

# Grundlagen Rechnertechnik

Prof. Dr. Peter Gerwinski

15. November 2012

## 2.7 Struktur von Assembler-Programmen

### IA-32-Assembler

Befehl Operanden



addl \$1, %eax

movb %al, b

cmpb (%ebx), %dl

jbe .L2

## 2.7 Struktur von Assembler-Programmen

### IA-32-Assembler – Adressierungsarten

unmittelbar

`addl $1, %eax` ← Register

`movb %al, b` ← Speicher (absolut)

`cmpb (%ebx), %dl`

`jbe .L2`

indirekt mit Register

Speicher (relativ)

## 2.7 Struktur von Assembler-Programmen

Core War[s] (Krieg der Kerne) – Redcode (ICWS '88)

Virtuelle Maschine: Memory Array Redcode Simulator (MARS)

Aufgabe: Schreiben Sie ein Redcode-Programm,  
das die Gegner **Nothing**, **Knirps** und **Mice** besiegt.

(ICWS-88-Standard, max. 64 Prozesse, Speichergröße zufällig)

Teams bis zu 3 Personen sind zulässig.

Instruktionen:

**dat B** – Daten – „Du hast verloren!“

**mov A, B** – kopiere A nach B

**add A, B** – addiere A zu B

**sub A, B** – subtrahiere A von B

**jmp A** – unbedingter Sprung nach A

**jmz A, B** – Sprung nach A, wenn  $B = 0$

**jmn A, B** – Sprung nach A, wenn  $B \neq 0$

**djn A, B** – „decrement and jump if not zero“

**cmp A, B** – „compare“: überspringe, falls gleich

**spl A** – „split“: Programm verzweigen

Adressierungsarten:

grundsätzlich: Speicher relativ

**#** – unmittelbar

**\$** – direkt (auch ohne **\$**)

**@** – indirekt

**<** – indirekt mit Prä-Dekrement

Programm „Nothing“:

**jmp 0**

Programm „Knirps“:

**mov 0, 1**

## 2.7 Struktur von Assembler-Programmen

Unbedingte Verzweigung

Beispiel: Endlosschleife von „Nothing“

```
jmp 0
```

Bedingte Verzweigung

Beispiel: Kopierschleife von „Mice“

```
ptr      dat #0
start    mov #12, ptr
loop     mov @ptr, <dest
          djn loop, ptr
          spl @dest
          add #653, dest
          jnz start, ptr
dest     dat #833
          end start
```

# 3 Architekturmerkmale von Prozessoren

## 3.1 Speicherarchitekturen

### Bezeichnungen

- *Bit* = 0 oder 1 – kleinste Einheit an Information
- *Byte* = Zusammenfassung mehrerer *Bits* zu einer Binärzahl, die ein Zeichen (*Character*) darstellen kann, häufig 8 Bits (*Oktett*)
- *Speicherwort* = Zusammenfassung mehrerer Bits zu der kleinsten adressierbaren Einheit, häufig 1 Byte
- *RAM* = *Random Access Memory* = Hauptspeicher
- *ROM* = *Read Only Memory* = nur lesbarer Speicher

# 3 Architekturmerkmale von Prozessoren

## 3.1 Speicherarchitekturen

Verschiedene Arten von Speicher

- *Prozessor-Register*  
können direkt mit ALU verbunden werden,  
besonders schnell (Flipflops),  
überschaubare Anzahl von Registern
- *Hauptspeicher*  
kann direkt adressiert und mit Prozessor-Registern abgeglichen werden,  
heute i. d. R. dynamischer Speicher (Kondensatoren)
- *I/O-Ports*  
sind spezielle Speicheradressen, über die  
mit externen Geräten kommuniziert wird
- *Massenspeicher*  
liegt auf externem Gerät, wird über I/O-Ports angesprochen,  
Festplatte, Flash-Speicher, ...

# 3 Architekturmerkmale von Prozessoren

## 3.1 Speicherarchitekturen

- *Von-Neumann-Architektur*

Es gibt nur 1 Hauptspeicher, in dem sich sowohl die Befehle als auch die Daten befinden.

Vorteil: Flexibilität in der Speichernutzung

Nachteil: Befehle können überschrieben werden.

—> Abstürze und Malware möglich

- *Harvard-Architektur*

Es gibt 2 Hauptspeicher. In einem befinden sich die Befehle, im anderen die Daten.

Vorteil: Befehle können nicht überschrieben werden

—> sicherer als Von-Neumann-Architektur

Nachteile: Leitungen zum Speicher (Bus) müssen doppelt vorhanden sein, freier Befehlsspeicher kann nicht für Daten genutzt werden.

- Weitere Kombinationen

Hauptspeicher und I/O-Ports gemeinsam oder getrennt,

Hauptspeicher und Prozessorregister gemeinsam oder getrennt



## 3 Architekturmerkmale von Prozessoren

### 3.1 Speicherarchitekturen

Beispiele:

- Intel IA-32 (i386, Nachfolger und Kompatible):  
Von-Neumann-Architektur (plus Speicherschutzmechanismen),  
Prozessorregister und I/O-Ports vom Hauptspeicher getrennt
- Atmel AVR (z. B. ATmega):  
Harvard-Architektur (Befehlsspeicher als Flash-Speicher grundsätzlich  
auch schreibbar),  
Prozessorregister und I/O-Ports in gemeinsamem Adressbereich mit  
Hauptspeicher
- 6502 (heute: Renesas-Mikro-Controller):  
Von-Neumann-Architektur,  
I/O-Ports in gemeinsamem Adressbereich mit Hauptspeicher,  
Prozessorregister und Hauptspeicher getrennt

## 3 Architekturmerkmale von Prozessoren

### 3.2 Registerarchitekturen

- Mehrere Register, einzeln ansprechbar
- *Akkumulator*: Nur 1 Register kann rechnen.
- *Stack-Architektur*: Stapel, „umgekehrte Polnische Notation“

Je nachdem, auf welche dieser Arten die Register eines Prozessors organisiert sind, muß er auf völlig unterschiedliche Weise programmiert werden.

# 3 Architekturmerkmale von Prozessoren

## 3.2 Registerarchitekturen

Beispiele:

- Intel IA-32 (i386, Nachfolger und Kompatible):  
Mehrere Register, für verschiedene Zwecke spezialisiert (unübersichtlich),  
Fließkommaregister: Stack-Architektur
- Atmel AVR (z. B. ATmega):  
32 Register
- 6502 (heute: Renesas-Mikro-Controller):  
3 Register: A, X, Y. Nur A kann rechnen → Akkumulator
- Java Virtual Machine (JVM):  
Stack-Architektur

## 3 Architekturmerkmale von Prozessoren

### 3.3 Befehlssatzarchitekturen

- *Complex Instruction Set Computer (CISC)*

Umfangreiche Befehlssätze, mächtige Befehle

→ komfortable manuelle Programmierung in Assembler

→ längere Abarbeitungszeit der einzelnen Befehle

Realisierung: „Prozessor im Prozessor“ – *Mikroprogramme*

Beispiel: IA-32

- *Reduced Instruction Set Computer (RISC)*

wenige, wenig mächtige Befehle

→ Programmierung in Assembler für Menschen unkomfortabel

→ schnelle Abarbeitung der Befehle

Beispiel: Atmel AVR

- Weitere Befehlssatzarchitekturen: VLIW, EPIC (z. B. IA-64)
- Orthogonaler Befehlssatz