Architecture

MIPS vs RISC V

Instruction Set Architecture
(ISA)

Rechnerarchitekturen: Beispiele RISC-Prozessoren

Befehlssatz MIPS

- Befehlswort
- Arithmetisch-Logische Befehle
- Transportbefehle
- Verzweigungen und Sprünge

Befehlssatz RISC

- Befehlswort
- Arithmetisch-Logische Befehle
- Transportbefehle
- Verzweigungen und Sprünge

Ableitung der Mikroarchitekturen aus dem Befehlssatz des RISC V

Befehlssatzarchitektur

(Instruction Set Architecture)

ISA

MIPS RISC V

Übersicht Arithmetisch - logische Befehle

MIPS

Arithmetische Befehle:

$R_A := \{ ADD, ADDV, SUB, SUBV, MUL, MULV, DIV \}$

$\square := \{ \boxplus, \boxplus, \boxminus, \boxminus, \boxdot, \boxplus, \boxslash \}$ $\quad d := s \square t$

$I_A := \{ ADDI, ADDIV \}$

$\blacksquare := \{ \blacksquare, \blacksquare \}$ $\quad d := s \blacksquare I$

Logische Befehle:

$R_A := \{ AND, OR, NOR, XOR, SLL, SLR, SLLV, SLRV, SRA \}$

$\boxdot := \{ \boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \quad , \quad , \boxdot \}$

$I_L := \{ ANDI, ORI, XORI \}$

$\boxdot := \{ \boxdot, \boxdot, \boxdot \}$

Registerdirekte Adressierung: R-Format

Arithmetisch-Logische Befehle: R(egister) Format

Struktur: $\langle '000000', \langle r, s, d \rangle^5, F^5, a^6 \rangle$ $\qquad 0^6 := '000000'$

$R_A := \{ ADD, ADDV, SUB, SUBV, MUL, MULV, DIV \}$

$\square := \{ \boxplus, \boxplus, \boxminus, \boxminus, \boxdot, \boxplus, \boxslash \}$

$R_L := \{ AND, OR, NOR, XOR, SLL, SLR, SRA \}$

$\boxdot := \{ \boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot, \boxdot \}$ (S)

$F := R_A \cup R_L$

$S \subseteq F := \{ SLL, SLR, SRA \}$

$\{SLLV, SLRV\}$?

Verhalten: $\{ F \notin \{S\}, \{\langle d \rangle := \langle r \rangle \square \langle s \rangle\} ? \langle d \rangle := \langle r \rangle \square a \}$

Arithmetisch-logische Befehle

Registerdirekte Adressierung: R-Format

Verhalten: $\{ F \notin \{S\}_1 \{ \langle d \rangle := \langle r \rangle \square \langle s \rangle \} ? \langle d \rangle := \langle r \rangle \square a \}$

Befehlszyklus:

IF: $\langle pc \rangle := (\langle pc \rangle \mapsto IM[\,]) + 4$

ID: $\left\{ \{ 0 = '000000' \mid \begin{matrix} \{ \square \in \{F\} \} ? \{ \square \in \{S\} \} \\ \{ \square \mapsto F \} ? \{ \square \mapsto S \} \end{matrix} \right\}$

OF & EX: $\langle\ \rangle := \langle r \rangle \square \langle s \rangle$ $\qquad$ $\langle\ \rangle := \langle r \rangle \square a$

MEM: $\{\}$

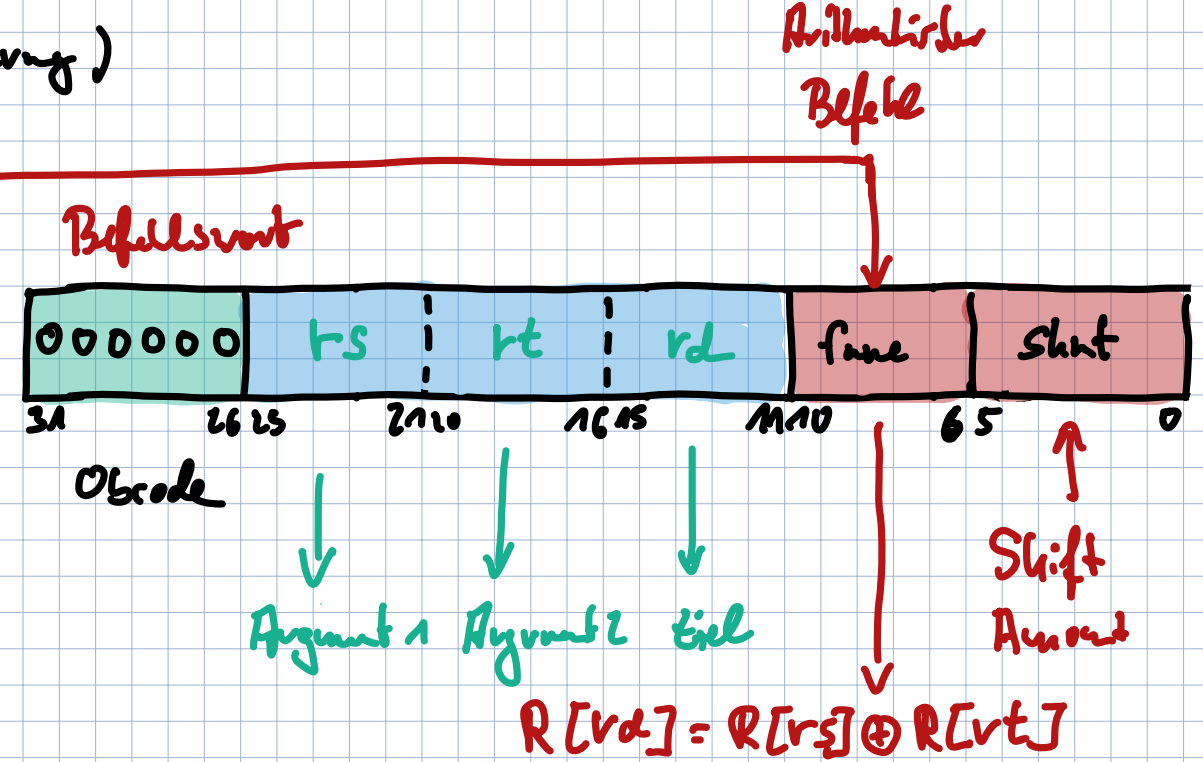
OS: $\langle d \rangle := \langle\ \rangle$

Architektur: Befehlssatz Mips

Befehlsformat: R(egister)-format

Arithmetische Befehle (Direkte Adressierung)

Assembler	Mnemonic	Objekt-Code (funct ⊕)
add	add	100000
addu	add unsigned	100001
sub	subtract	100010
subu	subtract unsigned	100011
and		100100
or		100101
xor		100110
nor		100111
sll	shift left logical	000000
srl	shift right logical	000010
mult	multiplication	011000
multu	mult unsigned	011001
div	division	011010
divu	div unsigned	011011



$$\langle r,s\rangle_{=1}\ \{\langle pc\rangle_{\cup} := \%\,(i\,'00')_{16}\ \}_{w}$$

Arithmetische - logische Befehle I (immediate Format) Unmittelbare Adressierung

Struktur: $\{ \square, \langle r, s \rangle^{16}, I^{16} \}_{32}$

$A_I := \{ \text{ADDI}, \text{ADDIU} \}$

$\square := \{ \square, \boxplus \}$

$L_I := \{ \text{ANDI}, \text{ORI}, \text{XORI} \}$

$\boxdot := \{ \boxdot, \boxdot, \boxdot \}$

$I := I_A \cup I_L$

Verhalten: $\{ d := R \square I \}$

Arithmetisch-logische Befehle

Registerdirekte Adressierung: R-Format

Verhalten: $\{d := R \square I\}$

Befehlszyklus:

IF: $\langle pc \rangle := (\langle pc \rangle \mapsto IM[\,]) + 4$

ID: $\begin{cases} '000100xx' \longmapsto \square := \square \end{cases}$

OF & EX: $\langle\ \rangle := \langle \mapsto \rangle\ \square \pm \{I\}$

MEM: $\{\}$

OS: $\langle d \rangle := \langle\ \rangle$

Architektur: Befehlssatz Mips

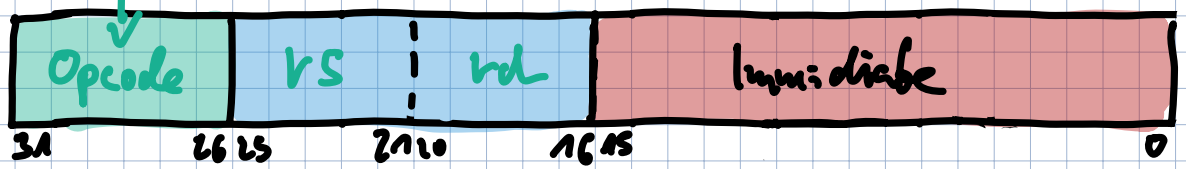
Befehlsformat: I(mmediate)-Format

Arithmetische Befehle (Immediate Addressing)

Assembler

Mnemonic

Opcode



R[rd] = R[rs] + <Immediate>

addi		001000
addiu		001001
slti	set on less than	001010
sltiu	set ... unsigned	001011

±{} Sign Extension: 16 → 32 Bit, Vorzeichenerhalt

Immediate

1111111111111111 1000110011000101

0000000000000000 1000110011000101

Transport-Befehle: I(mmediate)-Format Register-Relative Adressierung

Struktur: $\{O_c, \langle b, n \rangle_{10}, A_{16}\}_4$

$T := \{LB, LH, LUL, LV, SB, SH, SWL, SW\}$

Variablen: $\{O_c = 'xx0xxx, \{\langle r \rangle := [\langle b \rangle] + A\}, [\langle b \rangle + A] := \langle s \rangle\}$

Transport-Befehle: I(mmediate)-Format Register-Relative Adressierung

Verhalten: $\{O_6 = \text{'xx0xxx'}, \{\langle v\rangle := [\langle b\rangle] + A\}, [\langle b\rangle + A] := \langle s\rangle\}$

Befehlszyklus:

IF: $\langle pc\rangle := (\langle pc\rangle \mapsto IM[\,]) + 4$

ID:

'100001' $\{1 := 16\}$	lhu
'100010' $\{1 := 3\}$	lvl
'100011' $\{1 := 325\}$	lw
'100111' $\{1 := 8\}$	lb

OF & EX: $\langle x\rangle := \langle b\rangle + A$

MEM: $\langle y\rangle := [\langle x\rangle]$

OS: $\langle v\rangle := \langle y\rangle$

Transport-Befehle: I(mmediate)-Format Register-Relative Adressierung

Verhalten: $\{O_6 = '\times\times 0\ \times\times\times', \{\langle v\rangle := [\langle b\rangle + A]\}, [\langle b\rangle + A] := \langle s\rangle\}$

Befehlszyklus:

IF: $\langle pc\rangle := (\langle pc\rangle \mapsto IM[\,]) + 4$

ID:

$'1\,0\,1\ 0\,0\,0'\ \{l := 8\}$	sb
$'1\,0\,1\ 0\,0\,1'\ \{l := 16\}$	sh
$'1\,0\,1\ 0\,1\,0'\ \{l := 5\}$	
$'1\,0\,1\ 0\,1\,1'\ \{l := 32\}$	sw

OF & EX: $\langle x\rangle := \langle b\rangle + A$

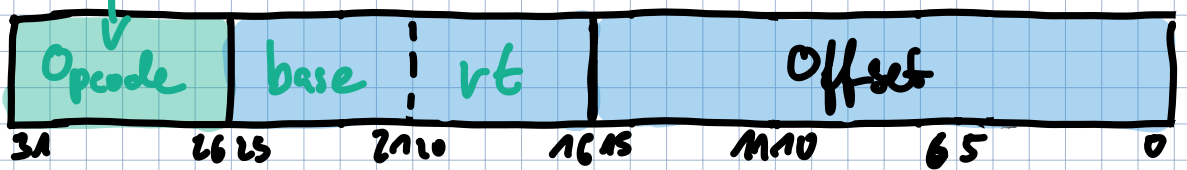
MEM: $[\langle x\rangle] := \langle y\rangle$

OS: $\{\}$

Architektur: Befehlssatz Mips

Befehlsformat: I(mmidiate)-Format

Transportbefehle (Register relative Adressing)



Mnemonic		Opcode
lb	load byte	100000
lh	ld half word	100001
lwl	ld word left	100010
lw	ld word	100011
sb	store byte	101000
sh	st half word	101001
swl	st word left	101010
sw	st word	101011

R[rt] = MEMORY [R[base] + offset] [0 : n-1]

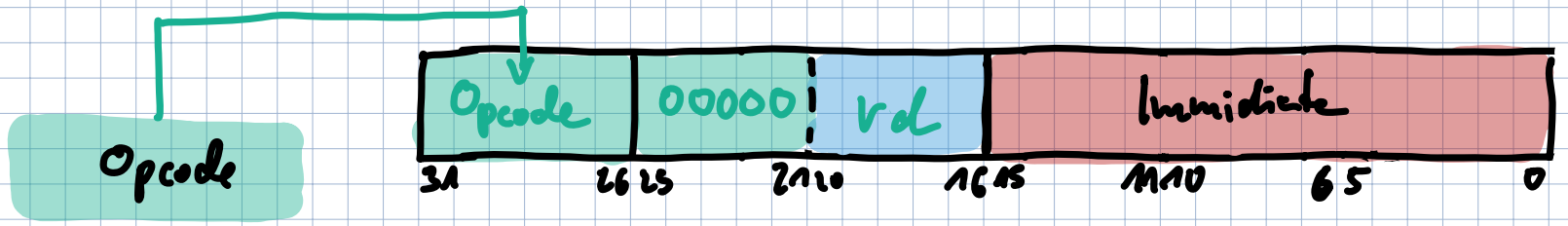
R[rt] = R[rt] MERGE [R[base] + offset]

MEMORY[R[base] + offset] = R[rt] [0 : n-1]

Architektur: Befehlssatz Mips

Befehlsformat: I(mmediate)-Format

Transportbefehle (Immediate Addressing)



Mnemonic		Opcode
lui	ld upper immediate	001111
sui	st upper immediate	

$R[rd] = Immediate \parallel 0_{16}$

← Shift + Concatenation 0_{16}

Verzweigungen (Bedingte Sprünge): I-Format

Befehlszeichenrelative Adressierung

Verhalten: $\{ B_{c1} \langle r,s \rangle_{w1}\ i_{16} \}_w \mapsto \{ \langle r,s \rangle_{=1}\ \{ \langle pc \rangle := \% (i \| '00')_{18} \}_w$

$B \in \{BEQ, BNE, BLEZ, BGTZ\}$ Mnemonics

$\square \in \{\boxplus, \boxminus, \boxtimes, \boxslash\}$ ALU-Operationen

Befehlszyklus:

IF: $\langle pc \rangle := (\langle pc \rangle \mapsto IM[\,]) + 4$

ID: $\begin{cases} \square \in B \times O \\ \square \mapsto B[i] = O[i] \end{cases}$

OF & EX: $\langle r \rangle \square \langle s \rangle$

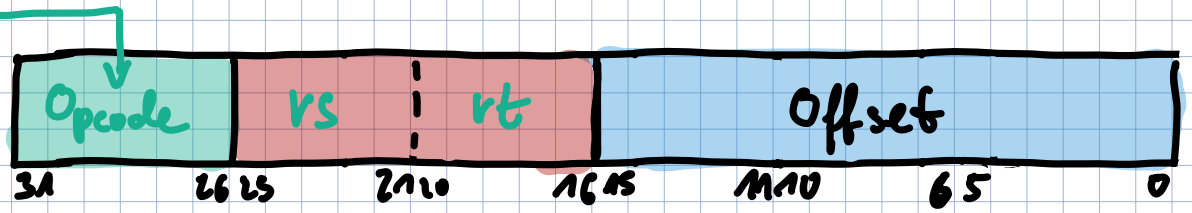
MEM: $\langle pc \rangle := \lceil pc \rceil + \langle x \rangle \ , \{\}$

OS: $\{\}$

Architektur: Befehlssatz Mips

Befehlsformat: I (immediate) - Format

Bedingte Sprünge (Verzweigungen) (Befehlszähler-Relative Adressierung)

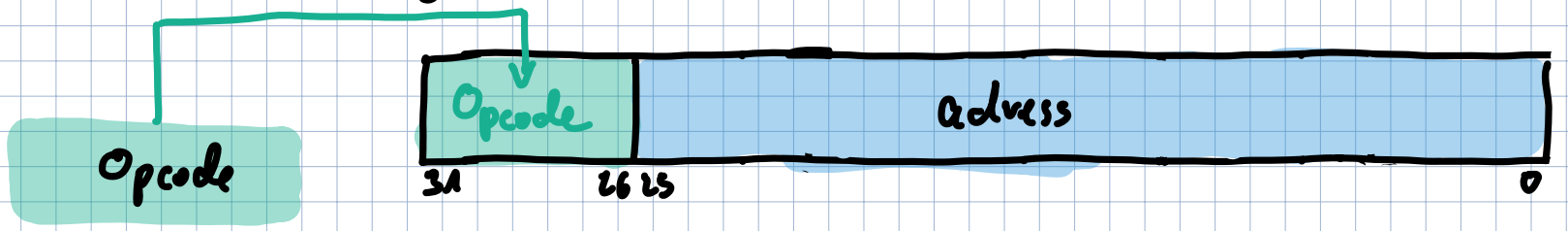


Mnemonic		Opcode
bne	Branch on non equal	000101
beq	Branch on equal	000100
blez	Branch on less than or equal to zero	000110
bgtz	Branch on greater than or equal to zero	000111

Architektur: Befehlssatz Mips

Befehlsformat: J(ump)-Format
R-Format

Sprünge (Befehlszähler-Region Adressierung)
(Register-Relative Adressierung)



Mnemonic		Opcode
j	jump	000 010
jal	jump and link	000 011
jr	jump register	001000

$PC = PC_{[PC]-2^{28}} + 4 \,||\, address \,||\, 0_2$

$PC = R[rs]$

Architektur: Befehlssatz MIPS

Programmierkonventionen der Register-Datei

x0	z	zero constant	–
x1	ra	return address	caller
x2	sp	stack pointer	–
x3	gp	global pointer	–
x4	tp	thread pointer	callee
x5	t0–t2	temporaries	caller
x8	s0, fp	saved / frame pointer	caller
x9	s1	saved register	callee
x10–x11	a0–a1	args / return values	caller
x12–x17	a2–a7	args	callee
x18–x27	s2–s11	saved registers	callee
x28–x32	t3–t6	temporaries	caller

Architektur: Befehlssatz RISC-V : Übersicht u. Vergleich MIPS & RISCV

Arithmetische Befehle:

$A_R := \{ ADD, ADDU, SUB, SUBU, MUL, MULU, DIV \}$

$\square := \{ +, +, -, -, \cdot, \cdot, / \}$ $\quad d := s \square t$

$A_I := \{ ADDI, ADDIU \}$

$\square := \{ \square, + \}$ $\quad d := s \square I$

Logische Befehle:

$L_R := \{ AND, OR, NOR, XOR, SLL, SLR, SLLV, SLLV, SRA \}$

$\square := \{ \odot, \odot, \odot, \otimes, \odot, \odot, \quad , \quad , \gg \}$

$L_I := \{ ANDI, ORI, XORI, SLLI, SRLI, SRAI \}$

$\square := \{ \odot, \odot, \odot \}$

Arithmetische Befehle

[register-direkte Adressierung]

RISC V

Struktur: **R(egister)-Format**

$$\langle f_B^7, \langle t,s \rangle^{10}, f_A^3, \langle d \rangle_1^5, \overbrace{0^7}^{\text{'0110011'}} \rangle$$

$O_A = \text{'0110011'}$

$A_1 := \{ ADD_0, SLL_1, SLT_2, SLTU_3, XOR_4, SRL_5, OR_6, AND_7 \}$

$\square := \{ \boxplus, \boxdot, \boxminus, \boxminus, \boxtimes, \boxtimes, \boxtimes, \boxtimes \}$

$A_1' := \{000, 001, 010, 011, 100, 101, 110, 111\}$

$A_2 := \{ SUB_0, SRA_5 \}$

$A_2' := \{000, 001\}$

$\square_2 := \{ \boxminus, \boxplus \}$

Arithmetische Befehle

[register-direkte Adressierung]

RISC V

Verhalten: $\{ O^t = O_A \} \triangleright \{ (f_A \neq \square) \triangleright \{ (f_B = \text{'0000000'}) \triangleright \{ \langle d \rangle := \langle s \rangle \square_1 \langle t \rangle \}_1$

$(f_B = \text{'0100000'}) \triangleright \{ \langle d \rangle := \langle s \rangle \square_2 \langle t \rangle \} \}$

$(f_A = \square) \triangleright \{ (s < t) \triangleright \{ \langle d \rangle := 1 \} ? \{ \langle d \rangle := 0 \} \}$

Arithmetisch-logische Befehle:

$0^7\}_V$

RISC V

[Absolute Adressing]

m: Shift amount

Struktur: I(mmediate) - Format

0^7

I_1 $\quad \langle i^{12}, \langle s\rangle^5, f_B^3, \langle d\rangle^5, '0010011'\rangle \qquad O_I = '0010011'$

I_2 $\quad \langle '0000000', m^5, \langle s\rangle^5, f^3, \langle d\rangle^5, '0010011'\rangle$

I_3 $\quad \langle '0100000', m^5, \langle s\rangle^5, f^3, \langle d\rangle^5, '0010011'\rangle$

$I_1 :=$ {	addi	,	slti	sltiu	xori	,	ori	andi }
$I_2 :=$ {		slli	,	,	,	srli	,	, }
$I_3 :=$ {		,	,	,	srai	,	, }
$I' :=$ {	000	001	010	011	100	101	110	111 }
$O_L :=$ {	$\oplus$	$\ll$	$<$	$<$	$\otimes$	,	$\oplus$	$\odot$ }
$O_A :=$ {	,	,	,	,	$\gg$	,	,	}

Arithmetisch-logische Befehle: I(mmediate)-Format

Verhalten: $\{(O = O_I) \triangleright \{(\square \neq \boxtimes) \triangleright \{ \langle d \rangle := \langle s \rangle \square \{\pm i_{n:0}\}_1$

$(\square = \boxtimes) \triangleright \{(\langle s \rangle \square \{\pm i\}) \triangleright \{ \langle d \rangle := (1 \,?\, 0)_1$

$(\square = \otimes) \triangleright \{(i^{n:7} = '000000') \{ \langle d \rangle := \langle s \rangle \square_L \; m\},$

$\{(i^{n:7} = '010000') \{ \langle d \rangle := \langle s \rangle \square_R \; m\}\}\}$

RISCV

Transport-Befehle : LOAD

[ugisha - relative Addressing]

Struktur : I(mmediate)-Format

$$\{ i^{12}, \langle s \rangle^{5}, f^{3}, \langle d \rangle^{5}, o^{7} \}_{W}$$

$$\{ i, \langle s \rangle, f, \langle d \rangle, '0000011' \}$$

$$L = \{ ldb, ldh, ldw, lbu, lhu \} \qquad O_L := '0000011'$$

$$L' = \{ 000, 001, 010, 011, 100 \}$$

$$D = \{ nop \}$$

$$l = \{ 8, 16, 32, 8, 16 \}$$

Verhalten : $O_L \triangleright \{ \langle d \rangle := [\langle s \rangle + \{\pm i\}] \}_{l[f]}$

RISC V

Transport-Befehle: STORE

[register-relative Addressing]

Struktur: S(tore)-Format

$$\{ i_A^7, \langle t,s \rangle^{10}, f_A^3, i_B^5, 0^7 \}_w$$

$$\{ i_A, \langle t,s \rangle, f_A, i_B, '0100011' \}$$

$$S := \{ sb, sh, sw \}$$

$$O_S := '0100011'$$

$$S' := \{ 000, 001, 010 \}$$

$$\ell = \{ 8, 16, 32 \} \qquad \square := \{ nop \}$$

Verkettung:

$$O_S \triangleright \{ [\langle s \rangle + \{ \pm [i_A \mid i_B] \}]_{\ell[f]} :: \langle t \rangle \}$$

Verzweigung (Bedingter Sprung)

RISC V

[Register-relative Addressing]

Struktur: SB-Format (Immediate)

$\{ i^{1}, \langle s \rangle^{5}, F^{3}, \langle d \rangle^{5}, '1100111' \}$ $\quad O_B = '1100111'$

$B := \{ beq, bne, blt, bge, bltu, bgeu \}$

$B' := \{ 000, 001, 010, 011, 100, 101 \}$

$\square := \{ =, \neq, \leq, \geq, <, > \}$

Verhalten: $\{ (O^{7} = O_B) \triangleright \{ (\langle s \rangle \square \langle d \rangle) \triangleright \{ \langle pc \rangle := \langle pc \rangle + \{ \pm i \} \} \} \}$

Unbedingte Sprünge (Unconditional jumps)

RISC V

[register-relative Adressierung]

Struktur: I(mmediate)-Format:

$O_J = '1100111'$

$\{ i^{12}, \langle s \rangle^5, F^3, \langle d \rangle^5, O^7 \}_V$

$\{ i^{12}, \langle s \rangle^5, '000', \langle d \rangle^5, '1100111' \}$

$B ::= \{ jalr \}$ jump and link register

$B' ::= \{$

$\square ::= \{\}$

– Format

Verhalten: $O_J \triangleright \{ \langle d \rangle := \langle pc+4 \rangle ; \langle pc \rangle := \langle s \rangle + \{ \pm i \} \}$

Unbedingte Sprünge (Unconditional jumps)

RISC V

[spider-direkte Adressierung]

Struktur: U(nconditional) J(ump)-Format:

$\{ i^{20}, \langle d \rangle^5, 0^7 \}_V$

$\{ i^{20}, \langle d \rangle^5, '1101111' \}$

$O_J = '1101111'$

$B := \{ jal \}$ jump and link

$B' := \{\}$

$\square := \{\}$

Verhalten: $O_J \triangleright \{ \langle d \rangle := \langle pc+4 \rangle ; \langle pc \rangle := \{ \pm i \} \}$

Architektur: Befehlssatz RISC-V

Programmierkonventionen der Register-Datei

x0	zero	zero constant	–
x1	ra	return address	caller
x2	sp	stack pointer	–
x3	gp	global pointer	–
x4	tp	thread pointer	callee
x5	t0–t2	temporaries	caller
x8	s0, fp	saved / frame pointer	callee
x9	s1	saved register	callee
x10–x11	a0–a1	args / return values	caller
x12–x17	a2–a7	args	callee
x18–x27	s2–s11	saved registers	callee
x28–x32	t3–t6	temporaries	caller

RISC-V

Befehlssatz-Architektur
Instruction Set Architecture
(ISA)

Mikroarchitektur

Verhalten ⟶ Struktur

Arithmetische Befehle

[register-direkte Adressierung]

RISC V

Verhalten: $\{ O^t = O_A) \triangleright \{ (f_A \neq \square) \triangleright \{ (f_B = '0000000') \triangleright \{ \langle d \rangle := \langle s \rangle \square_1 \langle t \rangle \}_1$

$(f_B = '0100000') \triangleright \{ \langle d \rangle := \langle s \rangle \square_2 \langle t \rangle \} \}_1$

$(f_A = \square) \triangleright \{ (s < t) \triangleright \{ \langle d \rangle := 1 \} ? \{ \langle d \rangle := 0 \} \}$

Befehlszyklus: IF: $\langle pc \rangle := (\langle pc \rangle \mapsto \lceil \rceil) + 4$

ID: $O_A \triangleright \{ (f_B = '0000000') \triangleright \{ \square := \square_1 \}_1$

$(f_B = '0100000') \triangleright \{ \square := \square_2 \}_1$, Mxxxx := REGISTER $\}$

OF & EX: $\ll \gg := \langle s \rangle \square \langle t \rangle$ $(\langle s \rangle < \langle t \rangle) \triangleright \{ \ll \gg := \{ {0 \atop 1}$

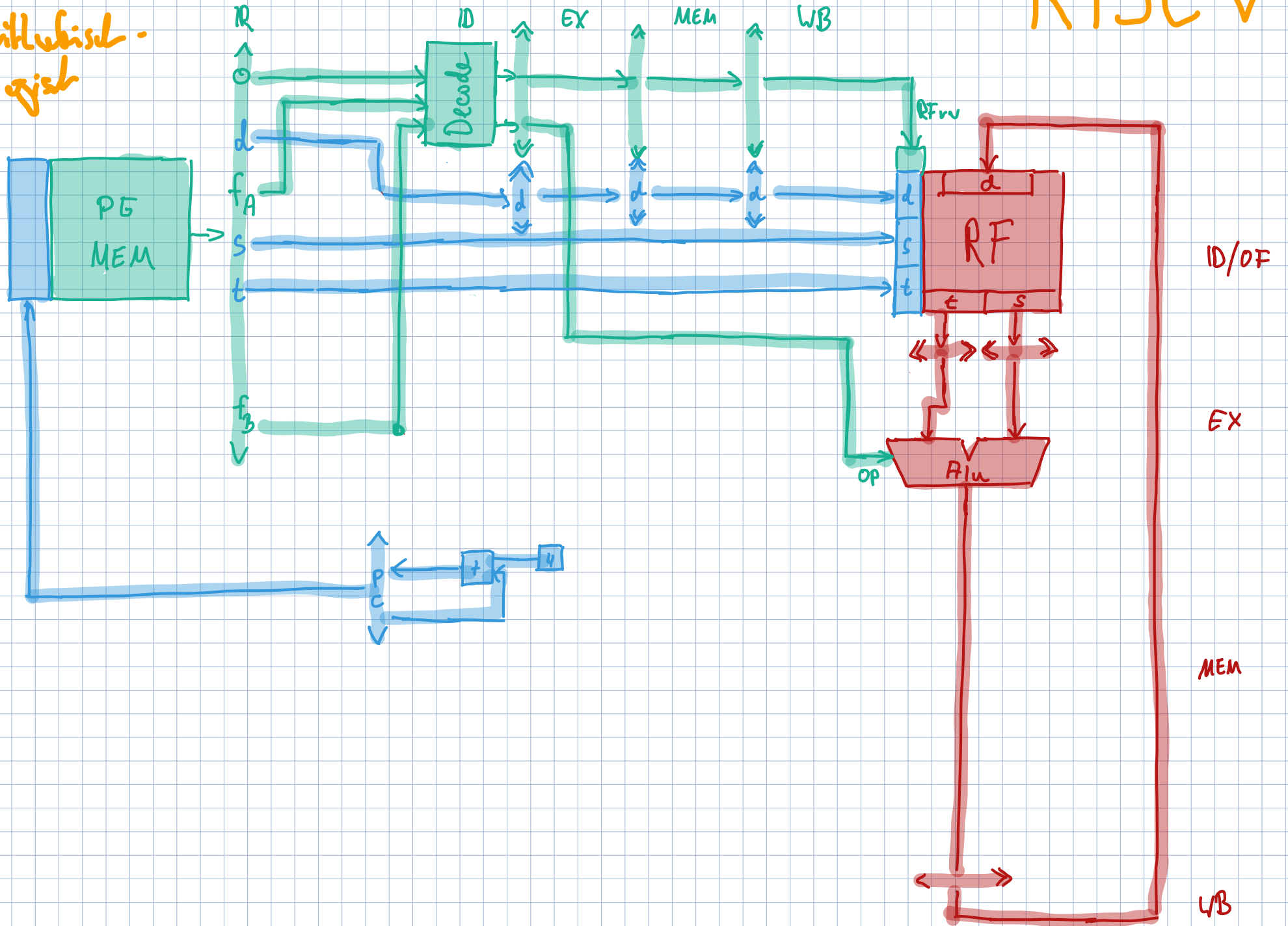
MEM:

OS: $\langle d \rangle := \ll \gg$

Mikroarchitektur : R-Format

Architektur-Ebene

RISC V



Arithmetisch-logische Befehle: I(mmediate)-Format

Verhalten: $\{(O = O_I) \triangleright \{(\square \neq \boxtimes) \triangleright \{ \langle d \rangle := \langle s \rangle \square \{\pm i_{11..0}\}_1$

$(\square = \boxtimes) \triangleright \{(\langle s \rangle \square \{\pm i\}) \triangleright \{ \langle d \rangle := (1 \,?\, 0)\}_1$

$(\square = \otimes) \triangleright \{(i^{31:7} = '0\,0000\,00') \{\langle d \rangle := \langle s \rangle \square_L \; m\}$,

$\{(i^{31:7} = '0\,1000\,00') \{\langle d \rangle := \langle s \rangle \square_A \; m\}\}\}$

Befehlsbyteloss:

IF: $\langle pc \rangle := (\langle pc \rangle \mapsto [\,]) + 4$

ID: $(O = O_I) \triangleright \{ MxAlu = IMMEDIATE \quad MxTD := IMMEDIATE\_T \}$

$MxByAlu = IMMEDIATE\}$

OF & EX: $MxSLEx := SE$ $\quad MxSLEx := SE$ $\quad MxSLEx := M$

$\ll \gg := \langle s \rangle \square \{\pm i\}_1 \quad (\langle s \rangle < \langle i \rangle) \triangleright \{\ll \gg := \{^{0}_{1}\}_1 \quad \ll \gg := \langle s \rangle \square s$

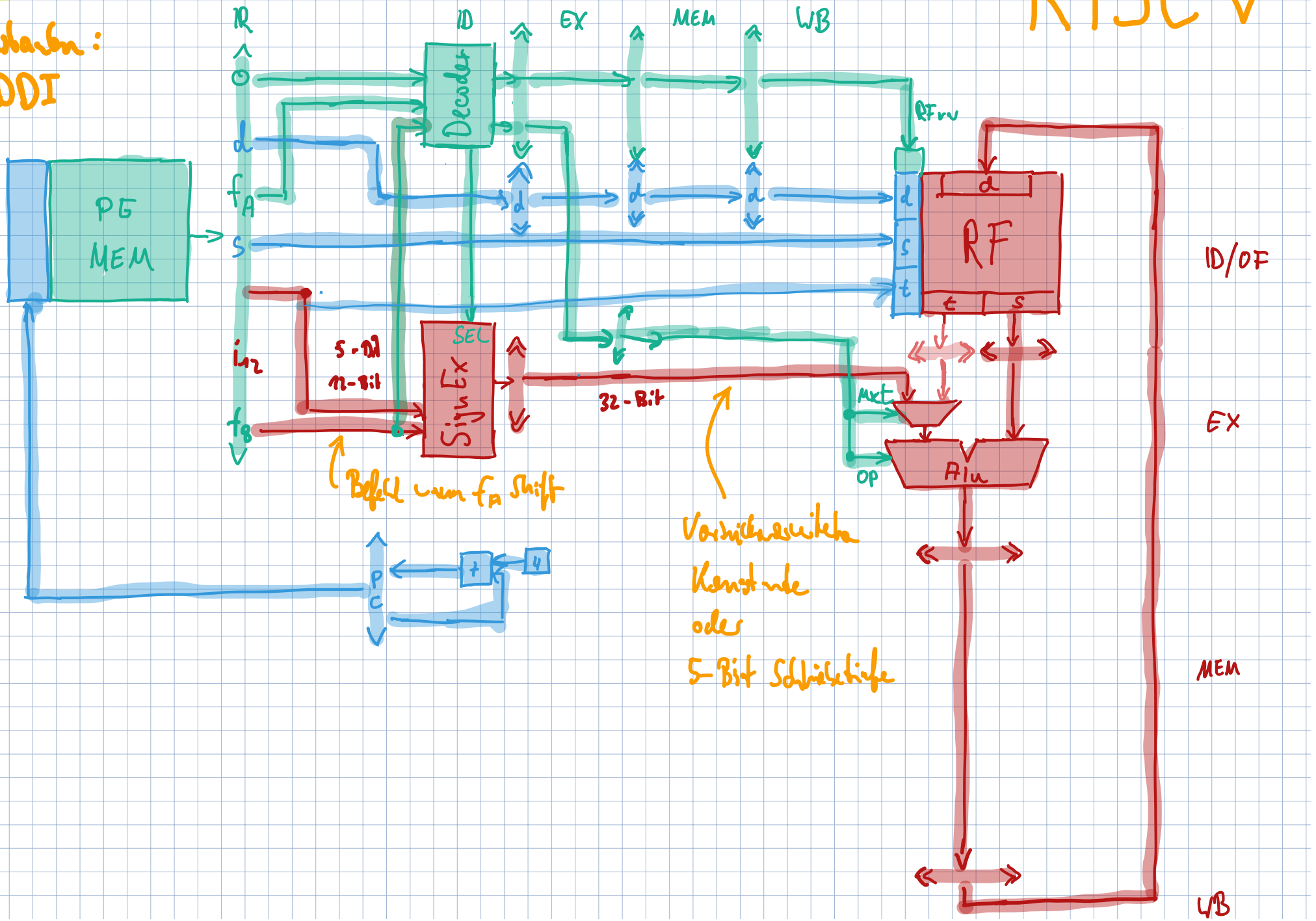
MEM:

OS: $\langle d \rangle := \ll \gg$

Microarchitektur I-Format

RISC V

Konstanten: ADDI



RISC V

Transport-Befehle: LOAD

[register-relative Addressing]

Variablen: $O_L \triangleright \{\langle d\rangle := [\langle s\rangle + \{\pm i\}]_{\ell[f]}\}$

Befehlszyklus:

IF: $\langle pc\rangle := (\langle pc\rangle \mapsto IM[\,]) + 4$

ID: $O_L \triangleright$

OF&EX: $\langle\langle x\rangle\rangle := \langle s\rangle + \{\pm i\}$

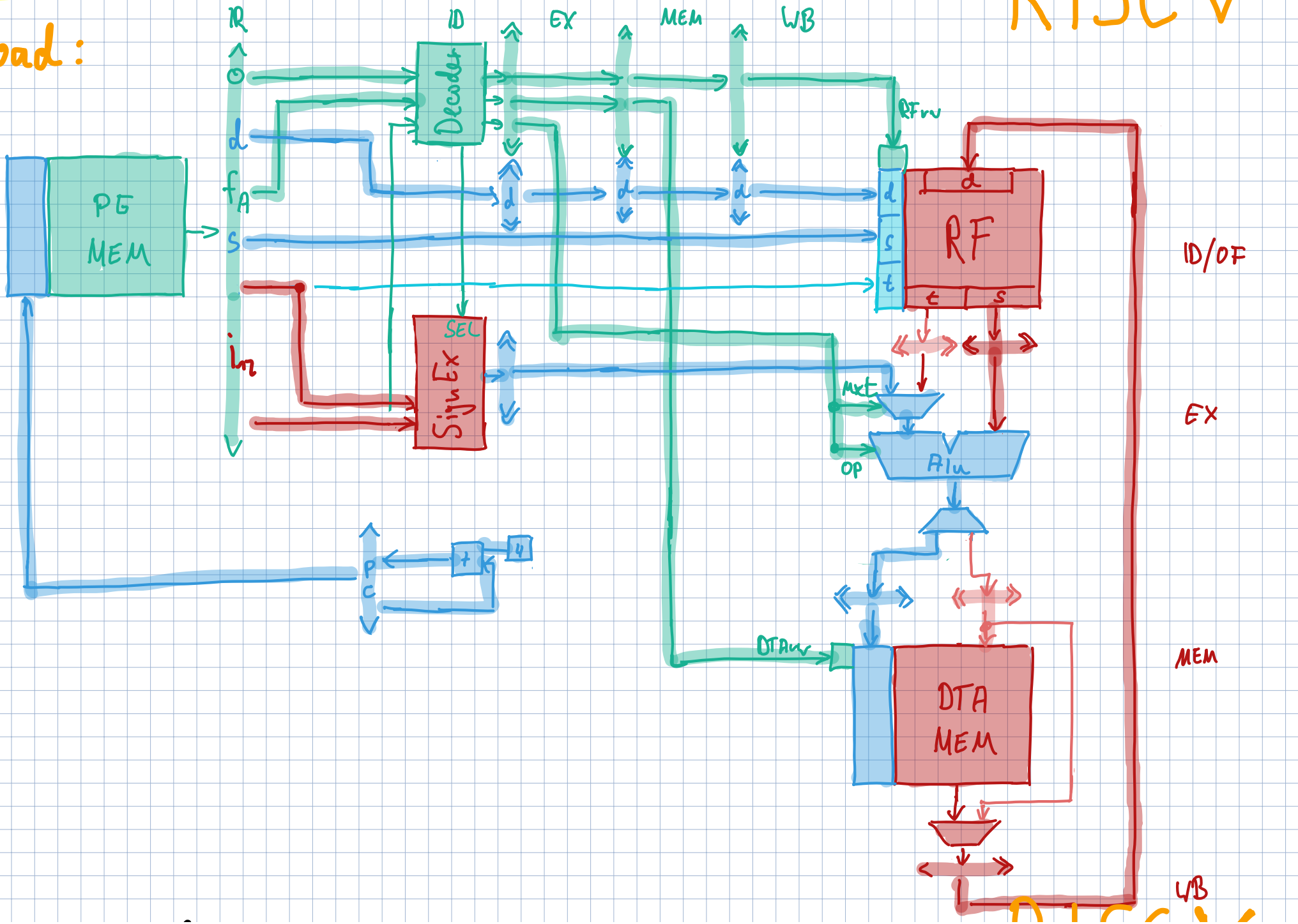
MEM: $\langle y\rangle := [\langle\langle x\rangle\rangle]\}_{\ell[f]}$

OS: $\langle d\rangle := \langle y\rangle$

Microarchitektur I-Format

RISCV

Load:



RISC V

Transport-Befehle: STORE

[register-relative Adressing]

Verhalten: $O_S \triangleright \{[<s> + \{\pm(i_A \mid i_B)\}]_{l[f]} := <t>\}$

Befehlszyklus:

IF: $<pc> := (<pc> \mapsto IM[]) + 4$

ID: $O_S \triangleright$

OF & EX: $\langle\!\langle\ \rangle\!\rangle := <s> + \{\pm(i_A \mid i_B)\}$

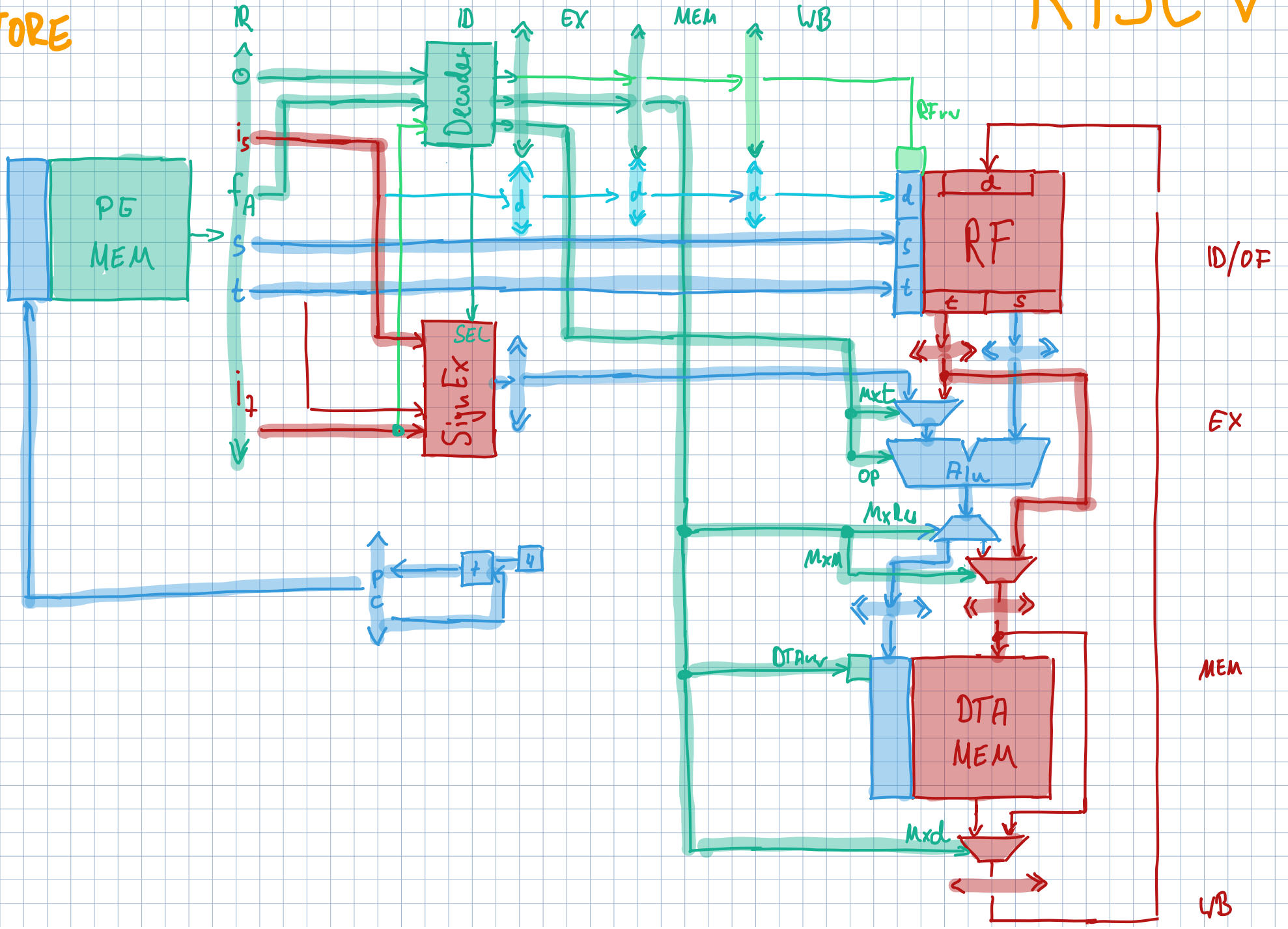
MEM: $[\langle\!\langle\ \rangle\!\rangle] := t_{l(F_B)}$

OS: $\{\}$

Microarchitecture S-Format

STORE

RISC V



Verzweigung (Bedingter Sprung)

RISC V

[Logisch - relative Adressierung]

Verhalten: $\{(O^3 = O_B) \triangleright \{ (\langle S \rangle \square \langle d \rangle) \triangleright \{ \langle pc \rangle := \langle pc \rangle + \{\pm i\}\}\}\}$

Befehlszyklus:

IF: $\langle pc \rangle := (\langle pc \rangle \mapsto IM[\,]) + 4$

ID: $O_B \triangleright$

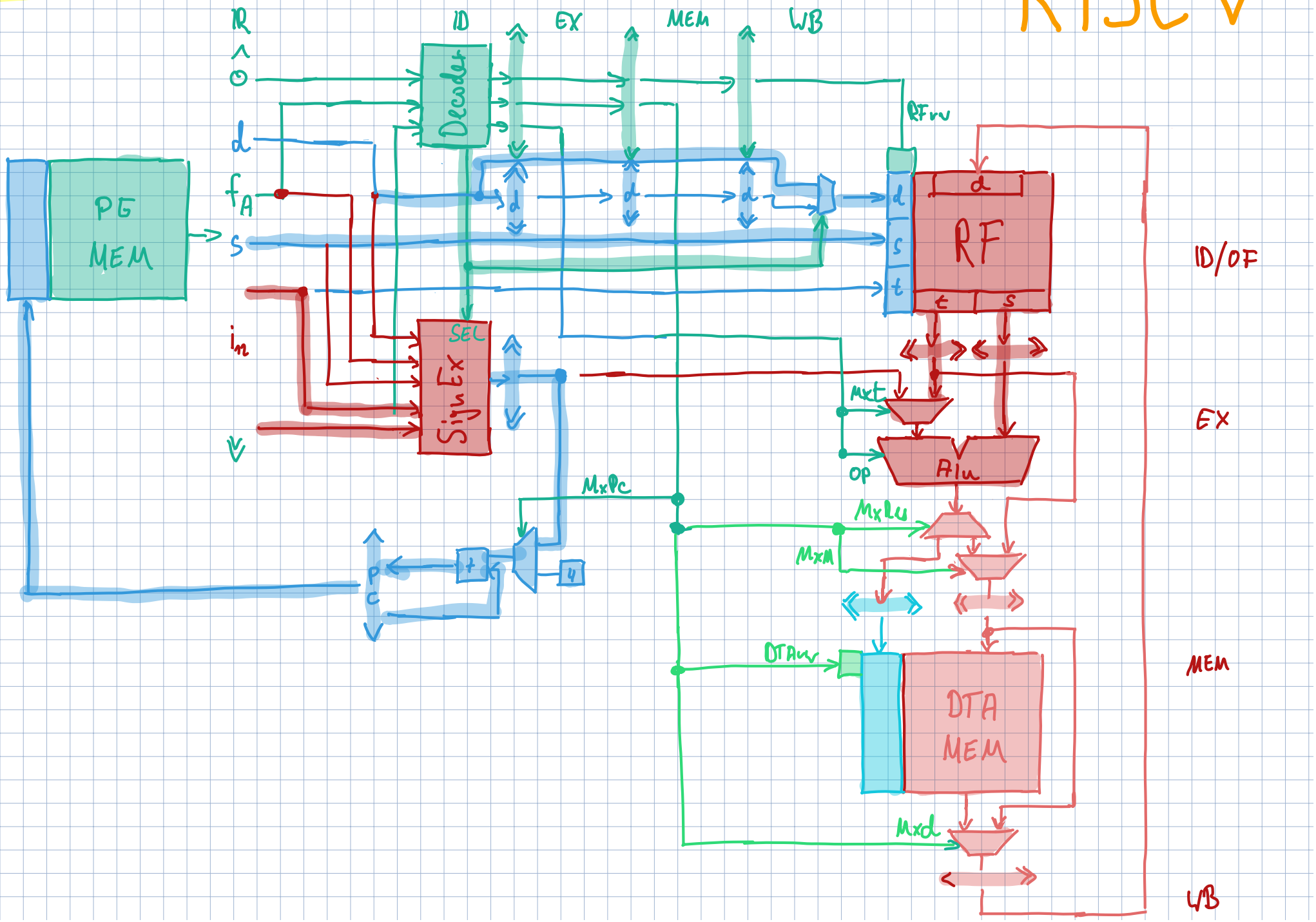
OF & EX: $\langle S \rangle \square \langle d \rangle, \langle\langle\ \rangle\rangle := \langle pc \rangle + \{\pm i\}$

MEM: $\langle pc \rangle := \langle\langle\ \rangle\rangle$

OS: $\{\}$

Microarchitecture

RISC V



Unbedingte Sprünge (Unconditional jumps)

RISC V

[register-relative Adressierung]

Struktur: I(mmediate)-Format:

$O_J = '1100111'$

$\{ i^{12}, \langle s \rangle^5, F^3, \langle d \rangle^5, O^7 \}_V$

$\{ i^{12}, \langle s \rangle^5, '000', \langle d \rangle^5, '1100111' \}$

$B ::= \{ jalr \}$ jump and link register

$B' ::= \{$

$\square ::= \{\}$

- Format

Verhalten: $O_J \triangleright \{ \langle d \rangle := \langle pc+4 \rangle ; \langle pc \rangle := \langle s \rangle + \{ \pm i \} \}$

Unbedingte Sprünge (Unconditional jumps)

RISC V

[spider-direkte Adressierung]

Struktur: **U (unconditional) J (jmp)-Format:**

$\{ i^{20}, \langle d \rangle^5, 0^7 \}_V$

$\{ i^{20}, \langle d \rangle^5, '1101111' \}$

$O_J = '1101111'$

$B := \{ jal \}$ jump and link

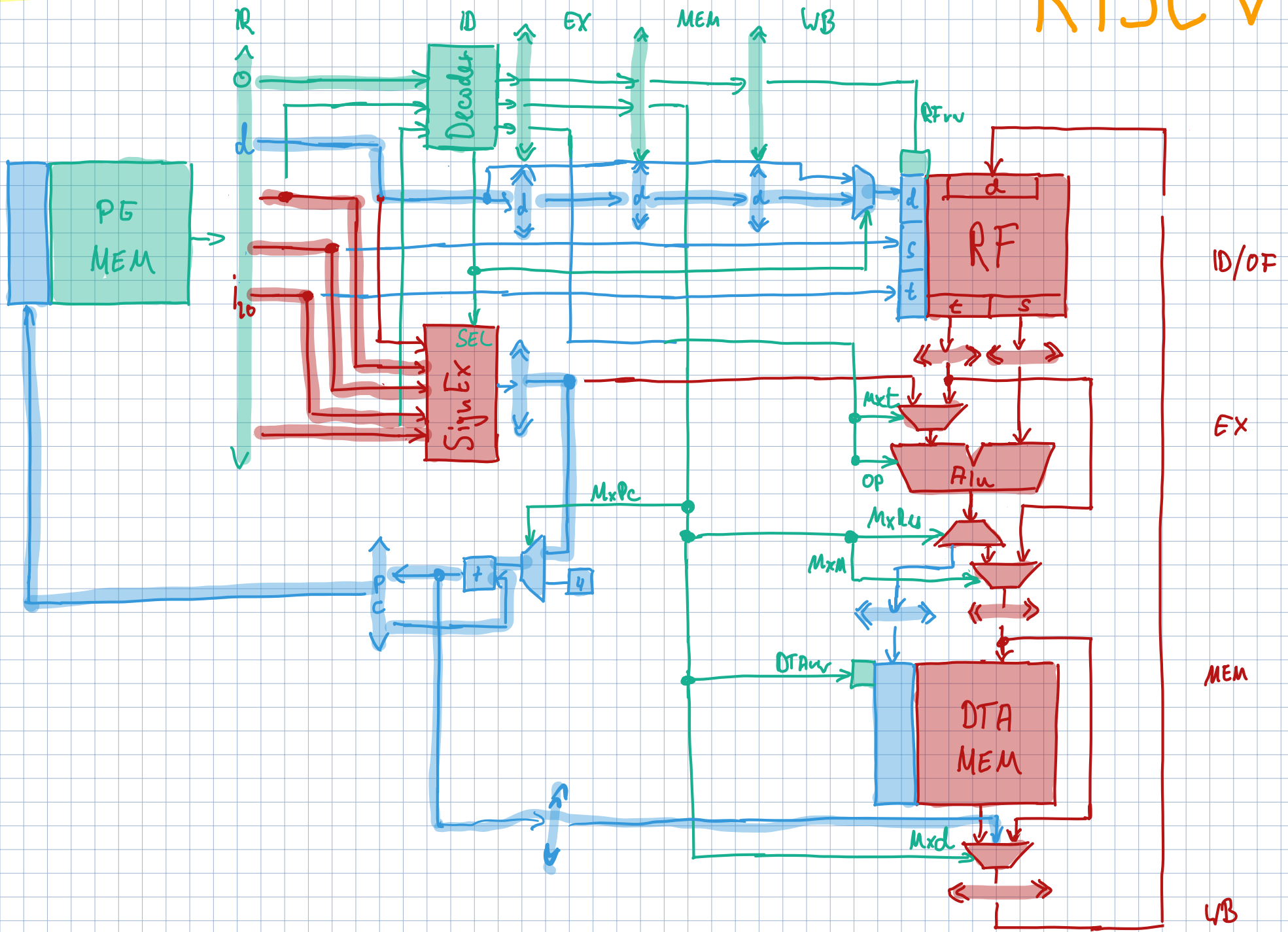
$B' := \{\}$

$\square := \{\}$

Verhalten: $O_J \triangleright \{ \langle d \rangle := \langle pc+4 \rangle ; \langle pc \rangle := \{ \pm i \} \}$

Microarchitecture

RISC V



Übersicht Befehlsformate

RISC V

R-Format: $\{ f_B^7, \langle t,s \rangle^{10}, f_A^3, \langle d \rangle_1^5, '0110011' \}$ — Add, And

I-Format: $\{ i_B^{12}, \langle s \rangle^5, f_A^3, \langle d \rangle^5, '0010011' \}$ — Addi, Ori

$\{ i_B^{12}, \langle s \rangle^5, f_A^3, \langle d \rangle^5, '0000011' \}$ — Load

S-Format: $\{ i_B^7, \langle t,s \rangle^{10}, f_A^3, i_A^5, '0100011' \}$ — Store

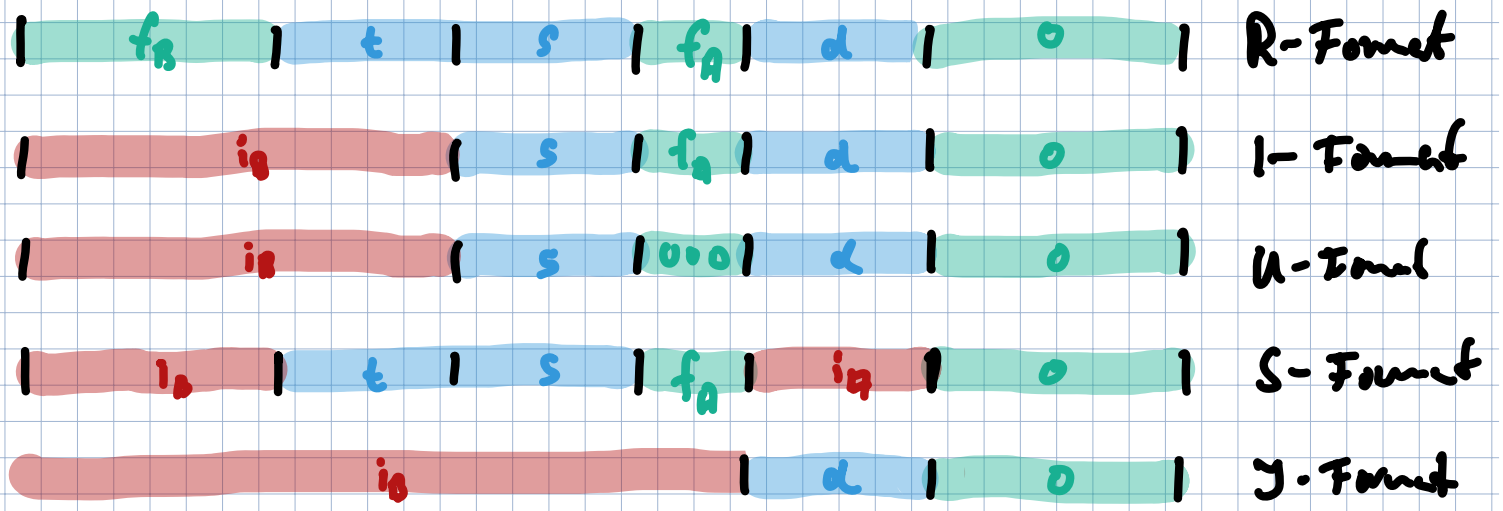
SB-Format: $\{ i_B^7, \langle t,s \rangle^{10}, f_A^3, i_A^5, '1100011' \}$ — Branch

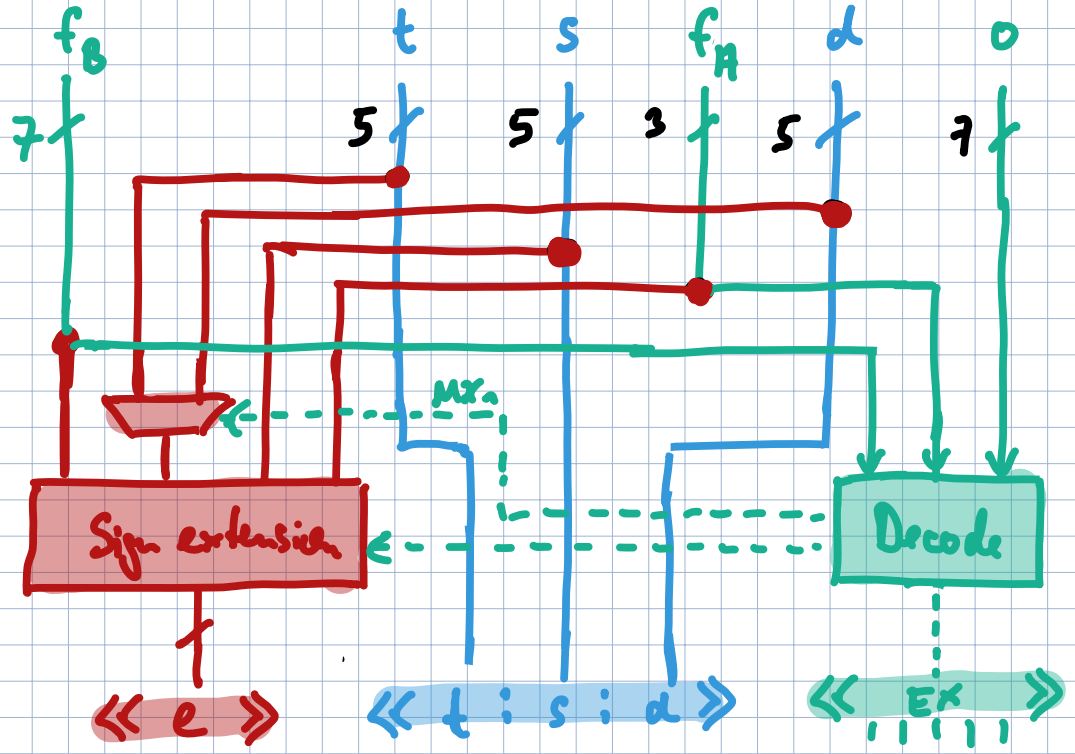
$\{ i^{12}, \langle s \rangle^5, '000', \langle d \rangle^5, '1100111' \}$ — Jump & Link Register

UJ-Format: $\{ i^{20}, \langle d \rangle^5, '1101111' \}$ — Jump & Link

Befehlsdekodierung:

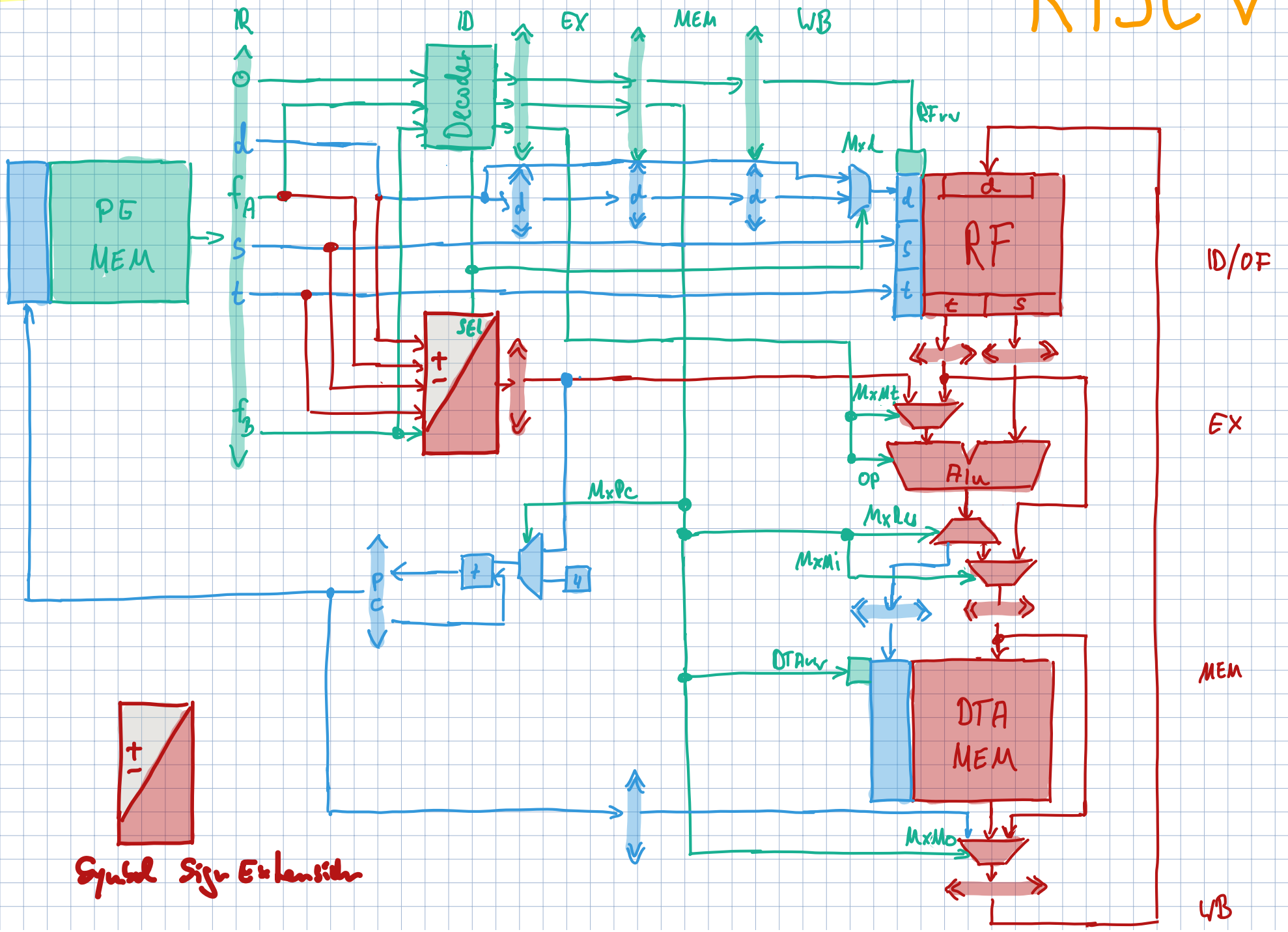
RISC V





Microarchitecture

RISC V



Steuerwerk

Multiplexer des Rechenwerkes:

MxPc Verzweigungsadresse in den Befehlszähler (PC)

Mxd Adresse Zielregister (D) wählt ins Flipflopband

MxMi Multiplexer vor dem Speicher (MemoryIn)
MxMo -"- Lücke -"- (MemoryOut)

AluOut Adresse für Datenoperationen

MxMt Konstante oder zweites Quellregister (T) in die linke ALU-Eingang

Lese- und Schreiben Register & Datenspeicher

RtRW Schreiben/Lesen Registerdatei

DtRW -"- Speicher

Befehlssatzarchitektur

Befehlswörter

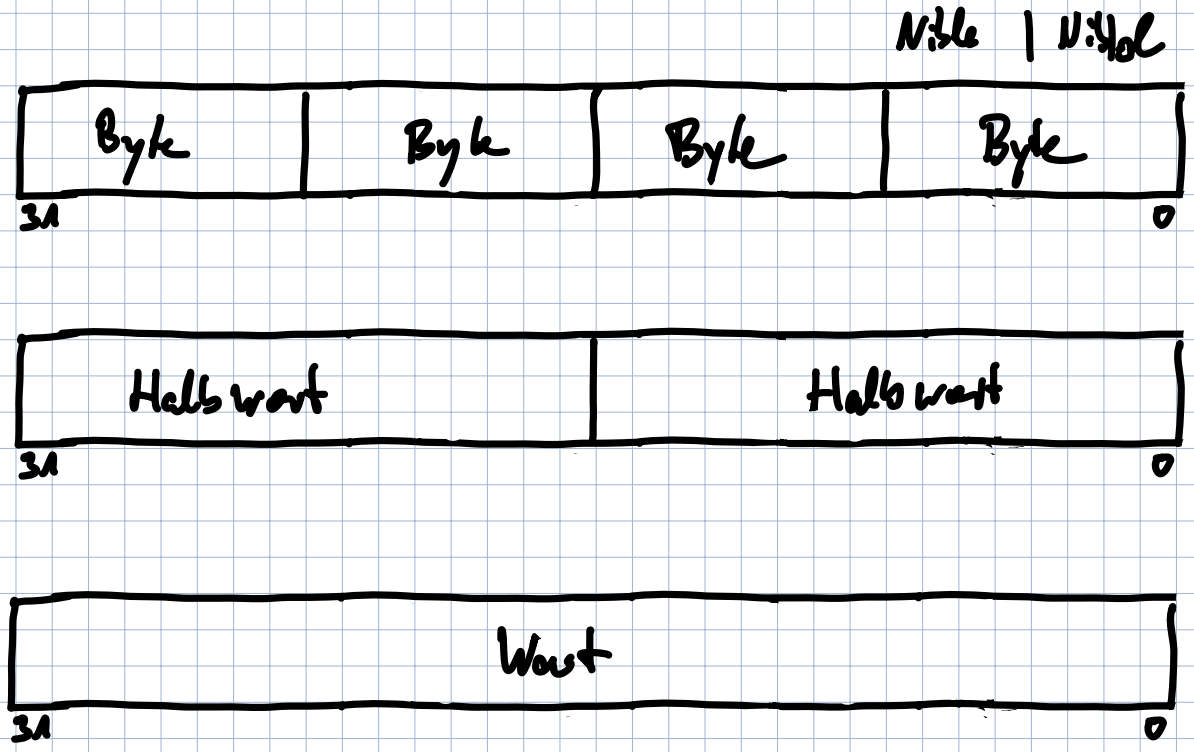
ISA - MIPS

ISA - RISC V

Mikroarchitektur

RISC-V

Daten-Ausrichtung (Data Alignment)



Daten-Ausrichtung (Data Alignment)

Byte-Reihenfolge (Endianness)

Big Endian "Big Starting"

Adresse	Datum
⋮	
...x 0100	5
...x 0101	A
...x 0110	F
...x 0111	B
⋮	

⋮	
5AFB	0
⋮	+4

Little Endian "Little Starting"

Adresse	Datum
⋮	
...x 0100	B
...x 0101	F
...x 0110	A
...x 0111	5

⋮	
BFA5	0
⋮	+4