15.5 Sehr gute Arbeit! n_n

Betriebssysteme

Übungsblatt 5

Anne Rossl Diana Hörth

November 24, 2022

5/6

Aufgabe 1

alle Compute und Load Befehle. Wenn es alle, sind, dann ist es einfach 1, da a + ¬a = 1 und a * 1 = a

a) - SPcken$_{pre}$

SPcken$_{pre}$ =[E * $\overline{s_1}$ * s_0+ //Von P1 bis P2 in der Executephase
$h_2 * h_1 * h_0$+ // h_7 oder
$\overline{h_2} * h_1 * \overline{h_0}$+ // h_2 oder
$\overline{h_2} * h_1 * h_0$+ //$h_3$ oder
$h_2 * \overline{h_1} * h_0$]+ //$h_5$ oder
$NB * [\overline{I_{31}} * \overline{I_{30}} * I_{24} * \overline{I_{23}} * I_{22}$+ // Compute mit D = SP
$\overline{I_{31}} * \overline{I_{30}} * I_{28} * I_{21} * \overline{I_{20}} * \overline{I_{19}}$+ //Compute (register only) mit S = SP
$\overline{I_{31}} * I_{30} * I_{24} * \overline{I_{23}} * \overline{I_{22}}$+ //LOAD mit D = SP
$\overline{I_{31}} * I_{30} * \overline{I_{29}} * I_{28} * I_{27} * \overline{I_{26}} * \overline{I_{25}}$+ //LOADIN mit S = SP
$I_{31} * \overline{I_{30}} * I_{29} * I_{28} * \overline{I_{24}} * I_{23} * I_{22}]$ //MOVE mit D = SP

Es geht nur um SP als destination, weil dann die clk vom dem Regsiter aktiviert werden muss

es geht um Destination

-0.5 weil Folgefehler

nur MOVE weil es um SP als Destination geht

genau anders rum

b) - IVNcken$_{pre}$

$IVNcken_{pre} = [E * s_1 * s_0] + [\overline{h_2} * \overline{h_1} * h_0] + [h_2 * \overline{h_1} * \overline{h_0}]$ //In der Executephase in P3 und $h_1 oder h_4$

Int i vergessesen -0.5

same here

kein Punktabzug, da Folgefehler

Aufgabe 2

a)

5.5/8

st(x) = (var, int , 128)
st(y) = (var, int, 129)
st(z) = (const, int, 5)

b)

das auf den Stack schreiben gehört zur Aufgabe -0.5

nur das Innitialisieren der Variablen darf man bei dieser Aufgabe auslassen

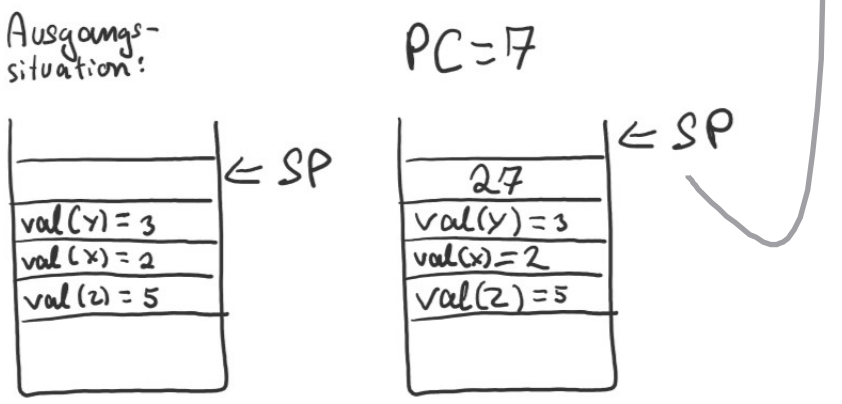
PC	Befehl	Kommentar
0	LOADIN SP ACC 3	Lade z in ACC
1	LOADIN SP IN1 1	Lade y in IN1
2	MUL ACC IN1	Multipliziere z und y
3	ADDI ACC 10	Addiere 10 noch dazu
4	LOADIN SP IN2 2	Lade x in IN2
5	ADD ACC IN2	Addiere das Ergebnis der Multiplkation mit x
6	SUBI SP 1	Stackpointer um eins nach oben verschieben
7	STOREIN SP ACC 1	Ergebnis auf Stack legen

3.5/4

SUBI SP 1

die Aufgabe war so gedacht, dass man die Patterns aus der Vorlesung verwendet, aber es geht auch ohne

Wenn man genug Register zu Hand hat, geht es mit viel weniger Zeilen ^_^



c)

0/2

1.

-1

2.

In den ersten Klammern, die berechnet werden müssen immer zwei Element drinnen sein und danach müssen die jeweiligen Teilergebnisse miteinander verrechnet werden. Es sind n-1 Teilergebnisse.
z.B.:
n = 7
$((x_1 \circ x_2) \circ (x_3 \circ x_4)) \circ ((x_5 \circ x_6) \circ x_7)$

minimal sind 2, das werden allerdings 3 >_<

2

-1

5/6
sehr gute Arbeit ^_^

Aufgabe 3

PC	Befehl	Kommentar
13	LOAD ACC 11	Lade y in ACC
14	JUMP$_<$ 6	Schaue ob y kleiner ist als 0
15	LOAD ACC 10	Wenn y größer als 0, lade x in ACC
16	JUMP$_<$ 11	Schau ob x kleiner ist als 0
17	SUB ACC 11	Wenn y und x größer sind als 0, x-y
18	JUMP$_\leq$ 9	Wenn x-y kleiner oder gleich ist wird 1 bei 12 eingespeichert
19	JUMP 6	Ansonsten wird 0 bei 12 eingespeichert da x größer ist als y
20	LOAD ACC 10	Wenn y kleiner als 0, wird x in ACC geladen
21	JUMP$_\geq$ 4	Wenn x größer als 0 ist, ist es auch größer als y also wird 0 in 12 eingespeichert
22	LOAD ACC 11	Wenn sowohl y als auch x kleiner als 0 sind wird y-x gerechnet
23	SUB ACC 10	Ist das Ergabnis größer 0 ist x größer als y.
24	JUMP$_\leq$ 3	1 wird in 12 gespeichert, da y größer/ gleich x
25	STORE 12 0	0 wird gespeichert wenn die Aussage falsch ist
26	JUMP 0	Programm wird beendet, damit der Speicher 12 nicht überschrieben wird
27	STORE 12 1	1 wird gespeichert, wenn die Aussage wahr ist

Ich hab den RETI-Code aller Studenten die Aufgabe 3 bearbeitet haben (Aufgabe 3 war diesmal die beliebteste Aufgabe) mithilfe des im PicoC-Compilers https://github.com/matthejue/PicoC-Compiler/releases eingebauten RETI-Interpreters ausgeführt, genauer mittels des Befehls `picoc_compiler -b -p c.reti -S -P 2 -D 15`. Ich habe versucht den Code von euch Studenten lauffähig zu machen, sodass dieser die Aufgabenstellung erfüllt.
Die Datei <fruit>.in enthält Eingaben für CALL INPUT REG. Die Datei <fruit>.out enthält die Ausgaben der CALL PRINT REG bei der Ausführung. Die Datei <fruit>.out_expected enthält die erwarteten Ausgaben.
Eure Korrektur ist unter https://github.com/matthejue/Abgaben_Blatt_3/tree/main/Blatt5/orange.reti zu finden.