



- Simulator für einen konfigurierbaren Vektorprozessor -

Übersicht:

- (1) Aufgabenstellung
- (2) Einführung HiCoVec
- (3) Anforderungen
- (4) Realisierung HiCoSim
- (5) Demonstration
- (6) Zusammenfassung und Ausblick

Aufgabe & Ziel

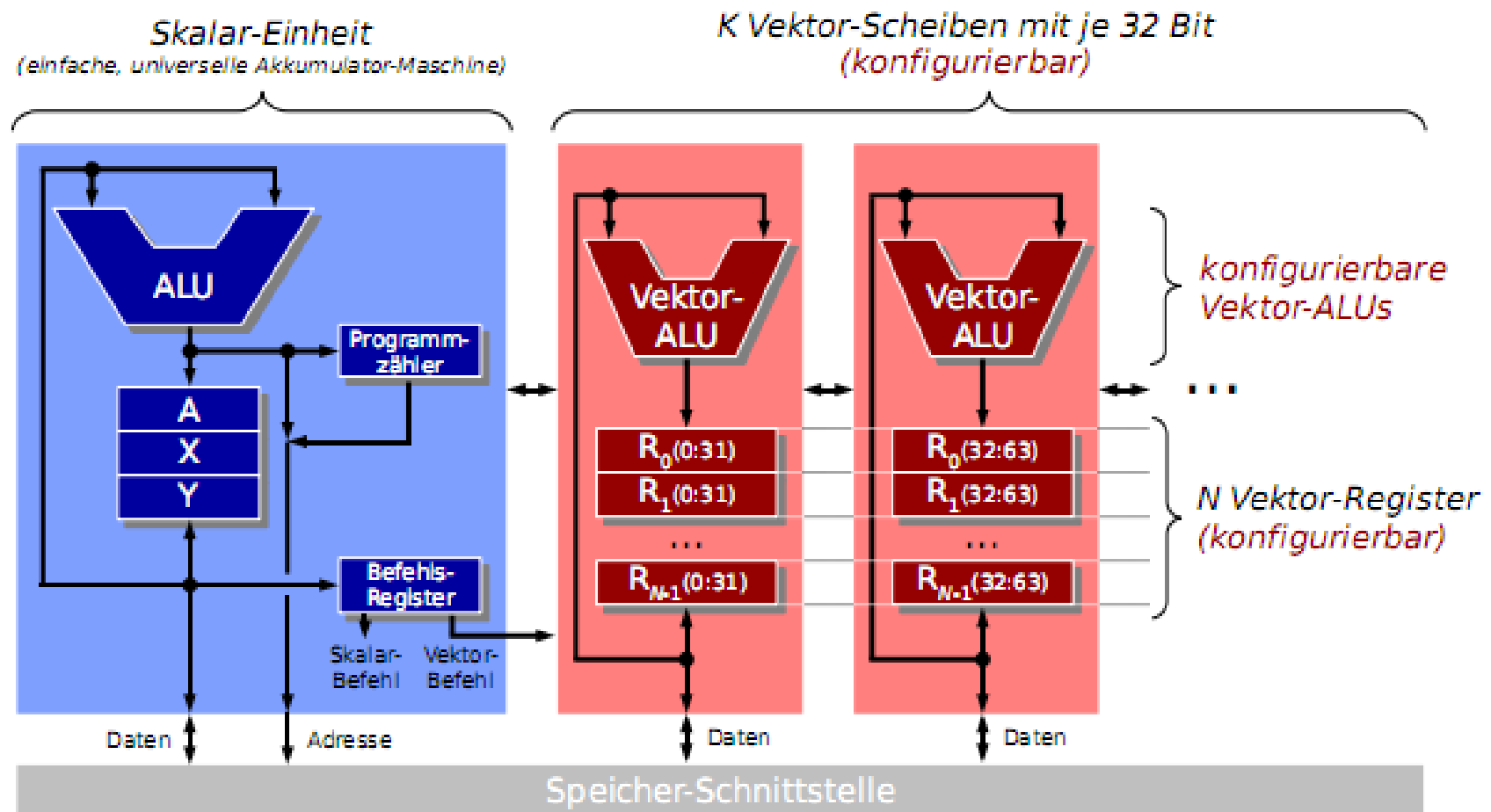
Aufgabe:

- Erstellung eines erweiterbaren Simulators für einen konfigurierbaren Vektorprozessor (HiCoVec)

Ziel:

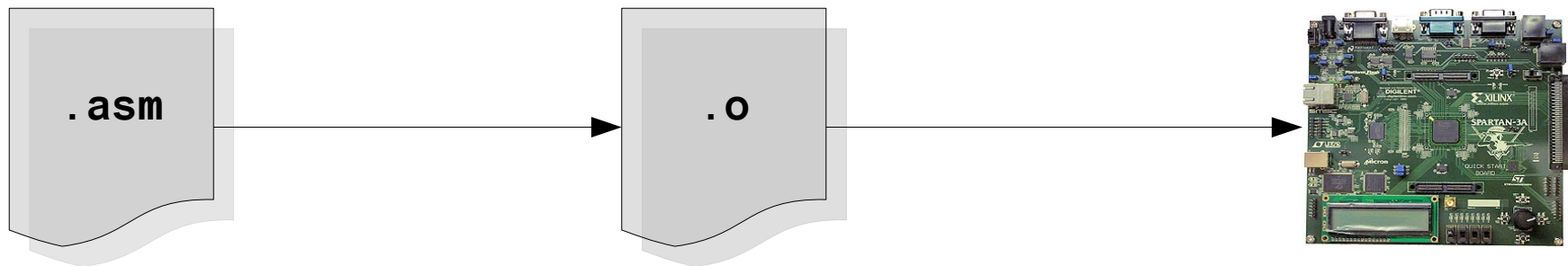
- Ausführung und Leistungsanalyse von beliebigen HiCoVec-Programmen
- Softwareentwurf auch ohne Hardware
- Simulation von beliebigen HiCoVec-Konfigurationen

- **Highly Configurable Vector Processor**
- Open Source
- An HS Augsburg entwickelt (Prof. Dr. Kiefer)
- Beschreibung in Diplomarbeit (H. Manske)
- Realisierung auf FPGA-Board
- Entwicklungsumgebung inkl. Assembler
- Ausblick:
 - C-Compiler (in Entwicklung)
 - Erweiterungen, z.B. Gleitkommaeinheit



Quelle: HiCoVec-Flyer für Embedded World

- Assemblerprogramm schreiben
- Assemblieren mit scotchAS
- Objektdaten auf FPGA laden
- On-Chip-Debugging möglich (aber eingeschränkt)
- Kein Testen ohne Hardware möglich



Anforderungen

- Simulation beliebiger HiCoVec-Programme
- Unterstützung beliebiger Konfigurationen
- Protokollierung des Programms (Statistiken)
- Erweiterbarkeit des Befehlssatzes
- Erweiterbarkeit der Peripherie
- Grafische Oberfläche unabhängig von Simulator
- Benutzerfreundlich und Portierbar

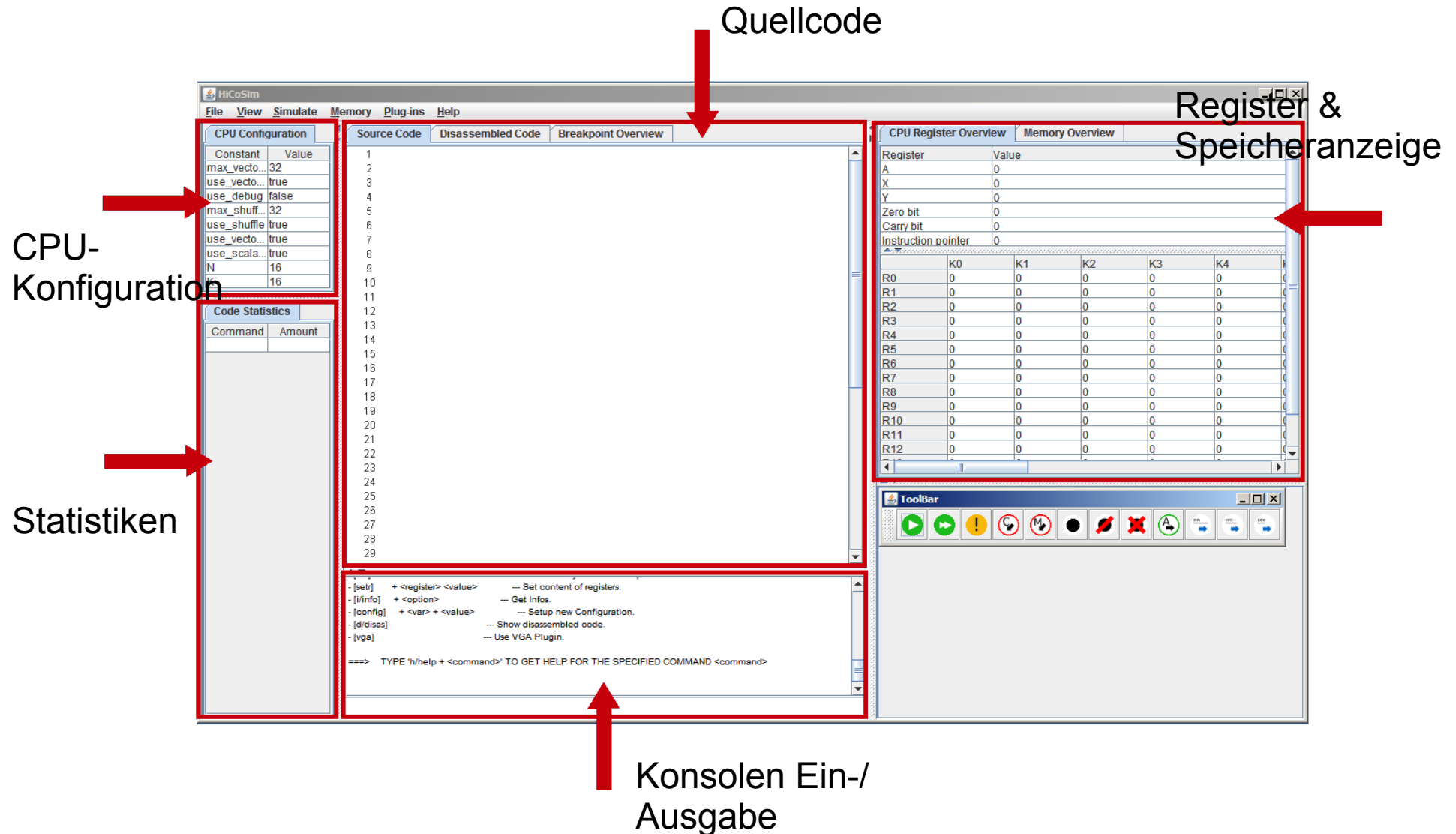
Komfortabler Software- Entwurf durch:

- Breakpoints setzen
- Watchpoints setzen
- Disassembler
- Simulation unterbrechen & Fortsetzen
- Single Stepping
- Speicher zur Laufzeit manipulierbar
- Programmanalyse durch Statistiken
- Anzeige des aktuellen Zustands (z.B. Register)
- Änderbare Konfiguration des HiCoVec

Übersicht Realisierung

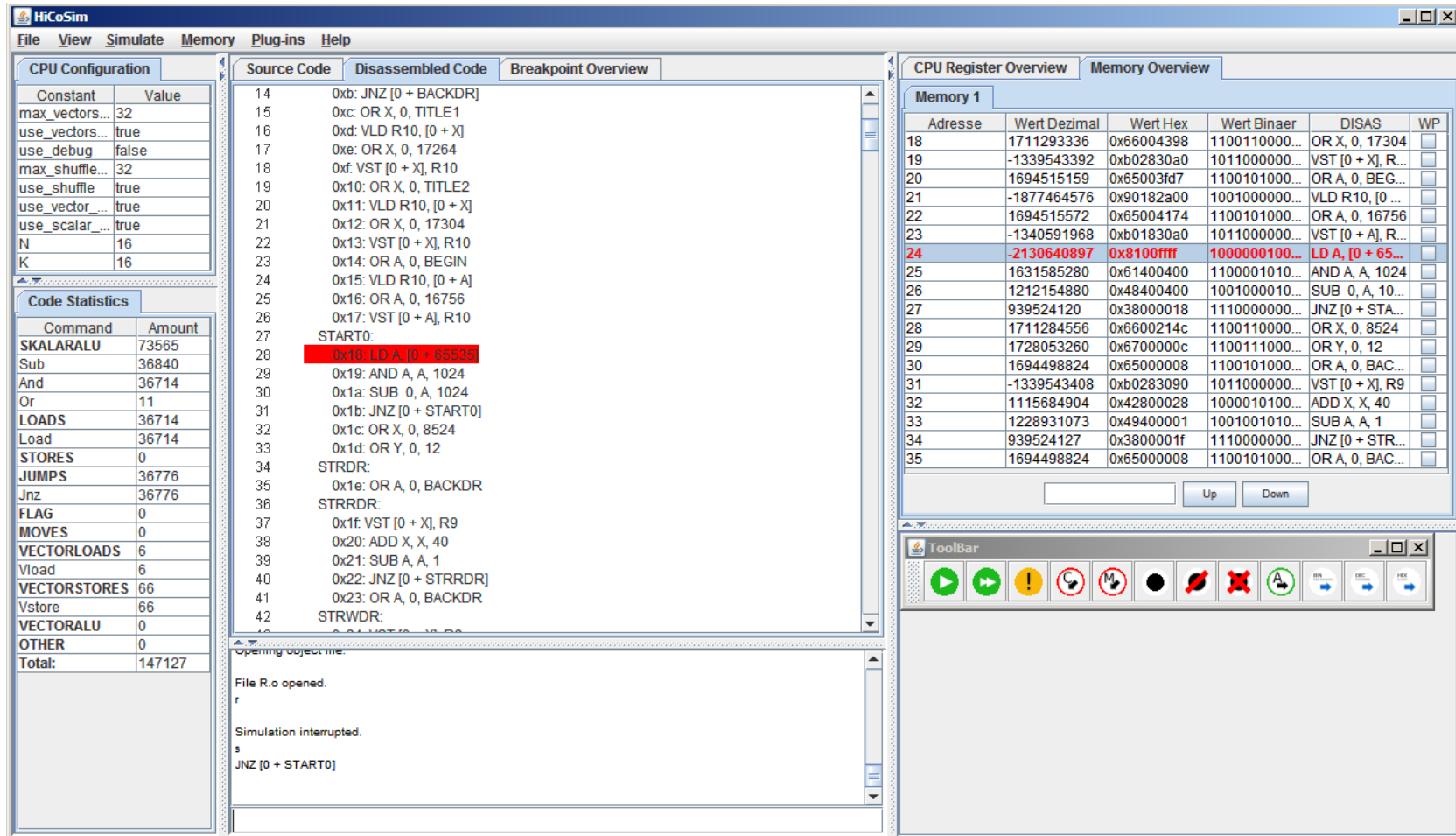
- Aufteilung in Backend / Frontend
 - *Backend*:
 - Simulation des HiCoVec
 - Einfache Konsoleneingabe
 - *Frontend*:
 - Benutzereingaben und Anzeige
- Verwendung von Java als Programmiersprache
 - Java 6
 - GUI: Swing
 - keine externe Bibliotheken notwendig

- Benutzerfreundliche und konfigurierbare Oberfläche
- Aufrufe der Backend-Methoden
- Grafische Darstellung für:
 - Code (Quellcode/Disassembliert)
 - Register & Speicher
 - Statistiken
 - CPU-Konfiguration
 - Konsole



(4) Realisierung

Programmablauf GUI



The screenshot displays the HiCoSim simulator interface. The main window is divided into several panes:

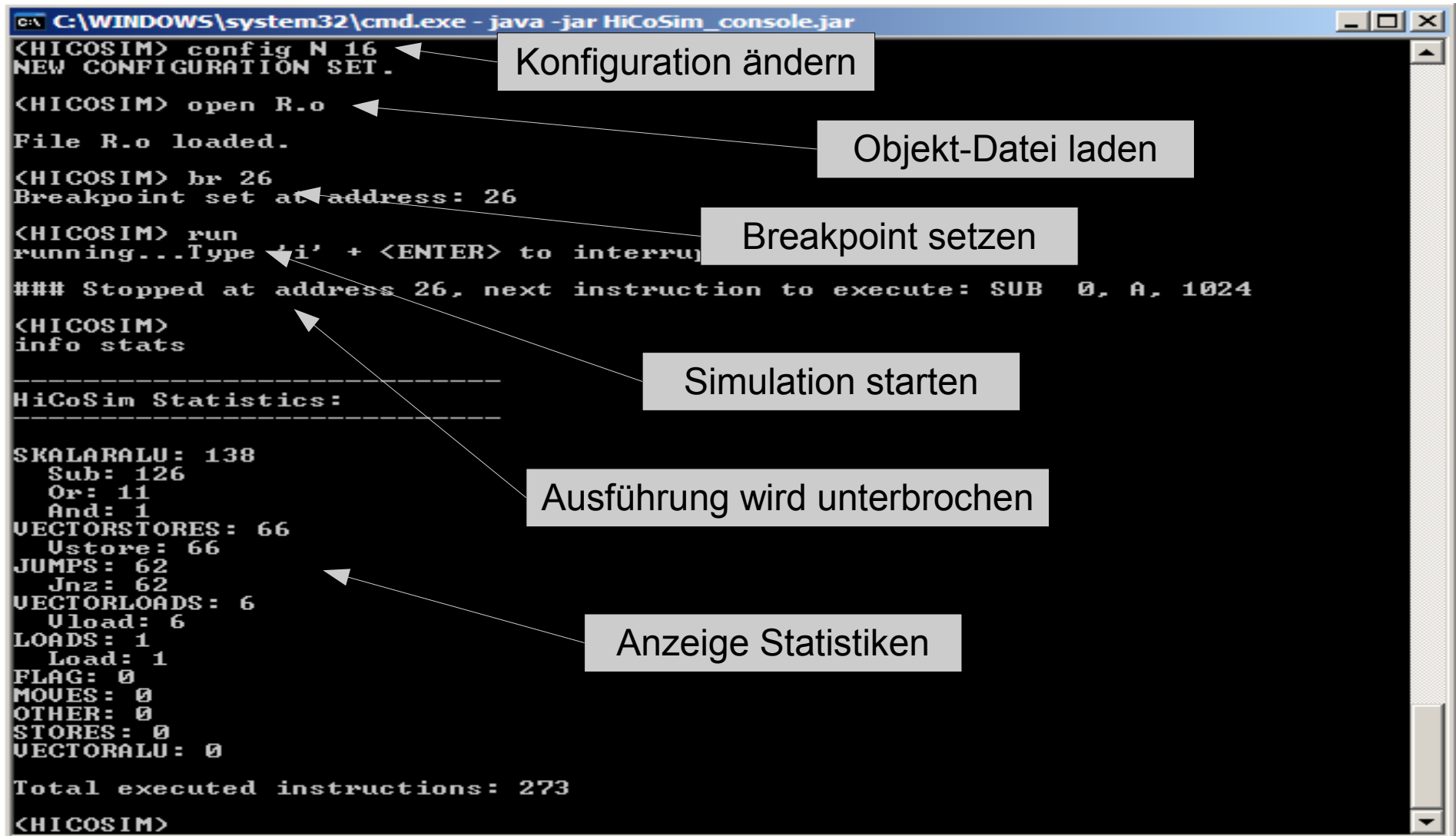
- CPU Configuration:** A table with columns 'Constant' and 'Value'. It lists various configuration parameters like 'max_vectors', 'use_vectors', 'use_debug', 'max_shuffle', 'use_shuffle', 'use_vector', 'use_scalar', 'N', and 'K'.
- Code Statistics:** A table with columns 'Command' and 'Amount'. It shows the frequency of various instructions like 'SKALARALU', 'Sub', 'And', 'Or', 'LOADS', 'Load', 'STORES', 'JUMPS', 'Jnz', 'FLAG', 'MOVES', 'VECTORLOADS', 'Vload', 'VECTORSTORES', 'Vstore', 'VECTOTALU', and 'OTHER'.
- Source Code / Disassembled Code:** A central pane showing assembly code. Line 28 is highlighted in red: `0x18: LD A, [0 + 65535]`.
- CPU Register Overview:** A table showing the current state of CPU registers, including 'Adresse', 'Wert Dezimal', 'Wert Hex', 'Wert Binaer', 'DISAS', and 'WP'.
- Memory Overview:** A table showing memory contents, including 'Adresse', 'Wert Dezimal', 'Wert Hex', 'Wert Binaer', 'DISAS', and 'WP'.
- ToolBar:** A row of icons for simulation control, including play, pause, stop, and other functions.

- unabhängig von GUI ausführbar durch Callbacks
- Simulation des HiCoVec
 - Verwaltung von CPU und Speicher
- Ablaufsteuerung (starten, unterbrechen)
- Einfache Konsolen Ein- und Ausgabe für scriptgesteuerte Aufrufe

Befehle auf Konsolenebene

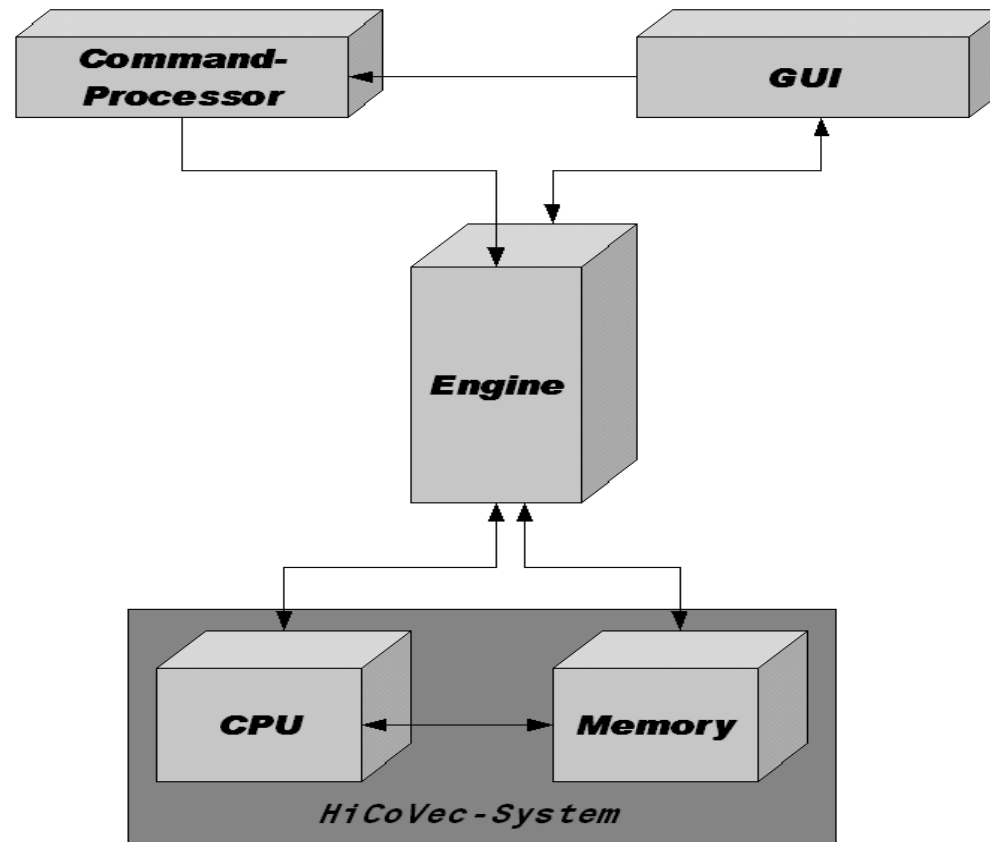
- > **open**
Öffnet Objektdaten und initialisiert Speicher
- > **list**
Zeigt Sourcecode an
- > **run**
Startet Simulation
- > **i**
Unterbricht Programmausführung
- > **step**
Führt einen Step durch
- > **br**
Setzt breakpoint. Beispiel: br 0x15
- > **set / setr**
Setzt Speicher/Register. Beispiel: setr A 0x3
- > **get**
Gibt Wert an Speicherstelle zurück. Beispiel: get 0x74

Programmablauf Konsole

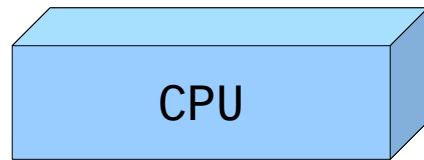


```
C:\WINDOWS\system32\cmd.exe - java -jar HiCoSim_console.jar
<HICOSIM> config N 16
NEW CONFIGURATION SET.
<HICOSIM> open R.o
File R.o loaded.
<HICOSIM> br 26
Breakpoint set at address: 26
<HICOSIM> run
running...Type ^i^ + <ENTER> to interrupt
### Stopped at address 26, next instruction to execute: SUB 0, A, 1024
<HICOSIM> info stats
-----
HiCoSim Statistics:
-----
SKALARALU: 138
  Sub: 126
  Or: 11
  And: 1
VECTORSTORES: 66
  Ustore: 66
JUMPS: 62
  Jnz: 62
VECTORLOADS: 6
  Uload: 6
LOADS: 1
  Load: 1
FLAG: 0
MOVES: 0
OTHER: 0
STORES: 0
VECTORALU: 0
Total executed instructions: 273
<HICOSIM>
```

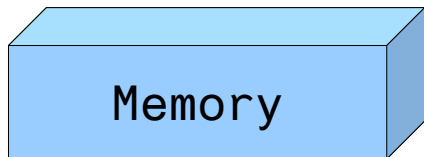
Struktur des HiCoSim



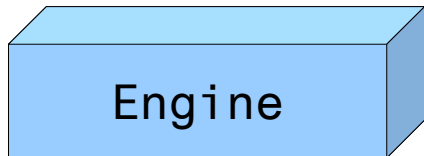
Die wichtigsten Module



- Abbildung des HiCoVec-Prozessors inkl. Register
- Decodierung und Ausführung der Maschinenbefehle
- Führt Lese-/Schreiboperationen auf Memory durch
- Lädt Prozessor-Konfiguration

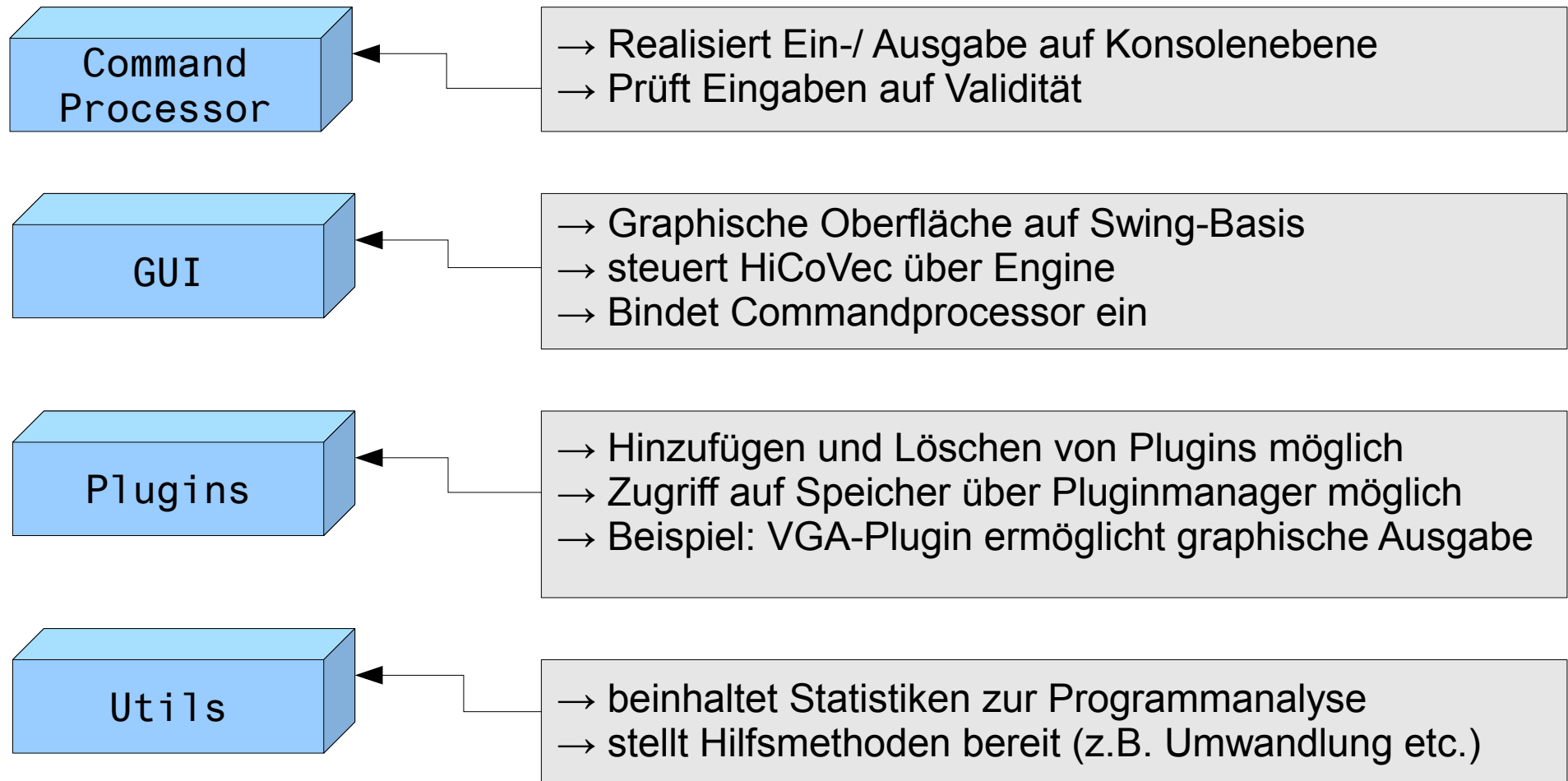


- Organisiert Speicher in Form einer SortedMap
- stellt Methoden zum Laden der Objektdatensatz bereit

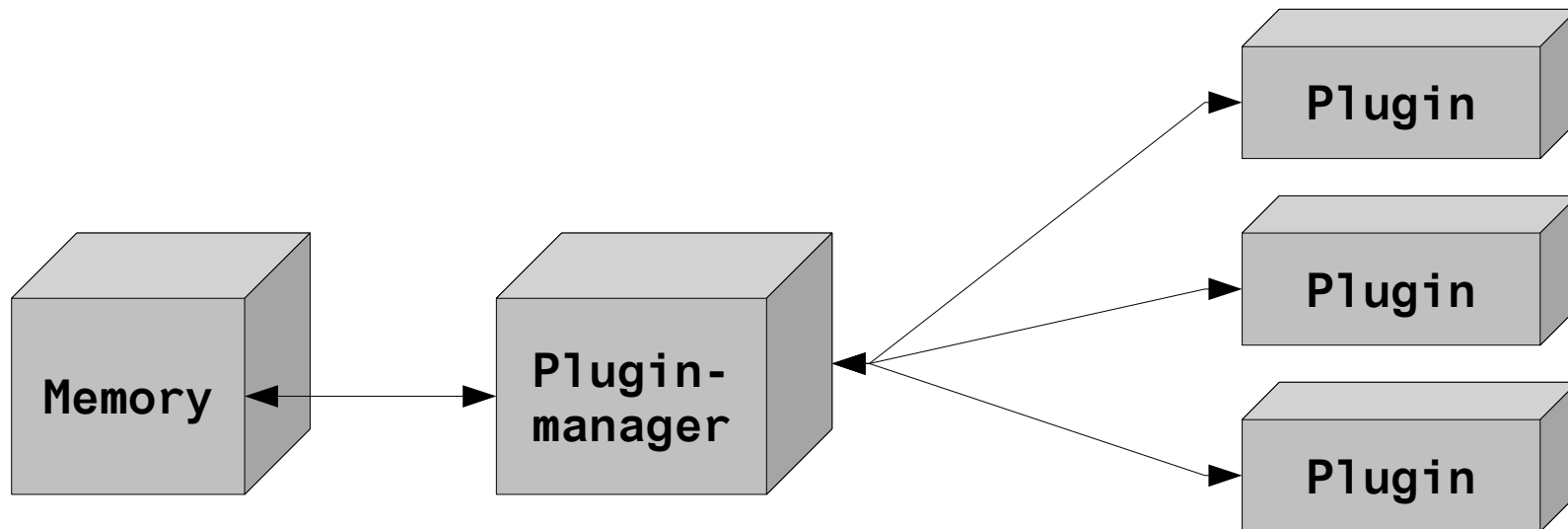


- Schnittstelle zwischen HiCoVec und GUI/Konsole
- Methoden zur Ablaufsteuerung des System (run, step)
- Benachrichtigt GUI bei Änderungen
- Kommunikation mit GUI über Listener

Die wichtigsten Module



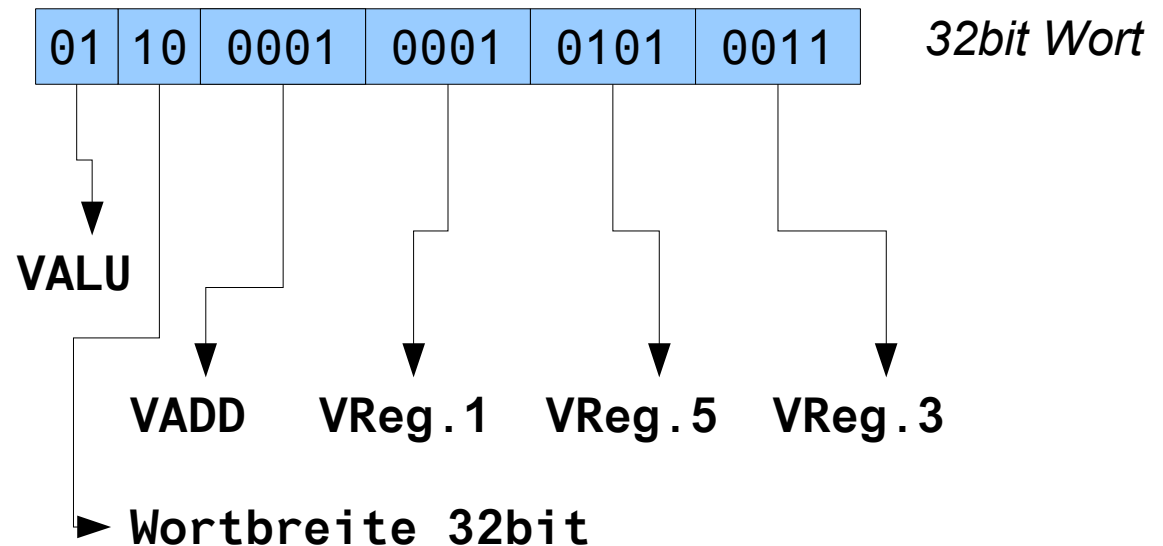
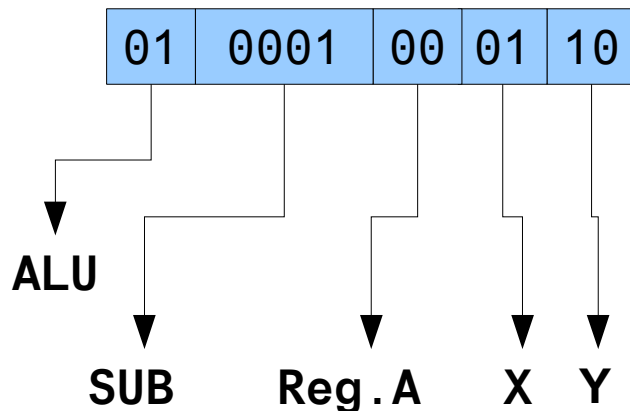
- PluginManager verwaltet Plugins zentral
- Plugins müssen Interface implementieren
- Observer-Modus möglich



Dekodieren eine Maschinenbefehls

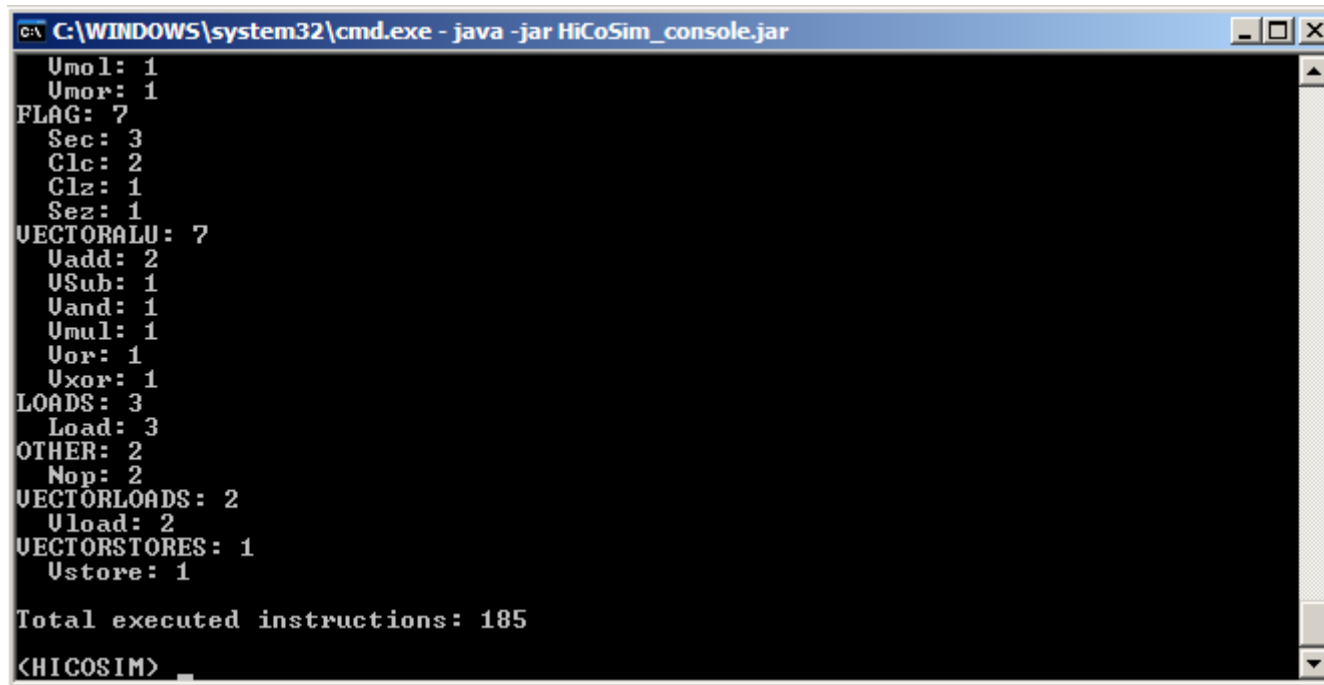
```
....  
.org 0x00  
.start  
ADD A, X, 0 + 5  
SUB A, X, Y # VADD.DW R1, R5, R3  
....  
HALT  
.end
```

- Wortlänge Prozessor: 32bit
- Besonderheit: parallele Befehle möglich
- Bei Vektorbefehlen: Wortbreite möglich
- Aufteilung:
 - 12bit für Skalarbefehle
 - 20bit für Vektorbefehle



Demonstration Konsole

Ablauf Testdatei all.o – Ausführung aller im HiCoSim implementierter Befehle



```
C:\WINDOWS\system32\cmd.exe - java -jar HiCoSim_console.jar
Umul: 1
Uxor: 1
FLAG: 7
Sec: 3
Clc: 2
Clz: 1
Sez: 1
VECTORALU: 7
Uadd: 2
USub: 1
Vand: 1
Umul: 1
Uor: 1
Uxor: 1
LOADS: 3
Load: 3
OTHER: 2
Nop: 2
VECTORLOADS: 2
Uload: 2
VECTORSTORES: 1
Ustore: 1

Total executed instructions: 185
<HICOSIM> _
```

→ Aufgabe: Addition zweier Arrays, je 16 Wörter

vek1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	+															
vek2	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	=															
ERG	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17

Skalareinheit:

```
OR Y, 0, 0
LOOP: LD A, [Y + vek1]
      LD X, [Y + vek2]
      ADD A, A, X
      ST [Y + erg], A
      INC Y, Y
      SUB 0, Y, 16
      JNZ [0 + LOOP]
```

C-Code:

```
for(i=0;i<16;i++){
    erg[i] = vek1[i] + vek2[i];
}
```

Vektoreinheit:

```
OR A, 0, vek1
OR X, 0, vek2
OR Y, 0, erg
VLD R0, [0 + A]
VLD R1, [0 + X]
VADD.W R2,R0, R1
VST [0 + Y], R2
```

Demonstration Scotchrace

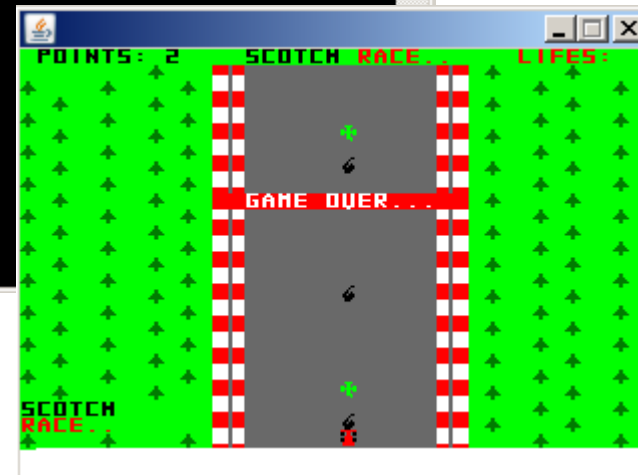
```
C:\WINDOWS\system32\cmd.exe - java -jar HiCoSim_console.jar
-- [reset]          --- Reset.
-- [i]              --- Interrupt. Type 'i' + <ENTER>
> while processor is running.
-- [get]            --- Show memory content.
-- [set]            --- Set memory content at a spec
ified address.
-- [setr]           + <register> <value>          --- Set content of registers.
-- [i/info]         + <option>                    --- Get Infos.
-- [config]         + <var> + <value>             --- Setup new Configuration.
-- [d/disas]        --- Show disassembled code.
-- [vga]            --- Use VGA Plugin.

==> TYPE 'h/help + <command>' TO GET HELP FOR THE SPECIFIED COMMAND <command>
>

<HICOSIM> o R.o
File R.o loaded.

<HICOSIM> vga start

<HICOSIM> run
running...Type 'i' + <ENTER> to interrupt.
```



Team

Backend

Müge Özugur

Michael Wager

Andreas Weber

Luca Calchera

Hibetoullah Ftima

Frontend

Sebastian Minning

Daniel Obermüller

Projektleiter

Sebastian Minning

Betreuer

Prof. Dr. Kiefer

***Wir freuen uns auf ihren Besuch an
unserem Stand im Raum J2.01***

Zusammenfassung

Umfang:

Codezeilen: 16.300

54 Klassen

468 Methoden

Fazit:

- Software-Entwurf auch ohne Hardware möglich
- Unterstützung für Compileroptimierungen
- Einfacher Test verschiedener CPU-Konfigurationen