Hochschule RheinMain
Fachbereich DCSM - Informatik
Prof. Dr. Reinhold Kröger
Prof. Dr. Martin Gergeleit

Betriebssysteme und Rechnerarchitektur
WS 2015/16
LV 3142

Übungsblatt 3
Bearbeitungszeit 2 Wochen
Abgabetermin: 30.11.2015, 4:00 Uhr

In dieser Übung entwickeln Sie selbst mit SPIM ein MIPS-Assembler-Programm. Sie sollen zeigen, dass Sie die grundsätzliche Funktionsweise der ISA-Schnittstelle verstanden haben, einen einfachen Algorithmus auf dieser Ebene umsetzen und Systemaufrufe verwenden können und in der Lage sind, dazu den Konventionen entsprechenden Assembler-Code zu schreiben.

Aufgabe 3.1 (Zählen):

Entwickeln Sie ein MIPS-Assembler-Programm `buchstaben.asm`, das einen beliebigen String von der Konsole einliest und die enthaltenen Groß- und Kleinbuchstaben zählt.

Das Programm soll als Eingabe den String (nur Klein- und Großbuchstaben a-z, A-Z und Leerzeichen, keine Umlaute, Maximallänge 100 Zeichen) erwarten und als Ausgabe die Anzahl der Groß- und Kleinbuchstaben ausgeben:

```
String:
Hochschule RheinMain
Anzahl Grossbuchstaben:
3
Anzahl Kleinbuchstaben:
16
```

Strukturieren Sie Ihr Programm so, dass es Unterprozeduren mit folgenden Parametern gibt, und rufen Sie diese Funktionen von `main()` passend auf:

```
int gross(char *text);
int klein(char *text);
```

Machen Sie sich auch Gedanken über die Speicherbereiche für die Strings, ggf. können Sie sie auch „in place“ kodieren.

Entwickeln Sie den Code so, dass er den Aufruf-, Stack- und Registerbelegungskonventionen des MIPS entspricht (s. Vorlesung).

Aufgabe 3.2 (Ändern):

Erweitern Sie Ihr Assembler-Programm aus Aufgabe 3.1 um eine weitere Funktion, so dass im eingegebenen String Groß- in Kleinbuchstaben und umgekehrt umgewandelt werden:

```
char *tausche(char *text);
```

Geben Sie den geänderten String sowie die Anzahl der Groß- und Kleinbuchstaben aus. Rufen Sie danach die Funktion erneut auf, um den String wieder in den Ursprungszustand zurückzubringen. Geben Sie wiederholt die Anzahl der Buchstaben aus. Beispiel (Eingaben in Fettschrift):

```
String:
Hochschule RheinMain
Veraendert:
hOCHSCHULE rHEINmAIN
Anzahl Grossbuchstaben:
16
Anzahl Kleinbuchstaben:
3
Zweimal veraendert:
Hochschule RheinMain
Anzahl Grossbuchstaben:
3
Anzahl Kleinbuchstaben:
16
```

Aufgabe 3.3 (Analyse):

Testen Sie Ihre Funktion mit verschiedenen Beispielen. Analysieren Sie, wie viele Maschinenbefehle die Unterroutinen zum Zählen der Buchstaben und für die Umwandlung benötigen (in Abhängigkeit von der Länge der Strings).

Aufgabe 3.4 (Rekursion):

Schreiben Sie ein MIPS-Assembler-Programm `pascal.asm`, das in `main()` zwei beliebige gültige Integer-Zahlen n und k größer oder gleich 0 sowie $k \leq n$ von der Konsole einliest und danach den Binomialkoeffizienten $\binom{n}{k}$ mit Hilfe der Funktion `pascal()` **rekursiv** berechnet und das Ergebnis auf der Konsole ausgibt:

```
int pascal(int n, int k);
```

Die Berechnungsvorschrift entspricht dem *Pascalschen Dreieck* und wird durch folgende Gleichung ausgedrückt: $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Die Funktion soll bei $k=0$ bzw. $k=n$ den Wert 1 zurückgeben (Abbruchbedingung, Ränder des Pascalschen Dreiecks). Sie brauchen die Wertebereiche von n und k vorher nicht auf Gültigkeit prüfen.

Siehe auch: https://de.wikipedia.org/wiki/Pascalsches_Dreieck

Beispiel (Eingaben in Fettschrift):

```
Zahl n groesser oder gleich 0:
7
Zahl k groesser oder gleich 0 und kleiner oder gleich n:
3
Binominalkoeffizient:
35
```

Bewertung:

Aufgabe	Kriterien	Punkte
3.1	Korrekte Funktionalität / geforderte Ein- und Ausgaben	2
	Korrekte Verwendung der Register bei den Systemaufrufen	1
	Prolog / Epilog main(), gross() und klein()	1
	Parameterübergabe in $a0 - $a3	1
	Return in $v0	1
	Register $s0 - $s7 vor Verwendung gesichert	1
3.2	Korrekte Funktionalität / geforderte Ausgaben	3
	Prolog / Epilog main() und tausche()	1
3.3	Analyse	2
3.4	Rekursion / Korrekte Funktionalität / geforderte Ein- und Ausgaben	2
	Prolog / Epilog main() und pascal()	1
	Parameterübergabe in $a0 - $a3	1
	Return in $v0	1
	Register $s0 - $s7 vor Verwendung gesichert	1
	Stack-Pointer $sp korrekt	1
	Extrapunkt: kein Crashes	(+1)
	Abzüge Lesbarkeit / Kommentare	(-3)
	Gesamt	**20 / 21**