64-041 Übung Rechnerstrukturen und Betriebssysteme

Aufgabenblatt 11 Ausgabe: 27.01., Abgabe: 03.02. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 11.1 (Punkte 10+10+10+10)

Hardwarearchitektur: Um zu zeigen, dass Algorithmen auch direkt als spezielle Hardware implementiert werden können, soll jetzt ein Spezialrechner für die Collatz-Vermutung[1] entworfen werden – für Details siehe zum Beispiel: de.wikipedia.org/wiki/Collatz-Problem. Man startet mit einem positiven ganzzahligem Eingabewert $X_0 > 0$ überprüft, ob der Wert gleich Eins ist. In diesem Fall endet die Berechnung. Ansonsten wird der nächste Wert X_{i+1} gemäß einer einfachen Iteration berechnet, nämlich $X_{i+1} = 3 * X_i + 1$ wenn X_i ungerade ist, und $X_{i+1} = X_i/2$ wenn X_i gerade ist. Die Iteration wird beendet, sobald der Wert $X_i = 1$ erreicht wird.

Die bis heute nicht endgültig bewiesene Vermutung besagt, dass der Wert 1 (bzw. die Folge $4, 2, 1$ für jeden beliebigen Eingabewert X_0 erreicht wird. Ein Beispiel für eine Folge ist: $11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1$.

(a) Entwerfen Sie eine Hardwarestruktur für die Berechnung der Collatz-Iteration für n-bit Zahlen. Man benötigt offenbar ein n-bit Register X zur Speicherung des aktuellen Werts X_i. Über einen Eingang A kann dieses Register dann mit dem Startwert X_0 initialisiert werden.

Überlegen Sie sich dazu ein möglichst effizientes Rechenwerk (ALU) für die Berechnung des Nachfolgewerts X_{i+1}. Außer den arithmetischen Operationen Addition und Subtraktion sind dabei natürlich auch alle Shift- und logischen Operationen zugelassen. Die ALU soll abhängig vom Eingabewert selbständig die jeweils notwendige Berechnung ausführen, also entweder $3 * X + 1$ oder $X/2$.

Außer dem eigentlichen Rechenergebnis sollen auch die zwei Statussignale *isOne* und *isOdd* ausgegeben werden. Wie berechnen Sie diese?

Zeichnen Sie Ihre Hardwarestruktur mit der vollständigen ALU und allen ggf. notwendigen Multiplexern. Hier ist ein Blockschaltbild gefragt, mit Elementen (ALU, Register, Multiplexer etc.) wie beispielsweise in den RSB-Unterlagen in Kapitel 11.2, Folie 733. Eine HADES Schaltung oder der interne Aufbau der ALU aus Gattern interessieren nicht.

[1] Prof. Collatz war lange Jahre Direktor des Instituts für Angewandte Mathematik an der Uni Hamburg.

(b) Entwerfen Sie das zugehörige Steuerwerk mit dem Startzustand *Init* (zum einmaligen Einlesen des Eingabewerts X_0) und dem Endzustand *One*, sowie allen weiteren Zuständen, die Sie benötigen. Der Automat bekommt das externe Steuersignal *Start* sowie die beiden Statussignale *isOne* und *isOdd* als Eingaben. Als Ausgaben hat der Automat die Steuerleitungen, die für die Hardware aus (a) benötigt werden, und einen Ausgabewert *Done*, um das Ende der Berechnung anzuzeigen. Zeichnen Sie das Zustandsdiagramm und geben Sie die logischen Gleichungen für das δ- und λ-Schaltnetz an.

(c) Schreiben Sie ein Java oder C Programm für die Collatz-Vermutung und erläutern Sie Ihren Algorithmus. Der Startwert X_0 wird als Kommandozeilenargument übergeben. Zählen Sie zusätzlich die Iterationsschritte i mit und geben Sie in jedem Schritt sowohl i, als auch das zugehörige Zwischenergebnis X_i aus.

(d) Für welchen Wert von $1 < X_0 < 100$ ergibt sich die längste Folge? Nehmen Sie auch Ihr Geburtsdatum (in der Form Jahr Monat Tag, also zum Beispiel 19990127) als Eingabe des Programms und geben Sie die Folge der Zwischenergebnisse an.

Aufgabe 11.2 (Punkte 3+3+3+3+3)

Adressierung: Auf einer 1-Adress Maschine (Akkumulatormaschine) werden Ladebefehle mit unterschiedlichen Adressierungsmodi ausgeführt. Der Speicher enthält folgende Werte:

Adresse	Inhalt
20	60
30	40
40	20
50	40
60	20
70	80
80	50

Welcher Wert steht jeweils nach Ausführung der folgenden Befehle im Akkumulator?

(a) `LOAD IMMEDIATE 30`

(b) `LOAD DIRECT 30`

(c) `LOAD INDIRECT 30`

(d) `LOAD DIRECT 50`

(e) `LOAD INDIRECT 80`

Aufgabe 11.3 (Punkte 4·10+5)

Befehlsformate: Vergleichen Sie 0-, 1-, 2- und 3-Adress Maschinen, indem Sie für jede Architektur ein Programm zur Berechnung des folgenden Ausdrucks schreiben. Dabei gilt (natürlich) Punkt- vor Strichrechnung:

$$R = (A + A * B)/(C^2 - D)$$

Die verfügbaren Befehle der entsprechenden Maschinen sind unten angegeben. `M` und `N` stehen dabei für 24-bit Speicheradressen, während `X`, `Y` und `Z` eine 5-bit Registernummer codieren. `MEM[M]` sei der Inhalt des Speichers an der Adresse `M`.

0-Adress Maschine mit einen unbegrenzten Stack (TOS „top of stack“)

Mnemonik	Bedeutung
`PUSH M`	push; TOS = MEM[M]
`POP M`	MEM[M] = TOS; pop
`ADD`	tmp = TOS; pop; TOS = tmp + TOS
`SUB`	tmp = TOS; pop; TOS = tmp − TOS
`MUL`	tmp = TOS; pop; TOS = tmp ∗ TOS
`DIV`	tmp = TOS; pop; TOS = tmp / TOS

1-Adress Maschine: Akkumulatormaschine mit genau einem Register

Mnemonik	Bedeutung
`LOAD M`	Akku = MEM[M]
`STORE M`	MEM[M] = Akku
`ADD M`	Akku = Akku + MEM[M]
`SUB M`	Akku = Akku − MEM[M]
`MUL M`	Akku = Akku ∗ MEM[M]
`DIV M`	Akku = Akku / MEM[M]

2-Adress Maschine: benutzt nur Speicheroperanden

Mnemonik	Bedeutung
`MOV M,N`	MEM[M] = MEM[N]
`ADD M,N`	MEM[M] = MEM[M] + MEM[N]
`SUB M,N`	MEM[M] = MEM[M] − MEM[N]
`MUL M,N`	MEM[M] = MEM[M] ∗ MEM[N]
`DIV M,N`	MEM[M] = MEM[M] / MEM[N]

3-Adress Register-Maschine: *load-store* RISC-Architektur, 32 Universalregister

Mnemonik	Bedeutung
`LOAD X,M`	X = MEM[M]
`STORE M,X`	MEM[M] = X
`MOV X,Y`	X = Y
`ADD X,Y,Z`	X = Y + Z
`SUB X,Y,Z`	X = Y − Z
`MUL X,Y,Z`	X = Y ∗ Z
`DIV X,Y,Z`	X = Y / Z

(a) Schreiben Sie für alle vier Maschinen (möglichst kurze) Programme für die Berechnung von $R = (A + A * B)/(C^2 - D)$. Dabei stehen $A \ldots D$ und R für Speicheradressen der Operanden bzw. des Resultats. Zwischenergebnisse können (bei Bedarf) auf ungenutzten Speicheradressen ($E \ldots Q$) abgelegt werden.

(b) Wenn die Befehlscodierung jeweils 8-bit für den Opcode verwendet, 24-bit für eine Speicheradresse und 5-bit für eine Registernummer, wie viele Bits werden dann für jedes der obigen vier Programme benötigt?

Welche Maschine hat also die kompakteste Codierung (gemessen an der Programmgröße in Bits) für dieses Programm?