

Implementierung eines Mikrorechners in VHDL auf einem FPGA

Von Markus Schneider

Educational Processing Unit

EPU

Von Markus Schneider

Gliederung

- **Beschreibung des Projektes**
- **Hardwareübersicht**
- **Umsetzung in VHDL**
- **EPU-Assembly**
- **Befehlsaufbau**
- **Funktionsweise**
- **Demo**

Beschreibung des Projektes

- **Ziel:**

- Entwurf eines funktionsfähigen Mikrorechners
- Von Grund auf selbst erstellt
- Umsetzung des Entwurfes in die Realität

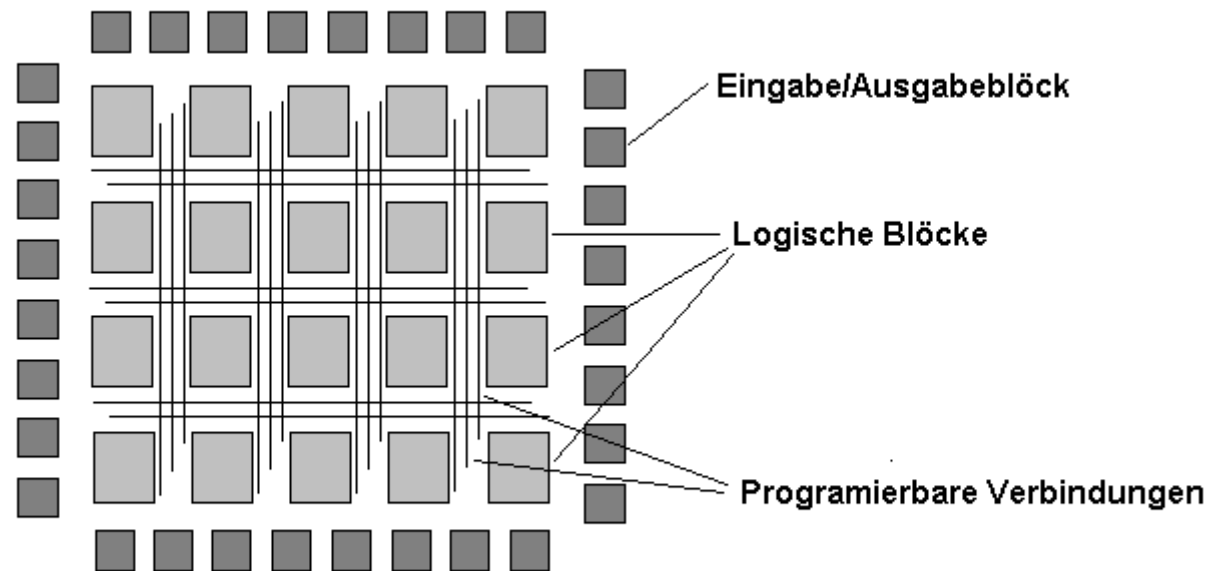
- **Umsetzung:**

- Hardware ist auf einem FPGA implementiert
- Hardwarebeschreibungssprache **VHDL** zum Beschreiben der Hardware

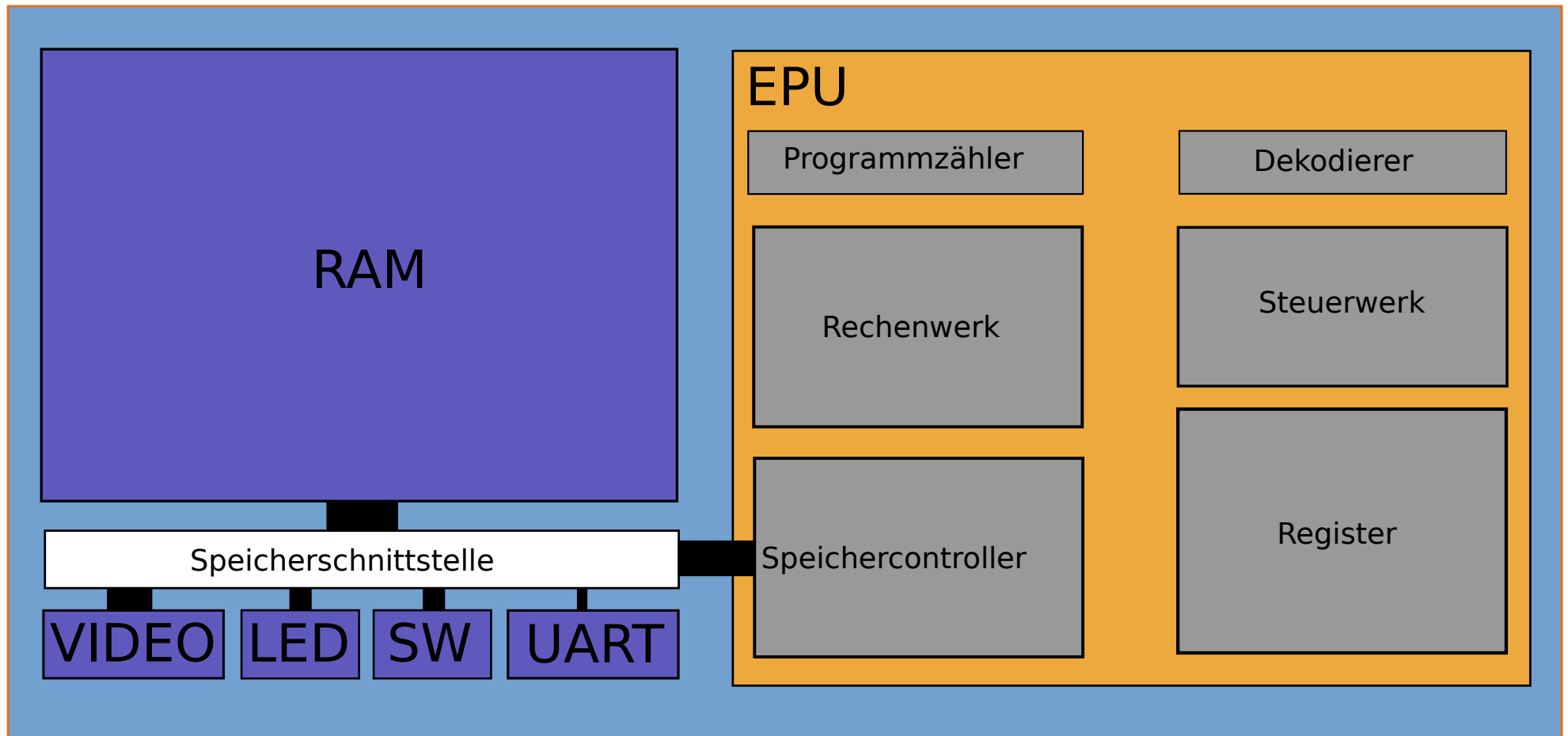
Was ist ein FPGA?

- **“Field Programmable Gate Array”**
- **Integrierter Schaltkreis**
- **Besteht aus Funktionsblöcken**
- **Funktionsblöcke können beliebig zusammengefügt**
- **Durch Programmierung entsteht die gewünschte Schaltung**

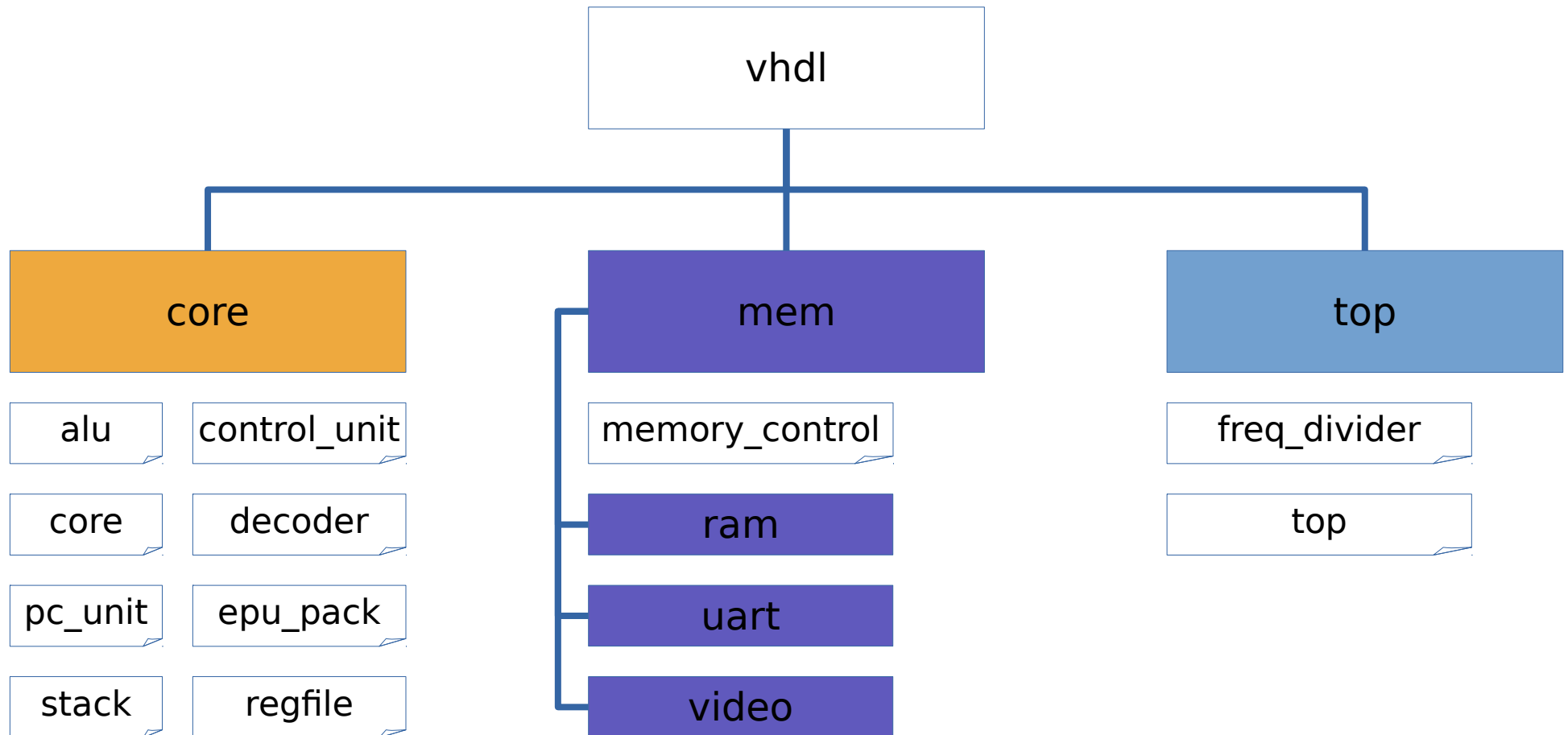
Was ist ein FPGA?



Hardwareübersicht



Umsetzung in VHDL



Umsetzung in VHDL

Deklaration des 'core'-Moduls

[...]

```
PCOp <= PC_OP_RESET when I_CORE_Reset = '1' else
```

```
    PC_OP_ASSIGN when SB = '1' and En_RegWrite = '1' else
```

```
    PC_OP_INC when En_Decode = '1' and DecDone = '0' else
```

```
    PC_OP_NOP;
```

```
PCIn <= StackData when En_RegWrite = '1'
```

```
    and AluOp(IFO_REL_OPCODE_BEGIN downto IFO_REL_OPCODE_END) = OPCODE_RET
```

```
    else ADDR_INT when En_RegWrite = '1'
```

```
    and AluOp(IFO_REL_OPCODE_BEGIN downto IFO_REL_OPCODE_END) = OPCODE_INT
```

```
    else Res;
```

```
uut_pc_unit : pc_unit port map (
```

```
    I_Clk => I_CORE_Clk,
```

```
    I_Op => PCOp,
```

```
    I_PC => PCIn,
```

```
    O_PC => PC
```

```
);
```

```
end behav_core;
```

EPU-Assembly

Befehlsaufbau

Mnemonic [.Option] [Operand 1][, Operand 2][, Operand 3]

Beispiele

add.u r2, r3, r4

$R2 = R3 + R4$

call.i \$test

Rufe Funktion mit dem Label **test** auf.

load r3, 0xDEAD

$R3 = 0xDEAD$

EPU-Assembly

Assembly

start:

load r4, 0x41

load r5, 0x90

cmp r4, r5

jge.i \$start



Maschinencode

0000: 3B40 0041

0004: 3B50 0090

0008: 6AE4 50

000B: 7F4E 0000

Befehlsaufbau

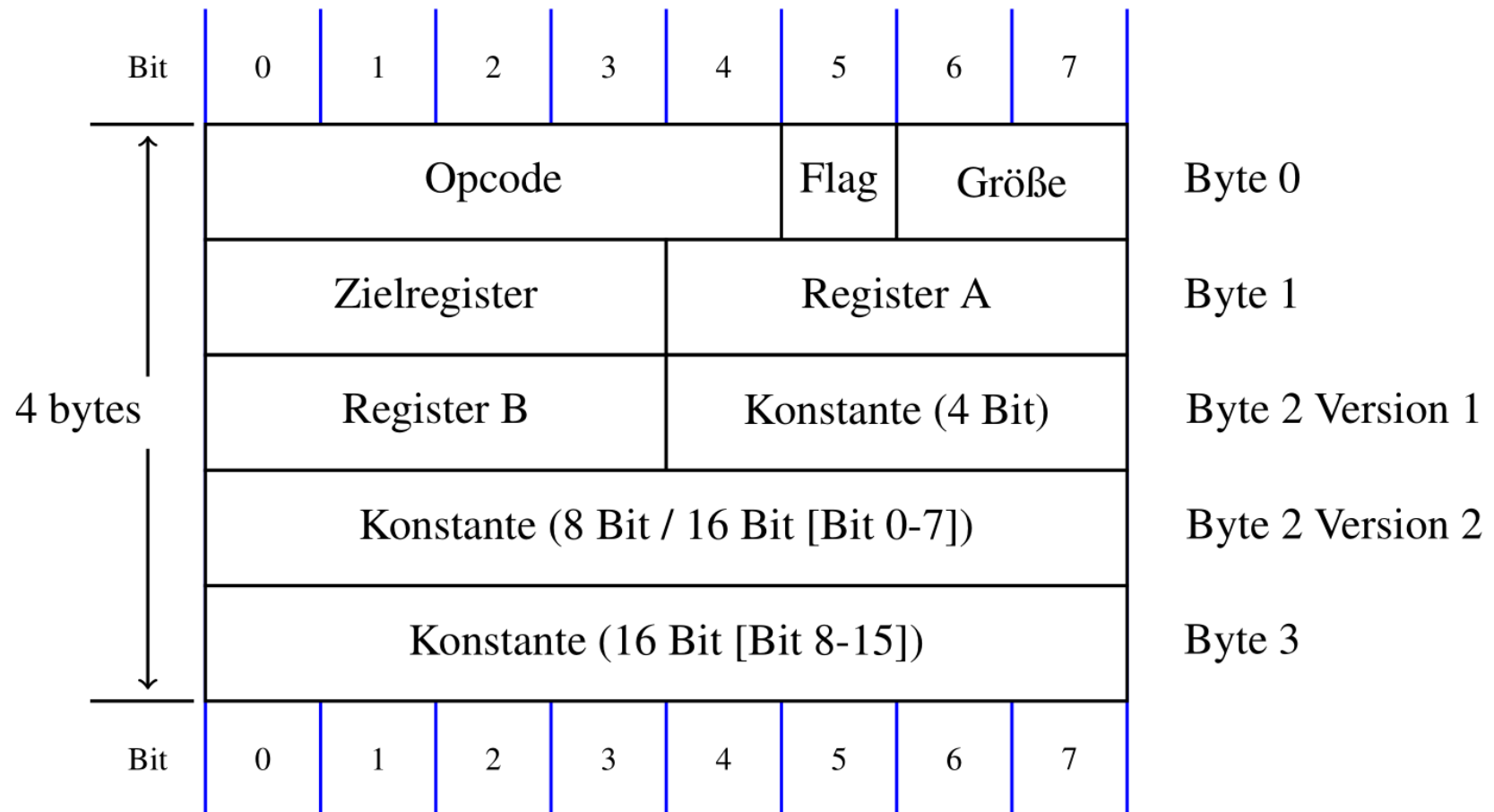
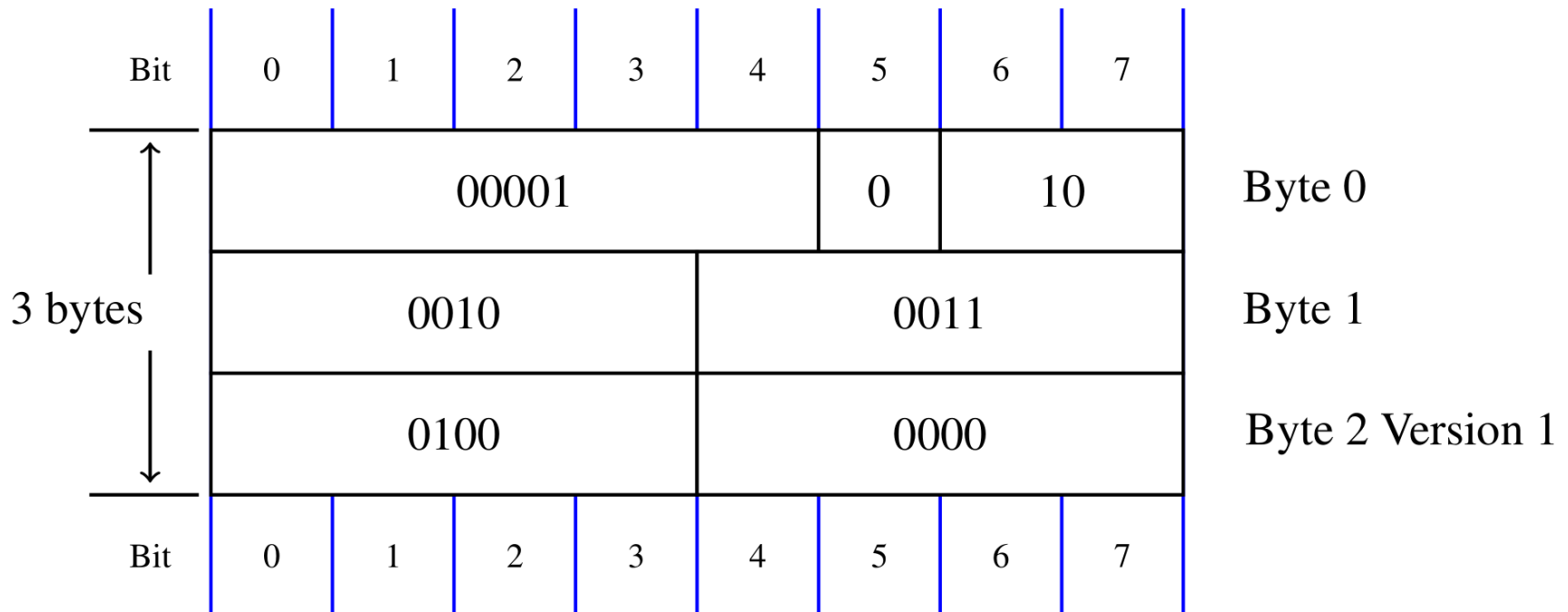


Abbildung 2.3.: Befehlsaufbau

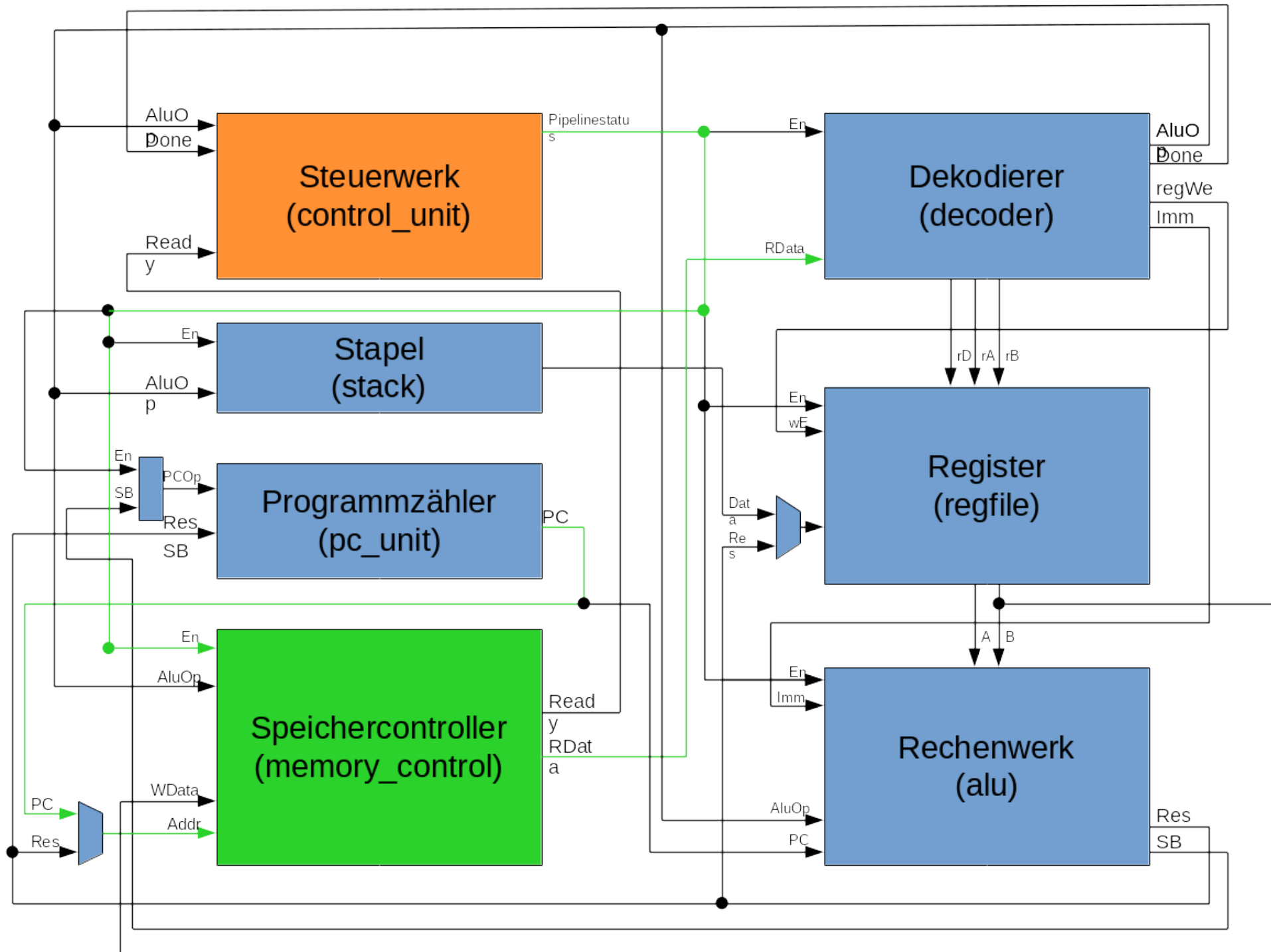
Befehlsaufbau

add.u r2, r3, r4

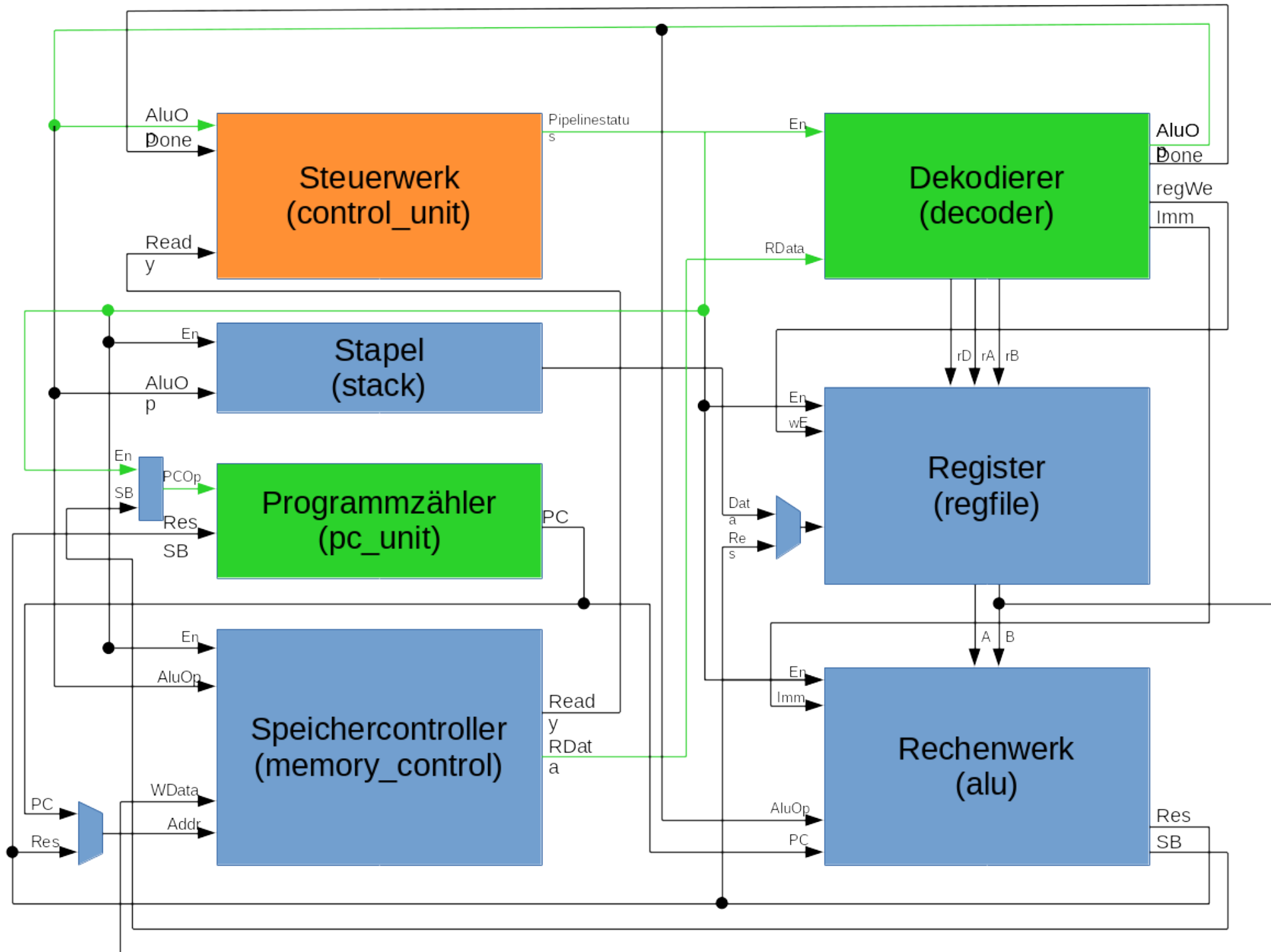


Funktionsweise
add.u r2, r3, r4

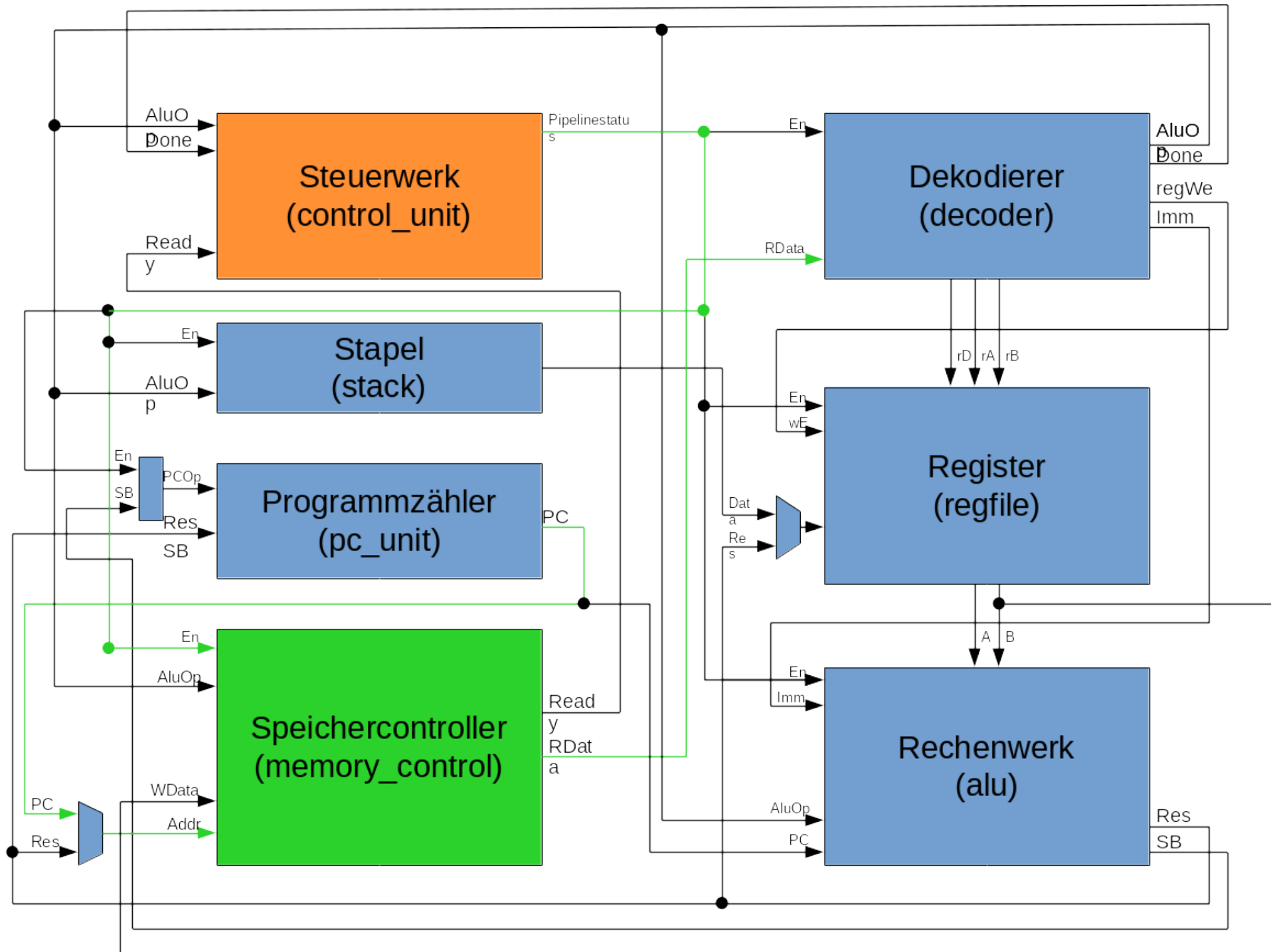
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



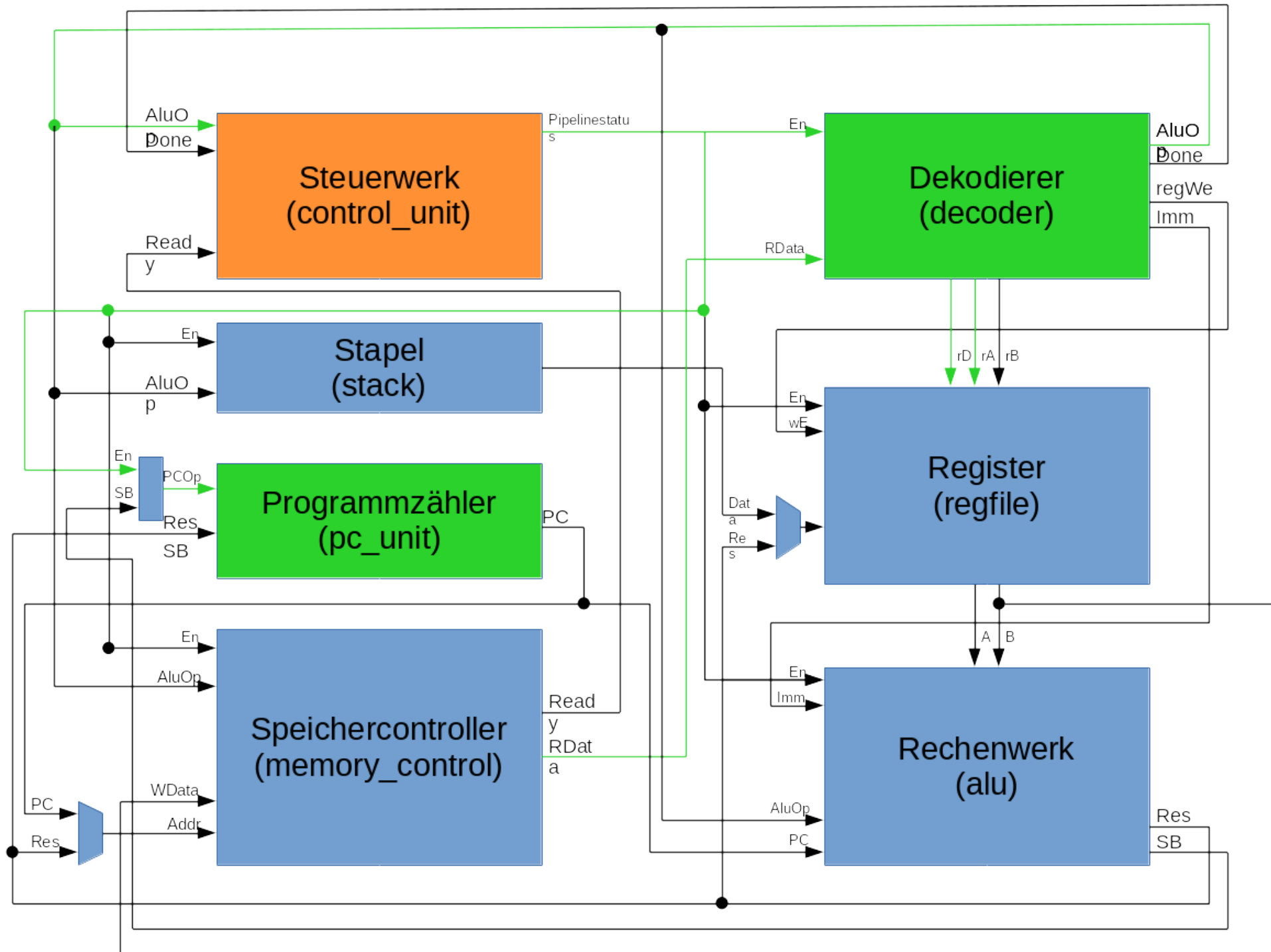
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



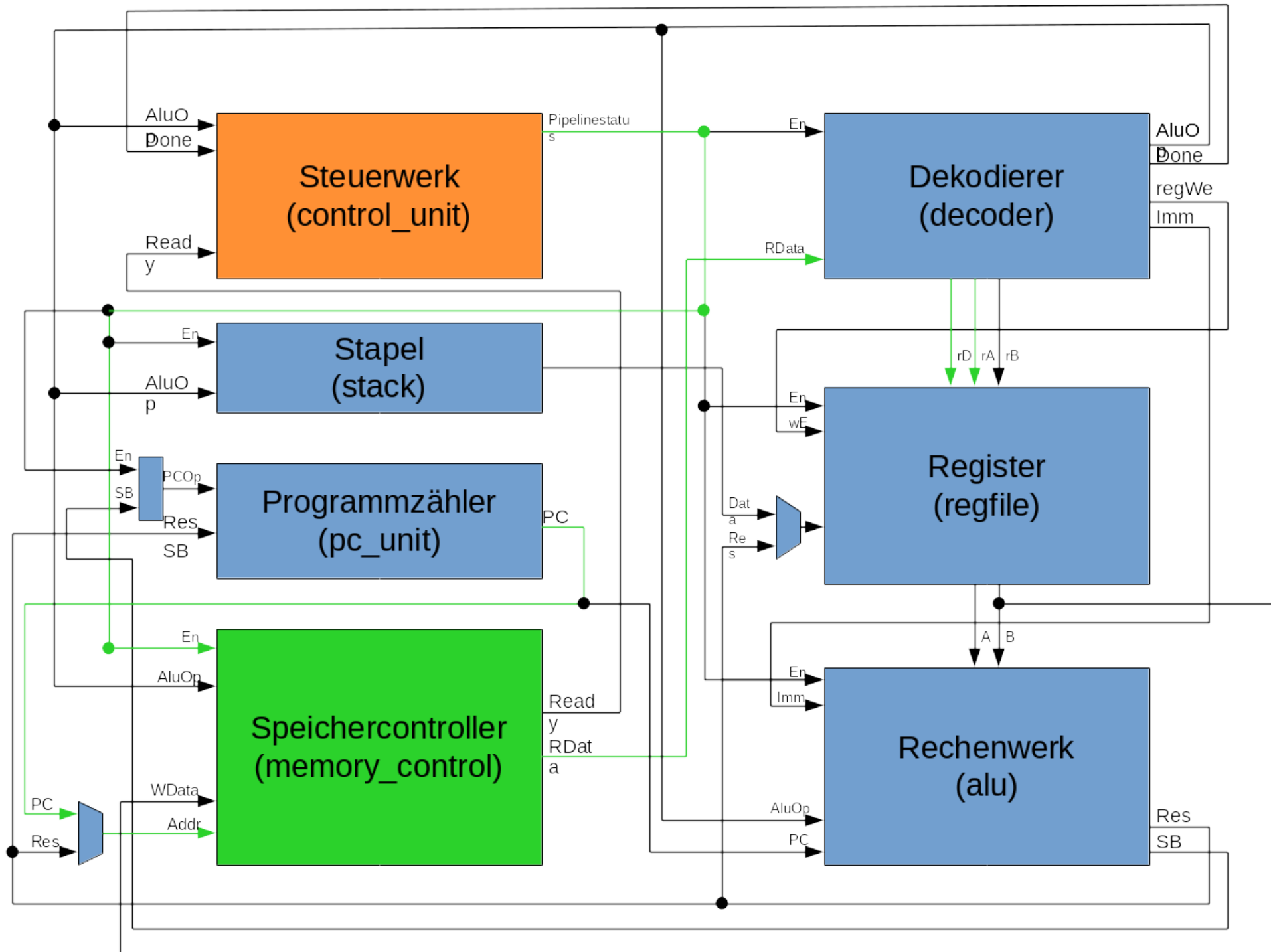
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



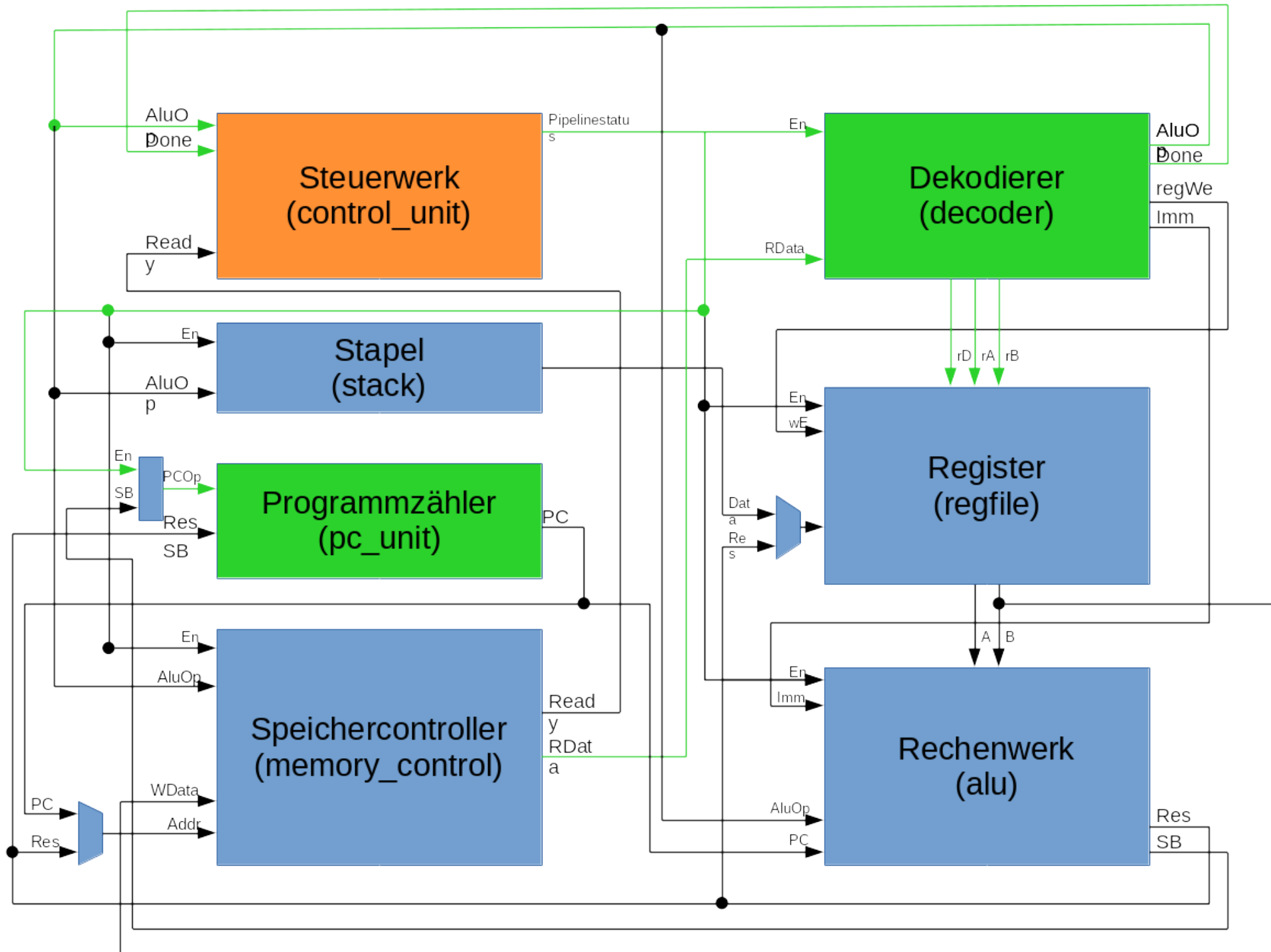
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



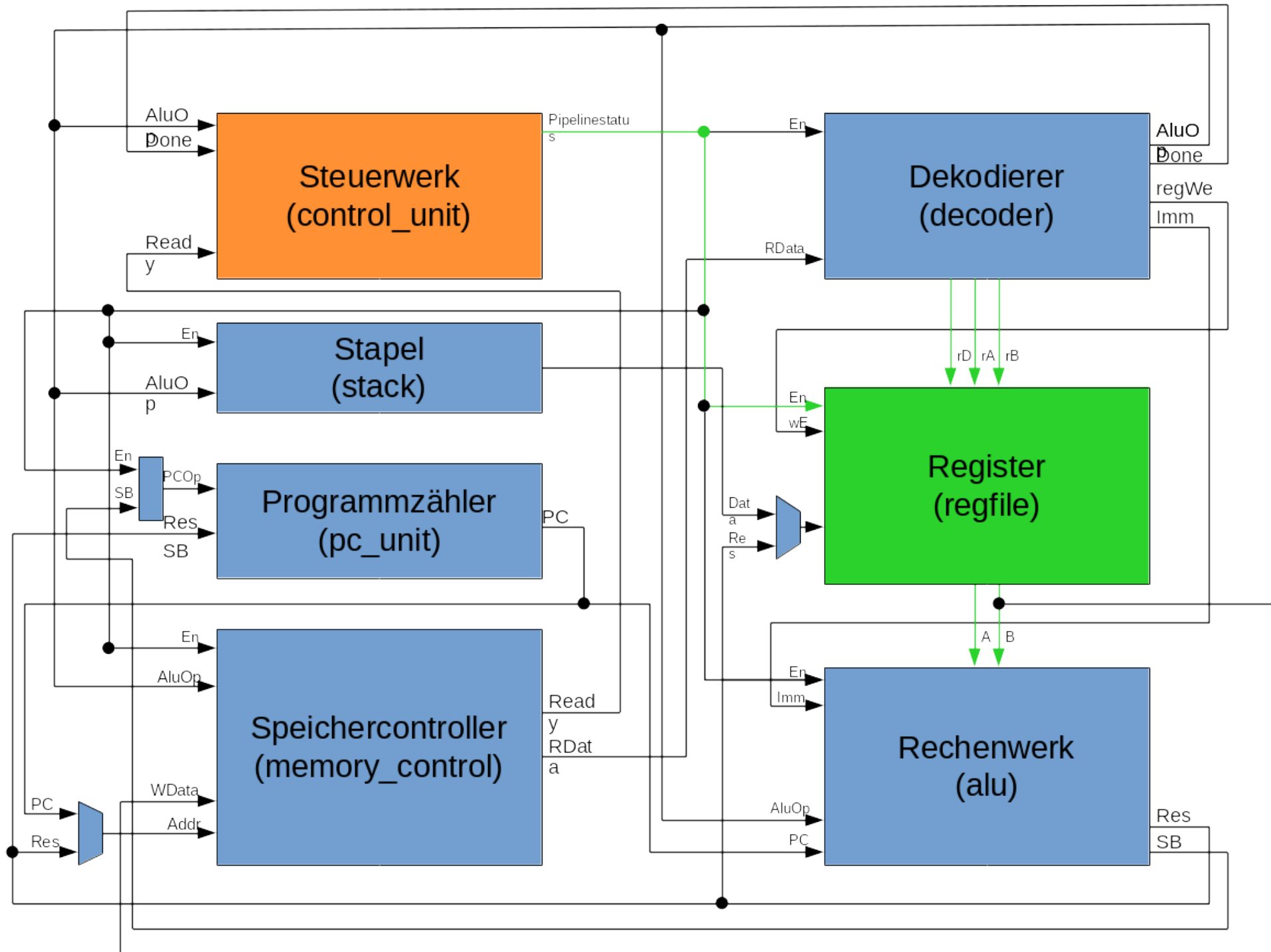
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



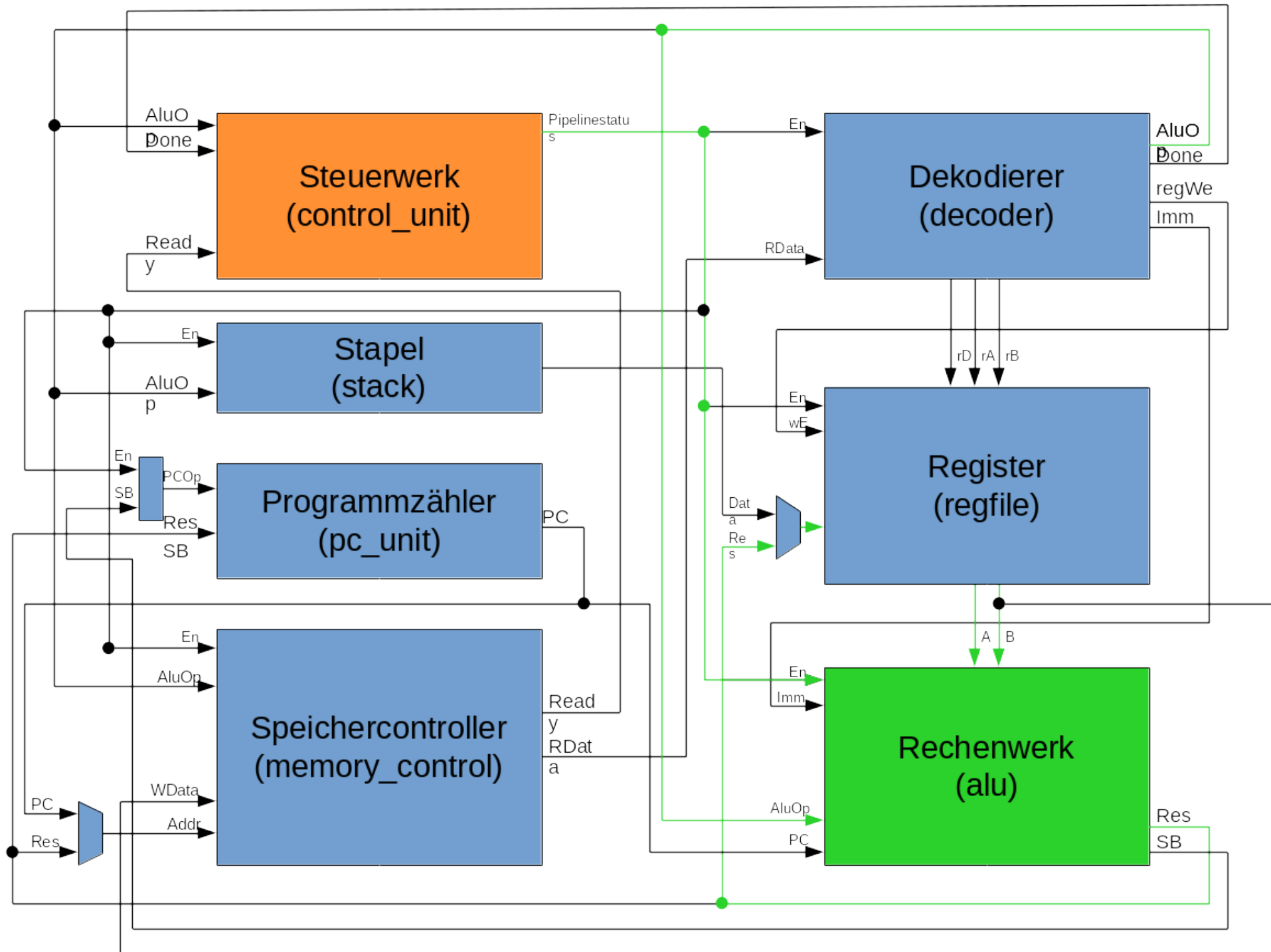
Fetch Decode Read Execute MemoryWrite StackReadWrite RegWrite



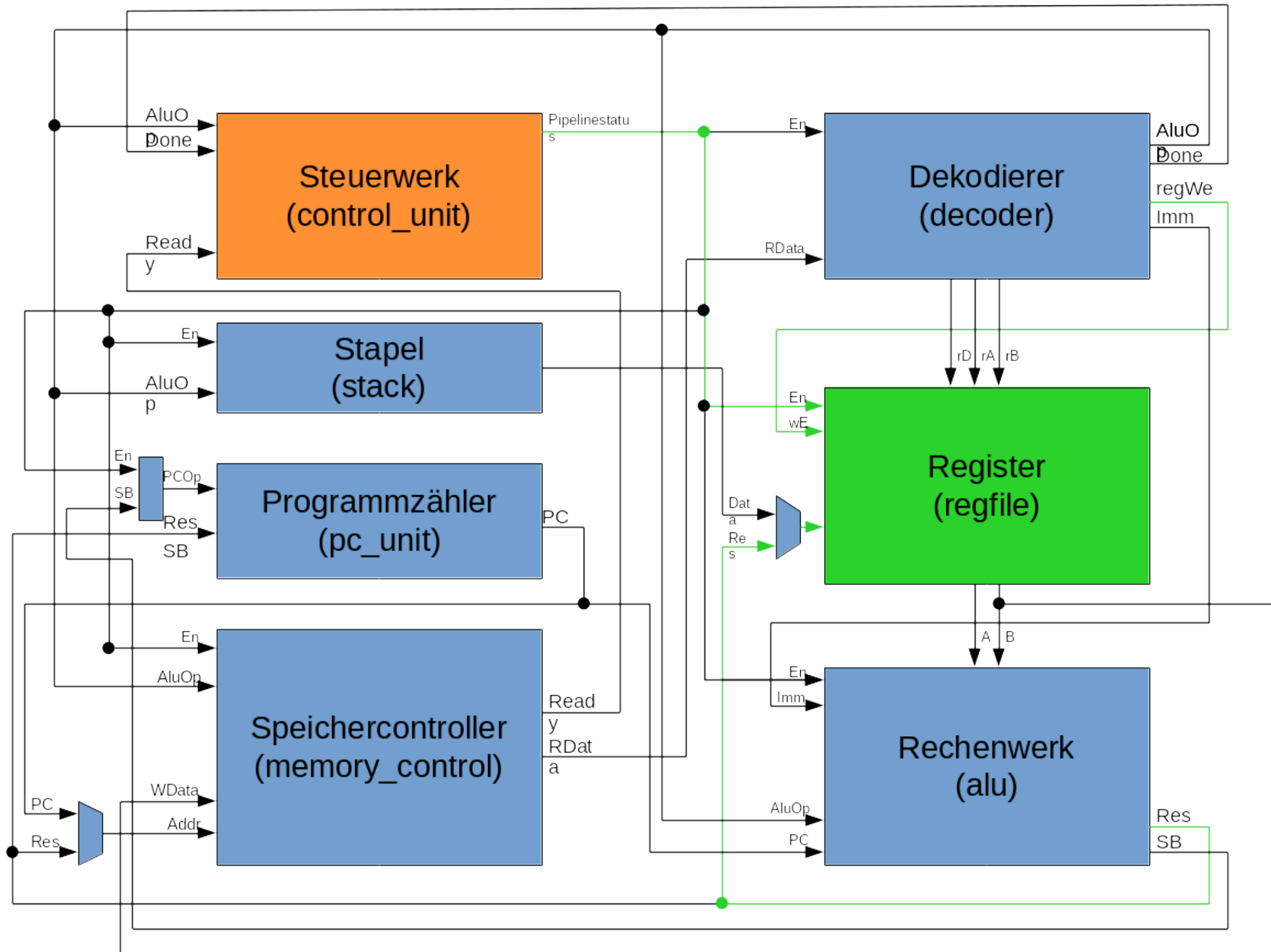
Fetch Decode **Read** Execute MemoryWrite StackReadWrite RegWrite



Fetch Decode Read **Execute** MemoryWrite StackReadWrite RegWrite



Fetch Decode Read Execute MemoryWrite StackReadWrite **RegWrite**





Demo

**Vielen Dank für ihre
Aufmerksamkeit**