georg.von-der-brueggen [☺] tu-dortmund.de
ulrich.gabor [☺] tu-dortmund.de
olaf.neugebauer [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerstrukturen
Wintersemester 2017

Übungsblatt 12 (Block C - 4)

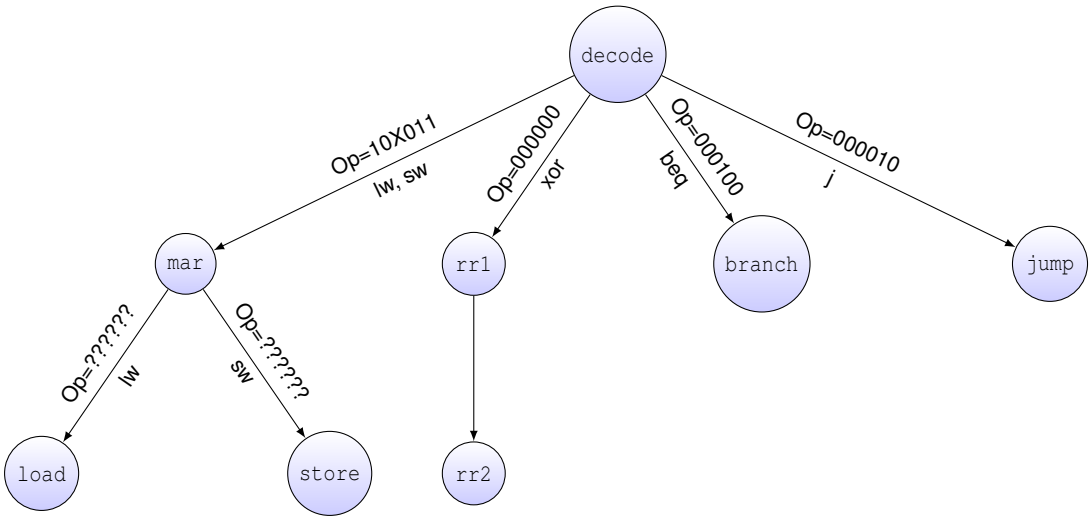
(16 Punkte)

Abgabe bis spätestens Mittwoch, 24. Januar 2018, 16:00 Uhr.
Besprechung ab Montag, 29. Januar 2018.

Hinweise zur Abgabe der Übungsblätter finden Sie am Ende des Dokuments

12.1 Mikroprogrammierung (4 Punkte)

Für die Dekodierung von MIPS Befehlen benötigt das Steuerwerk den Befehlscode aus dem Befehlsregister. Wir betrachten im Folgenden die Dekodierung der MIPS Befehle `lw`, `sw`, und `xor`:



Laden Sie in MARS die folgende Befehlsfolge und übersetzen Sie diese mit „Run“ → „Assemble“:

```
.data          # Prueft ob Eingabe gleich Null ist
ein: .word 2   # Eingabewert vom User
erg: .word 99 # Wenn Eingabewert gleich 0 dann 1 sonst 0
.text
.globl main
main:
  lw $2, ein
  beq $2, $0, gleich
  xor $3, $2, $2 # Setzt Register 3 auf 0
  j fertig
gleich:
  addi $3, $0, 1 # Setzt Register 3 auf 1
fertig:
  sw $3, erg
  li $2, 10
  syscall
```

Betrachten Sie im Tab „Execute“ im „Text Segment“ die Spalte „Code“. Dort finden sich die Inhalte des Befehlsregisters für die einzelnen Befehle. Für `lw $2,0x00000000($1)` ist das Befehlsregister 0x8c220000, für `sw, $3,0x00000004($1)` ist es 0xac230004 und für `xor $3,$2,$2` ist es 0x00421826.

In diesen Codes ist u.a. angegeben welcher Befehl ausgeführt werden soll, welche Operanden verwendet werden, wohin das Ergebnis gespeichert werden soll und welche arithmetische oder logische Operation ausgeführt werden soll.

a. Wie ergeben sich aus dem Befehlsregister die Befehlscodes (Opcodes) für das Steuerwerk und wie lauten diese für `lw` und `sw`?

   Hinweis: Siehe Vorlesungsfolien[1] auf Seite 16 und 33.

b. Für `xor $3,$2,$2` ergibt sich für das Befehlsregister:

$$(00421826)_{16} = (0000\ 0000\ 0100\ 0010\ 0001\ 1000\ 0010\ 0110)_2.$$

   Markieren Sie anhand der angegebenen Binärdarstellung vom Befehlsregister die Bits für:

   - den Befehlscode (Opcode),
   - das Ergebnis der `xor` Rechnung (Zielregister `$3`),
   - den ersten Operanden (Register `$2`) und
   - den zweiten Operanden (ebenfalls Register `$2`).

   Hinweis: Siehe Vorlesungsfolien auf Seite 24 und 26.

12.2 Paging (4 Punkte)

Die fünf Seiten $3-7$ sollen in einem Speicher mit drei Kacheln 0, 1 und 2 verwaltet werden. Zu den ersten drei Zeitpunkten werden die leeren Kacheln mit den Seiten 3, 4 und 5 gefüllt.

Zeitpunkt	1	2	3	4	5	6	7	8	9	10	11	12
Seitenzugriff	3	4	5	7	6	5	4	7	6	5	7	4
Kachel 0	3	3	3									
Kachel 1	-	4	4									
Kachel 2	-	-	5									

Führen Sie diese Tabelle ab Zeitpunkt 4 fort. Die folgenden Seitenzugriffe sind in der Zeile „Seitenzugriff“ vorgegeben. Zur Bestimmung der Kachel, aus der die Seite zugunsten der neuen Seite ausgelagert werden soll, verwenden Sie den LRU-Algorithmus (*Least Recently Used*).

[1] http://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/ws1617/rs/rs2.3-microarch.pdf

12.3 Assembler-Analyse (4 Punkte)

Im Folgenden ist ein Assemblerprogramm zur Berechnung des Zinseszins gegeben. Dieser berechnet aus dem Startkapital k und dem Zinsfuß p (z.B. 3 bei 3% Zinsen) das Kaptial k_n nach n Jahren basierend auf der folgenden rekursiven Formel:

$$k_0 := k$$
$$k_n := k_{n-1} \cdot \left(\frac{100+p}{100} \right).$$

Das Assemblerprogramm weicht von der Formel ab, da es iterativ und nur mit ganzen Zahlen rechnet.

```
.data               # Zinseszinsrechner
k: .word 518        # Startkapital
p: .word 3          # Zinsfuss (= Zinssatz in Prozent * 100)
n: .word 3          # Laufzeit in Jahren
x: .word 100        # Eine Zehnerpotenz

.text
main:
  lw $2, k          # Reg[2] := Speicher[k]
  lw $3, p          # Reg[3] := Speicher[p]
  lw $4, n          # Reg[4] := Speicher[n]
  lw $5, x          # Reg[4] := Speicher[n]
  mul $2,$5,$2      #
  addi $3,$3,100    #
schleife:
  beqz $4, ende     # Sprung bei Durchlauf:
  mul $2, $2, $3    #
  div $2, $2, 100   #
  subi $4, $4, 1    #
  j schleife
ende:
  div $2, $2, $5    #
  sw $2, k          #
  li $2, 10
  syscall
```

a. Geben Sie die Wirkung der Befehle in symbolischer Register-Transfer-Notation (z.B. `Reg[3] := Reg[4] + Reg[5]`) hinter den # an. Geben Sie beim Befehl `beqz` auch an, im wievielten Durchlauf der Sprung zum Ende ausgeführt wird.

b. Welchen Wert hat `Speicher[k]` nach Ausführung des Programms?

c. Wozu dient die Eingabe `x`?

12.4 Translation Look-Aside Buffer (TLB) (4 Punkte)

Gegeben sei eine TLB-Architektur mit *direct mapping*. Für diese Architektur sei eine Folge von virtuellen Adressen gemäß der nachfolgenden Tabelle gegeben. Der Eintrag „X“ für den Offset bedeutet, dass die Werte für diese Aufgabe nicht relevant sind.

Tragen Sie in der Tabelle in die Spalte „Treffer“ ein, ob Sie einen Treffer im TLB haben. Tragen Sie für jede Zeile ohne Treffer im TLB den Inhalt des *Tag*-Feldes nach der Ausführung der Adressumrechnung für die links angegebene Adresse ein. Leere Felder bedeuten: „derselbe Wert, wie im Feld darüber“. ? bedeutet „unbekannt“.

Virtuelle Adressen			Treffer	Inhalte der Tag-Felder			
Tag	Index	Offset	(ja/nein)	Index=00	Index=01	Index=10	Index=11
				?	?	?	?
0001	00	X	nein	0001			
0000	11	X	nein				0000
0001	00	X					
1101	01	X					
1010	01	X					
0001	00	X					
1010	10	X					
1010	01	X					
1111	11	X					
0010	00	X					

Hinweise:

Die Abgaben sollen bis Mittwoch, 24. Januar 2018, 16:00 Uhr in die Briefkästen in der Otto-Hahn-Straße 12 eingeworfen werden.

Die Briefkästen finden Sie in der ersten Etage der Otto-Hahn-Straße 12 am Übergang zum Erdgeschoss der Otto-Hahn-Straße 14. Die Briefkästen sind mit dem Namen der Veranstaltung, der Gruppennummer sowie der Zeit der Übung gekennzeichnet. Für Rechnerstrukturen sind dies die Briefkästen mit den Nummern 20 bis 32.

Schreiben Sie unbedingt Ihren **Namen**, Ihre **Matrikelnummer** und Ihre **Gruppennummer** rechts oben auf Ihre Abgabe. Sie dürfen als Team mit bis zu zwei weiteren Personen abgeben. Geben Sie dann nur eine einzige Lösung ab und schreiben Sie alle Namen und Matrikelnummern des Teams auf die gemeinsame Abgabe.

Heften Sie die Abgabe bitte zusammen (Tacker oder notfalls Büroklammer). Bitte die Abgabe **nicht falten** und **keine Schnellhefter oder Umschläge** abgeben.

Es gibt insgesamt 12 Übungsblätter, die in 3 Blöcke (A, B, C) aufgeteilt sind. In jedem Block müssen Sie 30 Punkte von 64 möglichen Punkten erreichen, um zur Prüfung zugelassen zu werden.

HelpDesk Rechnerstrukturen:

Neben den Übungen bieten wir dieses Jahr auch einen speziellen RS Help Desk an. Der Help Desk kann euch bei der Bearbeitung der Übungsaufgaben, der Klausurvorbereitung oder sonstigen vorlesungrelevanten Problemen helfen. Weitere Information finden Sie auf der Webseite zur Vorlesung.