

Betriebssysteme WS22/23

Blatt 6

Malte Pullich, Daniel Augustin

01.12.2022

Nummer 1

st(x) = (var, int, 128)
st(y) = (var, int, 129)
st(z) = (const, int, 2)

6+1 weil ihr eine der wenigen Gruppen seid die es korrekt gemacht haben = 7 /6 perfekt

```

1  ;y = 3
2  SUBI SP 1          ; Enlarging the SP
3  LOADI ACC 3         ; Loading 3 into ACC
4  STOREIN SP ACC 1    ; Saving 3 on SP
5  LOADIN SP ACC 1     ; Loading SP into ACC
6  ADDI SP 1           ; Deleting value 3 from SP
7  STORE ACC 129       ; Storing in place for var y
8  ;x = 15
9  SUBI SP 1           ;
10 LOADI ACC 15        ; Loading 15 into acc
11 STOREIN SP ACC 1    ; Saving 15 on SP
12 LOADIN SP ACC 1     ;
13 ADDI SP 1           ;
14 STORE ACC 128       ; Storing in place for var x
15 ;Saving value of x to SP
16 SUBI SP 1           ;
17 LOAD ACC 128        ; Loading value for x
18 STOREIN SP ACC 1    ; Saving value for x on SP
19 ;Saving value of z to SP
20 SUBI SP 1           ;
21 LOADI ACC 2         ; Loading 2 (const z) into ACC
22 STOREIN SP ACC 1    ; Saving 2 on SP
23 ;Saving value of y to SP
24 SUBI SP 1           ;
25 LOAD ACC 129        ; Loading value for y
26 STOREIN SP ACC 1    ; Saving value for y on SP
27 ;Checking loop condition
28 ;Computing z * y
29 LOADIN SP ACC 2     ; Loading value of z from SP into ACC
30 LOADIN SP IN2 1     ; Loading value of y from SP into IN2
31 MUL ACC IN2         ; Computing z * y
32 STOREIN SP ACC 2    ; Saving z * y on SP, overwriting z
33 ADDI SP 1           ; Deleting value of y from SP
34 ;Checking if x >= (z*y)
35 LOADIN SP ACC 2     ; Loading value of x from SP into ACC
36 LOADIN SP IN2 1     ; Loading the result of z * y into IN2
37 SUB ACC IN2         ; Computing x - z * y
38 JUMP< 3             ; If x - z * y < 0 Jump to saving

```

```

39      ; 0/FALSE as result
40  LOADI ACC 1      ; Saving 1 or TRUE as result
41  JUMP 2           ; Skip saving 0/FALSE as result
42  LOADI ACC 0      ; Saving 0 or FALSE as result
43  STOREIN SP ACC 2 ; Saving the result of the statement
44  ADDI SP 1        ; Cleaning the SP
45  ;Checking if the statement was correct
46  LOADIN SP ACC 1  ;
47  ADDI SP 1        ; Cleaning SP
48  JUMP= 15         ; If statement result = 0/FALSE
49                        ; jump to end of code
50  ;Code inside the loop
51  SUBI SP 1        ;
52  LOAD ACC 128     ; Loading the value of x
53  STOREIN SP ACC 1 ; Saving x on stack
54  SUBI SP 1        ;
55  LOADI ACC 3      ; Loading constant 3
56  STOREIN SP ACC 1 ; Saving const 3 on stack
57  LOADIN SP ACC 2  ; Loading value of x from stack
58  LOADIN SP IN2 1  ; Loading vlaue of 3 from stack
59  SUB ACC IN2      ; computing x - 3
60  STOREIN SP ACC 2 ; Saving the result on the SP of x
61  ADDI SP 1        ; Removing 3 from SP
62  LOADIN SP ACC 1  ; Loading value for x from SP
63  STORE ACC 128    ; Storing new value in x
64  JUMP -26         ; Jumping to the start of the Loop

```

jumpst nicht weit
 genug zurück, aber
 das ständig
 nachändern ist
 auch etwas zu viel

ich glaub ihr hattet davor an die richtige Stelle
 gezeigt und nur jetzt ist es nicht mehr der Fall

Nummer 3 a

```
a = &(p2.x);  
//Speichert die Adresse 15 in die Variable von a (Adresse 10)
```

```
p2.x = 7;  
// Speichert den Wert 7 in der Adresse 15
```

```
p2.y = 4;  
// Speichert den Wert 4 in der Adresse 16
```

```
p1 = (struct point *) malloc (sizeof (struct point));  
//Speichert die Adresse 33 in die Variable von p1 (Adresse 8)
```

```
(*p1).y = *a;  
Speichert den Wertes aus a (Wert 7) in die Adresse 34
```

```
p3 = p1;  
Speichert die Adresse 33 aus p1 in Variable p3
```

```
p1 = &p2;  
Speichert die Adresse 15 von p2 in die Variable p1
```

```
if ((*p1).y > 5)  
Auswerten ob der an der Adresse 16 (Wert 4) > 5 ist nur der else teil relevant
```

```
a = 1;  
Speichert den Werts 1 in die von Variable a referenzierte Adresse 15
```

```
free(p3);  
Übergabe der in der Variable p3 referenzierten Adresse 33 an free()
```

die b) beantwortet ja iwie auch schon
direkt die a)

Nummer 3 b

Marke 1

...	
4	16 (p2.y)
7	15 (p2.x)
...	
15	10 (a)
...	

10/10

Marke 2

...	
7	34 (p3.y)
unknown	33 (p3.x)
..	
4	16 (p2.y)
7	15 (p2.x)
...	
15	10 (a)
33	9 (p3)
15	8 (p1)
...	

Marke 3

...	
7	34 (p3.y)
unknown	33 (p3.x)
..	
4	16 (p2.y)
1	15 (p2.x)
...	
15	10 (a)
33	9 (p3)
15	8 (p1)
...	

Nummer 3 c

Die letzte Anfrage $\text{free}(p3)$ ist zulässig und die Adressen 33 und 34 werden freigegeben