georg.von-der-brueggen [☺] tu-dortmund.de
ulrich.gabor [☺] tu-dortmund.de
marco.duerr [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerstrukturen
Wintersemester 2018

Übungsblatt 9 (Block C – 1)

(16 Punkte)

Abgabe bis spätestens Mittwoch, 19. Dezember 2018, 16:00 Uhr.
Besprechung ab Montag, 7. Januar 2019.

Um die Aufgaben zu lösen, sollten Sie den in der Vorlesung vorgestellten MARS Simulator installieren und mit ihm arbeiten. Sie finden die Software unter:
http://courses.missouristate.edu/KenVollmar/MARS/index.htm
Installieren Sie den Simulator auf Ihrem Rechner.

9.1 Simulatoroberfläche (2 Punkte)

Wählen Sie unter „File“ → „New“ und geben Sie im Editorfeld das nachfolgende Assemblerprogramm ein. Speichern Sie Ihre Eingabe als Datei (z. B. Blatt9A1.asm).

```
.data                # 01 Die Zeilenzahlen sind zum
x: .word 4711        # 02 Besprechen in der Übung
y: .word 11          # 03
z: .word 0x0A90      # 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 ordnungsgemäß 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 (wiederholt F7). Achten Sie bei jedem Schritt auf Veränderungen in den Registern 2 bis 4 (rechts unter „Registers“ in den Spalten „Number“ und „Value“).

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

b. Was berechnet das Programm in Register Reg[3]? Geben Sie die Berechnung für die gegebenenen Werte für x, y, z im .data-Bereich an.

   Geben Sie außerdem die allgemeine Berechnung in Register-Transfer-Notation (siehe Skript[1] S. 8) für beliebige x, y, z an. Fassen Sie dabei das gesamte Programm in eine Zuweisung zusammen.

[1] http://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/ws1314/rs/script/rs2.pdf

c. Die erste Programmzeile, die im Simulator angezeigt wird, lautet: **`lui $1,`** `0x1001` (unter dem Reiter „Basic“). Warum fügt der Simulator diese Zeile ein und woher kommt die 0x1001?

d. Wo (Segment in MARS) 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 als Hexadezimalzahl an. Was bedeutet das Präfix 0x?

9.2 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`**. Des Weiteren berechnen Sie das Produkt der Zahlen 1030 und 4721471 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. Welche Ergebnisse liefern die Befehle **`mult`** und **`multu`** dezimal? Warum liefert der Befehl **`mulo`** kein Ergebnis?

9.3 Befehlssequenz (4 Punkte)

Gegeben sei folgende Sequenz von MIPS-Befehlen. Ergänzen Sie die folgende Tabelle, geben Sie dabei nur Veränderungen an:

	Registerinhalte nach Ausführung des Befehls			
	$2	$3	$4	$lo
bei Programmstart	?	?	?	?
`li $2, 0x04`				
`ori $4, $0, 49`				
`mul $3, $4, $2`				
`mfhi $3`				
`add $2, $2, $4`				
`addi $4, $2, 0xAB39`				
`and $3, $4, 0x6F57`				
`ori $3, $3, 0x84BE`				

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.

9.4 Assemblerprogrammierung (2 Punkte)

In einer Assemblerprozedur soll die folgende Register-Transfer-Anweisung durchgeführt werden:

```
Speicher[0x10010000] := (Speicher[0x10010004] - Speicher[0x10010008] + 8)
* Speicher[0x1001000C] + Speicher[0x10010010]
```

Vervollständigen Sie die nachfolgende Sequenz bis einschließlich zum Programmende. Die Verwendung von lw- und sw-Befehlen mit Offset-Werten, die nicht mit 16 Bit dargestellt werden können, ist nicht zulässig. Verwenden Sie möglichst wenige Register und möglichst wenige Befehle!

```
li   $2,0x10010000 # vorgegeben: Reg[2] := 0x10010000
lw   $3,0x4($2)    # vorgegeben: Reg[3] := Speicher[0x10010004]
lw   $4,0x8($2)    # vorgegeben: Reg[4] := Speicher[0x10010008]

_______________

_______________

_______________

_______________

_______________

_______________

_______________

_______________

_______________

_______________
```

9.5 Einfache Fallunterscheidung (4 Punkte)

Ein MIPS-Programm soll bei der Auswertung der Klausurkorrekturen helfen. Es soll feststellen, ob jemand bestanden hat (Punkte ≥ 40) oder nicht (Punkte < 40).

Zunächst ist die Variable „Bestanden“ mit 99 belegt, was soviel bedeutet wie „die Note steht noch nicht fest“. Ergänzen Sie das Programmfragment so, dass in „Bestanden“ entweder eine 1 für „bestanden“ oder eine 0 für „nicht bestanden“ steht.

```
.data
  Punkte: .word 42       # Klausurpunkte
  Bestanden: .word 99    # 0 Nein, 1 Ja, 99 weiß nicht
  Grenze: .word 40       # Bestehensgrenze 40 Punkte

.text
.globl main

.
.
.
```

Hinweise:

Die Abgaben sollen bis Mittwoch, 19. Dezember 2018, 16:00 Uhr in die Briefkästen in der Otto-Hahn-Straße 12 eingeworfen werden.

Die Briefkästen finden Sie in der ersten Etage der Otto-Hahn-Straße 12 am Übergang zum Erdgeschoss der Otto-Hahn-Straße 14. Die Briefkästen sind mit dem Namen der Veranstaltung, der Gruppennummer sowie der Zeit der Übung gekennzeichnet. Für Rechnerstrukturen sind dies die Briefkästen mit den Nummern 20 bis 32.

Schreiben Sie unbedingt Ihren **Namen**, Ihre **Matrikelnummer** und Ihre **Gruppennummer** rechts oben auf Ihre Abgabe. Sie dürfen als Team mit bis zu zwei weiteren Personen abgeben. Geben Sie dann nur eine einzige Lösung ab und schreiben Sie alle Namen und Matrikelnummern des Teams auf die gemeinsame Abgabe.

Heften Sie die Abgabe bitte zusammen (Tacker oder notfalls Büroklammer). Bitte die Abgabe **nicht falten** und **keine Schnellhefter oder Umschläge** abgeben.

Es gibt insgesamt 12 Übungsblätter, die in 3 Blöcke (A, B, C) aufgeteilt sind. In jedem Block müssen Sie 30 Punkte von 64 möglichen Punkten erreichen, um zur Prüfung zugelassen zu werden.

HelpDesk Rechnerstrukturen:

Neben den Übungen bieten wir dieses Jahr auch einen speziellen RS Help Desk an. Der Help Desk kann euch bei der Bearbeitung der Übungsaufgaben, der Klausurvorbereitung oder sonstigen vorlesungrelevanten Problemen helfen. Weitere Information finden Sie auf der Webseite zur Vorlesung.