Prof. Dr. Jan Reineke
Andreas Abel

Systemarchitektur SS 2018
Endsemesterklausur

1 Boolesche Ausdrücke und Synthese 3+3+6+4=16 Punkte

Die Boolesche Funktion f sei durch folgende Wertetabelle definiert.

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

1. Geben Sie einen Ausdruck in vollständiger konjunktiver Normalform an, der f beschreibt.
2. Geben Sie einen Ausdruck in vollständiger disjunktiver Normalform an, der f beschreibt.
3. Bestimmen Sie die Primimplikanten von f gemäß dem Algorithmus von Quine und McCluskey. Geben Sie explizit alle Zwischenschritte des Algorithmus an.
4. Zeigen Sie, dass in Booleschen Algebren die Regel des doppelten Komplements $x = \sim(\sim x)$ gilt. Greifen Sie ausschließlich auf axiomatische Umformungen zurück. Geben Sie in jedem Schritt an, welches Axiom Sie verwendet haben:
   A Assoziativität **K** Kommutativität **D** Distributivität **B** Absorption **O** Komplement
   Ansatz: $x \overset{\mathbf{O}}{=} x + (\sim x \cdot \sim(\sim x)) \overset{\mathbf{K}}{=} x + (\sim(\sim x) \cdot \sim x) \overset{\mathbf{D}}{=} (x + \sim(\sim x)) \cdot (x + \sim x) = \ldots$

2 Mikroarchitektur 2+6+4=12 Punkte

1. Erläutern Sie die Begriffe Befehlssatzarchitektur und Mikroarchitektur in jeweils einem Satz.
2. Betrachten Sie den Einzeltaktdatenpfad auf Ihrem Lösungsbogen. Vervollständigen Sie die dort angegebene Tabelle mit Decodersignalen für die Instruktionen `and`, `lw`, `sw` und `beq`.
   Hinweis: Es kann hilfreich sein, die Tabelle spaltenweise zu bearbeiten.
3. Nennen Sie zwei Beispiele für Hazards innerhalb einer Pipeline und beschreiben Sie die jeweilige Auswirkung auf die Ausführung. Beschreiben Sie knapp zwei Techniken um diese Auswirkungen zu reduzieren.

3 Schaltkreise - Teil 1 7+2+3=12 Punkte

1. Gegeben sei eine n-Bit-Sequenz a. Entwerfen Sie einen **rekursiven Schaltkreis** T_n mit drei Ausgabebits t_0, t_1 und $t_{>1}$, die folgende Werte haben sollen:

   - t_0 soll genau dann 1 sein, wenn in der Eingabe a keine 1 vorkommt.
   - t_1 soll genau dann 1 sein, wenn in der Eingabe a genau eine 1 vorkommt.
   - $t_{>1}$ soll genau dann 1 sein, wenn in der Eingabe a mehr als eine 1 vorkommt.

   Der Schaltkreis soll **logarithmische Tiefe** haben. Verwenden Sie nur die Gatter aus unserer Standard-Zellbibliothek $BIB = \{NOT_1, AND_2, OR_2, EXOR_2, NAND_2, NOR_2\}$. Sie können annehmen, dass n eine Zweierpotenz ist.

   (a) Entwickeln Sie zunächst einen Schaltkreis T_1 für eine 1-Bit lange Eingabe a.

   (b) Geben Sie nun eine rekursive Konstruktionsvorschrift an, um aus einem oder mehreren n-Bit Schaltkreisen T_n sowie zusätzlichen Gattern einen $2n$-Bit Schaltkreis T_{2n} zu konstruieren.

2. Erläutern Sie, warum die sechs Gatter unserer Zellbibliothek BIB ausreichend sind, um beliebige Boolesche Funktionen zu implementieren.

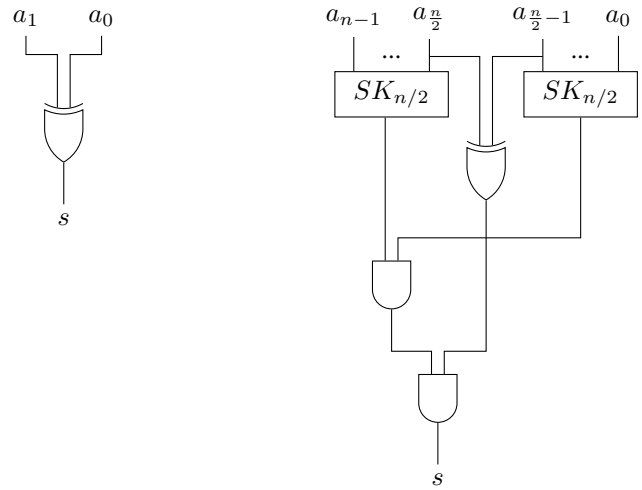
3. Wir möchten die Zellbibliothek durch die Menge $\{MUX_2\}$ ersetzen. Können wir damit immer noch alle Booleschen Funktionen implementieren?

Hinweis: Sie dürfen 0 und 1 als Eingabe für Gatter verwenden.

4 Schaltkreise - Teil 2 5+2+2+3=12 Punkte

Betrachten Sie den folgenden Schaltkreis. Sie können annehmen, dass n eine Zweierpotenz ist.

SK_2: SK_n:



1. Beschreiben Sie, welche Funktion der Schaltkreis berechnet. Geben Sie für $n = 8$ jeweils ein Beispiel an, für das SK_8 1 bzw. 0 ausgibt.

2. Geben Sie die Kosten K und die Tiefe T des Schaltkreises in rekursiver Form an.

3. Finden Sie eine Formel T' in geschlossener Form für die Tiefe.

4. Beweisen Sie die Äquivalenz von T und T'.

5 Scheduling 4+3+3+4=14 Punkte

1. Eine Variante des Multi-Level-Feedback-Queue Scheduling stützt sich auf folgende fünf Regeln.

   1: Priorität(A) $>$ Priorität(B), führe A aus.
   2: Priorität(A) $=$ Priorität(B), Round-Robin zwischen A und B.
   3: Neue Jobs erhalten höchste Priorität.
   4: Nutzt ein Job seine Zeitscheibe vollständig aus, wird seine Priorität reduziert. Ansonsten bleibt seine Priorität erhalten.
   5: Setze die Priorität eines Jobs nach S Zeiteinheiten zurück auf die höchste Stufe.

   Erläutern Sie knapp jeweils die Absicht hinter Regel 4 und 5. Regel 4 ist problematisch. Erläutern Sie knapp ein Problem von Regel 4 und geben Sie eine verbesserte Variante von Regel 4 an.

2. Nennen Sie drei Elemente eines Prozesskontrollblocks auf einem System ohne Speichervirtualisierung.

3. In der Vorlesung haben Sie gesehen, dass STCF optimal bezüglich der durchschnittlichen Umlaufzeit ist. Zeigen Sie anhand eines Beispiels: STCF ist **nicht** optimal bezüglich der durchschnittlichen Antwortzeit.

4. Entwickeln Sie ein Planungsverfahren, das optimal bezüglich der durchschnittlichen Antwortzeit ist.
   Beachten Sie, dass innerhalb einer Zeiteinheit kein Jobwechsel stattfinden kann, sondern lediglich zum Beginn einer Zeiteinheit.
   Sie brauchen die Optimalität Ihres Verfahrens nicht zu beweisen.

6 Systemprogrammierung 10+2+2=14 Punkte

1. In der Vorlesung haben Sie *Memory-Mapped I/O* als Technik zur Kommunikation mit externen Geräten kennengelernt. In dieser Aufgabe betrachten wir zwei externe Geräte: eine Tastatur und einen Bildschirm. Jedes dieser Geräte hat zwei Ports: einen Kontrollport und einen Datenport. Das unterste Bit des Kontrollports, genannt *ready*-Bit gibt an, ob das Gerät bereit zur Kommunikation ist. Das zweite Bit des Kontrollports, genannt *interrupt enable*-Bit, gibt an, ob das Gerät eine Unterbrechung generieren soll sobald das Gerät bereit wird. Das untere Byte des Tastatur-Datenports hält das letzte getippte Zeichen bereit. Das untere Byte des Bildschirm-Datenports kann mit dem nächsten auszugebenden Zeichen befüllt werden, wenn das Gerät bereit ist. Alle anderen Bits haben einen undefinierten Wert. Die Adressen der jeweiligen Ports finden Sie in der folgenden Tabelle:

	Kontrollport	Datenport
Tastatur	`0xffff0000`	`0xffff0004`
Bildschirm	`0xffff0008`	`0xffff000c`

   Schreiben Sie ein MIPS-Assembler-Programm, das zuerst zwei Zeichen von der Tastatur liest, und dann das Zeichen mit dem größeren ASCII-Code wieder auf dem Bildschirm ausgibt. Danach soll das Programm wieder zum Anfang springen. Verwenden Sie zur Kommunikation mit den externen Geräten Polling.

2. Beschreiben Sie kurz den Unterschied zwischen Interrupts und Polling. Nennen Sie einen Vorteil von Interrupts gegenüber Polling.

3. Schreiben Sie einen Bootup-Code, der mittels der `eret` Instruktion zu einem User-Programm springt, das an Label `task1` beginnt.

Hinweis: Auf der letzten Seite finden Sie Informationen zum MIPS-Befehlssatz sowie zu den Coprocessor-Registern.

7 Querbeet 2+2+2+2+2+2+3+2+1+1+1=20 Punkte

1. Geben Sie zwei Unterschiede zwischen SRAM und DRAM an sowie jeweils einen typischen Anwendungsfall.
2. Erläutern Sie, wieso Caches effektiv sind. Wieso wird normalerweise eine hohe Hit-Rate erreicht, obwohl der Cache im Vergleich zum Hauptspeicher klein ist?
3. Was ist der Unterschied zwischen Benutzer- und Kernelmodus und wozu trifft man diese Unterscheidung?
4. Beschreiben Sie kurz einen Vorteil und einen Nachteil von Paging im Vergleich zu Segmentierung.
5. Betrachten Sie die Dezimalzahl $5311{,}75$. Wie viele Nachkommabits werden für eine binäre Festkommakodierung benötigt? Geben Sie die Kodierung der Nachkommastellen an.
6. Wie erweitert man eine n-Bit Zahl im Zweierkomplement auf $n+1$ Bits ohne ihren Wert zu verändern? Wie erreicht man das Gleiche bei einer Zahl in Betrag-Vorzeichen-Darstellung?
7. Zeichnen Sie einen Moore-Automaten, der das Muster 110 erkennt. Das Ein-/Ausgabealphabet ist jeweils durch $\{0, 1\}$ gegeben. Die Ausgabe soll genau dann 1 sein, wenn die letzten drei Zeichen das obige Muster ergeben.
8. Nennen Sie einen Vor- und einen Nachteil des Carry-Ripple-Addierers gegenüber dem Conditional-Sum-Addierer.
9. Bestimmen Sie die Hamming-Distanz zwischen 010110 und 110011.
10. Ein Code c von fester Länge ist genau dann k-fehlererkennend, wenn

    ☐ $dist(c) \geq k + 1$

    ☐ $dist(c) \geq \left\lceil \frac{k^2}{2} \right\rceil$

    ☐ $dist(c) \geq 2^k - 1$

11. Ein Code c von fester Länge ist genau dann k-fehlerkorrigierend, wenn

    ☐ $dist(c) \geq 2k + 1$

    ☐ $dist(c) \geq 2^k - 1$

    ☐ $dist(c) \geq \Sigma_{n=1}^{k} \lceil log_2(n) \rceil$

Hilfsmaterial zur MIPS-ISA

Befehlsreferenz: Die Registernummern rs, rt und rd ergeben sich wie folgt aus einer Instruktion i :
rs $= i[25:21]$, rt $= i[20:16]$, rd $= i[15:11]$.

add/addu/sub/subu	rd, rs, rt	Addiere rs und rt/subtrahiere rt von rs; Ergebnis nach rd
addi/addiu	rt, rs, imm	Addiere rs und imm, Ergebnis nach rt
sra	rd, rt, shamt	Schiebe rt arithmetisch nach rechts um shamt, Ergebnis nach rd.
srav	rd, rt, rs	Schriebe rt arithmetisch nach rechts um rs, Ergebnis nach rd.
divu	rs, rt	Dividiere rs durch rt. Speichere Quotient in lo, Rest in hi
multu	rs, rt	Multipliziere rs mit rt. Speichere Ergebnis in hi und lo
slt/sltu	rd, rs, rt	Setze rd auf 1 wenn rs $<$ rt, sonst auf 0
slti/sltiu	rt, rs, imm	Setze rt auf 1 wenn rs $<$ imm, sonst auf 0
lui	rt, imm	Speichere imm um 16 bit nach links geschoben in Register rt und setze untere 16 bit auf 0
beq	rs, rt, label	Verzweige nach label wenn rs=rt
bne	rs, rt, label	Verzweige wenn rs $\neq$ rt
sll/srl	rd, rt, shamt	Schiebe rt logisch um shamt nach links/rechts, Ergebnis nach rd
sllv/srlv	rd, rt, rs	Schiebe rt logisch um rs nach links/rechts, Ergebnis nach rd
and/nor/or/xor	rd, rs, rt	Verknüpfe rs und rt, Ergebnis nach rd
andi/ori/xori	rt, rs, imm	Verknüpfe rs und die expandierte Konstante, Ergebnis nach rt
lb/lbu/lh/lhu/lw	rt, address	Lade das Byte/Halbwort/Wort an address aus dem Speicher in das Register rt. (Bei u-Varianten ohne Vorzeichenerweiterung)
sb/sh/sw	rt, address	Schreibe Byte/Halbwort/Wort aus Register rt nach adress
mfhi/mthi/mflo/mtlo	rd	Kopiere hi/lo-Register nach rd und umgekehrt
j/jal	target	Bedingungsloser Sprung, bei jal zusätzlich Adresse des nächsten Befehls nach $ra
jr/jalr	rs, rd	Springe zur in rs gespeicherten Adresse, bei jalr zusätzlich Adresse des nächsten Befehls nach rd
mfc0/mtc0	rg rc	Bewege Wert von/nach Register rc des Coprocessors 0 nach/von Register rg des Prozessors
syscall		Führe einen Systemaufruf aus
eret		Kehre von Ausnahmebehandlung zurück
Pseudoinstruktionen:		
li	rdest, imm	Lade imm in Register rdest
b	label	Bedingungslose Verzweigung
la	rdest, address	Lade address nach rdest

Exception Program Counter *epc* ist CP0-Register 14 und kann gelesen und geschrieben werden.

Status Register *status* ist CP0-Register 12 und kann gelesen und geschrieben werden. Für uns sind lediglich die folgenden Bits relevant.
Bit 0: *Interrupt Enable IE* gibt an, ob Ausnahmen global erlaubt sind.
Bit 1: *Exception Level EXL* gibt an, ob momentan eine Ausnahme behandelt wird. Während der Ausnahmebehandlung befindet sich der Prozessor im Kernel Mode.
Bit 4 - Bit 3: *KSU* gibt an in welchem Mode sich das System außerhalb der Ausnahmebehandlung befindet. Wir nehmen hier vereinfachend an, dass dies immer der *user mode* (Benutzermodus) (Bits 10) ist.
Bit 15 - Bit 10: *Interrupt Mask IM[7] - IM[2]* gibt an ob auf Unterbrechungsanfragen von externen Geräten reagiert werden soll.

Cause Register *cause* ist CP0-Register 13 und kann *nur gelesen* werden. Für uns sind lediglich die folgenden Bits relevant.
Bit 6 - Bit 2: *ExcCode* gibt den Grund für die Ausnahme an.
Bit 15 - Bit 10: *Interrupt Pending IP[7] - IP[2]* gibt an ob eine aktuelle Unterbrechungsanfrage eines externen Gerätes besteht.