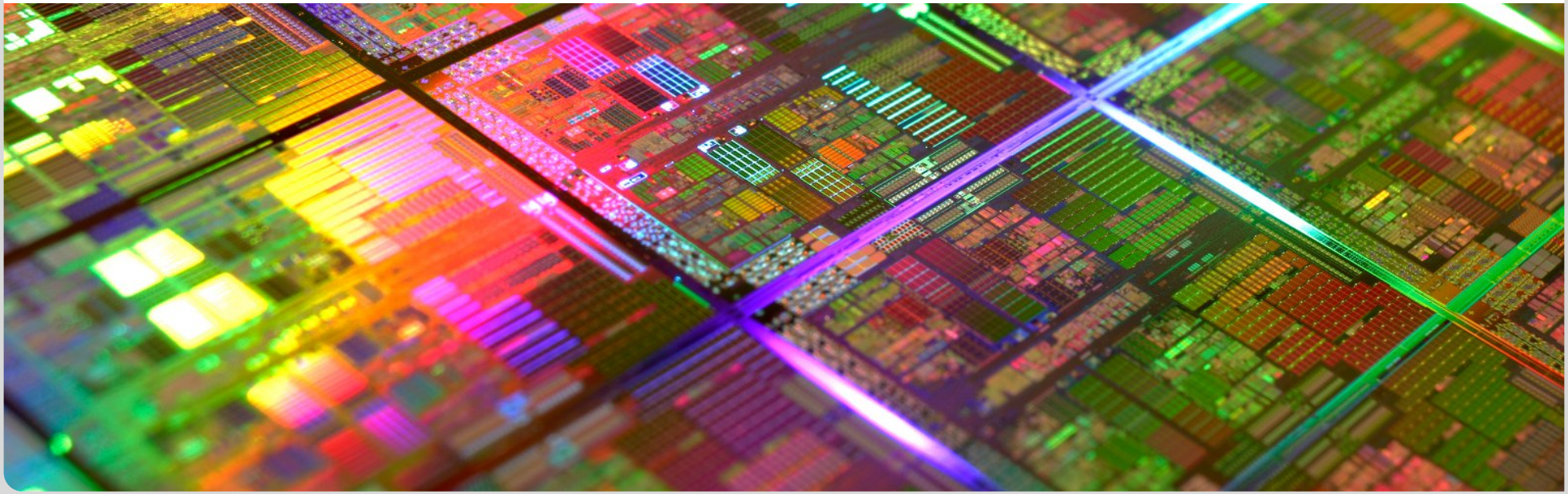
Karlsruher Institut für Technologie

Tutorium Rechnerorganisation

Woche 6

Tutorien 3 und 4 zur Vorlesung Rechnerorganisation



Heute

- Hello World, MIPS! (Wdh.)
- Übungsaufgaben

Hello World, MIPS! (Wdh.)

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

        .text
        .globl main

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

Aufgabe 1.1

```
            .data
x:          .word 3
y:          .word 1, 3, 7

            .text
subroutine: li $v0, 0
            li $t3, 0
marke1:     bge $t3, $a1, marke2
            lw $t0, 0($a0)
            mul $t1, $t0, $t0
            add $v0, $v0, $t1
            addi $a0, $a0, 4
            addi, $t3, $t3, 1
            b marke1
marke2:     jr $ra
```

```
# Fortsetzung von links
            .globl main
main:       la $a0, Y
            lw $a1, x
            jal subroutine

            move $a0, $v0
            li $v0, 1
            syscall

            li $v0, 10
            syscall
            jr $ra
```

Welche Funktion das Unterprogramm subroutine? Welche Ausgabe hat das gesamte Programm?

Aufgabe 1.2 - 1.4

- Geben Sie die MIPS-Instruktionen zu den folgenden Pseudoinstruktionen an:
  - b marke
  - neg $s3, $s2
- Was bewirkt die Assemblerdirektive .align 3?
- Warum dürfen bei Arithmetikoperationen mit doppelter Genauigkeit nur die Register mit gerader Registernummer verwendet werden?

Aufgabe 1.2 - 1.4

- Geben Sie die MIPS-Instruktionen zu den folgenden Pseudoinstruktionen an:
  - b marke
  - neg $s3, $s2
- Was bewirkt die Assemblerdirektive .align 3?
- Warum dürfen bei Arithmetikoperationen mit doppelter Genauigkeit nur die Register mit gerader Registernummer verwendet werden?

Aufgabe 1.2 - 1.4

- Geben Sie die MIPS-Instruktionen zu den folgenden Pseudoinstruktionen an:
  - b marke
  - neg $s3, $s2
- Was bewirkt die Assemblerdirektive .align 3?
- Warum dürfen bei Arithmetikoperationen mit doppelter Genauigkeit nur die Register mit gerader Registernummer verwendet werden?

Aufgabe 1.5 - 1.6

- Welche Adressen haben A, B, C und D im folgenden MIPS-Programmabschnitt?

```
   .data 0x10000003
   .align 3
A: .byte 6, 5
B: .word 7, 4
C: .double 3.1415
D: .float 2.71828
```

- Welche Gründe machen die Programmierung der MIPS-Architektur schwierig?

Aufgabe 1.5 - 1.6

- Welche Adressen haben A, B, C und D im folgenden MIPS-Programmabschnitt?

```
    .data 0x10000003
    .align 3
A:  .byte 6, 5
B:  .word 7, 4
C:  .double 3.1415
D:  .float 2.71828
```

- Welche Gründe machen die Programmierung der MIPS-Architektur schwierig?

Aufgabe 2.1

Schreiben Sie die folgenden Kontrollstrukturen in MIPS-Assembler um. Sie dürfen nur die MIPS-Befehle slt, beq und bne verwenden. Zur Speicherung temporärer Variablen verwenden Sie das Register $at. Die Variable a ist im Register $t4, die Variable b im Register $s0.

```
if ( a <= b ) { ... }
marke1:


if ( a >= b ) { ... }
marke2:


do {
  marke3: ...
          ...
} while ( a != b );
```

Aufgabe 2.1

Schreiben Sie die folgenden Kontrollstrukturen in MIPS-Assembler um. Sie dürfen nur die MIPS-Befehle slt, beq und bne verwenden. Zur Speicherung temporärer Variablen verwenden Sie das Register $at. Die Variable a ist im Register $t4, die Variable b im Register $s0.

```
if ( a <= b ) { ... }
marke1:


if ( a >= b ) { ... }
marke2:


do {
  marke3: ...
          ...
} while ( a != b );
```

Aufgabe 2.1

Schreiben Sie die folgenden Kontrollstrukturen in MIPS-Assembler um. Sie dürfen nur die MIPS-Befehle slt, beq und bne verwenden. Zur Speicherung temporärer Variablen verwenden Sie das Register $at. Die Variable a ist im Register $t4, die Variable b im Register $s0.

```
if ( a <= b ) { ... }
marke1:


if ( a >= b ) { ... }
marke2:


do {
  marke3: ...
          ...
} while ( a != b );
```

Aufgabe 2.2

Was sind die Unterschiede zwischen einer statischen Speicherallokierung und einer dynamischen Speicherallokierung?

Aufgabe 3.1a

Schreiben Sie die folgende C-Kontrollstruktur in MIPS-Assembler um.

```
a = b = c = 0;
if (i < 5) {
  a = 1;
  b = 2;
  c = 3;
}
d = 5;
```

Aufgabe 3.1b

Schreiben Sie die folgende C-Kontrollstruktur in MIPS-Assembler um.

```
a = b = c = 0;
if (i < 5) {
  a = 1;
  b = 2;
  c = 3;
} else {
  a = 4;
  b = 5;
  c = 6;
}
d = 5;
```

Aufgabe 3.1c

Schreiben Sie die folgende C-Kontrollstruktur in MIPS-Assembler um.

```
int a[100];
...
sum = 0;
for (i = 0; i < 100; i++)
  sum = sum + a[i];
```

Aufgabe 3.2 - 3.3

- Beschreiben Sie die Funktion der folgenden MIPS-Befehle:
  1. addu $t3, $t2, $t1
  2. andi $t3, $t2, 0x2000
  3. slt $t3, $t2, $t1
  4. lui $t3, 0x2000
- In welchem Register wird die Rücksprungadresse beim Unterprogrammaufruf gespeichert?

Aufgabe 4 (ohne 4.2)

- Geben Sie für das folgende MIPS-Programmstück den Inhalt des Zielregisters nach der Ausführung des jeweiligen Befehls in hexadezimaler Schreibweise an.

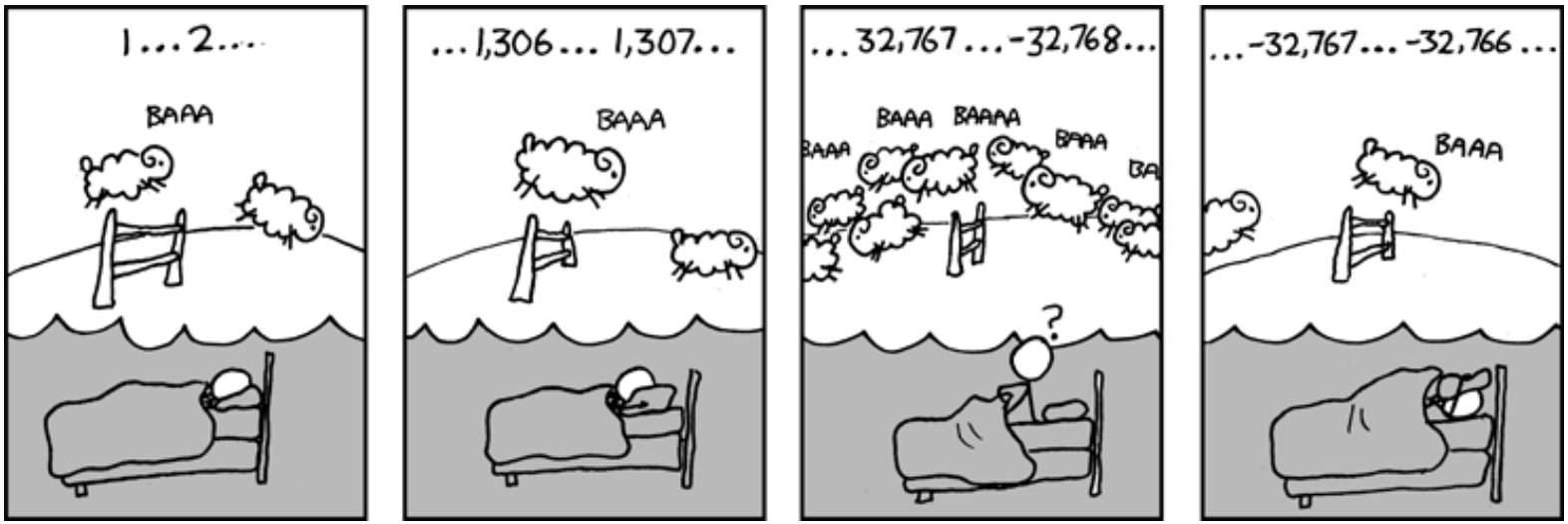
```
ori $s1, $zero, 20
sll $s2, $s1, 3
slti $s3, $s2, 100
sub $s4, $s3, $s2
lui $s5, -7
```

- Was ist ein Pseudobefehl? Was ist eine Assemblerdirektive?
- Wie ist die Trennung von Programmen und Daten bei der Ihnen bekannten MIPS-R2000-Architektur realisiert?

Organisatorisches

- Übungsblätter sind im Allgemeinen alleine zu bearbeiten und handgeschrieben abzugeben
- MIPS-Aufgaben:
  - dürfen elektronisch (Datenträger oder E-Mail) abgegeben werden
  - dürfen in Gruppen von bis zu drei Personen bearbeitet werden

Fertig!


1...2...
BAAA
...1,306... 1,307...
BAAA
...32,767...-32,768...
BAAA BAAAA
BAAA
BAAA
?
...-32,767... -32,766...
BAAA