

Rechnerarchitektur

Abgabetermin: 16.06.2008, 12:00 Uhr

Achtung: Am Mittwoch, dem 18. Juni (vorauss. 14:30 Uhr) findet im Rahmen der Vorlesung "Programmierung und Modellierung" eine Info-Veranstaltung zur Praktikumswahl statt. Informationen unter <http://www.nm.ifi.lmu.de/teaching/Praktika/2008ws/sysprak/>

Lesen: Rechnerarchitektur-Skript: Kapitel 13: Einführung in den SPIM Simulator. Programme und weitere Informationen zu SPIM sind auf der Vorlesungshomepage zu finden.

Aufgabe 23: (T) Minimierung mittels Karnaugh (6+6 Pkt.)

Ein **Karnaugh-Diagramm** einer Booleschen Funktion $f : B^n \rightarrow B$ mit $n \in \{3, 4\}$ ist eine graphische Darstellung der Funktionstafel von f durch eine 0-1-Matrix der Größe 2×4 für $n = 3$ bzw. 4×4 für $n = 4$, deren Spalten mit den möglichen Belegungen der Variablen x_1 bzw. x_1 und x_2 und deren Zeilen mit den möglichen Belegungen der Variablen x_3 bzw. x_3 und x_4 beschriftet sind. Die *Reihenfolge* der Beschriftung erfolgt dabei so, daß sich zwei zyklisch benachbarte Spalten oder Zeilen nur in genau einer Komponente unterscheiden. (Zyklisch benachbart heißt, daß auch obere und untere Zeile bzw. linke und rechte Spalte als benachbart angesehen werden.)

Die Beschriftung erfolgt zyklisch benachbart, damit der Resolutionssatz für Boolesche Funktionen einfach anwendbar ist:

$$a \cdot b + a \cdot \neg b = a$$

Beachte: Die Anordnung der Variablen ist frei, es muß lediglich die Nachbarschaftsbeziehung erfüllt sein und es müssen alle Kombinationen möglich sein. Zur späteren Identifikation der zu Resolutionsblöcken gehörenden Terme sollte die Beschriftung der Zeilen und Spalten wie folgt durchgeführt werden: Man bezeichne die Spalten bzw. Zeilen mit x , für welche die Variable x den Wert 1 annimmt, und die anderen mit $\neg x$.

Minimieren Sie folgende Funktionen im Karnaugh-Diagramm.

- a. $y_1 = (x_1 \cdot x_2 \cdot x_3) + (x_1 \cdot \neg x_2 \cdot x_3) + (\neg x_1 \cdot \neg x_2 \cdot x_3) + (\neg x_1 \cdot x_2 \cdot \neg x_3).$
- b. $y_2 = (x_1 \cdot x_2 \cdot \neg x_3) + (x_1 \cdot \neg x_2 \cdot \neg x_3) + (\neg x_1 \cdot \neg x_2 \cdot \neg x_3) + (\neg x_1 \cdot x_2 \cdot \neg x_3) + (x_1 \cdot \neg x_2 \cdot x_3) + (x_1 \cdot x_2 \cdot x_3).$

Aufgabe 24: (H) Don't Care-Argumente (8 Pkt.)

Betrachten sie folgende Funktion $y_3(x) = \begin{cases} 1 & \text{falls } x \in \{0, 1, 3, 4, 5, 8, 9\} \\ 0 & \text{falls } x \in \{2, 6, 7\} \end{cases}$

Zur Binärcodierung der Ergebniswerte verwenden wir (wie in der Vorlesung) vierstellige Dualzahlen. Damit könnten also 16 Argumente kodiert werden. Stellen Sie eine Funktionstabelle auf und minimieren Sie anschließend die Funktion mit Hilfe eines Karnaugh-Diagramms.

Aufgabe 25: (H) SPIM Programmieraufgabe

(12 Pkt.)

Erstellen Sie ein **vollständiges** SPIM-Programm, das folgendes durchführt:

- Es werden zwei positive Integer-Zahlen von der Konsole eingelesen.
- Es wird der Durchschnitt dieser beiden Zahlen auf eine Nachkommastelle genau berechnet.
- Das Ergebnis der Berechnung wird ausgegeben.

Tipp: Programmieren Sie diejenigen Schritte, die Sie auch beim handschriftlichen Dividieren durchführen!

Beachten Sie hierbei folgendes:

- Verwenden Sie nur die **unten aufgeführten Befehle**.
- Verwenden Sie für die Vorkommazahl das Register `$s0` und für die Nachkommazahl das Register `$s1`, ansonsten nur die temporären Register.
- **Kommentieren** Sie ihr Programm sinnvoll!
- Sowohl die Eingabe als auch die Ausgabe soll mit einem Anweisungstext versehen werden, wie z.B. *"Geben Sie die 1. Zahl ein: "*, etc.
- Die Ausgabe des Ergebnisses soll das Komma enthalten.

Aufgabe 26: (H) Quine–McCluskey–Verfahren

(6+2+1 Pkt.)

Eine Funktion von 6 Variablen ist durch folgende Vollkonjunktion gegeben:

$$y = k_8 \vee k_{24} \vee k_{25} \vee k_{26} \vee k_{27} \vee k_{35} \vee k_{56} \vee k_{57} \vee k_{58} \vee k_{59}$$

Für die Vollkonjunktionen k_j mit $j = 12, 28, 30, 40, 60, 62$ ist der Wert beliebig.

- Minimieren Sie die Funktion nach dem McCluskey-Verfahren.
- Geben Sie die Kern-Primimplikanten (= unverzichtbare Primimplikanten), die wählbaren und die unnötigen Primimplikanten an.
- Geben Sie zwei minimale Funktionen für y an.

Hinweis: Eine Vollkonjunktion k_j besteht aus einer Konjunktion aller x_i , bei der diejenigen x_i negiert sind, deren Stelle a_i in der Dualdarstellung von j Null ist ($j = \sum_{i=1}^6 a_i 2^{i-1}$).

Beispiel: $k_{25} = \bar{x}_6 \wedge x_5 \wedge x_4 \wedge \bar{x}_3 \wedge \bar{x}_2 \wedge x_1$

SPIM Assemblerbefehle

Befehl	Argumente	Wirkung
add	Rd, Rs1, Rs2	$Rd := Rs1 + Rs2$
addu	Rd, Rs1, Rs2	$Rd := Rs1 + Rs2$
addi	Rd, Rs1, Imm	$Rd := Rs1 + Imm$
addiu	Rd, Rs1, Imm	$Rd := Rs1 + Imm$
div	Rd, Rs1, Rs2	$Rd := Rs1 \text{ DIV } Rs2$
rem	Rd, Rs1, Rs2	$Rd := Rs1 \text{ MOD } Rs2$
mul	Rd, Rs1, Rs2	$Rd := Rs1 \times Rs2$
b	label	unbedingter Sprung nach label
j	label	unbedingter Sprung nach label
jal	label	unbed. Sprung nach label, Adresse des nächsten Befehls in \$ra
jr	Rs	unbedingter Sprung an die Adresse in Rs
beq	Rs1, Rs2, label	Sprung, falls $Rs1 = Rs2$
beqz	Rs, label	Sprung, falls $Rs = 0$
bne	Rs1, Rs2, label	Sprung, falls $Rs1 \neq Rs2$
bnez	Rs1, label	Sprung, falls $Rs1 \neq 0$
bge	Rs1, Rs2, label	Sprung, falls $Rs1 \geq Rs2$
bgeu	Rs1, Rs2, label	Sprung, falls $Rs1 \geq Rs2$
bgez	Rs, label	Sprung, falls $Rs \geq 0$
bgt	Rs1, Rs2, label	Sprung, falls $Rs1 > Rs2$
bgtu	Rs1, Rs2, label	Sprung, falls $Rs1 > Rs2$
bgtz	Rs, label	Sprung, falls $Rs > 0$
ble	Rs1, Rs2, label	Sprung, falls $Rs1 \leq Rs2$
bleu	Rs1, Rs2, label	Sprung, falls $Rs1 \leq Rs2$
blez	Rs, label	Sprung, falls $Rs \leq 0$
blt	Rs1, Rs2, label	Sprung, falls $Rs1 < Rs2$
bltu	Rs1, Rs2, label	Sprung, falls $Rs1 < Rs2$
bltz	Rs, label	Sprung, falls $Rs < 0$
syscall		führt Systemfunktion aus
move	Rd, Rs	$Rd := Rs$
la	Rd, label	Adresse des Labels wird in Rd geladen
lb	Rd, Adr	$Rd := \text{MEM}[\text{Adr}]$
lw	Rd, Adr	$Rd := \text{MEM}[\text{Adr}]$
li	Rd, Imm	$Rd := Imm$
sw	Rs, Adr	$\text{MEM}[\text{Adr}] := Rs$

Funktion	Code in \$v0	Funktion	Code in \$v0
print_int	1	read_float	6
print_float	2	read_double	7
print_double	3	read_string	8
print_string	4	sbrk	9
read_int	5	exit	10

Bemerkung:

Alle arithmetischen Befehle (mul, ...), Sprungbefehle und Vergleichsbefehle sind auch mit einem Imm-Argument (Immediate-Argument) statt Rs2 möglich. Zum Beispiel: `$t0, $t1, 5` statt `li $t5, 5` gefolgt von `mul $t0, $t1, $t5`. Dies funktioniert, da der Assembler dann den Wert zunächst in sein \$at-Register lädt.