EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Vorlesung Informatik der Systeme
SS 2015
Prof. Dr. M. Menth
Dipl.-Inform. W. Braun

Exercise 7

2. Juni 2015

Abgabe: 9. Juni 2015, 10.00 Uhr

Problem 7.1: *Leaf- and Non-Leaf-Procedures*

Die Fibonacci-Zahlenfolge ist eine unendliche Folge von Zahlen, bei der die ersten beiden Zahlen bekannt sind und die jeweils folgende Zahl sich durch Addition der beiden vorherigen Zahlen ergibt: $0, 1, 1, 2, 3, 5, 8, 13, \ldots$
Das n-te Element der Fibonacci-Zahlenfolge kann rekursiv oder iterativ berechnet werden.

C-Code für die rekursive Berechnung der Fibonacci-Zahlenfolge:

```
1  int fib_rek(int n){
2    if (n==0) return 0;
3    if (n==1) return 1;
4    return fib_rek(n-1)+fib_rek(n-2);
5  }
```

Assembler-Code für die rekursive Berechnung der Fibonacci-Zahlenfolge:

```
1  fib:
2      bgt $a0, 1, recurse
3      move $v0, $a0
4      jr $ra
5
6  recurse:
7      sub $sp, $sp, 12
8      sw $ra, 0($sp)
9      sw $a0, 4($sp)
10     sub $a0, $a0, 1
11     jal fib
12     sw $v0, 8($sp)
13     lw $a0, 4($sp)
14     sub $a0, $a0, 2
15     jal fib
16     lw $t0, 8($sp)
17     add $v0, $v0, $t0
18     lw $ra, 0($sp)
19     addi $sp, $sp, 12
20     jr $ra
```

C-Code für die iterative Berechnung der Fibonacci-Zahlenfolge:

```c
int fib_iter(int n) {
  int i, f0, f1, f;
  f0 = 0;
  f1 = 1;
  if (n == 0) return f0;
  if (n == 1) return f1;
  for (i = 2; i <= n; i = i + 1) {
    f = f0 + f1;
    f0 = f1;
    f1 = f;
  }
  return f;
}
```

1. Übersetzen Sie die iterative Prozedur in den MIPS-Assembler-Code, der in der Vorlesug vorgestellt wurde! 10 Points

   Beachten Sie: Die Lösung muss als eigenständige Datei mit dem Namen fib.asm abgeben werden. Verwenden Sie dazu das Formular im Moodle.

   Der Code in dieser Datei muss die folgenden Anforderungen erfüllen:

   - Die Datei fib.asm enthält nur den Code der Prozedur.
   - Der Code soll mit dem Label 'fib:' beginnen, wie es auch im Beispiel auf Seite 3 dargestellt ist.
   - Der Eingabeparameter wird in $a0 übergeben.
   - Das Ergebnis soll in $v0 gespeichert werden.

   Bei der Nichtbeachtung dieser Anforderungen gibt es Punktabzug!

2. Wieviele Register benötigen Sie für die beiden unterschiedlichen Umsetzungen? 1 Points

3. Wieviele Funktionsaufrufe werden jeweils benötigt um das 4. Fibonacci-Element zu berechnen? 2 Points

   Hint:
   Um den Überblick nicht zu verlieren, lohnt es sich die verschiedenen Funktionsaufrufe als Baumstruktur darzustellen. Die Anzahl der benötigten Unteraufrufe kann mit Hilfe der Baumstruktur abgezählt werden.

4. Illustrieren Sie für die rekursiv programmierte Funktion die Stack-Belegung in der tiefsten Rekursionsstufe, die benötigt wird, um das 4. Fibonacci-Element zu berechnen! 7 Points

Total: 20 Points

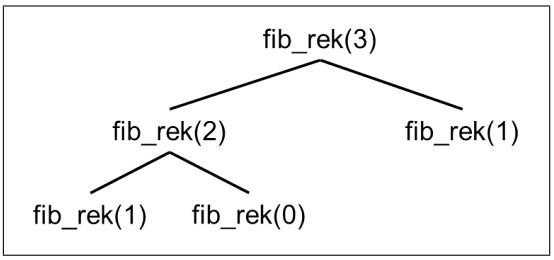

Abbildung 1: Baumstruktur für die Visualisierung der Prozedur-Aufrufe zur Berechnung des 3. Fibonacci-Elements