1. **Addiere zwei Zahlen**

```
--- simple RISCV emulator ---
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 1
name of .txt file in /test: 01
enable debug-mode? y/n: y
current line: ['ADDI', 'X1,', 'X0,', '5']
opcode: ADDI, operands: ['1', '0', '5']
labels: {'TARGET': 2}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 5      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// 5 wurde korrekt in Register x01 geschrieben

```
current line: ['ADDI', 'X2,', 'X0,', '10']
opcode: ADDI, operands: ['2', '0', '10']
labels: {'TARGET': 2}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 5      x02: 10      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADD', 'X3,', 'X1,', 'X2']
```

```
opcode: ADD, operands: ['3', '1', '2']
labels: {'TARGET': 2}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 5      x02: 10      x03: 15
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// 5+10 = 15, korrektes Ergebnis in x03

2. Speicherzugriff

```
--- simple RISCV emulator ---
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 1
name of .txt file in /test: 02
enable debug-mode? y/n: y
current line: ['ADDI', 'X1,', 'X0,', '42']
opcode: ADDI, operands: ['1', '0', '42']
labels: {}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 42      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X2,', 'X0,', '100']
opcode: ADDI, operands: ['2', '0', '100']
```

```
labels: {}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 42     x02: 100        x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['SW', 'X1,', '0', 'X2']
opcode: SW, operands: ['1', '0', '2']
labels: {}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 42     x02: 100        x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['LW', 'X3,', '0', 'X2']
opcode: LW, operands: ['3', '0', '2']
labels: {}
(next) instruction count: 4
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 42     x02: 100        x03: 42
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// 42 wurde korrekt aus Speicher gelesen, Memory Dump wird später noch gezeigt

```
--- simple RISCV emulator ---
```

```
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 3
start address: 80
end address: 150
Memory dump [0x00000050-0x00000096]:

0x00000050: 00000000 00000000 00000000 00000000
0x00000060: 00000000 0000002a 00000000 00000000
0x00000070: 00000000 00000000 00000000 00000000
0x00000080: 00000000 00000000 00000000 00000000
0x00000090: 00000000 00000000
```

// 42 in hex == 2A

3. Bedingte Verzweigung

```
-- simple RISCV emulator ---
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 1
name of .txt file in /test: 03
enable debug-mode? y/n: y
current line: ['ADDI', 'X1,', 'X0,', '7']
opcode: ADDI, operands: ['1', '0', '7']
labels: {'EQUAL': 4}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 7      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
```

```
x20: 0     x21: 0     x22: 0     x23: 0
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0

current line: ['ADDI', 'X2,', 'X0,', '7']
opcode: ADDI, operands: ['2', '0', '7']
labels: {'EQUAL': 4}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0     x01: 7     x02: 7     x03: 0
x04: 0     x05: 0     x06: 0     x07: 0
x08: 0     x09: 0     x10: 0     x11: 0
x12: 0     x13: 0     x14: 0     x15: 0
x16: 0     x17: 0     x18: 0     x19: 0
x20: 0     x21: 0     x22: 0     x23: 0
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0

current line: ['BEQ', 'X1,', 'X2,', 'EQUAL']
opcode: BEQ, operands: ['1', '2', 'EQUAL']
labels: {'EQUAL': 4}
(next) instruction count: 4
--- CURRENT REGISTER CONTENTS ---
x00: 0     x01: 7     x02: 7     x03: 0
x04: 0     x05: 0     x06: 0     x07: 0
x08: 0     x09: 0     x10: 0     x11: 0
x12: 0     x13: 0     x14: 0     x15: 0
x16: 0     x17: 0     x18: 0     x19: 0
x20: 0     x21: 0     x22: 0     x23: 0
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0
```

// x1 und x2 sind gleich -> ic wird auf Wert Label ‘EQUAL’ gesetzt. Ic = 4 statt 3

```
current line: ['EQUAL:']
opcode: EQUAL:, operands: []
labels: {'EQUAL': 4}
(next) instruction count: 5
--- CURRENT REGISTER CONTENTS ---
x00: 0     x01: 7     x02: 7     x03: 0
x04: 0     x05: 0     x06: 0     x07: 0
x08: 0     x09: 0     x10: 0     x11: 0
x12: 0     x13: 0     x14: 0     x15: 0
x16: 0     x17: 0     x18: 0     x19: 0
```

```
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X3,', 'X0,', '99']
opcode: ADDI, operands: ['3', '0', '99']
labels: {'EQUAL': 4}
(next) instruction count: 6
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 7      x02: 7      x03: 99
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// Sprung wurde korrekt ausgeführt, x03 hatte nie den Wert 1

4. Unbedingter Sprung

```
--- simple RISCV emulator ---
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 1
name of .txt file in /test: 04
enable debug-mode? y/n: y
current line: ['JAL', 'X0,', 'TARGET']
opcode: JAL, operands: ['0', 'TARGET']
labels: {'TARGET': 2}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 0      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
```

```
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0

// IC 2 statt 1 == TARGET idx
// x0 bleibt 0

current line: ['TARGET:']
opcode: TARGET:, operands: []
labels: {'TARGET': 2}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0     x01: 0     x02: 0     x03: 0
x04: 0     x05: 0     x06: 0     x07: 0
x08: 0     x09: 0     x10: 0     x11: 0
x12: 0     x13: 0     x14: 0     x15: 0
x16: 0     x17: 0     x18: 0     x19: 0
x20: 0     x21: 0     x22: 0     x23: 0
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0

current line: ['ADDI', 'X2,', 'X0,', '2']
opcode: ADDI, operands: ['2', '0', '2']
labels: {'TARGET': 2}
(next) instruction count: 4
--- CURRENT REGISTER CONTENTS ---
x00: 0     x01: 0     x02: 2     x03: 0
x04: 0     x05: 0     x06: 0     x07: 0
x08: 0     x09: 0     x10: 0     x11: 0
x12: 0     x13: 0     x14: 0     x15: 0
x16: 0     x17: 0     x18: 0     x19: 0
x20: 0     x21: 0     x22: 0     x23: 0
x24: 0     x25: 0     x26: 0     x27: 0
x28: 0     x29: 0     x30: 0     x31: 0
```

5. Schleife

```
select an option: 1
name of .txt file in /test: 05
enable debug-mode? y/n: y
current line: ['ADDI', 'X1,', 'X0,', '5']
opcode: ADDI, operands: ['1', '0', '5']
labels: {'LOOP': 1}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
```

```
x00: 0      x01: 5      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['LOOP:']
opcode: LOOP:, operands: []
labels: {'LOOP': 1}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 5      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X1,', 'X1,', '-1']
opcode: ADDI, operands: ['1', '1', '-1']
labels: {'LOOP': 1}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 4      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['BNE', 'X1,', 'X0,', 'LOOP']
opcode: BNE, operands: ['1', '0', 'LOOP']
labels: {'LOOP': 1}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 4      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
```

```
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

// IC auf 1 -> Schleife wird wiederholt

current line: ['LOOP:']
opcode: LOOP:, operands: []
labels: {'LOOP': 1}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 4      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X1,', 'X1,', '-1']
opcode: ADDI, operands: ['1', '1', '-1']
labels: {'LOOP': 1}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 3      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['BNE', 'X1,', 'X0,', 'LOOP']
opcode: BNE, operands: ['1', '0', 'LOOP']
labels: {'LOOP': 1}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 3      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
```

```
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// IC auf 1 -> Schleife wird wiederholt

```
current line: ['LOOP:']
opcode: LOOP:, operands: []
labels: {'LOOP': 1}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 3      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X1,', 'X1,', '-1']
opcode: ADDI, operands: ['1', '1', '-1']
labels: {'LOOP': 1}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 2      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['BNE', 'X1,', 'X0,', 'LOOP']
opcode: BNE, operands: ['1', '0', 'LOOP']
labels: {'LOOP': 1}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 2      x02: 0      x03: 0
```

```
x04: 0       x05: 0       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0

// IC auf 1 -> Schleife wird wiederholt

current line: ['LOOP:']
opcode: LOOP:, operands: []
labels: {'LOOP': 1}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 2       x02: 0       x03: 0
x04: 0       x05: 0       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0

current line: ['ADDI', 'X1,', 'X1,', '-1']
opcode: ADDI, operands: ['1', '1', '-1']
labels: {'LOOP': 1}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 1       x02: 0       x03: 0
x04: 0       x05: 0       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0

current line: ['BNE', 'X1,', 'X0,', 'LOOP']
opcode: BNE, operands: ['1', '0', 'LOOP']
labels: {'LOOP': 1}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 1       x02: 0       x03: 0
```

```
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

// IC auf 1 -> Schleife wird wiederholt

current line: ['LOOP:']
opcode: LOOP:, operands: []
labels: {'LOOP': 1}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 1      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['ADDI', 'X1,', 'X1,', '-1']
opcode: ADDI, operands: ['1', '1', '-1']
labels: {'LOOP': 1}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 0      x02: 0      x03: 0
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

current line: ['BNE', 'X1,', 'X0,', 'LOOP']
opcode: BNE, operands: ['1', '0', 'LOOP']
labels: {'LOOP': 1}
(next) instruction count: 4
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 0      x02: 0      x03: 0
```

```
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// BNE nicht erfüllt -> Schleifenende

6. Zahlen summieren

Ich denke branches und jumps sind damit klar; Für die nächsten Beispiele werden (aus Platz- und Übersichtsgründen) nur noch Lösungen kommentiert

// Register nach Ausführung

```
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 10      x02: 10      x03: 55
x04: 0      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// 1 bis 10 aufsummiert == 55 in x03 -> alles hat funktioniert

7. Fibonacci

// Register nach Ausführung

```
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 7      x02: 8      x03: 13
x04: 13      x05: 7      x06: 0      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// x04 enthält richtigen Wert

8. Maximum in Array finden

// zunächst wurde ‘prep08’ ausgeführt, um RAM in angegebenen Bereich mit Zahlen zu // // füllen.

```
// Werte in RAM. maximum = 8
start address: 80
end address: 200
Memory dump [0x00000050-0x000000c8]:

0x00000050: 00000000 00000000 00000000 00000000
0x00000060: 00000000 00000005 00000008 00000003
0x00000070: 00000005 00000001 00000000 00000000
0x00000080: 00000000 00000000 00000000 00000000
0x00000090: 00000000 00000000 00000000 00000000
0x000000a0: 00000000 00000000 00000000 00000000
0x000000b0: 00000000 00000000 00000000 00000000
0x000000c0: 00000000 00000000

// Register nach Ausführung von ‘08’
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 100        x02: 3      x03: 8
x04: 104        x05: 8      x06: 1      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0

// maximum (8) steht in x03 -> korrekte Ausführung
```

9. **Funktionsaufruf mit jal / jalr**

```
// hier wieder komplette Debug Ausgabe

--- simple RISCV emulator ---
1: read .txt file with Assembler Code
2: input Assembler Code oneliner
3: Memory Dump
4: print registers
5: reset registers
6: reset RAM
X: EXIT

select an option: 1
name of .txt file in /test: 09
enable debug-mode? y/n: y
current line: ['ADDI', 'X1,', 'X0,', '21']
opcode: ADDI, operands: ['1', '0', '21']
```

```
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 1
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 21      x02: 0       x03: 0
x04: 0       x05: 0       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0

current line: ['JAL', 'X5,', 'DOUBLE']
opcode: JAL, operands: ['5', 'DOUBLE']
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 3
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 21      x02: 0       x03: 0
x04: 0       x05: 2       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0
```

// in x05 wird die Rücksprungadresse vermerkt = current ic(1) + 1 = 2
// ic wird auf wert von double gesetzt (3)

```
current line: ['DOUBLE:']
opcode: DOUBLE:, operands: []
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 4
--- CURRENT REGISTER CONTENTS ---
x00: 0       x01: 21      x02: 0       x03: 0
x04: 0       x05: 2       x06: 0       x07: 0
x08: 0       x09: 0       x10: 0       x11: 0
x12: 0       x13: 0       x14: 0       x15: 0
x16: 0       x17: 0       x18: 0       x19: 0
x20: 0       x21: 0       x22: 0       x23: 0
x24: 0       x25: 0       x26: 0       x27: 0
x28: 0       x29: 0       x30: 0       x31: 0

current line: ['SLLI', 'X6,', 'X1,', '1']
```

```
opcode: SLLI, operands: ['6', '1', '1']
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 5
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 21     x02: 0      x03: 0
x04: 0      x05: 2      x06: 42     x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// der doppelte wert von x01 in x06

```
current line: ['JALR', 'X0,', '0', 'X5']
opcode: JALR, operands: ['0', '0', '5']
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 2
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 21     x02: 0      x03: 0
x04: 0      x05: 2      x06: 42     x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// Rücksprung zu x05 -> ic auf 2 gesetzt.

```
current line: ['J', 'END']
opcode: J, operands: ['END']
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 6
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 21     x02: 0      x03: 0
x04: 0      x05: 2      x06: 42     x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

```
current line: ['END:']
opcode: END:, operands: []
labels: {'DOUBLE': 3, 'END': 6}
(next) instruction count: 7
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 21      x02: 0      x03: 0
x04: 0      x05: 2      x06: 42      x07: 0
x08: 0      x09: 0      x10: 0      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// j zu END -> Programm beendet.

10. Eigenes Beispiel

// Ein eigenes Beispiel: ‘branching.txt’ in ‘./test’
// “umgekehrter” Fall zu 4.
// beq soll nicht ausgeführt werden,
// 42 soll in x10 geschrieben werden, x03 soll 0 bleiben

```
--- CURRENT REGISTER CONTENTS ---
x00: 0      x01: 7      x02: 0      x03: 0
x04: 9      x05: 0      x06: 0      x07: 0
x08: 0      x09: 0      x10: 42      x11: 0
x12: 0      x13: 0      x14: 0      x15: 0
x16: 0      x17: 0      x18: 0      x19: 0
x20: 0      x21: 0      x22: 0      x23: 0
x24: 0      x25: 0      x26: 0      x27: 0
x28: 0      x29: 0      x30: 0      x31: 0
```

// korrekt ausgeführt