Aufgabenblatt 14: Pipelining

Aufgabe 1: Pipeline ohne Harvard-Architektur

In dieser Aufgabe ist eine 5-stufige Pipeline (siehe Abbildung 1) eines RISC-Prozessors in verschiedenen Varianten zu betrachten.

In der ersten Variante kommt es zu strukturellen Konflikten, da auf das Registerfile nicht gleichzeitig schreibend und lesend zugegriffen werden kann. Datenkonflikte werden mittels Scoreboard aufgelöst.

In der zweiten Variante sind die strukturelle Konflikte beim Registerzugriff durch Halbtakt-Verarbeitung teilweise gelöst: zwei Zugriffe auf das Register-File in einem Takt (schreiben im ersten Halbtakt, lesen im zweiten Halbtakt).

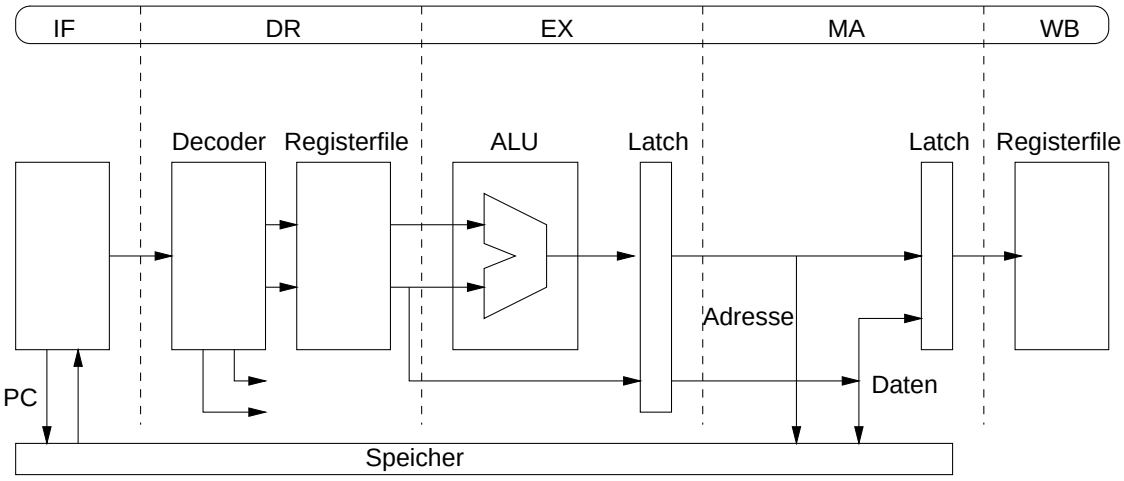

Abbildung 1: RISC-Pipeline mit 5 Stufen.

1. Füllen Sie das Ablaufdiagramm jeweils für die Varianten mit und ohne Halbtaktverarbeitung aus.
2. Optimieren Sie den Assemblercode durch semantikinvariante Operationen, um die Pipeline besser auszulasten. Füllen Sie nun wieder die Diagramme aus.

Keine Harvard-Architektur, kein Forwarding, unoptimiert

Zeile	Marke	Operation	Kommentar
1	L1:	Load R1,X	R1=X
2		Store W,R1	W =R1
3		Load R2,Y	R2=Y
4		Load R3,Z	R3=Z
5		Sub R2,R3	R2=R2-R3
6		Store U,R2	U =R2
7		Inc R1	R1=R1+1
8		Store X,R1	X =R1

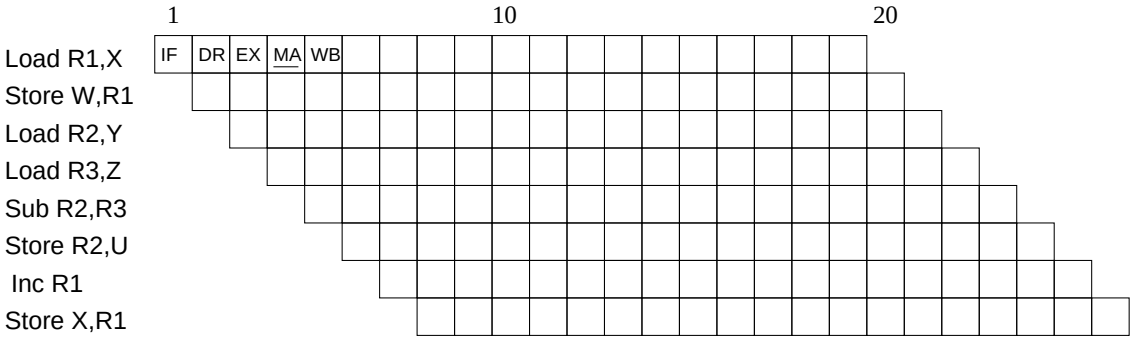

Abbildung 2: Ablaufdiagramm der Programmsequenz **(ohne Halbtaktverarbeitung)**.

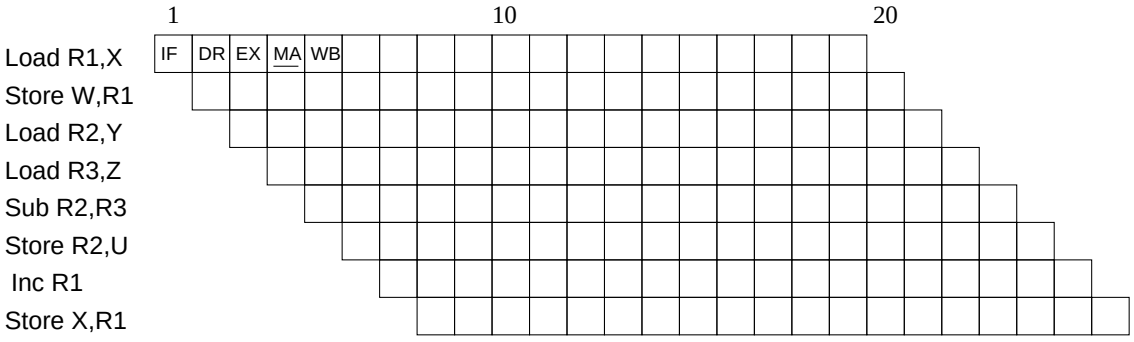

Abbildung 3: Ablaufdiagramm der Programmsequenz **(mit Halbtaktverarbeitung)**.

Keine Harvard-Architektur, kein Forwarding, optimiert

Zeile	Marke	Operation	Kommentar
1	L1:	Load R1,X	R1 = X
2			
3			
4			
5			
6			
7			
8			

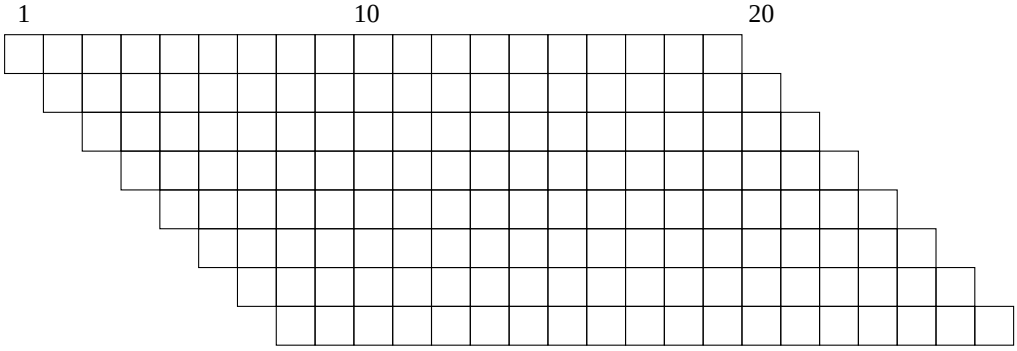

Abbildung 4: Ablaufdiagramm der Programmsequenz **(ohne Halbtaktverarbeitung)**.

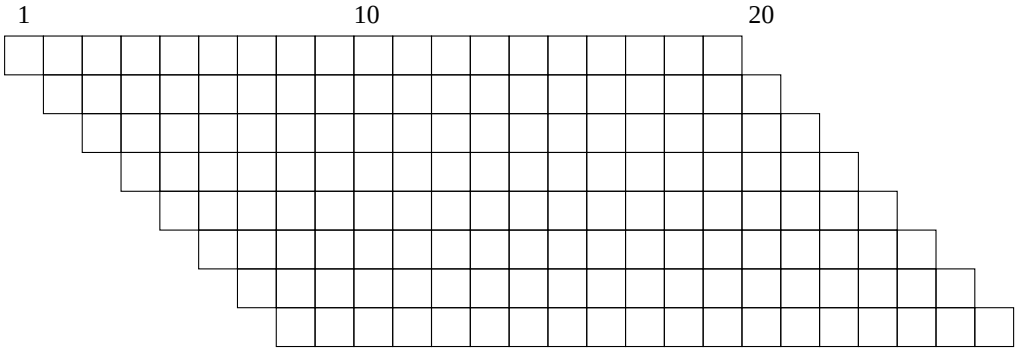

Abbildung 5: Ablaufdiagramm der Programmsequenz **(mit Halbtaktverarbeitung)**.

Aufgabe 2: Pipeline mit Harvard-Architektur, Forwarding und Halbtaktverarbeitung

Betrachten Sie nun den in Abbildung 6 gezeigten RISC-Prozessor mit Harvard-Architektur und Forwarding innerhalb der Pipeline. Erstellen Sie wieder das Ablaufdiagramm für die ursprüngliche Progammsequenz und versuchen Sie anschliessend diese wieder zu optimieren!

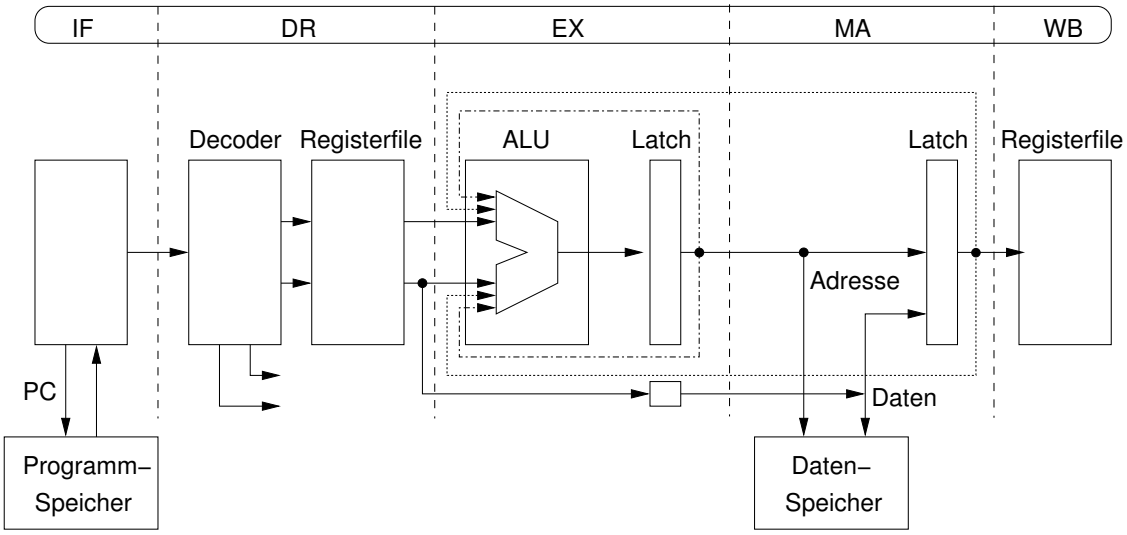

Abbildung 6: RISC-Pipeline mit Harvard-Architektur und Forwarding.

1. Füllen Sie das Ablaufdiagramm für den unoptimierten Code aus.
2. Optimieren Sie den Assemblercode durch semantikinvariante Operationen und tragen Sie Ihr Ergebnis in das zweite Diagramm ein.

Harvard-Architektur, mit Forwarding, Halbtaktverarbeitung, unoptimiert

Zeile	Marke	Operation	Kommentar
1	L1:	Load R1,X	R1=X
2		Store W,R1	W =R1
3		Load R2,Y	R2=Y
4		Load R3,Z	R3=Z
5		Sub R2,R3	R2=R2-R3
6		Store U,R2	U =R2
7		Inc R1	R1=R1+1
8		Store X,R1	X =R1

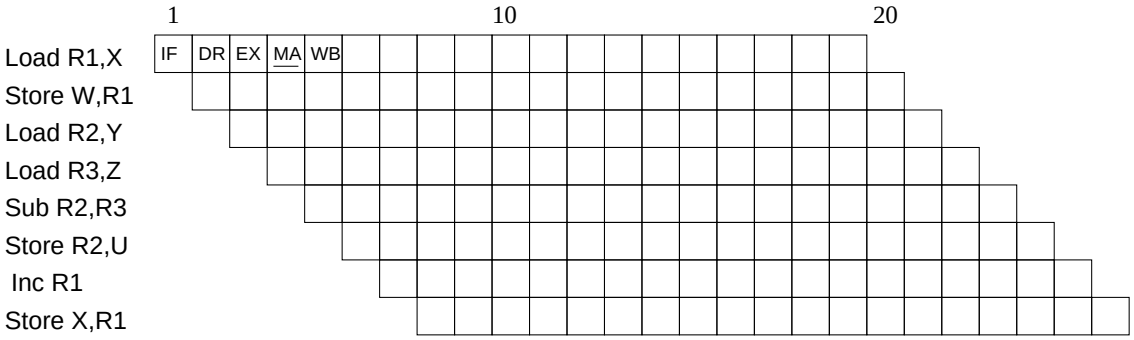

Abbildung 7: Ablaufdiagramm der Programmsequenz.

Harvard-Architektur, mit Forwarding, Halbtaktverarbeitung, optimiert

Zeile	Marke	Operation	Kommentar
1	L1:	Load R1,X	R1 = X
2			
3			
4			
5			
6			
7			
8			

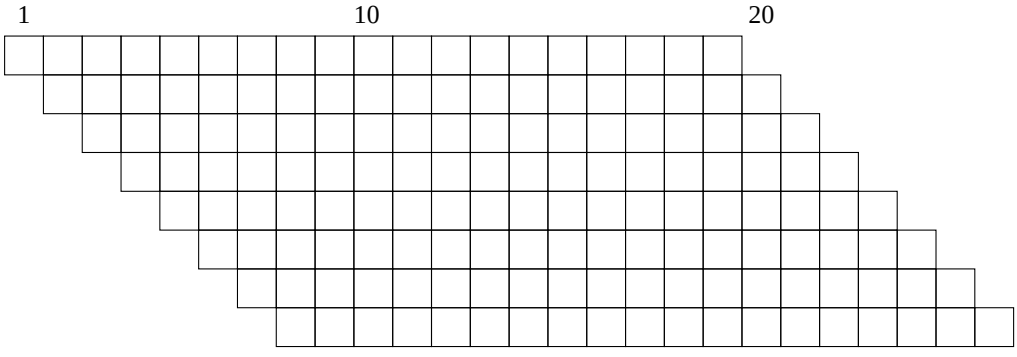

Abbildung 8: Ablaufdiagramm der Programmsequenz**(optimiert)**.