26. November 2014

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

Um die Aufgaben zu lösen, sollten 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
Installieren Sie den Simulator auf Ihrem Rechner.

Aufgabe 1 (Simulatoroberfläche) (4 Punkte)

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 0x0a93     # 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“).

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 für die Berechnung als auch das berechnete Ergebnis an.
(in der Register-Transfer-Notation soll x,y,z und das Ergebnisregister erscheinen)

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

d) Wo (Segment auf dem Bildschirm) 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. Je nach Wertebereich von k erzeugt der Assembler verschiedene Instruktionen für den Befehl "li r,k". Diese verschiedenen Instruktionen (in Abhängigkeit von k) sollen hier untersucht werden.
Geben Sie mit je einem Beispiel (wählen Sie ein geeignetes k) an, welche Anweisungen der Assembler (in der MARS-Umgebung unter Execute -> Basic) erzeugt. $3 für r und $1, um Zwischenwerte zu speichern.

a) $-2^{15} \le k < 2^{15}$: 16 Bit Zweierkomplement, 0x?000????
b) $0 \le k < 2^{16}$: 16 Bit Betragszahl, 0x0000????
c) $k = k1*2^{16}$, $2^{16} < k1 < 2^{32}$: 32 Bit Betragszahl , 0x????0000
d) $0 \le k < 2^{32}$: 32 Bit Betragszahl (Tipp: $k = k1*2^{16} + k2$, $0 \le k1 < 2^{16}$ und $0 \le k2 < 2^{16}$) , 0x????????

? bedeutet jeweils 0 oder 1.

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 $2,0x05`				
`ori $4,$0,48`				
`mul $3,$4,$2`				
`mfhi $3`				
`add $2,$2,$4`				
`addi $4,$2,0xAB39`				
`and $3,$4,0x7F56`				
`ori $3,$3,0x87BA`				

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 03. Dezember 2014 um 18.00 Uhr in die Briefkästen in der Otto-Hahn-Strasse 12 eingeworfen werden. Bitte Name (bei einem 3er-Team alle), Matrikel- und Gruppennummer oben auf der ersten Seite der Lösungen angeben.