Freie Universität Berlin

Übung 08

Rechnerarchitektur
Wintersemester 20/21
Prof. Dr.-Ing. Jochen Schiller

Abgabe 29.01.2021, 10:15

Aufgabe 1: Tomasulos Algorithmus

Gegeben sei folgende Floating Point Unit:

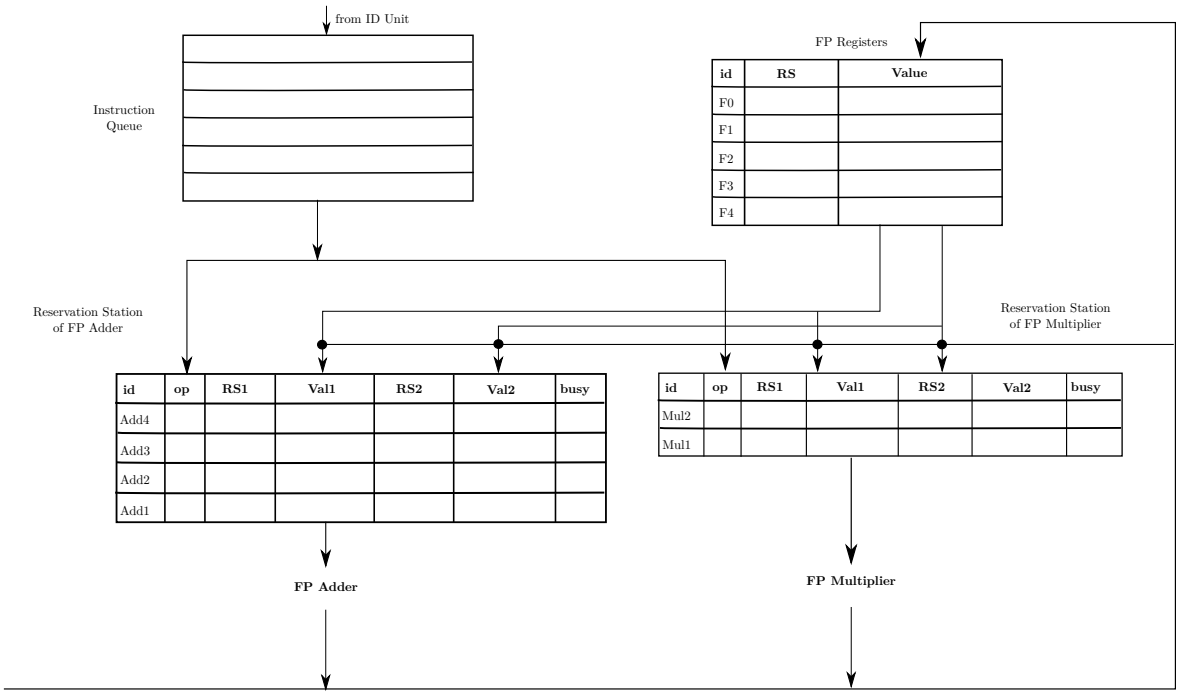


Abbildung 1: Tomasulo-Schema

1. Es handelt sich um einen Single Issue Processor, Instruktionen werden ausschließlich in-order ausgegeben.
2. Die Instruction Queue kann bis zu 6 Instruktionen fassen.
3. Es können bis zu 2 Instruktionen parallel geholt und dekodiert werden.
4. Es gibt 4 FP Register. Diese seien initialisiert mit $F0 = 1,0$, $F1 = 2,0$, $F2 = 3,0$, $F3 = 4,0$, $F4 = 5,0$.
5. Es gibt 1 FP Addierer mit 4 Reservation Stations.
6. Es gibt 1 FP Multiplizierer mit 2 Reservation Stations.
7. Es findet keinerlei spekulative Ausführung statt.
8. Das Holen der Operanden aus dem Register File benötigt 1 Takt. Das heißt: Die Ausführung einer Instruktion, die in eine freie Reservation Station gelangt, kann erst im darauffolgenden Takt beginnen, auch wenn alle Operanden ohne Abhängigkeiten verfügbar sind.
9. Additionen und Subtraktionen benötigen jeweils 2 Takte, Multiplikationen 4 Takte.
10. Das Ergebnis einer Instruktion wird im darauffolgenden Takt auf den Common Data Bus gegeben und sofort in das Register File und/oder wartende Reservation Stations geschrieben.

Aufgaben Gegegeben ist folgendes Codebeispiel:

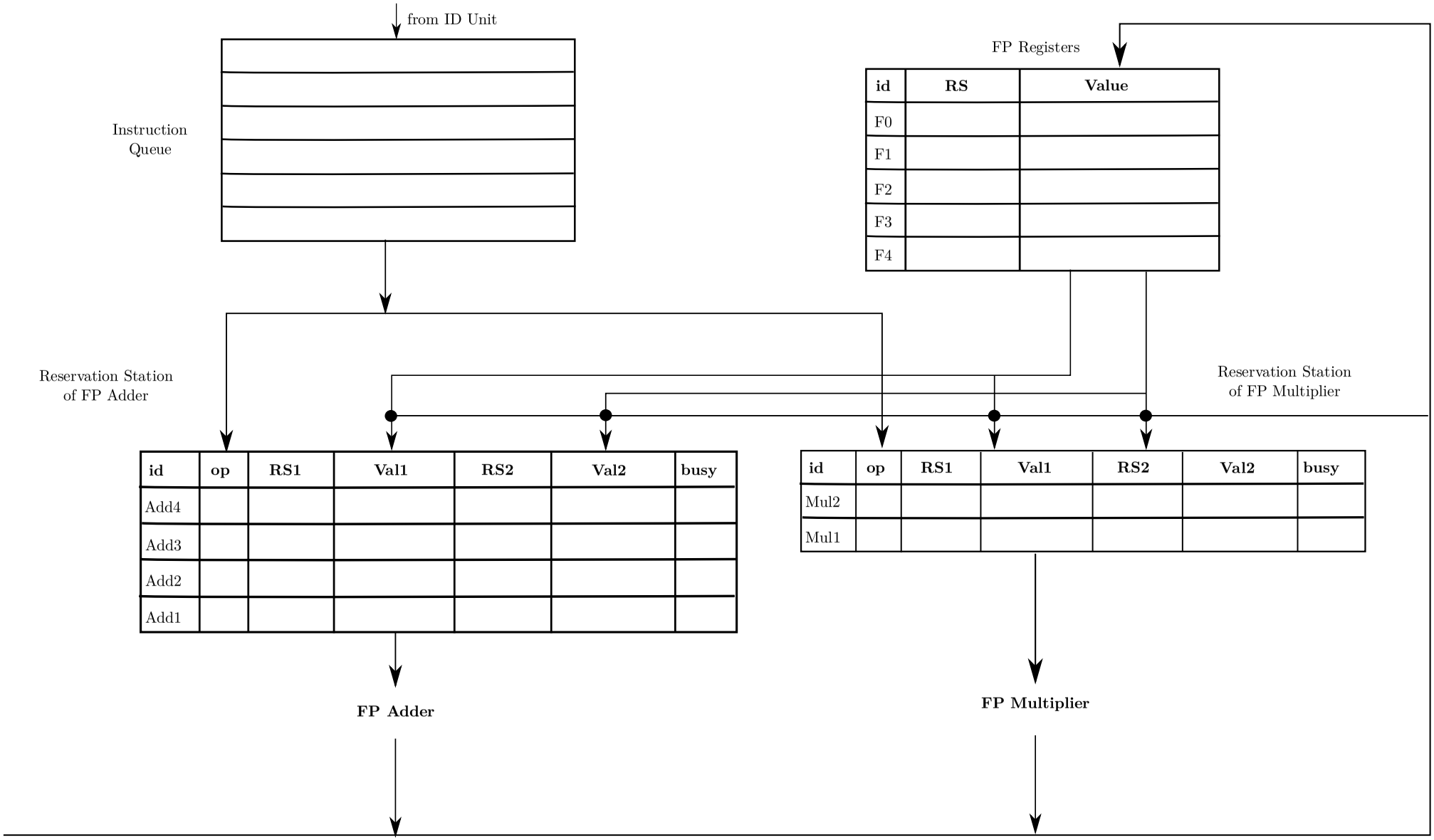
```
ADD.D F1, F2, F0
SUB.D F3, F1, F0
MUL.D F1, F2, F3
SUB.D F4, F2, F0
```

Hinweis: `ADD.D A, B, C` bedeutet `A = B + C`

a) Zeigen Sie die einzelnen Schritte des Dynamic Scheduling dieser Instruktionsfolge unter Anwendung von Tomasulos Algorithmus. Verwenden Sie dazu die folgende Grafik (vervielfältigen Sie diese geeignet) und tragen Sie die Zustände der Instruction Queue, aller Reservation Stations und des Register Files *für jeden einzelnen Takt* ein. Findet in einem Takt keine Änderung statt (z. B. weil nur gerechnet wird), müssen Sie hierfür die Grafik nicht erneut ausfüllen.

b) Wie viele Takte benötigt die vollständige Abarbeitung dieser Instruktionsfolge, bis alle Berechnungsergebnisse ins Register File eingetragen sind?

Bei Fragen zur Tabelle wenden Sie sich bitte an Ihren Tutor.


from ID Unit
Instruction Queue
FP Registers
id
RS
Value
F0
F1
F2
F3
F4
Reservation Station of FP Adder
Reservation Station of FP Multiplier
id
op
RS1
Val1
RS2
Val2
busy
Add4
Add3
Add2
Add1
id
op
RS1
Val1
RS2
Val2
busy
Mul2
Mul1
FP Adder
FP Multiplier

Aufgabe 2: Superskalare Pipeline

Gegeben sei folgende superskalare Pipeline:

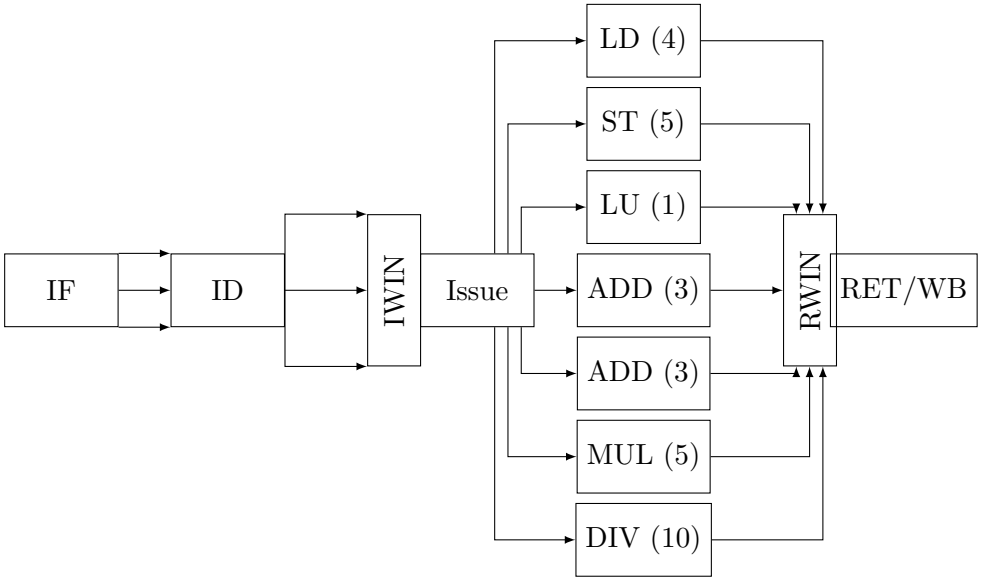


Abbildung 2: Pipeline-Schema

1. In der ersten Stufe (Instruction Fetch, IF) können pro Takt drei Befehle gleichzeitig geladen werden.
2. Die zweite Stufe (Instruction Decode, ID) kann ebenfalls pro Takt drei Befehle gleichzeitig dekodieren.
3. Dekodierte Befehle befinden sich anschließend architekturell bedingt mindestens einen Takt im Instruction Window (IWIN), welches maximal fünf Instruktionen vorhalten kann. Hier werden, sofern nötig, auch Operanden parallel geladen (Operand Fetch, OF).
4. Danach werden die Befehle durch die Issue-Einheit auf die entsprechenden (freien) Ausführungseinheiten aufgeteilt. Die Issue-Einheit benötigt dafür keine zusätzliche Zeit und kann bis zu drei Befehle aus IWIN gleichzeitig verteilen (Tripple Issue Unit).
5. Die Pipeline besitzt sieben verschiedene Ausführungs-Einheiten (EXE):

   Load (LD) zum Laden von Daten aus dem Speicher.
   Store (ST) zum Speichern von Daten in den Speicher.
   Logical Unit (LU) für logische / binäre Operationen, bspw. Schiebe-Operationen.
   Zwei Addierer (ADD) für Addition und Subtraktion.
   Multiplikator (MUL) für Multiplikationen.
   Dividierer (DIV) für Divisionen.

6. Anschließend stellt die Retirement/Write-Back-Einheit (RET/WB) die korrekte Reihenfolge der Befehle wieder her und schreibt eventuell das Ergebnis in die Ergebnis-Register. Der dazu nötige Puffer (Retire Window, RWIN) ist ausreichend groß. Einkommende Ergebnisse werden soweit möglich sofort verarbeitet, müssen also nicht zwingend im RWIN warten.
7. Es gibt kein Forwarding/Shortcuts zwischen den Ausführungs-Einheiten. Ein Ergebnis kann allerdings weiter verwendet werden, sobald es mind. 1 Takt in RWIN vorlag.

Einheit	Takte	Kapazität
IF	1	max. 3 Befehle / Takt
ID	1	max. 3 Befehle / Takt
IWIN	mind. 1	5 Plätze
Issue	0	3 Befehle / Takt
LD	4	1 Einheit
ST	5	1 Einheit
LU	1	1 Einheit
ADD	3	2 Einheiten
MUL	5	1 Einheit
DIV	10	1 Einheit
RWIN	mind. 0	10 Plätze
RET/WB	1	10 Befehle / Takt

Abbildung 3: Takte und Kapazitäten der jwg. Einheiten

Aufgaben Gegegeben ist folgendes Codebeispiel:

```
LD  A, [0x0AFFE0]
SHL B, 3
ST  B, [0xCAFFEE]
ADD E, F
SHR F, 1
ADD F, 7
SUB B, 3
SHL G, 1
MUL G, G
```

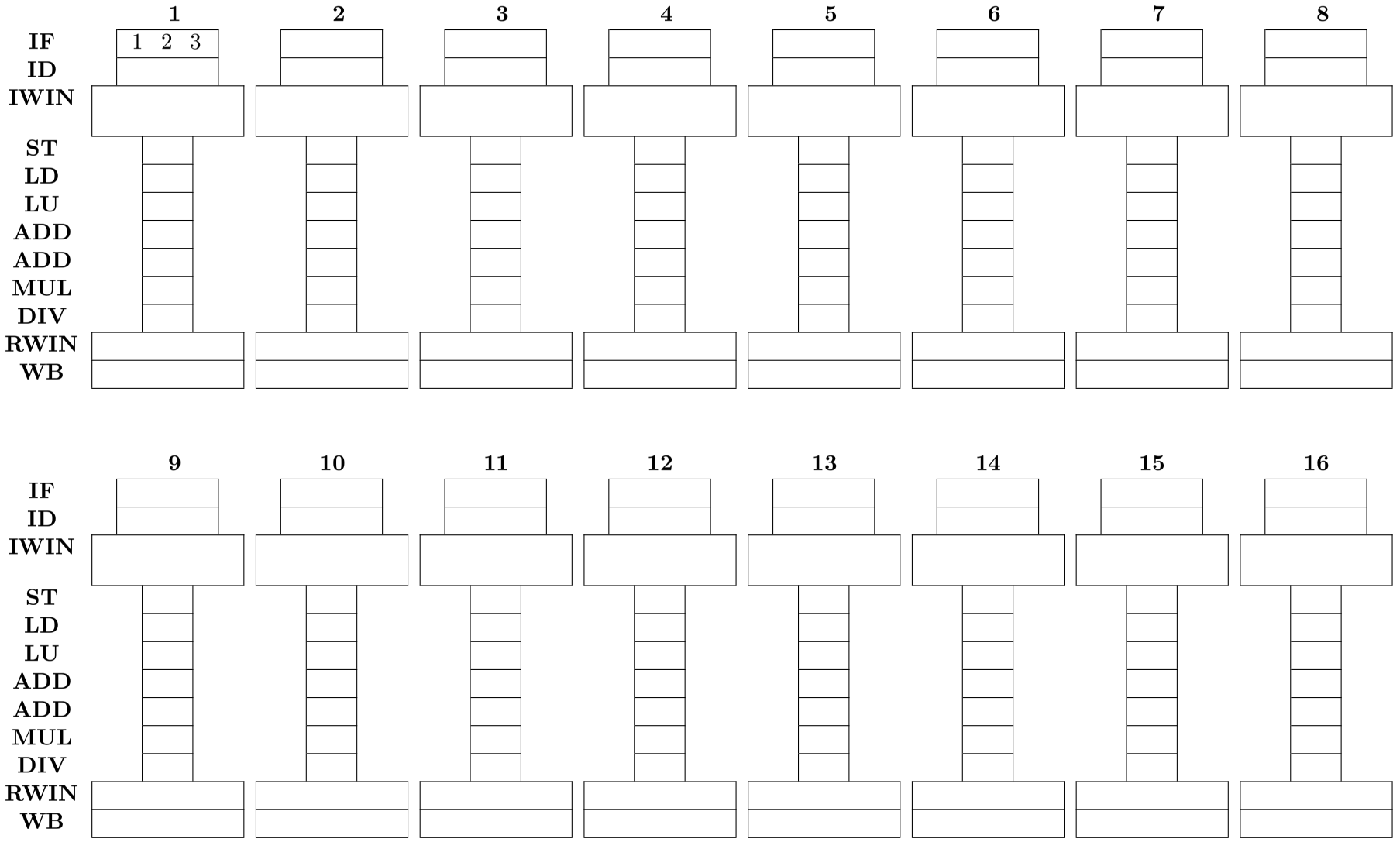
Hinweis: `ADD A, B` bedeutet `A = A + B`

1. Zeigen Sie eine mögliche Belegung der in-/out-of-order Ausführung des obigen Code-Beispiels anhand der angehangenen Tabelle auf. Wenden Sie pro Takt **first-come, first-served (FCFS)** in der Issue Einheit an. Tragen Sie dazu die Zeilennummern der entsprechenden Befehle ein. Sie können Annehmen, dass sowohl IWIN als auch alle EXE Einheiten zu Beginn leer sind.

2. Welchen Speed-up haben Sie mit Einführung dieser Pipeline gegenüber einer skalaren Pipeline mit nur einer (mächtigen) EXE Einheit erreicht? Nehmen Sie dazu an, Sie benötigen die gleiche Taktzahl pro Befehl wie bei dieser superskalaren Pipeline.

3. Warum ist Ihre Pipeline nicht optimal ausgelastet? Welche weiteren Maßnahmen führen zu einer (signifikant) größeren Beschleunigung?

Bei Fragen zur Tabelle wenden Sie sich bitte an Ihren Tutor.


1
2
3
4
5
6
7
8
IF
1 2 3
ID
IWIN
ST
LD
LU
ADD
ADD
MUL
DIV
RWIN
WB
9
10
11
12
13
14
15
16
IF
ID
IWIN
ST
LD
LU
ADD
ADD
MUL
DIV
RWIN
WB