TI II: Rechnerarchitektur SS 2007 Klausur 20.07.07

Prof. Dr.-Ing. Jochen Schiller, AG Technische Informatik, Freie Universität Berlin

Bearbeitungszeit 90 Minuten – Ohne Hilfsmittel!

Bitte schreiben Sie auf alle Blätter Ihren Namen und Ihre Matrikelnummer!
Maximale Punktzahl: 90, zum Bestehen sind 45 Punkte nötig (50% der max. Punkte).
Achten Sie darauf, dass Ihr Lösungsweg stets erkennbar ist – mit Ausnahme von Aufgabe 1.

Vorname: _____ **Name:** _____

Matrikelnummer: _____

---

1. Aufgabe: Richtig oder falsch? (12 Punkte)

Welche der folgenden Aussagen treffen generell zu?
(Punktabzug für falsche Antworten; minimale Punktzahl = 0 Punkte)

a)	Getrennte Caches für Daten und Befehle sind auch bei der von Neumann-Architektur zwischen CPU und Speicher denkbar und sinnvoll.	◯ richtig	◯ falsch
b)	Für die Übergabe von Parametern an ein Unterprogramm wird der Stack verwendet. Dies wird durch die verwendete Mikroarchitektur so vorgegeben.	◯ richtig	◯ falsch
c)	Das Blockieren einer Pipeline ist vom Effekt her (bezogen auf den Durchsatz) mit dem Einfügen von NOPs durch den Compiler vergleichbar.	◯ richtig	◯ falsch
d)	Superskalare Pipelines sind ein Beispiel für Parallelität auf Instruktionsebene.	◯ richtig	◯ falsch
e)	Eine k-stufige Pipeline benötigt zum Ausführen von n Befehlen im Idealfall nur n+k+1 Schritte.	◯ richtig	◯ falsch
f)	Register werden wie der Speicher über Adressen ausgewählt.	◯ richtig	◯ falsch
g)	Erst wenn man eine unendlich lange Folge von Instruktionen hat, erreicht eine k-Stufige Pipeline ihren maximalen Speedup.	◯ richtig	◯ falsch
h)	Die Stufen einer Pipeline sind über einen Bus verbunden.	◯ richtig	◯ falsch

2. Aufgabe: DLX-Pipeline (11 Punkte)

Skizzieren Sie den grundlegenden Aufbau einer DLX-Pipeline, benennen Sie die Stufen und erläutern Sie kurz deren Funktion. Zeichnen Sie zusätzlich die für das *Forwarding* notwendigen Leitungen ein und beschriften Sie diese entsprechend.

3. Aufgabe: Pipelines (11 Punkte)

a) [2 Punkte] Warum kann in VLIW-Prozessoren Platz für die Erkennungslogik zur parallelen Ausführung von Instruktionen im Gegensatz zu superskalaren Prozessoren eingespart werden?
b) [3 Punkte] Welche Voraussetzungen sollten erfüllt sein, damit Pipelining sinnvoll eingesetzt werden kann?
c) [3 Punkte] Was ist der Unterschied zwischen der Instruction Set Architecture (ISA) und einer Mikroarchitektur?
d) [3 Punkte] Welche Vor- und Nachteile haben kurze bzw. lange Pipelines?

Alternative Aufgaben

Lösen Sie EINE der folgenden beiden Aufgaben. Geben Sie nur EINE Lösung ab! Sollten zwei abgegeben werden, so wird nicht die bessere gewertet, sondern die zuerst korrigierte.

4. Aufgabe: Einfaches Assembler-Programm (15 Punkte)

Schreiben Sie ein einfaches MIPS Assemblerprogramm, das die Zeichen einer nullterminierten Zeichenkette im Hauptspeicher an eine neue Stelle kopiert. Folgende Punkte sind zu berücksichtigen:

- Verhindern Sie einen *Buffer Overflow*. Dies bedeutet, dass der Kopiervorgang beendet wird, sobald im Zielspeicherbereich kein Platz mehr vorhanden ist.
- Die Startadresse des *Strings* steht im Register `$t4`, die Zieladresse in `$t5` und die Länge des an dieser Stelle reservierten Bereichs in `$t6`.
- Achten Sie darauf, dass der kopierte (Teil-) String terminiert ist.
- Das Programm ist zu kommentieren!
- Gehen Sie davon aus, dass in einem Wort ein Zeichen steht.

4. Aufgabe: Mysteriöses Programm (15 Punkte)

Was berechnet das folgende (Unter-) Programm?

- Kommentieren Sie in sinnvoller Weise.
- Hinweis: Die Eingabe steht in `$t1` und das Ergebnis wird durch erreichen des Labels `endTrue` oder `endFalse` angezeigt. Der Rückgabewert wird nach `$t2` geschrieben.
- Sollten Sie nicht sofort die Funktion des Programms erkennen, hier ein kleiner Tip: Schauen Sie sich an, wie die Zahlen, die ein `true` zurückliefern, aufgebaut sind.

```
start:     addi $t4, $zero, 32       #
           addi $t5, $zero, 1        #
           and  $t7, $t7, $zero      #

loop:      beq  $t4, $zero, endFalse #
           and  $t6, $t1, $t5        #
           srl  $t1, $t1, 1          #
           bne  $t6, $zero, check    #
           addi $t7, $t7, 1          #
           subi $t4, $t4, 1          #
           b    loop                 #

check:     and  $t6, $t5, $t7        #
           beq  $t6, $t5, endFalse   #
           bne  $t1, $zero, endFalse #

endTrue:   addi $t2, $zero, 1        # return true
           jr $ra

endFalse:  addi $t2, $zero, 0        # return false
           jr $ra
```

5. Aufgabe: Frequently Asked Questions (13 Punkte)

Beantworten Sie die folgenden Fragen so verständlich und kurz wie möglich.

a) [2 Punkte] In meinem Assembler Programm ist eine Multiplikation der Form `multi $6, $6, 8` vorhanden. In dem letztendlich erstellten Binary finde ich allerdings die Anweisung `sll $6, $6, 3`. Ist dies ein Fehler im Assembler? Falls nicht, warum fand eine Ersetzung statt.
b) [2 Punkte] Wie werden Arrays auf Prozessorebene dargestellt? Ich kann hierzu in meinem MIPS Handbuch keine Angaben finden.
c) [3 Punkte] In einem Artikel las ich, dass in einem CISC-Prozessor heutzutage quasi ein RISC steckt. Ist dies wahr und was ist der Grund dafür? Warum wird dann nicht gleich eine RISC ISA verwendet?
d) [2 Punkte] Angenommen ich möchte ein von mir geschriebenes Programm im Speicher verschieben können. Was muss ich dabei beachten?
e) [2 Punkte] Mein Übungsgruppenpartner meint, die binäre Zahl `10011101` entspricht der Dezimalzahl 157. Ich habe aber -99 ausgerechnet. Wer hat denn nun Recht?
f) [2 Punkte] Eines meiner Programme verwendet Gleitkommazahlen. Beim manuellen Überprüfen ist mir aufgefallen, dass die berechneten Ergebnisse im Nachkommastellenbereich voneinander abweichen. Ist meine CPU defekt?

6. Aufgabe: Caches (16 Punkte)

Beantworten Sie die folgenden Fragen so verständlich und kurz wie möglich.

a) [2 Punkte] Durch welche Eigenschaften (eines Programms bzw. Anweisungsfolge) kann ein Cache den Speicherzugriff beschleunigen? Erklären Sie diese.
b) [2 Punkt] Angenommen ein Cache habe 16 Sätze und 128 Blockrahmen. Wie groß ist die Assoziativität?
c) [6 Punkte] Bestimmen Sie die Anzahl der Sätze in einem Direct Mapped (DM) und 2-Way-Set-Associative (A2) Cache, sowie die Anzahl der Bits, die zur Satzadressierung und zur Wortadressierung (innerhalb des Blocks) für jede der beiden Caches erforderlich sind.

DM	A2
• Kapazität: 256 Datenwörter • Blocklänge: 4 Datenwörter	• Kapazität: 64 Datenwörter • Blocklänge: 2 Datenwörter

d) [3 Punkte] Nehmen wir an, wir haben ein Multiprozessorsystem. Welche Auswirkungen haben die Cachestrategien *write invalidate* und *write update* auf den Datenzugriff? Gegen Sie dazu an, welche Informationen über den Bus geschickt werden.
e) [3 Punkte] Es existiert die sogenannte *Climb* Verdrängungsstrategie bei der ein neues Datum stets unten in eine (vertikale) Liste eingefügt wird. Verdrängt wird immer der unterste Eintrag, wenn der Cache voll ist. Bei jedem Zugriff steigt der entsprechende Eintrag um eine Ebene nach oben.
Welche Vor- und Nachteile bemerken Sie bei diesem Verfahren?

7. Aufgabe: Sprungvorhersage (12 Punkte)

Geben Sie für folgendes Programm die Anzahl der Sprungvorhersagen und deren Erfolgsrate bei einem Zwei-Bit Prädiktor mit Hysterese-Schema an. Dieser sei mit „*weakly taken*“ initialisiert.
Zur Unterstützung kann die angegebene Tabelle optional ausgefüllt werden.

```
void function() {
          int i=0;
     loop1:
          int j=1;
     loop2:
          irgendwas();
          j++;
          if(j<10)

               goto loop2;
          i++;
          if(i<3)
               goto loop1;
}
```

	Vorhersagen	Zwei-Bit, erfolgreich
loop1		
loop2		
gesamt		