Hinweise zur Klausur des Kurses 1709 am 02.08.2003

- Klausurdauer: drei Stunden (10:00 bis 13:00 Uhr)

- Überprüfen Sie die Vollständigkeit der Aufgabenstellungen! Die Klausur umfaßt insgesamt 10 Seiten und 6 Aufgaben.

- Zugelassene Hilfsmittel sind nichtprogrammierbarer Taschenrechner. Nicht zugelassen sind PC's sowie PC ähnliche Rechner wie Lap/Palm Top, schriftliche oder elektronisch gespeicherte Texte, welchen den Kurs 1709 betreffen; also insbesondere Kurstext, Einsendeaufgaben zum Kurs 1709 und deren Musterlösungen sowie kursbezogene Sekundärliteratur.

- Die zum Bestehen dieser Klausur zu erreichende Punktzahl liegt noch nicht fest. Sie wird jedoch sicherlich nicht höher sein, als die Hälfte der maximal erreichbaren Punkte.

- Füllen Sie bitte vor Inangriffnahme der Klausuraufgaben den/das Übungsschein/Zertifikat leserlich aus.

- Die Bescheinigung für das Finanzamt ist nur mit Unterschrift und Stempel gültig. Nur vollständig ausgefüllte Bescheinigungen werden von uns abgestempelt, unterschrieben und Ihnen zugestellt.

- Schreiben Sie auf alle Lösungsbögen Ihren Namen und Ihre Matrikelnummer !

- Bei Abgabe Ihrer Arbeit heften Sie bitte die ersten Seiten des übergebenen Klausurexemplares (Deckblatt, Bescheinigungen und Hinweise zur Klausur) vor Ihre Lösungsblätter. Kontrollieren Sie zum Schluß, daß Sie Ihre gesamte Arbeit geheftet abgeben. Nachträglich eingereichte Lösungen werden von uns nicht akzeptiert. Verwenden Sie nur Kugelschreiber oder Füllfederhalter für zu wertende Lösungen. Mit Bleistift geschriebene Lösungen (bzw. Lösungsteile) werden nicht berücksichtigt!

  Die Aufgabenstellung der Klausur können Sie nach der Klausur (bei Bedarf) mitnehmen.

Bei der Bearbeitung der Klausur wünschen wir Ihnen viel Erfolg!

Aufgabe 1: Grundlagen I (5 Punkte)

a) Was unterscheidet die Architektur von der Mikroarchitektur eines Prozessors?

b) Welche Rechner fallen in die SIMD-Klasse der Flynnschen Klassifizierung?

c) Worin unterscheidet sich ein Feldrechner von einem Vektorrechner?

d) Ein Direct-Mapped-Cache ist prinzipiell immer langsamer als ein 4-Wege-Assoziativ-Cache. Richtig oder Falsch? Begründen Sie.

e) Je mehr Stufen eine Pipeline für einen gegebenen Befehlssatz besitzt, desto schneller kann das System getaktet werden. Richtig oder Falsch? Begründen Sie.

Aufgabe 2: Pipelining (12 Punkte)

Ausgangspunkt des Befehls-Pipelining ist die Aufspaltung eines Befehls in Teilschritte. Es setzt voraus, dass für die zugrundelegende Befehlsarchitektur eine Folge von Teilschritten gefunden wird, die für *alle* Befehle gleich ist. Alle Befehle (inklusive der zugehörigen Adressierungsarten) können beispielsweise in den folgenden fünf Teilschritten, welche Ihnen aus DLX-Architektur bekannt sind, bearbeitet werden (*Tabelle 1*).

1. Instruction Fetch (IF): Befehl holen (OpCode und Registeradressen können in einem Maschinenwort untergebracht werden).

2. Instruction Decode (ID): Befehl dekodieren und gleichzeitig die Quelloperanden aus dem Registerblock lesen.

3. Execute (EX): Führe eine arithmetische bzw. logische Operation mit den Operanden aus.

4. Memory Access (MEM): Daten holen (*LOAD*) oder Speichern (*STORE*)

5. Write Back (WB): Ergebnis in einem prozessorinternen Register speichern.

IF	ID	EX	MEM	WB		
	IF	ID	EX	MEM	WB	
		IF	ID	EX	MEM	WB

Tabelle 1: Zeitlicher Ablauf der Teilschritte bei konfliktfreiem Befehls-Pipelining

Pipelinekonflikte können durch zusätzliche Hardware oder Software (Einfügen von `NOOP`, *no-operation-Befehle*) behoben werden. In den folgenden Teilaufgaben soll nur die Software-Lösung betrachtet werden. Wir untersuchen zunächst einen Pipelinekonflikt durch eine echte Datenabhängigkeit (*RAW Read After Write*), der in dem folgenden Programmfragment vorliegt:

`ADD R3,R1,R2`	R3=R1+R2
`SUB R4,R1,R3`	R4=R1-R3

a) Wie viele NOOP-Befehle sind zum Beheben dieses Konflikts nötig? Skizzieren Sie den Ablauf des Programms in der Pipeline und markieren Sie im Zeitdiagramm folgende Ereignisse:[1]

A) Wert von R3 durch den ADD-Befehl errechnet,

B) Wert von R3 liegt *vor* dem Registerblock an,

C) Wert von R3 wurde im Registerblock gespeichert und

D) R3 liegt als Eingangsoperand für den Sub-Befehl an der ALU vor.

b) Nun betrachten wir einen Kontrollfluß-Konflikt, der im folgenden Programmfragment für einen DLX Prozessor vorliegt:

1000	BEQZ 20	Falls Zero-Bit 1 ist, springe um 20 Wörter relativ zum aktuellen Befehlszählerstand
...	... ...	...
1024	ADD R1,R2,R3	R1=R2+R3

Wie viele NOOP-Befehle sind zum Beheben dieses Konflikts nötig? Skizzieren Sie den Ablauf des Programms in der Pipeline und markieren Sie im Zeitdiagramm (*Aufbau wie in Tabelle 1*) folgende Ereignisse:

i. Neuer Befehlszählerstand bestimmt und

ii. Wert von R1 wurde berechnet.

iii. Durch Umstellen von Befehlen kann ein optimierender Compiler die zur Konfliktbehebung erforderlichen NOOP-Befehle durch andere (unabhängige) Befehle ersetzen und so die Effizienz des Pipelining steigern. Gegeben seien weitere Befehle aus dem Umfeld des Programmfragments von Teilaufgabe a):

JR 20	
SUB R6,R2,R1	R6=R2-R1
AND R5,R2,R1	R5=R2∧R1
... ...	...
ADD R3,R1,R2	R3=R1+R2
SUB R4,R1,R3	R4=R1+R3
... ...	...
ADD R1,R4,R5	R1=R4+R5
XOR R6,R7,R8	R6=R7⊕R8

Eliminieren Sie die NOOP-Befehle in Teilaufgabe a) durch geeignetes Umstellen von Befehlen.

[1] Es seien weder *bypass* noch andere *forwarding-Hardware* vorhanden.

Aufgabe 3: Prediction (12 Punkte)

Das folgende Programm besteht aus zwei verschachtelten Schleifen. Das Register R4 sei mit $m > 0$ und das Register R3 mit $n > 0$ vorbelegt. Weiterhin sei das Register R1 mit 1 belegt.

Loop1:	SLE R10, R3, R0	(1)	; R3 <= R0 ? Status in R10
	BNEZ R10, Ende1	(2)	; wenn ja, dann gehe zu Ende1
Loop2:	SLE R11, R4, R0	(3)	; R4 <= R0 ? Status in R11
	BNEZ R11, Ende2	(4)	; wenn ja, dann gehe zu Ende2
...	... ...	...	...
	SUB R4, R4, R1	(19)	; R4 = R4 - R1
	J Loop2	(20)	; Goto Loop2
Ende2:	SUB R3, R3, R1	(21)	; R3 = R3 - R1
	J Loop1	(22)	; Goto Loop1
Ende1:	...	(23)	;

Es sollen verschiedene Sprungvorhersagetechniken verglichen werden, aber dabei nur bedingte Sprünge betrachtet werden. Wie viele richtige und falsche Vorhersagen gibt es bei:

a) statischer Sprungvorhersage mit „Always taken"-Technik,

b) dynamischer Sprungvorhersage: Ein-Bit-Prädiktor (Initialzustand „predict taken"),

c) dynamischer Sprungvorhersage: Zwei-Bit-Prädiktor mit Sättigungszähler (Initialzustand: „predict weakly taken")?

Begründen Sie Ihre Ergebnisse durch Aufstellen einer Tabelle.

Erläuterung der Abkürzungen:

√	Sprung richtig vorhergesagt
×	Sprung falsch vorhergesagt
g	Sprung wurde genommen
ng	Sprung wurde nicht genommen
T	predict taken
NT	predict not taken
ST	predict strongly taken
WT	predict weakly taken
WNT	predict weakly not taken
SNT	predict strongly not taken

Sprung		Ein-Bit-Prädiktor		Zwei-Bit_Prädiktor	
äußere Schleife	innere Schleife	äußere Schleife	innere Schleife	äußere Schleife	innere Schleife
Initialzustand		T	T	WT	WT
	⋮		⋮		⋮
⋮	⋮	⋮	⋮	⋮	⋮

Aufgabe 4: Scheduling & Renaming (8 Punkte)

Gegeben sei ein Maschinenprogramm und ein superskalarer Prozessor mit zwei Integer-ALUs (IU) und zwei Load/Store-Einheiten (LS). Bis auf die Multiplikation benötigen alle Operationen nur einen Taktzyklus. Die Multiplikation erfordert zwei Taktzyklen.

1.	LOAD	R1,[A]
2.	LOAD	R2,[B]
3.	MUL	R3,R1,R2
4.	LOAD	R1,[C]
5.	LOAD	R2,[D]
6.	MUL	R4,R1,R2
7.	ADD	R1,R3,R4
8.	STORE	[E],R1
9.	SUB	R2,R3,R4
10.	STORE	[F],R2

a) Wie viele Taktzyklen würden zur Ausführung des Maschinenprogramms auf einem skalaren Prozessor benötigt, der über nur je eine der o.g. Ausführungseinheiten verfügt und der diese nur nacheinander benutzen kann. Berücksichtigen nur die Zeiten, die von den Ausführungseinheiten benötigt werden!

b) Die Befehle sollen nun den o.g. Ausführungseinheiten des superskalaren Prozessors so zugeordnet werden, dass die Zeit zur Ausführung minimiert wird. Dabei darf zwar die vorgegebene Befehlsreihenfolge **nicht aber** die Registerzuordnung verändert werden (*Scheduling*).

c) Durch Umbenennung einiger Register (*register renaming*) kann die Ausführungszeit weiter verringert werden. Bestimmen Sie die maximal erreichbaren Beschleunigung und vergleichen Sie diesen Wert mit dem theoretischen Optimalwert.

Aufgabe 5: Cache (9 Punkte)

Gegeben seien drei Cache-Speicher **DM**, **A2** und **AV**, die jeweils acht Cacheblöcke besitzen, wobei jeder Cacheblock vier Bytes umfasst. Der Cache **DM** ist als direkt-abgebildeter Cache (*direct mapped*) organisiert, Cache **A2** als 2-fach assoziativer Cache (*2-way-set-associative*); Cache **AV** ist vollassoziativ (*fully associative*). Bei den Cachespeichern **A2** und **AV** soll die *„least recently used"* -Ersetzungsstrategie LRU angewendet werden. Nehmen Sie an, die Cachespeicher seien zu Beginn leer, und es soll eine Serie von einzelnen Bytes mit den folgenden 16-Bit-Adressen (**dezimal**!) gelesen werden:

70	09	39	83	66	68	35	80	93	67	79	37	84	09

	0	1	2	3	4	5	6	7	8	9
0									71	32
10	78	69								
20										
30			6	12	20	00	72	67	73	76
40										
50										
60					90	82	69	72	19	10
70	62	0					67	69	85	76
80	78	85	87	75	32	72	67	83		
90			76	76	79	84				
Speicherbelegung										

Die linke Tabelle gibt den für diese Aufgabe relevanten Inhalt des Hauptspeichers im Adressbereich 0 bis 99 wieder.

a) Geben Sie für die drei Cachespeicher an, wie viele Bits zur Verwaltung eines Cacheblocks benötigt werden. Dabei sollen für den Zustand des Cacheblocks zwei Statusbits verwendet werden (Valid-Bit und Dirty-Bit).

b) Geben Sie für die drei Cachespeicher die Anzahl der erforderlichen Vergleicher und die jeweils zu vergleichende Bitanzahl an.

c) Welche Zeilen sind bei diesen Caches hinsichtlich einer Blockersetzung gleichzeitig auszuwerten, z.B. für die Ersetzungsstrategie nach dem LRU- Prinzip?

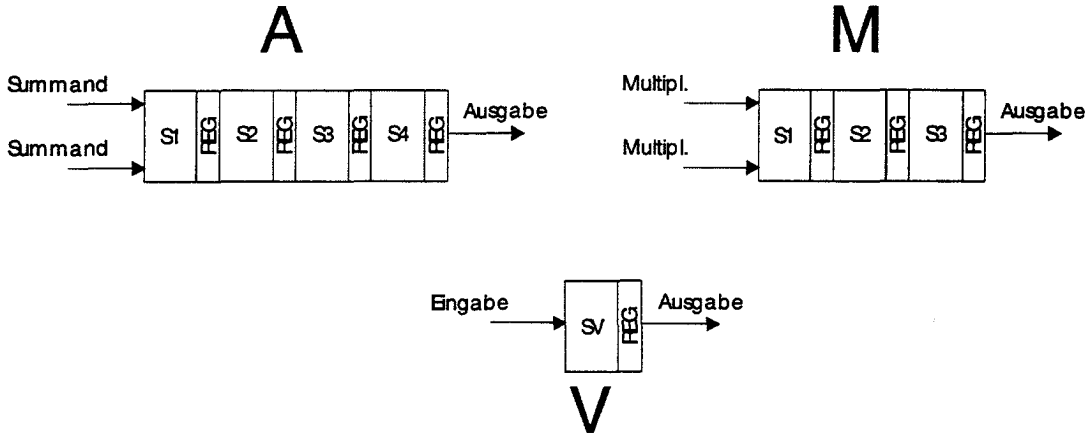
d) Geben Sie nun tabellarisch für jeden Cache an, ob es sich beim Lesezugriff auf die jeweilige Adresse um einen Treffer (Cache-Hit) oder um keinen Treffer (Cache-Miss) handelt.

e) Stellen Sie den Zustand der drei Caches nach dem letzten Speicherzugriff dar, d.h. für jeden Cacheblock den Cache-Tag und die vier Datenbytes. Entnehmen Sie die jeweils vier Datenbytes aus der obigen Speicherbelegung.

Aufgabe 6: Pipelineprozessor (14 Punkte)

Als Bauelemente für einen Pipelineprozessor seien folgende Funktionseinheiten gegeben:

- dreistufige Pipeline-Gleitkomma-Multiplizierer M,
- vierstufige Pipeline-Gleitkomma-Addierer A
- und Verzögerungselemente V.



a) Konstruieren Sie aus den oben gegebenen Elementen[2] eine synchrone Pipeline (alle Register werden mit demselben Taktsignal angesteuert), die aus einem Eingabestrom $e(1),e(2),e(3),...$ von Gleitkommazahlen die Ausgabewerte:

$$a(j)=e(i)+e(i+1)+e(i+2)\cdot e(i+3)$$

berechnet. Hierbei soll $e(i)$ während des i-ten Taktzyklus anliegen. Zeichnen Sie ein Blockdiagramm!

b) Die Stufen eines Multiplizierers bzw. eines Addierers besitzen folgende Verzögerungszeiten τ_{mi} bzw. τ_{ai}:

τ	S1	S2	S3	S4
Addierer	12 ns	14 ns	17 ns	11 ns
Multiplizierer	13 ns	16 ns	15 ns	

Die Verzögerungszeiten der Register bzw. der Verzögerungselemente beträgt τ_{reg}=3 ns und τ_v=2ns. Wie groß ist die maximale Taktfrequenz f dieser Pipeline?

c) Wie viele Verarbeitungsaufträge n können während einer Zeitspanne von 660 ns ausgeführt werden? Berechnen Sie den zugehörigen Durchsatz D_k (Anzahl Verarbeitungsaufträge / 660 ns). Ein Taktzyklus T_k dauert $1/f$ Sekunden. Setzen Sie voraus, dass die Pipeline ständig gefüllt ist.

---

[2] es dürfen durchaus mehrere Komponenten des gleichen Typs verwendet werden.