universität innsbruck

Proseminar Rechnerarchitektur

Aufgabenzettel 9

Wintersemester 2021/22

2. Dezember 2021

Zu bearbeiten bis Donnerstag, den **9. Dezember**.

1 Bedingungen

Geben Sie die ARM-Befehle für folgende bedingten Operationen an. Versuchen Sie wieder, die Operationen in so wenigen Befehlen wie möglich durchzuführen.

a) Subtrahiere `r1` von `r0`. Falls das Ergebnis negativ ist, addiere `r1` wieder zum Ergebnis.

b) Überprüfe, ob das niederwertigste Bit in `r3` gesetzt ist. Wenn es gesetzt ist, addiere `r2` zu `r1`.

c) Überprüfe, ob das i-te Bit in `r1` gesetzt ist. Wenn es gesetzt ist, addiere `r2`, um i Bit nach links verschoben, zu `r1`.

d) Gegeben sei eine 64-Bit Zahl, deren niederwertige Hälfte in `r0` liegt und ihre höherwertige Hälfte in `r1`. Addieren Sie nun eine 32-Bit Zahl, die in `r3` liegt, dazu.
Hinweis: Berücksichtigen Sie eventuelle Überträge zwischen den Hälften.

2 Schleifen

Gegeben sei folgender Assembler-Kode, der den Programmstart und einen Textpuffer definiert:

```
1 .arm
2 .global _start
3 .data
4 msg:
5 .ascii "Revert me!"
6 len = . -msg
7 .align
8 .text
```

Schreiben Sie ein Assemblerprogramm, das den Text in `msg` umdreht und von hinten nach vorne ausgibt. Ihre Lösung sollte für Texte beliebiger Länge funktionieren.

3 Funktionen

In Aufgabe 2 haben Sie ein Programm geschrieben, das einen Text umdreht und dann ausgibt. In dieser Aufgabe wollen wir die beiden Funktionen „Umdrehen“ und „Ausgeben“ öfters und voneinander unabhängig verwenden. Ergänzen Sie den Assembler-Kode an den gekennzeichneten Stellen, sodass beim Ausführen die gewünschte Ausgabe erscheint.

```
.global _start
.data
txt1:
  .ascii "Hallo Innsbruck!"
len1 = . - txt1
txt2:
  .ascii "Griass enk!"
len2 = . - txt2
newline:
  .ascii "\n"
.align
.text
_start:
  // txt1: vorwaerts, rueckwaerts, vorwaerts
  LDR r0, =txt1
  LDR r1, =len1
  BL print_revert_print
  // txt2: vorwaerts, rueckwaerts, vorwaerts
  LDR r0, =txt2
  LDR r1, =len2
  BL print_revert_print
exit:
  MOV r0, #0
  MOV r7, #1
  SWI #0
print_revert_print:
  STMFD sp!, {r4-r5, lr}
  MOV r4, r0
  MOV r5, r1
  BL print
  BL print_newline
  MOV r0, r4
  MOV r1, r5
  BL revert
  MOV r0, r4
  MOV r1, r5
  BL print
  BL print_newline
  MOV r0, r4
  MOV r1, r5
  BL revert
  MOV r0, r4
  MOV r1, r5
  BL print
  BL print_newline
  LDMFD sp!, {r4-r5, pc}
print_newline:
  STMFD sp!, {r7, lr}
  MOV r0, #1
  LDR r1, =newline
  MOV r2, #1
  MOV r7, #4
  SWI #0
  LDMFD sp!, {r7, pc}
revert:
  // Umdrehen-Funktion hier einbauen!
print:
  // Ausgeben-Funktion hier einbauen!
```

Gewünschte Ausgabe:

```
Hallo Innsbruck!
!kcurbsnnI ollaH
Hallo Innsbruck!
Griass enk!
!kne ssairG
Griass enk!
```