ERA - Übungsblatt 02

2. siehe Mitschrift WS 23/24

3. a) Da die meisten Register-Immediate-Instruktionen nur 12 Bit signed Immediates unterstützen, existiert die Instruktion 'lui' (load upper immediate), um die verbleibenden oberen 20 Bit eines Registers zu beschreiben.
Bezüglich der 12 Bit-Immediates: Instruktionen werden in 32 Bit lange Maschinenwörter assembliert. Dabei müssen auch Opcode, Register etc. enkodiert werden, daher stehen nicht die vollen 32 Bit zur Enkodierung von Immediates zur Verfügung, sondern nur 12 Bit.
Merke: 12 Bit Immediates werden auf 32 Bit sign-extended!

lui setzt die unteren 12 Bit auf '0', zum Laden von 32-Bit-Konstanten muss also immer lui vor addi kommen!

b) siehe Mitschrift WS 23/24

4. a) Wie aus Woche 01 bekannt, können Divisionen mit einer Zweierpotenz effizient mit Shifts gelöst werden. Mit $16 = 2^4$ also:
srli a0, a0, 4

b) x mod a entspricht dem Rest bei der Ganzzahldivision von x durch a. Das Ergebnis muss entsprechen in $[0, a-1]$ liegen, da ansonsten noch ein Vielfaches von a enthalten ist. Für Zweierpotenzen lässt sich der Rest durch eine Verundung mit $(a-1)$ "extrahieren". Also $a_0 := a_0 \bmod 256$:
andi a0, a0, 255 ≡ andi a0, a0, 0xff
Beispiel:

```
  1 0 0 1 1 1 0 1 1 0 0 0
& 0 0 0 0 1 1 1 1 1 1 1 1      (0xff)
-----------------------
  0 0 0 0 1 1 0 1 1 0 0 0
          |---------------|
          Rest bei Division mit 256
```

c)
```
slli a1, a1, 16      a1: xxx...xxx 000...000   (16 Bit | 16 Bit)
srli a1, a1, 16      a1: 000...000 xxx...xxx   (16 Bit | 16 Bit)
slli a2, a2, 16      a2: yyy...yyy 000...000   (16 Bit | 16 Bit)
or   a0, a1, a2      a0: yyy...yyy xxx...xxx
```

d, e) Siehe Musterlösung

Verwendung logischer Operatoren:
- Verundung mit '0' setzt Bits garantiert auf '0'.
- Veroderung mit '1' setzt Bits garantiert auf '1'.
- XOR mit '1' flippt/invertiert Bits.

```
  1 0 1 1 1 0 1
& 1 1 1 0 0 0 0
-------------
  1 0 1 0 0 0 0
```

```
  1 0 1 1 1 0 1
| 0 0 0 1 1 1 1
-------------
  1 0 1 1 1 1 1
```

```
  1 0 1 1 1 0 1
⊕ 0 0 0 1 1 1 1
-------------
  1 0 1 0 0 1 0
```

5. siehe ML und Mitschrift WS 23/24