Aufgabe 1)

or $t7, $t1, $t3
000000 $t7 (5bit) $t1 (5bit) $ t3 (5bit) 00000 (shamt) 100101 (funct)
R-Type

j 0x00406800
000010 00010000000110100000000000 (26 bit adress)
J-Type

andi $s4, $s2, 0xFAF0
001100 $s4 $ s2 1111101011110000 (16 bit immediate from hex to bin)
I-Type

srl $t3, $s3, 9
000000 00000 $t3 $s3 01001 000010
R-Type

Aufgabe 2:

128 Register:
Änderung an:
Instruktionsformat im R-Format: rs, rt, rd von 5 auf 7 Bit erweitert. shamt/funct schrumpfen um 6 Bit.
im I-Format: rs, rt von 5 auf 7 Bit erweitert. immediate schrumpft von 16 auf 12 Bit.
Länge einer Instruktion im R-Format: Bleibt 32 Bit, aber Felder innerhalb der Instruktion ändern sich.
im I-Format: Bleibt 32 Bit, aber Felder innerhalb der Instruktion ändern sich.
4x mehr mögliche Instruktionen:
Änderung an:
Instruktionsformat im R-Format: opcode von 6 auf 8 Bit erweitert. funct/shamt schrumpfen um 2 Bit.
im I-Format: opcode von 6 auf 8 Bit erweitert. immediate schrumpft von 16 auf 14 Bit.
Länge einer Instruktion im R-Format: Bleibt 32 Bit.
im I-Format: Bleibt 32 Bit.

Inwiefern kann ein Programm in MIPS-Assembler durch die genannten Änderungen kürzer werden?

- Mehr Register führen zu geringeren lw und sw Operationen.
- Komplexere Instruktionen ersetzen alte primitive Instruktionen.

Inwiefern kann es länger werden?

- Mehr Instruktionen nötig bei I-Type Instruktionen.

Aufgabe 3:

Schreiben Sie ein möglichst kurzes Assembler-Program, welches den Bereich zwischen dem 20. und 10. Bit (von rechts mit 1 beginnend gezählt) des Registers $a0 extrahiert und als Bits 28 bis 18 des Registers $v0 ablegt! Dabei sollen die übrigen Bits von $v0 mit 0 belegt sein. Testen Sie ihre Implementierung, indem Sie zuerst 0xFAAB1014 in das Register $a0 laden und geben Sie diese Version als einzelne Datei im Moodle ab!

Aufgabe 5:

1) swap
2) sll
3) $a1
4) $t1
5) $a0
6) lw
7) $t1
8) jr
9) $ra
10) sw