universität innsbruck

Proseminar Rechnerarchitektur

Aufgabenzettel 10

Wintersemester 2021/22

9. Dezember 2021

Zu bearbeiten bis Donnerstag, den **16. Dezember**.

1 Lieber arm dran als Arm ab

a) Das folgende Programm soll das Maximum der Werte in den Registern `r0` und `r1` zurückgeben, bleibt aber beim Ausführen in einer Endlosschleife hängen. Wo liegt der Fehler?

```
.global _start
.text
_start:
    MOV r0, #7
    MOV r1, #42
    BL return_max
max:
    // r0 := max(r0, r1)
    CMP r0, r1
    MOVLT r0, r1
    MOV pc, lr
return_max:
    B max
    MOV r7, #1
    SWI #0
```

arm-common-mistakes-a.S

b) Das folgende Programm soll das Ergebnis von $(x + y)^2$ berechnen, stürzt aber beim Ausführen ab. Wo liegt der Fehler?

```
.global _start
.text
_start:
    MOV r0, #4
    MOV r1, #5
    MOV r7, #1
    BL calc
    SWI #0
calc:
    // r0 := (r0 + r1)^2
    ADD r7, r0, r1
    MUL r0, r7, r7
    MOV pc, lr
```

arm-common-mistakes-b.S

c) Das folgende Programm soll das Ergebnis von $2 \cdot ((x - 1)^2 + 1)$ berechnen, bleibt aber beim Ausführen in einer Endlosschleife hängen. Wo liegt der Fehler?

```
.global _start
.text
_start:
    MOV r0, #4
    BL g
    MOV r7, #1
    SWI #0
f:
    // r0 := r0^2 + 1
    MUL r1, r0, r0
    ADD r0, r1, #1
    MOV pc, lr
g:
    // r0 := 2 * f(r0 - 1)
    SUB r0, r0, #1
    BL f
    ADD r0, r0, r0
    MOV pc, lr
```

arm-common-mistakes-c.S

2 Adressierung

ARM bietet verschiedene Adressierungsmodi für Speicherzugriffe. Die folgenden Kode-Schnipsel können jeweils durch eine einzige ARM-Instruktion ersetzt werden. Setzen Sie das um und beschreiben kurz mit eigenen Worten, was Ihre Instruktion tut.

a)
```
1 ADD r7, r7, #4
2 LDR r0, [r7]
3 SUB r7, r7, #4
```

b)
```
4 ADD r3, r3, #4
5 STR r2, [r3]
```

c)
```
6 LDRB r11, [r1]
7 SUB r1, r1, #1
```

d)
```
8  ADD r6, r6, r5
9  STRB r0, [r6]
10 SUB r6, r6, r5
```

e)
```
11 MOV r5, r1, LSL #1
12 LDRH r4, [r0, r5]
```

f)
```
13 MOV r5, r1, LSL #2
14 LDR r4, [r0, r5]
```

g)
```
15 ADD r3, r3, r2
16 LDRSB r1, [r3]
```

h)
```
17 LDRSH r1, [r10]
18 ADD r10, r10, r3
```

3 Mit maximaler Geschwindigkeit voraus

Gegeben sei ein C-Programm, das den maximalen Wert einer zufällig initialisierten Liste ermittelt.

```c
#include "./libRA.c"
extern int max(int* arr, int len);

/*
Erklärung:
  Da wir ohne C-Standard-Bibliothek kompilieren verwenden wir _start statt main
  als Einsprungspunkt.
  Die Bibliotheksfunktionen exit, hex, rand, srand, println werden in libRA.c
  definiert und zum Teil in libRA.S implementiert. Die einzelnen Dateien werden
  beim Linken zusammengefügt (siehe Makefile).
*/
void _start() {
    int size = 20;
    int random_data[size];
    srand(12309183);

    println("Initialisiere Array mit zufälligen Daten.");
    for (int c = 0; c < size; c++){
      int num = (int) rand();
      random_data[c] = num;
      hex(num);
    }

    int max_element = max(random_data, size);

    println("Das maximale Element ist:");
    hex(max_element);

    exit(0);
}
```

main.c

Ihre Aufgabe ist es, die Funktion `extern int max(int* arr, int len)` in ARM-Assembler zu implementieren, sodass sie vom C-Programm aus aufgerufen werden kann. Gehen Sie dazu wie folgt vor.

a) Laden sie das Archiv `max.zip` aus dem OLAT auf den ZID-GPL und entpacken Sie es dort.

b) Wechseln Sie in entsprechenden Ordner und führen Sie `make run` aus, um das vorbereitete Projekt zu kompilieren, assemblieren, linken und ausführen.

c) Verschaffen Sie sich einen Überblick über das Projekt und die ausgelieferten Dateien. Am besten beginnen Sie mit dem `Makefile`.

d) Implementieren Sie Ihre `max` Funktion in der Datei `max.S` und testen Sie mit `make run`.

Hinweis: Erinnern Sie sich an die ARM-Aufrufkonventionen, die in der Vorlesung besprochen wurden.