Karlsruhe Institute of Technology
Institut für Anthropomatik und Robotik (IAR)

Rechnerorganisation im Wintersemester 2017/18

Aufgaben zu den Tutorien in der Woche vom 05. bis 09. Februar 2018

Prof. Dr. Wolfgang Karl
Haid-und-Neu-Str. 7
Dr.-Ing. Ömer Terlemez
Adenauerring 2, Geb. 50.20
Email: ti@ira.uka.de
Web: http://ti.ira.uka.de

Lernziele:

- Wiederholung: Besprecht mit den Studenten bitte offene Fragen zum gesamten Stoff von Rechnerorganisation.
- Als Hilfe findet ihr untenstehend eine **unvollständige** Übersicht über Dinge, die ihr nochmal im Tutorium ansprechen könnt/sollt, und außerdem eine kleine Auswahl an Aufgaben, die ihr im Tutorium besprechen könnt.

- Grundlegende Begriffe/Definitionen:
  - Was besagt das Mooresches Gesetz?
  - Was beschreibt das Y-Diagramm?
  - Wie ist eine Speicherhierarchie aufgebaut?
  - Was heißt RISC und CISC? Worum handelt es sich bei MIPS?
  - Was ist eine Load-Store-Architektur?
  - ... und so weiter!
- Grundkenntnisse in C:
  - Kenntnis der C-Operatoren
  - Kenntnis der C-Kontrollstrukturen
  - Verständnis von C-Zeigern und ihren Operationen
  - Verständnis von (kleinen) C-Programmen
- Von-Neumann-Rechner:
  - Grundsätzlicher Aufbau (Komponenten, Strukturen, ...)
  - Befehlszyklus
- MIMA:
  - Was ist die MIMA?
  - Aufbau, Register, Busstruktur, Befehle, Mikrobefehle, etc.
- MIPS-Architektur:
  - Grundlegender Aufbau (Register, Befehlsformate, etc.)
  - Kenntnis der wichtigen MIPS-Befehle
  - MIPS-Assemblercode lesen, verstehen und selbst schreiben können
  - Was ist ein Pseudobefehl? Ein Beispiel?
  - Was ist eine Assemblerdirektive? Welche gibt es?
  - Wie funktionieren Systemaufrufe in MIPS?

- Pipelining:
  - Wozu dient Pipelining? Wie funktioniert es?
  - Leistungskennzahlen kennen und berechnen können
  - Verständnis der Aufgaben der einzelnen Pipelinestufen bei MIPS/DLX
  - Verständnis des Datenpfads bei MIPS/DLX
- Abhängigkeiten und Konflikte:
  - Kenntnis der verschiedenen Typen von Abhängigkeiten, insbesondere Datenabhängigkeiten
  - Erkennen von Abhängigkeiten in einem gegebenen Programmstück
  - Bestimmung, ob eine bestimmte Abhängigkeit bei gegebener Pipeline zum Konflikt führt
  - Kenntnis von SW- und HW-Methoden zum Verhindern von Pipeline-Konflikten
- Halbleiterspeicher:
  - Kenntnis der verschiedenen Halbleiterspeichertypen
  - Organisation von Halbleiterspeicher (Matrixaufbau)
  - Wie ist eine SRAM- oder DRAM-Zelle aufgebaut (in CMOS-Technik)?
  - Verständnis der DRAM-Timingparametern
- CPU-Caches:
  - Wozu dient ein Cache? Was meint man mit zeitlicher und örtlicher Lokalität?
  - Kenntnis des grundsätzlichen Aufbaus eines Caches
  - Orthogonale Entwurfskriterien von Caches (Assoziativität, Aktualisierungsstrategie, Ersetzungsstrategie usw.)
  - Nachvollziehen der Arbeitsweise eines Caches, Simulation "von Hand"
- Virtuelle Speicherverwaltung:
  - Warum benutzt man virtuelle Speicherverwaltung?
  - Was ist der Unterschied zwischen Segmentierung und Paging?
  - Verständnis der Verwaltung in ein- oder mehrstufigen Seiten-/Segmenttabellen
  - Übersetzung von virtuellen Adressen bei gegebener/-n Seiten-/Segmenttabelle(n)

Aufgabe 1

Bestimmen Sie das Ergebnis der folgenden C-Operationen die auf 32-Bit Integern ausgeführt werden:

1. `0xE44C | 0x8F9B =`
2. `0xE44C & 0x8F9B =`
3. `0xE44C ^ 0x8F9B =`
4. `~0xE44C =`
5. `6977 | 4510 =`
6. `0xE44C << 2 =`
7. `0xE44C >> 3 =`
8. `6977 << 2 =`
9. `6977 >> 3 =`

Aufgabe 2

Beantworten Sie folgende Fragen zur Funktionsweise einer arithmetisch-logischen Einheit (*Arithmetic Logic Unit*, ALU):

1. In welchem Register legen die meisten ALUs Informationen über das Ergebnis der letzten Operation ab? Nennen und erläutern Sie drei Flags, die auf einer gängigen Prozessorarchitektur aus diesem Register ausgelesen werden können. 2 P.
2. Viele ALUs bieten als Operation sowohl logisches als auch arithmetisches Rechtsschieben an. Erklären Sie den Unterschied zwischen beiden Operationen. Welcher mathematischen Operation entspricht das arithmetische Rechtsschieben. 2 P.

Aufgabe 3

Welche Funktion hat das folgende MIPS-Programm:

```
        .data
a:      .word 36, 20, 27, 15, 1, 62, 41
n:      .word 7
min:    .word 0

        .text
        .globl main

main:   li $t0, 0
        li $s0, 0
        lw $s1, n

m1:     bge $t0, $s1, m3
        mul $t1, $t0, 4
        lw $t2, a($t1)
        bge $t2, $s0, m2
        move $s0, $t2
m2:     addi $t0, $t0, 1
        b m1
m3:     move $a0, $s0
        li $v0, 1
        syscall
        li $v0, 10
        syscall
```

Aufgabe 4

Was machen die folgenden MIPS-Befehle und durch welchen Pseudobefehl könnte man sie ersetzen?

```
sra $a0, $s0, 31
xor $s0, $a0, $s0
sub $a0, $s0, $a0
```

Aufgabe 5

Gegeben sei folgendes Programmstück:

```
S1: addi $s0, $zero, 10
S2: lw   $s1, 0x10010000
S3: mult $s0, $s1
S4: mflo $s0
S5: addi $s1, $s0, -5
S6: sw   $s1, 0x10010000
```

1. Geben Sie den Wert der Speicherstelle `0x10010000` nach sequentieller Ausführung des Programmstücks an. **2 P.**

   Die Speicherstelle `0x10010000` sei vor Ausführung des Programms mit dem Wert `100` initialisiert.

2. Bestimmen Sie alle Datenabhängigkeiten im Programmstück. **3 P.**
   Hierbei können Sie Datenabhängigkeiten ignorieren, die Speicherstellen im Hauptspeicher anstelle von Registern betreffen.

   Geben Sie zu jeder Datenabhängigkeit die beiden beteiligten Befehle, das ursächliche Register und den Typ der Datenabhängigkeit an.

3. Das Programmstück wird nun auf einer DLX-Pipeline ausgeführt, die weder Forwarding noch andere Hardware-Mechanismen zur Verhinderung von Datenkonflikten realisiert. **3 P.**

   Geben Sie nach jedem Taktschritt den Zustand der Pipeline (welcher Befehl ist in welcher Pipeline-Stufe?) und den Inhalt der Register `$s0` und `$s1` und des LO-Registers an.

   Nehmen Sie an, dass Schreibvorgänge in den Registersatz im jeweiligen Taktzyklus bereits abgeschlossen werden

4. Das Programm soll auf einer DLX-Pipeline ausgeführt werden, für die keine Forwarding-Techniken implementiert sind. Fügen Sie dazu eine minimale Anzahl von Leerbefehlen (NOP-Instruktionen) ein, sodass keine der Datenabhängigkeiten zu Konflikten führt. **2 P.**

   Verändern Sie die Reihenfolge der Befehle nicht.

5. Betrachten Sie nun eine DLX-Pipeline auf der Result Forwarding und Load Forwarding implementiert sind. Fügen Sie eine minimale Anzahl von Leerbefehlen (NOP-Instruktionen) ein, sodass keine der Datenabhängigkeiten zu Konflikten führt. **2 P.**

   Verändern Sie die Reihenfolge der Befehle nicht.

6. Geben Sie die Anzahl der zur Ausführung notwendigen Taktzyklen für die sequentielle Ausführung und die (korrekte) Ausführung auf einer DLX-Pipeline ohne und mit Forwarding an. **3 P.**

   Nehmen Sie für die sequentielle Ausführung an, dass die Ausführung jedes Befehls fünf Taktzyklen benötigt.

   Für die Ausführung auf einer DLX-Pipeline analysieren Sie die modifizierten Programmstücke, die Sie in den vorhergehenden Aufgabenteilen erstellt haben.