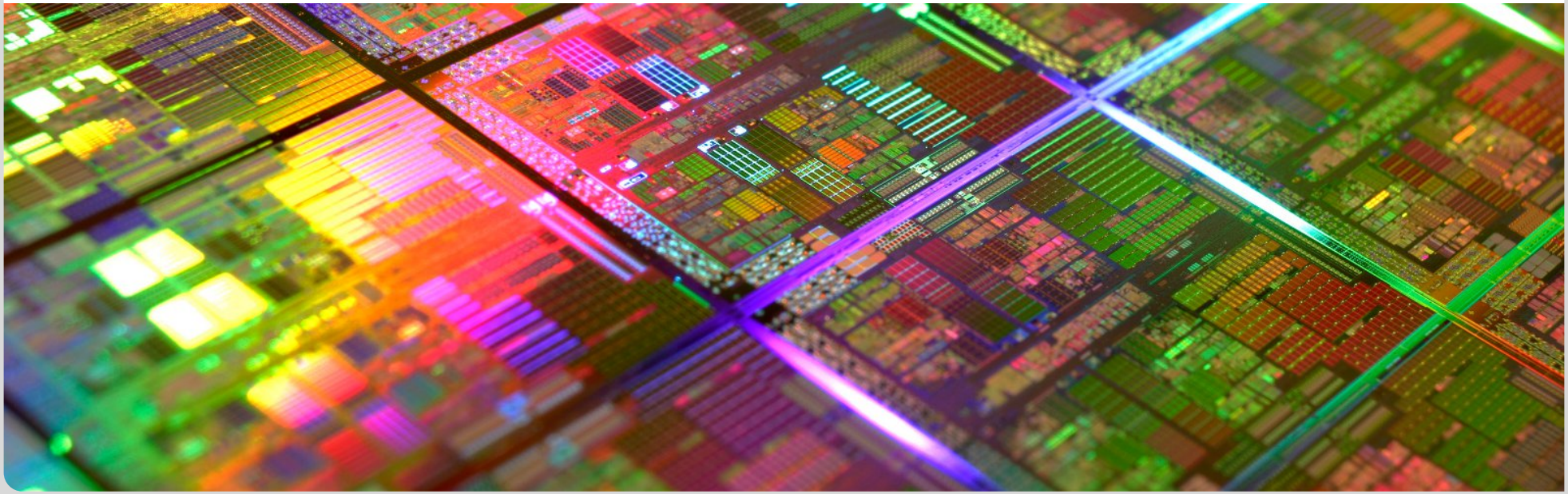
Karlsruher Institut für Technologie

Tutorium Rechnerorganisation

Woche 5

Tutorien 3 und 4 zur Vorlesung Rechnerorganisation



Heute

- MIMA-Interpreter
- Noch eine MIMA-Aufgabe...
- Einführung in MIPS
- Hello World, MIPS!

MIMA-Interpreter

- Download unter http://ti.itec.uka.de/Mima/MIMA_Interpreter.zip
- In C entwickelt und ohne Abhängigkeiten compilierbar und lauffähig
- Beispielprogramme sind im Paket enthalten
- Java-Applet zur Visualisierung von ADD, LDC, LDV, NOT und STV unter http://ti.itec.uka.de/Mima/Mima.php

Noch eine MIMA-Aufgabe...

- Aus Zeitgründen heute nur eine:
  - Primzahltest oder
  - Zahlenraten

Was ist MIPS?

- **M**icroprocessor without **I**nterlocked **P**ipeline **S**tages
- Prozessorarchitektur, die ab 1981 in Stanford entwickelt wurde
- Folgt der Designstrategie der RISC-Architektur (Reduced Instruction Set Computing)
  - Wenige, elementare Befehle; wenige Befehlsformate; viele Register
  - Festverdrahtetes Steuerwerk statt Mikroprogrammierung
  - Deshalb hohe Taktung möglich und einfaches Pipelining

Was ist MIPS?

- **M**icroprocessor without **I**nterlocked **P**ipeline **S**tages
- Prozessorarchitektur, die ab 1981 in Stanford entwickelt wurde
- Folgt der Designstrategie der RISC-Architektur (Reduced Instruction Set Computing)
  - Wenige, elementare Befehle; wenige Befehlsformate; viele Register
  - Festverdrahtetes Steuerwerk statt Mikroprogrammierung
  - Deshalb hohe Taktung möglich und einfaches Pipelining

SPIM

- SPIM := MIPS rückwärts
- Simulator, der eine MIPS32-CPU emuliert, und einen Assemblierer enthält
- Verfügbar für Linux, Windows und Mac
- Homepage: http://pages.cs.wisc.edu/~larus/spim.html

Befehlsformate von MIPS

- Alle Befehlsformate sind 32 Bit breit
- R-Typ (“Register”): 6 Bit Opcode, 3x 5 Bit Register-Nummer, 5 Bit Shift Amount, 6 Bit Funktions-Nummer
- I-Typ (“Immediate”): 6 Bit Opcode, 2x 5 Bit Register-Nummer, 16 Bit Immediate-Wert
- J-Typ (“Jump”): 6 Bit Opcode, 26 Bit Sprungziel

Befehlsformate von MIPS

- Alle Befehlsformate sind 32 Bit breit
- R-Typ (“Register”): 6 Bit Opcode, 3x 5 Bit Register-Nummer, 5 Bit Shift Amount, 6 Bit Funktions-Nummer
- I-Typ (“Immediate”): 6 Bit Opcode, 2x 5 Bit Register-Nummer, 16 Bit Immediate-Wert
- J-Typ (“Jump”): 6 Bit Opcode, 26 Bit Sprungziel

Befehlsformate von MIPS

- Alle Befehlsformate sind 32 Bit breit
- R-Typ (“Register”): 6 Bit Opcode, 3x 5 Bit Register-Nummer, 5 Bit Shift Amount, 6 Bit Funktions-Nummer
- I-Typ (“Immediate”): 6 Bit Opcode, 2x 5 Bit Register-Nummer, 16 Bit Immediate-Wert
- J-Typ (“Jump”): 6 Bit Opcode, 26 Bit Sprungziel

Befehlsformate von MIPS

- Alle Befehlsformate sind 32 Bit breit
- R-Typ (“Register”): 6 Bit Opcode, 3x 5 Bit Register-Nummer, 5 Bit Shift Amount, 6 Bit Funktions-Nummer
- I-Typ (“Immediate”): 6 Bit Opcode, 2x 5 Bit Register-Nummer, 16 Bit Immediate-Wert
- J-Typ (“Jump”): 6 Bit Opcode, 26 Bit Sprungziel

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Arithmetik)

- Grundsätzlich: “u” = unsigned, “i” = immediate
- add, addu, addi, addiu, sub, subu
- mult, div, divu
- and, andi, or, ori, xor, nor
- sll, srl, sra
- slt, slti

Einige MIPS-Befehle (Datentransfer)

- lb, lh, lw
- sb, sh, sw
- mfhi, mflo
- li (Pseudobefehl), lui
- mfc0, mtc0

Einige MIPS-Befehle (Datentransfer)

- lb, lh, lw
- sb, sh, sw
- mfhi, mflo
- li (Pseudobefehl), lui
- mfc0, mtc0

Einige MIPS-Befehle (Datentransfer)

- lb, lh, lw
- sb, sh, sw
- mfhi, mflo
- li (Pseudobefehl), lui
- mfc0, mtc0

Einige MIPS-Befehle (Datentransfer)

- lb, lh, lw
- sb, sh, sw
- mfhi, mflo
- li (Pseudobefehl), lui
- mfc0, mtc0

Einige MIPS-Befehle (Datentransfer)

- lb, lh, lw
- sb, sh, sw
- mfhi, mflo
- li (Pseudobefehl), lui
- mfc0, mtc0

Einige MIPS-Befehle (Kontrollfluss)

- j, jr, jal
- syscall
- beq, bne
- bgt, blt, bge, ble, bgtu, bgtz (alles Pseudobefehle)

Einige MIPS-Befehle (Kontrollfluss)

- j, jr, jal
- syscall
- beq, bne
- bgt, blt, bge, ble, bgtu, bgtz (alles Pseudobefehle)

Einige MIPS-Befehle (Kontrollfluss)

- j, jr, jal
- syscall
- beq, bne
- bgt, blt, bge, ble, bgtu, bgtz (alles Pseudobefehle)

Einige MIPS-Befehle (Kontrollfluss)

- j, jr, jal
- syscall
- beq, bne
- bgt, blt, bge, ble, bgtu, bgtz (alles Pseudobefehle)

Zusatzfunktionen eines MIPS-Assemblierers

- Die Pipeline kann nicht angehalten werden, also:
  - Einfügen von NOP-Befehlen in die Branch Delay Slots hinter Lade- und Sprungbefehlen oder
  - Umordnung der Befehlsreihenfolge
- Erweiterung des Befehlssatzes durch Pseudobefehle:
  - Zur Vereinfachung (Vergleichsbefehle) und
  - Zum Ausgleichen von Beschränkungen des Befehlssatzes (z.B. Immediate-Befehle mit Immediate-Wert mit mehr als 16 Bits)

Zusatzfunktionen eines MIPS-Assemblierers

- Die Pipeline kann nicht angehalten werden, also:
  - Einfügen von NOP-Befehlen in die Branch Delay Slots hinter Lade- und Sprungbefehlen oder
  - Umordnung der Befehlsreihenfolge
- Erweiterung des Befehlssatzes durch Pseudobefehle:
  - Zur Vereinfachung (Vergleichsbefehle) und
  - Zum Ausgleichen von Beschränkungen des Befehlssatzes (z.B. Immediate-Befehle mit Immediate-Wert mit mehr als 16 Bits)

Assemblerdirektiven

- Werden nicht (direkt) in Maschinencode umgesetzt
- Steuert das Verhalten des Assemblierers
  - Wahl des Segments
  - Reservierung von Speicher
  - Veränderung der Art, wie Daten abgelegt werden (z.B. Alignment)
- Beginnen in MIPS mit einem Punkt
- Wichtige Assemblerdirektiven: .data, .text, .globl, .ascii, .asciiz, .byte, .half, .word, .float, .double, .space, .align

Assemblerdirektiven

- Werden nicht (direkt) in Maschinencode umgesetzt
- Steuert das Verhalten des Assemblierers
  - Wahl des Segments
  - Reservierung von Speicher
  - Veränderung der Art, wie Daten abgelegt werden (z.B. Alignment)
- Beginnen in MIPS mit einem Punkt
- Wichtige Assemblerdirektiven: .data, .text, .globl, .ascii, .asciiz, .byte, .half, .word, .float, .double, .space, .align

Systemaufrufe

- Erlauben die Nutzung von vorgegebenen Betriebssystemfunktionen
- Details: Vorlesung Betriebssysteme (vormals Systemarchitektur)
- Werden in MIPS mit dem Befehl syscall ausgelöst
- Systemaufruf-Nummer muss vorher in das Register $v0 geschrieben werden
- Parameterübergabe ebenfalls über Register

Liste der Systemaufrufe

- Ausgabe von Werten:
  - print_int (#1), print_float (#2), print_double (#3), print_string (#4)
  - Parameter: Integer in $a0, Gleitkommazahl in/ab $f12, Startadresse des Strings in $a0
- Einlesen von Werten:
  - read_int (#5), read_float (#6), read_double (#7), read_string (#8)
  - Parameter: Rückgabe des Integers in $v0, Gleitkommazahl in/ab $f0, Startadresse des Stringpuffers in $a0, maximale Stringlänge in $a1
- Sonstiges:
  - sbrk (#9): Allokiert Speicherblock der Größe $a0 Bytes und schreibt die Startadresse in $v0
  - exit (#10): Beendet die Ausführung

Liste der Systemaufrufe

- Ausgabe von Werten:
  - print_int (#1), print_float (#2), print_double (#3), print_string (#4)
  - Parameter: Integer in $a0, Gleitkommazahl in/ab $f12, Startadresse des Strings in $a0
- Einlesen von Werten:
  - read_int (#5), read_float (#6), read_double (#7), read_string (#8)
  - Parameter: Rückgabe des Integers in $v0, Gleitkommazahl in/ab $f0, Startadresse des Stringpuffers in $a0, maximale Stringlänge in $a1
- Sonstiges:
  - sbrk (#9): Allokiert Speicherblock der Größe $a0 Bytes und schreibt die Startadresse in $v0
  - exit (#10): Beendet die Ausführung

Liste der Systemaufrufe

- Ausgabe von Werten:
  - print_int (#1), print_float (#2), print_double (#3), print_string (#4)
  - Parameter: Integer in $a0, Gleitkommazahl in/ab $f12, Startadresse des Strings in $a0
- Einlesen von Werten:
  - read_int (#5), read_float (#6), read_double (#7), read_string (#8)
  - Parameter: Rückgabe des Integers in $v0, Gleitkommazahl in/ab $f0, Startadresse des Stringpuffers in $a0, maximale Stringlänge in $a1
- Sonstiges:
  - sbrk (#9): Allokiert Speicherblock der Größe $a0 Bytes und schreibt die Startadresse in $v0
  - exit (#10): Beendet die Ausführung

Hello World, MIPS!

```
        .data
hello:  .asciiz "Hello World!\n"

        .text
        .globl main

main:   li $v0, 4
        la $a0, hello
        syscall
        jr $ra
```

Fertig!

Bitte bis zum nächsten Tutorium die MIPS-Befehlsreferenz (auf Homepage verlinkt) ausdrucken und mitbringen!