Rheinische Friedrich-Wilhelms Universität Bonn
Institut für Informatik VI, Technische Informatik

Prof. Dr. Joachim K. Anlauf
Oguzhan Sezenlik

10. Übung für die Vorlesung
Rechnerorganisation

Sommersemester 2019

Abgabe: Donnerstag, 4.7.2019

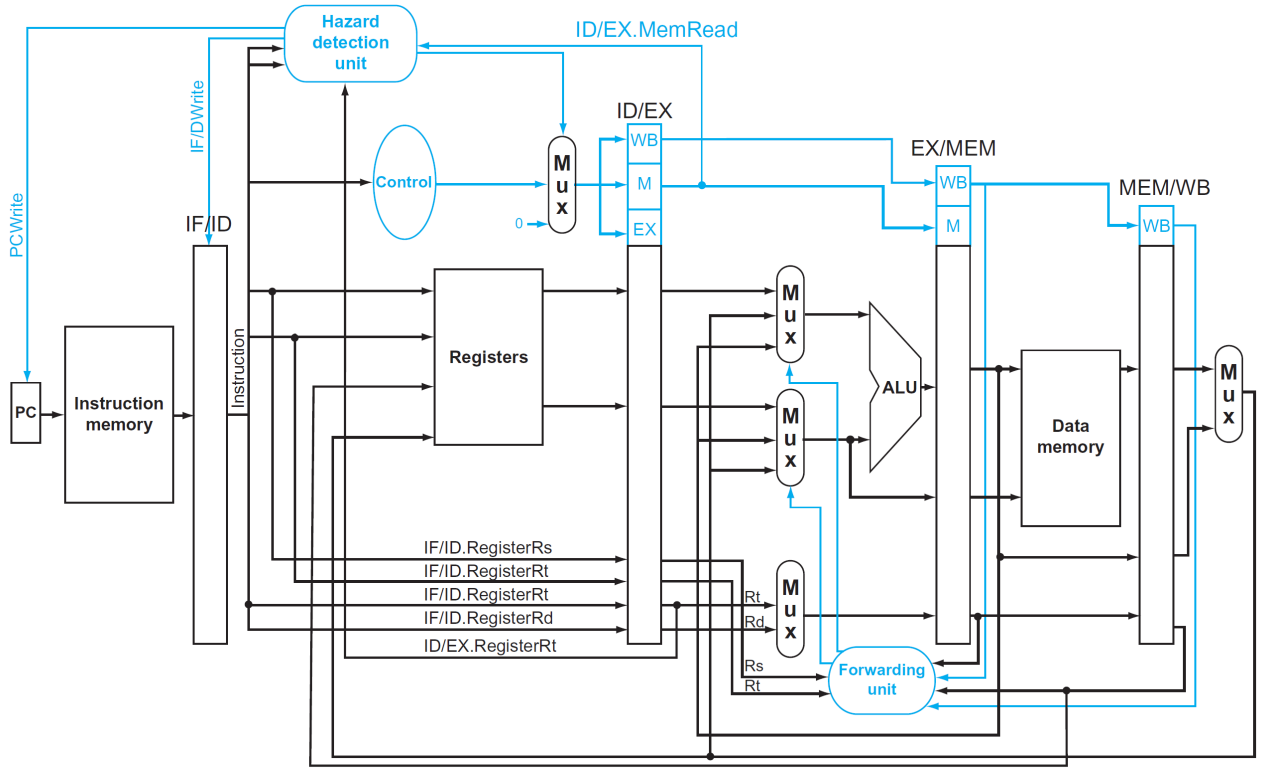
Wichtige Hinweise zum Übungsbetrieb:

1. Dieser Übungszettel wird nicht mehr für die Klausurzulassung gewertet, der Inhalt ist weiterhin prüfungsrelevant!
2. Aufgaben 1 bis 3 werden zusammen mit dem 9. Zettel in den Tutorien am Montag 1.7. bzw. Dienstag 2.7. besprochen. Die Lösungen zu den Aufgaben 4 und 5 können am 4.7. wie gewohnt vor Vorlesungsbeginn abgegeben werden. Diese werden dann von den Tutoren korrigiert und schließlich in den Tutorien am Mo 8.7. bzw. Di 9.7., zusammen mit dem 11. Übungszettel besprochen.
3. Alle Tutorien in den nächsten zwei Wochen (am 1.7. und 2.7. sowie am 8.7. und 9.7.) beginnen und enden pünktlich zur vollen Stunde (s.t.) und dauern damit 120 Minuten.

Anwesenheitsaufgaben für die Tutorien am Mo 1.7. bzw. Di 2.7.

Aufgabe 1. *Forwarding-Unit* 0 P.

1. Betrachten Sie den Datenpfad in der folgenden Abbildung. Listen Sie alle Ein- und Ausgänge der Forwarding-Unit und deren Bitbreiten auf.
2. Entwerfen Sie die Hardware für die Forwarding Unit und geben Sie den vollständigen Schaltplan an.



Aufgabe 2. *Branch Delay Slot* 0 P.

Modifizieren Sie folgenden Code, so dass ein `delayed branch slot` sinvoll genutzt wird:

```
Loop:  lw   $2, 100 ($3)
       addi $3, $3, 4
       beq  $3, $4, Loop
       nop
```

Aufgabe 3. *Stall Flush Konflikt* 0 P.

Betrachten Sie den Datenpfad mit Pipeline in Abbildung 1.

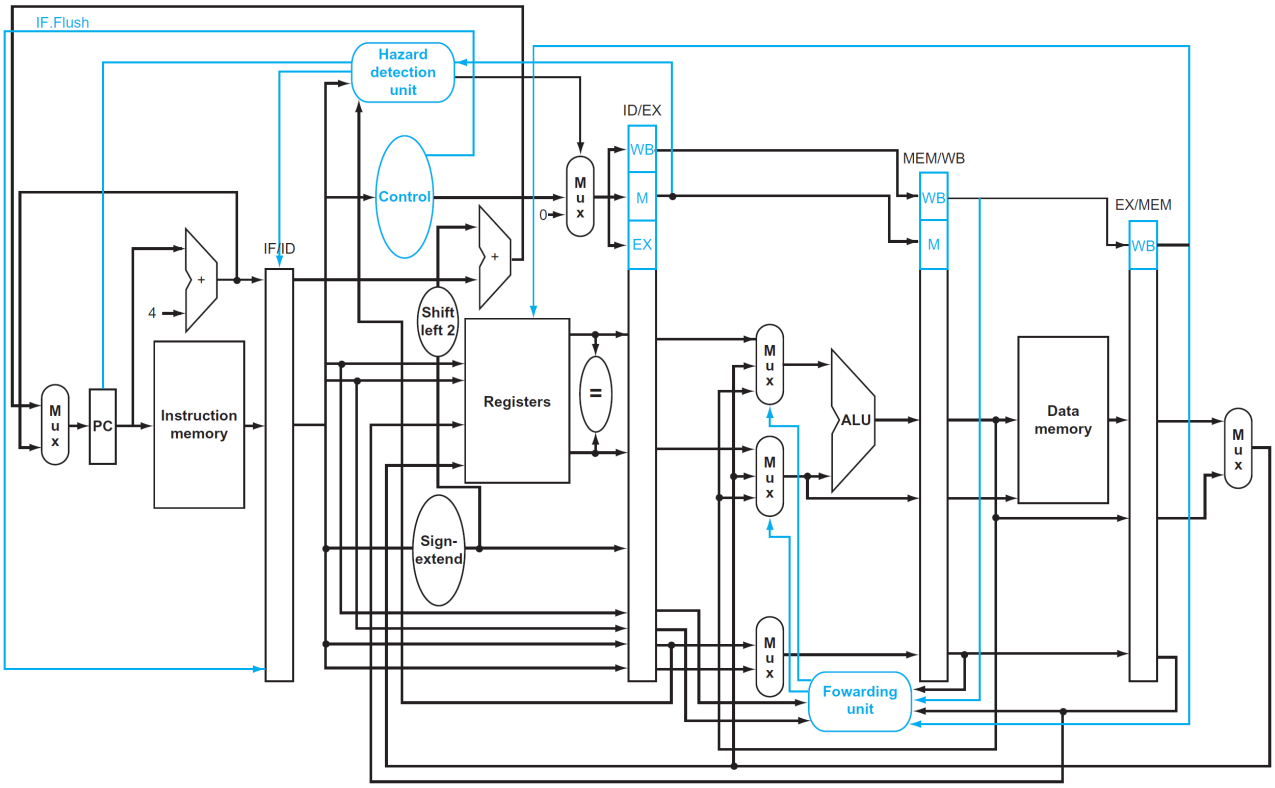


Abbildung 1: MIPS-Datenpfad

Zeigen Sie anhand eines kurzen MIPS-Assembler-Codes, wann ein Stall zeitgleich mit einem Flush auftreten kann. Verursacht dies einen Konflikt, wenn ja, wie ist dieser aufzulösen?

Abgabe am 4.7. und Besprechung am Mo 8.7. bzw. Di 9.7.

Aufgabe 4. *Pipeline-Architektur: Pipeline Hazards* 0 P.

Betrachten Sie den folgenden Programmausschnitt:

```
loop:  addi $t3, $t3, 4
       lw   $t2, 0($t3)
       beq  $t2, $zero, loop
       add  $t1, $t1, $t1
       slt  $t4, $t2, $t4
```

Nehmen Sie an, dieser Code würde auf einer MIPS Architektur mit 5-stufiger Pipeline, einer Forwarding-Unit und Unterstützung von Branch-Delay-Slots (siehe Abbildung 1) ausgeführt.

1. Welche Hazards treten auf? Wieviele Zyklen werden pro Schleifendurchlauf benötigt? Wie können die Hazards durch den Compiler naiv aufgelöst werden?

2. Erhöhen Sie die Performance durch Umordnung/Anpassung der Befehle. Wieviele Zyklen benötigt der verbesserte Code pro Schleifendurchlauf?

3. Gibt es eine semantisch äquivalente Implementation, die mit 3 Zyklen pro Schleifendurchlauf (eventuell plus eine konstante Anzahl von Anweisungen) auskommt? Wenn ja, geben Sie diese an und zeigen Sie ihre Korrektheit. Wenn nein, begründen Sie Ihre Antwort.

Aufgabe 5. *Branch Prediction Buffer* 0 P.

Gegeben sei der MIPS-Prozessor mit Pipeline, Operand-Forwarding und Hazard-Detection. Nehmen Sie desweiteren an, dass die Sprungentscheidung in der EX-Phase getroffen wird.
Betrachten Sie drei verschiedene Strategien: “Branch never taken”, “Branch always taken” und “2-Bit Branch prediction buffer”.

1. Vervollständigen Sie die Tabelle 1 mit den Pipeline-Verzögerungen für die verschiedenen Strategien. Diskutieren Sie die Vor- und Nachteile der verschiedenen Ansätze.

Strategie	Vorhersage	Sprung	Verzögerung
never	not taken	taken	
		not taken	
always	taken	taken	
		not taken	
BPB	taken	taken	
		not taken	
	not taken	taken	
		not taken	

Tabelle 1: Pipeline-Verzögerung bei Sprungbefehl

2. Erweitern Sie den MIPS-Datenpfad aus Abbildung 1, um einen 2-Bit Branch-Prediction-Buffer mit 8 Einträgen. Die Sprungvorhersage soll in der ID-Phase getroffen werden. Ob die Bedingung eines Branch-Befehls erfüllt ist, soll in der EX-Phase wieder von der ALU berechnet werden (*Hinweis: Die dazu notwendige* `Zero` *Leitung aus der ALU wird ohne Umwege über Pipeline-Register genutzt).*

   Erläutern Sie welche Steuersignale dazu erzeugt werden müssen und geben sie den vollständigen Schaltplan des Branch-Prediction-Buffers, sowie die Bitbreiten aller neuen Datenbusse an.