Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

Übungsblatt 8
Rechnerarchitektur im SS 22

Zu den Modulen I, J und K

Abgabetermin: 26.06.2022, 18:00 Uhr
Besprechung: T-Aufgaben: 20.06.22 - 24.06.22, H-Aufgaben: 27.06.22 - 01.07.22

Aufgabe 52: (T) Parameterübergabe bei Unterprogrammaufrufen

(- Pkt.)

Für die Parameterübergabe bei Prozeduraufrufen existieren verschiedene Möglichkeiten.

a. Erläutern Sie zunächst die Begriffe *call by value* und *call by reference*. Geben Sie zu beiden Konzepten je ein Beispiel in einer Hochsprache an.

b. Schreiben Sie nun ein SPIM-Programm, das den Durchschnitt der Werte eines Feldes berechnet. Die Berechnung selbst soll dabei ein Unterprogramm erledigen. Die Übergabe des Feldes soll nach dem Konzept *call by value* erfolgen.

   Achtung: Das Hauptprogramm soll dem Unterprogramm **alle** zur Berechnung notwendigen Werte über den Stack zur Verfügung stellen! Sie dürfen bei Ihrer Implementierung davon ausgehen, dass sich das Feld bereits im Speicher befindet.

c. Schreiben Sie Ihr Programm aus Aufgabe b) so um, dass die Übergabe des Feldes nach dem Konzept *call by reference* funktioniert.

   Achtung: Das Hauptprogramm soll dem Unterprogramm **ausschließlich** Speicheradressen zur Berechnung zur Verfügung stellen! Sie dürfen wieder davon ausgehen, dass sich das Feld bereits im Speicher befindet. Sie dürfen zur Übergabe der Adressen an das Unterprogramm die laut Konvention dafür vorgesehenen Register $a0 - $a3 verwenden. Das Ergebnis des Unterprogrammaufrufes dürfen Sie dem Hauptprogramm über das Register $v0 zur Verfügung stellen.

Aufgabe 53: (T) SPIM: 2er-Komplement-Darstellung

(- Pkt.)

a. Schreiben Sie ein MIPS-Assembler-Programm, das eine positive bzw. eine negative Dezimalzahl einliest und deren Binärdarstellung unter Verwendung der 2er-Komplement-Darstellung ausgibt. Verwenden sie den Systemaufruf `read_int ($v0 := 5)`, um die Dezimalzahl von der Konsole einzulesen. Testen Sie Ihr Programm mit verschiedenen positiven und negativen Eingaben.

b. Was ist die größte und die kleinste Dezimalzahl für die Ihr Programm korrekt funktioniert? Begründen sie Ihre Antwort

Aufgabe 54: (H) Zahlendarstellung im Rechner

(12 Pkt.)

a. Geben Sie die 1er- und 2er-Komplementdarstellung der folgenden Zahlen unter der Annahme an, dass 8 Bit **inklusive** des Vorzeichenbits zur Verfügung stehen.

(i) 0

(ii) -57

(iii) 127

b. Berechnen Sie in der 1er- und 2er-Komplementdarstellung folgende Differenzen unter der Annahme, dass 8 Bit **inklusive** des Vorzeichenbits zur Verfügung stehen. Achten Sie darauf, dass der Rechenweg ersichtlich ist.

(i) $44 - 37$

(ii) $64 - 32$

(iii) $-45 - 83$

c. Nennen Sie zwei Vorteile, die sich bei der Zweierkomplementdarstellung von Binärzahlen gegenüber der Vorzeichen/Betrag-Darstellung (sign/magnitude) in Rechnern ergeben.

d. Gegeben seien die Zahlen $u = 100110$ und $v = 101111$ in Zweierkomplementdarstellung auf Basis von 6 Bit. Addieren Sie diese beiden Zahlen und achten Sie auf einen nachvollziehbaren Rechenweg. Hat bei der Addition ein Überlauf stattgefunden?

Aufgabe 55: (H) Gleitkommazahlen nach IEEE 754

(9 Pkt.)

Geben Sie die Darstellung folgender Zahlen als Gleitkommazahl nach IEEE 754 in einfacher (32-Bit) und doppelter (64-Bit) Genauigkeit an. **Achtung:** Der Rechenweg muss ersichtlich sein!

a. $(10,5)_{10}$

b. $(0,1)_{10}$

c. $(\frac{-2}{3})_{10}$

Aufgabe 56: (H) Cäsar-Verschlüsselung unter SPIM

(6 Pkt.)

Bearbeiten Sie die folgende Aufgabe zum Thema Assemblerprogrammierung unter SPIM.
Hinweis: Eine Übersicht der SPIM-Befehle finden Sie am Ende des Übungsblatts.
Im Folgenden soll ein MIPS-Assembler Programm vervollständigt werden, welches einen gegebenen Text mittels der **Caesar-Verschlüsselung** in einen Geheimtext umwandelt. Bei der Caesar-Verschlüsselung wird jeder Buchstabe im zu verschlüsselnden Text um eine vorher festgelegte Distanz im Alphabet verschoben. Ist z.B. die Distanz 3, so wird der Buchstabe A zum Buchstaben D, der Buchstabe B zum Buchstaben E, ..., der Buchstabe Z zum Buchstaben C.
Das folgende MIPS-Assembler Programm erwartet als Nutzereingabe die Distanz, um die die Buchstaben verschoben werden sollen und verschlüsselt dann einen gegebenen Text.
Ergänzen Sie den unten 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 folgenden Coderahmen ein:

```
1  .data
2
3  shift_text: .asciiz "Um wieviele Stellen soll der Text verschoben werden: "
4  string1: .asciiz "Der verschluesselte Text lautet:  "
5  secret: .asciiz "GEHEIMNIS"
6  string_a: .asciiz "A"
7  string_z: .asciiz "Z"
8
9  result: .space 9
10
11 .text
12 main:
13         # t0  - Zum Zwischenspeichern der Position des aktuell betrachteten
               Buchstabens
14         # t1 - Gibt die Laenge des Geheimworts an
15         # t2 - ASCII Wert des Buchstaben A (65)
16         # t3 - ASCII - WERT des Buchstaben Z (90)
17         li  $t0, 0
18         li  $t1, 9
19         lb  $t2, string_a
20         lb  $t3, string_z
21
22         la $a0, shift_text      # String mit Anfangsadresse shift_text in $a0
               laden
23         li $v0, 4               # 4 in $v0 laden
24         syscall                 # Text mit Anfangsadreesse in $a0 auf der
               Konsole ausgeben
25
26         li $v0, 5               # 5 in $v0 laden
27         syscall                 # Zahl eingeben
28
29         move $s1, $v0           # Eingegebene Zahl in $s1 speichern
30
31 loop:   bge $t0, $t1, end       # Falls alle Buchstaben betrachtet wurden ->
       Springe end
32         lb  $t4, secret($t0)    # Lade den aktuellen Buchstaben in $t4
33
34 caesar: ###################################
35         # Fuegen Sie hier Ihre Loesung ein #
36         ###################################
37
38
39
40         ######################
41         # Ende Ihrer Loesung #
42         ######################
43         bgt $t4, $t3, cadd      # Falls das Ergebnis > Z --> springe zu cadd
44
45 save:   ###################################
46         # Fuegen Sie hier Ihre Loesung ein #
47         ###################################
48
49
50
51
52
53
54         ######################
```

```
55          # Ende Ihrer Loesung #
56          ######################
57          j loop                  # Springe zum Label loop
58
59  cadd:   ###################################
60          # Fuegen Sie hier Ihre Loesung ein #
61          ###################################
62
63
64
65
66
67
68
69          ######################
70          # Ende Ihrer Loesung #
71          ######################
72          j save                  # Springe zum Label save
73
74
75  end:    la $a0, string1         # Anfangsadresse des Strings string1 wird in
        $a0 geladen
76          li $v0, 4               # 4 wird in $v0 geladen
77          syscall                 # String string1 wird auf der Konsole ausgeben
78
79          la $a0, result          # Anfangsadresse des Strings result wird in
                $a0 geladen
80          li $v0, 4               # 4 wird in $v0 geladen
81          syscall                 # String result  wird auf der Konsole ausgeben
82
83          li $v0, 10              # 10 wird in $v0 geladen
84          syscall                 # Programm wird beendet
```

Aufgabe 57: (H) Einfachauswahlaufgabe: Wiederholung

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Was bewirkt der Spim-Befehl `li $v0 5`:			
(i) Der Wert 5 wird in das Register `$v0` geladen	(ii) Es wird ein Integer von der Konsole eingelesen	(iii) Es wird eine Zahl vom Typ double von der Konsole eingelesen	(iv) Es wird ein Integer auf der Konsole ausgegeben
b) Welche Aussage ist korrekt? MIPS ist eine...			
(i) Stack-Architektur	(ii) Load-Store-Architektur	(iii) Heap-Architektur	(iv) Last-in-First-Out-Architektur
c) Welche Aussage ist falsch? Die Funktion `syscall`...			
(i) führt eine Funktion des Betriebssystems aus	(ii) besitzt selbst keine Parameter	(iii) erwartet die Nummer der auszuführenden Funktion in `$v0`	(iv) beendet das Programm sofort

d) Bei welcher Belegung (x_1, x_2, x_3) ergibt die Boolesche Funktion $f(x_1, x_2, x_3) = (x_1 \cdot \overline{x}_2) + (x_2 \cdot \overline{x_3})$ den Wert 1?			
(i) $(0, 1, 0)$	(ii) $(0, 1, 1)$	(iii) $(1, 1, 1)$	(iv) $(0, 0, 1)$
e) Wofür steht CISC im Zusammenhang mit Mikroprozessoren?			
(i) Controversy Instruction Set Computer	(ii) Complex Instruction Set Calculator	(iii) Constructive Instruction Set Computer	(iv) Complex Instruction Set Computer

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 DIV Rs2
rem	Rd, Rs1, Rs2	Rd := Rs1 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 := MEM[Adr]
lw	Rd, Adr	Rd := MEM[Adr]
li	Rd, Imm	Rd := Imm
sw	Rs, Adr	MEM[Adr] := Rs (Speichere ein Wort)
sh	Rs, Adr	MEM[Adr] MOD 2^{16} := Rs (Speichere ein Halbwort)
sb	Rs, Adr	MEM[Adr] 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