Prüfungsklausur Computersysteme – Teil 1609
SS 2014

Prof. Dr. W. Schiffmann

13.09.2014

Inhaltsverzeichnis

Bewertungsschema

Aufgabe	a	b	c	d	e	total
1	2	1				3
2	3	2	1	6		12
3	3	2	9	4		18
4	3	2	3	3	3	14
5	2	1				3

1 Fragen zur Rechnerarchitektur

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

a) Kreuzen Sie bitte nur die zutreffenden Aussagen an.

- ◯ 1) Befehlspipelining ist eine Architekturtechnik.
- ◯ 2) Superskalare Prozessoren stellen spezielle Mikroarchitekturen dar.
- ◯ 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.
- ◯ 7) Statisches Befehlsscheduling wird durch den Compiler vorgenommen.

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	◯	◯
EPIC-Prozessor	◯	◯
Skalarer RISC-Prozessor	◯	◯
Superskalarer RISC-Prozessor	◯	◯
VLIW-Prozessor	◯	◯

Lösungsvorschläge

Die Antworten 2, 3, 5 und 7 treffen zu. Bei b) ist nur EPIC und VLIW ein A zuzuordnen. Die anderen Prozessoren sind mit M zu kennzeichnen.

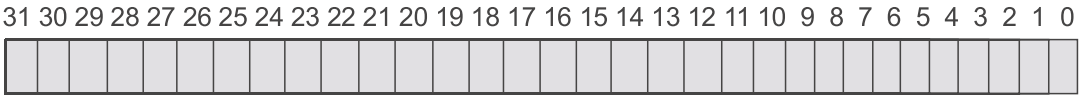
2 Gleitkommadarstellung

Gegeben sei die Dezimalzahl $Z_{10} = -268,40625$.

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:

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

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

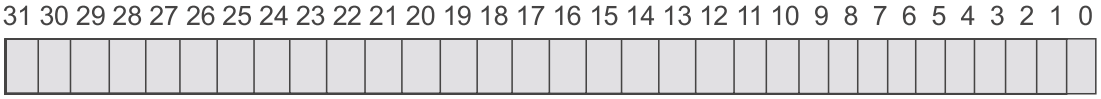


. . . . . .

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

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

d) Gegeben seien nun die Zahlen $Z1_{10} = 28,5$ und $Z2_{10} = 11,5$. 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:

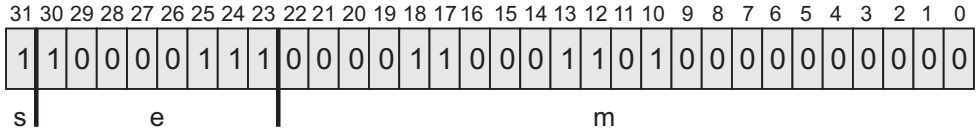


Hinweis:
Die Indizes $\cdots_2$, $\cdots_{10}$, $\cdots_{16}$ und $\cdots_{32}$ kennzeichnen jeweils Zahlen im Binär-, Dezimal- sowie Hexadezimal-System bzw. im 32-Bit-IEEE-Format.

Lösungsvorschläge

Zu a): $Z_{32} = (-1)^1 \cdot 2^{(135)_{10}} \cdot (1 \bullet 0000110001101)_2$

Zu b):



Zu c):
$Z_{16} = C3863400_{16}$

Zu d):
$Z1_{32} = 0\ 10000011\ 110010000...000$
$Z2_{32} = 0\ 10000010\ 011100000...000$
Der verschobenen Exponent von Z2 wird um 1 erhöht (10000011) und entsprechend die Mantisse um eins nach rechts geschoben (0.10111). Die Matisse von Z1 (1.11001) wird dazu addiert. Dies ergibt 10.1. Normalisieren, d.h. eins nach rechts schieben, und verschobenen Exponenten erhöhen (10000100). Gesamtergebnis: 0 10000100 0100000...0.

3 Code-Analyse

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 ein 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 25 in der Speicherstelle *dst* steht. (Hinweis: Konvertierung von Zahlenformaten)

. . . . . .

. . . . . .

Lösungsvorschläge

Zu a) Zeile 6: Branch Equal Zero; Ein Sprung zu Marke m1, wenn der Inhalt von R10 gleich 0 ist, ansonsten Fortführung des Programms mit nachfolgendem Befehl.

Zeile 9: Shift Right Logical Immediate; Der Inhalt von R3 wird logisch um eine Bitposition nach rechts verschoben. Links wird eine 0 eingefügt.

Zeile 22: AND R2, R2, R5; der Inhalt von R2 wird mit dem Inhalt von R5 (Maske 7FFFFF) bitweise konjugiert (UND-Verknüpfung). Danach bleiben die 23 unteren Bits werden erhalten, während die restlichen oberen Bits auf 0 gesetzt werden.

Zu b) Die Anweisung in Zeile 7 (exklusives ODER) bildet eine bitweise Negation durch die Verwendung der Maske 0xFFFFFFFF als zweiten Parameter nach.

Zu c) Zeile 7 – 8: Berechnet das Zweierkomplement des Inhalts von Register R2 und speichert dieses wieder in R2. In Falle dieses Programms wird eine negative Zahl in ihre entsprechende positive Darstellung überführt.

Zeile 9 – 15: Die Maske in Register R3 wird so verschoben, dass diese nun das Bit 30 aus einem 32-Bit-Wort extrahiert. Im weiteren Verlauf wird der Inhalt von R2 solange nach links verschoben, bis ein 1-Bit in Bit-Position 30 erkannt wird. Diese Anzahl der Verschiebungen wird in Register R11 gespeichert.

Zeile 16 – 20: Nun wird der eben ermittelte Wert von 30 subtrahiert (Zeilen 16 und 17). Damit wird die Anzahl der Verschiebungen berechnet, um eine IEEE-754-Zahl aus dem gegebenen Integer-Wert zu erstellen. Zu dem so berechnete Wert wird noch der Wert 127 addiert (Zeile 18), um die Charakteristik zu erzeugen. Dieser wird an die korrekten Stellen 23 – 30 geschoben (Zeile 19) und dann mit dem Inhalt von R10 kombiniert (Zeile 20), das schon das Vorzeichen enthält.

Zu d) Das Programm wandelt eine vorzeichenbehaftete 32-Bit-Integer-Zahl in die 32-Bit-IEEE-754-Form um und legt diesen an der Speicherstelle dst ab.

4 Cache-Organisation

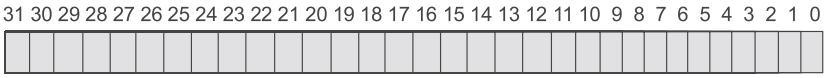
Ein Mikroprozessor besitze einen 32-bit-Adressbus und einen 16-bit-Datenbus. Er verfüge über einen *Direct Mapped Cache*, dessen Datenspeicher eine Kapazität von 64 kbyte hat und Einträge (*Cache Lines*) der Länge 16 Byte enthält. Unter der Organisation eines Speichers versteht man die Angabe der Anzahl m der Speicherzellen und ihrer Länge l (in Bits oder Bytes), meist in der Form $m \times l$.

a) Geben Sie die Anzahl der Cache-Einträge sowie die Organisation, d.h. die Anzahl der Speicherzellen und die Länge (in Bits oder Bytes) jeder Speicherzelle, des Datenspeichers an. (Rechnung erforderlich!)

. . . . . .

. . . . . .

b) Tragen Sie in das folgende Bild die unterscheidbaren Bitfelder einer Adresse für die Auswahl eines Bytes, eines Cache-Eintrags und die im Cache gespeicherte Teiladresse ein und benennen Sie diese Bitfelder:



Bitfelder: . . . . . .

c) Geben Sie für die gefundene Adressaufteilung die Kapazität und die Organisation des Adressspeichers (*Tag-RAM*) im Cache an. (Rechnung erforderlich!)

. . . . . .

. . . . . .

d) Das Datenregister DR enthalte den Wert DR = $ABCD, das Adressregister AR den Wert AR = $FACD3A6A. Mit diesen Registern werde der Schreibbefehl ST (AR), DR „*Schreibe DR in die Speicherzelle, deren Adresse in AR steht.*" ausgeführt. Geben Sie an, welcher Eintrag im Cache verändert wird, wenn dieser

i. nach dem Rückschreibverfahren verwaltet wird und ein *Write Hit* vorliegt,

Index: $..... = ...$_{10}$
(„$", „$_{10}$" kennzeichnen einen Hexadezimal– bzw- Dezimalwert!)

Adressen der veränderten Bytes im Eintrag: $..... und $.....

ii. nach dem Durchschreibverfahren mit *Write Around* verwaltet wird, bei dem nach einem *Write Miss* das Datum nur im Hauptspeicher abgelegt wird. Im Cache liege unter dem Index \$3A6 der Tag \$FA00.

. . . . . .

. . . . . .

e) Der Cache sei als 4-fach satzassoziativer Cache (*n-Way Set Associative Cache*) organisiert, besitze aber die gleiche Gesamtkapazität von 64 kbyte für die Datenspeicher, aber mit einer Länge der Einträge von 32 byte. Jeder Eintrag im Tag-RAM der Teil-Caches enthalte zwei Verwaltungsbits V (*Valid*) und D (*Dirty*).
Bestimmen Sie wiederum die Anzahl der Einträge pro Teil-Cache, die Organisation der Tag-RAMs und ihre Gesamtkapazität und die unterscheidbaren Bits einer Speicheradresse. (Rechnung erforderlich !)

Kapazität der Teil-Caches: .....

Einträge pro Teil-Cache: .....

Länge der Tag-RAM-Einträge: .....

Organisation der TAG-RAMs: .....

Gesamtkapazität: .....

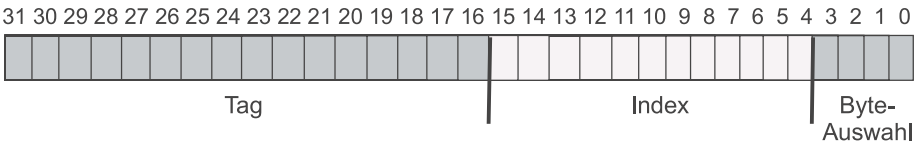
Bitfelder: .....

Lösungsvorschlag

a)

Anzahl der Einträge: 64 kbyte / 16 byte = 4 k = 2^{12} = 4.096 Einträge
Organisation: 4096 × 16 byte

b)



c)

Organisation: 4096 Einträge zu je 16 Bits ⇒ 4096 ×2 byte = $4k \times 2$ byte
Kapazität: $4k \cdot 2$ byte = 8 kbyte

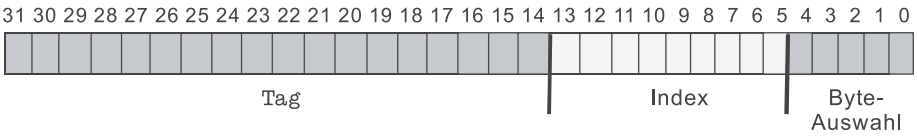
d)

i. Index: $\$3A6 = 934_{10}$
Adressen der veränderte Bytes im Eintrag: $A und $B

ii. Es wird kein Cache-Eintrag verändert, da ein *Write Miss* vorliegt und der Zugriff nur auf den Arbeitsspeicher geht.

e)

Kapazität der Teil-Caches: 64 kbyte / 4 = 16 kbyte
Anzahl der Einträge pro Teil-Cache: 16 kbyte / 32 byte = 512 = 2^9
Länge der Tag-RAM-Einträge: 18 + 2 bit = 20 bit
Organisation der Tag-RAMs in bit: 512 × 20 bit = 0,5 k × 20 bit
Gesamtkapazität der Tag-RAMs in kbyte: 4 · 10 kbit = 40 kbit = 5 kbyte



5 Amdahl's Gesetz

Ein Algorithmus A habe eine sequentielle Laufzeit von a Zeiteinheiten (ZE). Eine parallele Version dieses Algorithmus habe folgende Eigenschaften: Die Laufzeit a kann beliebig fein parallelisiert werden, indem jeder Prozessor einen Teil der Daten bearbeitet. Wegen der zusätzlichen Synchronisation der Prozessoren verlängert sich der parallele Anteil um 5%. Darüber hinaus kommt pro Prozessor ein sequentiellen Anteil von $0,015$ ZE hinzu. Für den Broadcast-Nachrichtenversand ergibt sich eine konstante Verlängerung der Laufzeit um 15 ZE.

a) Stellen Sie die Gleichung für die Berechnung der Laufzeit auf einem Parallelsystem in Abhängigkeit der verfügbaren Prozessoren auf.

b) Geben Sie die Speedup-Funktion nach Amdahl in Abhängigkeit der verfügbaren Prozessoren an. Vereinfachen Sie diese Gleichung.

Anmerkung: Leiten Sie Ihre Ergebnisse her.

Lösungsvorschläge

Zu a): $T(n) = 15 + \frac{1{,}05 \cdot a}{n} + 0,015 \cdot n$
Zu b): $S(n) = \frac{T(1)}{T(n)} = \frac{a}{15 + \frac{1{,}05 \cdot a}{n} + 0{,}015 \cdot n}$