Grundlagen der Rechnerarchitektur

Tim Luchterhand, Paul Nykiel (Abgabegruppe 117)

12. Februar 2019

1

(a)

Befehlsformat	Aufbau	Verwendung
Register/ALU-Operationen	ALU-Funktion (Bit 0-5), Verschiebung (Bit 6-10), Zielregister (Bit 11-15), Input Register 1 (Bit 16-20), Input Register 2 (Bit 21-25), OpCode (Bit 26-31)	Binäre Operationen mit zwei nicht konstanten Register-Operanden (z.B. add, sll)
Immediate Operationen	Direktoperand (Bit 0-15), Zielregister (Bit 16-20), Indexregister (Bit 21-25), OpCode (Bit 26-31)	Operationen mit Konstanten oder Adressen (z.B. lw, addiu)
Jump Befehle	Sprungziel (Bit 0-25), Opcode (Bit 26-31)	Sprungbefehle auf Basis von Adressen

(b) addiu \$t2, \$zero, -11 $\hat{=}$ 9 10 0 -11 $\hat{=}$ 001001 01010 00000 11111111 11110101

addi \$t1, \$t2, 53 $\hat{=}$ 8 9 10 53 $\hat{=}$ 001000 01001 01010 00000000 00110101

add \$s0, \$t1, \$zero $\hat{=}$ 0 0 9 16 0 20 $\hat{=}$ 000000 00000 01001 10000 00000 010100

(c) 000000 01011 10100 01010 00000 100110 $\hat{=}$ 0 11 20 10 0 38 $\hat{=}$ xor \$t2, \$s4, \$t3

000000 10010 10111 01001 00000 100010 $\hat{=}$ 0 18 23 10 0 34 $\hat{=}$ sub \$t2, \$s7, \$s2

001000 01001 10000 0001100001100101 $\hat{=}$ 8 9 16 6245 $\hat{=}$ addi \$t1, \$s0, 6245

000000 01001 01010 10000 00000 100000 $\hat{=}$ 0 9 10 16 0 32 $\hat{=}$ add \$t1, \$t2, \$s0

(d) Die Befehle unterscheiden sich im Format (Register und Immediate). Add wird genutzt, um zwei Register aufeinander zu addieren, wohingegen addi ein Register und eine Konstante addiert.

(e) „The Answer to the Ultimate Question of Life, the Universe, and Everything is 42“

2

(a)

```
1
2  ######## start own your own code ########
3  main:
4      li $t0, 53 # x = 53
5      li $t1, 2  # y = 2
6      li $t2, 6  # z = 6
7      mul $t3, $t1, $t2 # temp = y * z
8      sub $t3, $t0, $t3 # temp = x - temp
9      addi $t3, $t3, 1 # temp = temp + 1
10
11     move $a0, $t3 # print(temp)
12     li $v0, 1
13     syscall
14 ######## end own your own code #########
15
16     addi    $v0,$zero,10    #exit program
17     syscall
```

Aufgabe2a.asm

(b)

```
1  .data
2  str1:    .asciiz "Please enter your name and press Enter!\n\n"
3  str2:    .asciiz "\nHello "
4  str3:    .asciiz "An empty string is not a valid name!\n"
5  name:    .space  50
6
7  .text
8  main:    la  $a0,str1    #print str1
9      addi    $v0,$zero,4
10     syscall
11
12 ######### start of your own code ###########
13 label1:
14     addi    $v0,$zero,8 #read input
15     la  $a0,name
16     addi    $a1,$zero,50
17     syscall
18
19     lb $t0, 0($a0)
```

```
20      bne $t0, '\n', label2
21
22      la   $a0,str3     #print str3
23      addi     $v0,$zero,4
24      syscall
25
26      la   $a0,str2     #print str2
27      addi     $v0,$zero,4
28      syscall
29
30      j label1
31
32 ######### end of your own code ############
33
34 label2: la   $a0,str2     #print str2
35      addi     $v0,$zero,4
36      syscall
37      la   $a0,name     #print the entered name
38      syscall
39      j    end      #go to the programm end label
40
41 end:      addi     $v0,$zero,10     #exit the programm
42      syscall
```

Aufgabe2b.asm