Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Tutoriumsblatt 13
Rechnerarchitektur im SoSe 2021

Zu den Modulen P,Q

Aufgabe Ü35: Arbeitsweise Caches

(– Pkt.)

Nehmen Sie einen Speicher mit 64 und einen Cache mit 16 Blöcken an. Wir benötigen nacheinander folgende Adresszugriffe bei anfangs leerem Cache:

```
1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9, 17
```

a. Geben Sie für jede Referenz an, ob ein Cache-Hit oder ein Cache-Miss eintritt. Gehen Sie dabei von dem in der Vorlesung eingeführten Direktabbildungs-Verfahren aus.

b. Stellen Sie den Inhalt des Caches dar, nachdem alle Zugriffe erfolgt sind.

c. Wie viel Speicherplatz ist erforderlich, um einen Direct-mapped Cache zu realisieren, der 256 KByte Daten zwischenspeichern kann, wenn die Größe jedes Cache-Blocks und jedes Datenwortes im Speicher 32 Bit = 4 Byte beträgt. Gehen Sie von 32-Bit Adressen aus (es werden ganze Datenworte adressiert). Hinweis: Jeder Cache-Block benötigt ein Validierungs-Bit und ein geeignetes Tag!

Aufgabe T36: Pipeline

(– Pkt.)

Beantworten Sie die folgenden Fragen zum Thema Pipelining.

a. Erklären Sie die prinzipielle Idee von Pipelining.

b. Wie bestimmt man die Länge einer Pipelinestufe, d.h. wie lang muss das Zeitintervall (Taktzyklus) für eine Stufe mindestens sein?

c. Warum müssen die Pipelinestufen eine gleich lange Ausführungszeit besitzen?

d. Von welchen zwei Eigenschaften hängt der Leistungsgewinn einer idealen Pipeline ab (also ohne Berücksichtigung von Konflikten)?

e. Nennen Sie zwei Gründe, Warum es keinen Sinn macht bzw. nicht möglich ist, die Anzahl der Stufen beim Design einer Pipeline beliebig hoch zu wählen?

f. Benennen und erläutern Sie die drei verschiedenen Arten von Konflikten (Hazards), die durch die Einführung von Pipelining entstehen können? Geben Sie je ein Beispiel für diejenigen Hazards an, die bei der MIPS-Architektur auftreten können. Begründen Sie kurz, warum ein Hazard im Falle der MIPS-Architektur nicht entstehen kann,

g. Bei der folgenden Frage wird von einer 5–stufigen Pipeline mit einer Ausführungszeit von 2ns pro Stufe ausgegangen. Folgende Ausführungszeiten sind gegeben:

Befehl	IF	ID	EX	MA	WB
load word (lw)	2 ns	1 ns	2 ns	2 ns	1 ns
store word (sw)	2 ns	1 ns	2 ns	2 ns	—
add	2 ns	1 ns	2 ns	—	1 ns
Branch (beq)	2 ns	1 ns	2 ns	—	—

Die Stufen haben folgende Bedeutung:

IF (Instruction Fetch): Laden des nächsten Befehls aus dem Speicher.

ID (Instruction Decode and register file read): Lesen der Register und gleichzeitiges Entschlüsseln des Befehls. (Das Format der MIPS-Befehle ermöglicht das gleichzeitige Lesen und Entschlüsseln.)

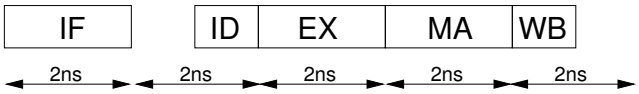
EX (EXecution or adress calculation): Ausführung der Operation oder Berechnung einer Adresse.

MA (data Memory Access): Zugriff auf einen Operanden im Datenspeicher.

WB (Write Back): Schreiben des Ergebnisses in ein Register.

Gehen Sie davon aus, dass das Schreiben eines Registersatzes (WB-Stufe) in der ersten Hälfte des entsprechenden Taktzyklus erfolgt und das Lesen aus dem Registersatz (ID-Stufe) in der zweiten Hälfte.

Die Pipeline sieht damit folgendermaßen aus:



Gegeben sei nun folgendes Programmfragment:

```
1 lw      $2, 100($5)
2 add     $3, $3, $4
3 add     $1, $4, $5
```

(i) Wie lang dauert die Ausführung dieses Programmfragments **ohne** Pipelining?

(ii) Wie lang dauert die Ausführung dieses Programmfragments **mit** Pipelining?

h. Ohne jegliche Gegenmaßnahme kommt es in folgendem Programmfragment offensichtlich zu Konflikten.

```
1 sw      $2, 100($3)
2 addi    $3, $3, 4
3 add     $4, $4, $2
4 beq     $2, $3, 200
```

(i) Lösen Sie zunächst die entsprechenden Konflikte, indem Sie die Pipeline für die dargestellete Befehlsfolge skizzieren und an den entsprechenden Stellen Pipelineleerläufe (Stalls) einfügen.

(ii) Wie würde ihre Pipeline aussehen, wenn Sie von Gegenmaßnahmen wie z.B. Bypassing und Frowarding Gebrauch machen? Begründen Sie Ihre Antwort.

(iii) Gelangt man an die Ausführung des beq-Befehls, kann der nächste Befehl erst geladen werden, wenn das Ergebnis der EX-Stufe des beq-Befehls bekannt ist. Eine Möglichkeit hier Pipelinestufen zu sparen, sellt der *Delayed Branch* dar. Erklären Sie den Begriff Delayed Branch und modifizieren Sie das Programmfragment so, dass es einen Delayed Branch verwendet. Die Semantik des Programmfragments darf sich dabei nicht ändern und es dürfen keine neuen Konflikte entstehen!