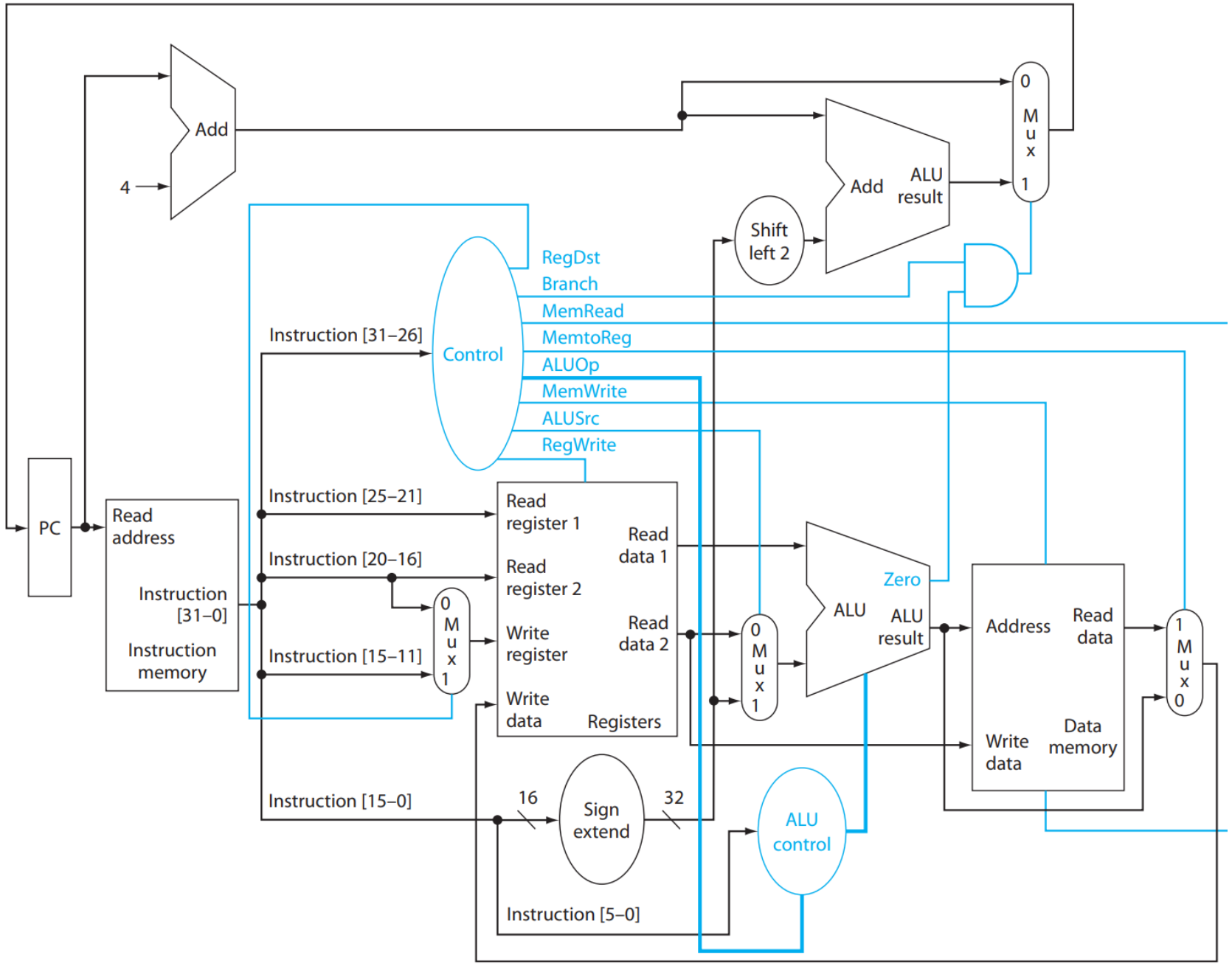
Eintaktprozessor

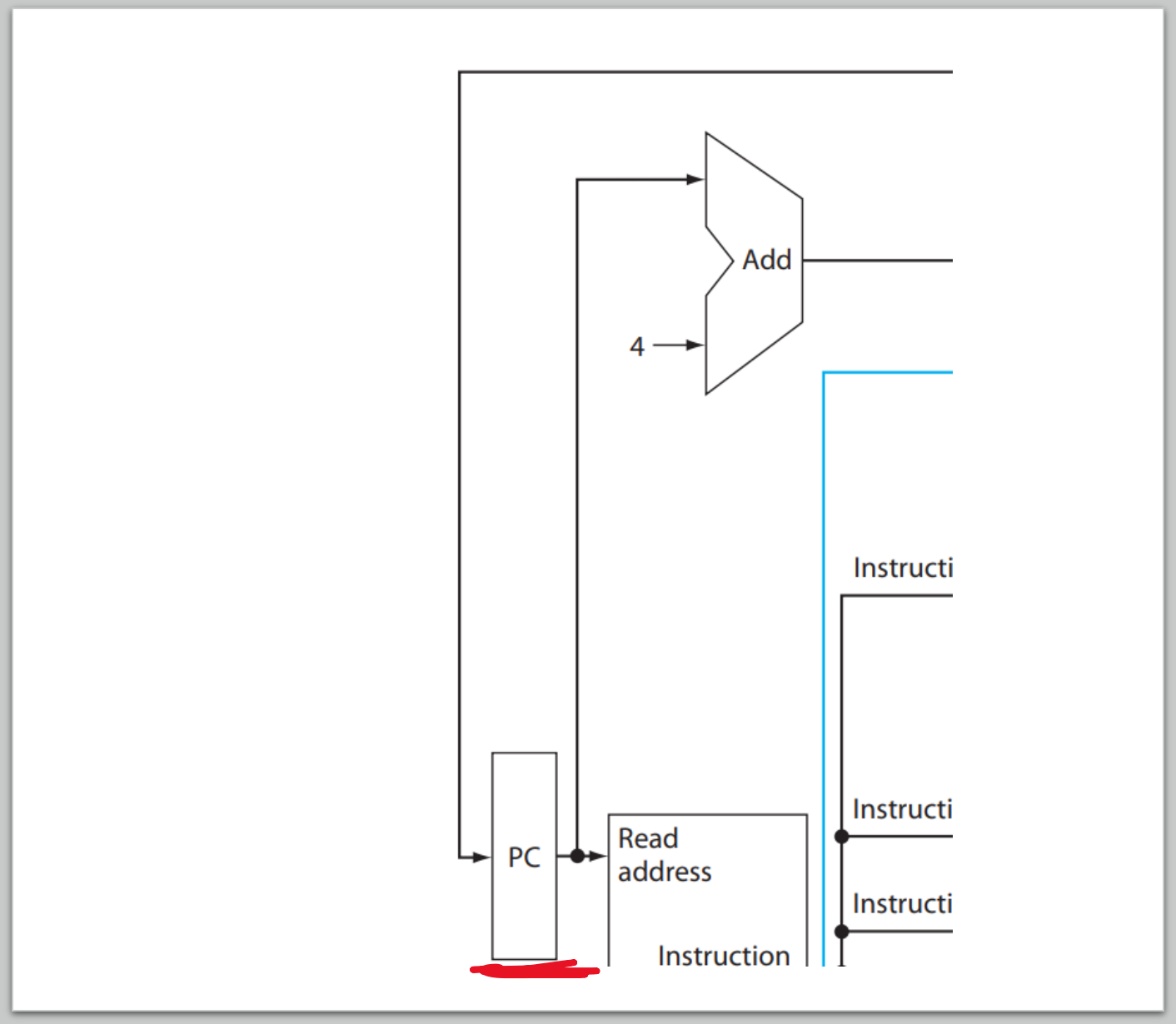


- Blau: Steuerung
  - Steuert den Prozessor
  - Kontrolliert den Datenpfad
  - Führt den Befehl aus
- Schwarz: Datenpfad
  - Führt tatsächliche Operationen aus
- Warum Eintaktprozessor?
  - Ein Takt für einen Befehl

Datenpfad

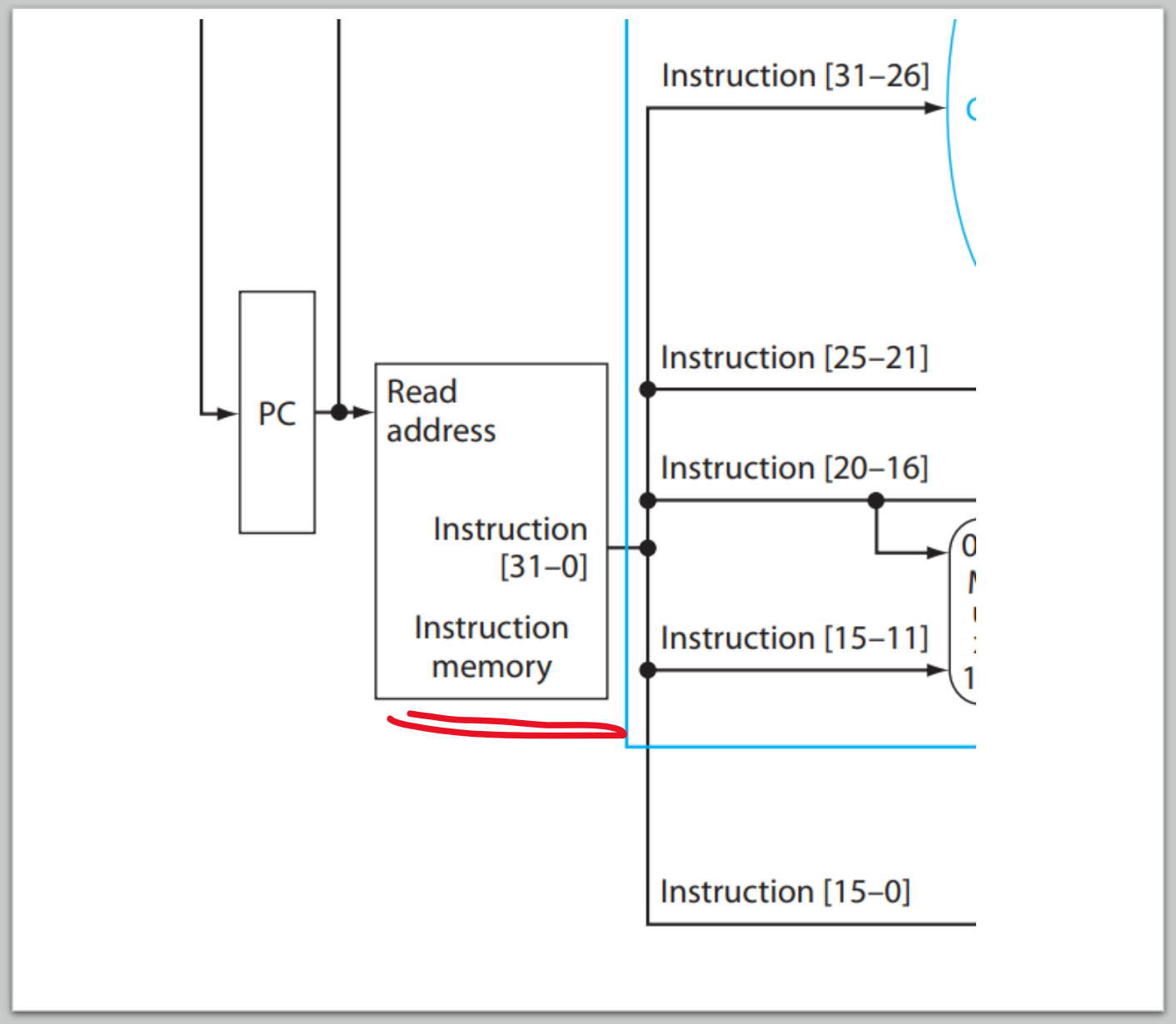
PC - Program Counter

- Zählt beim wievielten Byte wir im Programmtext sind
- Wird in jedem Takt bedingungslos um 4 erhöht
  - Nächster Befehl
  - Word-Abstand -> 1 Word = 4 Byte



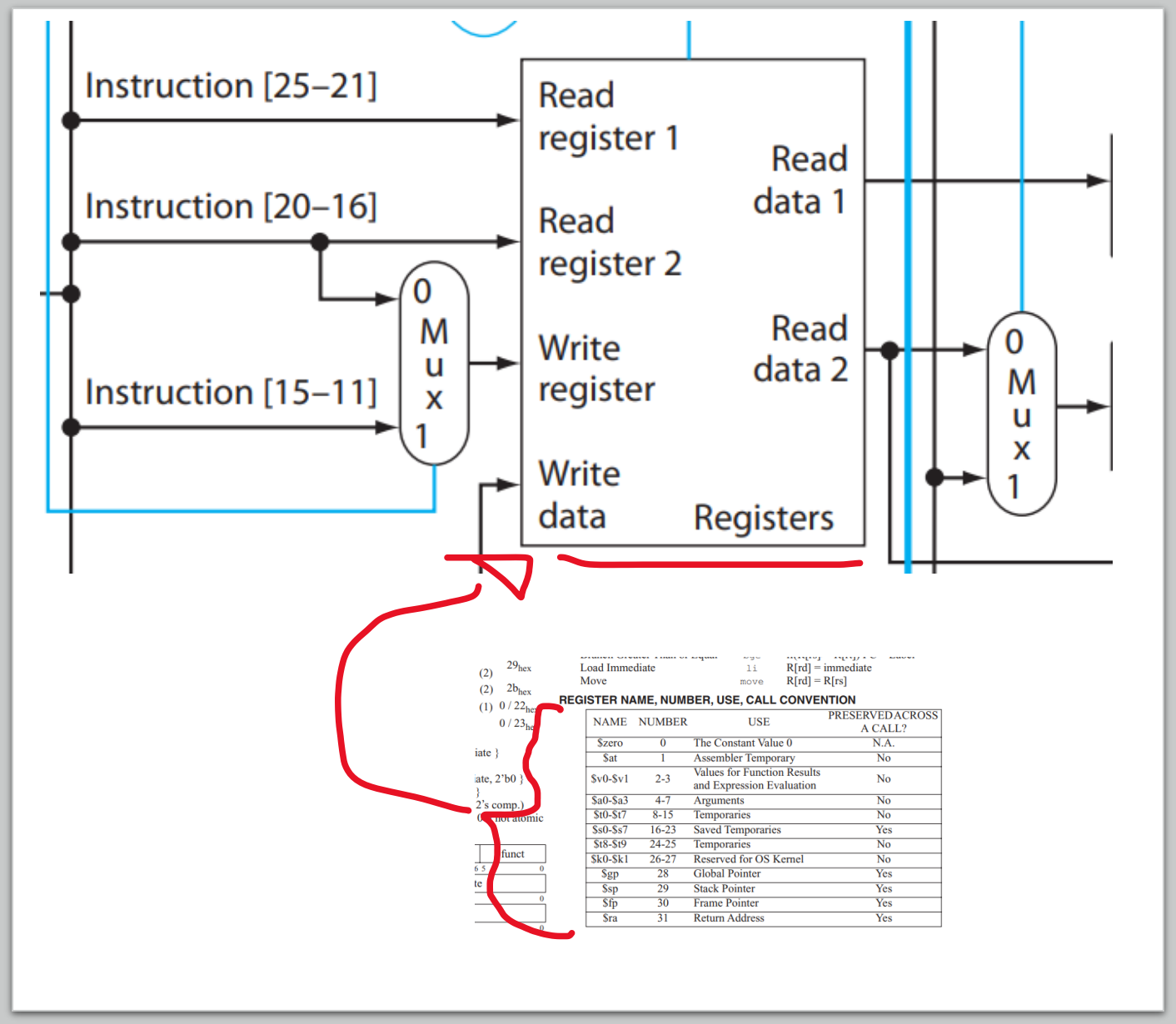
Instruction Memory

- Befehle vom Programm gespeichert
- Befehl wird vom PC geladen
- NächstBefehl = InsMem[PC]
- In jeden Takt: Befehl bei Adresse PC geladen
- Befehl wird in einzelne Fleder aufgeteilt
  - Op, rt, rs, rt, Imm



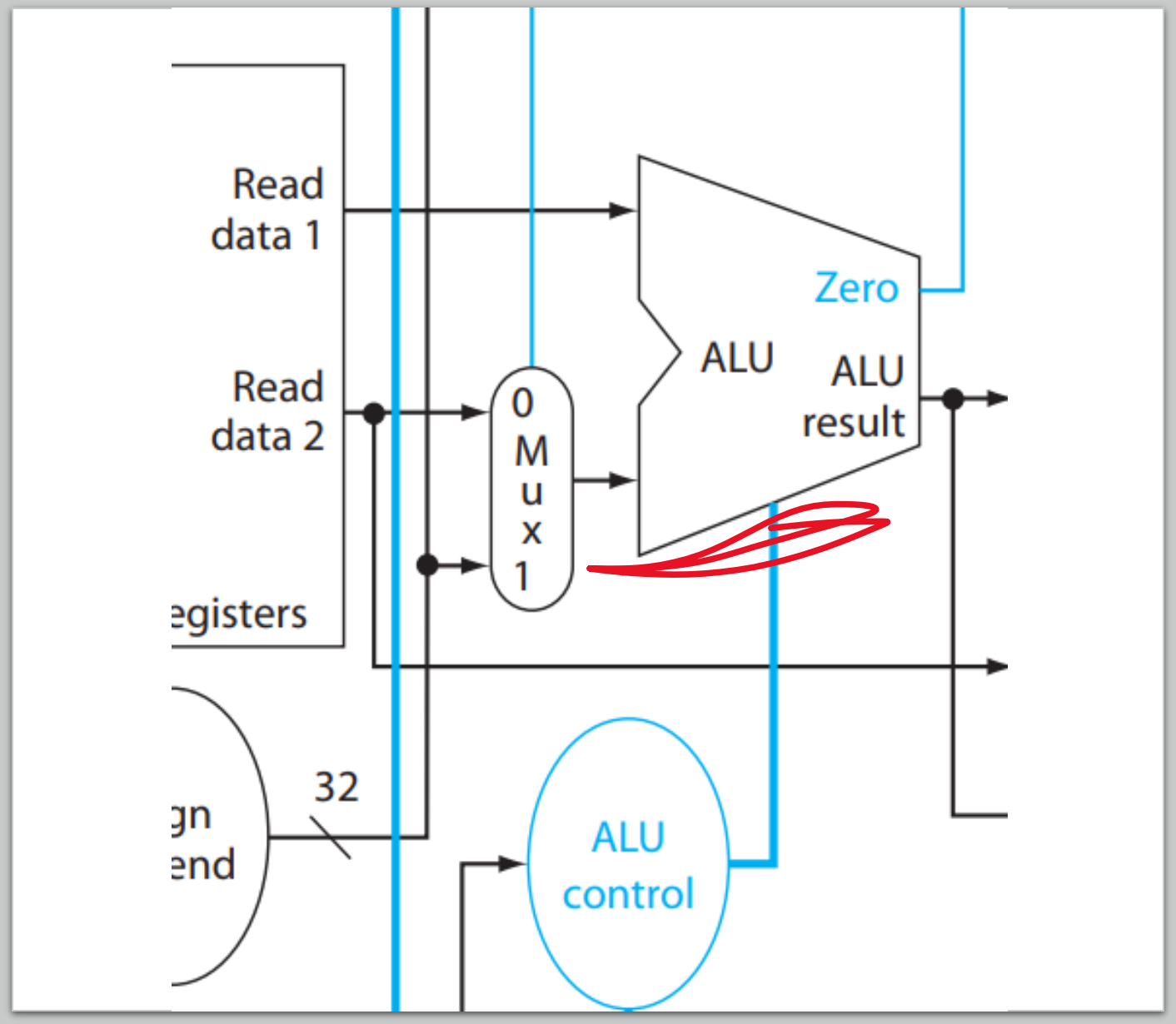
Registers

- Gleicher Registersatz, wie bei MARS
- Input: (vom Geladenen Befehl)
- Read reg 1 = Inst[25-21] = rs
- Read reg 2 = Inst[20-16] = rt
- Write reg = Inst[15-11] = rd (oder = Inst[20-16] = rt bei I-Format)
- Outputs:
- Read data 1= Reg[Inst[25-21]]
- Bsp: add $t0, $t1, $a0
- Read data 1 = $t0



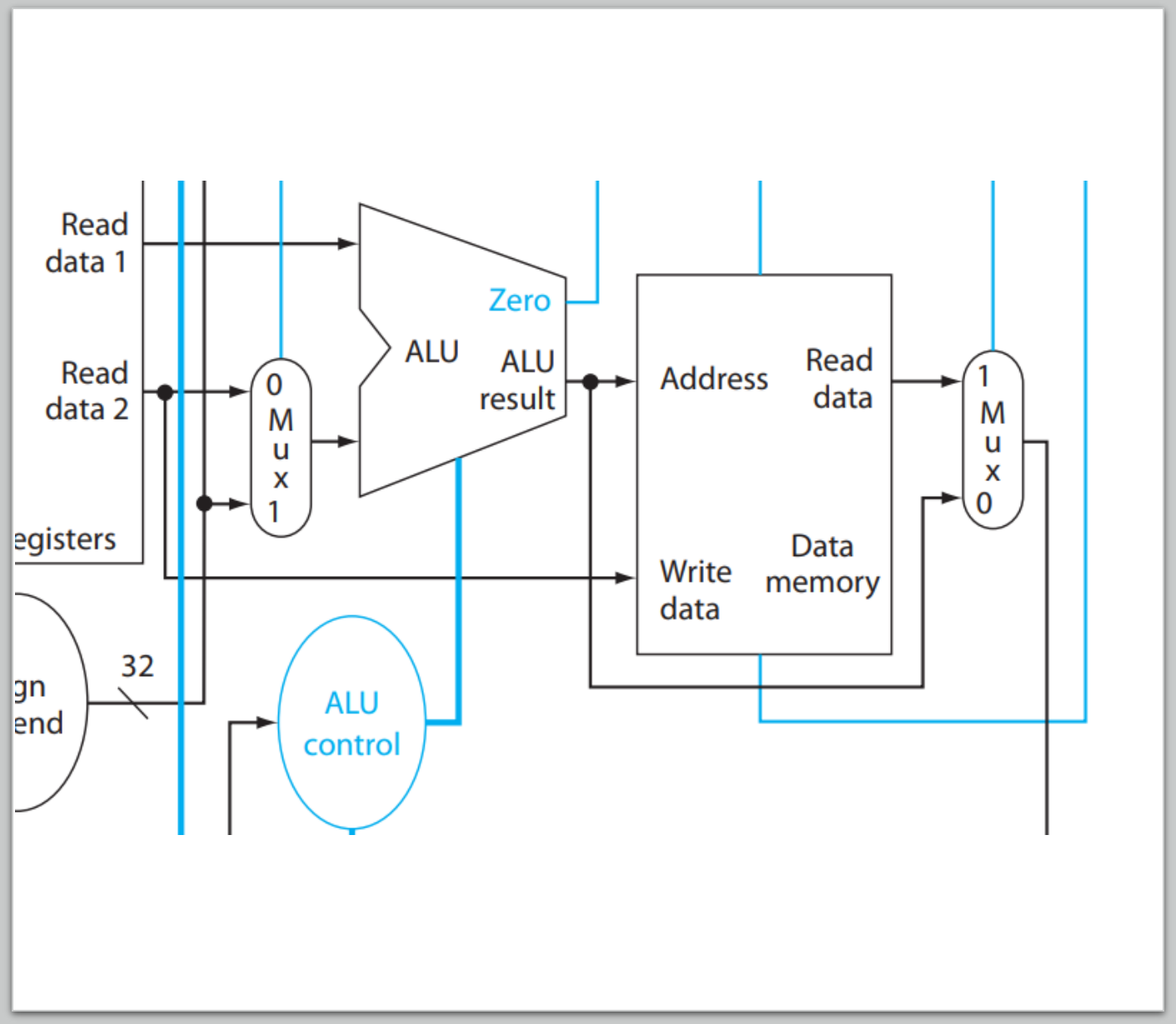
ALU

- Input:
- 1. Read data 1
- 2. Read data 2 / Imm. (Sign Ext.)
- Steuersignale (A_inv, B_inv, OP) <- ALU Control <- Befehl Op u. Funct
- Output:
- In Register geladen (add $t1, $t2, $t3 in $t1 geladen)
- Oder als Addresse für Speicher
- Zero für branch (bnq, bne)



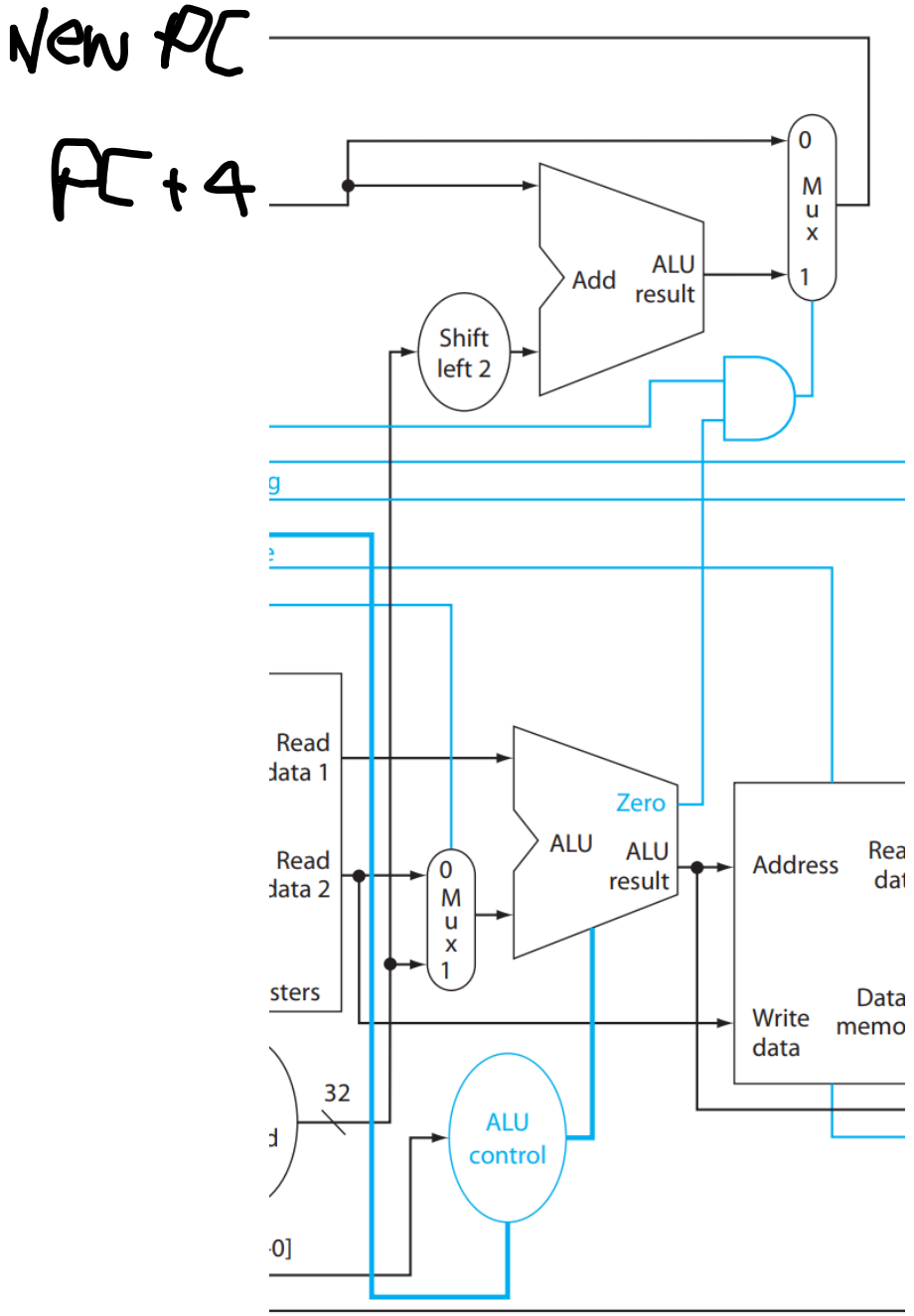
Data Memory

- Speicher
- Benutzt bei: lw, sw, lb, sb …
- Adresse ist Ergebnis von ALU
  - Reg. Mit Imm addiert: lw $sp, 4($sp) -> Addr = $sp + 4
- Mux entscheidet zwischen Ergebnis aus Speicher (bei lw) oder Ergebnis von ALU (bei add)



Branch Logik

- Für beq, bne
- Zielsprungadresse = Alter PC + 4 * Imm.
- Zero (ALU Output) gibt an ob gleich oder ungleich
- Zero entscheidet ob Zielsprungadresse oder PC + 4 (branch oder kein branch)



Steuerung

Steuerung

- Durch Op-Code bestimmt

- RegDst: Ergebnis in rt oder rd?
  - 0: rt, 1:rd
- Branch: Branch-Befehl
- MemRead: Aus speicher lesen
- MemtoReg: Aus speicher laden
  - 0: ALU, 1: MEM
- ALUOp: Was die ALU machen soll
- MemWrite: In speicher schreiben
- ALUSrc: 2. Operand Reg oder Imm?
  - 0: Register, 1: Imm
- RegWrite: Ergebnis in Register laden