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.)**

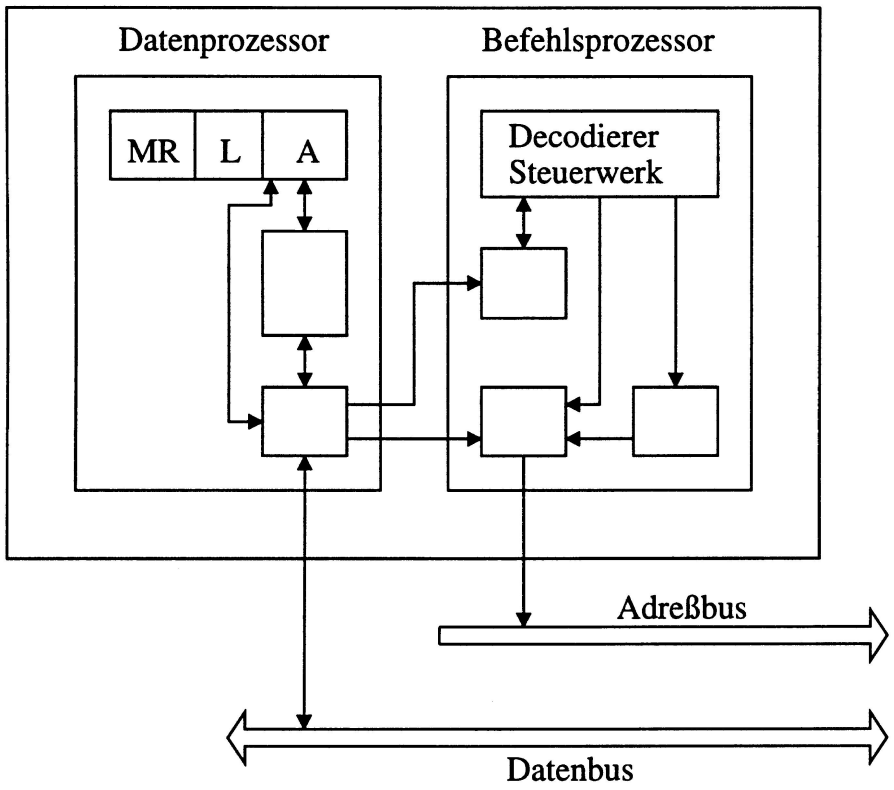
(ii) $(338)_{13}$ ins Hexadezimalsystem; **(1 P.)**

(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.)**

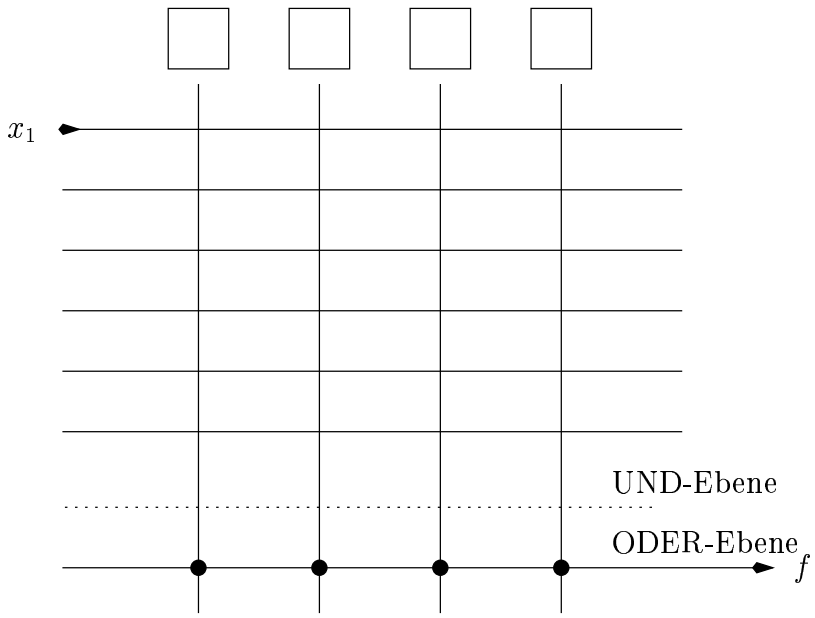
(g) Sei $f : B^5 \longrightarrow B$ gegeben wie folgt:

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

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.)**



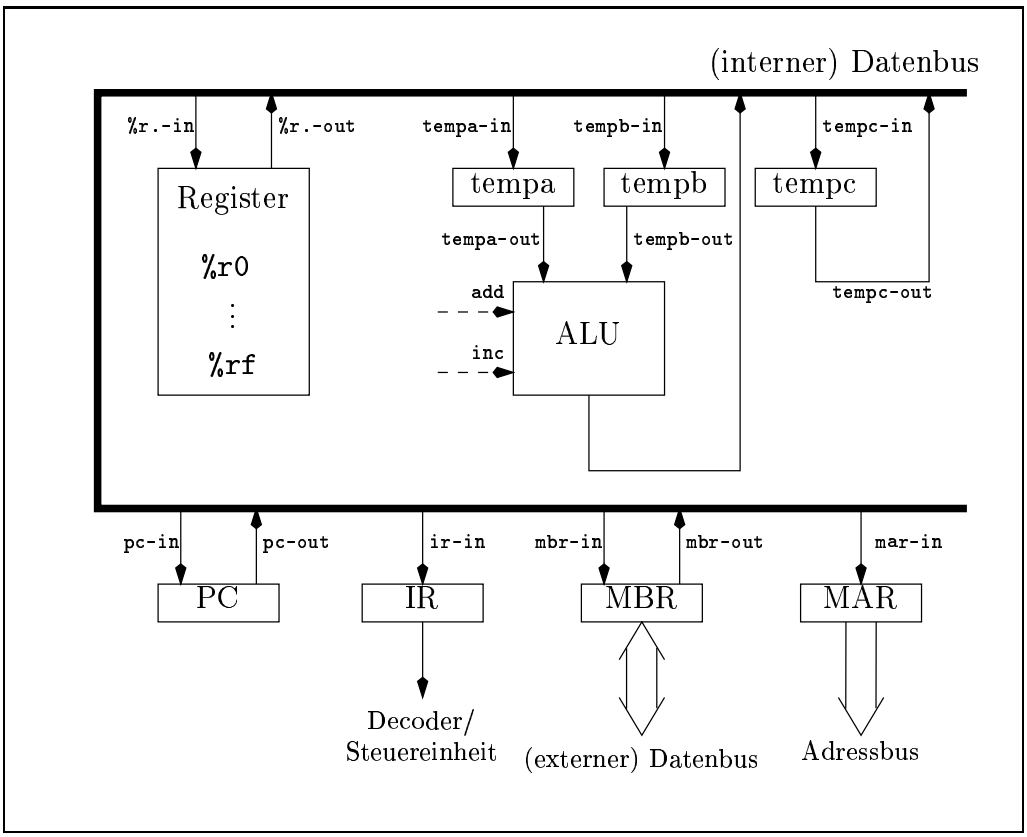
Bitte schreiben Sie deutlich!

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`.

Bitte schreiben Sie deutlich!

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

Bitte schreiben Sie deutlich!