

Übungsblatt 5

Rechnerarchitektur im Sommersemester 2023

Zu den Modulen C, D, F, K

Abgabetermin: 28.05.23, 18:00 Uhr
Besprechung: 30.05.23 bis 02.06.23

Aufgabe Ü1: NAND/NOR

(6 Pkt.)

Die beiden Mengen $\{\text{NAND}\}$ und $\{\text{NOR}\}$ von Booleschen Funktionen sind funktional vollständig, d.h. dass sich durch die Kombination von NAND- bzw. NOR-Funktionen jede beliebige Boolesche Funktion darstellen lässt. Dies ermöglicht es, NAND- bzw. NOR-Gatter kostengünstig in Massenproduktion herzustellen und daraus beliebige digitale Schaltungen aufzubauen.

NAND			
i	a	b	$\overline{a \cdot b} = a \text{ NAND } b$
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

NOR			
i	a	b	$\overline{a + b} = a \text{ NOR } b$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Stellen Sie die elementaren Booleschen Funktionen AND, OR und NOT jeweils unter ausschließlicher Verwendung von

- NAND-Gattern
- NOR-Gattern

dar!

Aufgabe Ü2: PLA-Entwurf

(8 Pkt.)

In der Vorlesung haben Sie das Konzept von programmierbaren logischen Arrays (PLAs) kennen gelernt.

Wenn man in einem PLA die Anordnung der Bausteine so vornimmt, dass in der oberen Hälfte nur Bausteine vom Typ 0, 2 oder 3 und in der unteren Hälfte nur Bausteine vom Typ 0 oder 1 existieren – man das PLA also in eine Und- und eine Oder-Ebene unterteilen kann – spricht man auch von einem *normierten PLA*.

Gegeben sei die folgende Boolesche Funktion $f : B^3 \rightarrow B^2$

$$f(a, b, c) = (a \cdot c + b \cdot \bar{c}, \bar{a} + c \cdot a)$$

Realisieren Sie diese Funktion durch ein normiertes PLA, welches aus der minimal möglichen Anzahl an Zeilen und Spalten besteht. Verwenden Sie ausschließlich Bausteine der oben genannten Typen 0 bis 3. Kennzeichnen Sie in Ihrer Skizze die Und- und die Oder-Ebene. Markieren Sie gesperrte und neutralisierte Eingänge. Beschriften Sie jeden Pfeil (sowohl ausgehende als auch die innerhalb des PLAs) mit der jeweils anliegenden logischen Funktion.

Aufgabe Ü3: Cäsar-Verschlüsselung unter SPIM

(6 Pkt.)

Bearbeiten Sie die folgende Aufgabe zum Thema Assemblerprogrammierung unter SPIM.

Hinweis: Eine Übersicht der SPIM-Befehle finden Sie am Ende des Übungsblatts.

Im Folgenden soll ein MIPS-Assembler Programm vervollständigt werden, welches einen gegebenen Text mittels der **Caesar-Verschlüsselung** in einen Geheimtext umwandelt. Bei der Caesar-Verschlüsselung wird jeder Buchstabe im zu verschlüsselnden Text um eine vorher festgelegte Distanz im Alphabet verschoben. Ist z.B. die Distanz 3, so wird der Buchstabe A zum Buchstaben D, der Buchstabe B zum Buchstaben E, ..., der Buchstabe Z zum Buchstaben C.

Das folgende MIPS-Assembler Programm erwartet als Nutzereingabe die Distanz, um die die Buchstaben verschoben werden sollen und verschlüsselt dann einen gegebenen Text.

Ergänzen Sie den unten angegebenen Coderahmen um insgesamt **6 Zeilen Code**, so dass das Programm wie beschrieben funktioniert. Tragen Sie Ihre Lösung unter den mit “# Ihre Loesung:” markierten Stellen direkt in den folgenden Coderahmen ein:

```

1  .data
2
3  shift_text: .asciiz "Um wieviele Stellen soll der Text verschoben werden: "
4  string1: .asciiz "Der verschluesselte Text lautet: "
5  secret: .asciiz "GEHEIMNIS"
6  string_a: .asciiz "A"
7  string_z: .asciiz "Z"
8
9  result: .space 9
10
11 .text
12 main:
13     # t0 - Zum Zwischenspeichern der Position des aktuell betrachteten Buchstabens
14     # t1 - Gibt die Laenge des Geheimworts an
15     # t2 - ASCII Wert des Buchstaben A (65)
16     # t3 - ASCII - WERT des Buchstaben Z (90)
17     li $t0, 0
18     li $t1, 9
19     lb $t2, string_a
20     lb $t3, string_z
21
22     la $a0, shift_text      # String mit Anfangsadresse shift_text in $a0 laden
23     li $v0, 4               # 4 in $v0 laden
24     syscall                 # Text mit Anfangsadresse in $a0 auf der Konsole ausgeben
25
26     li $v0, 5               # 5 in $v0 laden
27     syscall                 # Zahl eingeben
28
29     move $s1, $v0           # Eingegebene Zahl in $s1 speichern
30
31 loop:  bge $t0, $t1, end     # Falls alle Buchstaben betrachtet wurden -> Springe end
32        lb $t4, secret($t0)   # Lade den aktuellen Buchstaben in $t4
33
34 caesar: #####
35          # Fügen Sie hier Ihre Loesung ein #

```

```

36 #####
37
38
39
40 #####
41 # Ende Ihrer Loesung #
42 #####
43 bgt $t4, $t3, cadd      # Falls das Ergebnis > Z --> springe zu cadd
44
45 save: #####
46 # Fuegen Sie hier Ihre Loesung ein #
47 #####
48
49
50
51
52
53
54 #####
55 # Ende Ihrer Loesung #
56 #####
57 j loop                  # Springe zum Label loop
58
59 cadd: #####
60 # Fuegen Sie hier Ihre Loesung ein #
61 #####
62
63
64
65
66
67
68
69 #####
70 # Ende Ihrer Loesung #
71 #####
72 j save                  # Springe zum Label save
73
74
75 end:   la $a0, string1    # Anfangsadresse des Strings string1 wird in $a0 geladen
76       li $v0, 4           # 4 wird in $v0 geladen
77       syscall            # String string1 wird auf der Konsole ausgegeben
78
79       la $a0, result      # Anfangsadresse des Strings result wird in $a0 geladen
80       li $v0, 4           # 4 wird in $v0 geladen
81       syscall            # String result wird auf der Konsole ausgegeben
82
83       li $v0, 10          # 10 wird in $v0 geladen
84       syscall            # Programm wird beendet

```

Aufgabe Ü4: Einfachauswahlaufgabe: Wiederholung

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Bei welcher Belegung (x_1, x_2, x_3) ergibt die Boolesche Funktion $f(x_1, x_2, x_3) = (x_1 \cdot \overline{x_2}) + (x_2 \cdot \overline{x_3})$ den Wert 1?			
(i) (0, 0, 1)	(ii) (0, 1, 0)	(iii) (1, 1, 1)	(iv) (0, 1, 1)
b) Was bewirkt der Spim-Befehl <code>li \$v0 5</code> :			
(i) Es wird ein Integer von der Konsole eingelesen	(ii) Es wird eine Zahl vom Typ double von der Konsole eingelesen	(iii) Der Wert 5 wird in das Register <code>\$v0</code> geladen	(iv) Es wird ein Integer auf der Konsole ausgegeben
c) Wofür steht CISC im Zusammenhang mit Mikroprozessoren?			
(i) Complex Instruction Set Computer	(ii) Complex Instruction Set Calculator	(iii) Controversy Instruction Set Computer	(iv) Constructive Instruction Set Computer
d) Welche Aussage ist korrekt? MIPS ist eine...			
(i) Last-in-First-Out-Architektur	(ii) Stack-Architektur	(iii) Load-Store-Architektur	(iv) Heap-Architektur
e) Welche Aussage ist falsch? Die Funktion <code>syscall</code> ...			
(i) beendet das Programm sofort	(ii) erwartet die Nummer der auszuführenden Funktion in <code>\$v0</code>	(iii) besitzt selbst keine Parameter	(iv) führt eine Funktion des Betriebssystems aus