Hochschule Konstanz Fakultät Informatik

WS 11/12

Klausur: Rechnerarchitektur Datum: 08.02.12

Prüfer: Prof. Dr. J. Neuschwander Zeitdauer: 90 Min.

Name:
Matr.-Nr.:

Hinweise:

Beschriften Sie jedes weitere abgegebene Lösungsblatt deutlich lesbar mit Ihrem Namen und Ihrer Matrikelnummer.

Es sind für diese Klausur als Hilfsmittel nur Taschenrechner, ein DIN-A4 Blatt als Formelsammlung sowie der Befehlssatzes des Prozessors M68000 zugelassen.

Bei einer angegebenen Lösung muss der Lösungsweg nachvollziehbar sein.

Maximal erreichbare Punktzahl der Klausur:	37 Punkte
Mindestpunktzahl zum Bestehen der Klausur:	17 Punkte
Aufgabe 1:	8 Punkte
Aufgabe 2:	4 Punkte
Aufgabe 3:	6 Punkte
Aufgabe 4:	7 Punkte
Aufgabe 5:	6 Punkte
Aufgabe 6:	6 Punkte

1. Aufgabe: (8 P)

Gegeben sei folgende Programmsequenz für einen RISC-Prozessor:

```
S1:   ADD    R1, R2, 2        ; R1:= R2 + 2
S2:   SUB    R4, R3, R1       ; R4:= R3 – R1
S3:   LOAD   R5, (A1)         ; R5:= <(A1)>
S4:   MUL    R3, R5, 5        ; R3:= R5 * 5
S5:   ADD    R3, R3, 3        ; R3:= R3 + 3
S6:   ADD    R2, R1, R5       ; R2:= R1 + R5
```

Die einzelnen Befehle werden in einer DLX-Pipeline verarbeitet: Die Latenzzeit jeder Pipelinestufe beträgt 6 ns.

Beim Start des Programmstücks sind die Register folgendermaßen belegt:

R1	R2	R3	R4	R5	<(A1)>
3	5	7	9	2	1

a) Geben Sie die Registerbelegung nach Ablauf des Programmstücks auf der DLX-Pipeline an. Tragen Sie die jeweiligen Werte in die folgende Tabelle ein.

R1	R2	R3	R4	R5

b) Markieren Sie mit Pfeilen die Datenabhängigkeiten, die im gegebenen Programm zu Pipeline-Konflikten führen.

S1:

IF	ID/OF	EX	MA	WB

S2:

IF	ID/OF	EX	MA	WB

S3:

IF	ID/OF	EX	MA	WB

S4:

IF	ID/OF	EX	MA	WB

S5:

IF	ID/OF	EX	MA	WB

S6:

IF	ID/OF	EX	MA	WB

c) Die auftretenden Pipeline-Konflikte sollen vom Compiler behandelt werden. Ergänzen Sie obiges Programmstück mit möglichst wenigen NOP-Befehlen so, dass das Ergebnis dem der normalen, sequentiellen Bearbeitung entspricht.

d) Wie viele Takte werden benötigt, um dieses Programm aus c) abzuarbeiten?

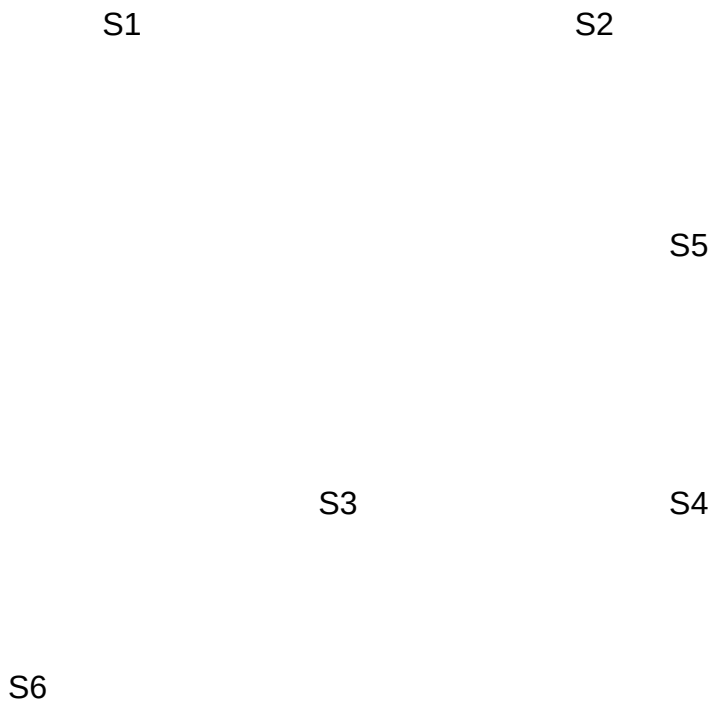
e) Die DLX-Pipeline wird mit der maximalen Taktfrequenz von 125 MHz betrieben. Wie groß ist dann die Latenzzeit eines Pipelineregisters?

f) Wie hoch ist der Durchsatz obiger DLX-Pipeline?

g) Geben Sie alle Datenabhängigkeiten im Programmstück an.

```
S1:   ADD    R1, R2, 2        ; R1:= R2 + 2
S2:   SUB    R4, R3, R1       ; R4:= R3 – R1
S3:   LOAD   R5, (A1)         ; R5:= <(A1)>
S4:   MUL    R3, R5, 5        ; R3:= R5 * 5
S5:   ADD    R3, R3, 3        ; R3:= R3 + 3
S6:   ADD    R2, R1, R5       ; R2:= R1 + R5
```

Stellen Sie die jeweiligen Abhängigkeiten durch einen Pfeil dar und benennen sie diese Abhängigkeit.



2. Aufgabe: (4 P)

a) In einem Rechner läuft die CPU mit einer Taktfrequenz von 1GHz. Die durchschnittliche Anzahl von Taktzyklen pro Befehl beträgt 4. Die CPU soll beim Re-Design so verbessert werden, dass die CPI-Rate auf 2 verringert und die Programmausführungszeit auf ein Viertel reduziert wird. Mit welcher Taktfrequenz muss die neue CPU dann betrieben werden?

b) Gegeben sei ein Prozessor mit getrennten Befehls- und Datencachespeicher. Die Fehlzugriffsrate des Befehlscachespeichers beträgt 20%, die des Datencachespeichers beträgt 50%. Der Zugriff auf den Befehlscachespeicher erfolgt in 10 ns, der Zugriff auf den Datencachespeicher erfolgt in 20 ns. Ein Hauptspeicherzugriff dauert 80 ns.
Auf dem Prozessor läuft ein Programm *test*, dessen Speicherzugriff zu 80% aus Zugriffen auf Befehle und zu 20% aus Zugriffen auf Daten bestehen. Gehen Sie davon aus, dass die Cachespeicher und der Hauptspeicher gleichzeitig adressiert werden und dass das Datum beim Laden aus dem Hauptspeicher direkt dem Prozessor zugeführt wird. Berechnen Sie die mittlere Speicher-Zugriffszeit für das Programm *test*.

3. Aufgabe: (6 P)

(Zur Ermittlung der Punktzahl in dieser Teilaufgabe werden von den richtig angekreuzten Aussagen die falsch angekreuzten Aussagen abgezogen; Nicht angekreuzte Aussagen zählen nicht und gehen somit nicht in die Bewertung ein.)

Fragen	richtig	falsch
Der *TAS*-Befehl (M68000) kann zur Implementierung binärer Semaphore benutzt werden.		
Die symbolischen Stackoperationen *push* und *pull* sind zur Berechnung rekursiver Funktionen erforderlich.		
Bei der „write-back“-Strategie eines Caches können Inkonsistenzen zwischen Cache und Arbeitsspeicher vermieden werden.		
Interleaving zwischen verschiedenen Speicherbänken dient zur Kompensation der Zugriffszeit von DRAMs.		
Der Austausch von Zeilen in Sätzen bei mehrwege-satzassoziativen Caches kann nach dem LRU-Prinzip erfolgen.		
Beim „Bus Snooping“ wertet der Cache-Controller die Adressen beim Zugriff anderer Busmaster auf den Hauptspeicher aus.		
Zur Auflösung globaler Adressbezüge durch den Linker benötigt der Assembler den *Define Constant (DC)*-Befehl.		
Unter Branch Recovery versteht man die Rücknahme getätigter Berechnungen bei falschen Vorhersagen im Fall spekulativer Befehlsverarbeitung.		
Die Datenübertragung zwischen L1- und L2-Caches erfolgt durch Blockbuszyklen.		
Eine statische Speicherallokierung während der Assemblierung erreicht man durch geeignete Assemblerdirektiven.		
Bei VLIW-Prozessoren übernimmt der Compiler die statische Befehlszuteilung auf die Ausführungseinheiten des Rechenwerks.		
Bei der Memory-Mapped-Adressierung existieren zwei Adressräume, einer für Speicher und einer für Ein-/Ausgabe-Einheiten.		

4. Aufgabe: (7 P)

Gegeben sei ein 8-fach-satzassoziativer Cache-Speicher (A8) mit einer Cachelinegröße von 64 Byte. Die Hauptspeicheradresse umfasst 32 Bit. Der Satzindex ist 13 Bit breit. Zur Verwaltung eines Cache-Blocks werden das Valid-Bit (*V*) und das Dirty-Bit (*D*) verwendet.

a) Wie viele Bit umfasst der Tag?

b) Wie viele Befehle mit einer Länge von 32 Bit können sich gleichzeitig im Cache-Speicher befinden?

c) Bestimmen Sie den insgesamt erforderlichen Speicherbedarf für die Realisierung des Cache-Speichers (Adress- und Datenspeicher).

Gegeben sei ein direkt abbildender Cache-Speicher mit einer Speicherkapazität von 128 Byte und einer Blockgröße von 16 Byte. Als Aktualisierungsstrategie wird das Rückschreibverfahren (*write back*) verwendet. Nehmen Sie an, dass der Cache-Speicher zu Beginn leer ist. Betrachten Sie die folgenden Lese- und Schreibzugriffe auf die in hexadezimaler Schreibweise angegebenen Adressen.

d) Ergänzen Sie in dieser Tabelle die fehlenden Einträge. Verwenden Sie dabei Miss für Cache-Miss und Hit für Cache-Hit. Geben Sie in der letzten Zeile der Tabelle an, ob der entsprechende Cache-Block in den Hauptspeicher zurückkopiert werden muss (ja) oder nicht (nein).

Adresse	**64**	**32**	**E4**	**18**	**E0**	**7A**	**A2**	**F0**	**E3**
read / write	w	r	w	r	r	r	r	w	r
Index									
Tag									
Hit / Miss									
write back ?									

5. Aufgabe: (6 P)

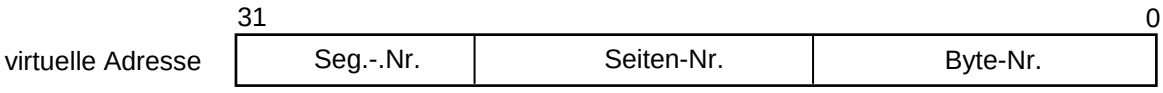
In einem 32-Bit-Rechnersystem wird das Segment- und das Seitenverfahren kombiniert eingesetzt. Die Segmentnummer umfasst 5 Bit. Das Segment S1 enthält ein Programm P1 der Größe 32 MB. In ein Segment passen maximal 2^{14} Seiten. Der Hauptspeicher ist zu einem Viertel der Größe des virtuellen Adressraums ausgebaut.

a) Wie groß kann ein Segment maximal sein?

b) Wieviele Frames passen maximal in den Hauptspeicher?

c) Wie groß ist der Verschnitt, wenn ein Programm P2 4096 Frames im Hauptspeicher belegt?

d) Skizzieren Sie für die obigen Angaben Aufbau und Adressierung eines 2-fach-satzassoziativen TLB mit 32 Zeilen, der zur Beschleunigung der Adressumsetzung eingesetzt werden soll



Gegeben sei nun eine Speicherverwaltungseinheit (MMU). Der virtuelle Speicher ist in 8 Seiten mit je 1 kByte unterteilt. Der physische Speicher hat eine Kapazität von 4 kByte. Der aktuelle Ausschnitt der Seitentabelle ist in folgender Tabelle angegeben.

Virtuelle Seitennummer	Physische Seitennummer
0	-
1	-
2	1
3	2
4	-
5	0
6	3
7	-

e) Ermitteln Sie die physischen Adressen (Dezimaldarstellung) zu den folgenden virtuellen Adressen (Dezimaldarstellung) :

2100, 4095

6. Aufgabe: (6 P)

In der folgenden Aufgabe soll für alle Teilaufgaben jeweils die in Tabelle 8.1 dargestellte Ausgangssituation herrschen. Das Zeichen $ kennzeichnet die hexadezimale Darstellung von Zahlen. Beachten Sie jeweils das Datenformat.

Prozessor-Register	
Datenregister	**Inhalt**
D0	$ 00 00 11 11
D1	$ 22 22 33 33
D2	$ 44 44 55 55
D3	$ 66 66 77 77
D4	$ 88 88 99 99
D5	$ AA AA BB BB
D6	$ CC CC DD DD
D7	$ EE EE FF FF

Adressregister	Inhalt
A0	$ 00 01 00 02
A1	$ 00 01 00 0A
A2	$ 00 01 00 08
A3	$ 00 01 00 00
A4	$ 00 01 00 00
A5	$ 00 01 00 00
A6	$ 00 01 00 00
A7	$ 00 01 00 00

Arbeitsspeicher (16 Bit)	
Adresse	**Inhalt**
$ 10000	$ 00 01
$ 10002	$ 02 03
$ 10004	$ 04 05
$ 10006	$ 06 07
$ 10008	$ 08 09
$ 1000A	$ 0A 0B
$ 1000C	$ 0C 0D
$ 1000E	$ 0E 0F
$ 10010	$ 10 11
$ 10012	$ 12 13
.	
.	
.	
$ FFFFA	$ 01 02
$ FFFFC	$ 03 04
$ FFFFE	$ 05 06

Tabelle 8.1 : Ausgangssituation Register und Hauptspeicher

a) Tragen Sie den Inhalt der Datenregister D0 und D1 (je 32 Bit) in Tabelle 8.2 ein, nachdem mit der Ausgangssituation entsprechend Tabelle 8.1 alle folgenden Befehle nacheinander ausgeführt wurden :

```
Z       EQU     $AA
K       DC.W    $44
ERG     DS.L    $2
        MOVE.W  K, D1
        ADD.W   #Z, D2
        ADD.L   D1, D2
        MOVE.L  D2, ERG
```

Prozessor-Register	
Datenregister	**Inhalt**
D1	$
D2	$

Tabelle 8.2 : Lösung zu Teilaufgabe a)

b) Tragen Sie den Inhalt der Datenregister D5, D6 und D7 sowie der Adressregister A0, A1 und A2 (je 32 Bit) in Tabelle 8.3 ein, nachdem mit der Ausgangssituation entsprechend Tabelle 8.1 die folgenden Befehle nacheinander ausgeführt wurden:

```
MOVE.L    (A1)+, D5
MOVE.W    (A1)+, D6
MOVE.B    -(A2),  D7
MOVE.W    -(A1), -(A0)
```

D5	$
D6	$
D7	$

Adressregister	Inhalt
A0	$
A1	$
A2	$

Tabelle 8.3 : Lösung zu Teilaufgabe b)

c) Tragen Sie den Inhalt der Datenregister D3 und D4 (je 32 Bit) in Tabelle 8.4 ein, nachdem mit der Ausgangssituation entsprechend Tabelle 8.1 die folgenden Befehle nacheinander ausgeführt wurden :

```
SUB.L  (A0)+, D3
LSR.L  #2, D4
```

Prozessor-Register	
Datenregister	**Inhalt**
D3	$
D4	$

Tabelle 8.4 : Lösung zu Teilaufgabe c)