Gedächnisprotokoll

Rechnerorganisation: Ersttermin WS 14/15

U.U. Gab es noch mehr kleine Aufgaben und die Daten bei den Wertetabellen sind nicht unbedingt die selben, aber das Prinzip gleich. Die meisten Aufgaben sollten aber drin sein.

Aufgabe 1

Multiple Choise

Gegeben ist folgende Wertetabelle.

a	b	c	out
0	1	1	0
1	1	0	1
1	1	1	0
0	0	1	1
0	1	0	1
1	0	1	1
1	0	0	0

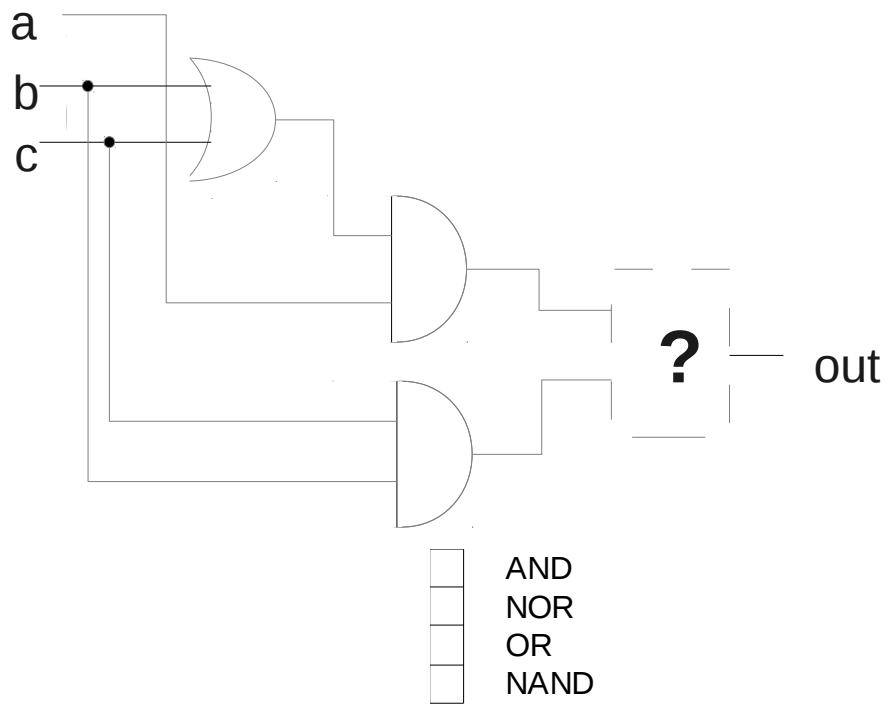
1) Welches ist die dazugehörige Disjunktive Normalform?

☐	$(a+\overline{\overline{b+c}})*(a+\bar{b}+c)*(\bar{a}+\bar{b}+c)$
☐	$\bar{a}\bar{b}\bar{c}+\overline{\overline{abc}}+\bar{a}\bar{b}\bar{c}+a\overline{\bar{b}\bar{c}}$
☐	$(a+\bar{b}+c)*(\bar{a}+\bar{b}+\bar{c})*(\bar{a}+b+c)$
☐	$ab\bar{c}+\overline{\bar{a}bc}+\bar{a}\bar{b}\bar{c}+a\bar{b}c$

Schaltbild erweitern

2) Welches Gatter muss eingefügt werden damit die Gleichung stimmt?

Out = ab+ac+bc



Gegeben ist die folgende Wertetabelle:

a	b	c	out
0	1	1	0
1	1	0	1
1	1	1	0
0	1	0	1
1	0	1	1
1	0	0	0

3) Malen sie das dazugehörige Schaltbild auf.

Aufgabe 2

1) Gegeben sind die Zahlen 168_{10} und -77_{10} im Dezimalsystem. Die beiden zahlen werden in einem 10-Bit Dualsystem **addiert**.

1. Tritt bei der Addition eine Bereichsüberschreitung statt? Sagen sie warum/warum nicht OHNE die Berechnung durchzuführen.
2. Führen sie die Berechnung durch und zeigen sie kurz, warum ihr Ergebnis aus 1 zutrifft.

2) Multiplikation

Gegeben sind die folgenden zwei Zahlen in IEEE Mini-float Format. Der BIAS beträgt dabei 15.

0	10001	1011010011
0	10110	0110000000

1) Multiplizieren sie die beiden Zahlen im binären.
2) Bei der Multiplikation treten Ungenauigkeiten auf. Erklären sie kurz warum.

Aufgabe 3

MIPS Assembler

Überführen Sie die folgende Funktion in MIPS Assembler. Die Funktion swap kann als gegeben vorausgesetzt werden. Sie vertauscht zwei nebeneinanderliegende Elemente. Dabei gibt array die Adresse des ersten Array-Elementes an und size die Größe des Arrays. Achten Sie darauf zu kommentieren.

```
void somefunction(int array[], size){
     int i;
     for(i=0; i < size - 1; i++){
          if(array[i] < array[i+1])
               swap(array,i);
     }
}
```

Aufgabe 4

Leistung

Ein Programm hat folgende Verteilung von Befehlen:

35% Load
50% ALU
10% Branch
5% Jump

Dabei gehen wir davon aus das 35% parallelisierbar sind.

Wir haben einen MIPS Multicycle-Prozessor, der mit einem L1-Cache ausgestattet ist. Dieser hat eine Hitrate von 60%. Wenn die richtigen Daten im Cache gespeichert sind, dann dauert es 1 Takt um die Daten zu lesen. Bei einem Fehlversuch braucht der Prozessor 35 Takte um die Daten aus dem Hauptspeicher zu holen.

1) Was ist die durchschnittliche Dauer eines Load-Befehls?
2) Berechnen Sie die CPI für unser Programm und Prozessor.
3) Es werden nun zwei Verbesserungsvorschläge gemacht. Geben Sie an welcher Vorschlag die Geschwindigkeit unseres Programms am meisten steigert und zeigen sie warum.
    1. Die Hitrate des Cache wird auf 90% erhöht.
    2. Wir erweitern das System auf 4 Prozessorkerne.

Aufgabe 5

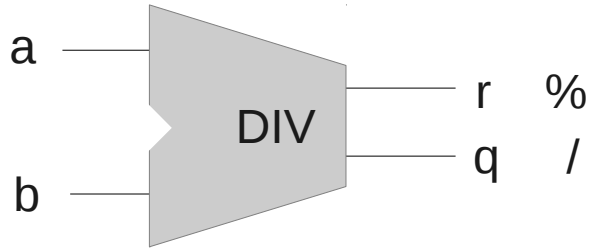
Datenpfaderweiterung

Wir haben den MIPS Singlecycle Datenpfad aus der Vorlesung, bei dem schon eine Miltipliere-Einheit hinzugefügt wurde.

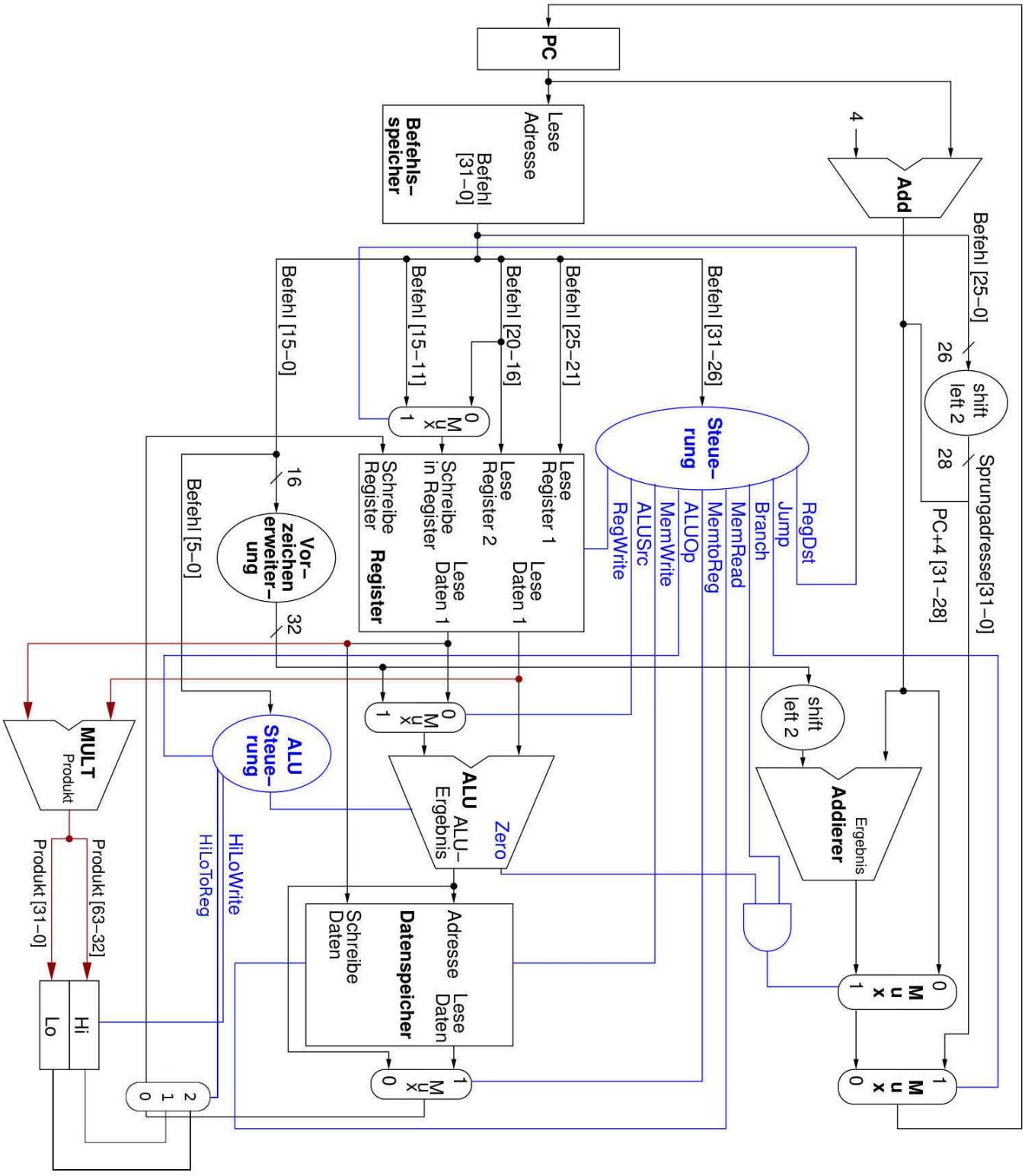
Dieser Datenpfad soll nun um die funktion `div` erweitert werden.
Laut der MIPS Spezifikation hat der Befehl das Format. Der Befehl soll auch die Division durch Null behandeln. Wenn durch Null devidiert wird soll das Ergebnis auch null sein.

1) Fügen sie den Befehl in den Datenpfad ein
2) Geben sie die Steuerbefehle an. Vergessen Sie dabei nicht HiLoWrite und HiLoToReg.

Lo = [Rs] % [Rt]
Hi = [Rs] / [Rt]



Name: Mtr.-Nr.:


PC
Befehls-speicher
Lese Adresse
Befehl [31–0]
4
Add
Befehl [25–0]
shift left 2
26
28
Sprungadresse[31–0]
PC+4 [31–28]
Befehl [31–26]
Befehl [25–21]
Befehl [20–16]
Befehl [15–11]
Befehl [15–0]
Befehl [5–0]
Steue-rung
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Register
Lese Register 1
Lese Register 2
Schreibe Register
Schreibe in Register
Lese Daten 1
Lese Daten 1
Vor-zeichen erweiter-ung
16
32
shift left 2
Addierer
Ergebnis
ALU
ALU Ergebnis
Zero
ALU Steue-rung
MULT
Produkt
Produkt [63–32]
Produkt [31–0]
HiLoWrite
HiLoToReg
Hi
Lo
Datenspeicher
Adresse
Schreibe Daten
Lese Daten
M u x

Aufgabe 6

Caches

Nach der Ausführung eines Programms ist der Zustand des Cache wie folgt gegeben.

		Tag	Daten
0	0	0x29A	0xAAAAAAAAAAAAAAAA
1	0	0x29A	0xBBBBBBBBBBBBBBBB
2	1	0x59B	0xCCCCCCCCCCCCCCCC
3	0	0x2A1	0xDDDDDDDDDDDDDDDD
4	0	0x591	0xEEEEEEEEEEEEEEEE
5	1	0x09A	0xFFFFFFFFFFFFFFFF
6	1	0x031	0xAAAAAAAAAAAAAAAA
7	0	0x111	0xBBBBBBBBBBBBBBBB
8	0	0x2CA	0xCCCCCCCCCCCCCCCC
9	1	0x89A	0xDDDDDDDDDDDDDDDD
10	1	0x293	0xEEEEEEEEEEEEEEEE
11	1	0x171	0xFFFFFFFFFFFFFFFF
12	0	0x1C0	0xAAAAAAAAAAAAAAAA
13	0	0x293	0xBBBBBBBBBBBBBBBB
14	0	0x237	0xCCCCCCCCCCCCCCCC
15	1	0x1CC	0xDDDDDDDDDDDDDDDD

1) Wie viele Bits werden benötigt um den Cache zu implementieren?
2) Die Adresse hat nicht die gewöhnliche Länge. Berechnen sie die Adresslänge und geben sie dabei auch benötigten Bits für Index, Offset und Tag an.
3) Als nächstes werden vom Prozessor folgende Daten generiert. Schreiben Sie auf wie der Cache anschließend aussieht. Füllen sie nur Kästchen aus, die sich ändern. Es wird immer ein ganzer Block überschrieben.

```
i)      0x29AA6   0xAAAAAAAAAAAAAAAA
ii)     0x29A9E   0xCCCCCCCCCCCCCCCC
iii)    0x29ADE   0xDDDDDDDDDDDDDDDD
iv)     0x29AD4   0xBBBBBBBBBBBBBBBB
```

		Tag	Daten
0	0	0x29A	0xAAAAAAAAAAAAAAAA
1	0	0x29A	0xBBBBBBBBBBBBBBBB
2	1	0x59B	0xCCCCCCCCCCCCCCCC
3	0	0x2A1	0xDDDDDDDDDDDDDDDD
4	0	0x591	0xEEEEEEEEEEEEEEEE
5	1	0x09A	0xFFFFFFFFFFFFFFFF
6	1	0x031	0xAAAAAAAAAAAAAAAA
7	0	0x111	0xBBBBBBBBBBBBBBBB
8	0	0x2CA	0xCCCCCCCCCCCCCCCC
9	1	0x89A	0xDDDDDDDDDDDDDDDD
10	1	0x293	0xEEEEEEEEEEEEEEEE
11	1	0x171	0xFFFFFFFFFFFFFFFF
12	0	0x1C0	0xAAAAAAAAAAAAAAAA
13	0	0x293	0xBBBBBBBBBBBBBBBB
14	0	0x237	0xCCCCCCCCCCCCCCCC
15	1	0x1CC	0xDDDDDDDDDDDDDDDD
