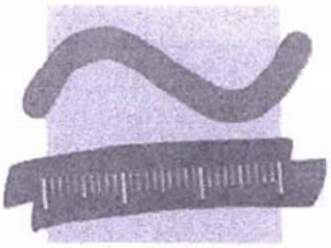
Klausur Technische Grundlagen der Informatik

Februar 2006 **Dr. Wolff**



Name: [redacted]

Zulässiges Material: Tabelle der booleschen Funktionen, Zahlentabelle 0–16 (binär/oktal/dezimal/hexadezimal), KV-Diagramm-Formular, Übersicht Intel-Befehlssatz.

Dauer: 90 Minuten. Aufgabenblatt bitte mit Name beschriften und als Deckblatt mit abgeben. Reklamationen nur bei Rückgabe. Bei Nichtabholung 1 Punkt Abzug.

Aufgabe 1

Punkte: 1 ½ 0

Wandeln Sie die Hexadezimalzahl 1C2A91 in das Oktalsystem.

Aufgabe 2

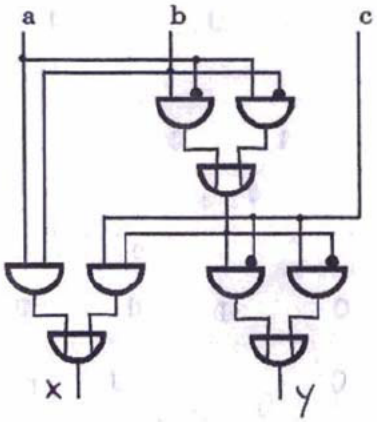
Punkte: 1 ½ 1 ½ 0

Berechnen Sie die folgenden Ausdrücke. Verwenden Sie die Zweierkomplementcodierung mit 8 Bit (7 Bit + 1 Bit Vorzeichen). Markieren Sie deutlich die **binären** Ergebnisse:

- $4B_{16} - 5E_{16}$
- $-4B_{16} - 5E_{16}$

Aufgabe 3

Punkte: 2 1 ½ 1 ½ 0



Analysieren Sie die abgebildete Schaltung. Stellen Sie die Wertetafel für die Ausgänge auf. Wie heißt diese Ihnen bekannte Schaltfunktion? Welche Schaltzeit hat die Schaltung bei einer Gatterdurchlaufzeit von $t_G = 2{,}3$ ns?

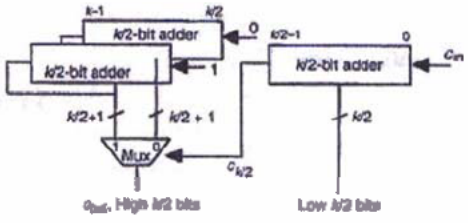
Aufgabe 4

Punkte: 1 ½ 1 ½ 0

Konstruieren Sie eine optimale (möglichst schnelle und einfache) Schaltung für die Schaltfunktion aus Aufgabe 3.

Aufgabe 5

Punkte: 1 ½ 1 ½ 0



$$C_{select-add}(k) = 3C_{add}(k/2) + k/2 + 1$$

$$T_{select-add}(k) = T_{add}(k/2) + 1$$

Der Conditional-Sum-Addierer berechnet in der ersten Ebene für jede Stelle die Summe doppelt, nämlich in jeweils einer Variante für den Fall, dass der Übertrag aus der vorherigen Stelle 0 bzw. 1 ist. Hierfür wird jeweils ein 2-stufiger Halbaddierer benötigt. Es entstehen Zwischenergebnisse der Länge 1 Bit.

In jeder weiteren, ebenfalls 2-stufigen Ebene werden jeweils 2 Zwischenergebnisse der Länge k Bit zu einem neuen Zwischenergebnis der Länge 2k Bit zusammengefasst (s. Bild, Mux), wobei aus den zuvor ermittelten Ergebnisvarianten diejenigen ausgewählt werden, die zu den nunmehr ermittelten Überträgen passen. Alle Ebenen seien 2-stufig realisiert.

In jedem Schritt entstehen also Zwischenergebnisse der doppelten Länge, bis zum Schluss das Gesamtergebnis in der vollen Länge berechnet wurde. Wie schnell berechnet der Conditional-Sum-Addierer mit $t_G = 2{,}3$ ns Zahlen der Längen 64 Bit bzw. 128 Bit?

Aufgabe 6

Punkte: 1 0

Beachten Sie die Rückseite!

Aufgabe 7

Punkte: 2 ½ 0

Wenn ein externer Baustein (z.B. Festplattencontroller) ein Signal auf die Interruptleitung legt, wer sorgt dafür, dass die CPU dem Beachtung schenkt und die Interruptroutine aktiviert wird? (Bitte markieren.)

- Der Controller (z.B. Festplattencontroller)
- Ein Interruptcontroller (der auch Interrupts von **verschiedenen** Bausteinen koordiniert)
- Das Steuerwerk der CPU
- Das Programm (vom Systemprogrammierer zu berücksichtigen)

Aufgabe 8

Punkte: 1 ½ 0

Auf welche Weise trägt die dargestellte Interruptroutine dazu bei, dass andere Programme durch ihre Aktivierung nicht im Ablauf gestört werden? Was fehlt?

```
isrtimer:
    push AX
    mov AX,counter1
    mov BX,counter2
    add AX,BX
    mov counter1,AX
    pop AX
    iret
```

Aufgabe 9

Punkte: 2 1½ 1 ½ 0

Wer trägt dazu bei, dass die Berechnungen der Interruptroutine das unterbrochene Programm nicht stören?

- Programmierer der Interruptroutine: muss zum Zeitpunkt der Rückkehr (`iret`) den vorherigen Registerzustand wiederhergestellt haben
- Interruptcontroller bzw. der den Interrupt verursachende Baustein: darf während des Interrupts keinen weiteren Interrupt anfordern
- CPU: muss die Bearbeitung des Interrupts mit der Befehlsabarbeitung so synchronisieren, dass dieser nach Rückkehr aus der Interrupt unbeeinträchtigt fortgesetzt bzw. wiederholt werden kann
- CPU: muss Programmzähler retten, bevor sie die Interruptroutine aktiviert
- CPU: muss sämtliche Register retten, bevor sie die Interruptroutine aktiviert
- CPU: muss den Cache leeren, bevor sie die Interruptroutine aktiviert
- Programmierer des Hauptprogramms: muss gelegentlich anhand eines Flags feststellen, ob es einen Interrupt gab, und seinen Zustand entsprechend anpassen
- Programmierer des Hauptprogramms: darf den Stack nur wie vorgesehen verwenden

Aufgabe 10

Punkte: 1½ 1 ½ 0

Geben Sie jeweils an, welche Komponente eines Rechners die beschriebenen Aufgaben ausführt. Wählen Sie dabei aus folgenden Komponentennamen: ~~ALU~~, CPU, CDU, ~~MMU~~, ~~FPU~~, Cache

- Hole nächsten Befehl aus dem Speicher Cache
- Erhöhe Programmzähler nach der Befehlsabarbeitung ~~CPU~~ ~~CPU~~ CDU
- 3+5 ALU
- 3.7+2.8 FPU
- Rette bei Interruptbearbeitung Rücksprungadresse auf dem Stack ~~CDU~~ ~~CDU~~ CPU
- Ermittle anhand einer vom Programm verwendeten Adresse, wo sich das Daten- bzw. Befehlswort befindet MMU

Aufgabe 11

Punkte: 2 1½ 1 ½ 0

Die CPU-Komponente, die den „virtuellen Speicher" verwaltet, bildet mit Hilfe einer Tabelle Programmadressen auf physische Adressen ab. (Zur einfachen Handhabung sind dabei Speicherblöcke fester Länge von 2^n Byte definiert, sogenannte „Seiten" je z.B. 4KB.) Diese „Seitentabelle" kann neben Speicheradressen noch weitere Information enthalten. Listen Sie diese auf (jeweils max. 1 Zeile):

- nicht zugeordnete, ungültig
- temporär nicht im Arbeitsspeicher (swap)
- dirty bit
-

Bewertung	Punkte	14	13	12	11	10	9	8	7	6	5	4
	Note	1,0	1,3	1,7	2,0	2,3	2,7	3,0	3,3	3,7	4,0	5,0

[redacted]

Aufg. 1

$1C2A91_{16}$

$0001|1100|0010|1010|1001_2$

7025221_8 ✓

Aufg. 2

a)

$4B_{16}$ − $5E_{16}$

01001011 (64, 8, 2, 1)

$ZK(01011110)$ (64, 16, 8, 4, 2)

$= 10100001 + 1$

$= 10100010$

-19_{10}

$$\begin{array}{r} 01001011 \\ +\ 10100010 \\ \hline 11101101 \end{array}$$ ✓

Kontrolle

$ZK(11101101)$

$= 00010011$

b) $-4B_{16}$ − $5E_{16}$

01001011 01011110

ZK

10110101 10100010

~~[illegible]~~

$$\begin{array}{r} 10110101 \\ +\ 10100010 \\ \hline 01010111 \end{array}$$ ✓

→ Überträge wechseln!! Überlauf!! ✓

[redacted]

Aufg. 3

a	b	c	x	y
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

Es handelt sich um einen Volladdierer.

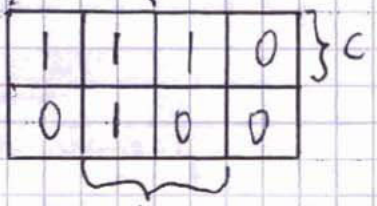
4-stufig also

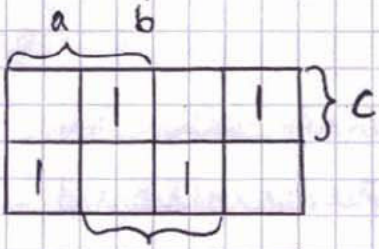
$t_g = 4 \cdot 2{,}3\,ns = \underline{9{,}2\,ns}$

Aufg. 4

KV

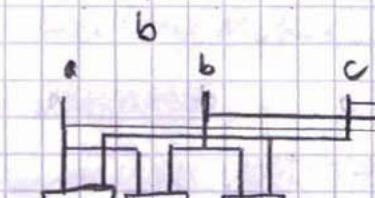
x





$ac + bc + ab$

y



$abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$

Nur 6 Gatter statt 9 und nur 2-stufig statt 4!

[redacted]

Aufg. 5

$t_{CSA} = t_q \cdot n \cdot ld(b)$ n stufen, b Bitlänge

64 $t_{CSA} = 2{,}3\,us \cdot 2 \cdot (6+1)$

$= 27{,}6\,us$

128 $t_{CSA} = 2{,}3\,us \cdot 2 \cdot (7+1)$ (U)

$= 32{,}2\,us$

Aufg. 6

:-)

Aufg. 7

s. Aufgabenblatt

Aufg. 8

[illegible]

Register AX wird mit „push AX" auf dem Stack gerettet. U

Es wäre sinnvoll BX ebenfalls zu retten. U nötig!

es fehlt also push BX