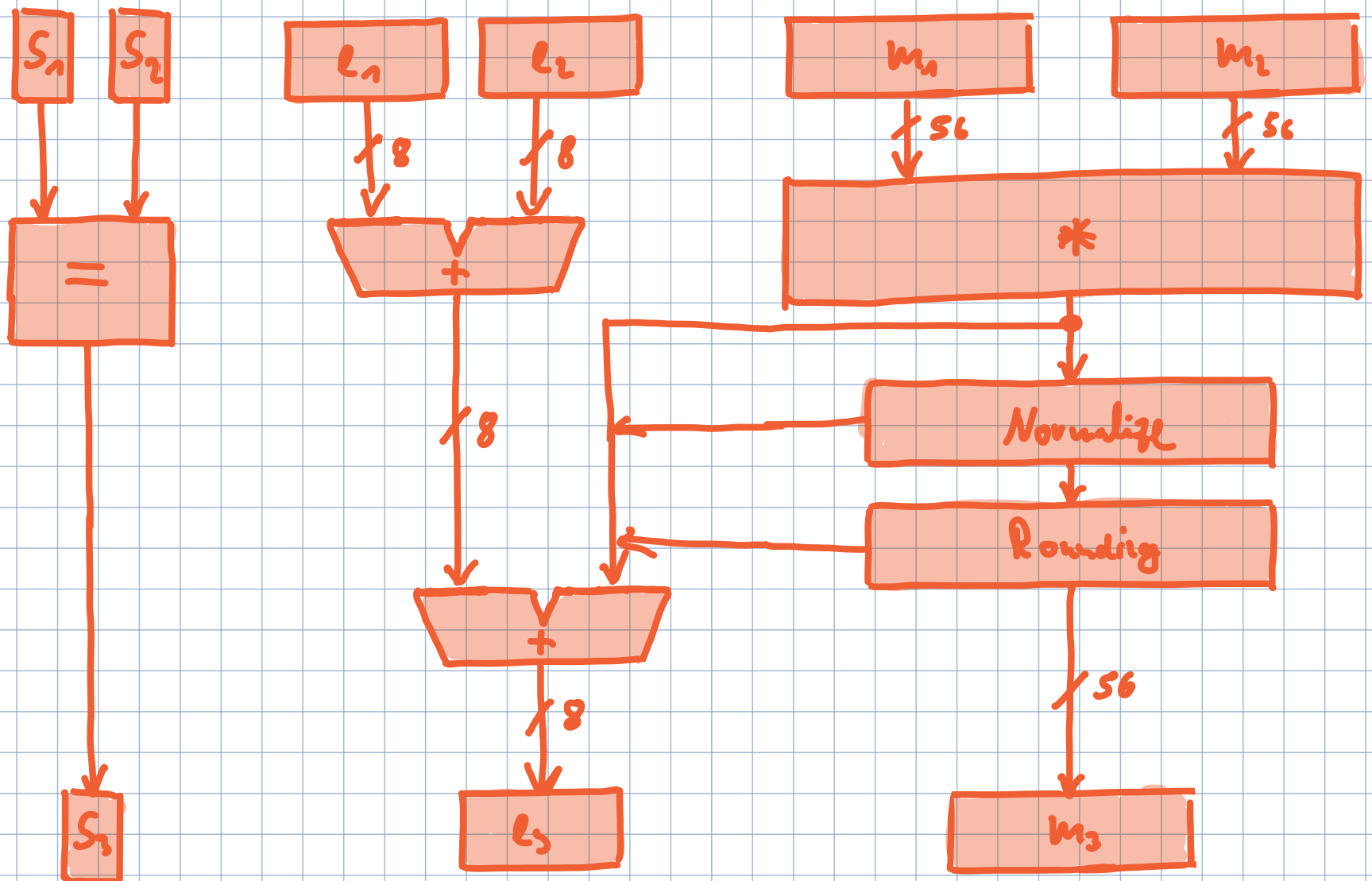
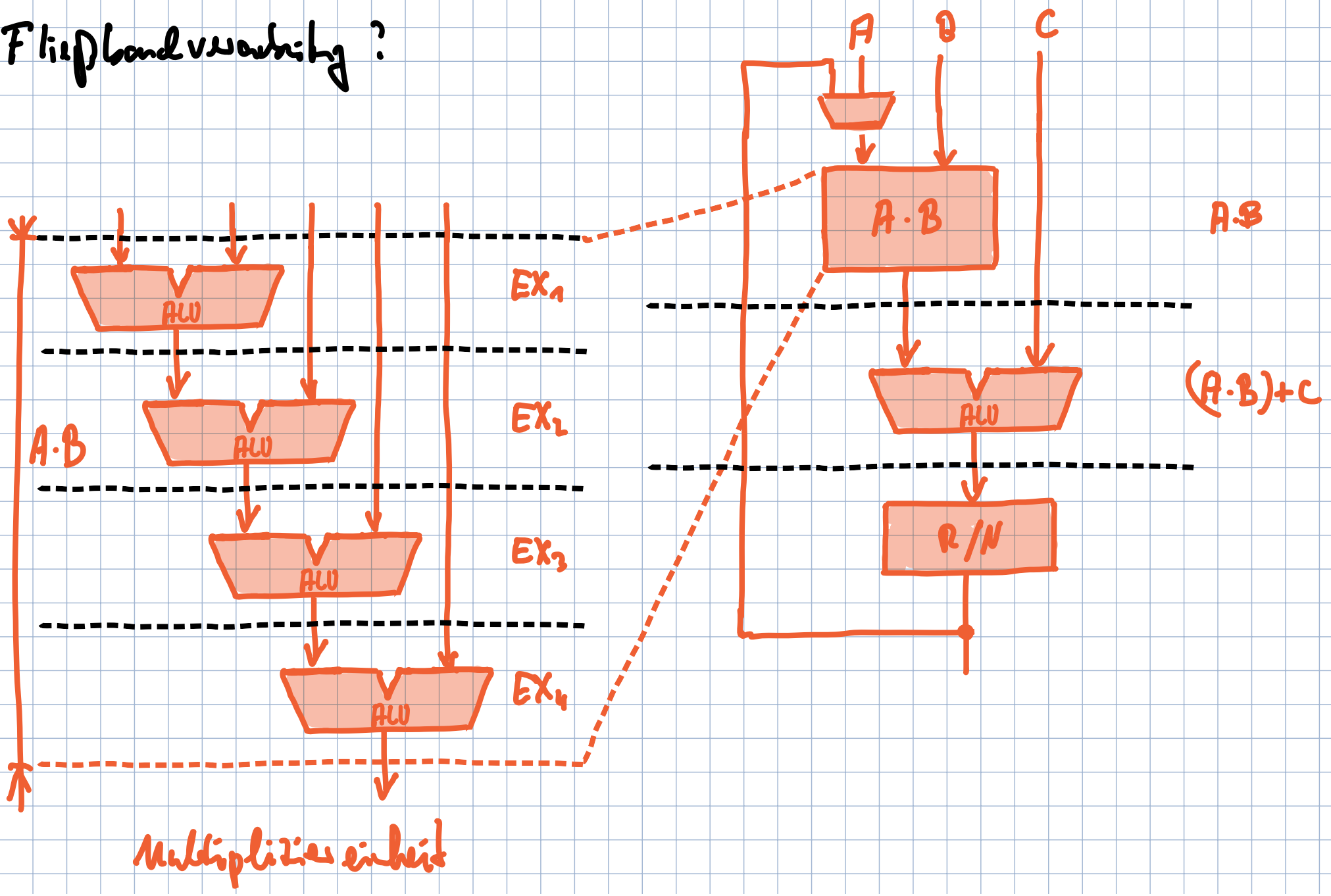
Fließkommma - Einheit



Unterschiedliche Ausführungszeiten (Fließbandmaschinen)

Fließbandverarbeitung:

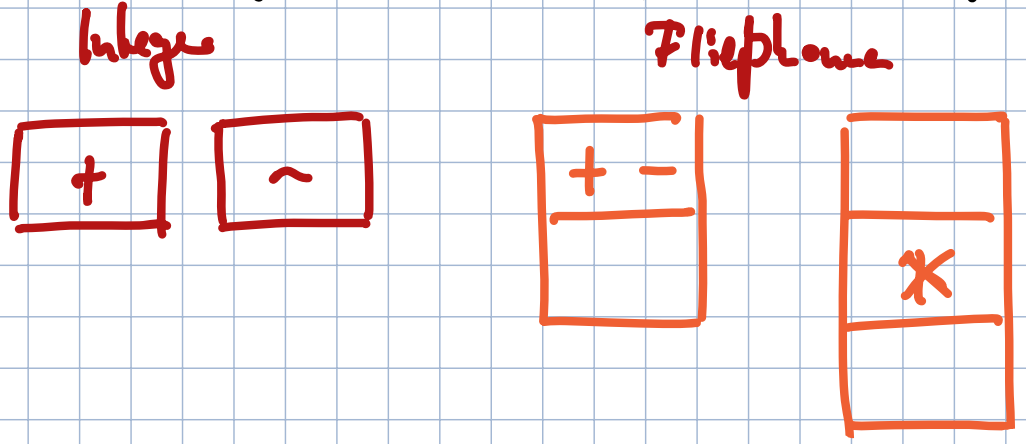


Maßnahmen zur Verbesserung der Performance

Instruction Level Parallelism (ILP)

i) Erhöhung des Durchsatzes durch Nebenläufigkeit

ii) Unterschiedliche Fließbandlängen führen zu dynamischer Fließbandverwaltung.



Ungleiche Ausführungspfade in Execute

iii) Dynamische Betrachtung der Fließband Hemmnisse ⟶ Umbenennen der Register

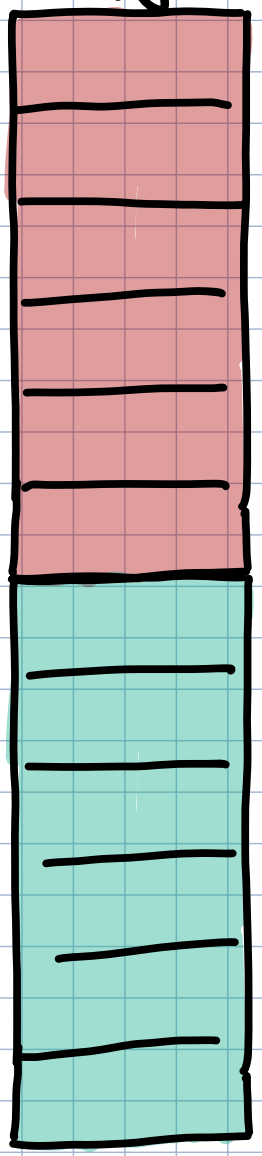
Register-Renaming

iv) Leerlauf der Fließbandverwaltung vermeiden (Sprungvorhersage)

v) Leerlauf durch Zugriffsfehler vermeiden

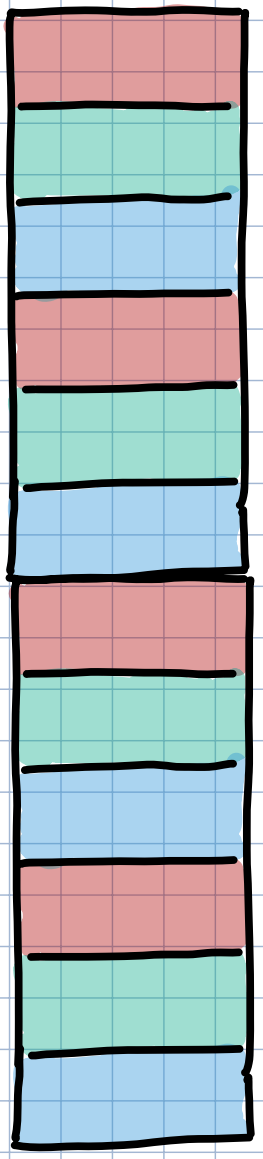
Nebenläufigkeit

keine Nebenläufigkeit



Sequentiell Ausführung

virtuelle Nebenläufigkeit

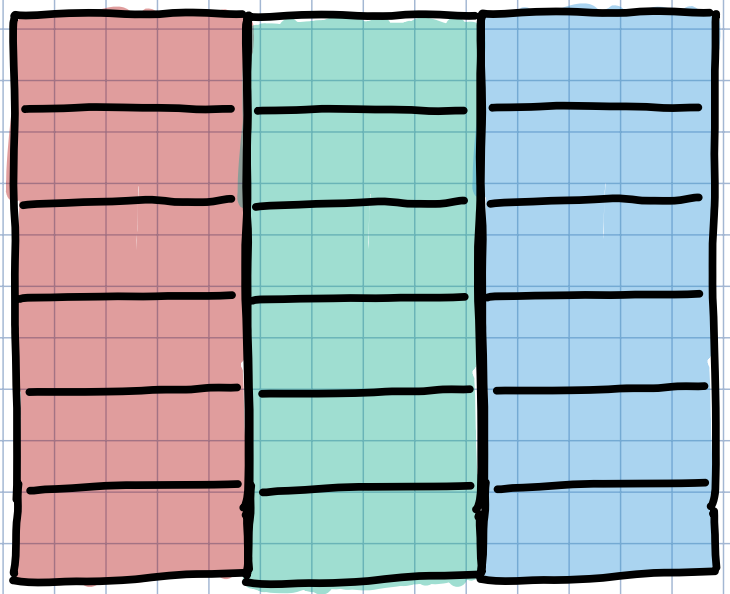


überlappende Sequentiell Ausführung

CPI < T erfordert Nebenläufigkeit

$IPC = \frac{1}{CPI}$

echte Nebenläufigkeit

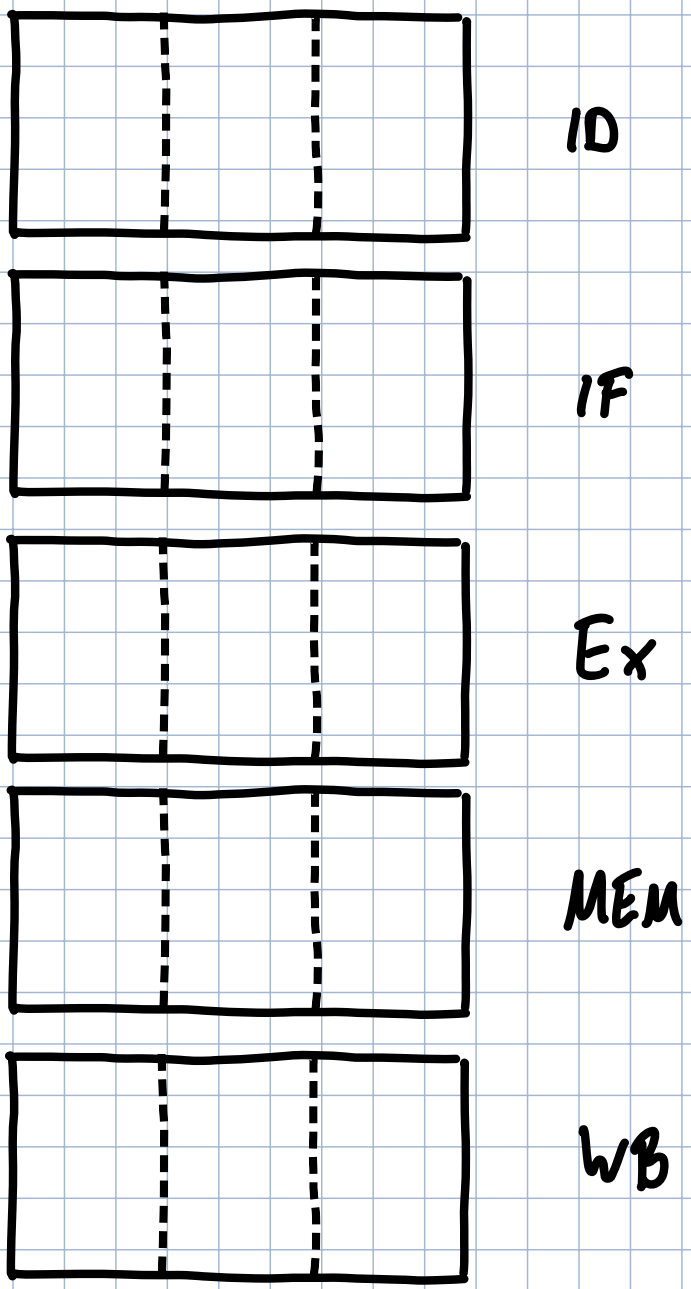


gleichzeitige nebenläufige Ausführung

Busy

Motivation

Jede Phase enthält mehrere Ressourcen, so dass echte Nebenläufigkeit entsteht.



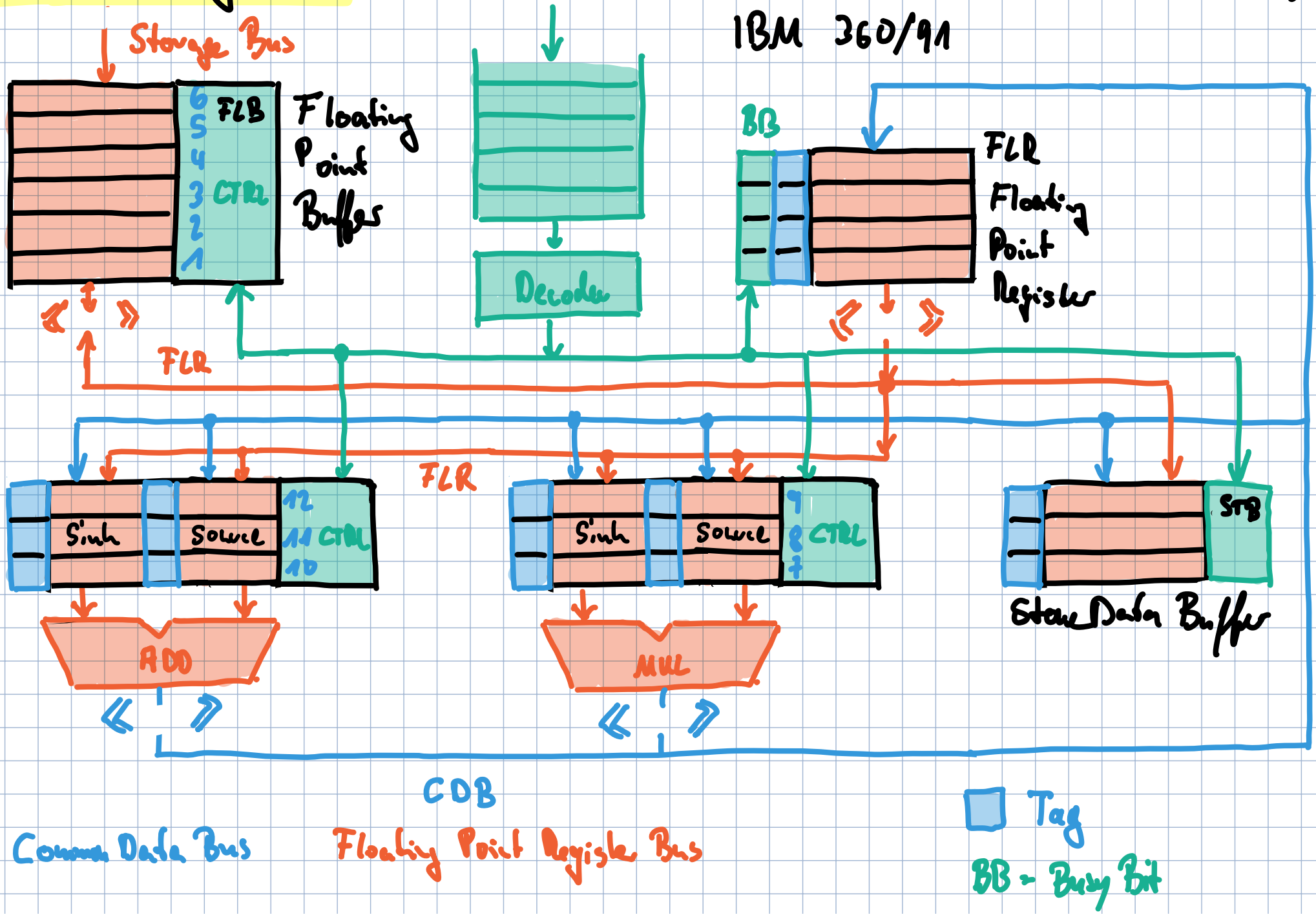
Probleme?

Tomasulo - Algorithms

Tomasulos Algorithmus

Robert Tomasulo (1934 - 2008)

IBM 360/91



Fehlen (Abbruch) Tomasulo's Algorithmus

RAW - Hemmisse und WAR - Hemmisse

i) Befehl ans Warteschlange (Dispatch)

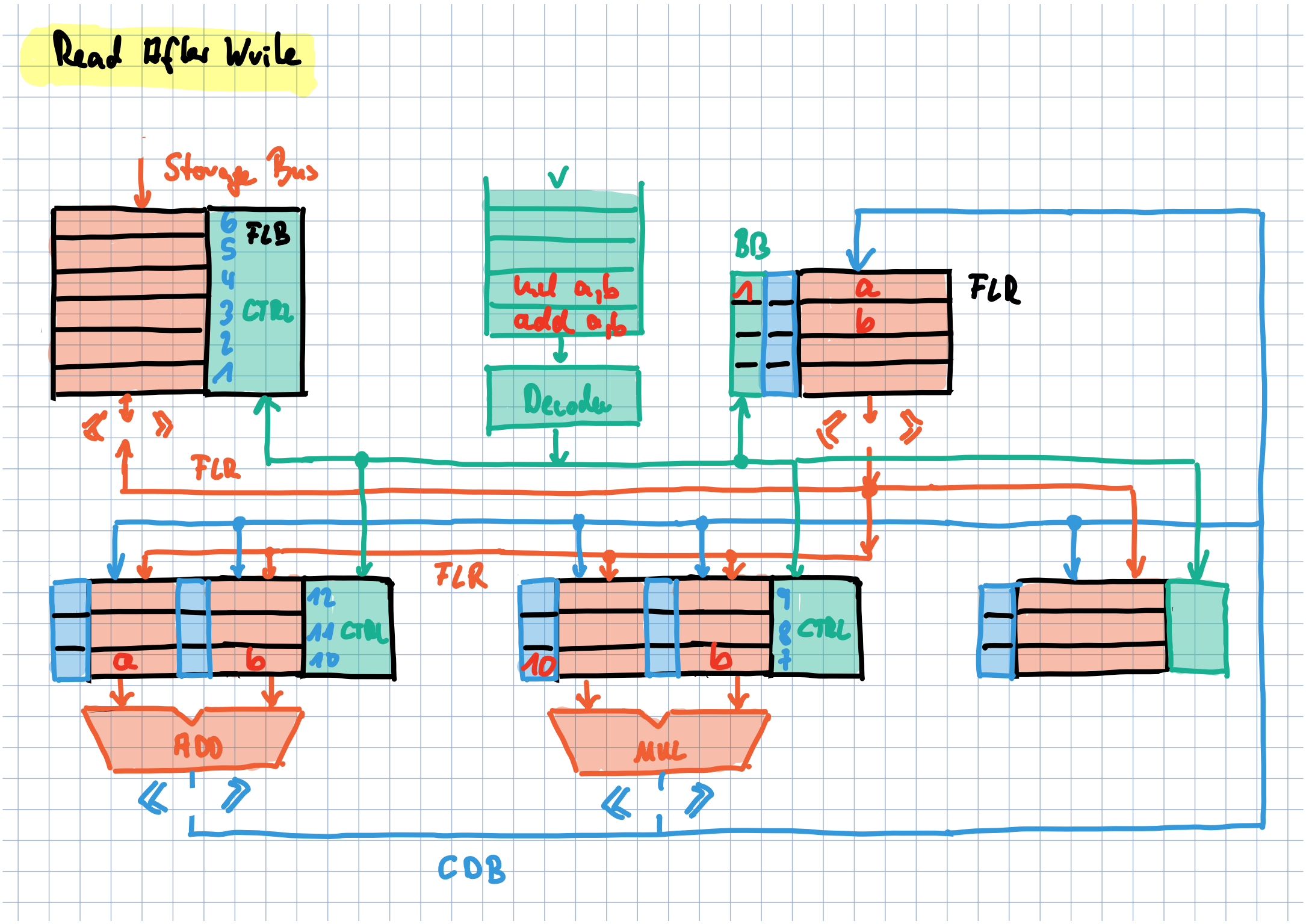
ii) Register lesen (OF)

ii a) Push in RF (EX)

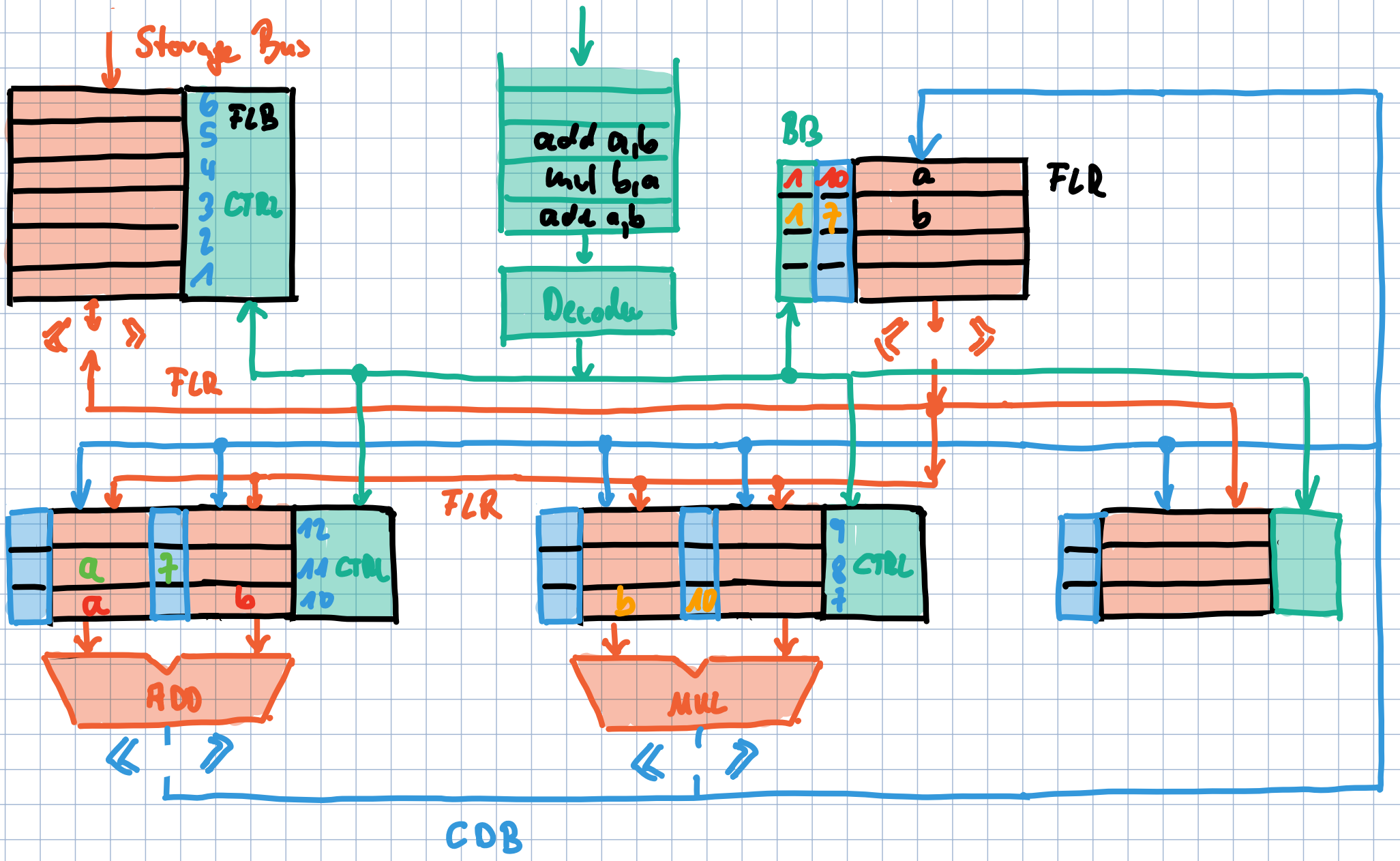
b) Befehl in RS und im (OF)
RF auf beobachtet setzen

iii) Warte bis Folgebefehl Variable schreibt

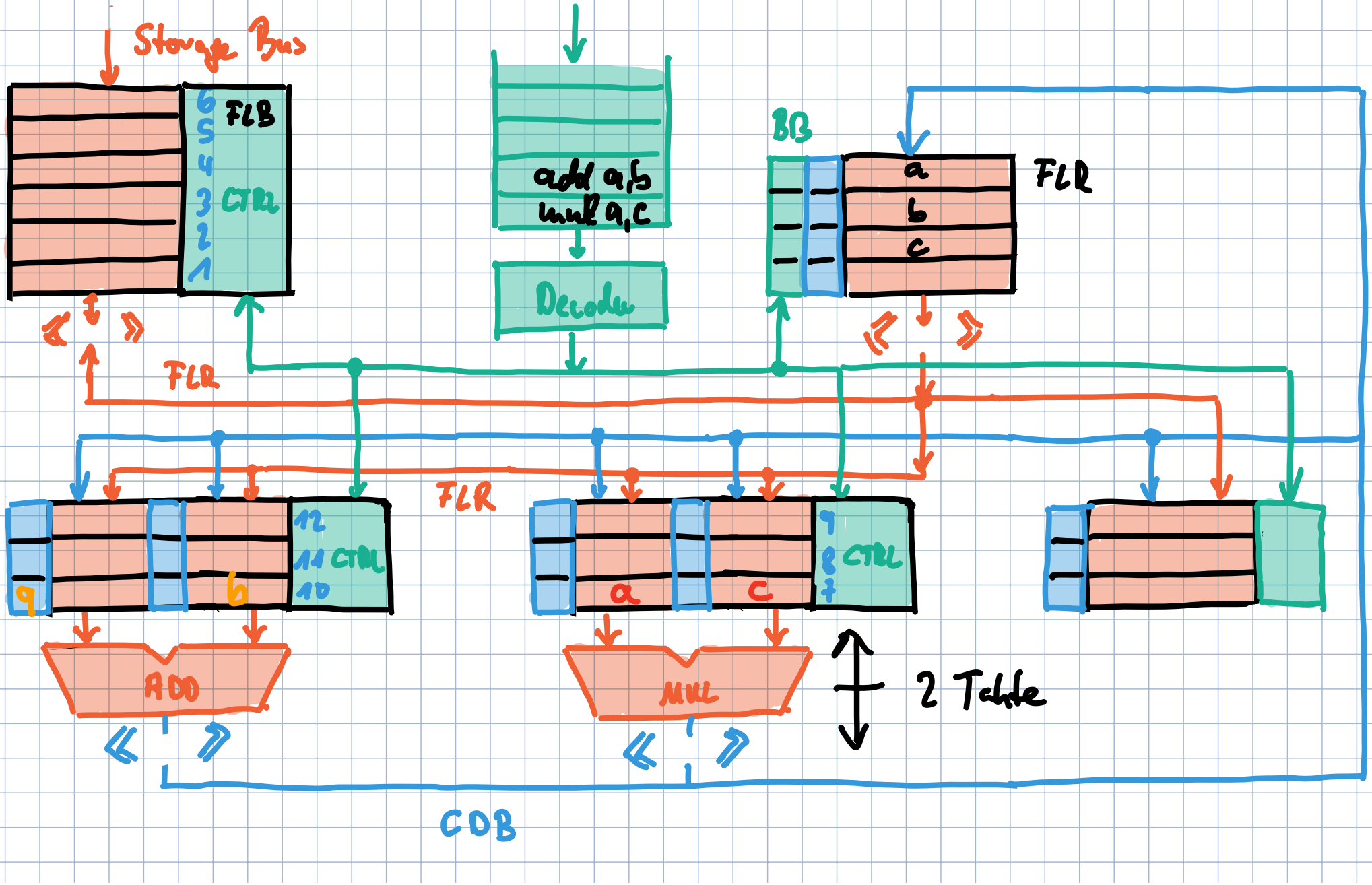
iv) Ergebnis in ROB


Read After Write
Storage Bus
FLB
6
5
4
3
2
1
CTRL
mul a,b
add a,b
Decode
BB
1
a
b
FLR
FLR
FLR
12
11
CTRL
10
a
b
10
b
9
8
7
CTRL
ADD
MUL
CDB

Write after Read (WAR)



Write After Write (WAW)



Register Renaming

Register - Umbenennung

Aufbau von RAV-Halbwerks

a: $R_4 = R_0 + R_8$

b: $R_2 = R_0 \cdot R_4$

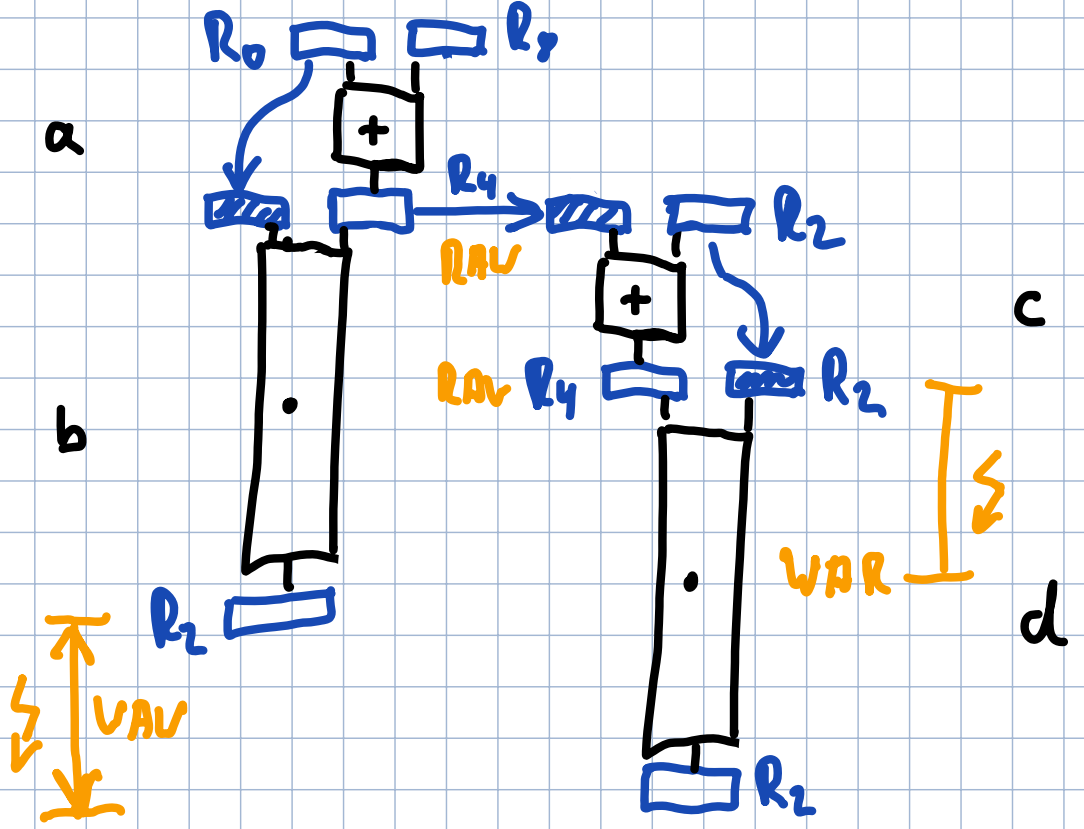
c: $R_4 = R_4 + R_2$

d: $R_2 = R_4 \cdot R_2$

RAV

VAV

VAR



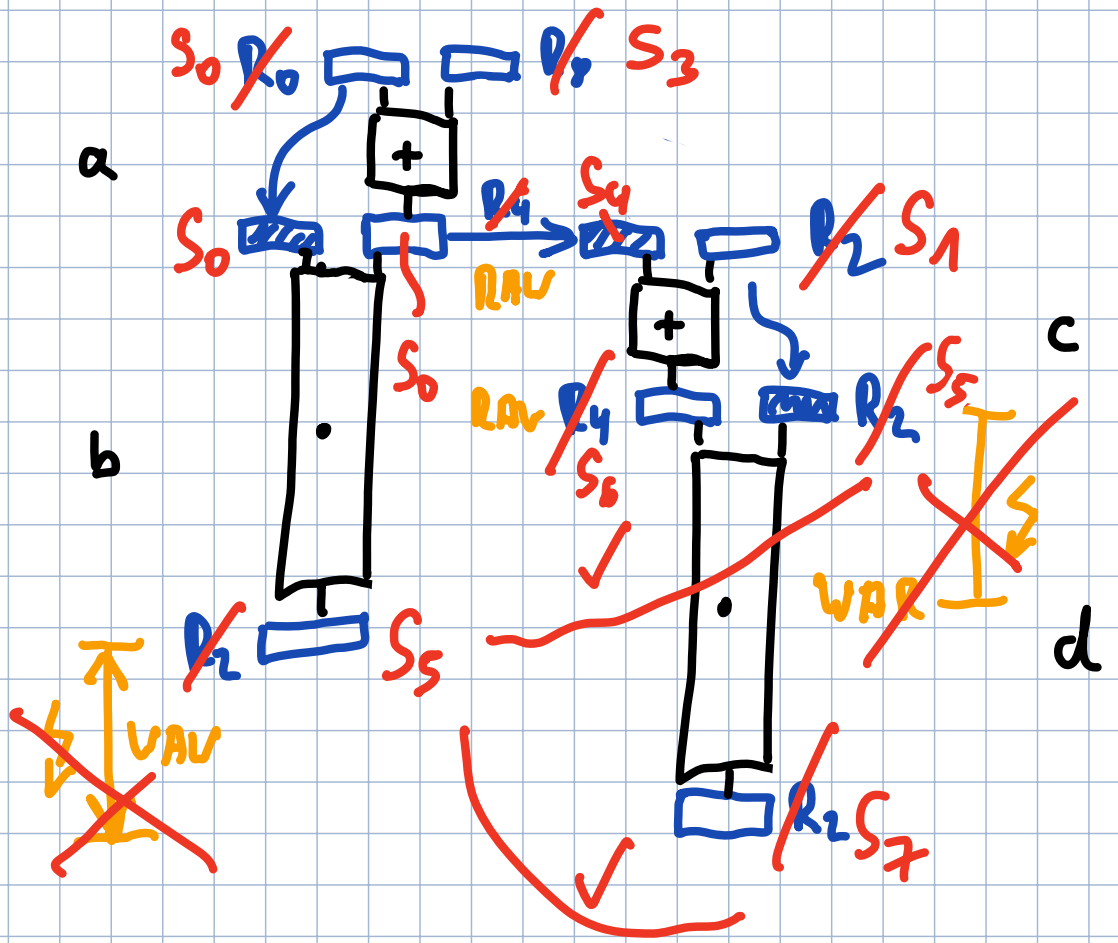
Auflösen von RAW-Hazards: Registerumbenennung (Register Renaming)

a: $R_4 = R_0 + R_8$ (S_0, S_3)

b: $R_2 = R_0 \cdot R_4$ (S_2, S_0, S_4) — RAW

c: $R_4 = R_4 + R_2$ (S_6, S_4, S_2) — WAW, VAR

d: $R_2 = R_4 \cdot R_2$ (S_7, S_6, S_5)



Auflösen von RAW-Hazards: Registerumbenennung (Register Renaming)

Originaler Code

R_0 R_2 R_4 R_8

S_0 S_1 S_2 S_3

Code mit umbenannten Registern

a: $R_4 = R_0 + R_8$ — S_4 — $S_4 = S_0 + S_3$

b: $R_2 = R_0 \cdot R_4$ — S_5 — $S_5 = S_0 \cdot S_4$

c: $R_4 = R_4 + R_2$ — S_6 — $S_6 = S_4 + S_5$

d: $R_2 = R_4 \cdot R_2$ — S_7 — $S_7 = S_6 \cdot S_5$

Registerumbenennung:

Auflösen der Hemmnisse durch Umbenennen der Register

Architekturregister $r \in R$ Physische Register $\nu \in R'$

i) Jedes Register eines Codeblocks bekommt einen neuen Bezeichner

$$r_i = \nu_i$$

ii) Ist ein Register in einer Codezeile ein Zielregister erhält dieses einen neuen Namen. Das Quellregister bekommt den physischen Namen ν' des umbenannten Registers.

$$\nu'_{neu} = \nu'_n \circ \nu_n' \quad ?$$

iii) Umbenennen der Register $R_0 = \{r_{n_1}, \dots, r_{n_m}\}$

$R'_0 = \{\nu_{n_1}, \dots, \nu_{n_m}\}$

iv) Umbenennen des Zielregisters $\nu_{neu} = R_n \times R_n$ $R_n' = \{\nu_{n_1}, \dots, \nu_{n_{i-1}}, \nu_{n_{i+1}}\}$

Modifikation der Fließbandstufen

Instruction Buffer

Dispatch Buffer

Issueing Buffer

Completion Buffer

Store Buffer

Fetch

Decode

Dispatch

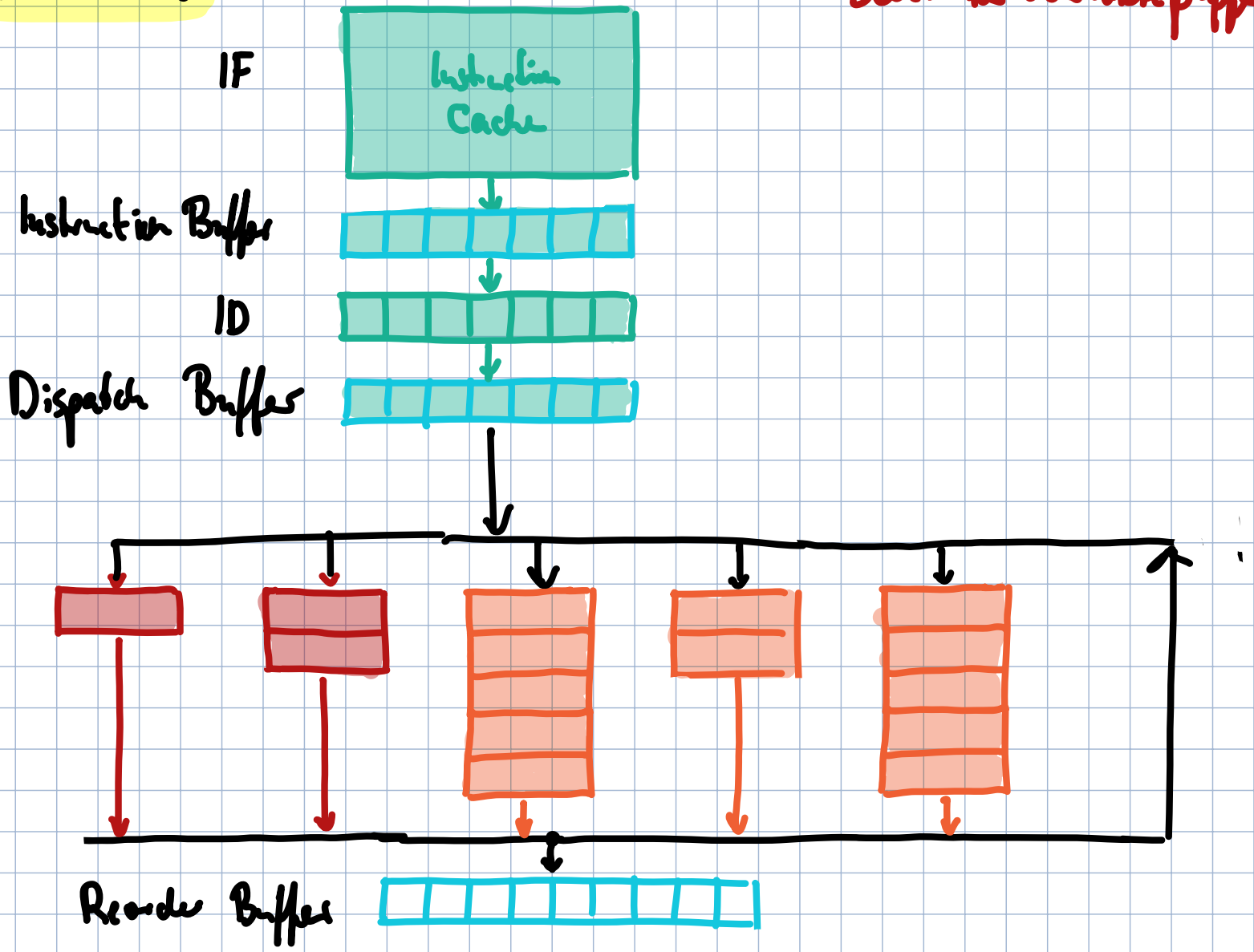
Execute

Complete

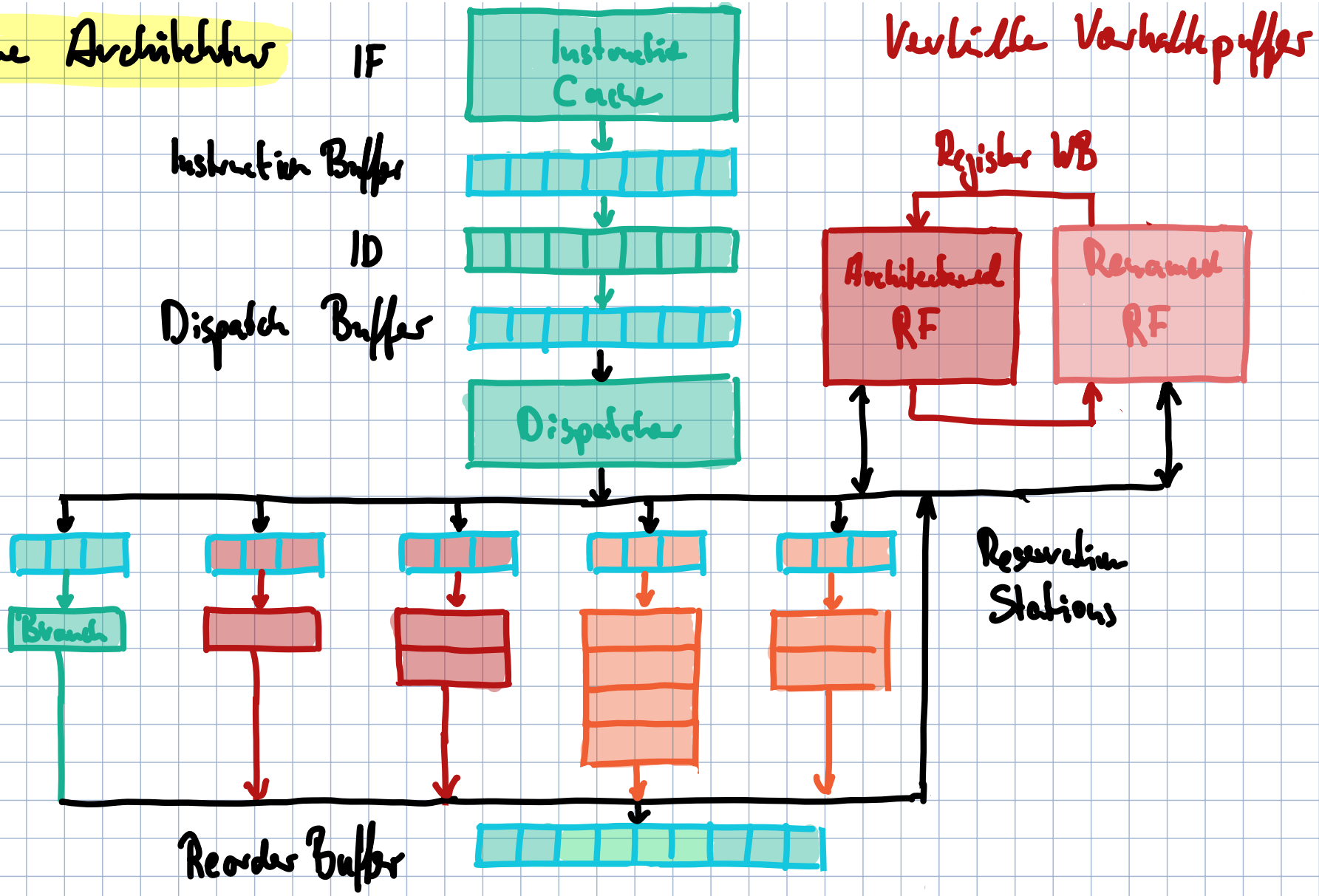
Retire

Superskalare Architekturen

Zentraler Verhaltspuffer

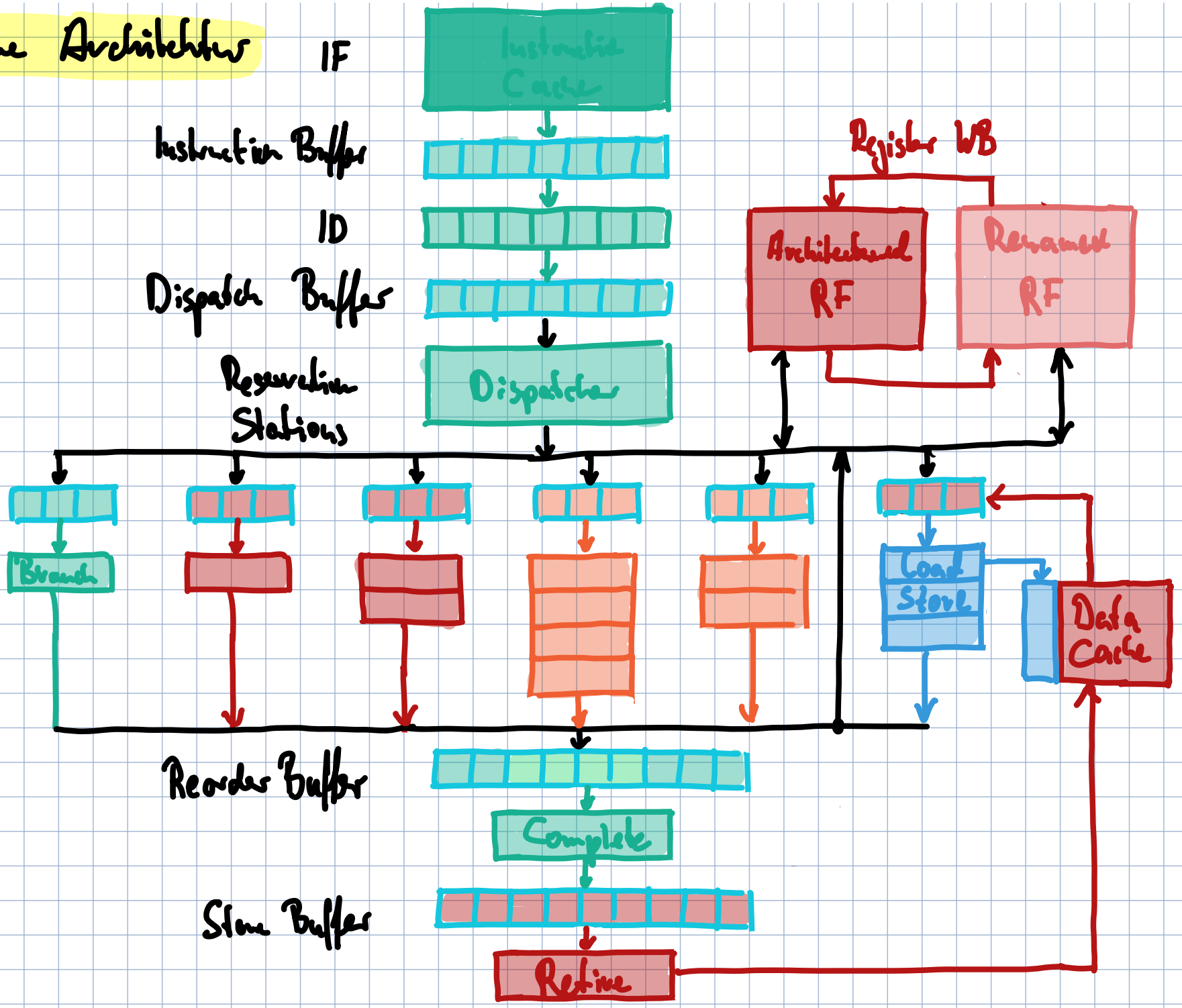


Superskalare Architektur



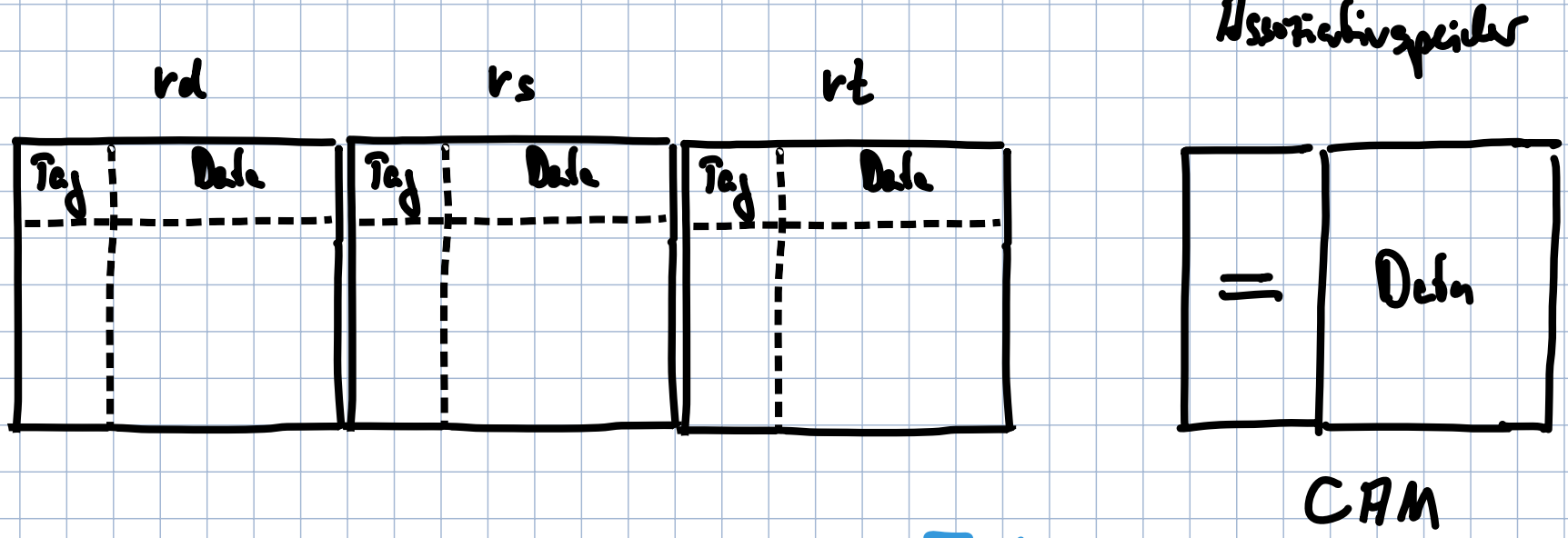
Einbinden des Speichers in die superskalare Architektur

Superscalar Architecture

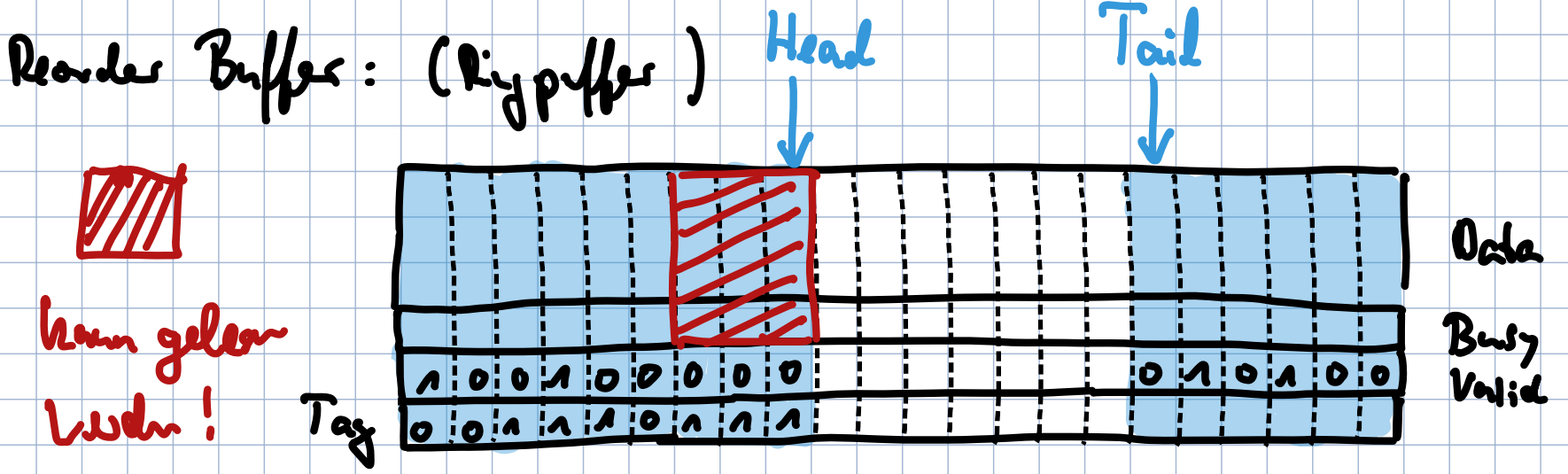


Implementierung der Registerumbenennung (Register Renaming Implementation)

Reservation Station:



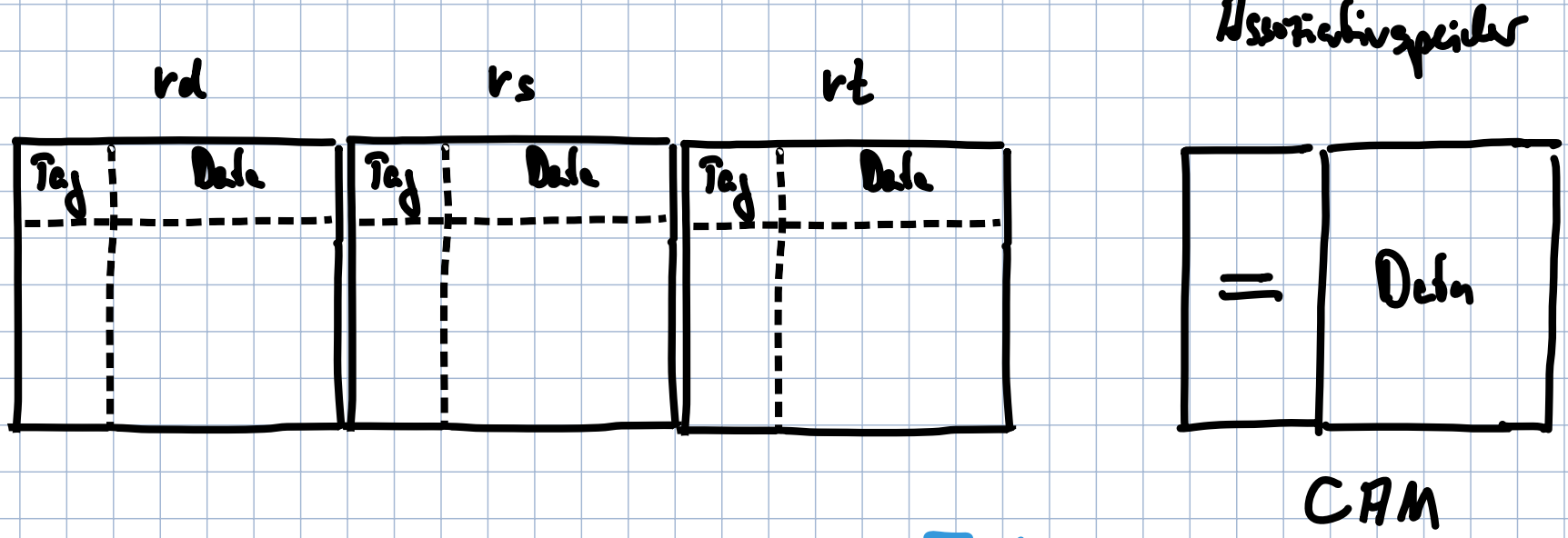
Reorder Buffer: (Ringpuffer)



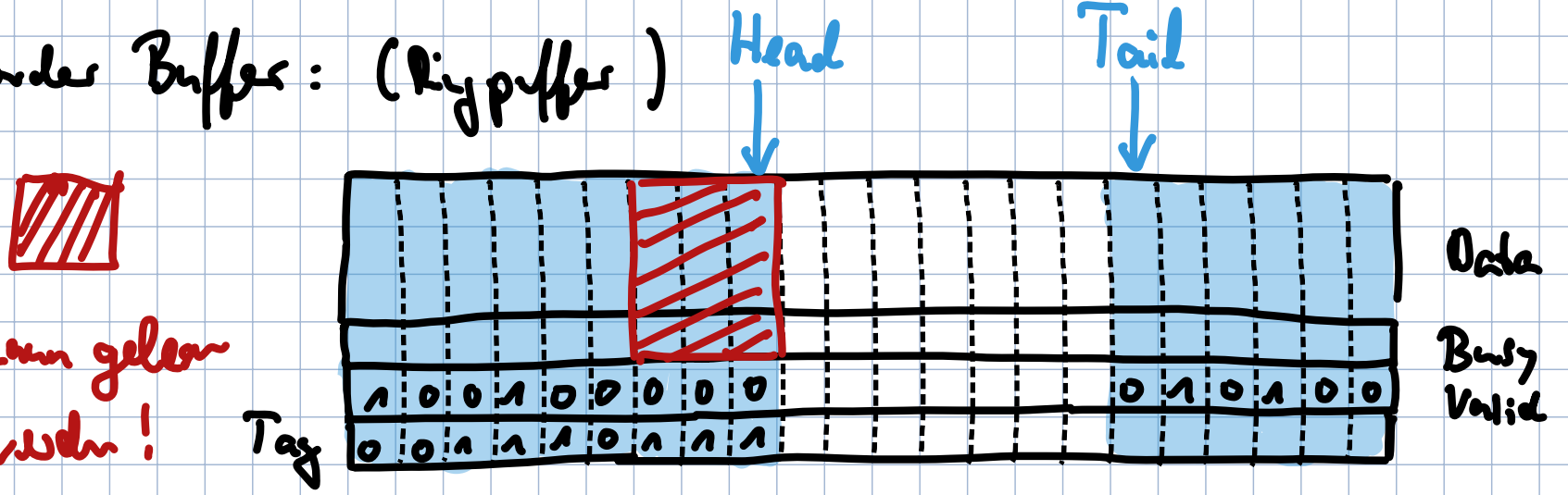
Lesen des Reorderbuffers in gleicher Reihenfolge

Implementierung der Registerumbenennung (Register Renaming Implementation)

Reservation Station:

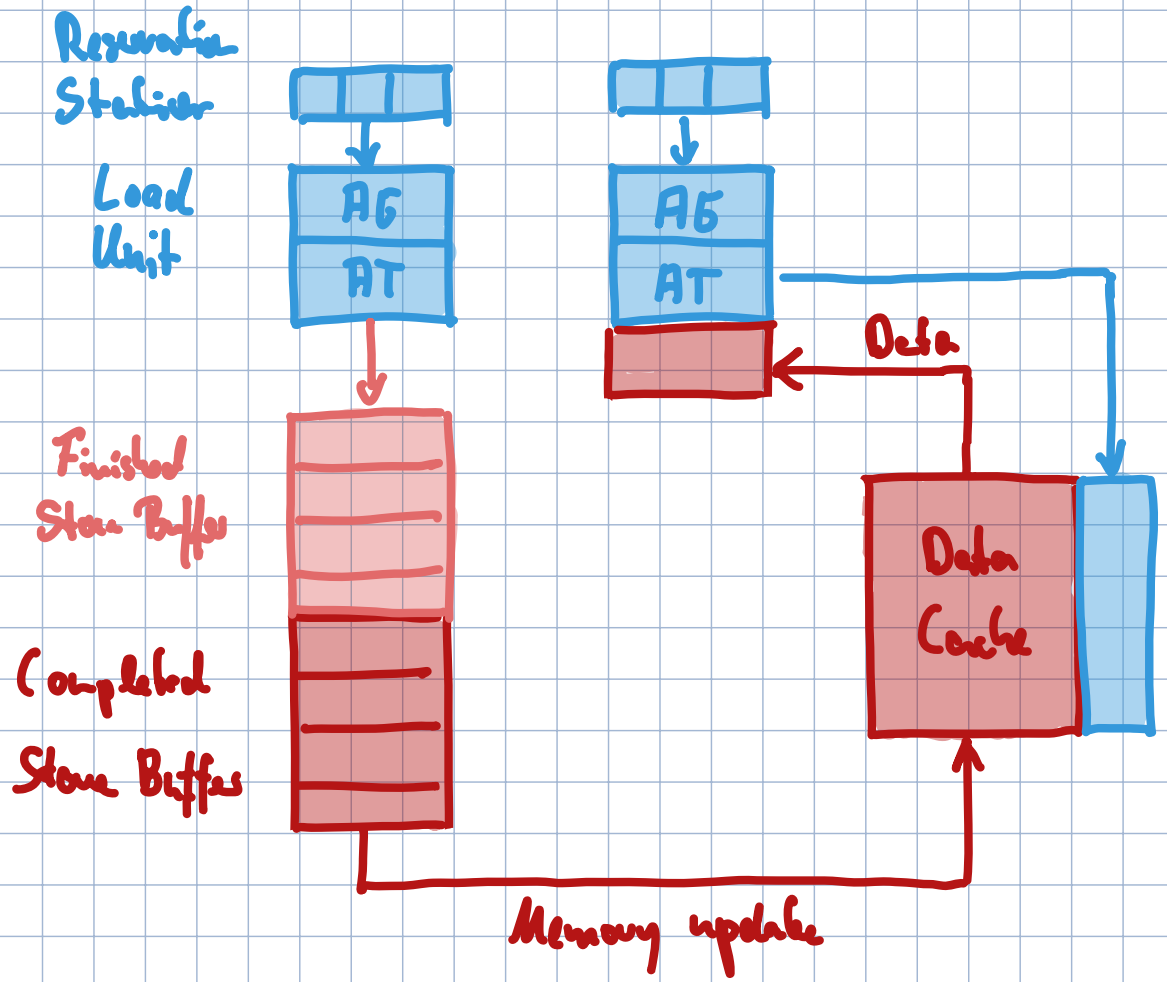


Reorder Buffer: (Ringpuffer)



Lesen des Reorderbuffers in gegebener Reihenfolge

Superskalarer Spitzbetrieb

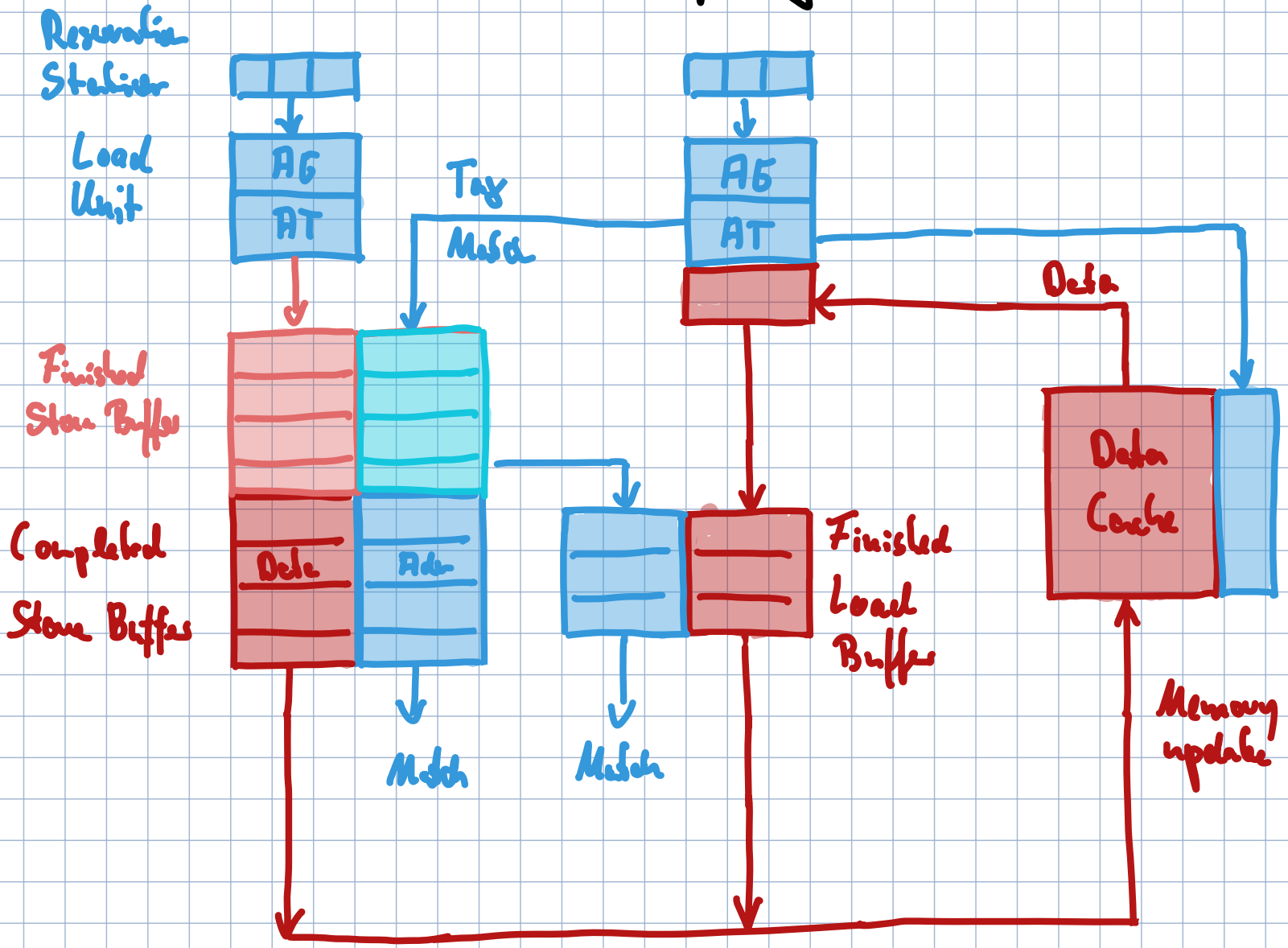


AG = Adress generation

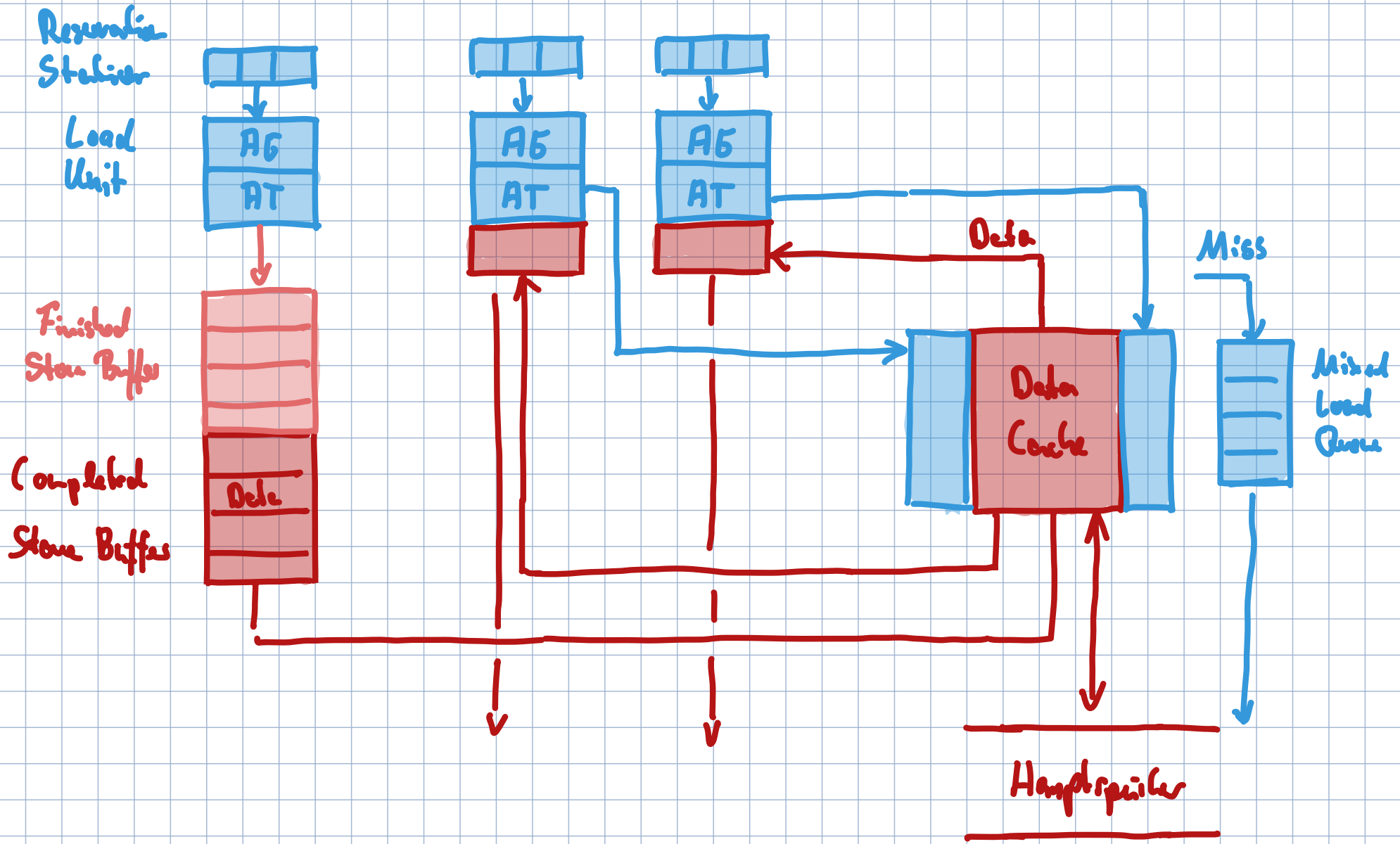
AT = Adress translation

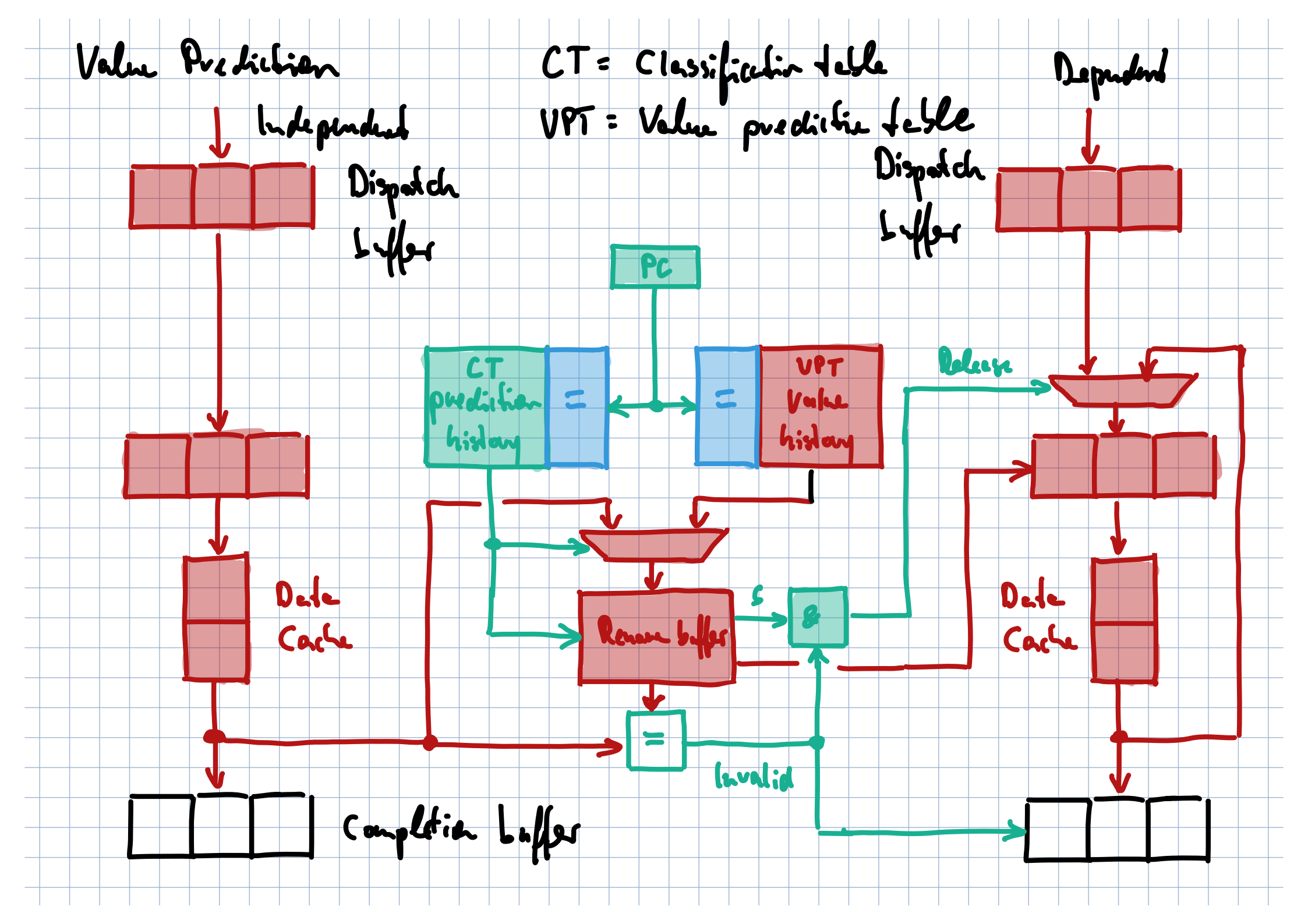
Superskalarer Spitzbetrieb: Vollständige Out-of-Order-Architektur

Load-Bypassing



Superskalarer Speicherzugriff: Weiterer Cache

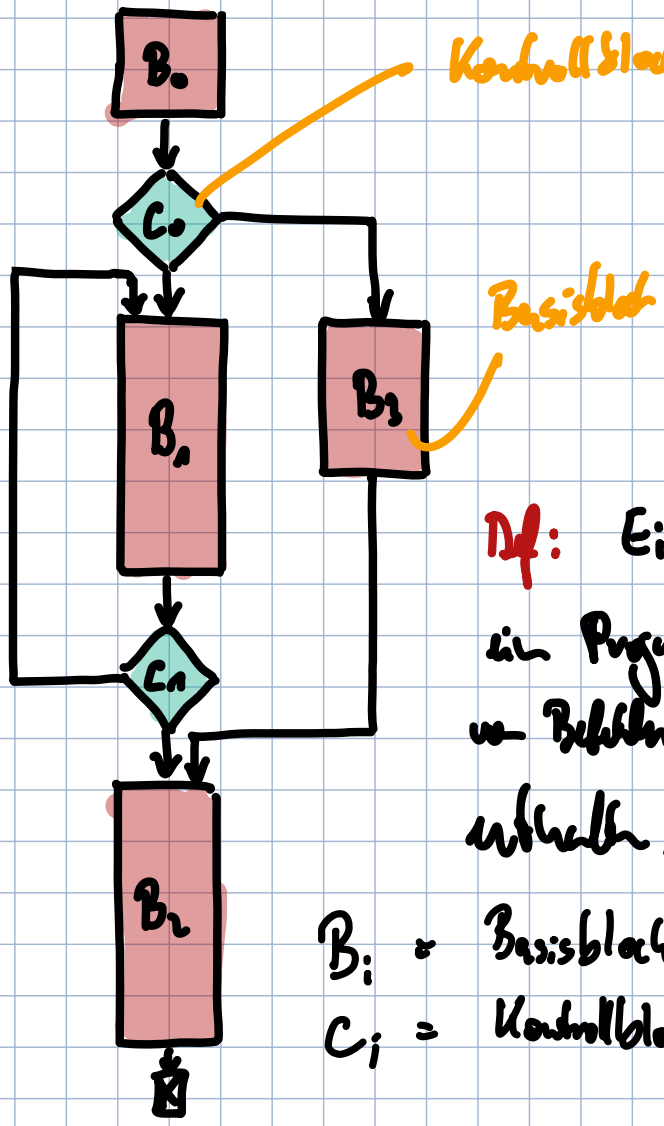



Value Prediction
CT = Classification table
VPT = Value prediction table
Independent
Dispatch Buffer
Dependent
Dispatch Buffer
PC
CT prediction history
=
=
VPT Value history
Release
Data Cache
Rename buffer
&
Data Cache
=
Invalid
Completion buffer

Branch Prediction

Sprungvorhersage: Programm im Speicher

Grundblockgraph



Def: Ein Basisblock ist ein Programmsegment (Folge von Befehlen) die keine Sprünge enthält.

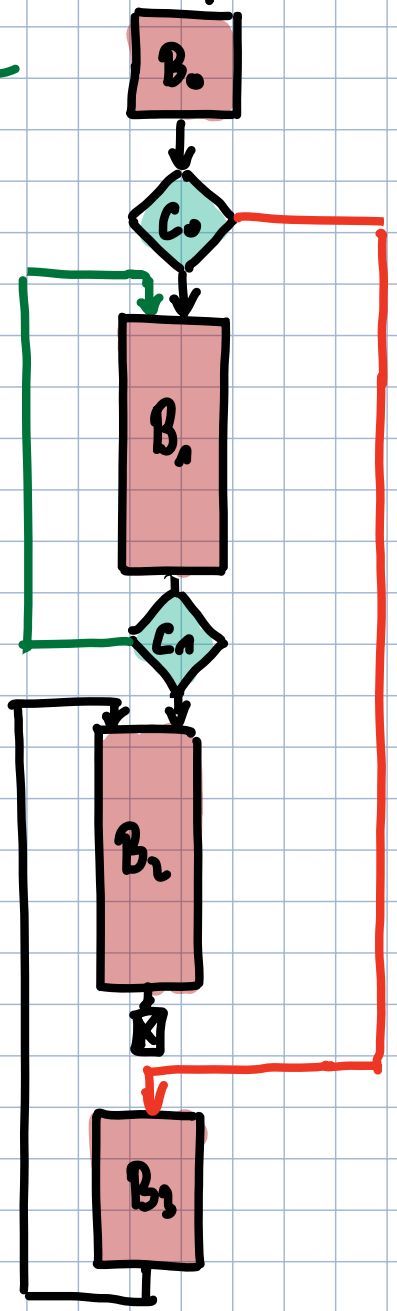
B_i = Basisblock
C_i = Kontrollblock
} Grundblockgraph

Sprungvorhersage auf Offset

Rücksprünge

Sprung Offset ⊖

Lineares Programm im Speicher



Vorwärts-sprünge kein Sprung Offset ⊕

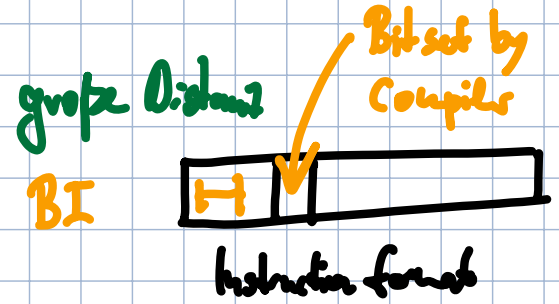
Statische Sprungvorhersage:

i) Program wird linear ausgeführt

Vorteil: Einfach

Nachteil: Schleifen ineffektiv

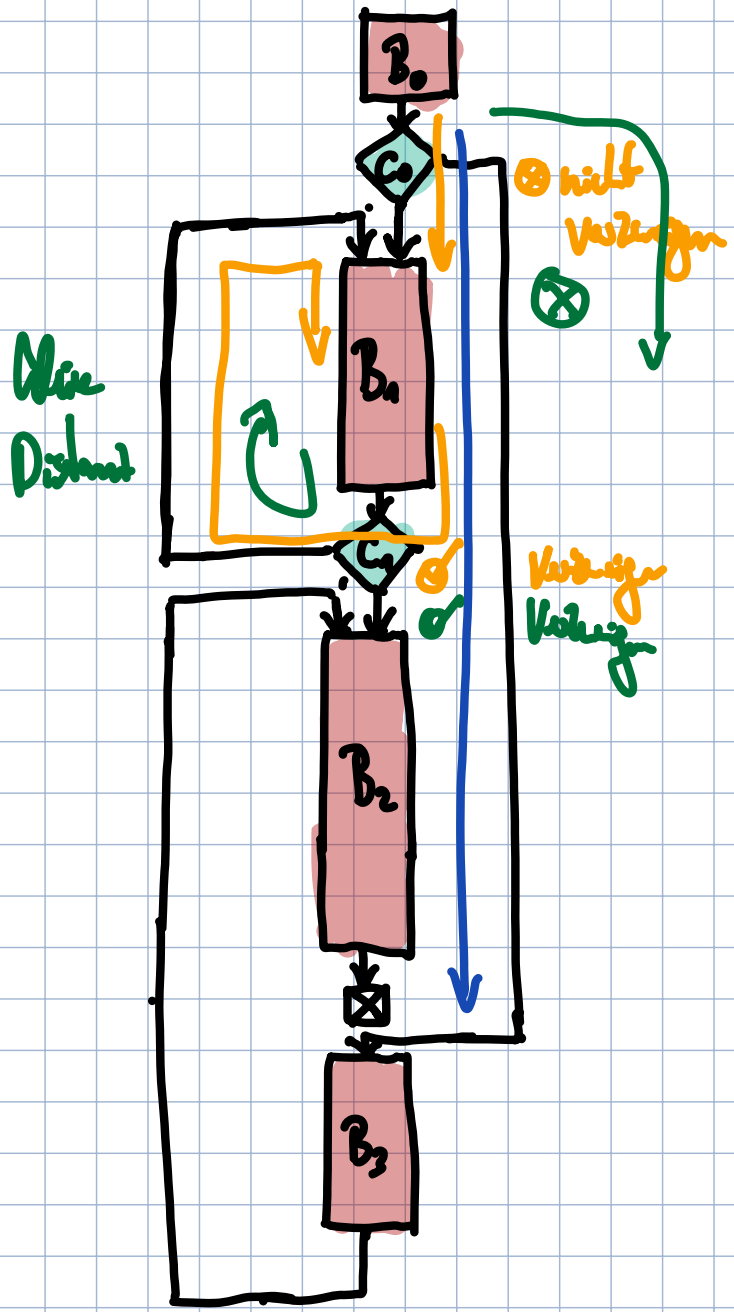
ii) Compiler bestimmt ob Verzweigt wird oder nicht:



iii) Bestimmung der Distanz zum Sprungziel

Idle ε Schleife!

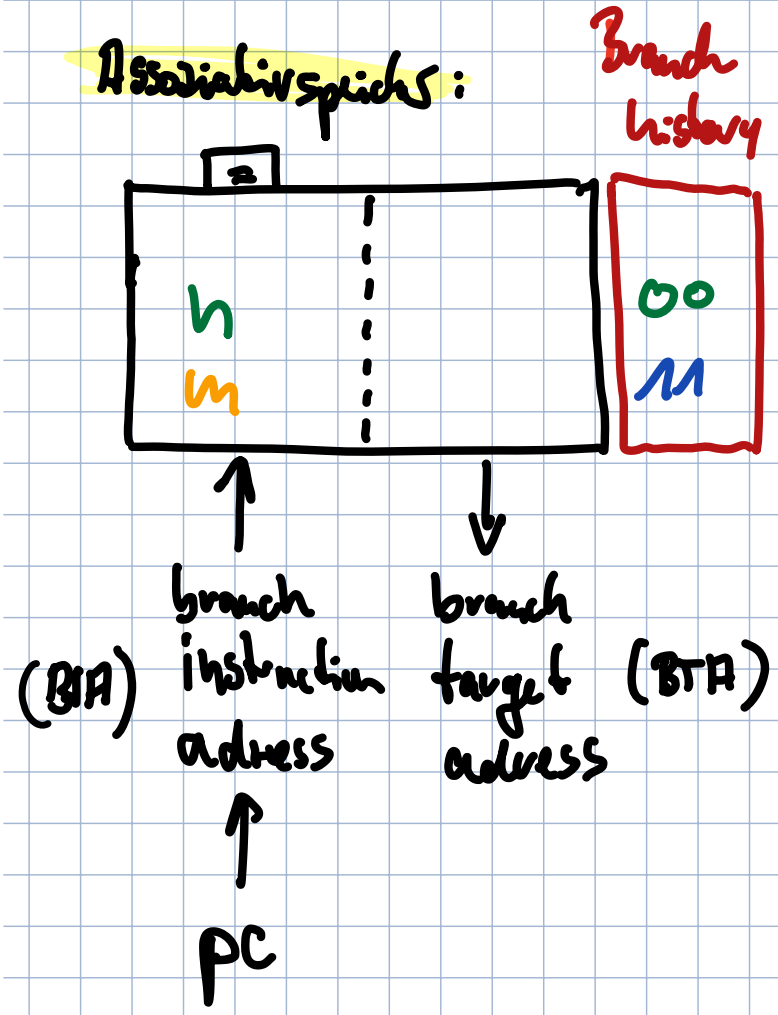
Sprung



Dynamische Sprungvorhersage (Dynamic Branch Prediction)

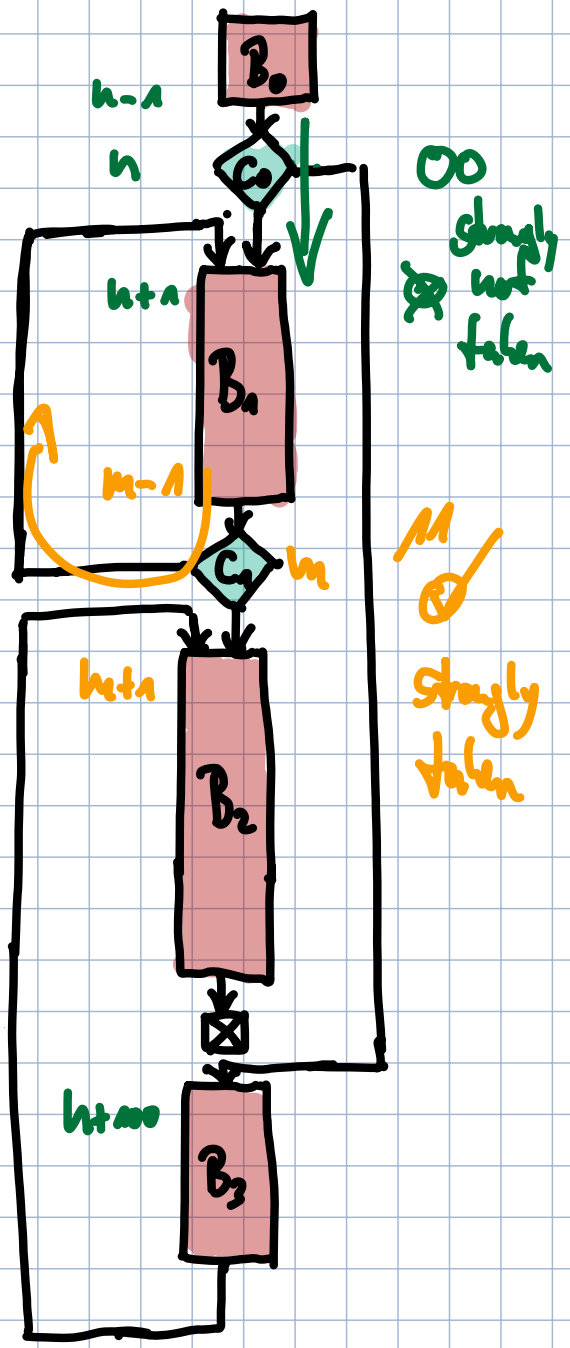
Branch Target Buffer:

Assoziativspeicher:



Möglich Zustände:

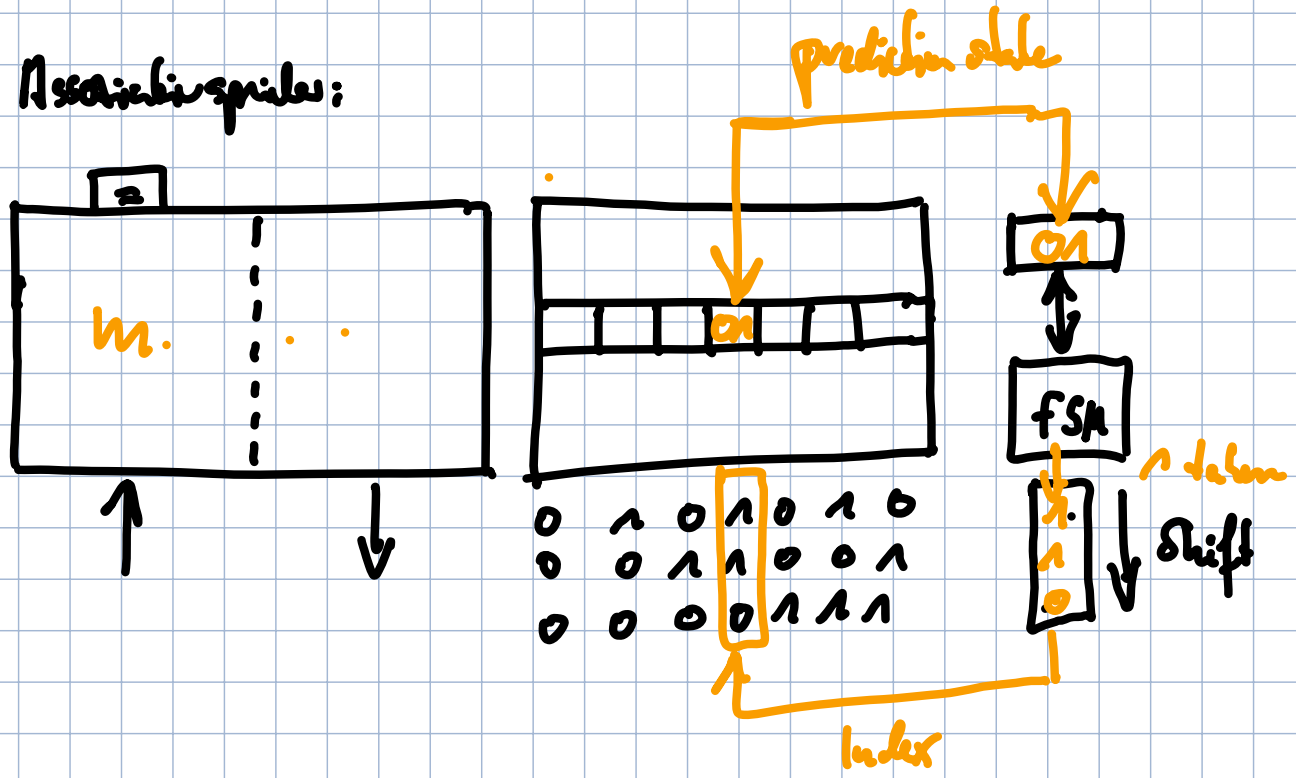
00	Strongly not taken
01	not taken
10	taken
11	strongly taken

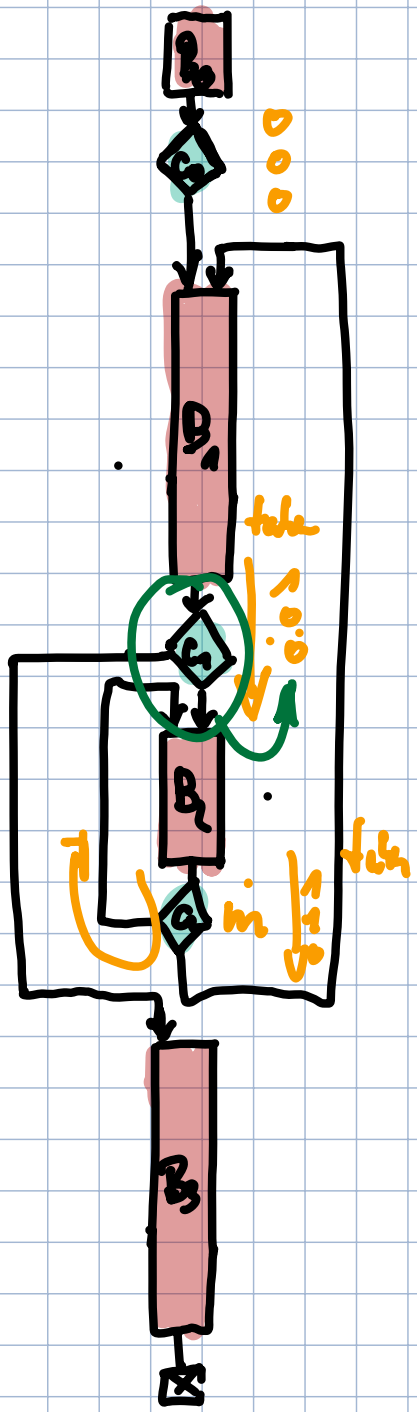


Hochdynamische Sprungvorhersage (95% Trefferrate)

(Correlated Branch Prediction)

Assoziativspeicher:

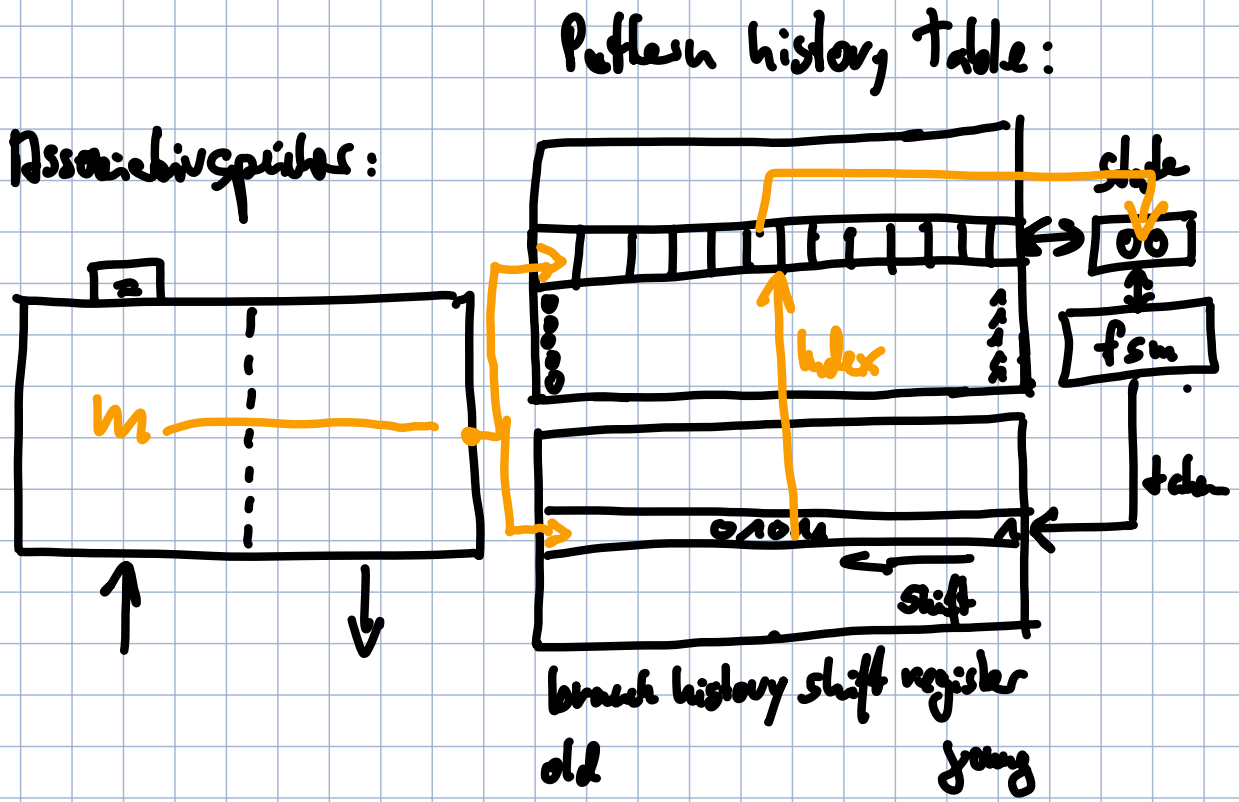


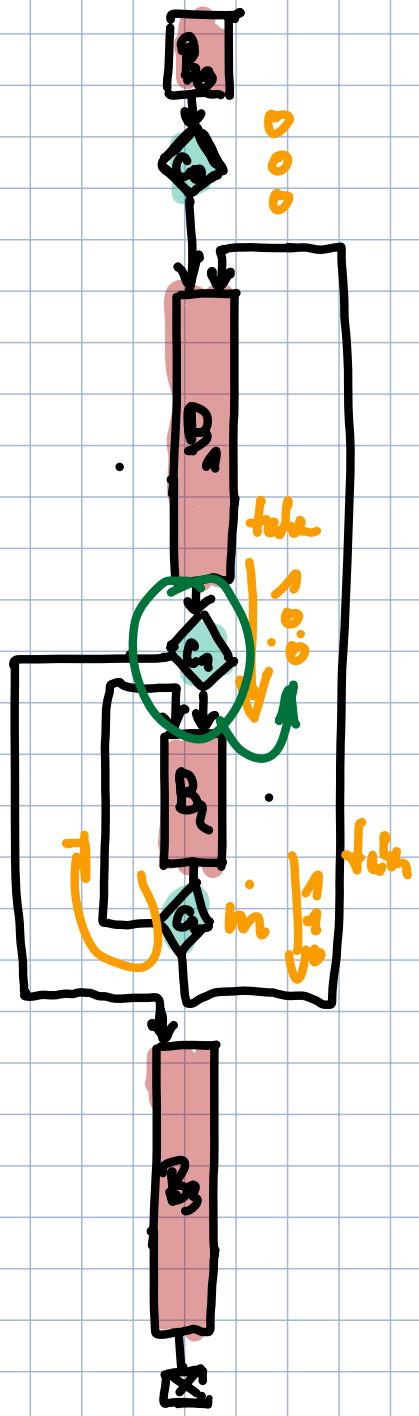


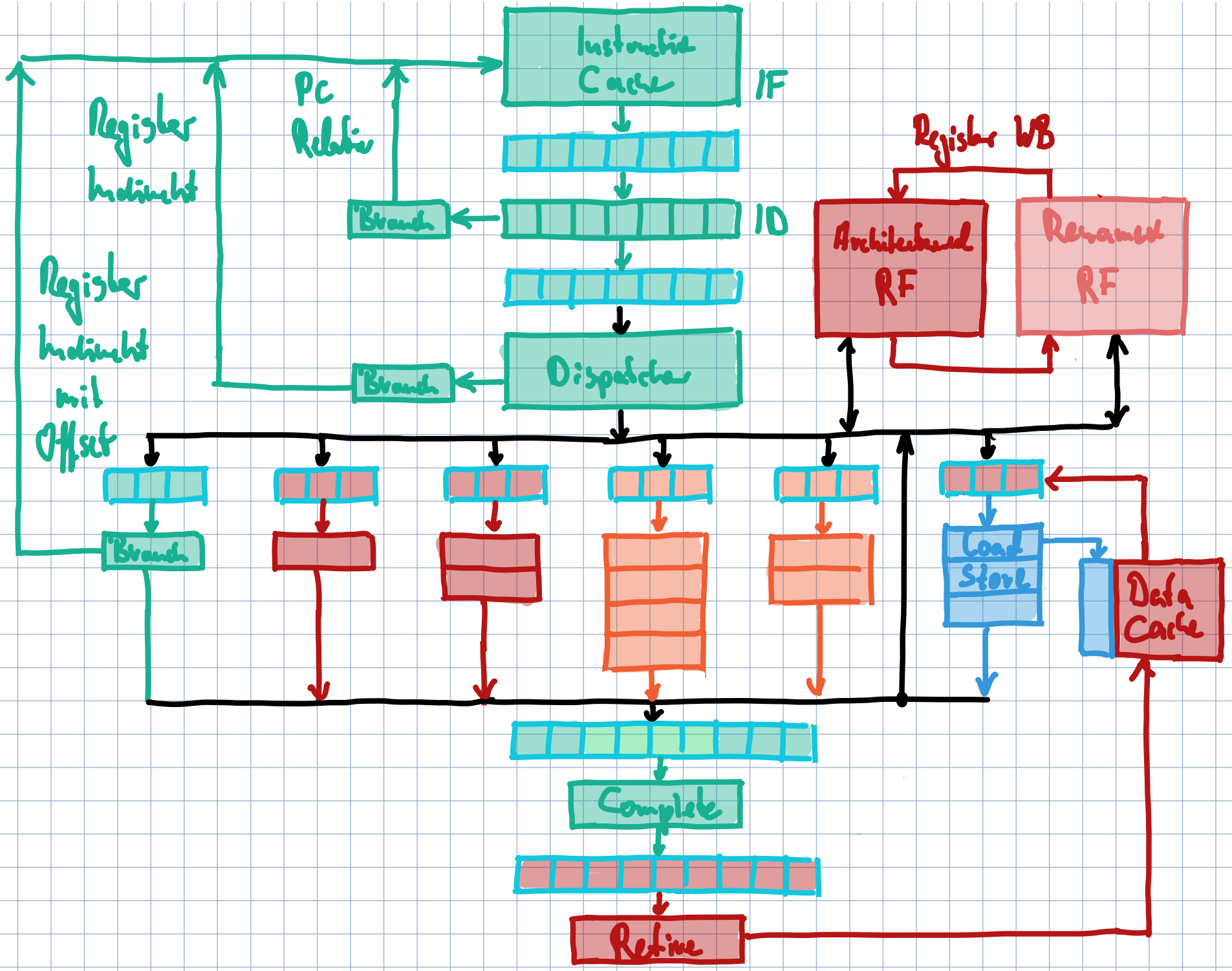
Berücksichtigung des Programmpfades:

In eine Schieberegister werden die vergangen Treffer durch '1' en gespeichert.

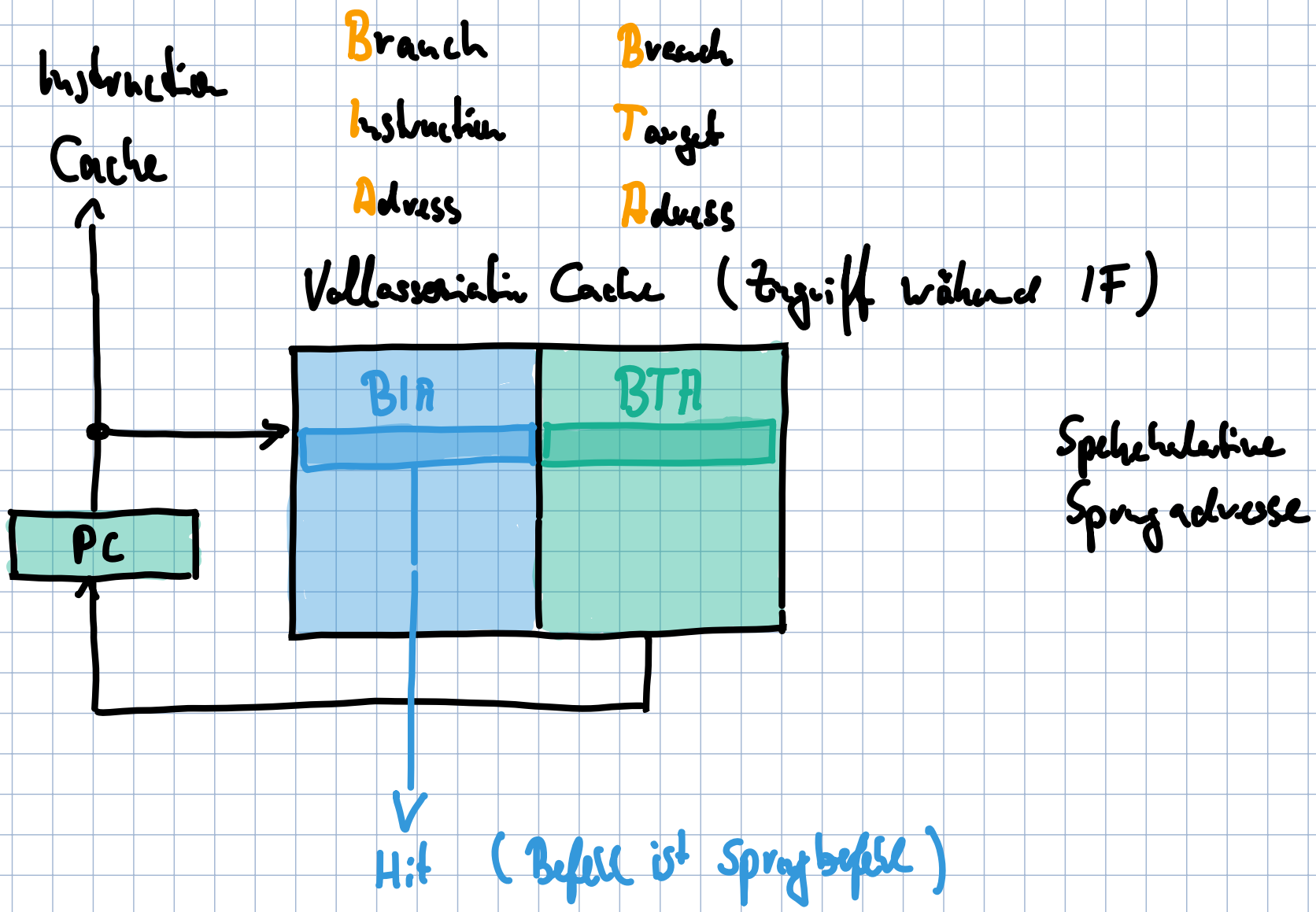
Hardwarebasierte Sprungvorhersage (95% Trefferrate)



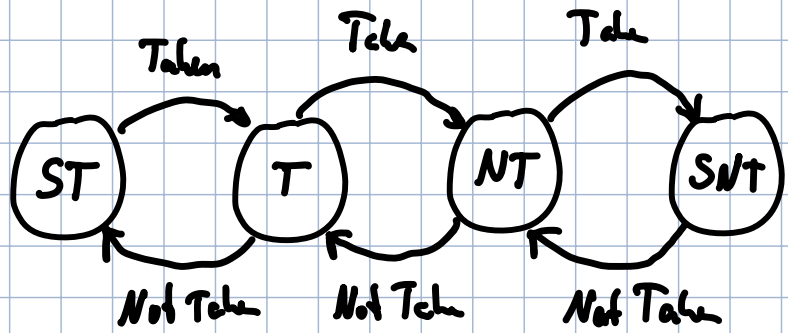



Instruction Cache
IF
ID
Register Indirect
Register Indirect with Offset
PC Relative
Branch
Branch
Branch
Dispatcher
Register WB
Architectural RF
Renamed RF
Load Store
Data Cache
Complete
Retire

Spekulative Sprungvorhersage (Branch Target Buffer)



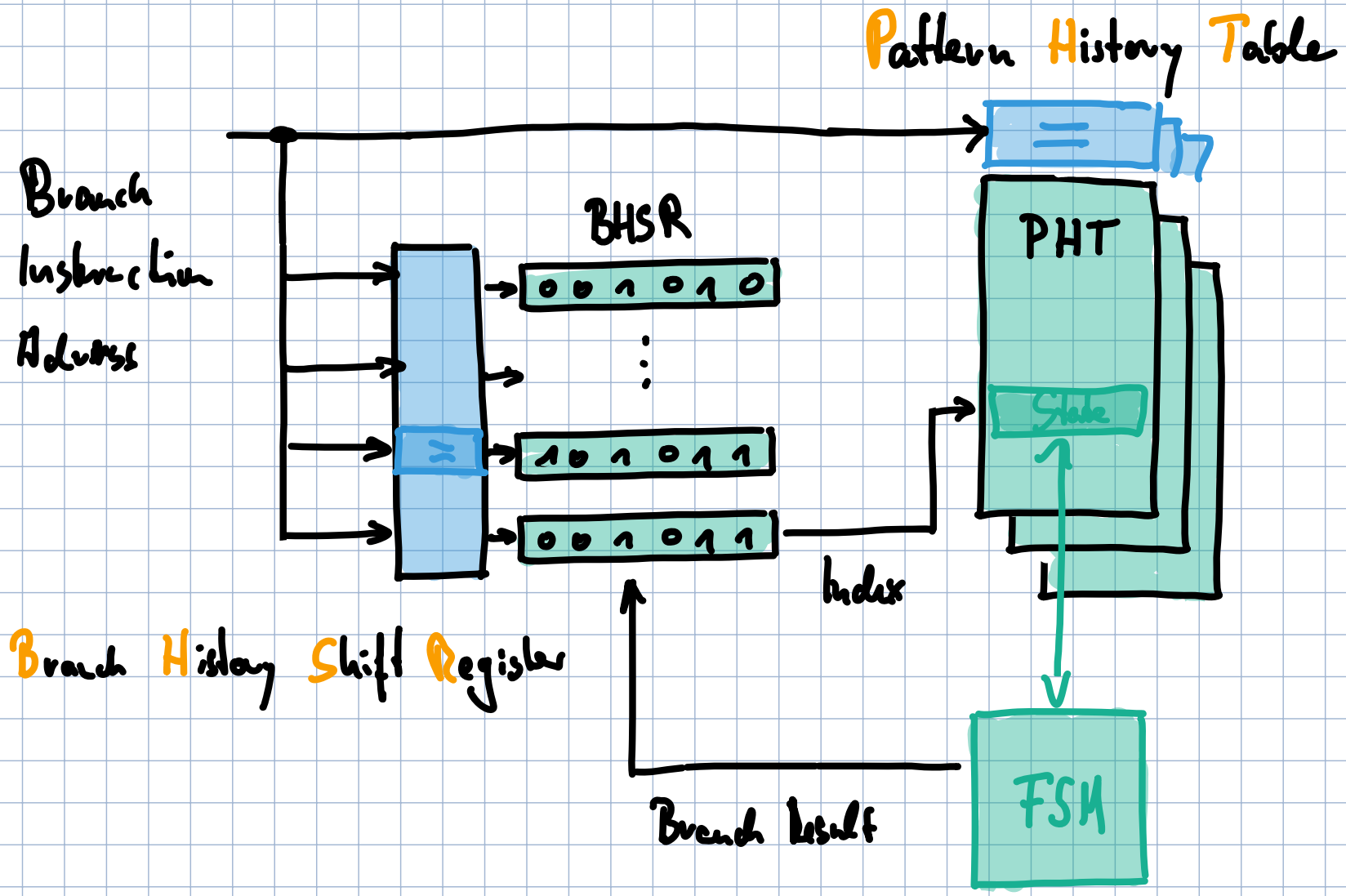
Dynamische Sprungvorhersage (Vorgehensweise)

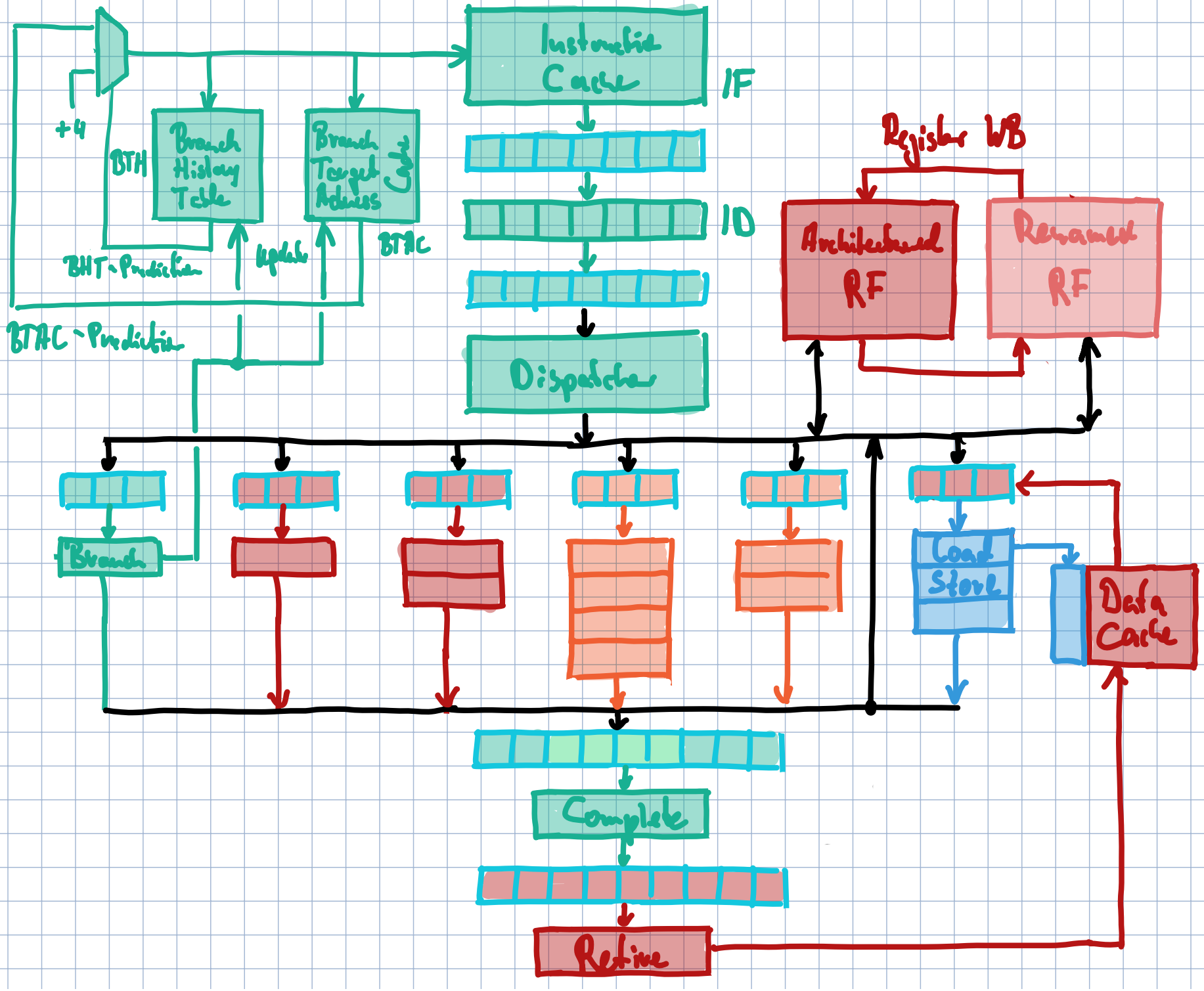


Branch Condition Speculation

Branch Target Speculation

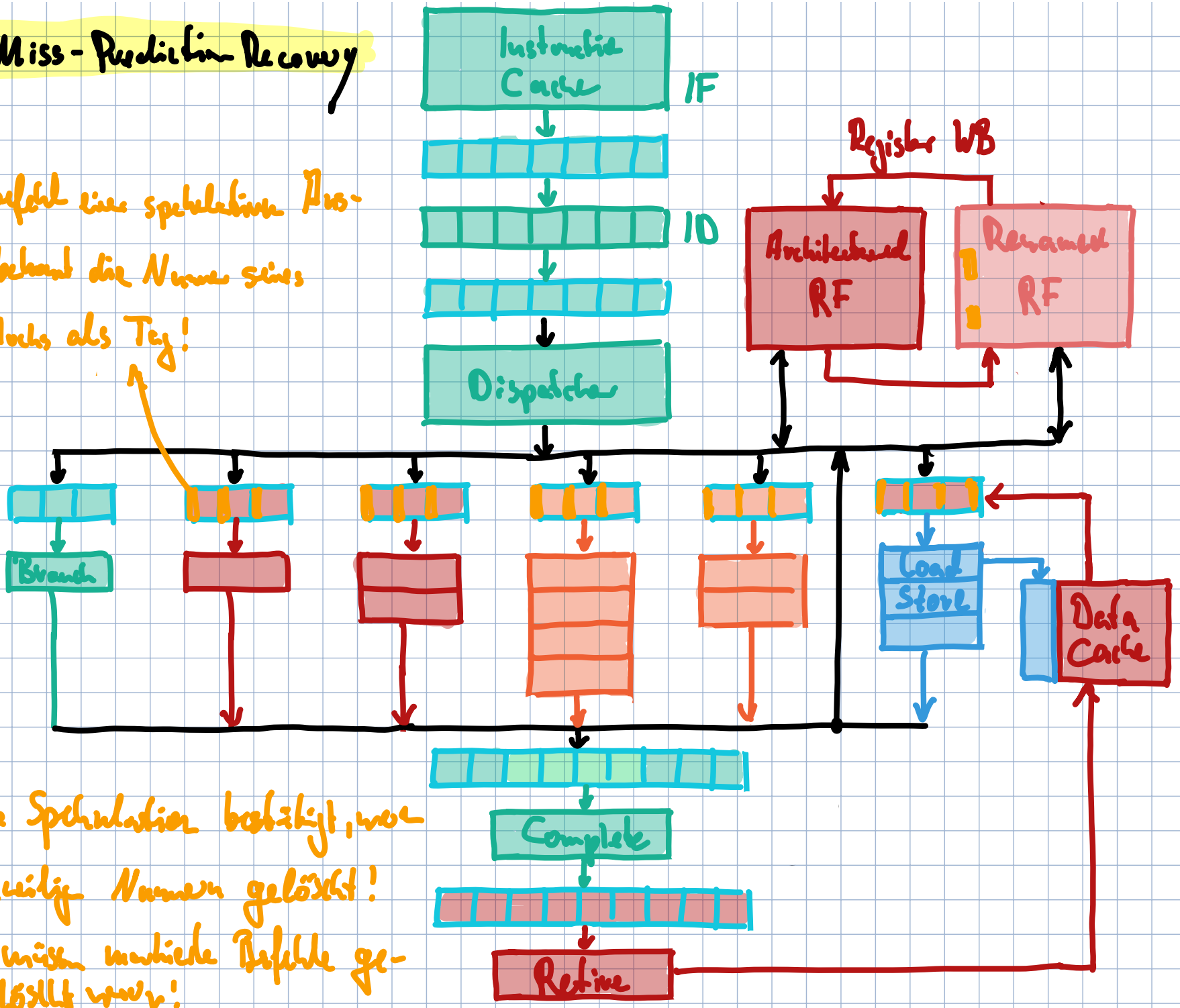
Dynamische Sprungvorhersage




+4
BTH
Branch History Table
Branch Target Address Cache
BHT-Prediction
Update
BTAC
BTAC-Prediction
Instruction Cache
IF
ID
Dispatcher
Register WB
Architectural RF
Renamed RF
Branch
Load Store
Data Cache
Complete
Retire

Branch Miss-Prediction Recovery

Jeder Befehl eines spekulativen Abschnitts behält die Nummer seines Basisblocks als Tag!

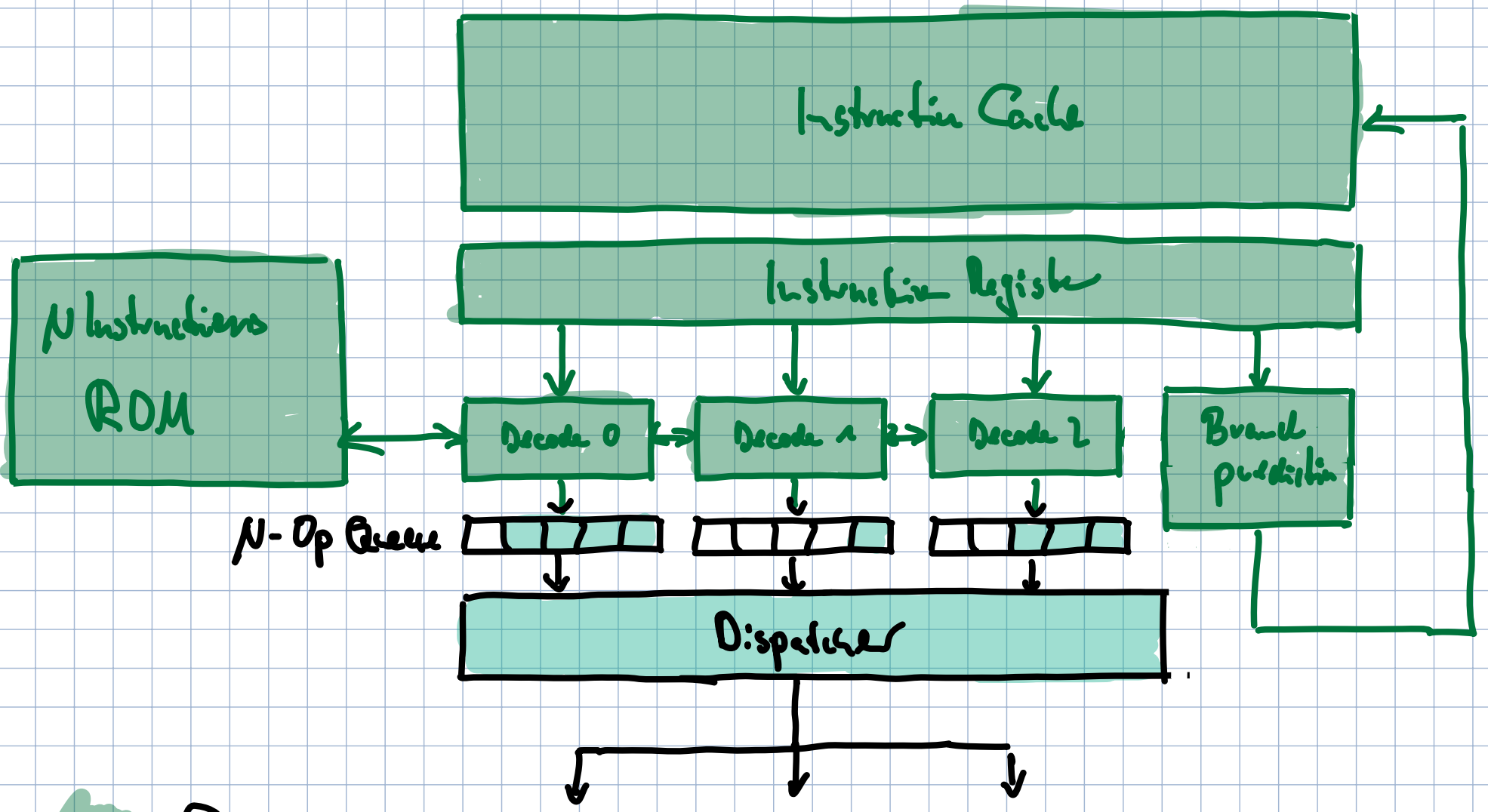


Ist die Spekulation bestätigt, werden die jeweiligen Nummern gelöscht!

Ggf. müssen mehrere Befehle gelöscht werden!

Microcode in superscalaren Architekturen

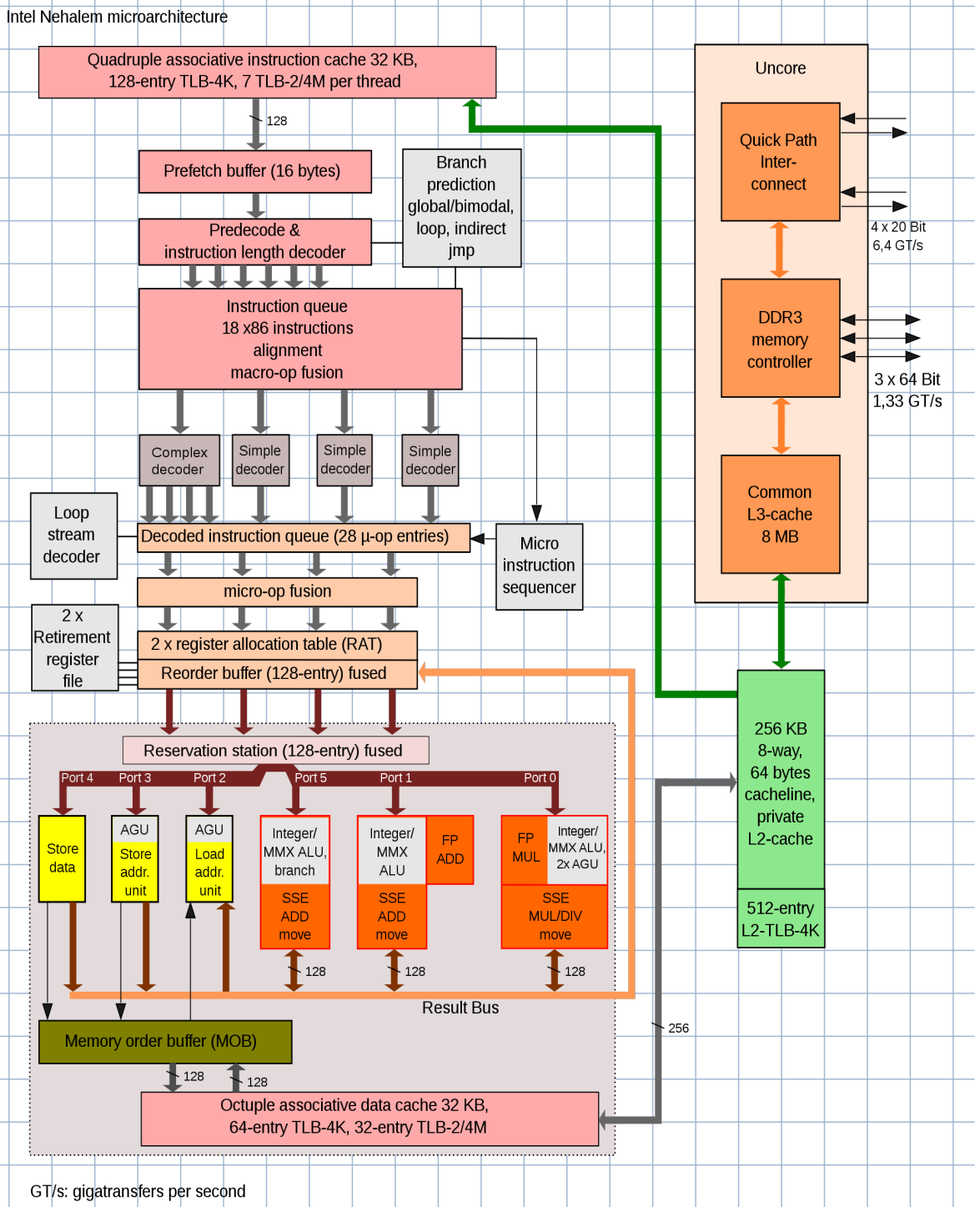
RISC in CISC : Mikroprogrammierbarkeit für superskalare Ausführung von CISC-Befehlen



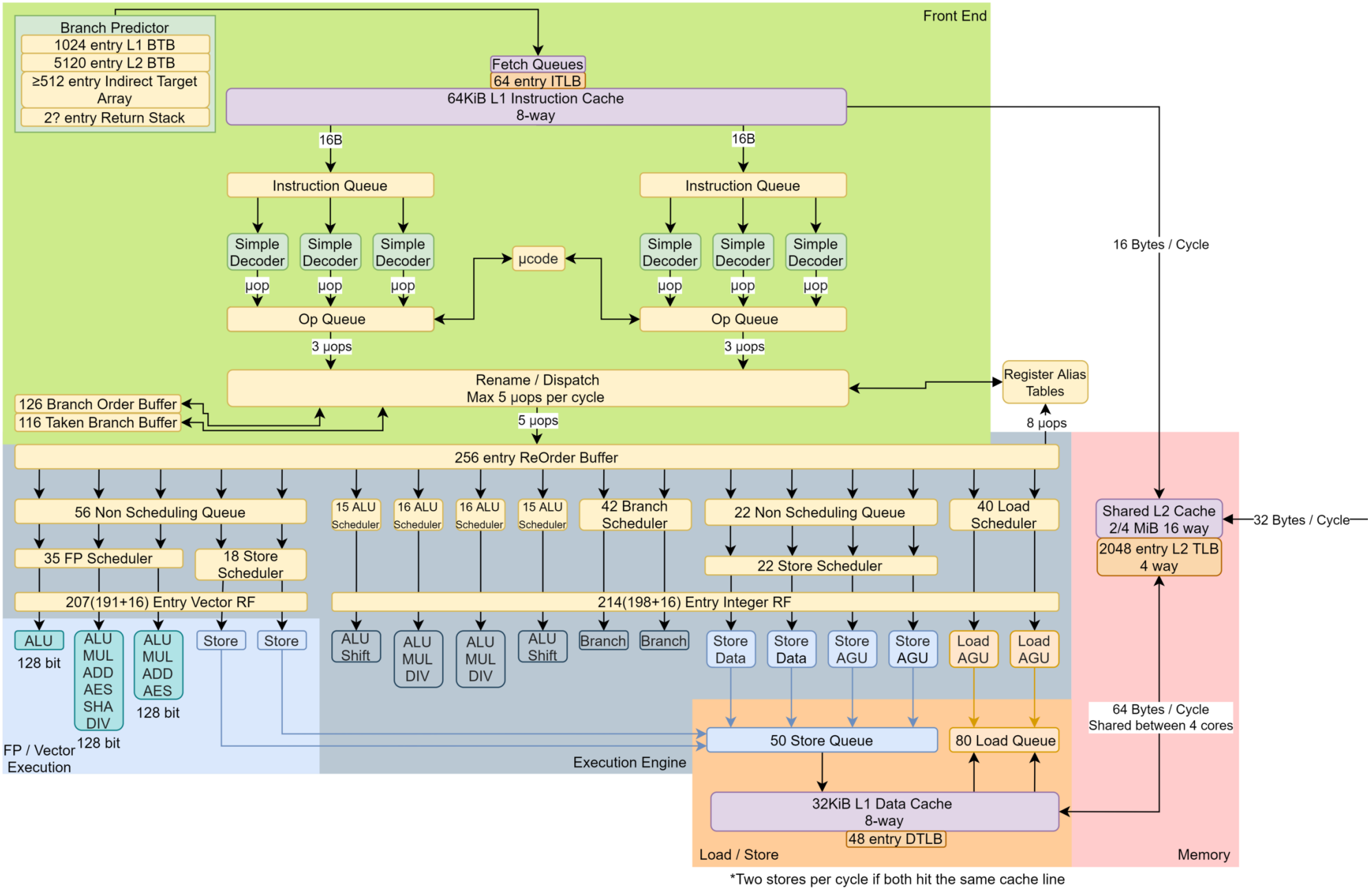
CISC

RISC

Intel : Nehalem - Microarchitektur



Intel : Gracemont - Mikroarchitektur



*Two stores per cycle if both hit the same cache line