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

RWTH Aachen
Ahornstraße 55
52074 Aachen

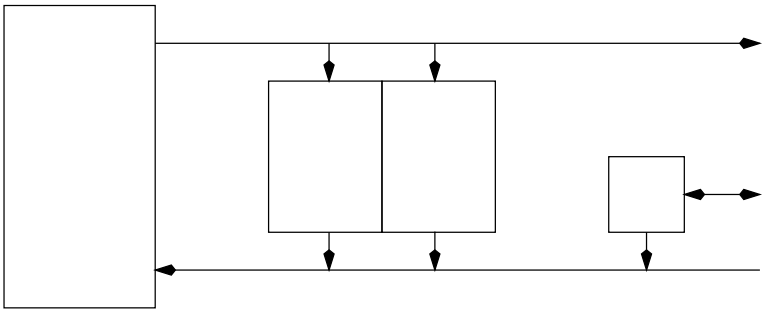
Diplomvorprüfung im Fach „Rechnerstrukturen“ (Wintersemester 97/98)

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!

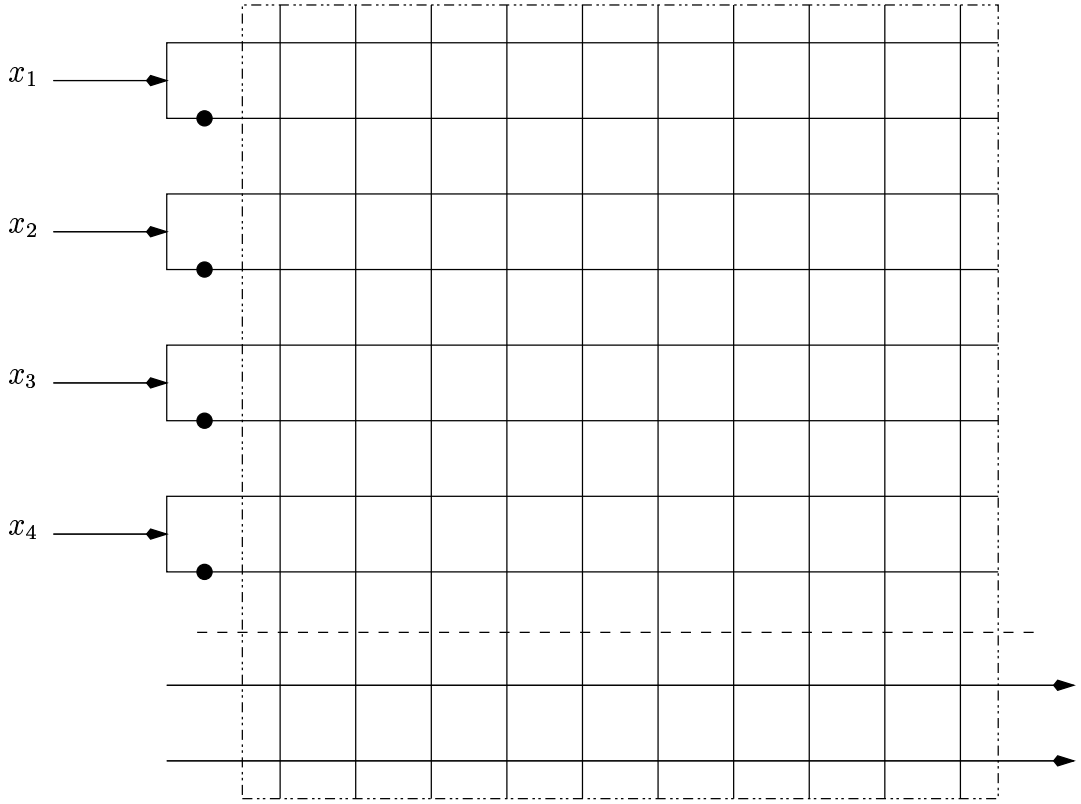


Abbildung 1: Lösungsskizze zu Aufgabe 2

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

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

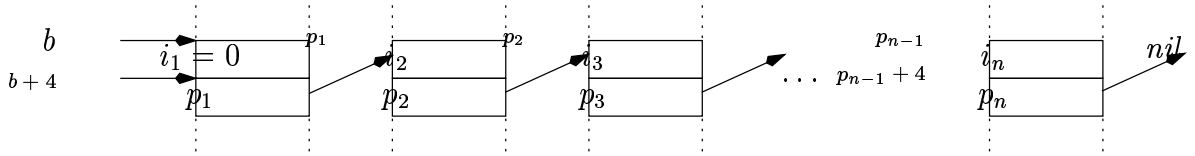
d.h. $g(x_i/a)$ entsteht aus g durch *feste* Belegung der Variablen x_i mit dem Wert $a \in B$.

(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)

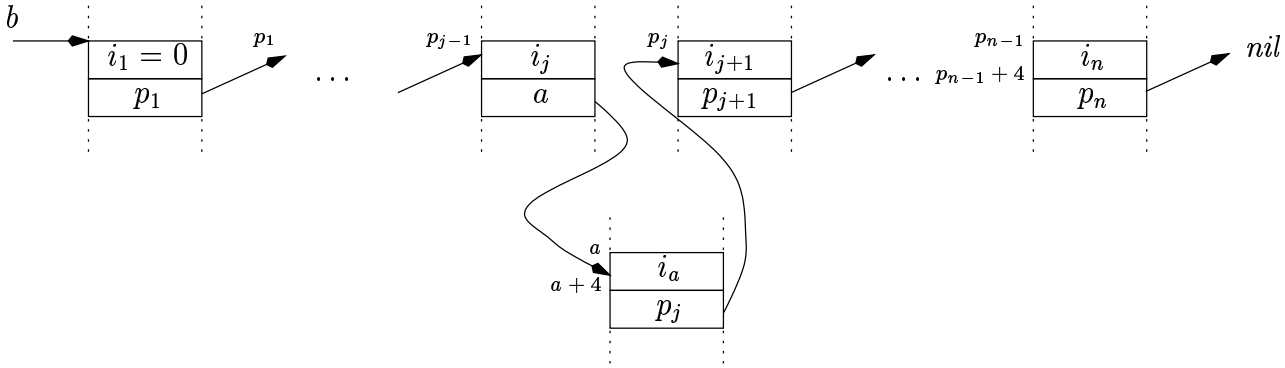
Bitte notieren Sie die Lösungen dieser Aufgabe auf einem separaten Blatt (kariert).
Eine (einfach) verkettete Integer-Liste der Länge n mit Inhalt $i_1 \dots i_n$ habe die folgende Struktur:



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.)**

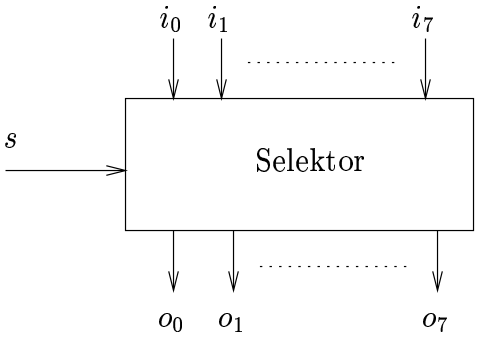
(d) Bauen Sie jetzt ein Schaltwerk zur Berechnung von $z := x$ modulo y, wobei Sie annehmen dürfen, daß $x, y > 0$; x und y sind mit 8 Bits im Zweier-Komplement kodiert. Als Bauelemente stehen zur Verfügung:

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

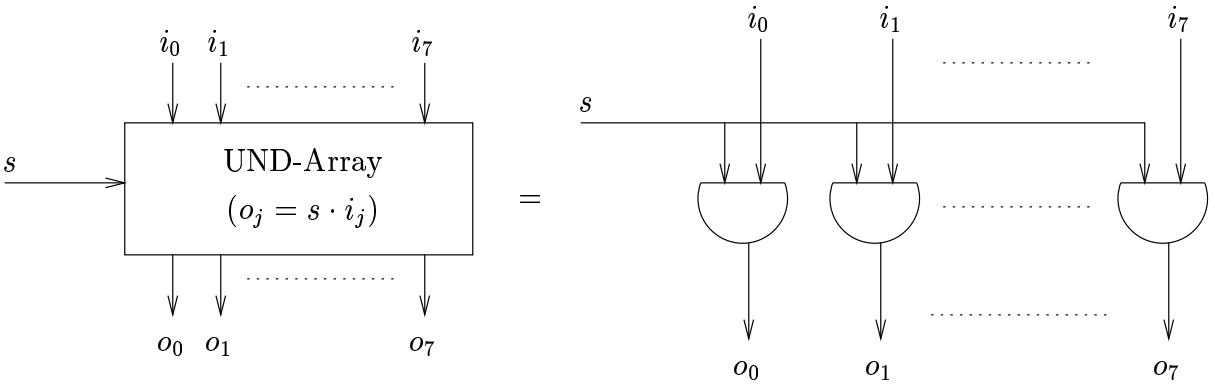
- Ein 1-Bit-Register für Start/Stop-Zwecke („Ready-Register“, siehe unten);

- 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.)**

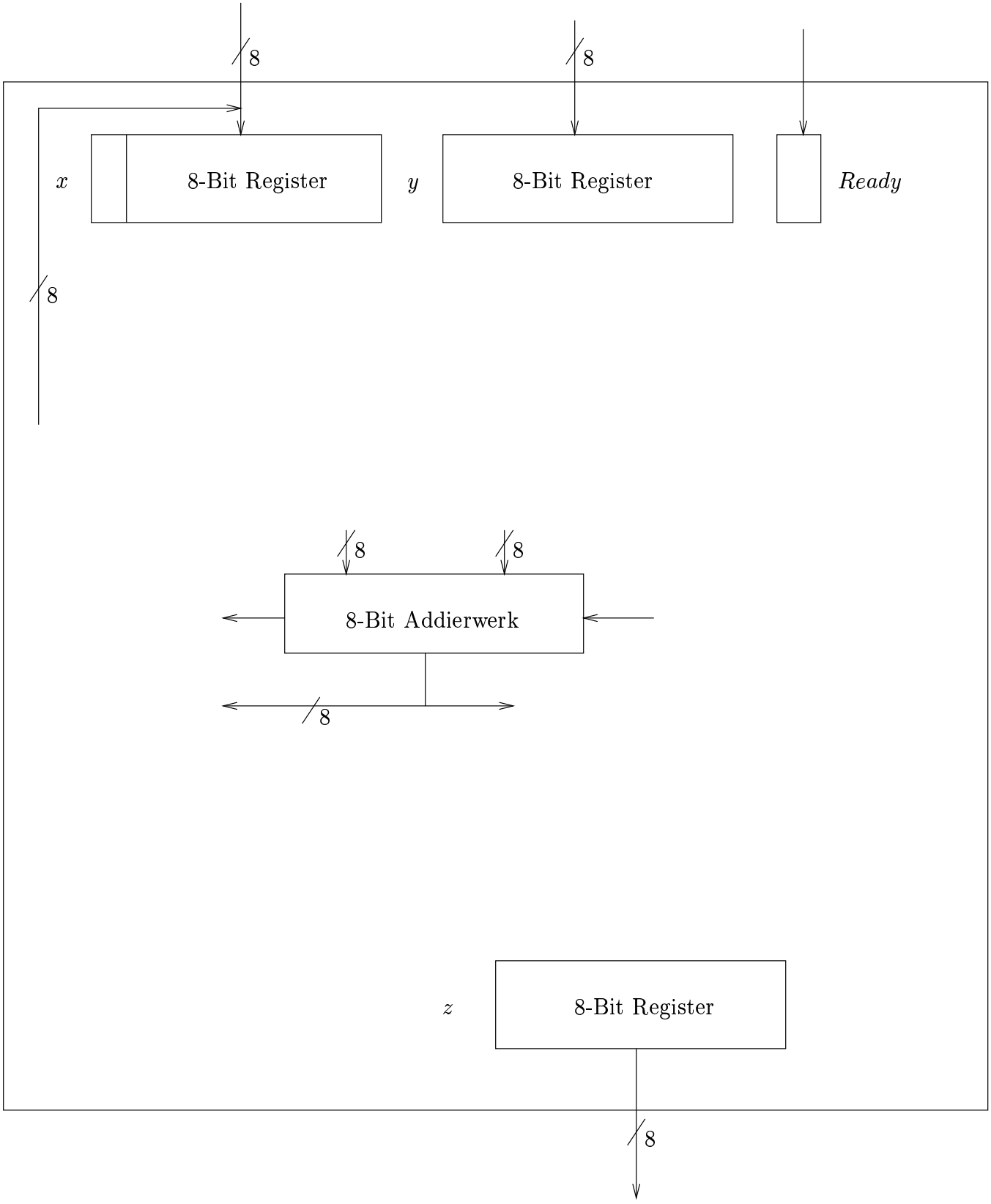


Abbildung 2: Lösungsskizze zu Aufgabe 5