Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Tutoren der Vorlesung

1. Probeklausur Systemarchitektur

(zu bearbeiten bis: 23.07.2011)

Name	Matrikelnummer

Allgemeine Hinweise

- Bitte schauen Sie nicht in die Klausur bis Sie dazu aufgefordert werden, so dass jeder die gleiche Bearbeitungszeit erhält.
- Sie haben **120 Minuten** für die Bearbeitung der Aufgaben. Das richtige Lösen aller Aufgaben bringt **175 + 20 Punkte**. Zum bestehen genügen **85 Punkte**.
- Nur das von uns gestellte Papier darf während der Klausur verwendet werden. Die Klausuraufgaben, Ihre Lösungen, sowie eventuelle Notizen werden im folgenden als Klausurmaterial bezeichnet.
- Alle Blätter des Klausurmaterials inklusive des Titelblatts müssen vor Abgabe mit **Namen und Matrikelnummer** versehen werden.
- Auf jedem Blatt darf maximal eine Aufgabe bearbeitet werden. Diese ist auf dem Blatt kenntlich zu machen. Blätter die entweder über keinen Namen, Matrikelnummer, Aufgabenbeschriftung verfügen oder mehr als die Lösung einer Aufgabe beinhalten **werden nicht bewertet**.
- Zugelassene Hilfsmittel sind neben einem nichprogrammierbarem Taschenrechnerden nur die von uns ausgegebenen Hilfsmitteln, z.B.:

Gatter	NOT	NAND/NOR	AND/OR	XOR/XNOR	MUX
Kosten	1	2	2	4	3
Tiefe	1	1	2	2	2

- Unsaubere und unleserliche Lösungen **werden nicht bewertet**. Dies gilt insbesondere für Konstruktionsskizzen. Zudem darf **nicht** mit Bleistift geschrieben werden.
- Betrugs- und Täuschungsversuche führen zum Nichtbestehen der Prüfung. Nur zugelsassene Studenten sind teilnahmeberechtigt.
- Die Klausur beinhaltet 12 Aufgaben plus zwei Zusatzaufgaben und hat insgesammt **7 Seiten**. Bitte kontrollieren Sie, **bevor** Sie mit der Bearbeitung beginnen, als erstes die Vollständigkeit der Aufgabenblätter.
- Makieren Sie die Aufgaben die Sie bearbeitet haben in der folgenden Tabelle. **Nur die makierten Aufgaben werden korrigiert**.

Aufgabe	1	2	3	4	5	6	7	8	9	10	11	12	ZA1	ZA2	$\sum$
Max	10	10	10	10	15	20	15	5	25	35	15	5	15	5	175 + 20
Bearbeitet															
Punkte															

Aufgabe 1: (Bool'sche Algebra und NAND) **(5+5 Punkte)**

(a) Stellen Sie ein NOR - Gatter nur mittels NAND - Gattern dar und beweisen Sie die Korrektheit der Konstruktion.

(b) Erstellen Sie einen Schaltkreis Tri - NAND, der $x_0 \barwedge x_1 \barwedge x_2$ berechnet und der drei Eingänge $x_0, \ldots, x_2$ und einen Augang y_0 besitzt.
Zeigen Sie dessen Korrektheit.

Aufgabe 2: (Induktion) **(4+6 Punkte)**

Beweisen Sie folgende Gleichungen für Summenformeln mittels vollständiger Induktion:

(a) $\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$

(b) $\sum_{i=0}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$

Aufgabe 3: (Rekurrenzen) **(5+5 Punkte)**

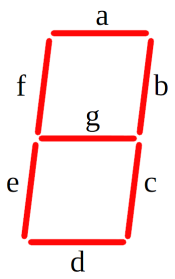
Beweisen Sie folgende Rekurrenzen mittels vollständiger Induktion:

(a) $C(1) = 1,\ C(n) = C(n-1) + n$ für $n \in \mathbb{N}$

(b) $C(1) = 1,\ C(n) = C(\frac{n}{2}) + 2$ für $n = 2^k,\ a, k \in \mathbb{N}$

Aufgabe 4: (Disjunktive Normalform) **(10 Punkte)**

Eine einfache digitale Anzeige benutzt oft die sogenannte Siebensegmentanzeige. Dabei werden alle Ziffern von 0 bis 9 aus bis zu sieben einzelnen "Strichen" zusammengesetzt:



Ziel ist es nun dieses Verhalten in einem Schaltkreis zu implementieren. Zur Steuerung dienen vier Eingangsbits $x_3, \ldots, x_0$ (binäre Darstellung der Ziffern 0 - 9). Die Ausgänge a - g entsprechen der obigen Abbildung.

(a) Stelle eine Wahrheitstabelle für die Ausgänge a, d und f auf.

(b) Bilde die disjunktive und die konjunktive Normalform für die Ausgänge a und f

(c) Zeichne einen Schaltkreis zur Berechnung von a.

(d) Gib Kosten und Tiefe dieses Schaltkreises an.

Hinweis: Eingangsbelegungen die keiner Dezimalziffer entsprechen dürfen beliebige Werte anzeigen und müssen nicht implementiert werden.

Aufgabe 5: (Two's complement Zahlen) **(3+3+3+3+3 Punkte)**

Beweisen Sie die Korrektheit der folgenden Lemmata über Two's Complement Zahlen:

(a) $[0a] = \langle a \rangle$

(b) $[a] \equiv \langle a[n-2:0] \rangle \mod 2^{n-1}$

(c) $[a] \equiv \langle a \rangle \mod 2^n$

(d) $[a[n-1], a] = [a]$

(e) $-[a] \equiv [\neg a] + 1$

Aufgabe 6: (Hardware die Erste) **(5+5+10 Punkte)**

Unter Paralleler Prefix Berechnung ($PP_\circ$) versteht man einen Schaltkreis der aus einem Bitstring $x[n-1:0]$ einen Bitstring $y[n-1:0]$ der Form $y_i = x_0 \circ \ldots \circ x_i$ berechnet. Bei dem Gatter/Schalkreis $\circ$ handelt es sich hierbei um irgendein beliebiges Gatter bzw. irgendeinen beliebigen Schaltkreis, der zwei Eingänge x_0, x_1 besitzt und aus ihnen einen Ausgang y_0 berechnet, mit $y_0 = x_0 \circ x_1$. Nehmen Sie der Einfachheit halber an, dass $n = 2^k$ gilt.
Ihre Aufgabe ist es nun:

(a) Einen Schaltkreis zu Konstruieren der die $PP_\circ$ implementiert.

(b) Die Kosten - und Tiefenformel für Ihren $PP_\circ$ - Schaltkreis aufzustellen und zu beweisen.

(c) **Bonus:** Den Schaltkreis in Aufgabenteil (a) so zu konstruieren, dass seine Tiefe $D_{PP_\circ} \in O(log_2(n))$ ist und darüber auch den Beweis aus Aufgabenteil (b) zu führen.

Aufgabe 7: (Hardware die Zweite) **(5+10 Punkte)**

In der Vorlesung, sowie in der Übung, wurde der ConditionalSumAdder (kurz: CSA) behandelt, hier sollen Sie nun:

(a) Den Schaltkreis des CSA angeben.

(b) Dessen Korrektheit bezüglich der Addition beweisen.

Aufgabe 8: (Typtabelle) **(5 Punkte)**

Betrachten Sie folgende Typdeklarationen und globale Variablendeklarationen:

```
typedef int[11] myIntArr;
typedef intarr[11] myMatrix;
myIntArr y;
myMatrix p;
int z;
```

Geben Sie die dazugehörige Typtabelle $tt(t)$ inkl. $size(t)$ an.

Aufgabe 9: (Arbeiten mit Bäumen) **(5+5+10+5 Punkte)**

(a) Leiten Sie das folgende Teilprogramm gemäß der C0-Grammatik ab: $y[3] - - z$

(b) Wenden Sie Post-Order-Traversal an um die Knoten des Ableitungsbaumes fortlaufend zu nummerieren, beginnen Sie mit 0 und ignorieren Sie Klammern sowie Operatoren.

(c) Sei eine Konfiguration c gegeben. Bestimmen Sie für alle Knoten i des Ableitungsbaums $etyp(i), va(c,i)$ sowie $lv(c,i)$, falls definiert. Beschränken Sie sich auf Knoten oberhalb von $\langle Bu \rangle$ und $\langle ZiF \rangle$.

(d) Wieviele Marken brauchen Sie laut dem Beweis in der Vorlesung höchstens um den Baum auszuwerten? Geben Sie eine Auswertungsreihenfolge an, die diese Obergrenze einhält.
Hinweis: Benutzen Sie $r(i)$ für $remove(i)$, $s(i)$ für $set(i)$ und $m(i,j)$ für $move(i,j)$.

Aufgabe 10: (C0 und Assembler) **(5+8+4+8+10 Punkte)**

Gegeben sei das folgende C0 - Programm:

```
typedef int* intp;
intp p;
bool b;

int fun(int c){
  if b = 0 then{
    p* = p* + 1;
    c = c-9;
    b = 1
  }else{
    c = c + 5;
    b = 0
  }
  return c
}

int main (){
  int a;
  p = new int*;
  a = 27;
  b = 0;
  p* = 0;
  while a>7 do {
    a = fun(a)
  }
  return a
}
```

(a) Geben Sie die Anfangskonfiguration c^0 an.

(b) Geben Sie die MIPS-Frames für $main$ und fun an.

(c) Gegeben sei folgende Konfiguration c:

$c.pr = ft(fun).body;\ while\ a > 7\ do\ \{a = fun(a)\};\ return\ a;$
$c.V = \{gm, (main,\ 0), (fun, 1),\ 0\}$
$c.ht = \{0 \to int\}$
$c.nh = 1$
$c.rd = 1$
$c.m = \{gm \to \{p \to 0_{32},\ b \to 0_{32}\},\ 0 \to 0_{32},$
$(main,\ 0) \to \{\$rv \to null,\ a \to 27_{32}\},$
$(fun,\ 1) \to \{\$rv \to (main,\ 0).a,\ c \to 27_{32}\}\}$

Berechnen Sie $ba(c,\ (fun,\ 1))$ sowie $ba(c,\ (fun,\ 1).c$ so genau wie möglich.

(d) Geben Sie die Konfigurationen bis

$$c^n.pr = while\ a > 7\ do\ \{a = fun(a)\};\ return\ a;$$

gilt an.

(e) Übersetzen Sie nun fun in Assemblercode.

Aufgabe 11: (Assembler) **(5+4+6 Punkte)**

Im folgenden soll die Assembler-Welt etwas bunter gestaltet werden (genauer gesagt der Bildschirm). Sie können die Makros aus den Teilaufgaben auch nutzen, wenn Sie diese nicht gelöst haben, ansonsten sind nur die mitgegebenen MIPS-Instruktionen erlaubt.

(a) Geben Sie Assembler-Code $modulo(r, a, n)$ mit folgender Semantik an:

$$gpr(r) = gpr(a) \mod n$$

Sie können davon ausgehen dass $n = 2^k$ und $k < 32$ gilt.

(b) Wir definieren uns nun $spr(9)$ als "Zeitregister", in dem die Zeit seit Systemstart in Millisekunden ($modulo\ 2^{32}$) gespeichert ist. Geben Sie Assembler-Code $random(n)$ an, der eine "Zufalls"-Zahl zwischen 0 und 2^n in Register 5 speichert. Sie können annehmen, dass $n < 32$ gilt. Jede Zahl soll dabei in etwa die gleiche Chance haben.

(c) Wir definieren den Assembler-Code $draw(x, y, f)$ so, dass auf dem Bildschirm an Position $(\langle gpr(x)\rangle, \langle gpr(y)\rangle)$ die Farbe $§\langle gpr(f)\rangle$ erhält. Dabei gilt §0 = "weiß", §1 = "rot", §2 = "grün", §3 = "blau". Alle anderen eingaben führen zu einem Aufruf an das Betriebssystem. Geben Sie nun Assembler-Code an, der bei einer $800 * 600$ Auflösung jeden Pixel mit einer zufälligen Farbe einfärbt. (Beginnend bei $(0, 0)$)

Aufgabe 12: (Einfacher Struct Zugriff) **(5 Punkte)**

Sei folgende Assemblersubroutine gegeben:

$$amul(u, x, y, z)$$

mit folgender Semantik:

$$d.gpr(x) = d.gpr(u) + d.gpr(y) \cdot d.gpr(z)$$

Geben Sie den Code an für den Struct-Zugriff indem Sie einen Test hinzufügen, ob der Selektorzugriff innerhalb des Layouts des Structs bleibt. Falls dies nicht der Fall ist, soll die Instruktion *sysc* ausgeführt werden, was einen Aufruf an das Betriebssystem zur Folge hat.

Aufgabe ZA1: (PPC Unsigned Divide) **(5+10 Punkte)**

Bonus Aufgabe

Sie sollen jetzt ein Assembler Programm angeben, dass Division nach Schulmethode implementiert und dabei einen neuen Schaltkreis benutzt. Der Schaltkreis zur Parallelen Prefix Berechnung ($PP_\circ$) ist ein sehr vielfältig einsetzbarer Schaltkreis. Konstruieren Sie zunächst einen Schaltkreis, der für $a, b \in \{0,1\}^{32}$ ein Resultat $r \in 0^{31}\{0,1\}^1$ liefert, so dass:

$$r[0] = 1 \Leftrightarrow \langle a \rangle \geq \langle b \rangle$$

Der Schaltkreis soll dabei jedes Bit gleichzeitig testen:

$$r = 1 \Leftrightarrow \forall i \in [0:31], a[i] \geq b[i]$$

Für binäre Division sind folgende Informationen hilfreich. Gehen Sie nach dem Prinzip der Tafelrechnung vor:

1. Merken Sie dich die Länge des Divisors in k.
2. Richten Sie den Divisor nach links aus, so dass die erste 1 des Divisors an der linkesten Stelle (des genau k langen Teils) des Dividenden steht.
3. Testen Sie ob der Divisor in den gleich langen Teil des Dividenden passt. Benutzen Sie dann den Befehl $fits\ rt\ ra\ rb$, der das Resultat des $PP_\circ$ Schaltkreises mit Registern ra und rb in Register rt schreibt.
4. Wenn ja subtrahieren Sie, wenn nein holen Sie eine weitere Stelle des Dividenden und fahren mit 2. und dem unveränderten Divisor fort.
5. Wenn subtrahiert wurde, fügen Sie an der entsprechenden Stelle von links eine 1 ein, ansonsten eine 0.
6. Sobald der Rest des Dividenden in der Länge kleiner als der Divisor wird, brechen Sie ab. Der verbleibende Teil des Dividenden ist der Rest der Division.

Aufgabe ZA2: (Assembler) **(5 Punkte)**

Bonus Aufgabe

Was macht folgendes Assemblerprogramm?

ori	R_6	R_0	1
and	R_7	R_4	R_6
bne	R_7	R_0	3
srl	R_4	R_4	1
beq	R_0	R_0	−3