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

mobile and distributed systems group

Tutoriumsblatt 8
Rechnerarchitektur im SoSe 2020

Zu den Modulen K

Tutorium: Die Aufgaben werden in Tutorien-Videos vorgestellt, die am 11. Juni 2020 (17 Uhr) veröffentlicht werden.

Aufgabe T24: 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 T25: 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

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