Bitte hier unbedingt Matrikelnummer und Adresse eintragen, sonst keine Bearbeitung möglich.

Postanschrift: FernUniversität, D-58084 Hagen

Name, Vorname

Straße, Nr.

PLZ, Wohnort

FERNUNIVERSITÄT
- Gesamthochschule -
EINGANG

INF

FERNUNIVERSITÄT
Gesamthochschule
D-58084 Hagen

Fachbereich Informatik

Kurs: 1708 „Technische Informatik II" und 1705 „Mikrorechner-Technik"

Hauptklausur am 03.02.2001

belegter Kurs	Hörerstatus:	Klausurort:
☐ 1705	☐ Vollzeitstudent	☐ Berlin
☐ 1708	☐ Teilzeitstudent	☐ Bochum
	☐ Zweithörer	☐ Frankfurt
	☐ Gasthörer	☐ Hamburg
	☐ Bachelor	☐ Karlsruhe
	☐ Lehramt	☐ Köln
	☐	☐ München
		☐ Bregenz
		☐ Wien
		☐ . . . . .

Zutreffendes unbedingt ankreuzen !

Aufgabe	1	2	3	4	5	Summe
erreichbare Punktzahl	10	20	25	20	25	100
bearbeitet						
erreichte Punktzahl						

Note: _____

Hagen, den _____ Betreuer: _____

©2001 FernUniversität - Gesamthochschule - in Hagen

Aufgabe 1: (10 Punkte)

Geben Sie für die folgenden Aussagen an, ob sie richtig oder falsch sind.

		ja	nein
a)	In einem *Direct-Mapped Cache* werden die Daten immer unter eindeutigen Speicheradressen abgelegt.	☐	☐
b)	In einem statischen Schreib/Lese-Speicherbaustein bleiben die Daten auch nach dem Abschalten der Betriebsspannung erhalten.	☐	☐
c)	Unter *Hardware Interlocking* versteht man eine Maßnahme, den Systembus gegen unerlaubte Zugriffe von externen Rechnerkomponenten zu sperren.	☐	☐

Ergänzen Sie die folgenden Sätze:

d) Unter dem **Auffrischen** von dynamischen Speicherbausteinen versteht man.....
.....
.....
.....

e) Die **schnellste Form der Datenübertragung** zwischen dem Hauptspeicher und einer Ein-/Ausgabe-Schnittstelle wird .....
..... genannt
und benötigt im Idealfall für jedes übertragene Datum ..... Buszyklus/Buszyklen.

f) Ein wesentlicher Unterschied zwischen den **bitorientierten** bzw. **zeichenorientierten Protokollen** der synchronen seriellen Datenübertragung besteht darin, daß .....
.....
.....
.....

Aufgabe 2: (nur Kurs 1705) (20 Punkte)

Ein einfacher RISC-Prozessor besitze nur die in der folgenden Tabelle angegebenen sechs verschiedenen Adressierungsarten, von denen für arithmetische, logische und Verschiebebefehle nur die erste benutzt wird. Für diese Befehle müssen alle Operanden in den Registern des Prozessors vorliegen. Die letzten fünf Adressierungsarten dienen nur in den Befehlen LOAD und STORE zur Selektion einer Speicherzelle sowie in Sprung- und Verzweigungsbefehlen zur Angabe des Sprungzieles. (EA: effektive Adresse).

Tabelle: **Adressierungsarten des RISC-Prozessors**

Bezeichnung	Notation	Effektive Adresse
für arithmetische, logische, Verschiebebefehle und Transferbefehle		
Explizite Register-Adressierung	Ri	EA = i
zusätzlich für LOAD/STORE-Befehle		
Unmittelbare Adressierung (nur Load)	#<Operand>	EA = (PC)
Register indirekte Adressierung	(Ri)	EA = (Ri) (Inhalt von Ri)
Register-relative Adressierung:	<Offset>(Ri)	EA = (Ri) + <Offset>
Register-relative Adressierung mit Index	(Ri)(Rj)	EA = (Ri) + (Rj).
Programmzähler-relative Adressierung	<Offset>(PC)	EA = (PC) + <Offset>

Der Prozessor besitze einen Satz von 32 universellen 32-bit-Registern R31,..,R0, die sowohl als Daten- wie Adreßregister benutzt werden können. Es stehen alle im Kurs beschriebenen Befehle zur Verfügung. Der RISC-Prozessor unterstütze nur das Zweiadreßformat für Operanden, die in Registern zur Verfügung stehen. Dabei entspreche dem Befehl

<Operation> Ri,Rj die Zuweisung *Ri = Ri <op> Rj.*

a) Geben Sie für die Nachbildung der im Kurs beschriebenen komplexen CISC-Adressierungsarten jeweils eine RISC-Befehlsfolge für den Lesebefehl *LD R0,<Operand>* an. Kommentieren Sie jeden RISC-Befehl. Versuchen Sie, mit möglichst wenigen Registern Ri auszukommen.

1. **Absolute Adressierung**: LD R0,<Adresse>

2. **Indirekte absolute Adressierung**: LD R0,(<Adresse>)

3. **Indirekte, indizierte Adressierung**: *LD R0,(<Offset>(R1)(R2))*

4. **Indizierte, indirekte Adressierung**: LD R0, <2.Offset>(<1.Offset>(R1))(R2)

b) Der RISC-Prozessor verfüge über keine speziellen Flag-Manipulationsbefehle. Geben Sie Befehlsfolgen an, mit der in einem Register Rj das Bit i gezielt gesetzt bzw. zurückgesetzt werden kann – alle anderen Bits aber unverändert bleiben. Kommentieren Sie die Befehle.

a. **Setzen**: (Vorgaben: Rj = R0, Bit i = Bit 15)

b. **Zurücksetzen**: (Vorgaben: Rj = R0, Bit i = Bit 22)

Aufgabe 2: (nur Kurs 1708) (20 Punkte)

Ein Intel-Prozessor soll mit dem Software-Interrupt[1] **INT 33** über die IDT (*Interrupt Descriptor Table*) auf eine Ausnahmebehandlungs-Routine (*Exception Handler*) zugreifen.

a) Unter welcher Adresse liegt der entsprechende Eintrag in der IDT, wenn das IDT-Register die Basisadresse $0000 A134 enthält ? (Angabe des Rechenwegs erforderlich.)

b) Spezifizieren Sie den IDT-Eintrag für den o.g. *Exception Handler*, wenn er die folgenden Bedingungen erfüllen soll:

- sämtliche Daten des unterbrochenen Prozesses sollen dem *Interrupt Handler* zur Verfügung stehen;
- der *Interrupt Handler* soll auch von anderen Unterbrechungsanforderungen als dem NMI (*Exceptions*) unterbrochen werden können;
- es soll gewährleistet sein, daß der *Interrupt Handler* nicht von einem Prozeß auf der Privileg-Ebene 3, jedoch von einem auf der Ebene 2 aufgerufen werden kann;
- das Programm zum *Interrupt Handler* beginnt ab der logischen Adresse[2] $0A16:3CB0ABC1.

Geben Sie in der folgenden Tabelle den IDT-Eintrag in binärer und hexadezimaler Form an (‚d' für ‚unbestimmt' - *don't care*, ‚0' für ‚ungenutzt'). Kennzeichnen Sie die einzelnen Bitfelder.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Hex.	
Byte 7,6																		
5,4																		
3,2																		
1,0																		

[1] Angabe der Interrupt-Nummer 33 in dezimaler Form !
[2] (<16-bit-Selektor>:<32-bit-Offset>)

c) In welcher Tabelle (GDT, LDT) befindet sich der (Code-)Segment-Deskriptor der Ausnahmebehandlungs-Routine und welchen Eintrag in dieser Tabelle belegt er ? Tragen Sie dazu den Selektor in die folgende Tabelle ein, kennzeichnen Sie die verschiedenen Bitfelder und begründen Sie Ihre Antwort auf die Fragen.

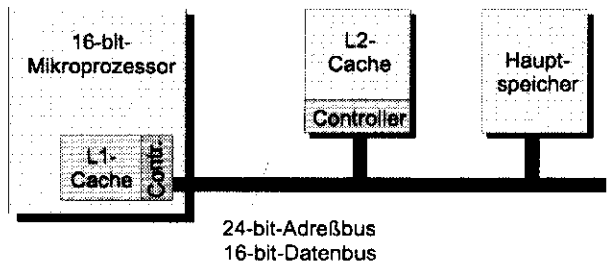
Selektor:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Begründung:

Aufgabe 3: (25 Punkte)

Gegeben sei ein 16-bit-Mikroprozessor (mit 24-bit-Adreßbus), der auf dem Chip einen integrierten kleinen Cache, den L1-Cache (*Level-1 Cache*), enthält. Zwischen Prozessor und Hauptspeicher sei ein weiterer Cache, der L2-Cache (*Level-2 Cache*), angeordnet. Bei jedem lesenden oder schreibenden Datenzugriff wird zunächst auf den L1-Cache zugegriffen und das referenzierte Datum dort gesucht. Nur wenn es dort nicht gefunden wird (*L1-Cache Read/Write Miss*), wird auch der externe L2-Cache angesprochen. Wird auch dort kein Treffer erzielt (*L2-Cache Read/Write Miss*), wird der Hauptspeicher adressiert.



Der L1-Cache sei als *Direct Mapped Cache* mit dem Rückschreibverfahren organisiert und enthalte maximal 64 Einträge mit je 4 byte. Der L2-Cache sei ebenfalls als *Direct Mapped Cache* organisiert. Er werde aber nach dem Durchschreibverfahren mit *Write Around* verwaltet, d.h. bei einem *Write Miss* wird das Datum nur im Hauptspeicher abgelegt. Er umfasse maximal 1024 Einträge mit je 4 byte.

a) Kennzeichnen Sie für beide Caches im folgenden Bild die Einteilung einer Speicheradresse in *Tag*, Index und Byteadresse.

L1-Cache

Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

L2-Cache

b) Berechnen Sie für beide Caches die jeweilige Speicherkapazität für *Tag*-RAM und Datenspeicher.

c) In der folgenden Tabelle sind für die zunächst gelöschten (*flushed*) Caches die Einträge nach den ersten Speicherzugriffen – in den grau unterlegten vier oberen Zeilen - dargestellt. Sie sind durch ihren Index, ihren *Tag* und die 4-byte-Datenblöcke gegeben.
Ergänzen Sie die Tabelle um die Auswirkungen der unter der Tabelle stehenden sechs Speicherzugriffe durch Lesebefehle (*Load* - LD) und Schreibbefehle (*Store* - ST) mit 16-bit-Registern als Quelle oder Ziel der Daten. Dabei seien nicht ausgerichtete (*non aligned*) Datenzugriffe auf ungerade Adressen möglich.

- Tragen Sie in den Spalten ‚Op.' die Nummer der Operation ein, durch die ein Eintrag angesprochen oder verändert wird.

- Streichen Sie Einträge, die durch eine Operation verändert werden, und ergänzen Sie die Tabelle durch den geänderten Eintrag.
- Ergänzen Sie die unten stehenden sechs Operationen durch die Angaben, ob für den L1- bzw. L2-Cache ein *Read Hit, Read Miss, Write Hit* oder *Write Miss* vorliegt. Geben Sie für die Lesebefehle zusätzlich den Ergebniswert im verwendeten 16-bit-Register an. (Beachten Sie dabei, daß bei einem *L1-Cache Hit* auf den L2-Cache nicht mehr zugegriffen wird.)

L1-Cache				L2-Cache				Hauptspeicher		
Op.	Index	Tag	Datum	Op.	Index	Tag	Datum	Op.	Adr.	Datum[3]
	38	F04C	45 DE 23 D0		7E2	5F6	DE 89 F5 3D		FA4027	FF 45 23 3A
	0D	F400	00 11 22 33		338	FA4	45 DE 23 D0		012FDC	CC AC 42 3A
	2D	023D	FC EE 56 32		3AE	022	01 23 45 67		DCE43C	CD ED FD 21
	1E	AA46	77 88 DD F0		10F	DEE	CA 5F DE 44		022EB8	01 23 45 67

		Registerwert	L1-Cache	L2-Cache
1.	LD R1,$022EB8	R1=$		
2.	ST R2,$023DB6	R2=$564D		
3.	LD R3,$F04CE2	R3=$		
4.	ST R4,$FA4CE0	R4=$FDE0		
5.	LD R5,$DCE43D	R5=$		
6.	ST R6,$012FDD	R6=$D456		

Zur Auswertung der Operandenadressen können Sie die folgende Hilfstabelle zum Eintragen der binären Adresse für jede Operation verwenden.

Operation																							
1																							
2																							
3																							
4																							
5																							
6																							

[3] Jeder Eintrag enthält hier zwei 16-bit-Speicherwörter. Die angegebene Adresse bezieht sich jeweils auf das rechts stehende Byte

Aufgabe 4: **(20 Punkte)**

Ein Mikroprozessor besitze einen 32 bit breiten Datenbus und einen 32 bit breiten Adreßbus. Er erlaube den nicht ausgerichteten (*non-aligned*) Zugriff auf 1-byte-, 2-byte- und 4-byte-Daten im Speicher (Byte, Wort, Doppelwort). Dabei geschieht die Adressierung eines Speicherwortes durch die 30 höherwertigen Adreßbits A31,A30,...,A3,A2; die Selektion des Datums im Speicherwort geschieht durch die niederwertigen Adreßbits A1, A0 sowie durch zwei Bits WL1 und WL0 zur Längenangabe des Datums (*Word Length*). Aus A1, A0, WL1, WL0 erzeugt ein spezieller Decoder die Signale BE3,..,BE0 (*Byte Enable*) zur Auswahl der angesprochenen Bytes im Speicherwort. Da Wort- und Doppelwort-Daten über zwei aufeinanderfolgende Speicherwörter verteilt sein können, müssen in diesen Fällen zwei konsekutive Speicherzugriffe mit entsprechend veränderter Oberadresse A:=A31,..,A2 sowie A1,A0 und Auswahlsignalen BE3,..,BE0 stattfinden. Die Erhöhung der Oberadresse A und Anpassung der Bits A1, A0 wird dabei durch die Systembusschnittstelle vorgenommen.

a) Tragen Sie in der folgenden Tabelle die fehlenden Belegungen für die Adreß- und Auswahlsignale A, Ai, BEi ein. Unterscheiden Sie, wo erforderlich, bei *non-aligned* Datenzugriffen den ersten (Z=0) und zweiten Zugriff (Z=1). (Zur Erleichterung des Verständnisses ist in der Tabelle ein Eintrag bereits aufgefüllt.)

Wortlänge	WL1	WL0	A / A+1	A1	A0	Z	BE3	BE2	BE1	BE0
1 byte (Byte)	0	0								
	0	0								
	0	0								
	0	0								
2 byte (Wort)	0	1								
	0	1	A --	0 --	1 --	0 --	0 --	1 --	1 --	0 --
	0	1								
	0	1								
4 byte (Doppel-Wort)	1	0								
	1	0								
	1	0								
	1	0								

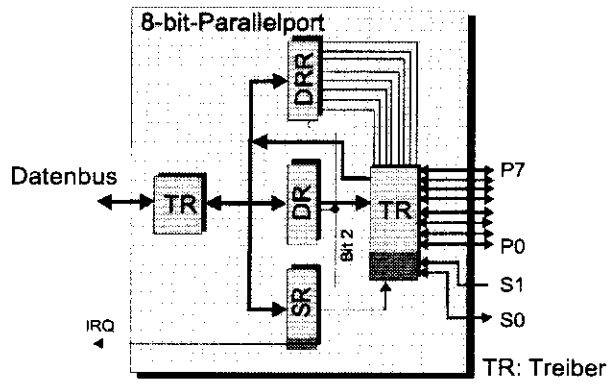
b) Geben Sie das Auswahlsignal BE0 als Funktion der Variablen WL1, WL0, A1, A0, Z an. (Es reicht ein genaues Hinschauen, eine formale Entwicklung ist hier nicht nötig.)

c) Geben Sie das Auswahlsignal BE1 als Funktion der Variablen WL1, WL0, A1, A0; bei 2- bzw. 4-byte-Zugrifen aber nur für den 1. Speicherzugriff Z=0 ! (Auch hier reicht ein – mehrfaches - genaues Hinschauen, eine formale Entwicklung ist nicht nötig.)

d) Begründen Sie, warum die Angaben in der oben stehende Tabelle nicht ausreichen, das Auswahlsignal BE1 auch für den 2. Zugriff auf 4-byte-Wörter eindeutig zu bestimmen und die Systembusschnittstelle deshalb ein weiteres Signal erzeugen und zur Verfügung stellen muß. Wie könnte solch ein Signal erzeugt werden ?

Aufgabe 5: **(25 Punkte)**

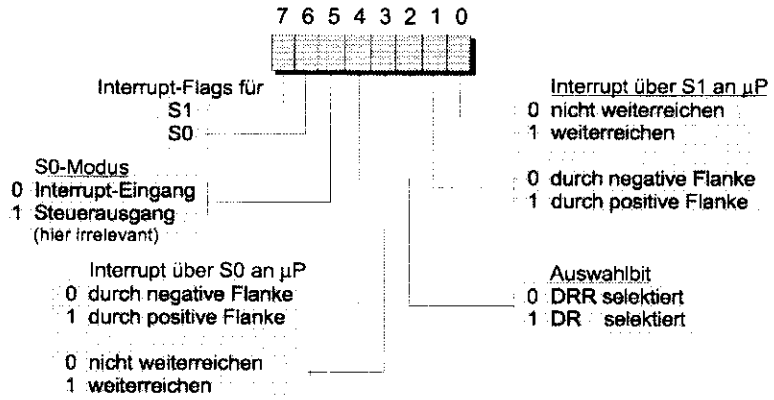
Gegeben ist ein Portbaustein mit einem 8 bit breiten bidirektionalen Parallelport P0,..,P7, einem Interrupteingang S1 und einem multifunktionalen Steuerein-/ausgang S0, der u.a. als zweiter Interrupteingang benutzt werden kann. Der Baustein enthält die im folgenden Blockschaltbild dargestellten Komponenten.



Die Funktion des Datenregisters **DR** (*Data Register*) sowie des Datenrichtungsregisters **DRR** (*Data Direction Register*) wurden bereits im Kurs 1705/1708 ausführlich beschrieben. Hier sei nur erklärt, daß für jede Portleitung P_i individuell festgelegt werden kann, ob sie als Eingang oder Ausgang betrieben wird. Dabei gilt:

$DRR_i = 0$: P_i ist Eingang; $DRR_i = 1$: P_i ist Ausgang.

Das sog. Steuerregister **SR** vereinigt die Funktion eines Steuer- und eines Statusregisters:



Bit 2 ist ein Auswahlbit zur Adressierung des Datenrichtungsregisters DRR oder des Datenregisters DR. Um diese Register mit Lese- oder Schreibbefehlen anzusprechen, muß vorher das Bit 2 des Steuerregisters entsprechend gesetzt werden. Der Zustand der Bits 5, 4, 3 legt die Funktion des Ein-/Ausgangs S0 fest. Bit 1 und Bit 0 legen die Betriebsart des Interrupteingangs S1 fest.

Bit 7 und Bit 6 sind die Interrupt-Flags für die Leitungen S1 und S0. Sie zeigen als Nur-Lese-Flags (*Read-Only*) jede an den Interrupt-Eingängen S1 bzw. S0 anliegende Unterbrechungsanforderung an und werden durch Schreibbefehle nicht beeinflußt. Gelöscht werden sie gemeinsam durch einen Lesezugriff auf das Datenregister DR (nicht DRR !). Ob diese Unterbrechungsanforderungen an den Prozessor weitergeleitet werden oder nicht, entscheiden die Bits 3 bzw. Bit 0.

a) Welche Werte müssen in die Register DRR und SR eingetragen werden, damit die Portleitungen P7,..,P4 als Ausgänge, P3,..,P0 als Eingänge im Programm verwendet werden können? Kennzeichen Sie nicht relevante Bits durch ‚d' (*don't care*).

Reg. \ Bit	7	6	5	4	3	2	1	0	hex
DRR									
SR									

b) Geben Sie ein Programmstück[4] als Folge von Assembler-Befehlen (bzw. Assembler-ähnlichen Befehlen) an, mit der der Portbaustein den Anforderungen in a) entsprechend initialisiert wird und das einen unmittelbar folgenden Zugriff auf das Datenregister DR ermöglicht. Numerieren und kommentieren Sie jeden Befehl.

c) Modifizieren Sie nun die Initialisierung aus a) und b) so, daß eine an S1 eintreffende negative Flanke als Unterbrechungsanforderung gewertet, aber nicht an den Prozessor weitergeleitet wird. An S0 sollen positive Flanken als Unterbrechungsanforderungen vermerkt, aber ebenfalls nicht an den Prozessor weiter gemeldet werden. (Es reicht, nur die geänderten, mit ihrer Nummer versehenen Befehle aufzuschreiben.) Geben Sie den erforderlichen Wert des Registers SR in binärer und hexadezimaler Darstellung an (‚d' = *don't care*).

Reg. \ Bit	7	6	5	4	3	2	1	0	hex
SR									

d) Ergänzen Sie das unter b) und c) ermittelte Programmstück durch eine Befehlsfolge, die

1. das Status/Steuerregister SR einliest, auswertet und danach
2. falls eine Unterbrechungsanforderung an S1 vorliegt,
   - das Datenregister DR liest,
   - den über DR eingelesenen Wert um vier Bitpositionen (genauer 4 mal um eine Bitposition) nach links schiebt,
   - den verschobenen Wert über DR wieder ausgibt,
   - nach 1. zurückspringt;

   sonst mit 3. fortfährt.
3. falls eine Unterbrechungsanforderung an S0 vorliegt, das Programmstück beendet, im anderen Fall nach 1. zurückspringt.

[4] Benutzen Sie dazu die in der unten stehenden Hilfstabelle angegebenen Assembler-Befehle und ergänzen Sie die Tabelle bei Bedarf um weitere Befehle.

e) Zeichnen Sie ein **Flußdiagramm** des Programmstücks.

Hilfstabelle der möglichen Befehle, Registerbezeichnungen und Adressierungsarten:

Mnemoc.	Bedeutung	Mnemoc.	Bedeutung
DR, DRR SR A	Register des Bausteins (s.o.), Akkumulator A	# $ Li	Unmittelbare Adressierung, Hexadezimalzahl, Marke (i ≥ 0)
LDA	Lade Akku A	ANDA	Und-Verknüpfung mit Akku A
STA	Speichere Akku A	ORA	Oder-Verknüpfung mit Akku A
LSLA	Verschiebe A um 1 Bit nach links	BRA	Verzweige unbedingt
CLR	Lösche Register	BEQ	Verzweige, falls Ergebnis = 0
CLRA	Lösche Akku A	BNE	Verzweige, falls Ergebnis ≠ 0