

Übungsblatt 7

Rechnerarchitektur im SS 2022

Zu Modul K

Abgabetermin: 19.06.2022, 23:59 Uhr

Besprechung: T-Aufgaben: 13.06.22 - 17.06.22, H-Aufgaben: 20.06.22 - 24.06.22

Aufgabe 45: (T) Test des MIPS Simulators

(- Pkt.)

Für diese Aufgabe sollten Sie sich mit dem MIPS-Simulator SPIM vertraut machen. Sie können einen MIPS-Simulator von der Vorlesungshomepage herunterladen.

- Laden Sie sich das Assemblerprogramm `simple.s` von der Rechnerarchitektur-Homepage herunter und speichern Sie es in Ihrem Home-Verzeichnis ab.
- Starten Sie Ihren Simulator.
- Laden Sie das Programm `simple.s` in den Simulator und führen Sie es aus. Dabei sollte eine Konsole erscheinen, über die die Ein- und Ausgabe erfolgt.

Beantworten Sie nun folgende Fragen.

- a. Welches Ergebnis liefert das Programm für die Eingabefolge "6, 7, 8, 9, 0"(d.h. nach Start des Programms erfolgt über die Konsole die Eingabe "6", gefolgt von *Enter*, dann die Eingabe "7", gefolgt von *Enter*, usw.)?
- b. Die folgenden Kommentare beschreiben Teile des Programms `simple.s`. Ordnen Sie den Kommentaren jeweils die minimale Anzahl an Codezeilen zu, die benötigt werden, um das beschriebene Verhalten im Code darzustellen, und geben Sie die Zeilennummer(n) dieser Zeile(n) an!
 - i) `str1` wird auf der Konsole ausgegeben.
 - ii) Es wird eine Zahl von der Konsole eingelesen.
 - iii) Das Programm wird beendet.
 - iv) Eine Zählvariable wird um den Wert 1 erhöht.
- c. In welchem Wertebereich müssen sich die eingegebenen Zahlen befinden, damit keine Fehlerbehandlung stattfindet (= damit das Label *error* nicht angesprungen wird).
- d. Welche mathematische Funktion berechnet das Programm?

Aufgabe 46: (T) Umsetzung Boolescher Ausdrücke

(- Pkt.)

Übersetzen Sie folgendes Pseudocodefragment in MIPS-Code. Gehen Sie davon aus, dass der Wert der Variablen *a* bereits in das Register *\$t0* geladen wurde.

```
1 IF (a < 0) OR (a > 99) THEN
2     a := a - 10;
3 ELSE
4     a := a - 1;
5 END;
```

Bedenken Sie dabei insbesondere: Der Ausdruck *a > 99* wird nur dann ausgewertet, wenn *a < 0* fehlgeschlagen ist.

Aufgabe 47: (T) SPIM Programmieraufgabe

(- Pkt.)

Erstellen Sie ein **vollständiges** SPIM-Programm, das folgendes durchführt:

- Es werden zwei positive Integer-Zahlen von der Konsole eingelesen.
- Es wird der Durchschnitt dieser beiden Zahlen auf eine Nachkommastelle genau berechnet.
- Das Ergebnis der Berechnung wird ausgegeben.

Tipp: Programmieren Sie diejenigen Schritte, die Sie auch beim handschriftlichen Dividieren durchführen!

Beachten Sie hierbei folgendes:

- Verwenden Sie nur die **unten aufgeführten Befehle**.
- Verwenden Sie für die Vorkommazahl das Register *\$s0* und für die Nachkommazahl das Register *\$s1*, ansonsten nur die temporären Register.
- **Kommentieren** Sie ihr Programm sinnvoll!
- Sowohl die Eingabe als auch die Ausgabe soll mit einem Anweisungstext versehen werden, wie z.B. *"Geben Sie die 1. Zahl ein: "*, etc.

Aufgabe 48: (H) Interaktion mit dem Stack

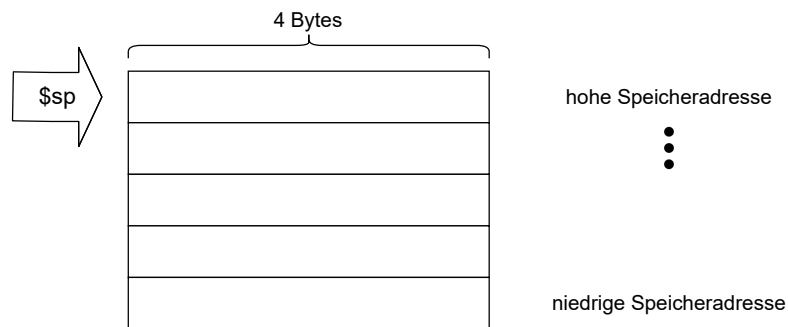
(5 Pkt.)

Sei folgendes MIPS-Codefragment gegeben, welches 4 Zahlen auf dem Stack speichern soll:

```
1     li $t0, 30
2     li $t1, 5
3     li $t2, 15
4     addi $sp, $sp, -16
5     sw $t2, 12($sp)
6     sw $t0, 8($sp)
7     sw $t1, 4($sp)
```

- Skizzieren Sie in folgenden Vorlage den Inhalt des Stacks nachdem alle Programmzeilen ausgeführt wurden. Kennzeichnen Sie auch die neue Position des Stackpointers!
- Welches Problem tritt hier auf?

- c. Wie könnte man dieses Problem verhindern. Geben Sie die Zeile und ihre Änderung an.



Aufgabe 49: (H) Assemblerprogrammierung unter SPIM

(15 Pkt.)

Laden Sie sich das Programm *simple-counter.s* von der Vorlesungs-Webseite herunter und beantworten Sie die folgenden Fragen zum Thema Assemblerprogrammierung des MIPS-Prozessors.

- a. Ordnen Sie dem Programm *simple-counter.s* jeder Zeile, die mit "# Kommentar <nr> Nr.:" versehen ist, den jeweils sinnvollsten der folgenden Kommentare zu. Ein Kommentar kann auch mehrfach benötigt werden. Schreiben Sie dazu eine Auflistung mit der Kommentarnummer und der Zuordnung zu jeweils einem der folgenden möglichen Kommentare (bspw.: Kommentar 13 Nr.:iii)
- (i) Schaffe Platz auf Stack
 - (ii) while i > 0
 - (iii) Übergebe Argument
 - (iv) Sichere i auf Stack
 - (v) Lade i vom Stack
 - (vi) i := 10
 - (vii) Sichere übergebenes Argument
 - (viii) i := i - 1
 - (ix) Setze Stackgröße zurück
- b. Geben Sie die letzten drei Zeilen an, die das Programm aus Teilaufgabe a) auf der Konsole ausgibt.

Aufgabe 50: (H) Assemblerprogrammierung unter SPIM

(6 Pkt.)

Im Folgenden soll ein MIPS-Assembler Programm vervollständigt werden, welches als Nutzereingabe eine 8-stellige Binärzahl von der Konsole entgegennimmt, sie in ihre dezimale Repräsentation umwandelt und diese auf der Konsole ausgibt. Die Eingabe wird vom Programm als String entgegengenommen. Danach soll in einer Schleife über die Zeichen diese Strings iteriert und geprüft werden, ob es sich beim aktuellen Zeichen um eine „0“ oder eine „1“ handelt und die entsprechende Wertigkeit aufsummiert werden. Zudem muss der Fall behandelt werden, dass das Ende des Strings, welches durch ein Byte mit dem Zahlenwert 0 markiert ist, erreicht wurde. Beachten Sie, dass der Sting nach 8 von der Konsole gelesenen Zeichen automatisch übernommen und Null-terminiert wird. Dementsprechend ist kein Zeilenumbruch enthalten.

Laden sie sich die `binarytodecimal.s` von der Homepage herunter und ergänzen Sie den dort angegebenen Coderahmen um insgesamt **6 Zeilen Code**, so dass das Programm wie beschrieben funktioniert. Tragen Sie Ihre Lösung unter den mit „# Ihre Loesung:“ markierten Stellen direkt in den Coderahmen der heruntergeladenen Datei ein.

Aufgabe 51: (H) Einfachauswahlaufgabe: MIPS/SPIM

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe explizit die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Der MIPS Prozessor besitzt die folgende Architektur:			
(i) MISC	(ii) RISC	(iii) CISC	(iv) Stack
b) Jedes MIPS-Register hat eine feste Breite. Sie beträgt:			
(i) 4 Bit	(ii) 8 Bit	(iii) 16 Bit	(iv) 32 Bit
c) In der MIPS Architektur steht ein Wort für..			
(i) ...die maximale Datengröße, die in einem Rechenschritt verarbeitet werden kann.	(ii) ...die Größe einer Speicherzelle.	(iii) ...die größte adressierbare Informationseinheit.	(iv) ...die kleinste adressierbare Informationseinheit.
d) Wie muss der Assembler-Befehl lauten, wenn der Inhalt von Register \$t1 durch den Inhalt von Register \$t2 dividiert und das Ergebnis im Zielregister \$t0 gespeichert werden soll?			
(i) <code>div \$t1,\$t0,\$t2</code>	(ii) <code>div \$t0,\$t1,\$t2</code>	(iii) <code>mul \$t2,\$t1,\$t0</code>	(iv) <code>div \$t2,\$t1,\$t0</code>
e) Gegeben sei folgende Zeile in SPIM Code: <code>var: .word 10, 11, 12, 13</code> Welcher Befehl lädt den Wert 12 in das Register \$t0?			
(i) <code>lw \$t0, var+8</code>	(ii) <code>la \$t0, var+4</code>	(iii) <code>lw \$t0, var</code>	(iv) <code>lw \$t0, var+4</code>

Überblick über die wichtigsten SPIM Assemblerbefehle		
Befehl	Argumente	Wirkung
add	Rd, Rs1, Rs2	$Rd := Rs1 + Rs2$ (mit Überlauf)
sub	Rd, Rs1, Rs2	$Rd := Rs1 - Rs2$ (mit Überlauf)
addu	Rd, Rs1, Rs2	$Rd := Rs1 + Rs2$ (ohne Überlauf)
subu	Rd, Rs1, Rs2	$Rd := Rs1 - Rs2$ (ohne Überlauf)
addi	Rd, Rs1, Imm	$Rd := Rs1 + Imm$
addiu	Rd, Rs1, Imm	$Rd := Rs1 + Imm$ (ohne Überlauf)
div	Rd, Rs1, Rs2	$Rd := Rs1 \text{ DIV } Rs2$
rem	Rd, Rs1, Rs2	$Rd := Rs1 \text{ MOD } Rs2$
mul	Rd, Rs1, Rs2	$Rd := Rs1 \times Rs2$
b	label	unbedingter Sprung nach label
j	label	unbedingter Sprung nach label
jal	label	unbed. Sprung nach label, Adresse des nächsten Befehls in \$ra
jr	Rs	unbedingter Sprung an die Adresse in Rs
beq	Rs1, Rs2, label	Sprung, falls $Rs1 = Rs2$
beqz	Rs, label	Sprung, falls $Rs = 0$
bne	Rs1, Rs2, label	Sprung, falls $Rs1 \neq Rs2$
bnez	Rs1, label	Sprung, falls $Rs1 \neq 0$
bge	Rs1, Rs2, label	Sprung, falls $Rs1 \geq Rs2$
bgeu	Rs1, Rs2, label	Sprung, falls $Rs1 \geq Rs2$
bgez	Rs, label	Sprung, falls $Rs \geq 0$
bgt	Rs1, Rs2, label	Sprung, falls $Rs1 > Rs2$
bgtu	Rs1, Rs2, label	Sprung, falls $Rs1 > Rs2$
bgtz	Rs, label	Sprung, falls $Rs > 0$
ble	Rs1, Rs2, label	Sprung, falls $Rs1 \leq Rs2$
bleu	Rs1, Rs2, label	Sprung, falls $Rs1 \leq Rs2$
blez	Rs, label	Sprung, falls $Rs \leq 0$
blt	Rs1, Rs2, label	Sprung, falls $Rs1 < Rs2$
bltu	Rs1, Rs2, label	Sprung, falls $Rs1 < Rs2$
bltz	Rs, label	Sprung, falls $Rs < 0$
not	Rd, Rs1	$Rd := \neg Rs1$ (bitweise Negation)
and	Rd, Rs1, Rs2	$Rd := Rs1 \& Rs2$ (bitweises UND)
or	Rd, Rs1, Rs2	$Rd := Rs1 Rs2$ (bitweises ODER)
syscall		führt Systemfunktion aus
move	Rd, Rs	$Rd := Rs$
la	Rd, label	Adresse des Labels wird in Rd geladen
lb	Rd, Adr	$Rd := \text{MEM}[\text{Adr}]$
lw	Rd, Adr	$Rd := \text{MEM}[\text{Adr}]$
li	Rd, Imm	$Rd := Imm$
sw	Rs, Adr	$\text{MEM}[\text{Adr}] := Rs$ (Speichere ein Wort)
sh	Rs, Adr	$\text{MEM}[\text{Adr}] \text{ MOD } 2^{16} := Rs$ (Speichere ein Halbwort)
sb	Rs, Adr	$\text{MEM}[\text{Adr}] \text{ MOD } 256 := Rs$ (Speichere ein Byte)

Funktion	Code in \$v0	Funktion	Code in \$v0
print_int	1	read_float	6
print_float	2	read_double	7
print_double	3	read_string	8
print_string	4	sbrk	9
read_int	5	exit	10