



Lösung 1

1. Aufgaben der einzelnen Pipeline-Stufen der DLX-Pipeline für

- **Arithmetisch-logische Befehle:** In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet die Inhalte der beiden Quellregister weiter. In der Execute-Stufe wird die Operation ausgeführt. Die Memory-Stufe wird bei arithmetisch-logischen Befehlen nicht benötigt und leitet deshalb die Daten weiter. In der Write-Back-Stufe wird das Ergebnis der Operation in das Zielregister geschrieben.
- **Lade-/Speicher-Befehle:** In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet bei einem Ladebefehl den Registerinhalt mit der Adresse weiter. Bei einem Speicherbefehl wird das Register mit dem zu speichernden Wert ausgelesen. In der Execute-Stufe wird die Speicheradresse aus dem Registerinhalt und dem Displacement berechnet. Der eigentliche Speicherzugriff erfolgt in der Memory-Stufe. Bei einem Ladebefehl speichert die Write-Back-Stufe den geladenen Wert in das Zielregister ab.
- **Bedingte Sprungbefehle mit PC-relativer Adressierung:** In der Instruction Fetch-Stufe wird der Befehl aus dem Speicher geladen. Die Instruction Decode-Stufe erzeugt die prozessorinternen Steuersignale und leitet das Displacement und den PC sowie den Vergleichsregisterinhalt weiter. In der Execute-Stufe wird einerseits entschieden, ob der Sprung genommen wird und andererseits aus dem PC und dem Displacement der neue PC berechnet, der dann, falls der Sprung genommen wird, in der MEM-Stufe den PC ersetzt. Im nächsten Takt kann in der IF-Stufe der Befehl von der Sprungzieladresse geladen werden. Die Write-Back-Stufe ist bei bedingten Sprüngen nicht weiter von Bedeutung.

2. Echte Datenabhängigkeiten und Steuerflussabhängigkeiten:

- Echte Abhängigkeiten:

$S1 \rightarrow S3$	$S1 \rightarrow S5$	$S1 \rightarrow S6$	$S1 \rightarrow S7$	$S1 \rightarrow S9$	$S1 \rightarrow S10$
$S2 \rightarrow S4$	$S3 \rightarrow S4$	$S4 \rightarrow S5$	$S6 \rightarrow S7$	$S6 \rightarrow S9$	$S6 \rightarrow S10$
$S7 \rightarrow S8$	$S9 \rightarrow S10$				

- Steuerflussabhängigkeiten:

$S8 \rightarrow S9$

3. Behebung der Pipelinekonflikte:

```

S1:          add  $t1, $zero, $zero
S2:          lw   $t3, 0x1500($zero)
S3:    loop:  lw   $t4, 0x5000($t1)
              nop
S4:          add  $t5, $t4, $t3
S5:          sw   $t5, 0x400($t1)
S6:          addi $t1, $t1, 4
S7:          subi $t2, $t1, 0x400
S8:          bnez $t2, loop
              nop
              nop
              nop
S9:    end:    srli $t1, $t1, 2
S10:         sw   $t1, 0x2000($zero)

```

4. Struktur- oder Ressourcenkonflikte:

Treten auf, wenn zwei oder mehrere Pipeline-Stufen gleichzeitig dieselbe Ressource benötigen, auf diese aber nur einmal zugegriffen werden kann.

Sie können bei der DLX-Pipeline nicht auftreten, da diese entsprechend entworfen ist.

5. Ausgabeabhängigkeiten (*Output dependence*) und Gegenabhängigkeiten (*Anti-dependence*) können in der DLX-Pipeline nicht zu Konflikten führen, da

- das Lesen aus Registern immer in der Stufe 2 (ID/RF) und
- das Schreiben in Register immer in der Stufe 5 (WB) erfolgt.

6. In den häufig in Programmen auftretenden Schleifen wird die Mehrzahl der Sprünge genommen. Falls ein Sprungbefehl genommen wird, müssen die drei Befehle hinter dem Sprungbefehl aus der Pipeline gelöscht werden (*pipeline flushing*).

Lösung 2

1. (a) Echte Datendbhängigkeiten:

```

S1 → S2    S1 → S3    S2 → S3    S3 → S7    S3 → S8
S4 → S5    S4 → S6    S5 → S6    S6 → S7    S6 → S8

```

(b)

```

S1:  addi  $t1, $zero, 10
S4:  addi  $t4, $zero, 5
      NOP
S2:  sll   $t2, $t1, 4
S5:  sll   $t5, $t4, 4
      NOP
S3:  or    $t3, $t1, $t2
S6:  or    $t6, $t4, $t5
      NOP
      NOP
S7:  or    $t7, $t3, $t6
S8:  and   $t8, $t3, $t6

```

(c) Inhalte von \$t7 und \$t8 nach der Ausführung:

Register	Inhalt
\$t7	0x0000 00FF
\$t8	0x0000 0000

Lösung 3

1. Zustand der Pipeline:

Taktzyklus	IF	ID/RF	EX	MEM	WB
1.	lw				
2.	nop	lw			
3.	nop	nop	lw		
4.	addi	nop	nop	lw	
5.	nop	addi	nop	nop	lw
6.	nop	nop	addi	nop	nop
7.	sw	nop	nop	addi	nop
8.	addi	sw	nop	nop	addi
9.	nop	addi	sw	nop	nop
10.	nop	nop	addi	sw	nop
11.	sub	nop	nop	addi	sw
12.	nop	sub	nop	nop	addi
13.	nop	nop	sub	nop	nop
14.	bnez	nop	nop	sub	nop
15.		bnez	nop	nop	sub
16.			bnez	nop	nop
17.				bnez	

Anzahl der Schleifendurchläufe: $1000/4 = 250$

\Rightarrow Anzahl der erforderlichen Taktzyklen: $250 \times 17 \text{ Takte} + 1 \text{ Takt} = 4251 \text{ Takte}$ (Ein Taktzyklus ist für die WB-Phase im letzten Durchlauf der Schleife)

2. Neue Reihenfolge der Befehle:

```

loop:  lw   $t1, 4($t2)
        addi $t2, $t2, 0x4
        sub  $t4, $t3, $t2
        addi $t1, $t1, 0x10
        sw   $t1, 4($t2)
        bnez $t4, loop

```

Zustand der Pipeline:

Taktzyklus	IF	ID/RF	EX	MEM	WB
1.	lw				
2.	addi	lw			
3.	sub	addi	lw		
4.	addi	sub	addi	lw	
5.	sw	addi	sub	addi	lw
7.	bnez	sw	addi	sub	addi
8.	<i>delay slot</i>	bnez	sw	addi	sub

Anzahl der Schleifendurchläufe: $1000/4 = 250$

\Rightarrow Anzahl der erforderlichen Taktzyklen: $250 \times 8 \text{ Takte} + 4 \text{ Takt} = 2004 \text{ Takte}$

Die Pipeline wird nach jedem Sprungbefehl für einen Takt angehalten (*delay slot*)