12.5/16

Betriebssysteme

Übungsblatt 7

Anne Rossl Diana Hörth

December 8, 2022

10/11

Aufgabe 1

-0.5 result wurde nicht allokiert

a)

```
1  int function ggt(int a, int b)
2  {
3    if(b == 0) {
4      result = a;
5    } else {
6      if (a = 0) {
7        result = b;
8      } else {
9        if (a < b){
11         result = ggt(a, b - a);
12       } else {
13         result = ggt(a - b, b);
14       }
15     }
16   }
17   return result;
18 }
19
20 void main()
21 {
22   int ggt_result;
23   ggt_result = ggt(16,12); //Rücksprungadresse 100
24 }
```

geht auch mit a == b
return a oder b;

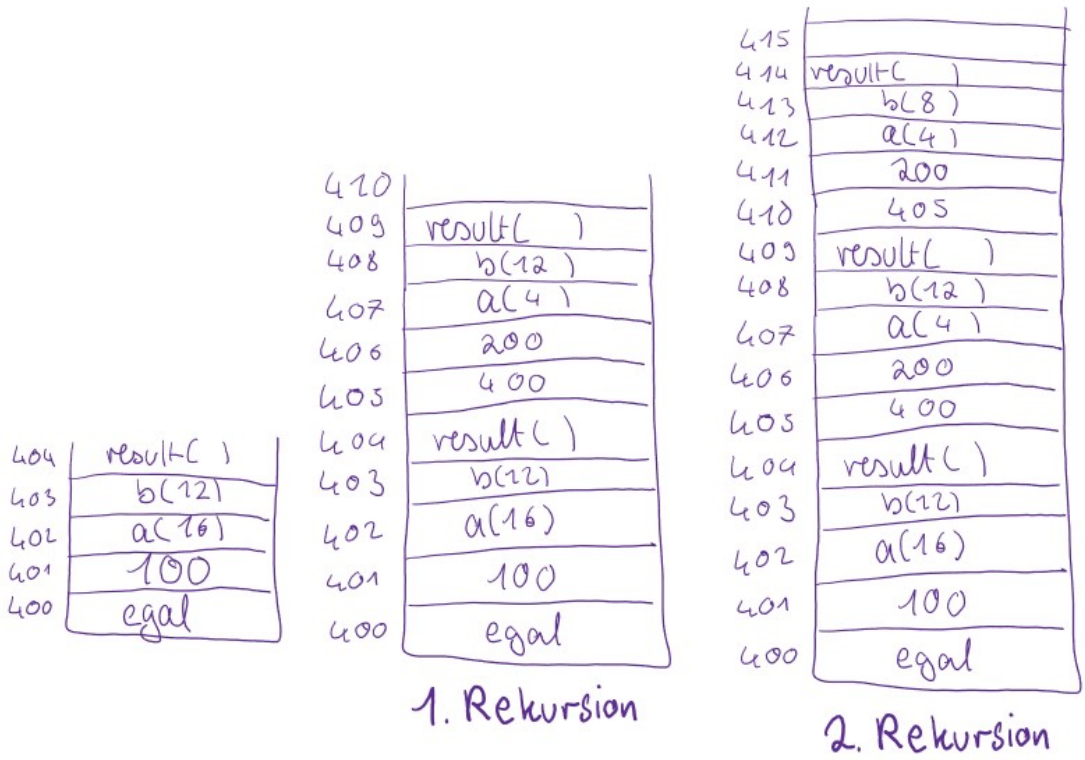
dann braucht ihr einen Funktionsaufruf weniger

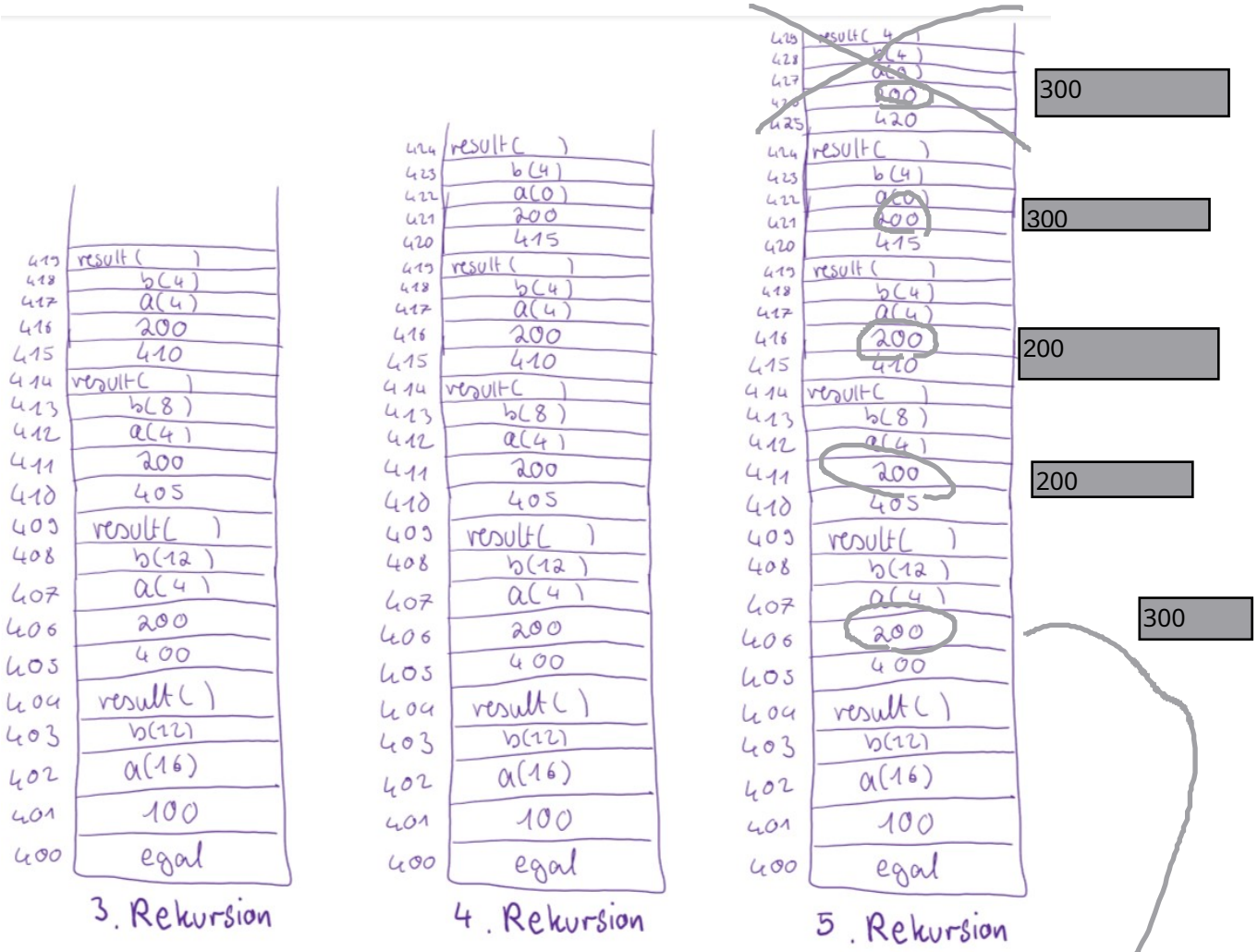
200

300

wäre gut Rücksprungaddressen anzugeben als Kommentar

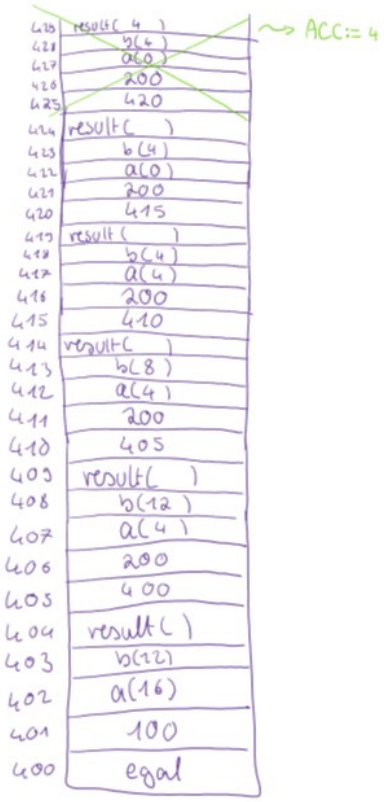
b)

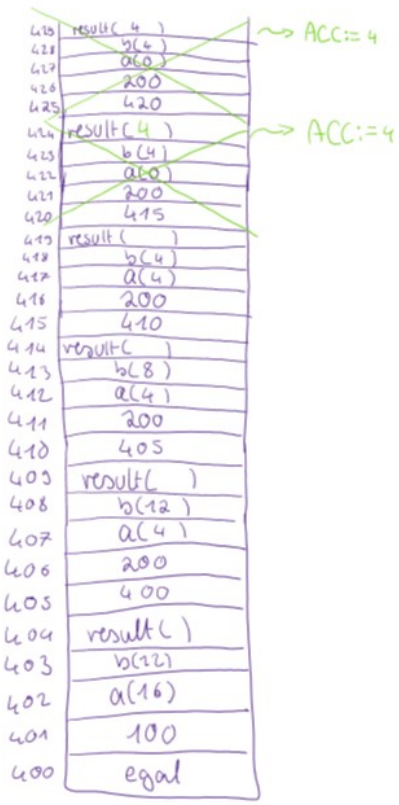

404 result( )
403 b(12)
402 a(16)
401 100
400 egal
410
409 result( )
408 b(12)
407 a(4)
406 200
405 400
404 result( )
403 b(12)
402 a(16)
401 100
400 egal
1. Rekursion
415
414 result( )
413 b(8)
412 a(4)
411 200
410 405
409 result( )
408 b(12)
407 a(4)
406 200
405 400
404 result( )
403 b(12)
402 a(16)
401 100
400 egal
2. Rekursion

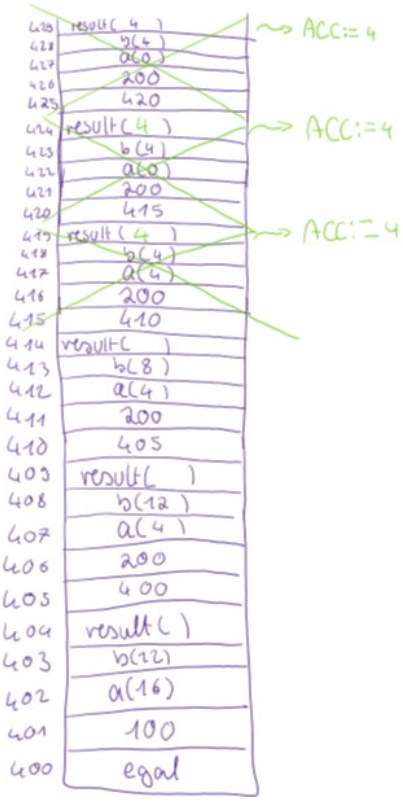


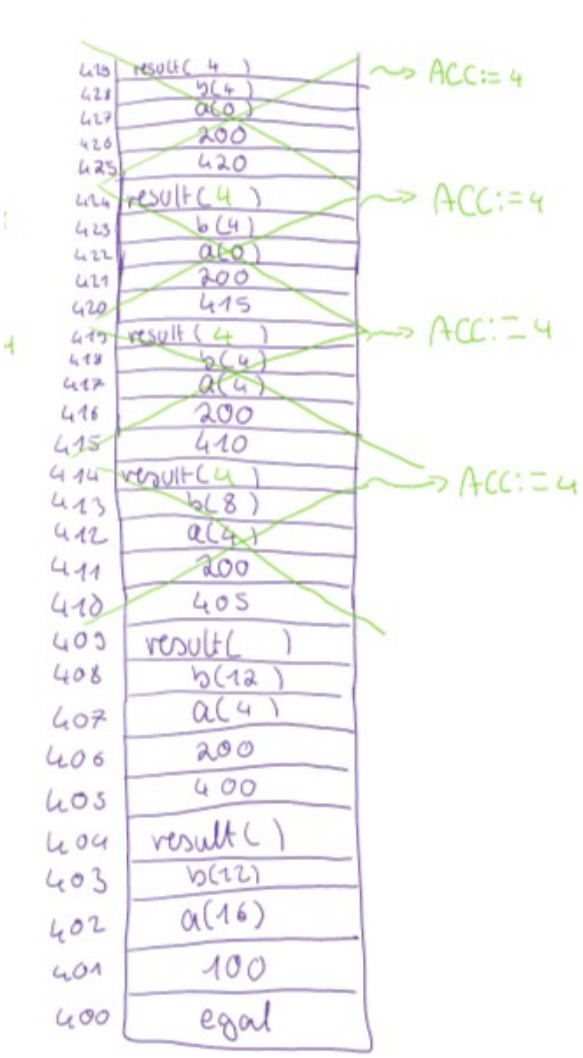
-0.5 Da ihr 2 mögliche Funktionsaufrufe habt, ist es allein deswegen schon ausgeschlossen, dass ihr immer die selbe Rücksprungadresse habt. Und ihr verwendet ja auch beide.

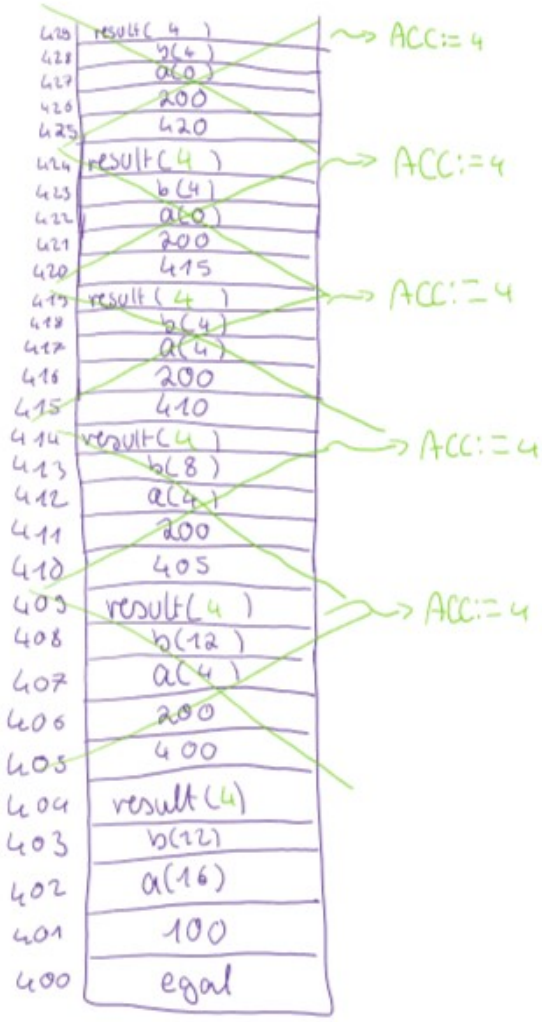
Rücksprungadressen beziehen sich auf die Speicheradressen im Codesegment, in dem die Maschinenbefehle in welche euer C Programm von einem Compiler übersetzt wird, gespeichert sind.


result(4)
b(4)
a(0)
200
420
result( )
b(4)
a(0)
200
415
result( )
b(4)
a(4)
200
410
result( )
b(8)
a(4)
200
405
result( )
b(12)
a(4)
200
400
result( )
b(12)
a(16)
100
egal
ACC:= 4


result(4)
b(4)
a(0)
200
420
result(4)
b(4)
a(0)
200
415
result( )
b(4)
a(4)
200
410
result( )
b(8)
a(4)
200
405
result( )
b(12)
a(4)
200
400
result( )
b(12)
a(16)
100
egal
ACC:= 4
ACC:= 4


result(4)
b(4)
a(0)
200
420
result(4)
b(4)
a(0)
200
415
result(4)
b(4)
a(4)
200
410
result( )
b(8)
a(4)
200
405
result( )
b(12)
a(4)
200
400
result( )
b(12)
a(16)
100
egal
ACC:= 4
ACC:= 4
ACC:= 4





1.

```
dh330@login1:/tmp$ ls -l ./werbinich
-rwxr-xr-x 1 dh330 uni 31480  7. Dez 16:08 ./werbinich
```

```
dh330@login1:/tmp$ ls -l /usr/bin/whoami
-rwxr-xr-x 1 root root 31480 16. Mai 2020  /usr/bin/whoami
```

Aufgabe 2

2.5/5

a)

0.5/1

- Typ des Eintrages: Verzeichnis
- Rechte:
  Die ak1062 ist der Besitzer des Verzeichnisses und er darf sie lesen, darin schreiben und sie ausführen.

-0.5 bei Verzeichnissen ist es:
ausführen = ins Verzeichnis wechseln
lesen = Verzeichnisinhalt auflisten
schreiben = Dateien erstellen, löschen usw.

manchmal ändert sich wegen umask auch die Zugriffsrechte

uni ist die Gruppe, die Zugriff hat und sie darf das Verzeichnis nur ausführen.
Alle können darauf zugriffen und haben sie dürfen ebenfalls das Verzeichnis nur ausführen.

2/2

b)

etwas unpräzisse für "Nutzer ändert sich"

aber passt

Zugriffszeit usw. auch

Bei ./werbinich ist die Gruppe, die Zugriff hat derjenige der eingeloggt ist und alle anderen, die Zugriff haben ist die uni.
Bei /usr/bin/whoami hingegen hat nur der root Zugriff auf die Datei und niemand sonst.

manchmal ändert sich wegen umask auch die Zugriffsrechte

2.

eigentlich waren noch konkrette Befehle verlangt, wenn das nich t so wäre, aber die Aufgabenstellung macht das nicht deutlich

Wir müssen nichts an den Rechten ändern, da schon alle die Datei ausführen dürfen.

Es wird der Name des Users angezeigt. In unserem Fall dh330.

3.

Mit dem Befehl "chmod u+s werbinich" gibt man denen die auf das Verzeichnis zugreifen die Rechte des Besitzers und "simuliert" sozusagen den Besitzer, wodurch bei der Ausführung dieser ausgegeben wird.

```
-rwsr-xr-x 1 dh330  uni   31480 Dec  7 16:08 werbinich
login1:/tmp$ ./werbinich
dh330
login1:/tmp$ cd
login1:~$ whoami
ar529
```

c)

0/2

1.

Mit dem Befehl "chmod 644 systeme-public" ändert man es so um, dass der Besitzer es verändern und lesen und die anderen zwei nur lesen können.

rw-r--r--

-1 falsche Rechte, es bräuchte ??????r-x (? = whatever), aber ihr habt rw-r--r---

```
login1:~$ chmod 644 systeme-public
login1:~$ ls -l
total 35
drwxr-xr-x 2 ar529 uni 25 Oct 25 17:42 newdir2
drw-r--r-- 2 ar529 uni  0 Dec  8 13:37 systeme-public
```

2.

Das Verzeichnis davor soll unsichtbar sein und dann verschickt man einen Link, der den der auf das Verzeichnis zugreifen will direkt dorthin führt.

ihr könnte immer noch durch das übergeordnete Verzeichnis durch, aber ihr könnt nicht mehr auflisten

```
login1:~$ ln -s /home/ar529/systeme-public shortcut
login1:~$ ls -l
total 59
drwxr-xr-x 2 ar529 uni 25 Oct 25 17:42 newdir2
lrwxrwxrwx 1 ar529 uni 26 Dec  8 15:02 shortcut -> /home/ar529/systeme-public
drw-r--r-- 2 ar529 uni  0 Dec  8 13:37 systeme-public
```

löst leider nicht wirlich die Aufgabenstellung. Der Überordner ~ (home/user/) soll Dateien nicht mehr auflisten können, aber immer noch navigierbar sein. Dafür gibt es passende Zugriffsrechte, die man für ~ setzen kann ^_^