Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Lehrstuhl für Technische Informatik
Prof.Dr. Manfred Schimmler
Dr.-Ing. Christoph Starke, M.Sc. Vasco Grossmann

Digitale Systeme
Wintersemester 2014/2015

Serie 12

Ausgabetermin: Mittwoch, 28.01.2015
Abgabetermin: Montag, 09.02.2015, 08:00 Uhr im Schrein

Rückgabe der Korrekturen zu Serie 12 am Mi, 11.2., 14-15 Uhr in HRS3, R510.

Bitte notieren Sie Ihre Namen sowie Ihre Gruppennummer auf der Abgabe!

Wichtiger Hinweis: Bei jeder Assembler-Programmieraufgabe gilt, auch wenn nicht ausdrücklich dazu aufgefordert wird:

(a) **Beschreiben Sie Ihr Programm ausführlich und geben Sie separat die Registerbelegung an.**

(b) **Kommentieren Sie Ihr Programm ausführlich.**

(c) **Sie können die folgende URL benutzen, um Ihre Lösungen zu prüfen:**
`http://huesersohn.de/cau/dlx/`

(d) **Fertigen Sie zusätzlich zu Ihrer schriftlichen Abgabe im Schrein für jede Assembler-Programmieraufgabe eine Text-Datei an, die Ihre Lösung enthält, und senden Sie diese bis zum angegebenen Abgabetermin per E-Mail an Ihre/n Übungsleiter/in.**

Präsenzaufgaben

Schreiben Sie ein DLX-Assemblerprogramm, welches den Inhalt des Registers R2 modulo 15 berechnet und im Register R3 ablegt. Der Inhalt von R2 soll dabei nicht verändert werden. Es dürfen keine Gleitkommaregister (und damit u.a. auch nicht die Befehle MULT und DIV) benutzt werden.

Hausaufgaben

Aufgabe 1 - Kurzfragen

(a) Der DLX-Befehl *SLL* entspricht welcher mathematischen Funktion?

(b) Geben Sie das Befehlsformat eines *immediate*-DLX-Befehls an und erläutern sie kurz die einzelnen Bestandteile. Gehen Sie dabei bitte auch auf die Binärcodierung ein.

$2^1/_2$, $2^1/_2$ Punkte

Aufgabe 2

Die Fibonacci-Folge $(1; 1; 2; 3; 5; 8; ...)$ ist eine mathematische Folge ganzer Zahlen, bei der die ersten beiden Folgenglieder 1 sind und alle weiteren Folgenglieder sich aus der Summe der beiden jeweils vorhergehenden Folgenglieder ergeben. Die n-te Fibonacci-Zahl $f(n)$ mit $n < N$ lässt sich also folgendermaßen berechnen:
$f(1) = 1$
$f(2) = 1$
$f(n) = f(n-1) + f(n-2)$, falls $n > 2$

Schreiben Sie ein Assemblerprogramm, das alle Fibonacci-Zahlen $< 2^{31}$ in aufsteigender Reihenfolge in den Speicher ab Adresse 1000 schreibt. Achten Sie dabei auf einen korrekten Umgang mit dem Zahlenbereich, insbesondere auch darauf, dass kein Element der Fibonacci-Folge ungewollt als negative Zahl interpretiert wird. In Register 1 soll am Ende das zum letzten gespeicherten Folgenglied zugehörige n stehen, also dasjenige n für das gilt $f(n) < 2^{31} \leq f(n+1)$.

25 Punkte

Aufgabe 3

Ein Automat $A = (X, Y, Z, \delta, \lambda)$, $X = \{0, 1\}$, $Y = \{0, 1\}$, $Z = \{Z_0, Z_1, Z_2\}$ mit Codierung 00 für Z_0, 01 für Z_1 und 10 für Z_2, soll wie folgt in den Registern eines DLX-Prozessors abgebildet sein:
Für jeden Zustand stehen zwei Register zur Verfügung, eines für jede mögliche Eingabe. Dies sind die Register R(2i+1) und R(2i+2) für Z_i mit $0 \leq i < |Z|$. In beiden niederwertigsten Bits dieser Register sind die Folgezustände eingetragen (niederwertigstes Bit Register = niederwertigstes Bit der Zustandskodierung, zweitniederwertigstes Bit Register = höchstwertigstes Bit der Zustandskodierung). Im drittniederwertigsten Bit des Registers steht die Ausgabe für diese Transition. Das Register R(2i+1) macht diese Angaben hierbei für die Eingabe 0, das Register R(2i+2) für die Eingabe 1.
In R10 steht ein vom niederwertigsten Bit an abzuarbeitender Binärstring von Eingabewerten. In R9 steht die Länge dieses Eingabestrings $L < 32$ (d.h. es sind L Bit von R10 als gültige Eingaben zu betrachten). Der jeweils aktuelle Zustand soll in R8 gespeichert werden.

(a) Zeichnen Sie den zu der Registerbelegung R1 = 5, R2 = 2, R3 = 5, R4 = 0, R5 = 1, R6 = 5 gehörenden Automatengraphen.

(b) Ermitteln Sie anhand des Automatengraphen für R8 = 0, R9 = 4, R10 = 7 den Endzustand und die Ausgabefolge.

(c) Schreiben Sie ein DLX-Assemblerprogramm, das für jede beliebige Belegung sowohl der Register, die für die Zustände zur Verfügung stehen als auch von R9 und R10 die Ausgaben von λ in richtiger Reihenfolge als Binärstring in R11 ablegt (letzte Ausgabe auf dem niederwertigsten Bit).

(d) Zeigen Sie für das in a) und b) genannte Beispiel schrittweise, wie Ihr Programm die Ausgabefolge berechnet.

10, 10, 30, 20 Punkte

Anhang: DLX-Assembler Befehlssatz

Die Befehle werden in der Form *Instr. / Ziel / Quelle(n)* verwendet.
Bsp: ADDI R3 R2 #15 ≈ R3:=R2+15

Instr.	Description	Format	Operation (C-style coding)
ADD	add	R	Rd = Rs1 + Rs2
ADDI	add immediate	I	Rd = Rs1 + extend(immediate)
AND	and	R	Rd = Rs1 & Rs2
ANDI	and immediate	I	Rd = Rs1 & extend(immediate)
BEQZ	branch if equal to zero	I	PC += (Rs1 == 0 ? extend(immediate) : 4)
BNEZ	branch if not equal to zero	I	PC += (Rs1 != 0 ? extend(immediate) : 4)
J	jump	J	PC += extend(immediate)
JAL	jump and link	J	R31 = PC + 4 ; PC += extend(immediate)
JALR	jump and link register	I	R31 = PC + 4 ; PC = Rs1
JR	jump register	I	PC = Rs1
LW	load word	I	Rd = MEM[Rs1 + extend(immediate)]
OR	or	R	Rd = Rs1 | Rs2
ORI	or immediate	I	Rd = Rs1 | extend(immediate)
SEQ	set if equal	R	Rd = (Rs1 == Rs2 ? 1 : 0)
SEQI	set if equal to immediate	I	Rd = (Rs1 == extend(immediate) ? 1 : 0)
SLE	set if less than or equal	R	Rd = (Rs1 <= Rs2 ? 1 : 0)
SLEI	set if less than or equal to immediate	I	Rd = (Rs1 <= extend(immediate) ? 1 : 0)
SLL	shift left logical	R	Rd = Rs1 << (Rs2 % 32)
SLLI	shift left logical immediate	I	Rd = Rs1 << (immediate % 32)
SLT	set if less than	R	Rd = (Rs1 < Rs2 ? 1 : 0)
SLTI	set if less than immediate	I	Rd = (Rs1 < extend(immediate) ? 1 : 0)
SNE	set if not equal	R	Rd = (Rs1 != Rs2 ? 1 : 0)
SNEI	set if not equal to immediate	I	Rd = (Rs1 != extend(immediate) ? 1 : 0)
SRA	shift right arithmetic	R	as SRL & see below
SRAI	shift right arithmetic immediate	I	as SRLI & see below
SRL	shift right logical	R	Rd = Rs1 >> (Rs2 % 32)
SRLI	shift right logical immediate	I	Rd = Rs1 >> (immediate % 32)
SUB	subtract	R	Rd = Rs1 - Rs2
SUBI	subtract immediate	I	Rd = Rs1 - extend(immediate)
SW	store word	I	MEM[Rs1 + extend(immediate)] = Rd
XOR	exclusive or	R	Rd = Rs1 ^ Rs2
XORI	exclusive or immediate	I	Rd = Rs1 ^ extend(immediate)

Beachten Sie: Die Befehle SRA und SRAI füllen die vorderen Bits des Registers mit dem aktuellen Vorzeichenbit auf.