04. Dezember 2013

Rechnerstrukturen im WS 2013/2014
Übungsblatt 8 (Block B-4)

Zum Verständnis von Rechnerarchitekturen sind Erfahrungen im Umgang mit einem Assembler unerlässlich. Sie werden auf diesem und den folgenden Übungsblättern eine Reihe von Assembler-Aufgaben finden. Um sie zu lösen, müssen Sie den in der Vorlesung vorgestellten Simulator installieren und mit ihm arbeiten. Sie finden eine vollständige Simulatorsoftware unter:
http://courses.missouristate.edu/KenVollmar/MARS/index.htm (in der Vorlesung empfohlen)

Installieren Sie den Simulator auf Ihrem Rechner.
Wählen Sie unter „File“ „New“ und geben Sie im Editorfeld das nachfolgende Assemblerprogramm ein. Speichern Sie ihre Eingabe als Datei. (z.B. Blatt8A1.asm)

```
.data              # 01 die Zeilenzahl ist zum
x: .word 4711      # 02 Besprechen in der Übung
y: .word 10        # 03
z: .word 0x0a94    # 04
e: .word 0         # 05 Ergebnisvariable
                   # 06
.text              # 07
.globl main        # 08
main:              # 09
      lw  $2,x     # 10
      lw  $3,y     # 11
      lw  $4,z     # 12
      add $2,$2,$3 # 13
      sub $3,$2,$4 # 14
      sw  $3,e     # 15
      li  $2,10    # 16 Programm ordnungsgemaess beenden
      syscall      # 17
```

Lassen Sie das Programm assemblieren (F3). Ihr Programm erscheint nun unter „Execute“ im Textsegment. Es beginnt mit der Zeile: lui $1,0x1001 (unter „Basic“). Ihr Eingabetext steht unter „Source“.
Lassen Sie das Programm schrittweise ablaufen (F7). Achten Sie bei jedem Schritt auf Veränderungen in den Registern 2 bis 4 (rechts unter Register „Number“ und „Value“).

Aufgabe 1 (Simulatoroberfläche) (4 Punkte)

a) Nach der Abarbeitung Ihres Programms erscheint unter „`Run I/O`“ -- program is finished running --. Welcher Wert steht im Register $3? Geben Sie dieses Ergebnis ebenfalls als Dezimalzahl an.

b) Was berechnet das Programm im Register Reg[3] (bezogen auf die Variablen x,y,z)? Geben Sie sowohl die Register-Transfer-Notation als auch die konkreten Werte an.

c) Die erste Programmzeile im Simulator lautet: lui $1,0x1001 (unter „Basic“). Woher kommt die 0x1001?

d) Wo findet man nach Abarbeitung von Zeile 15 das Ergebnis im Speicher? Geben Sie den Bereich der Simulatoroberfläche und sowohl die genaue Speicheradresse als auch deren Inhalt im Format `0x........` an. Was bedeutet das Präfix `0x`?

Aufgabe 2 (Laden von Konstanten) (4 Punkte)

Mit dem MIPS-Pseudobefehl "li r,k" kann eine 32-Bit Konstante k in ein Register r geladen werden. Um möglichst wenig Instruktionen zu erzeugen, kann der Assembler den Befehl „li“ in Abhängigkeit von der Konstanten verschieden umsetzen. Bestimmen Sie mithilfe der Vorlesungsfolien die kleinste Instruktionsfolge, mit der "li r,k" jeweils für die folgenden Wertebereiche von k umgesetzt werden kann, und überprüfen Sie diese mit einem Beispielwert in der MARS-Umgebung. Achten Sie hierbei darauf, dass Sie die Beispiele so wählen müssen, dass Sie nicht im Schnitt aus mehreren Bereichen liegen. Geben Sie außerdem an, ob in der MARS-Umgebung die kleinste Instruktionsfolge für die jeweilige Menge genutzt wird. Benutzen sie $3 für r und $1, um Zwischenwerte zu speichern.

a) $-2^{15} \leq k < 2^{15}$: 16 Bit Zweierkomplement
b) $0 \leq k < 2^{16}$: 16 Bit Betragszahl
c) $0 \leq k < 2^{32}$: 32 Bit Betragszahl (Tipp: $k = k1*2^{16} + k2$, $0 \leq k1 < 2^{16}$ und $0 \leq k2 < 2^{16}$)
d) $k = k1*2^{16}$, $0 < k1 < 2^{16}$: 32 Bit Betragszahl

Aufgabe 3 (Multiplikation) (4 Punkte)

Schreiben Sie ein Assemblerprogramm zum Testen der Multiplikationsvarianten mit den Befehlen mul mult multu und mulo nach dem folgenden Schema:
Legen Sie in zwei Speicherzellen (wert1 und wert2 im Bereich .data) die Konstanten +1 und –1 ab und multiplizieren Sie die Werte zunächst mit mult und dann mit multu. Anschließend berechnen Sie das Produkt der Zahlen 1035 und 4721467 mit den Befehlen mul und mulo.

a) Geben Sie die Ergebnisse der Multiplikationen tabellarisch an (Register Hi und Lo und ggf. Reg[4]). Das Reg[4] soll dabei, wenn benötigt, als Zielregister benutzt werden. Schreiben Sie nicht einfach die internen (möglicherweise falschen) Werte der Registerinhalte des Simulators ab, sondern geben Sie das an, was ein fertiges kompiliertes Programm sinnvollerweise anzeigen würde.

Befehl	**Hi**	**Lo**	**$4**
`mult`			
`multu`			
`mul`			
`mulo`			

b) Welches Ergebnis liefern die Befehle mult und multu dezimal? Warum liefert der Befehl mulo kein Ergebnis?

Aufgabe 4 (Befehlssequenz) (4 Punkte)

Gegeben sei folgende Sequenz von MIPS-Befehlen:

(Geben Sie ab der 2. Zeile nur Veränderungen an)

	Registerinhalte nach Ausführung des Befehls			
	$2	$3	$4	$lo
bei Programmstart	?	?	?	?
`li $2,0x04`				
`ori $3,$0,48`				
`mul $4,$3,$2`				
`mfhi $4`				
`add $2,$2,$3`				
`addi $3,$2,0xAB63`				
`and $3,$3,0xFFB6`				
`ori $4,$3,0x419C`				

Hinweise:
Es wird empfohlen, im Hexsystem zu rechnen. Logische *Immediate*-Befehle nutzen die *zero-extend*-, nicht die *sign-extend*-Funktion. Sie sollten diese Aufgabe ohne Benutzung des Simulators lösen können.

Die Abgaben sollen bis Mittwoch den 11. Dezember 2013 um 18.00 Uhr in die Briefkästen in der Otto-Hahn-Strasse 20 eingeworfen werden. Bitte Name (bei einem 3er-Team alle), Matrikel- und Gruppennummer oben auf der ersten Seite der Lösungen angeben.