Prüfungsklausur Computersysteme (25211)
Teil 1609
WS 2015

Prof. Dr. W. Schiffmann

19.03.2016

Inhaltsverzeichnis

Aufgabe II-1: Fragen zur Rechnerarchitektur (5 P) 2

Aufgabe II-2: Gleitkommadarstellung (8 P) 3

Aufgabe II-3: Code-Analyse (20 P) 6

Aufgabe II-4: Speicherverwaltung und Cache (14 P) 10

Aufgabe II-5: Parallelverarbeitung (3 P) 15

Aufgabe II-1: Fragen zur Rechnerarchitektur (5 P)

Anhand der nachfolgenden Fragen sollen die Unterschiede zwischen Architektur- und Mikroarchitekturtechniken erarbeitet werden.

a) Geben Sie an, ob die folgenden Aussagen wahr (W) oder falsch (F) sind:

W F

○ ○ 1) Befehlspipelining ist eine Architekturtechnik.

○ ○ 2) Superskalare Prozessoren besitzen eine spezielle Mikroarchitektur.

○ ○ 3) Architekturtechniken müssen durch entsprechende Systemsoftware (z.B. Compiler) unterstützt werden.

○ ○ 4) Mikroarchitekturen können ausschließlich durch die verfügbaren Maschinenbefehle, Adressierungsarten und für den Programmierer sichtbaren Register beschrieben werden.

○ ○ 5) Renaming ist eine Mikroarchitekturtechnik.

○ ○ 6) Dynamisches Befehlsscheduling in Superskalarprozessoren ist eine Architekturtechnik.

b) Ordnen Sie den folgenden Prozessortypen zu, ob sie hauptsächlich auf einer Architekturtechnik (A) oder einer Mikroarchitekturtechnik (M) basieren:

Prozessortyp	A	M
Mikroprogrammierter CISC-Prozessor	○	○
Skalarer RISC-Prozessor	○	○
Superskalarer RISC-Prozessor	○	○
VLIW-Prozessor	○	○

Aufgabe II-2: Gleitkommadarstellung (8 P)

Gegeben sei die Dezimalzahl $Z_{10} = -123,4375$.

a) Stellen Sie die Zahl Z_{10} als gebrochene, normalisierte Zahl Z_{32} im 32-bit-Format des IEEE-754-Standards dar und tragen Sie dazu die entsprechenden Werte für Vorzeichen, verschobenen Exponenten und Mantisse in das folgende Schema ein:

$Z_{32} = (-1)^{\cdots\cdots} \cdot 2^{(\cdots\cdots)_{10}} \cdot (\ldots\ldots \bullet \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots)_2$

b) Tragen Sie die Zahl Z_{32} in binärer Darstellung in den folgenden Bitrahmen ein und kennzeichnen sowie bezeichnen Sie die unterscheidbaren Bitfelder.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Tragen Sie hier die Bezeichnungen der Bitfelder ein

c) Geben Sie die Zahl Z_{32} als Hexadezimalzahl Z_{16} an.

$Z_{16} = \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

d) Gegeben seien nun die Zahlen $Z1_{10} = 12,75$ und $Z2_{10} = 18,25$. Wandeln Sie diese Zahlen in das IEEE-754-Format um und bilden Sie die Summe dieser beiden Zahlen. Wie erfolgt die Angleichung der Charakteristik? Die Nebenrechnung in binärer Form (Rechnung im IEEE-754-System) ist erforderlich. Führen Sie die Addition aus und tragen Sie das Endergebnis ein.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Hinweis:
Die Indizes $\cdots_2$, $\cdots_{10}$, $\cdots_{16}$ und $\cdots_{32}$ kennzeichnen jeweils Zahlen im Binär-, Dezimal- sowie Hexadezimal-System bzw. im 32-Bit-IEEE-Format.

Aufgabe II-3: Code-Analyse (20 P)

Gegeben sei der unten stehende DLX-Assembler-Code. Der Wert im Speicher an Stelle *src* sei eine vorzeichenbehaftete Integer-Zahl im Zweierkomplement.

```
Zeile  Marke   Anweisung
1              LW     R2,   src
2              LW     R3,   mask1
3              LW     R4,   mask2
4              LW     R5,   mask3

5              AND    R10,  R2,   R3
6              BEQZ   R10,  m1

7              XOR    R2,   R2,   R4
8              ADDI   R2,   R2,   #0x1

9       m1:    SRLI   R3,   R3,   #0x1
10             AND    R11,  R0,   R0
11      m2:    AND    R6,   R2,   R3
12             BNEZ   R6,   m3
13             ADDI   R11,  R11,  #0x1
14             SLLI   R2,   R2,   #0x1
15             J      m2

16      m3:    ADDI   R6,   R0,   #0x1e
17             SUB    R11,  R6,   R11
18             ADDI   R11,  R11,  #0x7f
19             SLLI   R11,  R11,  #0x17
20             OR     R10,  R10,  R11

21             SRLI   R2,   R2,   #0x7
22             AND    R2,   R2,   R5
23             OR     R10,  R10,  R2

24             SW     dst,  R10
25      m4:    J      m4

26      src:   .word  0xFFFFFF85
27      dst:   .word  0xDEADBEEF
28      mask1: .word  0x80000000
29      mask2: .word  0xFFFFFFFF
30      mask3: .word  0x7FFFFF
```

a) Erklären Sie in einem Satz folgende Assembler-Anweisungen (vgl. Beispiel am Ende dieses Aufgabenteils!):

Zeile 6: `BEQZ R10, m1:`

.....

.....

Zeile 9: `SRLI R3, R3, #0x1:`

.....

.....

Zeile 22: `AND R2, R2, R5:`

.....

.....

Beispiel:

Zeile 1: `LW R2, src`: Load Word: Diese Anweisung lädt das 32-Bit-Wort an der Speicherstelle src in das Register R2.

b) Welche Boole'sche Operation wird durch folgende Anweisung nachgebildet, die nicht explizit Teil des DLX-Sprachumfangs ist? (Hinweis: Beachten Sie den Inhalt des Registers R4!)

Zeile 7: `XOR R2, R2, R4`

.....

.....

c) Erklären Sie, was folgende Code-Blöcke funktional realisieren (vgl. Beispiel am Ende dieses Aufgabenteils!):

Zeilen 7 – 8:

.....

.....

Zeilen 9 – 15: (Hinweis: Beachten Sie, dass die Zeilen 12 – 16 eine Schleife darstellen, die i. A. mehrfach ausgeführt wird!)

.....

.....

.....

.....

Zeilen 16 – 20:

.....

.....

.....

.....

Beispiel:

Zeilen 5 – 6: Durch die Maske 0x80000000 in R3 wird das Vorzeichenbit aus R2 extrahiert und im MSB des Registers R10 gespeichert. In Abhängigkeit davon, ob dieses Bit 0 oder 1 ist, wird der Sprung in Zeile 6 nach Marke m1 genommen (0) oder die Ausführung bei Zeile 7 fortgesetzt (1).

d) Beschreiben Sie in **einem** Satz, was das obige Programm berechnet, also was nach der Ausführung der Zeile 24 in der Speicherstelle *dst* steht. (Hinweis: Konvertierung von Zahlenformaten)

.....

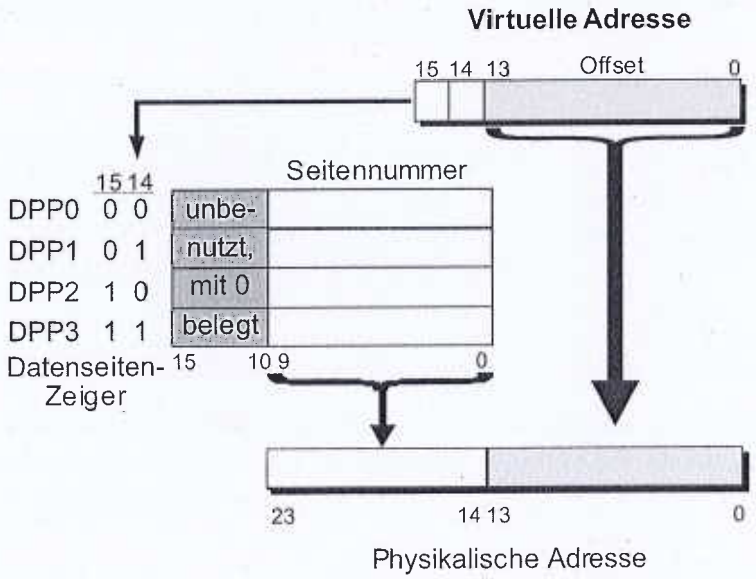
.....

Aufgabe II-4: Speicherverwaltung (14 P)

Ein einfacher Mikroprozessor mit einem 16-Bit-Datenbus und einem 24-Bit-Adressbus unterstütze eine rudimentäre Form der „virtuellen Speicherverwaltung" mit Seiteneinteilung, wie sie in der unten stehenden Zeichnung skizziert ist. Dazu stehen dem Prozessor vier 16-Bit-Register DPPi, i=0,..,3, (Datenseiten-Zeiger – *Data Page Pointer*) zur Verfügung, in denen jedoch nur die unteren 10 Bits zur Auswahl von bis zu vier aktuell benötigten Seiten ausgewertet werden. Die in einem Ladebefehl:

LD R, DPPi:<Offset>
(„Lade Register R mit dem Inhalt der adressierten Speicherzelle")

angegebene „virtuelle" Adresse ist 16 Bits lang. Ihre höchstwertigen beiden Bits A15, A14 selektieren das verwendete DPPi-Register. Die Speicheradresse (physikalische Speicheradresse) ergibt sich aus der „Verkettung" (Konkatenation) der 10-Bit-Seitennummer im gewählten DPPi-Register und der unteren 14 Bits der virtuellen Adresse als Offset in der gewählten Seite.



a) Wie groß sind die selektierbaren Seiten?

Größe (in kByte): .....

Begründung: .....

.....

b) Wie viele verschiedene Seiten können im physikalischen Speicher maximal angesprochen werden?

Anzahl: .....

Begründung: .....

.....

c) Wie groß ist der gesamte ansprechbare physikalische Speicher maximal?

Größe (in MByte): .....

Begründung: .....

.....

d) Welche physikalische Adresse (in hexadezimaler Form) wird ausgegeben, wenn die Register DPPi folgendermaßen belegt sind:
DPP0: $03FD DPP1: $027A DPP2: $00B4 DPP3: $01CC
und die virtuelle Adresse $A7D9 angesprochen wird?

phys. Adresse: .....

Herleitung: .....

.....

.....

.....

.....

e) Welcher Wert muss in DPP1 eingetragen und welche virtuelle Adresse muss im Befehl angegeben werden, wenn die Speicherzelle mit der physikalischen Adresse $D37FDA gelesen werden soll?

DPP1: $.....

virtuelle Adresse: $.....

Herleitung: .....

.....

.....

.....

.....

f) Welchen Vorteil bietet die beschriebene virtuelle Adressierung mit Seiteneinteilung, obwohl der virtuelle Adressraum kleiner ist als der physikalische?

.....

.....

.....