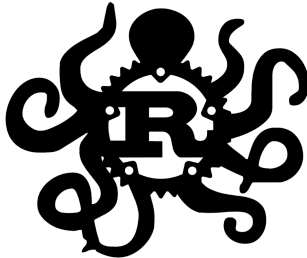
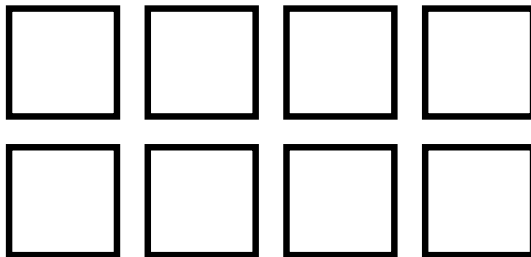


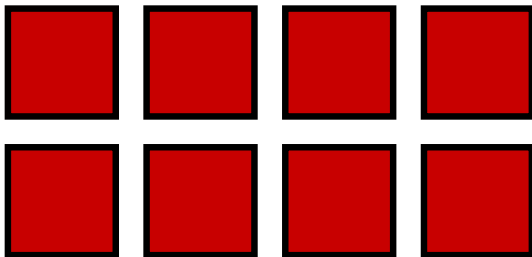
# Invasives Rust

Hermann Heinz Erich Krumrey

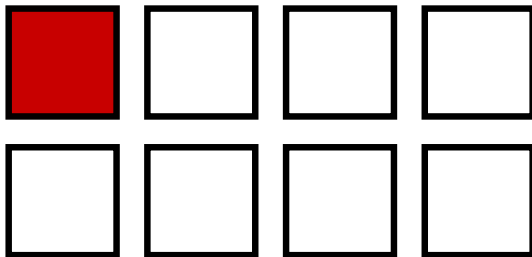
Lehrstuhl Programmierparadigmen, IPD Snelting



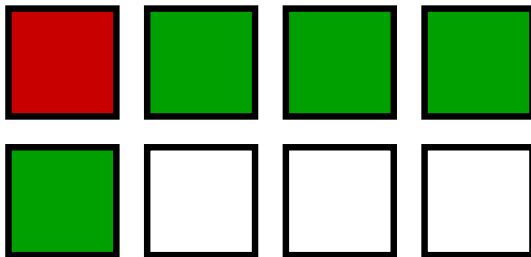




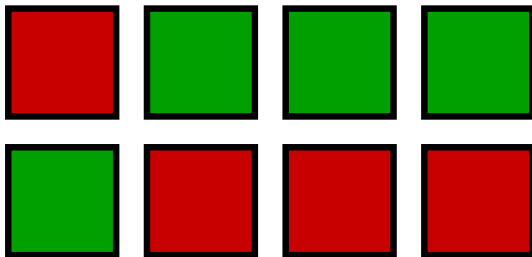
1. **Programm 1** beginnt Ausführung auf 8 Recheneinheiten



1. **Programm 1** beginnt Ausführung auf 8 Recheneinheiten
2. **Programm 1** sendet Ergebnisse über das Netzwerk



1. **Programm 1** beginnt Ausführung auf 8 Recheneinheiten
2. **Programm 1** sendet Ergebnisse über das Netzwerk
3. **Programm 2** beginnt Ausführung auf 4 Recheneinheiten



1. **Programm 1** beginnt Ausführung auf 8 Recheneinheiten
2. **Programm 1** sendet Ergebnisse über das Netzwerk
3. **Programm 2** beginnt Ausführung auf 4 Recheneinheiten
4. **Programm 1** führt wieder Berechnungen aus, jetzt auf 4 Recheneinheiten



- Ressourcenbewusstes Programmieren
- 3 Phasen:
  1. Invade - Ressourcen reservieren
  2. Infect - Ressourcen nutzen
  3. Retreat - Ressourcen freigeben
- OctoPOS und iRTSS bieten Software-Grundlage
- Angepasste Hardware um invasive Grundfunktionen zu unterstützen

- SPARC
  - Skalierbare Prozessorarchitektur
- SPARC-V8
  - Basierend auf SPARC
  - 32-bit Architektur
- LEON
  - SPARC-V8 basierte Prozessorfamilie
  - Konfigurierbare, Open-Source VHDL Designs
  - Geeignet für den Einsatz in angepassten ASICs, SOCS und FPGAs



- Sichere Speicherzugriffe
- Effiziente und sicherere Parallelberechnung
- Konzeptionell an C-artige Sprachen angelehnt
- Höhere Abstraktionen, um den Einstieg zu erleichtern
- Speichersicherheit und Abstraktionen sollen nicht auf Kosten der Leistung erreicht werden

## Das zentrale Alleinstellungsmerkmal der Programmiersprache

- Jeder Speicherbereich wird nur einer einzigen Variable zur Verfügung gestellt
- Beim Verlassen des Geltungsbereichs wird der Speicherbereich freigegeben

- Move-Semantik
  - Move
- Referenzen
  - Ref
  - mut Ref

Beispiel aus thesis, Colour-coded, step-by step

nostd JSON libcore Cargo

- Hilfsprogramm zum Kompilieren von invasiven Rust-Programmen
- Kompilierung von **.rs**-Dateien und Cargo-Projekten
- Unterstützung von SPARC-V8
- Automatische Verlinkung mit iRTSS/OctoPOS
- ca. 650 Zeilen Code (Python)

- Direkte C-Rust Bindings
- Rust-spezifische Änderungen
- ca. 750 Zeilen Code (Rust)

- Beispiel C Funktion zu Rust Funktion

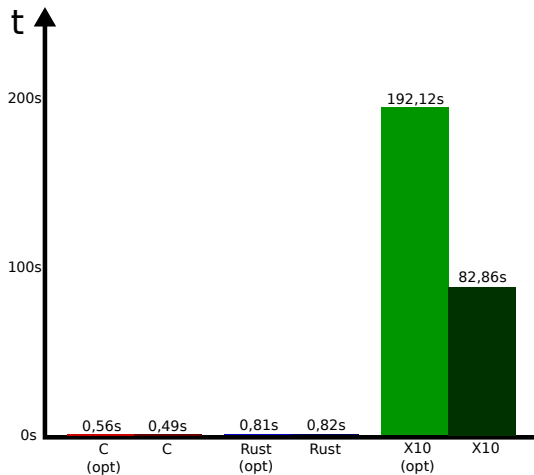


# octolib - Rust Improvements

- AgentClaim
- Closures

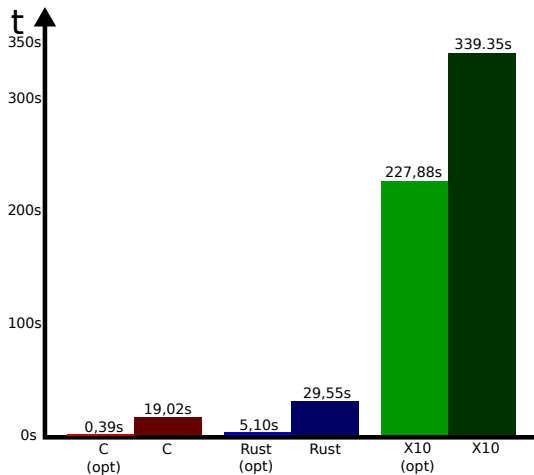
- temci
- 50-mal
- Intel i5...
- Benchmarks
  - Kompilierungsdauer
  - Parallele Primzahlenberechnung
  - Allokation von Objekten

# Kompilierungsdauer



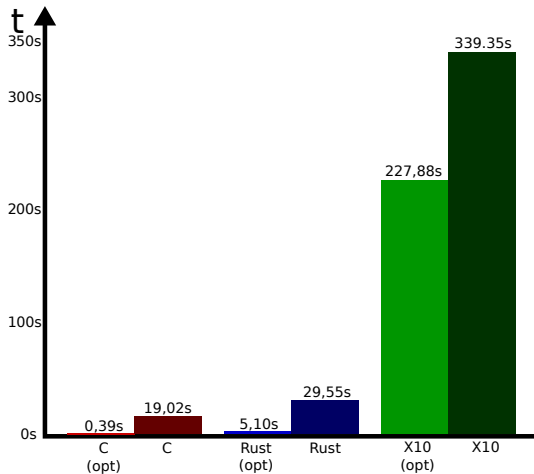
■ X10 langsam

# Parallele Primzahlenberechnung



■ Rust langsam

# Allokation von Objekten



■ X10 langsam

# Zusammenfassung und Fazit

4 Wichtigste Folien:  
invasive computing  
ownership - beispiel  
structs etc  
octolib - improvements