1. Pipelines

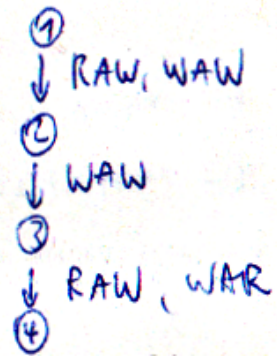
a) Eine 5-Stufen-Pipeline kann die Rechengeschwindigkeit nicht ver-5-fachen, weil

- Befehle im „Pipeline-losen" Prozessor nicht alle (theoretischen) Pipeline-Stufen durchlaufen, z. B. die Memory-Phase
- sich in der Pipeline die Länge eines Zyklus danach richtet, welche der Pipeline-Phasen am langsamsten ist – in „schnelleren" Phasen entstehen dadurch Wartezeiten
- i.d.R. Pipeline-Hemmnisse die Ausführung zusätzlich verzögern.

b) Man kann auch keine Ver-5-fachung erreichen, indem man ein spezielles („gutartiges") Programm schreibt. Durch Wahl geeigneter Befehle kann man zwar Pipeline-Hemmnisse umgehen, aber die übrigen Gründe für eine Verlangsamung (s. o.) bleiben.

2. Abhängigkeitsgraph, Reorder Buffer

```
1    FMUL $2, $2, $1
2    FADD $2, $2, $3
3    SET  $2, $4
4    FMUL $4, $2, $2
```



a) Abhängigkeitsgraph:

b) Ausführung mit Reorder Buffer

t_0

			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL $2,$2,$1	MUL	▶ 1	$2	$1			$2 (ung.)
FADD $2,$2,$3	FPU	‖	—	$3	FMUL		$2 (ung.)

t_1

			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL	MUL	▶ 2	$2	$1			$2 (u.)
FADD	FPU	‖	—	$3	FMUL		$2 (u.)
SET $2,$4	INT	▶ 1	$4	X			$2 (u.)

t_2

			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL	MUL	▶ 3	$2	$1			$2 (u.)
FADD	FPU	‖	—	$3	FMUL		$2 (u.)
SET $2,$4	—	✓	$4	X			$2

t_3

			Operanden				
Befehl	unit	Status	Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL	MUL	▶ 4	$2	$1			$2 (u.)
FADD	FPU	‖	—	$3	FMUL		$2 (u.)
SET $2,$4	—	✓	$4	X			$2

t_4

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL	—	✓	$2	$1			$2
FADD	FPU	1	$2*	$3			$2 (u.)
SET	—	✓	$4	X			$2
FMUL $4,$2,$2	MUL	1	$2*	$2*			$4 (u.)

t_5

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FADD	FPU	2	$2*	$3			$2 (u.)
SET	—	✓	$4	X			$2
FMUL	MUL	2	$2*	$2*			$4 (u.)

t_6

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FADD	FPU	3	$2*	$3			$2 (u.)
SET	—	✓	$4	X			$2
FMUL	MUL	3	$2*	$2*			$4 (u.)

t_7

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FADD	FPU	4	$2*	$3			$2 (u.)
SET	—	✓	$4	X			$2
FMUL	MUL	4	$2*	$2*			$4 (u.)

t_8

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FADD	—	✓	$2*	$3			$2
SET	—	✓	$4	X			$2
FMUL	—	✓	$2*	$2*			$4

t_9

Befehl	unit	Status	Operanden Q_Y	Q_Z	V_Y	V_Z	Ergebnis
FMUL	—	✓	$2*	$2*			$4

(Zum Zeitpunkt t_{10} ist der Reorder Buffer dann leer.)

3. Ändern der Reihenfolge

a)
```
FMUL   yk,xk,yk
SETH   xk,#4000   2,0 (Gleitkommawert!)
FMUL   yk,yk,xk   2 × y_k
```

Nein. SETH oben: → $4\,yk$
SETH unten: → $yk \cdot xk^2$

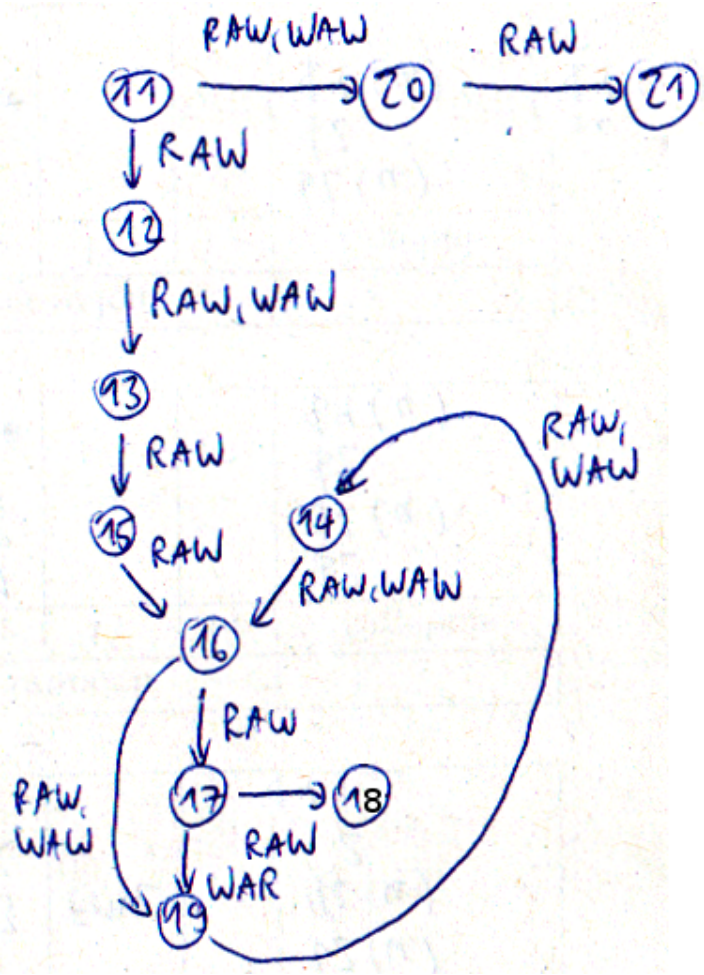
b)
```
FMUL   yk,xk,yk
SETH   temp2,#4000   2,0 (Gleitkommawert!)
FMUL   yk,yk,temp2   2 × y_k
```

Ja
ok

Alle Varianten, bei denen SETH vor FMUL..temp2 bleibt, sind ok

4. Superskalarität

```
1           PREFIX  :CRC:
2 GenP      GREG    #0000000104C11DB2
3 degree    IS      32
4 x         IS      $0
5 crc       IS      $1
6 b         IS      $2
7 msb       IS      $3
8 k         IS      $4
9 counter   IS      $5
10
11 :CRC     SET     k,7
12 2H       SRU     x,counter,k
13          AND     x,x,1
14          SLU     crc,crc,1
15          SLU     b,x,degree
16          XOR     crc,crc,b
17          SRU     msb,crc,degree
18          BZ      msb,weiter
19          XOR     crc,GenP,crc
20 weiter   SUB     k,k,1
21          BNN     k,2B
22          POP     1,0
```



Schwierigkeit 1: Schleife führt zu Kreisen im Abhängigkeitsgraphen. Ob eine Abhängigkeit „im Kreis“ besteht, hängt davon ab, ob gesprungen wird oder nicht...

Für Klausur: keine Schleifen.

Schwierigkeit 2: transitiver Pfeil von 16 nach 19. Hier eingezeichnet, weil „echter“ Konflikt (RAW), während 17/19 nur WAR.

Noch schlimmer: a —WAR→ b —WAR→ c und a —RAW→ c, z. B.

```
01          ADD   a,b,c
02          ADD   c,d,e
03          ADD   e,f,a
```

WAR, WAR, RAW

Die WAR-Abhängigkeiten bekommt man komplett durch Unbenennen weg:

```
01          ADD   a,b,c
02          ADD   x,d,e
03          ADD   y,f,a
```

RAW

aber die RAW-Abhängigkeit bleibt.