERAND
 - Voraussetzung: Akkumulator für Zwischenergebnisse
 - Zweiter Operand und Ergebnis implizit gegeben
 - Achtung: Änderung des Operanden befehlsabhängig
 - LD x Acc := x
 - ADD y Acc := Acc + y
 - ST z z := Acc ($\equiv x + y$)
- Zweiadreßform: OPCODE OPERAND1 OPERAND2
 - Ergebnis implizit in einem der Operanden (meist im 2.)
 - MOV x z
 - ADD y z
- Dreiadreßform: OPCODE OPERAND1 OPERAND2 OPERAND3
 - Ziel für Ergebnis explizit angegeben
 - ADD x y z



- Speicherbefehle
 - LD, ST, MOV
- Arithmetisch-logische Befehle
 - NOT, AND, OR, XOR, SHIFT, ...
 - NEG, ADD, SUB, MUL, DIV, EXP, ...
- Kontrollfluß-Befehle
 - unbedingter Sprung
 - bedingte Sprünge
- Privilegierte Befehle
 - Thread starten / beenden
 - ...
- Ungültige Befehle
 - Alles, was nicht in der Liste enthalten ist



Indirektion	Position	Modifikation	Zeitpunkt
Immediate (unmittelbar) 0 <u>Imm</u> $x = \text{Val } x$	Absolute (absolut) Abs $x = \text{Ref } x$	None (Keine)	Never (Nie)
Direct (direkt) 3 <u>Dir</u> $x = \text{Ref } x$	Relative (relativ) Rel $x = \text{Ref } x + \text{PC}$	Increment (Erhöhen)	Pre (Vor Zugriff)
Indirect (indirekt) 2 Ind $x = \text{Ref } m[x]$		Decrement (Erniedrigen)	Post (Nach Zugriff)

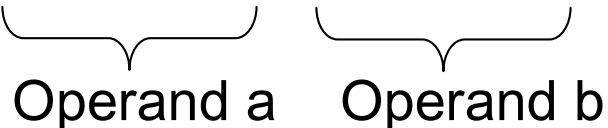
- Ergebnis der Adreßrechnung ist Wert oder Referenz
 - Mit Werten wird gerechnet
 - Je nach Operation werden Referenzen zum Lesen oder Schreiben dereferenziert



- Maschinenmodell
 - Speicher
 - Endlicher, zyklischer Speicher (Adreßrechnung modulo n)
 - Eine Speicherstelle enthält genau einen Befehl
 - Programmzähler
 - Anzahl unbegrenzt
 - Register
 - Keine

- Befehle in Zweiadreßform

- Instruction := Opcode Mode Value Mode Value





- Speicher
 - MOV Kopieren $b:=a$
- Arithmetisch
 - ADD Addition $b:=b+a$ (mod Speichergröße)
 - SUB Subtraktion $b:=b-a$ (mod Speichergröße)
- Kontrollfluß
 - JMP PC=a (unbedingt)
 - JMZ PC=a, wenn $b=0$
 - JMN PC=a, wenn $b \neq 0$
 - DJN PC=a, wenn $(--b) \neq 0$
 - CMP PC=PC+2, wenn $a = b$
 - SLT PC=PC+2, wenn $a < b$
- Priviligiert (Prozeßsteuerung)
 - SPL Starte neuen Thread bei Adresse a
- Ungültig
 - DAT Datum

Const → Ref
R → R

} bed.



- Imm x: Immediate

- Wert: x

- Dir x: Direkt, PC-relativ

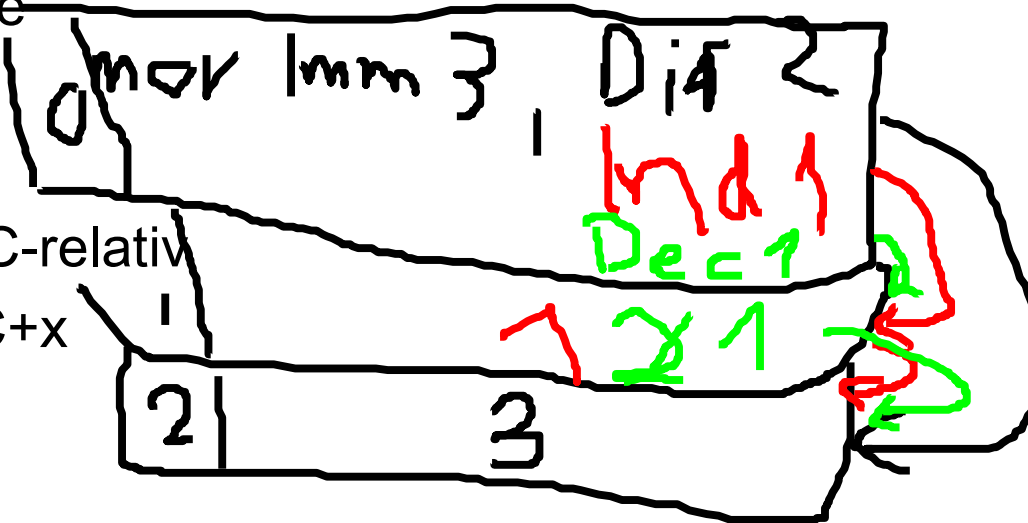
- Referenz: PC+x

- Ind x: Indirekt, doppelt PC-relativ

- Referenz: PC+x+ mem[PC+x]

- Dec x: Indirekt, PC-relativ, Prä-Dekrement

- Referenz: PC+x+ --mem[PC+x]



- Darstellung von Adressierungsmodi durch Sonderzeichen
 - Standardfall: Direct
 - # = Immediate
 - @ = Indirect
 - < = Predekrement
- Symbolische Namen für Speicherstellen
 - start: MOV ^Dstart [↑]next
 - next: DAT 0 0

Vergressen



- Core Wars Turniere
 - Programme mehrerer Autoren laufen in einer Maschine
 - Abwechselnd wird je ein Befehl jedes Programmes ausgeführt
 - Innerhalb eines Programmes beliebig viele Threads
 - Sieger ist, wer alle Threads des Gegners zum Absturz bringt
 - Remis tritt nach einer vordefinierten Anzahl von Zügen ein

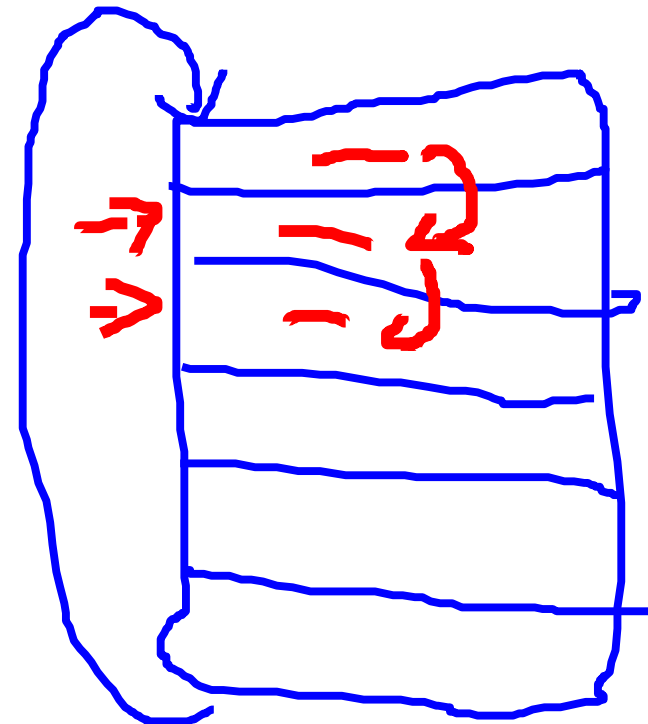
$$A = \{a_1, a_2, a_3\}$$

$$B = \{b\}$$

$$a_1, b, a_2, b, a_3, b, \dots$$



- Programm
 - `MOV Dir 0 Dir 1`
- Beschreibung
 - Überschreibt den gesamten Speicher durch Kopien seiner selbst
- Eigenschaften
 - Mobil (Ausführung nicht auf ursprüngliches Programm beschränkt)
 - Einfädig
- Kann maximal Remis erreichen



▪ Programm

- DAT Imm 0 Imm 0
- ADD Imm 4 Dir -1 <- Startadresse
- MOV Dir -2 Ind -2
- JMP Dir -2 Imm 0

▪ Verhalten

- Schießt mit ungültigen Instruktionen in den Speicher

▪ Eigenschaften

- Stationär (Ausführung auf ursprüngliches Programm beschränkt)
- Einfädig



▪ Programm

- DAT Imm 0 Imm 0
- MOV Imm 12 Dir -1 <- Startadresse
- MOV Ind -2 Dec 5
- DJN Dir -1 Dir -3
- SPL Ind 3 Imm 0
- ADD Imm 653 Dir 2
- JMZ Dir -5 Dir -6
- DAT Imm 833 Imm 833

▪ Beschreibung

- Kopiert sich selbst an entfernte Adressen und startet neue Threads

▪ Verhalten

- Mobil
- Mehrfädig



- Bisher bekannt: Listen, Paare, Enumerationen
- Wie können wir wahlfreien Zugriff wiedergeben?
- Datentyp Array

- `type a = Array IndexTyp WertTyp`

- Initialisierung: `a = array (min,max) [(Index, wert)]`

- Lesen

- `a !! adr`

- Schreiben

- `a // [(Index,wert)]`

Belegang



```
-- redcode instructions
data Insn    = Insn Opcode Mode Int Mode Int
              deriving (Eq, Show)

-- redcode opcodes
data Opcode = DAT | MOV | ADD | SUB | JMP | JMZ |
             JMN | DJN | CMP | SPL | SLT
             deriving (Eq, Ord, Enum, Show)

-- redcode addressing modes
-- (Dec is predecrement-indirect)
data Mode    = Imm | Dir | Ind | Dec
              deriving (Eq, Ord, Enum, Show)
```



- Nächstes Programm bestimmen
 - Nächsten Thread des Programms bestimmen (= PC)
 - Befehl aus Speicher laden
 - Operanden a,b dekodieren
 - Fallunterscheidung über Adressierungsmodi
 - evtl. aus Speicher laden / verändern
 - Befehl ausführen
 - Fallunterscheidung über Opcode
 - evtl. Speicher verändern,
PC verändern,
Thread stoppen / starten



- Von Grund auf neu bauen
 - Gegeben Redcode Spec
 - (Evtl.: Baue Übersetzer in Zwischendarstellung)
 - Baue Interpreter für Zwischendarstellung
 - Herumspielen
- Mit Gerüst
 - Gegeben Redcode Spec und vorgegebenen Rahmen
 - Fülle die Lücken aus
(i.W. Fallunterscheidungen für Adressierung, Opcodes)
 - Herumspielen
- Mit Wiederverwendung (Faule Auswertung)
 - Caesar + 3er Transposition:
AOAVWJUSVHZSZZOZSHPOLKUZPVAVUMAKVPBUZKHHORZ
YSOYBVLKLAKHZSBZOBLIZUNHISALAZLYZSHLODUPOOHJIA
ZASHVAKPAGEWE



- Gesucht:
 - Das beste Redcode-Programm
 - Muß in der Musterlösungs-VM laufen
- Austragung
 - 1. Rechnerübung 2003
 - Turniermodus je nach Andrang (Absprache mit dem Tutor)
 - KO
 - KO mit Hin- und Rückspiel
 - evtl. Qualifikationsrunden?
- Preise für die Erst- und Zweitplatzierte(n) jedes Tutoriums



- Originalpapier
 - <http://www.koth.org/info/akdewdney/index.html>
- Redcode 88 Spec
 - <ftp://www.koth.org/pub/corewar/documents/tutorial.1.Z> und <ftp://www.koth.org/pub/corewar/documents/tutorial.2.Z>
- CoreWar FAQ
 - <http://homepages.paradise.net.nz/~anton/cw/corewar-faq.html>
- Inspiration für GUIs
 - <http://corewars.sourceforge.net/>
- Prozessorarchitektur
 - Silc, Robic, Ungerer: *Processor Architecture. From Dataflow to Superscalar and Beyond*. Springer, 1999.

