Lehr– und Forschungsgebiet Informatik
Leistungsbewertung und Verteilte Systeme
Prof. Dr. Ir. B. Haverkort

RWTH Aachen
Ahornstraße 55
52074 Aachen

Diplomvorprüfung im Fach „Rechnerstrukturen“ (Sommersemester 1997)

Aufgabe 1: Boolesche Algebra (18 Punkte)

Gegeben sei folgende dreistellige Boolesche Funktion F:

x_1	x_2	x_3	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a) Wie sind die Maxterme sowie die Maxtermdarstellung einer Booleschen Funktion f definiert? Zeigen Sie durch einen Widerspruchsbeweis, daß die Maxtermdarstellung einer beliebigen Booleschen Funktion f stets eindeutig ist! **(6 P.)**

(b) Geben Sie die Minterme für obige Funktion F an! Ist im Falle von F eher die Maxterm- oder die Mintermdarstellung vorzuziehen (Begründung)? **(2 P.)**

(c) Berechnen Sie für die oben gegebene Funktion F die komplementfreie Ringsummenentwicklung nach Reed-Muller! **(4 P.)**

(d) Sei I_F die Menge der einschlägigen Indizes von F. M_F sei definiert als die Menge aller dreistelligen Booleschen Funktionen, die man erhält, indem man entweder ein Element aus I_F entfernt oder aber einen weiteren einschlägigen Index zu I_F hinzufügt. Schließlich bezeichne allgemein $F(.,1,.)$ diejenige *zwei*stellige Boolesche Funktion, die man aus dem *drei*stelligen F gewinnt, wenn man $x_2 = 1$ konstant hält.

Worin unterscheidet sich die letzte Spalte der Funktionstabelle eines beliebigen Elementes von M_F von der letzten Spalte in der oben gegebenen Funktionstabelle von F? Geben Sie ein $F' \in M_F$ an, so daß das Funktionensystem $\{F'(.,1,.)\}$ funktional vollständig ist, und weisen Sie dies explizit nach! Ist auch $\{F'(.,0,.)\}$ funktional vollständig? **(6 P.)**

Bitte schreiben Sie deutlich!

Aufgabe 2: Varianzwerk (22 Punkte)

Bei der Durchführung beliebiger Messungen interessiert man sich oft dafür, wie sehr die Meßergebnisse um den Mittelwert der Messung schwanken. Wichtigste Größe in diesem Zusammenhang ist die sogenannte *Varianz*, die definiert ist als die mittlere quadratische Abweichung vom Durchschnitt. Haben wir also im allereinfachsten Fall zwei Meßergebnisse a und b, so beträgt ihr Mittelwert

$$m = \frac{a+b}{2}$$

und ihre Varianz

$$v = \frac{|a-m|^2 + |b-m|^2}{2}.$$

Nehmen Sie im Folgenden an, daß es sich (wo nicht ausdrücklich anders angegeben) stets um Schaltungen für dreistellige Dualzahlen der Form $x = (x_2, x_1, x_0)$ handelt, und bemühen Sie sich bitte um übersichtliche Zeichnungen (z. B. durch Verwendung von Schienendarstellungen)!

(a) Beim Kramen in Ihrem alten Elektronikbastelkasten entdecken Sie fünf Volladdierer, zwei Halbaddierer und zwei Oder-Gatter. Realisieren Sie damit eine einfache Schaltung H zur Berechnung des Mittelwertes zweier sechsstelliger Dualzahlen (wobei die üblichen Rundungsregeln verwendet werden sollen, d. h. Rest < 0.5: abrunden; Rest ≥ 0.5: aufrunden)!

(6 P.)

(b) Zum bestandenen Vordiplom in Rechnerstrukturen bekommen Sie einen ganzen Sack mit Volladdierern geschenkt. Da Sie außerdem nach intensivem Suchen auf drei intakte Und-Gatter gestoßen sind, können Sie damit jetzt die Schaltung eines Quadrierwerkes für dreistellige Dualzahlen angeben, das auf dem Konzept der sog. Combinational Array Multiplier beruht. In der Vorlesung haben Sie ein derartiges Multiplizierwerk für dreistellige Dualzahlen kennengelernt (die beiden Abbildungen unten aus dem Buch von Hayes sollen zur Erinnerung dienen). Geben Sie davon ausgehend eine möglichst einfache Schaltung für ein solches Quadrierwerk Q an, und zwar in Form einer Schienendarstellung mit drei Eingabeschienen (für die dreistellige Dualzahl $x = (x_2, x_1, x_0)$) und sechs Ausgängen für das Ergebnis.

(6 P.)

(c) Als nächstes bauen Sie aus Ihren Volladdierern noch eine (möglichst einfache) Schaltung A, die auf Volladdiererebene aus vier Stufen besteht und für zwei dreistellige Dualzahlen a und b den Abstand $|a-b|$ berechnet. Hierzu sollten Sie sich u.a. überlegen, ob und wie man die Eingabezahlen in ihr Zweierkomplement überführen sollte, bevor man den Wert $a-b$ berechnet und davon dann den Betrag bildet. Beachten Sie hierbei, daß man Volladdierer auch als Inverter nutzen kann! **(6 P.)**

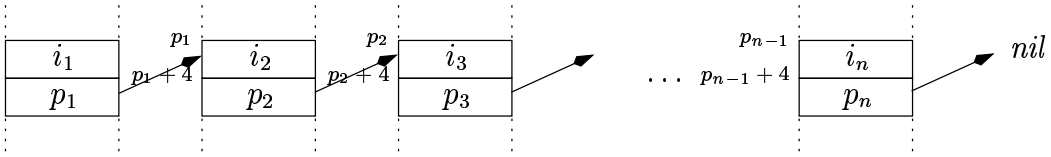
Bitte schreiben Sie deutlich!

Bitte schreiben Sie deutlich!

(d) Fügen Sie abschließend sämtliche Einzelteile zu einem Varianzwerk für dreistellige Dualzahlen zusammen. Es genügt, wenn Sie die bisher konstruierten Schaltungen für Halbier-, Quadrier- und Abstandswerke als "black boxes" H, Q und A mit den jeweiligen Ein- und Ausgängen darstellen. **(4 P.)**

Aufgabe 3: Assembler (10 Punkte)

Eine (einfach) verkettete Integer-Liste der Länge n mit Inhalt $i_1 \dots i_n$ habe die folgende Struktur:



Die i_j bzw. p_j sind dabei jeweils zwei im Speicher aufeinanderfolgende 32-bit-Worte, wobei das erste Wort den Inhalt des Listenelements, das zweite Wort aber die Adresse auf das nachfolgende Listenelement enthält. Insgesamt belegt also ein Listenelement 8 aufeinanderfolgende Bytes. Das Listenelement *nil* existiert eigentlich nicht, hat aber die Adresse `0x0` (d.h. für das letzte Listenelement n gilt $p_n =$ `0x0`).

Ihre Aufgabe besteht nun darin, in WE32100-Assembler eine (ausreichend kommentierte!!!) Prozedur zu implementieren, die das maximale Element der Liste herausfindet. Als Ergebnis soll die Funktion wahlweise die Adresse a auf das entsprechende Listenelement oder den Wert w selber zurückgeben.

Weitere Erläuterungen:

- Der Prozedur müssen also zwei Parameter übergegeben werden: die Anfangsadresse der Liste und ein Wort, das anzeigt, ob a oder w zurückgegeben werden soll. Im ersten Fall soll `0x0`, im zweiten Fall `0x1` übergeben werden.

  Folgendes Programmfragment beschreibt die Parameterübergabe:

```
Label   Programm              Kommentar
        ⋮
        PUSHW ADDRESS         Adresse auf Stack
        PUSHW &0x0            0x0 (oder 0x1) auf Stack
        CALL -8(%sp), max     Prozedur aufrufen
        ⋮
```

- Als Hilfs- bzw. lokale Variablen sollen nur Register verwendet werden. Ergreifen Sie also die bekannten Schutzmaßnahmen, um die betroffenen Registerinhalte des aufrufenden Programms zu schützen.

- Das Ergebnis soll im Argumentbereich auf dem Stack an Stelle der Anfangsadresse zurückgegeben werden.

- Auf der nächsten Seite finden Sie eine dem Buch von Oberschelp entnommene Zusammenstellung der wichtigsten Assemblerbefehle des WE32100.

Die wichtigsten Assemblerbefehle des WE32100:

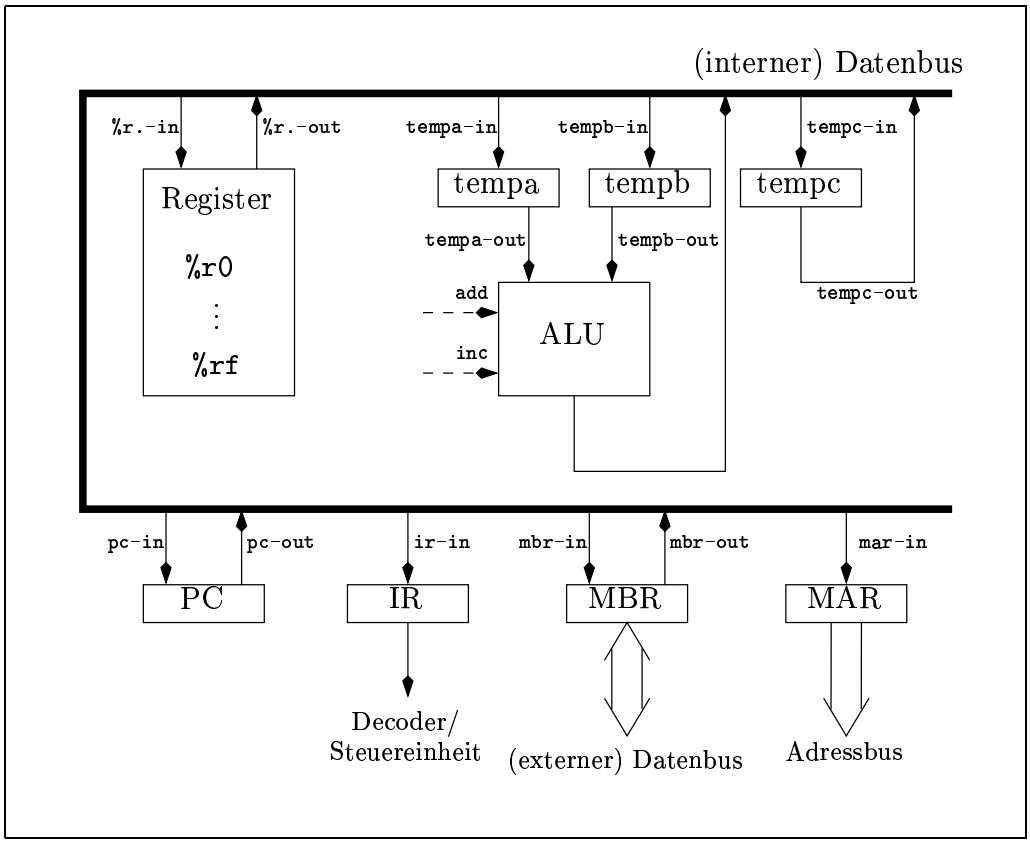
Bitte schreiben Sie deutlich!

Aufgabe 4: Mikroprogrammierung (15 Punkte)

Wir betrachten im folgenden den Befehl `MOVEW 0x3F(%r0),(%r1)`.

(a) Um welche Adressierungsart (bekannt vom WE32100) handelt es sich hier beim ersten Operanden? Erläutern Sie diese und zeigen Sie an einem typischen Beispiel, wie diese Adressierungsart bei der Implementierung höherer Programmiersprachen (wie Pascal oder C) Verwendung finden könnte! **(3 P.)**

(b) Schreiben Sie zu dem Befehl ein Mikroprogramm. Die Struktur des zugrundezulegenden Prozessors ist in der Abbildung gegeben. Es handelt sich im Wesentlichen um die aus der Vorlesung bekannte vereinfachte interne Struktur des WE32100. Wie ebenfalls bekannt, hat



jedes Register Steuerleitungen `x-in` bzw. `x-out`. Die ALU besitzt (u.a.) zwei Steuerleitungen `add` und `inc`, die die Addition von `tempa` und `tempb` bzw. das Inkrementieren von `tempa` bewirken. Der Mikrobefehl `wait for RCS` bewirkt, daß der Prozessor wartet, bis eine Speicheranfrage (`read`) beantwortet ist. Zusätzlich steht natürlich noch ein Steuersignal `write` zur Verfügung.

Die Fetchphase ist bereits vorgegeben. Gehen Sie davon aus, daß die Befehlsdekodierung wie auch die Entschlüsselung der Adressierungsarten bereits mit der Fetchphase abgeschlossen ist, d.h. das Mikroprogramm braucht wirklich nur die Schritte auszuführen, die zur konkreten Realisierung des Befehls nötig sind. Gehen Sie weiterhin davon aus, daß der Operand `0x3f` sich an der Stelle im Speicher befindet, auf den der PC nach der Fetch-Phase zeigt.

Bitte schreiben Sie deutlich!

Schritt	Aktion (Steuersignale)
1	`PC-out; MAR-in; read; tempa-in;`
2	`tempa-out; inc; tempc-in;`
3	`tempc-out; PC-in; wait-for-RCS;`
4	MBR-out; IR-in;
⋮	

Ihre Aufgabe besteht nun darin, das Mikroprogramm ab der 5. Zeile zu vervollständigen. Beachten Sie, daß hierzu etwa 11 zusätzliche Mikroprogrammschritte notwendig sind !

(12 P.)

Aufgabe 5: Knightrider-Zähler (20 Punkte)

Gegeben sei ein Haus, bestehend aus Erdgeschoß, sieben Stockwerken und – zugegeben etwas altmodisch – einem (vereinfachten) Paternoster-Aufzug. Ein derartiger Lift besteht aus Kabinen, die – im Erdgeschoß startend – der Reihe nach alle Stockwerke bis zum siebten anfahren, dort kehrtmachen und wiederum der Reihe nach in allen Stockwerken anhalten, bis sie wieder im Erdgeschoß sind und die Tour von neuem beginnen. Im Rahmen einer Modernisierungskampagne sollen nun die Kabinen mit einer Stockwerksanzeige versehen werden.
Hierzu bezeichne die dreistellige Dualzahl $x = (x_2x_1x_0)_2$ die Nummer des aktuell besuchten Stockwerks (Erdgeschoß = 0, oberste Etage = 7), ferner sei y eine einstellige Dualzahl, die angibt, ob der Lift gerade nach oben ($y = 0$) oder nach unten ($y = 1$) fährt.

(a) Sei $F : B^4 \to B^3$ mit $F(x, y) = (f_2(x, y), f_1(x, y), f_0(x, y))$ die Schaltfunktion, die bei Eingabe des aktuellen Stockwerks x und der aktuellen Fahrtrichtung y die Nummer des als nächstes angefahrenen Stockwerks berechnet. Geben Sie die zu F gehörende Funktionstabelle an! **(3 P.)**

(b) Vereinfachen Sie die drei Booleschen Funktionen f_2, f_1 und f_0 mit Hilfe von Karnaugh-Diagrammen so weit wie möglich! Nutzen Sie dabei evtl. auftretende "Don't Cares" möglichst geschickt aus! **(6 P.)**

(c) In der Vorlesung haben Sie das Konzept der PLAs kennengelernt. Skizzieren Sie kurz, welche Vorteile es hat, Schaltungen mit Hilfe von PLAs zu implementieren. Geben Sie dann ein punktorientiertes PLA für die Berechnung von f_2, f_1 und f_0 an, bei dem Und- und Oder-Ebene strikt getrennt sind, und zeigen Sie abschließend, wie man es durch Faltung noch etwas verkleinern kann! **(6 P.)**

(d) Wie sieht eine getaktete Schaltung (der sog. "Knightrider-Zähler") aus, die in jedem Takt das von unserem Paternoster als nächstes angefahrene Stockwerk $x = (x_2x_1x_0)_2$ berechnet, also in der Reihenfolge {0,1,2,3,4,5,6,7,6,5,4,3,2,1,0,1,2,3,...} auf- und abzählt? Benutzen Sie hierzu das Schaltnetz aus Teil (c) als "black box" Z mit vier Ein- und drei Ausgängen (siehe Abbildung links) und verwenden Sie außerdem vier Speicherelemente (für x_2, x_1, x_0 und y) sowie drei Und- bzw. Oder-Gatter und beliebig viele Inverter! Beachten Sie, daß die Gatter beliebig hohes Fan-In aufweisen können. **(5 P.)**

Viel Erfolg!!!

Lehr– und Forschungsgebiet Informatik
Leistungsbewertung und Verteilte Systeme
Prof. Dr. Ir. B. Haverkort

RWTH Aachen
Ahornstraße 55
52074 Aachen

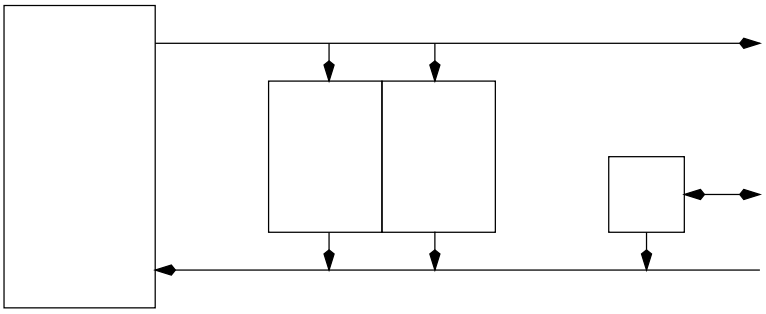
Diplomvorprüfung im Fach „Rechnerstrukturen“ (Wintersemester 97/98)

Aufgabe 1: Zum Aufwärmen (20 Punkte)

Beantworten Sie die folgenden Fragen kurz. Notieren Sie die Antworten bitte auf diesem Blatt in dem dafür freigelassenen Platz.

(a) Stellen Sie den 3-Bit-Gray-Code dar. **(2 P.)**

(b) Welcher Begriff fällt ihnen zu dem folgenden Blockbild ein? Beschriften Sie die Kästchen und die wichtigsten Pfeile. **(3 P.)**



(c) Aus wievielen und welchen Phasen besteht die Ausführung eines Assembler-Befehls? **(3 P.)**

Bitte schreiben Sie deutlich!

(d) Konstruieren Sie aus logischen Gattern Ihrer Wahl einen möglichst einfachen Halbaddierer. **(2 P.)**

(e) Konstruieren Sie nun einen Volladdierer. Verwenden Sie dazu die Halbaddierer aus Teil (d). **(2 P.)**

(f) Stellen Sie die Zahlen $(251)_{10}$ und $(-17)_{10}$ im Siebener-Komplement zur Basis 8 mit 3 Stellen dar. **(3 P.)**

(g) Wie verwandelt man das Siebener-Komplement ins Achter-Komplement? **(1 P.)**

(h) Welcher Unterschied besteht zwischen einem PAL und einem PLA? **(1 P.)**

(i) Stellen Sie die Zahl -23.625 als normalisierte Fließkommazahl zur Basis 2 im Dualsystem dar. Die Mantisse habe dabei eine Länge von 16 Bit, der Exponent von 8 Bit. **(3 P.)**

Aufgabe 2: Karnaugh-Diagramme (15 Punkte)

Sei folgende vierstellige Boolesche Funktion f gegeben:

x_0	x_1	x_2	x_3	$f(x_0, x_1, x_2, x_3)$	x_0	x_1	x_2	x_3	$f(x_0, x_1, x_2, x_3)$
0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	1	1	1	0	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1

(a) Minimieren Sie f mit Hilfe des folgenden Karnaugh-Diagramms. **(8 P.)**

		$\leftarrow x_0x_1 \rightarrow$			
		00	01	11	10
$\uparrow$ x_2x_3 $\downarrow$	00				
	01				
	11				
	10				

(b) Implementieren Sie die (minimierte) Funktion als PLA. Benutzen Sie dazu die Abbildung 1 auf Seite 4. **(7 P.)**

Bitte schreiben Sie deutlich!

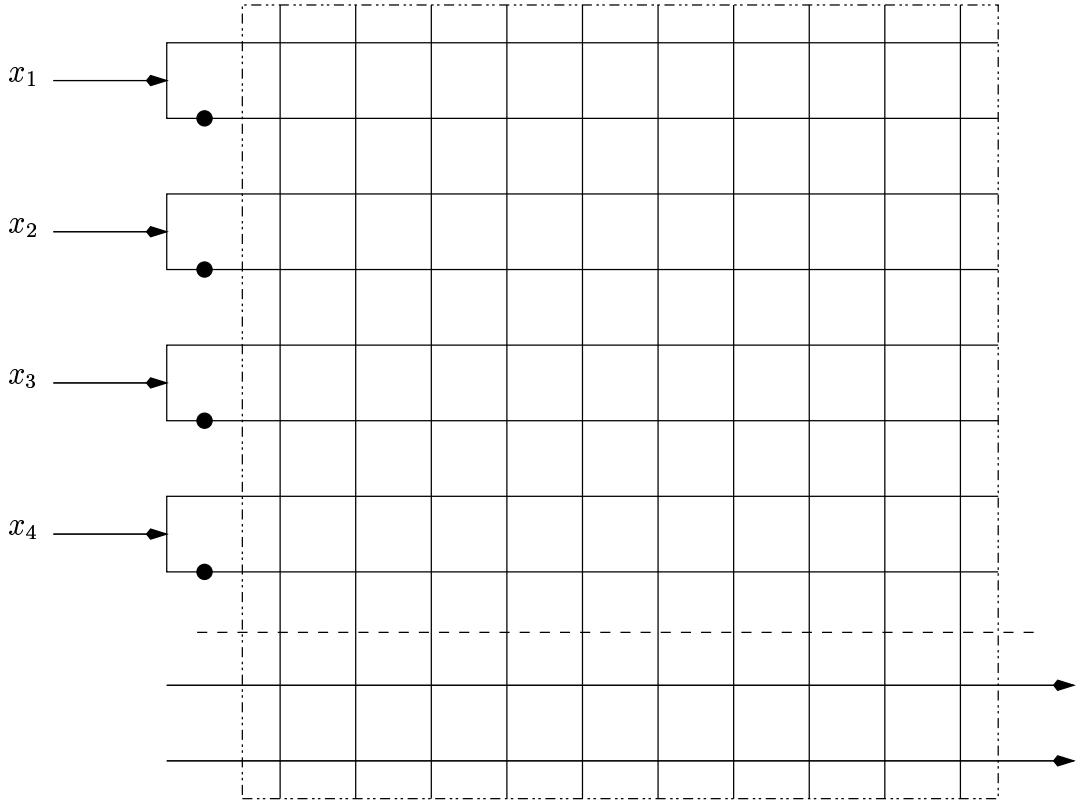


Abbildung 1: Lösungsskizze zu Aufgabe 2

Bitte schreiben Sie deutlich!

Aufgabe 3: Boolesche Algebra (27 Punkte)

Bitte notieren Sie die Lösungen dieser Aufgabe auf einem separaten Blatt (kariert).

Gegeben sei folgende dreistellige Boolesche Funktion f:

x_0	x_1	x_2	$f(x_0, x_1, x_2)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

(a) Definieren Sie ausführlich, was man unter einer *zweielementigen Booleschen Algebra* versteht. Geben Sie hierbei insbesondere auch alle für die Definition relevanten ein- bzw. zweistelligen Booleschen Funktionen an. **(5 P.)**

(b) Geben Sie die Minterme zu den einschlägigen Indizes von f an und berechnen Sie

(i) die DNF,

(ii) die KNF und

(iii) die komplementfreie Ringsummenentwicklung nach Reed-Muller.

Sind alle drei Darstellungen eindeutig? **(13 P.)**

(c) Es wird nun folgende Notation eingeführt: Sei $g : B^n \longrightarrow B$ eine beliebige n-stellige Boolesche Funktion und $a \in B$. Dann ist $g(x_i/a)$ definiert durch

$$g(x_i/a) := g(x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_n),$$

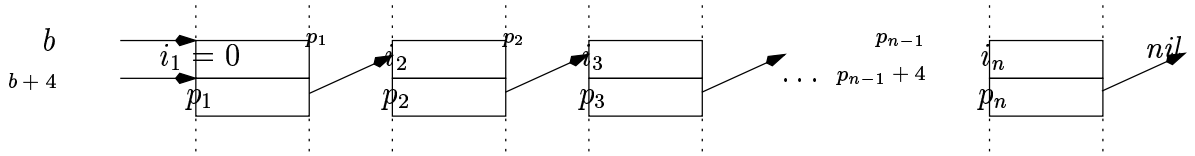
d.h. $g(x_i/a)$ entsteht aus g durch *feste* Belegung der Variablen x_i mit dem Wert $a \in B$.

(i) Geben Sie für die eingangs definierte Boolesche Funktion f drei Paare (i, a) mit $i \in \{0, 1, 2\}$ und $a \in \{0, 1\}$ an, so daß $f(x_i/a)$ funktional vollständig ist, und begründen Sie kurz Ihre Wahl.

(ii) Wie kann man durch Änderung *einer einzigen Zahl* in der letzten Spalte der Funktionstabelle von f erreichen, daß die entstehende Funktion für *alle sechs* Möglichkeiten von (i, a) funktional vollständig ist? **(9 P.)**

Aufgabe 4: Assembler (25 Punkte)

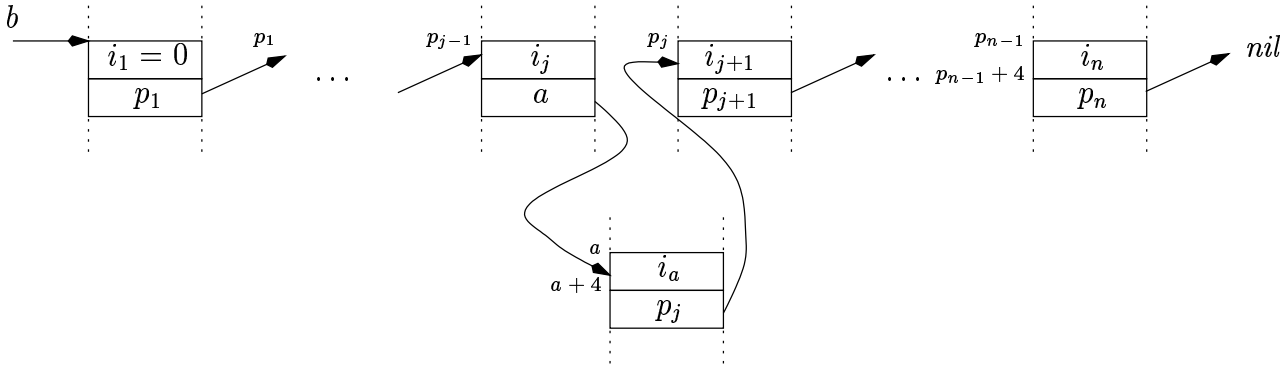
Bitte notieren Sie die Lösungen dieser Aufgabe auf einem separaten Blatt (kariert).
Eine (einfach) verkettete Integer-Liste der Länge n mit Inhalt $i_1 \dots i_n$ habe die folgende Struktur:



Die i_j bzw. p_j sind dabei jeweils zwei im Speicher aufeinanderfolgende 32-Bit-Worte, wobei das erste Wort den Inhalt des Listenelements, das zweite Wort aber die Adresse auf das nachfolgende Listenelement enthält. Insgesamt belegt also ein Listenelement 8 aufeinanderfolgende Bytes. Das Listenelement *nil* existiert eigentlich nicht, hat aber die Adresse `0x0` (d.h. für das letzte Listenelement n gilt $p_n =$ `0x0`). Die i_j-Werte sollen immer als positive Zahl interpretiert werden.

An der Adresse a im Speicher befindet sich ein Listenelement (also ein Speicherbereich von 8 Bytes der oben beschriebenen Form), das in eine aufsteigend geordnete Liste mit Anfangsadresse b eingeordnet werden soll, und zwar so, daß die entstehende Liste ebenfalls wieder aufsteigend sortiert ist. Das erste Element der Liste b soll immer 0 enthalten. Der Inhalt von a soll immer grösser als 0 sein (diese Annahmen dienen zur Vereinfachung der Aufgabe).
Schreiben Sie in WE32100-Assembler eine (ausreichend kommentierte!!!) Prozedur, die als Parameter die Anfangsadresse des einzufügenden Listenelements und die Anfangsadresse der Liste erwartet. Die Prozedur soll kein Ergebnis zurückliefern.

Beispiel: Nehmen wir an, für den Inhalt i_a von a gilt $i_j \leq i_a \leq i_{j+1}$. Dann sollte die Liste nach dem Einfügen von a folgende Gestalt haben:



Weitere Erläuterungen:

- Folgendes Programmfragment beschreibt die Parameterübergabe:

```
Label   Programm                  Kommentar
        ⋮
        PUSHW L_ADDRESS          Listenadresse auf Stack
        PUSHW E_ADDRESS          Elementadresse auf Stack
        CALL -8(%sp), einfueg    Prozedur aufrufen
        ⋮
```

- Als Hilfs- bzw. lokale Variablen sollen nur Register verwendet werden. Ergreifen Sie also die bekannten Schutzmaßnahmen, um die betroffenen Registerinhalte des aufrufenden Programms zu schützen.

Aufgabe 5: Pseudo-Zufallszahlengenerator (30 Punkte)

Bitte notieren Sie die Lösungen der Teilaufgaben (a)–(c) auf einem separaten Blatt (kariert). Für Aufgabe (d) benutzen Sie bitte die Skizze auf Seite 10.

Die Erzeugung von Pseudo-Zufallszahlen (PZZ) mittels PZZ-Generatoren (PZZG) spielt eine immer wichtigere Rolle bei vielen Anwendungen. PZZ werden häufig mit "linear congruential generators" erzeugt. Ein Sequenz von Zahlen $z^{(0)}, z^{(1)}, z^{(2)}, \ldots$ wird dann folgendermaßen erzeugt:

$$z^{(n)} := a \cdot z^{(n-1)} \text{ modulo } m,$$

wobei $a, m \in I\!N^+$. Die Zahl $z^{(0)}$ heißt "seed". Bei einer vernünftigen Wahl von $z^{(0)}$, a und m werden alle Zahlen aus $\{0, 1, \ldots, m-1\}$ in einer quasi-zufälligen Reihenfolge erzeugt.
Als Beispiel wählen wir $z^{(0)} = 1$, $a = 3$ und $m = 32$. Wir bekommen dann : 1, 3, 9, 27, 19, 26,....
In dieser Aufgabe werden wir Hardware für PZZG entwickeln.

(a) Betrachten wir den PZZG:

$$z^{(n)} := 3 \cdot z^{(n-1)} \text{ modulo } 32.$$

Die Zahlen $z^{(i)}$ werden mit 6 Bits im Zweier-Komplement kodiert. Bauen Sie mit sechs 1-Bit Registern, vier Volladdierern und einem Halbaddierer einen PZZG, der bei jedem Uhrtakt den nächsten Wert $z^{(i)}$ liefert. **(8 P.)**

(b) Warum ist es günstig, wenn m eine Zweierpotenz ist? **(2 P.)**

(c) Wir betrachten im folgenden nur noch den Fall $m \neq 2^k$ und beschäftigen uns vor allem mit der Modulo-Operation. Berechnen Sie iterativ im Zweier-Komplement der Dualdarstellung:

$$z := (21)_{10} \text{ modulo } 6_{10}.$$

Benutzen Sie dabei folgenden Algorithmus:

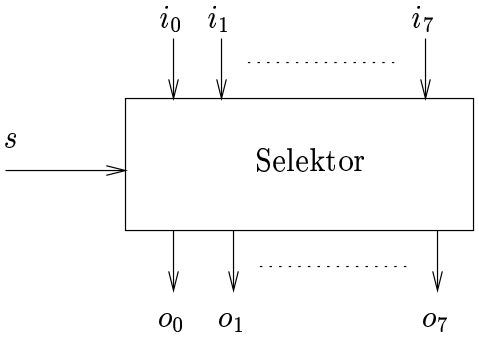
```
function modulo ( x, y:  integer):  integer;
begin
     if x < 0 then modulo := x + y
     else modulo := modulo(x-y, y);
end;
```

Nicht das Ergebnis (3) ist wichtig, sondern die Berechnung selber! **(4 P.)**

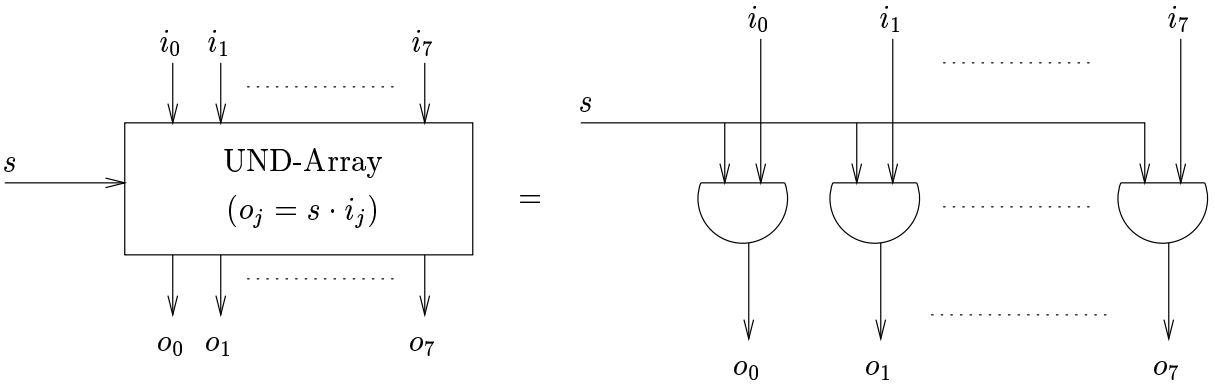
(d) Bauen Sie jetzt ein Schaltwerk zur Berechnung von $z := x$ modulo y, wobei Sie annehmen dürfen, daß $x, y > 0$; x und y sind mit 8 Bits im Zweier-Komplement kodiert. Als Bauelemente stehen zur Verfügung:

- Zwei Eingaberegister für x und y (jeweils 8 Bit breit);
- Ein Ausgaberegister für z (8 Bit breit);

- Ein 1-Bit-Register für Start/Stop-Zwecke („Ready-Register“, siehe unten);
- Ein 8-Bit breites Addierwerk;
- Ein 8-Bit breiter Selektor mit $o_j = (s \cdot i_j) + (\overline{s} \cdot \overline{i}_j)$: Für $s = 1$ gilt also $o_j = i_j$; mit $s = 0$ gilt $o_j = \overline{i}_j$, jeweils für $\forall j \in \{0, \ldots, 7\}$).



- Zwei 8-Bit breite Und-Arrays:



- beliebig viele Inverter, UND- und ODER-Gatter

Bei der Initialisierung sollten die Zahlen x und y in die beiden Eingaberegistern geschrieben und das Ready-Register auf 0 gesetzt werden. Wenn die Berechnung fertig ist, steht das Ergebnis im Ausgabe-Register. Das Ready-Register soll dann den Wert 1 haben.

Ergänzen Sie für Ihre Lösung die Abbildung 2 auf Seite 10. **(16 P.)**

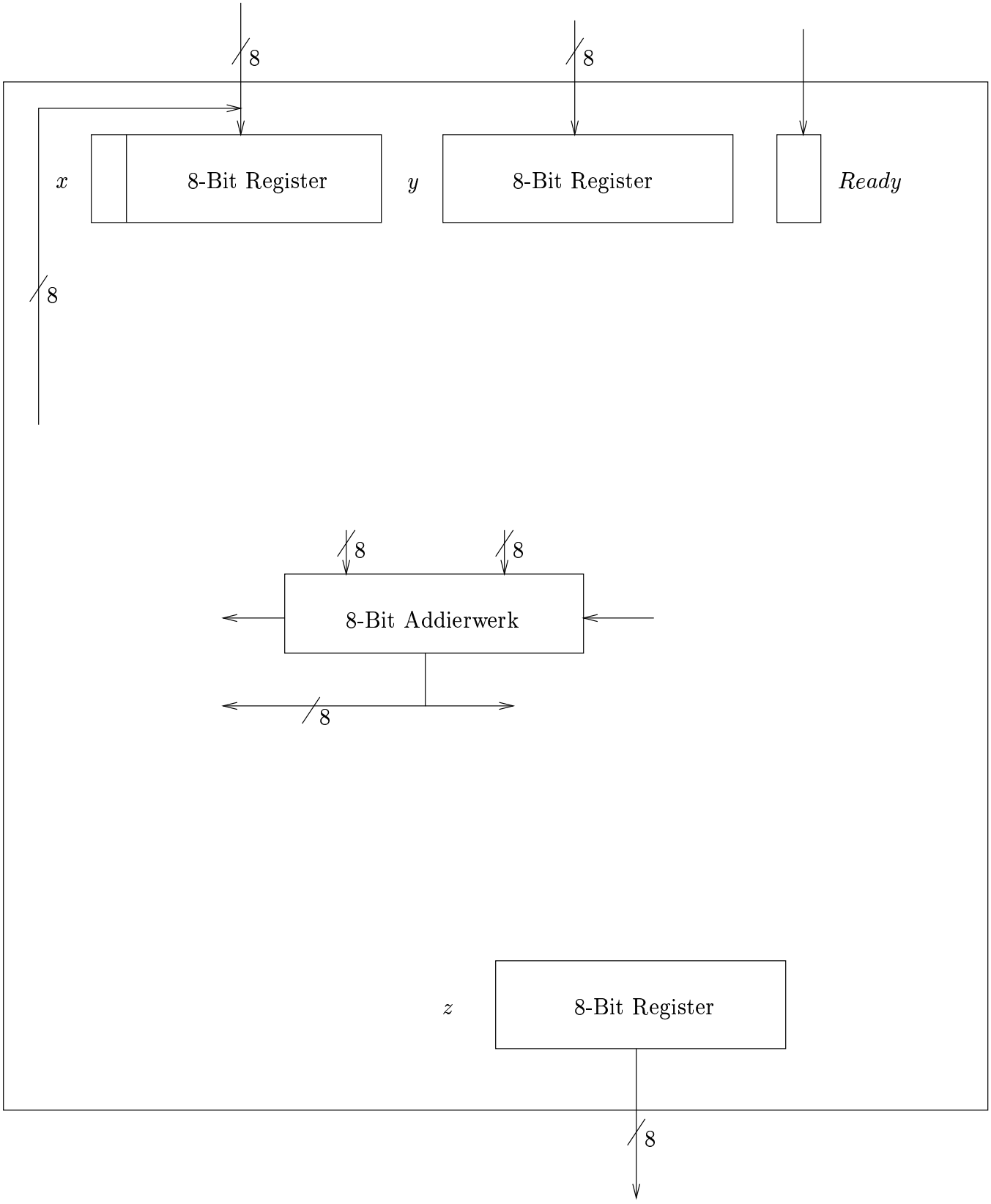


Abbildung 2: Lösungsskizze zu Aufgabe 5

RWTH AACHEN
Lehrstuhl für Informatik I
Prof. J. Hromkovič

Ahornstr. 55
52056 Aachen

Testklausur zur Vorlesung „Rechnerstrukturen“

10. August 1998

Aufgabe 1: (10 Punkte)

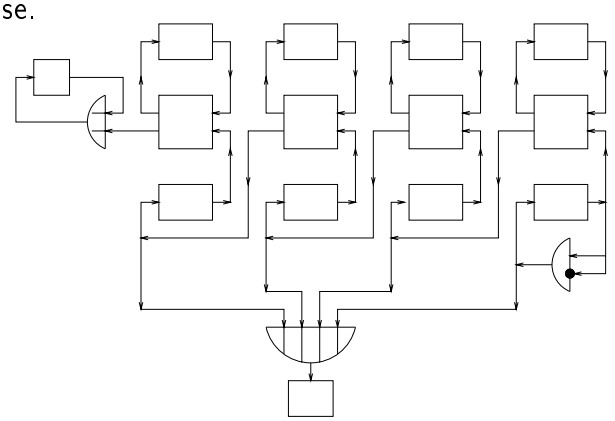
Beweisen Sie, daß die folgenden Mengen funktional vollständig sind:

a: $\{\vee, \wedge, \neg\}$, **b:** $\{\vee, \neg\}$, **c:** $\{\oplus, \wedge\}$, **d:** $\{\uparrow\}$, **e:** $\{\downarrow\}$.

Verwenden Sie nur den Darstellungssatz für Boolesche Funktionen, d.h. Zwischenschritte sind zu beweisen.

Aufgabe 2: (10 Punkte)

a: Vervollständigen Sie das folgende Schema eines 4-Bit-von Neumann-Addierwerks und beschreiben Sie dessen Funktionsweise.



b: Wieviele Takte braucht ein n-Bit-von Neumann-Addierwerk im schlechtesten Fall? Begründen sie Ihre Antwort.

c: Beschreiben Sie ein n-Bit-von Neumann-Addierwerk als Mealy-Automat für ein beliebiges $n \in \mathbb{N}$.

Aufgabe 3: (10 Punkte)

a: Definieren Sie die Aufgabe eines d-Demultiplexers ($d-$DeMUX) für ein beliebiges $d \in \mathbb{N}$.

b: Entwerfen Sie einen 3-DeMUX.

c: Benutzen Sie Ihren 3-DeMUX als Baustein für ein Schaltnetz zur Berechnung der folgenden Funktion:

$$f(y_1, y_2, y_3) = \overline{y}_1 y_2 y_3 \vee y_1 \overline{y}_2 y_3 \vee \overline{y}_1 \overline{y}_2 \overline{y}_3.$$

Aufgabe 4: (10 Punkte)

Schreiben Sie ein gut kommentiertes Assemblerprogramm im orthogonalen Assembler zur Lösung der folgenden Aufgabe:

Im Speicher steht eine Matrix M der Größe $n \times n$ mit Einträgen aus $\mathbb{N}$. Bestimmen Sie den Index der lexikographisch kleinsten Zeile. D.h. es ist $x \in \{1, ..., n\}$ zu bestimmen mit folgender Eigenschaft:

$\forall i \in \{1, ..., n\} : \exists k \in \{1, ..., n+1\}$ mit:

$$\forall j \in \{1, ..., k-1\} : m_{x,j} = m_{i,j} \quad \text{und} \quad m_{x,k} < m_{i,k} \vee k = n+1.$$

Die Speicherbelegung ist wie folgt gegeben:

$$n = \delta(M_0) \quad \text{und} \quad m_{i,j} = \delta(M_{j+n\cdot(i-1)}) \text{ für } 1 \leq i, j \leq n.$$

Der Index der lexikographisch kleinsten Zeile soll am Ende im Register 0 liegen.

RWTH AACHEN
Lehrstuhl für Informatik I
Prof. J. Hromkovič

Ahornstr. 55
52056 Aachen

2. Klausur zur Diplom–Vorprüfung (DPO 89) in „Rechnerstrukturen“

23. März 1999

Aufgabe 1: (17 Punkte)

(a) Gegeben sei die folgende Boolesche Funktion f:

x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$f(x_1, \ldots, x_4)$	1	1	0	1	0	0	0	0	1	1	0	1	1	1	0	1

(i) Wenden Sie das Karnaugh-Verfahren auf die DNF von f an, um eine möglichst kurze disjunktive Form von f zu erhalten.

(ii) Zeigen Sie, daß die in (i) gefundene Darstellung $\mathrm{DF}_{\min}$ in folgendem Sinne nicht optimal ist:
Es existiert eine Boolesche Formel, die f beschreibt und weniger zweistellige Operatoren benötigt als $\mathrm{DF}_{\min}$.

(b) Gegeben sei die folgende Boolesche Funktion f:

x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$f(x_1, \ldots, x_4)$	0	0	1	1	0	1	0	1	1	0	1	0	1	1	0	0

(i) Bestimmen Sie alle Primimplikanten von f.

(ii) Bestimmen Sie zwei unterschiedliche disjunktive Formen (DF) von f, die beide das Ergebnis des Quine-McCluskey-Verfahrens sein können.

Aufgabe 2: (16 Punkte)

(a) Geben Sie das Schaltbild eines 4-Bit-Serien-Addierwerks an und beschreiben Sie dessen Funktionsweise.

(b) Beschreiben Sie das 4-Bit-Serien-Addierwerk durch einen Mealy-Automaten.

Aufgabe 3: (17 Punkte)

Schreiben sie ein gut kommentiertes Assembler-Programm, welches das folgende Programm — den Euklidischen Algorithmus zur Bestimmung des größten gemeinsamen Teilers — umsetzt. Sie dürfen dabei jeden der in der Vorlesung vorgestellten Assembler benutzen.

```
Funktion ggt(a,b);

Eingabe: a,b
Ausgabe: ggt(a,b)

repeat
r := a mod b;
f := a div b;
a := b;
b := r;
until r = 0;
ggt(a,b) := a;
```

Beachten Sie:

- Sie dürfen nicht die Operation "mod" direkt in Assembler ausführen.
- Beim Teilen zweier Zahlen erhalten Sie nur den ganzzahligen Anteil.
- Gehen Sie von folgender Speicherbelegung aus:

a	steht in	M(1),	
b	steht in	M(2),	
ggt(a,b)	soll in	M(0)	abgelegt werden.

- Die Inhalte von M(1) und M(2) dürfen nicht verändert werden.

Lehr– und Forschungsgebiet Informatik
Leistungsbewertung und Verteilte Systeme
RWTH Aachen
Prof. Dr. Ir. B. Haverkort

Ahornstraße 55
52074 Aachen
Tel.: 0241/80–21430
Tel.: 0241/80–21432

Scheinklausur Rechnerstrukturen (SS 1999)

Name: _____

Vorname: _____

Matrikelnummer: _____

Benötigen Sie eine Schein ? (Wenn ja, bitte ankreuzen.) ..... ☐

Bitte beachten:

- Tragen Sie die Lösungen auf den Aufgabenblättern ein. Sollte der der Platz auf der Vorderseite nicht ausreichen, benuzten Sie die Rückseite.
- Die Klausurergebnisse hängen ab Donnerstag, 1. Juli aus und können auch über das WWW abgefragt werden.
- Die Klausuren können am LuFG abgeholt werden. Es gibt aber *keine* Fragestunde zur Klausur. Lediglich eine Musterlösung werden wir zur Verfügung stellen.
- Um einen Schein zu bekommen, benötigen Sie **55 von 90 Punkten**.

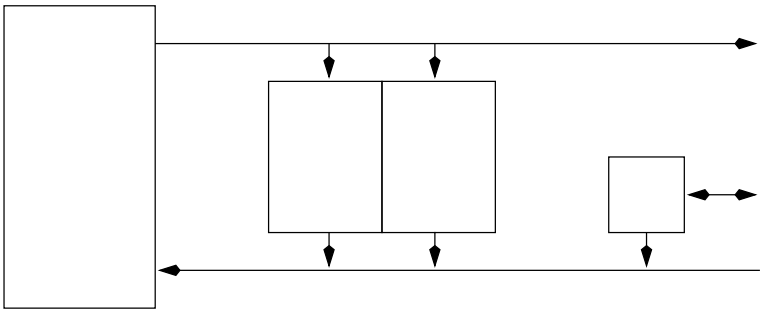
Nr.	Punkte	erreichte Punktzahl	Korrigiert von
1	22		
2	26		
3	16		
4	12		
5	14		
Gesamt	90		

Aufgabe 1: Zum Aufwärmen (22 Punkte)

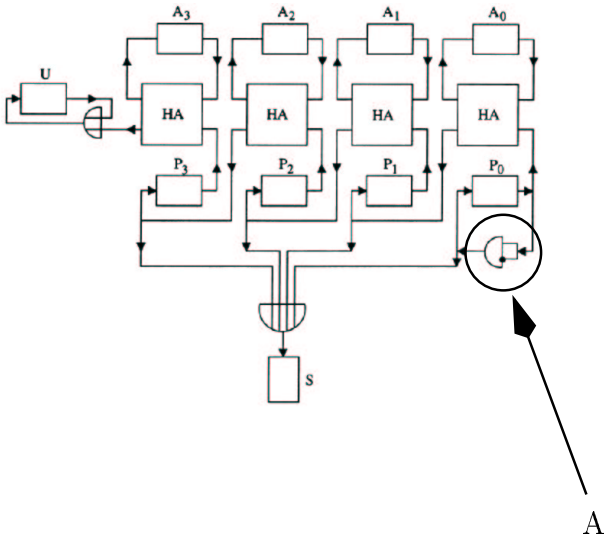
(a) Konstruieren Sie aus logischen Gattern Ihrer Wahl einen möglichst einfachen Halbaddierer. **(2 P.)**

(b) Konstruieren Sie nun einen Volladdierer. Verwenden Sie dazu die Halbaddierer aus Teil (a). **(2 P.)**

(c) Welches wichtige Konzept wird durch folgendes Blockbild (unvollständig) dargestellt? Beschriften Sie die Kästchen und die wichtigsten Pfeile. **(4 P.)**



(d) Beantworten Sie die folgenden Fragen zu dem abgebildeten von-Neuman-Addierwerk:



(i) Wozu dient das mit A gekennzeichnete Gatter? **(4 P.)**

(ii) Was zeigt das Delay S an? **(4 P.)**

(iii) Was ist der Vorteil eines von-Neumann-Addierers im Vergleich zu einem herkömlichen Parallel-Addierwerk? **(6 P.)**

Aufgabe 2: Boolesche Algebra (26 Punkte)

Gegeben sei folgende vierstellige Boolesche Funktion F:

x_0	x_1	x_2	x_3	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0

x_0	x_1	x_2	x_3	F
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(a) Wie sind die Maxterme sowie die Maxtermdarstellung einer Booleschen Funktion f definiert? **(6 P.)**

(b) Geben Sie die einschlägigen Minterme für obige Funktion F an! Ist im Falle von F eher die DNF oder die KNF vorzuziehen (Begründung)? **(4 P.)**

(c) Minimieren Sie F mit Hilfe des folgenden Karnaugh-Diagramms. Geben Sie die minimierte Funktion dann explizit an. **(8 P.)**

$\uparrow$ x_2x_3 $\downarrow$	$\leftarrow x_0x_1 \rightarrow$ 00	01	11	10
00				
01				
11				
10				

(d) Beweisen Sie, daß die Boole'sche Funktion f funktional vollständig ist, wobei f mit Hilfe der oben definierten Funktion F wie folgt definiert ist:

$$f(y_0, y_1, y_2) \quad := \quad \begin{cases} F(y_0, 1, y_2, y_2) & \text{falls } y_0 = 0 \\ F(y_0, 1, y_1, y_2) & \text{falls } y_0 = 1 \end{cases}$$

(8 P.)

Aufgabe 3: Zahlensysteme und Zahlendarstellung (16 Punkte)

(a) Können Sie die folgenden Zahlen ins Dezimalsystem konvertieren? Wenn ja, dann tun Sie es. Wenn nein, warum nicht?

(i) $(38)_9$ **(2 P.)**

(ii) $(10110)_{21}$ **(2 P.)**

(iii) $(25537)_7$ **(2 P.)**

(iv) $(4631)_7$ **(2 P.)**

(b) Stellen Sie die Zahlen $(129)_{10}$ und $(-13)_{10}$ im Siebener-Komplement zur Basis 8 mit 3 Stellen dar. **(4 P.)**

(c) Stellen Sie die Zahl -13.125 als normalisierte Fließkommazahl zur Basis 2 im Dualsystem dar. Die Mantisse habe dabei eine Länge von 23 Bit, der Exponent von 8 Bit. **(4 P.)**

Aufgabe 4: Ringzähler (12 Punkte)

Konstruieren Sie aus Ihnen genehmen Bauteilen einen Ringzähler, der erst die Primzahlen zwischen 0 und 15 aufzählt und dann mit den zusammengesetzten Zahlen incl. 0 und 1 fortfährt (also $\ldots, 2, 3, 5, 7, 11, 13, 0, 1, 4, 6, 8, 9, 10, 12, 14, 15, 2, 3 \ldots$).
Gehen sie folgendermassen vor:

(a) Definieren Sie die benötigten Schaltfunktionen.

(b) Minimieren Sie diese Funktionen m.H. von Karnaugh-Diagrammen

(c) Entwerfen Sie m.H. der minimierten Funktionen ein Schaltwerk für den Ringzähler.

Aufgabe 5: Assembler (14 Punkte)

Sei folgendes WE32100 Assembler-Programm gegeben:

```
start:  pushw %r8                   d:      save %r7
        pushw %r7                           movw &0x0, %r0
        call -8(%sp), d                     movw (%ap), %r7
stop:   ...                                 movw 4(%ap), %r8
                                    loop:   subw %r8, %r7
                                            blb done          # springe zu done,
                                                               wenn %r7 < %r8
                                            inc %r0
                                            jmp loop
                                    done:   restore %r7
                                            ret
```

Beim Programmstart sei das Register `%r7` mit `0x43`, das Register `%r8` mit `0xde` geladen. Der Programmlauf beginne bei `start`.

(a) Was berechnet die Prozedur `d`? **(8 P.)**

(b) Welchen Wert hat das Register `%r0`, wenn der Programmzähler `stop` erreicht? **(2 P.)**

(c) Der Assembler-Befehl `save` reserviert auf dem Stack Platz für alle Register, obwohl nicht immer alle dann auch tatsächlich dort gesichert werden. Geben sie dafür eine vernünftige Erklärung. **(4 P.)**

Vordiplomsprüfung Informatik II
Wintersemester 1999/2000

Rechnerstrukturen

Matrikelnummer: _____

Name: _____

Vorname: _____

- Die mit **(***)** gekennzeichneten Aufgaben gehen leicht über das hinaus, was in der Vorlesung bzw. den Übungen behandelt wurde. Sind Sie sich nicht sicher, daß Sie schnell eine Lösung für diese Aufgaben finden, sollten Sie diese Aufgaben bis zum Schluß aufheben.
- Am Ende der Klausur finden Sie drei Blätter, die Sie zusätzlich für Ihre Lösungen verwenden können.

Nr.	Punkte	erreichte Punktzahl	Korrigiert von
1	25		
2	11		
3	14		
4	11		
5	20		
6	19		
Gesamt	100		

Aufgabe 1: Zum Aufwärmen (20 Punkte)

(a) Wie unterscheiden sich RISC-Rechner von CISC-Rechner? **(2 P.)**

(b) In Von-Neumann-Rechnern haben wir das Zwei-Phasen-Konzept der Befehlsverarbeitung kennengelernt. Welches sind diese zwei Phasen und was wird darin jeweils gemacht? **(2 P.)**

(c) Beschreiben Sie die Klassifikation von Flynn. Insbesondere erklären Sie die Abkürzungen SISD, SIMD und MIMD. **(4 P.)**

(d) Konvertieren Sie die folgenden Zahlen. Jede Ziffer steht dafür für eine Stelle.

(i) $(219)_{11}$ ins Dezimalsystem; **(1 P.)**

(ii) $(338)_{13}$ ins Hexadezimalsystem; **(1 P.)**

Bitte schreiben Sie deutlich!

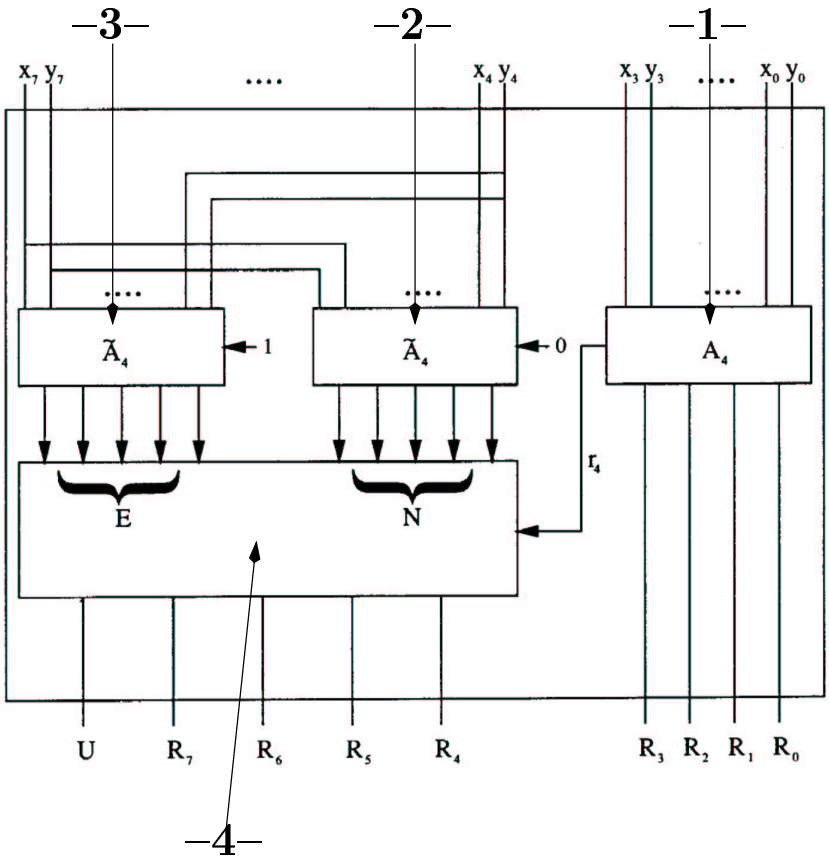
(iii) $(-2234)_{10}$ ins 10er-Komplement zur Basis 10 mit 5 Stellen; **(2 P.)**

(e) Stellen Sie 22.6875 als normalisierte Fließkommazahl zur Basis 2 im Dualsystem dar. Die Mantisse habe dabei eine Länge von 16 Bit, der Exponent von 8 Bit (Herleitung nicht nötig). **(3 P.)**

(f) Welche Ungleichheit gilt immer für die *normalisierte* Darstellung von Gleitkommazahlen $\pm m \cdot b^{\pm d}$? **(2 P.)**

Bitte schreiben Sie deutlich!

(g) Wir betrachten das Carry-Select-Addiernetz in der nachfolgenden Abbildung. Was wird in den Blöcken **–1–**, **–2–**, **–3–** und **–4–** jeweils berechnet? Was ist der Hauptvorteil dieses Addiernetzes? **(8 P.)**



Bitte schreiben Sie deutlich!

Aufgabe 2: Boole'sche Funktionen (12 Punkte)

Wir betrachten Boole'sche Funktionen der Form $f : B^n \to B$. In dieser Zusammenhang:

(a) Was ist ein einschlägiger Index? **(2 P.)**

(b) Was ist ein Minterm? **(2 P.)**

(c) Wie sieht die disjunktive Normalform (DNF) einer Boole'schen Funktion aus? **(1 P.)**

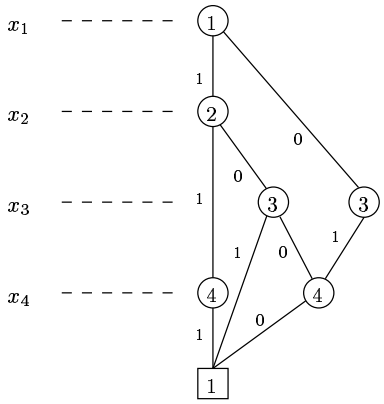
(d) Wann nennt man eine Menge $\mathcal{B} = \{f_1, \cdots, f_i\}$ von Boole'schen Funktionen funktional vollständig (in eigenen Worten)? **(2 P.)**

Bitte schreiben Sie deutlich!

(e) Ist $\{+\}$ funktional vollständig? Begründen Sie Ihre Antwort. **(4 P.)**

Aufgabe 3: BDDs (14 Punkte)

Gegeben sei das folgende BDD B:



(a) Geben sie in der Funktionstabelle an, welche Boole'sche Funktion f_{BDD} durch das BDD beschrieben wird.

	x_1	x_2	x_3	x_4	$f_{BDD}(x_1,x_2,x_3,x_4)$		x_1	x_2	x_3	x_4	$f_{BDD}(x_1,x_2,x_3,x_4)$
0	0	0	0	0		8	1	0	0	0	
1	0	0	0	1		9	1	0	0	1	
2	0	0	1	0		10	1	0	1	0	
3	0	0	1	1		11	1	0	1	1	
4	0	1	0	0		12	1	1	0	0	
5	0	1	0	1		13	1	1	0	1	
6	0	1	1	0		14	1	1	1	0	
7	0	1	1	1		15	1	1	1	1	

(4 P.)

Bitte schreiben Sie deutlich!

(b) Wie Sie sicher bemerkt haben, ist $f_{BDD}(0,1,0,0) = 0$. Wir definieren nun die Boole'sche Funktion f' so, daß $f'(x_1,x_2,x_3,x_4) = f_{BDD}(x_1,x_2,x_3,x_4)$ für alle $(x_1,x_2,x_3,x_4) \neq (0,1,0,0)$ und $f'(0,1,0,0) = 1$. Modifizieren Sie das oben angegebene BDD B so, daß es f' repräsentiert und weiterhin minimal ist. Skizzieren Sie Ihre Lösung. **(4 P.)**

(c) **(***)** Auf welche einfache Art läßt sich ein BDD wie z.B. B so modifizieren, daß es die inverse Funktion $\overline{f_{BDD}}$ repräsentiert? **(4 P.)**

(d) **(***)** Modifizieren Sie das oben gegebene BDD B nach ihrer Methode, so daß es $\overline{f_{BDD}}$ repräsentiert. Skizzieren Sie Ihre Lösung. **(2 P.)**

Bitte schreiben Sie deutlich!

Aufgabe 4: Ringzähler (14 Punkte)

Entwerfen Sie einen 3-bit binären Ringzähler, der folgenden Zahlenzyklus generiert: 0–7–1–6–2–5–3–4–0–7–1–6– usw.

(a) Vervollständigen Sie dazu die folgende Funktionstabelle: **(2 P.)**

Eingaben				*Ausgaben*			
x	x_2	x_1	x_0	y	y_2	y_1	y_0
0	0	0	0				
1	0	0	1				
2	0	1	0				
3	0	1	1				
4	1	0	0				
5	1	0	1				
6	1	1	0				
7	1	1	1				

(b) Minimieren Sie die 3 benötigten Schaltnetze mit dem Verfahren von Karnaugh. Tragen Sie Ihre Lösung in die unten angegebene Diagramme ein! Geben Sie auch explizit die minimierten Funktionen an! **(6 P.)**

y_0 :

	$\leftarrow x_1x_0 \rightarrow$			
	00	01	11	10
$\uparrow$ 0				
x_2 1				
$\downarrow$				

y_1 :

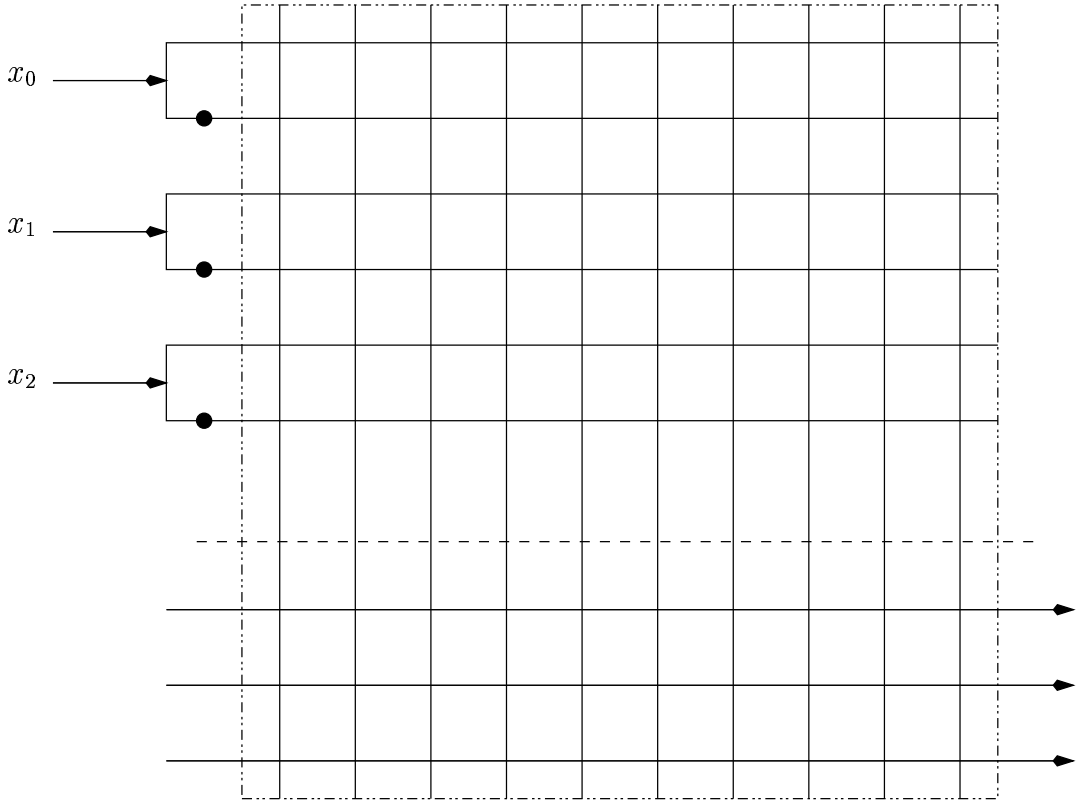
	$\leftarrow x_1x_0 \rightarrow$			
	00	01	11	10
$\uparrow$ 0				
x_2 1				
$\downarrow$				

y_2:

	$\leftarrow x_1x_0 \rightarrow$			
	00	01	11	10
$\uparrow$ 0				
x_2 1				
$\downarrow$				

(c) Implementieren Sie diesen Ringzähler mit Hilfe eines PLAs und Delays. Vervollständigen Sie dazu die Abbildung auf Seite 9 oben: **(3 P.)**

Bitte schreiben Sie deutlich!



Aufgabe 5: Assembler (20 Punkte)

(a) Wann nennen wir einen Assembler-Befehlsatz orthogonal? **(2 P.)**

(b) Der größte gemeinsame Teiler von zwei positiven natürlichen Zahlen x und y, notiert als $ggt(x, y)$, kann durch folgende rekursive Definition charakterisiert werden (wir nehmen an, daß $x, y > 0$):

```
ggt(x, y) := if (x = y) then x
             else if (x < y) then ggt(x, y-x)
                  else  ggt(x-y, y);
```

Der $ggt(x, y)$ kann sehr einfach iterativ berechnet werden. Bitte komplettieren Sie folgenden Programmabschnitt so, daß nach Ablauf der `while`-Schleife x und y den Wert des ggt's haben:

```
while (        )
do
   if (        ) then
   else
end;
```

(4 P.)

(c) Auf Seite 12 ist der Rahmen einer Assembler-Prozedur gegeben. Wofür dienen die `SAVE` und `RESTORE` Anweisungen, die dort vorgegeben wurden? **(2 P.)**

(d) Wo wird die Return-Adresse zum aufrufenden Programm gespeichert? **(2 P.)**

(e) Implementieren Sie jetzt eine WE32000 Assemblerprozedur, die iterativ den Wert $ggt(x, y)$ berechnet, wobei x in %r0 und y in %r1 abgelegt ist. Das Ergebnis soll in eine Speicherzelle geschrieben werden, deren Adresse in %r2 steht.

Benutzen Sie für diese Aufgabe das Lösungsblatt auf Seite 12 und kommentieren Sie die von Ihnen zugefügte Programmzeilen hinter dem #-Zeichen. Behalten Sie insbesondere die vorgegebenen Labels bei. Auf Seite 13 finden Sie eine Zusammenfassung der wichtigsten Assembler-Befehle.

Beachten Sie bitte, daß auf der Lösungsseite mehr Zeilen zwischen den Labels freigelassen wurden, als Sie vermutlich brauchen werden. **(10 P.)**

Lösungsblatt für Aufgabe 4 (c)**:**

```
        .globl ggt                      #   . . . . . .
ggt:    SAVE %fp                        #   . . . . . .
while:  . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
less:   . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
ready:  . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        RESTORE %fp                     #   . . . . . .
        RET                             #   . . . . . .
```

Bitte schreiben Sie deutlich!

hier Befehlsreferenz

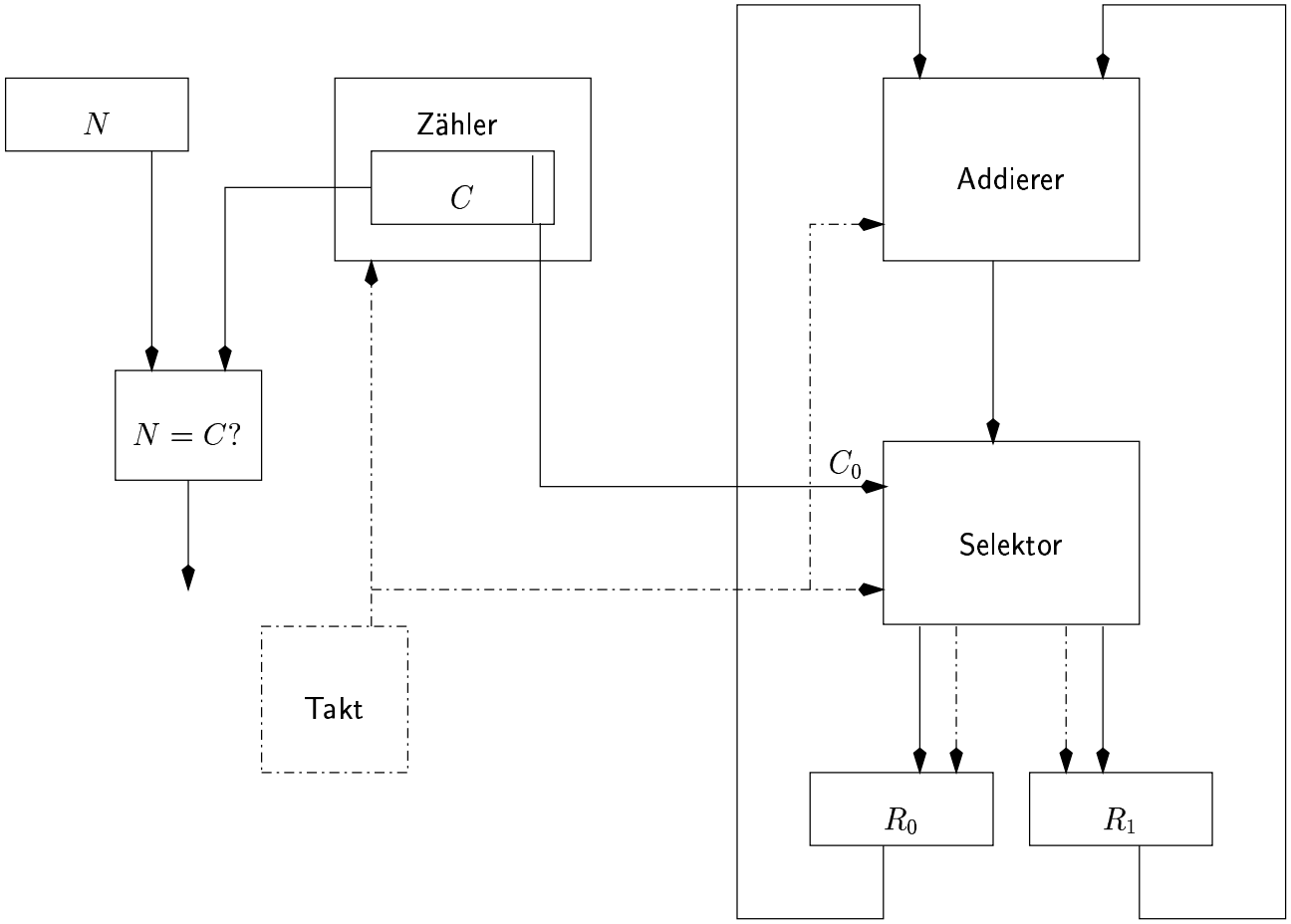
Bitte schreiben Sie deutlich!

Aufgabe 6: Schaltwerke (20 Punkte)

Die Fibonacci-Zahlen $fib(n)$ werden nach der rekursiven Formel

$$fib(n) := fib(n-1) + fib(n-2)$$

berechnet, mit $n \in \mathbb{N}$, $n \geq 2$, $fib(0) = 0$ und $fib(1) = 1$. Ein einfaches Schaltwerk für fib sieht folgendermassen aus:



Es gibt drei Register mit m bit. Diese Register sind aus den bekannten Delays zusammengesetzt. Die Register werden beim Start mit bestimmten Werten initialisiert: Register N wird auf n gesetzt und gibt an, daß $fib(n)$ berechnet werden soll. Register R_0 enthält $fib(0) = 0$, Register R_1 $fib(1) = 1$.
Weiterhin gibt es einen Addierer und Zähler C. Letzterer ist mit 0 initialisiert und zählt mit jedem Takt einen Schritt aufwärts. Die Berechnung stoppt, wenn $C = N$. Es wird also $fib(N + 2)$ berechnet.
Die Daten von Register R_0 und R_1 werden in den Addierer übernommen und addiert. **Für den Moment** nehmen wir an, daß die Addition verzögerungsfrei abläuft, d.h. das Ergebnis steht sofort, nachdem sich die Eingabeparameter geändert hat, zur Verfügung. Mit dem nächsten Taktimpuls wird das Ergebnis in R_0 oder R_1 zurückgeschrieben. Die Auswahl trifft das Schaltnetz Selektor, und zwar abhängig vom niedrigwertigsten Bit von C, C_0.
Die Takt-Leitungen sind für diese Aufgabe wichtig und deswegen als gestrichelte Linien eingezeichnet.

(a) Der Selektor dient dazu, das Additionsergebnis in das richtige Register zurückzuschreiben. Entwerfen Sie ein Schaltnetz für diesen Selektor. Diese Aufgabe wird nicht durch einen Demultiplexer gelöst, denn es ist wichtig, daß das Register, in das *nicht* geschrieben wird, seinen alten Wert behält. Ein Register behält seinen Wert, solange es keinen Taktimpuls empfängt. **(8 P.)**

Lösung Aufgabe 6 (a)

(b) **(***)** Nehmen wir nun an, daß der Addierer mehrere Taktzyklen (deren Anzahl nicht feststeht, wie z.B. beim von-Neuman-Addierer) braucht, um sein Ergebnis zu liefern. Somit muß das restliche Schaltwerk auf das Ergebnis des Addierers warten bevor fortgefahren werden kann. Der Addierer habe folgende Steuerleitungen:

- Reset-Eingang R: Wird diese Leitung für einen Taktzyklus auf 1 gesetzt, so übernimmt der Addierer die an den Eingängen anliegenden Operanden und beginnt mit der Addition.
- Ausgang S: dieser Ausgang wird für einen Taktzyklus auf 1 gesetzt, wenn der Addierer mit seiner Berechnung fertig ist.

Entwerfen Sie ein Schaltnetz SE, das die Steuerung von Addierer und dem restlichen Schaltwerk übernimmt. Sie benötigen nur ein Delay und ein UND-Gatter!

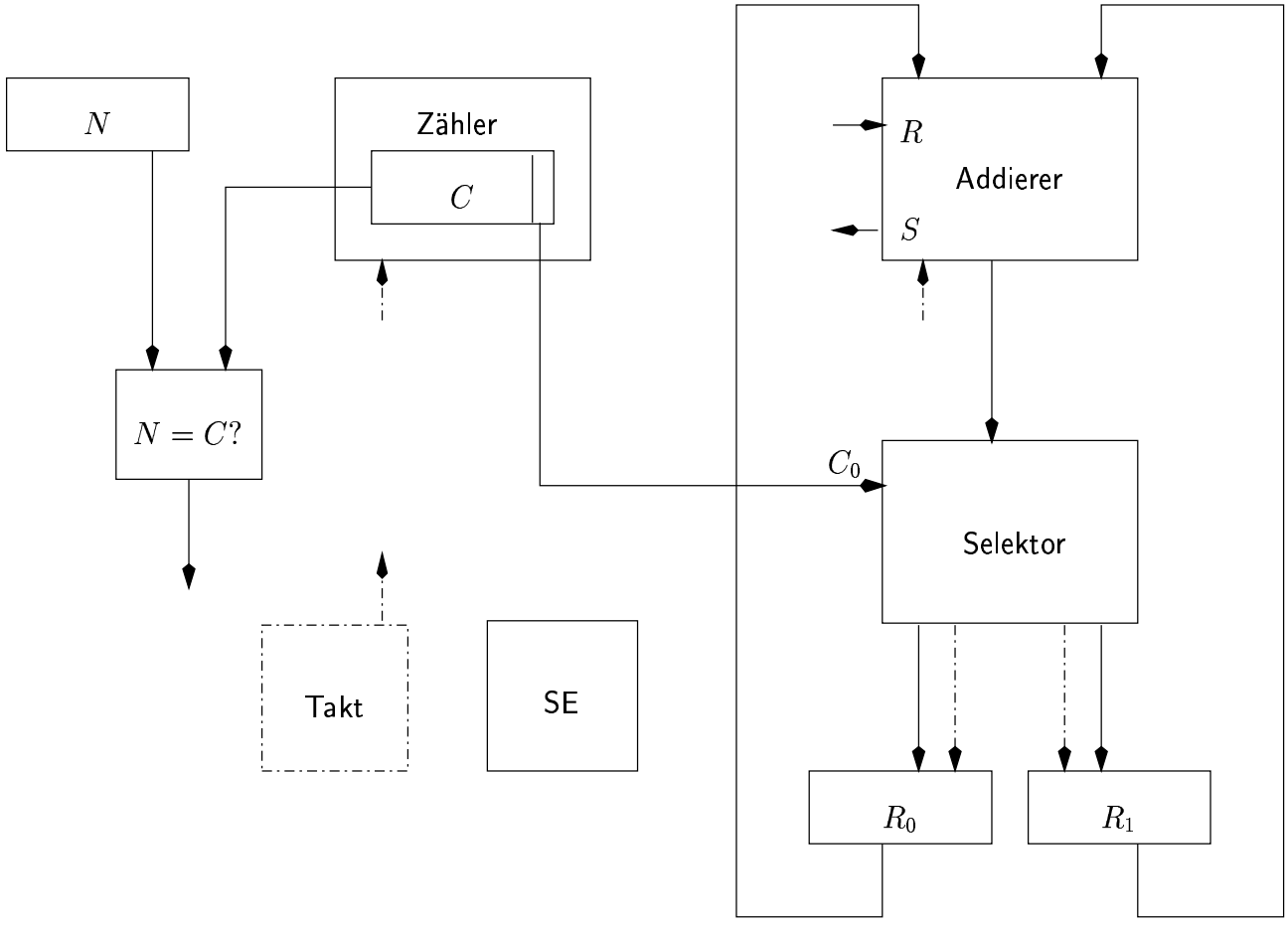
(i) Zeichnen sie Ihr Schaltnetz auf Seite 16(oben) auf. **(8 P.)**

(ii) Fügen sie in der auf Seite 17 vorgegebenen Skizze die von ihnen benötigten Steuerleitungen von und zur Steuereinheit ein. **(3 P.)**

Bitte schreiben Sie deutlich!

Lösung Aufgabe 6 (b) (i)

Lösung Aufgabe 6 (b) (ii)



Bitte schreiben Sie deutlich!

Bitte schreiben Sie deutlich!

Vordiplomsprüfung Informatik II
Sommersemester 2000

Rechnerstrukturen

Matrikelnummer: _____

Name: _____

Vorname: _____

- Am Ende der Klausur finden Sie drei Blätter, die Sie zusätzlich für Ihre Lösungen verwenden können.

Nr.	Punkte	erreichte Punktzahl	Korrigiert von
1	14		
2	20		
3	10		
4	15		
5	16		
6	25		
Gesamt	100		

Aufgabe 1: Zum Aufwärmen (14 Punkte)

(a) Was versteht man unter dem v.-Neumann'schen Flaschenhals? **(2 P.)**

(b) Was unterscheidet prinzipiell Schaltwerke von Schaltnetzen? **(2 P.)**

(c) Konvertieren Sie die folgenden Zahlen. Jede Ziffer steht dabei für eine Stelle.

(i) $(219)_{11}$ ins 13er-System; **(1 P.)**

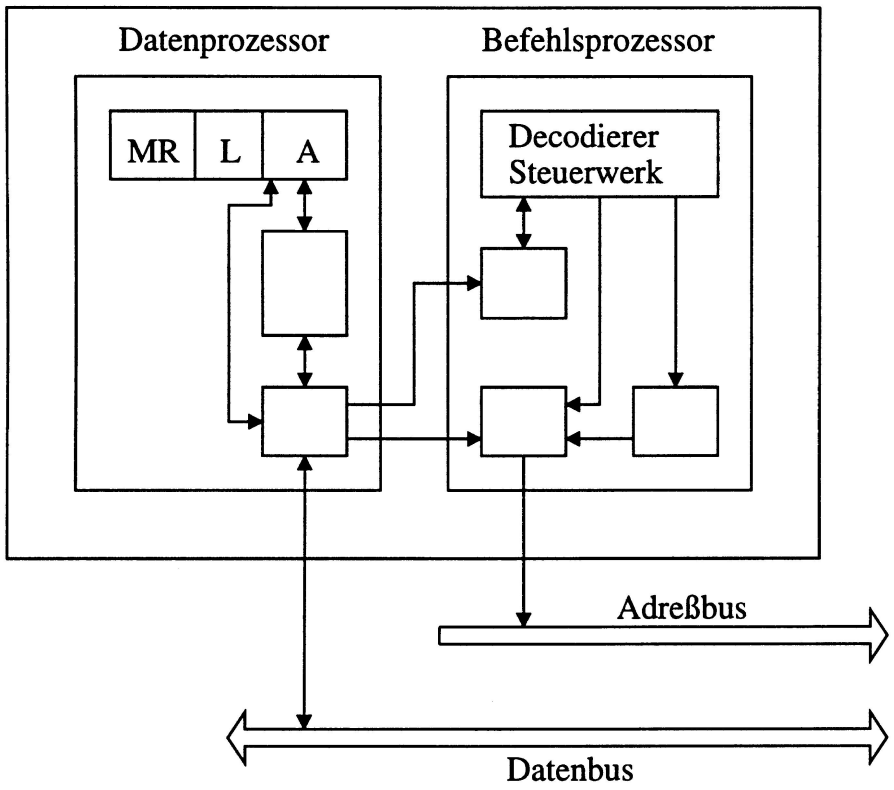
(ii) $(338)_{13}$ ins Hexadezimalsystem; **(1 P.)**

(iii) $(-2234)_{10}$ ins 7er-Komplement zur Basis 8 mit 7 Stellen; **(2 P.)**

Bitte schreiben Sie deutlich!

(d) Stellen Sie 170.3515625 als normalisierte Fließkommazahl zur Basis 2 im Dualsystem dar. Die Mantisse habe dabei eine Länge von 16 Bit, der Exponent von 8 Bit (Herleitung nicht nötig). Das Vorzeichen ist als Teil der Mantisse anzusehen. **(2 P.)**

(e) Betrachten Sie die folgende Abbildung:



(i) Tragen sie die Abkürzungen MBR, MAR, PC, IR und ALU in die leeren Kästchen ein.

(ii) Erläutern Sie kurz (!) ihre Funktion innerhalb der CPU.

Tragen Sie Ihre Lösung auf dem nächten Blatt ein. **(insges. 4 P.)**

Bitte schreiben Sie deutlich!

MBR

MAR

PC

IR

ALU

Bitte schreiben Sie deutlich!

Aufgabe 2: Boolesche Funktionen (20 Punkte)

Wir betrachten Boolesche Funktionen der Form $f : B^n \to B$. In diesem Zusammenhang:

(a) Was ist ein Maxterm? **(2 P.)**

(b) Wie sieht die konjunktive Normalform (KNF) einer Booleschen Funktion aus? **(2 P.)**

(c) Wann ist ein Menge von Booleschen Funktionen *funktional vollständig*? **(2 P.)**

(d) Sei die Menge $\mathcal{F} = \{0, 1, f\}$ von Booleschen Funktionen gegeben, wobei $f : B^3 \longrightarrow B$ und $f(x_2, x_1, x_0) = 1$ gdw. $(x_2 x_1 x_0)_2$ eine Primzahl ist. (Die Konstanten 0 und 1 interpretieren wir großzügig als Funktionen der Stelligkeit 0). Ist $\mathcal{F}$ funktional vollständig? Begründen Sie Ihre Antwort. Sie können sich auf Ergebnisse aus der Vorlesung beziehen. (**Hinweis:** Da die Frage oft gestellt wird: 1 ist keine Primzahl.) **(3 P.)**

Bitte schreiben Sie deutlich!

(e) Wir betrachten nun eine Boolesche Funktion $f : B^4 \to B$ mit

$$f(x_0, x_1, x_2, x_3) = \overline{x}_0\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_0 x_1 x_2 \overline{x}_3 + x_o x_1 x_2 \overline{x}_3 + x_0 x_1 x_2 x_3 + \overline{x}_0\overline{x}_1\overline{x}_2 x_3 + x_0\overline{x}_1\overline{x}_2 x_3.$$

Minimieren Sie diese Funktion mit dem Verfahren von Quine und McCluskey nach den folgenden Anweisungen:

(i) Bestimmen Sie alle Primimplikanten. Vervollständigen Sie dazu folgende Tabellen: **(6 P.)**

Gruppe	Minterm	Einschl. Index	Dezimaldarstellung
0	$x_0x_1x_2x_3$	1111	15
1			
2			
3			
4			

Gruppe	Minterm	Einschl. Index	Dezimaldarstellung
0		111*	14, 15
1			
2			
3			

(ii) Benutzen Sie die von Ihnen gefundenen Primimplikanten in einer Implikationsmatrix und wählen Sie eine kleinstmögliche Menge von Primimplikanten aus. **(4 P.)**

Minterm / Primimpl.						

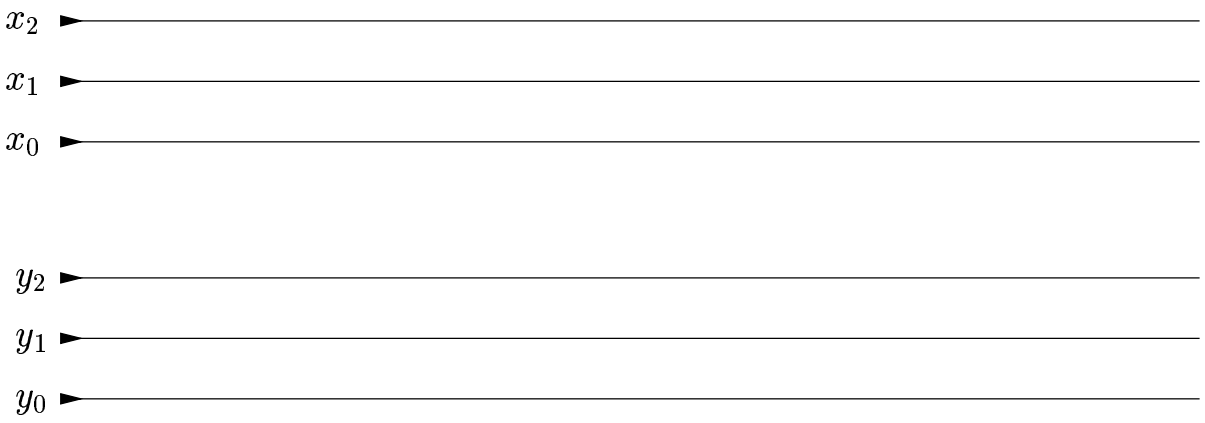
(iii) Geben Sie die minimierte Funktion f an: **(1 P.)**

Bitte schreiben Sie deutlich!

Aufgabe 3: Multiplizierernetze (10 Punkte)

Sie haben in der Vorlesung (eigentlich in der Grundschule) das Multiplizieren nach Schulmethode kennengelernt. Konstruieren Sie aus Halb-Addierern, Volladdierern und anderen Gattern Ihrer Wahl ein Schaltnetz, das zwei 3-stellige positive Dualzahlen $x_2x_1x_0$ und $y_2y_1y_0$ multipliziert. Um auf die Funktionsweise des Multiplizierers zu kommen, empfiehlt es sich, eine Beispiel-Multiplikation im Dualsystem durchzuführen. Sie brauchen höchstens 9 UND-Gatter, 2 HA und 3 VA.

Verwenden Sie die folgende Grafik:



Bitte schreiben Sie deutlich!

Aufgabe 4: PLA (15 Punkte)

(a) Wofür steht die Abkürzung PLA? **(1 P.)**

(b) Welche 4 Bausteintypen gibt es bei PLAs? Wie für Ein- und Ausgänge haben sie und was wird aus den Eingangssignalen berechnet? **(2 P.)**

(c) Im Allgemeinen ist einer der Bausteine entbehrlich. Zeigen Sie, mit welchen Trick das bewerkstelligt werden kann. **(2 P.)**

(d) Wofür steht PAL? **(1 P.)**

(e) Was versteht man unter Faltung eines PLAs? **(2 P.)**

Bitte schreiben Sie deutlich!

(f) Welcher Zusammenhang besteht zwischen PLAs und PALs? **(1 P.)**

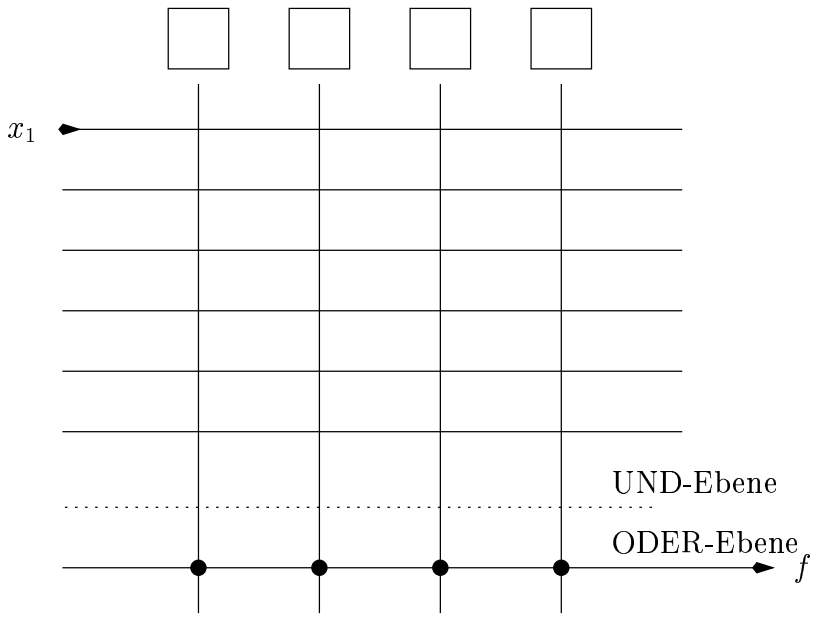
(g) Sei $f : B^5 \longrightarrow B$ gegeben wie folgt:

$$f(x_1, \ldots, x_5) = A + B + C + D$$

wobei

$$\begin{aligned} A &= x_1 x_2 x_5 \\ B &= \overline{x}_1 \overline{x}_3 x_4 x_5 \\ C &= x_1 \overline{x}_2 \overline{x}_5 \\ D &= \overline{x}_1 x_3 x_4 \overline{x}_5 \end{aligned}$$

Implementieren Sie die Funktion als PLA (Punktdarstellung), wobei sie so wenig wie möglich Zeilen in der UND-Ebene verwenden. Führen Sie dazu eine einfache Blockfaltung durch. Verwenden Sie die nachfolgende Abbildung. Kennzeichnen Sie, an welcher Leitung welcher Input anliegt und wo sie welche Leitungen zwischen linker und rechter Seite unterbrechen. Tragen Sie die Namen der Summanden (A, B, C, D) in die Kästchen ein. **(6 P.)**



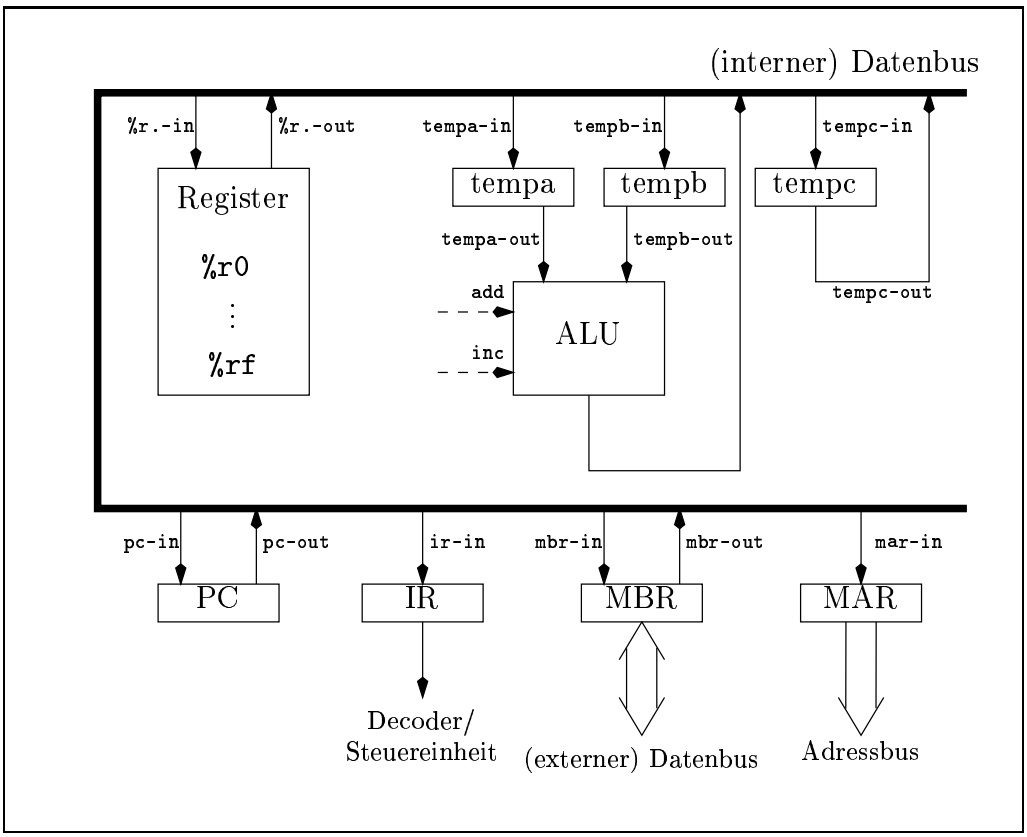
Bitte schreiben Sie deutlich!

Aufgabe 5: Mikroprogrammierung (16 Punkte)

Wir betrachten den folgenden Befehl:

```
ADDW3 *0x10(%r1),(%r0),%r1.
```

Schreiben Sie zu dem Befehl ein Mikroprogramm. Die Struktur des zugrundezulegenden Prozessors ist in der Abbildung gegeben. Es handelt sich im Wesentlichen um die aus der Vorlesung bekannte vereinfachte interne Struktur des WE32100.



Die Steuerleitungen haben folgende Bedeutung:

`x-in` Öffnet das Register `x` zum Schreiben (wobei `x` für `tempa`, `mbr` usw. stehen kann, siehe Abbildung);

`x-out` öffnet das Register `x` zum Lesen (wobei `x` ebenfalls für `tempa`, `mbr` usw. stehen kann, siehe Abbildung);

`add` Stößt die Addition von `tempa` und `tempb` in der ALU an;

`inc` stößt Inkrementierung von `tempa` in der ALU an;

`write` Inhalt von MBR wird zum Speicher übertragen

`read` das Datum, das in der vom MAR adressierten Speicherzelle steht, wird angefordert;

`wait-for-RCS` Mikrobefehl, bei dem solange gewartet wird, bis ein angefordertes Datum aus dem Speicher übertragen wurde.

Die Fetchphase ist bereits vorgegeben. Gehen Sie davon aus, daß die Befehlsdekodierung wie auch die Entschlüsselung der Adressierungsarten bereits mit der Fetchphase abgeschlossen ist, d.h. das Mikroprogramm braucht wirklich nur die Schritte auszuführen, die zur konkreten Realisierung des Befehls nötig sind. Gehen Sie davon aus, daß sich der benötigte Operand `0x10` im internen Register `tempc` befindet.

Bitte schreiben Sie deutlich!

Schritt	Aktion (Steuersignale)
1	`PC-out; MAR-in; read; tempa-in;`
2	`tempa-out; inc; tempc-in;`
3	`tempc-out; PC-in; wait-for-RCS;`
4	`MBR-out; IR-in;`
⋮	

Ihre Aufgabe besteht nun darin, das Mikroprogramm ab der 5. Zeile zu vervollständigen. Beachten Sie, daß hierzu etwa 9 zusätzliche Mikroprogrammschritte notwendig sind !

(16 P.)

Aufgabe 6: Assembler (25 Punkte)

(a) Geben Sie jeweils an, welcher Wert in Register `%r1` geladen wird:

(i) Register-Modus: `MOVW %r0, %r1` **(1 P.)**

(ii) Immediate-Modus: `MOVW &12, %r1` **(1 P.)**

(iii) Register-Deferred-Modus: `MOVW (%r0), %r1` **(1 P.)**

(iv) Register-Displacement-Deferred-Modus: `MOVW *4(%r0), %r1` **(1 P.)**

(b) Wie wird dafür Sorge getragen, daß die Werte der Register bei Aufruf der Prozedur nicht verloren gehen, so daß sie also später wieder hergestellt werden können? **(2 P.)**

(c) Die Fibonacci für $n \geq 0$ sind rekursiv definiert wie folgt:

```
Fib(n)  :=  if (n = 0) or (n = 1) then return n
            else return (Fib(n-1) + Fib(n-2));
```

Die Fibonacci-Zahlen lassen sich auch sehr gut iterativ berechnen, zum Beispiel mit Hilfe des auf Seite 13 angedeuteten while-Programms.

Komplettieren Sie die mit ??? gekennzeichneten Programmzeilen. **(5 P.)**

Bitte schreiben Sie deutlich!

```
                function fib (n: integer): integer;     (* Annahme: n ≥ 0 *)
                var l, fib1, fib2, ret: integer;
                begin
???                 if ( ..... )
???                 then ...
                    else begin (* n ≥ 1 *)
???                     fib1 := ...
???                     fib2 := ...
                        l := 1;
???                     while ( ..... )
                        begin
???                         ret := ...
???                         fib1 := ...
???                         fib2 := ...
???                         l := ...
                        end
                    end
                return ret;
                end;
```

Bitte schreiben Sie deutlich!

(d) Implementieren Sie nun eine WE32100-Assembler-Prozedur. Dabei sollen folgende Register den oben angegebenen Variablen entsprechen:

- $n \longrightarrow$ `%r0`
- $fib_1 \longrightarrow$ `%r1`
- $fib_2 \longrightarrow$ `%r2`
- $l \longrightarrow$ `%r3`
- Ergebnis $\longrightarrow$ `(%r4)`

Verwenden Sie das auf Seite 15 vorgegebenen Lösungsblatt. Die wichtigsten Assembler-Befehle sind auf Seite 16 aufgelistet. **(14 P.)**

Vereinbarung: Reihenfolge von Operatoren. Drei-Operanden-Befehle haben die Struktur `OP3` src_1, src_2, dst. Im Buch von Oberschelp ist nicht klar, in welcher Reihenfolge die Operanden behandelt werden. Für diese Klausur sei vereinbart, daß ein Befehl `OP3` src_1, src_2, dst folgendes berechnet:

$$dst \leftarrow src_2 \ op \ src_1,$$

wobei op die eigentliche zu berechnende Operation ist. Ein Zwei-Adressbefehl hat die Form `OP` src, dst und berechnet

$$dst \leftarrow dst \ op \ src.$$

Dies gilt auch für den `CMP`-Befehl: `CMP X,Y` berechnet `Y - X`.

```
        .globl fib                      #   . . . . . .
fib:    SAVE %fp                        #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
then:   . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
else:   . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
while:  . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
begin:  . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
        . . . . . .                     #   . . . . . .
ready:  RESTORE %fp                     #   . . . . . .
        RET                             #   . . . . . .
```

Bitte schreiben Sie deutlich!